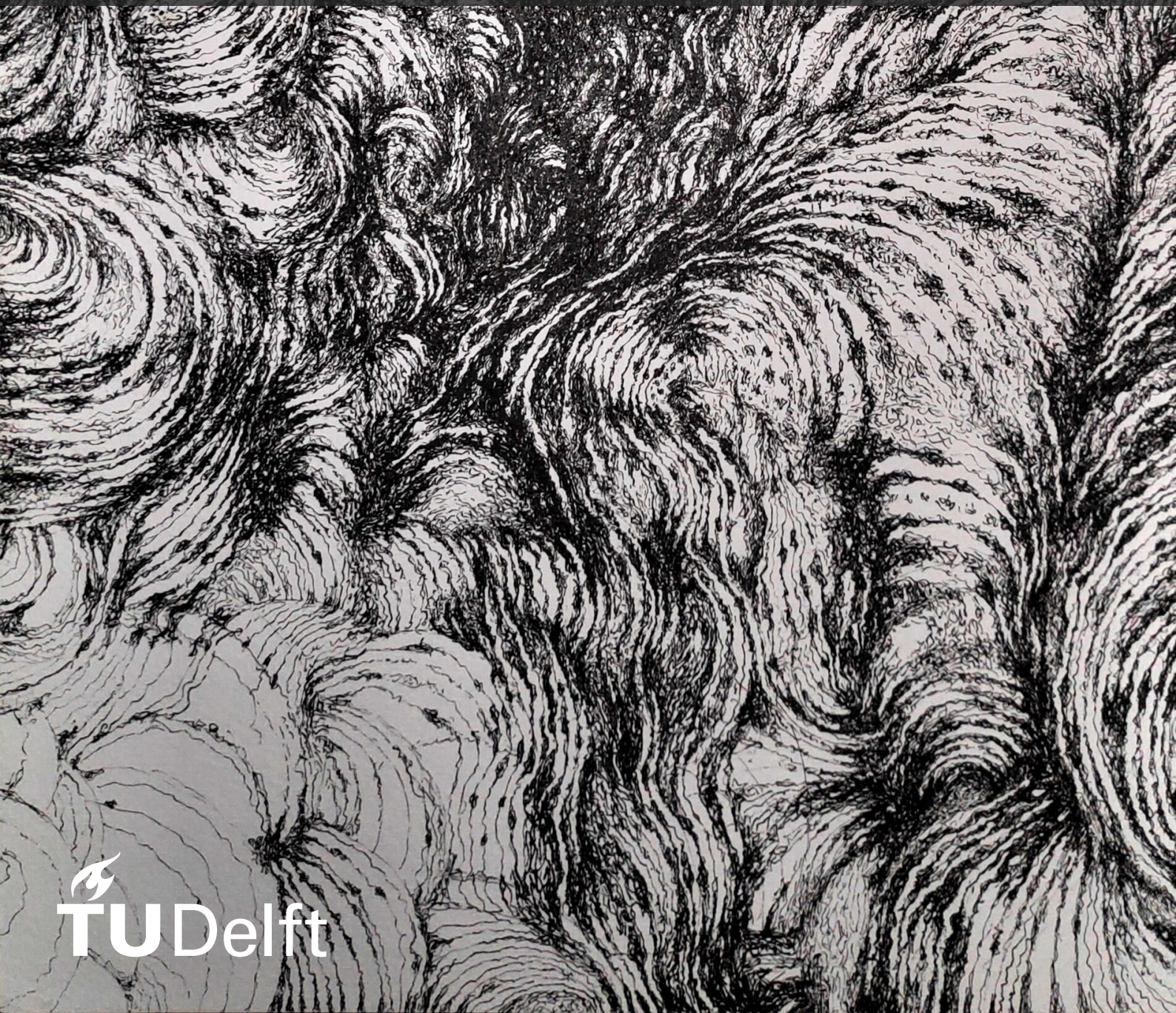
Nanosatellite Testing for Small Missions

A low-cost, no-nonsense approach to magnetic testing for nanosatellites

MSc Thesis Aerospace Engineering

Johannes Jacobus Monster



TU Delft

Nanosatellite Testing for Small Missions

A low-cost, no-nonsense approach to magnetic testing for nanosatellites

by

Johannes Jacobus Monster

Student number
4168755

A thesis presented for the degree of

Master of Science
Aerospace Engineering

Thesis Supervisor:	S. Speretta
	M.S. Uludag
Project Duration:	September, 2023 - December, 2025
Faculty:	Faculty of Aerospace Engineering, Delft

Cover:	*Gleaming 1* by Ivan Avaliani [1]
Style:	TU Delft Report Style 1.7 (Modified)

TU Delft

Preface

Outer space is an environment that seldom forgives mistakes. It is the most relentless environment into which man has ever ventured. It is an engineer's nightmare, a hell in heaven for the foolhardy and the stupid. Although to enter, live, and work there demands perfection in every detail, it is in man's nature to make mistakes, and therein lies the rub.
- from *"Orbital Decay"* by Allen M. Steele

Thou, who canst guide the wandering star
Through trackless realms of æther's space;
Who calm'st the elemental war,
Whose hand from pole to pole I trace.
- from *"The Prayer of Nature"* by George Gordon Byron

No battle plan survives first contact with the enemy.
- Helmuth von Moltke

This thesis was nearly titled *"An Unfortunate Series of Encounters with the Laws of Ohm, Lenz, and Murphy"*. I settled on something more prosaic in the end. Yet as I prepare to hand in this thesis, I can confidently claim intimate familiarity with all three.

In the field of engineering, one is frequently confronted with the laws that govern nature. We may create designs that look marvellous on paper, only to then find that the universe has many unexpected ways to dash our hopes. I find that engineering is as much about ingenuity and creative ideas as it is about having the patience and persistence to work through all the unforeseen problems along the way.

However, seeing this process through to the end will bring with it one of the finest experiences that life has to offer: to see one's own ideas made manifest, and to have them accepted by the laws that govern the universe. To me, this is what makes engineering so profoundly gratifying. It is a pure form of creation; an art of both function *and* form.

What you have in front of you is the product of a lengthy trial to formally become an engineer. The contents of this thesis cover most aspects of an engineering design process: devising solutions to identified problems, and then manifesting the best solutions into a tangible and working implementation. There is electronics design, systems engineering, software development, hardware integration, testing, and more; few stones remained unturned.

As such, it was a multifaceted project for which I could leverage my personal interests and skill set, both of which are notoriously broad. It was also an uncomfortably lengthy and difficult project which came at considerable personal expense. For both reasons, I am glad to be drawing it to its conclusion.

Of course this was no lonesome road, there were many people whose help I could rely on. In particular I would like to thank my supervisor, Stefano Speretta, for his guidance, patience, and keen insights. Furthermore, I am grateful for my fellow engineers, my friends, my family, and especially my parents, for their support, patience, and love. And finally, I thank the Creator of it all, for all of it.

Johannes Jacobus Monster
Delft, November 2025

Reader's Guide

During the research covered in this thesis, a magnetic testing system for nanosatellites was designed. Several different topics come together to achieve this, such as systems engineering, software design, signal processing, and electronics design. This thesis is written in such a way that there is a narrative arc spanning from beginning to end; a common thread that ties all these topics together.

This thesis was made with two target audiences in mind. First, it serves as a blueprint for those who wish to reproduce this test system design. These may be fellow engineers who are interested in methodology and design rationale, but also implementation details and other resources for reproduction. For this audience, the thesis was made to be as comprehensive as possible, which explains why it has become such a large document.

However, this thesis is also meant to be read and graded by academic staff, whose schedules tend to be busy. In order to facilitate their work, there is a suggested reader's guide if you are grading the thesis.

Section	Page count	Content
Chapter 2 - 4	20	Problem statement, research framework, satellite testing
Section 6.3	1	Problems with old Helmholtz cage system and related design implications
Section 7.5, 7.7	6	Test results and conclusions of ambient magnetic field measurements
Chapter 8, 9	17	Requirements and system design of the Demonstrator System
Chapter 10	44	Electronic design
Sections 11.1 - 11.5, 11.7	24	Problem statement, research framework, satellite testing
Chapter 12 - 14	32	System integration, verification, validation, thesis conclusion
	144	

Chapter 1 is a general introduction to the thesis structure and content and contains little unique information. Chapter 5 is contextual information about the theory of operation of a Helmholtz cage. Chapters 6 and 7 contain much information about methodology which are important to their content, but are less critical for the conclusions of the thesis as a whole, which is why it is recommended to focus on the concluding sections of these chapters. Section 11.6 is on the layout of a communication protocol that overall has lesser impact on the thesis conclusions.

Of course, if you are not reading this thesis for grading, you may read and skip as you wish. To this end, the table of contents and chapter 1 will help you find your way.

Statement on the use of AI tools

This thesis was written during a rapid emergence of easily accessible generative tools based on machine learning. Such tools, commonly referred to with the designation "Artificial Intelligence" (AI) tools, are capable of generating text, software code, images, and more with inhuman speed. What liberties and restrictions are ultimately appropriate for such tools in the context of education is a discussion that is still on-going, and likely will remain so for the foreseeable future. In this context, the author considers it relevant to declare in which ways AI tools were used during the creation of this thesis.

The author declares that during the creation of this thesis work, **no machine learning tools or any other generative AI tools were used** in any way whatsoever. This includes but is not limited to the creation of imagery, reported content, summaries, software code, etcetera. All thesis work was done by the author himself. In this regard, this thesis may be among the last of its kind.

Executive Summary

For a number of decades, CubeSats and other nanosatellites have been used more and more for space applications. Decreasing launch prices as well as the small mass and size of these satellites make such missions relatively cheap, and they are therefore popular with universities, research institutes, and even wealthy hobbyists. However, despite their popularity, nanosatellite missions still have a relatively high failure rate, particularly for missions with a constrained schedules and budgets[1].

There are divided opinions about what the root causes of this high failure rate might be. In this thesis, hardware-related root causes were investigated, and we conclude that at least part of the high failure rates can be explained by insufficient satellite testing. This does not necessarily indicate incompetence or poor testing procedure, but it seems that sometimes satellite tests are skipped because there is no time or budget for them. There have been several problematic nanosatellite missions where the problems could have been prevented by better testing[2].

For several reasons, among them being the results of the literature review, it was decided to further examine magnetic testing of nanosatellites. If it is true that developers of nanosatellites sometimes choose to skip certain magnetic tests, then the failure rate of these missions could be reduced simply by offering a more convincing, affordable, and approachable testing solution. If such a testing system design is available for all to use, then critical magnetic satellite tests might be performed where they otherwise may have been skipped completely.

This is the reasoning upon which this thesis is predicated. The research question underpinning the thesis work was formulated as follows:

"How can the magnetic properties of a small satellite be effectively tested?"

Within this research question, four main areas were to be investigated: the degree to which the testing is **representative** of the mission, the **reliability** of the testing, the **time-efficiency** of the testing, and finally its **cost-effectiveness**. The objective of the research was formulated as:

"To improve testing for CubeSat missions, especially those heavily constrained by schedule and budget, by improving and streamlining a testing strategy for the magnetic properties of a CubeSat."

In other words: if a magnetic testing system can be designed that is very affordable, easy to build and use, but can still perform effective magnetic testing of a nanosatellite, this can be a powerful argument in favour of performing such tests, even if a mission has a tightly constrained budget. The cost target for this system was set to €12,500 or less, whilst covering the necessary magnetic testing as best as possible.

What kind of magnetic testing would you want to perform on a nanosatellite before its mission starts? Based on documents such as the ECSS standards, an itinerary of six tests was identified:

1. Characterization of the satellite's net magnetic dipole
2. Characterization of the spacecraft's near-field properties
3. Characterization of the spacecraft resistance to AC emissions
4. Verification of any magnetic sensing equipment on the satellite
5. Verification of any magnetic actuators on the satellite
6. General operation validation

We can envisage an ideal magnetic testing system that could perform all these tests. However, for several reasons, it is not feasible to make such a system: it would be cost-prohibitive and too complex to build as part of a tightly constrained mission, and it certainly would be too much to design as part of a

[1] In the PDF version of this thesis, clicking will take you to the relevant section in the main body of the thesis.
[2] see for example PicSat[2], CANYVAL-X[3], DeOrbitSail[4], and MOVE-II[5][6]

single MSc thesis. Instead, this thesis focuses its efforts on building a so-called **Demonstrator System** (DS), which is a simpler version of the idealized testing system. It is depicted in Figure 1, and comprises a Helmholtz Cage, its driver electronics, magnetic sensing equipment and a testing/simulation computer interface. The idea is that this is a minimal testing system setup that can be adjusted and built easily by others, but that can still cover a significant portion of the aforementioned itinerary of magnetic tests. If the mission budget allows, the Demonstrator System could be expanded further to have better testing coverage, and indeed it was designed to facilitate such expansions.

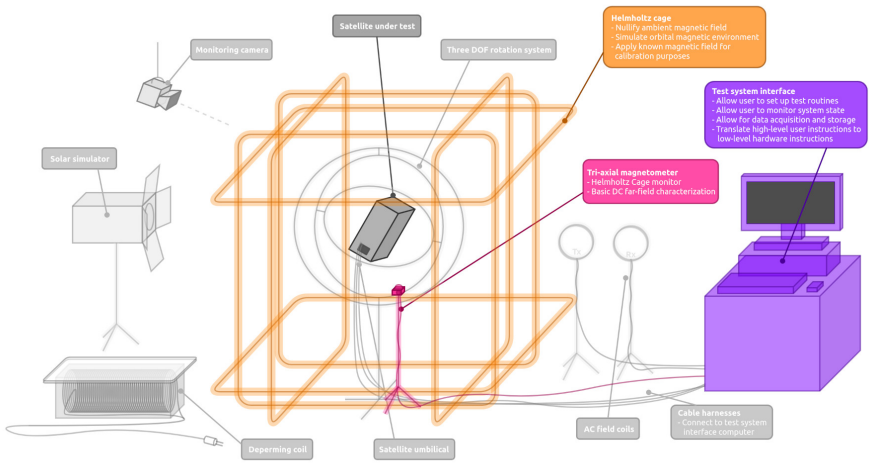


Figure 1: An overview of the Demonstrator System (DS), showing implemented system segments. Unimplemented segments are greyed out, but are implicitly supported by the design as future expansions.

Before the design of this Demonstrator System could start, a complete set of system requirements was needed. To formulate these requirements, two additional steps had to be taken: first, there was already an older Helmholtz cage system available at the university faculty, whose parts could be useful to the DS design. This system was disassembled and its theoretical performance was evaluated. It turned out that the coil structure could be used for the DS, but that there were a number of problems with the older system that warranted a complete redesign of the software and the driver electronics.

In addition, the location where the DS was to be built was investigated for local magnetic disturbances. After all, it would be terrible to build a highly capable system that turns out to be fundamentally hamstrung by local noise sources. After investigating a number of potential noise sources, it turned out that most of the magnetic field disturbance consists of a static vector, something that can be easily nullified by a Helmholtz cage. There are are also much smaller dynamic components of this noise in the order of several hundred nanotesla, primarily caused by three nearby building elevators. Most of the noise can be avoided or mitigated, for example by testing during the evenings and by post-processing the test data with digital filters.

With this information in hand, the system requirements of the DS could be formulated DS, and proper design could commence. To keep system cost low and availability high, commercially-off-the-shelf (COTS) components were used wherever possible; for example, three off-the-shelf programmable current supplies were used to drive the three coil pairs inside the Helmholtz cage, and these were controlled with analog control signals to increase system compatibility. That way, if anyone else wishes to replicate the DS for their purposes, but they cannot acquire the exact same model of power supplies, there should be a large set of possible drop-in replacements. In addition, a common-place Raspberry Pi 4 and off-the-shelf DAC/ADC board were used to cheaply and flexibly handle system interaction. Similarly motivated design decisions were made all throughout the design process.

However, it quickly turned out that some custom parts were unavoidable: in order to make the design open to future expansion and enable polarity switching of the power supply current, some additional

electronic functions were needed. Therefore, a **companion board** was designed and implemented on a custom printed circuit board (PCB). The completed and integrated companion board can be seen on the right of Figure 2.

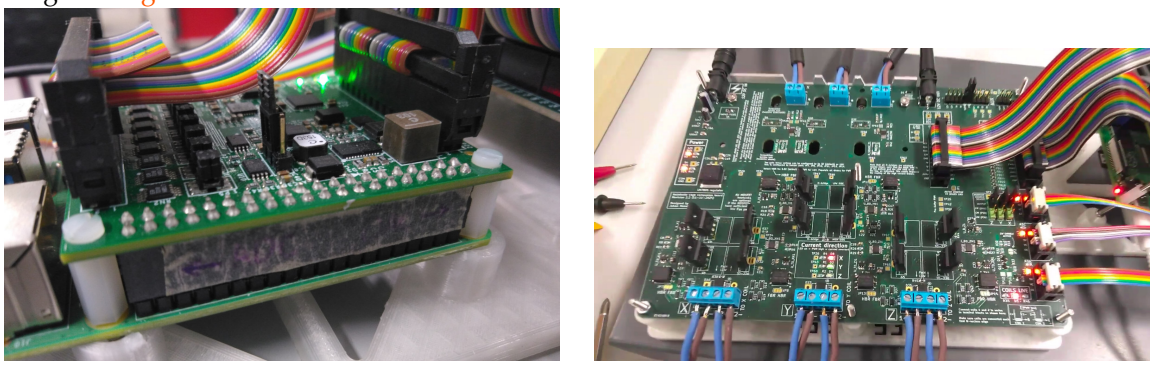


Figure 2: Left: A Raspberry Pi 4 with a stacked DAC/ADC board handles most of the hardware interactions in the system. Right: The assembled companion board interconnects several components and implements critical system functionality.

In order to make the system easily usable and attractive to potential satellite developers, it was important that the software supporting the DS adhered to several requirements. As part of the research, we are interested in tests that are **representative** of the satellite mission. So, it was not enough to simply make the system work with some back-end software, but to proactively offer the users tools to design test that can be representative to their particular mission. In addition, it is important for the system to be operated safely by users and to facilitate future expansion.

To this end, an open-source software package called the **Helmholtz Cage Toolkit** (HCT) was developed. This package is meant to contain all the software relevant to the system in a single bundle, so it can be distributed easily. It contains drivers for the more low-level hardware interfacing, but also higher level software functions that can be interfaced with by users. It even features a graphical user interface (GUI) application that very intuitively allows users to interact with the system and support their testing activities. An important task of the software is to translate the wishes of the human users into machine instructions for the system. We call these machine instructions **schedules**.

The Helmholtz Cage Toolkit contains several software tools to generate these schedules. Most importantly, a tool called **Orbital Generator** allows users to specify an orbital scenario, propagate a satellite along the orbit, and extract the dynamic magnetic field envelope that the satellite would experience in its own body frame. This is a highly laborious task to do by hand, but the Orbital Generator makes this very accessible, quick, and intuitive to do (see Figure 3). Detailed visualization tools allow users to spot their mistakes easily and to quickly learn to use the software without risk.

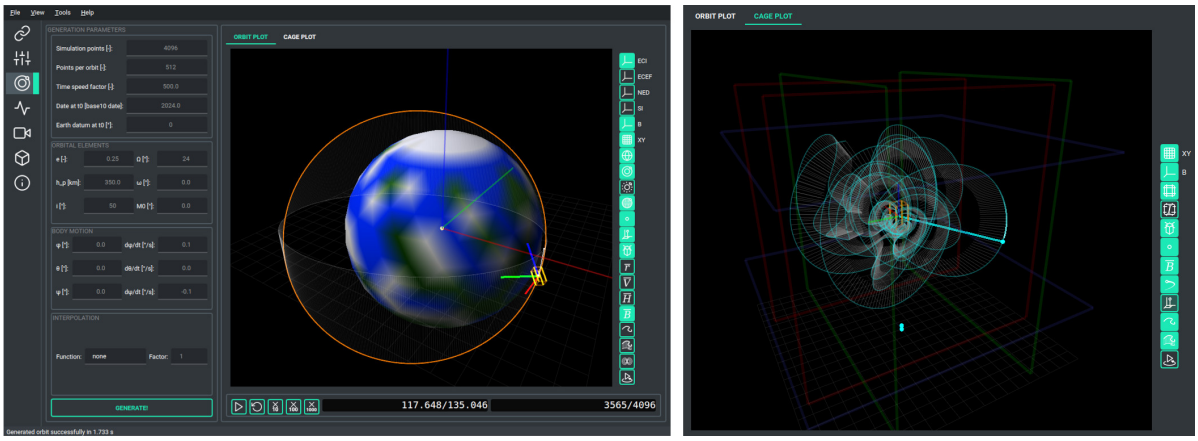


Figure 3: Overview of the Orbital generator, with the orbit design GUI on the left and the generated magnetic field envelope in the satellite's body frame on the right.

The generated magnetic field envelope can be saved and exported as a **schedule file**, at which point it is immediately able to be played back by the system. After assembly, calibration, and validation, the Demonstrator System performs quite well at this task. It is possible to simulate the magnetic field experienced by a satellite in an Earth orbit with a tracking error within a few degrees (see Figure 4). The system also passes all its design requirements. There are a few remaining issues that hold back system performance, but if these are overcome in the future it is feasible that tracking performance may yet be improved by a factor of two or more.

With the current system capabilities, what portion of the aforementioned test itinerary does the system currently support?

1. **Characterization of the satellite's net magnetic dipole**: **NO**, currently unsupported, straightforward to add as future upgrade.
2. **Characterization of the spacecraft's near-field properties**: **YES but** with limited support, depends on exact satellite configuration.
3. **Characterization of the spacecraft resistance to AC emissions**: **NO**, currently unsupported. Possible with additional hardware.
4. **Verification of any magnetic sensing equipment on the satellite**: **YES**, excellent support, both for basic calibration and testing, but also for mission-specific validation.
5. **Verification of any magnetic actuators on the satellite**: **YES**, decent support. Future expansions can further improve this.
6. **General operation validation**: **YES**, good support for ADCS, limited support for other subsystems. Future expansions can further improve this.

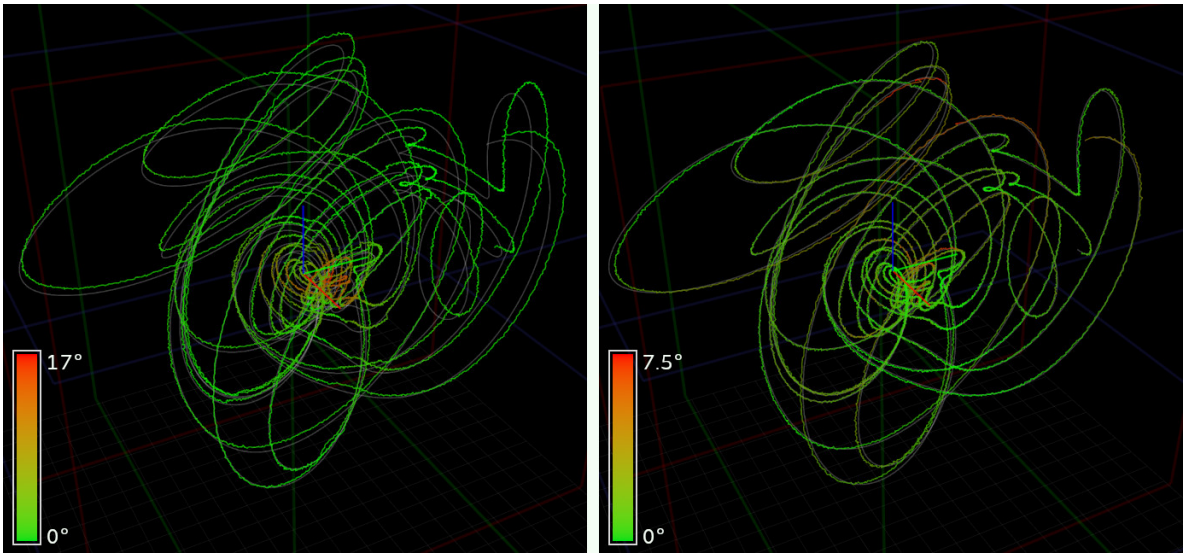


Figure 4: Left: Measured magnetic field envelope (coloured) and commanded envelope (gray). The colour indicates the angular tracking error in degrees, which is a few degrees on average, but can assume higher values near the origin of the Helmholtz cage.
Right: By addressing a few more issues, the performance can be improved at least twofold, resulting in better tracking.

Both the software as well as the hardware is open-source and licensed under a relatively permissive GPL license. Together with the information and blueprints contained in this thesis, the author hopes that this gives satellite developers a head-start with their testing efforts, and that particularly missions with a tight budget will still be tested properly through the use of the offered tools and designs. If it proves its worth, the HCT may even become a more established toolkit, developed through voluntary development and collaboration.

Contents

Preface ii

Reader's Guide iii

Statement on the use of AI tools iv

Executive Summary v

Nomenclature xxv

1 Introduction 1

I THE WHY - Problems, Root Causes, and Opportunities 3

2 On Satellite Failure 4
- 2.1 The Search for Root Causes . . . 6
  - 2.1.1 Bottom-up Approach . . . 6
  - 2.1.2 Top-down Approach . . . 9
- 2.2 Conclusion . . . 11

3 Research Goals and Scope 13
- 3.1 Motivation and choice of topic . . . 13
- 3.2 Thesis Objectives . . . 14
  - 3.2.1 Research Question . . . 14
  - 3.2.2 Research Objectives . . . 15
- 3.3 Research plan . . . 15

4 Magnetic Testing of Nanosatellites 17
- 4.1 Magnetic testing for CubeSats in literature . . . 17
- 4.2 An itinerary of magnetic testing for nanosatellites . . . 18
- 4.3 An idealized magnetic testing system for nanosatellites . . . 21
- 4.4 Requirements for the idealized system . . . 22
- 4.5 Conclusion to Part I . . . 24

II THE HOW - Physics, Requirements, and the Past 25

5 Helmholtz Cage Theory 26
- 5.1 Fundamentals . . . 26
- 5.2 Coil properties . . . 28
  - 5.2.1 Inductance . . . 28
  - 5.2.2 Magnetic field strength . . . 29
- 5.3 The Helmholtz Cage . . . 30

6 Overview of the old Helmholtz cage system 31
- 6.1 Overview of the old Helmholtz Cage system . . . 31
  - 6.1.1 Coordinate system . . . 32
  - 6.1.2 Internal system layout . . . 33
  - 6.1.3 H-Bridge circuitry . . . 36
  - 6.1.4 Switching board schematic . . . 38
  - 6.1.5 A brief specification sheet . . . 38
- 6.2 Theoretical Performance . . . 39
  - 6.2.1 Coil resistance . . . 39

6.2.2 Helmholtz cage static performance . . . 39
6.2.3 Helmholtz cage dynamic effects . . . 41
6.2.4 Helmholtz cage field homogeneity . . . 44
6.3 Identified Problems and Limitations . . . 47

7 Characterization of Ambient Magnetic Fields at the Test Location 48
7.1 Motivation . . . 48
7.2 Overview and Expectations . . . 49
7.3 Test Equipment . . . 52
7.4 Overview of tests . . . 54
7.5 Test results . . . 55
7.6 Noise rejection . . . 59
7.7 Conclusions . . . 61
7.8 Open questions and further research . . . 62

8 Demonstrator System Requirements 63
8.1 Demonstrator System Requirements . . . 64
8.1.1 Top-level Requirements . . . 64
8.1.2 Performance Requirements . . . 65
8.1.3 Electronic and Hardware Requirements . . . 66
8.1.4 Software Requirements . . . 67
8.2 Optimization and other considerations . . . 68
8.3 Conclusion to Part II . . . 71

III THE WHAT - Designs, Results, and Conclusions 72

9 Demonstrator System Layout 73
9.1 Hardware overview . . . 74
9.2 Degrees of separation . . . 75
9.3 Low-level hardware interface selection . . . 76
9.3.1 Analog and digital conversion . . . 77
9.3.2 Communication between low-level and high-level interfaces . . . 78
9.4 Companion board . . . 78

10 Electronic Design 80
10.1 Polarity Switching . . . 80
10.1.1 Polarity switching with transistors . . . 81
10.1.2 Preventing shoot-through by design . . . 83
10.1.3 MOSFET complications . . . 86
10.1.4 Integrated bridge drivers . . . 87
10.1.5 H-bridge design with the LTC7060 . . . 89
10.1.6 H-bridge design with the LTC7060 - Revisited . . . 91
10.1.7 MOSFET selection . . . 94
10.2 Current sense design . . . 96
10.2.1 Current sensing - First iteration . . . 97
10.2.2 Current sensing - Second iteration . . . 98
10.3 Thwarting inductive voltage transients . . . 103
10.3.1 Dropper diodes . . . 103
10.3.2 MOSFET body diodes . . . 104
10.3.3 TVS diodes . . . 104
10.3.4 Bleeder resistors . . . 107
10.4 Companion Board - First Iteration . . . 108
10.4.1 Implementation Functions . . . 108
10.4.2 Circuit Overview . . . 110
10.4.3 DAC as driver . . . 112
10.5 Issues and Design Improvements . . . 112
10.5.1 Issue: DAC incapable of driving LTC7060 PWM input . . . 113

10.5.2 Issue: 1685B incapable of driving inductive load . . . 113
10.5.3 Generalized output inhibition . . . 115
10.6 Companion Board - Second Iteration . . . 117
10.6.1 Functionalities of the companion board . . . 117
10.6.2 Circuit Overview . . . 120
10.7 Conclusion . . . 123

11 Software Development **124**
11.1 Role of Software . . . 124
11.2 High-level interface software . . . 125
11.2.1 Requirements and design philosophy . . . 126
11.3 Development . . . 127
11.4 Current design and feature set . . . 128
11.4.1 General overview . . . 129
11.4.2 Connection Window . . . 130
11.4.3 Command Window . . . 131
11.4.4 Cyclics Generator . . . 134
11.4.5 Orbital Generator . . . 136
11.4.6 Schedules . . . 140
11.5 Low-level interface software . . . 142
11.6 TCP communication protocol . . . 144
11.7 Software testing strategies . . . 147
11.8 General file layout . . . 149

12 System Integration **151**

13 Verification & Validation **156**
13.1 Top-level Requirements . . . 156
13.2 Performance Requirements . . . 160
13.3 Electronic Requirements . . . 161
13.4 Software Requirements . . . 164
13.5 Summarized Verification Matrix . . . 166
13.6 System Validation . . . 167
13.6.1 Assessing transfer characteristics . . . 167
13.6.2 Simulation of orbital motion . . . 171
13.6.3 Improving system performance . . . 176
13.6.4 Concluding the validation . . . 179

14 Conclusion **180**
14.1 The research question revisited . . . 180
14.2 Design Reflections and Lessons Learned . . . 183
14.2.1 Lessons in System Engineering . . . 183
14.2.2 Lessons in Electrical Engineering . . . 183
14.2.3 Lessons in Software Development . . . 185
14.2.4 Lessons Learned - In Summary . . . 186
14.3 Future Development . . . 186

References **188**

IV Appendices **192**

A Demonstrator System budget **193**
A.1 Summarized budget estimation . . . 193
A.2 Budget explanation and rationale . . . 193
A.3 Budget of structural parts . . . 194

B Considering alternative milligauss meters **196**

C Blueprints of Companion Board V1.0 **199**

D Blueprints of Companion Board V2.2 204
D.1 Tips for soldering and assembly . . . 211
D.2 Board assembly instructions . . . 214
D.3 Board configuration and supported mods . . . 218
D.4 Dropper diode mounting options . . . 221

E Modifying the ES-030-10 power supply for analog control 225

F Orbital Generator 237
F.1 Requirements . . . 238
F.2 Functional Breakdown . . . 239
F.3 Functional Implementation . . . 239
F.3.1 Input parameters . . . 241
F.3.2 Determining orbital characteristics . . . 241
F.3.3 Describing orbital motion . . . 242
F.3.4 Generating magnetic field data . . . 244
F.3.5 Transforming from *NED* frame and other reference frames . . . 246
F.3.6 Transforming to the satellite body reference frame . . . 249

G Server software and TCP codec 251
G.1 Functional flow . . . 252
G.1.1 Main server thread . . . 252
G.1.2 Socket server thread . . . 253
G.1.3 Control thread . . . 254
G.1.4 ADC thread . . . 255
G.1.5 DAC thread . . . 256
G.2 Source code - SCC codec . . . 256
G.3 Server code source code . . . 263

H Details of the orbital validation tests 285
H.1 Orbit 1 . . . 286
H.2 Orbit 2 . . . 288
H.3 Orbit 5 . . . 290
H.4 Orbit 6 . . . 292
H.5 Orbit 7 . . . 294
H.6 Orbit 8 . . . 296
H.7 Orbit 9 . . . 298
H.8 Orbit 10 . . . 300

I Extra plots on inductance dynamics 302

J Method for Dipole Fitting 304
J.1 Basic Theory . . . 305
J.2 Lassakeur Implementation . . . 305
J.3 Generalized Implementation (single dipole) . . . 306
J.3.1 Programming implementation . . . 307
J.4 Generalized Implementation (multi-dipole) . . . 309
J.4.1 Programming implementation . . . 311
J.5 Simulation Work . . . 313

K Extra materials concerning the characterization of ambient magnetic disturbances 317
K.1 Overview of Tests . . . 317
K.2 Data interpretation and results . . . 320
K.3 Instructions and tools for replication . . . 334
K.4 Extra Figures . . . 335

L Dataset of Satellite Case Studies 341

List of Figures

1 An overview of the Demonstrator System (DS), showing implemented system segments. Unimplemented segments are greyed out, but are implicitly supported by the design as future expansions. . . . vi

2 Left: A Raspberry Pi 4 with a stacked DAC/ADC board handles most of the hardware interactions in the system. Right: The assembled companion board interconnects several components and implements critical system functionality. . . . vii

3 Overview of the Orbital generator, with the orbit design GUI on the left and the generated magnetic field envelope in the satellite's body frame on the right. . . . vii

4 Left: Measured magnetic field envelope (coloured) and commanded envelope (gray). The colour indicates the angular tracking error in degrees, which is a few degrees on average, but can assume higher values near the origin of the Helmholtz cage. Right: By addressing a few more issues, the performance can be improved at least twofold, resulting in better tracking. . . . viii

1.1 Diagram of the structure of this thesis. . . . 2

2.1 CubeSat missions launched to space, by year and by developer class [8] . . . 4
2.2 Artist depiction of the Capstone mission, an example of a lunar CubeSat mission. [9] . . . 5
2.3 Bottom-up approach: Breakdown of likely subsystem interference based on laws of physics 8
2.4 Top-down approach: Collecting the reported issues of 54 CubeSat missions and mapping them onto subsystems. A full list with individual references can be found in Appendix L and [22] . . . 10

3.1 Summary of the research plan as a flow diagram, with the results in the coloured boxes at the bottom. The partitioning on the right indicates how these topics are divided across this thesis. . . . 16

4.1 Example of a commercial hardware-in-the-loop (HWIL) test setup including a Helmholtz cage (Astrofein, see [39]) . . . 17
4.2 University-built Helmholtz cage for testing Delfi-C3 [33] . . . 18
4.3 An example of an unintended magnetic dipole inherent in a solar panel design and its consequences on a satellite mission. The solar panel in a) is designed in such a way that its operation induces current inside a wire loop. If the solar panel is illuminated, it will generate a perpendicular magnetic field vector $\overline{B}$ in proportion to the current $\overline{I}$ and the area ab of the loop. In c), one can see that this illumination-dependent dipole can lead to unintended torque during the satellite mission, caused by angle between the local Earth magnetic field and the panel field, leading to rotation. Even if such torques are in the order of μNm, if they are allowed to compound in space they can lead to unsalvagable mission failures, especially for satellites with low mass moment of inertia and a constrained budget for attitude control. . . . 19
4.4 Example of a deperming procedure from [40]. . . . 19
4.5 Idealized testing system that can perform all magnetic tests described in section 4.2 . . . 21
4.6 Another look at Figure 3.1. . . . 24

5.1 a) The flow of current through a wire will induce a magnetic field around it. b) When the wire is bent into a circular shape, a loop, a current through it will induce a toroidal magnetic field. . . . 26
5.2 Placing two identical coils at a distance equal to their radius results in a so-called Helmholtz coil pair. . . . 27

5.3 Simulation of the magnetic field strength on the axis between two coils. Blue represents an ideal circular Helmholtz coil pair. Red is a square Helmholtz pair with the same spacing, showing a response with lesser flatness in the middle. Yellow is the square coil at the ideal spacing for square coils, namely 0.5445 times the coil side length. 28
5.4 3D visualization of a Helmholtz cage, with three pairs of square coils. A 3U CubeSat is placed in the middle for scale. The two yellow cuboids represent the control volume in which the field strength should vary no more than 1% (larger cube), and 0.1% (smaller cube). 30

6.1 A picture of the Helmholtz cage setup in the cleanroom. The storage rack on the left and the grey closet on the right are not part of the system. Inside the Helmholtz cage there is the yellow-topped *"grid table"*. On the right there is the burgundy-coloured supply cart. 32
6.2 The coordinate systems used in this chapter. Cage-centered system C is preferred above the others, so any information in the geographic system G or sensor system S will be converted to the C frame. In both cases, this can be achieved through a single-axis (Euler matrix) rotation, as depicted. 33
6.3 Simplified system diagram of the Helmholtz cage setup. 34
6.4 Locations of power terminals for each coil. Terminals 1-4 are near the bottom of their respective coils. Terminal 5 is on the top Z-coil and terminal 6 on the bottom Z-coil. . . 34
6.5 Connections at the rear of the supply cart's computer. Left is the silver DB-25 connector that connects to the H-Bridge circuitry, and in the middle are the three gray DB-9 connectors for the RS-232 connections to the power supplies. 35
6.6 Anchored grounding wires that ground the metal cart enclosure to the two power strip grounds. 36
6.7 Electronics inside the "H-Bridge" box. 37
6.8 Reverse-engineered schematic of the polarity switching circuit of the old Helmholtz cage system 38
6.9 Visualization of an example rotation, where a magnetic field vector B is rotated in the XZ-plane. The current curves for the power supplies of the X and Z axes are also drawn, in red and blue respectively. 42
6.10 Supplied voltage envelope of **X** and **Z** during the rotation (1 rpm, 241 μT). 43
6.11 Supplied voltage envelope of **X** and **Z** during the rotation (600 rpm, 241 μT). 43
6.12 Comparison of circular and square Helmholtz coil shapes. The absolute field variation within the large vertical markers is 1%, and within the small vertical markers it is 0.1%. 44
6.13 Visualized volumes corresponding to the calculated homogeneity ranges of Table 6.4 . . 45
6.14 Simulated magnetic field strength for the pair of X-coils. The two coils are oriented vertically. 46

7.1 The Magnetic Test Facility at the Goddard Space Flight Center is an example of a Helmholtz cage and magnetic testing environment specifically designed to minimize magnetic field disturbances [51]. 48
7.2 A picture of the Helmholtz cage setup in the cleanroom. The "test area" is on top of the yellow tabletop in the middle of the Helmholtz cage. 50
7.3 Basic map of the floor space around the Helmholtz cage setup. The test area is clearly marked with a yellow box. The potential sources of magnetic interference are marked in orange. 51
7.4 Schematic of the magnetic measurement setup 53
7.5 Picture of the magnetic measurement setup before deployment: Alphalab gaussmeter main unit (left), rainbow ribbon cable to probe (bottom left) cable converter and USB-6008 data acquisition (DAQ) (center), power supply supplying 12 VDC to the milligaussmeter (center top), laptop acquiring digital data (right). 53
7.6 Rendered Helmholtz cage elements and walls in the lab. The cage-centered frame C is plotted as a standard XYZ tripod. The orange vector is the local Earth magnetic field (EMF) field and the purple vector indicates the raw, rotated measurement data. The difference between these vectors results in the black vector, which is the disturbance vector, and has a magnitude of around 18.4 μT. 55

7.7 A top-view of the situation depicted in Figure 7.6. . . . 56
7.8 24 hours of measured data, with the three axial components plotted on a common axis. 56
7.9 The data of Figure 7.8, but zoomed in on a 30 minute time interval around the lunch break in the faculty. The vertical axis scale is in μT . . . 57
7.10 The spatial test series: Average disturbance vectors from nine 45+ minute measurements in a 0.5 m XY-grid around the test area. Boxed values are the magnitude difference and the vector angle difference, compared and normalized to the centre value. For example, a value of 1.99 $\mu T/m$ indicates that if we assume linear behaviour, the magnitude of the disturbance vector changes by 1.99 μT for each meter that we move away from the centre in that particular direction. . . . 58

8.1 After scoping down the idealized test system, we are left with a subset of it, which we call the demonstrator system. . . . 63
8.2 Another look at Figure 3.1. . . . 71

9.1 After scoping down the idealized test system, we are left with a subset of it, which we call the demonstrator system. . . . 73
9.2 Implemented degrees of separation between system user and system hardware. . . . 75
9.3 Left: MCC-152 DAC board by Digilent. Right: CN-0554 DAC/ADC board by Analog Devices. In both cases, the board is mounted atop a Raspberry Pi microprocessor unit. . 77
9.4 Block diagram of the hardware layout of the demonstrator. . . . 79

10.1 A polarity switcher reverses the direction of the current applied to the Helmholtz coil pairs by effectively making the control current bipolar. . . . 81
10.2 An H-bridge circuit with single-pole, single-throw (SPST) switches in several states . . . 82
10.3 A naive first design for the H-bridge circuit. The four MOSFETs are controlled by a common control line. The control signal to MOSFET pair 2-3 first passes through an inverting stage, so that MOSFET pairs 1-4 and 2-3 are always doing the opposite of one another. . . . 83
10.4 Modifying the control signal to the MOSFET pairs such that there is a dead time in between each switched state. . . . 83
10.5 Introducing a propagation delay in the rising edge response to ensure a dead time. . . . 84
10.6 (a) Simple RC filter circuit (b) RC filter with downstream comparator. . . . 84
10.7 Enhanced RC delay with buffered input . . . 85
10.8 Two parallel delay channels controlling an H-bridge circuit. The corresponding Falstad simulation is available at https://tinyurl.com/2lom5x6a . . . 85
10.9 Schematic diagram of an N-channel power MOSFET [53]. . . . 86
10.10 An issue arises when trying to control the high side MOSFETs of an H-bridge, as a suitable V_{GS} is not possible to manage with a static 10 V control signal. . . . 87
10.11 (a) DRV8262 dual bridge driver in a HTSSOP-44 package. (b) LTC7060 half-bridge driver in an MSOP-12 package. . . . 88
10.12 Typical application of the LTC7060 half-bridge driver recommended by the manufacturer [54]. The same device can be configured to be inverting or non-inverting the pulse width modulation (PWM) input signal. Also note the two different ground symbols, indicating that the ground of the H-bridge and the control circuitry are isolated. . . . 89
10.13 SPICE simulation of an N-channel MOSFET H-bridge circuit controlled by dual LTC7060 bridge drivers, similar as in Figure 10.12, but as a full H-bridge. The load corresponds approximately to one Helmholtz coil pair. The switching signal (PWM) is shown in yellow, the gate voltages of the inverting and non-inverting high-side MOSFETs are green and red respectively, and the current through the load is magenta. Note the brief, periodic "off" moments used to recharge the boost capacitor, happening approximately 250 times per second. . . . 90
10.14 Implementation of one H-bridge channel, this one for the Z-axis coil pair. The left LTC7060 is configured to be inverting, meaning that with a low input signal, current flow through the coils is from the left half-bridge to the right half-bridge. . . . 90

10.15On the left, typical application of the LTC7060 half-bridge driver recommended by the manufacturer [54]. On the right, a modified implementation where the boost capacitor and charging diode are replaced with a static voltage source. 91
10.16Testing whether two 9V batteries can be used in conjunction with the LTC7060 to turn on the H-bridge MOSFETs. A number of other changes are being prototyped here also, such as Schmitt-trigger input line buffers. Note the spaghetti of bodging wires, which will cause intermittent self-oscillatory behaviour issues, described later on in this chapter. 92
10.17Creating an everlasting yet galvanically insulated potential to replace the 9V batteries. . 92
10.18Prototyping the transformer-based implementation using a modified SN6501EVM-MULTI-XFMR evaluation board. 93
10.19The charge reservoir networks implemented on the boards use an SN6501 transformer driver in conjunction with a 1:1.1 transformer. The latter is the 760390012 push-pull transformer by Würth Elektronik, an affordable, effective and very pretty SMD transformer. 93
10.20Implementation of one H-bridge channel after the second design iteration. Where the previous iteration used boost capacitors, this circuit has replaced them by the more involved but more reliable transformer-based isolated voltage sources. 94
10.21A Kelvin connection is a method to perform electronic sensing that compensates for lead resistance. This can be very useful for applications where low resistances are being measured, such as measuring current with a low-resistance shunt resistor. 97
10.22First iteration of the current sensing implementation. **a)** Circuit for a single current sensing channel **b)** PCB layout for this circuit, with the Kelvin connection clearly visible **c)** Microscopy of the implementation on PCB **d)** CAD 3D render of the layout. 97
10.23Internal model of an INA290 current sense amplifier as given by the manufacturer [58]. 98
10.24Four viable current sense resistor series [59] [60] [61] [62]. **a)** and **b)** together are footprint compatible, and **c)** and **d)** are together also footprint compatible. 100
10.25Second iteration of the current sensing implementation. **a)** Circuit for a single current sensing channel **b)** PCB layout for this circuit **c)** Microscopy of the implementation on PCB **d)** CAD 3D render of the layout. 102
10.26Test to characterize the performance of the zener diodes used for transient suppression. The zener diodes were connected together anode-to-anode and then in parallel with the load. Oscilloscope probes at either end measure the voltage, and math function is used to find the difference. Temperature is monitored using a thermocouple and a nearby infrared camera. 105
10.27Close-up of the zener diode pair in parallel with the load. The blue-brown cables connect to the Helmholtz coil pair. The NTC thermistor makes contact with the body of the zener diode. 105
10.28Zener diode clamping response to a 1 A square wave. Yellow trace and blue trace are either side of the load. Orange trace is a math trace subtracting blue from yellow. On the left, the voltage spike is "negative" because of this convention. 106
10.29Clamping response of a 3 A square wave at different time bases. Left: 400 ns/div. Right: 400 us/div. 106
10.30Clamping response of a 4 A switch. At 1.4 ms into the transient, the zener diode cannot handle the energy and fails in near short-circuit. The dissipated energy rapidly mounts due to the current, and the device gets so hot it starts to smoke. 106
10.31Replacements for the unsuitable BZT03C75 [69] as a voltage suppressor diode clamp. The 1.5KE [70] and 1.5SMB [71] devices handle about five times the pulse current, and the 15KPA [72] device handles over forty times the pulse current of the BTZ03. 107
10.32Littelfuse 1.5SMB transient voltage suppression (TVS) diode clamps on a later iteration of the companion board. 107
10.33Overview of the transient voltage suppression strategy with which the H-bridge circuit is designed. 108
10.34Schematic of the power section (left) and the connector routing (right) of the Helmholtz cage companion board v1.0. 110
10.35Schematic of a single current channel on the Helmholtz cage companion board v1.0. . . 111
10.36CAD visualization of the trace routing of companion board v1.0. 111

10.37The CN-0554 board arrived with short header pins pre-soldered, rather than long pass-through header pins. This makes it more difficult to expose unused GPIO pins of the Raspberry Pi. 112
10.38Low input impedance of the LTC7060 PWM pin and LED indicator creating problems for the H-bridge control. **a)** Internal block diagram of the LTC7060 [54] **b)** equivalent circuit of the input impedance that the control pin has to overcome **c)** buffering the control signal. 113
10.39A fundamental flaw of the 1685B power supplies is revealed: it cannot drive the inductive load posed by the Helmholtz cage coils. **On the left**: Driving a non-inductive load with a 5 A peak-to-peak square wave, which the supply tracks without problems, despite a substantial slew rate. The orange trace is intended current, the red trace is output current. **On the right**: Output **voltage** of the power supplies when trying to drive a constant 1 A through the inductive load of a Helmholtz coil pair. The 1685B output voltage oscillates with a 7-10 V peak-to-peak at a rate of about 10 Hz, fundamentally disqualifying the 1685B for this application. 114
10.40Modifying the ES-030-10 power supplies by removing the factory-installed RS-232 interface card and instead exposing six analog interface pins: Four for current and voltage control and monitoring, one for a ground reference, and one for internal reference voltage. More details about this modification can be found in Appendix E 115
10.41Two possible circuits for implementing a two-condition inhibition scheme. The circuit on the right was incorporated in the companion board. 116
10.42Testing the implementation of Figure 10.41 by disconnecting the `EN` on the LTC7060 from the internal supply rail, and to a common breadboard-compatible jumper pin. This involved exposing, cutting (left) and soldering together traces (right), one of many occasions during the developments where such bodges were implemented. 117
10.43Complete overview of the trace layout of the companion board v2.2. A higher resolution version can be found in Appendix D. 118
10.44Power input section of the Helmholtz cage companion board v2.2 as **a)** schematic **b)** 3D render. **c)** is the current inhibition scheme discussed in subsection 10.5.3. 120
10.45Routing of board connections for the Helmholtz cage companion board v2.2. 121
10.46Power input section of the Helmholtz cage companion board v2.2 as **a)** schematic **b)** 3D render. **c)** is the current inhibition scheme discussed in subsection 10.5.3. 122
10.47Assembled V2.2 companion board receiving power and being tested. 123

11.1 Implemented degrees of separation between system user and system hardware, and the contents of the **Helmholtz Cage Toolkit** software package in relation to this. 125
11.2 Diagram showing how users can interact with the Helmholtz Cage Toolkit software on the high-level interface. One can either call the functions directly in a script, using the API, or use the graphical user interface. A command line interface was also considered early on, but was later abandoned. 127
11.3 Early mock-up to plan out how the GUI should be structured. 128
11.4 Interaction interface between human user and Helmholtz cage system for the creation of magnetic field vector schedules. The generator scripts inside the HCT translates parametric, human-interpretable instructions to instructions that the Helmholtz cage system can use. 129
11.5 Overview of the Connection Window. 131
11.6 Overview of the command window in manual mode. 132
11.7 Using the command window in scheduled mode. 134
11.8 Overview of the Cyclics generator. Input parameters are on the left, and some visualizations of the resulting schedule are on the right. In this case, a helical test pattern is generated using a sine (X), a cosine (Y), and a triangle wave (Z). The resulting schedule will be 10 seconds long, and describe a periodic helix with an absolute magnitude of at most 100 μT on every axis. 135
11.9 Reference frames used by the Orbitals generator. 137
11.10Overview of the Orbital generator, with input parameters on the left and orbital visualization on the right. 138

11.11 Two possible views of the Orbital generator's output in terms of the activity of the Helmholtz cage. The red-green-blue colours represent the X-Y-Z axes respectively. On the left, we see clearly what instantaneous magnetic field vector is being established by the Helmholtz cage at the current time, as well as a brief history of the previous positions using the trail. On the right, we have a more global overview of the whole magnetic contour described by the schedule. This view shows the complexity of the contour, and therefore shows why the Orbital generator was made: it would take a long time to create such a contour by hand, whilst the generator does it in under 2 seconds. 139
11.12 When saving a schedule to a file using the GUI application, the user is prompted with the option to save the generation parameters to the `.bsch` file, in addition to the raw schedule data. 140
11.13 An example of the first 29 lines of a B-schedule file. This schedule was created using the Orbitals generator in the GUI application, and as a result, the generation parameters are included in the header. 141
11.14 Abstract layout of the low-level interface. The interface software runs inside an Unix-like operating system environment on a Raspberry Pi 4. It functions as a transmission control protocol (TCP) server, hosting a unique transmission control protocol (TCP) socket for each client device connected to it. For advanced users, the operating system (OS) that runs the low-level interface can be directly interacted with by attaching a screen, keyboard, and mouse to the Raspberry Pi 4, which is useful for debugging and detailed configuration. 142
11.15 Functional flow of the main thread of the server script. Detailed functional flow diagrams for each thread can be found in Appendix G 143
11.16 Anatomy of all packet types in the 4th iteration of the Simple Command Codec 146
11.17 Automated software testing of the low-level interface software: on the left, the debug terminal of the low-level interface code can be seen, including raw packet information. On the right is the output of `*/tests/server_benchmarks.py`, a 900+ line testing script that connects to the server as a client and checks many functionalities. 148
11.18 Automated testing and benchmarking of the Simple Command Codec (SCC) codec using `*/tests/scc_benchmarks.py`, here with 10^5 iterations. 148
11.19 [SIDEWAYS] An overview of the layout, dependency relation, and general content of the files inside the Helmholtz Cage Toolkit package. 150

12.1 Early stages of integration with all system components connected, though left exposed for testing. 151
12.2 Pushing a constant current of 1 A through each pair of coils reveals the series resistance of the coil and the cabling in Ohms. From top to bottom: X, Y, Z. 152
12.3 Cable connection and label convention used during integration. Black cables have a blue and brown line each, brown is typically used as negative. Three different colours of tape are used; brown for X, white for Y, blue for Z, with two tapes denoting the negative coil of the pair. 152
12.4 Detail of cable labeling for the high-current coil cables. 153
12.5 Detail of cable labeling for the high-current coil cables. 153
12.6 Observing companion board temperatures with an infrared camera during a high current load test. 154
12.7 Using `server_debug_gui.py` to test the Helmholtz cage from the low-level interface. . . 154
12.8 Using the Helmholtz cage to rotate a compass needle at 120 rpm (2 Hz) during a stress test. Note the motion blur on the needle. 155

13.1 Simulated magnetic field strength for the pair of X-coils, for a centre field strength of 250 μT. The two coils are oriented vertically. On the left, all field values with a deviation from the centre value of 1% or more have been discarded, and the 415 mm outline of the control volume has been superimposed in white. On the right, the same has been done, but for a 2% deviation limit. Refer to section 6.2 for more details. 157

13.2 Pushing the power supplies to their maximum compliance voltage reveals the maximum current capability through each coil pair. From top to bottom, the supply units correspond to the $X/Y/Z$ coils. The Z-coils have the largest resistance, limiting the current to about 4.8 A. 162
13.3 During auto-calibration, the relationship between input current and output field strength are measured on each axis (dots in the figures). A cubic (left) and quintic (right) polynomial were fitted to the inverted data, in the hope that such a function would correct for the non-linearities. 167
13.4 After applying the calibration polynomial, another series of measurements is performed to measure the actual transfer characteristics. The cubic (left) polynomial was chosen over the quintic (right) polynomial, since it is cheaper to compute and appears to perform better at the extremes. The gray dots are the target for perfect linearity. 168
13.5 Periodic test schedule used to test some baseline tracking performance of the system. On the top, we see the $X/Y/Z$ components of $\overline{B}_C$ in red/green/blue respectively, and the measured magnetic field components $\overline{B}_M$ in magenta/yellow/cyan. In the middle, we see the absolute command error for each component (red/green/blue) as well as their norm (E_C) (white). In the bottom, we see the angular error Λ in orange. 169
13.6 The effect of using a schedule with a low resolution produces a larger overall error. The Helmholtz cage is forced to track between large discrete steps, which causes a large tracking error just after such a step and decreases as the power supplies adjust. 170
13.7 Helmholtz cage tracking a step function of various amplitudes. 171
13.8 Setting up a satellite orbit simulation using the *Orbitals* generator in the HCT. 172
13.9 Viewing the resulting instructions for the Helmholtz cage, which represent the magnetic field experienced by the satellite. 172
13.10 The envelope and error properties of the schedule after being played on the Helmholtz cage. 173
13.11 The envelope of the simulated satellite schedule seen as its 3D manifold. The gray line represents the commanded vector path $\overline{B}_C$, whereas the coloured one represents the measured data $\overline{B}_M$. The colour gradient is based on the magnitude of the absolute error, meaning that green line segments have a low absolute error, and red areas have a high absolute error. 174
13.12 The same schedule as before, but zoomed in on a minute long segment. The top graph shows the components of $\overline{B}_C$ and $\overline{B}_M$, the middle graph shows commanded and measured current $\overline{I}_C$ and $\overline{I}_M$. The bottom graph shows the magnitude errors as before. 175
13.13 Various problems found with the current system. Top left: A sporadic 1.2 second freeze-up of the Raspberry Pi, as seen from the commanded and recorded magnetic fields. Bottom left: The (rather useless) corresponding absolute error. Top right: Two freeze-up glitches causing temporary poor tracking for a prolonged period afterwards. Bottom right: Operating at a 60 S/s command rate results in timing de-sync, causing the actual schedule to be played almost three times slower than real-time. 176
13.14 The absolute field error after various stages of filtering, in order to assess how much error would remain if the known problems would be solved. The brighter the curve, the more filters were applied. The black line is a dimensionless representation of the commanded magnetic field and is plotted for reference purposes. 177
13.15 The angular error before (orange) and after (blue) the data has been filtered, indicating that addressing the H-bridge/diode related issue and improving the calibration can improve the tracking performance significantly. 177

14.1 Top left: Minor PCB defect from forgetting to flood-fill the ground planes. This one could be fixed by cutting away some ground plane copper. Lesson: always flood-fill your ground planes again before exporting your gerbers. Bottom left: give yourself room for boding or be doomed to work with what you've got. Right: Board testing with an incandescent bulb as an *ad hoc* dummy load. If it works, it's not stupid. However, the sheer number of bodging wires would later cause undesirable self-oscillatory behaviour in the board. 184

B.1 Reading out the AlphaLab milligauss at 2500 S/s on all three channels by connecting a high-quality ADC to its outputs. This is made possible because the outputs are analog, and therefore not restricted by any built-in ADC. . . . 196
B.2 The cuboid probe of the tri-axial magnetometer can be easily mounted and aligned in a test location. If three different single-axis magnetometers would have been used, the three probes would have had to be positioned with much greater difficulty. . . . 198

C.1 Companion board v1.0 . . . 199

D.1 Companion board v2.2 . . . 204
D.2 Component overlay for the V2.2 companion board, useful during soldering. . . . 213
D.3 Example of changing a board configuration by modifying the jumper pad. In this particular case, it will cause the gate of high-side MOSFET Q5 to be driven by 11 V rather than 5.5 V. . . . 221
D.4 Mounting the diodes underneath the board, facing up. Two screws through the auxiliary board holes on the side carry the weight of the heat sink and electrically ground it. The diodes are mounted to the heatsink by tightening a screw through the top, which is facilitated by the access slots in the board above. . . . 222
D.5 Same as Figure D.4 but viewed from the bottom. Note that the diodes are positioned before soldering, which is the recommended method of mounting them. . . . 222

E.1 The rear panel on two Delta Elektronik power supplies. The top one features the standard DB15 control ports, which does feature the analog control pins. The bottom one contains the RS-232 interface connector. The power supplies in question all contained the RS-232 interface. . . . 225
E.2 Two ES-030-10 power supplies in a twin rack mount. . . . 226
E.3 Disconnecting the power supplies from their rack mount. . . . 227
E.4 Opening the power supply chassis. . . . 227
E.5 Back of the internal RS-232 board. . . . 228
E.6 RS-232 board after disconnecting the mounting screws. . . . 228
E.7 Freeing the indicator LED. . . . 229
E.8 Removing the insulation sheet. . . . 229
E.9 Disconnecting the 10x2 ribbon connector from the RS-232 board. . . . 230
E.10 Successful extraction of the internal RS-232 card. . . . 230
E.11 Pin-out of the DB15 interface connector for the Delta Elektronik ES-300 series. . . . 231
E.12 Location of the analog pins on the internal 10x2 header. . . . 231
E.13 Preparing a temporary internal connector. . . . 232
E.14 Connecting and routing the temporary internal connector. . . . 232
E.15 Using a zip tie as strain relief for the cable. . . . 233
E.16 Closing the chassis. . . . 233
E.17 Two successfully modified ES-030-10 units back inside their rack mount. . . . 234
E.18 Replaced the old RS-232 interface with a custom Insulation Displacement Connector (IDC) connector interface. . . . 235
E.19 Internal wiring of the Insulation Displacement Connector (IDC) connector. . . . 235
E.20 Sealing the wire ends of the Insulation Displacement Connector (IDC) connector with heatshrink wrapping and relieving mechanical strain on the connections by epoxy blobbing them. The epoxy is only connected to the connector and the coupler, so if needed it could be easily replaced. . . . 236
E.21 All three power supplies modified to satisfaction. . . . 236

F.1 Interaction interface between human user and Helmholtz cage system for the creation of magnetic field vector schedules. The generator scripts inside the HCT translates parametric, human-interpretable instructions to instructions that the Helmholtz cage system can use. . . . 237
F.2 Functional breakdown of the Orbitals generator code. . . . 239
F.3 Reference frames used in this thesis. . . . 240

F.4 (a) Equidistant spacing - The travel distance between each point is equal (b) Isochronal spacing - The orbital transit time between each point is equal. 242
F.5 Performance of the Equation of the Centre approximation for various values of the eccentricity. Note the increase in error for high values of e. Increasing the order of the method yields better result (compare left and right) at a higher computation cost (compare the evaluation time). 243
F.6 Orbit simulation using the discussed tools. In orange, a highly inclined orbit has been plotted in the ECI frame (large tripod). The $ECEF$ frame (small tripod) rotates as Earth rotates over time. The yellow line from the centre of the Earth to the highlighted point is radius r. The other large yellow line indicates the instantaneous velocity vector of the satellite at the given time step. 245
F.7 Various local NED frames inside a global $ECEF$ frame. Point P can be expressed in the global $ECEF$ frame with coordinate $P(1,1,-1)\,|_{ECEF}$. It can also be expressed in the local NED frame at $(1,1,-1)\,|_{ECEF}$, where it now has coordinate $(-1,-1,1)\,|_{NED}$ 248
F.8 Integration of the International Geomagnetic Reference Field (IGRF) model. A magnetic field vector (blue) is measured at the location of the satellite, after which it is transformed from the NED frame (small tripod) to the general ECI frame. A number of International Geomagnetic Reference Field (IGRF)-based field lines are also plotted in blue. 249

G.1 Anatomy of all packet types in the 4th iteration of the Simple Command Codec 257

H.1 Left: orbital view in $ECEF$ frame. Right: generated magnetic envelope in B-frame. . . . 286
H.2 Test data of `TS_orbit1_2_202511211327_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue). 287
H.3 Left: measured magnetic field manifold. Right: manifold after filtering. 287
H.4 Left: orbital view in $ECEF$ frame. Right: generated magnetic envelope in B-frame. . . . 288
H.5 Test data of `TS_orbit2_202511211346.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue). 289
H.6 Left: measured magnetic field manifold. Right: manifold after filtering. Due the reduced overall scale of this schedule, the measurement noise is much more visibly pronounced. 289
H.7 Left: orbital view in $ECEF$ frame. Right: generated magnetic envelope in B-frame. . . . 290
H.8 Test data of `TS_orbit5_202511211512_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue). 291
H.9 Left: measured magnetic field manifold. Right: manifold after filtering. 291
H.10 Left: orbital view in $ECEF$ frame. Right: generated magnetic envelope in B-frame. . . . 292
H.11 Test data of `TS_orbit6_202511211603_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue). 293
H.12 Left: measured magnetic field manifold. Right: manifold after filtering. 293
H.13 Left: orbital view in $ECEF$ frame. Right: generated magnetic envelope in B-frame. . . . 294
H.14 Test data of `TS_orbit7_20251125_FIX`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue). 295
H.15 Left: measured magnetic field manifold. Right: manifold after filtering. 295
H.16 Left: orbital view in $ECEF$ frame. Right: generated magnetic envelope in B-frame. . . . 296
H.17 Test data of `TS_orbit8_20251125_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue). 297
H.18 Left: measured magnetic field manifold. Right: manifold after filtering. Zero-crossings in the X direction cause stepped errors. The filter that removes these leaves artifacts, which are clearly visible on the right. 297
H.19 Left: orbital view in $ECEF$ frame. Right: generated magnetic envelope in B-frame. . . . 298

H.20 Test data of `TS_orbit9_20251125_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue). . . . 299
H.21 Left: measured magnetic field manifold. Right: manifold after filtering. . . . 299
H.22 Left: orbital view in $ECEF$ frame. Right: generated magnetic envelope in B-frame. . . . 300
H.23 Test data of `TS_orbit10_20251125_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue). . . . 301
H.24 Left: measured magnetic field manifold. Right: manifold after filtering. . . . 301

I.1 Supplied voltage envelope of **X** and **Z** during the rotation (1 rpm, 241 μT). . . . 302
I.2 Supplied voltage envelope of **X** and **Z** during the rotation (60 rpm, 241 μT). . . . 302
I.3 Supplied voltage envelope of **X** and **Z** during the rotation (300 rpm, 241 μT). . . . 303
I.4 Supplied voltage envelope of **X** and **Z** during the rotation (600 rpm, 241 μT). . . . 303

J.1 Measuring the net magnetic moment of a CubeSat. We model it as a simple dipole, with a certain position referenced to an axis system, a direction, and a magnitude. Sensor S_2 will be able to measure this dipole more accurately than sensor S_1, as the measurements of S_1 will be affected by near-field effects. . . . 304
J.2 Sign and symbol convention . . . 306
J.3 Visualized simulation results. Blue: sensor points and measured vectors. Red: actual dipole. Gray: initial guess. Green: converged estimation. . . . 313
J.4 Attempting to gain insight into the performance of the estimator's sensitivity to initial guess. Blue: sensor points and measured vectors. Red: actual dipole. Gray: initial guess passing 1 checks. Orange: initial guess passing 2 checks. Green: initial guess passing all 3 checks. Initial conditions that do not pass any checks are not drawn at all. . . . 315

K.1 Basic map of the floor space around the Helmholtz cage setup. The test area is clearly marked with a yellow box. The potential sources of magnetic interference are marked in orange. . . . 318
K.2 Raw magnetic flux data showing coarse vertical discretization, which is caused by the 12bit limitation of the data acquisition (DAQ) card. . . . 321
K.3 The coordinate systems used in this chapter. Cage-centered system C is preferred above the others, so any information in the geographic system G or sensor system S will be converted to the C frame. In both cases, this can be achieved through a single-axis (Euler matrix) rotation, as depicted. . . . 321
K.4 Rendered Helmholtz cage elements and wall elements in the lab. The cage-centered frame C is plotted as a standard XYZ tripod. The orange vector is the local Earth magnetic field (EMF) field and the purple vector indicates the raw, rotated measurement data. The difference between these vectors results in the black vector, which is the disturbance vector, and has a magnitude of around 18.4 μT. . . . 322
K.5 A top-view of the situation depicted in Figure 7.6. An interactive 3D view of this image can be plotted using the file `disturbance_visualization.py` on the code repository. . 323
K.6 [SIDEWAYS] 24 hours of measured data, with the three axial components plotted on a common axis. . . . 324
K.7 [SIDEWAYS] The same data as in Figure K.6, but now plotted on three separate axes. The scale of the axes is forced to be identical, so that the proportional differences between the three axial components can be compared apples-to-apples. . . . 325
K.8 The data of Figure K.6, but zoomed in on a 30 minute time interval around the lunch break in the faculty. . . . 326
K.9 Diurnal variation in standard deviation of magnetic field data, calculated by hour. Source data from Weekday4 test. . . . 327
K.10 50 Hz oscillation within a half-second interval of Y-axis magnetic measurements during nighttime. Source: Weekday4 test, 2500 S/s. . . . 327
K.11 [SIDEWAYS] fast Fourier transfer (FFT) spectral data of Lights On Test 2, consisting of 5 minutes of data sampled at 2500 S/s with the cleanroom ceiling lights turned on. . . . 328

K.12 fast Fourier transfer (FFT) spectral data of Lights Off Test 2, consisting of another 5 minutes of data sampled at 2500 S/s, but this time with the cleanroom ceiling lights turned off. Compared to Figure K.11, the peaks associated with the 50 Hz alternating current (AC) frequency and its harmonics have been attenuated substantially. 329
K.13 Time plot of a 10-minute snippet during the night, without any filtering applied. 330
K.14 Spectrum plot of Figure K.13. 330
K.15 Time plot of a 10-minute snippet during the night, with the Savitzky-Golay filter (SGF) applied. 331
K.16 Spectrum plot of Figure K.15. 331
K.17 Time plot of a 10-minute snippet during the night, with the Savitzky-Golay filter (SGF) and 7 notch filters applied. 332
K.18 Spectrum plot of Figure K.17. 332
K.19 The spatial test series: Nine 45+ minute measurements in a 0.5 m grid around the test area. 334
K.20 The tri-axial probe of the milligaussmeter, fastened within a 3D printed retention clip, which in turn is fastened to the table at the test area. The retention clip was made to easily remove the probe in between tests if needed, without losing consistency in alignment of the probe across tests. 335
K.21 Structural suspension jig above the Helmholtz cage frame. 336
K.22 Helmholtz cage during the Coils Wide Test, with its three pairs of coils extended as far as they would go. Notice the absence of the storage rack, which was removed before these tests started. 336
K.23 Helmholtz cage during the Coils Close Test, with its three pairs of coils moved as close to the test area as possible. 337
K.24 The coils moved back to their theoretically ideal spacing for square Helmholtz coil pairs (0.5445 times the side length). 337
K.25 [SIDEWAYS] Weekday1 test time plot 338
K.26 [SIDEWAYS] Weekday2 test time plot 339
K.27 [SIDEWAYS] Weekday3 test time plot 340

L.1 Top-down approach: Collecting the reported issues of 54 CubeSat missions and mapping them onto satellite subsystems. 342

List of Tables

2.1 Clarification of the items in Figure 2.3 8

6.1 Characteristics of the old Helmholtz cage system 38
6.2 Measured coil resistances 39
6.3 More measured coil and cable resistances 39
6.4 Calculated ideal coil spacings and homogeneity ranges 45

7.1 Measured magnetic field properties on the XY inside the Helmholtz cage. 59
7.3 Standard deviations of data at various stages of filtering. All values are expressed in μT. 61

10.1 Considered MOSFETs in the trade-off 95
10.2 Prospective shunt resistor series found on Digikey 100

13.2 Command error observed during simple helical test 170
13.3 Typical step response settling times for the system (Z-coils) 171
13.4 Command error observed during orbit simulation test 173
13.5 Command error for the **filtered** orbit simulation data. This performance is plausible if the observed problems are addressed. 177
13.7 Command errors and angular errors, before and after filtering. Lower errors are better. 179

D.2 Configuration options and supported mods for the V2.2 companion board. 218

F.1 Overview of reference frames used in the following sections. 239
F.2 Input parameters of the Orbitals generator 241
F.3 Various constants 241

H.2 Command errors and angular errors, before and after filtering. 286
H.3 Command errors and angular errors, before and after filtering. 288
H.4 Command errors and angular errors, before and after filtering. 290
H.5 Command errors and angular errors, before and after filtering. 292
H.6 Command errors and angular errors, before and after filtering. 294
H.7 Command errors and angular errors, before and after filtering. 296
H.8 Command errors and angular errors, before and after filtering. 298
H.9 Command errors and angular errors, before and after filtering. 300

K.2 Standard deviations of data at various stages of filtering. All values are expressed in μT. 333

Nomenclature

List of Symbols

Symbol	Definition	Constant value	Unit
B	Magnetic flux density[3]		$[T]$ $[G]$
C	Capacitance		$[F]$
E	Eccentric anomaly		$[°]$
e	Ellipsoidal eccentricity		$[-]$
f	Frequency		$[Hz]$
f_S	Sample rate		$[Hz]$ $[S/s]$
f_c	Filter cut-off frequency		$[Hz]$
h_p	Orbital altitude at periapsis		$[m]$
G	Gain		$[-]$
I	Electric current		$[A]$
i	Orbital inclination		$[°]$
L	Inductance		$[H]$
M	Mean anomaly		$[°]$
M[4]	Dipole magnetic moment		$[A/m^2]$
P	Power		$[W]$
R	Electric resistance		$[\Omega]$
r_a	Radius of apoapsis		$[m]$
r_E	Mean Earth radius	$6.378137 \cdot 10^6$ [7]	$[m]$
r_p	Radius of periapsis		$[m]$
T	Temperature		$[K]$ $[°C]$
T_{orbit}	Orbital period		$[s]$
V	Velocity		$[m/s]$
V[5]	Voltage		$[V]$
Γ	Irradiance		$[W/m^2]$
θ	Body angle (Y-axis)		$[°]$
Λ	Angular pointing error		$[°]$
μ_0	Vacuum magnetic permeability	$4\pi \cdot 10^{-7}$	$[N/A^2]$
μ_E	Geocentric gravitational constant of Earth	$3.98600442 \cdot 10^{14}$ [7]	$[m^3/s^2]$
ν	True anomaly		$[°]$
ρ	Density		$[kg/m^3]$
φ	Body angle (X-axis)		$[°]$
ψ	Body angle (Z-axis)		$[°]$
Ω	Right ascension of the ascending node (RAAN)		$[°]$
Ω_E	Mean angular rate of Earth	$7.292115 \cdot 10^{-5}$ [7]	$[rad/s]$
ω	Argument of periapsis		$[°]$
ω[6]	Angular frequency		$[°/s]$ $[rad/s]$ $[rpm]$ $[Hz]$

[3] Throughout this thesis, we use "magnetic flux density" synonymously with "field strength". The more common definition of "field strength" expressed as the H-field we will leave to the side.

[4] In correspondence with common notation, we use M for both the mean anomaly and the dipole magnetic moment. There is a potential of ambiguity with this, but in this thesis these two quantities are used in separate discussions, so that no ambiguity should occur.

[5] See footnote 2

[6] See footnote 2

Acronyms

ABS	Acrylonitrile Butadiene Styrene (a common plastic used for additive manufacturing)
AC	alternating current
ADC	analog-to-digital converter
ADCS	attitude determination and control system
AE	faculty of Aerospace Engineering of TU Delft
API	application programming interface
ASCII	American Standard Code for Information Exchange (a common 7-bit character encoding standard)
BJT	bipolar junction transistor (a type of transistor)
BNC	Bayonet Neill-Concelman (a connector common in oscilloscopes and other test equipment)
BOM	bill of materials
CAD	computer-aided design
CC	current control
CLI	command line interface
COTS	commercially-off-the-shelf (typically indicates that something is not custom-made)
DAC	digital-to-analog converter
DAQ	data acquisition
DC	direct current
DIP	Direct In-line Package (a package type for microchips)
DIY	do-it-yourself (typically denotes that something is not professionally manufactured)
DOF	degree of freedom
DPDT	double-pole, double-throw (a type of switch with two sets of two contacts)
DS	Demonstrator System (see chapter 8)
DSP	Digital Signal Processing
ECEF	Earth-Centered, Earth-Fixed (a reference frame)
ECI	Earth-Centered Inertial (a reference frame)
ECSS	European Cooperation for Space Standardization
EMC	electromagnetic compatibility
EMF	Earth magnetic field
EMI	electromagnetic interference
ENU	East-North-Up (a reference frame, related to North-East-Down (NED))
EPS	electrical power system
ESA	European Space Agency
ESD	electrostatic discharge
FET	field effect transistor
FFT	fast Fourier transfer (a computational method used for spectral analysis)
FPAK	FullPAK (a variant of the TO-220 semiconductor package that is completely encased in electrically non-conductive plastic)
FWR	full-wave rectifier
GMAT	General Mission Analysis Tool (orbit design software developed by NASA)
GNSS	Global Navigation Satellite System
GPIO	general purpose input-output (An input-output concept on many microcontrollers)

GPL	GNU General Public License (a software license)
GPS	Global Positioning System
GUI	graphical user interface
HCT	Helmholtz Cage Toolkit
HHC	Helmholtz cage
HV	high voltage (formally 1000 VAC or 1500 VDC and above)
HWIL	hardware-in-the-loop (usually in the context of HWIL testing)
HWR	half-wave rectifier
IDC	Insulation Displacement Connector (a type of connector common to PCB interconnect ribbons)
IGBT	insulated gate bipolar transistor
IGRF	International Geomagnetic Reference Field
IMU	inertial measurement unit
ISR	interrupt service routine
ISS	International Space Station
JFET	junction field effect transistor (a type of FET)
LDO	low drop-out
LED	light emitting diode
LEO	low Earth orbit
LGPL	GNU Lesser General Public License (a software license)
LPF	low-pass filter
MEO	medium Earth orbit
MOSFET	metal oxide semiconductor field effect transistor (a type of FET)
MPPT	maximum power point tracker
MTM	magnetometer
MTS	magnetic testing system
NASA	National Aeronautics and Space Administration
NED	North-East-Down (a reference frame, related to East-North-Up (ENU))
NPN	NPN transistor (A bipolar transistor topology using alternating N-doped and P-doped segments)
OBS	on-board computer
opamp	operational amplifier
OS	operating system
PC	Polycarbonate (a common plastic used for additive manufacturing)
PCB	printed circuit board
PLA	polylactic acid (a bio-degradable plastic common in additive manufacturing)
PMAS	Passive Magnetic Attitude control System (a type of ADCS)
PNP	PNP transistor (A bipolar transistor topology, counterpart to NPN)
PSU	power supply unit
PWM	pulse width modulation
RAAN	Right ascension of the ascending node (a property of a modeled orbit)
RAM	random access memory
RF	radio frequency
RLC	RLC (an electrical circuit consisting of a resistor, inductor, and capacitor)
RMS	root mean square
rpm	Rounds per Minute (unit of angular frequency)
RRA	repetitive reverse avalanche
RTL	resistor-transistor logic
SCC	Simple Command Codec (a TCP communication codec described in section 11.6)
SGF	Savitzky-Golay filter
SMD	surface mounted device
SNR	signal-to-noise ratio
SPDT	single-pole, double-throw (a type of switch with two contacts)
SPICE	Simulation Program with Integrated Circuit Emphasis

SPST	single-pole, single-throw (a type of switch with a single contact)
SSH	Secure Shell (a serial interface used in remote computing)
SWIL	software-in-the-loop (usually in the context of SWIL testing)
TCP	transmission control protocol
TLE	Two-line element (minimalistic, standardized format for orbital telemetry)
TUDAT	TU Delft Astrodynamics Toolbox
TVS	transient voltage suppression
UDP	user datagram protocol
UI	user interface
USB	universal serial bus
VAC	voltage, explicitly for alternating current
VC	voltage control
VDC	voltage, explicitly for direct current
WMM	World Magnetic Model

1 Introduction

The work described in this thesis was performed with the explicit intention to reduce the failure rate of small satellite missions. In particular, that of nanosatellite missions with a comparatively small budget or a relatively short development cycle. As part of this, a magnetic testing system was designed and developed. This system was designed to be easily reproducible by others, which is why certain sections of this thesis serve as a blueprint for it.

The related engineering work is end-to-end: The thesis spans all the way from problem formulation to system validation testing. Simply put, many topics will be covered. To structure these topics properly, this thesis is divided roughly into three parts: "why", "how", and "what". This was done to give it a logical progression, a narrative arc; by the time many of the design choices are up for discussion, the reasons that motivated these choices will already have been explained.

First, Part I (**"the Why"**) covers the motivation for the thesis work: what problems were identified that that prompted the research (chapter 2)? How is the research framed, and what were the research questions and goals (chapter 3)? How was the scope of the work justified, and how will the proposed research address the identified problems (chapter 4)? In summary, at the end of Part I the foundation for the rest of the research will have been established.

Part II of the thesis covers **the "How"**. At the end of the thesis, a design for a magnetic field simulation device for satellites is presented, but how should such a system be designed? What principles from physics are leveraged to achieve this (chapter 5)? At the location that this thesis was conducted, the Faculty of Aerospace Engineering of the TU Delft, there was already a Helmholtz cage setup present, which this new design has replaced. What did the old setup look like, and what were the associated problems that the new system should address (chapter 6)? How suitable was the local environment for building a sensitive magnetic simulation device, and what does this imply for the research (chapter 7)? These questions were investigated in the early phases of the thesis in order to ensure that all design requirements shaping the system would both be relevant and complete.

At the end of Part II, we will be ready to start discussing the implementation of the system. But developing a system that covers all possible magnetic testing is infeasible: it is not practical to implement, it is certainly to not easy or cheap to do, and it is downright impossible to fit into the scope of a single MSc thesis. Therefore in chapter 8, a **Demonstrator System** is defined, which is a smaller version of the magnetic testing system devised in Part I. This Demonstrator System (DS) will have enough features to be a useful tool for magnetic testing, but is small enough in scope to complete within the context of a MSc thesis. The DS will be designed with an eye to future expansion, so that additional system capabilities can be added later. Part II ends with the system requirements for the DS, which are needed before implementation can begin.

In Part III of the thesis, the implementation of the Demonstrator System is started and completed; roughly speaking, this is **the "What"** of the thesis work. For although the previous parts involved much research, by the end of the thesis, a detailed system design and very tangible implementation will be

delivered. What exactly was designed, and how was it structured? How do certain design choices flow from the requirements that motivated them? How can someone replicate the design implementation, or build upon it?

There is a lot of ground to cover to answer these questions, so the discussion of the implementation was split up across several chapters. A broad system overview and design approach can be found in chapter 9. In chapter 10 we cover the electronic design, chapter 11 covers the software development, and chapter 12 covers the system integration.

At the end of Part III, the verification and validation of the Demonstrator System is discussed. In chapter 13 we return to the system requirements from chapter 8 and formally verify them. In this same chapter, we put the Demonstrator System through its paces and validate its performance. Although the system works, it has some limitations, which are described at the end of chapter 13.

Finally, we conclude the thesis with chapter 14, where the research question is formally answered. This chapter also contains a reflection of the design process, discussing various lessons that were learned along the way. Finally, chapter 14 ends with a starting point for future research, identifying unsolved issues and offering suggestions for addressing them going forward.

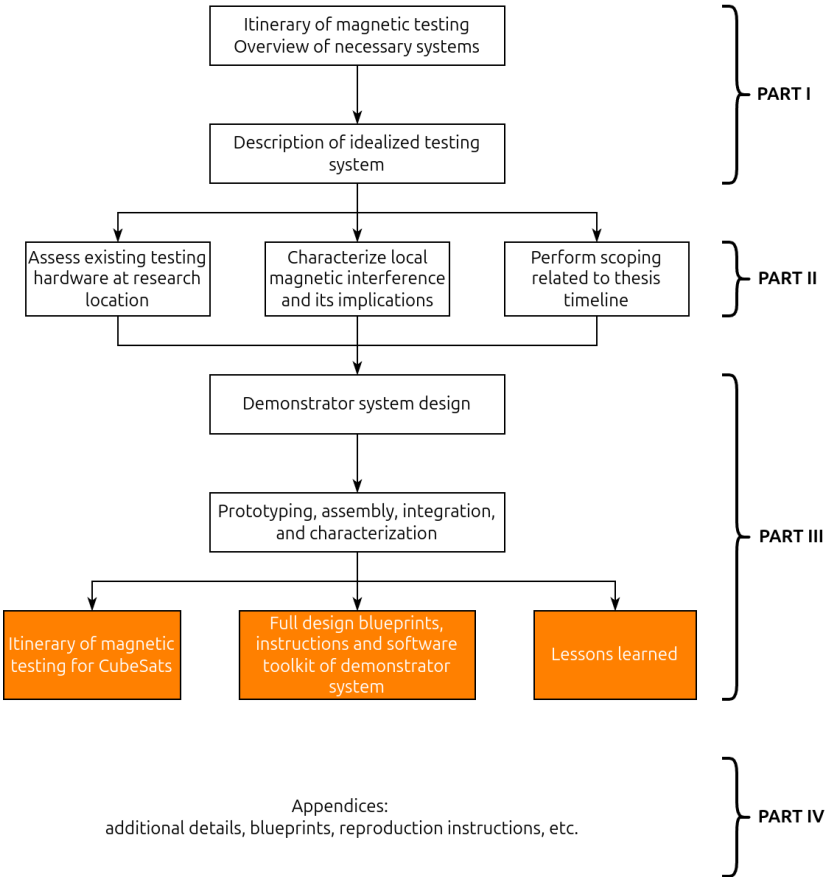


Figure 1.1: Diagram of the structure of this thesis.

.

The structure of the thesis can also be seen in Figure 1.1. Throughout the thesis, we will occasionally return to this figure to remind ourselves where we are.

This thesis is very long, as there is a lot of information to discuss. Parts I, II, and III can be considered the maintext, the **main body** of the thesis, and together they stand as a self-contained narrative. However, in order to avoid overloading this narrative with unnecessary details and excursions into side topics, many of these were moved to various appendices. Part IV contains these appendices, and for those who are interested, a lot more detailed information about the thesis work can be found here.

Part I

THE WHY - Problems, Root Causes, and Opportunities

2

On Satellite Failure

It is no secret that for a number of decades, nanosatellites have been used in more and more space applications. Although any spacecraft under 10 kilograms of mass is formally a nanosatellite, many of these nanosatellites are designed according to the CubeSat standard. According to Swartwout et al. [8], the year 2022 alone saw 290 such CubeSats being launched, with preceding years seeing similar numbers. Most of these CubeSats were launched as part of commercial or educational missions (see Figure 2.1).

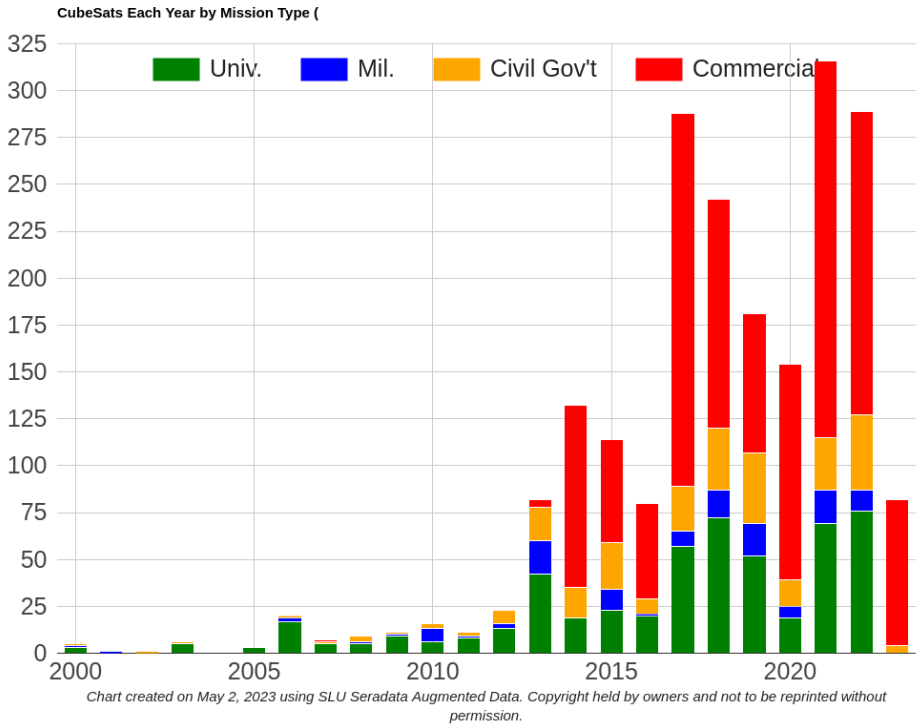


Figure 2.1: CubeSat missions launched to space, by year and by developer class [8]

It appears that CubeSats have an interesting future ahead of them. Whereas many of the missions displayed in Figure 2.1 have been used for low Earth orbit (LEO) applications, there are signs that these nanosatellites will play a key part in missions beyond Earth, both for lunar and martian destinations, as well as deep space exploration. Examples of this include Capstone [9] and LUMIO [10]. The established technological heritage of CubeSats, the ever-progressing miniaturization of electronics, as well as the

advances in swarming technology make CubeSats an interesting tool in space exploration not just for today, but also for tomorrow.



Figure 2.2: Artist depiction of the Capstone mission, an example of a lunar CubeSat mission. [9]

However, despite the popularity of CubeSats, missions involving them still have relatively high failure rates ([11], [12]). Opinions are divided on what the exact cause of this could be, and various suggestions have been made as to how to address this problem. Some studies indicate that more testing would be key in addressing this problem [12], [13] although Alanazi et al. [14] indicated that CubeSat projects that spend less time on testing were more likely to succeed. Further, [15] argued for improved testing over additional design redundancy as a better way to improve mission success. It appears therefore that to reduce failure rates in CubeSat missions, *"more effective testing"* is needed more so than simply *"more testing"*; quality over quantity. However, it is not obvious how current-day testing should be improved to become *"more effective"*.

In addition, many CubeSat projects have a relatively short development timeline and small budgets in comparison with larger satellite missions. Both of these factors can make it more difficult to test the satellite sufficiently to ensure a successful mission. As we will see later, there have been several CubeSat missions that experienced problems, but where one or more key tests that could have prevented the problems were not performed during development. One example of this is the PicSat satellite, which during its time in space was unable to point itself at the science target, as the team indicated that testing of its attitude determination and control system (ADCS) was abandoned due to time constraints [2]. In another example, the CANYVAL-X mission experienced issues with its ground station uplink, as certain communication testing was not performed owing to a lack of available parts [3].

In this chapter, the question of *"more effective"* testing for CubeSat missions will be investigated in more detail. Its results constitute the primary motivation for the rest of this thesis work.

In section 2.1, two approaches are used to attempt to investigate how CubeSats missions can fail, one based on theoretical speculation, and another based on case studies. In conclusion of this investigation, several avenues of further research are discovered. In chapter 3, a research plan is presented for further research into one of these avenues.

Throughout this chapter, special attention will be given to testing innovations that cause testing to be more attainable by development teams with a small budget and a tight schedule. Although it is likely that current-day tests can be improved as such, making it more likely that a certain test will be performed *at all* by a CubeSat team (as opposed to not) may prove equally important in reducing CubeSat mission failure in the future. We will revisit this point at the very end of this chapter, and it will remain a recurring theme throughout this thesis.

2.1. The Search for Root Causes

Given the high number of infant mortality cases and partial mission successes of CubeSat missions, it is of interest to find the most likely avenues for addressing this issue. It may be that there are "low-hanging fruits" that can be implemented relatively easily to significantly reduce the risk of mission failure for future CubeSat missions.

Previous literature has investigated organisational structure, design philosophy, subsystem failure rates, etc. to attempt to extract common root causes for the high failure rate. However, if improved testing is suggested as a potential solution, at least one of the root causes must be located in aspects of the satellite that can be tested for, such as hardware configuration. For this reason, it was investigated how the choice of hardware could lead to failure, and what tests would be necessary to weed these problems out during the testing campaign of a satellite mission.

Predicting different failure modes for even a small system such as a CubeSat can be difficult. One must not only take into account the ways in which the individual components can fail, but also what impact a failure of a component has on a larger subsystem, and what the effect is on the overall system. *Mission degradation* can quickly turn into *mission loss* depending on how different subsystems of a CubeSat interact with one another.

By way of example, consider the following: a malfunctioning ADCS can cause a satellite to start tumbling unexpectedly. The operators could attempt to fix the issue on-ground, but since many CubeSats have directional antennas, sending the fix to the satellite may become difficult if its attitude motion is disturbed. In fact, this was an important issue during the IT-SPINS CubeSat mission [16]. Moreover, if the power budget of the satellite relies on a certain attitude (pointing the solar panels at the sun), malfunction of the ADCS may mean that the power budget becomes negative and the battery may drain until the satellite shuts down forever. This may happen before the satellite operators are ever able to address the problem with the ADCS. For example, an issue such as this was observed with the MarCO-1 mission [17]. In conclusion, a minor failure of a subsystem can lead to total loss of the spacecraft merely by the way the satellite relies on its parts internally.

It is therefore important during testing that not only the subsystems themselves are tested thoroughly, but also the interaction between these subsystems in the context of the mission. However, this is difficult as these interactions are unique to a satellite, except perhaps multiple satellites that share the exact same hardware configuration, software, and mission profile. The issue therefore remains: the system engineer of a CubeSat mission will want to know what tests to run before the design is accepted as flight-ready. With limited resources available, how to prioritize such tests for the unique design in question?

Rather than relying on a generic, idealized test regiment that covers all possible tests, it may be possible to take an alternative approach: by looking at the properties of the satellite, and by learning from lessons learned from past mission failures, what if one could make a more informed decision about which tests to prioritize? This could serve as a basis for a more refined approach to testing that many future CubeSats missions can benefit from.

In order to investigate this possibility, two approaches were employed:

1. A bottom-up approach, based on hypothetical interference between subsystems based on the principles of physics.
2. A top-down approach, in which a survey of past CubeSat mission failures was made, to potentially extract commonalities from these cases.

2.1.1. Bottom-up Approach

In this approach, several common aspects of a CubeSat, here referred to as "subsystems" were listed. For each of these subsystems, it was investigated what aspects of the laws of physics each subsystem interacts with, and what aspects of the laws of physics could be a source for this subsystem to operate incorrectly. Thereafter, each subsystem was compared to every other subsystem, and potential interferences could be speculated upon. This can be seen in Table 2.1 and Figure 2.3.

Why was this approach used to investigate root causes? After all, the introduction of this chapter mentioned that there already has been different research into the problem of CubeSat failure. There are

several reasons for this. At the time, the author had worked on several satellite missions and had a relatively good idea of the typical subsystem division of many CubeSats. The idea was to make a "map" of common subsystems in satellites and identify potential sources of problems by looking at the way these systems could interact with one another. This was driven by the author's personal hypothesis about where some of the root causes of satellite failure could be hiding.

By way of illustration, let us consider two typical subsystems of a CubeSat; for example, the radio receiver and the power conditioning electronics downstream of the solar panels. In what ways do these systems interact that could lead to potential failure? Well, the receiver will be very sensitive to electromagnetic radiation within a certain frequency band, so that it is able to receive messages sent from far away. However, a local source of electromagnetic radiation from one of the other satellite subsystems may completely overwhelm the receiver, making it unable to distinguish incoming transmissions over the noise. Power conditioning electronics commonly may have inductive components, switching converters, and relatively high current lines, all of which may become risk factors in this regards. In this way, the electronics subsystem may negatively affect the telecommunication subsystem, even if it works otherwise correctly. In addition, any system that has digital communication or discrete clocks may become a potential source of problems, as the stark transitions of the internal square waves result in a rich spectrum of odd harmonics, any of which may be emitted and picked up by the transceiver.

These kinds of thoughts experiments were systematically repeated for each combination of subsystems, according to the author's knowledge at the time. The structure of a satellite may affect deployable subsystems by mechanically obstructing them. The same can be true for components that require line-of-sight, such as cameras or star trackers. The moving mass of a reaction wheel may vibrate enough to throw off the measurements of an inertial measurement unit (IMU). And so on.

The resulting "map" of Figure 2.3 is what resulted from this process. The subdivision of the subsystems may seem relatively arbitrary. It is based on the author's selection of smaller, self-contained systems that are commonly found across many of the CubeSat missions reviewed in the literature survey. Attention was paid to whether this subdivision also maps well onto the different physics principles that each subsystem relies upon, which it does to a large degree.

From Figure 2.3, a number of observations can be made. To begin, components with active deployables or those that require direct line-of-sight to the outside are liable to interfere with the structure of the satellite. Testing before launch should therefore ensure that the structure does not interfere with the satellite's operations.

Furthermore, many parts of a typical CubeSat can emit magnetic and electromagnetic noise. Several components in a satellite are especially sensitive to magnetic interference (ADCS) and electromagnetic interference (radio receivers and Global Navigation Satellite System (GNSS) receivers). The vast majority of the coloured cells in Figure 2.3 have to do with these issues. While this does not necessarily mean that these issues are more likely to occur in a real-world scenario, it does mean that there are many more *different ways* in which problems of this nature may manifest. A good test during the satellite development could be a test that simultaneously covers as many of these scenarios as possible. This is because testing for each interaction separately is likely impractical, especially for development teams on a tight schedule and a tight budget.

There are several other subsystem interferences that are restricted to only a single cell, suggesting that although they may not necessarily be easy to test for, they should be of a relatively predictable nature.

	Structure	Solar panel	Solar panel deployment mechanism	Power conditioning (MPPT, etc.)	Power pack	Antenna structure	Antenna deployment mechanism	Radio TX	Radio RX	GNSS	Thrusters	Magnetorquers	Reaction wheels	IMUs	Magnetometers	Sun sensor	Horizon sensor	Star tracker	Radiation sensor	Optical camera	Thermal management	On-board Computer	Data buses
Structure			1				1				1				4a	1	1	1		1			
Solar panel									2f	2f					4b								
Solar panel deployment mechanism									2f	2f					4c								
Power conditioning (MPPT, etc.)									2a, 2b	2a, 2b					4c								
Power pack									2a, 2b	2a, 2b					4c								
Antenna structure																							
Antenna deployment mechanism									2a	2a					4c								
Radio TX									2e	2e					4c								
Radio RX										2a	2a, 2c	2a	2a	2a	2a	2a	2a	2a	2a	2a	2f	2a, 2d	2f
GNSS										3	2a, 2c	2a	2a	2a	2a	2a	2a	2a	2a	2a	2f	2a, 2d	2f
Thrusters															4c, 4d								
Magnetorquers												5			4c, 4g								
Reaction wheels														6	4c, 4e								
IMUs																							
Magnetometers																						4f	
Sun sensor																							
Horizon sensor																							
Star tracker																							
Radiation sensor																							
Optical camera																							
Thermal management																							
On-board Computer																							
Data buses																							7

Figure 2.3: Bottom-up approach: Breakdown of likely subsystem interference based on laws of physics

Table 2.1: Clarification of the items in Figure 2.3

#	**Interference description**
1	Structure causing clearance issues
2	RF emissions from device interfering with radio receiver:
a	Anything with bus communication: solar panel deployment, power conditioning+pack, antenna deployment, thrusters, magnetotorquers, reaction wheels, IMUs, MTMs, sun sensor, horizon sensor, star tracker, radiation sensor, optical camera, GNSS receiver, OBS. Solar panels and thermal management have been excluded from this list, as these likely contain no communication equipment (thermal management on CubeSat missions seems thus far to be restricted to passive systems.)
b	Power conditioning: MPPTs, DC-DC converters, and large capacitors may have RF emissions that may interfere with the communication receiver and/or GNSS receivers.
c	Thrusters may produce RF interference, particularly ion propulsion [18], [19]
d	Small computers can emit significant RF emissions (see for example baseline Raspberry Pi RF emissions [20])
e	Radio transmitter feeding back into receiver
f	RF interference from miscellaneous electronic sources
3	Multiple GNSS devices interfering with one another
4	Magnetic interference to magnetometers:
a	Magnetic dipole of spacecraft structure (can change over time due to electrostatic charge buildup from environmental plasma and electric propulsion [21])
b	Net magnetic dipole due to solar panel layout (changes with incoming solar flux and direction)
c	Magnetic fields from high current cables (changes with current load on cables)
d	Magnetic fields from thrusters (notably electric propulsion, heating coils, and/or valves)
e	Magnetic fields of operating reaction wheels (increases with wheel rpm)
f	Unknown amount of magnetic field emission from computer
g	Net magnetic dipole of magnetorquers (changes depending on operation duty cycle)
5	Magnetorquers interfering internally (one torquer coil inducing current in others)
6	Reaction wheels microvibrations affecting IMU readings
7	Cross-bus interference

Limitations of the bottom-up approach

It should be mentioned that the discussed bottom-up approach has some important limitations, which hold back its usefulness. To begin, this analysis is completely theoretical and was made from speculation based on the laws of physics and the general theory of operation of different satellite subsystems. The advantage of this that one would be able to anticipate problems before there is an example of it happening in the reality; the analysis does not need to "wait" for particular faults to occur during a mission before being able to anticipate them. On the other hand, it is entirely unclear to what extent the analysis actually represents reality. Furthermore, there is no guarantee that the list of *known unknowns* in Table 2.1 is complete or even comprehensive. There may be many more ways in which subsystems can interact that are not captured in the analysis, and currently these remain *unknown unknowns*.

For these reasons, this analysis would be much improved if it could be anchored in reality in some way, for example by examining historical examples of CubeSat failure and see whether the coloured cells in Figure 2.3 are indeed represented in these examples. This is not to say that the analysis could not be improved by better theoretical groundwork, for example by a more comprehensive survey of parts common to CubeSats, and simulations of their behaviours and interactions.

2.1.2. Top-down Approach

In an attempt to improve the analysis of subsection 2.1.1, an additional *top-down approach* was used. A large number of problematic satellite missions from the past were investigated to see if any lessons could be learned from them. In this analysis 54 CubeSat missions from the past were reviewed. The aim was to find precedent that the problems speculated upon in subsection 2.1.1 do indeed occur in reality, and so an attempt was made to place each of the 54 missions inside Figure 2.3.

It quickly became clear that for hardly any of these missions sufficient information was available to conclude where to place them. For many missions, very little information could be found in literature to begin with. The only public information available about some missions could only be found in archived web pages or old Twitter posts, without more detailed publication being available. Indeed, in some cases it seems as though there had been a concerted effort to hide the information, presumably out of embarrassment or similar reasons. As a result, several CubeSat missions that were going to be part of the analysis were removed, as there was nothing meaningful to conclude about them. For those that were included, it was assessed whether the available information was of sufficient quality; where necessary a larger margin of error was used and occasionally things items were explicitly marked as "speculative".

For many of the missions where detailed information was available, the conclusion by the authors could often be summarized as: *"We will never know the exact reasons, but here are our most likely hypotheses"*. This is simply to be expected, as there is very little to conclude in such cases, for example when no telemetry is available. Radiation-related problems or software issues were frequently mentioned as theories, due to the number of ways in which they could interfere with satellite operation in ways that would leave no traces.

Finally, there were failed CubeSat missions for which one of more satellite subsystems were identified as the culprit. Although these were the most useful, the meaning of what constitutes a "subsystem" rarely matched that of the one used in the analysis of subsection 2.1.1. This is no surprise, given that the subdivision used there was somewhat arbitrary.

Improving the bottom-up approach of subsection 2.1.1 proved more of a challenge than expected because of these issues. In response, several changes were made to the top-down analysis to ensure that useful information could still be extracted from the case studies. Whilst this concession meant that the relevance to the bottom-up approach would be reduced, it seemed there was no other way to introduce real-world data into an otherwise entirely theoretical analysis. The changes that were made were as follows:

- The list of different "subsystems" was reduced from 23 to 7, now including *structure*, *solar panels*, *electrical power systems* (EPS), *communication systems* (COMMS), *on-board computers* (OBS), *attitude determination and control systems* (ADCS) and a generic item for *propulsion systems*, whether these be for attitude or trajectory control.
- A distinction was made between whether an issue merely caused setbacks during a mission, or

whether it proved fatal to a mission, as the latter is obviously more important to prevent, and therefore more pertinent to the problem of reducing CubeSat failure. Of the issues that caused fatal failures, a distinction was made whether the root cause was known/certain according to published information, or whether the root cause was speculative. This distinction allows one to treat the *"known root cause"* cases more seriously, and it also allows for the inclusion of multiple *"speculative root cause"* cases without making the *"known root cause"* cases ambiguous. In this way, the analysis includes some sense of the *"level of severity"* of an issue.

- Three extra categories were made for *"orphans"*: reasons for failure that do not necessarily involve hardware. A generic category was made for radiation-related problems, as these can occur nearly anywhere in a satellite where there are electronics. Another category was made for software issues, as such issues are often clearly distinct from issues with the hardware itself. Finally, there is a miscellaneous category that included all other reasons.

The resulting CubeSat cases that were included in the analysis can be found in Figure 2.4. The 54 cases were mapped onto the table in the following way: in cases where a satellite failed because of a single subsystem, it was placed in the left-most column. If issues arose because of multiple subsystems interacting with each other, it was placed further right in the table. Finally, as mentioned before there are the three extra orphan categories below the table. Colours were used to distinguish between the three *"levels of severity"*. Each data point contains a number, each of which refers to a certain CubeSat mission from the analysis. An exhaustive list with details about each data point is available Appendix L and the original literature study report [22].

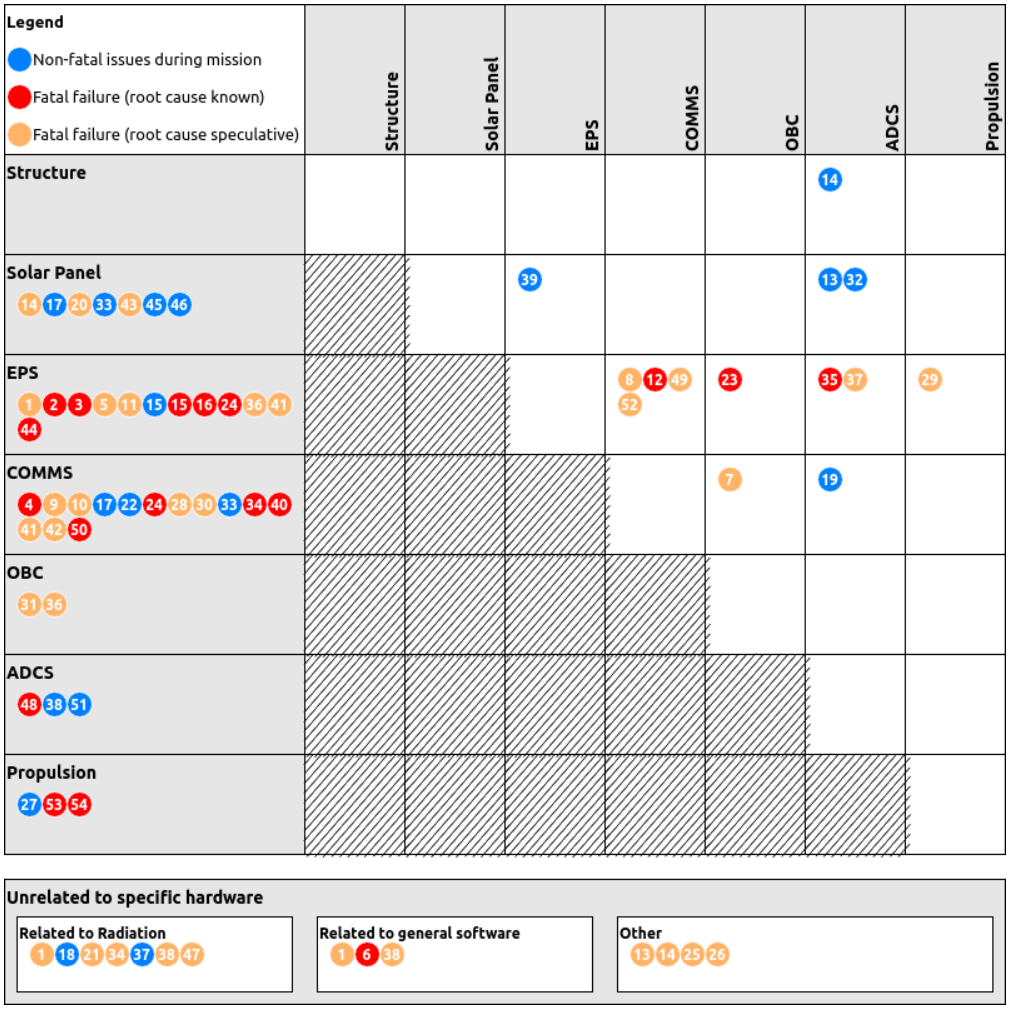


Figure 2.4: Top-down approach: Collecting the reported issues of 54 CubeSat missions and mapping them onto subsystems. A full list with individual references can be found in Appendix L and [22]

.

A number of observations can be made about Figure 2.4. It should be mentioned beforehand that each of these observations is subject to the limitations of the reported data, whose quality varied substantially.

First, it appears that the majority of identified root causes are **limited to one subsystem only**, as indicated by the number of entries in the left-most column. The communication system was mentioned most frequently, followed by the electrical power system and the solar panels. It should also be mentioned that the vast majority of red entries are here. This may be an indication that most CubeSats fail because of one subsystem only, suggesting that their designs may have been improved with a better fail-safe design. However, it may also simply mean that it is much easier *to be certain about malfunction* with one subsystem as root cause for the failure than for two interacting subsystems. In any case, the analysis does not provide a definitive answer to that particular question. If anything, it indicates how heavily dependent CubeSats are on their power system and communication system.

Looking further to the right in Figure 2.4, we can observe the cases where more than one subsystems were implicated. Judging by the EPS row, there are a substantial number of cases for where the cause was a interactions between the EPS and another subsystem. For some of these cases, (8, BRICSat-P and 12, CXBN), an underperformance of the power supply severely degraded the performance of the communication system. In another case (23, Kraksat), the satellite's OBS entered a perpetual boot loop.

Focusing now on the column associated with the ADCS, we can observe missions where the ADCS and another subsystem caused trouble. In the case of 14 (ESTCube-1), the satellite structure became sufficiently magnetized to hamper attitude control in the early segment of the mission. For other missions (13, DeOrbitSail and 32, MOVE-II), the solar panel design exhibited strong enough magnetic field emissions that it caused trouble during the mission. Satellites may have difficulty closing their power budget if the satellites rotation rates are unintentionally high (35, Omotenashi), or when they require the (defective) ADCS to sun-point the solar panels (37, Phoenix CubeSat).

Finally, looking at the bottom of Figure 2.4, we see that the orphan categories contain mostly yellow points. This seems to suggest that these are frequently implicated as possible root cause for failure when the actual root cause is not known. Or it may simply mean that it is harder to prove these as definitive root causes, which at least in the case of radiation-related issues seems plausible.

Limitations of the top-down approach

The top-down approach of subsection 2.1.2 has some glaring issues: First, it is difficult to prove how statistically significant its findings may be, owing to the ambiguity and variety of quality of the source material. It is difficult to quantify error bars or correlation coefficients when the data points themselves have to be interpreted from such limited information. What can be said is that "only" 54 CubeSats were analysed, and it would stand to reason that more case studies would improve the results, and potentially reveal more patterns in the data.

A significant number of missions were excluded from the research. In general, a mission was only included if information about possible root causes was available either from formally published/cited information or directly from primary sources: for this reason, information gained from the Twitter account of one of the satellite development engineers would be included over a well-written journalistic web article without visible citations. However, in many cases it was simply the nearly complete lack of information that would get a mission excluded from the test data.

Moreover, because of the modifications made to the process of analysis, the bottom-up approach of subsection 2.1.1 and the top-down approach of subsection 2.1.2 no longer map onto each other. The author concludes from this that there is not enough knowledge available to verify the model presented in subsection 2.1.1. As a result, it is difficult to directly conclude how CubeSat testing can be improved to reveal the largest amount of potential issues. However, what is revealed is where there are important gaps in knowledge, and therefore where there are interesting avenues of further research. This will be continued in the next section.

2.2. Conclusion

As mentioned before, the two approaches used before have limited correspondence to each other, because there was not enough data available to link the two together exactly. Therefore, when it comes to identifying common root causes for CubeSat mission failure that could be tested for, there is a significant

gap between theoretical analyses and real-world data. The real-world data is limited, and hence at the current time, the theoretical models cannot be verified with supporting real-life data.

What the analysis did reveal is several interesting avenues of research that may be key in reducing the rate of CubeSat failures.

1. As Figure 2.4 suggests, many problems that prove fatal to a CubeSat are exclusively caused by the power system or the communication system. If this is accurate, then further research into improving the reliability of these system could improve the reliability of CubeSats as a whole. But moreover, it could also be investigated whether a different design philosophy could be chosen for the overall satellite that ensure that these systems are not single points of failure. It may be the case there are conditions under which it is worth adding redundancy to the communication system.
2. Looking at the bottom left of Figure 2.4, there were also a number of satellites for which it was suspected that radiation was responsible for problems, but in none of the cases this could be confirmed for certain. Often the hypotheses surrounding such radiation-related root causes are supported by the notion that a satellite became unresponsive when passing over the South Atlantic Anomaly (see for example the NanoSatC-BR2 mission [23]) or during periods of heightened solar activity (see INSPIRE-II mission [24]). To shed better light on problems during such missions in the future, there may be better ways to detect such failures aboard the satellite. Better software algorithms or hardware layouts may be able to detect radiation-related events and correct/report these. Or, it may be chosen to equip a satellite with a small dosimeter near sensitive equipment, whose values could periodically broadcasted by an emergency beacon.
3. The analysis of subsection 2.1.1 shows that there are many ways in which electromagnetic interference (EMI) can interfere with the operation of a CubeSat, particularly by deafening the radio receiver and any on-board GNSS receivers. Given that precise modelling and simulation of the electromagnetic emissions of every component in even a small system is difficult, better testing could better classify a satellite's electromagnetic profile. Such tests would have to be performed ideally with the satellite operating as representatively of real operation as possible.
4. Both Figure 2.3 and Figure 2.4 show that there are several ways in which static magnetic fields can interfere with the operation of a CubeSat, particularly with the ADCS. For example, in the ESTCube-1 mission, the structure of the satellite became magnetized, causing it to behave like a compass needle in the Earth magnetic field and interfering with the ADCS operation [25]. In the DeOrbitSail mission, the solar panels were designed in such a way that they would continuously impart a magnetic moment during operation, causing the satellite to spin up over time. Before the mission begins, magnetic tests should weed out such problematic magnetic peculiarities of a satellite.

An additional conclusion that has yet gone unmentioned is that while working on the case studies, it was observed that for many CubeSat missions, pre-mission testing was not always comprehensive. In some cases, the satellite in question failed for reasons that could very plausibly be avoided if certain tests were not skipped. Testing of the ADCS stood out in particular in this regard, for example for the PicSat mission [2], the IT-SPINS mission [16], and the SwissCube mission [26] [27].

This is an interesting conclusion all by itself. At the start of this chapter we discussed the question of *"quality over quantity"* in regards to CubeSat testing, but if no test is performed in the first place, the question becomes a moot point. Instead, a more appropriate response may be: "could the current methods of testing a CubeSat be made more accessible, or performed at lower costs, so that this results in a higher likelihood of these tests being performed **at all**?". Many of these CubeSat missions are produced on a small budget and with a relatively rapid development time. It is easy to think about improving satellite testing in terms of metrics such as to sensor accuracy, resolution, etcetera; for example: "how much better is this test if the resolution is increased by a factor of two?". But this presumes that the test is performed at all. A different approach may be to investigate how much more effective the exact same test may be if its cost is a factor of two lower. This in itself may be an interesting topic of further research.

3

Research Goals and Scope

The previous chapter concluded with a data analysis of a number of satellite failure cases, and from this data a few interesting avenues for further research were revealed. For example, it seems that improving the reliability of a satellite's electrical power system (EPS) or telecommunication system would be a worthwhile goal, given the propensity with which these subsystems are associated with problems during the mission. Issues related to the attitude determination and control system (ADCS) were also identified as an interesting topic for further investigation, particularly related to magnetic interference. In short, there are enough leads to follow here to fuel several research projects; far too much for a single thesis, so a choice must be made as to which lead to follow.

The goal of this chapter is to outline how the conclusions from the previous chapter lead to the choice of topic for this thesis, and to establish the research questions and research goals. Furthermore, the scope of the thesis is explained.

3.1. Motivation and choice of topic

From an early point onwards, the topic of magnetic testing of satellites stood out to the author as particularly interesting. Given the literature study that was performed, there is precedence to consider this an interesting topic for further research, but there were three additional reasons why this topic stood out.

First, it is the realization that was mentioned at the conclusion of the previous chapter; it may be case that improving the quality of testing for future CubeSat missions actually involves making these tests more accessible to developers, for example by lowering their cost or improving the development schedule. Given some of the case studies, it seems that this conclusion is particularly relevant for ADCS testing.

Secondly, there is another factor to consider for future CubeSat and other small satellites, which is miniaturization of space technology. When the first CubeSats began appearing in space, these satellites were already much smaller than the average satellite. CubeSats have now established themselves more firmly as useful and versatile tools for applications in space. But there is a further push for miniaturization, as both off-the-shelf components for CubeSats become smaller, and other very small satellite standards make their way onto the stage. Examples of this include the PocketQube standard [28], whose units are 1/8th the size of CubeSat units, and satellites-on-a-chip, which further reduce the size of a satellite to a single printed circuit board (PCB) [29].

Shrinking the size of electronic systems has a disproportionately large effect on its susceptibility to magnetic and electromagnetic interference. With smaller distances between electronic components, the interference will be stronger. For static magnetic fields, we can show this using the Biot-Savart law:

$$\overline{B} = \frac{\mu_0}{4\pi} \int_C \frac{I d\overline{\ell} \times \overline{r}'}{|\overline{r}'|^3} \tag{3.1}$$

From this can be seen that for some magnetic field, halving the distance to the source ($\overline{r}$) causes an eight-fold increase in the absolute magnitude ($|\overline{B}|$) of the magnetic field. In other words, reducing the size of a satellite, given that the currents flowing through the electronics remain the same, will cause worsen potential interference issues.

But also for effects of electromagnetic radiation, for which electronic components essentially behave like small antennas, are not linearly proportional in regards to distance, but rather more like the inverse-square law. For applications such as consumer electronics or small drones, electromagnetic compatibility (EMC) is an essential aspect of design (see [30], [31]). It is no surprise that there has been increasing demand for research into areas such as electromagnetic shielding [32]. If space technology is further miniaturized in the future, electromagnetic interference may at some point become the main limiting factor.

Finally, it so happened that at the faculty of Aerospace Engineering, there was an old magnetic test setup available. This test setup was no longer in use due to its age and limited applicability. It therefore lent itself as a good opportunity to investigate how satellite testing could be improved, especially if parts of this setup could be re-used, cutting down on time and cost during the thesis.

To summarize, the topic of magnetic testing of small satellites was selected as the topic for the research, for four main reasons:

1. Given the outcome of the literature review of chapter 2, it appears to be a topic worth investigating.
2. There is good reason to believe that magnetic testing of nanosatellites may become more crucial in the future, given breakthroughs in spacecraft miniaturization.
3. The topic is of particular interest to the author.
4. At the location of the research, there is pre-existing hardware that may be able to help with the research and accelerate progress.

3.2. Thesis Objectives

It is at this point that the goal of the thesis can be formulated. We have a rough idea of what kinds of magnetic testing would be useful to perform on a developed spacecraft, given the failure modes from the case studies in chapter 2, and we will soon consider pre-existing specifications such as European Cooperation for Space Standardization (ECSS). We have also seen that the problem with missions on a tight time or monetary budget is not always that the tests were not good enough, but that they were not performed at all.

3.2.1. Research Question

With this context as a backdrop, we now formulate the research question of this thesis as follows:

"How can the magnetic properties of a small satellite be effectively tested?"

To elaborate on this, as a subtext to this research question there will be four main areas of interest that will be investigated:

1. How can the satellite's magnetic properties be tested such that the testing is **representative** for satellite? In other words, which environmental conditions can and should be simulated in a laboratory environment such that the tests measure something meaningful about the satellite's operations after its mission starts?
2. How can the satellite's magnetic properties be tested such that its results are **reliable**? The performed tests should be repeatable, such that they can be repeated, and the same results can be expected. In addition, conditions in the testing environment must be kept consistent, such that repeated tests are not influenced by changes in the environment.
3. How can the satellite's magnetic properties be tested in a manner that is **time-efficient**? To improve chances of mission success, testing must be improved. But the chances that a given test will in actuality be performed as part of a test campaign is influenced by how quickly such a test can be performed. Therefore, this research should pay attention to ensuring that the tests can be

performed in a way that is time-efficient, and preferably with a predictable and consistent time schedule.

4. How can the satellite's magnetic properties be tested in a manner that is **cost-effective**? In addition to the aforementioned, this research should also pay attention to ensuring that the tests can be performed with a low enough cost such that they can realistically fit within a typical CubeSat testing budget.

3.2.2. Research Objectives

The research objective attempts to capture the main goal of the project. For this research, the objective is as follows:

"To improve testing for CubeSat missions, especially those heavily constrained by schedule and budget, by improving and streamlining a testing strategy for the magnetic properties of a CubeSat."

with the obvious intention of improving overall reliability of future CubeSat missions. As discussed earlier in this report, the matter of constrained budgets and schedules is brought up for the following reason: Whereas innovations into the effectiveness of current-day testing would have an impact in reducing the failure rate of CubeSats, it has been observed on several occasions that development teams will forego certain tests for lack of time or resources. The author reasons that innovations that make current-day more accessible to development teams, for example by reducing their cost, or make them more efficient, will decrease the likelihood that development teams will choose to skip these test. In this way, it has the potential to reduce CubeSat failure rates also.

3.3. Research plan

To answer the research question, an overall research plan is needed. This section will describe this research plan.

1. First, it is required to investigate existing magnetic testing strategies for CubeSats. This includes both what exact properties of the CubeSats should be tested, as well as typical testing setups for these tests.
2. With this itinerary of satellite tests, an idealized testing system will be described. This system is one that performs the complete itinerary of tests, if at all possible.
3. It will quickly become clear that implementing this idealized testing system is not feasible. First, it is much too large in scope to actually develop within the scope of a thesis. Secondly, environmental factors must also be considered, such as magnetic interference at the test location, available research budget, etc. Finally, it is very likely that such a comprehensive system is not likely to be the most effective solution for the problem in question; as discussed, the cost and development time of such a system would in many cases be prohibitive, resulting in these tests not being performed at all.
4. For this reason, the thesis will contain within its scope the design and construction of a **Demonstrator System** (DS), which is a cut-down version of the idealized testing system. It will inherit some of the requirements of the idealized testing system, but it will be much smaller in scope, and some of the surrounding considerations may be different. The reason for this demonstrator system is to help answer the research question and aid in developing an example of a test system that implements the four sub-aspects of the research question.
5. The demonstrator system will be built partially out of components used for an older satellite testing system, already present at the research location. To assess how this can best be done, and also to improve on the system, this system will first be investigated in detail.
6. It is known that the research location is not an ideal location for sensitive magnetic testing, due to the presence of ambient magnetic interference. The nature of this interference will be measured and characterized. This will provide a rough idea of what one can expect of the performance of the demonstrator system, but also teach some valuable lessons about performing magnetic measurements in a non-ideal environment.

7. After this is done, the demonstrator system will be designed using a standard system engineering approach. It will subsequently be developed, constructed, integrated, and tested.
8. Finally, we return to the research question and conclude the research.

If this research plan is followed through, the results of the research are as follows:

1. An itinerary of magnetic testing for CubeSats.
2. Lessons learned from assessing and working around environmental magnetic interference.
3. An open-source, end-to-end design of a magnetic testing system, which can be copied and modified as needed by satellite developers. This includes any software written as part of this system.
4. Lessons learned from the design and implementation, as well as recommendations for further improvement.

The aforementioned is summarized in the flow diagram of Figure 3.1.

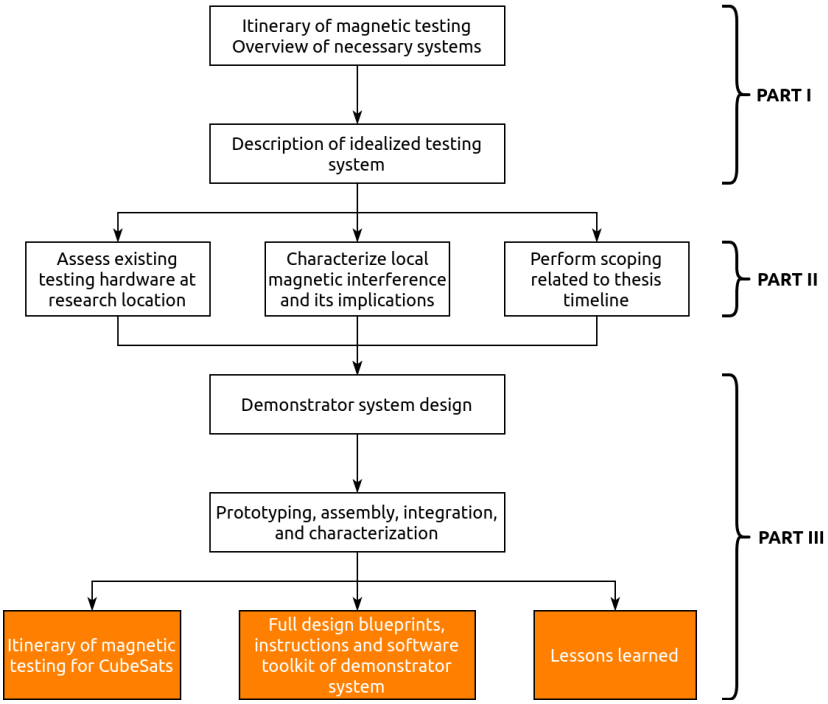


Figure 3.1: Summary of the research plan as a flow diagram, with the results in the coloured boxes at the bottom. The partitioning on the right indicates how these topics are divided across this thesis.

.

The rest of the thesis, up to the appendices, is set up to communicate these contents to the reader. It is written in such a way as to deliver an implementation-ready design to any potentially interested CubeSat developers. Given that the tests under discussion are in practice not always performed, one can think about this as an *"argument by way of engineering"*: If you are a satellite developer who is on a tight schedule, a tight budget, and are on the fence about whether to include or skip certain magnetic tests, then chapter 2 is an argument in favour of performing the test, and the rest of the thesis should serve as an argument why and how these tests can fit in your schedule and your budget. In other words, this thesis is written with the implicit goal of providing a design with which CubeSat developers already have a head start with their testing campaign.

4

Magnetic Testing of Nanosatellites

The previous chapter explained why the intention of this thesis work is magnetic testing of nanosatellites. Before we can move on, we must first take a closer look at what kind of tests that relate to magnetism are typically performed during the development of a nanosatellite. How can we improve if we do not yet know where things currently stand?

In this chapter, we investigate magnetic testing for nanosatellites, and devise a kind of idealized testing system, which could in theory perform most or all of these tests. It is important to do this, for it forms a foundation underneath some of the later sections of this thesis.

4.1. Magnetic testing for CubeSats in literature

There is a solid body of research available on the magnetic testing of CubeSats. Descriptions of such tests are frequently included in published works performed by universities and research institutes. In many cases, the tests are performed using self-built equipment (see for example [33], [34], [35], [36]). Commercial solutions are also available, for example the one depicted in Figure 4.1, although these may be expensive and beyond the budget of many CubeSat missions. As mentioned before, in some cases magnetic testing is not performed due to a shortage of resources or time (e.g. [2]). In addition, there have been investigations into design strategies for improving *magnetic cleanliness*, both for the satellite itself and for any test equipment (see [37] and [38]).



Figure 4.1: Example of a commercial hardware-in-the-loop (HWIL) test setup including a Helmholtz cage (Astrofein, see [39])

.



Figure 4.2: University-built Helmholtz cage for testing Delfi-C3 [33]

.

The European Space Agency (ESA) lists many development guidelines in its standardization documents, authored by the European Cooperation for Space Standardization (ECSS). Of particular interest here is ECSS-E-ST-20-07C [40], which in section 4.2.5 lists recommendations for a magnetic cleanliness plan for a spacecraft under development. Section 5.4.5 of the same document contains guidelines for characterizing the direct current (DC) magnetic field emissions of a spacecraft in development, as well as its magnetic moment. Furthermore, section 5.4.10 of this standard also covers testing a spacecraft's ability to withstand external magnetic emissions, from low frequency to high frequency. Finally, ECSS-E-ST-10-03C [41] places these tests in the broader context of a spacecraft's recommended testing regiment.

4.2. An itinerary of magnetic testing for nanosatellites

After going through these resources, one can develop a general idea of what magnetic aspects of the satellite should be tested. In this section, we take a look at a prospective itinerary of magnetic tests for a CubeSat developer who wishes to thoroughly test their satellite:

1. Characterization of the spacecraft's net magnetic dipole

That is, the far-field DC emission of the satellite. We distinguish here between the static dipole and the dynamic dipole. The static dipole is constant regardless of the spacecraft operational conditions, whereas the dynamic dipole may change as a function of time, mission segment, satellite attitude, etc. A constant magnetic dipole may impart a constant torque on the spacecraft in the presence of an external magnetic field, such as the Earth magnetic field (EMF). In space, constant forces and torques are always a point of attention, as without air friction, any unwanted momentum tends to compound.

The characteristics of the dynamic dipole highly depends on the design properties and mission design of the spacecraft, but in combination with the environment, they may contain positive feedback loops create motion that interfere with operations. An example is a magnetic dipole generated by a solar panel as soon as the panel is rotated towards the sun. If the effect of this dipole in conjunction with the EMF is to rotate the spacecraft in such a way that the panel rotates away from the sun, the spacecraft will constantly have to overcome this torque during its mission, which can have significant effects on the momentum and power budgets of the overall mission. An illustration of this issue can be seen in Figure 4.3.

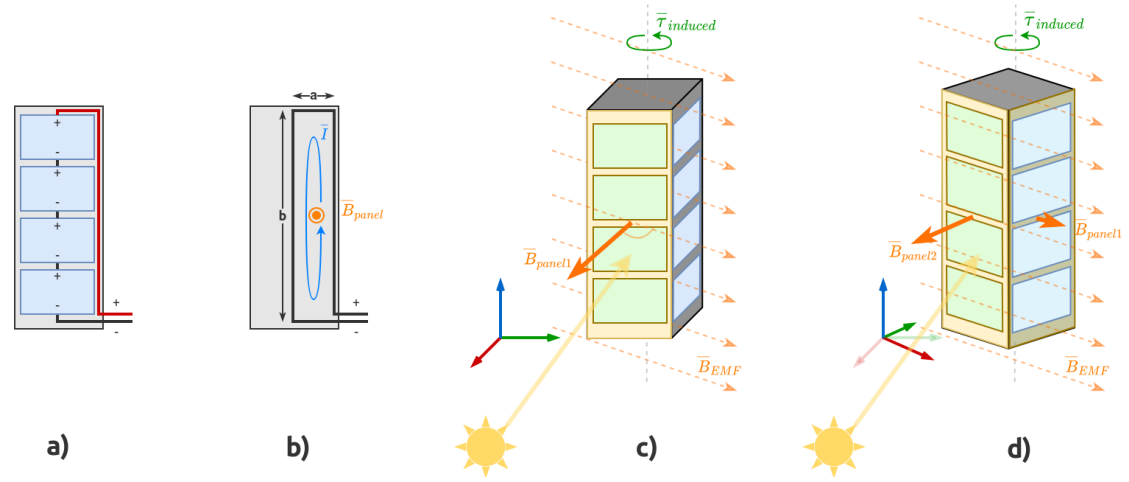


Figure 4.3: An example of an unintended magnetic dipole inherent in a solar panel design and its consequences on a satellite mission. The solar panel in a) is designed in such a way that its operation induces current inside a wire loop. If the solar panel is illuminated, it will generate a perpendicular magnetic field vector $\overline{B}$ in proportion to the current $\overline{I}$ and the area ab of the loop. In c), one can see that this illumination-dependent dipole can lead to unintended torque during the satellite mission, caused by angle between the local Earth magnetic field and the panel field, leading to rotation. Even if such torques are in the order of μNm, if they are allowed to compound in space they can lead to unsalvagable mission failures, especially for satellites with low mass moment of inertia and a constrained budget for attitude control.

The main reason we distinguish between the static dipole and the dynamic dipole is because the implications that measuring either has for the testing setup: The static dipole can be measured rather straightforwardly in a magnetically neutral environment and an appropriate magnetic field probe setup, such as a six-sensor configuration as specified in section 5.4.5.4 of ECSS-E-ST-20-07C [40]. However, characterizing the net magnetic dipole as a function of the spacecraft's operational conditions requires a test setup that can simulate the mission environment to a certain degree. This may necessitate among other things the simulation of sunlight using a solar simulator, or giving the spacecraft one or more degrees of freedom (DOFs). In summary, it generally demands that the test equipment is of a higher complexity.

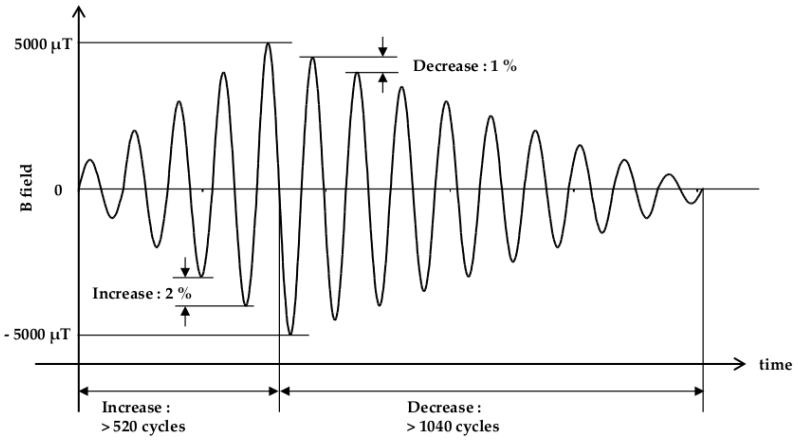


Figure 4.4: Example of a deperming procedure from [40].

.

Note that for this test, a deperming procedure is usually recommended, for example as outlined in section 5.4.5.3 of ECSS-E-ST-20-07C [40] (see Figure 4.4). Such a procedure will rid the metallic components of the spacecraft under test of most permanent magnetization effects. Note the vertical axis

in Figure 4.4; deperming requires magnetic field strengths several orders of magnitude above those typically used to simulate the EMF, implying that one would use a separate device for deperming.

2. Characterization of the spacecraft's near-field properties
The aim of this type of test is generally to characterize the magnetic compatibility with itself. An example where this is relevant is with on-board magnetometers intended to measure environmental magnetic fields. The readings of this sensor can be thrown off by a nearby magnet, such as an electromagnet, reaction wheel, or a nearby high current PCB trace. Induced currents within electronics as a result of nearby changing magnetic fields can also interfere with operations.

Due to the implications of the inverted cube law exhibited by magnetic field strength, as well as the small size of CubeSats, one can generally be confident that most of the satellite body of a CubeSat is subject to near-field effects. It is inherently difficult to simulate the multitude of interferences possible within the spacecraft due to the amount of detailed information required; many of these details are so specific that they will typically be known by the time the satellite is ready to physically test anyway. A verification test is typically the only way to be sure that these effects will not negatively impact the spacecraft during its mission. The quality of these tests are generally dependent on the accuracy with which one can simulate the spacecraft operation during its mission.

3. Characterization of the spacecraft resistance to AC emissions
This type of test will assess whether the spacecraft under test can withstand AC magnetic fields, meaning incoming field emissions that periodically change. An example of the procedure of such a test can be found in section 5.4.10 of ECSS-E-ST-20-07C [40], which applies to field frequencies between 30 Hz and 100 kHz. Such a test involves a calibrated source of radiated magnetic fields, a magnetic sensor for monitoring, and current probes. While it is not strictly necessary to perform this test in a magnetically neutral environment, given that any harmonic content under 30 Hz can be filtered from the measurement data using a high-pass filter, it may be useful to do so to avoid sensor saturation and prevent paramagnetic effects from interfering with the measurement.

4. Verification of any magnetic sensing equipment on the satellite
This one is relatively straightforward: to verify that any magnetometers on-board the satellite are operating correctly, one has to apply a controlled reference field to the satellite, for example with a Helmholtz cage. A magnetically neutral environment is useful here, as it will negate the effect of ambient fields and allow for a greater degree of control over the applied reference field.

Note that not only static fields are of interests here, but changing fields as well. A common use for magnetometers aboard CubeSats is the determination of the axial rates of the satellite. If the satellite during its mission is rotating so fast that it cannot be resolved due to a slow polling rate on the sensor, the satellite cannot assess its own rotation rates. In this case, the sensors may be operating perfectly within specification, but the overall system cannot perform its function. Such a defect should also be weeded out by the aforementioned test, and so testing with a rotating magnetic field can be of great use.

5. Verification of any magnetic actuators on the satellite
The performance of any magnetic actuators also has to be assessed. Given the small size of CubeSats, solar panel areas are usually relatively small, resulting in a tightly-constrained power budget. Magnetorquers in essence turn electrical power into a magnetic field output, and therefore a torque output. They do this with a certain efficiency, and the actual efficiency of such a system has to be tested beforehand.

Testing these things in practice can be quite difficult with a simple setup. Arguably the simplest way to perform this test is by suspending the satellite with a thin wire, allowing for single DOF rotation, and then simulating the magnetic field conditions during the mission and activating the actuator. This test has to be repeated for each axis separately, assuming that this is feasible. More elaborate testing setups allow more degrees of freedom, typically through the use of an air-bearing.

6. General operation validation
Beyond testing certain physical characteristics and individual subsystems, a well-integrated test system will also allow for testing multiple subsystems at once. This may reveal design flaws that are not always visible with a simpler test. For example: when the solar panels are illuminated, how do the panels

magnetically interact with the other subsystems? But also: how does the rest of the power system downstream of the panels interact with the other systems? What about the maximum power point trackers (MPPTs) or the battery system? Are there any surprises on the magnetic spectrum when these systems are put in the context in which they are intended to operate? Such integrated testing imply a degree of centralized control of the testing system.

4.3. An idealized magnetic testing system for nanosatellites

With the context of the previous section, as well as the considerations of the research project from chapter 3, we can now consider an idealized testing system. In essence we are asking: if we could command an infinite amount of development time and budget to build the perfect magnetic testing system for a CubeSat, what would this system look like? What would its requirements be? In this discussion, we bar considerations around simulating zero-g and vacuum environments, given their limited applicability to the points discussed in section 4.2, as well as the difficulty to realize this in combination with the other aspects of testing.

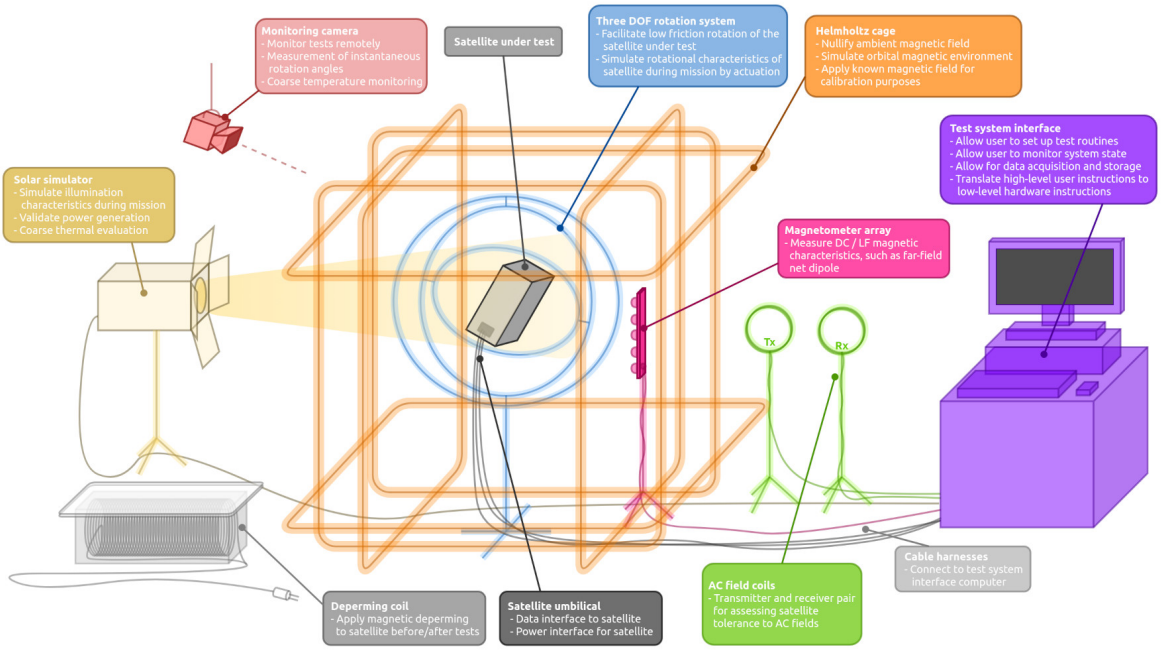


Figure 4.5: Idealized testing system that can perform all magnetic tests described in section 4.2

.

In summary of the previous section, Figure 4.5 shows an example test system that covers the tests that were discussed. In the middle, we have a satellite model that is to be tested. It is suspended inside a contraption that can rotate it along multiple degrees of freedom, such that its relative angle can be varied. A solar simulator lamp illuminates the satellite from one direction, allowing for hardware-in-the-loop (HWIL) evaluation of the power systems of the satellite. The test setup is surrounded by a large Helmholtz cage, ensuring that at the satellite's location, the local earth magnetic field can be nullified, and any mission-relevant magnetic field envelopes can be simulated. In conjunction with the solar lamp and the freedom to rotate, this can also reveal induced rotation issues, such as those described in Figure 4.3.

Besides this, there is more magnetic testing equipment, such as a pair of AC field coils, one for transmitting and one for receiving high frequency magnetic fields, for the tests described in section 5.4.10 of ECSS-E-ST-20-07C [40]. There is also an array of magnetometers that can be placed at an arbitrary distance from the satellite to map its emitted DC fields. It is important that this array can be moved, because it is typically not clear beforehand at what distance from the satellite the far-field effects start to dominate over the near-field effects. Finally, on the bottom left of Figure 4.5 there is a deperming

device so that most of the latent magnetic field properties of the structural elements can be purged before any testing.

4.4. Requirements for the idealized system

Given this general idea of an "idealized" testing system, we can now derive some more formalized requirements for it. The following table lists these requirements.

ID	Description	Verif.
MTS-01-02	**The magnetic testing system (MTS) shall have a magnetic control volume large enough to envelop a typical CubeSat. The size of this control volume shall be a cube with side lengths of at least 415 mm.** The required size is derived from the diagonal of a 12U (2Ux2Ux3U) CubeSat, allowing it to rotate in any direction whilst remaining fully enveloped by the control volume. Verification: Use a calibrated tri-axial magnetometer to measure the resulting field inside the activated Helmholtz cage.	**T**
MTS-02-02	**The MTS shall establish a neutral magnetic environment, reducing any ambient magnetic fields inside the control volume down to below 1 μT in any direction.** Verification: Same as MTS-01.	**T**
MTS-03-02	**Inside the control volume, the magnetic field strength uniformity shall be within 1%.** Verification: Either through simulation/analysis of the performance of the chosen coil configuration, or repeated measurements inside the control volume of the finished Helmholtz cage.	**A, T**
MTS-04-02	**Inside the control volume, the magnetic field directional uniformity shall be within 1%.** Verification: Either through simulation/analysis and calculating the local divergence and curl of the field, or repeated measurements inside the control volume of the finished Helmholtz cage.	**A, T**
MTS-05-01	**The MTS shall allow the satellite to be suspended inside the control volume.**	**D**
MTS-06-01	**The MTS shall allow the satellite to freely rotate around its three rotational axes.** Verification: calibration test	**D, T**
MTS-07-01	**The MTS shall be able to measure the satellite's instantaneous rotational angle inside the testing area to within 1°.** Verification: calibration test	**D, T**
MTS-08-01	**The MTS shall have the capability to rotate the satellite with an accuracy of 1°.**	**D, T**
MTS-09-01	**The MTS shall have the capability to rotate the satellite with a speed of 5 rpm.** For a typical nanosatellite, an unintended rotation rate of 1 rpm is already quite high, so a system target of 5 rpm will allow testing for edge cases up to five times higher.	**D, T**
MTS-10-01	**The MTS shall allow for illumination of the control volume with an intensity of 1500 W/m^2 and a spectral profile similar to in-orbit sunlight.** Designed to be a decent margin above expected solar flux at low Earth orbit (LEO), which is 1367 W/m^2. Verification: Same as MTS-01.	**T**
MTS-11-01	**The MTS shall be capable of measuring the far-field magnetic properties of satellite to an accuracy of 500 nT.** The required precision somewhat depends on the mission requirements and the design of the satellite. For most CubeSats, 500 nT may be a good baseline to target, as this is about two orders of magnitude of the Earth magnetic field, although for missions with very specific requirements or small torque budgets, this may be lowered to 10-100 nT as needed.	**D, T**

MTS-12-01	**The MTS shall be capable of emitting alternating magnetic fields with frequencies up to 100 kHz.** The required strength of this field generally depends on the mission profile, but ECSS-E-ST-20-07C [40] recommends targeting 10 dB above the intrasystem electromagnetic compatibility (EMC) requirements. Typical values of this are frequency dependent, and a graph with recommended data can be found in appendix A.13 of this document.	D, T
MTS-13-01	**The MTS shall be capable of emitting a deperming routine for properly demagnetizing test articles. Generally, this consists of a slowly alternating field with strengths up to 5000 μT.** For details and recommendations, see [40].	D, T
MTS-14-01	**The MTS shall be able to operate automatically according to a pre-programmed schedule, without requiring manual user input during testing.** This important in order to greatly improve the user experience and increase the repeatability of the tests.	T
MTS-15-01	**The MTS shall have a straightforward and convenient interface, suitable to be usable by space engineering students.**	T
MTS-16-01	**The MTS shall be safe to operate, by users unfamiliar with the inner workings of the system and at the level of engineering students.** For example, this implies that a user who is unfamiliar with the inner workings of the system can perform tests without being at risk of receiving electrical shocks, starting a fire, etc.	T
MTS-17-02	**The MTS shall facilitate time-efficient testing, allowing for tests to be performed in the span of half a day, wherever feasible.** As a rough target, it would be good if a satellite can be characterized in the span of a few hours (i.e. one afternoon session).	D
MTS-18-01	**The total cost of the MTS shall not exceed more than 10% of the total mission budget.** This is a broad requirement whose feasibility highly depends on the specifics of the mission, project structure, and typical launch costs at the time. As such, this requirement can be tightened as needed.	A
MTS-19-02	**The MTS shall be composed of commercially-off-the-shelf (COTS) components with high availability wherever feasible.**	D

4.5. Conclusion to Part I

It is now time to briefly pause and look at where we are. We return to the research plan of Figure 3.1.

Having zoomed in on the topic of satellite testing, we now re-emerge with a broad idea of what a suitable magnetic testing system for a nanosatellite may look like. But working out the finer details of such a design and building it completely is far too great a task to fit in the scope of a thesis. Instead, as mentioned previously, a demonstrator system will be designed, which will contain a subset of the functionality of the idealized system discussed up to this point. Before this can be done, it is time to first look at a number of other topics, which have relevance to this task. This section therefore marks the end of Part I of the thesis, and we are about to head into Part II, which covers these other topics. We return to the finer details of the demonstrator system at the start of Part III.

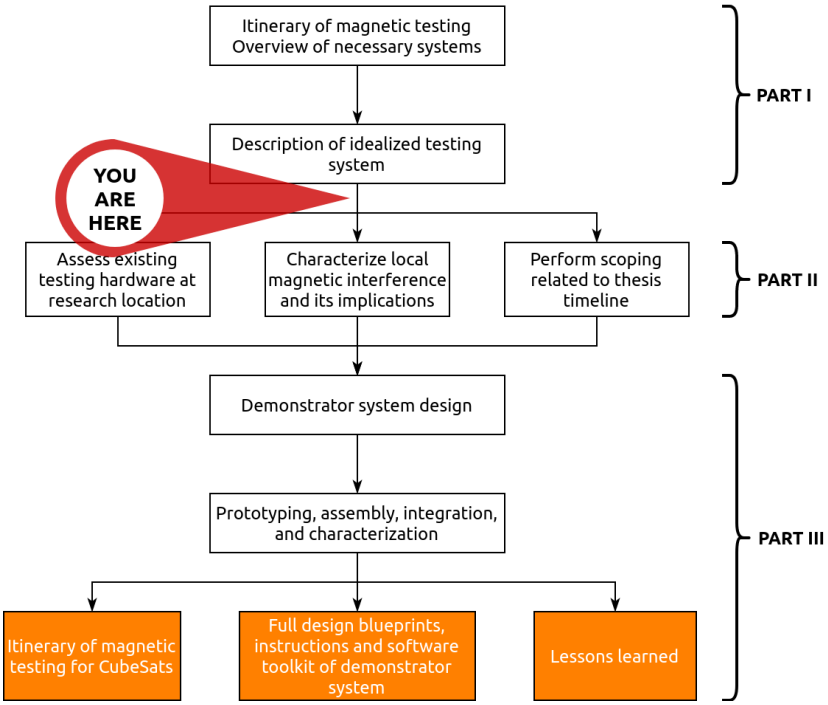


Figure 4.6: Another look at Figure 3.1.

.

Part II

THE HOW - Physics, Requirements, and the Past

5

Helmholtz Cage Theory

This thesis will cover the topic of the Helmholtz cage at length. This is a device that can precisely establish magnetic fields with a known strength and direction. In this chapter, we take a brief look at the theory of operation and underlying physics of this device, as this will become important in later chapters. A note before we start: The strength of a magnetic field can be expressed in several ways. We will concern ourselves only with the magnetic flux density $\overline{B}$:

Convention
In this thesis, the magnetic flux density $\overline{B}$ **will be used synonymously with "field strength"**. The more common definition of "field strength" expressed as the $\overline{H}$-field will be left aside.

The rationale to only use one of these two quantities is to keep discussions on magnetic fields a bit more straightforward. The flux density quantity can be intuitively thought of as how densely the magnetic field lines are concentrated at a certain location.

5.1. Fundamentals

To understand how a Helmholtz cage work, we start from fundamental physics. In 1820, Hans Christian Ørsted discovered that electric currents can produce magnetic fields [42]. In the years thereafter, Ampère and Sturgeon, among others, developed an early understanding of the principles of electromagnetics, which was further refined by Maxwell a few decades later.

The phenomenon that Ørsted described is depicted in Figure 5.1a. When we take a length of conductive wire and force a current through it, the movement of the charge carriers (such as electrons) inside the wire will induce a net magnetic field around the wire. If the wire is straight and presumed to be infinitely long, then the field around the wire will be homogeneously circular.

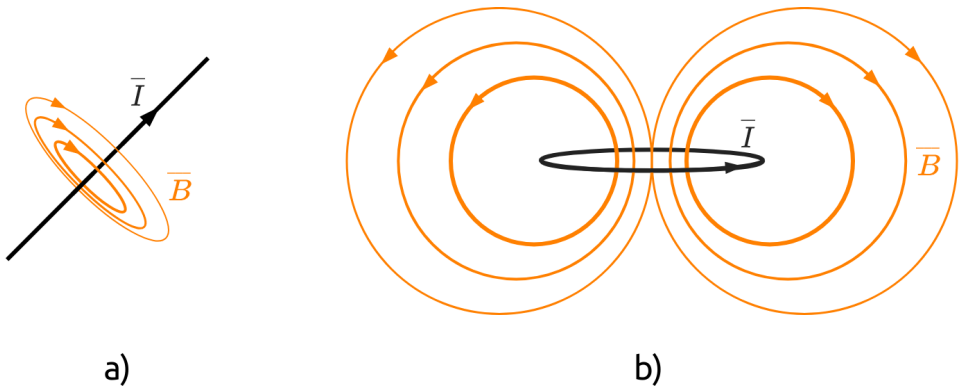


Figure 5.1: a) The flow of current through a wire will induce a magnetic field around it. b) When the wire is bent into a circular shape, a loop, a current through it will induce a toroidal magnetic field.

For the straight wire, the magnetic field will be the strongest near the wire, close to where the charge carriers are moving. As one moves farther away from the wire, the strength of the field will be attenuated more and more. The nature of this attenuation can be described by what we now refer to as the Biot-Savart Law:

$$\overline{B} = \frac{\mu_0}{4\pi} \int_C \frac{I d\overline{\ell} \times \overline{r}'}{|\overline{r}'|^3} \tag{5.1}$$

This equation describes the magnetic field induced by a current carrying wire segment. We can see that the attenuation of the magnetic field strength is inversely proportional to the cube of the distance r, meaning that when you increase the distance from the wire of Figure 5.1a by a factor of two, the local magnetic field strength will have been attenuated by a factor of eight.

Now consider Figure 5.1b, where the length of wire has been bent around into a circular loop. A current through this loop will create a toroidal magnetic field around it, and because of the nature of this shape, the magnetic field lines will be more concentrated inside the loop than on the outside. In practice, electromagnets rely on this principle, typically using many loops of wire to produce a coil. Each loop of the coil will contribute to the effect visualized in Figure 5.1b and as a result, such coils can be used to produce toroidal fields of significant strengths. Note that by convention, the direction of the magnetic field is right-handed with respect to the current, and in this thesis, we will use this convention without exception.

Now we take this principle one step further by taking two identical, circular coils, aligning their axes, and placing them at a distance from each other. By applying current to these coils in the same direction, they will each produce a toroidal magnetic field in the same direction. The net magnetic field at any location around these coils will be the sum of the contributions of both coils, and the two fields can be thought of as "blending together". As we move the coils farther away from each other, their composite effect becomes weaker.

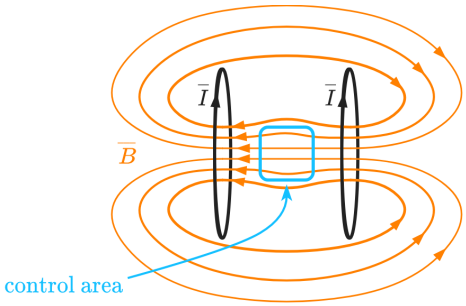


Figure 5.2: Placing two identical coils at a distance equal to their radius results in a so-called Helmholtz coil pair.

If we place these coils at a distance that is equal to the radius of the coil, we create what is known as a **Helmholtz coil pair**, named after Hermann von Helmholtz. This can be seen in Figure 5.2. Observe that in the region between the two coils, the magnetic field lines are quite straight and homogeneous. Let us refer to this as the 'control area' for now. A special property of the Helmholtz cage is that the distance between the coils is chosen such that the magnetic field lines at the control area are maximally homogeneous.

For a pair of circular coils, it can be shown that this ideal distance is exactly equal to the radius of the coils (half its diameter). For square coils, the mathematics are a bit more involved due to geometric anomalies presented by the corners, but can be shown to be approximately 0.5445 times the side length of the coil [43]. If the coil separation is decreased relative to the ideal case, higher flux densities can be achieved for a given current, but this will also increase the divergence of the field lines at the control area. Increasing the coil separation can somewhat decrease the divergence at the control volume, but the absolute field strength will be less constant. This can also be seen in Figure 5.3. Note the red contour, which is the magnetic field through two square coils, but at the non-ideal spacing of 0.5 times the coil length. The magnetic flux density that results between these coils is constant over a much smaller region than for the "ideal" value of 0.5445 times the coil length, which is depicted in yellow.

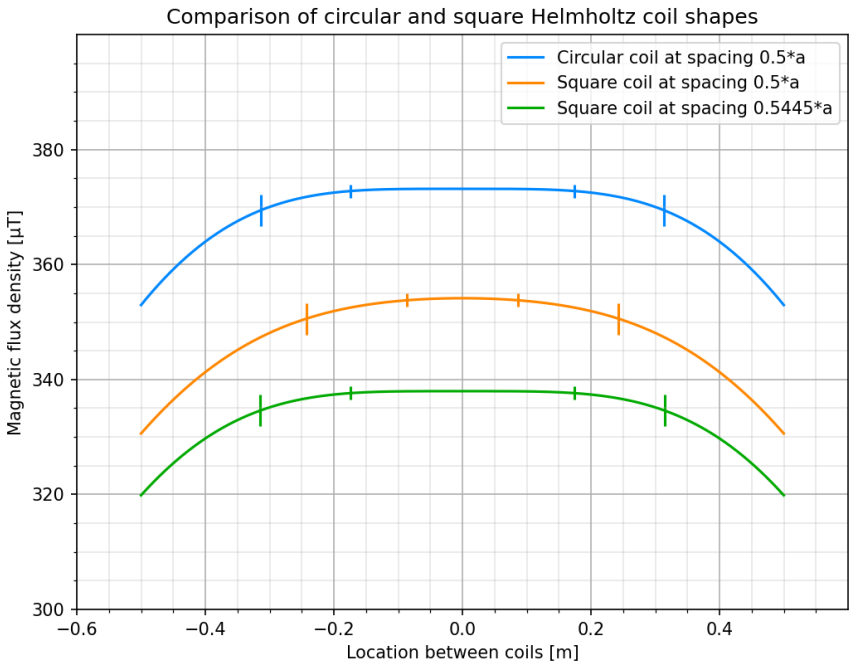


Figure 5.3: Simulation of the magnetic field strength on the axis between two coils. Blue represents an ideal circular Helmholtz coil pair. Red is a square Helmholtz pair with the same spacing, showing a response with lesser flatness in the middle. Yellow is the square coil at the ideal spacing for square coils, namely 0.5445 times the coil side length.

5.2. Coil properties

It will be useful to discuss some properties of electromagnetic coils in preparation for some of the later chapters. The kinds of coils that we will be focusing on will have some qualities in common. First, they will be thin coils, or "short coils": that is, their radius far exceeds the length of the coil. In addition, they will all be essentially large air-core inductors, and given that the magnetic permeability of air is very near that of free vacuum [44], these coils will exhibit near-linear behaviour without significant hysteresis.

5.2.1. Inductance

For the inductance of each coil, we can turn to the work of Wheeler [45], which describes approximations for the inductance of short coils of both circular and square shape. In general, the inductance of a coil equals

$$L = K\left(\frac{\mu_0 n^2 \pi a^2}{b}\right) \tag{5.2}$$

where a and b are the radius and axial length of the coil respectively. In addition, n is the number of coil windings, and μ_0 is the (constant) the permeability of free space. Finally, we have K, which is an inductance reduction coefficient that corrects for the shape of the coil. Deriving an appropriate value of K is not straightforward, so we use the final result from Wheeler [45] for a circular short coil:

$$L = \mu_0 n^2 a\left[\ln\left(1+\pi\frac{a}{b}\right)+\frac{1}{2.3+1.6b/a+0.44(b/a)^2}\right] \tag{5.3}$$

For a square short coil with side length $2a$, the inductance is:

$$L = \mu_0 n^2 a(4/\pi)\left[\ln\left(1+\pi\frac{a}{b}\right)+\frac{1}{3.64+2b/a+0.51(b/a)^2}\right] \tag{5.4}$$

Neither equation yields exact values, but both should have an accuracy better than 0.1% [45].

5.2.2. Magnetic field strength

The absolute magnetic field strength that a coil will output when supplied with current can usually be calculated quite straightforwardly by something resembling:

$$B = \frac{\mu_0 n I}{\ell} \tag{5.5}$$

where μ_0 is once again the permeability of free space, n the number of windings, I the current, and ℓ is the axial coil length. However, it is much more interesting to have an expression for the field strength in terms of the geometric distance from the coil. Frix et al. [43] provide such expressions for Helmholtz coil pairs.

This method relies on the computation of the elliptical integrals of the first and second kinds, which are defined as:

$$K(k) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 sin^2\theta}} \tag{5.6}$$

$$E(k) = \int_0^{\pi/2} \sqrt{1 - k^2 sin^2\theta} d\theta \tag{5.7}$$

For a circular coil pair, the magnetic field in the axial direction of the coil pair equals:

$$B_z = \frac{\mu_0 n I}{2\pi} \sum_{i=1}^{2} \frac{1}{\sqrt{(a+\rho)^2 + (z - z_i)^2}} \cdot \left[K(k_i) + E(k_i) \frac{a^2 - \rho^2 - (z - z_i)^2}{(a-\rho)^2 + (z - z_i)^2} \right] \tag{5.8}$$

with the k_i terms being:

$$k_i = \sqrt{\frac{4a\rho}{(a+\rho)^2 + (z - z_i)^2}} \tag{5.9}$$

and:

$$\rho = \sqrt{x^2 + y^2} \tag{5.10}$$

Equation 5.8 looks complicated, but is really just Equation 5.5 with a number of elaborate geometrical corrections tacked on. The parameter z here denotes the axial distance from the plane of the coil. To properly calculate use these equations for a Helmholtz coil pair, z_1 and z_2 have to be set equal to $a/2$ and $-a/2$ respectively, which will ensure that the distance between the coils is appropriate for a Helmholtz coil pair. We may observe that the magnetic field output is still directly proportional to the coil current.

Frix et al [43] also provides expressions for a square coil pair, each with side length $2a$. The flux density of such a coil pair equals:

$$B_z = \frac{\mu_0 n I}{4\pi} \cdot \sum_{i=1}^{2} \sum_{j=1}^{2} (-1)^{i+j} \cdot$$
$$\sum_{k=1}^{2} \left(\frac{(x - x_i)(y - y_j)}{\sqrt{(x - x_i)^2 + (y - y_j)^2 + (z - z_k)^2}} \cdot \left[\frac{1}{(x - x_i)^2 + (z - z_k)^2} + \frac{1}{(y - y_j)^2 + (z - z_k)^2} \right] \right)$$

This relation can also be rewritten to solve for the current I inside the coil:

$$I_{coil} = \frac{1}{n} \cdot \frac{4\pi}{\mu_0} \cdot \frac{B_{req}}{Q} \tag{5.11}$$

with:

$$Q = \sum_{i=1}^{2}\sum_{j=1}^{2}(-1)^{i+j}\cdot$$
$$\sum_{k=1}^{2}\left(\frac{(x-x_i)(y-y_j)}{\sqrt{(x-x_i)^2+(y-y_j)^2+(z-z_k)^2}}\cdot\left[\frac{1}{(x-x_i)^2+(z-z_k)^2}+\frac{1}{(y-y_j)^2+(z-z_k)^2}\right]\right)$$

and:

- $x_1 = a \quad , \quad x_2 = -a$
- $y_1 = a \quad , \quad y_2 = -a$
- $z_1 = 0.544505643 \cdot a \quad , \quad z_2 = -0.544505643 \cdot a$

Anywhere in this thesis where it is stated that the field strength was calculated, this method was used.

5.3. The Helmholtz Cage

With the context of the previous sections, it is very straightforward to understand the Helmholtz cage. It is simply an arrangement of three Helmholtz coil pairs, all placed perpendicular to one another. By applying a specific control current through each pair of coils, one can accurately modify the magnetic field inside the control area, which now has become a *control volume*. An example can be seen in Figure 5.4, which in fact depicts the dimensions of the Helmholtz cage at the faculty of Aerospace Engineering of TU Delft (AE), which will be formally introduced in chapter 6.

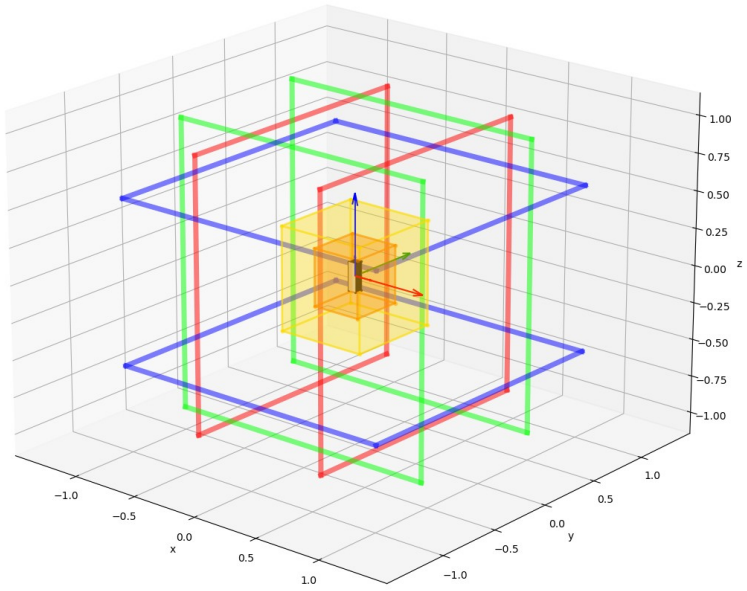


Figure 5.4: 3D visualization of a Helmholtz cage, with three pairs of square coils. A 3U CubeSat is placed in the middle for scale. The two yellow cuboids represent the control volume in which the field strength should vary no more than 1% (larger cube), and 0.1% (smaller cube).

In context of satellite testing, Helmholtz cages can be very useful. First of all, they can be used to cancel the local Earth magnetic field (EMF) at the test location, allowing a satellite to be tested in a magnetically neutral environment. This can be particularly useful for characterizing the magnetic emission profile of a satellite, verify the sensitivity of magnetic sensors, as well as specific mission segment testing for interplanetary missions. In addition, the Helmholtz cage can establish magnetic fields of its own allowing for the simulation of in-orbit conditions over time, so long as the corresponding control currents can be supplied to the coils.

6

Overview of the old Helmholtz cage system

Back in section 3.1, the motivation for the topic for this thesis was discussed. Part of this motivation was because at the faculty of Aerospace Engineering, there was hardware available that could be used for related thesis work. This refers to an older Helmholtz cage system, which is located in the cleanroom on the 8th floor of the Aerospace Engineering faculty building at the TU Delft. In the past, it was used to perform magnetic tests on satellites, such as *Delfi-C3*, *Delfi-Next*, and *Delfi-PQ*. On June 28th, 2023, this Helmholtz cage system was partially disassembled, pending future upgrades. This chapter contains the results of a detailed examination of the Helmholtz cage setup circa June 2023. The actual hardware has been examined, but the theoretical performance has also been investigated. Not only is this relevant for the sake of context and having a reference to compare future upgrades to, but we will see that it comes with some important implications and requirements for the Demonstrator System (DS) that is yet to be built.

We will refer to this system as the **old Helmholtz cage system** to distinguish from the resulting system after the work described in this thesis. Before any modifications could be made to this old system, its properties had to be evaluated. The reason is two-fold: First, it is necessary to analyse the setup in order to determine how it could be improved upon. The Helmholtz cage setup was not used regularly, and its interface computer had in fact already been disconnected from the faculty network. The documentation on the setup was dated and very limited, and it appears that much of the knowledge on how the device is used effectively left the university along with the creators of the device.

The second reason for evaluating the old Helmholtz cage in detail is to create a reference benchmark, so that future upgrades could be evaluated against it. Much of the control electronics of the Helmholtz cage setup were taken apart on June 28th, 2023, with the expectation that these would not be reassembled in the same way again. However, the documentation in this chapter is written with the guiding idea that the Helmholtz cage could in theory be reassembled in its original state. Even before the disassembly, it was no longer possible to activate Helmholtz cage itself, as the computer had been disconnected from the network by the faculty's IT department. As such, this chapter fills many of the gaps in documentation, and where it is not possible to measure the performance of the setup in practice, to estimate the performance of the setup using analysis.

6.1. Overview of the old Helmholtz Cage system

The Helmholtz cage in the cleanroom of the Aerospace Engineering faculty was built in the latter half of 2006 [33]. It was built primarily to test the Passive Magnetic Attitude control System (PMAS) of the Delfi-C3 nanosatellite. The Helmholtz cage has three pairs of square coils, each of which is attached to aluminium structural elements. Square coils were chosen over circular coils for their better size performance, and because they were easier to manufacture and suspend than circular coils. The size of the coils was chosen primarily to allow Delfi-C3 to rotate freely inside the cage with its antennas

deployed, which requires more than a meter of space. The coil size was ostensibly limited by the size of the door leading into the cleanroom.



Figure 6.1: A picture of the Helmholtz cage setup in the cleanroom. The storage rack on the left and the grey closet on the right are not part of the system. Inside the Helmholtz cage there is the yellow-topped *"grid table"*. On the right there is the burgundy-coloured supply cart.

The Helmholtz cage setup also contains a small test stand, or "grid table" on which test articles can be placed accurately. Finally, the cage is accompanied by a small cart, which contains the power supplies powering the coils, the supporting electronics, and the control computer. All three, cage, table, and supply cart, are visible in Figure 6.1.

6.1.1. Coordinate system

Before moving onwards, let's discuss coordinate systems. It is unclear what coordinate system was used by the original creators of the Helmholtz cage setup. During the analysis of the old Helmholtz cage system, a temporary coordinate system was used to refer the position of the Helmholtz cage to the rest of the Earth. In this chapter we will use this coordinate system, and since the Helmholtz cage was not repositioned over the course of this thesis, the reference frame definition for the Helmholtz cage will also be used in Part III of this thesis.

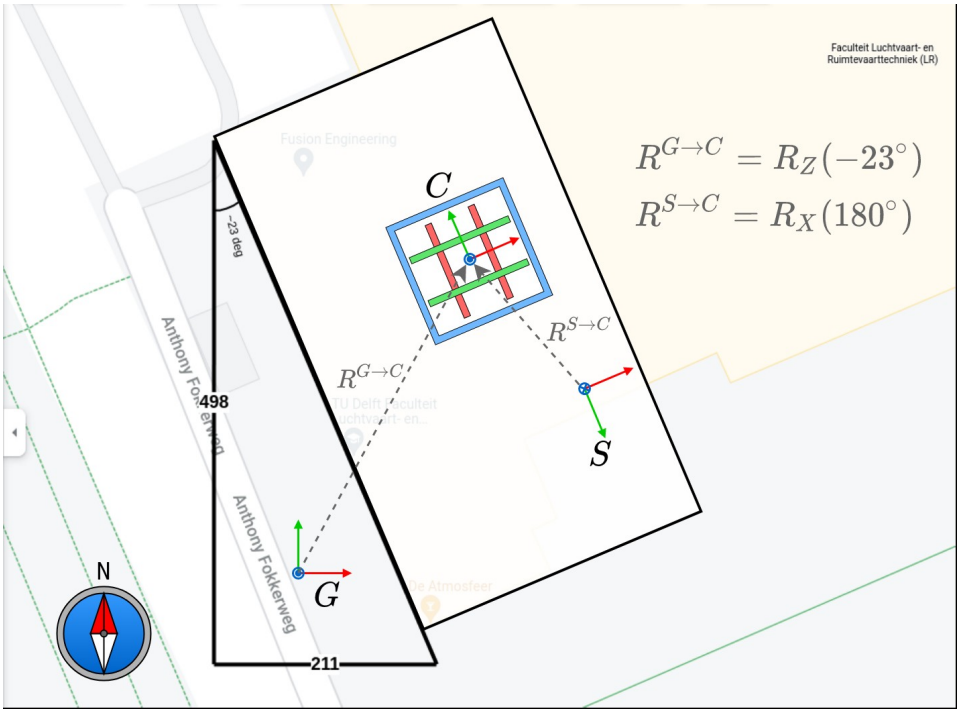


Figure 6.2: The coordinate systems used in this chapter. Cage-centered system C is preferred above the others, so any information in the geographic system G or sensor system S will be converted to the C frame. In both cases, this can be achieved through a single-axis (Euler matrix) rotation, as depicted.

Magnetic field strengths are expressed as vector quantities, and in order to express them a common coordinate system must be adopted. The coordinate convention depicted in Figure 6.2 uses a *"cage-centered"* coordinate system C as a main system. The system has its origin at the center of the Helmholtz cage, and is chosen such that the common axis of the smallest coil pair corresponds to the X-axis. The Z-axis is attached to the common axis of the largest coil pair, and points upwards.

System G is used to express vector quantities that are usually expressed in a geographic coordinate system, such as the Earth magnetic field (EMF) vector. G has its X-axis pointing due East, its Y-axis pointing due North, and its Z-axis pointing upwards. For this reason, it is an East-North-Up (ENU) frame. As can be seen in Figure 6.2, vector quantities in G can be rotated to C with a single-axis rotation:

$$R_{G\to C} = R_Z(-23°) = \begin{bmatrix} \cos(-23°) & -\sin(-23°) & 0 \\ \sin(-23°) & \cos(-23°) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(23°) & \sin(23°) & 0 \\ -\sin(23°) & \cos(23°) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

In addition, Figure 6.2 depicts an auxiliary sensor system S that can be used to express sensor measurement data, which then have to be converted to the C frame also. The definition of S is not that important in the greater context of this thesis, but was important to use consistently when any measurements where taken with a tri-axial magnetometer, which has its own coordinate system.

6.1.2. Internal system layout

Figure 6.3 shows a simplified system diagram of the setup. The supply cart contains a number of components to ensure that the Helmholtz cage setup can work correctly. The Helmholtz cage has three pairs of coils, one pair for each cardinal direction X, Y, and Z.

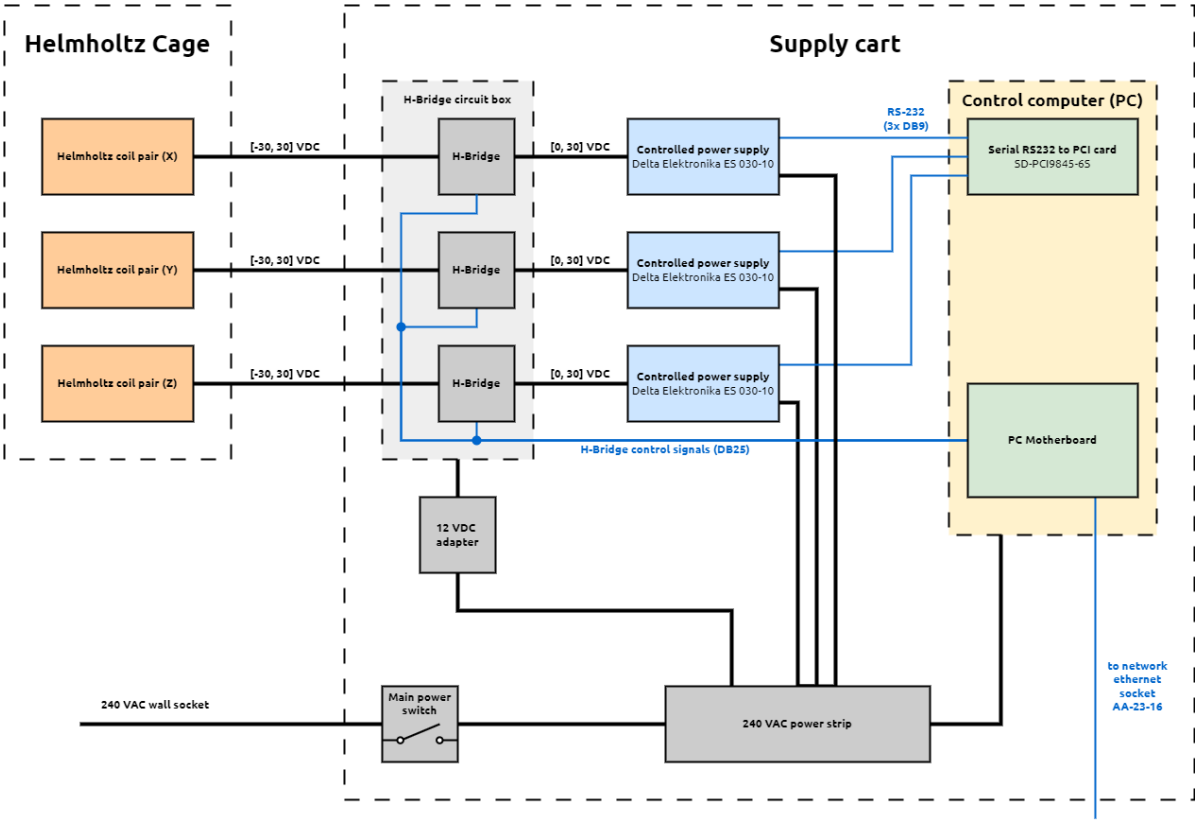


Figure 6.3: Simplified system diagram of the Helmholtz cage setup.

Cage Structure

The structure of the Helmholtz cage itself consists of a structural frame made of standard aluminium construction elements (MK profiles), the copper wire for the coils, and the supporting U-profiles for the copper wire. The U-profiles are connected by Delrin plastic corner parts. Besides acting as structural joints, these corner parts also allow a rounded corner shape such that the coil wire does not get damaged, and also electrically insulate the coil profiles on each side to prevent the possibility of current running through them. The resulting coils inside the profile are therefore square coils, albeit with slightly rounded corners.

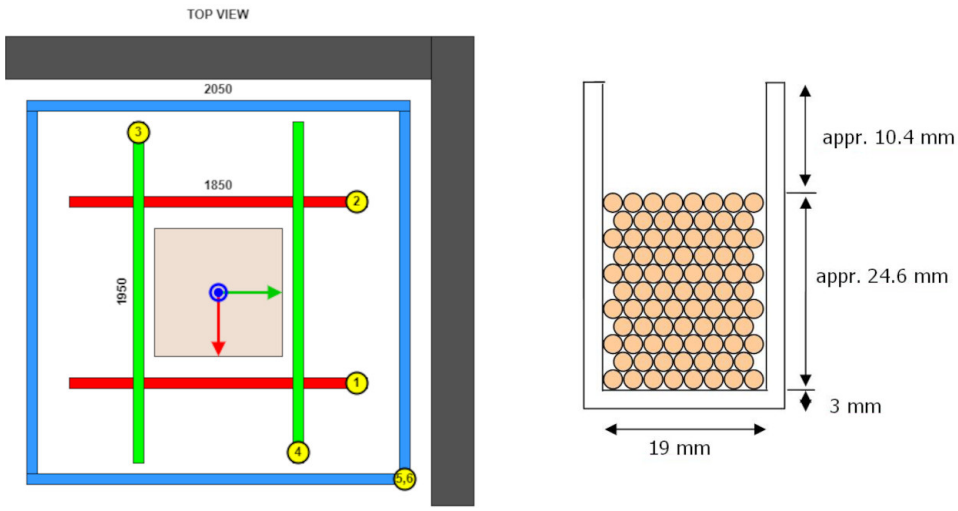


Figure 6.4: Locations of power terminals for each coil. Terminals 1-4 are near the bottom of their respective coils. Terminal 5 is on the top Z-coil and terminal 6 on the bottom Z-coil.

Each coil features 83 windings. In order to perpendicularly nest the coil pairs inside each other, they have different diameters. The largest coil pair has a side length of 2.05 m, the middle pair 1.95 m and the smallest pair 1.85 m. Each coil pair is connected in series, and positioned in the Helmholtz cage such that their fields align. Each individual coil has a terminal to which power connects. The points have been numbered, and their location can be seen on the left side of Figure 6.4.

The exact properties of the copper wire are not known, other than that they were supplied by Huibers B.V. Poppenk's thesis [46] mentions the cross-sectional area of square wire that was considered first, but this was later rejected in favour of the cheaper and more convenient round wires. The area of the round wire is not mentioned, but estimations based on the geometry on the right side of Figure 6.4 and the measured resistance places the cross-sectional area of the wire somewhere between 3.65 mm^2 and 3.73 mm^2. The closest match for this is AWG12 wire, which has a diameter of approximately 2.05 mm. The report further mentions that the manufacturer specifies the maximum current rating of the copper wire as 6 A.

Supply Cart

The supply cart is a burgundy-coloured metal enclosure with attachment rails for rack-mounted electronics, a storage drawer, and a flat work top. It can be seen in Figure 6.1. We refer to it as the *supply cart* because it has wheels to move it around, and in the old Helmholtz cage system, it houses the three coil power supplies.

Inside the supply cart, there is a dedicated power supply for each coil pair, each one an ES-030-10 unit by Delta Elektronik. These power supplies are unipolar, and able to supply up to 10 A of direct current at 30 V. To make sure that the magnetic field of each coil pair can be reversed in polarity, the direction of the current through each coil pair must be reversed. This is achieved by a trio of H-Bridge circuits, which are stored inside a gray electrical enclosure box inside the supply cart.

When the Helmholtz cage is activated, it will be able to generate a magnetic field vector in the middle of the cage. The magnitude of this magnetic field vector, or the local strength of the field, is limited by the current carried by the coil. If the Helmholtz cage is calibrated well, it will have a *control volume* in the middle of the cage in which the magnetic field varies very little in both magnitude and direction. In other words, in this control volume, the magnetic field vectors will be almost identical. As such, when we speak of "the magnetic field vector" inside the cage, we implicitly mean the magnetic field conditions inside the whole control volume.

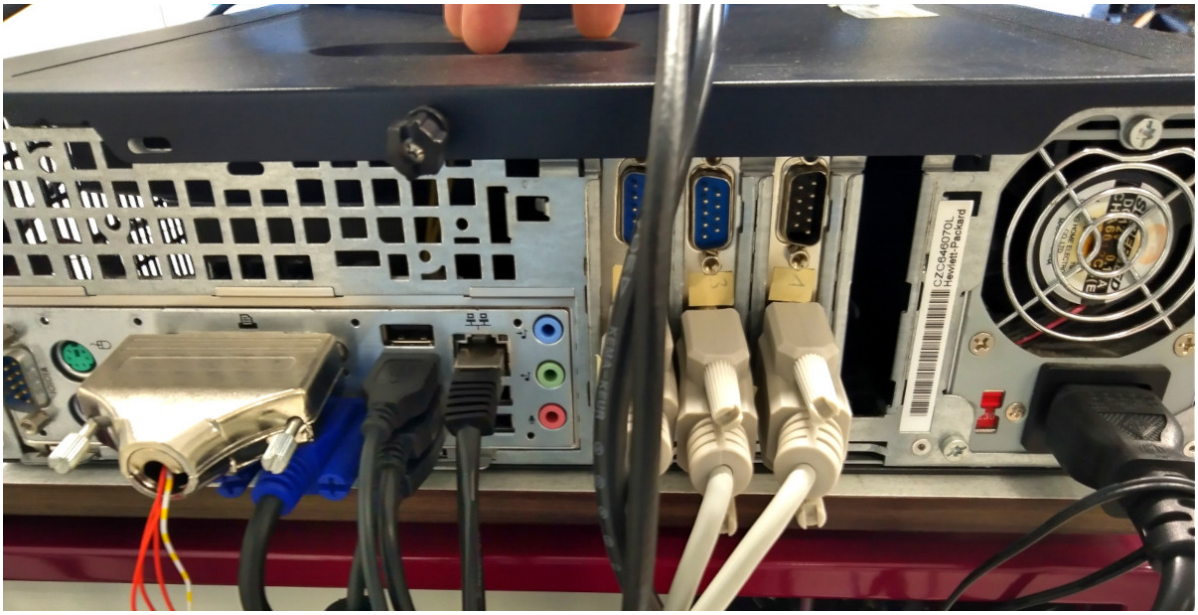


Figure 6.5: Connections at the rear of the supply cart's computer. Left is the silver DB-25 connector that connects to the H-Bridge circuitry, and in the middle are the three gray DB-9 connectors for the RS-232 connections to the power supplies.

Control of the magnetic field vector inside the Helmholtz cage is done by the computer placed on top of the supply cart. The power supplies are **voltage-controlled**, which means that the power supplies must be instructed which voltage value to track. These instructions are generated in a LabView interface running on the computer. Communication between the computer and each power supply is done via an RS-232 COM channel that runs on a dedicated DB-9 cable. The three DB-9 cables interface with the computer using a RS-232 to PCI card, which is a *SD-PCI9845-6S*.

The LabView interface also controls the state of each H-Bridge, to make sure that the generated magnetic field in each direction can also be inverted. This is achieved with discrete signals that flow through a modified DB-25 printer cable from the computer to the H-Bridge box. In Figure 6.5 these wires are coloured bright orange. A separate 12 VDC power line is needed to allow the H-Bridge box to work. This is achieved by siphoning it off one of the two 240 VAC power strip using a 12 volt adapter. The ground of the DB-25 cable and the 12 VDC adapter are connected together inside the H-Bridge box. The 240 VAC power strips also contains three more power cables for the supplies, one more for the supply cable for the computer, and one more for the computer display monitor. Ahead of this power strip, there is a large power switch that can cut off power to the whole supply cart, and the other side of this switch is connected to the wall power socket. The only other connection to the outside is the network cable, which plugs into one of the Ethernet sockets in the wall.

To prevent the possibility of dangerous charge build-up in the metal structure of the cart, the metal enclose is grounded to the two power strips (see Figure 6.6) which incidentally also normalizes the grounds of the power bars to each other.

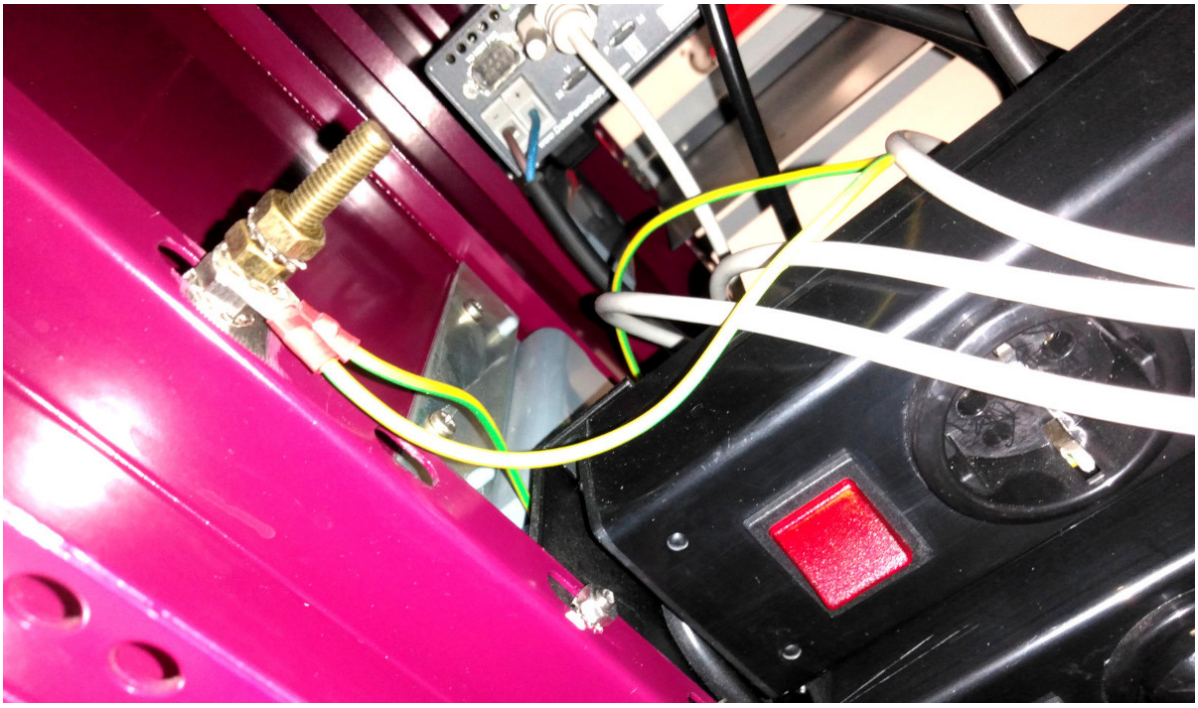


Figure 6.6: Anchored grounding wires that ground the metal cart enclosure to the two power strip grounds.

6.1.3. H-Bridge circuitry

As mentioned before, the supply cart features a gray plastic enclosure that houses the electronics required to switch the direction of current flow through the coils. Figure 6.7 shows a picture of the circuit board. The three top left input ports come straight from the variable power supplies. The port at the top right comes from the 12 VDC adapter and the DB-25 cable from the computer. The bottom six holes are outputs, with ports 1 and 2 running to the X-coil pair, 3 and 4 to the Y-coil pair, and 5 and 6 to the Z-coil. The coils are connected in series, and the connecting wire runs back into the box (see for example the line between ports 3 and 4.

On the right of the circuit board, there are three large fly-back diodes, rated for 12 A and 100 V RRA. These are needed to ensure that no large inductive voltage spikes make it back to the power supply. The coil pairs of the Helmholtz cage have an inductance in the order of 0.1 Henry, which means that the established magnetic fields contain a significant amount of energy. When the power supplies would be suddenly switched off, these fields will start to collapse in such a way that the current through the coils remains constant. If this happens in a short enough time frame, it causes high voltage spikes in the circuit. The fly-back diodes ensure that these spikes remain largely contained to the coil, rather than travelling further downstream and possibly causing damage in other components not rated for such voltages.

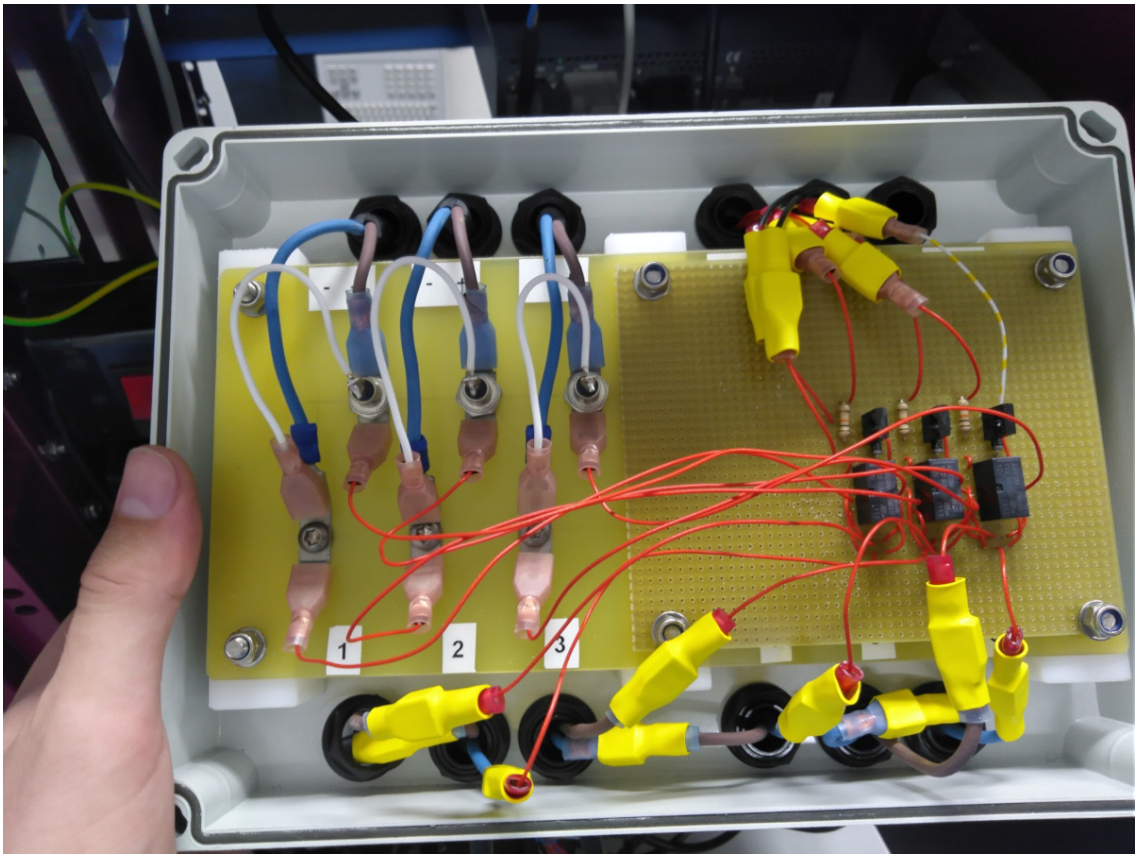


Figure 6.7: Electronics inside the "H-Bridge" box.

On the left, there is a trio of H-Bridge-like switching circuits consisting of a relay (FX2 D3202), an NPN transistor (BD677), a fly-back diode, and a 10k current limiting resistor. The three control pins coming from the DB-25 cable are connected to the base of the NPN transistors, and will thereby open or close the transistor. When one of the NPNs is activated, the relay circuit is connected to the 12VDC line, which will switch the state of the relay, which in turn switches the direction of current flowing through the connected Helmholtz coil pair. The small diode is also a fly-back diode, preventing inductive voltage spikes from the internal relay coil from causing damage. A complete wiring diagram can be found in Figure 6.8.

It does not take a huge amount of electronics expertise to spot some glaring issues with this circuit board. The main issue is that the full load fed through the Helmholtz coil pairs first runs through some components that were not designed to handle such currents. According to the datasheet of the FX2 D3202 [47], the maximum allowed carry current through the relay is 2 A. However, they are connected directly to a power supply that can supply up to 10 A. As we will see later, the resistance of the Helmholtz coil pair actually limits this current to 4.3 A or less, but this is still a hazard. In addition, where the variable power supplies output onto 1.5mm^2 wires, these are then connected to the orange AWG22 braided wires, which still carry the full load. Braided AWG22 wires typically have a maximum

current load rating of 3 A or lower [48]. So also for these wires, current limits may be exceeded during operation of these electronics. The components that experience currents in excess of their maximum rating are coloured red in the schematic of Figure 6.8.

An operator manual of the Helmholtz cage from 2012 [49] does indicate that the operators were aware of the 2 A limitation on the relays, and it appears from what little documentation that could be found that the Helmholtz cage was intentionally operated below spec to ensure that this current would not be exceeded. We will come back to this in section 6.3.

6.1.4. Switching board schematic

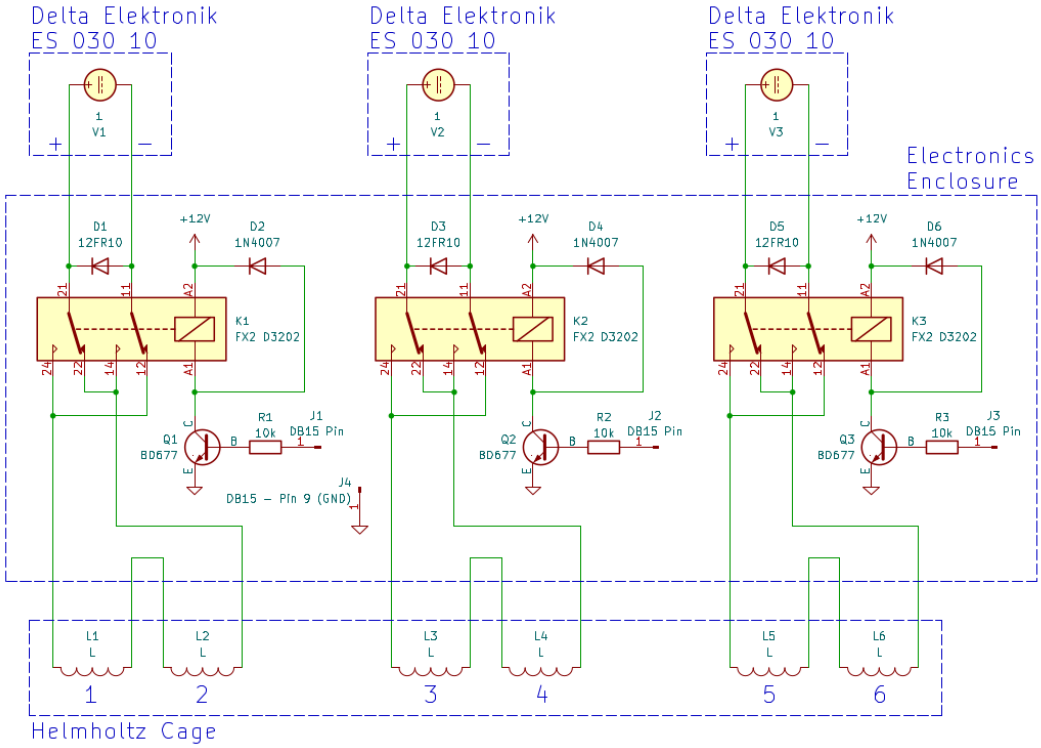


Figure 6.8: Reverse-engineered schematic of the polarity switching circuit of the old Helmholtz cage system

6.1.5. A brief specification sheet

Table 6.1: Characteristics of the old Helmholtz cage system

Parameter	Value	Unit	Source
Coil side length (X/Y/Z)	1.85/1.95/2.05	m	measurement
Perpendicular coil width	0.019	m	[46]
Windings per coil	83	-	[33]
Total wire length per coil pair	1230/1297/1363	m	calculation
Resistance per coil pair	6.27/6.85/6.92	Ω	measurement
Inductance per coil pair	108.2/115.2/122.3	mH	calculation
Reported max B	15	$\times B_{EMF}$	[33]
Homogeneous volume	0.5	m^3	[33]
Field magnitude homogeneity	10%	m	[33]
Power supply maximum supply voltage	30	V	[50]
Power supply maximum supply current	10	A	[50]
Actual achievable coil current	4.3	A	calculation
Actual max. current for safe operation	4.3	A	[47]
Power supply efficiency	86%	–	[50]

6.2. Theoretical Performance

To evaluate the theoretical performance that the old Helmholtz cage system should be capable of, we can use the values from subsection 6.1.5 as input. Note that the resistances of the individual coils was not given in documentation, despite it being in Table 6.1. So, this had to be measured first.

6.2.1. Coil resistance

The resistance of each coil was measured on 29-06-2023. This measurement includes the resistance of each singular coil (not pair) plus several meters of attached connection cabling. The measurements were done at room temperature, and currents were kept low such that the coils would not heat up much. A range of low voltages between 0.8 VDC and 2 VDC was applied to each coil with a lab supply, and the current was measured at each point. The measured values per coil were then averaged, and this was taken as the coil voltage. The results can be found in Table 6.2.

Table 6.2: Measured coil resistances

Coil	1	2	3	4	5	6
R_{avg} [Ω]	3.11	3.16	3.50	3.35	3.46	3.46

In other words, the resistance of each individual coil is between 3.0 Ω and 3.5 Ω. If we connect two coils together, and assume the resistance in the cables connecting the two coils is negligible, we get at most 7 Ω for each coil. To force a current through a conductor with this resistance, 7 volts are needed for each ampere. As a side note: it is preferable to have even more voltage available, to overcome the resistance of the coils when switching the field quickly, and to compensate for any resistance increase due to ohmic heating of the coil.

At a later date, on August 31st 2023, the resistances were measured again, this time with a digital multimeter. In this test, the coils resistances were measured separately from the connection cables that connect the coils to the supply cart, which is around 10 meters of 1.5 mm^2 wire per coil.

Table 6.3: More measured coil and cable resistances

Coil	1	2	3	4	5	6
Orientation	**X**	**X**	**Y**	**Y**	**Z**	**Z**
$R_{cable} + R_{coil}$ [Ω]	3.05	3.15	3.3	3.3	3.4	3.35
R_{coil} [Ω]	2.85	2.8	3.0	2.95	3.1	3.1
R_{cable} [Ω]	0.2	0.35	0.3	0.35	0.3	0.25

As a rule of thumb, the cables running to and from the coils amount to a 10% increase in resistance.

So, the Helmholtz cage design has an important bottleneck in its power supplies: the magnetic field that could be generated is **voltage-limited**. Each power supply could supply up to 10 A with a driving voltage up to 30 V. However:

$$V_{max} = I_{max} \cdot R = 10\ A\ \cdot\ 7\ \Omega = 70\ V$$

To supply 10A to the largest coil pair, one would need 70+ V for the aforementioned inductance reason. This causes the Helmholtz cage design to be limited at most to:

$$I_{max} = \frac{V_{max}}{R} = \frac{30\ V}{7\ \Omega} = 4.29\ A$$

and it will do worse if the field is being changed rapidly or if the internal resistance of the coils increases due to ohmic heating.

6.2.2. Helmholtz cage static performance

The strength of the static field established by the Helmholtz cage is directly dependent on the current delivered by the power supplies. With 5 A of current running through each coil, the strength of the manifested magnetic field for each coil pair is as follows:

$$\overline{B}_{max,HHC} = \left(\begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} \right)_{max} = \begin{bmatrix} 365.4 \\ 346.6 \\ 329.7 \end{bmatrix} \mu T \tag{6.2}$$

To give an idea of how this compares to the EMF in absolute terms:

$$\left| \overline{B}_{EMF,Delft} \right| = \left\| \begin{bmatrix} 8.161 \\ 17.367 \\ -45.457 \end{bmatrix} \right\| \mu T = 49.342 \ \mu T \tag{6.3}$$

$$\left| \overline{B}_{max,HHC} \right| = 602 \ \mu T$$

Note that the vectorial components are in the C-frame. So in absolute terms, the strongest possible static magnetic field vector that the Helmholtz cage will be able to generate has 12.2 times the magnitude of the local EMF.

This however, is not an "apples-to-apples" comparison. A better comparison would be: what is the strongest static magnetic field vector that the Helmholtz cage can establish **in any direction**? Better yet, since the Helmholtz cage will have to cancel the local EMF to create a magnetically neutral testing environment, what is the strongest static magnetic field that the Helmholtz cage can generate in conjunction with the EMF neutralization vector?

First, let's look at a static load: Imagine we require a fixed field strength B_s, supplied on each individual axis by each coil pair:

$$\overline{B}_{req} = \begin{bmatrix} B_s \\ B_s \\ B_s \end{bmatrix} \mu T$$

The Z-coil pair will have the highest coil area, so it will generate the strongest fields given the same current. But its larger size also increases its electrical resistance, which means it requires more voltage for the same current. It turns out that the latter effect is more pertinent: it is possible to generate a maximum of 285 μT on each axis, with the Z-coil pair limiting the setup as it reaches the limit of 30 V first:

$$I_{req} = \begin{bmatrix} 3.900 \\ 4.111 \\ 4.322 \end{bmatrix} A \quad , \quad V_{req,\,min} = \begin{bmatrix} 24.454 \\ 28.16 \\ 29.907 \end{bmatrix} V$$

Second, to purely neutralize the local EMF, the Helmholtz cage requires the following:

$$I_{req} = \begin{bmatrix} 0.08862 \\ 0.09453 \\ 0.1005 \end{bmatrix} A \quad , \quad V_{req,\,min} = \begin{bmatrix} -0.871 \\ -2.337 \\ 6.571 \end{bmatrix} V$$

which corresponds to a total of 5.45 W.

When we stack these two conditions: a static field on each axis, and neutralization of the local EMF, we can only achieve a static field of 241 μTon the Z coil before 30 V is reached:

$$I_{req} = \begin{bmatrix} 3.20 \\ 3.22 \\ 4.34 \end{bmatrix} A \quad , \quad V_{req,\,min} = \begin{bmatrix} 20.05 \\ 22.07 \\ 30.00 \end{bmatrix} V$$

Let us assume for now that generating a purely upward pointing magnetic field component is the most demanding condition for the power supply. In other words: if we can generate a field strength of

magnitude B_s in the +Z direction, we can do it in any direction. This may not be necessarily true, but it is close enough for the purposes of this analysis.

So then it is true that whilst cancelling the Earth magnetic field, the Helmholtz cage can with its current hardware limitations generate a magnetic field vector in any direction of at most 241 μT. It may be able to do better in specific directions, but for any direction 241 μT is the limit. Comparing this to the magnitude of the Earth magnetic field, we get:

$$\frac{241}{49.3} \approx 4.8 \times \overline{B}_{EMF}$$

To be fair to Poppenk [33], the local Earth magnetic field at Delft has slightly increased in magnitude since the original article on the Helmholtz cage setup was written. But claiming that the Helmholtz cage in the cleanroom of the Aerospace Engineering faculty can generate 15 times the local Earth magnetic field vector seems like an exaggeration. However, 4 times seems relatively achievable with the current hardware.

6.2.3. Helmholtz cage dynamic effects

We will now briefly look at dynamic effects. In the context of the supply circuit, we have up to now only really looked at the Helmholtz coils as resistive loads of around 7 Ω per pair. However, within the circuit, they should also be treated as inductive loads. The three coil pairs of the Helmholtz cage are air-core inductors capable of establishing fairly potent magnetic fields. As a result of Lenz's law, any change in current in the coil pairs will result in an opposing induced voltage. This voltage is established with the energy stored in the magnetic field, and is proportional to the inductance of the coils and the rate of change of the current:

$$V_{L(t)} = -L\frac{dI}{dt} \tag{6.4}$$

The minus sign in Lenz's law indicates the opposing direction of the voltage.

Within the context of the Helmholtz cage system, it means that dynamically changing the magnetic field will be opposed by the inductance of the coils, and the power supplies will have to compensate by overcoming the opposing voltage. If the power supplies are already at their maximum supply voltage and cannot supply more, it means that the magnetic field change on the coil cannot be fully controlled. Instead the field will slew only as fast as the inductance allows, which may be slower than desired.

Wheeler [45] provides a method for determining the inductance of a square conducting coil:

$$L = \mu_0 n^2 a(4/\pi)\left[\ln\left(1+\pi\frac{a}{b}\right) + \frac{1}{3.64 + 2b/a + 0.51(b/a)^2}\right] \tag{6.5}$$

where a is half the side length of the square coil, b is the width of the coil perpendicular to the direction of the windings, and n is the number of windings. μ_0 is the vacuum permeability ($1.256637062 \cdot 10^{-6}$ $[H/m]$). Using the dimensions of the Helmholtz cage from Table 6.1, we find the inductance of each coil pair to be:

$$\begin{bmatrix} L_X \\ L_Y \\ L_Z \end{bmatrix} = \begin{bmatrix} 108.2 \\ 115.2 \\ 122.3 \end{bmatrix} \; mH \tag{6.6}$$

Example rotation

Let us look at what this means in practical terms. Imagine that we wish to simulate a satellite rotating in a static magnetic field. We can simulate this using the Helmholtz cage setup by fixing the satellite model in the Helmholtz cage and rotate the magnetic field around it. As an example, imagine that the field performs a full rotation every minute (1 rpm) counter-clockwise in the XZ-plane. At the maximum achievable field strength of 241 μT, the current in each coil pair will undergo a sinusoidal motion. Between the X and the Z coil, sinusoidal curves of the currents are 90 degrees out of phase.

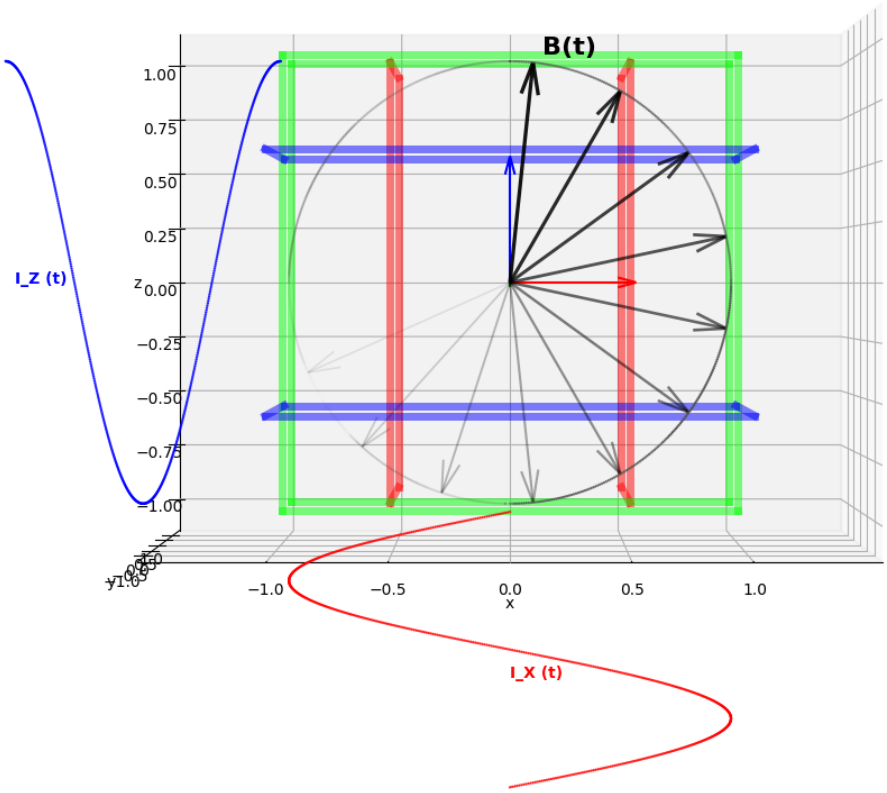


Figure 6.9: Visualization of an example rotation, where a magnetic field vector B is rotated in the XZ-plane. The current curves for the power supplies of the X and Z axes are also drawn, in red and blue respectively.

We focus on the Z-coil, since we know it is most heavily constrained by the power supplies, given that the resistance of this coil is the highest. The current in the Z-coil will look something like:

$$I_Z(t) = 4.34 \cdot \sin\left(\frac{2\pi}{60}t\right)\ A \tag{6.7}$$

For the purposes of Lenz's law, we are more interested in the rate of change of the current, so we differentiate:

$$\frac{dI_Z(t)}{dt} = 4.34 \cdot \frac{2\pi}{60} \cdot \cos\left(\frac{2\pi}{60}t\right)\ A/s \tag{6.8}$$

whose maximum absolute value equals:

$$\left[\frac{dI_Z(t)}{dt}\right]_{max} = 4.34 \cdot \frac{2\pi}{60} \cdot 1 = 0.4545\ A/s \tag{6.9}$$

which according to Lenz's law will result in a maximum opposing voltage of:

$$\left[V_{L(t)_Z}\right]_{max} = -L_Z \left[\frac{dI_Z(t)}{dt}\right]_{max} = -0.1223 \cdot 0.4545 = -55.6\ mV \tag{6.10}$$

This can also be seen in Figure 6.10. The yellow line, which is the voltage increase required to overcome the induced voltage, is insignificant in comparison with the voltage needed to drive the coils.

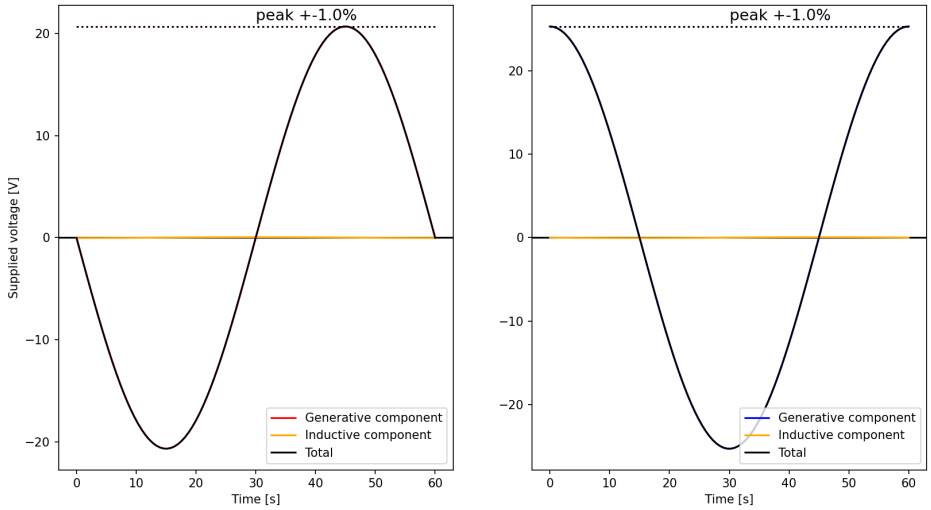


Figure 6.10: Supplied voltage envelope of **X** and **Z** during the rotation (1 rpm, 241 μT).

For 60 rpm, this will be approximately:

$$\left[V_{L(t)_Z}\right]_{max} = -0.1223 \cdot 4.34 \cdot 2\pi = -3.34\ V$$

and for 600 rpm (10 rotations per second) it will be:

$$\left[V_{L(t)_Z}\right]_{max} = -0.1223 \cdot 4.34 \cdot 2\pi \cdot 10 = -33.4\ V$$

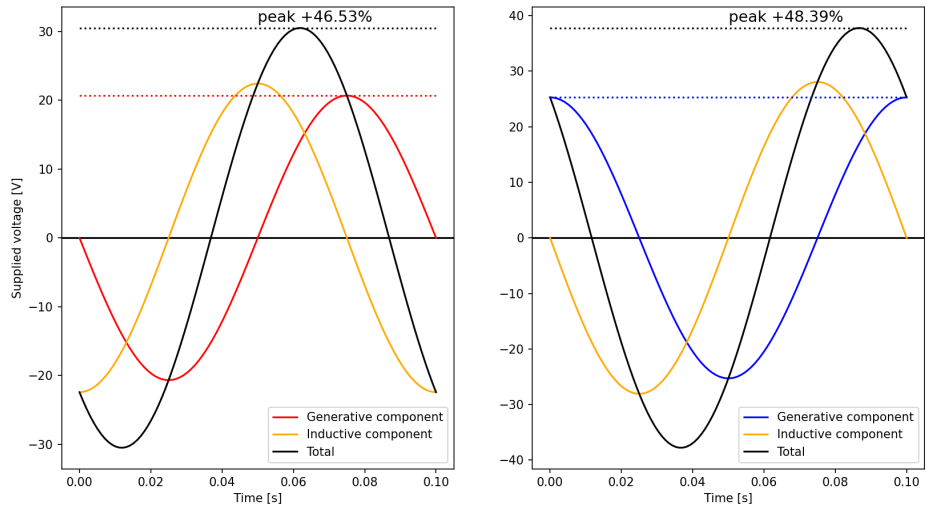


Figure 6.11: Supplied voltage envelope of **X** and **Z** during the rotation (600 rpm, 241 μT).

A rotation of 60 rpm for most spacecraft is quite rapid, so it is unlikely that this is a typical test scenario. Even more so for 600 rpm. However, the point of testing is often to investigate edge cases. In the case of 60 rpm, one can expect that an extra 3.34 V is needed from the power supply.

So really, the effect of induced voltages are rather minimal. This should not be a surprise. We have discussed two reasons for why the power supplies need to operate at a certain voltage: (1) to drive a current through a wire with non-zero resistance, which is needed to establish a magnetic field of a certain strength and (2) to overcome the induced voltage that results from changing the current running through an inductor. The former reason is directional proportional to the **current**, which we modelled

as sine wave. The latter is directly proportional to the **rate of change of the current**. The derivative of a sine is a cosine, and so the effects are 90 degrees out of phase; one will be zero when the other is maximum, and vice versa. This can also be seen in Figure 6.11: at higher field rotation rates, the extra voltage required to overcome the induced voltage is on par with the driving voltage. The sum of the two is still only 48% higher than the driving voltage, however. Appendix I contains several more such plots.

It is therefore expected that for a simple and stable rotational motion, these effects will not interfere with each other very much. For more complex magnetic vector motion, for example modelling an EMF envelope from the perspective of a satellite in low Earth orbit (LEO), this is less certainly so. It would altogether be wise to have a bit of headroom on the supply voltage, but it seems that it is not as likely to be a problem as initially expected, at least for most basic testing.

6.2.4. Helmholtz cage field homogeneity

As mentioned previously, the performance of the Helmholtz cage could not be tested, as its control software was not accessible. To assess the field homogeneity characteristics that the Helmholtz cage should be capable of, a theoretical analysis was performed. It is mainly focused on the variation of the field strength. The determination of the variation in direction is slightly more involved, and may be done at a later time, if needed.

But what is the field homogeneity anyway, and why is it relevant? The Helmholtz cage is able to generate a magnetic field vector with a given direction and magnitude. For the purposes of testing satellites, it is desirable to have minimal variation in either of these quantities in the area where the satellite is being tested. The spacing of each pair of coils is optimized such that there is a "control volume" in the middle of the Helmholtz cage in which there is very little variation in magnetic field strength and direction.

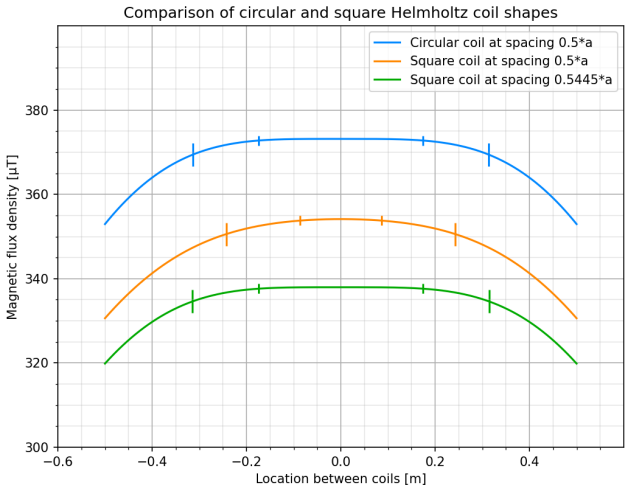


Figure 6.12: Comparison of circular and square Helmholtz coil shapes. The absolute field variation within the large vertical markers is 1%, and within the small vertical markers it is 0.1%.

For circular coils, the ideal distance between two Helmholtz coils is equal to the radius of the coils. This can also be seen from the blue line in Figure 6.12. Across the horizontal segment in between the two small markers, the field strength varies by only 0.1%, which in this case is around 0.38 μT. The ideal distance between square Helmholtz coil pair is not equal to half the side length, as can be seen from the red line in Figure 6.12. Instead, it can be shown to be equal to approximately 0.5445 times the side length of the coils (see [43] and chapter 5). This is demonstrated by the orange line in Figure 6.12.

Using the method of subsection 6.2.2, one can determine the field strength variation for the Helmholtz cage at the Aerospace Engineering faculty. One can then measure the length of distance for which the field remains within a certain percentage of the field strength in the middle. This is in fact how Figure 6.12 was made. This distance we will call the homogeneity range, and it is difference for each coil. The resulting values for 1% and 0.1% homogeneity ranges can be found in Table 6.4. In addition, Figure 6.13 visualizes these ranges as cuboids.

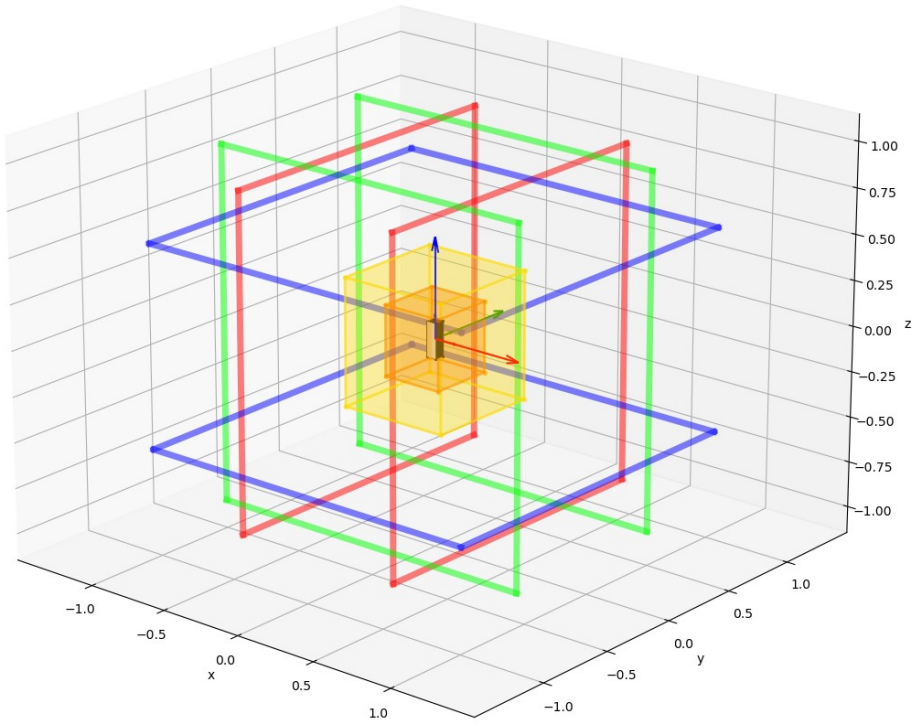


Figure 6.13: Visualized volumes corresponding to the calculated homogeneity ranges of Table 6.4

Table 6.4: Calculated ideal coil spacings and homogeneity ranges

	X	Y	Z
Coil side length [m]	1.85	1.95	2.05
Ideal coil spacing [m]	1.007	1.062	1.116
HR with 1% variation [m]	0.635	0.669	0.703
HR with 0.1% variation [m]	0.351	0.370	0.388

The 0.1% homogeneity volume is easily large enough to accommodate Delfi-PQ in any orientation (large diagonal of 209 mm), and just large enough for a 3U CubeSat (large diagonal of 332 mm). Larger CubeSats, such as 2X2X3 12U CubeSats (large diagonal of 412 mm) are too large for the 0.1% volume, but but easily fit within the 1% volume.

But is this "homogeneity volume" shaped like a cuboid at all? In reality, it is not. Using the method of Frix et al. [43], we can calculate the magnetic field strength on discretised 2D slices in the Helmholtz cage and get an idea what is going on. One example of this can be found in Figure 6.14. Here, a pair of coils is depicted with each coil aligned vertically. Since the magnetic field strength is very strong near the coils, the raw resulting image is not very interesting. When normalizing the data a bit, more interesting patterns become visible. In the bottom two pictures of Figure 6.14, a 1% and a 0.1% variation condition have been imposed, blanking out everything outside this range.

We will not dive too deeply into this here, but what is clear from first glance is that the homogeneity of the field strength in the Helmholtz cage is much more complex than a simple cube could capture. One should therefore be conservative about the homogeneity range, and take the values from Table 6.4 with a grain of salt.

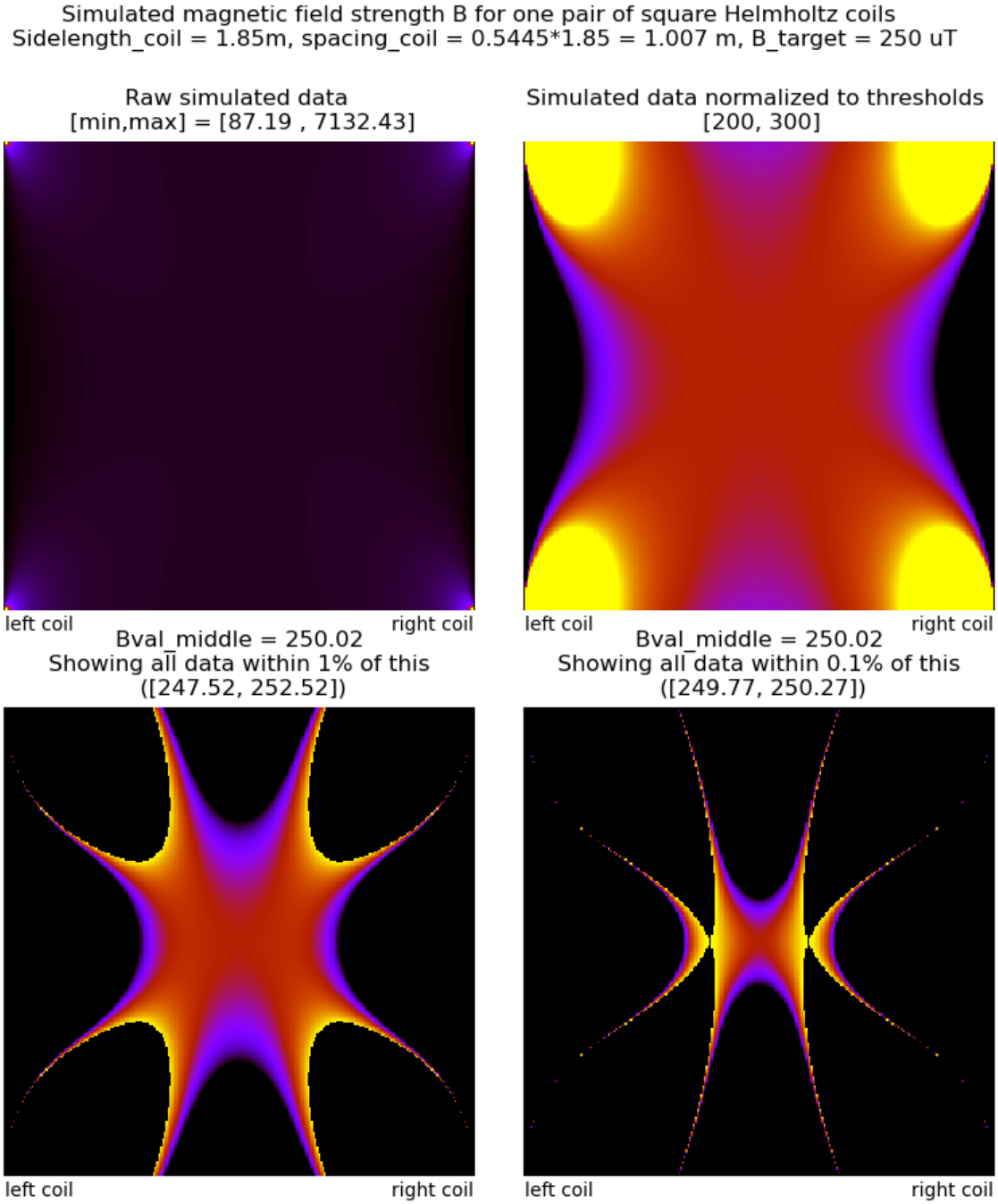


Figure 6.14: Simulated magnetic field strength for the pair of X-coils. The two coils are oriented vertically.

6.3. Identified Problems and Limitations

Now that the old Helmholtz cage setup has been described in detail, it is time to sum up the problems that were identified with the setup. This is so that future upgrades can address these issues.

1. **The power supplies of the setup are voltage-limited.** As a consequence, the field strength generated by the coils may be lower and less reliable than desired. The Z coil pair suffers worst from this as its internal resistance is the highest. To address this issue, either new coils with lower resistance should be built, or more better power supplies that can operate with higher voltage should be procured. The latter option seems like the easier and cheaper one. The coil current should still remain limited to 6 A, as this is maximum current rating of the copper wire in the coils.
2. **The H-bridge circuitry was not build according to correct maximum current specs.** Some of the components used in this box was rated for at most 2 A, essentially limiting safe operation of the Helmholtz cage setup to this current. However, the setup was designed to do more, and there are no hardware safety measures to prevent a user from going over 2 A, though there may have been software measures in the control software. Regardless, it appears as though the H-bridge electronics are due an overhaul to ensure that higher currents can be achieved safely. The new target should be 6 A, in line with the previous point.
3. **The power supplies are controlled in voltage.** This was not discussed in detail in the preceding chapters, but the fact that the power supplies track voltage is that when the resistance of the coils changes, the current will drop. This is actually a likely scenario, as the coils will heat up during operation. Since the power supplies only look at voltage, it will be unable to detect or compensate for the resulting change in current. The magnetic fields generated by the Helmholtz cage are dependent upon current, and so the fields will fluctuate unreliably as a result. The most obvious way to address this issue is to use power supplies that can track current instead of voltage.
4. **The X and Y coil pairs have an undesirable degree of freedom.** Since these coil pairs are suspended from the surrounding structure by threaded rods, they have the ability to swing along the horizontal axis parallel to the coil axis. This is undesirable, as the calibration of the setup may be thrown off by movement of the coils, for example by an operator accidentally bumping into one of the coils. There are no identifiable benefits to the coils having this degree of freedom. The author suggests addressing this by making four custom clamping connectors that can be used to connect the X and Y coils at the points where they meet. This restricts motion of these coils, since their degrees of freedom are perpendicular. Such clamping connectors can easily be made to size and 3D printed at the faculty.
5. **The software interface was designed a long time ago, is running on a deprecated system, and has scarce documentation.** This is unfortunate, because it presents a significant hurdle towards the Helmholtz being applied more broadly, for example as a more general test bed by other university departments, or as part of assignments for engineering students. To address this point, either the existing software setup has to be rediscovered, ported to more modern hardware, and re-documented, or it is time to start from scratch and build a new interface. It is unclear which of these is more work. The latter option has the disadvantage that the nuances in calibration that have been done over the years may be lost, but it has the advantage that new features are much easier to plan for, and if new hardware is procured, new software may also give more freedom when picking hardware.
6. **Low degree of automation.** When testing satellites, it would be very useful to be able to simulate orbital conditions with the Helmholtz cage. This includes not just the strength of the EMF around the satellite, but also the change of the field as a result of the attitude changes of the satellite in space. Of what could be pieced together from documentation, the old setup does a way to pre-program the Helmholtz cage with a magnetic envelope, by providing it with a pre-calculated array of target values. However, to streamline future testing it may be very helpful to have the generation of these commands as part of the setup itself. In the same vein, it would be helpful for the Helmholtz cage to be able to calibrate itself, to have an array of sensors ready for testing.

7

Characterization of Ambient Magnetic Fields at the Test Location

In the previous chapter we took a closer look at the old Helmholtz cage system, with the intent to discover how it could be used as part of a satellite testing system. However, a crucial characteristic of this old Helmholtz cage system was lacking: magnetic disturbances inside the Helmholtz cage. If one is attempting to perform sensitive magnetic tests inside the Helmholtz cage, but there are disturbances in the environment, it is difficult to know how good the measurements are. It is difficult to draw accurate conclusions from a tests if one cannot make an assessment about the signal-to-noise ratio of the measurements.

7.1. Motivation

It would be disastrous through go to the effort of designing and constructing a magnetic testing system, only to find out during the validation phase that its performance is hampered or even rendered impotent due to the nature of magnetic disturbances at the test location. Before we can design a magnetic testing system as discussed in section 4.3, the test location itself should be investigated beforehand. If there are particularly bad magnetic disturbances at its location, it may be necessary to address these as part of the design, or decide that a different testing location should be used.



Figure 7.1: The Magnetic Test Facility at the Goddard Space Flight Center is an example of a Helmholtz cage and magnetic testing environment specifically designed to minimize magnetic field disturbances [51].

To illustrate this point, consider the Magnetic Test Facility at the Goddard Space Flight Center, which was used for NASA missions such as the Apollo missions. This facility is located in an area with a low degree of magnetic field disturbance, and the facility itself was specifically designed to exploit this: materials such as wood and aluminium were used in its construction to ensure a very low amount of ambient magnetic field strength [51]. Of course, the ever-present Earth magnetic field (EMF) will still be there, but since this is a well-known and very stable static disturbance, a Helmholtz cage can easily be used to cancel it.

The cleanroom at the Aerospace Engineering facility of the TU Delft is where the old Helmholtz cage system currently located. By contrast, this is *not* a location specifically designed for neutral ambient magnetic field properties. It is located on the 8th floor of the building, sandwiched between other floors with offices and rather close to an elevator system and a crawlspace full of power cables. There is little that one can do to prevent this, other than moving the entire cleanroom facility elsewhere, which was not a likely prospect at the time this thesis work was being performed. The best one could hope for is characterizing the nature of the magnetic disturbance at the test location, and see to what extent its effects could be mitigated.

Therefore, in this chapter we will discuss the characterization of this testing location, which consisted of a series of measurements in July 2023. These measurements had two main goals:

1. Quantify the magnitude, direction, and variation of ambient magnetic fields at the Helmholtz cage location.
2. Establish a framework and toolkit for such tests in the future. This was done with the expectation that the Helmholtz cage in question may be moved to a different location in the future, and that other satellite developers may want to assess their own testing facilities in a similar manner.

The second reason requires some additional explanation. One of the main goals of this thesis work is to find ways to decrease the likelihood that satellite developers will skip magnetic testing of their satellites due to budgetary reasons. For this reason, it is not only important to test one Helmholtz cage setup for ambient magnetic fields, but to present a test method that can be easily adapted for other test setups. Simply put, the second goal is there so that others may benefit from the legwork described in this chapter.

It should be mentioned that in the interest of brevity, this chapter skips over a number of details. Those who are interested in these details may find them in Appendix K.

7.2. Overview and Expectations

The Helmholtz cage setup is located in an ISO 8, class 100,000 cleanroom. This cleanroom is located at the 8th floor of the Aerospace Engineering faculty building, a fifteen-floor building. The rest of the 8th floor contains offices and workshops. It should be no surprise that this is not an ideal test site for sensitive magnetic testing. However, the extent to which these ambient magnetic fields would interfere with magnetic testing of satellites is by no means obvious. In addition, virtually everywhere on the surface of Earth, there is the presence of the Earth magnetic field, a planet-spanning field generated by the liquid nickel-iron core inside the planet. Although the Earth magnetic field has been an active subject of research for many decades, there exist relatively good models of its strength and direction at various locations on Earth. The dynamics of the field are also relatively slow, meaning that for most purposes, the field can be assumed to be constant on a day-to-day basis.

In this thesis, the local EMF will be assumed to be constant. The value of the strength of the local EMF was determined by taking the GPS coordinates of the Helmholtz cage setup:

```
Latitude:   51.99002134132983 N
Longitude:  4.375289736921873 E
Elevation:              30 m GPS
```

and inserting these into the WMM-2020 and IGRF2020 magnetic field models. The value obtained from both models for the date 03-07-2023 was geometrically averaged and taken as the local magnetic field vector at the Helmholtz cage setup:

```
Constant local EMF:  [8.161 17.367 -45.457] uT (abs:  49.3 uT)
```

Before moving onward, let's discuss coordinate systems. Magnetic field strengths are expressed as vectorial quantities, and in order to compare them, a common coordinate system must be adopted. We will continue to use the cage frame C discussed in subsection 6.1.1 and Figure 6.2. This does mean that to obtain the EMF vector in terms of the cage frame C, it must be transformed as follows:

$$\overline{B}_{EMF}\Big|_C = R_Z(-23°)\,\overline{B}_{EMF}\Big|_G = \begin{bmatrix} \cos(23°) & \sin(23°) & 0 \\ -\sin(23°) & \cos(23°) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} B_{EMF,\,X} \\ B_{EMF,\,Y} \\ B_{EMF,\,Z} \end{pmatrix}\Bigg|_G \tag{7.1}$$

where frame G refers to the East-North-Up (ENU) reference frame that expresses the geographic coordinates of the location.

Before testing began, the location of the Helmholtz cage was inspected, primarily to set expectations. It was decided that unless there was good reason, all testing should be performed in the middle of the control area of the Helmholtz cage. This is because during real testing, this is where test articles would be placed, and it is also the place where the upgraded Helmholtz cage itself would be benchmarked in the future. See also Figure 7.2 and Figure 7.3.



Figure 7.2: A picture of the Helmholtz cage setup in the cleanroom. The "test area" is on top of the yellow tabletop in the middle of the Helmholtz cage.

After inspection of the location, it was possible to already speculate on potential sources of magnetic fields that could interfere with testing. Extra attention was paid to objects that are relatively close to the test area. This is because the strength of a magnetic field is inversely proportional to the cube of the distance to its source, as revealed by the Biot-Savart law:

$$\overline{B} = \frac{\mu_0}{4\pi} \int_C \frac{I d\overline{\ell} \times \overline{r}'}{|\overline{r}'|^3} \tag{7.2}$$

Simply put: if you double the distance to a magnetic field source, the field strength B drops by a factor of 2^3, which is 8.



Figure 7.3: Basic map of the floor space around the Helmholtz cage setup. The test area is clearly marked with a yellow box. The potential sources of magnetic interference are marked in orange.

With this in mind, several potential sources of magnetic disturbance were identified:

1. Directly bracing the Helmholtz cage at waist height is a distribution rail for power and Ethernet connection. This can also be seen in Figure 7.2. This is true for two sides of the Helmholtz cage. Both rails are within 1.5m distance from the test area.
2. The Helmholtz cage itself is made out of metal structural components. Most of this is aluminium, which although non-ferromagnetic by itself, may manifest *paramagnetic behaviour* when placed in an existing magnetic field. However, many of the fasteners, such as bolts, nuts, and threaded rods, appear to be made of stainless steel, which may have a non-zero latent magnetic field.
3. A large stainless steel rack with a variety of components stored on it is located right next to the Helmholtz cage setup (also pictured in Figure 7.2).
4. In the hallway, directly behind the Helmholtz cage, there is a large metal fire extinguisher hanging from the wall. Although these are typically made from aluminium alloy, it is not obvious that will have no effect at the test location.
5. The ceiling lights appear to be some sort of fluorescent tube lamps, which are notorious sources of electromagnetic interference (EMI), at least at radio frequencies. They may also have non-negligible magnetic emissions at the 50 Hz operating frequency. Two rows of lamps run across the entire ceiling of the cleanroom, and one row in the hallway right next to the lab.
6. The Helmholtz cage setup is directly at the "back" of the cleanroom. The "back wall" of the cleanroom is a non-reinforced wall made of masonry. Behind it is an off-limits maintenance area containing a power distribution closet, and the power cabling for the floors above also run through this space. Given its relative proximity to the Helmholtz cage, it likely has a substantial magnetic field at the test area, notably at the mains frequency of 50 Hz.
7. At a distance of approximately 15 meters from the Helmholtz cage setup, there are the elevator shafts of the building. The elevator system consists of one large cargo elevator and two smaller elevators. The larger elevator is closest to the cleanroom. Besides requiring large amounts of power, the elevators themselves are large metal objects that are moved up and down at speed, and may therefore have an impact on the local magnetic field at the test area.

These sources can also be seen in Figure 7.3.

7.3. Test Equipment

The magnetic measurements described in this chapter were carried out with equipment that was on-hand. At its core, in consists of a tri-axial milligaussmeter by Alphalab, which formally measures magnetic flux density. This sensor has two modes: It can measure with a precision down to 0.1 mG in a range of [-2000, 2000] mG, or it can measure with a precision of 0.01 mG in a range of [-200, 200] mG. In this report, the Tesla will be used as the main unit for magnetic field strength, rather than the Gauss. The two units have the same scale, but are offset by four orders of magnitude, according to:

$$1\ G \quad = \quad 1 \cdot 10^{-4}\ T$$

or

$$1\ mG \quad = \quad 1 \cdot 10^{-7}\ T \quad = \quad 0.1\ \mu T$$

So the Alphalab milligaussmeter can measure with a precision down to 0.001 μT, which is 1 nT.

The milligaussmeter has a 25mm long probe for its measurements, connected to the main unit with a long ribbon cable about 10 m in length. This ribbon cable is also visible on figures 2 and 5. The main unit outputs the magnetic values of each axis on an analog channel where 1 G corresponds linearly to 1 V. These three analog outputs are BNC plugs, but packaged together with the sensor was a trio of BNC-to dual-banana plug cables.

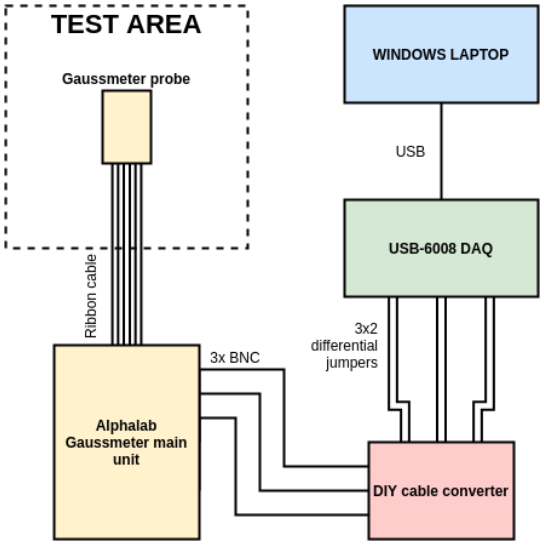


Figure 7.4: Schematic of the magnetic measurement setup

To convert this analog data at speed, an on-hand USB-6008 data acquisition card was used. This is a 4-channel, 12bit analog-to-digital converter (ADC) by National Instruments that can record up to 10 kilosamples per second. This data acquisition (DAQ) card reads analog data on four pairs of differential jumper terminals. Using some banana-to-crocodile clips, jumper wiring, and a polylactic acid (PLA) plastic bracket, a DIY cable converter was made to convert the BNC outputs on the Alphalab milligaussmeter into the differential jumper pairs needed for the USB-6008. The USB-6008 connects to a computer with universal serial bus (USB).

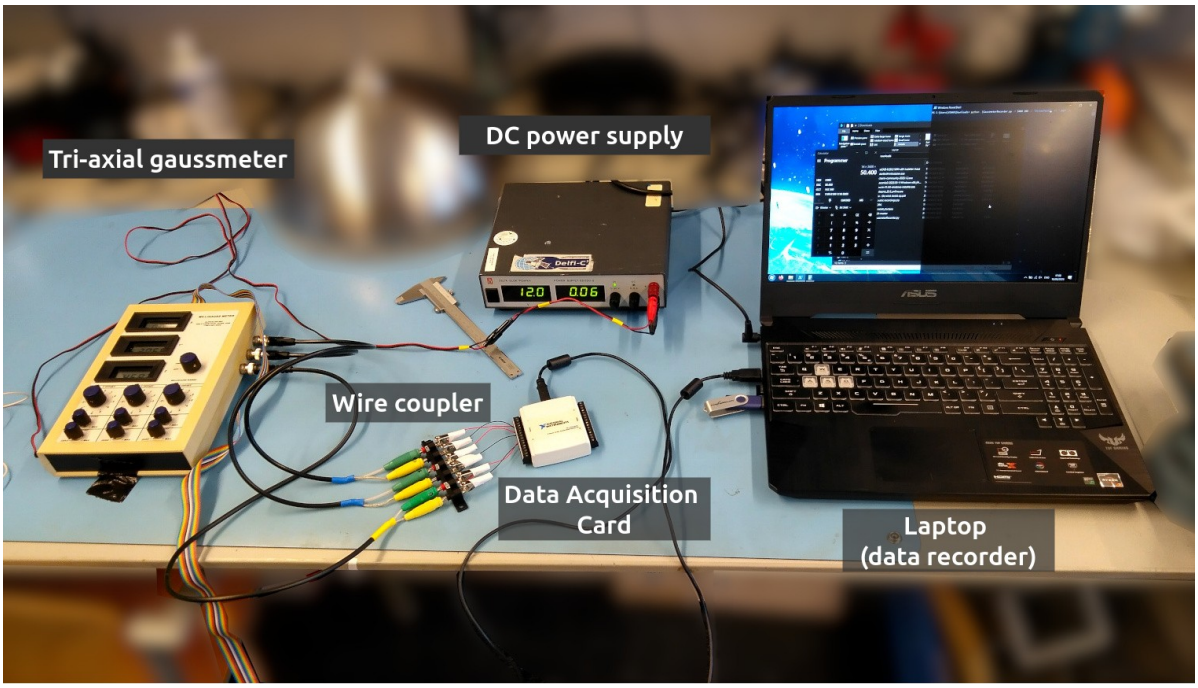


Figure 7.5: Picture of the magnetic measurement setup before deployment: Alphalab gaussmeter main unit (left), rainbow ribbon cable to probe (bottom left) cable converter and USB-6008 DAQ (center), power supply supplying 12 VDC to the milligaussmeter (center top), laptop acquiring digital data (right).

7.4. Overview of tests

During July of 2023 a slew of tests were performed in order to investigate the aforementioned suspicions. Most of these tests intended to measure *static* magnetic disturbances, which in this case means magnetic fields that do not change or have rates below 1 Hz. However, in a few cases it was important to measure field emissions that were higher in frequency, for example to measure the emissions correlated to the 50 Hz alternating current of power lines and fluorescent lights.

For the static tests, data was recorded at 100 S/s. This is enough resolution to tease out periodic signals up to 50 Hz and allow for averaging to get more accurate DC values, but not so much that the raw data files become enormous. Of course, for a tri-axial measurement device, this means that there are 3 simultaneous data channels needed that record this data. For the faster (high resolution) test, a 2500 S/s data rate was used, chosen because the USB-6008 data acquisition card has a collective limit of 10 kS/s for all channels together. For three channels, this means a collective data rate of 7500 S/s on the card, which was deemed enough for the purposes of the test without fully pushing the DAQ card to its limits, risking drop-out or data corruption. By virtue of the Nyquist-Shannon sampling theorem, this means that the high frequency tests can pick up periodic magnetic field disturbances up to 1250 Hz.

Let us now have a closer look at the different tests. Each test generally had a specific goal.

1. The first few tests were a few minutes long and served to validate the test setup and the pipeline for data recording.
2. After this, a longer test was carried out to stress-test the test setup for long test durations and large output data sets. This test lasted from a Friday afternoon to a Monday morning.
3. One of the first tests after that was to test the effects of a large nearby storage rack, made out of steel (see Figure 7.2). A control test was carried out beforehand.
4. To find out whether the paramagnetic properties of the aluminium structural elements of the Helmholtz cage had any adverse effects, a series of tests were carried out where the movable coil pairs were placed closer and farther away from the centre (see section K.4 for pictures).
5. A related test was carried out to test the "table" that holds any device-under-test in the Helmholtz cage. It was placed near the sensor, and then farther away as a control.
6. Elimination of ceiling lights, to witness the disturbances caused by the fluorescent bar lights. This was a 24 hour test with the lights on, and one with the lights off.
7. Weekday test: A full 24 hour test at 2500 S/s to have high quality data of the envelopes of any magnetic disturbances throughout the day. This helped reveal the diurnal variation of the effects of the elevator, nearby power lines going to higher floors of the building, etc.
8. Fire extinguisher test. Inconveniently, on the other side of one of the walls near the Helmholtz cage, there was a large wall-mounted fire extinguisher. It was placed farther away during the test and then back for the control test.
9. Spatial test: The probe of the milligauss meter was moved around in a 3x3 horizontal grid within the Helmholtz cage, at 0.5 m intervals. This was to test whether the magnitude and direction of any magnetic disturbances would be different there.

If we compare this to Figure 7.3, this covers most of the suspected disturbance sources. The only thing that could not be addressed was the bar with power and network cables that runs horizontally across the wall at thigh-height. Of course, any disturbance this item caused would be in the measured data, but since a control test was not possible, its magnitude could not be ascertained. A more detailed overview of the tests can be found in section K.1.

One notable hurdle was that the Python code was too slow to reliably poll the USB-6008 device for each single measurement. Polling the DAQ at more than 50 Hz caused notable slowdown and inconsistencies in sample timing. In response, the software tool was improved to utilize separate threads that pulled batches of pre-defined size from the internal buffer in the DAQ in a round-robin fashion. An extra thread was then used to convert the data from mV to μT and write the data to file.

7.5. Test results

The results from many of these tests were generally full of surprises, both because some seemingly innocuous sources could have a massive impact, and some suspected problems turned out to be no problem at all. The fact that there was a selection of high resolution tests meant that not only could the vectorial properties of magnitude and direction be measured, but also some spectral properties of the disturbances.

The first notable result is that at the testing location there is a static magnetic offset, whose magnitude is roughly of the same proportion as the local EMF. However, the direction of this offset has quite a large angle in comparison with what the local EMF should be. We therefore propose that there is some particular *static disturbance* present at the test location besides the EMF component, which we will define as:

$$\overline{B}_{D(isturance)} = \overline{B}_{M(easured)} - \overline{B}_{EMF} \tag{7.3}$$

These three components are also plotted in Figure 7.6 and Figure 7.7 with the disturbance vector in black. Its magnitude is about 18.4 μT.

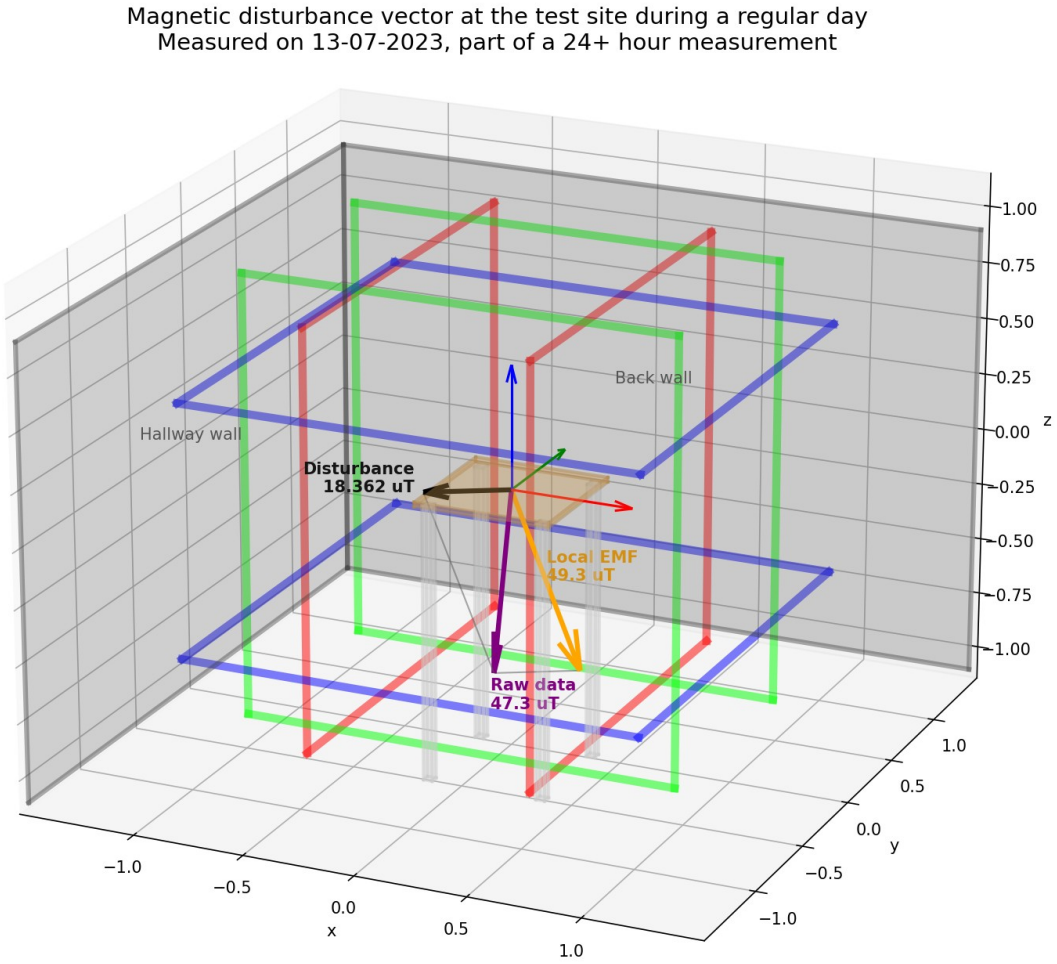


Figure 7.6: Rendered Helmholtz cage elements and walls in the lab. The cage-centered frame C is plotted as a standard XYZ tripod. The orange vector is the local EMF field and the purple vector indicates the raw, rotated measurement data. The difference between these vectors results in the black vector, which is the disturbance vector, and has a magnitude of around 18.4 μT.

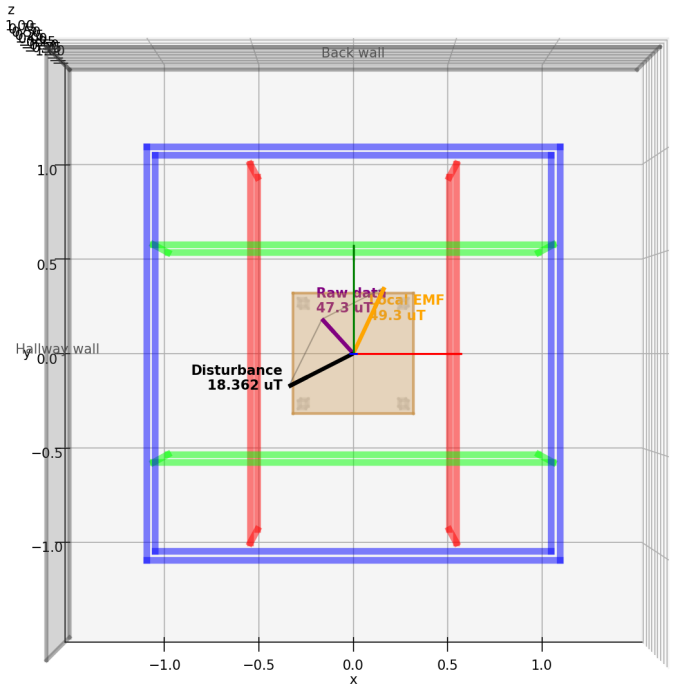


Figure 7.7: A top-view of the situation depicted in Figure 7.6.

The second result is that the dynamic component of this disturbance (the portion of the vector that changes with time) is much lower in magnitude than the static component. This can be seen in Figure 7.8, which shows the three components of this vector in a plot with their absolute values. It is clear that the variation of the axial components is much lower than the static offsets. In other words, over the course of the 24 hours, the disturbance vector is very stable, and could be said to only "lightly quiver" in the XZ-plane.

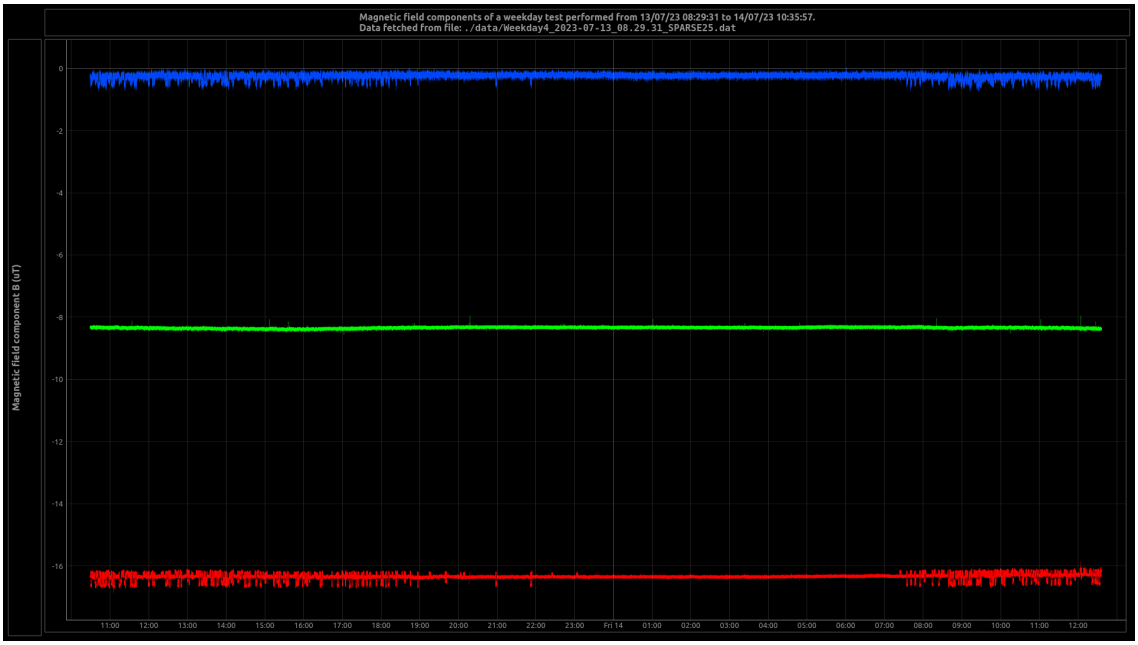


Figure 7.8: 24 hours of measured data, with the three axial components plotted on a common axis.

To sum up, the disturbance vector has the following properties:

```
Mean XYZ:    [-16.358 -8.338 -0.248] uT
SD XYZ:          [0.063 0.017 0.066] uT
Magnitude:                   18.362 uT
```

It is interesting to note that for some reason, the Y component of the test data exhibited much lower variance than the X and Z components. Although it was a point of curiosity, it was never conclusively established why this is so. Also, Figure 7.8 displays the data of a full 24 hour period. There is a strong diurnal difference in the amount of variance; that is to say, the disturbance vector moves much less during the night.

Let us zoom in on this data a bit more to see where the variance is coming from. What we find can be seen in Figure 7.9, which shows the components of the disturbance vector over a half hour period. The curious patterns that can be seen are the result of the magnetic interference of the elevator system, moving faculty employees between floors just after lunchtime.

Even though the elevator shafts are a good 15 meters away from the test location, they are by far the largest contributor to the dynamic disturbance inside the Helmholtz cage. At night, when there are no employees in the building, the variance of the disturbance vector is half an order of magnitude lower than during the day.

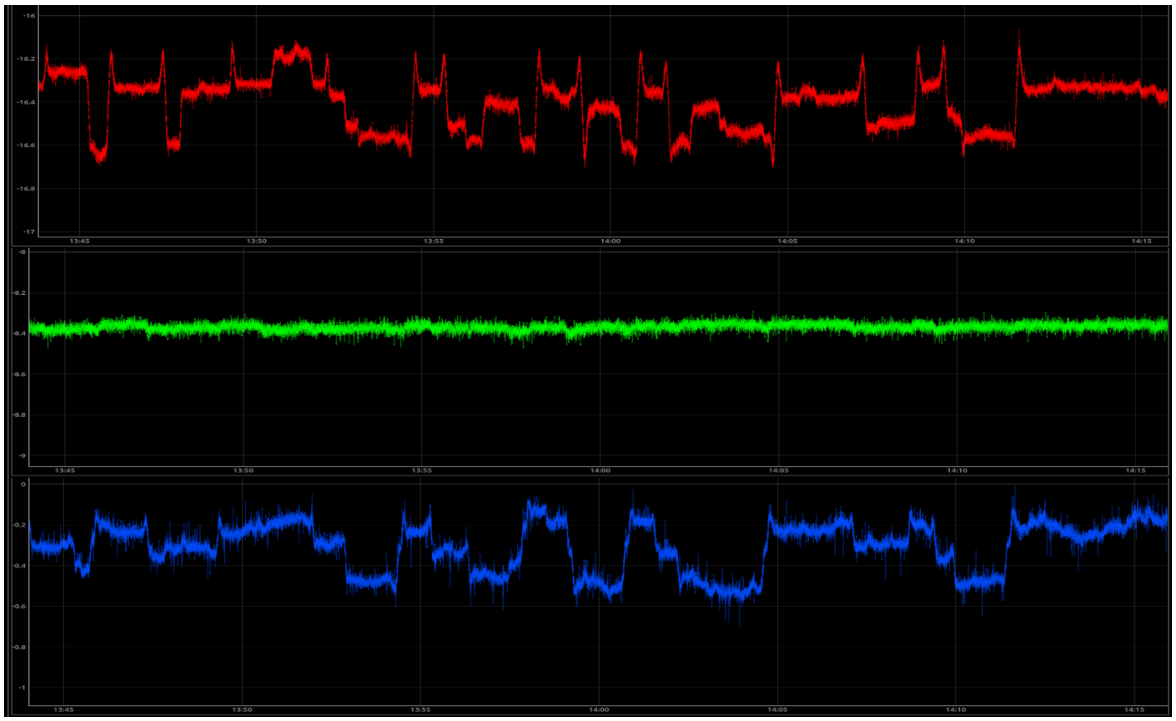


Figure 7.9: The data of Figure 7.8, but zoomed in on a 30 minute time interval around the lunch break in the faculty. The vertical axis scale is in μT

The high-resolution tests at 2500 S/s revealed some smaller dynamic components. Notably, there is a significant 50 Hz component to the magnetic field disturbance, causing a spike in the spectral plot. Furthermore, there are many harmonics of this frequency that also have peaks on the spectral plot, albeit with much smaller magnitudes. Several spectral plots can be found in section K.1.

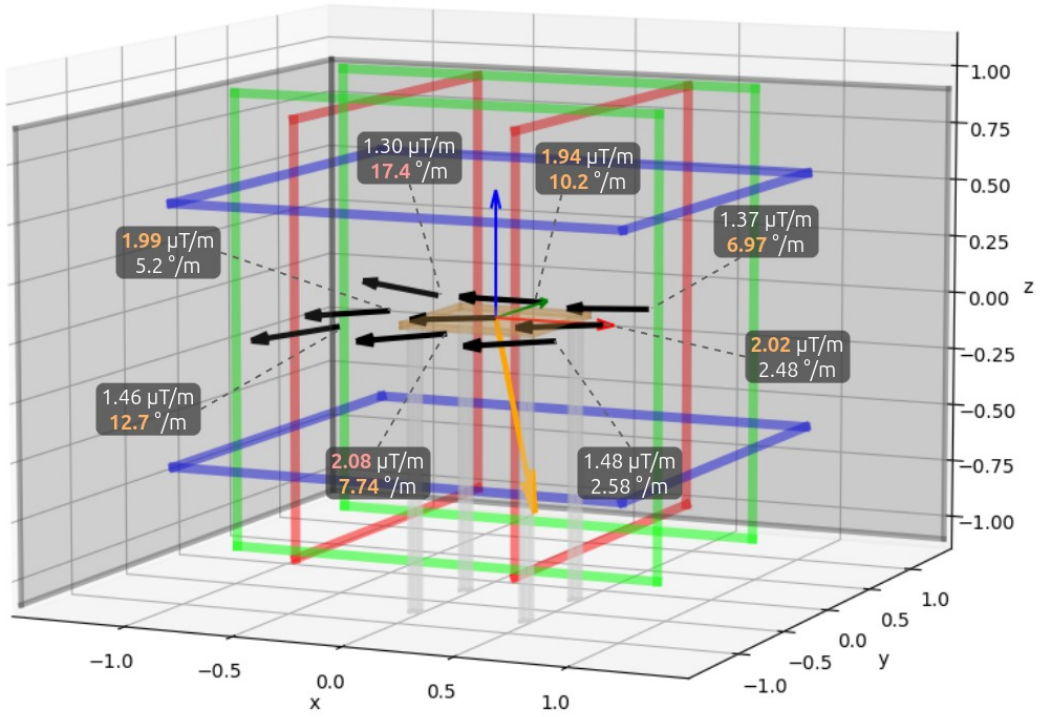


Figure 7.10: The spatial test series: Average disturbance vectors from nine 45+ minute measurements in a 0.5 m XY-grid around the test area. Boxed values are the magnitude difference and the vector angle difference, compared and normalized to the centre value. For example, a value of 1.99 $\mu T/m$ indicates that if we assume linear behaviour, the magnitude of the disturbance vector changes by 1.99 μT for each meter that we move away from the centre in that particular direction.

One of the final tests that was performed was a series of nine 45-minute tests in which the position of the sensor was moved horizontally around the Helmholtz cage. The resulting data is plotted in Figure 7.10. The absolute measured disturbances and with their variance values can be found in Table 7.1. Because the test points are not at an equal distance from the reference point (the centre of the Helmholtz cage), the values were normalized with respect to distance. These values can also be found in Figure 7.10 and give a better indication of how the disturbance varies along the XY-plane around the test location. Note that the variances are consistently around two orders of magnitude lower than the measured absolute values. This suggests that most of what is measured is a static disturbance, which is possible to correct for with good calibration data.

It should however be pointed out that this test has a few areas in which it falls short. First, the test duration was limited to 45 minutes, which means that there was not as much data available to smooth out the variation due to elevator movement as with some other tests. Also, all the test point were on the XY-plane, meaning that the disturbance variation in the vertical direction is still uncharted. The spatial tests should therefore not be considered a full calibration measurement, but are quite useful as an indication of what kind of variation to expect. It is clear that the static magnetic disturbance varies significantly within the test area, which should likely be corrected for during sensitive tests and for testing larger nanosatellites.

Table 7.1: Measured magnetic field properties on the XY inside the Helmholtz cage.

Test point	Distance from centre (m)	Abs. disturbance [μT]	Variance [μT]
Control	0	18.38 (+0%)	0.130
X	0.5	18.59 (+1.14%)	0.184
X -Y	0.707	19.32 (+5.13%)	0.163
-Y	0.5	19.13 (+4.11%)	0.164
-X -Y	0.707	18.95 (+3.12%)	0.159
-X	0.5	18.30 (-0.41%)	0.158
-X Y	0.707	16.87 (-8.22%)	0.144
Y	0.5	18.24 (-0.76%)	0.149
X Y	0.707	17.85 (-2.88%)	0.126

The results of many of the other tests were less spectacular. All are summarized in Table 7.2.

Investigated disturbance	**Contribution to magnetic disturbance**
Elevator movement	Primary source of dynamic disturbance, in absolute terms it is in the order of hundreds of nanotesla. It is non-random but unpredictable in nature, as the elevator movements are non-deterministic. Strong diurnal differences.
Ceiling lights	Significant contributor of periodic magnetic signal at 50 Hz and at higher harmonics. Roughly sinusoidal.
Nearby metal storage rack	No measurable effect
Paramagnetic effects of structural coil elements	No measurable effect.
Paramagnetic effects of test stand ("table")	No measurable effect.
Wall-mounted fire extinguisher	No measurable effect.
Wall-mounted power/network bar	Inconclusive, no control test possible.
Horizontal location inside Helmholtz cage	Significant variation in magnitude and direction of static disturbance vector.
Unknown source(s)	18.4 μT static disturbance vector
Unknown electrical sources	Most of the other 50 Hz components of the noise. No substantial diurnal differences measured.

7.6. Noise rejection

Although the results from Table 7.2 are interesting from an academic point of view, what we are really interested in is minimizing the effects of these disturbances. Back in section 4.4 we discussed some requirements for an idealized magnetic testing system. In this discussion, the requirements that are affected by the effects of magnetic disturbances are primarily **MTS-02** and **SW-11**, the latter of which implies that the magnetic noise may not exceed 500 nT. In addition, requirements **SW-02** and **SW-03** put limits on the amount of variance in both magnitude and direction of the magnetic field inside the Helmholtz cage, and the results of the spatial tests (Figure 7.10 show that these may be difficult to achieve depending on the test environment.

In this section, let us have a look at which strategies are available to suppress these magnetic disturbances and improve the overall signal-to-noise ratio:

1. The first conclusion that can be drawn from the previous section is that most of the disturbance is static, meaning that it can be nullified by **applying a constant negating offset** with the Helmholtz cage during the test. The combined effects of disturbance and EMF are actually lower in magnitude (47.3 μT) than the magnitude of the EMF vector on its own (49.3 μT), so it is feasible that the power supplies of the Helmholtz cage have enough headroom to deliver this negating offset without sacrificing any performance. From a user perspective, it would be helpful if the test system allowed

this vector to be quickly set each time. Another option is to subtract the constant offset from the test data in post-processing, but this approach may not be suitable for tests where the absolute value of the magnetic field matters, such as magnetic dipole measurements.

2. The spatial tests showed that there may be a significant amount of difference in the static offset when moving around the test chamber. This is not something that can be actively corrected by the Helmholtz cage, it can only correct for a single point. If the amount of variation is charted as a function of the distance from the centre, one can correct for the variation by post-processing the data. For some environments, it may be possible to improve the spatial homogeneity of the field by removing the source of the variation, but in the cleanroom of the TU Delft faculty of Aerospace Engineering of TU Delft (AE) faculty, it was not possible to determine these sources.
3. Dynamic effects pose a different problem. The approach for mitigating these effects depends highly on the nature of the signal and its source:
   (a) For the harmonic alternating current (AC) noise from nearby power lines and lamps, the noise can be filtered from the data rather effectively, because it is highly regular. We will look at some approaches in a moment. It was shown that the magnitude of the signal can be somewhat diminished by turning off the ceiling lamps, but this is not a complete solution.
   (b) The noise signal from the trio of building elevators is a more problematic issue, because it is non-deterministic. Since it was the primary dynamic disturbance present in the measurement data, it is important to have a strategy to mitigate it. One such strategy involves mounting an external reference sensor nearer to the elevator that can act as a proxy for the signal. Using this data and some calibrated scaling factor, it may be possible to remove most of the elevator noise from the measurement, either in real-time or in post-processing.
   (c) For the random noise, sample averaging (or another low pass filter) is always an option. Increasing the sampling rate of the sensors and the recording equipment allows filtering of random noise by exploiting the law of large numbers.
4. Finally, it is worthwhile to reduce disturbance simply by preventative measures; if there are plans to construct a satellite testing facility with a Helmholtz cage, simply spending some time measuring the magnetic disturbance at a few different locations can save a lot of hassle down the line. A remote location with few nearby disturbance sources (buildings, electrical facilities, elevators, roads, etc.) and also constructed mainly out of non-magnetic materials would be ideal. Furthermore, once such a facility is built, it should be managed carefully such that if any changes to the environment are made, the calibration of the testing location is not compromised.

A number of these strategies were put into action as part of the measurements discussed in this chapter. First of all, many tests were performed either at high resolution or over a long period of time in order to allow for noise to settle. Of course, this only works if one is interested in static measurement values. For dynamic values, a more refined approach was chosen. To mitigate the effects of the elevator movements in certain tests, either the test was performed in the evening, or a shorter snippet of data was chosen during which none of the elevators were moving.

In addition, for all tests, a Savitzky-Golay filter (SGF) was added to the data during the post-processing stage. The SGF acts like a low-pass filter, filtering out the effects of high frequency noise, but also smooths out the quantization errors of the 12-bit ADC of the data acquisition card. After the high-resolution measurements allowed for spectral analysis of the environment, a number of narrow-band notch filters were also added at 50 Hz and other offending harmonics. The depth of each notch could be set in proportion to the peak witnessed in the spectral plots, which was quite effective in suppressing the harmonic noise.

Table 7.3: Standard deviations of data at various stages of filtering. All values are expressed in μT.

	SD_X	SD_Y	SD_Z
Unfiltered, day	0.0449	0.0449	0.0482
Unfiltered, night	0.0985	0.0549	0.0528
SGF only, day	0.0228	0.0167	0.0263
SGF only, night	0.0195	0.0152	0.0150
SGF and notch filters, day	0.0051	0.0053	0.0085
SGF and notch filters, night	0.0046	0.0018	0.0038

Table 7.3 shows the effect of three mitigation strategies on the standard deviation of noise for each axial component of the measured disturbance (unfiltered, SGF, and SGF+notch filters, and day/night). These figures are based on a 10 second snippet during which the elevators were still, and a few datapoints at the domain edges was discarded. For this reason, the standard deviation was calculated using the middle 80% of the data only.

One can see that after applying the Savitzky-Golay smoothing and the set of second-order notch filters, the standard deviation is down to below 10 nT. This is an indication that if one is interested in performing magnetic measurements at the Helmholtz cage setup with a sensitivity of 1 nT, one can get fairly close by testing at night (to avoid elevator noise), and by filtering out the periodic noise from the AC power network. The filtering method applied here is fairly basic, and it may be possible to get even closer with better filtering techniques.

However, it should be mentioned that these results should be taken with a healthy amount of salt. The aforementioned results are based on several unspoken assumptions, such as:

1. The massive uncertainty of the sensor data caused by the 12-bit resolution of the ADC was magically fixed by the SGF.
2. The attenuation around the notch that is inherent to all notch filters has no impact on the quality of the data.
3. A 10 second interval is sufficient data to draw the aforementioned conclusions.

It may therefore be that the numbers in Table 7.3 are rather unreliable. For example, let us not gloss over the fact that the unfiltered data during the night has a standard deviation higher than the one during the day, which is difficult to explain with what has been discussed up to now. The bottom line is that without better data, it is not possible to know for certain how accurate Table 7.3 is. This is left to further research to determine.

However, what can be concluded with some degree of confidence is that a majority of the variation seen in the signal is caused by relatively few sources (elevator and electronic devices) and that the extent of this disturbance can be known fairly well. While this may not be true in general for other locations, it is a sign that despite sub-optimal conditions of the work environment, magnetic testing can be done to a relatively accurate degree, given the right testing equipment and approach. Also, although somewhat unsurprisingly, the filtering methods can be used effectively to improve the quality of the measurement data of a magnetic testing system such as this. Appendix K contains a lot more information about these approaches. If we consider requirements **MTS-02**, **SW-03** and **SW-04**, and **SW-11** once more it seems plausible that with the right combination of mitigation strategies, these requirements can be met.

7.7. Conclusions

In conclusion, this chapter discussed a thorough investigation into the ambient magnetic properties of the AE faculty cleanroom. Many interesting observations could be made, which amount to the following:

1. A nearly horizontal **magnetic disturbance vector with a magnitude of** 18.4 μT could be **consistently** measured at the test location, although its source remains unknown.
2. This magnetic disturbance vector seems largely **uncorrelated** with the potential sources of magnetic disturbance that were suspected beforehand. This includes a steel storage rack, a large

fire extinguisher placed nearby, and the metal structural elements of the Helmholtz cage itself.

3. The magnetic disturbance **is** different in both direction and magnitude elsewhere in the Helmholtz cage. It is not clear why exactly, and due to a lack of time, these measurements are currently taken at face value.
4. The variation of this disturbance is in the **order of hundreds of nanotesla**, or $O(0.1\ \mu T)$. This observation alone indicates that if this disturbance is compensated for, **it allows for magnetic testing in the Helmholtz cage with sub-microtesla sensitivity**.
5. In the X and Z directions, the variation of the disturbance is several times **larger during the day**. The vast majority of this difference can be attributed directly to the **movements of the building's elevator system**, which are not in use at night.
6. Aside from elevator movements, a large portion of the remaining variance is **harmonic and highly correlated with the 50 Hz oscillation of the power network and its harmonics**. The author applied some basic **filtering techniques** on the data, which **reduced the data variation to nanotesla levels** (near or below $O(10\ nT)$).
7. When on, the **ceiling lights in the lab have a significant effect** on the amplitude of the harmonic noise. Their effect is expected to be easily filtered along with the rest of the harmonic noise.

7.8. Open questions and further research

Despite the thorough testing approach there were some limitations, and some questions remain unanswered:

1. The real source of the 18.4 μT disturbance was not identified, although this likely matters little for the purposes of testing.
2. Some harmonic noise was identified that does not seem directly correlated with the power network, most notably at 3.83 Hz, 6.59 Hz, and at 377.14 Hz. Its source remains unknown.
3. No filtering technique was developed to filter out the elevator movements during the day. For future measurements, it should be possible to place reference sensors elsewhere in the cleanroom, and use these to correct measurements taken at the Helmholtz cage. However, this is left for future research, and the author currently recommends testing at night as a work-around.
4. The milligaussmeter used in the testing was not absolutely calibrated beforehand, although its basic functioning was verified beforehand with a digital oscilloscope.
5. The test setup was limited by the 12-bit bit depth of the data acquisition card. During data processing, the data was smoothed with a Savitzky-Golay filtering to ameliorate this issue somewhat, but the actual effect of the low bit depth on the quality of the data remains unquantified.
6. The magnitude and direction of the magnetic field should be mapped better elsewhere in the Helmholtz cage. The spatial tests that assessed this were the only ones that showed significant impact on the disturbance vector. The variation in the vertical direction was not investigated altogether due to a lack of time, but may also have an impact on future tests. Furthermore, it may indicate that there is a need for a way to quickly map the magnetic vector fields inside the Helmholtz cage at many locations, which would have to be developed.
7. The Savitzky-Golay smoothing should be replaced with a different type of smoothing. It has the effect of a comb filter, acting as a relatively wide notch filter at regular intervals. If this is done during real satellite testing, important spectral properties of this satellite may be missed because it is made invisible by falling within one of the "teeth" of the comb filter. A different type of smoothing filter should be found that does not so discriminately attenuate certain frequency bands. Or, if nothing else, the Savitzky-Golay should be tuned such that it attenuates at intervals of 50 Hz, but since the notch width of the comb filter is so much wider than the more surgical notch filters, it is likely far from an optimal solution. It may be that the Savitzky-Golay smoothing should just be replaced by a low pass filter and the previously discussed battery of notch filters.

8

Demonstrator System Requirements

Previously in chapter 4, we looked at magnetic testing of small satellites, and proposed an idealized testing system and a description of its functionality and technical requirements. The last sections of chapter 3 left us with the notion that a magnetic testing system for CubeSats should be designed and constructed. We have also established roughly which aspects will drive the design, which at this point can be interpreted as top level requirements. In this chapter, the requirements will be defined more systematically. Where appropriate, the lessons of chapter 6 and chapter 7 concerning the performance and limitations of the legacy Helmholtz cage system will also make their way into the requirements.

The design of the idealized system presented in Figure 4.5 comprises far too much work for the context of a thesis. For this reason several aspects of this systems were de-scoped, until a smaller system remains. Henceforth, we will refer to this smaller, scoped system as the Demonstrator System (DS), as it will be used in the research that underpins this thesis. The design of the demonstrator system is expected to be far more achievable for a single thesis, whilst still providing enough input to answer the research question, and also be expanded upon easily for future research and development.

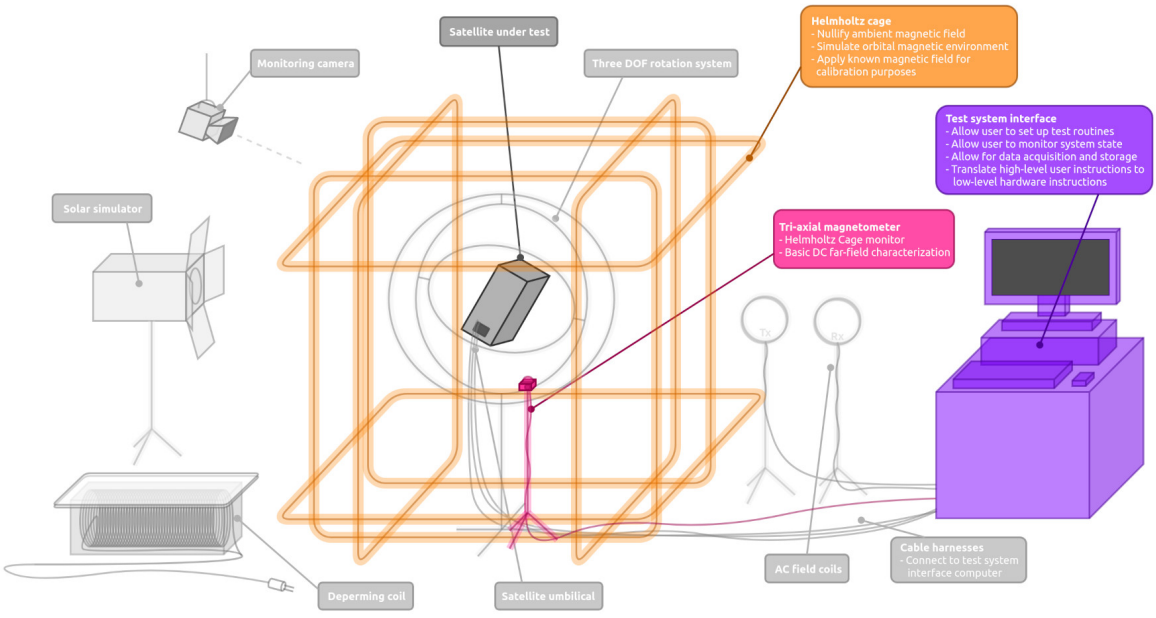


Figure 8.1: After scoping down the idealized test system, we are left with a subset of it, which we call the demonstrator system.

The scope of the Demonstrator System can be seen in Figure 8.1. It now mainly consists of a Helmholtz cage, a tri-axial magnetometer, and a test interface. With the Helmholtz cage, the local Earth magnetic field can be nullified and simultaneously the satellite can be subjected to the simulated magnetic conditions of a certain Earth orbit. The inputs for the magnetic actuation are handled by the interface computer, which convert a set of high level instructions (such as orbital elements, rotational properties of the satellite, simulation resolution, etc.) to low-level hardware instructions (voltages and currents for each Helmholtz coil pair). The magnetometer is there to calibrate the Helmholtz cage, to measure the local Earth magnetic field, as well as other local field disturbances which can then be nullified. In addition, the magnetometer can also be used as a way to measure the satellite's static far-field magnetic emissions. This could be done much better with an array of such sensors, but for now, it will still be a helpful additional feature for a sensor that is already included for other reasons.

The other subsystems from present in the idealized test systems have been de-scoped. First, the subsystem for providing rotational degrees of freedom to the satellite under test has been descoped because this system in particular can be very complex, especially for two or three degrees of rotational freedom. As as concession, we will add a requirement to the DS that the magnetic fields used to simulated orbital motion must also incorporate virtual satellite rotation.

Next, the solar simulator was de scoped because of two reasons: its usefulness is somewhat lessened by the lack of free rotational motion of the satellite, and also because its implementation would take too much time. However, it is expected that it is straightforward to incorporate this subsystem as an upgrade at a later point in time.

The coils necessary for AC robustness tests have also been de scoped, as they are used for one specific test, and nothing else. In addition, they are relatively separate from the rest of the test system, and can therefore be easily added as a later upgrade. The same is true for the deperming coil, which is a separate component that can be used to improve the quality of the magnetic tests, but its absence will not prevent one from performing these tests.

8.1. Demonstrator System Requirements

To design the demonstrator system, we must determine all relevant requirements. We have already looked at requirements for the idealized system in section 4.4, but these will not be sufficient here. First of all, since several aspects of the idealized system have been left out of the demonstrator system, the related requirements must also be discarded. Furthermore, the demonstrator system has a somewhat different purpose than any satellite testing system, as we are also interested in its relevance in answering the research question, which will impose extra requirements on the system (for example related to documentation and open-source development). Finally, since the demonstrator system will be built at the cleanroom of the Faculty of Aerospace Engineering of the TU Delft, there are several more requirements to add to the demonstrator system relevant to the conclusions from chapter 6 and chapter 7.

In the following sections, the demonstrator system requirements will be listed. For proper traceability, all requirements have a unique ID with the following format:

`<category>-<requirement number>-<requirement version>`

There are four categories:

1. Top-level system requirements (`DS`) that apply broadly to the demonstrator system as a whole.
2. Additional requirements related to the performance (`PER`) of the system.
3. Additional requirements related to the electronic and hardware design (`ELEC`).
4. Additional requirements related to the software (`SW`) aspects of the design.

8.1.1. Top-level Requirements

For the top-level requirements, we will directly copy the idealized testing system requirements that are still relevant. We change the requirement category from "`MTS`" to "`DS`", but the requirement numbers are kept the same, so that it is easy to see how these requirements are derived from the idealized test system requirements in section 4.4.

ID	Description	Verif.
DS-01-01	**The DS shall have a magnetic control volume large enough to envelop a typical CubeSat. The size of this control volume shall be a cube with side lengths of at least 415 mm.** This is based on a standard 12U (2x2x3) CubeSat, whose cube diagonal is 412.3 mm. In a cubic control volume with sides of this length or above, such a CubeSat size could assume any rotational position whilst remaining inside the control volume. Verification of this requirement can be done by testing if net B-field can be nullified in control volume.	A
DS-02-01	**The DS shall establish a neutral magnetic environment, reducing any ambient magnetic fields inside the control volume down to below 500 nT in any direction.** In Delft, the local Earth magnetic field vector has a magnitude of around 50 μT. We hereby require that this field can be nullified to 1% or less of this.	T
DS-03-01	**Inside the control volume, the magnetic field strength uniformity shall be within 1%.** For clarity: taking the magnitude of the field strength at the exact middle of the control volume as reference, we require that the field magnitude at any point in the control volume may not deviate more than 1%.	A, T
DS-04-01	**Inside the control volume, the magnetic field directional uniformity shall be within 1%.** Same as DS-03, but for the direction of the field vectors, in essence the combined curl and divergence properties of the field in the control volume.	A, T
DS-05-01	**The DS shall allow the satellite to be suspended inside the control volume.**	D
DS-14-01	**The DS shall be able to operate automatically according to a pre-programmed schedule, without requiring manual user input during testing.**	D
DS-15-01	**The DS shall have a straightforward and convenient interface, suitable to be usable by space engineering students.**	D
DS-16-01	**The DS shall be safe to operate, by users unfamiliar with the inner workings of the system and at the level of engineering students.** For example, this implies that a user who is unfamiliar with the inner workings of the system can perform tests without being at risk of receiving electrical shocks, starting a fire, etc.	D
DS-17-01	**The DS shall facilitate time-efficient testing, allowing for tests to be performed in the span of half a day, wherever feasible.**	D
DS-18-01	**The total cost of the DS shall not exceed €10k.** Given that the DS is smaller in scope, this requirement has been tightened compared to MTS-18. The 10k euros is a baseline figure that assumes a hypothetical CubeSat project budget of €200k. Of this budget, it is assumed that 5% or less of this can be reasonably justified to perform magnetic testing with a system such as the demonstrator system.	A
DS-19-01	**The DS shall be composed of COTS components with high availability wherever feasible.**	D

8.1.2. Performance Requirements

ID	Description	Verif.
PER-01-01	**The DS shall be able to simulate arbitrary field vector up to at least 210 μT inside the control volume, on top of the ambient field corrections.**	T

	This is about three times the strongest expected in-orbit Earth magnetic field (EMF), which is $70\mu T$ above the magnetic poles	
PER-02-01	**The DS shall be able to correct for local magnetic field variations down to 500 nT** 500 nT constitutes about 1% of the total expected ambient field. Note that "correct" does not necessarily imply that this is done with magnetic actuation: if it can be corrected for by post-processing the test data, this may be sufficient.	**T**
PER-03-02	**The DS shall allow for rotating fields up to speeds of 20 rpm.** Used to be 120 rpm. Analysis revealed that this is possible in theory, but the slew characteristics of the power supplies prevented this, so the requirement was slackened.	**A, T**
PER-04-01	**The DS shall be able to determine the generated field strength at the control volume with a resolution of at least 10 nT.** This is primarily for self-monitoring the Helmholtz cage, as one of the conclusions of chapter 7 is that the expected ambient magnetic disturbance in at the location of the test system will fluctuate, and so monitoring and/or periodic re-calibration will be needed, which this requirement facilitates.	**D**
PER-05-01	**The DS shall be able to determine the field direction at the control volume.** See also requirement PER-04	**D**
PER-06-01	**The DS shall be able to monitor temperatures at critical locations.** This assumes that such critical locations exist. If it is found through testing that thermals pose no problems anywhere in the system during nominal operation, this requirement can be discarded.	**D**
PER-07-01	**The coil temperature in the DS shall not exceed $60°C$ during nominal operation.** This is both to prevent the coil elements from degrading over time due to repeated thermal expansion, but also to limit the effects of ohmic heating on the coil performance.	**T**
PER-08-01	**The current direction through the coils must be able to be switched in under 1 ms.** Stakeholder requirement from thesis supervisor. This requirement implies that pretty much all relay-based switching solutions can be discarded.	**T**

8.1.3. Electronic and Hardware Requirements

ID	Description	Verif.
ELEC-01-01	**The design coil current shall be 5 A.**	**D**
ELEC-02-01	**The coil current shall not exceed 6 A.** Based on the thesis by Poppenk [46], the copper coil elements are rated for up to 6 A	**T**
ELEC-03-01	**The Helmholtz coils shall be driven by a current-controlled source.** This is to ensure that the magnetic field output remains constant even with variations in load impedance (for example due to ohmic heating).	**D**
ELEC-04-01	**The Helmholtz coils shall be driven by a voltage source capable of at least 40 V.** By virtue of Ohm's Law, the voltage required for driving 5 A (see ELEC-01) through a 7 ohm load equals 35 V. An extra 5 V of headroom is allocated for overcoming inductive impedance in the coils, resulting in a 40 V requirement.	**D**
ELEC-05-01	**The Helmholtz coils shall be driven by a bidirectional current.** This follows from TOP-02 and PER-03, bidirectional current flow is required to ensure that magnetic fields generated by a pair of Helmholtz coils can both point in positive and negative direction along the corresponding axis. If a unipolar current supply is used to drive the coils, a polarity-switching circuit must be incorporated.	**D**
ELEC-06-01	**The system shall be able to measure the current applied to each coil pair with a resolution of 10 mA.**	**T**

	Whilst closed-loop control is not being made an explicit requirement, this requirement should make provision for closed-loop control in the future. In addition, the current data can be used as a diagnostic for system verification.	
ELEC-07-01	**The power supplies shall be protected from inductive voltage transients by at least one layer of protection.** Although multiple layers are better. There are many possible ways of implementing this such as filtering, Zener or TVS diodes clipping circuits, gas discharge tubes, crowbar circuits, etc.	**A, D**
ELEC-08-01	**The system shall be capable of handling inductive voltage transients without damage**	**T**
ELEC-09-01	**Structural components and chassis elements near the electronics shall be grounded properly.**	**D**
ELEC-10-01	**The total system power consumption shall not exceed 1 kW.** The cleanroom has 3.5 kW allocated on its switchboard group. This requirement ensures the total power consumption will not consume more than about one third of this. This is a target power consumption, the limit can be raised if needed.	**T**
ELEC-11-02	**The voltage ripple though the Helmholtz coils shall not exceed 10 mVpp during nominal operation.** For a 7Ω load, this will result in a current ripple of about 1.5 mA, which is deemed adequate considering this will be roughly in the same order of magnitude as the power supply accuracy.	**T**
ELEC-12-01	**The power supplies should be current controllable using an analog control input voltage.** This requirement is imposed to ensure that the design can be made in a way that allows a large variety of different power supplies to be used for the system, but provided they have analog current control input. An analog voltage control input is not strictly required, but is better, and is typically implemented in a power supply that also features analog current control (CC) anyway.	**D**

8.1.4. Software Requirements

ID	Description	Verif.
SW-01-01	**The software shall have a clear separation between firmware and user interface.** This is for portability and organisation. It allows for the same user interface to be used with a different Helmholtz cage (hardware and calibration parameters), and it allows for the user interface (UI) to be improved without touching the firmware. It also allows for multiple types of UI (command line interface (CLI), graphical user interface (GUI), web interface, etc.) that all can interact with the same firmware.	**D**
SW-02-01	**If a graphical user interface is created, it should not supplant the underlying application programming interface (API) or CLI.**	**D**
SW-03-01	**The software shall enable automated data acquisition.** As in: fire-and-forget, the user should be able to configure this once and thence not be concerned by configuring the data recording process.	**D**
SW-04-01	**The software shall enable for reliable data acquisition.** The risk of data loss should be minimized (so minimal saving to random access memory (RAM), no risk of accidental file overwriting, etc.) and sampling jitter is to be minimized.	**D**
SW-05-01	**The source code of the software shall be open to the general public.** Use an open-source license and demonstrate public access (e.g. via GitHub)	**D**
SW-06-01	**The software shall allow for automatic EMF rejection in the Helmholtz cage.**	**D**

SW-07-01	**The software shall allow for manual control of field in the control volume.**	**D**
SW-08-01	**The software shall allow for pre-programmed magnetic field envelopes to be played back.**	**D**
SW-09-01	**The software shall include the capability to account for satellite rotation for the simulated magnetic field envelopes.** In the absence of rotational degrees of freedom for the satellite under test in the test system, it would be useful to simulate the virtual motion of the satellite and apply this to the simulated magnetic field. This requirement is added since it was chosen not to inherit requirement MTS-06	**D**
SW-10-01	**If the software contains any checks that are required for safe operation, these shall not be directly accessible through the user interface.** Ideally, all safety-related design should be done in hardware (see ELEC-14). In cases where it is unavoidable to implement safety-related checks and limits in software, it should not be done directly in the user interface, where an inexperienced user may accidentally turn them off.	**D**
SW-11-01	**The software shall allow for data acquisition of at least three magnetic sensors at a sample rate of at least 20 S/s.** For reasons related to the Nyquist limit, the sample rate should be at least twice the frequency given by PER-09.	**T**
SW-12-01	**The software shall be sufficiently documented such that other engineers unrelated to the project could use it.**	**D**

8.2. Optimization and other considerations

The thesis objectives discussed in section 3.2 make it clear that there are several directions for optimization. As is often the case in engineering, one can only optimize in multiple directions so far until optimization for one design aspect comes at the cost of another. For example, if one desires of the system that it can perform tests that are representative of reality, the complexity of the system will usually increase with increasing capability. But as a result, typically the cost of the system will increase also, which is a point of contention if one is also trying to optimize for cost.

There are multiple ways to address this. One is to set performance targets for design aspect that must be optimized, create a design that achieves all these targets, and then iterate to achieve a more optimal design. However, another way of approaching the design is to leave the optimization goals as "soft requirements" or "design desires". During the design process, every major design decision will then be made in explicit consideration of these design desires, and traded off in accordance. This thesis makes use of the latter, simply because there are many aspects of the design that are interesting to optimize:

- Cost (lower is better)
- Complexity (lower is better)
- Time associated with operation (lower is better)
- Capacity to perform representative testing (higher is better)
- Reliability of system and its performance metrics (higher is better)
- Design adaptability / portability (higher is better)
- Design reproducibility (higher is better)
- Performance efficiency, for example power efficiency (higher is better)

To make a case for why this style of design is used, consider that for every design decision to be in accordance with these optimization goals, one would need to have an idea of the "exchange rate" of each optimization goal in terms of every other one. There are no hard rules for this, and these exchange rates are often constrained by practical considerations anyway, which is why the author argues for this style of engineering here.

To give an example: in this thesis, one of the design decisions that was made was to try to make the system relatively adaptable. As part of this strategy, it was decided that the power supplies should be controlled by a computer system using analog input signals. Many different power supplies models have such an analog control input, and so implementing it this way means that somebody else can take this design and substitute different power supplies instead of having to purchase three units of the exact same model used in this thesis. It may be that the designer already has three suitable power supplies in-house that can be used, further reducing cost on their side, and in this way increasing the chances that they will decide to build this system.

However, this necessitates that the hardware implementation of the design has at least three digital-to-analog converter (DAC) channels by which to control these, and more realistically, six such channels (one for current control, the other for voltage control). This does increase the complexity and base-line cost of the design. The author fully believes that in this case, the increased complexity and cost are worth the extra design adaptability. However, it would have been very difficult to have written requirements beforehand that would have led one to make this exact decision.

So, in addition to the demonstrator system requirements, we also list a number of extra design considerations, which are not implemented as requirements, but are used to guide decision making in places where this does not conflict with any other requirements. The main reason for explicitly listing these here is so that they can be referred to where relevant.

ID	Description
EXTRA-01	**It is desirable that the system has low cost.** Requirement DS-18 already puts an upper bound on the cost, but any further cost optimization is of interest given the research question.
EXTRA-02	**It is desirable that the system shall allow for time-efficient testing.** Similar to EXTRA-01, this is an optimization goal related to the research question. As a rough target, it would be good if a satellite can be characterized in the span of a few hours (i.e. one afternoon session).
EXTRA-03	**It is desirable that the system shall be designed to be modular, with an eye to future expansion and improvement.** This is not only important to increase the chances of user adoption of the design, but also to ensure that it can be built upon as a baseline design. The designer should anticipate likely future expansions or features and pay attention that the design would not unduly limit such expansion.
EXTRA-04	**It is desirable that the system shall be composed of components that are likely to be readily available worldwide and in the years to come.** In other words: use specialized components only when absolutely necessary. In line with requirement DS-10, this improves the likelihood that the design will be adapted broadly and is still relevant in the future.
EXTRA-05	**It is desirable that the system is well-documented blueprints, easily accessible to any interested party.** This is more a requirement on the design process rather than the design content, but it will have design implications, for example the choice of open-source vs. closed-source software frameworks.
EXTRA-06	**It is desirable that care is taken in the design that naive user input can not lead to damage to the system.**
EXTRA-07	**It is desirable that the electronics are self-explanatory by form and incorporates measures to reduce user error.** This includes anything from clear silk-screen on any printed circuit boards (PCBs), unambiguous connector usage, indicator LEDs where needed, etc.
EXTRA-08	**It is desirable that any low level (safety) functionality is implemented in hardware rather than software, where this is reasonably possible.**

For example, an analog Schmitt trigger chip is preferable to a conditional software ISR, same for debouncing and noise filtering. It is especially relevant for safety features, such as emergency shut-off implementations. However, this should be done within reason, especially where electronic engineering knowledge is lacking.

EXTRA-09 **It is desirable for user software to be created with the underlying assumption that users will not have read the documentation.**
It is an old and tired reality of software development that users will not read the manual before using the software. This "design desire" stipulates that the software should embrace this fact. Clear user feedback from the software, sensible layout, good visualizations will help the user figure it out by themselves without having to consult much documentation.

EXTRA-10 **It is desirable that the software is be operating system (OS) agnostic.**
It is not appropriate for this to be a hard requirement, as it is a lot more important to build a working system than to chase universal OS compatibility for such a niche design targeted at fellow engineers. With this in mind, a working system that is OS agnostic is still preferable over a working system that is not, since it may affect user adoption.

8.3. Conclusion to Part II

We have reached the end of part II and are about to move onto part III. Before we do so, we take a brief moment to consider where we are in the overall arc of the thesis work. Part I dealt with defining the problem, and zooming in on an avenue of research. An 'idealized test system' was proposed for magnetic testing of nanosatellites.

In part II, we have looked at the design of a demonstrator system, which is a simpler version of the idealized system, in order to fit the contents within the scope of a thesis. To design the demonstrator system, a number of other aspects were investigated that would be relevant to take into account in the design of the demonstrator system. The demonstrator system was build based on an older Helmholtz cage design, and in order to investigate how this older design could be incorporated, we took a closer look at it during its disassembly.

In addition, we looked at the ambient magnetic field properties of the test location, found a number of sources of disturbance. These disturbances were quantified by a series of measurements and a number of mitigation strategies were proposed, such as data post-processing with filters.

The above is summarized in Figure 8.2. In part III, the demonstrator system will be designed in detail.

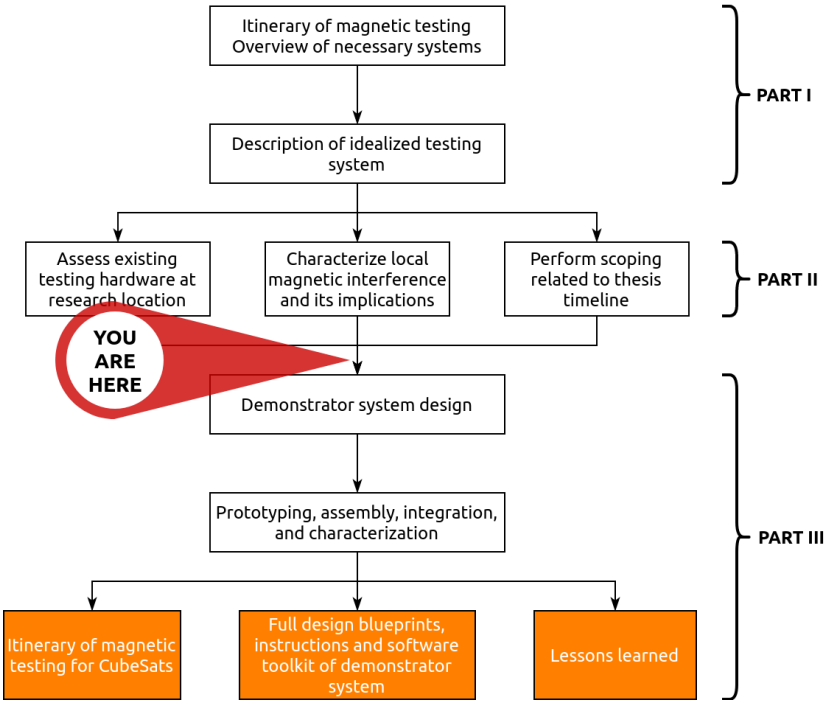


Figure 8.2: Another look at Figure 3.1.

Part III

THE WHAT - Designs, Results, and Conclusions

9

Demonstrator System Layout

In the previous chapter, we formulated the requirements for the design of the Demonstrator System (DS), along with several extra considerations. Part III of this thesis, of which this chapter is the first, will cover the design, implementation, and validation of this Demonstrator System .

Although the DS was designed as a whole, several aspects of it are reasonably self-contained, and are therefore covered in their own chapter. For example, the electronic design of the system is covered separately in chapter 10, and similarly the software design is covered in chapter 11. However, in this chapter we start with a more general look at the system of a whole and discuss how some of the design requirements were translated into design choices.

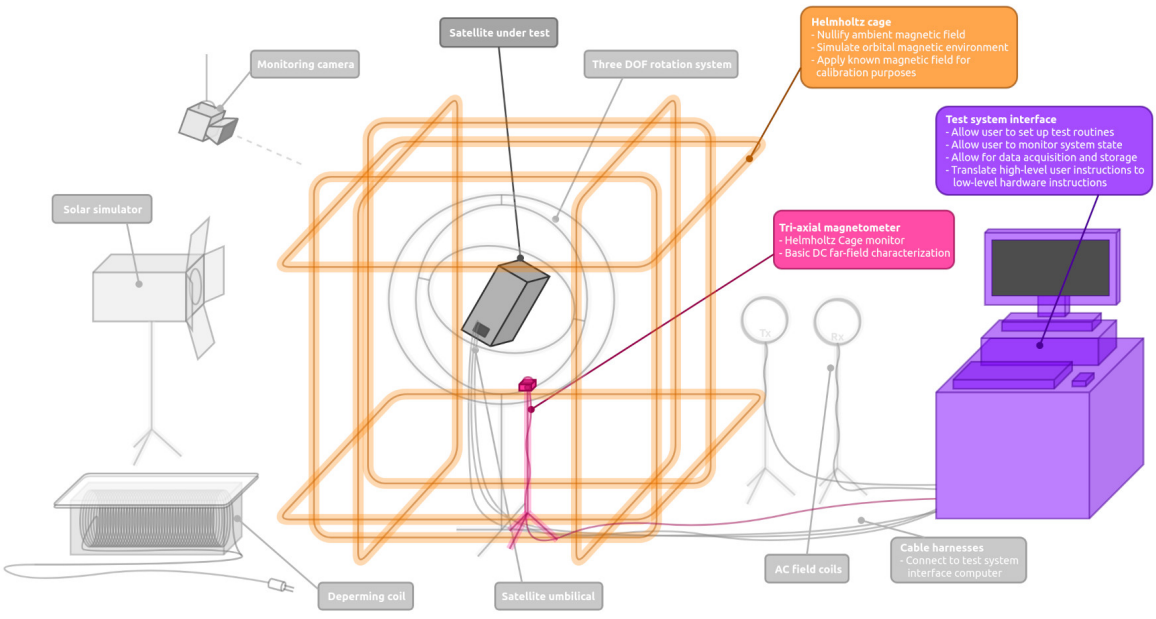


Figure 9.1: After scoping down the idealized test system, we are left with a subset of it, which we call the demonstrator system.

To start, Figure 9.1 shows the system overview of the demonstrator system from chapter 8. As mentioned before, the demonstrator system is a scoped version of a larger idealized magnetic testing system, described in chapter 4. In terms of functionality, the demonstrator system is primarily focused on the cancellation of local magnetic fields at the test location, and simulation of magnetic fields that can be

used for testing purposes. In addition, it is also meant as a base system, with the expectation that it can be expanded and upgraded later on.

9.1. Hardware overview

From Figure 8.1, we can already identify several hardware elements that the demonstrator system will have:

1. The frame and coils for the Helmholtz cage
2. Though not explicitly depicted, the Helmholtz cage also requires the electronics to drive the coil current
3. A tri-axial magnetometer to perform measurements inside the Helmholtz cage
4. A user interface for the test system, most likely a computer with an operating system, with which the user can issue commands, perform calibration, interact with measured data, etc.

The demonstrator system is to be built at the faculty of Aerospace Engineering of the TU Delft, in the dedicated cleanroom used for satellite development. In chapter 7, we had a detailed look at the suitability of this test location with regards to magnetic testing. There are some problems with this test location, such as an 18.4 μT static ambient magnetic field disturbance, as well as a much smaller dynamic magnetic field disturbance, in the order of 0.1 μT. The static magnetic field is known, and can therefore be cancelled along with the local Earth magnetic field (EMF) vector. The dynamic disturbance is primarily caused by the movements of a nearby elevator system, and can be treated as unpredictable noise. The test location is still adequate for coarse testing, but for more precise testing it is recommended to test at night, when there are no people using the elevators. In the future, it is recommendable to install a reference sensor in the lab that will measure the noise signal from the elevator, so it can be used to eliminate it from the test data, either in real-time or as a post-processing step.

Several other sources of magnetic disturbance were identified, but the magnitude of these was much smaller than the contribution of the elevator. In addition, magnetic interference from the local electric network also makes its way into the test location, which are oscillatory in nature and manifest at the harmonics of the 50 Hz alternating current (AC) frequency. It was demonstrated that much of these oscillations can be effectively filtered out of the measurement data (see section 7.6), and are not expected to present as an obstacle to any magnetic testing with the demonstrator system.

As covered in detail in chapter 6, there is already an older Helmholtz cage at the cleanroom, whose hardware can be used wherever appropriate. At the end of chapter 6, several conclusions were drawn about the suitability of this hardware:

1. The magnetic coils of the system, as well as the mounting frames that support them are adequate to be used for the demonstrator system. Some simulation work (see subsection 6.2.4) reveals that these coils will be able to match the Helmholtz cage requirements on control volume size and field homogeneity (**DS-01**, **DS-04**, and **DS-05**). The coils also allow a high enough current to achieve the field strength requirement of **PER-01**. The magnetic coils of the old system **WILL** therefore be used as part of the demonstrator system.
2. The driver electronics of Helmholtz cage were investigated in detail in subsection 6.1.3 and were found inadequate for the demonstrator system, for several reasons:
   (a) Several of the components in the circuits were used outside their design specifications, notably the switching relays, which were only rated for 2 A.
   (b) The relay-based polarity switching is too slow to be used in accordance with **PER-08**, and this timing requirement is such that it does not allow for any drop-in relay replacements. Instead it implies that, a polarity switching implementation must be designed that does not rely on relays, which heralds a more fundamental change to the driver electronics. Discarding the switching functionality altogether is not possible, since bipolar coil current capability is a system requirement (**ELEC-05**).
   (c) The power supplies used to drive the coils are three Delta Elektronik ES-030-10 units, each rated for 10 A at 30 V. Although these supplies can supply 10 A, they cannot push all this

current through the coils with only 30 V available. They therefore fail **ELEC-04** and by extension **ELEC-01**. In addition, these power supplies are a specific factory option which allow for control over RS-232, but not via analog input, thereby failing requirement **ELEC-12** also.

The driver electronics of the old Helmholtz cage system therefore **WILL NOT** be used for the demonstrator system. Before the start of the thesis, new power supplies for this express purpose were already purchased. These are three BK Precision 1685B units, each capable of supplying 5 A at 60 V. They are unipolar supplies, meaning they can supply this current only in one direction. For this reason, the polarity switching electronics for driving the Helmholtz cage will have to be redesigned also. An overhaul was already needed due to some other requirements, such as **ELEC-07**.

3. The tri-axial magnetometer that came with the old Helmholtz cage system is an Alphalab milligaussmeter that has a resolution of 1 nT. This instrument **WILL** also be used in the demonstrator system, as it is a very precise and reliable instrument with a long cable between measurement probe and base unit, allowing it to be used in many contexts and configurations. Its only caveat is that it outputs its data as analog signal, thereby requiring three dedicated analog-to-digital converter (ADC) channels to read out the data digitally, but this is deemed acceptable.
4. The interface computer that was already used for the old Helmholtz cage system **WILL NOT** be used. It is old and slow, and was not even allowed access to the local network by the IT department due to its antiquity.
5. In addition to the previous point, the control software that was used in this system was an old Labview-based control program. It also **WILL NOT** be used in any way, due to it being made specifically made for the old system, rather than a more generic interface that can be used broadly. There was also no documentation available for this software in any way, which means that extra work would be needed to investigate how the software functions. The dependence on the propriety Labview back-end also complicates requirement **SW-05** and consideration **EXTRA-10**. It was decided to discard this whole aspect of the old system, and start from scratch with the relevant requirements for the demonstrator.

9.2. Degrees of separation

From an early point in the design process, it was decided to implement a particular degree of separation between user and system hardware. This is shown in Figure 9.2. The user interacts with a normal computer near the test interface, but this interface computer is not directly connected to the hardware of the system, such as the Helmholtz cage power supplies. Instead, an intermediary controller, such as a microcontroller or microprocessor is connected in between.

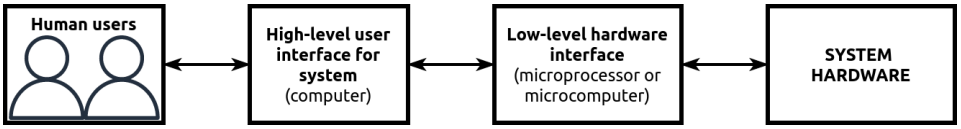


Figure 9.2: Implemented degrees of separation between system user and system hardware.

The old Helmholtz cage system did not have the third block, and instead used hardware directly connected to the interface computer. This begs the question: why was this more complicated scheme of Figure 9.2 decided upon? There are several arguments to be made for this:

1. First of all, personal computers are often rather inconvenient as direct interfaces with low-level hardware. There are no exposed pins such as voltage rails, serial communication lines, analog inputs or outputs, ADC or digital-to-analog converter (DAC) functionality, etc. All these options require additional peripheral interface cards anyway. To illustrate, the interface computer of the old system used three dedicated RS-232 interface cards to communicate with the power supplies. The demonstrator system connects several different types of hardware, such as sensors, power

supplies, and support electronics. Besides, it is expected to support unknown future upgrades as best it can. It is therefore obvious that one would use a different, more low-level interface controller for this situation, because it greatly simplifies interfacing with the system hardware and which then exposes the desired functionality to the user via a convenient and generic communication interface between the low-level and high-level interfaces.

2. To add to the previous point, the peripherals that would be needed to use the whole system with only one interface computer would not only be quite expensive (**DS-18**), but these peripherals are often quite specific, meaning they may not be off-the-shelf components (**DS-19**) and they may not be available forever (**SW-05**) or for every operating system (**EXTRA-10**).
3. Motivated by some requirements concerning user interaction (**DS-16**, **SW-10**), it is often disadvantageous for the user to have direct access to the hardware, especially when these users are not completely familiar with the system. To avoid putting the system at risk of damage or worse, the user at risk to injury due to naive usage, the level of control over the hardware is compartmentalized to a degree. Normal user operation for testing is done by interfacing with the high level interface, and more fundamental changes to the setup can be achieved by interacting with the low-level hardware interface. Such a compartmentalization can also be achieved in software, but given consideration **EXTRA-08**, it is implemented in hardware here.

With this established, the system can now be further developed. The high level user interface was chosen to be a standard personal computer. For the low level interface, a suitable hardware selection must be made.

9.3. Low-level hardware interface selection

To choose a suitable hardware platform for the low-level interface, we must first consider its functionality and which system requirements affect it. First, the low-level interface must be able to communicate to the high-level interface (a standard personal computer), as well as the system hardware, such as sensors, programmable power supplies, etcetera. Second, the low-level interface hardware must possess sufficient computational resources for tasks such as converting user commands to hardware instructions and data handling, although the specifics of this also depend on the software layout. Since the software layout is also yet to be defined, the software may also be designed to match the computational capabilities of the low-level interface hardware.

After a basic hardware survey, the Raspberry Pi series of microprocessor boards was chosen as the best candidate for the low-level hardware interface, for the following reasons:

- Excellent heritage both as educational platform as well as an industrial platform.
- Substantial on-board computational resources, sufficient to run its own graphic operating system (OS) if needed.
- Good selection of on-board hardware interfaces (I2C, SPI, various serial interfaces), as well as computer interfaces (Ethernet, serial, wifi, etc), which provides good confidence that it can perform all functionalities for the application at hand.
- Has an established market of third-party expansion boards that may be very convenient for hardware interfacing, potentially reducing the need for custom electronics.
- Excellent commercially-off-the-shelf (COTS) availability and affordability.
- Popular platform with wide user base providing high confidence that compatible hardware will be available for many years to come.

In brief, the Raspberry Pi provides a convenient bridge between the high-level interface any user may interact with all the way down to the hardware level. Several other microprocessor and microcontroller options were examined, but none of the other microcontroller options had as good an availability and development ecosystem as the Raspberry Pi, and most of the microcontrollers were held back by their limited computational performance.

9.3.1. Analog and digital conversion

One important feature that the Raspberry Pi does not have is analog interfacing; it has no on-board ADC or DAC. However, it is already known at this point that an ADC is required to interface with the Alphalab magnetometer, which is inherited from the previous Helmholtz cage setup, and only has analog data output. In addition, a few DAC channels would also allow for analog control of the power supplies, rather than using the particular software interface provided by the manufacturer. In case somebody else would want to build the system with different power supplies, they will have the best choice if they can pick any that has analog control, and therefore adding DAC capability would come with benefits for considerations **EXTRA-03** (since allowing for simple analog interfacing improves overall system modularity) and **EXTRA-10** (since analog control reduces dependence on manufacturer-specific software, which is not always agnostic to all operating systems).

After a hardware survey, two promising Raspberry Pi expansion boards were considered for providing ADC and DAC functionality. These boards were selected after a larger trade-off with criteria such as cost, number of channels, resolution, sampling rate, differential capability, etc. These were the MCC-128/MCC-152 board stack by Digilent and the CN-0554 development board by Analog Devices, also depicted in Figure 9.3. Both boards sit atop the standard 40-pin Raspberry Pi header.

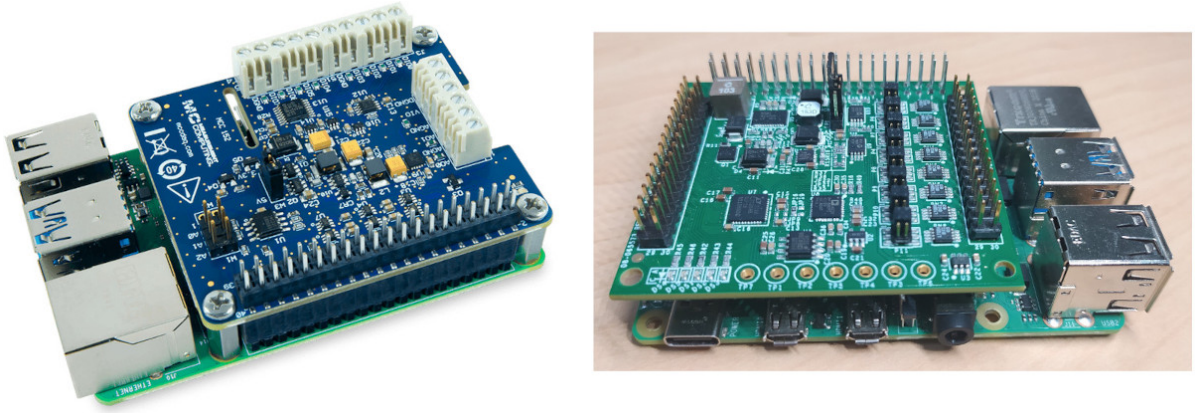


Figure 9.3: Left: MCC-152 DAC board by Digilent. Right: CN-0554 DAC/ADC board by Analog Devices. In both cases, the board is mounted atop a Raspberry Pi microprocessor unit.

The MCC-128 and MCC-152 by Digilent are part of a family of stackable boards for the Raspberry Pi, and supports up to four units being stacked on top of one Raspberry Pi. Both boards are in the order of 100-150 dollars. The MCC-128 contains a 16-bit ADC that can operate at rates up to 100 kS/s, and features four single-ended inputs that can also be configured as four differential channels. The MC-152 board has two 12-bit DAC channels suitable for 3.3V or 5V output. Interfacing with these boards is facilitated by open-source C/C++ and Python libraries on the Raspberry Pi.

The CN-0554 board by Analog Devices is a circuit evaluation board for the AD7124-8 ADC chip and the LTC2688 DAC chip, priced at approximately 100 dollars. The AD7124-8 has 16 independent 12-bit DAC channels with a variety of voltage ranges. The AD7124-8 contains eight differential channels, that can also be reconfigured to 15 pseudo-differential inputs. The channels are advertised as 24-bit, and can be operated at rates between approximately 1 kS/s and 10 kS/s. Interfacing is achieved using the Linux IIO framework, but this also requires a number of drivers, such as the Pyadi-iio drivers, all of which are available in a custom Linux distro provided by Analog Devices. This distro has to be run on the Raspberry Pi.

In the end, the CN-0554 was chosen due to being the cheaper option, as well as the more flexible option for this particular application. Because the DAC channels are intended for analog control of three power supplies, at least two MC-152 boards are needed to provide independent current control over all three supplies. For additional voltage control, another board would be needed. Then on top of that, a separate MCC-128 unit has to be bought to facilitate the ADC functionality. The total cost quickly approaches 400 euros, but all this functionality is also possible with a single CN-0554 board, making for

a cheaper and more compact solution. The CN-0554 is a development board, not exactly intended for use in designs, and so it was expected that working with it may feature some unwanted surprises.

In the end, the CN-0554 was purchased with the intention of getting it to work for the intended application, with the Digilent board as plan B if this did not work out. The CN-0554 turned out to be a workable solution, and the Digilent boards were never purchased.

9.3.2. Communication between low-level and high-level interfaces

Part of the reason that the Raspberry Pi was chosen for the low-level hardware interface is because it provides several option to communicate with the computer that serves as the high-level interface. It was not clear from the beginning which option would be the most convenient. In the end, a point-to-point transmission control protocol (TCP) connection between computer and Raspberry Pi was used, using a custom codec written for this exact purpose. This connection is explained in more detail in chapter 11 and Appendix G.

9.4. Companion board

To establish the system as depicted in Figure 9.1, the hardware discussed up to this point is not enough. Most notably, there is as of yet no way to reverse the current flow through the coils of the Helmholtz cage, as the power supplies that were already purchased for this purpose are unipolar supplies. As already discussed, the method for reversing the current flow from the old Helmholtz cage system could not be inherited due to its reliance on relays as switching devices and several other reasons. The CN-0554 ADC/DAC board does not feature screw terminals but header pins, and for a permanent system setup, it is preferable to connect these either via a converting wire harness or a routing board. In addition, it would be very nice to measure the current through the coils directly, so that an accurate assessment of the established magnetic field can be made. Finally, since the coils of the Helmholtz cage constitute an inductive load, rapid changes in current may result in large inductive voltage spikes, and the power supplies should be protected from these spikes.

In summary, there are several functionalities that are desirable for the system, but are not covered by the current selection of hardware. The peculiarity of these functionalities rules out many if not all COTS hardware to address these functionalities, unless a different hardware solution is acquired for each separate functionality. Instead, it was decided that an extra board would be needed to contain these functionalities and tie the system together. This board, which we will refer to as the **companion board**, was developed as part of the thesis.

An overview of the companion board as part of the broader system can be seen in Figure 9.4. The companion board, in orange, takes up a large block in the middle and contains several sub-blocks, and connects to the other hardware of the system in various ways.

First, on the left we see that the companion board interfaces with the output of the three power supplies. The power supplies are controlled via a channel on the DAC, which is routed to the power supply via the companion board. The output from these power supplies first goes through a diode, which is there for protection and to drop the voltage by a certain amount (for more information see subsection 10.3.1). Thereafter, the current from the power supply is measured, and this information is routed to a dedicated ADC channel. Then there is an H-bridge circuit which can present the power from the supply in forward direction, reverse direction, or disconnect it altogether. The state of the H-bridge is controlled from a channel on the DAC. Finally, this is connected to the coils of the Helmholtz cage, additionally protected by a suppression diode (see subsection 10.3.3).

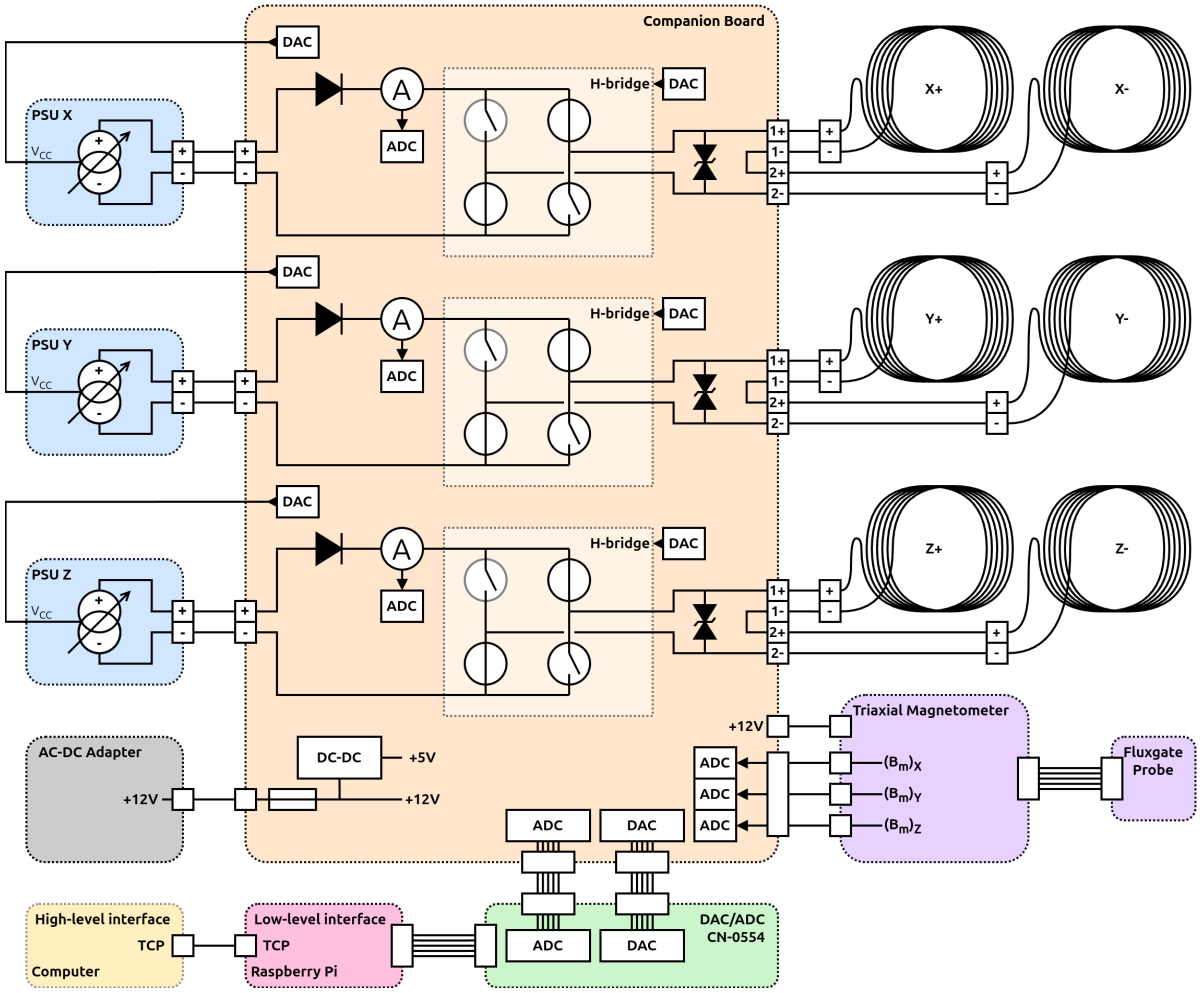


Figure 9.4: Block diagram of the hardware layout of the demonstrator.

Moving to the lower half of Figure 9.4, we see in yellow the high-level interface computer. This is the computer that the user interacts with. This computer is connected via TCP connection to the Raspberry Pi, which in turn serves as the low-level hardware interface. The CN-0554 ADC/DAC board sits atop the Raspberry Pi and is connected via the 40-pin Raspberry Pi header. Two ribbon cables connect this board to the companion board. The tri-axial magnetometer, which provides analog measurement data, is connected to the companion board also, and has its connections routed to three ADC channels on the CN-0554. Board power for the companion board is provided via a standard 12V DC adapter, and is on the companion board also converted to an auxiliary 5VDC rail.

The companion board is the main topic of chapter 10, where the whole design process of this board will be explained. Separately, chapter 11 will discuss the software that runs on the interface computer, on the Raspberry Pi low-level interface, and the software codec that facilitates communication between the two.

10

Electronic Design

The electronics design of the magnetic testing system turned into one of the most significant tasks of the thesis work. Many interesting problems were encountered during development, which made it both a valuable learning experience by necessity, as well as a frequent source of schedule delays. This chapter aims to take the reader along with this process, warts and all, in order to cast a light on the nature of these problems and how they were solved.

Some of the problems stemmed from the inexperience of the developer: electrical engineering is a field where the devil is often in the details; if one is not careful, neglecting important design details can very quickly turn into design-crippling issues, which may necessitate a full additional design iteration. However, even seasoned engineers can run into issues that could not have been foreseen and these have to be addressed when they rear their head. This is why this chapter is structured the way it is: to highlight not only those design aspects that *did* work, but also those that *did not*.

For example, it was discovered at a relatively late stage in development that the constant current supplies intended to be used for the system became unstable when driving the inductive load of the Helmholtz cage, where these supplies passed all other tests beforehand with a non-inductive load. Such instability could not have been foreseen based on the information provided by the manufacturer, nor could it be overcome using the same hardware. Another example is the behaviour of the bridge driver chips that were used in the design, which did not work as intended even if the application they were used for was very close to the intended behaviour by the manufacturer. Frustrating as such issues may be, these also provide valuable lessons about how to overcome such engineering problems, and they teach us how strategic choices in the design process can give the designer a much larger solution space.

At the end of this chapter, a complete description of the end product is given: a circuit board design containing many functions to make the magnetic testing system functional, but also many extra functionalities that make the design much more flexible, receptive to future expansion, and adaptive to modification by other parties. This board will henceforth be referred to as the Helmholtz cage **companion board**. The discussion of section 9.4 will be used as a starting point.

10.1. Polarity Switching

The electronic design process starts off with the relevant requirements, specifically at **ELEC-05-01**. Before the thesis started, the new power supplies that were supposed to drive the Helmholtz coils were already ordered. These were three units of model 1685B by BK Precision, capable of supplying loads with currents ranging from [0, 5] A with a voltage range of [0, 60] V. They were chosen to overcome the voltage-limitations posed by the older constant-current supplies, the ES-030-10 by Delta Elektronik, as discussed in subsection 6.2.2.

However, like the ES-030-10, the 1685B is still a unipolar power supply, only capable of producing positive voltages and currents. By applying a unipolar control current to a Helmholtz coil pair, the established field can point in one direction only; it cannot point in the other direction unless the direction

of current flow is also reversed. Therefore, to allow the Helmholtz cage to establish a magnetic field vector in any direction, a method of switching the polarity of the control current is required.

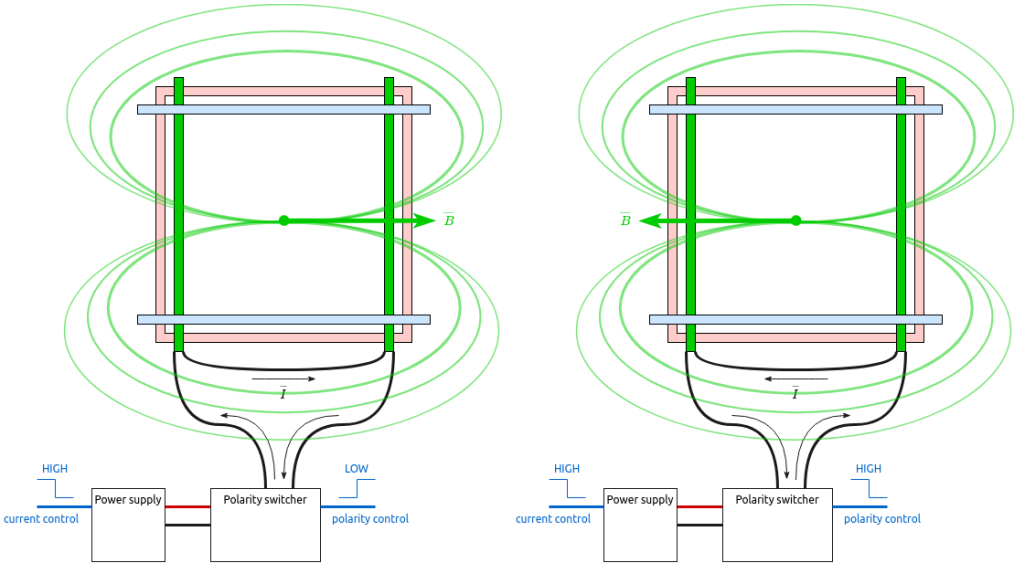


Figure 10.1: A polarity switcher reverses the direction of the current applied to the Helmholtz coil pairs by effectively making the control current bipolar.

How can this be achieved? The old Helmholtz cage design at faculty of Aerospace Engineering of TU Delft (AE) used a double-pole, double-throw (DPDT) electromechanical relay for every current channel (refer to Figure 6.8). In this network, the relays essentially act like a pair of hands that very quickly disconnects the cables from the power supplies and swaps them around. For this purpose, these relays are fairly useful, as they are break-before-make switches by their very nature, making it nearly impossible for them to cause a short-circuit, as long as they are operated within design specifications.

However, using electromechanical relays in this case has a very important downside: **speed**, or rather lack thereof. Relays use moving mechanical parts to make and break its connections. These parts have mass and therefore inertia, meaning the part has some travel time before it makes a connection. This travel time is not insignificant, especially not for power relays, whose physical dimensions tend to be larger. A few solid state relays exist within in the required power specifications of 60 VDC, 5 A, and these operate with a phototransistor, requiring no moving parts. Despite this, these components cannot switch much quicker than their electromechanical counterparts, and their main benefit lies mainly in their longevity rather than their switching speed.

When the common contactor of the relay is traveling from one terminal to the other, no current flows, resulting in a "dead time". During the dead time, the output current is zero. For the application of a magnetic simulation system, this would result in a period of zero lasting several dozen milliseconds. A smooth magnetic envelope moving through the zero line would have a horizontal kink in it.

Requirement **PER-14-01** was made to stipulate that this is not acceptable for the magnetic testing system. As such, it takes into account an established stakeholder requirement, which is to have faster switching than electromechanical relays could manage. As a result, a different approach is needed to switch the polarity of the coil current.

10.1.1. Polarity switching with transistors

Transistors can be coaxed into behaving like electronic switches [1]. The word "coaxed" is used here for good reason; their behaviour is much more nuanced than a simple binary switch, containing several half-states between "on" and "off", and they typically require quite specific control conditions to function

[1] A vast amount of 21st century electronics are based on this principle, such as switching mode power supplies, class D amplifiers, etc.

correctly. However, they are orders of magnitude faster when used as switching devices: A typical power MOSFET can switch states well within 10^{-7} s, whereas the fastest relays are in the order of 10^{-2} s.

In addition, when used as a switch, transistors behave like single-pole, single-throw (SPST) switches, whereas relays are typically single-pole, double-throw (SPDT) or DPDT switches. To make a polarity switching device using transistors, a circuit known as an H-bridge is used. This circuit uses four SPST switches to allow polarity switching on a single line.

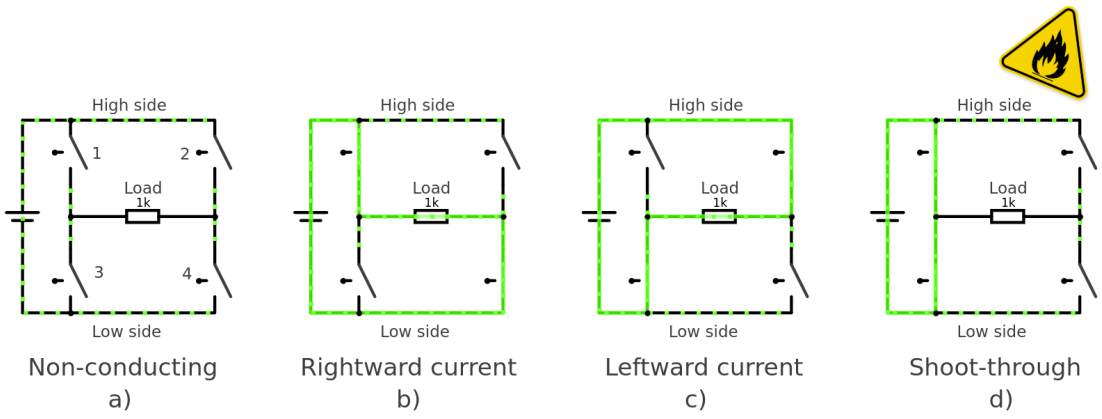


Figure 10.2: An H-bridge circuit with SPST switches in several states

Depending on the states of the switches, an H-bridge has several states, as can be seen in Figure 10.2. If any of switch pairs 1-2 or 3-4 are open, the circuit is **non-conductive** (also known as "free-wheeling", in the context of motor drivers). By closing switch pairs 1-4 **or**[2] 2-3, current can flow through the load in either direction, the direction depending on the combination of switches.

If either switch pairs 1-3 or 2-4 are closed, a condition known as **shoot-through** occurs. In this situation, the switches bypass the load and present a short-circuit to the power supply. Like any short-circuit, this is a hazard and it is therefore a requirement of the polarity switching circuit that shoot-through is to be prevented at all times.

For the application at hand, the SPST switches in the H-bridge circuit of Figure 10.2 can be replaced with transistors, and control can be applied to the base/gate of each transistor. Now, what kind of transistors should be used for this application? There are several common types, such as bipolar junction transistors (BJTs), junction field effect transistors (JFETs), MOSFETs, etc. After trade-off, MOSFETs were chosen, for the following reasons:

1. MOSFETs of the required power rating are cheaply available off-the-shelf.
2. MOSFETs of the required power rating are commonly available at many vendors, making them easy to source.
3. Power MOSFETs can have ultra low source-drain resistance, allowing some to handle hundreds of amps, which greatly simplifies the thermal aspects of the design.
4. MOSFETs can be controlled with a voltage (rather than a current), which in certain situations is easier to manage.

Additionally, insulated gate bipolar transistors (IGBTs) were also considered, as their insulated gate design makes driving them quite similar to MOSFETs. However, for this particular application they are not a good choice. Since these devices retain the PN junction of a bipolar transistor, they will retain the fundamental voltage drop between the collector and the emitter. This makes IGBTs useful in high voltage applications, where this voltage drop is comparatively insignificant. However, in low voltage, high current applications, they will be rather inefficient, given that the voltage drop along the current path will be substantial in comparison to the operating voltage, and they will dissipate significant

[2] "**or**" here means "**exclusive OR**": 1-4 XOR 2-3

amount of the power in direct proportion to the current. Therefore, for this design, MOSFETs were chosen instead. Still, it is expected that the MOSFETs in the design could be exchanged with IGBTs of similar specifications with no/little modification to the circuit, although they will almost certainly need some form of additional cooling.

This leads to the simple design shown in Figure 10.3. We will see in the next section why this is a flawed design, and adjust the circuit accordingly.

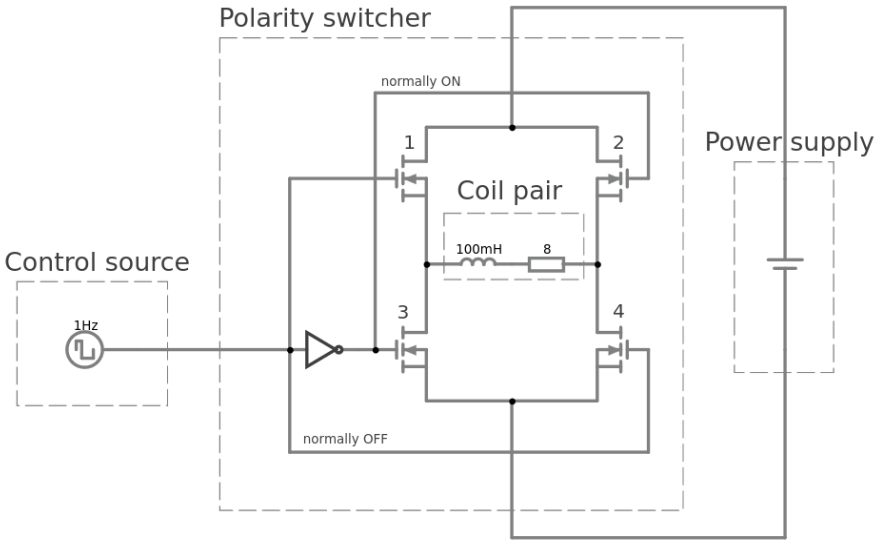


Figure 10.3: A naive first design for the H-bridge circuit. The four MOSFETs are controlled by a common control line. The control signal to MOSFET pair 2-3 first passes through an inverting stage, so that MOSFET pairs 1-4 and 2-3 are always doing the opposite of one another.

10.1.2. Preventing shoot-through by design

The circuit in Figure 10.3 is rather naive in its simplicity. The most notable problem is that there is no shoot-through prevention built in. MOSFETs typically have different switch-on and switch-off times, and so it cannot be guaranteed that shoot-through is avoided at the moment the polarity is switched. To address this, the circuit must be altered such that when switching polarity, the open[3] MOSFET pair always closes before the other MOSFET pair engages, preferably with a fixed, deterministic duration. This period between states we will now call the **dead time**. The control signal should therefore have an envelope somewhat resembling that of Figure 10.4.

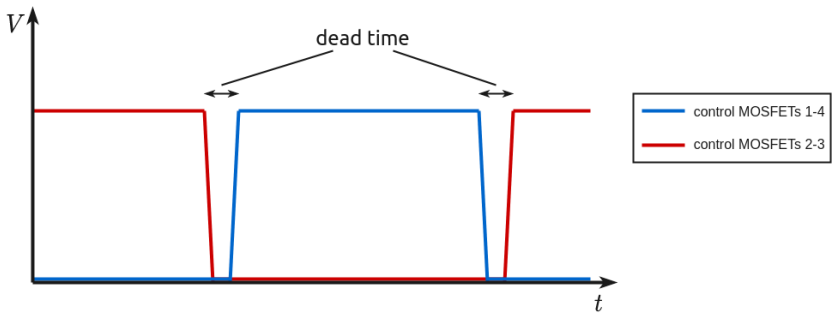


Figure 10.4: Modifying the control signal to the MOSFET pairs such that there is a dead time in between each switched state.

[3]Note: with MOSFETs, there is formally no such thing as "open" or "closed"; they are fundamentally messy, analog, non-discrete devices. However, since the MOSFETs under discussion are used as SPST switches, the terminology is borrowed for the sake of clarity.

One way to achieve this is by delaying the next rising edge of the control signal to the corresponding MOSFET pair, as depicted in Figure 10.5. Note that the delay must not apply to the falling edge of the other MOSFET pair too, because otherwise both control signals just shift in time, and the problem remains.

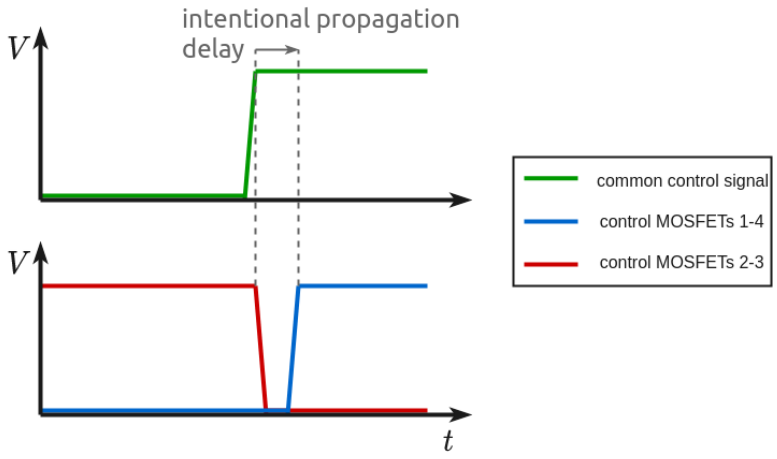


Figure 10.5: Introducing a propagation delay in the rising edge response to ensure a dead time.

The simplest way to enact a signal delay with common electronic components is to use an **RC delay** (Figure 10.6a). In essence, it is the simplest low-pass filter one can build, and it causes the sharp rising edge of the control signal to be smoothed out as higher frequency components of the rising edge are attenuated. By placing an operational amplifier (opamp) configured as a comparator downstream of this filter (Figure 10.6b), the sharp rising edge of the control signal can be retained, but the propagation delay remains in place. The voltage at which the comparator triggers the delayed rising edge is equal to V_{ref}, connected to the inverting input of the comparator.

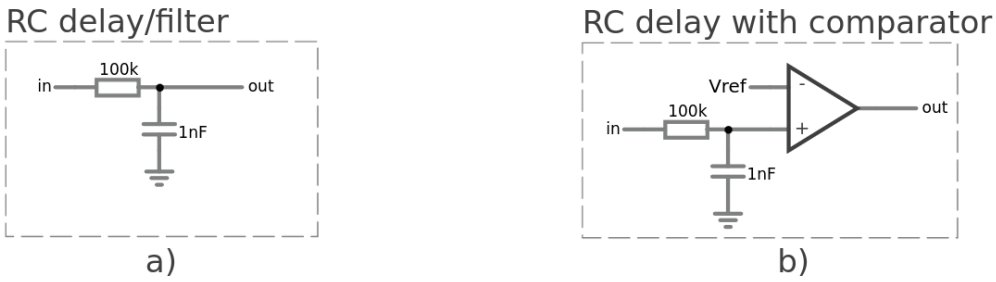


Figure 10.6: (a) Simple RC filter circuit (b) RC filter with downstream comparator.

This is however not enough, because as mentioned, only the rising edge should be delayed, not the falling edge. In order to address this, the RC delay is enhanced such that the discharging of the capacitor is done through a lower value resistor than during charging. This can be done with a diode, and essentially creates an RC filter with two RC constants, depending on the direction of the current flow. This looks something like Figure 10.7. First of all, the input signal is buffered by the leftmost op-amp to prevent loading effects on the control signal. When the control signal has a rising edge, the buffer will copy it and start charging the capacitor through the 100k resistor. Some time later, when the voltage on the capacitor reaches V_{ref}, the comparator will trigger and output a high signal, controlling the MOSFETs.

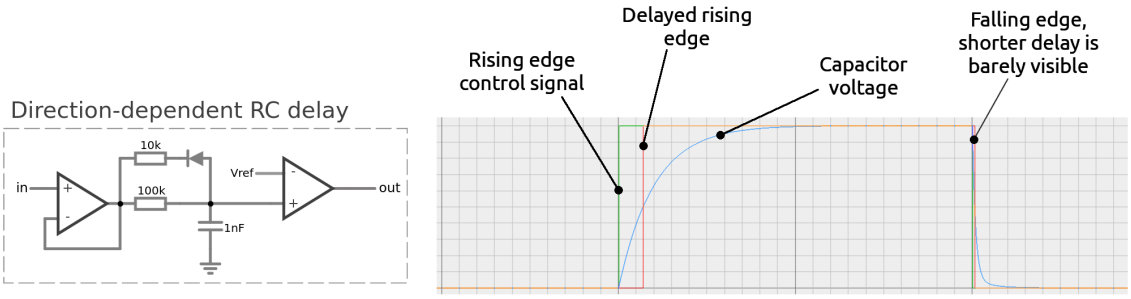


Figure 10.7: Enhanced RC delay with buffered input

When the control signal has a low edge, the capacitor will start discharging into the output of the op-amp through the 100k resistor. But due to the biasing of the diode, a second discharge path is now available through the diode and 10k resistor. The capacitor can discharge through these paths 11 times faster[4] than it could charge through the 100k resistor alone. As the charging rate of the capacitor determines the delay of the control signal, the effective delay of the rising edge on the control signal is now much larger than that of the falling edge.

This can also be seen in the graph on the right hand side of Figure 10.7. The rising edges are much farther apart in time than falling edges. The difference can be further increased by increasing V_{ref}.

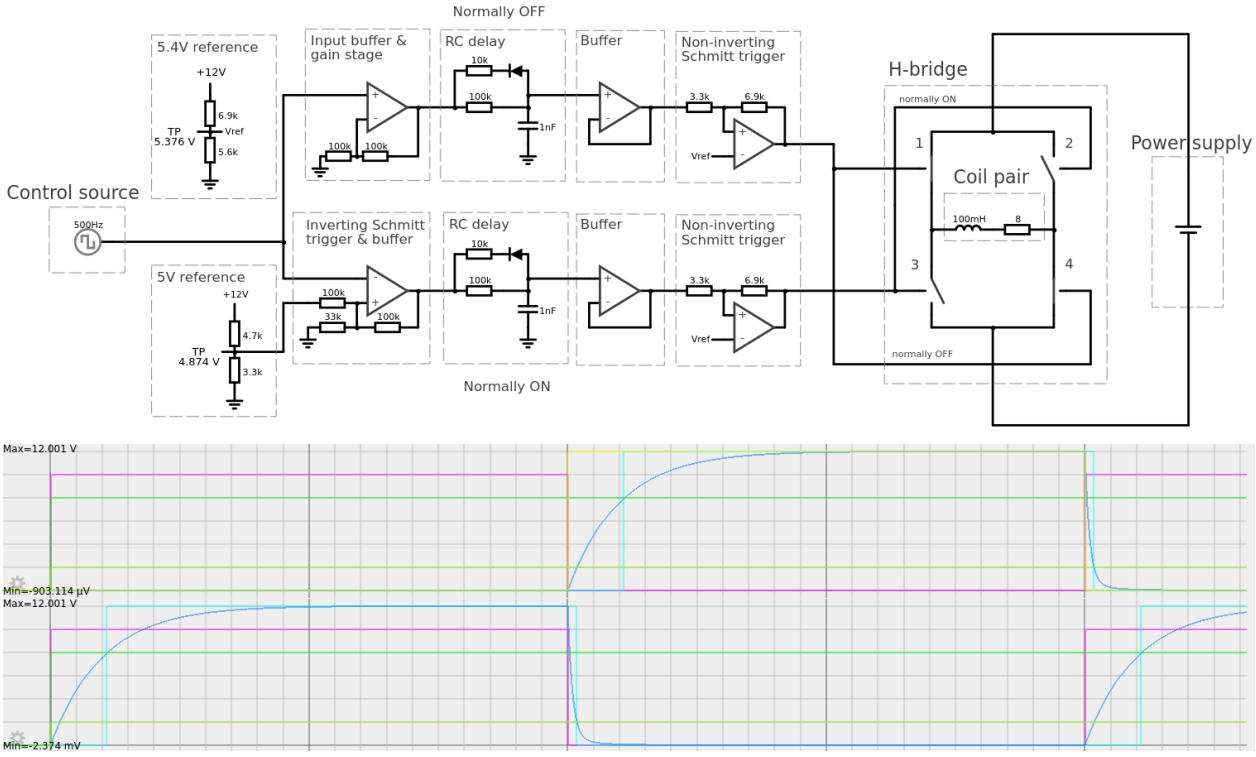


Figure 10.8: Two parallel delay channels controlling an H-bridge circuit. The corresponding Falstad simulation is available at https://tinyurl.com/2lom5x6a

In Figure 10.8, the circuit looks more complex, but it is essentially an extension of the same idea. There are now two channels, one in charge of MOSFET pair 2-3, the other in charge of 1-4. Since the MOSFET pairs must do the exact opposite of their counterpart, one of the channels (the bottom one, "Normally ON") is logic inverted. The control signal still comes from a single source, and for reasons that will

[4] If we neglect the forward voltage drop of the diode for a moment

become clear later, this control signal is 10V when high. The rest of the circuit is powered from 12V. The op-amps are powered with 12V on the upper rail, and connected to ground on the lower rail.

On the upper channel, the control signal is first buffered, and boosted with a non-inverting gain stage. The gain of this stage is 2, which should result in 20V, but this is clipped to 12V, since the op-amp cannot output more than its supply rail. The reason for boosting to 12V at all is to maintain consistency with the lower channel. Then the signal goes through the already familiar RC delay stage. Downstream from here used to be a comparator, which has been replaced by a non-inverting Schmitt trigger instead. The reason for this is both to make the system more robust to noise by adding hysteresis, and because general-purpose op-amps tend to be a bit happier in a feedback configuration (whereas the comparator is a feed-forward). It is however necessary to add a buffer stage between the RC delay stage and the Schmitt trigger. If this is not done, the RC delay and the Schmitt trigger will together form a voltage divider, since there is a virtual ground node at the non-inverting input of the Schmitt trigger, and the result is that the circuit will not work. So, a buffer stage is added in between.

Now turning to the lower channel, here the control signal is boosted to 12V, inverted, and buffered all by one inverting Schmitt trigger stage. This Schmitt trigger is not essential, especially if the control signal is clean, so an inverting op-amp or inverting resistor-transistor logic (RTL) stage could work here too. Regardless, the inverted signal will enter the RC stage and behave in the exact same way as the upper channel, except it does so when the control signal is low instead of high. Below the circuit in Figure 10.8 there is a double graph with a simulation of one full cycle of the two channels, indicating that the stages work as intended.

10.1.3. MOSFET complications

There is still a proverbial elephant in the room related to Figure 10.8 that has not yet been discussed. The H-bridge in Figure 10.8 does not use MOSFETs at all, but simple SPST switches. The reason for this is that while the control logic of the circuit is sound, the MOSFETs cannot actually be properly activated by it. To explain why, let us first take a more detailed look at MOSFETs.

Power MOSFETs have been around for around 50 years, and have nowadays become key component in many demanding switching applications across the electronics industry [52]. They are semiconductor devices that formally feature three terminals: a **drain**, a **source**, and a **gate**. In simple terms, a very high resistance exists between drain and source, such that the MOSFET does not allow current to flow between these terminals. But by applying enough charge to the gate, a conductive channel can be established between drain and gate across the epitaxial and substrate layers inside the MOSFET. For power MOSFETs, this conductive channel can be very low resistance and therefore high currents can flow without depositing much heat inside the device itself. The substrate layer inside the MOSFET comes in two flavours: P-doped and N-doped, which is the reason why one distinguishes between P-channel MOSFETs and N-channel MOSFETs. Figure 10.9 depicts the inside of an N-channel MOSFET.

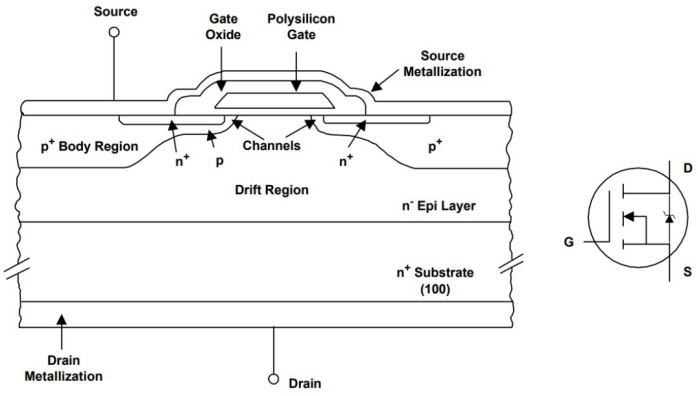


Figure 10.9: Schematic diagram of an N-channel power MOSFET [53].

If one applies sufficient voltage to the gate, the resulting electric field will adjust the distribution of charge carriers in the underlying epitaxial layer, which will form a concentration of charge carriers in the

region underneath the gate. This channel can carry current between source and drain. To use a MOSFET as an SPST switch, one would have to apply a controlled amount of charge to the gate of the MOSFET to *"open"* it, and remove the charge on the gate to *"close"* it. In practical terms, it involves applying a certain threshold voltage to the gate. This voltage is referred to the source terminal of the MOSFET, and so the gate is charged sufficiently when the voltage between gate and source (V_{GS}) exceeds the threshold voltage, typically in the order of a few volts. Good circuit design practice dictates the use of a higher gate voltage than the bare minimum threshold voltage [53].

However, a problem arises due to the layout of the H-bridge circuit. Figure 10.10a depicts a simplified overview of the H-bridge circuit with MOSFETs, at the moment no MOSFETs have any control signal applied to their gates. The high side MOSFETs have 60 V applied at their drains, and the low side MOSFETs have their sources grounded. The network between the MOSFET contains an inductive load representing a Helmholtz coil pair. Since none of the MOSFETs are conducting, the voltage in this network is technically floating, but in reality will likely tend to ground due to dissipation. Focusing on MOSFET 1, it can be seen that its gate-to-source voltage $V_{GS,1}$ equals zero volts, and so indeed the MOSFET is "closed".

Now if one wishes to run current through the load in a rightwards direction, MOSFETs 1 and 4 should become conductive. If the aforementioned 10 V control signal is applied to the gates of these MOSFETs, it stands to reason that $V_{GS,1}$ and $V_{GS,4}$ should become +10 V, which is plenty to "open" the MOSFETs, and current can flow from the supply through the load to ground, as depicted in Figure 10.10b. However, since the source of MOSFET 1 is ahead of the load, the instant after this current flow is established the voltage at this node will become near 60 V. This causes $V_{GS,1}$ to become −50 V, immediately closing the MOSFET and stopping the flow of current (see Figure 10.10c). This not only invalidates the circuit for its purpose, but also exposes MOSFET 1 to a dangerously high V_{GS}, as most power MOSFETs have a $|V_{GS}|_{max}$ of around 20 or 30 volts.

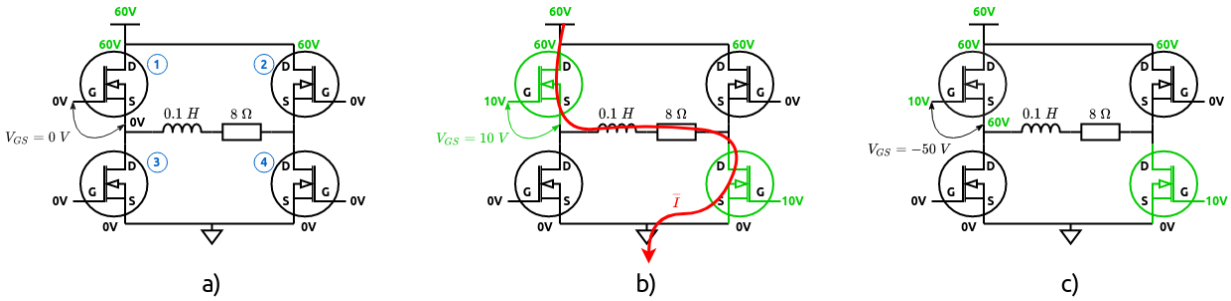


Figure 10.10: An issue arises when trying to control the high side MOSFETs of an H-bridge, as a suitable V_{GS} is not possible to manage with a static 10 V control signal.

In conclusion, to make the circuit in Figure 10.8 work with MOSFETs, it needs to be extended with a circuit that will charge the gates of the correct MOSFET pair exactly a few volts above the instantaneous source voltage. This is not trivial, because the source voltage of the MOSFETs on the high side of the H-bridge is not constant: The power supply that is powering the H-bridge is controlled in current, which means that it picks a voltage appropriate for the load. We are designing this system for 60V power supplies, which is far higher than the 10V of the control signal. A simple voltage multiplier or charge pump will not suffice here, because as mentioned, it has to react to the voltage at the MOSFET source and go a few volts above it when it is time to open the MOSFET. Development of such a network would likely complicate the circuit significantly and increase the number of test that have to be performed on it. Additionally, since the author is not an electrical engineer, there would likely be mistakes, increasing the development time further.

10.1.4. Integrated bridge drivers

When this conclusion became apparent during development, it was investigated whether commercially-off-the-shelf (COTS) solutions to this problem already exist. This was also driven by another desire: namely to ensure that the ground of the H-bridge be fully isolated from the signal ground of the

control circuitry. And indeed, there are some high performance half-bridge controllers and full-bridge controllers on the market. Not all of these are useful for the design at hand however, as many are designed for rapid switching, and cannot hold the MOSFETs open at 100% duty cycle. Others can only work with one common ground, and others yet are expensive or difficult parts to source. Moreover, the power requirements of 60 V, 5 A turn out to be slightly outside the range of most of the bridge drivers on the market. After rejecting unsuitable candidates, two main candidates for the design remained:

The DRV8262 bridge driver
This device is part of a series of bridge drivers consisting of the DRV8262, DRV8412 and the DRV8962. All are made by Texas Instruments and can handle 70V bridge voltages with at least 6A of current. The DRV8262 stands out as it contains two full-bridge drivers and includes integrated current sensing. It comes in a HTSSOP-44 package in two variants: one with the heat dissipation pad on the top, and the other on the bottom.

Advantages

- Two full-bridge drivers per package, so two packages needed at most for the design.
- Thermal management appears relatively straightforward, though it may complicate the soldering process.
- Integrated current sensing per channel, reducing the complexity of the electronics elsewhere in the design.

Disadvantages

- Richly featured and complex device, therefore not straightforward to incorporate in the design.
- Contains features such as current limiting that may interfere with operation, therefore requires extra testing to rule these cases out.
- Difficult to solder, due to it being an surface mounted device (SMD) component with many pins and a thermal pad.

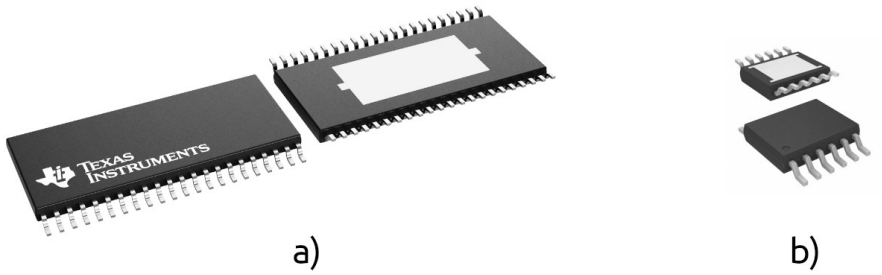


Figure 10.11: (a) DRV8262 dual bridge driver in a HTSSOP-44 package. (b) LTC7060 half-bridge driver in an MSOP-12 package.

LTC7060 half-bridge driver
This device is a relatively simple half-bridge driver designed for switching applications, but according to the datasheet is also capable of 100% duty cycle control. It is made by Analog Devices. Two LTC7060 units can be paired in a mirror configuration which turns them into a full-bridge driver. Despite being a relatively simple device, it contains a charge pump controller and fully insulates the bridge ground from the signal ground.

Advantages

- Does not contain many extra features beyond the necessary, making it more straightforward to incorporate into the design.
- Inexpensive.

Disadvantages

- Half-bridge driver, not a full-bridge driver, although it can be used in pairs as a full-bridge driver.

- No integrated current sensing.
- Difficult to solder, due to it being an SMD component with small and a thermal pad.

After trading these two devices off against each other, the LTC7060 was chosen. The deciding factors were as follows: the DRV8262 contained some safety features that may interfere with its intended function in the design, and this could not be ruled out from the datasheet, which therefore posed a risk. In addition, the thesis supervisor advised the less complex device.

10.1.5. H-bridge design with the LTC7060

Incorporating the half-bridge drivers into the existing design causes most of the control circuitry in Figure 10.8 to be replaced by a pair of LTC7060 devices and the corresponding support circuitry. Figure 10.12 shows how the manufacturer recommends implementing the device for controlling a non-inverting half-bridge and an inverting half-bridge. The only difference with the full bridge setup is that both halves are connected together at the top, in the middle, and at the bottom. Note that this leaves the ground reference of the H-bridge circuit fully isolated from the control side, meaning that the H-bridge can float without interfering with the control circuitry.

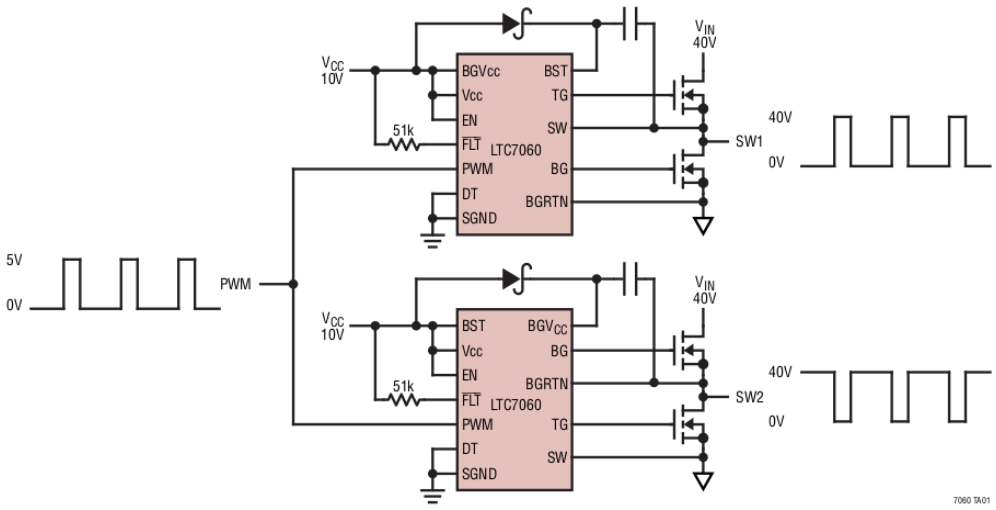


Figure 10.12: Typical application of the LTC7060 half-bridge driver recommended by the manufacturer [54]. The same device can be configured to be inverting or non-inverting the pulse width modulation (PWM) input signal. Also note the two different ground symbols, indicating that the ground of the H-bridge and the control circuitry are isolated.

Activation of the high side MOSFETs is achieved by charging the boost capacitor from the V_{cc} supply through the isolation Schottky diode, as depicted in Figure 10.12. The LTC7060 never charges this capacitor; it can only discharge it [54]. When it is time to switch on the MOSFET, the LTC7060 places the boost capacitor in series between the source and gate of the MOSFET. Because the voltage across the capacitor is now referred to the voltage at the source of the MOSFET, V_{GS} now equals the voltage across the capacitor, allowing the MOSFET to switch on. This will slowly begin to drain the capacitor, which the LTC7060 will track. When the capacitor reaches a certain threshold voltage (around 3.1 V), it is quickly switched out of the circuit to be topped up through the Schottky diode. SPICE simulation indicates that for the design at hand, this should be in the order of tens to hundreds of nanoseconds. This is an acceptably short period in this case, and this quick exchange may not even be noticed by the inductive load.

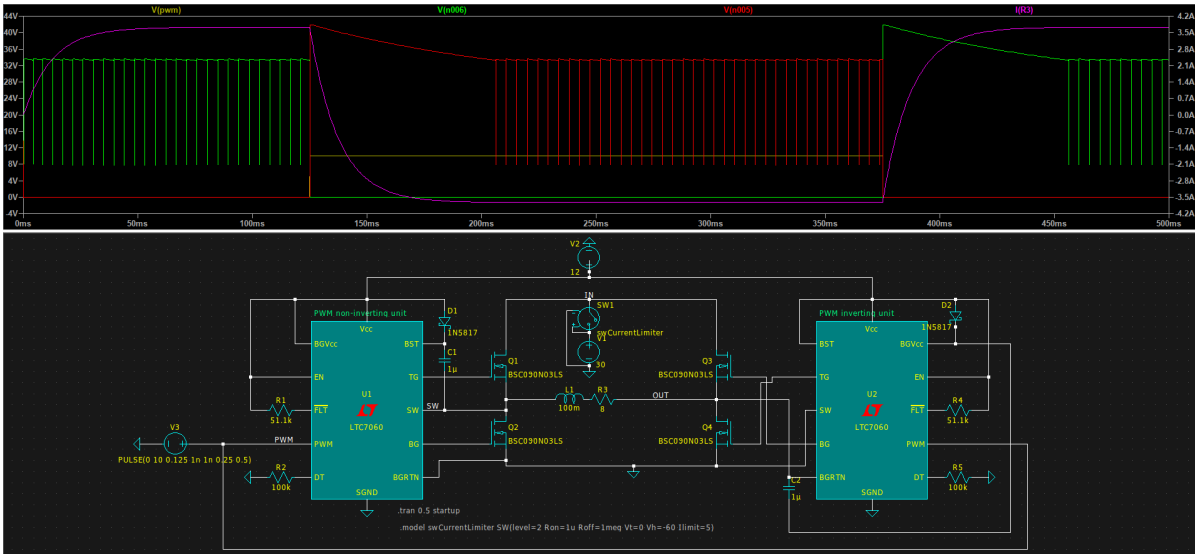


Figure 10.13: SPICE simulation of an N-channel MOSFET H-bridge circuit controlled by dual LTC7060 bridge drivers, similar as in Figure 10.12, but as a full H-bridge. The load corresponds approximately to one Helmholtz coil pair. The switching signal (PWM) is shown in yellow, the gate voltages of the inverting and non-inverting high-side MOSFETs are green and red respectively, and the current through the load is magenta. Note the brief, periodic "off" moments used to recharge the boost capacitor, happening approximately 250 times per second.

Figure 10.14 shows the H-bridge configuration that was used to switch the polarity of the power supply output. This was implemented in a printed circuit board (PCB) during the first design iteration.

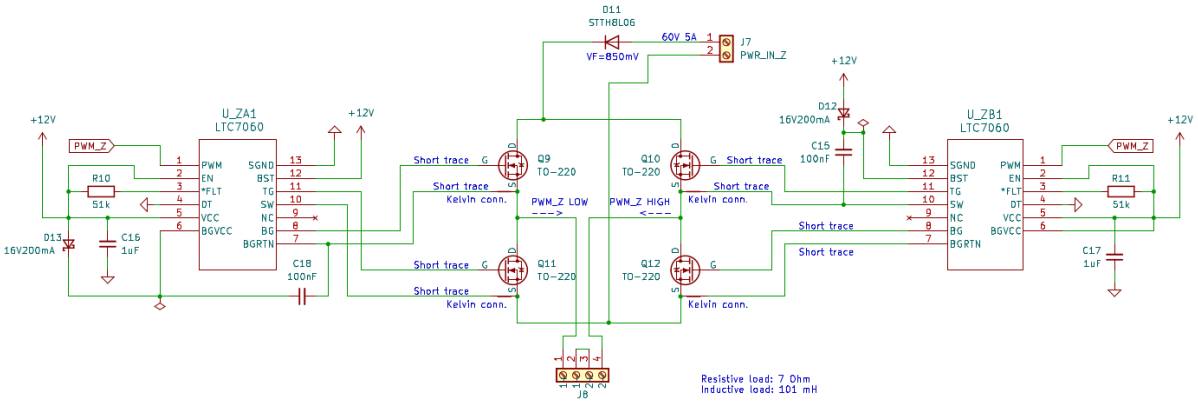


Figure 10.14: Implementation of one H-bridge channel, this one for the Z-axis coil pair. The left LTC7060 is configured to be inverting, meaning that with a low input signal, current flow through the coils is from the left half-bridge to the right half-bridge.

Given the small amount of time required to "top up" the boost capacitor, as obtained from the SPICE simulation, one would therefore expect the corresponding MOSFET pair to be conducting, every now and then interrupted by a small "off" period lasting less than a microsecond. It was even expected that this may not turn off the MOSFET, as the internals of the MOSFET gate and surrounding circuit may contain enough internal capacitance to keep the MOSFET on during the short "off" period. With a few hundred expected "off" periods per second, the expected uptime of the MOSFETs was in the order of 99.9%.

However, a nasty surprise awaited after the first version of the companion board was manufactured, assembled, and tested. The circuit of Figure 10.14 did not behave as expected at all:

1. The "off" periods were much more frequent than expected, in the order of tens of kHz.
2. The voltage across the capacitor at which this switching occurred did not correspond very precisely with what was expected.

3. The "off" periods were **much** longer than expected from the simulation.

The resulting behaviour amounted to the following: the MOSFETs that should have been on would be switching on and off at tens of kHz, with an effective duty cycle of anything between 50% and 70%. Given that the downstream load is a large inductor, with presumably some internal capacitance, this would in essence have the effect of supplying the load with only a fraction of the supplied voltage, similar to how pulse width modulation (PWM) voltage control is often implemented. As a result, only a fraction of the current would be delivered into the load, causing an insufficient and unpredictable magnetic field output. In conclusion, this behaviour made the circuit unusable for its intended purpose.

10.1.6. H-bridge design with the LTC7060 - Revisited

When this problem with the H-bridge was discovered, several solutions were explored. The capacitance of the boost capacitor was increased first by one, then by two orders of magnitude, to no avail. The circuit from Figure 10.8 was revisited to investigate whether it could be made to function acceptably, with an experimental charge pump design. Other bridge driver options were explored also, as well as different H-bridge topologies, incorporating P-channel MOSFETs.

However, most time was spent tinkering with the existing circuit board. Cutting some traces, soldering bodging wires onto others, to investigate the root causes of the issue and explain the observed behaviour. After all, the way in which the LTC7060 chips were used was very close to the application envisaged by the manufacturer. What complicated this process was the notion that the LTC7060 chips seemed quite fragile and would fail easily when used outside their specification. Moreover, when they did fail, they would fail in a variety of suprising and counter-intuitive ways, including as a 100 ohm short-circuit of the main power supply, a short circuit of one or more of the pins interacting with the MOSFETs, etc. This made it difficult to distinguish between an actual design issue with the circuit and a broken LTC7060 chip, and usually the only way to make sure was to try something after soldering on a fresh LTC7060 chip (the old chip would typically not survive the desoldering process anyway).

The solution that eventually prevailed was to replace the boost capacitor with a voltage potential of greater capacity and lesser caprice. This approach can be seen in Figure 10.15. The boost capacitor, as well as its accompanying charging Schottky diode are replaced with a static voltage potential. Rather than switching in a charged capacitor between source and gate, the LTC7060 would now switch a 9V battery in series with the source and gate, which has the same effect. So long as the voltage is not near the maximum allowed V_{GS} of the MOSFET (20V in this case), both the LTC7060 and the MOSFET will be none the wiser, and will behave the same as before. Now the V_{GS} is higher than it was before, which is advantageous to reduce the source-drain resistance of the MOSFET by "turning it on more", and also there is no constant "topping up" of the voltage source required.

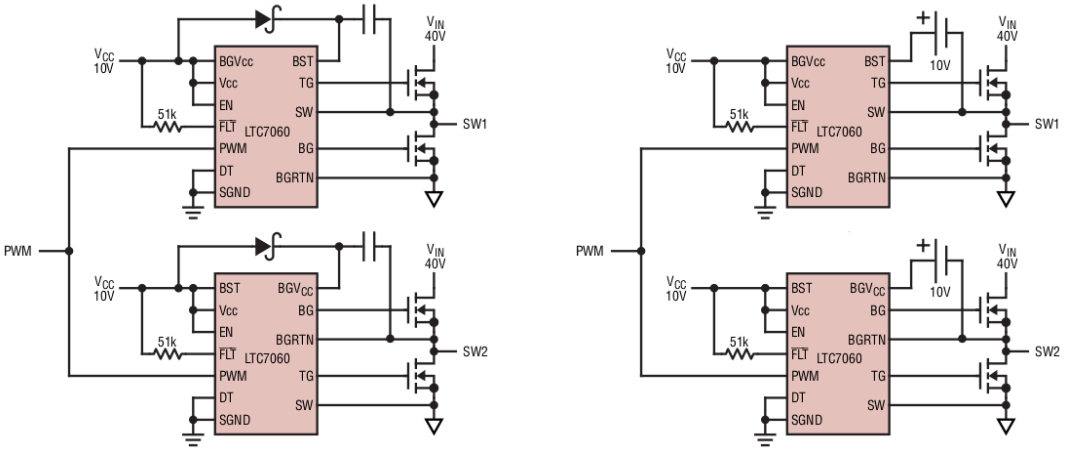


Figure 10.15: On the left, typical application of the LTC7060 half-bridge driver recommended by the manufacturer [54]. On the right, a modified implementation where the boost capacitor and charging diode are replaced with a static voltage source.

When testing this solution, two 9V batteries were used for this purpose, one for each LTC7060 chip (see

Figure 10.16). The H-bridge could now be reliably used to switch the polarity of the load.

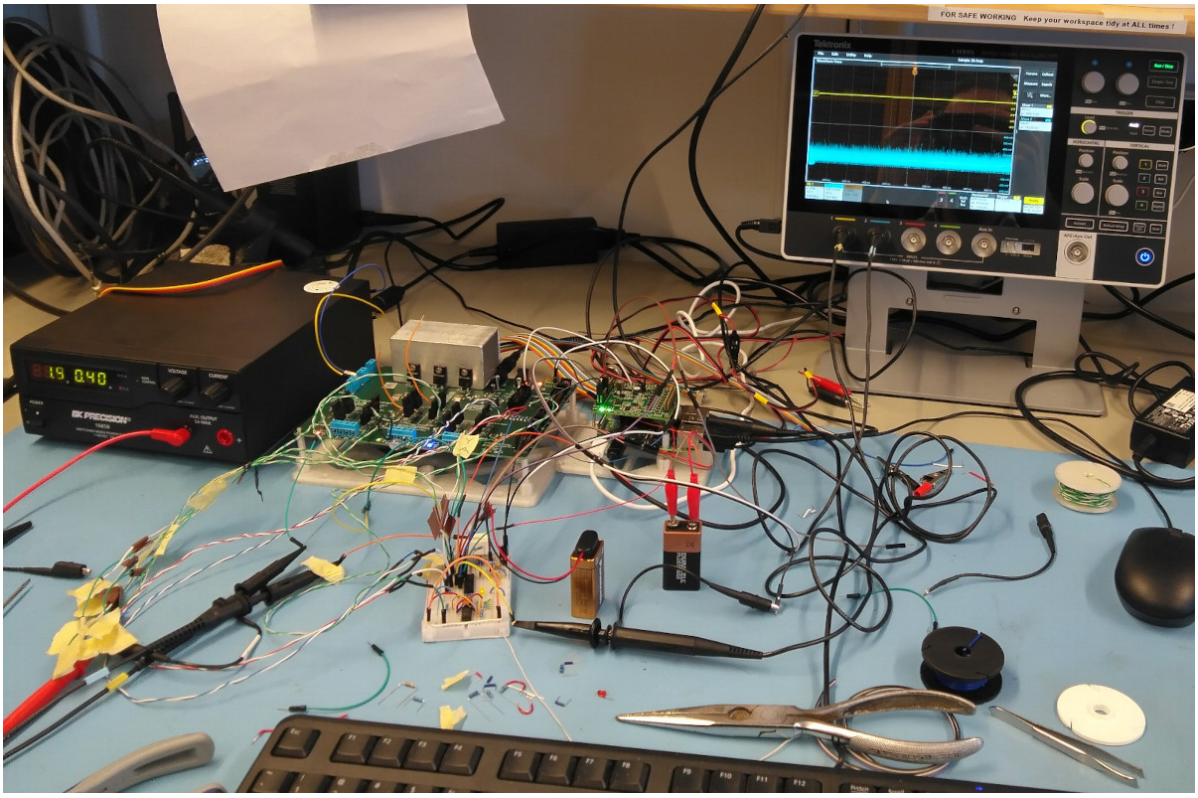


Figure 10.16: Testing whether two 9V batteries can be used in conjunction with the LTC7060 to turn on the H-bridge MOSFETs. A number of other changes are being prototyped here also, such as Schmitt-trigger input line buffers. Note the spaghetti of bodging wires, which will cause intermittent self-oscillatory behaviour issues, described later on in this chapter.

However, for a permanent testing setup, using batteries is a terrible solution. For three H-bridges, six 9V batteries would be needed, and they would have to be recharged or replaced regularly. A better solution would be to provide an everlasting voltage potential in a different way. This can be done by creating a galvanically insulated voltage supply using an isolation transformer. The transformer isolation is required because if it were directly connected to the supply rail of the board, it would break the ground isolation facilitated by the LTC7060.

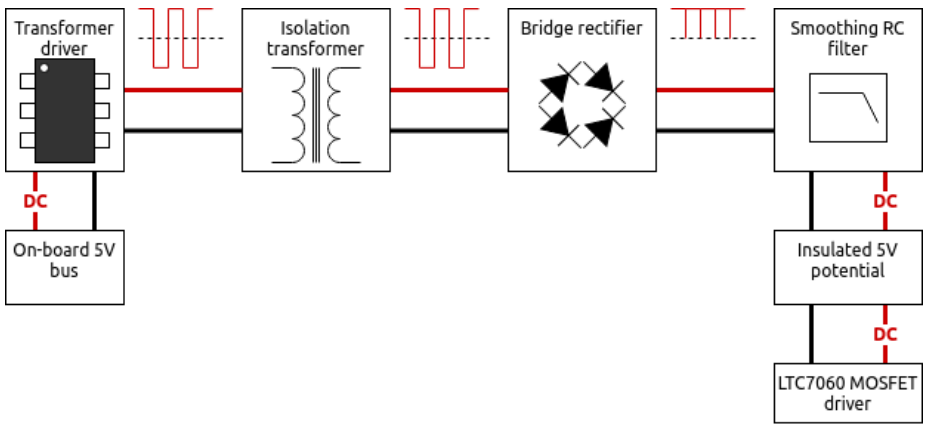


Figure 10.17: Creating an everlasting yet galvanically insulated potential to replace the 9V batteries.

The working principle of this is shown in Figure 10.17. A transformer driver chip takes a common DC

supply voltage, for example a 5V rail, and drives a transformer with it. It does this by rapidly and alternatingly applying the DC voltage to either end of the primary winding of the transformer, creating an alternating current through the primary that induces an alternating current at the secondary side. To turn this alternating current at the secondary side into a direct current, it is rectified by a diode bridge and smoothed by a sufficiently large capacitor. This voltage potential is then used to drive the MOSFETs instead of the 9V battery. Depending on the winding ratio of the transformer and the type of bridge rectifier used, the voltage can be configured as desired. In addition, an H-bridge safety feature could potentially be built into the circuit by disabling the transformer driver, which will ensure that the MOSFETs never have access to a high enough V_{GS} to turn on.

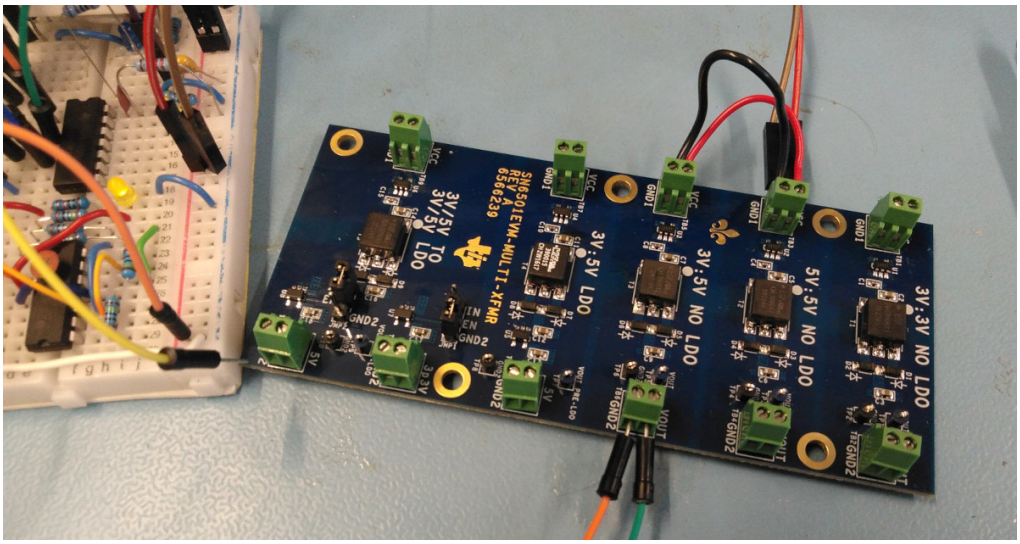


Figure 10.18: Prototyping the transformer-based implementation using a modified SN6501EVM-MULTI-XFMR evaluation board.

This was tested before implementation using an SN6501 evaluation board, which features a number of SN6501 driver chips, a variety of transformers, and rectifier circuits. One transformer on the board was replaced so that two identical channels were created. The aforementioned pair of 9V batteries was then replaced by these voltage sources, and these performed adequately.

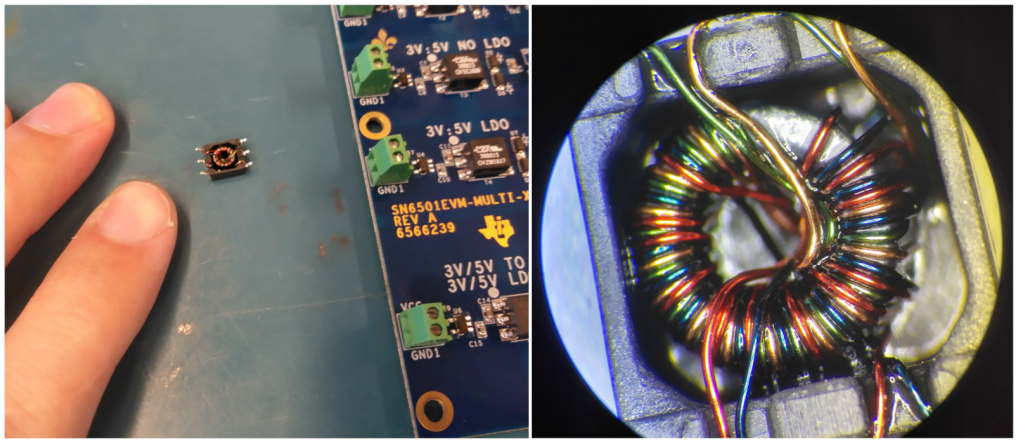


Figure 10.19: The charge reservoir networks implemented on the boards use an SN6501 transformer driver in conjunction with a 1:1.1 transformer. The latter is the 760390012 push-pull transformer by Würth Elektronik, an affordable, effective and very pretty SMD transformer.

Figure 10.20 shows the evolution of the schematic, now using the transformer-based voltage sources to drive the transformers. Although the complexity of the circuit has increased significantly, this

implementation works, and can also be configured to a much higher degree. A very similar circuit was implemented on the second iteration of the companion board. The circuit incorporates a user-configurable solder jumper, which can be used to choose between a half-wave bridge rectifier or a full-wave bridge rectifier, the latter of which will double the voltage.

Together with the 5V input voltage, and the transformer winding ratio of 1:1.1 (see Figure 10.19), the MOSFETs can be provided with a V_{GS} of either 5.5V or 11V [5]. The complete circuit that makes this possible will henceforth be referred to as a **charge reservoir network**, of which a total of six were implemented on the companion board.

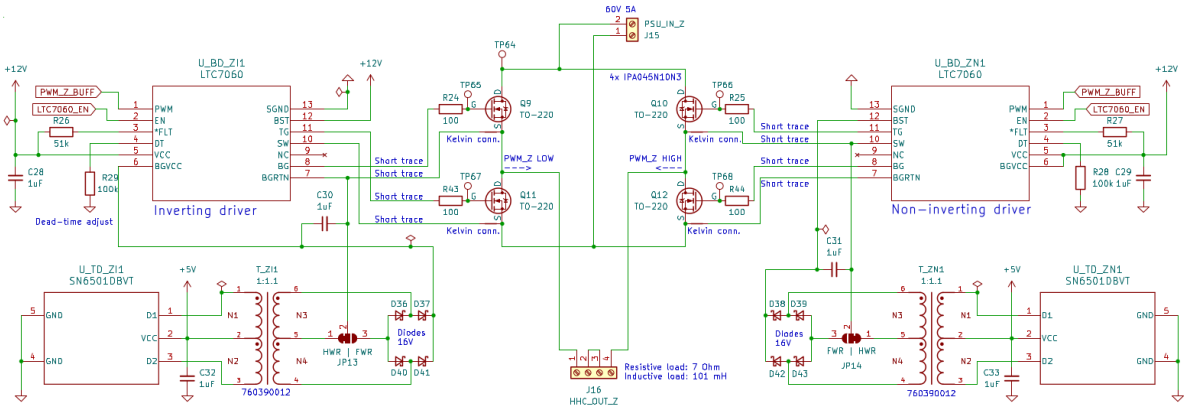


Figure 10.20: Implementation of one H-bridge channel after the second design iteration. Where the previous iteration used boost capacitors, this circuit has replaced them by the more involved but more reliable transformer-based isolated voltage sources.

Several other changes were made to this circuit during the design iteration, such as the addition of the 100 ohm gate stopper resistors on every MOSFET, which improve the robustness of the MOSFET switching by suppressing oscillatory behaviour of MOSFETs [55]. All these changes will be discussed in more detail later in this chapter.

10.1.7. MOSFET selection

Until this point, we have assumed that the H-bridge design incorporates N-channel MOSFETs, but we have not looked in detail at the MOSFETs used in the design. In this section, some selection criteria will be discussed, and a suitable part will be selected.

In the early stages of the design process, some prototype H-bridge designs were put together with general-purpose IRF520 MOSFETs, since these were on-hand in the workshop. However, we will see that these MOSFETs make for rather poor choices in this case. But how to select a better MOSFET? What properties of a MOSFETs are important for this application?

- The MOSFET should have a suitable V_{DS}, which is the maximum voltage between drain and source that the MOSFET can continuously and reliably block. At higher voltages, the body diode will start to exhibit breakdown. The 1685B power supplies intended to be used for the system can output up to 60 V, and some voltage transients are expected. As such, the MOSFET should be rated for 60 V at least, but it is more prudent to expect at least 1.5 times more as a safety factor.
- The MOSFET must also be rated for the design current of the system. Although the 1685B power supplies are rated for 5 A, it was witnessed they can comfortably output close to 7 A for a suitable load. The related properties one may find in the datasheet of a typical MOSFET is the continuous drain current I_D and the "on-resistance" between drain and source $R_{DS(on)}$ of the MOSFET. The former should ideally be high, and the latter should be low.
- Gate threshold voltage $V_{GS,th}$ indicates what voltage should be applied to the gate of the MOSFET in order to supply it with enough charge to fully turn it on. All MOSFETs also have a maximum allowable gate-source voltage $V_{GS,max}$ which may not be exceeded.

[5] Note that in practice, these voltages are a bit lower due to transformer losses and the voltage drop of the rectifier diodes.

- Maximum continuous power dissipation capacity, which is typically a function of the package type, but also of the internal construction of the MOSFET. A high power dissipation ensures the semiconductor will not be unnecessarily hot during normal operation, which increases its efficiency and prolongs its expected lifetime.
- Package insulation: Many power MOSFETs come in the popular TO-220 package, which is a three-pin device with an additional metal tab on its back. The tab is usually internally connected to the drain of the MOSFET. Any metal heatsinks directly connected to the package will therefore be at the same potential of the MOSFET drain, which can increase the risk of short-circuits. A short-circuit between the drains of an adjacent high-side MOSFET and low-side MOSFET is a form of shoot-through (see Figure 10.2d), and such an event can cause a powerful discharge spark, and may cause damage to the power supply. Typically, TO-220 can be electrically insulated from any attached heatsinks using mica insulation tapes and bushings, but there are also MOSFETs available in FPAK packages, which are encased in insulating plastic. This fully insulates the MOSFETs mechanical connection, at the cost of worse thermal dissipation properties (higher $R_{th,JA}$).
- Pin layout: It would be best if the chosen MOSFET has a common pin layout, such as G-D-S, so that if the need arises to replace the part on the board, there is a greater availability of drop-in replacements.
- Unit cost: a set of three H-bridges requires twelve MOSFETs in total, so choosing a MOSFET of significantly lower cost will have a substantial impact on the bill of materials (BOM). For context: for the first board iteration, the chosen MOSFETs made up 35% of the total component cost (PCB excluded).

Table 10.1 displays the list of MOSFETs that were considered as part of the trade-off process. All these MOSFETs have the following properties by default:

- A TO-220 package
- N-channel topology
- A G-D-S pin-out
- A $V_{GS,max}$ of ± 20 V
- An active product status

Table 10.1: Considered MOSFETs in the trade-off

Name	V_{DS} $[V]$	$V_{GS,th}$ $[V]$	I_D $[A]$	$R_{DS(on)}$ [mΩ]	**Package**	$R_{th,JA}$ $[K/W]$	**Cost** $[€/unit]$
IRF520	100	[2.0, 4.0]	9.2	**270**	TO-220AB	62	0.84
IRF640	200	[2.0, 4.0]	18	**180**	TO-220AB	62	1.75
IRF530	100	[2.0, 4.0]	14	**160**	TO-220AB	62	1.23
IPP016N08NF2S	80	[2.3, 3.8]	196	1.65	TO-220-3	62	3.3
IRF100B201	100	[2.0, 4.0]	192	4.2	TO-220AB	62	3.28
IRFB7545PbF	**60**	[2.1, 3.7]	95	5.9	TO-220AB	62	0.88
IRFB7730PbF	75	[2.1, 3.7]	195	2.6	TO-220AB	62	2.97
IRFB7537PbF	**60**	[2.1, 3.7]	173	3.30	TO-220AB	62	1.95
IPP019N08NF2S	80	[2.2, 3.8]	191	1.9	TO-220-3	62	2.57
TK3R9E10PL	100	[1.5, 2.5]	100	3.9	TO-220-3	83.3	2.09
IPA045N10N3	100	[2.0, 3.5]	64	4.5	TO-220-FP	80	2.7
IPA057N08N3	80	[2.0, 3.5]	60	5.7	TO-220-FP	80 (est.)	2.26

As can be seen in Table 10.1, some of the more garden-variety TO-220 MOSFETs from the IRF series have a hard time competing against more modern high-performance MOSFET designs. In a 2003 datasheet [56], International Rectifier lauded their own IRF520 MOSFETs by describing the associated technology as "extremely efficient" and capable of achieving "extremely low on-resistance". A mere two decades later, a power MOSFET of barely twice the price can do two orders of magnitude better.

At a worst-case continuous current of 7 A, a MOSFET with an $R_{DS(on)}$ of 5 mΩ will have to dissipate $7^2 \cdot 0.005 = 245$ mW of power. Even without any additional cooling, the temperature of a normal TO-220 package will only rise by about 15 Kelvin above ambient temperature. For the insulated TO-220-FP package, this is about 20 Kelvin. In other words, paying a little bit extra for a more expensive MOSFET with an $R_{DS(on)}$ of a few mΩ will ensure no cooling solution is required, the MOSFETs will have a long expected lifetime and high power efficiency, and will allow for a very low risk of short-circuit by allowing for FPAK variants to be used. This line of reasoning is essentially the full trade-off procedure for the MOSFET selection in Table 10.1, and the **IPA045N10N3** MOSFET is chosen for the design. For a lower-cost bill of materials, one could instead choose the IPA057, but with a V_{DS} of 80 V, it will also have a little bit less headroom for quashing voltage transients, the importance of which will become more relevant later on.

10.2. Current sense design

The magnetic field output of the Helmholtz coil pairs is directly proportional to the current applied through them, by virtue of the Biot-Savart law. To monitor the system performance, a current sensing method was designed to be able to measure the current through each coil pair, in accordance with requirement **ELEC-06-02**.

One of the most common methods of measuring current is by way of a low value **shunt resistor**, in combination with an amplifier. The basic premise is that one can put a low value resistor in series with the current path, and if its resistance is low enough, it will not significantly affect the load and will also not dissipate much power into the resistor. Still, there will be a small voltage drop across this resistor, which in an ideal world is equal to:

$$\Delta V_{shunt} = I \cdot R_{shunt} \tag{10.1}$$

As a result, if one assumes that the resistance of the shunt does not change significantly, the current going through the shunt will be directly and linearly proportional to the voltage drop. One can then use an amplifier to magnify the voltage signal into something that can be measured with sufficient precision by an analog-to-digital converter (ADC). This can be achieved by a simple opamp but in practice, this task is typically entrusted to a more refined instrumentation amplifier or a dedicated current sense amplifier device.

In reality, there are several things that complicate this process. First of all, most resistors change their resistance value as a function of temperature, and if the dissipated power into the shunt resistor is significant enough, it will no longer be a fixed value but indirectly a function of current. In other words, due to ohmic heating, dR/dT is no longer constant. In addition, the Johnson noise generated by the resistor will become more substantial as it heats up. This leads to the notion that ideally, R_{shunt} should be as small as possible, and one should preferably choose a shunt resistor with a temperature coefficient specified by the manufacturer.

However, when the shunt resistance is very small, the corresponding voltage drop across it will become small also, requiring a much larger amplification gains, or an ADC that can measure accurately down at the level of millivolt or microvolt level. The former causes the noise floor to be amplified further also, leading to a small signal-to-noise ratio (SNR) of the voltage drop signal, and the latter requires specialised and carefully calibrated hardware. Furthermore, the resistance measured by the amplifier will include also the resistance of solder connections, trace resistance of the sense lines, etc. For low shunt resistances, these can no longer be ignored and must be taken into account for accurate current measurements.

Given all this, a middle ground must be struck such that the design conforms to all its requirements. What exactly is needed here? To start, the power supplies that drive current through the Helmholtz coils are rated for an output of 5 A. A shunt resistor that is rated for a certain power dissipation P_{dis} must therefore have a resistance of:

$$R_{shunt} \leq \frac{P_{dis}}{I^2} \tag{10.2}$$

If we assume a baseline power dissipation of 500 mW, we find a value for the shunt resistance of 20 mΩ. From Ohm's law, the associated voltage drop at 5 A will be 100 mV. This signal then must be amplified with a suitable gain in order to best match the sensitivity range of the ADC.

10.2.1. Current sensing - First iteration

In the first iteration of the design, slightly different choices were made. At the time, it was intended to use the 24 bit ADC channels on the CN0554 board in its [0, 2.5] V mode, to increase the resolution of the current sensing. A shunt resistance of 5 mΩ was chosen, with the selected part being the CSS2H-3920-5L00 by Bourns [57]. This is a large, high-current, two-terminal resistor, which was integrated to the current sense amplifier by Kelvin connection (see Figure 10.21). This part was chosen for its large size, which reduces temperature influences on the measurement, reduces the added resistance of the solder connection, and makes it easy to implement the Kelvin sensing on the board layout.

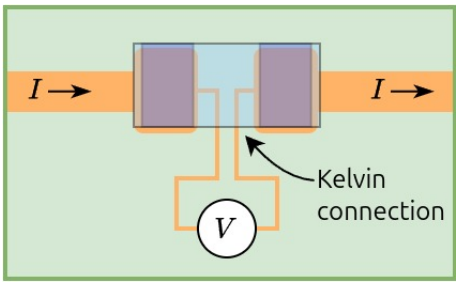


Figure 10.21: A Kelvin connection is a method to perform electronic sensing that compensates for lead resistance. This can be very useful for applications where low resistances are being measured, such as measuring current with a low-resistance shunt resistor.

To amplify the voltage drop across the long with a current sense amplifier gain of 100. At a current of 5 A, this should yield a voltage drop across the shunt resistor of 25 mV, which with a gain of 100 becomes 2.5 V. The part selected for the amplification is the INA2290A3 by Texas Instruments [58], which consists of two of the common INA290 current sense amplifiers in one integrated chip. This means that only two chips are needed for measuring three current channels. The chip comes in five gain variants, with the **A3** variant having a fixed gain of 100.

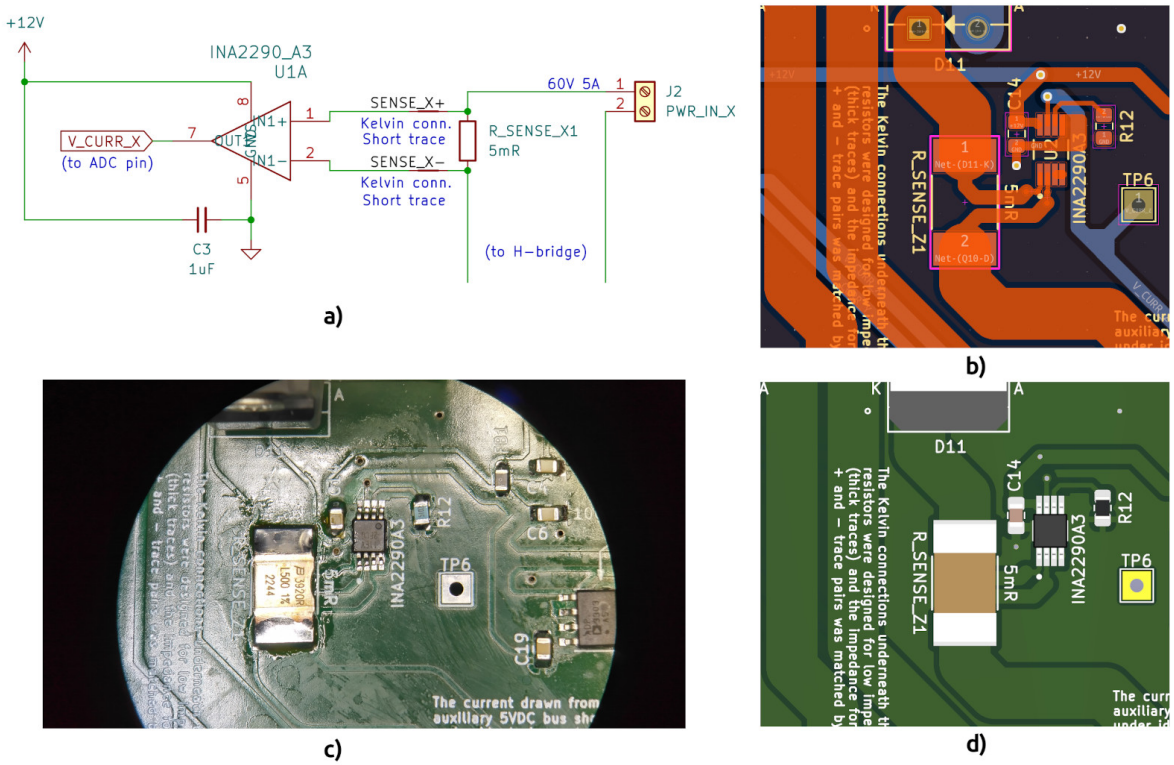


Figure 10.22: First iteration of the current sensing implementation. **a)** Circuit for a single current sensing channel **b)** PCB layout for this circuit, with the Kelvin connection clearly visible **c)** Microscopy of the implementation on PCB **d)** CAD 3D render of the layout.

However, whilst testing the implementation, a number of problems were discovered with this setup. The signal coming from the current sense amplifier was very noisy, and did not correspond well with the expected values. After configuring an oscilloscope to average 10k points as a way to filter high frequency noise and varying the current, the output voltage did seem to correspond with the current to an extend, but with a few caveats:

1. No output could be detected under 400 mA of the current.
2. Very high noise was present in the output at all times.
3. Around 400 mA, the noise would increase by a few decibels.
4. Over 500 mA, the noise would reduce again, and from here on out, the heavily averaged voltage output would correspond roughly linear with the input current, albeit with a different V-I slope than expected.

The lack of output at lower current values could be explained to an extent when one simulates the internal circuit of the INA290 (see Figure 10.23), which uses a combination of a differential amplifier and MOSFET. Put briefly, this topology is not able to measure zero-input, although the manufacturer datasheet suggests it should be able in this context to measure well below 400 mA. The simulations that were performed also suggest that with a higher value shunt resistance, the voltage drop is higher and this dead-zone" should reduce, allowing lower currents to be measured.

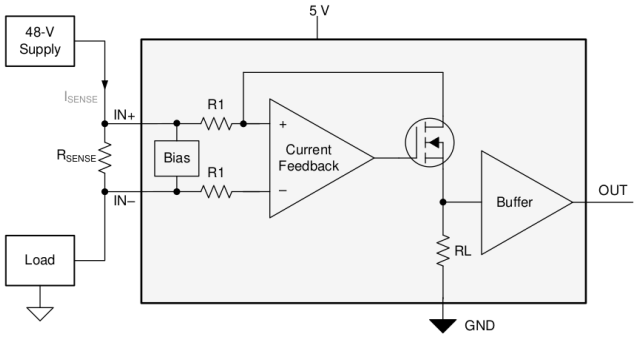


Figure 10.23: Internal model of an INA290 current sense amplifier as given by the manufacturer [58].

However, after much troubleshooting it was discovered that the main issue with the circuit was human error: instead of ordering the CSS2H-3920-5L00F, the similar component CSS2H-3920-L500F was ordered during the acquisition process. These current sense resistors are visually identical, except for a different product code, but the latter has a shunt resistance of 0.5 mΩ, rather than the intended 5 mΩ! As a result, the voltage gradient correlated with the sensed current is such that at the maximum designed measurable current (5 A), the sense voltage would be 164 mV. Not only does this reduce the signal-to-noise ratio, making it more difficult to perform accurate measurements, but it also reduces the effective bit depth of the ADCs by more than 3 bits, which is wasteful.

After discovering this error, the observed behaviour of the current sense circuit makes a lot more sense, both the high degree of noise seen in the output signal, as well as the unexpected linearity of the output.

10.2.2. Current sensing - Second iteration

After identifying the main issue with the first iteration (the substitution of an incorrect part), the design could have simply been left as-is after procuring the correct part. However, the design was adjusted in a few areas because of changes elsewhere in the system, and the desire to add a few extra features.

First of all, for specific configuration reasons, it was not very straightforward to configure certain ADC channels on the CN0554 with the [0, 2.5] V range whilst leaving the other channels untouched. Since the added resolution was not strictly required, it was decided to use the default ADC input range of [-13.75, 13.75] V, and adjust the shunt resistance and amplifier gain in response.

Second, the first iteration design would not be able to sense anything above 5 A, because at this current, the ADC will saturate at 2.5 V. However, it was discovered during testing that the BK Precision 1685B power supplies will comfortably output nearly 7 A, which would be undetectable by the design. Since it is desirable to know this information for the purposes of error detection, some headroom was built into the design such that the ADC would not saturate until the measured current would be well outside of the design range.

Thirdly, when it came to part selection, some improvements could also be made. There are advantages to switching to a four-terminal sense resistor that features built-in compensation for the sense lines. In addition, for design flexibility reasons, it would be better to select a part that is part of a family with several resistance values below and above the intended one, so that different values can be used as drop-in replacements if needed.

Another shortcoming is that the data output by the current sense amp is quite noisy, but a adequately linear current-voltage relationship can be observed when averaging a large number of these data points. The ADC that is used is high bit-depth but low speed. This means that it will measure the noise accurately, which is not very important in the grand scheme of things, and in addition it will not be able to acquire many points to average over time. To address this, it makes sense to average the noisy data beforehand with an analog filter, and this will come with the additional benefit that the computational load of obtaining current sense data will be reduced. However, care must be taken that the current sense signal is not loaded down so much that it will affect the ADC measurements. A single-pole passive RC filter with a low R value at the output of the current sense amp will likely be sufficient, and if at a late stage of production it is decided that the filter does more harm than good, it can be left out by simply replacing the resistor with a 0Ω link and leaving the capacitor unpopulated.

Additional requirements and considerations

With this discussion out of the way, let's now redesign the current sensing solution. The INA290 will still be used as a current sense amplifier, although a different gain variant may be used. The CN0554 board with the AD7124-8 ADC chip will also be used. We arrive with the following design requirements and considerations:

1. Maximum expected sensing current of 7 A.
2. Maximum allowable voltage to and from the current sense amplifier below 12 V.
3. Maximum allowable voltage into the ADC no greater than 10 V.
4. The current sense resistor shall have a power dissipation rating at least twice as high as expected at maximum current.
5. The current sense resistor shall have a common footprint and be part of a component series or product family that has comparable resistors of various resistance values available above and below the design resistance.
6. The current sense leads to the current sense amplifier shall be short and low impedance.
7. The current sense amplifier shall have an optional single-pole filter at its output, which uses a low resistance and has a cutoff frequency of around 100 Hz.

Other considerations:

- Current sense resistors with high resistance values allow for a current sense amplifier with lower gain. This results in a better signal-to-noise ratio.
- Current sense resistors with low resistance values and high power dissipation values allow for lower and more stable operating temperatures, which results in better sensing linearity and lower Johnson noise output, respectively.
- Current sense resistors with low value may contribute to an increase of low-current dead-zone output by the INA290, and therefore a high current sense resistance may be desirable. This will have to be balanced with the previous consideration.
- Current sense resistors with larger soldering connections should be preferred, as they will facilitate a better electrical and thermal conduction path between the resistor and the surrounding PCB tracing.

Component availability

After some research into available components available at Digikey, the following shunt resistor series were identified:

Table 10.2: Prospective shunt resistor series found on Digikey

Manufacturer	Series	Size	Values [mΩ]	$P_{D,max}$ [W]	Pad area [mm^2]	Cost [€/$unit$]	Comment
Littelfuse	L4CL	2512	3 - 16, 18, 20-22, 24, 27, 28, 30, 33, 34, 36, 39, 43, 47, 50, 51, 56, 59, 62, 68, 75, 82, 91, 100	1	1.00	1.01	
Ohmite	LVK24	2412	10, 12, 15, 20, 25, 30, 33, 39, 47, 50, 75, 100	1	2.52	1.40	
TFT Corp.	D1WRL-L4	2512	1, 2, 5	0.5 - 1	1.00	1.50	The 1, 2 mOhm are 0.5W rated
Vishay Dale	WSK2512	2512	1, 2, 4, 5, 10, 20, 25,	1	2.42	1.65	Opposing sense pads
Susumu	RL3264	1225	1, 4, 100	1-2	1.08	1.50-2.80	The 1, 4 mOhm are 2W rated, also these have longitudinal pads
VPG	CSM2512S	2512	5, 10, 20, 25, 30, 40, 50, 100	1	1.6	4.10-25.00	Cost per tray of 25, also opposing sense pads

Of these, the series by TFT Corp. and Susumu were disregarded due to the small variety of resistance values available within the series. In the interest of creating a design with drop-in replacement in mind, it was also investigated which of the remaining series would be drop-in compatible, and what this would imply for the PCB footprint layout.

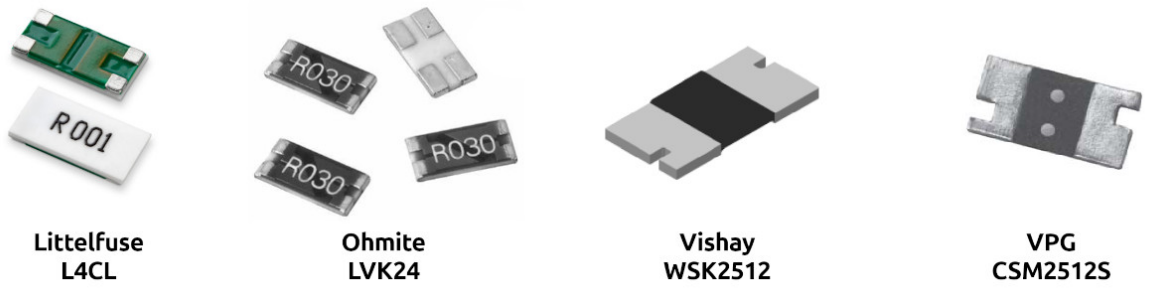


Figure 10.24: Four viable current sense resistor series [59] [60] [61] [62]. **a)** and **b)** together are footprint compatible, and **c)** and **d)** are together also footprint compatible.

It was found that the resistor series Littelfuse and Ohmite can be made to fit on the same footprint, as the sense terminals are on the same side of the resistor. In contrast, the series by Vishay Dale and VPG have the resistors laid out such that the sense terminals are on opposite sides of the footprint. As such, the L4CL and LVK24 series can be used interchangeably, and the same is true for the WSK2512 and CSM2512 series. However, both groups are mutually exclusive to one another. This can also be seen in Figure 10.24.

Resistor sizing

For the second design iteration, the LVK24 series resistor was chosen, owing to its low unit cost, large pad area, and good availability. But what value resistor should be used? Given that the maximum expected continuous current of 5 A, and a typical dissipation rating of 1 W, the resistance of the current sense resistor may not exceed:

$$R_{shunt} < \frac{PD_{max}}{I_{max}^2} = \frac{1}{5^2} = 40\ m\Omega$$

However, given that the 1685B power supplies can upon misconfiguration output nearly 7 A, the resistance limit for the current sense resistor should instead be:

$$R_{shunt} \leq \frac{1}{7^2} = 20\ m\Omega$$

In order to keep some options open for the exact resistance value of the resistor, let us evaluate several values at the same time: $[1, 2, 5, 10, 15, 20]\ m\Omega$. The maximum voltage drop across such a current sense resistor is described by:

$$\Delta V_{shunt} = \overline{R}_{shunt} \cdot I_{max} = \begin{bmatrix} 1 & 2 & 5 & 10 & 15 & 20 \end{bmatrix} \cdot 7 = \begin{bmatrix} 7 & 14 & 35 & 70 & 105 & 140 \end{bmatrix}\ mV$$

The INAx290 component comes in five gain variants:

$$\overline{G} = \begin{bmatrix} 20 & 50 & 100 & 200 & 500 \end{bmatrix}\ V/V$$

Which leads to the following output voltages at 7 A:

$$V_{sense} = \overline{G} \otimes \overline{R}_{shunt} = \begin{bmatrix} 0.14 & 0.28 & 0.7 & 1.4 & 2.1 & 2.8 \\ 0.35 & 0.7 & 1.75 & 3.5 & 5.25 & \mathbf{7.0} \\ 0.7 & 1.4 & 3.5 & \mathbf{7.0} & 10.5 & \mathbf{14} \\ 1.4 & 2.8 & \mathbf{7.0} & \mathbf{14} & \mathbf{21} & \mathbf{28} \\ 3.5 & \mathbf{7.0} & \mathbf{17.5} & \mathbf{35} & \mathbf{52.5} & \mathbf{70} \end{bmatrix}\ V$$

From this can be seen that there are several combinations of gain and shunt resistance that lead to an output voltage of 7 V at 7 A input current. This is the highest output voltage that can be obtained without going over 10 V. Values in **red** will saturate the output of the INA2290 chip, as it cannot output more than its supply rail, which will be 12 V. With a sensing range up to 13.75 V, the ADC should never be saturated and is in no danger of being damaged due to overvoltage.

After trading off, a shunt resistance of 20 $m\Omega$ was chosen in combination with the A2 variant of the INA290, which has an amplifier gain of 50. This combination satisfies all requirements, and it also has a relatively high shunt resistance, which generates a strong signal that should propagate well through the current sense amplifier, with minimal dead-zone. The only risk is that at 5 A continuous current, approximately 0.5 W will be dissipated into the shunt resistor, which may lead to a less linear current-voltage curve due to ohmic heating of the sense resistor. However, the temperature-resistance relationship of these resistors is usually quite linear and documented by the manufacturer, so this is something that can be corrected for. If for any reason one of these components is no longer available, there are three other combination of shunt resistance and amplifier gain that can be dropped in with very similar results.

In addition, to help with thermal management, shunt resistors with large solder pads should be preferred over those with small pads, and during the PCB design stage sufficient copper tracing should be available nearby to help dissipate the heat away from the sense resistors. If these strategies fail, one can instead use a lower value current sense resistor, and given that the sense resistor is part of a larger product family by requirement, a suitable drop-in replacement should always be available.

Optional output filtering

The output filtering comprises a simple passive RC network connected to the output of the current sense amplifier. The amplifier has a buffering stage before the output, so exposing the output to a small extra series resistance is not expected to meaningfully affect the analog output. Depending on the input impedance of the ADC however, there may be a voltage offset, and so this will have to be investigated and possibly corrected for with calibration. If all strategies fail, the capacitor of the RC network can be left unpopulated and the resistor replaced with a 0 Ω link.

For now, a resistor value of 100 Ω and a 10 μF has been selected. The capacitor has to be rated for 16 V or higher. When arranged as a passive RC filter, these components will act as a low-pass filter (LPF), with its cutoff frequency equal to:

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 100 \cdot 10^{-5}} = 159.2 \text{ Hz}$$

As this is a first-order filter, the roll-off of this filter will be around 20 dB/decade, meaning that noise with frequency components larger than 10 times the cutoff frequency (i.e. larger than 1592 Hz) will be attenuated by at least 20 dB relative to noise at the cutoff frequency. Similarly, noise above 16 kHz will be attenuated by at least 40 dB, and so forth. This improves the signal-to-noise ratio because the filter disproportionally suppresses high frequency noise compared to the relatively steady and slow-changing current signal.

Second iteration of the current sense implementation

In Figure 10.25, the implementation of the current sensing can be seen in more detail. All the considerations discussed in this sections found their way into this design. The INA2290 chip is still used, but the **A2** variant with a gain of 50 is now used instead of the **A3** variant. The single unused current sense amplifier is configured appropriately [63]. Although there are many ways to optimize shunt resistor footprints [64], the basic four-terminal footprint of the shunt resistor is used for simplicity and clarity.

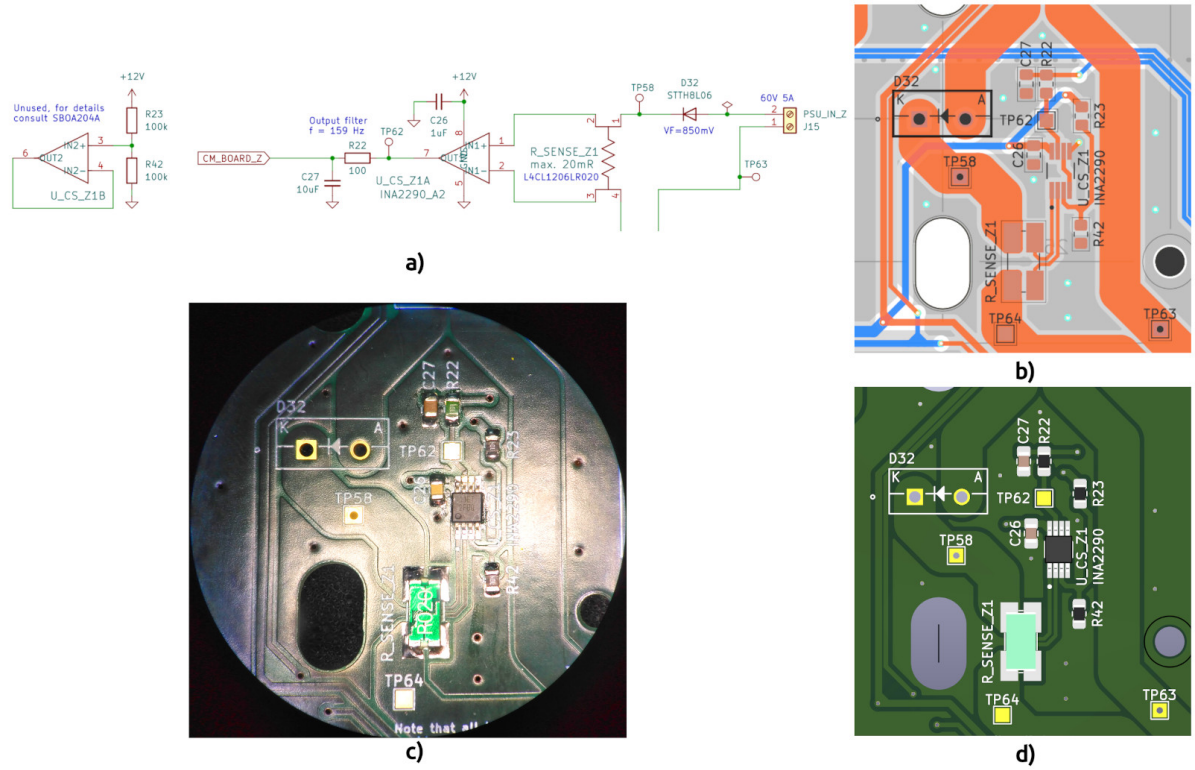


Figure 10.25: Second iteration of the current sensing implementation. **a)** Circuit for a single current sensing channel **b)** PCB layout for this circuit **c)** Microscopy of the implementation on PCB **d)** CAD 3D render of the layout.

10.3. Thwarting inductive voltage transients

As we discussed before, the H-bridge circuits will drive an inductive load in the order of 100 mH. When a current is allowed to flow through this inductive load, a magnetic field will be established. The energy of this established magnetic field is proportional to the inductance and the square of the current:

$$E_{field} = \frac{1}{2}LI^2$$

For a current of 5 A, the load will be storing a field energy of 1.25 J. What happens when the H-bridge switches the polarity of the current? Well, the current through the coils will go from 5 A to -5 A in a short time. According to Lenz's law, we can expect that a voltage will be induced in proportion to the rate of change of the current:

$$V_i(t) = L\frac{dI}{dt}$$

The IPA045 MOSFETs in the H-bridge have a typical turn-on time of 72 ns and a turn-off time of 65 ns [65]. The shortest possible dead time that the LTC7060 bridge driver can be configured to is 32 ns [54], but in practice it should really be configured to be at least equal to the turn-off time of the for the IPA045. So let us assume a fast switch time to be in the order of 140 ns. We can then expect an induced voltage of:

$$V_i(t) = 0.1\frac{10}{140 \cdot 10^{-9}} = 7.14MV$$

Due to the internal resistance and other parasitics inside the circuit, it is unlikely that the induced voltage will actually be in the order of **7 megavolts**[6], but one can certainly expect a hefty voltage transient to be unleashed into the circuit, where it can cause serious problems. Of primary concern is the positive terminal of the power supply, which is almost certainly not designed to handle such a spike. But sensitive semiconductor devices may also be blown apart by high voltage transients. In fact, this is something that MOSFET are somewhat notorious for: the very thin oxide layer insulating the gate from the drain and source can be damaged very easily by electrostatic discharge (ESD).

This section is concerned with designing the H-bridge circuit in such a way that these voltage transients can be dealt with in a reliable way. One could argue that one could just write software that automatically limits the slew rate of the current, and conditional checks that ensure that the H-bridges are not allowed to switch unless the channel current is nearly zero. However, this is a clear case where requirement **ELEC-14-01** is relevant; this is the kind of safety feature that should be delegated to predictable and reliable analog electronics. Otherwise, naive software usage or software bugs may destroy the system hardware. So, this problem will be solved through electronic design.

There is a large variety of electronic devices and circuits that can be used for "snubbing" or "transient suppression" [66] [67], such as varistors, diode clamps, filter circuits, crowbar circuits, spark gaps and other discharge devices, etc. The characteristics of the particular voltage transient we are dealing with is that is a low power transient, only about 1 J can be stored in the coils, and signal integrity or distortion characteristics are really of no concern here. As a result, we will enhance the circuit with simple "sponges" that can absorb the energy of the spikes, and "barriers" that can deny the spike entrance into certain areas of the circuit.

10.3.1. Dropper diodes

The first line of defence is putting a big diode directly in front of the positive output of the power supply. This diode should be able to handle a relatively high voltage before it starts to exhibit reverse breakdown. There is in fact another reason for placing this diode here, rather unrelated to transient voltage suppression: it was discovered early on that the 1685B power supplies exhibit somewhat poor current output linearity when they have to supply this current at a compliance voltage lower than 0.5 V.

[6] The magnitude of the actual spike was not measured, due to concerns about blowing up the front-end of an oscilloscope channel, and the lack of availability of HV probes.

An effective, though not especially elegant solution is to put a diode with a forward voltage V_f of at least this 0.5 V in series with the load, so that even if the load is a short-circuit, the power supply will need to maintain at least a little bit of voltage to supply this current, pushing it out of this non-linear region.

The solution is relatively inelegant because the diode will consume quite a bit of power: at a continuous 5 A, a diode with a forward voltage of 0.85 V will dissipate 4.25 W, which is waste heat and implies that measures have to be taken to sufficiently sink the heat from these diodes. However, since the diode also functions as an effective voltage transient barrier for the positive output terminal of the power supply, this is deemed a good addition to the circuit. In this thesis, we refer to this diode as a **dropper diode**, since one of its main jobs is to cause a voltage drop.

Figure 10.25 shows the dropper diode in the circuit of the Z-channel as **D32**. The part that was chosen is the STTH8L06 by ST [68], chosen for the following reasons:

1. It has typical voltage drop of 0.85 V, which is safely above the required 0.5 V, but not much higher that much unnecessary waste heat is generated.
2. It has a repetitive peak reverse voltage V_{RRM} of 600 V, meaning it will reliably block voltage transients up to 600 V.
3. It has a reverse recovery time of 75 ns, meaning that it will recover about as fast as the time it takes for the H-bridge MOSFETs to switch again and cause more trouble.
4. Since it is expected to dissipate a few watts of heat, this diode comes in a D2PAK and TO-220 package, so thermal management should be very feasible.
5. It is manufactured in TO-220FPAC and insulated TO-220AC variants, which have electrically insulated tabs, allowing multiple diodes to be mounted to the same heatsink without any extra insulation hardware. Of these two, the insulated TO-220AC variant has superior thermal conductivity, and is preferred if available.
6. At a cost of approximately €1.70 per unit, it is not particularly expensive.

10.3.2. MOSFET body diodes

An extra measure of protection comes from the fact that at 100 V, the IPA045 MOSFETs in the H-bridge circuit will start reverse breakdown and therefore conduction [65]. This will mean that before any voltage transient can travel to the dropper diode and reach more than 600 V, the conducting body diodes will start routing the energy elsewhere into the circuit, such as back into the load, but also to the negative terminal of the power supply, which should stand a much better chance of handling this.

10.3.3. TVS diodes

However, the first line of defence against transient voltage comes from specialized transient voltage suppression (TVS) diodes that will both absorb a portion of the energy of the spike, and allow the rest of the energy to freewheel back into the load, where it can be absorbed by its impedance. These TVS diodes should become active at or just above the maximum nominal current supply voltage, and should ideally clamp the voltage to below 100 V, so that the body diodes of the H-bridge MOSFETs will not yet conduct in reverse.

To briefly refer back to subsection 10.1.7, this is why the IPA045 MOSFET has a distinct advantage over the IPA057, which only has a V_{DS} of 80 V rather than 100 V. As a result, if the IPA057 were used, the voltage region in which the TVS diodes would be clamping would be more narrow. For this reason, it pays to have a little bit more voltage headroom on the body diodes of the MOSFETs.

During the first design iteration, six zener diode from the BZT03 series were used; two per channel, one in reverse to the other. This diode was the BZT03C75, with a nominal zener voltage at 75 V and a maximum clamping voltage of 103.5 V. However, it was later found that these zener diodes do not make the best TVS diodes for this particular application, because they have a maximum RMS clamping current of 2.9 A [69]. To put this to the test, it was subjected to a scenario where one pair of Helmholtz coils was driven by a current square wave. Figure 10.26 and Figure 10.27 show some details of the test setup.

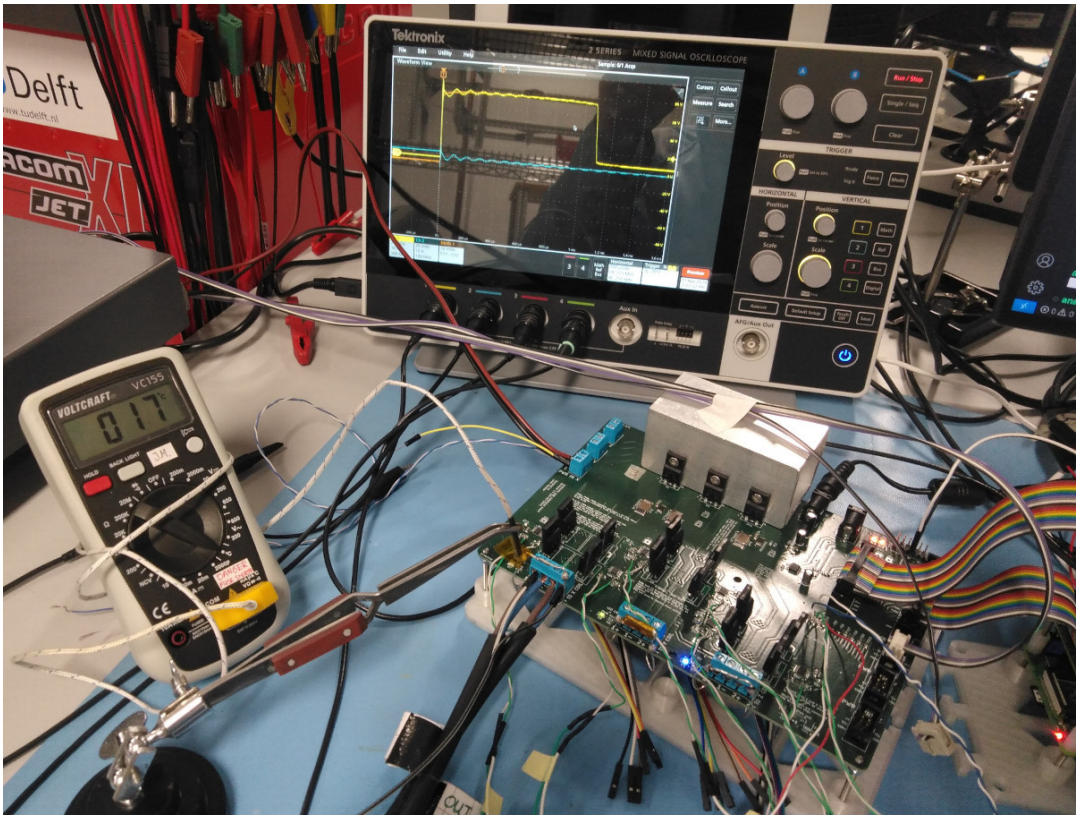


Figure 10.26: Test to characterize the performance of the zener diodes used for transient suppression. The zener diodes were connected together anode-to-anode and then in parallel with the load. Oscilloscope probes at either end measure the voltage, and math function is used to find the difference. Temperature is monitored using a thermocouple and a nearby infrared camera.

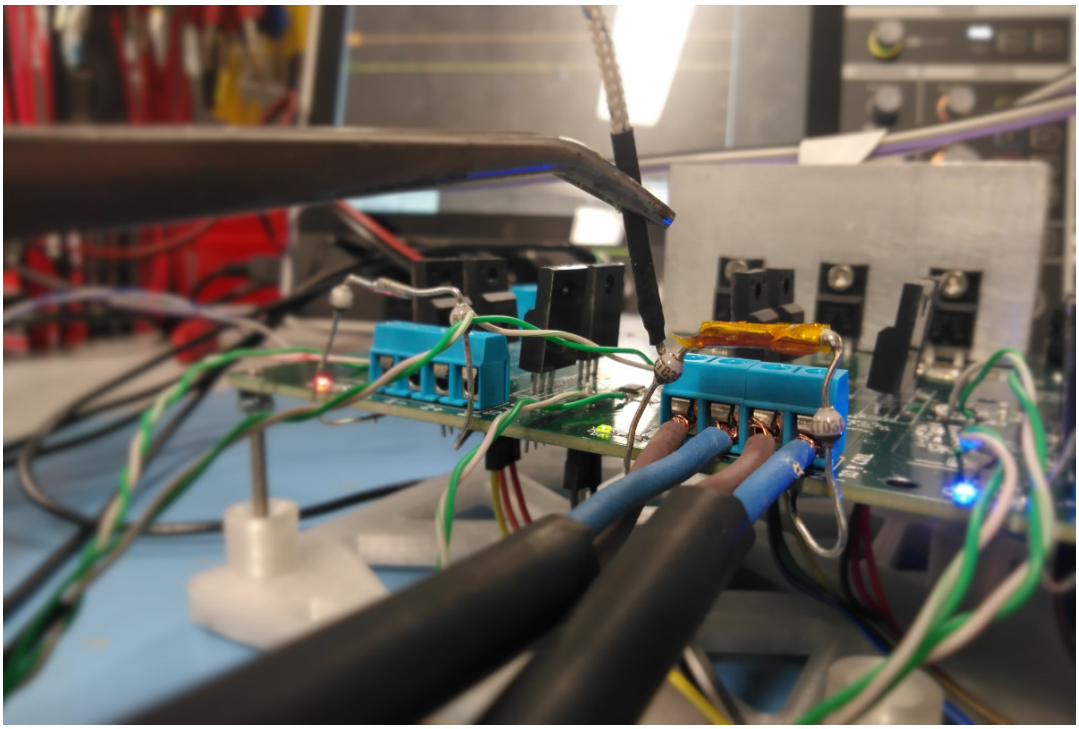


Figure 10.27: Close-up of the zener diode pair in parallel with the load. The blue-brown cables connect to the Helmholtz coil pair. The NTC thermistor makes contact with the body of the zener diode.

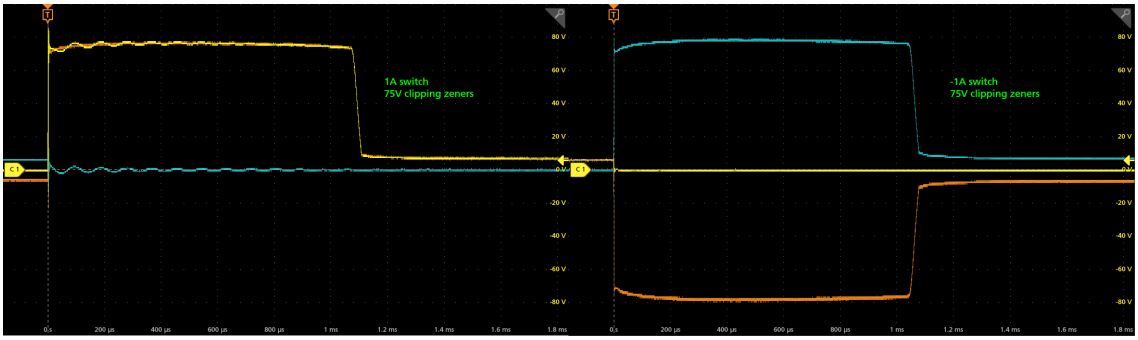


Figure 10.28: Zener diode clamping response to a 1 A square wave. Yellow trace and blue trace are either side of the load. Orange trace is a math trace subtracting blue from yellow. On the left, the voltage spike is "negative" because of this convention.

For a square wave with an amplitude of 1 A, the clamping behaviour of the zener diode can be seen from the oscilloscope data in Figure 10.28. A few things can be observed. First, the zener diode is working as intended, clamping the voltage voltage transient at a little over 75 V. The spike takes approximately 1.1 seconds to dissipate. The area under this curve is essentially the portion of the coil energy that has been absorbed into the zener diode, with the remainder of the energy feeding back into the load and dissipating elsewhere.

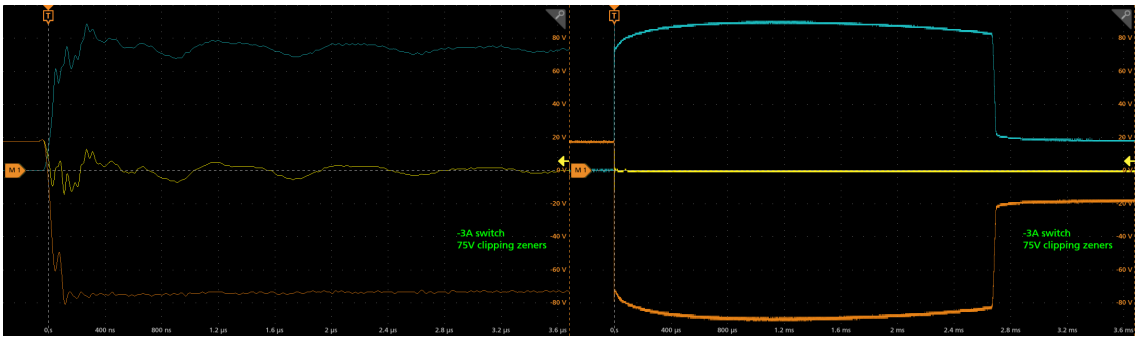


Figure 10.29: Clamping response of a 3 A square wave at different time bases. Left: 400 ns/div. Right: 400 us/div.

Figure 10.29 show the response for a 3 A polarity switch. Note the increased clamping voltage at this higher current (around 90 V peak here) and the increased transient time of approximately 2.7 ms, owing to the larger amount of inductive energy in the coil and the steeper dI/dt gradient.

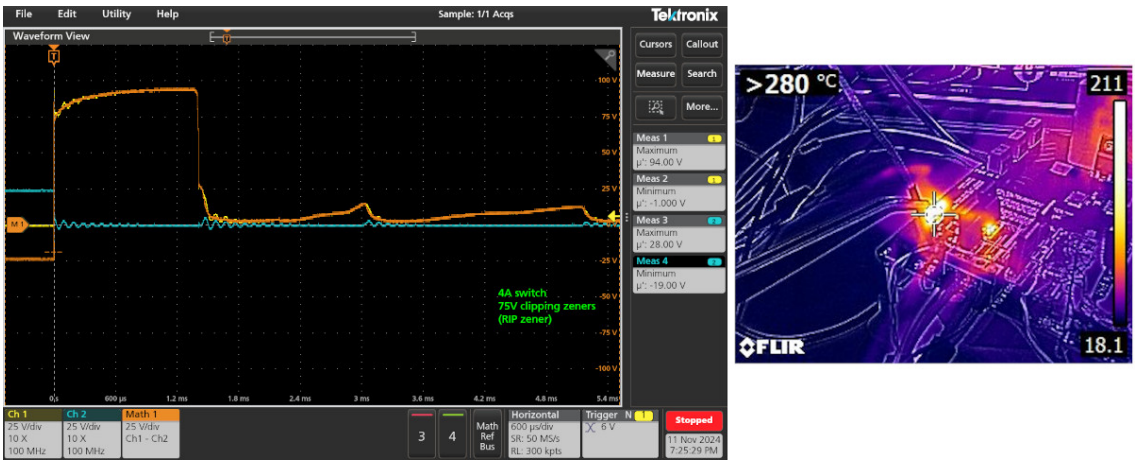


Figure 10.30: Clamping response of a 4 A switch. At 1.4 ms into the transient, the zener diode cannot handle the energy and fails in near short-circuit. The dissipated energy rapidly mounts due to the current, and the device gets so hot it starts to smoke.

To the surprise of nobody who carefully read the BTZ03C75 datasheet [69], the zener diode cannot handle a switch at a current of 4 A. When it fails, it seems to fail as a near short-circuit, which causes it to rapidly heat up.



Figure 10.31: Replacements for the unsuitable BZT03C75 [69] as a voltage suppressor diode clamp. The 1.5KE [70] and 1.5SMB [71] devices handle about five times the pulse current, and the 15KPA [72] device handles over forty times the pulse current of the BTZ03.

In response, a few suitable replacement components were found that can handle increased pulse currents. Littelfuse produces a number of nice TVS diodes that will suit this application very well. There is the 1.5KE series, which for the same zener voltage of 75 V and maximum clamping voltage of 103 V handles a peak pulse current of 14.8 A [70], where the BTZ03C75 handled only 2.9 A. The 1.5SMB [71] has similar specifications to the 1.5KE devices, but is in a surface mount form factor. The 15KPA series is even more robust, and should be able to handle 123.8 A of pulse current under the aforementioned conditions [72]. The latter are in the order of €10 per unit however, meaning they should only be used if absolutely necessary.

Even better, these devices are available also in bidirectional variants, so a single component will take care of clamping voltage spikes of either polarity.

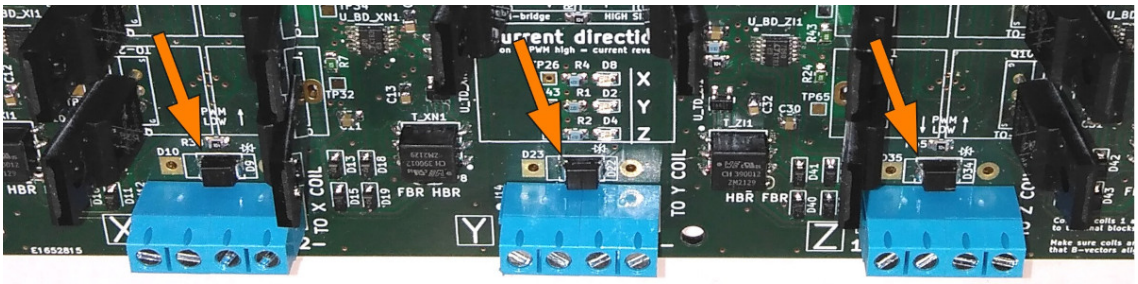


Figure 10.32: Littelfuse 1.5SMB TVS diode clamps on a later iteration of the companion board.

10.3.4. Bleeder resistors

The discussion of this section would not be complete without mentioning another edge-case that can cause voltage transients. When the coil is being supplied with current, but suddenly all H-bridge MOSFETs **turn off**, no current can flow to or from the coil. This situation can occur if power to the MOSFETs drivers is lost, or if they are commanded to turn off as part of a safety feature. The magnetic field erected by the coil, along with its energy, will remain suspended, ready to be unleashed when any of the MOSFETs turn on once again. To prevent this from causing problems, a bleeder resistor is installed in parallel with the inductive load, so that the load can in fact free-wheel at any time, albeit with a low current. In addition, leakage currents through the MOSFETs will likely also help drain the coil of its energy, but this current will be small and more importantly, uncontrolled. Therefore it is better to install the bleeder resistor so that the system as a whole behaves predictably during this scenario.

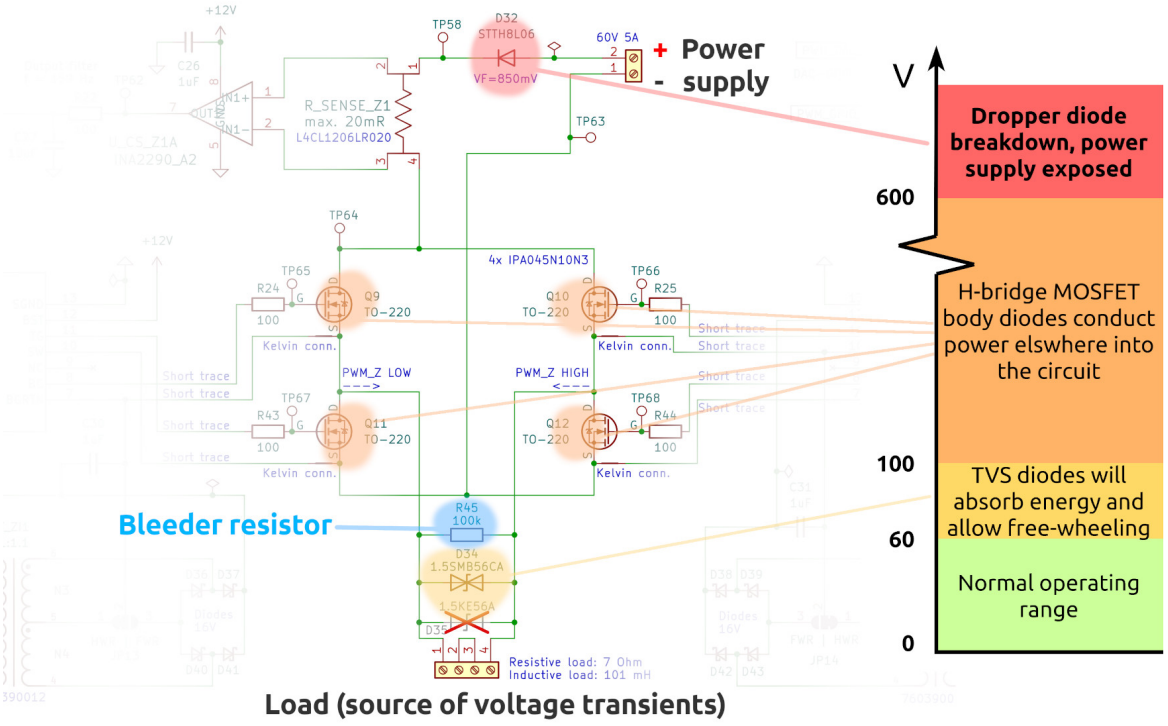


Figure 10.33: Overview of the transient voltage suppression strategy with which the H-bridge circuit is designed.

To wrap up this section, Figure 10.33 provides a summary of the various stages of the transient voltage suppression mitigation, at what voltages they come into play, and where they are located on the circuit diagram.

10.4. Companion Board - First Iteration

As mentioned in the introduction of this chapter, two full PCB implementations were developed during the thesis. Each board marks the final stage of a design iteration of the system electronics. The previous sections have focused on specific topics, that of H-bridge circuit design, current sensing, and transient suppression, and have covered their development across both design iterations.

In contrast, in this section will provide a general overview of the first design iteration of the electronics. As this design is unquestionably obsolete, this section is of lesser interest to those who wish to replicate the magnetic testing system described in this thesis. However, it is well-suited for those wishing to dive deeper into the design process, why certain design decisions were made, and uncover what engineering lessons were learned along the way. After all, there would have been no need to perform a second design iteration if there were no significant issues with the first iteration, and so to understand the impact of the second iteration, it is important to know from where it started off.

10.4.1. Implementation Functions

Even though the associated costs have gone down significantly, the production of printed circuit boards is still an involved and time-consuming process. Its design is a very deliberate process, sensitive to error. After the board has been produced, fixing errors is not without hassle. As such, before such a task is undertaken, one must have a clear idea of the functions that the board should implement, and under what conditions.

The requirements of chapter 8 help guide this process, but it is still useful to explicitly list all the functions that the companion board should fulfill:

1. Implement three polarity inversion channels, each of which take a unipolar output from a constant

current supply, and returns it either straight or with its polarity inverted. A single high/low control signal should engage/disengage the polarity inversion. This implements requirement **ELEC-05-01**.

2. The output to the coils should be a four-connection output, so that two coils can be connected in series without having to separately connect them. This will make it easier to maintain the system, especially if it ever has to be moved.
3. Each channel should implement a form of current sensing (see **ELEC-06-01**).
4. Each channel should feature a voltage drop of at least 0.5 V to ensure that the current supply is operated in its linear domain.
5. Each channel should have sufficient transient suppression capability to ensure counter-electromotive forces from the inductive loads will cause no problems, in accordance with **ELEC-07-01**, **ELEC-08-01**, and **ELEC-14-01**.
6. The board should have regulated power rails at 12 VDC and 5 VDC to power on-board components, as well as external peripherals, such as optional cooling fans. There is as of yet no internal use for the 5 VDC rail, but it was exposed on header pins for external peripherals.
7. The board should function as a central interconnect between other devices of the system, so that cable management remains straightforward and cable harnesses or looms are simple. The devices that the companion board should interconnect are:
    - 12VDC power from mains via barrel jack socket adapter
    - AlphaLab magnetometer (power and data)
    - ADC and digital-to-analog converter (DAC) connections from the CD0554 board
    - Common grounding with the Raspberry Pi/CN0554 boards
    - Constant current power supplies (3x control signal and grounding)
8. In line with **ELEC-13-01**, the board should have visual indicators to help indicate the internal state of the board:
    - Status of 12V rail (amber light emitting diode (LED))
    - Status of 5V rail (amber LED)
    - H-bridge status of X, Y, Z channels, colour coded with red/green/blue LEDs
9. Also as part of **ELEC-13-01** and to help during development, the board should figure test points where this is deemed useful.
10. Additionally, to align with **ELEC-13-01** and **SW-12-01**, clear silkscreen indication should be the norm. The user should not have to (for they will likely not) consult any manuals to find:
    - The location of the current channel input and outputs.
    - The default direction of current flow through the H-bridges.
    - The meaning of the LED indicators placed on the board.
    - The function or destination of any header pin connections.
    - The location, designation, and orientation of any board components, including through-hole components such as MOSFETs, diodes, and bipolar capacitors.
    - The version of the board.
11. The board layout should facilitate cooling of the three dropper diodes, and optional cooling for the twelve H-bridge MOSFETs.
12. The board should contain insulated mounting points that are close enough together so that the board will not sag under its own weight.
13. Any traces with significant current should have their impedance minimized to avoid excess heating of the board and reduce losses.
14. At least one layer should be a ground plane in order to improve trace routing and act as a thermal sink.

15. Any board layout procedures that will reduce the fabrication cost of the board should be implemented, such as sticking to default parameters for e.g. layer thicknesses, keeping the board area relatively small, etc. This is in line with **TOP-08-01**

10.4.2. Circuit Overview

The complete schematic of v1.0 of the companion board can be found in Appendix C, but we will take some time to discuss aspects of it here.

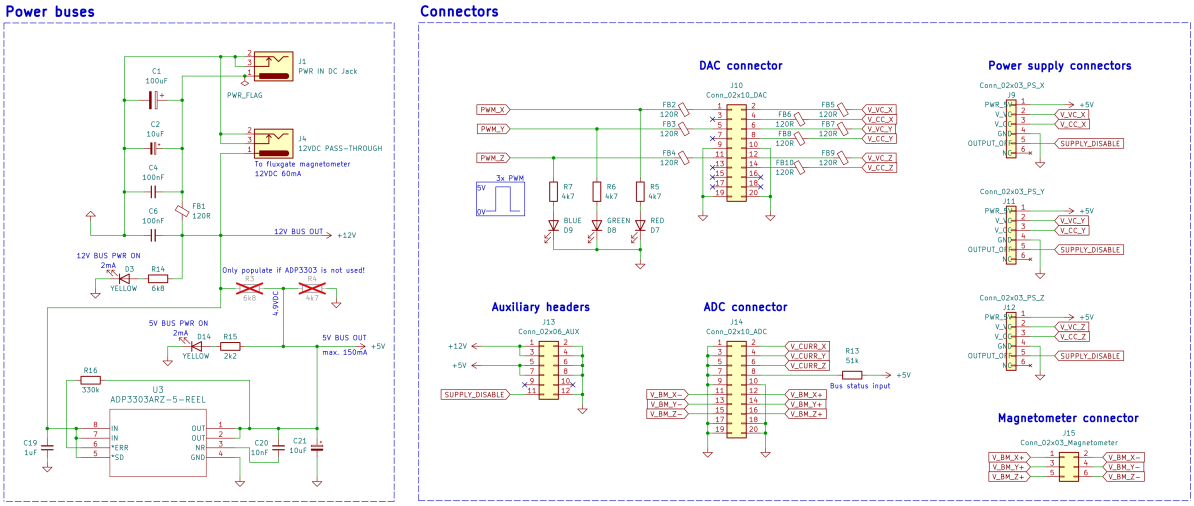


Figure 10.34: Schematic of the power section (left) and the connector routing (right) of the Helmholtz cage companion board v1.0.

The left side of Figure 10.34 shows the input power section of the board. It takes a 12VDC input via barrel jack, filters any ripple on the line with capacitors `C1`, `C2`, `C4` and `C6`. `FB1` provides some extra suppression of high frequency noise on the input. The 12V line is connected to the rest of the board, as well as the 12V pass-through connection for the magnetometer. An ADP3303-5 low drop-out (LDO) regulator is used to create the 5V board. LDO regulators can exhibit instability during operation, particularly if the output capacitance is mismatched, so the layout recommendations in the datasheet [73] were followed verbatim. Two LEDs also indicate the status of the voltage line: either will be off when the corresponding rail drops below 2 V.

The left side of Figure 10.34 shows how the different external devices are connected together and to the various parts of the companion board. Ferrites are used on the DAC output lines to quash possible high-frequency noise or oscillations. The indicator LEDs are directly connected to the corresponding DAC output lines. Unused DAC channels are kept floating. The power supply connectors route power, grounding, and the analog control lines for voltage and current. There is also a `SUPPLY_DISABLE` line, which is a feature of the 1685B power supply, which will inhibit output of the power supplies when this pin is grounded. The ADC is configured as eight pseudo-differential channels, and measures the three current sense lines relative to board ground, and takes the analog measurement channels from the magnetometer. An unused channel is used to input the 5V board power, which is used by the software as an OK check that the companion board is correctly providing power.

In Figure 10.35, we see the contents of one full current channel on the board. Many relevant design details have already been discussed in section 10.1, section 10.2, and section 10.3. `D11` is the channel's dropper diode. The used current sense amplifier in the INA2290 are supplied with 12V and decoupled using `C14`. The unused amplifier is terminated by tying its inputs together, and since it is a differential amplifier, it should output zero volts, which is then pulled down through `R12`.

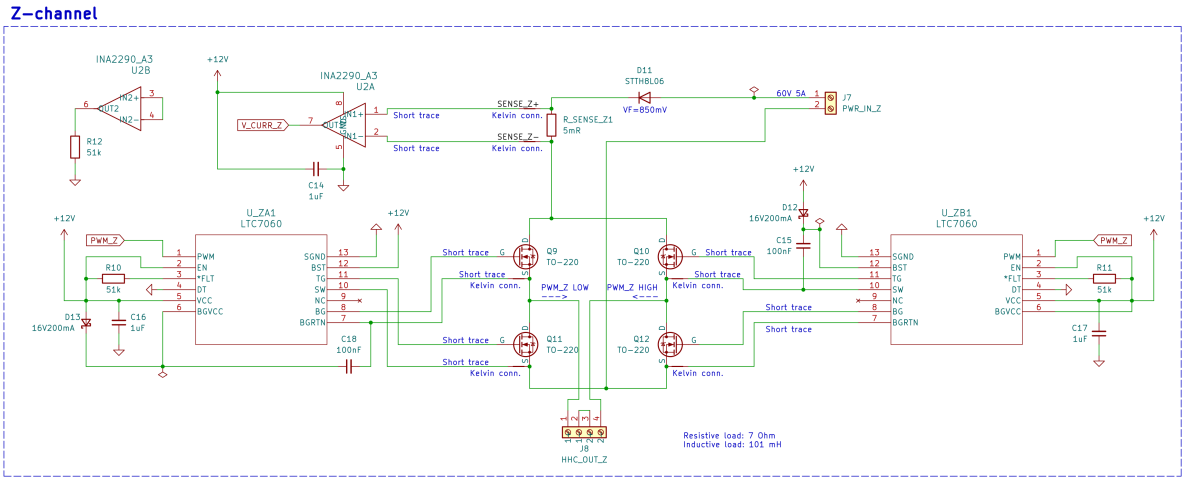


Figure 10.35: Schematic of a single current channel on the Helmholtz cage companion board v1.0.

The LTC7060 chips are configured much like specified in the datasheet [54]. Looking at the unit on the left: it is supplied with 12V from the board rail and decoupled by `C16`. Boost capacitor `C18` is charged through `D13` and provides gate charge to the MOSFETs. `R10` is a pull-up resistor for the fault flag pin. The `EN` is also pulled high to activate the device at all times. The `DT` pin is grounded, to provide the shortest possible dead time of 32 ns. The `SW` and `BGRTN` pins are connected to the MOSFET sources via Kelvin connection, something that can be seen more clearly in Figure 10.36. The four-terminal output block allows the user to connect one coil to pins `1+` and `1-`, and the other to `2+` and `2-`, which places the coils in series.

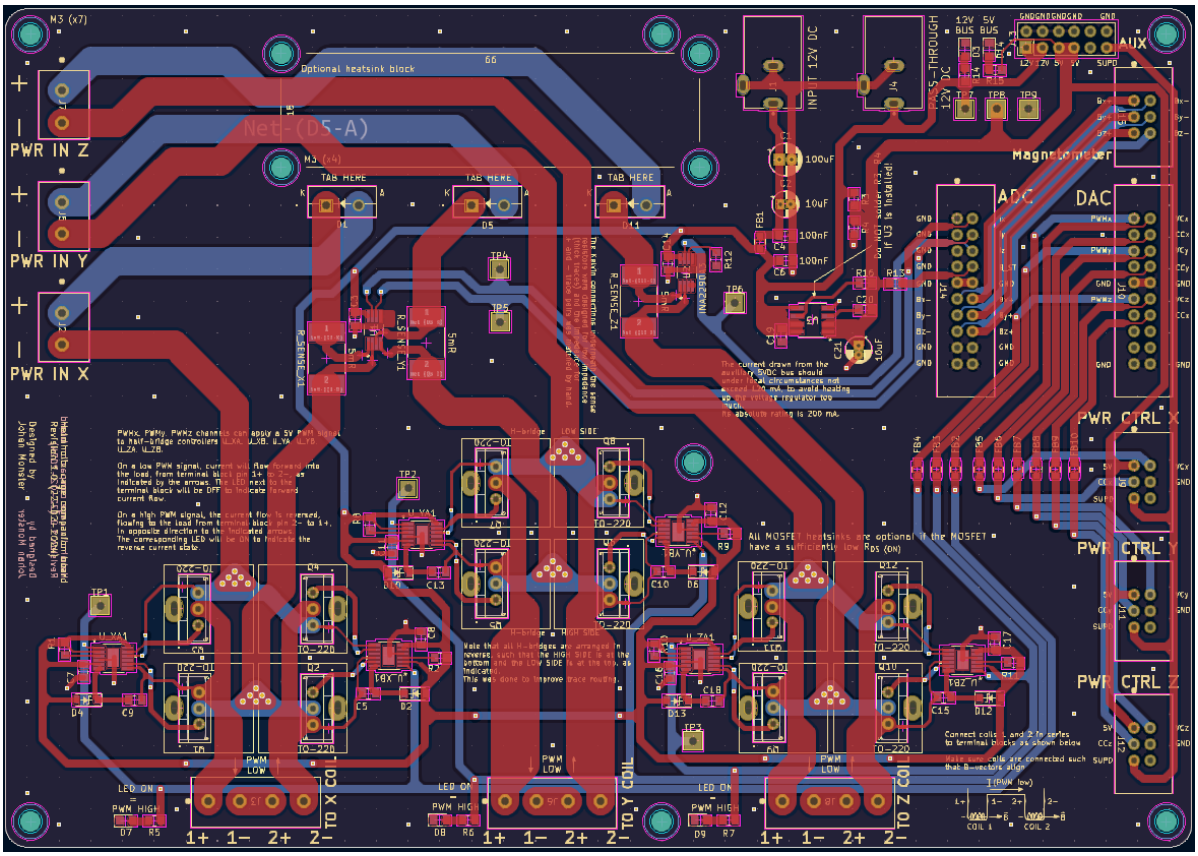


Figure 10.36: CAD visualization of the trace routing of companion board v1.0.

This leads to the trace layout as seen in Figure 10.36. The power supplies are connected on the top left, and the coils are connected along the bottom. The power distribution and connection routing are done on the right side of the board. Note the thicker traces used for the current channels, which should keep temperatures and internal resistances low. The board has seven (rather than four) mounting holes to prevent the board from sagging under its weight. There are also four mounting holes for a custom heatsink for the dropper diodes.

10.4.3. DAC as driver

One issue that had to be side-stepped in the design was the 40-pin header on the CN-0554. This board uses the standard 40-pin Raspberry Pi header to connect to the Raspberry Pi. This board is advertised with pictures in which it comes with this header pre-soldered, but with long stacking header pins. These pin allow one to connect the Raspberry Pi to the CN-0554 from the bottom using the female headers, and still have access to the individual pins from the top using the male header pins.

However, the two CN-0554 boards that were ordered did not have the long pins at the top, just the stubs of the solder connections (see Figure 10.37). This has the annoying effect that any free GPIO pins on the Raspberry Pi connection, as well as power and ground connections, are not explicitly exposed. One could solder some wires to the stubs, but this is a rather inelegant solution. On one of the CN-0554 boards, it was attempted to desolder the pre-soldered 40-pin header, but this involved a laborious process of sawing the header block into sections and desoldering each pin separately. These patient and careful efforts were rewarded with lifted and damaged solder pads regardless, and so it was decided that a different solution had to be found if the board was included in the design.

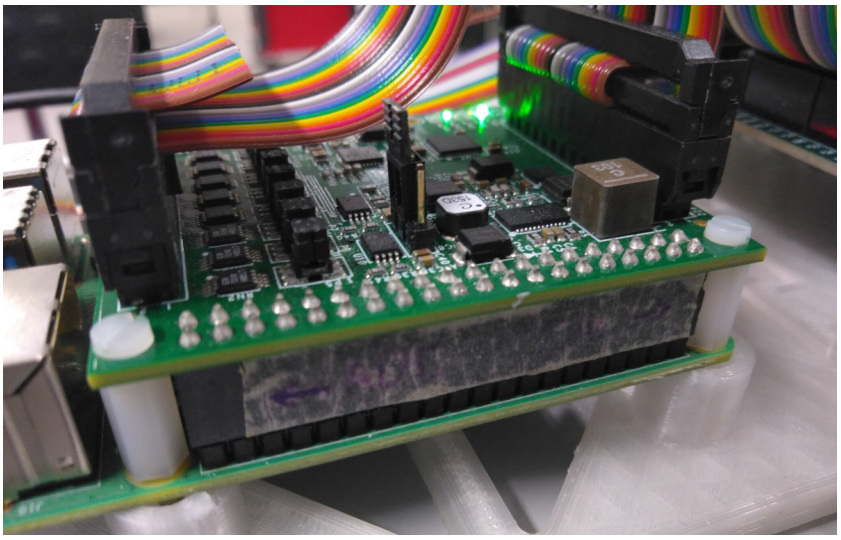


Figure 10.37: The CN-0554 board arrived with short header pins pre-soldered, rather than long pass-through header pins. This makes it more difficult to expose unused GPIO pins of the Raspberry Pi.

During the development process, an custom 40-pin riser block was made that allows one to "intercept" one or more GPIO pins to allow the Raspberry Pi to control the H-bridge pins, as the GPIO pins are generally made for this sort of task. However, in the end it was chosen to drive the H-bridge pins with three spare pins on the DAC.

10.5. Issues and Design Improvements

After v1.0 of the companion board was manufactured, its performance could be tested. A number of issues were found: issues with the board design, issues with the circuit design, improper use of the hardware, but also more fundamental issues with the system. The author considers the process of identifying and addressing these issues as one of the finest learning experiences of the thesis work, and as a result we will take some time in this section to discuss this.

Some of the larger issues were already discussed in preceding sections of this chapter, such as the

redesign of the current sensing and the addition of the charge reservoir networks to the H-bridge drivers. These will not be revisited in this section.

10.5.1. Issue: DAC incapable of driving LTC7060 PWM input

The LTC7060 bridge driver chips that were used to activate the H-bridge MOSFETs have another quirk, in that the internal input impedance of the `PWM` input is not very high. This can be seen in the bottom left corner of Figure 10.38a. The LTC7060 chip will switch the bridge depending on the two Schmitt-trigger comparators, so to drive it low, the `PWM` pin has to be pulled below 1.0 V, and to drive it high, it must be pulled above 3.1 V. There is a 42k resistor to ground, meaning that whatever circuit the `PWM` pin is driven by has to deliver enough current to ensure a high enough voltage on the pin to trigger the input comparator. In the context of the companion board, this is even harder: there are two LTC7060 `PWM` pins that are driven in parallel by the same line, as well as an indicator LED with a 4.7 kΩ resistor. The equivalent network of this can be seen in Figure 10.38b.

During testing of v1.0 of the companion board it was found that the H-bridges were not reliably switching, and this input impedance issue was identified as one of the possible root causes. Remember that the `PWM` pins were being driven by DAC pins, which is generally not something a DAC is used for. However, the issue with the CN-0554 headers discussed in subsection 10.4.3 made this a necessity.

To address the issue, the control signal from the DAC was buffered by a non-inverting buffer. There is no explicit need for this buffer to be a Schmitt-trigger type for overcoming line noise, as the LTC7060 already has Schmitt-trigger comparators at its `PWM` input. However, it does not hurt, and will ensure the indicator LED is at a fixed brightness when on.

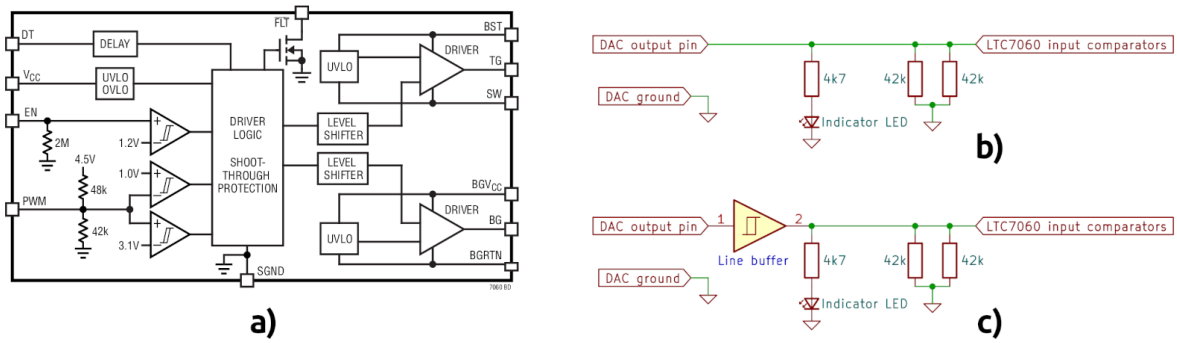


Figure 10.38: Low input impedance of the LTC7060 PWM pin and LED indicator creating problems for the H-bridge control. **a)** Internal block diagram of the LTC7060 [54] **b)** equivalent circuit of the input impedance that the control pin has to overcome **c)** buffering the control signal.

This implementation was prototyped by desoldering the ferrites on the `PWM` lines from the DAC, and using the pads to as solder points for bodging wires. A CD40106 on a Direct In-line Package (DIP) chip was used to breadboard the implementation (see Figure 10.16 and Figure 10.18), since these chips were on hand in the workshop. However, on the final iteration of the board, a small non-inverting buffer from the 74-series (SN74LVC1G17) was used instead, to allow for a compact board layout with the buffers positioned close to the bridge driver chips.

It was also tested whether the H-bridges would switch more reliably if they were driven by a few spare Raspberry Pi GPIO pins, rather than the DAC pins on the CN-0554. However, this does involve implementing a way to expose these GPIO pins and then running some spare wires over to the board. With the line buffer in place, no difference was observed and so the DAC outputs were used in the final design. However, there is still provision to switch to GPIO pins by using a user-configurable solder jumper on the board. This is low-cost solution to the problem that still allows some customisability later on if needed, in accordance with **TOP-08-01** and **TOP-10-01**.

10.5.2. Issue: 1685B incapable of driving inductive load

At a fairly late stage of development, it was found that the BK Precision 1685B power supplies, intended to drive the coil pairs of the Helmholtz cage, were completely unusable for this task. This issue was

allowed to go unnoticed for a long time, because much of the development and testing were done with dummy loads, such as high-wattage resistors and incandescent bulbs. These loads have negligible inductance, but the intended load has an inductance in the order of 100 mH.

When it was attempted to control the coil pairs with the 1685B power supplies, the internal current control circuitry would not converge to a stable current, but instead oscillate. This can be seen in Figure 10.39, where the left section displays the 1685B successfully tracking a 5 A square wave, albeit with some slew, going into a purely resistive 230 mΩ load. The right part of Figure 10.39 is from a different test, and shows the output voltage oscillating unacceptably when trying to drive a constant 1 A through a Helmholtz cage coil pair.

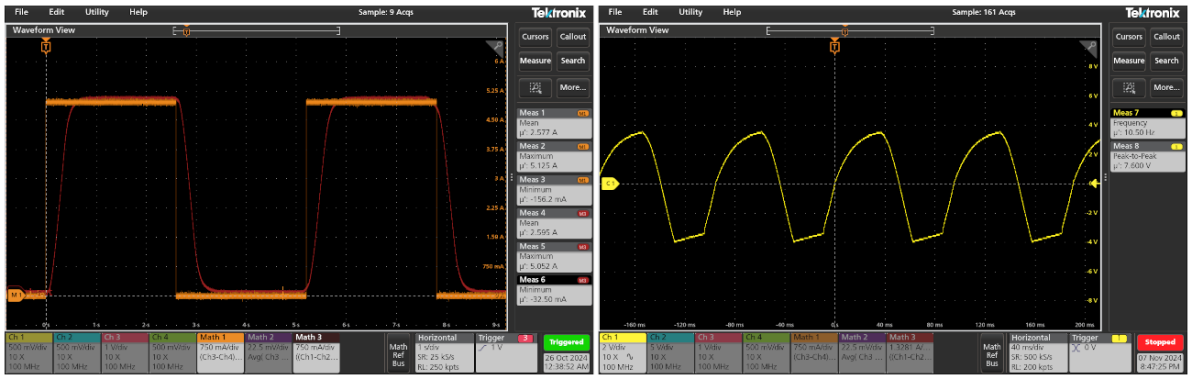


Figure 10.39: A fundamental flaw of the 1685B power supplies is revealed: it cannot drive the inductive load posed by the Helmholtz cage coils. **On the left**: Driving a non-inductive load with a 5 A peak-to-peak square wave, which the supply tracks without problems, despite a substantial slew rate. The orange trace is intended current, the red trace is output current. **On the right**: Output **voltage** of the power supplies when trying to drive a constant 1 A through the inductive load of a Helmholtz coil pair. The 1685B output voltage oscillates with a 7-10 V peak-to-peak at a rate of about 10 Hz, fundamentally disqualifying the 1685B for this application.

This posed a significant problem, as this issue made the three acquired 1685B supplies effectively unusable for their intended purpose. This revelation came at a time where there was no time left to procure a different trio of power supplies. No information in the datasheet of these power supplies revealed any information that this problem could have been foreseen, although details about driving inductive loads are also suspiciously absent in the datasheet.

The old ES-030-10 power supplies by Delta Elektronik were now suddenly looked upon in a new, very favorable light. After all, these power supplies had already proven themselves capable of driving the Helmholtz cage coils. And indeed, when the same test from Figure 10.39 was repeated with the ES-030-10 supplies, the current control would converge without problem, exhibiting a few dozen mV of ripple.

There are a few problems with this approach. First of all, it will mean that some of the system requirements are no longer achievable. The affected requirements are:

- **ELEC-01-01** cannot be achieved, due to the reasons discussed in section 6.2. The coil current achievable will be in the neighbourhood of 4 A instead of 5 A, and is limited by the 30 V voltage limit of the power supply. **PER-02-01** and **PER-03-01** should still be achievable, although with less margin.
- For similar reasons, **ELEC-04-01** becomes unachievable.
- Requirement **ELEC-12-01** is also in danger, because the Delta Elektronik power supply that were previously used in the Helmholtz cage have a particular factory option that removes the default rear connector with the analog control input, and instead replaces them with an RS-232 interface.

After opening up an ES-030-10 unit, it was found out that the power supply could be modified to re-expose the analog input pins, although at the cost of removing the RS-232 interface. Since the RS-232 interface was not needed, this was not much of a downside. The modification is relatively straightforward, reversible, and makes the power supplies fully compatible with the existing companion

board and control software. Detailed instructions on how to perform this power supply modification can be found in Appendix E.

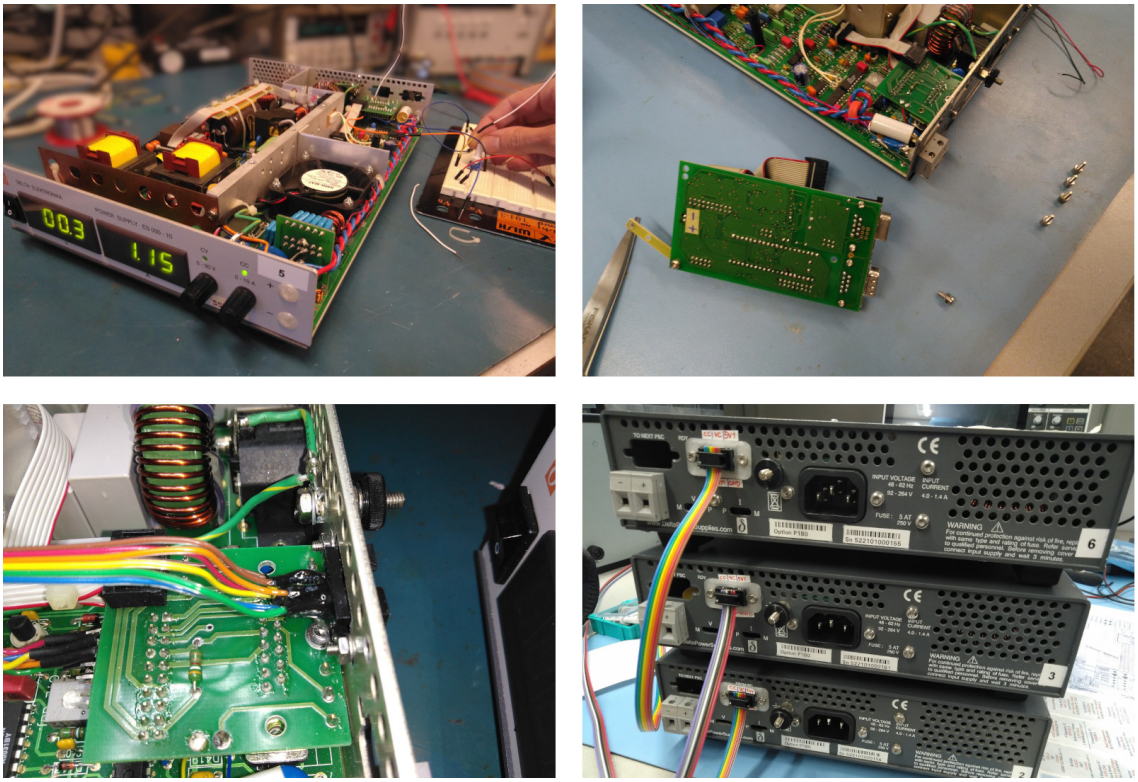


Figure 10.40: Modifying the ES-030-10 power supplies by removing the factory-installed RS-232 interface card and instead exposing six analog interface pins: Four for current and voltage control and monitoring, one for a ground reference, and one for internal reference voltage. More details about this modification can be found in Appendix E

The power supply switch did prompt a redesign of the power supply unit (PSU) control layout on the companion board. There would now be six pins: One for any exposed 5 V internal power in the power supply, and another internal ground connection. Two more pins would be designated for analog current control and analog voltage control. The final two pins could be connected to current and voltage sense lines, if these were available on the power supply. The latter would be disconnected by default, and the current sensing on the companion board would be used. However, using a user-configurable jumper option these can be routed to the relevant ADC pins instead.

The functionality to disable the output to the Helmholtz cage coils was moved away from the power supply, and instead moved to the LTC7060 for a more generalized implementation. This is described in subsection 10.5.3. The `SUPPLY_DISABLE` pin that was used previously is therefore no longer needed.

10.5.3. Generalized output inhibition

An additional feature that was planned for the second iteration of the companion board was the ability to completely inhibit power to the Helmholtz cage on command. In fact, it would be preferable for the board outputs to be inhibited by default, only turning on when necessary. The plan was to inhibit the power supplies by default unless the following two conditions are met:

1. **The 5V rail of the companion board is active.** The voltage of the 5 V rail is used as a proxy to determine whether the power supplies are in fact being actively controlled. If the 5 V is inactive, control over the power supplies cannot be guaranteed, and so it makes sense to disable the power supplies based on this condition.
2. **The output inhibition is explicitly lifted by a software-controlled `ENABLE` pin.** For reasons discussed in subsection 10.4.3, this signal will come from a channel on the DAC.

In line with **ELEC-13-01**, an red indicator LED should also be implemented to warn the user when the outputs to the coils are "live".

Before discovering that the 1685B power supplies were unsuitable, the plan was to implement this functionality by routing the `SUPPLY_DISABLE` pins on the 1685B to the board and connecting them to ground to disable the power supply outputs. This can be seen on the left side of Figure 10.41, where a MOSFET is used as a voltage-controlled switch to ground. The MOSFET is opened so long as at least one power supply is connected to the board, using a diode-based OR-gate. The power supply inhibition is lifted by closing the MOSFET, which is achieved by applying some current to the base of the bipolar transistor, which will then become conductive, redirecting any current coming through the 10k resistor to ground. This closes the MOSFET, as its gate will be approximately at ground level. This solution was prototyped with a spare IPA045 MOSFET and a garden-variety BC547 transistor and worked just fine.

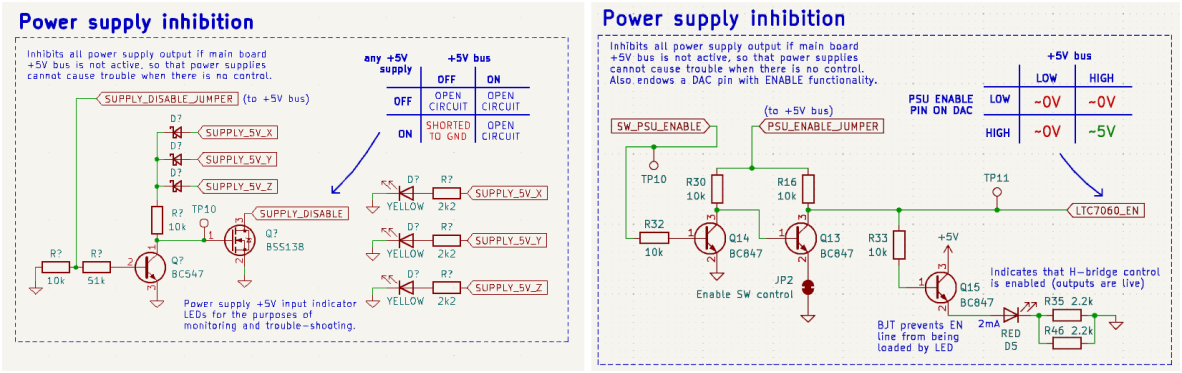


Figure 10.41: Two possible circuits for implementing a two-condition inhibition scheme. The circuit on the right was incorporated in the companion board.

This implementation has the advantage that the inhibition is done at the power supply, meaning that current cannot even flow to the companion board in the inhibited state. However, it has the distinct disadvantage that it relies on some implemented functionality of the 1685B power supply, which does not make it a very good general implementation. If someone were to implement the system with a different power supply without an inhibition pin, this functionality would not work. This became rather pertinent when it was found out that the 1685B could not be used as Helmholtz coil drivers, for reasons discussed in subsection 10.5.2. The ES-030-10 power supplies had to be used instead, which do not feature an inhibition pin with similar properties.

So, to make the "inhibition functionality" agnostic to whichever specific power supply is used, it had to be implemented on the companion board instead. There are two straightforward ways to achieve this:

1. **Cut off power to the six charge reservoir network powering the MOSFET drivers.** This can be implemented quite easily with the same circuit as the left one in Figure 10.41, but instead connecting the MOSFET between the 5V rail and the six SN6501 transformer drivers. Closing the MOSFET will then rob the LTC7060 MOSFET drivers of the source of charge they need to activate the H-bridge MOSFETs, and as a result none will be able to activate, preventing any current flow to and from the Helmholtz coils.
2. **Drive all MOSFETs low using the `EN` pin of the LTC7060.** The datasheet of the LTC7060 contains a truth table indicating that all MOSFETs can be disabled if the *EN* pin is pulled low. In v1.0 of the companion board, it was pulled up by default. There are many ways to implementing this, but the RTL circuit on the right side of Figure 10.41 is one straight-forward solution.

On the second version of the companion board, the second solution was implemented, using the enable pins of the LTC7060. From Figure 10.38a can be seen that the `EN` has an internal 2 MΩ pull-down resistor, and so six LTC7060 chips in parallel will present an input impedance of 333 kΩ, for which the required current can be easily supplied by the circuit in Figure 10.41. To ensure the indicator LED does not pull down the signal, it is buffered by a bipolar transistor as a matter of course. The other two bipolar

transistors essentially form a cascaded RTL AND-gate. Either conditions can be disabled on the board if this is required:

- The condition that the software-controlled pin must be high can be disabled by cutting PCB jumper.
- The condition that the 5V rail on the board must be active can be bypassed by soldering `TP10` and `TP11` together, which on the board are arranged next to each other, so that they can be bridged by a 0 Ω resistor or a blob of solder. If this is done, the RTL stage is bypassed completely and the inhibition functionality is directly driven by the DAC, although the LED indicator will not light up much, if at all.

This implementation was once again prototyped by modifying the existing companion board and putting the circuitry together on a breadboard.

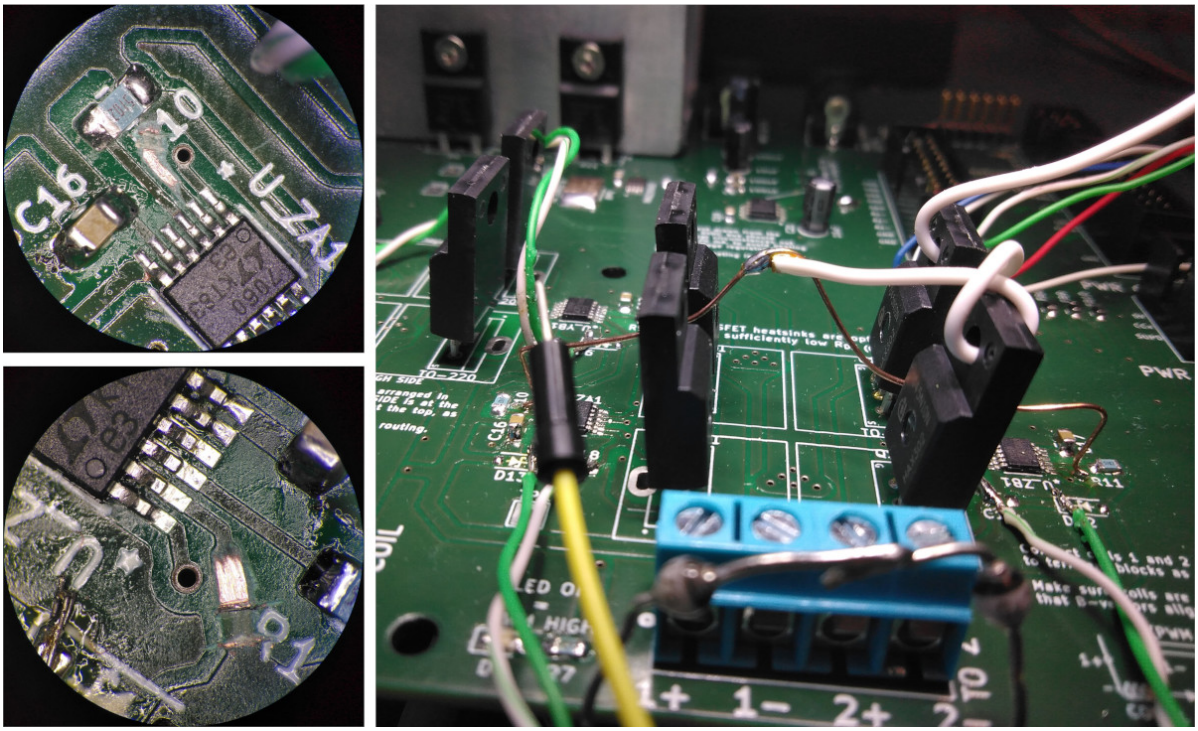


Figure 10.42: Testing the implementation of Figure 10.41 by disconnecting the `EN` on the LTC7060 from the internal supply rail, and to a common breadboard-compatible jumper pin. This involved exposing, cutting (left) and soldering together traces (right), one of many occasions during the developments where such bodges were implemented.

10.6. Companion Board - Second Iteration

It is now time to discuss the second implementation of the companion board. This section will follow a similar format as section 10.4, with a detailed walk-through of the circuit, explanation of certain design choices, and how these relate to the requirements and previous iteration.

10.6.1. Functionalities of the companion board

We start this section again by looking at the list of functions implemented by the board, as this list has grown significantly. The Helmholtz cage companion board implements the following functionalities:

1. Three polarity inversion channels, each of which take a unipolar output from a constant current supply, and returns it either straight or with its polarity inverted. A single high/low control signal should engage/disengage the polarity inversion. This implements requirement **ELEC-05-01**.
2. The output to the coils is a four-connection output, so that two coils can be connected in series without having to separately connect the cables itself. This will make it easier to maintain the system, especially if it ever has to be moved.

3. Each channel has implement a form of current sensing (see ELEC-06-01), with a filtered output. There is additionally an option to route voltage and current sense lines from the power supplies, if these are available, to the ADC instead. These connections can be configured on-the-fly via pin jumpers.
4. Each channel features a voltage drop of at least 0.5 V to ensure that the current supply is operated in its linear domain.
5. Each channel has sufficient transient suppression capability to ensure counter-electromotive forces from the inductive loads will cause no problems, in accordance with TOP-06-01, ELEC-07-01, ELEC-08-01, and ELEC-14-01. This is implemented using TVS diodes, the body diode properties of the H-bridge MOSFETs, and the dropper diodes.
6. There is a hardware-level implementation (ELEC-14-01) for inhibition the flow of current through the coils (TOP-06-01).

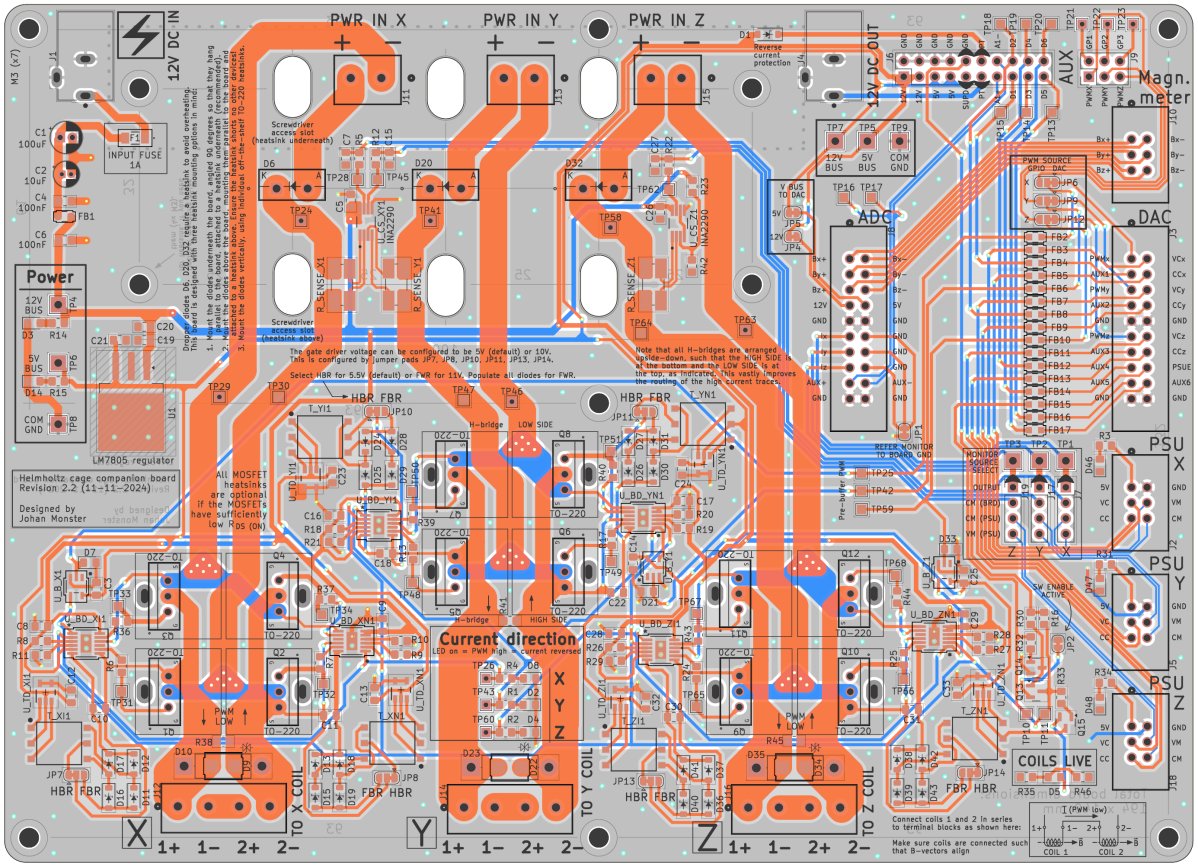


Figure 10.43: Complete overview of the trace layout of the companion board v2.2. A higher resolution version can be found in Appendix D.

7. Each current channel features a bleeder resistor that drains the magnetic energy from the coils in a controlled manner, even if it is completely isolated by the H-bridge circuits. This relates to requirements TOP-06-01, ELEC-07-01, ELEC-08-01, and ELEC-14-01.
8. The board has regulated power rails at 12VDC and 5VDC to power on-board components, as well as external peripherals, such as optional cooling fans.
9. The 12 VDC input is protected from overcurrent / short-circuit events by a 1 A input fuse. As the magnetometer can be connected to the 12 VDC pass-through output connection, it will also be protected by the fuse. The 12 VDC pass-through output for the magnetometer is equipped with reverse current protection, so that accidental insertion of the 12VDC input jack will not cause any current to flow, and the rail LEDs will not come on. To the same end, the input and pass-through

output barrel jacks have also been physically separated, and the 12VDC input is marked by a large silkscreen symbol. These measures are taken to satisfy **TOP-06-01**, **ELEC-13-01** and **ELEC-14-01**,

10. The board functions as a central interconnect between other devices of the system, so that cable management remains straightforward and cable harnesses or looms are simple. The devices that the companion board interconnects are:
    - 12VDC power from mains via barrel jack socket adapter
    - AlphaLab magnetometer (power and data)
    - ADC and DAC connections from the CD0554 board
    - Common grounding with the Raspberry Pi/CN-0554 boards
    - Constant current power supplies (3x control signal and grounding)
11. In line with **TOP-06-01** and **ELEC-13-01**, the board should have visual indicators to help indicate the internal state of the board:
    - Status of 12V rail (amber LED)
    - Status of 5V rail (amber LED)
    - H-bridge status of X, Y, Z channels, colour coded with red/green/blue LEDs
    - Status of the internal control voltage of the power supplies (amber LED)
    - "COILS LIVE" indicator to show that current can flow through the Helmholtz coils (bright red LED)
12. Also as part of **ELEC-13-01** and to help during development, the board features test points where this is deemed useful. The number of test points was greatly expanded compared to v1.0.
13. Additionally, to align with **TOP-06-01**, **ELEC-13-01** and **SW-12-01**, clear silkscreen indication are implemented as much as can be reasonably expected. The user should not have to (for they will likely not) consult any manuals to find:
    - The location of the current channel input and outputs.
    - The default direction of current flow through the H-bridges.
    - The meaning of the LED indicators placed on the board.
    - The function or destination of any header pin connections.
    - The location, designation, and orientation of any board components, including through-hole components such as MOSFETs, diodes, and bipolar capacitors.
    - The version of the board.
    - The physical dimensions of the board and hole spacing.
    - The location and current rating of the fuse.
14. The board layout facilitate cooling of the three dropper diodes, and optional cooling for the twelve H-bridge MOSFETs. There are three possible mounting configurations for the dropper diodes to facilitate different cooling options, as well as auxiliary mounting holes and screwdriver access ports.
15. The board contains insulated mounting points that are close enough together so that the board will not sag under its own weight.
16. Any components connected to the current-carrying channels that significantly protrude from the board use electrically insulated packages, to reduce the chances of accidental damage or injury (**TOP-06-01**).
17. The board supports several user-configurable options using jumpers where this is sensible. A number of anticipated board modifications are also supported. These changes were made in line with **TOP-10-01**, **TOP-11-01**, and **ELEC-14-01**.
18. Any traces with significant current should have their impedance minimized to avoid excess heating of the board and reduce losses.

19. At least one layer should be a ground plane in order to improve trace routing and act as a thermal sink.
20. Any board layout procedures that will reduce the fabrication cost of the board should be implemented, such as sticking to default parameters for e.g. layer thicknesses, keeping the board area relatively small, etc. This is in line with TOP-08-01

10.6.2. Circuit Overview

With the list of functionalities on the board out of the way, let us dig into the specifics of the schematic. The discussion will visit smaller portions of the schematic, but the full schematic can be found in Appendix D.

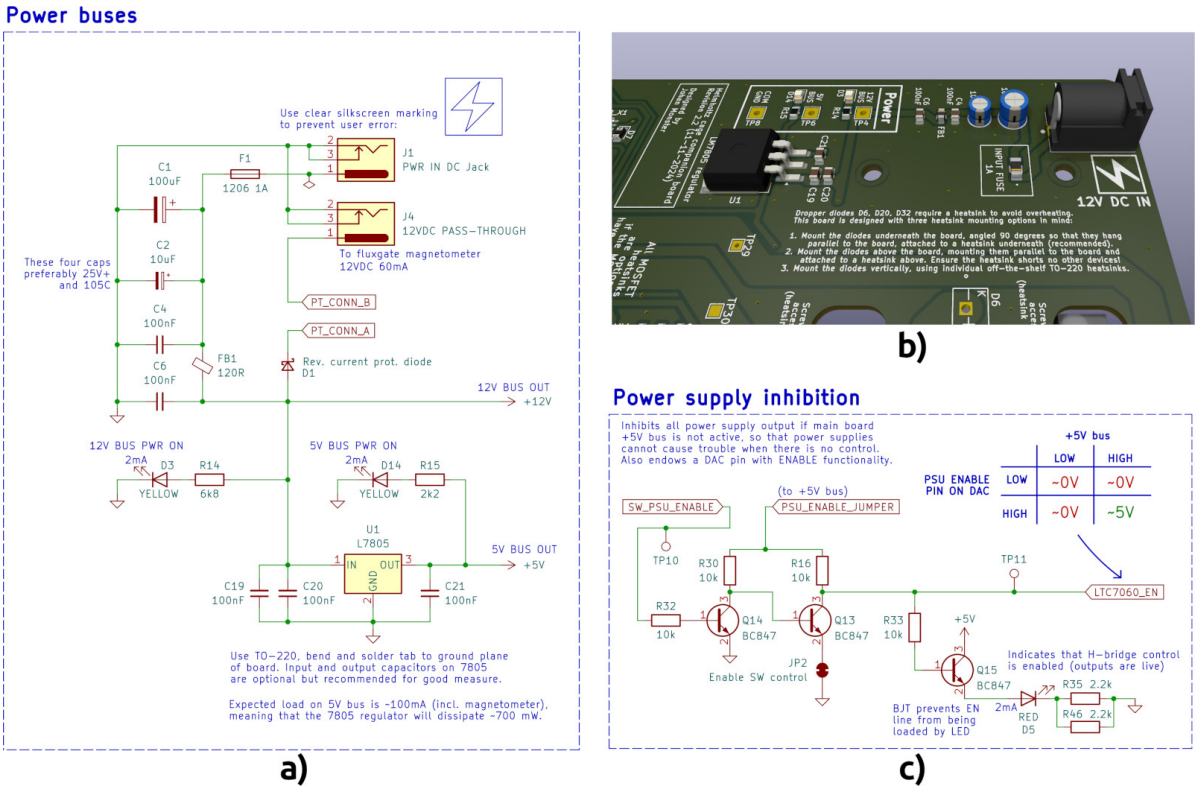


Figure 10.44: Power input section of the Helmholtz cage companion board v2.2 as **a)** schematic **b)** 3D render. **c)** is the current inhibition scheme discussed in subsection 10.5.3.

Starting off, Figure 10.44a shows the power input section of the board. Compared to the last iteration of the companion board, it is now fused at the input with a 1 *A* SMD fuse. In the case of a overcurrent or short-circuit event the fuse will blow, hopefully protecting other components on the board from damage. The value of 1 *A* is fairly arbitrary and was chosen as a typical value that is commonly available as a COTS component. There is no explicit requirement that the fuse should be of a fast blow or slow blow type, so either will do. Also, there is now a Schottky diode that prevents current from flowing out of the male pin of the `PASS-THROUGH` barrel jack connector. If anyone accidentally plugs a 12 VDC adapter into this output it will have no effect, and to signal this the board LEDs will not come on (**ELEC-13-01**)

Furthermore, it is clear from Figure 10.44a that the previous ADP3303-5 LDO regulator has been replaced by a large LM7805 regulator in a TO-263 package [7]. There are several reasons for this choice:

1. The 7805 regulator is an extremely common linear voltage regulator, which means that it should never be out of stock, and should be a very future-proof pick (**TOP-11-01**). It is also overkill for the current power demands of the board, which means there is room for future upgrades (**TOP-10-01**).

[7] Also commonly known as D2PAK

2. The TO-263 can dissipate a large amount of power directly into the board's double ground plane, which means no separate heatsink is needed and thermals should not be an issue.
3. The 7805 is cheaper than the previously used ADP3303-5 (**TOP-08-01**).
4. Since the regulator is dropping 12 V to 5 V, it does not use any of the advantages of a low dropout regulator, but it will still exhibit the instability problems of LDO regulators in general, as well as the extra components needed to address these. As a result, a 7805 is a much better pick overall. Compared to the previous board, the number of items on the BOM drops from five to two. The three 100 nF decoupling capacitors can even be left out in good conscience, given how close the regulator is to the other input filtering capacitors of the board [8].

Figure 10.44b shows a 3D render of the power input section of the board. A large silkscreen symbol indicates that the barrel jack beside it is the 12 VDC board input. Finally, Figure 10.44c shows the current inhibition implementation, already discussed in subsection 10.5.3.

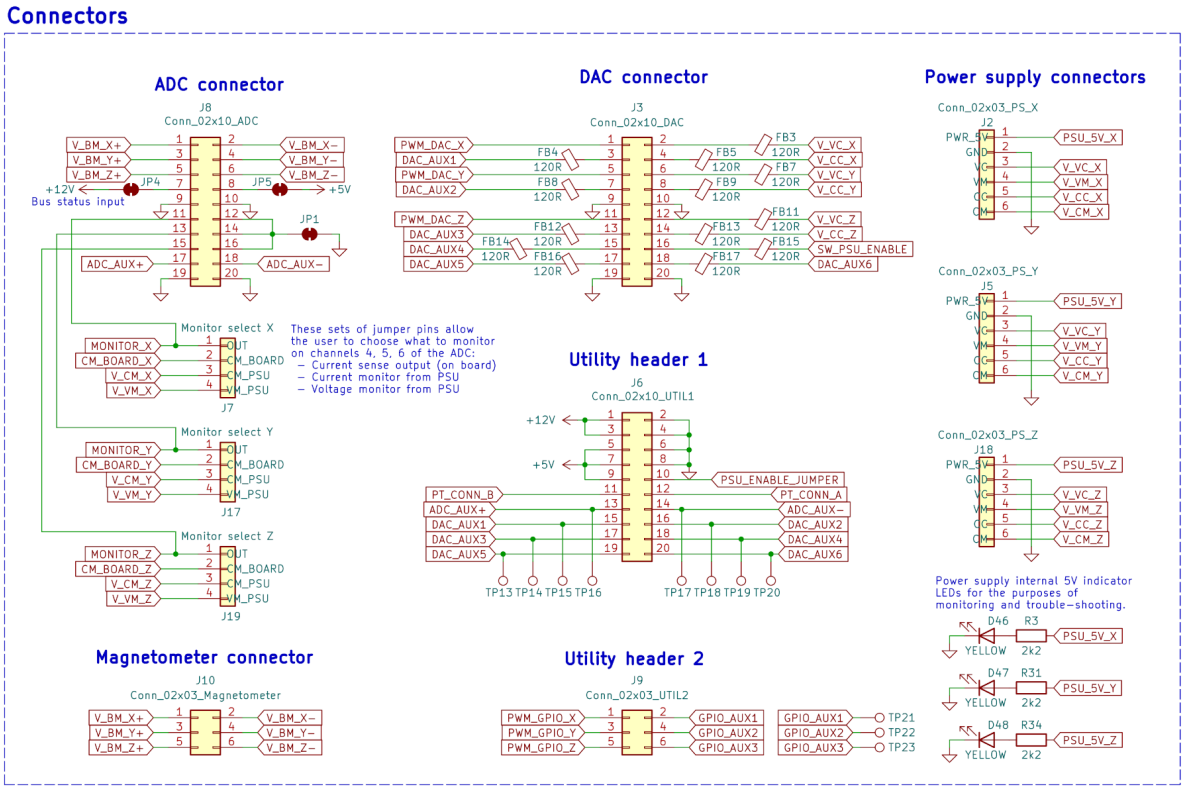


Figure 10.45: Routing of board connections for the Helmholtz cage companion board v2.2.

Moving on, Figure 10.45 shows most of the connector routing for the companion board. Several things have changed significantly:

- Pin 16 on the DAC connector is now the `SW_PSU_ENABLE` pin, and needs to be driven high by the software to lift the output inhibition on the board. The six unused DAC channels are exposed to the pins of Utility Header 1, so that users may use them to expand the board functionality with minimal modification to the board the board (**TOP-10-01**).
- All DAC channels have an input ferrite. This is dual purpose: it helps reject high-frequency noise, although the need for this is not pressing. However, the ferrite can also be desoldered to provide some soldering pads for bodging extra wires, making the board easier to modify (**TOP-10-01**).
- Both the +12V and +5V rails on the board go into a single differential channel, so that the software has a way to monitor the board rails via the ADC. If the board has no power, it will read 0 V. Under

[8] In other words, a fine application of the KISS principle.

nominal conditions, it should read 7 V. If the 5 V regulator malfunctions, it should read 12 V.

- Channels 4, 5, 6 on the ADC are now known as "monitor" channels. The source of what they monitor can be selected via some pin jumpers, with the available sources being the internal board current sensing, PSU current sensing, or PSU voltage sensing. The first option is recommended for general use. The measurements are in reference to the board ground plane via `JP1`, but this can be disconnected and modified if needed (**TOP-10-01**).
- The single unused differential ADC channel is also exposed on Utility Header 1.
- The modifications to the PSU connectors have already been discussed, but should be a better match for a large variety of possible power supply models. When 5V is supplied to the first pin, an indicator LED displays this on the board (**ELEC-12-01**).
- Control over the H-bridges is done by the DAC by default, as explained in subsection 10.4.3, but the door to using GPIO pins is not fully closed. Utility Header 2 features six GPIO inputs, three of which can be routed to the LTC7060 `PWM` input. This is disabled by default, but can be configured by solder jumper (also see Figure 10.46). The other three GPIO pins are auxiliary, and are routed to large test pads on the board that can be used as oscilloscope probe anchors or solder pads for bodging wires (**TOP-10-01**).

In summary, the routing options on the new companion board have been consolidated, and its potential for modification has been vastly expanded. It was mostly because the author himself struggled with a lack of these features during the development and debugging of the design that these features were implemented.

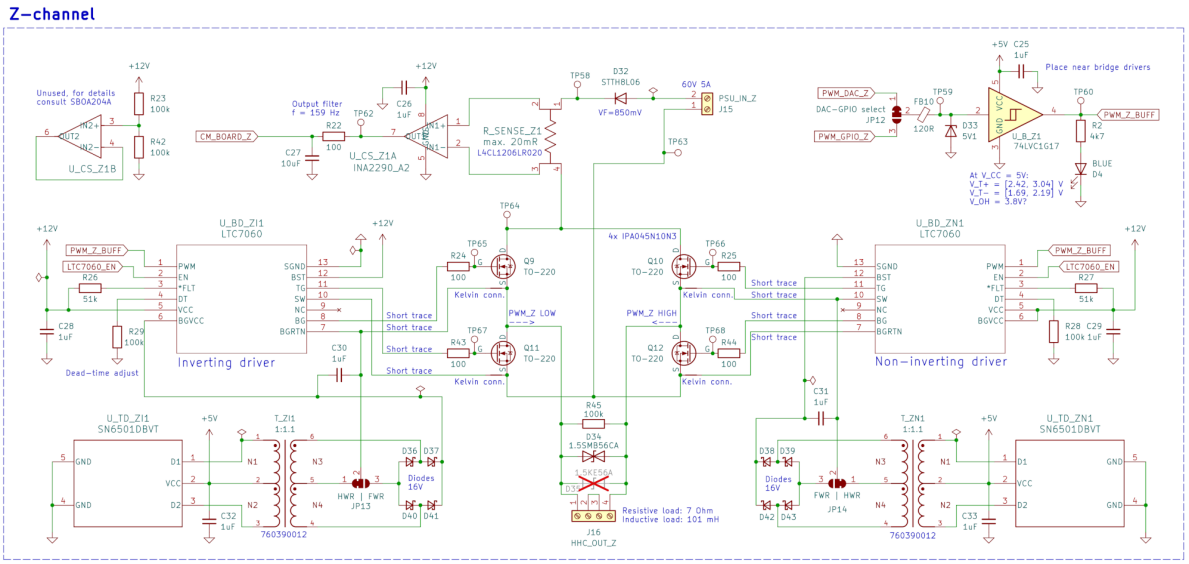


Figure 10.46: Power input section of the Helmholtz cage companion board v2.2 as **a)** schematic **b)** 3D render. **c)** is the current inhibition scheme discussed in subsection 10.5.3.

Finally Figure 10.46 shows the schematic of a single current channel.There should be few surprises here, given how most details have been discussed in previous channels. A few points of attention:

- The gate of each MOSFET has been equipped with a 100 Ω gate stopper resistor that should help reject high-frequency oscillatory behaviour of the MOSFET [55].
- A single-pole RC filter has been put on the output of the INA290 sense amplifiers to smooth out the output data. The cutoff of the filter can likely be further reduced for increased smoothing, but preferably by increasing the size of `C27`, rather than increasing the resistance of `R22`.
- The way the unused INA290 amplifier is terminated has changed [63].
- The source of the `PWM` signal to the bridge drivers can be configured by `JP12` as coming from the DAC (default) or the GPIO pin. Since GPIO pins can be sensitive to overvoltage and undervoltage, a 5.1 zener diode (`D33`) is used to protect it.

- The voltage output to the charge reservoir network can be configured with JP13. By using the half-wave rectifier (HWR) option, the voltage should be approximately 5 V. Using the full-wave rectifier (FWR) option outputs 10 V instead. The latter should be a bit better, as it will ensure that the MOSFET has a lower drain-source resistance. However, as this could not be tested before the end of the design iteration, the 5 V option is configured on JP13 by default.
- There are many more test points, which is something that the author dearly wishes was implemented on the first board iteration.

10.7. Conclusion

This concludes the discussion about the electronic design of the Demonstrator System and the companion board. The first iteration of the companion board revealed many problems and shortcoming, but after the second iteration things are looking much better. How well the board holds up to all the formal requirements for the DS will be discussed in chapter 13. There are many more details about the companion board that are too detailed to be discussed here. For example, if you are interested in more details about the companion board, its configuration, supported mods, or assembly process, Appendix D will provide much of that. This appendix can be considered a blueprint for the companion board. Even though its design has been surpassed, more details about version 1.0 of the companion board can be found in Appendix C.

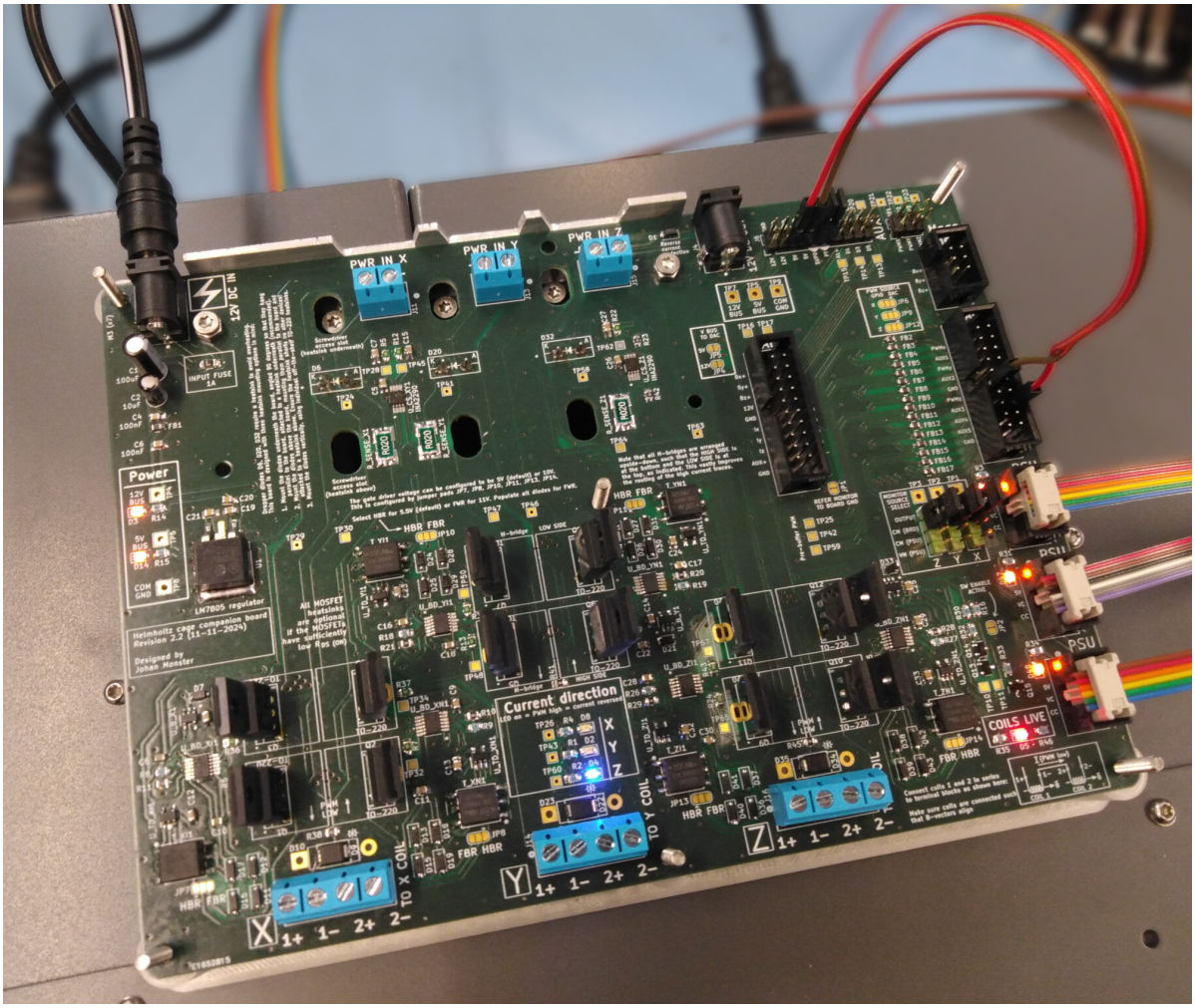


Figure 10.47: Assembled V2.2 companion board receiving power and being tested.

11

Software Development

One would be hard-pressed to find an embedded system for which software does not play a key role. Embedded computation can nowadays be facilitated at a great variety of scale, from humble 8-bit microcontrollers all the way to large processors. This means that embedded systems can be designed with software that handles many functionalities that were previously only possible by electronic or mechanical means. The Demonstrator System (DS) discussed before is no exception, and its software forms a critical aspect of it.

This chapter describes the development of the **Helmholtz Cage Toolkit (HCT)**, a Python-based package that bundles the necessary software and tools for the DS together. The source code for this software can be found in the `Helmholtz-Cage-Toolkit` repository [74]. Other software described in this thesis, such as the data processing software for the ambient magnetic measurements in chapter 7, are not included in the HCT, and will not be discussed here. Instead, the source code for these tools can be found at the `Gaussmeter-Utilities` repository [75].

11.1. Role of Software

To explain how the software for the DS was developed, we start at the overall system description from chapter 9, as well as the requirements related to software development from subsection 8.1.4. Due to the intentional separation between user interface and firmware as discussed in section 9.2, the software is can broadly be grouped into four sections:

1. User software for the **high-level interface** (desktop computer)
2. System firmware of the **low-level interface** (Raspberry Pi)
3. Communication **protocol for communication** between the two
4. Extra utilities and development tools

Back in chapter 9, we discussed how users interact with the system via a few degrees of separation (see Figure 9.2). We can place the software inside the Helmholtz Cage Toolkit inside this framework; in Figure 11.1 we see how the HCT connects and controls both the low-level and the high-level interface. This separation of functionality reflects requirement **SW-01**.

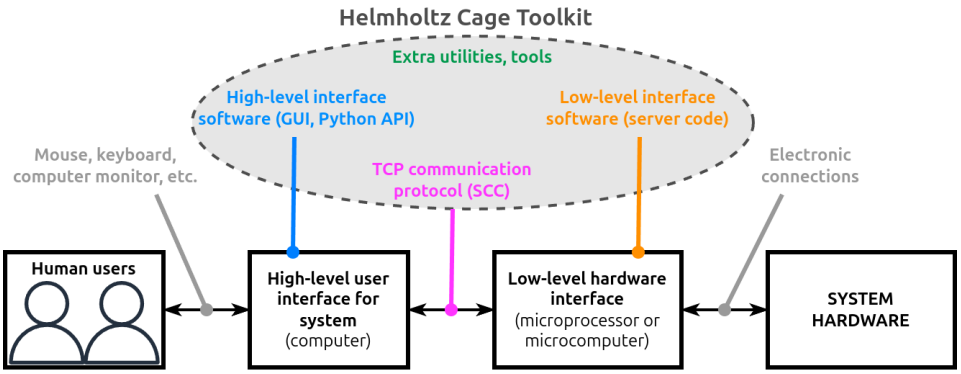


Figure 11.1: Implemented degrees of separation between system user and system hardware, and the contents of the **Helmholtz Cage Toolkit** software package in relation to this.

The remaining sections of this chapter will discuss the software in accordance with the four aforementioned groups. However, let us ask a more general question first: why was Python chosen as the language of choice? Python is not generally known for being a language that excels at low-level/bare metal applications, or for high execution speed. There are several reasons:

1. Python is supported on all major computer operating systems, as well as microprocessors such as Raspberry Pi (see EXTRA-10). For a number of years, it has also had a micro-controller compatible analogue in MicroPython.
2. Python is free and developing in Python is virtually devoid of any additional cost. (see EXTRA-01)
3. Python code can be easily bundled and distributed via the PyPI/pip system, GitHub, etc. (SW-05, EXTRA-05)
4. Python source code is relatively straightforward to read, which helps with documentation efforts (SW-12, EXTRA-05, EXTRA-09)
5. It is currently the most popular programming language [76], meaning that anyone interested in developing or modifying the HCT has a good chance to already possess the programming skills to start right away (EXTRA-05)
6. The author of this thesis had much more in-depth experience with Python as opposed to any language, making it an obvious choice from a development point of view. Some code developed beforehand could also be repurposed for this project; for example, some of the PyQtGraph visualization code used in the `Gaussmeter-Utilities` repository could be repurposed straight away.

Taking into account the considerations above, it seemed quite feasible from the beginning to package the full code of the HCT in a single Python package. Where speed is critical, it is important to rely on the right Python libraries (which frequently are written in C or some other fast compiled language), and use benchmarking to evaluate the performance. The door was left open to create C or C++ implementations where speed was critical and a particular Python implementation were to prevent the system from fulfilling its requirements, but at the time of writing, this has not yet been necessary.

11.2. High-level interface software

In this section we will have a closer look at the software of the high-level interface. By the layout outlined in Figure 11.1, this is the software that runs on the computer that the user directly interacts with. This software was not developed in a single development cycle, but started roughly at the same time as the design process of the system as a whole and the electronics. This lead to a series of iterative designs, each of which kept expanding and improving upon the existing infrastructure. Additional software features became available for implementation by exploiting advances in the electronics design, and vice versa.

This chapter will not discuss each separate design cycle of the software, but the design process and the result as a whole. We will start at how certain requirements were translated into design decisions.

11.2.1. Requirements and design philosophy

To design the software for the high-level interface, there are already a number of requirements listed in subsection 8.1.4. Let us briefly reflect on these. We want the user to be able to control the testing system with the interface, which in this case means controlling the Helmholtz cage.

We have requirements for manual (**SW-07**) as well as automatic control (**SW-09**) of this field, automatic rejection of the local Earth magnetic field (EMF) vector (which also implies the measurement of this vector, see **SW-06**), as well as the acquisition of measurement data (**SW-03**, **SW-04**, and **SW-11**). Requirement **SW-09** implies the need for a layer of abstraction to make sure that the instructions for the Helmholtz cage actually correspond to the orbital motion of a simulated satellite. The default interface should be a text-based interface, in the form of a Python application programming interface (API) or terminal command line interface (CLI), but as far as requirement **SW-02** is concerned, a graphical user interface (GUI) is optional.

With this starting point in mind, let us for a moment consider what the objective is. Back in section 3.3, we discussed that the system should be like an "argument by way of engineering". We know that performing these magnetic tests are important to verify the design of a small satellite. We also know that in this context, these tests are not always performed. The demonstrator system should be an argument as to how straightforwardly such tests can be implemented, and the software should reflect this.

For this reason, a text-based interface is nice, but it would be much better if *in addition to* this interface, there is an actual GUI. This allows a user to explore the functionality that the HCT provides much more quickly, intuitively, and without having to refer to a documentation file as much. In addition, it also allows the user much better feedback about what they are doing with the software, spot bugs and mistakes early, and overall get a much better picture of what is going on at any given moment.

Besides this advantage, the implementation of a GUI also allows the developer, that is, the author of this thesis, to verify his own work much more efficiently. Think about how easy it is to make a mistake during the implementation of coordinate frame transformations, and how difficult it can be to notice these mistakes without a graphic representation. Imagine trying to specify the orbital motion of a satellite; there are many ways of specifying orbital parameters. How are all the angles defined? Is the current orbit model correct? These questions can very easily be answered by a graphic representation, but can be rather painstaking to deal with if the only available feedback is text-based.

Furthermore, magnetic field lines are invisible. It is difficult to set up a Helmholtz cage so that it immediately shows its user what kind of field is being established at any given time. One would need some instrument that in real-time gives a proxy of both the field direction, as well as the magnitude. By contrast, a graphic representation can be made using software to easily provide this information, provided this information can be distilled from the sensor data.

For these reasons, it was decided to also implement a GUI into the high-level interface software. There are several GUI frameworks available with a Python API, but a very popular and comprehensive framework is the **Qt framework**. After some early experimentation, it was adopted as the GUI framework for the HCT. Its strengths are its wide platform compatibility and industry adoption, its rich feature set, and favourable licensing: for non-commercial applications such as the HCT, the Qt framework is free to use under the GNU General Public License (GPL).

Graphical user interfaces using Qt can be created with Python using the `PyQt5` or `PyQt6` packages, which serve as Python wrappers for Qt5 or Qt6 respectively. There is a second set of wrappers available in the `PySide2` and `PySide6` libraries, which are very similar to the aforementioned ones but are released under the GNU Lesser General Public License (LGPL). At the time of writing, the `PyQt5` is used: whilst the less restrictive licensing of the PySide libraries is desirable, some minor problems were discovered during development which were non-existent for the PyQt variant. In addition, whilst `PyQt6` is preferable since it is a more recent version, a drastic performance improvement was noticed for certain code running on the Raspberry Pi, which means that `PyQt5` is used for now, until it can be migrated later. The migration path is quite straightforward, and the HCT source code tries to consolidate package imports into only a few files, so upgrading to Qt6 at a later date should not pose much of a challenge.

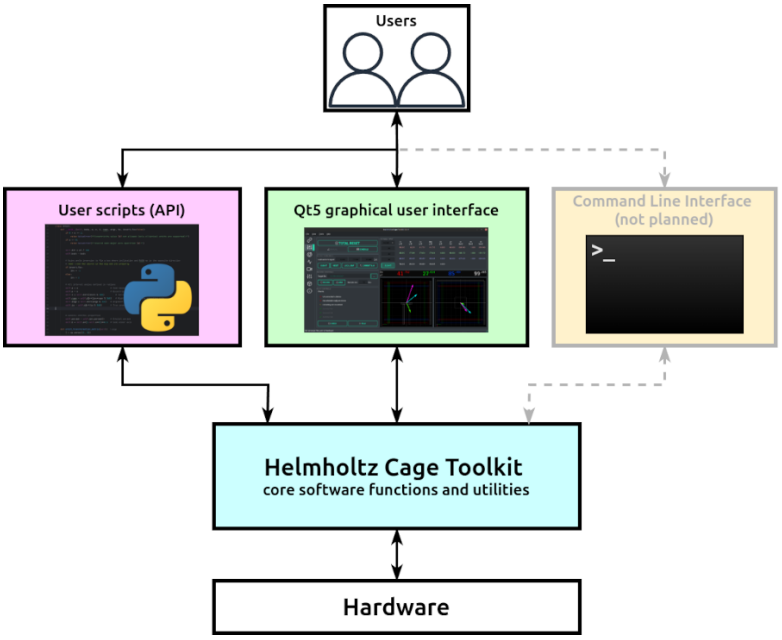


Figure 11.2: Diagram showing how users can interact with the Helmholtz Cage Toolkit software on the high-level interface. One can either call the functions directly in a script, using the API, or use the graphical user interface. A command line interface was also considered early on, but was later abandoned.

By virtue of **SW-02**, the GUI will be developed in such a way that it is a front-end for the underlying Python API of the HCT. This provides the user with some flexibility as to how they could use the software. If the GUI suffices, then this is the preferred option. If not, it is possible to just write Python scripts that call upon the same underlying functions as the GUI. This requires more familiarity with the source code, but allows the user much more flexibility beyond the limits of the GUI. This is also visualized in Figure 11.2. To view in more detail how this works, see also section 11.8.

By having both options available, one gets the best of both worlds without the need to manage two separate implementations of the HCT software. Such a dual setup is not uncommon in other software, an example being the CAD software FreeCAD. A CLI was also considered as a third option, but was later rejected due to the fact that besides the scripting and GUI options, it brought simply too little benefit to justify its development.

To conclude this section, it is important to reiterate how the software is taking shape. There are a number of requirements, as discussed before, but to fill the gaps, development is much more guided by a desire to have a software interface that is useful to its users, and can quickly be used out-of-the-box. During development, an informal goal was to structure the user interface in such a way that an engineering student working on a satellite project could download the software (for example from Github or PyPI), install the prerequisites, and get the general gist of the user interface in the span of about half an hour. The hope is that this user then realises the potential of the software for his own project, and that a lot of the heavy lifting of the code structuring, mathematics, and visualization coding has already been performed. With this thought in mind, we continue.

11.3. Development

Now that the general layout of the high-level interface software has been discussed, and a GUI has been decided upon, what is next? Specifically, how should this GUI be structured, and what should it contain? In Figure 11.3 one can see an early mock-up interface showing an early concept. Fundamentally, it consists of a series of tabs, each of which is accessible from the bar on the left. Each of these tabs then reveals a window on the right with widgets that correspond to a set of tools or functions.

This has three important advantages: First, it prevents the need for everything to be on the same screen all the time; whatever the user is looking at will be less cluttered, and more to-the-point. Second,

because of this, the software will be more light-weight to run, since not all functionalities need to be active at the same time, just those from the current tab and those running in the background. Finally, it also provides a useful framework for new features. If somebody has a wild idea that they wish to implement into the HCT, they do not need to worry about modifying the pre-existing GUI, they can add a new tab on the left and start fresh. These sorts of design decisions were made explicitly to ensure the HCT would be modular and malleable software, with good facilitation for adaption or future expansion by others (see EXTRA-03).

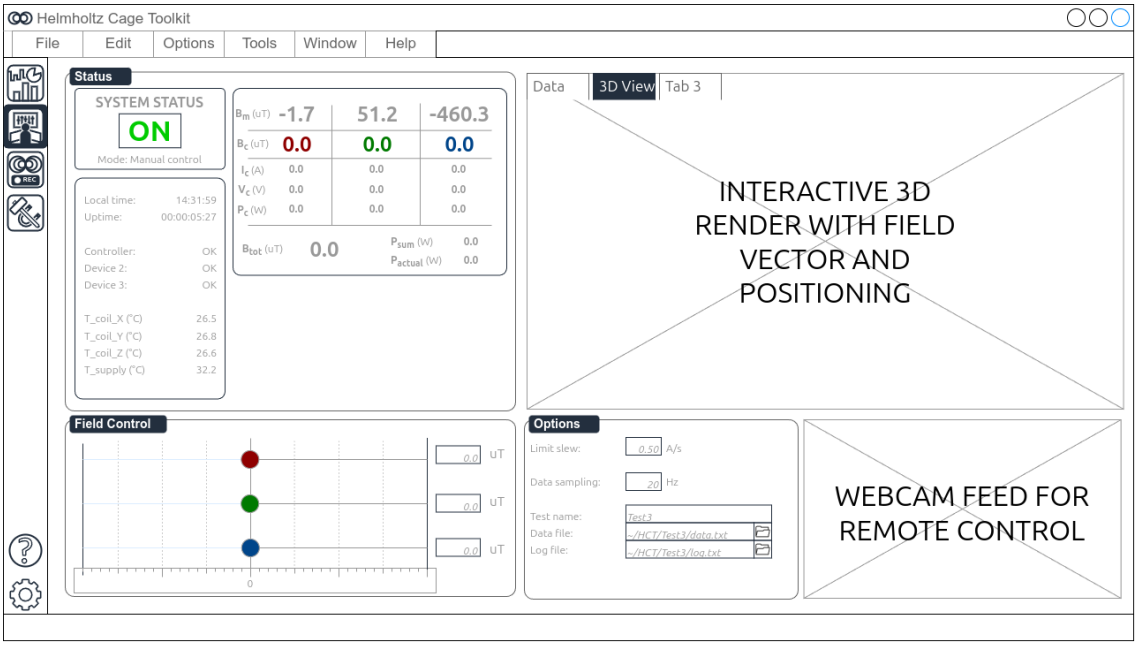


Figure 11.3: Early mock-up to plan out how the GUI should be structured.

As for the tabs, one could have a tab for configuring the setup, a tab for setting up data recording, a control tab for manual or automatic control of the Helmholtz cage, etc. The layout of Figure 11.3 shows a kind of control panel, with real-time hardware telemetry on the top left, a graphic representation of the Helmholtz cage in the top right. Furthermore, some control widgets for manual control of the magnetic field in the bottom left, and some space for loading and saving schedules for automated control, and an optional webcam feed. More advanced options would be accessible from the menu bar at the top.

As development on other aspects of the system progressed, the concept above was revisited and refined multiple times. For example, at the point in development when the concept of Figure 11.3 was made, the communication protocol between high-level interface and low-level interface was not yet defined. When some time later this was done, only then could a good connection utility in the GUI be made for it. Throughout the development, the software was carefully partitioned according to the principles of object-oriented programming, in line with EXTRA-03.

We will now shift gears, and stop discussing how the design of the high-level interface software evolved over time. Instead, the next section will deal with this software in the current state, and outline the features that were implemented there.

11.4. Current design and feature set

Once the electronics, low-level interface software and communication codec had been developed to relative maturity, the software of the high-level interface could be moved towards a version 1.0. From now on, we will restrict our discussion of the high-level interface software to this version.

11.4.1. General overview

In version 1.0, a number of new concepts have made their way into the design, and these must be discussed first. The most important of these are **schedules** and **generators**.

A **schedule** is a set of instructions for the Helmholtz cage. In essence, it is a long list of magnetic field values as a function of time that the Helmholtz cage can take and "play" on command. Typically, these field values represent the magnetic field as experienced by a virtual satellite, and as a result, these values are defined in the body reference frame of the satellite. If we assume that the satellite remains still inside the Helmholtz cage, the body reference frame of this satellite and the Helmholtz cage itself can be related by a simple, constant rotation matrix.

To generate these magnetic field values by hand is very laborious, and so it would be useful to have some tools to generate these schedules automatically, in a way that makes sense with the context given above. Version 1.0 of the high-level interface contains two such tools, and we refer to these as **generators**, because they generate schedules for the Helmholtz cage to use.

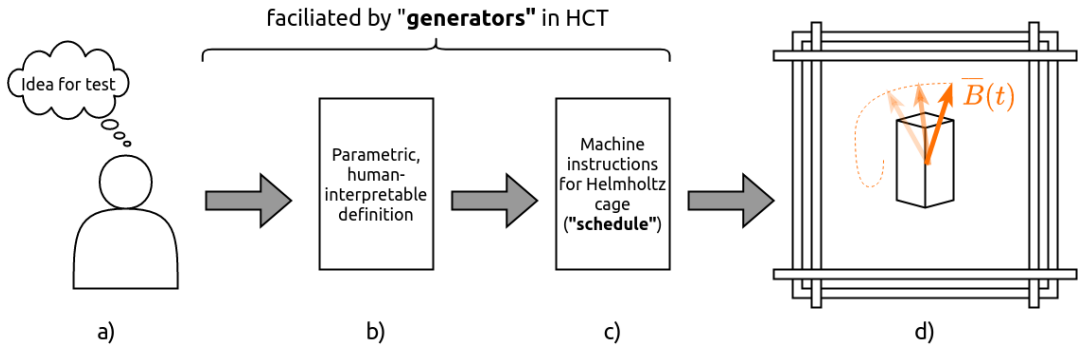


Figure 11.4: Interaction interface between human user and Helmholtz cage system for the creation of magnetic field vector schedules. The generator scripts inside the HCT translates parametric, human-interpretable instructions to instructions that the Helmholtz cage system can use.

The relationship between these concepts can be seen in Figure 11.4. In essence, generators take a couple of parametric inputs that are easily understood by a user, and turn them automatically into a schedule. This schedule could then be processed further, or sent to the Helmholtz cage via the transmission control protocol (TCP) connection and played back on command. The reason for this abstraction is that this structure allows for advanced users to write their own generators or schedule processors without having to fundamentally rewrite the high-level interface software. One could use scripts to generate new or edit existing schedules, or these tools could be implemented inside their own tab in the GUI application.

This particular approach was inspired by G-code in the context of 3D printing. In that case, the user defines a high level parametric geometry using CAD or 3D modelling software. This geometry is then turned into a machine-usable schedule called G-code by a **slicer**. The 3D printer of the user then executes this code to print the geometry. In this scenario, the modelling software, the slicing software, and the 3D printer itself can all be interchanged, but the overall interface between these parts remains the same. The generator and schedule concepts used by the HCT mirror this.

Version 1.0 of the high-level interface software contains two generators: the **Cyclics** generator and the **Orbital** generator, and we will take a closer look at these in a moment. Beyond this, version 1.0 has a very similar layout to its humble beginnings: there is a menu bar and a tab bar for navigation, and then there are individual tabs with goal-specific tools in each tab accessible to the user. In version 1.0 there are seven tabs:

1. The **Connection Window**, which allows the user to connect the GUI application as a client to the server. In addition, there is connection telemetry and the option to send and edit remote schedules that are currently loaded into the low-level interface.
2. The **Command Window**, which is the primary software interface to the Helmholtz cage. It displays information about the current status of the Helmholtz cage, allows control using a schedule or via manual input, and has data recording controls and ambient field vector rejection.

3. The **Orbital Generator**, which allows the user to generate magnetic field schedules in the body frame of a simulated satellite using a few parametric inputs, such as orbital elements, satellite body angular positions and rates, simulation time resolution, etc. In short, it is the currently recommended tool to generate schedules that represent satellite motion.
4. The **Cyclics Generator**, which allows the user to generate magnetic field schedules based on periodic functions. These are less useful for simulation of satellites, but useful as "test tone generators" for testing, calibration, and verification of the Helmholtz cage itself. It allows the user to choose from a number of parametric functions for each axis of the Helmholtz cage, such as sinusoids, square waves, etc. Gaussian or uniform noise can also be added.
5. The **Webcam Window**, which is planned to house a webcam feed for remote viewing of a test setup. It is currently placeholder, and its functionality will be implemented at a later date. As such, we will not discuss it further here.
6. A designated placeholder tab. It has currently no function, other than to provide a quick starting point for third parties to implement their own functions in a separate tab.
7. An **Information/Documentation panel**. This can be used to place documentation and help pages inside the program. However, due to the development schedule its functionality will not be implemented in version 1.0 of the software, and so this tab can be considered placeholder.

Let us now take a closer look at some of these tabs.

11.4.2. Connection Window

To enable information exchange between the high-level interface and the low-level interface, a TCP connection is used (see Figure 11.4). We will take a closer look at this connection in later sections of this chapter, but for now it is only necessary to understand some basic properties of this connection:

- The low-level interface hosts a TCP server that multiple clients may connect to
- Users can connect to this server using a framework that can host a TCP client. Here, this typically means connecting via a script or using the GUI application.
- Data over the connection is exchanged in packets (see section 11.6). When active, the server listens for incoming packets, handles these, and usually sends a response packet.
- The data in this context typically consists of commands to the server, telemetry data from the server, the segments of a transferring schedule, and server pings.
- To connect to the server, the client needs the **server address** and its **port number**.

When connecting via Python script, one can most easily do this with the Python package `socket`. In contrast, the GUI application uses the `QTcpSocket` class from `PyQt5`, because it integrates much better in the surrounding Qt framework. The *Connection Window* makes setting up this connection very straightforward.

Let us have a look at the contents of the connection window (Figure 11.5). At the top, there are the connect and disconnect buttons, and input boxes for the server address and its port number. When a user presses connect, the `QTcpSocket` running in the background takes the address and port number and tries to make a connection. It may establish connection successfully, or an error may occur; in both cases more details are put in the status bar at the bottom of the window.

In the case of Figure 11.5, a connection was made successfully to a dummy server running on the same computer, so the address of the host and the client are both the *localhost* address (`127.0.0.1`), but if the server is running on the Raspberry Pi, these addresses will be different. The default address and port number that are in the boxes can be set in the config file. There is also an option there to have the GUI application automatically attempt a connection upon launch.

When a connection between client and host is made, some connection details are displayed in the middle of the screen, such as the uptime, the number of packets exchanged, etc. There is also a response time tester, which fires a number of echo packets to the server, and displays the average response time. The number of packages it fires for this test can also be set in the config file.

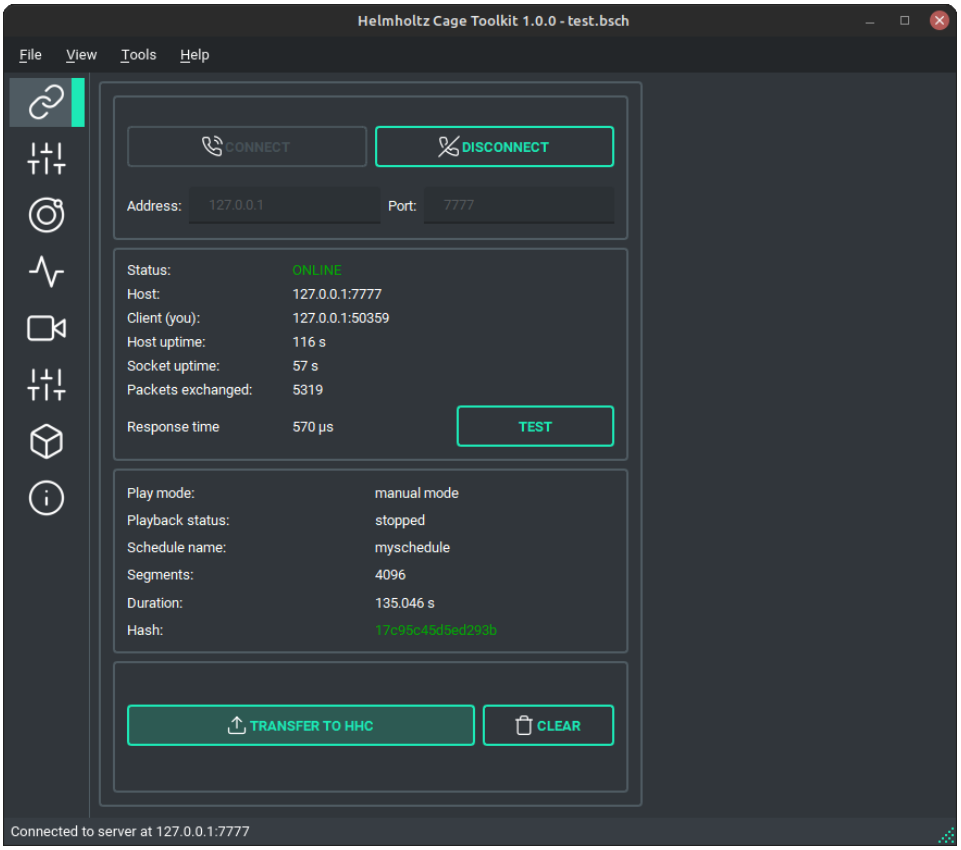


Figure 11.5: Overview of the Connection Window.

Underneath the information about the connection, there is information about the schedule currently loaded onto the low-level interface. This is the **remote schedule**, and information about it is periodically fetched from the server. This information also shows the status of the Helmholtz cage, and whether it is playing the schedule. Underneath this information, there are two more buttons: One of these buttons clears the remote schedule. The other button takes the schedule currently loaded inside the GUI application and sends it to the server, where it replaces the current remote schedule.

To make this a bit more clear, consider again Figure 11.5. At the top of the screen, the window shows that a schedule called `test.bsch` is loaded in the GUI application. The information box in the connection window shows that on the server, there is a remote schedule called `myschedule` currently loaded there, which was transferred in the past. If the user were to press the transfer button now, the schedule contained in `test.bsch` would be transferred in segments to the server, and if this all goes well, it would update the information in the connection window accordingly. This is intended way of sending schedules to the low-level interface. After transferring all segments, the schedule is hashed both locally and remotely using the BLAKE-2b hashing function, which serves as a checksum to verify that the schedule was transferred without corruption. If the checksum passes, the hash in the connection window will turn green like in Figure 11.5, otherwise it will be red.

This is all there is to the connection window. In the config file, there are options to automatically connect upon launch using the default address and port number, and there is also an option what tab is shown by default upon launch. Therefore in proven, well-configured setups, it is unlikely that the user has to spend much time here, and can start working as soon as the application launches.

11.4.3. Command Window

The *Command Window* is the primary method by which a user interacts with the Helmholtz cage hardware. There are several tools here that are helpful during calibration, debugging of the Helmholtz cage, as well as performing tests with it. Details about its layout can be seen in Figure 11.6.

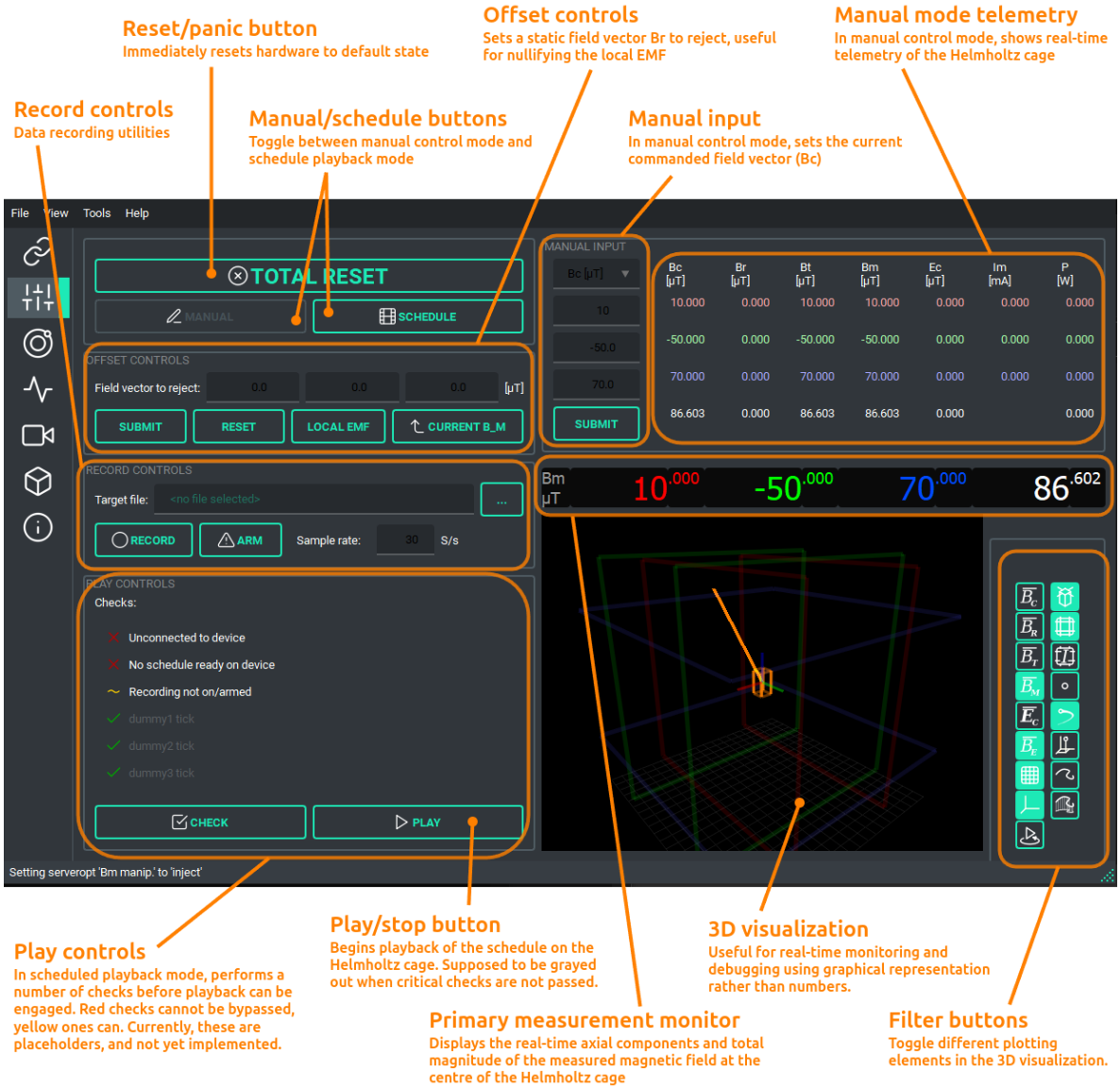


Figure 11.6: Overview of the command window in manual mode.

The command window currently has two modes of operation:

1. **Manual mode**: This is an interactive mode in which the Helmholtz cage can be commanded with user-defined magnetic field instructions. It is primarily intended for static tests and calibration. The panel in the top right of the window shows user inputs for the desired magnetic field vector, and shows real-time telemetry about the actuated field vector, the coil currents, and total power output.
2. **Scheduled mode**: This mode can be used if a schedule has been transferred to the low-level interface and is ready to be played back. Manual instruction of the field vector is not possible, as the schedule overrides it. The top right panel shows the magnetic field envelopes of all three axes in a graph (see Figure 11.7).

Let us have a closer look at the interface and see what tools are available. On the top left, we see a large reset button, which is the most powerful reset that can be implemented via software. It sends a command to the Helmholtz cage to interrupt everything it is doing and reset itself gracefully. The idea is that if anything goes wrong with the Helmholtz cage, this button can be used to return to a reset position to avoid damage or danger, so it is important that it is in easy reach of the user during operation. If this button does not work during a fault, the user can move directly to system's power button and perform a hard reset.

Underneath the reset and mode buttons, there is a section labeled **Offset Controls**. It can be used to specify a rejection value, which on the side of the low-level interface is subtracted from the field vector that is currently being commanded:

$$\overline{B}_{output} = \overline{B}_{C(ommanded)} - \overline{B}_{R(eject)}$$

When the user presses **"SUBMIT"**, the three field component specified in the boxes are sent to the low-level interface. This has a number of important uses: First, it can be used to nullify the local EMF at the test location. As a shortcut, the config file may contain either pre-specified local EMF value, or the local coordinates of the test location (with which the local values can be calculated using the internal International Geomagnetic Reference Field (IGRF) model). When the **"LOCAL EMF"** button is pressed, these values are automatically filled into the boxes.

Alternatively, it may be that the user is interested in cancelling all magnetic fields at the test location, including other local disturbances. In this case, the **"CURRENT B_M"** can be pressed, which fills the boxes with the currently measured value. The *"RESET"* button sets the values back to zero.

Underneath the offset controls there are the recording controls, with some basic tools to record the measured magnetic field data to a specified file at a constant rate. The **"RECORD"** button starts this process immediately, whilst the **"ARM"** button turns on the recording the moment that schedule playback starts. The recording tools are fairly basic, and for more flexible usage, it is recommended to use a scripted recorder that connects to the Helmholtz cage as a separate TCP client and records from there.

In the bottom left, there are the play controls, used when scheduled mode is active. There are a number of checks that can be performed here, to help the user ensure that everything is ready before playback is begun. The idea is that there are hard checks, such as the fact that a schedule must be present on the low-level interface before playback can begun. Such checks cannot be bypassed and show as red when not passed. Then there are checks that are simply warnings, which warn the user but do not block playback. These show up as yellow. The playback button is supposed to be grayed out until all checks are either green or yellow. However, whilst this functionality is implemented as a framework, it is not currently active, and the "checks" that are displayed are just impotent placeholders to be implemented in the future.

When the user hits the **"PLAY"** button, the low-level interface is instructed to commence playback of the schedule. At this point, the low-level interface will have control over the playback, and the high-level interface simply constantly asks for telemetry, from which it reconstitutes what part of the schedule is played back at any given time. The instantaneous schedule step number is part of this telemetry, and so the maximum synchronization delay between the low-level and the high-level interface equals the step size (resolution) of the schedule. However, the telemetry also includes Unix timestamps, and so as long

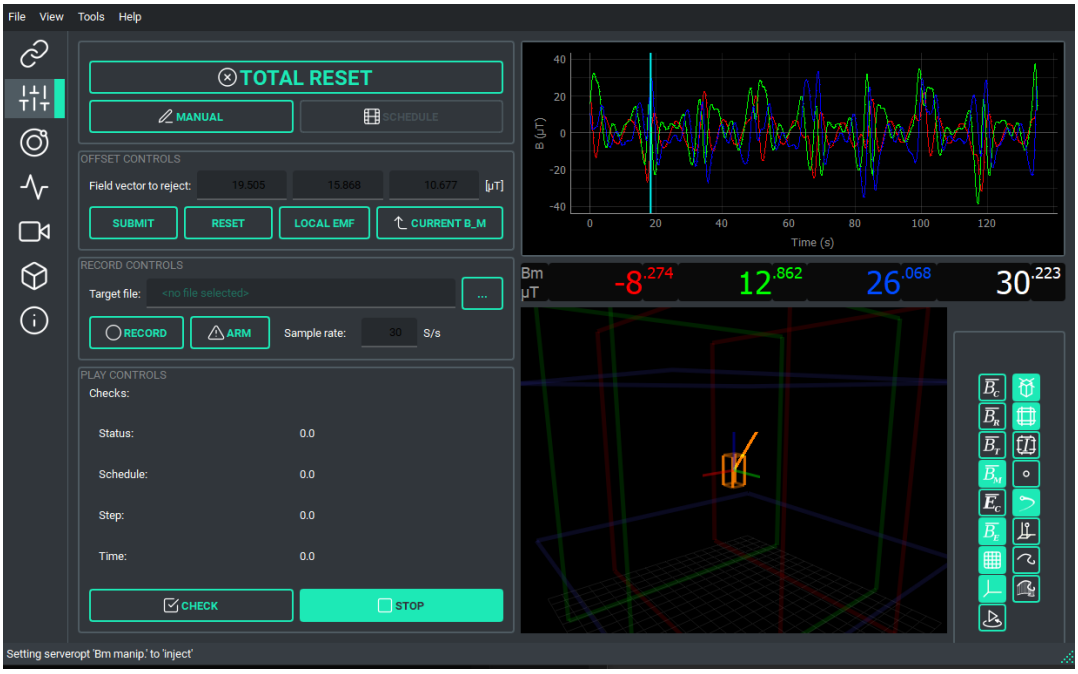


Figure 11.7: Using the command window in scheduled mode.

as the clocks on the high-level interface and the low-level interface are synchronized, any step delays can be removed from the recorded data afterwards.

During playback, the **"PLAY"** button turns into a **"STOP"** button. When this is pressed, playback is stopped, and playback becomes briefly unavailable to allow the Helmholtz cage to reset the hardware and the schedule playback. When it is done, playback can be started again. Pausing during live playback is currently not supported.

Finally, the bottom right corner of the command window has a 3D visualization of the Helmholtz cage. The magnetic field vector is directly visualized here, as well as many other things, so besides general system monitoring, this widget also shows well if the intended field vector and the realized, measured field vector align. A **command error vector** $\overline{E}_C$ can be visualized just for this purpose, where:

$$\overline{E}_C = \overline{B}_{M(easured)} - \overline{B}_{C(ommanded)}$$

11.4.4. Cyclics Generator

The *Cyclics generator* is one of the two primary ways of generating magnetic field schedules in the GUI application, and for this reason it has its own tab. It excels at generating test signals for benchmarking and debugging of the Helmholtz cage system itself. It is named *"Cyclics"* because of its selection of periodic functions, although a few aperiodic functions (constant, linear, noise) can be generated as well. The contents of the tab containing the Cyclics generator can be seen in Figure 11.8.

On the left side of the tabs, there are a number of parametrics for the user to define. By default, these values assume the default values specified in the config file, which can be changed by the user, if desired.

First, the duration and resolution of the schedule have to be specified. The duration is simply the total duration of the schedule in seconds, and the resolution is the number of samples per second that the schedule will generate. The resolution is therefore a constant, but this is an artificially imposed limit of the generator to simplify things; schedules with a variable spacing between points can still be played without problems by the actual Helmholtz cage hardware. If the resolution is set too low, the schedule is becomes quite choppy, and may even go below the Nyquist rate of the periodic waveforms that are specified. If the resolution is too high, the hardware of the Helmholtz cage may have trouble keeping up. A schedule of a few dozen samples per second (S/s) is typically safe, conservative starting point.

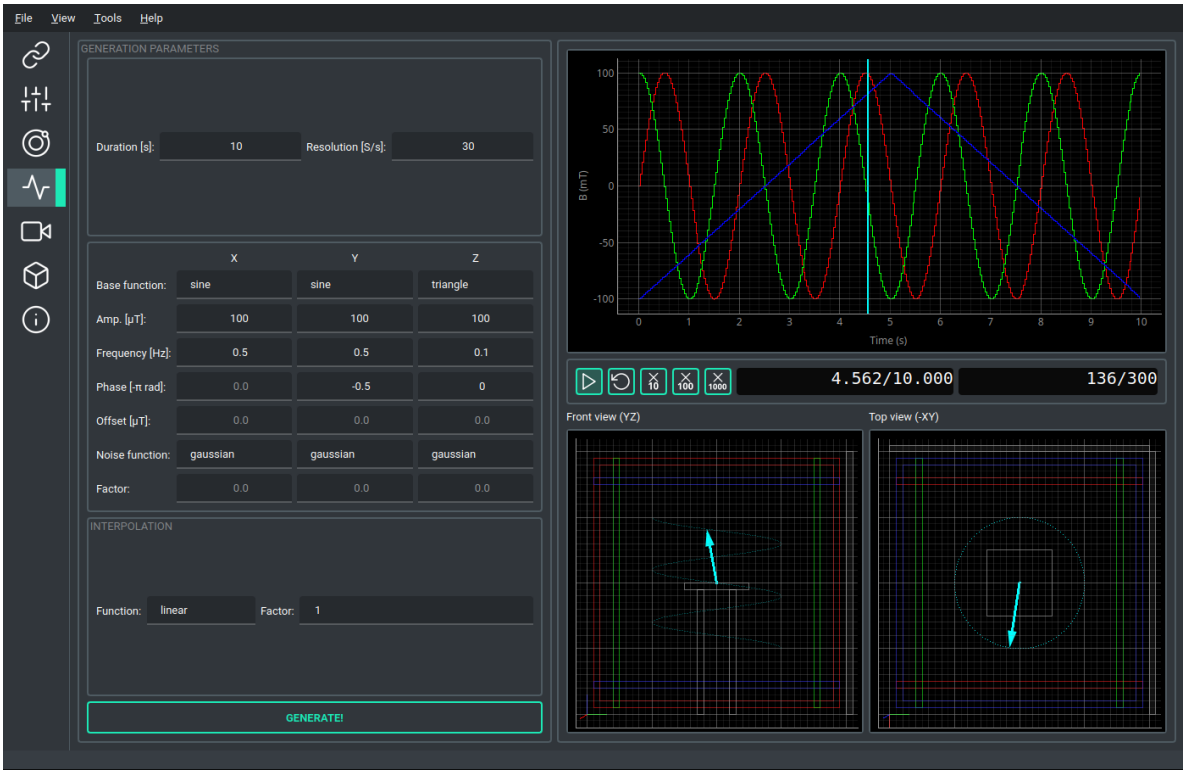


Figure 11.8: Overview of the Cyclics generator. Input parameters are on the left, and some visualizations of the resulting schedule are on the right. In this case, a helical test pattern is generated using a sine (X), a cosine (Y), and a triangle wave (Z). The resulting schedule will be 10 seconds long, and describe a periodic helix with an absolute magnitude of at most 100 μT on every axis.

Below this, there is an input section with three columns. Each column represents the function to be applied to a certain axis of the Helmholtz cage: X, Y, and Z. The user may first choose a base function from the drop-down menu, and this sets the function that will be parametrically generated for the corresponding axis. The following base functions are available:

1. Constant value (default)
2. Linear
3. Sinusoid
4. Sawtooth wave
5. Triangle wave
6. Square wave

Where relevant, the parametric values below can then be used to shape this base function. For example, for the sinusoid base function, the amplitude, frequency, phase, and absolute offset can be specified, but for the constant value, the frequency and phase will have no meaning, and do not affect it. In Figure 11.8, a combination of two sinusoids and a triangle wave is used to generate a helical test signal with a duration of 10 seconds.

In addition, each axis allows a degree of random noise to be added to the waveform. This is useful for evaluating the effect of noise on the system, and is therefore of great benefit to the generator. There are two noise functions: Gaussian and uniform. The magnitude of the noise is expressed as a factor of the base function's amplitude, so a *factor* of 0.1 will give a signal to noise ratio of 10. At a factor of 0.0 the noise is zero, which is the default value.

Below, there are some options for interpolation. This can be used to artificially increase the resolution of the schedule without changing the resolution of the underlying base function, for example for performance reasons. In practice, the Cyclics generator is very light-weight and increasing the overall

schedule resolution will typically be preferable to interpolation, but for the Orbital generator, which is much more computationally heavy, interpolation becomes quite a useful feature. The interpolation *factor* describes how many extra points will be added for each point in the schedule; a 10 second schedule with 300 points on which an interpolation factor of 2 is applied will result in a 10 second schedule with 600 (or rather, 599) points.

The order in which the parameters are arranged on the graphical interface is roughly how they will be applied in computation; When the user presses the *GENERATE!* button, the underlying code will first read all the generation parameters and store them. A schedule template is made using the *duration* and *resolution* parameters. Then, the base functions will be generated, and afterwards the noise components will be added. Finally, interpolation will be performed on the schedule.

On the right side of the panel, the user has the option to review the resulting schedule, and check these for any errors.

At the top right, there is the schedule visualizer, which plots the magnetic field strength as a function of time for each axis across the schedule. In the bottom right, the three axial components are combined into a 3D path, and this path is visualized with two 2D plots. The reason that these 2D plots are used, rather than a 3D plot, is that it is much easier to see the absolute magnitudes of the vectors on the 2D plots. As an example, take a look at Figure 11.8, which is displaying a helical test schedule created by the Cyclics generator.

Between the plots, there are playback controls. These do not control the playback of the Helmholtz cage, but instead simulate the propagation of the schedule across time, and the plots react in correspondence. There is a play/pause button, and a reset button to control the currently selected section in the schedule. For very long schedules, there are accelerator buttons available that increase the playback speed by a factor of 10, 100, and 1000 respectively. Finally, the current playback time and current schedule step are displayed to the right.

Visualizations like these greatly reduce the chances for user error and make the generator very intuitive to use by trial-and-error. Not only are these useful for verifying the system for the author, but also will help make users familiar with the way the software works very quickly.

11.4.5. Orbital Generator

Compared to the Cyclics generator, the *Orbital generator* is much more complicated, but ultimately much more useful to users who wish to test satellites. Its prime goal is to easily generate schedules that correspond to the magnetic field experienced by a satellite in orbit. The Orbital generator certainly cannot hold a candle to an advanced orbital propagator such as GMAT or TUDAT, but even with its limited feature set, many things can be done for simple missions.

This section will discuss the Orbital generator for practical purposes; we will cover the capabilities and limitations of the generator, as well as how to use it. For a more in-depth discussion on how the Orbital generator works under the hood, the reader is referred to Appendix F.

The orbital simulator has the following features:

1. Simulate low Earth orbit (LEO) motion of a satellite with constant circular or elliptical orbits.
2. Simulate constant angular motion of a satellite body.
3. Simulate rotation of the Earth as a function of time.
4. Generate magnetic field data using the IGRF model, and transform it to the body frame of the satellite.
5. Visualize this data both in an Earth-centred reference frame, as well as the satellite reference frame, for error-checking purposes.

There are also a few important limitation to be aware of:

1. The Orbital generator can only simulate elliptical Earth orbits, not hyperbolic trajectories or orbits around other planets.

2. The orbit model only takes into account Earth's gravity, not that of Luna, Sol, Jupiter, or any other celestial body.
3. The orbit model also assumes a spherical, homogeneous Earth, and therefore neglects nuances such as planetary oblateness and Earth's particular gravity distribution.
4. Since the generator uses the IGRF model, it inherits its limitations, such as the approximately 20 nT/year extrapolation error of the IGRF [77].
5. The Orbital generator performs worse for high orbital eccentricity values. It is generally not recommended to use it for orbits with e > 0.5, as this will introduce distortion in the orbit shape For an in-depth discussion about why this is so, consult subsection F.3.3.
6. Satellite models are limited to simple, point-mass inertial models with constant angular motion.

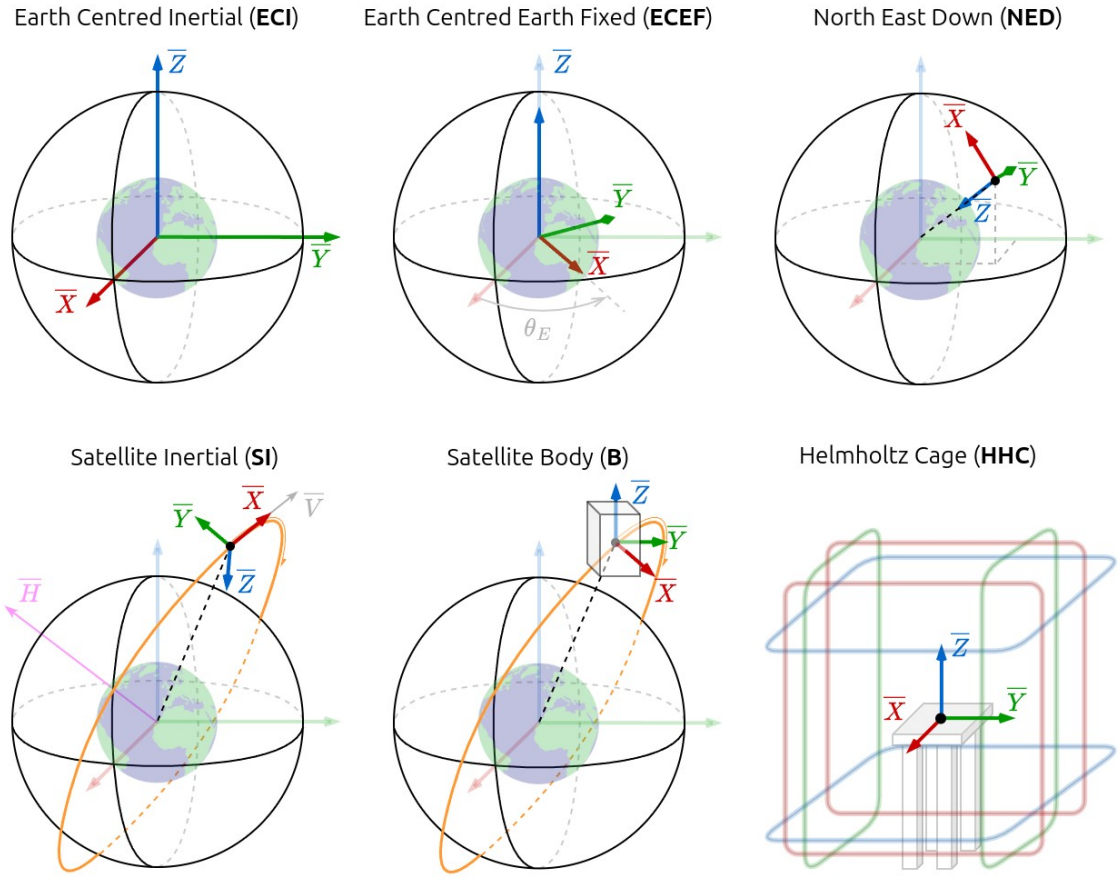


Figure 11.9: Reference frames used by the Orbitals generator.

Let us take a closer look at how the Orbitals generator works in practice. Like with the Cyclics generator, we have parametric inputs on the left, and visualizations on the right. The visualizations consist of a 3D orbital plot and a Helmholtz cage plot.

On the left, we have the generation parameters, which consist of a few simulation related parameters:

- Total **number of simulation points**, which dictates the length of the simulation.
- The number of **points per orbit**. The number of orbits propagated by the simulation is the number of simulation points divided by this.
- The **time speed factor**. At a value of 1.0, the simulation performs in real-time, meaning a simulation of a LEO will take about 90 minutes. At higher factors, time is sped up proportionally.

- The **time and date of the simulation's start**. This is expressed in decimal date format [1].
- **Earth datum** at t_0. This is the angle of Earth's axial rotation with respect to the zero datum of the Earth-Centered Inertial (ECI) inertial reference frame. It is also the rotation angle between the ECI and Earth-Centered, Earth-Fixed (ECEF) reference frames, often denoted by θ_t.

The next set of input parameters are a set of six orbital elements that fully describe the elliptical orbit.

- **Eccentricity** e ($[0, 1)$, but recommended to use $[0, 0.5]$).
- The orbital **altitude at perigee** h_p, expressed in kilometers. For circular orbits, this is identical to the orbital altitude everywhere. Recommended up to 1000 km only.
- Orbital **inclination** i in degrees ($[0, 180]$).
- The **Right ascension of the ascending node (RAAN)** Ω in degrees ($[0, 360]$).
- The **argument of periapsis** ω in degrees ($[0, 360]$).
- **Initial mean anomaly** M_0 (or θ_0) in degrees ($[0, 360]$).

Next, the user must specify the body angles of the satellite ϕ, θ and ψ in degrees, as well as their derivatives (angular rates) in degrees per second. Finally, there are interpolation options, with the choice between no, linear, and spline interpolation, and with an interpolation factor expressing the number of points added per point. All this can be seen in Figure 11.10. The config file contains the default input values and a number of additional constants for the advanced user to edit as he desires.

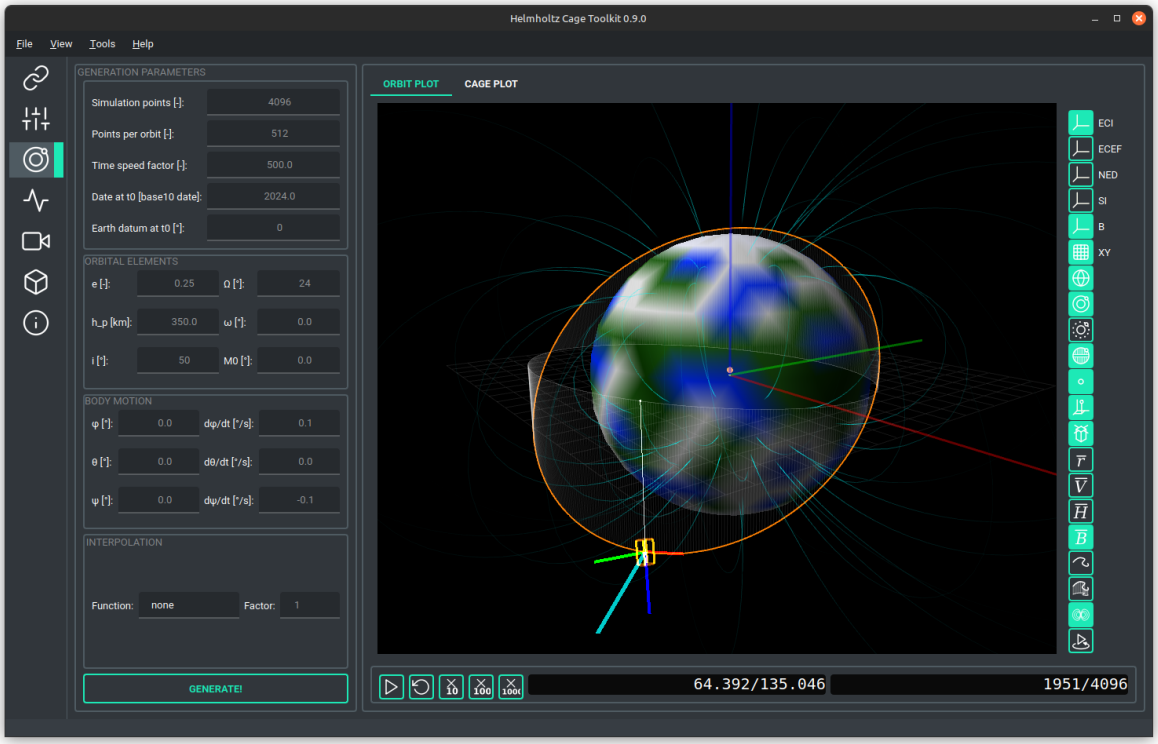


Figure 11.10: Overview of the Orbital generator, with input parameters on the left and orbital visualization on the right.

When the user presses the *GENERATE!* button, the input values are copied and stored. An orbital model is constructed, and the satellite is propagated along it, along with its motion. The Earth's rotation is also propagated at the same time. For every point, the local Earth magnetic field vector is calculated using the IGRF model and expressed a local North-East-Down (NED) frame. These values are first transformed to the ECEF and ECI frames for visualization purposes, and then to the satellite body frame

[1] The decimal date is also known as a **year fraction**. For example, 13:30:00 on April 19th, 2025 can be referred to as decimal date 2025.2973173515982. There are many tools to convert to and from this format, such as the Python package `datetime`

B for more visualization, but more importantly, these are used for the resulting schedule. This process can take a while for long schedules or high resolution settings.

On the right of Figure 11.10, there is an orbital plot that is able to show many things about the generated orbit. The primary purpose of this visualization is so that the user can assess his work and ensure that the resulting schedule matches his intentions. The plot has many buttons on the right-most side, each of which can toggle a particular plotted element on or off. These include tripods for particular reference frames, specific important vectors (velocity vector $\overline{V}$, angular momentum vector $\overline{H}$, local field vector $\overline{B}$, etc.), and various representations of the orbit, the Earth, and the satellite. Internally, all plotted elements have to be expressed in terms of the ECI frame to create this plot.

The contents of the plot match the contents of the created schedule. Similarly to the Cyclics plot, the plot can be animated using the buttons below; if activated, the satellite will start progressing through its orbit, rotating as it does so, whilst the Earth rotates underneath. All relevant plotting elements move in accordance with this animation, so there is a lot of useful information for the user to learn what is happening and adjust his course of action in accordance.

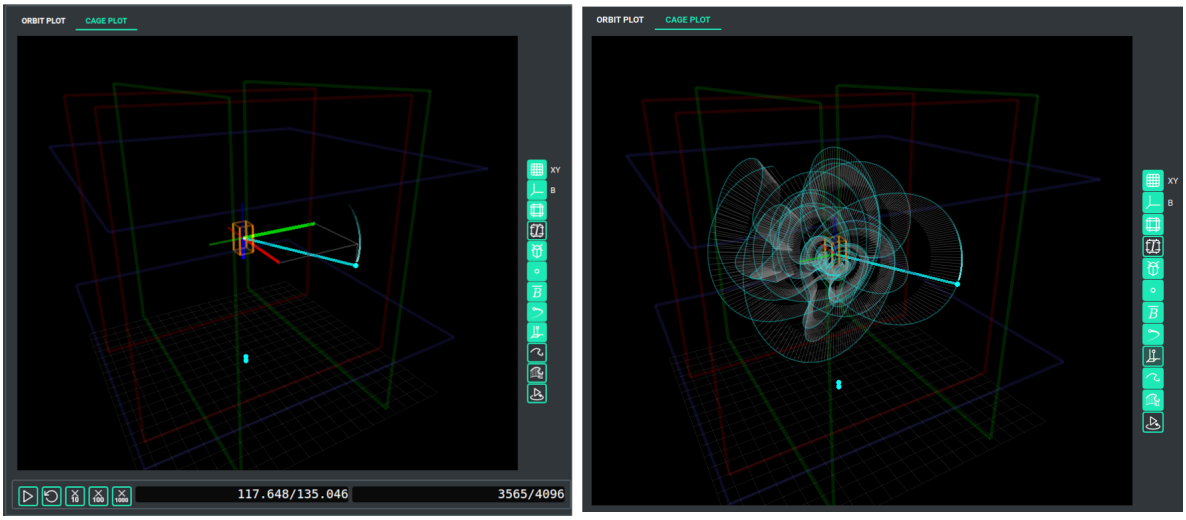


Figure 11.11: Two possible views of the Orbital generator's output in terms of the activity of the Helmholtz cage. The red-green-blue colours represent the X-Y-Z axes respectively. On the left, we see clearly what instantaneous magnetic field vector is being established by the Helmholtz cage at the current time, as well as a brief history of the previous positions using the trail. On the right, we have a more global overview of the whole magnetic contour described by the schedule. This view shows the complexity of the contour, and therefore shows why the Orbital generator was made: it would take a long time to create such a contour by hand, whilst the generator does it in under 2 seconds.

At the top of the orbital plot, there is a second tab, which can be used to access the Helmholtz cage plot. This can also be seen in Figure F.8. The cage plot moves along with the schedule just as the orbital plot does, but rather than showing the orbit, it shows the magnetic contour of the field vector experienced by the satellite across time. This plot is therefore made with values expressed in the satellite's body frame, rather than the ECI frame.

As with the other plot, there are numerous options available using the buttons on the right, such as the visibility of the various plottable elements, as well as the ability to slowly rotate the plot in 3D to improve the degree to which the plot's spatial properties can be interpreted. As with many aspects of the HCT, many of the properties of these plots can be changed from the config file, such as the rate of the aforementioned rotation, and all the way down to the colour and shape of the satellite model.

In conclusion, the Orbital generator excels at being able to very quickly propagate an orbital scenario and turn this into a set of Helmholtz cage instructions. It also has some detailed visualization options so that the user has excellent opportunity to spot errors and validate the schedule beforehand. For the purposes of hardware and software validation of a CubeSat, most of which go to a simple elliptical low-Earth orbit, this will get you most of the way.

So, if you are a developer working on such a mission, the HCT has most of the tools to set up such a test in minutes. A lot of the heavy lifting in terms of software development has already been done, meaning that a lot of development time can be saved. Only if additional features are needed will additional software need to be written, but as mentioned before, the HCT has good infrastructure to facilitate this, as it was built from the ground up with this goal in mind.

11.4.6. Schedules

In this section, we take a more detailed look at these *schedules* that have been mentioned several times so far. As a concept, schedules were heavily inspired by the G-code format. The internal generators inside the HCT produce a schedule as a dataset which is stored internally as Python data objects. This is useful for internal plots, but not very useful for much else. However, these internal schedules can either be **exported** to the low-level interface, or **saved as a file**.

If exporting the schedule to the low-level interface, the client first sends some instructions that a schedule is about to be sent, mentioning details such as the schedule's name, length, etc. Then it will turn each step of the schedule into a segment, each of which is encapsulated into an S-packet (as defined by the Simple Command Codec (SCC), see section 11.6) and sequentially sent over the TCP connection. After all packets has been transferred, the high-level interface locally hashes the schedule using the BLAKE-2b algorithm, and requests the that the low-level interface does the same and send the remote hash back. The local hash and remote hash are compared, and if they match, the schedule should have been transferred correctly.

A schedule can also be saved for a later date, which is useful for tests that are performed more than once, or just for general traceability during development. By saving the schedule, it is turned into a `.bsch` B-schedule file, which is just a plaintext document with a header and the schedule stored as comma-separated values. In Figure 11.13, one can see what a `.bsch` file contains.

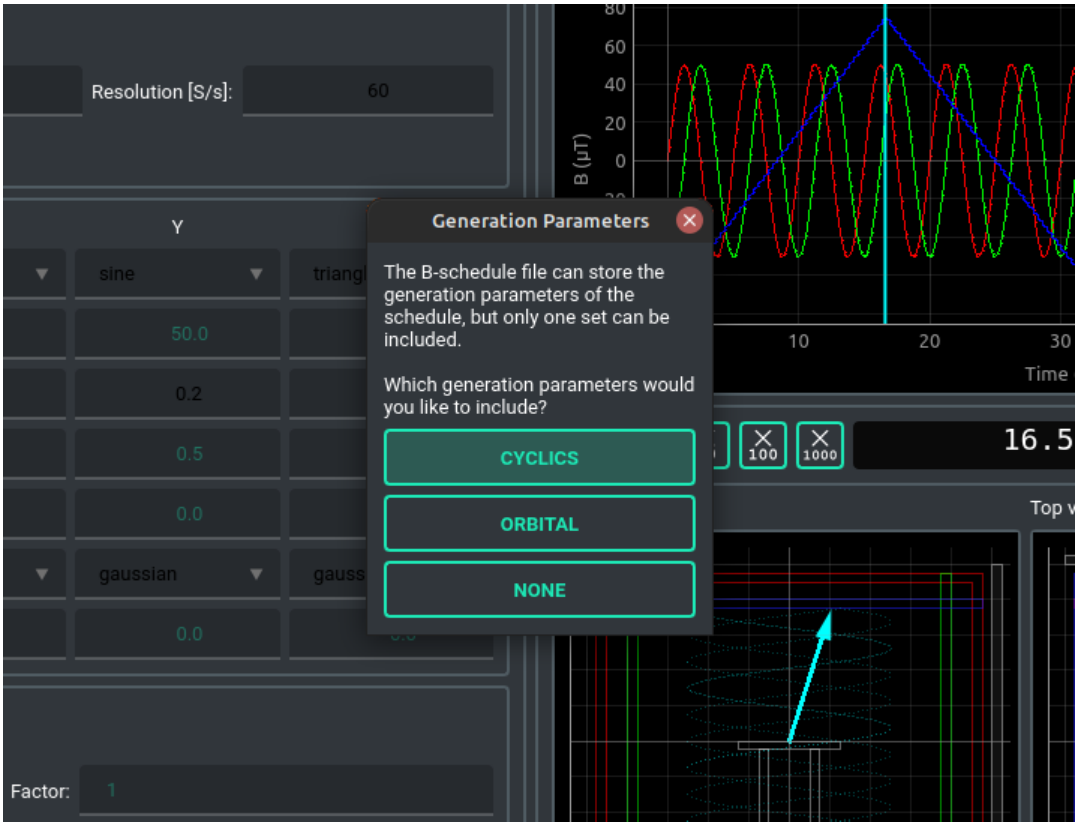


Figure 11.12: When saving a schedule to a file using the GUI application, the user is prompted with the option to save the generation parameters to the `.bsch` file, in addition to the raw schedule data.

The B-schedule files do not just store the raw schedule data, but also contain some metadata in a **header**, which comprises the first 10 lines of the file. Here, some extra information about the schedule is stored. Perhaps most importantly, if the schedule was generated using a generator, the header may optionally include the **generation parameters** that were used to create the schedule. In the GUI application, the user is automatically prompted for this when saving a schedule (see Figure 11.12). Note that these generation parameters are enough information to "store" a schedule, because the schedule data can simply be regenerated again at a later date if the generation parameters are known.

Let us look at the contents of a `.bsch` file in more detail. Every B-schedule file starts with the header command `!BSCH`, which is just an identifier that the HCT software uses to check that the file being loaded is indeed a B-schdule file. On the second line of the file, the **schedule name** is stored. The third line is used to indicate what generator was used to generate the schedule, if any. If a schedule was created by other means, none values are allowed, of course. The fourth line of the schedule then contains the generation parameters that were used by the generator to generate the schedule. The GUI application uses these values to automatically fill in the boxes in the generator. Similarly, the fifth line stores interpolation parameters. Lines 6 through 9 are reserved for additional metadata that may be added in the future. Line 10 is a line of # characters to visually indicate the end of the header.

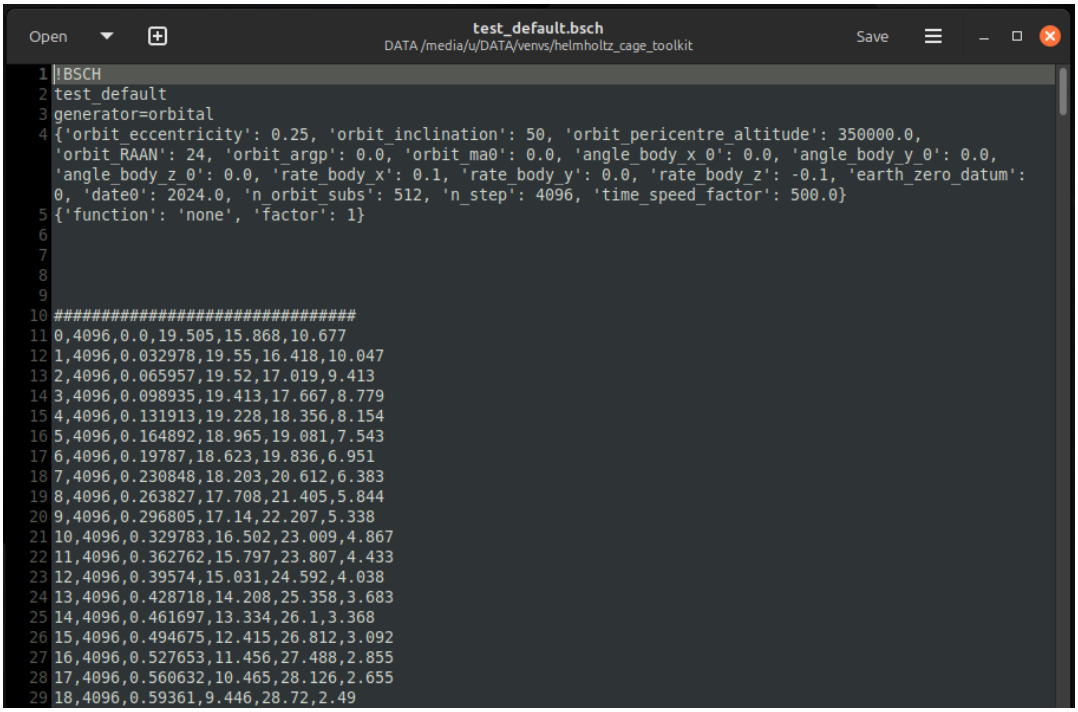


Figure 11.13: An example of the first 29 lines of a B-schedule file. This schedule was created using the Orbitals generator in the GUI application, and as a result, the generation parameters are included in the header.

From line 11 and onward, the B-schedule file contains comma-separated values that indicate each point (or 'segment') of the schedule. Each line is in the following format:

1. Step/segment number.
2. Total number of steps in schedule.
3. Time of start of step in seconds.
4. Magnetic field strength in X-direction in μT.
5. Magnetic field strength in Y-direction in μT.
6. Magnetic field strength in Z-direction in μT.

Due to the way this file format is set up, internal data compression is not available out-of-the-box. Instead, the simplicity of the file format was valued much more above file size, so that users can

write their own scripts if needed to generate these schedules, for example to load them into the GUI application later. If data compression is needed, one can always use any general-purpose compression algorithm to compress the files, since they are plaintext after all.

11.5. Low-level interface software

In this section, we will turn our attention to the software that is running on the low-level interface. From Figure 11.1, we can see that this software communicates with the high-level interface via the TCP connection, and is connected to the actual hardware of the system, such as sensors, actuators, etc. The former of these will be discussed in more detail in section 11.6, and the latter is elaborated in chapter 10.

As for the software that runs on the low-level interface, we must distinguish two layers: the low-level interface runs on a Raspberry Pi 4, which is a single-board computer with a quad-core Cortex-A76 processor. It is capable of running a fully fledged Unix-like operating system (OS), which will manage processes, schedules, memory handling, etc. The operating system on the demonstrator system is not the default Raspberry Pi OS, but rather a freely available distro called *ADI Kuiper Linux* by Analog Devices. This operating system contains drivers out-of-the-box for working with the analog-to-digital converter (ADC) and digital-to-analog converter (DAC) hardware on the CN-0554 board, which makes setting up the system much easier.

However, in this section, when we refer to the "low-level interface software", we do not refer to this operating system, although it is an essential part of the system. Rather, the low-level interface software is comprised of a number of Python scripts that are part of the HCT. These scripts set up a multi-threaded process that acts as a TCP socket server, so that client devices such as the high-level interface computer can connect to it. In Figure 11.14 a layout diagram of this setup can be seen. For this reason, the low-level interface software is sometimes also referred to as the *server software*, for example in the source code.

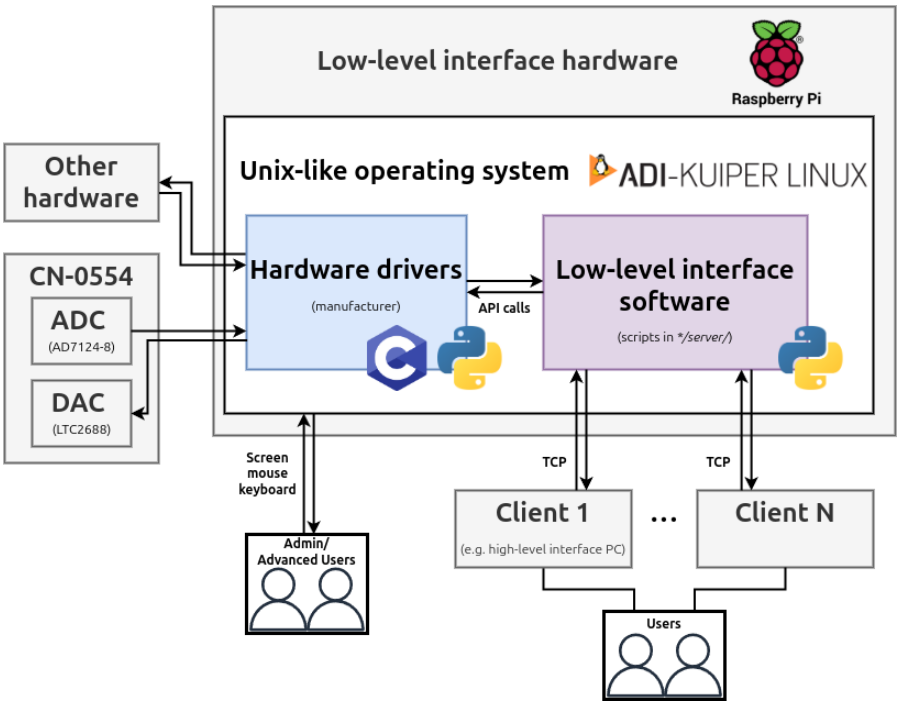


Figure 11.14: Abstract layout of the low-level interface. The interface software runs inside an Unix-like operating system environment on a Raspberry Pi 4. It functions as a TCP server, hosting a unique TCP socket for each client device connected to it. For advanced users, the OS that runs the low-level interface can be directly interacted with by attaching a screen, keyboard, and mouse to the Raspberry Pi 4, which is useful for debugging and detailed configuration.

The TCP connection is not restricted to a single physical layer, so this connection can be established between server and client over an internet connection, a point-to-point ethernet connection, the Wifi transponder of the Raspberry Pi, etc. The Raspberry Pi can also run a second script that functions as

a client, and then connect to the server by through the localhost address (`127.0.0.1`). Because the Raspberry Pi is a fully-fledged computer with an Unix-like OS, the server does not have function purely as a headless device; if one connects a screen, keyboard, and mouse to the Raspberry Pi, the OS can be interacted with like any other computer.

This has a number of important advantages: first, it makes debugging[2] and configuration[3] much more straightforward, and greatly facilitates further development by third parties. It also simplifies the server code, because it is not necessary to facilitate complete server access via the TCP connection. This last point also improves the cyber security of the system, because the server can be set up such that important properties of the server are only accessible via physical access to the Raspberry Pi.

Let us look at the actual low-level interface software. The script of primary interest is `*/server/server.py`, which defines a number of functions and classes and also operates as the entry point if this Python file is executed as `__main__()`. When the server script is executed, first the configuration file is loaded and applied, and a number of class objects are instantiated. Most of these classes are separate threads that handle a specific set of tasks. For example, there is a thread that periodically polls the ADC hardware and stores the values to a datapool. In total, there are five such threads:

1. **Main thread**: This is the main thread that runs the Python program. It is the first to be created, and the last one to fold.
2. **Socket server thread**: This thread facilitates communication over the TCP connection to the high-level interface. It hosts one parallel socket instance for each client connected to it.
3. **Control thread**: Used for managing, staging, and playing schedules.
4. **ADC thread**: Acquires data from the ADC and writes it to the datapool.
5. **DAC thread**: Instructs the DAC given a set of operational rules and information from the datapool.

The general functional flow of the server script can be seen in Figure 11.15.

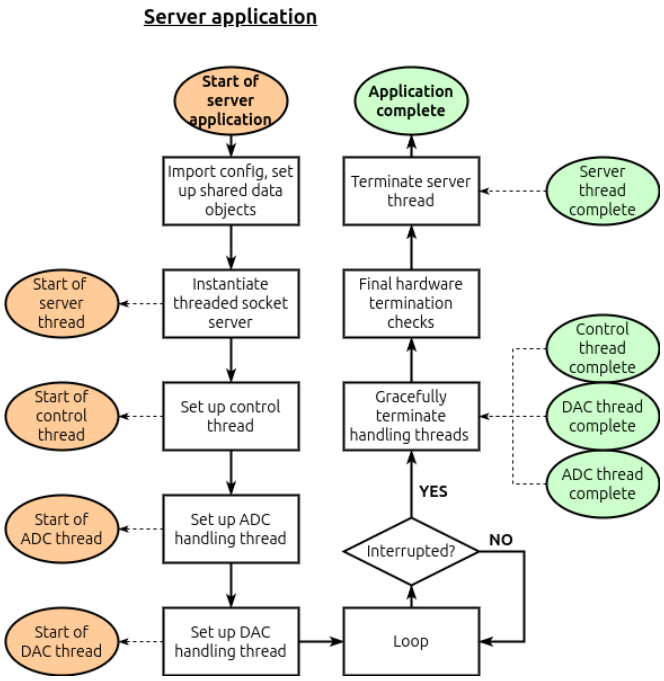


Figure 11.15: Functional flow of the main thread of the server script. Detailed functional flow diagrams for each thread can be found in Appendix G

[2]The server script proactively logs to the terminal, which helps with debugging. In addition, there are tools such as `*/server/server_debug_GUI.py`, which runs parallel to the server and prints real-time data on the screen in a small GUI.

[3]See the file `*/server/server_config.py`

The main reason for setting up the server application in this segmented and threaded way is related to performance and organizational advantages. The Raspberry Pi has limited computational resources, but it does support threading. With the threading set up this way, separate functionalities implemented by the code can be run at separate loop rates without much extra overhead.

For example: one of the threads polls the ADC hardware for current measurement values. Not exactly every acquired datapoint has to be sent over the TCP connection; it only has to do so when a request is sent for it over the TCP connection. This way, the ADC thread could be run at higher loop rates, for example to get more accurate data by filtering the data using a moving average, without having to run the socket code for the TCP connection unnecessarily frequently. To be fair, this could also be implemented without threading, by instead putting everything in a high frequency Digital Signal Processing (DSP) loop optimized with conditionals, where every action happens at its own given division of the DSP loop rate. Although the latter is not explicitly an important disadvantage, threading here seems to provide a more true compartmentalization between the functionalities, so it was preferred for this application.

Variables in the datapool that can be accessed by multiple threads are generally protected by *locks*, a concept from the Python library `threading`. These locks function similar to the concept of a *mutex*, which prevents a thread from accessing the variable if it is already being accessed by another thread. It is important that locks are used, because failing to do so can result in threading-related race conditions, which can be particularly stochastic and difficult to debug. Due to the way that the "threads" from the `threading` Python library are run by the processor of the Raspberry Pi, they probably are not needed, but it does not hurt and ensures that the baseline code will not break if the code is implemented using hardware multi-threading later.

The socket server thread uses the Python library `TCPServer` in conjunction with `ThreadingMixIn` to facilitate communication over the TCP connection. It also uses the `blake2b` library for schedule hashing. Importantly, it uses the same internal library for the communication codec as the high-level interface software, meaning that so long as the latest version of the source code is used on both low-level and high-level interface, they should be able to encode/decode correctly.

The TCP connection is set up such that the Raspberry Pi is the **server**, supporting multiple clients to be connected to it. The high-level interface is a **client**, and communication is set up such that the client requests something from the server, and may receive something back in response. This could be a request for data, a ping, a command instruction, etc. It is not necessary for the server to send something back, but the server is set up such that most client requests will result in some form of acknowledgment, which helps with error detection and debugging the software.

We will conclude the discussion of the low-level interface software here. More detailed information can be found in Appendix G and the source code [74].

11.6. TCP communication protocol

We have discussed how the low-level interface running on the Raspberry Pi communicates with the high-level interface running on a computer elsewhere. What has not yet been discussed in detail is how these entities communicate to each other. In this section, we will take a closer look at how this works. Looking back at Figure 11.1, we concern ourselves with the object indicated in magenta.

Much of the interface software was built from the ground up, which means that information about what data had to be communicated was already available. There was also a desire to ensure that the communication protocol would perform well: it would be a shame if the system's performance were to be bottle-necked by the data connection between the two interfaces. For these reasons, it made sense to design a custom implementation of a communication protocol that would be tailored to the needs of the system.

Very early on, an implementation based on user datagram protocol (UDP) was considered, but this was quickly replaced by TCP because this offered certain benefits. There was no need to re-invent the wheel to the point where handshakes required a custom implementation, so TCP offered a superior communication scheme to start from. The ability of TCP to be able to be configured as a server serving multiple clients also opened up some extra functionality, as we will see in a moment. Finally, TCP offers quite a lot of flexibility in regards to the physical layer. It can be implemented as a remote connection

over the internet, or using a physical cable, and in the latter case it is flexible enough to be implemented over Ethernet, universal serial bus (USB), etc.

Initally TCP was chosen with the assumption that the data throughput would be sufficient, because the required throughput was as of yet unknown. However, during the design of the system overall, the potential restrictions of this communication line were already taken into account.

For example, when the low-level interface is running a schedule on the Helmholtz cage hardware, it is important for real-time telemetry to be sent back to the high-level interface reasonably quickly, so that there are no problems with data desynchronization, and that a user who is monitoring the Helmholtz cage can react in a timely matter if anything goes wrong. For this reason, the software in the HCT is set up to pre-load the low-level interface with the schedule, so that each step does not have to be sent whilst the schedule is being executed, when it is best to have the line available as much as possible. This has the added advantage that the schedule can be error-corrected before it is executed, which in the case of the HCT is implemented by checksum matching.

The low-level interface also has a degree of autonomy baked into it; it runs the schedule by itself, only listening to relatively small commands from the high-level interface. It can do many things by itself without constantly being dependent on the high-level interface, which means that communication on the TCP connection is kept relatively minimal.

For these reasons, the communication line was designed from the start not to be loaded down too much. The next logical step was to serialize the data; that is, to package the data in such a way that overhead was minimal as well, to keep the communication protocol lean. The resulting implementation we will refer to as the **Simple Command Codec** (SCC). This codec describes how data is packaged to establish a two-way connection over TCP between low-level interface and high-level interface. Since Python is generally not known for its blisteringly fast performance, and the Raspberry Pi has limited resources as well, a deliberate effort was made to optimize this package structure. The following considerations were kept in mind for the codec:

1. The lower the overall encoding/decoding overhead, the more headroom is available for further development, and or the higher the operation rates are possible for the overall system.
2. Packages which contain time-critical information should ideally be small and fast to encode/decode.
3. The encode/decode functions of the codec should have no/minimal error checking, this should be implemented by the server/client API functions that call the codec.
4. The byte size of the package should be tailored to the contents, such that minimal padding is needed. For example, a package that stores a single boolean value should not be as large as a package that stores a 256-byte string.
5. The packet should have a single character type identifier as the first character.

As mentioned before, a further consideration was that the codec be usable by both client and server, such that only a single version of the codec would serve all needs (unless there are very good reasons not to do so). This was implemented as such, and version 1.0 of the HCT contains a single codec that both the low-level interface and the high-level interface refer to.

The SCC went through a total of four iterations. At first, design was very conservative and packets were designed to have a variable packet size, in order to minimize the amount of padding. Due to some technical reasons with the Python TCP frameworks that were being used, it became disadvantageous to use a variable package size. After taking measurements, it became clear that in this case encoding and decoding time have a much more profound impact on transmission delay than the package size. Consequently, the packet size was for all packets was fixed at 256 bytes, and packet types distinguished themselves by their encoding/decoding speed.

Iteration 4 of SCC contains seven packet types in total:

- **Message packet** (M): These packets are meant for sending long string messages.
- **Echo packet** (E): This packet type is identical to the message packet, but it is meant to signal to the receiving socket to echo the contents back. This is mainly useful for troubleshooting the TCP socket connection itself, specifically testing latency (pinging) and signal integrity. Because execution

packets have a much higher encoding and decoding overhead, echo packets lend themselves much better to the aforementioned purposes.

- **Control packet** (C): Packet meant to control the power supplies to be set to a certain current output proportional to a magnetic field strength. Packet contains 3 signed <float> values in μT.
- **Telemetry packet** (T): Packet containing telemetry in the form of many measured parameters. Mainly used to get as much useful data from the server to a client in a single package. Contains a 20 byte UNIX timestamp, commanded step index, and tri-axial values of measured current $\overline{I}_m$, measured field strength $\overline{B}_m$, and commanded field strength $\overline{B}_c$.
- **B-sample packet** (B): Packet containing a single measurement of the tri-axial magnetometer. Contains a 20 byte UNIX timestamp and 16 byte signed float value in μT for each component of $\overline{B}_m$. B-packets are preferred over T-packets if only measured field strength data is needed and speed is critical, as the encoding/decoding overhead of B-packets is much lower than that of T-packets.
- **Execution packet** (X): Packets meant for sending commands, taking a <string> command name, and then the number of appended arguments. The codec keeps track of the type of each argument, with the following types currently supported: <string>, <int>, <float>, <bool>.
- **B-schedule segment packet** (S): Packets containing a single line of a B-schedule, which can be used to send a complete B-schedule between client and server. Contains the segment number, the total number of segments in the schedule, the segment's time, and three $\overline{B}_c$ components.

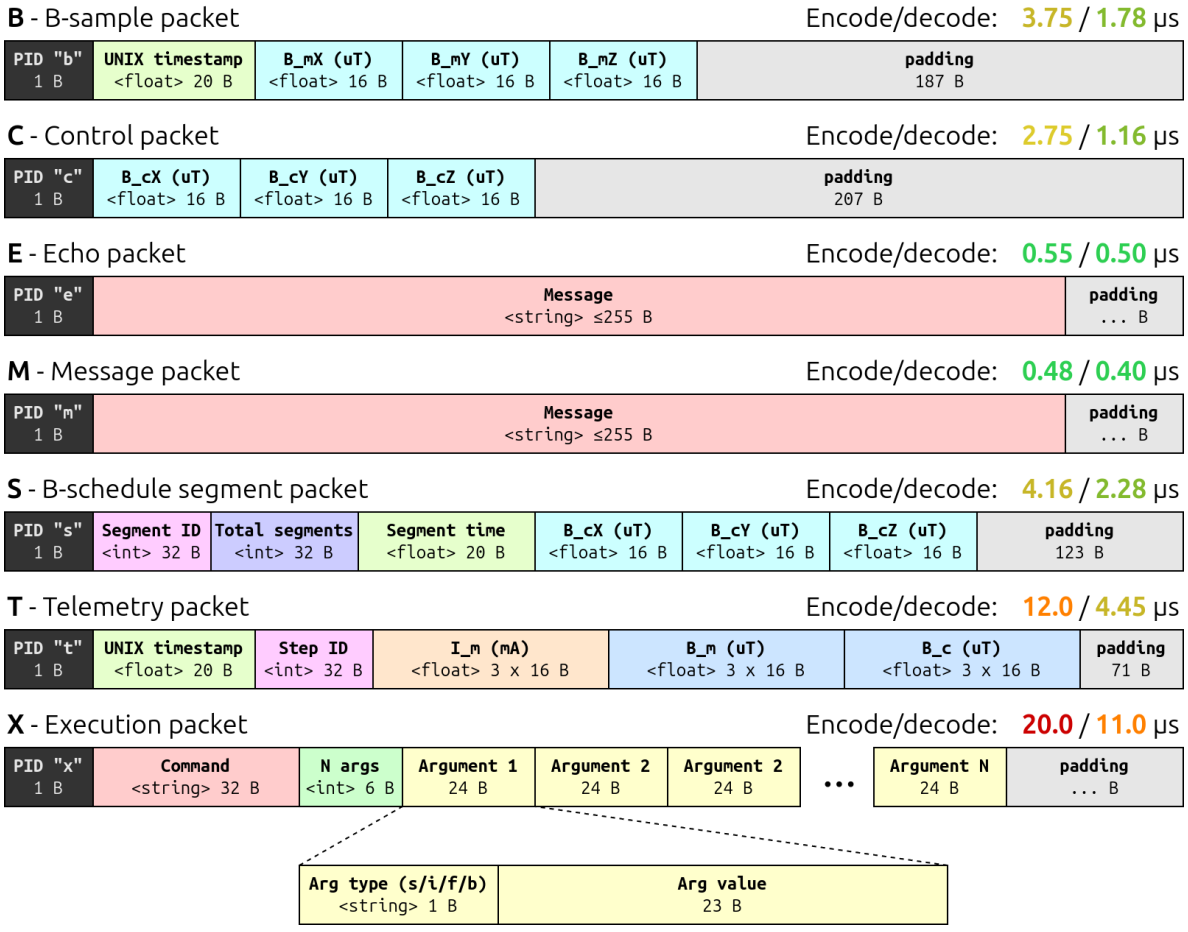


Figure 11.16: Anatomy of all packet types in the 4th iteration of the Simple Command Codec

Every packet starts with the packet ID, which is a single ASCII character which allows the codec to quickly identify the packet type, without having to read the whole packet. Therefore, a total of 255 bytes are left for the payload of the packet. If the size of an encoded packet does not add up to 256 bytes,

the codec will pad it with a padding character until it does reach that size. Again, this is because the used Python TCP frameworks like to receive packets of a fixed size.

The contents of any packets are serialized by the codec to a stream of characters. Any numeric values therefore have to be typecast to a string to allow them to be sent. Arranging this is where most of the overhead of the codec comes from. A significant amount of attention was spent on optimizing the codec to ensure that the codec would never be bottle-necking the communication line, or hogging unnecessary resources on the Raspberry Pi. By the end, most encode/decode operations are of the order of 5 microseconds or less, with the notable exception of telemetry packets and execution packets. The latter of these is unlikely to be sent at high frequency, so these are not much of a worry. The load of telemetry packets was benchmarked and found to be sufficient for the system to work adequately.

An overview of the anatomy of these packets, as well as an indication of their relative encoding time can be found in Figure 11.16. The encoding/decoding performance was benchmarked on an FX-8350 CPU, whose architecture is almost fifteen years old at this point. One may therefore expect performance on modern processors to be faster, but keep note that the codec's performance may be somewhat worse on the Raspberry Pi processor. As such, the values in Figure 11.16 should be viewed as an indication of the relative difference between different packets, but should not be taken at face value to predict overall system performance.

More information about the Simple Command Codec, as well as its complete source code, can be found in section G.2.

11.7. Software testing strategies

Because of the highly iterative nature of the software development process, software testing was not done in a single instance. Furthermore, the type of testing that was appropriate for a certain piece of software was not always appropriate for another, so multiple testing approaches were used across the whole code base. In general, the object-oriented structuring of the code facilitated working on a small isolated piece of code, which could then be tested straight away. This was not only useful for testing, but also optimization of code. In addition, software testing was greatly simplified by the elaborate graphs and other visualization tools that were developed as part of the HCT; for example, transformation matrices that are calculated wrong very easily show up in plots, because they will rotate differently than intended. In this way, both verification and validation could be performed, since it did not only test the software implementation, but also the overarching methodology.

Here is a brief overview of the software testing strategies used during development:

1. In general, every block of code was tested during development before being committed to the rest of the codebase.
2. The SCC codec has an automated testing script used for verification and benchmarking (see Figure 11.18).
3. The low-level interface software can be tested through an automated test script that connects to it as a TCP client (see Figure 11.17)).
4. The low-level interface can also be tested locally using the terminal logs and tools such as `*/tests/server_debug_GUI.py`.
5. The high-level interface was primarily tested with unit tests during development, and then integrated into the GUI environment in such a way that aberrant behaviour could be easily spotted from the visualization tools.
6. The high-level and low-level interfaces can be tested separately; the low-level interface can be tested on-site using a custom script on the Raspberry Pi that acts as a client and connects via `localhost`. The high-level interface can be tested anywhere using a dummy server running on the computer that emulates the server's behaviour. There are some extra commands available that can be sent to this dummy server, such as the ability to loop back incoming commands as sensor output.
7. The *Orbitals* generator outputs were additionally verified on a small number of case studies based on typical CubeSat orbital parameters.

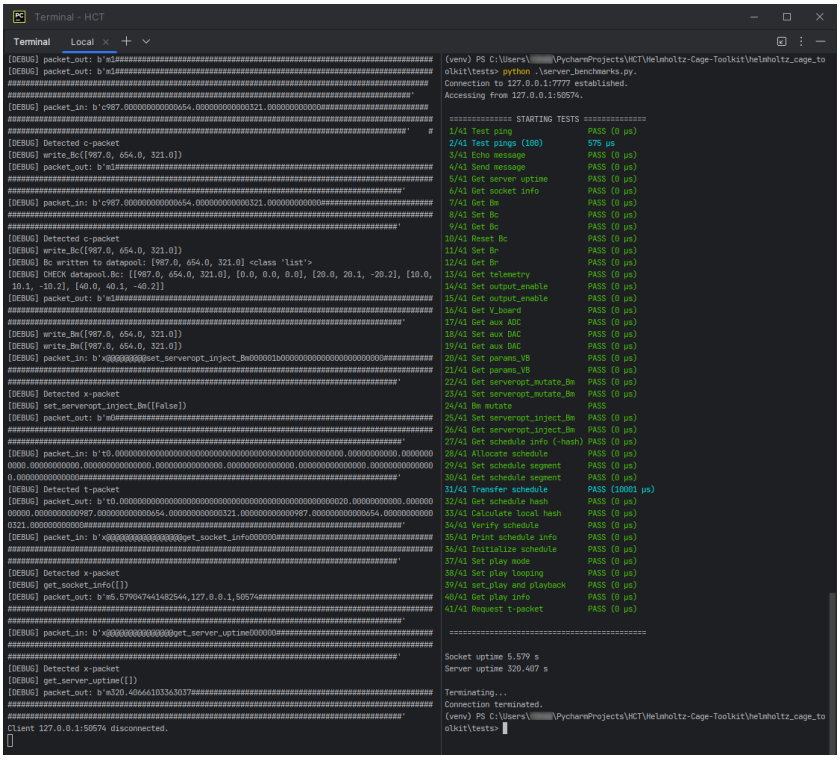


Figure 11.17: Automated software testing of the low-level interface software: on the left, the debug terminal of the low-level interface code can be seen, including raw packet information. On the right is the output of `*/tests/server_benchmarks.py`, a 900+ line testing script that connects to the server as a client and checks many functionalities.

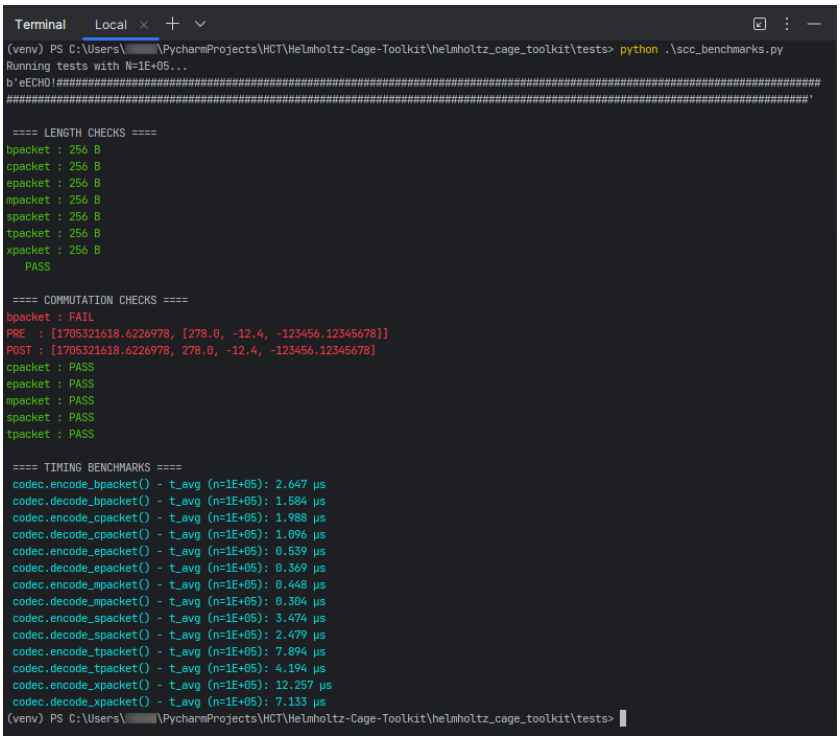


Figure 11.18: Automated testing and benchmarking of the SCC codec using `*/tests/scc_benchmarks.py`, here with 10^5 iterations.

Another aspect of the testing strategy is the way in which the code is made public: it is fully open-source and published on GitHub, where other users can easily communicate problems and even contribute to bug fixes via pull requests. This means that problems with the software that are discovered in the future can also be addressed effectively.

11.8. General file layout

We will not discuss the exact inner workings of the HCT in more detail than we have already done. There is simply too much to discuss and such a discussion is not relevant enough for the maintext of this thesis. For more detailed information, the reader is referred to the HCT source code [74] and the surrounding documentation.

However, in this section we will give a brief overview of the files contained in the HCT to get the interested reader started, as the code is spread out across many files. A lot of relevant information can be found in Figure 11.19. For example, all the files marked with red contain classes and functions that can be accessed as an API, so if your interest is in writing your own scripts using the HCT, these files are of particular interest.

One can also see in Figure 11.19 how the GUI application code is structured in broad terms: upon launch, a main window is created first, and each tab is placed in this as a separate window. Each of these windows contain particular widgets, such as the elaborate visualizations, which are often written in separate files. Finally, there are a number of automated tests available, notably for the server and the communication protocol, as well as a few example scripts to show how connection to the TCP connection works.

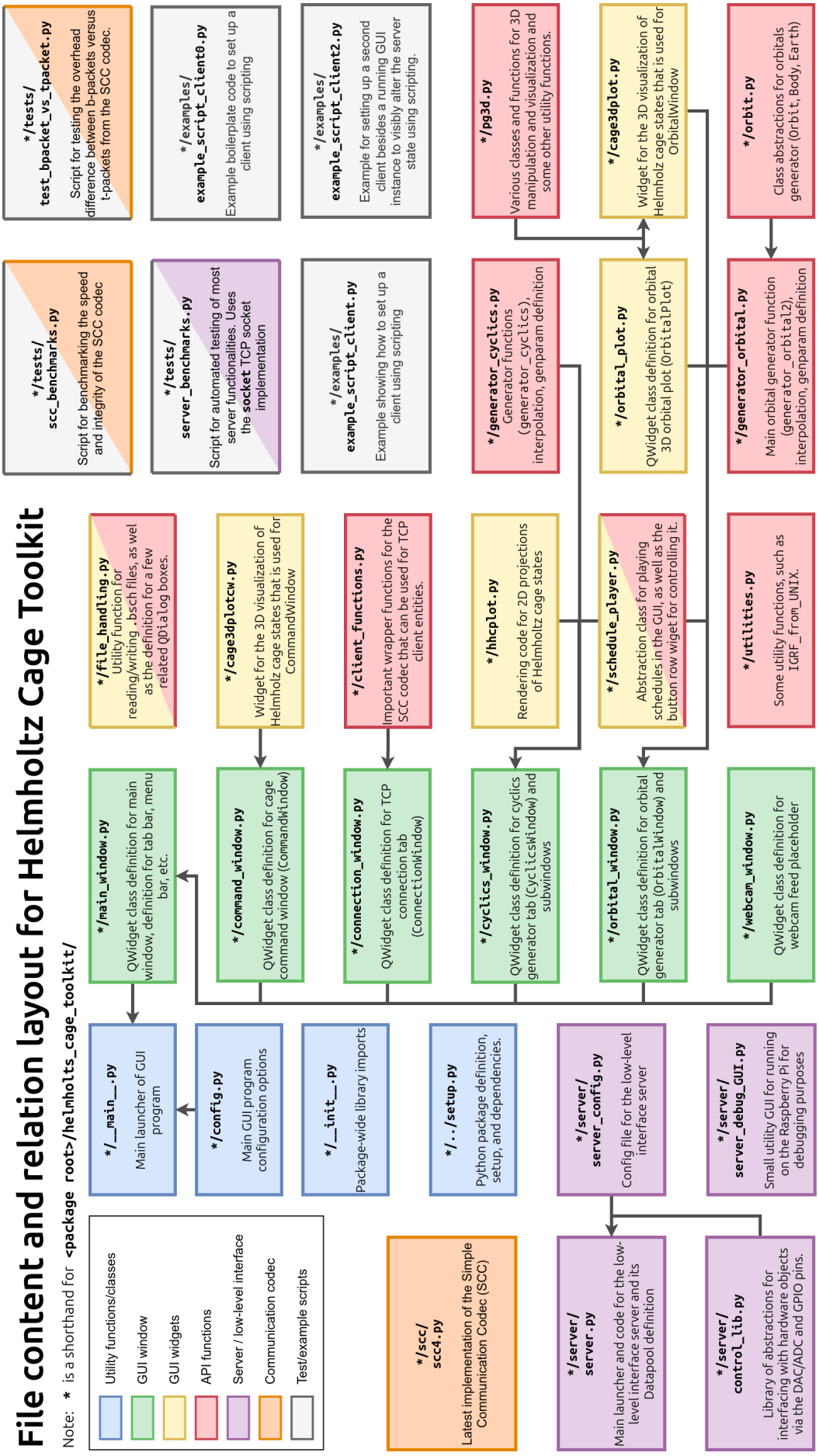


Figure 11.19: [SIDEWAYS] An overview of the layout, dependency relation, and general content of the files inside the Helmholtz Cage Toolkit package.

12

System Integration

After the efforts of designing and testing the companion board and writing most of the software in the Helmholtz Cage Toolkit (HCT), the time had come to perform system integration and connect the parts together into a whole. Truthfully however, the line between design, assembly, and integration was rather blurred. For example, before the designs for V2.2 of the companion boards were finalized, the power supplies, companion board, and part of the Helmholtz cage was connected (at which point it was discovered that the intended power supplies would not work).

Still, there was a point at which every part was ready to move out of the workshop and into the cleanroom. This was shortly after V2.2 of the companion board was assembled and tested. The integration that followed was a relatively brief process, and therefore this chapter will be rather brief also.

The milligauss meter, the companion board, and low-level interface were all placed on a small wheeled table, which would help with integration before moving everything into an enclosure. A special platform was 3D printed so that the companion board and the Raspberry Pi could be positioned in a fixed relative position to one another, and then clamped down to the surface.

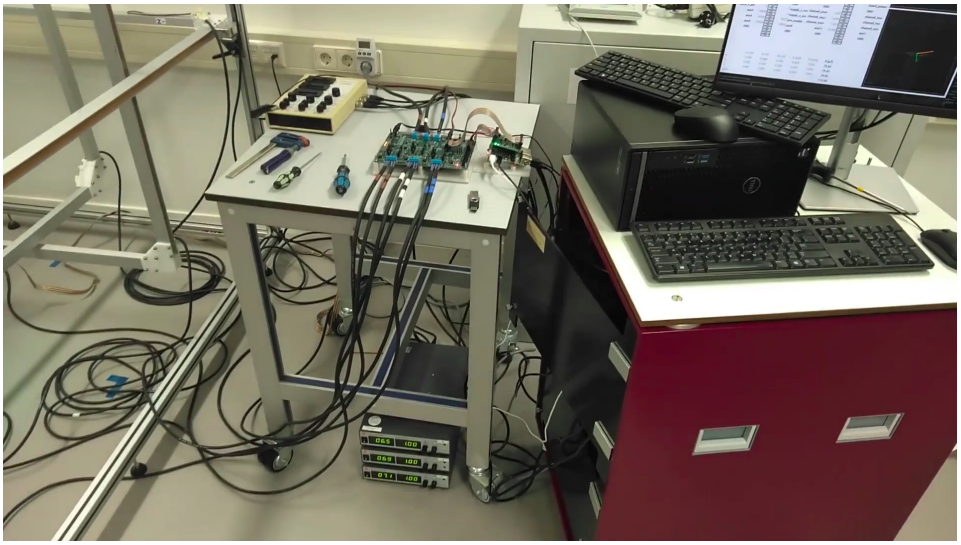


Figure 12.1: Early stages of integration with all system components connected, though left exposed for testing.

As can be seen in Figure 12.1, this was set up right next to the Helmholtz cage and the supply cart, in which the components would later be housed. The high-level interface computer is placed on top of the supply cart, a replacement for the old computer that was originally there. There are two sets of keyboards and mice; one set is for controlling the low-level interface Raspberry Pi, and the other set is

for the high-level interface computer. Both devices are connected to the same screen, so that the screen can be used for both by simply changing the input source.

The power supplies were left outside their rack mount early on, to allow for better access, but were later mounted so that the cable management could be improved. One test performed during integration was verifying the coil resistances once again, since the cables connecting them to the system had been shortened. This can be seen in Figure 12.2.



Figure 12.2: Pushing a constant current of 1 *A* through each pair of coils reveals the series resistance of the coil and the cabling in Ohms. From top to bottom: X, Y, Z.

When all components were positioned, they could be connected. This was done with deliberation, as this process was very helpful for figuring out the cable routing (since no formal harness design was made beforehand). Importantly, the high-current cables running to and from the companion board were labeled so they could not be confused. The label convention that was used can be found in Figure 12.3.

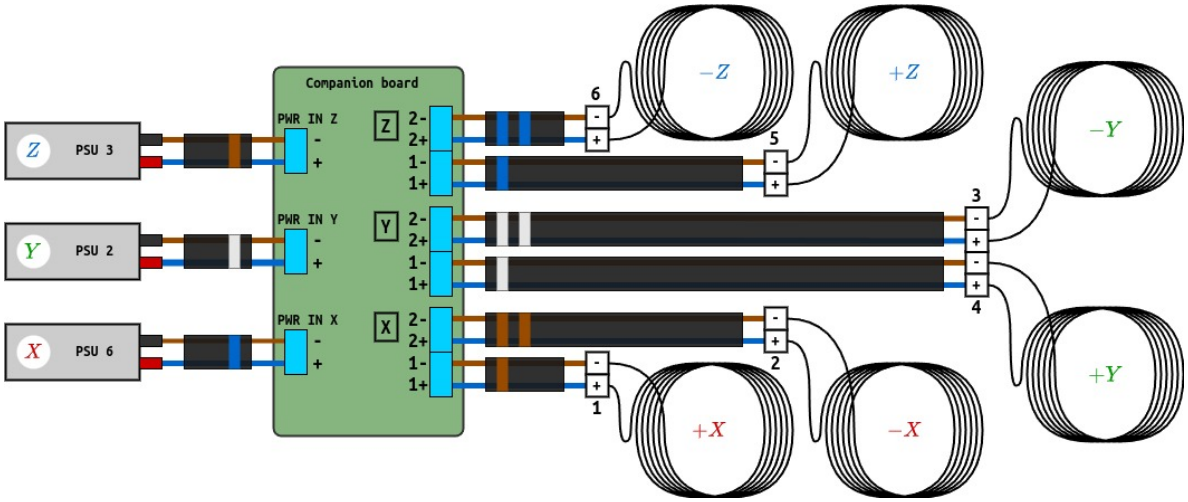


Figure 12.3: Cable connection and label convention used during integration. Black cables have a blue and brown line each, brown is typically used as negative. Three different colours of tape are used; brown for X, white for Y, blue for Z, with two tapes denoting the negative coil of the pair.

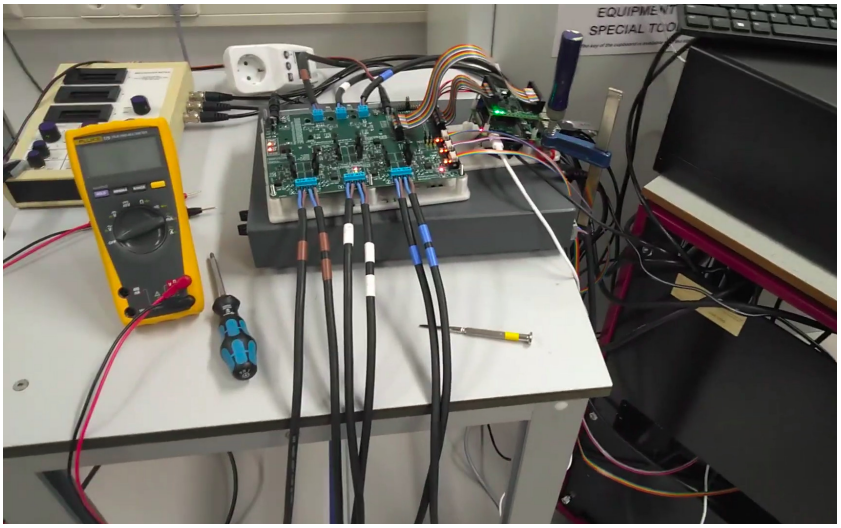


Figure 12.4: Detail of cable labeling for the high-current coil cables.

After this, all the other cables could be connected one by one. The control ribbon cables for the power supplies, the ADC and DAC ribbon cables, and the cables for the analog magnetometer readings. Finally, the power jack connectors for the board and magnetometer were connected, and basic function testing could begin. A multimeter was always kept nearby to test for continuity, open circuit, or measure voltage, so that integration would remain safe (Figure 12.4).

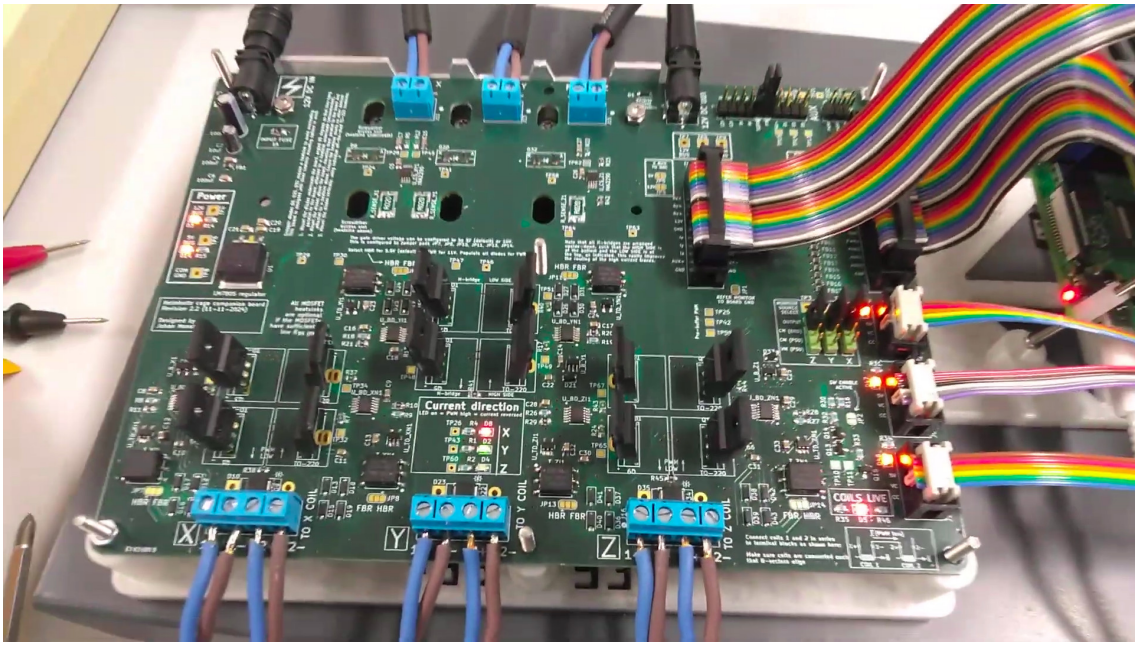


Figure 12.5: Detail of cable labeling for the high-current coil cables.

After basic functionality of the setup was demonstrated, stress tests could commence. One of the greatest unknowns was be the temperature of the board under high current load, since this was difficult to estimate beforehand. The current was increased across a series of tests, until at the final test, all three power supplies were steadily delivering 4 A into the coils until steady-state temperatures were observed. During this test, the diode heatsink reached a temperature of about 65 degrees, which implies a diode junction temperature of 90 degrees[1], which is perfectly acceptable. If more cooling is needed, a fan

[1]The STTH8L06 diodes have a rated maximum operational junction temperature of 175 degrees Celsius

could be installed, or the heatsink could be increased in mass. Temperature tests were performed with a hand-held thermocouple thermometer and a FLIR infrared camera (Figure 12.6).

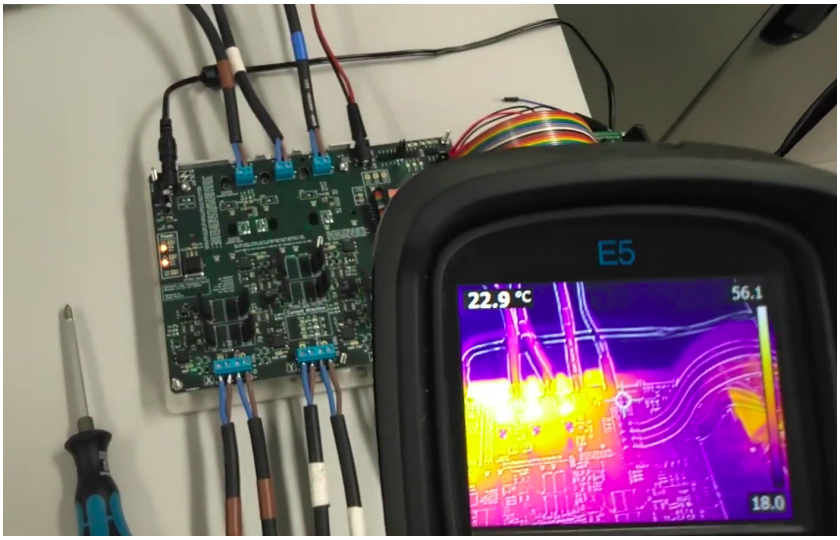


Figure 12.6: Observing companion board temperatures with an infrared camera during a high current load test.

After the limits of the system had been explored to satisfaction, it was time to start instructing the system using the low-level interface. To this end, a debug application[2] was used that could be run on the Raspberry Pi that allows basic control and data readout.

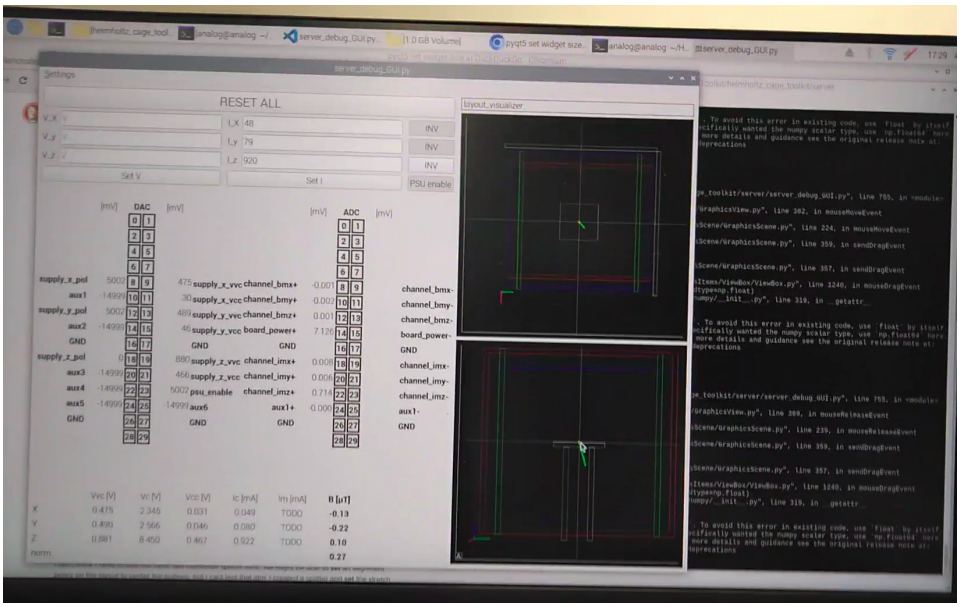


Figure 12.7: Using `server_debug_gui.py` to test the Helmholtz cage from the low-level interface.

First, fixed magnetic vectors of different magnitude and direction were actuated in the Helmholtz cage, so that the expected performance could be compared to the actual readout. This provided useful input for calibrating the system. Then, a test was carried out where the local magnetic field vector was nullified by hand, creating a magnetically neutral environment at the centre of the Helmholtz cage.

Lastly, a number of basic schedules were used to automate the vector and test the speed and slew

[2] It is available in the source code as `*/server/server_debug_gui.py`

characteristics of the system. A basic XY circle was used for this purpose, which was slowly increased in speed. Inside the Helmholtz cage, a compass was placed, so that the coil activity could also be seen physically. The rotation speed was ramped up to 20 rpm to verify requirement **PER-03**, and then up to 120 rpm just to see whether the system could handle it. The system could track the frequency reasonably well, but the amplitude of the input waveform became somewhat distorted: the magnitude of the vector would falter, creating erratic and uneven ellipses rather than nice circles.

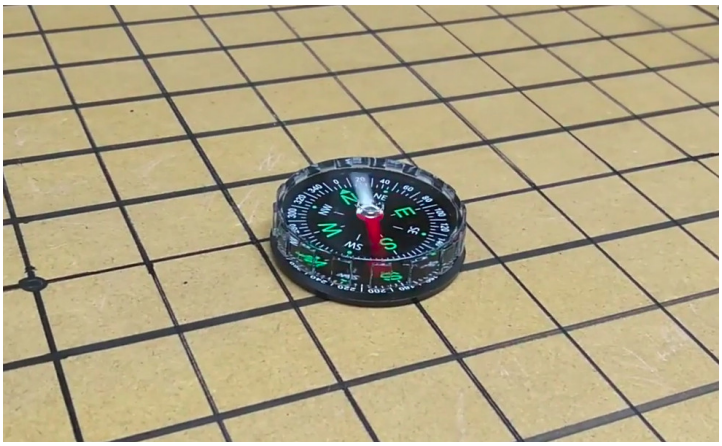


Figure 12.8: Using the Helmholtz cage to rotate a compass needle at 120 rpm (2 Hz) during a stress test. Note the motion blur on the needle.

13

Verification & Validation

We will now move onto going through the requirements for the Demonstrator System (DS) that were formulated in section 8.1, with the aim of seeing to what extent these requirements have been met. Each requirement will be discussed individually first. Finally, in section 13.5, a summarized overview of all requirements will be given, which is the formal verification matrix.

Some requirements have been adjusted as part of the verification process; for example, the fact that at a relatively late design stage the power supplies had to be changed meant that some requirements could not be achieved. Any modified requirements have an explicit discussion explaining the change, and the requirement designator has been updated in correspondence.

13.1. Top-level Requirements

DS-01-01	**The DS shall have a magnetic control volume large enough to envelop a typical CubeSat. The size of this control volume shall be a cube with side lengths of at least 415 mm.**	A

Back in section section 6.2, we discussed the concept of the control volume as being a cube in the middle of the Helmholtz cage in which the field homogeneity does not deviate more than 1% from the centre value. The rationale for requiring this cube to be 415 mm is so that a 2x2x3 unit CubeSat would be able to fit inside control volume in any orientation. There is no strong rationale for the 1% deviation figure; it was a convenient performance target that was low enough so that it would not hamstring any magnetic testing.

In figure Figure 6.14 from section 6.2, the results of numerical simulation show that the distribution of the homogeneity is not a simple square, but rather a more star-shaped pattern. To guarantee that the field homogeneity is in a simple value, the outline of the control volume must lie inside this star-shaped pattern. We now focus our attention on the pair of X-coils, because coil pair is the smallest, exhibiting more field variation in the same space as the larger Y- and Z-coil pairs. This means that if the control volume fits in the homogeneity distribution of the X-coils, it will also fit in that of the other coil pairs (assuming consistent manufacturing quality).

This has been plotted on the left side of Figure 13.1, in which the magnetic field of the X-coils has been simulated. All values that deviate more than 1% from the centre value have been discarded. The white square in the middle represents a cross-section of the 415 mm cube that **DS-01** requires. One can see that at the very corners of the control volume, the deviation exceeds 1%. The right of Figure 13.1 shows the same but for a 2% deviation. In this case, the control volume of 415 mm does fit inside.

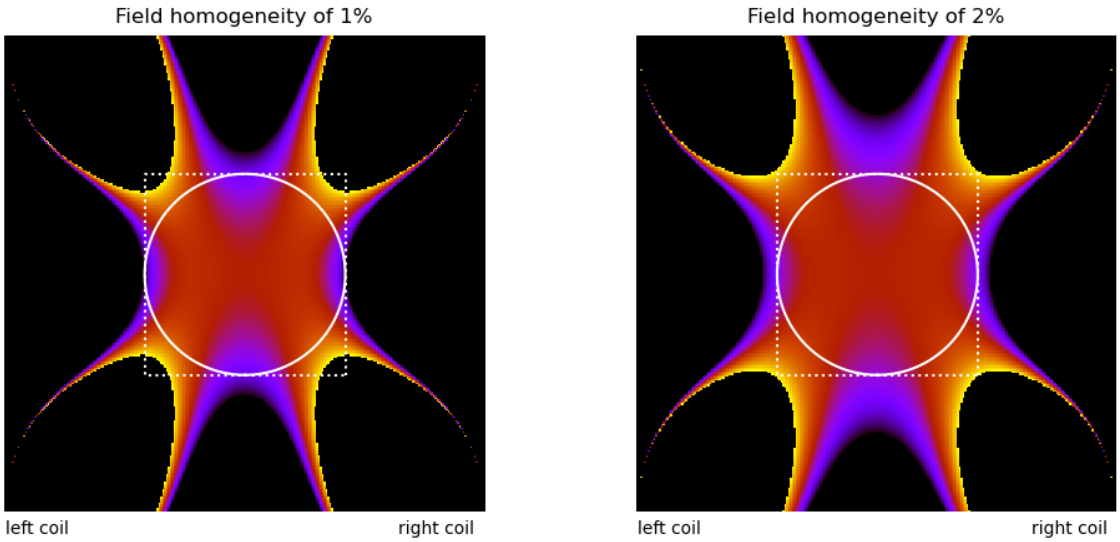


Figure 13.1: Simulated magnetic field strength for the pair of X-coils, for a centre field strength of 250 μT. The two coils are oriented vertically. On the left, all field values with a deviation from the centre value of 1% or more have been discarded, and the 415 mm outline of the control volume has been superimposed in white. On the right, the same has been done, but for a 2% deviation limit. Refer to section 6.2 for more details.

So strictly, requirement **DS-01-01** has not been achieved, unless either the 1% homogeneity restriction is raised to 2% or the control volume is shrunk down. However, there is a third option: turn the control surface into a spheroid. If the rationale is that a 12U satellite with a 412 mm diagonal has to stay inside the control volume in any orientation, then using a sphere is the more sensible choice to begin with. The cross-section of such a spherical control volume can also be seen in Figure 13.1.

Therefore, we will formally change requirement **DS-01** so that the control volume is a spherical volume with a diameter of 415 mm:

DS-01-02 **The DS shall have a magnetic control volume large enough to envelop a typical CubeSat. The size of this control volume shall be a sphere with a diameter of at least 415 mm.**

Given this adjustment, **DS-01-02** receives a **PASS**.

Finally, it should be noted that users that desire higher accuracy may correct for the local variations inside the control volume using post-processing. To do this, one can simulate the homogeneity variation around the centre of the Helmholtz cage, generate a calibration manifold, and subtract this from the data after the test. A simple 3D grid evaluation of the Biot-Savart law as was done in section 6.2 may be sufficient to generate such a calibration manifold.

DS-02-01	**The DS shall establish a neutral magnetic environment, reducing any ambient magnetic fields inside the control volume down to below 500 nT in any direction.**	**A**

If one assumes that (a) to cancel the local Earth magnetic field (EMF). a field strength of approximately 50 μT is required and (b) within the control volume, the field deviation may be as high as 1% from the centre value (**DS-01**), one can expect a deviation of at most 500 nT anywhere inside the control volume. Therefore, we can consider requirement **DS-02-01 PASSED**.

DS-03-01	**Inside the control volume, the magnetic field strength uniformity shall be within 1%.**	**A, T**
DS-04-01	**Inside the control volume, the magnetic field directional uniformity shall be within 1%.**	**A, T**

Due to the similarity of these two requirements, we will discuss them together. The properties of the Helmholtz cage itself allow for these requirements to be passed. However, the DS itself is in an

environment that contains local magnetic disturbance. Whilst this disturbance vector is relatively static as a function of time, it has been shown to exhibit significant variation in both magnitude and angle as a function of distance from the centre of the Helmholtz cage.

Taking the findings from Figure 7.10 and Table 7.1, on the XY-plane alone, one may expect a field strength deviation in the control volume as high as:

$$\frac{0.5 \cdot 0.451}{0.707} \cdot 8.22\% = 2.6\%$$

For the angular deviation, this may be as high as $0.451 \cdot 17.4° = 3.9°$. Whilst the system hardware passes the requirements, it fails them when placed at the current location. In chapter 7, many potential sources were investigated, but the source of the disturbance could not be avoided, meaning that **DS-03** and **DS-04** cannot be cleared by avoiding the sources themselves.

Besides moving the setup to a better location, another strategy would be data filtering. Since the local field variations seem relatively stable, one could perform a series of magnetic measurements around the control fields. With this data one can generate a calibration manifold, essentially a 3D map of how the local disturbance varies as a function of control volume position. After performing a test, the data can be offset or corrected using this calibration manifold.

The reason that such a calibration manifold was not produced as part of the thesis was because of the work involved: it requires taking many measurements, which have to be long to average out other variations in the disturbance (e.g. nearby elevators). Then one has to assess the stability of these values across time. It was therefore de-scoped from the thesis and left for further research.

Still, it is defensible that if the system is properly calibrated in the future, both **DS-03** and **DS-04** can be achieved. In additions, the deviations are also still below the typical pointing errors of a small CubeSat (10% as a rule of thumb), meaning that in practical terms, these disturbance variations may not even be especially noticeable. For example, small nanosatellites such as Delfi-PQ, which has a diagonal of 134 mm, for which the magnitude and angular deviations become 1.65% and 1.17% respectively.

We will therefore consider **DS-03** and **DS-04** **CONDITIONALLY PASSED**, with the caveat that to achieve them, a calibration of the local environment must be performed first.

DS-05-01	**The DS shall allow the satellite to be suspended inside the control volume.**	**D**

Requirement **DS-05** receives a **PASS** for the following reasons:

1. There is a grid table to position a device-under-test within the control volume, which has no measurable magnetic interference (see Table 7.2).
2. There is a suspension jig above the control volume to suspend a device-under-test from a wire (see Figure K.21). The length of this wire can be extended further if removing a ceiling panel and mounting an attachment point there.
3. Underneath the control volume, there is at least 1 m^2 of clear floor space available for additional mounting or suspension hardware (with the coils at their optimal spacing).

DS-14-01	**The DS shall be able to operate automatically according to a pre-programmed schedule, without requiring manual user input during testing.**	**D**

PASS - This has been implemented at a system level and both the system hardware and software reflect it.

DS-15-01	**The DS shall have a straightforward and convenient interface, suitable to be usable by space engineering students.**	**D**

PASS - At a system level, the user interface has been separated from the operational hardware. The software is designed such that both a easily-grasped graphical user interface (GUI) is available, as well

as a programming interface into the underlying functions. This interface can be used from the dedicated computer next to the Helmholtz cage, or by remotely connecting to the TCP server inside the low-level interface computer.

DS-16-01	**The DS shall be safe to operate, by users unfamiliar with the inner workings of the system and at the level of engineering students.**	**D**

PASS - Since the more safety-critical hardware functionality and coil control is handled by the low-level interface, and inexperienced end users will work with the GUI on the high-level interface, there is very little risk of bodily harm when working with the system. Appropriate safety checks are implemented in hardware (companion board and other built-in protection) and software (low-level interface config file, programmed safety checks, etc.). Inexperienced users can additionally experiment and play around in the GUI without ever sending instructions to the Helmholtz cage, so they can to an extent train themselves without risk.

DS-17-01	**The DS shall facilitate time-efficient testing, allowing for tests to be performed in the span of half a day, wherever feasible.**	**D**

PASS - It can be argued that this is very plausible with the current system, due to the following:

1. System calibration does not have to be performed for each test, since the low-level interface can store and use its own calibration data.
2. Local magnetic field deviations can be measured and cancelled at the press of a button using the GUI.
3. The low-level interface can be set to play back pre-programmed schedules automatically during a test, meaning that these schedules can be created and checked beforehand, without requiring access to the system hardware. This means this can be scheduled beforehand, leaving more time for the actual test activities whilst at the test location.
4. Existing schedules can be reloaded and edited, including their generation parameters, meaning that on-the-fly adjustments can be made quickly.
5. The GUI application provides good user feedback about what the Helmholtz cage is doing, allowing for problems or faults to be spotted early, which is likely to save time in the long run.

DS-18-01	**The total cost of the DS shall not exceed €10k.**	**A**

In Appendix A, the total raw construction cost of the DS was estimated to be €12,250 euros. Although this is certainly close to the target, it does exceed the €10k target by a good 22.5% which is not insignificant.

Either the requirement is **FAILED** and the system should be re-examined to lower the cost, or the requirement is reassessed and adjusted in accordance. Due to the way that cost has been taken into account constantly during the development process, the chances that the system may become substantially cheaper whilst maintaining its specifications are not very high.

The reason the cost target was set to €10k in the original requirement was that this is 5% percent of a hypothetical €200k CubeSat budget. It was reasoned that 5% of the budget could justifiably be allocated for magnetic testing. If the system costs €12.25k instead, this is now 6.1% of that budget.

Although it does not help the overall goal of the thesis, namely making a convincing case that magnetic testing of nanosatellites can be affordable, we will adjust the requirement to increase the cost. For missions on a small budget for which this is a significant increase in cost, it may be possible to lower the cost in a number of ways, if certain other performance targets do not need to be met.

For example, if the resolution of 1 nT for field measurement is not needed, and 100 nT is considered acceptable fine, then a much cheaper system magnetometer may be used. Also, the copper coil wire accounts for a third of this budget; what if this was replaced by cheaper aluminium cable? The resistance would be higher, meaning the wire gauge must be adjusted and a lower maximum magnetic field

strength is to be expected, but if these conditions are acceptable, this may reduce overall cost substantially. It may be the case that a desktop computer can be repurposed to serve as an interface computer, thereby nullifying its budget cost. These are a few examples where cost can be cut further, if this is needed. For more information about this, the reader is referred to Appendix A.

Regardless, requirement **DS-18** will be formally adjusted:

DS-18-02 **The total cost of the DS shall not exceed €12.5k.**

By virtue of the cost analysis Appendix A, this requirement now receives a **PASS**.

DS-19-01	**The DS shall be composed of COTS components with high availability wherever feasible.**	**D**

PASS - With the exception of the companion board and the coils themselves, every part of the system can be acquired as commercially-off-the-shelf (COTS). Furthermore, the full blueprint and bill of materials (BOM) of the companion board are publicly available, meaning that it can be ordered directly from a printed circuit board (PCB) manufacturer. Care was taken to allow for hardware variations (e.g. different H-bridge MOSFETs, different Raspberry Pi board variations, different power supplies, so long as they have analog CV control, etc.) meaning that different devices can potentially be substituted. It is hard to imagine how this system could rely much more on COTS that it already is.

13.2. Performance Requirements

PER-01-01	**The DS shall be able to simulate arbitrary field vector up to at least 210 μT inside the control volume, on top of the ambient field corrections.**	**T**

CONDITIONAL PASS - To verify this requirement, the Helmholtz cage was commanded to produce a 210 μT vector directly in opposition of the local EMF. This is because in this scenario, the Helmholtz cage will also have to nullify the local EMF on top of the aforementioned vector, and therefor is the most demanding on the power supplies. The system was demonstrated to produce well in excess of 210 μT in this situation. Another test was conducted with the Helmholtz cage nullifying the local disturbance rather than merely the EMF, with similar results.

During the verification process, it was noticed that the milligaussmeter used as a reference sensor seems to saturate near 240 μT. This is no surprise, given that its range is specified as [-2000, 2000] mG ([-200, 200] μT). Because it is difficult during normal operations to determine when and how the reference sensor starts behaving non-linearly, it is recommended to operate the Helmholtz cage within the range of [-200, 200] μT. This is why we give requirement **PER-01-01** a conditional pass.

PER-02-01	**The DS shall be able to correct for local magnetic field variations down to 500 nT**	**T**

CONDITIONAL PASS - This requirement be met because the noise in the environment, primarily nearby elevator movements, do not exceed 500 nT in magnitude. However, if this were not so, the system currently has no way to correct for this. A feature that was scoped from the thesis work was a reference sensor to measure this signal, and use it to correct the local disturbance at the Helmholtz cage in real-time. Whilst this would likely work (at least for some given bandwidth), this is not currently implemented. The requirement is therefore met in this case, but if the system were placed in a location with more substantial ambient noise, the requirement would not be met, hence the conditional pass.

PER-03-02	**The DS shall allow for rotating fields up to speeds of 20 rpm.**	**T**

PASS - After integration, the system could demonstrably rotate a compass needle up to 120 rpm, although at this point the measurements showed that the schedule waveforms had become quite distorted. At 20 rpm, no such distortion was observed.

PER-04-01	**The DS shall be able to determine the generated field strength at the control volume with a resolution of at least 10 nT.**	**D**
PER-05-01	**The DS shall be able to determine the field direction at the control volume.**	**D**

PASS - The DS incorporates the same AlphaLab milligaussmeter that was used for the measurements in chapter 7. This instrument can perform tri-axial field strength measurements with a resolution of 10 nT on the large range setting, and 1 nT on its small range setting. These values are read by a delta-sigma analog-to-digital converter (ADC) with a high bit-depth to not influence this resolution. The tri-axial results mean that also the direction of field can be resolved at this resolution.

PER-06-01	**The DS shall be able to monitor temperatures at critical locations.**	**D**

PASS - This requirement was formulated with the assumption that there may be locations in the test system that actively warranted temperature monitoring, particularly on the components carrying high current. However, after performing thermal testing, no such locations were identified, and the system was instead stress tested by measuring the temperature at various locations at high direct current load. All locations were within design specification.

The dropper diode were the hottest part of the companion board at 4 A, whose rudimentary heatsink reached temperatures around 65°C, implying a junction temperature around 90°C. These semiconductors are rated for a junction temperature of 175°C, so it is expected that these will reliably work at the 5 A system design rating. Board heating may become a problem, particularly for the current sensing section of the board. If this becomes a problem in the future, these problems can be easily addressed with a larger heatsink and a fan for active forced convection, both of which are amply facilitated by the design of the companion board.

PER-07-01	**The coil temperature in the DS shall not exceed** 60°C **during nominal operation.**	**T**

PASS - During a 4 A direct current load test, the temperature rise above ambient on the surface of the coils was so low it could not be resolved with a thermal camera. There was also no significant heating in the nearby aluminium structure.

PER-08-01	**The current direction through the coils must be able to be switched in under 1 ms.**	**A, T**

PASS - A major design requirement of the companion board was to implement fast polarity switching. Judging by the specifications of the H-bridge MOSFETs and dropper diodes, a switching time of around 75 ns should be possible. The H-bridge drivers are programmed with a 100$k\Omega$ dead time adjust resistor, which ensure a 75 ns dead time between switching. Therefore, in theory the switching time should be $3 \cdot 75 = 225$ ns. The results of performed transient tests (e.g. Figure 10.29) indicate that in practice, the switching time is of a similar order, even when switching at high current load.

13.3. Electronic Requirements

ELEC-01-01	**The design coil current shall be 5 A.**	**D**

The system was designed with a 5 A coil current in mind. At a late stage of development, the power supplies intended to drive the coils (BK Precision 1685B) turned out to be unsuitable due to undocumented stability issues. As a result, the original power supplies were swapped back in (Delta Elektronik ES-030-10). However, as discussed in chapter 6, these supplies are voltage limited and cannot drive the full 5 A through the load formed by the coils.

The reduction in current will in practical terms mean that the simulated fields cannot be as strong as intended. However, for the purposes of simulating the magnetic conditions for a satellite in orbit, the power supplies should easily be powerful enough. There is however not as much headroom to multiply the field strength by a factor, as originally intended.

We could list this requirement as a **CONDITIONAL FAILURE** of the original requirement, and

recommend a power supply upgrade in the future. However, after consultation with the stakeholder who specified this requirement (thesis supervisor), we will instead adjust the requirement to 4 A, as this value is still acceptable, and no short-term plans exist to upgrade the power supplies to achieve the additional current capability.

We will therefore formally adjust the requirement:

ELEC-01-02 The design coil current shall be 4 A.

With the ES-030-10 power supplies, the 30 V capability will limit the available coil current. For the largest coil (Z) this results in a current of 4.8 A (Figure 13.2). Designing the system around a more nominal 4 A seems therefore acceptable. This means that requirement **ELEC-01-02** can be met, and will therefore be listed as a **PASS**.



Figure 13.2: Pushing the power supplies to their maximum compliance voltage reveals the maximum current capability through each coil pair. From top to bottom, the supply units correspond to the $X/Y/Z$ coils. The Z-coils have the largest resistance, limiting the current to about 4.8 A.

ELEC-02-01 The coil current shall not exceed 6 A. T

PASS - The best way to implement this requirement is to do it in hardware: by electronically limiting the current from exceeding 6 A. It was originally intended to do so by exploiting the built-in current limits of the COTS power supplies (BK Precision 1685B), which cannot be programmed with control voltage to go above 5 A. However, given that the original power supplies were swapped back in at a late stage, this can no longer be exploited.

In practical terms, the power supplies are still voltage-limited to approximately 30 V, meaning they will not be able to drive more than around 4 A through the coils. The low level software interface additionally includes checks to filter out driving the power supplies harder than 6 A. This means that requirement **ELEC-02** has been met, though not in as elegant a way as originally intended.

ELEC-03-01 The Helmholtz coils shall be driven by a current-controlled source. D

PASS - The power supplies that drive the coils can operate as voltage-programmable constant-current power supplies.

ELEC-04-01 The Helmholtz coils shall be driven by a voltage source capable of at least 40 V. D

This requirement cannot be met in its current state, due to the power supplies that are currently used as part of the system. For similar reasons as for **ELEC-01-01**, we will formally adjust the requirement:

ELEC-04-02 The Helmholtz coils shall be driven by a voltage source capable of at least 30 V.

This requirement has been met, and will therefore be listed as a **PASS**.

ELEC-05-01	**The Helmholtz coils shall be driven by a bidirectional current.**	**D**

PASS - The power supplies supply a unipolar current, which is either inverted or non-inverted by H-bridge circuits on the companion boards. There is no difference in internal resistance or voltage drop for either state, meaning the performance is predictable.

ELEC-06-01	**The system shall be able to measure the current applied to each coil pair with a resolution of 10 mA.**	**T**

PASS - The instantaneous current measured by the shunt resistors on the companion board was compared to the values measured by the internal current monitor of the ES-030-10 power supplies. After some fine-tuning of the software calibration curve, the current from the companion board could be measured with a resolution of well below 10 mA. The noise floor of the current measurements were found to be in the range of a few mA.

It is recommended to improve this verification in the future by testing the impact of board temperature on the current readings with a sensitivity analysis, as the board can heat up significantly if the dropper diodes are conducting several amps of current. This can heat up the current sense resistors and signal amplifiers in turn, which may cause the software calibration to drift to an unacceptable degree.

ELEC-07-01	**The power supplies shall be protected from inductive voltage transients by at least one layer of protection.**	**A, T**

PASS - The companion board facilitates this by design, implementing multiple layers of protection. See section 10.3 for more comprehensive discussion.

ELEC-08-01	**The system shall be capable of handling inductive voltage transients without damage**	**T**

PASS - This is primarily handled by a bipolar transient voltage suppression (TVS) diode clamp. See Figure 10.32 and section 10.3 for more information.

ELEC-09-01	**Structural components and chassis elements near the electronics shall be grounded properly.**	**D**

PASS - The power supplies are grounded through their ground pin at the power bar, which like before serves as the star grounding point for most of the system. The chassis of the supply cart is also grounded to it. Given that this was originally part of the setup, confidence is high that this will work fine in the future. The companion board has its ground plane connected to that of the Raspberry Pi, which is not grounded at the power bar, and to the power supplies, meaning that the companion board is grounded via the digital ground of the power supplies. Note that the companion board uses multiple grounds, as the high current lines between the power supplies, H-bridges, and Helmholtz coils are disconnected from the board ground.

ELEC-10-01	**The total system power consumption shall not exceed 1 kW.**	**T**

PASS - This was tested by attempting to drive the system into a state where it was expected to consume the most power. All power supplies were ordered to pump the maximum current that they could deliver into their load. All other devices were turned on, including the high-level interface computer. The current was measured on the wall plug connection of the primary power bar, since all system power is routed through it. Current was measured with an ordinary digital power socket meter and a hand-held inductive current meter. No power consumption over 600 W was observed.

ELEC-11-02	**The voltage ripple though the Helmholtz coils shall not exceed 10 mV_{pp} during nominal operation.**	**T**

PASS - This requirement was added at the start to preclude the possibility of the magnetic field output of the Helmholtz cage rippling. However, practical experience shows that the Helmholtz coil and companion board appear to form a strong enough RLC filter to quash voltage ripple down to insignificance.

ELEC-12-01	**The power supplies should be current controllable using an analog control input voltage.**	**D**

PASS - This was implemented consistently at the design level. The current power supplies were modified to accept analog control inputs (see Appendix E).

13.4. Software Requirements

SW-01-01	**The software shall have a clear separation between firmware and user interface.**	**D**

PASS - The system design fully embodies such a separation, as does the software that goes along with it.

SW-02-01	**If a graphical user interface is created, it should not supplant the underlying application programming interface (API) or command line interface (CLI).**	**D**

PASS - The GUI that was created as part of the Helmholtz Cage Toolkit (HCT) software package broadly uses the same functions that can also be called from the API. To see how this takes form within the software package, the colour coding in Figure 11.19 is a good starting point.

SW-03-01	**The software shall enable automated data acquisition.**	**D**

PASS - Automated data acquisition can be done via the GUI, or it can be done via script on either the high-level interface (by connecting as a TCP client) or on the low-level interface (by running a script that uses the firmware functions).

SW-04-01	**The software shall enable for reliable data acquisition.**	**D**

PASS - The data acquisition was assessed during the overall system validation and found to be sufficiently accurate and reliable to perform all tests. A few recommendations to improve this further can be found in section 14.3.

SW-05-01	**The source code of the software shall be open to the general public.**	**D**

This requirement is considered to be met for the following reasons:

1. The source code is hosted in its entirety on GitHub [74].
2. The source code is licensed under a very permissive Creative Commons license, meaning that it is also open for reproduction and modification by the general public.
3. Sections of the source code are available in the appendix of this thesis, which is available to the general public by way of the TU Delft repository.

Therefore this requirement receives a **PASS**.

SW-06-01	**The software shall allow for automatic EMF rejection in the Helmholtz cage.**	**D**

PASS - The software allows automatic rejection insofar as the low-level interface can be configured to constantly reject a constant disturbance, independent of the schedule that is being played back. The GUI contains convenience buttons to automatically reject the local EMF (based on the International Geomagnetic Reference Field (IGRF) model and local GPS coordinates), or the instantaneous measured disturbance at the centre of the Helmholtz cage.

SW-07-01	**The software shall allow for manual control of field in the control volume.**	**D**

PASS - Besides the playing back of pre-programmed schedules, the Helmholtz cage allows the user to set the Cartesian components of the field in the Helmholtz cage with numerical values. However, there are no knobs or joysticks to do this by hand (literally "manually"), but this is considered acceptable.

SW-08-01	**The software shall allow for pre-programmed magnetic field envelopes to be played back.**	**D**

PASS - The concept of magnetic schedules has been developed and implemented to facilitate this. For a more detailed discussion, see subsection 11.4.6.

SW-09-01	**The software shall include the capability to account for satellite rotation for the simulated magnetic field envelopes.**	**D**

PASS - The HCT includes a tool to generate magnetic envelopes for the Helmholtz cage that are based on the orbital motion of a satellite. This tool takes into account satellite rotation by defining a separate body reference frame, and allow this reference frame to be rotated with respect to the simulated satellite's inertial reference frame. The current version of the tool is limited in that it can only take into account constant body angle offsets and constant rotation rates, but this is deemed adequate at the present moment.

SW-10-01	**If the software contains any checks that are required for safe operation, these shall not be directly accessible through the user interface.**	**D**

PASS - Many hardware related checks and safety functionality that is performed in the low-level interface are not exposed to connecting clients on the TCP connection. In other words, these properties can only be accessed by an admin who is directly interacting with the hardware of the low level interface, either by direct connection of keyboard and mouse, or via something like an Secure Shell (SSH) connection. This requires the user password of the Raspberry Pi, which is not required for access via the TCP connection.

SW-11-01	**The software shall allow for data acquisition of at least three magnetic sensors at a sample rate of at least 20 S/s.**	**T**

PASS - This has been implemented to specification.

SW-12-01	**The software shall be sufficiently documented such that other engineers unrelated to the project could use it.**	**D**

This requirement receives a **PASS** for the following reasons:

1. The HCT software package is fully open source, meaning that anyone can look at the source code.
2. There is extensive code documentation and docstrings in code files, sometimes including performance figures.
3. The source code is supported by this thesis and its appendices, which provide the kind of higher-level information and reasoning that is typically lost in code documentation.
4. Certain parts of the software are explained further by a number of unit test scripts and dedicated example scripts, which are part of the HCT.

13.5. Summarized Verification Matrix

Table 13.1 contains a summary of the evaluated requirements and their status. Conditionally passed requirements are indicated with an asterisk (*). Requirements that were adjusted in the previous section are marked$^{(ADJ)}$.

Status	ID	V	Summarized requirement
PASS	DS-01-02$^{(ADJ)}$	A	Control volume is a 415 mm diameter sphere
PASS	DS-02-01	A	Field nullification vector under 500 nT
PASS*	DS-03-01	A+T	Field strength uniformity of 1% in control volume
PASS*	DS-04-01	A+T	Field directional uniformity of 1% in control volume
PASS	DS-05-01	D	Possible to suspend satellite-under-test in control volume
PASS	DS-14-01	D	Capability to operate according to prepared schedule
PASS	DS-15-01	D	Straightforward interface usable by engineering students
PASS	DS-16-01	D	System is safe for user to operate
PASS	DS-17-01	D	Time-efficient testing
PASS	DS-18-02$^{(ADJ)}$	A	Total system cost does not exceed €12.5k
PASS	DS-19-01	D	System uses COTS parts where possible
PASS*	PER-01-01	T	Simulate arbitrary field up to at least 210 μT
PASS*	PER-02-01	T	Correct local magnetic field variations down to 500 nT
PASS	PER-03-02	A+T	Simulate rotating fields up to 20 rpm
PASS	PER-04-01	D	Determine generated field strength with a 10 nT resolution
PASS	PER-05-01	D	Determine field direction at the control volume
PASS	PER-06-01	D	Monitor temperature at critical locations
PASS	PER-07-01	T	Coil temperature does not exceed $60°C$
PASS	PER-08-01	A+T	Coil current direction switchable in under 1 ms
PASS	ELEC-01-02$^{(ADJ)}$	D	Design coil current of 4 A
PASS	ELEC-02-01	T	Coil current shall not exceed 6 A
PASS	ELEC-03-01	D	Coils driven by current-controlled source
PASS	ELEC-04-02$^{(ADJ)}$	D	Coils driven by voltage source capable of 30 V
PASS	ELEC-05-01	D	Coil current is bidirectional
PASS	ELEC-06-01	T	Measure coil current with a 10 mA resolution
PASS	ELEC-07-01	A+T	Power supplies protected from inductive voltage transients
PASS	ELEC-08-01	T	System will not be damaged by inductive voltage transients
PASS	ELEC-09-01	D	Chassis/structural components electrically grounded
PASS	ELEC-10-01	T	Total system power consumption under 1 kW
PASS	ELEC-11-02	T	Voltage ripple through Helmholtz cage not exceed 10 mV_{pp}
PASS	ELEC-12-01	D	Power supplies controllable using analog control voltage
PASS	SW-01-01	D	Software separation firmware / user interface
PASS	SW-02-01	D	Any GUI does not supplant underlying API
PASS	SW-03-01	D	Software includes automated data acquisition
PASS	SW-04-01	D	Software enables reliable data acquisition
PASS	SW-05-01	D	Software source code open to general public
PASS	SW-06-01	D	Software allows for automatic EMF rejection
PASS	SW-07-01	D	Software allows for manual magnetic field control
PASS	SW-08-01	D	Software allows for pre-programmed magnetic field playback
PASS	SW-09-01	D	Simulated magnetic field envelopes account for satellite rotation
PASS	SW-10-01	D	Operational safety software checks not directly accessible through UI
PASS	SW-11-01	T	Data acquisition of at least three magnetic sensors at 20 S/s
PASS	SW-12-01	D	Software sufficiently documented for use by other engineers

13.6. System Validation

With the system requirements verified to satisfaction, a general system validation could be carried out. Due to time constraints, the validation was not fully exhaustive but good enough to be confident that the system works as intended.

The system validation was focused specifically on assessing whether the following two conditions could be met:

1. The system's magnetic transfer characteristics are linear in the domain [-200, 200] μT.
2. The magnetic field envelope experienced by a satellite in Earth orbit can be simulated by the Helmholtz cage with an acceptable accuracy and precision.

13.6.1. Assessing transfer characteristics

By transfer characteristics, we specifically mean the transfer function between the commanded magnetic field vector $\overline{B}_C$ and the measured output field vector $\overline{B}_M$. Not only is it important to measure the transfer characteristics, but it is also important to be able to have some form of infrastructure to automatically calibrate the system such that it automatically compensates for its own non-linearities.

To do this, an auto-calibration script was written. This script commands the magnetic field vector on a single axis to a certain field value, waits for the system to settle, and measures the steady-state field vector that was produced. It does this for a range of field values between and up to [-200, 200] μT, and then moves on to the other axes. Between each operation the system pauses briefly to measure the local disturbance vector and nullifies it, so that local disturbances are periodically minimized.

After it has gathered all these I_{in}-B_{out} data pairs, the script uses least-squares regression on the inverse of this data to fit a calibration curve, with the intent of using it to correct any non-linearities. It was found that this data is fairly linear in the middle, but shows signs of saturation at higher current values (Figure 13.3).

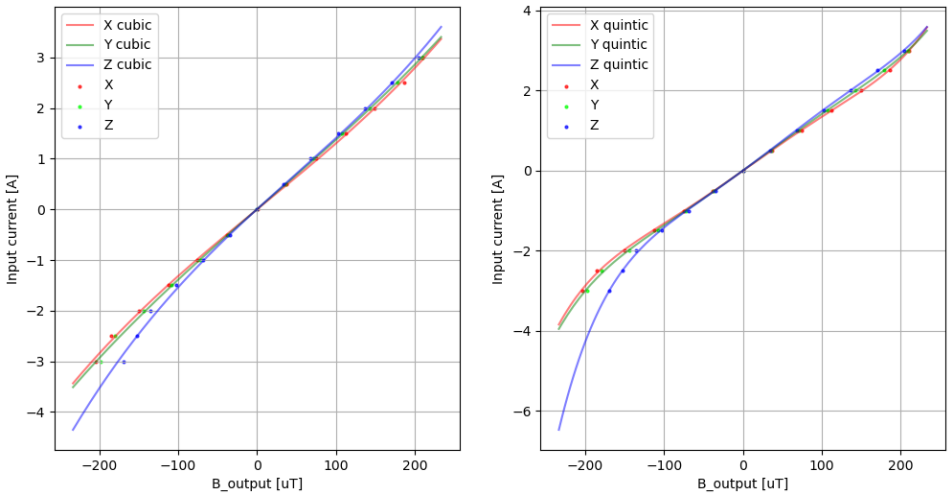


Figure 13.3: During auto-calibration, the relationship between input current and output field strength are measured on each axis (dots in the figures). A cubic (left) and quintic (right) polynomial were fitted to the inverted data, in the hope that such a function would correct for the non-linearities.

For each axis, a cubic polynomial was fit through the points, and the coefficients were saved so that these could be used to tell the system what current to output for a desired magnetic field. The fitting process prioritized fitting the curve through 0 A with a 10:1 ratio compared to the other points, to ensure calibration would be most accurate around the origin and no offset would remain. A quintic polynomial was also tried, but it was later discarded since its performance was not any better than the cubic fit and evaluating is more expensive too.

The auto-calibration script also has an optional validation test, which takes the computed calibration curves and performs a series of measurements with the calibration effects active. If the system has been

perfectly calibrated, it means that on each axis a certain magnetic field input will correspond 1:1 with the output. The result can be seen in Figure 13.4. The cubic polynomial takes the non-linearities observed in Figure 13.4 and uses them to re-linearize the system. It is not perfect, but nowhere is it far off either.

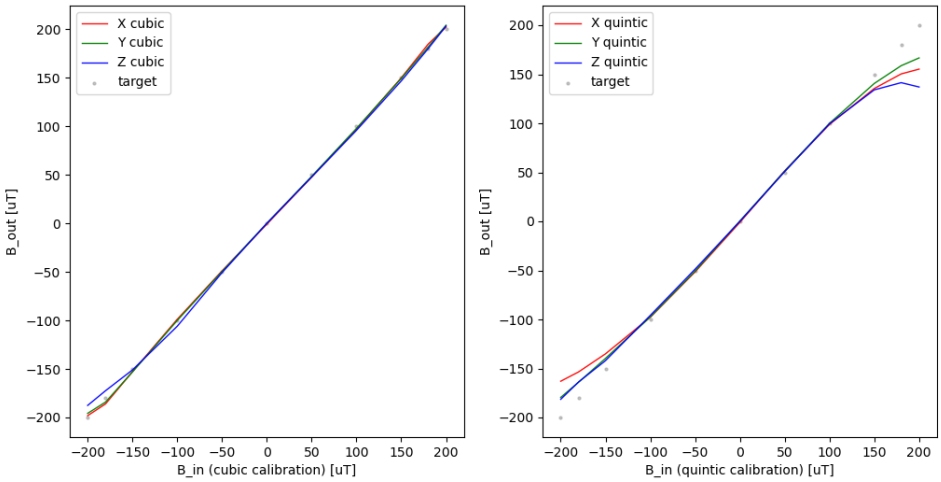


Figure 13.4: After applying the calibration polynomial, another series of measurements is performed to measure the actual transfer characteristics. The cubic (left) polynomial was chosen over the quintic (right) polynomial, since it is cheaper to compute and appears to perform better at the extremes. The gray dots are the target for perfect linearity.

This auto-calibration can be done once, but since it takes a few minutes only, it may be a good idea to run it at the start of a series of tests, so that any changes to the system (temperature, humidity, coil mounting imperfections, etc.) can be corrected for. In chapter 7, we noted that the position of the elevators had a significant impact of the magnetic field inside the test setup, and so it should be noted that its noise signal may also make its way into a calibration like this.

To get a better idea of the tracking performance of system after calibration, a simple test schedule was made using the *Cyclics* generator in the HCT. It describes a simple 100 μT helix inside the Helmholtz cage. After playing this schedule on the hardware, the difference between the commanded input vector $\overline{B}_C$ and the measured output vector $\overline{B}_M$ could be compared. This can be seen in Figure 13.5. The top of this figure shows the envelope of the three axial components of the schedule, and the measured components in magenta/yellow/cyan.

The tracking performance is good enough that at this scale, the error is difficult to see. Let us look at the error more closely. In this chapter, the following error definition will be used:

$$\overline{E} = \overline{B}_M - \overline{B}_C \tag{13.1}$$

From this we can calculate the **absolute command error**, which can either be expressed in terms of absolute magnitude or in the cardinal components of this vector:

$$\overline{E}_C = \sqrt{E_{C,X}^2 + E_{C,Y}^2 + E_{C,Z}^2} \tag{13.2}$$

The lower half of Figure 13.5 shows the three $X/Y/Z$ components of the absolute command error in red/green/blue respectively. The combined error corresponding to these three components is shown in white. $\overline{E}_C$ can be thought of as the length (in μT) of the vector that points from $\overline{B}_M$ to $\overline{B}_C$.

To evaluate the performance of the system, we would like to express the error as one or more scalars. We can calculate the maximum and root mean square (RMS) errors like so:

$$E_{max,X} = \max(\overline{B}_{M,X} - \overline{B}_{C,X}) \quad , \quad E_{max,Y} = \max(\overline{B}_{M,Y} - \overline{B}_{C,Y}) \quad , \quad E_{max,Z} = \max(\overline{B}_{M,Z} - \overline{B}_{C,Z}) \tag{13.3}$$

and

$$E_{max} = \max(\overline{E}_C) \tag{13.4}$$

Furthermore,

$$E_{RMS,X} = \sqrt{\sum_{i=0}^{n-1} \frac{(B_{M,X,i} - B_{C,X,i})^2}{n}}, \quad E_{RMS,Y} = \sqrt{\sum_{i=0}^{n-1} \frac{(B_{M,Y,i} - B_{C,Y,i})^2}{n}}, \quad E_{RMS,Z} = \sqrt{\sum_{i=0}^{n-1} \frac{(B_{M,Z,i} - B_{C,Z,i})^2}{n}} \tag{13.5}$$

and

$$E_{RMS} = \sqrt{E_{RMS,X}^2 + E_{RMS,Y}^2 + E_{RMS,Z}^2} \tag{13.6}$$

Whilst the absolute error is useful to assess the performance of the Helmholtz cage, the **angular error** is going to be more interesting to users, since it tells them essentially what pointing error they can expect from the system. We can compute it as follows:

$$\overline{\Lambda} = \overline{E}_{angle} = \cos^{-1}\left(\frac{\overline{B}_M \cdot \overline{B}_C}{|\overline{B}_M| \cdot |\overline{B}_C|}\right) \tag{13.7}$$

which we will express as:

$$\Lambda_{max} = \max(\overline{\Lambda}) \tag{13.8}$$

and

$$\Lambda_{RMS} = \sqrt{\sum_{i=0}^{n-1} \frac{\overline{\Lambda}_i^2}{n}} \tag{13.9}$$

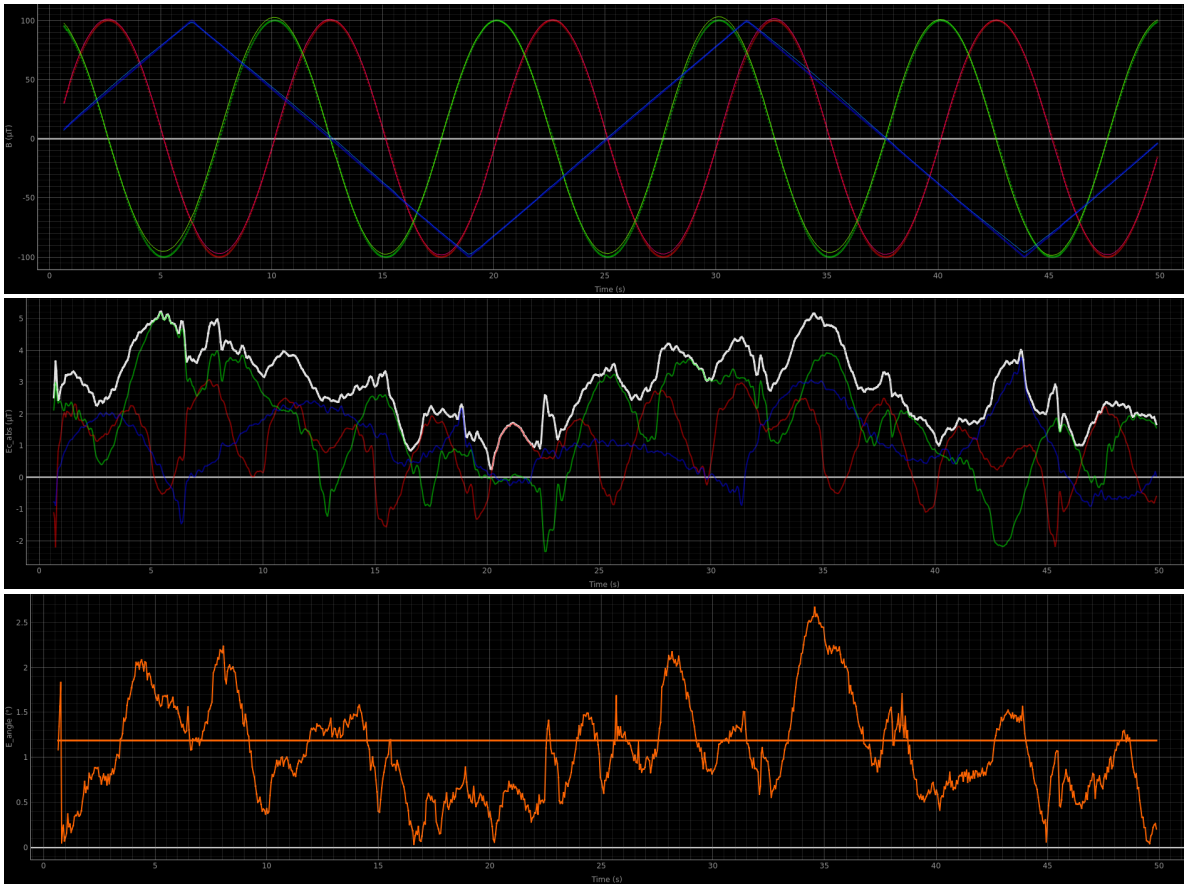


Figure 13.5: Periodic test schedule used to test some baseline tracking performance of the system. On the top, we see the $X/Y/Z$ components of $\overline{B}_C$ in red/green/blue respectively, and the measured magnetic field components $\overline{B}_M$ in magenta/yellow/cyan. In the middle, we see the absolute command error for each component (red/green/blue) as well as their norm (E_C) (white). In the bottom, we see the angular error Λ in orange.

The various error components that we just discussed can also be seen in the other plots in Figure 13.5. In more quantitative terms, the error components for this particular test schedule are as follows:

Table 13.2: Command error observed during simple helical test

	X $[\mu T]$	**Y** $[\mu T]$	**Z** $[\mu T]$	$\mathbf{E_C}$ $[\mu T]$	Λ [°]
Max error	3.160	5.218	4.027	**5.239**	**2.67**
RMS error	1.499	2.254	1.441	**3.067**	**1.19**

Note that the schedule used here was set to a **command rate** of 20 Hz, meaning the commanded field values are updated every 50 ms. This is an important concept, because if the schedule is too low resolution, the schedule will not be very smooth, but rather stepped and jagged. This can be seen in Figure 13.6, where the same schedule has been re-evaluated but with a command rate of 2 Hz instead. The resulting maximum error rose to 21 μT and the average error to 5 μT, illustrating the need to choose a command rate and schedule resolution that are sufficiently high. However, if the command rate is chosen too high, this may place too much strain on the low-level interface. It was observed that this increases the frequency of glitches and sporadic drop-outs, and at command rates over 50 Hz, schedules were played back at a significantly slower rate than real-time, which is highly undesirable.

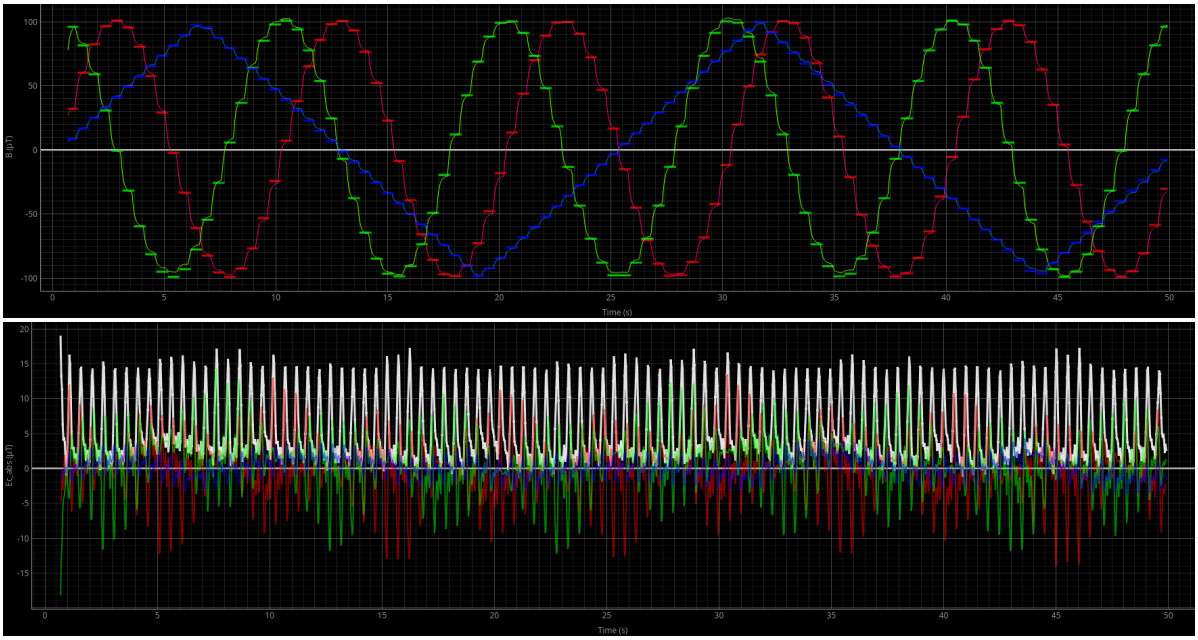


Figure 13.6: The effect of using a schedule with a low resolution produces a larger overall error. The Helmholtz cage is forced to track between large discrete steps, which causes a large tracking error just after such a step and decreases as the power supplies adjust.

In regards to the step response timing characteristics, the following can be concluded: The step response for the system shows very little to no ringing, so we will will just investigate the settling time. We define t_{step} as the time until 98% of the final value is reached. It was observed that the settling times of the Z-coils were slightly higher than those for X and Y, which can be explained by its slightly higher coil inductance. As can be seen in Figure 13.7 and Table 13.3, the settling time of a step response will be somewhere between 400 and 500 ms, with higher step size leading to marginally higher settling times.

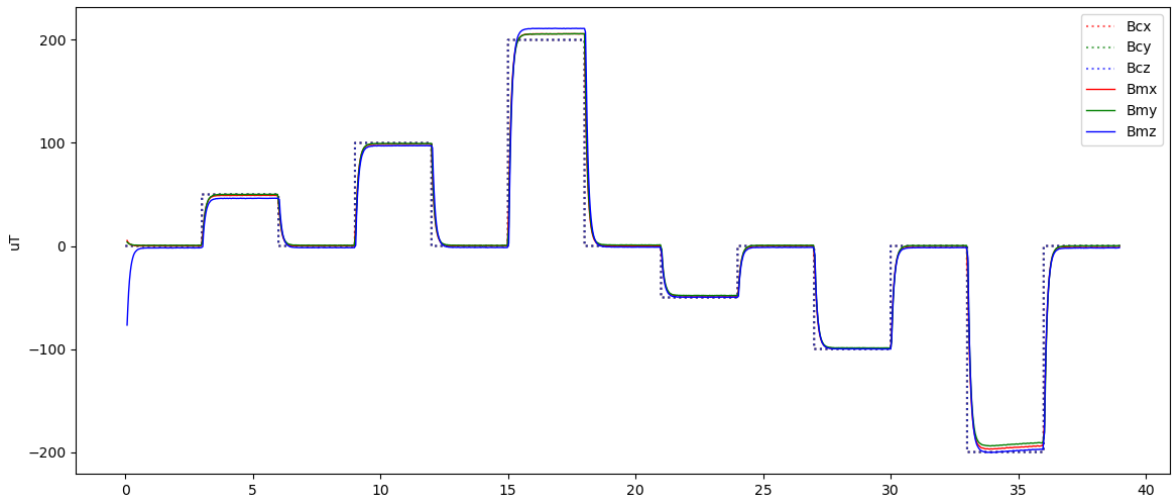


Figure 13.7: Helmholtz cage tracking a step function of various amplitudes.

Table 13.3: Typical step response settling times for the system (Z-coils)

Edge [μT]	50	-50	100	-100	200	-200
Rising t_{settle} [ms]	405	407	407	429	461	416
Falling t_{settle} [ms]	437	445	439	448	501	459

Also note that for the steps with a magnitude of 200 μT, there is a visible error, which not only has to do with the fact that the transfer function is less linear for these higher values, but also with some effects that will be discussed later in this chapter.

What has not been discussed yet is that a propagation delay of approximately 100-150 ms was observed between command execution and response on the measurement equipment. This is not visible in Figure 13.5 and Figure 13.6 because it has been corrected for in post-processing. This propagation delay is not expected to be a significant hindrance to a system of this size and intended to be operated at low frequencies, but it is important to correct for this during data processing. Correction can simply be achieved by shifting the data by a corresponding number of samples.

13.6.2. Simulation of orbital motion

The second part of the validation concerned the simulation of orbital motion of a satellite with the Helmholtz cage. To do this, the GUI of the Helmholtz Cage Toolkit was used to generate a magnetic schedule of a certain satellite orbit (Figure 13.8). This was then automatically processed into a schedule, which defines the movement of the local magnetic field vector in the body frame of the satellite as a function of time (Figure 13.9).

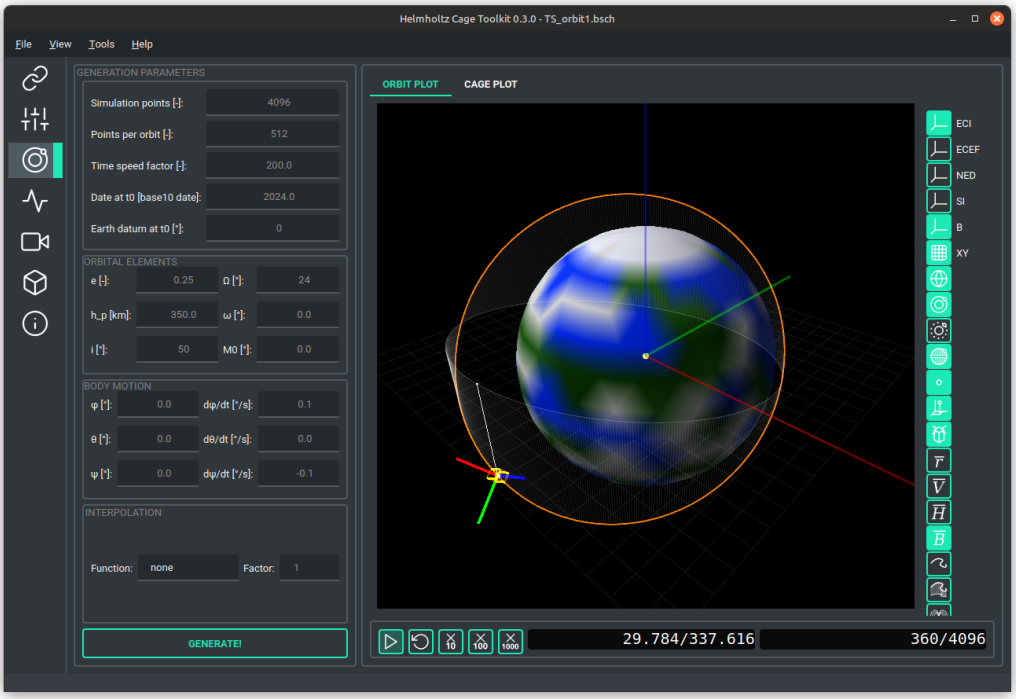


Figure 13.8: Setting up a satellite orbit simulation using the *Orbitals* generator in the HCT.

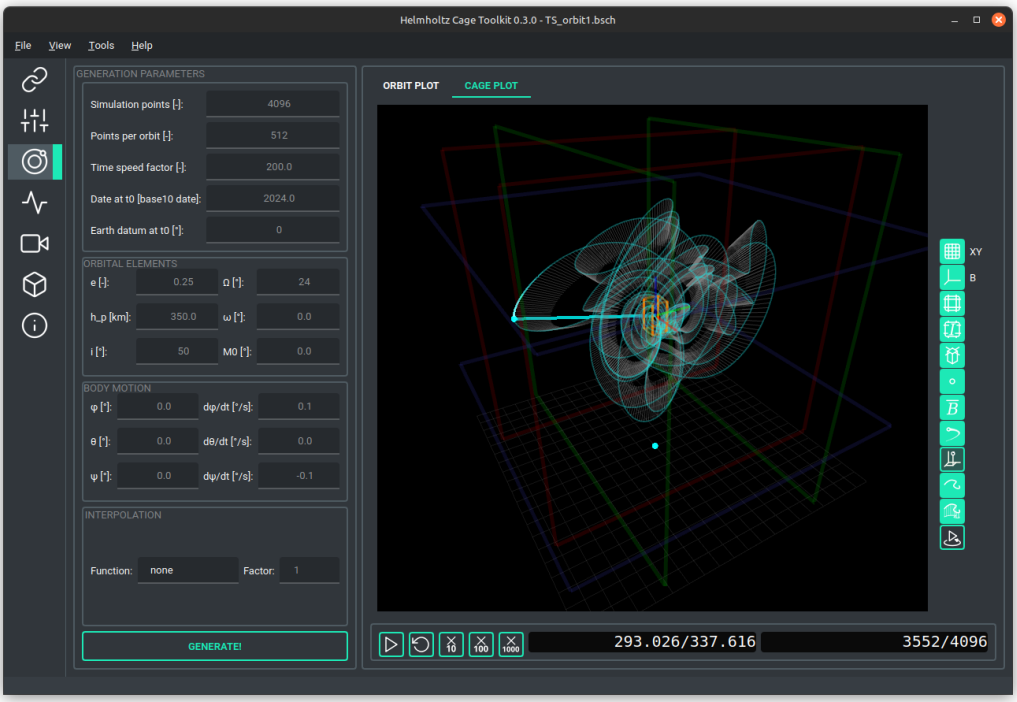


Figure 13.9: Viewing the resulting instructions for the Helmholtz cage, which represent the magnetic field experienced by the satellite.

This schedule was then transferred to the low-level interface (Raspberry Pi) and played by the Helmholtz cage, whilst the magnetic sensor was constantly monitoring the instantaneous magnetic field vector. The results could then be processed and visualized in a similar manner as in the last section.

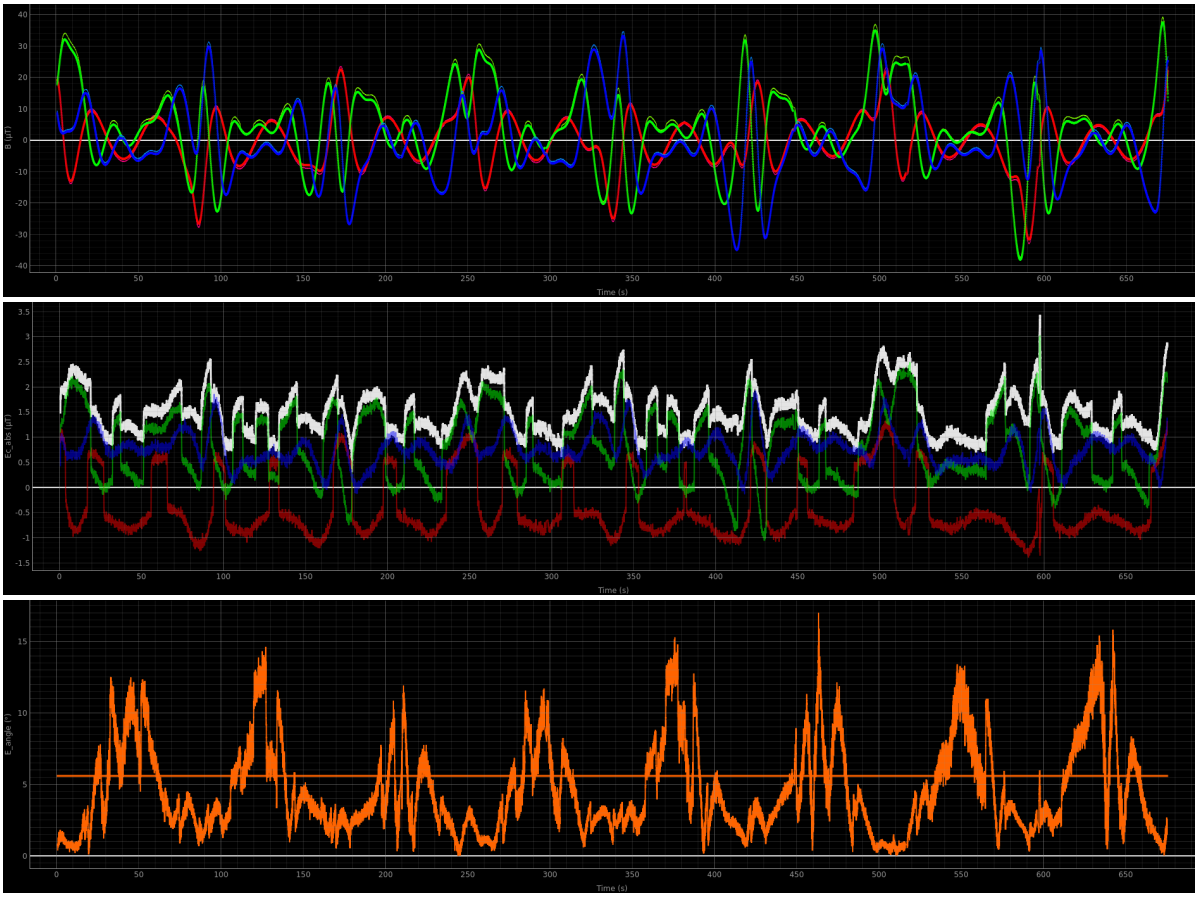


Figure 13.10: The envelope and error properties of the schedule after being played on the Helmholtz cage.

Table 13.4: Command error observed during orbit simulation test

	X $[\mu T]$	**Y** $[\mu T]$	**Z** $[\mu T]$	$\mathbf{E_C}$ $[\mu T]$	Λ [°]
Max error	1.428	3.051	2.132	**3.418**	**16.97**
RMS error	0.746	1.078	0.838	**1.556**	**5.59**

The numerical values from Table 13.4 reveal that although the absolute errors are lower compared to the helical test case, the angular error is significantly larger. At certain locations in the schedule, the vector established by the Helmholtz cage may be off by as much as 17°.

The reason for this large angular error can be deduced easily from Figure 13.11. This figure shows the commanded $\overline{B}_C$ manifold and the measured $\overline{B}_M$ manifold in 3D space. The latter has been coloured in proportion to the local angular error: the greener sections have a low angular error and the orange-red sections have an angular error closer to the maximum error of 17°. We see that all red sections are clustered around the origin, and this makes sense: when some commanded $\overline{B}_C$ vector has a very small magnitude, a constant offset error produced by the Helmholtz cage will mean that the resulting $\overline{B}_M$ points in a rather different direction, compared to the case where both these vectors have a large magnitude. This can also be seen in Figure 13.10 by comparing the top and bottom plots: the angular error (bottom plot) spikes when the three components of the magnetic field schedule (top plot) are all small, even if the absolute error (middle plot) is not doing anything unusual there.

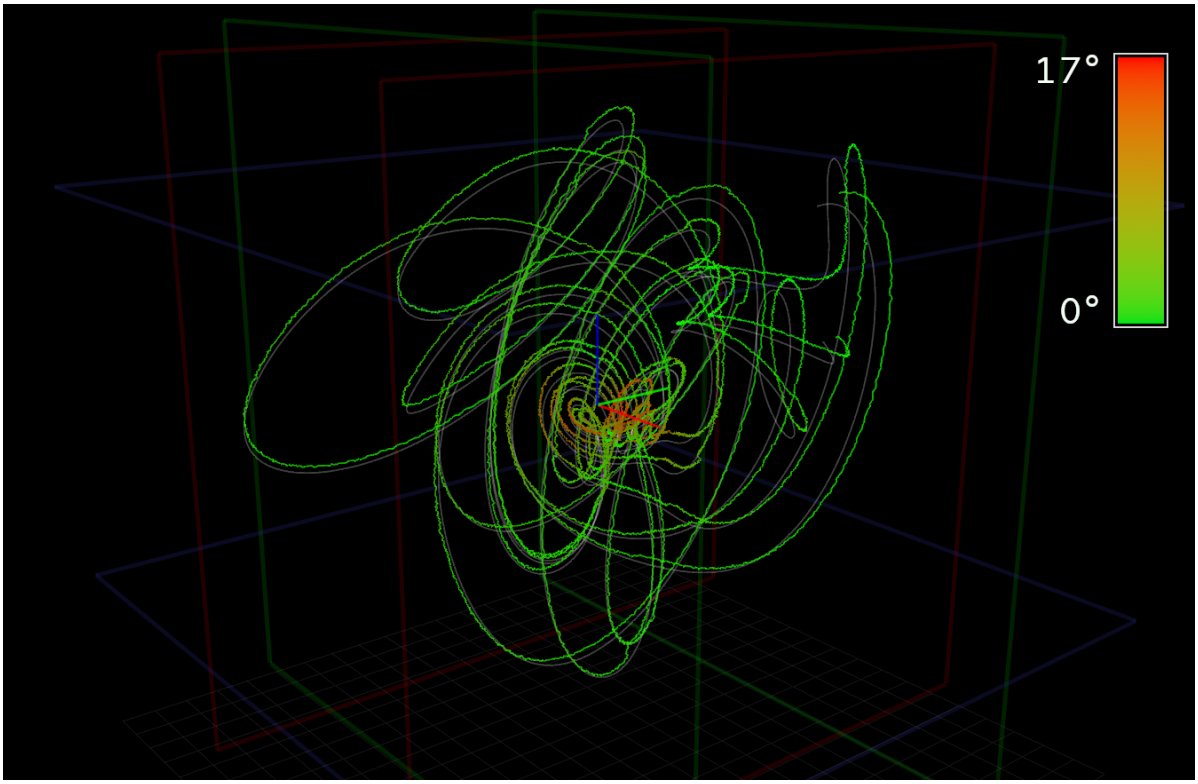


Figure 13.11: The envelope of the simulated satellite schedule seen as its 3D manifold. The gray line represents the commanded vector path $\overline{B}_C$, whereas the coloured one represents the measured data $\overline{B}_M$. The colour gradient is based on the magnitude of the absolute error, meaning that green line segments have a low absolute error, and red areas have a high absolute error.

This problem seems difficult to avoid in practice: any schedule that contains vectors that are close to the origin will have large maximum angle errors. In a moment, we will look at a number of other orbits based on real missions to see what the implications for the system are.

Before we do this, there are more anomalies that can be found in the test data. If we look at the absolute errors in Figure 13.10, we see that the green (Y) and blue (Z) plots are not centered around zero. Their mean will be non-zero, meaning that there is at least a constant offset error present that should have been corrected for with perfect system calibration. We also that the error changes with time in movements that are vaguely reminiscent of the movements of the schedule itself (top plot of Figure 13.10). This could indicate that there is a scaling error, which also should be corrected by perfect calibration but clearly is not.

However, the elephant in the room is the curious stepped behaviour. Rather sporadically, the red (X) and green (Y) curves will shoot up or down sharply. The rising and falling edges of this phenomenon are consistently about 200 ms wide and about 1 μT in height. The blue (Z) curve does not exhibit this behaviour at all. What could explain these strange jumps?

The answer can be found in Figure 13.12. If we look at the middle and bottom plots, we see that the jumps in magnitude error occur exactly in the places where the H-bridges switch the polarity of the coil current. It was difficult to conclude this from the envelope data alone, because besides playing the schedule, the Helmholtz cage is also actively neutralizing the local disturbances such as the EMF. The envelope data has this vector automatically subtracted. The jumps were not present in the blue (Z) data because the local EMF has a large vertical component, and cancelling it pushes the coil current to a high enough value so that the schedule never pulls it below zero. As a result, the H-bridge does not ever switch for the whole duration of the schedule. However, for other schedules this may not be so, in which case the Z-axis too will exhibit these stepped errors.

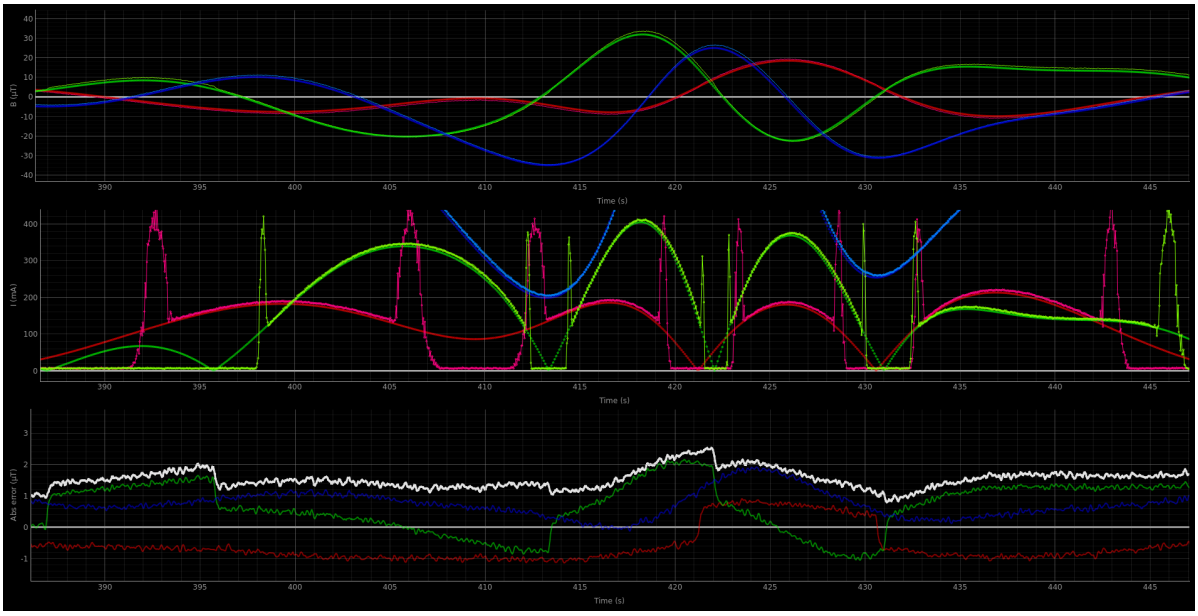


Figure 13.12: The same schedule as before, but zoomed in on a minute long segment. The top graph shows the components of $\overline{B}_C$ and $\overline{B}_M$, the middle graph shows commanded and measured current $\overline{I}_C$ and $\overline{I}_M$. The bottom graph shows the magnitude errors as before.

In Figure 13.12 one can also see current spikes when the current goes through a particular threshold somewhere between 100 and 200 mA. This current corresponds to the compliance voltage of 0.85 V, which is the forward voltage of the dropper diodes. If the power supply has to supply a compliance voltage lower than 0.85 V, the dropper diode becomes non-conductive and will block current. This is also evident from the data, which shows that the current flat-lines for such low current values. However, the moment that the diode starts conducting or blocking will cause the current to change rapidly (dI/dt will be large). Lenz's law teaches us that in this case, the coils will induce an opposing voltage to try to counteract this change. This results in the crests that can be seen in Figure 13.12.

Several more issues with schedule playback were found:

1. Due to the slew rate of the power supplies, there is a **startup transient** at the beginning of the schedule where the error is temporarily very high. This was addressed during testing by discarding the first 600 ms of test data.
2. For command rates (Helmholtz cage instructions) of over 30 S/s, strange **timing desynchronization** was observed (bottom right of Figure 13.13). It is presumed that the code that instructs the CN-0554 DAC/ADC is far from ideal and cannot be looped at the required rates. This results in the schedules being played back at a lower rate than real-time, resulting in frequency distortion and much longer test times than intended. All subsequent testing was done at a command rate below 25 S/s, and the issue was no longer present then. Users are therefore recommended to do the same until the driver code's efficiency can be improved.
3. Whilst addressing the CN-0554 with the driver software, there are **sporadic freeze-ups of the Raspberry Pi**. These typically last between 800 and 1300 ms (see Figure 13.13). It is not quite clear what causes this issue, but the running theory is that somewhere within the software abstraction stack, an error occurs that takes some time to fix. For the Python drivers, there is some terminal error logging, but after a thorough investigation it seems that the error log is merely symptomatic and does not point to an exact cause. The freeze-ups result in blind spots in the data, as no data is logged during this period, and also potential distortion of the played-back schedule. Frequency during validation tests were observed to be between once per half hour up to twice a minute. The frequency of the freeze-ups appears linked to the stress on the Raspberry Pi, and until the issue can be addressed by software improvements, users are advised to run as few processes on the Raspberry Pi in parallel as possible and potentially use headless interaction with the Raspberry Pi

(e.g. using SSH) so that processor overhead is reduced. Although the freeze-ups are sporadic, they can be detected by the software with 100% efficacy, and the validation testing activities used a specific column in the data to earmark such errors when they occurred. In the test data of this chapter, all these errors have been filtered away beforehand.

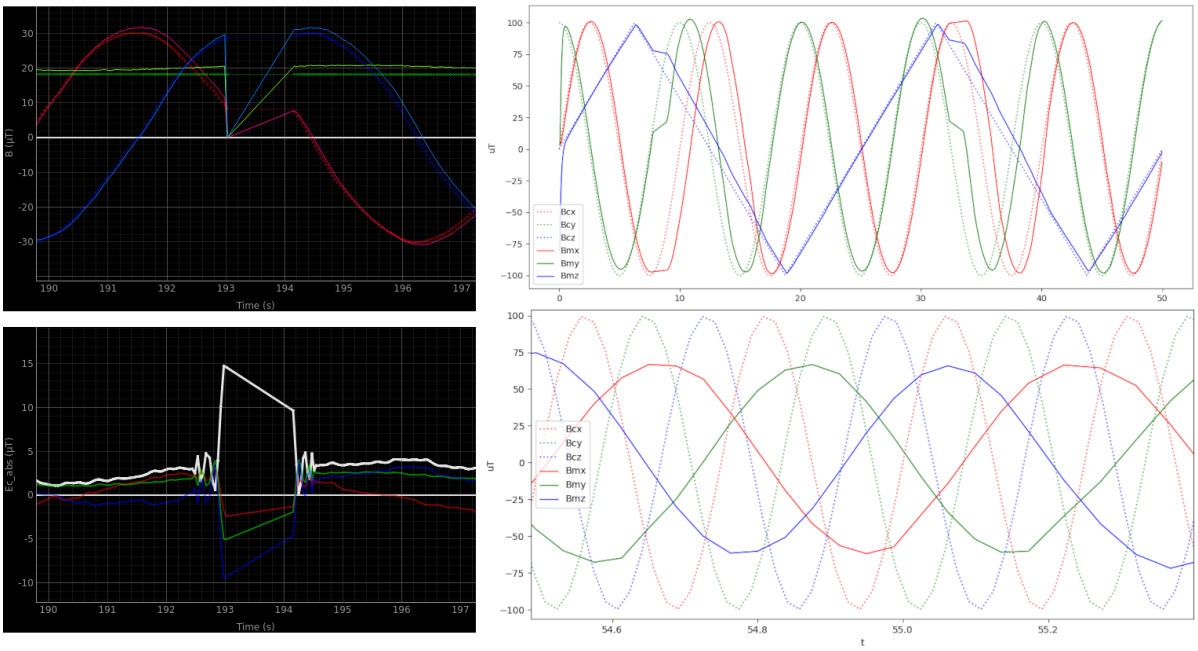


Figure 13.13: Various problems found with the current system. Top left: A sporadic 1.2 second freeze-up of the Raspberry Pi, as seen from the commanded and recorded magnetic fields. Bottom left: The (rather useless) corresponding absolute error. Top right: Two freeze-up glitches causing temporary poor tracking for a prolonged period afterwards. Bottom right: Operating at a 60 S/s command rate results in timing de-sync, causing the actual schedule to be played almost three times slower than real-time.

None of these phenomena are good, as they degrade the performance of the system. Curiously though, even with these problems, the system tracking is still reasonably good, as we saw before. The Helmholtz cage seems to even track the schedule when the current is demonstrably flat-lining, the reasons for which are yet unclear. Still, if we wanted the performance to be even better, what should be done?

13.6.3. Improving system performance

Within the scope of the thesis, there was not enough time to address the problems identified in the previous section; these will be left as future work, since the system does work reasonably well already. In this section, we will look at the following question: **if** someone were to address the identified problems, by how much would the system performance improve? How much mileage can be gotten out of going through the trouble of making additional improvements to the system?

The test data of the orbital simulation was run through a post-processing script that did the following:

1. For each axis, the data was corrected by 1 μT depending on the state of the corresponding H-bridge. This mostly cancels out the stepped error behaviour.
2. The data for each channel had its mean subtracted from itself, which removes the constant offset and centres the data around zero.
3. Some multiple of the commanded magnetic field envelope was subtracted from the data, to attempt to remove the scaling error by attempting to remove the correlated error components. This was not a very good filter, but it did lower the overall correlation between commanded field input and observed command error substantially.
4. The absolute error and angular error components were calculated before and after and their differences were expressed as percentages.

The aim of this was to attempt to artificially remove the observed system flaws from the data, and then see what error would remain. Although coarse, this can give some indication of how much the system performance could improve if the known problems are addressed.

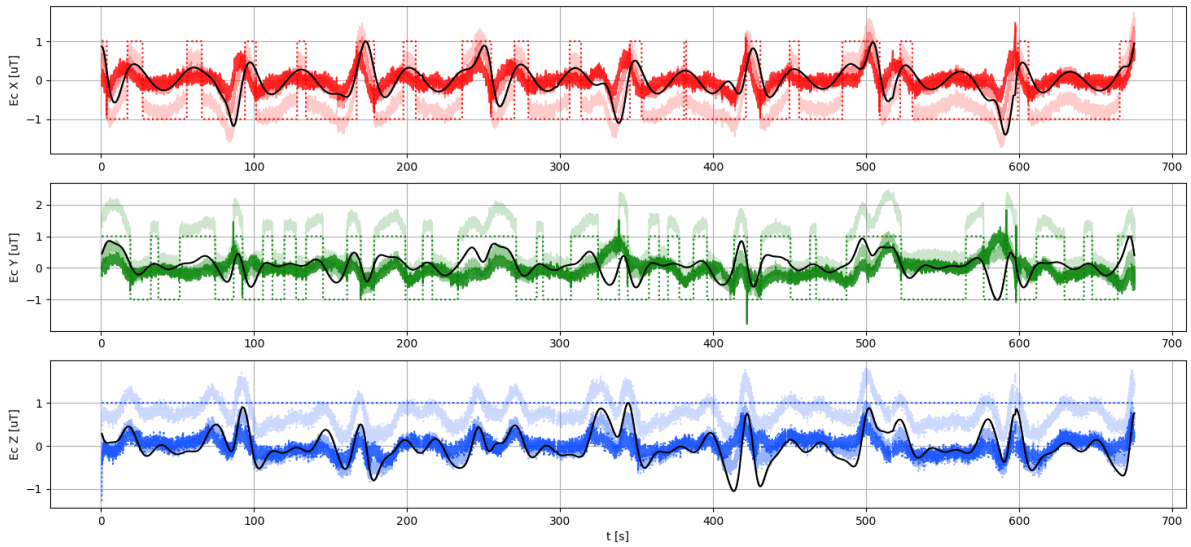


Figure 13.14: The absolute field error after various stages of filtering, in order to assess how much error would remain if the known problems would be solved. The brighter the curve, the more filters were applied. The black line is a dimensionless representation of the commanded magnetic field and is plotted for reference purposes.

When we assess the angular error from the filtered data (Figure 13.15), we can see that this has become substantially lower. The other errors have decreased as well (compare Table 13.4 and Table 13.5).

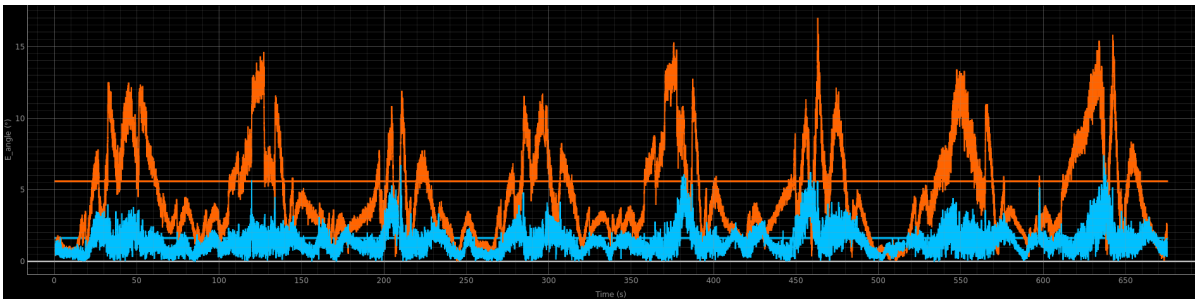


Figure 13.15: The angular error before (orange) and after (blue) the data has been filtered, indicating that addressing the H-bridge/diode related issue and improving the calibration can improve the tracking performance significantly.

Table 13.5: Command error for the **filtered** orbit simulation data. This performance is plausible if the observed problems are addressed.

	X $[\mu T]$	**Y** $[\mu T]$	**Z** $[\mu T]$	$\mathbf{E_C}$ $[\mu T]$	Λ [°]
Max error	2.058	2.702	1.736	**2.911** (-14.8%)	**6.44** (-62.1%)
RMS error	0.339	0.361	0.333	**0.597** (-32.7%)	**1.86** (-66.7%)

To investigate the reproducibility of these numbers, several more orbital simulations were performed. Some of these correspond to realistic mission scenarios, such as an Earth-pointing CubeSat in a sun-synchronous orbit. The goal of this is to discover patterns across all these tests to give a better estimation of how much the system can be improved. A total of eight such tests were performed, details of which can be found in Table 13.6.

Test	Date	Description
Orbit1	21-11-2025	Test orbit used throughout the thesis project, identical to the one from subsection 13.6.2. Satellite has mild body rates on two axes, and orbits at 350 km with a 50° inclination and a 0.25 eccentricity.
Orbit2	21-11-2025	A MEO case at 4600 km with an eccentricity of 0.2 and an inclination of 54°. At these altitudes, the IGRF yields degraded accuracy. The test case is mainly interesting because the field values are relatively small, the effect of which this tests aims to investigate. The satellite rotates mildly along one of its body axes.
Orbit5	21-11-2025	Simulates a common LEO CubeSat mission deployed from the ISS. A circular orbit with an altitude of 350 km and a 51.6° inclination. The satellite is nadir-pointing and does not rotate in its inertial reference frame. The test performs 20 orbits at a time speed factor of 100.
Orbit6	21-11-2025	The same CubeSat mission as the **Orbit5** test, but simulating 4 orbits at a time speed factor of 10. One of the goals of this test is to investigate the effect of the time speed factor on the overall test accuracy.
Orbit7	25-11-2025	Simulates a mission based on the 6U SpIRIT CubeSat mission [78], nadir-pointing in a circular, sun-synchronous orbit at 97.4° inclination. The orbital parameters assume a pericentre altitude of 415 km and are based on historical TLE data from the mission.
Orbit8	25-11-2025	Simulates a mission based on the IT-SPINS mission [16]. It does not simulate the troubled early section of the mission with the ADCS issues, but rather the intended -2 rpm Y_B-axis rotation used for gathering tomography data. Orbital parameters set shortly after the Cygnus NG-15 deployment at 500 km altitude with an inclination of 51.6°.
Orbit9	25-11-2025	Simulates a hypothetical Delfi-PQ orbit scenario where the Z_B-axis of the satellite is nadir-pointing, and the satellite is slowly rotating around this axis. Orbital parameters based on historical TLE data (457 km altitude, 97.4° inclination).
Orbit10	25-11-2025	Simulates the Hubble Space Telescope in a 28.5° inclination orbit at 488.7 km, pointing at a fixed hypothetical celestial target.

It took some time to run each simulation, post-processing the data, and calculate the various error components and improvements. However, the software tools developed during the thesis made this process a lot more straightforward and efficient. In Table 13.7, a complete overview of the error components of each test can be found, as well as the absolute improvement after applying the post-processing.

Although Table 13.7 is rather dense, there are some interesting things to point out. First, we see that for the **Orbit2** test, the angular errors Λ are *much* higher than for the other tests. This is because that particular test simulates a 4600 km MEO orbit, in which the experienced Earth magnetic field strength is much lower in magnitude. As a result, the magnetic field vectors established by the Helmholtz cage are closer to the origin, which we have seen before inevitably increases the angular error. However, for the same reason this also allows for the largest increase in performance from fixing the current issues with the system: the RMS angular error can be improved by a factor of three, and the maximum angular error by a factor of four!

Another observation that can be made is that the prospective improvements to the RMS command error are roughly in the same ballpark, suggesting that at least half of the error components are caused by the H-bridge/dropper diode problem and imperfect calibration. There is a bit more variation for the maximum command errors, but still improvements as high as 50% were observed.

Ultimately, the data Table 13.7 makes a strong case that further improvements to the system can improve the performance substantially. To expect an order of magnitude of improvement seems and unreasonable expectation, but a halving of the currently observed errors seems plausible.

There are many more interesting things to say about the orbital validation tests, but to do so would derail the overall narrative of this already large chapter too much. Instead, a more detailed overview of these tests can be found in Appendix H, which contain enough information to understand the tests better and also reproduce them.

Table 13.7: Command errors and angular errors, before and after filtering. Lower errors are better.

Test	E_{max} [μT]	E_{RMS} [μT]	Λ_{max} [°]	Λ_{RMS} [°]
Orbit1	3.418	1.556	16.97	5.59
Orbit1 (filt.)	2.581 (**-24.5%**)	0.483 (**-69.0%**)	7.35 (**-56.7%**)	1.63 (**-70.8%**)
Orbit2	2.159	1.075	49.62	15.22
Orbit2 (filt.)	1.017 (**-52.9%**)	0.353 (**-67.2%**)	16.39 (**-67.0%**)	3.75 (**-75.4%**)
Orbit5	3.127	2.206	4.58	2.14
Orbit5 (filt.)	1.735 (**-44.5%**)	0.842 (**-61.8%**)	2.91 (**-36.5%**)	1.16 (**-45.8%**)
Orbit6	2.915	2.119	3.92	1.96
Orbit6 (filt.)	1.780 (**-38.9%**)	0.868 (**-59.0%**)	2.35 (**-40.0%**)	1.15 (**-41.1%**)
Orbit7	3.966	2.000	6.12	2.37
Orbit7 (filt.)	2.848 (**-28.2%**)	1.279 (**-36.1%**)	5.31 (**-13.2%**)	1.86 (**-21.9%**)
Orbit8	4.249	2.565	7.67	3.76
Orbit8 (filt.)	3.867 (**-9.0%**)	1.397 (**-45.5%**)	6.35 (**-17.2%**)	2.86 (**-24.0%**)
Orbit9	3.229	1.854	4.80	2.15
Orbit9 (filt.)	1.581 (**-51.1%**)	0.840 (**-54.7%**)	2.56 (**-46.8%**)	1.15 (**-46.7%**)
Orbit10	3.020	2.217	3.53	2.09
Orbit10 (filt.)	1.395 (**-53.8%**)	0.612 (**-72.4%**)	2.31 (**-34.6%**)	0.72 (**-65.8%**)

13.6.4. Concluding the validation

One of the main questions that this chapter should address is: can the system be validated? Does it work to a satisfactory degree?

The author argues that this is indeed the case. First, the system passes verification of all its requirements, provided that some requirements had to be adjusted to a minor, acceptable degree. Second, the orbital validation tests reveal that end-to-end, the system works well and is very pleasant to work with. Generating the schedules for the tests usually took a matter of minutes[1]. Transferring these to the low-level interface was easy, calibrating the system was automatic and took no longer than polishing off a cup of coffee. And the system tracks the schedule reasonably well and the data can afterwards be analysed very comfortably with the tools from the HCT.

The system performance is not yet perfect: we have discussed several ways in which the system can be developed further, and can be relatively certain that these would lead to significant improvements to the system performance. For most validation tests that were performed, the pointing accuracy of the system can be expected to be around a few degrees on average, and within 10 degrees at worst. For certain schedules that contain very small vectors, the pointing errors can grow much larger.

Given that most CubeSats that attitude determination systems bottom out at a pointing accuracy of 10° and over, we can assert that the system is typically better and so is ready to test satellite hardware. If the test operators pay close attention to the caveats discussed in this chapter, it should be very achievable to perform critical magnetic testing as outlined in chapter 4. There will be more on this in the next chapter. If additional performance is required, the path to do so is already paved, although this will be expanded upon further in section 14.3. For the moment however, we will consider the system to be **successfully validated**.

[1] More time was spent chasing down historical satellite mission data on the internet than actually operating the software

14

Conclusion

The contents of this thesis form a research project, one that was born from the observation that the reliability of nanosatellites could be improved if testing was better facilitated. Having observed the Demonstrator System (DS) working, it is time to conclude the project. In this chapter, we return to the research question of chapter 3 and answer it with the results found throughout the thesis work. Following that there is a brief section containing some important lessons that were learned throughout this research, such as things that the author wished he would have done differently if he could go back, or practical problems that unexpectedly appeared along the way. Finally, this chapter closes off with a section on future research and work that can be done with this thesis as a stepping stone.

14.1. The research question revisited

Back in chapter 3, the research of this thesis was underpinned by the following research question:

"How can the magnetic properties of a small satellite be effectively tested?"

This research question was broken down further into four categories which expand on the word "effective". These four categories are:

1. The degree to which the testing are **representative**
2. The **reliability** of the testing
3. The efficiency of the testing in terms of **time**
4. The **cost-effectiveness** of the testing

Based on the observed notion that satellite missions with a small budget are frequently under-tested, and have been known to result in problems or outright failure, all these four categories address this in their own way. The tests have to be representative of the intended operational conditions of the satellite, for example. Furthermore, testing that can be performed in a time-efficient and cost-effective manner will decrease the likelihood that this kind of testing will be skipped by the developers.

With this as a backdrop, the thesis work then received an explicit research objective:

"To improve testing for CubeSat missions, especially those heavily constrained by schedule and budget, by improving and streamlining a testing strategy for the magnetic properties of a CubeSat."

As mentioned before, much of this thesis serves as an "argument by way of engineering" to show that magnetic testing of a CubeSat (or any nanosatellite) is feasible even on a small budget. The Demonstrator System that was developed may serve as a blueprint and a starting point for developers of such a mission, such that they are not starting this task from scratch. Besides engineering details, even aspects such as budget estimations were included in this. In regards to the word "streamlining" in the research objective, the work described in this thesis has much to offer. From the open-source nature of the software and design, which can be quickly distributed and modified, to the automated tools such as the

magnetic schedule generators and auto-calibration tool; the DS was designed to be a stepping stone rather than a stumbling block.

Let us now assess the research question. The question of how the magnetic properties of small satellites can be tested was already addressed in chapter 4. In this chapter an itinerary of tests was described, and an ideal, feature-complete testing system was defined that would perform all these tests.

However, the research question is about how these tests can be performed **effectively**, and therein lies the rub. The idealized testing system described in chapter 4 is outright prohibitive for most small satellite missions in terms of complexity, cost, and development time. What the Demonstrator System designed in this thesis shows is how one can build a relatively simple and affordable baseline system that can perform several of the tests described in chapter 4. Let us take a look at these tests again:

1. **Characterization of the satellite's net magnetic dipole**
   In its current state, the DS cannot perform this test, but it can be adapted relatively straightforwardly to do so. The Helmholtz Cage of the DS can neutralize the magnetic field vector down to the noise floor, which was proven to be several hundreds of nT. In chapter 7 we showed how this noise floor can be reduced by avoiding the elevator signal and filtering the test data, which was proven to reduce the standard deviation of the noise floor to down below 10 nT. This is good enough for a dipole measurement. What is needed is either a six-sensor magnetometer array[1], or using a single tri-axial magnetometer and spinning the satellite round on a rotating table. Both methods have their own advantages and drawbacks, but neither are particularly difficult to implement. A magnetic deperming device is quite important to have for such a test, and this will have to be procured separately; the magnetic field strengths needed for a deperming procedure (max. 5000 μT) are over an order of magnitude too high to be delivered by the Helmholtz cage itself.
2. **Characterization of the spacecraft's near-field properties**
   This test can be carried out in a limited capacity provided that the satellite under test has on-board magnetometers that can be read out in real-time. The Helmholtz cage is useful in that it can provide a precise, known magnetic reference field (or a neutral environment), and so anything that the on-board sensors measure in addition must be near-field emissions by the spacecraft. This can be done for various flight conditions. A problem is that the on-board sensors must provide reliable information, so they must be calibrated and tested beforehand. Another way of measuring the near-field effects is by acquiring a second precise milligaussmeter and use it to map the magnetic field all around the satellite.
3. **Characterization of the spacecraft resistance to AC emissions**
   This test cannot be carried out by the DS. You need a high-frequency AC field emitter and receiver to do this properly[2].
4. **Verification of any magnetic sensing equipment on the satellite**
   This can be done very well by the DS. First, the Helmholtz cage can provide precise, known static reference fields that can be used to calibrate and test any on-board sensing equipment[3]. However, the DS can also simulate long-term magnetic field envelopes similar to those that the satellite will experience in space, with an angular tracking error of within a few degrees.
5. **Verification of any magnetic actuators on the satellite**
   The DS is well-suited for this kind of testing, though it has some limitations. The best setups for this kind of test provide the satellite under test with multiple degrees of rotational freedom. A single degree of freedom is relatively straightforward with methods such as wire suspension, which the DS supports. What is unsupported is multiple degree of freedom rotation testing, since more advanced hardware is needed, and there is currently no way to let the satellite-under-test influence the magnetic field directly. A possible future upgrade of the system should contain a way to let the satellite apply a transformation matrix to the schedule, so that the satellite can 'rotate' itself in the magnetic field; not by physically rotating, but by implicitly instructing the Helmholtz cage to rotate the field in the opposite direction. Although currently unsupported, this would not be difficult to implement, and would allow for a higher degree of hardware-in-the-loop (HWIL) testing.

[1] As specified in section 5.4.5.4 of ECSS-E-ST-20-07C [40]
[2] see section 5.4.10 of ECSS-E-ST-20-07C [40]
[3] This assumes that near-field effects can be disentangled from explicit sensor malfunction

6. **General operation validation**
   The DS supports this kind of testing to a limited degree. For a satellite-under-test, the system interactions between the attitude determination system and the attitude control system can be validated quite straightforwardly. The interaction between attitude determination and control system (ADCS) and other subsystems can be tested to a more limited degree: certain parts of the power supply or radio can be artificially activated so that their effects can be observed. But general operation validation would benefit from having additional simulation capabilities, particularly a solar simulator lamp. There have been several CubeSats whose solar panel properties led to unexpected magnetic side effects (DeOrbitSail, MOVE-II), so a solar simulator would be a good upgrade for the system in order to test for such problems.

In summary, the DS in its current state can simulate the magnetic field envelope of a satellite in orbit and also create a magnetically neutral environment using the Helmholtz cage. It is not currently possible to perform magnetic dipole measurements, AC field compatibility, or very comprehensive HWIL testing. However, if the cost and time budgets of the mission allows it, the system can also be expanded in several ways to achieve better testing coverage. This built-in adaptability is one of the strongest features of the Demonstrator System.

Let us also consider the four categories that expand the research question and discuss for each how the research in this thesis addresses them.

1. **Representative testing:** There is little point in performing a satellite test if it does not represent the intended operational conditions. Most CubeSats are launched into low Earth orbit (LEO), where they orbit close to the Earth and constantly pass through different regions of the Earth magnetic field (EMF). To this end, the DS includes an orbit propagator for elliptical low-Earth orbits that simulates the motion of the satellite and calculates the magnetic field experienced by the satellite. This is then converted to a form that can be reproduced by the Helmholtz cage of the DS to within a few percent accuracy. As such, placing a satellite-under-test in the Helmholtz cage will therefore mimic the conditions as they would be during actual operation of the satellite.
2. **Reliable testing:** The DS was designed in such a way that two identical tests could be performed at different times, and the results would be minimally different. The magnetic envelope schedules that actuate the Helmholtz cage can be saved and stored for later, and can be encoded by only a small set of "generation parameters". The data collection system was proven to be adequately reliable. As for the hardware side, much attention was spent on characterizing any local magnetic field disturbances at the test location and devising strategies to mitigate this variable. In addition, the system is capable of automatic calibration so that even if environmental conditions can differ from day to day, the system will operate with high linearity.
3. **Time-efficient testing:** The DS was designed with explicit requirements to perform tests quickly. To this end, much of the setup work for a test can be done separately: for example, the programming of the magnetic envelopes can be done before-hand, away from the test location. Calibration can be done in a matter of minutes and is automated. Much time was spent on developing good data visualization tools, which allows users to spot any issues during quickly, avoiding scenarios where test have to be repeated at a later date because of hidden issues that were found afterwards. Furthermore, the software of both the low-level interface and the high-level interface are flexible enough that much can be automated further according to specific needs, leaving the door open for future improvements.
4. **Cost-efficient testing:** The word "cost-effective" does not mean much without an expectation of what kind of cost would be acceptable. The cost for the DS was calculated to be under €12.5k (see Appendix A), which was considered an acceptable cost considering that a low end budget for a small satellite mission can be estimated in the region of several hundreds of thousands of euros. The cost-effectiveness was further improved by designing the system in such a way that it used commercially-off-the-shelf (COTS) components as much as possible, and one of the largest custom components (companion board PCB) was completely provided as blueprint, such that it could be produced in-house. The DS was also designed to allow for component substitutions in certain areas, which combined with the high number of COTS components in the design improves the chances that certain parts do not even have to be purchased if they are already available. Finally, a

major component of the system is the software, which is publicly available to anyone completely free of cost.

In these ways, the Demonstrator System that was designed as part of this thesis work answers the research question. To summarize this in a single answer:

"The magnetic properties of a small satellite can be effectively tested by using a design such as the Demonstrator System, which allows for representative and reliable testing that should be affordable even to satellite missions with small budgets."

14.2. Design Reflections and Lessons Learned

As is the case in any decently-sized engineering project, there will always be unexpected problems and findings that are discovered along the way. In addition, there will be choices along the way that in hindsight appear uninformed or would be approached differently next time. This section will meander past a few such experiences for the work described in this thesis.

One of the truly unique things about this thesis work is the degree to which it is end-to-end; it covers everything from problem definition to validation testing of the end product. This is a great educational experience, because it allows one to face the consequences of one's own design choices. This motivates making choices carefully and taking ownership of the problems that come up along the way.

14.2.1. Lessons in System Engineering

Let us first touch upon the system's engineering side of things. The requirements and overall system design were not bad as such things go, but it did have to be adjusted along the way. The main problem was to do with scoping: a more rigorous scoping of topics and tasks could have more tightly controlled the overall project scheduling. There are also a number of requirements that should have been posed differently; making the control volume spherical and using the angular error to express the overall system performance should have been obvious from the start. In addition, a few "optimization goals" ran parallel to the requirements, which was not a bad approach in the end, but could probably have been captured better in better requirement formulations. Still, enough care was taken during the prototyping and integration process that most critical design aspects came up naturally, even if not formally encoded in the requirements.

Without a doubt, most lessons were learned in the department of electronics design. The author was at the start of the thesis not very experienced with analog circuit design, and no prior knowledge of PCB design was present. Despite the relative simplicity and prevalence of H-bridge circuits, properly driving one with N-channel MOSFETs based on first principles was a difficult thing to get right. There were also a few important lessons learned about signal integrity and input impedance of devices, like the issue that came up from driving the half-bridge drivers with an unbuffered signal. When it comes to debugging PCBs, oscilloscopes and trace cutting tools are your best friend.

14.2.2. Lessons in Electrical Engineering

Perhaps the greatest lesson in the department of electronics was that even expensive, lab-grade power supplies can perform hilariously bad when driving certain inductive loads. The BK Precision power supplies could never have been used to drive the Helmholtz cage, but that this was discovered so late in the process could have been avoided: an important lesson. Fail early! The difference with the Delta Elektronik power supplies was like night and day, and despite having not quite the desired compliance voltage capability, they are excellent power supplies for driving the Helmholtz cage.

On the topic of failing early, during prototyping the electronics, some strategic purchases of expander boards for SMD chips could have been very helpful. Being able to breadboard comprehensively before placing an order for a PCB can decrease the time spent debugging the PCB, prevent unwanted surprises, and may even prevent a board iteration from being necessary. You can also never have too many test points in your PCB design, and making opportunities for fixes and bodges later on can save a lot of headache.

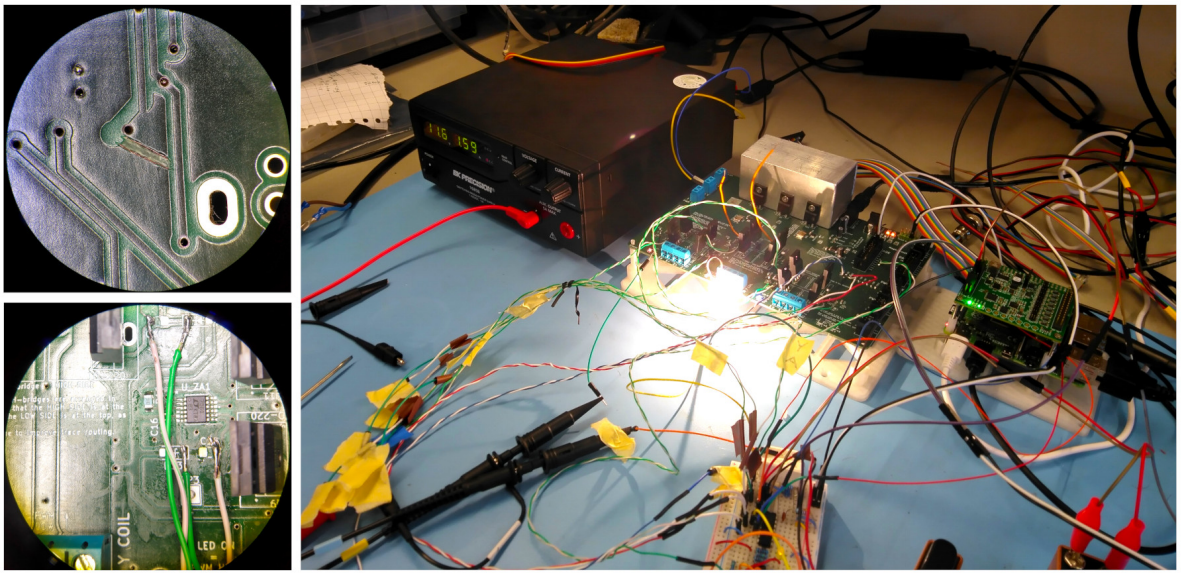


Figure 14.1: Top left: Minor PCB defect from forgetting to flood-fill the ground planes. This one could be fixed by cutting away some ground plane copper. Lesson: always flood-fill your ground planes again before exporting your gerbers. Bottom left: give yourself room for boding or be doomed to work with what you've got. Right: Board testing with an incandescent bulb as an *ad hoc* dummy load. If it works, it's not stupid. However, the sheer number of bodging wires would later cause undesirable self-oscillatory behaviour in the board.

The LTC7060 half-bridge drivers are an interesting and seemingly vulnerable chip. Due to them having a thermal pad on the bottom, soldering them properly was not trivial, and they would sometimes not survive the ordeal. Desoldering them would typically destroy them internally. What could have been done to help here is place thermal vias into the footprint of the thermal pad, which creates an option to solder them from the bottom by wicking solder and flux up through the vias. When these chips would break, they would break in interesting, unexpected ways: sometimes creating a dead short between supply and ground, sometimes becoming a low resistance short, and sometimes passing all pin tests only to then not work at all. This made it necessary to check each and every one of them after soldering, and almost all the stock that was bought for several boards ended up being used up.

The MOSFETs were an excellent choice. Although relatively expensive, their exceedingly low on-resistance caused them to never be anything but room temperature, even when passing 5 A of current. The lack of necessary cooling made for a much more straightforward board design, and more attention could be paid to a convenient layout for the 5 A capable PCB traces, with equal impedance for both H-bridge states. There are also similar MOSFETs that come in a D2PAK, which is a surface-mount package rather than a through-hole one. These were at one point also considered, as using them reduces the overall board height, it looks more elegant, and they are compatible with pick-and-place manufacturing. However, in practice the fact that TO-220 MOSFETs were used helped the board layout; the through-hole mounting holes also act as high current-capacity vias which the trace routing makes clever use of.

Electronic design is a place where the devil is in the details. At some point, a day of debugging the non-functional current sensing ended fruitlessly. Afterwards, it was discovered that the current sense resistor that should have been 5 $m\Omega$ was in fact a 0.5 $m\Omega$ variant. The part numbers were so similar (`CSS2H-3920R-L500F` vs. `CSS2H-3920R-5L00F`) that unbeknownst to everyone the wrong version had been ordered. The second iteration of the board used different current sense resistors partly because they had less error-prone part numbers.

Another example of the devil in the details was encountered whilst debugging the first iteration of the companion board. At this point, many bodging wires were attached to the board to inject and extract signals. At some point, a strange defect started showing up in the board that caused the MOSFETs to start switching at rates of a few MHz. After many attempts at fixing this, the issue went away by itself after V2.2 of the PCB was ordered. Why? In all likelihood, the tangle of bodging wires was inducing

self-oscillatory behaviour in the board, because the issue would come and go as you would slap around the bunch of spaghetti wiring.

Are there many things that would be done differently with the electronics design, for example if the companion board had to be designed again? As it turns out: *not really*, because the companion board was already redesigned once, given that two design iteration were performed. At this point, the companion board could benefit from having the dropper diodes re-assessed, as this turned out to reveal some problems during the validation process. It may be better if the diodes were left out (even if this degrades the transient voltage protection), or replaced with an ideal diodes [79] or another solution. In addition, the H-bridge is currently designed to recirculate excess energy at the moment that the H-bridge switches. In a motor driver, this is no problem, but for this application it may explain some of the tracking offsets that were observed during validation. The circuit may be redesigned in such a way that the coil energy can be dumped somewhere when the current changes rapidly.

Other than that, the companion board turned out rather nice, considering the starting experience in PCB design was non-existent. The coloured light emitting diodes (LEDs) were a great help during verification and validation testing to inform the operator of what the system was doing. The added flexibility of the solder jumper options, routing options using header pins, and exposed auxiliary ADC/DAC pins turned out to be very helpful also.

14.2.3. Lessons in Software Development

It is estimated that over 40,000 lines of code were written during the thesis work. Considering the size of the code base and the complexity of the software, many things were done right and are looked back upon as good choices.

The choice to exclusively use Python as a programming language was a mixed blessing: it does make the code accessible to a wide user base, makes the code portable, and does bring all the benefits of a single-package solution. However, it made certain things difficult: the TCP protocol had to be bench-marked carefully to ensure that it was fast enough to be usable, rather than being a bottle-neck to the system. In addition, writing low-level code for interfacing with the hardware on the Raspberry Pi could be a bit of a disaster: the current code seems very inefficient and bottle-necks the system in a number of important ways (see subsection 13.6.2). To address these, the author recommends that a custom driver is written (likely in C) to interact with the CN-0554 board more effectively, and if possible tie these drivers to a low-level hardware clock so that de-synchronization issues can be overcome. At the time of development the Python-based API for this board, and the corresponding documentation, simply were not good enough.

Other than these issues, Python is very quick to develop and adjust. Because much of the system can be used using Python, this creates the ability for all sorts of automated scripting, and this was effectively leveraged in many places. Using Qt5 as a framework with the Python-based bindings turned out to be an excellent choice. The graphical user interface (GUI) of the Helmholtz Cage Toolkit (HCT) looks good and is easy to work with. The highly class-organized structure of Qt5 allows for very organized code structuring, which helped keep the GitHub codebase from quickly turning into a rat's nest of spaghetti code. Still, the GUI code was refactored several times to keep things organized.

Many visualization tools were developed as part of the thesis, and these were significant investments. However, they made debugging both software and hardware much easier in the end, and are sure to help with user adoption. Many of the plots in this thesis were generated using these tools. In particular, the `PyQtGraph` library was extremely useful for creating both 2D and 3D plots with high efficiency, effectiveness, and customisability. This library uses OpenGL bindings underneath most of its functions, so that a large amount of data can be visualized without problem. It is a lot more verbose than the popular `Matplotlib` and also has a steeper learning curve, but it was the right tool for the HCT. Hopefully development of this library will continue in the future. One downside of `PyQtGraph` is that the 3D plotting environment cannot be used on the low-level interface: the Raspberry Pi and `PyQtGraph` use mutually exclusive versions of OpenGL that prevent this from working. However, given the aforementioned performance issues of the software on the Raspberry Pi, this would not have been used much anyway.

From an early point in development, it was made a requirement that any GUI interface would not

supplant the underlying API, and turned out to be a good choice. It meant that any part of the HCT could easily be manipulated using scripting, which not only helps with user adoption, but also was a great help for writing test scripts and all sorts of extra utilities.

One aspect of the software development that should be done different is the approach to documentation: there are many automated tools that can generate and organize documentation directly from annotated code. Such a tool should have been used from an early point onward to keep on top of things and encourage code being written in a more "documentable" manner. However, at this early stage, it was not yet clear how large the codebase was to become.

Another few interesting lessons were learned during the ambient field measurements. Analog measurements were gathered by a digital acquisition card and logged by a computer. At first, the data rate of this setup was limited to a few dozen S/s, because Python code was manually polling each sample. However, after learning how to exploit the hardware buffering of the acquisition card, this could be pushed to the 10 kS/s. Then a new problem arose: at these data rates, a day-long test would generate gigabytes of raw test data. To process it requires reading the data file, which normally has to be loaded into the random access memory (RAM) of the computer first. If the data is larger than the amount of available RAM, you run into trouble. There is a whole body of literature on writing data processing code for larger-than-RAM datasets, but generally none of the methods are straightforward.

For the thesis work, the problem was solved as follows: The large data sets were cut up into chunks of 1 hour each. The exact data points on the hour marked were found using binary search, which does not require read the whole data file. For the non-spectral data processing, the 2500 S/s per-channel data was artificially undersampled and stitched back together. The spectral data processing benefitted little from the whole data set, and so was done with the full-resolution hour-long chunks instead. More details can be found in the `Gaussmeter Utilities` repository [75]. Lesson learned: if you can avoid it, prevent generating larger-than-RAM datasets.

Ultimately, the future success of the developed software lies in the degree of user adoption, and the amount of willingness there is to keep developing it. If nobody will maintain the code, the HCT will likely become derelict over time and fade into obscurity, as is the fate of many software tools.

14.2.4. Lessons Learned - In Summary

In summary, the lessons that were learned roughly boil down to the following one-liners:

1. When designing, carefully plan the scope, requirements, verification, and scheduling.
2. When designing within unfamiliar territory, there will always be more problems than you expect, and sometimes the problem is you.
3. It is in man's nature to make mistakes. Ensure that you make them early.
4. If it's stupid but it works, it's not stupid.
5. Sensitive magnetic equipment and building elevators do not mix well.
6. An abundance of test points and bodging accessibility is an important feature of your PCB design.
7. Good data visualization tools are a serious investment, but often a worthwhile one.
8. Writing software is easy. Writing good software less so. Future-proofing software is hard.

Whilst these are of course worded somewhat amusingly, it does indicate that the thesis work did not just teach unconnected specifics, but also more general lessons that can be applied to engineering more broadly.

14.3. Future Development

Despite a tenacious attempt at delivering a flawless system, the last few chapters have revealed many flaws. Therefore, there is room to further improve the DS in the future, after this thesis work is concluded. This section aims to provide a concise yet comprehensive summary of open issues, as well as suggestions on how to address them.

Software development

1. After the thesis concludes, re-organise the Helmholtz Cage Toolkit codebase for easy adaption by others and make it less specific to the thesis. The current repository state can then be moved to a separate branch for preservation.
2. Improve the documentation by moving away from the thesis format to an auto-generated, wiki-like documentation system, such as Sphinx. During the thesis, there was no time to do this.
3. The HCT code is due a refactoring. Several classes, such as the OrbitalPlot class, have one or more variants that were written for specific purposes. These classes should be unified where possible, to pave the way for future expansion of the codebase.
4. Rewrite the drivers for the CN-0554 ADC/DAC board to improve their efficiency. This will likely fix the freeze-up issue and the time de-synchronization issue that are currently bottle-necking playback of schedules. Then stress-test the system, for example at higher command rates.
5. Improve the auto-calibration routine to ensure that no offset or scaling problems can be seen for the whole range of [-200, 200] μT. This should centre the system noise around zero.
6. Replace the feed-forward control of the Helmholtz cage with feed-back control, such as PD or PID. This is not straight-forward, because it becomes a requirement that the reference sensor measures the Helmholtz cage field alone. It was observed that a nearby compass rotating in sympathy will very noticeably show up in the test data; a nearby satellite-under-test may affect it even more. Also as with any feedback control system, controller stability will have to be assessed carefully.
7. Allow attitude control commands from a satellite-under-test to be interfaced with the DS, so that active attitude control can be simulated by rotating the Helmholtz cage schedule in the opposing direction.

Hardware development

1. The stepped tracking error caused by the dropper diodes should be addressed, for example by removing the these diodes. However, this strips away one layer of transient voltage protection for the positive output of the power supply. In addition, the power supply may not be linear all the way down to 0 V, but may cut out similarly to the 1685 power supplies at about 0.4 V compliance voltage. It may be possible to use a diode-like arrangement that has a zero voltage threshold (an "ideal" diode), or the current may be biased in some way, for example using a constant-current sink in parallel with the coil channel. In the latter case should also be investigated whether the power supply output is not at risk from back-EMF overvoltage spikes.
2. The tracking performance from the H-bridges likely suffers from the fact that the coil energy that is released during rapid current changes is recirculated back into the system. To address this, the coil energy needs to be redirected somewhere else.
3. At high operational currents, the X and Y coils have a tendency to slightly wobble around from Lorentz forces. This changes their direction and will introduce a tracking error. It would be best to measure the coil spacing exactly, adjust the attitude of the coils to perfection, and then fix the coils into place by affixing them to one another.
4. Although the companion board looks nice, eventually it should be integrated into the supply cart. However, the companion board also provides user information using its LEDs, so it is recommended to install it behind an acrylic window.
5. It may be worth performing a sensitivity analysis of the system performance to companion board temperature. If the shunt resistors heat up too much, they may start to produce an error in the current measurements, for example. However, if the dropper diodes in their current state are removed, this should remove a potentially 12 W source of heat on the board, and heat may not be an issue at all anymore.
6. Install a reference sensor nearer to the elevator system, so that its effects can be measured and subsequently removed from the test data.
7. Install a small grid of six magnetometers such that far-field magnetic emissions can be measured, and the magnetic dipole of a satellite can be measured. This can potentially also be achieved by rotating the satellite at a certain distance from the reference Gaussmeter.
8. Integrate a solar simulator into the system to be able to test the effect of solar panel illumination on magnetic properties.

References

[1] Ivan Avaliani. *Artstation*. 2022. URL: https://user-327423.artstation.com/ (visited on 04/01/2024).

[2] Mathias Nowak et al. "Short life and abrupt death of PicSat, a small 3U CubeSat dreaming of exoplanet detection". In: *arXiv: Instrumentation and Methods for Astrophysics* (2018). DOI: 10.1117/12.2313242.

[3] Youngbum Song et al. "CANYVAL-X: Operational Scenario and Strategy". In: *2018 SpaceOps Conference*. May 25, 2018. DOI: 10.2514/6.2018-2636. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2018-2636. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2018-2636.

[4] Amandine Denis et al. "QARMAN re-entry CubeSat: from design and pre-flight testing to post-flight lessons learned". In: 2022. URL: https://iafastro.directory/iac/archive/browse/IAC-22/B4/9-GTS.5/69263/.

[5] Sebastian Rückerl et al. "First Flight Results of the MOVE-II Satellite". In: Aug. 2019.

[6] Lucien Volk et al. "New Results and Lessons Learned from the MOVE-II and MOVE-IIb CubeSats". In: Aug. 2022. URL: https://digitalcommons.usu.edu/smallsat/2022/all2022/224/.

[7] National Geospatial-intelligence Agency (NGA). *Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems*. Version 1.0.0. July 8, 2014.

[8] M. Swartwout. *CubeSat Census*. Apr. 4, 2023. URL: https://sites.google.com/a/slu.edu/swartwout/cubesat-database/census.

[9] NASA. *What is CAPSTONE?* URL: https://www.nasa.gov/smallspacecraft/capstone/.

[10] Stefano Speretta et al. "LUMIO: A CubeSat to Monitor Micro-meteroid Impacts on the Lunar Farside". In: Mar. 2022, pp. 1–8. DOI: 10.1109/AERO53065.2022.9843311.

[11] Maeve Doyle et al. "Mission testing for improved reliability of CubeSats". In: *International Conference on Space Optics* (2021). DOI: 10.1117/12.2600305.

[12] Michael Swartwout and Clay Jayne. "University-Class Spacecraft by the Numbers: Success, Failure, Debris. (But Mostly Success.)" In: (2016).

[13] Martin Langer and Jasper Bouwmeester. "Reliability of CubeSats – Statistical Data, Developers' Beliefs and the Way Forward". In: Aug. 2016.

[14] Abdulaziz Alanazi and Jeremy Straub. "Statistical Analysis of CubeSat Mission Failure". In: (2018).

[15] J. Bouwmeester, A. Menicucci, and Eberhard Gill. "Improving CubeSat reliability: Subsystem redundancy or improved testing?" In: *Reliability Engineering & System Safety* (2022). DOI: 10.1016/j.ress.2021.108288.

[16] Sarah Dolan. *Building and Operating CubeSats - Successes and Struggles : Montana State's Experience with FIREBIRD & IT-SPINS (presentation)*. 2022. URL: https://www.youtube.com/watch?v=I4FetOoJwhM&t=11583s.

[17] NASA Jet Propulsion Laboratory. *The MarCO Mission Comes to an End*. Feb. 5, 2019. URL: https://www.jpl.nasa.gov/news/the-marco-mission-comes-to-an-end.

[18] Brian E. Gilchrist et al. "Electromagnetic Wave Scattering Experiments in Hall Thruster Plasma Plumes". In: 1998.

[19] Y. Zhang et al. "A test method for electromagnetic interference caused by plasma plumes". In: May 2016, pp. 1–5. DOI: 10.1109/AeroEMC.2016.7504594.

[20] Vaclav Mach et al. "Electromagnetic compatibility of Raspberry Pi development platform in near and far-field". In: *2017 Progress in Electromagnetics Research Symposium - Fall (PIERS - FALL)*. 2017, pp. 2466–2472. DOI: 10.1109/PIERS-FALL.2017.8293550.

[21] J. P. Catani and D. Payan. "Electrostatic behaviour of materials in a charging space environment". In: *Proceedings of the 2004 IEEE International Conference on Solid Dielectrics, 2004. ICSD 2004.* Vol. 2. 2004, 917–927 Vol.2. DOI: 10.1109/ICSD.2004.1350582.

[22] Johannes Jacobus Monster. *Literature Study Report*. Version 1.0. May 2, 2023.

[23] Brazilian Space. *NANOSATC-BR2: Brazilian Space apura informações sobre o status do satélite*. Aug. 11, 2021. URL: https://brazilianspace.blogspot.com/2021/08/nanosatc-br2-brazilian-space-apura.html.

[24] Iver H. Cairns et al. *INSPIRE-2 and Lessons Learned*. Ed. by Australian CubeSat Conference 2019. 2019. URL: https://www.acser.unsw.edu.au/sites/acser/files/CAIRNS_INSPIRE_CubeSatConf_2019_2.pdf.

[25] Vladislav Pustõnski. *ESTCube-1 ceased working after 2 years in orbit*. May 2015. URL: https://web.archive.org/web/20180311140551/https://www.eas.ee/kosmos/en/estonian-space-office/news/article/389-estcube-1-ceased-working.

[26] Jean-Noël Pittet et al. "Space debris attitude determination of faint LEO objects using photometry: SwissCube's cubesat study case". In: 2017. DOI: https://doi.org/10.7892/BORIS.106923. URL: https://conference.sdo.esoc.esa.int/proceedings/sdc7/paper/291/SDC7-paper291.pdf.

[27] EPFL. *SWISSCUBE - The first swiss satellite*. 2013. URL: https://archiveweb.epfl.ch/swisscube.epfl.ch/.

[28] J. Bouwmeester et al. "Utility and constraints of PocketQubes". In: *CEAS Space Journal* 12.4 (Dec. 2020), pp. 573–586. ISSN: 1868-2510. DOI: 10.1007/s12567-020-00300-0. URL: https://doi.org/10.1007/s12567-020-00300-0.

[29] David Barnhart and Tanya Vladimirova. "Satellite-on-a-Chip Development for Future Distributed Space Missions". In: vol. 2006. Jan. 2006. DOI: 10.1115/CANEUS2006-11045.

[30] Phalguni Mathur and Sujith Raman. "Electromagnetic Interference (EMI): Measurement and Reduction Techniques". In: *Journal of Electronic Materials* 49 (5 May 1, 2020), pp. 2975–2998. DOI: 10.1007/s11664-020-07979-1.

[31] Jialong Dai et al. "Research on Magnetic Characteristics of Small UAV for Aeromagnetic Measurement". In: *IOP Conference Series: Earth and Environmental Science* 310.3 (Aug. 2019), p. 032068. DOI: 10.1088/1755-1315/310/3/032068. URL: https://dx.doi.org/10.1088/1755-1315/310/3/032068.

[32] Dimuthu Wanasinghe and Farhad Aslani. "A review on recent advancement of electromagnetic interference shielding novel metallic materials and processes". In: *Composites Part B: Engineering* 176 (2019), p. 107207. ISSN: 1359-8368. DOI: https://doi.org/10.1016/j.compositesb.2019.107207. URL: https://www.sciencedirect.com/science/article/pii/S1359836819312569.

[33] F.M. Poppenk, R Amini, and GF Brouwer. "Design and Application of a Helmholtz Cage for Testing Nano- satellites". In: (2007).

[34] Charles Grassin. *Helmholtz Magnetic Simulation Environment*. ECE Paris, Mar. 21, 2019. URL: https://charleslabs.fr/en/project-ECE3SAT+-+Helmholtz+Coils.

[35] Dario Modenini et al. "A Dynamic Testbed for Nanosatellites Attitude Verification". In: *Aerospace* 7.3 (2020). ISSN: 2226-4310. DOI: 10.3390/aerospace7030031. URL: https://www.mdpi.com/2226-4310/7/3/31.

[36] Guglielmo Cervettini et al. "Development and Experimentation of a CubeSat Magnetic Attitude Control System Testbed". In: *IEEE Transactions on Aerospace and Electronic Systems* (2020). DOI: 10.1109/taes.2020.3040032.

[37] Abdelmadjid Lassakeur and Craig Underwood. "Magnetic cleanliness program on cubesats for improved attitude stability". In: *2019 9th International Conference on Recent Advances in Space Technologies (RAST)*. IEEE. 2019, pp. 123–129.

[38] Serhiy Belyayev et al. "Practical analysis and implementation of a low magnetic contamination CubeSat platform". In: *The 4S Symp*. 2016.

[39] Astrofein. *ACS Test Bed*. 2023. URL: https://www.astrofein.com/en/products/acs-test-stand/.

[40] European Cooperation for Space Standardization. *ECSS-E-ST-20-07C - Electromagnetic Compatibility*. Version 2. Jan. 3, 2022. URL: https://ecss.nl/standard/ecss-e-st-20-07c-rev-2-electromagnetic-compatibility-3-january-2022/.

[41] European Cooperation for Space Standardization. *ECSS-E-ST-10-03C - Testing*. Version 2. May 31, 2022. URL: https://ecss.nl/standard/ecss-e-st-10-03c-rev-1-testing-31-may-2022/.

[42] Hans Christian Ørsted. "Experiments on the effect of a current of electricity on the magnetic needles". In: *Annals of Philosophy* (1820).

[43] W.M. Frix, G.G. Karady, and B.A. Venetz. "Comparison of calibration systems for magnetic field measurement equipment". In: *IEEE Transactions on Power Delivery* (1994). DOI: 10.1109/61.277684.

[44] C. D. Graham B. D. Cullity. *Introduction to magnetic materials*. 2nd ed. Wiley-IEEE Press, 2008. ISBN: 9780471477419.

[45] H.A. Wheeler. "Inductance formulas for circular and square coils". In: *Proceedings of the IEEE* (1982). DOI: 10.1109/proc.1982.12504.

[46] F M Poppenk. "Design and Testing of an Attitude Control System for a Nano Satellite [SSE-TUD-TW-0433]". TU Delft, 2009.

[47] TE Connectivity. *FX2 Relay datasheet*.

[48] The Engineering ToolBox. *AWG - American Wire Gauge Current Ratings [online]*. 2003. URL: https://www.engineeringtoolbox.com/wire-gauges-d_419.html.

[49] Krister de Ridder. *Helmholtz Cage manual (EXT-MA-0001)*. 2012.

[50] Delta Elektronika B.V. *Datasheet ES 300 - Series 300W DC POWER SUPPLIES*.

[51] Goddard Engineering and Technology Directorate. "Spacecraft Magnetic Testing Facility". In: (). URL: https://etd.gsfc.nasa.gov/capabilities/facilities-listing/spacecraft-magnetic-test-facility/.

[52] Laszlo Balogh. *Application Report - Fundamentals of MOSFET and IGBT Gate Driver Circuits [SLUA618A]*. Tech. rep. Texas Instruments, 2018.

[53] Infineon. *Application Note - Power MOSFET basics*. Tech. rep. 2023.

[54] Analog Devices. *LTC7060 - 100V Half-Bridge Driver with Floating Grounds and Adjustable Dead-Time*. Tech. rep. Version Rev. B. 2022.

[55] John Linsley Hood. *The Art of Linear Electronics*. 2nd ed. 1998. ISBN: 0750637463.

[56] International Rectifier. *Datasheet - IRF520NPbF*. May 11, 2003.

[57] Bourns. *Datasheet - Model CSS2H-3920 series current sense resistor*. 2023.

[58] Texas Instruments. *Datasheet - INAx290 2.7-V to 120-V, 1.1-MHz, Ultra-Precise, Current-Sense Amplifier*. 2021.

[59] Littelfuse. *Datasheet - L4CL Series - Four-Terminal Surface Mount Resistor*. Apr. 17, 2023.

[60] Ohmite. *Datasheet - LVK Series Four Terminal High Precision Current Sense*. 2023.

[61] Vishay. *Datasheet - WSK2512*. Dec. 11, 2023.

[62] Vishay Precision Group. *Datasheet - CSM2512S*. May 3, 2011.

[63] Edward Mullinas et al. "SBOA204A - How to Properly Configure Unused Operational Amplifiers". Version Rev. A. In: *Texas Instrument Application Note* (2018). URL: https://www.ti.com/lit/an/sboa204a/sboa204a.pdf.

[64] Marcus O'Sullivan. "Optimize High-Current Sensing Accuracy by Improving Pad Layout of Low-Value Shunt Resistors". In: *Analog Devices* (2012).

[65] Infineon. *Datasheet - IPA045N10N3 G MOSFET*. Jan. 22, 2016.

[66] Philip C. Todd. "SLUP100 - Snubber Circuits: Theory, Design and Application". In: *Unitrode Corporation* (1993).

[67] Jim Lepkowski. "AND8229/D - An Introduction to Transient Voltage Suppression Devices". Version Rev. 0. In: *ON Semiconductor* (2005).

[68] ST. *Datasheet - STTH8L06 Turbo 2 ultrafast high voltage rectifier*. 2012.

[69] Vishay. *Datasheet - BZT03-Series Zener Diodes with Surge Current Specification*. Sept. 11, 2019.

[70] Littelfuse. *Datasheet - 1.5KE Series*. Oct. 30, 2020.

[71] Littelfuse. *Datasheet - 1.5SMB Series*. May 12, 2023.

[72] Littelfuse. *Datasheet - 15KPA Series*. Aug. 27, 2020.

[73] Analog Devices. *Datasheet - ADP3303 High Accuracy anyCAP 220mA Low Dropout Linear Regulator*. 2017.

[74] Johan Monster. *Helmholtz Cage Toolkit source code repository*. 2023. URL: https://github.com/Hans-Bananendans/Helmholtz-Cage-Toolkit.

[75] Johan Monster. *Gaussmeter-Utilities code repository*. 2023. URL: https://github.com/Hans-Bananendans/Gaussmeter-Utilities.

[76] TIOBE Software. *TIOBE index for september 2025*. URL: https://www.tiobe.com/tiobe-index/.

[77] National Centres for Environmental Information (NCEI). *IGRF Health Warning, Errors, and Limitations*. 2010. URL: https://www.ncei.noaa.gov/products/international-geomagnetic-reference-field/health-warning.

[78] Michele Trenti et al. *SpIRIT Mission: In-Orbit Results and Technology Demonstrations*. 2024. DOI: https://doi.org/10.48550/arXiv.2407.14034. arXiv: 2407.14034 [astro-ph.IM]. URL: https://arxiv.org/abs/2407.14034.

[79] Texas Instruments. "Application note - Basics of Ideal Diodes (SLVAE57B)". Version Rev. B. In: *Texas Instrument Application Note* (2021). URL: https://www.ti.com/lit/an/slvae57b/slvae57b.pdf.

[80] National Centers for Environmental Information. *International Geomagnetic Reference Field (IGRF)*. 2024. URL: https://www.ncei.noaa.gov/products/international-geomagnetic-reference-field.

[81] Karel F. Wakker. *Fundamentals of Astrodynamics - AE4874 course reader*. 2015.

[82] Dirk Brouwer and Gerald M. Clemence. *Methods of Celestial Mechanics*. 1961.

Part IV

Appendices

A

Demonstrator System budget

In this section, an estimated budget for the Demonstrator System (DS) is presented. This budget is used as part of the valiadation of requirement **DS-18** in chapter 13, but it can also be considered a rough bill of materials (BOM) cost for anyone interested in reproducing this system.

A.1. Summarized budget estimation

Budget item	Qty.	Cost [€]	Notes
Helmholtz cage structure	1	1241.82	See Table A.2
Heavy enameled copper wire (12 AWG)	1	3928.90	(≈ 4000 meters)
Delta Elektronik ES030-10	3	2670	
Tri-axial milligauss meter	1	1600	
Interface computer	1	850	
Lump sum cost for enclosure + mounting		150	
Companion board PCB	1	209.15	(order of 5 boards)
Companion board components	2	221.36	(2 batches of parts)
Raspberry Pi 4B	1	50	
CN0554 ADC/DAC board	1	116	
Lump sum cost for electronics extras		100	
Sum		11137.23	+10% margin
TOTAL COST		**12250.95**	

A.2. Budget explanation and rationale

The cost of the Helmholtz cage structure is estimated based on a separate budget, which can be found in section A.3.

The price of the "magnet wire", which is annealed copper wire heavily coated by enamel, was difficult to estimate. For an order of 4 kilometers of such wire ((almost 120 kilograms), only industrial suppliers would suffice, and none of those who were approached seemed interested in even providing a quote for such an order. Based on third-party resale prices for enameled copper wire, a price as low as 0.80 euro per meter could be estimated, but to remain conservative, a price of 1 euro per meter was used. The price of the copper wire is **about a third of the total raw cost**, meaning that the total estimated cost of the system could change substantially based on the price estimation of the copper wire.

For the cost of the Delta Elektronik ES030-10 power supplies, a quote for an order of three units was requested. Based on this quote, the current price for these power supplies is 890 euros per unit.

The tri-axial milligauss meter that the system currently uses is a model that AlphaLab no longer appears to sell. The model that appears most similar in AlphaLabs catalogue contains some critical drawbacks that likely make this device unsuitable for the application at hand. A detailed discussion about this issue and potential solutions can be found in Appendix B. For the purposes of this budget, it means that the cost of this device has a large uncertainty. It has currently been set at 1600 euros, but in practice, it may very well be double or half that.

The budget includes an item for a simple desktop computer which can serve as the high-level interface of the system. This does not have to be a fancy computer: it just needs to have a screen, keyboard, mouse, an ethernet connection, and reasonable computing power. The test system as a whole benefits very little from the interface computer having high computing power. There is a good chance that an older computer can be repurposed for this, decreasing or nullifying its allocated budget. Still, a cost of 850 euros has been included for a new, basic desktop computer.

The companion board and its components have already been ordered before as part of the thesis, and therefore their cost is available with high certainty. For the PCBs, the cost for an order of five unpopulated boards has been used. This is the cost of the order that was placed at Eurocircuits for V2.2 of the companion board. There is a possibility that there is a slight reduction in cost if ordering three boards instead of five, but in general, the price for manufacturering printed circuit boards does not decrease when ordering fewer than that, if it is possible at all.

The parts of the components are the sum of the components as ordered at Digikey for V2.2 of the companion board. The budget assumes that these component are soldered by hand, so no pick-and-place costs are accounted for. Instead, the budget includes two batches of components so that a full set of spares is available. If the population of the PCBs is outsourced, fewer components may be needed, but the manufacturing costs will very likely be higher; using pick-and-place manufacturing for a single board is hardly ever economical.

The Raspberry Pi 4B has been listed as 50 euros, rounded up from a listing price of 47.15 euros at Digikey. It is possible that a Raspberry Pi 5 can be substituted for additional features and computing power, but this will not be investigated as part of this thesis.

The CN0554 is formally an evaluation board by Analog Devices and may become difficult to source in the future. Its price at the time was 116 euros. If the board can no longer be sourced, 116 euros is available for a replacement.

The budget also includes two lump sum items: one lump sum for additional supplies and components for the electronics, and another lump sum for enclosing the electronics and other devices. For the latter, one can consider rails for rack mounting, project boxes, storage, etc.

A 10% margin has been added to the whole budget to account for some of the uncertainty in the cost estimations.

A.3. Budget of structural parts

The first item of Table A.1 is based on a cost estimation of the Helmholtz cage structure, if its materials were to be purchased at contemporary market prices. This was necessary because the structure was built many years ago, but the cost would almost certainly have increased significantly since.

The price of the MK 4040 and 4080 profiles were based on the listed prices of the company AluxProfile[1] and compared to several other European sources, to check that these prices would be representative. A 5% margin is used to account for offcuts and wastage. The other MK profile accessories, such as L-connectors and 3-way connectors, were based on the AluxProfile listed prices.

The price of 25mm x 38mm aluminium U-profiles, such as the ones that form the backbone of the copper wire coils, the price was estimated at 10 euros per meter. The listed prices vary significantly and depend on other aspects such as the quality of the finish.

The Helmholtz cage uses four corner pieces for each coil, so 24 of such corners in total. These parts neatly guide the coil wires around a 90 degree turn, preventing damage and improving the field homogeneity.

[1] https://www.aluxprofile.com/

The parts are made out of machined Delrin, which is expensive. If these parts were to be made again, the author recommends 3D printing them out of ABS or PC, which is likely to be possible in-house, and expected to bring the cost down to approximately five euros per part.

Finally, a one-time lump sum of 200 euros was added to account for bolts, nuts, and other accessories.

Budget item	**Qty.**	**Cost [€]**	**Notes**
MK profile 4040	23.7 m	302.57	Assume 12.79 €/m +5% margin
MK profile 4080	13.2 m	294	Assume 22.40 €/m +5% margin
3-way corner 4040	8	35.12	
L-piece corner 4040	19	14.25	
L-piece corner 4080	12	23.88	
Adjustable feet	12	18	
25x38mm aluminium U-profile	23.4 m	234	
Plastic coil corners	24	120	Assume ABS/PC replacements
Lump sum other structural components		250	e.g. bolts, nuts, spacers, etc.
TOTAL STRUCTURE COST		**1241.82**	

B

Considering alternative milligauss meters

The Demonstrator System (DS) discussed in this thesis uses an Alphalab tri-axial milligauss meter. This device has three shielded analog outputs with which the field measurements can be read out in real-time. It would appear that this device is no longer being sold by Alphalab, and because this device is a key component of the DS, this hurts the reproducibility of the test device. This appendix is written for those who wish to build an identical or similar test setup, but who have the task of finding a similar milligauss meter.

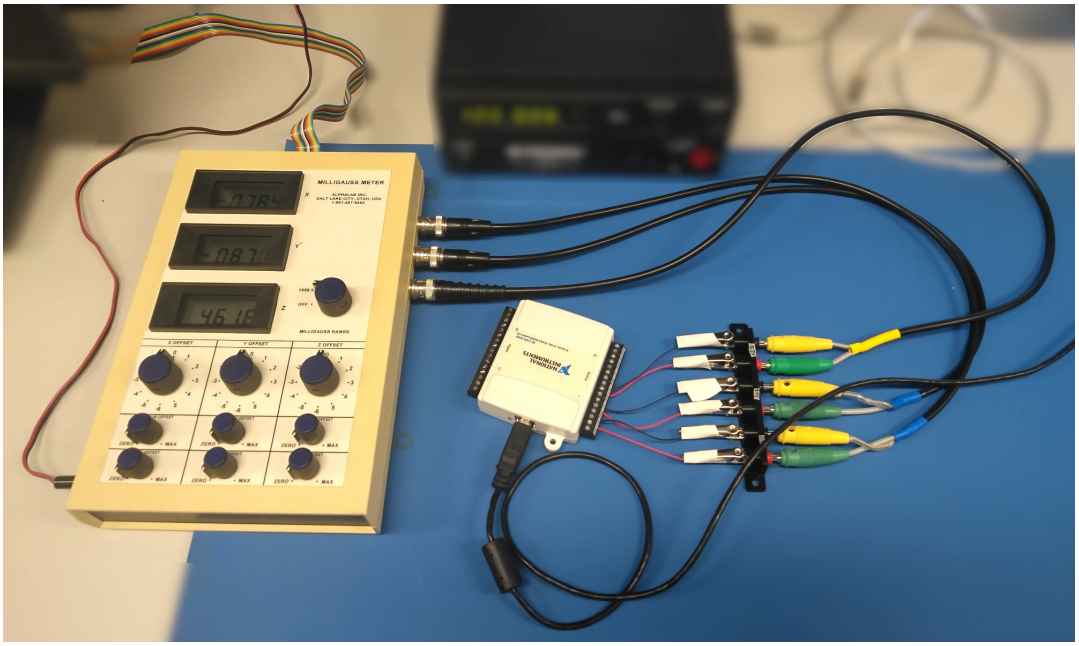


Figure B.1: Reading out the AlphaLab milligauss at 2500 S/s on all three channels by connecting a high-quality analog-to-digital converter (ADC) to its outputs. This is made possible because the outputs are analog, and therefore not restricted by any built-in analog-to-digital converter (ADC).

The specifications of the Alphalab milligauss meter that are critical to the DS have been listed below, in order to find a replacement that is similar:

- Ability to measure three perpendicular magnetic field components (tri-axial sensor)
- Resolution of 0.01 mG (1 nT)

- Range of 2000 mG (200 μT)
- Analog output on shielded (BNC) connectors
- Powered by 12 VDC
- External probe with long (5 meter) cable

Nowadays, many milligauss meters of similar specifications have digital outputs rather than analog inputs. This includes most of the models currently offered by AlphaLab. These devices have an internal analog-to-digital converter (ADC), meaning that the sample rate of magnetic measurements of the testing system is dictated by this ADC. Most are much worse than the AD7124 ADC that is part of the DS. This means that any replacement milligauss must be considered carefully, lest it would hamstring the performance of the testing system as a whole.

Here are a few properties that are desirable in any prospective replacement milligauss meter:

- Analog outputs are highly desirable, as the DS already facilitates their data acquisition (see Figure B.1).
- For digital devices, a sample rate of at least 20 S/s is a must, preferably much more if spectral environmental measurements are planned
- For digital devices, its digital outputs should preferably be serial or some other protocol that can easily be interfaced with a Raspberry Pi
- Care should be taken about how a device could be calibrated, since it generally operates as a reference sensor here

So how to solve this problem? There are a few possible approaches:

1. Find the same milligauss meter second-hand. So long as the device still operates properly this is likely a safe and affordable option, but it is possible that no second hand offers exist. Note that such a device may need to be re-calibrated beforehand.
2. The most similar milligauss meter currently offered by AlphaLab is the MR3[1]. However, this device offers no analog outputs, and is hamstrung by the fact that its internal sample rate is a most regrettable 2 S/s. It is also unclear whether the digital data over USB can be read by the Raspberry Pi. This device is only an option to be considered if the aforementioned caveats are acceptable. There may be a possibility to hack the device and intercept the analog measurement signals before the internal ADC. However, for a device of this price class, this is a risky endeavour, although Alphalab may possibly be willing to perform such a modification commercially.
3. Another option is to find a comparable tri-axial analog milligauss meter. There is not a large selection of devices, but there are some. These are typical bench-top or rack-mounted laboratory devices. Purchasing a brand-new device may be cost-prohibitive, but a second-hand device may be available at an acceptable price point. For example, an older Model 9903 by F.W. Bell would substitute the AlphaLab gaussmeter just fine, so long as 01x probes are used, and these are of sufficient length.
4. Another possibility is to use three single-axis milligauss meters in conjuction to measure the three field components in the Helmholtz cage. The advantage of this is that it broadens the the range of available products, as the market for these devices is much larger than that of tri-axial milligauss meters, and such devices are commonly equal or less than three times the cost. The disadvantage is that three devices have to be interfaced with the Raspberry Pi, the internal ADC may still be a bottle neck, and the mounting of the three sensors requires more attention (see Figure B.2).
5. Finally, it may be possible to construct a milligauss meter using a DIY design, as there are high-quality sensors on the market. It is not so strange a route to take if one considers that a team building a satellite will likely already have the expertise to work with magnetic sensors. However, as this milligauss meter serves as a reference sensor, it would have to be calibrated exceptionally well, which is unlikely to be a trivial task.

[1] https://www.alphalabinc.com/products/mr3/

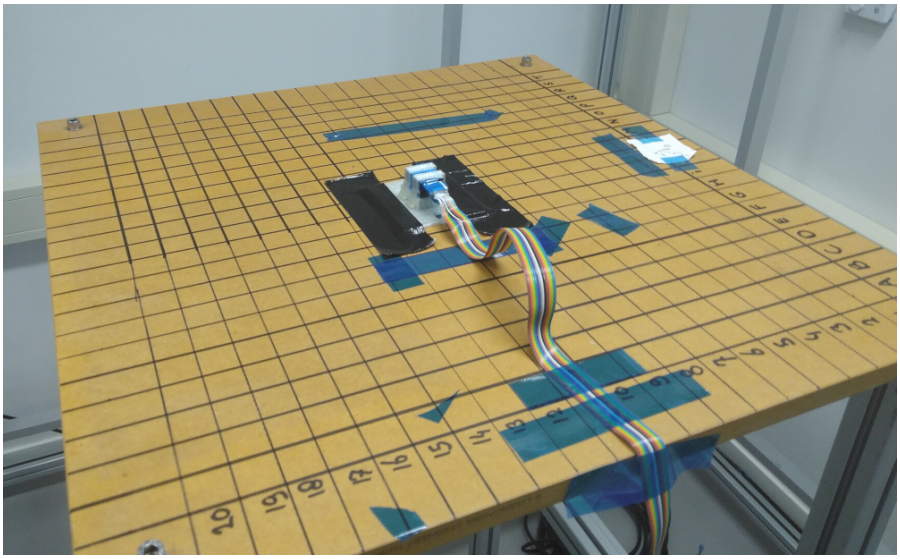


Figure B.2: The cuboid probe of the tri-axial magnetometer can be easily mounted and aligned in a test location. If three different single-axis magnetometers would have been used, the three probes would have had to be positioned with much greater difficulty.

C

Blueprints of Companion Board V1.0

This appendix contains the original schematic of version 1.0 of the Helmholtz cage companion board. The reader must be aware in no uncertain terms that these schematics are **obsolete** and therefore **not for production**. They are only included for academic interest.

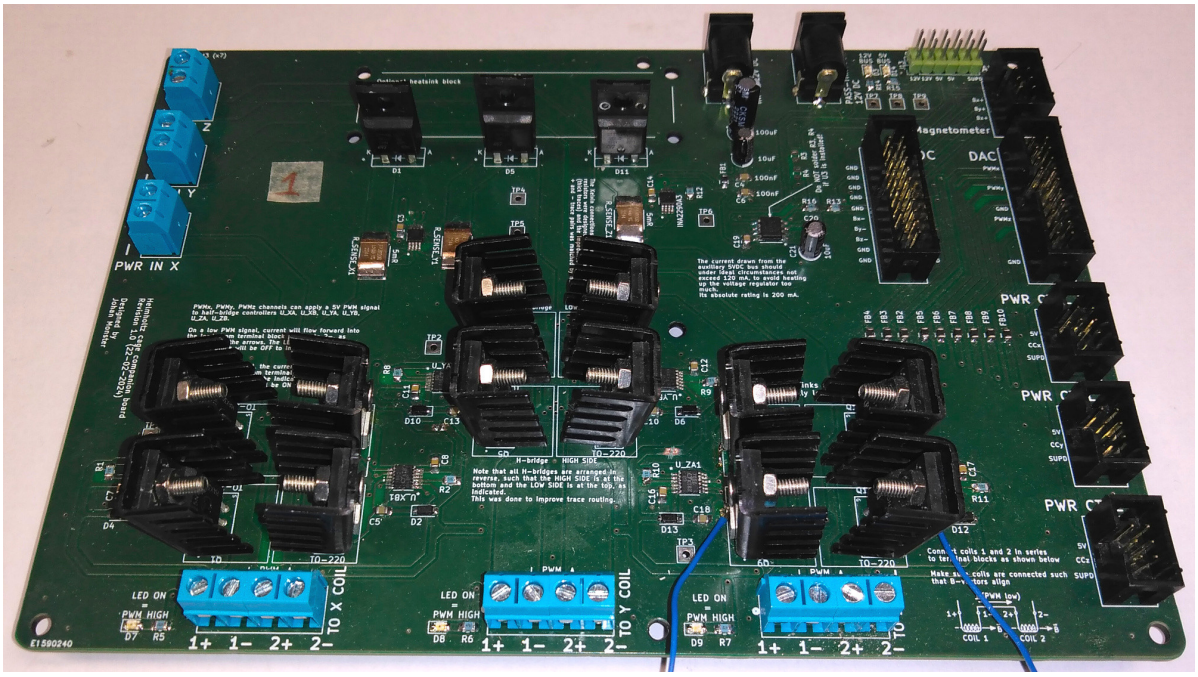
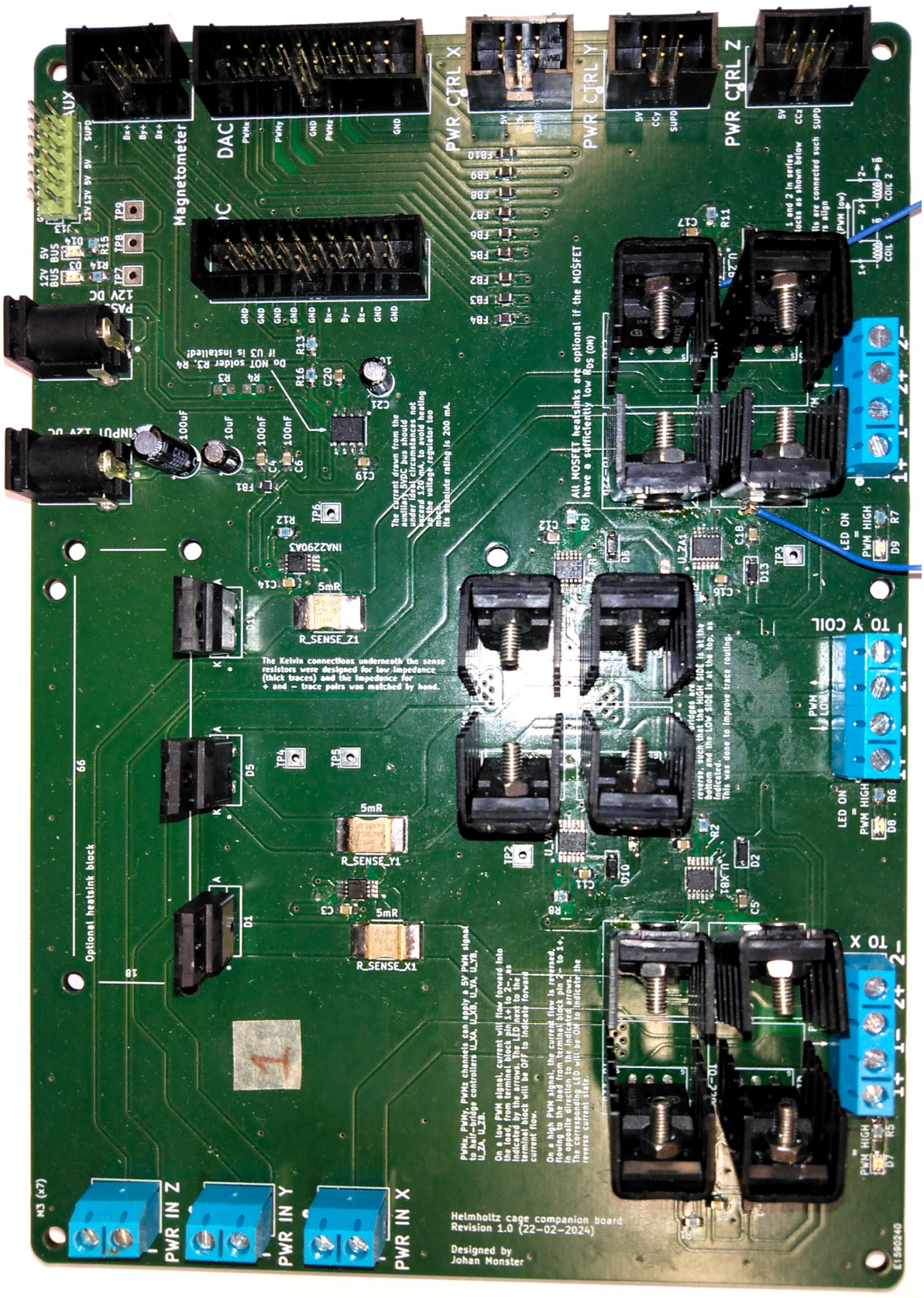
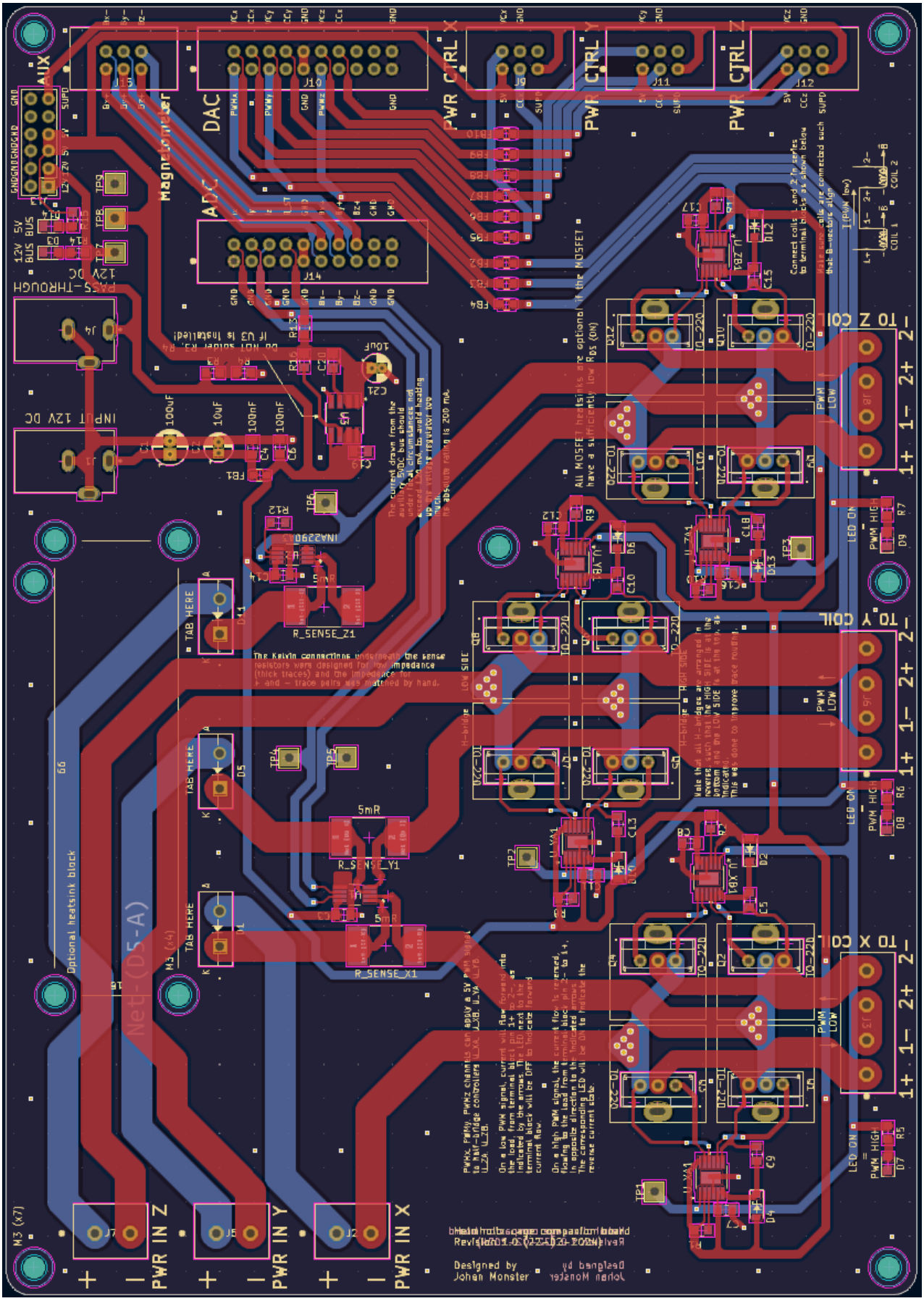
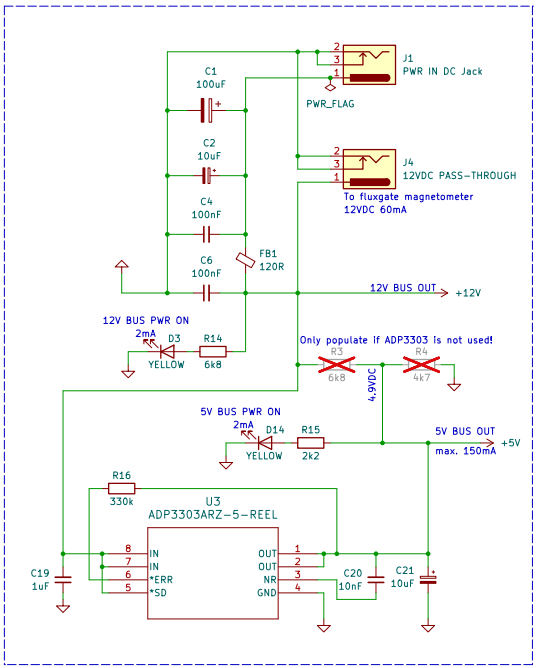


Figure C.1: Companion board v1.0

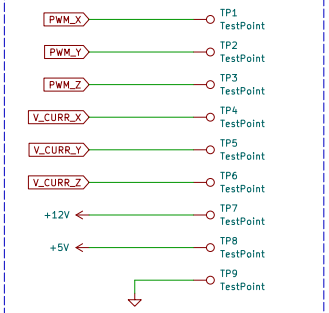

PWR CTRL X
PWR CTRL Y
PWR CTRL Z
DAC
Magnetometer
I2C
AUX
PWR IN X
PWR IN Y
PWR IN Z
TO X COIL
TO Y COIL
LED ON
PWM HIGH
PWM LOW
R_SENSE_X1
R_SENSE_Y1
R_SENSE_Z1
5mR
Optional heatsink block
Helmholtz cage companion board
Revision 1.0 (22-02-2024)
Designed by
Johan Monster


Magnetometer
DAC
ADC
AUX
PWR CTRL X
PWR CTRL Y
PWR CTRL Z
INPUT 12V DC
PASS-THROUGH
R_SENSE_Z1
R_SENSE_Y1
R_SENSE_X1
5mR
INA229
TAB HERE
Optional heatsink block
Net-(D5-A)
TO Z COIL
TO Y COIL
TO X COIL
PWR IN Z
PWR IN Y
PWR IN X
H-bridge
HIGH SIDE
LOW SIDE
Designed by
Johan Monster
Revision 01 (22-10-2024)

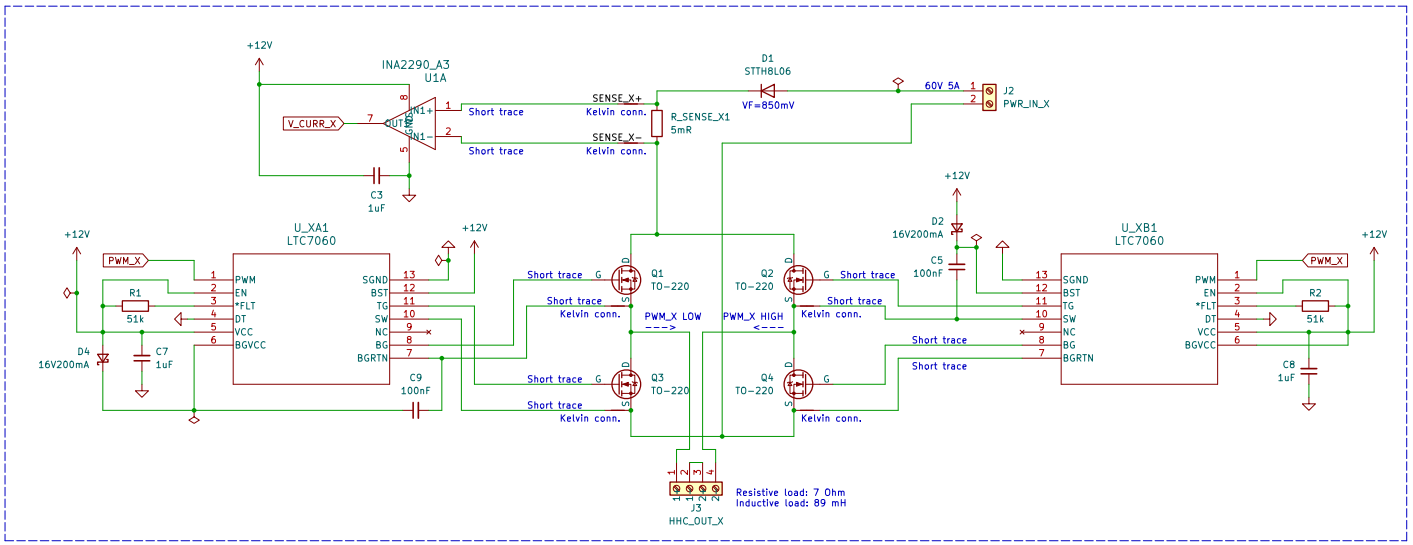
Power buses



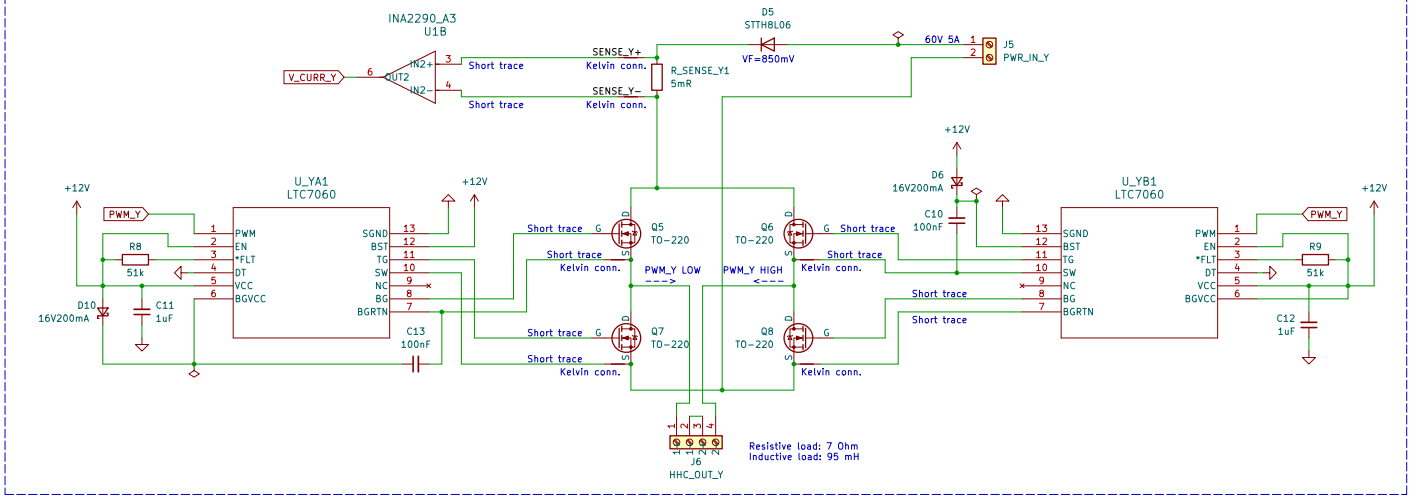
Test points



X-channel



Y-channel



Author: Johan Monster

Sheet: /
File: switching_board.kicad_sch

Title: Helmholtz cage companion board

Size: A3	Date: 2024-02-22	Rev: 1.0
KiCad E.D.A. 8.0.6		Id: 1/2

Board mounting holes

All M3

H1 H2 H3 H4 H5 H6 H7

Heatsink mounting holes

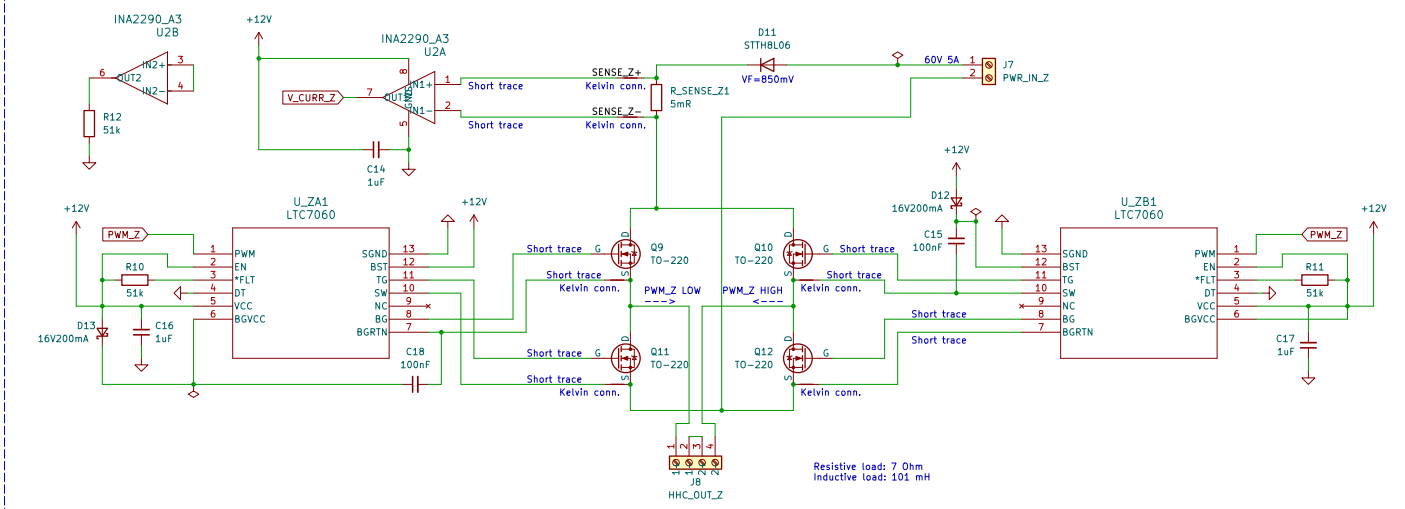
All M3

H8 H9 H10 H11

MOSFET heatsinks

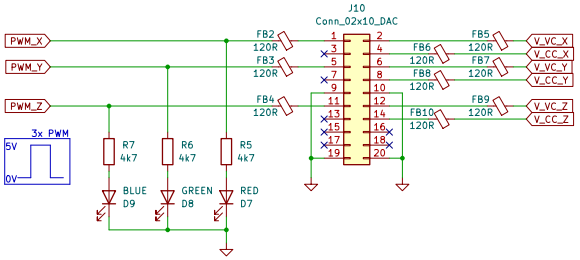
HS1 V7237C heatsink
HS2 V7237C heatsink
HS3 V7237C heatsink
HS4 V7237C heatsink
HS5 V7237C heatsink
HS6 V7237C heatsink
HS7 V7237C heatsink
HS8 V7237C heatsink
HS9 V7237C heatsink
HS10 V7237C heatsink
HS11 V7237C heatsink
HS12 V7237C heatsink

Z-channel

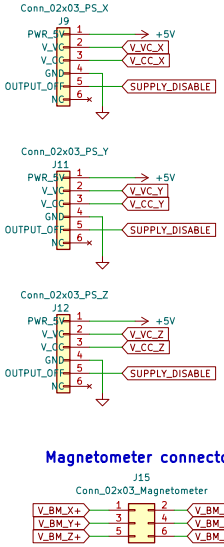


Connectors

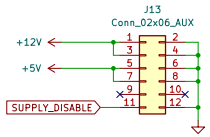
DAC connector



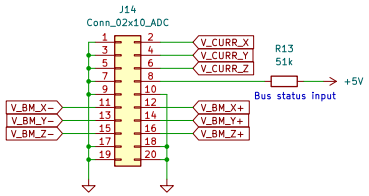
Power supply connectors



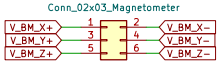
Auxiliary headers



ADC connector



Magnetometer connector



Author: Johan Monster

Sheet: /Untitled Sheet/
File: untitled.kicad_sch

Title: Helmholtz cage companion board

Size: A3	Date: 2024-02-22	Rev: 1.0
KiCad E.D.A. 8.0.6		Id: 2/2

D

Blueprints of Companion Board V2.2

This appendix contains the blueprints of version 2.2.1 of the Helmholtz cage companion board. This appendix contains:

- The schematics of the board
- The trace layout and an RC component value overlay of the board
- Pictures of the manufactured board
- A bill of materials
- Tips and procedures for soldering and assembly
- Explanation of board configuration and supported mods
- A cheat sheet with pin layouts for all devices on the board

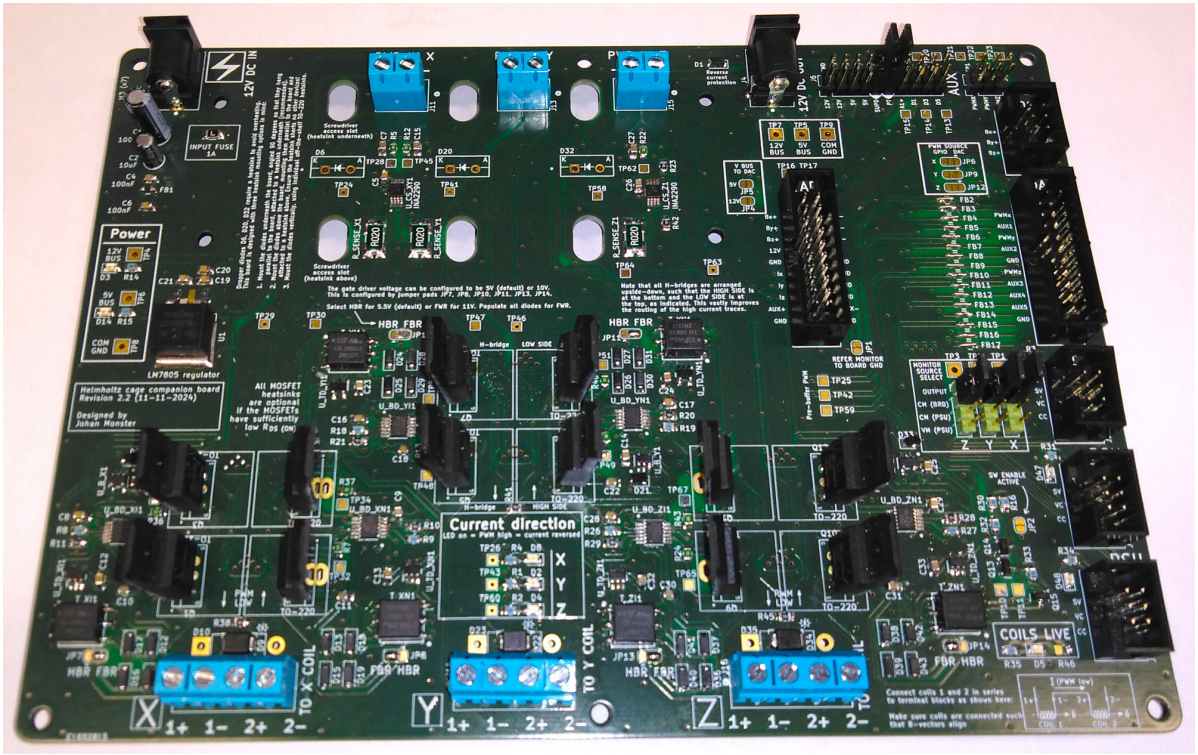
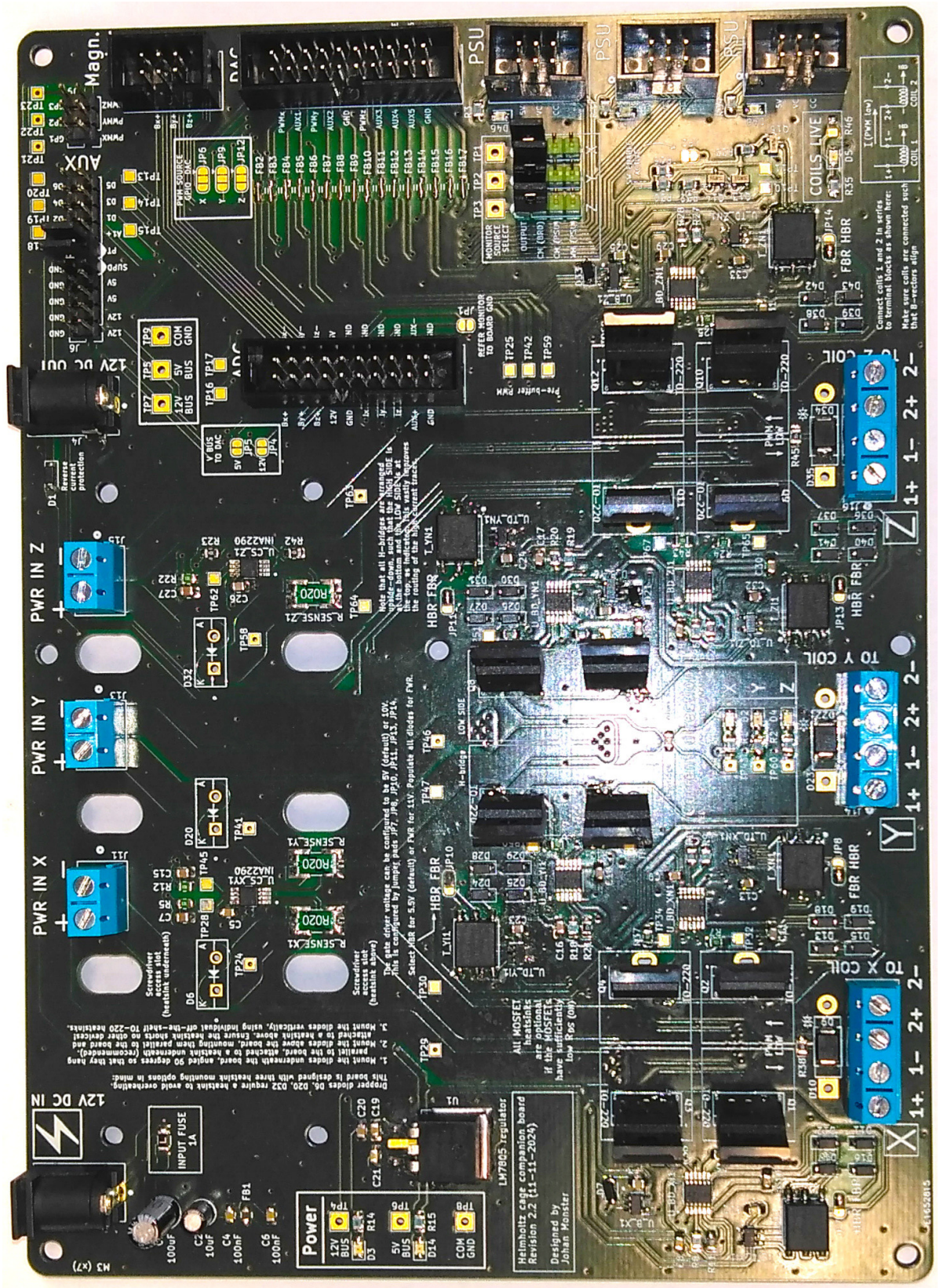
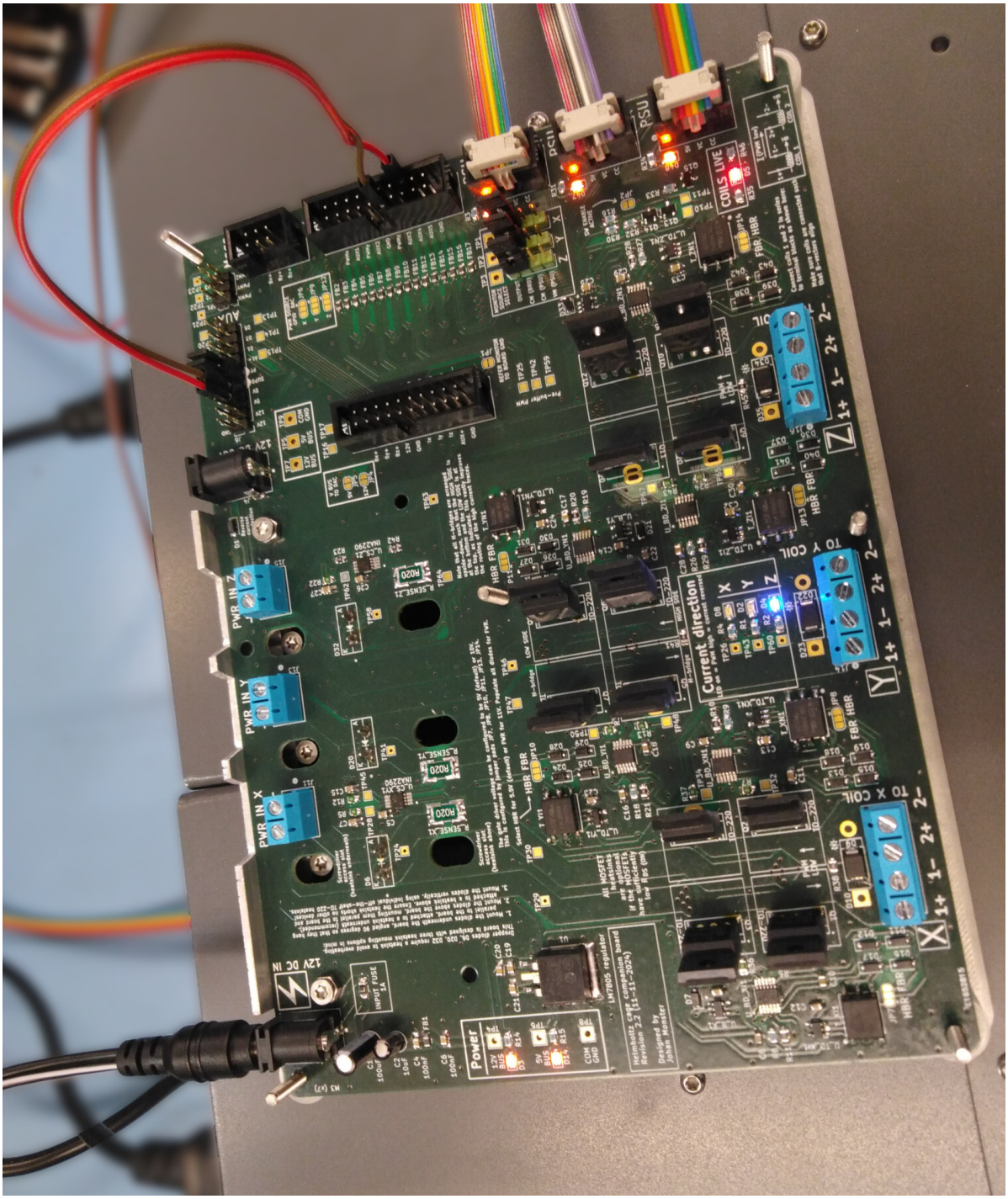
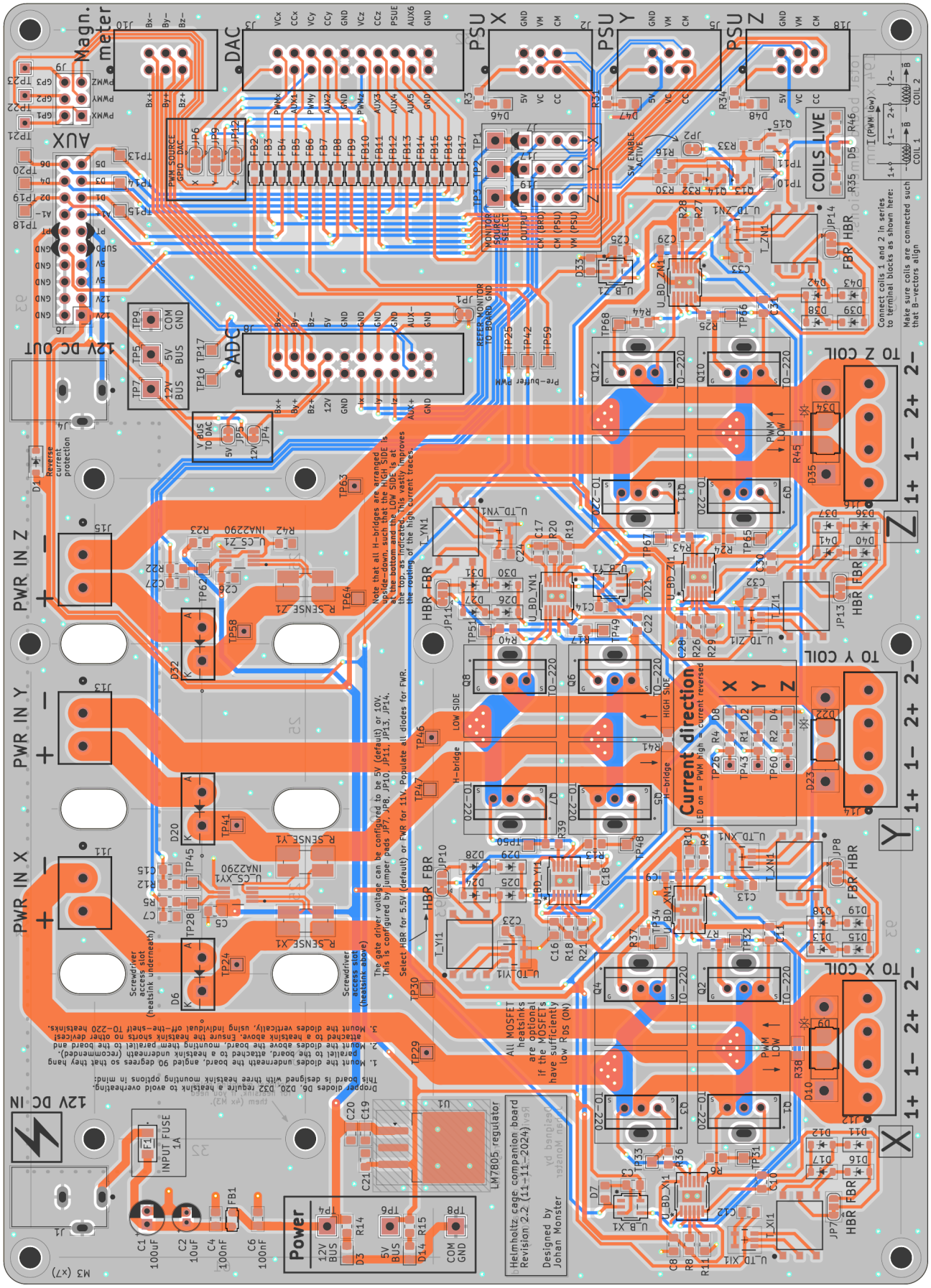


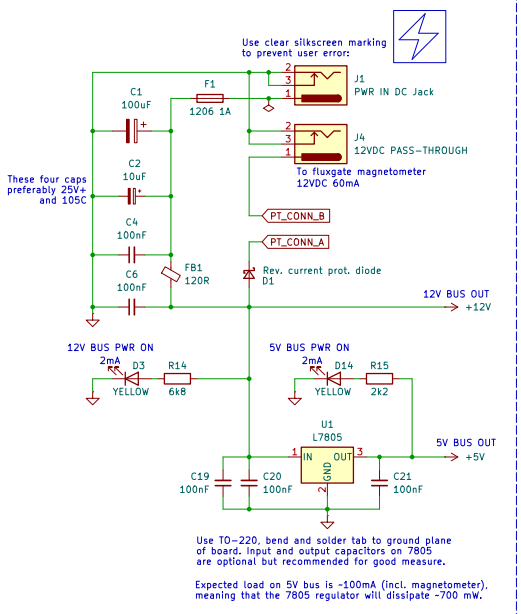
Figure D.1: Companion board v2.2


Helmholtz cage companion board
Revision 2.2 (11-11-2024)
Designed by
Johan Monster
LM7805 regulator
INPUT FUSE 1A
12V DC IN
PWR IN X
PWR IN Y
PWR IN Z
Power
PSU
DAC
AUX
Magn.
TO X COIL
TO Y COIL
TO Z COIL
COILS LIVE
FBR HBR
HBR FBR
Screwdriver access slot (heatsink underneath)
Make sure coils are connected such that B-vectors align
Connect coils 1 and 2 in series to terminal blocks as shown here:
Dropper diodes D6, D20, D32 require a heatsink to avoid overheating. This board is designed with three heatsink mounting options in mind:
1. Mount the diodes underneath the board, angled 90 degrees so that they hang parallel to the board, attached to a heatsink underneath (recommended).
2. Mount the diodes above the board, mounting them parallel to the board and attached to a heatsink above. Ensure the heatsink contacts no other circuitry.
3. Mount the diodes vertically, using individual off-the-shelf TO-220 heatsinks.
The gate driver voltage can be configured to be 5V (default) or 10V. This is configured by jumpers: pins JP7, JP8, JP10, JP11, JP13, JP14.
Select HBR for 5.5V (default) or FBR for 11V. Populate all diodes for PWR.
All MOSFET heatsinks are optional if the MOSFETs have sufficiently low Rds(on)
Note that all H-bridges are arranged spider-down, such that the HIGH SIDE is at the bottom and the LOW SIDE is at the top. This minimizes the routing of the high-current tracks.
Pre-buffer PWM
REFER MONITOR TO BOARD GND
MONITOR SOURCE SELECT
OUTPUT ON (5V0)
PWM SOURCE ZERO-SEL
V BUS TO DAC
Reverse current protection
H-bridge
HIGH SIDE
LOW SIDE
R_SENSE_X1
R_SENSE_Z1
R_SENSE_Y1
R020
INA229
U_LCS_X1
U_LCS_Y1
U_LCS_Z1
U_LTD_XN1
U_LTD_YN1
U_LTD_ZN1
U_BD_XN1
U_BD_YN1
U_BD_ZN1
I_XN1
I_YN1
I_ZN1
TO-220
INA229
100uF
10uF
100nF
100nF
M3 (x7)

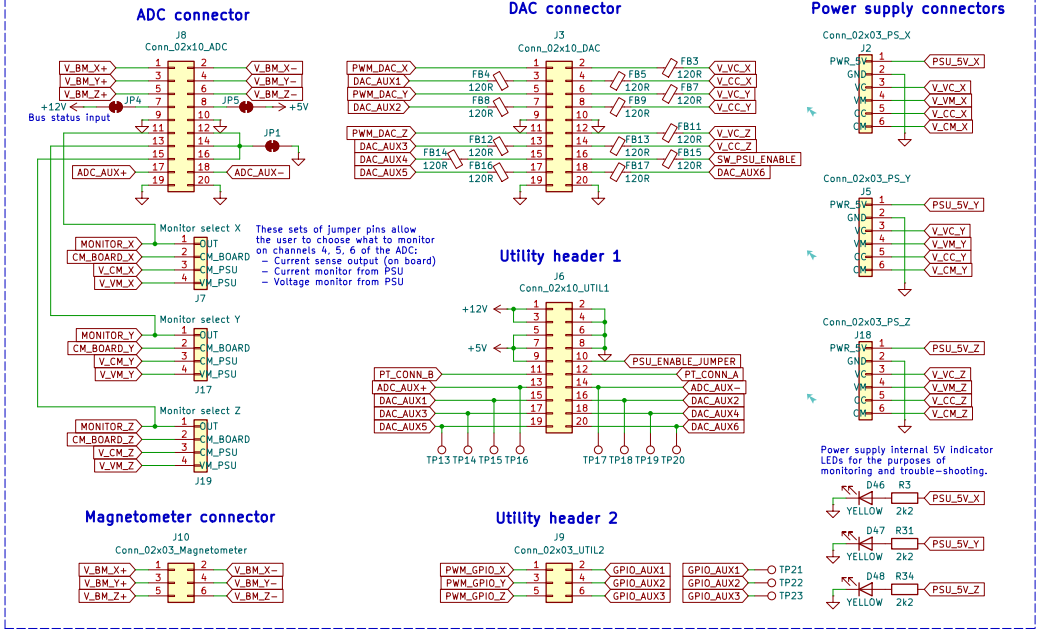

PSU
PCH
COILS LIVE
PWR IN Z
PWR IN Y
PWR IN X
12V DC IN
Current direction
TO Y COIL
TO X COIL
Power


Helmholtz cage companion board
Revision 2.2 (11-11-2024)
Designed by
Johan Monster
Power
12V BUS
5V BUS
COM GND
INPUT FUSE 1A
12V DC IN
12V DC OUT
LM7805 regulator
PWR IN X
PWR IN Y
PWR IN Z
DAC
ADC
PSU X
PSU Y
PSU Z
Magn. meter
AUX
COILS LIVE
TO X COIL
TO Y COIL
TO Z COIL
Current direction
LED on = PWM high = current reversed
H-bridge
HIGH SIDE
LOW SIDE
HBR FBR
Screwdriver access slot (heatsink underneath)
Screwdriver access slot (heatsink above)
All MOSFET heatsinks are optional if the MOSFETs have sufficiently low RDs (on)
Drapper diodes D6, D20, D32 require a heatsink to avoid overheating. This board is designed with three heatsink mounting options in mind:
1. Mount the diodes underneath the board, angled 90 degrees so that they hang parallel to the board, attached to a heatsink underneath (recommended).
2. Mount the diodes above the board, mounting them parallel to the board and attached to a heatsink above. Ensure the heatsink shorts to no other devices!
3. Mount the diodes vertically, using individual off-the-shelf TO-220 heatsinks.
Note that all H-bridges are arranged upside-down, such that the HIGH SIDE is at the bottom and the LOW SIDE is at the top, as indicated. This vastly improves the routing of the high current traces.
The gate driver voltage can be configured to be 5V (default) or 10V. This is configured by jumper pads JP7, JP8, JP10, JP11, JP13, JP14.
Select HBR for 5.5V (default) or FWR for 11V. Populate all diodes for FWR.
Connect coils 1 and 2 in series to terminal blocks as shown here:
Make sure coils are connected such that B-vectors align

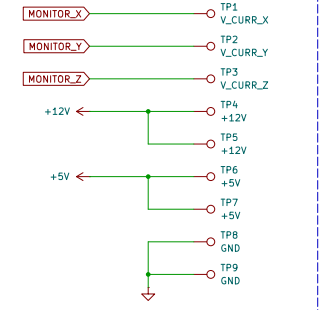
Power buses



Connectors



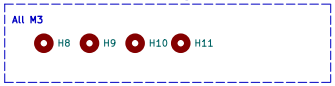
Test points



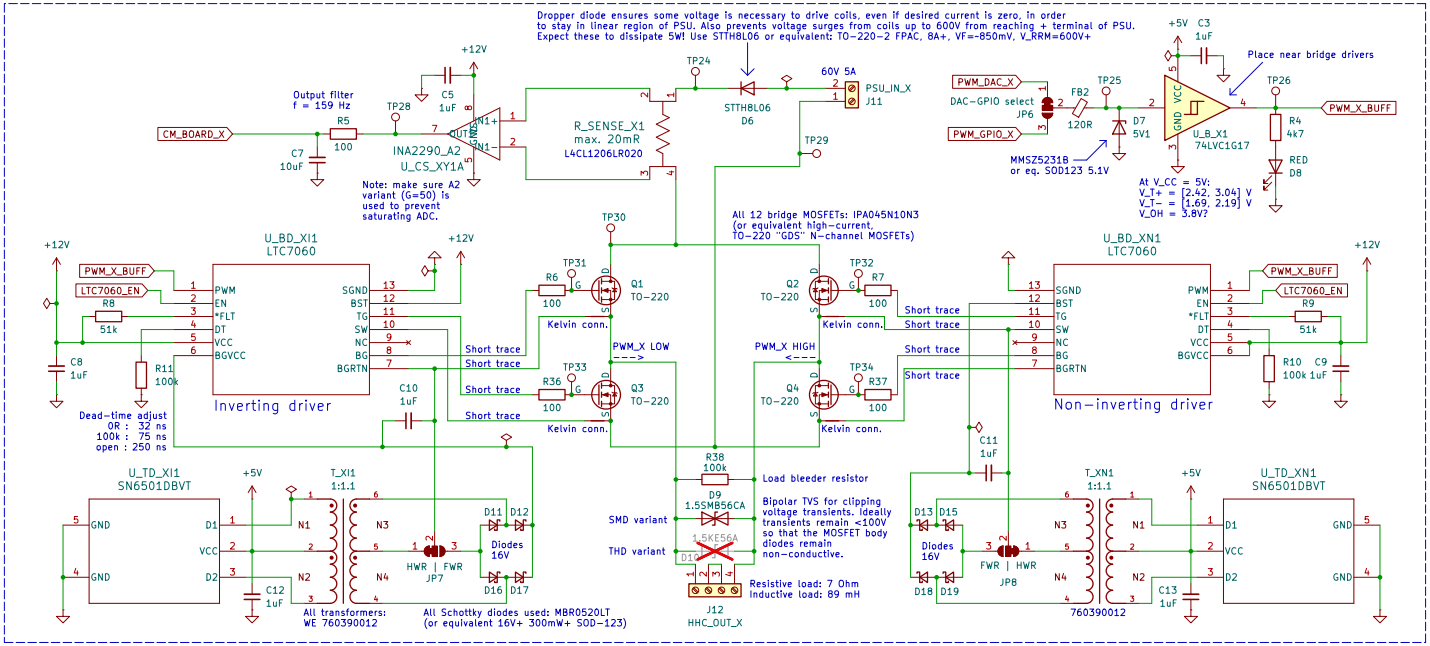
Board mounting holes



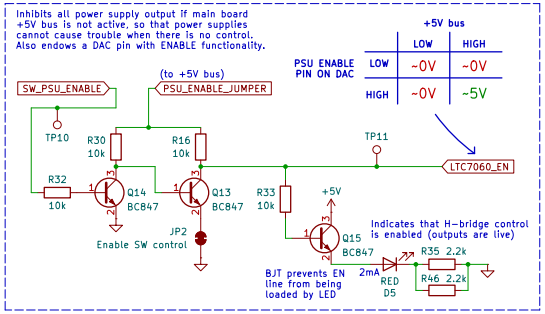
Heatsink mounting holes



X-channel



Power supply inhibition



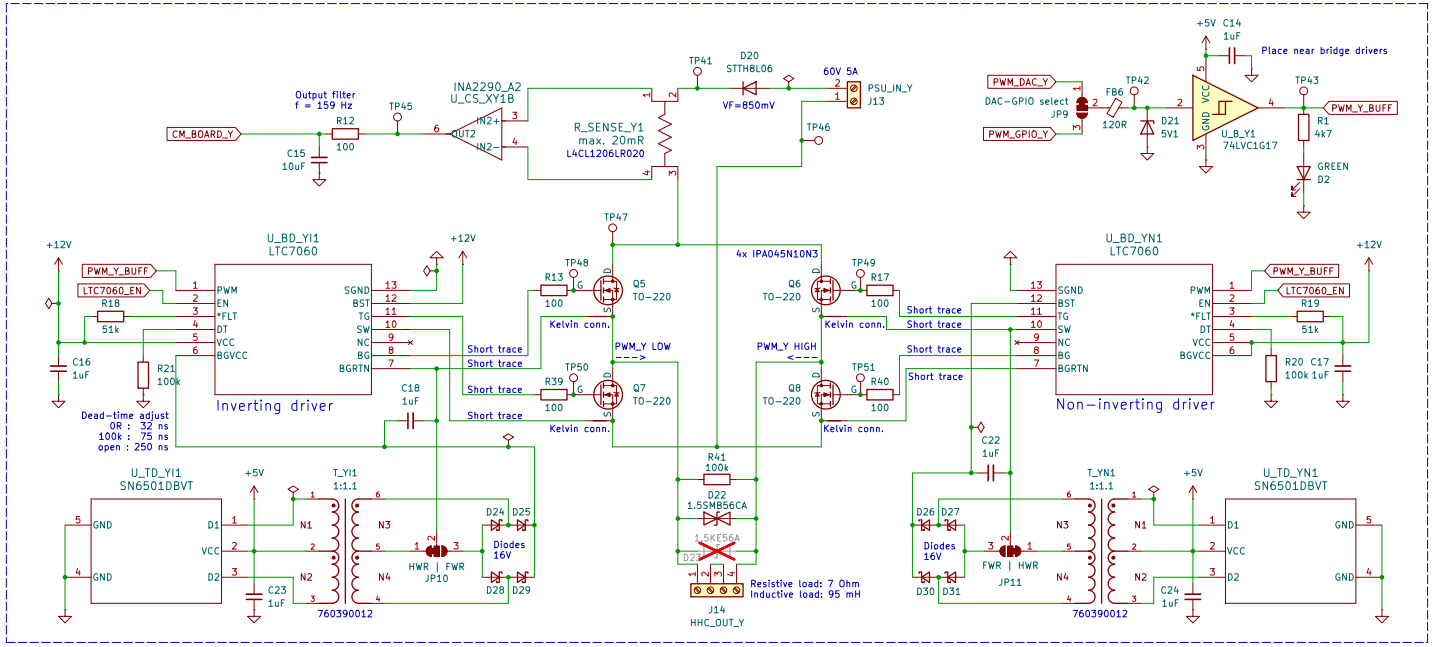
Author: Johan Monster

Sheet: /
File: switching_board.kicad_sch

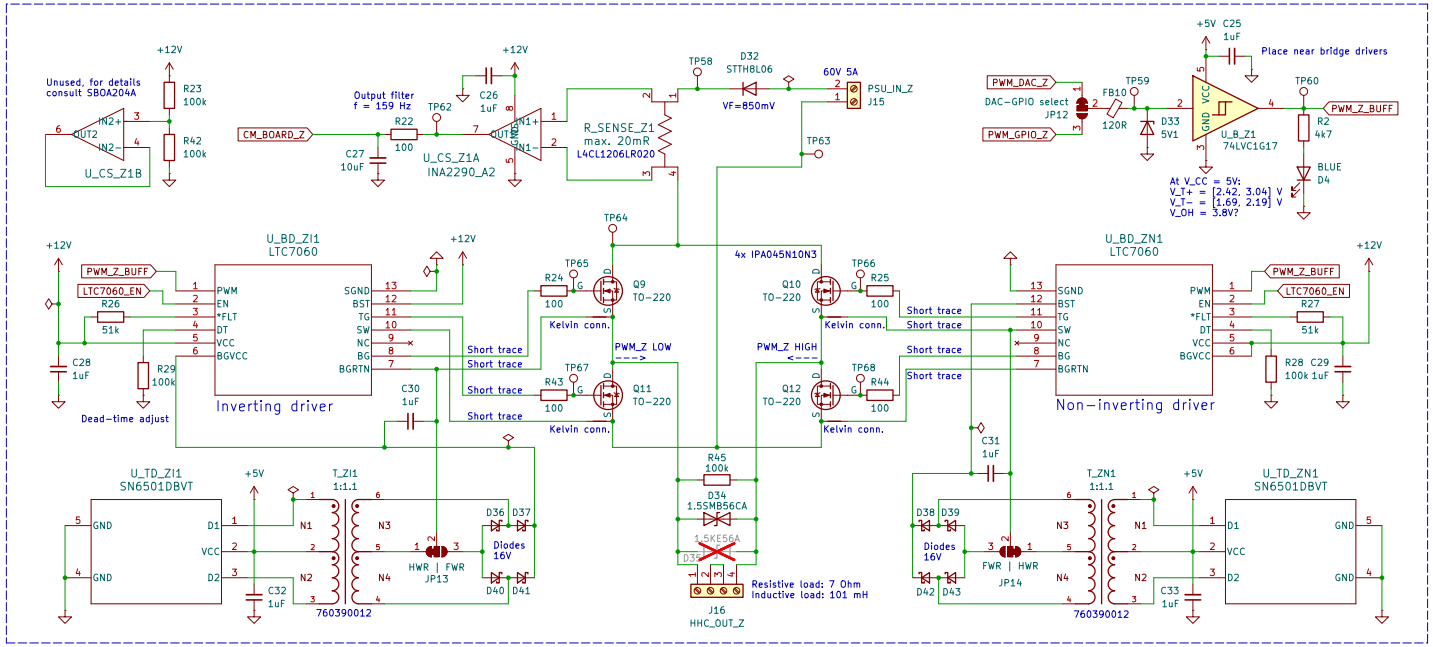
Title: Helmholtz cage companion board

Size: A3	Date: 2024-11-11	Rev: 2.2
KiCad E.D.A. 8.0.6		Id: 1/2

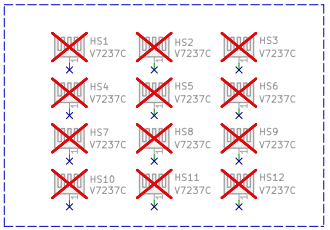
Y-channel



Z-channel



MOSFET heatsinks



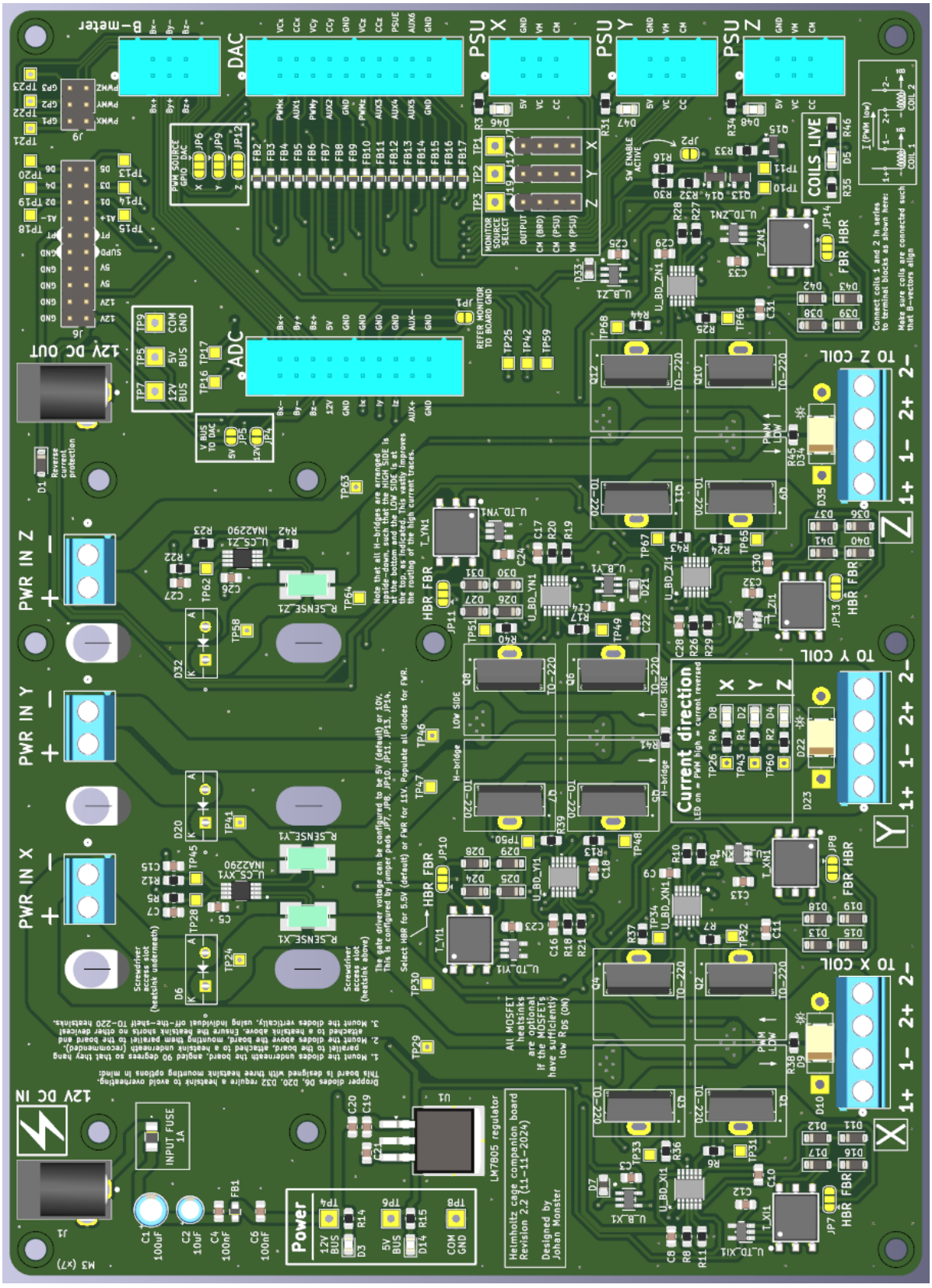
Consider using these heatsinks if using bridge MOSFETs that do not have ultra-low R_DS_ON (a few mOhm).
If using the indicated IPA045 MOSFETs, R_DS_ON is 4.5mOhm and the heat sinks are not necessary.

Author: Johan Monster

Sheet: /Z-channel/
File: zchannel.kicad_sch

Title: Helmholtz cage companion board

Size: A3	Date: 2024-11-11	Rev: 2.2
KiCad E.D.A. 8.0.6		Id: 2/2


B-meter
DAC
ADC
PSU X
PSU Y
PSU Z
12V DC OUT
12V DC IN
PWR IN X
PWR IN Y
PWR IN Z
TO X COIL
TO Y COIL
TO Z COIL
COILS LIVE
HBR FBR
Current direction
Power
LM7805 regulator
Heimholtz cage companion board
Revision 2.2 (11-11-2024)
Designed by
Johan Monster

D.1. Tips for soldering and assembly

During the thesis, a total of five boards of V1.0 were assembled, and a total of two V2.2 boards were assembled. This has given the author some practical experience with the assembly of these boards, on top of the considerations already implicitly included into the design. The choice was made to hand-solder the boards completely to provide the author of the thesis with surface mounted device (SMD) soldering experience. Reflow soldering with a masking stencil would also have been possible, and conveniently all SMD components are on one side of the board. However, a number of through-hole components will likely still require some hand-soldering.

This section is for people who have ordered this board and are about to assemble it, or are considering whether to do so.

Because the board had few size constraints, an SMD component size of 0805 or greater was chosen for all passive devices where possible. Semiconductor packages where chosen with sizes and pin pitches of roughly the same scale. The only case where this was not possible was with the INA2290 current sense amplifiers and the LTC7060 half-bridge driver chips, which were slightly smaller than desired, but were not available in more convenient packages.

For hand-soldering, the following tools and facilities are needed:

1. Soldering microscope (recommended) or other good magnification
2. Well-illuminated workplace
3. Anti-static work equipment
4. Solder fume extractor
5. SMD soldering iron with medium-sized SMD tip
6. Precision hot air gun
7. Heating plate (optional but recommended)
8. One or more fine, preferably anti-static soldering tweezers
9. Desolder wick
10. Flux-core solder (0.5mm or finer, leaded recommended)
11. SMD soldering paste
12. Syringed flux paste
13. Flux cleaner compound
14. Isopropyl alcohol or similar cleaning agent
15. Cleaning brushes
16. Disposable paper towels
17. Knife or scalpel
18. Empty printed circuit board (PCB)s
19. Bill of Materials
20. All components on the Bill of Materials
21. Board mounting solution and helping hands
22. Multimeter with continuity function and resistance measurements

Before we get to soldering, here are a few considerations beforehand:

- The value of a good work environment, a reliable board clamp, and quality tools cannot be understated.
- The value of good component management and book-keeping cannot be understated. The board assembly instructions in section D.2 are written in such a way that each step and each individual component can be systematically crossed off. The component overlay of Figure D.2 provides an additional way to find and cross off components.

- When soldering, we generally start with the most challenging components and move on to larger components later. Solder SMD components before through-hole components, and solder large objects such as terminal blocks after that.
- The component pinout cheat sheet at the end of this appendix can be useful to quickly sanity check your component placement at a glance.
- Troubleshooting and fixing the board is easier the fewer components are on them. Before you start, inspect the empty board for defects, and repair if needed. If it is beyond repair, soldering it is a waste of time and components. At various stages of soldering, especially before introducing larger components such as header blocks and heatsinks, perform multimeter tests and visual inspections regularly to make sure components are soldered on correctly and nothing was bridged by accident. Fixing such problems are greatly aided if there is still plenty of room for access around the components.
- If the room smells like chicken, you may be holding the soldering iron incorrectly.

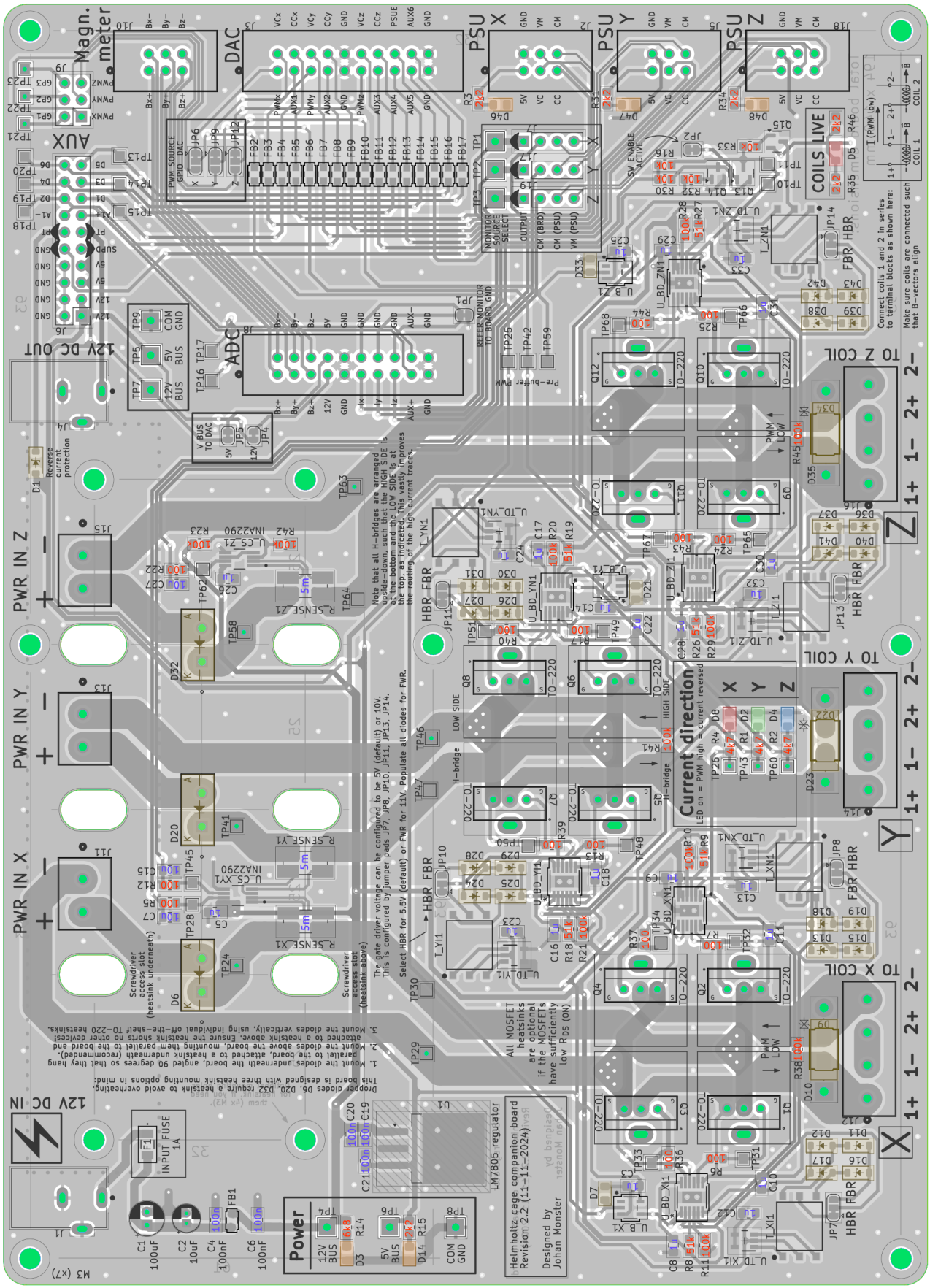


Figure D.2: Component overlay for the V2.2 companion board, useful during soldering.

D.2. Board assembly instructions

Full board assembly instructions:

☐	#	Instruction	Value	Components
☐	1	Position board and prepare work environment. Check board for damage. Clean soldering pads with isopropyl alcohol. Mark the board to distinguish it from other boards (e.g. with a sticker).	-	-
☐	2	Solder thermal pads of LTC7060 half bridge drivers to pads on the board (x6). These are 12 pin MSOP chips with a ground pad. The ground pad is both an electrical connection and a thermal pathway. They can be soldered either by applying solder paste to the top and heating this with a heat gun, or by wicking solder from underneath through the vias. Ensure that the pins align with the pads on the board, bearing in mind that the surface tension of the molten solder underneath the chip may move it around a bit. Experience teaches that these chips are fairly fragile and break easily from heavy-handed soldering. **CHECK ALIGNMENT OF ICs BY MATCHING DOT POSITIONS!**	-	`U_BD_XI, U_BD_XN, U_BD_YI, U_BD_YN, U_BD_ZI, U_BD_ZN`
☐	3	Solder other 12 pins of each LTC7060 half bridge controller to the board (x6). Check that no shorts between the heat pad and any pin exists by checking for continuity between each pin and the ground plane. **Only** pin 4 should have contact the ground plane (`TP8`).	-	`U_BD_XI, U_BD_XN, U_BD_YI, U_BD_YN, U_BD_ZI, U_BD_ZN`
☐	4	Solder the SN6501 transformer driver chips to the board (x6). These are SOT-23-5 chips; **be careful not to confuse these** transformer drivers (`U_TD_**`) with the buffer chips (`U_B_**`).	-	`U_TD_XI, U_TD_XN, U_TD_YI, U_TD_YN, U_TD_ZI, U_TD_ZN`
☐	5	Solder INA229A3 chips to the board (x2). These are 8-pin VSSOP packages. **CHECK ALIGNMENT OF ICs BY MATCHING DOT POSITIONS!** Check for continuity to ground plane (`TP8`). Only pin 5 should be connected to ground.	-	`U_CS_XY, U_CS_Z`
☐	6	Solder the SN74LVC1G17 Schmitt-trigger buffer chips to the board (x3). These are SOT-23-5 chips; **be careful not to confuse these** buffer chips (`U_B_**`) with the transformer drivers (`U_TD_**`).	-	`U_B_X, U_B_Y, U_B_Z`
☐	7	Solder the BC847 NPN transistors onto the board (x3). They are in SOT-23 packages.	-	`Q13, Q14, Q15`
☐	8	Solder the LM7805 voltage regulator to the board (x1). This is a TO-263 package with a large ground plane. The ground pad is both a ground connection and a necessary thermal pathway, so it must be soldered well. If it is difficult to solder the ground pad, pre-apply some solder to the pad, use plenty of flux, use a large soldering tip, and try to wick solder underneath the chip via the sides or from underneath the board, through the vias. The chip is fairly robust and can handle being heated for a while.	-	`U1`
☐	9	*This concludes most of the SMD ICs. Now is a good time to check the solder work:* • *Test each soldered pin, between pad and top of the pin, near the chip package, to ensure solder connection is made.* • *Test each soldered pin for possible solder bridging by testing the pin with its nearby neighbours, or by carefully "raking" the multimeter probes across the pins and listening.* • *Test that the board power supply lines are not shorted to ground* • *Many pins have a test point on the board, test these too. For example, the three line buffers (`U_B_*`) for X, Y, Z should have their inputs connected to `TP25`, `TP42`, `TP59` respectively.* *Once confident that the solder work is good, we can move on.*		

☐	10	Solder the ferrite beads onto the board (x17). FB1 is on the left of the board, amidst the filter capacitors of the power section. The other ferrite beads are on the right of the board in a single column. The ferrite beads are not essential and can in a pinch be replaced with zero ohms or links, but they cannot remain unsoldered.	120R	FB1, FB2, FB3, FB4, FB5, FB6, FB7, FB8, FB9, FB10, FB11, FB12, FB13, FB14, FB15, FB16, FB17
☐	11	Solder 1uF bypass capacitors for the INA2290A3 chips (x2).	1u	C5, C26
☐	12	Solder 1uF bypass capacitors for the SN74LVC1G17 buffer chips (x3).	1u	C3, C14, C25
☐	13	Solder 1uF bypass capacitors for the SN6501 transformer driver chips (x6).	1u	C12, C11, C23, C24, C32, C33
☐	14	Solder 1uF bypass capacitors for the LTC7060 half-bridge driver chips (x6).	1u	C8, C9, C16, C17, C28, C29
☐	15	Solder 1uF rectifier smoothing capacitors (x6).	1u	C10, C11, C18, C22, C30, C31
☐	16	Solder 10uF current sensing filter capacitors (x3).	10u	C7, C15, C27
☐	17	Solder 100 nF power smoothing capacitors (x2).	100n	C4, C6
☐	18	Solder 100 nF regulator stabilization capacitors (x3).	100n	C19, C20, C21
☐	19	*If done correctly, a total of 31 SMD capacitors were soldered.*		
☐	20	Solder 100 ohm current sensing filter resistors (x3).	100	R5, R12, R22
☐	21	Solder 100 ohm MOSFET gate stopper resistors (x12).	100	R6, R36, R7, R37, R13, R39, R17, R40, R24, R43, R25, R44
☐	22	Solder 2.2k ohm current limiter resistor for 5V bus indicator (x1).	2k2	R15
☐	23	Solder 2.2k ohm current limiter resistors for 5V PSU ref indicators (x3).	2k2	R3, R31, R34
☐	24	Solder 2.2k ohm current limiter resistors for "COILS LIVE" indicator (x2). Solder both for a brighter LED, solder one of either for a dimmer LED.	2k2	R35, R46
☐	25	Solder 4.7k ohm current limiter resistor for bridge status LEDs (x3).	4k7	R1, R2, R4
☐	26	Solder 6.8k ohm current limiter resistor for 12V bus indicator (x1).	6k8	R14
☐	27	Solder 10k ohm resistor-transistor logic (RTL) resistors (x4).	10k	R16, R30, R32, R33
☐	28	Solder 51k pull-down resistors to the half-bridge drivers (x6).	51k	R8, R9, R18, R19, R26, R27
☐	29	Solder 100k dead-time adjust resistors for half-bridge drivers (x6).	100k	R11, R10, R21, R20, R29, R28
☐	30	Solder 100k tie-down resistors for empty INA2290A3 channel (x2).	100k	R23, R42
☐	31	Solder 100k coil bleeder resistors (x3).	100k	R38, R41, R45
☐	32	If done correctly, a total of 45/46 SMD resistors were soldered, depending on whether R35 *and* R46 *were soldered as a pair or not.*		
☐	33	Solder the current shunt resistors to their pads. If using the ones from the bill of materials (BOM), they can be placed in either direction, so long as the pads point down.	5m	R_SENSE_X, R_SENSE_Y, R_SENSE_Z
☐	34	Solder reverse current protection Schottky diode (x1). **ENSURE CORRECT DIODE ORIENTATION!**	-	D1
☐	35	Solder rectifier Schottky diodes (x24). **ENSURE CORRECT DIODE ORIENTATION!**	-	D11, D12, D16, D17, D13, D15, D18, D19, D24, D25, D28, D29, D26, D27, D30, D31, D36, D37, D40, D41, D38, D39, D42, D43
☐	36	Solder buffer over-voltage protection zener diodes (x3). **ENSURE CORRECT DIODE ORIENTATION!**	5V1	D7, D21, D33
☐	37	Solder amber indicator LEDs for 12V and 5V power bus (x2). **ENSURE CORRECT DIODE ORIENTATION!**	AMBER	D3, D14
☐	38	Solder amber indicator LEDs for power supply references. (x3). **ENSURE CORRECT DIODE ORIENTATION!**	AMBER	D46, D47, D48
☐	39	Solder red indicator LED for COILS LIVE indicator. (x1). **ENSURE CORRECT DIODE ORIENTATION!**	RED	D5

☐	40	Solder red indicator LED for X coil current direction. (x1). **ENSURE CORRECT DIODE ORIENTATION!**	`RED`	`D8`
☐	41	Solder green indicator LED for Y coil current direction. (x1). **ENSURE CORRECT DIODE ORIENTATION!**	`GREEN`	`D2`
☐	42	Solder blue indicator LED for Z coil current direction. (x1). **ENSURE CORRECT DIODE ORIENTATION!**	`BLUE`	`D4`
☐	43	Check the continuity of the 1206 input fuse. Solder it to the board (x1). Check the continuity again after it cools.	`1A`	`F1`
☐	44	*We are about to start soldering larger components, so now is a good time to perform any tests on the previous solder work, as clearance to fix issues will be reduced when larger components are soldered onto the board.*		
☐	45	Place the three 1x4 terminal blocks for the coil outputs to the board, and tape them in place. **Do not solder them yet.** We are placing them so that we can assess whether they are too close to the soldering locations for the bi-directional TVS diodes. After soldering the TVS diodes, you may remove the terminal blocks again and put them aside for now, so we can solder them when the board is mounted upside down.	-	`J12, J14, J16`
☐	46	Ensuring enough clearance with the terminal blocks nearby, position and solder the three bi-directional transient voltage suppression (TVS) diodes to the board. These are bi-directional devices so their orientation does not matter. If you are using SMD TVS devices, these are `D9`, `D22`, `D34`. If instead, you are using through-hole TVS devices, use `D10`, `D23`, `D35`, and their corresponding soldering connection holes; In the latter case the SMD pads should not be soldered.	`56V`	`D9, D22, D34`
☐	47	Solder the 100 uF electrolytic capacitor for the DC input power ripple filter	`100u`	`C1`
☐	48	Solder the 10 uF electrolytic capacitor for the DC input power ripple filter	`10u`	`C2`
☐	49	Solder the SMD transformers onto the board (x6). Orientation matters so ensure you align pin 1 of the package with the dot on the silkscreen.	-	`T_XN, T_XI, T_YN, T_YI, T_ZN, T_ZI`
☐	50	If you are using individual heat sinks for the H-bridge MOSFETs, connect MOSFETs to their heat sink and ensure they are flat against one another (x12). Heatsinks are optional if you are using very low on-resistance MOSFETs, such as the ones in the BOM.	-	-
☐	51	Solder the H-bridge MOSFETs in place (x12). MOSFETs gate pins are sensitive to electrostatic discharge (ESD), so take proper precaution. If you are using individual MOSFET heatsinks, solder the heatsinks to their anchor point onto the board. Note that these anchor points are **NOT GROUNDED** to the ground plane, meaning that even if you use different MOSFETs that are not in insulated TO-220 packages, they should not short-circuit via the individual heatsinks.	-	`Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11, Q12`
☐	52	*At this point, the microscope is no longer very necessary, so move the board into a position where it can be easily flipped over, or worked on from both sides.*		
☐	53	Solder the coil output terminal blocks to the board (x3). Ensure they are flush with the board and do not touch the TVS diodes. The terminals should face outwards from the board.	`1x4`	`J12, J14, J16`
☐	54	Solder the power supply input terminal blocks to the board (x3). The terminals should face outwards from the board. Ensure the block is flush with the board before soldering.	`1x2`	`J11, J13, J15`
☐	55	Solder the barrel jack connectors for the 12V DC IN and the 12V DC OUT (x2). Ensure the connectors are flush with the board before soldering.	-	`J1, J4`
☐	56	Solder the primary AUX header pin block to the board (x1). Ensure the block is flush with the board before soldering.	`2x10`	`J6`
☐	57	Solder the secondary AUX header pin block to the board (x1). Ensure the block is flush with the board before soldering.	`2x3`	`J9`

☐	58	Solder the monitor source select header pins to the board (x3). Ensure the block is flush with the board before soldering.	`1x4`	`J7, J17, J19`
☐	59	Solder the Magnetometer input Insulation Displacement Connector (IDC) header pins to the board (x1). Ensure the block is flush with the board before soldering.	`2x3`	`J10`
☐	60	Solder the IDC header pin block for the ADC and the digital-to-analog converter (DAC) to the board (x2). Ensure the blocks are flush with the board before soldering. Orient the blocks using the dot on the silk screen. If you solder these the wrong way, they are difficult to desolder, and you will have to make an inverted IDC cable, which will work but will be confusing to work with.	`2x10`	`J3, J8`
☐	61	Solder the IDC header pin block for the PSU connectors to the board (x3). Ensure the blocks are flush with the board before soldering. Orient the blocks using the dot on the silk screen. If you solder these the wrong way, they are difficult to desolder, and you will have to make an inverted IDC cable, which will work but will be confusing to work with.	`2x3`	`J2, J5, J18`
☐	62	Connect the dropper diodes in whatever configuration is most convenient. Heat-sinking these is highly recommended. Before soldering them into place, check clearance with surrounding components and to make sure they can interface well with the heatsink. If using a remote heat sink, use cables rated for 8 A or higher. Since these are diodes: **ENSURE CORRECT DIODE ORIENTATION!**	-	`D6, D20, D32`
☐	63	*The board should be ready at this point. Check whether you soldered everything by counting components and checking the board for empty spaces. Perform the necessary tests with the multimeter to ensure that the board is soldered correctly and there are no unintentional short-circuits or open connections.*		
☐	64	Clean the whole board with solder cleaner, isopropyl alcohol, and whatever else is necessary. Paper towel is fine to use, but lint-free towels and brushes give cleaner end results. Note that precautions regarding ESD still apply.		
☐	65	If desired, configure the board using the jumper pads. For a detailed overview, consult Table D.2. All but the first two are optional; if you do not touch any jumper pads, the board will be ready to use in its default configuration. You can change the setting on a jumper pad by cutting the trace in between the connected pads, and/or bridging the unconnected pads with solder. For an example, see Figure D.3.	-	-
☐	66	Besides the jumper pads, there are a few pin jumpers that should be used to correctly set up the board. Most notably, the board inhibits the H-bridges with a hardware switch and a software switch. To enable the hardware switch, bridge pins `J6_9` (`SUPD`) and `J6_10` on the AUX header. Also, if the 12V pass-through barrel connector should be active, pins `J6_11` and `J6_12` on the AUX header should be jumped. Also, the current monitoring source needs to be selected using jumpers.		

D.3. Board configuration and supported mods

Table D.2: Configuration options and supported mods for the V2.2 companion board.

Pad/pins	Location on board	Function
J7, J17, J19	Right side of the board, near PSU X header block.	**Default: NONE, USER MUST SET AT LEAST ONE** These jumper pins set the current/voltage monitor source that is sent to the Ix, Iy, Iz pins of the ADC. - CM (BRD) + OUTPUT: Uses current measurement from current sensing setup on board (using shunt resistors) - CM (PSU) + OUTPUT: Uses current measurement from your constant-current power supplies. Use if this is more accurate than on-board solution. - VM (PSU) + OUTPUT: Uses voltage measurement from your constant-current power supply. Use if you value this more than current measurements. - OUTPUT DISCONNECTED: No signal sent to ADC. Possible bonus configuration for debugging / sensing comparison: Multiple measurements of a single channel: Connect e.g: - CM (BRD) X → OUTPUT X (ADC Ix) - CM (PSU) X → OUTPUT Y (ADC Iy) - VM (PSU) X → OUTPUT Z (ADC Iz) This will redefine the Ix, Iy, Iz pins on the ADC to show the on-board current monitor on Ix, the PSU built-in current monitor on Iy, and the PSU built-in voltage monitor on Iz. This allows you to compare current sensing solutions and better insights in how a single channel behaves. Of course, the software reading the values has to be made aware of this change.
J6_pin9, J6_pin10	Top right of board, primary AUX header block	**Default: DISCONNECTED, USER MUST CONNECT MANUALLY** Connecting the SUPD (J6_pin9) with the pin next to it (J6_pin10) is the "hardware enable" for the H-bridge. The H-bridges will **NOT** turn on when this pin is disconnected (unless the "hardware enable" is disabled using JP2). By this we mean: All H-bridge MOSFETs will be **OFF**, because we pull the enable pins of the LTC7060 chips low. For the H-bridges to work, you must connect these with a simple jumper, or connect a big latched switch in between, which can be an optional safety feature to quickly turn off the H-bridge. Note that the silkscreen text of J6_pin10 is wrong: It says GND, but it connects to R30 rather than the board ground plane.

J6_pin11, J6_pin12	Top right of board, primary AUX header block	**Default: DISCONNECTED, USER MUST CONNECT MANUALLY** The 12V DC OUT can be used as an auxiliary power source for a 12V device, whilst protected by the on-board fuse and ripple filter. The barrel jack is additionally protected by a reverse-current protection Schottky diode, which lowers the voltage to approximately 11.7 VDC. For this thesis, this barrel jack drives a 12V tri-axial magnetometer by AlphaLab. By default, the 12V DC OUT barrel jack connection is disabled, so you must bridge the two pins labeled PT, or put a latched switch between them as a bonus enable feature. If you are feeling adventurous, you **can** turn the barrel jack into a 5V power source instead by connecting J6_pin5 (5V) or J6_pin7 (5V) to J6_pin11 (PT). If you do this, a few things must be noted: - The 5V will be directly regulated by the LM7805 on the board. If is **NOT** directly protected by a fuse, so if you accidentally short the barrel jack connection, this chip will pay the price for your error. Although these chips are cheap, the large ground plane on the TO-263 makes them cumbersome to desolder, so this issue is best avoided altogether. - Choose with discernment and ensure the device that you put on the now 5V barrel jack does not pull excessive current. The LM7805 has quite a bit of current headroom, but since it is passively cooled by the ground plane of the board, pulling excessive current will heat up the board in direct proportion. - Take care to protect this setup from naive users. If you are using the barrel jack with 5V, and an uninformed user does not RTFM and connects both PT pins, it may apply over-voltage damage to the connected device. A strategic nearby sticker on the PCB or a cover over J6_pin12 may help avoid this.
JP1	Middle-right of board, near ADC header block	**Default: CONNECTED** When connected, forces the ADC to read current monitor values with board ground as reference. Cut trace to disconnect board ground.
JP2	Bottom right of board, next to PSU Y header block	**Default: CONNECTED** When connected, forces the H-bridge to work only when the DAC sends a high signal on the SW_PSU_ENABLE pin. When disconnected, cuts the emitter-ground connection of Q13 such that the H-bridge works regardless of the SW_PSU_ENABLE pin. The hardware enable (PSU_ENABLE_JUMPER) ignores the setting of JP2.
TP10, TP11	Bottom right of board, next to PSU Z header block	**Default: DISCONNECTED** When connected, forces the H-bridge to work regardless of the status of the hardware enable (PSU_ENABLE_JUMPER), and only makes it a function of the SW_PSU_ENABLE pin. To access this setting: Bridge test pads TP10 and TP11 with a link (they are intentionally positioned adjacent on the board). To make this setting more reliable, resistors R16 and R30 may need to be desoldered and JP2 cut.
JP3	-	Does not exist, no need to look for it.
JP4	Top-right quarter of board, between PWR IN Z terminal block and ADC header block.	**Default: CONNECTED** When connected, makes the 12V bus on the board accessible to the ADC on pin 7 of the ADC connector. This is useful for monitoring the board status in software. The 12V bus is filtered on the board, but if you fear that a spike exceeding 12V may damage your ADC pin front end, you can disconnecting the jumper, which disconnects the 12V bus from the pin.
JP5	Top-right quarter of board, between PWR IN Z terminal block and ADC header block.	**Default: CONNECTED** When connected, makes the 5V bus on the board accessible to the ADC on pin 8 of the ADC connector. This is useful for monitoring the board status in software. The 5V bus is regulated on the board, but if you fear that some voltage spike may damage your ADC pin, you can disconnecting the jumper, which disconnects the 5V bus from the pin.

JP6, JP9, JP12	Top-right of board, next to magnetometer input header block.	**Default: LEFT-MID CONNECTED** By default, the pulse width modulation (PWM) signals that flip the polarity of the H-bridges are supplied through the DAC, because the used CN-0554 board did not come with pass-through headers. The DAC pins have been verified to work, but are in theory less suited for fast, discrete PWM signals. The Schmitt trigger buffers U_B_* on the board ensure that the H-bridge drivers still see a fast rising and falling edge. However, if you would like to drive the bridges using another signal, such as three GPIO pins from the Raspberry Pi, you can insert them into pins 1, 3, and 5 of the secondary AUX header block. You must then enable them with JP6, JP9, and JP12. - When LEFT-MID connected, the H-bridge polarity is driven by pins 1, 5, 11 of the DAC connector (labeled PWMx, PWMy, PWMz), and it ignores the inputs from the AUX header block. This is default behaviour. - When RIGHT-MID connected, the H-bridge polarity is driven by pins 1, 3, 5 of the secondary AUX connector (labeled PWMX, PWMY, PWMZ), and it ignores the inputs from the DAC connector. - When MID is disconnected from either side, the H-bridges cannot be switched in polarity, which cripples the functionality of the board. - When MID is connected both to RIGHT and LEFT, the H-bridges can be driven by both sets of pins. **However, this is has NOT been tested, and the author does NOT recommend ever doing this to prevent potentially overdriving the Schmitt-trigger buffers or the DAC pins on your CN-0554 board.**
JP7, JP8, JP10, JP11, JP13, JP14	Bottom half of the board, all are next to their respective SMD transformer and quartet of rectifier diodes.	**Default: HBR SETTING** This setting changes the voltage at which the H-bridge high-side MOSFET gates are driven. Each jumper affects a single top-side MOSFET. At setting "HBR", the nearby quartet of diodes functions as a half-wave bridge rectifier, driving the corresponding MOSFET with 5.5V (default). With the jumpers at setting "FBR", the rectifiers are configured as a full-wave bridge rectifier, driving the corresponding MOSFET with 11V. See Figure D.3 for an example. The MOSFETs that are specified in the Bill of Materials are rated to work with both settings, and this has been verified by the author. If you are using different MOSFETs that seem to not switch well or run excessively hot, you may drive them with 11V, so long as they can handle it. - HBR setting: drive corresponding H-bridge high-side MOSFET with 5.5 V. - FBR setting: drive corresponding H-bridge high-side MOSFET with 11 V. - Disconnecting both sides will break the H-bridges. - Connecting both sides serves no purpose beyond asking for trouble.

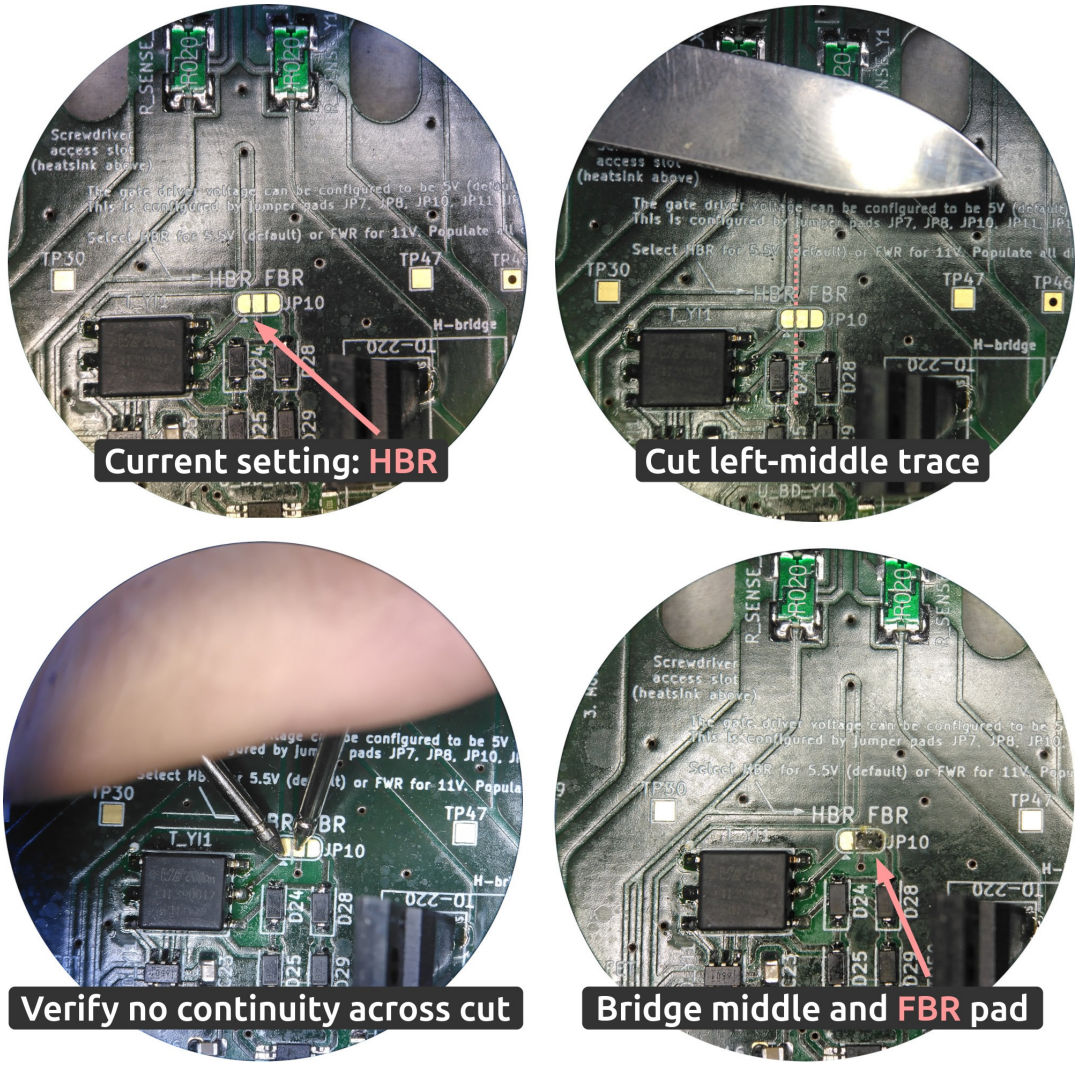


Figure D.3: Example of changing a board configuration by modifying the jumper pad. In this particular case, it will cause the gate of high-side MOSFET Q5 to be driven by 11 V rather than 5.5 V.

D.4. Dropper diode mounting options

The V2.2 board offers a few possible mounting configurations for the dropper diodes. The author chose to mount them underneath the board, legs bent at a 90 degree angle, front of the TO-220 pointing upwards. There are several other ways to mount them, including pointing upwards. Just ensure that the direction of the anode and the cathode is not reversed (if you do, the board should still work, but the input polarity is reversed, making the silkscreen markings confusing). The diodes will want some sort of heat sink with either passive or forced convection, as they can get pretty warm. Because these diodes are jacketed with plastic, they can be mounted to any large block of metal without making electrical contact.

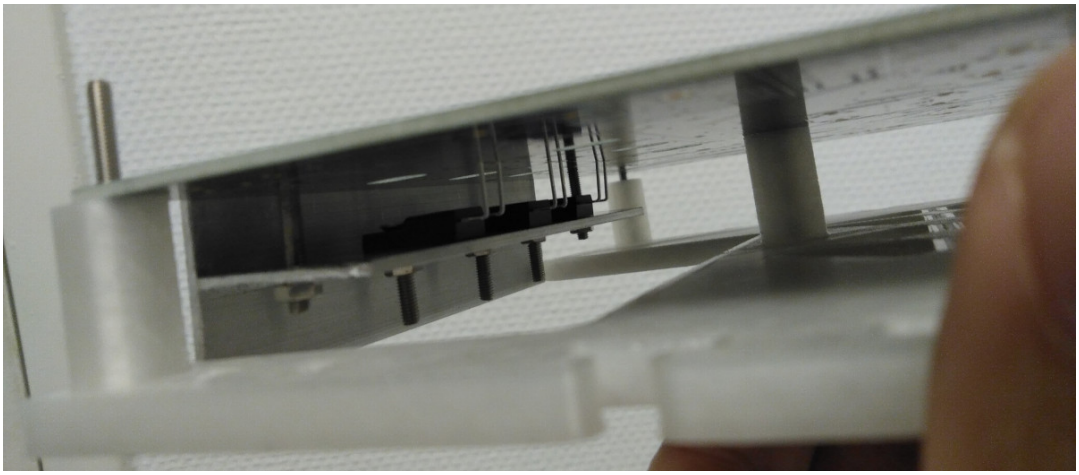


Figure D.4: Mounting the diodes underneath the board, facing up. Two screws through the auxiliary board holes on the side carry the weight of the heat sink and electrically ground it. The diodes are mounted to the heatsink by tightening a screw through the top, which is facilitated by the access slots in the board above.

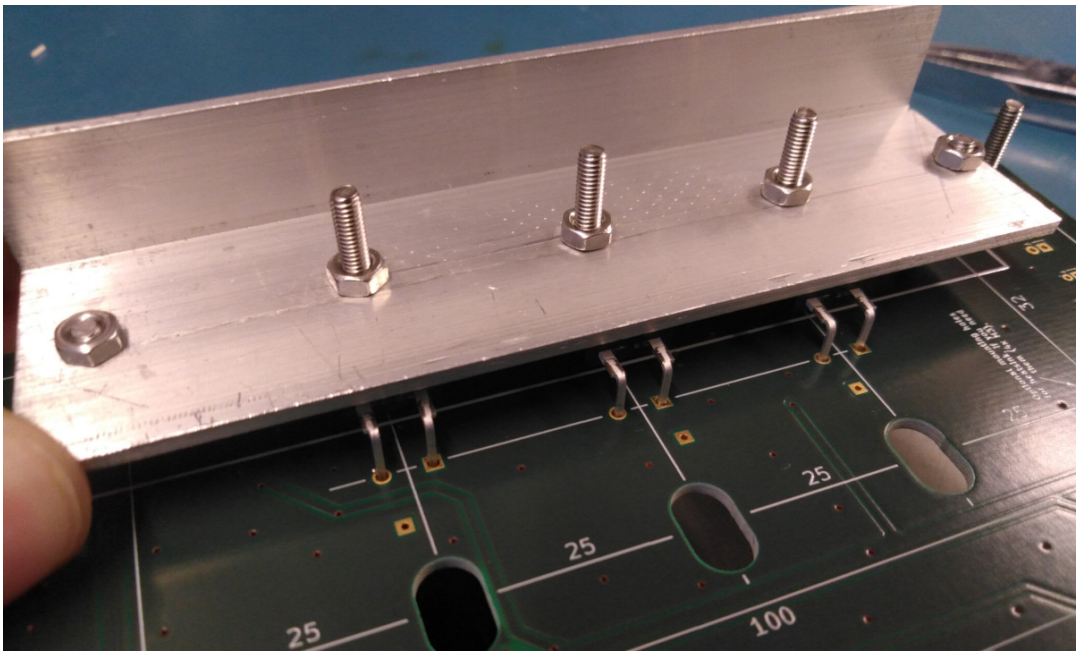


Figure D.5: Same as Figure D.4 but viewed from the bottom. Note that the diodes are positioned before soldering, which is the recommended method of mounting them.

Bill of materials

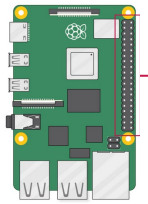
3	107CKS025MEM	Electrolytic Capacitor	100uF	D5	C1	Input filtering	Digi-Key	1572-1659-ND	1	1	0.206	1	0.206	https://www.digikey.nl/en/products/detail/cornell-dubilier-illinois-capacitor/107CKS025MEM/5410
4	EEA-GA1E100B	Electrolytic Capacitor	10uF	D4	C2	Input filtering	Digi-Key	P15798CT-ND	1	1	0.29	1	0.29	https://www.digikey.nl/en/products/detail/panasonic-electronic-components/EEA-GA1E100B/251
5	CL21A106KOQNNNE	Ceramic Capacitor	10uF	0805	C7,C15,C27	Filter	Digi-Key	1276-1096-1-ND	3	3	0.109	1	0.327	https://www.digikey.nl/en/products/detail/samsung-electro-mechanics/CL21A106KOQNNNE/3886
6	CL21B105KOFNNNG	Ceramic Capacitor	1uF	0805	C3,C5,C8,C9,C10,C11,C12,C13,C14,C16,C17,C18,C22,C23,C24,C25,C26,C28,C29,C30,C31,C32,C33	Bypass, filter	Digi-Key	1276-6471-1-ND	23	23	0.109	1	2.507	https://www.digikey.nl/en/products/detail/samsung-electro-mechanics/CL21B105KOFNNNG/3894
7	CL21B104KBCNNNC	Ceramic Capacitor	100nF	0805	C4,C6,C19,C20,C21	Bypass, filter	Digi-Key	1276-1003-1-ND	5	5	0.109	1	0.545	https://www.digikey.nl/en/products/detail/samsung-electro-mechanics/CL21B104KBCNNNC/3886
8	STTH8L06FP	Diode		TO-220	D6,D20,D32	Dropper, surge protection	Digi-Key	497-12691-5-ND	3	3	1.609	1	4.827	https://www.digikey.nl/en/products/detail/stmicroelectronics/STTH8L06FP/1039820
9	MMSZ5231B-7-F	Zener diode	5.1V	SOD-123	D7,D21,D33	Overvoltage, undervoltage protection	Digi-Key	MMSZ5231B-FDICT-ND	3	3	0.109	1	0.327	https://www.digikey.nl/en/products/detail/diodes-incorporated/MMSZ5231B-7-F/755472
10	MBR0520LT1G	Schottky diode		SOD-123	D1,D11,D12,D13,D15,D16,D17,D18,D19,D24,D25,D26,D27,D28,D29,D30,D31,D36,D37,D38,D39,D40,D41,D42,D43	Rev. current protection, rectifier	Digi-Key	MBR0520LT1GOSCT-ND	25	25	0.254	1	6.35	https://www.digikey.nl/en/products/detail/onsemi/MBR0520LT1G/918574
11	1.5SMB56CA	Bi-directional TVS diode	56V	DO-214	D9,D22,D34	H-bridge transient suppression	Digi-Key	1B-1.5SMB56CACT-ND	3	3	0.774	1	2.322	https://www.digikey.nl/en/products/detail/littelfuse-inc/1-5SMB56CA/10477923
12	150080AS75000	LED		0805	D3,D14,D46,D47,D48	Indicator (amber)	Digi-Key	732-150080AS75000CT-ND	5	5	0.266	1	1.33	https://www.digikey.nl/en/products/detail/w%C3%BCrth-elektronik/150080AS75000/11479384
13	150080BS75000	LED		0805	D4	Indicator (blue)	Digi-Key	732-4982-1-ND	1	1	0.206	1	0.206	https://www.digikey.nl/en/products/detail/w%C3%BCrth-elektronik/150080BS75000/4489912
14	150080VS75000	LED		0805	D2	Indicator (green)	Digi-Key	732-4986-1-ND	1	1	0.206	1	0.206	https://www.digikey.nl/en/products/detail/w%C3%BCrth-elektronik/150080VS75000/4489924
15	150080RS75000	LED		0805	D5,D8	Indicator (red)	Digi-Key	732-4984-1-ND	2	2	0.206	1	0.412	https://www.digikey.nl/en/products/detail/w%C3%BCrth-elektronik/150080RS75000/4489918
16	SF-1206F100-2	Fuse	1A	1206	F1	Overcurrent/short-circuit protecti	Digi-Key	SF-1206F100-2CT-ND	1	1	0.629	1	0.629	https://www.digikey.nl/en/products/detail/bourns-inc/SF-1206F100-2/1948166
17	MH2029-121Y	Ferrite bead	120R	0805	FB1,FB2,FB3,FB4,FB5,FB6,FB7,FB8,FB9,FB10,FB11,FB12,FB13,FB14,FB15,FB16,FB17	HF noise rejection	Digi-Key	MH2029-121YCT-ND	17	17	0.109	1	1.853	https://www.digikey.nl/en/products/detail/bourns-inc/MH2029-121Y/1970499
18	PJ-102AH	Connector	5.5mm	DC Jac	J1,J4	Power in, power out	Digi-Key	CP-102AH-ND	2	2	0.64	1	1.28	https://www.digikey.nl/en/products/detail/cui-devices/PJ-102AH/408448
19	TB002-500-02BE	Connector	01x02	Terminal block	J11,J13,J15	PSU connector	Digi-Key	102-6145-ND	3	3	0.823	1	2.469	https://www.digikey.nl/en/products/detail/cui-devices/TB002-500-02BE/10064069
20	TB002-500-04BE	Connector	01x04	Terminal block	J12,J14,J16	Load connector	Digi-Key	102-6147-ND	3	3	1.198	1	3.594	https://www.digikey.nl/en/products/detail/cui-devices/TB002-500-04BE/10064071
21	61200621621	Connector	02x03	Box head	J2,J5,J10,J18	Hardware interfacing	Digi-Key	732-5394-ND	4	4	0.532	1	2.128	https://www.digikey.nl/en/products/detail/w%C3%BCrth-elektronik/61200621621/4846913
22	302-5201	Connector	02x10	Box head	J3,J8	Hardware interfacing	Digi-Key	ED10524-ND	2	2	0.665	1	1.33	https://www.digikey.nl/en/products/detail/on-shore-technology-inc/302-5201/2794235
23	10129381-920001BLF	Connector	02x10	Pin head	J6	Utility/auxiliary interfacing	Digi-Key	10129381-920001BLF-ND	1	1	0.46	1	0.46	https://www.digikey.nl/en/products/detail/amphenol-cs-fci/10129381-920001BLF/7916094
24	10129381-906002BLF	Connector	02x03	Pin head	J9	Utility/auxiliary interfacing	Digi-Key	10129381-906002BLF-ND	1	1	0.944	1	0.944	https://www.digikey.nl/en/products/detail/amphenol-cs-fci/10129381-906002BLF/7916071
25	826646-4	Connector	01x04	Pin head	J17,J19	Monitor select pins	Digi-Key	A122909-ND	3	3	0.557	1	1.671	https://www.digikey.nl/en/products/detail/te-connectivity-amp-connectors/826646-4/1152071
26	BC847C,215	BJT	NPN	SOT-23	Q13,Q14,Q15	H-bridge enable logic	Digi-Key	1727-2924-1-ND	3	3	0.109	1	0.327	https://www.digikey.nl/en/products/detail/nexperia-usa-inc/BC847C-215/763240
27	FDPF045N10A	MOSFET	N-channel	TO-220	Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11, Q12	H-bridge switches	Digi-Key	FDPF045N10A-ND	12	12	4.731	1	56.772	https://www.digikey.nl/en/products/detail/onsemi/FDPF045N10A/3163348
28	SG73S2ATTD1000F	Resistor	100	0805	R5,R6,R7,R12,R13,R17,R22,R24,R25,R36,R37,R39,R40,R43,R44	Gate stopper resistor, filtering	Digi-Key	2019-SG73S2ATTD1000FCT-ND	15	15	0.206	1	3.09	https://www.digikey.nl/en/products/detail/koa-speer-electronics-inc/SG7352ATTD1000F/10279947
29	RK73H2ATTD2201F	Resistor	2.2k	0805	R3,R15,R31,R34,R35,R46	LED current limiter	Digi-Key	2019-RK73H2ATTD2201FCT-ND	6	6	0.109	1	0.654	https://www.digikey.nl/en/products/detail/koa-speer-electronics-inc/RK73H2ATTD2201F/1023426
30	RK73H2ATTD4701F	Resistor	4.7k	0805	R1,R2,R4	LED current limiter	Digi-Key	2019-RK73H2ATTD4701FCT-ND	3	3	0.109	1	0.327	https://www.digikey.nl/en/products/detail/koa-speer-electronics-inc/RK73H2ATTD4701F/1023429
31	RK73H2ATTD6801F	Resistor	6.8k	0805	R14	LED current limiter	Digi-Key	2019-RK73H2ATTD6801FCT-ND	1	1	0.109	1	0.109	https://www.digikey.nl/en/products/detail/koa-speer-electronics-inc/RK73H2ATTD6801F/1023456
32	RK73H2ATTD1002F	Resistor	10k	0805	R16,R30,R32,R33	H-bridge enable logic	Digi-Key	2019-RK73H2ATTD1002FCT-ND	4	4	0.109	1	0.436	https://www.digikey.nl/en/products/detail/koa-speer-electronics-inc/RK73H2ATTD1002F/1023413
33	RK73H2ATTD5102F	Resistor	51k	0805	R8,R9,R18,R19,R26,R27	IC configuration	Digi-Key	2019-RK73H2ATTD5102FCT-ND	6	6	0.109	1	0.654	https://www.digikey.nl/en/products/detail/koa-speer-electronics-inc/RK73H2ATTD5102F/1023405
34	RK73B2ATTD104J	Resistor	100k	0805	R10,R11,R20,R21,R23,R28,R29,R38,R41,R42,R45	IC configuration, bleeder	Digi-Key	2019-RK73B2ATTD104JCT-ND	11	11	0.109	1	1.199	https://www.digikey.nl/en/products/detail/koa-speer-electronics-inc/RK73B2ATTD104J/10236578
35	LVK24R020DER	Resistor	20m	1206	R_SENSE_X1,R_SENSE_Y1,R_SENSE_Z1	Current sense shunt resistor	Digi-Key	LVK24R020DERCT-ND	3	3	1.658	1	4.974	https://www.digikey.nl/en/products/detail/ohmite/LVK24R020DER/1632754
36	760390012	Transformer	1:1.1	6.73x7.4.32	T_XI1,T_XN1,T_YI1,T_YN1,T_ZI1,T_ZN1	Isolation transformer	Digi-Key	732-4456-1-ND	6	6	3.013	1	18.078	https://www.digikey.nl/en/products/detail/w%C3%BCrth-elektronik/760390012/3831260
37	MC7805BD2TR4G	Voltage regulator		TO-263	U1	Linear voltage regulator 12VDC -> 5VDC	Digi-Key	MC7805BD2TR4GOSCT-ND	1	1	1.609	1	1.609	https://www.digikey.nl/en/products/detail/onsemi/MC7805BD2TR4G/919330
38	LTC7060EMSE#TRPBF	Half-bridge driver		MSOP-heatsi	U_BD_XI1,U_BD_XN1,U_BD_YI1,U_BD_YN1,U_BD_ZI1,U_BD_ZN1	Drive H-bridge networks	Digi-Key	505-LTC7060EMSE#TRPBFCT-ND	6	6	5.08	1	30.48	https://www.digikey.nl/en/products/detail/analog-devices-inc/LTC7060EMSE-TRPBF/12385830
39	SN74LVC1G17DBVR	Buffer		SOT-23	U_B_X1,U_B_Y1,U_B_Z1	Schmitt-trigger buffer, line driver	Digi-Key	296-11933-1-ND	3	3	0.109	1	0.327	https://www.digikey.nl/en/products/detail/texas-instruments/SN74LVC1G17DBVR/389051
40	INA2290A2IDGKR	Amplifier		8-VSSO	U_CS_XY1,U_CS_Z1	Current sense amplifier	Digi-Key	296-INA2290A2IDGKRCT-ND	2	2	5.227	1	10.454	https://www.digikey.nl/en/products/detail/texas-instruments/INA2290A2IDGKR/13636614
41	SN6501DBVR	Transformer driver		SOT-23	U_TD_XI1,U_TD_XN1,U_TD_YI1,U_TD_YN1,U_TD_ZI1,U_TD_ZN1	Push-pull transformer driver	Digi-Key	296-37700-1-ND	6	6	2.081	1	12.486	https://www.digikey.nl/en/products/detail/texas-instruments/SN6501DBVR/3431175
42	NPC02SXON-RC	Jumper			EXTRA1, EXTRA2, EXTRA3, EXTRA4, EXTRA5	Default configuration jumpers Note: NOT on schematic	Digi-Key	S9341-ND	5	5	0.109	1	0.545	https://www.digikey.nl/en/products/detail/sullins-connector-solutions/NPC02SXON-RC/2618266
43	V02	Barrel jack plug (f)			EXTRA6, EXTRA7	Power from board to magnetome	Digi-Key	1939-V02-ND	2	2	1.2	1	2.4	https://www.digikey.nl/nl/products/detail/globtek-inc/V02/12341457
44									1	1			0	

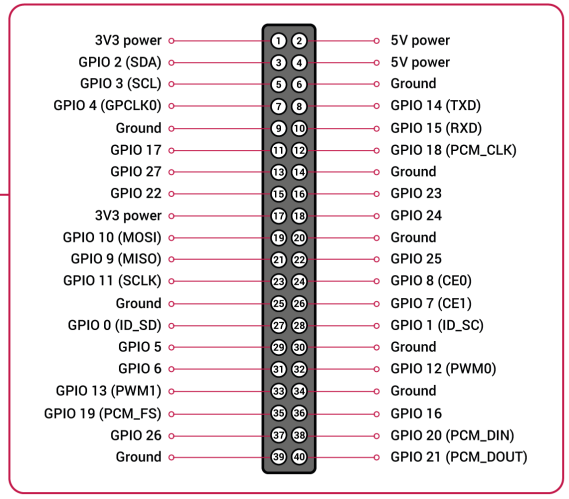
CN0554 DAC/ADC BOARD



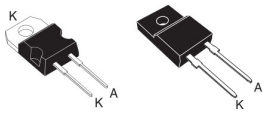
DAC CONNECTOR				ADC CONNECTOR			
	1	2			1	2	
	3	4			3	4	
	5	6			5	6	
	7	8			7	8	
OUTPUT0	9	10	OUTPUT1	INPUT0	9	10	INPUT1
OUTPUT2	11	12	OUTPUT3	INPUT2	11	12	INPUT3
OUTPUT4	13	14	OUTPUT5	INPUT4	13	14	INPUT5
OUTPUT6	15	16	OUTPUT7	INPUT6	15	16	INPUT7
GND	17	18	GND	GND	17	18	GND
OUTPUT8	19	20	OUTPUT9	INPUT8	19	20	INPUT9
OUTPUT10	21	22	OUTPUT11	INPUT10	21	22	INPUT11
OUTPUT12	23	24	OUTPUT13	INPUT12	23	24	INPUT13
OUTPUT14	25	26	OUTPUT15	INPUT14	25	26	INPUT15
GND	27	28	GND	GND	27	28	GND
	29	30			29	30	

RASPBERRY PI 40PIN HEADER





STTH8L06 diode



TO-220AC
STTH8L06D

TO-220FPAC
STTH8L06FP

SN6501 TRANSFORMER DRIVER

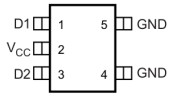
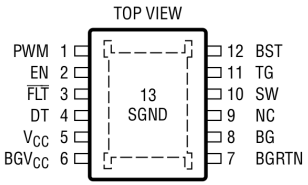


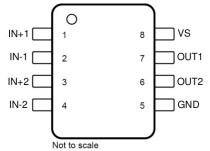
Figure 4-1. 5-Pin SOT-23 DBV Package Top View

LTC7060 BRIDGE DRIVER



MSE PACKAGE
12-LEAD PLASTIC MSOP

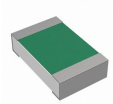
INA2290 DUAL SENSE AMP



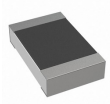
gure 5-2. INA2290: DGK Package 8-Pin VSSOP Top View

SMD RESISTORS

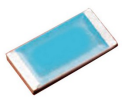
100
green



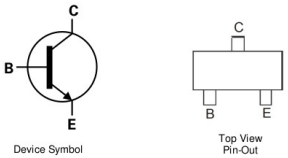
100K
black



OTHERS
blue



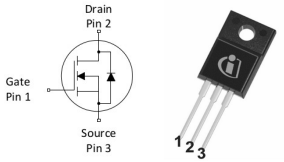
BC847 NPN



IPA045 N MOSFET

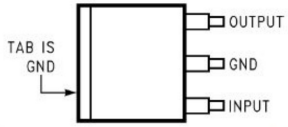
Pinout: GDS

TO-220-FP



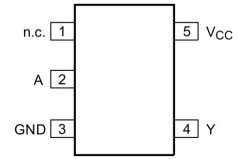
LM7805 5V REGULATOR

LM7805 and LM7812 KTT Package
3-Pin DDPAK/TO-263
Top View



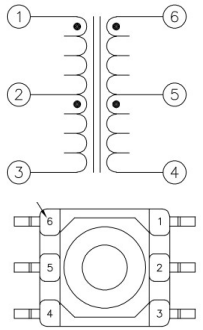
SN74LVC1G17 Schmitt-trigger buffer

GW package
SOT353-1 (TSSOP5)



aaa-035693

760390011



LVK24 SHUNT RESISTOR

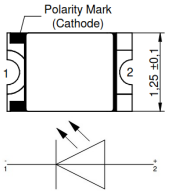


V = sensing terminal (voltage)
I = current terminal

Layout for illustration only, part can be rotated 180° without effect to the circuit:



SMD LED



E

Modifying the ES-030-10 power supply for analog control

After it was discovered that the BK Precision 1685B power supplies were incapable of properly driving the inductive load represented by the Helmholtz cage coils, it was investigated whether the previous power supplies could be used. These were Delta Elektronik ES-030-10 power supplies.

Unfortunately, these power supplies were manufactured according to factory option P120, which removes the DB15 control port on the back of the base model power supply, and instead replaces it with an RS-232 interface on a DB9 connector. This was useful for the Helmholtz cage setup in the past, in which a PC was controlling the three power supplies via these RS-232 connectors. However, for the new Helmholtz cage setup, analog control is very much preferred. Sadly, the RS-232 connectors do not expose the analog control inputs.

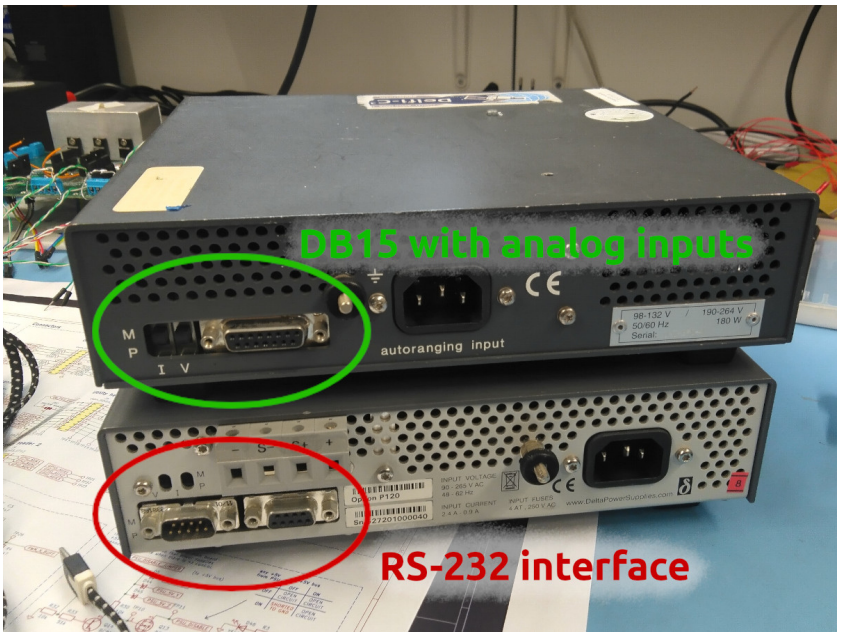


Figure E.1: The rear panel on two Delta Elektronik power supplies. The top one features the standard DB15 control ports, which does feature the analog control pins. The bottom one contains the RS-232 interface connector. The power supplies in question all contained the RS-232 interface.

After opening up a power supply, it was discovered that the RS-232 interface was on a separate board inside the power supply. The main task of this board is to handle the instructions sent over the RS-232, and control the power supply accordingly. The board contains a digital-to-analog converter (DAC) which generate analog voltages that can control the voltage and/or current of the power supply. Conveniently, the DB15 connector is still present inside the power supply, but its pins are not easily exposed.

This appendix describes the process of modifying the power supply by removing this RS-232 board, and interfacing with the analog control pins directly. It is a fairly straightforward modification that is completely reversible.

Required tools

The modification requires the following tools:

- Proper ESD-protected working environment
- Multimeter
- Soldering iron
- Small hex-wrench or pliers
- Cutting pliers
- A screwdriver with a 6-lobe Torx bit
- A length of IDC ribbon cable (at least width 4, recommended at least 0.5 m long)
- Two 3x2 female IDC connector or other desired cable interfacing
- Small zip ties for cable management
- Storage container for spare parts (preferably ESD protected)

Modification process

First, prepare a clean working environment with plenty of space. **Ensure that any power cables are disconnected from the supply!**

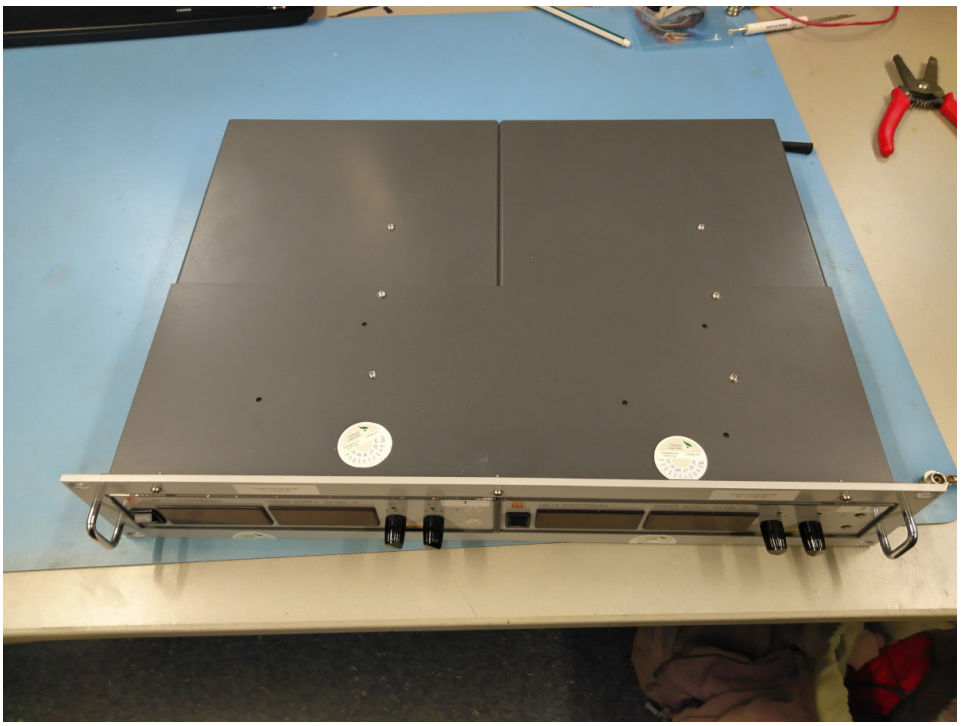


Figure E.2: Two ES-030-10 power supplies in a twin rack mount.

If the power supply is in a twin rack mount, remove it from the rack mount first, by unscrewing two of the three Torx screws at the top.

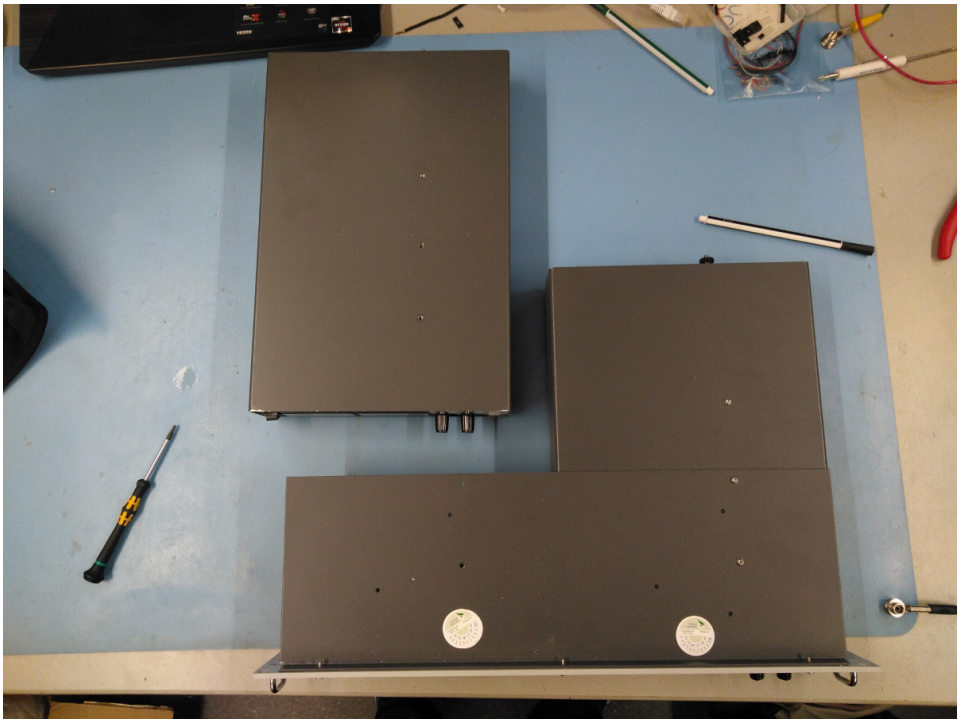


Figure E.3: Disconnecting the power supplies from their rack mount.

Unscrewing the final Torx screw will allow for the top of the chassis to come off off.

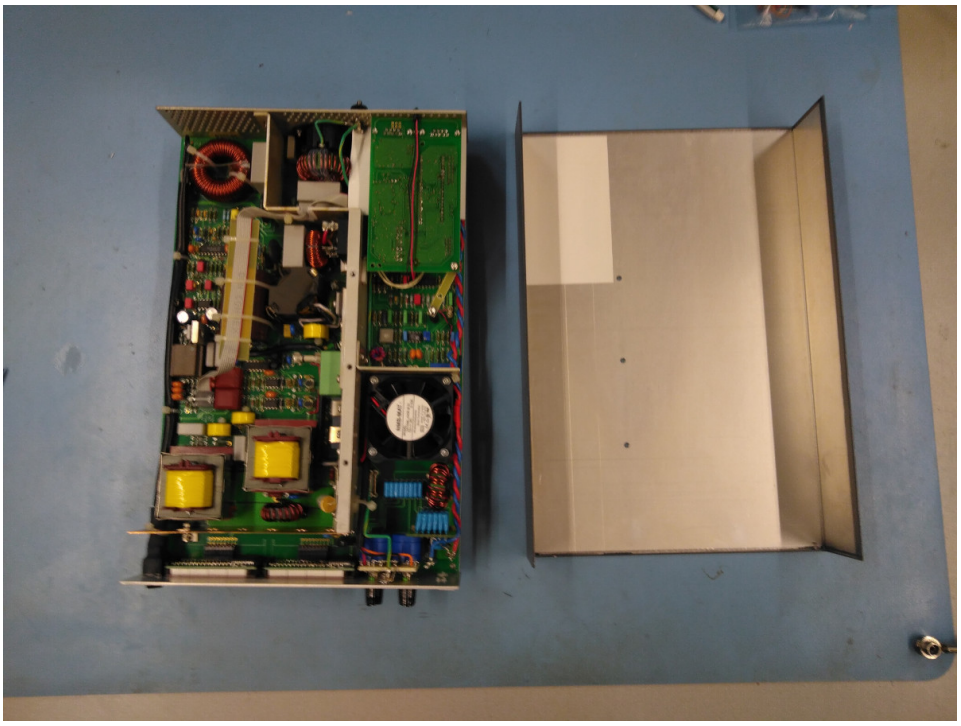


Figure E.4: Opening the power supply chassis.

With the internals of the power supplies exposed, it would be wise at this point to measure the input capacitors and the enormous electrolytic output capacitor inside the power supply with the multimeter, and ensure that they have been discharged. If they contain charge, wait for them to discharge, or discharge them manually with an external bleeder resistor.

Now turn to the back of the power supply. Unscrew the two DB9 connectors (four screws) and the single screw at the diagonal PCB strip on the other end of the board. The RS-232 interface board should now come loose.

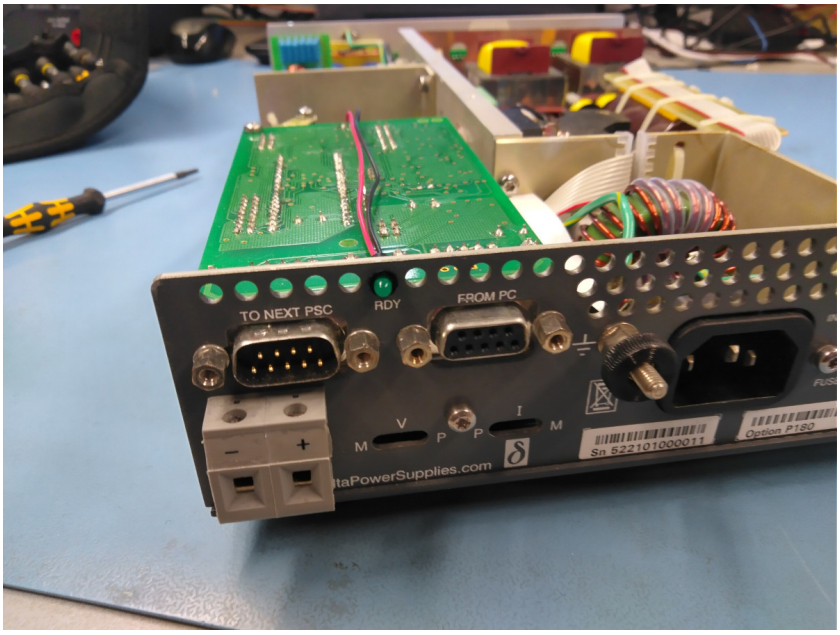


Figure E.5: Back of the internal RS-232 board.

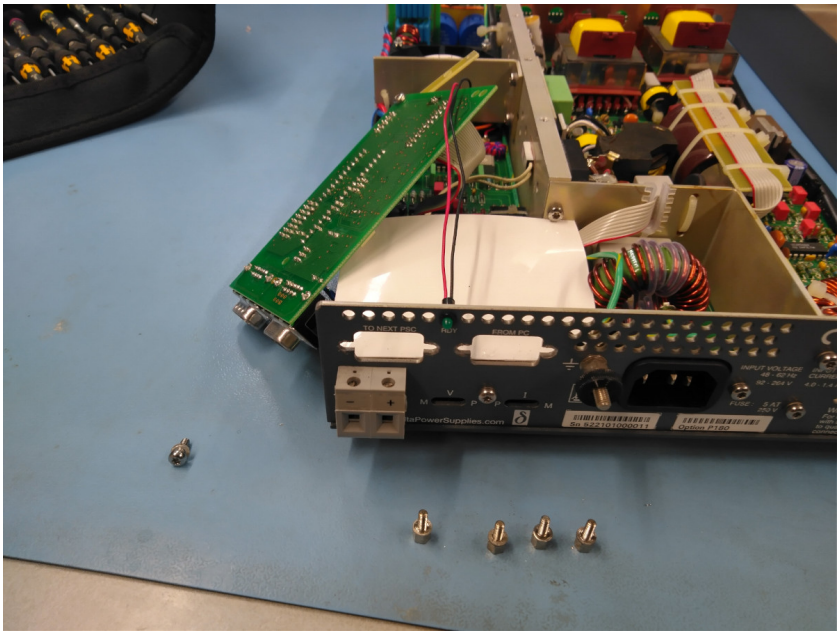


Figure E.6: RS-232 board after disconnecting the mounting screws.

Now heat up the soldering iron and desolder the two wires connected to the "RDY" indicator LED.

It would be wise to indicate which wire corresponds to which hole with a label/sticker, in case the modification should be reversed in the future. Now push the LED and its socket through the chassis to free it.

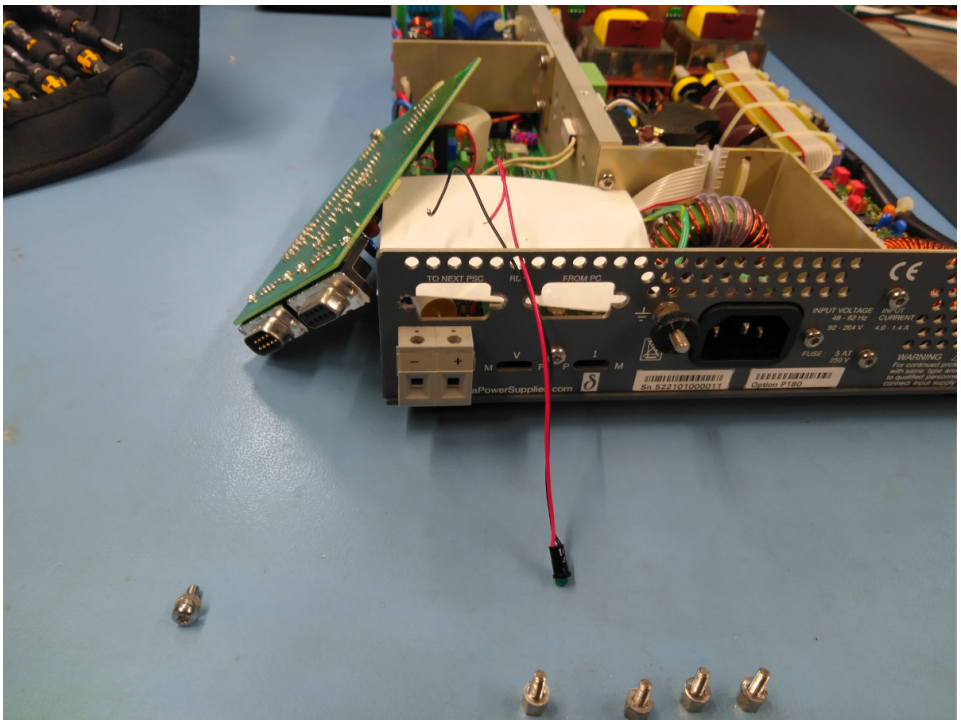


Figure E.7: Freeing the indicator LED.

Remove the sheet of insulation paper from underneath the RS-232 board.

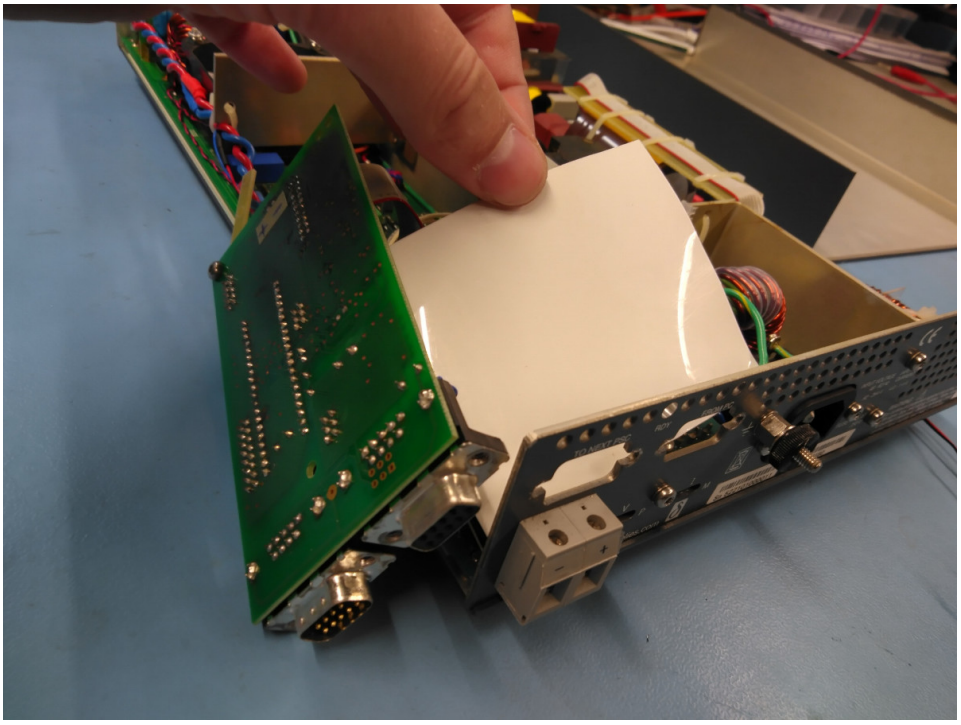


Figure E.8: Removing the insulation sheet.

Disconnect the 10x2 ribbon connector that links the RS-232 board with the smaller routing board underneath. Leave the other connections to this routing board as they are, including the other ribbon connectors.

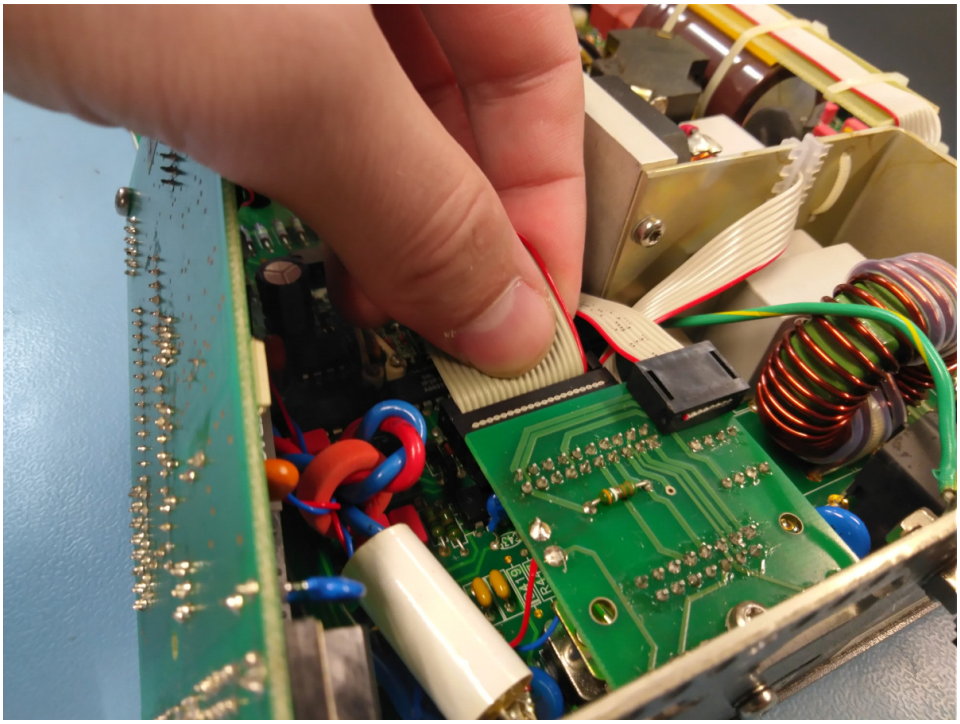


Figure E.9: Disconnecting the 10x2 ribbon connector from the RS-232 board.

Extract the RS-232 interface board and store it along with the screws, the insulation paper, and the socketed LED.

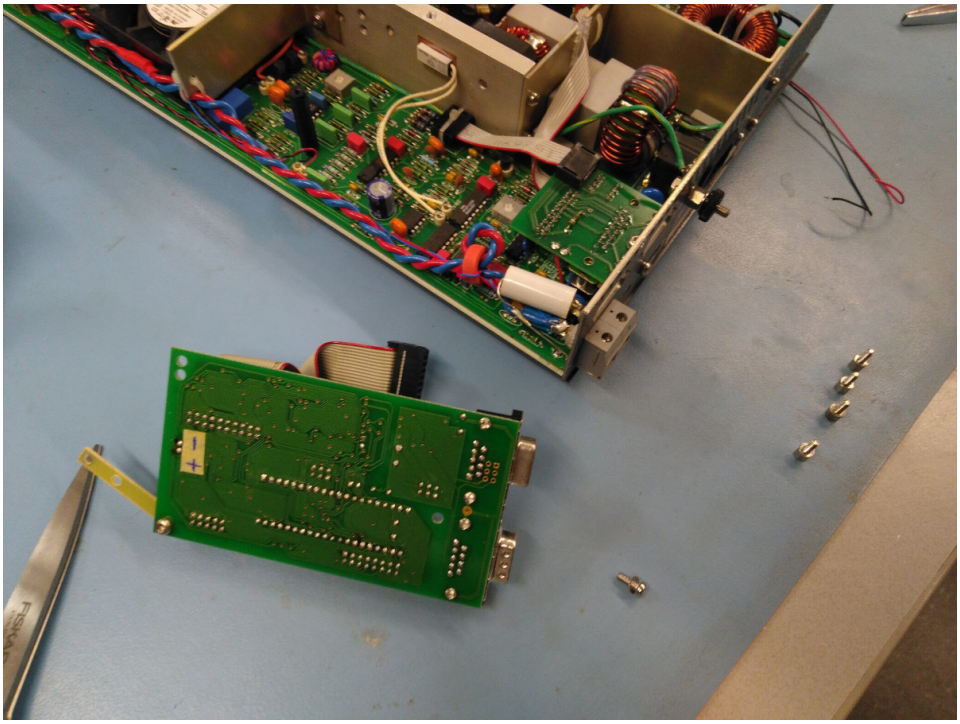


Figure E.10: Successful extraction of the internal RS-232 card.

The small routing board in the power supply contains a 15 pin DB15 connector that acts as an interconnect between it and the main power supply board. The pin-out of the DB15 connector seems identical to the DB15 connector that comes with the standard ES-030-10 power supplies. While there is a temptation to remove the routing board and just connect to the DB15 board directly, the power supply will not properly without it. So instead, the analog control pins will be exposed via the 10x2 header pins that previously connected to the RS-232 interface board.

	description	pin	description
1	0, return of reference, prog. inputs and monitor outputs.	9	Reference voltage 5.16V
2	current monitor output (0-5V)	10	Voltage monitor output (0-5V)
3	current programming input (0-5V)	11	Voltage programming input (0-5V)
4	CC status output, logic 1 = CC mode (5V / 4mA)	12	NC
5	Remote Shuntdown (+5V / 1mA)	13	NC
6	NC	14	NC
7	+12V output (Ri=500 Ohm)	15	NC
8	0, return of status outputs, +12V and Remote Shuntdown.		

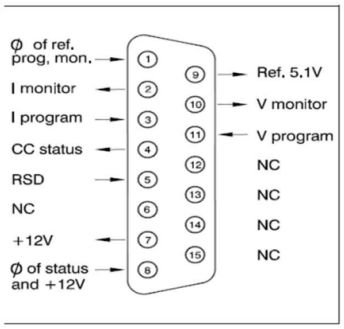


Figure E.11: Pin-out of the DB15 interface connector for the Delta Elektronik ES-300 series.

To do this, one needs to know how the routing board maps the DB15 pins onto the 10x2 header pin block. The pins that are needed in this case are the **"I program"** pin and the ground pin that the input voltage is referred to. Optionally, the **"V program"** can be connected to allow software control for it, and the **"I monitor"** pin can be connected to measure the output current.

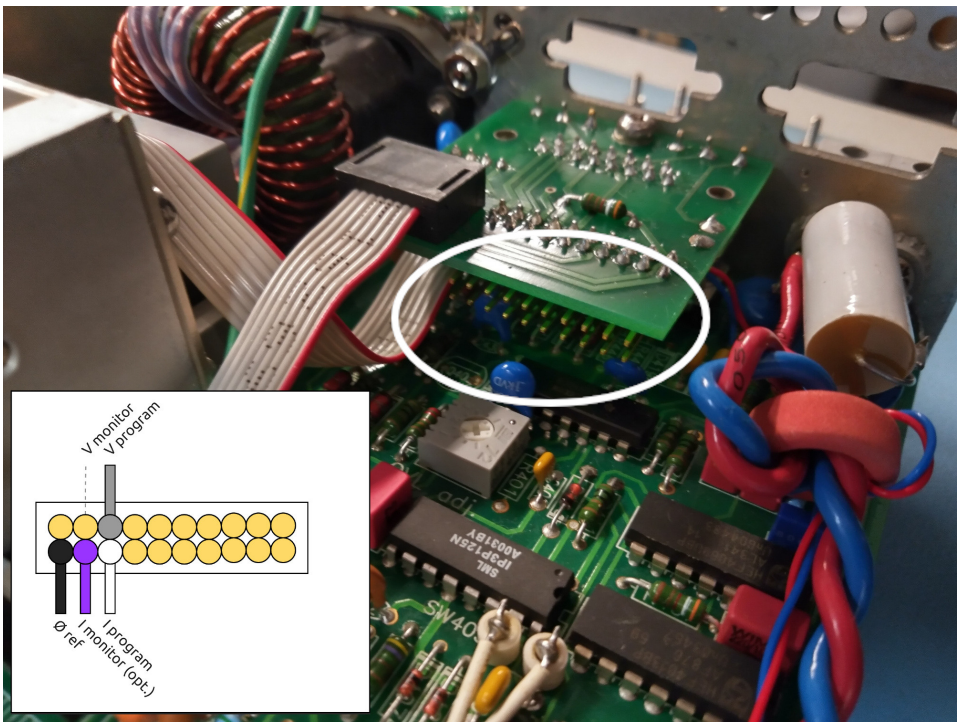


Figure E.12: Location of the analog pins on the internal 10x2 header.

Prepare an interfacing cable to expose the desired pins to the outside of the power supply. Since the header pins allow for bidirectional insertion of a connector, it may be wise to mark one side of the connector so it is less likely to be inserted the wrong way around. Also ensure that any unused cables have their ends insulated so that they cannot make any electrical connection inside the power supply.

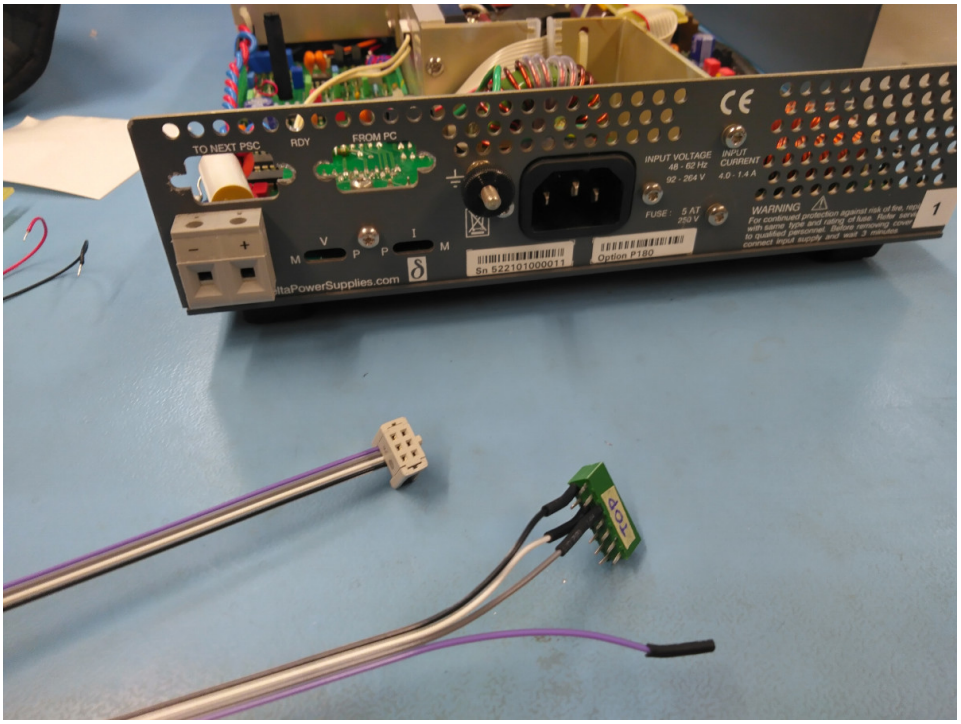


Figure E.13: Preparing a temporary internal connector.

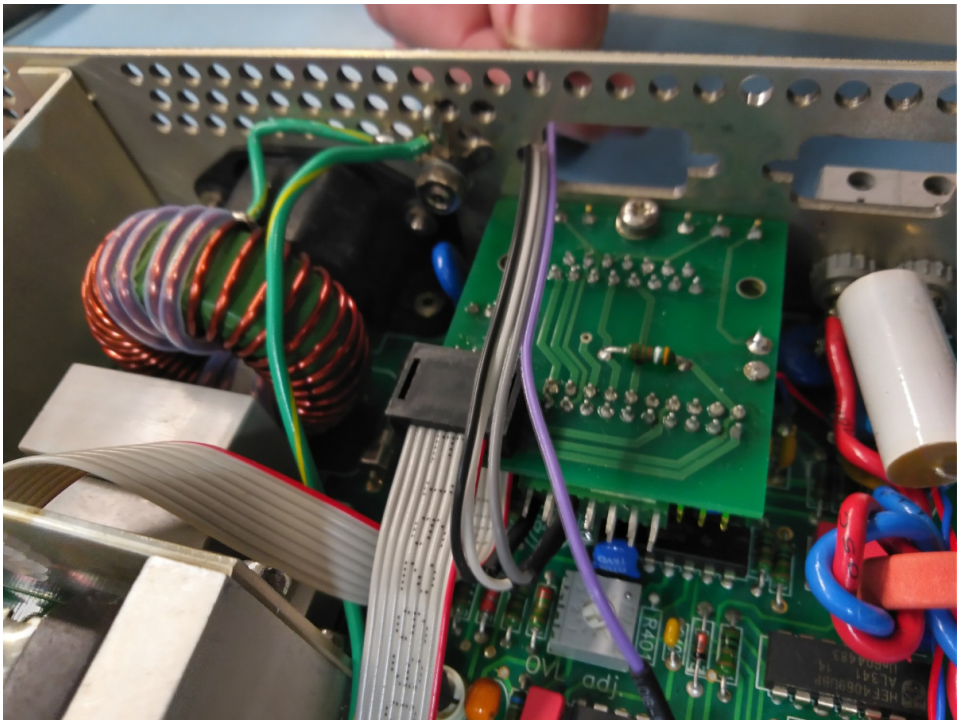


Figure E.14: Connecting and routing the temporary internal connector.

It is recommended to use a tightly secured zip tie around the ribbon to act as a strain relief, so that the connections at the header pins do not take any strain in case the ribbon is pulled from the outside of the power supply.

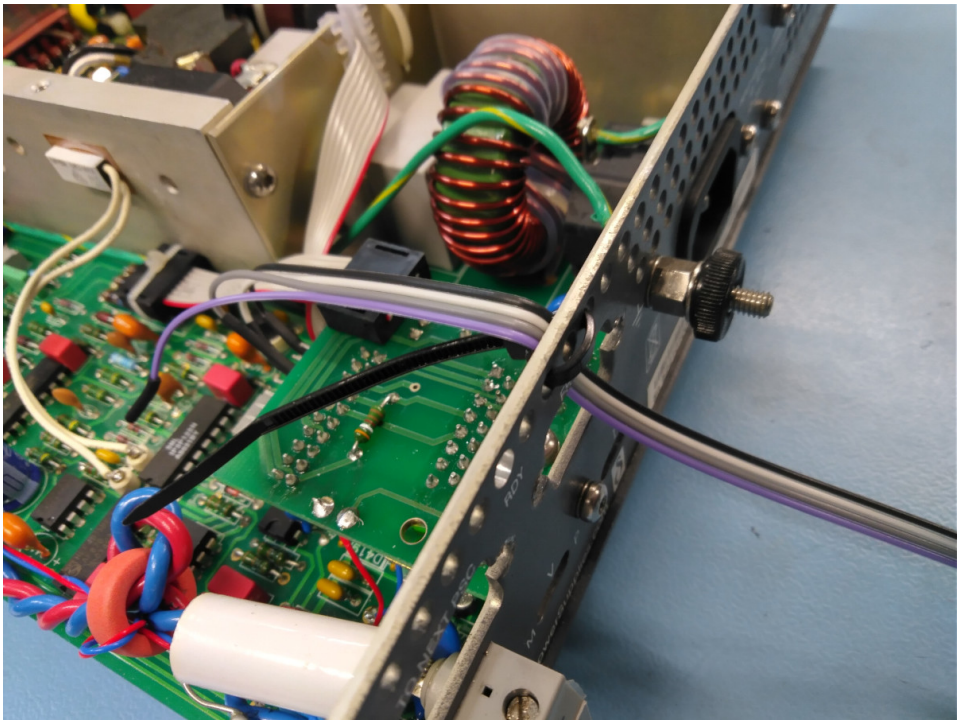


Figure E.15: Using a zip tie as strain relief for the cable.

Close the chassis back up, and reconnect it to the rack mounting plate using the three Torx screws.

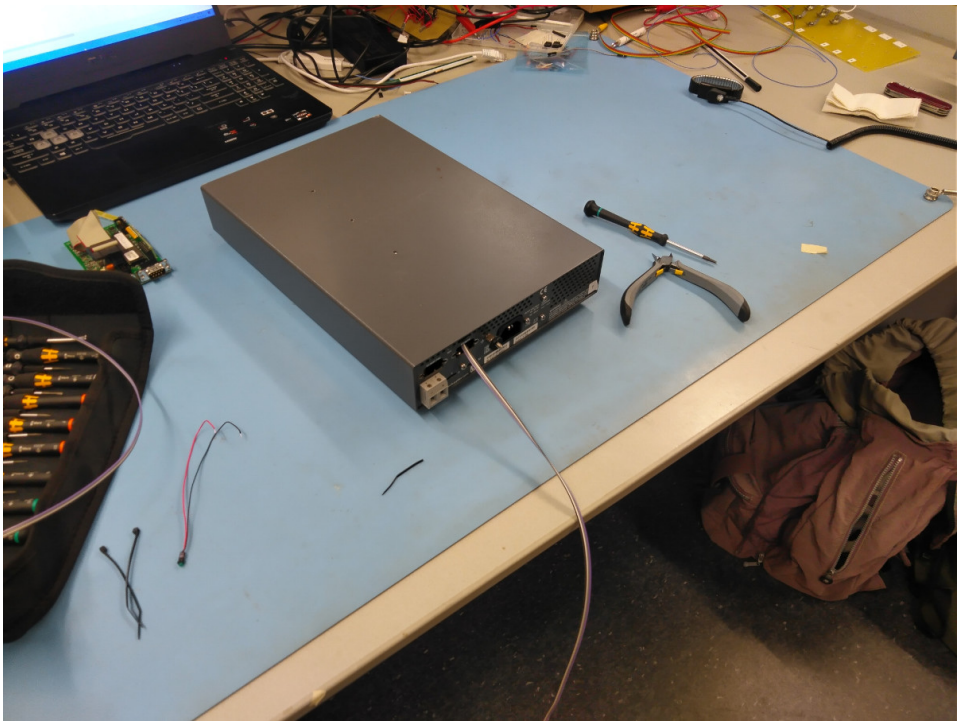


Figure E.16: Closing the chassis.

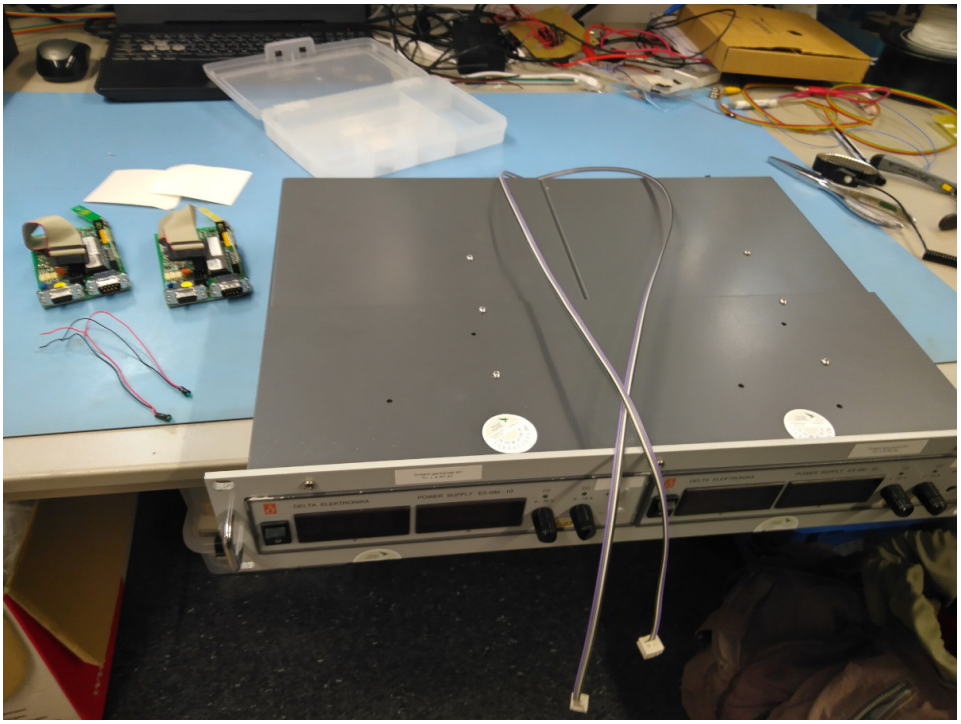


Figure E.17: Two successfully modified ES-030-10 units back inside their rack mount.

The current output by the power supplies can now be controlled by applying a voltage in the range of $[0, 5]V$ between the **"I program"** pin and its reference ground. Before this is attempted, the **"I"** toggle switch should be on **"P"** (program), rather than on **"M"** (manual).

The **"V"** toggle switch can either be on **"P"** or **"M"** depending on the user needs. When on "manual", a voltage limit can be set with the potentiometer on the front of the unit, and if on "program", this voltage limit can be set with a $[0, 5]V$ analog control voltage on the **"V program"** pin.

If calibrated properly, this power supply is impressively linear, and a control voltage of 1 V should yield very close to 20% of the current output, which for the ES-030-10 should be 2 A.

After verifying that the power supply mod worked as desired, the power supplies were re-opened to tidy up the cable connection. The empty hole from the RS-232 connector was filled with a 2x3 Insulation Displacement Connector (IDC) terminal block, and mechanically interfacing via a small 3D-printed coupler. This allows for a more flexible solution, and allows the cable to detach upon receiving excessive strain, rather than straining until breakage or internal damage occurs. It also documents itself more explicitly to the user.

CC (I control) [0-5V]	VC (V control) [0-5V]	PSU internal 5.1V
CM (I monitor) [0-5V]	VM (V monitor) [0-5V]	PSU internal ground

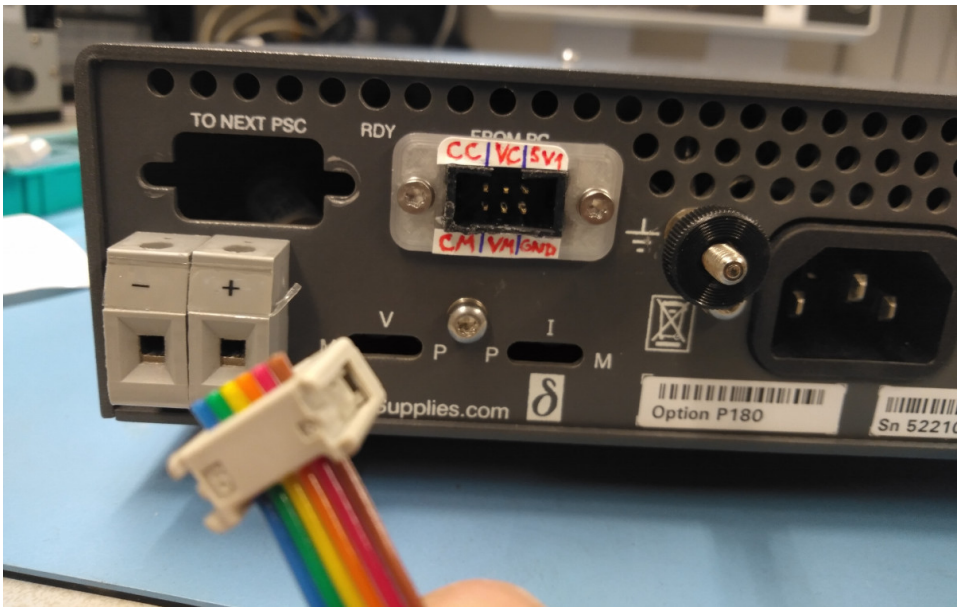


Figure E.18: Replaced the old RS-232 interface with a custom IDC connector interface.

Connector layout as depicted in Figure E.18:

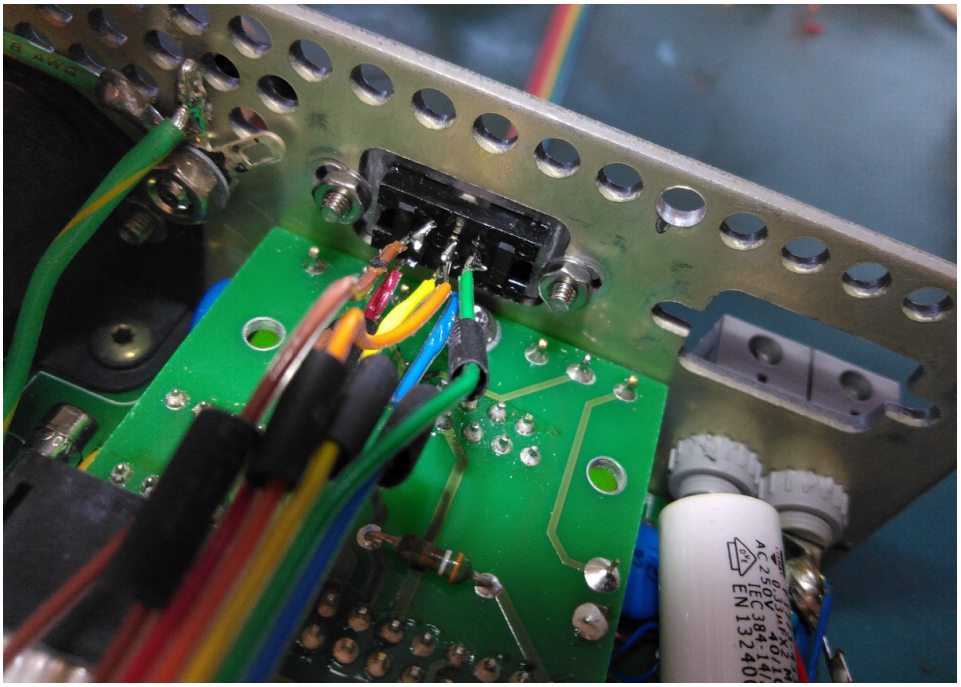


Figure E.19: Internal wiring of the IDC connector.

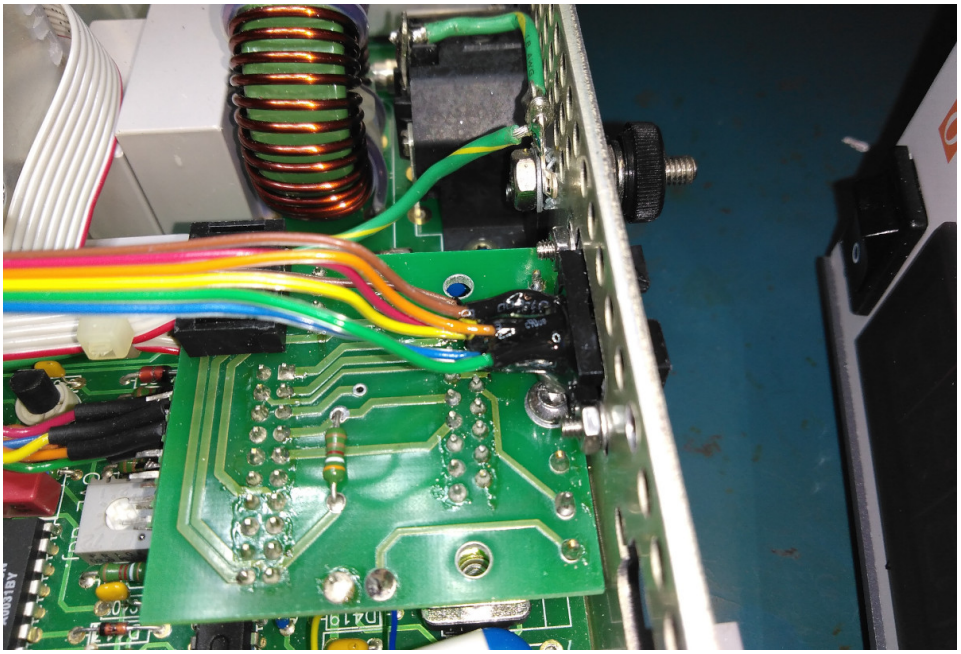


Figure E.20: Sealing the wire ends of the IDC connector with heatshrink wrapping and relieving mechanical strain on the connections by epoxy blobbing them. The epoxy is only connected to the connector and the coupler, so if needed it could be easily replaced.

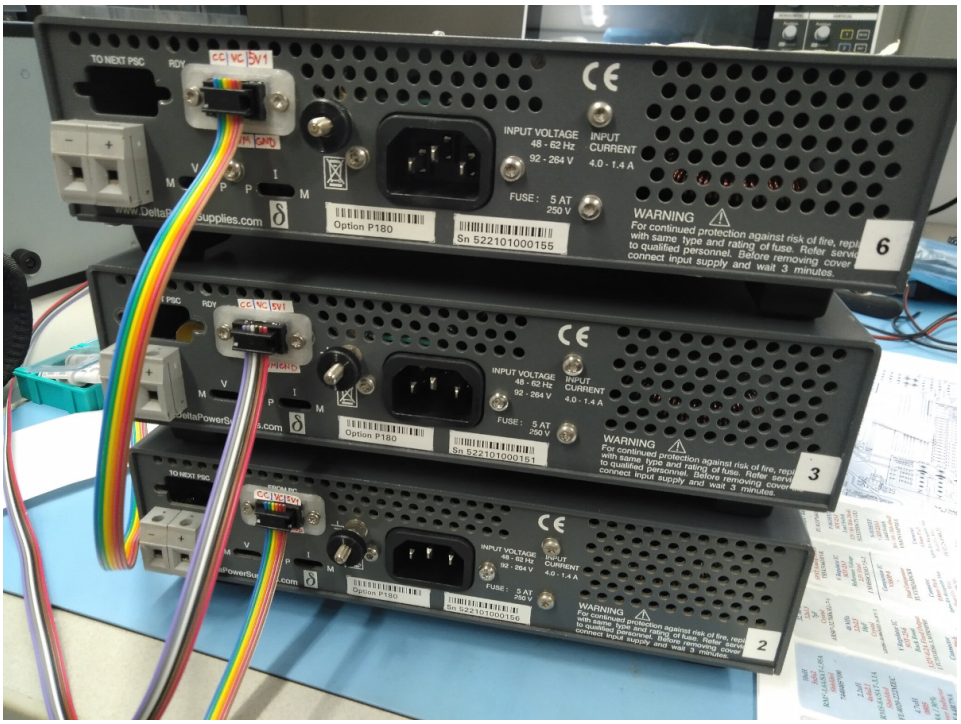


Figure E.21: All three power supplies modified to satisfaction.

F

Orbital Generator

As part of the Helmholtz Cage Toolkit (HCT) software, a basic orbit modeller was created. There was good reason for making this tool; the HCT should take a user-defined program of what magnetic field to play inside the Helmholtz cage over a given period of time. In the context of the HCT, these are called **schedules**. For human users, these schedules would be very laborious to make by hand, and so an interfacing tool was needed to automatically generate such a schedule in a way that is contextually **useful** to the user, and in a way that is **straightforward**, requiring as few user inputs as possible.

This approach is used in other places, such as in 3D printing, where a complex geometry is first parametrically defined into a 3D model, then converted by a slicer software tool into a long list of path coordinates for the printer head, for example in the form of G-code. The 3D printer can then interpret this G-code and use it to print the 3D object into reality. The point here is that the G-code is usually an file of hundreds of thousands of lines with coordinates, something that would be incredibly labour-intensive to define by hand. Even the exact geometry of the object to be defined is rather complex. However, the human user interacts simply with a convenient interface that allows for straightforward parametrization of the object. The rest of the toolchain then contains facilitating software that translate the human-understandable instructions into machine-understandable instructions. This makes the instructions resilient to human error, and also saves the user a vast amount of tedious work.

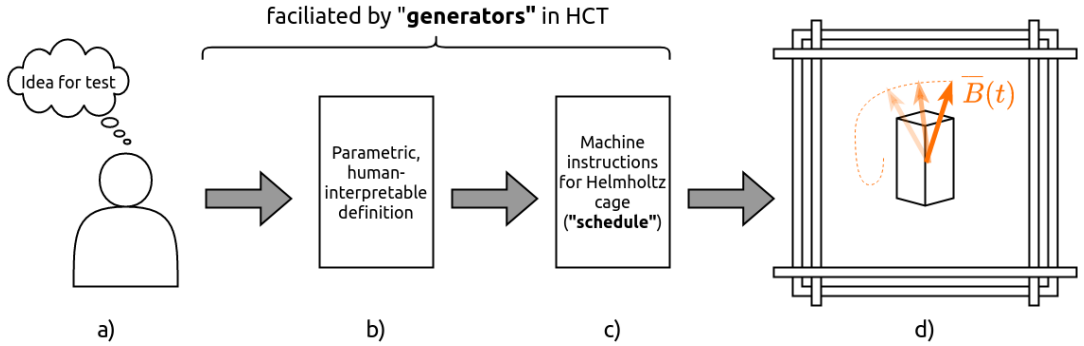


Figure F.1: Interaction interface between human user and Helmholtz cage system for the creation of magnetic field vector schedules. The generator scripts inside the HCT translates parametric, human-interpretable instructions to instructions that the Helmholtz cage system can use.

In the context of the HCT, a scripting tool that generates a schedule is called a **generator**. One such a generator is called the *Cyclics* generator, which is capable of generating a set of periodic functions to make cyclical vector paths inside the Helmholtz cage. This tool is very useful for the validation and verification of the system as a whole, as well as testing satellites in simple scenarios. However, a fully

computer-controlled Helmholtz cage is in theory capable of doing something much more powerful: simulating magnetic conditions during a satellite orbit, whilst the satellite under test remains immobile within the Helmholtz cage. To bestow this capability onto the Helmholtz cage setup, the HCT must include a way to model satellite orbits, compute the magnetic field conditions along this orbit at every point, and then translate this into a *schedule* that the Helmholtz cage can interpret. This tool will henceforth be referred to as the *Orbital* generator, which together with the Cyclics generator form the two main methods of generating schedules with the HCT.

F.1. Requirements

Using software to model satellite orbits is nothing new; there are many such tools on the market, some focusing on specific applications, some made for general-purpose orbit propagation. To determine the scope and purpose of the Orbitals generator inside the HCT, a number of guiding requirements were created:

1. **The generator shall be designed for simulating steady-state, elliptical low-Earth orbits.** This is because this covers most use-cases for the Helmholtz cage system to be designed, as most nanosatellites are in low Earth orbit (LEO). It also implies the need for a single magnetic model of the Earth, rather than requiring many magnetic models for each modelled body. Last, the magnetic effects that affect the performance of satellites are often very small in magnitude, but with the capacity to accumulate over time. This is why the steady state orbits are the main focus here, rather than one-off transfer orbits or hyperbolic trajectories.
2. **The generator shall be straightforward to use, without requiring detailed knowledge about orbital propagators.** The generator is not an end in and of itself, but rather a means to an end; this end being to generate instructions for the Helmholtz cage. It would therefore be best to keep the complexity and number of features of the generator to a minimum.
3. **The generator shall employ as few parameters as is feasible to generate the schedule.** The fewer variables are needed to fully parameterise the user instructions, the faster the user can work, and the easier they are to interpret by the user.
4. **The generator shall have both a graphical user interface (GUI), as well as a software application programming interface (API) for more advanced use.**
5. **The generator shall provide visual feedback to the user.** This makes the the tool much easier to understand by the user, and makes user error much easier to identify.

An obvious question at this point is to what extent the Orbitals generation has to be built from the ground up. In regards to orbital modelling, requirements 2 and 3 mostly rule out the use of 3rd party orbital propagators, such as GMAT or the TU Delft's own TUDAT. These propagators are too complex for simple use, as they have many more features than the generator strictly requires, and they typically require significant application-specific knowledge from the user. Furthermore, they are not exactly trivial to interface with the rest of the HCT. It was therefore decided early on that this part would be built from scratch.

The magnetic model is however a 3rd party model, as these models obviously require extensive research to make from scratch. Several magnetic field models have implementations in Python, and so these were much easier to interface with the HCT. The model that was chosen in the end is the 13th generation of the International Geomagnetic Reference Field (IGRF) [80], as it is up-to-date, easy to use, light on resources, and has a readily available Python implementation.

Requirement 4 is relatively easy to meet, so long as the generator is packaged into an easily readable Python class structure according to the standards of object-oriented programming. Requirement 5 has heavy implications for the layout of the GUI and may require some extra tools for visualising the orbits in 3D space. This will cost extra development time, but it will create a very efficient way to debug the generator code itself, as well as improving the user experience.

F.2. Functional Breakdown

In order to create the Orbitals generator in a more structured way, the functionality was broken down into more manageable tasks. This was useful to gain traction on the problem during implementation, as well as already providing a skeleton for the class structures of the actual code implementation. Figure F.2 depicts this breakdown in a diagram.

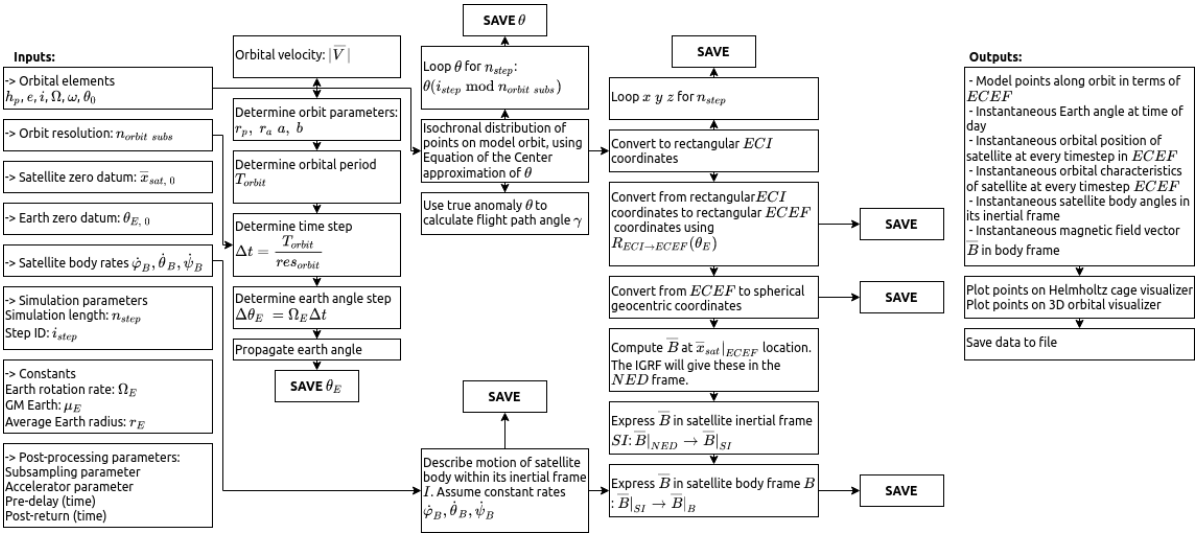


Figure F.2: Functional breakdown of the Orbitals generator code.

In simple terms, the Orbitals generator can be seen as a simple input-output machine. It takes a set of parameters relating to the motion of a certain satellite, as well as some extra settings, and turns them into a a description of the motion of the magnetic field vector inside the Helmholtz cage ($\mathbf{B}(t)|_{HHC}$). This assumes that the satellite under test will be immobile inside the Helmholtz cage at all times, or in other words, that the relationship between the satellite body frame and the Helmholtz cage reference frame is constant:

$$\frac{dR_{B \to HHC}}{dt} = \mathbf{0}$$

Therefore, the Orbitals generator has to output the dynamic magnetic field vector expressed in the body frame of the satellite ($\mathbf{B}(t)|_B$)

F.3. Functional Implementation

So how can $\mathbf{B}(t)|_B$ be generated? With Figure F.2 as reference, each step will now be explained. Before doing so, it is important to give an overview of the various reference frames involved, because otherwise things will get confusing very quickly. There are a total of six references frames used in the following sections:

Table F.1: Overview of reference frames used in the following sections.

B	Body reference frame attached to the simulated satellite
$ECEF$	Earth-centred, Earth-fixed reference frame
ECI	Earth-centred inertial reference frame
HHC	Reference frame of the Helmholtz cage
NED	Local North-East-Down coordinate frame
SI	Inertial reference frame of the simulated satellite in its orbit

These can also be found in Figure F.3.

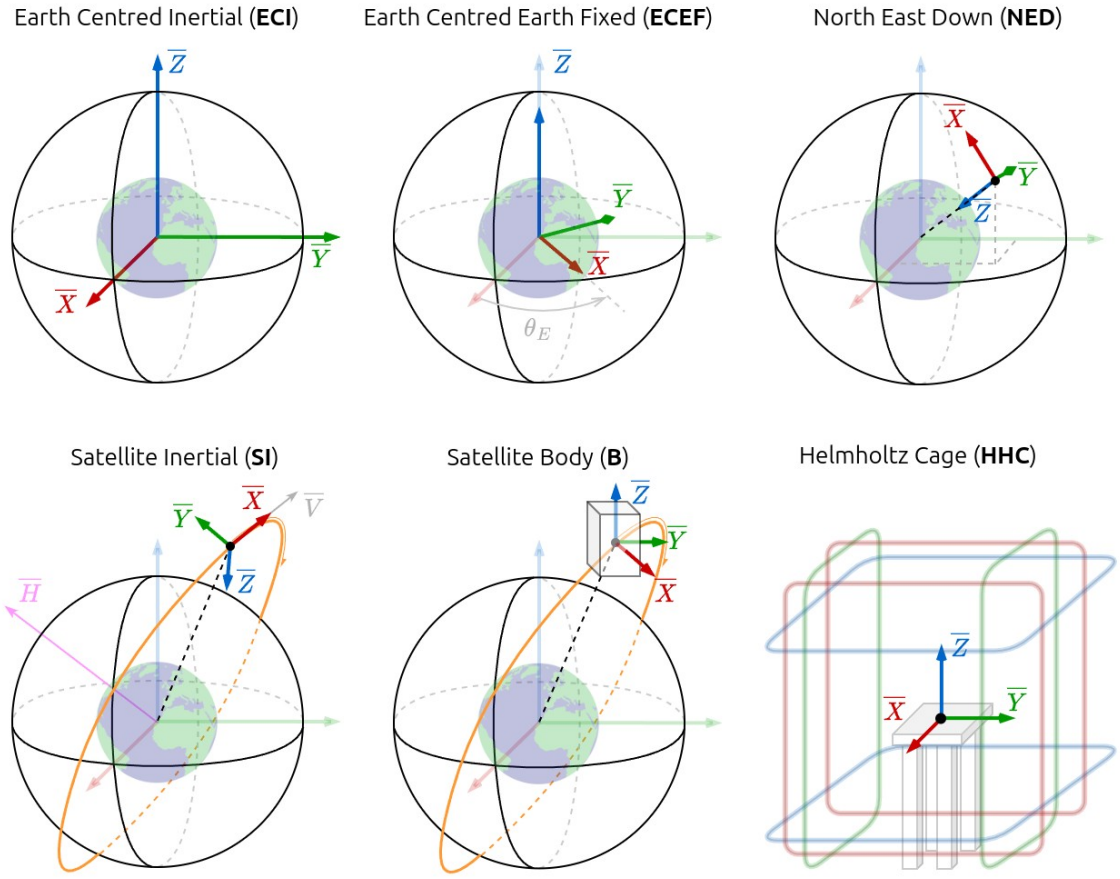


Figure F.3: Reference frames used in this thesis.

Why are there so many reference frames used? First, the Earth magnetic field is generated by the contents inside Earth's core, it will rotate along with the Earth itself during its daily rotation. As such, there is a need to distinguish between the $ECEF$ and ECI, such that the rotation of the Earth can be taken into account. Second, it is necessary to distinguish between the B and SI frames to give the simulated satellite the capacity to be unaligned within its inertial frame, or even rotating with respect to it. Without this ability, one could only simulate satellites perfectly aligned with their inertial reference frames, which is not very realistic. Finally, there is a need to use the NED because this is the reference frame in which the IGRF outputs its magnetic field vectors. The NED frame is closely related to the more common ENU frame, and the relationship between the two is relatively simple:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{ENU} = \begin{pmatrix} y \\ x \\ -z \end{pmatrix}_{NED} \quad \text{with} \quad R_{ENU \to NED} = R_{NED \to ENU} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

The ENU frame will not be discussed further as it will be possible to convert between NED and $ECEF$ directly.

For the purposes of generating instructions for the Helmholtz cage, it is important to express everything in terms of frame B. However, for the purposes of providing user feedback in the form of a 3D visualisation, everything should also be described in terms of the ECI frame, as this frame will coincide with the axis system used in that plot. This is why it is important to keep an inventory of which parameter is expressed in which frames, and how to transform between each reference frame.

F.3.1. Input parameters

Table F.2: Input parameters of the Orbitals generator

Symbol	Unit	Value	Parameters	Set
e	[-]	$[0, 1\rangle$	Orbital eccentricity	Orbital elements
i	[°]	[0, 180]	Orbital inclination	Orbital elements
h_p	[km]	[0, 1000]	Altitude of pericentre	Orbital elements
Ω	[°]	[0, 360]	Right ascension of the ascending node (RAAN)	Orbital elements
ω	[°]	[0, 360]	Argument of periapsis	Orbital elements
θ_0	[°]	[0, 360]	Initial mean anomaly	Orbital elements
$\theta_{E,0}$	[°]	[0, 360]	Earth zero datum	-
$\varphi_{B,0}$	[°]	-	Initial body position angle around X_{SI}	Body configuration
$\theta_{B,0}$	[°]	-	Initial body position angle around Y_{SI}	Body configuration
$\psi_{B,0}$	[°]	-	Initial body position angle around Z_{SI}	Body configuration
$\dot{\varphi}_B$	$[°/s]$	-	X_{SI} rotation rate	Body configuration
$\dot{\theta}_B$	$[°/s]$	-	Y_{SI} rotation rate	Body configuration
$\dot{\psi}_B$	$[°/s]$	-	Z_{SI} rotation rate	Body configuration
$year$	[-]	-	Year of orbital simulation, expressed as decimal number	-
n_{orbit_subs}	[-]	-	Number of subdivisions for one orbit	Simulation settings
n_{steps}	[-]	-	Number of simulation steps	Simulation settings
f_{interp}	[-]	-	Interpolation factor	Simulation settings

Table F.3: Various constants

Symbol	Unit	Value	Parameters	Source
Ω_E	$[rad/s]$	$7.292115 \cdot 10^{-5}$	Mean angular rate of Earth	WGS84 [7]
μ_E	$[m^3/s^2]$	$3.98600442 \cdot 10^{14}$	Geocentric gravitational constant of Earth	WGS84 [7]
r_E	$[km]$	6378.1370	Mean Earth radius	WGS84 [7]

F.3.2. Determining orbital characteristics

First, some basic Keplerian properties of the elliptical orbit can be derived. These are listed below, and all are sourced from Wakker [81]:

1. Radius of periapsis:
$$r_p = r_E + h_p \tag{F.1}$$
2. Radius of apoapsis:
$$r_a = \frac{1+e}{1-e} r_p \tag{F.2}$$
3. Semi-major axis:
$$a = (r_p + r_a)/2 \tag{F.3}$$
4. Semi-minor axis:
$$b = \sqrt{r_p r_a} \tag{F.4}$$
5. Orbital period:
$$T_{orbit} = 2\pi \sqrt{\frac{a^3}{\mu_E}} \tag{F.5}$$
6. Instantaneous absolute orbital velocity:
$$|\mathbf{V}| = \sqrt{\mu_E \left(\frac{2}{r} - \frac{1}{a}\right)} \tag{F.6}$$

This defines the ellipse model of the simulated orbit.

F.3.3. Describing orbital motion

For simulation purposes, ellipse model of the orbit should be divided up into points, such that the instantaneous orbital properties can be calculated at every point. There are two main ways in which this can be done:

1. **Equidistant spacing**: The orbit is subdivided up into sections of equal path length.
2. **Isochronal spacing**: The orbit is subdivided up into sections of equal travel time.

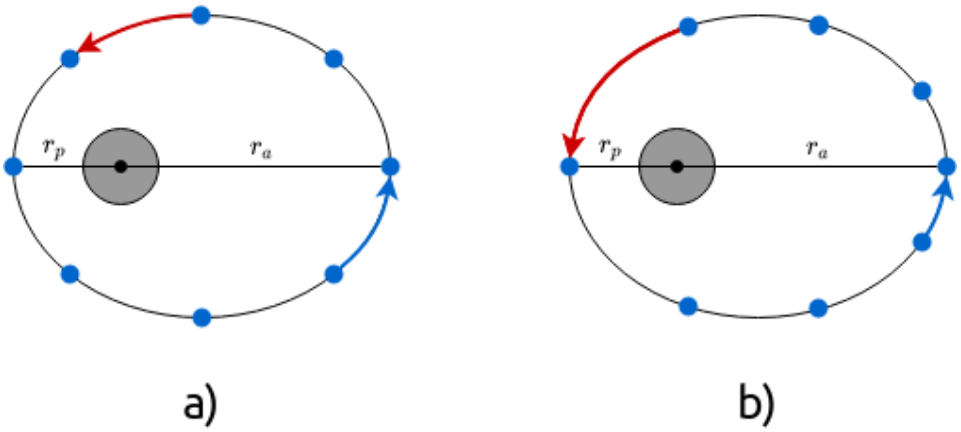


Figure F.4: (a) Equidistant spacing - The travel distance between each point is equal (b) Isochronal spacing - The orbital transit time between each point is equal.

Generating an equidistant spacing is quite straightforward, as it is merely a geometrical problem. However, dividing up the orbit into isochronal segments is much more useful for simulation purposes, because it allows for the use of a constant Δt throughout the whole process. It is however less straightforward to implement this, because the problem involves relating the true anomaly ν to the mean anomaly M at every point in the orbit. As an intermediary step, the true anomaly can be related to the eccentric anomaly E using [81]:

$$\tan\left(\frac{\nu}{2}\right) = \sqrt{\frac{1+e}{1-e}}\tan\left(\frac{E}{2}\right) \tag{F.7}$$

The eccentric anomaly E and the mean anomaly M are related through [81]:

$$M = E - e\sin E \tag{F.8}$$

Both these equations have to be solved for ν, but no closed-form expression for this exists. However, for orbits with low eccentricity, a sufficiently accurate estimation can be obtained by using the series expansion of the Equation of the Centre [82]:

$$\nu = M + 2\sum_{s=1}^{\infty}\frac{1}{s}\left(J_s(se) + \sum_{p=1}^{\infty}\beta^p\left[J_{s-p}(se) + J_{s+p}(se)\right]\right)\sin(sM) \tag{F.9}$$

with β being a constant dependent solely on the eccentricity:

$$\beta = \frac{1}{e}\left(1 - \sqrt{1-e^2}\right) \tag{F.10}$$

This form uses Bessel function terms J_s to shape the mean anomaly ever closer to the eccentric anomaly. Increasing the order s of the method yields a more accurate answer, at the cost of increased computation

time, since more expansion terms are taken into account. From Figure F.5 can be clearly seen that eccentricity values above 0.5 start to show notable distortion. If these were used as input for the orbit model, the simulated satellite would bounce forward and backwards along certain segments of the orbit, which is obviously highly undesirable. However, for the application at hand this is acceptable, as there are very few nanosatellite missions in LEO whose orbital eccentricity exceeds 0.5. So the limitation is accepted in this case, with the caveat to the user that high values of e will degrade performance.

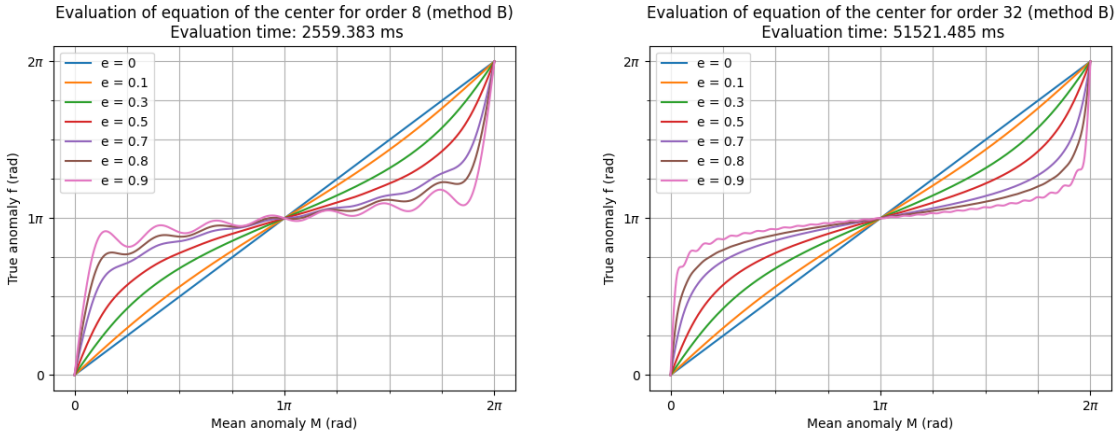


Figure F.5: Performance of the Equation of the Centre approximation for various values of the eccentricity. Note the increase in error for high values of e. Increasing the order of the method yields better result (compare left and right) at a higher computation cost (compare the evaluation time).

A code implementation of the method can be found below (Python).

```
from scipy.special import jv

def equation_of_the_center(self, M, e, order=12):
    # General expression in terms of Bessel functions of the first kind
    v = M
    if 0 < e < 1:  # If orbit is circular or hyperbolic, don't bother
        b = 1 / e * (1 - (1 - e * e) ** 0.5)
        for s in range(1, order):
            bt = 0
            for p in range(1, order):
                bt += b**p*(jv(s-p, s*e)+jv(s+p, s*e))
            v += 2/s * (jv(s, s*e) + bt)*sin(s*M)
    return v
```

The goal of this process was to divide a model orbit up into isochronal segments, which can now be done straightforwardly. Let the number of points on this model orbit be n_{orbit_subs}. Together with Equation F.5, this yields the value of Δt corresponding to each segment, which is constant.

$$\Delta t = \frac{T_{orbit}}{n_{orbit_subs}} \tag{F.11}$$

The last remaining properties that are important to know are the velocity components, as the inertial reference frame of the satellite depends on it. To this end, first the flight path angle of the simulated spacecraft along its orbit will be calculated. This is easy to calculate given that the true anomaly has been estimated [81]:

$$\gamma = \arctan \frac{e \sin \nu}{1 + e \cos \nu} \tag{F.12}$$

The absolute value of the velocity vector along the orbit is already known by virtue of Equation F.6. But the direction of the vector must still be expressed in terms of the Cartesian coordinates of the rest of the orbit. At the pericentre, the velocity vector is a vector of length $|\mathbf{V}|$ that points up purely into the Y

direction. As one moves along the orbit, this vector will be rotated by a combination of the true anomaly and the flight path angle. It can be shown that the velocity vector at every point can be described by:

$$\mathbf{V} = \begin{bmatrix} \cos(\nu - \gamma) & -\sin(\nu - \gamma) \\ \sin(\nu - \gamma) & \cos(\nu - \gamma) \end{bmatrix} \cdot \begin{pmatrix} 0 \\ |\mathbf{V}| \end{pmatrix} = \begin{pmatrix} -\sin(\nu - \gamma) \\ \cos(\nu - \gamma) \end{pmatrix} \cdot |\mathbf{V}| \tag{F.13}$$

The astute reader may have noticed that so far, the orbit model has been a purely 2D representation. The Earth magnetic field is a three-dimensional object, and so this will not do. To transform a 2D coordinate $(X,\ Y)$ of the model orbit into a 3D coordinate $(x,\ y,\ z)$, all that is needed is a few orbital elements and the following transformation [81]:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \cdot \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix} \tag{F.14}$$

with

$$R = \begin{bmatrix} \cos\omega\cos\Omega - \sin\omega\sin\Omega\cos i & -\sin\omega\cos\Omega - \cos\omega\sin\Omega\cos i & \sin\Omega\sin i \\ \cos\omega\sin\Omega + \sin\omega\cos\Omega\cos i & -\sin\omega\sin\Omega + \cos\omega\cos\Omega\cos i & -\cos\Omega\sin i \\ \sin\omega\sin i & \cos\omega\sin i & \cos i \end{bmatrix} \tag{F.15}$$

The velocity vectors can be similarly transformed into 3D vectors.

F.3.4. Generating magnetic field data

Up to this point, a list of 3D coordinates has been generated, representing points on the model orbit. The points are spaced isochronally, such that when moving between them at a constant rate, the actual velocity of a simulated orbiting body is described. If it is assumed that the orbit is stable, the simulation can simply loop this list of coordinates, and when it jumps back to the start, the next lap of the orbit is described.

These coordinates are all represented in the same inertial reference frame. Which reference frame is this? It is not the $ECEF$ (see Table F.1), because the orbit that a satellite makes around Earth is not fixed to the body of the Earth, but rather to its gravity well. Instead, it should be the ECI reference frame, which is Earth-centred but *not* Earth-fixed. However, the Earth's magnetic field *is* fixed to the Earth, and so this should be described in some Earth-fixed reference frame, such as the $ECEF$. As a result, two different laps of the same orbital ellipse will not necessarily yield the same magnetic data, as the Earth is rotating "underneath" the path of the orbit. From the perspective of the satellite, it will be moving through a different region of the Earth magnetic field compared to the last lap. To capture this behaviour, the Orbital generator should be able to distinguish between these two reference frames and include a description of the motion of Earth.

To transform between the ECI frame and the $ECEF$, the following rotation matrix can be used:

$$R_{ECI \to ECEF} = R_Z(\theta_t) = \begin{bmatrix} \cos\theta_t & -\sin\theta_t & 0 \\ \sin\theta_t & \cos\theta_t & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{with} \quad \theta_t = \Omega_E \cdot \Delta t \tag{F.16}$$

As mentioned previously, the IGRF model will be used to generate the magnetic field data. This model is readily available as the Python package `PyIGRF` and can therefore be easily invoked by the Orbital generator. As input, the model requires a 3D coordinate for the position, expressed in longitude, latitude, and altitude (obviously expressed in $ECEF$). This is a spherical coordinate system. Furthermore, it requires one temporal coordinate, expressed as a decimal date[1]. Since the points of the model orbit have thus far been expressed as Cartesian coordinates in the ECI frame, a way to transform between

[1] A decimal date is a date expressed as a decimal number in years. For example, April 19th 1824 would be expressed as 1824.297814208

these two frames is required. The spherical representation of a Cartesian coordinate in the ECI frame is calculated using:

- Radius r:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{F.17}$$

- Geocentric longitude λ:

$$\lambda = \frac{y}{|y|} \cdot \arccos \frac{x}{\sqrt{x^2 + y^2}} \tag{F.18}$$

- Geocentric latitude ϕ:

$$\phi = \frac{\pi}{2} - \arccos \frac{z}{r} \tag{F.19}$$

Note that when r is zero, the longitude and latitude are not defined. Furthermore, at ±90° longitude, the latitude is no longer defined. The former exception is not expected to occur as this location is at the centre of the Earth, not exactly a place that is frequented by satellites. However, the latter exception is fairly common, especially for orbits with a 90° inclination. To address this, the following addendum is made to the previous equations:

$$\begin{bmatrix} r \\ \lambda \\ \phi \end{bmatrix} = \begin{bmatrix} |z| \\ 0 \\ \frac{z}{|z|} \cdot \frac{\pi}{2} \end{bmatrix} \quad \text{if} \quad x,\ y = 0 \tag{F.20}$$

Furthermore, note that the IGRF calls for the altitude h rather than the radius r. The two differ by the value of the radius of the Earth. For this simulation the oblateness of the Earth is neglected, which makes r_E a constant and simplifies matters substantially. The altitude is then:

$$h = r - r_E \tag{F.21}$$

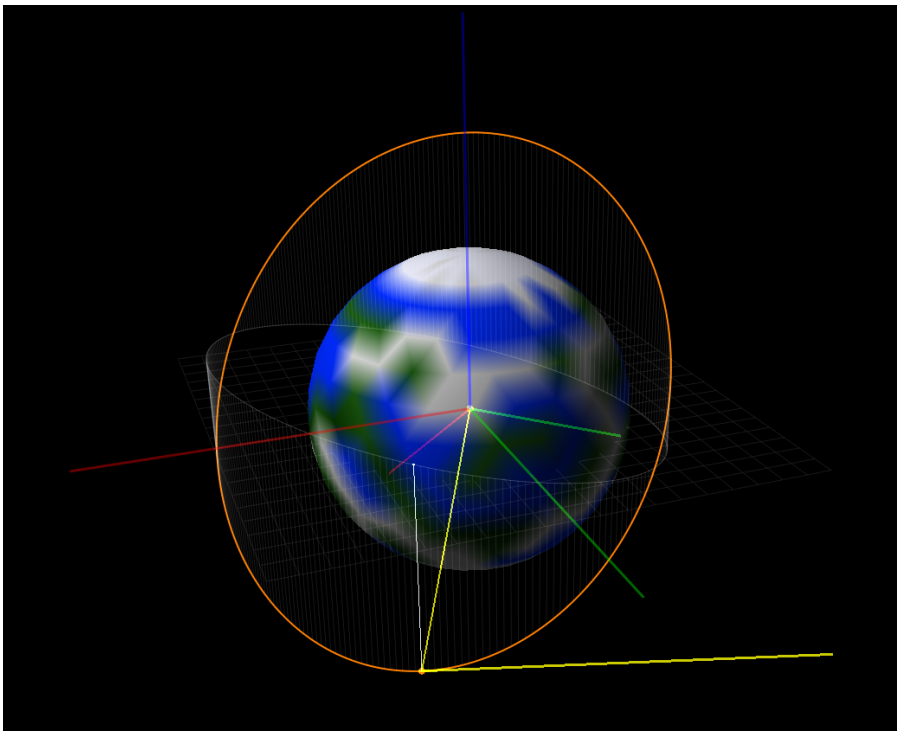


Figure F.6: Orbit simulation using the discussed tools. In orange, a highly inclined orbit has been plotted in the ECI frame (large tripod). The $ECEF$ frame (small tripod) rotates as Earth rotates over time. The yellow line from the centre of the Earth to the highlighted point is radius r. The other large yellow line indicates the instantaneous velocity vector of the satellite at the given time step.

Usually longitude and latitude refer to mappings of Earth's surface, which in this case means the spherical coordinate in the $ECEF$ frame, rather than the ECI frame. Fortunately, converting between the two is very straightforward as their difference is just the angle that the Earth makes with respect to the ECI frame:

$$\begin{bmatrix} r \\ \lambda \\ \phi \end{bmatrix}\Bigg|_{ECEF} = \begin{bmatrix} r \\ \lambda \\ \phi \end{bmatrix}\Bigg|_{ECI} - \begin{bmatrix} 0 \\ \theta_t \bmod 2\pi \\ 0 \end{bmatrix} \tag{F.22}$$

The result of the progress so far can be see in Figure F.6.

F.3.5. Transforming from NED frame and other reference frames

The IGRF model outputs three Cartesian components of a vector **B** that represents the local magnetic field vector at the time and location that was provided. However, this vector is represented in a local NED frame, whose origin coincides with the location where the field data was generated. To relate the magnetic field data to the broader picture (and to be able to plot it in Figure F.6), a transformation between the local NED frame and the ECI frame is needed.

Converting from the $ECEF$ frame to a global NED frame is done by two successive rotations: once around the Z_{ECEF} axis and then around the intermediate Y' axis. The angles involved in these rotations relate to the geocentric longitude λ and latitude ϕ.

The rotation matrix $R_{ECEF \to NED}(\lambda,\ \phi)$ can be constructed as follows:

$$\begin{aligned} R_{ECEF\to NED}(\lambda,\ \phi) &= R_Y\left(-\left(\frac{\pi}{2}-\phi\right)\right)\cdot R_Z(-\lambda) \\ &= \begin{bmatrix} \cos\left(-\left(\frac{\pi}{2}-\phi\right)\right) & 0 & \sin\left(-\left(\frac{\pi}{2}-\phi\right)\right) \\ 0 & 1 & 0 \\ -\sin\left(-\left(\frac{\pi}{2}-\phi\right)\right) & 0 & \cos\left(-\left(\frac{\pi}{2}-\phi\right)\right) \end{bmatrix} \cdot \begin{bmatrix} \cos(-\lambda) & -\sin(-\lambda) & 0 \\ \sin(-\lambda) & \cos(-\lambda) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{F.23}$$

Using the following trigonometric identities:

$$\begin{aligned} \cos(-\alpha) &= \cos(\alpha) \\ \sin(-\alpha) &= -\sin(\alpha) \\ \sin(\alpha) &= \cos\left(\frac{\pi}{2}-\alpha\right) \\ \cos(\alpha) &= \sin\left(\frac{\pi}{2}-\alpha\right) \end{aligned} \tag{F.24}$$

this expression can be simplified:

$$\begin{aligned} R_{ECEF\to NED}(\lambda,\ \phi) &= \begin{bmatrix} \sin(-\phi) & 0 & \cos(-\phi) \\ 0 & 1 & 0 \\ -\cos(-\phi) & 0 & \sin(-\phi) \end{bmatrix} \cdot \begin{bmatrix} \cos(-\lambda) & -\sin(-\lambda) & 0 \\ \sin(-\lambda) & \cos(-\lambda) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} -\sin(\phi) & 0 & \cos(\phi) \\ 0 & 1 & 0 \\ -\cos(\phi) & 0 & -\sin(\phi) \end{bmatrix} \cdot \begin{bmatrix} \cos(-\lambda) & -\sin(-\lambda) & 0 \\ \sin(-\lambda) & \cos(-\lambda) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &\qquad \begin{bmatrix} -\sin\phi\cos\lambda & -\sin\phi\sin\lambda & \cos\phi \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\phi\cos\lambda & -\cos\phi\sin\lambda & -\sin\phi \end{bmatrix} \end{aligned} \tag{F.25}$$

To go in the opposite direction, one needs the inverse of the matrix from Equation F.25. The inverse of any real-numbered rotation matrix is equal to its transpose, and so:

$$R_{NED\to ECEF} = R^{-1}_{ECEF\to NED} = R^{T}_{ECEF\to NED} = \begin{bmatrix} -\sin\phi\cos\lambda & -\sin\lambda & -\cos\phi\cos\lambda \\ -\sin\phi\sin\lambda & \cos\lambda & -\cos\phi\sin\lambda \\ \cos\phi & 0 & -\sin\phi \end{bmatrix} \tag{F.26}$$

However, rather than the global NED frame, it would be interesting to know how to transform from the $ECEF$ frame to a *local* NED frame, and so any required translation of the origin must be taken into account:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{NED} = R_{ECEF\to NED} \cdot (\mathbf{O}_{ECEF} - \mathbf{O}_{NED}) \,|_{ECEF} + R_{ECEF\to NED} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{ECEF} \tag{F.27}$$

The origin of the $ECEF$ frame expressed in its own frame is zero by definition:

$$\mathbf{O}_{ECEF} \,|_{ECEF} = \mathbf{0} \tag{F.28}$$

Further, because of the distributive property of the dot product, we can simplify the expression to:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{NED} = R_{ECEF\to NED} \cdot \left[\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{ECEF} - \mathbf{O}_{NED} \,|_{ECEF} \right] \tag{F.29}$$

where $\mathbf{O}_{NED}|_{ECEF}$ is a vector pointing from the $ECEF$ origin to the NED origin expressed in terms of the $ECEF$.

In the interest of converting to ECI rather than $ECEF$, what has to be adjusted to Equation F.26 in order to facilitate this? It turns out that nothing has to be adjusted at all, and this is relatively easy to intuit: ECI and $ECEF$ have coinciding origins, which means that $\mathbf{z}_{NED}$ axis points to the exact same point, and the origin shift from the NED frame is also equivalent. Furthermore, the principal difference between ECI and $ECEF$ is that $ECEF$ rotates along with the Earth. But the definitions of "North" and "East" remain the same whether the Earth is rotating or not. Thus it can be posed that:

$$R_{NED\to ECEF} = R_{NED\to ECI}$$

To transform an arbitrary magnetic field vector $\mathbf{B}$ from the NED frame to ECI, the following transformation can be used:

$$\begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}\Bigg|_{ECI} = \begin{bmatrix} -\sin\phi\cos\lambda & -\sin\lambda & -\cos\phi\cos\lambda \\ -\sin\phi\sin\lambda & \cos\lambda & -\cos\phi\sin\lambda \\ \cos\phi & 0 & -\sin\phi \end{bmatrix} \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}\Bigg|_{NED} + \begin{pmatrix} x \\ y \\ z \end{pmatrix}\Bigg|_{ECI} \tag{F.30}$$

where first term on the right-hand side of Equation F.30 rotates the reference frame, and the second term shifts the origin. As can be seen in Figure F.8, this allows the determination of the magnetic field at any point in the 3D space spanned by the ECI frame.

Example - Transforming between *ECEF* and *NED*

To illustrate the previous discussion on frame transformations, let's consider the axes system in Figure F.7. The large axes system is the *ECEF* frame. We can transform to a local *NED* frame at any point, and several such systems are drawn in the figure. The geocentric longitude λ is the angle that is formed by rotating positively around the Z_{ECEF} axis, from the positive X_{ECEF} axis. The geocentric latitude ϕ is the angle that is formed by rotating negatively around the Y_{ECEF} axis, from the positive X_{ECEF} axis.
Imagine we place an *NED* frame at the point $(0, 2, 0)$. The point $(0, 2, 0)$ is O_{NED}, the location of the origin of the *NED* frame expressed in the *ECEF* frame. At this point, the longitude λ is +90° and the latitude ϕ is 0°.
Imagine we now have a point P, whose coordinates in the *ECEF* frame are $(1, 1, -1$, and that we wish to find its coordinates in the *NED* frame instead. Following the transformation described earlier, we get:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{NED} = R_{ECEF \to NED}(\lambda,\ \phi) \cdot \left[\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{ECEF} - \mathbf{O}_{NED}\,|_{ECEF} \right] \tag{F.31}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{NED} = \begin{bmatrix} -\sin\phi\cos\lambda & -\sin\phi\sin\lambda & \cos\phi \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\phi\cos\lambda & -\cos\phi\sin\lambda & -\sin\phi \end{bmatrix} \cdot \left[\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{ECEF} - \mathbf{O}_{NED}\,|_{ECEF} \right] \tag{F.32}$$

Plugging in the values, we get:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{NED} = \begin{bmatrix} -\sin 0\cos\frac{\pi}{2} & -\sin 0\sin\frac{\pi}{2} & \cos 0 \\ -\sin\frac{\pi}{2} & \cos\frac{\pi}{2} & 0 \\ -\cos 0\cos\frac{\pi}{2} & -\cos 0\sin\frac{\pi}{2} & -\sin 0 \end{bmatrix} \cdot \left[\begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} - \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix} \right] \tag{F.33}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{NED} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \cdot \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} \tag{F.34}$$

which can be verified in Figure F.7.

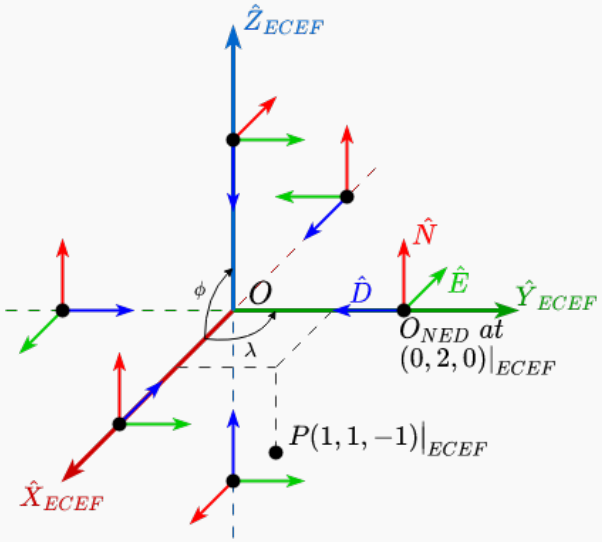


Figure F.7: Various local *NED* frames inside a global *ECEF* frame. Point P can be expressed in the global *ECEF* frame with coordinate $P(1, 1, -1)\,|_{ECEF}$. It can also be expressed in the local *NED* frame at $(1, 1, -1)\,|_{ECEF}$, where it now has coordinate $(-1, -1, 1)\,|_{NED}$

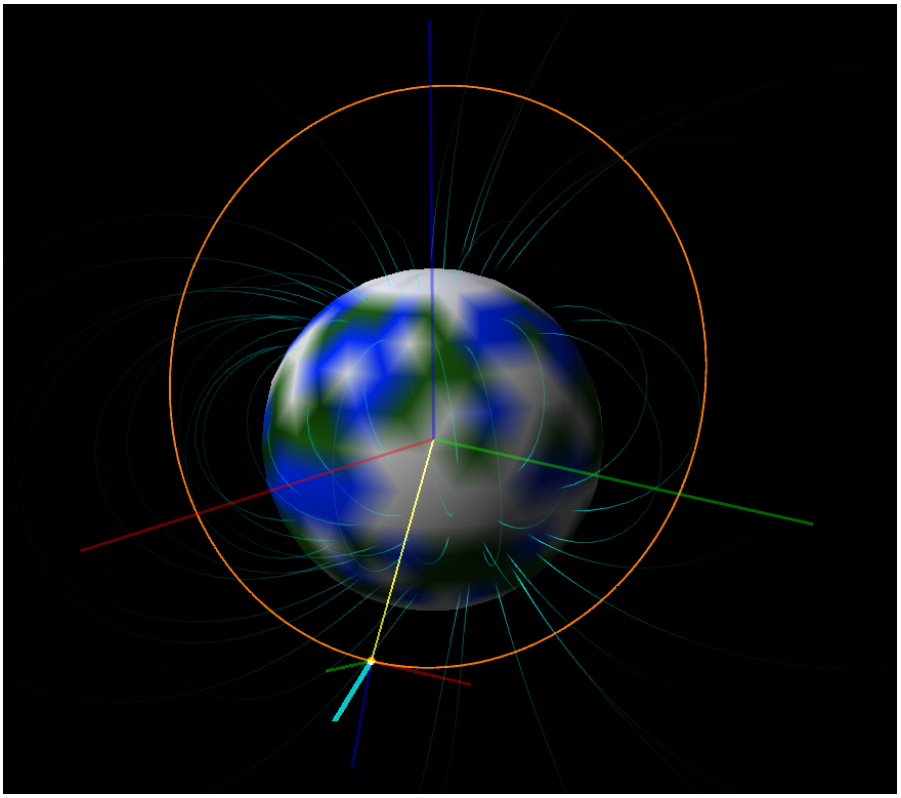


Figure F.8: Integration of the IGRF model. A magnetic field vector (blue) is measured at the location of the satellite, after which it is transformed from the NED frame (small tripod) to the general ECI frame. A number of IGRF-based field lines are also plotted in blue.

F.3.6. Transforming to the satellite body reference frame

As mentioned before, plotting everything in terms of the ECI frame is useful, but only of secondary interest. Primarily, the interest is in transforming everything to the body frame B of the satellite, as this will lead to the magnetic field vectors that should be generated by the Helmholtz cage. Therefore, a few steps remain. A transformation relating the inertial reference frame of the satellite (SI) with the ECI must be found. Then, the relationship between the SI frame and the B frame should be found, which is formally the attitude of the satellite and may vary over time.

First, the SI frame is defined with its X-component aligned with the instantaneous velocity vector of the satellite, which is always in the orbital plane. Its Y-component is aligned with the angular momentum unit vector $\mathbf{H}$ of the orbit, which is always perpendicular to the orbital plane. The Z-component is formed by taking the cross product of the two.

The unit vector of the angular momentum vector has not been discussed yet, but can be constructed early on along with the definition of the 2D orbital ellipse. The angular momentum vector lies perpendicular to the orbital plane, and so before transforming the orbit into 3D, this is simply a unit vector pointing in the Z-direction. This unit vector is then transformed with the same matrix R as the rest of the orbit, and so:

$$\mathbf{H} = R \cdot \mathbf{z} = R \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \sin\Omega \sin i \\ -\cos\Omega \sin i \\ \cos i \end{pmatrix} \tag{F.35}$$

This now allows for a definition of the transformation matrix that transforms between ECI and SI. Its

first row consists of the unit vector components of the instantaneous velocity vector:

$$\mathbf{V} = \frac{1}{|\mathbf{V}|} \cdot \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} \tag{F.36}$$

The second row consists of the angular momentum unit vector components, and the third row consists of the cross product of the two:

$$R_{ECI \to SI} = \begin{bmatrix} \mathbf{V}^T \\ \mathbf{H}^T \\ (\mathbf{V} \times \mathbf{H})^T \end{bmatrix} \tag{F.37}$$

Finally, the satellite inertial reference frame SI must be related to the satellite body frame B. To keep things simple, the Orbitals generator supports only constant rotation rates around any of the three body axes of the satellite:

$$\begin{pmatrix} \varphi_B \\ \theta_B \\ \psi_B \end{pmatrix}(t) = \begin{pmatrix} \dot{\varphi}_B \\ \dot{\theta}_B \\ \dot{\psi}_B \end{pmatrix}(t - t_0) + \begin{pmatrix} \varphi_{B,\,0} \\ \theta_{B,\,0} \\ \psi_{B,\,0} \end{pmatrix} \tag{F.38}$$

To also include rotational acceleration would complicate the internal equations of motion substantially, and in the current iteration of development, there seem to be few advantages to implementing this. Let φ_B, θ_B, and ψ_B be the rotation angles of the B frame with respect to the SI frame. Then any vector $\mathbf{X}$ can be transformed between frames using three Euler angle rotations:

$$\mathbf{X}|_B = R_{SI \to B}(\varphi_B, \theta_B, \psi_B) \cdot \mathbf{X}|_{SI} = R_X(\varphi_B) \cdot R_Y(\theta_B) \cdot R_Z(\psi_B) \cdot \mathbf{X}|_{SI} \tag{F.39}$$

Working this out yields:

$$R_{SI \to B} = \begin{bmatrix} \cos\theta_B \cos\psi_B & \cos\theta_B \sin\psi_B & -\sin\theta_B \\ -\cos\varphi_B \sin\psi_B - \sin\varphi_B \sin\theta_B \cos\psi_B & \cos\varphi_B \cos\psi_B - \sin\varphi_B \sin\theta_B \sin\psi_B & -\sin\varphi_B \cos\theta_B \\ -\sin\varphi_B \sin\psi_B + \cos\varphi_B \sin\theta_B \cos\psi_B & \sin\varphi_B \cos\psi_B + \cos\varphi_B \sin\theta_B \sin\psi_B & \cos\varphi_B \cos\theta_B \end{bmatrix} \tag{F.40}$$

G

Server software and TCP codec

In the main text of this thesis, there was not enough space to go into fine detail about the server firmware running on the Raspberry Pi inside the Demonstrator System (DS). This appendix describes it in more detail.

As mentioned in chapter 11, this is a program with several purpose-specific threads that share a common data pool. There are five threads in total:

1. **Main thread**: This is the main thread that runs the Python program. It is the first to be created, and the last one to fold.
2. **Socket server thread**: This thread facilitates communication over the transmission control protocol (TCP) connection to the high-level interface.
3. **Control thread**: Used for managing, staging, and playing schedules.
4. **ADC thread**: Acquires data from the analog-to-digital converter (ADC) and writes it to the datapool.
5. **DAC thread**: Instructs the digital-to-analog converter (DAC) given a set of operational rules and information from the datapool.

The main reason for setting up the server application is related to performance and organizational advantages. The Raspberry Pi has limited computational resources, but it does support threading. With the threading set up this way, separate functionalities implemented by the code can be run at separate loop rates without much unnecessary overhead.

For example: one of the threads polls the ADC hardware for current measurement values. Not exactly every acquired datapoint has to be sent over the TCP connection; it only has to do so when a request is sent for it over the TCP connection. This way, the ADC thread could be run at higher loop rates, for example to get more accurate data by filtering the data using a moving average, without having to run the socket code for the TCP connection unnecessarily frequently. To be fair, this could also be implemented without threading, by instead putting everything in a high frequency Digital Signal Processing (DSP) loop optimized with conditionals, but then every action happens at a given division of the DSP loop rate. Although the latter is not explicitly an important disadvantage, threading here seems to provide a more true compartmentalization between the functionalities, so it was preferred for this application.

Variables in the datapool that can be accessed by multiple threads are generally protected by *locks*, a concept from the Python library `threading`. These locks function similar to the concept of a *mutex*, which prevents a thread from accessing the variable if it is already being accessed by another thread. It is important that locks are used, because failing to do so can result in threading-related race conditions, which can be particularly stochastic and difficult to debug. Due to the way that the "threads" from the `threading` Python library are run by the processor of the Raspberry Pi, they probably are not needed,

but it does not hurt and ensures that the baseline code will not break if the code is implemented using hardware multi-threading later.

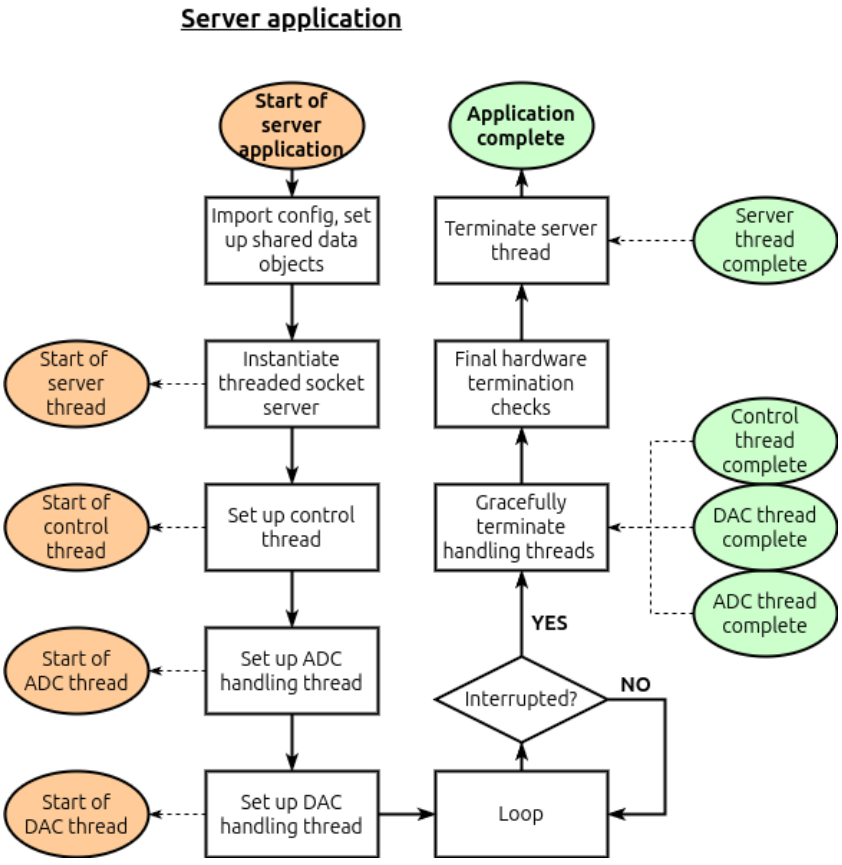
The socket server thread uses the Python library `TCPServer` in conjunction with `ThreadingMixIn` to facilitate communication over the TCP connection. It also uses the `blake2b` library for schedule hashing. Importantly, it uses the same internal library for the communication codec as the high-level interface software, meaning that so long as the latest version of the source code is used on both low-level and high-level interface, they should be able to encode/decode correctly.

The TCP connection is set up such that the Raspberry Pi is the **server**, supporting multiple clients to be connected to it. The high-level interface is a **client**, and communication is set up such that the client requests something from the server, and may receive something back in response. This could be a request for data, a ping, a command instruction, etc. It is not necessary for the server to send something back, but the server is set up such that most client requests will result in some form of acknowledgment, which helps with error detection and debugging the software.

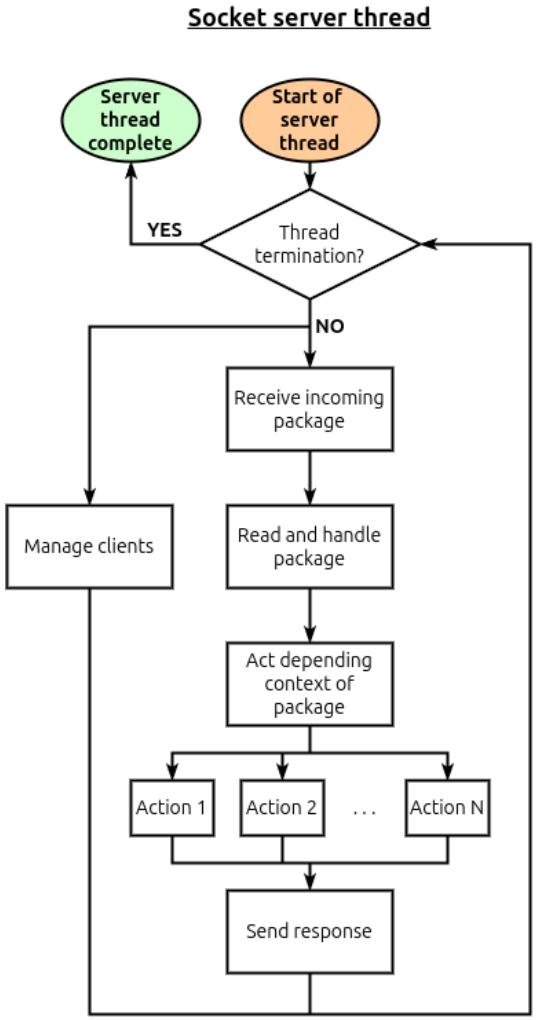
G.1. Functional flow

This section contains a number of flow diagram that describes the functional logic of the code implementation. Each thread has a separate flow diagram.

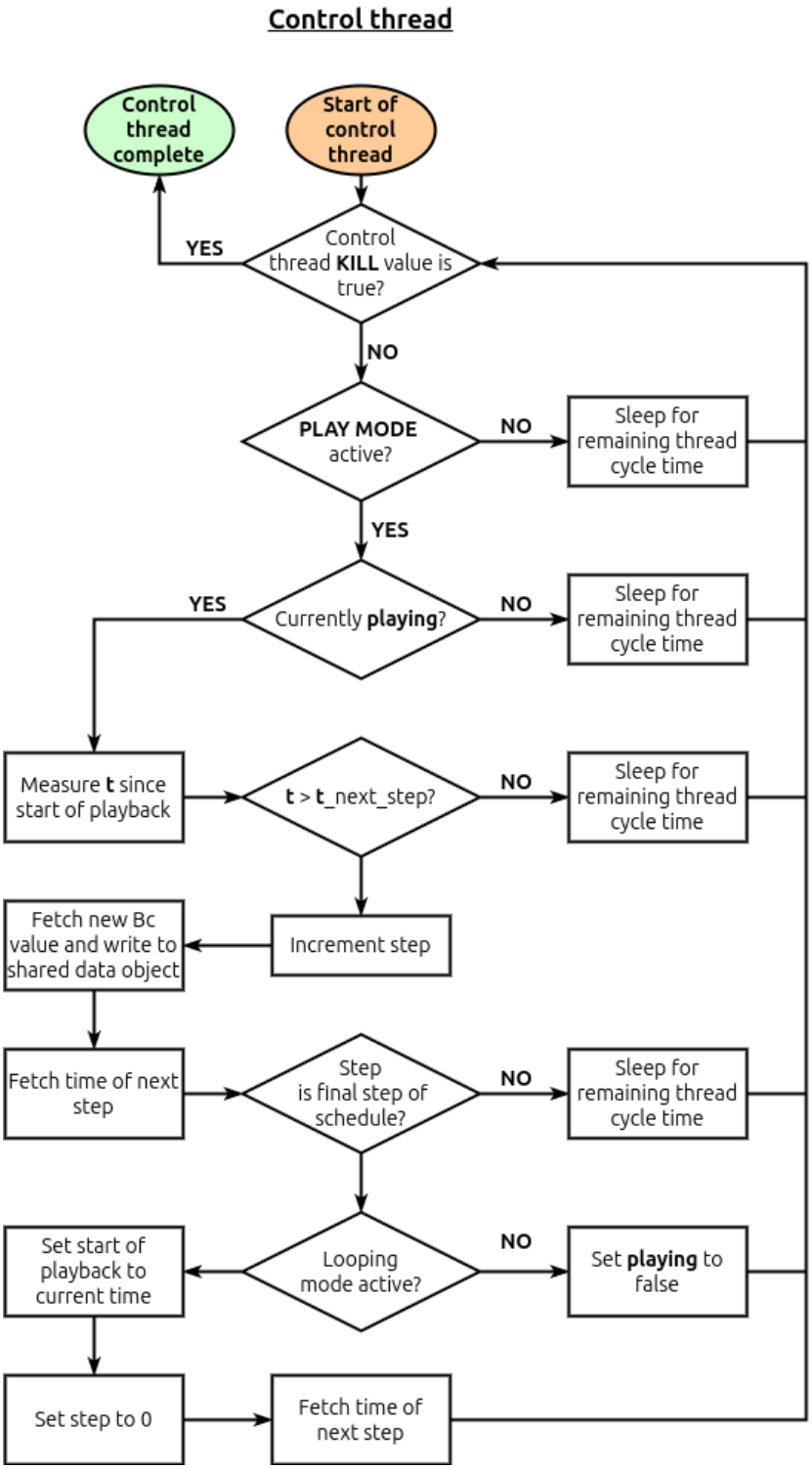
G.1.1. Main server thread



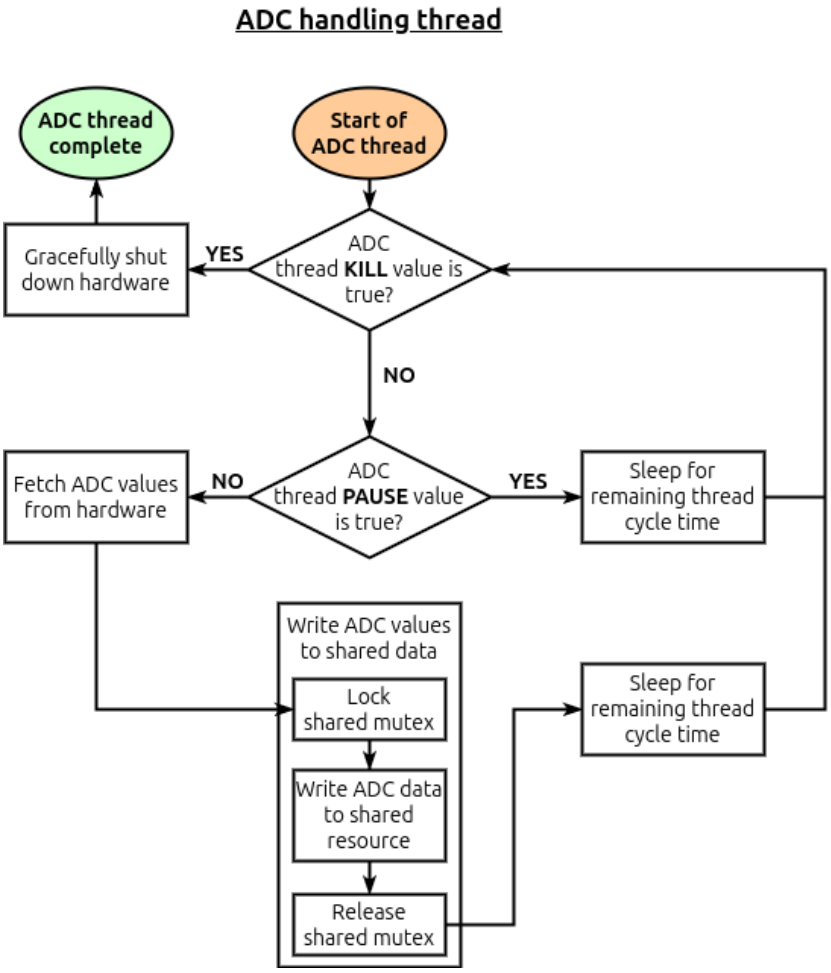
G.1.2. Socket server thread



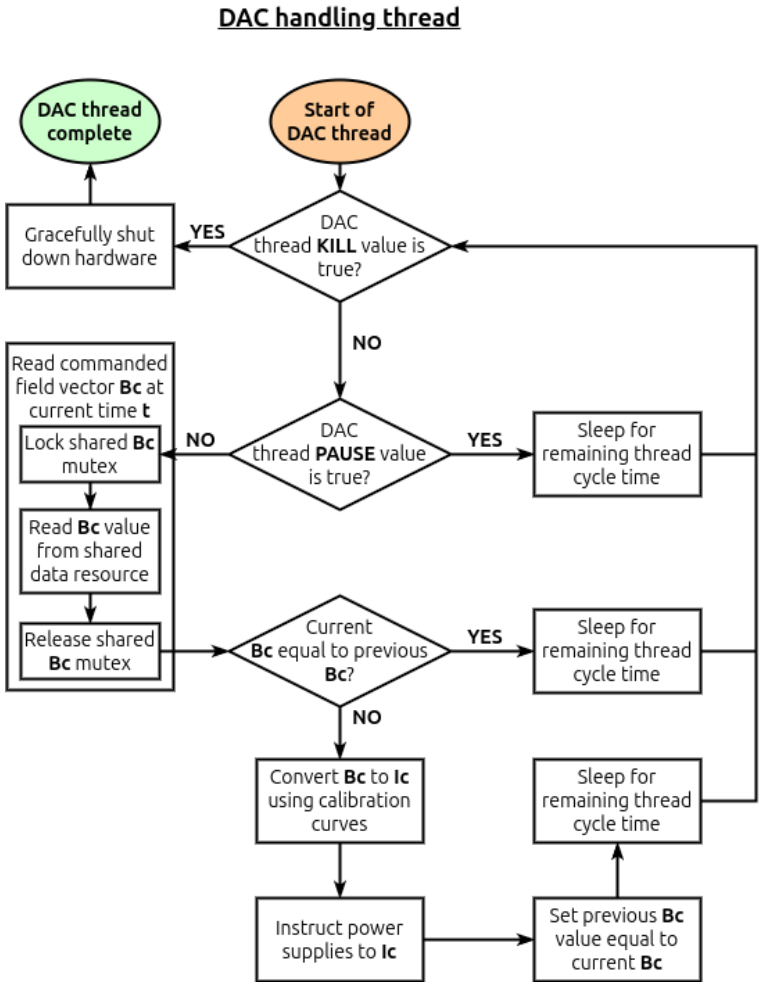
G.1.3. Control thread



G.1.4. ADC thread



G.1.5. DAC thread



G.2. Source code - SCC codec

To facilitate communication over the TCP connection, a custom communication codec was developed from first principles. We refer to this codec as Simple Command Codec (SCC), the latest version of which is version 4. In the early stages of development, it was not quite known how bottle-necked the TCP connection would be, and since Python is also not known to result in fast code, a deliberate effort was made to optimize communication over the TCP connection. This optimization was done with a few considerations in mind:

1. The lower the overall encoding/decoding overhead, the more headroom is available for further development, and or the higher the operation rates are possible for the overall system.
2. Packages which contain time-critical information should ideally be small and fast to encode/decode.
3. The encode/decode functions of the codec should have no/minimal error checking, this should be implemented by the server/client API functions that call the codec.
4. The byte size of the package should be tailored to the contents, such that minimal padding is needed. For example, a package that stores a single boolean value should not be as large as a package that stores a 256-byte string.
5. The packet should have a single character type identifier as the first character.

As mentioned before, a further consideration was that the codec be usable by both client and server, such that only a single version of the codec would serve all needs (unless there are very good reasons not to do so).

Iteration 4 of SCC contains seven packet types in total (see also Figure G.1):

- **Message packet** (M): These packets are meant for sending long string messages.
- **Echo packet** (E): This packet type is identical to the message packet, but it is meant to signal to the receiving socket to echo the contents back. This is mainly useful for troubleshooting the TCP socket connection itself, specifically testing latency (pinging) and signal integrity. Because execution packets have a much higher encoding and decoding overhead, echo packets lend themselves much better to the aforementioned purposes.
- **Control packet** (C): Packet meant to control the power supplies to be set to a certain current output proportional to a magnetic field strength. Packet contains 3 signed `<float>` values in μT.
- **Telemetry packet** (T): Packet containing telemetry in the form of many measured parameters. Mainly used to get as much useful data from the server to a client in a single package. Contains a 20 byte UNIX timestamp, commanded step index, and tri-axial values of measured current $\overline{I}_m$, measured field strength $\overline{B}_m$, and commanded field strength $\overline{B}_c$.
- **B-sample packet** (B): Packet containing a single measurement of the tri-axial magnetometer. Contains a 20 byte UNIX timestamp and 16 byte signed float value in μT for each component of $\overline{B}_m$. B-packets are preferred over T-packets if only measured field strength data is needed and speed is critical, as the encoding/decoding overhead of B-packets is much lower than that of T-packets.
- **Execution packet** (X): Packets meant for sending commands, taking a `<string>` command name, and then the number of appended arguments. The codec keeps track of the type of each argument, with the following types currently supported: `<string>`, `<int>`, `<float>`, `<bool>`.
- **B-schedule segment packet** (S): Packets containing a single line of a B-schedule, which can be used to send a complete B-schedule between client and server. Contains the segment number, the total number of segments in the schedule, the segment's time, and three $\overline{B}_c$ components.

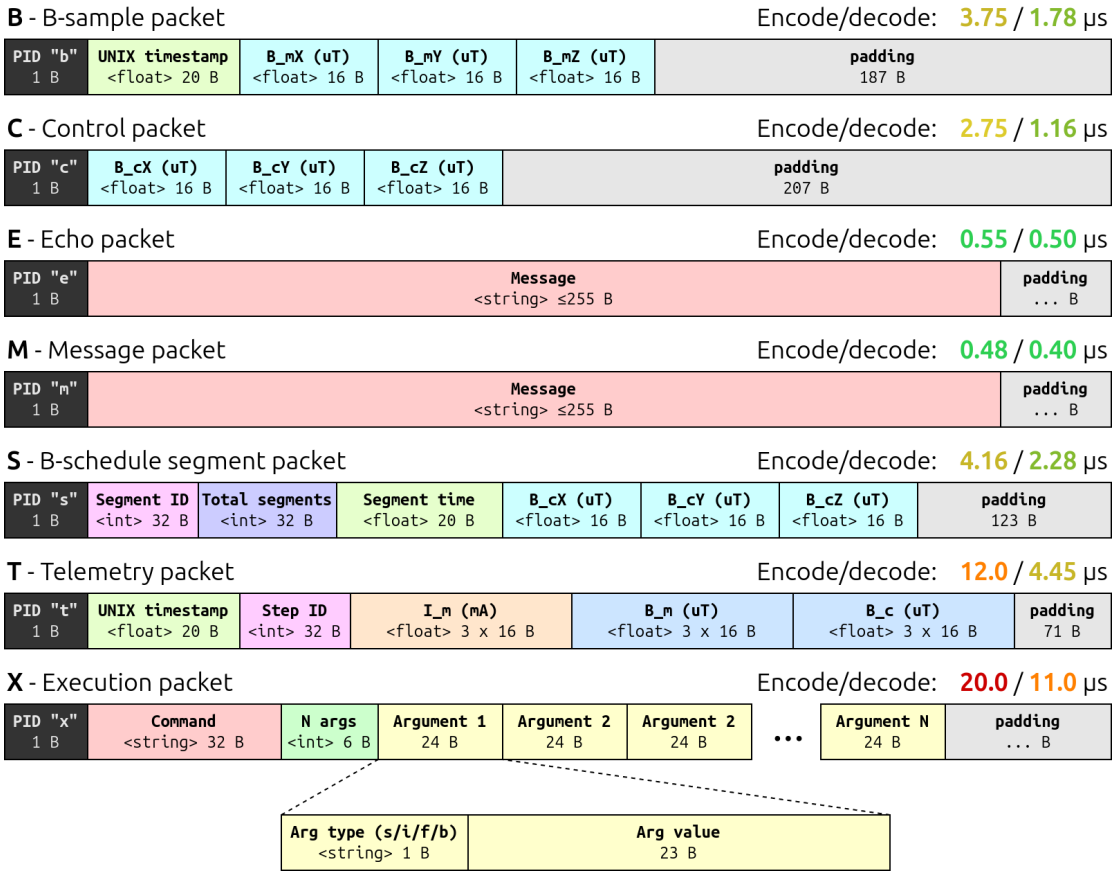


Figure G.1: Anatomy of all packet types in the 4th iteration of the Simple Command Codec

Below follows a source code copy of the latest version of the codec at the time of writing. Further documentation can be found in the comments of this code.

"Simple Command Codec - /helmholtz_cage_toolkit/scc/scc4.py"

```
"""
Simple Command Codec - iteration 4 (SCC4)

Code implementation to facilitate command and message serialization
over a point-to-point TCP/IP connection.

Author: Johan Monster


SSC4 comes with seven packet types. Two of these, the m_packet and x_packet are
best suited for general-purpose, non-time-critical applications. The other
packets are more specialized and designed with a specific purpose in mind, such
as reduction of overhead.


Packet types in SSC4:

Message packet      m_packet    Packets meant for sending long <str> messages

Echo packet         e_packet    This packet type is identical to the message
                                packet, but it is meant to signal to the
                                receiving socket to echo the contents back.
                                This is mainly useful for troubleshooting the
                                TCP socket connection.

Control packet      c_packet    Packet meant to control the power supplies to
                                be set to a certain current output
                                proportional to a magnetic field strength.
                                Packet contains 3 signed <float> values in uT

Telemetry packet    t_packet    Packet containing telemetry in the form of many
                                measured parameters. Mainly used to get as much
                                useful data from the server to a client in a
                                single package. Contains a 20 B UNIX timestamp,
                                commanded step index, and tri-axial values of
                                Im, Bm, and Bc.

B-sample packet     b_packet    Packet containing a single measurement of the
                                triaxial magnetometer. Contains a 20 B UNIX
                                timestamp and 16 B signed float value in uT for
                                each axis. Preferred over t_packet if only Bm
                                data is needed and speed is critical:
                                Encoding + decoding b_packet:   5.55 us
                                Encoding + decoding t_packet:  18.75 us

                                Full in-the-loop test on 03-02-2024:
                                1E5 b-packets exchanged + processed: 57.6 us/p
                                1E5 t-packets exchanged + processed: 81.1 us/p
                                Conclusion: the additional encoding, decoding,
                                and server-side processing of t-packets result
                                in an absolute average overhead of 23.5 us per
                                package compared to b-packets.

Execution packet    x_packet    Packets meant for sending commands, taking
                                a <str> command name, and then any number of
                                arguments. The scc keeps track of the type
                                of each argument, with the following types
                                currently supported: <str>, <int>, <float>,
                                <bool>.

B-schedule segment  s_packet    Packets containing a single line of a
                                B-schedule, which can be used to send a
                                complete B-schedule between client and server.
                                Consists of a type_id, a segment number, the
                                total number of segments in the schedule, the
                                segment's time, and three Bc values.
```

```
67
68
69 Each packet starts with a single character type_id, which can be used to
70 identify an incoming packet even if only a single byte has arrived. The
71 encoding and decoding functions in the scc automatically take care of these,
72 and the packet_type() function can be used on an encoded packet to find its
73 type.
74
75 None of the implemented functions allow you to manually set the type_id of the
76 packet, and this is to force a degree of consistency across implementations.
77 Users desiring additional functionality are advised to subclass SCC instead.
78 """
79
80
81 packet_size = 256   # Referable packet length for buffer size configuration
82 _pad = "#"          # Packet padding character (cannot be @)
83 # _xps = "@"          # (now hardcoded!) Internal separator used for x-packets
84
85
86 def packet_type(packet):
87     """ Returns the first character of the packet, which is the type
88     identifier.
89
90     On optimizations:
91     Previously, this function was implemented as:
92         return packet.decode()[0]
93     Whilst both implementations are similar in performance for short
94     packets, the execution time of the current implementation does not
95     grow with increasing packet size:
96
97     Package size:               69 B     256 B    2048 B
98     ------------------------------------------------------
99     Old implementation:       175 ns    210 ns    512 ns
100    Current implementation:   165 ns    165 ns    165 ns
101
102    (n = 1E7, CPU = AMD FX-8350 @ 4.0 GHz)
103    """
104    return packet[0:1].decode()
105
106
107
108 def encode_bpacket(tm, Bm):
109     """ Encodes a b_packet, which has the following anatomy:
110     b (1 B)    UNIX_time (20 B)    B_X (16 B)    B_Y (16 B)    B_Z (16 B)
111
112     Optimization:
113     3750 ns/encode (FX-8350)
114          ns/encode (Raspberry Pi 4)
115     """
116     return "b{:0<20.20}{:0<16.16}{:0<16.16}{:0<16.16}{}".format(
117         str(tm),
118         str(Bm[0]),
119         str(Bm[1]),
120         str(Bm[2]),
121         _pad*187).encode()
122
123 def decode_bpacket(b_packet):
124     """ Decodes a b_packet, which has the following anatomy:
125     b (1 B)    UNIX_time (20 B)    B_X (16 B)    B_Y (16 B)    B_Z (16 B)
126
127     Efficiency by virtue of the KISS principle:
128     1780 ns/decode (FX-8350)
129          ns/encode (Raspberry Pi 4)
130     """
131     b_decoded = b_packet.decode()
132     return [
133         float(b_decoded[1:21]),
134         float(b_decoded[21:37]),
135         float(b_decoded[37:53]),
136         float(b_decoded[53:69])]
137
```

```
def encode_cpacket(Bc):
    """ Encodes a c_packet, which has the following anatomy:
    c (1 B)    Bc_X (16 B)    Bc_Y (16 B)    Bc_Z (16 B)

    Optimization:
    2750 ns/encode (FX-8350)
         ns/encode (Raspberry Pi 4)
    """
    return "c{:0<16.16}{:0<16.16}{:0<16.16}{}".format(
        str(Bc[0]),
        str(Bc[1]),
        str(Bc[2]),
        _pad*207).encode()

def decode_cpacket(c_packet):
    """ Decodes a c_packet, which has the following anatomy:
    c (1 B)    Bc_X (16 B)    Bc_Y (16 B)    Bc_Z (16 B)

    Efficiency by virtue of the KISS principle:
    1160 ns/decode (FX-8350)
         ns/encode (Raspberry Pi 4)
    """
    c_decoded = c_packet.decode()
    return [
        float(c_decoded[1:17]),
        float(c_decoded[17:33]),
        float(c_decoded[33:49])]



def encode_epacket(msg: str):
    """ Encodes an e_packet, which has the following anatomy:
    e (1 B)    msg (n B)

    Optimization:
    550 ns/encode (FX-8350)
     ns/encode (Raspberry Pi 4)
    """
    return ("e"+msg[:255]+_pad*(packet_size-len(msg)-1)).encode()

def decode_epacket(e_packet):
    """ Encodes an e_packet, which has the following anatomy:
    e (1 B)    msg (n B)

    Optimization:
    500 ns/encode (FX-8350)
     ns/encode (Raspberry Pi 4)
    """
    # return e_packet.decode()[1:].rstrip(_pad)
    e_packet_decoded = e_packet.decode()
    return e_packet_decoded[1:e_packet_decoded.find(_pad)]


def encode_mpacket(msg: str):
    """ Encodes an m_packet, which has the following anatomy:
    m (1 B)    msg (n B)

    The allowed length of msg is equal to packet_size-1

    Optimization:
    480 ns/encode (FX-8350)
     ns/encode (Raspberry Pi 4)
    """
    return ("m"+msg[:255]+_pad*(packet_size-len(msg)-1)).encode()

def decode_mpacket(m_packet):
    """ Encodes an m_packet, which has the following anatomy:
    m (1 B)    msg (n B)

    The allowed length of msg is equal to packet_size-1

```

```
    Optimization:
    400 ns/encode (FX-8350)
     ns/encode (Raspberry Pi 4)
    """
    return m_packet.decode()[1:].rstrip(_pad)



def encode_spacket(
    i_seg: int,
    n_seg: int,
    t_seg: float,
    Bx_seg: float,
    By_seg: float,
    Bz_seg: float):
    """ Encodes an s_packet, which has the following anatomy:
    s (1 B)    segment number (32 B)    number of segments (32 B)
        segment_time (20 B)    B_X (16 B)    B_Y (16 B)    B_Z (16 B)

    Optimization:
    4160 ns/encode (FX-8350)
     ns/encode (Raspberry Pi 4)
    """
    return "s{:0>32.32}{:0>32.32}{:0<20.20}{:0<16.16}{:0<16.16}{:0<16.16}{}".format(
        str(i_seg),
        str(n_seg),
        str(t_seg),
        str(Bx_seg),
        str(By_seg),
        str(Bz_seg),
        _pad*123).encode()

def decode_spacket(s_packet):
    """ Decodes an s_packet, which has the following anatomy:
    s (1 B)    segment number (32 B)    number of segments (32 B)
        segment_time (20 B)    B_X (16 B)    B_Y (16 B)    B_Z (16 B)

    to segment values.

    Optimization:
    2280 ns/encode (FX-8350)
     ns/encode (Raspberry Pi 4)
    """
    s_decoded = s_packet.decode()
    return [
        int(float(s_decoded[1:33])),
        int(float(s_decoded[33:65])),
        float(s_decoded[65:85]),
        float(s_decoded[85:101]),
        float(s_decoded[101:117]),
        float(s_decoded[117:133])
    ]



def encode_tpacket(tm, i_step, Im, Bm, Bc):
    """ Encodes a t_packet, which has the following anatomy:
    b (1 B)    UNIX_time (20 B)    i_step (32 B)    Im (3x12 B)
               Bm (3x16 B)         Bc (3x16 B)      padding (71 B)
    Optimization:
    12000 ns/encode (FX-8350)
     ns/encode (Raspberry Pi 4)
    """
    output = [str(tm), str(i_step), ]
    for par in (Im, Bm, Bc):
        output += [str(par[0]), str(par[1]), str(par[2])]
    return ("t{:0<20.20}"+"{:0>32.32}"+"{:0<12.12}"*3+"{:0<16.16}"*6+"#"*71
            ).format(*output).encode()


def decode_tpacket(t_packet):
```

```
    """ Decodes a t_packet, which has the following anatomy:
    b (1 B)    UNIX_time (20 B)    i_step (32 B)    Im (3x12 B)
               Bm (3x16 B)         Bc (3x16 B)      padding (71 B)
    Optimization:
    4450 ns/encode (FX-8350)
     ns/encode (Raspberry Pi 4)
    """
    t_decoded = t_packet.decode()
    return float(t_decoded[1:21]), \
        int(t_decoded[21:53]), \
        [
            float(t_decoded[53:65]),        # \
            float(t_decoded[65:77]),        # Im
            float(t_decoded[77:89]),        # /
        ], [
            float(t_decoded[89:105]),       # \
            float(t_decoded[105:121]),      # Bm
            float(t_decoded[121:137]),      # /
        ], [
            float(t_decoded[137:153]),      # \
            float(t_decoded[153:169]),      # Bc
            float(t_decoded[169:185]),      # /
        ]



def encode_xpacket(cmd: str, *args):
    """ Encodes an x_packet, which has the following anatomy:
    x (1 B)    cmd (32 B)    n_args(6 B)    type_id , arg (1+23 B each)

    The tail of an x_packet consists of n_args segments. Each starts with
    an arg_type_id ('s', 'i', 'f', 'b' for string, integer, float, and
    boolean respectively), followed by 23 bytes of encoded value.

    Unoptimized as of 25-01-2024:
    ~10000-20000 ns/encode (FX-8350)
     ns/encode (Raspberry Pi 4)
    """
    recognized_types = (int, float, str, bool)

    n_args = len(args)
    n_correct_type = sum([type(arg) in recognized_types for arg in args])

    if n_correct_type != n_args:
        raise TypeError(
            f"encode_xpacket() received {n_args - n_correct_type} argument(s) of incorrect
                type! (must be str, int, float, or bool)")

    xpacket_unencoded = "x{:@>32}{:0>6}".format(cmd, n_args)

    for arg in args:
        if type(arg) == float:
            xpacket_unencoded += "f{:0<23}".format(round(arg, 8))
        elif type(arg) == int:
            xpacket_unencoded += "i" + ("{:@>23}".format(arg))[:23]
        elif type(arg) == bool:
            if arg is True:
                xpacket_unencoded += "b" + "{:1>23}".format(1)
            else:
                xpacket_unencoded += "b" + "{:0>23}".format(0)
        elif type(arg) == str:
            xpacket_unencoded += "s{:@>23}".format(arg[:23])

    if len(xpacket_unencoded) > packet_size:
        raise AssertionError(f"Total packet length exceeds 256 B ({len(xpacket_unencoded)} B)
            !")
    else:
        xpacket_unencoded += "{}".format(_pad*(packet_size-len(xpacket_unencoded)))
        return xpacket_unencoded.encode()

def decode_xpacket(x_packet):
```

```
    """ Decodes a c_packet, which has the following anatomy:
    x (1 B)    cmd (32 B)    n_args(6 B)    type_id , arg (1+23 B each)

    Unoptimized as of 15-01-2024.
    ~5000-11000 ns/encode (FX-8350)
     ns/encode (Raspberry Pi 4)
    """

    xpacket_decoded = x_packet.decode().rstrip(_pad)

    cmd_name = xpacket_decoded[1:33].strip("@")
    n_args = int(xpacket_decoded[33:39])

    # print("cmd_name:", cmd_name, "n_args:", n_args)

    args = []
    for i_seg in range(n_args):
        seg = xpacket_decoded[39 + 24 * i_seg:39 + 24 * (i_seg + 1)]
        # print("[DEBUG]", i_seg, seg, end="  ->  ")

        if seg[0] == "f":
            args.append(float(seg[1:]))
        elif seg[0] == "i":
            args.append(int(seg[1:].strip("@")))
        elif seg[0] == "b":
            args.append(bool(int(seg[1])))
        elif seg[0] == "s":
            args.append(seg[1:].strip("@"))
        else:
            raise ValueError(
                f"decode_xpacket(): Encountered uninterpretable type_id '{seg[0]}' in segment
                    {i_seg}: {seg}")

        # print(f"{args[i_seg]} ({type(args[i_seg])})")

    return cmd_name, args
```

G.3. Server code source code

We conclude this appendix with a snapshot of the source code for the server. This is the code that runs on the Raspberry Pi

"Helmholtz cage server code - /helmholtz_cage_toolkit/server/server.py"

```
"""
This is the main script for the server side of the Helmholtz Cage Toolkit,
intended to be run from the Raspberry Pi controlling all the hardware.
Its design is relatively simple, but since the implementation fully supports
multi-threading, things tend to look a bit more complicated than they ought
to.

This script is intended to be run directly, preferably invoked via the
terminal. The code to set up the server can be found after the line:
    if __name__ == "__main__":

The basic layout of the server setup is as follows:
1. Create Datapool(), a general-purpose data container that the server will use
    as a thread-safe data hub. The definition of the server Datapool is also in
    this file, and is altogether distinct to the Datapool used by the clients.
2. Fetch own address and port information from the server_config, via the
    Datapool.
3. Create a ThreadedTCPServer instance, which is the main and only server
    instance. Its creation also defines a ThreadedTCPRequestHandler. Each
    connected client will interact with their own instance of this handler,
    running in a separate thread to keep things compartmentalized.
4. The server object is placed in its own thread, which is then started.
5. Two separate threads are created and started:
    - A thread running the function threaded_write_DAC(), used for DAC hardware
    interactions.
    - A thread running the function threaded_read_ADC(), used for ADC hardware
```

```
    interactions.

After this, the server will be fully operational and run in perpetuity unless
terminated or an exception occurs. When this occurs, the threads are one by one
stopped, some work is done to finish gracefully, and finally the server Thread
will be terminated.

The server implementation relies on the server config and the SCC codec.
"""


from socketserver import TCPServer, ThreadingMixIn, BaseRequestHandler
from threading import Thread, Lock, main_thread, active_count

from hashlib import blake2b
from numpy import array, zeros
from numpy.random import rand
from time import time, sleep

import helmholtz_cage_toolkit.scc.scc4 as codec
from helmholtz_cage_toolkit.server.server_config import server_config as config



def hardware_shutdown(datapool):
    """
    In case of a software error or upon termination of the server, it is of
    critical importance that the hardware is properly terminated. Without this,
    the hardware can still output power without user control, store energy,
    cause voltage surges, etc.

    This routine should be called in such cases, and will properly terminate
    the hardware to a safe idling state. Given that the Helmholtz Cage Toolkit
    is in many places relatively hardware agnostic, the proper termination
    procedure is hardware specific nonetheless, depending for example on the
    slewing properties of power supplies. If you are using particular hardware,
    note that you may have to rewrite/overload this function with your own
    implementation.

    The basic overview of the termination procedure is as follows:
    1. Command all power supplies to output zero amp.
    2. Sleep for a specific period to allow the power supplies to slew to zero.
    3. Reset the H-bridges to their default off state.
    4. Inhibit the output of either the PSUs or the H-bridges with the
        dedicated software toggle implementation.
    """
    print("hardware_shutdown() called")


def threaded_control(datapool):
    """This thread controls information routing based on the playback modes
    and some serveropts. It implements the following scheme:

    play_mode is False (Manual mode):   user input -> Bc

    play_mode is True (playback mode): i_step schedule -> Bc
        if play is True:                move schedule through time

    serveropt_inject_Bm:                Bc -> Bm

    serveropt_mutate_Bm:
        if not serveropt_inject_Bm:     mutate() -> Bm
    """
    print(f"Started 'control' thread with period {datapool.threaded_control_period}")
    datapool.kill_threaded_control = False

    t_prev_mutate = 0.0
    t_prev_inject = 0.0

    while not datapool.kill_threaded_control:
        # Bm routing based on certain serveropts:
```

```
        if datapool.serveropt_inject_Bm and \
            (2*(time() - t_prev_inject) > datapool.threaded_read_ADC_period):
            Bc = datapool.read_Bc()
            datapool.write_Bm(Bc)
            t_prev_inject = time()

        if datapool.serveropt_mutate_Bm \
                and not datapool.serveropt_inject_Bm \
                and (2*(time() - t_prev_mutate) > datapool.threaded_read_ADC_period):
            Bm_prev = datapool.read_Bm()[1]
            # print("[DEBUG] Bm_prev", Bm_prev)
            Bm = []
            for i in (0, 1, 2):
                Bm.append(datapool.mutate(
                    Bm_prev[i],
                    datapool.params_mutate[0][i],
                    datapool.params_mutate[1][i],
                    datapool.params_mutate[2][i]
                ))
            datapool.write_Bm(Bm)
            t_prev_mutate = time()
            # datapool.write_Bm(datapool.mutate(datapool.read_Bm()[1]))
            # datapool.write_buffer(datapool.Bm, datapool.mutate(datapool.Bm[0]))


        # ==== PLAY MODE ====
        if datapool.play_mode:
            # Do only if playback is active
            if datapool.play is True:
                # Measure current time since start of play
                datapool.t_current = time() - datapool.t_play

                # If it is time, move to the next step, unless end of schedule is reached
                if datapool.t_current >= datapool.t_next:
                    if datapool.i_step[0] == datapool.n_steps - 2:
                        datapool.write_buffer(datapool.i_step, datapool.i_step[0]+1)
                        datapool.write_Bc(datapool.schedule[datapool.i_step[0]][3:6])
                        # print(
                        #     f"[DEBUG] Current step: {datapool.i_step}/{datapool.n_steps}
                            (+{round(datapool.t_current, 3)} s)")
                        if datapool.play_looping:
                            datapool.write_buffer(datapool.i_step, 0)
                            datapool.t_play = time()
                            datapool.t_next = datapool.schedule[1][2]
                            print(f"[DEBUG] Reached end of schedule -> RESETTING")

                        else:
                            confirm = datapool.set_play(False)
                            print(f"[DEBUG] Reached end of schedule -> STOPPING")

                    else:
                        datapool.write_buffer(datapool.i_step, datapool.i_step[0] + 1)
                        # datapool.i_step += 1
                        # instruct_DACs(datapool, datapool.schedule[datapool.i_step][3:6])
                        datapool.write_Bc(datapool.schedule[datapool.i_step[0]][3:6])
                        datapool.t_next = datapool.schedule[datapool.i_step[0] + 1][2]
                        # print(
                        #     f"[DEBUG] Current step: {datapool.i_step}/{datapool.n_steps}
                            (+{round(datapool.t_current, 3)} s)")
                    sleep(max(0., datapool.threaded_control_period - (time() - t0)))

                else:
                    sleep(datapool.threaded_control_period)

            else:
                sleep(datapool.threaded_control_period)

        # ==== MANUAL MODE ====
        else:
            sleep(datapool.threaded_control_period)

```

```
    print(f"Closing control thread")



def threaded_write_DAC(datapool):
    # TODO UPDATE DOCSTRING
    """The thread running this function will periodically read the value of
    DAC inputs from the datapool, and apply it to the DAC hardware. It can do
    so in manual mode, where it applies the value every time a DAC input is
    changed, or it can do it in play mode, where it plays the pre-defined
    schedule stored in datapool.schedule.

    The basic idea is this:
    -1. Run until datapool.kill_threaded_write_DAC is set to True, then finish
    0. Determine if in "play mode" or not ("manual mode" -> default)
    If in "play mode":
        1. Compare current time to the time of next point in schedule
        2. If time of next point is reached, move forward, write Bc value to
            DACs, and calculate next time to move

    If in "manual mode":
        1. Thread-safely read datapool.Bc
        2. Compare value to previously stored value
        3. If no change was made, sleep for 'period' and start again
        4. If a change was detected, apply this change by instructing the DACs
        5. Also store a copy to datapool.Bc_applied

    The thread running this function will be in charge of controlling hardware.
    It is important for this hardware that in the event of a software
    exception, the hardware is reset to zero before the code is fully
    terminated. Otherwise, the power supplies will hang at a non-zero value
    and remain uncontrollable until the server is restarted.

    In order to ensure that this is done properly, the thread should not be
    run as a daemon, and when calling .join() on it, the parent thread should
    give this thread a timeout that is larger than the period.

    So, in order to gracefully kill the thread you do the following:
    1. Set datapool.kill_threaded_write_DAC to True
    2. This will break the loop, and run a hardware shutdown routine.
    3. Join thread with nonzero timeout; e.g. Thread.join(timeout=1.0)
    This will end the thread on its next cycle, and should ensure that the
    code is never terminated with a non-zero value on the power supplies.

    TODO: Currently, this thread cannot be stopped, restarted, or its period
    adjusted. This would however be desirable. To fix this, the way this
    thread is initiated in the server thread should be changed.
    """

    print(f"Started 'write DAC' thread with period {datapool.threaded_write_DAC_period}")
    datapool.kill_threaded_write_DAC = False

    Bc_prev = [0., 0., 0.]

    # TODO Implement software slew rate limiting for dI/dt
    # TODO Basically for all three values compute abs(dI/dt) and do not change
    # TODO Bc more than the limit allows within the DAC writing speed.
    # TODO Useful for ensuring no hardware is damaged.

    while not datapool.kill_threaded_write_DAC:  # Kills loop when set to True
        if not datapool.pause_threaded_write_DAC:  # Pause loop when set to True
            # print("threaded_write_DAC() loop")
            t0 = time()
            Bc_read = [0., 0., 0.]
            if Bc_read == Bc_prev:
                sleep(max(0., datapool.threaded_write_DAC_period - (time() - t0)))
            else:
                pass # TODO Apply Bc_read to the hardware!
        else:
            sleep(datapool.threaded_write_DAC_period)

```

```
    # When loop is broken, set power supply values to zero.
    # TODO implement safe shutdown
    print(f"Closing write_DAC thread")
    hardware_shutdown(datapool)


def threaded_read_ADC(datapool):
    # TODO UPDATE DOCSTRING
    """The thread running this function will periodically read the value of
    ADC inputs from the hardware, and write them to the datapool.

    The basic idea is this:
    -1. Run until datapool.kill_threaded_read_ADC is set to True, then finish


    So, in order to gracefully kill the thread you do the following:
    1. Set datapool.kill_threaded_read_ADC to True
    2. Join thread with nonzero timeout; e.g. Thread.join(timeout=1.0)
    This will end the thread on its next cycle, and should ensure that the
    code is never terminated with a non-zero value on the power supplies.

    """

    print(f"Started 'read ADC' thread with rate {1/datapool.threaded_read_ADC_period}")
    datapool.kill_threaded_read_ADC = False

    while not datapool.kill_threaded_read_ADC:  # Kills loop when set to True

        if not datapool.pause_threaded_read_ADC:  # Pause loop when set to True
            t0 = time()
            # print("threaded_read_ADC() loop")

            pass # TODO DO ADC READING STUFF

            # TODO if serveropt_mutate_Bm or serveropt_inject_Bm -> Do not write_Bm() here!

            # print(f"[DEBUG] Bm = {datapool.read_Bm()}")
            sleep(max(0., datapool.threaded_read_ADC_period - (time() - t0)))

        else:
            sleep(datapool.threaded_read_ADC_period)

    print(f"Closing read ADC thread")


# DataPool object
class DataPool:
    def __init__(self):
        # Thread controls
        self._lock_ADC = Lock()                 # Thread lock for ADC value buffers
        self._lock_DAC = Lock()                 # Thread lock for DAC value buffers
        self._lock_schedule = Lock()            # Thread lock for schedule

        self.pause_threaded_read_ADC = False    # Not thread-safe
        self.pause_threaded_write_DAC = False   # Not thread-safe

        self.kill_threaded_control = False     # Not thread-safe
        self.kill_threaded_read_ADC = False     # Not thread-safe
        self.kill_threaded_write_DAC = False    # Not thread-safe

        self.threaded_control_period = 1/config["threaded_control_rate"]  # Not thread-safe
        self.threaded_read_ADC_period = 1/config["threaded_read_ADC_rate"]  # Not thread-safe
        self.threaded_write_DAC_period = 1/config["threaded_write_DAC_rate"]  # Not thread-
            safe


        # ==== Data buffers ================================================
        """ Here all data buffers are defined. They are all defined as lists,
        such that previous data can be stored for potential analysis.

        Entry 0 will always be the most recent value, and should therefore be
```

```
        useful in most cases. To write to the buffer, use self.write_buffer().
        """

        ibs = config["internal_buffer_size"]

        self.tm = self.init_buffer(ibs, 1)      # Time at which Bm, Im were taken
        self.Im = self.init_buffer(ibs, 3)      # Measured current Im in [A]
        self.Bm = self.init_buffer(ibs, 3)      # Measured field Bm in [uT]

        self.Bc = self.init_buffer(ibs, 3)      # Control vector Bc to be applied [uT]
        self.Vvc = self.init_buffer(ibs, 3)  # Currently unused
        self.Vcc = self.init_buffer(ibs, 3)  # Currently unused

        self.Br = self.init_buffer(ibs, 3)      # Magnetic field vector to be rejected

        self.V_board = self.init_buffer(ibs, 1) # Measured value of +12V bus - +5V bus

        self.i_step = self.init_buffer(ibs, 1, dtype=int)

        self.aux_adc = self.init_buffer(ibs, 1)
        self.aux_dac = self.init_buffer(ibs, 6)

        # ==== Other parameters ================================================
        self.params_tf_VB = {
            "x": config["params_tf_VB_x"],
            "y": config["params_tf_VB_y"],
            "z": config["params_tf_VB_z"],
        }

        self.params_mutate = config["params_mutate"]

        self.output_enable = False              # Enable/disable H-bridge output using PSUE
            pin

        # Initialize schedule
        self.initialize_schedule()


        # ==== Play controls ===================================================
        # Looping: True: Plays schedule over and over | False: One-shot schedule playback
        self.play_looping = config["default_play_looping"]
        self.play_mode = False          # True: Can playback | False: Manual control only
        self.play = False               # True: Playing | False: Stopped
        self.n_steps = 1                # Number of steps in schedule
        self.t_play = 0.0               # UNIX time at which "play" began
        self.t_current = 0.0            # Current time in schedule
        self.t_next = 0.0               # Time of next step in schedule


        # ==== Serveropts ======================================================
        self.serveropt_mutate_Bm = config["mutate_Bm"]
        self.serveropt_inject_Bm = config["inject_Bm"]

    # ==== INTERNAL FUNCTIONS ==================================================

    def write_buffer(self, buffer: list, value):
        """First-in, last-out buffer update function.
        - Increases length of buffer by 1 by inserting new value at position 0
        - Deletes the last value of the buffer using pop()
        Beware: There is NO input validation or type/size checking, you must
        ensure that the value that is written is appropriate!
        """
        buffer.insert(0, value)
        buffer.pop()

    def init_buffer(self, buffer_size, entry_size, dtype=float) -> list:
        """Automatically creates a buffer object, which is a list with a number
        of entries. The number of entries is equal to <buffer_size>.
        The entries themselves are lists if <entry_size> is larger than 1,
        making the buffer 2D. For <entry_size> equal to 1, the buffer will be
        1D instead. The entries will be floats by default, but the data typing
```

```
        of the entries can be controlled using the dtype argument.

        [DEV NOTE] Buffers are created by creating a Numpy array of zeroes of
        the right type, then using .tolist() to cast it into a list of lists.
        This will create the correct data format, but for some reason the
        output of (array).tolist() will be signalled as an <object> rather than
        a <list>, causing the linter to misreport it. Hence, the operation is
        enclosed in another list() operation, at negligible performance cost.
        This now causes a linter misreport on the list() calls, but the error
        is now localized to this function and can be ignored using # noqa.
        """
        if buffer_size <= 0:
            raise ValueError(f"init_buffer(): buffer_size cannot be {buffer_size}!")

        if entry_size == 1:
            return list(zeros(buffer_size, dtype=dtype).tolist()) # noqa
        elif entry_size > 1:
            return list(zeros((buffer_size, entry_size), dtype=dtype).tolist()) # noqa
        else:
            raise ValueError(f"init_buffer(): Negative entry_size given!")

    def mutate(self, v_prev, v_central, mutation_scale, fence_strength):
        """Mutate a value from a starting value, but push back if it gets too
        far from some defined central value.
         - v_prev is the value to mutate
         - v_central is the value around which the mutation will move
         - mutation_scale is the size of the mutation steps
         - fence_strength increases how tightly the mutation is kept near
            v_central
        Convergence is not guaranteed for all values. If mutation_scale is too
        large in comparison to fence_strength and v_central, the results may
        become unbound.
        """
        # strs = ["v_prev", "v_central", "mutation_scale", "fence_strength"]
        # for i, var in enumerate((v_prev, v_central, mutation_scale, fence_strength)):
        #     print("[DEBUG]", strs[i], var)

        m = mutation_scale * v_central * (2 * rand() - 1)
        d = v_prev - v_central

        # d=0 causes a singularity later, so side-step it
        if d == 0:
            d = 0.1

        if d / abs(d) == m / abs(m):
            mutagen = m * (1 - abs(d / v_central) * fence_strength)
        else:
            mutagen = m * (1 + abs(d / v_central) * fence_strength)

        return v_prev + mutagen

    def measure_performance(self):
        pass # TODO

    def auto_calibrate(self):
        pass # TODO


    # ==== IO FUNCTIONS ======================================================

    def read_Bm(self):
        """Thread-safely reads the current Bm field from the datapool.

        The lock prevents other threads from updating self.Bm whilst it is
        being read. Useful to prevent hard-to-debug race condition bugs.
        """
        self._lock_ADC.acquire(timeout=0.001)
        try:
            tm = self.tm[0]
            Bm = self.Bm[0]
        except:  # noqa
```

```
449             print("[WARNING] DataPool.read_Bm(): Unable to read self.Bm!")
450         self._lock_ADC.release()
451         return tm, Bm
452 
453     def write_Bm(self, Bm: list):
454         """Thread-safely write Bm to the datapool.
455 
456         The lock prevents other threads from accessing self.Bm whilst it is
457         being updated. Useful to prevent hard-to-debug race condition bugs.
458         """
459         print(f"[DEBUG] write_Bm({Bm})")
460         self._lock_ADC.acquire(timeout=0.001)
461         try:
462             self.write_buffer(self.Bm, Bm)
463         except:  # noqa
464             print("[WARNING] DataPool.write_Bm(): Unable to write to self.Bm!")
465         self._lock_ADC.release()
466 
467     def read_Bc(self):
468         """Thread-safely reads the current Bc field from the datapool.
469 
470         The lock prevents other threads from updating self.Bc whilst it is
471         being read. Useful to prevent hard-to-debug race condition bugs.
472         """
473         self._lock_DAC.acquire(timeout=0.001)
474         try:
475             Bc = self.Bc[0]
476         except:  # noqa
477             print("[WARNING] DataPool.read_Bc(): Unable to read self.Bc!")
478         self._lock_DAC.release()
479         return Bc
480 
481     def write_Bc(self, Bc: list):
482         """Thread-safely write Bc to the datapool.
483 
484         The lock prevents other threads from accessing self.Bc whilst it is
485         being updated. Useful to prevent hard-to-debug race condition bugs.
486         """
487         print(f"[DEBUG] write_Bc({Bc})")
488         self._lock_DAC.acquire(timeout=0.001)
489         try:
490             self.write_buffer(self.Bc, Bc)
491         except:  # noqa
492             print("[WARNING] DataPool.write_Bc(): Unable to write to self.Bc!")
493         self._lock_DAC.release()
494 
495     def read_Br(self):
496         """Thread-safely reads the current Br field from the datapool.
497 
498         The lock prevents other threads from updating self.Bc whilst it is
499         being read. Useful to prevent hard-to-debug race condition bugs.
500         """
501         self._lock_DAC.acquire(timeout=0.001)
502         try:
503             Br = self.Br[0]
504         except:  # noqa
505             print("[WARNING] DataPool.read_Br(): Unable to read self.Br!")
506         self._lock_DAC.release()
507         return Br
508 
509     def write_Br(self, Br: list):
510         """Thread-safely write Br to the datapool.
511 
512         The lock prevents other threads from accessing self.Bc whilst it is
513         being updated. Useful to prevent hard-to-debug race condition bugs.
514         """
515         self._lock_DAC.acquire(timeout=0.001)
516         try:
517             self.write_buffer(self.Br, Br)
518         except:  # noqa
519             print("[WARNING] DataPool.write_Br(): Unable to write to self.Br!")
```

```
520         self._lock_DAC.release()
521
522     def read_telemetry(self):
523         """Thread-safely reads the telemetry data from the datapool.
524
525         The lock prevents other threads from updating this data whilst it is
526         being read. Useful to prevent hard-to-debug race condition bugs.
527         """
528         self._lock_ADC.acquire(timeout=0.001)
529         self._lock_DAC.acquire(timeout=0.001)
530         try:
531             tm = self.tm[0]
532             i_step = self.i_step[0]
533             Im = self.Im[0]
534             Bm = self.Bm[0]
535             Bc = self.Bc[0]
536         except:  # noqa
537             print("[WARNING] DataPool.read_telemetry(): Unable to read telemetry!")
538         self._lock_ADC.release()
539         self._lock_DAC.release()
540         return tm, i_step, Im, Bm, Bc
541
542     def write_adc_data(self, Bm, Im, V_board, aux_adc):
543         """Thread-safely writes all ADC data to their corresponding fields
544          in the datapool.
545
546         The lock prevents other threads from accessing this data whilst it is
547         being read. Useful to prevent hard-to-debug race condition bugs.
548         """
549         self._lock_ADC.acquire(timeout=0.001)
550         try:
551             self.write_buffer(self.Bm, Bm)
552             self.write_buffer(self.Im, Im)
553             self.write_buffer(self.V_board, V_board)
554             self.write_buffer(self.aux_adc, aux_adc)
555         except:  # noqa
556             print("[WARNING] DataPool.read_Bm(): Unable to read self.Bm!")
557         self._lock_ADC.release()
558
559     def read_aux_adc(self):
560         """Thread-safely reads the ADC aux1 channel data from the datapool.
561
562         The lock prevents other threads from updating this data whilst it is
563         being read. Useful to prevent hard-to-debug race condition bugs.
564         """
565         self._lock_ADC.acquire(timeout=0.001)
566         try:
567             aux_adc = self.aux_adc[0]
568         except:  # noqa
569             print("[WARNING] DataPool.read_aux_adc: Exception!")
570         self._lock_ADC.release()
571         return aux_adc
572
573     def read_V_board(self):
574         """Thread-safely reads the board voltage data from the datapool.
575
576         The lock prevents other threads from updating this data whilst it is
577         being read. Useful to prevent hard-to-debug race condition bugs.
578         """
579         self._lock_ADC.acquire(timeout=0.001)
580         try:
581             V_board = self.V_board[0]
582         except:  # noqa
583             print("[WARNING] DataPool.read_V_board(): Exception!")
584         self._lock_ADC.release()
585         return V_board
586
587     def read_output_enable(self):
588         """Thread-safely reads the output_enable toggle from the datapool.
589
590         The lock prevents other threads from updating this data whilst it is
```

```
        being read. Useful to prevent hard-to-debug race condition bugs.
        """
        self._lock_DAC.acquire(timeout=0.001)
        try:
            output_enable = self.output_enable
        except:  # noqa
            print("[WARNING] DataPool.read_output_enable(): Exception!")
        self._lock_DAC.release()
        return output_enable

    def write_output_enable(self, output_enable: bool):
        """Thread-safely writes the output_enable toggle to the datapool.

        The lock prevents other threads from updating this data whilst it is
        being read. Useful to prevent hard-to-debug race condition bugs.
        """
        self._lock_DAC.acquire(timeout=0.001)
        try:
            self.output_enable = output_enable
        except:  # noqa
            print("[WARNING] DataPool.write_output_enable(): Exception!")
        self._lock_DAC.release()


    # ==== SCHEDULE HANDLING ================================================

    def read_schedule_segment(self, segment_id: int):
        """Thread-safely reads a schedule segment from the schedule in the
        datapool, if it exists, and returns it. If it does not exist, it
        returns 0.

        The lock prevents other threads from accessing the schedule whilst it
        is being edited. Useful to prevent hard-to-debug race condition bugs.
        """
        # print("[DEBUG] read_schedule_segment()")

        self._lock_schedule.acquire(timeout=0.001)

        if segment_id > len(self.schedule) - 1:
            print(f"[WARNING] get_schedule_segment(): Tried to set " +
                  f"segment {segment_id}, but this is out of bounds for a " +
                  f"schedule of length {len(self.schedule)}!")
            return 0

        segment = self.schedule[segment_id]
        self._lock_schedule.release()
        return segment

    def write_schedule_segment(self, segment: list):
        """Thread-safely writes a schedule segment into the schedule, by
        consulting the ID of the segment (first entry, segment[0]) and writing
        it to the schedule object.

        The lock prevents other threads from accessing the schedule whilst it
        is being edited. Useful to prevent hard-to-debug race condition bugs.
        """
        # print("[DEBUG] write_schedule_segment()")

        self._lock_schedule.acquire(timeout=0.001)
        self.schedule[segment[0]] = segment
        self.schedule_hash = ""     # Modifying the schedule voids the hash.
        self._lock_schedule.release()

    def initialize_schedule(self):
        """Thread-safely writes the default init schedule to the datapool.
        Returns a 1 for remote confirmation purposes.

        The lock prevents other threads from accessing the schedule whilst it
        is being edited. Useful to prevent hard-to-debug race condition bugs.
        """
        # print("[DEBUG] initialize_schedule()")
```

```
662
663         self._lock_schedule.acquire(timeout=0.001)
664
665         self.schedule_hash = ""     # Modifying the schedule voids the hash.
666
667         self.schedule = [[0, 0, 0., 0., 0., 0.], ]
668         self.schedule_name = "init"
669         self.schedule_duration = 0.0
670
671         self._lock_schedule.release()
672
673         # print(f"[DEBUG] HASH INIT {self.schedule_hash}")
674         return 1
675
676     def allocate_schedule(self, name: str, n_seg: int, duration: float):
677         """Thread-safely allocates an empty schedule of a defined size. This
678         schedule then has to be filled with segments by using
679         write_schedule_segment(). The schedule name and duration are also
680         written to the datapool. Returns a 1 for remote confirmation purposes.
681
682         Note that the dtypes of the empty schedule values will all be float,
683         where some have to be int, but write_schedule_segment() addresses this.
684
685         The lock prevents other threads from accessing the schedule whilst it
686         is being edited. Useful to prevent hard-to-debug race condition bugs.
687         """
688         # print("[DEBUG] allocate_schedule()")
689
690         self._lock_schedule.acquire(timeout=0.001)
691
692         self.schedule_hash = ""     # Modifying the schedule voids the hash.
693
694         self.schedule_name = name
695         self.schedule_duration = duration
696         # self.schedule = [[0, 0, 0., 0., 0., 0.], ]*n_seg  # Risky implementation
697         self.schedule = self.init_buffer(n_seg, 6)
698
699         self._lock_schedule.release()
700
701         # print(f"[DEBUG] HASH ALLOC {self.schedule_hash}")
702
703         return 1
704
705     def read_schedule_hash(self, generate_hash=True):
706         """Thread-safely reads the value of self.schedule_hash and returns it.
707         If the hash is empty, generate it, but only when allowed.
708
709         The reason for potentially disallowing it is that hashing takes
710         significant time and computational resources, and since accessing it
711         here thread-locks the object, it means that it cannot be accessed for
712         schedule playback, which would result in execution lag.
713
714         The lock prevents other threads from accessing the schedule whilst it
715         is being edited. Useful to prevent hard-to-debug race condition bugs.
716
717         [DEV NOTE] As an extra safety/performance measure, the schedule could
718         be deep-copied so that the thread lock can be lifted while the hashing
719         function works on the schedule copy. However, this will take twice the
720         memory resources, and for large schedules, where you expect the hashing
721         to take a long time, available memory will already be limited. So this
722         idea has been shelved for now, but may be revisited in the future.
723         """
724         # print("[DEBUG] read_schedule_hash()")
725
726         self._lock_schedule.acquire(timeout=30.000)
727
728         schedule_hash = self.schedule_hash
729
730         if schedule_hash == "" and generate_hash is True:
731             t0 = time()
732             schedule_hash = blake2b(
```

```
                array(self.schedule).tobytes(), digest_size=8
            ).hexdigest()
            print(f"[BLAKE2B] Hash generated in {int((time()-t0)*1E6)} \u03bcs")
            self.schedule_hash = schedule_hash

        self._lock_schedule.release()
        return schedule_hash

    def read_schedule_info(self, generate_hash=True):
        # print("[DEBUG] read_schedule_info()")

        self._lock_schedule.acquire(timeout=0.001)

        name = self.schedule_name
        length = len(self.schedule)
        duration = self.schedule_duration

        self._lock_schedule.release()

        # print(f"[DEBUG] HASH SI 1 {self.schedule_hash}")

        schedule_hash = self.read_schedule_hash(generate_hash=generate_hash)

        # print(f"[DEBUG] HASH SI 2 {self.schedule_hash}")

        return name, length, duration, schedule_hash

    def print_schedule_info(self, max_entries=16):
        self._lock_schedule.acquire(timeout=0.001)

        print(" ==== SCHEDULE INFO ==== ")
        print("Name:    ", self.schedule_name)
        print("n_seg:   ", len(self.schedule))
        print("Duration:", self.schedule_duration)
        print("Hash", self.schedule_hash)

        if len(self.schedule) == 0:
            pass
        elif len(self.schedule) == 1:
            print(" [Preview]")
            print(self.schedule[0])
        elif len(self.schedule) == 2:
            print(" [Preview]")
            print(self.schedule[0])
            print(self.schedule[1])
        else:
            print(" [Preview]")
            n_prints = min(len(self.schedule), max_entries)
            for i_seg in range(0, int(n_prints/2)):
                print(self.schedule[i_seg])
            if len(self.schedule) > max_entries:
                print("...          ...")
            for i_seg in range(-int(n_prints/2), 0):
                print(self.schedule[i_seg])

        self._lock_schedule.release()
        return 1


    # ==== SCHEDULE PLAYBACK ================================================

    def set_play_mode(self, play_mode_on: bool):
        """Activates or deactivates play_mode."""
        # print("[DEBUG] set_play_mode():", play_mode_on, type(play_mode_on))
        if play_mode_on is True:
            """Activates play mode, which primes the server to start playing
            back the current schedule. The idea is that everything is set up
            such that the moment self.play is set to True, playback can
            commence immediately. This includes priming hardware such as power
            supplies to the first step of the schedule, such that no slew
            transients need to happen the moment that playback is started.
```

```
804
805             To this end, this function does the following:
806                 1. self.play_mode will be set to True, resulting in adjusted
807                     event loops of the ADC and DAC handling threads.
808                 2. The internal play tracking parameters will be initialized to the
809                     first step of the current schedule.
810                 3. The hardware is instructed to output the first schedule step.
811             """
812             # print("[DEBUG] activate_play_mode")
813
814             self.play_mode = True
815             self.play = False
816
817             self.n_steps = len(self.schedule)
818             self.write_buffer(self.i_step, 0)
819
820             self.t_play = time()
821             self.t_current = self.schedule[self.i_step[0]][2]
822             self.t_next = self.schedule[self.i_step[0]+1][2]
823
824             # Set hardware to first schedule step
825             self.write_Bc(self.schedule[datapool.i_step[0]][3:6])
826
827             return 1
828
829         else:
830             """
831             Deactivates play mode, which deprimes the server for schedule playback.
832
833             This covers:
834                 1. self.play_mode will be set to False, resulting in adjusted
835                     event loops of the ADC and DAC handling threads.
836                 2. self.play is also set to False to stop any playback.
837                 3. The hardware is returned to a reset/idle state.
838             """
839             # print("[DEBUG] deactivate_play_mode")
840
841             self.play_mode = False
842             self.play = False
843
844             # Reset hardware to zero
845             self.write_Bc([0., 0., 0.])
846
847             return 0
848
849     def set_play(self, play: bool):
850         """
851         Starts or stops schedule playback. Pausing is not implemented.
852
853         When playback is started, self.play is set to True and the unix start
854         time is recorded.
855
856         When playback is stopped, the playback parameters are reset to the
857         beginning of the schedule. The hardware is also set to the first
858         schedule point.
859         """
860         # print("[DEBUG] set_play:", play)
861
862         if play is True:
863             self.t_play = time()
864             self.play = True
865             return 1
866         else:
867             self.play = False
868             self.t_current = 0.0
869             self.write_buffer(self.i_step, 0)
870             self.t_next = self.schedule[self.i_step[0]][2]
871
872             # Reset hardware to first schedule step
873             self.write_Bc(self.schedule[datapool.i_step[0]][3:6])
874
```

```
            return 0


# Server object
class ThreadedTCPServer(ThreadingMixIn, TCPServer):
    """ This is a subclassed TCP Server class with threading enabled.

    Its constructor has been overloaded to ensure that the datapool object
    can be passed to it, and so that extra functionality can be added.
    """
    def __init__(self, host, handler, datapool):

        self.datapool = datapool
        self.server_tstart = time()

        super().__init__(host, handler)

    def uptime(self):
        """Returns the uptime of the ThreadedTCPServer instance in [s]"""
        return time() - self.server_tstart


# Thread handler object
class ThreadedTCPRequestHandler(BaseRequestHandler):
    """ This is a subclassed RequestHandler class with threading enabled.

    A unique instance of this class is created every time a client connects
    to the ThreadedTCPServer, and is destroyed when this connection is
    terminated. This server setup supports multiple clients, which all
    interface to the server through their own assigned instance of
    ThreadedTCPRequestHandler.

    Because the DataPool object implements its own thread locks, it should
    also be possible for the clients to safely interact with the datapool, so
    long as the thread-safe read and write functions are used.

    Its behaviour is as follows: It will run setup() once, after which it will
    loop handle() forever. Within handle(), the more specialized function
    command_handle() may be invoked. When the handle() loop is broken, finish()
    will run once, after which the class instance will terminate.
    """

    def setup(self):
        """Overloaded from base class. Runs once before handle() loops."""
        self.socket_tstart = time()
        print(f"Client {self.client_address[0]}:{self.client_address[1]} connected.")

        self.v = config["verbosity"]

    def handle(self):
        """
        This is the default handling routine for any data packets sent to the
        server from any client. Its behaviour is to while-loop forever until
        interrupted.

        Every server-client interaction is blocking for that client. This
        means that every incoming packet warrants a response packet from the
        server before a new packet can be sent from the client. A packet from
        one client will not block a packet coming from another client, as it
        will be interacting with a different instance of the handler running
        in its own thread.

        The first thing that is done when a packet arrives is its type is
        identified. This involves the reading of the first byte character
        through the packet_type() function in the SCC codec, which can be done
        very quickly. Based on the type, a different subroutine is chosen to
        handle the packet. For x-packets, whose contents may vary a lot, the
        handling routines are bundled in the command_handle() function.

        Every subroutine for handling a particular package type will result in
        the creation "package_out", which holds the package that will be sent
```

```
        in response. Empty packets will not be sent.
        """
        print("handle()")
        while True:
            packet_out = None
            packet_in = self.request.recv(256)
            # t0 = time()  # [TIMING]
            if packet_in == b"":
                break
            type_id = codec.packet_type(packet_in)
            # t1 = time()  # [TIMING]

            if self.v >= 3:
                print("[DEBUG] packet_in:", packet_in)

            if type_id == "b":
                """b-packets received from a client are considered a request
                for Bm, and always have their contents discarded. The server
                sends back a b-packet with a timestamp and the current value
                of Bm."""
                if self.v >= 2:
                    print("[DEBUG] Detected b-packet")
                # print("[DEBUG] READ_BM()", self.server.datapool.read_Bm())
                packet_out = codec.encode_bpacket(
                    *self.server.datapool.read_Bm()
                )


            elif type_id == "c":
                """c-packets contain power supply instructions in the form of
                Bc: a desired flux density value for all three axes. Return
                an m-packet with 1 if successfully passed on to the Datapool,
                or -1 if it was unsuccessful."""
                if self.v >= 2:
                    print("[DEBUG] Detected c-packet")
                Bc = codec.decode_cpacket(packet_in)
                try:
                    # Safety measure that prevents manual control when play
                    # mode is active.
                    if self.server.datapool.play_mode is True:
                        packet_out = codec.encode_mpacket("-2")
                    else:
                        self.server.datapool.write_Bc(Bc)
                        if self.v >= 4:
                            print("[DEBUG] Bc written to datapool:", Bc, type(Bc))
                            print("[DEBUG] CHECK datapool.Bc:", self.server.datapool.Bc)
                        packet_out = codec.encode_mpacket("1")
                except:  # noqa
                    packet_out = codec.encode_mpacket("-1")


            elif type_id == "e":
                """The purpose of e-packets is to require as little processing
                as possible, and they will be comstantly coming in from the
                client, so they are instantly decoded, re-encoded, and echoed
                back without any additional action."""
                if self.v >= 2:
                    print("[DEBUG] Detected e-packet")
                packet_out = codec.encode_epacket(
                    codec.decode_epacket(packet_in)
                )


            elif type_id == "m":
                """m-packets comprise simple string messages. They are always
                displayed in the server terminal, and a simple acknowledgement
                m-packet is sent in response."""
                if self.v >= 2:
                    print("[DEBUG] Detected m-packet")

                msg = codec.decode_mpacket(packet_in)
```

```
                print(f"[{self.client_address[0]}:{self.client_address[1]}] {msg}")
                packet_out = codec.encode_mpacket("1")  # Send 1 as confirmation


            elif type_id == "s":
                """s-packets are schedule segments and usually sent in large
                batches. The corresponding entry in the schedule located in
                the Datapool is updated with the information in the s-packet.
                The segment number is sent back as a acknowledgement and
                verification."""
                if self.v >= 2:
                    print("[DEBUG] Detected s-packet")

                segment = codec.decode_spacket(packet_in)
                self.server.datapool.write_schedule_segment(segment)
                # Send segment number back as a verification
                packet_out = codec.encode_mpacket(str(segment[0]))


            elif type_id == "t":
                """t-packets received from a client are considered a request
                for telemetry, and always have their contents discarded. The
                server responds with a t-packet with a timestamp and the
                latest telemetry values as specified by the SCC codec."""
                if self.v >= 2:
                    print("[DEBUG] Detected t-packet")
                packet_out = codec.encode_tpacket(
                    *self.server.datapool.read_telemetry()
                )


            elif type_id == "x":
                """x-packets always contain a command, the number of additional
                arguments, as well as these additional arguments. This routine
                simply identifies the function name and its arguments, and
                delegates these to command_handle() for further processing.
                When done, command_handle() returns the contents of packet_out.
                """
                if self.v >= 2:
                    print("[DEBUG] Detected x-packet")
                fname, args = codec.decode_xpacket(packet_in)
                if self.v >= 2:
                    print(f"[DEBUG] {fname}({args})")
                packet_out = self.command_handle(fname, args)


            else:
                """Raise exception when encountering an unrecognised packet
                type."""
                raise ValueError(f"Encountered uninterpretable type_id '{type_id}' in
                    received packet.")

            if self.v >= 3:
                print("[DEBUG] packet_out:", packet_out)

            # t2 = time()  # [TIMING]

            # Send packet_out:
            if packet_out is not None:
                self.request.sendall(packet_out)

            # t3 = time()  # [TIMING]
            # print(f"Sent {codec.packet_type(packet_out)}-packet. Time: {int((t1-t0)*1E6)},
                {int((t2-t1)*1E6)}, {int((t3-t2)*1E6)} \u03bcs")  # [TIMING]


    def command_handle(self, fname, args):
        """
        This function handles the interpretation of all x-packets sent to the
        server, executes the corresponding actions, and composes the
        appropriate properties of the response packet.
```

```
        Here is a list of all functions the server currently supports:

        get_aux_adc
        get_aux_dac / set_aux_dac
        get_Bc
        get_Br / set_Br
        get output_enable / set output_enable
        get_schedule_info
        get_server_uptime
        get_socket_info
        get_serveropt_mutate_Bm / set_serveropt_mutate_Bm
        get_V_board

        """
        packet_out = None

        if fname == "get_aux_adc":
            packet_out = codec.encode_mpacket(
                str(self.server.datapool.read_aux_adc()))


        elif fname == "get_aux_dac":
            dac = self.server.datapool.aux_dac[0]               # Not thread-safe
            packet_out = codec.encode_mpacket(
                f"{dac[0]},{dac[1]},{dac[2]},{dac[3]},{dac[4]},{dac[5]}"
            )

        elif fname == "set_aux_dac":
            dac_str = [args[0], args[1], args[2], args[3], args[4], args[5]]
            dac_vals = [float(item) for item in dac_str]
            self.server.datapool.write_buffer(
                self.server.datapool.aux_dac, dac_vals
            )                                                   # Not thread-safe
            dac = self.server.datapool.aux_dac[0]               # Not thread-safe
            packet_out = codec.encode_mpacket(
                f"{dac[0]},{dac[1]},{dac[2]},{dac[3]},{dac[4]},{dac[5]}"
            )


        # Requests the Bc value:
        elif fname == "get_Bc":
            packet_out = codec.encode_cpacket(self.server.datapool.read_Bc())


        # Requests the Br value:
        elif fname == "get_Br":
            packet_out = codec.encode_mpacket("{},{},{}".format(
                *self.server.datapool.read_Br()))
        # Sets the Br value:
        elif fname == "set_Br":
            Br = [args[0], args[1], args[2]]
            self.server.datapool.write_Br(Br)
            if self.v >= 4:
                print("[DEBUG] Br written to datapool:", Br, type(Br))
                print("[DEBUG] CHECK datapool.Br:", self.server.datapool.Br)
            packet_out = codec.encode_mpacket("1")


        elif fname == "get_params_VB":
            p = self.server.datapool.params_tf_VB           # Not thread-safe
            packet_out = codec.encode_mpacket(
                f"{p['x'][0]},{p['x'][1]},"
                + f"{p['y'][0]},{p['y'][1]},"
                + f"{p['z'][0]},{p['z'][1]},"
            )

        elif fname == "set_params_VB":
            count = 0
            if args[0] is not False:
                self.server.datapool.params_tf_VB["x"][0] = args[0]     # Not thread-safe
```

```
1157                count += 1
1158            if args[1] is not False:
1159                self.server.datapool.params_tf_VB["x"][1] = args[1]     # Not thread-safe
1160                count += 1
1161            if args[2] is not False:
1162                self.server.datapool.params_tf_VB["y"][0] = args[2]     # Not thread-safe
1163                count += 1
1164            if args[3] is not False:
1165                self.server.datapool.params_tf_VB["y"][1] = args[3]     # Not thread-safe
1166                count += 1
1167            if args[4] is not False:
1168                self.server.datapool.params_tf_VB["z"][0] = args[4]     # Not thread-safe
1169                count += 1
1170            if args[5] is not False:
1171                self.server.datapool.params_tf_VB["z"][1] = args[5]     # Not thread-safe
1172                count += 1
1173
1174            packet_out = codec.encode_mpacket(str(count))
1175
1176
1177        elif fname == "get_output_enable":
1178            packet_out = codec.encode_mpacket(
1179                str(int(self.server.datapool.read_output_enable()))
1180            )
1181
1182        elif fname == "set_output_enable":
1183            bool_val = bool(int(args[0]))
1184            self.server.datapool.write_output_enable(bool_val)
1185            packet_out = codec.encode_mpacket(str(int(bool_val)))
1186
1187        # Requests the server uptime:
1188        elif fname == "get_server_uptime":
1189            packet_out = codec.encode_mpacket(str(self.server.uptime()))
1190
1191        elif fname == "get_socket_info":
1192            """Return an m-packet with some information about the client from
1193            the perspective of the server:
1194            1. The time for which the client socket has been active
1195            2. The client address
1196            3. The client port
1197            This information is packaged as a csv string.
1198            """
1199            uptime = time()-self.socket_tstart
1200            packet_out = codec.encode_mpacket(
1201                f"{uptime},{self.client_address[0]},{self.client_address[1]}"
1202            )
1203
1204
1205        # ==== Playback functions ============================================
1206        elif fname == "set_serveropt_mutate_Bm":
1207            self.server.datapool.serveropt_mutate_Bm = args[0]              # Not thread-safe
1208            packet_out = codec.encode_mpacket(
1209                str(int(self.server.datapool.serveropt_mutate_Bm)))         # Not thread-safe
1210
1211        elif fname == "get_serveropt_mutate_Bm":
1212            packet_out = codec.encode_mpacket(
1213                str(int(self.server.datapool.serveropt_mutate_Bm)))         # Not thread-safe
1214
1215
1216        elif fname == "set_serveropt_inject_Bm":
1217            self.server.datapool.serveropt_inject_Bm = args[0]              # Not thread-safe
1218            packet_out = codec.encode_mpacket(
1219                str(int(self.server.datapool.serveropt_inject_Bm)))         # Not thread-safe
1220
1221        elif fname == "get_serveropt_inject_Bm":
1222            packet_out = codec.encode_mpacket(
1223                str(int(self.server.datapool.serveropt_inject_Bm)))          # Not thread-
                    safe
1224
1225
1226        # ==== Playback functions ============================================
```

```
1227        elif fname == "get_play_info":
1228            packet_out = codec.encode_mpacket(
1229                f"{int(self.server.datapool.play_mode)}," +     # Not thread-safe
1230                f"{int(self.server.datapool.play)}," +          # Not thread-safe
1231                f"{int(self.server.datapool.play_looping)}," +  # Not thread-safe
1232                f"{self.server.datapool.n_steps}," +            # Not thread-safe
1233                f"{self.server.datapool.i_step[0]}," +             # Not thread-safe
1234                f"{self.server.datapool.t_play}," +             # Not thread-safe
1235                f"{self.server.datapool.t_current}," +          # Not thread-safe
1236                f"{self.server.datapool.t_next}"                # Not thread-safe
1237            )
1238
1239        elif fname == "set_play_mode":
1240            packet_out = codec.encode_mpacket(
1241                str(self.server.datapool.set_play_mode(args[0])))       # Not thread-safe
1242
1243        elif fname == "set_play":
1244            packet_out = codec.encode_mpacket(
1245                str(self.server.datapool.set_play(args[0])))            # Not thread-safe
1246
1247        elif fname == "set_play_looping":
1248            self.server.datapool.play_looping = args[0]                 # Not thread-safe
1249            packet_out = codec.encode_mpacket(
1250                str(int(self.server.datapool.play_looping)))            # Not thread-safe
1251
1252        # ==== Schedule functions ================================
1253
1254        # Prints info about the schedule into the terminal
1255        elif fname == "print_schedule_info":
1256            confirm = self.server.datapool.print_schedule_info(max_entries=args[0])
1257            packet_out = codec.encode_mpacket(str(confirm))
1258
1259        # Requests the schedule name, length, and duration as csv string
1260        elif fname == "get_schedule_info":
1261            name, length, duration, schedule_hash = \
1262                self.server.datapool.read_schedule_info(
1263                    generate_hash=bool(int(args[0]))
1264                )
1265            packet_out = codec.encode_mpacket(
1266                f"{name},{length},{duration},{schedule_hash}"
1267            )
1268
1269        elif fname == "get_schedule_segment":
1270            seg = self.server.datapool.read_schedule_segment(args[0])
1271            packet_out = codec.encode_spacket(
1272                seg[0], seg[1], seg[2], seg[3], seg[4], seg[5]
1273            )
1274
1275        # Initialize the schedule (reset)
1276        elif fname == "initialize_schedule":
1277            confirm = self.server.datapool.initialize_schedule()
1278            packet_out = codec.encode_mpacket(str(confirm))
1279
1280        # Allocate an empty schedule (args: name: str, n_seg: int, duration: float)
1281        elif fname == "allocate_schedule":
1282            confirm = self.server.datapool.allocate_schedule(args[0], args[1], args[2])
1283            packet_out = codec.encode_mpacket(str(confirm))
1284
1285
1286        elif fname == "get_V_board":
1287            packet_out = codec.encode_mpacket(
1288                str(self.server.datapool.read_V_board()))
1289
1290        # # Requests the value of play mode (False indicates 'manual mode')
1291        # elif fname == "get_play_mode":
1292        #     packet_out = codec.encode_mpacket(str(self.server.datapool.get_play_mode()))
1293        #
1294        # # Requests the value of play status (False indicates 'manual mode')
1295        # elif fname == "get_play_status":
1296        #     packet_out = codec.encode_mpacket(self.server.datapool.get_play_status())
1297
```

```
1298
1299            else:
1300                raise ValueError(f"Function name '{fname}' not recognised!")
1301
1302            return packet_out
1303
1304
1305        def finish(self):
1306            """Overloaded from base class. Runs once after handle() loop ends."""
1307            print(f"Client {self.client_address[0]}:{self.client_address[1]} disconnected.")
1308
1309
1310    if __name__ == "__main__":
1311        t0 = time()
1312
1313        # Initialize common DataPool object
1314        datapool = DataPool()
1315
1316        # Server object
1317        # HOST = ("127.0.0.1", 7777)  # TODO remove
1318        HOST = (config["SERVER_ADDRESS"], config["SERVER_PORT"])
1319
1320        server = ThreadedTCPServer(HOST, ThreadedTCPRequestHandler, datapool)
1321
1322        # When set to True, this makes all child threads of the server thread
1323        # daemonic, forcing them to terminate when main thread terminates.
1324        # It is recommended to keep this on False, because operation of the
1325        # write_DAC_thread will be safer if it is non-daemonic.
1326        server.daemon_threads = False
1327
1328        # Place server in own thread
1329        server_thread = Thread(target=server.serve_forever)
1330
1331        # Make server thread daemonic, so it too will terminate upon main thread
1332        # termination
1333        server_thread.daemon = True
1334
1335        # Start server thread
1336        server_thread.start()
1337
1338        # Set up thread for control
1339        thread_control = Thread(
1340            name="Control Thread",
1341            target=threaded_control,
1342            args=(datapool,),
1343            daemon=True)
1344        thread_control.start()
1345
1346        # Set up thread for continuously polling the ADC, and writing it to the
1347        # server datapool.
1348        thread_read_ADC = Thread(
1349            name="Read ADC Thread",
1350            target=threaded_read_ADC,
1351            args=(datapool,),
1352            daemon=True)
1353        thread_read_ADC.start()
1354
1355        # Set up thread that finds when a change in control parameters occurs, and
1356        # when it occurs, ensures that it is written to the DAC.
1357        thread_write_DAC = Thread(
1358            name="Write DAC Thread",
1359            target=threaded_write_DAC,
1360            args=(datapool,),
1361            daemon=False)  # Not set as daemon to ensure that it finishes properly
1362        thread_write_DAC.start()
1363
1364        print(f"Server is up. Time elapsed: {round((time()-t0)*1E3, 3)} ms. Active threads: {
            active_count()}")
1365
1366
1367        # ==== Main loop of server ================================================
```

```
    while True:
        try:
            # Do something useful
            sleep(0.001)
            pass

        except:  # noqa
            break


    # ==== Server shutdown ================================================

    # Gracefully terminate control threads
    print("Shutting down - finishing threads.")
    datapool.kill_threaded_read_ADC = True
    datapool.kill_threaded_write_DAC = True
    datapool.kill_threaded_control = True

    ttest = time()
    thread_read_ADC.join(timeout=1.0)
    print(f"Shut down read_ADC thread in {round((time()-ttest)*1E3, 3)} ms")

    ttest = time()
    thread_write_DAC.join(timeout=5.0)
    print(f"Shut down write_DAC thread in {round((time()-ttest)*1E3, 3)} ms")

    ttest = time()
    thread_control.join(timeout=1.0)
    print(f"Shut down control thread in {round((time()-ttest)*1E3, 3)} ms")

    # For safety: set power supplies to zero, separately from apply_Bc_thread
    print(f"Resetting Bc just in case")
    hardware_shutdown(datapool)

    # Server termination
    total_uptime = server.uptime()
    server.shutdown()
    server.server_close()
    print(f"Server is closed. Total uptime: {round(total_uptime, 3)} s")
```

"Helmholtz cage server config file - /helmholtz_cage_toolkit/server/server_config.py"

```
server_config = {
    # ==== General settings ====
    "SERVER_ADDRESS": "127.0.0.1",
    "SERVER_PORT": 7777,

    # ==== Thread settings ====
    "threaded_read_ADC_rate": 8,   # S/s
    "threaded_write_DAC_rate": 8,  # S/s
    "threaded_control_rate": 100, # Hz

    "internal_buffer_size": 5,
    "verbosity": 4,
    # "verbosity_printtimestamp": True,

    # ==== ADC settings ====
    "adc_pollrate": 30,     # S/s TODO DEPRECATED?

    # ==== DAC settings ====
    "vmax_dac": 5.0,        # V

    # ==== Settings for Bm mutation ====
    "mutate_Bm": False,
    "params_mutate": [
        [100, 100, 100],    # central value
        [0.1, 0.1, 0.1],    # mutation scale
        [0.5, 0.5, 0.5],    # fence strength
    ],

    # ==== Settings for Bm injection ====
```

```
    "inject_Bm": False,


    # ==== Power supply settings ====
    "vmax_supply": 30.0,     # V
    "imax_supply": 5.0,      # A
    "r_load": 7,             # Ohm
    "v_above": 2,            # V
    "i_above": 0.2,          # A

    "vlevel_pol": 5.0,       # V
    "v_psu_enable": 5.0,     # V

    "I_slew_max": [1000, 1000, 1000],    # [A/s] Maximum allowed current slew rate

    # ==== Playback settings ====
    "default_play_looping": True,

    # Linear regression coefficients for VC transfer function ([b0, b1] -> B_out = b0 + b1*V)
    "params_tf_VB_x": [0, 100],
    "params_tf_VB_y": [0, 100],
    "params_tf_VB_z": [0, 100],

    # Linear regression coefficients for VC transfer function ([a, b] -> v_cc = a*v + b)
    "params_tf_vc_x": [15.05305, -4.80750],
    "params_tf_vc_y": [15.05305, -4.80750],
    "params_tf_vc_z": [15.05305, -4.80750],

    # Linear regression coefficients for CC transfer function ([a, b] -> v_vc = a*i + b)
    "params_tf_cc_x": [2.000974, -0.012608],
    "params_tf_cc_y": [2.000974, -0.012608],
    "params_tf_cc_z": [2.000974, -0.012608],


    # ==== Hardware and cable routing ====
    # Note: Only pin mappings must start with 'pin_', otherwise config will not
    # be interpreted correctly by code elsewhere.
    "pin_adc_channel_imx": 0,
    "pin_adc_channel_imy": 1,
    "pin_adc_channel_imz": 2,

    "pin_adc_board_power": 3,

    "pin_adc_channel_bmx": 4,
    "pin_adc_channel_bmy": 5,
    "pin_adc_channel_bmz": 6,

    "pin_adc_aux1": 7,

    "pin_dac_supply_x_vcc": 3,
    "pin_dac_supply_x_vvc": 1,
    "pin_dac_supply_x_pol": 0,

    "pin_dac_supply_y_vcc": 7,
    "pin_dac_supply_y_vvc": 5,
    "pin_dac_supply_y_pol": 4,

    "pin_dac_supply_z_vcc": 11,
    "pin_dac_supply_z_vvc": 9,
    "pin_dac_supply_z_pol": 8,

    "pin_dac_psu_enable": 13, # TODO Rename to pin_dac_output_enable

    "pin_dac_aux1": 2,
    "pin_dac_aux2": 6,
    "pin_dac_aux3": 10,
    "pin_dac_aux4": 12,
    "pin_dac_aux5": 14,
    "pin_dac_aux6": 15,
}
```

H

Details of the orbital validation tests

This appendix contains more detailed information about the orbital validation tests described in subsection 13.6.3. A total of eight tests were carried out, which can be found in Table H.1. Each test is then described in its own section, mostly using figures. As a note beforehand: Before each test, the local disturbance vector was measured and cancelled. While this data does not show up in the tests, it may contribute to slight variations between tests. The tests carried out on 21-11-2025 use a different calibration as 25-11-2025, as the auto-calibrate script was run at the start of each day.

Test	Date	Description	Relevant files
Orbit1	21-11-2025	Test orbit used throughout the project, identical to the one from subsection 13.6.2. Satellite has mild body rates on two axes, and orbits at 350 km with a 50° inclination and a 0.25 eccentricity.	`TS_orbit1_2_202511211327_FIX.dat`
Orbit2	21-11-2025	A MEO case at 4600 km with an eccentricity of 0.2 and an inclination of 54°. At these altitudes, the IGRF yields degraded accuracy. The test case is mainly interesting because the field values are relatively small, the effect of which this tests aims to investigate. The satellite rotates mildly on one body axis.	`TS_orbit2_202511211346.dat`
Orbit5	21-11-2025	Simulates a common LEO CubeSat mission deployed from the ISS. A circular orbit with an altitude of 350 km and a 51.6° inclination. The satellite is nadir-pointing and does not rotate in its inertial reference frame. The test performs 20 orbits at a time speed factor of 100.	`TS_orbit5_202511211512_FIX.dat`
Orbit6	21-11-2025	The same CubeSat mission as the **Orbit5** test, but simulating 4 orbits at a time speed factor of 10. One of the goals of this test is to investigate the effect of the time speed factor on the overall test accuracy.	`TS_orbit6_202511211603_FIX.dat`
Orbit7	25-11-2025	Simulates a mission based on the 6U SpIRIT CubeSat mission [78], nadir-pointing in a circular, sun-synchronous orbit at 97.4° inclination. The orbital parameters assume a pericentre altitude of 415 km and are based on historical TLE data from the mission.	`TS_orbit7_20251125_FIX.dat`
Orbit8	25-11-2025	Simulates a mission based on the intended IT-SPINS mission [16]. It does not simulate the troubled early section of the mission with the ADCS issues, but rather the intended -2 rpm Y_B-axis rotation used for gathering tomography data. Orbital parameters set shortly after the Cygnus NG-15 deployment at 500 km altitude with an inclination of 51.6°.	`TS_orbit8_20251125_FIX.dat`
Orbit9	25-11-2025	Simulates a hypothetical Delfi-PQ orbit scenario where the Z_B-axis of the satellite is nadir-pointing, and the satellite is slowly rotating around this axis. Orbital parameters based on historical TLE data (457 km altitude, 97.4° inclination).	`TS_orbit9_20251125_FIX.dat`
Orbit10	25-11-2025	Simulates the Hubble Space Telescope in a 28.5° inclination orbit at 488.7 km, pointing at a fixed hypothetical celestial target.	`TS_orbit10_20251125_FIX.dat`

H.1. Orbit 1

Test orbit used throughout the project, identical to the one from subsection 13.6.2. Satellite has mild body rates on two axes, and orbits at 350 km with a 50° inclination and a 0.25 eccentricity.

TS_orbit1_2.bsch

```
!BSCH
TS_orbit1_2
generator=orbital
{'orbit_eccentricity': 0.25, 'orbit_inclination': 50, 'orbit_pericentre_altitude':
    350000.0, 'orbit_RAAN': 24, 'orbit_argp': 0.0, 'orbit_ma0': 0.0, 'angle_body_x_0':
     0.0, 'angle_body_y_0': 0.0, 'angle_body_z_0': 0.0, 'rate_body_x': 0.1, '
    rate_body_y': 0.0, 'rate_body_z': -0.1, 'earth_zero_datum': 0, 'date0': 2024.0, '
    n_orbit_subs': 1024, 'n_step': 8192, 'time_speed_factor': 100.0}
{'function': 'none', 'factor': 1}




################################
```

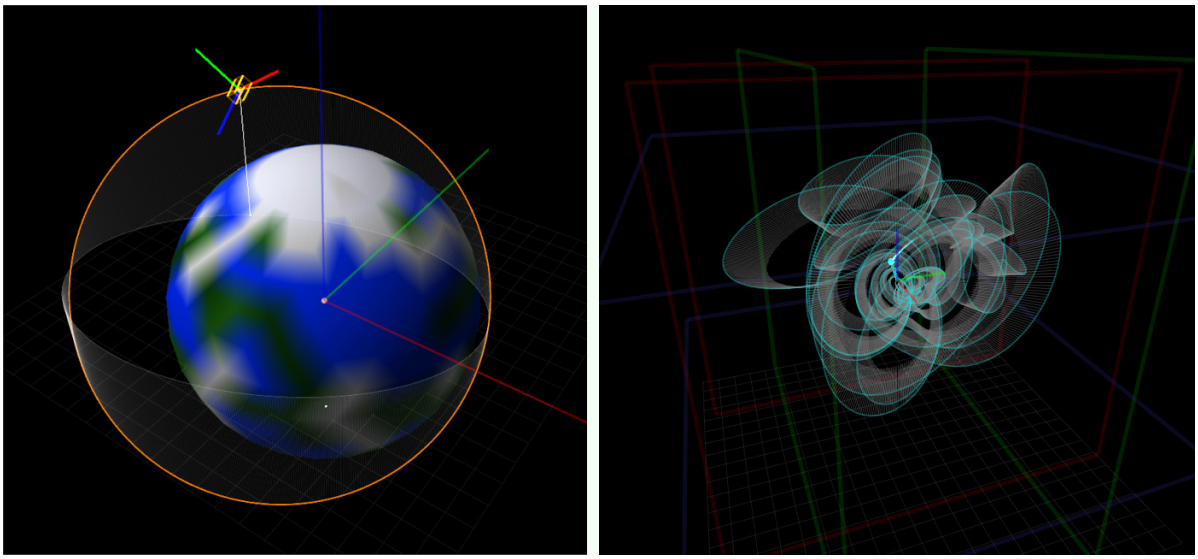


Figure H.1: Left: orbital view in *ECEF* frame. Right: generated magnetic envelope in *B*-frame.

Table H.2: Command errors and angular errors, before and after filtering.

		X [μT]	Y [μT]	Z [μT]	$\mathbf{E_C}$ [μT]	Λ [°]
Unfiltered	Max error	1.428	3.051	2.132	**3.418**	**16.97**
	RMS error	0.746	1.078	0.838	**1.556**	**5.59**
Filtered	Max error	2.058	2.702	1.736	**2.911** (-14.8%)	**6.44** (-62.1%)
	RMS error	0.339	0.361	0.333	**0.597** (-32.7%)	**1.86** (-66.7%)

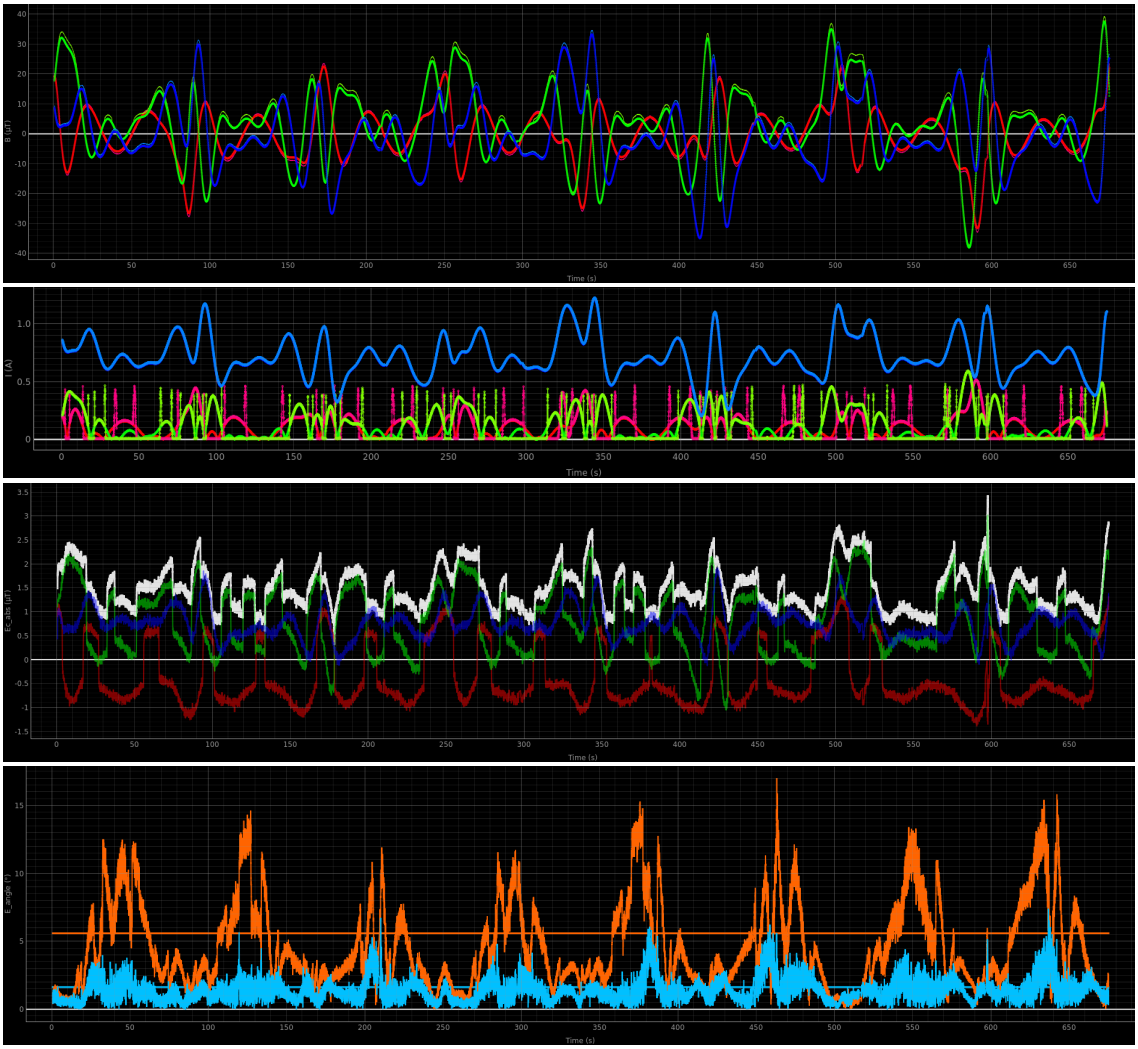


Figure H.2: Test data of `TS_orbit1_2_202511211327_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue).

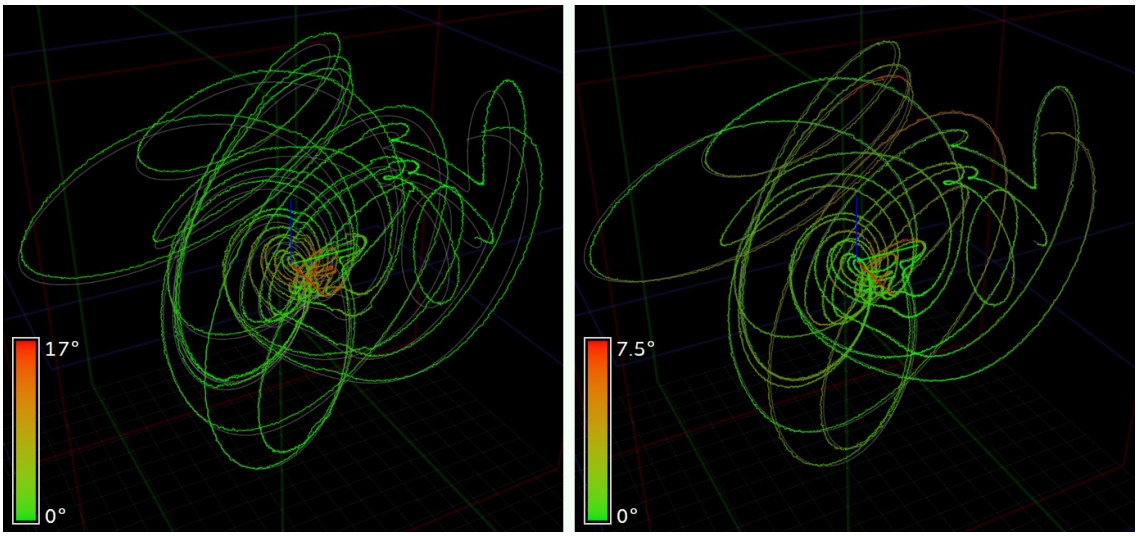


Figure H.3: Left: measured magnetic field manifold. Right: manifold after filtering.

H.2. Orbit 2

A MEO case at 4600 km with an eccentricity of 0.2 and an inclination of 54°. At these altitudes, the IGRF yields degraded accuracy. The test case is mainly interesting because the field values are relatively small, the effect of which this tests aims to investigate. The satellite rotates mildly on one body axis.

TS_orbit2.bsch

```
!BSCH
TS_orbit2
generator=orbital
{'orbit_eccentricity': 0.2, 'orbit_inclination': 54.0, 'orbit_pericentre_altitude':
    4600000, 'orbit_RAAN': 0.0, 'orbit_argp': 0.0, 'orbit_ma0': 0.0, 'angle_body_x_0':
     0.0, 'angle_body_y_0': 90.0, 'angle_body_z_0': 0.0, 'rate_body_x': 0.0, '
    rate_body_y': 0.0, 'rate_body_z': 0.1, 'earth_zero_datum': 0, 'date0': 2024.0, '
    n_orbit_subs': 512, 'n_step': 4096, 'time_speed_factor': 200.0}
{'function': 'none', 'factor': 1}




################################
```

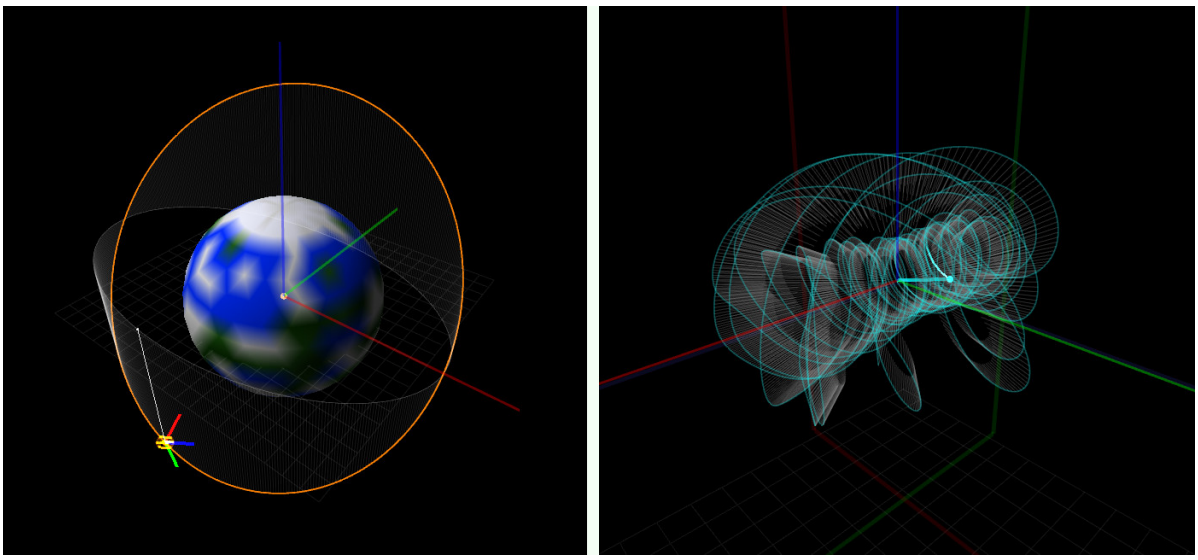


Figure H.4: Left: orbital view in *ECEF* frame. Right: generated magnetic envelope in *B*-frame.

Table H.3: Command errors and angular errors, before and after filtering.

		X [μT]	Y [μT]	Z [μT]	$\mathbf{E_C}$ [μT]	Λ [°]
Unfiltered	Max error	1.095	1.938	1.254	**2.154**	**49.62**
	RMS error	0.611	0.432	0.772	**1.075**	**15.22**
Filtered	Max error	0.813	0.869	0.760	**1.017** (-52.9%)	**16.39** (-67.0%)
	RMS error	0.195	0.203	0.213	**0.353** (-67.2%)	**3.75** (-75.4%)

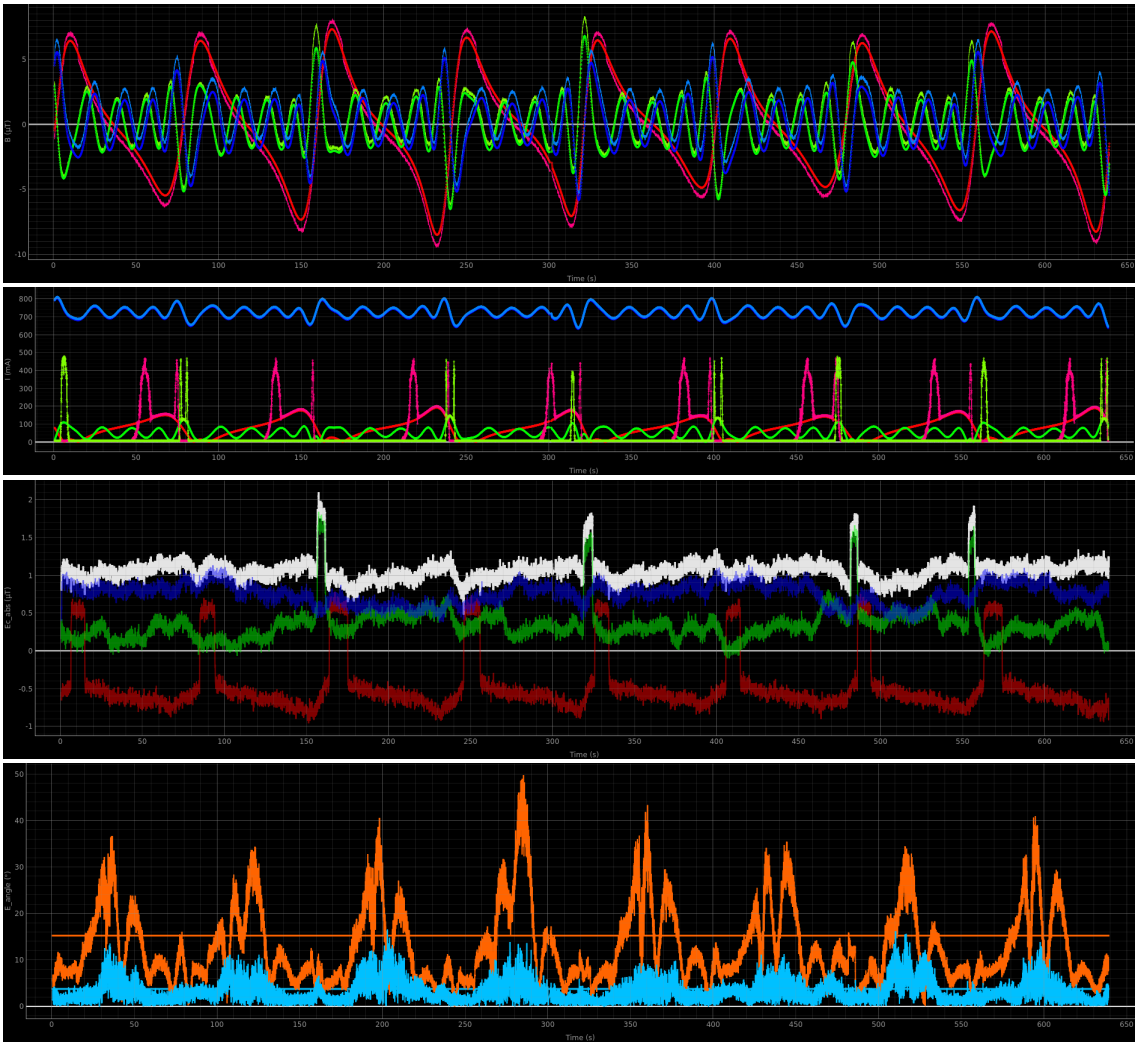


Figure H.5: Test data of `TS_orbit2_202511211346.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue).

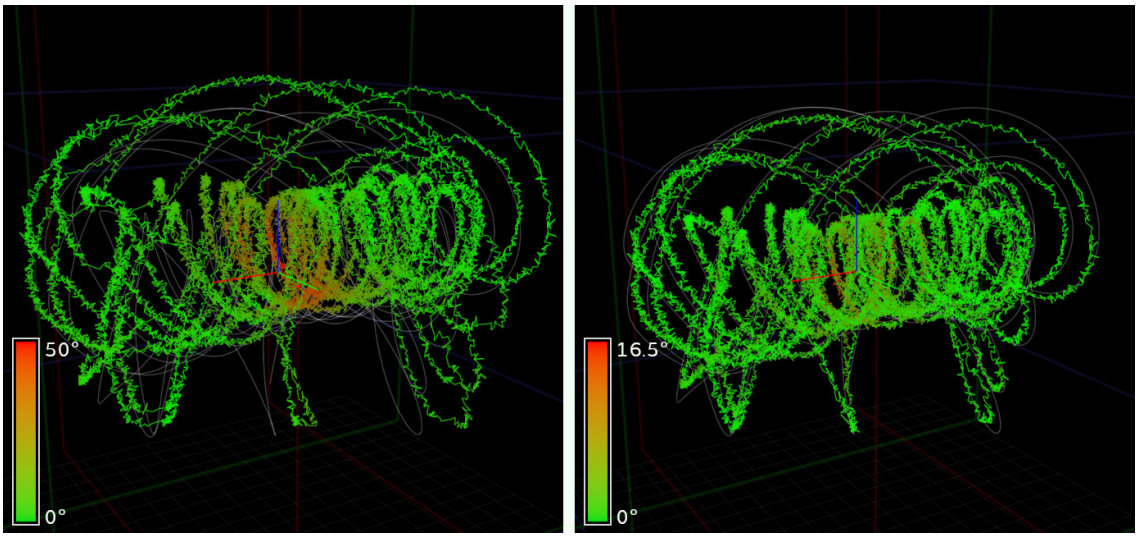


Figure H.6: Left: measured magnetic field manifold. Right: manifold after filtering. Due the reduced overall scale of this schedule, the measurement noise is much more visibly pronounced.

H.3. Orbit 5

Simulates a common LEO CubeSat mission deployed from the ISS. A circular orbit with an altitude of 350 km and a 51.6° inclination. The satellite is nadir-pointing and does not rotate in its inertial reference frame. The test performs 20 orbits at a time speed factor of 100.

TS_orbit5.bsch

```
!BSCH
TS_orbit5
generator=orbital
{'orbit_eccentricity': 0.0004, 'orbit_inclination': 51.6325, '
    orbit_pericentre_altitude': 350000.0, 'orbit_RAAN': 253.957, 'orbit_argp': 150.0,
    'orbit_ma0': 0.0, 'angle_body_x_0': 0.0, 'angle_body_y_0': 90.0, 'angle_body_z_0':
     0.0, 'rate_body_x': 0.0, 'rate_body_y': 0.0, 'rate_body_z': 0.0, '
    earth_zero_datum': 0, 'date0': 2025.8, 'n_orbit_subs': 1024, 'n_step': 20480, '
    time_speed_factor': 100.0}
{'function': 'none', 'factor': 1}



################################
```

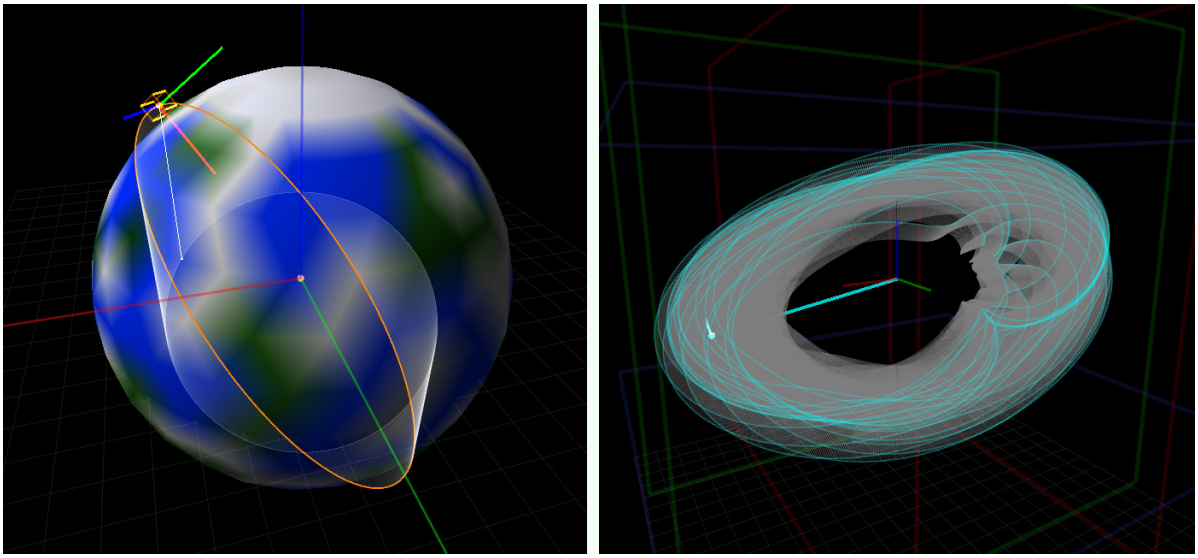


Figure H.7: Left: orbital view in *ECEF* frame. Right: generated magnetic envelope in *B*-frame.

Table H.4: Command errors and angular errors, before and after filtering.

		X [μT]	Y [μT]	Z [μT]	$\mathbf{E_C}$ [μT]	Λ [°]
Unfiltered	Max error	2.010	2.873	1.590	**3.127**	**4.58**
	RMS error	1.180	1.705	0.753	**2.206**	**2.14**
Filtered	Max error	1.308	1.604	0.868	**1.735** (-44.5%)	**2.91** (-36.5%)
	RMS error	0.444	0.656	0.286	**0.842** (-61.8%)	**1.10** (-45.8%)

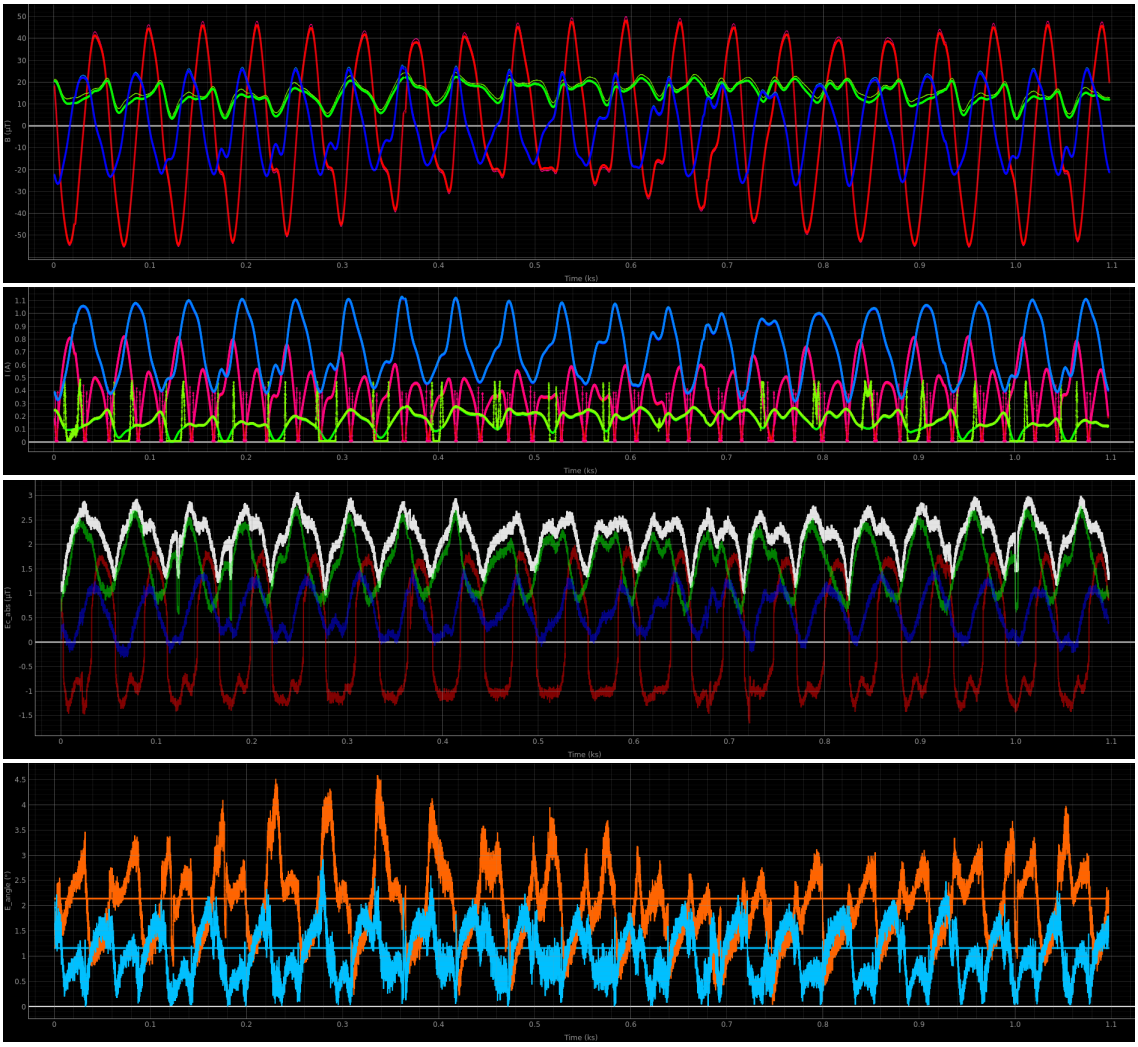


Figure H.8: Test data of `TS_orbit5_202511211512_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue).

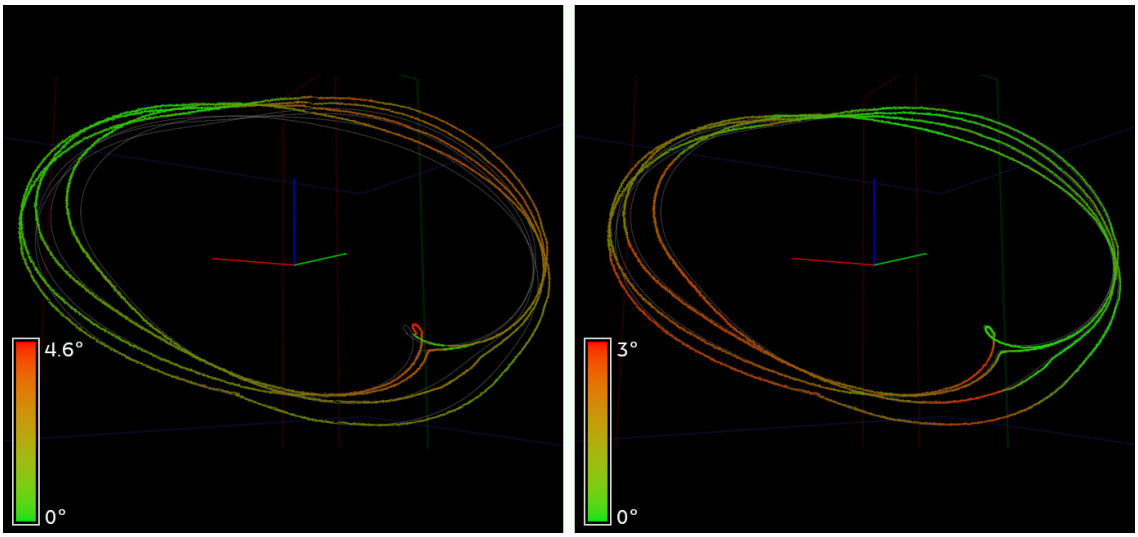


Figure H.9: Left: measured magnetic field manifold. Right: manifold after filtering.

H.4. Orbit 6

The same CubeSat mission as the **Orbit5** test, but simulating 4 orbits at a time speed factor of 10. One of the goals of this test is to investigate the effect of the time speed factor on the overall test accuracy.

TS_orbit6.bsch

```
!BSCH
TS_orbit6
generator=orbital
{'orbit_eccentricity': 0.0004, 'orbit_inclination': 51.6325, '
    orbit_pericentre_altitude': 350000.0, 'orbit_RAAN': 253.957, 'orbit_argp': 150.0,
    'orbit_ma0': 0.0, 'angle_body_x_0': 0.0, 'angle_body_y_0': 90.0, 'angle_body_z_0':
     0.0, 'rate_body_x': 0.0, 'rate_body_y': 0.0, 'rate_body_z': 0.0, '
    earth_zero_datum': 0, 'date0': 2025.8, 'n_orbit_subs': 10240, 'n_step': 40960, '
    time_speed_factor': 10.0}
{'function': 'none', 'factor': 1}
################################
```

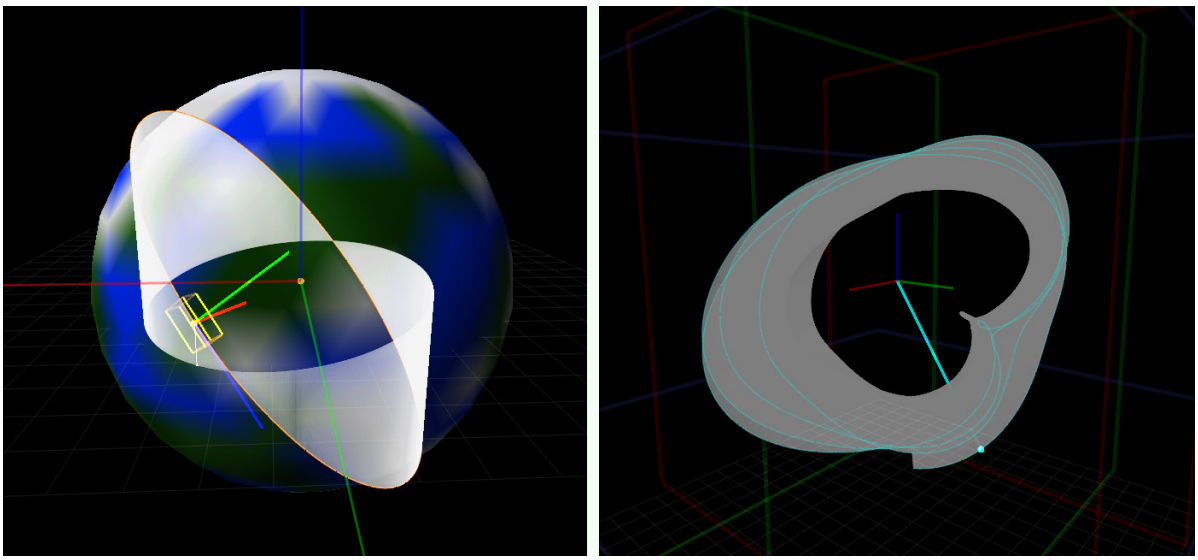


Figure H.10: Left: orbital view in *ECEF* frame. Right: generated magnetic envelope in *B*-frame.

Table H.5: Command errors and angular errors, before and after filtering.

		X [μT]	Y [μT]	Z [μT]	$\mathbf{E_C}$ [μT]	Λ [°]
Unfiltered	Max error	2.253	2.791	1.600	**2.915**	**3.92**
	RMS error	1.216	1.562	0.755	**2.119**	**1.96**
Filtered	Max error	0.929	1.631	0.814	**1.780** (-38.9%)	**2.35** (-40.0%)
	RMS error	0.424	0.708	0.269	**0.868** (-59.0%)	**1.15** (-41.1%)

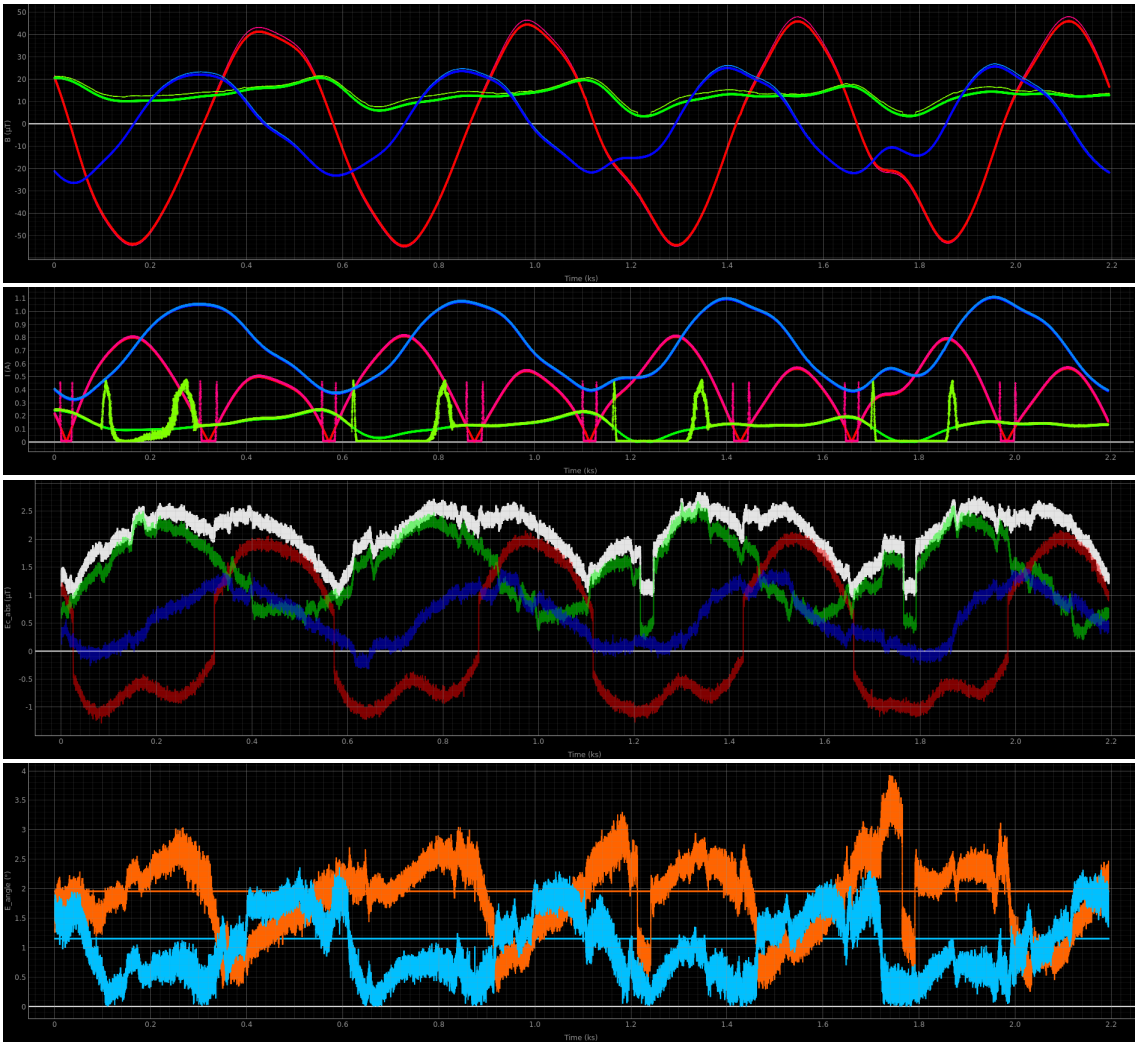


Figure H.11: Test data of `TS_orbit6_202511211603_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue).

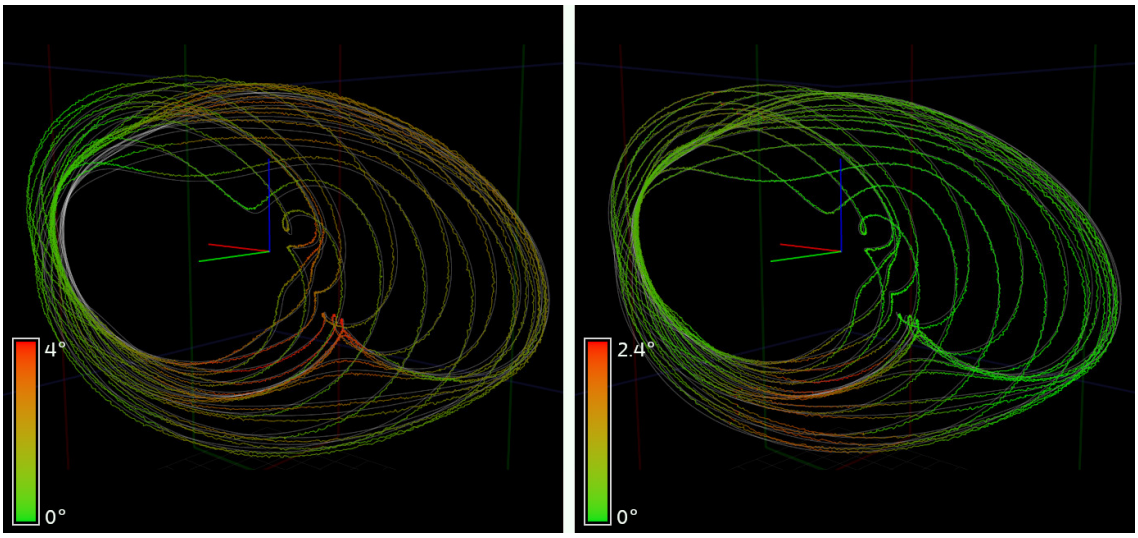


Figure H.12: Left: measured magnetic field manifold. Right: manifold after filtering.

H.5. Orbit 7

Simulates a mission based on the 6U SpIRIT CubeSat mission [78], nadir-pointing in a circular, sun-synchronous orbit at 97.4° inclination. The orbital parameters assume a pericentre altitude of 415 km and are based on historical TLE data from the mission.

TS_orbit7.bsch

```
!BSCH
TS_orbit7
generator=orbital
{'orbit_eccentricity': 0.00065, 'orbit_inclination': 97.3578, '
    orbit_pericentre_altitude': 415000, 'orbit_RAAN': 38.1608, 'orbit_argp': 209.2889,
     'orbit_ma0': 0.0, 'angle_body_x_0': 0.0, 'angle_body_y_0': 0.0, 'angle_body_z_0':
     0.0, 'rate_body_x': 0.0, 'rate_body_y': 0.0, 'rate_body_z': 0.0, '
    earth_zero_datum': 0, 'date0': 2024.0, 'n_orbit_subs': 512, 'n_step': 8192, '
    time_speed_factor': 250.0}
{'function': 'none', 'factor': 1}



################################
```

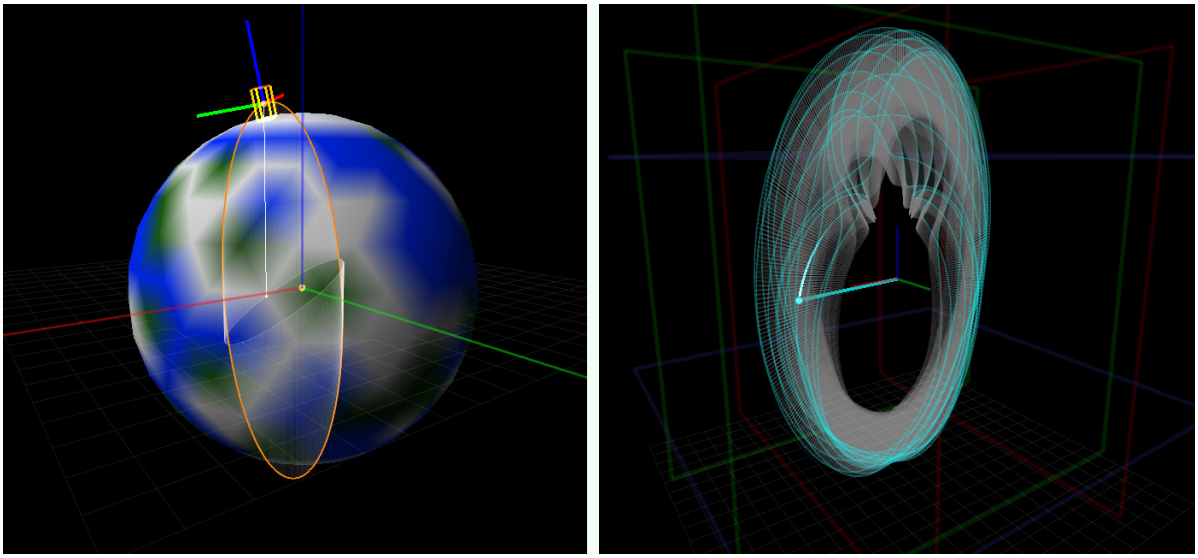


Figure H.13: Left: orbital view in *ECEF* frame. Right: generated magnetic envelope in *B*-frame.

Table H.6: Command errors and angular errors, before and after filtering.

		X [μT]	Y [μT]	Z [μT]	$\mathbf{E_C}$ [μT]	Λ [°]
Unfiltered	Max error	1.846	2.883	3.782	**3.966**	**6.12**
	RMS error	0.933	0.976	1.476	**2.000**	**2.37**
Filtered	Max error	1.037	1.776	2.755	**2.848** (-28.2%)	**5.31** (-13.2%)
	RMS error	0.261	0.734	1.014	**1.279** (-36.1%)	**1.86** (-21.9%)

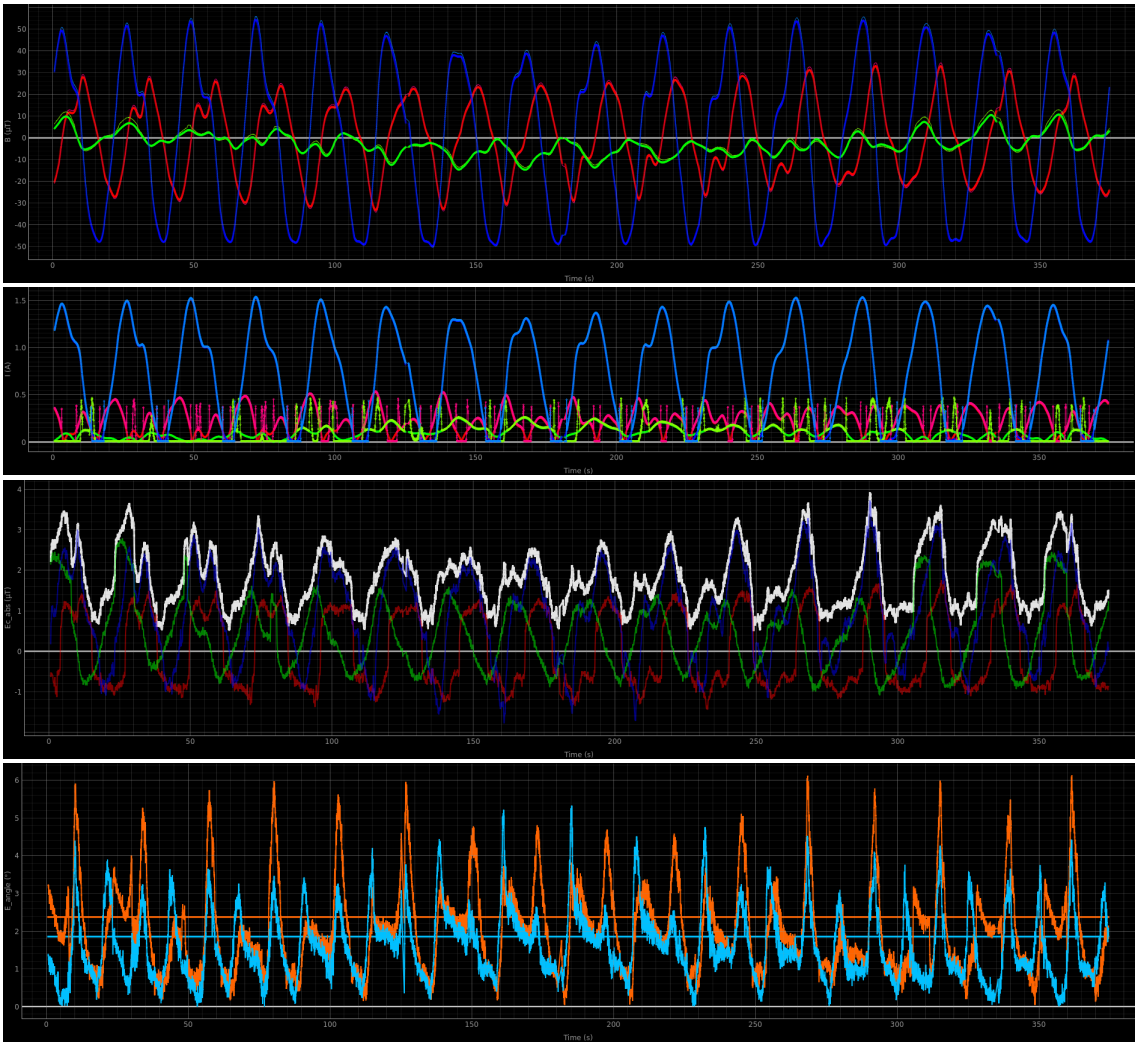


Figure H.14: Test data of `TS_orbit7_20251125_FIX`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue).

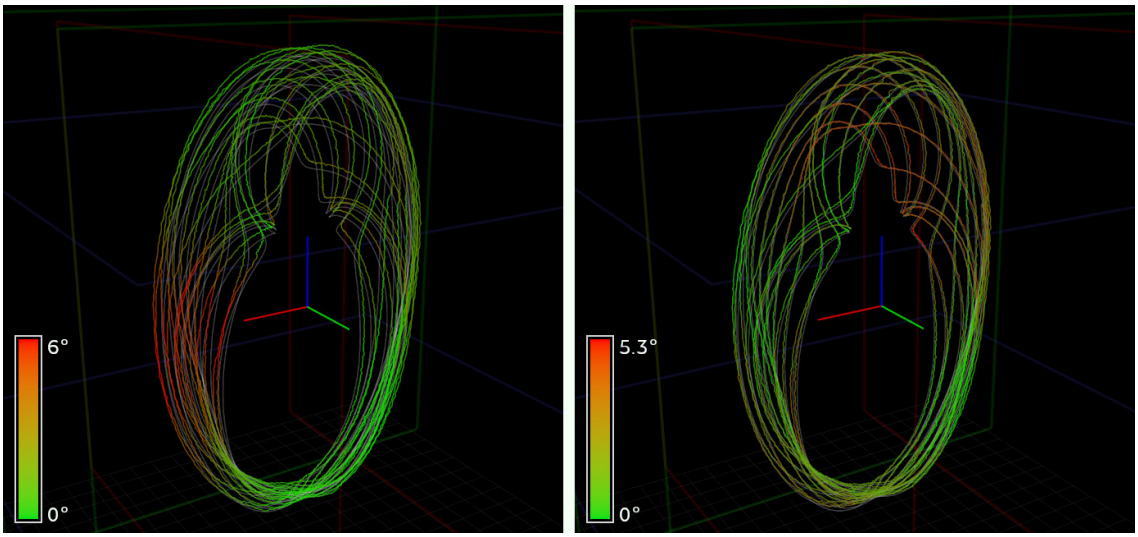


Figure H.15: Left: measured magnetic field manifold. Right: manifold after filtering.

H.6. Orbit 8

Simulates a mission based on the intended IT-SPINS mission [16]. It does not simulate the troubled early section of the mission with the ADCS issues, but rather the intended -2 rpm Y_B-axis rotation used for gathering tomography data. Orbital parameters set shortly after the Cygnus NG-15 deployment at 500 km altitude with an inclination of 51.6°.

TS_orbit8.bsch

```
!BSCH
TS_orbit8
generator=orbital
{'orbit_eccentricity': 0.0005, 'orbit_inclination': 51.6, 'orbit_pericentre_altitude':
     500000, 'orbit_RAAN': 0.0, 'orbit_argp': 0.0, 'orbit_ma0': 200.0, 'angle_body_x_0
    ': 0.0, 'angle_body_y_0': 0.0, 'angle_body_z_0': 0.0, 'rate_body_x': 0.0, '
    rate_body_y': 12.0, 'rate_body_z': 0.0, 'earth_zero_datum': 0, 'date0': 2021.58, '
    n_orbit_subs': 32768, 'n_step': 16384, 'time_speed_factor': 4.0}
{'function': 'none', 'factor': 1}



################################
```

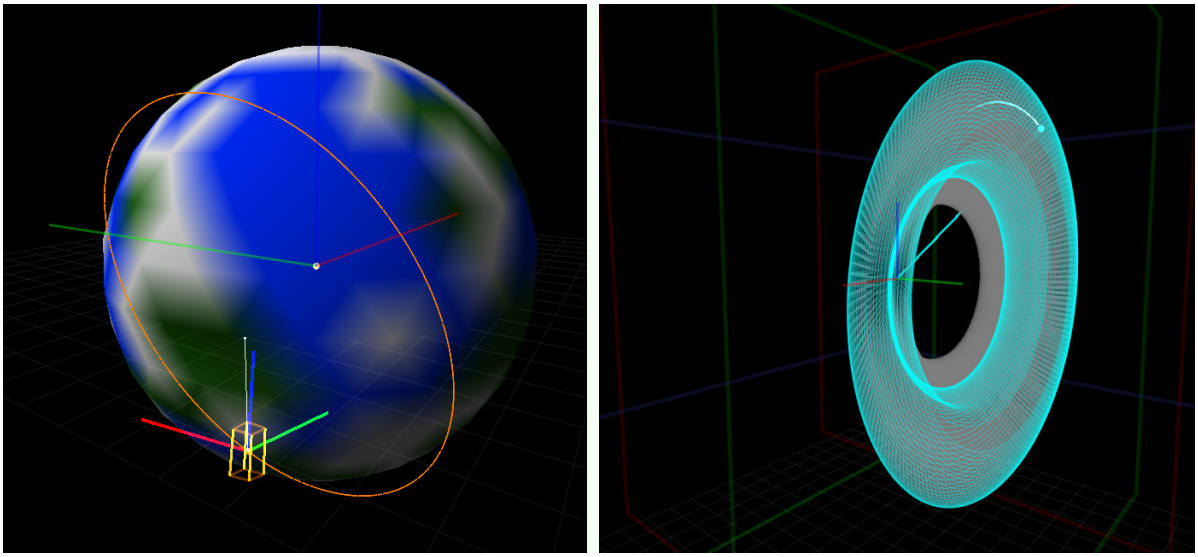


Figure H.16: Left: orbital view in *ECEF* frame. Right: generated magnetic envelope in *B*-frame.

Table H.7: Command errors and angular errors, before and after filtering.

		X [μT]	Y [μT]	Z [μT]	$\mathbf{E_C}$ [μT]	Λ [°]
Unfiltered	Max error	2.965	2.952	3.429	**4.249**	**7.67**
	RMS error	1.168	1.869	1.312	**2.565**	**3.76**
Filtered	Max error	3.764	1.516	2.489	**3.867** (-9.0%)	**6.35** (-17.2%)
	RMS error	0.865	0.573	0.935	**1.397** (-45.5%)	**2.86** (-24.0%)

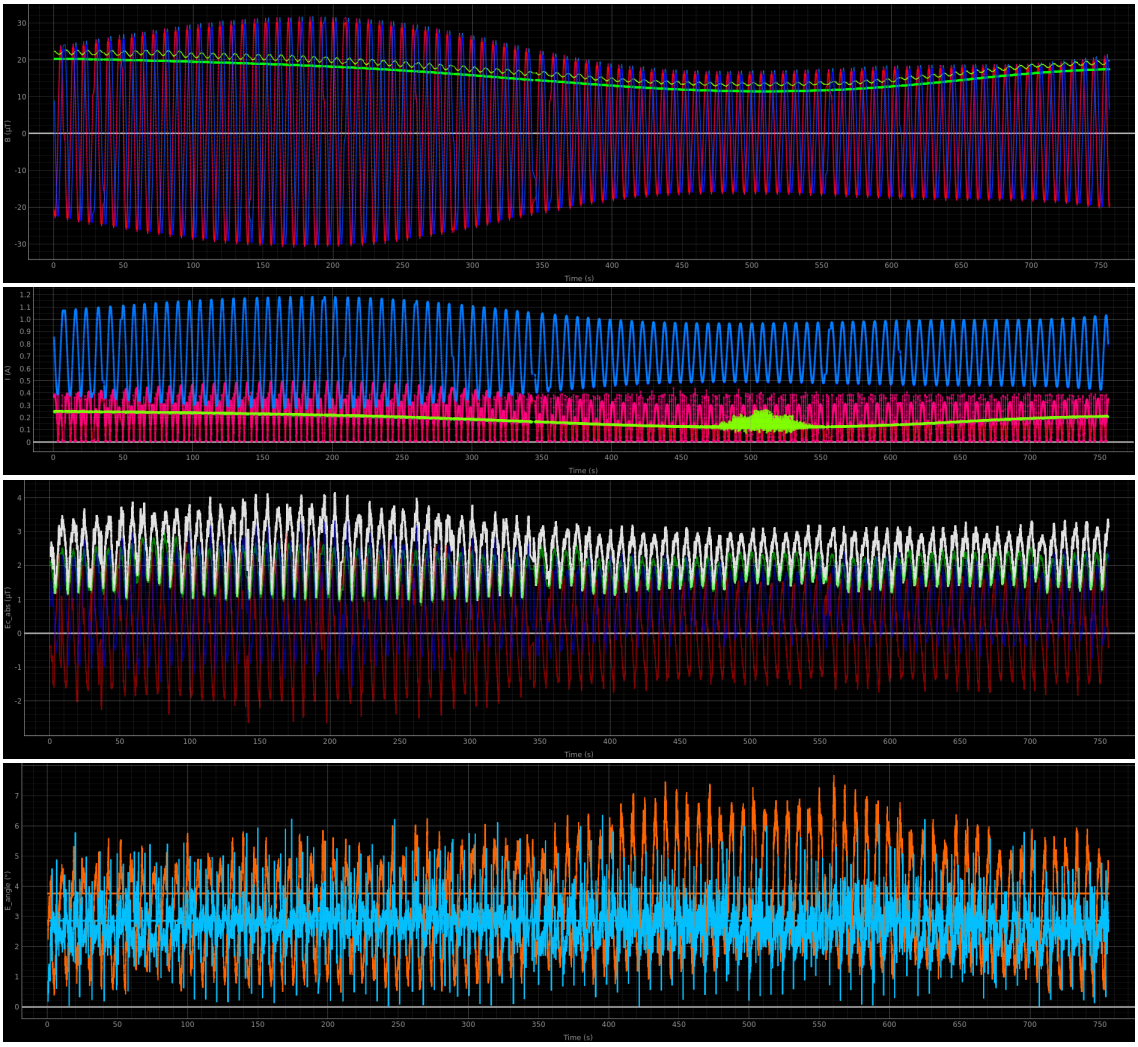


Figure H.17: Test data of `TS_orbit8_20251125_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue).

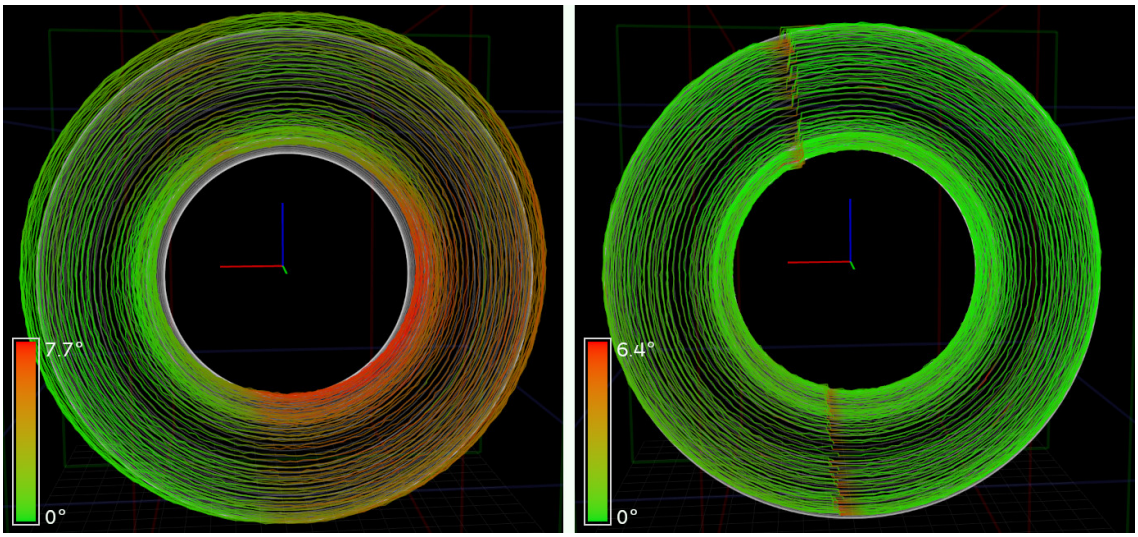


Figure H.18: Left: measured magnetic field manifold. Right: manifold after filtering. Zero-crossings in the X direction cause stepped errors. The filter that removes these leaves artifacts, which are clearly visible on the right.

H.7. Orbit 9

Simulates a hypothetical Delfi-PQ orbit scenario where the Z_B-axis of the satellite is nadir-pointing, and the satellite is slowly rotating around this axis. Orbital parameters based on historical TLE data (457 km altitude, 97.4° inclination).

TS_orbit9.bsch

```
!BSCH
TS_orbit9
generator=orbital
{'orbit_eccentricity': 0.00078, 'orbit_inclination': 97.4, 'orbit_pericentre_altitude'
    : 457200, 'orbit_RAAN': 0.0, 'orbit_argp': 0.0, 'orbit_ma0': 0.0, 'angle_body_x_0'
    : 0.0, 'angle_body_y_0': 0.0, 'angle_body_z_0': 0.0, 'rate_body_x': 0.0, '
    rate_body_y': 0.0, 'rate_body_z': 0.1, 'earth_zero_datum': 0, 'date0': 2023.05, '
    n_orbit_subs': 1024, 'n_step': 8192, 'time_speed_factor': 100.0}
{'function': 'none', 'factor': 1}




################################
```

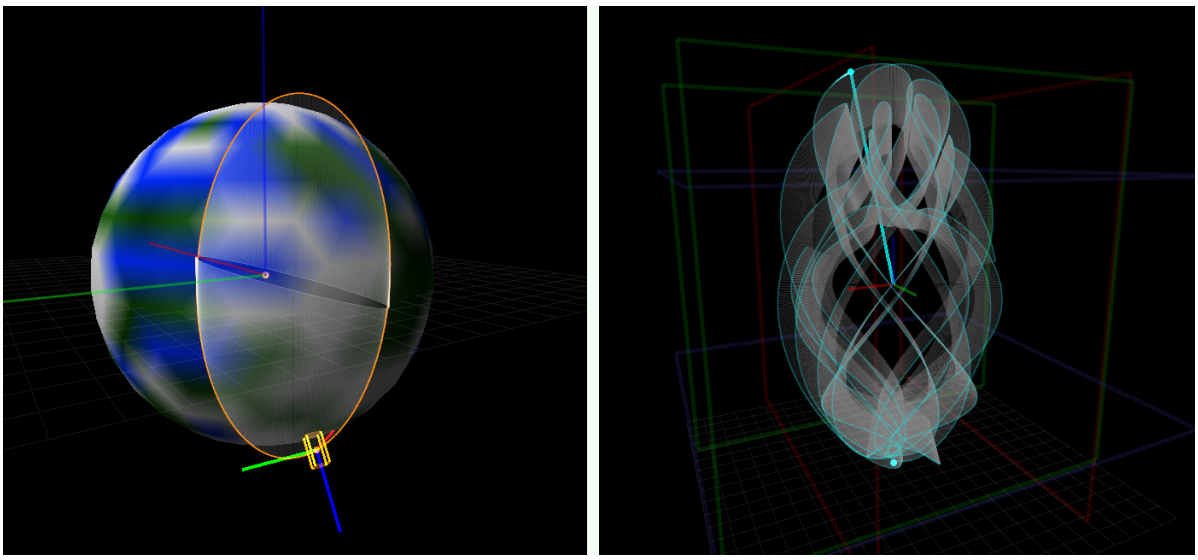


Figure H.19: Left: orbital view in *ECEF* frame. Right: generated magnetic envelope in *B*-frame.

Table H.8: Command errors and angular errors, before and after filtering.

		X [μT]	Y [μT]	Z [μT]	$\mathbf{E_C}$ [μT]	Λ [°]
Unfiltered	Max error	1.945	2.592	2.175	**3.229**	**4.80**
	RMS error	0.871	1.150	1.165	**1.854**	**2.15**
Filtered	Max error	0.885	1.516	0.947	**1.581** (-51.1%)	**2.56** (-46.8%)
	RMS error	0.317	0.704	0.332	**0.840** (-54.7%)	**1.15** (-46.7%)

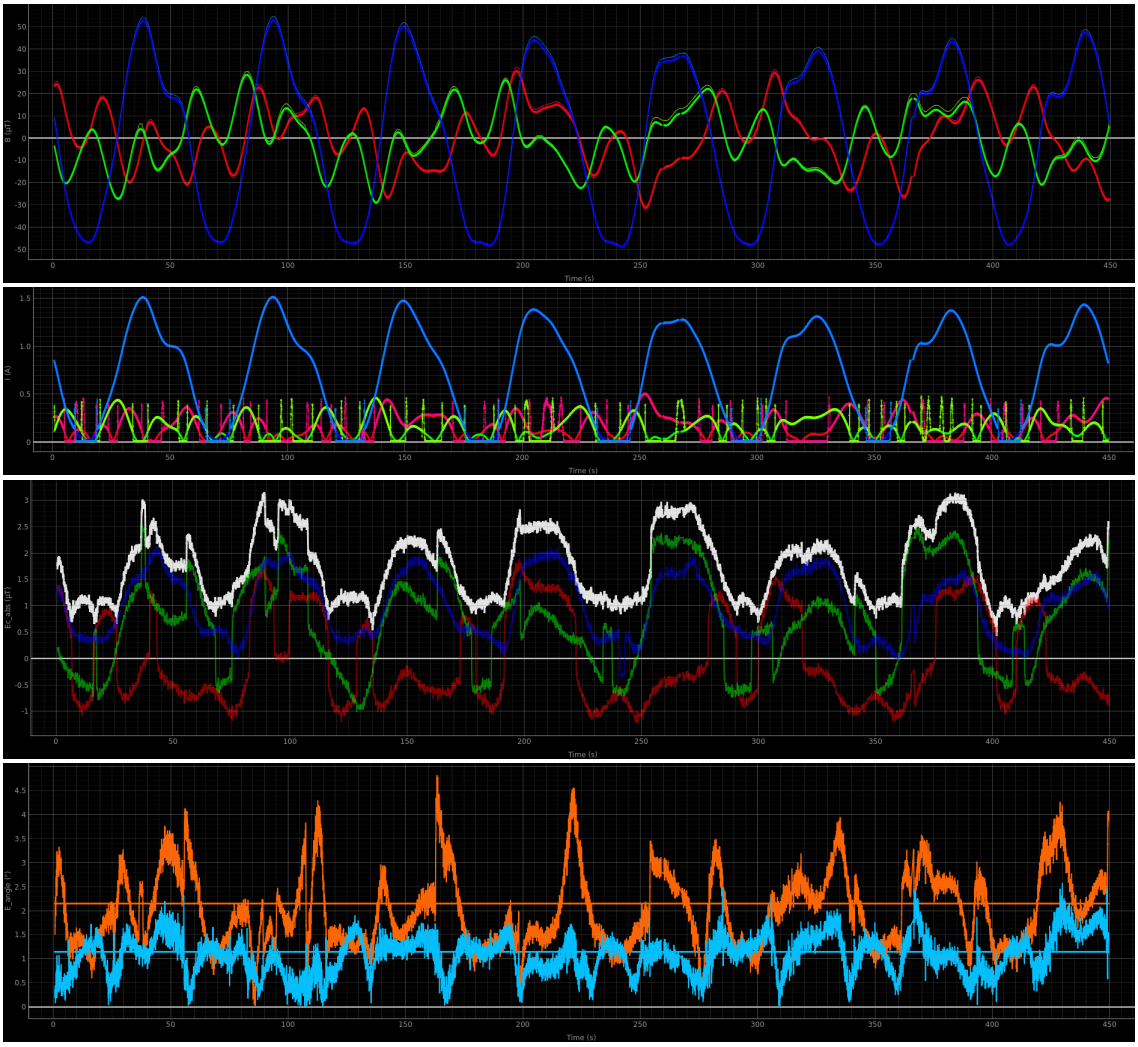


Figure H.20: Test data of `TS_orbit9_20251125_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue).

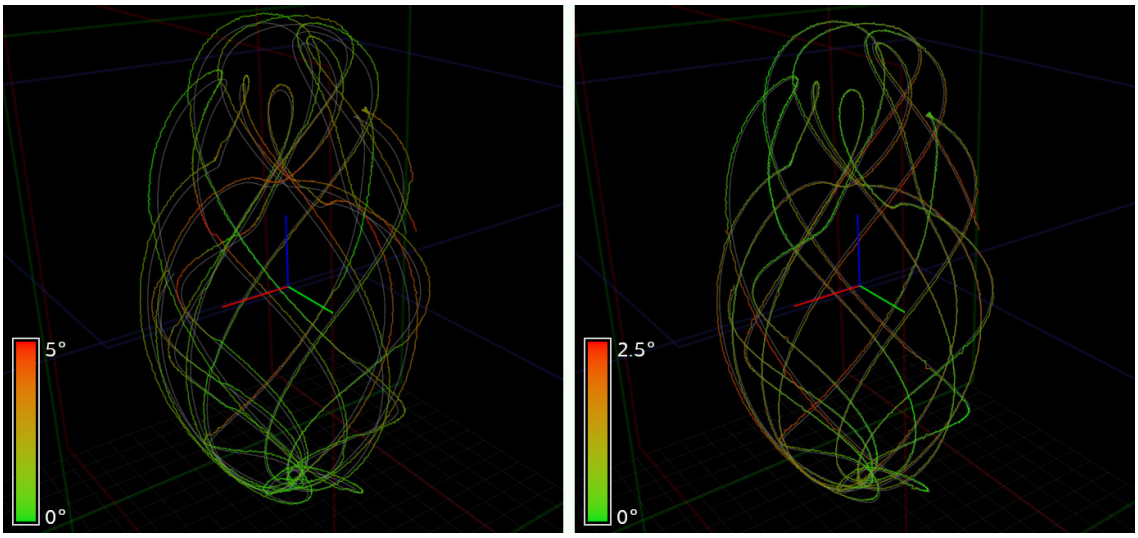


Figure H.21: Left: measured magnetic field manifold. Right: manifold after filtering.

H.8. Orbit 10

Simulates the Hubble Space Telescope in a 28.5° inclination orbit at 488.7 km, pointing at a fixed hypothetical celestial target.

TS_orbit10.bsch

```
!BSCH
TS_orbit10
generator=orbital
{'orbit_eccentricity': 8e-05, 'orbit_inclination': 28.5, 'orbit_pericentre_altitude':
    488700, 'orbit_RAAN': 0.0, 'orbit_argp': 0.0, 'orbit_ma0': 0.0, 'angle_body_x_0':
    90.0, 'angle_body_y_0': 0.0, 'angle_body_z_0': 0.0, 'rate_body_x': 0.0, '
    rate_body_y': -0.06361, 'rate_body_z': 0.0, 'earth_zero_datum': 0, 'date0':
    1991.0, 'n_orbit_subs': 2048, 'n_step': 16384, 'time_speed_factor': 50.0}
{'function': 'none', 'factor': 1}




################################
```

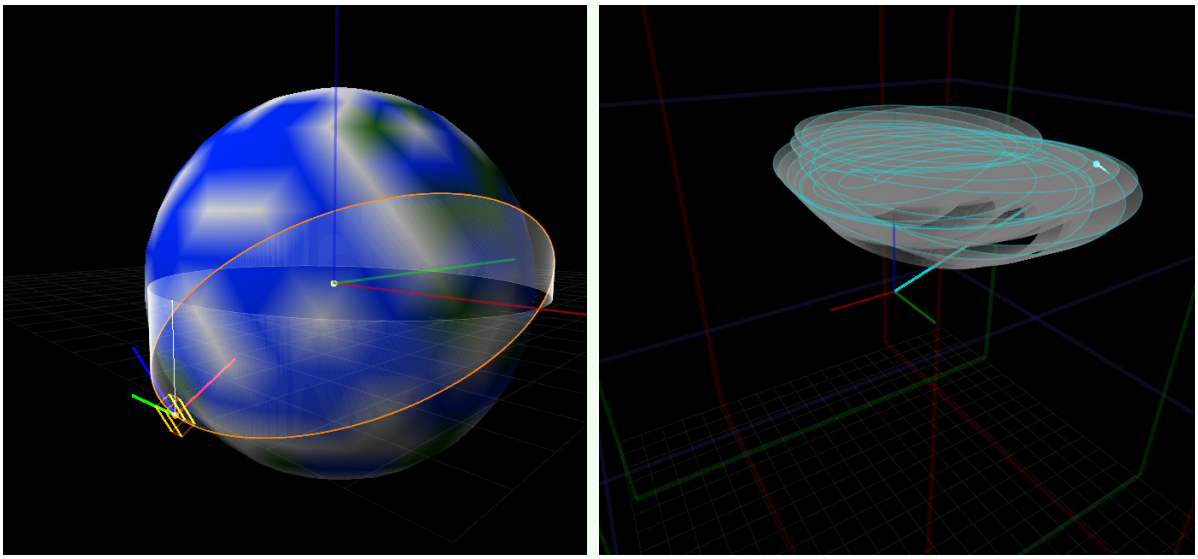


Figure H.22: Left: orbital view in *ECEF* frame. Right: generated magnetic envelope in *B*-frame.

Table H.9: Command errors and angular errors, before and after filtering.

		X [μT]	Y [μT]	Z [μT]	E_C [μT]	Λ [°]
Unfiltered	Max error	1.600	2.544	2.188	**3.020**	**3.53**
	RMS error	0.765	1.399	1.540	**2.217**	**2.09**
Filtered	Max error	1.014	1.006	1.063	**1.395** (-53.8%)	**2.31** (-34.6%)
	RMS error	0.295	0.325	0.426	**0.612** (-72.4%)	**0.72** (-65.8%)

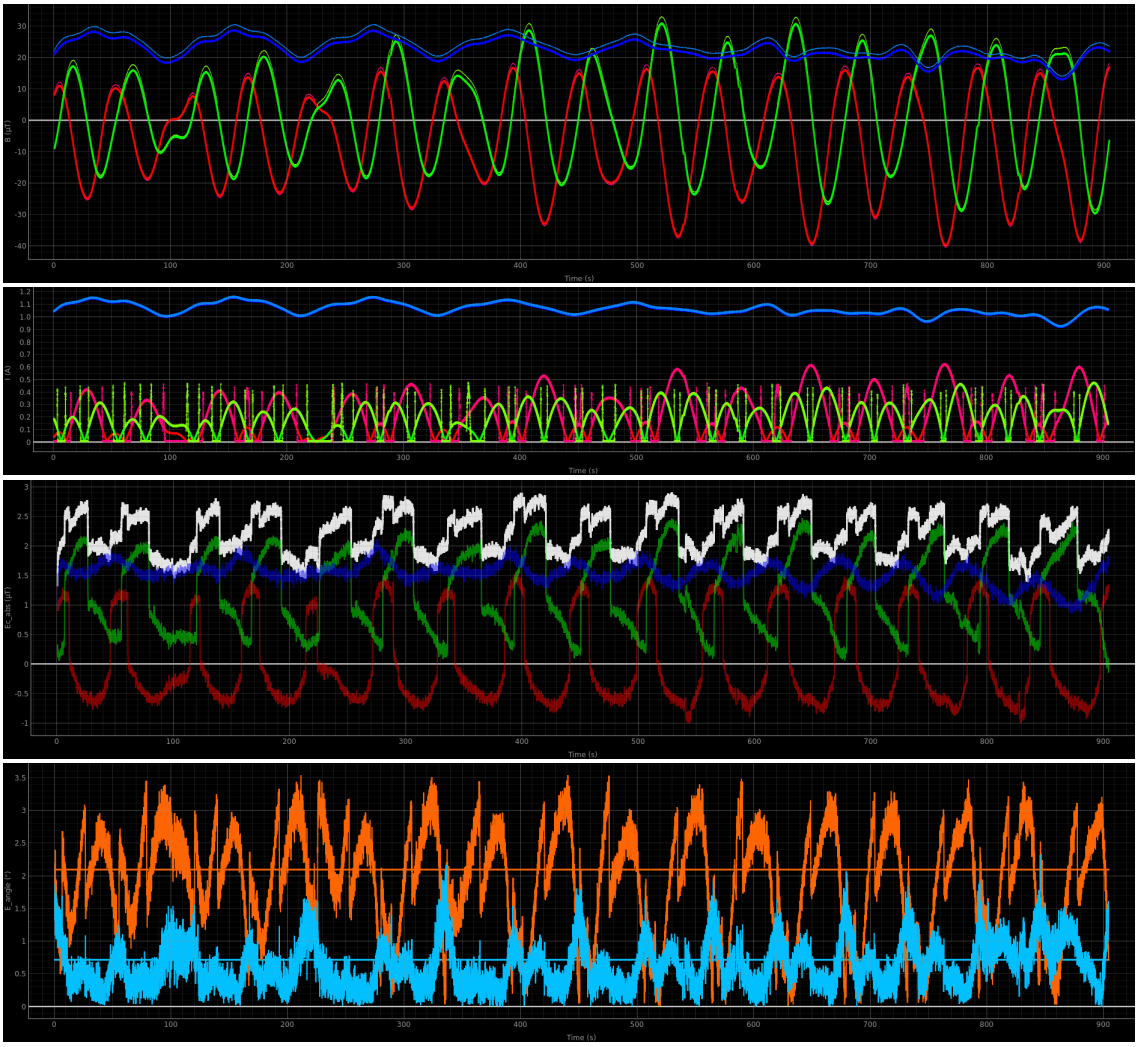


Figure H.23: Test data of `TS_orbit10_20251125_FIX.dat`. Top: $\overline{B}_C$ and $\overline{B}_M$. Top middle: $\overline{I}_C$ and $\overline{I}_M$. Bottom middle: $\overline{E}_C$ with XYZ components. Bottom: Angular error before filtering (orange) and after filtering (blue).

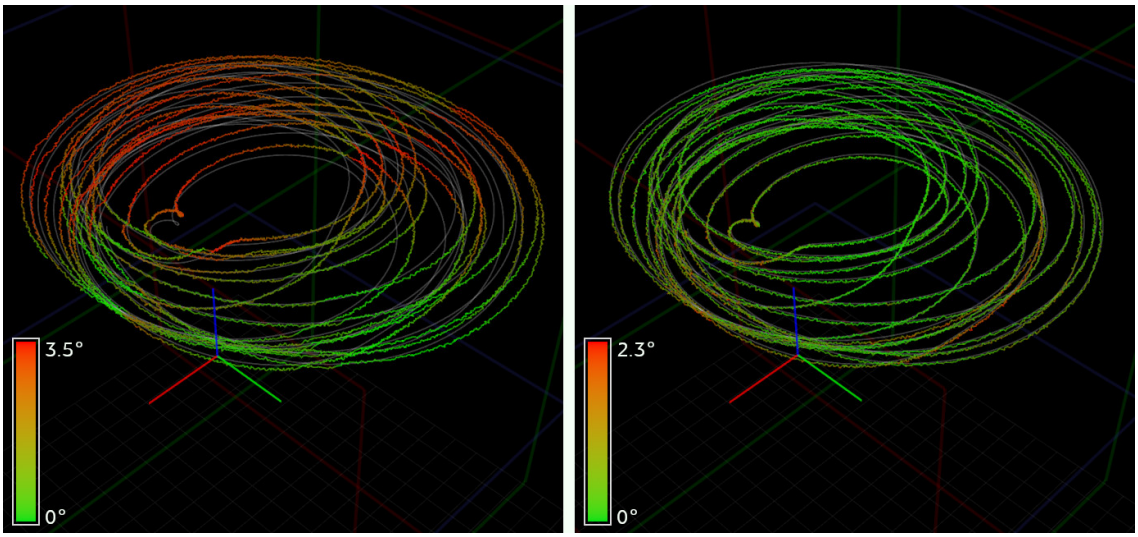


Figure H.24: Left: measured magnetic field manifold. Right: manifold after filtering.

Extra plots on inductance dynamics

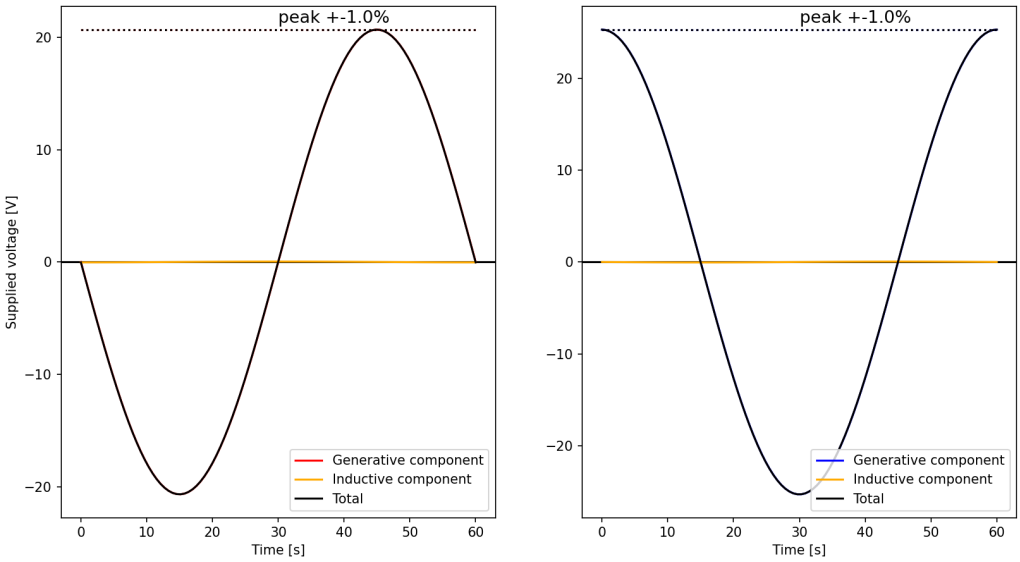


Figure I.1: Supplied voltage envelope of **X** and **Z** during the rotation (1 rpm, 241 μT).

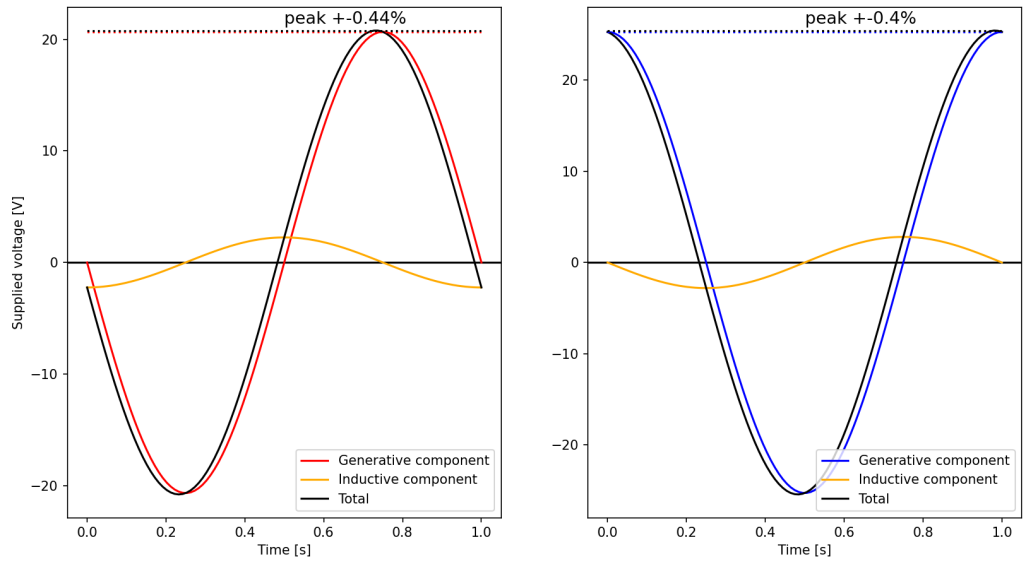


Figure I.2: Supplied voltage envelope of **X** and **Z** during the rotation (60 rpm, 241 μT).

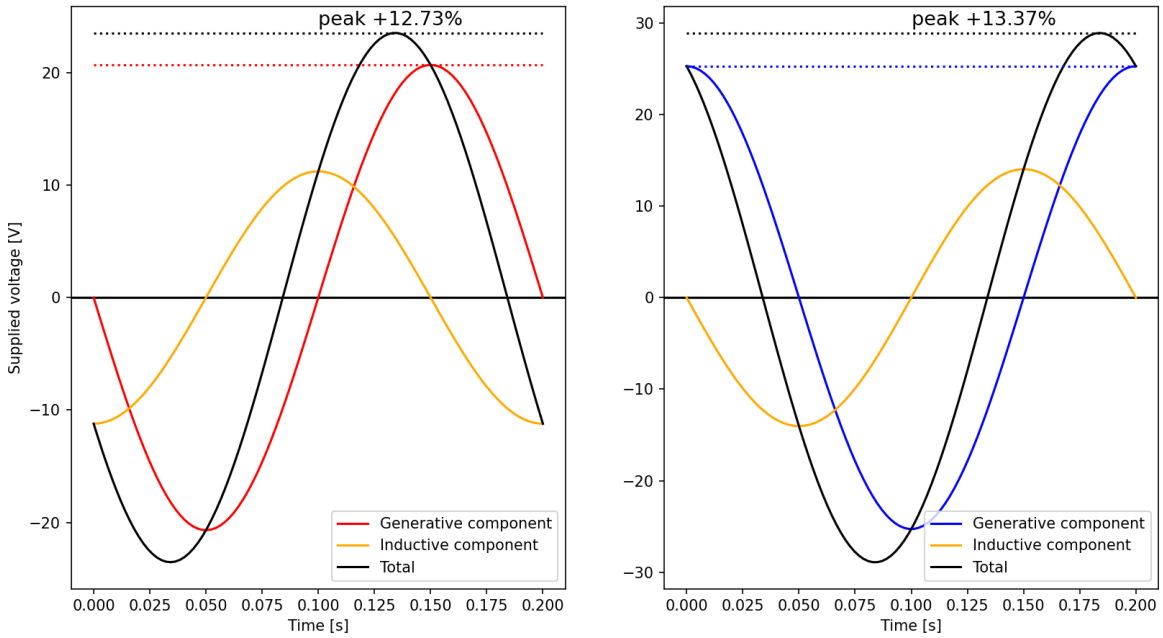


Figure I.3: Supplied voltage envelope of **X** and **Z** during the rotation (300 rpm, 241 μT).

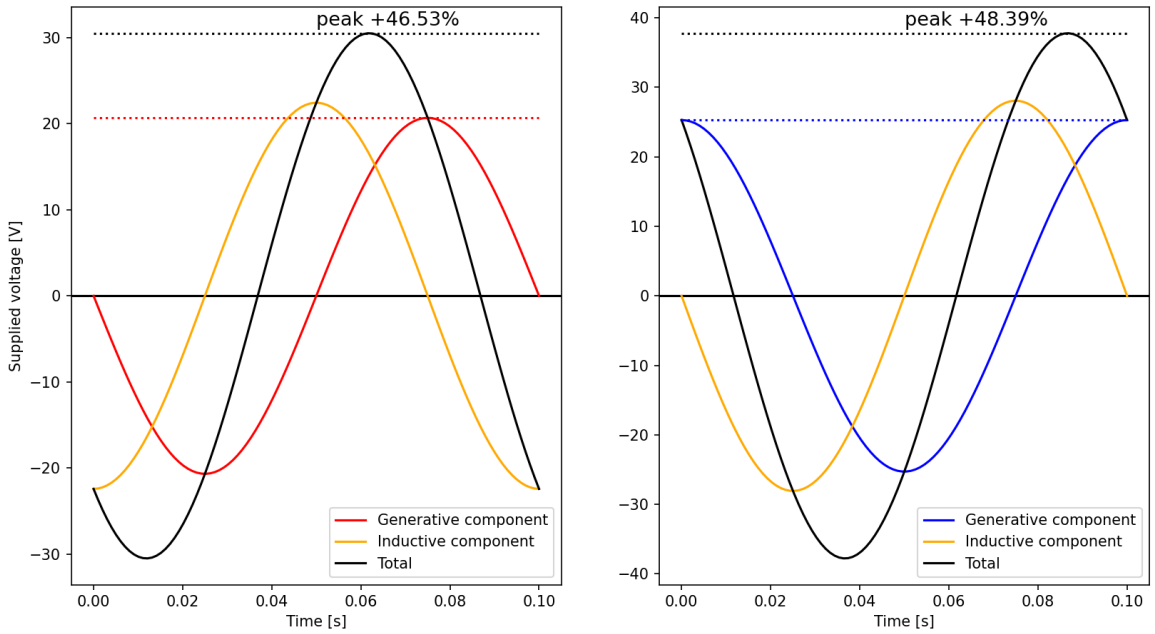


Figure I.4: Supplied voltage envelope of **X** and **Z** during the rotation (600 rpm, 241 μT).

J

Method for Dipole Fitting

In this thesis, we deal with the measurement of magnetic fields. For an object that is exhibiting magnetic properties, the magnetic field around it is often a complex tangle of field lines that can be difficult to describe. A satellite is a good example, as it is often full of dense and complicated electronics. However, as one moves away farther from the source of the magnetic fields, we enter a region where the far-field effects dominate; most complexity of the near-field magnetic properties have mostly been merged into a net field that looks like a simple dipole field with a point source as the origin. Even though the dipole is a theoretical object, it can be very useful in describing the net magnetic field strength of a certain source from far away.

This section deals with fitting the magnetic measurements from one or more sensor locations to a dipole model. The utility is obvious: if we wish to measure the net dipole of a satellite before it is launched into space, we will usually be using sensors to measure the field at several points, and then fit a dipole from this data. This section describes the theoretical background for doing the latter. It is halted mid-way as too much work was involved to properly do it justice. Nonetheless, its implications are interesting, and can serve as the basis for future research.

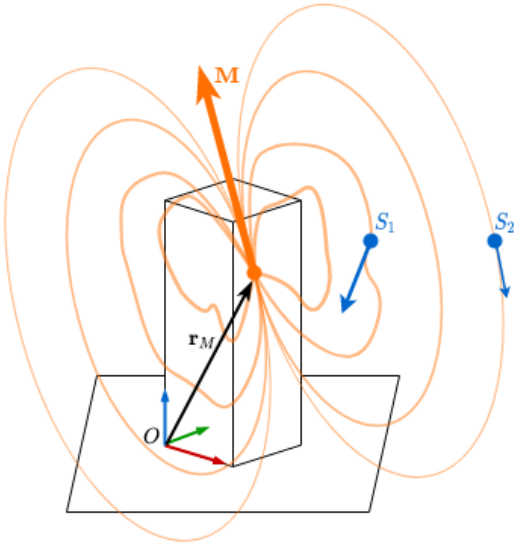


Figure J.1: Measuring the net magnetic moment of a CubeSat. We model it as a simple dipole, with a certain position referenced to an axis system, a direction, and a magnitude. Sensor S_2 will be able to measure this dipole more accurately than sensor S_1, as the measurements of S_1 will be affected by near-field effects.

J.1. Basic Theory

We know from physics that the magnetic flux density around an ideal dipole in free space can be modelled by the equation

$$\mathbf{B}(\mathbf{r}) = \mu_0 \mathbf{H}(\mathbf{r}) = \frac{\mu_0}{4\pi}\left[\frac{3\mathbf{r}(\mathbf{m}\cdot\mathbf{r}) - \mathbf{m}}{|\mathbf{r}|^3}\right] = \frac{\mu_0}{4\pi}\left[\frac{3\mathbf{r}(\mathbf{m}\cdot\mathbf{r})}{|\mathbf{r}|^5} - \frac{\mathbf{m}}{|\mathbf{r}|^3}\right] \tag{J.1}$$

where:

- $\mathbf{B}$ is the magnetic flux density in T
- $\mathbf{H}$ is the H-field strength in $A \cdot m^2$
- $\mathbf{r}$ is the position of the measured point with respect to the dipole, with $\mathbf{r}$ being its unit vector
- $\mathbf{m}$ is the magnetic moment in $A \cdot m^2$
- μ_0 is the free space permeability, which is $4\pi \cdot 10^{-7}\ H/m$

J.2. Lassakeur Implementation

Lassakeur et al. [37] propose to use Equation J.1 to infer the properties of an unknown dipole moment by using magnetometer readings. Assuming that there is a single dipole moment and there are no ambient magnetic fields, this problem becomes a non-linear optimization problem with seven unknowns.

Each magnetic field sensor will give a tri-axial reading of the local magnetic field, which can be expressed as $\mathbf{B}_i$, or $[B_x,\ B_y,\ B_z]$. In addition, each sensor has a known location $(x_i,\ y_i,\ z_i)$. We express the unknown dipole using seven parameters, namely its absolute strength M, the unit vector components of its magnetic moment $[p,\ q,\ r]$ and its Cartesian coordinates $[a,\ b,\ c]$.

Each of these can be related to the unknown dipole using the non-vectorized set of equations below:

$$B_x = \frac{\mu_0 m}{4\pi}\left[\frac{3[n(x-a)+p(y-b)+q(z-c)](x-a)}{R^5} - \frac{n}{R^3}\right] \tag{J.2}$$

$$B_y = \frac{\mu_0 m}{4\pi}\left[\frac{3[n(x-a)+p(y-b)+q(z-c)](y-b)}{R^5} - \frac{m}{R^3}\right] \tag{J.3}$$

$$B_z = \frac{\mu_0 m}{4\pi}\left[\frac{3[n(x-a)+p(y-b)+q(z-c)](z-c)}{R^5} - \frac{p}{R^3}\right] \tag{J.4}$$

with R being the distance from the dipole moment to the sensor location, expressed as:

$$R = \sqrt{(x-a)^2 + (y-b)^2 + (z-c)^2} \tag{J.5}$$

In addition, since we expressed the dipole magnetic moment using a unit vector, we gain an extra compatibility equation, which forces the norm of the magnetic moment to unity:

$$1 = p^2 + q^2 + r2 \tag{J.6}$$

By using n magnetic field sensors, one therefore ends up with a system of $3n + 1$ equations. The methodology proposed by Lassakeur et al. [37] revolves around solving this using a non-linear Levenberg-Marquardt solver.

J.3. Generalized Implementation (single dipole)

The generalized implementation that was created is heavily based on the methodology by Lassakeur et al. [37], because it promises an immense freedom with regards to number of sensors, their positioning, and even the potential to solve for multiple dipoles. However, the implementation by Lassakeur et al. [37] was a bit too limited, as it was optimized for a measurement setup of exactly eight magnetometers at fixed locations.

As a result, the methodology was generalized to allow for a greater amount of freedom. The primary desire was to make the implementation agnostic to the number of sensors, as this was still a free variable in the implementation of the Helmholtz Cage setup at the TU Delft.

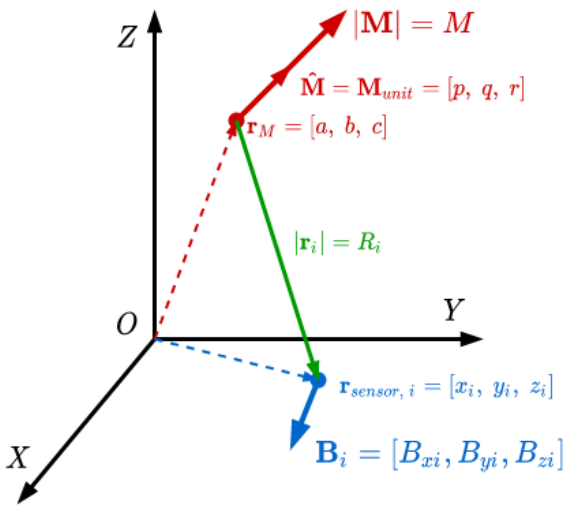


Figure J.2: Sign and symbol convention

First, a sign and symbol convention, as displayed in Figure J.2. We have one unknown dipole moment, which we describe using seven parameters:

- Its strength M $[A \cdot m^2]$
- The unit vector of its magnetic moment $[p,\ q,\ r]$
- Its Cartesian coordinates $[a,\ b,\ c]$ $[m]$

Now, a number of sensors will be placed at various locations around the unknown dipole. Each such sensor has six known quantities, namely:

- Its Cartesian coordinates $[x_i,\ y_i,\ z_i]$ $[m]$
- The measured magnetic flux density components $[B_{x,i},\ B_{y,i},\ B_{z,i}]$ $[T]$

When we take n such sensors, we need a function that will transform the sensor data into the seven unknown dipole moment parameters:

$$f\left(\begin{bmatrix} B_{x,0} \\ B_{x,1} \\ B_{x,2} \\ \vdots \\ B_{x,n-1} \end{bmatrix} \begin{bmatrix} B_{y,0} \\ B_{y,1} \\ B_{y,2} \\ \vdots \\ B_{y,n-1} \end{bmatrix} \begin{bmatrix} B_{z,0} \\ B_{z,1} \\ B_{z,2} \\ \vdots \\ B_{z,n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{n-1} \end{bmatrix}\right) = M, a, b, c, p, q, r \tag{J.7}$$

Much like Lassakeur et al. [37], a non-linear optimization solver will be used to solve for the seven unknowns. But to do this, we need to rewrite the problem somewhat to ensure that the problem can be

parsed by the solver. We do this by explicitly writing out each equation into scalar form using equations J.2 through J.4. We also include the unit vector compatibility (Equation J.6) discussed earlier at the bottom of the system.

$$\begin{cases} B_{x,0} = \dfrac{\mu_0 M}{4\pi}\left[\dfrac{3[p(x_0-a)+q(y_0-b)+r(z_0-c)](x_0-a)}{{R_0}^5} - \dfrac{p}{{R_0}^3}\right] \\ B_{y,0} = \dfrac{\mu_0 M}{4\pi}\left[\dfrac{3[p(x_0-a)+q(y_0-b)+r(z_0-c)](y_0-b)}{{R_0}^5} - \dfrac{q}{{R_0}^3}\right] \\ B_{z,0} = \dfrac{\mu_0 M}{4\pi}\left[\dfrac{3[p(x_0-a)+q(y_0-b)+r(z_0-c)](z_0-c)}{{R_0}^5} - \dfrac{r}{{R_0}^3}\right] \\ \vdots \\ B_{x,n-1} = \dfrac{\mu_0 M}{4\pi}\left[\dfrac{3[p(x_{n-1}-a)+q(y_{n-1}-b)+r(z_{n-1}-c)](x_{n-1}-a)}{{R_{n-1}}^5} - \dfrac{p}{{R_{n-1}}^3}\right] \\ B_{y,n-1} = \dfrac{\mu_0 M}{4\pi}\left[\dfrac{3[p(x_{n-1}-a)+q(y_{n-1}-b)+r(z_{n-1}-c)](y_{n-1}-b)}{{R_{n-1}}^5} - \dfrac{q}{{R_{n-1}}^3}\right] \\ B_{z,n-1} = \dfrac{\mu_0 M}{4\pi}\left[\dfrac{3[p(x_{n-1}-a)+q(y_{n-1}-b)+r(z_{n-1}-c)](z_{n-1}-c)}{{R_{n-1}}^5} - \dfrac{r}{{R_{n-1}}^3}\right] \\ 1 = p^2+q^2+r^2 \end{cases} \tag{J.8}$$

with the n values of R_i given by:

$$R_i = \sqrt{(x_i-a)^2+(y_i-b)^2+(z_i-c)^2} \tag{J.9}$$

J.3.1. Programming implementation

There are a number of things to remark about Equation J.8, and these observations can be used to simplify the implementation somewhat.

1. First of all, we see that the equations have a variations amongst each triplet of equations, but that many terms remain the same across the triplet. Particularly the R_i term is constant for each triplet, but also most of the nominator of the large bracketed fraction. We will introduce an extra shorthand Q_i to compute these terms beforehand, to simplify Equation J.8 and to optimize computation.
2. Since we are computing B-fields, we have a term $\mu_0/4\pi$ appearing in every equation. However, since μ_0 itself equals $4\pi \cdot 10^{-7}$, this can be canceled out to optimize computation further.
3. All equations are uncoupled and independent of each other, short of using the same inputs from the magnetometers and featuring the unknowns of the dipole.

To implement this in software form, the Python package `scipy.optimize` will be used, which features a non-linear root finder with a Levenberg-Marquardt implementation. We choose a state vector `x` such that it contains all unknowns that we want to find:

```
x = [M, a, b, c, p, q, r]
```

The ordering of the state vector is chosen not arbitrarily, but such that indices `1 2 3` and `4 5 6` correspond to the Cartesian components and magnetic moment unit vector components, respectively, as this may aid the eye during debugging. Besides the state vector, the constant values of the sensors have to be fed to the solver. This is done in the following way:

$$\texttt{B} = \begin{bmatrix} 1 & x_0 & y_0 & z_0 & B_{x,0} & B_{y,0} & B_{z,0} \\ 2 & x_1 & y_1 & z_1 & B_{x,1} & B_{y,1} & B_{z,1} \\ 3 & x_2 & y_2 & z_2 & B_{x,2} & B_{y,2} & B_{z,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n & x_{n-1} & y_{n-1} & z_{n-1} & B_{x,n-1} & B_{y,n-1} & B_{z,n-1} \end{bmatrix}$$

The first column is a sensor index which is only used for plotting, but not by the solver. As discussed earlier, there are certain constants which can be pre-calculated for computational efficiency. These are formatted as follows:

$$\texttt{R} = \begin{bmatrix} R_0 \\ R_1 \\ R_2 \\ \vdots \\ R_{n-1} \end{bmatrix}, \quad \texttt{Q} = \begin{bmatrix} Q_0 \\ Q_1 \\ Q_2 \\ \vdots \\ Q_{n-1} \end{bmatrix}$$

The system can now be implemented using the following system of equations:

$$\texttt{R} = \begin{bmatrix} \sqrt{(\texttt{B[0][1]} - \texttt{x[1]})^2 + (\texttt{B[0][2]} - \texttt{x[2]})^2 + (\texttt{B[0][3]} - \texttt{x[3]})^2} \\ \sqrt{(\texttt{B[1][1]} - \texttt{x[1]})^2 + (\texttt{B[1][2]} - \texttt{x[2]})^2 + (\texttt{B[1][3]} - \texttt{x[3]})^2} \\ \vdots \\ \sqrt{(\texttt{B[n-1][1]} - \texttt{x[1]})^2 + (\texttt{B[n-1][2]} - \texttt{x[2]})^2 + (\texttt{B[n-1][3]} - \texttt{x[3]})^2} \end{bmatrix} \tag{J.10}$$

$$\texttt{Q} = \begin{bmatrix} \texttt{x[4]}(\texttt{B[0][1]} - \texttt{x[1]}) + \texttt{x[5]}(\texttt{B[0][2]} - \texttt{x[2]}) + \texttt{x[6]}(\texttt{B[0][3]} - \texttt{x[3]}) \\ \texttt{x[4]}(\texttt{B[1][1]} - \texttt{x[1]}) + \texttt{x[5]}(\texttt{B[1][2]} - \texttt{x[2]}) + \texttt{x[6]}(\texttt{B[1][3]} - \texttt{x[3]}) \\ \vdots \\ \texttt{x[4]}(\texttt{B[n-1][1]} - \texttt{x[1]}) + \texttt{x[5]}(\texttt{B[n-1][2]} - \texttt{x[2]}) + \texttt{x[6]}(\texttt{B[n-1][3]} - \texttt{x[3]}) \end{bmatrix} \tag{J.11}$$

$$\begin{cases} \texttt{B[0][4]} = 10^{-7}\,\texttt{x[0]} \left[\dfrac{3\,\texttt{Q[0]}(\texttt{B[0][1]} - \texttt{x[1]})}{\texttt{R[0]}^5} - \dfrac{\texttt{x[4]}}{\texttt{R[0]}^3} \right] \\ \texttt{B[0][5]} = 10^{-7}\,\texttt{x[0]} \left[\dfrac{3\,\texttt{Q[0]}(\texttt{B[0][2]} - \texttt{x[2]})}{\texttt{R[0]}^5} - \dfrac{\texttt{x[5]}}{\texttt{R[0]}^3} \right] \\ \texttt{B[0][6]} = 10^{-7}\,\texttt{x[0]} \left[\dfrac{3\,\texttt{Q[0]}(\texttt{B[0][3]} - \texttt{x[3]})}{\texttt{R[0]}^5} - \dfrac{\texttt{x[6]}}{\texttt{R[0]}^3} \right] \\ \\ \texttt{B[1][4]} = 10^{-7}\,\texttt{x[0]} \left[\dfrac{3\,\texttt{Q[1]}(\texttt{B[1][1]} - \texttt{x[1]})}{\texttt{R[1]}^5} - \dfrac{\texttt{x[4]}}{\texttt{R[1]}^3} \right] \\ \texttt{B[1][5]} = 10^{-7}\,\texttt{x[0]} \left[\dfrac{3\,\texttt{Q[1]}(\texttt{B[1][2]} - \texttt{x[2]})}{\texttt{R[1]}^5} - \dfrac{\texttt{x[5]}}{\texttt{R[1]}^3} \right] \\ \texttt{B[1][6]} = 10^{-7}\,\texttt{x[0]} \left[\dfrac{3\,\texttt{Q[1]}(\texttt{B[1][3]} - \texttt{x[3]})}{\texttt{R[1]}^5} - \dfrac{\texttt{x[6]}}{\texttt{R[1]}^3} \right] \\ \\ \vdots \\ \\ \texttt{B[n-1][4]} = 10^{-7}\,\texttt{x[0]} \left[\dfrac{3\,\texttt{Q[n-1]}(\texttt{B[n-1][1]} - \texttt{x[1]})}{\texttt{R[n-1]}^5} - \dfrac{\texttt{x[4]}}{\texttt{R[n-1]}^3} \right] \\ \texttt{B[n-1][5]} = 10^{-7}\,\texttt{x[0]} \left[\dfrac{3\,\texttt{Q[n-1]}(\texttt{B[n-1][2]} - \texttt{x[2]})}{\texttt{R[n-1]}^5} - \dfrac{\texttt{x[5]}}{\texttt{R[n-1]}^3} \right] \\ \texttt{B[n-1][6]} = 10^{-7}\,\texttt{x[0]} \left[\dfrac{3\,\texttt{Q[n-1]}(\texttt{B[n-1][3]} - \texttt{x[3]})}{\texttt{R[n-1]}^5} - \dfrac{\texttt{x[6]}}{\texttt{R[n-1]}^3} \right] \\ \\ 1 = \texttt{x[4]}^2 + \texttt{x[5]}^2 + \texttt{x[6]}^2 \end{cases} \tag{J.12}$$

If one looks at Equation J.12, one sees that there are two main sources of variation: there is variation for each three equations, and there is variation across each triplet of equations. This means that these equations can be coded using a double `for`-loop, with an extra manual entry at the end for the compatibility equation. As a result, the code implementation is rather compact and easy to follow, as the snippet shows.

```
# (...)

def system(x, B):

    # Array of distances between sensor position and supposed dipole position
    R = np.zeros(len(B))
    # Array of isolated numerator terms, separated for computational efficiency
    Q = np.zeros(len(B))

    for i in range(len(B)):
        R[i] = ( (B[i][1]-x[1])**2 + (B[i][2]-x[2])**2 + (B[i][3]-x[3])**2 )**0.5
        Q[i] = x[4]*(B[i][1]-x[1]) + x[5]*(B[i][2]-x[2]) + x[6]*(B[i][3]-x[3])

    # Gather system equations
    e = []
    for i in range(len(B)):
        e.append(B[i][4] - 1E-7*x[0]*((3*Q[i]*(B[i][1]-x[1]))/R[i]**5-x[4]/R[i]**3)) #Bxi
        e.append(B[i][5] - 1E-7*x[0]*((3*Q[i]*(B[i][2]-x[2]))/R[i]**5-x[5]/R[i]**3)) #Byi
        e.append(B[i][6] - 1E-7*x[0]*((3*Q[i]*(B[i][3]-x[3]))/R[i]**5-x[6]/R[i]**3)) #Bzi

    # Append unit vector compatibility equation
    e.append(1 - (x[4]**2 + x[5]**2 + x[6]**2))

    return e
#                M     a     b     c    p    q    r
x0 = np.array([1.0, 0.01, 0.01, 0.01, 1.5, 1.5, 1.5])

x_s = root(system, x0, args=(B,),
           method='lm', jac=None, tol=None,
           callback=None, options={"xtol": 1.49012E-20,
                                   "ftol": 1.49012E-20})
# (...)
```

This implementation works for an arbitrary number of sensors, given that a minimum number of sensors is present to make the system determined, which in theory is two. The computation time depends on the speed at which the Levenberg-Marquardt algorithm converges to a solution, which strongly depends on the initial guess and the sensor properties. However, to give an idea: for a simulation with 128 sensors, in which Equation J.12 becomes a system of 385 equations, a typical single-core computation time of around 1000 ms was recorded.

J.4. Generalized Implementation (multi-dipole)

The implementation of the previous section can also be expanded to support multiple unknown dipoles. We will for now ignore the practical implications of this, such as additional constraints upon sensor placements to ensure that the data is measured inside the near-field of the multi-dipole system. Let us first see whether the methodology of the previous section can be extended and successfully converge.

Suppose we expect to find m dipoles in close proximity to one another, and have n magnetometers to place in the vicinity. By approaching this along the same lines as in the previous section, the problem now looks something like this:

$$f\left(\begin{bmatrix} B_{x,0} \\ B_{x,1} \\ B_{x,2} \\ \vdots \\ B_{x,n-1} \end{bmatrix} \begin{bmatrix} B_{y,0} \\ B_{y,1} \\ B_{y,2} \\ \vdots \\ B_{y,n-1} \end{bmatrix} \begin{bmatrix} B_{z,0} \\ B_{z,1} \\ B_{z,2} \\ \vdots \\ B_{z,n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{n-1} \end{bmatrix}\right) = \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{m-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{m-1} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{m-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{m-1} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{m-1} \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ \vdots \\ q_{m-1} \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ \vdots \\ r_{m-1} \end{bmatrix} \tag{J.13}$$

In theory, for this problem to not be under-determined, the number of equations must be equal or greater than the number of unknowns. Adding each sensor adds three equations, whereas adding an additional unknown dipole adds seven new unknowns plus one extra compatibility equation. Barring physical and measurement limitations of the real world, mathematically the number of sensors must outnumber the number of unknown dipoles by a certain factor.

$$3n >= (7-1)m \tag{J.14}$$

$$n >= 2m$$

In other words, to be able to solve the problem given mathematical constraints alone, there must be at least two times as many sensors as there are unknown dipoles.

Exploring further, Equation J.8 can be expanded to accommodate a separate term for the effect of every dipole.

$$\begin{cases} B_{x,0} = \sum_{j=0}^{m-1} \frac{\mu_0 M_j}{4\pi} \left[\frac{3[p_j(x_0 - a_j) + q_j(y_0 - b_j) + r_j(z_0 - c_j)](x_0 - a_j)}{{R_{0,j}}^5} - \frac{p_j}{{R_{0,j}}^3} \right] \\ B_{y,0} = \sum_{j=0}^{m-1} \frac{\mu_0 M_j}{4\pi} \left[\frac{3[p_j(x_0 - a_j) + q_j(y_0 - b_j) + r_j(z_0 - c_j)](y_0 - b_j)}{{R_{0,j}}^5} - \frac{q_j}{{R_{0,j}}^3} \right] \\ B_{z,0} = \sum_{j=0}^{m-1} \frac{\mu_0 M_j}{4\pi} \left[\frac{3[p_j(x_0 - a_j) + q_j(y_0 - b_j) + r_j(z_0 - c_j)](z_0 - c_j)}{{R_{0,j}}^5} - \frac{r_j}{{R_{0,j}}^3} \right] \\ \vdots \\ B_{x,n-1} = \sum_{j=0}^{m-1} \frac{\mu_0 M_j}{4\pi} \left[\frac{3[p_j(x_{n-1} - a_j) + q_j(y_{n-1} - b_j) + r_j(z_{n-1} - c_j)](x_{n-1} - a_j)}{{R_{n-1,j}}^5} - \frac{p_j}{{R_{n-1,j}}^3} \right] \\ B_{y,n-1} = \sum_{j=0}^{m-1} \frac{\mu_0 M_j}{4\pi} \left[\frac{3[p_j(x_{n-1} - a_j) + q_j(y_{n-1} - b_j) + r_j(z_{n-1} - c_j)](y_{n-1} - b_j)}{{R_{n-1,j}}^5} - \frac{q_j}{{R_{n-1,j}}^3} \right] \\ B_{z,n-1} = \sum_{j=0}^{m-1} \frac{\mu_0 M_j}{4\pi} \left[\frac{3[p_j(x_{n-1} - a_j) + q_j(y_{n-1} - b_j) + r_j(z_{n-1} - c_j)](z_{n-1} - c_j)}{{R_{n-1,j}}^5} - \frac{r_j}{{R_{n-1,j}}^3} \right] \\ \\ 1 = {p_0}^2 + {q_0}^2 + {r_0}^2 \\ \vdots \\ 1 = {p_{m-1}}^2 + {q_{m-1}}^2 + {r_{m-1}}^2 \end{cases} \tag{J.15}$$

In addition, R is now an m-by-n matrix, with entries

$$R_{i,j} = \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2 + (z_i - c_j)^2} \tag{J.16}$$

This is a system with $3n + m$ equations, solving for $6m$ unknowns.

J.4.1. Programming implementation

Returning again to a software implementable formulation, the state vector `x` is now a vector of vectors:

$$\mathtt{x} = \begin{bmatrix} M_0 & a_0 & b_0 & c_0 & p_0 & q_0 & r_0 \\ M_1 & a_1 & b_1 & c_1 & p_1 & q_1 & r_1 \\ M_2 & a_2 & b_2 & c_2 & p_2 & q_2 & r_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ M_{m-1} & a_{m-1} & b_{m-1} & c_{m-1} & p_{m-1} & q_{m-1} & r_{m-1} \end{bmatrix}$$

The layout of the `B` matrix remains exactly the same:

$$\mathtt{B} = \begin{bmatrix} 1 & x_0 & y_0 & z_0 & B_{x,0} & B_{y,0} & B_{z,0} \\ 2 & x_1 & y_1 & z_1 & B_{x,1} & B_{y,1} & B_{z,1} \\ 3 & x_2 & y_2 & z_2 & B_{x,2} & B_{y,2} & B_{z,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n & x_{n-1} & y_{n-1} & z_{n-1} & B_{x,n-1} & B_{y,n-1} & B_{z,n-1} \end{bmatrix}$$

However, the pre-calculated vectors `R` and `Q` now both have become m-by-n matrices.

$$\mathtt{R} = \begin{bmatrix} R_{0,0} & R_{0,1} & R_{0,2} & \dots & R_{0,m-1} \\ R_{1,0} & R_{1,1} & R_{1,2} & \dots & R_{1,m-1} \\ R_{2,0} & R_{2,1} & R_{2,2} & \dots & R_{2,m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{n-1,0} & R_{n-1,1} & R_{n-1,2} & \dots & R_{n-1,m-1} \end{bmatrix}$$

$$\mathtt{Q} = \begin{bmatrix} Q_{0,0} & Q_{0,1} & Q_{0,2} & \dots & Q_{0,m-1} \\ Q_{1,0} & Q_{1,1} & Q_{1,2} & \dots & Q_{1,m-1} \\ Q_{2,0} & Q_{2,1} & Q_{2,2} & \dots & Q_{2,m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Q_{n-1,0} & Q_{n-1,1} & Q_{n-1,2} & \dots & Q_{n-1,m-1} \end{bmatrix}$$

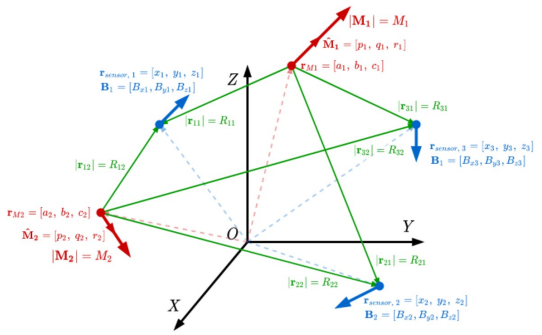
The system can now be implemented using the following system of equations:

$$
\begin{cases}
B[0][4] = 10^{-7}\left[x[0][0]\left[\frac{3\,Q[0][0]\,(B[0][1]-x[0][1])}{R[0][0]^{5}} - \frac{x[0][4]}{R[0][0]^{3}}\right] + x[1][0]\left[\frac{3\,Q[0][1]\,(B[0][1]-x[1][1])}{R[0][1]^{5}} - \frac{x[1][4]}{R[0][1]^{3}}\right] + \cdots + x[m-1][0]\left[\frac{3Q[0][m-1]\,(B[0][1]-x[m-1][1])}{R[0][m-1]^{5}} - \frac{x[m-1][4]}{R[0][m-1]^{3}}\right]\right] \\
B[0][5] = 10^{-7}\left[x[0][0]\left[\frac{3\,Q[0][0]\,(B[0][2]-x[0][2])}{R[0][0]^{5}} - \frac{x[0][5]}{R[0][0]^{3}}\right] + x[1][0]\left[\frac{3\,Q[0][1]\,(B[0][2]-x[1][2])}{R[0][1]^{5}} - \frac{x[1][5]}{R[0][1]^{3}}\right] + \cdots + x[m-1][0]\left[\frac{3Q[0][m-1]\,(B[0][2]-x[m-1][2])}{R[0][m-1]^{5}} - \frac{x[m-1][5]}{R[0][m-1]^{3}}\right]\right] \\
B[0][6] = 10^{-7}\left[x[0][0]\left[\frac{3\,Q[0][0]\,(B[0][3]-x[0][3])}{R[0][0]^{5}} - \frac{x[0][6]}{R[0][0]^{3}}\right] + x[1][0]\left[\frac{3\,Q[0][1]\,(B[0][3]-x[1][3])}{R[0][1]^{5}} - \frac{x[1][6]}{R[0][1]^{3}}\right] + \cdots + x[m-1][0]\left[\frac{3Q[0][m-1]\,(B[0][3]-x[m-1][3])}{R[0][m-1]^{5}} - \frac{x[m-1][6]}{R[0][m-1]^{3}}\right]\right] \\
B[1][4] = 10^{-7}\left[x[0][0]\left[\frac{3\,Q[1][0]\,(B[1][1]-x[0][1])}{R[1][0]^{5}} - \frac{x[0][4]}{R[1][0]^{3}}\right] + x[1][0]\left[\frac{3\,Q[1][1]\,(B[1][1]-x[1][1])}{R[1][1]^{5}} - \frac{x[1][4]}{R[1][1]^{3}}\right] + \cdots + x[m-1][0]\left[\frac{3Q[1][m-1]\,(B[1][1]-x[m-1][1])}{R[1][m-1]^{5}} - \frac{x[m-1][4]}{R[1][m-1]^{3}}\right]\right] \\
B[1][5] = 10^{-7}\left[x[0][0]\left[\frac{3\,Q[1][0]\,(B[1][2]-x[0][2])}{R[1][0]^{5}} - \frac{x[0][5]}{R[1][0]^{3}}\right] + x[1][0]\left[\frac{3\,Q[1][1]\,(B[1][2]-x[1][2])}{R[1][1]^{5}} - \frac{x[1][5]}{R[1][1]^{3}}\right] + \cdots + x[m-1][0]\left[\frac{3Q[1][m-1]\,(B[1][2]-x[m-1][2])}{R[1][m-1]^{5}} - \frac{x[m-1][5]}{R[1][m-1]^{3}}\right]\right] \\
B[1][6] = 10^{-7}\left[x[0][0]\left[\frac{3\,Q[1][0]\,(B[1][3]-x[0][3])}{R[1][0]^{5}} - \frac{x[0][6]}{R[1][0]^{3}}\right] + x[1][0]\left[\frac{3\,Q[1][1]\,(B[1][3]-x[1][3])}{R[1][1]^{5}} - \frac{x[1][6]}{R[1][1]^{3}}\right] + \cdots + x[m-1][0]\left[\frac{3Q[1][m-1]\,(B[1][3]-x[m-1][3])}{R[1][m-1]^{5}} - \frac{x[m-1][6]}{R[1][m-1]^{3}}\right]\right] \\
\vdots \quad \vdots \quad \ddots \quad \vdots \\
B[n-1][4] = 10^{-7}\left[x[0][0]\left[\frac{3\,Q[n-1][0]\,(B[n-1][1]-x[0][1])}{R[n-1][0]^{5}} - \frac{x[0][4]}{R[n-1][0]^{3}}\right] + x[1][0]\left[\frac{3\,Q[n-1][1]\,(B[n-1][1]-x[1][1])}{R[n-1][1]^{5}} - \frac{x[1][4]}{R[n-1][1]^{3}}\right] + \cdots + x[m-1][0]\left[\frac{3\,Q[n-1][m-1]\,(B[n-1][1]-x[m-1][1])}{R[n-1][m-1]^{5}} - \frac{x[m-1][4]}{R[n-1][m-1]^{3}}\right]\right] \\
B[n-1][5] = 10^{-7}\left[x[0][0]\left[\frac{3\,Q[n-1][0]\,(B[n-1][2]-x[0][2])}{R[n-1][0]^{5}} - \frac{x[0][5]}{R[n-1][0]^{3}}\right] + x[1][0]\left[\frac{3\,Q[n-1][1]\,(B[n-1][2]-x[1][2])}{R[n-1][1]^{5}} - \frac{x[1][5]}{R[n-1][1]^{3}}\right] + \cdots + x[m-1][0]\left[\frac{3\,Q[n-1][m-1]\,(B[n-1][2]-x[m-1][2])}{R[n-1][m-1]^{5}} - \frac{x[m-1][5]}{R[n-1][m-1]^{3}}\right]\right] \\
B[n-1][6] = 10^{-7}\left[x[0][0]\left[\frac{3\,Q[n-1][0]\,(B[n-1][3]-x[0][3])}{R[n-1][0]^{5}} - \frac{x[0][6]}{R[n-1][0]^{3}}\right] + x[1][0]\left[\frac{3\,Q[n-1][1]\,(B[n-1][3]-x[1][3])}{R[n-1][1]^{5}} - \frac{x[1][6]}{R[n-1][1]^{3}}\right] + \cdots + x[m-1][0]\left[\frac{3\,Q[n-1][m-1]\,(B[n-1][3]-x[m-1][3])}{R[n-1][m-1]^{5}} - \frac{x[m-1][6]}{R[n-1][m-1]^{3}}\right]\right]
\end{cases}
$$

$$
\begin{aligned}
1 &= x[0][4]^{2} + x[0][5]^{2} + x[0][6]^{2} \\
1 &= x[1][4]^{2} + x[1][5]^{2} + x[1][6]^{2} \\
&\vdots \qquad\qquad \vdots \\
1 &= x[m-1][4]^{2} + x[m-1][5]^{2} + x[m-1][6]^{2}
\end{aligned}
$$

$$
R_{i,j} = \sqrt{(B[i][1]-x[j][1])^{2} + (B[i][2]-x[j][2])^{2} + (B[i][3]-x[j][3])^{2}}
$$

$$
Q_{i,j} = x[j][4](B[i][1]-x[j][1]) + x[j][5](B[i][2]-x[j][2]) + x[j][6](B[i][3]-x[j][3])
$$

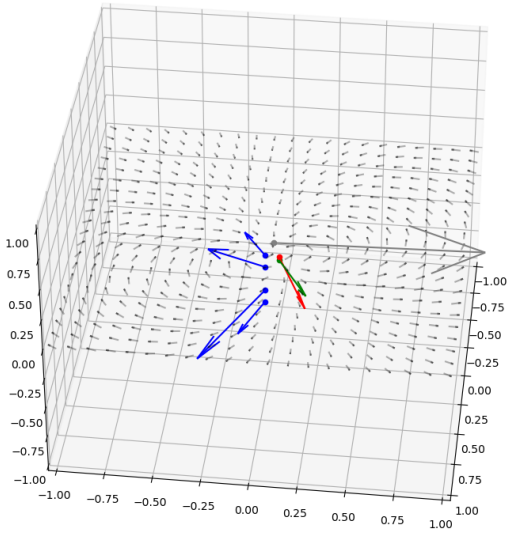


J.5. Simulation Work

The developed methodology could be put to the test by implementing it. A few Python scripts were written, at first to capture essential elements such as magnetic dipoles and sensors as class abstractions. From there, they could be placed in a 3D environment in any desired geometrical arrangement. Then, a solver could be implemented using the methodology from section J.3.

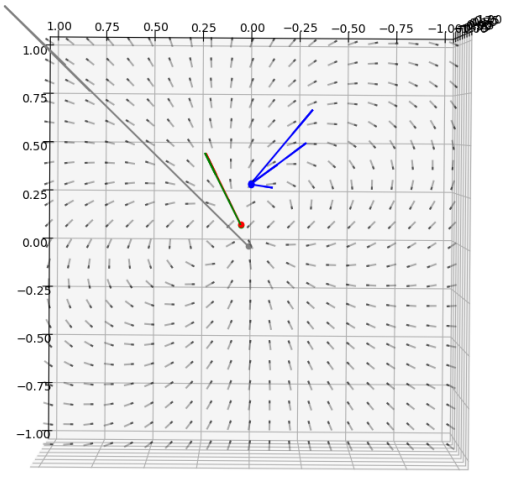
An example of such a simulation can be seen in Figure J.3. Imagine we have a single, unknown dipole (red) which has to be measured. Say we have a linear array of sensors ready nearby to measure it (four blue points). These sensors are each going to measure the local magnetic field strength and direction caused by the dipole. We need an initial guess to get the simulation started, which is the large gray error. The implementation of the estimator runs a simple root finder until it converges at a solution (green). The table in Figure J.3 shows the performance metrics.

We see that the estimator converges relatively well: The location and magnitude of the dipole are very close to the actual values. The direction however is off by over 20°, mostly in the Z-direction. As it turns out, if you choose a different initial location, you may get wildly different results: the estimator may not converge at all, converge perfectly, or land somewhere in between like we see in Figure J.3.

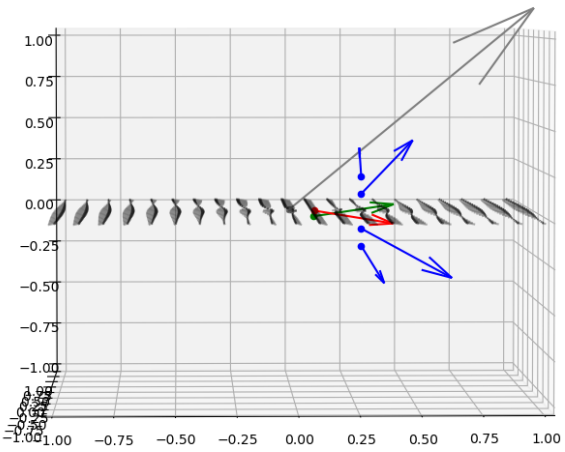


(a) Isometric view

Property	Value
Actual dipole location $[m]$	$[0.1, 0.05, 0]$
Estimated dipole location $[m]$	$[0.096, 0.049, -0.033]$
Position error $[m]$	0.032948
Actual dipole direction [-]	$[0.0873, 0.0436, -0.0218]$
Estimated dipole direction [-]	$[0.0891, 0.0449, 0.0143]$
Direction error	20.762°
Actual dipole magnitude $[Am^2]$	0.1
Estimated dipole magnitude $[Am^2]$	0.101
Magnitude error $[Am^2]$	0.101 (0.8%)



(b) Top view



(c) Side view

Figure J.3: Visualized simulation results. Blue: sensor points and measured vectors. Red: actual dipole. Gray: initial guess. Green: converged estimation.

This indicates that the current implementation is not very robust or reliable. It could be that the root finder we use behaves poorly for this problem, where another solver would perform much better. Or alternatively, it may be that there is fundamentally something about this problem that makes it difficult to reliably solve with this particular methodology.

To investigate it a bit more, we are going to take a similar problem as in Figure J.3, but investigate a large number of different initial guesses. We place a sphere of points in the 3D space, and put an initial guess on each point in the sphere, facing away from the centre of the sphere. Then we try to repeatedly find the dipole, each from a different initial guess.

We can rate the performance of each estimation by using the three error metrics above: the magnitude error, the dipole location error, and the direction error (in degrees). The solver was adjusted to rate each solution by checking whether the error for any particular metric did not exceed a certain tolerance value:

Position error tolerance $[m]$	0.05
Directional error tolerance $[^\circ]$	20
Magnitude error tolerance $[\mu T$	5

If the estimation from a particular initial guess vector did not pass any of the checks, the result was discarded outright. In this way, all initial guesses were distinguished by how many checks they passed, with the output looking something like this:

dipole_model_1P_expand.py (output)

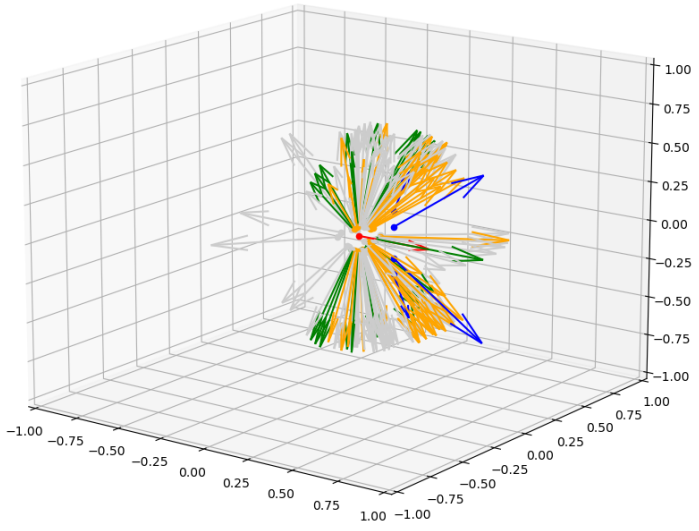
```
                          mag pos ang
Point    0 passed tests:  X   _   _
Point    1 passed tests:  X   _   _
Point    8 passed tests:  X   _   _
Point    9 passed tests:  _   _   X
Point   12 passed tests:  X   _   _
Point   31 passed tests:  _   _   X
Point   32 passed tests:  X   _   _
Point   45 passed tests:  X   _   _
Point   46 passed tests:  X   X   X
Point   48 passed tests:  _   X   _
Point   64 passed tests:  _   X   _

...

Point 1197 passed tests:  X   _   _
Point 1198 passed tests:  X   X   X
Point 1200 passed tests:  _   X   _
Point 1216 passed tests:  _   X   _
Point 1218 passed tests:  X   X   X
Point 1219 passed tests:  X   _   _
Point 1232 passed tests:  X   _   _
Point 1233 passed tests:  _   _   X
Point 1252 passed tests:  X   _   _
Point 1255 passed tests:  _   _   X
Point 1256 passed tests:  X   _   _

Evaluated 1262 estimation cases.
Found 134 partial passes.
Found 21 perfect passes.

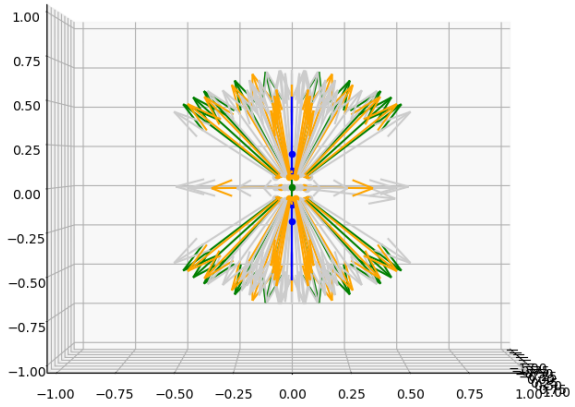
Calculation time: 17.016 ms
Estimator time:   43156.508 ms
Plotting time:    359.102 ms
```



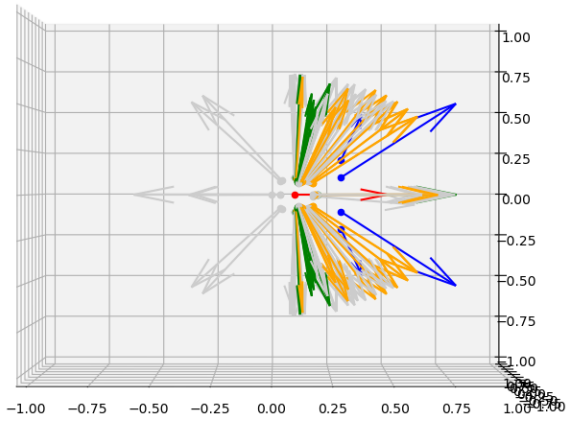
Results:

Total initial guesses	1262
Partial success (1 or 2 checks)	134 (10.6%)
Perfect success (3 checks)	21 (1.66%)

(a) Isometric view



(b) Front view



(c) Side view

Figure J.4: Attempting to gain insight into the performance of the estimator's sensitivity to initial guess. Blue: sensor points and measured vectors. Red: actual dipole. Gray: initial guess passing 1 checks. Orange: initial guess passing 2 checks. Green: initial guess passing all 3 checks. Initial conditions that do not pass any checks are not drawn at all.

The results of a simulation of 1262 points can be seen in Figure J.4. The 3D picture is not much easier to look at than this. What observations can we make?

1. Most initial guesses (88%) lead to no result whatsoever. This estimator is not very reliable at all.
2. An initial guess that coincides with the actual dipole converges fine. You would hope that it would.
3. There is a rather large cone around the dipole position where no initial guess converges at all, except a small scattering of points directly on the XY-plane (no Z component).
4. Outside this cone, but inside the same hemisphere as the actual dipole is pointing, there is a large number of partial successes, and even a small number of successful convergences. Curiously, if the Z-component of the initial guess is below a certain value, it seems to have almost no chance of converging.
5. On the hemisphere facing away from the actual dipole, there is hardly any chance that the solution will converge.

What does this mean? First, it may mean that there is an error somewhere in the methodology or its implementation. The math is reasonably involved, so it is prone to human error. The fact that the Z-direction does not exhibit the same behaviour as the Y axis may be a hint that the problem may be in one of the terms relating to the Z direction. However, no such errors have as of yet been found.

It could also be the case that as mentioned before, the underlying solver has trouble solving this particular problem when posed this way. Confirming or ruling this out will take additional analysis.

However, at this point in the analysis, an important result is found: this is a complicated problem that will take a significant amount of time to get right. Intellectually stimulating as this may be, within the broader context of the thesis work, it is something that should be left aside. It may be that someone else wishes to carry on this work, as there may be a whole thesis worth of work in here. Therefore, the discussion on this dipole estimation methodology will be halted here, and it will be scoped out of the thesis work.

K

Extra materials concerning the characterization of ambient magnetic disturbances

This appendix contains a few extra materials relevant to the documentation of chapter 7, but that were too voluminous to fit there. First, we will look at the tests that were performed. Then we will look at some interesting data that was received from these tests. Then follows a discussion about the methods used for data post-processing to increase the signal-to-noise ratio. Finally, we end with some replication instructions.

K.1. Overview of Tests

During July of 2023 a slew of tests were performed in order to investigate the magnetic noise and other disturbances present at the location of the old Helmholtz cage system.

Table K.1 contains a somewhat messy but comprehensible overview of all relevant tests performed. An even more detailed document called `Test Results.ods` can be found at the official code repository [75] of the data analysis:

`https://github.com/Hans-Bananendans/Gaussmeter-Utilities/blob/master/other/Test%20Results.ods`

The tests were performed over a relatively long time period, as the software tools were developed alongside and became more powerful and stable over time. The tools were written in Python, but particularly the data acquisition code heavily relied on the C-based NI-DAQmx framework to do the data acquisition at speed.

As a reminder, there were a number of objects and parts of the building that were suspected of being sources of magnetic disturbance. These are summarized in Figure K.1.



Figure K.1: Basic map of the floor space around the Helmholtz cage setup. The test area is clearly marked with a yellow box. The potential sources of magnetic interference are marked in orange.

Test Name	Date	Description	Relevant files
Weekend 1	30-06-2023	First long duration test, 100 S/s, recorded over the weekend. **Not used in data analysis.**	`WeekendTest_2023-06-30_15.07.51.dat`
Weekday 1	03-07-2023	Long duration test of a weekday, 100 S/s	`WeekdayTest_2023-07-03_10.41.03.dat` `Weekday1_readout.py` `Weekday1_timeplot.py` `Weekday1_vectorplot.py`
Light Off Test 1	04-07-2023	Goal was to identify the magnetic effects of fluorescent ceiling lights at the test location. Four 5-minute measurements, under identical conditions, with the lab lights off. 100 S/s. **Not used in data analysis.**	`LightOff_2023-07-04_10.57.37.dat` `LightOff_2023-07-04_11.13.19.dat` `LightOff_2023-07-04_11.27.14.dat` `LightOff_2023-07-04_11.42.11.dat`
Light On Test 1	04-07-2023	Control test for *"Light Off Test 1"*. Consists of four 5-minute measurements, each interspersed with the measurements of the other test, in an attempt to smooth the effect of the elevator. 100 S/s. **Not used in data analysis.**	`LightOn_2023-07-04_10.52.22.dat` `LightOn_2023-07-04_11.05.10.dat` `LightOn_2023-07-04_11.18.38.dat` `LightOn_2023-07-04_11.33.19.dat`
Rack Elimination Test	04-07-2023	60-minute 100 S/s measurement at test location, but with the large metal storage rack removed.	`RackElimination_2023-07-04_12.51.32.dat` `RackElimination_preprocessing.py` `RackElimination_vectorplot.py`
Rack Elimination Control	04-07-2023	60-minute slice of the 100 S/s Weekday 1 test corresponding to 13:00-14:00. This was deemed the most appropriate control data, as the storage rack was still present.	`RackEliminationControl_2023-07-03_13.00.00.dat` `RackElimination_preprocessing.py` `RackElimination_vectorplot.py`
Coils Wide Test	04-07-2023	Goal was to get an impression of the latent magnetic effect of the aluminium structure of the Helmholtz cage. Coils were moved far out, such that the spacing was [X: ~1700 mm, Y: ~1700 mm, Z: ~1835 mm] and a 60-minute, 100 S/s test was performed.	`CoilsWide_2023-07-04_16.01.53.dat` `CoilSpacing_vectorplot.py`
Coils Close Test	04-07-2023	Goal was to get an impression of the latent magnetic effect of the aluminium structure of the Helmholtz cage. Coils were moved as close as the table would allow, such that the spacing was [X: ~560 mm, Y: ~560 mm, Z: ~640 mm] and a 60-minute, 100 S/s test was performed.	`CoilsClose_2023-07-04_17.31.25.dat` `CoilSpacing_vectorplot.py`
Coils Normal Test	04-07-2023	Control test for the Coils Close/Wide tests. Original test data replaced in favour of Spatial control test during data analysis.	`SpatialX0Y0_2023-07-12_14.15.31_SPARSE25.dat` `CoilSpacing_vectorplot.py`
Light Off Test 2	05-07-2023	After looking at spectral data of a previous, 100 S/s Light Off/On tests, a higher sample rate test was performed to investigate this further. Brief 5-minute tests at 2500 S/s with lights off.	`2500HzTestLightsOff_2023-07-05_11.13.36.dat` `LightsOnOff_spectralplot.py`
Light On Test 2	05-07-2023	Control test for *"Light Off Test 2"*. Brief 5-minute tests at 2500 S/s with lights off.	`2500HzTestLightsOn_2023-07-05_11.04.28.dat` `LightsOnOff_spectralplot.py`
Weekday Test 2	05-07-2023	Intended to be a full 24-hour test of the lab at 2500 S/s with the lights on. However, it was discovered that the lights in the lab are automatically switched off, so the lights were off from an unknown time at night until around 10 in the morning.	`Weekday2500Hz_LightsOn_2023-07-05_11.46.50.dat` `Weekday2_preprocessing.py` `Weekday2_timeplot.py`
Fire Extinguisher Away Test	06-07-2023	Stefano astutely remarked that there was a large fire extinguisher hanging on the other side of the cleanroom wall, around in the same direction as the normalized disturbance vector. The extinguisher was moved away, and a 1650 second, 2500 S/s was performed.	`FireExtinguisherAway_2023-07-06_18.03.29.dat` `FireExtinguisher_vectorplot.py`
Fire Extinguisher Control Test	06-07-2023	Short 12 minute control test with the fire extinguisher back in its original position.	`FireExtinguisherControl_2023-07-06_18.32.02.dat` `FireExtinguisher_vectorplot.py`
Weekday Test 3	06-07-2023	Intended as a 24 hour lights-off recording, but the lights were accidentally turned on for around an hour in the morning.	`Weekday2500Hz_LightsOff_2023-07-06_19.12.02.dat` `Weekday3_preprocessing.py` `Weekday3_timeplot.py` `Weekday3_vectorplot.py`
Spatial Test	12-07-2023	Eight 45-minute long 2500 S/s recordings in a 0.5 meter grid around the test area, and then one 60 minute control test back in the test area. The main goals were to investigate how the disturbance varies inside the Helmholtz cage, and to attempt to triangulate the main source of the disturbance.	`SpatialX+50Y+0_2023-07-12_06.49.34.dat` `SpatialX+50Y-50_2023-07-12_07.38.19.dat` `SpatialX0Y-50_2023-07-12_08.37.24.dat` `SpatialX-50Y-50_2023-07-12_09.46.26.dat` `SpatialX-50Y0_2023-07-12_10.35.41.dat` `SpatialX-50Y+50_2023-07-12_11.29.49.dat` `SpatialX0Y-50_2023-07-12_08.37.24.dat` `SpatialX+50Y+50_2023-07-12_13.18.39.dat` `SpatialX0Y0_2023-07-12_14.15.31.dat` `SpatialTest_vectorplot.py`
Weekday Test 4	13-07-2023	24+ hour recording at 2500 S/s, with the cleanroom ceiling lights rigged to be on for 24 hours. Main goal was to have a 'definitive' weekday recording, but with the status of the ceiling lights eliminated. Preferred above the other Weekday tests during data analysis.	`Weekday4_2023-07-13_08.29.31.dat` `Weekday4_preprocessing.py` `Weekday4_preprocessing2.py` `Weekday4_timeplot.py` `Weekday4_vectorplot.py` `Weekday4_filtering.py`
Table Elimination	14-07-2023	45 minute 2500 S/s recording where the magnetic probe was taped to one of the Z-coils of the Helmholtz cage, and the 'table' upon which it was usually sat removed from the vicinity. Main goal was to ensure the table itself did not contribute to the constant disturbance measured during the other tests.	`TableElimination_2023-07-14_13.12.26.dat` `TableElimination_preprocessing.py` `TableElimination_vectorplot.py`
Table Elimination Control	14-07-2023	45 minute 2500 S/s control test for Table Elimination. Magnetic probe remained in the same location, but the 'table' was moved right next to it.	`TableEliminationControl_2023-07-14_12.19.22.dat` `TableElimination_preprocessing.py` `TableElimination_vectorplot.py`

One notable hurdle was that the Python code was too slow to reliably poll the USB-6008 device for each single measurement. Polling the data acquisition (DAQ) at more than 50 Hz caused notable slowdown and inconsistencies in sample timing. In response, the software tool was improved to utilize separate threads that pulled batches of pre-defined size from the internal buffer in the DAQ in a round-robin fashion. An extra thread was then used to convert the data from mV to μT and write the data to file. This had four important advantages:

1. It allowed the DAQ to use its own internal clock to more precisely time sample acquisition, without relying on Python-generated signals through the universal serial bus (USB) connection.
2. This in turn allowed data acquisition at rates appropriate for the DAQ, which during testing was set to a maximum of 3x2500 S/s (7500 S/s total), given that there were 3 channels to measure. Though the DAQ card is capable of 3x3333 S/s (10 kS/s total), some headroom was left to avoid pushing the card too hard, and since there was little benefit from the slightly higher sample rate (aside from a somewhat higher Nyquist rate for the spectral analysis).
3. In addition, it allowed for continuous acquisition, as both the hardware buffer on the DAQ and the software buffer in the separate thread ensured that the write-to-file commands would not interrupt the constant flow of incoming data.
4. The improvements to the code also meant that the data acquired during the test spent minimal time in random access memory (RAM), and were written to file quickly. The maximum data size that could be acquired during one test grew from the size of available RAM to the size of the storage disc, and it also meant that test could be interrupted midway without loss of data.

K.2. Data interpretation and results

This section will explain the approach to data handling and the resulting measurement data in a format that can be easily understood.

The test data was stored as plaintext `.dat` files with four columns of data. The first column contains Unix timestamps of the moment when the measurement was performed, and the remaining three columns are the value of the magnetic flux density in μT for the X, Y, and Z axis respectively. A small header is attached to the top of the file to store relevant test information without having to rely on filenames alone. It was important to do this to ensure datasets would never be confused with one another.

The largest datasets were those of the 2500 S/s Weekday tests, which uncompressed lead to a filesize of over 9 GB. This is the territory in which the data handling code has to work with assumption that the data is "larger-than-RAM". Some of the data handling was complicated somewhat because the code had to be written in such a way that it would not read all the data first, which can be problematic in Python, though not impossible. Also, due to the size of the datasets, they have to be stored separately from the software tool repository. More information on this in the appendix.

The large filesizes also lead to limitations on the data handling front. Typically, datasets over 1 GB were cut up to more manageable levels, and where possible, the data was undersampled by factors of 25 or 50. This was possible for all but the part of the data analysis that dealt with spectral results, which requires a high sampling rate. The reason for this high sampling rate requirement is that the frequency range of Fourier analysis is limited to the Nyquist rate of the dataset, which is half the data acquisition rate. Therefore, the spectral part of the data analysis used 2500 S/s data so that frequencies up to 1250 Hz could be resolved. For all other tests, 100 Hz was typically used. The undersampling was implemented as simple, periodic sampling and does not include any randomization.

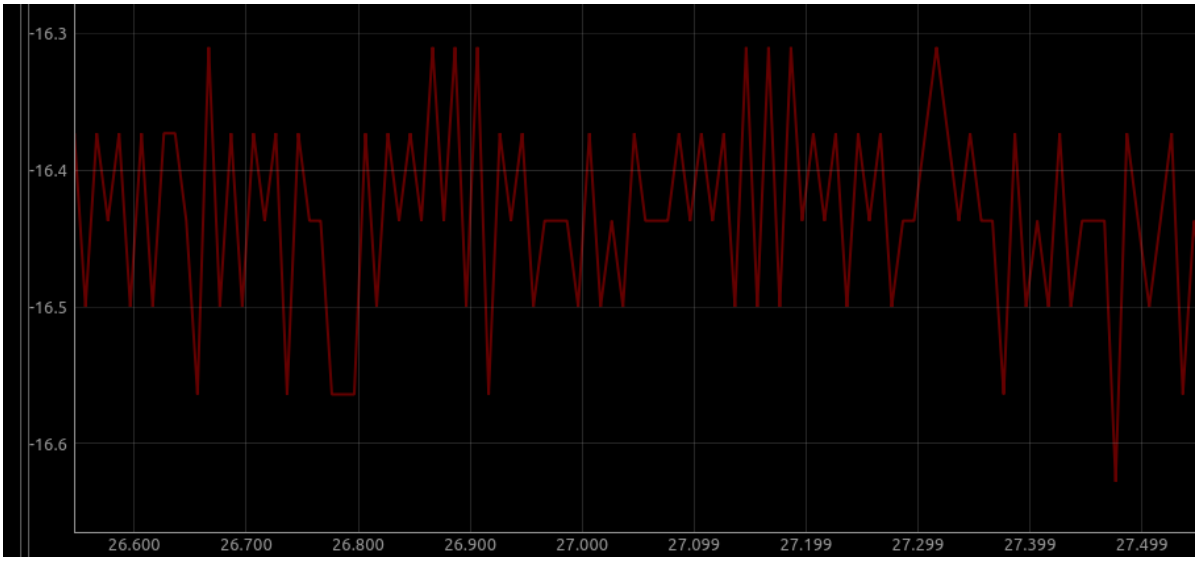


Figure K.2: Raw magnetic flux data showing coarse vertical discretization, which is caused by the 12bit limitation of the DAQ card.

The first problem with the data is that the accuracy of the milligaussmeter was unfortunately quite limited by the 12bit resolution of the USB-6008 DAQ card. The effect of this can be clearly seen in figure 6. Beyond acquiring more accurate DAQ hardware, there was no way to avoid this. However, to attempt to improve the quality of the data, a Savitzky-Golay filter was used to smooth the data. This filter was configured to use a windows of 51 samples, and use a fourth-order fitting polynomial. This seemed to smooth out the data adequately, though realistically, without better quality data to compare it to, there is no way to assess the quality of the filtered data.

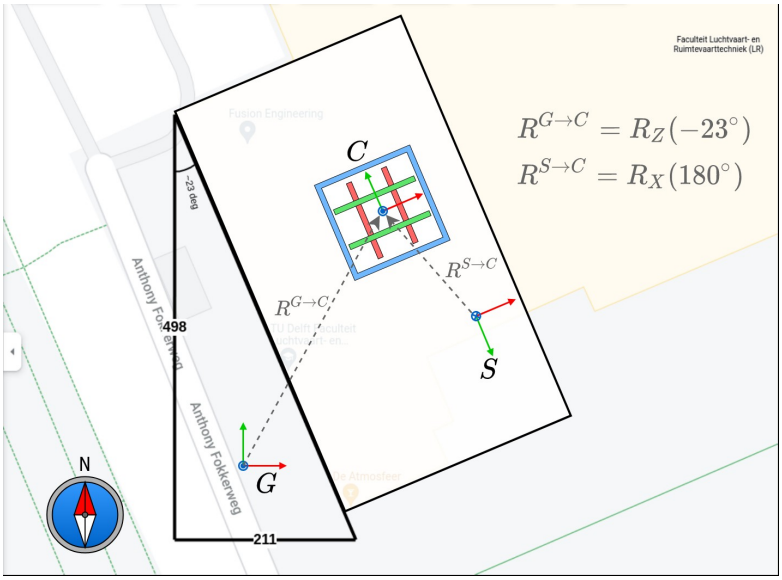


Figure K.3: The coordinate systems used in this chapter. Cage-centered system C is preferred above the others, so any information in the geographic system G or sensor system S will be converted to the C frame. In both cases, this can be achieved through a single-axis (Euler matrix) rotation, as depicted.

This brings us to the first result: the average disturbance vector. A dataset of measurements for a period of 24 hours was taken, and its datapoints treated as a normally distributed set of points for each axis. First, the data had to be rotated from the sensor-frame S to the cage-centered frame C, according to the convention depicted in Figure K.3. Then, the mean and standard deviation could be calculated for each

axis. The three mean values formed the X, Y, and Z components of the measured data vector, by taking the norm of this 3D vector, the magnitude of the measured vector expressed in μT could be found. This resulted in a measured vector with a magnitude of around 47 μT.

Since this magnitude is very close to the magnitude of the local Earth magnetic field vector (49 μT), this may lead one to believe that the measurement consist almost exclusively of the local Earth magnetic field (EMF). However, as we will see in a minute, the measured vector points in a different direction than the local EMF vector. The local EMF can be subtracted from the measured vector. To do this, the local EMF vector first had to be rotated from the geographic frame G into the cage-centered frame C, again according to Figure K.3. After subtracting the local EMF component from the measured vector, we are left with a vector that will henceforth be referred to as the **disturbance vector**. That is: the field vector that can be measured at the test area, but with the EMF component removed. These vectors are depicted in Figure K.4 and Figure K.5.

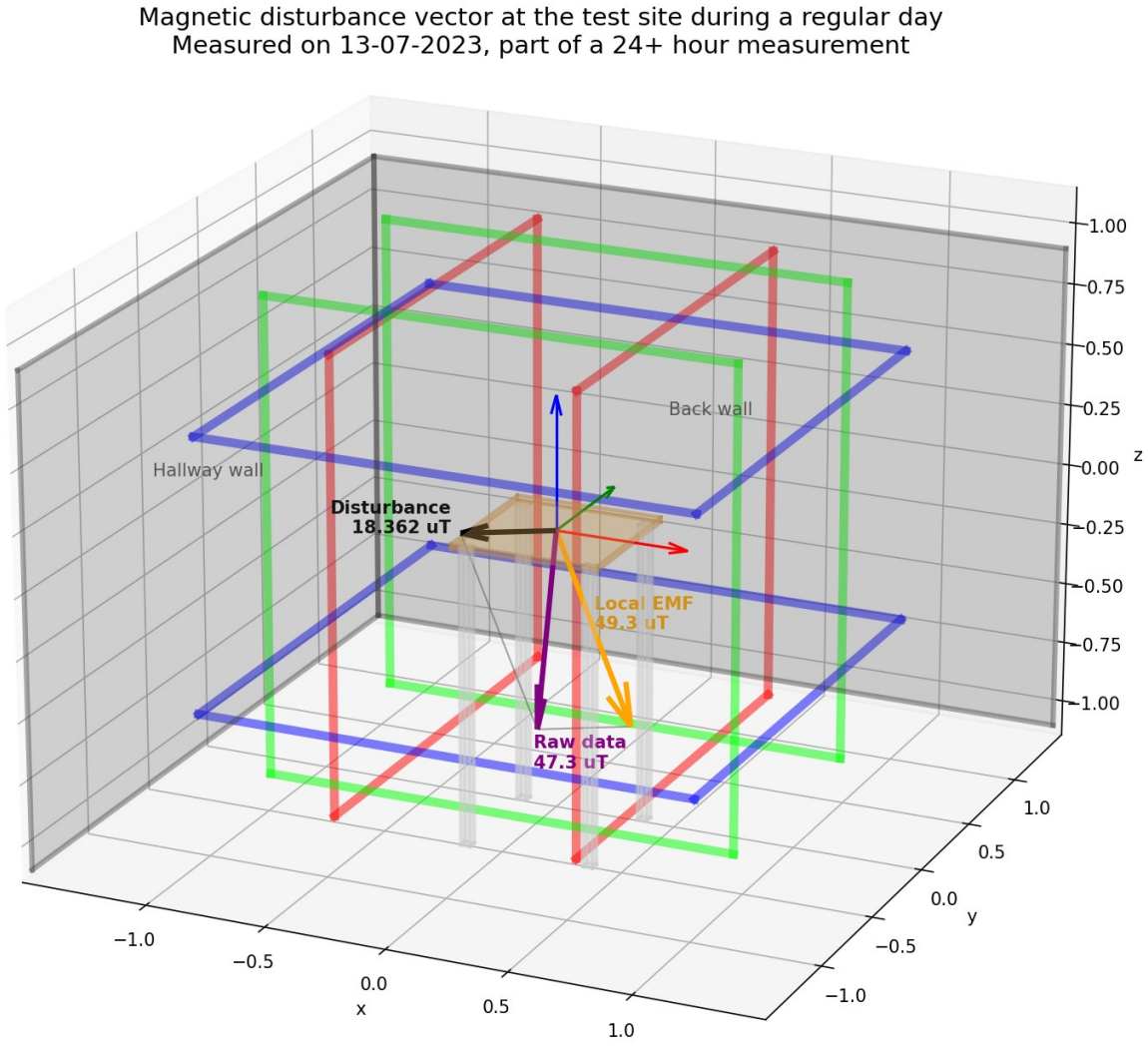


Figure K.4: Rendered Helmholtz cage elements and wall elements in the lab. The cage-centered frame C is plotted as a standard XYZ tripod. The orange vector is the local EMF field and the purple vector indicates the raw, rotated measurement data. The difference between these vectors results in the black vector, which is the disturbance vector, and has a magnitude of around 18.4 μT.

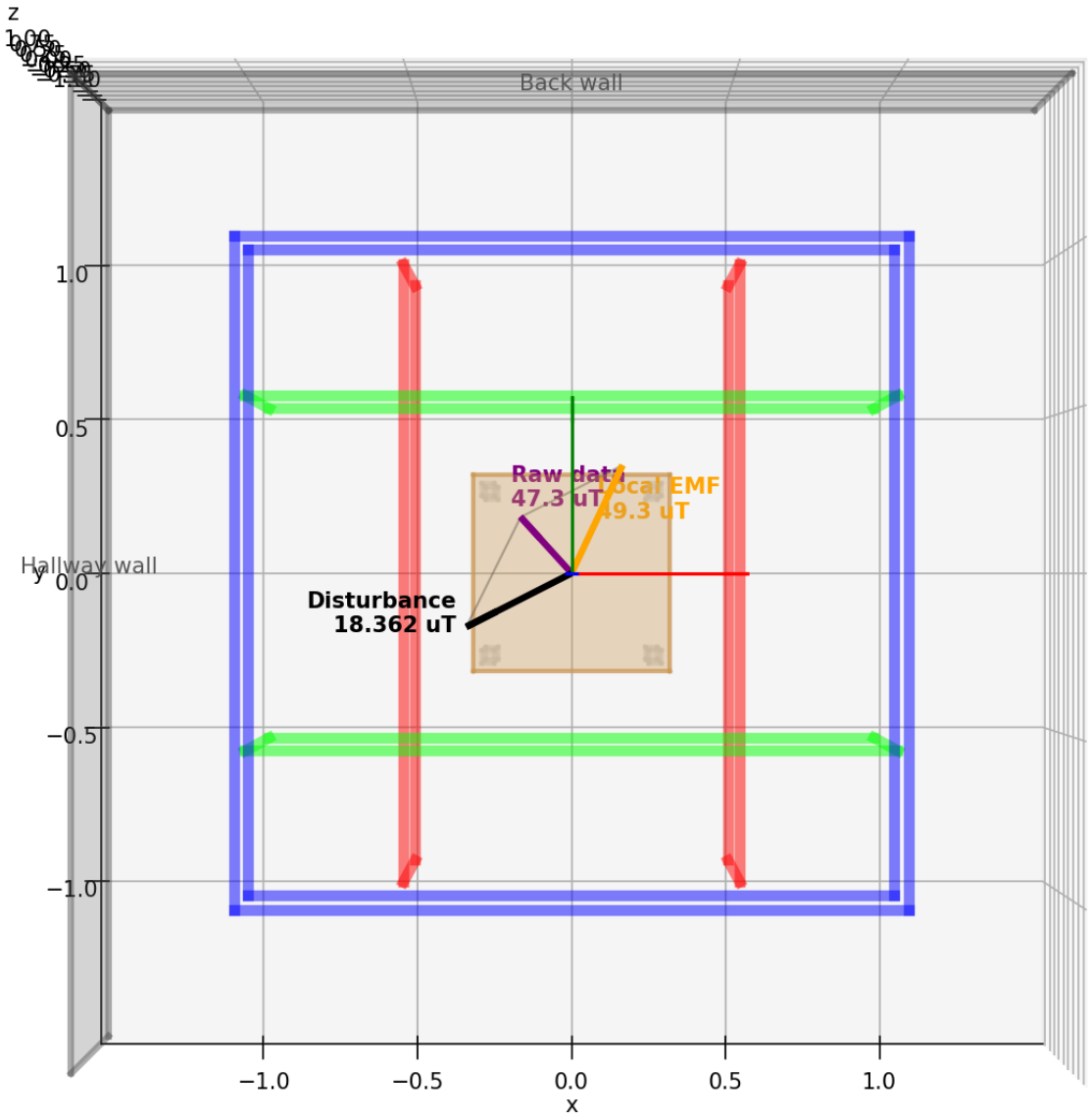


Figure K.5: A top-view of the situation depicted in Figure 7.6. An interactive 3D view of this image can be plotted using the file `disturbance_visualization.py` on the code repository.

The average disturbance vector has the following properties:

```
Mean XYZ:    [-16.358 -8.338 -0.248] uT
SD XYZ:          [0.063 0.017 0.066] uT
Magnitude:                    18.362 uT
```

However, this vector is merely an average. How does it behave over time? The standard deviation of the axis components already indicates that the data fluctuates little, particularly for the Y axis, but we will dive deeper.

Figure K.6 shows the three magnetic field components of a 24 hour recording of the magnetic flux density at the test site. This data has been rotated and smoothed using Savitzky-Golay filtering, as described previously. What is most striking about this image is how little the magnetic field data varies in comparison with the absolute length of the vector. In other words, over the course of the 24 hours, the disturbance vector is very stable, and could be said to only "lightly quiver" in the **XZ**-plane.

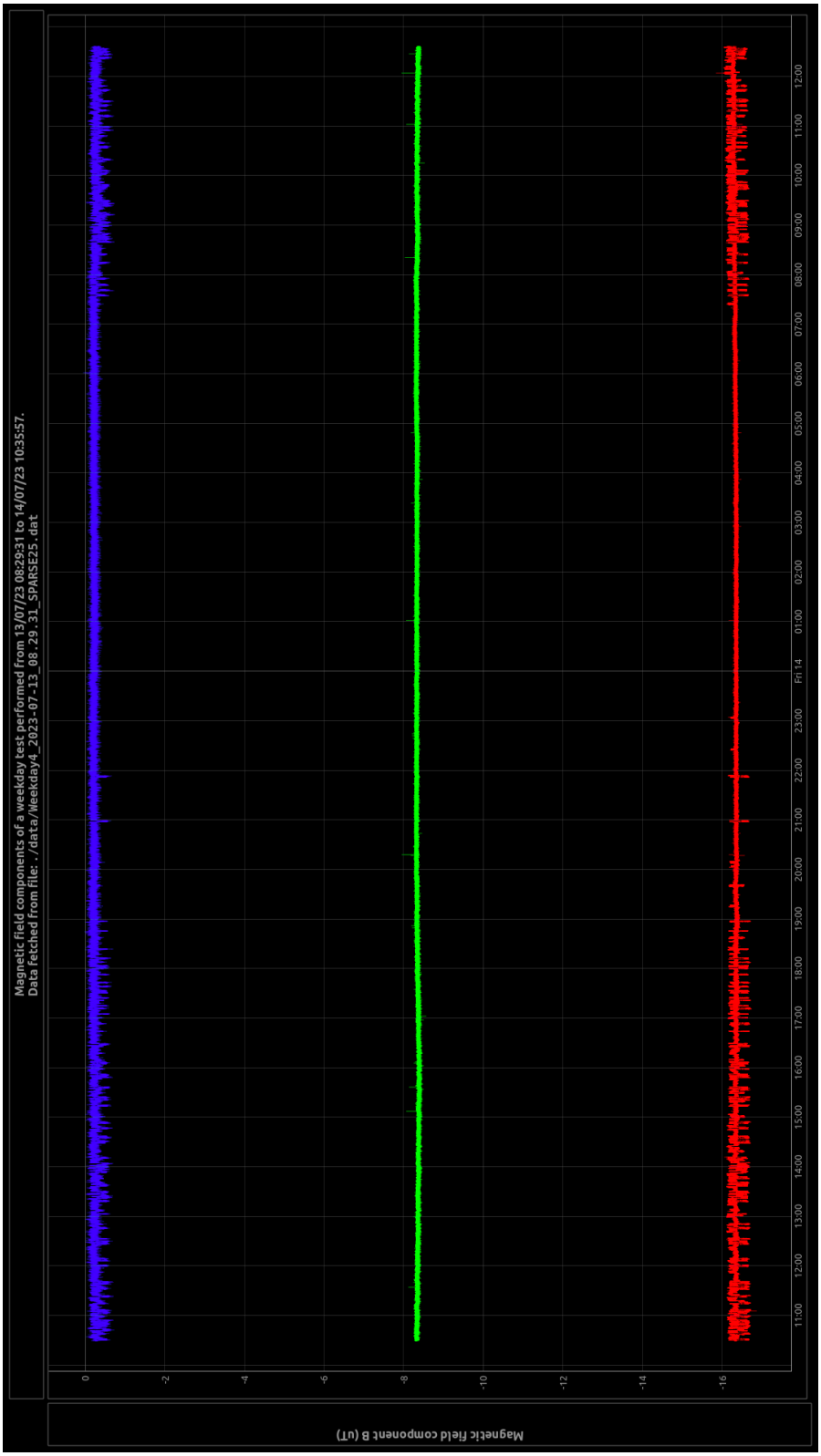


Figure K.6: [SIDEWAYS] 24 hours of measured data, with the three axial components plotted on a common axis.

Now consider Figure K.7, which is the same data, but now plotted such that the three axial components can be compared more easily. One thing immediately stands out: The X and Z components feature strong diurnal variation; this data is much noisier during daytime.

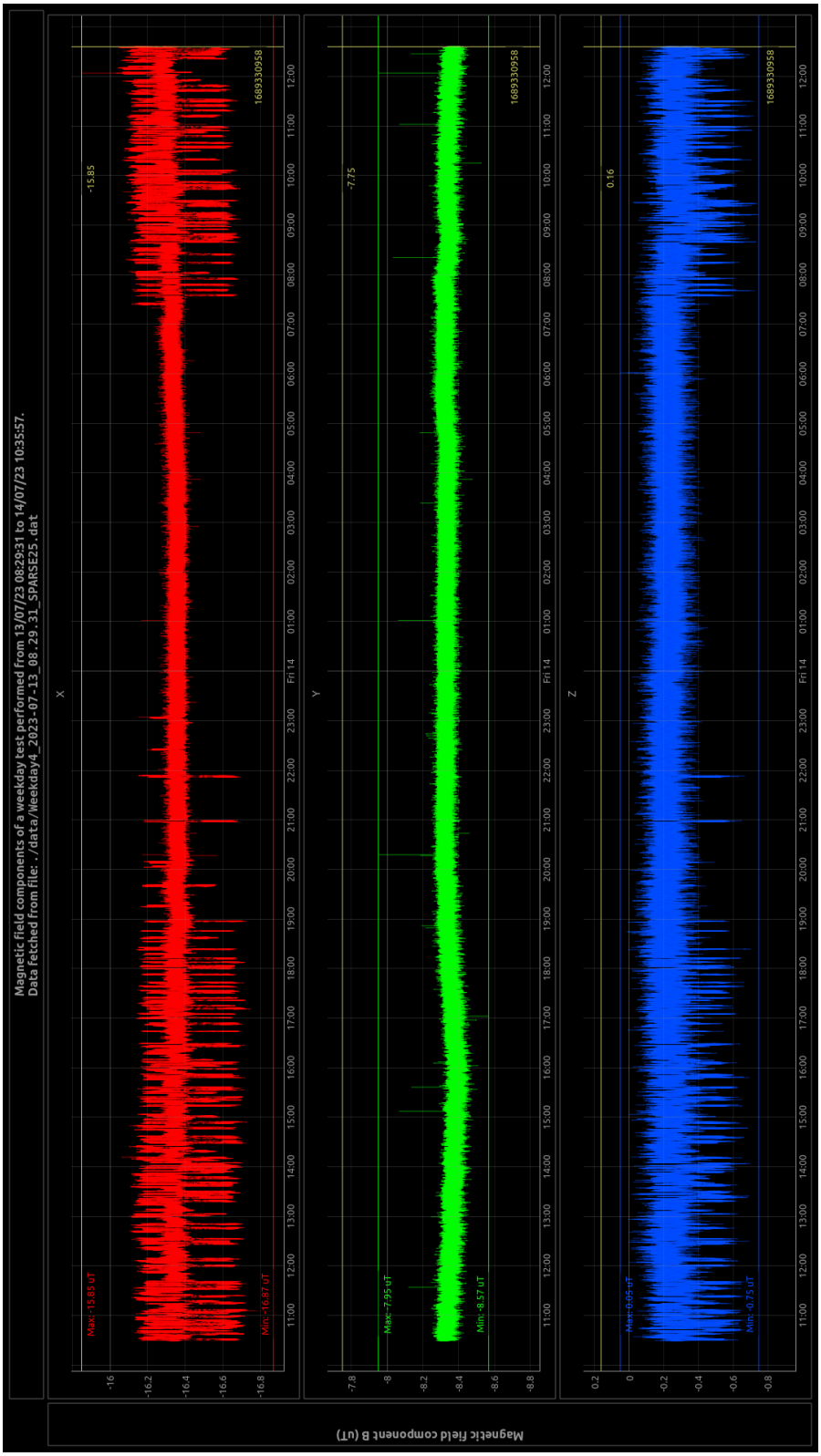


Figure K.7: [SIDEWAYS] The same data as in Figure K.6, but now plotted on three separate axes. The scale of the axes is forced to be identical, so that the proportional differences between the three axial components can be compared apples-to-apples.

When we zoom in on the noisier daytime data of the X components, we get Figure K.8. The noise starts displaying a pattern, one that strongly varies every few minutes. This is magnetic variation due

to moving elevators. One can even see stairs like behaviour around 13:52 that resembles an elevator stopping at multiple floors. However, it is not immediately obvious whether the magnetic variations are caused by the actual movement of the elevator, by the movement of some counter-weight, or whether it is some byproduct of another part of the elevator system. Still, that it is correlated with elevator operation is beyond question, and it also explains why Figure K.7 displays a large diurnal variation in variation: At night, there is nobody in the building to use the elevator, and as a result, the magnetic field data calms down to a lower noise floor.

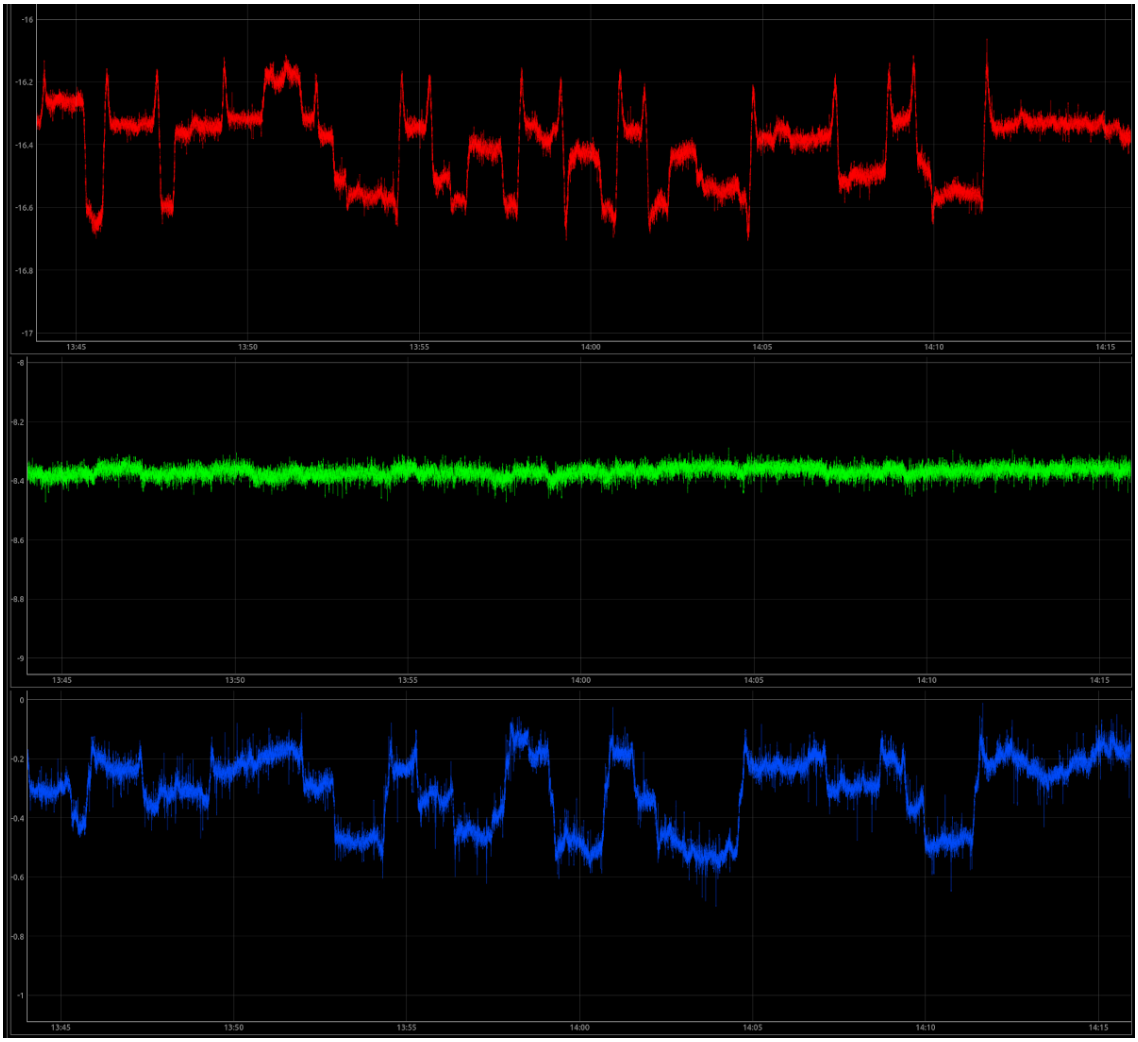


Figure K.8: The data of Figure K.6, but zoomed in on a 30 minute time interval around the lunch break in the faculty.

So, a few more results have now become apparent:

1. Although the disturbance vector moves relatively little, a large portion of the remaining motion can be attributed directly the motion of the elevator.
2. The variation due to the elevator is nearly invisible for the Y axis. Why there is a strong difference for different axes forms a clue to what exactly is causing the magnetic variation, but the author can currently not fully explain it.
3. The presence of a lower noise floor suggests that there is at least one more source that causes the magnetic field at the test area to fluctuate. The effect is weaker than the effect of the elevator, and it has no observable diurnal variation. It is also similar in scale for the X and Y axes, but stronger for the Z axis.
4. Compared to testing during daytime, during nighttime the variation boundaries reduce:

(a) For X, from 0.6 μT to around 0.15 μT, around a factor of 4 in absolute reduction
(b) For Y, little changes. Its variation still has a range around 0.15 to 0.2 μT
(c) For Z, from around 0.75 μT to around 0.4 μT.
(d) A more accurate assessment of this, using the standard deviations of the data, can be found in Figure K.9. One can see that at night, the standard deviation drops by almost an order of magnitude for the X and Z measurements.

5. Practical conclusion for testing: If you want to do magnetic testing in the cleanroom with an optimal signal-to-noise ratio, you either have to test at night, settle for testing in the Y-direction only, or convince every faculty member to take the stairs.

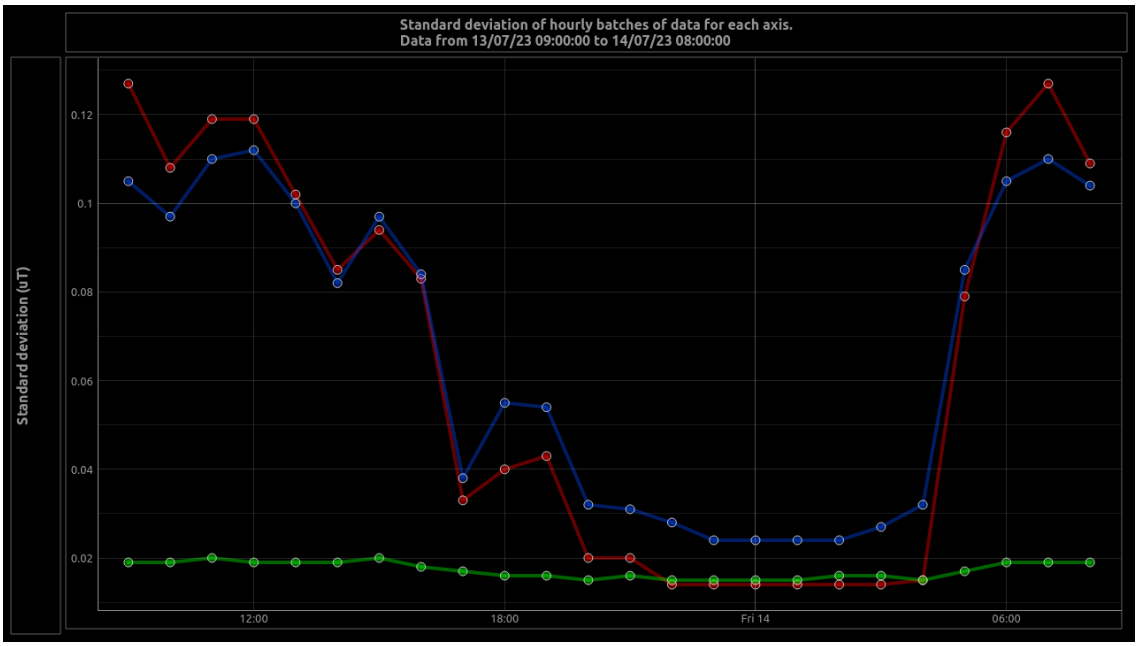


Figure K.9: Diurnal variation in standard deviation of magnetic field data, calculated by hour. Source data from Weekday4 test.

Continuing on, we can zoom in even further on the data and discover something else:

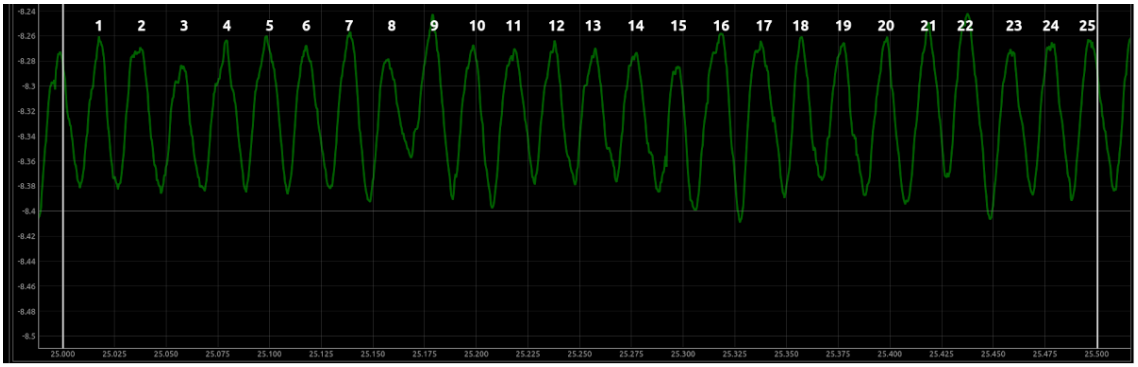


Figure K.10: 50 Hz oscillation within a half-second interval of Y-axis magnetic measurements during nighttime. Source: Weekday4 test, 2500 S/s.

It turns out that the magnetic fields at the test location fluctuate periodically. It is no coincidence that its frequency is exactly equal to the alternating current (AC) frequency of the building power network. To further investigate this, a basic spectral analysis was performed on the data: A one-sided fast Fourier transfer (FFT) was applied to the measurements on each axis, and this was plotted on a log-log grid. This results in something like Figure K.11.

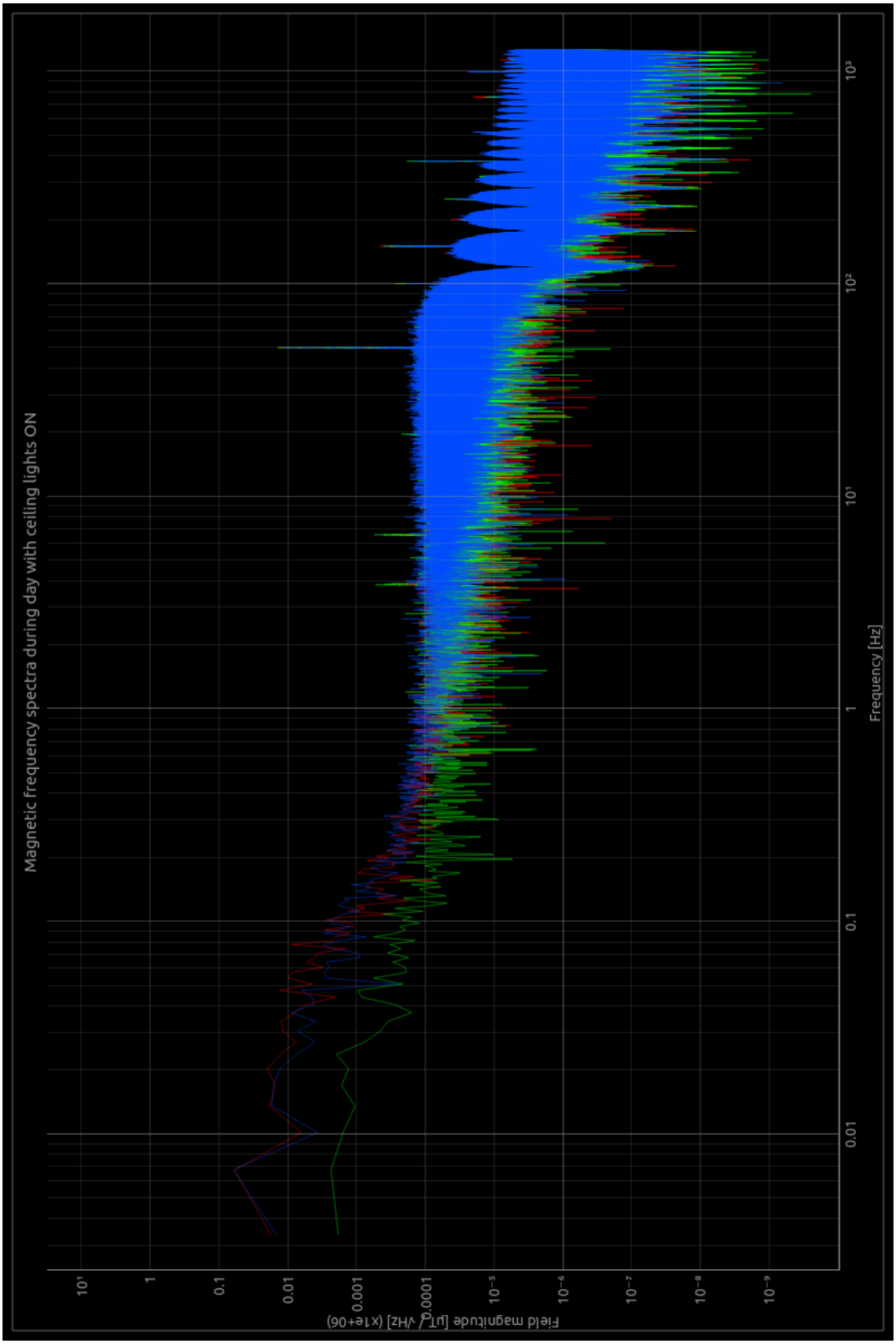


Figure K.11: [SIDEWAYS] FFT spectral data of Lights On Test 2, consisting of 5 minutes of data sampled at 2500 S/s with the cleanroom ceiling lights turned on.

This spectral plot clearly shows several peaks associated with the 50 Hz AC frequency and its higher harmonics at 100 Hz, 150 Hz, 200 Hz, 250 Hz, 500 Hz, 750 Hz, and 1000 Hz. In other words, the effect of

electronic systems in the vicinity of the Helmholtz cage are causing periodic fluctuations in the magnetic readings. Fortunately, given that the AC frequency is typically astonishingly regular, the locations of these peaks should not change, and it should be possible to filter these oscillations away.

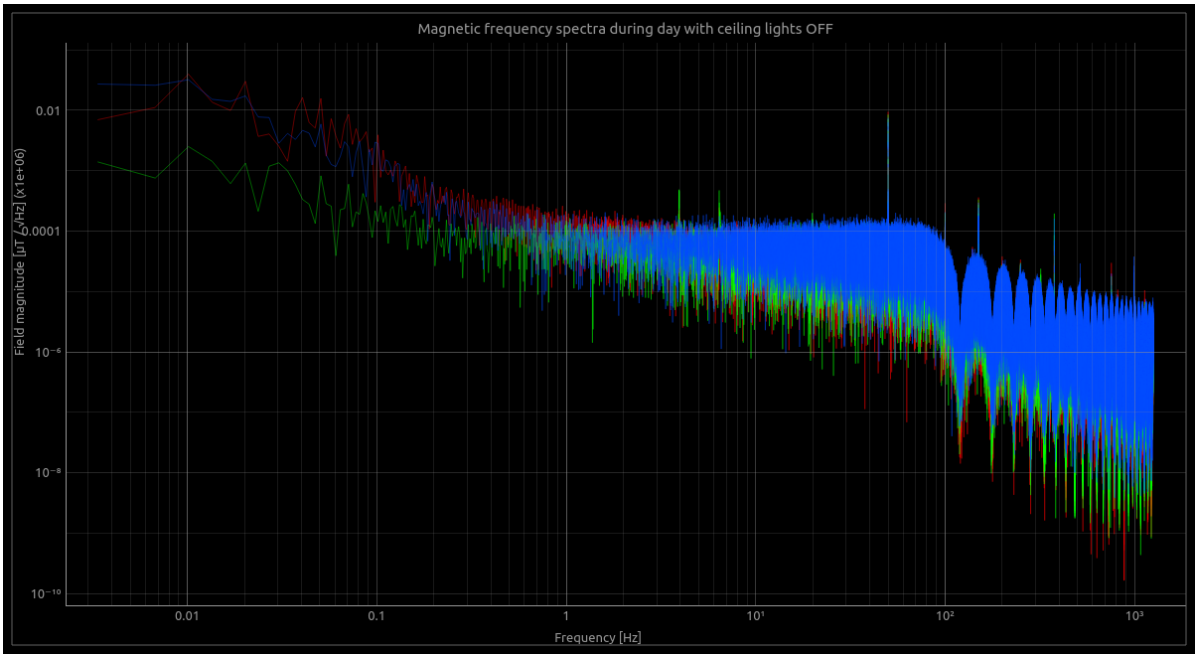


Figure K.12: FFT spectral data of Lights Off Test 2, consisting of another 5 minutes of data sampled at 2500 S/s, but this time with the cleanroom ceiling lights turned off. Compared to Figure K.11, the peaks associated with the 50 Hz AC frequency and its harmonics have been attenuated substantially.

It is interesting to note that the amplitude of the harmonic disturbances are still subject to change throughout the day. For example, Figure K.12 shows the spectral data of another 5 minute test, but this time with the ceiling lights in the cleanroom turned off. When comparing this with Figure K.11, it can be seen that the peaks associated with the 50 Hz AC frequency and its harmonics have been attenuated substantially, merely by turning off the ceiling lights. It should also be expected that at night, the amplitude of this disturbance should be diminished, as more electronic devices near the Helmholtz cage are turned off. Remember also that the main power line to the floors above runs directly behind the back wall of the cleanroom. However, this effect does not show up on the time plot in Figure K.7, and we will see that the effect is minor, if not altogether negligible.

This was not known at the time of these tests, however, and so in order to investigate the effect of other electronic devices turning off at night, a full 24 hour weekday test was scheduled in which the ceiling light were to remain on, to see the effect of all other electronic systems. However, the lights in the cleanroom were hardwired into a central system that turns off all lights at 22:00 in the faculty. This was found out after performing the Weekday2 test, whose data was mostly unusable as a result. After consulting with the building management, an electrician was sent to disconnect the cleanroom from this system for a 24 hour period, so that the test could still be carried out as planned. This is the Weekday4 test, and so any day/night differences in the spectral peaks associated with the AC power network are due to all other electronic systems except the ceiling lamps.

Let's dive deeper. Since the disturbance is so highly periodic, there is a good chance that it can be filtered out to further improve the signal-to-noise ratio of future tests. There are many ways to filter specific frequencies from datasets, such as inverse FFT methods, moving average filters, and notch filters. The notch filter was chosen here as a demonstration, since at high quality factors, it is very surgical, and it was also easy to implement.

Two 25,000 samples-long snippets (10 seconds) were taken from the Weekday4 dataset, one during the day, and one during the night. First the standard deviation of the dataset was taken without any filtering. Then, the Savitzky-Golay filter was added. Finally, a set of narrow-band, second-order notch filters was applied, each filter placed at one of the prominent harmonics of the AC power system discussed earlier.

Figure K.13 and Figure K.14 show the time plot and spectrum plot of the completely unfiltered data. This data is from the snippet taken during the night, to ensure that any elevator operations cannot affect the conclusions drawn here.

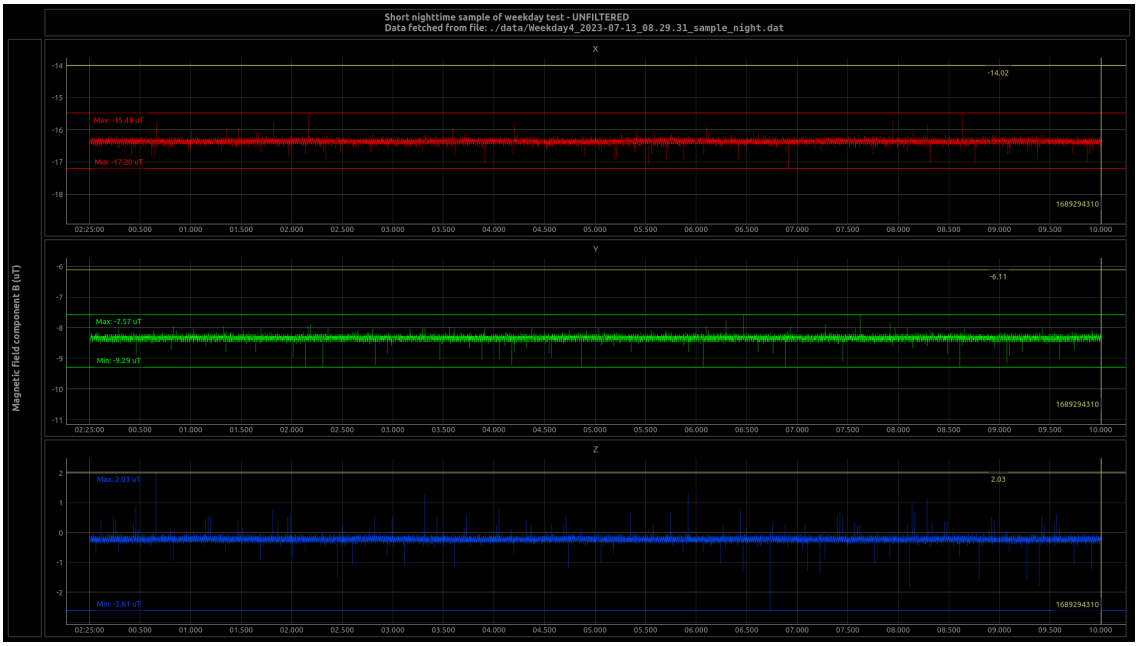


Figure K.13: Time plot of a 10-minute snippet during the night, without any filtering applied.

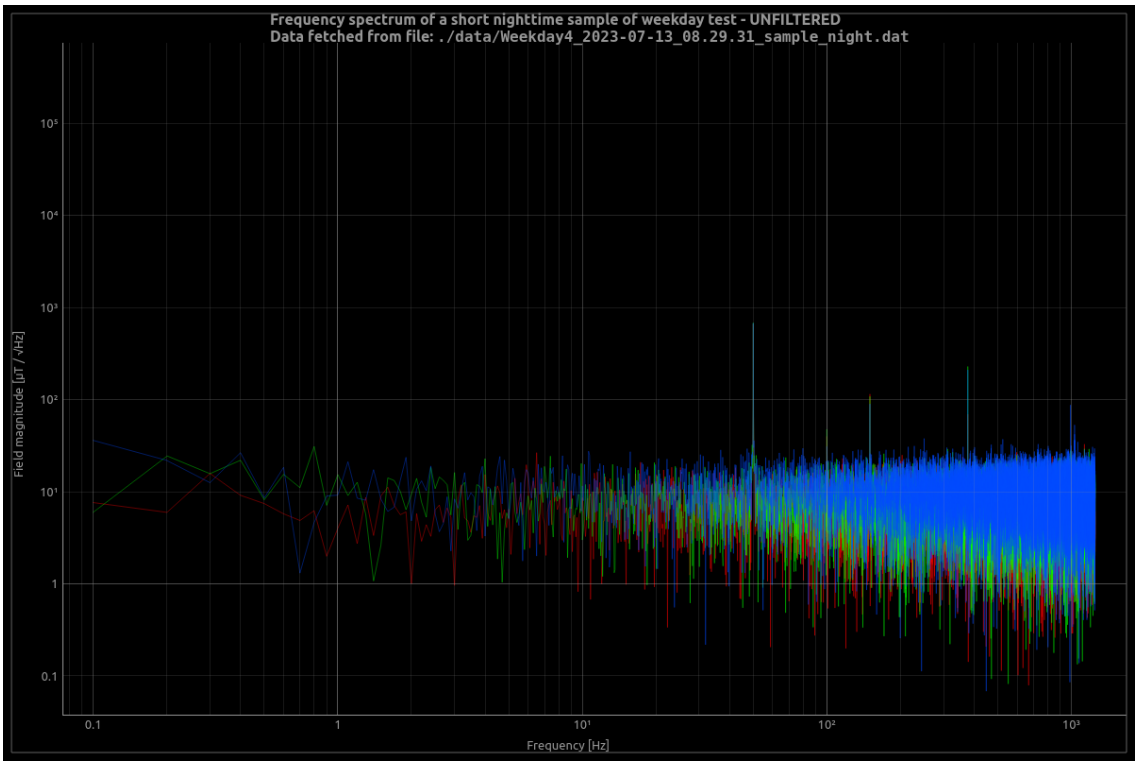


Figure K.14: Spectrum plot of Figure K.13.

Figure K.15 and Figure K.16 show the same data, but with the Savitzky-Golay filter applied. One can see from Figure K.16 that this filter has the combined effect of a low-pass filter and a comb filter.

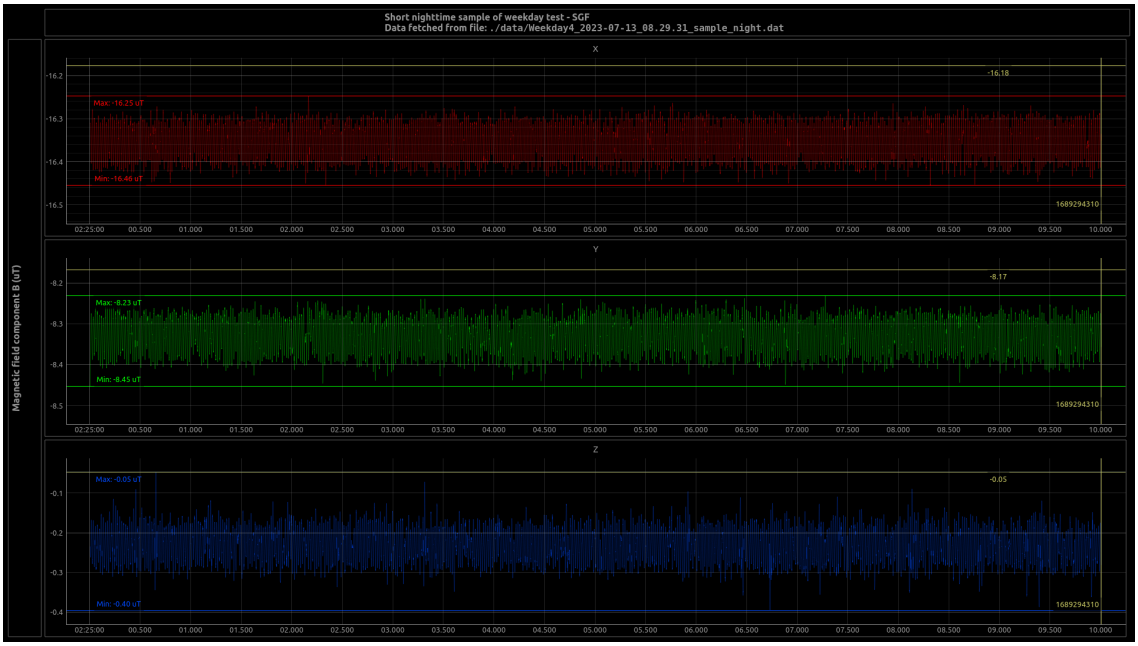


Figure K.15: Time plot of a 10-minute snippet during the night, with the Savitzky-Golay filter (SGF) applied.

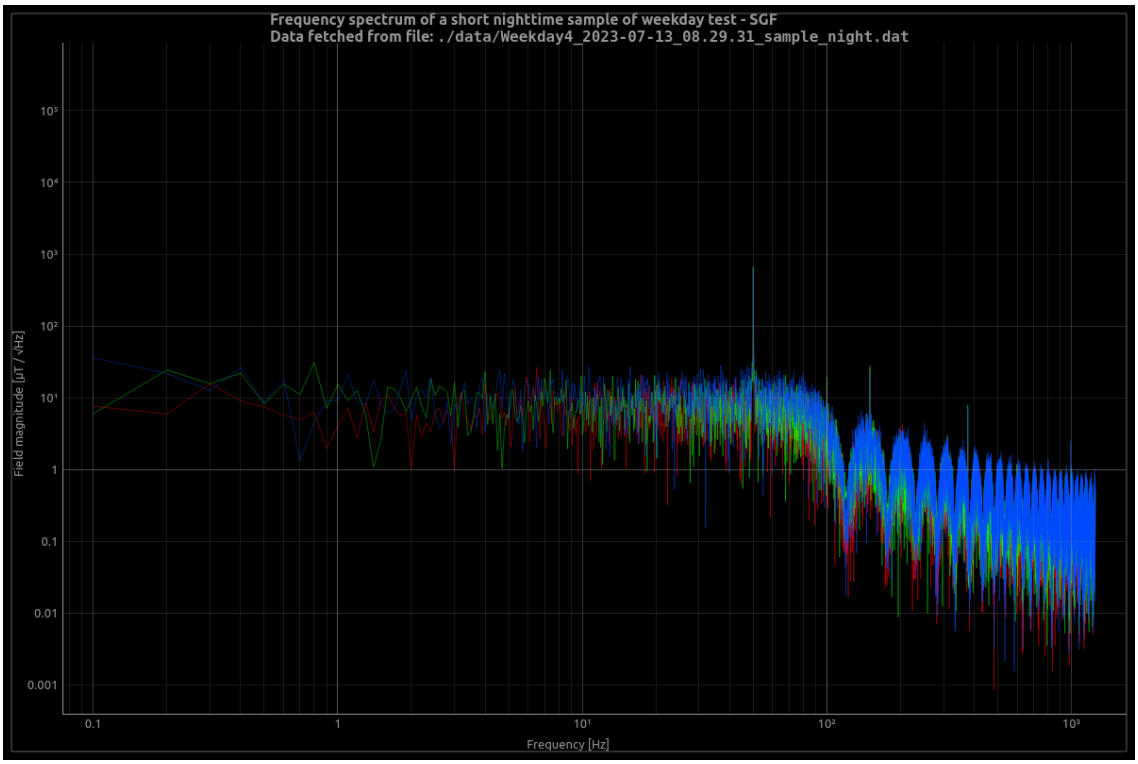


Figure K.16: Spectrum plot of Figure K.15.

Finally, Figure K.17 and Figure K.18 show the same data, but with the addition of seven narrow-band notch filters applied at 100 Hz, 150 Hz, 200 Hz, 250 Hz, 500 Hz, 750 Hz, and 1000 Hz. A glance at Figure K.18 shows that the prominent peaks associated with AC power systems have been pushed down to at least the level of the surrounding noise.

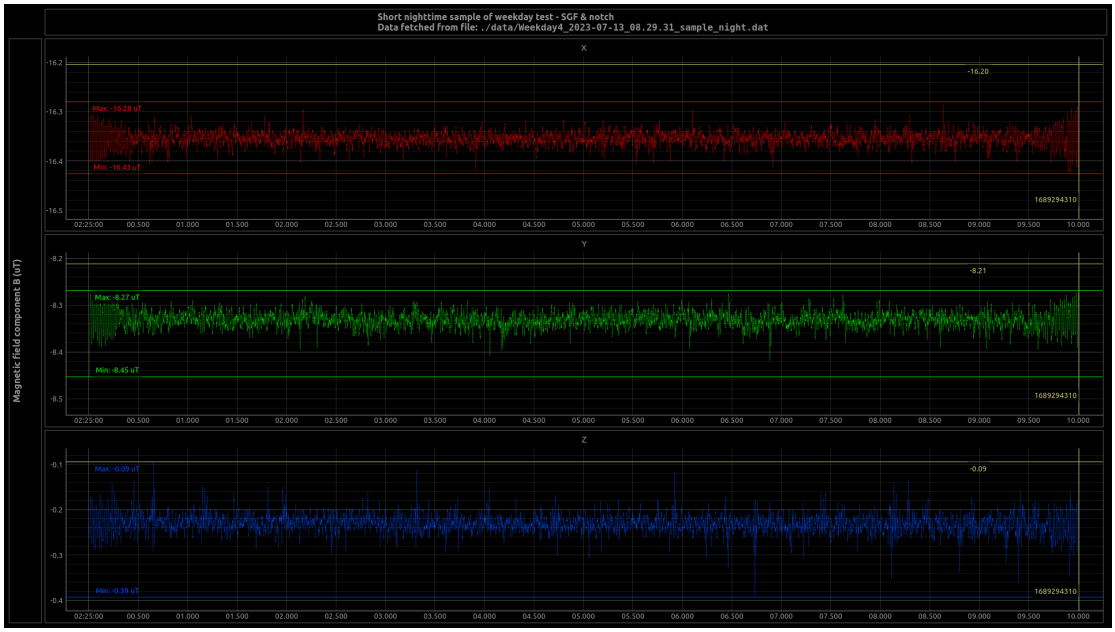


Figure K.17: Time plot of a 10-minute snippet during the night, with the Savitzky-Golay filter (SGF) and 7 notch filters applied.

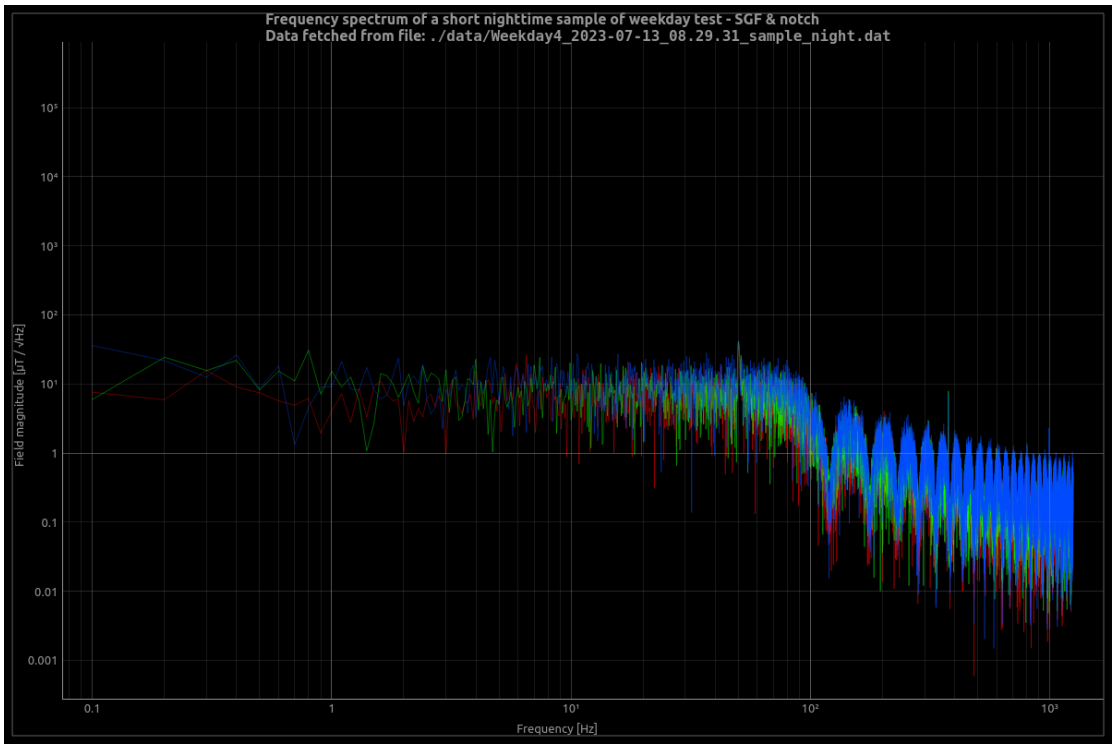


Figure K.18: Spectrum plot of Figure K.17.

Now, for all three sets, (unfiltered, Savitzky-Golay filter (SGF), and SGF+notch filters), the variance of the data was calculated. The data at the edges was discarded, because one can see from Figure K.17 that the filters are not fully applied near the edges of the domain, and this is only a 10 second sample. For this reason, the standard deviation was calculated using the middle 80% of the data only. The result is as follows:

Table K.2: Standard deviations of data at various stages of filtering. All values are expressed in μT.

	SD_X	SD_Y	SD_Z
Unfiltered, day	0.0449	0.0449	0.0482
Unfiltered, night	0.0985	0.0549	0.0528
SGF only, day	0.0228	0.0167	0.0263
SGF only, night	0.0195	0.0152	0.0150
SGF and notch filters, day	0.0051	0.0053	0.0085
SGF and notch filters, night	0.0046	0.0018	0.0038

One can see that after applying the Savitzky-Golay smoothing and the set of second-order notch filters, the standard deviation is down to below 10 nT. This is an indication that if one is interested in performing magnetic measurements at the Helmholtz cage setup with a sensitivity of 1 nT, one can get fairly close by testing at night (to avoid elevator noise), and by filtering out the periodic noise from the AC power network. The filtering method applied here is fairly basic, and it may be possible to get even closer with better filtering techniques.

The author recommends here to take these result with a healthy amount of salt. The aforementioned results are based on several unspoken assumptions, such as:

1. The massive uncertainty of the sensor data caused by the 12bit resolution of the ADC was magically fixed by the SGF.
2. The attentuation around the notch that is inherent to all notch filters has no impact on the quality of the data.
3. A 10 second interval is sufficient data to draw the aforementioned conclusions.

It may be that the numbers in Table 2 are completely worthless. Let's not gloss over the fact that the unfiltered data during the night has a standard deviation higher than the one during the day, which is difficult to explain with what has been discussed up to now. The bottom line is that without better data, it is not possible to know for certain how accurate Table 2 is. This is left to further research to determine.

However, the author is confident in the assessment that a majority of the variation seen in the signal is caused by relatively few sources (elevator and electronic devices) and that the extent of this disturbance can be known fairly well. It is a sign that despite sub-optimal conditions of the work environment, magnetic testing can be done to a relatively accurate degree, given the right testing equipment and approach.

A number of other tests were performed, and their results will be succinctly discussed here.

1. Surprisingly, the large storage rack seems to have very little effect on the constant disturbance at the test area. The difference between the "Rack Removal test" and its control test was smaller than the measured standard deviation. After all tests had been concluded, the rack was moved back to its original position.
2. The fire extinguisher on the other side of the wall was also not found to have a measurable impact on the strength of the disturbance vector.
3. The structure of the Helmholtz cage itself does not seem to exhibit latent magnetic emissions. This is true for both the coils, which were moved both very close and very far away from the test area, and the table underneath the test area. During the "Coils Wide test", the coils were spaced [1700, 1700, 1835] mm apart for the X, Y, and Z coil pairs respectively. During the "Coils Close test", it was [560, 560, 640] mm.
4. Out of interest, a series of spatial tests were performed in a 0.5 meter grid around the test area. The results can be seen in in Figure K.19. The magnitude of the disturbance varied markedly between points tested with a minimum of 17.85 μT and a maximum of 19.32 μT. There was also a marked variation in direction, especially for the points near to the back wall. The nine vectors do not seem to point to anything in particular. It is unclear what is causing the variation. It has not been investigated further due of a lack of time, and since it is not obvious to what extend it will affect future tests. Nonetheless, this would make for interesting further research.

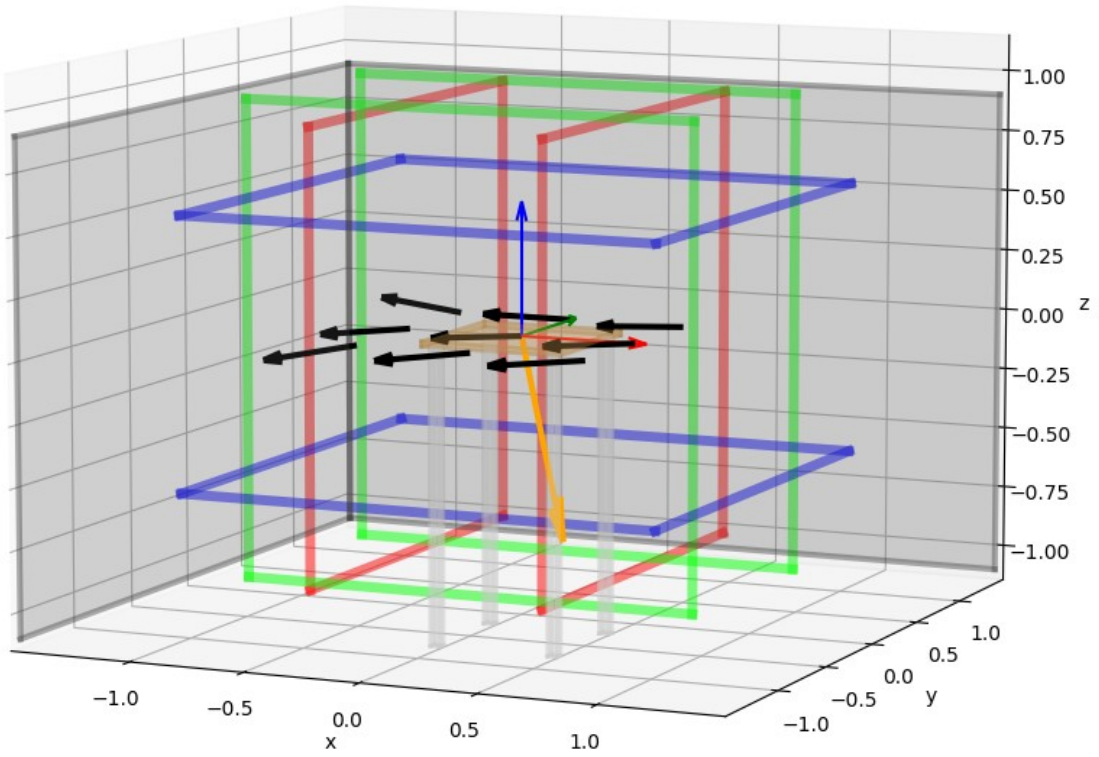


Figure K.19: The spatial test series: Nine 45+ minute measurements in a 0.5 m grid around the test area.

K.3. Instructions and tools for replication

The codebase for all tools used in this report can be found at:

https://github.com/Hans-Bananendans/Gaussmeter-Utilities

To replicate the figures and results in this report, two things are needed:

1. The toolkit in the aforementioned repository.
2. The data files of the tests. The total raw, uncompressed test data is 43.4 GB. This is not stored on GitHub. The best way to get this data is to contact the author.
3. The data files then need to be placed inside the Gaussmeter-Utilities repository in a folder named `/data/`.
4. The file `/other/Test Results.ods` provides the best overview of what files bear relation to one another.
5. Follow the instructions in the README of the repository:
   (a) (recommended) Use a Unix-based operating system. If using Windows, you may have to edit some of the hardcoded file paths to use double backslashes (\\), instead of single forward slashes (/).
   (b) (recommended) Open a new `venv`.
   (c) Clone this repository.
   (d) Install the Python dependencies listed in `requirements.txt`.

(e) If you wish to regenerate the plots, first download the magnetic test data files from the server and place them in the cloned repository, in a folder named `/data/`. All files placed in this folder are automatically ignored by git. The total size of the data files is 43 GB. If you have trouble getting access to the measurement data, send me a message.

(f) Consult `/other/Test Results.ods` to see a comprehensive list of performed tests, outcome, and what files are relevant.

(g) If you are missing certain files, check to see if any of the scripts in the root containing the word `preprocessing` can generate the files for you.

(h) Filenames containing the term `_SPARSE` indicate that the contents of the file have been downsampled.

(i) Filenames containing term `_s**of**` indicate that the file is part of a segmentation of a larger file.

(j) All tools in `/tools/` were designed to be called with Python commands, except `GaussmeterRecorder.py`, which has a CLI and is best run from the console.

K.4. Extra Figures

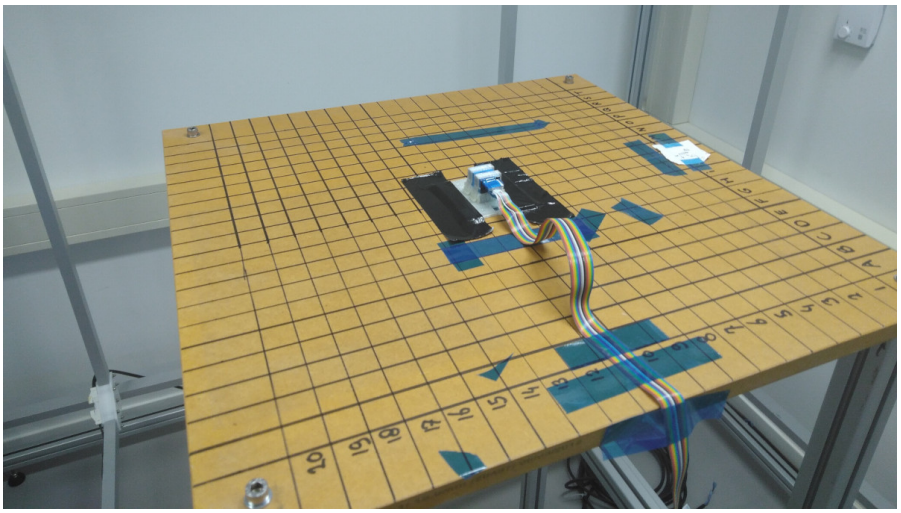


Figure K.20: The tri-axial probe of the milligaussmeter, fastened within a 3D printed retention clip, which in turn is fastened to the table at the test area. The retention clip was made to easily remove the probe in between tests if needed, without losing consistency in alignment of the probe across tests.



Figure K.21: Structural suspension jig above the Helmholtz cage frame.



Figure K.22: Helmholtz cage during the Coils Wide Test, with its three pairs of coils extended as far as they would go. Notice the absence of the storage rack, which was removed before these tests started.



Figure K.23: Helmholtz cage during the Coils Close Test, with its three pairs of coils moved as close to the test area as possible.



Figure K.24: The coils moved back to their theoretically ideal spacing for square Helmholtz coil pairs (0.5445 times the side length).

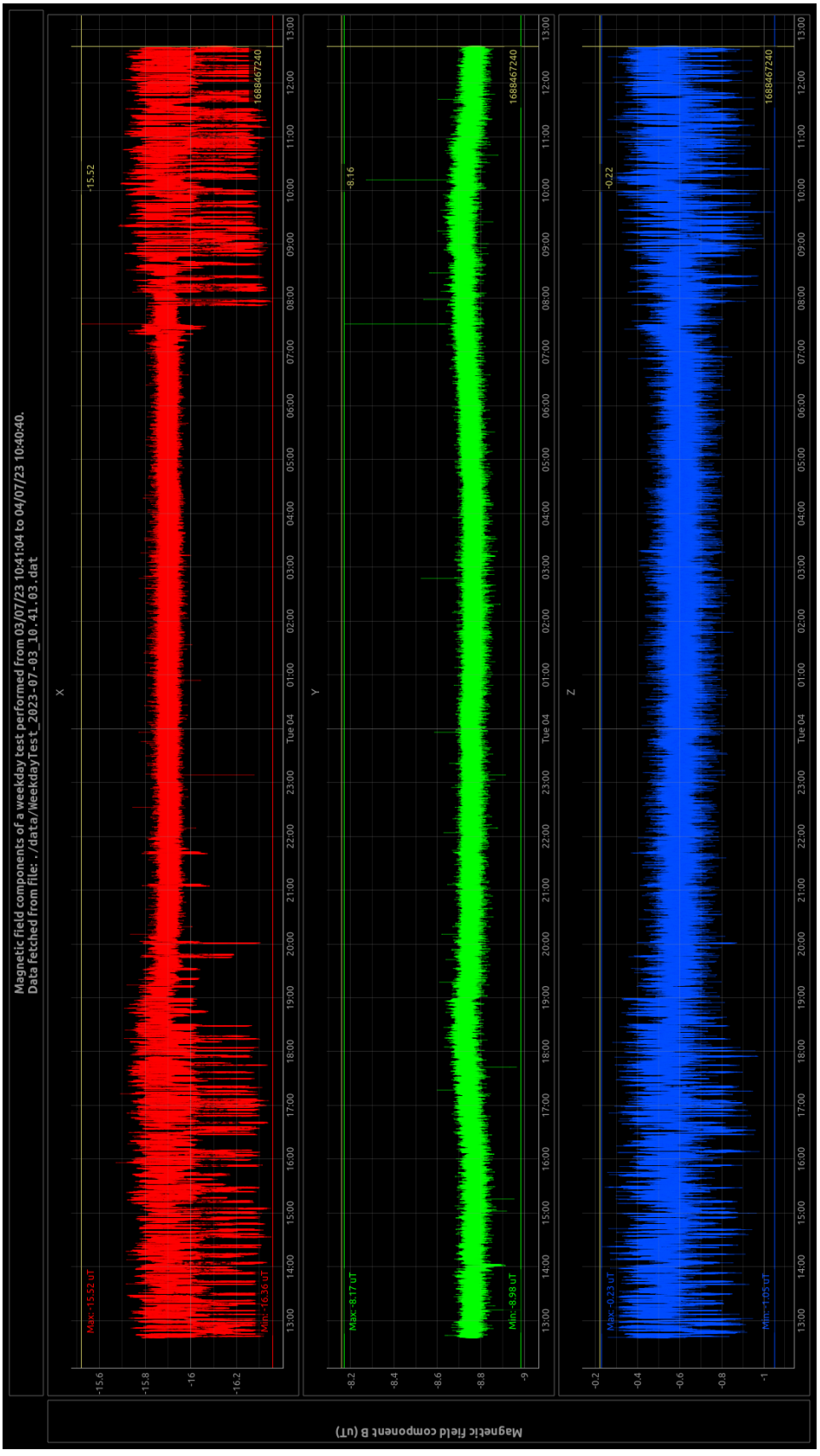


Figure K.25: [SIDEWAYS] Weekday1 test time plot

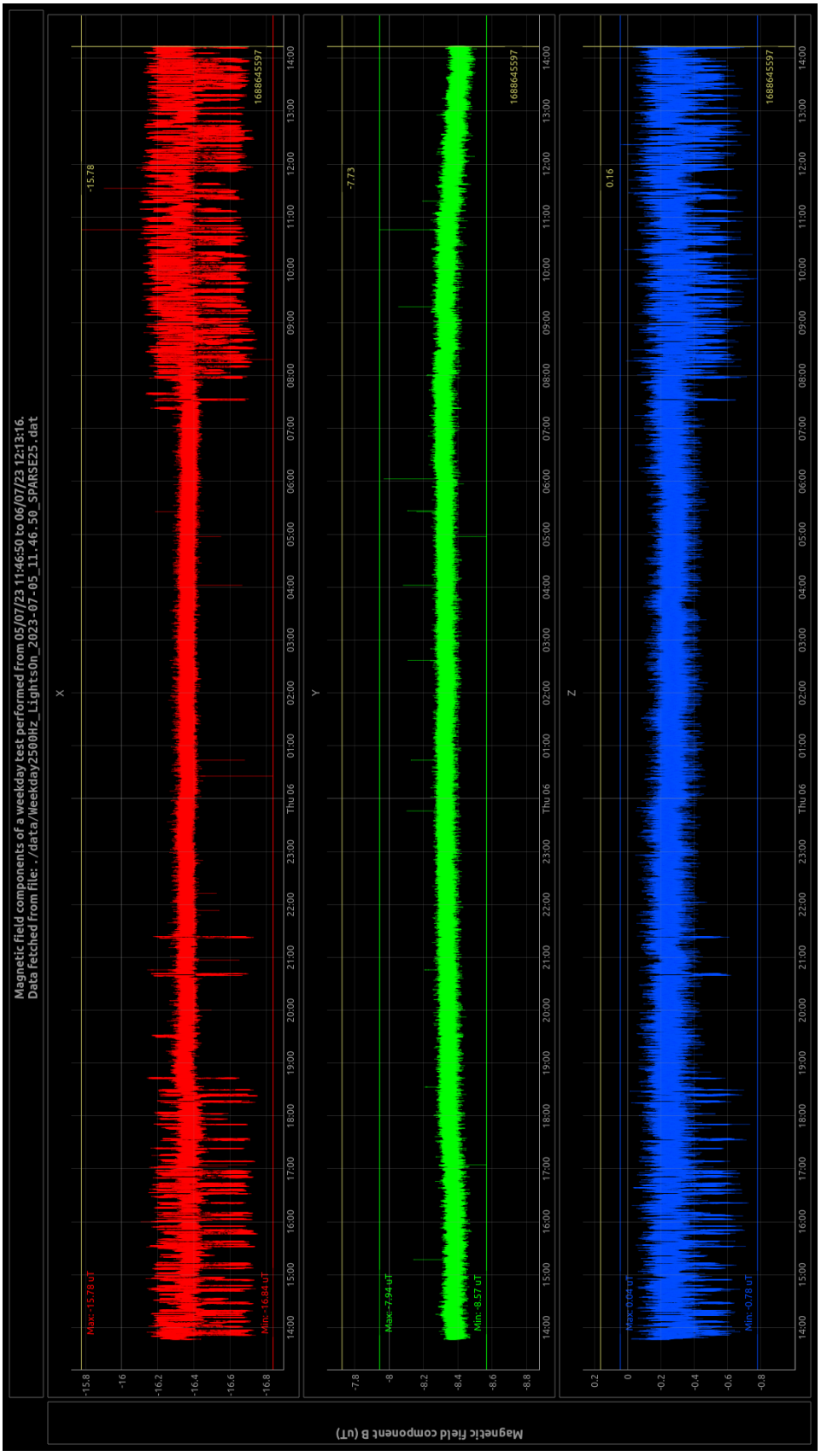


Figure K.26: [SIDEWAYS] Weekday2 test time plot

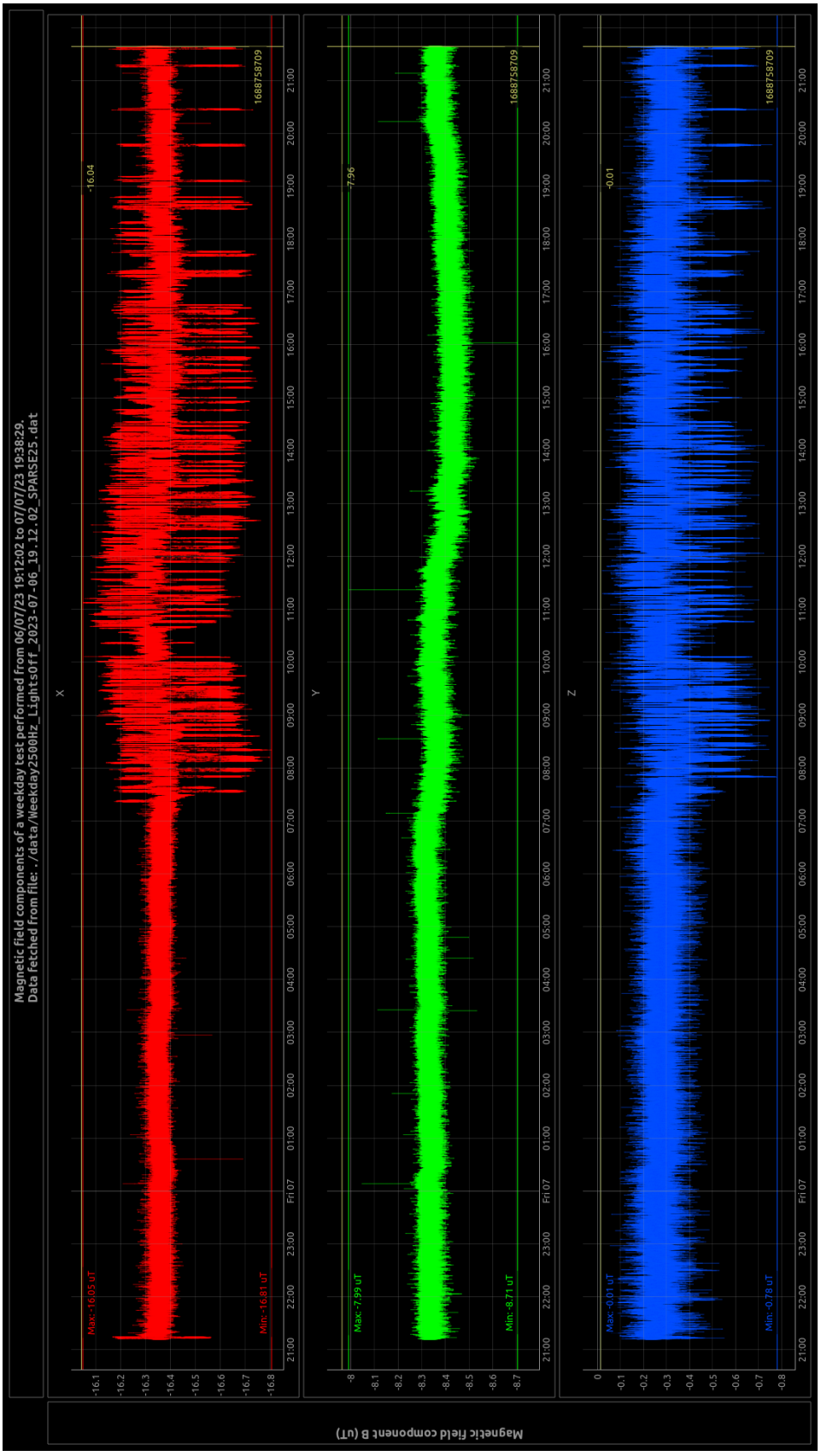


Figure K.27: [SIDEWAYS] Weekday3 test time plot

L

Dataset of Satellite Case Studies

This appendix contains the full dataset of examined satellite missions as part of the literature review of subsection 2.1.2. The analysis of this data produced Figure 2.4, which is included again below for convenience (Figure L.1). The full literature study is a separate document entirely [22].

Figure L.1 is a rather compressed way of looking at this data, which is good for finding macroscopic patterns across the whole set. However, by necessity, the figure hides many details to produce such an overview, and this information is no longer explicit from the figure. For example, for a given point in the figure we cannot see which exact satellite mission relates to it. This appendix is included to provide such details. First, the complete dataset is presented as a table, and an accompanying bibliography is included thereafter.

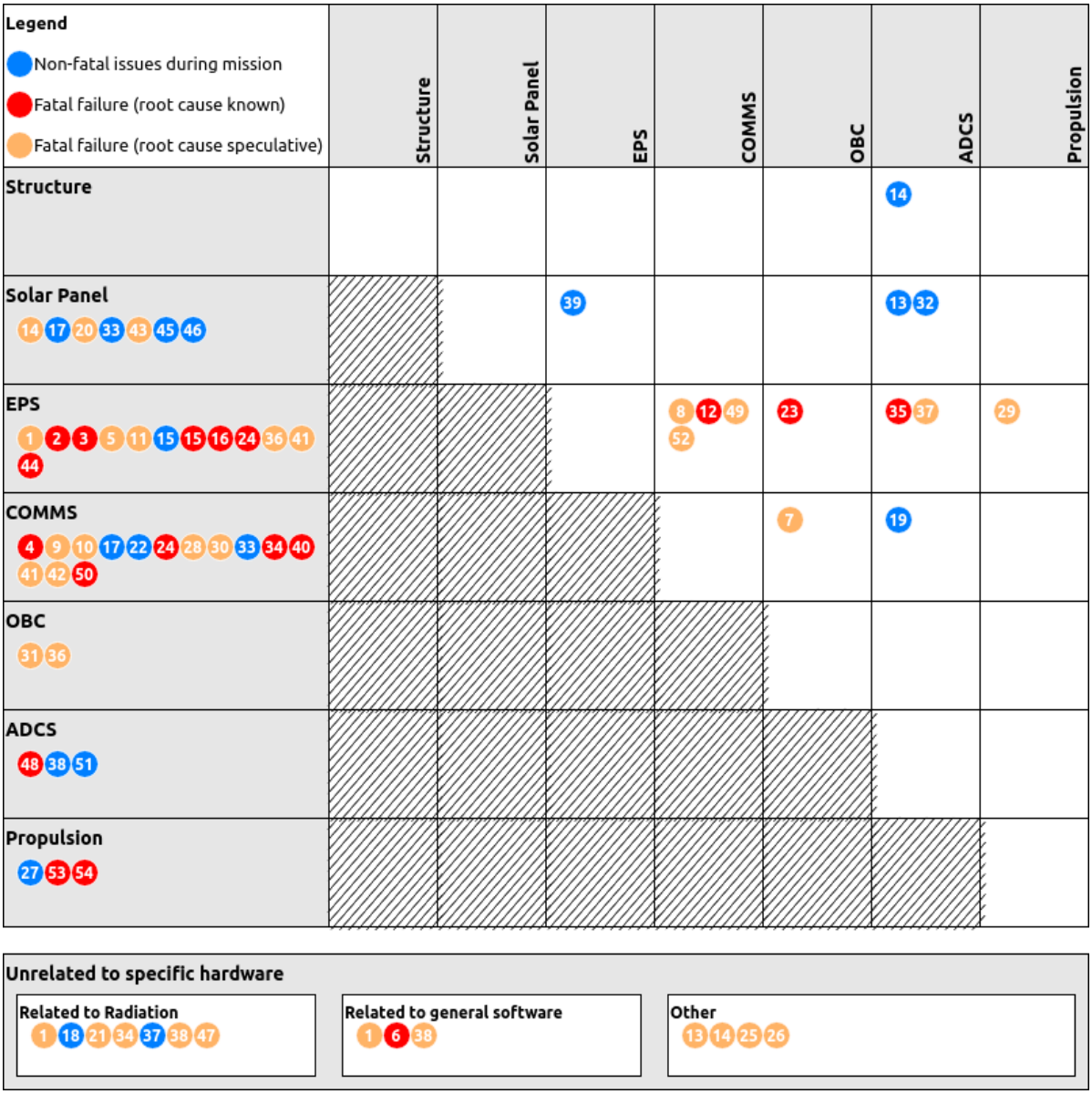


Figure L.1: Top-down approach: Collecting the reported issues of 54 CubeSat missions and mapping them onto satellite subsystems.

.

Table 1: Dataset of the top-down approach of the literature review

#	CubeSat Name	NORAD ID	Launch year	Relevant publications	Form factor	Non-fatal issues during mission	Fatal failure (root cause known)	Fatal failure (root cause speculative)
1	**Aalto2**	42729	2017	[1]	2U			Stopped responding suddenly. Team indicates as possible causes: - Radiation damage - Problem in EPS - Software problem
2	**AAUSAT3**	39087	2013	[2]	1U		After about 1.5 years of operation, the mission was terminated as battery problems caused a continuous decline in power production.	
3	**AeroCube-2**	31133	2007	[3], [4]	1U		Failure of the power subsystem	
4	**Audacy Zero**	43810	2018	[5]	3U		Problem is isolated to omnidirectional antenna system	
5	**AUTCube2**	43681	2018	[6]	1U			Contact could not be established because the main power supply was not turned on.
6	**BDSat-1**	52175	2022	[7]	1U		A problem in the BDSat-1 software caused a communication failure.	
7	**Binar 1**	49272	2021	[8]	1U			Downey et al. lists that the most likely cause of the satellite failure a faulty connection between the flight computer and the transceiver.
8	**BRICSat-P**	40655	2015	[9]	1.5U			Communication with the satellite was significantly hampered by power budget issues. As far as can be told from reports, no uplink has been established.
9	**CANYVAL-X Tom**	43136	2018	[10]	2U			Telemetry and commanding was not available "owing to several problems". The lessons learned section of Song et al. [10] lists that the beacon transmission procedures and telemetry handling were not implemented well for the mission. In addition, they were not able to test the ground station, and they did not have a CubeSat transceiver on-ground to mimic the CubeSat transmissions. The ground station turned out to be a source of problems, as they had a large 435MHz noise source near their ground station receiver, and had a difficult time receiving the satellite.
10	**CANYVAL-X Jerry**	43136	2018	[10]	1U			See entry for "CANYVAL-X Tom".
11	**CuSP**	N/A – COSPAR: 2022-156M	2022	[11]	6U			EPS/Battery
12	**CXBN**	38762	2012	[12]	2U		Weak transmit SNR owing to an anomalous low power mode	
13	**Deorbitsail**	40719	2015	[13], [14], [15], [16], [17], [18], [19], [20]	3U	High spin rates due to inherent solar panel magnetic properties		Physical disconnection of sail deployment motor cable
14	**ESTCube-1**			[21]	1U	Structure became magnetized, hampering attitude control early on, which took several months updating the control algorithms		The payload tether/solar sail could not be deployed, most likely due to some mechanical malfunction. And in addition, solar panel degradation eventually caused negative power budget and battery depletion
15	**FIREBIRD A**	39463	2013	[22]	1.5U	Battery degradation caused by overcharging	Design flaw in EPS battery protection circuit, couldn't properly boot when batteries completely depleted	
16	**FIREBIRD B**	39464	2013	[22]	1.5U		Design flaw in EPS battery protection circuit, couldn't properly boot when batteries completely depleted	
17	**IDEASSat**	47458	2021	[23]	3U	Blackout issues with S-band downlink Likely incomplete deployment of one solar panel		
18	**INSPIRE-II**	42731	2017	[24]	2U	Went quiet for 1.5 months due to damage to the communication system after a space weather event. Contact was re-established afterwards.		
19	**IT-SPINS**	49017	2021	[25]	3U	Problem with the ADCS control caused the satellite to spin up to very high rates (up to 200 RPM). ADCS has since been turned off, but it makes it very difficult to communicate with the satellite.		
20	**K2SAT**	43801	2018	[26]	3U			Likely undeployed solar panel, causing low battery voltage and power budget issues

21	**KickSat**	39685	2014	[27], [28]	3U			The watchdog was reset, likely by a radiation event, which caused the primary mission objective, the deployment of the sprite payloads, to not be triggered before the satellite burned up in the atmosphere again.
22	**KickSat-2**	44046	2018	[29]	3U	The primary objective (the deployment of the sprites) was successful, but very poor reception of packages from the satellite, which is suspected to be caused by failed antenna deployment.		
23	**KrakSat**	44426	2019	[30], [31]	1U		Power budget issues causing battery to drain during mission and enter perpetual boot loop (the team documented over a million reboots over the course of the satellite lifetime)	
24	**LabSat**	51086	2018	[32]	3U		Unfortunately, due to problems with communication and power supply, the planned biological experiments could not be carried out.	
25	**LINCS1 A**	48902	2021	[33], [34]	12U			Unable to establish communication with both vehicles. One possible theory is that vented helium from the launch vehicle interfered with some MEMS components on-board.
26	**LINCS1 B**	48903	2021	[33], [34]	12U			See entry for "LINCS1 A".
27	**LunaH-Map**	N/A – COSPAR: 2022-156J	2022	[35], [36]	6U	Stuck valve in propulsion system		
28	**LunIR**	N/A – COSPAR: 2022-156K	2022	[37]	6U			Comms?
29	**MarCO 1 (WALL-E)**	43458	2018	[38]	6U			Attitude control issues due to leaky thruster causing inability to send and receive commands. ADCS sun sensors unable to point solar panels at the sun, causing battery charging issues
30	**MicroMAS-1**	40457	2014	[39]	3U			COMM, transmitter failure
31	**MOVE**	39439	2013	[40]	1U			After one month in orbit, the satellite stopped sending AX.25 packets. The reason for this is suspected to be a booting problem with the OBC.
32	**MOVE-II**	43780	2018	[41], [42]	1U	High spin rates believed to be caused by magnetic dipole inherent in solar panel design		
33	**MOVE-IIb**	44398	2019	[42]	1U	One solar panel not deployed successfully, causing degradation of power budget. Also, TX power is significantly lower than expected. Most likely hypothesis implicates the telecommunication system as the culprit.		
34	**NANOSATC-BR2**	47950	2021	[43]	2U		Operating temperatures were well below nominal range, causing significant fluctuations in the uplink and downlink frequencies. This made it impossible to command the satellite properly, and as a result, the payloads were not commissioned. Some time after, the I2C communication bus seems to have locked up, an issue that could not be resolved. Contact was lost at a time that coincided with a pass of the satellite across the SAA.	
35	**Omotenashi**	N/A – COSPAR: 2022-156D	2022	[44], [45], [46], [47]	6U	High spin rates caused low power input, attempts to detumble caused batteries to drain excessively (unsure if design fault or operator fault)		
36	**OrigamiSat-1**	43933	2019	[48]	3U			Suspected that EPS or main microcomputer OBC is the problem
37	**Phoenix CubeSat**	45258	2019	[49]	3U	Likely bit flip in the radio and resulting stream of packets at the end of the first day depleted battery below voltage cut-off level.		After battery drain on first day, OBC isn't responding to any command other than pings. Cause unclear, but given that pings were only performed successfully during day, it suggest that ADCS is not pointing the solar panels at the sun, preventing the battery from charging sufficiently to power the spacecraft during eclipse.

38	**PicSat**	43132	2018	[50]	3U	- ADCS		Two most likely causes for the ending of the mission is either radiation-caused latch-up, or a software boot loop. It is plausible that this mission could have been saved by better ADCS and/or electronics radiation testing, judging by the failure analysis published by the team.
39	**POPSAT-HIP1**	40028	2014	[51]	3U	Mission degradation caused by a combined problem with the solar panels, battery charging capability and the EPS itself, which caused the satellite to be only active in direct sunlight.		
40	**PW-Sat**	38083	2012	[52], [53], [54], [55], [56]	1U		Issue with communication module	
41	**QARMAN**	45263	2019	[57]	3U			UHF transceiver malfunction and battery thermal failure as potential failure root causes.
42	**RAAF M1**	43779	2018	[58], [59]	3U			Unknown reason for DOA, but COTS radio is suspected
43	**RAX**	37223	2010	[60]	3U			Inactive after 2 months of operation, caused by solar panel degradation
44	**Robusta 1**	38084	2012	[61], [62]	1U		Battery charging system	
45	**SENSE SV1**	39388	2013	[63]	3U	Solar panel deployment failure		
46	**SENSE SV2**	39392	2013	[63]	3U	Solar panel deployment failure		
47	**skCube**	42789	2017	[64]	1U			Suspected that a radiation event caused the OBC to fail.
48	**SNaP3**	39468	2013	[65]	3U		Spacecraft could not successfully detumble, as the rotational momentum caused by orbital injection was too high.	
49	**Snuglite-II**	52899	2022	[66]	3U			An error in the electrical part of the satellite caused problems with the antenna and the comms module data transmission and reception.
50	**STARE-A**	38765	2012	[67]	3U		Communication issues prevented operation	
51	**SwissCube**	35932	2009	[68], [69]	1U	- High rotation rates after launch, which couldn't be measured by the ADCS as it fell outside the sampling rate range.		
52	**TechEdSat-8**	44032	2018	[70]	6U			A power anomaly caused transmissions from the satellite to end.
53	**Tyvak-0171**	46295	2020	[71]	6U		Propulsion system failed	
54	**Tyvak-0172**	46267	2020	[71]	6U		Propulsion system failed	

References

1. University, A. *Aalto-2 no longer responds to commands* https://phys.org/news/2017-06-aalto-longer.html.
2. ESA. *AAUSat3 (Aalborg University CubeSat-3)* https://www.eoportal.org/satellite-missions/aausat3#status-of-mission.
3. NASA. *CubeSats Backgrounder* (NASA). https://www.nasa.gov/centers/wallops/pdf/326337main_Cubesat%20backgrounder.pdf.
4. Krebs, G. D. *AeroCube 2* https://space.skyrocket.de/doc_sdat/aerocube-2.htm.
5. Krebs, G. D. *Audacy 0* https://space.skyrocket.de/doc_sdat/audacy-zero.htm.
6. Of Technology, A. U. *10th - Why can't I find radio waves? (translated from Japanese)* https://www.aut-cubesat.jp/autcube2/10.html.
7. Sensors, B. *BDSAT-2* https://www.bdsensors.cz/en/company/news/detail/bdsat-2.
8. Downey, F. *et al.* Binar Space Program: Binar-1Results and Lessons Learned. *SSC22-WKII-06.* https://digitalcommons.usu.edu/smallsat/2022/all2022/56/ (2022).
9. DK3WN. *PSAT/BRICsat status* https://www.satblog.info/psatbricsat-status/.
10. Song, Y., Lee, Y. R., Park, J.-P. & Park, S.-Y. in *2018 SpaceOps Conference* (May 25, 2018). doi:10.2514/6.2018-2636. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2018-2636. https://arc.aiaa.org/doi/abs/10.2514/6.2018-2636.
11. Hill, D. *Artemis I Payload CuSP CubeSat Mission Update* https://blogs.nasa.gov/sunspot/2022/12/08/artemis-i-payload-cusp-cubesat-mission-update/.
12. ESA. *CXBN (Cosmic X-ray Background Nanosatellite* https://www.eoportal.org/satellite-missions/cxbn#mission-status.
13. Spaceflight101.com. *DeOrbitSail* https://spaceflight101.com/spacecraft/deorbitsail/.
14. Stohlman, O. R. *et al.* in *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* (2013). doi:10.2514/6.2013-1807. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2013-1807. https://arc.aiaa.org/doi/abs/10.2514/6.2013-1807.
15. Stohlman, O. R., Schenk, M. & Lappas, V. Development of the deorbitsail flight model. doi:10.2514/6.2014-1509 (2014).
16. Lappas, V. *et al.* DEORBITSAIL: De-orbiting of satellites using solar sails. *2011 2nd International Conference on Space Technology.* doi:10.1109/icspt.2011.6064667 (2011).
17. For Amateur Radio, T. N. A. *DeOrbitSail Satellite Unsuccessful in Deploying Sail* https://www.arrl.org/news/deorbitsail-satellite-unsuccessful-in-deploying-sail.
18. Steyn, W. H. & Jordaan, H. W. An active attitude control system for a drag sail satellite. *Acta Astronautica* **128,** 313–321. doi:https://doi.org/10.1016/j.actaastro.2016.07.039. https://www.sciencedirect.com/science/article/pii/S0094576516300819 (2016).

19. Krebs, G. D. *DeOrbitSail (DOS)* https://space.skyrocket.de/doc_sdat/deorbitsail-1.htm.
20. AMSAT-UK. *DeOrbitSail Update and Initial Camera Image* https://amsat-uk.org/2015/11/13/deorbitsail-update-and-initial-camera-image/.
21. Pustõnski, V. *ESTCube-1 ceased working after 2 years in orbit* https://web.archive.org/web/20180311140551/https://www.eas.ee/kosmos/en/estonian-space-office/news/article/389-estcube-1-ceased-working.
22. Klumpar, D. *et al. Flight System Technologies Enabling the Twin-CubeSat FIREBIRD-II Scientific Mission* in (Aug. 2015). doi:10.13140/RG.2.1.3368.4562.
23. Chang, L. C. *et al.* Integration, Launch, and First Results from IDEASSat/INSPIRESat-2 – A 3U CubeSat for Ionospheric Physics and Multi-National Capacity Building. *SSC21-WKIII-06* (July 9, 2019).
24. Cairns, I. H. *et al. INSPIRE-2 and Lessons Learned* (ed 2019, A. C. C.) https://www.acser.unsw.edu.au/sites/acser/files/CAIRNS_INSPIRE_CubeSatCoonf_2019_2.pdf.
25. Dolan, S. *Building and Operating CubeSats - Successes and Struggles : Montana State's Experience with FIREBIRD & IT-SPINS (presentation)* https://www.youtube.com/watch?v=I4FetOoJwhM&t=11583s.
26. K2SAT. *Official K2SAT Twitter Account* https://twitter.com/K2SAT_KAIST/.
27. Manchester, Z. *KickSat's Current Status* https://www.kickstarter.com/projects/zacinaction/kicksat-your-personal-spacecraft-in-space/posts/831509.
28. Manchester, Z. *KickSat has reentered* https://www.kickstarter.com/projects/zacinaction/kicksat-your-personal-spacecraft-in-space/posts/843807.
29. ARRL. *KickSat-2 is Alive and Being Tracked* https://www.arrl.org/news/view/kicksat-2-is-alive-and-being-tracked.
30. Musiał, A., Markowski, D. & Życzkowski, J. KRAKsat Satellite Mission - Lessons Learned [EN]. doi:10.13140/RG.2.2.14725.50408 (Dec. 2019).
31. KrakSat. *Awaria satelity* https://www.kraksat.pl/space/.
32. Graja, A. *et al. LabSat - The first Polish satellite to provide biological research on the LEO* in (2022). https://iafastro.directory/iac/archive/browse/IAC-22/D1/5/72468/.
33. Erwin, S. *Space Development Agency, General Atomics eye options after setback in laser comms experiment* https://spacenews.com/space-development-agency-general-atomics-eye-options-after-setback-in-laser-comms-experiment/.
34. Comga, U. *Post on NASA Space Flight Forum* https://forum.nasaspaceflight.com/index.php?topic=51898.msg2338790#msg2338790.
35. NASA. *NASA's LunaH-Map Post-Launch Update* https://blogs.nasa.gov/lunah-map/2022/11/22/nasas-lunah-map-post-launch-update/.
36. Dickinson, D. *Status update: Artemis 1's CubeSat Missions* https://skyandtelescope.org/astronomy-news/status-update-artemis-1s-smallsat-missions/.
37. Duda, L. *Comment on Twitter* https://nitter.net/LMSpace/status/1600926003232182280#m.

38. Laboratory, N. J. P. *The MarCO Mission Comes to an End* https://www.jpl.nasa.gov/news/the-marco-mission-comes-to-an-end.

39. Bill Blackwell, J. P. *New Small Satellite Capabilities for Microwave Atmospheric Remote Sensing: The Earth Observing Nanosatellite-Microwave (EON-MW)* Aug. 8, 2015. https://digitalcommons.usu.edu/smallsat/2015/all2015/126/.

40. PEØSAT, A. R. .-. *First-Move* https://www.pe0sat.vgnet.nl/satellite/cube-nano-picosats/first-move/.

41. Rückerl, S. *et al. First Flight Results of the MOVE-II Satellite* in (Aug. 2019).

42. Volk, L. *et al. New Results and Lessons Learned from the MOVE-II and MOVE-IIb CubeSats* in (Aug. 2022). https://digitalcommons.usu.edu/smallsat/2022/all2022/224/.

43. Space, B. *NANOSATC-BR2: Brazilian Space apura informações sobre o status do satélite* https://brazilianspace.blogspot.com/2021/08/nanosatc-br2-brazilian-space-apura.html.

44. Hernando-Ayuso, J., Campagnola, S., Yamaguchi, T., Ozawa, Y. & Ikenaga, T. OMOTENASHI trajectory analysis and design: Landing phase. *Acta Astronautica* **156,** 113–124. doi:https://doi.org/10.1016/j.actaastro.2018.10.017. https://www.sciencedirect.com/science/article/pii/S0094576518301139 (2019).

45. Krebs, G. D. *OMOTENASHI* https://space.skyrocket.de/doc_sdat/omotenashi.htm.

46. Club, J. H. R. *Radio Info* https://www.isas.jaxa.jp/home/omotenashi/JHRCweb/jhrc.html.

47. JAXA. *Omotenashi - Latest News* https://www.isas.jaxa.jp/home/omotenashi/news.html.

48. OrigamiSat1. *Official OrigamiSat1 Twitter account* https://twitter.com/OrigamiSat1.

49. Rogers, S. *et al. Phoenix: A CubeSat Mission to Study the Impact of Urban Heat Islands Within the U.S.* in (July 28, 2020). https://digitalcommons.usu.edu/smallsat/2020/all2020/12/.

50. Nowak, M. *et al.* Short life and abrupt death of PicSat, a small 3U CubeSat dreaming of exoplanet detection. *arXiv: Instrumentation and Methods for Astrophysics.* doi:10.1117/12.2313242 (2018).

51. DK3WN. *POPSAT-HIP1 beacon* https://www.satblog.info/popsat-hip1-beacon-2/.

52. Mosdorf, M., Kurowski, M., Cichocki, A., Mosdorf, Ł. & Kocoń, M. PW-Sat on-board flight computer, hardware and software design (2019).

53. ESA. *PW-Sat (CubeSat - Warsaw University of Technology)* https://www.eoportal.org/satellite-missions/pw-sat#spacecraft.

54. Wolanski, P. *PW-Sat First Polish Satellite* 2012.

55. Stolarski, M., Kurek, K., Dobrowolski, M. & Graczyk, R. *SYSTEM KOMUNIKACJI I STEROWANIA SATELITY PW-SAT* in *PRZEGL D TELEKOMUNIKACYJNY - ROCZNIK LXXXII - i WIADOMO!CI TELEKOMUNIKACYJNE - ROCZNIK LXXVIII - nr 6/2009* (2009). https://marcinstolarski.pl/mstolars/publikacje/MS1_PTiWT_6_2009.pdf.

56. Wutsquare. *The PW-Sat project marked the beginning of a new era for the Polish space sector* https://wutsquare.wordpress.com/2015/01/13/the-pw-sat-project-marked-the-beginning-of-a-new-era-for-the-polish-space-sector/.

57. Denis, A., Turchi, A., Bernd, H., Magin, T. & Chazot, O. *QARMAN re-entry CubeSat: from design and pre-flight testing to post-flight lessons learned* in (2022). https://iafastro.directory/iac/archive/browse/IAC-22/B4/9-GTS.5/69263/.

58. Horton, A. *Comment on Twitter* https://twitter.com/vacant3rdman/status/1181744729735626753.

59. Boyce, R. *Update on the M1 spacecraft (post on Linkedin)* https://www.linkedin.com/feed/update/urn:li:activity:6514437076136660992.

60. ESA. *RAX (Radio Aurora eXplorer)* https://www.eoportal.org/satellite-missions/rax#mission-status.

61. Perez, S. *et al. ROBUSTA, a Student Satellite to Serve the Radiation Effects Community* in (2019). https://digitalcommons.usu.edu/smallsat/2009/all2009/85/.

62. Krebs, G. D. *Robusta 1, 1B* https://space.skyrocket.de/doc_sdat/robusta.htm.

63. Sondecker, G., Frazier, A., Kim, J. & Abramowitz, L. R. *SENSE: The USAF SMC/XR Space Environmental NanoSatellite Experiment* in (2014). https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=2990&context=smallsat.

64. SOSA. *Tlačová správa: Prvá slovenská družica zaznamenala na orbite prvý problém* https://kozmonautika.sk/2017/07/10/tlacova-sprava-prva-slovenska-druzica-zaznamenala-na-orbite-prvy-problem/.

65. Anderson, M. A. *et al. Army Decade in Space* in (June 4, 2020). https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=4712&context=smallsat.

66. Team, S. C. *SNUGLITE-II press release bulletin (translated from Korean)* https://snuglitecubesat.wixsite.com/website/forum/bodojaryo/2022-07-22-snuglite-ii-bodojaryo.

67. ESA. *STARE (Space-based Telescopes for the Actionable Refinement of Ephemeris)* https://www.eoportal.org/satellite-missions/stare#mission-status.

68. Pittet, J.-N. *et al. Space debris attitude determination of faint LEO objects using photometry: SwissCube's cubesat study case* in (2017). doi:https://doi.org/10.7892/BORIS.106923. https://conference.sdo.esoc.esa.int/proceedings/sdc7/paper/291/SDC7-paper291.pdf.

69. EPFL. *SWISSCUBE - The first swiss satellite* https://archiveweb.epfl.ch/swisscube.epfl.ch/.

70. Mares, J. R. R. *SJSU Virtual Reality, View from a Cube Satellite (presentation)* https://www.youtube.com/watch?v=kQHg2fs8rBE&t=24203s.

71. Werner, D. *EchoStar reports Jupiter-3 delay and nanosatellite failures* https://spacenews.com/echostar-q4-2020/.