Development of a Synchronous Sensor Suite for Use in Test Setups

An analysis of test setup verification through model development

by

Benjamin Lewis

to obtain the degree of Master of Science

at the Delft University of Technology

Student number: 5916666

Project duration: March, 2025 – December, 2025

Thesis committee: Prof. Dr. E. Gill, TU Delft, supervisor

M.Sc L. Farçoni, Deutsches Zentrum für Luft- und Raumfahrt e.V.

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft DLR

Contents

List of Figures

List of Tables

Abstract

At the Deutsches Zentrum für Luft- und Raumfahrt (DLR), a test bench is being designed and constructed. The software and electrical hardware solutions needed present an interesting challenge, particularly in the area of sensor synchronization. Designs for the software and electrical hardware systems are discussed, and then the executed construction of these designs is examined. An additional problem is the issue of verifying that the test setup properly adheres to requirements. A literature gap in space test setup verification is identified, regarding the selection of a rigorous verification approach and the development of verification test plans. In order to address this gap, a method of determining what verification approach should be used, and a systematized format for developing verification test plans is created. In order to determine the efficacy of these methods, the desired test setup is used as a model example. A verification approach for the requirements is made, and then test plans for the selected requirements are written using the developed methodology. These test plans are subsequently executed, and the efficacy of the test plans and their creation methodology are evaluated. From the evaluation criteria for these test plans, it can be seen that the standardized test plan methodology does a good job of ensuring that that requirements and metrologic traceability is maintained, and that there is room for improvement in the isolation of influencing factors.

Nomenclature

Abbreviations

Abbreviation	Definition
API	Application Programming Interface
COTS	Commercial Off-The-Shelf
cRIO	Compact Reconfigurable IO
DLR	Deutsches Zentrum für Luft- und Raumfahrt
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
IO	Input/Output
NI	National Instruments
PSU	Power Supply Unit
ROD	Review Of Design
DUT	Device Under Test
TDMS	Technical Data Management Solution
TTL	Transistor-Transistor Logic
VI	Virtual Instrument
V&V	Verification and Validation

Symbols

Symbol	Definition	Unit
...		
σ	Standard Deviation	[ ]

1

Introduction

Currently at the Deutsches Zentrum für Luft- und Raumfahrt's (DLR's) Institut für Raumfahrtsysteme, research and development of a new test setup has been started. The research presented here forms a significant part of this test setups development cycle. This setup needs to capture data from multiple sensors in a time synchronous manner for post-analysis, and the development of the software and electrical hardware used for this purpose is discussed in detail in later chapters. Because of the nature of the desired devices to be tested, the requested performance capabilities of this test setup are rigorous. Thus, a methodology of validation and verification of the performance of this test setup needs to be developed.

The issue of validation and verification presents an interesting challenge. The performance of the final system must meet the requirements set by DLR, and thus a testing procedure for evaluating the compliance of the test setup is needed. What is the best methodology for evaluation? Ideally, the test setup will be designed such that the verification and validation methods are planned ahead of time. The literature review presented explores previous research and industrial standards related to the verification and validation of test setups in aerospace applications in order to provide a background for the desired verification and validation processes.

This thesis is divided into six chapters covering more specific topics. First, the rest of Chapter 1 covers the overall objectives from DLR, the literature review, the research question that can be derived from the literature review, and the requirements set by DLR. Then, in Chapter 2 the general methodology and approach of research is detailed. The general software and electrical design are discussed in Chapter 3, and the end results of this design are then shown in Chapter 4. The results related to verification testing are disclosed in Chapter 5, followed by an analysis and conclusion in Chapter 6.

1.1. DLR Research Objectives

Ultimately, the test setup being verified in this research needs to fulfill the objectives and requirements set by DLR. Rigorous requirements are shown in Section **??**, but these specifics of, e.g. sensor characteristics, are not necessarily "objectives". In order to understand the context of the provided requirements in Section **??** and the setup

design decisions made in Chapter 3, it is useful to discuss the broad goals of the test setup being used as a model project. The objectives of this project from DLR's perspective can be described as:

Primary:

1. Functional software and electrical hardware for synchronously capturing data from a test setup

Secondary:

1. Ability to use different devices as the Device Under Test (DUT)
2. Data is recorded in a format conducive to analysis
3. All recorded data is correlated in time
4. Test is user controllable (e.g. different test profiles and custom DUT settings)
5. Utilizes what already available hardware it can (i.e. NI cRIO)
6. Runs headless (i.e. no secondary computer is required to run test)

1.2. Literature Review

This section discusses the results of the literature review with relation to the design and V&V of test setups. While this test setup is for specific devices, relevant information can be found in more general applications. Thus, the literature included here discusses the design of test setups in general, which can then be applied to this specific use case.

Before discussing specific literature, it is useful to provide context as to what is meant by the Verification and Validation processes in a space engineering context. NASA provides a useful definition that provides the distinction between the two: Verification is a process that shows compliance with requirements, while Validation is a process that proves the product is able to fulfill its intended purpose (Deiss and Bowman, 2023). This distinction is incredibly important in the two contexts that a product will find itself in: supplier development and customer usage. Verification is a supplier process, while Validation is a customer process. While DLR is acting as both supplier and customer for this project, this particular development portion is focused on requirements fulfillment. Thus, a Verification approach is desired.

The Mars Exploration Rover (MER) project was a NASA Mars lander project in the early 2000's with an extremely abbreviated development cycle in order to meet the required launch deadline. *Verification and Validation of Mars Exploration Rover Surface Capabilities*, discusses the V&V process used for MER's surface operations hardware (Welch, 2005). The verification items used were derived from requirements, and organized around specific functions of the MER. From these items an incompressible test list, which had to be successfully completed in order for launch, was created. Due to timeline and fidelity constraints and concerns, tests were performed on the actual flight.

In the *Design and Construction of Test Benches for Small Scale Aerospace Systems*, a series of three test benches for CubeSat systems is designed and evaluated (Cortes et al., 2020). The design and construction of these benches was split into three distinct

phases: Needs identification, design prototyping, and operation verification. Needs identification was based on research into the operational requirements of the previous systems. The design prototyping step consisted of Computer Aided Design (CAD) ideation and analysis, along with rough construction. Operation verification was done through the use of known sensors to test the accuracy of adherence to commanded performance.

The development of a testbed for a Rotating Synthetic Aperture (RSA) telescope CubeSat is described in *Development of a Hardware-In-The-Loop Testbed for Rotating Synthetic Aperture Telescopes* (Hernandez et al., 2022). This process begins with the establishment of mission objectives, which then flow down to system performance objectives and an operating profile. From this profile, the testbed objectives were then laid out. In order to assure testbed performance, it was broken up into two subsystems (actuators and sensors) which were verified independently and then integrated.

In *GNSS Test Bench Calibration for Space Service Volume Applications*, the design and performance of a test bench for GNSS systems is discussed (Piccolo and Menzione, 2024). The design of this test bench begins with system requirements, such as communication systems specifics. These requirements are then used to inform the development of subsystem tests, such as one that tests antenna pattern effects. These subsystem tests are then integrated into a single hardware-in-the-loop test.

Some industrial testing standards are also useful to investigate in this discussion, especially those focused on test setups. Kavlico, a sensor manufacturer, has a published standard for the development of Aerospace test equipment (KAVLICO, 2020). This standard outlines the general rules that should be followed when developing a setup for the testing of aerospace devices. Much of these relate to the calibration of the setup, along with demonstrable accuracy. For instance, a minimum test accuracy ratio of 4:1 is nominally required, i.e. the test equipment needs to be 4 times more accurate than the sensor being tested. When calibrating, the facilities used must have standards that are directly traceable to the US National Institute of Standards and Technology (NIST). It also states that equipment calibration must be kept well up to date to ensure measurement accuracy. In relation to the project of interest, this standard illustrates how test setups accuracy is ensured through traceability to a fixed standard.

EUROLAB is a non-profit organization that consists of over two dozen European institutes, and publishes short "Cook Books" that provide advice for conforming to the International Organization for Standardization (ISO) standards. The first of these deals with the "Selection, Verification, and Validation of Methods" (Eurolab, 2020). Some of the advice presented in this document are relevant for this project, such as a three-step systematic approach towards validation planning: Distinguish between test method and specimen production/processing, consider test/measurement factors, and consider supplementary changing factors. The validation should describe which factors are significant, why, and how they are handled. Two main validation principles to use are outlined: scientific knowledge to describe factors involved, and reference materials to create a traceable train of measurements. For verification, Eurolab recommends that it should be documented in such a way that proves that the verifying laboratory is able to achieve the required performance necessary to carry out the verification method.

Provided examples include an estimation of repeatability, the characteristics of used test instrumentation, the qualifications of the test operator, etc.

ISO/IEC 17025:2017 is the current document describing the "General requirements for the competence of testing and calibration laboratories" from the International Standards Organization (ISO, 2017). These are the standards that EUROLAB is advising upon. These standards discuss many portions of the setup of testing/calibration labs, but several relevant standards are mentioned. The first of which is the traceability of measurements to metrological standards, which is discussed in several of the previous literary sources as well. The second is a set of methods to use for validation, either independently or in combination. These methods include:

1. Calibration using reference standards/materials
2. Systematic assessment of influential factors
3. Testing method robustness though a variation of controlled parameters
4. Comparison of test results with those achieved from other validated methods
5. Interlaboratory comparisons
6. Evaluation of measurement uncertainty based on the theoretical principles used

ECSS-E-ST-10-02C is a standard for space engineering verification from the European Cooperation for Space Standardization (ECSS, 2018). It describes in detail the principles and requirements for the verification of space engineering technologies. Four verification approaches are described, and each verification should use at least one approach:

1. Test (including demonstration)
2. Analysis (including similarity)
3. Review-of-Design
4. Inspection

The test method consists of measuring the systems performance under a simulated environment similar to the expected environment during actual operation. The analysis method involves a theoretical or empirical evaluation. Review-of-Design is the useage of records or evidence to prove that the requirement for verification has been met. Inspection is the visual determination of physical characteristics.

From these examples in literature, a general methodology used in the development of test benches can be described. The first step is the establishment of test setup requirements. These are obviously closely tied with the mission and system requirements, but here a distinction can be made in process responsibility. Mission requirements are typically one of the first things established after the broad concept has been created. These requirements can be described as the ones necessary for the mission goal to be achievable. For instance, in a scientific mission to map the lunar surface, these requirements include the orbit of the satellite, presence of an imaging system, and the ability to communicate data back to a ground station. The system requirements are

derived from these mission requirements and are more specific, such as the specific fidelity of the imaging system used. The system requirements are what informs the actual hardware choices that will be tested in the test setup. The test setup requirements can thus be described as the methodologies used to confirm the fulfillment of system requirements in a traceable, repeatable manner. An example would be that the test needs to prove that the imaging system has the required fidelity, and be able to prove this with a level of uncertainty that is acceptable.

Once the requirements of the test setup have been established, the practicalities of achieving those requirements are investigated. This is the "how" in response to the "why" posed by the test requirements. The solutions used depend on the tests themselves, and those relevant to this project's particular case are discussed in the following section. Ultimately, the requirements for the test bench are limited by the implementation possibilities. No test bench will be able to deliver the exact environmental conditions that the device under test will experience. Thus, the design of test systems needs to account for the fact that the tests themselves need to be verified. The discussed literature covers the flow down from mission requirements to test requirements, but discussion of the verification of the test systems themselves is brief and not present in many of the papers.

Industrial standards provide a more detailed description of the methodologies used for test bench verification. The discussed literature frequently focuses on calibration and metrological traceability. In order for a proper verification and validation procedure, the equipment used in the methodology needs to be accurate in a way that is fully traceable back to some known measurement. This consistency is important for the validity of the procedures used, and forms a theoretical basis on which test bench verification can be designed. Thus, the verification is structured in such a way to tie back to fundamental metrological values. There are also some general design standards provided, such as a 4:1 test accuracy ratio, how validation factors should be incorporated, and generally acceptable validation methodologies. These industrial standards provide a more definitive starting point in the design of test bench verification, but ultimately the design of test bench verification tests is still left as an open-ended question without a specific procedure.

1.3. Research Questions

The literature discussed in Section 1.2 elaborates on how verification was handled on various space test systems, but the specifics of the process are generally vague. The derivation of the verification tests followed some procedure, but the exact details of the process are left out. The industrial standards give several specific requirements and general rules to follow when designing test bench V&V, but ultimately fall short of a rigorous formal approach. These standards either consist of features that the test benches need to have that must be verified (such as accuracy when compared to the DUT) or more broad recommendations (such as how to consider significant factors). The previously discussed industrial standards can be divided into two different topical levels: what the general system verification process should be, and what features a verification test should have. It can be seen that there is a gap in between these two levels, in that the selection of verification approach and the specific design process for a verification test are not described or provided in detail. This literature gap, combined with the research objectives from DLR, result in several research questions:

Table 1.1: Research Questions

RQ1	What software design solution can be used to fulfill DLR's objectives and requirements for this test setup?
RQ2	What electrical hardware design solution can be used to fulfill DLR's objectives and requirements for this test setup?
RQ3	How can the derivation of specific verification methodologies for test setups be systematized?
RQ3.1	What method should be used for selecting the verification approach?
RQ3.2	What standard process should be used for creating verification test plans?

The design decisions made which answer RQ1 and RQ2 are discussed in detail in Chapter 3, and the execution of this design is covered in Chapter 4. The methodology used to answer RQ3 and its subquestions are elaborated on in Chapter 2, and the results are discussed in Chapter 5.

RQ1 and RQ2 cover the more practical aspects of this project, relating to the design and construction of the model system. RQ3 and its subquestions relate to the literature gap found in the literature review in Section 1.2. This theoretical question can be answered through the development and verification of the DLR desired system. Here, systematized refers to the process in which a methodical approach to a task is established.

2

Verification Methodology

The third research question and its subquestions are focused on the verification of test setups, with the DLR desired system serving as a useful model. In order to address this research question, a methodology for developing these verification processes is needed. This methodology is based on an industrial standard selected as the systemic background, namely ECSS-E-ST-10-02C. Then, a methodology used to determine the verification approach is developed. A systematic method for deriving verification plans is then developed.

The development of the model system itself is feature oriented, in that the specific features are developed, and then integrated into the whole. For example, the method of determining the temperature is developed as a discrete phase, from ideation to implementation to verification, and is then integrated into the current setup. This development structure is discussed further in Chapter 3.

2.1. Systematic Background

ECSS-E-ST-10-02C provides a systematic background to follow during verification planning. According to this standard, the first step that shall be done during verification planning is establishing the verification approach. This approach establishes:

- "What" are the products and requirements to be covered by the verification process
- "How" to verify them
- "When" to implement the chosen verification methodology

The approach developed for this project establishes the answer to these questions ("What", "How", and "When") in two different processes: Verification Method Determination, and Verification Test Plan Derivation. The specifics of these processes are detailed in Sections 2.2 and 2.3. The specifics of these processes represent the main theoretical development in this project.

ECSS-E-ST-10-02C also discusses how the system should be decomposed for verification. This decomposition will vary for the complexity of the system to be tested, and the

current project being used as a model requires three levels:

1. **Equipment**: Base level functionalities (e.g. Reading/writing to a port, saving a file, etc)
2. **Subsystem**: Combination of base functions that perform a specific system function (e.g. full serial communication and processing routine, analog sensor reading, etc)
3. **Integrated System**: System features as a whole that depend on the interactions of various subsystems (e.g. Multi-source sensor synchronization, GUI system interactions, etc)

Combined with the feature-based development elaborated on in Chapter 3, this project decomposition provides a general sequence in which to address requirements.

The verification stages listed in ECSS-E-ST-10-02C (qualification, acceptance, pre-launch, in-orbit, and post-landing) are more tailored to a launch vehicle or satellite, rather than an aerospace test setup. For the current portion of the project (i.e. initial design and prototyping), qualification is the major verification stage of focus. Qualification is defined by ECSS-E-ST-10-02C as the stage in which "the supplier shall demonstrate that the design, including margins, meets the applicable requirements" (ECSS, 2018). This makes up the majority of the verification processes being taken with the current project. The ECSS-E-ST-10-02C provides four product categories that determine what level of qualification program is required:

Table 2.1: ECSS-E-ST-10-02C Product Qualification Categories

Category	Description	Qualification Requirement
A	COTS with no modification, and tested within product specifications	No qualification program required
B	COTS with no modifications, but not tested within product specifications	Partial qualification program required
C	COTS with modifications	Partial or full qualification program required
D	Custom components	Full qualification program required

Component selection has been carried out carefully to ensure that all COTS components fall into category **A**. Thus, the only qualification testing needs to be done on the custom portions of the project, i.e. components fitting into category **D**.

2.2. Verification Method Determination

An important factor to consider is what requirements need testing for verification, and which requirements can be verified using the other methodologies specified in ECSS-E-ST-10-02C (Inspection, Review-of-Design, Analysis) previously discussed in Section 1.2. This is the question posed by RQ3.1 in Table 6.1. Many of the requirements laid out by DLR are related to the presence of specific features, rather than the actual performance of those features. Thus, the selection of verification method (test or Review of Design (RoD)/inspection) will depend on the end goal of the requirement. If the requirement is that a feature be implemented or have a specific characteristic (e.g. the test setup will measure temperature) the verification method will be Review of Design, or if possible inspection (although most of these requirements are related to software implementations). If the requirement is related to an actual performance characteristic (e.g. the the minimum accuracy of temperature measurement is **x**), the verification method will be through testing.

2.3. Verification Test Plan Derivation

The methodology behind the verification test plans requires a solid theoretical basis upon which to build. This can be found in the previously discussed literature in Section 1.2. The industrial standards in particular yield significant insight into the recommended features. A common thread in all the presented industrial standards is the importance of traceability to known metrological standards. This provides known values for which to compare against, and proves that the test setup is calibrated properly. This traceability can take the form of adherence to known physical quantities (e.g. the inertia of a component based on its dimensions and mass) or with a calibrated instrument (e.g. a scale that has been properly calibrated at both a zero point and a known weight).

The design of each verification test plan is based on two major factors: what exactly needs to be verified, and what the metrological basis for the verification is. In this sense, there is a direct link between the verification plan and the project requirements. The general procedure that is used to create a verification plan is:

1. Identify requirement for verification
2. Determine metrologic basis
3. Determine what certified equipment or known physical quantities is needed
4. Ideate and select method to compare system to known values
5. Formalize plan

Each verification plan provides a summary of what, why, and how is being tested. The metrological reasoning behind the verification methodology is also discussed. The verification criteria is then given. An abbreviated list test items, setup steps, and operation steps are provided, along with the format the results will be in. The results of the verification and a discussion of the efficacy of the metrologic methodology are then presented. The sum of these verification plans and their results are then used to answer RQ3.2 posed in Table 6.1. The template for the verification plan is provided

below:

Table 2.2: Template Verification Test Plan

Goal Requirement ID
Summary
Reasoning
Verification Criteria
Test Items
• ...
Test Setup
1. ...
Operation Steps
1. ...
Results Format

2.3.1. Verification Test Plan Evaluation Criteria

There is no simple way to quantifiably evaluate the efficacy of these verification test plans, and the development of a quantitative evaluation method would be an undertaking in itself. Thus, a qualitative method of verification test plan efficacy evaluation is needed. In order to determine what criteria are important to consider, the industrial standards discussed in Section 1.2 are reviewed for commonly mentioned features or considerations for verification tests.

The first important criteria is not necessarily discussed explicitly in literature, but is nonetheless foundational to the rest of the verification: **"Does the test address a requirement?"** If the verification test has no association with a requirement, then there is no reason for the test to exist. This assumes that the requirements list has been thoroughly vetted and reviewed. For the model system being used, the requirements provided by DLR have been vetted internally already.

Metrologic reasoning and traceability is a commonality between the discussed industrial standards, which highlights its importance in verification testing. The Kavlico standard states that the testing equipment used needs to be calibrated in a manner that is traceable

to a recognized standards organization such as NIST (KAVLICO, 2020). One of the two validation/verification principles discussed by Eurolab is that reference materials of some form should be used to create a chain of measurement traceability. The ISO approved methods for validation/verification include calibration using reference standards and a comparison of test results with those from validated methods. From these standards, it can be seen that **"Is the metrologic reasoning sound?"** is of primary importance to the verification procedure chosen.

The second validation/verification principle discussed by Eurolab is the usage of scientific knowledge to describe the influencing factors (Eurolab, 2020), while ISO recommends a systematic assessment of influential factors and/or a variation of controlled parameters to test method robustness (ISO, 2017). These recommendations reflect the importance of isolating influential parameters in a test. At its core a test is an observation of cause and effect, and in order to understand why a particular effect has occurred the potential causes need to be limited. Thus, another important evaluation criteria for the verification test plans is **"Does the test properly isolate influencing variables?"**

The repeatability of the verification test represents a more practical concern. The results of a single iteration of a test cannot be taken as absolute, as there is the possibility of measurement or performance errors, or general outliers. Eurolab recommends generating an estimation of the repeatability of the verification test in order to prove that the verifying body is capable of achieving the required performance to carry it out. **"How repeatable is the test?"** as a criteria is important not necessarily because it increases the validity of the test itself, but rather that a repetition of the test increases the validity of the results.

Table 2.3: Evaluation Criteria

C1	Does the test address a requirement?
C2	Is the metrologic reasoning sound?
C3	Does the test properly isolate influencing variables?
C4	How repeatable is the test?

By considering these criteria for each test plan, an evaluation of the individual test plan and later the test plan format as a whole can be created. Although this evaluation is qualitative, it is still useful to assign a value to the criteria to allow for an easy comparison. For this purpose, a simple graphical scale is developed. An explanation of what each rating represents is provided below:

Table 2.4: Evaluation Criteria Scale

RED	Does not address criteria in any way
YELLOW	Somewhat addresses the criteria
GREEN	Thoroughly addresses the criteria

This graphical scale, along with the justifications for the assigned value, provide a quick evaluation while preventing the assumption of a quantitative evaluation.

While each of the criteria given in Table 2.3 are important, they do not necessarily have an equal weight. To determine the relative weight of each criteria, an analytic hierarchy process criteria weighing is performed.

This analysis is performed utilizing the methodology laid out by Cremades and Ponsich in *Simple and Objective Determination of Criteria Weights for Evaluating Alternatives when using the Analytic Hierarchy Process* (Cremades and Ponsich, 2025).

First, the relative importance of each criteria needs to be determined. This is done through a pair-wise comparison, where the relative importance of a criteria in relation to another is determined to be greater, equal, or less. The importance of the created evaluation criteria relative to each other is determined through a combination of the overall impact on the verification test plan, and how frequently it is discussed in literature. This creates a qualitative hierarchy of criteria. In order to facilitate an analytical method, a numerical value is then assigned based on this pair-wise comparison, where a lower number indicates a higher importance. Criteria are assigned the same value if they are of equal importance. From this pair-wise evaluation, the criteria are assigned the following importance values:

C1: 1
C2: 1
C3: 2
C4: 3

Performing the weighing calculations yields percentage values of:
C1: 43%
C2: 43%
C3: 10%
C4: 4%

The full calculation can be found in Appendix A. These resulting weights can be accounted for in the evaluation table used for each verification test plan by adjusting the width of each column to reflect the relative impact of the associated criteria. This provides an intuitive way to evaluate the overall rating of the verification plan, and can be used to directly compare verification plans to each other. The width of each column is directly correlated to the weight given to the criteria.

Table 2.5: Verification Test Plan Evaluation Template

C1	**C2**	**C3**	**C4**
GREEN	GREEN	GR	GR

3

Design

It is important to note that the mechanical design of the system was fixed before the start of the software/electrical hardware portion or the project, along with the required sensors. Data acquisition systems and sensor integration were not, however, and the design of the software and electrical hardware needed to integrate and acquire data from the selected sensors is covered in this chapter. A general overview of the system can be found in Figure 3.1 below:

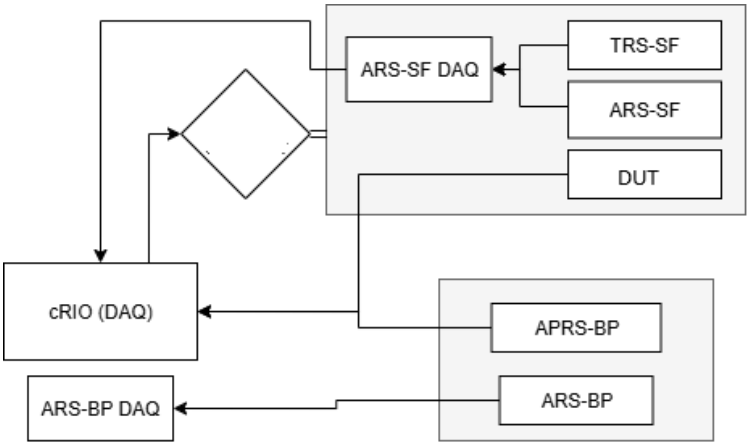

Figure 3.1: System Functional Diagram

The National Instruments' cRIO serves as the main Data Acquisition (DAQ) system, and also sends commands to the controller. Data from the APRS-BP is sent directly to the cRIO, while data from the ARS-BP is sent to a specialty DAQ. The reasons for this DAQ are explained later in Section 3.2. Data from the DUT is sent directly to the cRIO, while data from the ARS-SF and TRS-SF is acquired by a DAQ. This data is then forwarded to the cRIO. The necessity of this DAQ is also detailed in Section 3.2.

Data acquisition, along with some command and control algorithms, are achieved through the National Instruments cRIO. This device is designed for use in data

acquisition and control applications, and is capable of handling many types of sensors simultaneously. The cRIO consists of a Linux based Real-Time computer, linked Field-Programmable Gate Array (FPGA), and a modular backplane that allows for different IO modules to be inserted. For example, there are modules for analog voltage output, 5V digital IO, serial communication, etc. The linked FPGA and Real-Time computer allow for complex real-time processes to be completed at high clock rates in parallel.

A feature-based development structure was used to create this design. Each specific feature (e.g. temperature measurement, serial communication, etc) was developed as a discrete phase, and was then integrated into the project as a whole once working. This ensured that desired features were fully functioning within the overall system, rather than multiple simultaneous tasks being worked on at the same time. This also allowed for a high level of organization, preventing any accidental discarding of desired capabilities. Each development phase resulted in a combination of hardware and software solutions, which are discussed separately in this chapter. For example, the temperature measurement phase involved the selection of electrical hardware to be used (such as the actual sensors), its future electrical integration with the overall system, and the measurement software that is needed to acquire the data generated by the sensors.

3.1. Software

The cRIO is natively designed to utilize NI's LabVIEW coding language, and thus the vast majority of the software is coded in LabVIEW. LabVIEW's inherent characteristics make it ideal for data acquisition applications. It is simple to implement parallelization, and has many built in features and utilities for data acquisition. Unlike most programming languages it is constructed graphically, with blocks connecting to each other through wires similarly to how circuit diagrams are laid out. Debugging is also similar to circuit debugging, with probes being attached to wires and the results observed during operations. This can make programming and debugging intuitive if one is experienced in electrical hardware solutions, and the process of switching between working on hardware and working on software is smooth. LabVIEW code is compiled into self-contained "Virtual Instruments" (VI), which are highly modular. Even if it was not the natively supported software for the selected hardware system, LabVIEW would be an ideal language to use for this application.

Previous research conducted at DLR has resulted in the development of a software library for LabVIEW that allows for easy integration with various digital and analog sensors, along with synchronzation and data saving utilities. This software is based on the DAQmx API, and is designed such that the data from multiple sensors can be saved to the same TDMS file while retaining determinism. This library will be utilized to acquire data from the various sensors in use.

The implementation of this library is reliant on a LabVIEW architecture known as "Producer-Consumer" architecture. In order to display data at near real-time rates, parallel loops of data acquisition and display programs can be used, with data passing between the two using a real-time First-In First-Out (FIFO). This allows for data to be

passed between two loops running at different speeds, while maintaining the correct order of data in time. This architecture, where one loop is consuming data at a different rate from the loop producing data, is known as Producer-Consumer. This structure is essential in fulfilling the GUI display requirements laid out in Section **??**. When a GUI display of the incoming data is not required, the library functions are usable in such a manner that data is automatically saved without requiring a Producer-Consumer structure. This also allows for faster acquisition rates, as the data is written directly to flash storage rather than having to be parsed by additional computational layers.

LabVIEW is capable of packaging code into a "subVI" that is executable inside of other VIs, similar to writing a new function in MATLAB. Usage of subVIs greatly increases the portability and reusability of code, and the structure of the overall program is based around their usage. The previously discussed data acquisition library is implemented through these subVIs, along with other communication protocols that were developed for this project but with portability in mind (e.g. serial communication and processing library). The main VI contains two parallel structures. One is a loop that handles front panel user interactions, e.g. starting, stopping, data viewing for testing, etc. The second is a sequence that contains all of the data acquisition subVIs, and then the subVIs for processing the recorded data into a desired format. Most of the data acquisition and synchronization processes are contained within a single subVI, for easier organization. The general model diagram for the software is shown in Figure 3.2:

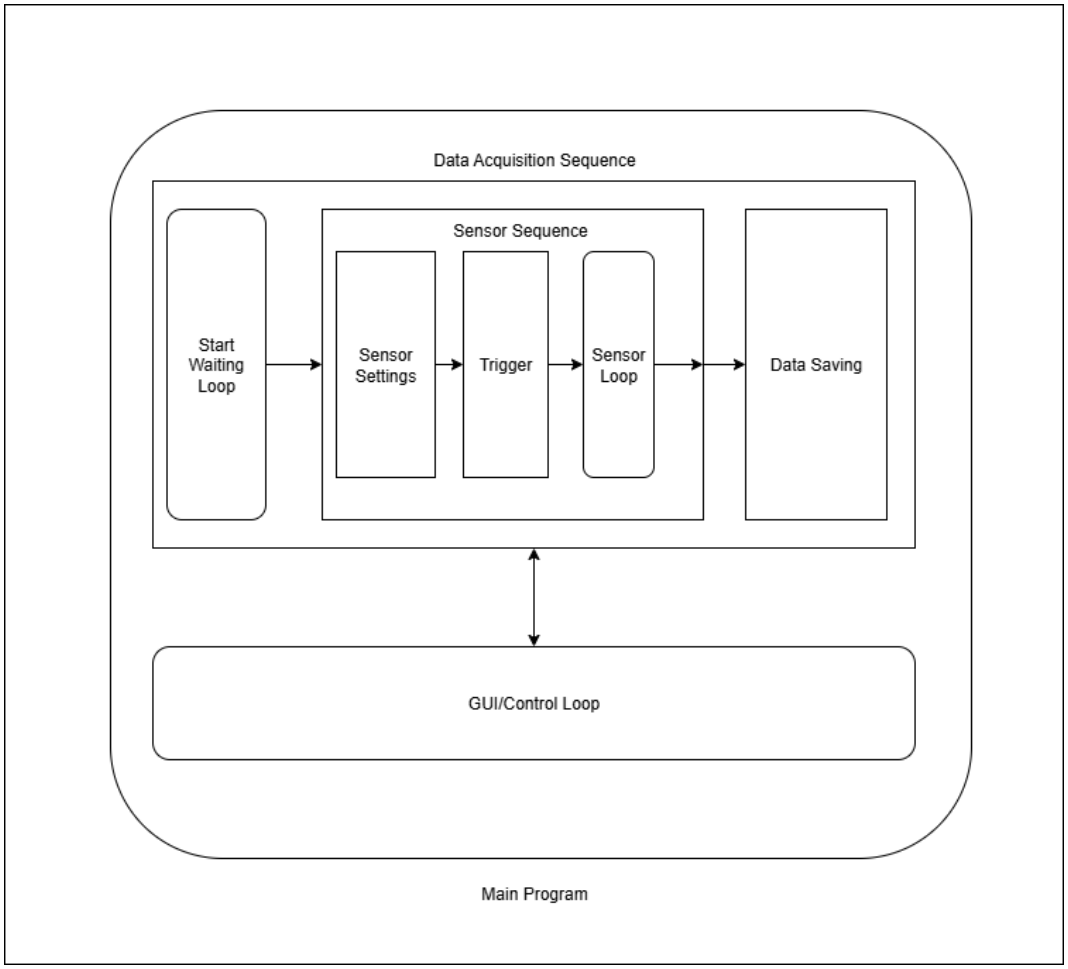

Figure 3.2: Model Diagram of Software

The software is divided into these two parallel structures so that the GUI does not have to run at the same frequency as the sensor sequence. This reduces the computational load on the cRIO. The GUI/control loop contains the initial settings that are passed on to the sensor settings, and displays data when the correct settings are chosen. Start and stop control are also handled in this loop, and passed asynchronously to other operations as necessary through two FIFOs. In the data acquisition sequence, the start loop is essentially an empty loop that exists to allow for settings to be adjusted after the program run is started. Once the start command is passed, the loop is ended and the next step in the sequence can start. The sensor handling is divided into a three step sequence. The first step handles the sensor settings, such as active port, rates, data type, etc. The start trigger is then sent, which is used to synchronize the internal and external data acquisition processes. The sensor loop starts immediately after the trigger is sent, and runs until a stop command is sent from the GUI loop. Once the sensor loop is stopped, the data collected is sent to a single file for later processing.

Serial communication in LabVIEW can be achieved through several different methods. The first is NI Virtual Instrument Software Architecture (VISA), which is an built-in software library that allows for the easy construction of a serial program using a handful of subVIs. Because of how this library in constructed, however, it is not able

to interface with devices that have a baud rate higher than 115200 (for the selected serial cRIO module). Higher baud rates can be reached by programming directly onto the FPGA, rather than using a pre-made interface. By using the FPGA serial commands, and then passing the data from the FPGA to the RT computer through a FIFO, serial communication at baud rates higher than 115200 is achievable. This is more complicated to program than the VISA library, but the end result is more capable and performant.

Synchronization of internal and external data acquisition is achieved through a hardware trigger. The internal data acquisition routines are tied to this hardware trigger, and the processes start when this trigger is sent out. This allows for the start time of data coming from sensors that need to be triggered and sensors that generate data at a fixed frequency to be shared, providing synchronization.

3.2. Electrical Hardware

Much of the specific electrical devices (e.g. data acquisition device, sensors, etc) were pre-selected by DLR before project start. The integration of these components into a functioning data acquisition system is a primary concern. These devices consist of:

- cRIO, a DAQ and control device
- cRIO Modules:
  - 5V DIO
  - Low Voltage (3.3V) DIO
  - Serial Interface
  - Voltage Output
- APRS-BP
- ARS-SF
- ARS-BP
- TRS-SF
- Controller

4

Design Execution

4.1. Electrical Hardware

The final electrical hardware resulting from the execution of the design presented in Chapter 3 mostly consists of the connections between the pre-selected sensors and the cRIO. The connection between the individual sensors and the specific cRIO I/O modules is dependent on the output type of the sensor. Each of the selected sensors has its own output format, such as scaled current or serial packets, and each cRIO module is only able to handle one type of input. The resulting wiring harness utilizes standard hardware connectors (e.g. BNC, DSUB, IDE, etc) in order to allow for quick assembly and disassembly.

A single Power Supply Unit (PSU) is used to provide power to the various components present. The cRIO is the main DAQ unit for the setup, and utilizes various input/output modules in order to perform the various operations as needed. The 10V analog output module is used to send a control signal to the Controller. This module also sends out a repeated trigger used to trigger the DUT to record data. Output from the APRS-BP is sent into the 5V DIO, while a continuous start trigger is sent from the 3.3V DIO to the ARS-BP and ARS-SF DAQs. The ARS-BP DAQ saves data internally, while the ARS-SF DAQ sends the recorded data back to the cRIO through a serial interface.

The power requirements also need a relatively complex solution, as there are numerous voltages being used by the setup. Both the cRIO and its serial module require a 26V input, the DUT requires 15V, and the ARS-SF, TRS-SF, and ARS-SF DAQ require 5V. The controller is operating on an entirely independent power system, as it requires a 48V high-current input. The controller has a high current draw, as the controlled device itself is powered through the controller, but the rest of the system has a net current draw on the order of 3-4 Amps (2A on the 26V line and approximately 1 Amp on the 15V/5V line). A 48V 16A power supply is needed for the controller, accounting for the draw of both the and the controller. Simple desktop power supply units (PSU) are used to provide power to the various system components.

4.2. Software

The implemented software follows the same structure as laid out in Chapter 3. This implementation can be divided into three main sections for discussion: the serial communication section, the sensor acquisition section, and the main overarching program. The sensor acquisition section is based on an implementation of the previously developed sensor library.

4.2.1. Serial Communication

As discussed in Chapter 3, the serial communication portion of the program utilizes the cRIO's FPGA. The software itself consists of four subVIs. The first is the FPGA VI, which interfaces directly with the serial port hardware. Then there are three VIs that run on the Real-Time system, in order to handle the interface between the FPGA VI and the rest of the Real-Time system. The first RT VI handles the configuration of the FPGA VI (e.g. baud rate, stop bits, number of bytes to read, etc). The second VI handles the reading of bytes from the FPGA to the Real-Time system, through a FIFO. The third VI handles the writing of bytes from the Real-Time system to the FPGA, through a different FIFO. This third VI is not used in the current implementation, but is included in order to allow for future use of these programs in different applications. These serial subVIs are integrated into a single serial communication VI that handles the communication settings, and passes the data from the serial port to an internal processing system to sort the packet data and convert it to engineering data.

4.2.2. Sensor Library Implementation

As the sensor library was designed with synchronous data acquisition from a diverse array of sensors in mind, its application here is straightforward. Each sensor type has two VIs associated with it; the first is configuration, and the second is the acquisition. The configuration VI handles the sensor settings, along with data logging and trigger sources. The acquisition VI contains the loop that acquires and logs data. It also can pass data to the front panel, if the correct configuration settings are applied. The sensor VIs are all bundled into a single sensor subVI that is ran in the main program, and consists of a sequence structure that runs all of the configuration VIs, and then all of the acquisition VIs. The acquisition VIs are triggered internally.

4.2.3. Main Program

The main program in which the other subVIs are hosted in has the same structure discussed in Chapter 3. Global start/stop variables are asserted as false before either the GUI loop or the operation sequence begin. The GUI loop runs continuously after the program is run, and passes the control values (Start and Stop) to the appropriate global variables. These real-time global variables are used to ensure that all operations within the program start and stop at the same time. The first step in the operation sequence is the Start loop, which essentially holds the program until the start button is pressed. This prevents the program from starting prematurely, along with giving the option to double check inputted settings before running. After the start loop is ended, the general settings are fed from the front panel into the respective sensor subVI (e.g. DUT trigger frequency, etc). The sensor subVIs are then ran until the stop button is

pressed. The loops for these VIs are internal, hence the necessity of a real-time global variable for stopping. After the sensor subVIs finish, the data is handed to a data processing subVI. Errors are then handled using LabVIEWs built-in error management VI.

5

Verification Results

Following the methodology laid out in Chapter 2, first the major points of verification were identified. These are the components or functions whose requirements compliance needed to be verified, and are derived from the phase-based requirements organization presented in Section REMOVED. The full list of verification subjects is provided below:

1. APRS-BP
2. Serial
    (a) Serial Communication
    (b) Serial Data Processing
    (c) Serial Integration
3. DUT Trigger
4. TRS-SF
5. ARS-SF
6. Control
    (a) Test profiles
    (b) Position Control
    (c) Velocity Control

Once this list was developed, a general approach for verification was selected. Rather than constructing the entire system and then proceeding with verification, the process was broken up into subsystems closely related to the verification subject list. These subsystems were then built, underwent verification, and were then integrated into the final setup.

It is important to note the two subsystems that have additional feature breakdowns. The complexity of the serial program resulted in its division into three different phases, which allows for a faster and more stable development due to the isolation of potential

error causing factors. The control subsystem was divided into three phases for similar reasons, along with hardware availability restricting specific phases in the development. Due to issues with the development timeline, the Position and Velocity control systems were not finished, and were thus unable to be verified.

5.1. Verification Plans

5.1.1. APRS-BP Response Time

Table 5.1: APRS-BP Response Time Verification Test Plan

Goal Requirement ID
1.8
Summary
The APRS-BP works by using functions from the sensor software library to count incoming digital edges. When a digital edge is detected, the counter increases, and a timestamp is recorded. It is necessary to determine the accuracy of this timestamp. In order to do so, the timestamps recorded by the cRIO test setup must be compared to the timestamps recorded by a known system, in this case an oscilloscope, both utilizing the same sensor.
Reasoning
In order to verify that the cRIO is able to accurately record the time at which the APRS-BP is triggered, a comparison with a known system is used, in this case an oscilloscope. This provides a form of metrological traceability to properly certify requirement compliance.
Verification Criteria
Percent difference between values has an average of 1% and standard deviation of 0.5%
Test Items
• APRS-BP • cRIO with TTL digital IO module • Calibrated Oscilloscope • Control laptop • Trigger mechanism
Test Setup
1. connect the cRIO to an appropriate power supply 2. connect the cRIO digital IO module to the output leads of the APRS-BP 3. connect the control laptop to the cRIO 4. connect the oscilloscope to the output leads of the APRS-BP
Operation Steps

1. power on the oscilloscope
2. power on the cRIO
3. start the APRS-BP LabVIEW program using the control laptop
4. Trigger the APRS-BP
5. once the maximum amount of readings on the oscilloscope have been captured, stop recording

Results Format

The end results of this verification test are the timestamps for each sample. The timestamps from the cRIO and the Oscilloscope are different formats, as the cRIO gives absolute timing while the Oscilloscope gives relative timing between each sample, but what is of interest is the difference in timing between the two sources rather than the absolute timing. Thus, the difference between successive timestamps from each source is calculated, and then the percent difference between the two sources is calculated. These results then show the deviation of the designed system from a known reference, allowing for performance verification.

For **C1** ("Does the test address a requirement"), this verification test only somewhat addresses the requirement. The goal requirement (1.8) simply states that the APRS-BP will be present and what data will be measured. This verification test itself does verify the presence and operation of these sensors, but additionally determines the actual response time. This information is useful, but not specified in the requirements. With regards to **C2** ("Is the metrologic reasoning sound"), a line of metrologic traceability to a piece of certified equipment is established through the usage of a calibrated oscilloscope. Criteria **C3** ("Does the test properly isolate influencing variables") is only partially addressed, as the trigger mechanism used is not isolated. Variance in the trigger may affect the response of the APRS-BP, and it would be better to use a more rigid element for triggering. The repeatability of this particular test is high as it is simple to set up and carry out, showing that **C4** ("How repeatable is the test") has been thoroughly addressed. The resulting evaluation chart is given in Table 5.2 below:

Table 5.2: APRS-BP Response Time Verification Test Plan Evaluation

C1	**C2**	**C3**	**C4**
YELLOW	GREEN	YL	GR

5.1.2. Serial Data Processing

Table 5.3: Serial Data Processing Verification Test Plan

Goal Requirement ID
1.6
Summary
The form these data packets take are unique to each device, and thus the translation from raw bytes to usable engineering data needs to be verified. Previously verified translation programs exist, but they are not compatible with the current system. This does, however, allow for the same set of packets to be passed through both the verified and unverified systems. The differences in final values can then be analyzed. A control VI is written such that the raw data can be read from a file into the processing program, one byte at a time. This allows for both the packet alignment method and the translation method to be tested simultaneously.
Reasoning
The metrological basis with this verification test is a comparison to another certified standard. The verified processing program and the program that is being used with this test setup should provide the same output when given the same data packet, and the results from the unverified program are used to determine where errors in the program occur.
Verification Criteria
Provided pre-processed data and data processed by the setup are the same values
Test Items
• cRIO • USB drive with raw data packet(s) • Results from data processing with verified program • Control laptop • Data processing program saved on laptop
Test Setup
1. Connect the cRIO to an appropriate power supply 2. Plug the USB drive into the USB port on the top of the cRIO 3. Attach the control laptop to the cRIO through an ethernet cable
Operation Steps

1. Power on the cRIO 2. Connect to the cRIO in the appropriate LabVIEW project 3. Ensure that the correct file path for the desired test packet is selected 4. Start the data processing test program 5. The program will stop automatically once the entire data packet has been read, and the data is saved 6. Compare the resulting data with the known values
Results Format
The data outputted contains different data, so the results for each test will have a different format. Ultimately, the values from the unverified and verified program will be directly comparable, and the differences can be analyzed. The data from the test is saved as a TDMS file, which is readable as an excel file.

This test had to be performed for each program multiple times, as errors were uncovered and the programs were refined. The programs were dealt with sequentially, meaning that the errors with one program were fully resolved before moving on to the next one. After the program was reliably providing the same results as the verified program, additional data packets were used to test error rejection and performance with different data sets. These uncovered some additional issues with error rejection processing, which were corrected. Through vigorous and repeated application of this verification test, the issues with the processing programs were resolved, and they were verified to consistently produce the correct data, while rejecting incorrect raw packets.

A requirement is somewhat addressed in the presented test plan. This verification test is used to verify the functionality of the entire serial data processing scheme, from raw serial packets to engineering data, while the goal requirement just covers the presence of data validity checks for data. Metrologically, this verification test plan provides a clear link to a certified standard, in that the processed results can be directly compared to the results from a certified processing program. Influencing variables have been isolated such that the data processing mechanism is the only potential influence on the outcome. This test is repeatable, and additional pre-processed data can be generated for comparison easily by running the certified program.

Table 5.4: Serial Data Processing Verification Test Plan Evaluation

C1	**C2**	**C3**	**C4**
YELLOW	GREEN	GR	GR

5.1.3. Serial Communication

Table 5.5: Serial Communication Verification Test Plan

Goal Requirement ID
1.3, 1.4
Summary
The software used to handle serial communication is a modification of Commercial-Off-The-Shelf (COTS) software. These modifications were necessary to ensure synchronization and proper handling of specific communication requirements. In order to properly verify this program, a two phase verification test is used. The first phase uses a USB to Serial converter to transmit known data in the same format, and the second phase uses the actual hardware.
Reasoning
In order to properly verify that the serial communication software is functioning properly, a comparison with a known/verified device is needed. Ultimately, there are several factors that are required to mimic the output of a serial device. The first are the serial communication settings (baud rate, stop bits, parity, etc), the second is the data format (packet size, form of transmitted data, etc), and the third is the transmission frequency (the rate at which packets are sent). These first two factors are very easy to mimic using a basic USB to Serial converter, along with the raw data packets from the Serial Data Processing Verification Test. The transmission settings will be the same, and the data will be the same type as transmitted. The transmission timing is more difficult to simulate. Thus, testing with the actual hardware is required as well. Separating the verification into two phases also allows for the isolation of issues and potential causes.
Verification Criteria
1. Phase 1: File saved by the setup contains information identical to that in the file transmitted 2. Phase 2: No processing errors or flags are tripped during testing
Test Items
• cRIO with serial module • Control laptop with PuTTY installed and raw data packet(s) saved • USB to serial converter • Hardware setup
Test Setup (**Phase 1: USB to Serial Converter**)

1. Connect cRIO to an appropriate power supply
2. Connect the USB to Serial converter to the control laptop
3. Power on the cRIO power supply
4. Connect the USB to Serial converter to the cRIO serial module port
5. Connect the control laptop to the cRIO

Operation Steps (**Phase 1: USB to Serial Converter**)

1. On the control laptop, open a command prompt
2. Start the serial communication protocol on the cRIO
3. In the command prompt, execute the command to send a file's content over serial, ensuring the correct file and serial session are selected
4. Once transmission stops, stop the serial program on the cRIO
5. Exit the PuTTY session

Results Format (**Phase 1: USB to Serial Converter**)

As the serial communication program is designed to save the results automatically in the format in which they were transmitted, a file containing the 8 bit integer version of the transmitted data is saved. This file can be compared to the 8 bit version of the transmitted file, and analyzed for errors.

Test Setup (**Phase 2: Hardware**)

This assumes that the hardware has already been setup for serial communication (i.e. power and serial communication harness have already been attached)

1. Connect cRIO to an appropriate power supply
2. Connect the hardware to the appropriate trigger and serial ports on the cRIO
3. Connect the control laptop to the cRIO

Operation Steps (**Phase 2: Hardware**)

1. Power on the cRIO
2. Power on the hardware
3. Start the serial communication program
4. Wait until a good amount of packets have been sent (10-20) and then stop the program

Results Format (**Phase 2: Hardware**)

The results here are saved automatically as well. As the transmitted data is unknown, the content of the transmitted data cannot be checked, but the function of the program with the actual hardware can be checked with LabVIEW's error handling mechanisms. If there is an issue in the actual communication, there will be a force stop and an error message. The previous phase of testing allows us to narrow down the error source to specifics in the transmission timing, rather than the communication settings or packet contents.

The second phase of this verification has less of a metrological basis, as it is essentially supplementary error finding after comparison with known values. This test was highly repeatable by design, as it was assumed there would be programming errors initially that would need to be resolved. The results from these tests were useful in finding the specific error sources, and ensured that this system was working as intended.

This verification test can be directly linked to the specified goal requirement, in that this test addresses that the hardware should be triggered in order to generate measurements, and that the test setup is sampling the resulting measurements from the hardware. Two different known devices are provided, each covering the area of metrologic reasoning that is left out by the other. The use of multiple transmission devices does result in a larger amount of influencing variables, which is not desirable. Error tracing with this particular verification test will be difficult due to the number of influencing factors. The repeatability of this verification test is very high, with new raw data packets being easily generated as needed.

Table 5.6: Serial Communication Verification Test Plan Evaluation

C1	**C2**	**C3**	**C4**
GREEN	GREEN	YL	GR

5.1.4. Serial Integration

Table 5.7: Serial Integration Verification Test Plan

Goal Requirement ID
1.4, 1.5, 1.6
Summary
After the communication and processing portions of the serial program have been tested, they are integrated together. This allows for communication and the outputting of human-readable engineering data, which is directly saved. While each portion have been individually verified, the integration of these two sections into a functional program needs to be verified as well. The overall test is similar to that used for serial communication (Table 5.5), except that the final data should be the correct engineering values rather than 8 bit. In addition, sample timestamps are applied to the saved data. The spacing of these time stamps can be compared with the desired trigger frequency in order to verify trigger spacing.
Reasoning
The metrologic basis for this test is much the same as that in the serial communication verification test, i.e. the partial comparison with a known device. The expected results with a serial converter are known, and the comparison of inputs and outputs can be used for verification and error checking. An additional phase of testing with the actual hardware is required to ensure full verification, as the serial converter is not able to fully emulate the hardware. The second phase exists more for error checking, but combined with the previous verification of known input/output, allows for a full verification of the integrated serial program. This second phase also allows for the verification of trigger spacing/timing, as the timestamps of the saved data should match the set trigger frequency.
Verification Criteria
1. Phase 1: Data processed by the setup are equal to the values transmitted 2. Phase 2: No errors occur during run time, and the saved data matches expected format
Test Items
• cRIO with serial module • Control laptop with PuTTY installed and raw data packet(s) saved • USB to serial converter • Hardware setup
Test Setup (**Phase 1: USB to Serial Converter**)

1. Connect cRIO to an appropriate power supply
2. Connect the USB to Serial converter to the control laptop
3. Power on the cRIO power supply
4. Connect the USB to Serial converter to the cRIO serial module port
5. Connect the control laptop to the cRIO

Operation Steps (**Phase 1: USB to Serial Converter**)

1. On the control laptop, open a command prompt
2. Start the serial communication protocol on the cRIO
3. In the command prompt, execute the command to send a file's content over serial, ensuring the correct file and serial session are selected
4. Once transmission stops, stop the serial program on the cRIO
5. Exit the PuTTY session

Results Format (**Phase 1: USB to Serial Converter**)

The data received by the cRIO is automatically saved to a TDMS file. These engineering values can then be directly compared to the engineering values transmitted.

Test Setup (**Phase 2: Hardware**)

This assumes that the hardware has already been setup for serial communication (i.e. power and serial communication harness have already been attached)

1. Connect cRIO to an appropriate power supply
2. Connect the hardware to the appropriate trigger and serial ports on the cRIO
3. Connect the control laptop to the cRIO

Operation Steps (**Phase 2: Hardware**)

1. Power on the cRIO
2. Power on the hardware
3. Start the serial communication program
4. Wait until a good amount of packets have been sent (10-20) and then stop the program

Results Format (**Phase 2: Hardware**)

Although the transmitted data is not able to be confirmed, the saved data will be engineering values. If there is an error in the communication or sorting of the packets, LabVIEW error handling should force stop and present a notification. If there is not an error, but the saved values are scrambled or are not in the expected range, the process should be further examined undetected errors. The timestamps included with this data will be used for trigger timing verification.

This test is extremely similar to the serial communication verification test (Table 5.5), and thus has the same merits and detriments.

Table 5.8: Serial Integration Verification Test Plan Evaluation

C1	**C2**	**C3**	**C4**
GREEN	GREEN	YL	GR

5.1.5. DUT Trigger

Table 5.9: DUT Trigger Verification Test Plan

Goal Requirement ID
3.3, 3.4, 3.5
Summary
The DUT requires an external trigger. This trigger signal is incredibly important in synchronizing the DUT to the rest of the system, and thus the accuracy and the stability of the trigger signal needs to be verified. This trigger signal is also tied to the Start Trigger, and the difference in timing between the two signals needs to be analyzed. This means that the testing will have two phases: one to verify the timing of the external trigger, and the other to verify synchronization with the Start Trigger.
Reasoning
Here the oscilloscope is a known metrologic device. This allows for precise measurement of the actual characteristics of the outputted trigger, which can then be compared to the desired characteristics.
Verification Criteria
1. Phase 1: Setup is able to generate a trigger at each of the frequencies with a standard deviation of 5% 2. Phase 2: Skew between DUT and Start triggers has a standard deviation below 1% of the expected frequency
Test Items
• cRIO with analog IO module and LV TTL digital IO module • Certified Oscilloscope • Control laptop
Test Setup
1. Connect the cRIO to an appropriate power supply 2. Connect the control laptop to the cRIO 3. Connect the analog output to channel 1 of the oscilloscope 4. Connect the digital output to channel 2 of the oscilloscope
Operation Steps **Phase 1: Timing**

1. Power on the oscilloscope
2. Power on the cRIO
3. Open the DUT Trigger program on the control laptop
4. Set the frequency of the trigger
5. Start the trigger, and record the results from the oscilloscope for 10000 cycles (25 seconds)
6. Stop the trigger, and ensure the results are saved
7. Start the trigger, and observe the frequency displayed. Note the max, min, and average over 25 seconds
8. Repeat the previous step for each desired frequency

Operation Steps **Phase 2: Synchronization**

1. Power on the oscilloscope
2. Power on the cRIO
3. Open the DUT Trigger program on the control laptop
4. Set the frequency of the trigger
5. Start the synchronized program
6. Record the start time on both channels

Results Format

The results from this test will take several forms. The first is the waveform data for a specific frequency trigger. This data will be used to verify the cycle-to-cycle jitter of the trigger signal. The second is max, min, and average frequency values for the desired range. This is used to show that the system is capable of reaching this range of frequencies, and how accurate it is inside this range. The last is the comparative start times of the two trigger sources, which allows for verification of synchronization.

The results from this verification test can be directly applied to the specified goal requirements, providing a thorough link to the requirements. By using a certified and calibrated oscilloscope, a chain of metrologic traceability is created to the certifying institute and the relevant standards. The only potential influencing variable is limited to the cRIO system itself, which is the point of interest for this verification test. This test is also extremely repeatable, as it just requires the program on the cRIO to be run again.

Table 5.10: DUT Trigger Verification Test Plan Evaluation

C1	**C2**	**C3**	**C4**
GREEN	GREEN	GR	GR

5.1.6. ARS-SF

Table 5.11: ARS-SF Verification Test Plan

Goal Requirement ID
1.1, 3.1
Summary
The ARS-SF sensors are sampled by a DAQ, which converts the analog output into a digital form, and then transmits it to the cRIO. The ARS-SF sensors themselves output a continuous analog voltage signal, and thus the duties of sample timing and data transmission fall to the DAQ. This DAQ is to sample the output at a fixed frequency, although the amplitude and frequency will fluctuate greatly. The ability to sample the ARS-SF at the correct frequency simultaneously thus needs to be verified. In order to ensure that a proper reference value is available, the ARS-SF sensors themselves are substituted with the sinusoidal output of a function generator, with a fixed amplitude. This signal is duplicated up to the number of desired ARS-SF sensors.
Reasoning
The signal from the function generator can closely mimic the output of the ARS-SF sensors, while providing a consistent value with which to compare against. The function generator represents a certified piece of hardware, and thus a valid reference value.
Verification Criteria
All ARS-SF simulated inputs are able to be read simultaneously, at the desired sample rate. The recorded signal values are able to be fitted to a sine of the specified transmission frequency with a gain error less than or equal to 1% and a phase error of less than or equal to 2° with statistical confidence of 3σ. Less than or equal to 5% of packets have been dropped during transmission.
Test Items
• Function Generator • ARS-SF DAQ • cRIO • Control laptop
Test Setup

1. Connect the cRIO to an appropriate power supply 2. Connect the control laptop to the cRIO 3. Connect the function generator to the ARS-SF DAQ, duplicating to input to the requested number of channels 4. Connect the ARS-SF DAQ to the cRIO 5. Power on the function generator 6. Set the waveform type to sin wave 7. Set the offset to 1V, and the amplitude to 2V
Operation Steps
1. On the function generator, set the frequency to (F) 2. On the control laptop, start the LabVIEW program for ARS-SF data logging 3. Start the function generator output 4. Record for 10 seconds 5. Stop the output, then stop the recording on the cRIO 6. Ensure that the data has been saved properly 7. Repeat at frequencies from (10F, 50F, 100F, 150F)
Results Format
The results of this test are the voltage measurements from each "sensor" at each experiment, along with the packet count. Sample timing is assumed to be fixed, and the resulting data can be fitted to a sine wave. The resulting sine wave can then be compared to the transmitted one. The packet count is a 16 bit unsigned integer, and can be used to determine the number of dropped packets during communication.

Here, the frequencies are normalized with respect to the lowest tested frequency. Data is fitted to a sine function using a least-squares method. The full derivation can be found in Appendix B. Provided frequencies values are normalized with respect to the starting frequency used in testing.

The average errors and standard deviations for each channel are presented in the tables below:

Table 5.12: ARS-SF Verification Errors Channel 1

Freq	CH1 Amp Error [%]	CH1 Phase Error [°]	CH1 Gain Error [%]
F	6.25	-0.34	6.51
10F	5.90	-2.56	6.07
50F	1.17	-16.14	6.66
100F	12.34	-27.87	6.63
150F	36.00	-40.11	6.40

Table 5.13: ARS-SF Verification Errors Channel 2

Freq	CH2 Amp Error [%]	CH2 Phase Error [°]	CH2 Gain Error [%]
F	7.16	-0.28	7.14
10F	7.08	-2.54	7.09
50F	2.36	-14.99	7.32
100F	11.34	-27.64	7.16
150F	35.82	-40.49	7.26

Table 5.14: ARS-SF Verification Errors Channel 3

Freq	CH3 Amp Error [%]	CH3 Phase Error [°]	CH3 Gain Error [%]
F	6.70	-0.34	6.78
10F	6.52	-2.53	6.73
50F	1.20	-15.96	6.58
100F	12.20	-27.49	6.59
150F	35.82	-39.52	6.53

Table 5.15: ARS-SF Verification Errors Channel 4

Freq	CH4 Amp Error [%]	CH4 Phase Error [°]	CH4 Gain Error [%]
F	6.77	-0.28	6.78
10F	6.69	-2.51	6.85
50F	1.97	-14.82	6.95
100F	11.79	-27.28	6.88
150F	36.25	-39.91	6.88

Table 5.16: ARS-SF Verification Standard Deviations

Channel	Amplitude Sdev [%]	Phase Sdev [°]	Gain Sdev [%]
CH1	12.35	25.26	0.21
CH2	11.88	21.66	0.08
CH3	12.17	24.92	0.10
CH4	12.18	21.35	0.05

In all of the conducted tests, there were no dropped packets.

This verification test plan loosely addresses the goal requirements, which are concerned with the ability to read the requested number of sensors and the minimum sample frequency, rather than the ability of the ARS-SF DAQ to properly record different frequency inputs. This verification test does provide verification for the goal requirements, but performs additional verification for capabilities not given by the requirements. By using a function generator, both a line of metrologic traceability is established, and the influencing variables are isolated. This test does not consider the input of the actual ARS-SF sensor, however, which may introduce unknown behavior. This test is highly repeatable, as it just requires the data acquisition program to be restarted on the cRIO.

Table 5.17: ARS-SF Verification Test Plan Evaluation

C1	**C2**	**C3**	**C4**
YELLOW	GREEN	YL	GR

5.1.7. Test Profiles

Table 5.18: Test Profile Verification Test Plan

Goal Requirement ID
1.15, 1.16, 2.12.13, 2.12.14
Summary
In order to facilitate testing, a pre-made profile for the position or speed is desired. These test profiles are made available as a CSV file, with the first column being the position or velocity, and the second the time to hold that value. The output control voltage is linearly scaled to the input value. Thus, there are two properties that need to be verified: output voltage accuracy and timing accuracy. By recording the output with a certified oscilloscope, the desired values can be easily compared with the actual output for accuracy in timing and in voltage.
Reasoning
Since the input CSV file directly correlates to the desired output, a performance comparison is easy to carry out in a metrologically valid method, especially through the use of a certified oscilloscope.
Verification Criteria
Average timing is within 1% of prescribed dt and average voltage is within 1% of desired value, with a confidence of 3σ.
Test Items
• cRIO with analog voltage output module • Control laptop • Oscilloscope
Test Setup
1. Attach the cRIO to an appropriate power supply 2. Connect the control laptop to the cRIO 3. Connect the analog voltage output from the cRIO to the input of the oscilloscope 4. Set the oscilloscope to record when triggered 5. Set the desired test profile on the control laptop
Operation Steps
1. Start the program on the control laptop 2. Once the sequence has finished, ensure that the oscilloscope has saved the data
Results Format

The results of the test will be a spreadsheet containing the transmitted voltages and their relative timing. Since the input values are directly scaled to determine the voltage the expected and actual results can be easily compared.

After the data is collected, a comparison is made between the expected and actual voltage, and the period for which the voltage lasts. Each voltage sample has its error calculated, and then the total error is averaged over the samples. Additionally, a standard deviation of the errors is calculated. The time error is calculated by comparing the time period the voltage was supposed to last, and the actual period it lasted. Average time error and time error standard deviation are calculated in the same manner as voltage error.

Table 5.19: Test Profile Error and Standard Deviation

Average Voltage Error [%]	Voltage Error Sdev [%]	Average Time Error [%]	Time Error Sdev [%]
2.42	1.33	0	0

For the measured sample time, there was no variance in expected vs actual timing, with the sample timing being 50 times that of the control timing.

This test addresses several specific requirements regarding the ability to use pre-made test profiles, and thus can be stated to thoroughly address this criteria. Metrologic traceability is provided through a combination of known values and certified equipment. By testing the test profile system separately from the actual control interface, the number of influencing variables is significantly reduced. This test is also extremely repeatable, as new test profiles can be quickly made and then results compared.

Table 5.20: Test Profiles Verification Test Plan Evaluation

C1	**C2**	**C3**	**C4**
GREEN	GREEN	GR	GR

5.1.8. Complete Verification Test Plan Criteria Evaluation

Table 5.21: Verification Test Plan Criteria Evaluation

Test Table Number	**C1**	**C2**	**C3**	**C4**
Table 5.1	YELLOW	GREEN	YL	GR
Table 5.3	YELLOW	GREEN	GR	GR
Table 5.5	GREEN	GREEN	YL	GR
Table 5.7	GREEN	GREEN	YL	GR
Table 5.9	GREEN	GREEN	GR	GR
Table 5.11	YELLOW	GREEN	YL	GR
Table 5.18	GREEN	GREEN	GR	GR

6

Conclusion

From the design, implementation, and verification of the necessary software and electrical hardware for DLR's desired test setup, the research questions proposed in Section 1.3 can be answered.

6.1. Goals

The overall purpose of this project can be referred to as the project goals. These goals can be divided into three distinct portions, which have been previously discussed:

- DLR Setup Objectives (Section 1.1, what the overall purpose and capabilities the setup needs to have)
- DLR Requirements (Section REMOVED, the specific requirements derived by DLR in order to achieve the Setup Objectives)
- Verification Literature Gap (Section 1.2, the gap in literature regarding the verification of test setups)

These goals resulted in the research questions provided in Section 1.3. Here, they are repeated for convenience:

Table 6.1: Research Questions (Repeated)

RQ1	What software design solution can be used to fulfill DLR's objectives and requirements for this test setup?
RQ2	What electrical hardware design solution can be used to fulfill DLR's objectives and requirements for this test setup?
RQ3	How can the derivation of specific verification methodologies for test setups be systematized?
RQ3.1	What method should be used for selecting the verification approach?
RQ3.2	What standard process should be used for creating verification test plans?

6.2. Software

The design solution resulting from RQ1 is shown in Section 3.1, and the actual implementation of this design in Section 4.2. The software for this project is written using LabVIEW, which was chosen for its hardware compatibility and data acquisition capabilities. The software has several tasks: sensor synchronization, data acquisition, test control, and data handling. Internal and external sensor processes need to be synchronized in order for measurement data to be compared. In data acquisition tasks raw data from sensors is recorded by the cRIO. Some portions of the test setup are controlled by the cRIO data acquisition (DAQ) unit. The software necessary for this task falls under the test control category. After data has been acquired, it needs to be processed into a usable format for later analysis. This is essentially a conversion from raw digital or analog data to engineering values, and then the process used to save the data in a single file.

Data acquisition tasks from analog sensors are handled using a previously developed LabVIEW library that utilizes National Instruments DAQmx Application Programming Interface (API). The Device Under Test (DUT) is triggered using functions from the same library. Data processing routines are built using a producer-consumer architecture, which allows data acquisition and data processing to run simultaneously but at different rates. Thus, data is acquired without introducing delays, and once acquisition stops data processing continuous until all the data has been handled. Synchronization is maintained, data is acquired as quickly as possible, and no data is lost during processing.

In order to achieve sensor synchronization, data acquisition processes both internal and external to the cRIO are tied to a single start trigger. This ensures that the start time of each process is simultaneous. Once data acquisition is stopped, the collected data is sorted using LabVIEW's built-in file handling functions to collect the data into a single file for later analysis. Control routines are handled using this same DAQmx based library, and synchronized in the same manner.

The software solution created utilizes a number of custom and standard libraries/APIs

in order to achieve the required tasks. The solutions reached in order to capture data from the wide array of selected sensors, along with synchronizing the data acquisition such that the results from these various sensors can be compared to each other, have a high degree of complexity. Not only does this software provide a solution to the project requirements presented by DLR, but also demonstrates the usage of in-house sensor libraries and points to possible avenues of future development in these libraries.

6.3. Electrical Hardware

The electrical hardware design and solution are the resulting answer to RQ2. Sections 3.2 and 4.1 cover the design solution and implementation created, respectively. Before project start the overall system design and several specific hardware components, such as sensors and main DAQ unit, were already selected by DLR. Much of the designed solution is thus related to the integration of these components into a functional system, keeping in mind both the operational requirements and the physical constraints of the setup.

The finalized electrical hardware solution takes the desired system scheme and selected components and integrates them into a functional whole. The main data acquisition unit is tied into sensors directly when possible, and through synchronized sub-DAQs when not. The overall solution takes into account the physical restrictions of the design, and minimizes the impacts.

6.4. Verification

The methodology used to answer RQ3 and its subquestions is elaborated on in Chapter 2 while the results of applying this methodology to the test setup system are shown in Chapter 5. Here, the validity of this methodology is examined based on the acquired results.

6.4.1. Verification Method Selection

It can be seen that the verification selection methodology outlined in Chapter 2 is not necessarily valid. The proposed selection methodology is that requirements related to the specific performance of the system (i.e. those laid out in Table **??**) would be verified through testing, and those dealing with the presence of features or capabilities would be verified through review of design or inspection. Instead, it can be seen that in order to ensure that some features or capabilities are actually present, testing rather than a review of design is required. Data acquisition feature requirements in particular need to undergo testing for verification. It is not enough to have software written that is *supposed* to read the output of a sensor, that software needs to actually be tested in order for the verification of it to be valid. This verification can take place at the same time as the verification test for the specific performance of the feature (e.g. testing that temperature measurement is working in general in addition to testing what the accuracy of the measurement is). The serial tests and the APRS-BP tests are used to verify requirements that simply specify that a feature is present, rather than the actual accuracy or precision of the feature. These verification tests are necessary, however, to prove that the function is actually working. Thus, the selection of verification method

is more complex than choosing testing for requirements related to performance and review of design/inspection for requirements regarding the presence of features. There is a level of complexity of a feature that requires a verification test, and the feature requirements chosen for testing are dependent on the interaction of hardware and software, or multiple modules working together. This creates a more rigorous verification method selection, where simple feature requirements are verified through review of design while complex feature requirements are verified through testing. Further research along this line may produce a more refined approach to method selection that is able to take these nuances properly into account.

6.4.2. Verification Test Plan Template

The overall goal of the created verification test plan template was to create a standard method for verification test planning that ensured traceability, both metrologically and to specific requirements. The evaluation criteria for these test plans then evaluates how well the individual test plans are written, and the total sum of these evaluations shows the general efficacy of the test plan format and the plans themselves. From the combined evaluation criteria table, it can be seen that some improvement to the requirements traceability can be made, while metrologic traceability is thoroughly kept with all of the created test plans. Additional improvement in methods to ensure variable isolation are needed as many of the created test plans do not thoroughly address this criteria, while the repeatability of the generated test plans are overall acceptable.

The developed test plan format does an acceptable job of ensuring that the developed evaluation criteria is fulfilled, along with addressing the previously discussed literature gap. These results also provide several points for further research and development. The verification test plan template provides a starting point for an updated template that is more effective in ensuring traceability to a specific requirement and isolation of influencing variables. For example, providing a justification for the verification criteria in relation to the target requirement, and providing a list of influencing variables and how they are mitigated. Additional research into a more quantifiable evaluation scale would also be beneficial, especially as it can be applied to verification test plans in general, not just ones following this template. This would allow easy comparison between proposed verification tests for the same system, and provide an acceptability baseline that verification test plans need to exceed.

References

Cortes, E. D., Mendoza, D. A., & Rodriguez, G. W. (2020). Design and construction of test benches for small scale aerospace systems [Export Date: 28 April 2025; Cited By: 0]. *2020 IEEE ANDESCON, ANDESCON 2020*. https://doi.org/10.1109/ANDESCON50619.2020.9271985

Cremades, L. V., & Ponsich, A. (2025). Simple and objective determination of criteria weights for evaluating alternatives when using the analytic hierarchy process. *International Journal of the Analytic Hierarchy Process*, *16*(3). https://doi.org/10.13033/ijahp.v16i3.1177

Deiss, H., & Bowman, A. (2023). Nasa: Distinctions between product verification and product validation. https://www.nasa.gov/reference/2-4-distinctions-between-product-verification-and-product-validation/

ECSS. (2018). Space engineering verification. https://ecss.nl/wp-content/uploads/2018/02/ECSS-E-ST-10-02C-Rev.1(1February2018).pdf

Eurolab. (2020). Eurolab cook book no .1: Selection, verification and validation of methods. https://drive.google.com/file/d/1Gnou_d4xLM6uHsoQ9-swNlmoEP04zSoW/view

Hernandez, A. D. C., Kramer, E. L., & Masterson, R. A. (2022). Development of a hardware-in-the-loop testbed for rotating synthetic aperture telescopes [Export Date: 28 April 2025; Cited By: 2]. *IEEE Aerospace Conference Proceedings*, *2022-March*. https://doi.org/10.1109/AERO53065.2022.9843452

ISO. (2017). General requirements for the competence of testing and calibration laboratories. https://www.iasonline.org/wp-content/uploads/2021/02/ISO-IEC-17025-2017-IAS.pdf

KAVLICO. (2020). Aerospace procedure for control of inspection test equipment. https://www.sensata.com/sites/default/files/a/kavlico-control-of-inspection-manual-ap0411.pdf

Piccolo, A., & Menzione, F. (2024). Gnss test bench calibration for space service volume applications [Export Date: 13 May 2025; Cited By: 1]. *2024 IEEE International Workshop on Metrology for AeroSpace, MetroAeroSpace 2024 - Proceeding*, 134–139. https://doi.org/10.1109/MetroAeroSpace61015.2024.10591570

Welch, R. V. (2005). Verification and validation of mars exploration rover surface capabilities. https://hdl.handle.net/2014/37825

A

Criteria Weight Determination

The following is the full process used to calculate the relative weights of the verification test plan evaluation criteria.

This analysis is performed utilizing the methodology laid out by Cremades and Ponsich in *Simple and Objective Determination of Criteria Weights for Evaluating Alternatives when using the Analytic Hierarchy Process* (Cremades and Ponsich, 2025).

Comparison of criteria begins through the construction of a judgment matrix, where each element represents the relative importance of a criteria in relation to another on a scale of 1-9.

$$M = \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{31} & v_{32} & v_{33} & v_{34} \\ v_{41} & v_{42} & v_{43} & v_{44} \end{bmatrix} = \begin{bmatrix} 1 & v_{12} & v_{13} & v_{14} \\ {}^{1}/_{v_{12}} & 1 & v_{23} & v_{24} \\ {}^{1}/_{v_{13}} & {}^{1}/_{v_{23}} & 1 & v_{34} \\ {}^{1}/_{v_{14}} & {}^{1}/_{v_{24}} & {}^{1}/_{v_{34}} & 1 \end{bmatrix} \tag{A.1}$$

Each element v_{ij} is a representation of the relative importance of criteria C_i with respect to criteria C_j. Once this matrix is constructed, a criteria weights vector is calculated by normalizing the values in matrix M by dividing each element by the sum of their respective column. The weight of each criteria is then found by averaging each row.

In order to determine the relative weights of evaluation criteria n, each criteria is first assigned a value between 1 and p, where $p \leq n$ as criteria can be assigned the same value. A value of 1 represents the highest relative importance, while a value of p represents the lowest relative importance. The element value is then found through:

$$v_{ij} = \begin{cases} \frac{\left(\frac{c_j}{c_i} - 1\right) * 8}{p-1} + 1 & \frac{c_i}{c_j} < 1 \\ 1 & \frac{c_i}{c_j} = 1 \\ \frac{1}{\frac{\left(\frac{c_i}{c_j} - 1\right) * 8}{p-1} + 1} & \frac{c_i}{c_j} > 1 \end{cases} \tag{A.2}$$

The criteria are assigned the following importance values based on a pair-wise comparison:

$c_1 = 1$
$c_2 = 1$
$c_3 = 2$
$c_4 = 3$

It can be seen then that $p = 3$ for this criteria selection. The value of each element must then be calculated:

v_{12} :

$$\frac{c_1}{c_2} = 1,$$
$$v_{12} = 1$$

v_{13} :

$$\frac{c_1}{c_3} = \frac{1}{2} < 1,$$
$$v_{13} = \frac{\left(\frac{2}{1} - 1\right) * 8}{3-1} + 1 = \frac{8}{2} + 1 = 5,$$
$$v_{13} = 5$$

v_{14} :

$$\frac{c_1}{c_4} = \frac{1}{3} < 1,$$
$$v_{14} = \frac{\left(\frac{3}{1} - 1\right) * 8}{3-1} + 1 = \frac{(2) * 8}{2} + 1 = \frac{16}{2} + 1 = 9,$$
$$v_{14} = 9$$

v_{23} :

$c_1 = c_2, v_{13} = v_{23},$
$v_{23} = 5$

v_{24} :

$v_{14} = v_{24}$
$v_{24} = 9$

v_{34} :
$\frac{c_3}{c_4} = \frac{2}{3} < 1$
$v_{34} = \frac{\left(\frac{3}{2} - 1\right) * 8}{3 - 1} + 1 = \frac{\left(\frac{1}{2}\right) * 8}{2} + 1 = \frac{\left(\frac{8}{2}\right)}{2} + 1 = \frac{4}{2} + 1 = 3,$
$v_{34} = 3$

The resulting M matrix is:

$$M = \begin{bmatrix} 1 & 1 & 5 & 9 \\ 1 & 1 & 5 & 9 \\ 1/5 & 1/5 & 1 & 3 \\ 1/9 & 1/9 & 1/3 & 1 \end{bmatrix}$$

Next, the normalized matrix M^* must be calculated by dividing each column j by the sum of its elements. The sum vector **S** is:

$S_1 = 1 + 1 + \frac{1}{5} + \frac{1}{9} = \frac{104}{45}$

$S_2 = S_1 = \frac{104}{45}$

$S_3 = 5 + 5 + 1 + \frac{1}{3} = \frac{34}{3}$

$S_4 = 9 + 9 + 3 + 1 = 22$

$$\mathbf{S} = \begin{bmatrix} \frac{104}{45} & \frac{104}{45} & \frac{34}{3} & 22 \end{bmatrix}$$

Dividing each column of M by its sum yields the matrix M^*:

$$M^* = \begin{bmatrix} \frac{45}{104} & \frac{45}{104} & \frac{15}{34} & \frac{9}{22} \\ \frac{45}{104} & \frac{45}{104} & \frac{15}{34} & \frac{9}{22} \\ \frac{45}{520} & \frac{45}{520} & \frac{3}{34} & \frac{3}{22} \\ \frac{45}{936} & \frac{45}{936} & \frac{1}{34} & \frac{1}{22} \end{bmatrix}$$

The weight w_i is then found by averaging the values of each respective row in matrix M^*:

$$w_i = \frac{\sum_{j=1}^{n} v_{ij}}{n}$$

$$w_1 = \frac{\sum_{j=1}^{4} v_{1j}}{4} = \frac{\frac{45}{104} + \frac{45}{104} + \frac{15}{34} + \frac{9}{22}}{4} = \frac{\frac{16683}{9724}}{4}$$

$$w_2 = w_1 = \frac{\frac{16683}{9724}}{4}$$

$$w_3 = \frac{\sum_{j=1}^{4} v_{3j}}{4} = \frac{\frac{45}{520} + \frac{45}{520} + \frac{3}{34} + \frac{3}{22}}{4} = \frac{\frac{3867}{9724}}{4}$$

$$w_4 = \frac{\sum_{j=1}^{4} v_{4j}}{4} = \frac{\frac{45}{936} + \frac{45}{936} + \frac{1}{34} + \frac{1}{22}}{4} = \frac{\frac{1663}{9724}}{4}$$

Converting to a decimal rounding to the fourth significant figure, and then converting to a percentage yields percentage weights of:

$w_1 = 42.89\%$
$w_2 = 42.89\%$
$w_3 = 9.94\%$
$w_4 = 4.28\%$

Rounding these values to a whole number yields the final weights of:

$w_1 = 43\%$
$w_2 = 43\%$
$w_3 = 10\%$
$w_4 = 4\%$

B

Least Squares Sine Function Derivation

In order to fit the recorded data to a sine curve, a least-squares methodology is used. The fit formula is:

$$y(t) = A\sin(2\pi f t + \varphi) + B \tag{B.1}$$

Where $y(t)$ is the current voltage value at a given time, A is the amplitude, f is the ordinary frequency, t is the time in seconds, φ is the phase, and B is the offset. Time t is given on the assumption of a fixed sample frequency, and ordinary frequency f is set to the value of the transmitted data.

This formula is linearized to:

$$y(t) = A\sin(2\pi f t)\cos(\varphi) + A\cos(2\pi f t)\sin(\varphi) + B \tag{B.2}$$

Since ft is known at any given point, the system can be further reduced to:

$$y(t) = AS_t\cos(\varphi) + AC_t\sin(\varphi) + B \tag{B.3}$$

Where $S_t = \sin(2\pi f t)$ and $C_t = \cos(2\pi f t)$. This can then be turned into a matrix equation:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} S_0 & C_0 & 1 \\ S_1 & C_1 & 1 \\ S_2 & C_2 & 1 \\ \vdots & \vdots & \vdots \\ S_n & C_n & 1 \end{bmatrix} \begin{bmatrix} A\cos(\varphi) \\ A\sin(\varphi) \\ B \end{bmatrix} \tag{B.4}$$

Let:

$$\mathbf{Y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{B.5}$$

$$\mathbf{L} = \begin{bmatrix} S_0 & C_0 & 1 \\ S_1 & C_1 & 1 \\ S_2 & C_2 & 1 \\ \vdots & \vdots & \vdots \\ S_n & C_n & 1 \end{bmatrix} \tag{B.6}$$

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \begin{bmatrix} A\cos(\varphi) \\ A\sin(\varphi) \\ B \end{bmatrix} \tag{B.7}$$

X can then be solved for through:

$$\mathbf{X} = (\mathbf{L}^T\mathbf{L})^{-1}\mathbf{L}^T\mathbf{Y} \tag{B.8}$$

Once vector **X** is obtained, the unknown variables can then be found:

$$\varphi = \arctan\left(\frac{X_2}{X_1}\right) \tag{B.9}$$

$$A = \frac{X_1}{\cos(\varphi)} \tag{B.10}$$

$$B = X_3 \tag{B.11}$$