6-9-2025

Santa Clara Radio Astronomy Project IV: Data Acquisition

Nicholas Alva

Peter Lattimer

SANTA CLARA UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Date: June 9, 2025

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Nicholas Alva
Peter Lattimer

ENTITLED

Santa Clara Radio Astronomy Project IV: Data Acquisition

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN ELECTRICAL ENGINEERING

Thesis Advisor: Dr. Kurt Schab

Department Chair: Dr. Shoba Krishnan

Santa Clara Radio Astronomy Project IV: Data Acquisition

by

Nicholas Alva
Peter Lattimer

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Electrical Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 9, 2025

Santa Clara Radio Astronomy Project IV: Data Acquisition

Nicholas Alva
Peter Lattimer

Department of Electrical and Computer Engineering
Santa Clara University
June 9, 2025

ABSTRACT

Santa Clara Radio Astronomy Project (SCRAP) is a multi-year project that aims to make radio astronomy accessible to the Santa Clara University community. Previous iterations have focused on constructing and improving a parabolic radio telescope located on the balcony of the fourth floor at The Sobrato Campus for Discovery and Innovation at Santa Clara University. Until this year, the radio telescope has been stationary. As a consequence, the telescope is only capable of observing celestial objects directly above it, severely limiting the data capture area. This thesis describes the contributions of SCRAP IV Data Acquisition to the project: enabling 2D scanning through a rotational system. This thesis will describe the defined objectives, subsystems, integration and results of the addition of the rotation system. The addition of a rotational system greatly increases the opportunities of the telescope, allowing users to collect significantly more data as well as track specific celestial events.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

SCRAP, or Santa Clara Radio Astronomy Project, is a multiyear project that aims to bring radio astronomy capabilities to the Santa Clara University community. This year's iteration, SCRAP IV, aims to improve the capabilities of the radio telescope through two teams: a Data Acquisition Team and a Data Processing Team. This thesis will provide some background information on the system and its development, as well as the contributions of the SCRAP IV: Data Acquisition Team.

1.1 About Radio Telescopes

The electromagnetic (EM) spectrum is the vast span of all electromagnetic radiation in the universe. This encompasses all types of radiation including radio waves, infrared, visible light and all others. In early astronomy, observations of space were made based on visible light. While this provided a lot of insight about the galaxy, visible light is only a narrow band within the the EM spectrum. There are many phenomena in space that emit radiation in other different parts of the spectrum outside the visible band. An example of a different part of the spectrum are radio waves, which include electromagnetic waves from 3kHz to 300GHz[1]. As the name would imply, radio telescopes are telescopes that make observations of radio waves to survey the universe. These radio telescopes typically consist of a parabolic dish, a feed element, and a component to aim the telescope in particular directions as illustrated in **Figure 1.1**[2]. It is important to note in **Figure 1.1**, the feed element is placed in the center of the parabolic dish whereas SCRAP's feed element is located where the secondary reflector is. A feed element is a component used to collect all radiation the parabolic dish has caught and focused. The feed element is designed to allow electromagnetic waves with specific frequencies to be observed by astronomers. Astronomers process the information observed by the feed element using software to interpret the electromagnetic waves as information about phenomena that occur in space.

Radio astronomy is particularly effective because celestial radio waves can be observed from Earth's surface as radio waves can pass through mediums that visible light cannot. Another valuable aspect of radio astronomy is that the source of the radio waves that are observed are hydrogen atoms as they undergo changes in their state of energy.

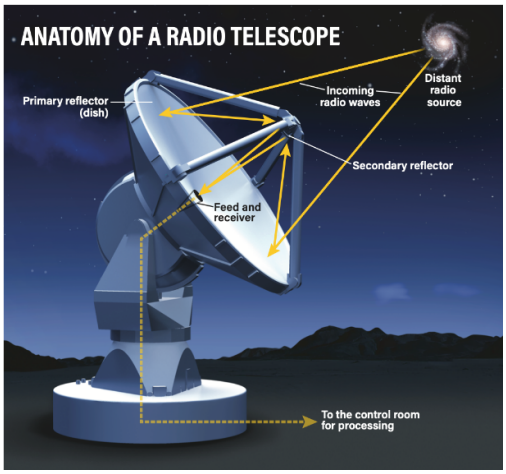

Figure 1.1: Typical Structure of Radio Telescope[2]

Since hydrogen atoms that undergo these changes are so abundant in space, there are countless observations that can be made. For these reasons, radio astronomy has made discoveries, including the first image of a black hole[3]. The radio waves that are emitted from hydrogen have a frequency of 1.42GHz and have been designated by the FCC[1].

1.2 The Development of SCRAP

Prior to our iteration of SCRAP, there were three previous teams that developed the radio telescope, all of which created or improved different elements of the radio telescope.

1.2.1 SCRAP-I 2022

SCRAP-I was the first team to begin developing our radio telescope by determining what parameters should be evaluated for a horn antenna as shown by the structure in **Figure 1.2**. This team also worked to develop Software Defined Radio (SDR) and microstrip bandpass filters. SCRAP-I demonstrated a proof of concept that radio waves can be collected by an antenna designed to resonate with radio waves[4]. With the help of an SDR, they established the first major step by collecting data from electromagnetic radio waves.

1.2.2 SCRAP-II 2023

SCRAP-II was responsible for creating the physical structure we are using now. This structure is a parabolic radio telescope built to provide significantly improved performance of data acquisition and potential for improved data processing of the recovered data.[5] This was an essential step for developing SCRAP, as the parabolic radio telescope as seen in **Figure 1.3** is the foundation of the project that was used for SCRAP-III and our current iteration of SCRAP (SCRAP IV) to obtain information about the galaxy.

Figure 1.2: SCRAP-I Proof Of Concept Horn Antenna[4]

Figure 1.3: SCRAP-II Parabolic Antenna with Initial Feed Element[5]

Figure 1.4: SCRAP-III Parabolic Radio Telescope with Improved Feed Element[6]

1.2.3 SCRAP-III 2024

SCRAP-III was responsible for developing the feed element that further improved the performance of our radio telescope. This feed element is installed on top of the radio telescope, and collects focused radio radiation for the SDR to process. This feed element is designed to act as a focal point of the electromagnetic waves that reflect off the dish to further improve the ability to observe radio waves.[6] Not only did they successfully design a feed element that improved the performance of the telescope, as seen above the telescope in **Figure 1.4**, they also developed code to visualize the data obtained by the radio telescope.

After all contributions from SCRAP-I to SCRAP-III, we now have the fourth iteration of SCRAP known as SCRAP-IV. SCRAP-IV consists of four members that have been broken into two teams: Data Acquisition and Data Collection, which work in parallel to improve their respective aspects of the radio telescope. The Data Acquisition Team is in charge of improving the amount of data that the Radio Telescope can receive and how the data is provided in real time, while the Data Processing Team is responsible for developing the processing of data to refine insights about what the radio telescope is observing. Each team will provide their own thesis, meaning this thesis will only cover the contributions of the Data Acquisition Team.

1.3 Initial Abilities of SCRAP

As discussed in the previous chapter, SCRAP has been refined over three years to bring the end goal of providing accessible radio astronomy to the Santa Clara Community closer. The initial abilities and features of SCRAP as of June 2024 included:

1. A Parabolic Radio Dish that bounces radio waves at a focal point
2. An improved feed element to enable better detection of radio waves
3. Basic data acquisition script to help with data processing via Python
4. Basic data Processing script to translate data into useful and understandable image (waterfall plots) using Matlab

1.3.1 Parabolic Radio Dish

The parabolic radio dish is the largest structure of the telescope. This dish is essential for radio telescope operations as its surface area, material, and shape allows it to act as a net for radio waves. Depending on the strength and distance of the source, incoming radio waves can have very low power, and having a large surface area improved the directivity of SCRAP's radio telescope from 19.89dBi to 31.23dBi[5]. The shape of the dish reflects incoming radio waves towards the focal point, where the feed element is placed.

1.3.2 Improved Feed Element

The radio telescope has an improved feed element that is attached to the center atop the parabolic radio telescope that acts as focal point for the radio waves that are reflected off of the surface. This feed element was implemented with a stainless steel "can" design with an operating frequency of approximately 1.42GHz that is capable of withstanding weather conditions without sustaining critical damage.

1.3.3 Basic Data Acquisition Script

SCRAP's radio telescope was also initially capable of interpreting radio waves as digital data. This digital data was handled and stored on a computer that operated next to the dish and stored the data on a file as numerical information using Python.

1.3.4 Basic Data Processing Script

The radio telescope and the computer that operated it was capable of obtaining, storing, and processing the data that was recovered from the dish. This data was processed in a way that could illustrate energy levels of radio waves from space and would project that information in the form of a waterfall plot via Matlab. This waterfall plot shows a range of frequencies (frequencies that the radio telescope is designed to observe) and the corresponding energy of those incoming waves at certain orientations of the sky (in terms of Right Ascension).

1.4 Obstacles of SCRAP

SCRAP's radio telescope is capable of recording data along a line of right ascension that is above the telescope's zenith. We aim to improve the coverage of the radio telescope by allowing it to scan 20 degrees below the line of its

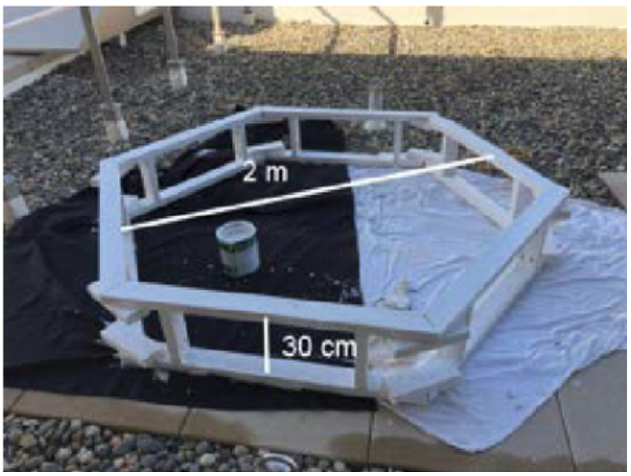

Figure 1.5: SCRAP-II Wooden Ground Base[5]

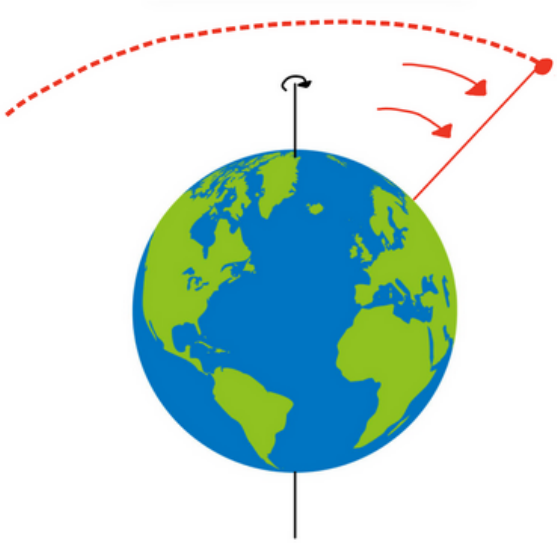

Figure 1.6: Dashed Red Line Showing Line of Right Ascension Captured by Initial SCRAP Radio Telescope[7]

right ascension. To enable the desired angular coverage, three obstacles that must be dealt with are as follows:

1.4.1 Lack of Mobility

The first obstacles that SCRAP faces is that it is initially stationary not only in its position where it stands, but also in the direction that it listens to. This is because of the fact that the dish lays on top of a stationary wooden ground base with a geometry that holds the dish in place as shown in **Figure 1.5**. As a result of this, the radio cannot aim at any position other than its zenith (directly above the ground). This is a problem because this only allows the dish to observe a line in the sky using Earth's rotation as its only axis as shown in **Figure 1.6**.

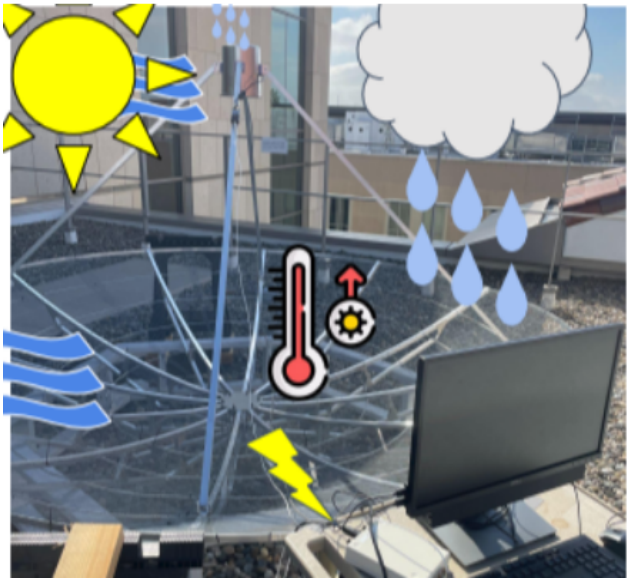

Figure 1.7: SCRAP-III Radio Telescope Being Operated by Outdoor Computer System[6]

1.4.2 Short Term Operation

The second obstacle SCRAP faces is that it can not operate for long periods of time. This is because it is currently designed to be operated by a local computer that has to be outdoors in order to make digital observations about any detected radio waves, as shown in **Figure 1.7**, taken by the last SCRAP team. This prevents SCRAP from being able to develop a radio spectrum image of space by gathering data in real time as it would take significant periods of time and would need to operate by itself while withstanding natural elements such as rain and long exposure to the sun.

1.4.3 Short Term Data Storage

The third obstacle that SCRAP faces is the fact that it is incapable of storing large amounts of data in a format that can be easily extracted by our Data Processing team for them to process it into useful visuals.

1.5 Objectives

We aim to enable the SCRAP radio telescope to be able to change its inclination in real time in order to obtain a two dimensional radio map. This radio map is called a raster image: an image created by scanning line by line to create a complete image. This would solve the limitation imposed by the radio telescope being stationary described in the previous section by allowing the user to aim the dish. With the rotation of the Earth, the radio telescope is able to map an “arc” in the sky. Our goal is to design and implement a tilting system that is capable of orienting our antenna along its right ascension to enable a two dimensional scan in space. **Figure 1.8** illustrates how Earth’s rotation moves in the direction of the telescope’s declination and how moving the telescope along it’s right ascension covers a two

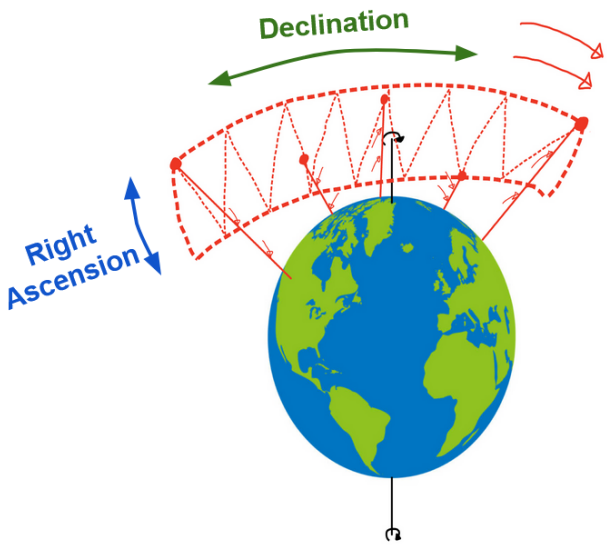

Figure 1.8: Illustration of How Right Movement in Right Ascension and Declination from Earth's Rotation Creates a 2D map[7]

dimensional area of the sky. With the appropriate speed and accuracy, these objectives would enable the creation of raster images via radio emissions.

Our specific design goals for implementing dish-tilting functionality are:

- **Angle of rotation:** Angular range of motion that the telescope can move
- **Angular velocity:** Speed at which the radio telescope can change its angle
- **Mode of rotation:** Source code that makes rotation system move to different angles
- **Accuracy of angle:** The ability for the radio telescope to accurately move to its target angle

With these specifications, if implemented, the radio telescope will be able to change its inclination in real time to obtain the data needed to create a 2D raster map.

1.6 The Relevance of our Solution

The goals of this project is important in bringing the full capabilities of radio astronomy to Santa Clara University. The current solution provides some insight into radio astronomy, however to take full advantage of radio emissions, it is necessary to enable movement in the antenna. By allowing the radio telescope to point in different right ascensions, it can observe more of the universe. With more data we will be able to observe and study more of our universe as well as create a visual mapping of RF data emitted by the universe.

1.6.1 Why SCRAP's Goal is Important

SCRAP's goal is important because it enables the department and the wider Santa Clara University Community to observe significantly more of the universe in a personal and accessible way. This will hopefully foster an interest in space exploration from the wider community of Santa Clara University.

1.6.2 Who Benefits from SCRAP

SCRAP aims to provide exploration and learning opportunities for the RF department and physics department as well as any Santa Clara University student with an interest in astronomy. SCRAP aims to provide both departments with the capability to observe this spectrum from Santa Clara's campus in an affordable manner.

1.6.3 Roadmap

SCRAP-IV Rotation was broken into three major parts throughout the academic year. The first phase was how we chose and utilized the core components of our structure being the actuator and the angle sensor all together on indoor prototype. The second part was the creation of the rotation module that integrated the entire system together in a way that provided safety features. In the final quarter, we chose to build our final structure and install it onto the radio telescope, finally enabling the movement of the Radio Telescope and connecting the system to a computer system.

Chapter 2

System Scope and Overview

For this project, we want to create a “plug and play” device to enable our radio telescope to move in order to capture 2D images of the sky. Factors considered during the design process include location, budget, and available components. As a result, these design constraints shape how we approach the design of our project. This chapter will explain how the design was approached with respect to constraints, requirements and goals.

2.1 Technical Project Constraints

There are several limitations and requirements that factored into the final design of this project. When considering constraints, the first was the surrounding environment. Our environment works both with and against us in the design of our radio telescope. As stated in previous sections, a big advantage of radio telescopes is that they function from Earth’s surface. As a result, this enables earth’s rotation to naturally provide us with one dimension of the radio map. This means to achieve a two dimensional scan of the sky, it is only required to enable one axis of rotation, specifically in the north-south plane of the sky as this is normal to earth’s rotation. This will be done by components described in Chapter 3, however our environment partially restricts movement in this axis. As shown in **Figure 2.1**, movement in certain directions are limited by whatever physical structures may be in the way of the radio telescope that won’t allow electromagnetic waves to propagate through such as local buildings. These buildings that surround the radio telescope are the electromagnetics lab and the roof top of Sobrato Campus of Discovery and Innovation.

As a result of these obstacles, the effective angular range of the radio telescope is limited. This constraint can be described by a maximum angle of approximately 60 degrees facing south.

Resources were also a constraint for the rotation system design. The two major resource limitations in our project are the $500 limit imposed by the School of Engineering, as well as the nine month time constraint. Considering the antenna, computer and some parts were already provided, the spending limit was not as restrictive as our time constraint. Also, given that the current base is structurally sound and that we only have three quarters to complete the project, we decided to append to the current base.

Figure 2.1: Antenna in Relation to SCDI Location

The second constraint of the rotation system is the weight that the system can push. This means that the the radio telescope and anything attached to it must not exceed the weight limit of said system. The weight of the motor that we're using is discussed in further detail in **Chapter 3**.

2.2 Requirements

The rotation system for the radio telescope needs to be able to operate in an easily controllable way for future users. To ensure this, we identified several requirements that must satisfy all technical, operational, and compliance related aspects of the design.

2.2.1 Technical Requirements

One of the requirements of the rotation system is that it must have some digital interface, such as a computer, for a user to input commands in order to operate the system. These commands would include the angle that they would like the telescope to move. Another requirement for the system is that it must be modular. This means users should be able to easily add or remove the rotation system onto the radio telescope. This is important because it allows for users to be able to protect the rotation system equipment from extreme weather conditions.

The last requirement of our rotation system is that it must maintain at least 3° of accuracy with respect to the desired input angle from the user. Radio astronomy is a popular branch of astronomy for reasons stated in Chapter 1, including being accessible on Earth's surface. This means there is plenty of open data accessible online[8]. Ensuring minimum accuracy means that we will be confident in where our radio telescope is pointing, which will enable users to cross reference data and observe specific astrological events.

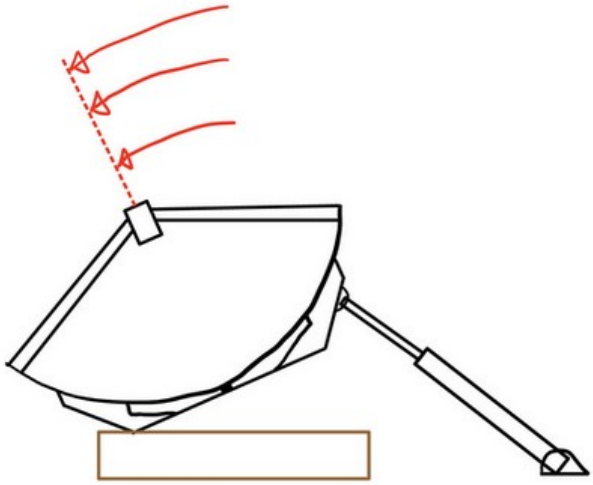

Figure 2.2: Rough Design Showing how the Rotation System was Designed

2.2.2 Range of Motion

As previously stated, in the beginning phase of the design, time and money were important considerations. One of the ways that this impacted our design was the decision to limit the rotation system from being able to tilt left and right to only allow it to tilt in one direction as shown in **Figure 2.2**. By only allowing the system to rotate in one direction, this allowed us to keep the design relatively simple and cost effective. This design only requires one linear actuator and a stationary point for the telescope to pivot on. Once the decision of rotating only in one direction was made, we chose to allow the rotation to occur towards the South. This decision was made because of the location of the radio telescope as the buildings that surrounded it blocked its line of sight far more in the North than the South.

The last decision for scope that was made was to not allow the system to change it's cardinal direction mechanically. The cardinal direction refers to the telescope's ability to point to the north, south, east, and west with respect to its position. This would require a mechanical system that would likely be very complex and would need to be able to rotate a large system. It was decided that this would be too complex and as a result would not be part of the design.

2.2.3 The Source Code

Another limitation of scope that was considered in the design of the rotation system was the complexity of the source code. Programming is a very deep and complex area of study and the complexity of the code for this system could be quite extensive. Going too deep into the code for our purposes could lead to an over-engineered design that is unnecessarily complex for users. As a result, we kept the code very methodical and only focused on implementing features that were needed as defined by the requirements of the system.

2.2.4 Safety Requirements

In accordance with Santa Clara University's safety requirements, there were several hazards we had to be aware of during the development of our project. Since we needed to work on the roof of the SCDI building, we had to ensure that none of the components we installed could get blown off the roof and possibly injure people walking below. Another issue was that of personnel safety while working on the roof of SCDI. In order to prevent this, we simply utilized the metal rail that was installed by the school to prevent us from getting too close to the edge which kept us safe during our roof work.

2.2.5 Legal Requirements

The radio telescope complies with the legal requirements in the United States established by the FCC[9]. SCRAP is only receiving and is not transmitting any radio waves that could potentially interfere with communication on the electromagnetic spectrum.[9] As a result, there is no danger of violating FCC laws to protect communications.

One potential concern of SCRAP is that it is listening to radio waves at 1.42GHz and because of this, SCRAP could detect communications at that frequency. However, this band is designated by the FCC as an EM spectrum protected for radio astronomy and can only be used by passive systems.[9] Because of this, SCRAP will not violate privacy or FCC regulations.

2.2.6 Requirements of Stakeholders

SCRAP has two major stakeholders that are taken into consideration: the Electrical and Computer Engineering Department, and the users of the radio telescope. It is important that our system is easy to interact with and operation is accessible. In order to achieve this, we will make the user interface extremely intuitive and straightforward with clear prompts. This includes methodical coding practices as well as minimal connections as previously described. We have also provided additional documentation listed in the following reference[10] that future users can reference in order to understand how to operate our system and the code that controls it.

2.2.7 Ethical Considerations

SCRAP aims to make radio astronomy accessible to the greater Santa Clara University community. SCRAP does this by consuming a very small amount of power in order to record radio emissions from the 1.42GHz hydrogen line which is streamed into a very affordable computer. This computer will stream this raw data to the internet in a way that makes our data accessible to anyone with a computer and an internet connection. Since SCRAP only requires the power of a small motor, several microcontrollers, and an angle sensor, the power consumption of the system pales in comparison to the power usage of the SCDI building. In addition, the system doesn't need constant replacements of parts, hence the environmental impact is negligible.

Function	Objective
Angle of Rotation	Minimum of 20°
Angular Velocity	Minimum of 5° per minute
Mode of Rotation	Angular Sweep
Uncertainty of Angle	Maximum of ± 3°

Table 2.1: Objectives and Scope of SCRAP IV: Data Acquisition

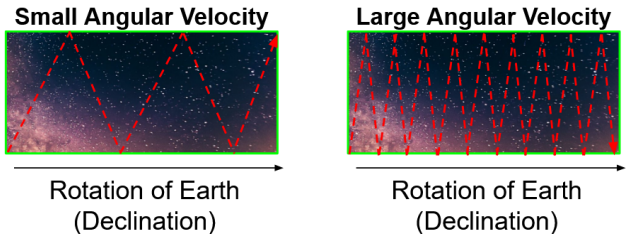

Figure 2.3: How Different Angular Velocities Impacts Radio Telescope Coverage of the Sky(Dotted Red Line)[11]

2.3 Scope of Services

The scopes and services of the project are what capabilities that the rotation system aims to provide its future users. These include the objectives as shown in **Table 2.1**.

2.3.1 Angle of Rotation

The angle of rotation is the amount of degrees that the radio telescope can change its inclination from rest. This objective is important because it dictates how much of the sky that the radio telescope can cover. The more degrees of rotation, the larger the area the radio telescope can cover of the sky and the more data it can cover on the radio map.

2.3.2 Angular Velocity

The angular velocity of the system is the rate of speed that the radio telescope can change its inclination with respect to time. Angular velocity is valuable as the faster the angular velocity is, the better the resolution of the radio map will be. Angular velocity translates to an increased resolution because it increases the amount of times that the radio telescope scans across the sky with respect. The affect that angular velocity has on the 2D radio map is illustrated in **Figure 2.3**. Since the velocity that the radio telescope pans across the sky remains constant as a result of Earth's rotation being constant, the radio telescope can better utilize the area covered by the angle of rotation.

2.4 Design Rationale, Criteria, and Standards

An important aspect of the project was how the physical components of the rotation system were to interface with a host computer through code that we could write. Discussions about which language to use for this project ultimately

boiled down to two options: Matlab and Python. We ranked these programming languages on a scale of 1-5 (5 being favored) in the following categories: familiarity (how popular the language is), ease of use, libraries and our experience with them. Both are more than capable for the scope of our project, and after weighing pros and cons shown in **Table 2.2**, our team selected Python. Every day Python becomes more industry standard for many reasons. With the online resources for learning Python and the beginner friendly syntax of the programming language, we hope future SCRAP teams will have no issues following our code.

Another reason we chose Python is because of the amount of support that it provides for interfacing with common microcontrollers in the market. This is valuable for the project because microcontrollers can act as a link between a host computer and the analog components of the system. This in turn would enable the host computer to control the entire system, enabling all processing to be done digitally.

Coding Decision Matrix	**Familiarity**	**Ease of Use**	**Libraries**	**Experience**	**TOTAL**
Python	5	4	5	3	17
Matlab	3	3	3	4	13

Table 2.2: Coding Decision Matrix Comparison

2.5 Motivating Factors and Design Principles

During the development of the rotation system, there were several aspects of the project that were always considered and incorporated into each design decision. These factors were simplicity, affordability, and modularity.

2.5.1 Simplicity

The reason simplicity was an important aspect of the design was to prevent going beyond the scope of the project. If the design is too complex, it introduces the possibility of over-engineering or taking too much time to accomplish our goal by the due date. Another reason for simplicity to be part of the design approach is because SCRAP is a multiyear project. Since it is a multiyear project, there is a high likelihood that there will be future teams working with our rotation system. If the design is simple, this makes it possible for future teams to not only use our design but to also make it easier to conduct maintenance on the system.

2.5.2 Affordability

In order to make this system accessible, we felt that it was important that the system is affordable and so the design must have reasonably priced components. Another motivation for our system to be affordable is that it helps future users maintain its functionality over time. For example, if the motor for the system fails after a long period of use, it could affordably be replaced.

2.5.3 Modularity

The final aspect of the design methodology for this project is modularity. If the system is modular, it allows the components of the rotation system to be easily attached and detached to the radio telescope when needed. This is important because there can be different types of weather conditions that pose certain problems for our electrical components. Even though we designed the system to withstand normal weather conditions, it is important to protect it during extreme conditions. When these situations arise, the components can be disconnected and placed indoors a safe place.

Another motivation of our system being modular is that it makes troubleshooting the system much easier. This is because if the system presents errors during its operation, the malfunctioning component can be traced to its module allowing the users to decide if they want to replace or repair it.

2.6 Project Design Methodology

The rotation system consists of three essential components: an actuator to provide force to move the telescope, a sensor to determine what angle the radio telescope is at, and a control module control and integrate the components together. All of these components have different approaches to their design, each with their own reasons as to why it was chosen. The following subsections are the design methodologies that we used for each component of our data acquisition system.

2.6.1 Choice of Force

In order to physically change the inclination of the radio telescope, a means of applying force to the system is required. There were several possible ways of achieving this which include some type of pulley system or a rotary motor system. A pulley system could be attached to one end of the telescope where a rope or chain and sprocket could pull one side of the telescope when a change in inclination is needed. Another potential method of applying some type of force could be a rotary motor that applies its force through a rotating axle where the radio telescope can be attached to the rotating motor.

Despite these potential methods of applying force to the system, we decided on utilizing a linear actuator. We chose a linear actuator because it applies its force in a linear fashion from a single position. This allows the motor be placed in a stationary position and significantly decreases the amount of moving parts compared to a pulley system. Another benefit to using a linear actuator is that it only requires a very simple point of contact between the radio telescope and the motor. If a rotary motor was utilized, it would likely complicate the required design for the point of contact between the radio telescope and the rotary motor. As a result, we felt that it was best to utilize a linear actuator

for our design.

2.6.2 Choice of Angular Detection

If the rotation system is to change the inclination of the radio telescope to a target angle in real time, it needs to know how far to push. The design needs a feedback mechanism when rotating the radio telescope to its targeted inclination. There are several ways that this can be achieved. One of the designs could be an analog angle protractor that could be mounted on the radio telescope while the user has direct control over the system. The idea would be that the user would watch the angle protractor while the radio telescope is moving. While the system is moving, the user can stop the rotation of the system when the desired angle is achieved.. Another method that can be used to determine what the inclination of a system is in real time could be the use of video assisted software. An example of this video assisted software would be OpenCV, also known as contour recognition[12]. OpenCV is a python based software that is capable of recognizing varying geometries from videos. This software and others like it, could be used to recognize the geometry of the radio telescope and the orientation in real time. If this type of video assisted software was used, a camera would consistently feed video data to a host computer. The host computer would use the video assisted software to make visual approximations of the angle that the radio telescope is pointed to.

For our design, we ultimately decided to use an electronic angle sensor to detect the angular inclination of the system. We chose this option because electronic angle sensors are accurate, quick, and lightweight in nature. Since these types of angle sensors can provide angular data electronically, we can capitalize on this data with a computer for feedback control. This would translate into a higher likelihood of the rotation system having an accurate interpretation of where the radio telescope is pointing to in the sky.

2.6.3 Choice of Microcontroller

Another major component of our design is the circuitry that will be used to control our linear actuator. The linear actuator can be electronically controlled and so the circuitry must be able to interact with the motor as well as a computer that sends commands to the system. One option for this circuitry would be to design a PCB from scratch. The benefits of doing this would be that it allows us to have the most control over the circuit's behavior and would also enable the circuit to be extremely efficient and compact. As efficient as this would be, problems of the complexity would arise. Since this circuit would be used to control a linear actuator, we would need to spend time and effort into finding the correct integrated circuits to safely operate the motor and the main computer. These types of circuits have already been designed and optimized for their performance and robustness so we decided to use off the shelf compoents whenever possible.

Two candidate microcontrollers that we could utilize that fall under these categories: the Raspberry Pi and Arduino Uno microcontrollers. The Raspberry Pi is a very capable single-board computer with a processing speed of 1.8GHz

[13], which would translate to a faster reaction time. This compared to the Arduino processing speed of 16MHz [14] is magnitudes faster in processing speed. This means that the Raspberry Pi would be able to execute the source code and make decisions in real time faster than the Arduino. Although the Raspberry Pi is faster than the Arduino and can be used as a microcontroller, both the Arduino and Raspberry Pi would be able to process the needed information fast enough for our operation. We ultimately decided to use the Arduino Uno R3 because it was a microcontroller that met our needs and one that we already had experience with and had access to. This was important to us as it saved us precious time enabling us to focus on other important aspects of the project. In addition, the Arduino Uno R3 is a dedicated a microcontroller whereas the Raspberry Pi is a single-board computer making it a more general purpose solution.

Chapter 3

Subsystems

In this chapter, we will cover all subsystems and components involved in our design. This will include their purposes and how we chose the components, and then discuss how they will be integrated together in **Chapter 4**.

3.1 The Linear Actuator

To rotate the dish, we will need a motor to apply force. Typically, electric motors convert power into rotational energy; however, as described in our solution in the previous chapter, we would like to apply force linearly. To do this, we use a linear actuator: a device that converts the rotational force generated by a DC motor into a linear displacement. The linear actuator that we used was a Vevor OK648-1000N as shown in **Figure 3.1**.

The linear actuator runs off 24 W DC power and comes with a AC-DC converter to convert power from the wall into 12 V at 2 A. This will be important in the next section.

Figure 3.1: Vevor OK648-1000N Linear Actuator[15]

Figure 3.2: HiLetgoBTS7960 High Current 43A H-Bridge Motor Driver used to Control Actuator[16]

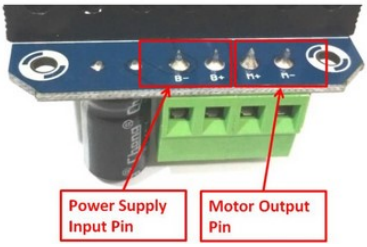

Figure 3.3: Green Casing of the H-bridge Motor Driver that can Create a Connection for the Large Wires of the Actuator[16]

3.1.1 Motor Driver

In order to dictate movement of the radio antenna, we need a component capable of regulating and controlling the motor. To do this, we use a component called an H-Bridge, also known as a motor driver. The H-Bridge will serve as the controller that receives commands from the Arduino, which, by extension, receives commands from the main computer. The H-Bridge enables us to control the actuator despite the physical and electrical problems presented by the actuator for two reasons:

First, the actuator cables are thick and do not fit into the Arduino pins. This is because the actuator operates on 2 amps, 100 times larger than the current of a typical Arduino digital output, so it requires thicker wire. Arduino pins are designed for thin jumper wires, or 22 gauge wire, which are much narrower than the X gauge wire used by the actuator. The H-Bridge motor driver is controlled by pins that connect to the Arduino as shown in **Figure 3.2**. These pins receive signals from the Arduino and regulate the output connections with respect to the input as shown in **Figure 3.3**. It can seen that the green casing has pin holes where the large wires, such as the ones that control actuator, can connect to. As a result, we were able to control the actuator through Arduino control pins despite the incompatible wires.

Second, The actuator pulls 2A and 12V, which is beyond the 5V and the roughly 40mA that the Arduino Micro-controller is capable of providing[14].

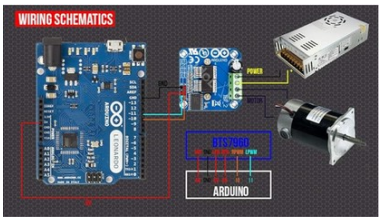

Figure 3.4: HiLetgo BTS7960 Wiring Diagram[17]

Figure 3.5: Arduino Uno Rev3 [14]

To circumvent these problems, we decided to use the BTS7960 High Current 43A H-Bridge Motor Driver as shown in **Figure 3.2**. This H-Bridge is is capable of handling 43A and 27V, well beyond what we need [16]. With the BTS7960 Motor Driver, we get a safe and reliable way of controlling our actuator from the Arduino despite the large amount of power that the actuator draws. The approach to how the software runs this component is described in detail in **Section 3.5** and is visually represented in **Figure 3.4**.

3.2 Microcontroller

To control our components, we need something capable of providing power to the various subsystems described in this chapter. Computers are great for serial communication, however they are not able to provide the 5V signals needed in this system. This is where microcontrollers are helpful. A microcontroller is a small, single board computer that has input and output pins (I/O) for controlling and providing power for various components.

The Arduino Uno, the microcontroller chosen for this project shown in **Figure 3.5**, comes with three types of pins: analog, digital, and pulse width modulation (PWM). For the H-Bridge, and by extension the actuator, we will be using a PWM pin. PWM pins are digital pins that have additional functionality of controlling the duty cycle, or how long the pin is driven high relative to low within the boards 1 ms period [14]. For example, a 50

3.3 Angle Detection

When changing the inclination of the radio telescope, it is important to know at what angle the telescope points in real time. This is important because if the angle of inclination is unknown, the rotation system won't know when to

Figure 3.6: SINDT-TTL Digitaal Accelerometer (Angular Sensor) [18]

stop changing angle when attempting to aim at a target of interest in the sky.

3.3.1 The Angular Sensor

In order to determine what the angle of our radio telescope is in real time, we decided to use a **SINDT-TTL Digital Accelerometer**, as shown in **Figure 3.6**, created by WITMOTION as the angular sensor for the system. In order to power the sensor, it must be connected to a computer's USB 3.0 port where it will simultaneously send angular data. This minimizes wiring, as one wire bus between the angular sensor and computer is sufficient.

3.3.2 Data Extraction from Sensor (Communication Protocol)

In order to extract data from the angular sensor, transmitted hexadecimal data from the sensor has to be interpreted using WITMOTION's communication protocol. This communication protocol explains how the data is deciphered in order to read the angular data. Each set of hexadecimal data output by the sensor consists of 11 bytes, all containing different types of information measured by the sensor as shown by **Figure 3.7**. Since the data of interest is the angle of the sensor, we want to access the 0x53 hexadecimal segment of the data that contains data such as roll, pitch, and yaw. The hexadecimal data that is transmitted by the sensor is sent directly to the master computer's USB port, which is a serial port. By using a Python library PySerial 3.4, Python code on the master computer can be used to read the pitch, roll, and yaw encoded in the hexadecimal data provided by the sensor.

By using the communication protocol and PySerial, we wrote a Python script that repeatedly captures and deciphers the angular data from the angular sensor. The process of finding and extracting the angular information from the data stream is illustrated in **Figure 3.10**. The block diagram illustrates how the data stream from the angular sensor is collected and interpreted by the Python code. It would continue to read the data until it encountered 5554 in hex code, which indicates when we have read enough data from the sensor to have captured the angular information that we desired. Once this point in the data was encountered, the Python code essentially stores 10 bytes of the data stream that holds all of the angular information. At this point, the script can extract the roll, pitch, and yaw into separate variables called Roll_Angle, Pitch_Angle, and Yaw_Angle. It is worth noting that the angular sensor outputs these data streams at a rate of 200 Hz[18] which is more than enough for our needs because our actuator pushes the dish at a

Protocol header	Data content	Data lower 8 bits	Data higher 8 bits	Data lower 8 bits	Data higher 8 bits	Data lower 8 bits	Data higher 8 bits	Data lower 8 bits	Data higher 8 bits	SUMC RC
0x55	TYPE【1】	DATA1L[7:0]	DATA1H[15:8]	DATA2L[7:0]	DATA2H[15:8]	DATA3L[7:0]	DATA3H[15:8]	DATA4L[7:0]	DATA4H[15:8]	SUMC RC【2】

【1】TYPE(Data content):

TYPE	Remark
0x50	Time
0x51	Acceleration
0x52	Angular velocity
0x53	Angle

Figure 3.7: Showing how Different Types of Information is Encoded in Streams of Hexadecimal Data [19]

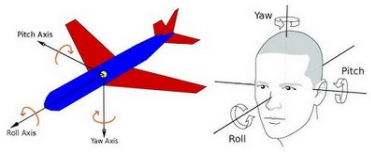

Figure 3.8: Image Illustrating Pitch, Roll, and Yaw [20]

much slower rate than the rate at which the angular sensor outputs data. The Python code that was created for this purpose is listed at the end of the thesis called **Python_Angle_Extraction_Code.py**.

3.3.3 Choosing Which Angle to Read and Use

There are three types of angular information provided by the sensor data stream: pitch, roll, and yaw. The data needed in order to tell the angle of the telescope is pitch, which is visually illustrated in **Figure 3.8**. As shown, pitch would be useful for illustrating the direction that our telescope is pointing into the sky. However, it is important to note that the types of angles are relative and for the way that the manufacturers chose to define pitch and roll shown in **Figure 3.9**. We installed the angle sensor on the radio telescope where the sensor's inclination changes with the radio telescope's inclination, the roll was essentially acting as our pitch. It is for this reason that when we extracted angular information from the sensor, we recorded the data that the sensor labeled as roll.

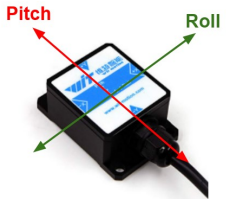

Figure 3.9: How Pitch and Roll are Defined with Respect to the Angle Sensor[18]

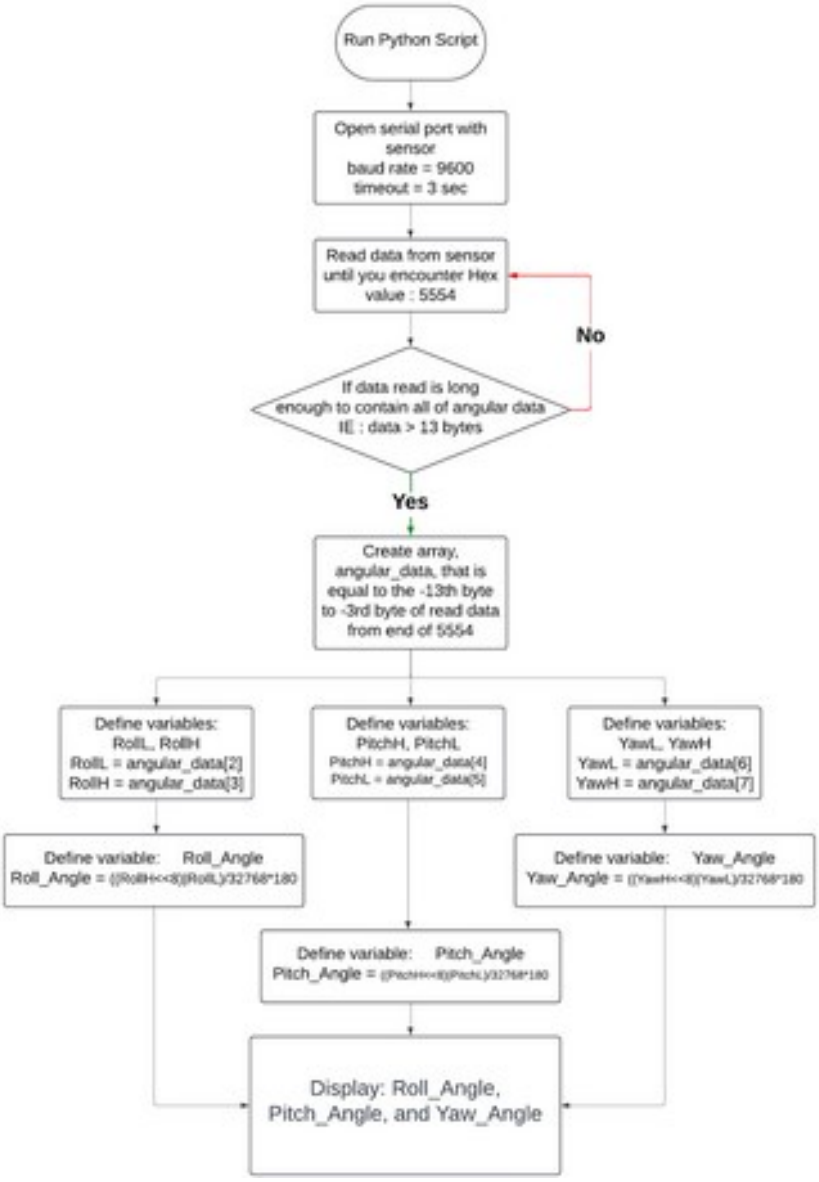

Figure 3.10: Block Diagram to Illustrate How the Angle is Extracted From the Data Stream of the Angular Sensor

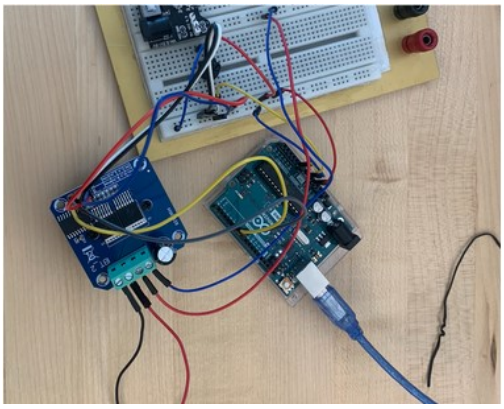

Figure 3.11: Two Main Circuits of the System Loosely Connected Before Placing Them in the Their Own Housing

3.4 Rotation Module

The entire rotational system is controlled electronically through code and a microcontroller. As a result, users of the system have no visual means to know what the system is attempting to do in real time. For example, if the code operating the motor thinks that it has stopped the motor but the motor is continuously operating, the user won't know that the system is malfunctioning. We also mentioned in our objectives the desire to create a plug and play system that is easy to operate. To address both of these issues, a rotation module has been created. The rotation module will house multiple components into one component, as well as provide real time diagnostics on the state of operation. In addition, rotation module will contain an analog kill switch to manually shut the system off if any issues arise. These will be described in more detail in following sections.

3.4.1 Housing of Critical Circuitry

The rotation system consists of two main circuits: the Arduino microcontroller and the H-Bridge. These two circuits originally had relatively loose connections as shown in **Figure 3.11**. This is a problem, as the wires can easily disconnect. When the circuitry of the system was first designed, it wasn't necessary to house them, as changes could have been necessary after conducting tests. However, after the testing was complete and the wiring finalized, the circuitry needed a place to be contained securely. The solution is the rotation module: a module that houses our rotational system's circuitry. To fit all components, the rotation module must have dimensions large enough to accomodate the Arduino microcontroller and the motor driver which have the following dimensions shown in **Table 3.1**.

Adding the dimensions of the two components, the housing must have a width of at least 105mm, a length of 120mm, and a height of 45mm. As a result, we selected a container with a length of 170mm, width of 130mm, and height of 79mm. This gives ample space for both components and the wiring throughout. The microcontroller and the motor driver can be held in place securely by using standoffs that are bolted through holes drilled in the bottom of the

Circuit	Width	Length	Height
Arduino Microcontroller	53mm	69mm	16mm
Motor Driver	50mm	49mm	43mm

Table 3.1: Dimensions of the two Critical Circuits of the Rotational System

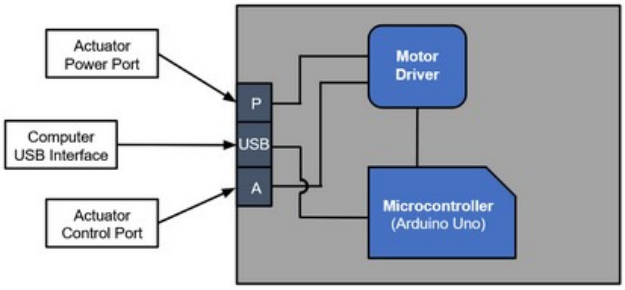

Figure 3.12: Rotation Module Layout of Motor Driver and Arduino Microcontroller

module. With these dimensions, the Arduino microcontroller and the H-Bridge are placed in the module as shown in **Figure 3.12**.

In order for the rotation module to be to interface with user commands and the telescope itself, it must have ports to connect to the microcontroller and motor driver. As a result, three ports must be installed into the rotational module which are shown in **Figure 3.12**. To connect the power needed to drive the actuator, one male C8 bulkhead labeled Actuator Power Port is drilled into the back panel of the container. Another male C8 bulkhead labeled Actuator Control Port is drilled into the back panel to transfer power from the H-Bridge to the actuator. The second type of port that is connected to the back of the panel is a USB-B bulkhead. The USB-B bulkhead will connect directly into the USB-B port of the Arduino microcontroller, and the other side will connect to the master computer via USB 3.0. The geometry of the USB and C8 bulkhead are square and capsule respectively, which cannot be cut with the drill press. To create these types of interfaces, we utilized the Wazer Pro Waterjet in Santa Clara University's Maker Lab [21]. The Wazer Pro Waterjet is a CNC machine that uses a high pressure water jet in addition to an abrasive to cut almost any material with precision. To use the waterjet, the ports were first modeled in CAD software. We designed our ports in Autodesk Fusion because it was familiar and licensed to students of Santa Clara University. Finally, the CAD file was processed in Wazer's Wazercam software, where we specified the coordinates dimensions of the design in order to create the job file the water jet can execute. **Figure 3.13** shows the completed back panel, with ports for our USB and C8 bulkheads.

3.4.2 LED System Indicators

For the users of the rotational system to know what the system is trying to do in real time, LED indicators have been installed onto the Arduino microcontroller. The are four different LEDs, indicating real time information as shown in **Table 3.2**. These LEDs are controlled by the Arduino microcontroller pins that write high signals to them

Figure 3.13: Finalized Back Panel of Rotation Module

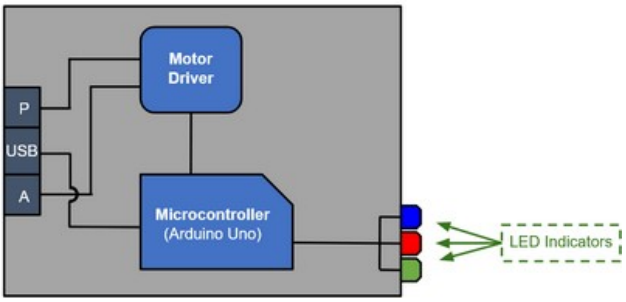

Figure 3.14: Layout of Housing Module Showing How LED Indicators Are Laid Out and Connected to Microcontroller

whenever the system is attempting to do a certain task. As a result, it gives the user a real time indicator as to what the system is trying to do, which is essential for troubleshooting and ensuring proper function. The LEDs have been laid out as shown in **Figure 3.14**.

LED Color	**Indication**
Green	Attempting to increase inclination by extending linear actuator
Yellow	Attempting to decrease inclination by retracting linear actuator
Red	Attempting to stop the linear actuator in place
Blue	Attempting to collect angular data to check the stability of angular sensor

Table 3.2: LED Indications and Corresponding Actions

The Arduino pins can only provide 3.3 V and 50 mA [14], as a result, resistors are needed to prevent sending too much current into them and burning them out. We decided to use 220 ohm 0805 packaged resistors because: the package is both small enough to fit on our PCB while being big enough to easily solder, and 220 ohms results in 15 mA of current through our LEDs, which is enough to be bright while not burning them out. In order to securely connect the LEDs to the Arduino with resistors in series, a PCB with surface mount resistors was installed.

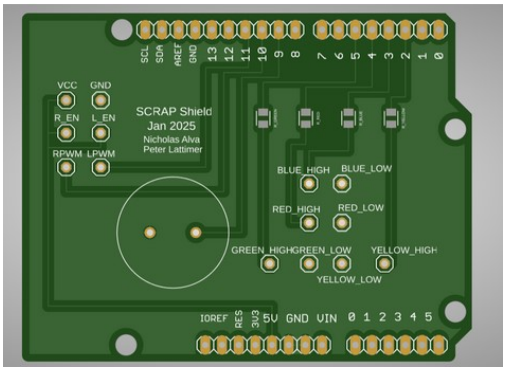

Figure 3.15: PCB Design Created Using Fusion 360 for Rotation Module

3.4.3 Arduino Shield

One of the tasks of the rotation module is to securely hold the circuitry of the system in place. To do this, a PCB was designed to connect directly to the pins of the Arduino microcontroller. This PCB was designed for the following reasons: to secure the wiring and prevent components from being easily disconnected, to hold resistors in place for LED system indicators, and to clean up wiring and label all connections

The PCB used to achieve the listed requirements above was designed using Autodesk Fusion 360 and sent to PCBway.com to be manufactured.Shown in **Figure 3.15**, all three requirements of the PCB are met, enabling a crucial step in the development of the rotation module. Also shown in **Figure 3.15** are visible lines that connect the numbered pins to the labeled holes on the board. For example, pin 11 has a thin wire connecting directly to the LPWM pin hole. These thin wires are copper traces that are imprinted into the boards and are clean and secure replacements for loose wires in the previous setup as shown in **Figure 3.11**. The pinholes have conductive metal around them allowing wires to be directly soldered into the board. These pinholes act as points of contact that can be soldered which secures electrical contact preventing any unintentional disconnections. Another important feature of the PCB is the 220 ohm 0805 packaged surface mount resistors that are soldered onto the board. These surface mount resistors are important as they enable the LEDs to connect directly to the PCB without needing additional nodes to add resistors in series. As a result, the wiring complexity is significantly reduced making the connections within the rotation module far cleaner. PCB's that securely fit on the Arduino are typically called shields and will be referred to as such from this point on.

3.4.4 Built in Speaker

Another analog indicator that has been built into the Arduino shield is a 5 V Piezo Speaker. The speaker has two prongs can be soldered into the shield within the white circle near the center of **Figure 3.15** and is activated when it receives a voltage from the Arduino. The piezo sits on top of the Arduino shield shown in **Figure 3.16**. This speaker is very loud and acts as a second form of an indicator to inform the user that the targeted angle has been achieved. This was introduced in case of an electrical failure from the LEDs which will eventually occur over long periods of time.

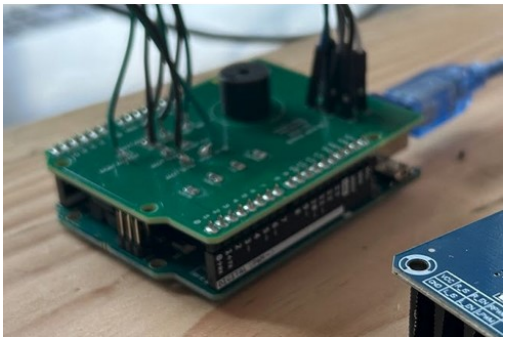

Figure 3.16: Piezo Speaker Soldered on Arduino Shield

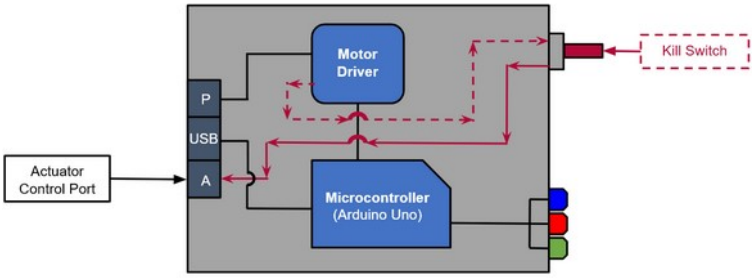

Figure 3.17: Configuration of Kill Switch Installed on Front Panel of Rotation Module Between Power Flow to Actuator Pin From Motor Driver

3.4.5 Analog Kill Switch

The last feature of the rotation module is the analog kill switch to stop the actuator in case of malfunction. A kill switch can be installed by implementing a switch in series with the actuator, resulting in either an open or closed circuit depending on the switch's position. The kill switch is important because it reduces the number of potential failures while being completely analog. The shutoff switch will be completely analog, providing the users direct control in real time in case of a potential malfunction. This kill switch was installed on the front panel as shown in **Figure 3.18**. The motor driver is what directs power to the actuator (connected to Box A) from the power source (connected to Box P) as shown by the red arrows in **Figure 3.17**. The dotted red arrow is the power coming from the motor driver that connects directly to the kill switch. The kill switch, when activated, allows the power to flow into the actuator itself as shown by the solid red arrow. As a result, whenever the users need to shut off the actuator, they can turn the switch off, creating an open circuit to prevent power flow. The rotation module and the circuitry it houses is shown in **Figure 3.19**.

3.5 Main Computer

The rotation system consists of many components, however they're only as good as their coordination. For coordination of functions and data in order to achieve proper rotation of the radio telescope, the main computer executes

Figure 3.18: Front Panel of Rotation Module with Kill Switch and LED Indicators

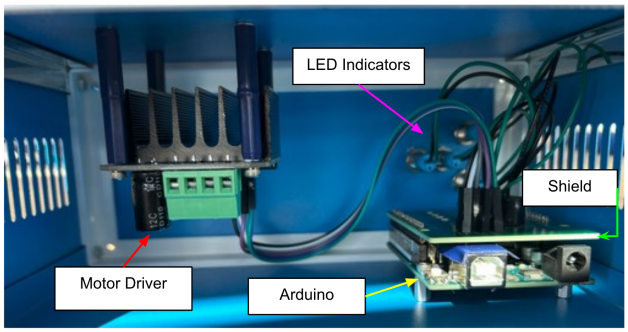

Figure 3.19: Internal Circuitry of the Rotation Module Showing Secure Circuitry and Clean Wiring

code.

3.5.1 System Requirements

The main computer needs to have the performance specifications needed to run Python and MATLAB scripts along with the supporting libraries needed to control the entire system. These system requirements for MATLAB would be the minimum computer specifications, such as the processor speed and storage of the system, which can be found at their website [22]. The other requirement of our system is that it must have at least four USB 3.0 Ports, as well as an HDMI or Display port. Two USB ports must be available for a keyboard and mouse so that the user can interface with the computer itself. One USB port is needed to connect to the angular sensor, and one for the Arduino. There must also be an HDMI or a display port to connect to the monitor so that the user can interact with the prompts from the Python script.

3.5.2 System Interface

The main computer has multiple functions which include executing Python and MATLAB scripts to run the system, powering the Arduino and the angular sensor, and acting as an interface between the user and the system. The interface is generated by a Python script designed to be simple in nature. As a result, the prompts provided to the user are extremely straightforward and use a .py from the Windows Command Prompt. Despite the simplicity of the interface, the system performs all tasks as designed. The main computer will sit next to the radio telescope as shown in **Figure 3.20**.

Figure 3.20: Main Computer Connected to Rotation System

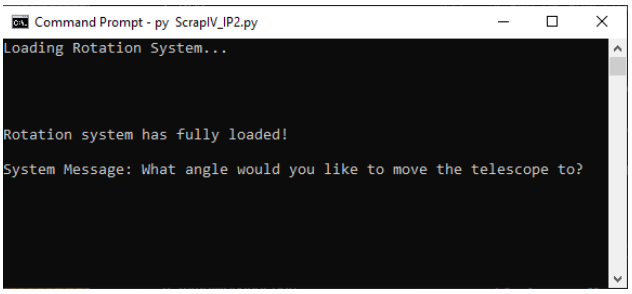

Figure 3.21: System User Interface Menu Options

As shown in **Figure 3.21**, the interface asks the user what angle to set the radio telescope to, while ensuring the telescope is not moving. If the user inputs an invalid input (a real integer inside the acceptable range), the code will crash and will require the user to restart the program. When the user inputs a proper angle, the Python script will move the telescope towards the entered angle until it has achieved that inclination. When this is complete, the shield will emit a noise using its speaker to indicate the angle has been reached. After the user inputs an angle into the interface, the Python script will utilize NumPy and SciPy in order to generate a .mat file that can be loaded by MATLAB. This .mat file generates a diagnostic file named **AngleTimePlotter.mat** that shows the radio telescope approaching the target as shown in **Figure 3.22**. This MATLAB script is important as it allows the users to assess how the rotation system is functioning with respect to the target and accuracy. The dotted line represents the target angle that was entered by the user at the interface and the thin line represents how the angle of the radio telescope was changing in real time. The script that generates the user interface is described in more detail in **Section 4.2 System Integration : Main Computer Integration**.

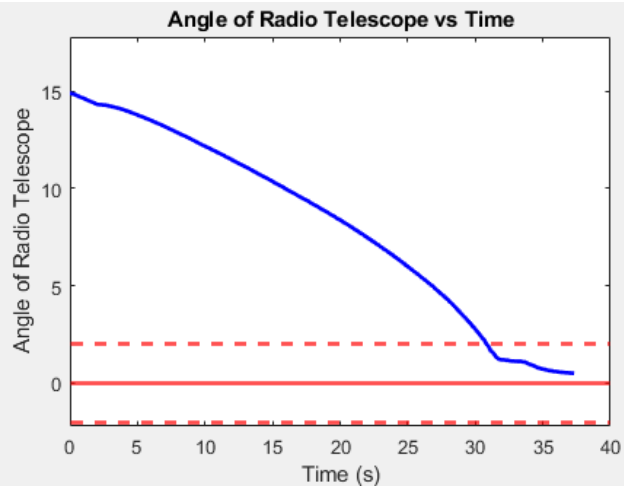

Figure 3.22: Plot Showing Inclination of Radio Telescope Approaching Target Angle

Chapter 4

System Integration

The rotation system consists of multiple components that need to work together in order for the system to operate as intended. This chapter describes how the components of the system are integrated together and why.

4.1 Rotation Module

As described in the previous chapter, the rotation module is a housing that simplifies the user experience by combining multiple components into one. It is responsible for integrating the main computer, motor driver, microcontroller, and linear actuator.

The rotation module further assists in the integration of these components because of clearly labeled ports. These are the USB-B port and the two C8 ports. With these, the user can simply connect the main computer to the USB-B port, the power cable into the power port, and the linear actuator to the actuator port shown in **Figure 3.13** . Since these ports are wired to the microntroller and the motor driver, the rotation module acts as a ”black box” with two inputs and one output. The two inputs are the input power from the actuator's power supply and the USB-B cable from the main computer where commands are sent to the microntroller. The output of the rotation module would be the power that is sent to the linear actuator which is controlled by the microcontroller illustrated by **Figure 4.1** and explained in further detail in **Section 5.2: Microcontroller Testing with Angle Sensor**. As a result, the rotation module plays a key role in integrating the main computer, microcontroller, motor driver, and linear actuator while also providing diagnostic visuals and safety features.

4.2 Main Computer Integration

The rotation system consists of several components that must work together to create a feedback network in order to change the inclination of the radio telescope in real time. To achieve this, the main component responsible was the main computer that is controlled by a Python script that interacts with the angle sensor(**ScrapIV_IP2.py**), linear actuator, and the Arduino microcontroller.

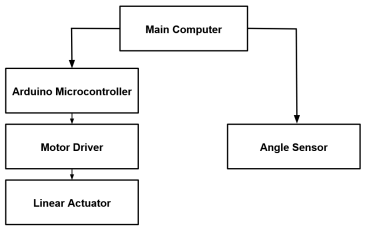

Figure 4.1: Block Diagram Showing how Main Computer Interacts with Linear Actuator and Angle Sensor

4.2.1 How the Main Computer Interacts with all Components

As previously mentioned in chapter 3 subsystems, the main computer directly connects to all of the components within the system as illustrated by **Figure 4.1**. The main computer connects to the microcontroller via USB 3.0. The microcontroller connects to the motor driver which is directly connected to the actuator. As a result, the main computer has a pathway of connection to the linear actuator through its connection to the microcontroller. This pathway enables the main computer to send commands to the linear actuator via the Arduino microcontroller. The angle sensor has a jumper cable adapter that allows its wires to plug into a USB 3.0 port which can be plugged directly into the main computer. This in turn enables the main computer to connect to the angle sensor to both deliver power and receive the angular data that is being measured.

4.2.2 Backend System Integration

The main computer is capable of communicating with all of the components of the rotation system, because of this, the main computer is the best component to facilitate the entire operation of the system. To facilitate the components of the system, a Python script was designed in order to take the angular data from the sensor to decide how to control the linear actuator while also controlling the LED and speaker. In order to better visualize how the Python script operates, a flowchart is shown in **Figure 4.2**.

1. **Establishing a connection to the sensor and microcontroller**

   Every time that the Python script is executed, it begins by loading pySerial and pyFirmata2 when mode_gta is initiated. The Python script loads pySerial so that it can listen to the USB port of the main computer that is connected to the angle sensor. By doing this, the script can listen to all of the reported angular data constantly provided by the angular sensor. The script then establishes a connection to the microcontroller by using the pyFirmata2 library so that the main computer can control the Arduino using Python commands. This in turn enables the main computer to control the linear actuator via Python commands.

2. **The Stability Test**

   Before the script begins controlling the linear actuator, it conducts an anglular sensor data check. The purpose of

Figure 4.2: Flowchart of Main Code that Controls the Rotation System

this test is to ensure that the rotation system does not make decisions based on unreliable angular data. For example, if the angle sensor was reporting extremely inconsistent values, the system would behave unpredictably and could malfunction. This safety feature is called a Stability Test. The Stability Test has two essential variables, the TOLERABLE_DELTA and the TEST_COUNT. The TOLERABLE_DELTA is a variable that represents the acceptable variation between the reported angles from the sensor. The Stability Test will stop the motor, which in turn will trigger the red LED, keeping radio telescope oriented at the same angle it was pointed to when given the command. The test will then trigger the blue LED while conducting the stability test so that the user can know that the system is conducting the test. The test will then take TEST_COUNT samples of the angle reported by the angle sensor and will store each value in an array of size TEST_COUNT. The Stability Test will then compare each value to each other and take the magnitude of their difference. If the magnitude of the difference is greater than the TOLERABLE_DELTA, the script will reset the stability test and repeat the cycle. If the magnitude of the difference is equal to or less than TOLERABLE_DELTA TEST_COUNT times, then the stability test will have been considered successful and the cycle will end allowing the next phase of the script to proceed.

3. **The Main Operation (mode_gta Function)**

   After the Stability Test has reported a success, the main function that changes the inclination of the radio telescope begins to execute. The main function will begin by creating two arrays, one array will store each angle that the function records while changing the inclination as well as the amount of time that has passed into the other array. The data from these arrays will then be stored into a .mat file for the users to be able to load with MATLAB later to graphically observe the performance of the system. The MATLAB script that generates this plot is shown in **AngleTimePlotter.mat**.

   After creating the two arrays, the mode_gta function will check the current inclination of the radio telescope. A variable is created, called desa, which refers to the desired angle that the user would like to point the radio telescope to. The function will then round the current angle to a whole integer. This is done to prevent the system from being stuck in an infinite loop as it is virtually impossible for the recorded angle and all of its decimals to be exactly equal to the entered target angle. By rounding the current angle up to the closest integer, it allows the system to be able to fall within 0.5 degree range of the desired angle. After rounding the current angle, the system will compare the current angle to desa. If the current angle is equal to desa, the function will stop the motor in its place and trigger the red LED. If the current angle is greater than desa, the function will make the linear actuator retract decreasing the inclination while activating the yellow LED. If the current angle is less than desa, the function will cause the linear actuator to extend, increasing the inclination while triggering the green LED. Whenever the current angle is not equal to desa, it will either increase or decrease inclination

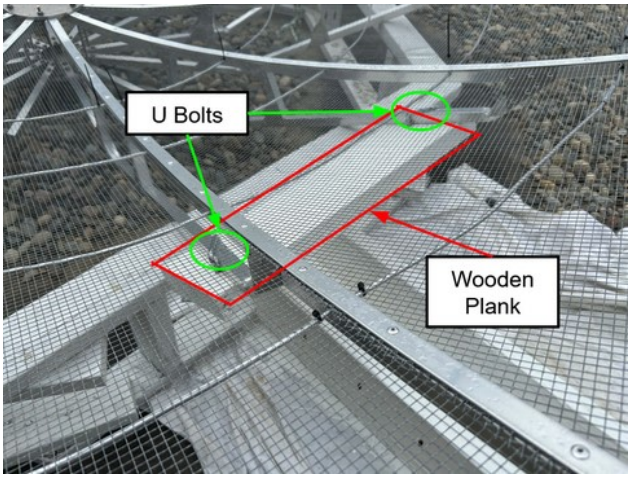

Figure 4.3: Contact Point for Linear Actuator on Radio Telescope

as mentioned above but it will also repeat the cycle until the current angle equals desa. When the current angle equals desa, it will not only stop the motor but it will also trigger the speaker to give an audio indication to the user that the desired angle has been achieved.

4.3 Supporting Structure

As shown in **Chapter 1**, the SCRAP-II team built a wooden structure to hold the radio telescope. This structure is sturdy and perfect for taking stationary data, however modifications were needed to enable the radio telescope to safely rotate. Enabling rotation required two additions:

1. **The Contact Point:** Where we apply force to rotate the radio telescope. This is where the actuator will connect to the antenna in order to push and change its angle.

2. **Hinge Point:** A stationary anchor of which the radio telescope is free to rotate on. By anchoring this securely, we make sure the antenna is free to rotate around this point, but will not move in any other direction.

4.3.1 Point of Contact for Actuator

For the contact point, it was important to ensure that the force applied by the actuator would not generate any unnecessary stress on the radio telescope. To solve this, we implemented a design that would span two steel arms of the radio telescope, in order to disperse the actuation force over a larger area. By doing this, we ensure that the radio antenna is not enduring any excessive stress that would, over time, degrade its structural integrity. Since the demo structure was designed with the same dimensions as our radio antenna, the distance that the actuator needed to be placed from the radio telescope was already known. The point of contact was implemented using the configuration shown in **Figure 4.3**.

Figure 4.4: Wooden Base for Linear Actuator to be Mounted to While Pushing Radio Telescope

The point of contact for the actuator only consists of a wooden plank, U-bolts, and bike tire tubes. The wooden plank is attached to the radio telescope's metal frame by being bolted onto it with metal U-bolts. These U-bolts were chosen as their grip around the frame can be adjusted on the fly by simply tightening the nut that secures them. Between the U-bolt and the wooden plank, bike tire tubes were placed in order to maximize the friction between the point of contact and the radio telescope. This was essential as it prevents the wooden plank from sliding up and down the radio telescope's metal frame. Once the wooden plank was locked into place through the use of U-bolts, the front end of the linear actuator was bolted to it using screws provided by the manufacturer.

The second component that the point of contact for the actuator needs is a wooden base for the linear actuator to be connected to while it pushes on the radio telescope. This is necessary as the linear actuator will experience the same force that it applies to the radio telescope in the opposite direction. To provide this wooden base for the linear actuator to be mounted to, we created the wooden base shown in **Figure 4.4**. The wooden base for the actuator is bolted to the wooden base that the radio telescope rests on. When the linear actuator is secured to its wooden base, it is still capable of leaning due to the weight of the radio telescope. If the linear is allowed to lean, this would cause the linear actuator to push the radio telescope off of the base instead of upwards. To prevent this from happening, two small wooden blocks have been bolted to the wooden base of the actuator shown in **Figure 4.5**. These wooden blocks surround the linear actuator, only allowing the linear actuator to move in the direction that it is pushing the radio telescope to move. If the linear actuator begins to experience forces that would cause it to lean to the right or left, the wooden blocks will simply not allow it to move in the direction of lean.

Since the contact point for the linear actuator is a flat wooden board that lies against the radio telescope, the inclination of the wooden board changes the same way the radio telescope does. We take advantage of this fact by bolting the angular sensor to the wooden board shown in **Figure 4.6** and recording the angle of sensor when the radio telescope is at its resting position(when the linear actuator is fully retracted). This recorded angle is approximately 230

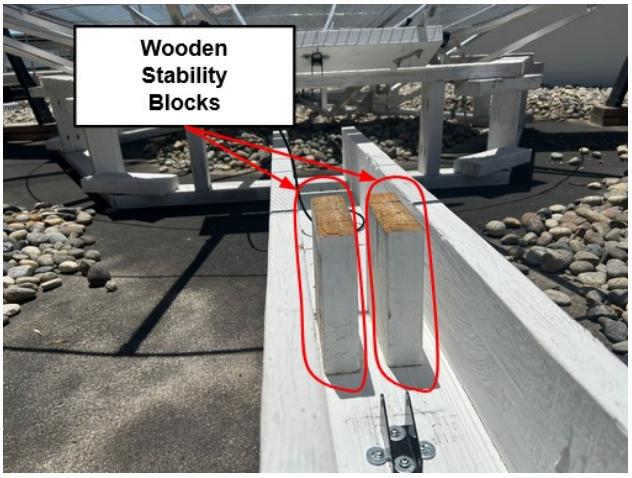

Figure 4.5: Wooden Blocks to Prevent Linear Actuator from Leaning to its Side

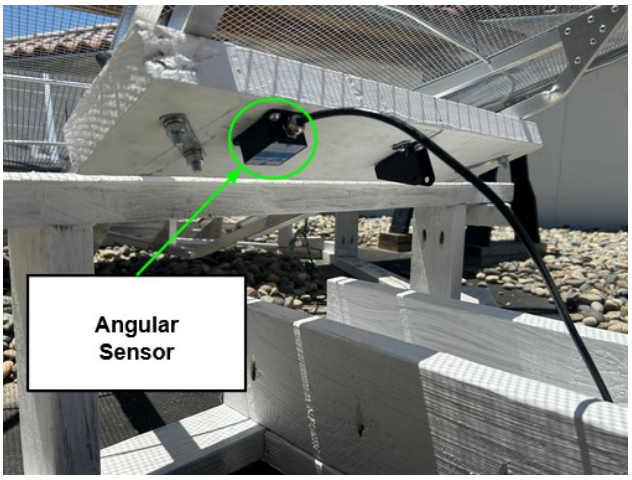

Figure 4.6: Angular Sensor Mounted to Contact Point for Actuator

degrees and as a result, any angle that is reported by the angular sensor is subtracted by 230 degrees to normalize it's position. After the angle reported by the angle sensor is normalized, it is used by the rotation system script described in chapter 4.2 to determine how the linear actuator should adjust the inclination of the radio telescope.

4.3.2 Hinge Point

For the hinge point, we needed to make sure of two things:

1. The actuator cannot push the radio antenna away, which could lead to instability and safety concerns
2. There is no side to side give. Side to side movement would put stress on the base of the actuator and could eventually cause it to give way under the weight of the radio telescope.

To achieve this, the design was broken into two sections. The first section was the axle of rotation and the second section was the point of contact between the radio telescope. The first section of the hinge point is the axle of rotation.

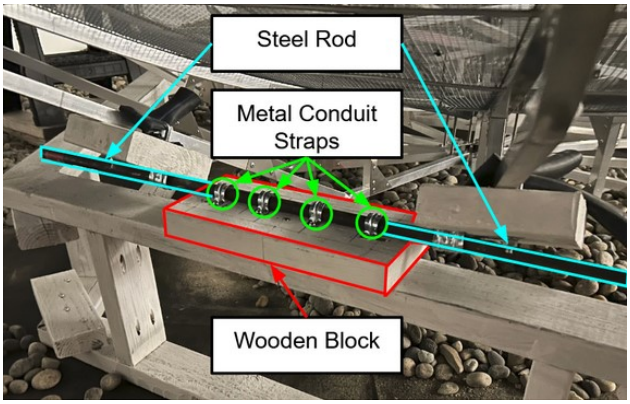

Figure 4.7: Axle of Rotation for the Hinge Point

This section consists of a wooden block, four metal conduit straps, and a steel rod. To illustrate how these components are utilized to create an axle of rotation, **Figure 4.7** can be observed. By observing **Figure 4.7**, all of the components of the axle of rotation can be seen. The wooden block, shown in red, is secured to the wooden base of the radio telescope with 2-½" bolts. It is important as it provides space between the radio telescope and the wooden base so that the radio telescope can rotate without colliding with the base. The metal conduit straps, shown in green, are used to secure the steel rod, shown in blue, in place. The metal conduit straps are sized and positioned in a way that prevents the steel rod from twisting while stilling allowing it to have room to rotate. This design holds the axle of rotation in place for our rotation telescope to rotate on while the second section connects the radio telescope to it.

The second section, being the contact point between the telescope and the hinge point, consists of two wooden blocks, two U-bolts, and four metal conduit straps as illustrated by **Figure 4.8**. All three components of the contact point for the hinge point are highlighted. The U-bolts highlighted in green grip onto the metal frame of the radio telescope itself. These U-bolts are connected to the wooden blocks, as shown in red, allowing them to secure themselves to the radio telescope. With these wooden blocks attached to the radio telescope scope, the metal conduit straps, shown in blue, were attached to them using screws where the conduits were able to grip onto the axle of rotation from the first section. With these conduits gripping onto the axle of rotation and the metal frame of the radio telescope, the radio telescope can pivot on the axle of rotation in the same way that a door can pivot on a door hinge.

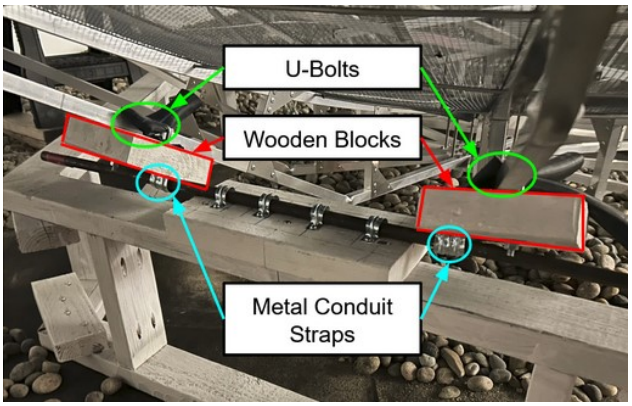

Figure 4.8: Contact Section of the Hinge Point

Chapter 5

Testing and Results

With the components of the rotation system designed, it is essential that they're tested before they could be fully integrated. This is important as it ensures that everything is performing as expected while reducing possible sources of error when troubleshooting the system. This chapter covers testing of individual components as well as the complete system.

5.1 Angle Sensor Testing

When testing the functionality of the angle sensor for the rotation system, there are two main features that are of concern. The two concerns being the readability of the information and the accuracy of the sensor itself.

5.1.1 Testing the Readability

In order for our system to interact with the angle sensor in a meaningful way, the main computer must be able to interpret the digital information as values of inclination. To determine the readability of the angle sensor, the sensor was connected directly to the main computer's USB 3.0 port using the angle sensor's USB adapter as shown in **Figure 5.1**.

As previously mentioned in Chapter 3, a communication protocol is used in order to decipher the digital information being sent to the main computer. Using this protocol, it must be determined that the information that the computer

Figure 5.1: Test Configuration of Angle Sensor Connected to Main Computer via USB 3.0

Position (Orientation) of Angle Sensor	Angle Reported by Angle Sensor (Degrees)
Flat	0
Upwards	90
Upside Down	180
Downwards	270

Table 5.1: Test of Different Reported Values from Angle Sensor Vs. Different Positions

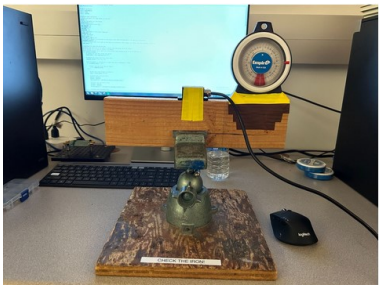

Figure 5.2: Test Device Created to Test the Accuracy of Angle Sensor

is interpreting is actually the inclination of interest. When conducting this test, the angle sensor was placed flat on the testbench where it wouldn't move. The main computer was listening for the data sent by the angle sensor and printing the inclination on the monitor for it to be observed. There were four different positions that this test was conducted with and the corresponding results are listed in **Table 5.1**. The data provided by the angle sensor to the main computer was consistent with the visual inclination of the angle sensor. As a result, this test demonstrated that the data provided by the angle sensor was consistent and made sense.

5.1.2 Testing the Accuracy

After confirming that the angle sensor is sending the main computer information that correctly corresponds to the inclination of the angle sensor, the accuracy of the angles reported must be tested. The objective of this test was to determine if the accuracy of the angles reported by the angle sensor had an accuracy of at least +/- 3 degrees.

To conduct this test, a makeshift angle tester was created to slowly and securely change the angle of the angle sensor. A flat piece of plywood was clamped into place so the angle sensor could rest on top of it as well as an analog angle protractor so they could be at the same inclination as shown in **Figure 5.2**. Since the manufacturer of the analog angle sensor reported an accuracy of +/- 2 degrees, this implies that if the angle sensor reports an angle that agrees with the analog angle sensor, the angle sensor would have an accuracy of at least +/- 2 degrees. This level of accuracy is enough to achieve the desired minimum accuracy of at least +/- 3 degrees.

To determine if the angle sensor and the analog angle protractor were in agreement, the inclination of the wooden board was changed to varying angles. At every different inclination of the wooden board, the data reported by the angle sensor and the analog protractor were manually recorded. These values were then rounded to the closest integer value of a degree. Data from this test are shown in **Figure 5.3**. The plot from the test shows two lines. The solid blue

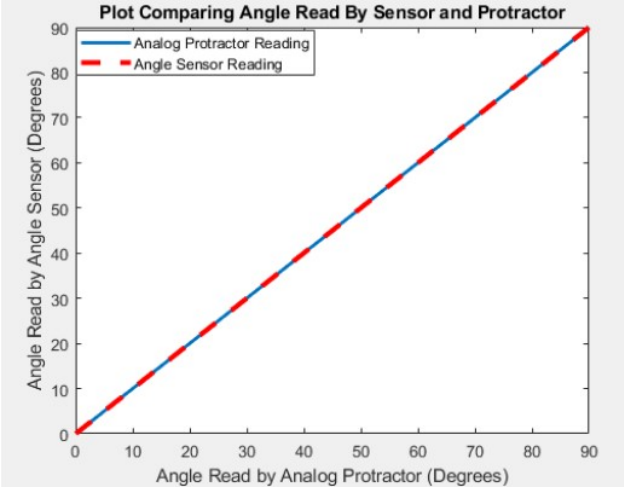

Figure 5.3: Plot Showing Inclination Reported by Angle Sensor vs Angle Protractor to Test Accuracy of Angle Sensor

line represents the angles reported by the analog angle protractor and the dashed red line represents the angle reported by the angle sensor. It is important to note the lines are solid because matlab is interpolating from each measured point. This was done to make it visually easier to compare the two inclinations against one another. As can be seen, the two lines overlap perfectly indicating that the analog angle protractor and the angle sensor report the same inclination. This test demonstrates that the angle sensor has a tested accuracy of at least +/- 2 degrees.

5.2 Microcontroller Testing with Angle Sensor

After confirming that the angle sensor is properly sending angular information to the main computer, a test needs to be conducted to ensure that the Arduino could properly interact with the angle sensor. This is important as the actuator is controlled by a motor driver that is directly controlled by the Arduino microcontroller. Since the movement of the actuator is a function of the inclination of the radio telescope and the desired angle, it is essential that the microcontroller can interact with the angle sensor.

5.2.1 Arduino and Angle Sensor Test

The main objective of this test is to determine if the Arduino microcontroller can demonstrate specific actions depending on the inclination of the angle sensor. To test this, both the angle sensor and the Arduino are connected directly to the main computer via USB 3.0 ports. The Arduino was programmed to control four different LEDs to simulate the electrical signals that it would send to the actuator to control it. The color of the LEDs shown in **Table 5.2** indicates which direction the Arduino would tell the actuator to move in order to achieve the target angle.

Using the same device test device that was used for testing the accuracy of the angle sensor, the angle of the angle sensor was manually controlled by hand to simulate the actuator pushing the radio telescope. The objective was to

LED Color	Indication
Green	Extend Linear Actuator Change inclination in positive direction
Yellow	Retract Linear Actuator Change inclination in negative direction
Red	Stop the linear Actuator
Blue	Conducting Stability Test

Table 5.2: Color of LEDs and what Actions they Simulate

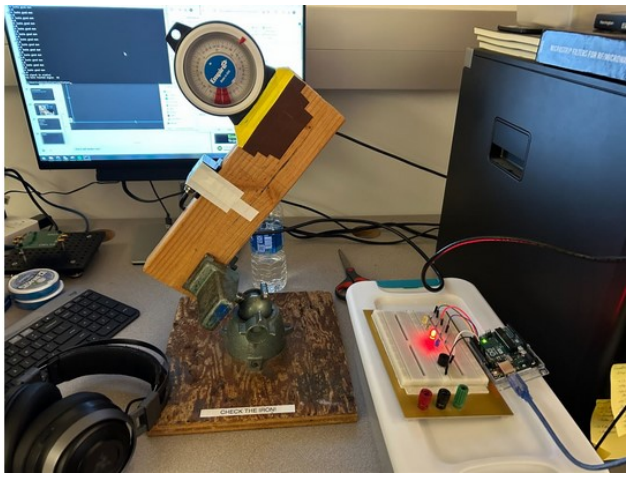

Figure 5.4: Test Setup For Testing Angle Sensor and Arduino

input a desired angle into the code controlling our test and to change the inclination above and below the desired angle to trigger the Arduino to activate each LED. The set up we used to conduct this test is shown in **Figure 5.4**. The first test that was conducted was to see if the main computer could run a stability test with the angle sensor. While the stability test is being conducted, the Arduino should activate the blue LED. After conducting this test, the blue LED did activate which was a success. The next tests were to check if the green, yellow, and red LEDs were working when they should. When the angle sensor was oriented to an angle higher than the desired angle, the yellow LED activated indicating a successful test of the yellow LED. When the angle sensor was oriented below the desired angle, the green LED activated as desired, which indicated another successful test. Finally, when the angle sensor was oriented to the desired angle, the red LED activated as it was supposed to, hence it was a successful test.

It is important to note that the LEDs are simply a visual indicator to determine that the Arduino is behaving correctly in accordance with the data provided by the angle sensor. After conducting this test, it was confirmed that the microcontroller and the angle sensor were properly interacting with each other as they will need to in the realized rotation system.

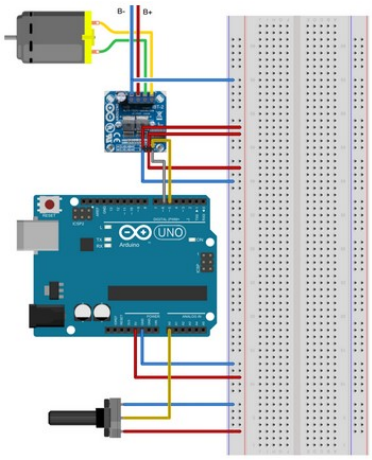

Figure 5.5: Controlling Motor Driver Using Arduino[16]

5.3 Actuator Testing

In testing the actuator, we decided the best way to build a deep understanding of the mechanisms and structure was to start with a basic prototype and build up. The advantages of this strategy are starting with components we know how to connect and operate, and ease of testing. By starting with a minimal circuit we know works, we were able to add in our final components one by one, making debugging and testing easier.

5.3.1 Motor Driver Code Development

To start, we built a system included in our BTS7960 Motor Driver Manual shown below in **Figure 5.5**. This system provided us with a perfect starting point for the actuator-Arduino interface. Not only did it provide us with the wiring diagram, this system also provided us with the Arduino code to run this system that would set the groundwork for our final code. This system includes the Arduino, as well as the BTS7960 Motor Driver however does not include the main computer or the actuator. As shown in the wiring diagram, instead of the actuator, the system uses a simple 5V DC Motor. The reason we start with the 5V DC Motor is that the Arduino can supply 5V, however it cannot supply the 12V our actuator requires. The other simplification is the use of the potentiometer. In this circuit, the potentiometer serves the purpose of the main computer. It does so by varying its resistance, which will be read by an analog to digital converter inside the Arduino and then written to the motor driver.

The next step was incorporating the main computer. As mentioned, the above schematic utilizes a variable resistor potentiometer in order to control movement of the motor.

The final element we needed to incorporate into the system was the actuator. Again because the Arduino can only provide 5 V, less than the 12 V needed for the actuator, we needed to route power from the wall. This was made easy through the supplied AC-DC converter, which takes power from the wall socket (120 V AC) and converts it into 12 V 2 A power needed for the actuator. The actuator uses two-prong connectors that, when cut off and stripped, reveal

a positive and negative wire. Feeding the two wires from the AC-DC converter into the 'power in' terminals on our motor driver, and the two wires from our actuator into the 'power out' terminals, we have our system fully operational with all final parts.

5.4 Rotation Module

The rotation module is a conglomerate of the actuator system, safety features, and diagnostic tools. This means that rotation module testing only required testing the kill switch and the diagnostic LEDs as the motor driver had been tested with the linear actuator in a previous section. We had already tested the software of the LED's in prototypes described in previous sections, however we had to make sure our new waterproof LEDs mounted on the front panel of our rotation module function the same. To test them, we ran the code through all modes of operation (stability test, ascension, dissension, final position) to make sure that all four LEDs operate as intended. We found that our new LEDs functioned as designed.

Testing the kill switch was straightforward: execute the code, and check whether the switch quickly and effectively shuts down the system. We performed this safety check under all conditions, and confirmed that the kill switch effectively shuts the system down under any and all conditions.

The final check we had to make in the rotation module was the mounting of the components. As previously stated in chapter 3, we want our rotational module to be robust. This means we need our components to be mounted inside so they cannot move around and potentially come loose. To mount the motor driver and Arduino, we used standoffs that could be drilled through the steel module to secure the components in place. As the two components contain exposed metal pins, we had to make sure that our metal pins would not short pins and potentially cause damage. The only other thing we had to be cautious of is the placement of the motor driver. The motor driver runs relatively high current at 2A, and if running for long periods can give off lots of heat. This is why it has these large heat sync that can be observed in **Figure 3.19**. For precautionary reasons, we made sure our motor driver had plenty of space so heat can effectively dissipate and not cause any issues with other system components. After successfully mounting both components, we ran the system and confirmed that everything was functioning properly.

5.5 Test Demo Structure

To avoid complications that could arise from installing our system onto the radio telescope structure that we have, we decided to first build a demo structure to test our rotation system. It is important to note that this system did not include the installation of our modular housing for our rotational control system but it did have the rotational control system itself.

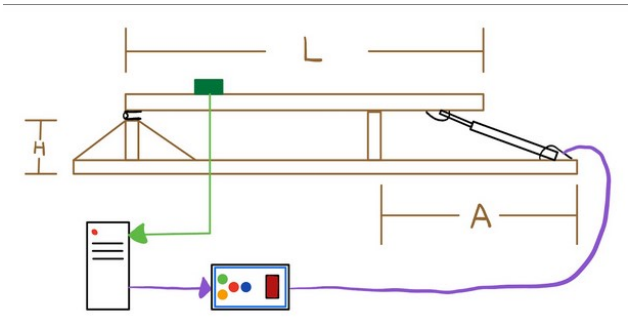

Figure 5.6: Redesigned Demo Structure Design Sketch

5.5.1 Demo Structure

When building this system, we wanted our design to mimic the dimensions of the radio telescope structure that we had as possible so we wanted to use important dimensions of our radio telescope. These dimensions are shown in **Figure 5.6** alongside **Table 5.3**. From these measurements, we found that the height from the ground to the edge of the radio telescope that we're applying our force to was approximately 13 inches. We also found that the distance from the pivoting point to the point of contact between where our actuator would be was approximately 72 inches.

Dimension	**Dimension Variable**	**Dimension Length**
Height Radio Telescope is above Ground	H	13 in.
Distance from Pivoting point to point of contact	L	72 in.
Distance Actuator Connection is from Radio Telescope	A	32 in.

Table 5.3: Initial Design Concept Sketch of our Demo Structure

It also had the added benefit of allowing our linear actuator to be mounted onto the same wooden plank as the entire structure removing the need for an additional wooden plank. This wooden plank that the actuator would is connected to the same board that is acting as the base of the demo structure.

We obtained the supplies needed to build it from our local Home Depot and some available supplies from our RF Lab in Santa Clara University. These supplies included 2 pieces of 10ft 2x12 lumber, a 4 inch door hinge, a pocket hole jig, 2-½ inch zinc coarse zinc plated screws, and a saw. Using these supplies and the guidance of our advisor, Dr. Kurt Schab, we were able to successfully construct our redesigned demo structure shown in **Figure 5.7**. With this Demo Structure, we were able to create a testbench for the rotation system where we could test any code that we created before having the final structure installed onto the radio telescope. This demo structure also provided us with an experimental platform that aided in the design process.

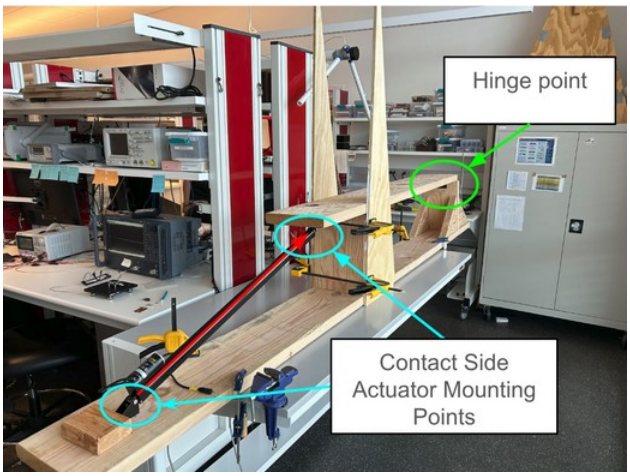

Figure 5.7: Test Demo Structure

5.6 The Rotation System

After the demo structure demonstrated that the components of the system were working in tandem, we began installing the rotation system onto the radio telescope.

5.6.1 Testing the Hinge Point

The main structure to be tested was the hinge point of the system. To conduct this test, we simply took turns manually lifting the side of the radio telescope by the linear actuator point of contact (wooden plank attached to the metal frame of the radio telescope). We simply lifted the telescope by hand at different angles and observed the integrity of the hinge point. After conducting our quick test of the hinge point, we found that the radio telescope didn't slide along its frame confirming that the points of contact were secure as desired. We also observed that the steel rod that acted as the axle of rotation supported the entire structure without any signs of failure. Overall, it was observed that the structural integrity of both the points of contact and the hinge point were successful.

5.6.2 Testing the Final Rotation System

To test the entire system, all of the components of the rotation system were installed onto the radio telescope which include the rotation module (which houses the microcontroller and motor driver), the linear actuator, the angle sensor, and the main computer. Since the test demo structure was designed to have the same physical dimension of the radio telescope, we expected to observe the following results from our test shown in **Table 5.4**.

When conducting the test, we utilized our main computer that controlled the entire system as designed. It was known by the length of the actuator and geometry of the radio telescope that it couldn't be oriented 90 degrees from its zenith. We took advantage of this limitation by simply entering 90 degrees into the system so that it would push our radio telescope to its maximum inclination for us to test. The system that we utilized to operate the telescope

Test Parameter	Expected Result
Angle of Rotation	~ 30 Degrees
Angular Velocity	~ 18 Degree/min
Mode of Rotation	Angular Sweep
Uncertainty of Angle	± 2 Degrees

Table 5.4: Test parameters and their expected results

Figure 5.8: Radio Telescope at Maximum Inclination

and conduct this test is shown in **Figure 3.20** which shows the main computer operating the rotation system python script that is described in **Section 4.2 Main Computer Integration**. 90 degrees was entered into the system using this computer and the diagnostic lights were in operation with the rotation module shown next to the main computer. We observed that when the radio telescope was ascending, the green light activated, which is what we expected. After waiting some time, the linear actuator reached its maximum actuation length and the maximum inclination of 29 degrees was reached and shown in **Figure 5.8**. It is important to note that although the maximum angle of inclination was 29 degrees, the range of the angle is still 30 degrees as observed from our test demo. The reason that it is different is because the hinge point is slightly taller than the contact points of the linear actuator. As a result, when the linear actuator has applied force to the system, it is essentially pointing 1 to 2 degrees in the opposite direction; this position is what we call the starting position. This is to be expected and still results in the angular range of approximately 30 degrees as desired.

To test that the rest of the diagnostic lights were working correctly with the rotation system, the rotation system was instructed to move the radio telescope back to the starting position. While observing the radio telescope descend, the yellow LED activated as desired. When the radio telescope reaches its target position (the starting position), the red LED activates, stopping the system and indicating that the radio telescope stopped at the target angle it was instructed to point to. We also tested the blue LED by activating the STABILITY_TEST function that is described in **section 4.2) Main Computer Integration**. It was observed that the blue LED activated and that the angles reported by the sensor

Figure 5.9: Setup for Testing the Accuracy of the Radio Telescope with the Target Angle of Fifteen Degrees

were in fact stable when reported as such.

Finally, to test the accuracy of the system, the rotation system was instructed to orient the radio telescope at given angles. When the radio telescope finished moving to the target angle, the angle was measured using an analog protractor on a wooden plank shown in **Figure 5.9**. The wooden plank acted as a flat surface that the protractor could rest on where the plank was perpendicular to the zenith of the radio telescope. When the radio telescope reported that it has reached the target angle, we measured the angle of the telescope and compared it with the sensor and found that they agreed with one another. For one of the tests, we set the target angle to fifteen degrees and measured fifteen degrees with the analog protractor as shown in **Figure 5.10**. This test confirmed that the rotation system achieved the accuracy of at least +/- 2 degrees as desired. Since the radio telescope was capable of moving between different desired angles as we instructed it to, we were also able to confirm that the rotation system achieved the angular sweep that we desired.

Figure 5.10: Value of the Analog Protractor when Testing the Rotation System at Fifteen Degrees

Chapter 6

Conclusion

As mentioned in **Chapter 1) Introduction**, SCRAP is a multi-year project that has consisted of many teams that have all moved the project forward in its capabilities. We were not the only team to work on this project this year as another SCRAP-IV team worked on the radio telescope's ability to store observed data over long periods of time. After applying our work and design, we have enabled the radio telescope to be capable of moving in a way that can allow it to capture enough data for 2D maps of the sky.

Despite the work done with SCRAP, we believe there is still room for future work on the system. This would include improved software to indicate what objects and phenomena that the radio telescope is observing onto a 2D map. We also foresee that a lot of the work of the future SCRAP group will consist of fine tuning the systems that are already present. This would include the rotation system itself and its accuracy which is extremely important in radio astronomy. If a future group was to work on adding another form of movement for the rotation system, they could add both cardinal direction and multiple axis of rotation. This would enable SCRAP to be able to track specific objects in real time to observe their intensity and movement over time. This would further enhance the capabilities of the project and enable full range of motion. Another addition to the project we think would greatly increase interest in the field is a live monitoring system displaying captured data in the Electromagnetics Lab.

Overall, this project was extremely rewarding and exciting. We hope that our contributions to this project will enable astronomers and students interested in electromagnetics to learn more about the field and collect data first hand.

Chapter 7

Appendices

7.1 ScrapIV_IP2.py

```
# This is the code that we're writing in order to make our gta_function be its own function instead of a

import pyfirmata2
import time
import serial
from scipy.io import savemat
import numpy as np

# Timeout value to determine when you want the function that reads data to wait until it gets what it ne
global timeout_time
global timeout_counter
global timeout_flag

timeout_time = 1000
timeout_counter = 0
timeout_flag = False

# Norrmmalizaationn factor
# Experimental value that will give you the true angle is 205~206, so use these values when using the sy
# If you want to return the system back to it's resting position, set to  203 then set target angle to 0
normalization_factor = 203

# Inform user that the system is beginning to load
print('\n\n\nLoading Rotation System...')

# Create a variable in order to define which ports to use
arduino_port = 'COM3'
sensor_port = 'COM6'

# Create the arduino object board
board = pyfirmata2.Arduino(arduino_port);

# Pin connected to motor driver set up as pwm outputs
RMB = board.get_pin('d:10:p') #reverse motor bias
FMB = board.get_pin('d:11:p') #forward2
```

```
#LED Pins for System Diagnostics
PIN_GREEN = board.get_pin('d:5:o')
PIN_RED = board.get_pin('d:4:o')
PIN_YELLOW = board.get_pin('d:2:o')
PIN_BLUE = board.get_pin('d:3:o')

# Pin connected to the sound component
SOUND_PIN = board.get_pin('d:9:o')

# Time variables
short_time = 1
very_short_time = 2

# BEGINING OF FUNCTION DEFINITION

def light_pattern():

 # Pretty light pattern
 PIN_BLUE.write(1)
 time.sleep(0.1)
 PIN_BLUE.write(0)
 PIN_RED.write(1)
 time.sleep(0.1)
 PIN_RED.write(0)
 PIN_GREEN.write(1)
 PIN_YELLOW.write(1)
 time.sleep(0.3)
 PIN_GREEN.write(0)
 PIN_YELLOW.write(0)
 PIN_RED.write(1)
 time.sleep(0.1)
 PIN_RED.write(0)
 PIN_BLUE.write(1)
 time.sleep(0.1)
 PIN_BLUE.write(0)


# END OF FUNCTION DEFINITION

light_pattern()

# BEGINING OF FUNCTION DEFINITION

def FWB_motor(board):

 # Deactivate Stop motor
 PIN_RED.write(0)
 # Deactivate the Reverse movement
 PIN_YELLOW.write(0)

 RMB.write(0)
 FMB.write(1)

 # Activate the Green LED to indicate the motor is pushing forward (up)
```

```
 PIN_GREEN.write(1)

# END OF FUNCTION DEFINITION

# BEGINING OF FUNCTION DEFINITION

def RVB_motor(board):

 # Deactivate Forward movement
 PIN_GREEN.write(0)
 # Deactivate stop movement
 PIN_RED.write(0)
 FMB.write(0)
 # Activate the Green LED to indicate the motor is pushing forward (up)
 PIN_YELLOW.write(1)
 RMB.write(1)

# END OF FUNCTION DEFINITION

# BEGINING OF FUNCTION DEFINITION

def STOP_motor(board):

 # Deactivate Forward movement
 PIN_GREEN.write(0)
 FMB.write(0)
 # Deactivate the Reverse movement
 PIN_YELLOW.write(0)
 RMB.write(0)
 # Activate the red LED to indicate the motor has been stopped
 PIN_RED.write(1)

# END OF FUNCTION DEFINITION

# BEGINING OF FUNCTION DEFINITION

def sound_success(board) :

 # Turn the speaker on
 SOUND_PIN.write(1)
 # Wait 0.25 seconds
 time.sleep(0.25)
 # Turn the speaker off
 SOUND_PIN.write(0)

 # Wait 0.25 seconds
 time.sleep(0.25)

 # Turn the speaker on
 SOUND_PIN.write(1)
 # Wait 0.25 seconds
 time.sleep(0.25)
 # Turn the speaker off
 SOUND_PIN.write(0)
```

```
# END OF FUNCTION DEFINITION

# BEGINING OF FUNCTION DEFINITION

def serial_getRoll():

 global timeout_counter, timeout_flag

 # Open the serial port
 ser = serial.Serial(sensor_port, 9600, timeout = 5)

 # Listen for the data fromt the sensor
 heard_data = ser.read_until(b'UT')

 # Define a variable that will hold the angle of roll
 angle_roll = -1

 # Loop that will continue to occur until the function got the data that it needs

 # Define a flag to determine when we get the data we needed
 loop_flag = False

 while loop_flag == False :

  # If statement to decide how to hand the data it managed to read
  if len(heard_data) >= 13 :

   # There was enough data, process the angle
   # There was enough data to calculate angle of roll, process it
   data = heard_data[-13:-2]



   # Obtain and calculate Roll
   RollL = data[2]
   RollH = data[3]
   angle_roll = ((RollH<<8)|(RollL))/32768.0*180.0

   # Close the serial object
   ser.close()

   # Change the status of the loop flag just in case
   loop_flag = True

   # Normalize the angle to compensate for the radio telescope inclination
   angle_roll = angle_roll - normalization_factor;

   # Return the angle of roll
   return angle_roll

  else :
   # Iterate the timeout counter
   timeout_counter = timeout_counter + 1
```

```
   # Check if the getRoll function has timed out
   if timeout_counter > timeout_time :
    # At this point, there was a timeout, set the timeout flag to true
    timeout_flag = True

    print('Failed to communicate')

    # Possibly redundant
    loop_flag = True


    # Return a negative one so user can process the timeout error as a number (Easier to process)
    return -1

   else :
    # This means that there wasn't enough data, ensure that the flag is false to keep loop going
    loop_flag = False


# END OF FUNCTION DEFINITION

# BEGINING OF FUNCTION DEFINITION

# This function will keep running until it's over
def stability_test(TOLERABLE_DELTA, test_size) :

 # Stop the motor
 STOP_motor(board)

 # Wait short time and do nothing
 time.sleep(short_time)

 # Loop until the last test_size roll angles read are close in their values to prevent using crazy data

 # Flag to determine if the sensor values are stable
 tolerable = False

 # Do a while loop until the angle values are stable
 while tolerable == False :

  # Create an array that will hold the values that we're testing stability with and load it with 0s
  angle_test = [0]*test_size

  print(angle_test)

  # Define the iterating value for the next while loop
  iter_1 = 0

  # Begin the next while loop to load angle_test
  while iter_1 != test_size:

   # Activate the blue LED to indicate that we are obtaining angular data
   PIN_BLUE.write(1)
```

```
   # Obtain the angle of roll
   angle_temp = serial_getRoll()

   # Deactivate the blue LED to indicate no longer obtaining angular data
   PIN_BLUE.write(0)

   # Get the iter_1 position for our angle_test array (so we're loading the array with test_size roll ar
   angle_test[iter_1] = angle_temp

   # Diagnostic print
   print(angle_test)

   # Increate the iterator
   iter_1 = iter_1 + 1

  # Inner most while loop has ended, now we test for variation (stability of angle)

  # Define the iter_2 variable
  iter_2 = 1

  # Begin second while loop to determine stability of angle data
  while iter_2 != test_size:

   # Do an if statement to determine if the data is stable
   if abs(angle_test[0] - angle_test[iter_2]) <= TOLERABLE_DELTA :

    # Set the tolerable flag to true until told otherwise
    tolerable = True
    print('It looks good man')
    print(abs(angle_test[0] - angle_test[iter_2]))
    # The variation between them is good so iterate normally
    iter_2 = iter_2 + 1

   else :

    # If this occured, the variation was too large and hence unstable therefor it failed the check
    # Set tolerable flag to false
    tolerable = False
    # Kill the loop by setting the iter_2 to max value
    iter_2 = test_size
    print('We killed the program')

 print('The signal is stable!')

# END OF FUNCTION DEFINITION

# BEGINING OF FUNCTION DEFINITION

def mode_gta(desa):

 # Set initial time to 0
 time_elapsed = 0
```

```
true_angle = serial_getRoll()

print(true_angle)

# Obtain the current angle
current_angle = round(true_angle)

# Check if there was a timeout error
if current_angle == -1 :
 #print('TIMEOUT ERROR')
 return
else :
 # Proceed with rest of the code
 random = 0

# Print the angle and time in real time
print('Current Angle: ')
print(current_angle)
print('Time Elapsed')
print(time_elapsed)

# Create the dynamic array for angle
angle_array_np = np.array([true_angle])
# Create the dynamic array for elapsed time
time_array_np = np.array([time_elapsed])

# Load the angle on the dynamic array using np
# Load the elapsed time on the dynamic array using np

# Beginning counting time
start_time = time.time();

# Set a timer flag to determine when to start counting time in the loop
timerFlag = False

# Flags to indicate if the direction was flipped
forward_flag = True
reverse_flag = False

print('Desa while loop initiated')

# Begin the loop that will move the actuator
while current_angle != desa :

 # Determine when to start counting time in the loop
 if timerFlag == False :

  # Don't do anything, just set timerFlag to true
  timerFlag = True

 else :

  # This means that we have already gone through the loop, starting counting time now
  # Obtain the current angle
```

```
 true_angle = serial_getRoll()

 current_angle = round(true_angle)
 # Get the time
 time_elapsed = time_elapsed + (time.time() - start_time)
 # Start counting the time again
 start_time = time.time()

 # Display the vital information
 print('Current Angle: ')
 print(current_angle)
 print('Time Elapsed')
 print(time_elapsed);

 # Append the angle to the angle array
 angle_array_np = np.append(angle_array_np, true_angle);
 # Append the elapsed time to the time array
 time_array_np = np.append(time_array_np, time_elapsed);


# If statement to determine what should be done
if current_angle < desa :

 # Angle is too small, push the dish forward
 FWB_motor(board)
 # Delay a short amount of time
 #time.sleep(short_time)
 # Stop the motor when done to prevent over correction
 #STOP_motor(board)

elif current_angle > desa :

 # The angle is too large, we need to correct our position.  Do a very small reverse
 RVB_motor(board)
 # Delay a very short amount of time
 #time.sleep(very_short_time)
 # Stop the motor when done to prevent over correction
 #STOP_motor(board)

else :

 # This means that the current angle is the desired angle (desa) so indicate success with sound and er
 # Stop the motor
 STOP_motor(board)

 #sound_success(board)


 # Show the content of the angle and time
 print(angle_array_np)
 print(time_array_np)

 # Save both arrays to a .m file for plotting on MATLAB
 data = {
```

```
    'angle_array': angle_array_np,
    'time_array': time_array_np
   }
   savemat('arrays.mat', data)

   # Wait short time before shutting off red light indicator
   time.sleep(2)

   # Turn the red LED off
   PIN_RED.write(0);

# END OF FUNCTION DEFINITION

# Main code goes here

print('\n\n\n\nRotation system has fully loaded!\n')

angle_of_choice = input('System Message: What angle would you like to move the telescope to?\n')

print('\nSystem Message: Attempting to execute system rotation.')

time.sleep(1.5)

mode_gta(int(angle_of_choice))

if timeout_flag == True :

 print('\n\n\nSystem Message: The telescope timed out when trying to read angular data from it! \n\n\n')

else :

 print('\n\n\nSystem Message: The telescope is now at ', angle_of_choice, ' degrees!')
 light_pattern();
```

7.2 AngleTimePlotter.m

```
% Time vs Angle Plotter
% The purpose of this script is to evaluate how the rotation system
% is performing in terms of its ability to orient itself towards the target
% angle

% Variable to represent the target angle
target_angle = 15;

% Plot the angle of radio telescope against time
plot(time_array, angle_array, 'LineWidth', 2, 'Color', 'b');
title('Angle of Radio Telescope vs Time');
xlabel('Time (s)');
ylabel('Angle of Radio Telescope');
yline(target_angle + 2, 'Color', 'r', 'LineStyle','--', 'LineWidth', 2);
yline(target_angle, 'Color', 'r', 'LineWidth', 2);
```

```
yline(target_angle - 2, 'Color', 'r','LineStyle', '--', 'LineWidth', 2);
```

7.3 AngleComparison.m

```
% Angle Comparison Script
% The purpose of this scrip tis to manually compare the rounded reported
% angles by the angle sensor to the angles reported by the analog angle
% protractor
% By Nicholas Alva

% Clear everything
clc
close all
clear

% Create an array with the angles (in degrees) read from the analog
% protractor
analog_protractor = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90];

% Create an array with the angles (in degrees) read from the angle sensor
angle_sensor = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90];

% Plot the angle protractor's values against the angle sensor's value
plot(analog_protractor, analog_protractor, 'LineWidth', 2);
hold;
plot(analog_protractor, angle_sensor, 'LineWidth', 3, 'LineStyle', '--', 'Color', 'r');
xlabel('Angle Read by Analog Protractor (Degrees)');
ylabel('Angle Read by Angle Sensor (Degrees)');
title('Plot Comparing Angle Read By Sensor and Protractor');
legend('Analog Protractor Reading', 'Angle Sensor Reading');
```

Bibliography

[1] U.S. Department of Transportation, "What is the radio spectrum?." https://www.transportation.gov/pnt/what-radio-spectrum. Accessed: May 25, 2025.

[2] A. Klesman, "How do radio telescopes work?," 2019. Accessed: May 30, 2025.

[3] C. Cofield, "How scientists captured the first image of a black hole," 2019. Accessed: May 30, 2025.

[4] N. Arroyo, B. Benedicto, and T. Ikehara, "Cost efficient radio telescope." https://scholarcommons.scu.edu/elec_senior/68, 2022. Electrical and Computer Engineering Senior Theses, Paper 68. Accessed: May 25, 2025.

[5] L. Barnes, M. Quang, and A. Rouzmehr, "Santa clara radio astronomy project ii (scrap ii)." https://scholarcommons.scu.edu/elec_senior/82, 2023. Electrical and Computer Engineering Senior Theses, Paper 82. Accessed: May 25, 2025.

[6] T. DeCastro, J. Espinosa, and D. Stathakis, "Santa clara radio astronomy program (scrap) iii." https://drive.google.com/drive/u/0/folders/0ABICefMz8oWAUk9PVA, 2024. Accessed: May 25, 2025.

[7] "Earth globe atlantic." https://www.istockphoto.com/vector/earth-globe-atlantic-gm1390066370-447144372, 2023. Accessed: May 30, 2025.

[8] "National radio astronomy observatory (nrao)." https://public.nrao.edu/, 2025. Accessed: May 30, 2025.

[9] Federal Communications Commission, "Fcc online table of frequency allocations," 2022. Accessed: 2025-05-27.

[10] N. Alva and P. Lattimer, "Scrap-iv rotation documentation." https://docs.google.com/document/d/1rZzB5pyo6u4nXJpzMFtlKpwXZKmC_-XjdoRFxZAcgCE, 2024. Accessed: May 2025.

[11] M. Capello, "Reclaiming the night and the dark sky movement," 2022. Accessed: May 30, 2025.

[12] OpenCV, "Contour features — opencv documentation." https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html, 2024. Accessed: 2025-06-01.

[13] Raspberry Pi Foundation, "Raspberry pi 4 model b specifications," 2025. Accessed: May 26, 2025.

[14] The Arduino Team, "Arduino uno rev3," 2025. Accessed: May 26, 2025.

[15] VEVOR, "Vevor 2pcs 12v linear actuator kit, 30 inch, 0.35"/s, 220lbs, 1000n, ip54 protection." https://www.vevor.com/linear-actuators-c_11633/vevor-2pcs-12v-linear-actuator-kit-30-inch-0-35-s-220lbs-1000n-ip54-protection-p_010775883039, 2024. Accessed: May 2025.

[16] Handson Technology, "BTS7960 Motor Driver Datasheet." https://www.handsontec.com/dataspecs/module/BTS7960%20Motor%20Driver.pdf, n.d. Accessed: October 2024.

[17] HiLetgo, "Hiletgo bts7960 43a high power motor driver module/smart car driver module for arduino current limit," 2015. Accessed: 2025-06-01.

[18] WitMotion Shenzhen Co., Ltd, "Sindt ttl datasheet." https://drive.google.com/drive/u/1/folders/1l74VhDvlR9x-oLzpLNZch2VA6hCTx-2X. Accessed: Oct. 2024.

[19] WitMotion Shenzhen Co., Ltd, "Wit standard communication protocol." https://drive.google.com/file/d/1xrfK9bAEncgFQYjvT_c6vwSEH0ZhzaUZ/view. Accessed: May 25, 2025.

[20] K. lectures Dr. Stela, "Pitch, roll and yaw in orthodontics @kapdentlectures, 2022," 2022. Accessed: 2025-05-26.

[21] Santa Clara University School of Engineering, "Maker Lab." https://www.scu.edu/engineering/makerlab/. Accessed: May 31, 2025.

[22] The MathWorks, Inc., "MATLAB System Requirements," 2025. Accessed: May 26, 2025.