Fresnel Lens Control Tracking

Samuel Dixon
Jordan George

Concept of Operations

REVISION – Draft E
29 April 2023

Concept of Operations
for
Fresnel Lens Control Tracking

Team <66>

Approved by:

Jordan George	4/29/2023
Project Leader	Date
Dr. Kalafatis	4/29/2023
Prof. Kalafatis	Date
Dalton Cyr	4/29/2023
T/A	Date

Change Record

Rev.	Date	Originator	Approvals	Description
A	9/9/2022	Samuel Dixon, Jordan George		Draft Release
B	9/30/2022	Samuel Dixion, Jordan George		Updated System Description to accurately display new subsystem division. Updated block diagram to represent new systems. Updated Executive Statement.
C	10/13	Samuel Dixon		Removed highlighting, updated subsystems, and fixed grammar issues based on feedback.
D	11/16	Samuel Dixon		Updated project changes, repartitioning of subsystems
E	4/29/2023	Jordan George		Updated project name

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

1. Executive Summary

The goal of this research project is to help develop hardware and software for automatic and manual control of a fresnel lens for a small-scale solar thermal system consisting of a heliostat to Fresnel lens to target optical path. There is quite a bit of work Currently, manufacturers who design their products for very large CSP systems (concentrating solar power, i.e. power towers) are not suitable for this R&D project as it would be too cumbersome, costly, have proprietary issues, and an unforeseeable length in project timeline.

To demonstrate basic system performance in a timely manner, we will assemble a heliostat and demonstrate solar tracking, through a lens, to a focused target spot with motor control for two axis positioning of the heliostat into a fixed lens/target fixture and incorporate position feedback for closed loop operation. Having a stand-alone heliostat unit that can be easily setup, measured under sun, and returned to the lab for storage is desired. A small battery and possibly a PV panel and controller should be integrated onto the heliostat unit. Wireless control, sensor feedback, possibly a camera on the target area, data acquisition and data logging are desirable using a laptop (MS Windows) and Matlab. The solar input should be measured and an estimate of the solar energy delivered to the target area made (more on this later, we have to measure the direct normal irradiance - ie. DNI - separate from the global horizontal irradiance using a homemade rotating shadowband on an Apogee pyranometer).

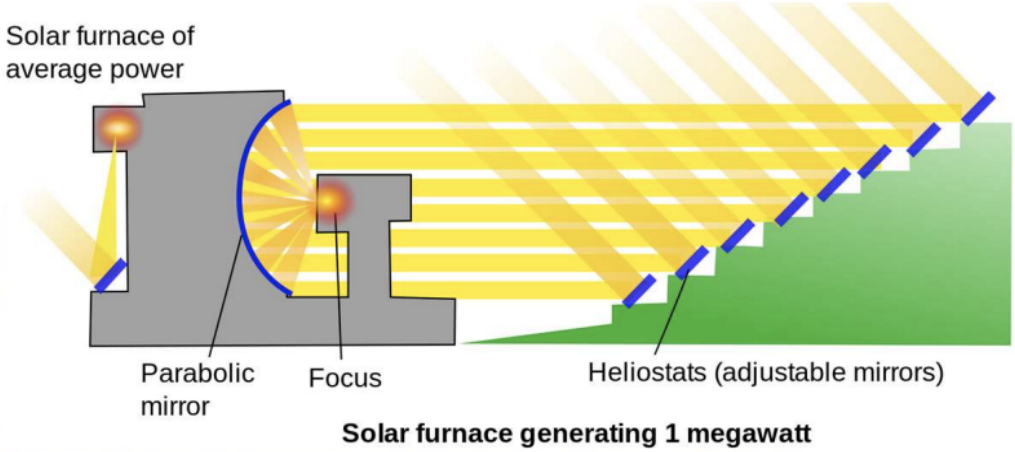


Figure 1: General concept of a solar furnace

2. Introduction

This paper provides an introduction to a Solar Tracking Mount for a solar furnace. This tracking mount will rotate and tilt the planar mirrors based on where the sun is located. The purpose of this would be to allow for the mirrors to concentrate more of the sun's energy onto the focal point of the solar furnace. A more efficient solar furnace would allow for the varying equipment that are powered by these devices to have a stronger source of power.

2.1. Background

The project will improve energy infrastructure and generation systems around the world by focusing light onto a target from the sun. Making use of the sun's available energy limits the dependence on non-renewable energies, which are currently being depleted at an unsustainable rate. Additionally, this sustains less environmental impact. This project seeks to improve the feedback control aspect for tracking the sun's position. This will be done by providing the control based on permutations of a multitude of parameters. Additionally, the method of connecting the mirrors and the motors to a provided frame for optimal optical alignment will be improved. This will be done through the use of hardware fasteners, actuators, and motors. This new configuration will provide optimal support for collecting the maximum amount of thermal energy on the target, while using minimal amounts of input power.

2.2. Overview

The Solar Tracking Mount will be started off by assembling the array of planar mirrors onto a single frame. The frame will have the ability to rotate and tilt the mirrors. Each mirror will have a motor to allow each mirror to have its own tilt. Also, there will be actuators that will be assisting the motors in the mission to tilt the mirrors in the optimal position. There will be sensors that will detect the optical output and input for each mirror. That data will be used to adjust the mirror until the maximum optical output is obtained. A sun tracking algorithm will be coded into the microprocessor to keep the mirrors in the most optimal position to output the most of the sun's energy. There will also be data visualization that the user of this tracking mount will be able to view. This will provide insight to the user of what exactly is happening with the mirrors and their optics information.

2.3. Referenced Documents and Standards

- https://www-sciencedirect-com.srv-proxy1.library.tamu.edu/science/article/pii/S0960148116311624
- http://www.powerfromthesun.net/Book/chapter08/chapter08.html
- https://www-sciencedirect-com.srv-proxy1.library.tamu.edu/science/article/pii/S0038092X00001560
- https://www.sunearthtools.com/dp/tools/pos_sun.php?lang=en#annual

3. Operating Concept

3.1. Scope

The Solar Tracking Mount will accelerate the transition to renewable energy by adaptively focusing the sun's light onto a target through the use of a rotating mirror array, commonly referred to as a heliostat. The target will be equipped with a sensor used to provide feedback into the processor of the motor drivers, actuating the adjustment of mirrors to an optimal point for maximized thermal output. The system will provide a data visualization device to communicate system-level information, such as sensors, current mode of operation, and log criteria observed for decision making.

3.2. Operational Description and Constraints

This project will be used in thermal energy conversion applications, particularly involving a thermal plate or furnace, in which heat is converted into another form of energy. Particularly, this system will serve as an input to a larger system, where electrical energy can be provided to consumers through renewable energy.

3.3. System Description

The Solar Mount Tracking will consist of controlling the motors that can rotate and tilt the mirrors, measuring the optical output and input of the mirrors, and an algorithm to allow the array of mirrors to track the sun and rotate accordingly. There are 6 subsystems for this project:

- PCB design
    - The PCB board will interface the tracking controls processor with the power supply, motors, data visualization tool, and sensors. The board is a place where the electrical components are organized and the signals are routed to their respective positions.The board also handles the power distribution and delivery to all the various components.
- Data visualization
    - This tool will provide valuable insight to verify different aspects of the system and communicate different metrics across the system to the user.
- Tracking Controls
    - The tracking controls are the processor and brain of the system. Through feedback received from the sensors, time of day, and other sources of valuable information, will decide how to change the positioning of the mirrors in the system.
- Optical Mirror Setup
    - The physical setup is how the motors, mirrors, actuators, and frame will be configured. By creating a controllable array of mirrors on a frame, the controller's job will be easier to effectively channel the light onto the target. Although

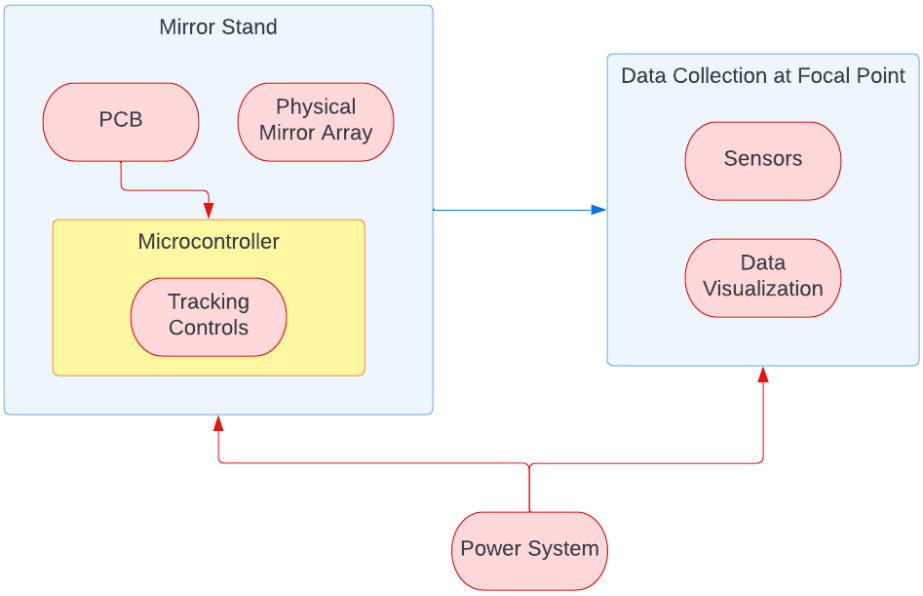


Figure 2: Subsystem block diagram

3.4. Modes of Operations

The system will have different modes of operations depending on the time, weather, and season. Specifically, the system will have four modes: a sleep mode (for times when it is impractical to have the system, such as night), full range operating (for times when the sun is fully available), partial range operating (for seasons, such as winter, or weather, such as cloudy for when the optimum amount of sunlight is not available), and lastly, the system will have a debug mode (where the system is expecting to be tested, changed, and manipulated).

3.5. Users

Researchers, utility providers, and customers are all potential users of this system. This system intends to be a basis for providing sustainable energy for use. By providing robust tracking of the movements in the sun, change in weather, and different seasons, the system gives researchers a head start in thermal energy conversion systems, where heat can be converted into a more useful form, such as electricity. This would provide utility providers a method of incorporating a renewable energy source to distribute power to different businesses.

3.6. Support

Support will be provided through a user manual. The document will describe the scope of the system, the purpose, and how to change operating modes and parameters in the program. Additionally, the manual will provide information on how to “power on” the system and use the Android application for data interpretation.

4. Scenario(s)

4.1. Campsite bathroom lighting

When camping there is commonly a public bathroom available for people to use. Sometimes these public bathrooms are transportable and have no electrical power, but other times they are established bathrooms with lighting. A potential use case of this application would be to replace the energy source of the bathrooms with lighting. Fortunately, when camping there is typical natural light during the day. Using this application to collect and store energy during the day, when natural light is available in the bathroom would let non-renewable energy be replaced in the evening, when lighting is needed.

4.2. Airport Charging Applications

When traveling there is a varying demand of people needing to charge appliances in airports. This variation is due to the nature of supply and demand. Increased travel typically occurs when people have an abundance of time, cultural holidays, and the price of tickets. In airports, there is a large amount of variation in power required to charge electrical appliances, such as phones or laptops. In a smaller airport near the tropical, sunny climate, these spikes of variation could be accounted for by this problem.

4.3. Thermal Energy System Research

There is substantial research being done in thermal energy conversion systems. This system would provide a valuable input into those systems, as the optimal supply would be provided to enable the largest throughput possible for usable energy at the output of their energy conversion system.

5. Analysis

5.1. Summary of Proposed Improvements

This system will provide improvements in tracking, by basing the feedback on a diverse suite of parameters. Additionally, the proposed system will provide data driven-insights based on the light collected on the target and the positioning of the motors. Lastly, the system will offer in-program functionality adjustments and calibration to accommodate different geographic locations and conditions.

5.2. Disadvantages and Limitations

The proposed system will have some apparent limitations. Quite notably, is the fluctuation in the ability to focus light during inclement weather, different seasons, and the time of day. Additionally, there may be substantial tradeoffs in the power needed to power the system and the amount of light focussed on the target.

5.3. Alternatives

An alternative solution may be to use solar panels rather than mirrors when tracking the light, although this alternative could potentially be more costly to implement for collecting the sun's energy. Additionally, maintenance for mirror damage is likely less expensive than the cost of damage incurred on a solar panel. Another alternative could be to control the thermal plate to collect the energy, rather than focus the light through mirrors. This method would reduce the complexity of having both a target and a focussing system, although without focussing the light, a reduction in luminous intensity would be observed. The tradeoff here is observed between complexity and energy collected. Additionally, there would be less spatial coverage, as heat dissipates with a larger surface area.

5.4. Impact

The project impacts the way society uses energy. Currently, the majority of the world's energy consumption is derived from fossil fuels. Recent trends and investments show that the world is on track to change the majority of reliance to more sustainable alternatives. Through applications like this the world can accelerate the push to sustain the shift to a different form of energy. The implications of this shift would be an increase in the availability of fossil fuel supply, an alleviation of environmental stress and pollution ,a transformation of job markets in certain industries, and a shift in the marketplace value of certain commodities and businesses.

Fresnel Lens Control Tracking

Samuel Dixon
Jordan George

Functional System Requirements

REVISION – Draft E
3/20/2023

Functional System Requirements
FOR
Fresnel Lens Control Tracking

Prepared by: Jordan, Samuel

Jordan, Samuel 4/29/2023

Author Date

Approved by:

Jordan George 3/20/2023

Project Leader Date

Dr. Kalafatis 3/20/2023

Dr. Kalafatis. Date

Dalton Cyr 3/20/2023

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
A	9/30/2022	Samuel Dixon, Jordan George		Draft Release
B	10/13/2022	Samuel Dixon		Removed highlighting and made revisions based on draft feedback.
C	11/16/2022	Samuel Dixon		Updated system requirements after changes done to subsystem organization from midterm presentation feedback.
D	2/28/2023	Samuel Dixon		Updated system requirements to reflect changes in subsystem partitions and transitioning from phone application to web application
E	3/20/2023	Samuel Dixon		Updated subsystem requirements to reflect changes in system and reflect tests being performed in the validation plan for success of desired functionality

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Purpose and Scope

The world is rapidly transitioning from non-renewable to renewable energy production on a basis of conscious consumption about environmental impact. Solar Energy is being developed and has been improving Most Fresnel Lens products are made for large solar power systems. For this R&D project, a much smaller-scale Fresnel Lens system is desired. This is because larger scale heliostats are costly, burdensome, and have timelines that are unpredictable. The purpose of this research project is to create a stand-alone Fresnel Lens unit that can be set up easily, output data, and can be stored in the lab whenever desired. This solar tracking mount will consist of a mirror array that will be tilted and rotated based on where the sun is. A sun tracking algorithm will be coded onto an Arduino board that will move the mirror array accordingly with the help of motors. There will be a web application that will visualize the data such as the input and output optical rays as well as the rotation and tilt angle of the array.

1.2. Responsibility and Change Authority

Subsystem	Responsibility
PCB design & Power Delivery	Jordan George
Data Visualization / User Interface	Samuel Dixon
Tracking Algorithm	Jordan George
Physical Mirror Array and Sensors	Samuel Dixon

Table 1: Subsystem partitioning

2. Applicable and Reference Documents

2.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
978-1-491-92176-0	First Edition - 5/5/2016	Arduino A Technical Reference
IEEE 802.11-2016	12/14/2016	Standard for WIFI Networks

Table 2: Applicable Documents

2.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
1	01/03/2017	Run-time detection and correction of Fresnel Lens tracking errors
2	10/2003	Concentrator Optics
3	9/15/2001	Computing the solar vector

Table 3: Reference Documents

2.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

3. Requirements

3.1. System Definition

This project entails refining the tracking controls, structural arrangement, hardware organization, and data visualization of a small-scale Fresnel Lens control system. These refinements include tracing the trajectory of the sun at a given longitude and latitude, while receiving feedback from sensors, incorporating planar mirrors with increased stability, routing signals on a PCB shield, and utilizing internet and bluetooth protocols to visualize system metrics in real-time and on a historical basis.

This proposed solution contains the following subsytems: PCB Design and Power Delivery, Data Visualization, Tracking Algorithm, and Physical Mirror Array. PCB Design and Power involves the physical hardware and the routing of electrical signals, while interfacing the capabilities of the Arduino. Data Visualization involves receiving data from sensors and displaying the data in real time, while also storing the historical data into a remote server for future analysis. The Tracking Algorithm is the C programming within the Espressif systems IDF to orchestrate the signals to actuate the motor control and adjustment of the mirrors as a function of the sun's position.

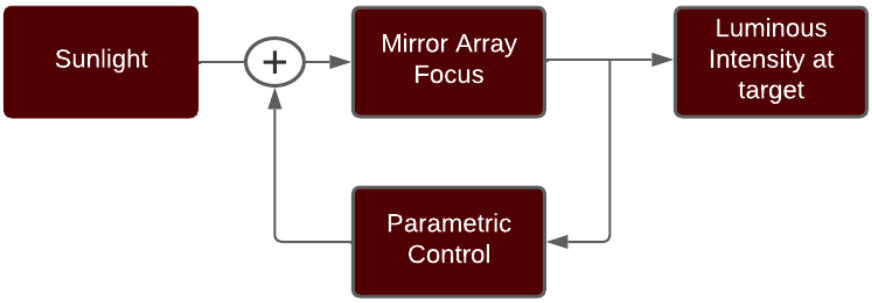


Figure 2. Block Diagram of System

The block diagram in Figure 2 visualizes the control aspects of this project. The input to the system is the position of the sun at a given time. In the forward path, the mirror array is focused based on the time and longitude data. The system then adjusts the position of the mirror array though other aspects of parametric control, such as data received from feedback in the program. This reduces the error signal between the optimal maximum luminous intensity and the observed maximum intensity at a given time.

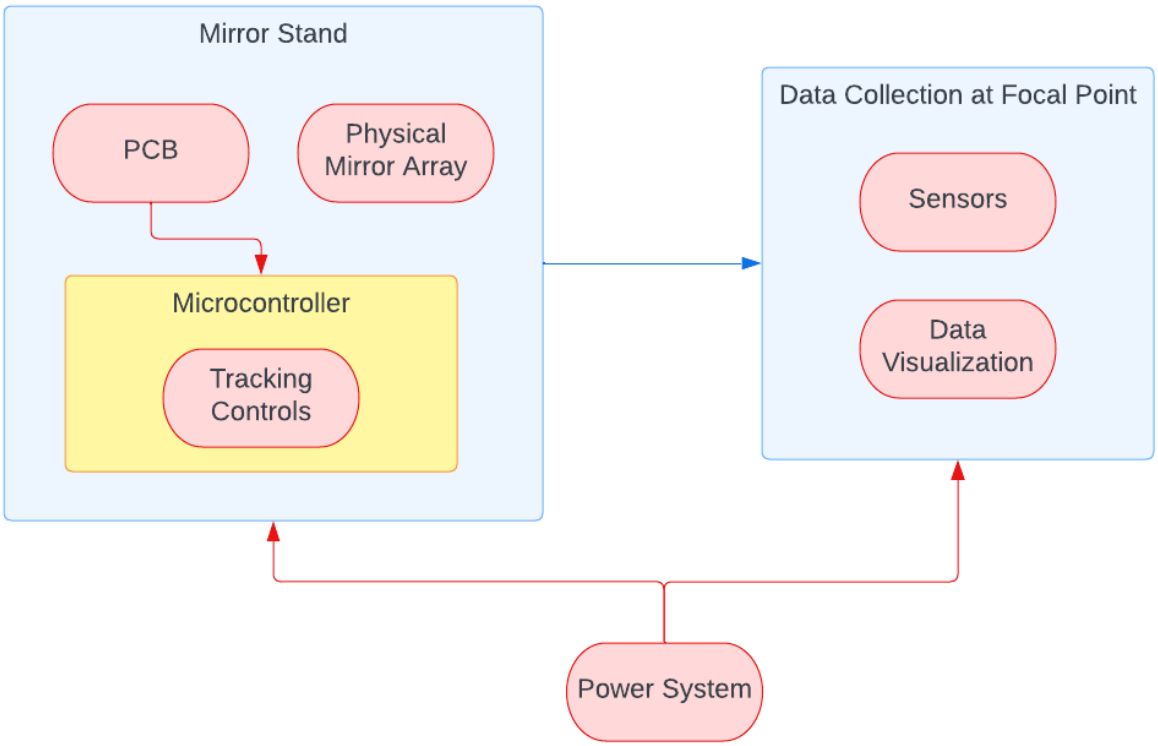


Figure 3. System Relation Diagram

The relational diagram in Figure 3 provides the relationship between the subsystems of the project. Items contained in the respective blue boxes are groups that correspond to each other symbolically and physically. On the left hand side, the PCB shield interfaces the capabilities of the Arduino microcontroller, which serves as the brain for the tracking control of the mirrors. The mirror array and hardware are physically connected together routing light to the right hand of the diagram, the target. At the target, there will be a photodiode with an optic filter for providing feedback to the microcontroller. The target also provides the source of the data for the visualization portion of the code.

3.2. Characteristics

3.2.1. Functional / Performance Requirements

3.2.1.1. Tracking the Sun

The Fresnel Lens tracking system will adjust motors every ten minutes and record ten data points from the accelerometer and magnetometer sensors.

Rationale: This is the core system performance requirement. Operating conditions occur in Texas A&M Research Park (30.6280, -96.3344).

3.2.1.2. Data Collection and Processing

The Fresnel Lens tracking system will process the data acquired via I2C from the accelerometer and magnetometer sensors. The data will be acquired serially with the I2C sensors. Upon receiving the measurements the EP32 Wroom microcontroller will use the IDF wifi module to send information in packets through wireless communication to a MySQL web server hosted on AWS. The data will be then sent to a front end website hosted on one of Google's cloud computing platforms, firebase. The data will be sent through amazon's microservices: lambda and api gateway which serve as the serverless backend for querying the data that will be displayed using javascripts, cascading style sheets (CSS), and HTML. The graphing package will be used from the Google Visualization API.

3.2.2. Physical Characteristics

3.2.2.1. Volume Envelope

The volume envelope of the Fresnel LensTracking Control System shall be less than or equal to 60 inches in height, 48 inches in width, and 48 inches in length.

3.2.2.2. Mounting

There will be a Fresnel Lens where the PCB and sensors will be mounted to the frame. They will be mounted through a CAD modeled 3d printer, electrical tape, screws, and other tools for mounting.

3.2.3. Electrical Characteristics

3.2.3.1. Inputs

The inputs of the Fresnel LensTracking system include the light given from the sun as well as data from I2C sensors, that includes the elevation and azimuth of the sun throughout the day depending on the latitude of College Station and the date of the year. This data will determine the tilt and rotation of the mirror array.

Rationale: This system is meant to output the optimal amount of solar energy so it is required that the frame is angled to best serve this purpose.

3.2.3.2. Power Consumption

The maximum peak power of the system shall not exceed 72 watts.

Rationale: This is a requirement specified by our customer due to the value of the pre-existing battery (12 V, 6A rated) within the system.

3.2.3.3. Input Voltage Level

The main battery of the Fresnel LensTracking System can supply up to 12 Volts.

3.2.3.4 Outputs

The output of the Fresnel Lens Tracking System is the positioning of the mirrors, which is realized through motor control and voltage inputs. The output for the sensor pins on the microcontroller should be 3.3 V and the battery terminal supply voltage should be 12 V.

3.2.3.5 Data Output

The Fresnel Lens System will contain a web application that will have a view where the customer can obtain information about the readings of the sensors. This will be crucial in ensuring the system is operating as intended and to validate that the product offered is within the operating guidelines and functional requirements specified above.The Fresnel Lens Tracking Control System will contain a web application that will have a GUI view where the customer can externally debug and test certain functionalities, rather than adjustments being made by the program. This external interface will allow the customer to rotate the two motors to move the mirror array translationally and rotationally. Additionally, the application will provide verification of connectivity and option to halt system execution.

Rationale: Provides the ability to debug the system and manually control the mirrors for rotating the mirror around two different axis points.

3.2.4. Environmental Requirements

The Fresnel Lens Tracking System shall be designed to withstand and operate in the environments and laboratory tests except for days with heavy rain, days that are very cloudy, and throughout the night.

Rationale: This is because the system is meant to track the sun so when the sun is not in the sky or is blocked heavily, then the Fresnel Lens would not be optimal in these conditions.

3.2.5. Failure Propagation

The Fresnel Lens Tracking System will not allow propagation of error to occur in the system beyond the visual interface. For software, when the sensor consistently is off with respect to the ideal value (determined

by experimental testing), the system interface will alert that there are either non-ideal environmental conditions occurring or a physical bug in the system. In hardware, there will be an overcurrent detection device to ensure that the pins of the esp-32 are not being strained past their operating conditions.

4. Support Requirements

Provided support for this project include an activity page within the android application, involving purpose statement, navigation, and change date. Additionally, there will be a feedback form within the application, where users can query for support to the service team. The service team will be able to consult subsystem owners for expertise regarding specific questions and provide valuable feedback in a timely manner to the customers as needed. The requirement to access this technical support would be a device that utilizes google's android operating system, can download applications from the play store, and has the capability of connecting to the internet.

Appendix A: Acronyms and Abbreviations

BIT	Built-In Test
CCA	Circuit Card Assembly
DNI	Direct Normal Irradiance
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
EO/IR	Electro-optical Infrared
FOR	Field of Regard
FOV	Field of View
GPS	Global Positioning System
GUI	Graphical User Interface
Hz	Hertz
ICD	Interface Control Document
kHz	Kilohertz (1,000 Hz)
LCD	Liquid Crystal Display
LED	Light-emitting Diode
mA	Milliamp
MHz	Megahertz (1,000,000 Hz)
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
mW	Milliwatt
PCB	Printed Circuit Board
RMS	Root Mean Square
TBD	To Be Determined
USB	Universal Serial Bus

Appendix B: Definition of Terms

IDF	Internet of things development framework

Appendix C: Interface Control Documents

The Interface Control Document is attached below

Fresnel Lens Control Tracking

Samuel Dixon
Jordan George

INTERFACE CONTROL DOCUMENT

REVISION – Draft C
29 April 2023

Interface Control Document
for
Fresnel Lens Control Tracking

Prepared by: Jordan, Samuel

Jordan, Samuel 4/29/2023

Author Date

Approved by:

Jordan George 4/29/2023

Project Leader Date

Dr. Kalafatis 4/29/2023

Dr. Kalafatis Date

Dalton Cyr 4/29/2023

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
A	9/30/2022	Erica Quist, Samuel Dixon, Jordan George		Draft Release
B	10/13/2022	Samuel Dixon		Removed highlighting and made revisions based on draft feedback
C	4/29/2023	Jordan George		Updated project name

Table of Contents

List of Tables

List of Figures

No table of figures entries found.

1. Overview

The Interface and Control Document (ICD) will explore in-depth detail regarding the subsystem information described in the Concept of Operations (ConOps) and Functional System Requirements (FSR) respective documents. This will involve detailing the specific descriptions of the hardware and software being used in this project and how they interface with each other. Lastly, the ICD will give information regarding the characteristics of protocols being used for design.

2. References and Definitions

2.1. References

IEEE Standard 802.11
Standard for WIFI Networks

Android Studio API Online Reference

The other Reference Documents are located in Section 2.2 of the Functional System Requirements document.

2.2. Definitions

API	Application Programmable Interface
DNI	Direct Normal Irradiance
g	Grams
GUI	Graphical User Interface
in	Inches
lbs	Pounds
mA	Milliamp
mW	Milliwatt
V	Voltage
Baud Rate	

3. Physical Interface

3.1. Weight

3.1.1. Weight of PCB Shield and ESP-32 System

Component	Weight
ESP-32 PCB System	13 g
L298n Motor Driver	26 g

Table 1: Weight of PCB Shield and ESP-32

3.1.2. Weight of Battery

Component	Weight
Battery	4.6 lbs

Table 2: Weight of the battery

3.2. Dimensions

3.2.1. Dimension of PCB Shield and ESP-32 System

Component	Dimensions
ESP-32 System	4.00 in x 2.10 in
L298n Motor Driver	1.69 in x 1.69 in x 1.02 in

Table 3: Dimensions of PCB Shield and ESP-32

3.2.2. Dimension of Frame

Component	Length	Width	Height
Frame	48 in	48 in	60 in

Table 4: Dimensions of the frame

3.2.3. Dimension of Mirror

Component	Length	Width	Height
Planar Mirror	12 in	12 in	0.16 in

Table 5: Dimensions of the planar mirrors

3.2.4. Dimension of Battery

Component	Length	Width	Height
Battery	5.94 in	2.56 in	3.70 in

Table 6: Dimensions of the battery

3.3. Mounting Locations

3.3.1. Mounting of Sensors

There will be two magnetometers mounted on the system frame. Also, a photodiode will be mounted to a target that will be used to measure how much of the sun's energy the Heliostat Control Tracking system outputs.

3.3.2. Mounting of Mirrors

The planar mirrors will be mounted onto the frame in a 3x3 array since there will be nine mirrors in this system. Support rails, epoxy, and fasteners will be used to complete this mounting task. The frame will have rotational and tilt capabilities to allow the mirror array to be angled for optimal output of solar energy.

3.3.3. Mounting of ESP-32 Control Unit

The ESP-32 Control Unit will be within a metal box that is mounted on one of the sides of the frame. Most of the circuitry, circuit boards, and wires will be located in this box.

4. Electrical Interface

4.1. Primary Input Power

Component	Max Voltage	Max Current
Battery	12 V	7 A

Table 7: Max Voltage and Current Values for the battery

4.2. Max Voltage and Current Values

Component	Max Voltage	Max Current
ESP-32 Wroom	3.6 V	260 mA
Magnetometer Sensor	12 V	40 mA
L298n Motor Driver	35 V	2 A

Table 8: Max Voltage and Current Values for PCB Shield and ESP-32

4.3. Signal Interfaces

4.3.1. Rotation & Tilt Sensors

A triple axis magnetometer will be used to get the rotation and tilt of the mirror array. This magnetometer will be connected to the ESP-32 mega which will communicate with the Android app to display this data.

4.3.2. Luminosity Sensor

To measure the output of the system, a photodiode will be attached to a target. This target will be set up to receive the outputted solar energy from the planar mirrors. The ESP-32 also will be responsible for relaying the data from this sensor to the Android app to visualize the data. This data will be useful to find the most optimal angle of the mirror array for the highest output.

4.4. User Control Interface

The user control interface is an web application that will provide the user with data visualization and system analytics through bluetooth communication of the ESP-32 and wireless connection to the internet. The bluetooth communication will service instantaneous data received from the ESP-32 and the wireless communication will service historical data via a MySQL WebServer database hosted on AWS accumulated over time.

5. Communications / Device Interface Protocols

5.1. Wireless Communications (WiFi)

In order for the android device to store the sensor data received through bluetooth communication from the microcontroller, it must connect to the internet and upload the data to the particular web server. The device will adhere to the Wireless Communication Protocol as specified in the IEEE Standard 802.11.

5.2. Device Peripheral Interface

The respective TX and RX pins of the microcontroller will be sending and receiving serial data to and from the smartphone though the ESP-32 rover microcontroller. This data received on the pins will be adhering to serial communication protocol, specifically UART. The data being shared between the two devices is asynchronous, meaning the devices do not have their own clocks defining when information is sent or received. Therefore, it is important that these devices adhere to the protocol specifications and agree on a common data sharing rate, which is commonly known as the baud rate.

Fresnel Lens Control Tracking

Samuel Dixon
Jordan George

SCHEDULE

	Oct. 5th	Oct. 19th	Nov. 2nd	Nov. 16th
PCB Design	Get Altium Schematic Drawn Up	Get Altium layout completed and pass all DRC checks. Order PCB and components	Assemble the voltage regulator part of the PCB and test varying input voltages.	Final soldering, testing, and debugging of the PCB.
Data Visualization	Investigate and begin development android packages for visuals	Save and store data in arduino and map out front end app	Program manual control functionality with firebase database	Final testing of application and additional tweaks
Tracking Control	Understand ESP32 configuration, research sun tracking equations	Align motors according to given input, locate the sun in the sky, model the appropriate angle the mirror should point	Move the motors in accordance with the sun while taking in sensor data	Setup a “daily” script
Mirror Array	Order mechanical parts	Put together parabolic array	Attach to overall frame	Test for vibrations

Fresnel Lens Control Tracking

Samuel Dixon
Jordan George

VALIDATION PLAN

Status Indicators	
Completed	
On Schedule/ In Progress	
Behind Schedule	

Task	Deadline	Current Status	
PCB Subsystem		Completed	
Get an understanding of the desired functionality of the PCB	Sept. 28	Completed	
Draw up schematic in Altium	Oct. 5	Completed	
Order components	Oct. 5	Completed	
Design PCB layout in Altium and pass all design rule check tests	Oct. 19	Completed	
Create gerber files and NC drill files to order PCB	Oct. 19	Completed	
Solder on voltage regulator part of the circuit for test for a constant 3.3V output based on a range of input voltages	Nov. 2	Completed	
Assemble the rest of components on the board for final testing and debugging	Nov. 16	Completed	
User Interface / Data Visualization Subsystem		Completed	
Download Android Studio and investigate examples for programming in java	Oct. 5	Completed	
Front End of Android Application 2/4 front end activity pages complete	Oct. 12	Completed	
Front End of Android Application all front end activity pages complete	Oct. 19	Completed	

Prototype Wifi communication on Application for data reading and Investigate Web Server for storage, and research Optometrika	Oct. 27	Completed	
Integrate firebase database through wifi on device	Nov. 3	Completed	
Debug application, test on phone, and programmatically check for wifi connection	Nov. 11	Completed	
Integrate firebase database through wifi and api key on device	Nov. 18	Completed	
Test reading, writing, and simple motor control utilizing database flag section as ISR routine	Nov. 24	Complete	
Tracking Controls Software and Sensor Interface	(Project Member Discontinuation)	(Project Member Discontinuation	

Performance on Execution Plan:

The performance on the execution plan is as follows. The PCB system design was drawn up, soldered, and tested. The Data Visualization system design was completed, constructed, and tested. The Tracking Controls Software and Sensor Interface was discontinued, as the last teammate was unable to continue with this project. The other two subsystems were followed through execution according to the schedule given in the format above. However, for the data visualization subsystem some advice was given to revise the current organization to an alternative database.

Performance on Validation Plan

The performance on the validation plan is indicated by the status markers in the table above. The tasks involving the two subsystems, PCB and data visualization, were able to be validated according to the plan above. After demoing, a few more tests needed to be run for the PCB subsystem. These tests included testing the noise levels for the voltages as well as testing the voltage with a varying load. Also, the data visualization sub system needs some modification in terms of the database which will be investigated during the break. Additionally, the last subsystem, tracking controls and sensor interface, was not able to be validated by the end of the system. Therefore, there will be more work to be done before integration in the next portion of the project scope.

Update

The data visualization and user interface subsystem was changed from a phone to web application. This was validated in-depth in subsystem results portion. The main motivation for the change was lack of organization in the phone application, which utilized a NoSQL database. The considerations for the design of the web application are also seen in the System Result document.

Fresnel Lens Control Tracking

Samuel Dixon
Jordan George

Subsystem Report

REVISION – Draft D
April 29, 2023

Subsystem Report
FOR
Fresnel Lens Control Tracking

Team <66>

Approved by:

Jordan George	4/29/2023
Project Leader	Date
Dr. Kalafatis	4/29/2023
Prof. Kalafatis	Date
Dalton Cyr	4/29/2023
T/A	Date

Change Record

Rev.	Date	Originator	Approvals	Description
A	12/2/2022	Samuel Dixon, Jordan George		Draft Release
B	1/24/2023	Samuel Dixon		Updating Subsystem Changes
C	2/28/2023	Samuel Dixon		Updating Substem Changes
D	4/29/20239	Jordan George		Updated Project Name

Table of Contents

Table of Contents

List of Tables

List of Figures

1. Introduction

This project intends to develop the control methodology and design behind an energy application device known as a Fresnel Lens in a broader energy system that converts thermal (renewable) energy to accessible energy in a chemical process. This project focusses on the tracking of the suns movements and focussing The device seeks to track the rotating motion of the sun throughout the day and rotate some mirrors to focus light onto a thermal plate for further energy conversion. The subsystems in this project entailed design and fabrication of a PCB, development of a user interface and data visualization, and tracking controls software for the microcontroller.

2. PCB Subsystem Report

2.1. Subsystem Introduction

The PCB subsystem intends to service the power to the microcontroller as well as provide a common interface for uploading code to the ESP32 Wroom through a micro USB connector as well as being able to access the GPIOs of the ESP32 Wroom for the various sensors of this project. The PCB will be powered by a 12V / 7A battery. There will be a voltage regulator part of this circuit to step down the 12V to 3.3V as the microcontroller and USB-UART bridge require around 3.3V to operate.

2.2. Subsystem Details

The PCB subsystem was created with the help of the PCB design software called Altium. Below is the schematic drawing of the PCB:

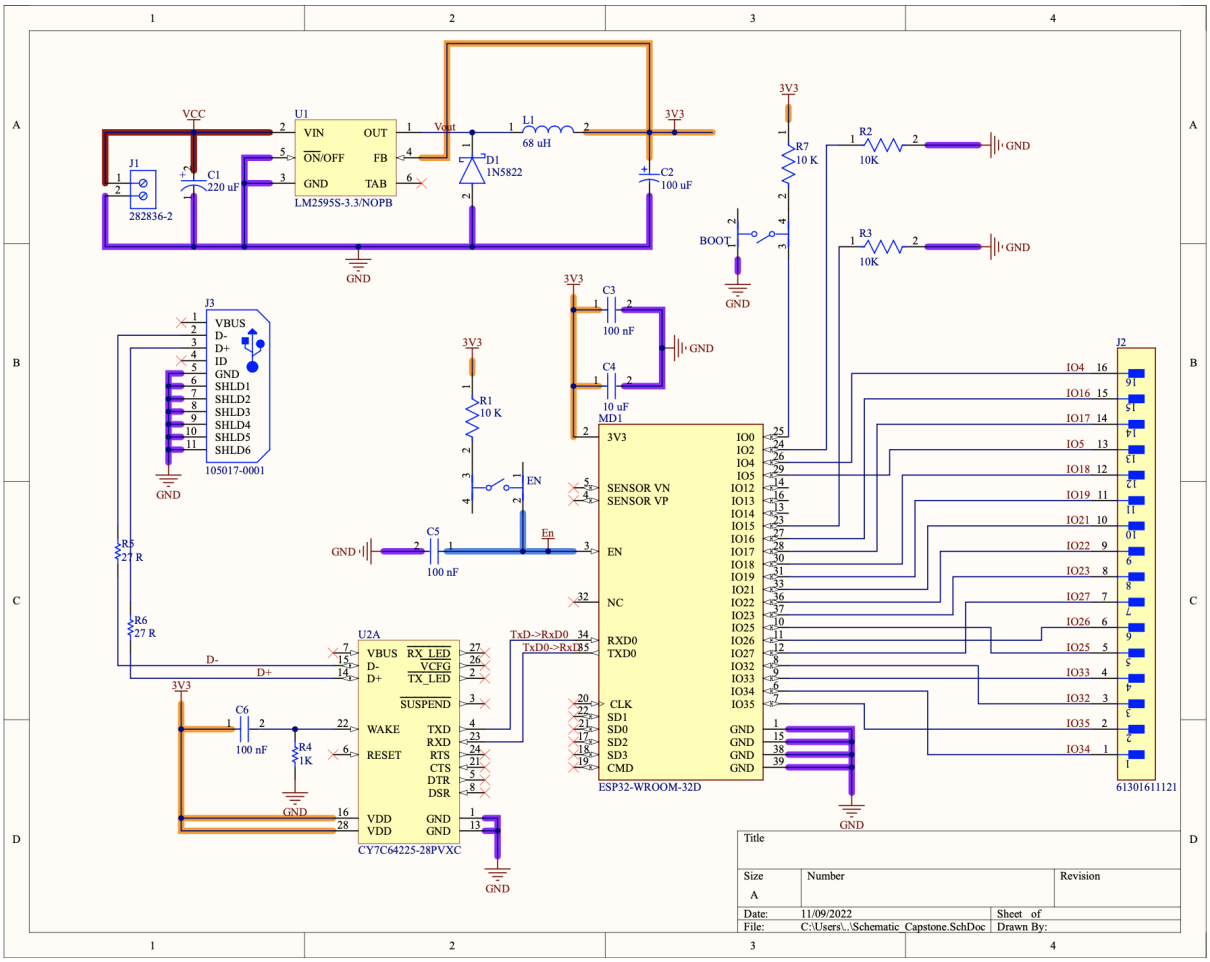


Figure 1: PCB Schematic

In the top left corner of the schematic is the voltage regulator that will step down the input voltage of 12V to 3.3V for the rest of this schematic. The input voltage of 12V is color coded by the dark red lines. The orange colored lines represent the 3.3V node while the purple colored lines represent the common ground node. The datasheet of the ESP32, voltage regulator, and the USB-UART bridge were read through to help best set up this schematic.

On the right side of the schematic are the 16 header pins that will be used for the multiple GPIOs of the ESP32. These will be hooked up to the different sensors in this project. In order to turn the ESP32 into bootloader mode, the BOOT button must be held down and shortly after, the EN button will be pressed. After, both buttons may be released. The next part of design process was the PCB layout pictured below:

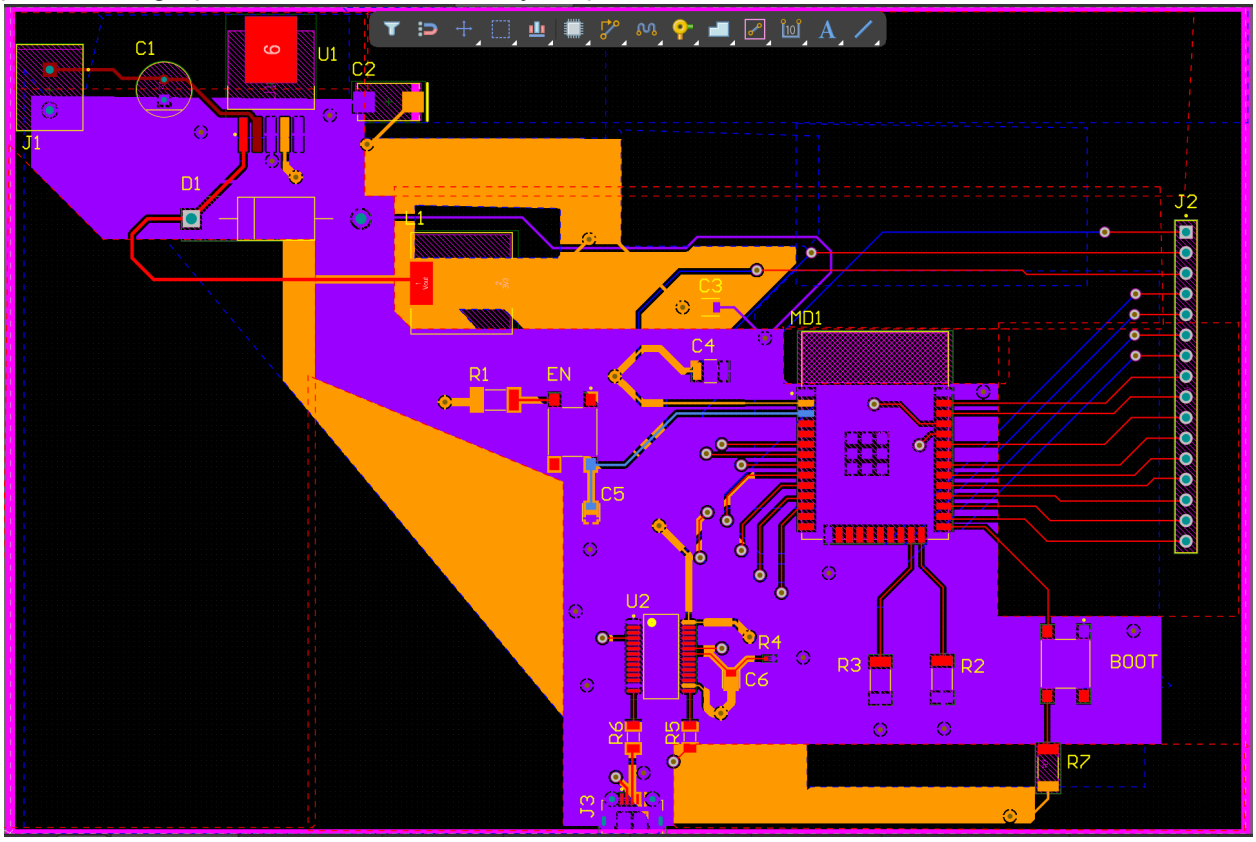


Figure 2: PCB layout

This layout starts with the screw terminals named J1 in the top left of this picture. This is where the input power will be plugged into. The board was able to be designed as a 2-layer board. There are two polygon pours to create a common node for needed voltages. The orange polygon pour is on the bottom layer and is the 3.3V net, while the purple polygon pour is on the top layer and is the common ground net. This Altium design was sent to JLCPCB for manufacturing. Once the board got here, all the components were soldered where they were designed to be, and it resulted into the following.

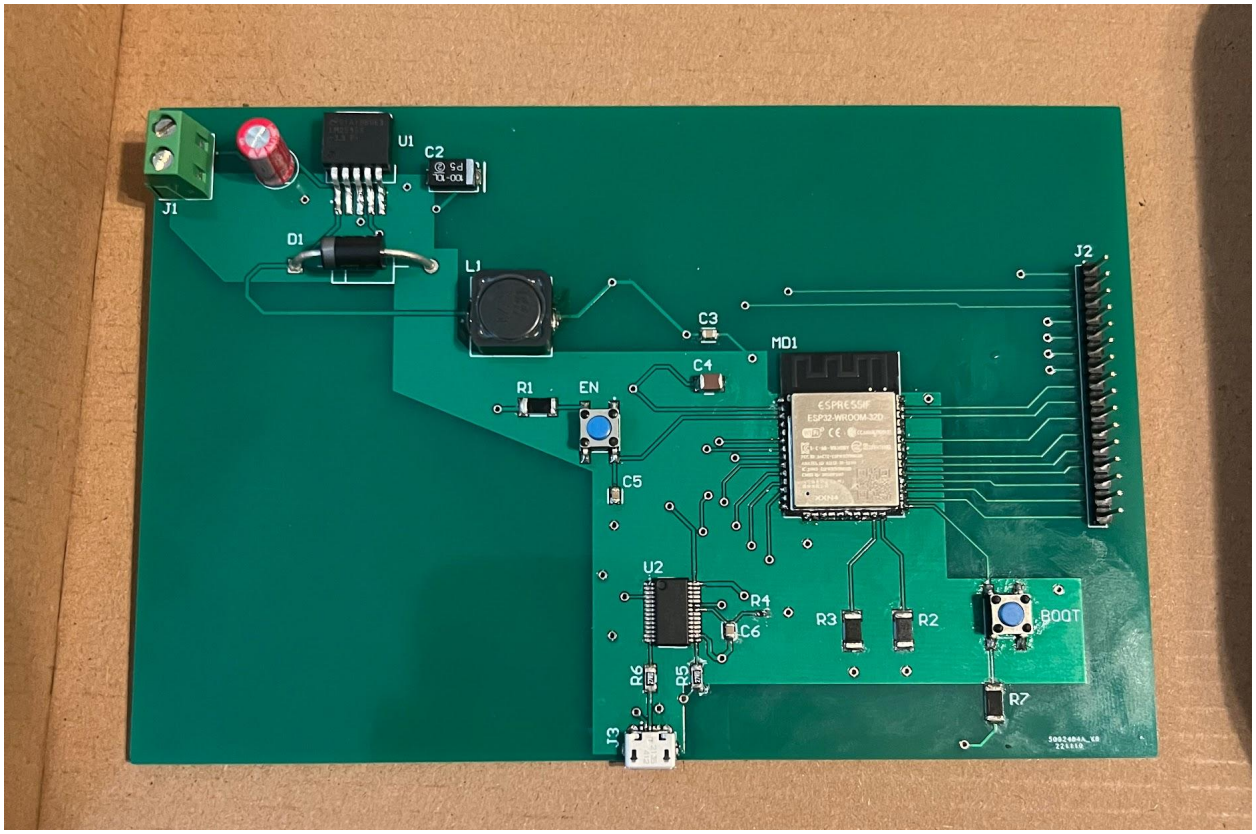


Figure 3: PCB with components soldered on

2.3. Subsystem Validation

The voltage regulator chip went under a lot of testing to see if it could hold a constant output of 3.3V. The first thing that was tested was varying the value of the input voltage.

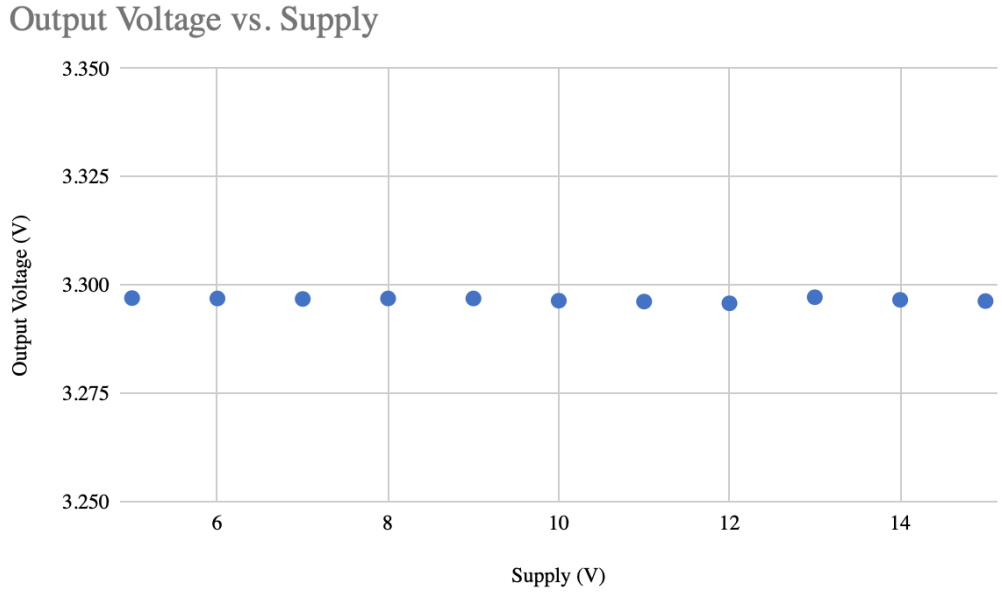


Figure 4: Output Voltage vs Supply of Voltage Regulator

As shown above, the output constantly holds slightly under 3.3. The nominal voltage of this experiment is 12V so this is working as desired, since the ESP32 and the UART bridge both operate at 3.3V. The next thing that was tested was by varying the load of the voltage regulator to see if the output still holds at 3.3V. The input voltage for these tests was held at a constant 12V.

Varying Load Tests			
Supply (V)	**Regulator Output (V)**	**Load being tested (Ohms)**	**Current At the Load (A)**
12	3.2958	Open circuit	0.006
12	3.263	15.83980583	0.206
12	3.227	7.665083135	0.421
12	3.193	5.295190713	0.603
12	3.155	3.933915212	0.802
12	3.122	3.144008056	0.993

Table 1: Voltage Regulator with varying loads

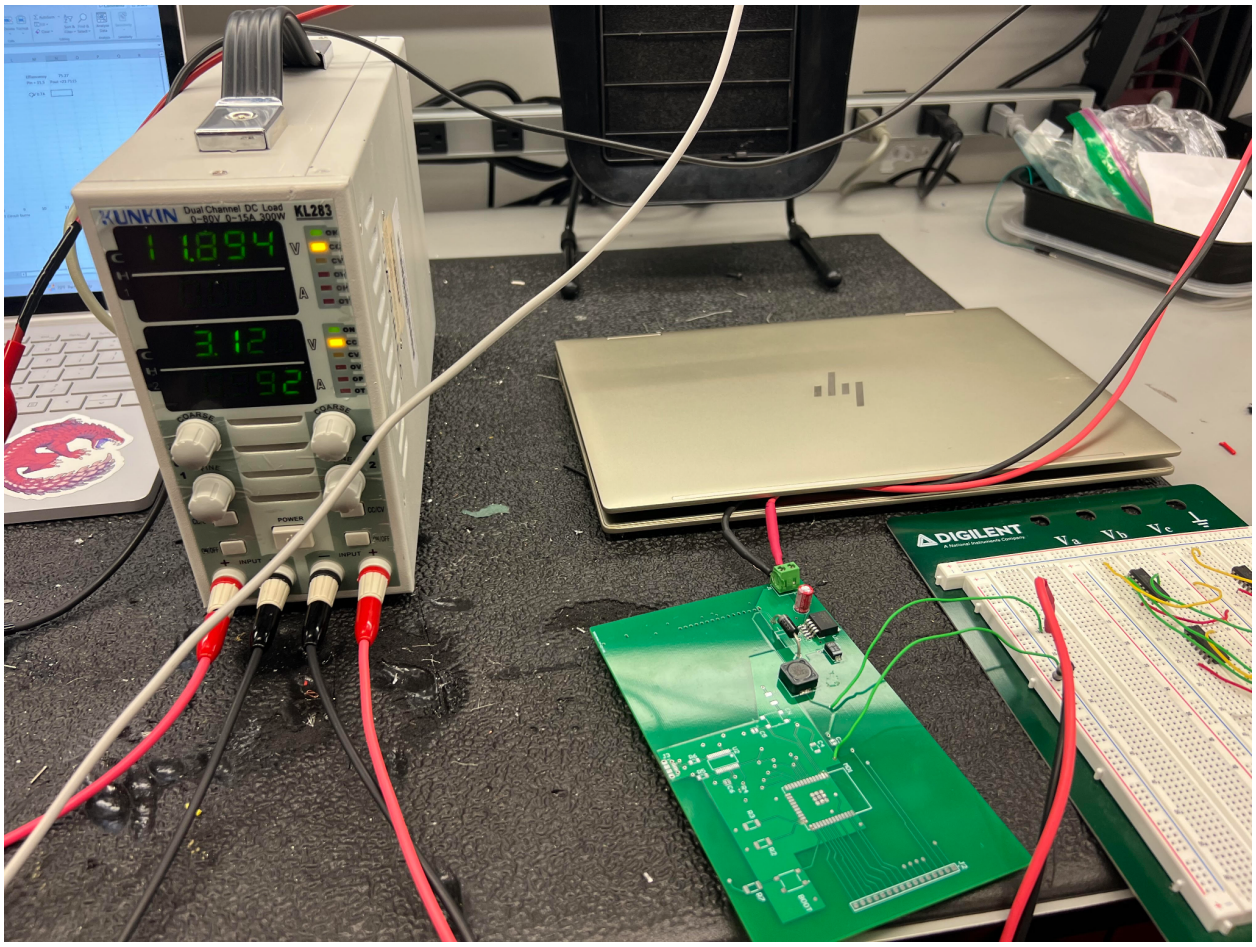


Figure 5: Voltage Regulator with varying loads

The max output current of the voltage regulator is 1 Amp so various data points were taken as the current drawn approached 1A. From Table 1, it can be seen that the voltage does

start to decrease slightly as the load decreases. The next set of data that was recorded was the efficiency of the voltage regulator. This was done by using one of the data points from Table 1, specifically the third row of the table. The input current was recorded by seeing how much current was being drawn from the power supply which was about 0.152 Amps.

Power In (W)	Power Out (W)	Efficiency
1.824	1.358567	0.7448283991

Table 2: Efficiency of the Voltage Regulator

The next set of testing was done on the circuit board that had all the components soldered on (Figure 3). First, it was tested to see if the input voltage would be stepped down to provide 3.3V to the ESP32. When the ESP32 was not in bootloader mode, almost no current was being drawn from the power supply. So the data in the following table was recorded while the ESP32 was on.

ESP32 when turned on		
Supply (V)	**ESP32 Voltage (V)**	**Current Drawn From Supply (A)**
5	3.2889	0.103
6	3.2893	0.086
7	3.29	0.08
8	3.3018	0.066
9	3.303	0.059
10	3.3022	0.057
11	3.2977	0.049
12	3.2978	0.045
13	3.2975	0.042
14	3.2968	0.04
15	3.2967	0.038

Table 3: Voltage at ESP32 with a varying input

This table shows that the circuit was working as expected. The output voltage would hold at a constant 3.3V which means the orange polygon pour from Figure 2 was at this voltage. This showed that the ESP32 and the USB-UART bridge were both being powered at 3.3V which is within their operating range. The last set of data recorded was taken by using the oscilloscope to measure the input and output voltage to check for any noise. Below are the pictures of what the oscilloscope showed for the input voltage of 12V and the output voltage of 3.3V.

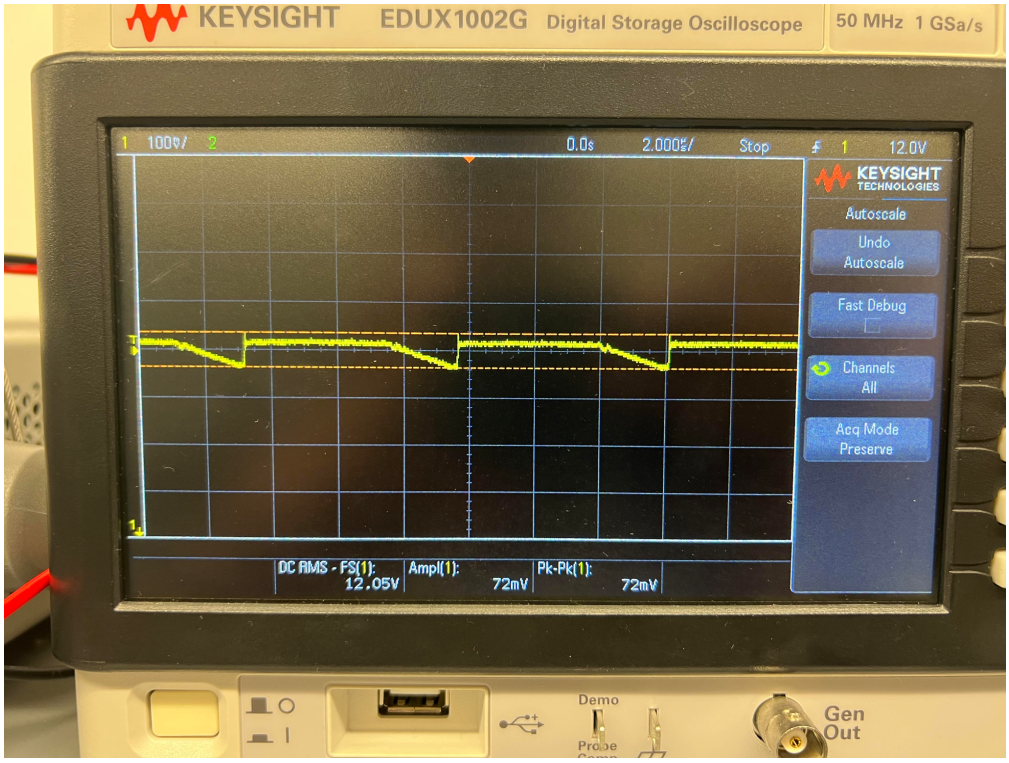


Figure 6: Oscilloscope Image of the Input Voltage (12V)

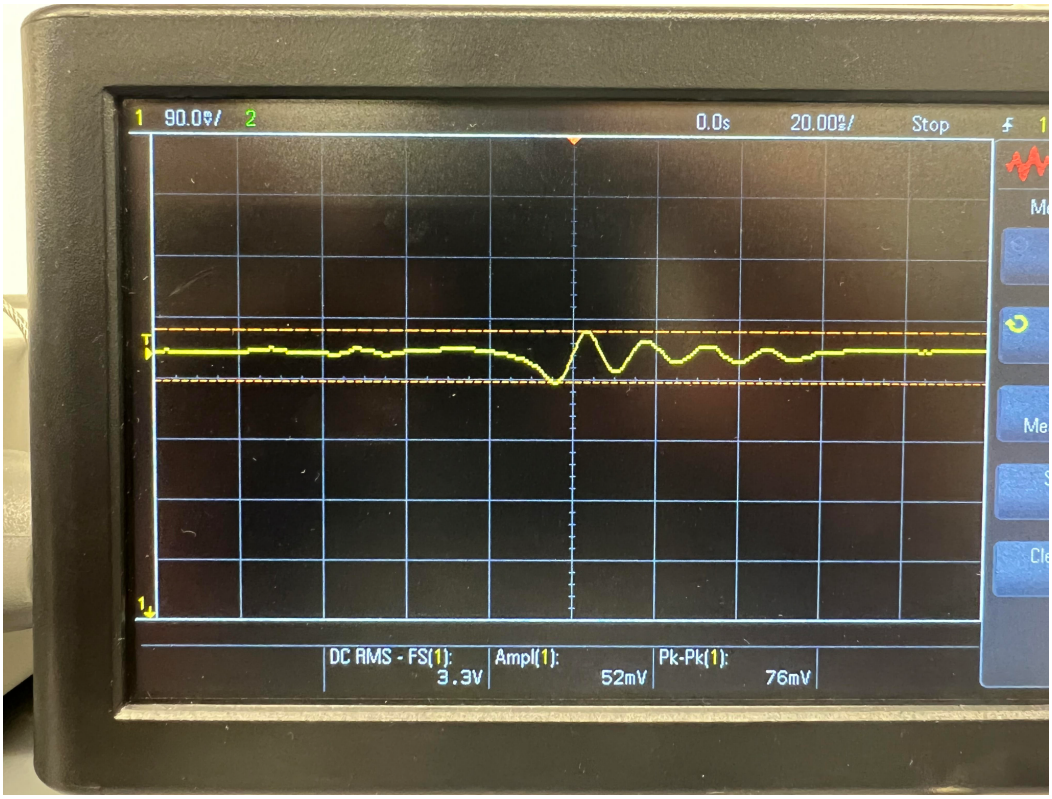


Figure 7: Oscilloscope Image of the Output Voltage (3.3V)

2.4. Subsystem Conclusion

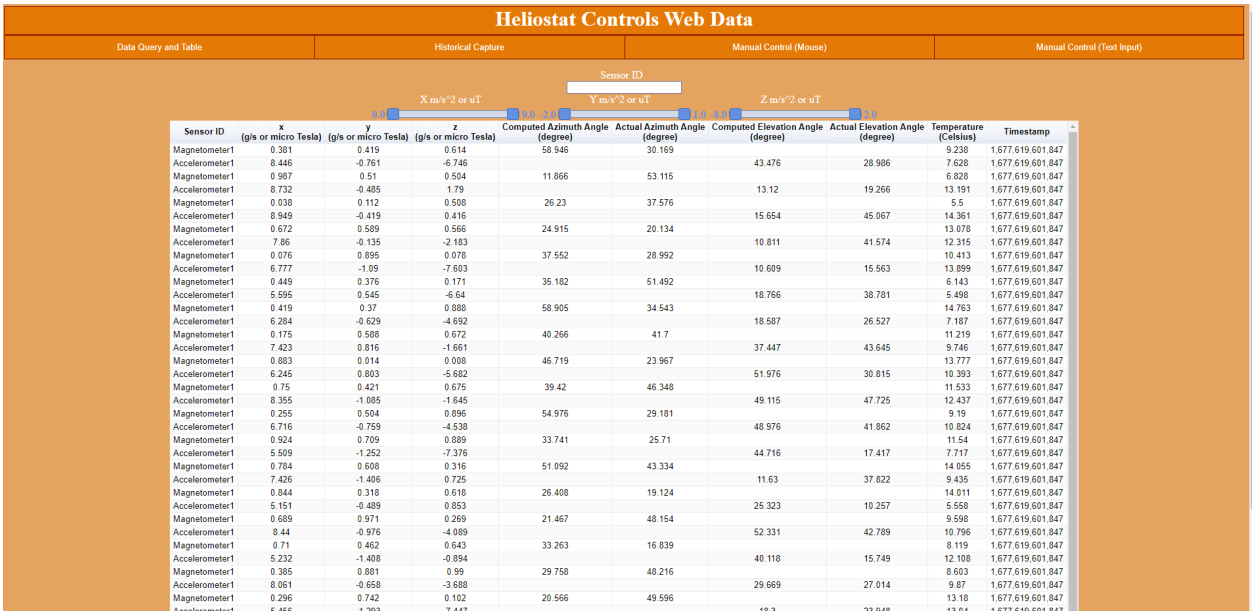
From the validation tests shown in the section above, the PCB subsystem works as planned. This shows that the 12V battery that is being used for the whole project can be stepped down to power the ESP32 and USB-UART bridge. With the ESP32 and USB-UART bridge being operational, code can be written to these devices that will be able to control the motors of the project.

3. Data Visualization and User Interface

3.1. Introduction

The data visualization and user interface seeks to provide human machine interaction to the heliostat energy system. This interaction is done through use of two databases, a front-end web application, a backend lambda script, and a gateway API.

This subsystem involved creating a web application to provide a front-end visualization for the sensor readings stored in the online database of the system. This was done through using javascript, html, css, and hosted on the google software suite as a firebase web app. This software provides support for hosting websites on the cloud and a NoSQL database that can be queried as JSON HTTP GET requests to communicate with the microcontroller for manual control. Additionally, the subsystem involved working with Amazon Web Services (AWS) to integrate a MySQL database for storing data from sensors in a structured manner. The services used are referred to as AWS Lambda (Serverless Computing Service where database querying happens), AWS RDS (Relational Database Server where MySQL database stores sensor data), and AWS API Gateway (where data can be requested from the database on the website and displayed providing visual insight and historical capture).



Sensor ID	x (g/s or micro Tesla)	y (g/s or micro Tesla)	z (g/s or micro Tesla)	Computed Azimuth Angle (degree)	Actual Azimuth Angle (degree)	Computed Elevation Angle (degree)	Actual Elevation Angle (degree)	Temperature (Celsius)	Timestamp
Magnetometer1	0.381	0.419	0.614	58.946	30.169			9.238	1,677,619,601,847
Accelerometer1	8.446	-0.761	-6.746			43.476	28.986	7.628	1,677,619,601,847
Magnetometer1	0.987	0.51	0.504	11.866	53.115			6.828	1,677,619,601,847
Accelerometer1	8.732	-0.485	1.79			13.12	19.266	13.191	1,677,619,601,847
Magnetometer1	0.038	0.112	0.508	26.23	37.576			5.5	1,677,619,601,847
Accelerometer1	8.949	-0.419	0.416			15.654	45.067	14.361	1,677,619,601,847
Magnetometer1	0.672	0.589	0.566	24.915	20.134			13.078	1,677,619,601,847
Accelerometer1	7.86	-0.135	-2.183			10.811	41.574	12.315	1,677,619,601,847
Magnetometer1	0.076	0.895	0.078	37.552	28.992			10.413	1,677,619,601,847
Accelerometer1	6.777	-1.09	-7.603			10.609	15.563	13.899	1,677,619,601,847
Magnetometer1	0.449	0.376	0.171	35.182	51.492			6.143	1,677,619,601,847
Accelerometer1	5.595	0.545	-6.64			18.766	38.781	5.498	1,677,619,601,847
Magnetometer1	0.419	0.37	0.888	58.905	34.543			14.763	1,677,619,601,847
Accelerometer1	6.284	-0.629	-4.692			18.587	26.527	7.187	1,677,619,601,847
Magnetometer1	0.175	0.588	0.672	40.266	41.7			11.219	1,677,619,601,847
Accelerometer1	7.423	0.816	-1.661			37.447	43.645	9.746	1,677,619,601,847
Magnetometer1	0.883	0.014	0.008	46.719	23.967			13.777	1,677,619,601,847
Accelerometer1	6.245	0.803	-5.682			51.976	30.815	10.393	1,677,619,601,847
Magnetometer1	0.75	0.421	0.675	39.42	46.348			11.533	1,677,619,601,847
Accelerometer1	8.355	-1.085	-1.645			49.115	47.725	12.437	1,677,619,601,847
Magnetometer1	0.255	0.504	0.896	54.976	29.181			9.19	1,677,619,601,847
Accelerometer1	6.716	-0.759	-4.538			48.976	41.862	10.824	1,677,619,601,847
Magnetometer1	0.924	0.709	0.889	33.741	25.71			11.54	1,677,619,601,847
Accelerometer1	5.509	-1.252	-7.376			44.716	17.417	7.717	1,677,619,601,847
Magnetometer1	0.784	0.608	0.316	51.092	43.334			14.055	1,677,619,601,847
Accelerometer1	7.426	-1.406	0.725			11.63	37.822	9.435	1,677,619,601,847
Magnetometer1	0.844	0.318	0.618	26.408	19.124			14.011	1,677,619,601,847
Accelerometer1	5.151	-0.489	0.853			25.323	10.257	5.558	1,677,619,601,847
Magnetometer1	0.689	0.971	0.269	21.467	48.154			9.598	1,677,619,601,847
Accelerometer1	8.44	-0.976	-4.089			52.331	42.789	10.796	1,677,619,601,847
Magnetometer1	0.71	0.462	0.643	33.263	16.839			8.119	1,677,619,601,847
Accelerometer1	5.232	-1.408	-0.894			40.118	15.749	12.108	1,677,619,601,847
Magnetometer1	0.385	0.881	0.99	29.758	48.216			8.603	1,677,619,601,847
Accelerometer1	8.061	-0.658	-3.688			29.669	27.014	9.87	1,677,619,601,847
Magnetometer1	0.296	0.742	0.102	20.566	49.596			13.18	1,677,619,601,847

Figure 8: Heliostat Controls Website Data Query and Table Page

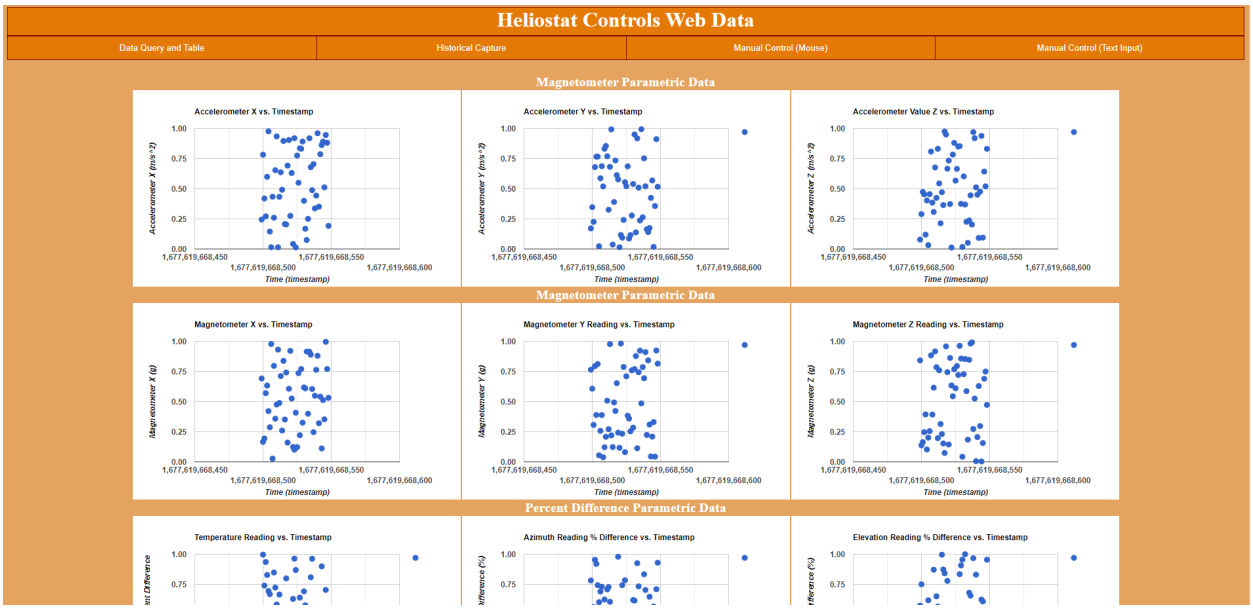


Figure 9: Heliostat Controls Website Historical Capture Page

Figure 10: Heliostat Controls Website Manual Control Page

3.2. Firebase Platform Web Application and NoSQL Query

Firebase is a software development platform provided by Google Inc. This software was used to host the website as a firebase application and provide an endpoint for the microcontroller to receive manual commands from the website. The website was coded in HTML, CSS, and Javascript. Additionally, the platform provides access to a NoSQL database which can be used as a web endpoint for querying manual control data from the joystick element on the page. These endpoints can be seen in the firebase real time NoSQL database below.

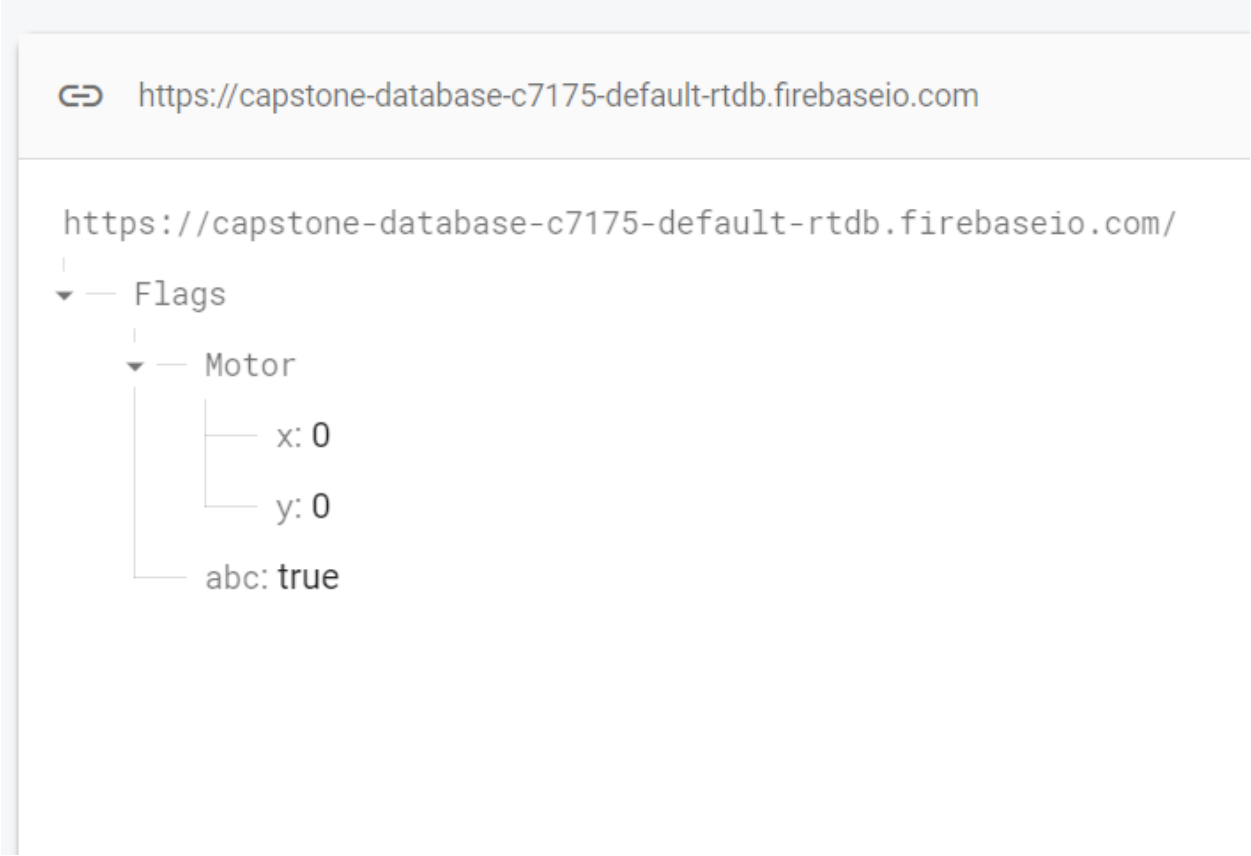


Figure 11: PWM Flags as endpoints from Firebase dashboard

3.3. AWS Microservices (Lambda, RDS, and API Gateway)

Amazon web services provides many web services that can be used for various tasks in an application. For the sake of this project three microservices were used: AWS Lambda, AWS RDS, and AWS API Gateway. AWS Lambda is a serverless computing service that serves

as the backend for the website. The subsystem utilizes two lambda functions written in python 3.8. The first lambda function receives sensor data from the microcontroller through HTTP POST request, opens a connection to the MySQL database, and appends the new entries to the data table. The second lambda function opens a connection to the MySQL database and pulls all the data into a JSON format, which can be queried as a get request from an endpoint provided and triggered by AWS API Gateway. The second lambda function is used to populate the sensor data in the historical capture and data query and table pages in the website. AWS RDS is a relational database service which hosts the web server storing the MySQL database. This database is queried in the second lambda function. AWS API Gateway provides the trigger to the lambda functions and endpoints to where the HTTP requests can be called. These HTTP requests are fundamental in allowing data transfer between different technology platforms and services.

3.4. Validation

Validation for 3.2. Firebase Platform Web Application and NoSQL Query*:*

The first functional test seen below in *Figure 12* shows how the joystick element in the manual controls page can update the flags in the database through the time-defined interval functions created in javascript. Once the joystick is an element, these functions are called periodically on the order of milliseconds and immediately update the NoSQL FIrebase RTDB (Real-Time Database).

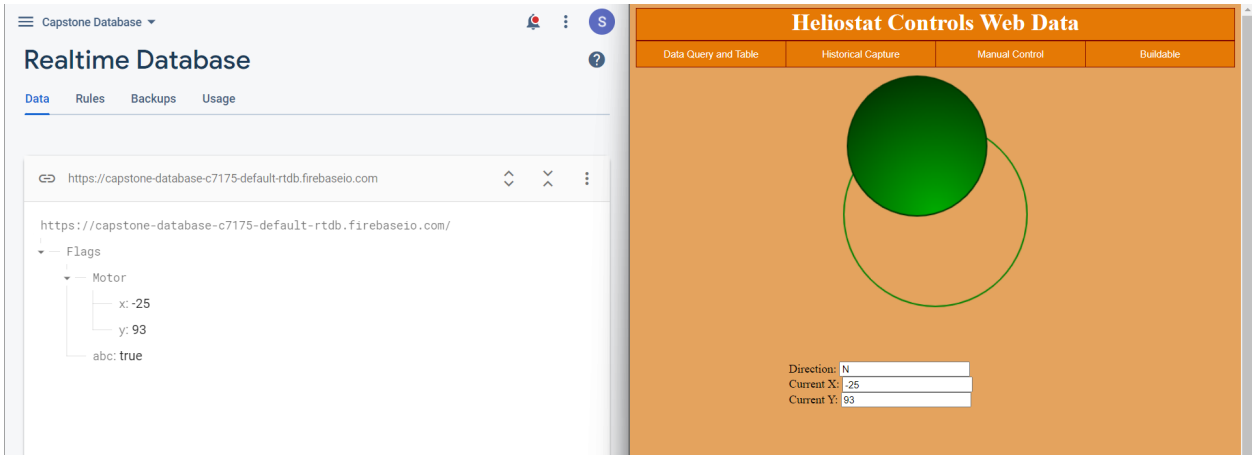


Figure 12: Manual Control and Communication Test

\

Another test involving the manual controls page is seen below. Rather than granting all users access to the database a try catch block is implemented in the javascript code when the manual control button is clicked. This try catch block prompts the user for a password stored within the NoSQL RTDB. With the user-provided password a query is then done to the password flag in the database and if there is a match, the joystick is loaded into the page. Otherwise, the catch block of the code is taken, because an error is returned from trying to query a key that is non-existent. Then the user is prompted that the password is wrong. The dialogue is seen below in *Figure 13.* And the two outcomes previously described are seen when the password is wrong in *Figure 14* and right in *Figure 15.*

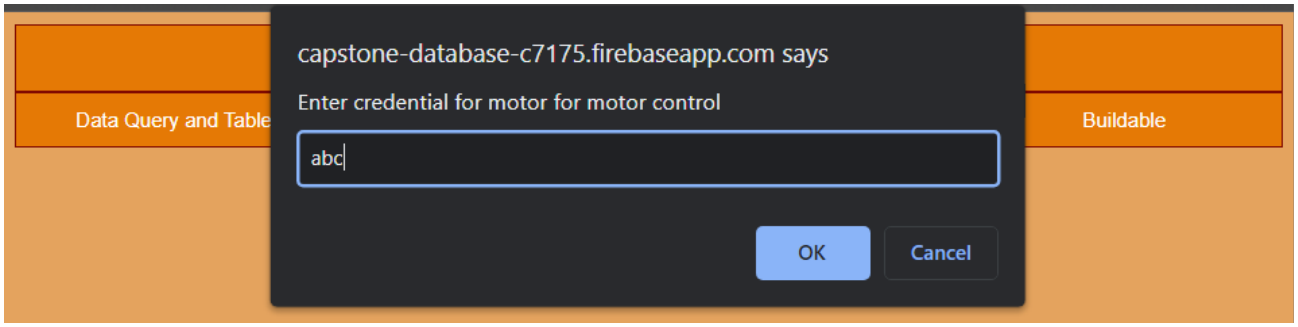


Figure 13: Password Test Prompt

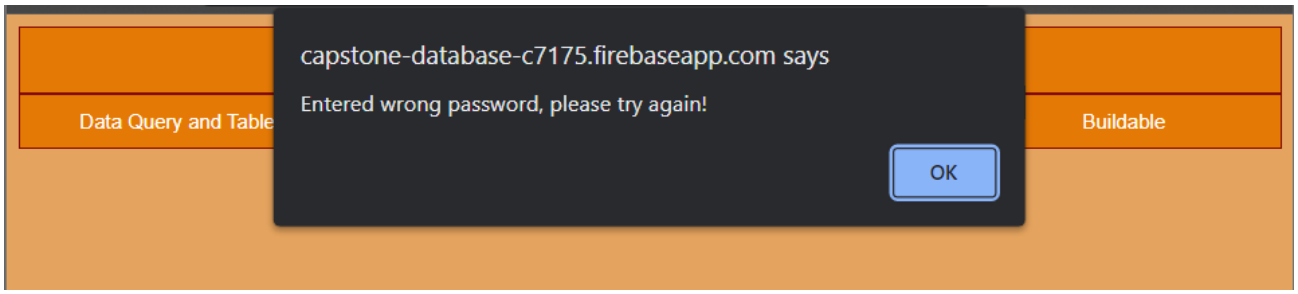


Figure 14: Password Test Wrong Answer

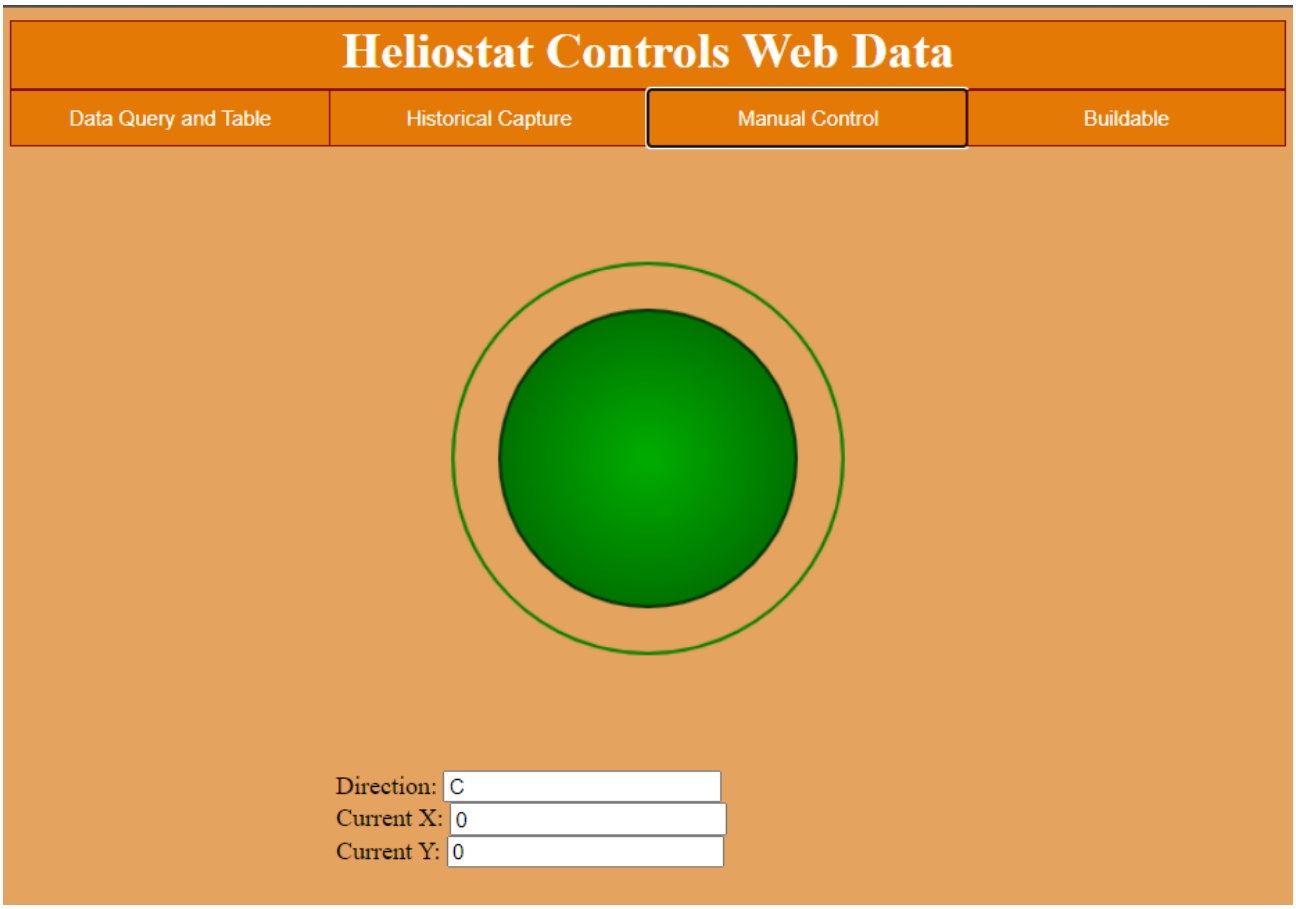
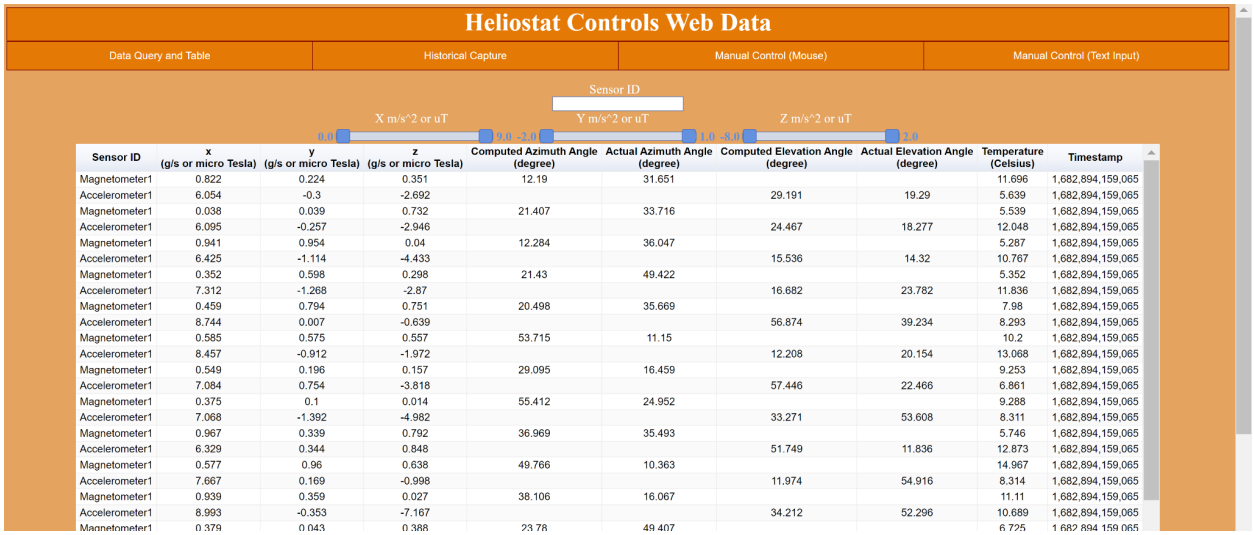


Figure 15: Correct Password with Access Granted

Validation for 3.3. AWS Web Services



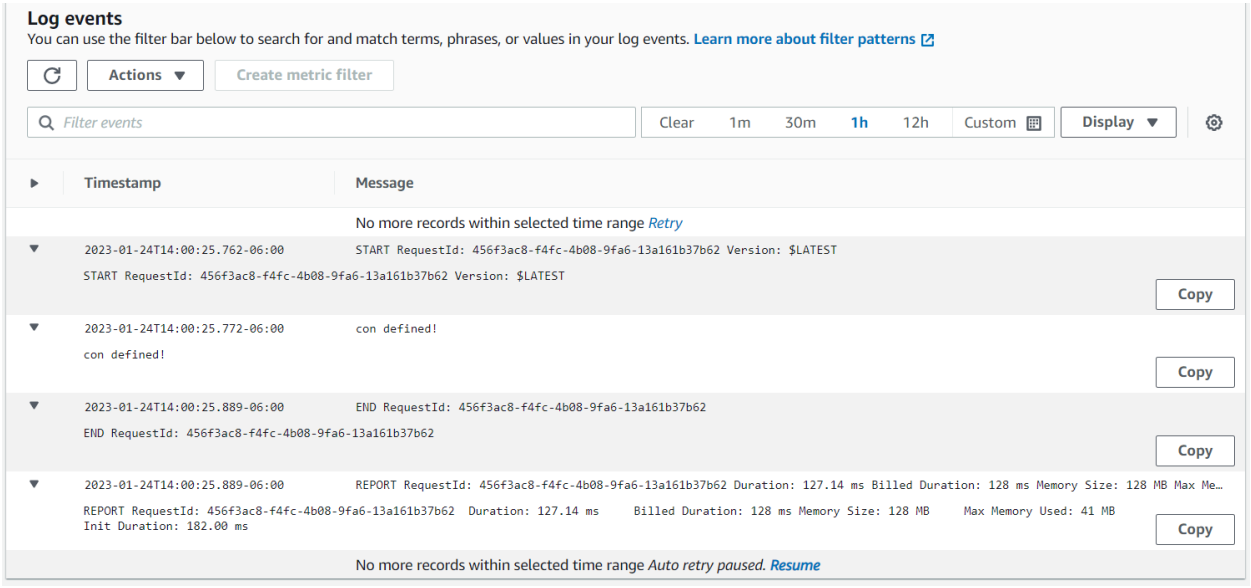
Heliostat Controls Web Data

Data Query and Table | Historical Capture | Manual Control (Mouse) | Manual Control (Text Input)

Sensor ID

X m/s^2 or uT 0.0 9.0 | Y m/s^2 or uT -2.0 1.0 | Z m/s^2 or uT -8.0 2.0

Sensor ID	x (g/s or micro Tesla)	y (g/s or micro Tesla)	z (g/s or micro Tesla)	Computed Azimuth Angle (degree)	Actual Azimuth Angle (degree)	Computed Elevation Angle (degree)	Actual Elevation Angle (degree)	Temperature (Celsius)	Timestamp
Magnetometer1	0.822	0.224	0.351	12.19	31.651			11.696	1,682,894,159,065
Accelerometer1	6.054	-0.3	-2.692			29.191	19.29	5.639	1,682,894,159,065
Magnetometer1	0.038	0.039	0.732	21.407	33.716			5.539	1,682,894,159,065
Accelerometer1	6.095	-0.257	-2.946			24.467	18.277	12.048	1,682,894,159,065
Magnetometer1	0.941	0.954	0.04	12.284	36.047			5.287	1,682,894,159,065
Accelerometer1	6.425	-1.114	-4.433			15.536	14.32	10.767	1,682,894,159,065
Magnetometer1	0.352	0.598	0.298	21.43	49.422			5.352	1,682,894,159,065
Accelerometer1	7.312	-1.268	-2.87			16.682	23.782	11.836	1,682,894,159,065
Magnetometer1	0.459	0.794	0.751	20.498	35.669			7.98	1,682,894,159,065
Accelerometer1	8.744	0.007	-0.639			56.874	39.234	8.293	1,682,894,159,065
Magnetometer1	0.585	0.575	0.557	53.715	11.15			10.2	1,682,894,159,065
Accelerometer1	8.457	-0.912	-1.972			12.208	20.154	13.068	1,682,894,159,065
Magnetometer1	0.549	0.196	0.157	29.095	16.459			9.253	1,682,894,159,065
Accelerometer1	7.084	0.754	-3.818			57.446	22.466	6.861	1,682,894,159,065
Magnetometer1	0.375	0.1	0.014	55.412	24.952			9.288	1,682,894,159,065
Accelerometer1	7.068	-1.392	-4.982			33.271	53.608	8.311	1,682,894,159,065
Magnetometer1	0.967	0.339	0.792	36.969	35.493			5.746	1,682,894,159,065
Accelerometer1	6.329	0.344	0.848			51.749	11.836	12.873	1,682,894,159,065
Magnetometer1	0.577	0.96	0.638	49.766	10.363			14.967	1,682,894,159,065
Accelerometer1	7.667	0.169	-0.998			11.974	54.916	8.314	1,682,894,159,065
Magnetometer1	0.939	0.359	0.027	38.106	16.067			11.11	1,682,894,159,065
Accelerometer1	8.993	-0.353	-7.167			34.212	52.296	10.689	1,682,894,159,065
Magnetometer1	0.379	0.043	0.368	23.78	49.407			6.725	1,682,894,159,065



Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

Actions | Create metric filter

Filter events | Clear | 1m | 30m | 1h | 12h | Custom | Display

Timestamp	Message
	No more records within selected time range *Retry*
2023-01-24T14:00:25.762-06:00	START RequestId: 456f3ac8-f4fc-4b08-9fa6-13a161b37b62 Version: $LATEST
	START RequestId: 456f3ac8-f4fc-4b08-9fa6-13a161b37b62 Version: $LATEST — Copy
2023-01-24T14:00:25.772-06:00	con defined!
	con defined! — Copy
2023-01-24T14:00:25.889-06:00	END RequestId: 456f3ac8-f4fc-4b08-9fa6-13a161b37b62
	END RequestId: 456f3ac8-f4fc-4b08-9fa6-13a161b37b62 — Copy
2023-01-24T14:00:25.889-06:00	REPORT RequestId: 456f3ac8-f4fc-4b08-9fa6-13a161b37b62 Duration: 127.14 ms Billed Duration: 128 ms Memory Size: 128 MB Max Me…
	REPORT RequestId: 456f3ac8-f4fc-4b08-9fa6-13a161b37b62 Duration: 127.14 ms Billed Duration: 128 ms Memory Size: 128 MB Max Memory Used: 41 MB Init Duration: 182.00 ms — Copy
	No more records within selected time range *Auto retry paused.* *Resume*

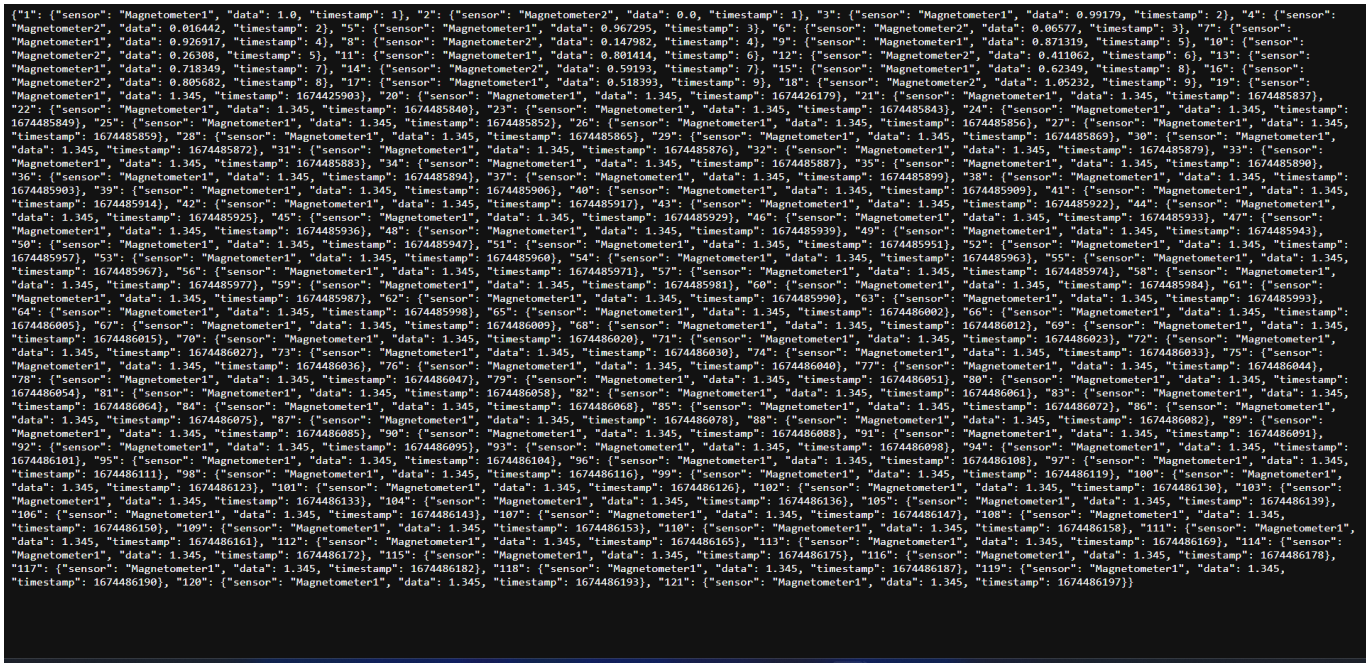

{"1": {"sensor": "Magnetometer1", "data": 1.0, "timestamp": 1}, "2": {"sensor": "Magnetometer2", "data": 0.0, "timestamp": 1}, "3": {"sensor": "Magnetometer1", "data": 0.99179, "timestamp": 2}, "4": {"sensor": "Magnetometer2", "data": 0.016442, "timestamp": 2}, "5": {"sensor": "Magnetometer1", "data": 0.967295, "timestamp": 3}, "6": {"sensor": "Magnetometer2", "data": 0.06577, "timestamp": 3}, "7": {"sensor": "Magnetometer1", "data": 0.926917, "timestamp": 4}, "8": {"sensor": "Magnetometer2", "data": 0.147982, "timestamp": 4}, "9": {"sensor": "Magnetometer1", "data": 0.871319, "timestamp": 5}, "10": {"sensor": "Magnetometer2", "data": 0.26308, "timestamp": 5}, "11": {"sensor": "Magnetometer1", "data": 0.801414, "timestamp": 6}, "12": {"sensor": "Magnetometer2", "data": 0.411062, "timestamp": 6}, "13": {"sensor": "Magnetometer1", "data": 0.718349, "timestamp": 7}, "14": {"sensor": "Magnetometer2", "data": 0.59193, "timestamp": 7}, "15": {"sensor": "Magnetometer1", "data": 0.62349, "timestamp": 8}, "16": {"sensor": "Magnetometer2", "data": 0.805682, "timestamp": 8}, "17": {"sensor": "Magnetometer1", "data": 0.518393, "timestamp": 9}, "18": {"sensor": "Magnetometer2", "data": 1.05232, "timestamp": 9}, "19": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674425903}, "20": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674426179}, "21": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485837}, "22": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485840}, "23": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485843}, "24": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485849}, "25": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485852}, "26": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485856}, "27": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485859}, "28": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485865}, "29": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485869}, "30": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485872}, "31": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485876}, "32": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485879}, "33": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485883}, "34": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485887}, "35": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485890}, "36": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485894}, "37": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485899}, "38": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485903}, "39": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485906}, "40": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485909}, "41": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485914}, "42": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485917}, "43": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485922}, "44": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485925}, "45": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485929}, "46": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485933}, "47": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485936}, "48": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485939}, "49": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485943}, "50": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485947}, "51": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485951}, "52": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485957}, "53": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485960}, "54": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485963}, "55": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485967}, "56": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485971}, "57": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485974}, "58": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485977}, "59": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485981}, "60": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485984}, "61": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485987}, "62": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485990}, "63": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485993}, "64": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674485998}, "65": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486002}, "66": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486005}, "67": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486009}, "68": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486012}, "69": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486015}, "70": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486020}, "71": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486023}, "72": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486027}, "73": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486030}, "74": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486033}, "75": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486036}, "76": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486040}, "77": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486044}, "78": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486047}, "79": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486051}, "80": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486054}, "81": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486058}, "82": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486061}, "83": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486064}, "84": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486068}, "85": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486072}, "86": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486075}, "87": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486078}, "88": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486082}, "89": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486085}, "90": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486088}, "91": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486091}, "92": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486095}, "93": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486098}, "94": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486101}, "95": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486104}, "96": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486108}, "97": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486111}, "98": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486116}, "99": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486119}, "100": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486123}, "101": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486126}, "102": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486130}, "103": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486133}, "104": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486136}, "105": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486139}, "106": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486143}, "107": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486147}, "108": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486150}, "109": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486153}, "110": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486158}, "111": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486161}, "112": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486165}, "113": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486169}, "114": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486172}, "115": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486175}, "116": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486178}, "117": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486182}, "118": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486187}, "119": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486190}, "120": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486193}, "121": {"sensor": "Magnetometer1", "data": 1.345, "timestamp": 1674486197}}

Figure 16: Website HTTP Request , AWS Lambda Log Event, and API Trigger Update

id	sensor	x	y	z	actual_azimuth	computed_azimuth	actual_elevation	computed_elevation
1	Accelerometer1	0.395461	0.973715	0.670713	NULL	NULL	0.026368	0.909253
2	Magnetometer1	0.98161	0.25987	0.599293	0.71999	0.186637	NULL	NULL
3	Accelerometer1	0.949557	0.831022	0.880158	NULL	NULL	0.198748	0.535919
4	Magnetometer1	0.324192	0.844821	0.895902	0.316787	0.012922	NULL	NULL
5	Accelerometer1	0.266954	0.86204	0.939823	NULL	NULL	0.106907	0.960907
6	Magnetometer1	0.685833	0.509794	0.201472	0.967613	0.575299	NULL	NULL
7	Accelerometer1	0.232406	0.397718	0.573122	NULL	NULL	0.084013	0.027435
8	Magnetometer1	0.843697	0.411942	0.64876	0.873144	0.019995	NULL	NULL
9	Accelerometer1	0.057032	0.390471	0.273266	NULL	NULL	0.361616	0.752366
10	Magnetometer1	0.23833	0.599019	0.049246	0.568528	0.897205	NULL	NULL
11	Accelerometer1	0.119124	0.597196	0.191398	NULL	NULL	0.496061	0.350163

Figure 17: AWS RDS MySQL Connection and Query Test through Dashboard Software

3.5. Subsystem Conclusion

Through the validation and diagnostic and mitigation section of the report above, the subsystem has functionality that is currently sufficient for the purposes of this project. The services used provide the user interface and data capture needed for the scope of the functional requirements of this project

Fresnel Lens Control Tracking

Samuel Dixon
Jordan George

SYSTEM REPORT

REVISION – Draft A
April 29, 2023

Subsystem Report
FOR
Fresnel Lens Control Tracking

Team <66>

Approved by:

Jordan George 4/29/2023

_____

Project Leader Date

Dr. Nowka 4/29/2023

_____

Prof. Kalafatis Date

Dalton Cyr 4/29/2023

_____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
A	4/29/2023	Samuel Dixon, Jordan George		Draft Release

Table of Contents

List of Tables

List of Figures

1. Overview

The goal of this project was to create a control system for a fresnel lens unit that would automatically follow the sun throughout the day. What was developed in this project was a web application, PCB, and a coded algorithm to track the sun for motor movement. The whole system is powered by a 12V battery that powers the PCB and motor driver. The PCB has an ESP32 that code was flashed onto by the means of a USB port. For the automatic movement algorithm to work, it needed a handful of input parameters. These parameters include longitude, latitude, timezone, date, and time. The longitude, latitude, and timezone are hardcoded in and can be changed based on location of the fresnel lens frame. The time and date parameters were obtained over Wi-Fi through the Simple Network Time Protocol. Once these parameters were received, the azimuth and elevation angles of the sun were calculated. The azimuth angle is where the sun is with respect to true north, and the elevation angle is the angle of how high the sun is. Based on these angles, the system would rotate to the desired azimuth angle, then tilt down to the proper elevation. The system updates every ten minutes since the sun does not move fast with respect to time. After ten minutes, the ESP32 would request the current time and date stamp to calculate the new solar angles. The difference between the new angles and the previous angles is what was used to determine how much movement in rotation and tilt needed to be done. This process would repeat until the user would unplug the battery to stop the automatic tracking.

2. Development Plan & Execution Plan

2.1. Design Plan

The design for this project involved hardware and software development. The hardware portion of the design had multiple components to ensure system requirements were met, seen in *Table 1* below.

Key System Design Parameters	
Input supply	12 V
ESP32 supply	3.3 V
Motor driver supply	12 V
Trace width max Amperage (motor supply block)	3.2 A
Trace width max Amperage (signal supply block)	1.5 A
PWM Frequency	10 kHz
PWM Signal Logic Level	3.3 V
I2C Frequency	100 kHz
I2C Clock and Data GPIO	22 / 23
Minimable Latency Time for joystick update	500 ms

Table 1: System Design Parameters

Firstly, a WiFi capable MCU was chosen from Espressif Systems, the ESP32-S2 Wroom. Next, a power solution was developed for interfacing the board with a packaged motor driver and GPIO pins configured as both PWM timers and I2C data and clock lines. This power solution also entailed development of a 12 V (system supply battery) to 3.3 V active buck converter to step down the voltage to 3.3 V (microcontroller voltage). Since a 12 V battery was used to power the whole system, there was a terminal block on the PCB to connect to the battery and then another terminal block that split off from the 12 V input to connect the PCB to the motor driver. Then, the hardware to flash compiled code onto the board was integrated onto the PCB board. This was done through USB-UART Integrated Circuit and two buttons on the respective boot and enable pins of the microcontroller. Finally, physical constraints of the PCB had to be considered, such as the trace of the current carrying conductor for the motor driver supply, the terminal blocks to interface wires, and the location of the antenna for the WiFi ESP32 MCU. The motor driver was from a company called Drok. It is called the 160W 2 Channel DC Motor Driver Module. This driver can supply up to 7 Amps for each motor which was more than enough current for the two motors. The rotation motor required about 2.2 Amps and the motor for tilting required about 0.5 Amps. There were six GPIOs as control signals required for the motor driver, two being PWM signals and the other four were digital signals. These control signals were used

to determine the speed and direction of the motors. Also, two 3.3V and two GND pins were required on the motor driver to power the control signal logic.

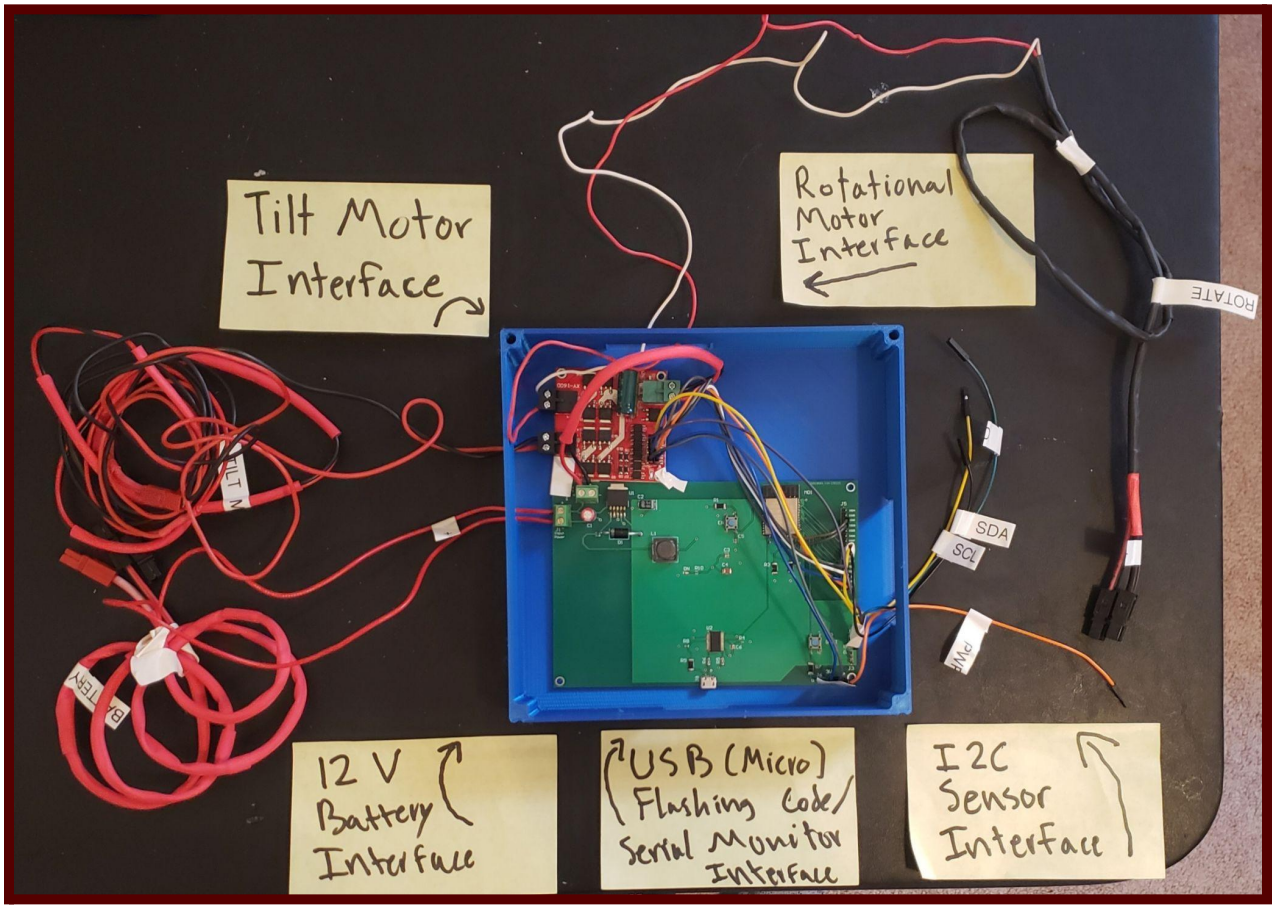


Figure 1: All Hardware Components

The software portion of the design utilized a cross-platform approach that exploited strengths of services offered across two different web service providers: Amazon (AWS) and Google (Firebase). A formal presentation of these considerations is seen below in *Table 2.*

Database and Website Considerations on Changes	
Cross Platform Approach (Firebase Hosting , AWS Backend)	**Exclusively One Platform (Firebase)**
Pros	
• Prevents repeatable entries in Amazon's MySQL database • Utilizes unlimited HTTP free tier requests for joystick PWM speed manual controller • Greater interoperability with other systems and broader exposure to new technologies	• Consolidation of tasks and APIs • Improved Latency for real time database and listener asynchronous capabilities allows for dynamic refresh without javascript interval functions
Cons	
• Complexity working with different technologies and microservices	• Less accessibility and requires greater parsing schemes to obtain data stored in NoSQL format.

Table 2 Web organization considerations and tradeoffs

Firstly, a web application was created to be hosted on google's cloud server to display system metric data and provide a user interface to the system where motors could be used via a joystick. This web application served as the pseudo front-end for the project and was designed through javascript, html, and css among a few javascript libraries, such as the google visualization API. This final result is seen in figure x below.

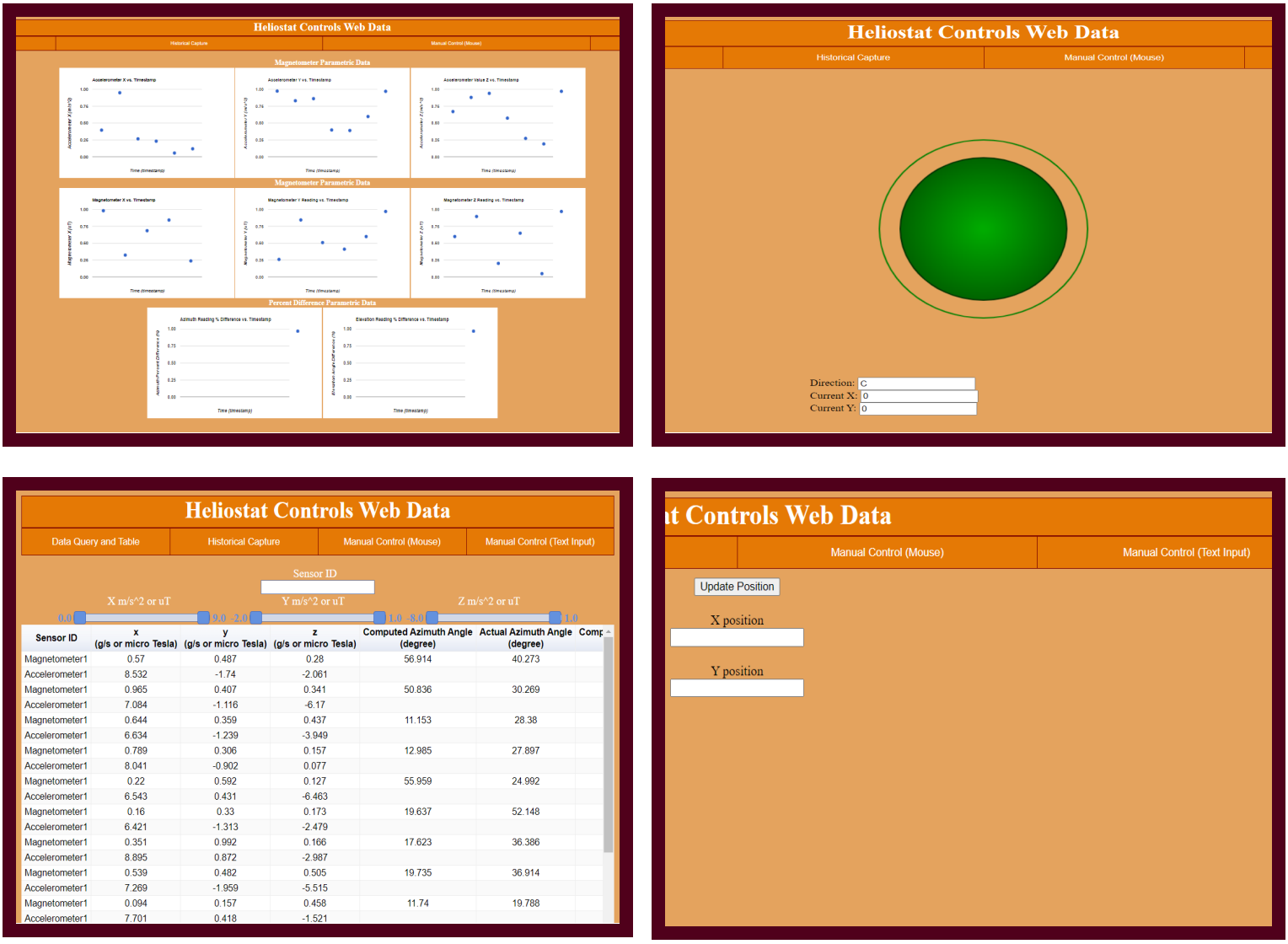


Figure 2: Final views of website activity pages

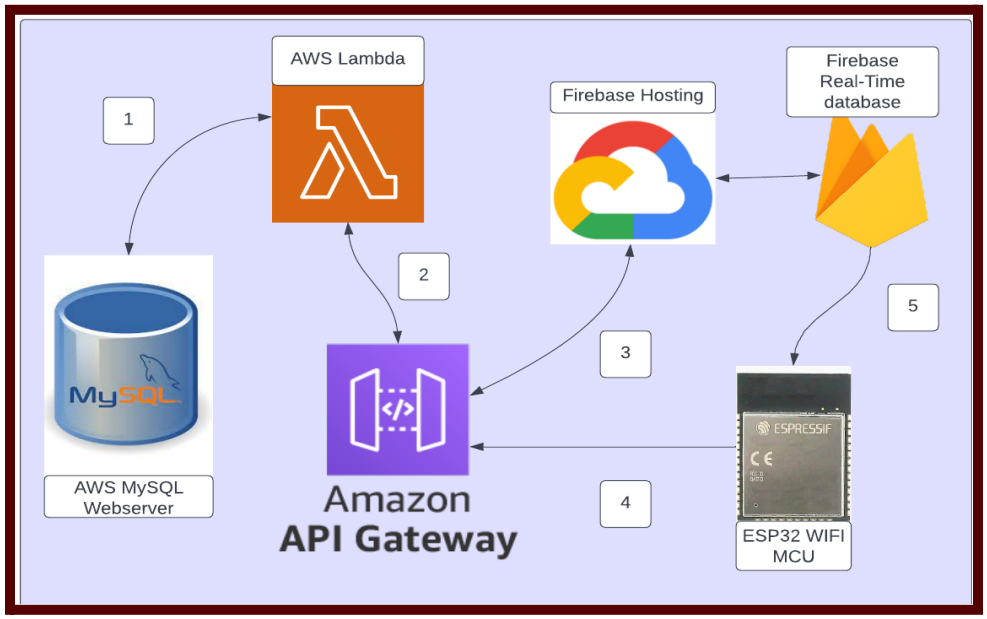


Figure 3: Web Interaction Information Flow

Web Data Communication Guide

1. MySQL Web Server to and from AWS Lambda Serverless Microservice
2. AWS Lambda Serverless to and from API Gateway
3. API Gateway to and from Firebase Hosting
4. ESP32 to API Gateway
5. Firebase Hosting to and from Firebase RTDB (real-time database)

1. AWS lambda handles the backend of posting and receiving data from the MySQL web server
2. API Gateway is an endpoint that triggers get / post actions for the AWS serverless backend lambda script
3. Data is queried through the AWS API Gateway (update-database) which handles receiving (GET) data from the MySQL web server
4. Data is posted from the ESP32 through the AWS API Gateway (update-database) which handles receiving (GET) data from the MySQL web server
5. Communicates X/Y values of PWM joystick from Firebase RTDB in real time with low latency for motor actuation

Table 3: Web Data Guide corresponding to Figure 1 information flow

id	sensor	x	y	z	actual_azimuth	computed_azimuth	actual_elevation	computed_elevation
1	Accelerometer1	0.395461	0.973715	0.670713	NULL	NULL	0.026368	0.909253
2	Magnetometer1	0.98161	0.25987	0.599293	0.71999	0.186637	NULL	NULL
3	Accelerometer1	0.949557	0.831022	0.880158	NULL	NULL	0.198748	0.535919
4	Magnetometer1	0.324192	0.844821	0.895902	0.316787	0.012922	NULL	NULL
5	Accelerometer1	0.266954	0.86204	0.939823	NULL	NULL	0.106907	0.960907
6	Magnetometer1	0.685833	0.509794	0.201472	0.967613	0.575299	NULL	NULL
7	Accelerometer1	0.232406	0.397718	0.573122	NULL	NULL	0.084013	0.027435
8	Magnetometer1	0.843697	0.411942	0.64876	0.873144	0.019995	NULL	NULL
9	Accelerometer1	0.057032	0.390471	0.273266	NULL	NULL	0.361616	0.752366
10	Magnetometer1	0.23833	0.599019	0.049246	0.568528	0.897205	NULL	NULL
11	Accelerometer1	0.119124	0.597196	0.191398	NULL	NULL	0.496061	0.350163

Figure 4: Final Organization of the database

2.2. Execution

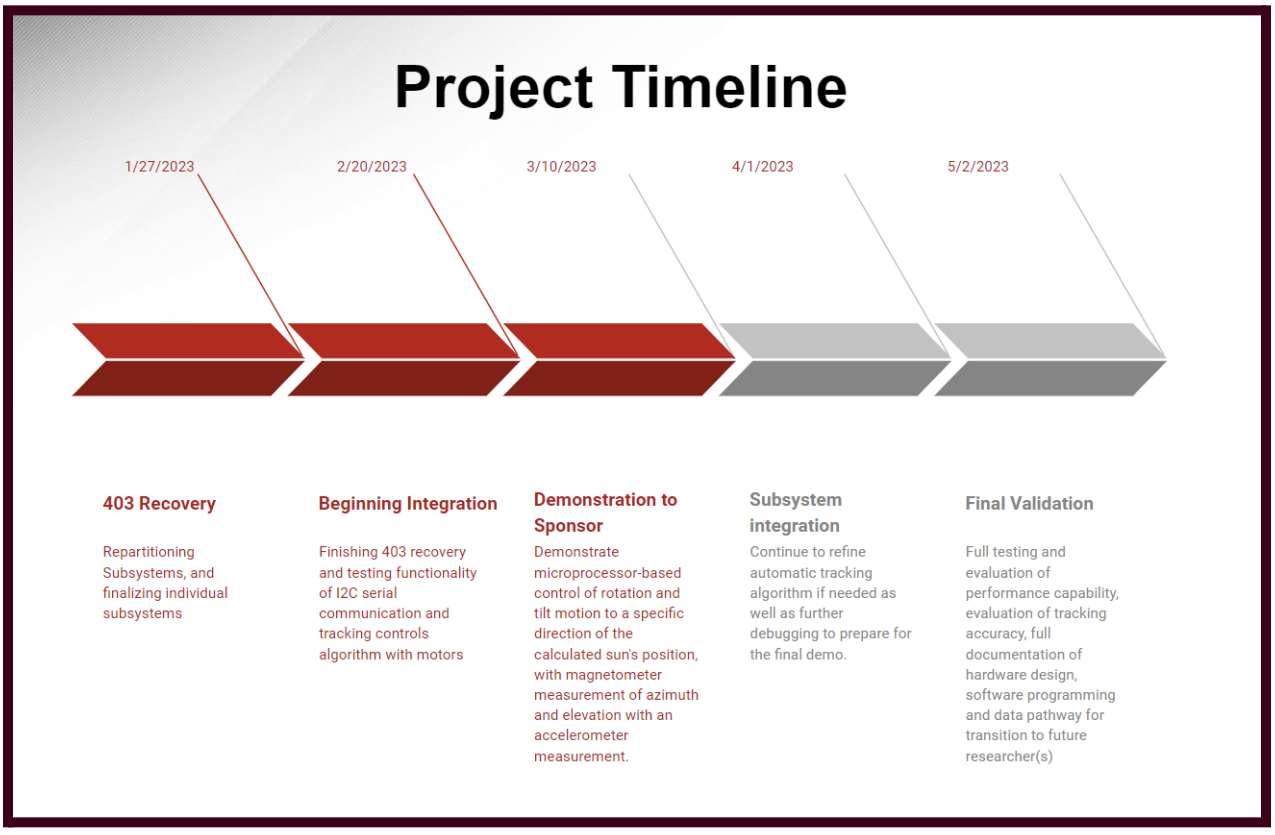


Figure 5: Project Timeline

2.3. Validation

The first thing to be validated on the hardware side of this project was the voltage regulator. The nominal supply voltage is 12V, but it was tested to receive an input voltage of a range from 5-15V to see if the output voltage would still hold at 3.3V. Looking at Table 4, it is clear that the voltage regulator was operating as expected. The next test of validation was seeing if code could be flashed onto the PCB's ESP32. This was accomplished by checking the solder joints and continuity of the USB port, USB-UART chip, and ESP32. Simple programs such as the Hello World and Blinking LED were first flashed onto the ESP32 to test this (Figure 6). Once this was accomplished, the full automatic movement algorithm was flashed onto the PCB, and the motor driver was connected to simple testing motors to check for the desired movement and speed (Figure 7). Once this was all set up, the circuitry was hooked up to the actual system. After ten minutes, the system would update its position. During the wait time, the compass and inclinometer on an iPhone were used to measure where the system was actually rotated and tilted to. Those numbers were compared with the calculated angle and the percent difference of the data points is graphed in Figure 8. The data obtained in *Figure 9* provided insight into the current and voltage being supplied from the motor driver terminal to each of the respective motors.

ESP32 when turned on		
Supply (V)	**ESP32 Voltage (V)**	**Current Drawn From Supply (A)**
5	3.2889	0.103
6	3.2893	0.086
7	3.29	0.08
8	3.3018	0.066
9	3.303	0.059
10	3.3022	0.057
11	3.2977	0.049
12	3.2978	0.045
13	3.2975	0.042
14	3.2968	0.04
15	3.2967	0.038

Table 4: Results from voltage regulator

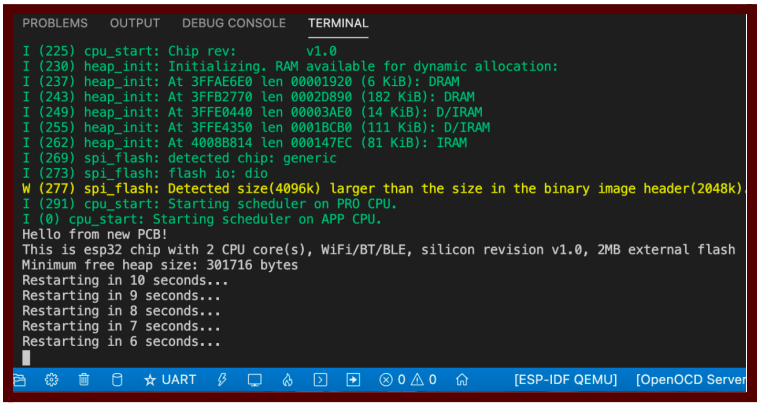


Figure 6: Hello World program flashed onto PCB

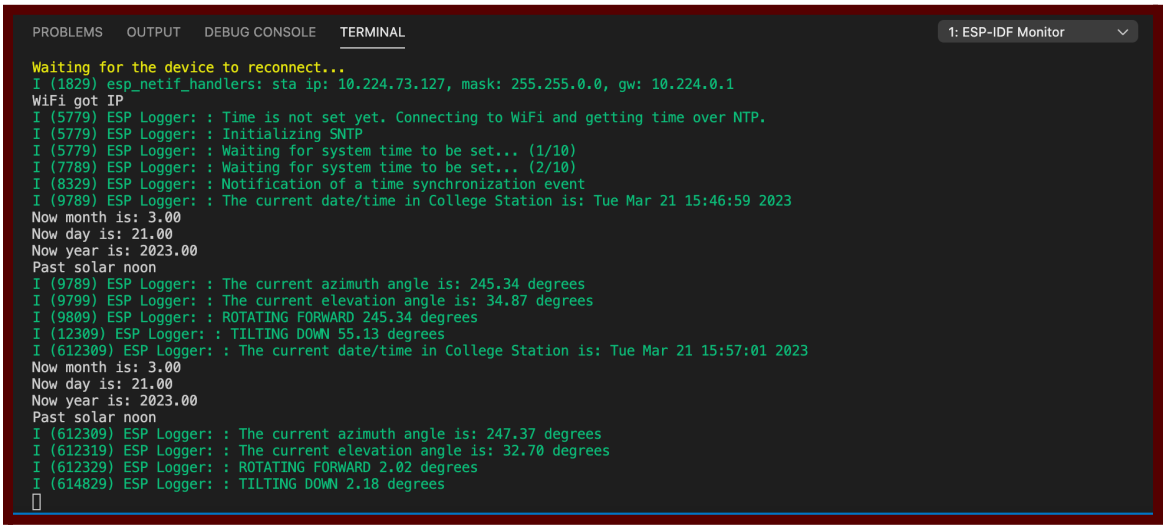


Figure 7: Automatic Movement Program Flashed onto PCB

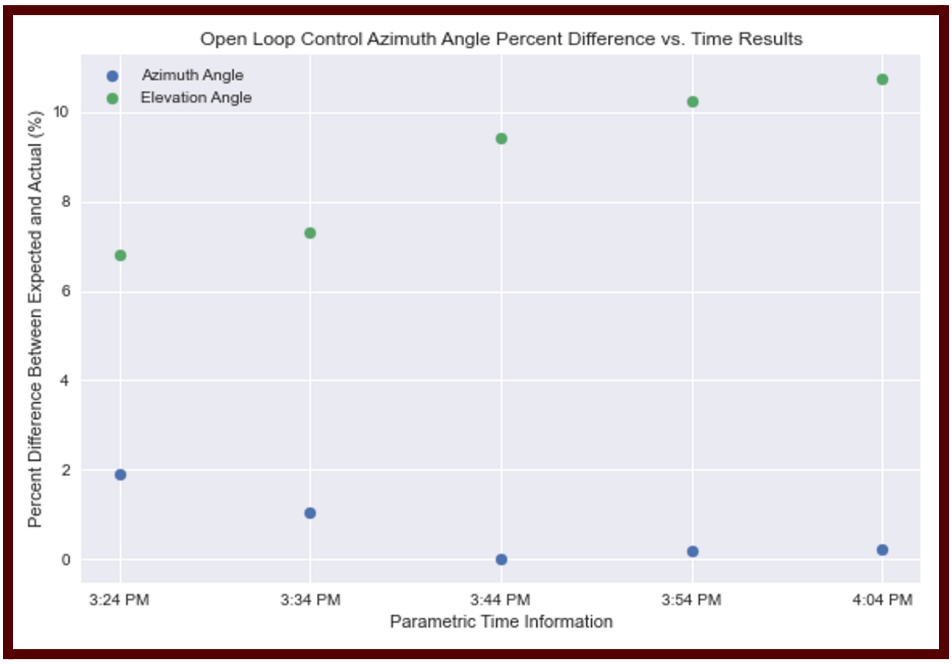


Figure 8: Results from automatic sun tracking

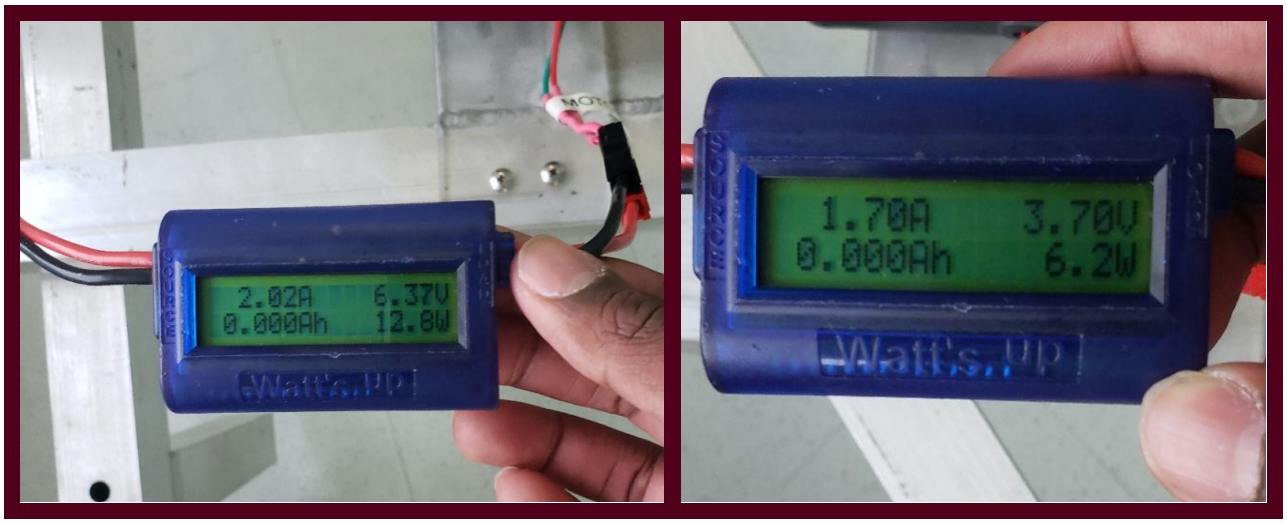


Figure 9: Results from current draw from each respective motor (rotation motor & tilt motor)

The software portion of this project consisted of two main systems and their interactions. Web-related communications and I2C sensor configuration and collection.

Seen below are the results for the MMA8451Q (accelerometer) used in the system in an inclinometer elevation sweep. The accelerometer first began in a vertical position where gravity was seen in one component of the sensor, the x-axis. Then, the accelerometer was slid down and the elevation angle was computed through the arctangent trigonometry function.

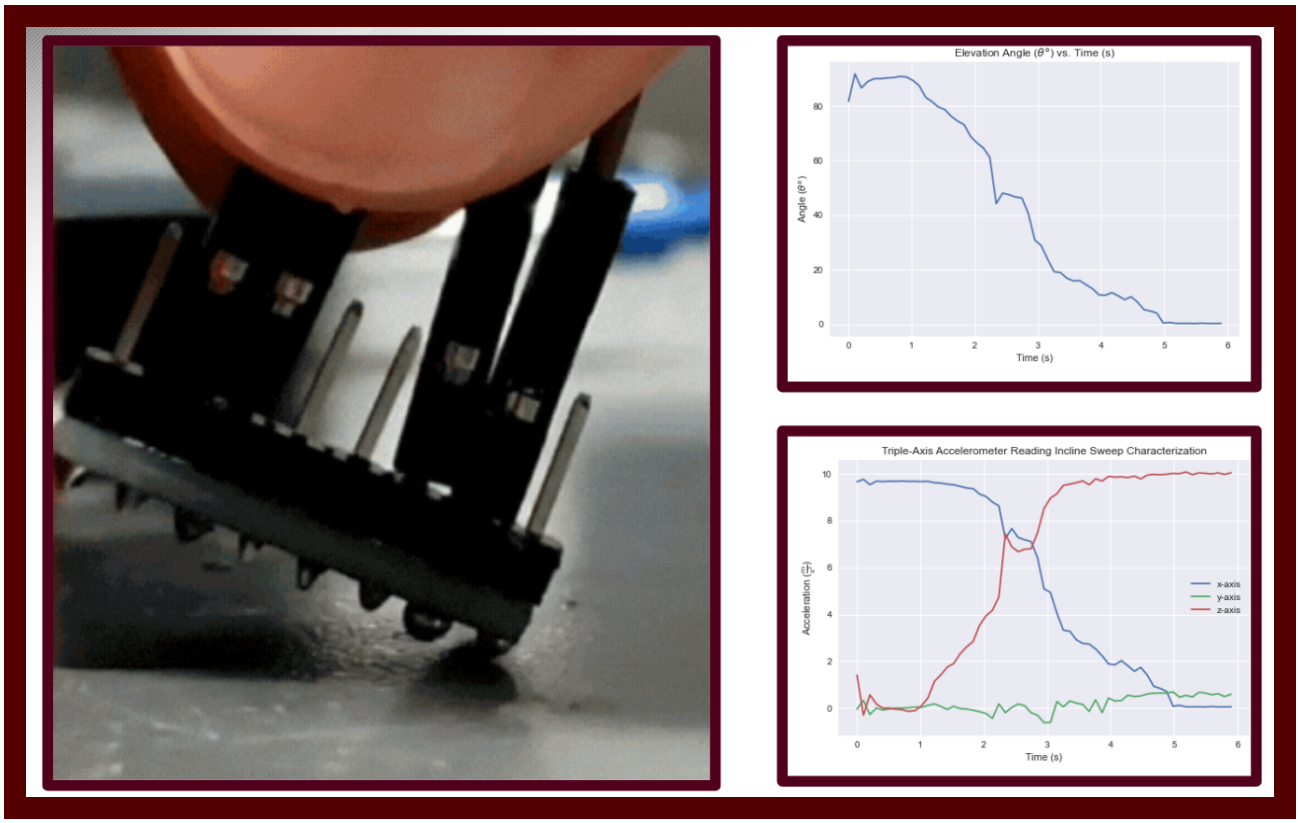


Figure 10: Accelerometer I2C Sensor 2-d sweep and elevation angle computation characterization

The magnetometer seen in *Figure 11* was calibrated using an external tool called MotionCal developed by Paul Stroffegen. The values obtained were used to account for local hard and soft iron offsets. This was done and a compass bearing accuracy was achieved to be withing 2 degrees for a sample data collection of 100 points with a reference magnetometer aligned at magnetic north, or 0 degrees.

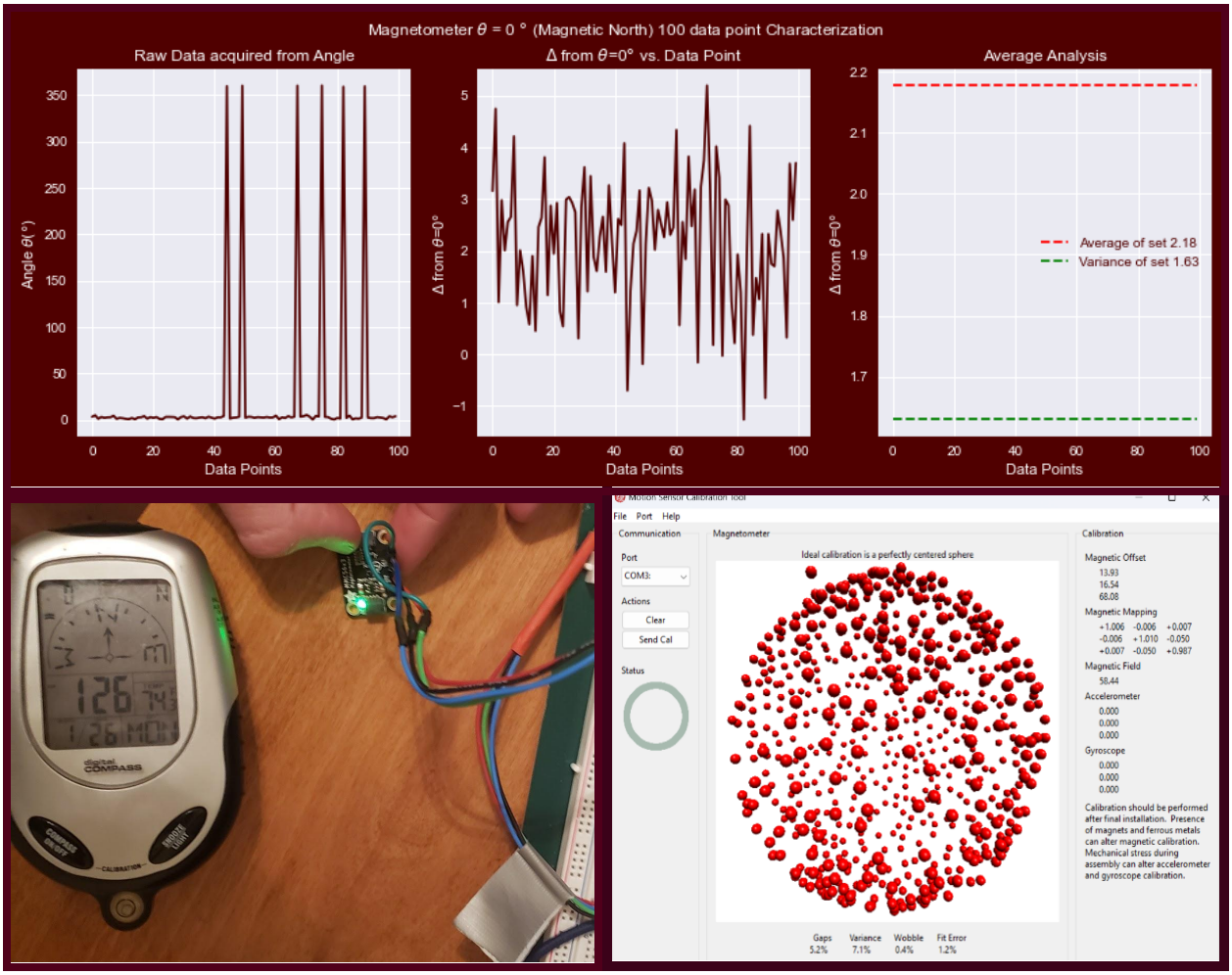


Figure 11: Magnetometer I2C Sensor 2-d magnetic heading and elevation calibration characterization

Device	MMC56 Magnetometer I2C Sensor	Reference Magnetometer
Θ°	2.18	0.00

Table 5: Magnetometer characterization results

3. Conclusion

3.1. Current Status

For this project, a PCB solution integrated with a store-bought motor driver (supplying sufficient motor current) developed through Altium PCB Design Tool was used to actuate motor movement for manual and automatic control algorithms. An open loop control algorithm to track sun motion with a maximum of 11 percent difference Implemented IoT interaction with a website, I2C sensors, and ESP32 MCU to provide real-time system interaction with low latency over TAMU IoT Wi-Fi network. Developed and organized a Relational Database (MySQL) for data storage, website backend, and future analytics. The sensory feedback was not able to be fully implemented due to dynamically changing magnetic fields around on the fresnel lens. So, the magnetometer can provide some feedback on the website that is correct with calibration factors in some regions of the rotational sweep, however the full sweep cannot be used as a robust parameter for closed loop control. On unit, the accelerometer was able to be fully validated and can be used as a parameter for closed loop control for the tilt motor. The data is integrated and viewable in realtime on the website.

3.2. Future Work

For ongoing progress with this project, there are a few changes and improvements that would be beneficial to the project as a whole. One of those being to find a different solution for getting the current time and date stamp. Currently, that stamp is obtained over Wi-Fi. If the fresnel lens unit were to be placed where the connection is not good, this would cause issues. A possible solution would be to integrate a real-time clock that keeps time and date without the use of Wi-Fi. Also, another improvement to the system would be to advance the current open loop control to a closed loop control system. This would require getting the sensors functional for sensor feedback. Sensor feedback would be advantageous for more accurate positioning of the system with respect to the sun's current location.