DALGONA

RF based Antenna Tracking System

Version 1.0

Date: 06.04.2025

Design Studio Section: 4

Presented by:

Alkım Bozkurt

Emirhan Yolcu

Sevda Sıla Yıldız

Özgür Akyaz

Bora Özkan

Table of Contents

Executive Summary

This report presents the Critical Design Review of the RF-based Antenna Tracking System (RFATS), a real-time tracking platform developed to maintain reliable wireless communication with mobile transmitters in GPS-denied or jamming-prone environments. Designed with adaptability, compactness, and responsiveness in mind, the system leverages directional RF sensing and mechanical alignment to preserve continuous signal lock between a receiver and a mobile RF source.

The system operates by estimating the direction of arrival of RF signals through phase difference analysis between two spatially separated directional receiver antennas. This estimation is translated into steering commands that adjust the antenna's orientation in both azimuth and elevation, enabling continuous tracking within an angular error margin of 5 degrees. The signal processing and control logic are implemented in C++, running on a Raspberry Pi platform.

Core architectural components such as the antenna configuration, direction estimation strategy, and two-axis mechanical tracking structure have been finalized. Early-stage tests confirmed the validity of the direction estimation method at short to medium ranges, and demonstrated mechanical responsiveness within required angular speeds and positioning tolerances. However, practical tests revealed increasing angle error in large-range movements, primarily due to servo motor limitations. To address this, a transition to stepper motors is under consideration for azimuth control. While this change is expected to improve tracking continuity, it introduces risks in mechanical integration and control stability that are acknowledged and evaluated in the relevant sections of this report.

The direction estimation module has also shown degradation in accuracy at longer distances, likely due to environmental interference and limitations in antenna separation. Planned mitigation strategies include refined antenna layout and noise-aware signal processing adjustments. Integration between the estimation output and mechanical actuation is ongoing, and full closed-loop operation is targeted for the next design phase.

In summary, RFATS has reached a mature design state, with validated subsystems and well-defined performance targets. Major system components are in place, and ongoing development focuses on mechanical optimization and full subsystem integration. Remaining technical risks are identified and addressed throughout this report. Upon completion, the system is expected to offer a flexible, interference-resilient tracking capability suitable for use in UAV communication, disaster response, and military scenarios.

Introduction

Background and Motivation

In modern communication systems, maintaining a stable wireless connection with mobile units is critical across a wide range of applications, from military operations to disaster response and UAV navigation. Traditional tracking systems often rely on GPS or explicit location data to maintain alignment between the transmitter and receiver. However, such dependencies become unreliable or entirely infeasible in GPS-denied environments or under intentional signal jamming. In these cases, communication stability becomes vulnerable, risking signal loss and degraded performance.

The RF-based Antenna Tracking System (RFATS) addresses these challenges by eliminating the need for location data, instead using RF signal sensing to estimate and track the direction of the transmitting unit. By calculating the steering angle based on signal characteristics such as phase differences, the system can dynamically adjust its antenna orientation to maintain lock on the source, even when the environment is noisy or unpredictable. This capability is especially relevant in remote operations where infrastructure may be unavailable.

From a technical perspective, the motivation behind RFATS is to combine robustness, modularity, and real-time responsiveness into a lightweight and cost-effective solution. Using software-defined radio (SDR) hardware and embedded processing, the system aims to meet strict angular accuracy and reaction time constraints, all while resisting interference and reducing reliance on fragile infrastructure. This approach reflects a growing demand in modern communication systems for adaptable and self-sufficient tracking mechanisms that can perform reliably under real-world constraints.

Problem statement

The core engineering problem addressed by this project is the real-time tracking of a mobile RF transmitter using only signal reception data, without any external positioning input. Specifically, the task is to rotate a directional receiver system to maintain optimal alignment with a moving source that emits RF signals, even in the presence of interference and environmental noise.

To meet its objectives, the system must dynamically rotate its receiver antenna to maintain accurate alignment with the moving RF source. This involves estimating the transmitter’s direction through signal processing — primarily phase difference analysis across two spatially separated directional receiver antennas — and translating this estimation into motor control commands for azimuth and elevation adjustment.

Key performance targets include an angular error of less than 5 degrees RMS, a tracking lock time under 3 seconds, and a minimum tracking speed of 15 degrees per second in the azimuth axis. The system must also remain functional in jamming environments with defined interference characteristics and should be able to continue estimating the general direction of the transmitter for up to 3 seconds of signal loss.

The problem requires developing a compact, modular, and cost-effective system that integrates signal estimation algorithms, directional RF front-end design, and mechanical actuation. It must perform reliably in changing and potentially hostile environments, without manual recalibration or dependency on fragile infrastructure. RFATS must autonomously adapt to the transmitter’s motion and signal behavior in real-time, ensuring stable and responsive operation under diverse conditions.

Scope and organization of the report

This report documents the current design status, validation efforts, and development plan of the project. It outlines the system architecture and its subsystems as they stand at the critical design stage, where most major design decisions have been implemented or are in the final stages of refinement. The report captures not only the finalized elements but also identifies areas where continued development, integration, or adjustment is anticipated. These points are discussed transparently, along with the risks they may introduce.

The report begins with a description of the overall system, including its block architecture, subsystem roles, and physical modeling efforts. Following this, the requirement analysis section traces customer needs to measurable performance objectives and system constraints. Subsystem compatibility and integration topics are then explored, including the communication and synchronization mechanisms that support cross-functional operation.

Design modifications since the Conceptual Design Report are highlighted in a dedicated section, including justifications for changes and evaluation of trade-offs. The test procedure and evaluation results section reflects current performance metrics obtained from individual and partial system tests. Resource management is addressed in terms of cost, power, and project scheduling. The report concludes with a summary of progress and outlines the key development steps leading to the final implementation.

Throughout the project, responsibilities were divided across team members based on technical background. Team members contributed to signal processing, RF front-end design, motor control systems, embedded software development, mechanical design, and system testing. The modular structure of RFATS allowed parallel development of subsystems, followed by coordinated integration phases. This report reflects the collective outcomes of those efforts.

Requirement Analysis

The RF-based Antenna Tracking System (RFATS) is developed to ensure robust, GPS-independent signal tracking by estimating the direction of arrival (DoA) of RF signals and aligning a directional receiver antenna accordingly. System and subsystem level requirements have been finalized and structurally defined based on the mission objectives and performance expectations established in the conceptual design phase. These requirements address functional, mechanical, electronic, and software components in a cohesive and interdependent framework, ensuring that the design meets all operational constraints under realistic use conditions.

At the system level, the platform must maintain a reliable line-of-sight (LOS) communication link with a moving RF source. To accomplish this, the tracking mechanism must rotate the receiving antenna in azimuth and elevation axes with a root-mean-square angular error below 5 degrees. The system must acquire and lock onto a new target within 3 seconds and maintain angular tracking speeds of at least 15 degrees per second in azimuth. These performance metrics are derived from use-case scenarios involving mobile transmitters such as UAVs, where high responsiveness and minimal latency are critical for maintaining data flow and signal integrity.

To fulfill these specifications, each subsystem has been developed around its respective requirements. The receiving antenna module features a hybrid arrangement of Yagi-Uda and Vivaldi antennas, combined via a real-time controlled RF switch to enable dual-axis spatial sampling using a reduced number of coherent SDR channels. Requirements for this module include sufficient antenna gain and directivity to operate reliably in the 2 GHz band, physical alignment within spatial tolerances to preserve phase coherence, and compatibility with the signal processing chain. Signal sampling is managed by an ADALM-PLUTO SDR, which is required to deliver accurate, time-aligned I/Q data with minimal phase distortion across switching cycles.

The processing module, centered around the Raspberry Pi 5, is responsible for executing high-resolution DoA estimation using Root-MUSIC and correlation-based alignment algorithms. Requirements for this module include the ability to interface with both the SDR and motor control systems via serial and USB protocols, ensure sub-second latency from signal capture to motor actuation command, and maintain computational integrity while running a

C++ implementation of the processing pipeline. The software must also sustain processing performance in the presence of environmental noise and jamming, including real-time detection of signal degradation and fallback behavior when signal is lost.

Mechanical requirements emerged from the need for precise and stable antenna orientation. On the azimuth axis, the NEMA 23 stepper motor—controlled via a TB6600 driver—is required to deliver 360° continuous rotation with microstepping capability, and maintain stability under payload torque. On the elevation axis, testing showed that the required holding torque exceeded 50 kg·cm. As a result, the finalized requirement for this axis is a motor with sufficient stall torque and braking performance to support bi-directional movement under gravitational loading. The entire motor subsystem must maintain responsiveness within the required angular speed and positioning tolerances, while avoiding overshoot and mechanical oscillation.

The control subsystem—implemented on an Arduino Uno—must interpret directional commands from the Raspberry Pi and generate precise PWM, direction, and enable signals for the motors. The controller must account for variable torque conditions, implement acceleration/deceleration profiles, and allow dynamic speed control without feedback oscillation. Communication between Pi and Arduino must be stable under load and capable of real-time command execution.

Power and thermal requirements dictate that all components must be supported by a 3-cell Li-Po battery upgraded to 2800 mAh, capable of simultaneously powering the Raspberry Pi, motor drivers, and RF frontend. This module must include voltage monitoring and over-discharge protection, with wiring designed for high current tolerance and thermal resilience. Total power consumption is expected to remain under 15 W during normal operation, allowing extended runtime within portable deployment conditions.

Finally, the overall system must maintain mechanical compactness and modularity. The final dimensions must remain within approximately 20 x 20 x 16 cm, and the system must be lightweight enough for easy transportation and field deployment. Physical integration of components must allow for maintenance, cable routing, and protection against mechanical shock and RF interference.

Taken together, these finalized requirements form a complete, relevant, and inclusive specification set that maps directly to the functional and environmental challenges of the RFATS system. Each subsystem requirement has been derived and validated through analytical modeling, simulation, and real-world testing, ensuring traceability to the project objectives and compatibility across all interfaces.

Overall System Description

Our project aims to develop an RF-based antenna tracking system designed to autonomously follow the direction of a UAV or RF-emitting target object in motion, using real-time received signals and implementing high-level signal processing techniques on them. The system's core objective is to maximize communication link quality by keeping a LOS (Line of sight) between the target and the receiving unit.

Most similar products in the market employ omnidirectional antennas to keep a LOS between a moving target and a stationary receiving station. Our product is designed in a way so that we can use directional antennas and maintain a communication link. The main advantage of our product is as we are using a directional antenna, we can provide a higher SNR, a better communication quality and a lower power consumption.

The highlight of our system is it utilizes the RF signals emitted from the target and eliminates the use of GNSS signals. This is particularly useful in scenarios where the tracked object (e.g., a UAV) is equipped with a radio transmitter and where traditional GPS-based tracking is not viable or sufficiently precise.

Top level architecture and Block Diagram

The product is designed in detail and with extensive engineering solutions to overcome challenges. Our system consists of several sub-blocks, each having a crucial role in the overall architecture. Figure 1 describes the overall system architecture, and design. We are using high-quality components, cutting-edge signal processing methods and integrating hardware and software in a harmonic, compatible approach to ensure robust and stable performance for our users.

There exist five main blocks in our system. These are Transmitting Antenna Module, Receiving Antenna, Processing, Motor and Power modules.
Transmitting Antenna Module: This module is the target source that emits RF signals. Although this sub-system is not within our product, the design of the transmitted signal waveform and specification of the transmitting antenna done by our team.

Receiving Antenna Module: This module can be considered as the RF frontend of the system. The RF signals are captured and fed through to processing unit.

Processing Unit: Signal processing is done within this module. Control and decision-making algorithms are operating here.

Motor Module: This module is responsible for providing the necessary mechanical mobility to the entire system.

Power Module: Power module ensures the powering of components and sub-systems continuously and reliably

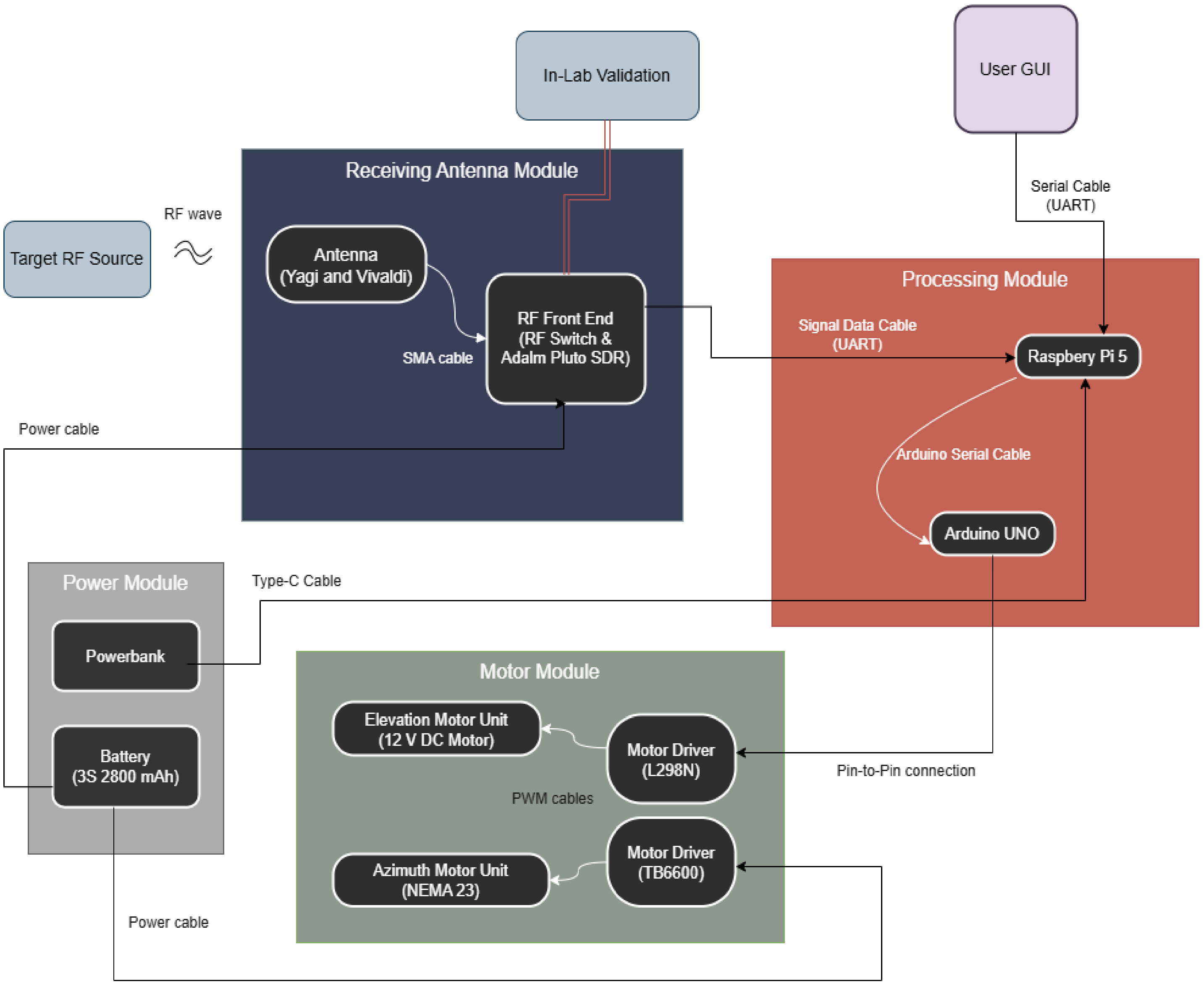


Figure 1 - System and subsystem level block diagram

All the sub-systems are configured in a way that the communication and data transfer between them are at the finest quality and fastest speed the meet the system requirements. Moreover, compatibility of these sub-systems was considered by our team during the design process. As mentioned above, the sub-systems, their connections and the algorithms running within the modules arranged in a way so that the system can operate with the best performance. The figure 2 is a flowchart that describes the flow of the operation within the overall system.

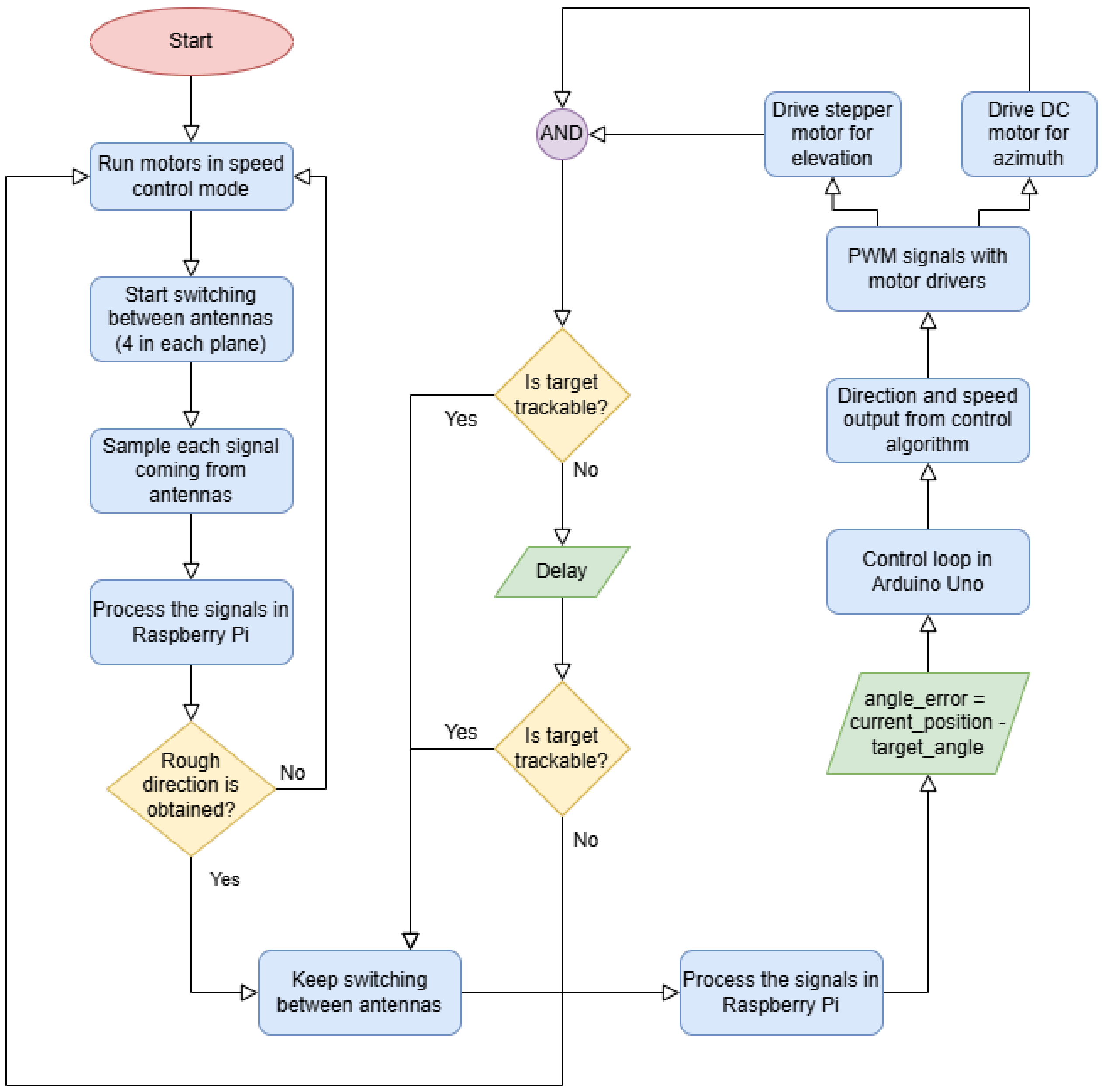

Start
Run motors in speed control mode
Start switching between antennas (4 in each plane)
Sample each signal coming from antennas
Process the signals in Raspberry Pi
Rough direction is obtained?
No
Yes
Keep switching between antennas
Process the signals in Raspberry Pi
angle_error = current_position - target_angle
Control loop in Arduino Uno
Direction and speed output from control algorithm
PWM signals with motor drivers
Drive stepper motor for elevation
Drive DC motor for azimuth
AND
Is target trackable?
Yes
No
Delay
Is target trackable?
Yes
No

Figure 2. Working Principle in Flowchart Diagram

The flow of the operation consists of many checkpoints, caution and protection mechanism and feedback loops to provide a stable and reliable operation. The main algorithm is also designed in a way so that the specifications and limitations of the hardware is considered and utilized in the most efficient way.

Technical Drawing and 3D Model

The one can observed 3D drawing of the finalized product in the figure 3.

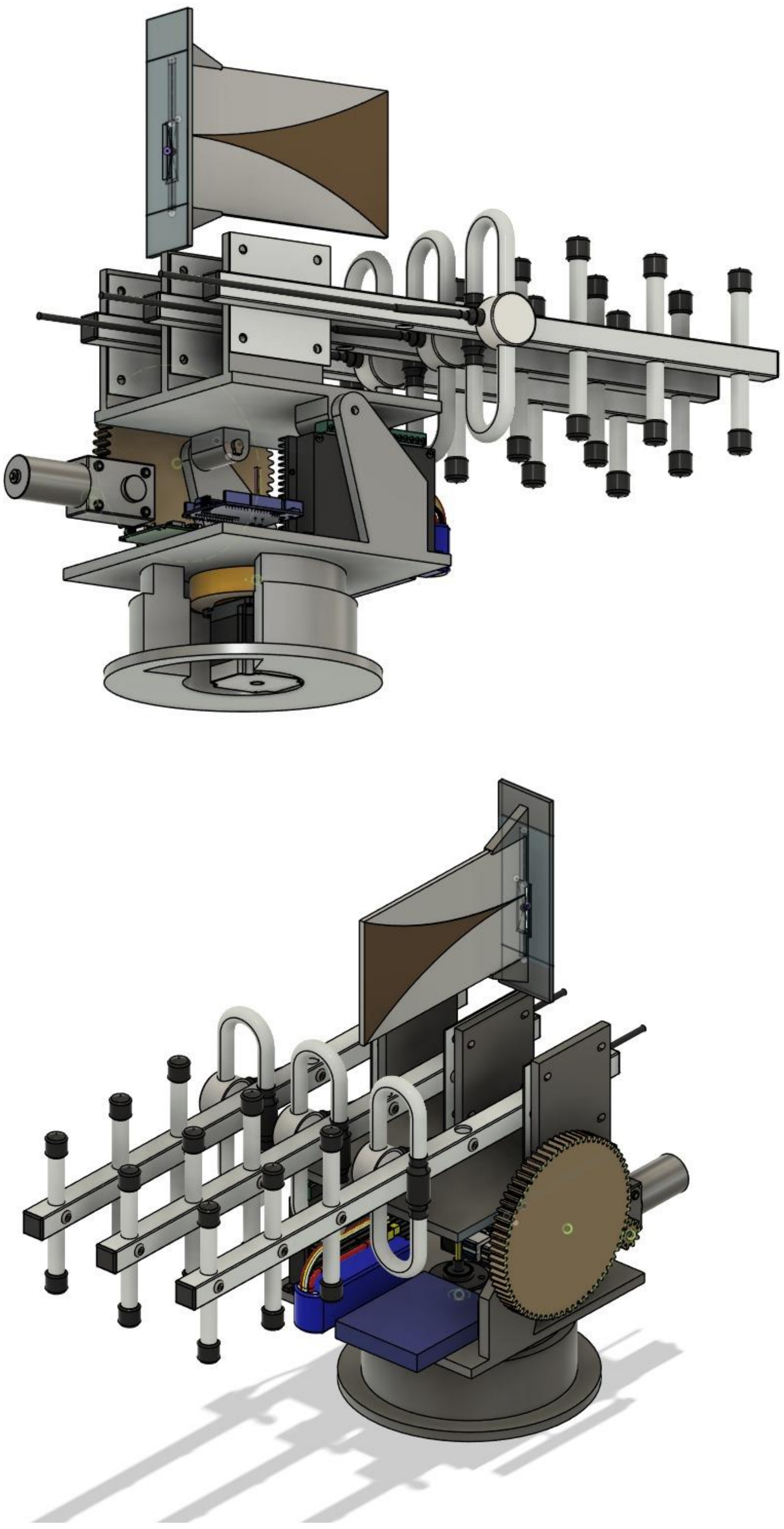


Figure 3 – 3D Model of the finalized product

Subsystem Compatibility and Integration

Overall system is a complex, large scale system. Therefore, integration this system requires high precision of communication and data transfer and high caution. In this chapter, the communication between subsystems, compatibility between them and how these sub-systems can form an operating overall system will be discussed.

Interface Analysis

Each subsystem is integrated through well-defined interfaces, with the Raspberry Pi 5 serving as the central processing hub that ties together the SDR, antennas, motor controls, and user interface. The Arduino Uno plays a critical role in real-time motor control, while the power and validation subsystems ensure that the system remains reliable and functional. By ensuring compatibility and efficient communication between subsystems, the overall system can autonomously track and maintain a line of sight with a moving RF-emitting target, delivering improved signal quality and power efficiency. To explain the signal and data transition, we define three main hubs: SDR, Raspberry Pi and Arduino Uno.

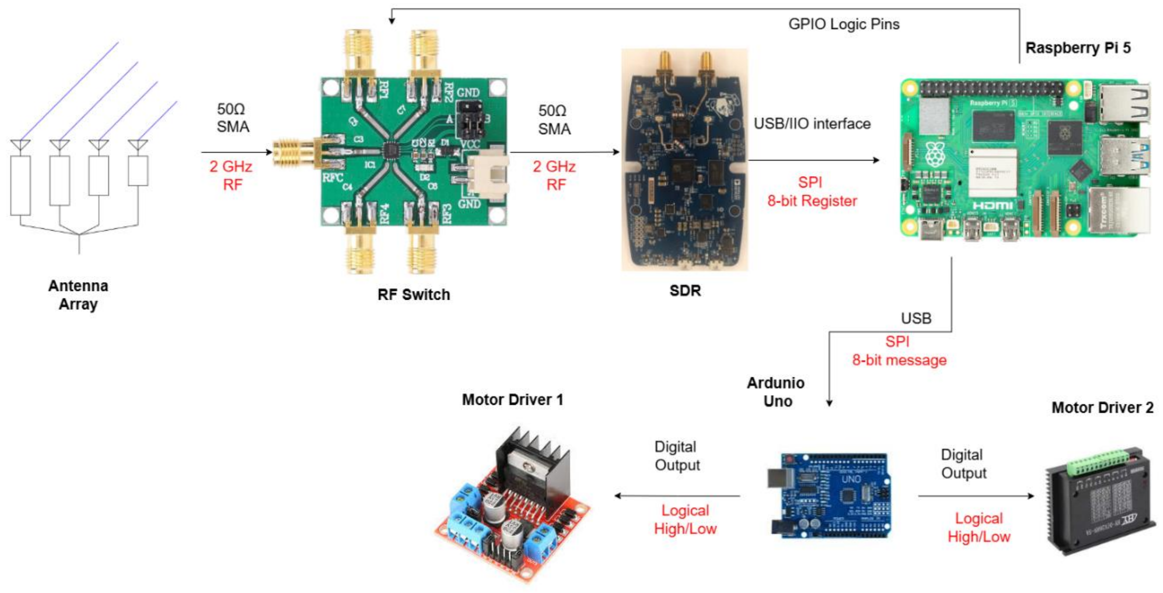


Figure 4 – Interface Diagram

<u>ADALM-PLUTO SDR</u>

RF signals captured from antennas are connected to an RF switch. Antennas are connected to the RF switch with SMA cables and switch is logically controlled by Raspberry Pi. The RF output of the switch is connected to ADALM-PLUTO SDR via 50Ω SMA cables.

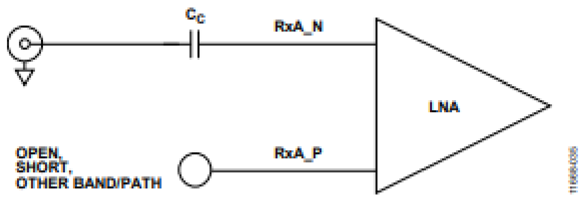


Figure 5 – 50Ω Rx interface of SDR

The SDR converts the RF signals into digitals signals thanks to its software-controllable RF components such as LNA, ADC, mixer etc[1]. Due to the limitations of the ADC, RF signal level is selected and processed register is treated cautiously. The digital signal is then loaded into an 8-bit SPI register. The AD9361 outputs 8-bit register from its mini-USB port and then this port is connected to Raspberry Pi 5.

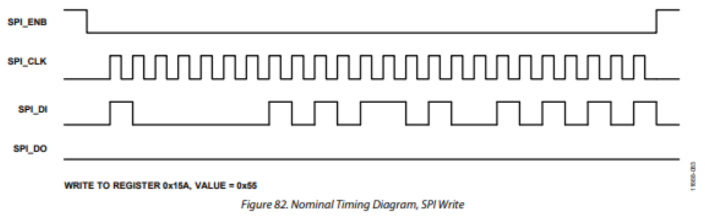


Figure 6 – SPI output of AD9361

Raspberry Pi 5

The digitized RF analog signal is fed to Raspberry Pi 5. The AD9361 chip provides a driver[2] developed by Analog Devices named “libiio”. Main script on Raspberry Pi 5 is selected as C++ to communicate with AD9361 chip from the operating system and execute signal processing algorithms converted from MATLAB. By using the mentioned driver, we can combine the digital signal fed to Pi 5[3] with our signal processing functions in a script.

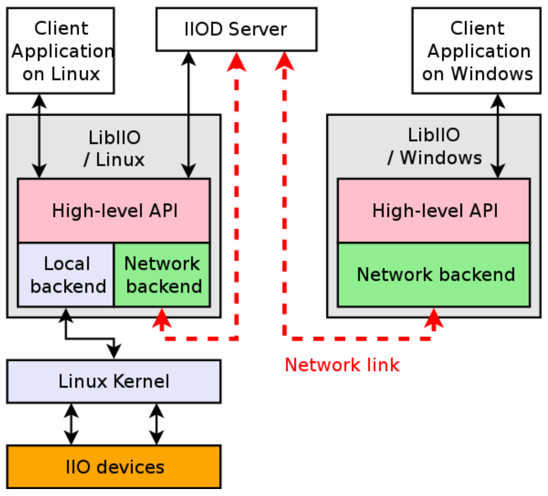


Figure 7 – Interface of IIO device and Operating System

Moreover, the RF switch located in the RF frontend is powered by Raspberry Pi 5, 5V and GND outputs. The logic pins of the switch is controlled by the GPIO pins of the Pi 5. The logic pins are connected to a 2:4 TTL decoder and have an interface as shown below.

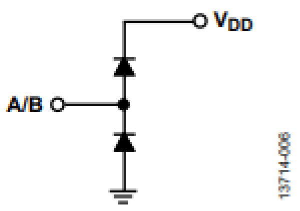


Figure 8 – Logic control interface schematic[4]

Finally, the Arduino UNO is also powered by Raspberry Pi 5 and communicating with it in a serial fashion. We are sending serial messages from our main script to the Arduino UNO which then executes the controlling algorithm for the motors. The Arduino is connected to Raspberry Pi 5 with USB interface and receives 8-bit customized serial commands sent from the main C++ script.

Arduino Uno

Arduino Uno is connected to Raspberry Pi 5 with USB. It receives a serial message from the Pi 5 in a SPI fashion with a specified baud rate. Then this message is processed by the algorithm within the Arduino. The algorithm controls the digital output pins of the Arduino. These digital pins are connected to the two motor drivers TB6600 and L298N. The control logic in the algorithm determines the states of these logic output pins connected to motor drivers and in this way, Arduino can control the speed, direction and movement of the step and DC motors.

With this system architecture we can convert the captured RF signal into meaningful interpretable temporary data and then we can control the entire system. The data flow and transition of the data can be seen in the figure 9.

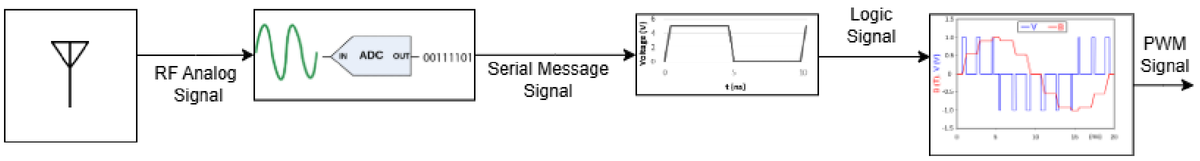


Figure 9 – Flow and transition of data

Hardware and Software Synchronization

The AD9361 chip inside has its own reference clock to ensure a coherent reception between two ports. The data fed to Raspberry Pi 5 is in SPI fashion which has its own clock. The main script in Raspberry Pi 5 is written in C++. We ensure a proper communication considering serial messages by providing necessary delays, callback and those kinds of implementations. Data logic voltage levels, fanout problems are always considered and overcame with the current design to ensure a reliable data transfer. Moreover, drivers and libraries are installed with proper release date to provide safe communication with peripherals along the system.

Subsystem Overviews

Transmitting Antenna Module

The transmitting antenna module consists of an ADALM-PLUTO SDR and an omnidirectional WIFI antenna that can work at a center frequency of 2 GHz. This module serves as a mobile RF source, transmitting a signal with a fixed center frequency to function as the target unit for the overall system. A primary design challenge for this module is designing the transmitted waveform to meet system-level requirements. The main algorithm used in the receiver module is Root MUSIC Algorithm. This algorithm assumes that all incoming signals are narrowband, so the phase difference across array elements can be modelled simply. Therefore, our prior choice of single tone sine works well with the algorithm. Thus, the transmitted signal has been chosen as a single-tone sinusoidal waveform, and it hasn't changed since the CDR. The operating frequency is carefully selected. Our Yagi-Uda antennas are actually cheap WIFI signal extenders and they are designed to work on the 2.4 GHz band. However, to not interfere with the WIFI band, we did S parameter measurements with VNA and found that our Yagi-Uda antennas are compatible with the 2 GHz band. Measuring that this is also compatible with our Vivaldi antenna at the top, it is decided to work on 2 GHz as a center frequency.

Receiving Antenna Module and Angle of Arrival Estimation

The updated passive direction-finding system uses a dual-channel Pluto Software-Defined Radio (SDR) in conjunction with a hybrid antenna configuration to estimate the direction of arrival (DoA) of radio frequency (RF) signals in both azimuth and elevation domains. This system is designed for enhanced angular resolution while maintaining a simplified hardware architecture, using a combination of sequential sampling, correlation-based signal alignment, and subspace-based spectral estimation techniques.

The antenna subsystem consists of one central Yagi-Uda[A1.1] antenna connected directly to Pluto SDR Channel 1 (common reference antenna), an SP4T RF switch, controlled in real time to alternate Pluto SDR Channel 2 between, Left Yagi-Uda antenna, Right Yagi-Uda antenna, Vivaldi antenna (top-mounted). This hybrid architecture enables spatial sampling in both the azimuth and elevation planes using a minimal number of RF receivers and ADCs.

The azimuth plane is scanned using a three-element Yagi-Uda phased array:

- Left Yagi (connected via switch)
- Middle Yagi (directly connected to Channel 1)
- Right Yagi (connected via switch)

Elevation angle estimation uses:

- The same middle Yagi (common element)
- A top-mounted Vivaldi antenna, connected via the SP4T switch

The azimuth antennas are aligned horizontally and form a linear uniform array along the x-axis. In ideal conditions, the phased antenna array structure requires coherent sampling to accurately estimate the direction of arrival. However, due to the excessive cost of adding additional coherent SDR channels, we followed a different approach using RF switches and two coherent receive channels of the ADALM-Pluto SDR. Unlike traditional systems that perform simultaneous sampling across all antenna elements using multiple synchronized channels, this design uses a sequential acquisition scheme to reduce hardware complexity.

In Azimuth sampling, first, the SP4T switch connects the left Yagi, and samples are captured from left and middle antennas. Then the SP4T switches to the right Yagi, and samples are captured from right and middle antennas. During operation, each switching cycle generates partial data corresponding to two antenna pairs. To reconstruct the full three-element coherent snapshot, a temporal alignment step is performed. Accurate Angle of Arrival (AoA) estimation requires coherent antennas. However, Sequential sampling introduces phase discrepancies due to non-simultaneous acquisition. To tackle this, the system implements calibration and signal alignment via cross-correlation techniques

The common antenna (middle Yagi) is sampled in all configurations and is connected to one receiver channel of the SDR all time. Using the repeated signal from the common antenna, cross-correlation is performed between adjacent acquisition windows to estimate the relative delays. Based on these offsets, a time-alignment correction is applied to simulate simultaneous signal snapshots across the antenna array.

After these operations, elevation sampling is performed. The SP4T switches to the Vivaldi antenna, and samples are captured from both Vivaldi and middle Yagi antennas. In this configuration, the temporal alignment is not applied since ADALM-Pluto receiver channels are enough for two coherent measurements.

Angle of Arrival Estimation

After alignment and sampling, each plane (azimuth and elevation) is treated as a linear sensor array. The system employs the Root-MUSIC (Multiple Signal Classification) algorithm for high-resolution AoA estimation.

For azimuth scanning, a 3-element linear array (left, middle, right Yagi antennas) is reconstructed. The signal covariance matrix is estimated from aligned samples. Root-MUSIC computes the eigen structure of the covariance matrix and identifies the signal and noise subspaces. The roots of the array polynomial on the unit circle are evaluated, and the angles corresponding to the minimum orthogonality to the noise subspace are selected as the AoAs. This provides high-resolution estimates, particularly when the signal-to-noise ratio is sufficient and inter-element spacing is appropriately chosen (typically half-wavelength).

For elevation scanning, a 2-element vertical array (Vivaldi and middle Yagi) is used. The same Root-MUSIC framework is applied in one dimension to resolve elevation angle. Due to the reduced number of elements, the elevation scan resolution is not as good as the azimuth but is sufficient for the requirement of the project.

Processing Module

The processing modules mainly consists of two units, which are Raspberry Pi 5 and Arduino Uno. Below, these two units are elaborated separately.

Raspberry Pi 5 (Main Processor) The Raspberry Pi 5, “work machine”, is responsible for capturing signal samples from the SDR/antenna module, processing them (using C++), and determining the target angle. Its main tasks are listed below.

- Perform angle-of-arrival and signal strength computations.
- Communicate with a GUI that provides real-time system monitoring.
- Send the calculated angle commands to the Arduino via a serial bus.

The digital signal processing methods which will be utilized for the methods described previously will run within a C++ implementation on Raspberry Pi 5. Our main method will be using a correlation method between the captured and sampled signals in order to extract the phase difference information. This phase difference information will be then converted to an interpretable data for the control system. The processing module works in the similar fashion as we explained in the CDR. Only difference is, now our processing algorithm is completely converted to C++. Therefore, all the signal analysis and serial communication will be done on C++ scripts.

Arduino Uno (Control Loop)

Arduino executes the low-level, real-time control of the motors. It receives angle targets from the Pi and drives the motor module (e.g., stepper or DC motors) to the correct orientation. Below are the main tasks of Arduino Uno.

- Interface with motor drivers through PWM, direction, and enable lines.
- Implement a control algorithm to ensure fast and precise aiming.

The communication between the Raspberry Pi and Arduino will employ UART interface using the Serial Bus. The Arduino can respond with status updates. For the communication between the RF Front End and Pi, communication will happen over USB. In our trials, we have seen the USB connection can create communication fast enough to provide a healthy control.

Why choose a Two-Controller Setup?

- **Real-Time Motor Control:** The Arduino's dedicated microcontroller architecture ensures consistent timing for motor pulses, avoiding Linux scheduling delays.
- **Task Separation:** The Pi handles computationally intense signal processing and the GUI, while the Arduino focuses on stable, low-level control.
- **Hardware Compatibility:** Arduino has abundant libraries and shields for motor control and sensor reading, speeding up development.
- **Fault Protection:** In case of an electrical fault from the motor module side, the worst scenario would be losing the Arduino which is extremely cheaper compared to Pi.

Since the CDR, we ensured that the SDR connection to the Raspberry will be over USB Serial Port. With another port being occupied by Arduino. We will still have enough USB ports for GUI communication over UART. Therefore, all the challenges we have foreseen in the CDR have been eliminated.

Motor Module

1. Motor Selection and Mechanics

The range of motion of NEMA 23 stepper motor is full 360° rotation. Its step size is 1.8° per full step. With microstepping we achieved a 0.225 step size. It allowed our motor to rotate in a more precise and controlled manner. The controller employs a linear acceleration and deceleration profile. Below are the several reasons behind this stepper motor choice.

- NEMA 23 typically offers high positional accuracy and holding torque (often in the 1–1.5 N·m or more range), making it top preference for larger antenna assemblies.
- No mechanical stops, enabling a full 360° sweep and a continuous motion.

In the horizontal rotation, we have designed and implemented a 3:1 ratio ring gear. The upper body is mounted on a 60mm inner diameter bearing. The motor will rotate the inner gear on the side. This setting will allow us to place slipper ring in the middle of rotation which transmits the motor and GUI cabling to ground level.

Since the CDR, we have replaced the servo motor with a 12V 18 RPM DC Motor. It has a holding torque of 49 kg/cm ensuring the movement capability on the elevation axis when combined with a 7:1 ratio gear. The range of motion is 0°–90° in this application. The active torque applied by the system changes as it rotates on the motor axis. A controller will overcome the problem of changing torque in this case. We still haven't finalized our idea on this axis rotation, further tests will allow us to confirm choosing a motor over another.

2. Motor Driver and Power Supply

The motor driver has changed since the CDR. TB6600 stepper motor driver has replaced the previously used L298N H-bridge driver due to heating issues at higher current levels. The TB6600 offers improved performance, especially in applications requiring precise control. It supports microstepping, which allows for smoother and more accurate motion control of the stepper motor (NEMA 23). This upgrade resolves the thermal limitations of the L298N and enables better handling of the motor's current demands. The advantages of TB6600 are as stated below:

- Microstepping support for higher precision and smoother motion.
- Better heat dissipation and current control capabilities, which are more suitable for the operational demands of the stepper motor used.

Now, the L298N will be used to drive the DC Motor on the side for the elevation movement control. While the L298N lacked features required for driving a stepper motor, it is still an effective choice for driving the 12 V DC motor on the side. The Arduino library for the L298N allows us to easily implement a control algorithm for driving the DC motor.

To support longer operation time, the 3-cell Li-Po battery has been upgraded from 1750 mAh to 2800 mAh capacity. It nominally supplies 11.1 V (3.7 V per cell), reaching up to ~12.6 V when fully charged. Both the stepper motor and the Draw power from this Li-Po source. Special attention is needed for current spikes, particularly under load or stress, which can significantly increase power consumption.

To ensure battery health and prevent over-discharge, a voltage tester module is attached to continuously monitor cell voltages throughout operation.

3. Control Logic and Speed Requirements

The speed requirement is at least 15°/s to track rapidly moving targets. We have also achieved similar results on the elevation module with a high ratio gear implementation.

Stepper motor control loops will function as listed below.

- The Raspberry Pi calculates the desired azimuth angle error and sends it to the Arduino.
- The Arduino's controller algorithm adjusts the step rate (and direction) for the motors.
- The position feedback will come from the antenna module. Therefore, we will have a speed control based on position feedback.

The horizontal motor control will implement a speed control with linear acceleration profiles. For the different sectors on the polar plane, we have different speed profiles. As the system rotates to the targeting angle, it will enter a sector with a smaller speed profile. The acceleration between the speed profiles will be linear. A more complex control algorithm was not needed since the speed requirement is already minimal. It also allowed the system to reach the target speed without an overshoot. With correct tuning of speed and acceleration profiles and the deadband, the target will easily be tracked with small occasional errors.

The 12V 18 RPM DC motor, known for its high holding torque, is well-suited for handling elevation adjustments with both stability and sufficient speed. Its rotational capability allows it to cover typical angular ranges within a practical time frame, while maintaining the force needed to hold position under varying loads.

The Arduino controls the motor through PWM signals, adjusting the duty cycle in real time based on the error. Additionally, the control logic accounts for torque variation during elevation changes (for instance, when the motor must work against gravity at steeper angles or stabilize under shifting loads). By dynamically adjusting the power input in response to changing torque demands, the system ensures consistent motion and prevents stalling or overshooting. Fine-tuning will achieve smooth acceleration and deceleration, avoiding oscillation while maintaining precise control throughout the motor's movement range.

Both motors should be sized to handle the weight of the antenna assembly plus any additional components (e.g., cables, protective housing). The NEMA 23 is often rated for moderate torque (e.g., 1.0–1.5 N·m) which is sufficient for rotating a small to medium antenna array if the assembly is balanced. DC motor can handle moderate loads (torque ~49 kg·cm). After a high number of trials, we could achieve a speed of 30 rpm, meaning 180°/s on horizontal rotation. This was achieved thanks to the ring gear and bearing combination.

Power Module

The power module will also supply energy to the RF switches in addition to the stepper motor and Raspberry Pi 5. Given the high current draw of the power module, thicker, heat-resistant cables will be used to ensure safe and efficient power delivery. This will minimize the risk of overheating and voltage drops, preserving the stability of the system during extended operation. To maintain continuous performance, the Li-Po battery and power bank levels must be regularly monitored and recharged as needed. Ensuring these power sources remain sufficiently charged will prevent interruptions in system operation. Additionally, careful attention must be given to the power cable connections to avoid loose or faulty wiring, which could disrupt the power flow or damage sensitive components. Overall, effective power management and proper cabling will be essential to maintaining system reliability and preventing power-related failures during operation.

Computer Controlled Signal Emulation and In-Lab Validation Subsystem

To evaluate and verify the functionality of the passive direction-finding system under controlled conditions, a laboratory-based signal emulation framework is integrated into the design. This subsystem allows rigorous testing of the azimuth and elevation estimation algorithms without relying on external over-the-air RF transmissions or physical antennas. The goal is to enable reproducible, high-fidelity simulations of angular trajectories by injecting known signals directly into the receive paths of the system.

In this test configuration, the antenna elements are replaced with direct coaxial cable connections to the Pluto SDR receiver and SP4T switch input ports. A calibrated signal generator is used to emulate the received RF wavefronts corresponding to desired AoA trajectories. The system components involved include ADALM-Pluto SDR Channel 1 connected via SMA coaxial cable to the central signal feed, representing the middle Yagi antenna. SP4T Switch Input ports are connected via cables to left channel feed, right channel feed and top channel feed. This setup allows software-controlled injection of signals that mimic the relative phase and amplitude shifts experienced by spatially distributed antennas for arbitrary source trajectories.

A graphical user interface (GUI) is developed to simulate a moving signal source across both azimuth and elevation planes. Users can determine 2D spatial trajectories, including linear paths, circular or elliptical motion. Based on the drawn path and antenna geometry, the GUI computes theoretical phase differences that would be observed across the virtual antenna elements. The GUI synchronizes signal emission across the relevant ports (left, right, top, middle) using the additional SDRs for transmission (signal generator). The signal generator applies these phase shifts to its outputs to simulate a wave arriving from a specific direction. For azimuth scanning, the signal generator injects signals with appropriate phase offsets into the left and right channels during their active switching periods. For elevation scanning (top, middle), the signal generator switches to the top path and sends signals corresponding to elevation profiles. The result is a time-varying signal injection sequence that corresponds to a moving RF source, emulating realistic AoA patterns.

Graphical User Interface

To enhance usability and ensure seamless interaction between the user and the system, a Graphical User Interface (GUI) is being developed using C#. This desktop application will serve as the primary interface for controlling and monitoring the device. Communication between the GUI and the embedded system will be established through a UART (Universal Asynchronous Receiver-Transmitter) serial connection, allowing for reliable, real-time data exchange.

The C# GUI will offer a range of essential functionalities, including:

- **Mode Switching**: Users can toggle between different operation modes, such as manual control, automated tracking, or calibration mode.
- **Calibration Initiation**: A dedicated interface will allow users to start calibration routines for motors or sensors, ensuring accurate system behavior.

The GUI will be designed with a clean, intuitive layout to ensure accessibility for users of varying technical backgrounds. Its UART-based communication ensures efficient command delivery and data retrieval, making it a reliable bridge between the user and the underlying hardware.

Compatibility with Requirements

Table 1: Compatibility Table

Requirement	**Design Decision**	**Compliance Justification**
Constant center frequency operation	Transmitter operates at 2 GHz with single-tone sine wave	Ensures narrowband assumption validity for Root-MUSIC; avoids Wi-Fi interference while staying within antenna operational range
Tracking lock-in within 3 seconds	Fast data acquisition via coherent subarray and fast processing time using C++ High constant speed mode when target is not detected	Using C++ enables fast processing time,which results in approximately 5 angle measurements per second. Constant speed mode allows the system to quickly scan the area and find a rough result.
RMS angular error < 5° after lock	Root-MUSIC algorithm with 3-element azimuth array and calibrated alignment Microstepping with motor driver	High-resolution AoA estimation with subspace methods provides fine angular resolution Microstepping has allowed motor to move with much smaller discrete steps providing more precision
Track target moving at ≥15°/s in azimuth	A ring gear design mounted on a bearing with 3:1 ratio	A speed of 180°/s is achieved in azimuth during speed tests

Resilience to single-source jamming (100 kHz, -80 dBm)	Narrowband signal with temporal correlation; Root-MUSIC focuses on spatial signal structure	Narrowband high-SNR processing reduces jamming impact; common antenna acts as a stable reference
Modularity & compactness	Using only 2-channel SDR and an RF switch reduce hardware count.	Ensures hardware simplicity while enabling spatial sampling in two dimensions

Discussion of Engineeing Trade-offs

Hardware Cost vs. Sampling Coherency: Coherent AoA estimation typically requires multiple synchronized receivers. This design uses a cost-effective alternative—RF switch and dual-channel SDR—with post-acquisition alignment via correlation. While this requires temporal calibration and may worsen the measurement slightly , it avoids the cost and complexity of multi-SDR coherence.
Azimuth Resolution vs. Elevation Capability: The system prioritizes azimuth accuracy by allocating three elements to azimuth and only two to elevation. This results in slightly lower elevation resolution, which is acceptable as the elevation requirement is less stringent.

Choice of Lesser Antenna: Choosing 1 additional antenna for the elevation reduces the antenna number but brings lesser precision in elevation measurements. Adding one more antenna may increase the precision but may conflict with modular design

Jamming Resilience vs. Signal Design Flexibility: A single-tone waveform is less flexible than wideband or spread-spectrum signals for hybrid approaches but maximizes SNR for the Root-MUSIC algorithm and allows easier correlation-based temporal alignment. It satisfies jamming constraints by being narrowband and coherent.

Compactness vs. Speed: The implementation of DC motor for elevation movement brings up a much more resilient system. But the DC motor is pretty bulky and heavy compared to the servo motor. For now, our choice is the DC motor but further tests will confirm the final decision.

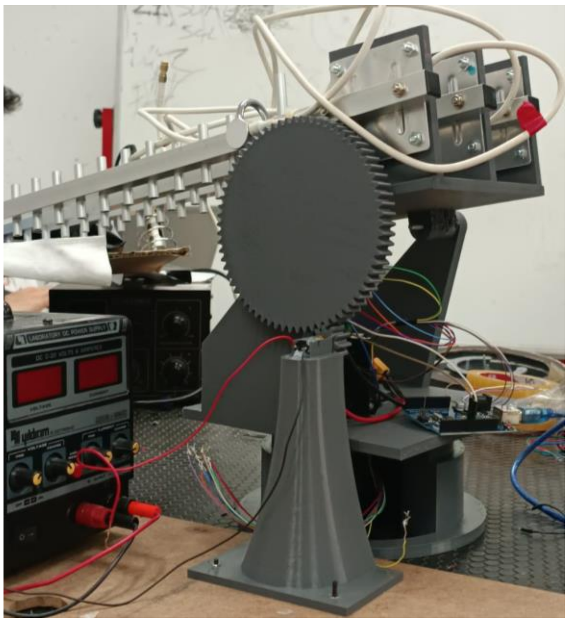
Simple Design vs. User-Friendliness: To build a user-friendly easy-to-use system, we had to come over some design complexities such as adopting a mechanical design to place a slipper ring in order to deliver a cable-connected reliable GUI connection. We also needed to come up with some tricky solutions in order to tackle the mechanical flaws such as disturbing noises etc.

Test Procedure

To evaluate the performance of our system in terms of accuracy and precision, we conducted several tests with different setups. Those tests showed us the limitations of our system and some major/minor problems. Furthermore, we were able to observe some parameters and specification of our system thanks to the results of those test. We modified and even replaced some components with respect to the deductions we have made according to the outcomes of various test. Below test is conducted to determine the accuracy of our angle of arrival estimation.

Test Setup

We fixed the target on a predefined position. Later, we swept the angle of the motor with respect to the normal plane between the target and receiver antennas. Initial angle was 0° which means receiver was perfectly aligned with the target antenna. We then swept the angle value of the motor with a step size of 3°. For every step, we measured the angle of arrival estimated by our system and we compared the results with our ground truth. The transmitted signal was a narrowband cosine signal at 2GHz carrier frequency with a power around +2dBm.



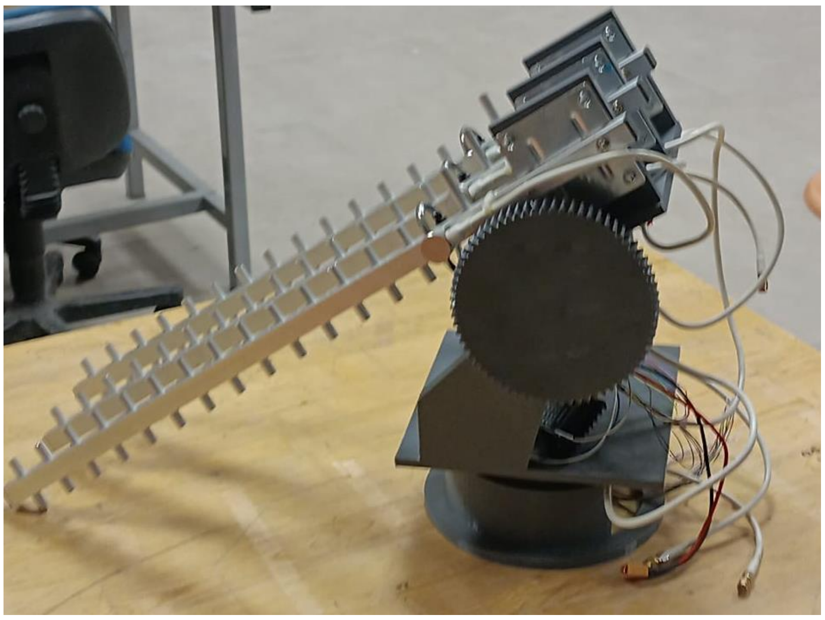


Figure 10-11.Test environment

To evaluate the rotation capabilities of the elevation axis, we conducted a series of preliminary tests using a DC motor controlled via an Arduino and a dedicated motor driver. The aim was to achieve smooth and stable motion from a position parallel to the ground (0°) up to +75°, with the motor capable of pausing accurately at any intermediate angle. While the setup functioned mechanically, the overall performance did not meet our expectations in terms of precision and torque stability. Therefore, no quantitative results were recorded during this phase.

In upcoming test plans, we aim to replace the current motor and control unit with components capable of delivering higher torque and improved motion control. These future evaluations will involve systematic angle measurements during both upward (+0° to +75°) and downward (+75° to 0°) movements. Our objective is to validate the system's response by comparing the target and actual angles during motion, enabling us to optimize the control logic and mechanical integration. Ultimately, this iterative testing and refinement process will help us approach the ideal performance required for robust elevation tracking.

Results and Evaluation

We have collected several measurements for each step in order to calculate our error in RMS. For the error calculation, we compared our ground truth angle value with angle of arrival estimation value. The plots regarding the measurement and ground truth can be seen below. One plot is for angle rotation in clockwise direction and the other is for counterclockwise direction.

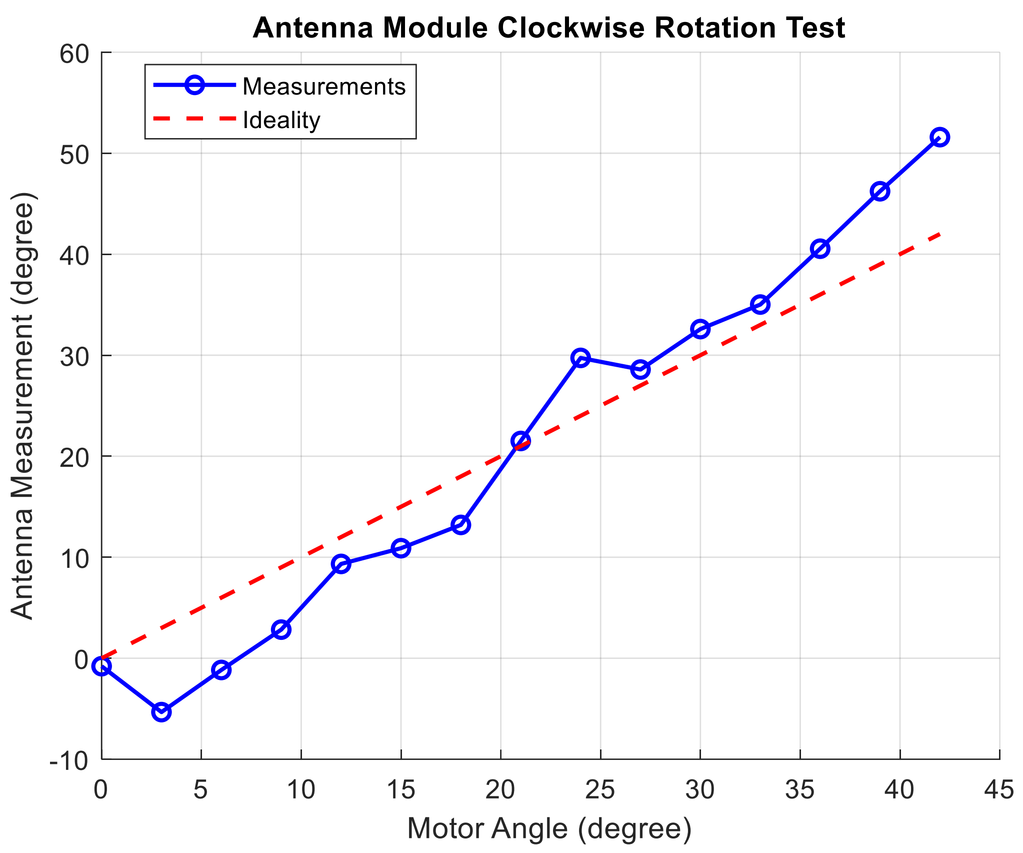


Figure 12-Test results

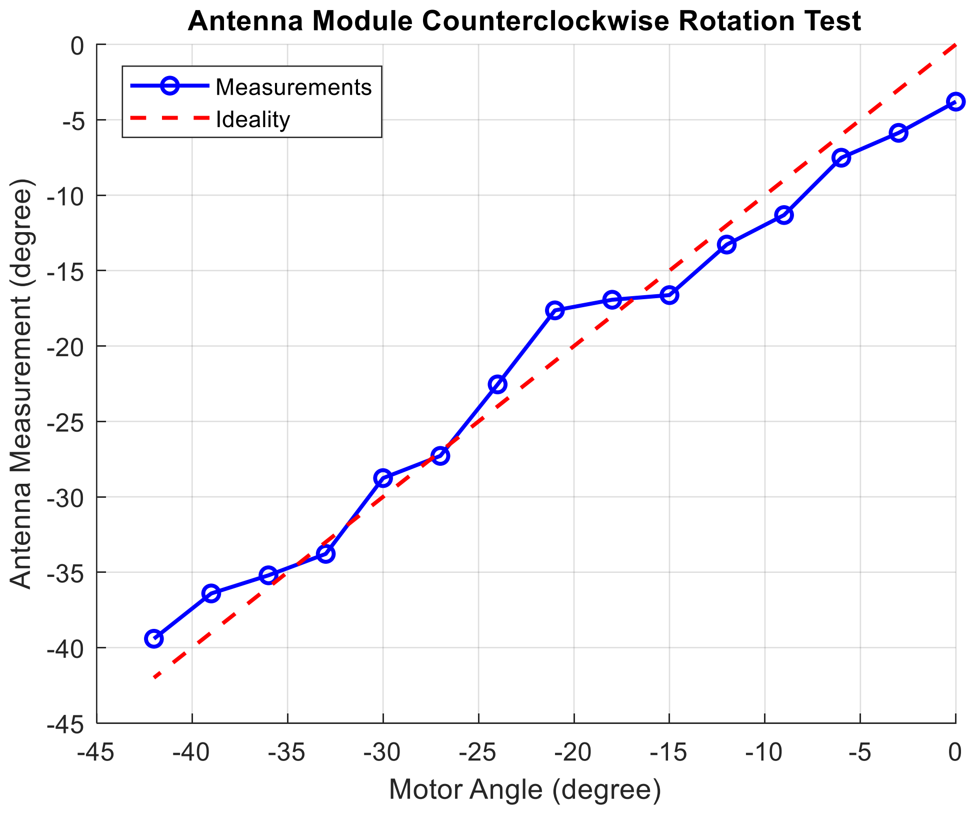


Figure 13-Test results

The test results indicate that our system is able to track the target with an error value. In general case we can see the trend of the measured angle is very similar to the nominal data. We also compared our calculated RMS error with the system requirements described at the beginning of this report. Our observations indicates that our system needs slight and soft signal processing to smooth out the AoA variations. The test data and results can be found in Appendix A1.2 and A1.3.

We concluded that for a stationary target, our AoA estimation is valid up to an angle of around 45°. However, since our system is going to rotate, the sufficient mobility will ensure that the nominal AoA will never exceed such a value. Therefore, the current performance is meeting the requirements of the project.

In one of the elevation axis rotation tests, we employed a DC motor with a holding torque of 14 kg·cm. However, this torque value fell significantly short of the system's estimated requirement of at least 50 kg·cm. To address this limitation, we integrated a gear mechanism with a 7:1 ratio to amplify the effective torque delivered to the elevation axis. Although this solution introduced a reduction in rotation speed, it did not pose a problem since the resulting motion remained within the acceptable speed limits defined for the system.

After torque amplification through the gear system, the motor successfully achieved upward rotation along the elevation axis. However, during downward movement, the motor's holding torque proved insufficient to maintain controlled descent. As a result, the system was unable to achieve precise or stable motion in the downward direction. We anticipate that this issue will be resolved in future test phases by replacing the current motor with a higher torque alternative. The updated setup is expected to provide adequate torque in both upward and downward directions, enabling full control and stability throughout the elevation range.

Resource Management

We divide our resource allocation into to main parts namely, cost and power allocation.

Cost Analysis

Table 2. Cost Analysis Table

Component	Single Cost	Amount	Total Cost
NEMA-23 Stepper Motor	14.56$	1	14.56$
12 V DC Motor with Gearmotor	10.49$	1	10.49$
Raspberry Pi 5	102.47$	1	102.47$

Arduino Uno	4.93$	1	4.93$
HMC7992 RF Switch	8.94$	1	8.94$
ADALM-PLUTO	200$	1	Subsidized
L298N Motor Driver	2.05$	1	2.05$
TB6600 Motor Driver	10.3$	1	10.3$
3D Filament	14.89$/kg	400 grams	5.95$
Slipper Ring	12.77$	1	12.77$
SMA Cables	5.12$/m	5 meters	25.6$
u.FL to SMA Cable	3.2$	1	3.2$
FR4 Copper Plaque	1.43$	1	1.43$
2.4GHz Yagi-Uda Antenna	9.75$	3	29.25$
3S Li-Po 2800mAh Battery	15$	1	15$
TOTAL ESTIMATED COST			246,94$

In general, our product breakdown consists of the motor structure, RF frontend components, structural frame components, processing components, and some middle components such as cables. After selecting a component, we compared the existing models in the market from cost and specifications point of view. We often bought a single sample to test and ensure the performance and compatibility of the component in our system before purchasing them as an whole.

During the product selection, as a team we paid attention to select the high-quality products which can affect the performance of the overall system. We bought a high-power processor to implement complex signal processing algorithms. Moreover, to handle the weight of the system and to ensure the desired mobility, we selected motors with sufficient figures of merit. Middle components such as cables or drivers were selected according to our budget. As a team, we adopted a mentality such that our main goal was not only to build an operating the system but also build it in a cost-effective approach.

Power Analysis

The main components that consume the significant power are Raspberry Pi 5 and motors. Motor unit is powered by a Li-Po battery. We separated the power units for the motor and other modules to ensure high power for the mobility unit and to not distort the processing power. The other modules are powered up by Raspberry Pi. Only the switch and SDR receive power at the receiver module as antennas are passive. We also power up the Arduino, which runs the control algorithms from Raspberry Pi

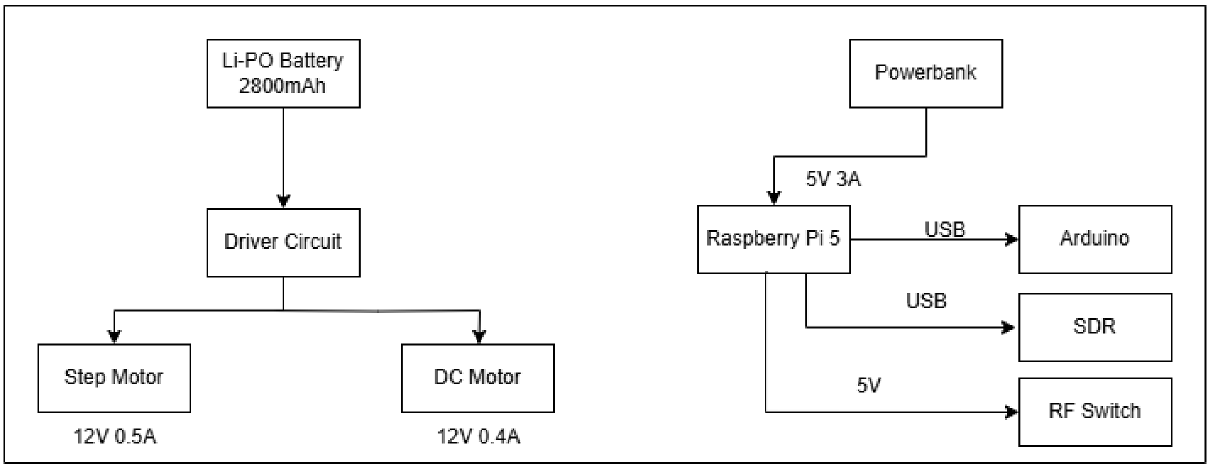


Figure 14. Power Distribution Diagram

Table 3. Maximum power consumption

Component	Max. Power Consumption
Step Motor	~24W
DC Motor	~24W
Raspberry Pi 5	~7.5W

Table 3 maximum power consumption of the components that consume the most power. However, the design will never use the maximum rated values for power consumption. Therefore, by estimating an average torque for motors, considering the ratings of our battery units we can estimate a power consumption.

Table 4. Average power consumption

Component	Max. Power Consumption
Step Motor	~6W
DC Motor	~4.8W
Raspberry Pi 5	~5W

In table 4 and Figure 14, the one can observe the average power consumption of the previously mentioned components. The average power consumption is calculated as ~12.25W. Datasheets[5],[6] of the components were utilized for this calculation.

For runtime calculations, we considered average values (a total current around 1A), and our calculations yielded that the current power configuration is sufficient for a stable and lasting operation of the system. Moreover, we have done numerous tests such as powering on the system and measuring the battery levels periodically. Those results show that the current selection of power units is meeting the demands.

$$\frac{2800\ mAh}{1A} = 2.8\ hours$$

Equation for nominal runtime calculation

Several tests were conducted to observe the power consumption. We measured how much time it takes for the battery to reach %75 level .

Table 5. Battery level test results

Test	Time
Idle Mode	100-120 Minutes
Operating Mode	75-90 Minutes

In the event of performance degradation of Raspberry Pi 5 due to unstable electrification, we are going to replace the power bank with a battery unit by using a proper voltage regulator (5V in our case). This minor modification is going to be done if such happens during further tests.

Schedule Analysis

An updated project schedule with a Gantt chart is provided to reflect the latest timeline and progress of the project. As of the Critical Design Review phase, the development is proceeding according to plan. While some phases required timing adjustments due to design changes and real-world constraints, these were managed without jeopardizing the final delivery.

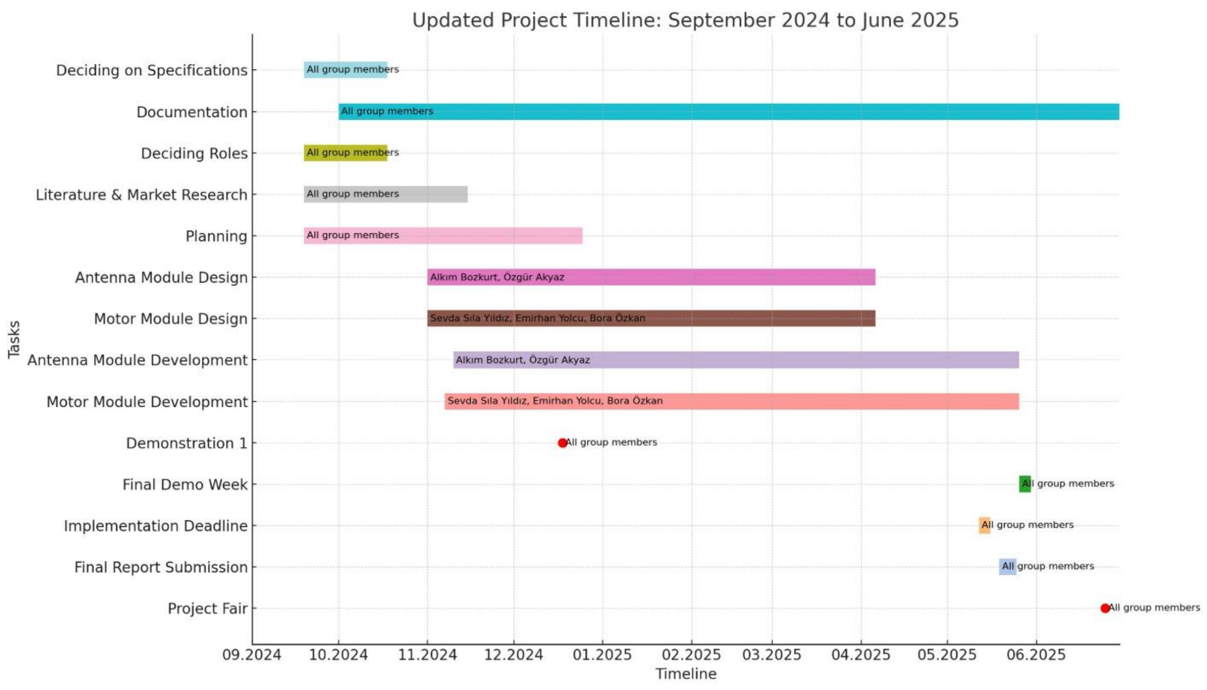


Figure 15. Time Schedule of Our Team Gantt Chart

There were periods when progress slowed temporarily, mainly due to component shipping delays and 3D printing lead times. Although these instances did not cause significant delays, they did limit available workdays. During those times, efforts were shifted toward software implementation, integration planning, and system documentation to maintain momentum.

One important change during this period was a shift in the signal processing implementation. The initial idea to use MATLAB for embedded processing was reconsidered, and the algorithms were restructured in a more lightweight environment. This adaptation ensured smoother integration with the rest of the system and better compatibility with the processing unit.

Hardware-related tasks generally progressed as planned. Major updates include replacing the initial servo-based rotation mechanism with a more robust stepper motor system for azimuth control, and a high-torque DC motor for elevation. These changes improved mechanical reliability and range of motion. The motor driver was also upgraded to a more capable unit to prevent thermal issues and enable microstepping. Power capacity was increased with a larger Li-Po battery to ensure stable performance during extended operation.

In parallel, the team developed an in-lab signal emulation system that allows controlled testing of direction-finding algorithms without needing actual RF transmissions. This setup, along with a desktop interface for real-time monitoring and control, contributed to smoother debugging and system evaluation.

Overall, even though a few technical areas required more attention than initially expected, the project remains on track. The CDRR process helped clarify the architecture and finalize key design decisions. With core modules implemented and integration underway, the team is confident in meeting the final project timeline.

Conclusion

The RF-based Antenna Tracking System (RFATS) presented in this Critical Design Review has evolved into a functionally mature and structurally integrated platform capable of addressing real-time tracking challenges in GPS-denied and interference-heavy environments. Through a combination of directional RF signal reception, advanced signal processing, and precise mechanical actuation, the system offers a self-contained solution for maintaining communication lock with mobile RF sources.

The final design integrates multiple subsystems—antenna arrays, signal processing modules, motor control units, and a centralized power supply—into a compact and modular architecture. Each component was selected and developed based on specific performance criteria, such as angular accuracy under 5 degrees RMS, tracking speeds exceeding 15 degrees per second, and rapid reacquisition capabilities within 3 seconds of signal loss. These metrics were validated through both analytical modeling and early-stage testing, demonstrating the system's responsiveness and adaptability under constrained conditions.

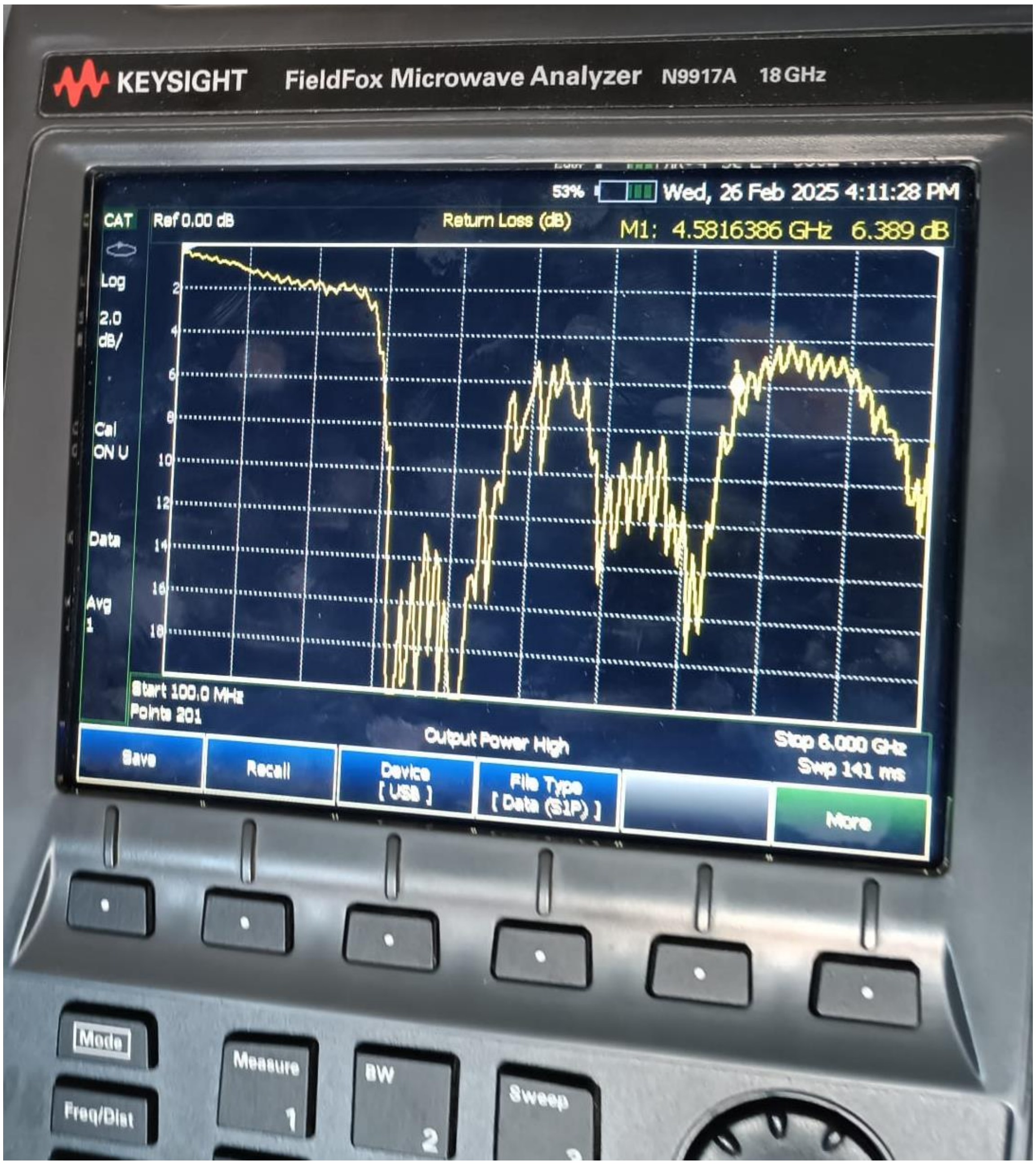
One of the most critical advancements since the conceptual phase has been the full transition of signal processing algorithms from MATLAB to C++, enabling real-time execution on a Raspberry Pi 5 and allowing for seamless integration with hardware-level control logic. This transition, along with the redesign of the mechanical tracking system—most notably the replacement of servo motors with a high-torque stepper and geared DC motor configuration—has improved system reliability and expanded its operational limits. Despite these upgrades, certain areas such as downward elevation stability still present challenges, which are being actively addressed through updated torque specifications and control strategies.

The subsystem compatibility, interface coherence, and synchronization mechanisms developed throughout this phase have laid the foundation for a robust and extensible tracking platform. All functional modules communicate through well-defined protocols, and power distribution has been optimized to support long-duration operation with appropriate protection and monitoring

In conclusion, RFATS now stands as a cohesive, real-world-capable system that meets its initial design goals and offers flexibility for future enhancement. The report comprehensively documents the development journey—from requirement formulation and architectural design to subsystem implementation and evaluation—and serves as both a technical reference and a roadmap for the final integration and testing stages that will follow in the remaining project timeline.

Appendix

A1.1 Scattering Parameters of 2.4GHz Yagi-Uda Antenna



A1.2

Table A1.2. Angle Error Test Table for Test 1

Ground Truth	Measured Value after RMS	RMS Error
0°	-0.788	0.79°
3°	-5.33°	8.33°
6°	-1.16°	7.16°
9°	2.83°	6.16°
12°	9.33°	2.67°
15°	10.89°	4.1°
18°	13.20°	4.8°
21°	21.5°	0.50°
24°	29.73°	5.73°
27°	28.58°	1.58°
30°	32.58°	2.59°
33°	35.01°	2.02°
36°	40.54°	4.55°
39°	46.23°	7.24°
42°	51.58°	9.58°

A1.3

Table A1.3. Angle Error Test Table for Test 1

Ground Truth	Measured Value after RMS	RMS Error
0°	-3.79°	3.79°
-3°	-5.86°	2.86°
-6°	-7.52°	1.51°
-9°	-11.31°	2.31°
-12°	-13.27°	1.28°
-15°	-16.62°	1.62°
-18°	-16.83°	1.06°
-21°	-17.64°	3.36°
-24°	-22.55°	1.45°
-27°	-27.28°	0.22°
-30°	-28.76°	1.24°
-33°	-33.79°	0.78°
-36°	-35.20°	0.79°
-39°	-36.40°	2.59°
-42°	-39.41°	2.59°

References

[1]: AD9361 User Manual https://www.analog.com/en/products/ad9361.html
[2]: https://wiki.analog.com/resources/tools-software/linux-software/libiio
[3]: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf
[4]: https://www.analog.com/media/en/technical-documentation/data-sheets/HMC7992.pdf
[5]: https://pages.pbclinear.com/rs/909-BFY-775/images/Data-Sheet-Stepper-Motor-Support.pdf
[6]: https://www.robotpark.com/12V-10Rpm-L-Geared-DC-Motor-KWL-FP