FireFly

Technical Report

DATE: 4/28/2023

AUTHORS

TRISTAN MASSA

PAYTON BLAKELY

GARETT MORRISON

JOSE SOSA

Individual Contact Information

Kiju Lee	Sponsor/Advisor	kiju.lee@tamu.edu
Tristan Massa	Project Manager	tmass04@tamu.edu
Payton Blakely	Hardware Engineer	payton925@tamu.edu
Garett Morrison	Firmware Engineer	garettm123@tamu.edu
Jose Sosa	Mechanical Engineer	josesosa7@tamu.edu

Team Contact Information

Team Email	fireflyled2023@gmail.com

Table of Contents

List of Figures and Table

List Of Images

List Of Tables

Executive Summary

A novel localization system was created by FireFly and applied to the navigation of a team of rovers. Each robot is equipped with an array of LEDs and multiple proximity sensors. These LEDs are used to identify the rover's position relative to a static camera placed in the room. The proximity sensors detect unexpected obstacles during motion. The Localization Module (LM), reads in image data, processes localization, and sends instructions to the rovers.

FireFly ensures that the robots can be used in a wide range of situations. Within the production facility, a robot can be used for many tasks. These tasks include, delivery services, object tracking, and item movement. In education, these robots can be used to help students learn about robots and their applications in the real world. Existing localization methods are too expensive for STEM educators, and FireFly aims to fill this gap. The goal is to ensure that these robots are cheap, affordable, and reliable. The system is open source, allowing anyone to implement, modify, and build on the platform.

Background

Localization is the process of calculating the relative position of an object in space, and is utilized in a wide range of industries. Motion capture artists use trackers placed on their bodies to calculate their pose for animation. Augmented reality works on mobile phones by tracking the motion of the phone and adjusting the image in real time. The largest application, however, is mobile robotics, where nearly every product utilizes some form of localization for navigation. There is a wide variety of existing systems, each with different benefits and drawbacks.

The most well known is the Global Positioning System (GPS), which locates a receiver relative to an array of orbiting satellites. This is very useful, as it can operate anywhere on earth, but even high-end systems are limited to meters of resolution, which is not nearly precise enough to track small mobile robots.

Similarly, radio navigation requires an array of radio beacons. By comparing signals from different beacons, a receiver can estimate it's relative to each one and triangulate itself from this data. This has similar results to GPS, struggling to calculate precise position and thus being more commonly used for outdoor, mobile applications.

Simultaneous Localization And Mapping (SLAM) is a popular option for rover tracking. This involves processing data from an obstacle detector, such as a LiDAR system, and using it to generate a map of the environment while navigating through the space. This can provide accurate position data without requiring a base station, but needs high resolution input data and requires significant computational power to run.

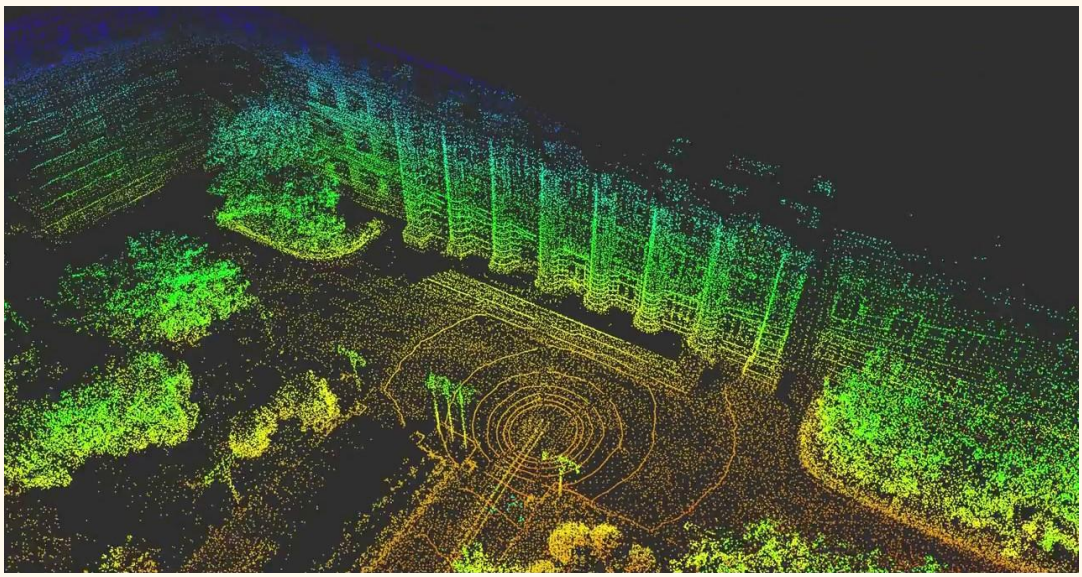


Figure 1. Generated Point Cloud using SLAM

Reflector tracking uses specialized cameras and small reflective spheres or dots placed on the target. These reflective spheres can be individually identified, and provide clean data to positional algorithms in real time. The consistency of the data makes this a popular choice for motion capture setups, but is extremely expensive, as it is reliant on purpose-built high end camera hardware.

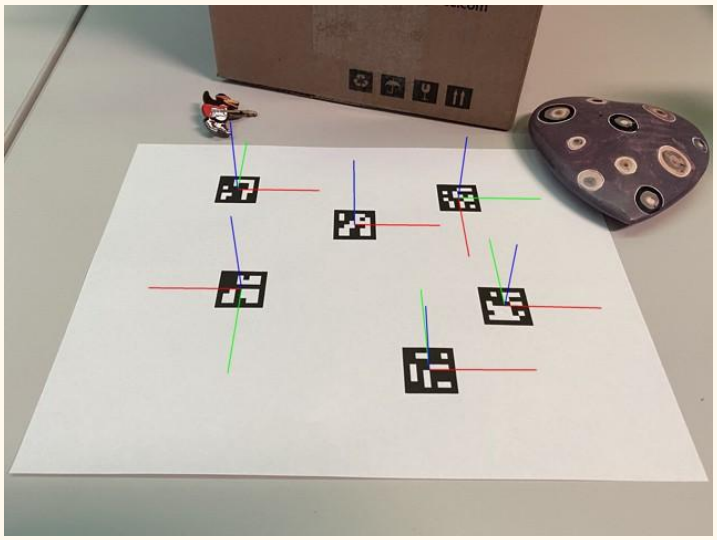


Figure 2. Printed Fiducials being Tracked

Computer vision (CV) identifies objects and features from raw computer data. CV is used in a wide variety of applications, but is frequently applied to localization by tracking features on a mobile platform from a fixed position or tracking fixed objects from a rover. Frequently, fiducial images are printed on paper to allow the algorithm to exclusively look for a few specified patterns. This is by far the cheapest existing option for rover tracking but is severely limited in scope. The tracking algorithm is very processor intensive, requiring searching for distinct geometry across the whole image, and is significantly limited by the camera resolution, as a distant fiducial will only take up a few pixels and be indistinguishable from the background.

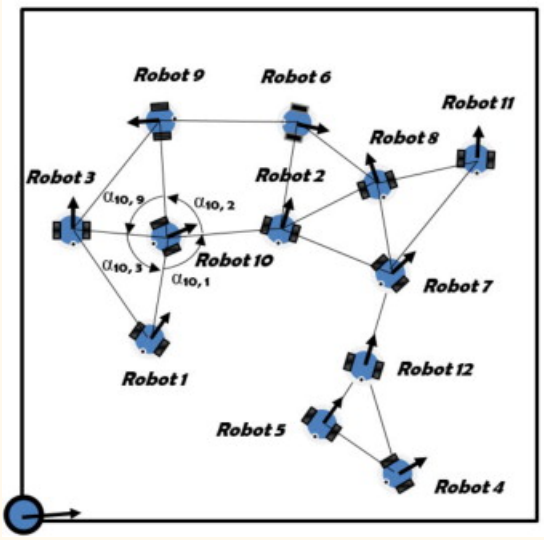


Figure 3. Multi-Robot Localization

Problem Statement

With the price of localization systems being prohibitively expensive, FireFly is seeking to reduce the cost with competitive performance to existing systems. The localization system will be paired with a rover using localization and sensor data to navigate. These robots are capable of performing tasks assigned to them by the localization module.

Solution

FireFly has constructed a scalable team of three robots, using a localization module as the main control hub. To identify the robot's position, LEDs are placed on each robot. Identifying which robot is which is dependent on when the LEDs on the robot flash. In a specific pattern, the LEDs individually flash and adjust its brightness to be visually identified by the camera. The localization module will be capable of communicating with the robots to assign them tasks. This means a majority of the programming will be within the localization module. However, by making the code open-source, it will allow everyone to have access to it to learn about coding and to adjust the program accordingly to their needs.

The team will run in an autonomous nature, where alternatively it can be connected to a computer and run manually. In addition to the LEDs, each robot is equipped with time of flight sensors. The sensors continuously collect data which inform the robot itself to avoid any unexpected obstacles while traveling to its destination.

By utilizing the use of 3D printing and low-cost components FireFly developed an operating LED localization system that is both budget-friendly and efficient.

Concept of Operation

One of the primary objectives is to have the system be accessible to everyone. Open-sourcing the codebase allows anyone to learn from and build on the project. By keeping the localization module hardware-agnostic, users can run localization on their own computers, making the platform available to anyone with a laptop and some basic components.

In a default setting, the robot team will be placed in an area with a camera overlooking the entire space. With activated lights on the robot and in the room, the camera will send data to the localization module to determine the positions within the space. Once calculated, the localization module will wirelessly communicate with the robot to inform it where it should go based on the LED positions received.

Proximity sensors on the robot gather data when the robot is in motion. In traversing to the desired position, the sensors are used as an obstacle avoidance system. This means the robots have their own program which enables them to autonomously complete their tasks in the event of an expected obstacle.

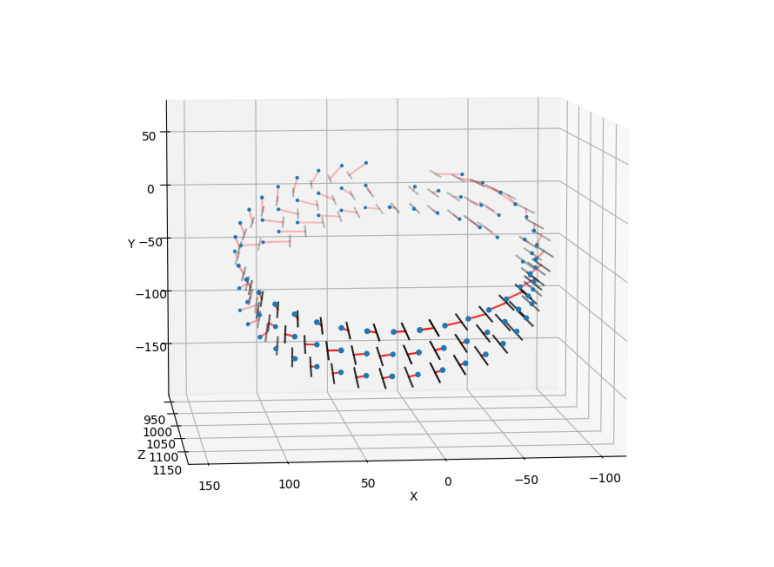


Figure 4. 3D LED Localization Plot

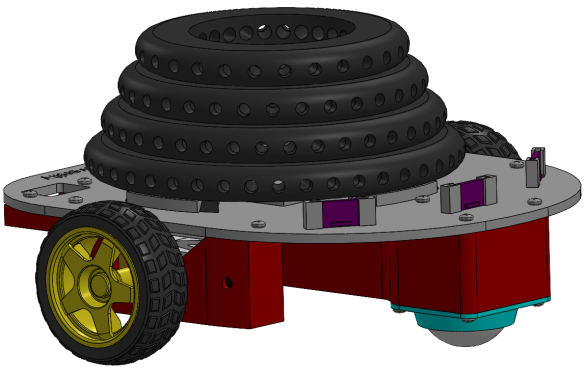


Figure 5. Robot Concept

Functional Requirements

LED Tracking

Each robot in the FireFly team has a minimum of 8 bright LEDs around the body. The LEDs are visible at all points within a room. With the use of a camera placed in an overlooking fixed position, moving LEDs can be tracked. The localization module computes the local position of a robot in the global coordinate system. The LEDs always remain on throughout the entire time the robots remain on. The intensity of the light is important in tracking as the brightness of the room itself correlates to the efficiency of the LEDs being viewed by the camera.

Location Calculation

Also with the use of the camera, desired locations that the team will move toward is viewed. LEDs can be placed in varying positions to identify specific locations within the room. The difference between this and the robots is the moving while the other LEDs are fixed to a location. This allows the localization module to compute the global coordinates of the fixed LEDs to tell the robot where to go. Due to the localization module's ability to compute the pitch (or depth) of the LEDs, the robot can determine the distance needed to travel.

Object Avoidance

The robots move either in unison or individually. With each robot having four DC motors, the motors are able to move around the room and in multiple directions. Equipped on the robot are three laser time-of-flight sensors. They are spaced apart by 30°. These sensors run continuously to detect potential collisions. If an object is within a certain distance of the robot, the robot will stop moving and recalibrate its position.

Communication

The robot and localization module communicate bi-directionally over bluetooth, allowing a controller-peripheral hierarchy to be used between the two. The localization module sends commands to the robot, requesting a specific LED to activate or a certain motion to be executed. These messages use a custom byte encoding, where the first byte indicates the type of the message. The following bytes are the main data, specifying which LED to trigger or the rotation, speed and duration of the motion to execute. Finally, a checksum is utilized to confirm the accuracy of the message.

The robot makes the requested changes and then sends a confirmation message to the localization module, preventing synchronization issues between the two systems. Synchronization is essential, as the main localization loop is reliant on LED changes consistently happening between camera captures, and completely falls apart if the two systems get out of phase. As a result minimizing communication latency is of the utmost priority, and is anticipated to be the primary bottleneck on localization loop frequency.

Conceptual Block Diagram

A conceptual block diagram is a visual representation used to explain relationships between different systems and devices. A collection of subsystems is shown together which interact with each other. FireFly's conceptual block diagram is shown below in *figure 6*. It includes all the functional relationships between the robot and localization module, as well as components on the robot itself.

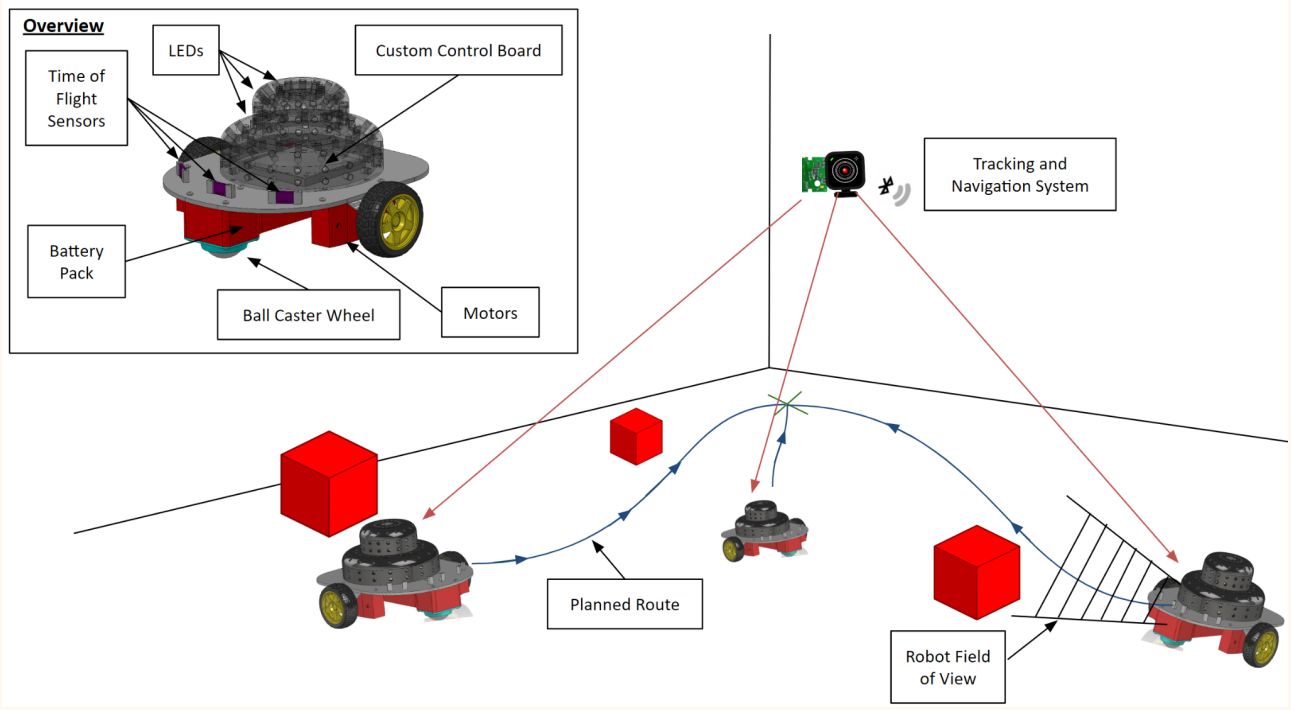


Figure 6. FireFly Conceptual Block Diagram

This diagram shows the process of a three robot team moving toward the direction of green LEDs. The positioning and trajectory calculation unit can use a microcontroller or computer, connected with a camera encompasses what is the localization module. This is the brain of the entire system. The module could use a Raspberry Pi to analyze and compute data given by the camera and determine the positions of the robots and fixed LEDs in a separate location.

Once the data is accumulated by the Raspberry Pi, the information is transferred to the robot. The robot has a custom designed circuit board with the ESP32-S2-MINI-2-N4R2 chip with an onboard trace antenna. Connected to this are the LEDs, time of flight sensors, and two H-Bridge controllers. As the robot moves, it collects separate data from the time of flight sensors. These allow the robot to avoid obstacles in its path if present.

As shown on the conceptual block diagram, every component present is on every robot regardless of modification. These subsystems are responsible for the essential functions including obstacle avoidance, location tracking and positioning, and system-wide communication.

Performance Specifications

Movement

The robots will have two DC motors and a specially designed free wheel at the front, which will let the robot move in a unique and agile manner. During straight movement, both motors rotate in the forward direction, while the free wheel minimizes resistance. For turning, one motor rotates, allowing the robot to pivot on its axis with the aid of the free wheel, ensuring smooth and precise turns. Spinning is achieved by rotating the motors in opposite directions, enabling the robot to rotate around its center. Reverse movement is accomplished by rotating both motors in the opposite direction, with the free wheel facilitating smooth backward motion. When the motors come to a stop, the free wheel ensures stability and a controlled halt. The utilization of these TT DC motors, as depicted in *Figure 7*, in combination with the specially designed free wheel, empowers the robot vehicle with exceptional maneuverability in various directions, making it a highly efficient and agile system.

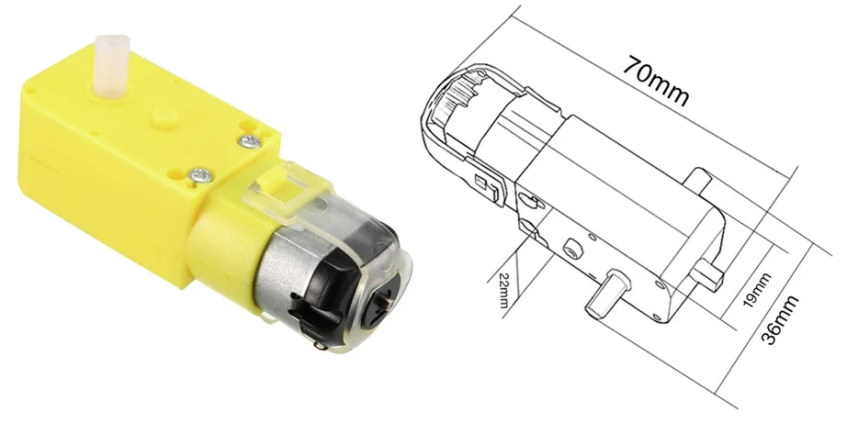


Figure 7. TT DC Motor

Sensors

The robots are strategically designed to be equipped with three time of flight sensors (TOF), each strategically positioned to provide maximum coverage. These time of flight sensors will serve as proximity sensors for the front and front-lateral sides of the robot, playing a critical role in anti-collision and obstacle avoidance capabilities. The system will use time of flight sensors, known for their reliable performance. Each VL53L0X sensor on a 940nm GY-530 GY-VL53L0XV2 breakout board. These sensors communicate via I2C bus to the MCU. They do possess the same I2c address but this is not an issue because they can be shut off and on with a GPIO pin to ensure correct data. All three sensors work in tandem to detect objects or obstacles using signal emissions, as illustrated in *Figure 8*.

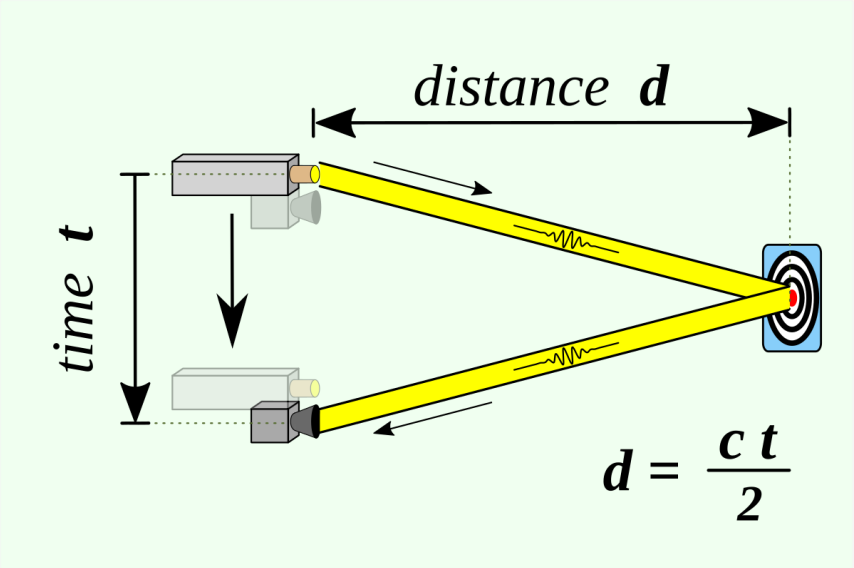


Figure 8. Transmitter and Receiver Signals

This economical sensor provides an accurate ranging distance of 2 meters with 1 mm error. It operates with a voltage of 2.8 V but has an internal voltage regulator. The sensor returns data very accurately in a straight line. *Figure 9* shows the 2 different board layouts for the chip but both can be used interchangeably.

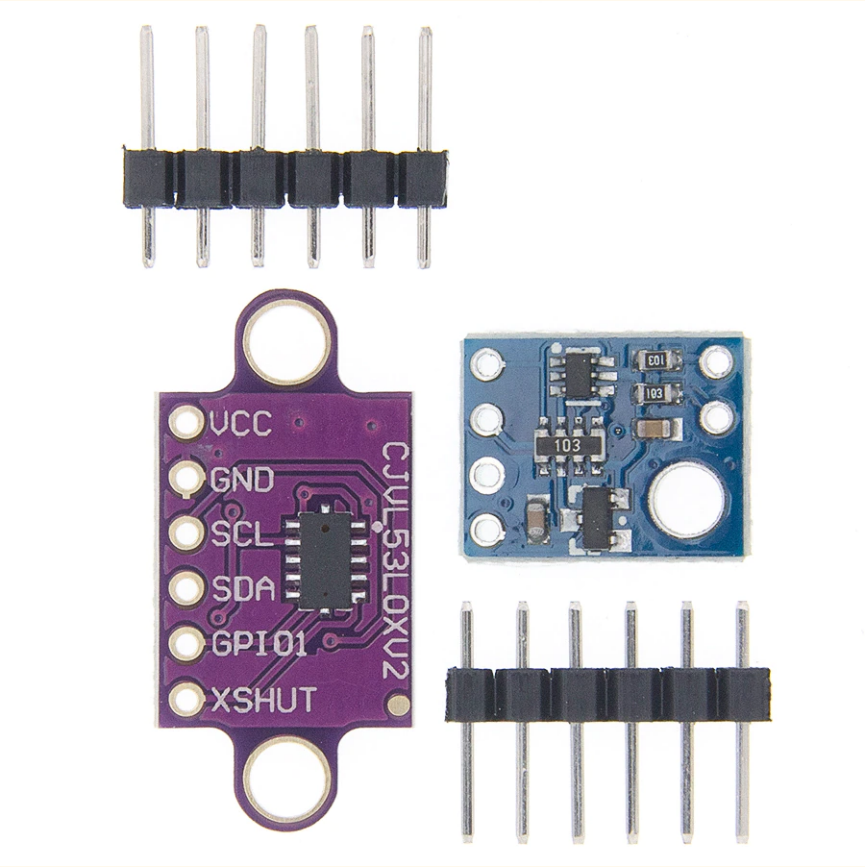


Figure 9. Specifications of the Sensor

The communication will be done using an 802.11 bluetooth connection using the ESP32-S2-MINI-2-N4R2 on board trace antenna. To accomplish this the PCB will incorporate a ESP32-S2-MINI-2-N4R2 with the correct power and configuration set up to use bluetooth or wifi connections to give more variety in set up options for the consumer. The trace antenna is shown in *figure 10*.

Figure 10. Trace Antenna

To accomplish automated movement this communication is key because the localization module will decide where the robots are and where they need to go and send this information to each robot. The robot will use this information in conjunction with its own sensors to create a path to follow and then execute that path. The use of bluetooth is to make the system more accessible to the populace while sacrificing some performance qualities like further range and faster speeds. The system could be modified to use different communication types for varied applications. The Esp line of chips supports a number of different communication options. For the scope of this project the 28 meter range will be adequate while keeping the project accessible.

Functional Block Diagram

Overview

A function block diagram is a tool used often with a conceptual block diagram. These diagrams show the inputs and outputs of each subsystem and go into specific low and high level metrics of the entire system. *Figure 11* shows the overall functional block diagram of FireFly's team robotics.

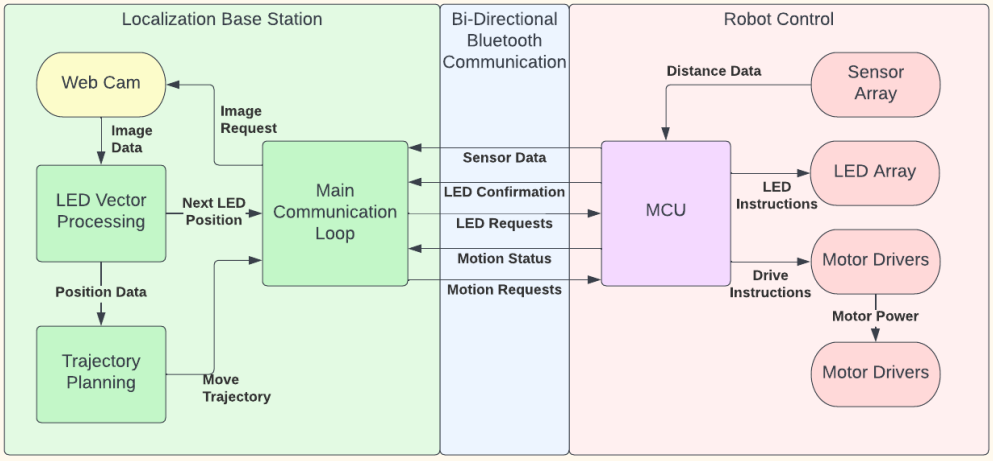


Figure 11. Functional Block Diagram Overview

There are three sections of the diagram, the localization base station, bi-directional bluetooth communication, and robot control. Starting from the left side of the diagram, the localization base station begins at the webcam. When an initial image is given, it is processed into LED vectorization to determine the position. A path is then computed for the robots to eventually take. This process is looped through the main communication loop where through bi-directional communication, information is sent and received by the robot. The localization system receives sensor data (TOF sensors), LED confirmation, and motion status. It then transmits LED requests and motion requests to the robot. When the microcontroller on the robot receives the data given to it, inputs a voltage to the motor drives to move the robot. Also received is the instructions for the LED array on each robot. This means the LEDs will flash at some frequency to stay within the frame rate of the

webcam. For the TOF sensors, they always remain on and always send distance data to the microcontroller which can be received by the localization system.

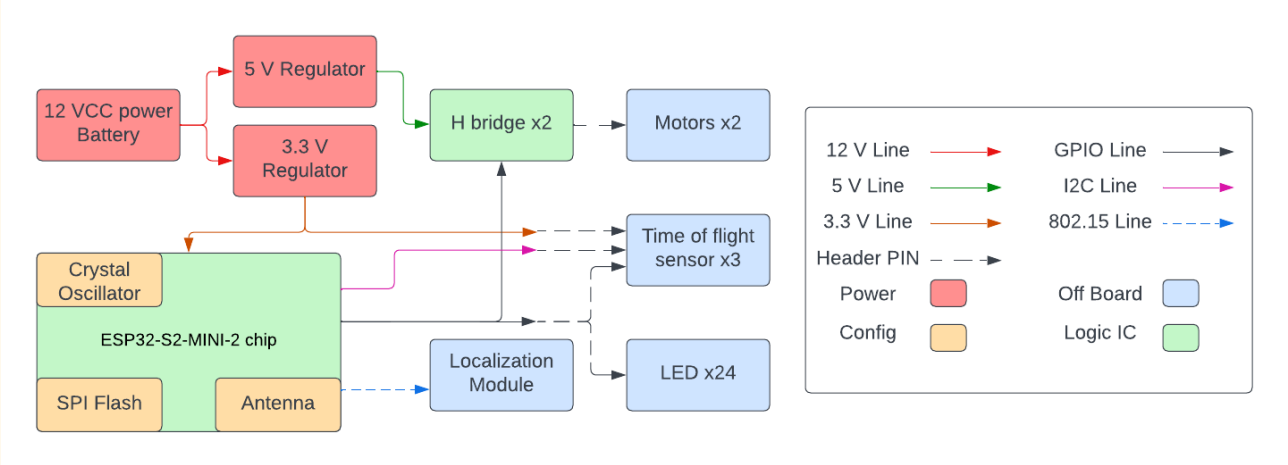


Figure 12. PCB Functional Block Diagram Overview

Above in *figure 12* shows the functional block diagram for the printed PCB. This shows that the system is powered by both a 3.3V and 5V power rails. The H bridge power supply takes from the 5V line while everything else runs on the 3.3V line. The MCU has most of the configuration components built in so that does not have to be considered in the design. It connects to the localization module with the onboard trace antenna. The rest of the peripherals connect to the board through header pins.

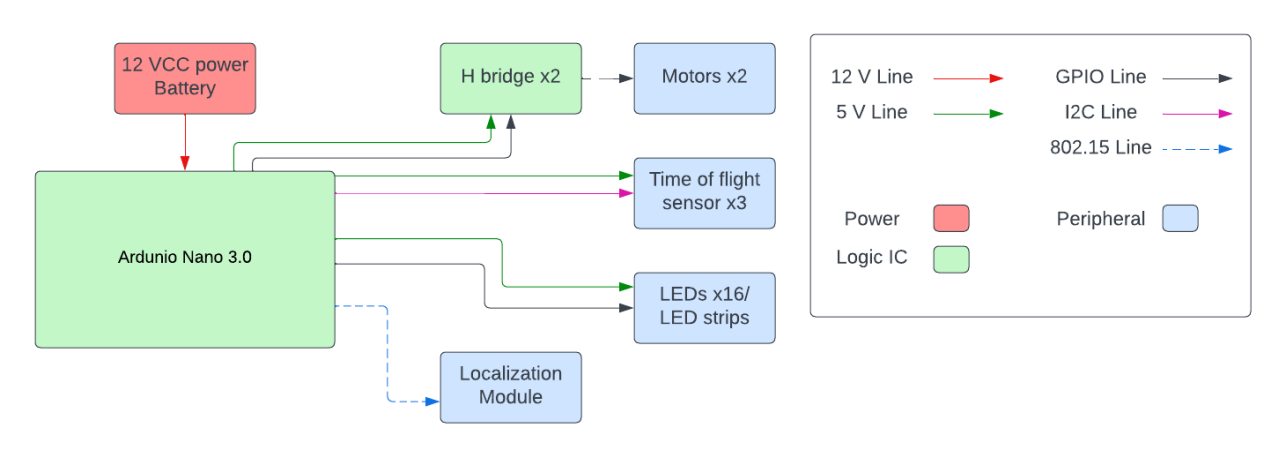


Figure 13. Arduino Config A/B Functional Block Diagram Overview

Below, *Figure 14* shows the functional block diagram for the Arduino hardware setup. This has two separate configurations labeled A and B in the electrical design section later. This uses an Arduino for the main MCU and the whole system runs on a 5V rail while the Ardiuno has an internal voltage regulator intake 12V from the battery. This setup uses mostly the same components as the PCB but with direct wire connections. They connect to the localization module using a bluetooth adapter, and in the configuration A, it uses a multiplexor to directly control the array of LEDs. TOF sensors and H bridges are connected directly to the microcontroller.

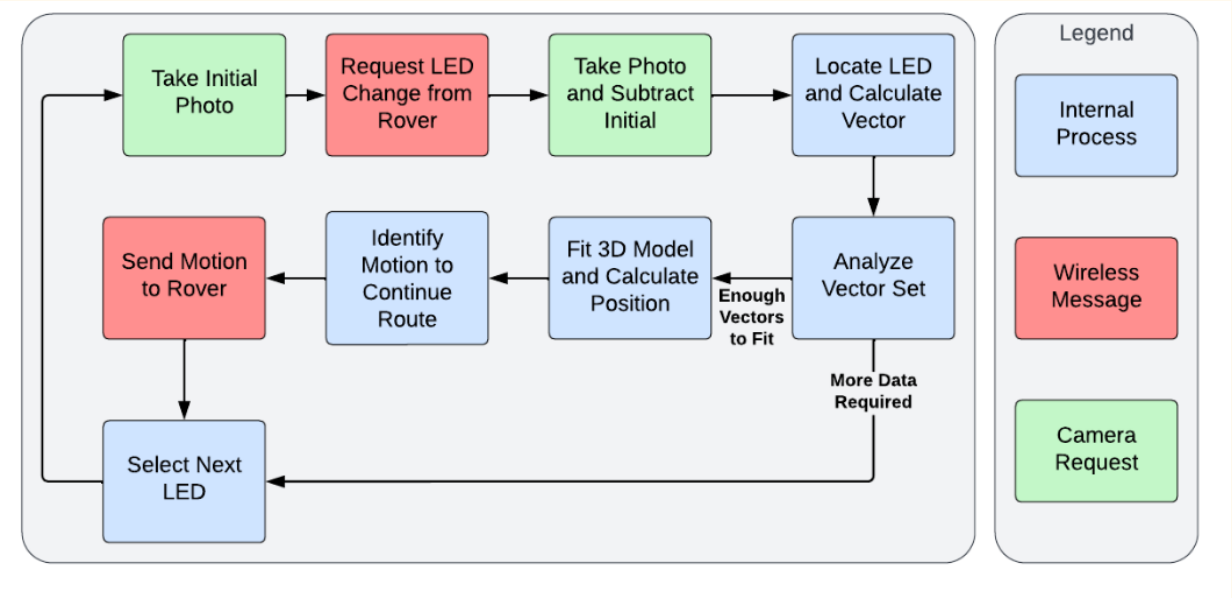


Figure 14. Program Functional Block Diagram Overview

Software Design

This appears to be a novel method of rover localization. Although some existing systems identify LED position using a camera, the LED is used to calculate the relative bearing of the light, not absolute position. The critical difference is the Lantern, a rigid 3D structure which holds the LEDs in place. When combined with closed loop control of each LED, each bright spot can be matched to a specific position on the rover. This allows position calculation with better precision and a larger operational area than was previously possible using existing hardware.

Data Intake

The data intake process activates each LED sequentially, taking a photo each time. These images are processed to calculate the precise angle of each light, relative to the camera. This information is then sent to the position calculation process over ZMQ.

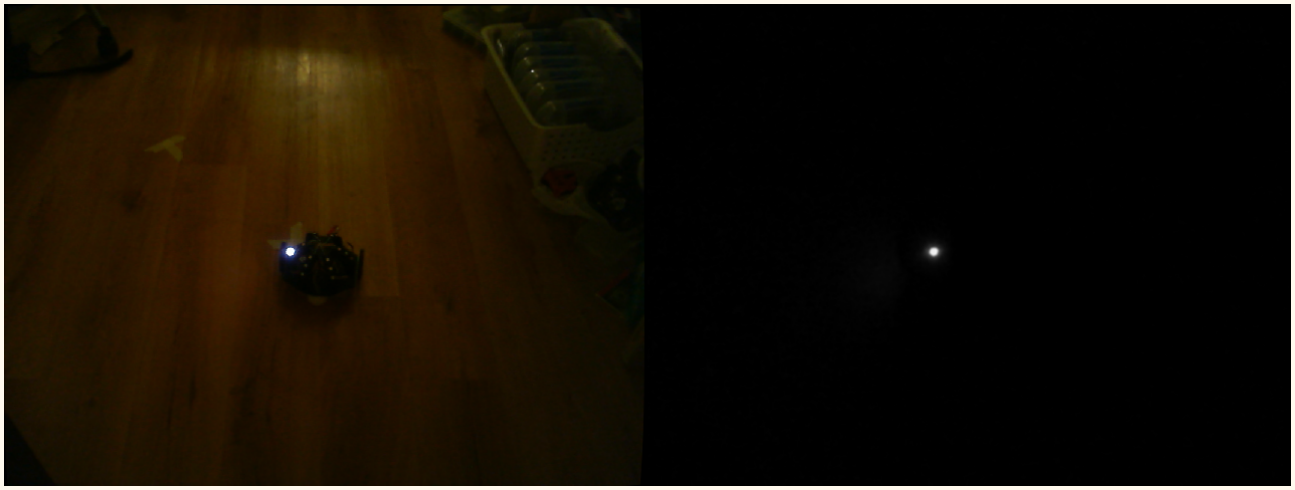


Figure 15. Initial Image Processing

The first step in the process is adjusting for lens distortion. The standard OpenCV2 camera calibration procedure is used to calibrate the camera before operation, saving its parameters to a YAML file. This allows each image to be undistorted as it is read, and also allows accurate calculation of vectors to each LED. The left display of *figure 15* is a webcam image which has been corrected.

The LED spot is also immediately obvious in this image. The poor contrast bandwidth of low cost webcams causes the LED to completely blow out the image, as any light sources in frame will produce a bright white dot. This is usually considered a drawback of low cost webcams, but hugely improves the quality of collected data, as any white objects will be comparatively dark.

Although the system can operate by simply checking for this bright spot in each image, precision can be improved by comparing the photos. Each image is converted to grayscale and has the previous grayscale image subtracted from it. This method detects an increase in brightness, rather than just any bright point. This eliminates false positives from other LEDs in the frame, which can hugely mess with the data. A subtracted image is also displayed in *figure 16*.

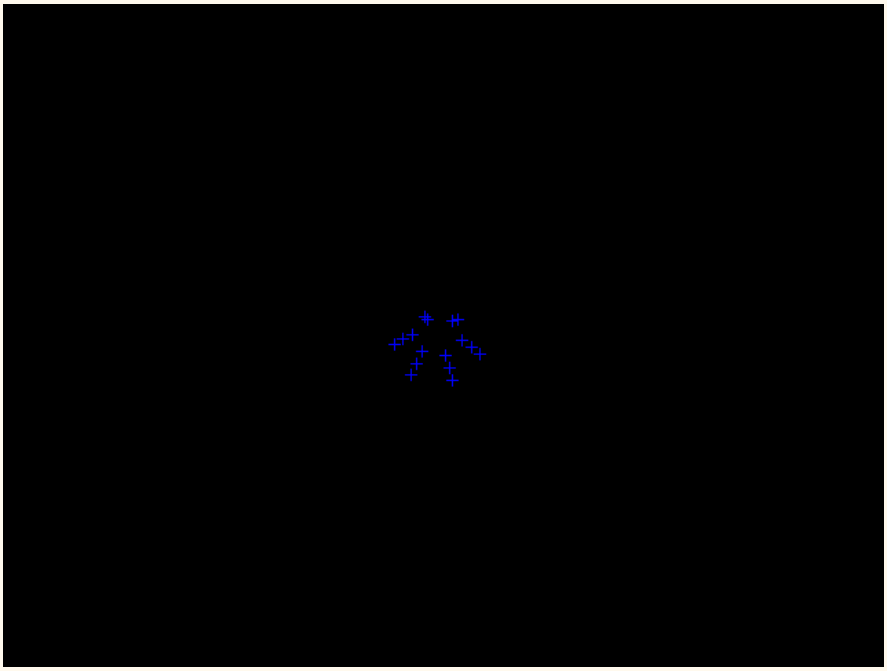


Figure 16. LED Positions in Image

Finally, the average of the bright spot is taken as the position of the LED. *Figure 16* shows a complete set of data points. The X and Y angles of the detected LED are calculated relative to the center of the image, using information from the camera

calibration process. These values, along with additional metadata, are saved for analysis and the process repeats until every LED has been tested.

There are several dynamic feedback elements that catch error and improve accuracy. The size of the bright spot is measured, and the brightness of the LEDs is adjusted in real time to attempt to stay within a specified size range. This helps the Lantern stand out in bright rooms, but prevents warping from excessive image blowout.

The next LED to measure is selected algorithmically. This the same LED to be run multiple times, as the brightness is adjusted. The algorithm also considers spacing, preventing overlapping bright spots between images. Reflections are a common issue, and each output image is checked for multiple bright spots to detect errors. This also provides resilience in case of interference from other light sources, as any external light turning on will register as a reflection. If a reflection is found, the data point is tossed out and the LED will be rerun later.

Position Calculation

The position calculation converts a given set of LED angles to a position and rotation in 3D. It achieves this by fitting a 3D model of the Lantern to the given angles. This process runs in a separate thread, and communicates with the main process using ZMQ. To improve performance, the localization math is written in C++, and called from Python using SWIG.

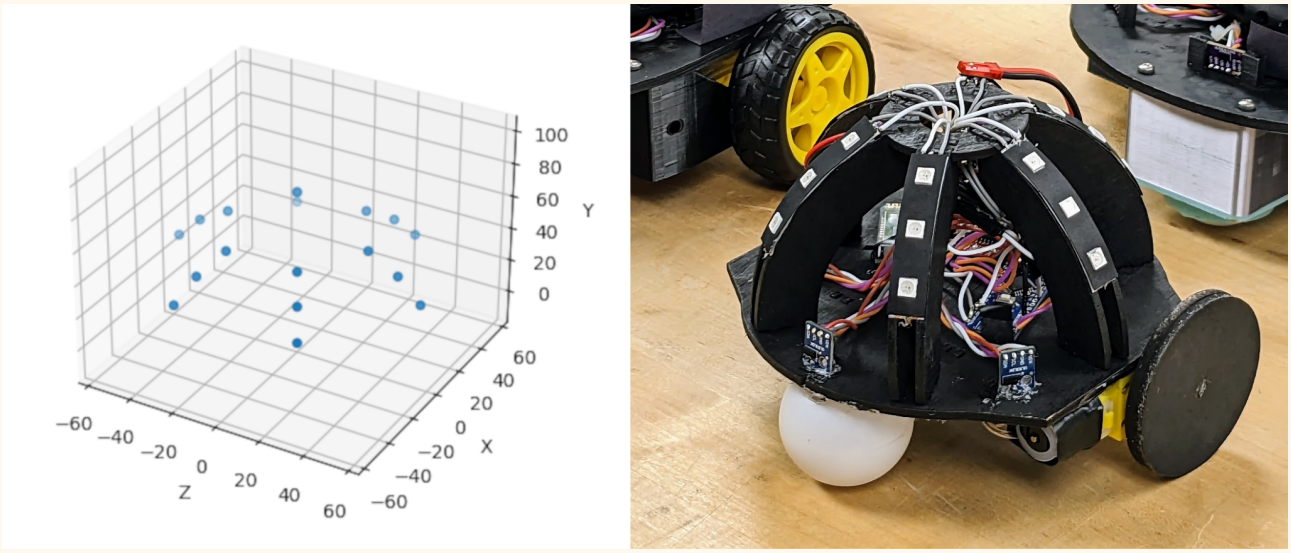


Figure 17. Lantern Comparison

Each Lantern model has a dataset containing the position of each LED on the Lantern. These are calculated from the CAD model lantern, and saved as XYZ positions in millimeters. *Figure 17* compares the saved position data to an actual rover.

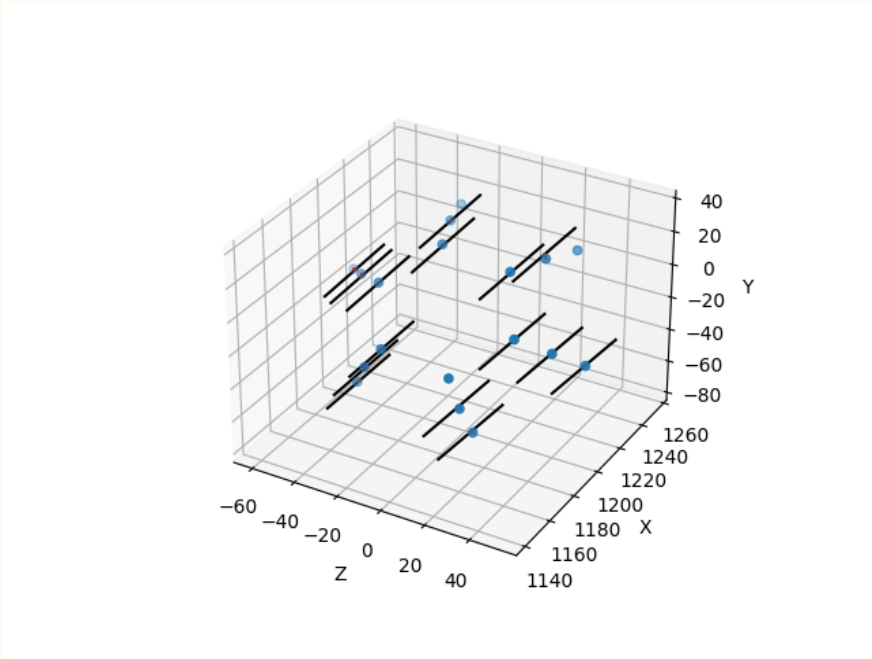


Figure 18. Rover Position Convergence Example

The actual position calibration is achieved with an evolutionary optimization algorithm. Using the angles from the intake process, a line is calculated for each LED. The camera position is set as the origin to simplify the math, making all LED lines meet at (0,0,0). At this stage, for each LED, there is a known 3D line that passes through it in 3D space and a known position relative to each other LED. By finding a position with minimal distance between each LED and its corresponding line, the system converges on the accurate position of the Lantern.

The actual calibration is achieved by transforming the Lantern to a new position, calculating the squared distance from each point to each line, and saving any changes that reduce this value. This process is iterated, testing variations of the current best transformation and reducing the degree of variation over time. This consistently converges on the correct position, with a squared error of <1 mm per LED. *Figure 18* visualizes this convergence, where the blue shows the calculated positions of each LED and the black is the line being fit to. There are also red lines which represent the distance between the two, but are barely visible due to the accuracy of the system.

Navigation

For demonstration purposes, the rover was programmed to loop through a set of waypoints. Future users can set arbitrary waypoints themselves depending on their application. The loop waypoints are established running the calibration script. The rover is moved by hand to each target point, localizes, and saves this position as a waypoint. The rover then begins to cycle through these positions.

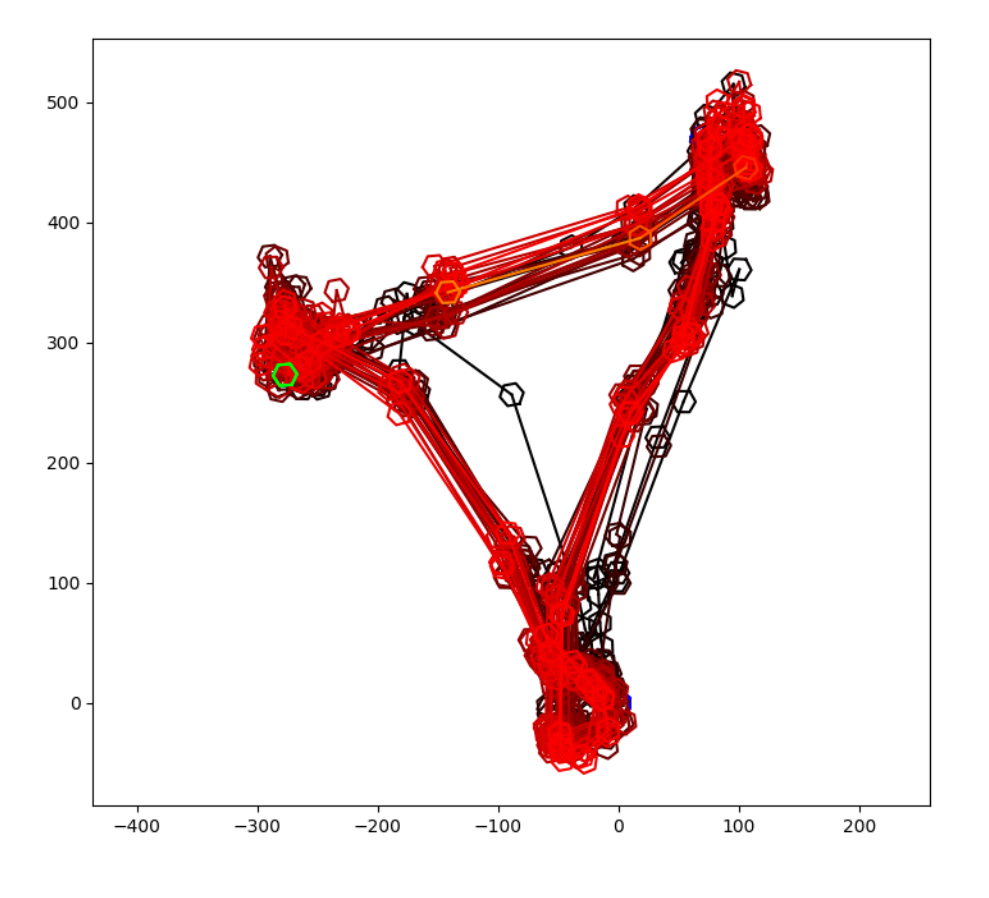


Figure 19. Rover Positions during Waypoint Mission

The navigation algorithm reduces the location data to 2D, using only the XY position and Z rotation relative to the first waypoint. The current position is compared to the target position, and the change in rotation and distance are used to calculate the required motion. *Figure 19* shows the results from a three hour repeatability test, in which the rover repeatedly moved through three specified waypoints. Each hexagon is a point on the floor where localization was completed, with a line connecting it to the previous position. The colors range from black to orange depending on recency, and the green hexagon represents the current target position. The system did not require any intervention for the duration of this run. The only deviation, the path which cuts through the center, is from the rover not being at a waypoint when it started the route.

Machine Learning

Calculating the actual corrective actions proved to be quite challenging. To minimize cost, the rovers do not have sensored motors, and thus do not have information about their motion. As they use a tank drive system, each localization step involves turning to face the waypoint and driving forward, ideally stopping on the waypoint. However, the lack of feedback means that the motion control only has three inputs: which direction to turn, how long to turn, and how long to drive. Early versions relied on simple, hard coded equations to calculate these values from the desired turn angle and drive distance.

There is a wide range of factors which impact how the rover moves, such as battery voltage, floor friction, and motor wear. To adjust for this, a simple machine learning algorithm was implemented. Initially, the rover moves using an intentionally undertuned equation, causing it to move inefficiently but safely. For each motion, the start point and end point are recorded, as well as the motion durations used. When a threshold number of data points is reached, the steering algorithm is replaced by a line of best fit through its recent motion history.

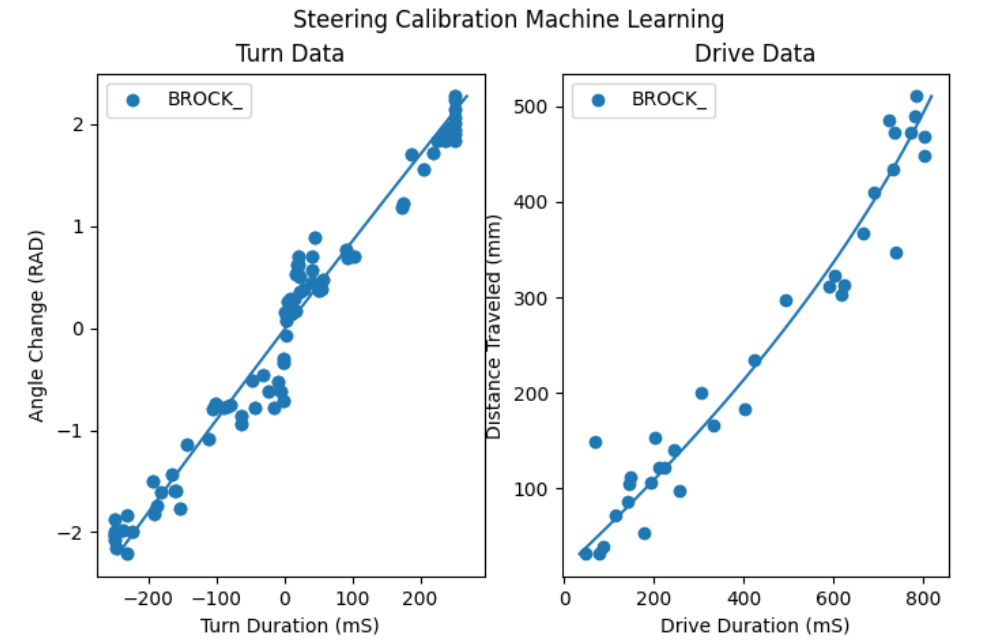


Figure 20. Steering Calibration using Machine Learning

These relationships are plotted in *figure 20*. The turn duration is calculated from the angle change, and the drive duration is calculated from the distance to the waypoint. Right turns are recorded as negative values. Both systems use a two degree polynomial function. Despite its simplicity, this system massively improved rover performance, and they noticeably improved in just 15 minutes of learning. This information is also saved between runs, so the initial calibration only runs once per rover.

Mechanical Design

Designing and manufacturing of the robots requires meticulous attention to detail and thoughtful consideration of each component's design and functionality. By leveraging detailed engineering drawings, 3D printing technology, innovative design elements, and color selection, anyone can produce these robots. Overall, attention to these details in the manufacturing process results in a robot that is affordable, expendable, functional and visually appealing, and easy to assemble.

Hardware

A universal fastener is used throughout the robot. What this means is a single screw type can be used for every fastened component. The type of screw is an important first step to designing the robot as this would minimize the material and overall size of the robot and its components.

Because the robot is 3D printed, the right hardware needed to be selected. From McMaster-Carr, heat-set inserts for plastics were chosen as a majority of the robot was FDM printed. The heat-set inserts selected were brass M3x0.5 at 5.7 mm long.

After selecting the heat-set inserts, the screws were next. Since M3x0.5 inserts were selected, M3x0.5 screws were needed. Also chosen from McMaster-Carr, 18-8 stainless steel M3x0.5 at 8 mm long screws were selected. 18-8 stainless steel is the selected material due to its low cost. The screw head is a rounded pan top with a Phillips type drive style. An 8 mm long screw was picked because the chassis has a thickness of 4 mm which goes into a 5.7 mm long insert. Then in order to have a minimum thread engagement of 3 mm, the screw needed to be a minimum of 7 mm. Therefore the range of the screw must be between 7 to 9 mm long.

Lantern

The LED fixture located at the top of the robot, as shown in *Appendix 1*, is aptly named the "lantern." Its unique design draws inspiration from the organic pattern of firefly lanterns, lending an element of originality to the robot's appearance while effectively transmitting the light emitted by the LEDs. Moreover, the lantern's design permits it to be twisted into the chassis, thereby creating a stable and secure connection. To ensure optimal LED reflection for localization purposes, the lantern requires specific color selection, namely dark and matte colors.

The lantern is engineered to be twisted to the base, hence featuring four extrusions at the bottom which has a radius of 65 mm from the center point of the line to the origin, each measuring

60 mm long and 5.01 mm tall. These extrusions follow the radius of the last row of LEDs. The lantern comprises four levels of rows, with each row's outside diameter reducing from bottom to top. The numerous holes in each row offer multiple localization options, allowing for experimentation to determine the optimal setup for the system. Additionally, the lantern's relatively spacious interior was planned to accommodate electronics, imparting a professional and neat look to the overall design.

Beginning from the bottom to the top, the first row of the lantern has an inside diameter of 115 mm, an outside diameter of 160 mm, and 43 holes. The second row features an inside diameter of 115 mm, an outside diameter of 145 mm, and 38 holes. The third row has an inside diameter of 85 mm, an outside diameter of 130 mm, and 33 holes, while the fourth and final row boasts an inside diameter of 70 mm, an outside diameter of 115 mm, and 28 holes. The thickness of each row is 7.5 mm. All holes in the lantern are identical, with a through hole of 5.5 mm in diameter and a slot of 6 mm diameter with a depth of 17.13 mm from the inside of the lantern.

Chassis

The chassis developed shown in *Appendix 2* comprises a 4 mm thick base that serves as a solid foundation for the diverse components of the robot. The chassis is 229.25 mm long and 193.31 mm wide which includes 2 filets in the back with a radius of 50 mm and two at the front with a radius of 90 mm.

The center of the chassis features a square extrusion measuring 80x80 mm. While originally intended for the PCB, it was subsequently repurposed for other electronics as per Configuration B of the Electrical Design section. The said extrusion is elevated by 6 mm and has a small 2 mm rail, which incorporates a 20 mm opening on the upper left corner for the ease of access of the power cables. Additionally, it contains one hole with a diameter of 4.8 mm that goes through the base for the battery holder and three more with a depth of 6 mm for the inserts and screws.

Four 55.94-degrees angled pairs of extrusions originating from the center, with an internal radius of 65 mm and a 10 mm distance between each other. Their function is to twist-fit the connectors for the lantern being 5 mm tall. Positioned at the front, three 11.5 mm tall extrusions serve as sensor holders, placed at 35-degree intervals, covering left, right, and forward directions effectively. These sensor holders' design facilitates sliding the sensors from the top and firmly securing them in place. The extrusions' length is 30 mm, rendering them capable of accommodating the sensors in use.

The rear features a square-shaped cut measuring 26 x 18 mm with a radius of 50 mm filets. This is explicitly designed to hold the voltmeter display, providing users with an easy way to check the robot's battery life. The chassis's sides have two holes each for motor holders, placed 77.5 mm apart. and two front holes hold the battery holder. The 4.8 mm diameter through holes accommodates screws used throughout the chassis.

The thoughtful design of this chassis optimizes the robot's performance and functionality. The sturdy base provides a robust foundation for all the components, while the elevated square extrusion streamlines wiring and accommodates multiple components. The angle extrusions provide a convenient and secure way to attach the lantern, while the strategically placed sensor holders ensure maximum coverage. The voltmeter display cutout provides an effective means of monitoring the robot's battery life, and the holes for motor and battery holders are perfectly sized for inserts and screws. Overall, the design of this chassis is an efficient and effective way to house the robot's various components, resulting in optimal performance and functionality.

Component Holder Parts

A voltmeter holder allows users to monitor the electrical components of the robot, such as the battery life, and check if it requires charging. The voltmeter holder is a single rectangular piece that extrudes upwards to create a box-like structure with two openings. *Appendix 4* shows the drawing of this component. There are three holes of 6 mm depth which are used for heat inserts which help it tightly attach to the chassis with screws.

The motor holders consist of two boxes measuring 70.5x21.24x32.5 mm that are designed to securely house the TT DC motors. A hole with a diameter of 4.5 mm and a depth of 5.8 mm at both the front and back ends are made and is centered between a wall width of 8.74 mm. The side walls, on the other hand, have a thickness of 3 mm and are characterized by a 90-degree filet from a specific position. Additionally, the extrusion cut in the side of the motor features a check shape with a curvature of a radius of 8.75 mm, a depth of 32.24 mm, and a small hole cut with a radius of 2.25 mm. It is important to note that the hole in the middle of the check shape cut is necessary to allow for certain aspects of the motor to protrude. Finally, this part includes a through hole measuring 5 mm in diameter on either side, primarily for wiring purposes.

The motor spacer is an "X" shaped extrusion with extrusions at the top and bottom to hold the motor in place. The spacer measures 69.95 mm in length, 21.35 mm in width, and 10.50 mm in height. The inside angles that form the "X" are 83.25 degrees, and the outside angles are 6.75 degrees. The purpose of this part is to maintain motor stability in the event that the robot

encounters rough terrain. The spacer acts as a spring and damper, reducing any unexpected vibrations. Two motor spacers are utilized for each motor.

The battery/ball holder is a multifunctional part. Firstly, it serves as a battery holder, measuring 106x34x35 mm. The side thickness of this part is 3 mm, while the back is 7 mm thick (to accommodate for the heat-set insert diameter), with a 4.5 mm diameter hole and 5.8 mm depth for a heat-set insert. The placement for the battery is a rectangular through cut of 10.3x20 mm on the rear side to facilitate wire placement from the battery. This part also functions as a ball holder, measuring 55x48.5x35 mm. The front corners have a filet of 5 mm radius, and the back corners have a filet of 10 mm radius. The top of the box has three holes, the two holes at the front are through holes, and the other is 4.5 mm in diameter and 6 mm in depth. At the bottom, there are four holes in each corner. Two holes are through holes as just mentioned. The other two holes are spaced 37.9 mm apart from the through holes. Additionally, the bottom features a 21 mm radius circle cut at the center of the box to hold the front ball in place.

The ball caster component serves a crucial role in the robot's overall functionality. It features a simple yet effective design that holds the ping pong ball securely in place. The base of the caster measures 55x 49.05 mm and features filet cuts in each corner with a radius of 5 mm. The corners also have 4.5 mm diameter holes, providing ample space for connecting screws. A revolve cut sphere of radius 20.5 mm is offset by -0.2 mm from the origin is present to ensure there ball freely spins (given this tolerance measure). The bottom portion is also revolved cut at 105.34 degrees to reduce the amount of material, giving us a thickness of 5.01 mm. Overall, the ball caster component serves an essential role in ensuring the robot moves smoothly and efficiently, and its simple design makes it easy to integrate into the overall structure.

3D Printing

Every component on the robot was 3D printed (excluding the motors and hardware). The printers used for manufacturing were the FDM printers, Creality Ender 3 and Anycubic Kobra Plus, with a print size capability of 200x200x250 mm and 300x300x350 mm respectively. The SLA printer used for manufacturing was an Elegoo Mar 3 Pro with a print size capability of 173x89.6x175 mm. UltiMaker Cura 5.3.0 was the slicer used for FDM and Chitubox was the slicer used for SLA printing. The material selected was PLA at 1.75 mm diameter for the FDM process and standard resin for the SLA printed parts. Only two components require the need for a specific color, this is the chassis and the lantern. They are required to be printed in a matte black PLA filament. As requested by the firmware design to reduce light reflection during the localization process. 3D printing was the selected manufacturing process because different test iterations could be made without worry

about unfixable errors. In the long run, if a swarm of robots is constructed, injection molding would be the most cost effective manufacturing process.

The chassis was printed on the Anycubic Kobra Plus due to its large size. It was oriented with the extruded features facing upward as shown in *figure 21* below. The printing parameters were set as follows: layer height of 0.1mm with an initial layer height of 0.3 mm and line width of 0.4 mm. The walls have a thickness of 0.8 mm however can be increased to increase structure support. The same can be said about the top/bottom thickness which is set to 0.8 mm. The printing temperature is set to 210°C with an initial printing temperature of 200°C and the build plate temperature is set to 60°C. The infill density is at 12% with an infill pattern of *Tri-Hexagon*. This pattern is selected because the chassis is not subjected to any high stress. Therefore this pattern can be used as it is for low stress parts but at the same time still has a strong enough structure compared to material usage. The print speed can be set to 150 mm/s, providing a print time of under 9 hours. Note that if an Ender 3 is used to print this, the print speed should not be higher than 120 mm/s. Travel and cooling are set at the default setting and there are no supports generated for this build. A skirt is used as the build plate adhesion type, but note that any adhesion type could be used. Lastly, the experimental section has a slicing tolerance of *Middle* and *Infill Travel Optimization* is checked.

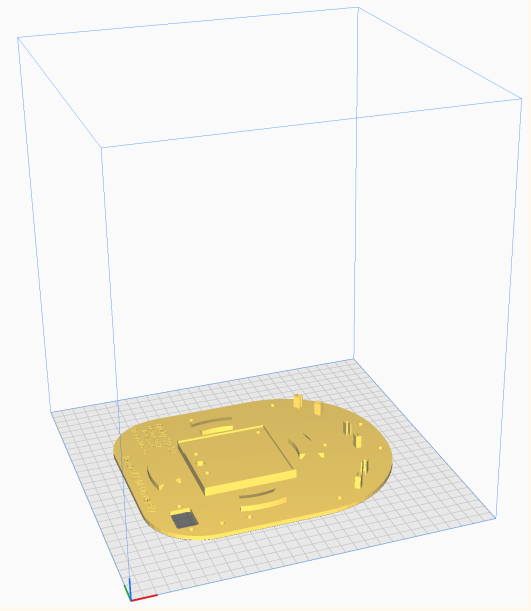


Figure 21. Chassis Printing Orientation

Similar to the chassis printing parameters, the lantern's printing parameters are the same for the layer height, line width, wall thickness, top/bottom thickness, printing temperature, build plate temperature, travel and cooling. The infill density is set to 8%, which is the lowest percentage with the minimum material usage while keeping the lantern rigid. The infill pattern is *Grid*, which allows it to print 25% faster as compared to *Line*. If printed on the Anycubic Kobra Plus, the maximum print speed is 120 mm/s, while on the Ender 3 the maximum print speed is 90 mm/s without the part misaligning. To keep the exterior of the part smooth, the lantern is printed as shown in *figure 22*. However because of this orientation, the part needs to have generated support with a support structure of *Normal* and support placement of *Touching Buildplate*. The support pattern is set to *Zig Zag* and has a density of 15%. Regardless of the printing, the support interface is enabled with a density of 33.333% and pattern of *Grid*. The selected build plate adhesion type is skirt, but similar as the chassis parameters, any type can be selected.

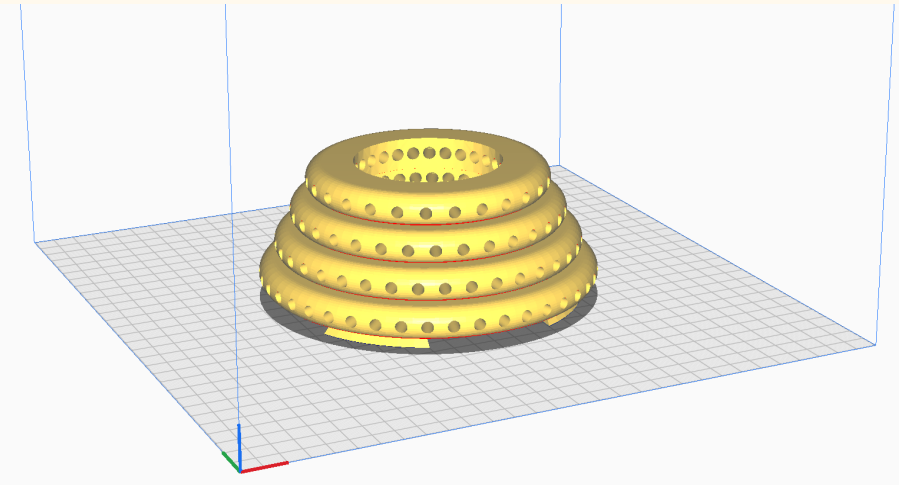


Figure 22. Lantern Print Orientation

The printing parameters for the voltmeter holder and the motor holders are the same. They are printed with the same initial parameters as the lantern and chassis. *Figure 23* and *figure 24* show the orientation of the parts. The print speed for the voltmeter holder can be set to 150 mm/s on the Anycubic while the Ender 3 print speed can be set to 120 mm/s. The print speed for the motor holders can be set to 100 mm/s and 70 mm/s for the Anycubic and Ender 3 respectively. Both components are printed with an 8% infill and a *Grid* infill pattern. No supports are needed for these parts. In the travel section, *Avoid Printed Parts When Traveling* is enabled.

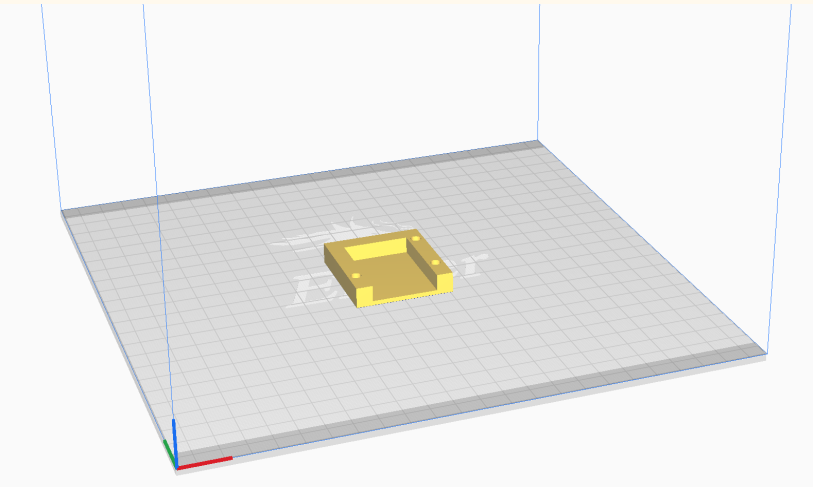


Figure 23. Voltmeter Holder Print Orientation

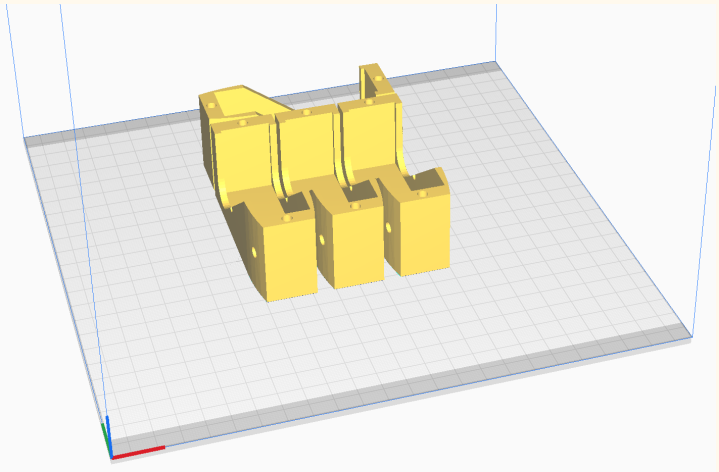


Figure 24. Motor Holder Print Orientation x4

The last FDM printed part is the battery holder. Again the parameters are the same for the layer height, line width, wall thickness, top/bottom thickness, printing temperature, build plate temperature, travel and cooling. The part is orientation with the ball wheel holder facing upward as shown in *figure 25* This is important as the surface roughness must be smooth for the ball to freely turn without any issues. To ensure the roughness is smooth, the print speed on the Anycubic is set to 80 mm/s and on the Ender 3 it can be set to 60 mm/s. The infill density is set to 10% with a *Grid* infill pattern. While supports are not required, they can be enabled to print this. It is recommended to use tree supports, however *Zig Zag* supports only touching the buildplate can also be used.

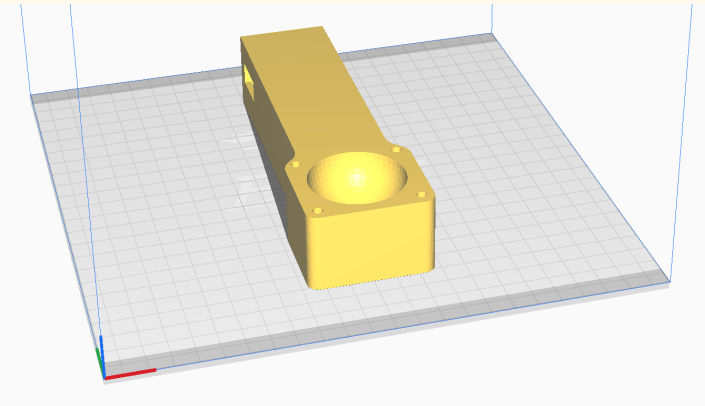


Figure 25. Battery Holder Print Orientation

The remaining components are SLA printed using a Mars 3 pro. *Figure 26* and *figure 27* show the printing orientation of the ball caster holder and the motor spacers. The ball caster holder is printed with support on the exterior side of the part. The reason for this is to keep the interior section smooth and as little surface roughness as possible. The printing parameters for the ball caster are: an exposure time of 2.5 seconds, bottom exposure time of 30 seconds, lift distance is 5 mm, lift speed of 80 mm/min and a retract speed of 210 mm/min. The motor spacer has the same printing parameters with the only difference being the lift distance is set to 3 mm.

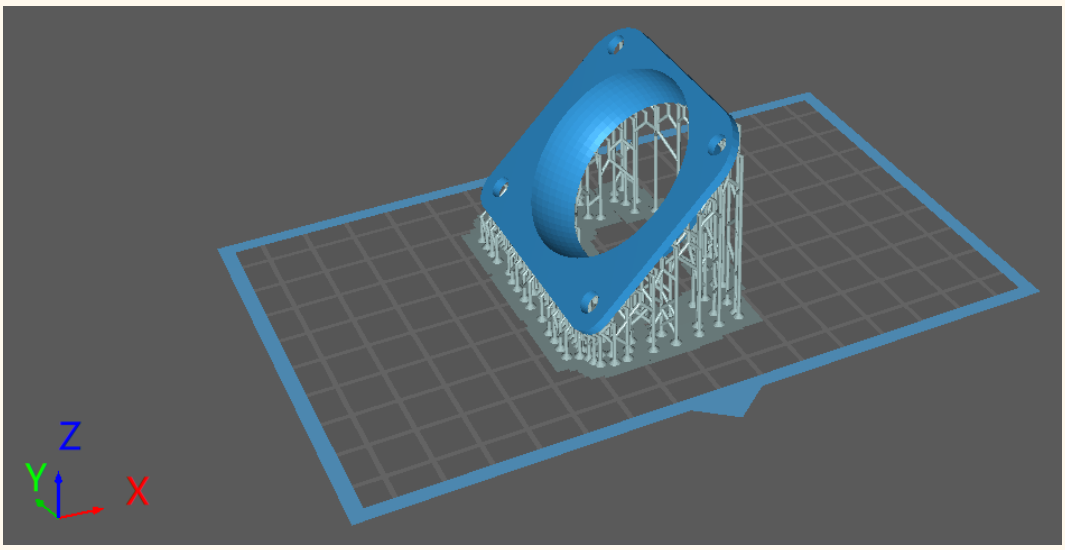


Figure 26. Ball Caster Holder Print Orientation

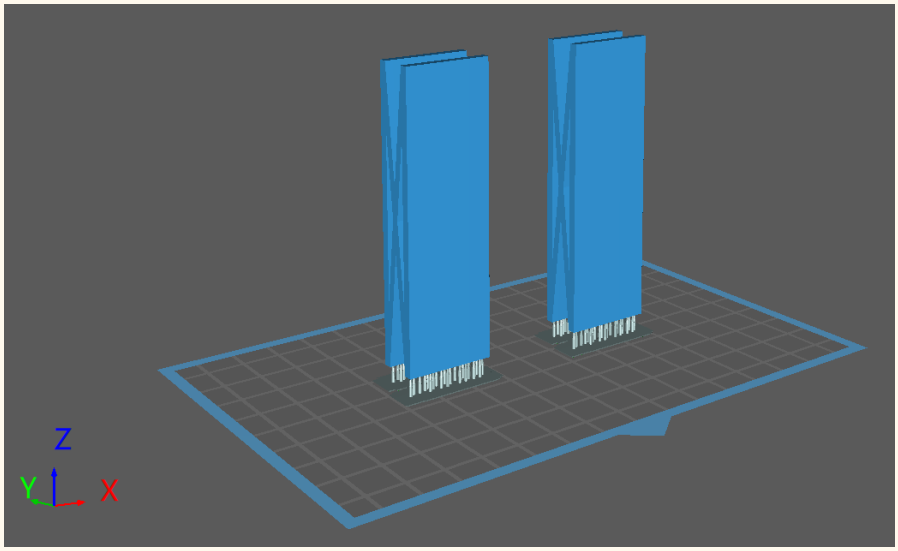


Figure 27. Motor Spacer Print Orientation

Full Assembly

The robotic assembly featured in this project is a cost-effective and user-friendly kit, consisting of all the necessary components for a FireFly robot. The package includes a lantern, chassis, voltmeter holder, 2 motor holders, motor retainers/spacers, battery holder, ball wheel holder, pin pon ball, wheels, and fasteners.

The procedure is relatively straightforward, involving the use of friction fit, twisting, and tightening screws. The start of the raw assembly begins at pressing in heat-set inserts into the previously mentioned parts.

First the heat-set inserts are placed into the 3D printed parts using a soldering iron. Then the battery and ball caster holder(with the ping pong ball) are attached to the battery holder. The ball caster holder uses 4 screws to fasten them together. Then the battery holder can be fastened to the chassis. Next the motors and motor spacers are placed in the motor holders with the motor going in first then the spacers. Those can then be fastened to the chassis. The voltmeter holder can then be attached with the voltmeter. Finally the lantern can be twist-fitted onto the chassis. In case of reflections against the lantern or chassis, small pieces of black construction paper can be folded and placed underneath the LEDs.

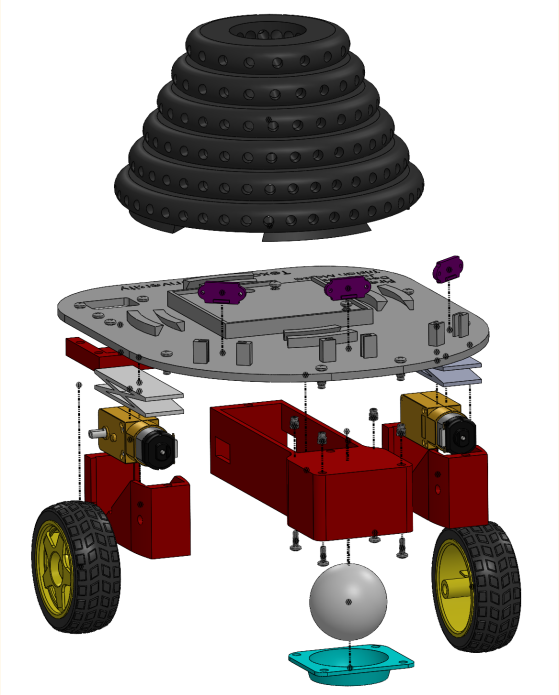


Figure 28. Robot Assembly

Electrical Design

This project had 3 different hardware configurations that have advantages and disadvantages across the board. Different designs were created to increase overall accessibility and to test a variety of setups due to the complexity of the project. The systems that we created can be almost indefinitely optimized, so no one design is correct.

Option 1: Config PCB

A PCB design was created in Altium. The list of parts used can be seen below in Table 1. Each component is arranged in the circuit according to their data sheet or user manual to ensure the components do not break. Multiple decoupling capacitors, inductors, and diodes are also added to the circuit to ensure protection from any power fluctuations.

Part number	Description	Designator	Footprint	LibRef
LD1117V33C	3.3 V Regulator	U7	To-220	CMP-0244-00316-1
L7805CDT-TR	5 V Regultor	U6	TO-252_N	CMP-0244-00216-1
ESP32-S2-MINI-2-N4R2	ESP32-S2-MINI-2 chip	U0	ESFS-QFN-32_L	CMP-2000-07453-1
DRV8838DSGT	H bridge	U1, U2	FP-DSG0008A-IPC_B	CMP-04915-000053-1
JST Connector	Male Header Pins	P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15	1x3, 1x5, 1x8	N/A
1N4148WS	Diode	D0	SODFL2512X100N	1N4148WS
MC0603B105K160CT	Capacitors 1uF	C5	CAPC1608X87N	MC0603B105K160CT
06036D106MAT2A	Capacitors 10uF	C3, C14	CAPC1608X90N	06036D106MAT2A
C1608X7R1H334K080AC	Capacitors .33uF	C13	CAPC1608X90N	C1608X7R1H334K080AC
MC0603B104J160CT	Capacitors 100nF	C1, C2, C4, C6, C7, C8, C9, C10, C11	CAPC1608X87N	MC0603B104J160CT
AC0603FR-0710KL	Resistors 10kohm	R1, R2, R4, R17, R18	RESC1608X55N	AC0603FR-0710KL
ERJ-H3G0R00V	Resistors 0 ohm - 603	R3, R7, R8, R9	RESC1608X55N	ERJH3G0R00V
RT0603FRE07100RL	Resistors 100 ohm	R5, R6, R10, R11, R12, R13, R14, R15, R16, R19, R20, R21, R22, R23, R24, R25, R26, R27, R28, R29, R30, R31, R32, R33	RESC1608X55N	ERJH3G0R00V
0ZCF0100AF2A	2A Fuse	F1	RES_ERJH3G0R00V	ERJH3G0R00V
TL3305AF260QG	E-Switch	Boot, Reset	Button	N/A

Table 1. PCB Parts

The main chip is the ESP32-S2-MINI-2-N4R2 which acts as the brain for the robot and all of its direct operations. All data will run through this chip and it is where the motion, object avoidance, communication, and TOF data collection will be taking place. The configuration requires a decoupling capacitor of 1 µF at the En port, 100 nF, and 10 µF at the 3V3. The board has an array of ground pins that are all connected to the ground to isolate the on-board antenna. It also has a do not connect pin at the NC port. To set up the I2C buses 10 kohm resistors are used to pull up the line. The UART pins are used for the programming pins while the EN pin is used to reset the device. There is a test point on the voltage input of the board. The board is powered by the 3.3 V and draws 500 mA in usage.

If someone were to recreate this project, the use of the ESP32-S2-MINI-2-N4R2 would be preferably switched out for a different chip. This is due to the relatively new release of the ESP32-S2-MINI-2-N4R2, we discovered multiple issues that were not well documented by the manufacturer. The main ones are packet loss on the I2C bus and broken internal coding libraries that should be used on the device. This section can be seen below in *figure 29*.

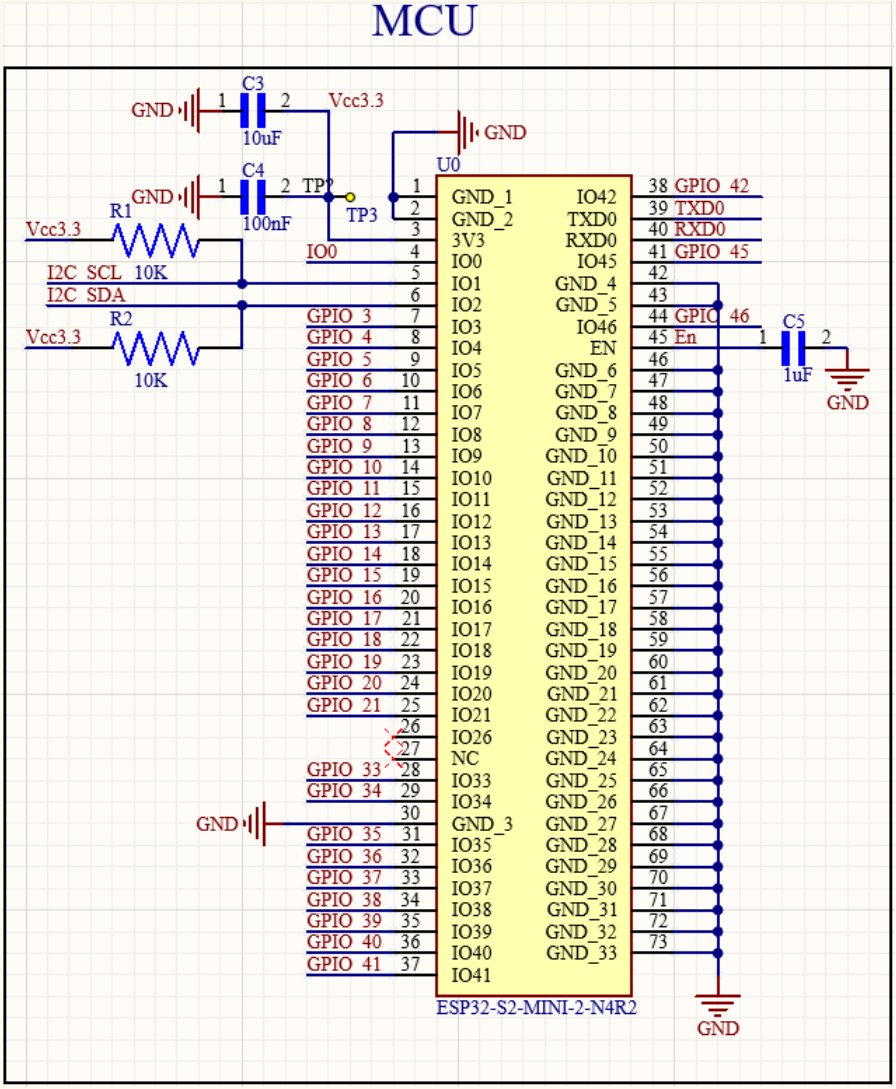


Figure 29. PCB Design MCU

The next section is the power input for the circuit. The PCB is powered by a 12 Battery and the battery leads soldiers to the board. It is protected by a 2 A resettable fuse.

If there is any short the fuse will disconnect the circuit and protect the parts. It has a diode for more protection and is isolated by the 0 ohm resistor. There are also four test points to be put on the edges of the board for testing and configuration purposes. The Altium design for this section can be seen below in *Figure 30*.

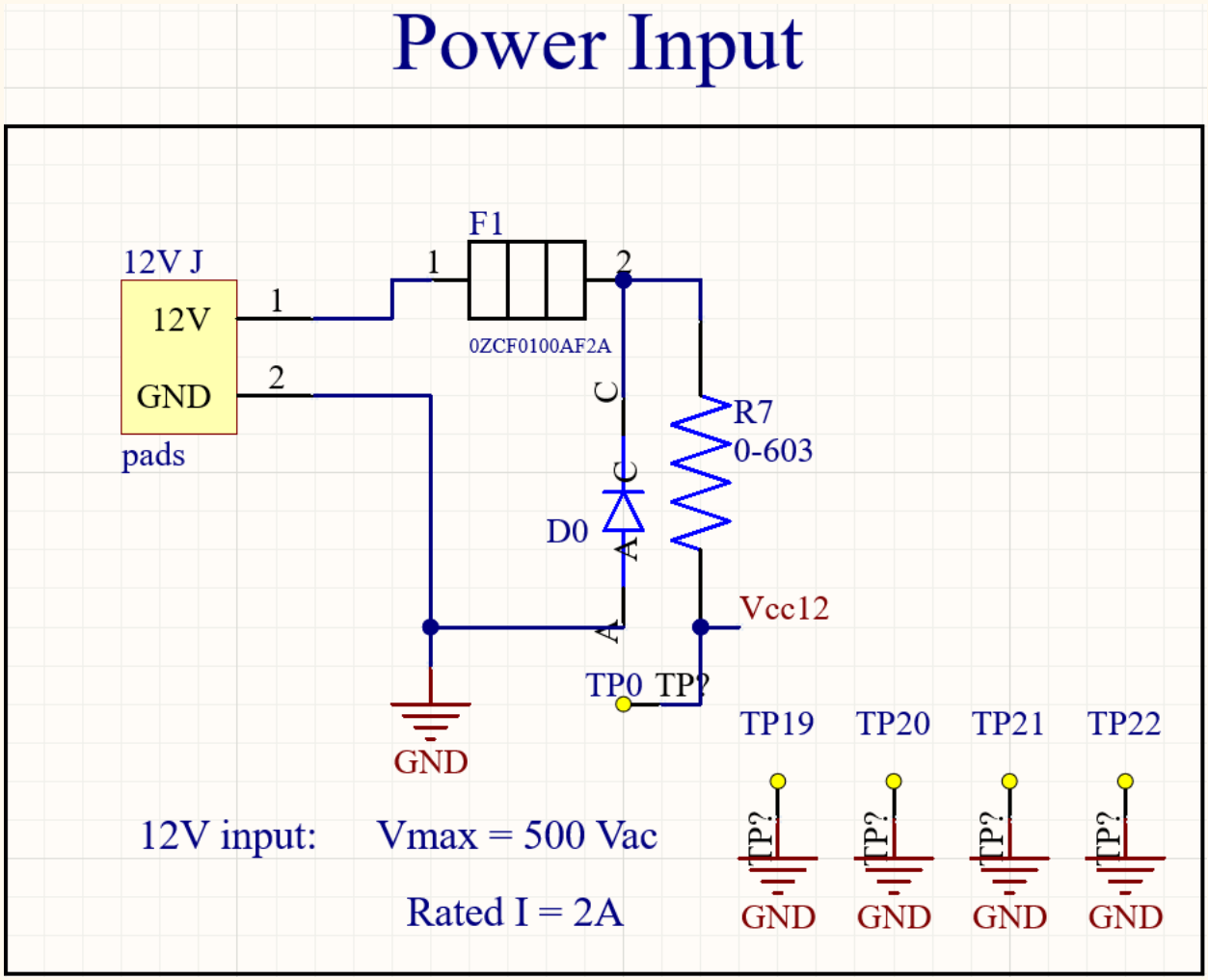


Figure 30. PCB Design Input

Below are the 5 V and 3.3 V regulators that power the rest of the components. They have decoupling capacitors as specified on their datasheet. The max input voltage is 15V for the 3.3V regulator and 35V for the 5V regulator which will be acceptable with the 12V power

supply. Both packages allow for the components to not overheat when the circuit is run. Both regulators are isolated with 0 ohm resistors. This can be seen in *Figure 31*.

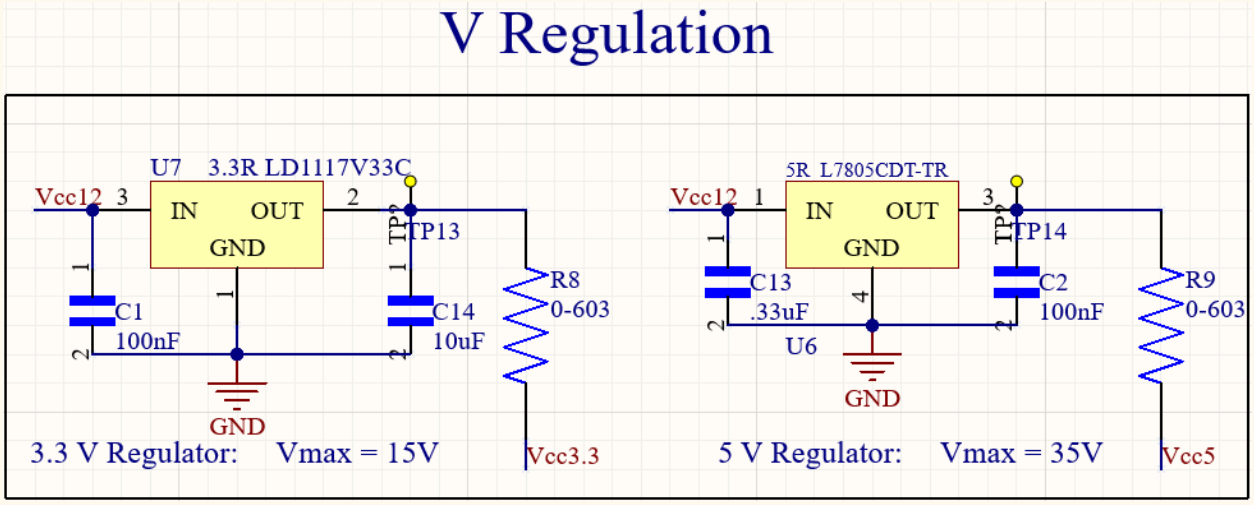


Figure 31. PCB Design Power Regulation

The next section is the H-bridges for the circuit. The H bridge is connected to both powerlines. The 3.3V is powering the logic circuit while the 5V is powering the motors connected via the header pin. Both sides of power inputs are decoupled by 100 nF capacitors. For configuration purposes, the thermal pad is connected to the ground. The Altium design for this section can be seen below in *Figure 32*.

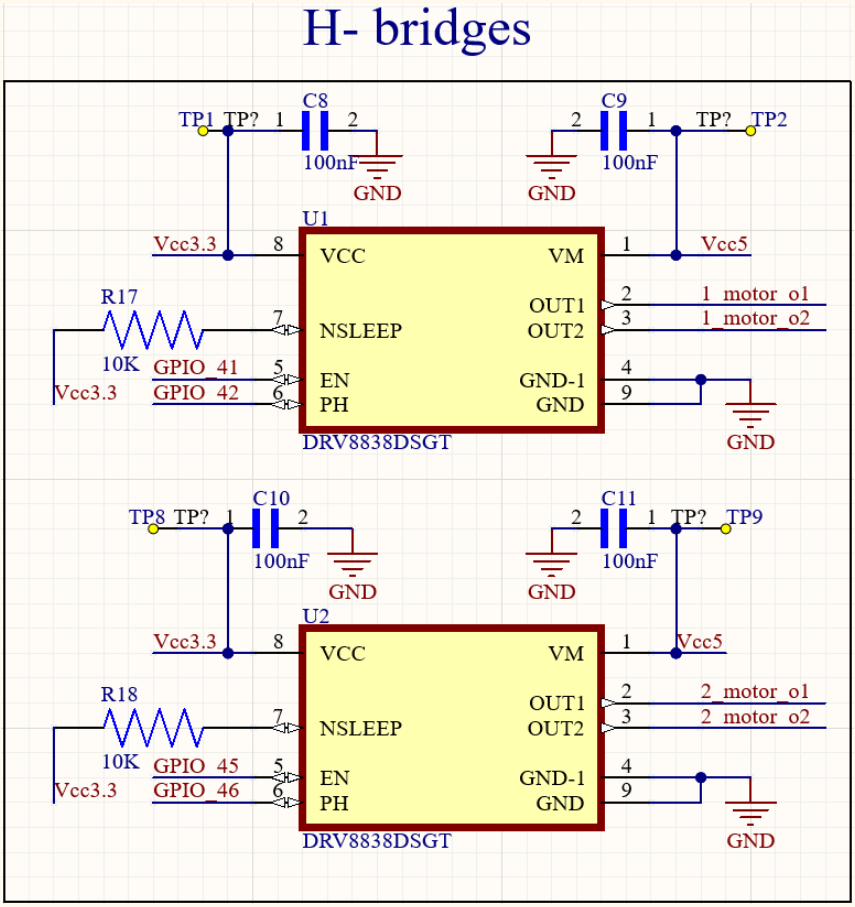


Figure 32. PCB Design H-bridges

The next section is the buttons for the circuit. There are two buttons on the board. The first is the boot button. This button sets the ESP into flashing mode to receive code from the computer. It has a 100 nF capacitor which is required by the ESP specifications sheet. Both sides of the button also have a test point. The second button is used to reset the ESP chip. It pulls the EN pin to Low and turns off the device. It is configured in the same way as the Boot button and is also isolated using a 0 ohm resistor. Both of these configurations are according to the ESP data sheet. The Altium design for this section can be seen below in *Figure 33*.

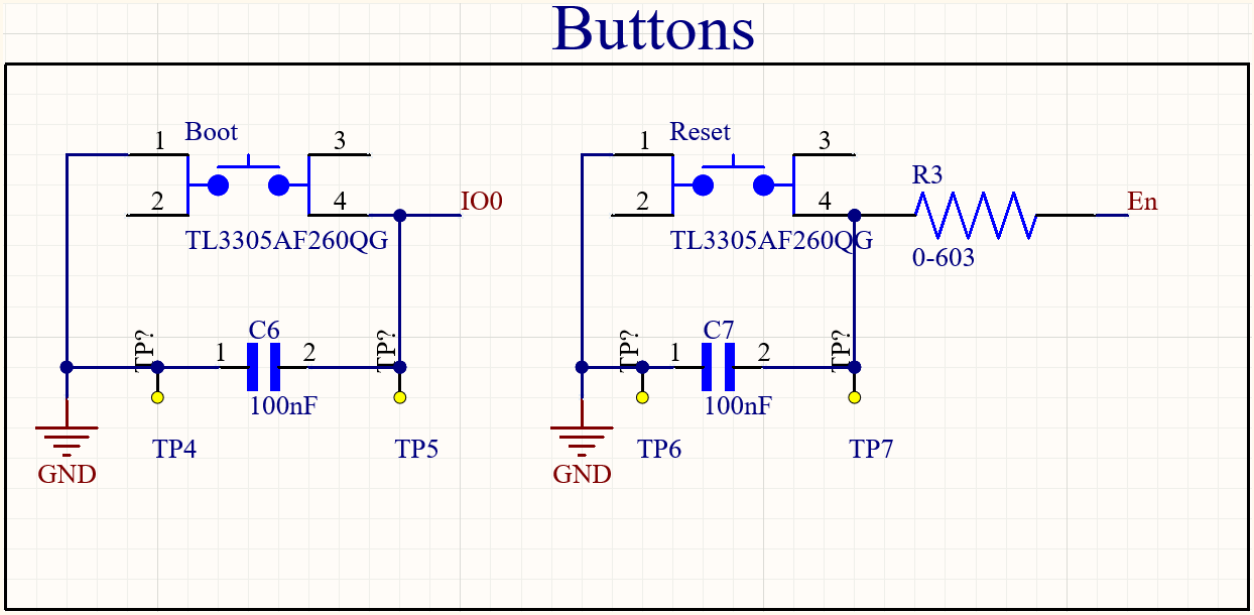


Figure 33. PCB Design Buttons

The next section is the header pins for the circuit. These are used to connect external objects to the PCB. Each known item has its own block of connections. The LEDs will be connected to P6, P7, and P8. The Motors will be connected to P4 and P5. Each time of flight sensor will be connected to either P9, P10, or P11. The program pins can be connected to one P0. To program the device a Serial to UART adapter is used and plugged into the P0 header pins. Keep in mind the boot button must be pressed to enter boot mode and flash the device. Since the board can be adapted to other projects, extra pins are made available. The Altium design for this section can be seen below in *Figure 34 and 35*.

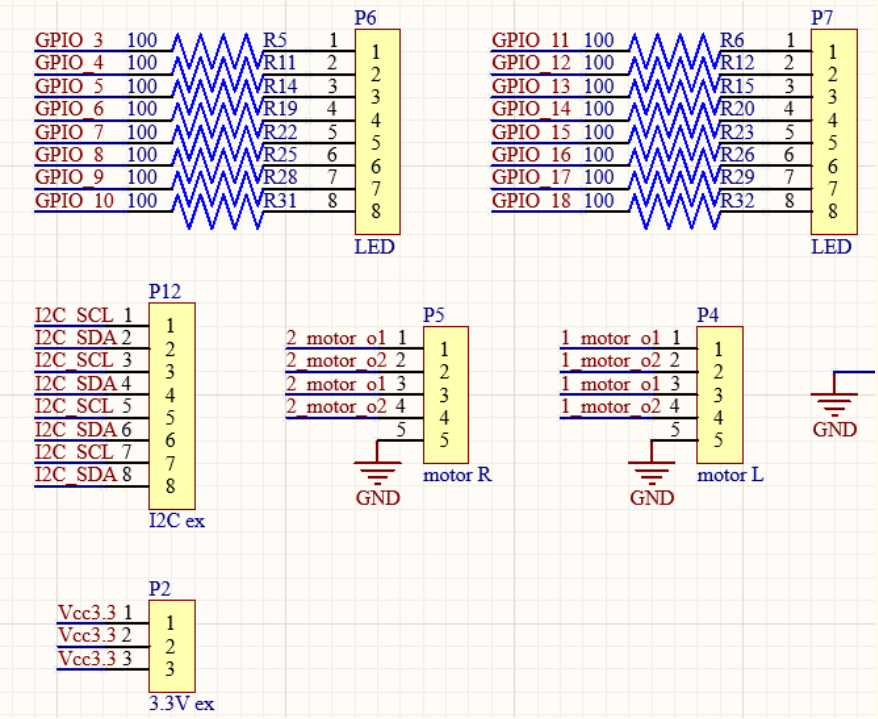


Figure 34. PCB Design Header Pins 1

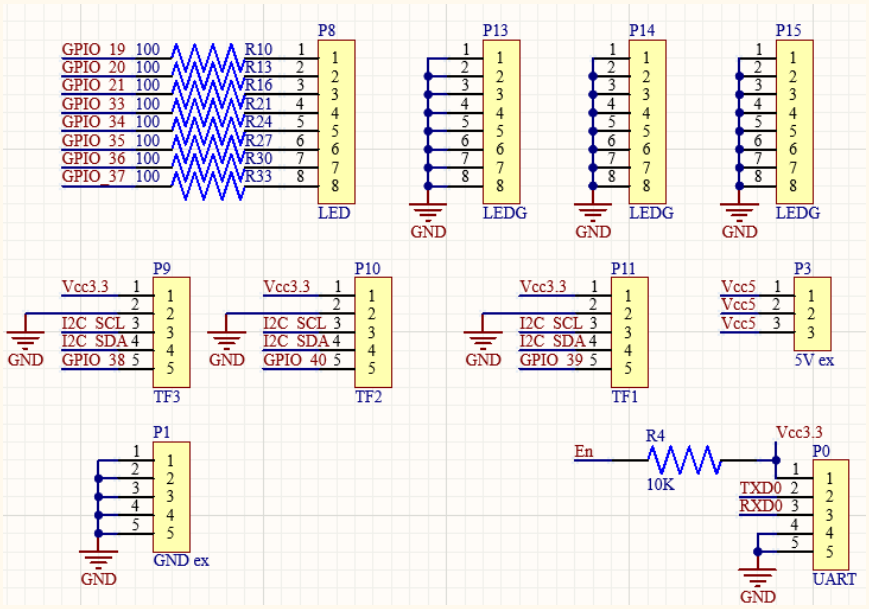


Figure 35. PCB Design Header Pins 2

For the PCB as a whole, all components are SMD packages to reduce space requirements. The PCB will be attached to each robot so that all components can connect without stretching or twisting of wires to ensure the safety of the components. The overall price for the PCB came out to $13.26. However, this price does not include shipping which can add a significant cost depending on the situation. This price in conjunction with cheap robot materials can meet the low-cost goal of the project. This is accomplished by making use of processing power and a higher number of GPIO ports. The MCU did cost more due to an onboard antenna and internal configurations, but this was a must to not spend more money on a full Bluetooth module and configuration components. The full Altium design can be seen below in *Figure 36*.

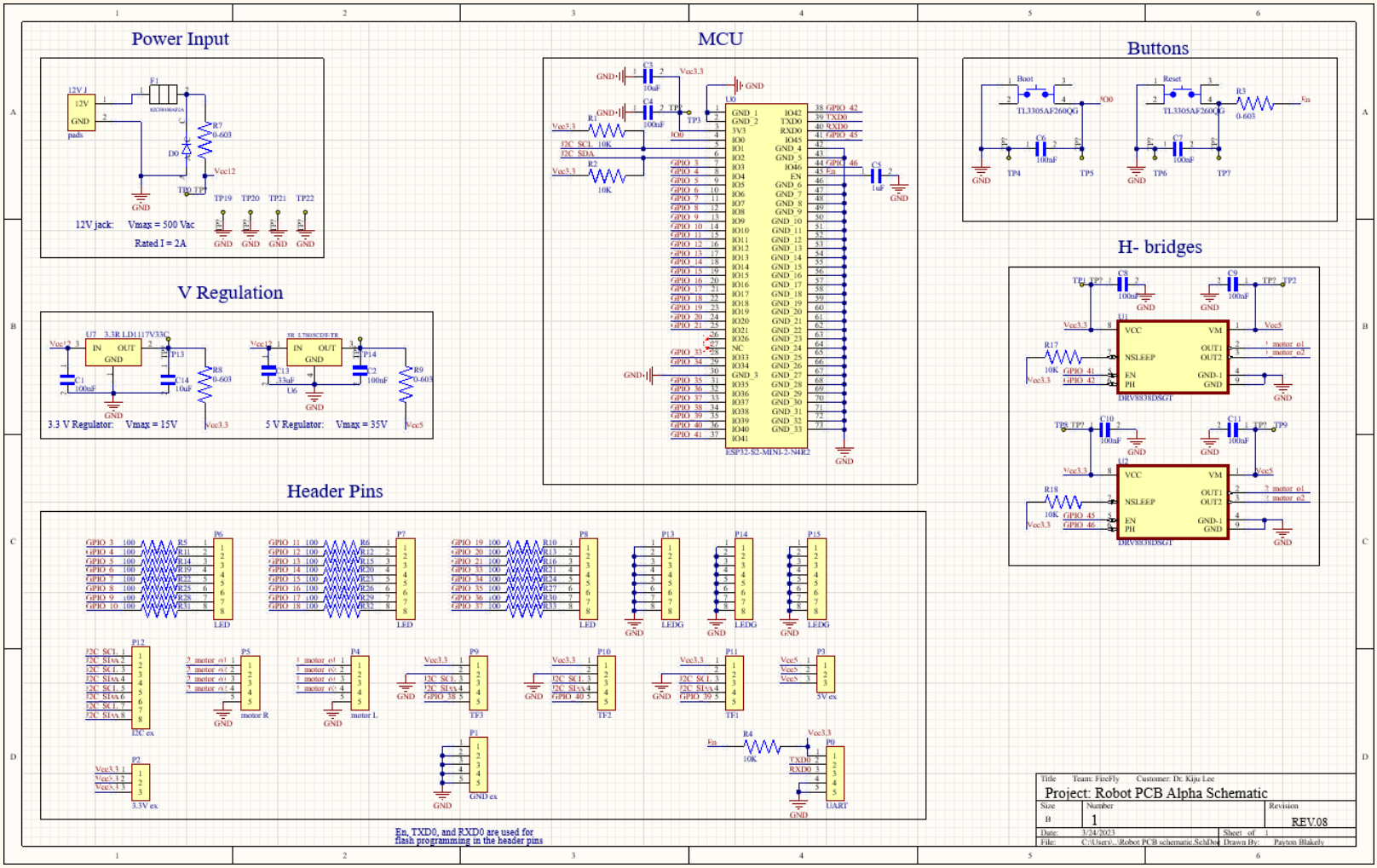


Figure 36. PCB Design

Below *Figure 37* shows the PCB layout. This layout allows for room to connect peripherals to the boards in organized clusters. All LED pins are located on the right of the board, TOF pins on the left, motor pins in the middle, program pins on the bottom left, extra I2C pins on the bottom right, and extra power pins at the top.

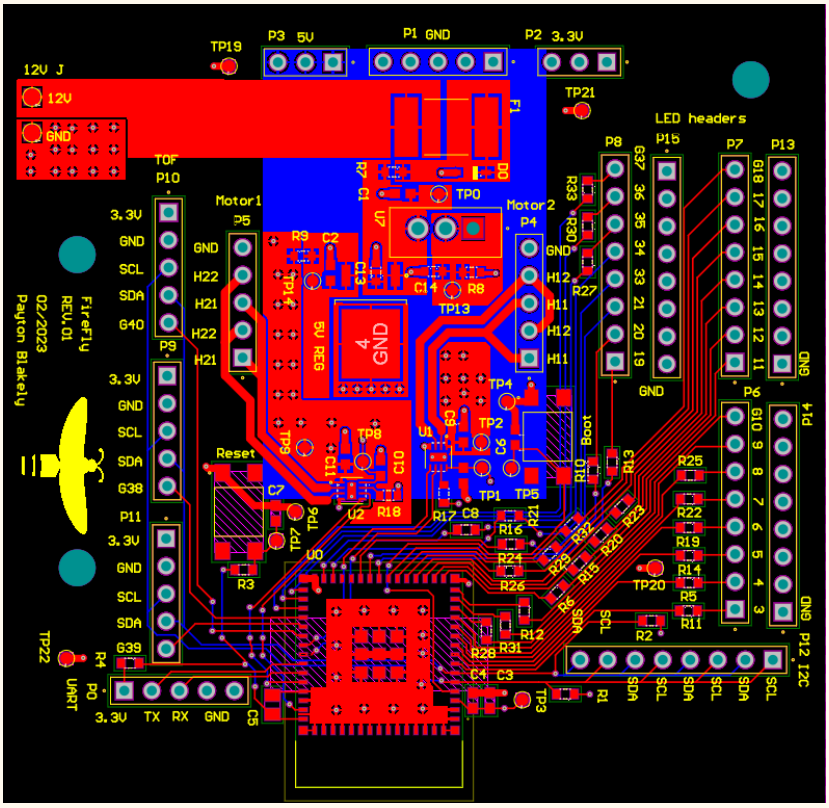


Figure 37. PCB Layout

Option 2: Config A

This configuration uses the same general components as the PCB design, but it is a bit cheaper overall due to lower shipping costs. This design's key components are the use of an Arduino Nano 3, multiplexor, and individual LEDs. One issue that was encountered was that some Arduino internal voltage regulators were better than others. When implementing them one might consider adding a higher quality voltage regulator between the battery and Arduino boards.

Part	Number
Arduino Nano	1
VLX53L01 Breakout Board	3
DC Motor	2
HC06 Serial-BL Converter	1
L298N H Bridge	1
LiPo Battery Tester	1
LEDs	16
Multiplexer	1
ZIPPY Flightmax 1500mAh 3S1P 20C	1
XT60H LiPo Connector	1

Table 2. Config A Parts

The schematic can be seen below in *figure 38*. The same TOF sensors are used in the same configuration, as well as the LEDs. A difference can be seen in the resistor setup in the design. Since the design requires only one LED to be activated at a time, only a single 100-ohm resistor is required to control the current of the LED array. There are some design considerations we learned while making the robots. First if implementing the project it is best to put all light sources such as the LEDs at the outermost part of the robot chassis. Also, use a material or cover that minimizes the reflection of light.

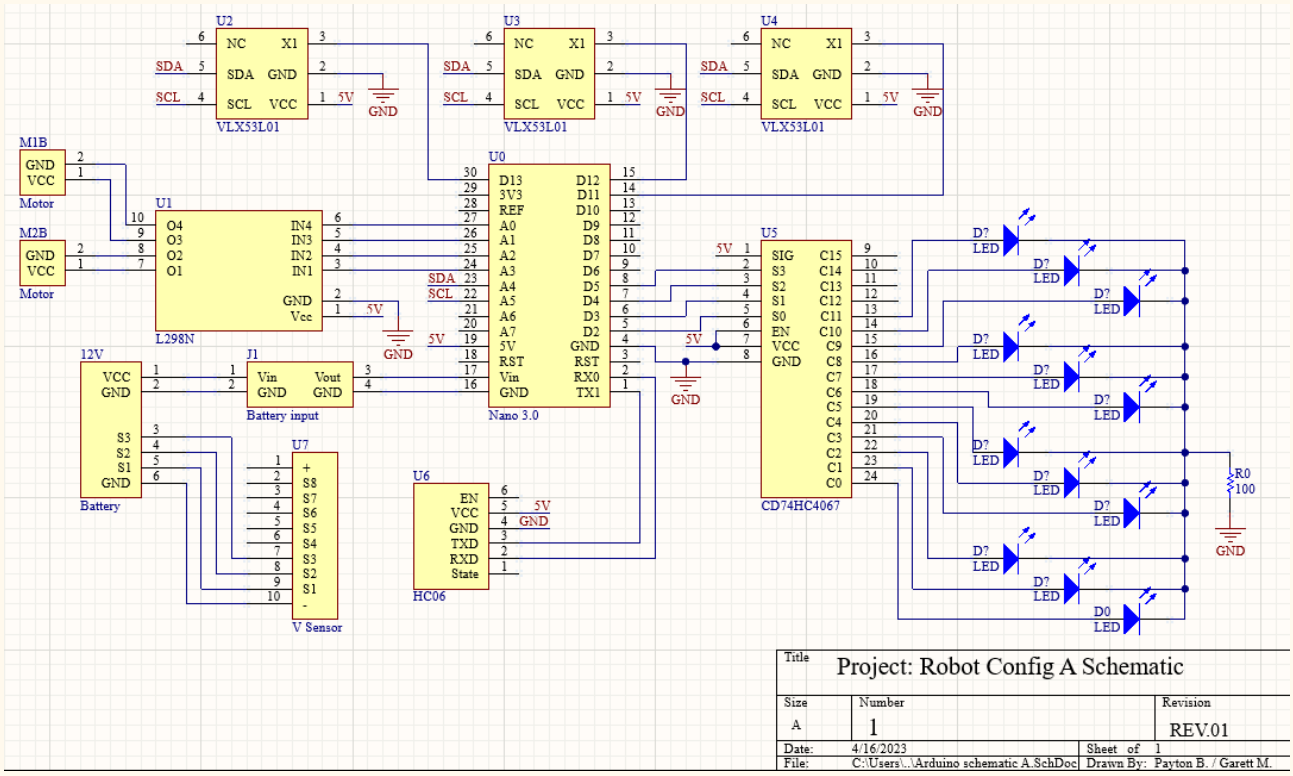


Figure 38. Config A Design

Option 3: Config B

Configuration B is similar to Configuration A. The main difference is that the light source used is a set of LED strips rather than individuals. This lowers the amount of GPIO pins required to drive the system so it also does not use the multiplexer-like configuration A. This configuration is also slightly cheaper than configuration A which makes this the cheapest option. The same issue in Configuration A was encountered, that some Arduino boards internal voltage regulators were better than others. When implementing them one might consider adding a higher quality voltage regulator between the battery and Arduino boards. The parts list and schematic can be seen below.

Part	Number
Arduino Nano	1
VLX53L01 Breakout Board	3
DC Motor	2
HC06 Serial-BL Converter	1
L298N H Bridge	1
LiPo Battery Tester	1
50mm LED Strip Segment	6
Turnigy 1000mAh 2S 20C LiPoly Pack	1
XT60H LiPo Connector	1

Table 3. Config B Parts

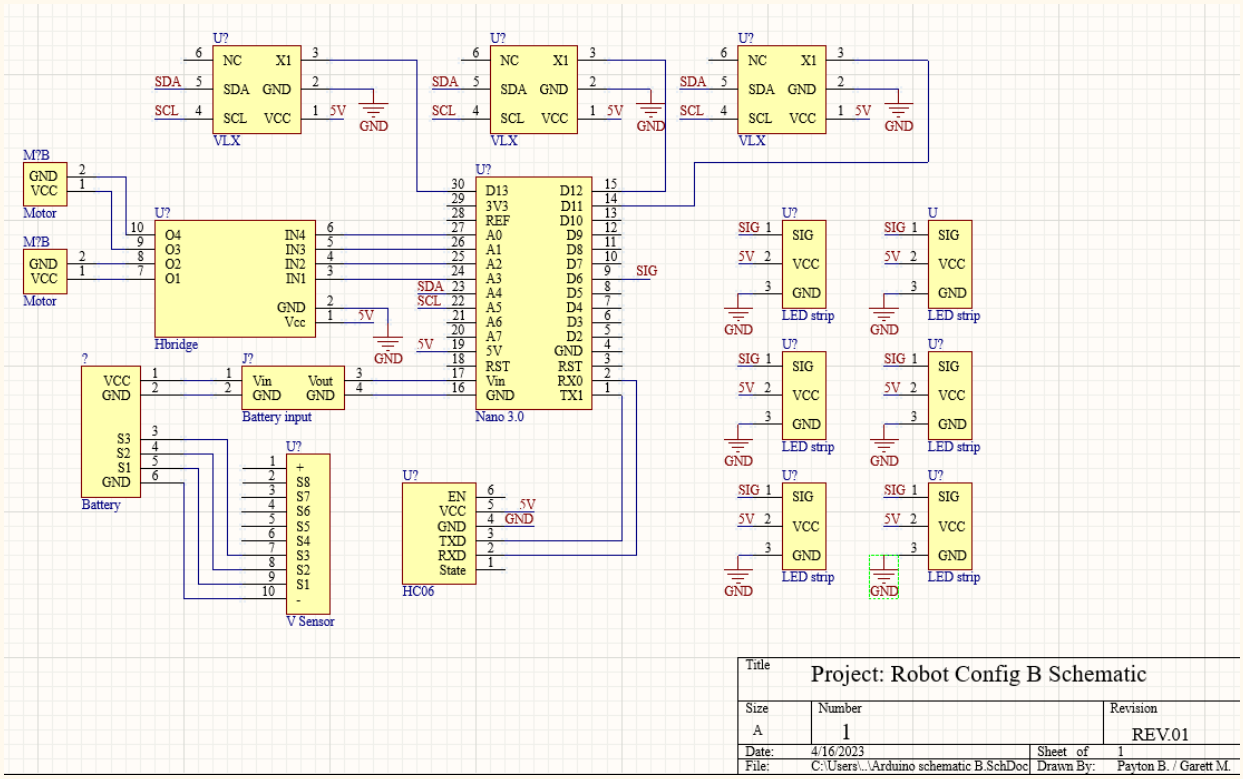


Figure 39. Config B Design

Software Flowchart

The software flowchart demonstrates the procedure that the localization control is going to be following. The whole procedure is a loop in which the system will interact in three different communication modes: Internal Process, Wireless message, and Camera Request. The system begins with the camera requiring an initial photo to analyze the location where the robots are, then a wireless message is going to request the position difference between the LEDs and the rover. The camera is going to take a photo and subtract the initial position and the internal process is going to locate the LED and start calculating the vector. If the vector is not completely analyzed the internal process is going to select the next LED and repeat the process. However, if the vector set is enough to fit, the system is going to fit a 3D model and calculate the position. After that, the module is going to identify the motion to continue the route, and a wireless message is going to send the motion information to be able to select the next LED and repeat the loop.

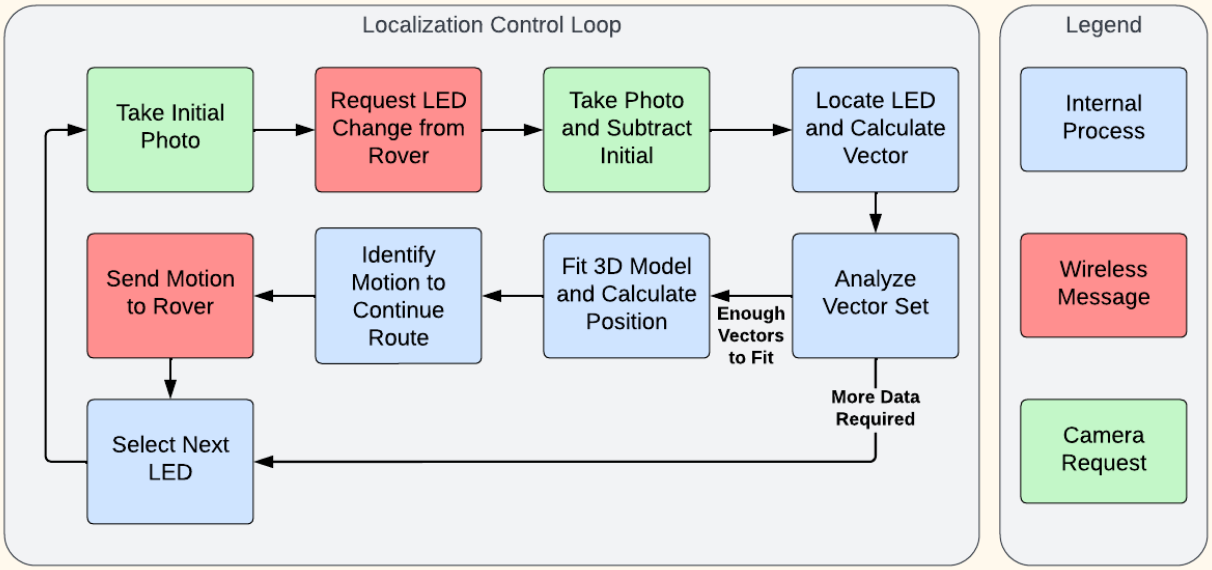


Figure 40. Localization Module Flowchart

Program Flowcharts

The robot will follow a methodology that can be seen in *figure 41* below. The program begins with receiving important information from the localization module. This includes which LED's need to be blinked and what are the current motor values that should be used for the motors. After receiving this the robot will then blink the correct LED immediately, and define a path to use from the data. The data is then received through the sensors on the robot. If there is an object that is detected it will send that data to the localization module and edit the current pathing to avoid the obstacle. After this is completed the motors will be engaged in a way that follows the current pathing definition. This line of logic will run with closed loop feedback. This is to ensure that the robots path is always being updated by both the localization module and the robot sensors.

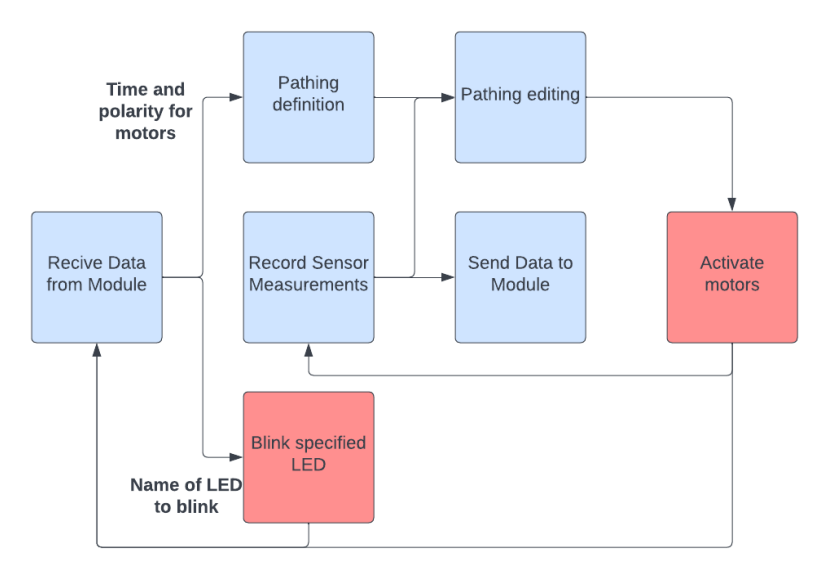


Figure 41. Robot Program flow chart

Gantt Chart

The Gantt chart is a project management tool that demonstrates the work completed over a period of time in relation to the time planned for the project. The Gantt chart consists of tasks and goals outlined for the project within the design, manufacturing, and testing and integration phase.The left side outlines a list of tasks, while the right side has a timeline with schedule bars that visualize work. The following figures demonstrate the tasks planned until April 21st, 2023.

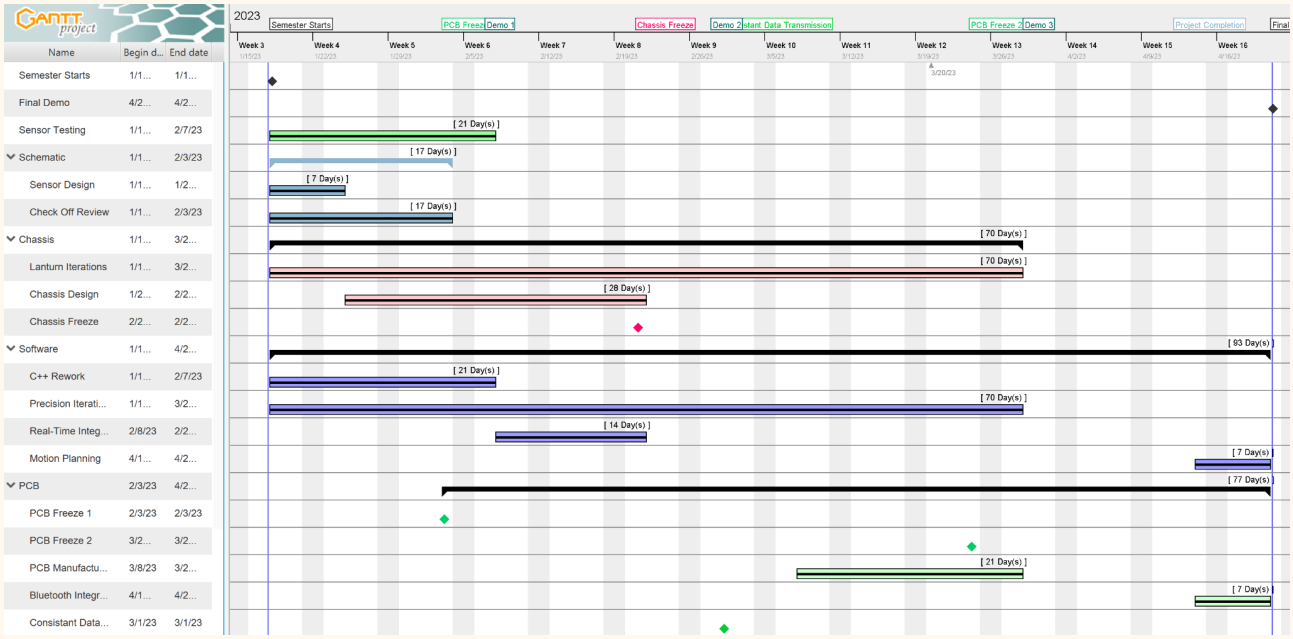


Figure 42. Gantt Chart

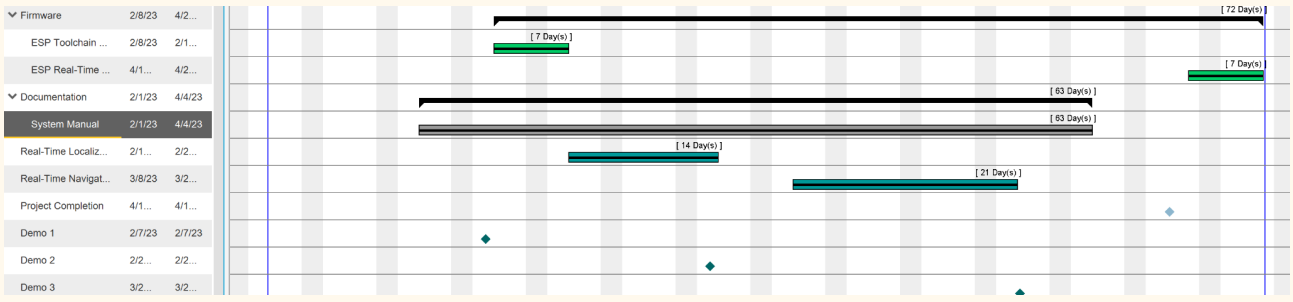


Figure 43. Gantt Chart Continued

Testing

The test matrix (*Figure 44*) demonstrates the components that need to be tested and the requirements that each needs to meet. The components that need to be tested are selected because they are important to make the full system perform its job correctly. The requirements are the goals that need to be met. The team is going to test each of the sections on the left side carefully and if the requirements are met, try to improve them. This test matric helps to keep everything in order and make sure that everything is going to work when it is going to be put together. Some of the components have the same requirements, which in some cases can or have to be tested together.

To test the electrical design, test points were included in the PCB for voltage testing. Most testing included probing all major components with a Voltmeter to see if everything was appropriately powered. The data lines were also tested before connecting them to the peripherals. This is how it was determined there was package loss on the I2C bus, but all other lines were functioning properly. A similar testing was done on the Arduinos configurations but instead probing the wired connections. The components were also checked for excessive heat. There are many manufacturing issues that can arise from cheaper parts, so if there seems to be a heat problem there is most likely a faulty part or bad connection. If the problem is not obviously found then replacing the component is a good troubleshooting option.

Test / Requirements	*Movement Generation*	*Wireless Communication*	*Mapping*	*Navigation*	*Power Supply*	*Decision Making*	*Team Search*
Motor Speed	■		■	■	■		
Ultrasonic Sensor Accuracy	■		■	■			
Wireless Connection Strength		■					■
Camera Resolution			■	■			
Image Processing Speed		■					
Image Processing Accuracy			■			■	
Hardware Connections					■	■	
Object Detection	■		■	■		■	
Path Generation Accuracy			■	■			■

Figure 44. Test Matrix

Technical Impact

Firefly's team of robots and localization system technical impact lies in the ability to accurately travel through a space autonomously, as well as reducing the cost to access localization by thousands of dollars. By using low budget components, several robots can be purchased and modified to produce a swarm of robots. Because the robots are low budget, they are disposable, meaning companies or schools can buy multiple robots without having to worry about costs if they break.

Due to the scalable and disposable nature of these robots, the Firefly team of robots is capable of being used in a multitude of fields. For example, mapping a path based on the environment can be used for searching for outlets in collapsed spaces. Or in education, each student can be given a robot and the introduction to robotics can begin at an earlier age. This is supported by the custom designed code and circuits built into this project.

Upgrading the system to swarm configuration would only require adding a camera to each robot and creating a network for them to communicate. The main process of localization would remain the same and the core functions of both robot and program wouldn't require much changing. Doing this opens the door to many new complex applications for the FireFly system.

Cost Breakdown

There are three bills of materials (BOM) as this project was revised due to electronic errors. The first BOM show the cost of a robot, the localization module, and the component to build a PCB. The PCB section has bought parts to manufacture at least 6 spares. Overall The total cost of everything bought is $212.43 plus another $70 for shipping costs. However the cost to build one robot costs $88.14 which a team of (minimum of 3) robots costs $148.08. This is for the robots alone. Remember that 3D printing is not the best method of mass production and can be subtracted with injection molding.

As a result of PCB error, the new electronics were put into place as shown with configuration A and B. The total of config A is $196.01 (not including shipping). Whereas the total for config B is $160.35 (not including shipping). It can be noticed that the localization module for the system remains the same throughout, however the design approach for the robot changes the overall cost.

The total cost is something the team is very aware of and something that ended up being a great success. A budget of $1500 was given to the team and a $500 personal goal budget was used as FireFly's inspiration to make an affordable robot and localization system. As shown in the tables below, the cost projections to build a full system with 3 robots came at under half the personalized budget.

Description	Quantity	Unit Cost	Extended Cost
ROBOT (for one robot)			
2 x TT Motors + Tires	1	$2.26	$2.26
Ping Pong Ball (pack of 15)	1	$3.99	$3.99
1kg PLA+ Filament	1	$24.99	$24.99
Time of Flight Sensor	3	$1.26	$3.78
Screw Insert (pack of 50)	1	$12.65	$12.65
M3 Screws (pack of 100)	1	$5.51	$5.51
Battery	1	$12.97	$12.97
LOCALIZATION MODULE			

1080p Desktop Camera	1	$12.99	$12.99
Raspberry Pi	1	$35.00	$35.00
PCB (all components are SMD; includes 3 PCBs + 6 spares)			
Button Switch	18	$0.21	$3.69
ESP Module	9	$2.64	$23.76
100 ohm Resistors	216	$0.02	$4.97
H-Bridge	10	$1.30	$13.00
Diodes	10	$0.18	$1.80
10uF Capacitor	18	$0.08	$1.37
0.33uF Capacitor	10	$0.11	$1.10
10kOhm Resistor	36	$0.02	$0.79
0 ohm Resistor	63	$0.07	$4.35
1A 60V Fuses	10	$0.54	$5.39
5V 1.5A Voltage Regulator	10	$1.00	$10.00
3.3V 0.8A Voltage Regulator	10	$0.78	$7.77
1uF Capacitor	10	$0.06	$0.62
0.1F Capacitor	100	$0.01	$1.00
3mm 20mA White LED	40	$0.14	$5.68
Printed Circuit Board (10 boards)	1	$13.00	$13.00

Table 4. Cost Breakdown (PCB)

Total	**$212.43**
Total+Shipping (Estimated)	$287.43
True Total of 1 Robot	**$88.14**
True Total of 3 Robots	**$148.08**

Table 5. Total (PCB)

Description	Quantity	Unit Cost	Extended Cost
ROBOT (for one robot) Config A			
2 x TT Motors + Tires	1	$2.26	$2.26
Ping Pong Ball (pack of 15)	1	$3.99	$3.99
1kg PLA+ Filament	1	$24.99	$24.99
Time of Flight Sensor	3	$1.26	$3.78
Screw Insert (pack of 50)	1	$12.65	$12.65
M3 Screws (pack of 100)	1	$5.51	$5.51
Battery -ZIPPY Flightmax 1500mAh 3S1P 20C	1	9.32	9.32
XT60H Lipo Connector	1	7.99	7.99
Arduino Nano	1	2.1	2.1
HC06 Serial-BL Converter	1	1.96	1.96
L298N H Bridge	1	0.46	0.46
Lipo Battery Tester	1	1.14	1.14
Multiplexer (pack of 5)	1	6.99	6.99
LEDs	16	0.143	2.288
LOCALIZATION MODULE			
1080p Desktop Camera	1	$12.99	$12.99
Raspberry Pi	1	$35.00	$35.00

Table 6. Cost Breakdown Config A

Total	**$196.01**
Total+Shipping (Estimated)	$266.01
True Total of 1 Robot	**$85.43**
True Total of 3 Robots	**$148.02**

Table 7. Total Config A

Description	Quantity	Unit Cost	Extended Cost
ROBOT (for one robot) Config B			
2 x TT Motors + Tires	1	$2.26	$2.26
Ping Pong Ball (pack of 15)	1	$3.99	$3.99
12" x 8" x 1/8" Plywood	1	$4.99	$4.99
Time of Flight Sensor	3	$1.26	$3.78
Screw Insert (pack of 50)	1	$12.65	$12.65
M3 Screws (pack of 100)	1	$5.51	$5.51
Battery - Turnigy 1000mAh 2S 20C LiPoly Pack	1	5.09	5.09
JST-2Pin Lipo Connector	1	10	10
Arduino Nano	1	2.1	2.1
HC06 Serial-BL Converter	1	1.96	1.96
L298N H Bridge	1	0.46	0.46
Lipo Battery Tester	1	1.14	1.14
50mm LED Strip Segment (pack of 20)	1	4.85	4.85
LOCALIZATION MODULE			
1080p Desktop Camera	1	$12.99	$12.99
Raspberry Pi	1	$35.00	$35.00

Table 8. Cost Breakdown Config B

Total	**$160.35**
Total+Shipping (Estimated)	$230.35
True Total of 1 Robot	**$58.78**
True Total of 3 Robots	**$112.36**

Table 9. Total Config B

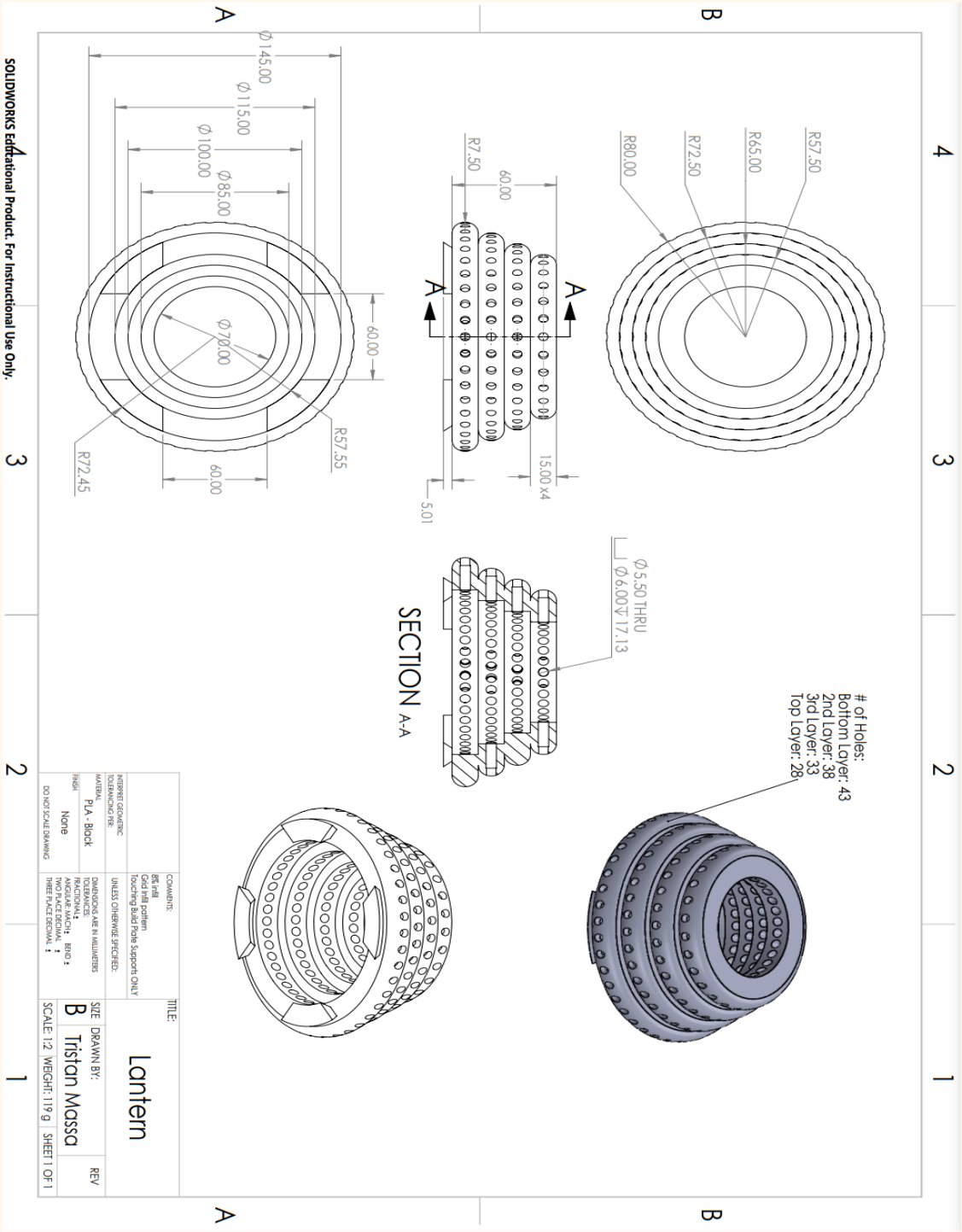
Conclusion

In summary, there is a lack of low-budget high performance localization systems currently on the market. FireFly plans to design, build, and test this system using LEDs and integrating this with a team of robots. These robots can be further integrated by forming a team of robots which can be used in a variety of applications. Each robot can move independently or in unison to successfully complete tasks. By exchanging data with the localization module, the robots move autonomously throughout an environment while having sensors on itself, allowing them to avoid obstacles within its path.

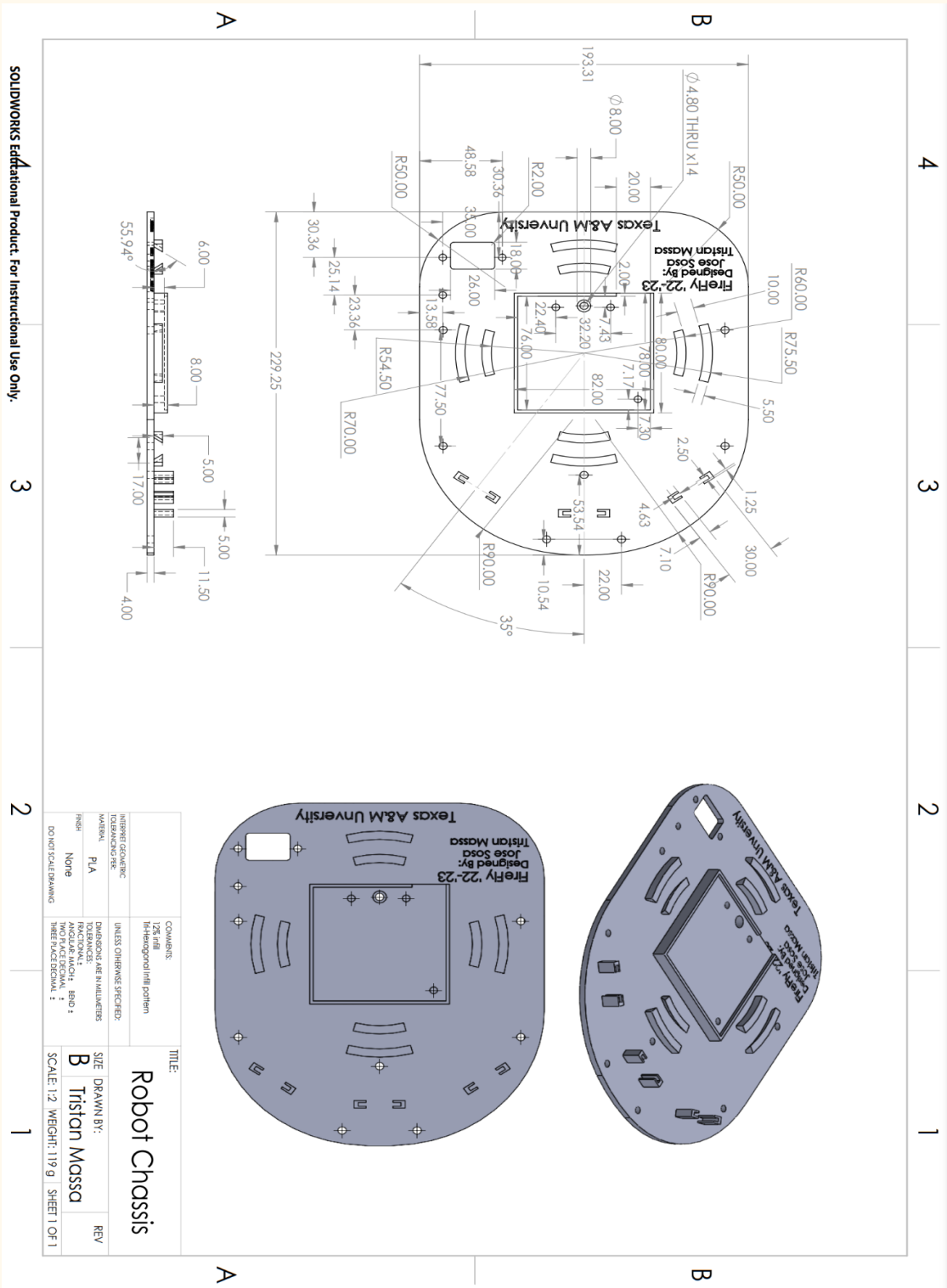
Special Recognition

A special recognition to Dr. Kiju Lee, Prof. Harley Willey and the Engineering Technically and Industrial Distribution department at Texas A&M University for providing us with the guidance and necessary tools to work on this project. We have and will continue to achieve what would not have been possible without the broader support of the engineering community that allowed us to take on such an ambitious project.

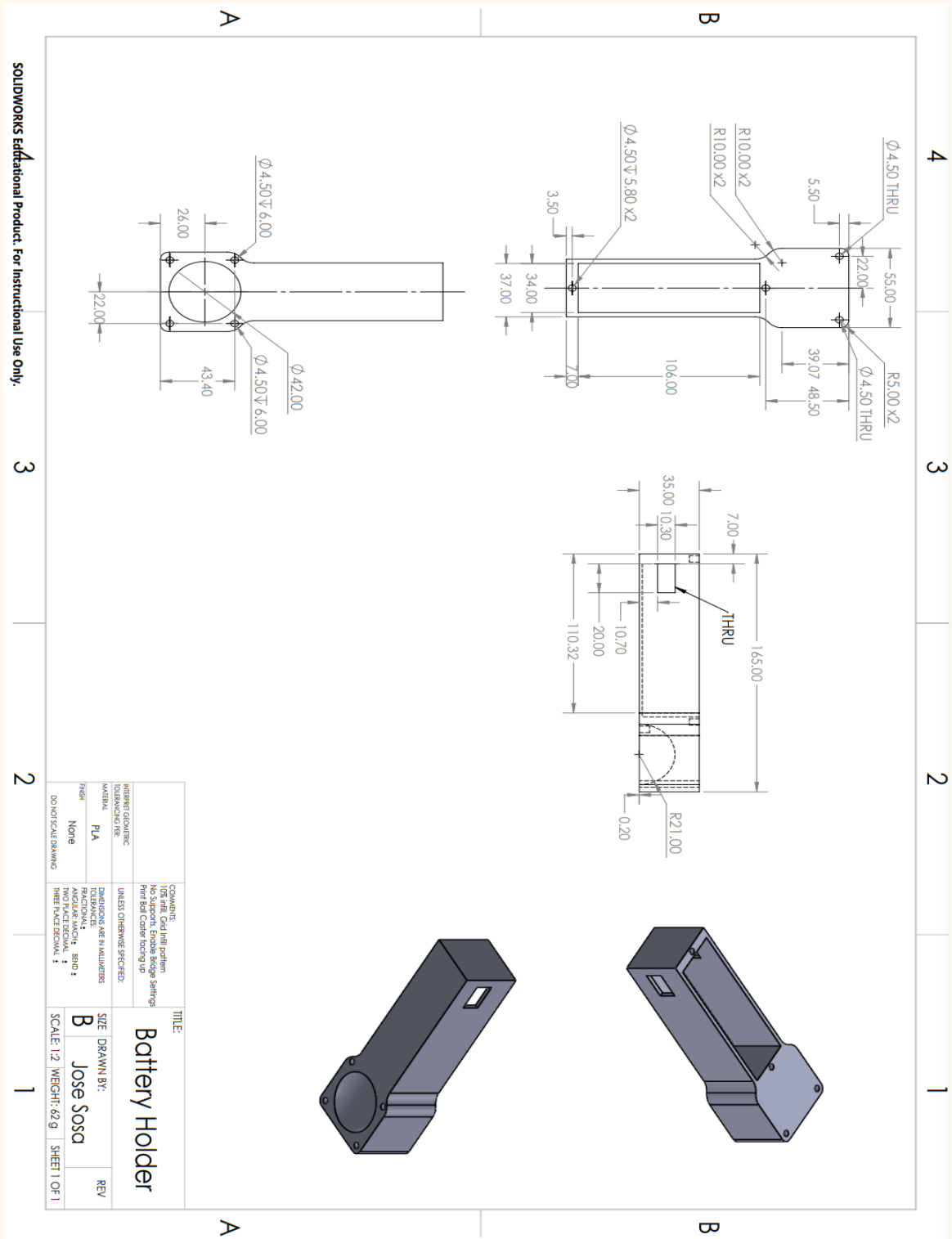
Appendix



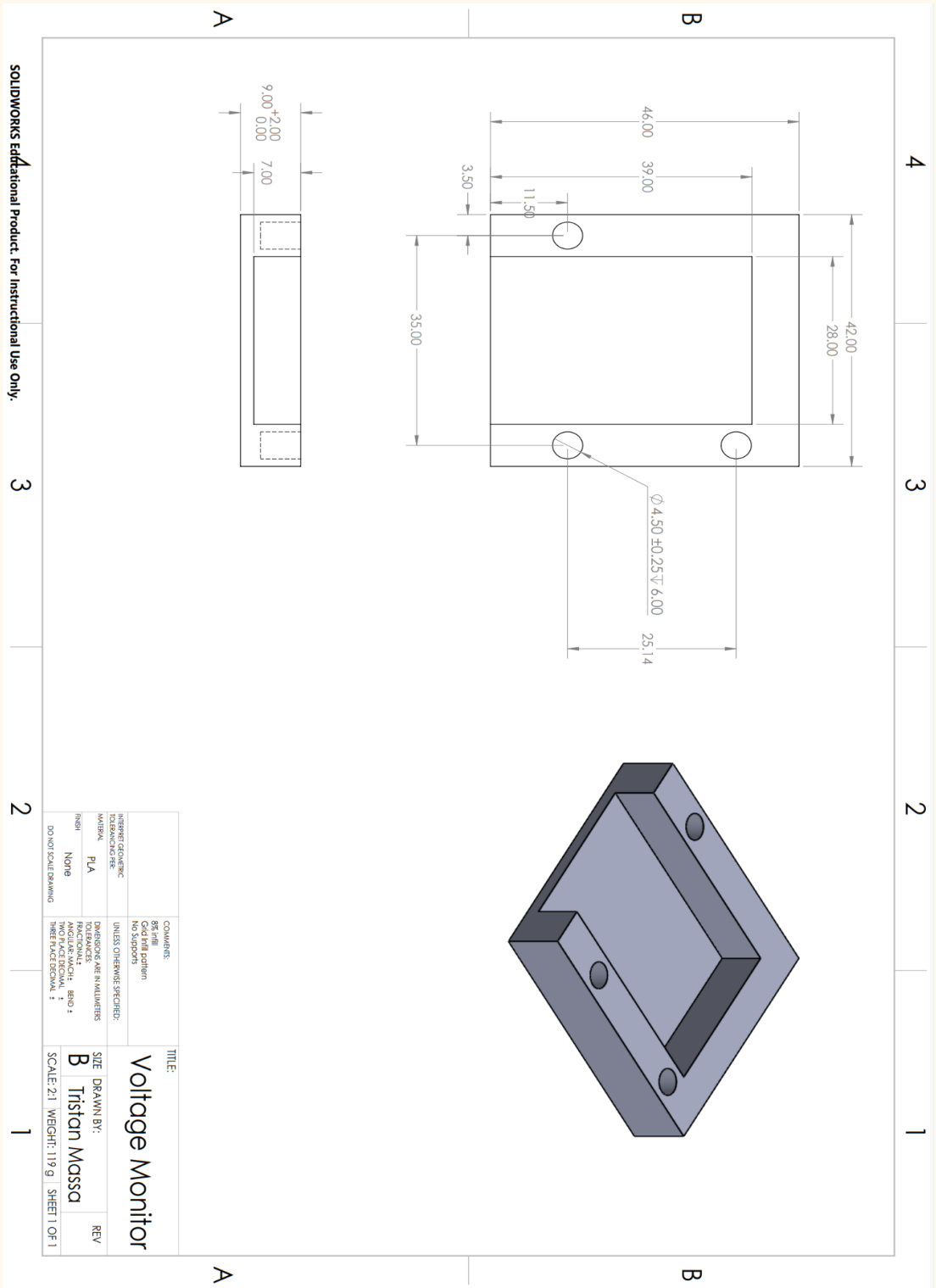
Appendix 1: Lantern



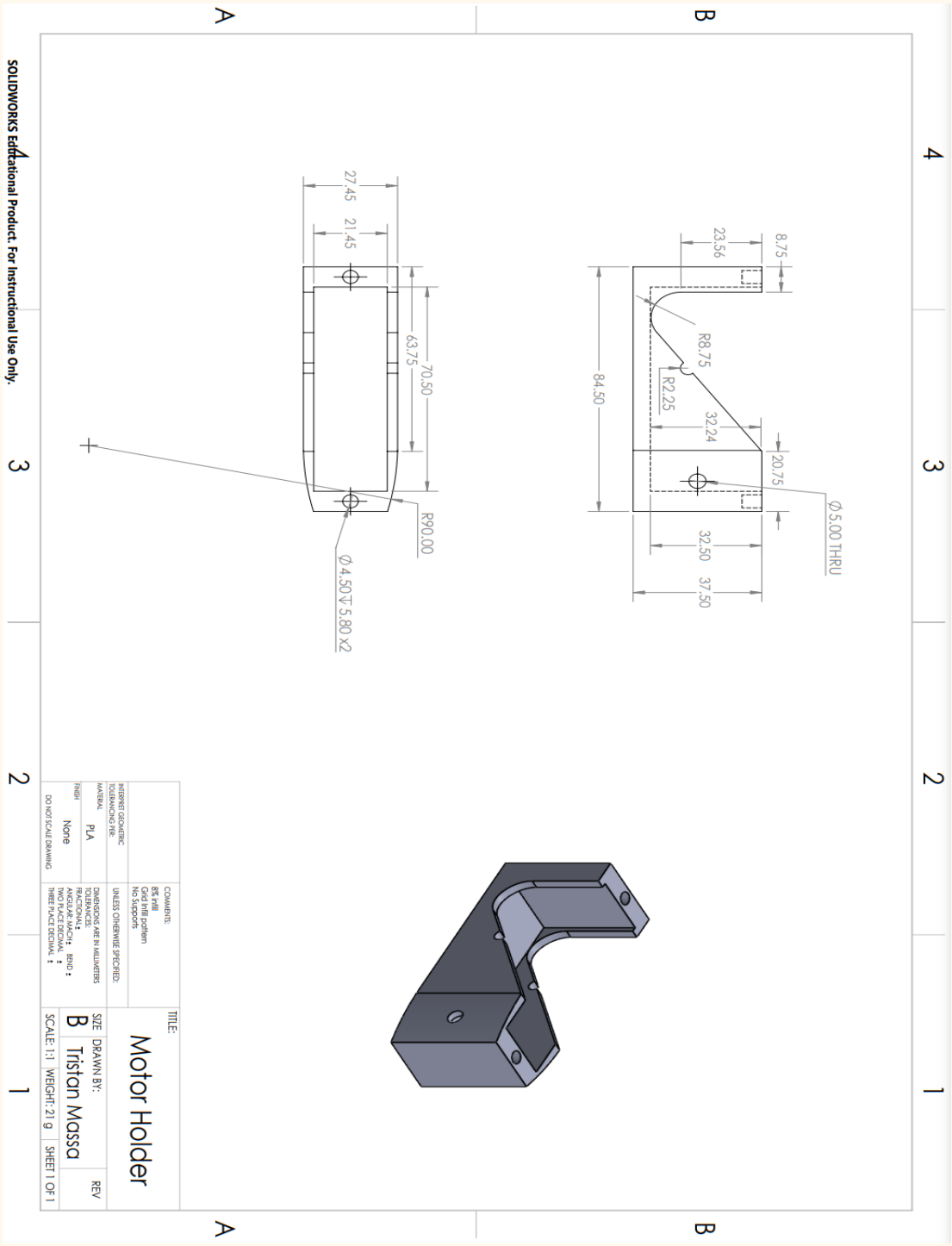
Appendix 2: Chassis



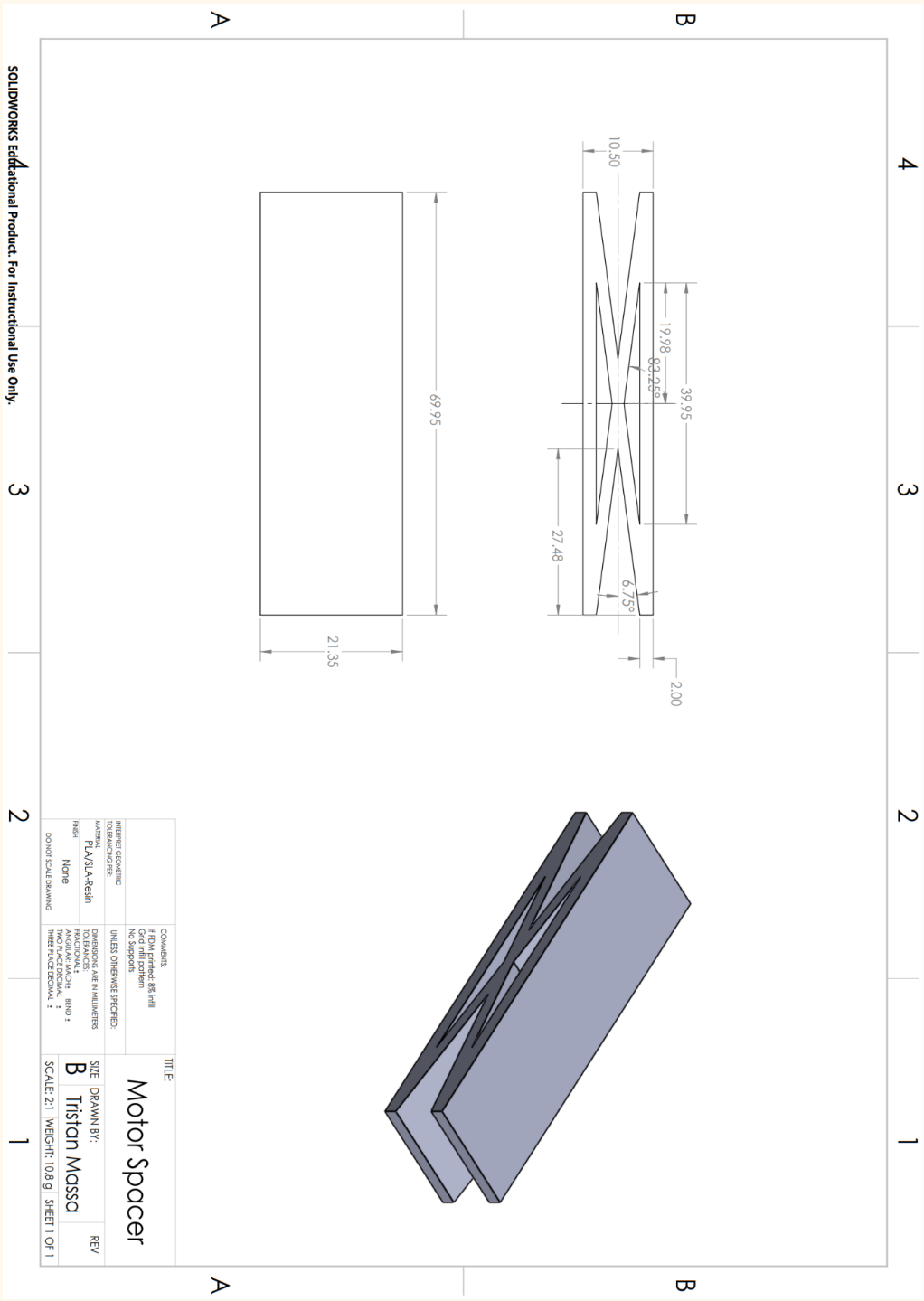
Appendix 3: Battery/Ball Holder



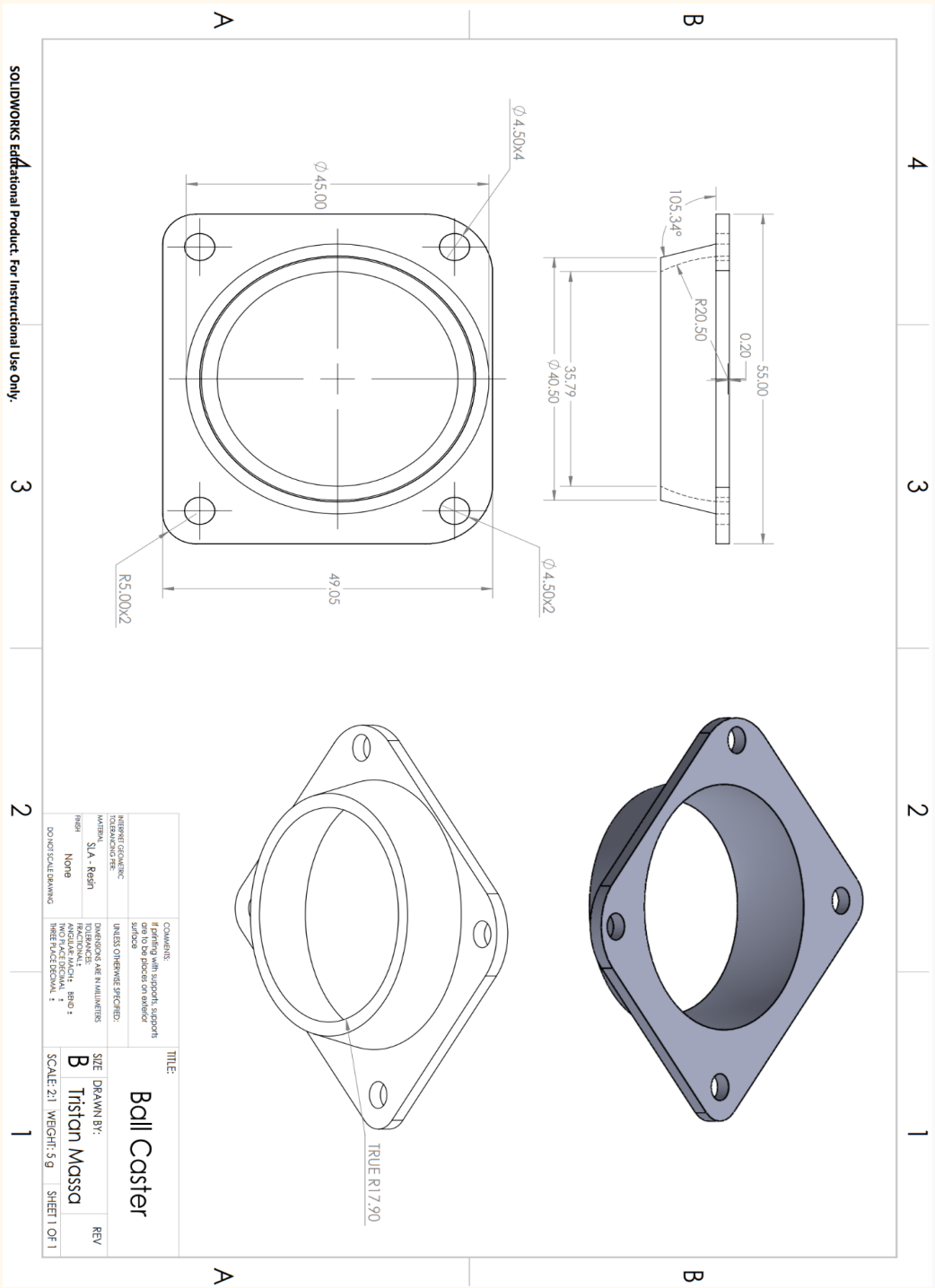
Appendix 4: Voltmeter Holder



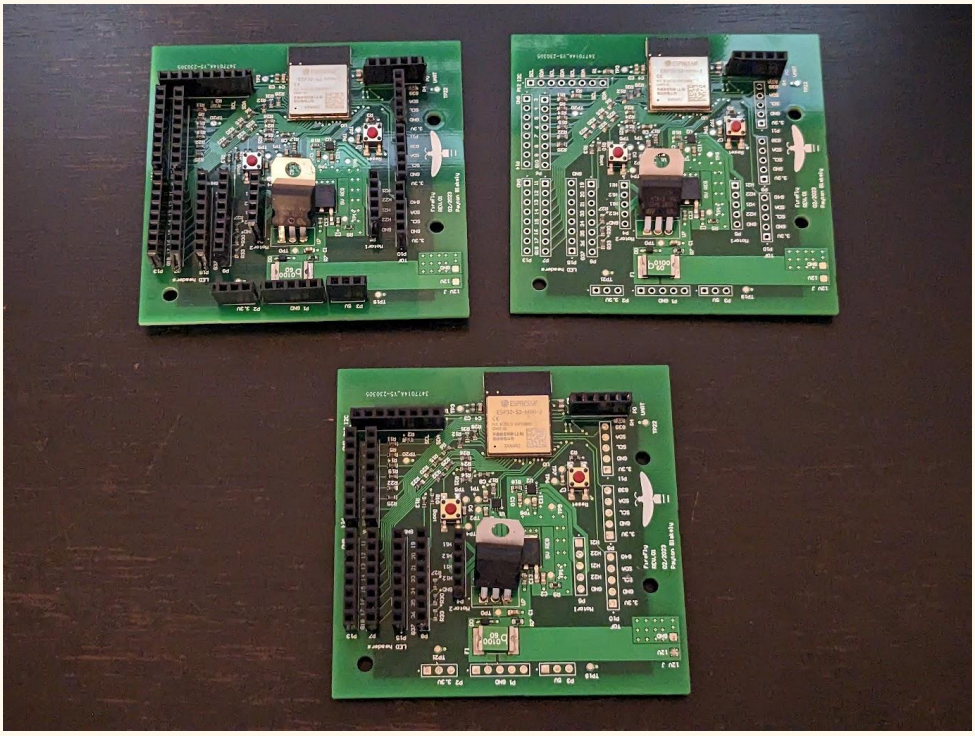
Appendix 5: Motor Holder



Appendix 6: Motor Spacer



Appendix 7: Ball Caster



Appendix 8: PCBs