

NCRA • TIFR

National Centre for Radio Astrophysics
Student Training Program 2024 Report

ANTENNA POINTING AND CONTROL SYSTEM DEVELOPMENT FOR RFI

(GUI Integration for Real-Time Antenna Control and Monitoring)

July – August, 2024

By

SHREYA PANDEY

Under Supervision of

Dr. Shubhendu Joardar & Mr. Yogesh Gaikwad

**GIANT METERWAVE RADIO TELESCOPE
NATIONAL CENTER FOR RADIO ASTROPHYSICS
TATA INSTITUTE OF FUNDAMENTAL RESEARCH**

Narayangaon, Tal-Junnar, Dist-Pune

Acknowledgement

I would like to express my profound gratitude to Dr. Shubhendu Joardar and Mr. Yogesh Gaikwad for giving me the opportunity to work on the ANTENNA POINTING AND CONTROL SYSTEM DEVELOPMENT FOR RFI project. Their guidance and support have been valuable throughout this journey.

I am deeply grateful to the NCRA Director, Prof. Y. Gupta, the GMRT Dean, Prof. Ishwar Chandra, and the Telemetry and Operation Group Coordinator. I would also like to extend my heartfelt thanks to the NCRA and GMRT staff, including Mrs. Deshmukh, for their assistance and support during my project.

Working with GMRT has been a truly wonderful experience. I had the opportunity to learn about GMRT's work culture and interact with many experienced professionals. This interaction enriched my understanding and provided me with a broader perspective on the intricacies of antenna systems and their applications in astronomy. The insights I gained into the antenna pointing and control system for the RFI antenna were particularly valuable, allowing me to deepen my knowledge of the antenna pointing mechanism and the practical use of the antenna system.

This experience has significantly enhanced my technical skills and my understanding of real-world applications of the knowledge I have gained in the field of astronomy. I have developed a more comprehensive understanding of the complexities involved in antenna control systems and the importance of precision in the detection of RF signals. The hands-on experience with the antenna pointing mechanism has been instrumental in improving my problem-solving skills and my ability to apply theoretical knowledge to practical challenges.

In conclusion, I am immensely thankful for the opportunity to work on this project. The support and guidance from everyone at GMRT and NCRA have been crucial to my learning and growth. I look forward to applying the skills and knowledge I have gained to future projects and continuing my journey in the field of astronomy.

Shreya Pandey

Abstract

The primary objective of this project is to achieve precision and total control over an antenna system, with a focus on antenna pointing to detect RF signals effectively. This involves the integration of an Arduino microcontroller as the primary controller and Python for providing user inputs. The Arduino will receive commands specifying the desired angle and duration for which the antenna should move, and it will execute these commands with high precision. The precision of these movements is crucial for accurate signal detection, as the antenna must be positioned accurately to capture the desired RF signals.

The system's architecture involves a seamless interaction between hardware and software components. The Arduino is interfaced with motor drivers that control the motors responsible for adjusting the antenna's position. The motors must be capable of fine movements to ensure the antenna can be directed with high accuracy. On the software side, a Python-based user interface allows users to input the desired angle and time parameters. These inputs are then communicated to the Arduino, which processes the commands and drives the motors accordingly. The Python interface not only provides a user-friendly means of entering commands but also handles any necessary calculations and data processing to ensure the inputs are translated into precise movements.

A critical aspect of the project is the feedback mechanism to ensure precision. Sensors such as encoders or potentiometers may be employed to provide real-time feedback on the antenna's position, allowing the system to make necessary adjustments and corrections. This feedback loop is essential for maintaining the desired accuracy and ensuring the antenna remains pointed correctly throughout the specified duration.

Additionally, the system must be robust and reliable, capable of operating in various environmental conditions without significant degradation in performance. The integration of error-handling mechanisms and calibration routines will be crucial to achieving consistent and reliable operation. The overall design must also consider power management to ensure the system can operate for extended periods without interruption.

This project aims to develop a highly precise and controllable antenna system for detecting RF signals. By leveraging the capabilities of an Arduino controller and a Python-based input interface, the system will allow for precise adjustments to the antenna's position based on user-defined parameters. The success of this project hinges on the seamless integration of hardware and software components, robust feedback mechanisms, and meticulous attention to precision and reliability in the system's design and operation.

INDEX

Sr. no	TOPIC	PAGE No
1	Introduction to GMRT (Giant Metrewave Radio Telescope)	05
2	Existing System	08
3	Problems with Existing System	12
4	System Requirements	13
5	Proposed System	16
6	Design and Flow Diagram	17
7	Components Used	18
8	Circuit Diagram and Connections	24
9	RFI Antenna Control System Application	26
10	Observation with the Proposed RFI Antenna Control System	27
11	Conclusion and Future scope	29
12	References	30

Chapter 1

Introduction to GMRT (Giant Metrewave Radio Telescope (GMRT))

The Giant Metrewave Radio Telescope (GMRT) is a premier facility for radio astronomical research located near Pune, India. It is one of the largest and most sensitive radio telescopes in the world, operating in the meter wavelength range.



Fig. GMRT Antenna

- **Location:** Near Pune, Maharashtra, India.
- **Inception:** Commissioned in 2002.
- **Primary Purpose:** Observing celestial objects and phenomena at meter wavelengths.
- **Number of Antennas:** 30.
- **Type:** Parabolic dish antennas.
- **Diameter:** Each antenna is 45 meters.
- **Configuration:** Distributed in a Y-shaped array, similar to the Very Large Array (VLA) in the USA, with baselines (distances between antennas) ranging up to 25 km.
- **Frequency Range:** Operates in the frequency range of approximately 150 MHz to 1420 MHz.

1.1 Design and Construction:

The GMRT was designed and constructed by the National Centre for Radio Astrophysics (NCRA) of the Tata Institute of Fundamental Research (TIFR). The design leveraged local resources and expertise, making it a cost-effective project.

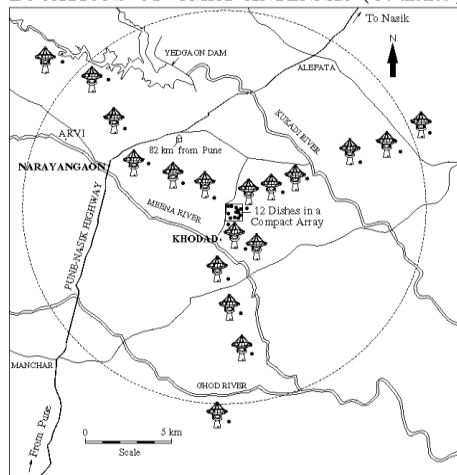
Unique Features: The GMRT is unique due to its large number of antennas and the wide range of frequencies it can observe. It is one of the few radio telescopes capable of operating at such low frequencies, which are crucial for studying various astronomical phenomena.

1.2 Antennas and Array Configuration:-

- **Antennas Design:** Each antenna is a fully steerable parabolic dish, 45 meters in diameter. The antennas are designed to be highly sensitive and can be pointed in different directions to observe different parts of the sky.
- **Materials:** The dishes are made of wire mesh, which is lightweight yet effective for reflecting radio waves at the GMRT's operating frequencies.

1.3 Array Configuration:

LOCATIONS OF GMRT ANTENNAS (30 dishes)



- **Y-Shape Layout:** The 30 antennas are arranged in a Y-shaped configuration, with each arm of the Y being 14 km long. This layout provides good coverage of the sky and helps in achieving high-resolution imaging.
- **Central Cluster:** Twelve antennas are located in a compact central array within a 1 km region, enhancing sensitivity for observations of small-scale structures.

1.4 Operations and Technology:

➤ Interferometry and Aperture Synthesis:

- **Interferometry:** The GMRT uses radio interferometry, where signals from pairs of antennas are combined to simulate a much larger telescope. This method enhances the resolution and sensitivity of the observations.
- **Aperture Synthesis:** By observing a source for an extended period and combining data from different antenna pairs, the GMRT can create high-resolution images of celestial objects, a technique known as aperture synthesis.

➤ Receivers and Signal Processing:

- **Receivers:** Each antenna is equipped with receivers that amplify and convert the radio signals into a form that can be processed digitally.
- **Signal Correlation:** The signals from all the antennas are brought to a central processing facility, where they are correlated. The correlator is a supercomputer that processes the vast amount of data to produce images and spectra.

➤ Data Analysis:

- **Software:** The GMRT employs advanced software for data calibration, imaging, and analysis. Tools like AIPS (Astronomical Image Processing System) and CASA (Common Astronomy Software Applications) are commonly used.
- **Research Output:** The processed data enable scientists to study a wide range of phenomena, from the early universe and galaxy evolution to the interstellar medium and transient events like pulsars and fast radio bursts (FRBs).

1.5 Scientific Contributions:

- **Cosmology and Galaxy Formation:** The GMRT has been instrumental in studying the large-scale structure of the universe and the formation and evolution of galaxies.

- **Pulsars and Neutron Stars:** The high sensitivity of the GMRT makes it ideal for discovering and studying pulsars, which are rapidly rotating neutron stars that emit regular radio pulses.
- **Interstellar Medium:** Observations of the 21 cm hydrogen line provide valuable insights into the distribution and properties of neutral hydrogen in the Milky Way and other galaxies.
- **Transient Phenomena:** The GMRT's wide field of view and sensitivity allow it to detect transient events like fast radio bursts (FRBs), which are brief but powerful radio pulses of unknown origin.

1.6 Upgrades and Future Prospects:

- **GMRT Upgrade (uGMRT):** The GMRT has undergone a significant upgrade to enhance its capabilities. The uGMRT project has improved sensitivity, extended the frequency coverage, and upgraded the receivers and digital back-end systems.
- **International Collaborations:** The GMRT collaborates with various international observatories and participates in global scientific projects, contributing to the worldwide effort in advancing our understanding of the universe.

The GMRT remains a vital facility for the global astronomy community, continuously contributing to groundbreaking discoveries and advancing our knowledge of the cosmos.

CHAPTER 2 EXISTING SYSTEM

2.1 EXISTING SYSTEM MODEL:

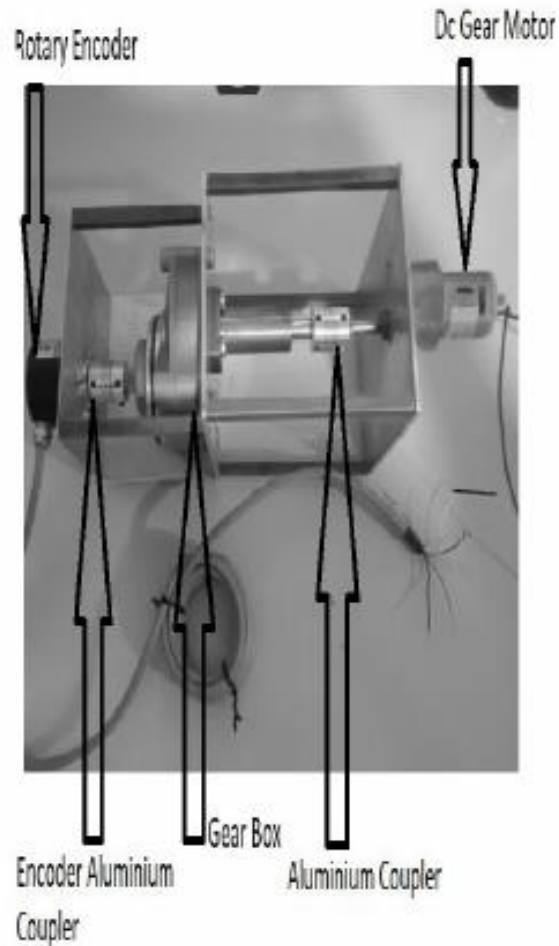
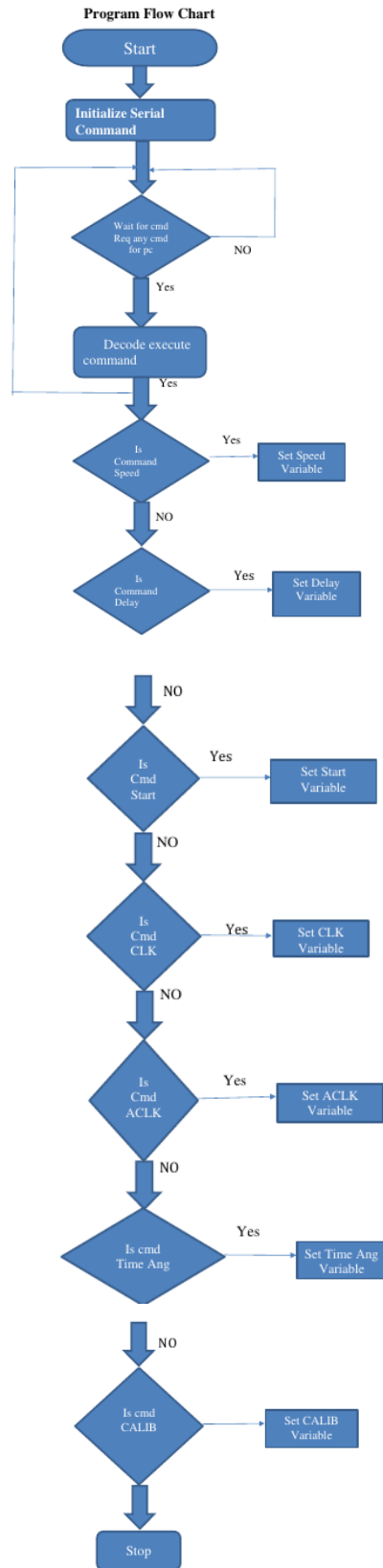


Fig. Refernece model

The above figure shows the Output of the previously made model. There is a trail base model which had same configuration as the real model. After selecting a direction from the arduino serial monitor. And using the serial monitor to give the direction of the rotation. This cyclic process is continued throughout the observation.

2.2 Flowchart for System Operation



2.3 OUTPUT FOR THE PREVIOUSLY EXISTING PROGRAM:

```
import time
import serial

arduino_port = 'COM5'

ser = serial.Serial(arduino_port, 9600, timeout=1)
time.sleep(2)

def send_command(command):
    time.sleep(0.9)
    #ser.write(" ".encode('utf-8'))
    ser.write(command.encode('utf-8'))
    response = ser.readline().decode('utf-8').rstrip('\r\n')
    time.sleep(0.3)
    return response

# Define commands
SPEED_COMMAND = "SPEED-150"
DELAY_COMMAND = "DELAY-0.4"
START_COMMAND = "START-999"
CLOCKWISE_CMD = "CLKWI-123"
ANTICLOCKWISE_CMD = "ANTIC-241"
STOP_COMMAND = "STOP-000"
ENCO_COMMAND = "ENCRD-001"
ROT_DEG_CLK = "RCDEG-30"
ROT_DEG_ACLK = "RADEG-100"
ROT_CALIB_CMD = "CALIB-000"

# Send and confirm SPEED command
response = send_command(SPEED_COMMAND)
print(f"Received from Arduino: {response}")
#time.sleep(0.5)

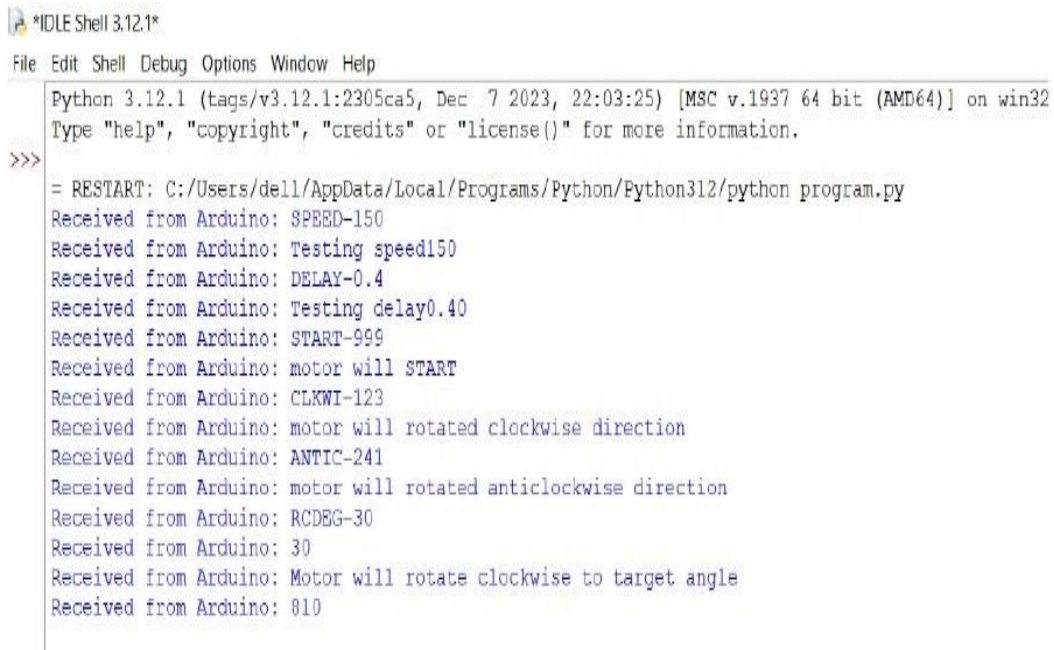
# Send and confirm DELAY command
response = send_command(DELAY_COMMAND)
print(f"Received from Arduino: {response}")
#time.sleep(0.5)

# Send and confirm START command
response = send_command(START_COMMAND)
print(f"Received from Arduino: {response}")
#time.sleep(0.5)

# Send and confirm CLOCKWISE command
response = send_command(CLOCKWISE_CMD)
print(f"Received from Arduino: {response}")
```



```
*IDLE Shell 3.12.1*
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/dell/AppData/Local/Programs/Python/Python312/11111.py
Traceback (most recent call last):
  File "C:/Users/dell/AppData/Local/Programs/Python/Python312/11111.py", line 6, in <module>
    ser = serial.Serial(arduino_port, 9600, timeout=1)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:/Users/dell/AppData/Local/Programs/Python/Python312\Lib/site-packages\serial\serialwin32.py", line 33, in __init__
    super(serial, self).__init__(*args, **kwargs)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:/Users/dell/AppData/Local/Programs/Python/Python312\Lib/site-packages\serial\serialutil.py", line 244, in __init__
    self.open()
          ^^^^^
  File "C:/Users/dell/AppData/Local/Programs/Python/Python312\Lib/site-packages\serial\serialwin32.py", line 64, in open
    raise SerialException("could not open port {}: {}".format(self.portstr,
crpyes.WinError()))
serial.serialutil.SerialException: could not open port 'COM5': FileNotFoundError (2, 'The system cannot find the file specified.', None, 2)
>>>
==== RESTART: C:/Users/dell/AppData/Local/Programs/Python/Python312/11111.py ====
Received from Arduino: SPEED-150
Received from Arduino: Testing speed150
Received from Arduino: DELAY-0.4
Received from Arduino: Testing delay0.40
|
```



```
*IDLE Shell 3.12.1*
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/dell/AppData/Local/Programs/Python/Python312/python program.py
Received from Arduino: SPEED-150
Received from Arduino: Testing speed150
Received from Arduino: DELAY-0.4
Received from Arduino: Testing delay0.40
Received from Arduino: START-999
Received from Arduino: motor will START
Received from Arduino: CLKWI-123
Received from Arduino: motor will rotated clockwise direction
Received from Arduino: ANTIC-241
Received from Arduino: motor will rotated anticlockwise direction
Received from Arduino: RCDEG-30
Received from Arduino: 30
Received from Arduino: Motor will rotate clockwise to target angle
Received from Arduino: 810
```

The previous system required users to input values through the serial monitor, which was inefficient and cumbersome. The rotation and direction values were hardcoded into the program, leading to a rigid operation where the motor continuously cycled through forward and backward rotations without the ability to pause or adjust.

Additionally, the system's reliance on serial communication created significant delays. Values had to be transmitted to the Arduino one at a time, necessitating pauses in the program to ensure proper reception and processing. This sequential transmission slowed down the system, compromising its responsiveness, especially in applications where timing and precision were critical.

Moreover, the lack of flexibility was a major drawback. Once the motor started its operation, there was no way to interrupt or modify the cycle in real-time, making it difficult to adapt to changing conditions or user needs. Any adjustments required reprogramming, which was impractical for most users.

These limitations underscored the need for a more dynamic and user-friendly approach to motor control, one that would allow for real-time adjustments, minimize delays, and provide greater control over the system's operation.

CHAPTER 3

PROBLEMS WITH EXISTING SYSTEM

The precision of an antenna's movement is critical for detecting RF signals accurately. Any error in the antenna program can lead to physical damage and operational failures. To achieve the necessary precision, all components of the antenna system, from the motors to the GUI, must function correctly and in harmony. Developing a code that synchronizes the antenna with a Python GUI is crucial, yet numerous challenges have been encountered.

Previously, the antenna system was controlled using the Arduino IDE, which proved to be unreliable. The code was tested using a reference model, which did not accurately replicate the real-world problems encountered during implementation. Consequently, several issues arose:

- 1. Inaccurate Rotation of the Antenna:** The antenna failed to achieve a full 360-degree azimuth rotation.
- 2. Lack of GUI for Motor Control:** Previously, there was no graphical user interface (GUI) to control the motor, complicating the control process.
- 3. No Physical Antenna Implementation:** The previous phase of the project did not include an actual antenna, which limited the scope of testing and validation.
- 4. Motor Control Issues:** When the code was implemented, the motor did not stop at the desired angle, indicating a problem with precision and control.
- 5. Platform-Related Errors:** The existing system, which ran on Windows, frequently encountered errors during file execution, necessitating a switch to Debian 12 with KDE Plasma desktop for better stability.
- 6. Serial Communication Problems:** There were interruptions and errors in the serial communication between the Arduino and the Python machine, disrupting the synchronization and control processes.

Given these challenges, it is essential to redesign the antenna system to have simpler circuitry to reduce potential complications. This includes developing a reliable and precise control system integrated with a Python-based GUI, ensuring smooth communication and control of the antenna for accurate RF signal detection.

CHAPTER 4

SYSTEM REQUIREMENTS

4.0. Software Specification:

- Operating System
- Programming Language and Software

4.1 Operating System:

Debian 12 (Bookworm) is a major update to the Debian GNU/Linux operating system, featuring enhanced performance, security, and a rich array of software packages. Released in June 2023, it includes over 11,000 new packages and updates to key applications such as GNOME 43 and KDE Plasma 5.27. The release emphasizes stability and security with improvements like enhanced Secure Boot support and timely security updates. Debian 12 is designed to offer a modern, versatile platform suitable for various use cases, from desktops to servers.

Key Features:

- **Software Packages:** Over 11,000 new packages; includes GNOME 43, KDE Plasma 5.27, and updated programming languages.
- **Security:** Enhanced support for Secure Boot; timely updates and patches from Debian Security Team.
- **Accessibility:** Improved installer, better hardware detection, and accessibility tools.
- **Performance:** Optimized for both modern and older hardware; includes Linux kernel 6.1 for better performance and power management.

System Requirements:

- **Minimum:**
- Processor: 64-bit(amd64) recommended a 32-bit (i386) supported
- RAM: 512 MB
- Storage: 10 GB
- Graphics: Basic VGA adapter
- **Recommended**
- Processor: Dual-core
- RAM: 4 GB
- Storage: 30 GB
- Network: Required for updates and additional packages

4.2 Programming Language and Software:

Spyder is an open-source Integrated Development Environment (IDE) tailored for scientific and engineering applications in Python. It provides a powerful code editor, an interactive IPython console, and tools for variable exploration and debugging. Spyder is designed to streamline data analysis and visualization, making it a popular choice among researchers and developers in scientific fields.

Purpose: IDE for scientific and engineering development in Python.

Features: Code editor, IPython console, variable explorer, documentation viewer
Debugging tools.

System Requirements for Linux:

Minimum: Processor (Intel/AMD), RAM 4 GB, Disk Space 100 MB, Python 3.6+

Recommended: Multi-core processor, RAM 8 GB, Disk Space 200 MB+, Python3.8+,
Additional packages like numpy, pandas.

Python:

Python is a high-level, interpreted programming language renowned for its simplicity and versatility. It features easy-to-read syntax, extensive standard libraries, and cross platform compatibility. Python supports various programming paradigms, including object-oriented and functional programming, making it suitable for web development, data analysis, and scientific computing.

Features:

1. Simple syntax
2. interpreted
3. versatile
4. extensive standard library
5. cross-platform
6. supports OOP and functional programming.

Key Libraries:

1. NumPy
2. Pandas
3. TensorFlow
4. Django.

Arduino IDE:

Arduino IDE is a software platform used for programming Arduino microcontroller boards. It offers a user-friendly code editor, compiler, and uploader, along with a serial monitor for debugging. The IDE simplifies the process of writing, uploading, and troubleshooting code on Arduino boards, making it accessible for both beginners and experienced users.

Features:

1. Code editor
2. Compiler
3. Uploader
4. serial monitor
5. library manager
6. board manager

Purpose:

1. Simplifies writing
2. Uploading
3. debugging code on Arduino boards.

- **Software Packages Used**

Python packages: PySerial is a Python library that encapsulates access for the Serial port. It provides backend support for serial communication allowing python backend support for serial communication allowing python scripts to read from and write to serial devices such as Arduino, serial consoles etc.

- **Troubleshooting:**

Permission Denied: If you encounter a permission denied error when accessing the serial port you may need to add your user to the `dialout` group:

```
^^^
```

```
sudo usermod -aG dialout $USER
```

```
^^^
```

Log out and log back in for the changes to take effect.

Serial Port Identification: To identify the correct serial port, you can use the `dmesg grep tty` command after plugging in your serial device.

CHAPTER 5

PROPOSED SYSTEM

Purpose:

This project report delves into the development and implementation of an advanced antenna pointing and control system designed specifically for enhancing RFI monitoring capabilities.

The core objective of this project is to design and develop a sophisticated system that enables precise directional control of an antenna. This is crucial for effectively locating and analyzing sources of RFI, which can otherwise obscure valuable data and disrupt critical communications.

Our approach integrates various electronic components and programming techniques to achieve seamless control over the antenna's orientation. The system utilizes an Arduino microcontroller as the central hub for managing the control mechanisms. This microcontroller communicates via serial communication to coordinate the operation of an L298N motor driver, which in turn controls the movement of the antenna through a series of gears.

To ensure accurate positioning, the system incorporates a 2500 PPR (pulses per revolution) optical encoder, which provides high-resolution feedback on the antenna's position. This feedback is crucial for precise adjustments and maintaining the antenna in the desired direction. The entire system is orchestrated through a custom-designed graphical user interface (GUI), which allows users to easily input and adjust directional parameters for the antenna.

The GUI, developed using Python programming, offers an intuitive interface for controlling the antenna's pointing direction. It integrates with the Arduino through serial communication, enabling real-time adjustments and monitoring of the antenna's position. The combination of these technologies ensures a robust and user-friendly system capable of meeting the demands of modern RFI monitoring.

This project represents a significant advancement in antenna control systems, leveraging a blend of electronics, programming, and mechanical components to provide precise and efficient RFI monitoring capabilities.

CHAPTER 6 DESIGN AND WORKFLOW DIAGRAM

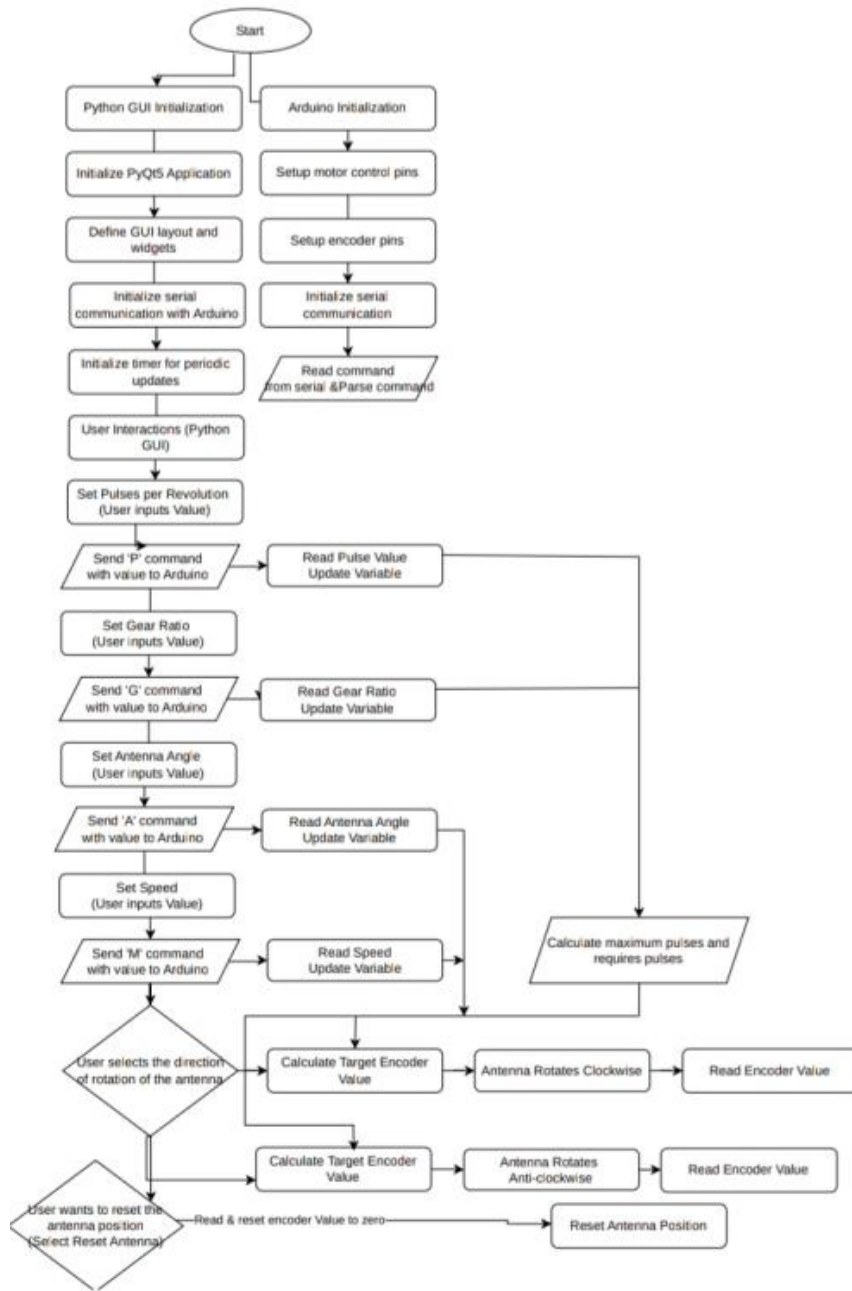


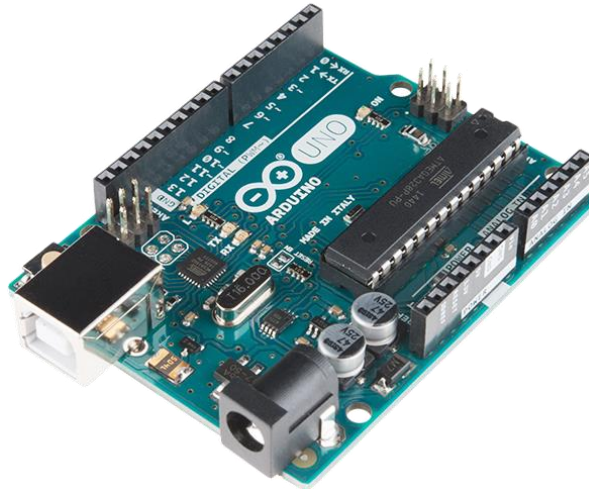
Fig. Workflow Diagram for the Proposed System

This diagram represents the workflow of the RFI Antenna Control System application. The process begins with initialization, where Python and Arduino components are set up, establishing the user interface and connection via PySerial. Following initialization, the Main Control Window is displayed to the user. The system then enters a continuous loop, executing tasks such as rotating the antenna to a specified position. During this phase, user interactions are managed, allowing the user to set parameters like speed and angle as needed. The process concludes with the termination of the application.

Chapter 7

COMPONENTS USED

1. Arduino UNO R3 (ATMEGA328P):



Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller.

Memory

- AVR CPU at up to 16 MHz
- 32 kB Flash
- 2 kB SRAM
- 1 kB EEPROM

Peripherals

- 2x 8-bit Timer/Counter with a dedicated period register and compare channels
- 1x 16-bit Timer/Counter with a dedicated period register, input capture and compare channels
- 1x USART with fractional baud rate generator and start-of-frame detection
- 1x controller/peripheral Serial Peripheral Interface (SPI)
- 1x Dual mode controller/peripheral I2C
- 1x Analog Comparator (AC) with a scalable reference input
- Six PWM channels
- Interrupt and wake-up on pin change

ATMega16U2 Processor

- 8-bit AVR® RISC-based microcontroller

Memory

- 16 kB ISP Flash
- 512B EEPROM
- 512B SRAM
- debugWIRE interface for on-chip debugging and programming

Power: 2.7-5.5 volts

2. 12 Volt Johnson Geared DC Motor:



10RPM 12V DC Johnson high torque geared motors for robotics applications. It gives a massive torque of 120Kgcm. The motor comes with metal gearbox and off-centered shaft.

Features:

- 10RPM 12V DC motors with Metal Gearbox and Metal Gears
- 18000 RPM base motor
- 6mm Dia shaft with M3 thread hole
- Gearbox diameter 37 mm.
- Motor Diameter 28.5 mm
- Length 63 mm without shaft
- Shaft length 30mm
- 180gm weight
- 120kgcm Holding Torque
- No-load current = 800 mA, Load current = upto 7.5 A(Max)

3. Pro-Range 2500 PPR ABZ 3-Phase Incremental Optical Rotary Encoder



A rotary encoder is a type of position sensor which is used for determining the angular position of a rotating shaft. It generates an electrical signal, either analog or digital, according to the rotational movement.

Orange 2500 PPR Incremental Optical Rotary Encoder is a hi-resolution optical encoder with quadrature outputs for increment counting. It will give 10000 transitions per rotation between outputs A and B. whereas the Z phase will produce one transition per rotation. A quadrature decoder is required to convert the pulses to an up count. The Encoder is built to Industrial grade.

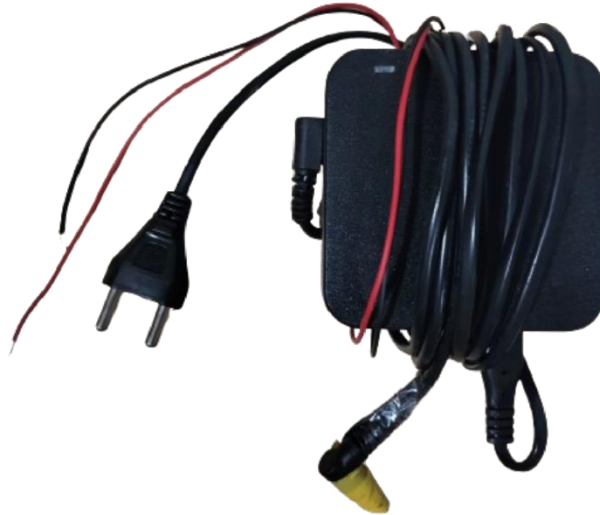
The Encoder comes with a Standard 2mt long cable which can be extended with extra cable if needed.

Phase A	White	Quadrature encoded output A
Phase B	Green	Quadrature encoded output B
Phase Z	Yellow	Quadrature encoded output Z
VCC	Red	VCC should be connected to +ve 5V of supply
GND	Black	The ground should be connected to negative the supply
Shield	Golden	The shield should be connected to GND

Features:

- High cost-efficient advantages.
- Incremental rotary encoder internal adopts ASIC devices
- High reliability, long life
- Anti-jamming performance
- Small size, lightweight, compact structure
- Easy installation
- Stainless steel shaft, High resolution, High quality, line interface with waterproof protection

4. 12V 5amp power supply:



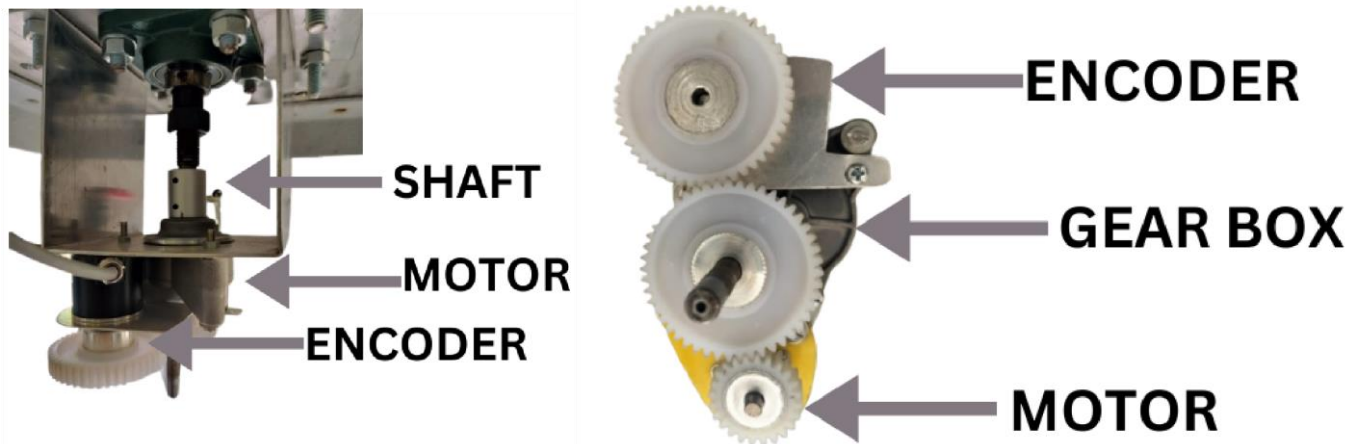
Power your devices efficiently with our 12 Volt 5 Amp Adapter, designed to deliver reliable performance for a wide range of applications. This adapter provides a stable 12V output at 5 Amps, ensuring consistent power for your electronics.

Key Features:

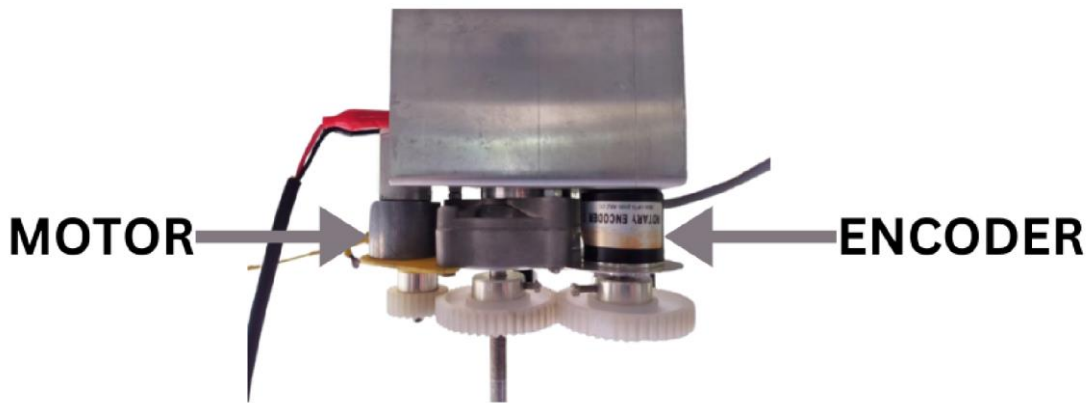
- **Model Name:** 12 Volt 5 Amp
- **Connector Pin Type:** 2.5mm x 5.5mm, perfect for compatibility with various devices.
- **Power Input:** 220V, suitable for use with standard household outlets.
- **Output Current:** 5 Amp, offering ample current for power-hungry devices.
- **Cable Length:** 5 meters, giving you flexibility in placement and use.
- **Short Circuit Protection:** Built-in safety feature to protect your devices from potential damage due to short circuits.

Model Id	• 12 Volt 5 Amp Adapter/ Power Charger
Model Name	• 12 Volt 5Amp
Connector Pin Type	• 2.5mm x 5.5mm
Power Input	• 220V
Output Current (A)	• 5Amp A
Cable Length (m)	• 5 m
Short Circuit Protection	• Yes

5. Gear Box:



Fig, Gear box for Antenna rotation



The above shown gear box has a 12 Volt Johnson DC Motor, 2500 PPM Optical Incremental Encoder, Gears, and a shaft.

Specification:

Motor to Shaft ratio = 5.8

Encoder to Shaft Ratio = 1.12

Calculating the Angular Speed in the Degree per Seconds:

To calculate the angular speed in revolutions per minute (RPM) using PWM (Pulse Width Modulation) and the time it takes to rotate 90 degrees, you can follow these steps:

Step 1: Calculate the Angular Speed in Degrees per Second

1. Given:

- Time for 90 degrees= t_{90} seconds

2. Calculate Degrees per Second:

- Angular Speed in Degrees per Second= $90/t_{90}$

Step 2: Convert Degrees per Second to Revolutions per Minute (RPM)

1. Convert Degrees per Second to Revolutions per Second:

- $\text{Revolutions per Second} = \text{Degrees per Second} / 360$
- $\text{Revolutions per Second} = 90 / t_{90} \times 360$

2. Convert Revolutions per Second to RPM:

- $\text{RPM} = \text{Revolutions per Second} \times 60$
- $\text{RPM} = 90 \times 60 / t_{90} \times 360 = 90 / t_{90} \times 60$
- $\text{RPM} = 15 / t_{90}$

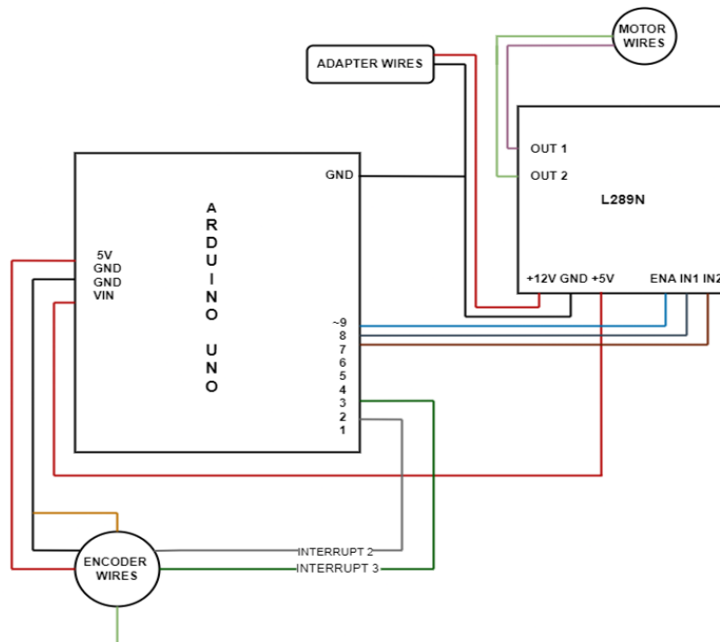
Final Formula:

$$\text{RPM} = 15 / t_{90}$$

Where t_{90} is the time in seconds to complete a 90-degree rotation.

CHAPTER 8

CIRCUIT DIAGRAM AND CONNECTIONS



Circuit Description

Components and Connections:

1. Arduino Uno:

- Digital Pins:
 - Pin 3: Connected to the white wire (Pulse A) of the encoder.
 - Pin 4: Connected to the green wire (Pulse B) of the encoder.
 - Pin 8: Connected to IN1 of the L298N motor driver.
 - Pin 7: Connected to IN2 of the L298N motor driver.
 - Pin 9: Connected to the ENA (Enable) pin of the L298N motor driver.
- Power Pins:
 - 5V Pin: Connected to the +5V input of the L298N motor driver.
 - GND Pin: Connected to the ground of the power supply adapter.

2. Encoder:

- Pulse A (White Wire): Connected to Arduino Pin 3.
- Pulse B (Green Wire): Connected to Arduino Pin 4.
- +5V (Red Wire): Connected to the +5V pin on the Arduino.
- Ground (Black Wire): Connected to the GND pin on the Arduino.

3. L298N Motor Driver:

- Motor Output Pins:
 - Motor 1: Connected to the motor terminals.
 - Motor 2: Connected to the motor terminals.
- Enable Pin (ENA): Connected to Arduino Pin 9 for PWM control.
- IN1: Connected to Arduino Pin 8.
- IN2: Connected to Arduino Pin 7.
- +5V Input: Connected to the +5V pin on the Arduino.

4. Power Supply:

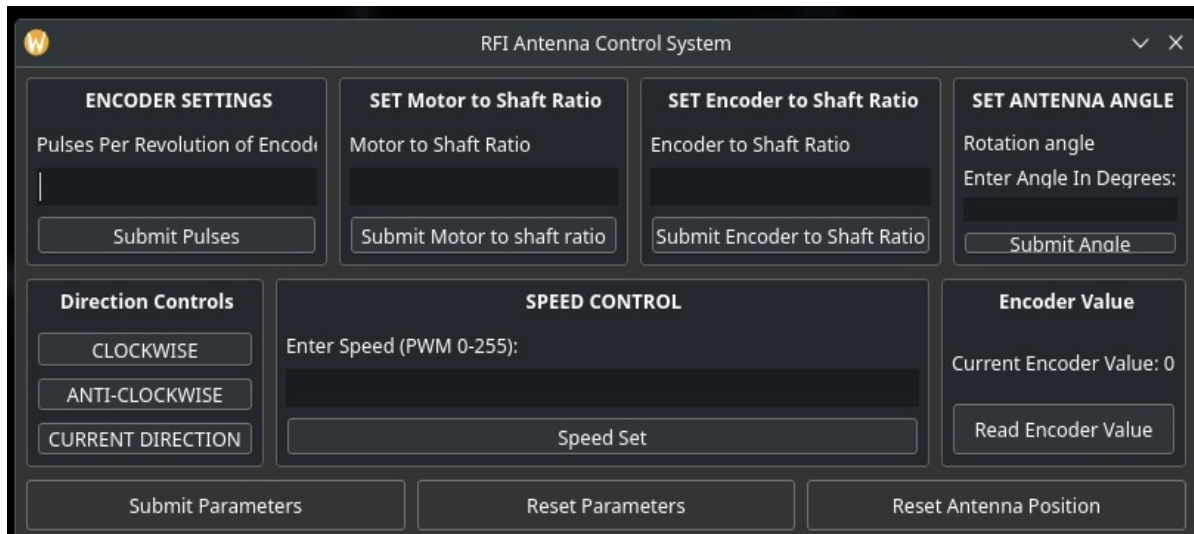
- 12V Adapter:
- +12V Output: Connected to the motor power input of the L298N motor driver.
- GND Output: Connected to the GND pin on the Arduino to ensure a common ground.

Operation:

- The encoder's Pulse A and Pulse B signals are used by the Arduino to monitor the motor's rotation and speed.
- The Arduino controls the L298N motor driver through digital pins to manage the motor's direction and speed.
- PWM signal from Arduino Pin 9 is used to control the motor's speed via the L298N motor driver's enable pin.
- The 12V power supply provides necessary voltage to the motor driver, while the Arduino is powered through its own USB connection or an external 5V source, which is also connected to the L298N.

CHAPTER 9

RFI ANTENNA CONTROL SYSTEM APPLICATION



Encoder Configuration Panel: This section allows precise configuration of the encoder parameters. The encoder count is specified here, set at 2500 pulses per revolution (PPR). The motor-to-shaft ratio is configured to 5.8, enabling the system to translate motor revolutions into corresponding shaft rotations. Additionally, the encoder-to-shaft ratio is set at 1.12 to ensure accurate measurement and control of angular displacement.

Antenna Angle Setting: This field requires the user to input the desired angular displacement for the antenna. This input directly dictates the target position for the antenna, enabling precise pointing based on the calculated ratios.

Direction Control Panel: This panel facilitates the directional control of the motor's rotation. The user can select either clockwise or counterclockwise rotation, depending on the operational requirements. The "Current Direction" button displays the current motor rotation direction in the Serial Monitor panel, providing real-time feedback for monitoring.

Speed Control Panel: The PWM (Pulse Width Modulation) value is entered here, with a valid range between 60 and 255. This range has been determined based on prior motor performance observations, ensuring optimal motor speed and response.

Encoder Value Display: This panel continuously monitors and displays the current encoder value, reflecting the real-time rotational position of the motor shaft.

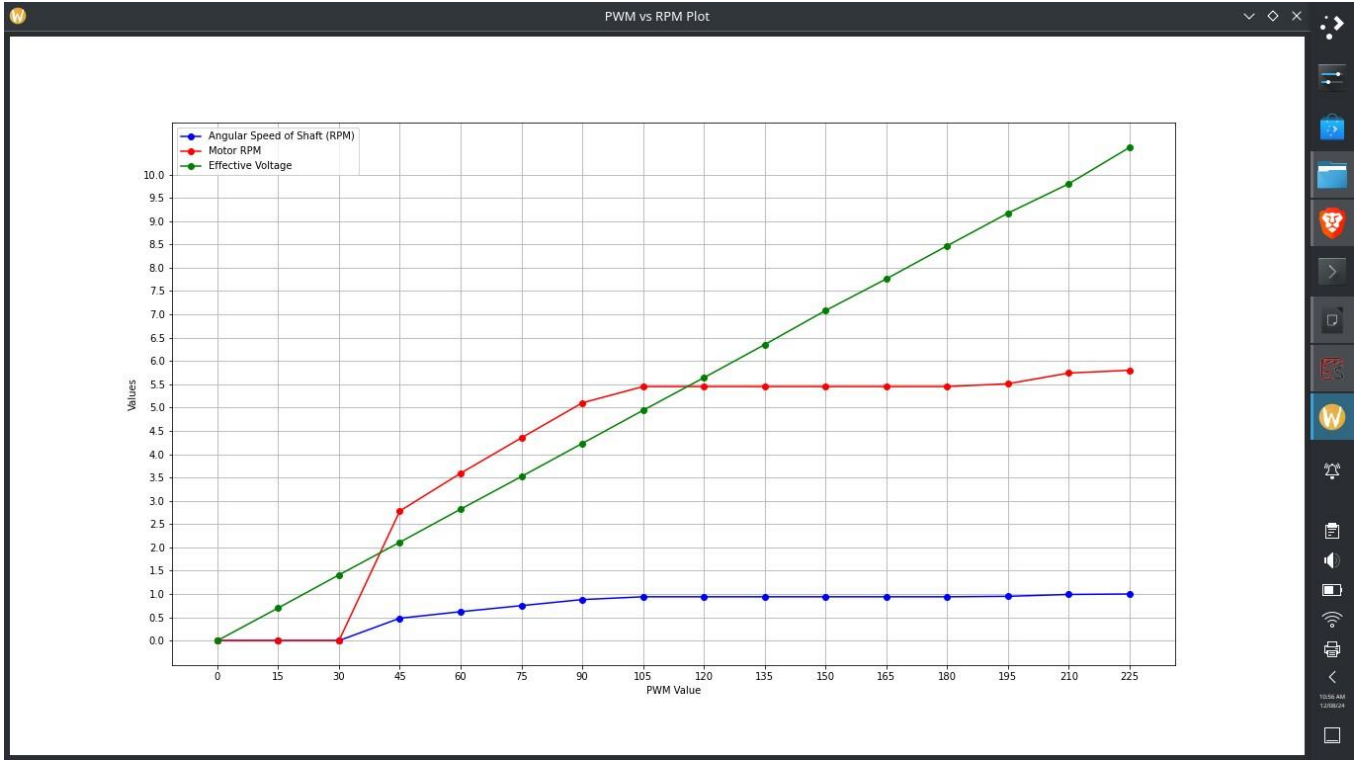
Parameter Submission: The "Submit Parameter" button consolidates all configured values and transmits them to the Arduino via serial communication from the Python script. This ensures synchronized execution of the motor control algorithm.

Reset Functions:

- The "Reset Parameter" button clears all input fields within the GUI, allowing for fresh configuration inputs.
- The "Reset Antenna Position" button commands the system to return the antenna to its default parking position, resetting the antenna's orientation to zero degrees.

CHAPTER 10

OBSERVATION WITH THE PROPOSED RFI ANTENNA CONTROL SYSTEM



	PWM Value	Rotation Time (s)	RPM Value	Motor RPM	Effective Voltage
1	0	0	0	0	0
2	15	0	0	0	0.7
3	30	0	0	0	1.41
4	45	31	0.48	2.78	2.11
5	60	24	0.62	3.59	2.82
6	75	20	0.75	4.35	3.52
7	90	17	0.88	5.1	4.23
8	105	16	0.94	5.45	4.94
9	120	16	0.94	5.45	5.64
10	135	16	0.94	5.45	6.35
11	150	16	0.94	5.45	7.08
12	165	16	0.94	5.45	7.76
13	180	16	0.94	5.45	8.47
14	195	15.8	0.95	5.45	9.17
15	210	15.17	0.99	5.51	9.8
16	225	15	1.0	5.74	10.58

The table shows the relationship between the PWM value applied to a motor and the resulting RPM of the antenna shaft.

- **PWM Value:** This column represents the pulse width modulation value applied to the motor, which controls the motor's speed. Higher PWM values correspond to higher motor speeds.
- **Rotation Time (s):** This column shows the time it takes for the antenna shaft to complete 90 degrees of rotation. A lower rotation time means a faster RPM.
- **RPM Value:** This column represents the revolutions per minute of the antenna shaft, calculated based on the rotation time. Higher RPM values indicate faster rotation.

Observations:

- As the PWM value increases, the rotation time generally decreases, leading to a higher RPM.
- There is a non-linear relationship between the PWM value and the RPM, with larger increments in PWM resulting in smaller increases in RPM, especially at higher PWM values.
- The relationship between PWM and RPM is likely influenced by factors such as motor characteristics, load on the antenna shaft, and the efficiency of the motor.

Explanation:

The motor's speed is determined by the frequency and duty cycle of the PWM signal. A higher duty cycle (i.e., a larger PWM value) means more power is applied to the motor, resulting in faster rotation. However, the motor's mechanical characteristics and the load on the antenna shaft can affect the actual RPM achieved.

This data can be used to understand how PWM controls the motor speed and its impact on the antenna shaft's RPM. It can be used to optimize the PWM value to achieve the desired antenna rotation speed for various applications.

CHAPTER 11

CONCLUSION AND FUTURE SCOPE

In conclusion, this project has successfully achieved its objective of developing a highly precise and controllable antenna system for effective RF signal detection. Through the seamless integration of an Arduino microcontroller with a Python-based user interface, the system delivers precise antenna positioning based on user-defined parameters. The Arduino's efficient command processing and motor control, combined with the user-friendly Python interface, have ensured that the antenna can be accurately directed to capture desired RF signals.

The incorporation of feedback mechanisms, such as encoders, has been pivotal in maintaining the desired accuracy, allowing for real-time adjustments and ensuring consistent performance. This feedback loop has not only enhanced the precision of the antenna movements but also contributed to the system's robustness and reliability.

Additionally, the project's attention to error-handling and calibration routines has guaranteed that the system operates reliably under various environmental conditions, with minimal performance degradation. Effective power management has been implemented to support extended operational periods, ensuring the system's endurance and reliability.

Overall, this project stands as a testament to the successful fusion of hardware and software, achieving a sophisticated antenna control system capable of precise RF signal detection. The meticulous design, implementation, and testing have culminated in a robust, reliable, and highly accurate system that meets the project's ambitious goals.

Future Scope:

1. Implementing a Braking System:

- Introduce a braking system to mitigate motor backlash and enhance precision.

2. Replacing the DC Motor with a Stepper Motor:

- Use a stepper motor instead of a DC motor to achieve finer control over antenna movements.

3. Limiting Motor Rotation to 270 Degrees:

- Restrict the motor's rotation to 270 degrees to prevent damage to the wiring system.

4. Integrating an Absolute Encoder:

- Add an absolute encoder alongside the existing encoder to obtain precise positional data of the antenna.

5. Installing Limit Switches:

- Place limit switches at both ends to prevent the antenna from exceeding its operational limits.

6. Developing a Power Management Plan:

- Create a power management strategy to ensure the Arduino remains operational during power failures.

7. Establishing a Reset Position:

- Define a reset position for the antenna, so that in case of encoder failure, the antenna returns to a known parking position, providing a reliable reference point for subsequent operations.

CHAPTER 12

REFERENCES

- 1. Tata Institute of Fundamental Research. Handle/2301/321. n.d.**
<http://library.ncra.tifr.res.in:8080/jspui/handle/2301/321>
- 2. Spyder Project. (n.d.). Spyder. GitHub. Retrieved August 6, 2024, from**
<https://github.com/spyder-ide/spyder>
- 3. KDE (a.d.). Tutorials. KDE UserBase. Retrieved August 6, 2024, from**
<https://userbase.kde.org/Tutorials>
- 4. Arduino. Retrieved August 6, 2024, from**
<https://www.arduino.cc/en/software>
- 5. Debain. (n.d.). Debain tutorial. Retrieved August 6, 2024, from**
<https://www.debian.org/doc/manuals/debian-reference/ch01.en.html>
- 6. Tata Institute of Fundamental Research. Servo System**
<http://www.ncra.tifr.res.in/ncra/gmrt/sub-systems/servo-system-for-gmrt-antennas>