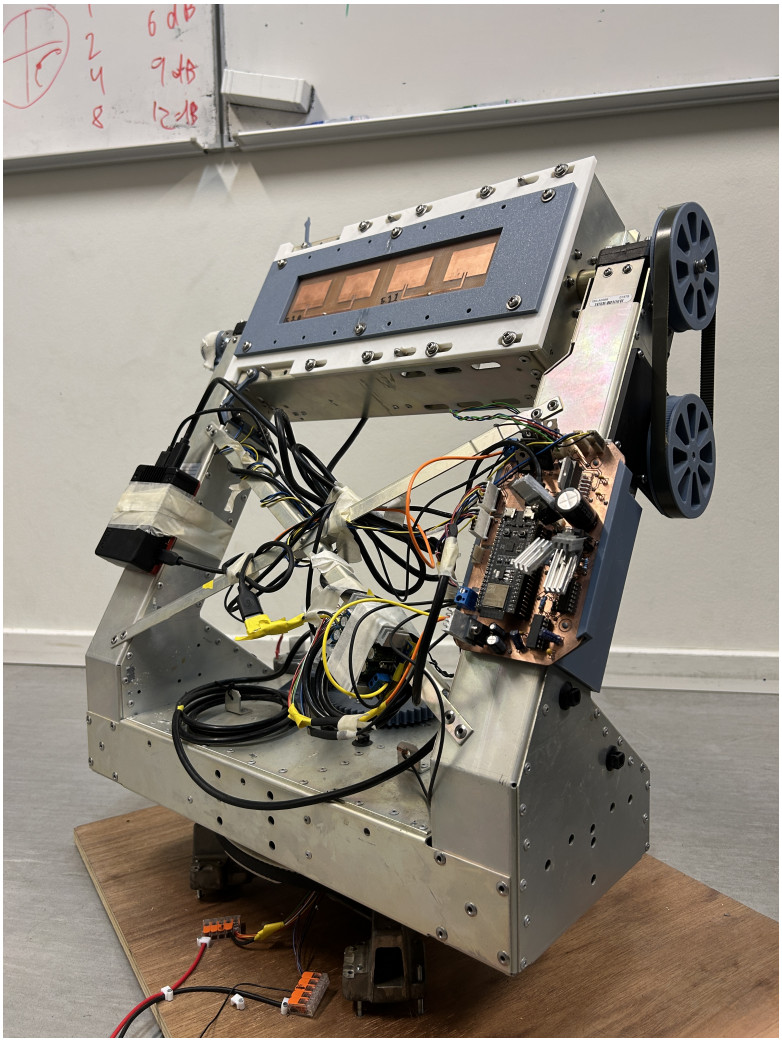
Scan & Track System for Recreational Drones



ESD5 125 - P5 Project - 2025

Emil Beier Emil Therkelsen Helge Clausen

Janus N. Gudiksen Jeppe Pedersen Noah H. D. Nielsen

Institute for Electronic Systems, winter, 2025

P5 project | Electronics and systems design

AALBORG UNIVERSITY

STUDENT REPORT

Title:
Scan and track system for recreational drones

Theme:
Digital and analog systems interacting with the surroundings

Project Period:
Fall Semester 2025

Project Group:
ESD5-125

Participant(s):
Emil Beier - Student no.: 20235025
Emil Therkelsen - Student no.: 20234115
Helge Clausen - Student no.: 20233374
Janus Nørgaard Gudiksen - Student no.: 20234714
Jeppe Pedersen - Student no.: 20233473
Noah Haakon Degn Nielsen - Student no.: 20230449

Supervisor(s):
Jesper Ødum Nielsen

Page Numbers: 150

Date of Completion:
December 21, 2025

Abstract:

Drones trespassing in restricted airspace is an increasing problem, especially in and around airports. This problem requires countermeasures that can detect and deal with these drones. Using recreational drones as a reference, such as those a consumer might buy, the aim of this project was to design a system, that can track and point at drones in flight. To this end an analysis is conducted on how drones communicate wirelessly, where drones are allowed to operate, and how drones can be detected. A use case is set up within this project, based on Danish legislation surrounding drones, which limits the flight of a drone within 8 km of the landing strip. A system prototype is made, consisting of a 'Two Axis Rotational Pedestal' (TARP), that allows for mechanical steering of the system. Mounted on it is a two-element antenna array, made from two microstrip patch antenna subarrays, to receive within the 2.4 GHz ISM band. A SDR bladeRF 2.0 micro A9 that functions as the front end, samples incoming signals and passes them to a digital delay-and-sum beamformer running in Python on a Raspberry Pi 5. Due to individual submodules not attaining specifications, and errors in testing, it is concluded that the system is too incomplete to be a solution, to the problem statement. To this end a discussion on which areas should be developed further and the viability of the concept, is also detailed.

Preface

This project has been composed by group 5-125, during the 5th semester of Electronics and Systems Design on Aalborg university. The main semester theme is "Digital and analog systems interacting with the surroundings".

Throughout this project all available decimals are used for calculations, but for brevity only the decimals deemed significant are shown.

The group would also like to give special thanks Sigurd Sándor Petersen, from the APMS lab at AAU for helping selflessly, providing additional feedback and talking in our favor, allowing us to lend equipment.

Emil Beier
<ebeier23@student.aau.dk>

Helge Clausen
<hclaus23@student.aau.dk>

Janus Nørgaard Gudiksen
<jgudik23@student.aau.dk>

Jeppe Pedersen
<jp23@student.aau.dk>

Noah Haakon Degn Nielsen
<nhdn23@student.aau.dk>

Emil Therkelsen
<etherk23@student.aau.dk>

Contents

Chapter 1

Introduction

The vast expansion of commercial drones has lead to an increased number of disturbances in air traffic, creating unsafe conditions in the airspace. For example, a hobby drone crashed into a Super Scooper firefighting plane during the Palisades fire in Los Angeles, despite Federal Aviation Administration flight restrictions. The collision forced the plane to land, endangered the two pilots and stopped its participation in the firefighting operations [1]. This is not an isolated event, London Gatwick Airport reported a loss of 50 million pounds due to a drone entering the airspace, disrupting flight operations [2]. Similar incidents have also occurred in Denmark, at Aalborg Airport, increasing the drone disturbances from 5 in 2022 to 40 in 2023 [3].

The increase in drone disturbances has caused the Danish government to take action, instating new laws restricting drone flights. In 2024, a law was enacted in Denmark, further restricting drones near airports, military zones and areas with critical infrastructure, such as hospitals, telecommunication stations, major transportation hubs, etc. [4]. Despite this in January 2025, 20 unknown drones were reported seen above Køge Havn for several hours [5]. This highlighted a critical issue, even though it is illegal to fly in certain areas, actionable consequence cannot be taken if the drone pilot cannot be identified. The harbor lacked the necessary defenses to mitigate a potential act of espionage or terrorism. This act along with previous disruptions has lead the transport minister in Denmark to propose a new law that allows for airports, military and areas with critical infrastructure to neutralize drones that are non-cooperative, which will be active from January 1st, 2026 [6]. Non-cooperative drones are defined as drones that are used for illegal activity whether deliberately or unintentionally [7].

The increase in drone disturbances, clearly outlines a rising issue. The lack of accountability for unidentified drone pilots, creates two challenges: identifying the drone pilot and neutralizing the drone. Since the problem primarily lies within protecting restricted airspace from drone disturbances, it is ideal to first detect and neutralize the drones. Identifying the drone pilot could be a secondary response. In order to help address this issue, a track and point system is needed. This project will focus on how to detect and track commercial drones flying in restricted airspace at Danish airports, and not on neutralizing them. Initializing the project involves analyzing commercial drones, aviation laws at airports and detecting systems in relation to drones.

The following problem analysis will go in depth with relevant topics required to develop a solution which can counter these drones. The analysis should give a basis for a specific use case which will be used to define a set of specifications used in the development.

Chapter 2

Problem Analysis

The purpose of this project, as mentioned in the introduction, is to develop a system that can detect and track drones in and near airports. This requires knowledge of the functionality and classification of drones, detection methods, and relevant regulations regarding airports.

2.1 Drones

In this project, the focus is on recreational drones; recreational use is defined as personal interest, hobby and enjoyment [8]. This section determines the specific size and weight. Furthermore, drone speeds are considered to determine how fast a tracking device needs to be. A critical point of this section is to look into which frequencies are utilized for communication between the drone and the drone controller.

2.1.1 Drones Sizes and Speed

Table 2.1 presents an overview of the different drone categories, highlighting their weight, size, and typical use. Since no official standard for drone categories and their specifications exists, the following table has been made by merging data from different sources.

Category	Weight	Size / Length	Typical Use Cases
Nano	$< 0.25\,\text{kg}$	$< 0.3\,\text{m}$	Indoor flying, amateur photography
Micro	0.25 - 2 kg	0.3 - 0.5 m	Indoor/outdoor flying, racing, light payloads
Small	2 - 25 kg	0.50 - 2 m	Agriculture, mapping, industrial inspections
Medium	25 - 150 kg	2 - 10 m	Military use, surveillance, long-range commercial operations
Large	$> 150\,\text{kg}$	$> 10\,\text{m}$	Heavy-lift cargo, strategic military operations

Table 2.1: Drone Classification by weight, size, and typical use [9] [10] [11] [12].

In Table 2.1, it is shown that the typical use of the categories medium and large drones is mainly military. Since military use does not fit the scenario described in the Introduction 1 nor the description of recreational drones mentioned at the start of Section 2.1, medium and large drones are not considered recreational drones. The nano and micro drones are typically used by amateurs, because operation requires either no license or one that is easily obtained, thereby classifying them within the category of recreational drone [12]. Small drones are used more in industrial settings, however they are still available to private citizens in Denmark with

a license [12], so for this project they are classified as recreational drones, to avoid overlooking possible scenarios. This project focuses on nano, micro, and small drones, as these types are suitable for the scenario described in the Introduction 1 and their typical use aligns with recreational use of drones.

It is crucial to get an estimate of how fast these kinds of drones can fly. Some recreational drones have a max speed up to 290 km/h, these are of the type First-person view (FPV) drones, often used for racing due to their high speed [13]. They are much faster than typical recreational drones, which usually have a max speed between 55 and 95 km/h [13].

Therefore recreational drones has been determined to have weight of up to 25 kg, a size of up 2 m, and a max speed of 290 km/h [13].

2.1.2 Drone Signal and Antennas

Drones communicating via radio-frequency signals use antennas to transmit and receive them, which enable communication between the control unit and the drone. There are two primary types of antennas used in drones. These are omnidirectional antennas and directional antennas [14]. Omnidirectional antennas are widely favored in recreational drones, providing a 360 ° coverage in the horizontal plane [14], often limited in the vertical plane, making their radiation toroidal [15]. Drones used in industrial settings often require a stable connection over longer distances, meaning they often use direction antennas [14]. As the name implies, directional antennas transmit or receive in a specific direction, which can increase the range and reduce interference [14]. FPV racing drones often use patch antennas, which are a type of directional antenna, because of their high-gain potential. Their high gain strengthens the signal in the desired direction, providing the stable link needed for real-time video [14]. The radio frequency (RF) output power is limited by law for drone communication, it states that the output power should be equal to or less than 20 dBm [16].

2.1.3 Drone Transmission Frequencies

This section focuses on the most common frequencies used for drone communication. Some communication types between the drone controller and the drone include steering and video transmission. For steering, the most commonly used frequency band is 2.4 GHz, as it offers a good balance between obstacle penetration, sufficient range, and compatibility with a wide range of devices [14]. 2.4 GHz is a free-to-use frequency band, which spans from 2400 - 2500 MHz [17]. Within this band the 2400 - 2483.5 MHz frequency range, is used for data transmissions [16]. This band is also one of the most used in drone communication [18]. The rest, 2483.5 - 2500 MHz, is allocated for satellite systems and medical devices [19] [20]. Other most used frequency bands by drones are 5.8 GHz, 1.3 GHz, 1.2 GHz, 900 MHz, and 433 MHz each with there own intervals just like the 2.4 GHz band. However 1.3 GHz to 433 MHz are mostly used in special applications, for example, telemetry data transmissions used in long-range drones, and agriculture [14]. A lot of recreational drones have video feed, which typically utilises the frequency bands 2.4 GHz and 5.8 GHz [14]. Some more advanced drones use 4G and 5G networks for long-range video transmission [14]. The 2400 - 2483.5 MHz frequency band is chosen as the operating frequency band for this project.

2.2 Drone Detection

This section covers the different methods of detecting drones in the airspace. The different detection methods that are covered here are radar, radio frequency (RF) analysis, acoustic sensing and optoelectronic sensing, as these are the four main ways of detecting drones [21].

Optoelectronic and acoustic detection methods are often used to improve the accuracy of an already existing solution, as the main detection method of most solutions is radar and/or RF analysis [22]. Therefore, is the optoelectronic and acoustic detection methods are only briefly outlined and not analyzed further. Optoelectronic sensing, as the name implies, is based on visual contact, and relies on cameras to detect the drones. As such, this method is highly limited by visual conditions (buildings, fogs, heavy rain etc.) or by the quality of the optic sensor [22]. Acoustic sensing functions by detecting soundwaves emitted by drones using microphones. Because it relies on audio, background noise from the surrounding area of the sensor, has a high impact on the precision and range of this detection method [22].

2.2.1 Detection by Radar

Detection by radar is a method of locating objects in and out of motion. A typical pulse radar system works by transmitting radio waves in pulses, and listening for reflected echo signals, which are then processed to measure distance from the base station to the target [23]. Today radar systems are numerous and have many distinguishing features from one another, so a comparison of said different systems is necessary, to know which systems are applicable in drone detection [23].

Radar systems can generally be categorized by one of two functioning principles. Either by emitting pulsed waves or continuous waves. Besides this categorization, radars are also distinguished by frequency band, and intended function, for example weather radars [23]. As of writing, the two most relevant radar types for drone detection are frequency modulated continuous wave (FMCW) radar, and Pulse-Doppler radar [21]. FMCW radar as the name implies, uses frequency modulation, this could be in the form of periodic chirps, to measure range and velocity [24] [23]. A Pulse-Doppler radar works by pulsing a signal, and analyzing the frequency shift of the echo signal caused by the Doppler effect, to determine range and velocity of a target [23]. Both are well documented methods of detecting drones, but are still susceptible to general difficulties of the technology, when it comes to distinguishing drones. An advantage of using radar over RF detection, is the ability to detect autonomous drones, as it does not need to intercept data transmissions.

What makes drones particularly difficult for radar, is that drones vary significantly in size. Smaller drones are generally harder to detect, which is only compounded by low altitude flying, and line of sight problems [25]. Another problem is that smaller drones and birds are easily confused with one another, as they have similar radar cross sections. This has spawned another way of processing Pulse-Doppler radars, examining so called micro-Doppler signatures [26]. By using micro-Doppler signatures it is possible to effectively distinguish birds and quadcopter drones, at least in the K-band (18 to 27 GHz) and in the W-Band (75 to 110 GHz), and if neural networks are utilized - the X-band (8 to 12 GHz) [21].

2.2.2 Detection by Intercepting RF Transmissions

Detecting drones by RF signals works by intercepting transmission signals, to or from a target drone, and analyze the signals. The information can be analyzed in two main ways that are covered below, decryption method and beamforming method. A concrete example of this, is a drone that transmits a signal, might be an update, and might be a response to a command. The signal is then picked up by a detection unit using RF analysis, which processes the signal, and tracks the position of the drone.

Section 2.1.2 explains that some recreational drones transmit signals omnidirectionally as they have no information about the drone controller location. Obtaining the location this way is called the information extraction method. The transmitted information from the drone to the drone controller varies between drone models and brands [22]. This leads to the different methods of detecting using RF signals. If the GPS location is included in the information transmitted, then by using an RF sensor, to extract the relevant information (amplitude, phase and frequency), and a lookup table to compare to common communications protocols. The GPS location can be extracted and tracked [21].

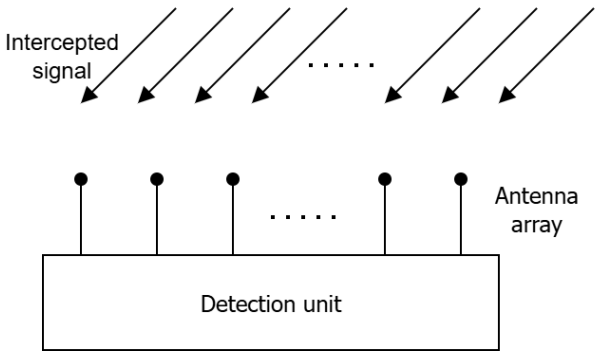


Figure 2.1: Figure showing concept behind drone range estimation via the beamforming method using antenna arrays.

In some drone models, the GPS location is not included in the transmitted data [22]. If this is not the case, a different detection technique such as beamforming is required. Beamforming uses an antenna array, and the time difference of signals received on each array element to estimate the direction of a signals' origin [27]. This can be seen on Figure 2.1.

The main disadvantage with the decryption method is that the specific communication protocol is not known, this means that additional software is needed to decrypt the signal. The beamforming method requires one array in the horizontal plane, and one in the vertical plane to detect the direction of the signals origin in both a horizontal and vertical direction. For both of these methods, there is a problem with drones that neither send RF (ex. cabled drones and programmable drones) or has a directional transmitter.

2.3 External Restrictions

The focus for this section is to cover the general requirements that a solution placed in or close to an airport must adhere to. Since this project is based in Denmark, most of the laws and regulations are outlined within regulations from the European Union (EU).

2.3.1 Regulations

There are a few different authorities relevant in Denmark, the Ministry of Transportation, which is in charge of legislation, whereas municipalities and Trafikstyrelsen, who are in charge of daily operation [28]. The only Danish laws regarding installations and construction in and near airports, is that it cannot have a height above 25 m near runways [28]. The other law is that any construction within 25 m of runways needs to be approved by Trafikstyrelsen [29]. Another regulation concerns restricted airspace for drones near military airports, where a no-fly zones for drones are set at a horizontal distance of 8 km near the airports [30]. There is some leeway for licensed drone operators based on the altitude of the drone, which allows operators to fly as close as 3 km at an altitude of up to 40 m, and until 2 km at an altitude of maximum 30 m [30]. Limited drone operation can however take place here with the correct license and permission from the correct authorities.

The regulations set forth by the EU impose more restrictions on the form and function of installations in airports. Some of these restrictions include limits to the visual appearance. These limits state that installations cannot have confusing lights or be too reflective [31]. Another essential regulation states that frequencies emitted by instillation may not interfere with any equipment aboard aircraft or any other systems critical for flight operations [32]. How much this is is unspecified.

2.3.2 Avoiding Interference

Electronic installations, as mentioned in Section 2.3.1, are not allowed to emit radiation that can disturb essential equipment in airports and aircraft. The aim of this section is to outline how this is avoided.

A wide range of frequencies is utilized within an airport, including 2.4/5 GHz for Wi-Fi [33], 23.35 GHz for flight safety [34], among others. A comprehensive list of all allocated frequencies and their respective significance is not considered essential, given the vast number of frequencies used. If a specific frequency becomes of interest, it will be found in a lookup table made by the European Communications Office (ECO) [35]. The focus in this context is therefore placed on the allowed levels of electromagnetic interference (EMI) outside a designated frequency band.

In general, a device may not introduce EMI so strong that other devices stop working as intended [36]. However, due to the large number of devices, this does not allow for a single concrete threshold. Instead, the standards from the EU specify allowed limits that should be used through the specifications outlined in ETSI 301-489 [37]. Likewise, the acceptable EMI outside the designated frequency range is not one threshold, but is often below -47 dBm [38].

2.4 Summary and Problem Statement

In Section 2.1, an analysis shows the variation of drones that exist. Recreational drones have through this analysis been classified as nano to small drone with speeds up to 290 km/h. These drones typically use an omnidirectional antenna, and communicate on the frequency band from 2400 to 2483.5 MHz.

For the detection methods, there are two general ways to detect drones, radar or RF interception. For radar detection of smaller drones, it is necessary to use Pulse-Doppler radars, specifically looking at micro-Doppler signatures, which are used to distinguish birds from quadcopter drones. In RF intereption, there are two ways, decryption and direction of arrival. Since the goal of this project is to detect and track drones, the decryption method is limited because it needs a GPS location within the message to be able to track the drone. Since that is not a certainty, it will not be considered any further for this project. Therefore, the two optimal ways of detecting drones are Pulse-Doppler radars and direction of arrival.

The external restrictions Section 2.3 states that there is a no-fly zone 8 km radius around the airports in Denmark for drone operators without a license. Additionally, any installation at the airport cannot exceed a height of 25 m near the runways with explicit permission. It should also be noted in this regard, that airports use many different frequencies, some of which are critical for flight safety. RF emitting solutions should therefore be designed with great care to not interfere in these frequencies.

This problem analysis has lead to an understanding of each element of what is needed to develop a solution. This then leads to the following problem statement.

How can a detection and tracking system be realized for recreational drones entering the restricted airspace of Danish airports?

Chapter 3

Proposed Solution and Specifications

3.1 Use Case

Before a solution and specifications can be put forward, a use case must first be determined. This is required, as solving the entire problem statement for all recreational drones is a substantial challenge. Consequently, a specific drone is selected for the use case. The chosen drone is the DJI Matrice 210 RTK, as it is made available to the project. Both its size, weight and speed is within the category "recreational drone" set forward in Section 2.1. As stated in Section 2.3 the restricted fly-zone around an airport is defined in three layers, with a complete no-fly zone in $0 - 2\,\text{km}$ range of the runway, and two partial allowed fly-zones in $2 - 3\,\text{km}$ and $3 - 8\,\text{km}$ range. Each zone has a different severity level therefore the three zones have been named *kill*, *tracking* and *detection* zone. As soon as the drone enters the kill zone, it has to be shot down, which is why it needs to track the drone in the previous zone. An illustration of this is available in Figure 3.1. The prototype needs to be precise enough at 2 km (kill zone) to shoot down the drone. To simplify the use case, this project will only focus on the scenario where one drone flies within the restricted airspace. Even though the radius of the three zones depends on the runway of an airport, the prototype development will simplify this to a circle as illustrated in Figure 3.1.

Besides the perimeter, a height must also be determined. As the maximum flying height of the chosen drone is 2.5 km, this is the height of the detection and tracking cylinder [39].

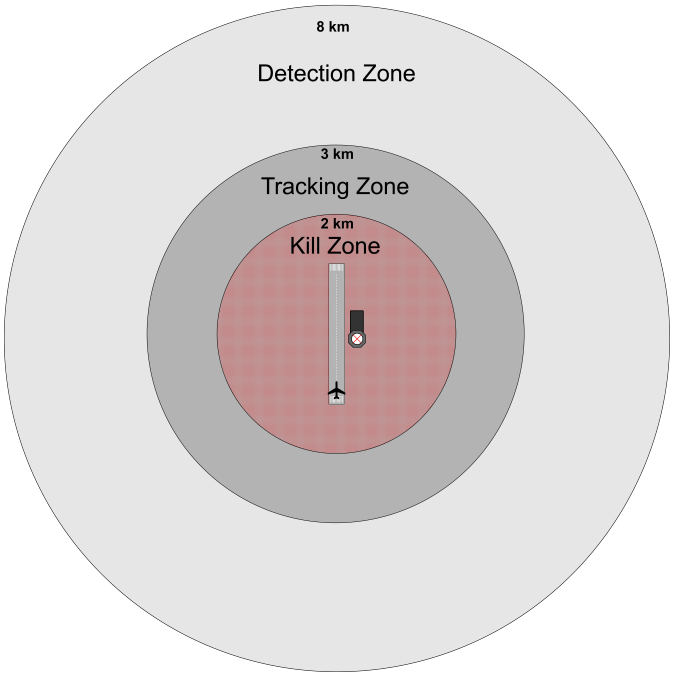


Figure 3.1: Illustration of the drone flight zones around an airport.

3.2 Proposed Solution

This section presents different potential solutions to the use case, along with a chosen solution.

The most apparent solution, given the problem analysis in Chapter 2, is to use radar or intercept the RF transmission. As stated in Section 2.2, both of these technologies can detect and track drones, however they each operate in significantly different ways. Two primary difference is that RF interception only receives, while the radar both transmits and receives. The other is the vastly different way of which they process the data.

One of the main challenges of implementing a radar system is the complexity required to differentiate between small drones and birds as mentioned in Section 2.2.1. Another problem is that strict guidelines are enforced when transmitting electromagnetic waves in or near an airport, see Section 2.3. With regards to the interception of RF signals, the main challenge is that the drone uses the same frequency bands as WIFI signals, the 2.4 GHz band, see Section 2.1.3. This introduces interference, which potentially could be interpreted as a drone. As radar requires both transmitting and receiving RF signals, while RF interception only requires receiving, it is seen as an extension of RF interception. Figure 3.2 shows a simplified block diagram of a radar system.

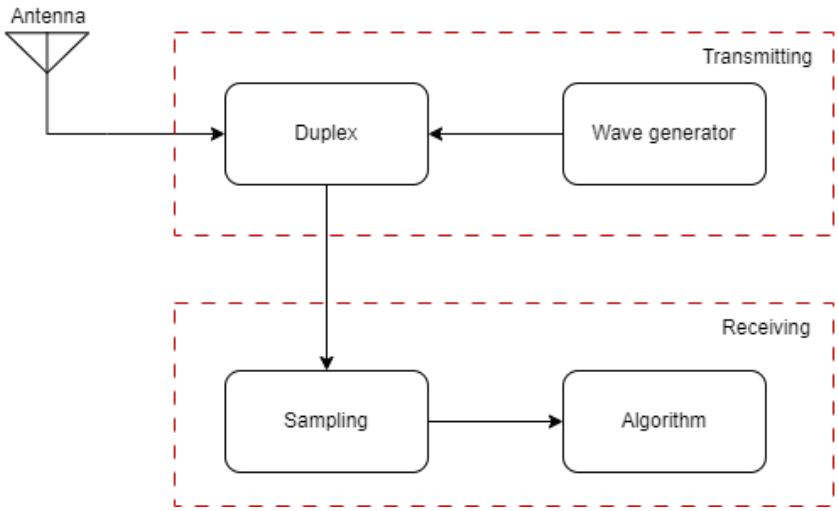


Figure 3.2: Block diagram of a radar system showing a simplified version of the parts of the circuit.

In the block diagram, the duplexer switches between the transmitting and receiving signals. The entire transmission block is only relevant for radar, as RF interception only receives signals. Due to this, the solution using RF interception is chosen as the one to develop, as it is a less complex starting point compared to radar. Although radar provides extra capabilities as touch upon in Section 2.2, RF interception shares several of the fundamental elements while avoided the high system level complexity.

With respect to the placement of the solution two options are viable, either the solution is one unit placed centrally, or multiple placed in different locations, each covering a smaller area. Figure 3.3 contains an illustration of this.

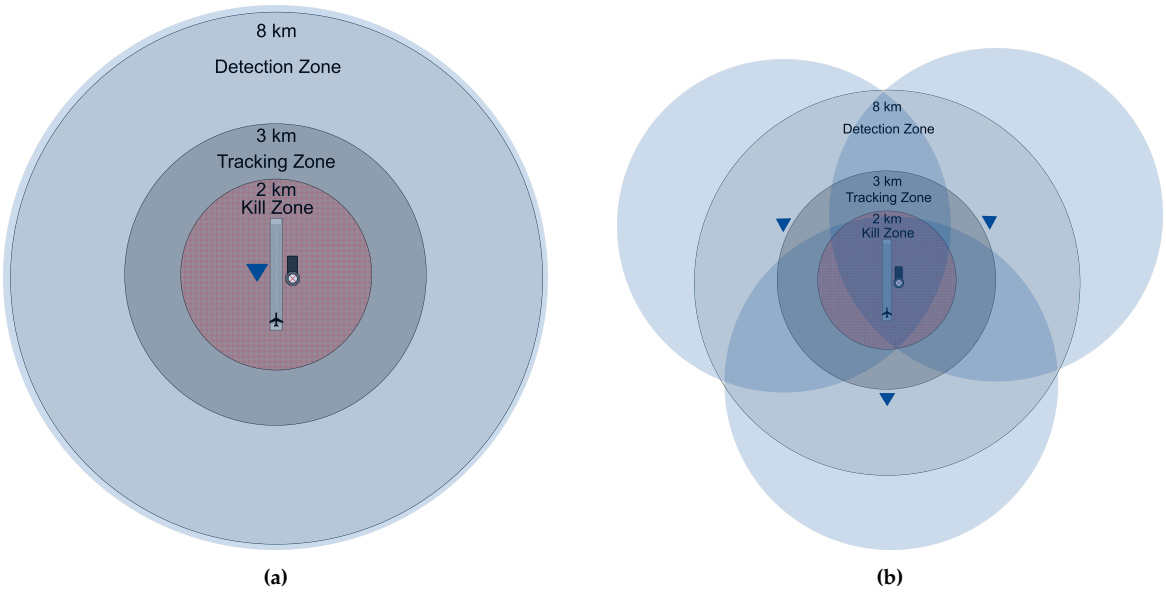


Figure 3.3: Illustration of the two possible placement options. The blue triangles represent the placement of the prototype. The blue zone represents the range of the detection zone for the prototype. **(a)** is the single unit centrally placed. **(b)** is the multiple unit with the detection area of each marked.

With the placement of multiple units, it is potentially a more expensive solution, but it might allow for detecting the distance to the drones through triangulation. For triangulation to function properly, the drone needs to have an omnidirectional antenna. This is necessary for comparing the measurement of signal strength at each unit. If the transmitted signal is not identical in all 3 directions, a wrong location will be reported. Due to this major oversight a single unit solution is chosen to continue development of. This however lacks a distance measurement device, which must be implemented, but will not be the focus for this project.

3.2.1 Explanation of Solution

A block diagram has been constructed to ensure a better overview of what must be developed, which can be seen on Figure 3.4. This diagram illustrates a signal transmitted from a drone, received by an antenna, digitized by an RF front end, where an algorithm estimates the angle of arrival, that the device - Two Axis rotational Pedestal (TARP) - points towards using a control system.

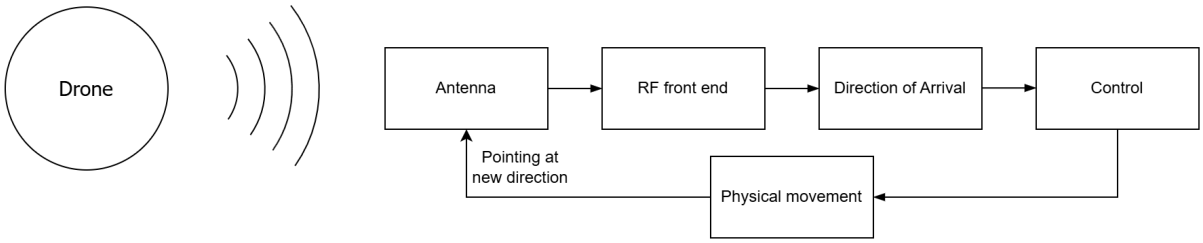


Figure 3.4: High abstraction block diagram of the chosen solution for the prototype. An antenna block is present where the radio frequency signals are received. A RF front end where the signals must be sampled. A direction of arrival block where the direction of the signals is found. Lastly, a Control block that moves rotates the prototype to a new direction where the antennas will receive a new signal.

The solution needs to operate in two different states, scanning and tracking. The detection zone is for scanning, and the tracking zone is for tracking. In the kill zone, it should be shot down, but as this is not a focus of this project, it is interpreted as an extension to the tracking zone. In the scanning state, it needs to be able to detect the different drones and ideally the distance to it, however as described above, this is not a focus for this project. As soon as a drone enters the tracking zone, the prototype switches to the second state, tracking, and pointing to the drone. This is done, as it gives a faster updating location of the drone. The precision of each state is therefore the same, as only the refresh rate is changed.

Scanning State

An assumption must be made with regard to the beam width of the antenna and the algorithm estimating the direction of arrival. It is assumed that at any given position, the antenna combined with the algorithm can detect drones in the area $\pm 60\,^{\circ}$ from the center point in horizontal direction. The prototype must therefore scan at three different directions of 360°, see Figure 3.5. Likewise, it is assumed that in an elevation of $\pm 45\,^{\circ}$ is detectable. The elevation must therefore be locked at 45° with regard to the horizon, which would give the solution 90° view of the airspace.

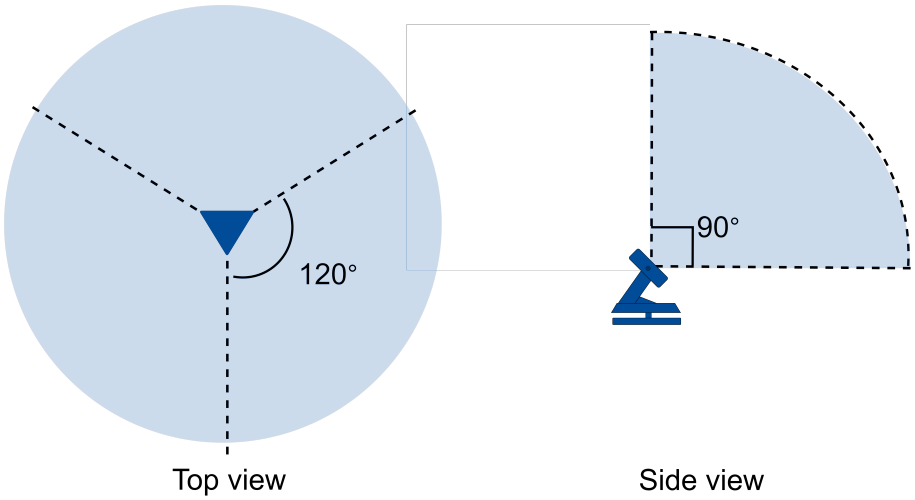


Figure 3.5: Illustration of the detection hemisphere cut into three azimuth slices and tilt locked at 45°, resulting in a view angle of 90°.

Tracking State

When the scanning state detects a drone that is within the tracking zone, the prototype must automatically switch to the tracking state. Here, the TARP should move to point directly at the drone, both in the azimuth and elevation plane. A new measurement is done, and the TARP moves again. This continues while the drone remains within the tracking zone. As soon as the drone enters the kill zone, it will be shot down, which is not a focus for this project.

3.3 Specifications

This section presents the different specifications required for the solution, chosen in Section 3.2.

3.3.1 Mechanical and DSP Specifications

This section involves setting forward the mechanical and digital signal processing (DSP) specifications based on the use-case defined in Section 3.1. This section describes multiple zones, with the important one for these specifications being the tracking zone from 2 to 3 km. This is due to it being the zone that sets the most strict requirements. The solution must therefore be able to detect a drone entering this zone, before it reaches the 2 km mark. As the drone has a max speed of 82.8 km/h [39], a full scan (360 °) must be completed within the time of t_{detect} (Rounded down, to ensure spec is fulfilled).

$$t_{detect} = \frac{d}{v} = \frac{3\,\text{km} - 2\,\text{km}}{82.8\,\text{km/h}} \approx 0.012077\,\text{h} \approx 43.47\,\text{s}$$

It is assumed that a drone can be detected in one try/scan. It is also expected that the mechanical part requires most of this time (t_{detect}), compared to the DSP. A 80/20 split is therefore implemented resulting in $t_{mec} \approx 34.78\,\text{s}$ and $t_{dsp} \approx 8.69\,\text{s}$. It should be noted that this is the

time for a full scan of the airspace.

With regards to the second demand, that being the solution is accurate enough to neutralize a drone at 2 km, then it is more difficult to specify as the method of interception is not determined. The objective is therefore to constrain the total uncertainty such that it remains fully contained within the physical dimensions of the drone. As the drone is approximately 80 x 80 x 40 cm, assuming that the RF transmitter is located in the center of the drone, an accuracy of 40 cm horizontally and 20 cm vertically at 2 km is then required [39]. This results in an accuracy in degrees of:

$$\theta_h = \frac{distance}{circumference} \cdot 360 = \frac{0.4\,\text{m}}{2 \cdot 2000\,\text{m} \cdot \pi} \cdot 360 \approx 11.46\,° \cdot 10^{-3}$$

$$\theta_v = \frac{distance}{circumference} \cdot 360 = \frac{0.2\,\text{m}}{2 \cdot 2000\,\text{m} \cdot \pi} \cdot 360 \approx 5.73\,° \cdot 10^{-3}$$

It should be noted that this is an extremely high accuracy, as it is a worst-case scenario. It might be lowered when an actual interception method is chosen. Once again, a 80/20 split is used with the same reasoning, result rounded down to ensure being accurate enough. The calculated numbers, and a summary of the specifications set forward in this section is available in Table 3.1.

Parameter	Specification
Mechanical detection time	34.78 s
DSP detection time	8.69 s
Mechanical accuracy horizontal	$9.17\,° \cdot 10^{-3}$
DSP accuracy horizontal	$2.28\,° \cdot 10^{-3}$
Mechanical accuracy vertical	$4.58\,° \cdot 10^{-3}$
DSP accuracy vertical	$1.14\,° \cdot 10^{-3}$

Table 3.1: Summary of mechanical and DSP specifications.

Test of specifications

While each specification in Table 3.1 could be tested individually, this approach is unnecessarily cumbersome with no real benefit. Instead, the combined specification is tested, as the mechanical/DSP split is an estimate, with the overall compliance being the critical factor.

Detection time
The detection time of a drone is reliant on the time to perform a full 360 ° scan. This test therefore requires to activate the prototype in *scanning* mode and measure the time before it is done with a full scan. All signals should be attenuated enough for it to stay out of *tracking* mode.

Accuracy
The accuracy of the device can be tested by placing a source at a known angle with regards to the prototype. Then let the prototype enter the *tracking* mode. When the homing is stabilized, the difference between the estimated angle and actual angle should be calculated. It should be noted that this test must be done twice, once for horizontal and once for vertical.

3.3.2 Antenna Specifications

This section will present the specifications for the antenna. These specifications include the resonant frequency, frequency range, bandwidth, impedance matching, as well as the range and gain.

The minimum frequency range 2400 to 2483.5 MHz is chosen based on Section 2.1.3, making the bandwidth at least 83.5 MHz. The resonant frequency is 2441.75 MHz, which is the center of this frequency band. The resonant frequency is mainly a design parameter, so no tolerances are applied to this number. The return loss (RL), when defined as the ratio of reflected to incident power expressed in decibels, can be used interchangeably with the S11-parameter, also in decibels. By this definition, it is related to the magnitude of the reflection coefficient $|\Gamma|$ by Equation (3.1) [40] [41].

$$RL = 20 \cdot \log_{10}|\Gamma| \quad \text{[dB]} \quad (3.1)$$

The magnitude of the reflection coefficient $|\Gamma|$ can then be calculated as.

$$RL = 20 \cdot \log_{10}|\Gamma| \rightarrow |\Gamma| = 10^{\frac{RL}{20}}$$

The percentage of reflected power can be found by Equation (3.2) [42].

$$\text{Reflected power} = 100 \cdot |\Gamma|^2 \quad [\%] \quad (3.2)$$

A return loss of 0 dB means all of the signal is reflected, -10 dB corresponds to a reflection of 10 percent of the signal, and -20 dB means only 1 percent of the signal is reflected. However, -10 dB is generally considered acceptable as 90 percent of the signal is transmitted to the antenna, and a return loss lower than -10 dB only yields marginally better performance [43][44]. Therefore, -10 dB return loss is used to define the bandwidth of the antenna [40], meaning across the frequency range $2400 - 2483.5$ MHz, the return loss should be -10 dB or lower. The return loss of an antenna is measured using a Vector Network Analyzer (VNA). This measurement is called an S11-parameter, hence why the system design chapter uses this to refer to return loss. The desired characteristic impedance is 50 Ω. This is chosen since this is a standard within RF design [45]. However, this is primarily a design parameter, therefore no tolerances are applied to this number.

The gain of the antenna is a crucial characteristic in determining the range at which the receiver can reliably detect drones. The gain can be found using Friis transmission law, which calculates the received power through a plethora of variables [46]. These variables are the power $P_{t,r}$, reflection coefficient $\Gamma_{t,r}$, gain $G_{t,r}$ and $\rho_{t,r}$ polarization for both transmitter and receiver. The distance D and wavelength λ also need to be calculated.

$$P_r = (1 - |\Gamma_r|^2)(1 - |\Gamma_t|^2) \cdot G_r \cdot G_t \frac{\lambda^2}{(4\pi D)^2} P_t |\hat{\rho}_r \cdot \hat{\rho}_t|^2 \quad \text{[W]} \quad (3.3)$$

Some assumptions are taken to reduce this equation. The first assumption is identical polarization between transmitter and receivers, which sets $|\hat{\rho}_r \cdot \hat{\rho}_t|^2$ to 1. The second assumption is that the antennas are perfectly matched to 50 Ω at both ends. This reduces $(1 - |\Gamma_r|^2)(1 - |\Gamma_t|^2)$ to 1. The third assumption is that the transmitter is isotropic, which reduces the gain G_t to 1. These three assumptions leave an equation only reliant on the gain of the receiver G_r, the

wavelength of the highest frequency (2483.5 MHz) λ, assuming worst case, the distance between transmitter and receiver D, and the transmitted power P_t. These assumptions reduce the equation as follows.

$$P_r = G_r \frac{\lambda^2}{(4\pi D)^2} P_t \quad [\text{W}] \quad (3.4)$$

In order to calculate the gain, the wavelength, distance and received power need to be calculated. The transmitted power $P_t = 17\,\text{dBm}$ is stated in the specifications of the Matrice 210 RTK [47]. To find the maximum distance, based on Section 3.1, simple geometry is used. Here, the max horizontal distance is 8 km and the max vertical distance is 2.5 km.

$$D = \sqrt{(8 \cdot 10^3\,\text{m})^2 + (2.5 \cdot 10^3\,\text{m})^2} = 8382\,\text{m}$$

This results in a maximum distance of 8382 m between the drone and the receiver. The signal-to-noise ratio (SNR) is now needed, as this determines the lowest amount of power the receiver needs to detect a signal at 8382 m. This is done using the Johnson-Nyquist noise formula (3.5), which calculates the thermal noise at the receiver terminal [40].

$$P_n = S_V \cdot B \cdot \frac{R_L}{(R + R_L)^2} \quad [\text{W}] \quad (3.5)$$

This formula relies on the impedances between the receiver R and the load R_L, the load being the RF front end. These are both assumed to be matched to 50 Ω. The formula also relies on the bandwidth B, which is known to be 83.5 MHz as stated earlier. The last factor is the one-sided power spectral density (PSD) of the voltage noise S_V [48]. PSD describes the average power over a frequency range. The PSD of the voltage noise is defined as such:

$$S_V = 4 \cdot k_B \cdot T \cdot R \quad [\text{V}^2/\text{Hz}] \quad (3.6)$$

This equation relies on the Boltzmann constant $K = 1.38 \cdot 10^{-23}\,\text{J/K}$, T is the temperature in kelvin. The chosen temperature is 309.55 K, which equates to 36.4 °C [49], the highest temperature in Denmark, thus assuming a worst-case scenario [50]. The last variable to calculate the PSD is the resistance of the receiver R, which should be matched to 50 Ω. Inserting (3.6) into Equation (3.5) gives a complete equation to calculate the Nyquist noise.

$$P_n = 4 \cdot k_B \cdot T \cdot R \cdot B \cdot \frac{R_L}{(R + R_L)^2} \quad [\text{W}] \quad (3.7)$$

Inserting values into Equation (3.7) to calculate the Nyquist noise.

$$P_n = 4 \cdot 1.38 \cdot 10^{-23}\,\text{J/K} \cdot 309.55\,\text{K} \cdot 50\,\Omega \cdot 83.5\,\text{MHz} \cdot \frac{50\,\Omega}{(50\,\Omega + 50\,\Omega)^2} = 326.79\,\text{fW}$$

Now to convert the power to dBm to make it easier to calculate the SNR.

$$P_n = 10 \cdot \log10\left(\frac{326.79 \cdot 10^{-15}\,\text{W}}{1 \cdot 10^{-3}\,\text{W}}\right) \approx -94.6\,\text{dBm}$$

Thus resulting in a noisefloor of −94.6 dBm. The received power then has to lie 5 dBm higher to ensure a decent SNR [51].

$$P_r = -94.6 + 5 = -89.6\,\text{dBm}$$

The last unknown variable is the wavelength, which can be calculated using the speed of light in a vacuum $c = 3 \cdot 10^8\,\mathrm{m/s}$ and the highest frequency in the band $f = 2483.5 \cdot 10^6\,\mathrm{Hz}$.

$$\lambda = \frac{3 \cdot 10^8\,\mathrm{m/s}}{2483.5 \cdot 10^6\,\mathrm{Hz}} \approx 120.8\,\mathrm{mm}$$

The receiver gain can then be calculated, since the transmitted power of the drone is about $P_t = 50.12\,\mathrm{mW}$ along with the just calculated variables R, P_r and λ [47].

$$P_r = G_r \frac{\lambda^2}{(4\pi D)^2} P_t \quad [\mathrm{W}] \quad (3.8)$$

Isolating for receiver gain G_r.

$$G_r = \frac{16 \cdot P_r \cdot \pi^2 \cdot D^2}{\lambda^2 \cdot P_t} \quad [-] \quad (3.9)$$

Inserting the values to calculate the needed receiver gain, using $10^{\frac{x\,\mathrm{dBm}-30}{10}}$ to convert from dBm to W [52].

$$G_r = \frac{16 \cdot 10^{\frac{-89.6\,\mathrm{dBm}-30}{10}} \cdot \pi^2 \cdot 8382\,\mathrm{m}^2}{(120.8 \cdot 10^{-3}\,\mathrm{m})^2 \cdot 50.12 \cdot 10^{-3}\,\mathrm{W}} = 16.63$$

Converted to dBi, which is the gain in relation to an isotropic antenna.

$$G_r = 10 \cdot \log10(16.63) = 12.21\,\mathrm{dBi}$$

This means that to ensure a decent SNR at max range, the receiver needs to have a gain of 12.21 dBi [51].

Parameter	Specification
Resonant frequency	2441.75 MHz
Min. frequency range	$2400 - 2483.5\,\mathrm{MHz}$
Bandwidth	$\geq 83.5\,\mathrm{MHz}$
Return loss in band	$\leq -10\,\mathrm{dB}$
Gain	$\geq 12.21\,\mathrm{dBi}$
Characteristic impedance	$50\,\Omega$

Table 3.2: Summary of antenna specifications.

Test of Specifications

Not all specifications set forward in Table 3.2 must be directly tested. This mainly regards the impedance, as it primarily serves as a design guideline to achieve the required return loss within the frequency range. Its impact is however directly measured when measuring return losses. The remaining parameters are validated in the following tests, ensuring that the fabricated antenna performances as required.

Resonant frequency, frequency range, return loss and bandwidth
These four parameters can be tested together. These can be tested by using a network analyzer to make a S-11 measurement. This measurement shows the return loss of the antenna. Now take the readings below $-10\,\mathrm{dB}$. The lowest frequency should be below 2400 MHz, while the

highest should be above 2483.5 MHz. A fallacy to be aware of is to keep the area around the antenna clear, as objects/obstacles here will change the result. The resonant frequency is the minimum of curve and can directly be read.

Gain
The gain should be tested by keeping the antenna still, then transmitting a signal from different angles and measure the received signal strength of the antenna. This can be done with a Multi-Probe Antenna Measurement System, such a has made available to this project by AAU AMPS. This specifically being the SG24 from MVG.

3.3.3 Front End Receiver Specifications

As the prototype must be able to receive and sample signals within the chosen frequency band, a RF front end needed. Through this section, different front end specifications are found, this includes sample rate, bandwidth, attenuation of unwanted signals, impedance and gain. To ensure that the signal from the drone reaches a sufficiently high level to be sampled by the ADC, the system must amplify the incoming signal. The signal strength is found in Section 3.3.2, and has an amplitude is $-89.6\,\text{dBm}$ which can be converted to Watt by doing:

$$10^{\frac{-89.6-30}{10}} \approx 1.1\,\text{pW}$$

With the power received by the RF front end can the the maximum gain of the LNA can be found.

It is found later in the project that the ADC that is being used has an input range of $1.25\,V_{pp}$, see Section 4.2. This is used to calculate the gain needed.
To use this, the power received must be converted to a peak voltage. Here is $50\,\Omega$ being used as this is the impedance of the antenna.

$$\sqrt{1.1 \cdot 10^{-12}\,\text{W} \cdot 50\,\Omega} = 7.4 \cdot 10^{-6}\,V_{\text{RMS}}$$

$$7.4 \cdot 10^{-6} V_{\text{RMS}} \cdot \sqrt{2} = 10.47 \cdot 10^{-6}\,V_{\text{p}}$$

The received power and maximum input of the ADC can then be used to find the gain of the LNA.

$$G_{LNA} = \frac{0.625\,\text{V}}{10.47 \cdot 10^{-6}\,V_{\text{p}}} = 59687.04$$

This means that the gain of the LNA should be 59687.04 $\approx$ 95.51 dB. However, this is slightly lowered to 59000 $\approx$ 95.42 dB, this is done to ensure that clipping of the signal does not occur, and this means that the maximum output signal is expected to be $10.47 \cdot 10^{-6}\,\text{V} \cdot 59000 = 617.8\,\text{mV}$.

The filter must dampen unwanted signals outside the frequency band of the Matrice 210 RTK. How this is done is dependent on the specific architecture that is chosen. The pass band frequency of the signal, the part of the frequency band that carries information, is 2400 MHz to 2483.5 MHz. The attenuation of unwanted signals should be at least 60 dB to limit the interference of noise and unwanted signals [53]. This is deemed as a realistic specification as this can be achieved within one decade, using a third-order filter. It should however be noted that a

passive filter would be preferred, as active filters have limited bandwidth at high frequencies [54].

This also means that a potential ripple in the stopband is not of concern as long as it's peak is 60 dB under the wanted signal. The amount of passband ripple is not known as the sensitivity of the direction of arrival algorithm is not known. This is a realistic specification as it can be achieved within one decade using a third order filter. The filtering part of the front end is often made up of multiple steps, which will be taken into consideration when constructing the filter. The filters can also be used to divide the frequency band into different channels that can be sampled individually from each other [55].

It is known from Nyquist and Shannon's sampling theorem that the ADC must sample at two times the highest frequency that carries information [56]. This means that the sample rate must be $2483.5\,\text{MHz} \cdot 2 = 4967\,\text{MHz}$. If the whole band is sampled at once, this is called direct sampling. There are, however certain techniques that can be used to lower this requirement [57] [55]. This is analyzed further in Section 4.2

As an SNR specification for the system is set forward in Section 3.3.2 this can be used to find the minimum number of bits needed in the ADC to achieve the desired SNR. The SNR of an ADC can be found using equation (3.10), where N is the number of bits [58].

$$SNR = 6.02 \cdot N + 1.76 \quad \text{[dB]} \tag{3.10}$$

As stated in the antenna specifications the SNR must be 5 dB. By inserting the required SNR-level and solving for the least number of bits needed.

$$N = \frac{SNR - 1.76}{6.02}$$

$$N = \frac{5 - 1.76}{6.02} = 0.54\,\text{bits}$$

This means that the ADC must have at least 1 bits. This does however not account for other noise sources such as thermal noise, nonlinearities, jitter ect. Considering this, an assessment was conducted, resulting in raising the requirement to 4 bits [59].

To ensure maximum signal transfer the antennas and the front end must have the same impedance. As the antennas has an output impedance for $50\,\Omega$, the front end must have an input impedance of $50\,\Omega$.

The specifications for the RF front end can be seen in Table 3.3.

Parameter	**Specification**
Amplifier gain	$59000 \approx 95.42\,\text{dB}$
ADC Sample rate[1]	$> 4967\,\text{MHz}$
ADC resolution	$\geq 4\,\text{bit}$
Pass band	$[2400\,;\,2483.5]\,\text{MHz}$
Attenuation outside pass band	$60\,\text{dB}$
Input impedance	$50\,\Omega$

Table 3.3: Summary of RF front end specifications. 1: The exact sample rate will change depending on which architecture is chosen.

Test of specifications

Every specification listed in Table 3.3.3 will be tested. This is the case, even though parameters such as ADC sample rate and resolution is explicitly stated in its datasheet. The test is to confirm correct configuration.

ADC and gain
Sample rate, gain and resolution can all be measured in a single test. To perform this test, sample a signal with a known voltage and frequency. The sample rate can be calculated by counting samples per period. The resolution can be determined by examining the quantization steps. Lastly the gain can be calculated by comparing the measured output amplitude with the known input amplitude.

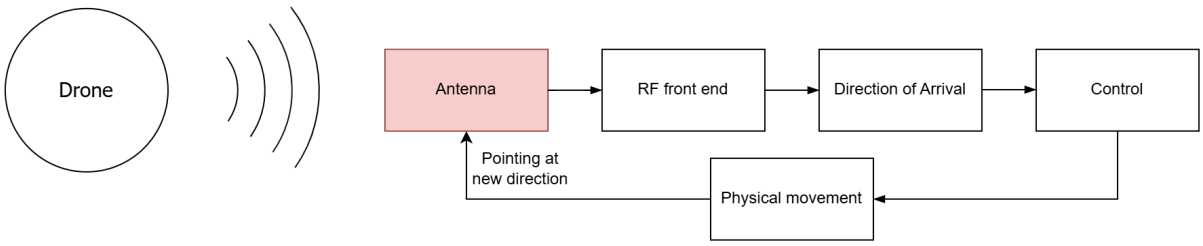
Filter
As stated in this section the filter characteristics is a pass band between 2400 MHz and 2483.5 MHz with an attenuation of 60 dB one decade outside the band. To validate this filter characteristic a frequency sweep from one decade below to one decade above the pass band with a constant voltage should be applied. Now verify that it matches the specified filter.

Impedance
It is expected that a number of active components reside in the RF front end. When measuring the impedance it should therefore be turned on, and configured in accordance to the specifications from Table 3.3. Before connecting it to a network analyzer verify that the output voltage does not exceed the maximum input voltage of the front end. When compatibility is ensured connect the network analyzer and measure the impedance from 2400 MHz to 2483.5 MHz.

Chapter 4

System Design

4.1 Antenna

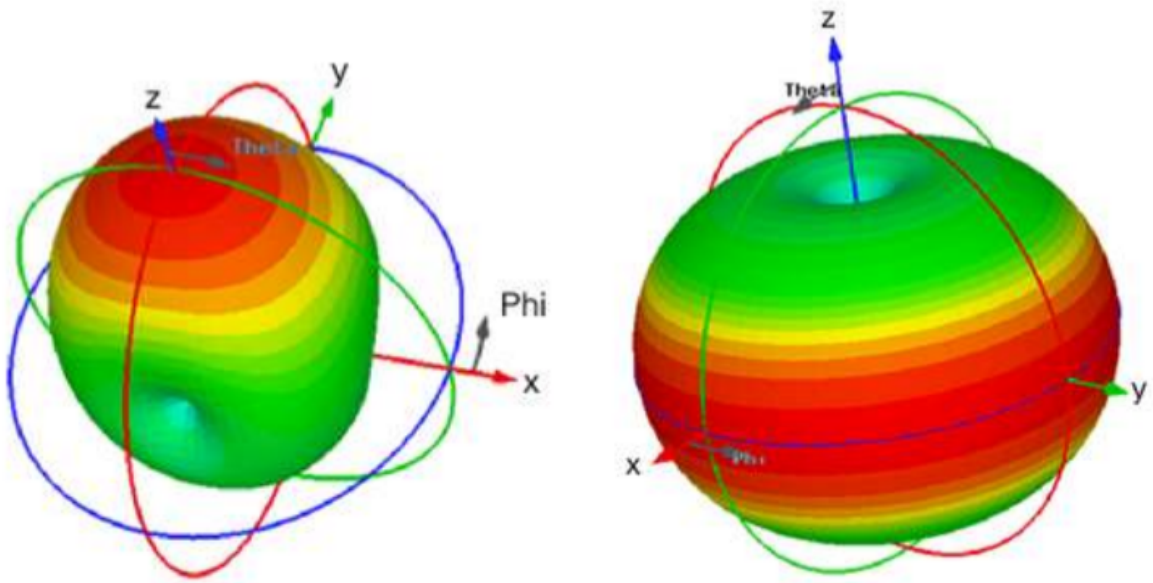


This section explains the development of the antennas along with the calculations, simulations and measurements of the manufactured antennas.

The antenna specifications require a minimum frequency range of $2400 - 2483.5\,\text{MHz}$, where the return loss has to be below $-10\,\text{dB}$. The bandwidth should be at least $83.5\,\text{MHz}$. The S11-parameters of an antenna represent the return loss. The gain should be $12.21\,\text{dBi}$, as detailed in Section 3.3.2. The desired characteristic impedance of the antenna is $50\,\Omega$. To enable beamforming in azimuth and elevation, an antenna array with at least four channels stacked in a 2x2 array is required. The reason for this is explained later in Section 4.3. However, as the RF front end only has two channels (see Section 4.2), a 2x1 will be produced to prove the concept.

4.1.1 Antenna Type

There exists a wide variety of antenna types. In this project, the focus is limited to dipole and patch antennas due to their simplicity, predictable performance, and ease of fabrication [60]. A 3D-view of the radiation pattern of a dipole and a patch antenna is shown in Figure 4.1.



(a) (b)

Figure 4.1: (a) Typical radiation pattern of a single patch antenna. **(b)** Typical radiation pattern of a dipole. The coloring indicates the strength of the radiation, where red is the strongest radiation, and green is the weakest radiation [61].

A patch antenna can determine the angle from which a drone signal arrives, as it has a radiation pattern with a distinct direction of maximum gain, as seen in Figure 4.1 **(a)**. In contrast, a dipole antenna has a circular radiation pattern with uniform gain in the horizontal plane, as seen in Figure 4.1 **(b)**. This prevents the ability to estimate the angle of arrival of the signal. Additionally, patch antennas have inherently more gain than dipoles [62]. The gain of an antenna refers to how well the antenna can direct power in a specific direction [60]. For these reasons, a patch antenna is selected over a dipole antenna.

4.1.2 Overview of Antenna Section

Firstly, a single patch antenna, Antenna 0, is realized to make sure that measurements match the calculations and simulations in Computer Simulation Technology (CST) Studio Suite. Antenna 1 is an iteration of Antenna 0 to adjust the resonant frequency. Antenna 2 is a subarray to increase the gain of the antenna. Antenna 3 is an iteration of Antenna 2, to further adjust the resonant frequency. Then, two identical subarrays are constructed, Antenna 3.1 and 3.2. Antenna 4 is then the two subarray antennas 3.1 and 3.2 aligned as the two-element array. A summary of the manufactured antennas is seen in Table 4.1

Name of the Antenna	Antenna configuration
Antenna 0	Single patch antenna
Antenna 1	Single patch antenna
Antenna 2	Antenna array (subarray)
Antenna 3	Antenna array (subarray)
Antenna 3.1 and 3.2	Two identical subarrays
Antenna 4	Two-element array consisting of the two subarrays Antenna 3.1 and 3.2

Table 4.1: Summary of the fabricated antennas, detailing the configuration of each antenna.

4.1.3 Antenna 0

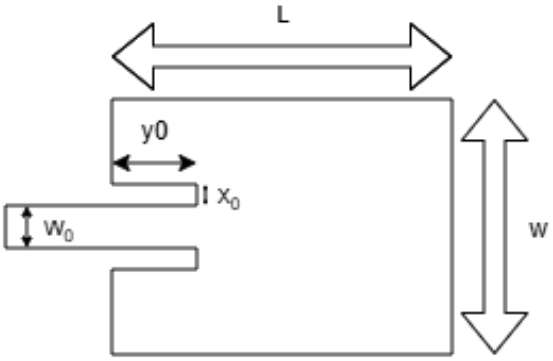


Figure 4.2: Antenna parameters for a single patch antenna.

The dimensions of a single patch antenna have to be calculated for Antenna 0. The parameters to be determined for the antenna are shown in Figure 4.2. This is done with the use of a design procedure where the resonant frequency, dielectric constant and the height of the substrate are known. The rest of the parameters for the patch antenna can then be calculated [40]. The width (W) of the patch antenna is calculated by the formula (4.1) [63].

$$W = \frac{c}{2 \cdot f_r} \cdot \sqrt{\frac{2}{\varepsilon_r + 1}} \quad \text{[m]} \quad (4.1)$$

Where c is the speed of light in vacuum, which is $3 \cdot 10^8\,\text{m/s}$, and the resonant frequency f_r is the center frequency of the band $f_r = \frac{2400+2483.5\,\text{MHz}}{2} = 2441.75\,\text{MHz}$ [64]. The dielectric constant ε_r of the substrate, which is FR4, ranges between 3.8 and 4.8 [65], and is chosen to be 4.3, as this is the median value. The width of the patch antenna is then found.

$$W = \frac{3 \cdot 10^8\,\text{m/s}}{2 \cdot 2441.75 \cdot 10^6\,\text{Hz}} \cdot \sqrt{\frac{2}{4.3 + 1}} = 37.74\,\text{mm}$$

The length (L) can be found by the formula (4.2) [63].

$$L = \frac{c}{2 \cdot f_r \cdot \sqrt{\varepsilon_{reff}}} - 2 \cdot \Delta L \quad \text{[m]} \quad (4.2)$$

To do this, the effective dielectric constant (ε_{reff}), and the change in length ΔL have to be found first by formula (4.3) and (4.4) accordingly [63].

$$\varepsilon_{reff} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2} \cdot (1 + 12 \cdot \frac{h}{W})^{-\frac{1}{2}} \quad [-] \quad (4.3)$$

$$\Delta L = 0.412 \cdot \frac{(\varepsilon_{reff} + 0.3) \cdot (\frac{W}{h} + 0.264)}{(\varepsilon_{reff} - 0.258) \cdot (\frac{W}{h} + 0.8)} \cdot h \quad [\text{m}] \quad (4.4)$$

The effective dielectric constant is based on the dielectric constant and the antenna geometry, which is dependent on the resonant frequency of the antenna [40]. The length of the antenna is found by subtracting 2 times the change in length, as seen in Equation (4.2), because of the fringing effects. Fringing occurs because the E-field at the patch edges extends beyond the length of the antenna, effectively increasing the electrical length of the antenna [40]. This is seen in Figure 4.3.

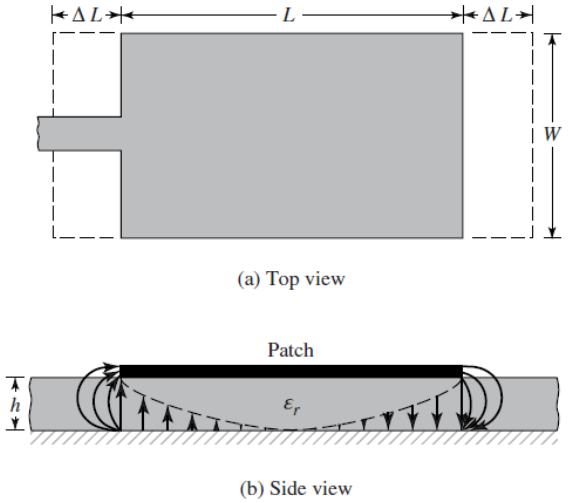


Figure 4.3: (a) Top view of the geometry of the change in length. **(b)** Side view of the fringing effects, where the E-field increases the electrical length of the antenna [40].

Then ε_{reff} and ΔL are calculated with the known values for width (W), ε_r, and the height (h) of the substrate, which is 1.61 mm [66].

$$\varepsilon_{reff} = \frac{4.3 + 1}{2} + \frac{4.3 - 1}{2} \cdot (1 + 12 \cdot \frac{1.61 \cdot 10^{-3}\,\text{m}}{37.74 \cdot 10^{-3}\,\text{m}})^{-\frac{1}{2}} = 3.99$$

$$\Delta L = 0.412 \cdot \frac{(3.99 + 0.3) \cdot (\frac{37.74 \cdot 10^{-3}\,\text{m}}{1.61 \cdot 10^{-3}\,\text{m}} + 0.264)}{(3.99 - 0.258) \cdot (\frac{37.74 \cdot 10^{-3}\,\text{m}}{1.61 \cdot 10^{-3}\,\text{m}} + 0.8)} \cdot 1.61 \cdot 10^{-3}\,\text{m} = 745.59\,\mu\text{m}$$

The length of the patch antenna is then determined.

$$L = \frac{3 \cdot 10^8\,\text{m}}{2 \cdot 2441.75 \cdot 10^6\,\text{Hz} \cdot \sqrt{3.99}} - 2 \cdot 745.59 \cdot 10^{-6}\,\text{m} = 29.26\,\text{mm}$$

A feed is the coupling between the antenna and the RF front end that ensures proper impedance matching, allowing the antenna to radiate efficiently. There are many ways to construct a feed. For this project, a microstrip feed line is selected as the feed, which is shown in Figure 4.2.

This feed line is chosen due to its simple design and ease of calculation. The feed line width (W_0) is calculated using Equation (4.5) [63]. In combination with choosing a feed line width to match the impedance, two insertions are made on either side of the feed line, which is seen in Figure 4.2, where y_0 is the length of these. The length of these insertions can be adjusted to match the impedance of the antenna to 50 Ω [63]. The insertion length y_0 is calculated using the Formula (4.6) [63]. An increase in the width of the feed line will result in a decrease in impedance, and an increase in the insertion length will result in a decrease in impedance, until the impedance is 0 Ω. A further increase of the insertion length would increase the impedance until the impedance is 240 Ω. The 240 Ω in the Equation (4.6) is chosen as it is a typical value for the edge impedance of a patch antenna [63]. For the width of these insertions, x_0 in the Figure 4.2, 1 mm is used as a rule of thumb [63].

$$Z_c = \frac{\frac{\eta_0}{\sqrt{\varepsilon_{reff}}}}{\frac{W_0}{h} + 1.393 + \frac{2}{3} \cdot \ln(\frac{W_0}{h} + 1.444)} \quad [\text{m}] \quad (4.5)$$

$$Z_c = 240\,\Omega \cdot cos(\frac{\pi}{L} \cdot y_0)^2 \quad [\text{m}] \quad (4.6)$$

Where the intrinsic impedance of free space η_0 is $120\pi\,\Omega$ and the desired characteristic impedance Z_c is 50 Ω. The feedline width W_0 and the insertion length y_0 are thereby found.

$$50\,\Omega = \frac{\frac{120\pi\,\Omega}{\sqrt{3.99}}}{\frac{W_0}{1.61 \cdot 10^{-3}\,\text{m}} + 1.393 + \frac{2}{3} \cdot \ln(\frac{W_0}{1.61 \cdot 10^{-3}\,\text{m}} + 1.444)} \text{ isolate for } W_0 \quad [W_0 = 2.63\,\text{mm}]$$

$$50\,\Omega = 240\,\Omega \cdot cos(\frac{\pi}{29.26 \cdot 10^{-3}\,\text{m}} \cdot y_0)^2 \text{ isolate for } y_0 \quad [y_0 = 10.21\,\text{mm and } 19.04\,\text{mm}]$$

The Equation (4.6) results in two lengths for y_0, which is 10.21 mm and 19.04 mm. This is because two angles can have the same cosine value. The chosen value for the insertion length is $y_0 = 10.21$ mm. The calculated dimensions are used in CST to simulate the antenna. As the equations to determine the size of the substrate are known, some initial values are chosen, the substrate width W_s is 56.5 mm and the length L_s is 52.3 mm. The dimensions of the calculated antenna are seen in Figure 4.4.

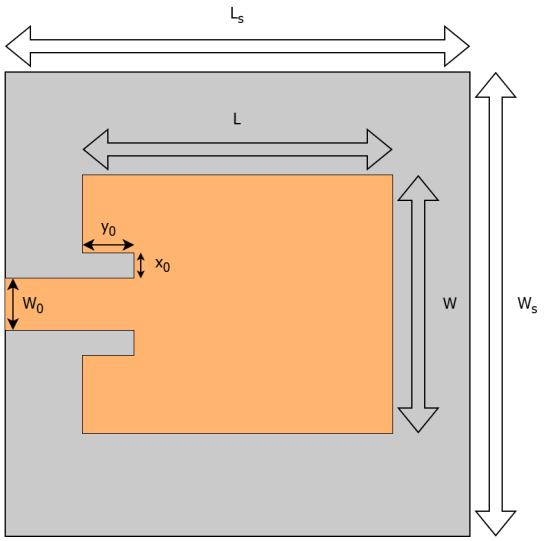


Figure 4.4: Diagram of calculated antenna, the dimensions are as follows $W_0 = 2.63$ mm, $y_0 = 10.21$ mm, $x_0 = 1$ mm, L = 29.26 mm, W = 37.74 mm, $L_s = 52.3$ mm and $W_s = 56.5$ mm.

After determining the patch antenna dimensions, the calculated antenna is simulated in CST to obtain a more accurate prediction of its performance, as shown in Figure 4.5. A simulation of S11-parameters is made to ensure that the antenna meets the specifications, which can be seen in Figure 4.5. The S11-parameters represent the return loss, where $-10\,\text{dB}$ return loss is used to define the frequency range and the bandwidth of the antenna.

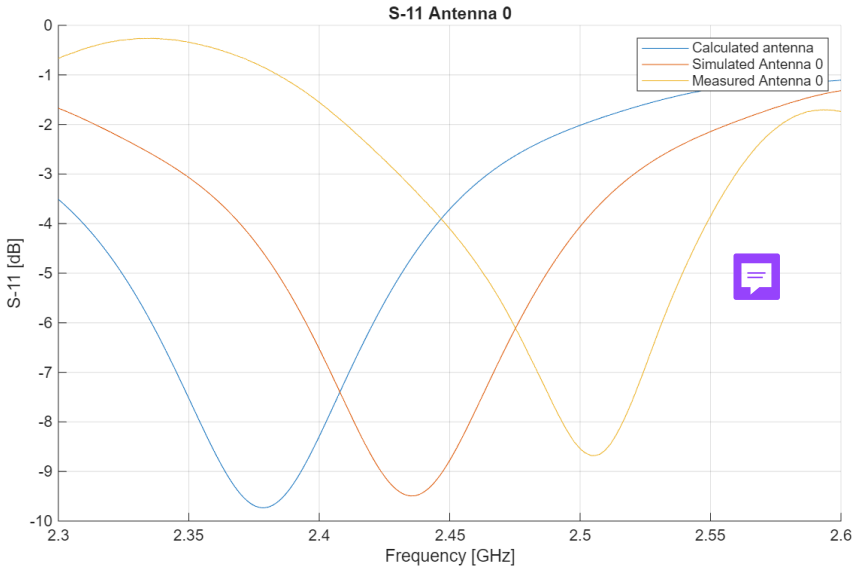


Figure 4.5: Comparison in the S11-parameters between the calculated antenna and simulation and measurement of Antenna 0.

For the calculated antenna, the resonant frequency is 2378.4 MHz, as seen in Figure 4.5. The dimensions are then adjusted in CST to match the required resonant frequency of 2441.75 MHz. This antenna will henceforth be denoted as Antenna 0, the dimensions of Antenna 0 are seen in Figure 4.6.

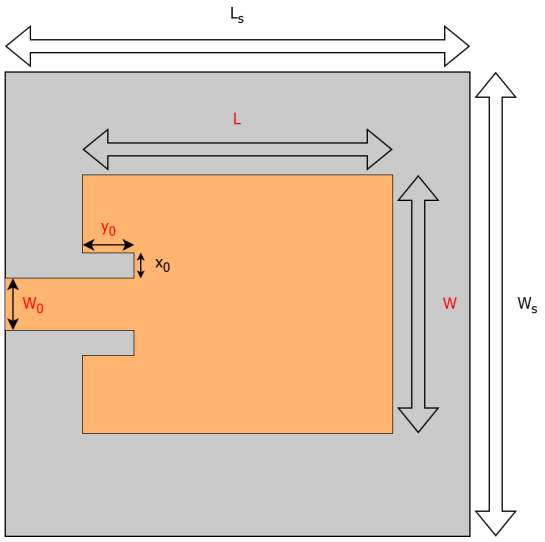


Figure 4.6: Diagram of Antenna 0, where the dimensions marked by red are changed to $W_0 = 2\,\text{mm}$, $y_0 = 9.8\,\text{mm}$, $L = 28.6\,\text{mm}$ and $W = 34.8\,\text{mm}$. The following dimensions are still $x_0 = 1\,\text{mm}$, $L_s = 52.3\,\text{mm}$ and $W_s = 56.5\,\text{mm}$.

Antenna 0 is manufactured, and the S11-parameters are measured. The test journal is seen in Appendix 3. In Figure 4.5, the measured resonant frequency of Antenna 0 is 2504.3 MHz, compared to the desired resonant frequency, which is 2441.75 MHz. Meaning there is an offset in frequency of $2504.3 - 2441.75 = 62.55$ MHz. Therefore, the resonant frequency needs to be adjusted adequately in the next antenna. This offset is due to a variation in the dielectric constant, which is investigated further in Section 4.1.9. In short, the dielectric constant of the used FR4-PCB is lower than expected. Adjusting the dielectric constant in CST would have resulted in more accurate simulations and should ideally have been done. However, during the antenna development process, the resonant frequency was lowered in CST instead.

4.1.4 Antenna 1

Since there is an offset in the resonant frequency in Antenna 0, seen in Figure 4.5. The resonant frequency is set approximately 62.55 MHz lower in CST, by increasing the width and length of the Antenna until the desired resonant frequency is reached. This results in new dimensions for Antenna 1, which are shown in Figure 4.7.

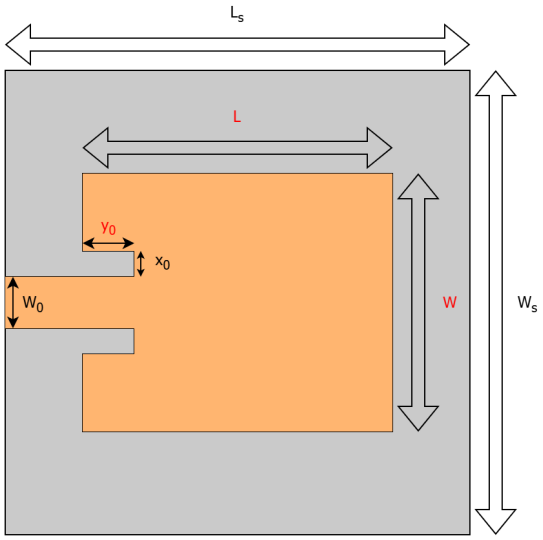


Figure 4.7: Diagram of Antenna 0, where the following dimensions changed to $y_0 = 10$ mm, $L = 29.4$ mm and $W = 35$ mm. The following dimensions are kept the same $W_0 = 2$ mm, $x_0 = 1$ mm, $L_s = 52.3$ mm and $W_s = 56.5$ mm.

S11-parameters

Figure 4.8 shows the measured S11-parameters of Antenna 1, compared to the simulation in CST. This test uses the same procedure as the test journal in Appendix 3.

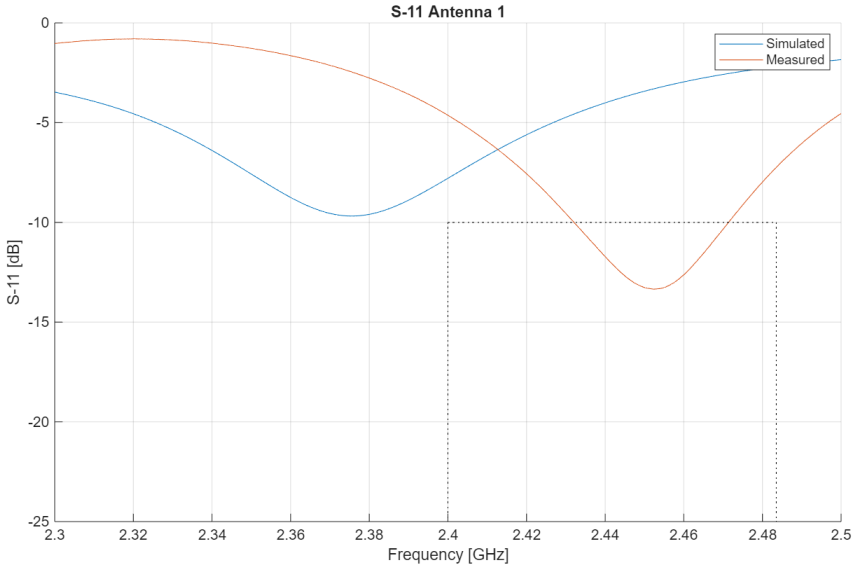


Figure 4.8: Comparison between simulation and measured S11-parameters of Antenna 1. The black dotted lines indicate the desired minimum frequency range, where the return loss is below -10 dB.

The offset in frequency in Figure 4.8 is due to a lower dielectric constant than expected, as mentioned in Section 4.1.3. It is seen in Table 4.2 that the measured resonant frequency of Antenna 1 is offset by about 10 MHz compared to the desired resonant frequency. The frequency range, where the return loss is below -10 dB, lies in the desired frequency range. However, the bandwidth is only 39 MHz, which is significantly lower than the desired bandwidth of 83.5 MHz. Therefore, this must be improved.

Parameter	Specifications	Antenna 1
Min. frequency range	2400-2483.5 MHz	2432.2-2471.2 MHz
Bandwidth	$\geq$ 83.5 MHz	39 MHz
Resonant frequency	2441.75 MHz	2452.4 MHz

Table 4.2: Comparison between Antenna 1 and specifications for bandwidth, frequency range and resonant frequency.

Antenna impedance analysis

Since the S11-parameters of the simulated model of Antenna 1 do not reach the requirement in the specifications 4.2, an analysis of the relation between impedance and S11-parameters is performed.

Three different frequencies, 2300.3, 2377.1 and 2450 MHz, including the resonant frequency, are selected from the S11-parameters of Antenna 1. Note, the frequencies are offset compared to the specifications, since they are based on the simulation, which is offset compared to the physical antenna. The chosen frequencies demonstrate how impedance correlates to return

loss. The three frequencies are seen in Figure 4.9, marked as the curve markers 1, 2 and 3.

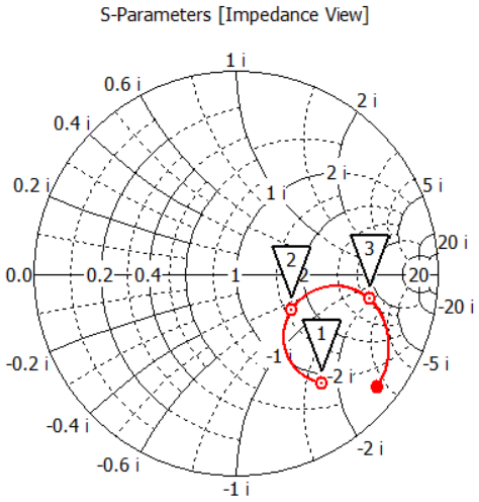


Figure 4.9: Smith Chart for Antenna 1, from 2.3 GHz to 2.6 GHz, where the curve marker 1 is at 2300.3 MHz, curve marker 2 is at 2377.1 MHz, which is the resonant frequency in the simulation. Curve marker 3 is at 2450 MHz.

The three selected frequencies, and their corresponding impedance, are shown in Table 4.3.

Curve Marker	Frequency	Impedance
1	2300.3 MHz	$42.87\,\Omega - j86.08\,\Omega$
2	2377.1 MHz	$80.40\,\Omega - j31.66\,\Omega$
3	2450 MHz	$209.41\,\Omega - j94.82\,\Omega$

Table 4.3: The curve markers and their corresponding frequency and impedance.

The impedance of the antenna in the three frequencies clearly does not match the $50\,\Omega$ impedance set in the specifications 4.2. This means that a part of the signal will be reflected when connecting the antenna and the RF front end. The corresponding magnitude of the reflection $|\Gamma|$ coefficient can be calculated with Equation (4.7) [40].

$$|\Gamma| = |\frac{Z_1 - Z_2}{Z_1 + Z_2}| \qquad [-] \quad (4.7)$$

Where Z_2 is $50\,\Omega$, since it is the input impedance of the RF front end. Z_1 is the impedance at the different frequencies shown in Table 4.3. The magnitude of the reflection coefficient is then calculated for the three different frequencies.

At 2300.3 MHz

$$|\Gamma_1| = |\frac{(42.87\,\Omega - j86.08\,\Omega) - 50\,\Omega}{(42.87\,\Omega - j86.08\,\Omega) + 50\,\Omega}| = 0.68210$$

At 2377.1 MHz

$$|\Gamma_2| = |\frac{(80.40\,\Omega - j31.66\,\Omega) - 50\,\Omega}{(80.40\,\Omega - j31.66\,\Omega) + 50\,\Omega}| = 0.32707$$

At 2450 MHz

$$|\Gamma_3| = |\frac{(209.41\,\Omega - j94.82\,\Omega) - 50\,\Omega}{(209.41\,\Omega - j94.82\,\Omega) + 50\,\Omega}| = 0.67154$$

Then $|\Gamma|$ is used in Equation (4.8) to find the return loss (RL) [40].

$$RL = 20 \cdot \log_{10} |\Gamma| \quad \text{[dB]} \quad (4.8)$$

At 2300.3 MHz

$$RL_1 = 20 \cdot \log_{10}(0.68210) = -3.32297\,\text{dB}$$

At 2377.1 MHz

$$RL_2 = 20 \cdot \log_{10}(0.32707) = -9.70719\,\text{dB}$$

At 2450 MHz

$$RL_3 = 20 \cdot \log_{10}(0.67154) = -3.45855\,\text{dB}$$

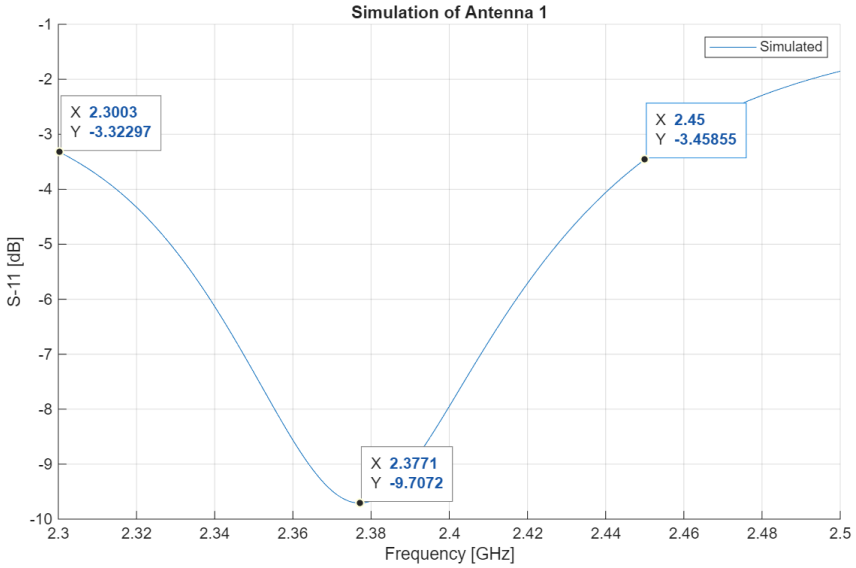


Figure 4.10: Simulated S11-parameters for Antenna 1, with the three frequencies, corresponding to the curve markers 1,2 and 3 in the Smith chart shown in Figure 4.9.

It is seen in Figure 4.10 that the impedance of these three frequencies corresponds to the S11-parameters, since the calculated return loss corresponds to the exact simulated S11-parameters. It can thereby be concluded that to have a better return loss, meaning a lower number in dB, the reflection coefficient should be as low as possible, which it is when $Z_1 = Z_2$. To facilitate the development of new antennas, the Smith chart impedance tool in CST is used to optimize the length of y_0. This ensures the input impedance is matched as closely as possible to 50 Ω at the target resonant frequency.

Gain

The gain of Antenna 1 is simulated in CST and measured to ensure that it meets the specifications.

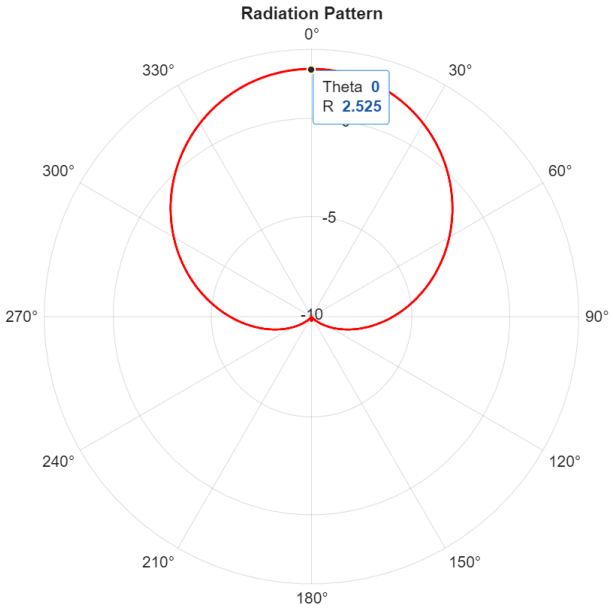


Figure 4.11: Simulation of the gain of Antenna 1 at the resonant frequency.

It is seen in Figure 4.11 that the simulated gain is 2.53 dBi. To ensure the simulation is accurate, the gain is also measured. The test journal is seen in Appendix 4.

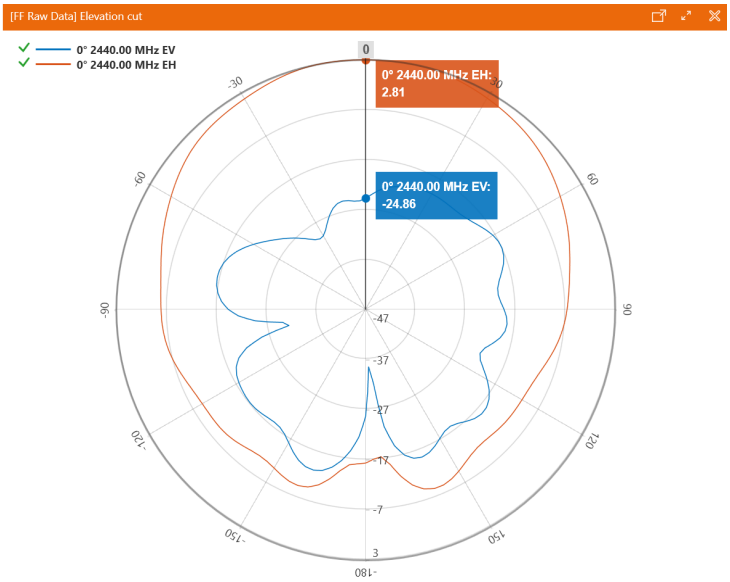


Figure 4.12: Test of the gain for Antenna 1 at 2440 MHz. EH is the strength of the electric field in the horizontal plane, and EV is the strength in the vertical plane.

The measured gain of Antenna 1 is seen in Figure 4.12, the gain is 2.81 dBi, which is close to the simulated gain of 2.53 dBi. The measured gain is higher than the simulated gain, the cause of this discrepancy is unknown. This is still much lower than the 12.21 dBi, which is the specification for the antenna, shown in Section 3.3.2. Therefore, the implementation of an antenna array is explored, since it is known to enhance gain [67].

4.1.5 Antenna 2

Assuming that designing the antenna with an improved impedance match should increase the bandwidth, Antenna 2 will focus on improving the gain. There are a few ways to increase the gain of the antenna [68]:

- Increase the size
- Redirect waves towards the antenna using a parabolic dish antenna
- Antenna array

Since an increase in the size of the patch antenna will result in a lower resonant frequency, this option is avoided. Redirecting the electromagnetic waves towards the antenna is an option, but it would require a dish to be built surrounding the antenna. Building an antenna array is a simple solution that has been chosen to move forward with.

Before designing an antenna array, it is possible to calculate the number of antennas needed to actually obtain the correct amount of gain with the following equation(4.9)[40]:

$$\vec{E}_{total} = \vec{E}_n \cdot AF \quad [\mathrm{V/m}] \quad (4.9)$$

$\vec{E}_{total}$ is the electromagnetic field of the entire linear uniform antenna array. $\vec{E}_n$ is the electromagnetic field of a single antenna in the array. AF is the array factor which describes the radiation pattern. For Antenna 1 the gain is found in Figure 4.12 at 2.81 dBi, this means that Equation (4.9) must be related to gain instead of the electromagnetic field. Thus, a proof in Appendix 2 shows how Equation (4.9) is converted to Equation (4.10):

$$G_{total} = G_n \cdot N \quad [-] \quad (4.10)$$

G_{total} is the total linear gain produced. G_n is the linear gain of a single antenna. N is the number of antennas/elements in the array. In the previous Section 4.1.4, the gain of Antenna 1 is found to be 2.81 dBi. Note, since the impedance of Antenna 1 is not matched to 50 Ω, the gain can be increased further. Since Equation (4.10) uses linear gain, the gain of Antenna 1 is converted to linear gain $10^{\frac{2.81\,\mathrm{dBi}}{10}} = 1.91$. The linear gain required for the prototype is 16, see Section 3.3.2. This is inserted into Equation 4.10 to determine the number of elements to reach the required linear gain.

$$16 = 1.91 \cdot N \Rightarrow N = 8.38$$

Since it is not possible to have fraction of an antenna, N is rounded up to the next whole number, meaning $N = 9$. To acquire the required gain, the prototype needs at least 9 antennas in an array. An ideal antenna array would be 3**x**3 to create symmetric pattern radiation. Since the proof of concept only focuses on 2**x**1 array, this will not be focused on further.

Antenna Array Theory

The following theory is based on the book Antenna Theory by Constantine A. Balanis [40]. Placing antennas in an array essentially changes the radiation pattern and increases the gain. The distance and the phase shift between the antennas have a large influence on the radiation pattern. Once placed in an array, the antennas are referred to as elements. Different types

of arrays exist, though the simple form is the uniform linear array (ULA), which means the gain and the dimensions are identical, as well as the spacing between each element. This is chosen as an initial radiation pattern, where the individual elements and spacings can be tweaked if necessary. Within ULAs, four common types of arrays exist: broadside, Ordinary end-fire, Hansen-Woodyard end-fire and scanning. In a broadside array the maximum gain is perpendicular (90°) to the antenna array. For both the Hansen-Woodyard and ordinary end-fire, the maximum gain is at 0° or 180°. Since the patch antenna's radiation is perpendicular to the surface, creating an end-fire array alters the radiation pattern of a patch antenna from being perpendicular to the array, and for that reason it is not looked further into. Scanning array is able to scan from 0° to 180° in the horizontal direction and beam steering in the vertical direction allows for it to scan the upper side of the airspace, see Figure 4.13.

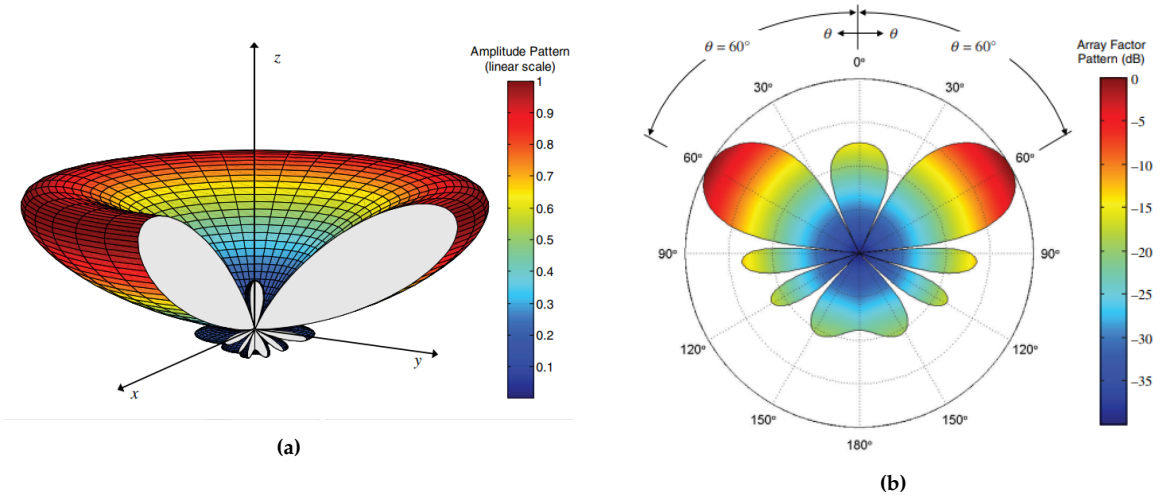


Figure 4.13: (a) 3D radiation pattern of a scanning array [40]. **(b)** Horizontal radiation pattern of a scanning array [40].

The scanning array is able to receive signals and scan the airspace quickly, but it is not able pinpoint the exact horizontal direction of the signal. Figure 4.13 shows the radiation pattern of a scanning array. Through phase shifting, the beam is steered in the elevation angle (θ). This demonstrates that while a signal can be detected at a specific vertical angle, the precise horizontal angle cannot be determined. Based on this, the broadside array is identified as the most suitable type of uniform linear array to move forward with.

When designing a ULA, a key part is the distance between the elements. Changing the spacing between the elements alters the type of radiation pattern significantly, see Figure 4.14.

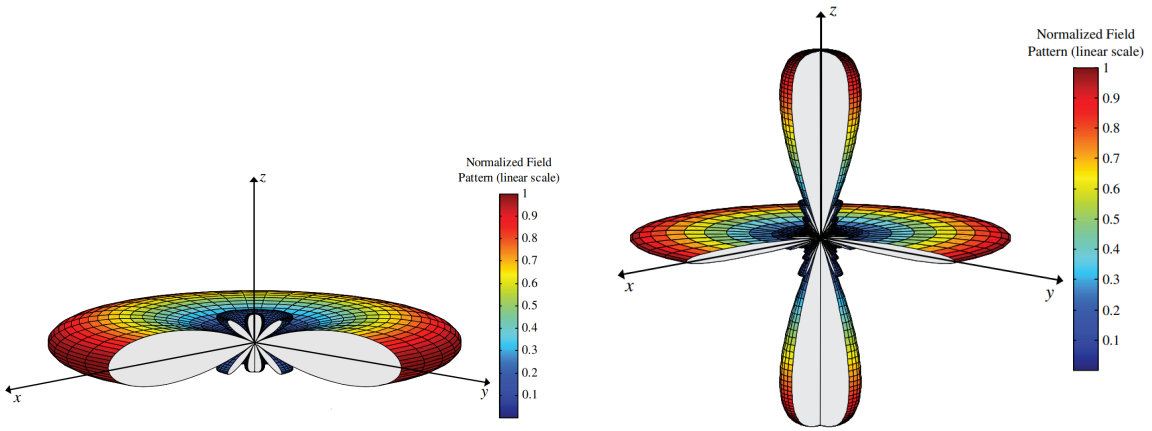


Figure 4.14: (a) Radiation pattern broadside array ($\beta = 0$, $d = \frac{\lambda}{4}$) **(b)** Radiation pattern broadside array ($\beta = 0$, $d = \lambda$). The antenna array lies along the Z-axis [40].

Figure 4.14 shows that when the element spacing is increased from $\lambda/4$ to λ, the array develops grating lobes. Grating lobes are strong, unintended beams in an antenna or antenna array. Consequently, the direction of maximum gain shifts from a broadside (90°, 270°) to include end-fire directions as well (0°, 90°, 180°, 270°). For the broadside array, the distance between the elements must be lower than λ, or the maximum gain can be received in four angles. The phase (β) between the elements alters the direction of the beam. In a broadside array, the phase needs to be 0, keeping the beam perpendicular to the array.

Adding more elements to the array creates a more narrow beam, as seen in Figure 4.15. The more elements in the array, the narrower the main lobe and the more side lobes appear.

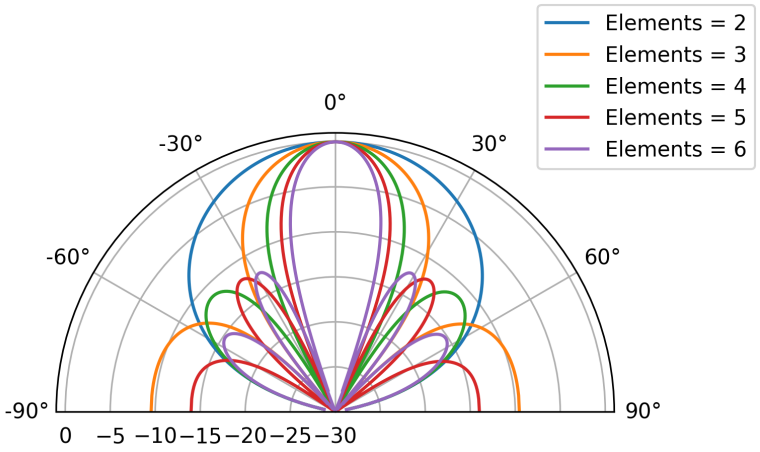


Figure 4.15: A 1D isotropic ULA showing the effect of multiple elements on the radiation pattern.

Designing the antenna

To increase the gain of the antenna, it has been chosen to duplicate Antenna 1 and place them in a subarray, see Figure 4.16.

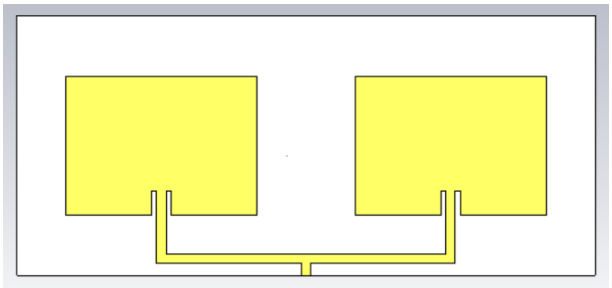


Figure 4.16: The Feed network of Antenna 2, which is a subarray.

According to Equation (4.10), this subarray should double the linear gain to 3.82, which is 5.82 dBi. Since it is chosen to design a broadside array, the phase between the elements needs to be 0. This means the distance from the connector to each of the elements needs to be the same, which is seen in Figure 4.16.

For a broadside array, the distance between the elements needs to be less than the wavelength to ensure the maximum gain is not at multiple directions. Through some simulations of different distances between the elements, $\frac{\lambda}{2}$ is chosen, based on it not having any side lobes and being close to the expected gain of 5.82 dBi with a gain of 5.72 dBi, see Figure 4.17.

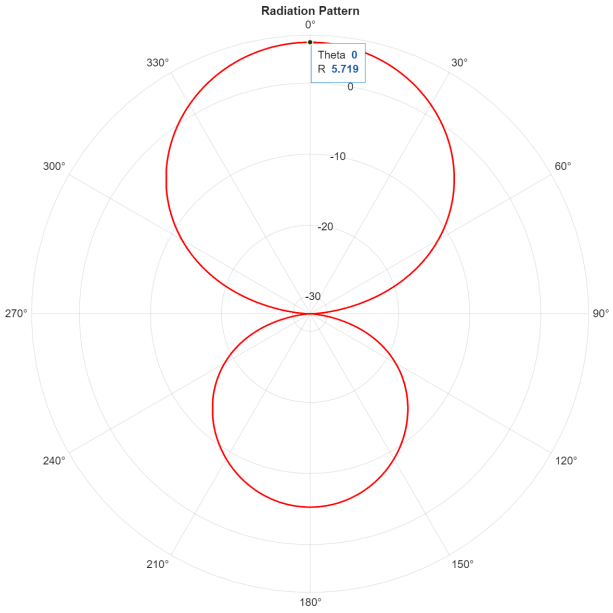


Figure 4.17: Simulated radiation pattern broadside/endfire array ($\beta = 0$, $d = \frac{\lambda}{2}$).

When arranging the two elements in an array, it halves the impedance, since they are in parallel. This means that the subarray will have an impedance of 25 Ω, if Antenna 1 has an input impedance of 50 Ω. The analysis in Section 4.1.4 also indicates the importance of the impedance in order to obtain better results in the S11-parameters. Therefore, the insertion

length (y_0) needs to be adjusted through simulation to obtain a collected impedance of 50 Ω of the subarray. All these considerations result in the following dimensions for Antenna 2, see Figure 4.18.

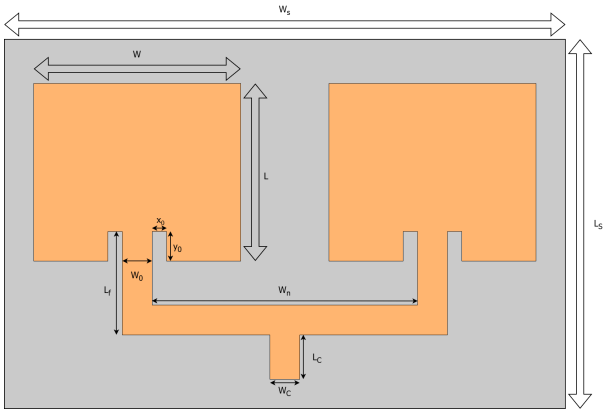


Figure 4.18: Diagram of Antenna 2, where the dimensions are the following $W_s = 120\,mm$, $L_s = 60.9\,mm$,$W = 39.6\,mm$, $L = 28.7\,mm$, $x_0 = 1\,mm$, $y_0 = 5\,mm$, $L_f = 15\,mm$, $W_0 = 2\,mm$, $W_n = 58\,mm$, $L_C = 2.6\,mm$ and $W_C = 2\,mm$.

Figure 4.19 shows the Smith chart of the simulated model.

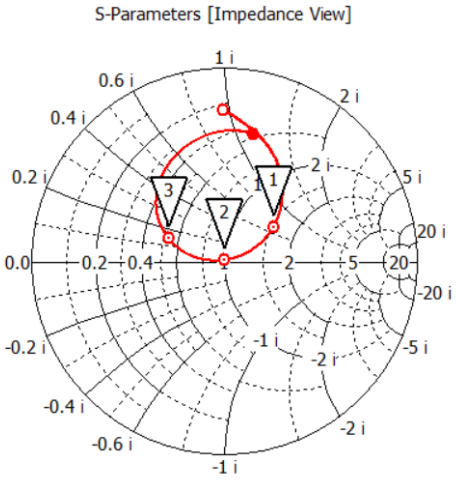


Figure 4.19: Smith Chart Impedance View of Antenna 2. Curve markers 1 and 3 correspond to the frequencies that define the frequency range of the antenna, and curve marker 2 shows the resonant frequency of the antenna. The red curve represents the simulated impedance response over the frequency sweep, which is from 2.2 to 2.6 GHz.

In Figure 4.19, curve marker 2 shows the resonant frequency, which is in the middle of the Smith chart, and therefore closely matches 50 Ω. Curve marker 1 is at 2327 MHz and curve marker 3 is at 2418 MHz, indicating the edges of the frequency range in simulation, when taking into account the offset seen in Figure 4.5. But they do not match 50 Ω. This aligns with the curve in the S11-parameters, where the resonant frequency dip is much deeper than at the edges of the frequency range, as the edges only need to be below $-10\,dB$. The impedance at the edges of the frequency range and the resonant frequency, and their corresponding return loss is shown in Table 4.4, where the edges of the frequency range are below $-10\,dB$ in the S11-parameters.

Curve Marker	Frequency	Impedance	Return Loss
1	2327 MHz	$76.48\,\Omega + j30.49\,\Omega$	-10.16 dB
2	2370.4 MHz	$49.94\,\Omega + j0.79\,\Omega$	-41.97 dB
3	2418 MHz	$26.89\,\Omega + j7.34\,\Omega$	-10.06 dB

Table 4.4: Overview of how closely the frequency range edges and the resonant frequency match $50\,\Omega$, and their corresponding return loss.

S11-parameters

Figure 4.20 shows the S11-paramters of the simulated and measured model. This test uses the same procedure as the test journal in Appendix 3.

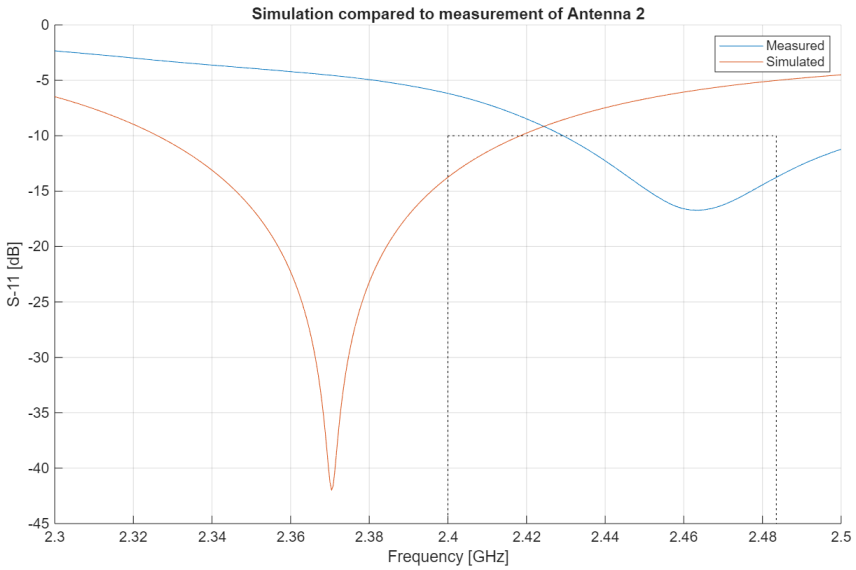


Figure 4.20: Comparison between simulation and measured S11-parameters of Antenna 2. The black dotted lines indicate the desired minimum frequency range, where the return loss is below -10 dB.

In Figure 4.20 the measured resonant frequency is at 2463.2 MHz. Meaning there is an offset, for the desired resonant frequency of $2463.2 - 2441.75 = 21.45$ MHz. The placement of the resonant frequency should be improved by lowering the resonant frequency in CST according to the offset. Another crucial note is that the measured S11-parameters show that the impedance is not matched to $50\,\Omega$ as expected, since this would have resulted in a lower dB-value in the S11-parameters, especially at the resonant frequency, which is mentioned in Section 4.1.4. This difference might be due to tolerances in the cuts when milling the PCBs for the antennas.

Effects of fabrication tolerances on S11-parameters

There has been observed quite a large difference between the simulated Antenna 2 and the fabricated equivalent. This difference includes a large variation in both frequency and the depth of the S11-parameters. The frequency offset is intentional, as mentioned earlier, since it was decided to merely offset the frequency in simulation due to the offset seen in Figure 4.5. The offset is related to a lower dielectric constant than expected. An ideal way to do this would have been to find a dielectric constant that was closer to reality instead of offsetting the simulation, as mentioned in Section 4.1.4. The hypothesis for why reality deviates so much from the simulation is that the PCB-router has some manufacturing tolerances that can affect the matching of the antenna. The actual manufactured Antenna 2 has thus been modeled in

CST with the measurements of the antenna as measured by a caliper. This allows confirmation of whether or not the tolerances can result in the $\sim 25\,\text{dB}$ difference. The measured dimensions can be seen in Figure 4.21 with the differences from the simulated dimensions and the measured seen in Table 4.5.

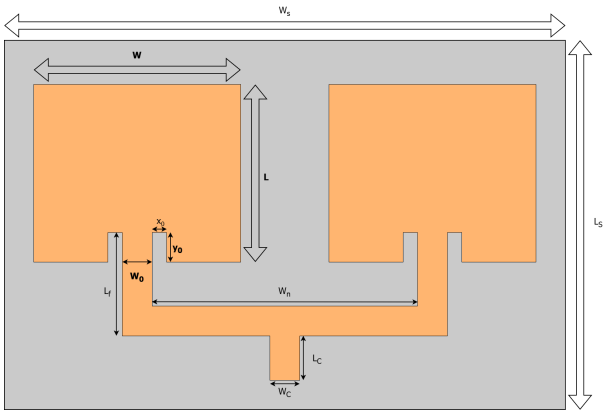


Figure 4.21: Measured values of the fabricated Antenna 2.

Dimension	Expected	Measured	Difference
W	39.6 mm	39.62 mm	0.02 mm
L	28.7 mm	28.75 mm	0.05 mm
y_0	5 mm	4.7 mm	0.3 mm
W_0	2 mm	2.07 mm	0.07 mm
h	1.61 mm	1.59 mm	0.02 mm
ε_r	4.3	3.985	0.315

Table 4.5: Measured differences between simulated and fabricated Antenna 2, with a dielectric constant found by stepping it in CST until it fit the fabricated antenna.

It is seen that the differences are minor, the resulting S11-parameters have been simulated in CST, stepping the dielectric constant to get close to what is seen in the testing of Antenna 2, that being 3.985. It is seen through simulations in Section 4.1.9, that the dielectric constant primarily affects the resonant frequency with minor changes in the return loss, the difference between minimum and maximum dielectric constant is 3 dB. The resulting simulation can be seen in Figure 4.22.

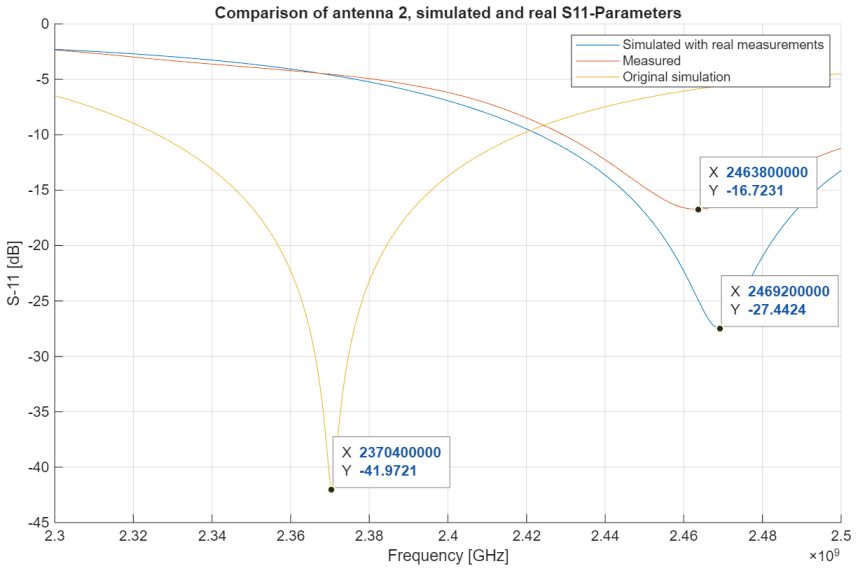


Figure 4.22: Comparison of the S-11 parameters of the original simulation and fabricated Antenna 2 as well as the simulation using the measurements of the fabricated antenna.

It can be seen through the simulation that the tolerances only result in a $\sim 10\,\text{dB}$ difference. This still leaves a difference of $\sim 10\,\text{dB}$, the reason for the remaining offset is not known. It should be noted that since the effect of these tolerances has been analyzed after the fabrication and testing of the antennas, the correct dielectric constant has not been used in the design of antenna 3.

4.1.6 Antenna 3

The resonant frequency of Antenna 3 is lowered in CST with approximately 21.45 MHz, because of the offset in Antenna 2, see Figure 4.20. The dimensions of Antenna 3 are shown in Figure 4.23.

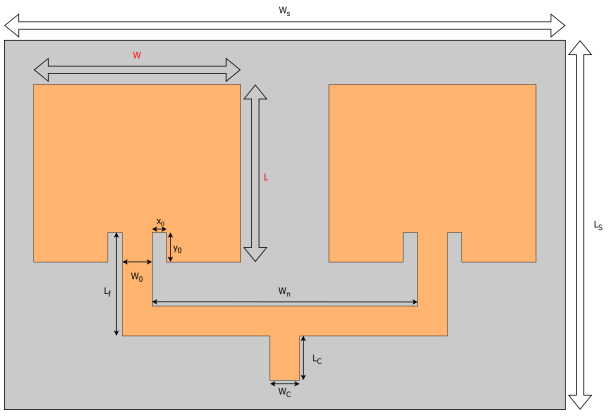


Figure 4.23: Diagram of Antenna 3, where the following dimensions changed to $W = 40.8\,\text{mm}$ and $L = 28.9\,\text{mm}$. The following dimensions are kept the same $x_0 = 1\,\text{mm}$, $y_0 = 5\,\text{mm}$, $L_f = 15\,\text{mm}$, $W_0 = 2\,\text{mm}$, $W_n = 58\,\text{mm}$, $L_C = 2.6\,\text{mm}$ and $W_C = 2\,\text{mm}$.

S11-parameters

Figure 4.24 shows the S11-paramters of the simulated and measured model. This test uses the

same procedure as the test journal in Appendix 3.

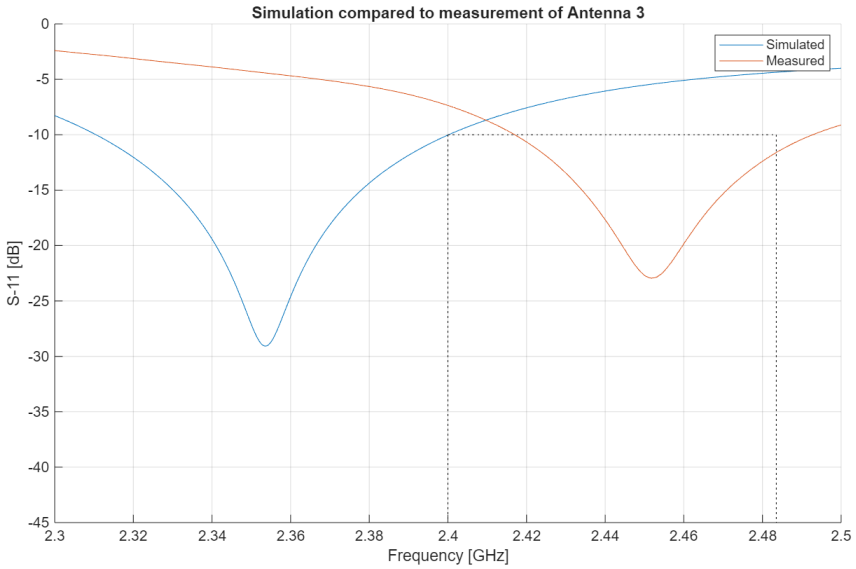


Figure 4.24: Comparison between simulation and measured S11-parameters of Antenna 3. The black dotted lines indicate the desired minimum frequency range, where the return loss is below -10 dB.

The Figure 4.24 shows a measured resonant frequency of 2451.8 MHz. The measured frequency range is from 2417 MHz to 2492.6 MHz, which corresponds to a bandwidth of 75.6 MHz. The specifications in Section 3.3.2, has a desired resonant frequency of 2441.75 MHz, and a bandwidth of at least 83.5 MHz. The measured performance of Antenna 3 closely aligns with the specified requirements in terms of the bandwidth, resonant frequency and frequency range.

Currently, the simulated gain of Antenna 3 is at 5.78 dBi, see Figure 4.25, which is close to the calculated gain of 5.82 dBi. To reach the required gain of 12.21 dBi, 9 elements are needed in the array.

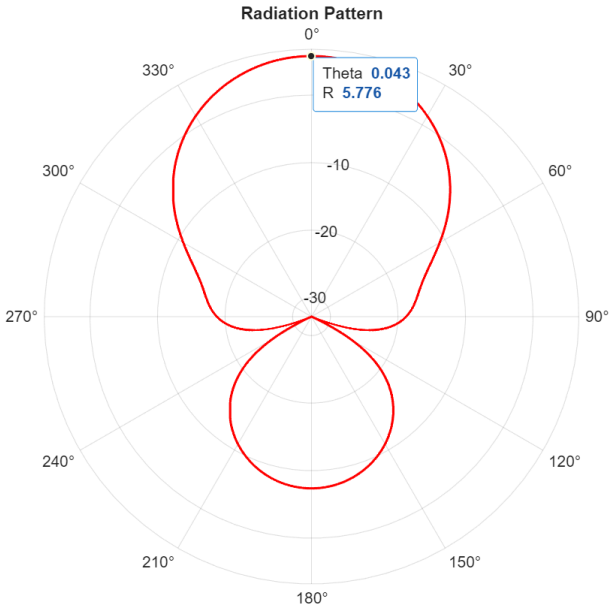


Figure 4.25: Simulation of the gain for Antenna 3.

Since the beamformer needs two ports to work, it is chosen to duplicate Antenna 3 and arrange them as shown in Figure 4.26. This also increases the gain of the antenna since there will be four elements in the array.

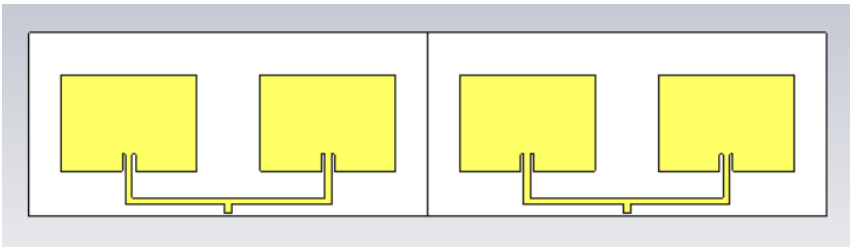


Figure 4.26: Duplicates of Antenna 3, placed next to one another, each having its own connector.

4.1.7 Antenna 3.1 and 3.2

As Antenna 3 closely aligns with the specifications, two identical antennas are constructed to allow for beamforming, as mentioned at the start of Section 4.1. Since it is known that the substrate of the FR4-PCB has varying ε_r values, the same FR4-PCB is used when manufacturing Antenna 3.1 and Antenna 3.2 in an attempt to minimize this variance. The dimensions of Antenna 3.1 and Antenna 3.2 are the same as Antenna 3, seen in Figure 4.23.

S11-parameters

The measured S11-parameters of Antenna 3.1 and 3.2 are shown in Figure 4.27, where they are compared with Antenna 3. This test uses the same procedure as the test journal in Appendix 3.

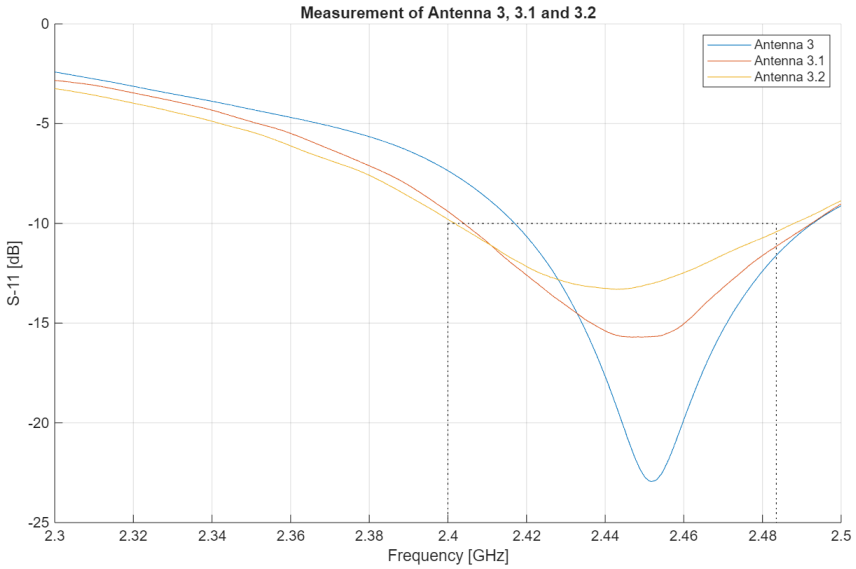


Figure 4.27: S11-parameters of Antenna 3.1 and 3.2 compared to Antenna 3, which is made from a different FR4-PCB. The black dotted lines indicate the desired minimum frequency range, where the return loss is below -10 dB.

It is observed in Figure 4.27 that the bandwidth of Antenna 3 is lower than the bandwidth for Antenna 3.1 and 3.2. The return loss in the resonant frequency of Antenna 3 is also lower than for Antenna 3.1 and 3.2. The differences in the antennas are probably due to the tolerances when milling the PCBs for the Antennas, as mentioned in Section 4.1.5. A summary of the resonant frequency and the bandwidth for Antenna 3, 3.1 and 3.2, compared to the specifications, is seen in Table 4.6.

Parameter	Specifications	Antenna 3	Antenna 3.1	Antenna 3.2
Min. frequency range	2400-2483.5 MHz	2417-2492.6 MHz	2404.1-2492.3 MHz	2402.3-2487.8 MHz
Bandwidth	$\geq$ 83.5 MHz	75.6 MHz	88.2 MHz	85.5 MHz
Resonant frequency	2441.75 MHz	2451.8 MHz	2449.4 MHz	2442.8 MHz

Table 4.6: Comparison between Antenna 3, 3.1 and 3.2 of the specifications for bandwidth, frequency range and resonant frequency.

Only minor differences are observed between Antenna 3.1 and Antenna 3.2, primarily in terms of the resonant frequency. As shown in the Table 4.6, both antennas meet the required bandwidth specification, while the frequency range and the resonant frequency are shifted slightly to the right. The small deviation is considered the best achievable under the given conditions and is therefore acceptable for the proof of concept prototype.

Gain

A test of the gain of the two antenna arrays is made to ensure that the antenna array increases the gain of the antenna as expected, compared to the single patch antenna, shown in Section 4.1.4.

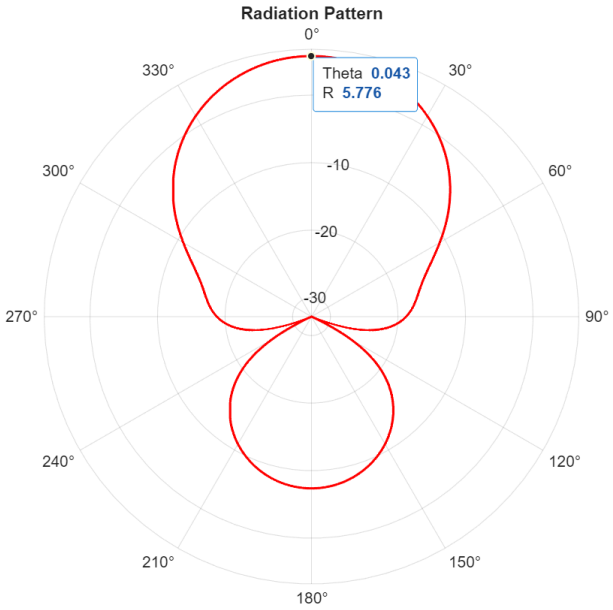


Figure 4.28: Simulation of the gain for Antenna 3. Theta represents the angle at which the gain is measured, while R denotes the gain in dBi.

A simulation in CST of the radiation pattern and gain is performed to get an estimate of the gain. The simulation can be seen in Figure 4.28, the simulated gain, seen as R in the Figure, is 5.78 dBi. A test is performed to ensure that the measurement matches the simulation. This test uses the same procedure as the test journal in Appendix 4

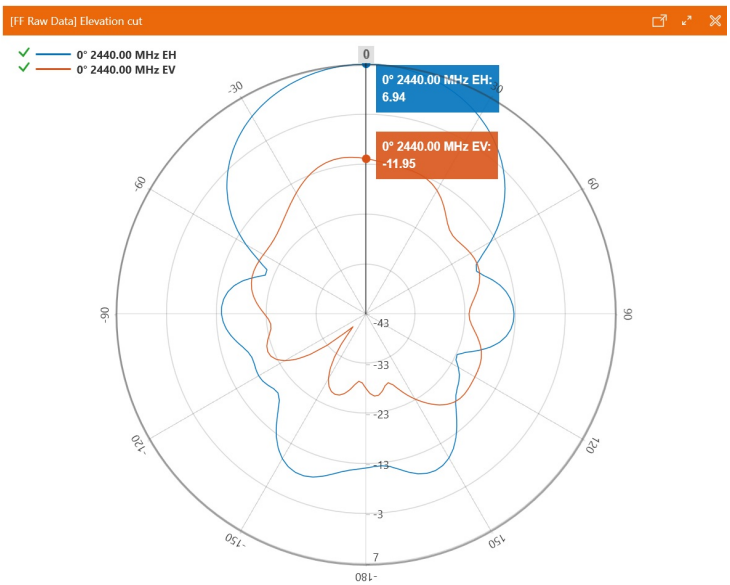


Figure 4.29: Test of the gain for Antenna 3.1. EH is the strength of the electric field in the horizontal plane, and EV is the strength in the vertical plane.

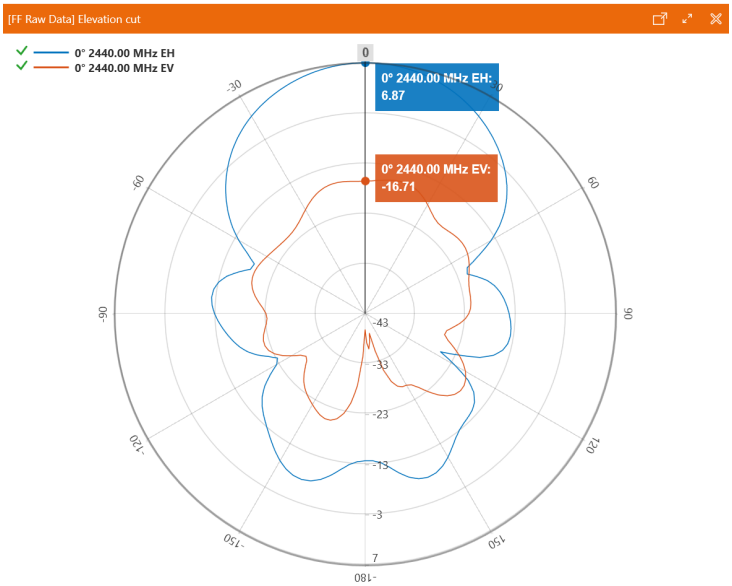


Figure 4.30: Test of the gain for Antenna 3.2. EH is the strength of the electric field in the horizontal plane, and EV is the strength in the vertical plane.

The gain of Antenna 3.1 is measured to be 6.94 dBi, it can be seen in Figure 4.29. The gain of Antenna 3.2 is 6.87 dBi, it is seen in Figure 4.30. The measurement shows a variation in the shape of the radiation pattern when compared to the simulation, and a much higher gain than the simulation. The reason for this discrepancy in the gain is unknown. However, it can be concluded that the gain is increased by using an array instead of a single patch antenna. Now that Antenna 3.1 and Antenna 3.2 have been designed and produced, they can be combined to create Antenna 4.

4.1.8 Antenna 4

Now that Antenna 3.1 and 3.2 have been tested individually, they are combined to create Antenna 4, as seen in Figure 4.26. The distance between each of the patches is $\frac{\lambda}{2} = 61.4\,\text{mm}$, which is based on the theory in Section 4.1.5. Using the gain from Antennas 3.1 and 3.2, a total gain for Antenna 4 can be found. In this case, the antenna gain from each of the individual antennas is added, since it gives a more precise total gain. First, the gain of each antenna is converted from dBi to linear gain.

$$10^{\frac{6.87\,[\text{dBi}]}{10}} = 4.86 \;\;\&\;\; 10^{\frac{6.94\,[\text{dBi}]}{10}} = 4.94$$

$$G_{total} = 6.94 + 6.87 = 9.8 \Rightarrow 10 \cdot \log_{10}(9.8) = 9.91\,\text{dBi}$$

The calculated linear gain using Equation (4.10) is expected to be 9.8, which is 9.91 dBi. Figure 4.26 shows the merging of the two subarrays, creating a new radiation pattern, with a main lobe and two side lobes shown in Figure 4.31.

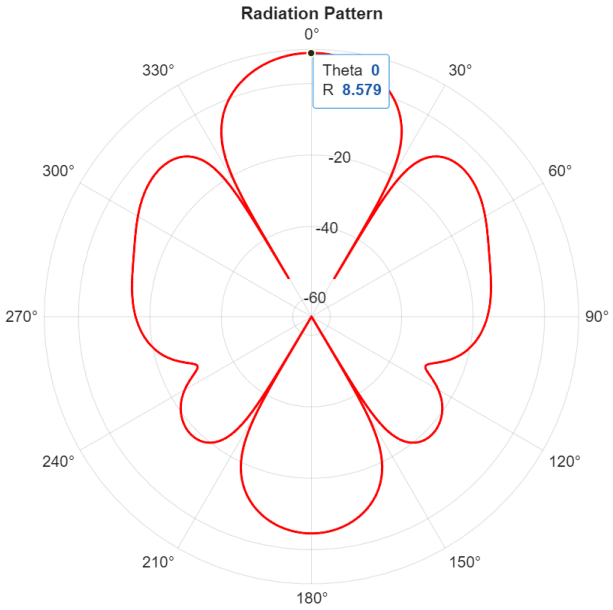


Figure 4.31: Radiation pattern of the coupled Antenna 3.1 and 3.2.

The simulated gain is 8.58 dBi, which is close to the calculated gain of 9.91 dBi. Since the Equation (4.10) does not take the impedance match into account and the simulation does, the simulated gain will therefore be lower. Antenna 4 is then tested to ensure that the simulation matches the measurement. As both of the subarrays, Antenna 3.1 and 3.2, are made to match 50 Ω, the ZFRSC-183-S+ power splitter is used as a combiner in the test of the gain [69]. It ensures that there is 50 Ω at the output port when combining the two antennas. The power splitter introduces an inherent 3 dB combining loss. Connecting the power splitter to the antennas adds a resistance of 25 Ω in series with the two antennas. The antennas are in parallel with approximately 50 Ω impedance each. This creates a voltage division, where half the signal is lost going from the power splitter to the antennas, resulting in the 3 dB loss. A 3D print was then constructed in order to place Antenna 3.1 and 3.2 next to each other as simulated in Figure 4.26. The test of the gain of Antenna 4 is seen in Figure 4.32. This test uses the same procedure as the test journal in Appendix 4.

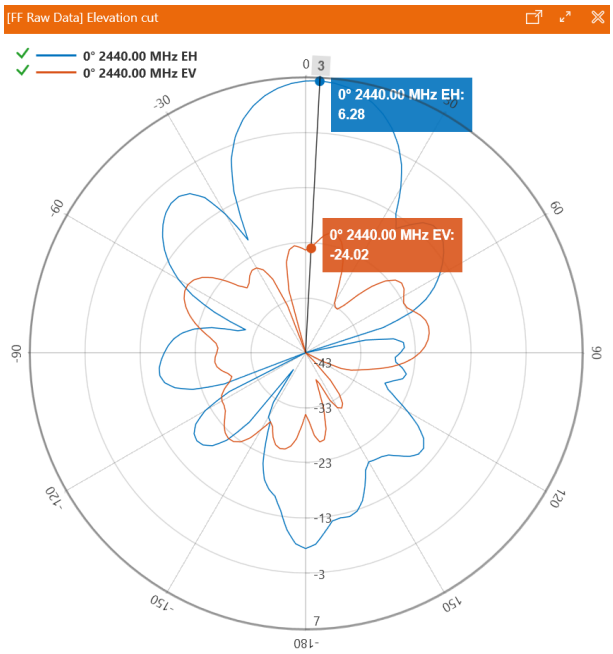


Figure 4.32: Gain test of the antenna array used for the prototype, consisting of Antenna 3.1 and 3.2. EH relates to the strength of the electric field in the horizontal plane, and EV is the strength in the vertical plane.

Figure 4.32 shows that the radiation pattern is not symmetrical. This is due to the radiation pattern and gain of Antenna 3.1 and Antenna 3.2 are not completely identical, as seen in Section 4.1.7. The tested gain is 6.28 dBi, factoring in attenuation in the power splitter, this results in an actual gain of 9.28 dBi, which is close to the simulated gain of 8.58 dBi. The deviation between the simulation and measurement is unknown.

To reach the specification of 12.21 dBi, a 3 dB increase would be needed. According to the Equation (4.10), this would mean the total number of elements required is 8. As the gain of Antenna 4 is 9.28 dBi $\approx$ 8.47 linear gain.

$$G_{total} = 8.47 \cdot 2 = 16.94$$

$$10 \cdot log_{10}(16.94) = 12.29\,\text{dBi}$$

In Section 4.1.5, it was calculated that the necessary number of elements in the array is 9. Based on the calculations with Antenna 4, one less element is needed. The difference could potentially be caused by the calculations in Section 4.1.5, which are based on Antenna 1, which is not impedance matched to 50 Ω, whereas Antenna 4 is. However, for a proof of concept, the current gain of 9.28 dBi is assessed to be good enough.

4.1.9 The Dielectric Constant of the FR4

The frequency offset seen in previous the S11-parameters shown in Figure 4.5, 4.8, 4.20, 4.24 and 4.27, is potentially correlated to the deviating dielectric constant. Figure 4.33 shows a simulation of Antenna 3, where the dielectric constant has been incremented from 3.8 to 4.8, with each step having a 0.1 higher dielectric constant.

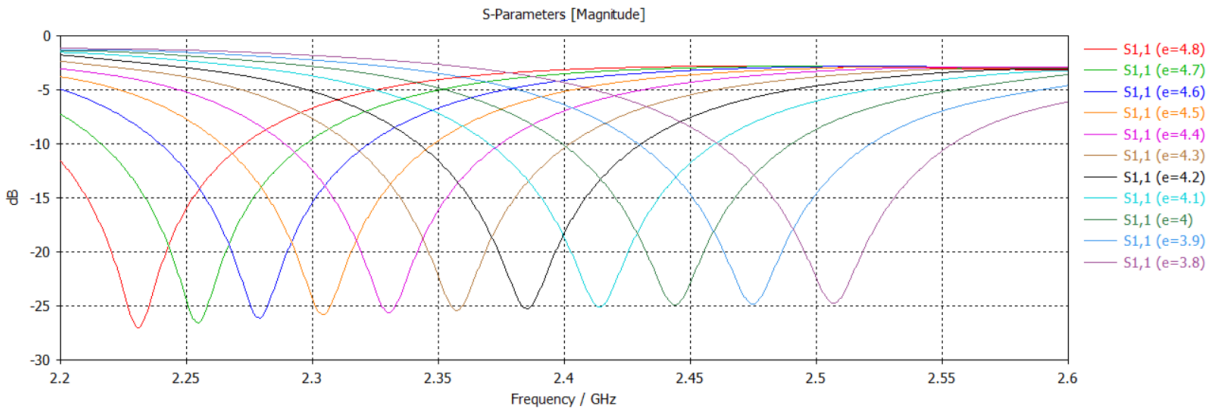
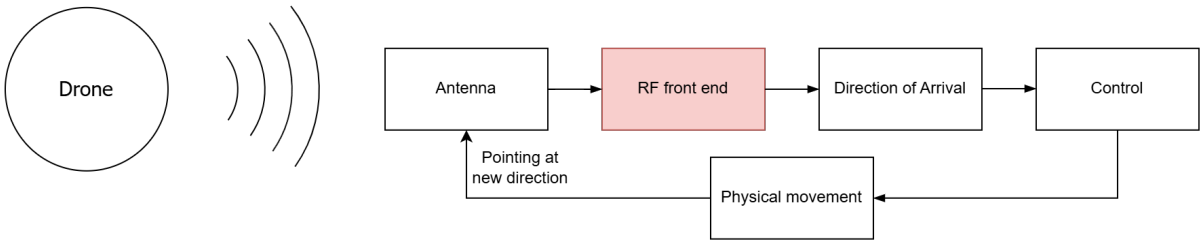


Figure 4.33: Different dielectric constant values for the substrate of Antenna 3.

It is shown in Figure 4.33 that the value of the dielectric constant impacts the frequency offset. Therefore, it is estimated that the dielectric constant of the substrate used for the FR4-PCB for Antenna 3 is closer to 4 or 3.9, based on the measurement compared to the simulation shown in Figure 4.24. A similar offset is seen for all the comparisons between the simulated and measured S11-parameters, including Figure 4.5, 4.8, 4.20, 4.24 and 4.27. Therefore, it is estimated that all the FR4-PCBs used in this project have a lower dielectric constant than the expected 4.3, mentioned in Section 4.1.3.

4.2 RF Front End



This section documents the general structure of a front end, and the selected architecture. The implementation is tested in accordance to the specifications, found in Section 3.3.3.

4.2.1 System Design

The goal of this system is to sample the output signals from the antennas. Certain challenges must be addressed to make this possible, two of these are related to the antennas and RF waves. The first problem is that the antenna is not strict frequency selective, causing them to receive unwanted signals alongside the intended ones. This creates a need for filters. The second problem is that other objects may send signals within the chosen frequency band.

However the second problem will be disregarded as the goal of this project is to prove the concept.

The goal of sampling the output of the antennas can be achieved using RF front end architectures, three architectures are chosen to be compared in this project, as these are the most fundamental architectures. These being Direct RF sampling architecture, intermediate frequency (IF) architecture, and Zero IF architecture [55].

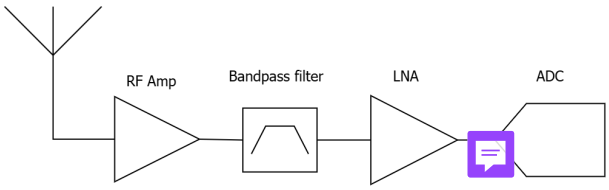


Figure 4.34: Diagram of a direct RF sampling architecture.

Direct RF sampling architecture is the most simple of the architectures. As the name indicates the RF signal is directly sampled, before being sampled is the signal amplified. The amplifications happens in a Low Noise Amplifier (LNA), the signal is then filtered using a bandpass filter. This architecture can be seen on Figure 4.34. This means that it involves less components than the other architectures, but these components are often more expensive as they have to operate at a higher frequency [70].

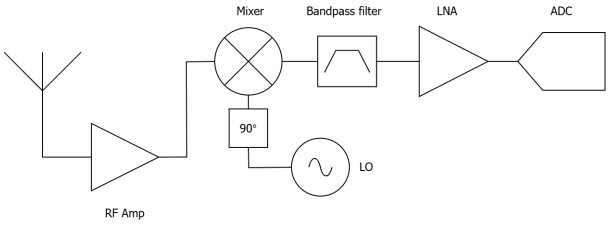


Figure 4.35: Diagram of a IF architecture.

The IF architecture works on the same principle as direct RF sampling architecture, but a key difference is that the IF architecture down scales the frequency of the RF signal. This is down scaled to an intermediate frequency, hence the name. This eases the specifications for the operating frequency of the other components than the LO [55]. This is seen on Figure 4.35. The RF signal is mixed with a signal from a local oscillator (LO), this produces two signals one at the sum of the frequency and one at the difference [71]. This means mathematical that the following holds true [72]:

$$f_{IF} = f_{LO} \pm f_{RF}$$

A visual representation is seen in Figure 4.36.

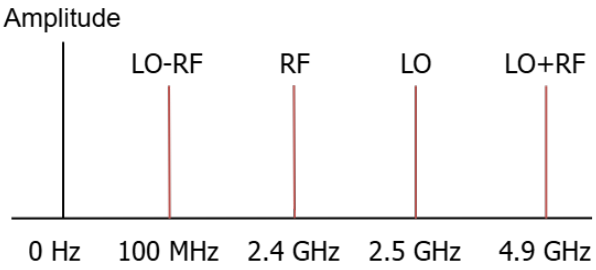


Figure 4.36: Diagram of the effect, when a mixer is used alongside a LO.

This also leads to the need of a bandpass filter to attenuated low and high frequency noise. As well as the high frequency output by the mixer. As with direct RF sampling, the signal is amplified and passed through a bandpass filter, before being sampled [55]. As can be seen on Figure 4.35.

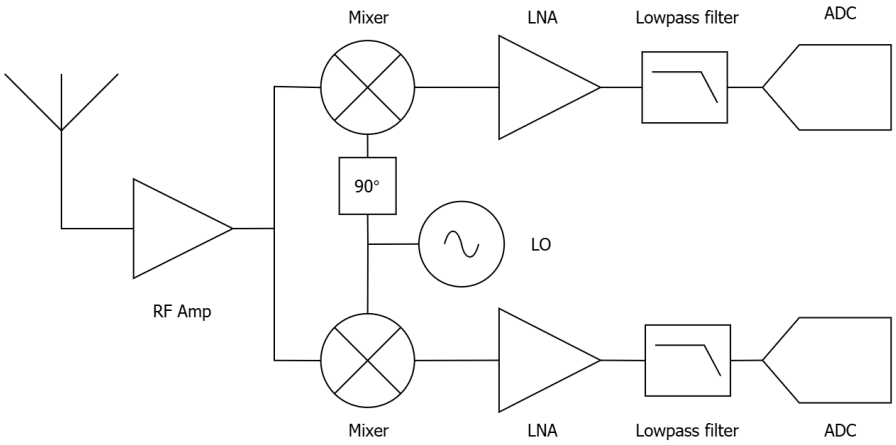


Figure 4.37: Diagram of a Zero IF architecture.

On Figure 4.37 the general structure for a Zero IF architecture is seen. The zero IF architecture takes the same general idea as an IF architecture but downmixes the frequency of the received signal down to DC, this is also done using an LO [73]. This means that there is a possibility of a negative frequency being the output of the mixer. This is not possible in the real world and this means that an image frequency will appear at the absolute value of the frequency [74]. This can be seen on on Figure 4.38.

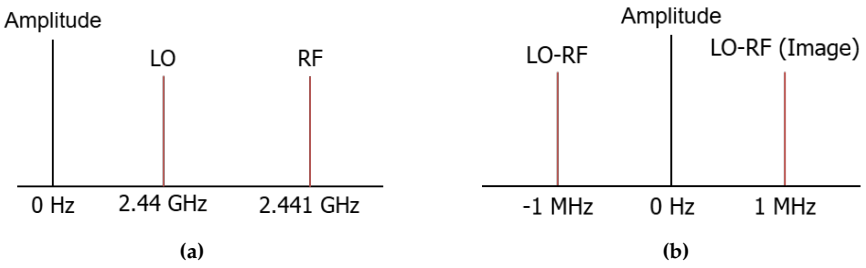


Figure 4.38: A visual representation of mixing a RF signal with a LO signal, 4.38a shows before mixing and 4.38b shows after mixing the signals.

This also changes the math to:

$$f_{IF} = |f_{LO} \pm f_{RF}|$$

Most zero IF architectures uses in-phase and quadrature (IQ) sampling. IQ sampling works by taking the output from the antenna into two different paths and phase shifting one of them 90°. The two signals which are sampled can be seen in Figure 4.39.

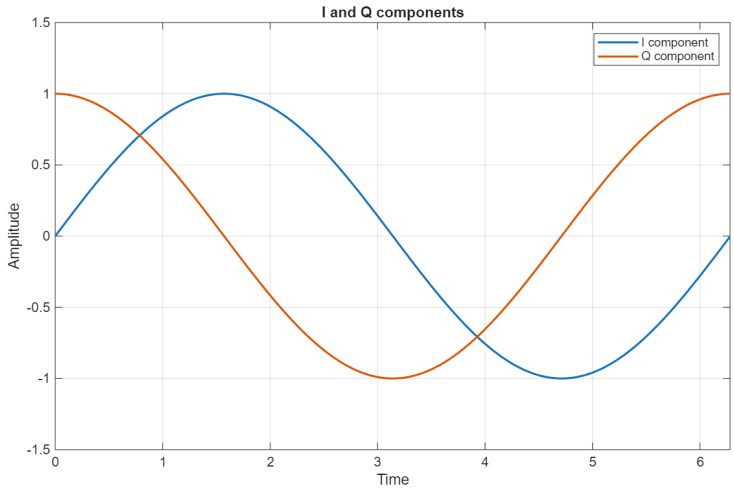


Figure 4.39: Diagram that shows the I and Q samples of a signal over time.

This makes sure that both frequency and phase information are sampled by the ADC. This makes it possible to lower the sampling rate to the highest frequency that carries information, after the down conversion [73].

For this project is the Zero IF architecture chosen, as it requires fewer components, in the filters, then the IF architecture [73] and because a SDR using this architecture is available. It should be noted that the direct RF sampling architecture requires the fewest amount of components, but these tend to be more expensive as they need to operate at a higher frequency [70].

4.2.2 Implementation

The focus of this project is not to make a front end, due to this a software-defined radio (SDR) is used. A SDR is a radio where the hardware components can be tuned using software. For this project, the NUAND bladeRF 2.0 micro A9 is the SDR of choice. This SDR is chosen as it uses an Zero IF architecture and is available at AAU [75]. The analog circuit of the BladeRF can be seen on Figure 4.40. Certain limitation comes when choosing an SDR over making the architecture from scratch. As the SDR is made to be applicable in multiple situations, it also has its limitations in customization. This is seen in the number of inputs, as the bladeRF only supports two receiver ports, which means a 2x2 antenna array is not possible as it would require four receiver ports [75]. Another limitation is the bandwidth of the bladeRF as it is limited to 56 MHz [75]. This is a problem as if the whole band is to be sampled in one, the needed bandwidth is 83.5 MHz. As mentioned in Section 3.3.3, the frequency band can be divided into multiple channels that can get sampled individually. The two channels would be $[2400\,;\,2441.75]$ MHz and $[2441.75\,;\,2483.5]$ MHz. This will double the amount of time it takes to sample the whole bandwidth, this was chosen not to be done as this would still prove

the concept of the prototype. It is therefore decided that this will not be implemented in the prototype. The frequency band is chosen to be symmetric around the resonant frequency as makes the gain on either side of it close to equal, this mean that the frequency band is $[2413.75\,;\,2469.75]$ MHz. The maximum standard sampling frequency of the BladeRF is 61.44 MHz. The bladeRF can however be overclocked to double the sampling rate changing the bandwidth to 122.88 MHz. Doing this has the downside of lowering the resolution of the ADC from 12 bits to 8 bits [76]. This is not done for this project as it is not needed to prove the concept of the prototype.

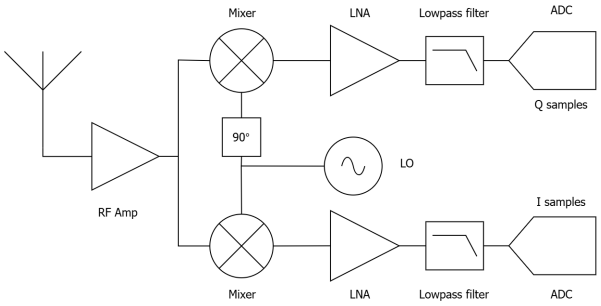


Figure 4.40: Diagram of an bladeRF analog part.

To program the bladeRF a python script is made that communicates with the analog part of the bladeRF, the analog circuit of the bladeRF can be seen on Figure 4.40. The control script for the bladeRF can be found in Appendix 17. The script that is used to read the data from the SDR is seen in Appendix 16. The parameters within the python script is set according to the discussion above. This means that the sampling rate is set to 61.44 MHz with a center frequency of 2441.75 MHz. The gain can however not be set according to the specification as the bladeRF has a maximum gain of 73 dB at 2300 MHz, when using two receiver ports [77]. According to the specifications the gain must be $59000 \approx 95.42$ dB. It should be noted that there is an option to activate automatic gain control in the bladeRF [75]. This is not enable as this could change the amplitude of the signals doing sampling. The number of samples is set to 8192, this is set as this corresponds to one buffer on each channel. This can however be changed later after the direction of arrival algorithm is made. It should be noted that the data saved from the python script is unitless as this is how the library from the bladeRF outputs the data. The data is also normalized to be between -1 and 1, by dividing with the number of possible voltage levels ($2^{12} = 2048$).

4.2.3 Output of Front End

Before the validation test could start, a problem was discovered in the output of the bladeRF. Appendix 8 shows a test revealing there is a large difference between the inputs of the two channels, which is caused by the internal LNAs amplifying the signals differently.

Since the output of the python program is unitless normalized values, this is converted to volt with Equation (4.11). Where x is the normalized unitless values and U is the data in volt. These are multiplied with $0.625\,\mathrm{V_p}$, as the maximum input of the ADC is $1.25\,\mathrm{V_{pp}}$.

$$U = x \cdot 0.625\,\mathrm{V} \qquad [\mathrm{V}] \quad (4.11)$$

While testing, it was found that the frequencies within the pass band have a large amplitude difference between channel 1 and channel 2, this can be seen on Figure 4.41.

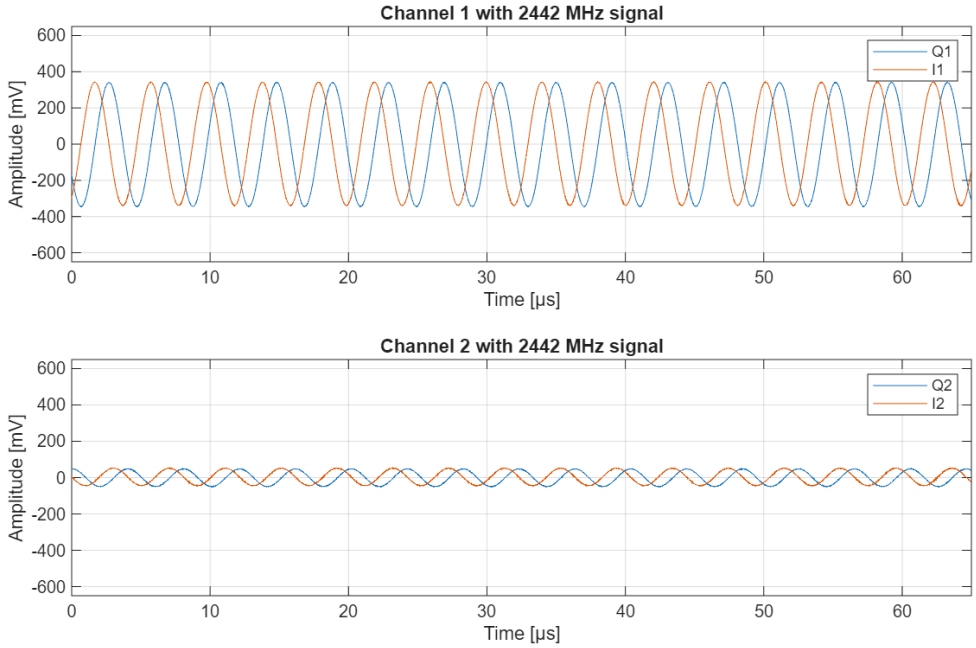


Figure 4.41: Measured data on the bladeRF, with a signal of 2442 MHz and an amplitude of 400 $\mathrm{mV_{pp}}$.

	I amplitude	Q amplitude
Channel 1	$[-343.63\,;\,347.6]$ mV	$[-348.51\,;\,343.93]$ mV
Channel 2	$[-48.52\,;\,55.85]$ mV	$[-53.1\,;\,52.8]$ mV

Table 4.7: IQ samples taken by the blade RF with signal being transmitted at a frequency of 2442 MHz.

As seen on Figure 4.41 and in Table 4.7 the amplitude on channel 2 is at most ≈ 6.22 times lower than channel 1. The reason for the difference in the amplitude of the two channels are unknown, but it is wanted that the amplitude of the two channels are the same. Another problem is found when compering the two channels amplitude to the noise measured, the noise can be seen on Figure 4.42.

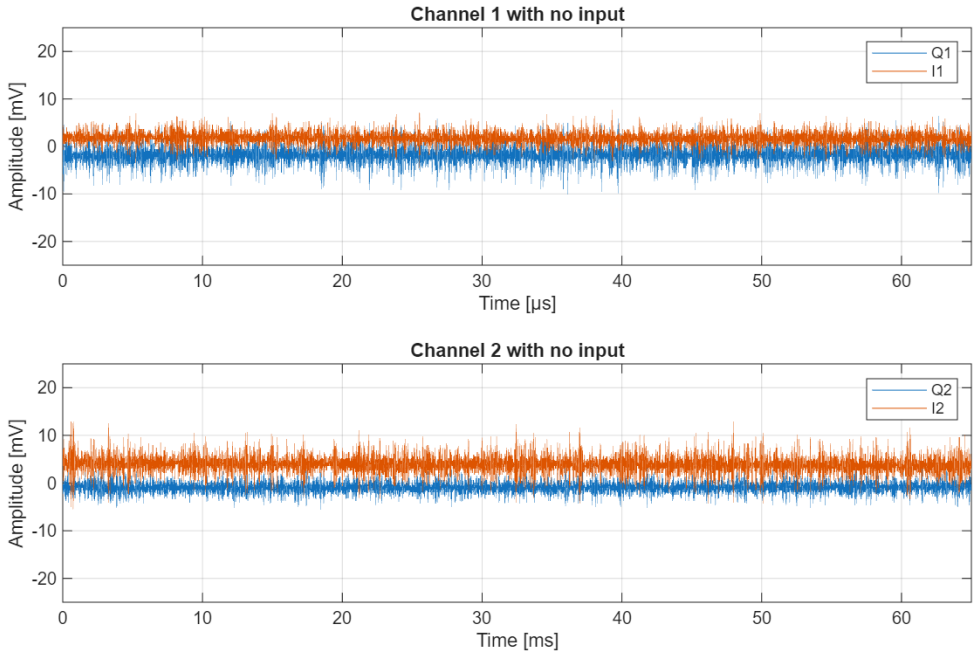


Figure 4.42: Measured noise on the bladeRF, with no signal input.

As it can be seen on Figure 4.42, the maximum amplitude of the noise on channel 2 is 12.82 mV. This is a problem as this means that the SNR on channel 2 is 17.84 dB. This is a problem as the signals outside the passband are attenuated down to the noisefloor. The maximum and minimum amplitudes that where measured for them, can be seen on Table 4.8.

	I amplitude	**Q amplitude**
Channel 1	$[-4.88\,;\,8.85]$ mV	$[-10.99\,;\,7.63]$ mV
Channel 2	$[-5.19\,;\,12.51]$ mV	$[-5.49\,;\,3.97]$ mV

Table 4.8: IQ samples taken by the blade RF with signal being transmitted at a frequency of 2500 MHz.

Note, the sample rate has been set to 40 MHz down from 61.44 MHz. This is done as the bladeRF library flagged a risk of data loss at the original sample rate. This means that bandwidth will be reduced to 40 MHz, but it should still be sufficient for POC.

Gain calibration

In order to ensure that the two channels have the same amplitude, the gain is calibrated manually. This is possible as the two LNA, with in the SDR, are independent of each other. The gain is tuned, by stepping the gain 5 dB at a time, through trial and error until the signals have the same amplitude. This results in channel 1 having a gain of 0 dB and channel 2 a gain of 20 dB. Table 4.9 contains the amplitude sampled signal after the gain has been changed.

I components	**Q components**
$[-389.71\,;\,395.81]$ mV	$[-392.79\,;\,390.62]$ mV

Table 4.9: Ampltide of I and Q components of the gain calibrated signal on channel 2.

4.2.4 Front End Validation Test

To validate that the front end is operating as expected a test is made, where the goal is to validate the pass band, down conversion and attenuation. This is tested by feeding a signal generator's output directly into the Rx1 and Rx2 ports on the bladeRF. This is tested for three different inputs: a generated signal inside the pass band, one that is outside the pass band, and one where no signal is inputted. The test uses the same method as described in Appendix 8. As explained in Section 4.2.3 is the test conducted with a sample rate of 40 MHz, and the gain on channel 2 to 20 dB. This is as explained above done with the same procedure as in Appendix 8.
As the gain should be 101.54 dB, but this is not possible on the bladeRF will this not be tested. The input impedance is also not tested as it is stated in the datasheet that it is 50 Ω [77].

It is found through the test that the input with frequency 2442 MHz, has maximum and minimum voltage levels seen on Table 4.10, with the sampled signals seen on Figure 4.43.

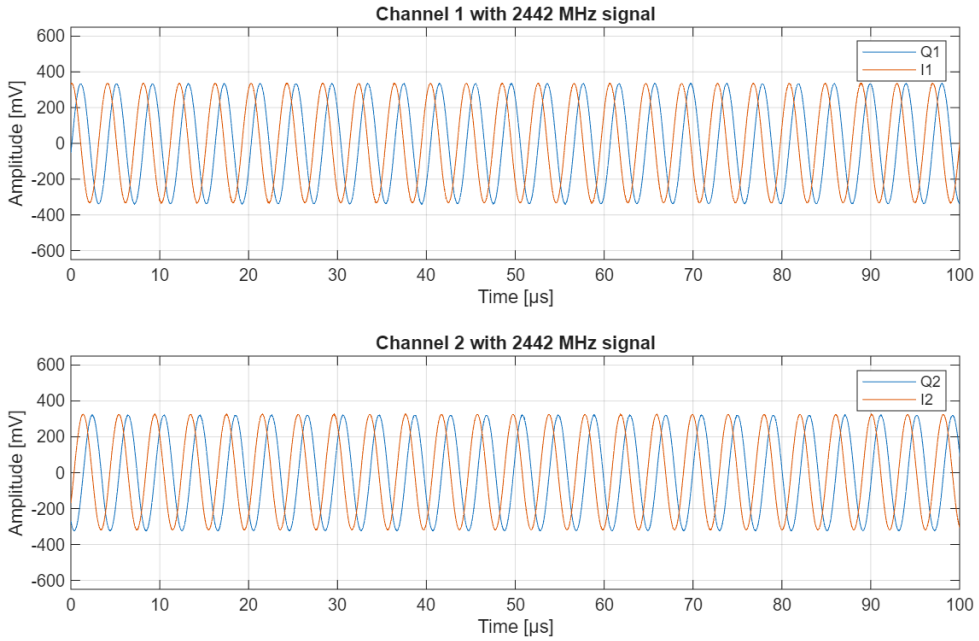


Figure 4.43: Measured IQ data on bladeRF with an input signal with a frequency 2442 MHz.

	I amplitude	Q amplitude
Channel 1	$[-337.52\,;\,339.66]$ mV	$[-342.41\,;\,338.13]$ mV
Channel 2	$[-321.35\,;\,328.06]$ mV	$[-326.54\,;\,325.93]$ mV

Table 4.10: IQ samples taken by the blade RF with signal being transmitted at a frequency of 2442 MHz, with gain adjustment.

It is seen that there is only a small difference between channel 1 and 2, which is desired. Neither of the channels reach 400 mV as expected. This might be due to internal loss in the system.

The noise generated by the bladeRF is also measured it was found that the voltage level is between -10.68 mV and 11.90 mV on both channels. This means that by using Equation (4.12), the SNR on each channel can be found.

$$dB_{\Delta} = 20 \cdot \log_{10}(\frac{V_{Input\ RMS}}{V_{Noise\ RMS}}) \quad \text{[dB]} \quad (4.12)$$

This means that the SNR on channel 1 is 40.44 dB while on channel 2 being 35.02 dB using 400 mV as the signal. This difference is due to the noise not being the same on both channels and the maximum amplitude with a signal being passed not being the same.

To test the attenuation specification a signal is input with a frequency of 2.5 GHz and an amplitude of 400 mV_{pp}. This results in a measured amplitude of -11.90 mV and 12.82 mV. When calculating the the attenuation of the signals outside the frequency band, can the Q components be ignored as the phase of a signal does not affect the attenuation. This mean that I components, can be converted to to Root Mean Square (RMS) values. The difference of the 2442 MHz and 2500 MHz signal outputs can then be calculated in decibel (dB) using Equation (4.12). This is done for both channels and gives a attenuation outside the pass band of respectively 40.82 dB and 35.13 dB. Where the specification was that it should be 60 dB. The difference in SNR is due to the sampled signal amplitudes not being the same on both channels. One reason for there not being a attenuation of 60 dB, is that the signals outside the frequency band are already attenuated down to the noise floor. This means that by increasing the gain of the LNAs the attenuation outside the frequency band should also increase.

Down conversion

An FFT is made based on the sample signals to ensure that the down conversion works as intended, this is done in matlab using a Hanning window with a size of 4096 (The same size as the data set). The FFT has a size of 16384 This can be seen on Figure 4.44 and 4.45.

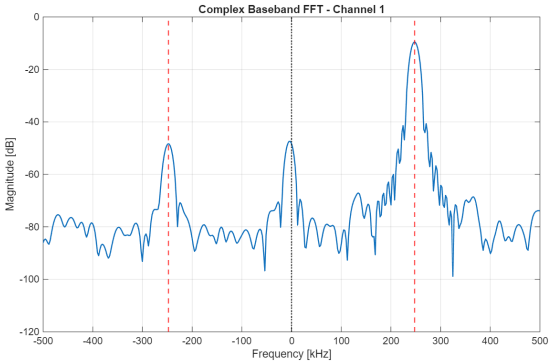


Figure 4.44: FFT on channel 1 using a Hanning window. With the following frequencies marked -247.75 kHz (marked by a red dotted line), 0 MHz (marked by black dotted line) and 247.75 kHz (marked by a red dotted line).

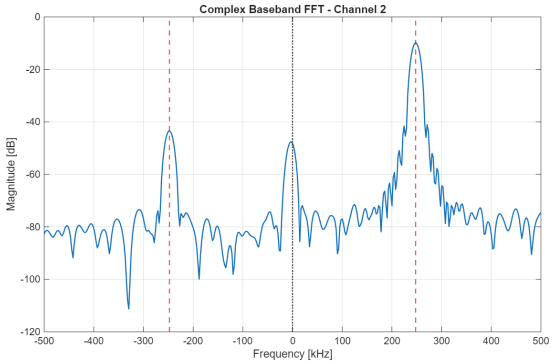


Figure 4.45: FFT on channel 2 using a Hanning window. With the following frequencies marked -247.75 kHz (marked by a red dotted line), 0 MHz (marked by black dotted line) and 247.75 kHz (marked by a red dotted line).

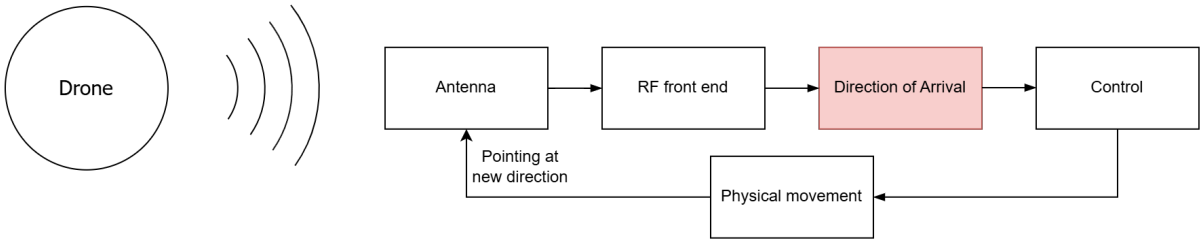
It is seen that on both of the channels, the signal is not down converted to 250 kHz as expected, but at $\approx$ 247.75 kHz. The reason for this is not known, but a hypothesis is that is due to tolerances in the LO. On Figure 4.45 and Figure 4.44 the negative frequency component is seen at the expected frequency of -247.75 kHz and a component slightly negative offset from 0 Hz.

A summary of the fulfilled specifications from Section 3.3.3 can be seen in Table 4.11. It can be seen through this summary that most of the specifications are not fulfilled. Many of these specifications are not strictly needed to prove the concept of the prototype.

Parameter	Specification	Achieved values	Fulfilled
Amplifier gain	59000 $\approx$ 95.42 dB	0 dB in current state	X
ADC Sample rate	83.5 MHz	40 MHz	X
ADC resolution	$\geq$ 4 bits	12 bits	✓
Pass band	[2400 ; 2483.5] MHz	[2413.75 ; 2469.75] MHz	X
Attenuation outside pass band	60 dB	35.13 dB on channel 2	X
Input impedance	50 Ω	50 Ω	✓

Table 4.11: Summary of validation of specification of front end.

4.3 Beamforming and Direction of Arrival Estimation



To identify the direction of arrival of an incoming signal, there are several algorithms that can be implemented, one of the simpler ones is called a delay-and-sum (DAS) beamformer [78]. It can be implemented digitally, so for the proof of concept, this is chosen. This technique is used on a ULA, where one antenna element is chosen as the reference. A range of phase delays are added to the signals obtained on subsequent elements, and then cross-correlated with the signal received by the reference element [79]. The phase delay with the largest correlation to the reference is the angle of arrival.

To be able to implement and simulate the algorithm digitally, the math behind DAS beamforming is detailed. The following model assumes all antennas are isotropic and that the transmitter is in far field.

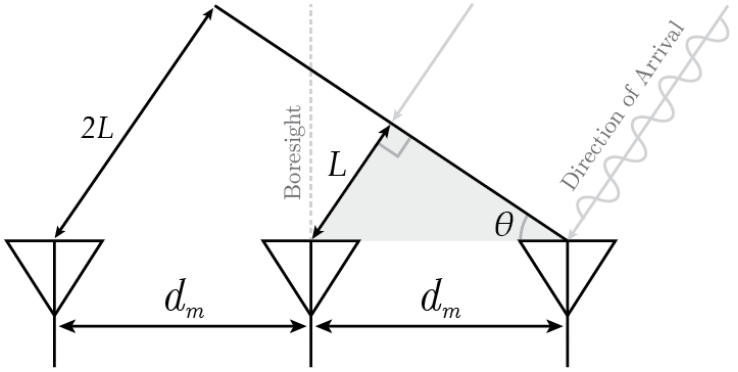


Figure 4.46: An illustration of a sinusoidal incident plane wave, hitting a one dimensional ULA.

On Figure 4.46 a model can be seen of a 1D ULA, with three elements. The incident waves' angle of arrival is described by θ. d_m is the distance between each array element in meters. The length L is the distance the incident wave has to travel, to hit the second array element. This length can be found, using the following relation.

$$\frac{L}{d_m} = \sin(\theta) \;\Rightarrow\; L = d_m \cdot \sin(\theta) \tag{4.13}$$

With a known length, the time difference between the incident wave hitting the reference element, and the second element, can be found by dividing with the speed of light.

$$\Delta t = \frac{d_m \cdot \sin(\theta)}{c} \tag{4.14}$$

Here Δt is the time delay for the subsequent element, to be in phase with the reference antenna. Where each k'th element will subsequently receive the signal $k \cdot \Delta t$ later. The transmitted signal is a base-band signal $x(t)$ upconverted by a complex carrier wave $e^{j2\pi f_c t}$. Thus the signal hitting the element adjacent to the reference $k = 1$, can be described by:

$$f_{rx_1}(t - \Delta t) = x(t - \Delta t)e^{j2\pi f_c(t-\Delta t)} \tag{4.15}$$

To sample the signal on the receiver side, a downconversion must be made to extract the base-band signal, as noted in Section 4.2.1. This downconversion is done by multiplying the complex conjugate of the carrier wave, which is $e^{-j2\pi f_c t}$. By doing this, the received downconverted signal can be modelled as such. In this project, this is handled by the analog local oscillator in the RF receiver on the SDR.

$$r_1(t - \Delta t) = f_{rx_1}(t - \Delta t) \cdot e^{-j2\pi f_c t} = x(t - \Delta t) \cdot e^{-j2\pi f_c \Delta t} \tag{4.16}$$

Since the beamforming will happen in discrete time, switching out Δt in the exponent of e, with the right term of Equation (4.14) makes it easier to separate from continuous time.

$$r_1(nT - \Delta t) = r_1(t - \Delta t)|_{t=nT} = x(nT - \Delta t) \cdot e^{-j2\pi f_c d_m \sin(\theta)/c} \tag{4.17}$$

Δt can be seen as insignificant as long as the sampling period T is significantly larger, which it will be as long as the sampling frequency f_s is significantly numerically smaller than c. Comparing the sampling frequency of the SDR $61.44 \cdot 10^6$ Hz with $3 \cdot 10^8$ m/s, this relation holds true. Disregarding Δt means the Equation (4.18) reduces to the discrete function, because of the relation $x(nT) = x[n]$ [80].

$$r_1(nT) = r_1[n] = x[n] \cdot e^{-j2\pi f_c d_m \sin(\theta)/c} \tag{4.18}$$

Equation (4.18) describes the received signal at the second array element with the reference element being to the right of it. It would be possible to implement the discrete function (4.18), but to reduce variables, it should be simplified.

Now using the relation $f = c/\lambda$, f_c is swapped out to cancel out the division.

$$r_1[n] = x[n] \cdot e^{-j2\pi d_m \sin(\theta)/\lambda} \tag{4.19}$$

A new division occurs, which is then removed by normalizing the distance with respect to the wavelength of the carrier signal, this is done with the relation $d = d_m/\lambda$.

$$r_1[n] = x[n] \cdot e^{-j2\pi d \sin(\theta)} \tag{4.20}$$

While a small change, removing the negative operator in the exponent, has the effect that instead of indexing elements k from right to left, they are indexed from left to right. But it removes the need for a signed value in the exponent [81].

$$r_1[n] = x[n] \cdot e^{j2\pi d \sin(\theta)} \tag{4.21}$$

The signal received at subsequent array elements can be described by simply multiplying d by k where k is the element number, with the reference element being $k = 0$. This is what is called the steering vector $\mathbf{s}$ [81].

$$\mathbf{s} = \begin{bmatrix} e^{j2\pi 0 d \sin(\theta)} \\ e^{j2\pi 1 d \sin(\theta)} \\ \dots \\ e^{j2\pi k d \sin(\theta)} \end{bmatrix} = \begin{bmatrix} 1 \\ e^{j2\pi 1 d \sin(\theta)} \\ \dots \\ e^{j2\pi k d \sin(\theta)} \end{bmatrix} \tag{4.22}$$

By multiplying the discrete time base-band signal $x[n]$ with the steering vector, the reference element (first row) in the resulting vector $\mathbf{r}[n]$, will always be the unmodified discrete base-band signal. $\mathbf{r}[n]$ describes the received signal as it is sampled by the ADC in the RF front end.

$$\mathbf{r}[n] = x[n] \cdot \mathbf{s} = x[n] \begin{bmatrix} 1 \\ e^{j2\pi 1 d \sin(\theta)} \\ \dots \\ e^{j2\pi k d \sin(\theta)} \end{bmatrix} \tag{4.23}$$

From the sampled signal, the angle of arrival can be extracted by multiplying a weight vector $\mathbf{w}(\theta)$, which for a DAS beamformer, is equal to the complex conjugated steering vector $\mathbf{s}^*(\theta)$, with the incoming signal $\mathbf{r}[n]$, and then sums all of the outputs $y[n]$ together [82]. This is mathematically expressed in equation 4.24.

$$y[n] = |\sum_{k=0}^{k-1} s_k^* \cdot \mathbf{r}[n]| = |\sum_{k=0}^{k-1} w_k \cdot \mathbf{r}[n]| \tag{4.24}$$

To find θ_1, a brute force search is done by sweeping θ_2 and summing for each increment, then the θ_2 resulting the largest $y[n]$ is returned. This is called cross-correlation, and is most easily explained taking basis in equation 4.24 and a system of 2 antennas (but can theoretically be expanded to any k):

$$y[n] = |\sum_{k=0}^{k-1} w_k \cdot r[n]| = |\sum_{k=0}^{k-1} x[n] \cdot e^{j2\pi kd(sin(\theta_1)-sin(\theta_2))}|$$

For simplicity sake, $x[n]$ is set to 1, but the proof is identical for other values:

$$y[n] = |1 \cdot (e^{j2\pi 0d(sin(\theta_1)-sin(\theta_2))} + e^{j2\pi 1d(sin(\theta_1)-sin(\theta_2))})| = |e^0 + e^{j2\pi d(sin(\theta_1)-sin(\theta_2))}|$$

$$= |1 + e^{j2\pi d(sin(\theta_1)-sin(\theta_2))}|$$

This can never return more than 2 and only returns that when $\theta_1 = \theta_2$. If that condition is not met, then imaginary parts exist, and the absolute value will always be less than 2 [83].

4.3.1 Initial Implementation

The implementation is a modified version of the DAS beamformer code presented in the *Beamforming & DOA* chapter of the PySDR textbook [81]. It is implemented in Python, due to ease of interfacing with the SDR. It could be implemented on an FPGA or directly in C, which would reduce compute time [84]. To reduce the impact Python has on performance, the NumPy library is used as it utilizes C code to speed up calculations [85].

```python
def beamforming_das(rx, distance, no_ele):
    theta_sweep = np.linspace(-1*np.pi/2, np.pi/2, 1000)
    results = []
    for thetas in theta_sweep:
        w = np.exp(2j * np.pi * distance * np.arange(no_ele) * np.sin(thetas))
        y = w.conj().T @ rx
        results.append(np.abs(np.sum(y)))
    return np.rad2deg(theta_sweep[np.argmax(results)])
```

Figure 4.47: Delay and sum beamformer function implementation in Python

The beamformer function (Figure 4.47) takes in three parameters, the sampled signal data *rx*, the distance between elements in wavelengths normalized to the carrier frequency *distance*, and the amount of antenna array elements *no_ele*. The data set is a matrix, where the rows correspond to receiver channels, and the columns is the sample size. The beamformer function

then successively takes each row of the data matrix, sweeps through a defined linear distribution of θ between $-90°$ and $90°$ or $-\pi/2$ and $\pi/2$ in radians, and multiplies the weight vector with the data vector for each angle in the distribution. It then sums all elements in a single vector containing the correlation of the two signals, takes the absolute value of the resulting complex number, and stores it in an array. The linear distribution allows for more or less angle steps, it is chosen to be 1000 initially for testing general functionality of the algorithm.

4.3.2 Simulation Testing

Since the antenna array has 2 elements spaced $\lambda/2$ apart, the beamformer will have directional ambiguity when a signal arrives from either $-90°$ or $90°$. The antenna pattern for a $90°$ DAS beamformed ULA, like the one described, is plotted on Figure 4.48. It shows that it is not possible to definitively tell where the antenna main lobe is pointed, in contrast to the 2-element array directed in boresight, seen in Figure 4.15.

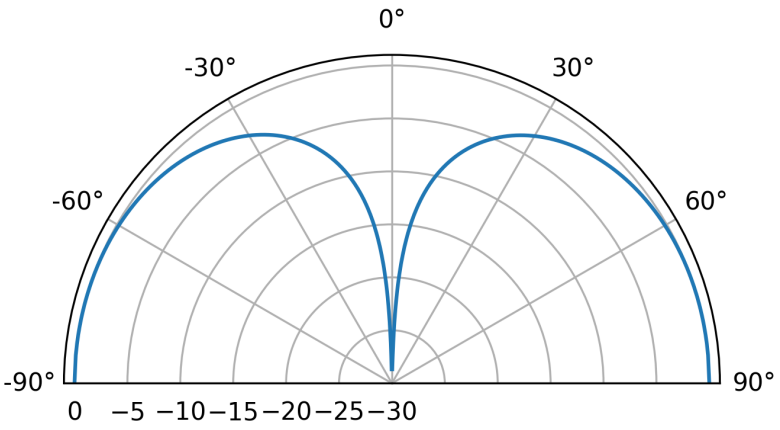


Figure 4.48: Radiation pattern of a $90°$ DAS beamformed ULA with 2 elements. Showing equal gain at both $-90°$ & $90°$.

Despite the use case specifying a scanning angle space of $120°$, or $-60°$ to $60°$. A choice is made to define the angular space for simulation and testing as $-89.9°$ to $89.9°$, as this avoids ambiguity with ideal signals, and allows for assessment of the algorithm's potential outside the specified range.

To test the algorithm, a base-band test signal is simulated, based on the formulas in the theory part of this section. This signal is a $1\,V_{pp}$, $20\,kHz$ sine wave with a variable angle of arrival. Furthermore the up- and downconversion of the signal by a carrier frequency, has not been simulated, as they are merely opposing multiplications. A two-element receiver array has also been simulated, by applying the steering vector to the test signal, to simulate a received signal. The chosen distance between the elements is half a wavelength with respect to the carrier frequency, which is ideally the same as the resonant frequency of the antennas used, as explained in Section 4.1. All tests in this section are done on a Intel Core Ultra 9 185H processor. The code for the simulated sine wave is seen in Figure 4.49.

```
### PARAMETERS
sample_rate = 61.44e6
N = 10000 # sample size
t = np.arange(N)/sample_rate # time vector

### TX SIGNAL
theta_deg # reference AoA
theta_rad = np.deg2rad(theta_deg)
f = 2e4 # base-band signal frequency
tx = np.exp(2j * np.pi * f * t) # base-band signal
d = 0.5 # distance between elements
elements = 2 # number of elements in the array
k = np.arange(elements)

s = np.exp(2j * np.pi * d * k * np.sin(theta_rad)) # steering vector
s = s.reshape(-1,1) # row -> column
tx = tx.reshape(1,-1) # column -> row

### RX SIGNAL
X = s @ tx # received signal matrix
```

Figure 4.49: Code for Simulating the received signal

For the functionality test, *theta_deg* is sweeped through all whole numbers, from $-89°$ to $89°$, with $\pm 89.9°$ appended. Henceforth theta_deg is referenced as θ_{Ref}, and the angle of arrival returned from the algorithm, θ_{Meas}. If the beamformer functions correctly, it should return θ_{Meas} corresponding to θ_{Ref}. This test is done using sample size $N = 10000$. Test code is found in Appendix 19.

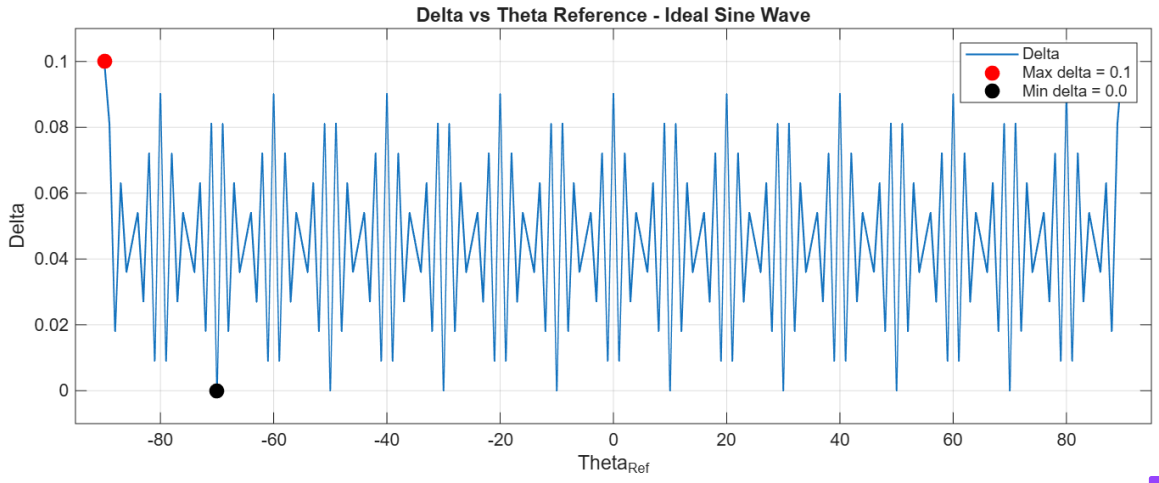


Figure 4.50: Line chart with θ_{Ref} is on the x-axis, and absolute difference Δ between θ_{Ref} and θ_{Meas} on the y-axis. Done with no noise. The largest and smallest Δ are highlighted.

The data plotted in Figure 4.50 confirms that the beamformer is functional, but is not accurate enough to meet the $2.28° \cdot 10^{-3}$ accuracy specification, at a maximum delta of $0.1°$. This is simply a quantization problem, as 1000 is too few angle steps, to attain the accuracy specification. The amount of angle steps needed is at least $180/2.28 \cdot 10^{-3} \approx 78948$. Which is confirmed by repeating the test using this amount of angle steps. The test code found in Appendix 19 is modified, swapping out 1000 for 78948 in line 5. There also appears to be oscillation like

symmetry in the data, the cause of which is unknown.

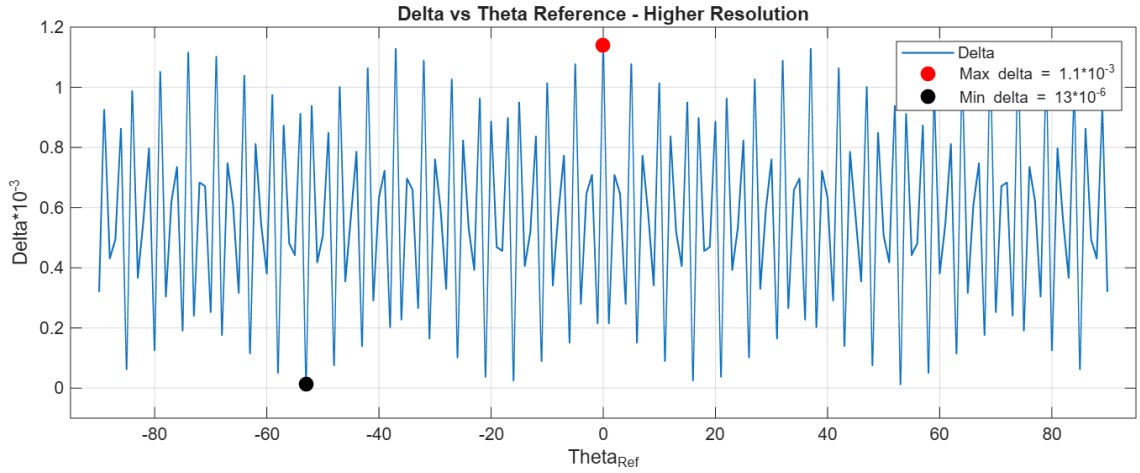


Figure 4.51: Line chart with θ_{Ref} is on the x-axis, and absolute Δ between θ_{Ref} and θ_{Meas} on the y-axis. Done with 78948 angle steps and no noise. The largest and smallest values are highlighted.

As is shown on Figure 4.51, Δ becomes low enough to meet the accuracy specification of $2.28° \cdot 10^{-3}$ when increasing angle steps, however compute time increases. The beamforming was significantly slower at mean 9.694 seconds to compute, compared to the 0.111 seconds the first test took, therefore not within specification. A way to reduce the time taken, would be to either reduce the angle steps again, or the amount of samples to be processed. Because of the accuracy specification, reducing angle steps is not viable. The algorithm requires at least $N = 1$ to beamform. With $N = 1$ two IQ-samples are obtained, one for each channel. When this is the case there will only be one θ within the scanned angle space, where the two samples correlate, resulting in maximum amplitude. This is verified by repeating the test from Figure 4.51 setting $N = 1$. The test code from Appendix 19 is used again, keeping the changes to angle steps from before, and setting $N = 1$ in line 15.

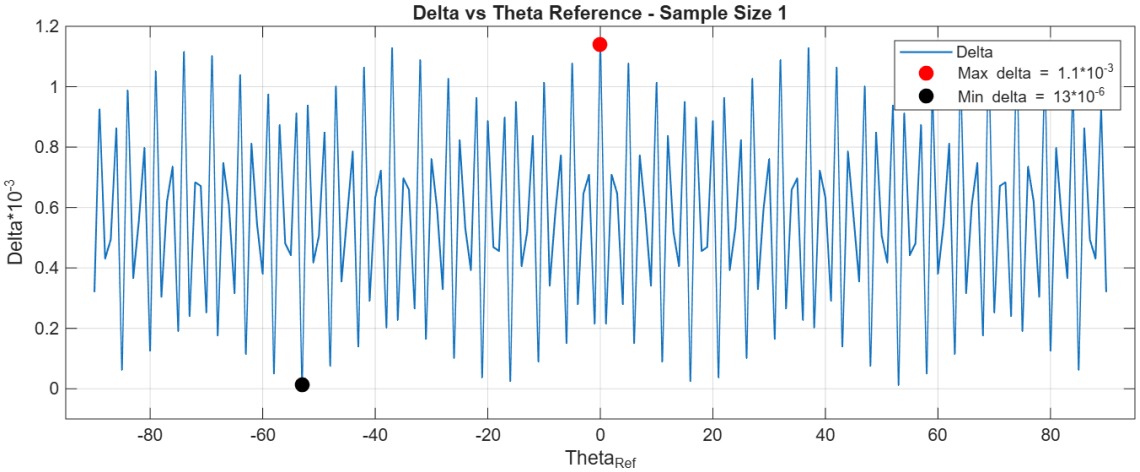


Figure 4.52: Line chart of data obtained for $N = 1$ with no noise. θ_{Ref} on the x-axis representing the actual angle of arrival. The y-axis shows the mean variation between the calculated angle (θ_{meas}) and the actual angle (θ_{ref}). The largest and smallest mean deltas are highlighted.

The data shown on Figure 4.52, shows that at a max measured delta of $1.14° \cdot 10^{-3}$ the beam-

former still stays within the specifications when $N = 1$. It also computes significantly faster, at 0.501 seconds on average, over 181 measurements.

While the beamformer functions in ideal conditions, in reality noise will occur. To this end, the beamformer is tested using a noisy signal as well. The noise used is additive white gaussian noise (AWGN) on both the real and imaginary part, with generated values for each element and each sample. Likewise a function has been made that converts SNR into a peak amplitude, that is then multiplied onto the AWGN, to get the right ratio. To generate a noisy signal the code in Figure 4.53, is appended to the code in Figure 4.49.

```
1 def snr_to_peak_amplitude(SNR):
2     snr_lin = 10 ** (SNR / 20)
3     a_noise_rms = 1 / np.sqrt(snr_lin)
4     a_noise = a_noise_rms * np.sqrt(2)
5     return a_noise
6
7 n = np.random.randn(elements, N) + 1j*np.random.randn(elements, N)
8 X_n = X + snr_to_peak_amplitude(snr)* n
```

Figure 4.53: Code simulating AWGN noise on received complex samples.

To see how noise affects the functionality of the beamformer, different signal-to-noise ratios are tested. Like the tests above, it steps θ_{Ref} by 1, between $-89°$ and $89°$, with $\pm 89.9°$ appended. The mean Δ over the 181 measured degrees is then taken, for each SNR value between -50 and 100. Test code is found in Appendix 20.

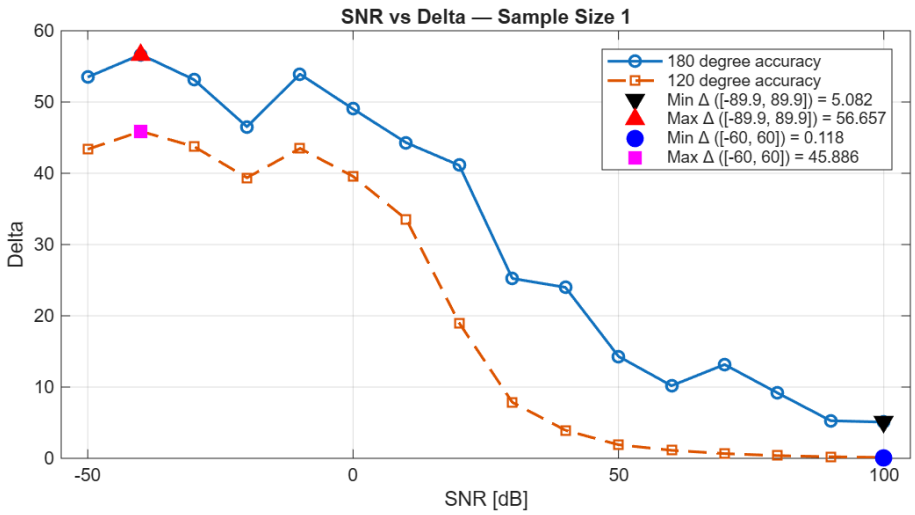


Figure 4.54: Line chart of data obtained for $N = 1$, when varying the SNR. SNR value is on the x-axis, and absolute Δ between θ_{Ref} and θ_{Meas} on the y-axis. This is plotted against the SNR used for obtaining the data points. Largest and smallest Δ are highlighted.

The plot on Figure 4.54 shows that when SNR increases at N=1, the beamformer becomes progressively more accurate, for both the extended angle space, and the one specified in the use case. Even so, at and SNR of 100, not even the use case angle space $-60°$ to $60°$, attain the accuracy specification. Using 5 dB SNR as a standpoint, different values for N are tested, to

see if accuracy can be improved. As more samples taken, should give better accuracy when cross-correlating. Test code is found in Appendix 20.

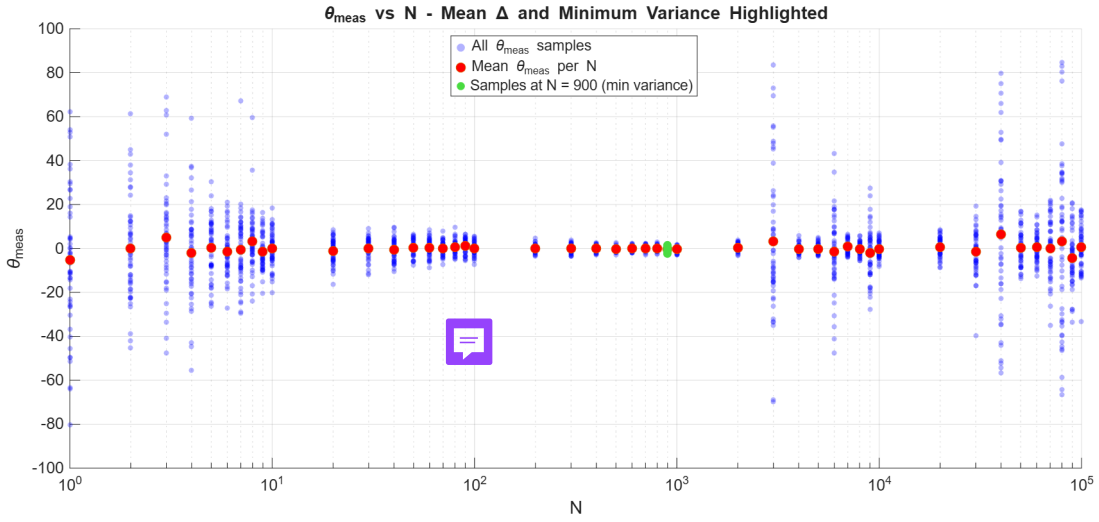


Figure 4.55: Scatter plot of sample sizes N plotted against θ_{Meas}, $\theta_{Ref} = 0°$. N is on the x-axis, and θ_{Meas} is on the y-axis. The value of the sample size with lowest variance is highlighted in green.

A simulation with 50 iterations for each sample size is plotted on Figure 4.55. As expected, the low sample sizes are much more sensitive to noise, as there is less data to correlate. What is meant by this, is that assuming a gaussian distribution, more samples mean the signal would resemble an ideal signal more closely. It also shows that the sample size with least variance occurs when 900 samples are taken. What is not expected is that above $N = 2000$, the beamformer becomes very inaccurate, with some sample sizes being significantly worse. The cause of this was not identified. The test code is seen in Appendix 21

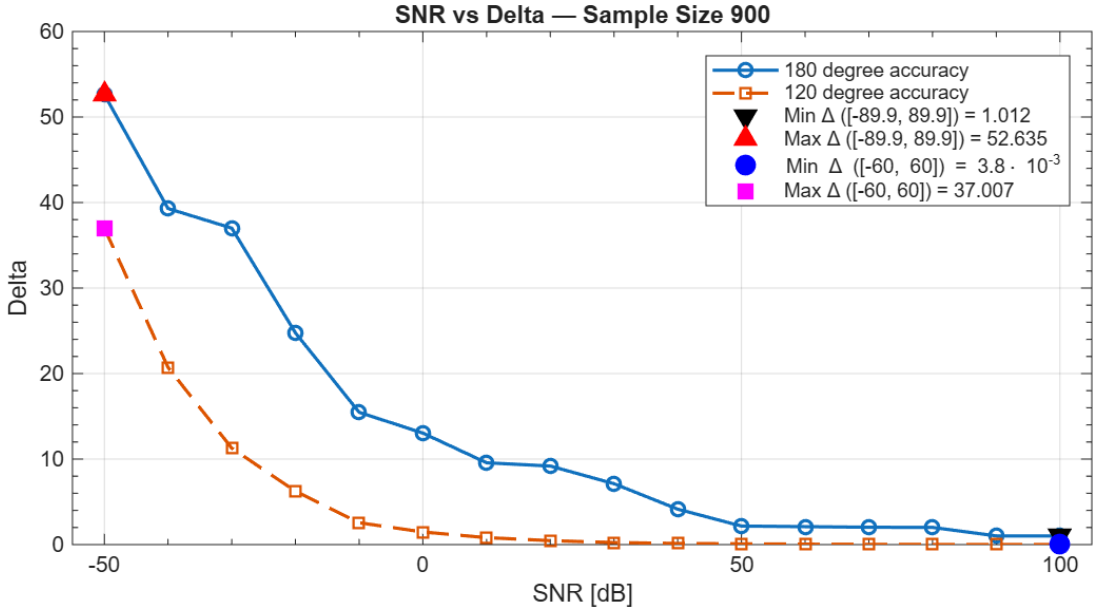


Figure 4.56: A plot of the difference between θ_{Ref} and θ_{Meas}, plotted against the SNR values used for obtaining the data point. Here N = 900 with the minimum and maximum delta being highlighted.

As seen in Figure 4.56, the beamformer becomes much more resistant to noisy signals. It can be seen that for 120° accuracy, it follows a descending logarithmic curve. and in contrast to the curve on Figure 4.54 the beamformer is generally functional above 0 dB SNR. For 180° accuracy, to get somewhat accurate performance the SNR has to be 50 dB or above.
What can be concluded from the simulated tests, is that for an ideal sine wave, the beamformer functions as intended, and attains the $2.28° \cdot 10^{-3}$ accuracy specification with $1.14° \cdot 10^{-3}$. Once noise is introduced, the accuracy of the beamformer worsens, requiring an SNR of 100 dB to get down to $3.8° \cdot 10^{-3}$. Running the beamformer with $N = 900$, and 78948 angle steps on the Raspberry Pi 5 chosen to handle the processing, the average runtime is 1.754 seconds, which attains the 8.69 second detection time specification. What can be gained from the SNR tests is that either the SNR should at least be larger than 100 dB when $N = 900$, and/or methods for increasing accuracy should be investigated further. Based on the tests in this section, the amount of angle steps have been changed to 78948, and the samples to take, is set to 900. The beamforming function that is implemented can be seen in Figure 4.57.

```python
def beamforming_das(rx, distance, no_ele):
    theta_sweep = np.linspace(-1*np.pi/2, np.pi/2, 78948)
    results = []
    for thetas in theta_sweep:
        w = np.exp(2j * np.pi * distance * np.arange(no_ele) * np.sin(thetas))
        y = w.conj().T @ rx
        results.append(np.abs(np.sum(y)))
    return np.rad2deg(theta_sweep[np.argmax(results)])
```

Figure 4.57: Python code of the revised delay and sum beamformer function

4.3.3 Beamforming Validation Test

Now that the beamformer has been tested with simulated signals, it needs to be tested with real signals, to observe how it behaves in reality. The first test is to examine, whether the beamformer works with the SDR. To do this, two signal generators have been connected to the RX ports, to make sure antennas do not play a role in the baseline functionality of this module. The full test journal and results are seen in Appendix 9. The results of this test show that, with the exception of the data points in 90° ambiguity, the mean θ_{Meas} is $\pm 1.3226°$ off from the calculated θ_{Ref}. The deviation from each mean is at most $\pm 0.5405°$. With the 90° measurement removed, this test shows that the beamformer functions as intended, but lacks the accuracy needed to attain the accuracy specifications from Section 3.3. Which was expected, based on the results of the simulation tests.

A second test was made using the entire system, which includes antenna 4 made in Section 4.1, the SDR, and the Raspberry Pi 5 that processes the data and beamforming, all mounted on the TARP. This test was conducted in a RF shielded anechoic chamber to minimize interference. The purpose of this test was to verify the beamforming algorithm functionality with non-ideal antennas, and a real transmit signal.

θ_{Ref}	mean θ_{Meas}	θ_{σ}
45°	−25.72°	$409.50° \cdot 10^{-3}$
30°	70.13°	$671.58° \cdot 10^{-3}$
15°	24.89°	$294.84° \cdot 10^{-3}$
0°	−1.01°	$196.56° \cdot 10^{-3}$
−15°	−31.26°	$180.18° \cdot 10^{-3}$
−30°	55.13°	$638.82° \cdot 10^{-3}$
−45°	13.64°	$573.30° \cdot 10^{-3}$

Table 4.12: Test results from DAS algorithm test with antennas. θ_{Ref} is the transmitted signals angle in relation to boresight of the antenna array, θ_{Meas} is the mean over 5 measurements, and θ_{σ} is the distance from mean to max/min observed value.

The test results seen in Table 4.12, show that the beamformer does not function as expected, when the transmit signal comes from any other angle than 0°. Here the mean θ_{meas} was −1.01°, with a deviation of 0.1965°. The rest of the measurements, do not match the transmit signal, but all deviate less than 1° over 5 measurements. The most likely cause of the discrepancy between θ_{Ref} and θ_{Meas}, is because the antennas were incorrectly placed one wavelength apart, instead of half a wavelength, which the beamformer expects. Trying this in a simulation by, setting element distance to 1, and passing 0.5 to the beamformer function produces somewhat similar errors, seen in Table 4.13.

θ_{Ref}	θ_{Meas}
45°	−35.8689°
30°	−53.3757°
15°	31.1739°
0°	0.0020°
−15°	−31.1675°
−30°	−53.6858°
−45°	35.8557°

Table 4.13: Results from simulated test with mismatched antenna distance. Measured mean over 5 samples.

There are still significant differences between the measured data and simulated data, especially around ±30°, this looks like it could be caused by ambiguity in the resulting antenna pattern, with the real antennas not being isotropic.

Since the beamformer did not attain the specifications in the simulated tests, it was expected that the physical tests would not either. What can be concluded, is that the general functionality of the beamformer works as intended, as shown in the test without antennas, but not to the $2.2° \cdot 10^{-3}$ accuracy specification set for this module. The timing specification is attained at 1.754 seconds to run, for $N = 900$.

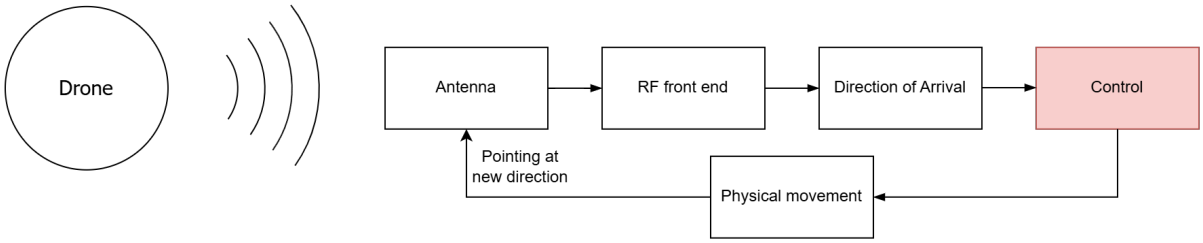
The amount of angle steps are sufficient, so the next steps towards increasing accuracy, could be increasing the amount of elements in the array, to narrow the main lobe, as well as look into other direction of arrival algorithms, or ways to build out the DAS beamformer. Additionally, more could be done to reduce compute time. Implementation on an FPGA or in C and/or parallelization of computation would be beneficial. Additionally, the test with the patch antenna sub-arrays should be remade, with the sub-arrays placed half a wavelength apart, and

done for the full angle space of $-90°$ to $90°$.

Parameter	Specification	Fulfilled
DSP detection time	8.69 s	✓
DSP accuracy horizontal	$2.28° \cdot 10^{-3}$	X
DSP accuracy vertical	$1.14° \cdot 10^{-3}$	X

Table 4.14: Summary of validation of specification of front end.

4.4 Control of TARP



This section explains how the mechanical control system of the TARP is developed. This involves constructing a mathematical model of the system, designing a controller using feedback and implementing this on the physical device. While a controller could be designed only based on the step response, a choice was made to construct an actual model.

However to construct such a model, one must have an understanding of the physical system. In Figure 4.58, a callout diagram of the entire system can be seen. The encoder on azimuth and tilt is a SICK DBS36E-S3EK00500 incremental encoder, having a resolution of 1000 pulses pr. rotation [86]. The encoder on tilt is directly mounted, while the one on azimuth has a 1:5 gearing, to make space for a slip ring.

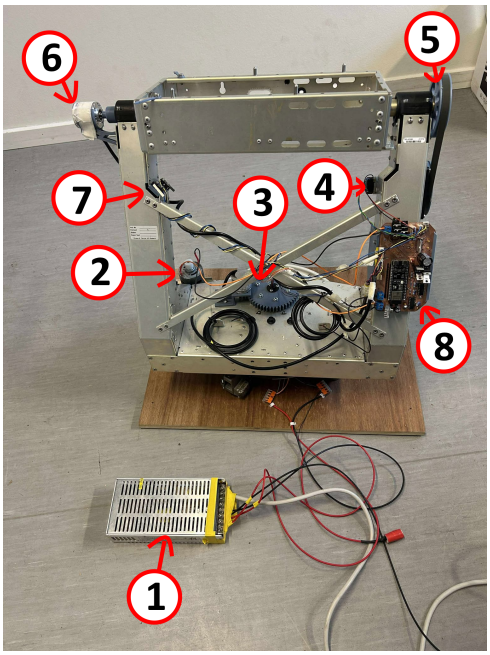


Figure 4.58: Callout diagram of the TARP. **(1)** Power supply **(2)** Azimuth motor **(3)** Encoder for azimuth **(4)** Tilt motor **(5)** Tilt drive train **(6)** Encoder for tilt **(7)** End stops for tilt **(8)** Control PCB

Closeup pictures of Figure 4.58 can be seen in Appendix 27. The metal parts of Figure 4.58 where given, but all 3D-printed parts have been custom made.

4.4.1 TARP General Model

As both azimuth and tilt motors principally function the same, only with different parameters, a general model is made, wherein parameters for each motor are found and inserted. The general model is used to derive the system's transfer function. As the objective of the controller is to regulate position, the output is position in degrees and the input is the voltage over the motor. While the motor is going to be controlled by a PWM signal, it is modeled using analog voltage. This is done both to simplify calculations, and because the inductor resists instantaneous current change, producing a smoothing effect on the PWM signal.

Initially it is easier to look at the system without a gearing, and account for it afterwards. To make a complete model, firstly an electric model is made, secondly a physical model is made and lastly the two models are combined. The electrical model of a motor is known, and can be seen on Figure 4.59 [87].

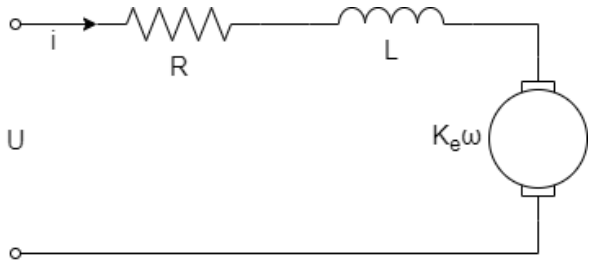


Figure 4.59: Illustration of the electric model of a motor. R is the resistance of the wires inside the motor, L is the inductive effect of the coils in the motor. $K_e\omega$ is the back EMF generated by the motor during rotation. U is voltage and i is current over/into the motor [87].

Equation 4.25 relates voltage across the motor, with the current going through it and the angular velocity of it.

$$u(t) = i(t) \cdot R + \frac{di(t)}{dt}L + K_e \cdot \omega(t) = i(t) \cdot R + \frac{di(t)}{dt}L + K_e \cdot \frac{d\theta(t)}{dt} \tag{4.25}$$

The equation is Laplace-transformed and there is isolated for current $I(s)$, in the s-domain, assuming initial conditions are 0:

$$U(s) = I(s) \cdot (R + L \cdot s) + K_e \cdot \theta(s) \cdot s$$

$$I(s) = \frac{U(s) - K_e \cdot \theta(s) \cdot s}{(R + L \cdot s)} \tag{4.26}$$

While Equation 4.26 describes the system electrically in the s-domain, the system is electromechanical, and so must be coupled with a physical model. Therefore a physical model is to be made, where two formulas are essential:

Newtons second law of rotation, relating torque τ, inertia J and angular acceleration α [88].

$$\sum \tau(t) = J \cdot \alpha(t) \tag{4.27}$$

The **torque constant equation** uses Maxwells equations to relate the torque τ produced by the motor, to the current i, using the torque constant K_t. This relation is proportional, and given by Equation 4.28 [87].

$$K_t \cdot i(t) = \tau_{motor}(t) \tag{4.28}$$

In Equation 4.27 the sum of torques, assuming no external forces apply, is $\tau_{motor}(t) - \tau_{fric}(t)$, where τ_{fric} is the combined frictional torque.

$$\tau_{motor}(t) - \tau_{fric}(t) = J \cdot \alpha(t)$$

τ_{fric} consists of two types, static friction τ_{fs} and kinetic friction τ_{fk} [88]. Static friction is left out to simplify the model, as the main goal is to model the dynamic behavior of the TARP. This leaves only kinetic friction, which again consists of two main components: a viscous friction, also called fluid friction, τ_v, and a Coulomb friction, τ_c, also called dry friction [88]. Where τ_v is proportional to the velocity, modeled as $\tau_v = B_v \cdot \omega(t)$, with B_v being the viscous friction constant for the system and τ_c is a constant drag [88] and $\omega(t)$ the angular velocity. Even though $\tau_{fk} = B_v \cdot \omega(t) + \tau_c$ is mathematically linear, it is physically non-linear as the

superposition principle does not hold [89]. Proof of this can be seen in Appendix 1. To simplify the model, non-linear Coulomb friction is disregarded; however, it is included during the parameter estimations to improve accuracy. Combining this with Equation 4.27 and 4.28 then produces:

$$K_t \cdot i(t) - B_v \cdot \omega(t) = J \cdot \alpha(t) \tag{4.29}$$

Laplace transforming and making it a function of angular position instead of angular velocity, yields the following Equation.

$$K_t \cdot I(s) - B_v \cdot \theta(s) \cdot s = J \cdot \theta(s) \cdot s^2 \tag{4.30}$$

Combining equation 4.26 and 4.30, a general transfer function from voltage to angular position can then be derived.

$$K_t \cdot (\frac{U(s) - K_e \cdot \theta(s) \cdot s}{(R + L \cdot s)}) - B_v \cdot \theta(s) \cdot s = J \cdot \theta(s) \cdot s^2$$

$$\frac{K_t}{s((J \cdot s + B_v) \cdot (R + L \cdot s) + K_t \cdot K_e)} = \frac{\theta(s)}{U(s)} \tag{4.31}$$

4.4.2 Motor Parameters

To fit Equation 4.31 to a specific motor, the parameters for both motors must be determined.

Determining resistance (R)

To determine R, the motor should be kept still, a known voltage should be applied, then measure the current when the inductor is charged up. This will simplify Equation 4.25 to:

$$u(t) = i(t) \cdot R \Rightarrow R = \frac{u(t)}{i(t)} \tag{4.32}$$

Determining inductance (L)

Isolating L con be done by keeping the motor still, as this results in Equation 4.26 being simplified to:

$$I(s) = \frac{1}{R + s \cdot L} \cdot U(s) = \frac{1/R}{1 + s \cdot L/R} \cdot U(s) \tag{4.33}$$

Which has the time constant L/R, that can be measured as the time from a voltage step until current hits 63.2% of max value [90].

Determining the back EMF constant (K_e)

The back EMF constant can be determined by keeping the motor at a constant angular velocity (I.e. applying constant voltage), as this causes the inductor to have no effect. Simplifying Equation 4.25 and solving for K_e results in:

$$K_e = \frac{u(t) - R \cdot i(t)}{\omega_{motor}}$$

The angular velocity of the TARP is directly affected by the gearing, therefore the gear ratio N is taken into account. N is determined by the following Equation (4.34) [91] [92]:

$$N = \frac{\omega_{load}}{\omega_{motor}} = \frac{teeth_{load\_gear}}{teeth_{motor\_gear}} \tag{4.34}$$

Where ω is the angular rotation and $teeth$ is the amount of gear teeth. This scales the motor parameter K_e with N as:

$$K_{e\_motor} = \frac{u(t) - R \cdot i(t)}{\omega_{TARP}/N} \Rightarrow K_e = K_{e\_motor} \cdot N = \frac{u(t) - R \cdot i(t)}{\omega_{TARP}} \tag{4.35}$$

Determining torque constant (K_t)

To determine K_t, Equation 4.28 is used with a torque meter and a measured current. The result needs to be upscaled with N as the gearing increases the torque affecting the load, this can be seen in equation 4.36 [92] [91].

$$K_t = K_{t\_motor} \cdot N = \frac{\tau}{i(t)} \cdot N \tag{4.36}$$

Determining viscous friction (B_v)

To determine a collected viscous friction, the motor is set to a constant angular velocity. This ensures that the torque produced by the motor is equal to the braking torque. This means that Equation 4.28 can be combined with the formula for kinetic friction ($B_v \cdot \omega(t) + \tau_c$). This time accounting for the Coulomb friction, to increase the accuracy of the estimate [88]:

$$K_t \cdot i(t) = B_v \cdot \omega(t) + \tau_c \tag{4.37}$$

With multiple motor currents and angular velocities, a linear relationship can be estimated using regression. This line has B_v as the slope and the Coulomb friction as the intercept [88]. As this combines motor and load friction, the gear ratio is irrelevant.

Determining intertia (J)

The inertia J can be determined using Equation 4.29 isolated for J:

$$\frac{K_t \cdot i(t) - B_v \cdot \omega(t)}{\alpha(t)} = J$$

However, to get more accurate approximation of J the Coulomb friction should also be accounted for:

$$\frac{K_t \cdot i(t) - B_v \cdot \omega(t) - \tau_c \cdot sgn(\omega(t))}{\alpha(t)} = J \tag{4.38}$$

Where $sgn()$ is the sign operator returning either -1 or 1 depending on the sign of the input function [93]. This is required, as the direction of friction is always opposite that of the movement. Equation 4.38 only applies while α is defined, i.e. during acceleration.

Determining Motor Transfer Functions

The measurement and calculations of the motor parameters for the azimuth motor can be seen in Appendix 5. From this, the following following values can be extracted, inserted into Equation 4.31 and simplified to Equation 4.39.

$$N_{azimuth} = 18.6 \qquad R = 724.31\,\text{m}\Omega \qquad L = 1.88\,\text{mH} \qquad K_v = 810.90\,\frac{\text{mV}}{\text{rad/s}}$$

$$K_t = 17.63\,\frac{\text{mNm}}{\text{A}} \qquad B_v = 3.97\,\frac{\text{mNm}\cdot\text{s}}{\text{rad}} \qquad J = 7.73\,\frac{\text{mNm}\cdot\text{rad}}{\text{s}^2}$$

$$\frac{\theta(s)}{U(s)} = \frac{17.63\cdot 10^3}{s((7.73\cdot s + 3.97)\cdot(724.31 + 1.88\cdot s) + 17.63\cdot 810.90)} \tag{4.39}$$

Note, units have been omitted, as it gives a better overview. Likewise parameter determination for the tilt motor happens in Appendix 6, with the result being:

$$N_{tilt} = 24 \qquad R = 6.73\,\Omega \qquad L = 3.87\,\text{mH} \qquad K_e = 285.57\,\frac{\text{mV}}{\text{rad/s}}$$

$$K_t = 404.91\,\frac{\text{mNm}}{\text{A}} \qquad B_v = 5.16\,\frac{\text{nNm}\cdot\text{s}}{\text{rad}} \qquad J = 15.44\,\frac{\text{mNm}\cdot\text{rad}}{\text{s}^2}$$

$$\frac{\theta(s)}{U(s)} = \frac{404.91\cdot 10^3}{s((15.44\cdot s + 5.16)\cdot(6.73\cdot 10^3 + 3.87\cdot s) + 404.91\cdot 285.57)} \tag{4.40}$$

4.4.3 Construction of a Controller

With both transfer functions calculated, controllers can now be implemented. A controller topology can be selected by inspecting the open loop (OL) bode plots of the transfer function. These can be seen on Figure 4.60.

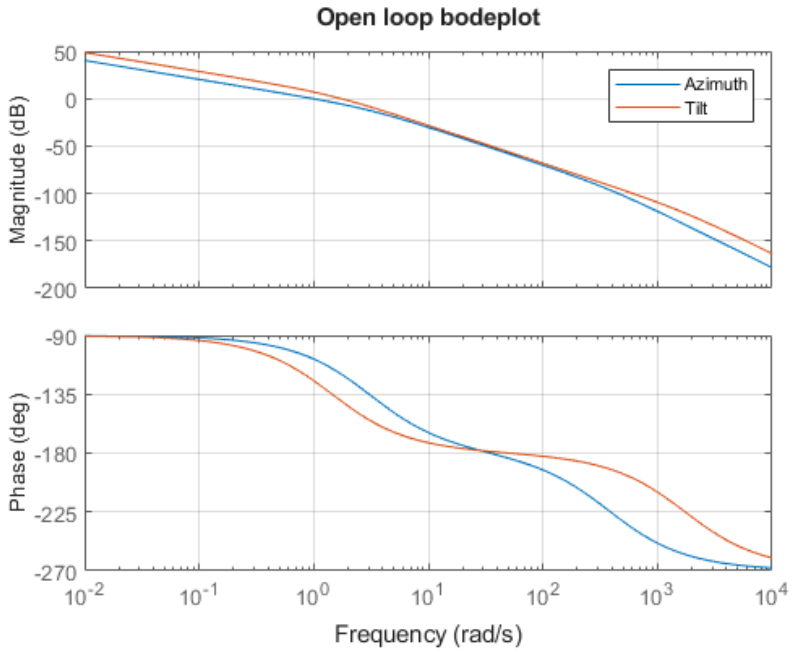


Figure 4.60: Open loop bode plot of azimuth and tilt without controller

When choosing a specific controller topology, closed-loop (CL) stability should be closely managed. If the phase is lower than -180 degrees, at the OL gain crossover frequency (where gain $= 0\,\text{dB}$) the CL is unstable [89]. However, the speed of the system is determined by the crossover frequency, with a higher frequency resulting in a faster regulation. A note about CL is that, in this report, the feedback path is modeled as 1, as sensors are expected to be close to ideal. This also means that any non-ideal properties of the sensors are modeled as disturbances [89]. After stability has been determined, the performance is evaluated against the specifications set forward in Section 3.2. Initial consideration of the standard controller options, proportional (P), proportional-integral (PI), proportional-derivative (PD) and Proportional-integral-derivative (PID), has been made [89]. The PI can quickly be disregarded as a PI-controller has an initial phase change of -90 degrees, which would cause instability, since it would start in -180 degrees [89]. A P-controller is prioritized first for implementation, as it is the most simple of the remaining controllers.

Implementing a P-controller

A P-controller changes the gain crossover frequency, without changing the phase [89]. This means that it gives direct control of the phase margin (PM), i.e. the distance until -180 degrees at the gain crossover frequency [89]. However, a smaller phase margin brings the system closer to instability and results in increased overshoot. The azimuth system is not affected by overshoot, while the tilt system is. As can be seen on Figure 8.24, a slip ring has been installed, which allows for infinite rotations, and thereby the amount of overshoot is indifferent for the azimuth. Since the tilt motor contains cables from the antennas and no slip ring, excessive overshoot could cause damage to the hardware. Although a specific tolerance limit is currently undefined, the magnitude of the overshoot will be evaluated in a subsequent section. The PM of 45° ensures that the system is stable, while a PM of 0° creates a marginal stable system [89]. Therefore, a PM of 20° has been chosen as an acceptable level, as this is expected to increase the speed of the system and leave enough room for disturbances, ensuring the system is stable. To achieve this, it can be seen on Figure 4.61, that it requires having a gain of 27.9 dB for azimuth and 12.7 dB for tilt. Converting this to gain using Equation 4.41 results in 24.83 gain for azimuth and 4.32 gain for tilt respectively.

$$\text{gain} = 10^{\text{dB}/20} \tag{4.41}$$

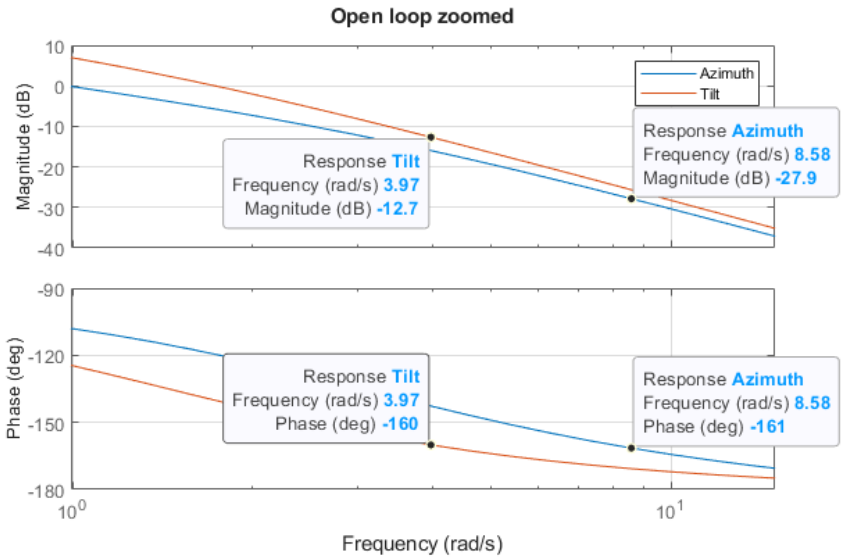


Figure 4.61: Zoomed version of Figure 4.60, with *datatips* for better data extraction.

Applying a step of 120° on the CL of azimuth and 45° on the CL of tilt with their P-controllers results in the graph visible on Figure 4.62. 120° has been used in accordance with Section 3.3.1, where the azimuth axis is cut into three equally big chunks, and tilt locked at 45°. Resulting in a full scan requiring azimuth to move 3x120° and tilt to move 45°.

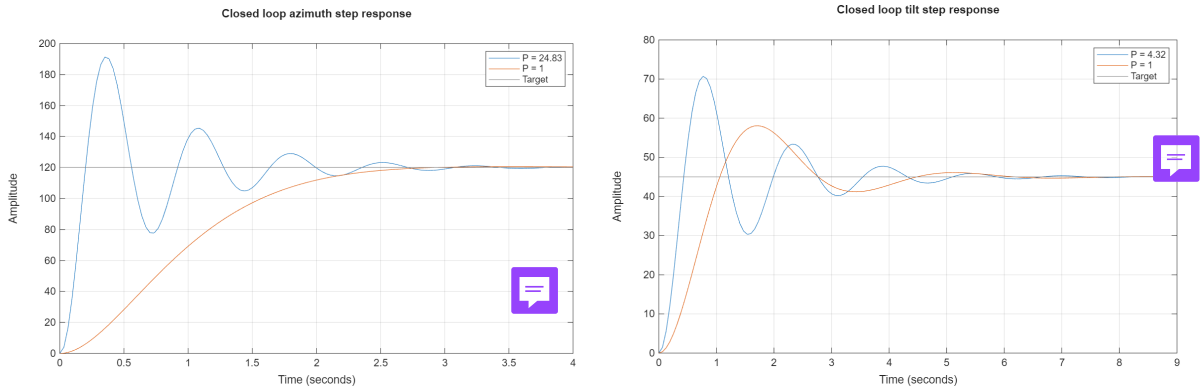


Figure 4.62: CL step response of azimuth and tilt using the calculated P-controller gain, and using a gain of 1.

In Section 3.3.1, a horizontal accuracy of $9.17° \cdot 10^{-3}$ and a vertical accuracy of $4.58° \cdot 10^{-3}$ within a time of 34.78 s is required. On Figure 4.62 it can be seen that it takes about 3.65 s and 8.07 s to reach the required accuracy of horizontal (azimuth) and vertical (tilt) respectively. In order to scan the entire airspace, the prototype needs scan three points in the azimuth while the tilt motor is locked at 45°. When the prototype is moving towards the first point, both azimuth and tilt have to move to the target degrees. Due to the slower positioning time of the tilt system, its duration is explicitly added to the total scan time calculation and not the azimuth:

$$t_{scan} = 2 \cdot 3.65\,\text{s} + 1 \cdot 8.07\,\text{s} = 15.37\,\text{s}$$

This is less than half the required time of 34.78 s stated in Section 3.3.1. This means the current P-controller is sufficient in the theoretical model to scan the airspace within the mechanical

requirements and the accuracy is sufficient as well. This should leave enough time for the prototype to scan and detect the drones within the airspace and track it.

4.4.4 Physical Implementation

In order to implement a physical version of the P-controller, a hardware platform needs to be created. First, a micro-controller is needed to control the azimuth and tilt motors. Since micro-controllers can typically output 5 V and 40 mA, and the motors require 12 V, an H-bridge is needed [94] [95]. The micro-controller will control the H-bridge with a PWM signal. To specify where the TARP is compared to the target angle, encoders are used as feedback, see Figure 8.24. Lastly a couple of buttons for the tilt motor are also used, to indicate start position and prevent endless rotation, resulting in the wires getting tangled, see Figure 8.25.

The available encoders have 1000 interrupts per rotation. This means the current equipment can maximum give a resolution of:

$$\theta_{azi} = \frac{360^\circ}{5 \cdot 10^3} = 72^\circ \cdot 10^{-3}$$

$$\theta_{tilt} = \frac{360^\circ}{10^3} = 360^\circ \cdot 10^{-3}$$

Note: the accuracy of azimuth is increased due to the gearing, which gives it 5000 interrupts per rotation. The resolution is far from the required accuracy of $9.17^\circ \cdot 10^{-3}$ for azimuth and $4.58^\circ \cdot 10^{-3}$ for tilt. Although this is assessed to be good enough for a proof of concept prototype.
An ESP32 microcontroller is used, as it is dual-core, which allows one core to control the motors and one to communicate with the PI, and deemed fast enough.

PCB

Functioning as the central processing unit, the ESP32 handles both the actuation, sensor data acquisition and communicating to the Pi. Therefore, it requires interfaces with the following:

- Azimuth motor through an H-bridge
- Azimuth encoder
- Tilt motor through an H-bridge
- Tilt encoder
- Two limit buttons for the tilt rotation
- PI communication

It should be noted that the communication to the Pi will be through a micro-USB port already installed on the ESP32, therefore no additional hardware is required for the PCB. The H-bridge is chosen based on the specifications of the motor it needs to control. Based on the datasheet of the tilt motor, it needs 12 V and 2 A with no load to run at max speed [95]. An L298 H-bridge is chosen to control the tilt motor, as it is able to handle up to 46 V and 4 A

and available at AAU [96]. The part number for the azimuth motor is not known, therefore a preliminary test was conducted to find the voltage needed to determine the motor's operating range. The voltage was incremented until it reached 12 V, where the motor had high rotational velocity. The voltage was limited to this threshold to prevent potential thermal damage. In the motor parameter test (see Appendix 5), an initial current of 10 A is needed to overcome the static friction and initiate motion, see Figure 8.8. As an H-bridge able to handle 10 A is not available, one is built.

H-Bridge

The H-bridge needs to be able to handle 12 V and 10 A and switch the direction of the motor. Figure 4.63 shows the current flow through the motor, depending on the direction pin being high or low, while the enable pin is high.

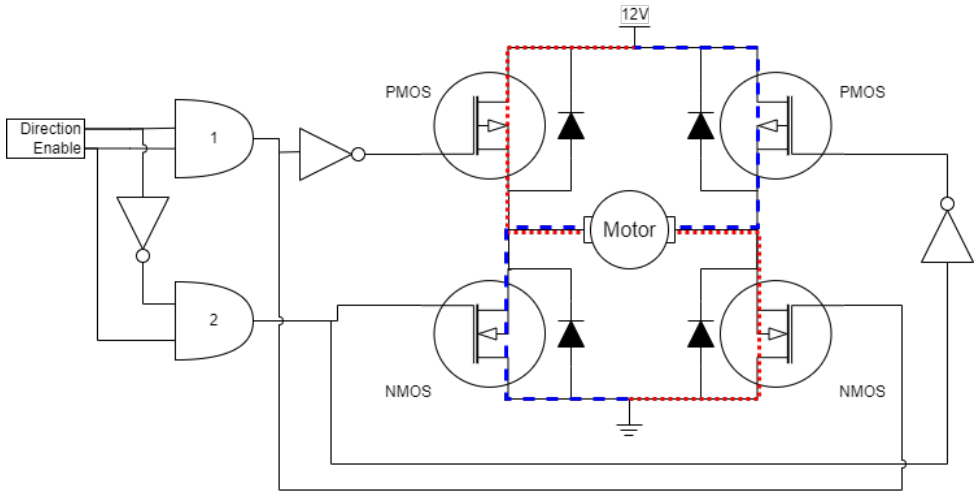


Figure 4.63: H-Bridge current flow. Red dotted line shows current flow when Direction and Enable are high. Blue dashed line shows the current flow when Direction is low and Enable is high.

The H-brigde consists of a pair of PMOS and NMOS. The NMOS turns on when there is a positive voltage V_{GS} and PMOS when there is a negative V_{GS} [97]. For that reason the gate voltage should be inverted compared to NMOS, hence the NOT-gate before the PMOS. The NOT-gate (placed at the input of the AND-gates) combined with the AND-gates ensures that only one pair of PMOS and NMOS can be turned on at the same time, when the enable pin is high. This prevents short-circuit of the right and left side of the H-bridge. The diodes are placed to protect the transistors from what is known as an inductive kick [98]. An inductive kick happens when the transistor switches off, then the magnetic field of the inductor collapses. The inductor resists the change in current resulting in a large reverse polarity. This can be seen in the voltage equation (4.42)for an inductor:

$$V = -L\frac{di}{dt} \quad [\text{V}] \quad (4.42)$$

It can be seen, that when the current rapidly changes, the differentiation becomes a large value, which results in a momentarily large reverse voltage spike. The negative sign in the Equation (4.42) is Lenz's law which states that when there is a change in magnetic flux, there is an opposite change in the magnetic field [99].

In the H-bridge, when the PMOS or NMOS turns off, it creates the inductive kick. When this happens, the diodes open, allowing the current to flow through the diode, preventing the high voltage spike and protecting the transistors [98]. The physical PCB can be seen in Appendix 15 and the schematics in Appendix 14.

In the physical H-bridge, the NOT-gates driving the PMOS are implemented using transistors, since the available NOT-gate circuits maximum voltage rating is 5 V. The PMOS conducts when V_{GS} drops below -1 V. In order to keep the PMOS closed, the voltage needs to be 12 V at the gate, since the source is connected to the 12 V. This will result in $V_{GS} = 0$ V, hence it is closed. When the NOT-gate transistor is turned on, the voltage at the gate drops to 0 V opening the PMOS fully. Figure 4.64 shows the implementation of an NPN BJT to the H-bridge.

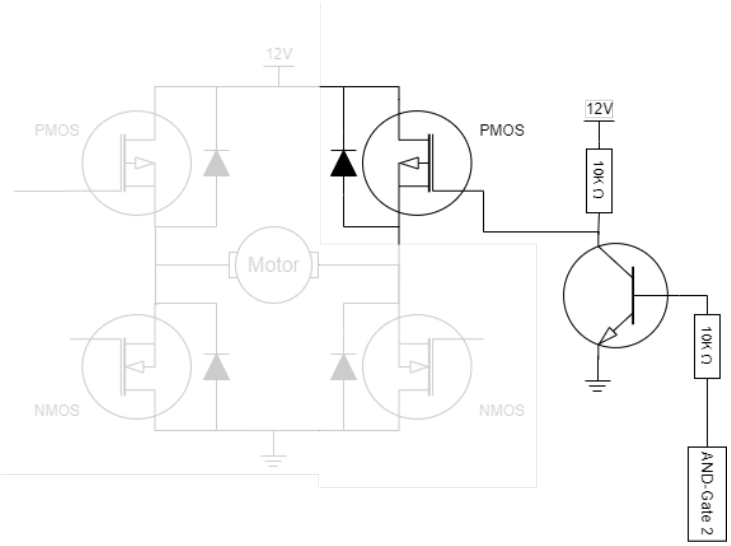


Figure 4.64: H-bridge short circuit issue showing the overlap when the NMOS is open before the PMOS is closed.

The chosen transistor is NPN BC546 since it can handle a 65 V [100]. The current through the resistor at the collector of the BJT is chosen to be 1 mA, to keep power consumption low. The voltage across the resistor when the BJT is fully saturated is 12 V, from the voltage supply. Therefore, $R = \frac{12\,\text{V}}{1\,\text{mA}} = 12\,\text{k}\Omega$, which is rounded down to 10 kΩ, for simplicity and is used for the base resistor as well. Reducing the resistor's resistance increases the quiescent current at the collector 1.2 mA, which isn't an issue. A calculation is needed to ensure the base is not pulling too much current from the logic circuit when the BJT is saturated. There is an approximate voltage drop over base to emitter (V_{BE}) of 0.7 V and the AND-gate supplies 5 V [101]. This means, the resistor in series with the base will have 4.3 V voltage drop and the resistor is chosen at 10 kΩ:

$$I = \frac{4.3\,\text{V}}{10\,\text{k}\Omega} = 430\,\mu\text{A}$$

The maximum continuous current from the AND-gate is 25 mA [101], which is far above what is necessary for the BJT.

The logic circuit SN74HC08 is chosen based on availability at AAU. The PMOS in the circuit are IRF9530 and NMOS are SUP85 as they are able to handle the current of 10 A and a voltage

of 12 V [102] [103].

H-Bridge Short Circuit
When testing the H-bridge, it initially works when rotating at full power, but it overheats extremely quickly when constantly switching. Through an analysis it was found that the left side of the H-bridge was open at the same time for a short period of time (2.5 µs), see Figure 4.65. This means that H-bridge is shorted every time it switches direction. This is also true for the right side of the H-bridge.

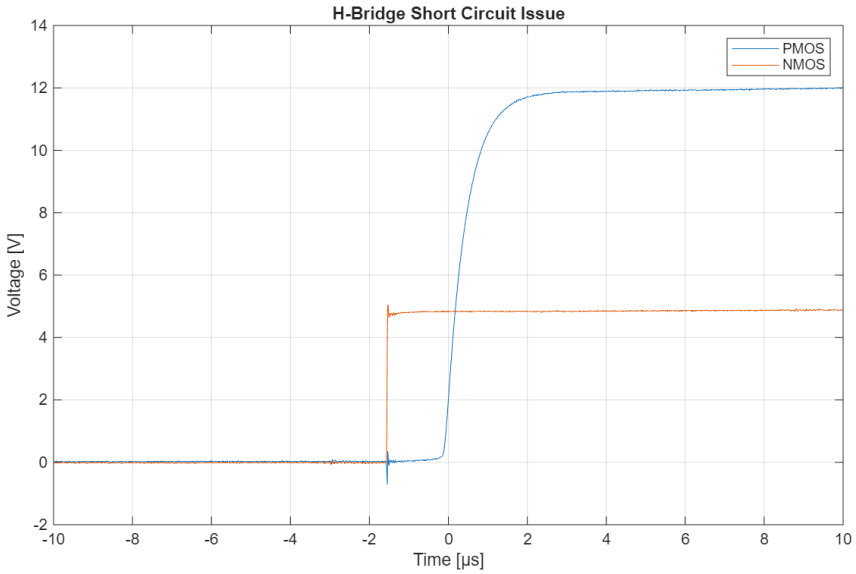


Figure 4.65: H-bridge shortcircuit overlap showing PMOS and NMOS open at the same time for 2.5 µs.

Figure 4.65 shows that NMOS opens and the PMOS starts to close −1.8 µs later. The PMOS turns off at 10 V, which means the short circuit lasts for 2.5 µs. This becomes a critical issue once the control code is running, since it will increase switching frequency, the closer it gets to the desired angle.

Through another analysis, a potential cause of the problem appears. The chosen BJT used as the NOT-gate, has a large capacitive effect, which is slow to discharge, see Figure 4.66.

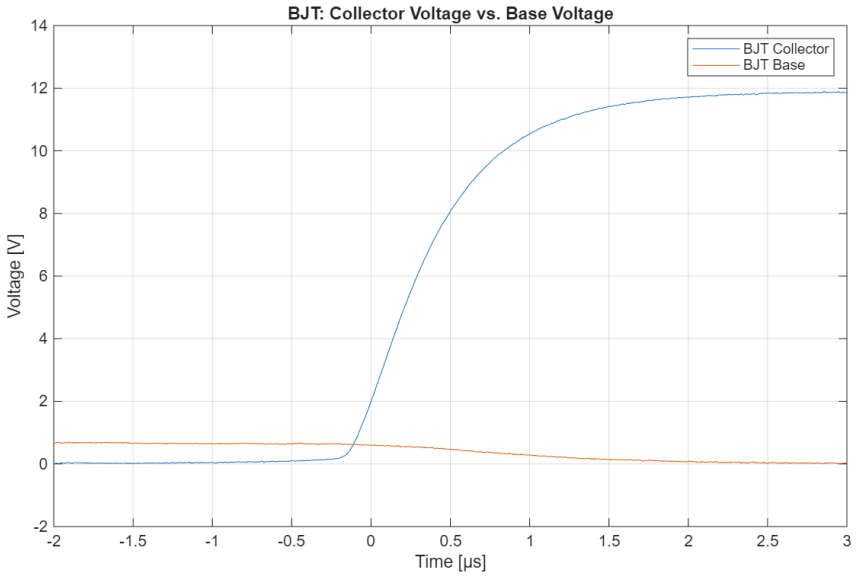


Figure 4.66: H-bridge short circuit issue showing the voltage at the base and the collector of the BJT acting as the NOT-gate.

This can be improved by either changing the BJT to another transistor, reducing the resistor size or by creating a small delay when switching. If the resistor size is decreased, the discharge time will likewise decrease, however current consumption increases. For this project, it has been chosen to create a delay in the code of 5 µs, which should be more than sufficient in preventing a short circuit.

Code

Now that there is a physical platform to implement the code for the p-controller, the code can be written. Figure 4.67 shows a snippet of the code controlling the azimuth motor and tilt motor, depending on the input angle. In Appendix 23 the entire code of TARP control can be seen.

```
24  //-----------------TILT CONTROL--------------------
25  gearing = 1;  // which gearing is running on the sensor
26  tiltInDegrees = convertPulsesToAngle(pos_tilt, gearing);
27  float error = angleTilt - tiltInDegrees;
28  float deltaVolt = error * controllerGainTilt;
29  Serial.println(deltaVolt);
30  setVelocity(deltaVolt, ena_pin_tilt, tiltOffset);
```

Figure 4.67: Snippit of TARP control code showing the tilt control code.

Initially TARP receives an angle from the Raspberry PI, which TARP needs to move towards. Additionally, if no new angle is received, the previous angle is used. Then the position is calculated based on the amount of interrupts received from the encoders. This is then used to find the error, how far TARP is from the desired angle. The error is multiplied with the controller gain, which has been calculated in Section 4.4.3. This gives the required output voltage for the tilt motor, which is then set moving the TARP.

4.4.5 TARP Validation Test

The TARP has been tested to see the step response of the physical system, see Appendix 7. The following section shows the test of the physical version of the azimuth and tilt motors versus the corresponding simulated model.

Azimuth test

In this test, the TARP has received a target angle at 120°, moving the azimuth motor. TARP saves the position as it moves towards the target angle. Based on the simulated model, the physical model should reach the target angle of 120° within 3.65 s, oscillating within $\pm 0.072°$. Figure 4.68 shows the saved positions, giving the step response of the azimuth motor targeting 120° versus the simulated model.

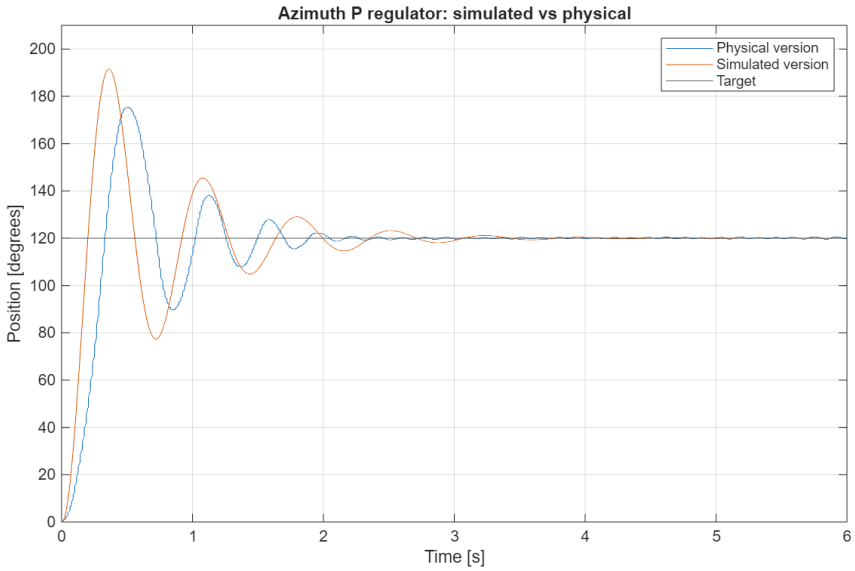


Figure 4.68: Step response of the azimuth p-controller of the physical version vs. the simulated version moving towards a target angle of 120°.

The response of the azimuth in the physical model is initially slower compared to the simulated, looking at the slope of the initial rise. This also means the physical version has less overshoot than the simulated. This is potentially because of the friction in the physical model is higher than that in the simulated model. The physical version oscillates around the target from $-0.216°$ to 0.288° after approximately 3.2 s, while the simulated reaches the target within the requirement after 3.65 s. The physical version is presumably faster than the simulated model, but the physical model does not reach the expected resolution of 0.072°, oscillating 3 to 4 times that resolution. This means that concluding whether the simulated model is faster than physical model, is not possible, since the physical model never reaches the expected angle resolution. Therefore, another controller needs to be implemented, which should help reduce the oscillations. This will ensure the prototype stabilizes within the expected resolution of 0.072°, showing the actual response time of the physical model.

Tilt test

In this test, the TARP has received a target angle at 45°, moving the tilt motor. TARP saves the position as it moves towards the target angle. Based on the simulated model, the physical

model should reach the target angle of 45° within 8.07 s, oscillating within ±0.36°. Figure 4.69 shows the saved positions, giving the step response of the tilt motor targeting 45° versus the simulated model.

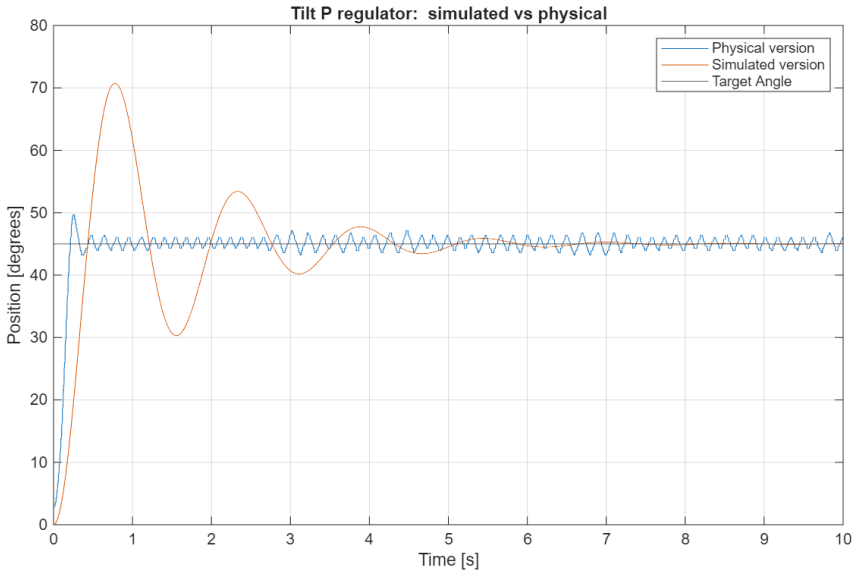


Figure 4.69: Step response of the tilt p-controller of the physical version vs. the simulated version moving towards a target angle of 45°.

Note, while testing the tilt motor, it was found the motor has a backlash 3°. This is due to the gearing within the tilt motor.

Figure 4.69 shows the initial slope of the physical model is steeper than the simulated model. This indicates that the motor is more powerful or the friction is less than assumed in the simulated model. The overshoot is far less than simulated model, which indicates that the CL PM is much higher than simulated. Figure 4.69 also shows that the tilt motor oscillates approximately ±1.5° after 0.5 s. The main cause of the oscillations is the backlash in the gearing of the motor, which means when switching direction, there is approximately 3° where the controller is not able to control the position tilt. This leaves the tilt motor imprecise and currently the control of tilt motor does not comply with the expected resolution of 0.36°. It seems as if the physical version is much faster than the simulated model, but due to the large imprecision of the physical version, no conclusions can be made. The tilt motor needs to be mechanically improved before the test results of the physical and simulated model can be compared. To improve the tilt controller the gearing needs to be removed to remove the backlash. Removing the backlash, should help make the physical version more precise. Only after this, conclusions can be made on the physical versus the simulated model.

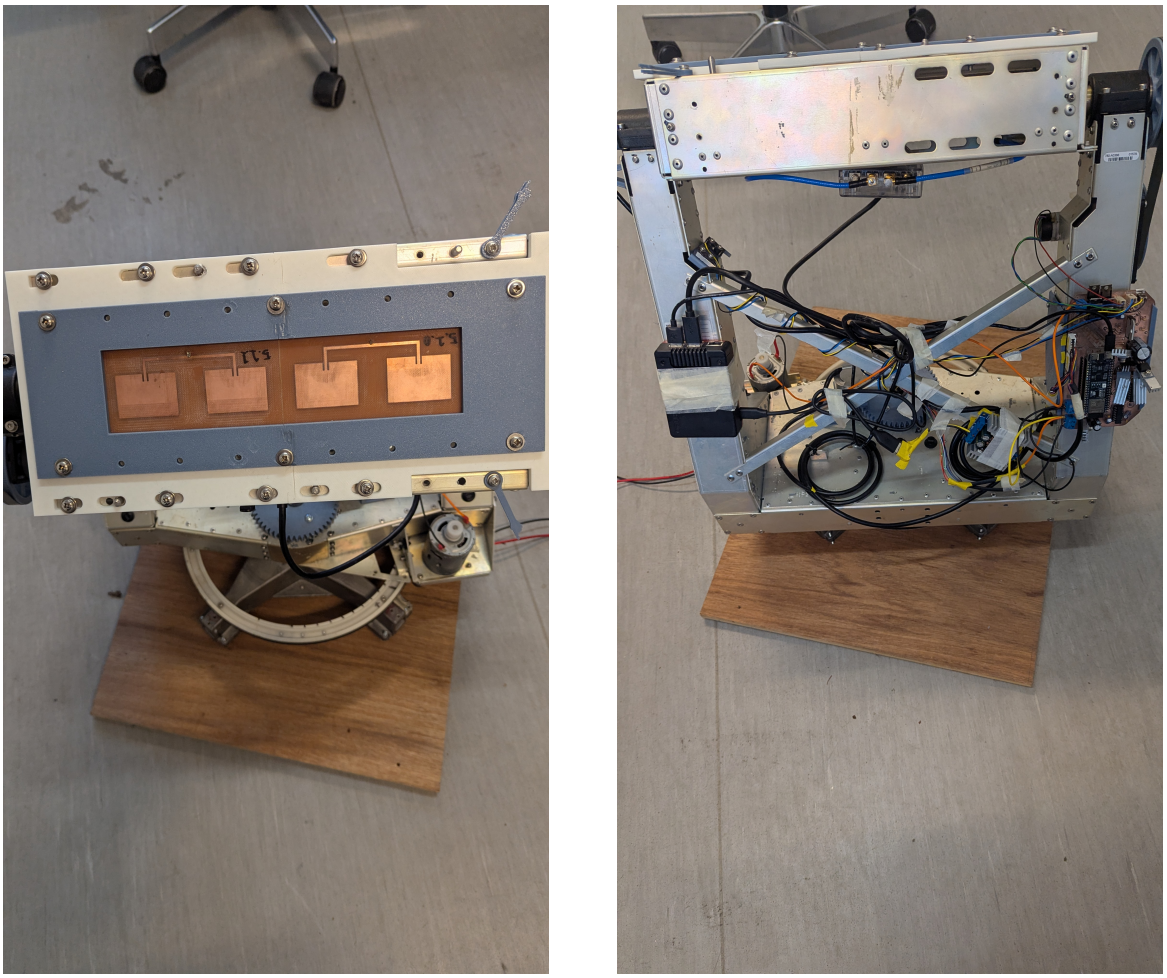
Chapter 5

Integration

This chapter explains the integration of all the submodules from Chapter 4, followed by a description of the different tests the assembled prototype should be subjected to.

5.1 Physical Integration

Regarding physical integration two issues must be handled. The first concerns mounting to the TARP, specifically the antennas, the Raspberry Pi and SDR. The Pi and SDR are mounted using masking tape, a 3D printed mount has been constructed for the antennas. Secondly powering the Pi. This is done using a generic buck converter to step down the 12 V supply to a stable 5 V output. This was mounted directly on the TARP to reduce the voltage drop, due to wire lengths. Pictures of the setup can be seen on Figure 5.1.



(a) (b)

Figure 5.1: Pictures of the physical setup, where all submodules have been combined.

5.2 Code Integration

The current submodules explained in Chapter 4 lack the ability to interact. This section explains the code required to integrate these submodules together. Much of the code from the submodules can be reused. Specifically the beamforming algorithm in Figure 4.47, the SDR code from Appendix 16 and some of the TARP control code from Appendix 23. The modifications made to the control code is removal of test functions, and implementation of serial communication. This is already touched upon in Section 4.4, but it is essential as it establishes communication between the ESP32 controlling the motors and the Pi. This new code can be seen in Appendix 24. The Pi part of the serial communication can be seen in Appendix 25. The code integrating all these individual submodules can be seen in Figure 5.2. Line $12-15$ is the starting position, which is currently set to 180° for azimuth and 155° for tilt, as this results in the elevation being about 45°. Line $17-22$ is a continuously tracking loop, which takes a sample, uses the beamforming algorithm to calculate a strongest signal angle, then moves the motor to this position and starts over.

```
1  from serial import Serial
2  from SerialRW.MotorComms import MotorCtrl
3  from bf import beamforming_das
4  from sdr_ctrl import sdr_ctrl
5  from time import sleep
6
7  sdr = sdr_ctrl(40e6, 2.44175e9)
8  esp = Serial(port='/dev/ttyUSB0', baudrate=115200, timeout=1)
9  motor = MotorCtrl(esp)
10
11
12 motor.azi(180)
13 sleep(1)
14 motor.tilt(155)
15 sleep(1)
16
17 while True:
18     arr = sdr.sample(900)
19     azi_pos, tilt_pos = motor.read_pos()
20     bf = beamforming_das(arr, 0.5, 2)
21     print("bf: ", bf)
22     motor.azi(azi_pos+bf)  # Move to biggest signal
```

Figure 5.2: Main code for the Pi, running and controlling all submodules.

5.3 Testing of Prototype

As stated in Section 4.3, the beamforming algorithm currently does not work, due to the placement of the antennas. This means a test for detection time is performed, while tests for validation of tracking accuracy, tracking speed and detection range is not performed. Though all the test journals have been prepared along with a presentation of them below.

5.3.1 Detection Time

This test evaluates the prototypes detection time of a drone. In Section 3.3 the specification of maximum scanning time is set to 43.47 s. To test this, the new integration code is modified, so that it moves to 120° and after the location is stable ($\pm 0.288°$, which happens after approximately 3 s) the beamforming algorithm will be executed once. The modified code is depicted on Figure 5.3. This time is then multiplied by three to get the detection time for 360°. The test journal can be seen in Appendix 11.

```
Main:
from serial import Serial
from SerialRW.MotorComms import MotorCtrl
from bf import beamforming_das
from sdr_ctrl import sdr_ctrl
from time import sleep, time

sdr = sdr_ctrl(40e6, 2.44175e9)
esp = Serial(port='/dev/ttyUSB0', baudrate=115200, timeout=1)
motor = MotorCtrl(esp)

for _ in range(0,5):
    motor.azi(180)
    sleep(1)
    motor.tilt(155)
    sleep(1)

    t0 = time()
    motor.azi(180-120)
    sleep(3)#wait until azi is stable
    arr = sdr.sample(900)
    beamforming_das(arr, 0.5, 2)
    print(time()-t0)
sdr.close()
```

Figure 5.3: Modified main code used to test the detection time.

The test results show that the prototype has a 360° average scanning time of 15.18 s, where as the specification for the detection time is 43.47 s. This means that the prototype complies with the specification.

5.3.2 Validation of Tracking Accuracy

This test validates that the tracking accuracy of the prototype is better than $11.46° \cdot 10^{-3}$ for azimuth and $5.73° \cdot 10^{-3}$ for elevation, as set forward in Section 3.3. Since the constructed antenna is a 2x1 array, it does not allow for beamforming in the elevation axis. Therefore, this only validates the azimuth accuracy. This is done by placing a dipole antenna at different known locations on the horizontal plane and saving the angles the beamforming algorithm outputs. The dipole antenna must output a signal within the chosen frequency band to ensure that the prototype works as intended. This is done with a dipole antenna to prove the concept of the prototype, but in later iterations it should be tested with the use case drone. To test this, the code should be modified to save the output of beamforming and the position reported by the encoders. Combining the beamforming and position, it is possible to find the absolute

position of which the prototype estimates the signal to originate from. This should then be compared with the actual physical location. A test journal can be seen in Appendix 12, where the test is described in more detail. Figure 5.4 illustrates the physical test setup.

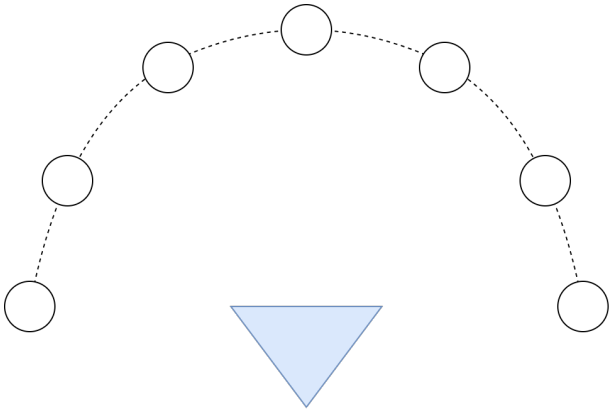


Figure 5.4: Diagram of physical test setup for tracking test. The circles are points with known angles to boresight and the upside down triangle is the prototype.

5.3.3 Tracking Speed

The goal of this test is to ensure that the prototype can track the use case drone, in real time, moving at its maximum speed of 82.8 km/h at a radius of 2 km [39]. This test must be conducted in a place with minimal interfering signals (such as WIFI) as this could potentially create disturbances. A test journal that documents the test procedure can be seen in Appendix 13.

5.3.4 Detection Range

This test validates that a drone at 8382 m can be detected as stated in Section 3.3. It is however already known that this specification is not met with the current RF front end and antennas, as they do not met their gain specifications (seen in Section 4.1 and 4.2). The maximum distance where a SNR of 5 dB is possible, can be found using the simplified Friis Equation (5.1) [46], and solving for the distance.

$$P_r = G_r \cdot G_t \cdot \frac{\lambda^2}{(4\pi \cdot D)^2} \cdot P_t \quad \text{[W]} \quad (5.1)$$

Solving for D in Equation (5.1):

$$D = \frac{\lambda}{4\pi} \cdot \sqrt{\frac{G_r \cdot P_t}{P_r}}$$

The actual gain from the antennas is 9.28 dBi $\approx$ 8.47, this is seen in Section 4.1.8. The wavelength, power transmitted and power received (using 5 dB SNR) is found in Section 3.3.2, these being 120.8 mm, 17 dBm and -89.6 dBm respectively. The two powers can be converted using Equation 5.2: 50.12 mW and 1.1 pW [52].

$$P_{watt} = 10^{\frac{P_{dBm}-30}{10}} \quad (5.2)$$

Inserting these into the equation results in a distance of:

$$D = \frac{120.8\,\text{mm}}{4\pi} \cdot \sqrt{\frac{8.47 \cdot 50.12\,\text{mW}}{1.1\,\text{pW}}} \approx 5981\,\text{m}$$

The test of whether this this distance actually can be achieved has not been carried out.

Chapter 6

Discussion

This chapter discusses the findings made during the development of the prototype while evaluating the performance and limitations of the current V1 prototype. The intention of this is to serve as a foundation for further development and refinement.

In general, the current prototype V1 does not meet the requirements set forward in Section 3. This is seen in Chapter 5, where testing of the entire prototype is severely limited. This is due to the antennas mistakenly being placed a whole wavelength apart instead of half a wavelength. This mistake is caused by a misunderstanding of the antenna array theory. It was assumed that to get a separation of $\lambda/2$, the edges of each element should be $\lambda/2$ apart. But when testing the beamforming, it was discovered that the distance between them had to be based on the center of the antennas. This will cause an additional issue when a 2x2 array must be made. The current subarray on the long side is already λ, which leaves no feasible arrangement for placing two elements adjacent with $\lambda/2$ spacing. A solution must be found to allow for beamforming in both the horizontal and vertical axis. One solution is to connect each element directly to the front end, without the use of subarrays. That will however require 8 inputs on the front end.

In order to create a V2 of the prototype, several issues must be resolved. Firstly, the maximum gain in the SDR should be increased, either by using another SDR or creating a custom front end, which should be investigated further when developing a prototype V2. Secondly, the gain of the antenna should also be increased, this can be done using an 8-element array. This antenna array should also have two more channels for the elevation, to enable beamforming in this direction as well. Thirdly, the bandwidth should be increased for the SDR from 40 MHz to ensure the required bandwidth of 83.5 MHz is sampled. This also means that the current sampling rate of the SDR at 40 MHz needs to be increased to 83.5 MHz. Fourth, the controller for the TARP should be improved to get the required accuracy and reduce oscillations. This can be improved by designing a more complex controller than the current P-controller. With regards to the elevation, the mechanical part of the elevation is unnecessary if there is a 2x2 antenna array. Locking the elevation at 45 ° is enough to cover the airspace when the prototype is rotating in the azimuth. This would also remove the tilt motor, removing the slack issues with the gearing of the motor. Lastly, an algorithm capable of filtering out irrelevant signals from non-target sources, such as Wi-Fi access points, is required. This might work by detecting motion. When the algorithm then detects a non-target, it signals to the rest of the system to ignore it.

Looking at the choice of solution regarding solving the problem of protecting the airspace against drones, some critical concerns arise. If prototype V1 is proven to work with the further development, the prototype is not designed to handle multiple drones in the detection zone. To keep track of multiple drones at once, it would constantly need to stay in scanning mode, meaning the location refresh rate on individual drones is significantly lowered. This

happens as it will be limited to the speed of the physical rotation of the TARP. This ultimately means that the current choice of solution needs to be changed. The prototype needs to be split into two separate prototypes, one that scans and keeps track of all the drones in the airspace, and the other that points and shoots down the drones. The scanning and tracking could be solved by either using one antenna array that rotates continuously in the azimuth. Then, using the beamforming algorithm in the vertical and horizontal planes to detect and track the drones. Another option would be to create a motionless prototype that has three separate antenna arrays, covering 120° in the azimuth of the airspace, and 90° in the vertical of the airspace. The current designed modules in this project would still be applicable in these new solutions. The point-and-shoot device could use the current control system and point to the drone, based on the location received from the scan and track device.

The RF sensing approach used in this prototype results in limitations on the types of drones that can be detected. Initially, drones utilizing wired control or fully autonomous drones are not detectable. Further advanced drones equipped with directional antennas may produce quite weaker signals at the antenna, thereby complicating detection and tracking. Additionally, it is not able to accurately measure the distance to the drone. This means that the RF sensing is not a solo solution. In further development, another module(s) must be added, able to detect the distance to the drone, along with detecting the wired drones and drones with directional antennas.

Chapter 7

Conclusion

This project aimed to develop a prototype capable of detecting and tracking a recreational drone within restricted Danish airspace, as put forth in the problem statement. A specific use case was thus defined in order to simplify the development of this prototype, as there exist many recreational drones with varying specification. A DJI RTK 210 Matrice was then chosen for the use case. The restricted airspace was divided into three zones based on Danish legislation. These zones laid the ground for the range requirement, for which the prototype should be able to detect drones, as mentioned in Section 4.1.4. This range resulted in a required antenna gain of 12.21 dBi. This gain was not reached as the current gain is 9.28 dBi, as measured in Section 4.1.8. Reaching the required gain would require twice as many antennas as currently implemented, which would require a doubling of the available ports in the front end. More ports would also allow for a two dimensional array, allowing beamforming in both azimuth and elevation. Despite this, the current antennas just about live up to the rest of the requirements defined in Section 3.3.2.

The radius of these zones and the maximum speed of a DJI RTK 210 Matrice drone set the foundation for the requirements of both detection time and precision. These requirements were split 80/20 between the mechanical system and the beamforming as specified in Section 3.3.1. The prototype is well within specs regarding the detection time as the mechanical system takes about 15 s to perform a full scan, which is about half of the required 34.78 s. As for the beamforming has been simulated to be 0.669 s, which is much less than the specification of 8.69 s. The precision of the tracking is however not as compliant with the specifications. With the TARP oscillating about 0.25 ° in azimuth and about 1.5 ° in the tilt, this is way too much for the required $9.17\,^\circ \cdot 10^{-3}$ and $4.58\,^\circ \cdot 10^{-3}$ respectively. Regarding beamforming, it is determined in Section 4.3.3 that it currently does not achieve the desired accuracy. A major reason for this lack of accuracy is the antenna array, which currently has a configuration that limits the precision of the algorithm. This could be the reason for the results of the test detailed in Section 4.3.3. This test shows that the angle of arrival is measured to be $-1°$ at $0°$ with preceding angles being incorrect. This does not fulfill the required $2.28° \cdot 10^{-3}$ horizontally.

The RF front end section (Section 4.2) has shown that the SDR used in this prototype is not adequate for what is required. It is seen that it only fulfills two out of six specifications, that being the impedance match and the resolution. A future prototype could solve this using a more expensive SDR or a custom front end. Despite of these short comings, the current SDR is still sufficient for proving the concept as mentioned in Section 4.2.

So to conclude, the current version of the prototype is not able to track a signal. Considerations for further development of the prototype and ways to fix many of the current problems have been documented. This will allow for a future version that should be able to track a signal from a drone. Version 1 does not have these changes implemented due to a lack of time. So with the current state of the prototype, and based on the amount of specifications not attained

by its submodules, the problem statement cannot be definitively concluded upon.

Bibliography

[1] U.S. Attorney's Office. *Culver City Man Agrees to Plead Guilty to Recklessly Crashing Drone into Super Scooper Firefighting Aircraft During Palisades Fire*. 2025. URL: https://www.justice.gov/usao-cdca/pr/culver-city-man-agrees-plead-guilty-recklessly-crashing-drone-super-scooper.

[2] Hallie Detrick. *Gatwick's December Drone Closure Cost Airlines $64.5 million*. 2019. URL: https://fortune.com/2019/01/22/gatwick-drone-closure-cost/.

[3] Asta Holst Bach and Mads Søndergaard Thøgersen. *Markant flere droner i Aalborgs luftrum skaber problemer: Det kan være 'ekstremt farligt', lyder det fra Forsvaret*. 2024. URL: https://www.dr.dk/nyheder/indland/markant-flere-droner-i-aalborgs-luftrum-skaber-problemer-det-kan-vaere-ekstremt.

[4] Trafikstyrelsen. *Nye vilkår for at flyve med droner efter 1. januar*. 2023. URL: https://www.trafikstyrelsen.dk/nyheder/2023/dec/nye-vilkaar-for-at-flyve-med-droner-efter-1-januar.

[5] Frederik Mahler Bank. *Politiet undersøger droneflyvninger ved havnen i Køge*. 2025. URL: https://www.dr.dk/nyheder/seneste/politiet-undersoeger-droneflyvninger-ved-havnen-i-koege.

[6] Transportministeriet. *Høring af Forslag til lov om ændring af forskellige love på transportministeriets område (Styrket beredskab på transportområdet m.v.)* 2025. URL: https://hoeringsportalen.dk/Hearing/Details/70295.

[7] CUAS-HUB. *Non-Cooperative Drone*. URL: https://cuashub.com/en/glossary/non-cooperative-drone/.

[8] UAVSYSTEMS INTERNATINAL. *RECREATIONAL USE OF DRONES*. URL: https://uavsystemsinternational.com/blogs/drone-guides/recreational-use-of-drones?.

[9] Qassim A. Abdullah. *Classification of the Unmanned Aerial Systems*. 2023. URL: https://www.e-education.psu.edu/geog892/node/5.

[10] UAV MODEL. *UAVs: Types, Classifications, and Key Features Explained*. 2025. URL: https://www.uavmodel.com/blogs/news/uavs-types-classifications-and-key-features-explained.

[11] Olivia Campbell. *The Ultimate Drone Guide 2025: From Military UAVs to Micro Surveillance Drones*. 2025. URL: https://www.arcadian.ai/blogs/blogs/the-ultimate-drone-guide-2025-from-military-uavs-to-micro-surveillance-drones.

[12] Inside FPV. *Exploring the Different Types of Drones: A Comprehensive Overview*. 2024. URL: https://insidefpv.com/blogs/blogs/exploring-the-different-types-of-drones-a-comprehensive-overview.

[13] JOUAV. *How Fast Can a Drone Fly? Top Speeds of Various Types*. 2025. URL: https://www.jouav.com/blog/how-fast-can-a-drone-fly.html.

[14] Jacob Stoner. *Drone Communication Systems*. 2024. URL: https://www.flyeye.io/drone--technology-communication/.

[15] Sarah Greiner. *Omnidirectional Antenna Radiation Patterns Explained*. 2019. URL: https://www.mpantenna.com/omnidirectional-antenna-radiation-patterns/.

[16] European Telecommunications Standards Institute. *ETSI EN 300 328 V2.2.2*. 2019. URL: https://www.etsi.org/deliver/etsi_en/300300_300399/300328/02.02.02_60/en_300328v020202p.pdf.

[17] Powertec. *2.4 GHz*. URL: https://portal.powertec.com.au/technical-library/wireless/core-concepts/itu-bands/24-ghz.

[18] Fang Liu Huan LV and Naichang Yaun. *Drone Presence Detection by the Drone's RF Communication*. 2021. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1738/1/012044/pdf.

[19] Electronics Communications Committee. *Compatibility studies between RDSS and other services in the band 2483.5-2500 MHz*. 2010. URL: https://docdb.cept.org/download/608.

[20] Czech Telecommunication Office. *Detail of Frequency Band*. URL: https://spektrum.ctu.gov.cz/en/band/2483.5-2500-mhz.

[21] Vladislav Semenyuk et al. "Advances in UAV detection: integrating multi-sensor systems and AI for enhanced accuracy and efficiency". In: *International Journal of Critical Infrastructure Protection* 49 (2025), p. 100744. ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2025.100744. URL: https://www.sciencedirect.com/science/article/pii/S1874548225000058.

[22] Ulzhalgas Seidaliyeva et al. *Advances and Challenges in Drone Detection and Classification Techniques: A State-of-the-Art Review*. 2023. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC10780901/.

[23] Merrill Skolnik. *The radar handbook*. McGraw Hill, 2008. ISBN: 978-0-07-148547-0.

[24] GeeksforGeeks. *FMCW Radar*. Accessed: October 23, 2024. 2024. URL: https://www.geeksforgeeks.org/electronics-engineering/fmcwr-radar/ (visited on 10/23/2024).

[25] V. P. Riabukha. "Radar Surveillance of Unmanned Aerial Vehicles (Review)". In: *Radioelectronics and Communications Systems* 63.11 (Nov. 2020), pp. 561–573. ISSN: 1934-8061. DOI: 10.3103/S0735272720110011. URL: https://doi.org/10.3103/S0735272720110011.

[26] Samiur Rahman and Duncan A. Robertson. "Radar micro-Doppler signatures of drones and birds at K-band and W-band". In: *Scientific Reports* 8.1 (Nov. 2018), p. 17396. ISSN: 2045-2322. DOI: 10.1038/s41598-018-35880-9. URL: https://doi.org/10.1038/s41598-018-35880-9.

[27] SIMON R. SAUNDERS and ALEJANDRO ARAGON-ZAVALA. *ANTENNAS AND PROPAGATION FOR WIRELESS COMMUNICATION SYSTEMS*. John Wiley & Sons, Inc, 2007. ISBN: 978-0-470-84879-1.

[28] Transportministeriet. *Bekendtgørelse af lov om luftfart*. 2025. URL: https://www.retsinformation.dk/eli/lta/2025/570#idf1e57361-1752-4634-a873-4187055b1dcc.

[29] Trafikstyrelsen. *Flyvepladser og planlægning*. 2024. URL: https://www.trafikstyrelsen.dk/arbejdsomraader/luftfart/flyvepladser/flyvepladser-og-planlaegning.

[30] Transportministeriet. *Bekendtgørelse om ændring af bekendtgørelse om supplerende bestemmelser til gennemførelsesforordning (EU) 2019/947 om regler og procedurer for operation af ubemandede luftfartøjer*. 2025. URL: https://www.retsinformation.dk/eli/lta/2025/527.

[31] José Manuel Barroso. *COMMISSION REGULATION (EU) No 139/2014*. 2014. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32014R0139#fnp_1.

[32] Antonio Tajani and Karoline Edtstadler. *REGULATION (EU) 2018/1139 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL*. 2018. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:32018R1139#fnp_1.

[33] IEEE. *802.11ax-2021 - IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN*. 2021. URL: https://ieeexplore.ieee.org/document/9442429.

[34] Digitaliseringsstyrelsen. *Frekvensplan juli 2025*. 2025. URL: https://digst.dk/media/mf5bi11c/frekvensplan-juli-2025-1.pdf.

[35] European communications office. *ECO Frequency Information System*. URL: https://efis.cept.org/.

[36] *LBK nr 958 af 22/06/2022*. 2022. URL: https://www.retsinformation.dk/eli/lta/2022/958.

[37] ETSI. *Electro Magnetic Compatibility*. 2025. URL: https://www.etsi.org/technologies/emc.

[38] European Telecommunications Standards Institute. *ETSI EN 300 220-1 V3.1.1*. 2017. URL: https://www.etsi.org/deliver/etsi_en/300200_300299/30022001/03.01.01_60/en_30022001v030101p.pdf.

[39] DIJI. *M200 SERIES COMPARISON*. URL: https://www.dji.com/dk/products/compare-m200-series.

[40] Constantine A. Balanis. *ANTENNA THEORY*. John Wiley & Sons, Inc, 2016. ISBN: 978-1-118-64206-1.

[41] Ramanda Grace Aulia. *Design Of A 2.4 Ghz Microstrip Patch Antenna For Wifi Communication Using Cst Studio Suite 2019*. 2025. URL: https://jotecs.org/index.php/jotecs/article/view/37/6.

[42] Northeastrf. *Return Loss to Mismatch Calculator*. URL: https://northeastrf.com/return-loss-calculator/.

[43] Steve Ponton. *Understanding VSWR vs. Return Loss: A Comprehensive Guide*. URL: https://rfcom.co.uk/understanding-vswr-vs-return-loss-a-comprehensive-guide/.

[44] Antenna Test Lab. *Return Loss and VSWR*. URL: https://antennatestlab.com/antenna-education-tutorials/return-loss-vswr-explained.

[45] Zachariah Peterson. *The Mysterious 50 Ohm Impedance: Where It Came From and Why We Use It*. 2021. URL: https://resources.altium.com/p/mysterious-50-ohm-impedance-where-it-came-and-why-we-use-it.

[46] Gert Frølund Pedersen. *Antenna Arrays and Microstrip antennas*. 2025. URL: https://www.moodle.aau.dk/pluginfile.php/3681630/mod_resource/content/2/CommunicationSystemsLecture2PresentationProblems.pdf.

[47] DJI. *M200 SERIES COMPARISON*. URL: https://www.dji.com/dk/products/compare-m200-series.

[48] Marco Prevedelli Leonardo Ricci Alessio Perinelli. *The Physics Behind Electronics*. Springer, 2024. ISBN: 978-3-031-55460-5.

[49] Clay S. Turner. *Johnson-Nyquist Noise*. 2007. URL: https://pearl-hifi.com/06_Lit_Archive/14_Books_Tech_Papers/Johnson-Nyquist%20Noise.pdf.

[50] John Cappelen. *Temperaturen i Danmark*. 2021. URL: https://www.dmi.dk/klima/temaforside-klimaet-frem-til-i-dag/temperaturen-i-danmark.

[51] Matthias Rudolph Peter Heymann. *A Guide to Noise in Microwave Circuits*. Wiley-IEEE Press, 2021. ISBN: 978-1-119-85939-0.

[52] Digikey. *dBm to Watts conversion*. URL: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-dbm-to-watts.

[53] Julius O. Smith III. *Spectral Audio Signal Processing*. W3K Publishing (December 16, 2011), 2007. ISBN: 0974560731.

[54] Por Tamara Schmitz. *Filter Selection & Design: The Gateway to System Performance*. URL: https://www.digikey.es/es/articles/filter-selection--design-the-gateway-to-system-performance.

[55] Michael Steer. *MICROWAVE AND RF DESIGN I - RADIO SYSTEMS*. LibreTexts.

[56] Marc Lichtman. *3. IQ Sampling*. 2025. URL: https://pysdr.org/content/sampling.html#iq-sampling.

[57] SAE Media Group. *Design Approaches for Established and Emerging RF Receiver Architectures*. 2024. URL: https://www.mobilityengineeringtech.com/component/content/article/50045-design-approaches-for-established-and-emerging-rf-receiver-architectures.

[58] Peter Koch. *Digital Signal processing (ESD5/DE5/IV5-Elektro)*. 2025. URL: https://www.moodle.aau.dk/pluginfile.php/3806046/mod_resource/content/1/Slides_8_25.pdf.

[59] Texas Instruments Precision Labs. *Types of noise in ADCs*. URL: https://www.ti.com/content/dam/videos/external-videos/en-us/3/3816841626001/6117424453001.mp4/subassets/adcs-types-of-noise-in-adcs-presentation.pdf.

[60] Georgios Giannakopoulos and khushbu Mehboob Shaikh. *Phased Array Antennas: Advancements and Applications*. 2025. URL: https://www.preprints.org/manuscript/202502.1016/v1.

[61] Cisco. *Antenna Patterns and Their Meaning*. URL: https://www.avw.co.nz/wp-content/uploads/2020/08/Antenna-Patterns-and-Their-Meaning.pdf.

[62] Di-Patch Antenna Array Comparison. *Hazim A. Abdulsada, Sagar Kumar Sharma and Huma Razzaq*. 2019. URL: https://ieeexplore.ieee.org/document/8942518.

[63] Gert Frølund Pedersen. *Antenna Arrays and Microstrip antennas*. 2025. URL: https://www.moodle.aau.dk/pluginfile.php/3681633/mod_resource/content/3/CommunicationSystemsLecture3Part1Presentation.pdf.

[64] tutorialspoint. *Antenna Theory*. 2016. URL: https://www.tutorialspoint.com/antenna_theory/antenna_theory_tutorial.pdf.

[65] Zachariah Peterson. *FR4 Dielectric Constant and Material Properties*. 2021. URL: https://resources.altium.com/p/fr4.

[66] Komponenten. *PCB fraeer*. 2020. URL: https://www.komponenten.es.aau.dk/fileadmin/komponenten/billeder/Information/PCBFraeserGuide2020.pdf.

[67] Seyed S. Ahranjan Ali Khaleghi and Ilangko Balasingham. *High Gain and Wideband Stacked Patch Antenna for S-Band Applications*. 2018. URL: https://www.jpier.org/ac_api/preview.php?t=ab&id=18031505.

[68] Ansys. *How to Calculate and Increase Antenna Gain*. URL: https://www.ansys.com/blog/calculate-and-increase-antenna-gain.

[69] *Power Splitter/Combiner Coaxial*. ZFRSC-183-S+. Mini-Circuits. 2013. URL: https://www.minicircuits.com/pdfs/ZFRSC-183-S+.pdf.

[70] Highleap Electronic. *The Factors Behind Chip Cost in PCB Design and Assembly*. URL: https://hilelectronic.com/chip-cost/.

[71] All About Circuits. *Practical Guide to Radio-Frequency Analysis and design*. URL: https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/selected-topics/the-benefits-of-an-intermediate-frequency-in-rf-systems/.

[72] DigiKey. *The Basics of Mixers*. 2020. URL: https://www.digikey.dk/en/articles/the-basics-of-mixers.

[73] Next electronics. *Zero-IF Receiver Design*. URL: https://next.gr/tutorials/rf-and-wireless-basics/zero-if-receiver-design-tutorial.

[74] C. Li and D.M.M.-P. Schreurs. "Chapter 1 - Fundamentals of microwave engineering". In: *Principles and Applications of RF/Microwave in Healthcare and Biosensing*. Ed. by Changzhi Li et al. Academic Press, 2017, pp. 1–52. ISBN: 978-0-12-802903-9. DOI: https://doi.org/10.1016/B978-0-12-802903-9.00001-1. URL: https://www.sciencedirect.com/science/article/pii/B9780128029039000011.

[75] NUAND. *bladeRF 2.0 micro xA9*. URL: https://www.nuand.com/product/bladeRF-xA9/.

[76] Ryan Thompson. *2023.02 Release – 122.88MHz instantaneous bandwidth*. 2023. URL: https://www.nuand.com/2023-02-release-122-88mhz-bandwidth/.

[77] *Data Sheet AD9361 RF Agile Transceiver*. ADRV9361. ANALOG DEVICES. 2024. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ad9361.pdf.

[78] Damien Garcia. *So you think you can DAS?* 2019. URL: https://ewh.ieee.org/conf/ius/2019/media/files/0995.pdf.

[79] Jørgen Grythe. *Beamforming algorithms - beamformer*. 2015. URL: https://norsonic.no/wp-content/uploads/2023/09/TN-beamformers.pdf.

[80] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Pearson Education Limited, 2014. ISBN: 978-1-292-02572-8.

[81] Marc Lichtman. *19. Beamforming and DOA*. 2025. URL: https://pysdr.org/content/doa.html.

[82] Saanvi Kulkarni et al. "A Comprehensive Review of Direction of Arrival (DoA) Estimation Techniques and Algorithms". In: *Journal of Electronics and Electrical Engineering* (2024).

[83] Wolfram. *Triangle Inequality*. URL: https://mathworld.wolfram.com/TriangleInequality.html.

[84] Programming Language and compiler Benchmarks. *C VS Python benchmarks*. URL: https://programming-language-benchmarks.vercel.app/c-vs-python.

[85] NumPy Team. *NumPy*. 2025. URL: https://numpy.org/.

[86] *DBS36E-S3EK00500*. SICK. 2025. URL: https://www.sick.com/media/pdf/0/70/970/dataSheet_DBS36E-S3EK00500_1060543_en.pdf.

[87] Jan Tommy Gravdahl Olav Egeland. *Modeling and Simulation for Automatic Control*. 2003. ISBN: 82-92356-01-0.

[88] WILLIAM MOEBS SAMUEL J. LING JEFF SANNY. *University Physics Volume 1*. 2016. ISBN: 978-1-947172-20-3.

[89] Abbas Emami-Naeini Gene F. Franklin J. David Powell. *Feedback Control of Dynamic Systems*. 2019. ISBN: 978-1-292-27454-6.

[90] R. Mark Nelms J. David Irwin. *Engineering Circuit Analysis - Twelfth Edition*. John Wiley and Sons, 2021. ISBN: 978-1-119-66796-4.

[91] JOHN MAZURKIEWICZ. *The basics of motion control*. 1995. URL: https://fab.cba.mit.edu/classes/961.04/topics/motion_control2.pdf.

[92] Clarence W. de Silva. *Modeling of Dynamic Systems with Engineering Applications 2nd Edition*. 2022. ISBN: 978-1-003124-47-4.

[93] Wolfram. *Sign*. URL: https://mathworld.wolfram.com/Sign.html.

[94] Analog Devices. *Micro Tutorial 1: Understanding DC Electrical Characteristics of Microcontrollers*. 2002. URL: https://www.analog.com/en/resources/technical-articles/micro-tutorial-1-understanding-dc-electrical-characteristics-of-microcontrollers.html.

[95] *RH158*. Micro Motors. 2022. URL: https://www.micromotors.eu/wp-content/uploads/2022/07/RH158.pdf.

[96] *L298 - Dual full-bridge driver*. ST. 2023. URL: https://www.st.com/resource/en/datasheet/l298.pdf.

[97] Brendan Massey. *NMOS Transistors and PMOS Transistors Explained*. 2005. URL: https://builtin.com/hardware/nmos-transistor.

[98] Venus Kohli. *The flyback diode explained*. 2025. URL: https://www.power-and-beyond.com/the-flyback-diode-explained-a-1ef338a15d64171ceaac8c55a8f77672/#:~:text=Definition%20Flyback%20diodes%20Flyback%20diodes%20are%20protective,to%20%E2%80%9CFlyback%E2%80%9D%20to%20the%20power%20source%20(switch%2Dbattery)..

[99] Monash University. *Lenz's law and its applications*. URL: https://www.monash.edu/student-academic-success/physics/generation-and-transmission-of-electricity/lenzs-law-and-its-applications#Lenzs_law.

[100] *BCC546*. Fairchild. 2024. URL: https://cdn.sparkfun.com/assets/d/5/e/5/d/BC547.pdf.

[101] *SNx4HC08 Quadruple 2-Input AND Gates*. Texas Instruments. 2025. URL: https://www.ti.com/lit/ds/symlink/sn74hc08.pdf.

[102] *IRF9530*. Vishay. 2024. URL: https://www.vishay.com/docs/91076/irf9530.pdf.

[103] *SUP85N10-10*. Vishay. 2010. URL: https://www.vishay.com/docs/71141/supsub85.pdf.

[104] Khan Academy. *L'Hôpital's rule review*. URL: https://www.khanacademy.org/math/differential-calculus/dc-context-app/dc-lhopital-composite-exp/a/lhopitals-rule-review.

Chapter 8

Appendices

1 Superposition of Coulomb Friction

This appendix proves that superposition does not hold for the coulomb friction. Superposition requires that the following holds (Where $u(t)$ is the input and $y(t)$ is the output) [89]:

$$u(t) = \alpha_1 u_1(t) + \alpha_2 u_2(t) \qquad , \qquad y(t) = \alpha_1 y_1(t) + \alpha_2 y_2(t)$$

Which it does not for $\tau_{fk} = B_v \cdot \omega(t) + \tau_c$ (using $\alpha_1 = \alpha_2 = 1$ for simplicity). The initial states:

$$u(t) = \omega_1 + \omega_2 \qquad , \qquad y(t) = \tau_1 + \tau_2 \qquad , \qquad y(t) = B_v \cdot u(t) + \tau_c$$

Where u is the input (angular velocity, ω), y is the output (combined friction torque).
Resulting in

$$\tau_1 = B_v \cdot \omega_1 + \tau_c \qquad , \qquad \tau_2 = B_v \cdot \omega_2 + \tau_c$$

$$y(t) = B_v \cdot \omega_1 + \tau_c + B_v \cdot \omega_2 + \tau_c = B_v(\omega_1 + \omega_2) + \tau_c \cdot 2$$
$$y(t) = B_v \cdot u(t) + \tau_c = B_v(\omega_1 + \omega_2) + \tau_c$$

$$B_v(\omega_1 + \omega_2) + \tau_c \cdot 2 \neq B_v(\omega_1 + \omega_2) + \tau_c$$

The physical system is therefore non-linear.

2 Proof of Antenna Array Size Equation

The goal is to show the relation between linear gain and number of elements in an array. Equation (8.1) shows the relation between total electromagnetic field and electromagnetic field of one antenna multiplied with the array factor. The following equations are based on the book Antenna Theory [40].

$$\vec{E}_{total} = \vec{E}_n \cdot AF \qquad [\mathrm{V/m}] \quad (8.1)$$

The electromagnetic field can also be defined in relation to the radiation intensity (U), radius (r) and intrinsic impedance (η).

$$U = \vec{E}^2 \cdot \frac{r^2}{2 \cdot \eta} \Rightarrow \vec{E} = \sqrt{\frac{\eta \cdot U \cdot 2}{r^2}}$$

This is then inserted into equation (8.1):

$$\sqrt{\frac{\eta \cdot U_{total} \cdot 2}{r^2}} = \sqrt{\frac{\eta \cdot U_n \cdot 2}{r^2}} \cdot AF$$

This can then be reduced to:

$$U_{total} = U_n \cdot AF^2 \tag{8.2}$$

Now the radiation intensity can be related to the gain (G) through directivity (D_0):

$$G = e_{cd} \cdot D_0 = e_{cd} \cdot 4 \cdot \pi \cdot \frac{U}{P}$$

P is the radiated power. Assuming the antenna is ideal, the efficiency (e_{cd}) is 1, which is never the case in the real world, but it is simplified to create this proof. Then the radiation intensity (U) can be isolated to:

$$U = \frac{G \cdot P}{4 \cdot \pi}$$

This is then inserted into Equation (8.2)

$$\frac{G_{total} \cdot P_{total}}{4 \cdot \pi} = \frac{G_n \cdot P_n}{4 \cdot \pi} \cdot AF^2$$

The antenna array is assumed to be an ideal uniform array, meaning all the antennas have identical amplitudes in far-field and the distances between them are identical, as well. The antenna array is also assumed to be smaller than the beam sent from the transmitter. The power is then distributed evenly between them.

$$P_n = \frac{P_{total}}{N}$$

$$\frac{G_{total} \cdot P_{total}}{4 \cdot \pi} = \frac{G_n \cdot \frac{P_{total}}{N}}{4 \cdot \pi} \cdot AF^2$$

The equation simplifies to:

$$G_{total} = \frac{G_n}{N} \cdot AF^2 \quad [-] \tag{8.3}$$

Where G_{total} is the total linear gain, G_n is the linear gain from a single antenna, N is amount of elements in the array and AF is the array factor. Now the gain has been related to the array factor, but the array factor varies depending on the receiving angle. In this case, the interest lies in the angle with the most gain. To simplify Equation (8.3) further the array factor needs to be simplified. The array factor is also defined as:

$$AF = \frac{sin(\frac{N}{2}\Psi)}{sin(\frac{1}{2}\Psi)} \tag{8.4}$$

Where N is still the amount of elements in the array and Ψ is defined in the following equation:

$$\Psi = k \cdot d \cdot cos(\theta) + \beta \quad [-] \tag{8.5}$$

Where the wavenumber, $k = \frac{2 \cdot \pi}{\lambda}$, d is the distance between the antennas, θ is the elevation angle and β is the phase shift. Based on the requirements it is necessary to get a gain of 12.21 dBi. Combining equations (8.3), (8.4) and (8.5) gives the following equation:

$$G_{total} = \frac{G_n}{N} \cdot (\frac{sin(\frac{N}{2}(k \cdot d \cdot cos(\theta) + \beta))}{sin(\frac{1}{2} \cdot (k \cdot d \cdot cos(\theta) + \beta))})^2 \quad [-] \tag{8.6}$$

Some values can be inserted to reduce the equation. $k = \frac{2 \cdot \pi}{\lambda}$ is known. $d = \frac{\lambda}{2}$, since the distance between antennas has to be lower than λ, to ensure the beam one directional main lobe. $\beta = 0$, since the phase shift also depends on the type of radiation pattern, it will be set to 0, since the phase only affects the angle of gain, and not the magnitude.

$$G_{total} = \frac{G_n}{N} \cdot \left(\frac{sin(\frac{N}{2}(\frac{2 \cdot \pi}{\lambda} \cdot \frac{\lambda}{2} \cdot cos(\theta) + 0))}{sin(\frac{1}{2} \cdot (\frac{2 \cdot \pi}{\lambda} \cdot \frac{\lambda}{2} \cdot cos(\theta) + 0))}\right)^2$$

This can then be simplified to equation (8.7) by multiplying the different variables in the $sin()$ functions.

$$G_{total} = \frac{G_n}{N} \cdot \left(\frac{sin(\frac{N}{2} \cdot \pi \cdot cos(\theta))}{sin(\frac{1}{2} \cdot \pi \cdot cos(\theta))}\right)^2 \quad [-] \quad (8.7)$$

Focusing on the array factor part of the equation, when the angle, (θ) reaches 90° the equation equates to $\frac{0}{0}$

$$\frac{sin(\frac{N}{2} \cdot \pi \cdot cos(90))}{sin(\frac{1}{2} \cdot \pi \cdot cos(90))} = \frac{0}{0}$$

This means that there is an indeterminate form, which can be solved using L'Hospital's rule [104].

$$\lim_{\theta \to 90°} \frac{sin(\frac{N}{2} \cdot \pi \cdot cos(\theta))}{sin(\frac{1}{2} \cdot \pi \cdot cos(\theta))}$$

First the equation has to be differentiated with regards to θ:

$$\frac{d}{d\theta} \frac{sin(\frac{N}{2} \cdot \pi \cdot cos(\theta))}{sin(\frac{1}{2} \cdot \pi \cdot cos(\theta))} = \frac{cos(\frac{N}{2} \cdot \pi \cdot cos(\theta)) \cdot (-\frac{N}{2} \cdot \pi \cdot sin(\theta))}{cos(\frac{1}{2} \cdot \pi \cdot cos(\theta)) \cdot (-\frac{1}{2} \cdot \pi \cdot sin(\theta))}$$

Now, inserting for 90° for θ.

$$\frac{cos(\frac{N}{2} \cdot \pi \cdot cos(90)) \cdot (-\frac{N}{2} \cdot \pi \cdot sin(90))}{cos(\frac{1}{2} \cdot \pi \cdot cos(90)) \cdot (-\frac{1}{2} \cdot \pi \cdot sin(90))} = \frac{-\frac{N}{2} \cdot \pi}{-\frac{1}{2} \cdot \pi} = N$$

This can then be inserted into Equation (8.7), simplifying the equation to:

$$G_{total} = \frac{G_n}{N} \cdot N^2 = G_n \cdot N \quad [-] \quad (8.8)$$

This equation can then be used to calculate the amount of elements required in an array to achieve the correct gain.

3 Test of the S11-parameters for Antenna 0

3.1 Purpose of test

The purpose of this test is to determine the S11-parameters of Antenna 0. This test determines the resonant frequency, frequency range and the bandwidth of the Antenna. This test is made to ensure that the measured S11-parameters of the antennas match the simulations in CST.

3.2 Inventory and setup

An equipment list needed for the test can be seen in Table 8.1.

Inventory

qty	item	AAU identification number
1	PNA Network Analyzer N5227A	AAU 57016
1	1.85 mm Flexible test port cable	AAU 57029-01
1	Antenna 0	NA
1	Connector adapter	NA

Table 8.1: Inventory list for the S11-parameter test of Antenna 1.

Setup

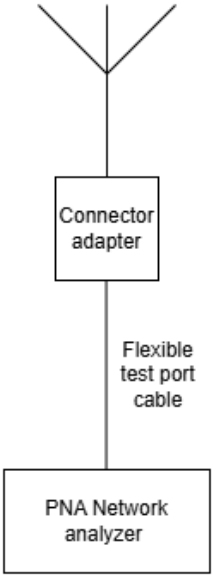


Figure 8.1: S11-parameters test setup.

The test setup for the measurement of S11-parameters is shown in Figure 8.1. For the S11-parameters test, the PNA Network Analyzer N5227A is used with a flexible test port cable. To be able to test the antenna, a connector adapter is used to connect the flexible test port cable to the antenna. The test is performed by the employees at APMS AAU.

3.3 Test results

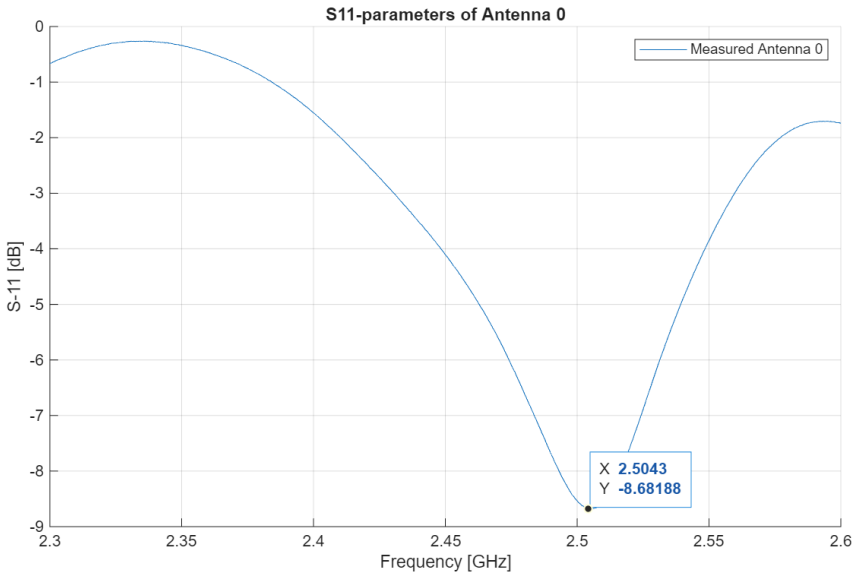


Figure 8.2: S11-parameters for Antenna 0. From 2.3 to 2.6 GHz.

The S11-parameters are seen in Figure 8.2. The resonant frequency is seen at the lowest S11-parameters, which is 8.68 dB at 2504.3 MHz. The frequency range corresponds to the frequencies below −10 dB. The bandwidth is the difference between the highest and lowest frequencies in the frequency range. The tested parameters are summed up in Table 8.2.

Parameter	Antenna 0
Frequency range	NA
Bandwidth	NA
Resonant frequency	2504.3 MHz

Table 8.2: Summary of the tested parameters for Antenna 0

3.4 Sources of error

To avoid reflection-induced distortion of the S11-parameters, the antennas must be kept clear of obstacles, and it should also be ensured that the coax cable is tightened probably. The test is however done by the employees at APMS, which minimizes possible errors.

4 Test of the Gain of Antenna 1

4.1 Purpose of test

The purpose of this test is to determine the gain of the antennas, and to ensure that the measured gain is similar to the simulated gain from CST. The test were performed in an anechoic to prevent reflections from surfaces, for example the wall or the ceilings.

4.2 Inventory and setup

An inventory list for the test can be seen in Table 8.3.

Inventory

qty	item	AAU identification number
1	Satimo Stargate 24 multi-probe system	NA
1	SMA-cable inherent to the anechoic chamber	NA
1	Antenna 1	NA

Table 8.3: Inventory list for the gain test of Antenna 1.

Setup

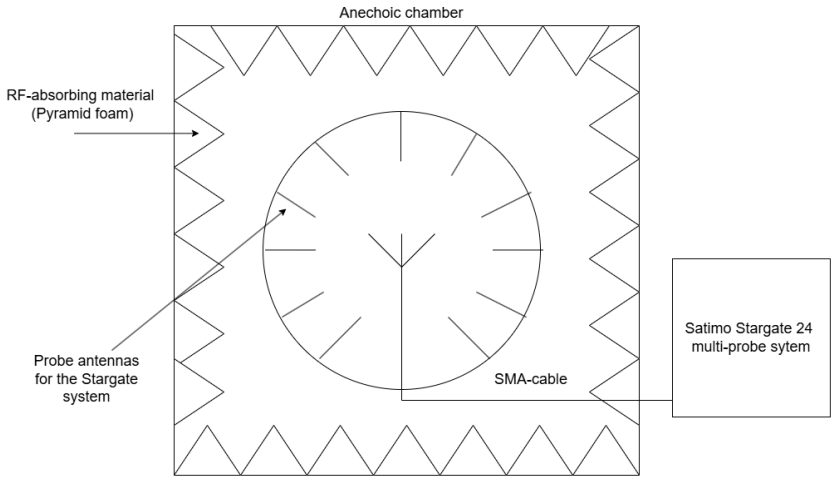


Figure 8.3: Test setup for the Gain test at APMS AAU.

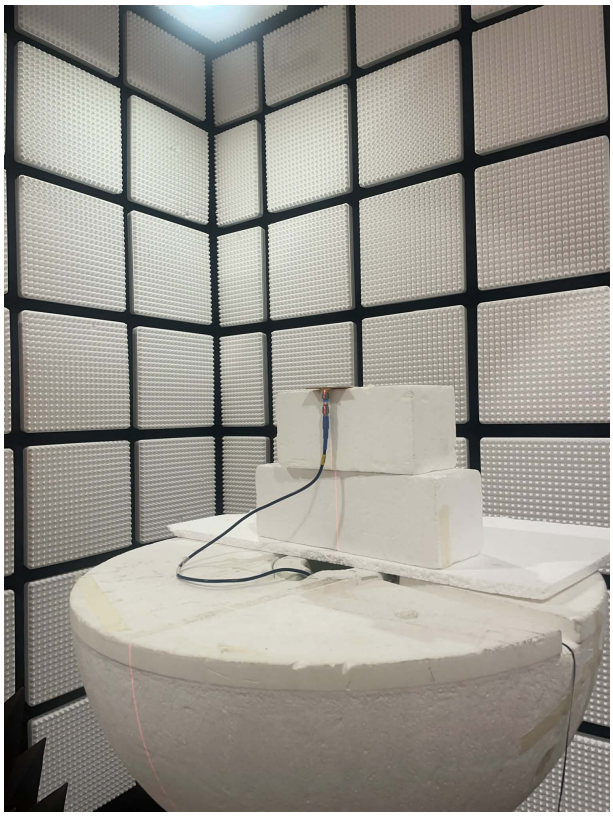


Figure 8.4: Picture of the setup of the tested antenna for the test of gain in the anechoic chamber.

The setup for the gain test is seen in Figure 8.3, showing how the antenna is positioned in the anechoic chamber. The setup and calibration of the Stargate 24 system is performed by the employees at APMS AAU.

4.3 Test results

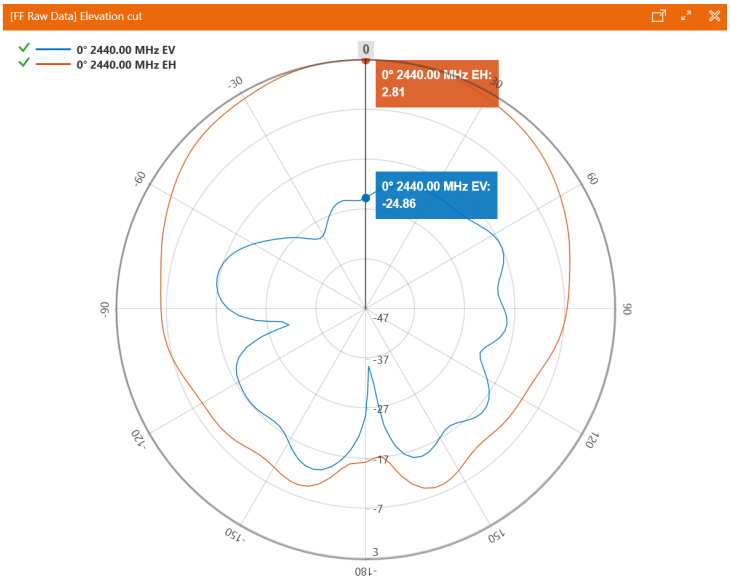


Figure 8.5: Test of the gain for Antenna 1 at 2440 MHz. EH is the strength of the electric field in the horizontal plane, and EV is the strength in the vertical plane. The unit of the values 2.81 and −24.86 is dBi.

The measured gain of Antenna 1 is shown in Figure 8.5, the maximum gain is in 0 °, where it is 2.81 dBi.

4.4 Sources of error

If the SMA cable is not tightened correctly or the test setup is not properly positioned. The measurement of gain would not correspond to what is actually possible by the tested antenna. The test is done by the employees at APMS, which minimizes possible errors.

5 Determining Transfer Function Parameters for Azimuth

5.1 Purpose of test

The purpose of this test journal is to determine the parameters required in section 4.4.2 to construct the azimuth transfer function. The determination of parameters is split into two categories - solo and combined. Where solo determines R, L, K_e and K_t while combined determines N, B_v and J.

5.2 Inventory and setup

Inventory solo

qty	item	AAU ID no.
1	HAMEG power supply HM7042-3	AAU 52752
1	Danica EA-PS 7032 power supply	AAU 77076
1	KEYSIGHT DSOX1102G	AAU 105615
1	FLUKE 37 MULTIMETER	AUC 08525
1	DR-2477 (torque meter)	AAU 105645
1	HIOKI FT3406 TACHO HiTESTER	AAU 124127
1	CP 35 CURRENT PROBE	
1	Azimuth motor	

Table 8.4: Equipment list for determining motor parameters.

Inventory combined:

qty	item	AAU ID no.
1	TARP	
1	CP 35 CURRENT PROBE	
1	Analog discovery 1 rev. C	

Table 8.5: Equipment list for combined TARP parameters.

Setup solo

The *Danica* PSU powers the motor, and the *HAMEG* PSU powers the torque meter (DR-2477). To measure the current, the CP 35 current probe is clamped around the positive wire to the motor. The output of the current probe is connected to port 2 on the oscilloscope. With port 1 either being connected to the motor voltage unless torque must be measured. Then it is connected to the output from the DR-2477. A diagram of the entire setup can seen been on figure 8.6.

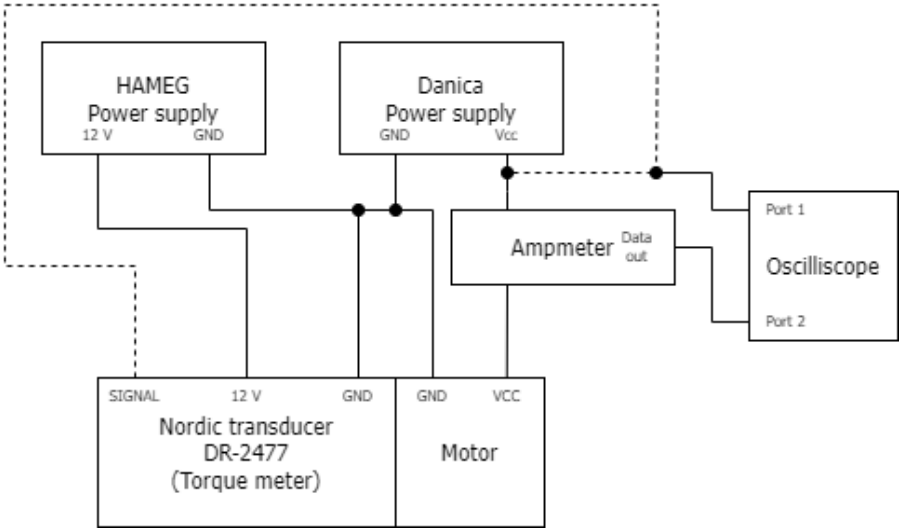


Figure 8.6: Diagram of motor parameter test setup.

Measurement procedure:

- Lock the motor so it can not physically turn
- The resistance (R)
  - Set the voltage generator to 3 V
  - At steady state, measure the voltage and current over/into to the motor
- The inductance (L)
  - Apply a step voltage to the motor of 3 V
  - Measure the voltage and current over/into to the motor
- Unlock the motor so it can spin
- Back EMF constant (K_e)
  - Set the voltage generator to 6 V
  - Let the motor run at a constant velocity
  - Measure the voltage and current over/into the motor
  - Measure the angular velocity of the motor
- Torque constant (K_t)
  - Mount the DR-2477
  - Apply a voltage of 6 V
  - Measure the output of the DR-2477
  - Measure the current into the motor

Setup combined
The current probe should be clamped on the TARP's +12 V wire. The analog discovery should be connected to the CP 35 current probe.
Measurement procedure:

- Count teeth on the motor gear and on the TARP gear.
- Measure idle current draw.
- To the motor apply PWM voltages of 255, 230 and 200.
- Measure the current and position.

5.3 Test results

Gear ratio - N

Finding the gear ratio, is a matter of inputting gear teeth amount (186 for azimuth gear and 10 for motor gear) in Equation 4.34 resulting in:

$$N_{azimuth} = \frac{186}{10} = 18.6$$

Resistance - R

The measured current is 3.99 A at voltage of 2.89 V. Using Equation 4.32 results in:

$$R = \frac{2.89\,\text{V}}{3.99\,\text{A}} \approx 724.31\,\text{m}\Omega$$

Inductance - L

The measurement can be seen on Figure 8.7. It can be seen, that the time constant is 2.6 ms. Equation 4.33 specifies the time constant to be L/R which is isolated for L results in:

$$L = R \cdot \tau = 724.31\,\text{m}\Omega \cdot 2.6\,\text{ms} \approx 1.88\,\text{mH}$$

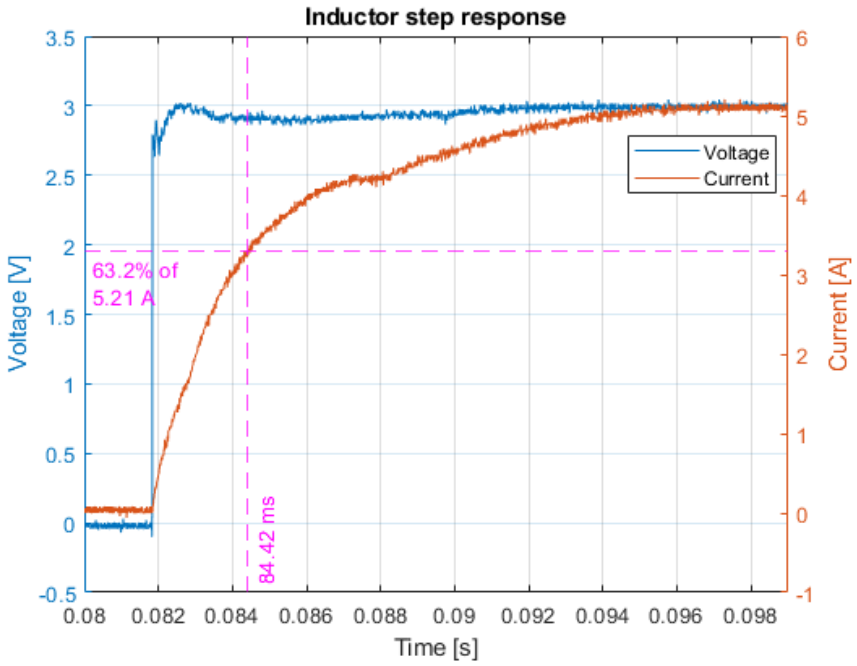


Figure 8.7: Data graph from the inductance test of the motor.

Back EMF constant - K_e

The measured voltage and current is 5.98 V and 2.30 A respectively, resulting in an angular velocity of 98.96 rad/s. Using Equation 4.35 K_e becomes:

$$K_{e-motor} = \frac{5.98\,\text{V} - 724.31\,\text{m}\Omega \cdot 2.30\,\text{A}}{\frac{98.96\,\text{rad/s}}{18.6}} \approx 810.90\,\frac{\text{mV}}{\text{rad/s}}$$

Torque constant - K_t

The measured torque output of the DR-2477 is 46.19 mNm and the current is measured to 2.62 A. Using Equation 4.36 results in:

$$K_t = \frac{46.19\,\text{mNm}}{2.62\,\text{A}} \cdot 18.6 \approx 17.63\,\frac{\text{mNm}}{\text{A}}$$

.

Viscous friction - B_v

Due to the inconsistent velocities visible on Figure 8.8 means have been used. Raw data is used, i.e. in encoder pulses (ep) and not in radians.

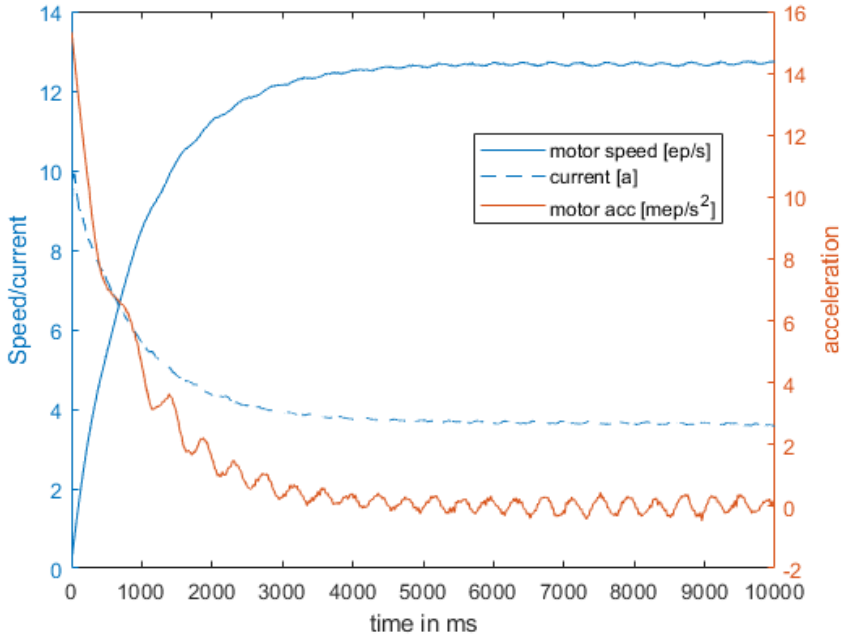


Figure 8.8: Unstable velocity of the TARP at constant voltage.

The measurements that have been made can be seen in Table 8.6. It should be noted that the inconsistent velocity gets more dominant at lower velocities, which is why the minimum PWM signal is set to 200. Average acceleration for each measurement close to zero.

PWM signal (0 **to** 255)	255	230	200
Time interval	$[5917, 8155]$ ms	$[7516, 7900]$ ms	$[5810, 6421]$ ms
Avg. velocity	15.97 $\frac{\text{ep}}{\text{ms}}$	12.67 $\frac{\text{ep}}{\text{ms}}$	8.24 $\frac{\text{ep}}{\text{ms}}$
Avg. current	5.56 A	4.42 A	3.35 A

Table 8.6: Measurements at close to zero acceleration.

Before Equation 4.37 can be used, the velocity needs to be converted from ep/ms to rad/s. From Section 4.4 it is known that the encoder outputs a combined 1000 pulses pr. encoder rotation and, due to the physical structure, has 5 encoder rotations pr. TARP rotation. It is now possible to convert the measurements using:

$$\frac{\text{rad}}{\text{ms}} = \frac{\text{ep}}{\text{ms}} \cdot \frac{2\pi}{5 \cdot 1000} \Rightarrow \frac{\text{rad}}{\text{s}} = \frac{\text{ep}}{\text{ms}} \cdot \frac{2\pi}{5 \cdot 1000} \cdot 1000 = \frac{\text{ep}}{\text{ms}} \cdot \frac{2\pi}{5} \tag{8.9}$$

Applying this on the measurements from Table 8.6 results in:

$$\text{Avg. velocity} = [15.97, 12.67, 8.24] \frac{\text{ep}}{\text{ms}} \cdot \frac{2\pi}{5} \approx [20.07, 15.92, 10.35] \frac{\text{rad}}{\text{s}}$$

A linear regression of Equation 4.37 (Where 281.17 mA has been subtracted from all currents, as this is the idle current draw) is now possible, resulting in a R^2 of 0.99 with the fitted

relationship:

$$K_t \cdot i(t) = 3.97 \cdot 10^{-3} \cdot \omega(t) + 11.99 \cdot 10^{-3}$$

Thereby B_v is estimated to 3.97 $\frac{\text{mNm·s}}{\text{rad}}$ and the dry friction is estimated to 11.99 mNm. The linear regression can be seen on Figure 8.9.

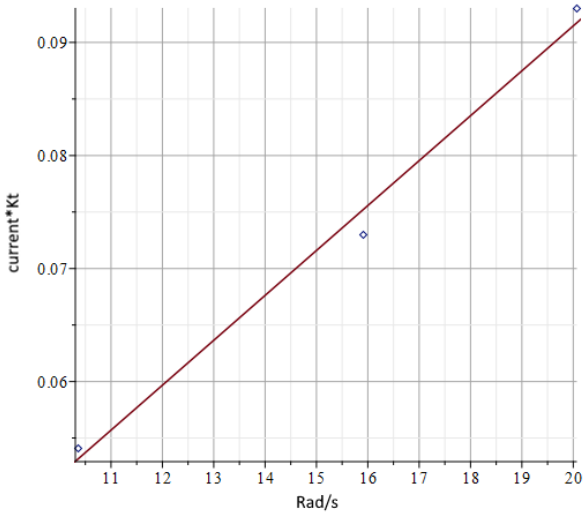


Figure 8.9: Illustration of the linear regression used to fit data from viscous friction test.

Inertia - J

Measurements for inertia is done in a similar way as to B_v, however this time with acceleration close to maximum. Measurements can be seen in Table 8.7.

PWM signal (0 to 255)	255	230	200
Time interval	$[50, 64]$ ms	$[37, 44]$ ms	$[136, 161]$ ms
Avg. velocity	1.13 $\frac{\text{ep}}{\text{ms}}$	0.72 $\frac{\text{ep}}{\text{ms}}$	1.47 $\frac{\text{ep}}{\text{ms}}$
Avg. acceleration	15.52 $\frac{\text{mep}}{\text{ms}^2}$	14.54 $\frac{\text{mep}}{\text{ms}^2}$	8.19 $\frac{\text{mep}}{\text{ms}^2}$
Avg. current	12.20 A	9.20 A	5.87 A

Table 8.7: Measurements at close to maximum acceleration. NB: the velocity of 200 is higher than that of 230, this is due to it having more time to accelerate.

The velocities must be converted to rad/s which is done using Equation 8.9:

$$\text{Avg. velocity} = [1.13, 0.72, 1.47]\frac{\text{ep}}{\text{ms}} \cdot \frac{2\pi}{5} \approx [1.42, 0.90, 1.85]\,\frac{\text{rad}}{\text{s}}$$

The acceleration likewise must be converted, which is done by modifying Equation 8.9 to:

$$\frac{\text{rad}}{\text{s}^2} = \frac{\text{ep}}{\text{ms}^2} \cdot \frac{2\pi}{5 \cdot 1000} \cdot 1000^2 = \frac{\text{ep}}{\text{ms}^2} \cdot \frac{2000\pi}{5} = \frac{\text{ep}}{\text{ms}^2} \cdot 400\pi$$

Which results in:

$$\text{Avg. acceleration} = [15.52, 14.54, 8.19]\frac{\text{ep}}{\text{ms}^2} \cdot 10^{-3} \cdot 400\pi \approx [19.50, 18.27, 10.29]\,\frac{\text{rad}}{\text{s}^2}$$

The idle current draw (281.17 mA) once again needs to be subtracted from the current. Using these values in equation 4.38 results in:

$$J = \frac{17.63\,\frac{\text{mNm}}{A} \cdot ([12.20, 9.20, 5.87]\,\text{A} - 281.17\,\text{mA}) - 3.97\,\frac{\text{mNm}\cdot\text{s}}{rad} \cdot [1.42, 0.90, 1.85]\,\frac{\text{rad}}{\text{s}} - 11.99\,\text{mNm}}{[19.50, 18.27, 10.29]\,\frac{\text{rad}}{\text{s}^2}}$$

$$\approx [9.87, 7.75, 7.70]\,\frac{\text{mNm}\cdot\text{rad}}{\text{s}^2}$$

As the last two of the values, are close to each other, it is assumed that some disturbance was inflicted on the first measurement. The inertia is therefore estimated to be the mean of the two last values (7.73 $\frac{\text{mNm}\cdot\text{rad}}{\text{s}^2}$)

5.4 Sources of error

Multiple sources have contributed errors. The most predominate is the inability to run the motor at a constant velocity due to unstable braking force seen on Figure 8.8.
Estimates have been used to estimate values, such as B_v which uses K_t.
Measurements of position in B_v and J where only made every 10 ms.

6 Determining Transfer Function Parameters for Tilt

6.1 Purpose of test

The purpose of this test journal is to determine the parameters required in Section 4.4.2 to construct the tilt transfer function. The determination of parameters is split into two categories - solo and combined. Where solo determines R, L, K_e and K_t while combined determines N, B_v and J.

6.2 Inventory and setup

Inventory solo

qty	item	AAU ID no.
1	HAMEG power supply HM7042-3	AAU 52752
1	Danica EA-PS 7032 power supply	AAU 77076
1	KEYSIGHT DSOX1102G	AAU 105615
1	FLUKE 37 MULTIMETER	AUC 08525
1	NORDIC TRANSDUCER	AAU 105645
1	HIOKI FT3406 TACHO HiTESTER	AAU 124127
1	CP 35 CURRENT PROBE	
1	Tilt motor	

Table 8.8: Equipment list for determining motor parameters.

Inventory combined:

qty	item	AAU ID no.
1	TARP	
1	CP 35 CURRENT PROBE	
1	Analog discovery 1 rev. C	

Table 8.9: Equipment list for combined TARP parameters.

Setup solo

The *Danica* PSU powers the motor, and the *HAMEG* PSU powers the torque meter (DR-2477). To measure the current, the CP 35 current probe is clamped around the positive wire to the motor. The output of the current probe is connected to port 2 on the oscilloscope. With port 1 either being connected to the motor voltage unless torque must be measured. Then it is connected to the output from the DR-2477. A diagram of the entire setup can seen been on Figure 8.10.

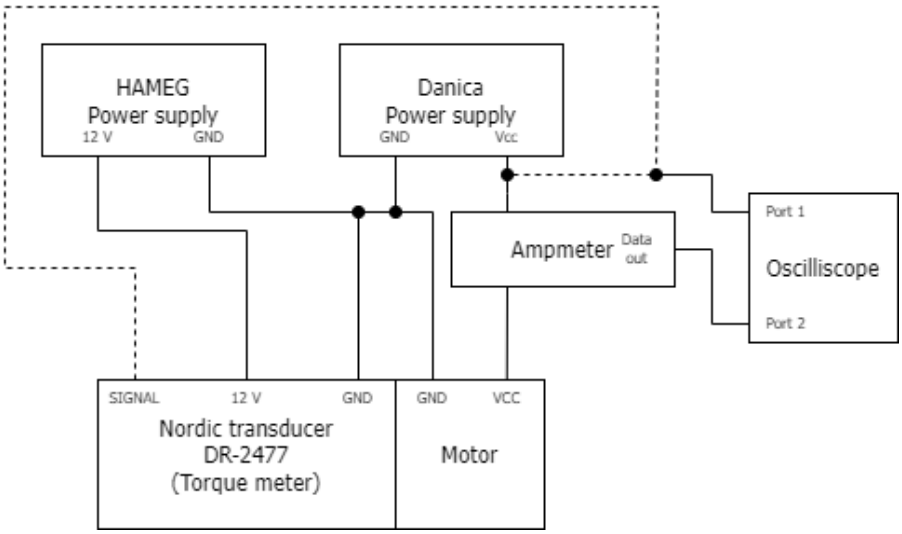


Figure 8.10: Diagram of motor parameter test setup.

How to measure:

- Lock the motor so it can not physically turn
- The resistance (R)
  - Set the voltage generator to 6 V
  - At steady state, measure the voltage and current over/into to the motor
- The inductance (L)
  - Apply a step voltage to the motor of 6 V
  - Measure the voltage and current over/into to the motor
- Unlock the motor so it can spin
- Back EMF constant (K_e)

  - Set the voltage generator to 6 V
  - Let the motor run at a constant velocity
  - Measure the voltage and current over/into the motor
  - Measure the angular velocity of the motor

- Torque constant (K_t)
  - Mount the DR-2477
  - Apply a voltage of 6 V
  - Measure the output of the DR-2477
  - Measure the current into the motor

Setup combined
The current probe should be clamped on the TARP's +12 V wire. The analog discovery should be connected to the CP 35 current probe.
Measurement procedure:

- Unmount the motor gearing and count motor rotations for one full axis rotation.
- Measure idle current draw.
- To the motor apply PWM voltages of 255 and 200.
- Measure the current and position.

6.3 Test results

Gear ratio - N

As the gear ratio is motor rotations pr. axis rotation (Equation 4.34), then it is directly known that the gear ratio must be:

$$N_{tilt} = \frac{24}{1} = 24$$

Resistance - R

The measured current is 893.35 mA at voltage of 6.01 V. Using Equation 4.32 results in:

$$R = \frac{6.01\,\text{V}}{893.35\,\text{mA}} \approx 6.73\,\Omega$$

Inductance - L

The measurement can be seen on Figure 8.11. It can be seen, that the time constant is 459.5 ms. Equation 4.33 specifies the time constant to be L/R which is isolated for L results in:

$$L = R \cdot \tau = 6.73\,\Omega \cdot 575.00\,\mu\text{s} \approx 3.87\,\text{mH}$$

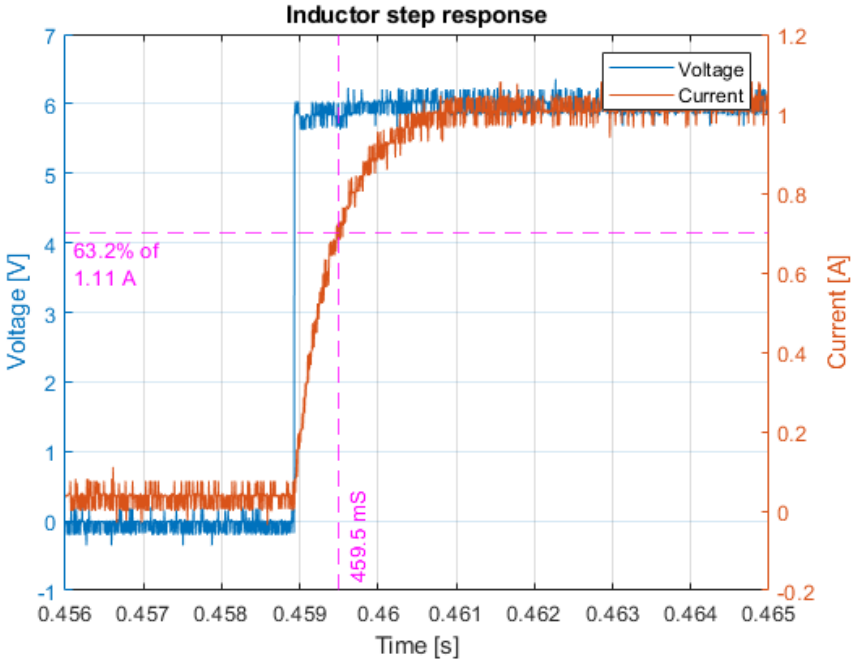


Figure 8.11: Data graph from the inductance test of the tilt motor.

Back EMF constant - K_e

The measured voltage and current is 6.02 V and 802.67 mA respectively, resulting in an angular velocity of 51.94 rad/s. Using Equation 4.35 K_e becomes:

$$K_{e-motor} = \frac{6.02\,\text{V} - 6.73\,\Omega \cdot 802.67\,\text{mA}}{\frac{51.94\,\text{rad/s}}{24}} \approx 285.57\,\frac{\text{mV}}{\text{rad/s}}$$

Torque constant - K_t

The measured torque output of the DR-2477 is 15.72 mNm and the current is measured to 931.76 mA. Using Equation 4.36 results in:

$$K_t = \frac{15.72\,\text{mNm}}{931.76\,\text{mA}} \cdot 24 \approx 404.91\,\frac{\text{mNm}}{\text{A}}$$

.

Viscous friction - B_v

Due to the inconsistent velocities visible on Figure 8.12 means have been used. Raw data is used, i.e. in encoder pulses (ep) and not in radians.

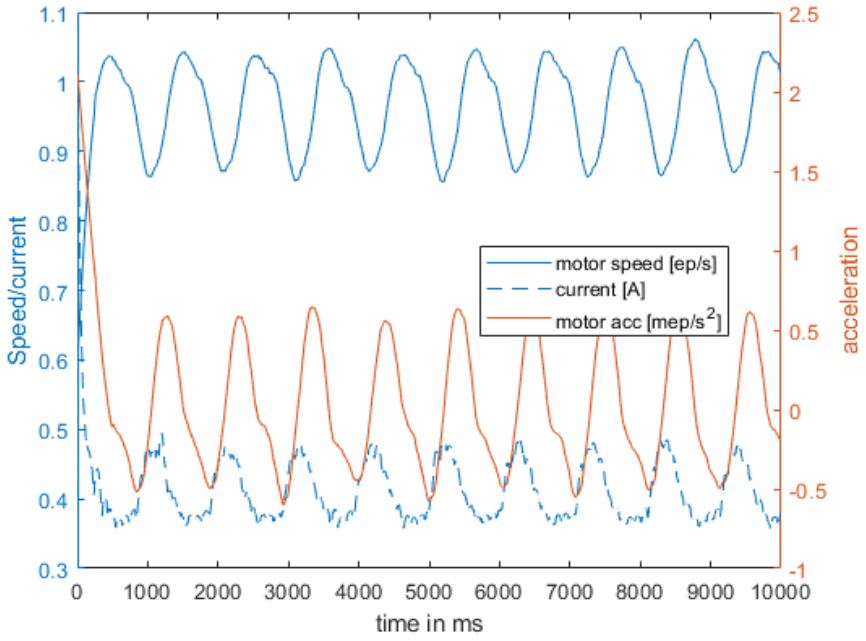


Figure 8.12: Unstable velocity of the TARP at constant voltage.

The measurements that have been made can be seen in Table 8.10. It should be noted that the inconsistent velocity gets more dominant with lower velocities, which is why the minimum PWM signal is set to 200. Average acceleration for each measurement close to zero.

PWM signal (0 **to** 255)	255	200
Time interval	$[2208, 3034]$ ms	$[7073, 8115]$ ms
Avg. velocity	$1.21 \frac{\text{ep}}{\text{ms}}$	$0.96 \frac{\text{ep}}{\text{ms}}$
Avg. current	0.43 A	0.41 A

Table 8.10: Measurements at close to zero acceleration.

Before Equation 4.37 can be used, the velocity needs to be converted from ep/ms to rad/s. From Section 4.4 it is known that the encoder outputs a combined 1000 pulses pr. encoder rotation, and due to the physical structure, has 1 encoder rotations pr. TARP rotation. It is now possible to convert the measurements using:

$$\frac{\text{rad}}{\text{ms}} = \frac{\text{ep}}{\text{ms}} \cdot \frac{2\pi}{1000} \Rightarrow \frac{\text{rad}}{\text{s}} = \frac{\text{ep}}{\text{ms}} \cdot \frac{2\pi}{1000} \cdot 1000 = \frac{\text{ep}}{\text{ms}} \cdot 2\pi \tag{8.10}$$

Applying this on the measurements from Table 8.10 results in:

$$\text{Avg. velocity} = [1.21, 0.96] \frac{\text{ep}}{\text{ms}} \cdot 2\pi \approx [7.60, 6.03] \frac{\text{rad}}{\text{s}}$$

A linear regression of Equation 4.37 (Where 277.33 mA has been subtracted from all currents, as this is the idle current draw) is now possible, resulting in a R^2 of 1 with the fitted relationship :

$$K_t \cdot i(t) = 5.16 \cdot 10^{-3} \cdot \omega(t) + 22.62 \cdot 10^{-3}$$

Thereby B_v is estimated to $5.16 \frac{\text{nNm·s}}{\text{rad}}$ and the dry friction is estimated to 22.62 mNm. The linear regression can be seen on Figure 8.13.

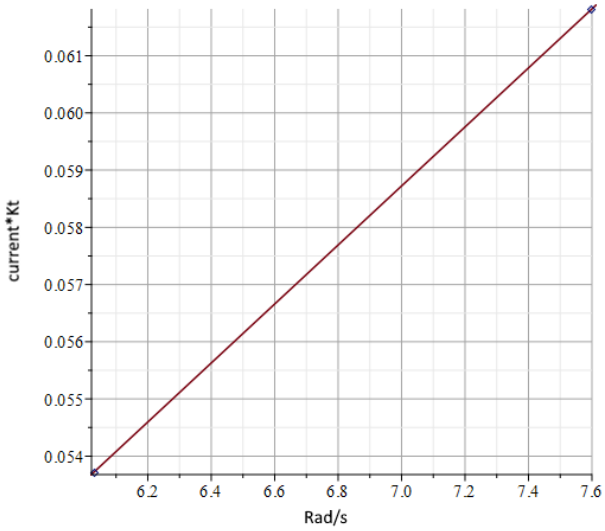


Figure 8.13: Illustration of the linear regression used to fit data from viscous friction test.

Inertia - J

Measurements for inertia is done in a similar way as to B_v, however this time with acceleration close to maximum. Measurements can be seen in Table 8.11.

PWM signal (0 to 255)	255	200
Time interval	$[10, 20]$ ms	$[11, 21]$ ms
Avg. velocity	$0.86\ \frac{\text{ep}}{\text{ms}}$	$0.63\ \frac{\text{ep}}{\text{ms}}$
Avg. acceleration	$3.32\ \frac{\text{mep}}{\text{ms}^2}$	$2.03\ \frac{\text{mep}}{\text{ms}^2}$
Avg. current	1.38 A	1.02 A

Table 8.11: Measurements at close to maximum acceleration.

The velocities must be converted to rad/s which is done using Equation 8.10:

$$\text{Avg. velocity} = [0.86, 0.63]\frac{\text{ep}}{\text{ms}} \cdot 2\pi \approx [5.40, 3.96]\ \frac{\text{rad}}{\text{s}}$$

The acceleration likewise must be converted, which is done by modifying Equation 8.10 to:

$$\frac{\text{rad}}{\text{s}^2} = \frac{\text{ep}}{\text{ms}^2} \cdot \frac{2\pi}{1000} \cdot 1000^2 = \frac{\text{ep}}{\text{ms}^2} \cdot 2000\pi$$

Which results in:

$$\text{Avg. acceleration} = [3.32, 2.03]\frac{\text{ep}}{\text{ms}^2} \cdot 10^{-3} \cdot 2000\pi \approx [20.86, 12.75]\ \frac{\text{rad}}{\text{s}^2}$$

The idle current draw (277.33 mA) once again needs to be subtracted from the current. Using these values in Equation 4.38 results in:

$$J = \frac{404.91\ \frac{\text{mNm}}{A} \cdot ([1.38, 1.02]\,\text{A} - 277.33\,\text{mA}) - 404.91\ \frac{\text{mNm}\cdot\text{s}}{rad} \cdot [5.40, 3.96]\ \frac{\text{rad}}{\text{s}} - 22.62\,\text{mNm}}{[20.82, 12.75]\ \frac{\text{rad}}{\text{s}^2}}$$

$$\approx [15.44, 15.43]\ \frac{\text{mNm}\cdot\text{rad}}{\text{s}^2}$$

The inertia is therefore estimated to be the mean of the two values ($15.44\ \frac{\text{mNm}\cdot\text{rad}}{\text{s}^2}$)

6.4 Sources of error

Multiple sources have contributed errors. The most predominate is the inability to run the motor at a constant velocity due to unstable braking force seen on Figure 8.8.
Estimates have been used to estimate values, such as B_v which uses K_t.
Measurements of position in B_v and I where only made every 10 ms.
The R^2 value should be interpreted with caution, as only two values have been used.

7 TARP Control Test

7.1 Purpose of test

The purpose of this test is to document the step response of the p-controller of the physical system. The TARP receives a target angle and moves the azimuth or tilt motor, depending on the test. The position is saved as it moves.

7.2 Inventory and setup

Inventory

qty	item	AAU identification number
1	TARP	

Table 8.12

Setup
Test setup for step response of azimuth P-controller:

1. Upload the code to TARP, see Appendix 23.
2. Plug in TARP to 230 V outlet.
3. Connect your PC to "TARP" wifi-hotspot.
4. Open PuTTY and use the following settings: IP address: 192.168.4.1 Port: 1234 Connection type: Other - Raw Then press "Open", which opens a new window.
5. In this window, type in "0;120". This sets the target angle for the azimut motor to 120°.
6. Save the data that pops on the screen.

Test setup for step response of tilt P-controller:

1. Change TARP code in line 128 and 129 of Primary Code 23 from "angleAzi" to angleTilt and aziInDegrees to tiltInDegrees.
2. Upload the code to TARP, see Appendix 23.
3. Plug in TARP to 230 V outlet.
4. Connect your PC to "TARP" wifi-hotspot.

5. Open PuTTY and use the following settings: IP address: 192.168.4.1 Port: 1234 Connection type: Other - Raw Then press "Open", which opens a new window.
6. In this window, type in "1;45". This sets the target angle for the tilt motor to 45°.
7. Save the data that pops on the screen.

7.3 Test results

Figure 8.14 shows the saved positions resulting the step response of the azimuth and tilt motor. The oscillations for physical version of the azimuth oscillates $\pm 0.228°$ within 3.2 s. The oscillations for physical version of the azimuth oscillates $\pm 1.5°$ within 0.5 s.

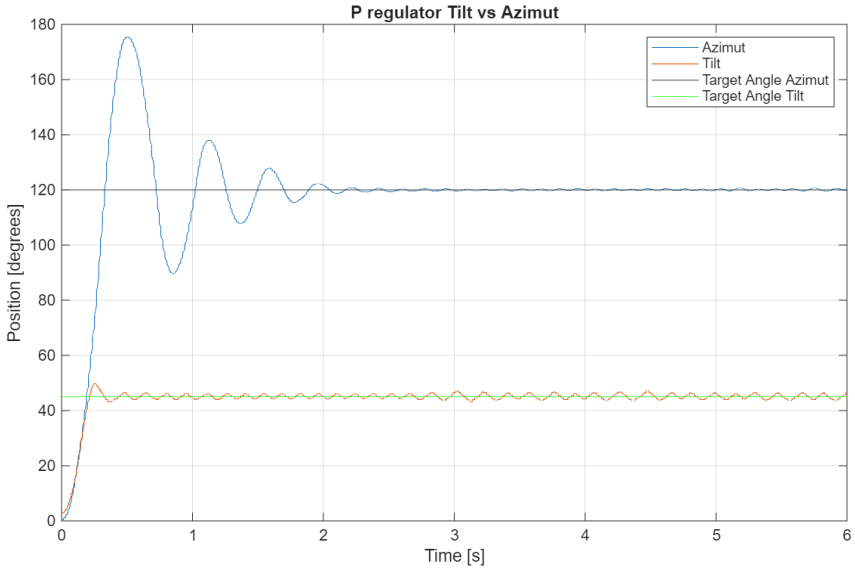


Figure 8.14: Azimut and tilt P-controllers step response plot.

7.4 Sources of error

The gearing for the tilt motor is observed to have a backlash of 3°.

8 Validation of RF Frontend

8.1 Purpose of test

The purpose of this test is to validate that the RF front end works as intended. This is done by testing the sample rate, the down conversion, pass band and attenuation. This is done over three different measurements, one with a signal that has a frequency within the pass band, one where the signals frequency is outside the pass band and one where no signal is input.

8.2 Inventory and setup

Inventory

qty	item	AAU identification number
1	BladeRF A9	AAU 102715
1	Signal generator	
1	Raspberry Pi	
1	BNC to SMA adapter	

Table 8.13: Equipment list for validation test of RF frontend.

Setup

To setup the test, the signal generator is coupled directly to the bladeRF A9 Rx ports, Rx1 and Rx2. This is done using the shortest possible cables. The signal generator is set to an amplitude of 400 $\mathrm{mV_{pp}}$ and a frequency of 2.442 GHz. The python script is edited so that 4096 samples is taken on each channel. After this a measurement is done with a frequency of 2500 MHz, then one with no signal. It should be noted that the measurements are first made on channel one and then on channel two, to avoid the use of a power splitter. The setup can be seen on Figure 8.15.

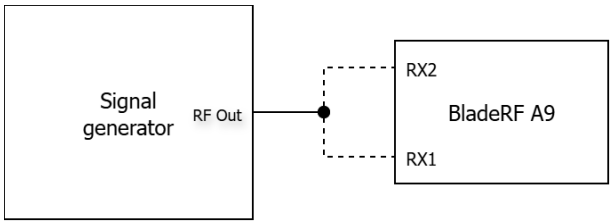


Figure 8.15: Test setup for validation test of RF frontend,

8.3 Test results

To convert the the unit less normalized values to voltage can Equation 8.11 be used. Where x is the normalized unitless values and U is the data in volt. These are multiplied with 0.625 $\mathrm{V_p}$, as the maximum input of the ADC is 1.25 $\mathrm{V_{pp}}$.

$$U = x \cdot 0.625\,\mathrm{V} \qquad [\mathrm{V}] \quad (8.11)$$

To make the time axis are an array containing the whole numbers from 0 to 8192 made in matlab, each number is then divided by the sampling rate (61.44 MHz). To get the time each sample is taken.

Figure 8.16 shows the IQ samples taken by the bladeRF A9 on both channels, without any signal being transmitted. The maximum and minimum values for the both the I and Q samples on both channels can be seen on Table 8.14.

	I amplitude	Q amplitude
Channel 1	$[-4.58\,;\,7.63]$ mV	$[-10.38\,;\,6.71]$ mV
Channel 2	$[-6.10\,;\,12.82]$ mV	$[-5.8\,;\,12.82]$ mV

Table 8.14: IQ samples taken by the blade RF with no input.

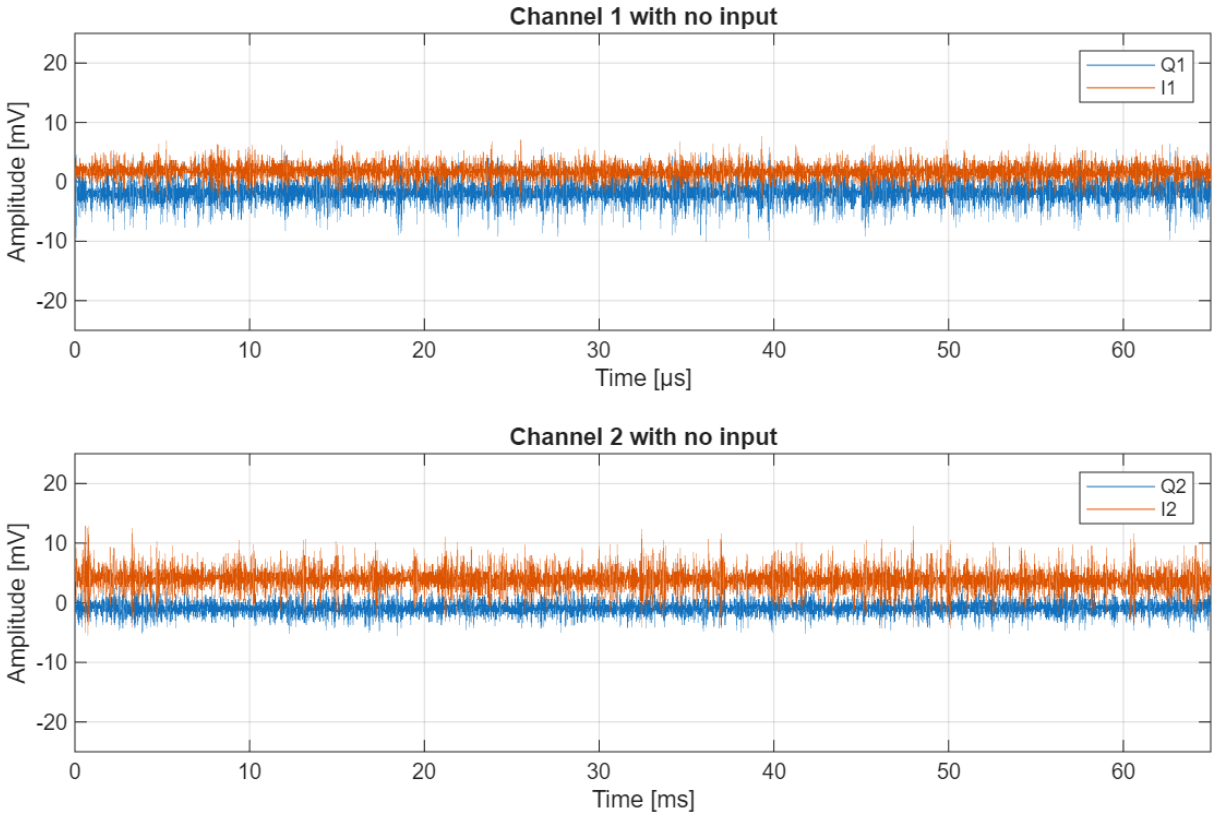


Figure 8.16: Measured noise from the RF front end validation test.

Figure 8.17 shows the IQ samples taken by the bladeRF A9 on both channels, with a signal being transmitted at a frequency of 2500 MHz. The maximum and minimum values for the both the I and Q samples on both channels can be seen on Table 8.15.

	I amplitude	**Q amplitude**
Channel 1	$[-4.88\,;\,8.85]$ mV	$[-10.99\,;\,7.63]$ mV
Channel 2	$[-5.19\,;\,12.51]$ mV	$[-5.49\,;\,3.97]$ mV

Table 8.15: IQ samples taken by the blade RF with signal being transmitted at a frequency of 2500 MHz.

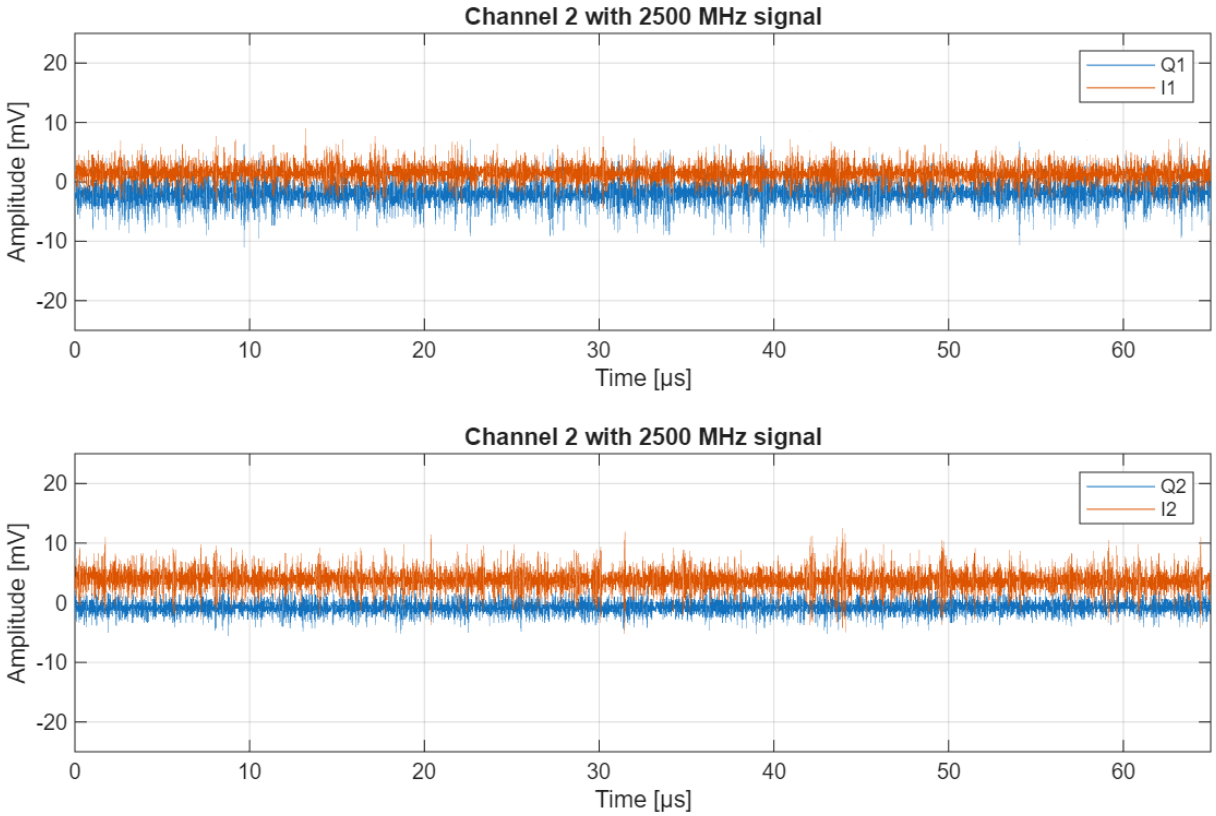


Figure 8.17: Measured data from the RF front end validation test with a 2500 MHz signal.

Figure 8.18 shows the IQ samples taken by the bladeRF A9 on both channels, with a signal being transmitted at a frequency of 2442 MHz. The maximum and minimum values for the both the I and Q samples on both channels can be seen on Table 8.16.

	I amplitude	**Q amplitude**
Channel 1	$[-343.63\,;\,347.6]$ mV	$[-348.51\,;\,343.93]$ mV
Channel 2	$[-48.52\,;\,55.85]$ mV	$[-53.1\,;\,52.8]$ mV

Table 8.16: IQ samples taken by the blade RF with signal being transmitted at a frequency of 2442 MHz.

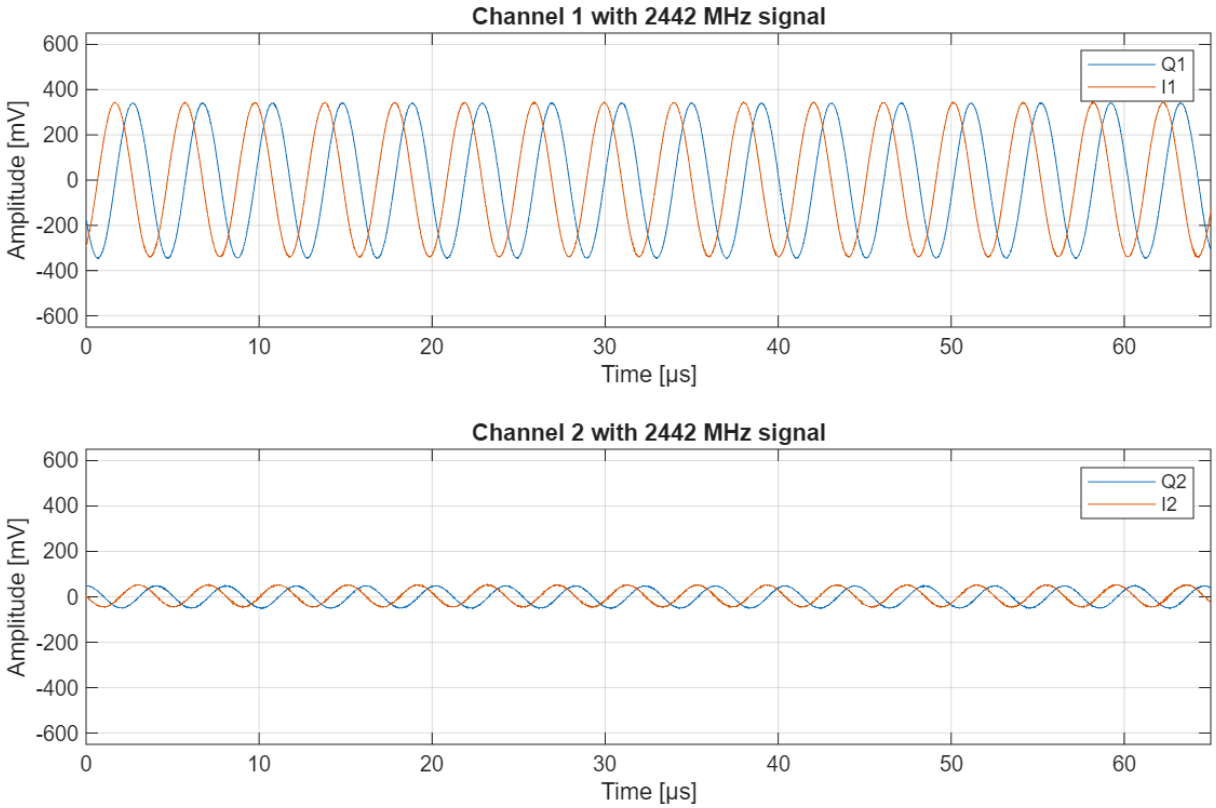


Figure 8.18: Measured data from the RF frontend validation test with a 2442 MHz signal.

8.4 Sources of error

The main source of error is that an adapter from BNC to SMA was required. The adapters that where used consists of three different parts, this may introduce unwanted distortions in the signal.

9 DAS Algorithm Test Using Phase Delay

9.1 Purpose of test

This appendix tests the DAS beamforming using known signal sources, thereby verifying if the beamforming algorithm combined with the SDR works as intended.

9.2 Inventory and setup

Inventory

qty	item	AAU identification number
1	Pi and BladeRF from prototype	
1	Rohde and Schwarz signal generator SMIQ 06B	AAU-52765
1	Rohde and Schwarz signal generator SME 06	AAU-102716
1	Tektronix AFG3252 function generator	AAU-56909
1	Tektronix TDS7704B Digital Phosphor Oscilloscope	AAU-102880
1	BNC T-piece	
2	BNC to BNC cable	
2	N-connector to SMA	

Table 8.17: Inventory required for testing the DAS algorithm using phase delays.

Setup
Use the shortest possible BNC cables and the T-piece to connect the function generator to both signal generators "Ref"-port. Configure the function generator with a square wave of 10 MHz, 1 V_{pp}. Enable each signal generator at 200 mV with a frequency of 2.442 GHz and a phase shift of 0 °. These should be connected to the oscilloscope, and configured, so that the output voltage of each is about the same, the phase shift should also be adjusted, to ensure they are synchronous. After this they should be connected directly to the BladeRF of the prototype, bypassing the antennas. This configuration can be seen on figure 8.19

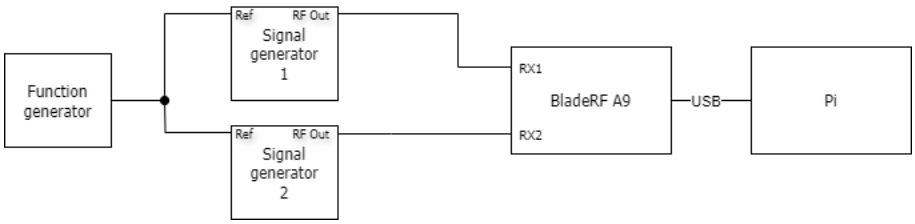


Figure 8.19: An illustration of the setup. It should be noted, that initially instead of the BladeRF, it should be connected to the oscilloscope.

The code on the Pi is the beamforming from Figure 4.47, the RF frontend code from Appendix 16, these are controlled by the code in Appendix 18. This code should be executed with different phases, these being $0, \pm[15, 30, 45, 90, 125, 135, 150, 180]$. It should be noted that the distance between the two antennas is set to $2/\lambda$ as stated in section 4.3.

9.3 Test results

To interpretate the test results, an equation converting from phase delay to an angle is required. Equation 4.14 is a good starting point, as this relates time difference to an angle. To convert a phase delay (ϕ) to a time delay (Δt), the period of the signal (T) must be divided 360 then multiplied with the phase delay in degrees. This can be seen in equation 8.12.

$$\Delta t = \frac{T}{360} \cdot \phi \tag{8.12}$$

Combining equation 4.14 and 8.12, with the angle of arrival (θ) isolated, results in equation

8.13:

$$\frac{T}{360} \cdot \phi = \frac{d_m \cdot sin(\theta)}{c}$$

$$\theta = sin^{-1}(\frac{T \cdot \phi \cdot c}{360 \cdot d_m}) \tag{8.13}$$

Filling in the blanks it is known that $T = f^{-1} = 2.442\,\text{GHz}^{-1} = 2.442\,\text{ns}$, $c = 3 \cdot 10^8\,\text{m/s}$ and that $d_m = \lambda/2 = c/f/2 = 61.43\,\text{mm}$ [88].
The first step is to sync the two signal generators, this required AAU-52765 to be set to 183 mV with no phase shift, and the AAU-102716 set to 200 mV with a phase shift of 180 °. The results of the test can be seen in Table 8.18, with "calculated AoA"-column using equation 8.13.

Degree	Calculated AoA	Measured mean AoA	Deviation from measured mean
180	90	52.16	88.74
150	56.44269020	55.12	$90.09 \cdot 10^{-3}$
135	48.59	47.84	$180.18 \cdot 10^{-3}$
125	43.98	42.72	$360.36 \cdot 10^{-3}$
90	30	29.96	$90.09 \cdot 10^{-3}$
45	14.48	13.24	0.00
30	9.59	8.59	$180.18 \cdot 10^{-3}$
15	4.78	3.98	$90.09 \cdot 10^{-3}$
0	0	$450.45 \cdot 10^{-3}$	0.00
−15	−4.78	−3.95	$90.09 \cdot 10^{-3}$
−30	−9.59	−9.86	$90.09 \cdot 10^{-3}$
−45	−14.48	−14.50	$180.18 \cdot 10^{-3}$
−90	−30	−29.93	$270.27 \cdot 10^{-3}$
−125	−43.98	−44.49	$90.09 \cdot 10^{-3}$
−135	−48.59	−48.99	$90.09 \cdot 10^{-3}$
−150	−56.44	−56.16	$540.54 \cdot 10^{-3}$
−180	−90	86.43	1.71

Table 8.18: Results of phase delay beamforming test. *Degree* is the phase delay, *calculated AoA* is the calculated angle of arrival, *measured mean AoA* is the mean of the 5 measurements and *deviation* is the distance from mean to max/min observed value.

9.4 Sources of error

It would have been better to synchronize the signal generators when connected to the bladeRF, as small changes might occur when switching from scope to SDR. A continuation of this, is that a single signal generator with internal phase synchronization would have been more optimal to use.

10 DAS Algorithm Test with Antennas

10.1 Purpose of test

This is a test of the DAS beamforming algorithm with connected antennas. It thereby verifies that the beamforming algorithm works as intended together with antennas radiation pattern.

10.2 Inventory and setup

Inventory

qty	item	AAU identification number
1	MVG SG 24 "stargate"	
1	Prototype	

Table 8.19: Equipment required for testing DAS algorithm with antennas.

Setup
Place the prototype in the center of the stargate. Enable a 8 dB, 2.442 GHz signal, at different positions, and log the DAS algorithm estimated direction of arrival. The different positions being $0, \pm[15, 30, 45]$. The code running on the prototype, can be seen in Appendix 18 with the imported submodules available in Appendix 16 (SDR front end) and Figure 4.47 (Beamforming).

10.3 Test results

As the degree which the stargate is set to, is equal to the direction of arrival no processing is required. The test results can be seen in table 8.20.

Degree	Measured mean	Deviation from measured mean
45	-25.72	$409.50 \cdot 10^{-3}$
30	70.13	$671.58 \cdot 10^{-3}$
15	24.89	$294.84 \cdot 10^{-3}$
0	-1.01	$196.56 \cdot 10^{-3}$
-15	-31.26	$180.18 \cdot 10^{-3}$
-30	55.13	$638.82 \cdot 10^{-3}$
-45	13.64	$573.30 \cdot 10^{-3}$

Table 8.20: Test results from DAS algorithm with antennas. *Degree* is the direction of arrival, *measured mean* is the mean of all the measurements and *deviation* is the distance from mean to max/min observed value.

10.4 Sources of error

Placement of the prototype might not have been exactly centered.

11 Detection Time Prototype

11.1 Purpose of test

The goal of this test is to validate that the prototype can detect a drone with in 43.47 s as set forward in Section 3.3.

11.2 Inventory and setup

Inventory

qty	item	AAU identification number
1	Prototype	
1	Power supply	

Table 8.21: Equipment needed for detection time test.

Setup
In this test is the python script modified so prototype moves to 120° and waits for the the prototype to point in a stable direction ($\pm 0.288°$) which happen after 3.2 s. The beamforming algorithm is then executed. The time for this to be executed is timed enternally in the program, this time is multiplied by three to get the time for detecting the full 360°. The modified python script can be seen in Appendix 26.It should be noted that this test is made five times.

11.3 Test results

The time for each run can be seen in the list under here.

- 5.12 s
- 5.03 s
- 5.08 s
- 5.05 s
- 5.02 s

This gives an average time of 5.06 s, which is multiplied by three to give the full rotation time.

$$5.06\,\text{s} \cdot 3 = 15.18\,\text{s}$$

12 Tracking Test Prototype

12.1 Purpose of test

The purpose of this test is to test the tracking capabilities of the prototype along side this is the precession of the prototype also tested.

12.2 Inventory and setup

Inventory

qty	item	AAU identification number
1	Prototype	
1	Signal generator	To be determined
1	Dipole antenna	To be determined

Table 8.22: Equipment list for the tracking test of the prototype.

Setup
The signal generator is set to output a signal with a frequency of 2450 MHz, this frequency is

chosen as it is within the frequency band the prototype is able to sample. The amplitude of the signal is not known yet as the dipole antenna is not chosen yet, but the output strength should be identical to the drones signal strength at 3 km. This test is consists of a repeating many steps:

1. Mark boresight of the antennas, and 15° step in both direction until 90°
2. Start the signal generator while the dipole antenna is at 0°
3. Move to the first spot on the positive degrees and wait for the TARP to get a stable location
4. Then repeat step 3, until 90° is reached.
5. Step 3 is repeated until $-90°$ is reached and back to 0° again.

This will ensure that the whole the prototype can track a drone moving though free space. A diagram of the test setup can be seen on Figure 8.20.

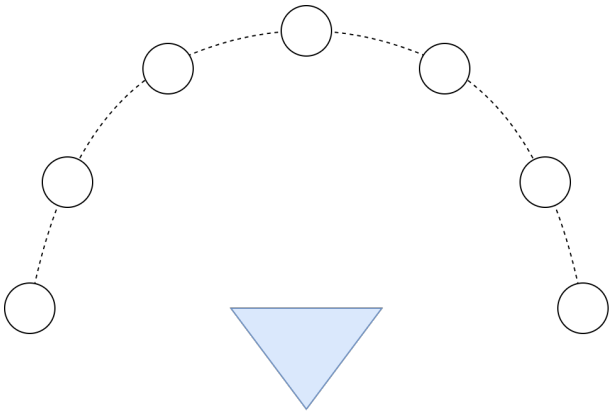


Figure 8.20: Diagram of physical test setup for tracking test. The circles are points with known angles to boresight and the upside down triangle is the prototype.

13 Tracking Speed

13.1 Purpose of test

The goal of this test is to ensure that the prototype can track the use case drone, moving at is maximum speed at a range of 2 km, in real time. This has to be tested in a place with minimal WIFI signal as this can create disturbances.

13.2 Inventory and setup

Inventory

qty	item	AAU identification number
1	Use case drone	
1	Prototype	

Table 8.23: Equipement needed for tracking time test.

Setup

First step in the setup is to setup the prototype in a location with minimal WIFI signals to ensure that interference is limited. Then the drone must be made to fly around the prototype in a circle with a radius of 2 km at maximum speed. The code that controls the prototype must be modified to save the beamforming outputs and the encoder data must also be saved. This is then lated analyzed to ensure the prototype can track the drone in real time.

14 TARP Motor Control Schematics

14.1 Main Schematic

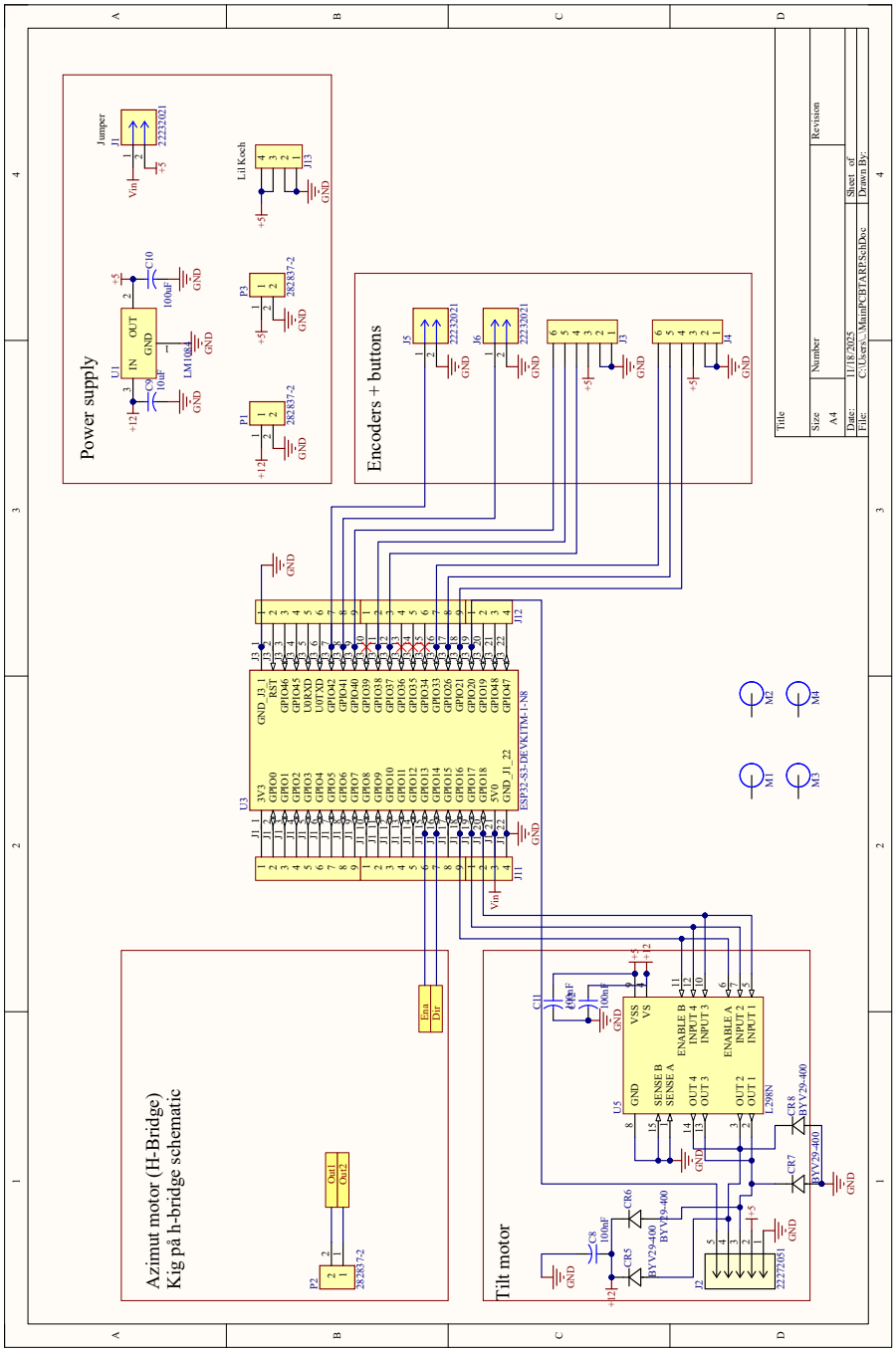


Figure 8.21: Tarp Motor Control main schematic containing buttons, encoders, power supply and ESP32.

14.2 H-bridge Schematic

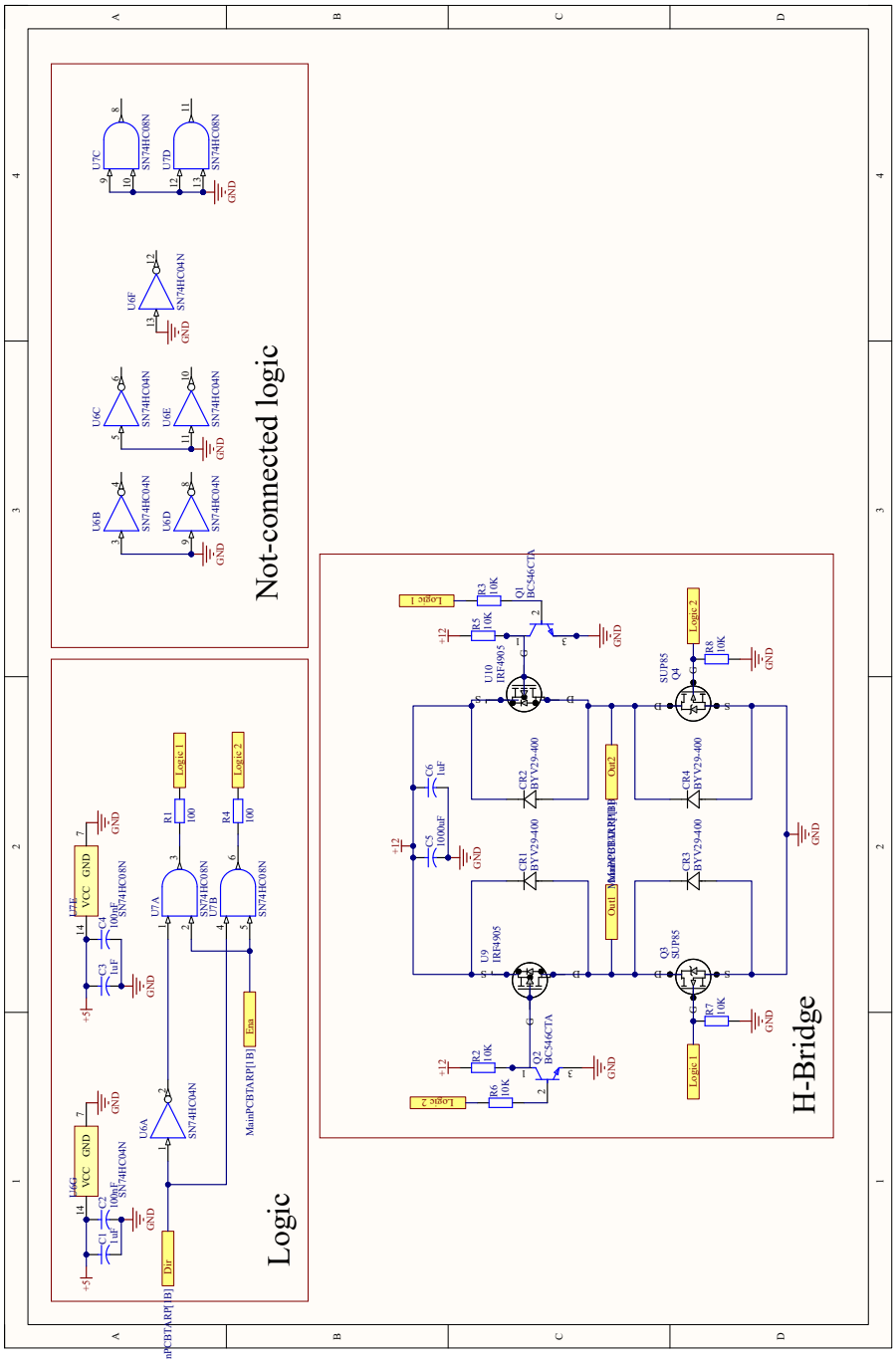


Figure 8.22: TARP H-bridge schematic

15 TARP Motor Control PCB

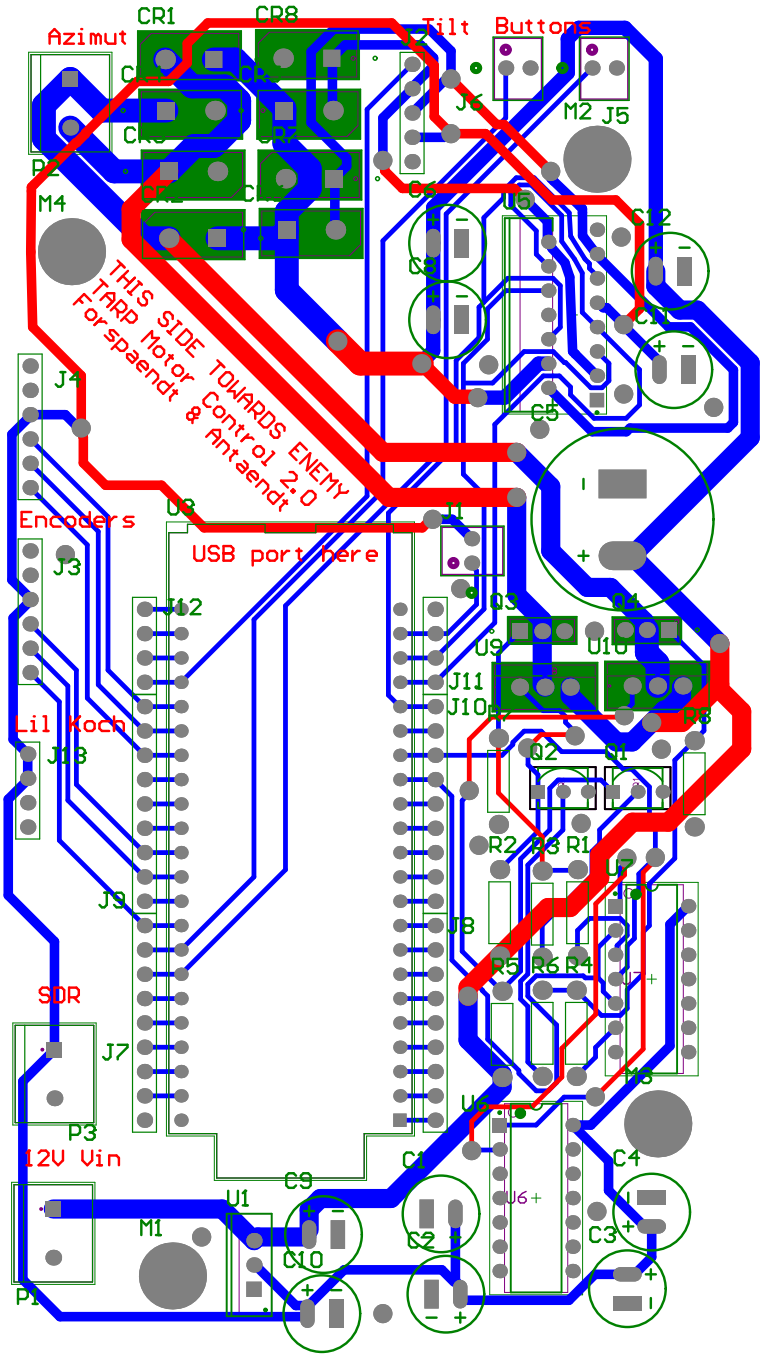


Figure 8.23: Tarp Motor Control PCB

16 Front End Python Code

```
import numpy as np
from bladerf import _bladerf

class sdr_ctrl:
    def __init__(self, sample_rate: int, center_freq: int):
```

```
        self.sdr = _bladerf.BladeRF()
        self.sample_rate = sample_rate
        self.center_freq = center_freq

        self.buffer_size = 8192

        # --- Setup RX1 ---
        self.rx1 = self.sdr.Channel(_bladerf.CHANNEL_RX(0))
        self.rx1.frequency = center_freq
        self.rx1.sample_rate = sample_rate
        self.rx1.bandwidth = sample_rate
        self.rx1.gain_mode = _bladerf.GainMode.Manual
        self.rx1.gain = 0

        # --- Setup RX2 ---
        self.rx2 = self.sdr.Channel(_bladerf.CHANNEL_RX(1))
        self.rx2.frequency = center_freq
        self.rx2.sample_rate = sample_rate
        self.rx2.bandwidth = sample_rate
        self.rx2.gain_mode = _bladerf.GainMode.Manual
        self.rx2.gain = 20

        # --- Configure one synchronous 2 RX stream ---
        self.sdr.sync_config(
            layout=_bladerf.ChannelLayout.RX_X2,  # Layout is 2 RX
            fmt=_bladerf.Format.SC16_Q11,  # Data format is Signed Complex 16bit
            # Q11 = fixed-point scaling. The 16 bits are split into 1 sign bit, 4 integer, and 11
            ↪  decimals.
            num_buffers=16,  # buffer amount
            buffer_size=self.buffer_size,  # Size of buffer
            num_transfers=8,  # Amount of transferes active.
            stream_timeout=3500,  # Timeout, after this time, python will crash
        )

        self.bytes_per_sample = 4  # SC16_Q11: 2 x int16 = 4 bytes
        self.buf = bytearray(self.buffer_size * self.bytes_per_sample *2)  # *2 due to two channels (Rx1 +
        ↪  Rx2)

        # Enable both channels
        self.rx1.enable = True
        self.rx2.enable = True

    def sample(self, num_samples: int):
        # Storage (Sets to zero)
        x1 = np.zeros(num_samples, dtype=np.complex64)
        x2 = np.zeros(num_samples, dtype=np.complex64)

        samples_read = 0  # How many samples have been read?

        while samples_read < num_samples:
            # amount of samples to take
            max_samps = min(self.buffer_size, num_samples - samples_read)

            # Fill buffer with samples
            self.sdr.sync_rx(self.buf, max_samps*2)  # *2 because there is 2 bytes in an int16

            # Convert from buffer to ints
            raw = np.frombuffer(self.buf, dtype=np.int16)

            # Reshape to [4, N] buffer
            # I.e. matrix, row0 = I1, Row1=Q1, Row2=I2, Row3=Q2
            raw = raw.reshape(-1, 4).T

            # Converting to complex samples
            c1 = (raw[0] + 1j * raw[1]) / 2048.0  # 2048 = Scale to -1 to 1 (12 bit ADC)
            c2 = (raw[2] + 1j * raw[3]) / 2048.0

```

```
            # Saving samples
            x1[samples_read: samples_read + max_samps] = c1[:max_samps]
            x2[samples_read: samples_read + max_samps] = c2[:max_samps]

            samples_read += max_samps  # How many samples have we read?

        arr = np.vstack([x1, x2])  # Stack the two arrays to one array
        return arr

    def close(self):
        self.rx1.enable = False
        self.rx2.enable = False
```

17 Front End Control Code

```
import numpy as np
from sdr_ctrl import sdr_ctrl

sdr = sdr_ctrl(40e6, 2.44175e9, 0)
arr = sdr.sample(8192)

arr = arr.T

with open('Noise.txt', 'a') as f:
    for index, item in enumerate(arr):
        f.write(str(arr[index][0])+";"+str(arr[index][1])+'\n')
sdr.close()
```

18 Antennaless Test Code for Beamforming

```
from bf import beamforming_das
from sdr_ctrl import sdr_ctrl

sdr = sdr_ctrl(40e6, 2.44175e9)

deg = -135  # Insert the phase delay here. from -180 to +180

i = 0
bfs = []  # Array for holding data
while True:
    i = i+1
    arr = sdr.sample(4)
    bf = beamforming_das(arr, 0.5, 2)
    print("bf: ", bf)  # debugging
    bfs.append(bf)

    arr = arr.T
    with open(str('BF_data_'+str(deg)+'deg'+str(i)+".txt"), 'a') as f:
        for index, item in enumerate(arr):
            f.write(str(arr[index][0])+";"+str(arr[index][1])+'\n')
    if i >= 5:
        with open('BF_results_'+str(deg)+'deg.txt', 'a') as f:
            for item in bfs:
                f.write(str(item)+'\n')
        sdr.close()
        exit()
```

19 Ideal Signal Beamforming Test Code

```
import time
import numpy as np
```

```

def beamforming_das(rx, distance, no_ele):
    theta_sweep = np.linspace(-1*np.pi/2, np.pi/2, 1000)
    k = np.arange(no_ele)
    results = []
    for thetas in theta_sweep:
        w = np.exp(2j * np.pi * distance * k * np.sin(thetas))
        y = w.conj().T @ rx
        results.append(np.abs(np.sum(y)))
    return np.rad2deg(theta_sweep[np.argmax(results)])

sample_rate = 61.44e6
N = 10000
t = np.arange(N)/sample_rate
angles = np.concatenate(([-89.9], np.arange(-89, 90, 1), [89.9]))
f = 2e4
d = 0.5
elements = 2
k = np.arange(elements)
tx = np.exp(2j * np.pi * f * t)
print("THETA_REF, THETA_MEAS, DELTA, COMPUTE TIME")
for theta_deg in angles:
    theta_rad = np.deg2rad(theta_deg)
    s = np.exp(2j * np.pi * d * k * np.sin(theta_rad))
    s = s.reshape(-1,1)
    tx = tx.reshape(1,-1)
    X = s @ tx

    start = time.perf_counter()
    bf_signal = beamforming_das(X, d, elements)
    stop = time.perf_counter()
    print(f"{theta_deg}, {bf_signal:.6f}, {np.abs(theta_deg-bf_signal):.6f}, {stop-start:.4f}")
```

20 N=1 SNR Sweep Beamforming Test Code

```
import time
import numpy as np

def beamforming_das(rx, distance, no_ele):
    theta_sweep = np.linspace(-1*np.pi/2, np.pi/2, 78948)
    k = np.arange(no_ele)
    results = []
    for thetas in theta_sweep:
        w = np.exp(2j * np.pi * distance * k * np.sin(thetas))
        y = w.conj().T @ rx
        results.append(np.abs(np.sum(y)))
    return np.rad2deg(theta_sweep[np.argmax(results)])

def snr_to_peak_amplitude(SNR):
    snr_lin = 10 ** (SNR / 20)
    a_noise_rms = 1 / np.sqrt(snr_lin)
    a_noise = a_noise_rms * np.sqrt(2)
    return a_noise

sample_rate = 61.44e6
N = 1
t = np.arange(N)/sample_rate
angles = np.concatenate(([-89.9], np.arange(-89, 90, 1), [89.9]))
f = 2e4
d = 0.5
elements = 2
k = np.arange(elements)
tx = np.exp(2j * np.pi * f * t)

print("SNR, THETA_REF, THETA_MEAS, DELTA, COMPUTE TIME")
```

```
for snr in np.arange(-50, 110, 10):
    for theta_deg in angles:
        theta_rad = np.deg2rad(theta_deg)
        s = np.exp(2j * np.pi * d * k * np.sin(theta_rad))
        s = s.reshape(-1,1)
        tx = tx.reshape(1,-1)
        X = s @ tx
        n = np.random.randn(elements, N) + 1j * np.random.randn(elements, N)
        X_n = X + snr_to_peak_amplitude(snr) * n

        start = time.perf_counter()
        bf_signal = beamforming_das(X_n, d, elements)
        stop = time.perf_counter()
        print(f"{snr}, {theta_deg}, {bf_signal:.6f}, {np.abs(theta_deg-bf_signal):.6f},
        ↪  {stop-start:.4f}")
```

21 N Sweep Monte Carlo Test Code

```
import numpy as np

def beamforming_das(rx, distance, no_ele):
    theta_sweep = np.linspace(-1*np.pi/2, np.pi/2, 78948) # Deler -pi/2 til pi/2 op i 78948 segmenter og
    ↪  gemmer i et array
    results = [] # Laver tomt array til resultater
    for thetas in theta_sweep: # For alle de værdier i theta sweep, kører vi
       w = np.exp(2j * np.pi * distance * np.arange(no_ele) * np.sin(thetas)) # Weight vektor tilsvarende
       ↪  "steering vektoren" køres for theta
       y = w.conj().T @ rx # Vi kompleks konjugere for at modarbejde faseforskydningen og transponerer så
       ↪  matricen har den rigtige form
       results.append(np.abs(np.sum(y))) # Vi finder modulus af de to beamformede datapunkter summeret og
       ↪  gemmer dem i arrayet
    return np.rad2deg(theta_sweep[np.argmax(results)]) # Vi kigger efter den theta hvor amplituden er
    ↪  størst og returnere den i grader

sample_space = np.concatenate([np.arange(1, 10) * 10**k for k in range(6)])
sample_space = np.append(sample_space, 1000000)
sample_rate = 61.44e6
f = 2e4
d = 0.5 # half wavelength spacing
theta_deg = 0 # Angle of arrival
theta_rad = np.deg2rad(theta_deg)
elements = 2
k = np.arange(elements)


print('---Noisy Test Start---')
print('SAMPLE SIZE, ELAPSED TIME (s), AOA')
for sample_size in sample_space:
    N = sample_size  # samples
    print(N)
    for i in range(101):
        t = np.arange(N)/sample_rate # time vector
        tx = np.exp(2j * np.pi * f * t)
        s = np.exp(2j * np.pi * d * k * np.sin(theta_rad))
        s = s.reshape(-1,1) # make s a column vector
        tx = tx.reshape(1,-1) # make tx a row vector
        X = s @ tx
        n = np.random.randn(elements, N) + 1j*np.random.randn(elements, N) #Noise
        X_n = X + 0.1 * n
        start = time.perf_counter()
        results = beamforming_das(X_n, d, elements)
        stop = time.perf_counter()
        print(f'{results:.6f}', end=', ')
    print('')
```

22 Integrated Beamforming Test Simulation Code

```python
import numpy as np
import timeit

def beamforming_das(rx, distance, no_ele):
    theta_sweep = np.linspace(-1*np.pi/2, np.pi/2, 78948)
    results = []
    for thetas in theta_sweep:
        w = np.exp(2j * np.pi * distance * np.arange(no_ele) * np.sin(thetas))
        y = w.conj().T @ rx
        results.append(np.abs(np.sum(y)))
    return np.rad2deg(theta_sweep[np.argmax(results)])

sample_rate = 61.44e6
N = 900 # sample size
t = np.arange(N)/sample_rate # time vector

for AoA in np.arange(-45, 46, 15):
    print(AoA)
    for i in range(5):
        ### TX SIGNAL
        theta_deg = AoA # reference AoA
        theta_rad = np.deg2rad(theta_deg)
        f = 2e4 # base-band signal frequency
        tx = np.exp(2j * np.pi * f * t) # base-band signal
        d = 1 # distance between elements
        elements = 2 # number of elements in the array
        k = np.arange(elements)

        s = np.exp(2j * np.pi * d * k * np.sin(theta_rad)) # steering vector
        s = s.reshape(-1,1) # row -> column
        tx = tx.reshape(1,-1) # column -> row

        ### RX SIGNAL
        X = s @ tx # received signal matrix
        n = np.random.randn(elements, N) + 1j*np.random.randn(elements, N) # AWGN Noise
        X_n = X + 0.01 * n

        ### TEST CODE
        bf_signal_c = beamforming_das(X_n, 0.5, elements)
        print(bf_signal_c, end=', ')
        if i == 5:
            print()
```

23 TARP Control Code

23.1 Primary Code

```
float tiltInDegrees = 0;
float aziInDegrees = 0;
float angleAzi = 0;
float angleTilt = 0;
//encoder_Azimut pins
static int pinA_azi = 38;  //Pin A
static int pinB_azi = 40;  //Pin B
static int pinZ_azi = 37;  //Pin Z
//Motor_Azimut pins:
static int ena_pin_azi = 13;  //enable pin controls the motor speed with PWM
static int dir_azi = 14;      // logic input 1
//encoder_tilt pins
static int pinA_tilt = 26;  //Pin A
static int pinB_tilt = 33;  //Pin B
static int pinZ_tilt = 21;  //Pin Z
```

```
16  //Encoder_tilt_motor pins
17  static int pinA_tilt_motor = 20;  //Pin A
18  //Motor_tilt pins:
19  static int ena_pin_tilt = 16;  //enable pin controls the motor speed with PWM
20  static int dir1_tilt = 18;     // logic input 1
21  static int dir2_tilt = 17;     //logic input 2
22  //btn
23  static int btn_yellow = 42;
24  static int btn_blue = 41;
25  //------------------------------------------------------------------------------//
26  #include "soc/gpio_struct.h"
27  #include "driver/gpio.h"
28  #include <WiFi.h>
29  #include "esp_timer.h"  //used for setting a specific sampling rate
30  // Fast read
31  #define READ_PIN(pin) ((pin) < 32 ? ((GPIO.in >> (pin)) & 1) : ((GPIO.in1.val >> ((pin)-32)) & 1))
32  //Vars for encoder azimut
33  volatile int pos_azi = 0;
34  volatile int rot_azi = 0;
35  //Vars for encoder tilt
36  volatile int pos_tilt = 0;
37  volatile int rot_tilt = 0;
38  //Vars for button interrupt
39  //Needs to be used for emergency braking
40  volatile bool yellow_interrupt = false;
41  volatile bool blue_interrupt = false;
42  //variables for intercore communication
43  volatile float targetAzi = 0;  //input angle from PI or PC
44  volatile float targetTilt = 0;
45  volatile float currentAzi = 0;
46  volatile float currentTilt = 0;
47  //Vars for P controller
48  static float controllerGainAzi = 24.83;  //from simulated model
49  static float controllerGainTilt = 4.32;  // from simulated model
50  static int aziOffset = 110;              // 110 - minimum voltage required for the azimut motor to run
51  static int tiltOffset = 300;              // 300 - minimum voltage required for the tilt motor to run
52  static int sampleRate = 200;              // sample rate in micros seconds
53  //------------------MAX 150 micros seconds right now!!!!!!!
54  WiFiServer server(1234);  // TCP server on port 1234
55  WiFiClient client;
56  //setting up 2 cores to run in parallel (FreeRTOS)
57  TaskHandle_t core1;
58  TaskHandle_t core2;
59  SemaphoreHandle_t targetAngleMutex;   //for safe passing of variables between the two cores
60  SemaphoreHandle_t currentAngleMutex;  //for safe passing of variables between the two cores
61
62  void setup() {
63    pinSetup();
64    init_serial();
65    attachInt();
66    init_wireless();
67
68    targetAngleMutex = xSemaphoreCreateMutex();   // Create the lock for target angle
69    currentAngleMutex = xSemaphoreCreateMutex();  // Create the lock for current angle
70    xTaskCreatePinnedToCore(
71      Core1Loop,      /* Task function. */
72      "Control Loop", /* name of task. */
73      10000,          /* Stack size of task */
74      NULL,           /* parameter of the task */
75      3,              /* priority of the task */
76      &core1,         /* Task handle to keep track of created task */
77      1);             /* pin task to core 0 */
78
79    xTaskCreatePinnedToCore(
80      Core2Loop,            /* Task function. */
81      "Communication Loop", /* name of task. */
82      10000,                /* Stack size of task */
```

```
    NULL,                 /* parameter of the task */
    1,                    /* priority of the task */
    &core2,               /* Task handle to keep track of created task */
    0);                   /* pin task to core 0 */

  disableCore1WDT();  //Disables watchdog on core1 as maintance is handled by core 0

  tiltHome();  // We need to home before the timer starts, form there everything is automatic

  init_sample_rate_timer();  //start sample rate timer
}

void Core1Loop(void* pvParameters) {

  while (true) {
    ulTaskNotifyTake(pdTRUE, portMAX_DELAY);  // Sleep until timer pulses
    saveSamples();
    controlCode();                            // Runs on Core 1
  }
}

void Core2Loop(void* pvParameters) {
  while (true) {
    vTaskDelay(5 / portTICK_PERIOD_MS);  // essential to ensure watchdog timer is not triggered
                                         //----------- Connection to PC ----------------
    client = server.available();
    if (client) {  //connect to PC, once connected
      Serial.println("Connected to PC");
      client.println("Give me an angle");
      client.setTimeout(1);  // controls the timeout needed for ESP32 to read input from PuTTy
      while (true) {
        readFromPC();
        printData();
        vTaskDelay(5 / portTICK_PERIOD_MS);  // essential to ensure watchdog timer is not triggered
      }
    }
  }
}

//Used for saving samples
int count = 0;
bool sampleFlag = 0;
float samples[50000];

void saveSamples() {
  if (count < 50000 && angleAzi != 0) {
    samples[count] = aziInDegrees;
    count++;
  } else if (count == 50000) {
    sampleFlag = 1;
    count++;  //stops further sampling from happening
    while(true){ //turn off
      analogWrite(ena_pin_tilt, 0);
      analogWrite(ena_pin_azi, 0);
    }
  }
}

float convertPulsesToAngle(float pos, int gearing) {
  float position = (pos / (gearing * 1000)) * 360;  // current position converted to degrees
  return position;
}

//never used
void loop() {}
```

23.2 Initialization Code

```
//Interrups:
//Azimut:
void IRAM_ATTR PinA_R_azi() {
  if (READ_PIN(pinB_azi) == 1) {  //Pin B is high, Pin A is rising
    pos_azi--;
  } else {  //Pin B is low, Pin A is rising
    pos_azi++;
  }
}
void IRAM_ATTR PinA_F_azi() {
  if (READ_PIN(pinB_azi) == 1) {  //Pin B is high, Pin A is falling
    pos_azi++;
  } else {  //Pin B is low, Pin A is falling
    pos_azi--;
  }
}
void IRAM_ATTR PinB_R_azi() {
  if (READ_PIN(pinA_azi) == 1) {  //Pin A is high, Pin B is rising
    pos_azi++;
  } else {  //Pin A is low, Pin B is rising
    pos_azi--;
  }
}
void IRAM_ATTR PinB_F_azi() {
  if (READ_PIN(pinA_azi) == 1) {  //Pin A is high, Pin B is falling
    pos_azi--;
  } else {  //Pin a is low, Pin B is falling
    pos_azi++;
  }
}
void IRAM_ATTR PinZ_R_azi() {
  if (READ_PIN(pinZ_azi) == 1) {  //Pin A is high, Pin Z is Rising
    //rot_azi++;
  } else {
    //rot_azi--;
  }
}

//Tilt:
void IRAM_ATTR PinA_R_tilt() {
  if (READ_PIN(pinB_tilt) == 1) {  //Pin B is high, Pin A is rising
    pos_tilt--;
  } else {  //Pin B is low, Pin A is rising
    pos_tilt++;
  }
}
void IRAM_ATTR PinA_F_tilt() {
  if (READ_PIN(pinB_tilt) == 1) {  //Pin B is high, Pin A is falling
    pos_tilt++;
  } else {  //Pin B is low, Pin A is falling
    pos_tilt--;
  }
}
void IRAM_ATTR PinB_R_tilt() {
  if (READ_PIN(pinA_tilt) == 1) {  //Pin A is high, Pin B is rising
    pos_tilt++;
  } else {  //Pin A is low, Pin B is rising
    pos_tilt--;
  }
}
void IRAM_ATTR PinB_F_tilt() {
  if (READ_PIN(pinA_tilt) == 1) {  //Pin A is high, Pin B is falling
    pos_tilt--;
  } else {  //Pin a is low, Pin B is falling
    pos_tilt++;
```

```
  }
}
void IRAM_ATTR PinZ_R_tilt() {
  if (READ_PIN(pinZ_tilt) == 1) {  //Pin A is high, Pin Z is Rising
    //rot_tilt++;
  } else {
    //rot_tilt--;
  }
}
//tilt buttons
void IRAM_ATTR btnYellowInterrupt() {
  yellow_interrupt = !READ_PIN(btn_yellow);
}
void IRAM_ATTR btnBlueInterrupt() {
  blue_interrupt = !READ_PIN(btn_blue);
}
// Samplerate timer interrupt
void IRAM_ATTR timer_callback(void* arg) {
BaseType_t higherWoken = pdFALSE;
    vTaskNotifyGiveFromISR(core1, &higherWoken);
    if(higherWoken) portYIELD_FROM_ISR();
}

void attachInt() {
  //Interrups:
  //Azi
  attachInterrupt(digitalPinToInterrupt(pinA_azi), PinA_R_azi, RISING);
  attachInterrupt(digitalPinToInterrupt(pinA_azi), PinA_F_azi, FALLING);
  attachInterrupt(digitalPinToInterrupt(pinB_azi), PinB_R_azi, RISING);
  attachInterrupt(digitalPinToInterrupt(pinB_azi), PinB_F_azi, FALLING);
  attachInterrupt(digitalPinToInterrupt(pinZ_azi), PinZ_R_azi, RISING);
  //Tilt
  attachInterrupt(digitalPinToInterrupt(pinA_tilt), PinA_R_tilt, RISING);
  attachInterrupt(digitalPinToInterrupt(pinA_tilt), PinA_F_tilt, FALLING);
  attachInterrupt(digitalPinToInterrupt(pinB_tilt), PinB_R_tilt, RISING);
  attachInterrupt(digitalPinToInterrupt(pinB_tilt), PinB_F_tilt, FALLING);
  attachInterrupt(digitalPinToInterrupt(pinZ_tilt), PinZ_R_tilt, RISING);
  //buttons
  attachInterrupt(digitalPinToInterrupt(btn_blue), btnBlueInterrupt, FALLING);
  attachInterrupt(digitalPinToInterrupt(btn_yellow), btnYellowInterrupt, FALLING);
}

void init_sample_rate_timer() {
    // Declare a timer handle
  esp_timer_handle_t periodic_timer;
  // Define the timer creation arguments
  const esp_timer_create_args_t timer_args = {
    .callback = &timer_callback,
    .name = "my_periodic_timer"
  };
  // Create the timer
  ESP_ERROR_CHECK(esp_timer_create(&timer_args, &periodic_timer));
  // 3. Start the Timer Periodically
  ESP_ERROR_CHECK(esp_timer_start_periodic(periodic_timer, sampleRate));
}
void pinSetup() {
  //Init azi pins
  //Motor:
  pinMode(ena_pin_azi, OUTPUT);
  analogWriteFrequency(ena_pin_azi, 20000); // 19kHz frequency for the PWM signal
   analogWriteResolution(ena_pin_azi, 9); // 2^9 resolution
  pinMode(dir_azi, OUTPUT);
  digitalWrite(ena_pin_azi, 0);
  digitalWrite(dir_azi, 0);
  //Encoder:
  pinMode(pinA_azi, INPUT_PULLUP);
  pinMode(pinB_azi, INPUT_PULLUP);
```

```
  pinMode(pinZ_azi, INPUT_PULLUP);
  //Init tilt:
  //Motor:
  pinMode(ena_pin_tilt, OUTPUT);
  analogWriteFrequency(ena_pin_tilt, 20000); // 2kHz frequency for the PWM signal - sets resolution to 2^9 (512)
  analogWriteResolution(ena_pin_tilt, 9); // 2^9 resolution
  pinMode(dir1_tilt, OUTPUT);
  pinMode(dir2_tilt, OUTPUT);
  digitalWrite(ena_pin_tilt, 0);
  digitalWrite(dir1_tilt, 0);
  digitalWrite(dir2_tilt, 0);
  //Encoder:
  pinMode(pinA_tilt, INPUT_PULLUP);
  pinMode(pinB_tilt, INPUT_PULLUP);
  pinMode(pinZ_tilt, INPUT_PULLUP);

  //Encoder_motor:
  pinMode(pinA_tilt_motor, INPUT_PULLUP);
  //buttons:
  pinMode(btn_yellow, INPUT_PULLUP);
  pinMode(btn_blue, INPUT_PULLUP);
}
```

23.3 Motor Control Code

```
void tiltHome() {
  digitalWrite(dir1_tilt, 0);  // homing
  digitalWrite(dir2_tilt, 1);
  analogWrite(ena_pin_tilt, 350);

  while (digitalRead(btn_blue) == true) {
    //wait til the blue wired button is hit, ensuring we home
  }
  //turn off
  analogWrite(ena_pin_tilt, 0);
  digitalWrite(dir1_tilt, 0);
  digitalWrite(dir2_tilt, 0);
  pos_tilt = 0;  //setting starting position to 0
}

int gearing;  //gearing used in the encoders
void controlCode() {
  //Converting volitile variables to non-volitile
  if (xSemaphoreTake(targetAngleMutex, portMAX_DELAY)) {
    angleAzi = targetAzi;
    angleTilt = targetTilt;
    xSemaphoreGive(targetAngleMutex);
  }
  //-----------------TILT CONTROL--------------------
  gearing = 1;  // which gearing is running on the sensor
  tiltInDegrees = convertPulsesToAngle(pos_tilt, gearing);
  float error = angleTilt - tiltInDegrees;
  float deltaVolt = error * controllerGainTilt;
  Serial.println(deltaVolt);
  setVelocity(deltaVolt, ena_pin_tilt, tiltOffset);
  //----------------AZIMUT CONTROL-------------------
  gearing = 5;  //gearing on sensor for azimuth
  aziInDegrees = convertPulsesToAngle(pos_azi, gearing);
  error = angleAzi - aziInDegrees;  //calculate error
  deltaVolt = error * controllerGainAzi;
  setVelocity(deltaVolt, ena_pin_azi, aziOffset);
  // -------- update position to PC or PI --------------
  if (xSemaphoreTake(currentAngleMutex, portMAX_DELAY)) {  //update current position
    currentTilt = tiltInDegrees;
    currentAzi = aziInDegrees;
```

```
    xSemaphoreGive(currentAngleMutex);
  }
}
//Note: motor = 0 -> Azimut motor
//      motor = 1 -> Tilt motor
void setVelocity(float deltaVolt, int motor, float offset) {
  bool direction = 1;
  float velocity = 0;
  if (deltaVolt < 0) {                    // ensures the offset is inverted if the delta volt is negative
    velocity = offset + abs(deltaVolt);  //get the absolute value, cant use negative values
    direction = 0;                        //change direction
  } else {
    velocity = offset + deltaVolt;
    direction = 1;
  }
  if (velocity > 511) {  //capping so this is the max speed
    velocity = 511;
  }
  //----------------AZIMUT MOTOR CONTROL-------------------
  if (motor == ena_pin_azi) {
    analogWrite(ena_pin_azi, 0);          // prevents short circuit
    delayMicroseconds(5);                 // needed to take care of time delay. Recorded 2.3 us delay from
    ↪  switching
    digitalWrite(dir_azi, !direction);   //control direction
    analogWrite(ena_pin_azi, velocity);  //control speed
  }
  //-----------------TILT MOTOR CONTROL--------------------
  else if (motor == ena_pin_tilt) {
    digitalWrite(dir1_tilt, direction);  //control direction
    digitalWrite(dir2_tilt, !direction);
    analogWrite(ena_pin_tilt, velocity);  //control speed
  }
}
```

23.4 Communication Code

```
////////////////////// USED FOR COLLECTING DATA ////////////////////////
void readFromPC() {
  // Variables to hold the parsed data
  int motorID = -1;  // Default to -1 (error/no motor selected)
  int angle = 0;     // angle input
  String data = client.readStringUntil('\n');

  // Check if the received string is NOT empty (meaning there is a message)
  if (!data.isEmpty()) {

    data.trim();  // Clean up the string (removes newline/carriage return and leading/trailing spaces)

    // 1. Find the position of the separator (space)
    int separatorIndex = data.indexOf(';');

    // Check if a space was found AND it's not at the very beginning or end
    if (separatorIndex > 0) {

      // 2. Extract the first part (Motor ID)
      String motorIDStr = data.substring(0, separatorIndex);
      motorID = motorIDStr.toInt();  // Convert the substring to an integer

      // 3. Extract the second part (Angle)
      // Start reading AFTER the space, to the end of the string
      String angleStr = data.substring(separatorIndex + 1);
      angle = angleStr.toInt();  // Convert the substring to an integer

      if (angle < 0 || angle > 360) {  //protection code ensuring the angle is not higher
        motorID = (-1);                // ensures the angle is not used
      }
```

```
      // 4. Verification and Action
      client.println("Angle Received!");  // Acknowledge receipt

      // --- Execute the Motor Control Logic ---
      if (motorID == 0) {
        if (xSemaphoreTake(targetAngleMutex, portMAX_DELAY)) {  //ensures safe passing
          targetAzi = angle;
          xSemaphoreGive(targetAngleMutex);
        }
        client.print("Controlling Azimut. Angle: ");
        client.println(angle); //number represents amount of decimals
      } else if (motorID == 1) {
        if (xSemaphoreTake(targetAngleMutex, portMAX_DELAY)) {  //ensures safe passing
          targetTilt = angle;
          xSemaphoreGive(targetAngleMutex);
        }
        client.print("Controlling Tilt. Angle: ");
        client.println(angle);
      } else {
        // Neither 0 nor 1 was specified
        client.print("Error: Invalid Motor ID received: ");
        client.println(motorID);
      }
    } else {
      // Handle case where input was received but was not in the expected format (e.g., "1180" or "1 180
      ↪  extra")
      client.print("Error: Input received but format invalid: ");
      client.println(data);
    }
  }
}

void printData() {
  //printing samples
  if (sampleFlag == 1) {
    for (int i = 0; i < 50000; i++) {
      client.println(samples[i], 6);
      delay(1);
    }
    sampleFlag = 0;
    //client.println("Done printing");
  }
}
```

23.5 Wireless Connection Code

```
// TARP credentials
//H123 Alt wifi skal udkommenteres når pien sidder på
const char* ssid = "TARP";
const char* password = "12345678H";  // min 8 characters
void init_wireless() {
  // Start the ESP32 in Access Point mode
  WiFi.softAP(ssid, password);
  Serial.println("TARP Started!");
  Serial.print("Connect your PC to SSID: ");
  Serial.println(ssid);

  IPAddress IP = WiFi.softAPIP();
  Serial.print("ESP32 IP address: ");
  Serial.println(IP);
  // Start TCP server
  server.begin();
  Serial.println("TCP server started on port 1234");
}
```

24 TARP Integration Code

The TARP integration code contains the following and the *Initialization* and *Motor control* code from Appendix 23.

24.1 Main Control Code

```
float tiltInDegrees = 0;
float aziInDegrees = 0;
float angleAzi = 0;
float angleTilt = 0;

//encoder_Azimut pins
static int pinA_azi = 38;  //Pin A
static int pinB_azi = 40;  //Pin B
static int pinZ_azi = 37;  //Pin Z

//Motor_Azimut pins:
static int ena_pin_azi = 13;  //enable pin controls the motor speed with PWM
static int dir_azi = 14;      // logic input 1

//encoder_tilt pins
static int pinA_tilt = 26;  //Pin A
static int pinB_tilt = 33;  //Pin B
static int pinZ_tilt = 21;  //Pin Z

//Encoder_tilt_motor pins
static int pinA_tilt_motor = 20;  //Pin A

//Motor_tilt pins:
static int ena_pin_tilt = 16;  //enable pin controls the motor speed with PWM
static int dir1_tilt = 18;     // logic input 1
static int dir2_tilt = 17;     //logic input 2

//btn
static int btn_yellow = 42;
static int btn_blue = 41;

//------------------------------------------------------------------------------//
#include "soc/gpio_struct.h"
#include "driver/gpio.h"
#include <WiFi.h>
#include "esp_timer.h"  //used for setting a specific sampling rate
// Fast read
#define READ_PIN(pin) ((pin) < 32 ? ((GPIO.in >> (pin)) & 1) : ((GPIO.in1.val >> ((pin)-32)) & 1))

//Vars for encoder azimut
volatile int pos_azi = 0;
volatile int rot_azi = 0;

//Vars for encoder tilt
volatile int pos_tilt = 0;
volatile int rot_tilt = 0;

//Vars for button interrupt
//Needs to be used for emergency braking
volatile bool yellow_interrupt = false;
volatile bool blue_interrupt = false;

//variables for intercore communication
volatile float targetAzi = 0;  //input angle from PI or PC
volatile float targetTilt = 0;
volatile float currentAzi = 0;
volatile float currentTilt = 0;

```

```
59  //Vars for P controller
60  static float controllerGainAzi = 24.83;  //from simulated model
61  static float controllerGainTilt = 4.32;  // from simulated model
62  static int aziOffset = 110;              // 110 - minimum voltage required for the azimut motor to run
63  static int tiltOffset = 300;              // 300 - minimum voltage required for the tilt motor to run
64  static int sampleRate = 200;              // sample rate in micros seconds
65  //------------------MAX 150 micros seconds right now!!!!!!!
66
67  WiFiServer server(1234);  // TCP server on port 1234
68  WiFiClient client;
69
70  //setting up 2 cores to run in parallel (FreeRTOS)
71  TaskHandle_t core1;
72  TaskHandle_t core2;
73  SemaphoreHandle_t targetAngleMutex;   //for safe passing of variables between the two cores
74  SemaphoreHandle_t currentAngleMutex;  //for safe passing of variables between the two cores
75
76  void setup() {
77
78    pinSetup();
79
80    init_serial();
81
82    attachInt();
83
84    targetAngleMutex = xSemaphoreCreateMutex();   // Create the lock for target angle
85    currentAngleMutex = xSemaphoreCreateMutex();  // Create the lock for current angle
86
87
88
89    xTaskCreatePinnedToCore(
90      Core1Loop,      /* Task function. */
91      "16384 Loop", /* name of task. */
92      16384,          /* Stack size of task */
93      NULL,           /* parameter of the task */
94      3,              /* priority of the task */
95      &core1,         /* Task handle to keep track of created task */
96      1);             /* pin task to core 0 */
97
98    xTaskCreatePinnedToCore(
99      Core2Loop,            /* Task function. */
100     "Communication Loop", /* name of task. */
101     16384,                /* Stack size of task */
102     NULL,                 /* parameter of the task */
103     1,                    /* priority of the task */
104     &core2,               /* Task handle to keep track of created task */
105     0);                   /* pin task to core 0 */
106
107   disableCore1WDT();  //Disables watchdog on core1 as maintance is handled by core 0
108
109   tiltHome();  // We need to home before the timer starts, form there everything is automatic
110
111   init_sample_rate_timer();  //start sample rate timer
112 }
113
114
115 void Core1Loop(void* pvParameters) {
116
117   while (true) {
118     ulTaskNotifyTake(pdTRUE, portMAX_DELAY);  // Sleep until timer pulses
119     controlCode();                            // Runs on Core 1
120   }
121 }
122
123
124 void Core2Loop(void* pvParameters) {
125   while (true) {
```

```
    vTaskDelay(5 / portTICK_PERIOD_MS);  // essential to ensure watchdog timer is not triggered
    get_serial_cmd();
    send_serial_pos();
  }
}

float convertPulsesToAngle(float pos, int gearing) {
  float position = (pos / (gearing * 1000)) * 360;  // current position converted to degrees
  return position;
}

//never used
void loop() {}
```

24.2 Code for Serial Communication

```
void init_serial() {
  Serial.begin(115200);
  Serial.setTimeout(50);  //50 ms timeout
  delay(1000);            //wait for serialport
}

//1 bit is used to define motor (0 = azi, 1=tilt)
//12 bits are required to send 360 degrees with 0.1 degree resolution.
//This is sent as 3600 then divided locally
//3 bits CAN be used for CRC (currently not implemented)
//then ; as end
void get_serial_cmd() {
  if (Serial.available()>2) {
    byte recv_bytes[2];
    int read_bytes = Serial.readBytesUntil(';', recv_bytes, sizeof(recv_bytes));

    if (read_bytes != 2) {
      while (Serial.available() > 0) Serial.read();  // clear input buffer
      return;
    }
    byte b1 = recv_bytes[0];
    byte b2 = recv_bytes[1];

    Serial.read();


    uint16_t degrees = (uint16_t)(b1 << 8) | b2;
    bitClear(degrees, 15);
    degrees = degrees >> 3;
    float deg = degrees * 0.1;

    if (bitRead(b1, 7) == 0) {
      //Serial.print("azi degrees: ");
      //Serial.println(deg);
      if (xSemaphoreTake(targetAngleMutex, portMAX_DELAY)) {  //ensures safe passing
        targetAzi = deg;
        xSemaphoreGive(targetAngleMutex);
      }
    } else {
      //Serial.print("tilt degrees: ");
      //Serial.println(deg);
      if (xSemaphoreTake(targetAngleMutex, portMAX_DELAY)) {  //ensures safe passing
        targetTilt = deg;
        xSemaphoreGive(targetAngleMutex);
      }
    }
  }
}

void send_serial_pos() {
```

```
  //Input should be in degrees
  float tempAzi;
  float tempTilt;
  if (xSemaphoreTake(currentAngleMutex, portMAX_DELAY)) {  //ensures safe passing
    tempTilt = currentTilt;
    tempAzi = currentAzi;
    xSemaphoreGive(currentAngleMutex);
  }
  Serial.print(tempAzi);
  Serial.print(";");
  Serial.print(tempTilt);
  Serial.println("");  // /n
}
```

25 Python Communication to ESP

25.1 Motor Communication

```python
from serial import Serial
from SerialRW.SerialRW import serial_read, serial_write


class MotorCtrl:
    def __init__(self, serial_inst: Serial):
        self.serial_inst = serial_inst

    def tilt(self, degrees: float | int) -> None:
        """ Controls tilt motor
        args:
            degrees (float/int): degrees in which tilt motor should move to
        returns:
            None
        """
        self.__motor_cmd(True, degrees)

    def azi(self, degrees: float | int) -> None:
        """
        Controls Azimuth motor
        args:
            degrees (float/int): degrees in which azimuth motor should move to
        returns:
            None
        """
        self.__motor_cmd(False, degrees)

    def __motor_cmd(self, motor: bool, degrees: float | int) -> None:
        """ Commands a motor, to move to a certain angle
        args:
            motor (bool): If true -> tilt, if false -> Azi
            degrees (float/int): degrees in which the motor should move to
        returns:
            None
        """
        while degrees > 360:
            degrees -= 360
        while degrees < 0:
            degrees += 360

        if not isinstance(motor, bool):
            raise ValueError("Motor must bool")

        if isinstance(degrees, (int, float)):
            degrees = round(degrees, 1)
            degrees *= 10  # For transfere, degrees are multiplied by 10, then divided again on the ESP
```

```
            package = self.__build_motor_packet(motor, int(degrees))

            serial_write(self.serial_inst, package)
        else:
            raise ValueError("MotorCtrl expects int or float")

    def __build_motor_packet(self, motor: bool, degrees: int) -> bytes:
        """ Builds the serial package for controlling a motor
        args:
            motor (bool): If true -> tilt, if false -> Azi
            degrees (float|int): degrees in which the motor should move to
        returns:
            serial package (bytes): Serial package to send to ESP
        """
        if not isinstance(motor, bool):
            raise ValueError("Motor must be bool")
        if not isinstance(degrees, int) or not (0 <= degrees <= 3600):
            raise ValueError("Degrees must be an integer between 0 and 3600")

        # Bit packing: [motor(1)][degrees(12)][unused(3)=0] -> 16 in total
        motor_bit = int(format(motor, 'b'))
        # Forces to only keep 12 bits (0xFFF is just to tell python this)
        degrees_bits = degrees & 0xFFF

        # Shifts motor_bit to MSB, and shifts degrees 3 to have 3 zeros in the end
        packed = (motor_bit << 15) | (degrees_bits << 3)

        # Prepare bytes:
        b1 = (packed >> 8) & 0xFF  # Bit 1 containing the MSB part of package
        b2 = packed & 0xFF  # Bit 2 containing the LSB part of package

        result = bytes([b1, b2, ord(';')])
        return result

    def read_pos(self) -> tuple:
        """ Reads position of azimuth and tilt motor.
        args:
            None
        returns:
            pos (tuple): Containing azimuth position, tilt position
        """
        self.serial_inst.reset_input_buffer()
        data = serial_read(self.serial_inst, timeout=0.2)

        azi, tilt = data.split(";")

        return (float(azi), float(tilt))


# EXAMPLE USAGE
if __name__ == "__main__":
    esp = Serial(port='/dev/ttyUSB0', baudrate=115200, timeout=1)
    motor_ctrl = MotorCtrl(esp)
    motor_ctrl.tilt(370)
    motor_ctrl.read_pos()
```

25.2 SerialRW

```
"""Module handling serial read and write"""
from time import time, sleep
from serial import Serial


def __check_serial_open(serial_inst: Serial) -> None:
    """ Checks if serial instance is open, if not tries to open
            args:
```

```
                serialInst (Serial): The serial instance in question
            returns:
                None
            On error:
                throws runtime error
    """
    if not serial_inst.isOpen():
        try:
            serial_inst.open()
        except Exception as e:
            raise RuntimeError("Error cant open serialport") from e


def serial_read(serial_inst: Serial, utf: bool = True, timeout: int = 5) -> bytes | object | None:
    """Reads serial data from a serial instance. MUST BE SET UP prior to running.
        Args:
            serialInst (Serial): a serial Instance, defined using serial.Serial()
            utf (Bool): Default True, if true, input will be decoded using UTF. NB this also splits chars
            ↪  at newline ('\n')
            timeout (int): Default 5, secounds to wait for data.
        returns:
            if uft=true: an object containing strings split at newline
            if utf=false: raw bytes
            On timeout: None
    """
    __check_serial_open(serial_inst)

    t0 = time()
    while t0+timeout > time():
        sleep(timeout/30)  # To avoid unnessesary high CPU usage
        if serial_inst.in_waiting:
            packet = serial_inst.readline()
            if utf:
                return packet.decode('utf').split('\n')[0]
            return packet
    return None


def serial_write(serial_inst: Serial, data: bytes | str) -> None:
    """
    Writes data out to an initilized serial instance. MUST BE SET UP prior to running.
        args:
            serialInst (Serial): a serial Instance, defined using serial.Serial()
            data (Bytes|str): data to write, either as bytes or string. NB it will be encoded using utf-8
        Returns:
            None
    """
    __check_serial_open(serial_inst)


    if isinstance(data, str):  # If data is string, encode it propperly
        serial_inst.write(data.encode('utf-8'))
        return
    serial_inst.write(data)

```

26 Detection Time Test

```
Main:
from serial import Serial
from SerialRW.MotorComms import MotorCtrl
from bf import beamforming_das
from sdr_ctrl import sdr_ctrl
from time import sleep, time

```

```
sdr = sdr_ctrl(40e6, 2.44175e9)
esp = Serial(port='/dev/ttyUSB0', baudrate=115200, timeout=1)
motor = MotorCtrl(esp)

for _ in range(0,5):
    motor.azi(180)
    sleep(1)
    motor.tilt(155)
    sleep(1)

    t0 = time()
    motor.azi(180-120)
    sleep(3)#wait until azi is stable
    arr = sdr.sample(900)
    beamforming_das(arr, 0.5, 2)
    print(time()-t0)
sdr.close()
```

27 TARP Closeup of Callouts on Diagram

This appendix contains closeup pictures of the callouts on Figure 4.58.

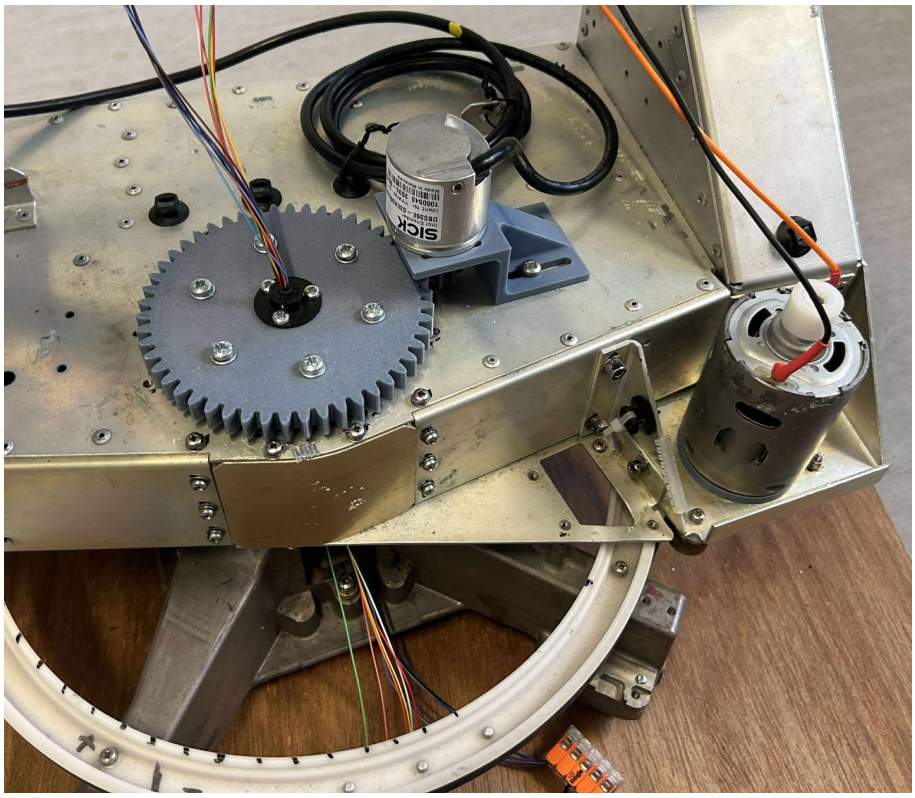
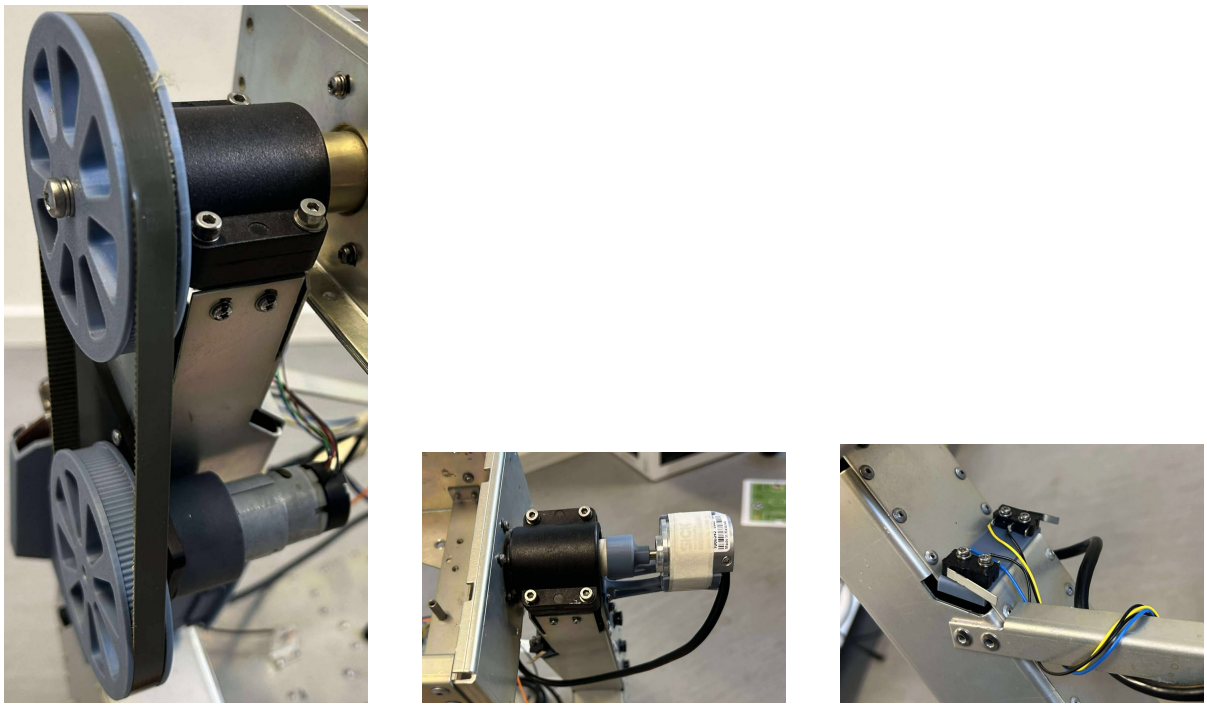


Figure 8.24: A closer view of callout 2 and 3 from figure 4.58. Here the drivetrain for azimuth can also be seen.



(a) **(b)** **(c)**

Figure 8.25: (a) is a closer view of callout 4 and 5, **(b)** of callout 6 and **(c)** of callout 7, all from figure 4.58. It should be mentioned, that the motor, seen in the bottom of **(a)** also contains an internal gearing.