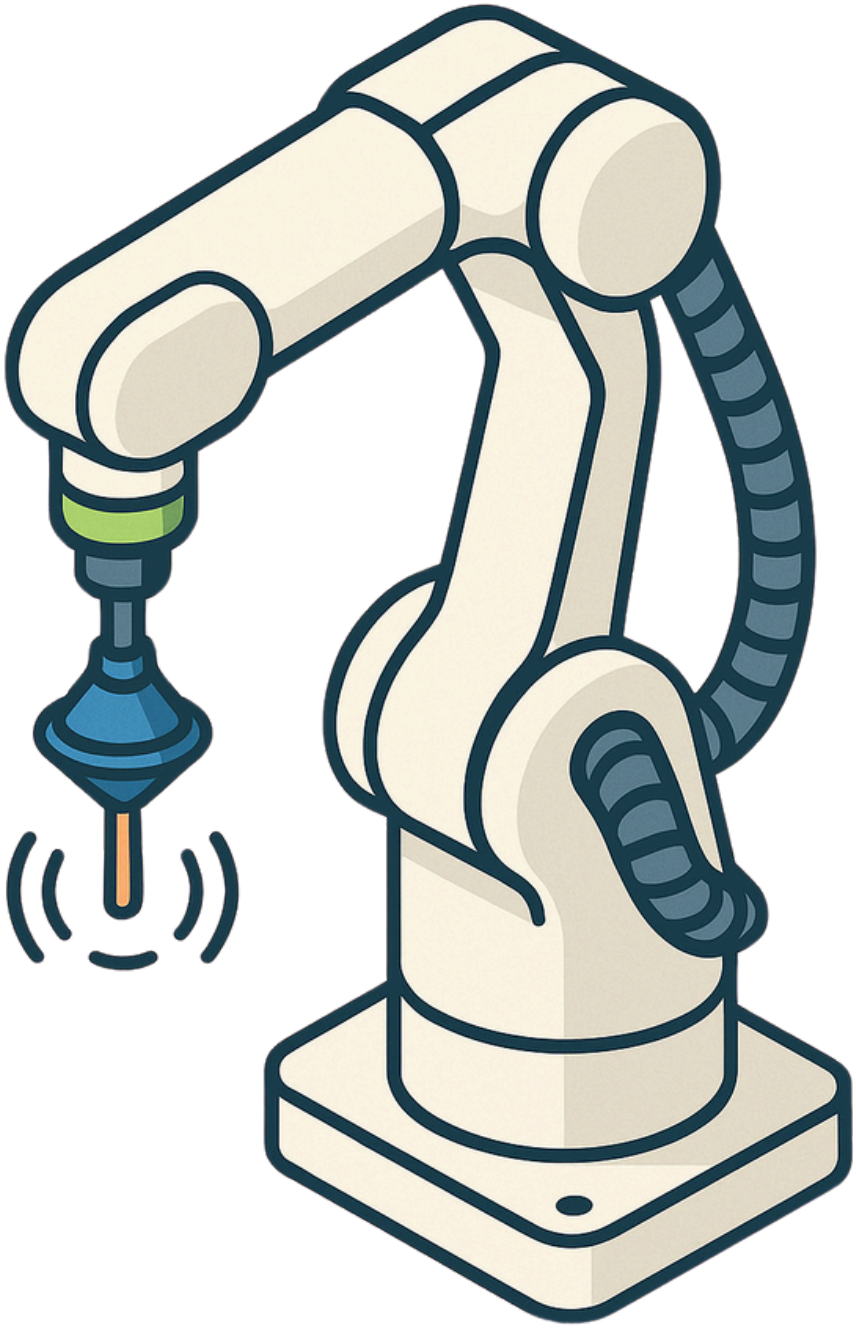
BSc Thesis

Automated Antenna Radiation Measurements: Control interface

G.M.Hak
T. Rietjens

Delft University of Technology



THz Sensing Group

TU Delft Delft University of Technology

Subgroup 2

Bachelor Graduation Project

by

G. Hak T. Rietjens

to obtain the degree of Bachelor of Science

at the Delft University of Technology,

Student numbers:	5572231 & 5768322
Project duration:	April 23, 2025 – June 16, 2025
BAP supervisors:	Prof. Dr. N. Llombart, Dr. H. Zhang,
External jury member:	Prof. Dr. Ir. W. van Driel

Cover image generated using OpenAI's ChatGPT/DALL·E using an image of the real measurement setup.

TU Delft

Preface

Over the last eight weeks, this Bachelor Graduation Project (BAP) provided the opportunity to work with the Terahertz Sensing Group at the TU Delft to help contribute to the promising technology of robotic arm antenna characterisations. Together with Huib Maas, Fabian Wichman, Sven Lemsom, Sitti Romijn, Danylo Zavoloko, and Matthieu Gillain, we have successfully developed an automated measurement system for both near-field and far-field antenna tests, leveraging a graphical control interface and various optimization strategies to perform accurate measurements. We would like to thank our peers for their efforts and enjoyable collaboration.

We are especially grateful to our project supervisors, Prof. Dr. Nuria Llombart, Dr. Daniele Cavallo, Dr. Huasheng Zhang, Ir. Jingling Geng, Ir. Dunja Lončarević, and Ir. Roderick Barroso for their devoted efforts and valuable feedback.

G. Hak T. Rietjens
Delft, June 26, 2025

Abstract

Accurately characterizing antenna radiation patterns in the millimetre-wave (mmW) frequency range presents significant challenges due to the short wavelengths involved. A measurement system that incorporates a 6-axis robotic arm is implemented to gain more positional control over the antenna measurement. This thesis presents the software to control the system, including the robotic arm and a Vector Network Analyser, in a user-friendly manner. Additionally, it dives into the influence of the characteristics and movement of the robotic arm on the setup. To ensure reliability and maintainability of the codebase, the software was structured using several objects and data classes, each responsible for a specific part of the system. Tests of the system showed a solid foundation of the graphical user interface and the back-end software architecture. Analysis of the repeatability of the system revealed that results may deviate by magnitude differences of 0.8 dB and phase differences up to 30°. Compared to a calibrated position of the system, displacements in individual joints led to significant phase difference of up to 20°. There was no correlation found in deviations of magnitude or phase response and the settle time between the movements of the robot and the measuring of the VNA.

Glossary

BAP Bachelor Graduation Project

VNA Vector Network Analyser

DUT Device Under Test

AUT Antenna Under Test

GUI Graphical User Interface

IFBW Intermediate Frequency Bandwidth

API Application Programming Interface

DFT Discrete Fourier Transform

IDFT Inverse Discrete Fourier Transform

SOLT Short-Open-Load-Thru

TRL Thru-Reflect-Line

List of Figures

Contents

1 Introduction

Accurate measurements of electric fields are essential for the characterisation of fabricated antennas. Traditional test setups often rely on scan with fixed positions or manually operated positioners of the antennas, which can be time-consuming, inflexible, and sometimes inaccurate. This thesis proposes the use of a 6-axis robotic arm to provide more flexibility in different measurement setups; a robotic arm allows for reliable and precise position, and does not require manual labour while conducting the measurement.

This project's main focus lies on the development of a control interface which allows for multipurpose antenna measurements using the robotic arm. The control interface is divided into several subcomponents; a path planning algorithm for controlling the robot arm's movements; grid generating functions for defining measurement points in both spherical and planar surfaces; analytical functions for processing and plotting the measured s-parameters; and a user interface that integrates all the hardware and software components required for each step of the measurement process. This thesis focuses on the last component.

Next to developing a control unit to combine all subcomponents, this thesis focuses on the design of a graphical user interface (GUI) in MATLAB, which includes controlling the devices via a user-friendly interface and visualizing the results with real-time plots. This makes field measurements of an antenna accessible to users without requiring extensive training. At last, the thesis analyses possible errors caused by the robotic arm, such as phase errors due to cable movement and repeatability considerations.

1.1. State-of-the-art Analysis

This section discusses the state of the art, covering existing research that supports further development, as well as areas where the technology can still be improved.

1.1.1. Usage of Robotic Arms

The use of a robotic arm in antenna measurements is not a new concept. Several scientific studies have already explored the possibilities for near-field measurements, due to the increasing availability of precise robotic positioning units and improving accessibility of the tool-chain to control these [1]. With ongoing research in the sub-THz band (100-300 GHz) for antenna designs, particularly for potential sixth-generation (6G) wireless communication systems, the accurate characterization of complex antennas demands more sophisticated measurement setups[2]. The research around the use of a robot arm relies mostly on the use of smaller robot arms that would fit in a lab environment. Therefore, near-field measurements, such as azimuthal sweeps or planar grids of sampling points, are conducted, with the results transformed into far-field data [1], [2].

In addition to azimuthal sweeps and planar grids, several studies raise the potential of using a robotic arm for spherical sampling points. Researches, like [3] from S.F. Gregson, even use the spherical sampling points from the near-field to obtain the far-field data.

Another robotic approach is presented in [4] by B. Sievert and J.T. Svejda; instead of using a 6-axis robotic arm, this study explores a dome-shaped robotic system, specifically designed for taking spherical measurements or performing azimuthal sweeps only.

1.1.2. Optimization Technics

Measuring the electromagnetic field of an antenna using a robotic setup can be a time-consuming process, particularly when numerous sample points are required. The total duration of the measurement process depends on both the number of sampling points and the time required to acquire data at each point. The time per point is influenced by the movement speed of the robotic arm and the measurement time of the VNA. Several strategies can be applied to reduce this sampling time, such as increasing the intermediate frequency bandwidth, reducing the number of frequency points per position, or bypassing the graphical user interface of the VNA [5].

Other considerations to reduce the time needed for one measurement would be to decrease the number of points or time required for the robotic arm between points. While subgroups 3 and 4 address these aspects more, one study by B. L. Moser and J. A. Gordon [6] introduces an interesting variation to the setup by incorporating a second robotic arm. One of the arms carrying the antenna under test (AUT) and the other holding the probe antenna. In this setup, the two robots collaborate to position themselves such that the absolute position becomes less important, and only the relative position between the probe and the AUT matters. This significantly reduces the required movement between sampling points and contributes to faster measurements[6].

1.1.3. Possible Errors

Next to the advantages of this setup with a robotic arm, this technology may also introduce certain errors. Each degree of freedom in the robotic arm can contribute to positioning errors, potentially affecting the accuracy of the measurements. There are methods available to simulate and predict the impact of these positional deviations on the measurement results [7]. The research of B. Sievert and J.T. Svejda [4] also discusses the link between mechanical repeatability of the robotic system and the wavelength of the measured signal. Moreover, the movement of the robot arm may influence the calibration of the cables attached to the antennas; this is what part of this thesis also aims to discover.

1.1.4. Graphical Controller

For the measurement process to function reliably, it is essential that all components work together seamlessly and that each step is executed and recorded accurately. The required components and basic steps for near-field antenna measurements are explained in existing literature. For example, Kalashnikov [8] describes a clear approach, describing each step of the process. For this thesis, it was also important to include the user in the design process by providing a graphical interface to control the system. Fortunately, MATLAB offers an easy way to design and build such an app using its App Designer tool [9], and it provides clear and extensive documentation to support development [9], [10].

1.2. Problem Definition

The introduction of a robotic arm into the antenna measurement process offers greater flexibility and reduces the amount of manual labour compared to traditional methods. However, using a robot also complicates the setup and may introduce new issues and errors. All the components in the setup need to work together; the robot arm needs to move correctly; the VNA needs to collect the data; the collected data must be processed and saved correctly. All while the user should be able to easily start a new antenna measurement, monitor its progress, and access the results through a clear graphical interface.

This thesis mainly focuses on the control interface part of the system. While other subgroups work on controlling the robot arm and analysing the collected data, this subgroup develops the software needed to integrate all components and make the system work as a whole. As shown in figure 1.1, this includes data collection, a graphical interface, and controlling the VNA through a central control unit. The software also needs to work together with the tools from other subgroups to control the robot

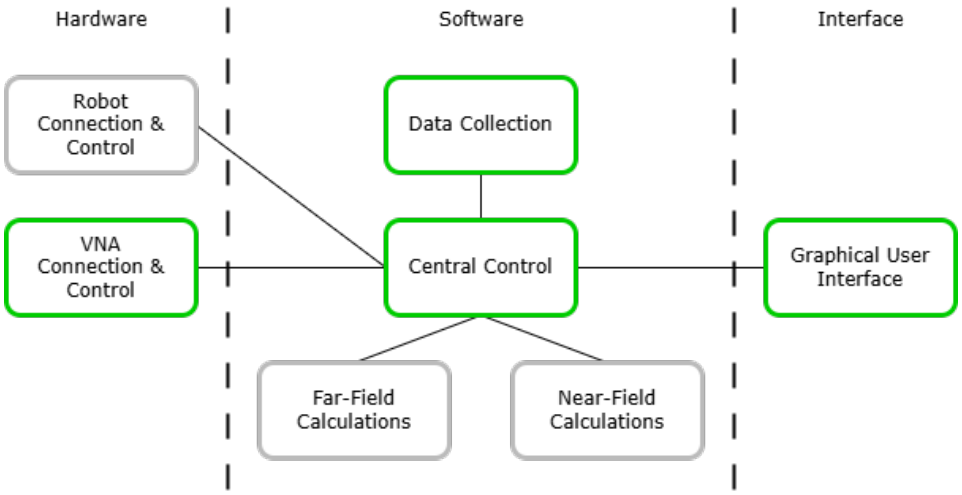


Figure 1.1: Diagram showing the relationship between each subcomponent

and process the measurement data. The boxes in green represent systems that are developed by this subgroup, and the ones in grey are systems developed by the other subgroups that need to be integrated into the control interface.

1.3. Thesis Synopsis

This thesis is the documentation for the work done by subgroup 2 over the last eight weeks of the Electrical Engineering Bachelor Graduation Project (BAP). It begins by briefly providing a description of the overall project and defining some overarching requirements, before focusing on the specific content for subgroup 2. The detailed project description for this thesis is presented in Chapter 2, along with the programme of requirements that will be pursued. Afterwards, chapter 3 delves into the design process and justifies the choices made with respect to the programme of requirements; important algorithms and design structures will be explained and diagrammed in detail. Chapter 4 presents the prototype implementation and results, and then validates the performance. Furthermore, difficulties and successes experienced during development will be reviewed. Next, Chapter 5 will use the prototype system to analyse the potential error sources introduced by robotic solutions for antenna measurements, with a focus on phase characteristics and repeatability. These will be state-of-the-art findings which will contribute to the field of antenna measurements. Additionally, discussions in section 5 will analyse the overall findings of the investigations and propose further experiments. lastly, recommendations for future work will be discussed.

2

Programme of Requirements

This chapter presents the targets and deliverables that have significance for subgroup 2. It will start with presenting the general goals for the project by clearly summarising the expectations of the supervisors. Next, it will delve into the functional and non-functional requirements and constrains of the project, which will be presented as SMART Targets.

2.1. Project Description

The process by which antenna radiation measurements are done is often expensive, inflexible and requires a certain extent of expertise to perform. Current implementations involve the use of large anechoic chambers[4], [11], [12] or only perform Near-Field radiation measurements [8], [13]. With the goal of streamlining this process for the professional market, the Terahertz Sensing Group at the TU Delft has invested in the promising technology of antenna measurements using a robotic arm[2], [14], [15]. The project that this BAP group will undertake is to build atop the initial research conducted by this department, to automate and generalise the antenna radiation measurement.

Table 2.1: Subgroup responsibilities

Subgroup 1	**Subgroup 2**	**Subgroup 3**	**Subgroup 4**
Programming and automation of robotic arm	Control interface for robotic arm and network analyser	Far-field antenna pattern measurements (spherical acquisition)	Near-field antenna pattern measurements (planar acquisition)

The BAP project has been split into four subgroups of two students, which will focus on independent capabilities of such a general antenna measurement system. The distribution of the tasks is summarised in table 2.1. The authors of this thesis represent subgroup 2, which is focussed on designing and building a general control interface to orchestrate the measurement process. This involves interfacing with all the hardware and software components and handling user inputs through a graphical user interface (GUI). Once completed, the focus moves to the testing and analysis of potential error sources introduced by robotic solutions, including phase errors from cable movement and repeatability considerations.

2.1.1. Needs and Expectations

Subgroup 2 will build atop the current robotic measurement system of the Terahertz Sensing Group; thus, it is instructive to define the current capabilities of the measurement system and the progressions that subgroup 2 will aim to make. Table 2.2 summarises the current capabilities and a general list of key future performance indicators for subgroup 2's system. The key performance indicators will later be refined to form the specific programme of requirements.

Table 2.2: A summary of the current capabilities and the key performance indicators that subgroup 2 will implement

Current Capabilities	Key Future Performance Indicators
Move a planar CNC Robot using MATLAB.	Move a 6-axis robotic arm from within MATLAB.
Measure and set VNA parameters from within MATLAB.	An updated version of MATLAB code to measure data from the VNA from within MATLAB.
A centralised controller for planar near-field measurements with limited range using a CNC.	A centralised controller interfaces with the VNA and a 6-axis robot to perform near- and far-field measurements.
An intuitive GUI allows section of only planar measurement grids.	An intuitive GUI allows flexible selection of planar and spherical measurement grids and VNA parameters.
	The 6-axis robot's motion is dependent on user inputs and various paths can be followed.
Separate processing needs to be applied to display measurements.	The data and processed results can be displayed on the GUI.

To aid in achieving the desired key performance indicators, the supervisors proposed certain project milestones that subgroup 2 will work towards. Figure 2.1 presents the advised workflow, which illustrates their expectations and aids in the formation of the refined final requirements. Additionally, It acts as a reference that subgroup 2 will use to communicate the progression of the project to the supervisors.

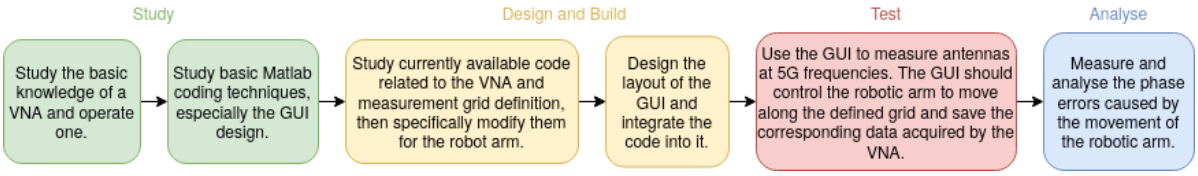


Figure 2.1: The workflow expected by the supervisors for subgroup 2.

2.2. Functional Requirements

With the project defined, and the expectations identified, the final requirements can be formulated. They will be classified into four classifications, mandatory requirements, trade-off requirements, functional requirements and non-functional requirements. Mandatory requirements are capabilities that must be present in the final solution, these must be atomic and described as binary, meaning it either complies or doesn't. Trade-off requirements are capabilities that can be optimised to increase end-user satisfaction. The functional requirements are specific capabilities the system shall perform, and non-functional requirements are properties that the system shall exhibit.

2.2.1. Mandatory Requirements

Functional

1. The system must be able to connect to a Fanuc-CRX-20iA-L Robot arm.
2. The system must be able to move the robot arm in discrete steps manually from the GUI.
3. The system must be able to move the robot arm to a specific x, y, z and w, p, r position from the GUI, measured in mm and degrees respectively .
4. The system must be able to move the robot arm to all positions on a predefined grid of points.
5. The system must allow the user to enter the address of the VNA to connect to.
6. The system must be able to connect to a Keysight E8361A VNA
7. The system must be able to set the following settings on the VNA from within MATLAB:

   (a) The start Frequency of the sweep, measured in Hz.
   (b) The stop Frequency of the sweep, measured in Hz.
   (c) The number of equally spaced frequency points to measure during the frequency sweep.
   (d) The intermediate frequency bandwidth for each point in the sweep, measured in Hz.

8. The system must be able to retrieve the same settings as stated in the specifications of 7 from the VNA. from within MATLAB:

9. The system must be able to measure the arrays of complex S11, S12, S21 and S22 parameters from the VNA using MATLAB in every measurement cycle.

10. The system must be able to store the following information on the hard disk for each measurement

    (a) The (x, y, z) coordinate of where the measurement is taken, measured in mm.
    (b) The (w, p, r) orientation of the probe when the measurement is taken, measured in degrees.
    (c) The polarisation of the measurement, represented as a logical 0 or 1 for co- and cross-polarisations, respectively.
    (d) The arrays of complex s-parameters S11, S12, S21, S22.
    (e) The start Frequency of the sweep, measured in Hz.
    (f) The stop Frequency of the sweep, measured in Hz.
    (g) The intermediate frequency bandwidth for each point in the sweep, measured in Hz.
    (h) The frequency of interest for the DUT, measured in Hz.
    (i) The aperture of the DUT, measured as a fraction of the wavelength of interest.
    (j) The Near Field distance of the DUT, measured in mm.
    (k) The x dimension of the DUT, measured in mm.
    (l) The y dimension of the DUT, measured in mm.
    (m) The angle (θ): the rotation angle around the X-axis, measured in degrees.
    (n) The angle (ϕ): the rotation angle around the Z-axis, measured in degrees.
    (o) A logical 0 or 1 indicating if the grid is centred around the DUT.

11. The system must allow the user to choose which data point they want to see the result of.

12. The system must allow the user to choose which S-parameter they want to see the result of.

13. The system must be able to display the raw frequency-domain magnitude and phase data from a user-selected data point and S-parameter.

    (a) The x-axis (frequency) must be measured in GHz.
    (b) The y-axis (magnitude response) must be measured in dB.
    (c) The y-axis (phase response) must present the unwrapped phase in degrees (continuous phase).

14. The system must be able to display the raw time-domain data from a user-selected data point and S-parameter.

    (a) The x-axis (time) must be measured in ns.
    (b) The y-axis (absolute response) must be purely real and is unitless.

15. The system must allow the user to select a start time for time-gating.

16. The system must allow the user to select a gate size for time-gating.

17. The system must be able to display the time-gated frequency-domain magnitude and phase response using user-selected gate start time and size.

    (a) The x-axis (frequency) must be measured in GHz.
    (b) The y-axis (magnitude response) must be measured in dB.
    (c) The y-axis (phase response) must present the unwrapped phase in degrees (continuous phase).

18. The system must present a graph showing the progress of the measurement process, updated in real-time.

19. The system must present the final directivity plot when all measurements are completed.

Non-functional

20. The antenna measurement system must operate at a minimum frequency of 26.5 GHz.
21. The antenna measurement system must operate at a maximum frequency of 40 GHz
22. The system must be written in MATLAB.
23. The system must be able to continuously perform measurements for at least 6 hours.

2.2.2. Trade-off Requirements

Functional

24. The system should remind the user to perform SOLT Calibration before measuring starts.
25. The system should remain interactive when the robot is disconnected.
26. The system should remain interactive when the VNA is disconnected.
27. The system should allow the user to load previously taken measurement points to use for post-processing.

Non-functional

28. All used MATLAB Toolboxes should be listed as dependencies in the readme.txt file in the git repository.
29. An explanation of the deployment of the system should be given in the readme.txt file in the git repository.

3

Software Design & Development

This chapter discusses the design choices made during the development of the software and how these choices meet the requirements outlined in Chapter 2. It begins with an overview of the complete system and how the various subcomponents are integrated. Afterwards, the design decisions for each subcomponent are discussed in detail.

3.1. Backend - High level Overview

The software of the system is structured around an object-oriented design, to ensure maintainability and readability of the code. Figure 3.1 presents an overview of the main software components and how they interact with each other.

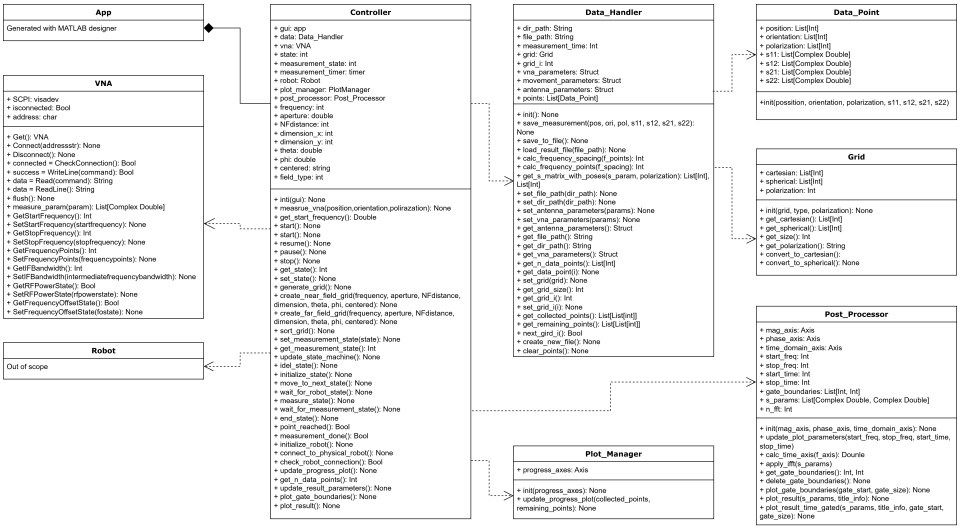


Figure 3.1: Class diagram of the Object-oriented software design

Each class is responsible for a specific task, promoting a clear separation of functionality throughout the code. As shown in the UML class diagram in figure 3.1, the Controller class is connected to all components of the system and acts as the central hub that connects and coordinates all other components, effectively tying the entire system together. Below is a brief description of each class and its responsibilities:

Controller: Central class responsible for managing interactions between the hardware devices and other software components. It also executes all the steps required to perform an antenna measurement through a state machine

App: Responsible for the GUI. It includes all the properties related to the graphics and implements the methods that control the app's behaviour.

VNA: Responsible for controlling the VNA. All the methods needed to connect to the VNA as well as reading data are included in this component.

Robot: This class was designed by subgroup 1. It contains all functionality to control the robot arm.

Data_Handler: Manages the all data and saves it in a file.

Data_Point: Data class that stores the measurement data of a single sampling point in a format that can be easily used in the code.

Plot_Manager: Manages the progress plot in the GUI.

Grid: Data class that stores the coordinates of the sampling point in a format that can be easily used in the code.

Post_Processor: Manages the logic behind time gating and deals with the results plots in the GUI.

3.2. Controller

While each class handles its own specific task, the controller ties everything together, from integrating with the GUI to managing data flow. It also oversees the entire measurement process, tracking its state and ensuring each step is executed in order.

3.2.1. Measurement process

Requirement 4 states that the system must be able to move the robot arm to all positions on a predefined grid of points. To ensure this functionality is carried out in the correct order, the Controller class manages the measurement sequence using a state machine. As shown in figure 3.2, it consists of six functional states, with the system looping through the middle four during the measurement process, moving the robot arm to the next point and collecting the data.

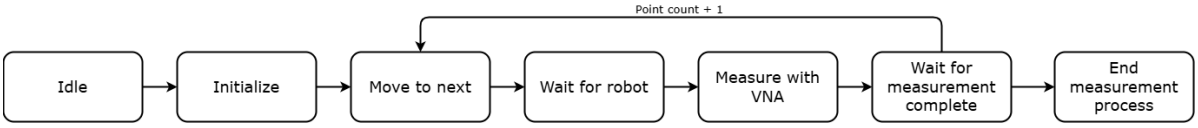


Figure 3.2: Measurement state machine

The list below provides a brief description of each state in the measurement process.

0. **Idle:** Do nothing.
1. **Initialize:** Resetting and preparing all necessary program variables, so the measurement can start from a clean setup.
2. **Move to next:** Move the robot arm to the next (or first) sampling point.
3. **Wait for robot:** Stay in this state until the robot arm has reached the sampling point
4. **Measure with VNA:** Measure the S-parameters using the VNA
5. **Wait for measurement complete:** Stay in this state until the VNA has collected the S-parameters arm has reached the sampling point.
6. **End measurement process:** Finalising the measurement. This state is reached when there are no more sampling points left.

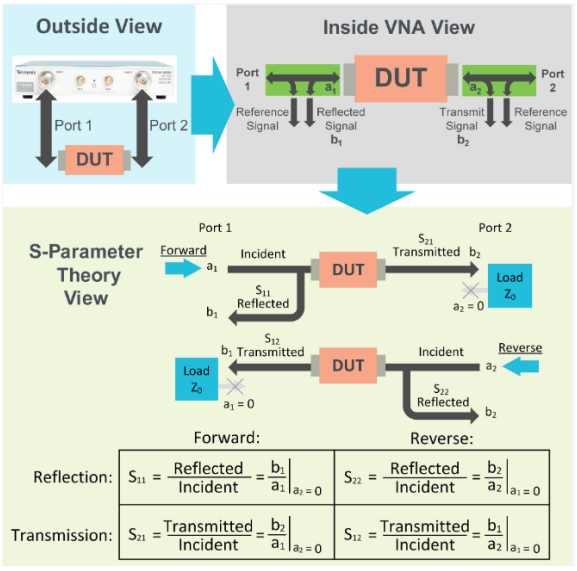
A timer is used to periodically trigger the state machine by calling the `update_state_machine()` function, as shown in listing D.1 in Appendix D. This function evaluates the current state and calls the corresponding handler method to execute the appropriate functionality. This approach ensures that the logic remains responsive and non-blocking, allowing the system to remain reactive to GUI inputs. The timer starts when the user initiates the measurement process and stops when the user presses the pause or stop button, or when there are no remaining sampling points and the system reaches the end state.

3.2.2. Integration with the GUI

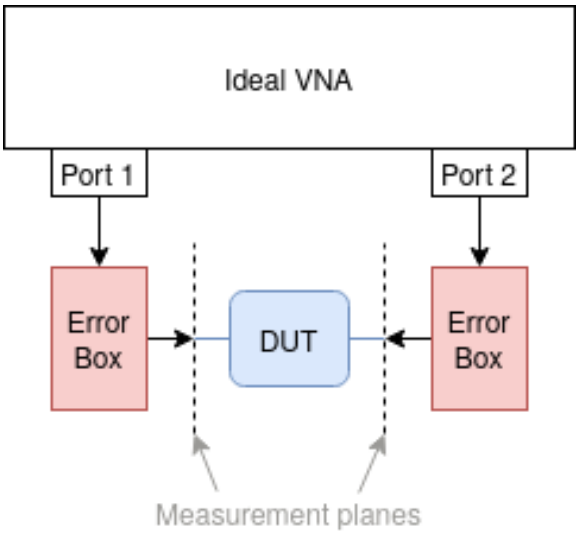
MATLAB offers a design tool to intuitively build an App, as further discussed in Section 3.6. However, due to the characteristics of this tool, integrating it with version control systems like Git is nearly impossible. Therefore, it is more practical to have as much logic and functionality as possible within a separate class, allowing the app to simply call its methods. This approach minimizes changes in the app itself and allows to work primarily in MATLAB files are supported by version control. A drawback of this approach is that the controller class can become quite large, containing many simple wrapping (forwarding) methods, that just call other functions. See listing D.2 in Appendix D for an example of a wrapper function that connects a GUI call to the method from the robot object.

3.3. Vector Network Analyser

A Vector Network Analyser (VNA) is a device that generates a known stimulus signal and measures changes to this stimulus signal caused by the device under test (DUT)[16]. The data obtained from VNA measurements are called scattering parameters (S-parameters). Often in electrical engineering, voltages and currents are the quantities of interest, but at high frequencies it is convenient to consider these quantities as waves instead[17], [18]. Certain reflection and transmission coefficients of these travelling waves are measured by the VNA as the S-parameters, which can then be related to measurements such as gain, loss, and reflection coefficients[16]. Figure 3.3a introduces how a VNA measures the S-parameters and sets up an intuition for what each S-parameter measures.



(a) Explaining S-parameters. *Diagram reproduced from [16] © Tektronix. All rights reserved.*



(b) Error boxes model for explaining VNA Calibration.

Figure 3.3: Explaining S-parameters and the error box model

3.3.1. Calibrating the VNA

A vector network analyser is an advanced device that can take accurate measurements of scattering parameters. However, it can only measure the s-parameters at the input of its two ports. Consequently, the characteristics of the coaxial cables and devices connected to these ports will also be measured,

as the measurement plane remains at the ports of the VNA. This is often an undesired effect, for which calibration is introduced to compensate. Calibration is the process of using well characterised standards to remove deviations from the ideal characteristics of the measurement setup[19]. These deviations are systematic errors produced by the length of cables and connections between the VNA ports and the DUT.

There are two types of two-port VNA calibration methods that will be discussed: Short-Open-Load-Thru (SOLT) and Thru-Reflect-Load (TRL). Figure 3.3b illustrates the “error box” model for visualising VNA calibration. The role of these error boxes is to move the measurement planes from the ports of the VNA, to the inputs of the DUT by removing the “error” caused by the measurement equipment between these two points. Both calibration techniques aim to find parameters for these error boxes by using well characterised standards to transform the measured s-parameters to the expected results of these well-defined standards.

SOLT calibration is performed by connecting a short, open, and load in turn to both ports and saving the measurements. Lastly, the ports are connected using a through connection. These standards are often bought together in a calibration kit. These seven measurements are sufficient to calculate the error terms for the two error boxes. The SOLT can be performed in the lab using the Calibration Wizard on the VNA; thus, the mathematics behind determining the error terms need not be known. SOLT is an excellent calibration method for high bandwidth measurements, as the calibration is valid from arbitrarily low frequencies up to the frequency limit of the calibration kit. Unfortunately, SOLT requires the use of a “perfect load”, which doesn’t exist in actuality, so the accuracy of the calibration is limited by the accuracy of the load in the calibration kit[19].

TRL calibration does not suffer from this limitation[20]. Instead of a load it uses only a through, a total reflection (often a short), and a high accuracy quarter wavelength line. A quarter wavelength line is one which is a quarter wavelength longer than its length at a certain centre frequency f_c, and the accuracy of the calibration depends on the quality of this line. Such high-quality lines can lead to calibrations that are more accurate than SOLT calibrations (≈ 10 dB difference in residual errors after calibration[19]). However, TRL calibrations have a limited band on which the calibration is valid; the frequency range is valid when the line is 20° to 160° longer than the through. For this project, the frequency band of interest is 26.5 to 40 GHz, so taking $f_c = 33.25$ as the halfway point and using equations 3.1a and 3.1b found from [19] means a valid calibrated frequency band between 7.4 to 59.1 GHz, thus suitable. For this project, it was chosen to use the SOLT calibration, mainly because that calibration kit was readily available and the movement of the robot would likely negate the benefits of a highly accurate calibration achieved through TRL.

$$f_1 = \frac{4 \cdot 20 \cdot f_c}{360} \tag{3.1a}$$

$$f_2 = \frac{4 \cdot 160 \cdot f_c}{360} \tag{3.1b}$$

3.3.2. Interfacing with the VNA

As mentioned in table 2.2 in the Programme of Requirements, the Terahertz Sensing group can already measure and set parameters of the VNA from within MATLAB. Given example code that has been used previously (listing D.11), subgroup 2 was tasked with modifying it to be compatible with the robot arm measurement system. This section will discuss the improvements made and justify these decisions.

The first mandatory requirement for the VNA is connecting to the VNA from within MATLAB (Requirement 6). This functionality has been implemented in the example code using the VISA communication API, which facilitates bus-independent development of programs that need to communicate with devices with various interfaces[21]. In other words, the programmer can connect to various measurement devices using VISA, without needing to worry about the specific communication interface (USB, GPIB, Ethernet, for example). Furthermore, VISA has been partly developed by Keysight[22], the manufac-

turer of the specific VNA that will be used for this project. Consequently, it can be safely decided that the VISA communication API is the most suitable and justifies why subgroup 2 will build upon the example code to modify it for their specific purpose. Although less relevant for this thesis, it is worthwhile to note that MATLAB's implementation of VISA is not supported on the Linux operating system; Linux distributions, especially Ubuntu, are the de facto industry standard in the field of robotics due to their compatibility with the Robot Operating System (ROS) architecture[23]. If the Terahertz Sensing Group wishes to perform more advanced robotic measurements in future, they may be required to implement other communication protocols like GPIB instead to facilitate the use of ROS on Linux. However, for this project the Windows 10 operating system is used, and thus this mandatory requirement is satisfied.

Having justified the use of the VISA API to connect to the VNA, the system must allow the user to enter the address of the VNA to fulfil requirement 5. The example code provides the user with some basic functionality to enter a GPIB address, but the implementation was not user-friendly. Firstly, it only allows for GPIB addresses to be connected to, meaning other VNA's which may use USB or LAN connections cannot be used. Secondly, the input field to select an address were multiple counters which needed to be set to the correct number, which was time-consuming and error-prone. Instead, the solution implemented was to first search for all the available VISA addresses, and then display these in a dropdown menu to choose from. This removes the possibility of incorrect address inputs and provides a clear visual overview. Figure A.1 in Appendix A shows a screenshot of how the dropdown menu displays the available VISA Addresses. This modification drastically simplified selecting an address, but it doesn't fulfil the requirement completely. The requirement states that the user must be able to enter an address, instead of selecting it. A text input field was therefore added with the option to select the manually entered address. A text input field was chosen over the old counter method, as it allowed the address to be copy-pasted into the GUI, which is quicker and less error-prone. This text input fulfils the requirement completely, and the dropdown was kept as a convenience. Section 3.6 discusses the MATLAB App designer, which was used to provide the dropdown and text input functionalities mentioned.

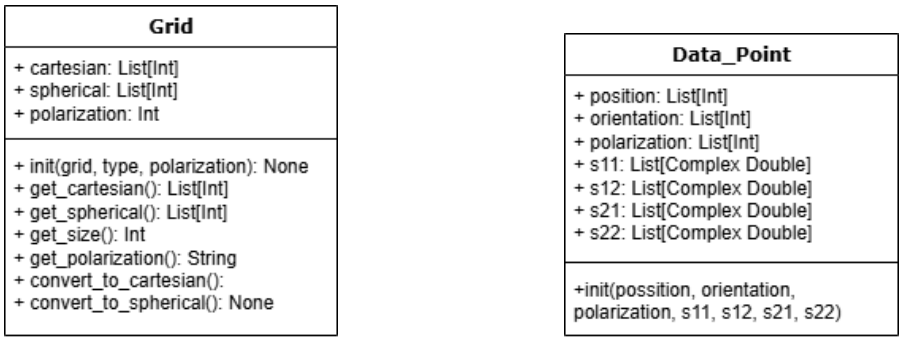
The next phase is setting and retrieving specific parameters as dictated by requirements 7 and 8. The official programmer guides[24] from Keysight (formerly named Agilent Technologies) require the use of the SCPI command set, which is a standard for syntax and commands often used with measuring devices like the VNA. Listing D.3 in Appendix D shows example MATLAB code to use the SCPI commands. The example code provided the read and write functionalities required to fulfil these requirements, so they have been integrated directly into the GUI. Figure 4.1b shows a screenshot of the four text input fields that show the current VNA parameters, and give the possibility to input new values for the parameters. It is important to note the unit conversions done here, the VNA requires and returns all frequencies measured in Hertz, whereas on the GUI, specific parameters have units of GHz, MHz, and kHz. With the functionalities of the example code integrated into the GUI, these two requirements are satisfied. To test these requirements, the values retrieved and set can be compared to the values presented on the VNA itself to ensure they coincide.

Lastly, requirement 9 states that the MATLAB code must be able to start a measurement and read the data from the VNA. The general functionality is implemented in the example code, but it has been modified to fit the specific needs of this project. In the previous code, there are certain setup steps which need to be performed before the first measurement can be read; the required data first needs to be defined independent of taking a measurement. However, it was chosen to modify it such that the definition and measurement are done together. This is because the measurement process could be paused and restarted throughout the process, which raises the possibility of losing the definitions. Thus, measurements cannot be taken, and the process is interrupted. Defining them for every measurement increases the reliability of the measurements. To fulfil the requirement, the SCPI commands are adapted from the example code and integrated into the state machine for the measurement process. Section 3.2.1 elaborates on this state machine. Lastly, the data returned from the VNA are comma-separated values of interleaved real and imaginary parts, which are processed to be stored properly as MATLAB's complex-double data type. Plotting the magnitude and phase data received in MATLAB and comparing it to the output screen on the VNA proves that this requirement is fulfilled. Listing D.5 in Appendix D the MATLAB implementation that for reading a certain s-parameter from the

VNA.

3.4. Data Handler

During the measurement process the system needs to store all relevant data, to allow for later analysis and processing of the results. In accordance with Requirement 10, the system is required to store not only the measurement data itself, but also the settings and parameters used during the measurement. Additionally, a system must be in place to keep track of all data in different formats, such as spherical and Cartesian coordinates, as well as to record which sampling points have already been measured.



(a) Class diagram: Grid

(b) Class diagram: Data_Point

Figure 3.4: Class diagrams: handling data

To manage the sampling process effectively, a dedicated data structure is used to keep track of the measurement points. The `Data_Handler` class is responsible for various functions, including determining where to save the data files, tracking the progress of the measurement process, keeping record of the current measurement point, and handling the actual saving of data. Separating this functionality into a dedicated class helps keep the codebase organized and consistent.

The system must be capable of handling both Cartesian and spherical coordinate systems. Therefore, a dedicated data class, `Grid` as seen in figure 3.4a, is used to store the coordinates of all sampling points and provides functionality to convert between Cartesian and spherical representations.

To link the measurements from the VNA to the coordinates of each point, a data class `Data_Point` is used, as seen in figure 3.4b. This class doesn't have any functionality, but just stores all the data related to one measurement point. Keeping the data in a separate class like this makes the code easier to manage and consistent.

3.5. Post-processor

The post-processor class designed by subgroup 2 focuses on the plotting and analysis of the raw and time-gated frequency and time domain data. It differs from the post-processing done by subgroups 3 and 4, as their focus lies with the frequency analysis and directivity plots. The purpose of this post-processing module is to provide the user with an intuitive user interface to display results and time-gating functionality. This section will discuss the design decisions made to provide this functionality, starting with plotting the raw data and later explaining and implementing the time-gating.

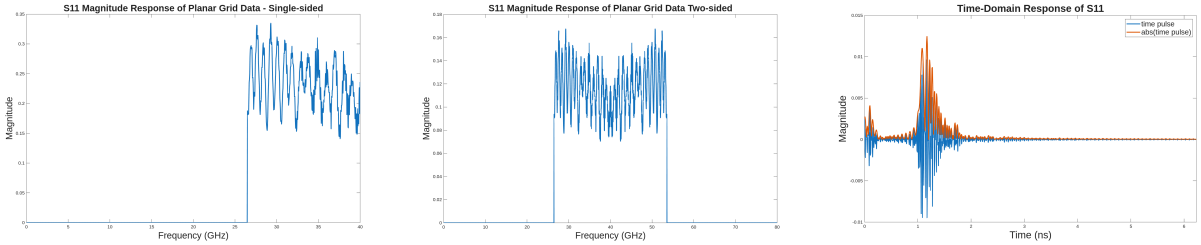
3.5.1. Plotting Raw Data

During the measurement process, a large volume of data is collected; for each point, there are four complex arrays of s-parameters. To fulfil requirements 11 and 12, the user must be able to choose which point and specific S-parameter to see the results of. Since the post-processing is independent of the point or s-parameter measured, simple dropdown menus for selecting the required point and s-parameter suffice to deliver the relevant result. This fulfils the requirements completely, and their functionality can be tested by measuring two distinctly different points and S-parameters, and then comparing the raw-data result plots to the signals plotted on the screen of the VNA.

Once the required data point and s-parameter are selected, requirement 13 states that the raw frequency response should be plotted. Since the S-parameters are measurements in the frequency-domain, plotting this is trivial in MATLAB and only the frequency axis needs to be calculated. Listing D.6 in Appendix D shows the required code to produce plots with the frequency axis in GHz, magnitude response in decibels and the unwrapped phase (removing jumps of 2π radians to give a continuous phase) shown in degrees as specified by the sub-requirements. It also shows how the plots can be integrated into the GUI by passing in the `mag_axis` and `phase_axis` GUI handles as arguments to MATLAB's `plot` function.

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{i2\pi kn/N} \tag{3.2}$$

Requirement 14 states that the time-domain response must also be plotted as part of the results. Recall that the frequency-domain and time-domain representations of a signal are related by the inverse Fourier transform[25]; when discretised in both time and frequency, this relationship becomes the inverse discrete Fourier transform (IDFT) shown in equation 3.2. By applying the IDFT, the discrete time-domain response can be generated mathematically from the discrete frequency-domain S-parameters. Applying the IDFT in MATLAB is done via the inverse Fast-Fourier transform algorithm, implemented using the built-in `ifft` function. Listing D.7 in Appendix D presents the code required to correctly implement the IDFT for an array of s-parameters.



(a) Step 1: setting lower frequencies to zero.

(b) Step 2: flipping and conjugating to produce a two-sided spectrum.

(c) Step 3: Output of the zero-padded IFFT in the time domain.

Figure 3.5: Applying the inverse discrete Fourier transform. *Magnitude not plotted in dB as to express zero values*

There are three steps that need to be taken before the `ifft` function[10] should be called. For each step indicated in the code, figure 3.5 presents the corresponding spectrum to give a visual summary of the process. Firstly, the s-parameters are measurements in a specific frequency band determined by the VNA start and stop frequencies, and need to be formatted to the one-sided spectrum from 0 Hz DC to the stop frequency. This can be done by zero padding down to 0 Hz, using the same frequency intervals as used by the VNA. It is important to note here that the sudden change between zero and the signal will produce a discontinuity, of which the effects are discussed later in this section.

Next, to produce a purely real time-domain response after the IFFT the input must be a Hermitian symmetric spectrum. This is a fundamental property of the Fourier transform[25], [26] and is mathematically defined as presented in equation 3.3. Due to the periodic nature of the frequency spectrum produced by a discrete Fourier transform, the one-sided spectrum can be flipped horizontally, conjugated and then appended onto the end as shown in the code and figure 3.5b to produce the two-sided spectrum. Additionally, since the IDFT is a summation over the entire spectrum, the peak amplitude must be halved to maintain the same power magnitude over the two-sided spectrum.

$$X[k] = \overline{X[N-k]} \tag{3.3}$$

lastly, the spectrum is zero-padded at the end to a length equal to the next greatest power of 2. Zero-padding yields the advantage that is doesn't change the original signal content, but that it increases

the resolution of the time domain signal. But more notably for the ifft algorithm, it is much more efficient when applied to a spectrum with the length equal to a power of 2[26]. Once these steps are completed, the output from the `ifft` can be seen to be purely real in figure 3.5c as required. The time-domain spectrum will largely be used for time-gating, in which only the peaks in time are relevant. As such, the absolute value of the time signal will be plotted to accentuate the peaks.

3.5.2. Time-gating

Now that the time-domain response can be plotted, the time-gating functionality should be provided to the user as mandated by requirements 15, 16 and 17. First, an explanation of the process of time gating will be given, followed by how it has been implemented to prove that the relevant requirements are fulfilled.

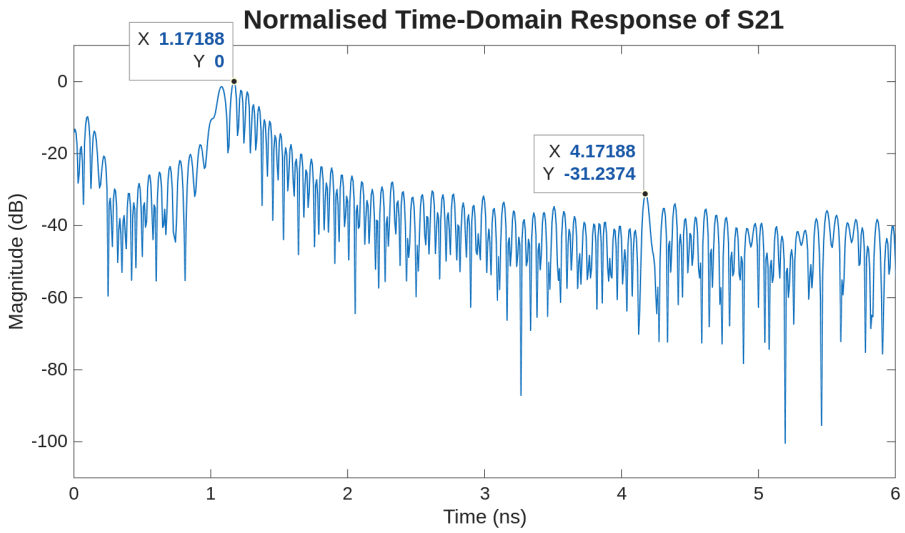


Figure 3.6: Normalised time-domain response of the S11 parameters measured 45 cm above a reflecting surface.

The time spectra show instances in time for which the signal is reflected or transmitted, depending on which s-parameter is plotted. Using S-11 as an example, figure 3.6 shows the reflections measured in time. It can be seen that there are two peaks, the first represents the reflection of the probe, the second is the reflection of the table it is above. This can be validated by calculating the travelled distance by an EM wave: $\Delta t = 4.17 - 1.17 = 3.0$ ns $\implies d = (3.0 \times 10^{-9})c = 90$ cm for the incident + reflected wave distance travelled, coinciding with the physical setup. If this reflection is unwanted, time-gating is a possible solution; time-gating is the process of applying a time filter over the response of interest, thus removing the effects of unwanted responses (likely reflections of the environment[27]) outside the interval of the filter.

To perform time gating, the start time of the gate and the size of the gate need to be defined. For this project, the user must define these as stated by requirements 15 and 16. To provide this functionality, the user interface provides GUI elements to input the start and size parameters for the gate, as shown in figure 4.2a and the red lines drawn onto the time plot show the gate boundaries. This allows the user to select the interesting time-window with the greatest accuracy, to extract the best results from the process. Applying time-gating in the time-domain is trivial, and listing D.8 in Appendix D presents the algorithm used. The time-domain results are shown in figure 4.2b with the window selected and everything else zero, which proves to the user that their gate parameters have been used. This satisfies these two requirements.

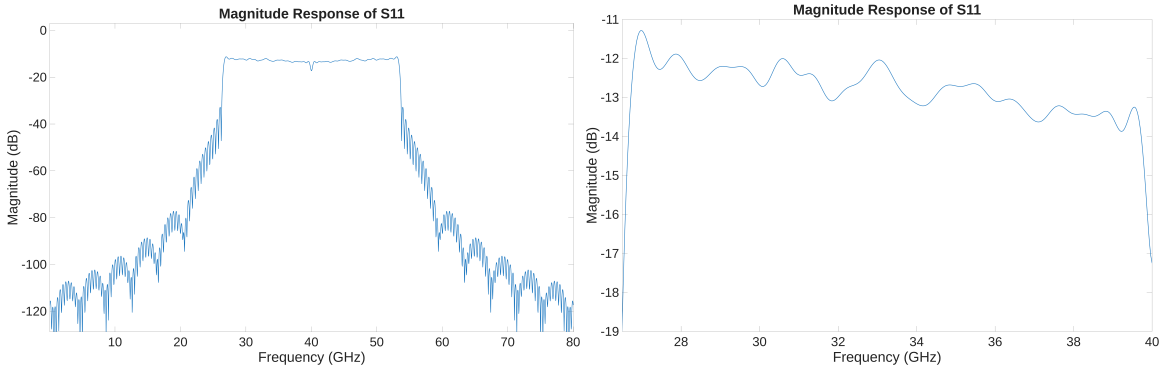
The next phase is returning to the frequency domain after time-gating is performed, as stated by requirement 17. This can be done through the discrete Fourier transform (DFT), which is the inverse operation of the IDFT applied previously. Thus, returning to the frequency domain can be done simply by applying the `fft` function to the purely real time spectrum, and zero padding to the next power of two. As expected, this produces a two-sided spectrum in the frequency domain, which has half the

peak amplitude of the one-sided spectrum. Consequently, multiplication by two and focusing on only the relevant frequency spectrum (for this project, 26.5 to 40 GHz) produces the required output. listing D.9 in Appendix D shows the MATLAB implementation, and figure 4.2b shows the magnitude response in decibels, unwrapped phase response in degrees and frequency-axis in GHz as required.

3.5.3. Effect of Discontinuities

As mentioned previously, when converting the measured s-parameter spectrum to the one-sided spectrum shown in figure 3.5a, there exists a sudden “jump” (or discontinuity) between zero and the first measured s-parameter. This discontinuity is also present in the two-sided spectrum. When the IDFT is applied, this jump produces large oscillation peaks at the discontinuities, known as the Gibbs Phenomenon. This exhibits itself as a peak at the start and end of the time spectrum. Fortunately, This effect is not significant in this specific application, as the edge peaks can simply be ignored or removed through time-gating to focus only on the true peaks of reflection/transmission shown in the data. As such, reducing the discontinuities (and thus the Gibbs peaks) by applying windowing methods, like Hamming windows, is not necessary.

On the other hand, time-gating again produces discontinuities which cannot be ignored. Time-gating is effectively multiplying the time-domain spectrum by a rectangular window, which in the frequency domain is equivalent to a convolution with a Sinc function. Consequently, Sinc-like ripples can be observed in the two-sided frequency spectrum produced when the DFT is applied. This can be seen in figure 3.7a. The effect of these Sinc ripples can be observed as a slight dropoff in the frequency response at the edges of the frequency band of interest, shown in figure 3.7b. This is undesired, so reducing this effect by smoothing out the discontinuities is beneficial, and can be accomplished by applying windowing[25]. A Tukey window is a simple window made of part of a cosine function at the discontinuities and a flat pass band in the middle, shown in figure 3.8. The parameter α determines what proportion of the window is a cosine function versus a flat pass band. Listing D.10 in Appendix D shows the MATLAB implementation for applying the Tukey window, and its effect can be seen in figure 4.2b where the dropoff is reduced.



(a) Frequency domain two-sided spectrum showing dropoff due to the delta functions.

(b) The frequency spectrum of interest showing the slight dropoff at the edges.

Figure 3.7: Example frequency spectrum showing the effect of delta functions.

3.6. Frontend - MATLAB Graphical User Interface

In order for the user to operate the system is a user-friendly manner, a graphical user interface (GUI) is designed. The GUI must meet several requirements discussed in Chapter 2 to allow users to operate the robotic arm and the vector network analyser without requiring high-level training. It should also enable the user to perform antenna measurements while clearly presenting both the progress and results of the test.

Inspired by previous work done by the THz group, the GUI is divided into multiple tabs, separating functionalities to maintain clarity. This section will briefly discuss the functionality of each tab. A more in-depth implementation will be discussed in Chapter 4.

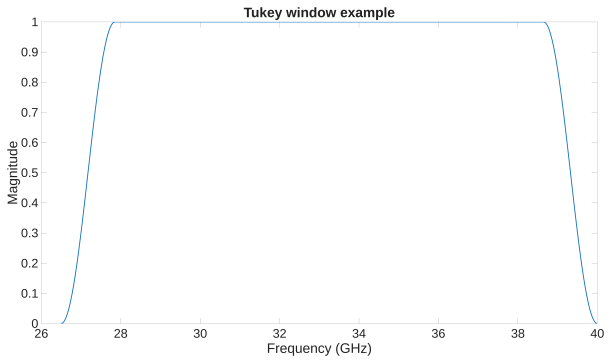


Figure 3.8: Example Tukey window with $\alpha = 0.2$.

Movement This tab allows the user to manually control the robot arm. Moving it manually in all directions in steps and sending it to specified potions, meeting Requirement 2 and 3. This tab also includes any settings that can be pushed to the robot, such as setting the speed of the linear movement.

VNA The VNA tab provides control over key parameters such as the start and stop frequencies, IF bandwidth, and the number of frequency points, fulfilling Requirement 7 and 8. In addition, this tab includes an interface to establish a connection with the VNA where the user can enter or select the correct address of the VNA, meeting Requirement 5 and supporting Requirement 6.

Antenna test This tab includes all the information and settings needed to conduct an antenna measurement. The user can select the type of test, Near Field or Far Field, and configure antenna parameters such as aperture size and operating frequency. When selecting the test parameters, the GUI previews a plot of all the sampling points. The Antenna tab also provides information about the measurement progress, such as the remaining time and a visual overview indicating which sampling points have been measured and which are still remaining, meeting Requirement 18.

Results This tab presents the measurement data in plots, such as magnitude and phase. Users can select sampling points from the current measurement or choose to view the S-parameters of previous measurements by uploading the corresponding files, meeting Requirement 11. Additionally, the user has the option to apply time gating to the selected sampling points, with a visualisation of the time-gated signal, fulfilling Requirements 15 - 17.

3.6.1. MATLAB App Designer

The GUI is designed and programmed using MATLAB App Designer, which provides an intuitive interface for building and customizing graphical user interfaces within the MATLAB environment, supporting Requirement 22. The interface layout is constructed using the basic elements provided by the tool, with the visual component code automatically generated. All functional behaviour, such as callbacks and logic handling, is implemented manually.

4

Control Interface Implementation and Validation

This chapter discusses the process of implementing the relevant algorithms and structures developed in the design to produce the prototype of the control interface for the antenna measurement system. Since subgroup 2's focus lies with integrating all hardware and software components together into a centralised control interface, this section will cover almost all aspects of the BAP project. It will begin by focusing on the implementation of specific systems, outlining the difficulties and successes experienced. For each system, the results of the implementation will be presented my means of screenshots of the GUI or photos of the specific setup. Next, the results will be validated in terms of performance and how well they conform to the programme of requirements.

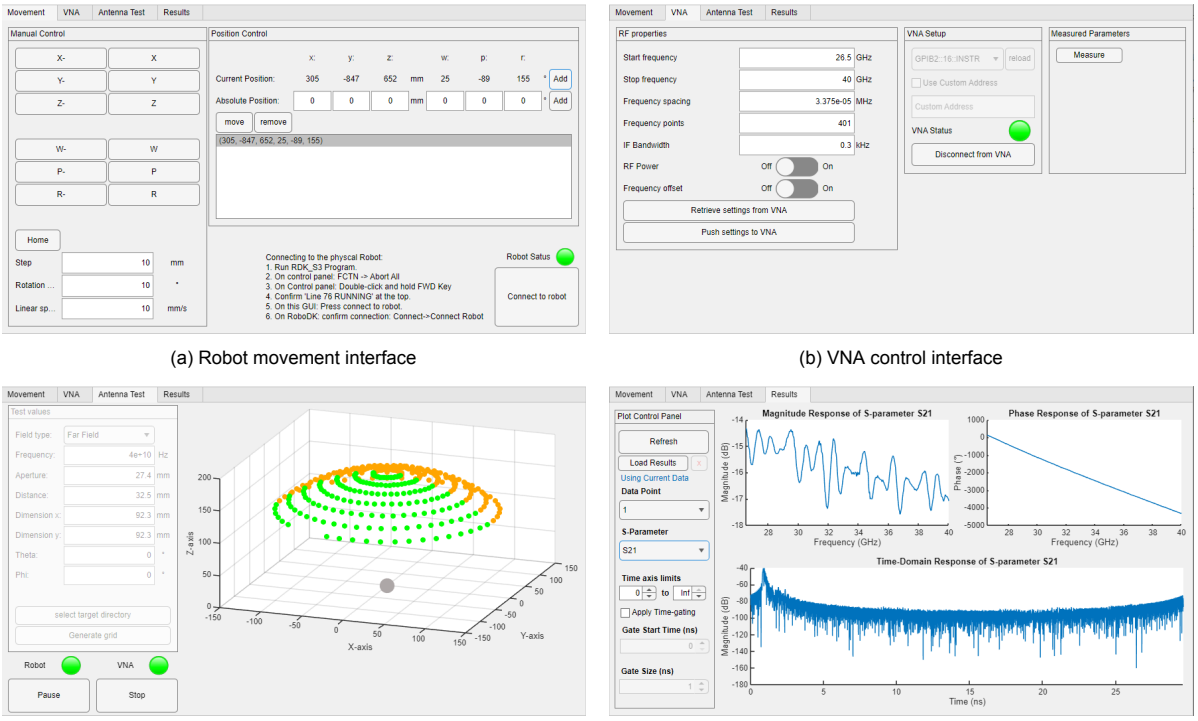
4.1. MATLAB graphical user control interface

To promote modularity and keep the GUI focused on its core visual role, the GUI script was intentionally kept lean, containing primarily user interface components. All significant functionality resides in external modules like the controller, which users can access via the GUI. As a result, the GUI script excels as a visual presentation layer and is exceptionally stable, which is beneficial when the final controller is exported as a standalone executable to be used by the Terahertz Sensing Group, or when sold as a product. Before being exported as an executable, though, the GUI still needs to be polished to enhance the user experience. It is currently in the prototype stage, meaning helpful debugging elements like logs and extra buttons are present that can clutter the experience. The prototype version of the GUI and all its tabs can be seen in figure 4.1. After the completion of this thesis, the GUI will be optimised by removing the extra features to provide a streamlined user interface. Furthermore, the TU Delft and Terahertz Sensing Group logos will be added, as well as the author names.

4.2. Movement tab

The first tab on the control interface is the robot arm movement tab. This tab primarily interfaces with RoboDK and subgroup 1's robot movement script. The communication with subgroup 1 was frequent and instructive, so the integration of their code into the controller was effective.

Before movement can be done, the GUI must be able to connect to the robot. The movement tab provides the user with a brief tutorial on how to connect to the robot, shown in figure 4.1a, and a `Connect to robot` button which can be pressed. When it's pressed by the user, the GUI attempts to connect to the robot via RoboDK. RoboDK is a widely used robot motion simulation software, that subgroup 1 has decided to use to control the robot's movement. Should the connection to the robot fail, a red status indicator is shown on the GUI and the user should follow the tutorial again. When the robot is disconnected, the interactive GUI elements for moving the robot are disabled to remove the possibility of an error being thrown if the user tries to move an unconnected robot. Should the connection succeed, the status indicator turns green, and the movement panels are enabled for the user.



(a) Robot movement interface

(b) VNA control interface

(c) Measurement execution screen

(d) Measurement results interface

Figure 4.1: Overview of the graphical user interfaces.

Next, The movement tab provides input fields for the user to input linear and rotational step sizes, as well as linear speed. Additionally, buttons to move the robot one step in a certain direction are also available to the user. When these are pressed, the relevant step size and speed are used to calculate an increment, which can be passed to the robot script to move the robot. Together, this fulfils requirement 2 as the user can move the robot manually through inputs on the MATLAB GUI. Continuing, this tab also gives the option for the user to save a certain robot arm position. This position can be selected and moved to when the user presses the move button; the saved robot arm positions are passed as a new movement target to the robot script, which moves the robot through RoboDK. Thus, requirement 3 is satisfied.

One issue that arose during testing is that the robot's linear speed could not be selected by the user through MATLAB. Although this isn't a mandatory requirement, it is a useful parameter for the user to control to improve the safety of them and the robot. It was noticed that the selected speed in the GUI had no effect on the robot. This was due to the controller tablet (included with the Fanuc robot) overwriting the user-entered linear speed value, and unfortunately, no solution was found.

4.3. VNA tab

This tab is dedicated to enabling the user to interface with the VNA. It handles the connection and setup of the VNA parameters that the user must be able to control, as defined in requirements 8 and 7. Throughout the implementation process, the VNA tab proved exceptionally reliable when connecting and adjusting VNA settings.

As discussed in depth in the design section (3.3.2), the process of connecting to the VNA has been streamlined compared to the example code. Figure A.1 shows the dropdown menu for selecting accessible addresses, and figure 4.1b shows the text input field provided to the user. Like when connecting to the robot, a status indicator is used to show if a connection to the VNA has been established. Additionally, when the VNA is disconnected, the user input fields are disabled to prevent the user from making changes to an unconnected VNA.

Once connected, the user can input their desired VNA settings. The options given to the user are the ones mandated by requirements 8 and 7: start frequency, stop frequency, the number of frequency points and the intermediate frequency bandwidth. The option for frequency spacing is also given to the user for convenience, but isn't directly set on the VNA. This is because the number of frequency points and the frequency spacing are related for a known bandwidth. When the user inputs their desired spacing, the corresponding number of points is calculated and this is the value subsequently set on the VNA. The opposite is true for retrieving the frequency spacing from the VNA; the number of points is retrieved, and the corresponding frequency spacing is calculated and presented to the user.

Next, when measuring values from the VNA using the algorithm in section 3.3.2, an important bug was found. The VNA was measuring continuously throughout the robotic antenna measurement; which introduced the possibility that VNA measurements were taken during the robot's motion between points, instead of strictly at points. This produced unreliable s-parameter results and sometimes discontinuities in the directivity plots made by subgroups 3 and 4. To solve this issue, the VNA needs to be set to single mode using the command shown in listing D.4 in Appendix D, and then repeat the measurements.

4.4. Antenna test tab

This tab controls the robotic antenna measurement process. Consequently, it interfaces with all the other subgroups and all the hardware, so it presented quite a challenge to implement. Efficient communication with the other subgroups, as well as reliable interfacing with the hardware, accelerated the development of this tab. Fortunately, this tab was able to rely heavily on the modular back-end control system that will be discussed in section 4.6.

Figure 4.1c shows the measurement tab, and the user input fields available on the left side. Here, the user can enter all the parameters that need to be stored as stated by requirement 10. These are the relevant antenna parameters that determine the near- and far-field grids to be measured and influence the post-processing done by subgroups 3 and 4. A dropdown menu allows for the selection of near- or far-field measurements. Additionally, two status indicators for the VNA and robot connection are included to ensure the user only starts a measurement process when both are connected. Tradeoff requirement 24 states that the user could also be reminded to perform calibration. This is currently not implemented, but would be an improvement to be considered for future work. The screenshot shows that the inputs fields are disabled when a measurement is running, which prevents the user from making changes and interrupting the process.

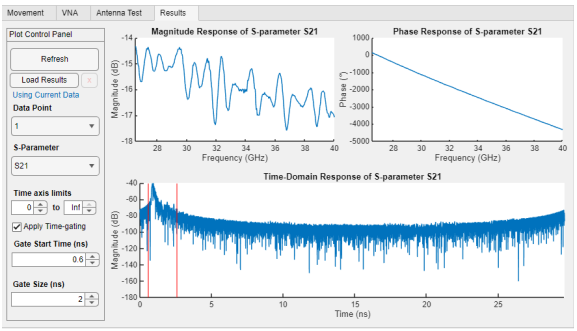
With the user inputs defined, the grid of measurement points needs to be generated for the robot arm. The control interface is integrated with the code from subgroups 3 and 4, which generate the grid of positions for each measurement. This grid is then passed to the subgroup 1's pathfinding code, which is also integrated into the GUI. Finally, the movement path between all the points is returned as an ordered list of points. This list contains all the measurement points, for both co- and cross-polarisation, which are plotted on the antenna test tab GUI. This tab then relies on the back-end controller to run the measurement state machine, and updates the colour-coded plot of measurement points in real time corresponding to the progression of the robot arm; blue points correspond to unmeasured points, orange to points measured at only one polarisation, and green for points measured at both co- and cross-polarisations. This fulfils requirement 18 completely. An extra convenience was provided to the user in the form of an estimated time remaining timer, which calculates the average time taken per point and predicts the remaining time based on the points remaining.

4.5. Results tab

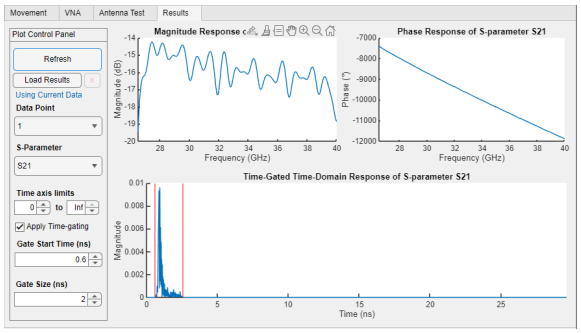
The final tab on the GUI is the results tab, where the user can plot and time-gate the results in real-time during a measurement. Figure 4.1d shows the control panel on the left, on which the user can select which s-parameter and measured point they want to plot from dropdown menus. Furthermore, the dropdown menu for the measured points can be updated by pressing the refresh button. Updating the points' dropdown menu was not done continuously, as this would slow down the measurement process unnecessarily; the user is not expected to want to plot every point exactly when they're measured. This

functionality satisfies mandatory requirements 11 and 12.

For the selected measurement and s-parameter, the results tab plots the magnitude and phase response in the frequency-domain, and the absolute response in the time domain as shown in figure 4.2. These graphs have been implemented using the algorithms discussed in design section 3.5.1 and are automised within the GUI; when the user selects a different measurement or s-parameter, the axes automatically update with the corresponding plots. The user can interact with the graphs, allowing them to magnify specific sections, and they can always return to the original state by pressing the refresh button. Attention has been given to ensuring the plot axes show the correct units, thus plotting the raw results on these axes and integrating it into the GUI satisfies requirements 13 and 14 completely.



(a) User can select the gate using the side panel. The red boundary lines show the resulting window.



(b) The output after the gate is applied. *Magnitude not plotted in dB to express zero values*

Figure 4.2: Time-gating functionality in the GUI.

Additionally, the time-gating functionality has also been implemented using the algorithm as discussed in design section 3.5.2. The user can use the input panel on the left to choose a time gate size and start time, with the GUI confirming the inputs with red boundary lines on the plot. Pressing the refresh button then applies the time gating algorithm and updates the plots. The results of the time-gating can be done simultaneously with measurements, allowing the user to perform time gating on any point in real-time. The results of time gating with a suitable window produces a smoother frequency spectrum with similar amplitude, as expected. Consequently, requirement 17 is fulfilled.

For analysis purposes, it is useful for the user to also be able to load old data into the GUI to plot it. This is not stated as a mandatory requirement, but rather as a trade-off requirement (27). This functionality has been implemented into the GUI and was especially useful for this thesis, as it meant that the code could be tested with old data; the functionality is also beneficial for an end user if they want to quickly time-gate certain old measurements. The loading functionality is achieved through selecting a file from a directory using a pop-up graphical interface, and then loading a `.mat` file with the correct format. The functionality worked reliably, but it is important to note that loading a file clears all memory of the current measurement, so an old datafile should not be loaded during the robotic antenna measurement process.

4.6. Backend control system

The backend control system is what the graphical control interface relies on to perform the measurement process. The validation of the control system can only be done with a test of the entire system, which fortunately has been done multiple times and is working. Furthermore, the control system has also been used to collect the results in chapter 5, further proving the intended functionality.

The state machine provides the core behaviour of the measurement system. It has been implemented as discussed in design section 3.2.1. Each state takes advantage of the highly modular code structure that has been used, which aided the development process substantially. The bugs and errors that occurred in the control system were easy to track to a specific state and thus a specific submodule, and then tested and fixed individually. This led to the stable central control system currently implemented.

4.7. Validating remaining requirements

Since most of the functional requirements are integrated into the GUI and discussed in the previous sections, the remaining requirements to be discussed are the non-functional and trade-off requirements. Firstly, however, there is one mandatory functional requirement that must be discussed briefly. Requirement 19 states that the final directivity plot needs to be shown to the user at the end. This is currently not integrated into the GUI, but instead the figure is produced by subgroups 3 and 4 and presented in a separate plotting window. Integrating this functionality into the GUI would be straightforward and is an improvement for future work.

Next, trade off requirements 28 and 29 are currently not implemented, since the code may change after the submission of this thesis. Updating the `Readme.md` in the repository will be done before the final defence presentation to fulfil these requirements.

Lastly, three trivial requirements that are fulfilled will be listed: requirements 20 and 21 state the minimum and maximum operational frequencies for this project. Using the GUI, these are entered into the input fields and have been measured for the entirety of the project. Next, requirement 22 states that everything must be implemented in MATLAB. Since the GUI is designed with the MATLAB App Designer, and all the code is designed to be written in MATLAB, this requirement is fulfilled.

5

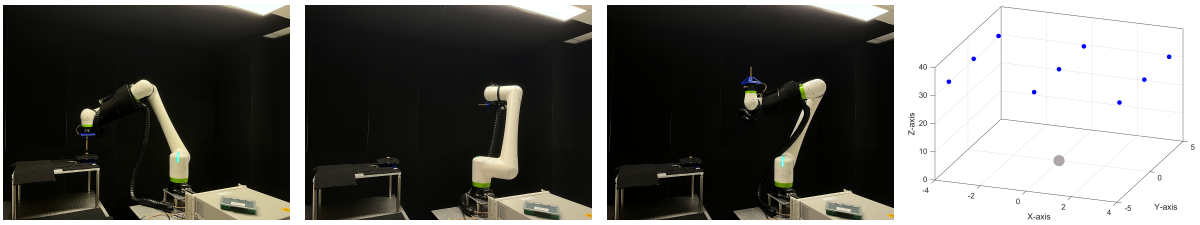
Robot Motion Effects on Field Measurements

This chapter investigates potential error sources within the antenna measurement system. More specifically, it will be examined how the motion of a robot arm affects the phase of the measurements. The findings presented represent a significant contribution to the field of robotic antenna measurements and are expected to be highly beneficial for the Terahertz Sensing Group.

5.1. Repeatability

Repeatability refers to the ability of a test or measurement process to produce consistent results when repeated under the same conditions. The robot arm introduces a repeatability error of ±0.04 mm [28], which is significantly small compared to the wavelength of about 7.5 mm at 40 GHz. This means the positional error caused by the robot arm is unlikely to significantly impact measurement accuracy. However, during testing, it was noticed that the cable does not always return to exactly the same position for repeated measurements. This investigation aims to determine if this produces variability in the results.

5.1.1. Setup



(a) Above antenna (b) Between test 2 and 3 (c) Between test 4 and 5 (d) Sampling points

Figure 5.1: Repeatability test setup

To evaluate the repeatability of the system, a few measurements were conducted according to the following steps:

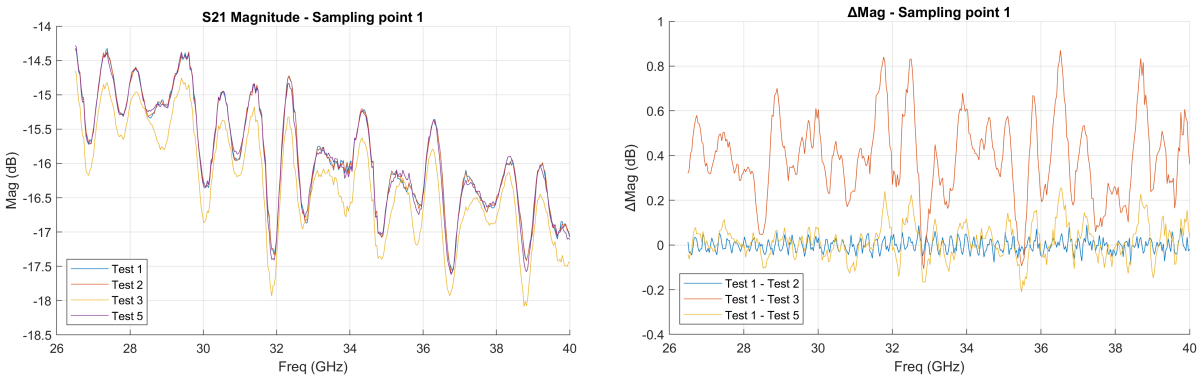
1. **Starting position** above the antenna as seen in figure 5.1a.
2. **Two consecutive tests** (1 and 2) were conducted where the robot arm remained stationary between the measurements.
3. **Random manual movement of the robot** as seen in figure 5.1b
4. **Two consecutive tests** (3 and 4) were conducted where the robot arm remained stationary between the measurements.
5. **Random manual movement of the robot** as seen in figure 5.1c

6. **Two consecutive tests** (5 and 6) were conducted where the robot arm remained stationary between the measurements.

Comparing the results between Tests 1 and 2 reveals the baseline repeatability without movement. A comparison of Tests 3 and 5 with Test 1 highlights the impact of the robot arm's repositioning on the consistency of the measurements. The test consisted of nine sampling points, all measured in the co-polarisation configuration, as shown in figure 5.1d.

5.1.2. Results

In this section, only one sampling point will be discussed, as the results are similar across all points. For the complete set of plots for all 9 sampling points, please refer to figures C.1–C.4 in Appendix C.

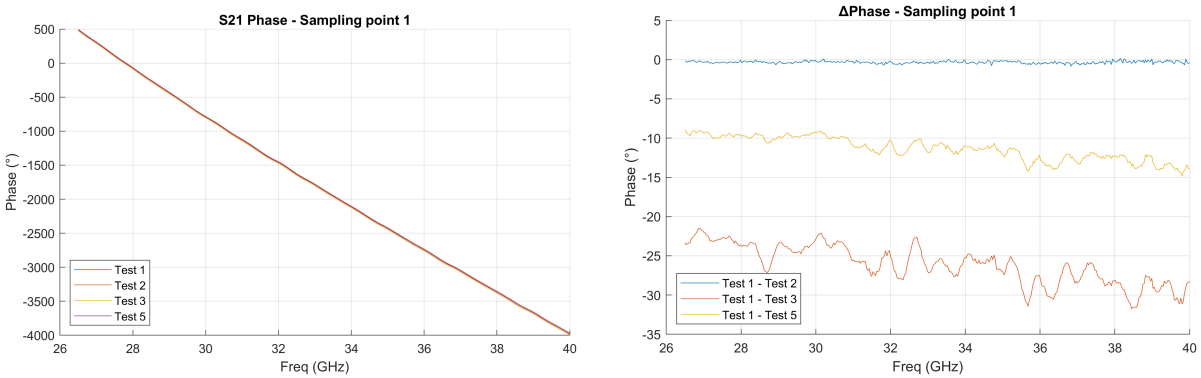


(a) Magnitude (b) Magnitude difference

Figure 5.2: Magnitude Repeatability test at point 1.

The test results clearly show a difference between consecutive measurements and those taken after significant movement of the robot. Figure 5.2a displays the magnitude responses of Tests 1, 2, 3, and 5. As shown in the plot, Tests 1 and 2 show very similar results. Tests 3 and 4, and Tests 5 and 6 also showed similar results and are excluded from the plot for better visualisation.

Figure 5.2b illustrates the differences in magnitude relative to Test 1, with a maximum difference of 0.8 dB between test 1 and 3. Interestingly, the deviation of Test 5 from Test 1 is smaller than that of Test 3, despite the robot having been manually repositioned twice by the time of Test 5.



(a) Phase (b) Phase difference

Figure 5.3: Phase Repeatability test at point 1.

Similar results are observed in the phase measurements. Figure 5.3a shows that all tests follow a similar phase trend. However, figure 5.3b clearly reveals differences in phase between the tests, with

deviations reaching up to 30°, although consistent over the whole frequency range. Similar to the magnitude differences, the phase difference between Tests 1 and 3 is again larger than that between Tests 1 and 5.

5.1.3. Discussion

A clear difference is observed between measurements taken consecutively without movement and those taken after repositioning the robot arm. With a maximum magnitude difference of 0.8 dB and a phase deviation of up to 30°, there is a correlation between movement of the arm and variations in the measured results.

To improve reliability, an extra step may be added to the measurement process. The robot can move to its first position and give the user the option to perform a calibration at that location. This does not necessarily improve the repeatability of the robot, but it ensures that the VNA is optimally calibrated for each measurement, thus improving the consistency of the results

5.2. Settle time

A condition known as cable hysteresis happens when a cable gets bent or moved or does not precisely return to its initial position due to internal friction, mechanical stress, or imperfect elasticity in the cable.

Even though the cable does not return exactly to its original position after moving, effecting the repeatability as discussed in Section 5.1, over some time the cable settles closer to its original position. Therefore, a test with a settling time in between moving the robot and measuring the s-parameters is conducted.

5.2.1. Setup

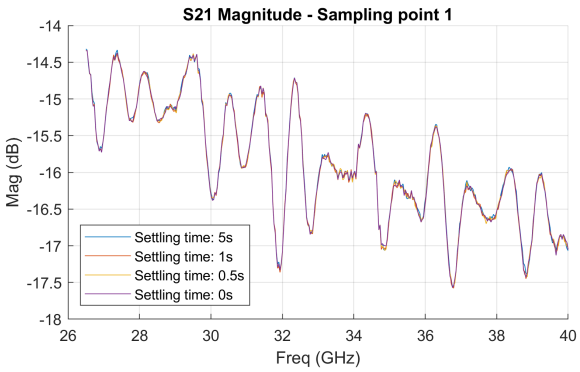
To evaluate the effect of settling time in between the movement of the robot and the measurement of a point, the following measurement sequence was conducted:

1. Settle time of 0 seconds
2. Settle time of 0.5 seconds
3. Settle time of 1 seconds
4. Settle time of 1.5 seconds
5. Settle time of 5 seconds

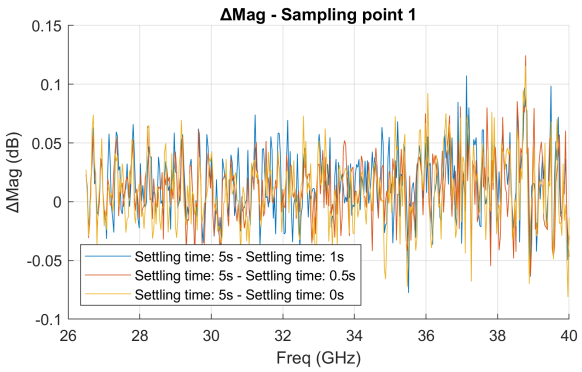
In all cases, the robot arm moved to the same sampling points, only the time between movements and measurements was varied, which was done by adding a line `pause(5);` to the state machine. The test consisted of the same nine sampling points as in the repeatability test, all measured in the co-polarisation configuration, as shown in figure 5.1d. Since the test with the longest settling time is expected to be the most stable, the test with 5 seconds settle time will be used as the reference against which the other measurements are compared.

5.2.2. Results

In this section, only two sampling points will be discussed, as the results are similar across all points. For the complete set of plots for all 9 sampling points, please refer to figures C.5, C.6, C.7 and C.8 in Appendix C. Not all measurements are shown in the plots to maintain readability, but the data follows the same trend.



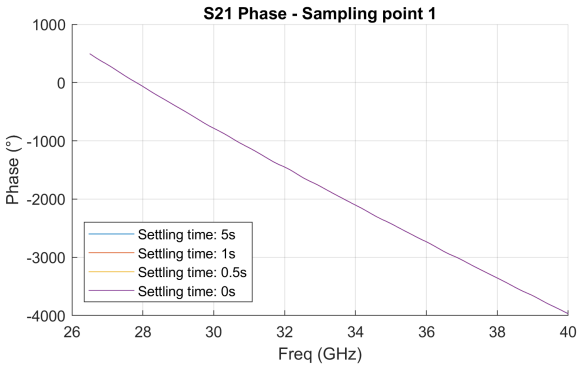
(a) Magnitude



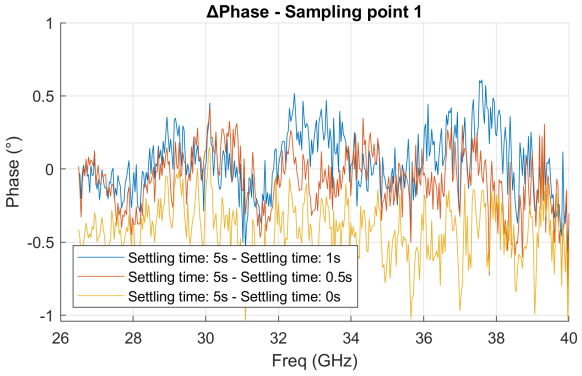
(b) Magnitude difference

Figure 5.4: Magnitude settle time test at point 1.

The magnitude of each measurement appears very similar across all settling times. As seen in figure 5.4a, the maximum deviation is approximately 0.1 dB, but no clear correlation between settling time and magnitude is observed. This suggests that, within the tested range, the effect of settling time on the measured magnitude is minimal.

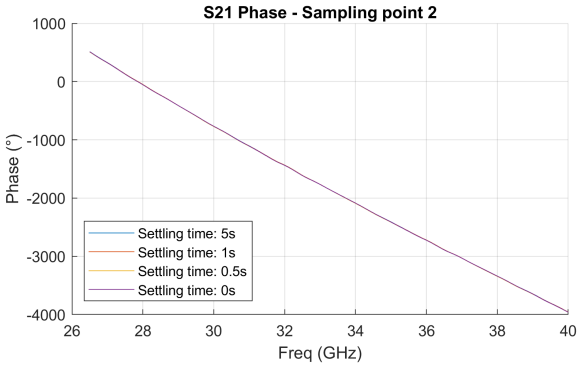


(a) Phase

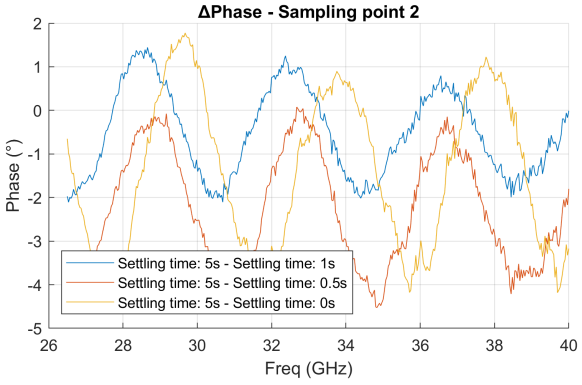


(b) Phase difference

Figure 5.5: Phase settle time test at point 1.



(a) Phase



(b) Phase difference

Figure 5.6: Phase settle time test at point 2.

Although figure 5.5a shows that the phase follows a similar overall trend for all settling times. Figure 5.5b reveals differences in phase of up to 4°. However, just like the magnitude, no clear correlation

between settling time and phase is observed. Indicating the effect of settling time on the measured phase is minimal.

One interesting observation is that every third sampling point shows significantly more noise in the phase difference. This can be seen in the full plot in figure C.7 in Appendix C, as well as in the examples for individual points: figure 5.5a (representative of every third point) and figure 5.6a (representative of the remaining points). One possible cause could be the path the robot arm follows to measure all sampling points, which uses a snake-like pattern. In this pattern, every third point requires movement primarily in the x-direction, whereas the others involve movement in the y-direction. This directional difference may introduce mechanical inconsistencies or disturbances, resulting in increased phase noise at every third point. However, further testing is required to investigate and confirm this phenomenon.

5.2.3. Discussion

The results do not show a clear correlation between the settling time and deviations in the magnitude or the phase. However, the effect of the robot arm's approach direction to the sampling points needs to be explored.

5.3. Phase error due to joint motion

The next area to consider is the relationship between specific robot joint movements and the phase changes measured. Since the cable is attached to the body of the robot, it moves together with each joint's movement. The goal of this analysis is to understand and quantify the effects certain joint movements could have on the phase measured. The findings from this analysis could be substantial to improve the system; for example, if it can be demonstrated that certain joints produce a greater phase offset, optimising the robot's movement to keep this joint stationary will result in better antenna measurements. Additionally, the outcomes of this investigation could be used to justify and improve subgroup 1's cost values for certain movements, which in turn has an effect on the optimum path the robot should take.

5.3.1. Setup

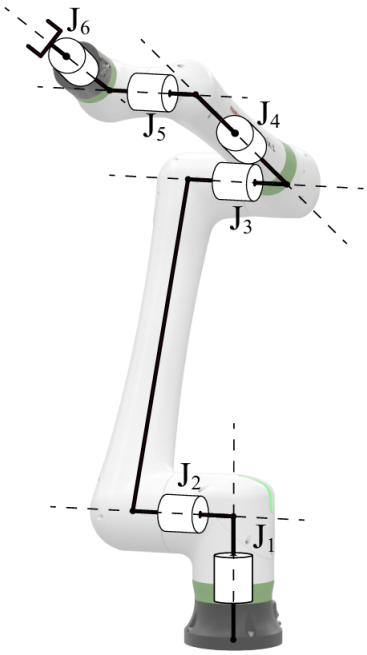
To collect informative data and support the conclusions drawn later, it is helpful to define the measurement setup that was used. The process for the measurements is as follows:

1. Bring the robot to a certain home position.
2. Calibrate the VNA for the port connected to the robot arm
3. Attach a short to the measurement terminal on the robot arm.
4. Take repeat measurements to determine the deviation between measurements at the home position.
5. Move each joint through a total angle greater or equal to 90 degrees, recording measurement for each position.
6. Plot and analyse the results of the S11 parameters.

The robot's home position was chosen to be the pose in figure 5.7a, and the VNA was assumed calibrated, since calibration had been performed a few days previously and the VNA setup remained unchanged. This assumption turned out to be incorrect, and will be clarified in the results of this analysis. A short was attached at the port instead of the probe, since it was thought that the probe would measure undesired reflections when oriented several ways during the test; a short would give a constant reflection no matter the orientation, which makes the phase results for S11 more comparable. All measurements performed in this analysis were done with identical VNA settings (IF Bandwidth, start/stop frequency, etc) as with the full-scale antenna measurements for consistency.

Next, repeat measurements in one position should be taken to test the deviation between measurements. For this, a repeat measurement is taken immediately after the initial measure without movement in between, and one is taken with movement in between the measurements, but ensuring the robot re-

(a) The home position for testing phase errors per joint.



(b) Fanuc CRX-20iA/L joint architecture. *Image reproduced from [29] © 2023 by the authors of [29]*

Figure 5.7: The home position for the robot and it's joint architecture.

turns to the home position before the next measurement. From these, we can determine if there is a deviation caused by the VNA between measurements or if noise is a major factor.

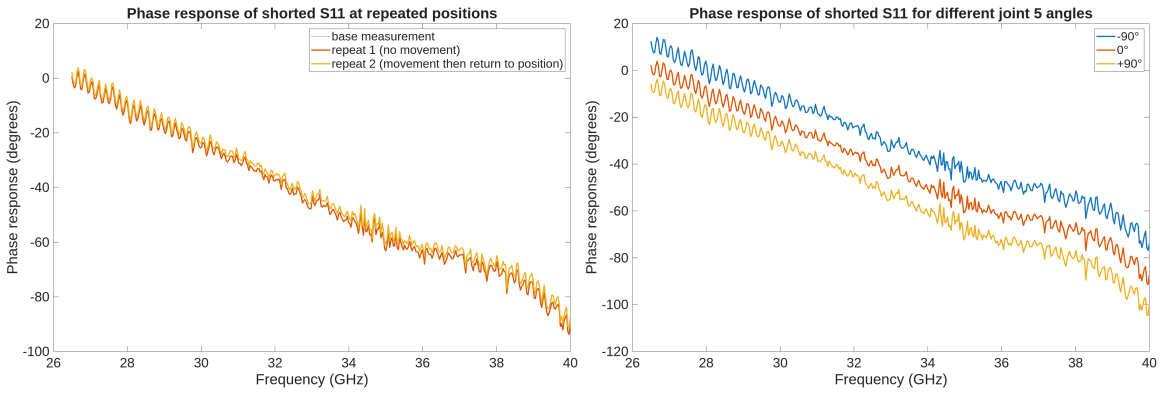
Lastly, the process of manually moving each joint though large angles and performing measurements can begin. For each angle, a picture is taken, and the corresponding measurement is recorded. This allows for comparing the plotted results with the images to aid in proposing an explanation.

5.3.2. Results

The results of the repeated measurements are shown in figure 5.8a. All three results show a slight linear slope decreasing from 0° as frequency increases. The rate at which the slope decreases is approximately $-7°$ / GHz. Furthermore, it can be seen that the artifacts (certain peaks or volatile areas, at 34.5 GHz for example) are conserved for all measurements.

Figure C.9 presents a collection of graphs that show the effect of each joint rotation on the phase offset. For each joint rotation, a corresponding image of the real setup has been included in appendix B to give the reader a visual understanding of what each joint's rotation means in physical space. Additionally, figure 5.7b shows a digram with all the joints labelled corresponding to the labels on the plots.

The plots clearly show that there is a significant phase offset for certain joints. An example plot for one of the joints is presented in figure 5.8b to offer a visual aid for the ensuing discussion. For joints 1 and 2, a phase difference of over 20° can be observed, which is significant. Continuing, for joints 3, 4 and 5, a phase difference of around 10° is observed, which is still notable. Lastly, only joint 6 shows a phase offset within 4° which could be considered minor, but not negligible for high accuracy measurements. The discussion will delve into the analysis of these results.



(a) Phase offsets discovered for repeated measurements at the home position

(b) Phase offset discovered when moving joint 5 through a $\pm 90°$ relative angle

Figure 5.8: Results from the phase error per joint measurements

5.3.3. Discussion

From the repeatability results (figure 5.8a), two major conclusions can be made. Firstly, the VNA is not calibrated for the home position. Since a short is attached, the expected reflection coefficient is $\Gamma = -1$, thus a 180° constant phase (flat response) is expected over all frequencies. However, the results show a slightly decreasing linear slope as frequency increases. This shows the importance of calibration before every measurement, even if the VNA parameters are unchanged like in this situation. Furthermore, the phase beginning from 0° at 26.5 GHz with this slope shows that the phase starts at 180° for lower frequencies as expected. Fortunately, the results measured can still be used to analyse the phase offset per joint, even though the VNA is not calibrated. This is because the reflection of the short remains constant between measurements. This is proven in figures 5.8a and C.9 which all show the same downwards slope for different joint positions, just at different phase offsets. This means that the phase offset between measurements can still be compared, which is precisely what this analysis tries to address.

The second conclusion that can be made is that the phase offset remains constant for measurements taken at the same robot configuration. Figure 5.8a shows a deviation of around 2°, which is deemed negligibly small. As such, the effect of noise or other random errors can be ignored.

The results in figure C.9 show a significant offset of up to 20°. It can be seen that the joints that control the greatest end-effector distance (joints 1 and 2), also produce the largest phase offsets. This pattern is continued for joints 3,4 and 5, which control moderate motion ranges, producing phase offsets of around $10°$, and the smallest joint (6) producing the smallest offset. This suggests that a relationship exists between the robot motion and the phase offset, which is an important conclusion that should be investigated further. To quantify these phase offsets, three further experiments have been conceived by subgroup 2, which will be explained below.

Assuming there exists an offset per joint, the objective of the first test is to graph the phase offset as a function of the joint angle. This reveals if there are certain predictable relationships which can be corrected for in post-processing. For example, if a strong quadratic relationship is discovered, a model could be fitted to the curve and used to cancel the effect of the joint angle. Additionally, even if a predictable relationship isn't found, it may reveal joint regions in which the phase offset is minimised or maximised, which can then be used to alter the cost values for certain joint movements in the path planning algorithm.

The second investigation that could be conducted is determining if the phase offset is a systematic error. This can be done by performing the intended robot arm path beforehand and measuring exclusively the phase offset during the motion. This can be done by attaching a short and repeating measurements for the intended path multiple times. If these repeated measurements are similar, it could suggest that the phase offsets are unaltered for a certain robot motion path. Thus, after performing the full antenna

measurement, the phase offset can be corrected for by subtracting the pre-measured phase offset for the path.

Lastly, the purpose of the third experiment is to find the effect of bending the cable on the phase offset. This test can be used to justify why the joint movements cause a change in phase by using a valid logical argument; if bending the cables produces a phase offset determined by a certain relationship, and the cables bend due to joint motion, then it can be concluded that robot motion contributes to a phase offset.

Unfortunately, there was insufficient time to perform these tests before the deadline for this thesis, but there is a possibility to continue the investigation to collect results for the BAP defence presentation.

6

Conclusion

This thesis explored the implementation of far-field and near-field antenna measurements using a 6-axis robot arm. This robotic setup offers new features and improvements over traditional measurement setups.

The robot arm can be manually controlled using a graphical user interface. The GUI allows you to send the robot arm to a specified location or move in steps in each direction or rotation. Although the setting of the speed of the robot is implemented in the GUI, the system was not yet capable of changing the speed setting on the physical robot arm itself.

This system enables a user-friendly interaction with the Vector Network Analyser. Through the GUI of the system, the user can easily connect to the VNA either by automatically detecting and selecting the correct address or by manual entering it. Additionally, the user can read the current frequency setting and change them when necessary.

In order for the antenna measurements to be performed in the correct sequence, a central control unit was implemented, taking care of integrating all the software and hardware components. This controller included a state machine that splits up all the functionality of each step in the measurement, improving the structure and readability of the code. Measurement data is automatically stored in a well-organised file. Saving all the obtained results, as well as all the antenna parameters and measurement settings, allowing for analysis at a later time. Through the GUI, the user can configure all the antenna measurement settings, track the process using time trackers and plots indication of the progress. Additionally, the user can observe the results of the test in real time and use the interface to apply time-gating to the current measurement and observe the effect on the results.

To assess the reliability of the system, several tests were conducted. The focus of these tests included the repeatability of the system, the effect of a settle time between movement of the robotic arm and obtaining the data with the VNA, and the relation between joint movements and the phase errors observed in the measurements. Analysis of the repeatability of the system revealed that results may deviate by magnitude differences of 0.8 dB and phase differences up to 30°. Compared to a calibrated position of the system, displacements in individual joints led to significant phase difference of up to 20°. There was no correlation was found in deviations magnitude or phase response and the settle time between the movements of the robot and the measuring of the VNA.

6.1. Future work

Although the setup in this thesis already offers several improvements over existing work, observations made during the project have led to several recommendations for future work:

Selection of required S-parameter: To reduce the measurement time, the system could be optimised to only measure the required S-parameters, specified by the user.

Systematic error correction: Further research can be done on predictable causes of errors,

such as error introduced by specific joint movements. If there is a relation between the joint movement and the error, the system adds a correction to the results.

Re-calibration option: It was found that large movement of the robot arm and the cable effect the repeatability of the measurement setup. Therefore, it is recommended to investigate methods to lower these errors or allow the users to recalibrate the VNA in a defined starting position at the beginning of each measurement.

Analyse approach direction: While testing the systems, a potential error was found seen related to the approach direction of the robot arm to the sampling points. Further research into this phenomenon is recommended.

Processed data visualisation: Currently the functionality, developed by subgroup 3 and 4, to process the obtained data is not yet implemented. Upcoming updates could add these methods and visualise the processed results in the GUI.

Bibliography

[1] K. Dausien, T. Korner, C. Schulz, N. Pohl, I. Rolfes, and J. Barowski, “Ultrawideband Millimeter-wave Robotic Antenna Measurements enabled by FMCW Radar Sensors,” en,

[2] A. Husein, K. Rasilainen, J.-P. Makela, A. Parssinen, and M. E. Leinonen, “6G OTA Measurements at Sub-THz Band Using a Compact Robotic System,” en,

[3] S. F. Gregson and C. G. Parini, “Use of Compressive Sensing Techniques for the Rapid Production Test of Commercial Nose-Mounted Radomes in a Robotic Antenna Measurement System,” en,

[4] B. Sievert, J. T. Svejda, D. Erni, and A. Rennings, “Spherical mm-Wave/THz Antenna Measurement System,” *IEEE Access*, vol. 8, pp. 89 680–89 691, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2993698. [Online]. Available: https://ieeexplore.ieee.org/document/9091044/ (visited on 05/27/2025).

[5] admin, *Optimizing VNA Measurement Speed*, en-US, Jan. 2023. [Online]. Available: https://coppermountaintech.com/optimizing-vna-measurement-speed/ (visited on 04/30/2025).

[6] B. L. Moser, J. A. Gordon, and A. J. Petruska, “Increased Efficiency in Planar Near-Field Scanning Using Combined Multi-Robot Motion,” en,

[7] H. Jansen and D. Heberling, “Numerical Analysis of Positioning Errors in Irregular Near-Field Antenna Measurements Using Robot-Based Systems,” en,

[8] V. S. Kalashnikov, M. Y. Ponomarev, O. Y. Platonov, *et al.*, *Near-Field Antenna Measurements: Calculations and Facility Design* (Springer Aerospace Technology), en. Singapore: Springer Singapore, 2021, ISBN: 978-981-336-435-6 978-981-336-436-3. DOI: 10.1007/978-981-33-6436-3. [Online]. Available: https://link.springer.com/10.1007/978-981-33-6436-3 (visited on 04/29/2025).

[9] Mathworks, *Creating GUIs*. [Online]. Available: https://www.mathworks.com/help/matlab/creating_guis/share-data-among-callbacks.html (visited on 04/30/2025).

[10] Mathworks, *MATLAB ifft*. [Online]. Available: https://nl.mathworks.com/help/matlab/ref/ifft.html#bvizm5c-1-symflag (visited on 06/06/2025).

[11] C. Parini, S. Gregson, J. McCormick, D. J. v. Rensburg, and T. Eibert, “Theory and practice of modern antenna range measurements. volume 1,” eng, in ser. Electromagnetics and Radar, 2nd expanded edition, Num Pages: 1, London: The Institution of Engineering and Technology, 2021, ISBN: 978-1-83953-127-9. [Online]. Available: https://app-knovel-com.tudelft.idm.oclc.org/web/view/pdf/show.v/rcid:kpQXFZ2QB1/cid:kt012I13X3/viewerType:pdf//root_slug:theory-practice-modern/url_slug:introduction?cid=kt012I13X3&b-toc-cid=kpQXFZ2QB1&b-toc-root-slug=theory-practice-modern&b-toc-title=Theory%20and%20Practice%20of%20Modern%20Antenna%20Range%20Measurements%20%282nd%20Expanded%20Edition%29%2C%20Volume%201&b-toc-url-slug=introduction&kpromoter=marc (visited on 05/28/2025).

[12] G. E. Evans, *Antenna measurement techniques* (The Artech house antenna library), eng. Boston London: Artech house, 1990, ISBN: 978-0-89006-375-0.

[13] R. Johnson, H. Ecker, and J. Hollis, “Determination of far-field antenna patterns from near-field measurements,” *Proceedings of the IEEE*, vol. 61, no. 12, pp. 1668–1694, 1973, ISSN: 0018-9219. DOI: 10.1109/PROC.1973.9358. [Online]. Available: http://ieeexplore.ieee.org/document/1451288/ (visited on 05/28/2025).

[14] D. M. Lewis, J. Bommer, G. E. Hindman, and S. F. Gregson, “Traditional to Modern Antenna Test Environments: The Impact of Robotics And Computational Electromagnetic Simulation on Modern Antenna Measurements,” in *2021 15th European Conference on Antennas and Propagation (EuCAP)*, Dusseldorf, Germany: IEEE, Mar. 2021, pp. 1–5, ISBN: 978-88-31299-02-2. DOI: 10.23919/EuCAP51087.2021.9411068. [Online]. Available: https://ieeexplore.ieee.org/document/9411068/ (visited on 05/28/2025).

[15] C. Matos, J. Humanchuk, and N. Ghalichechian, “Robotically□controlled antenna measurement system for millimeter□wave applications,” en, *Microwave and Optical Technology Letters*, vol. 63, no. 5, pp. 1520–1525, May 2021, ISSN: 0895-2477, 1098-2760. DOI: 10.1002/mop.32773. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/mop.32773 (visited on 05/21/2025).

[16] Tektronix, *Introduction to VNA Basics*, en, Mar. 2017. (visited on 04/28/2025).

[17] D. M. Pozar, *Microwave engineering*, eng, Fourth edition. Hoboken, NJ: John Wiley & Sons, Inc, 2012, ISBN: 978-0-470-63155-3 978-1-118-21363-6.

[18] F. Caspers, *RF engineering basic concepts: S-parameters*, arXiv:1201.2346 [physics], Jan. 2012. DOI: 10.48550/arXiv.1201.2346. [Online]. Available: http://arxiv.org/abs/1201.2346 (visited on 06/01/2025).

[19] Copper Mountain Technologies, *VNA Calibration Theory Introduction*, Apr. 2023. [Online]. Available: https://coppermountaintech.com/an-introduction-to-vna-calibration-theory/ (visited on 06/12/2025).

[20] G. Engen and C. Hoer, “Thru-Reflect-Line: An Improved Technique for Calibrating the Dual Six-Port Automatic Network Analyzer,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 27, no. 12, pp. 987–993, Dec. 1979, ISSN: 0018-9480, 1557-9670. DOI: 10.1109/TMTT.1979.1129778. [Online]. Available: https://ieeexplore.ieee.org/document/1129778/ (visited on 04/28/2025).

[21] Tektronix, *What is VISA*. [Online]. Available: https://www.tek.com/en/support/faqs/what-visa (visited on 06/02/2025).

[22] Mathworks, *Get started with VISA*. [Online]. Available: https://nl.mathworks.com/help/instrument/visa-overview.html (visited on 06/02/2025).

[23] Open Robotics, *ROS Developer Documentation*. [Online]. Available: https://docs.ros.org/#ros-for-beginners (visited on 06/02/2025).

[24] Agilent Technologies, *VNA Guide*. [Online]. Available: https://www.keysight.com/us/en/assets/9018-05243/programming-guides/9018-05243.pdf (visited on 06/02/2025).

[25] L. F. Chaparro and A. Akan, *Signals and systems using MATLAB®* (MATLAB examples), eng, Third edition. London San Diego, CA Cambridge, MA Kidlington, Oxford: Elsevier, Academic Press, 2019, ISBN: 978-0-12-814204-2.

[26] P. Heckbert, *Fourier Transforms and the Fast Fourier Transform (FFT) Algorithm*, Jan. 1998. [Online]. Available: https://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/99/pub/www/notes/fourier/fourier.pdf (visited on 06/06/2025).

[27] J. W. Boyles, F. G. Parkway, and S. Rosa, “Far-Field Antenna Measurements with the HP 8510 Network Analyser,” en,

[28] Fanuc, *Crx-20ia-l-data-sheet*. [Online]. Available: https://www.fanucamerica.com/docs/default-source/crx-data-sheets/crx-20ia-l-data-sheet.pdf (visited on 06/11/2025).

[29] L. Carbonari, M.-C. Palpacelli, and M. Callegari, “Inverse Kinematics of a Class of 6R Collaborative Robots with Non-Spherical Wrist,” en, *Robotics*, vol. 12, no. 2, p. 36, Mar. 2023, ISSN: 2218-6581. DOI: 10.3390/robotics12020036. [Online]. Available: https://www.mdpi.com/2218-6581/12/2/36 (visited on 06/11/2025).

A

GUI Screenshots

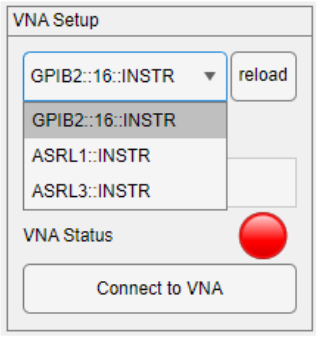
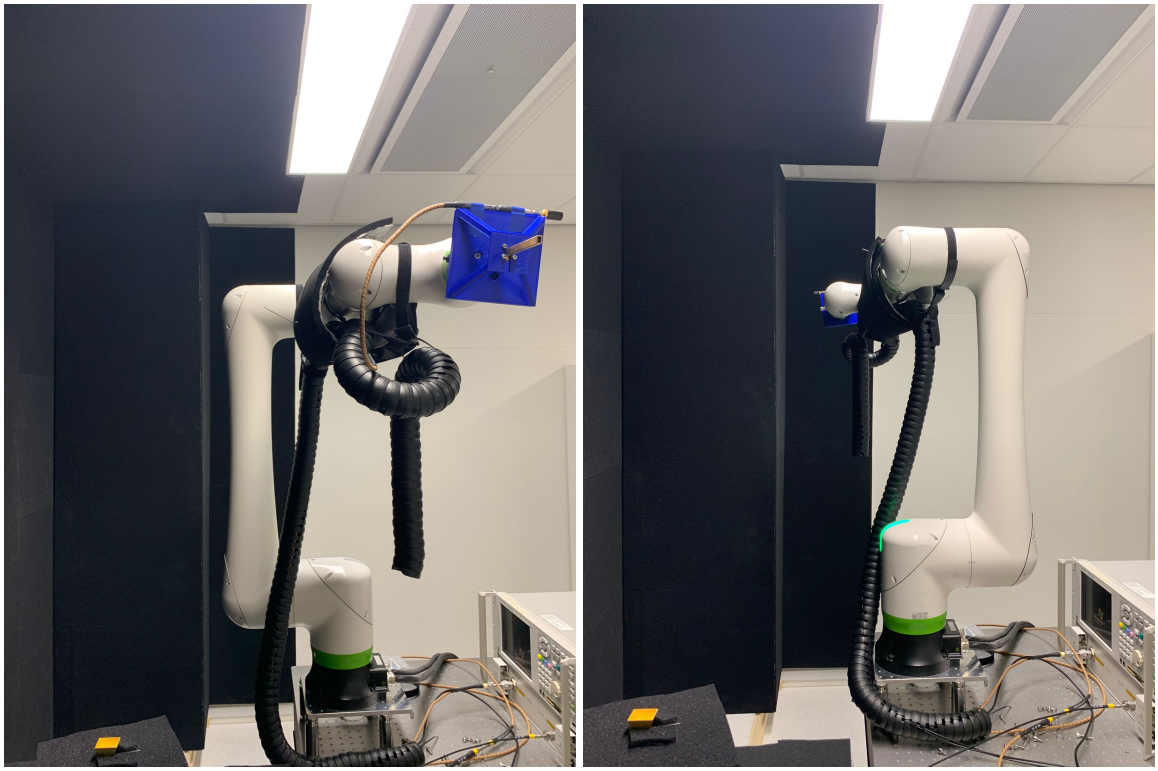


Figure A.1: Drop-down menu VNA selection.

B

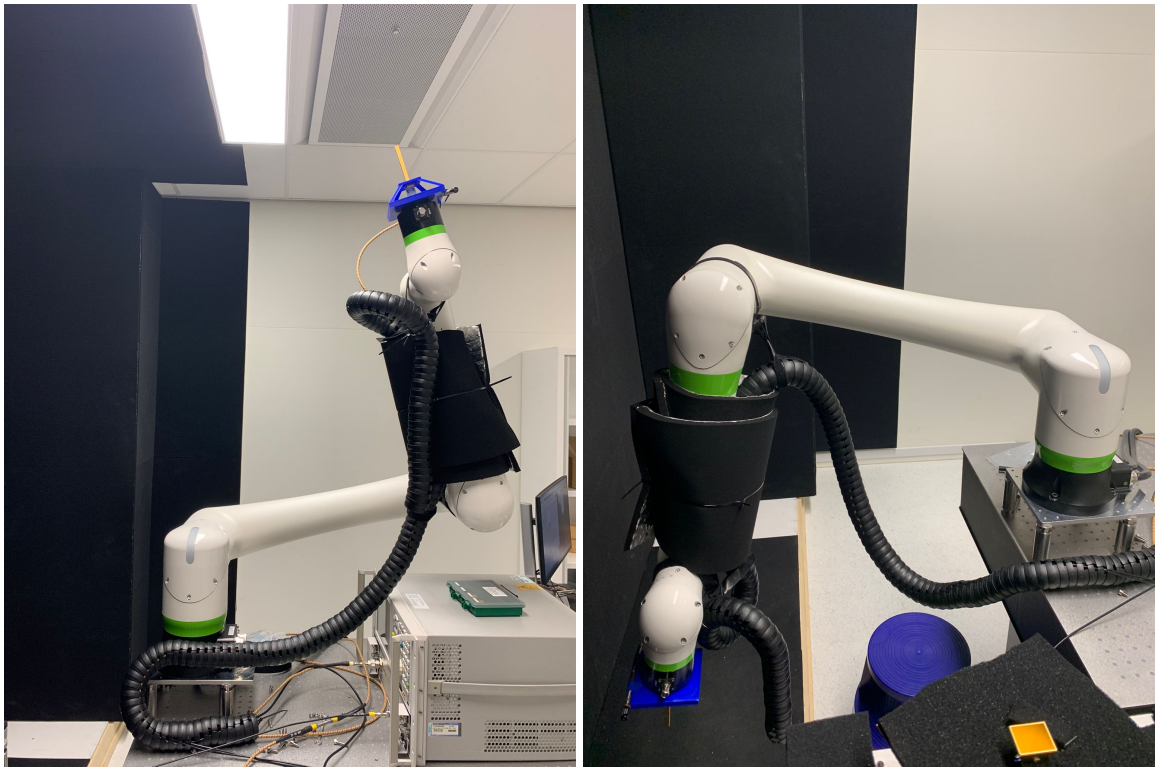
Robot Configurations

Figure B.1: Home position for the robot for the joint phase offset analysis.



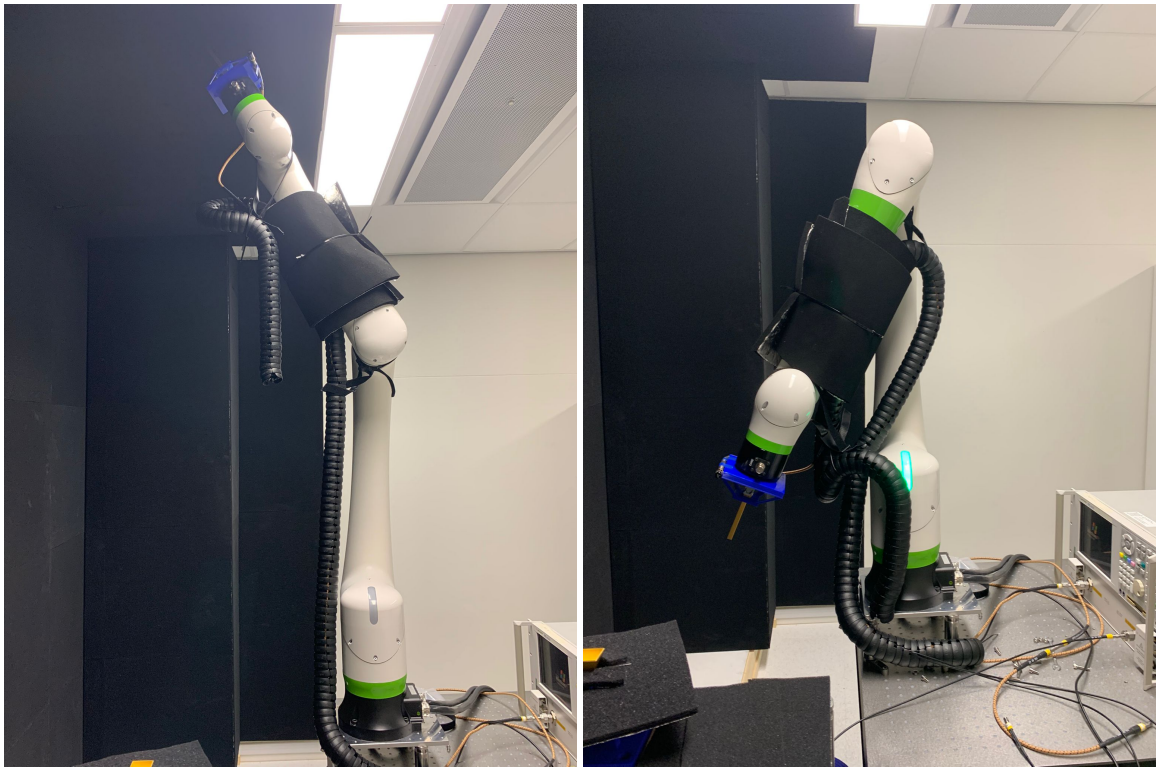
(a) Joint 1 at $90°$ (b) Joint 1 at $-90°$

Figure B.2: Joint 1 angle offset from home.



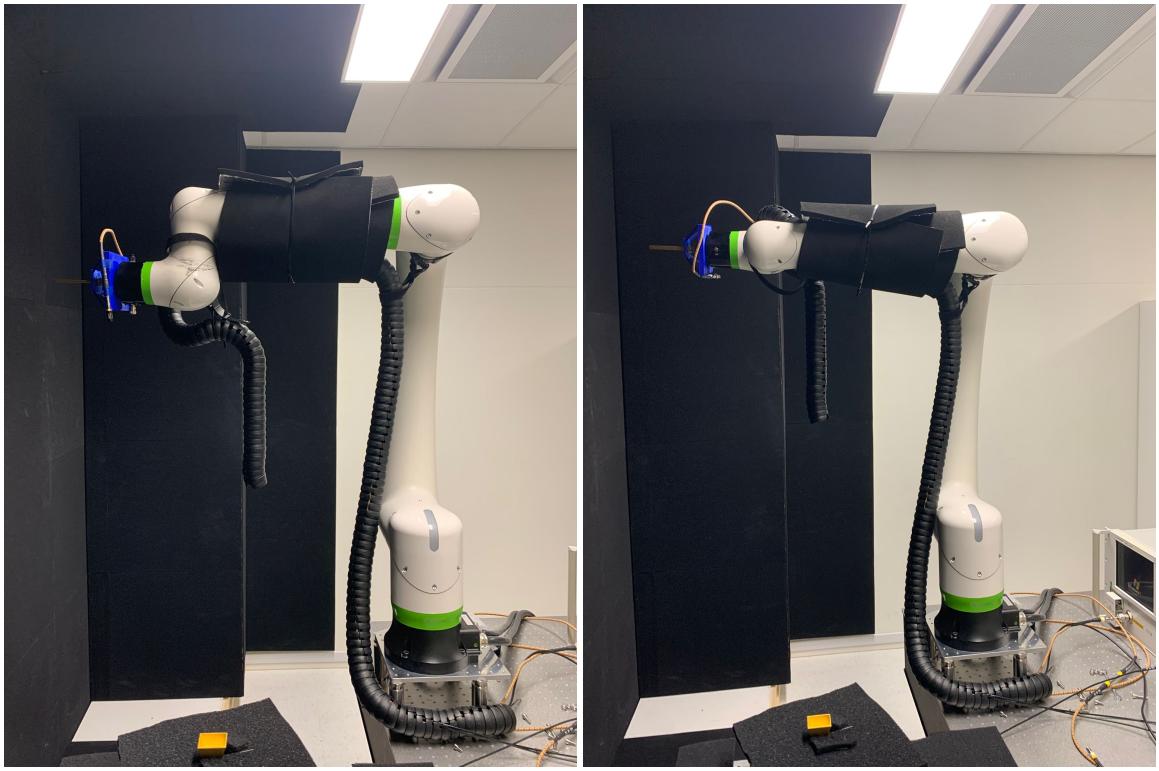
(a) Joint 2 at $80°$ (b) Joint 2 at $-80°$

Figure B.3: Joint 2 angle offset from home.



(a) Joint 3 at $60°$ (b) Joint 3 at $-60°$

Figure B.4: Joint 3 angle offset from home.

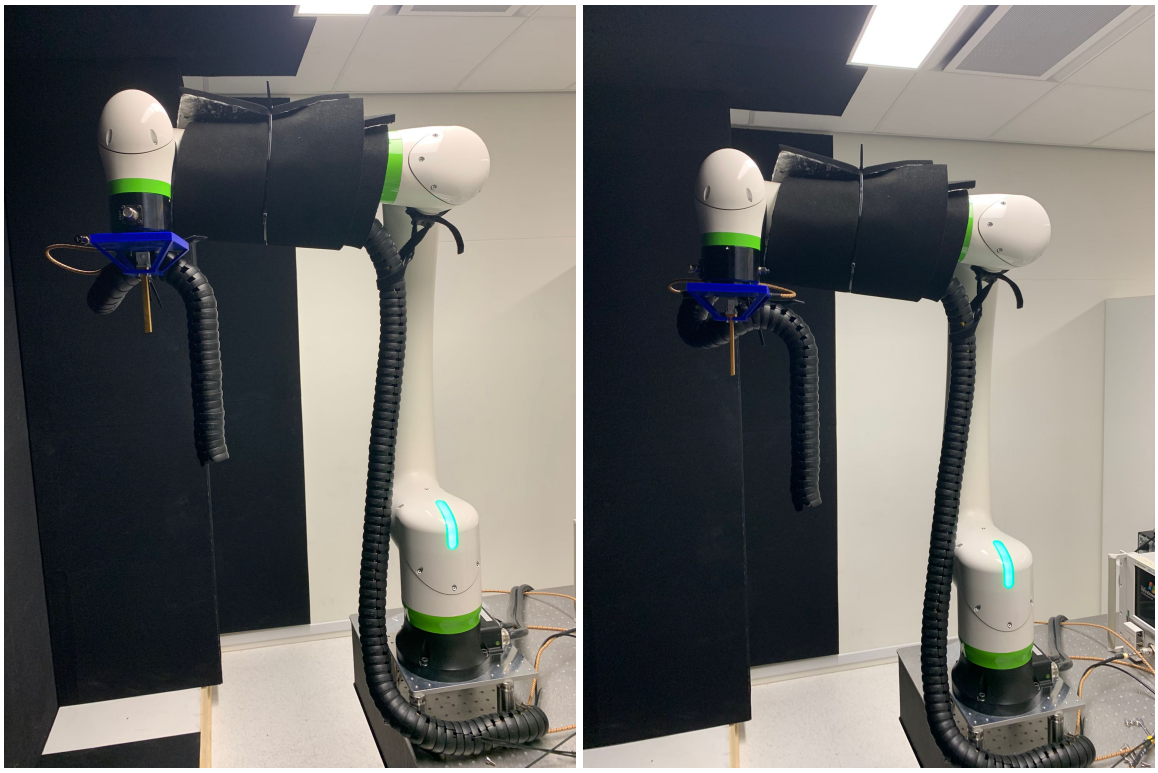


(a) Joint 4 at $0°$ (b) Joint 4 at $-90°$

Figure B.5: Joint 4 angles. *Note: Home position has joint 4 at* $90°$

(a) Joint 5 at 90° (b) Joint 5 at -90°

Figure B.6: Joint 5 angle offset from home.



(a) Joint 6 at 0° (b) Joint 6 at 90°

Figure B.7: Joint 6 angles. *Note: other configuration than home used here as probe is often in this orientation.*

C

Result plots

C.1. Repeatability

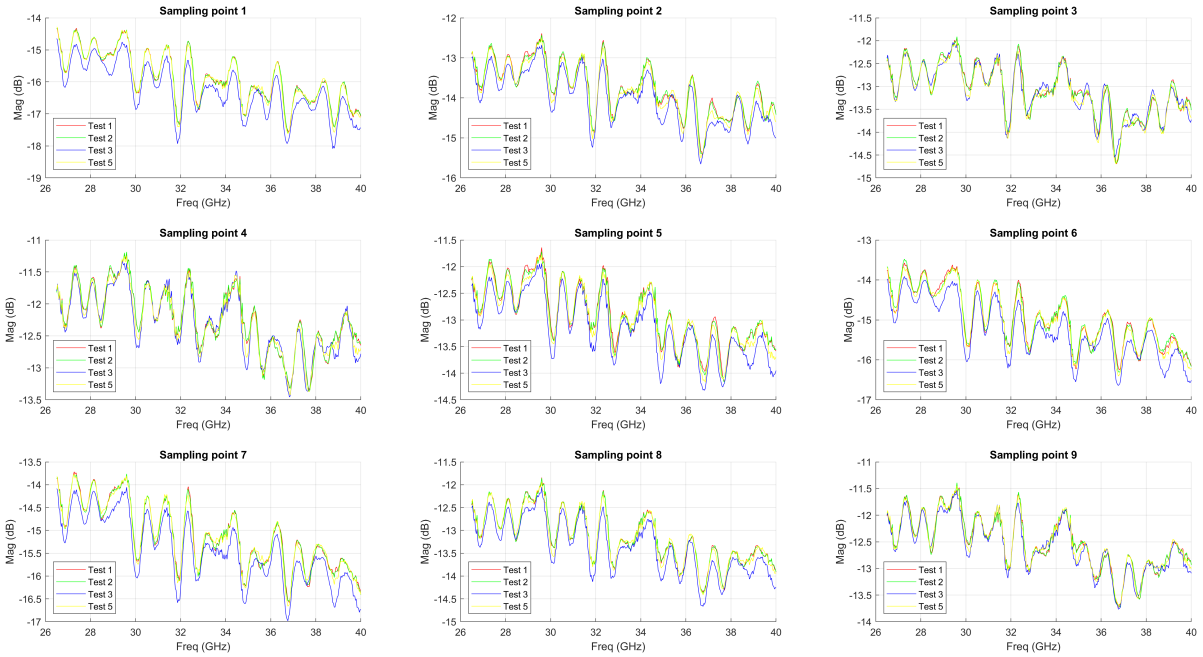


Figure C.1: Magnitude repeatability test

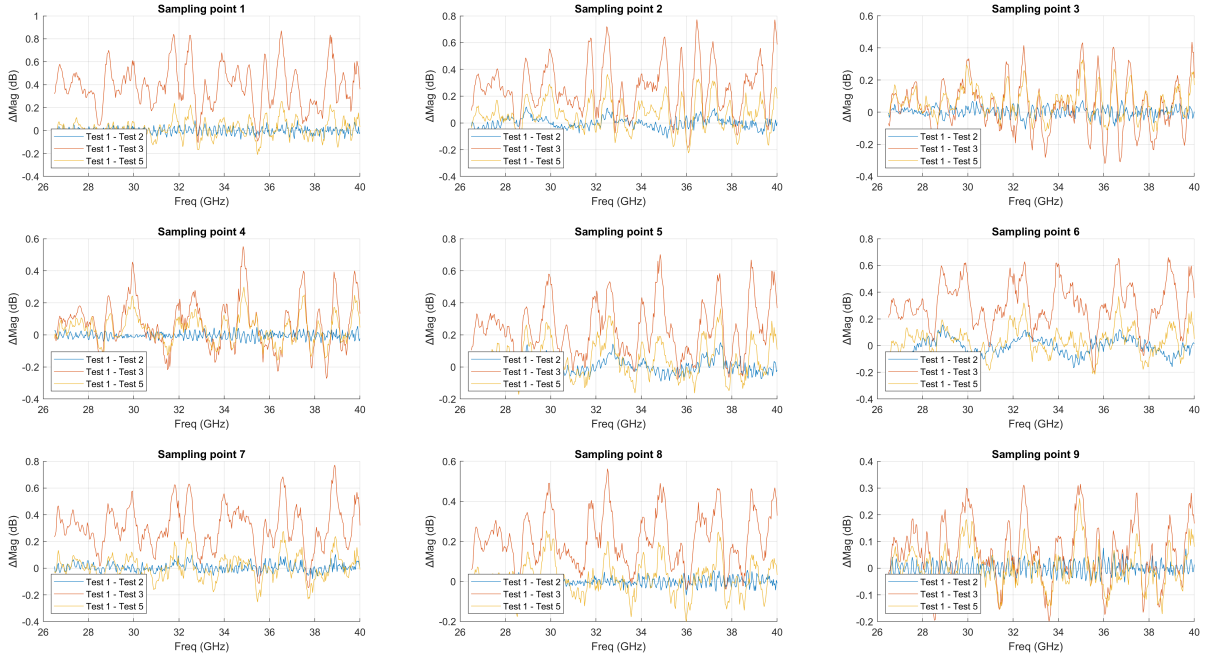


Figure C.2: Magnitude difference repeatability test

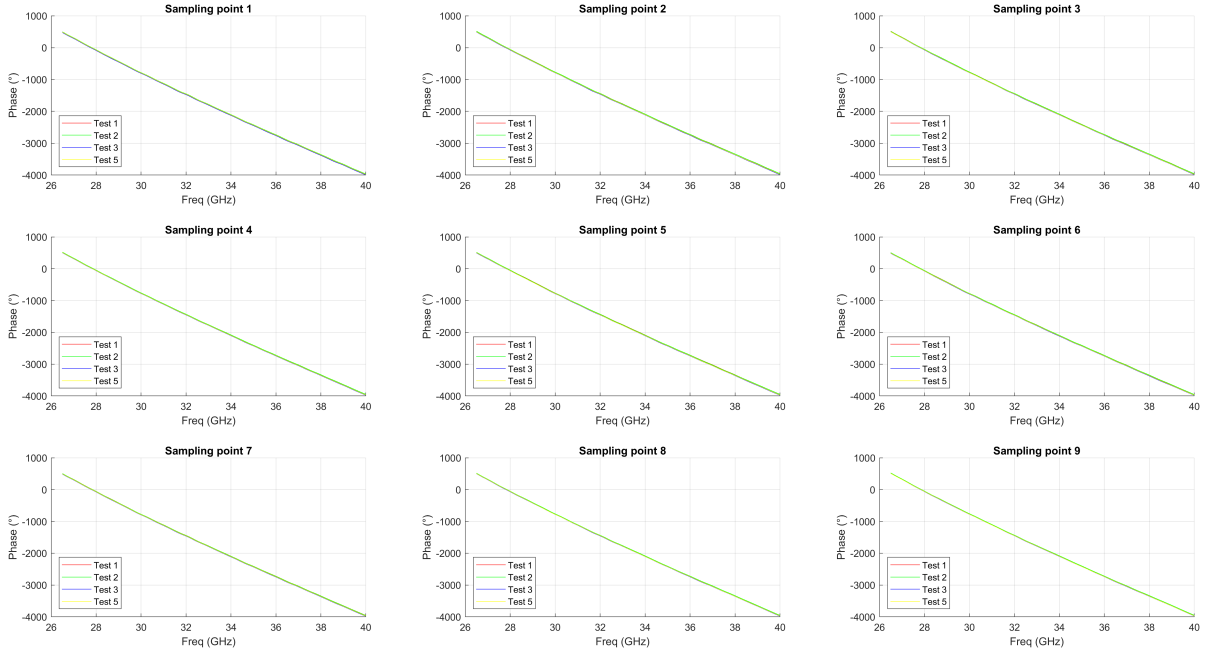


Figure C.3: Phase repeatability test

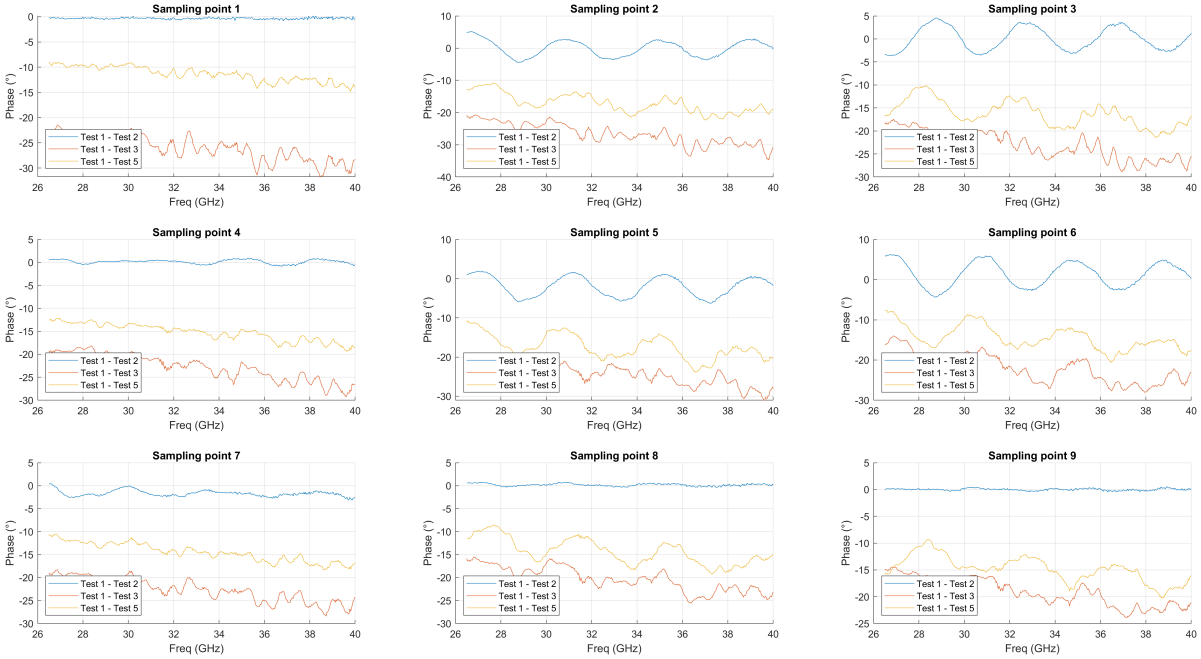


Figure C.4: Phase difference repeatability test

C.2. Settle time

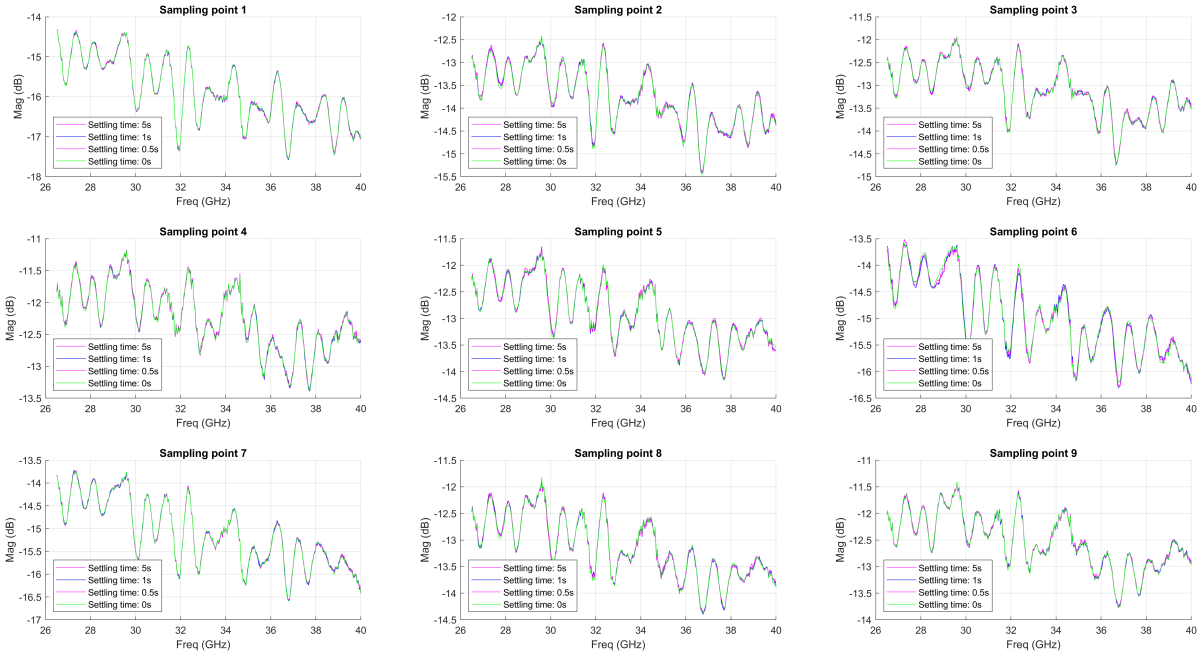


Figure C.5: Magnitude settle time test

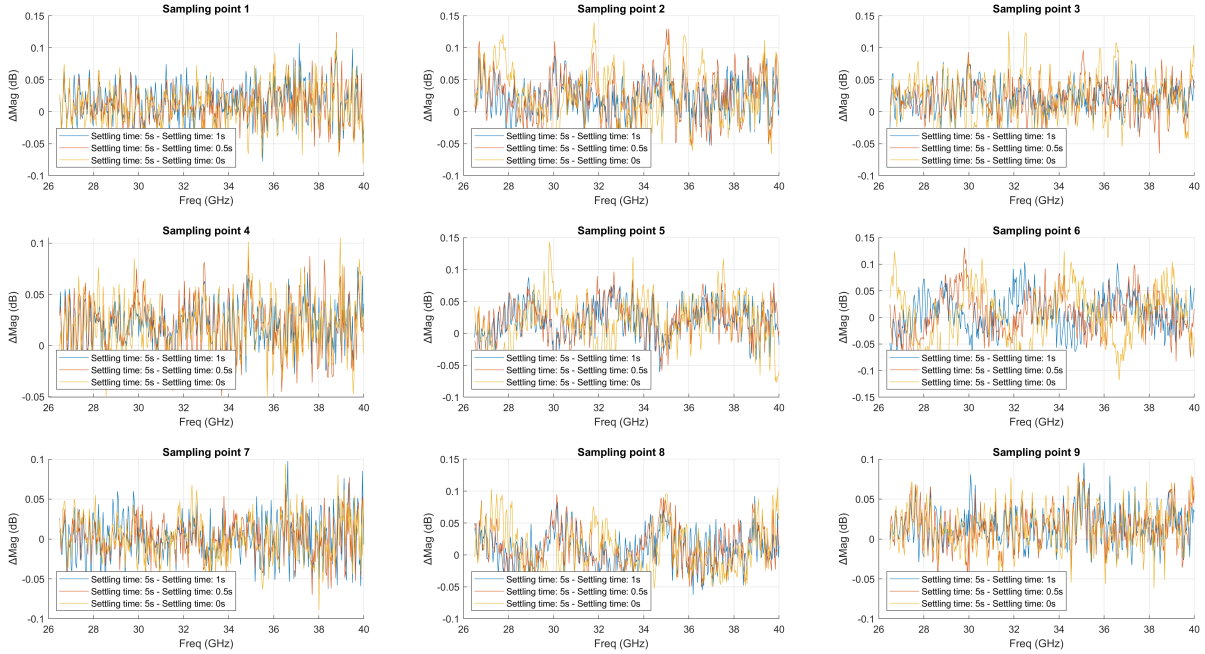


Figure C.6: Magnitude difference settle time test

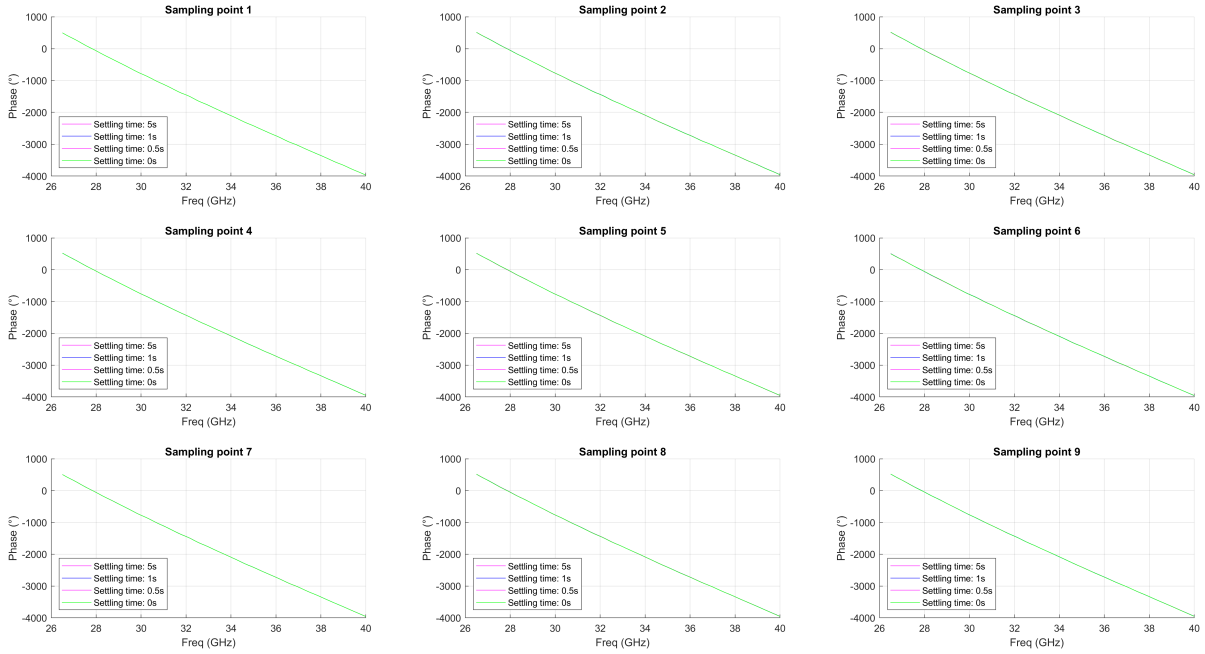


Figure C.7: Phase settle time test

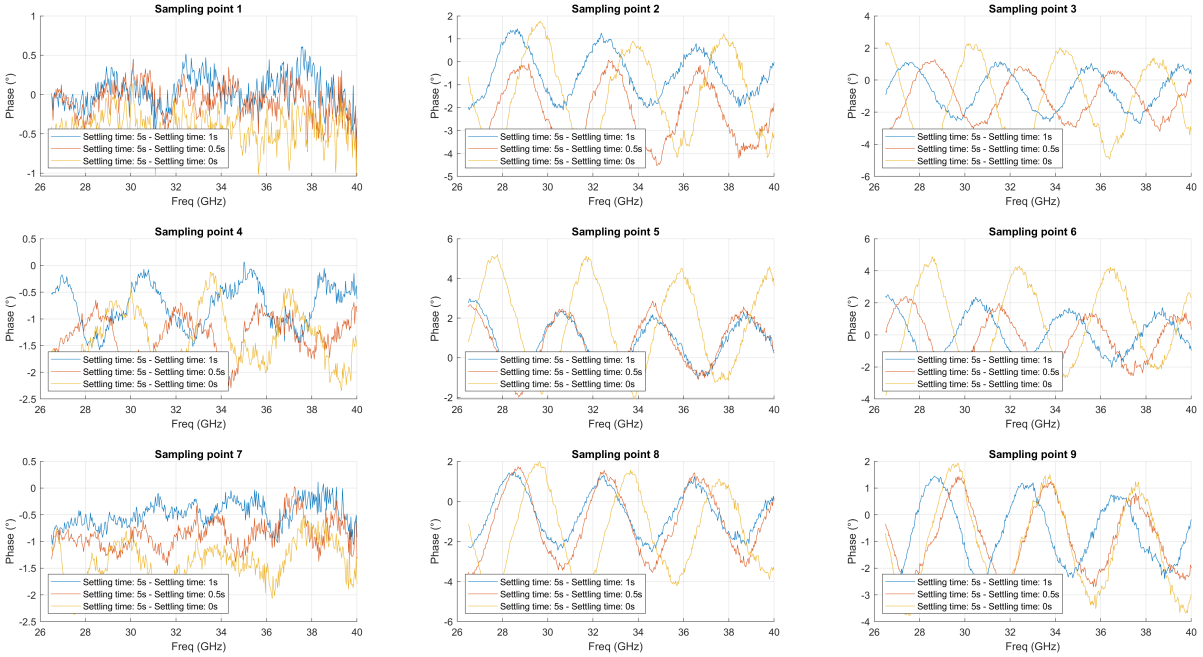


Figure C.8: Phase difference repeatability test

C.3. Phase offset per joint

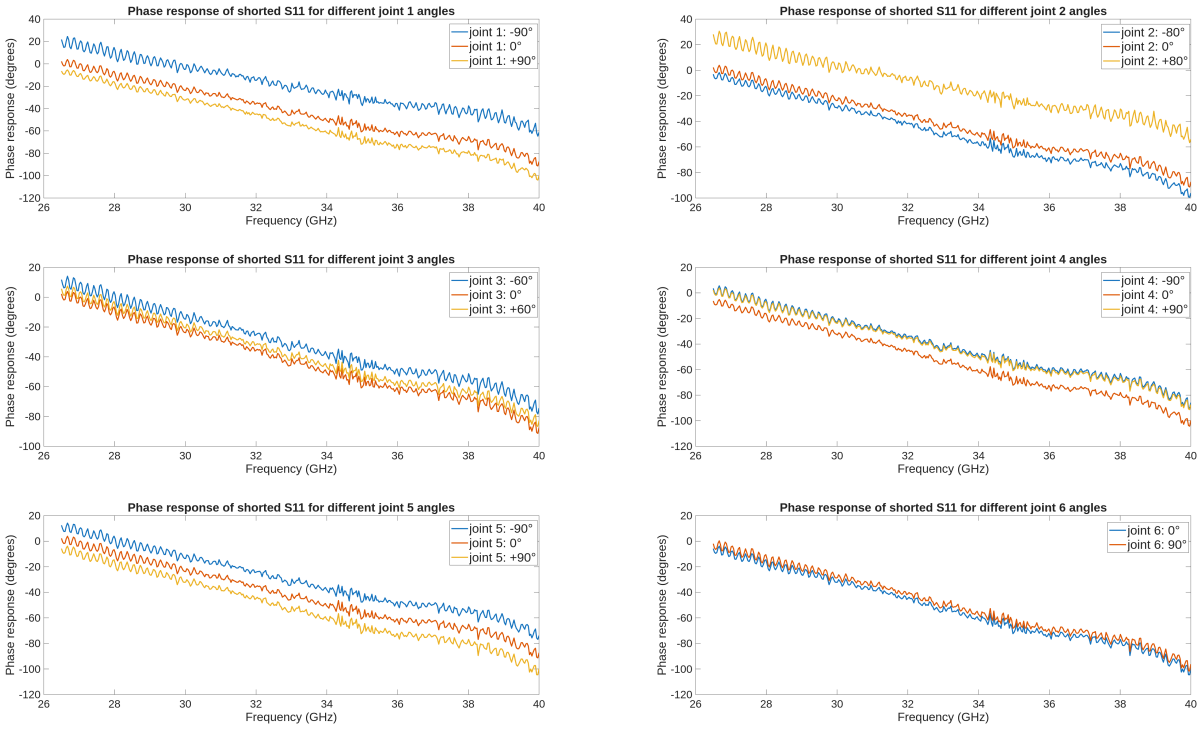


Figure C.9: The phase offsets per joint

D

Code snippets

The completed and most recent code can be found at https://gitlab.ewi.tudelft.nl/ghak/thz-bap-2025-prj.git.

D.1. Controller

```
function update_state_machine(self)
    switch current_state
    case 0  % Idle
        self.idel_state();
    case 1  % Initialize measurement
        self.initialize_state()
    case 2  % Move to next point
        self.move_to_next_state();
    case 3  % Wait for robot to reach point
        self.wait_for_robot_state();
    case 4  % Measure with VNA
        self.measure_state();
    case 5  % Wait for measurement to complete
        self.wait_for_measurement_state();
    case 6  % End measurement process
        self.end_state();
    end
end
```

Listing D.1: Function: `update_state_machine()`

```
function move_to_target(self,target_point)
    self.robot.move_to_target(target_point)
end
```

Listing D.2: Wrapped Method: `move_to_target()`

D.2. VNA

```
% Get the start frequency in Hz currently set on the VNA.
startfrequency = str2num(vna.Read('sense:frequency:start?'));

% Set the start frequency in Hz on the VNA.
command = sprintf('sense:frequency:start %i', startfrequency);
vna.WriteLine(command);
```

Listing D.3: Example SCPI commands to set and retrieve the start frequency VNA parameter

```
% Put the VNA in single mode.
vna.WriteLine('init:cont off');
```

Listing D.4: Setting the VNA to single mode.

```
% start an new measurement (Trigger channel 1)
vna.WriteLine('initiate1:immediate');
vna.Read('*OPC?');                          % Wait until commands are done.

% Set ASCII character encoding
vna.WriteLine('format ASCII');

% Define a VNA Measurement
command = sprintf('calculate%i:parameter:define '+"'meas_%s'"+',%s',
    channel, param, param);
vna.WriteLine(command);

% Select this measurement for reading
command = sprintf('calculate%i:parameter:select '+"'meas_%s'", channel,
    param);
vna.WriteLine(command);

% Read the data
command = sprintf('calculate%i:DATA? SDATA', channel);
temp_data = str2num(vna.Read(command));

% temp data consists of interleaved real and imag. parts.
data = temp_data(1:2:end)+1j*temp_data(2:2:end);
```

Listing D.5: Code to take a measurement of a single s-parameter from the VNA

D.3. Post processing

```
% Generate the frequency axis in Hz
f_axis = linspace(start_freq, stop_freq, length(s_params));
% Plot the magnitude in dB with frequency axis in GHz
plot(mag_axis, f_axis ./ 1e9, 20*log10(abs(s_params)));
% Plot the unwrapped phase with the frequency axis in GHz
plot(phase_axis, f_axis ./ 1e9, unwrap(angle(s_params)));
```

Listing D.6: Plotting raw frequency data.

```
% Step 1: zero pad to get single-sided spectrum from DC to the highest
    frequency
delta_f = f_axis(2) - f_axis(1);
n_zeros = round(start_freq / delta_f);
s11_single_sided = [zeros([1, n_zeros]), S11_1];

% Step 2: flip and conjugate to get the two sided spectrum
s11_conj_flip = conj(flip(s11_single_sided));
s11_two_sided = 0.5 * [s11_single_sided(1), s11_single_sided(2:end),
    s11_conj_flip(1:end-1)];

% Step 3: apply ifft and zeropad to next power of 2
n_fft = 2^nextpow2(length(s11_two_sided));
s11_time = ifft(s11_two_sided, n_fft);
```

Listing D.7: Applying inverse discrete Fourier transform.

```
% find the indices of the time data that are within the gate boundaries.
% Gate boundaries (user input) are in ns,
% t_axis is in seconds, so need to convert.
start_index = find(t_axis .* 1e9 >= gate_start, 1, 'first');
stop_index = find(t_axis .* 1e9 <= gate_start+gate_size, 1, 'last');

% Allocate memory for gated_data as zeros
gated_data = zeros(1, length(t_axis));
% Only copy the data between the boudaries to the gated data,
% the rest remains zero.
gated_data(start_index:stop_index) = time_data(start_index:stop_index)
```

Listing D.8: Applying time-gating to the time domain.

```
% return to frequency domain by applying fft
% n_fft is the same as applied during the ifft previously.
gated_s_params = fft(gated_data, n_fft);
% trim the zero-padded data to the length of the two-sided spectrum
% the ifft was performed on previously
gated_s_params = 2 * gated_s_params(1:two_sided_length);
```

Listing D.9: Applying fft to return to frequency domain

```
% generate Tukey window
tukey_alpha = 0.2;
tukey_length = stop_index - start_index + 1;
tukey_window = tukeywin(tukey_length, tukey_alpha);

% apply the tukey window to the gated data
gated_data = zeros(1, length(t_axis));
gated_data(start_index:stop_index) = time_data(start_index:stop_index) .*
    tukey_window';
```

Listing D.10: Applying a Tukey window.

D.4. Previously Used Example Code

```
%% How to use this class:
% vna = VNA.Get();
% vna.Connect('address_string');
% fsstart = vna.GetStartFrequency();

% Suppress warnings:
%#ok<*NO4LP> % "Parentheses are not needed in a FOR statement."
%#ok<*ST2NM> % "If you are operating on scalar values, consider using
              % STR2DOUBLE for faster performance."
              % Unfortunately str2num handles the returned values from the
              % VNA well, while str2double does not.

classdef VNA < handle
    %% Static methods, can be called as VNA.<method>().
    methods(Static)
        function vna = Get()
            % Get the current VNA handle.
            persistent vna_;
            if(isempty(vna_))
                vna_ = VNA();
            end
```

```
            vna = vna_;
        end
    end
    properties(SetAccess = protected)
        SCPI
        isconnected (1,1) logical = false % Are we connected to the VNA?
        address (1,:) char = ''
    end

    %% Public methods, for interaction with the VNA.
    methods
        % Connect to the given address.
        % The address should be the complete string. E.g. 'GPIB0::16::
           INSTR'.
        function Connect(vna, addressstr)
            arguments
                vna (1,1) VNA
                addressstr (1,:) char
            end
            % Old method using visa (which is to be removed).
            % vna = visadev('keysight', addressstr);
            % vna.InputBufferSize=4e6;
            % vna.Timeout=120;
            % fopen(vna);

            % Debug code when VNA is not available.
%             vna.isconnected = true;
%             vna.address = addressstr;
%             return;

            % Get rid of any old connections we may have had.
            global VNA_SCPI; % #ok<GVMIS> % Globals are bad, I know.
            VNA_SCPI = [];

            % Connect to the new address.
            try
                fprintf('VNA.m: Connecting to ''%s''...\n', addressstr);
                % New method
                vna.SCPI = visadev(addressstr);
                % vna.SCPI.InputBufferSize=4e6; % No longer needed in
                   visadev, auto-managed.
                vna.SCPI.Timeout=120;
                % fopen(vna.SCPI); % No longer needed in visadev.

                % Store the interface for when Matlab decides to reset the
                % VNA class again...
                VNA_SCPI = vna.SCPI;

                % If we made it until here, we connected to the VNA.
                vna.address = addressstr;
                vna.isconnected = true;
                fprintf('VNA.m: Initial connection established.\n');

                % Now try to get some data from it.
                fprintf('VNA.m: Testing connection...\n');
                writeline(vna.SCPI, 'outp:state?');
```

```
            readline(vna.SCPI);

            fprintf('VNA.m: Connection successfully established.\n');
        catch err
            vna.Disconnect();
            fprintf('VNA.m: Failed to connect to ''%s''.\n',
                addressstr);
            rethrow(err);
        end
    end
    % Disconnect from the VNA.
    function Disconnect(vna)
        % This should do it, I think.
        vna.SCPI = [];

        % Also clear the global version.
        global VNA_SCPI; %#ok<GVMIS> % Globals are bad, I know.
        VNA_SCPI = [];

        vna.isconnected = false;
        vna.address = '';
    end
    % Makes sure we are connected to the VNA. Throws an error if not.
    function CheckConnection(vna)
        arguments
            vna (1,1) VNA
        end
        if(~vna.isconnected)
            errormsg = 'Not connected to VNA. Please connect before
                reading and writing to it.';
            msgbox(errormsg, 'Error', 'Error');
            error(errormsg);
        end
    end
    function Flush(vna)
        arguments
            vna (1,1) VNA
        end
        % Make sure we're connected.
        vna.CheckConnection();

        flush(vna.SCPI);
    end
    % Write the given command to the VNA.
    function WriteLine(vna, command)
        arguments
            vna (1,1) VNA
            command (1,:) char
        end
        % Make sure we're connected.
        vna.CheckConnection();

        % Write the given command to the interface.
        writeline(vna.SCPI, command);
    end
    % Write the given command to the VNA, then read the returned data.
```

```
        function data = Read(vna, command)
            arguments
                vna (1,1) VNA
                command (1,:) char
            end
            % Make sure we're connected.
            vna.CheckConnection();

            % Write the given command to the interface.
            write(vna.SCPI, command);
            try
                % Read the returned value.
                data = readline(vna.SCPI);
            catch err
                if(strcmp(err.message, 'Failed to read data from the
                   instrument due to a driver error.'))
                    uiwait(msgbox({err.message, 'Please unplug the USB for
                        the VNA, and plug it back in.', 'Then disconnect
                       and reconnect in the GUI.'}, 'Error', 'Error'));
                end
                rethrow(err);
            end
        end
        % Read a line from the interface.
        function data = ReadLine(vna)
            arguments
                vna (1,1) VNA
            end
            data = readline(vna.SCPI);
        end

    end
    methods
%         function data = ReadData(vna, channel, datatype)
%             arguments
%                 vna (1,1) VNA
%                 channel (1,1) double
%                 datatype (1,:) char
%             end
%             % Tell the machine to give us the data from the given
    channel.
%             command = sprintf('calculate%i:data? %s', channel, datatype)
    ;
%             write(vna.SCPI, command)
%
%             data = read(vna.SCPI);
%         end
        function ClearMeasurements(vna, channel)
            arguments
                vna (1,1) VNA
                channel (1,1) int32
            end
            command = sprintf('calculate%i:parameter:delete:all', channel)
                ;
            vna.WriteLine(command);
        end
```

```
function AddMeasurements(vna, channel, measurements)
    % Accepts any number of arguments that each specify a
       measurement.
    % measurements: 'S11', 'S12', 'S13', ..., 'S43', 'S44'
    %                'A', 'B', 'C', 'D', 'R1', 'R2', 'R3', 'R4'
    %                'A/B', 'A/C', 'A/D', 'A/R1, ..., 'R4/R2', 'R4/
       R3'
    arguments
        vna (1,1) VNA
        channel (1,1) int32
    end
    arguments(Repeating)
        measurements (1,:) char
    end
    % Remove the division bar from the measurements.
    % Ratioed measurements are defined in the VNA as AR1 for A/R1.
    measurements = strrep(measurements, '/', '');
    for(imeas = 1:length(measurements))
        measurement = measurements{imeas};
        command = sprintf('calculate%i:parameter:define ''meas_%s'
            ',%s', channel, measurement, measurement);
        vna.WriteLine(command);
        command = sprintf('display:window1:trace%i:feed ''meas_%s'
            '', imeas, measurement);
        vna.WriteLine(command);
    end
end
function startfrequency = GetStartFrequency(vna)
    % Get the start frequency in Hz currently set on the VNA.
    startfrequency = str2num(vna.Read('sense:frequency:start?'));
end
function SetStartFrequency(vna, startfrequency)
    % Set the start frequency in Hz on the VNA.
    command = sprintf('sense:frequency:start %i', startfrequency);
    vna.WriteLine(command);
end
function stopfrequency = GetStopFrequency(vna)
    % Get the stop frequency in Hz currently set on the VNA.
    stopfrequency = str2num(vna.Read('sense:frequency:stop?'));
end
function SetStopFrequency(vna, stopfrequency)
    % Set the start frequency in Hz on the VNA.
    command = sprintf('sense:frequency:stop %i', stopfrequency);
    vna.WriteLine(command);
end
function frequencypoints = GetFrequencyPoints(vna)
    % Get the number of frequency points currently set on the VNA.
    frequencypoints = str2num(vna.Read('sense:sweep:points?'));
end
function SetFrequencyPoints(vna, frequencypoints)
    % Set the number of frequency points on the VNA.
    command = sprintf('sense:sweep:points %i', frequencypoints);
    vna.WriteLine(command);
end
function frequencystep = GetFrequencyStep(vna)
    % Get the distance between frequency points in Hz currently
```

```
                set on the VNA.
            frequencystep = str2num(vna.Read('sense:sweep:step?'));
        end
        function SetFrequencyStep(vna, frequencystep)
            % Set the distance between frequency points in Hz on the VNA.
            command = sprintf('sense:sweep:step %i', frequencystep);
            vna.WriteLine(command);
        end
        function intermediatefrequencybandwidth = GetIFBandwidth(vna)
            % Get the IF bandwidth in Hz currently set on the VNA.
            intermediatefrequencybandwidth = str2num(vna.Read('sense:
                bandwidth?'));
        end
        function SetIFBandwidth(vna, intermediatefrequencybandwidth)
            % Set the IF bandwidth in Hz on the VNA.
            command = sprintf('sense:bandwidth %i',
                intermediatefrequencybandwidth);
            vna.WriteLine(command);
        end
        function rfpower = GetRFPowerState(vna)
            % Gets the state of the RF power (on/off) currently set on the
                VNA.
            rfpower = str2num(vna.Read('outp:state?'));
        end
        function SetRFPowerState(vna, rfpowerstate)
            % Sets the state of the RF power (on/off).
            command = sprintf('outp:state %i', rfpowerstate);
            vna.WriteLine(command);
        end
        function fostate = GetFrequencyOffsetState(vna)
            % Gets the state of the RF power (on/off) currently set on the
                VNA.
            fostate = str2num(vna.Read('sense:fom:state?'));
        end
        function SetFrequencyOffsetState(vna, fostate)
            % Sets the state of the RF power (on/off).
            command = sprintf('sense:fom:state %i', fostate);
            vna.WriteLine(command);
        end
    end
    %% Private methods, to ensure they are not called by user.
    methods(Access = protected)
        % Do not call this function directly. Use VNA.Get().
        function vna = VNA()
            global VNA_SCPI; %#ok<GVMIS>
            if(~isempty(VNA_SCPI))
                vna.SCPI = VNA_SCPI;
                vna.isconnected = true;
            end
        end
    end
end
```

Listing D.11: Example code VNA