Solar Agribot

A Major Qualifying Project Report submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the degree of Bachelor in Science

Submitted By:

Noelle Crump	Mechanical Engineering
Taylor Frederick	Mechanical Engineering
Thananart Jay Piyajarawong	Electrical and Computer/Robotics Engineering
Lin Guan	Electrical and Computer Engineering
Ashley Pavlov	Mechanical Engineering
Philgen Simpson	Mechanical Engineering
Travis Thompson	Computer Science/Electrical and Computer Engineering
Tianshu Wang	Electrical and Computer Engineering
Vasil Bozdo	Electrical and Computer Engineering

August 2022 - May 2023

Professor Mehul Bhatia, Advisor
Professor Nicholas Bertozzi, Advisor
Professor Joseph Beck, Advisor
Professor Markus Nemitz, Advisor

Abstract

The agriculture industry is in crisis because of population increase, labor shortages, and rising costs of supplies. We developed a battery-powered robot prototype to improve corn production efficiency while limiting use of fossil fuels. The robot takes soil samples and selectively administers fertilizer. It communicates with and parks in a hub station that refills the fertilizer tank and uses a solar collector to recharge the battery. To traverse the uneven crop rows, a rover design was created with a rocker suspension and skid steering. A dashboard and data storage system was created to allow for easy visualization and storage of soil data. The robot's modular design permits swift component replacement while a base station offers solar charging, navigational aid, and elemental protection.

Acknowledgements

The team would like to thank Professor Bhatia and Professor Bertozzi for advising us throughout the entirety of this project. We would also like to thank Professor Beck and Professor Nemitz for joining our project with late notice. We would also like to acknowledge the staff members and students that work in Washburn Shops and PracticePoint for assisting us with manufacturing.

Authorship

Abstract	**Taylor, Travis**
Acknowledgements	**Taylor**
Authorship	**All**
Table of Contents	**All**
List of Figures	**Taylor**
List of Tables	**Taylor**
Executive Summary	**Taylor, Travis**
Methodology	Travis
Key Features	Taylor, Travis
Implications and Recommendations:	Travis
Conclusion	Travis
References	**All**
Chapter 1: Introduction	**Vasil**
Chapter 2: Background	
2.1 Previous Agribots and Rovers	Taylor
2.2 Statement of Work	Ashley
2.3 Robot Subsections	Travis
Chapter 3: Design and Development	
3.1.0 System Overview	Travis
3.1.1 Mechanical Requirements	Ashley
3.1.2 Electrical Requirements	Travis
3.1.2.1 Power in the Robot	Lin, Vasil
3.1.2.2 Navigation	Jay
3.1.2.3 Fertilization	Tianshu
3.1.2.4 Soil Testing	Jay
3.1.3 Software Requirements	Travis
3.2.0 Conceptual designs and prototyping/modeling feasibility studies	Travis
3.2.1 Chassis	Taylor, Ashley, Philgen
3.2.1.1 Chassis Preliminary Design	Taylor
3.2.1.2 Torque Calculations	Taylor, Philgen
3.2.1.3 Rocker Redesign	Taylor
3.2.1.4 Final Chassis Design	Taylor
3.2.1.5 Manufacturing and Assembly of Final Chassis Prototype	Taylor, Ashley
3.2.2 Soil Probe Design	Philgen
3.2.3 Battery Level Indicator Design	Vasil
3.2.3.1 Circuit Design	Vasil
3.2.3.2 Selection of resistors	Vasil
3.2.3.3 Implementation	Vasil
3.2.4 Radio Transmitter Circuit Prototype	Travis

Table of Contents

List of Figures

List of Tables

Executive Summary

The Solar Agribot Project aimed to address the rising costs of corn production and the critical decline in the agricultural workforce due to decreasing interest in agriculture, inefficient labor programs, and an aging population of farm operators (Ku, 2022)(USDA). In light of the expected 70% increase in calorie demand to sustain the projected global population of 9.7 billion by 2050, the Solar Agribot team designed and implemented a robot that is ready to develop autonomous robot capable of collecting soil viability data and selectively administering fertilizer in corn fields to optimize growth (Goedde et al., 2020). With corn being the most widely produced feed grain in the United States, the implementation of this technology could significantly improve the efficiency of the agricultural industry and help meet the increasing global demand for food.

Methodology

The Solar Agribot Project divided the project into four phases: research, development, integration, and testing. During the research phase the team looked into the work of the past year and into other agribots on the market. In the development phase the team split into three sub groups: Electrical, Mechanical, and Software. Each group focused on developing their systems in depth while keeping communication between the groups to ensure seamless integration. During The integration phase, the team came together to assemble all of the components of the robot onto the main system. In the testing phase the team tested the integration of all components to assembly to ensure the requirements of the project were met.

Key Features

The Solar Agribot project team was successfully able to develop and implement a robot that can begin to address the problems of farmers. The robot was successfully able to reach all of its minimum requirements.

1. The Solar Agribot successfully designed and built a rocker suspension to handle uneven and slippery terrain

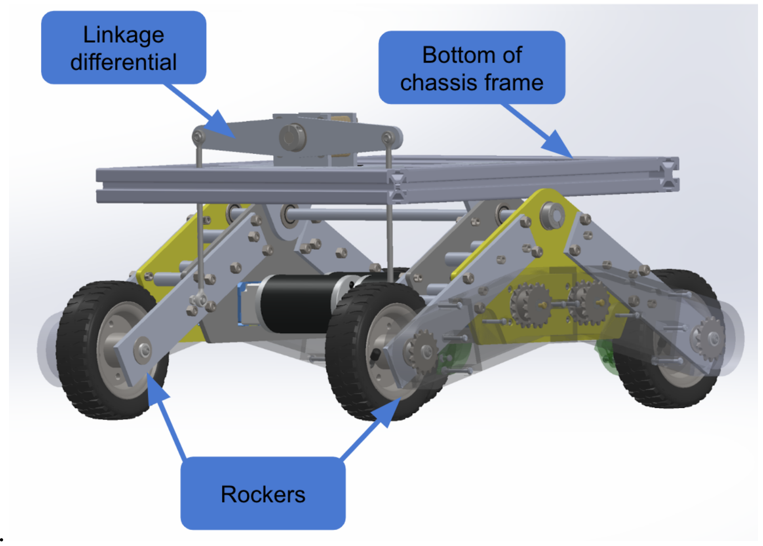


Figure E.1: Chassis

2. The Solar Agribot successfully drives, and skid steers on dirt terrain

3. The Solar Agribot team was successfully able to power and communicate with each onboard electronic system reliably

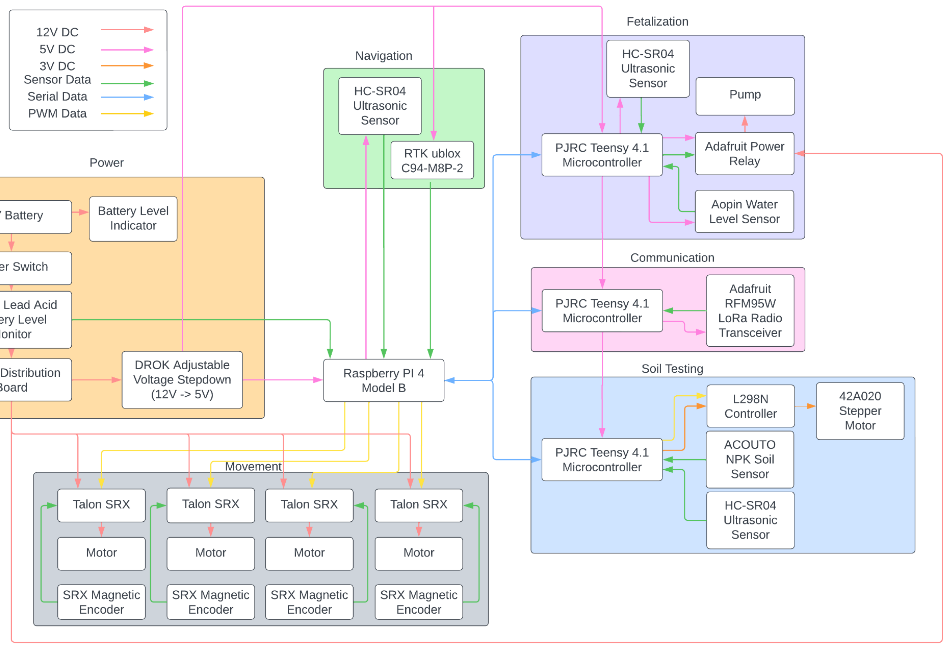


Figure E.2: Robot Electrical Design

4. The Solar Agribot Team was successfully able to develop a dashboard to improve the efficiency of robot testing and development

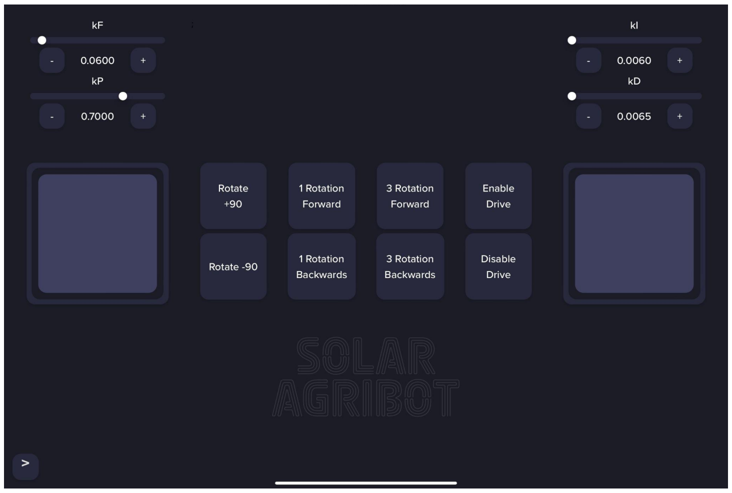
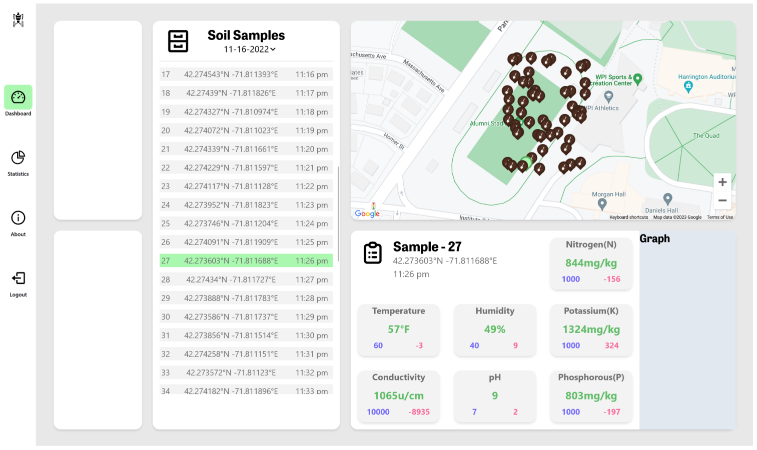


Figure E.3: Robot Controller

5. The Solar Agribot team was successful in designing and implementing a simple way to monitor and store the soil data collected by the robot



Soil Samples
11-16-2022

17	42.274543°N -71.811393°E	11:16 pm
18	42.27439°N -71.811826°E	11:17 pm
19	42.274327°N -71.810974°E	11:18 pm
20	42.274072°N -71.811023°E	11:19 pm
21	42.274339°N -71.811661°E	11:20 pm
22	42.274229°N -71.811597°E	11:21 pm
23	42.274117°N -71.811128°E	11:22 pm
24	42.273952°N -71.811823°E	11:23 pm
25	42.273746°N -71.811204°E	11:24 pm
26	42.274091°N -71.811909°E	11:25 pm
27	42.273603°N -71.811688°E	11:26 pm
28	42.27434°N -71.811727°E	11:27 pm
29	42.273888°N -71.811783°E	11:28 pm
30	42.273586°N -71.811737°E	11:29 pm
31	42.273856°N -71.811514°E	11:30 pm
32	42.274258°N -71.811151°E	11:31 pm
33	42.273572°N -71.81123°E	11:32 pm
34	42.274182°N -71.811896°E	11:33 pm

Sample - 27
42.273603°N -71.811688°E
11:26 pm

Nitrogen(N)	844mg/kg	1000	-156
Temperature	57°F	60	-3
Humidity	49%	40	9
Potassium(K)	1324mg/kg	1000	324
Conductivity	1065u/cm	10000	-8935
pH	9	7	2
Phosphorous(P)	803mg/kg	1000	-197

Figure E.4: Famer Dashboard

6. The Solar Agribot Team was successfully able to implement a solar charging system for the robot to recharge off of
7. The Solar Agribot Team was successfully able to implement radio communication with the hub

Implications and Recommendations:

1. Software:
   - Analyze data for accurate farm field mappings.
   - Evaluate hub server necessity and consider cellular chip integration.
   - Improve autonomous systems for enhanced performance.
2. Mechanical:
   - Test the robot on real farms to ensure its suitability for the intended environment.
   - Optimize robot size and weight with smaller components and lighter materials.

    - Waterproof main body and secure key components.
    - Modify design of soil probe to allow for greater range of motion and ground clearance
3. Electrical:
    - Fully integrate and test subsystems on the rover.
    - Improve wire management and implement additional sensors for better control.
    - Focus on signal processing, power analysis, and robust control designs.

Conclusion

The Solar Agribot Project has successfully reached its goals, creating a robot that could change the corn farming industry. With a specially designed chassis, the robot can easily move on uneven and slippery surfaces. The team has made sure the robot can communicate and power its systems reliably, while also offering a user-friendly dashboard for both robot control and soil data monitoring. The Solar Agribot can even recharge using solar energy, making it environmentally friendly and cost-effective. In short, the Solar Agribot Project has achieved its initial goals and has the potential to greatly improve farming efficiency and sustainability, benefiting farmers and the environment.

References

"Corn and Other Feed Grains." *USDA ERS - Corn and Other Feed Grains*, https://www.ers.usda.gov/topics/crops/corn-and-other-feed-grains/.

Goedde, Lutz, et al. "Agriculture's Connected Future: How Technology Can Yield New Growth." *McKinsey & Company*, McKinsey & Company, 11 Nov. 2020, https://www.mckinsey.com/industries/agriculture/our-insights/agricultures-connected-future-how-technology-can-yield-new-growth.

"Guide to Iowa Corn Planting." *Iowa State Extension and Outreach*, Iowa State, Nov. 2019, https://store.extension.iastate.edu/product/5475.

Hudgins, Nicholas, et al. *Solar Agribots.* : Worcester Polytechnic Institute, 2022. https://digital.wpi.edu/concern/student_works/hh63t003j?locale=en

Wright, Emily and Mark Licht. "Corn Row Spacing Considerations." *Corn Row Spacing Considerations | Integrated Crop Management*, Iowa State University, https://crops.extension.iastate.edu/cropnews/2018/03/corn-row-spacing-considerations#:~:text=In%20conclusion%2C%2020%2Dinch%20row,control%20and%20soil%20moisture%20retention.

Jeff. "(Infographic) the U.S. Farm Labor Shortage." *AgAmerica*, 28 June 2022, https://agamerica.com/blog/the-impact-of-the-farm-labor-shortage/.

"New Agriculture Technology in Modern Farming." *Plug and Play Tech Center*, https://www.plugandplaytechcenter.com/resources/new-agriculture-technology-modern-farming/.

Parker, Jacob, et al. *Nasa Lunabotics.* : Worcester Polytechnic Institute, 2022. https://digital.wpi.edu/concern/student_works/ms35tc734?locale=en

"Rover Wheels." *NASA*, NASA, https://mars.nasa.gov/mars2020/spacecraft/rover/wheels/.

"Small Robot Company." *Small Robot Company*, https://www.smallrobotcompany.com/.

Walsh, Cameron M, et al. *Beach Swarm - Phase Ii.* : Worcester Polytechnic Institute, 2021. https://digital.wpi.edu/concern/student_works/w6634668x?locale=en

Yang, J., Dong, M., & Ye, J. (2017, December 1). *A literature review of the rocker-bogie suspension for the Planetary Rover*. A Literature Review of the Rocker-Bogie Suspension for the Planetary Rover | Atlantis Press. Retrieved November 2, 2022, from https://www.atlantis-press.com/proceedings/anit-17/25889330/

"2.5 In. Cim Motor." *AndyMark, Inc*, 27 Apr. 2021, https://www.andymark.com/products/2-5-in-cim-motor.

nemo4all. "How to Calculate the Battery Percentage under Load...?" *Arduino Forum*, 23 July 2021, https://forum.arduino.cc/t/how-to-calculate-the-battery-percentage-under-load/888412.

"Talon Srx." *CTR Electronics*, https://store.ctr-electronics.com/talon-srx/.

Team, The Arduino. "Uno R3." *Arduino Documentation*, https://docs.arduino.cc/hardware/uno-rev3.

Chapter 1: Introduction

The agriculture industry is facing tremendous challenges due to rising costs of supplies as well as shortage of labor (Ku, 2022). Demand for food is rapidly growing as the world's population is growing at an alarming rate. By 2050, the world's population is projected to reach 9.7 billion people. In order to keep up with the rapidly increasing demands, the agriculture industry would have to provide a 70% increase of calories available for consumption (Goedde, Katz, Menard, Revellat, 2020). In order to avoid shortages, solutions are required as soon as possible. As a means of keeping up with the increasing demands, either an increase in labor force or an improvement in efficiency is required.

However, recent data gathered from the United States Department of Agriculture (USDA) shows the agriculture workforce is critically declining. This data reveals that from 1950 to 2000, the number of hired farmworkers has declined by 52%, while the number of family farmworkers has decreased by 73%. Some factors that have contributed to the concerning rate of the decline in the agriculture workforce are a decline in interest in agriculture, and inefficient agriculture labor programs as well as aging farm operators. Farmers ages 35 or younger only account for 9% of the workforce due to heavy physical demand as well as unequal work to life balance (Jeff, 2022). As the older farmers reach retirement age, the workforce is projected to drop further at an alarming rate. In order to make up for this decline, technological solutions need to be implemented to increase the efficiency of agriculture.

The goal of this project is to utilize technology to provide an efficient and affordable solution to the decreasing agricultural workforce. We propose the solution of a swarm of agricultural robots which will carry the repetitive and tedious tasks in farming. This swarm of agribots would be stored inside a base station (hub) that serves as a shelter, refuel, and resupply station. The swarm of robots would leave the hub in order to perform the required tasks by autonomous means and automatically return to the hub as needed. By utilizing renewable energy sources such as solar energy, this concept would provide a solution that is not only efficient, affordable and autonomous, but environmentally friendly as well.

Chapter 2: Background

2.1 Previous Agribots and Rovers

In order to develop a functioning agribot that addresses prominent needs of farmers, the team looked at the 2021-2022 agribot as well as other agribots and rovers. These examples provided insight into areas of improvement as well as viable components and functions.

The 2021-2022 agribot, shown in Figure 2.1, was designed for low profile crops, such as carrots, that the robot could roll over with wheels on either side of the crop row. The objectives of last year's agribot were to carry a heavy load, move linearly and skid turn, water a large area, navigate autonomously, conduct soil testing, and measure atmospheric data and recharge at the hub.
The chassis was composed of welded aluminum to meet the load goal. Only linear motion was executed, the robot lacked a full suspension system resulting in lack of skid turning. The agribot succeeded in being able to carry enough water to irrigate a portion of a carrot field, but a water pump with higher pressure output could have been implemented to spray the crops. Autonomous navigation was not fully realized as the GPS system was not fully functional. A crank slider linkage successfully allowed for its soil probe to be inserted into the ground. This probe was able to measure soil moisture, and could be adapted to measure NPK values.

The sensors on the hub were able to measure weather conditions and save them on an SD card. Docking and charging was performed, but not autonomously. This version of the hub requires the battery to be manually plugged into the solar panel for charging.

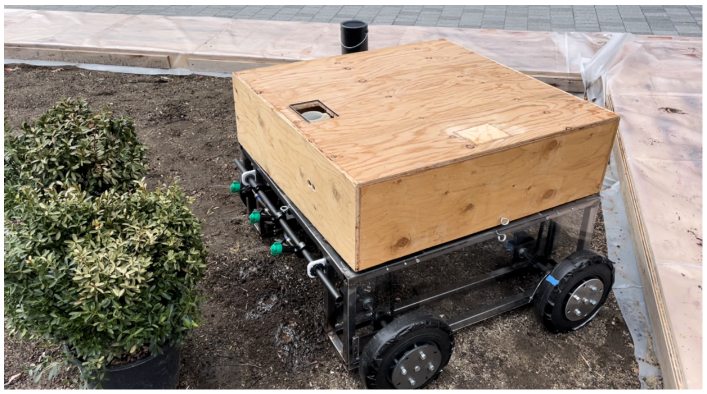


Figure 2.1: 2021-2022 Agribot

After learning about the successes and areas of improvement of the prior agribot, the team researched commercial agribots. The most significant agribot examined was Tom from Small Robot Company, which is shown in Figure 2.2. Similar to the 2021-2022 agribot, Tom was designed to roll over crop rows, and measures 147.8 x 129.5 x 114.5 cm. Tom's current functions include autonomous navigation, counting winter wheat plants, weed identification and location, providing and analyzing data for each individual plant, and spraying glyphosate, herbicide, or fertilizer. The ability to analyze large amounts of data and map out farms is going to be crucial for creating an efficient agribot.

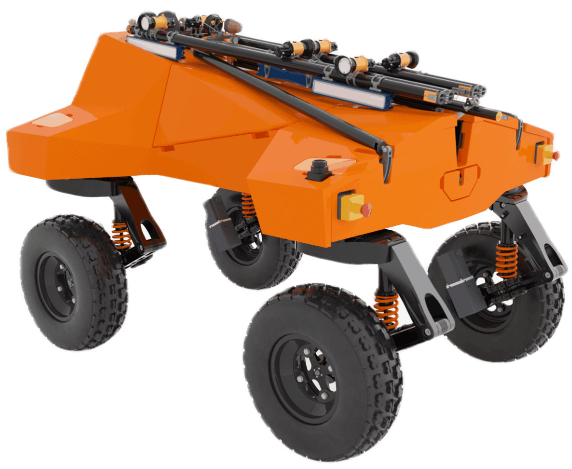


Figure 2.2: Tom from Small Robot Company (Small Robot Company)

To address the skid turning issues of the 2021-2022 agribot, research upon rugged terrain rovers was conducted. A majority of these robots incorporated a type of suspension. The most frequently used suspension in these robots were the rocker-bogie, rocker, and spring. The rocker-bogie type is depicted below in Figure 2.3, and is used on many rovers such as the 2020 Mars Perseverance rover. This suspension is composed of six wheels, a rocker, a bogie, and a differential. The rocker connects the front wheel to the bogie as well as the rest of the chassis and the differential. The bogie connects the middle and back wheels, and allows a rover to maintain stability while clearing obstacles. The differential connects the two different sides using a pivot or a gearbox, ensuring that all wheels are always in contact with the ground (NASA). The rocker suspension type is similar to the rocker-bogie type, but just lacks the bogie and only has 4 wheels. This design is a little less stable than the rocker-bogie, but reduces cost and weight. The final suspension type utilized in commercially available agribots is the spring suspension. This suspension is implemented in Tom in Figure 2.2 above.

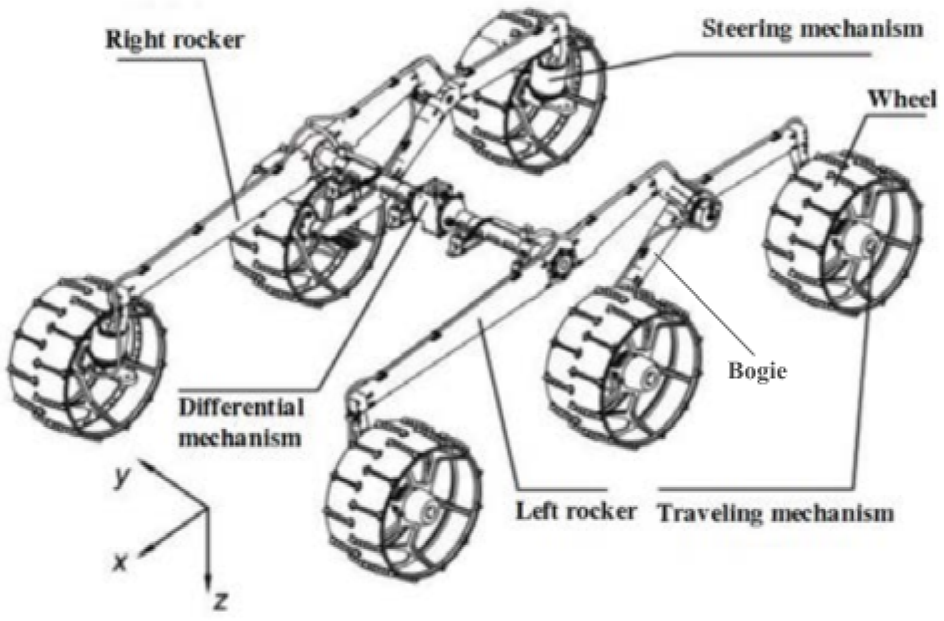


Figure 2.3: Rocker Bogie Suspension (Yang)

In addition to the commercial agribots and rovers, the team looked into examples of student-designed rovers that were able to successfully skid turn. One such example is the NASA Lunabotics 2021-2022 MQP, depicted in Figure 2.4. This rover was designed to mine regolith on the moon, avoid obstacles, and navigate with partial autonomy. It uses four wheel skid steering, which is what the 2021-2022 Solar Agribot MQP attempted. Unlike the agribot, the Lunabotics rover utilizes a rocker suspension to ensure that all four wheels are always on the ground, allowing for skid steering to function properly. This suspension enabled the rover to roll over obstacles up to 40 cm tall. Instead of having a differential gear box that similar rovers possess, this design incorporates a push-pull differential cable which takes up less space.

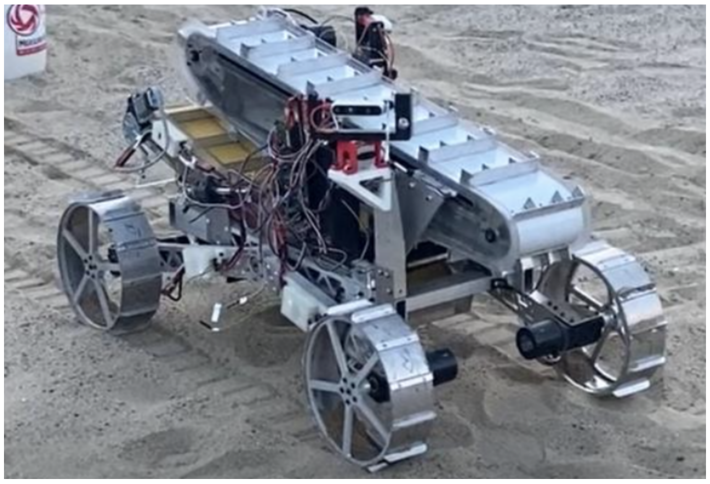


Figure 2.4: 2021-2022 Lunabotics Rover

Another example of successful skid turning that was studied is phase II of the Beach Swarm MQP shown in Figure 2.5. This robot also incorporated a rocker suspension system and was able to drive, turn, and reach a top speed of .62m/s on sand. Instead of a differential gear box, this design involves a 3-part linkage as a differential. This enabled the top plate of the chassis to stay parallel to the ground despite uneven terrain.

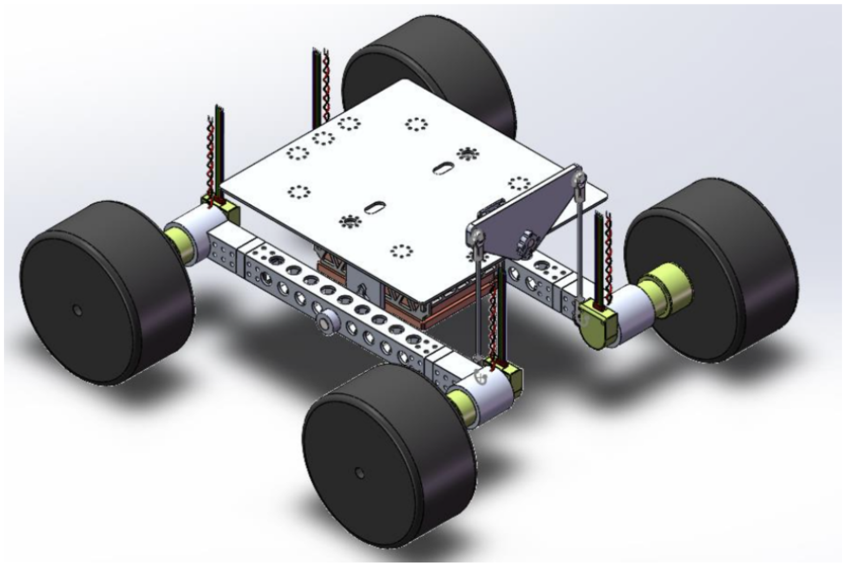


Figure 2.5: Beach Swarm Chassis

2.2 Statement of Work

The Solar Agribot team's goal is to design and build an autonomous robot that improves the efficiency of corn farming. Corn was chosen as the ideal crop due to it being "the most widely produced feed grain in the United States, accounting for more than 95 percent of total feed grain production and use" (USDA). Our design aims to collect data regarding soil viability in corn fields and administer fertilizer selectively to promote corn growth. Additionally we utilize a hub for the robot that provides protection from the elements in addition to charging the robot from solar power. The robot will be able to drive through corn fields without disturbing the agriculture, drive from and return to a docking hub and probe the soil around corn stalks to measure pH level, soil saturation, and NPK values. The robot will administer fertilizer if the soil probe measurements deem it necessary. The robot's hub will contain a solar panel to provide power to the system and charge the robot. The hub will also detect when there is any weather not suitable for the robot and be able to communicate to the robot that it should not leave the hub at that time.

The minimum requirements and reach goals for the robot and hub are outlined in Table 2.1. The team decided to focus on creating a small robot that must be able to navigate through a corn field without disturbing the agriculture. The size required for the robot to traverse a cornfield restricts the width to a maximum of 30 inches since yield is optimal around row spacing of 20-30 inches (Licht). The robot must be large enough to hold the probe and its associated motors, sensors and motors for navigation purposes, the battery, a tank for fertilizer, and a mechanism to distribute the fertilizer. The material of the bot is an

8020 aluminum frame, with steel axles. This improves the modular design of the bot. The bot is also charged by a battery charged by a solar panel that at minimum must plug directly into the robot.

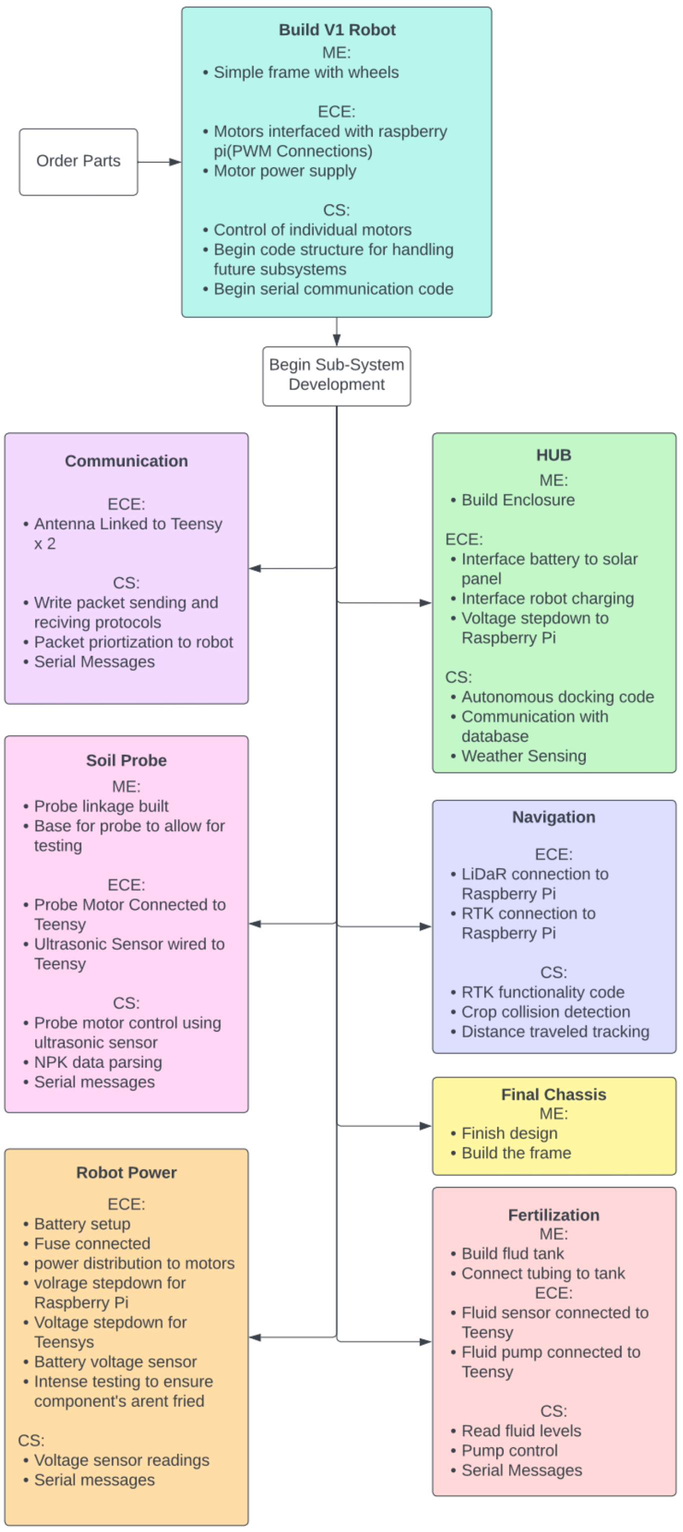
The team's minimum goal for data collection is to have a database where all the collected fertilization information and soil measurements are stored. As a reach goal, the team would like to have a working website where all the data is displayed in a map view.

Parameters	Minimum Requirements	Reach Goals
Robot Size	30" wide	20" wide
Robot Material	Aluminum/Steel	Carbon Fiber / PLA
Robot Charging	Plug in charging (male female prongs)	Wireless charging (metal contact)
Data Visualization	Website that displays where robot has taken measurement and fertilized	Website that includes live robot location with measurement and fertilization information
Robot Navigation	Remote controlled driving	Autonomous driving

Table 3.1: Project Parameters

2.3 Robot Subsections

To ensure we achieve the most functionality possible during our project, we have separated the robot into subsections (Figure 2.6). These subsections were built and tested separately from the robot, preventing multiple people from needing the robot for independent testing. Additionally, it helps organize the ME, ECE, and CS teams, informing each of what needs to be done at each step. Once a subsection has been built and tested it can be added to the current working robot to begin interfacing with the device.



Order Parts
Build V1 Robot
ME:
• Simple frame with wheels
ECE:
• Motors interfaced with raspberry pi(PWM Connections)
• Motor power supply
CS:
• Control of individual motors
• Begin code structure for handling future subsystems
• Begin serial communication code
Begin Sub-System Development
Communication
ECE:
• Antenna Linked to Teensy x 2
CS:
• Write packet sending and reciving protocols
• Packet priortization to robot
• Serial Messages
HUB
ME:
• Build Enclosure
ECE:
• Interface battery to solar panel
• Interface robot charging
• Voltage stepdown to Raspberry Pi
CS:
• Autonomous docking code
• Communication with database
• Weather Sensing
Soil Probe
ME:
• Probe linkage built
• Base for probe to allow for testing
ECE:
• Probe Motor Connected to Teensy
• Ultrasonic Sensor wired to Teensy
CS:
• Probe motor control using ultrasonic sensor
• NPK data parsing
• Serial messages
Navigation
ECE:
• LiDaR connection to Raspberry Pi
• RTK connection to Raspberry Pi
CS:
• RTK functionality code
• Crop collision detection
• Distance traveled tracking
Final Chassis
ME:
• Finish design
• Build the frame
Robot Power
ECE:
• Battery setup
• Fuse connected
• power distribution to motors
• volrage stepdown for Raspberry Pi
• Voltage stepdown for Teensys
• Battery voltage sensor
• Intense testing to ensure component's arent fried
CS:
• Voltage sensor readings
• Serial messages
Fertilization
ME:
• Build flud tank
• Connect tubing to tank
ECE:
• Fluid sensor connected to Teensy
• Fluid pump connected to Teensy
CS:
• Read fluid levels
• Pump control
• Serial Messages

Figure 2.6: Robot Subsections

Chapter 3: Design and Development

3.1.0 System Overview

The Solar Agribot comprises three main sections, Electrical, Mechanical, and Software. The mechanical section of the robot is responsible for building a chassis for the robot that will handle the weight of all the components on board while maintaining good wheel contact. The Electrical section is responsible for designing and implementing each sub system and ensuring that all sensors and motors are powered properly. The Software team is responsible for interfacing with each sub system and creating a way to manage and visualize the data collected from the robot.

3.1.1 Mechanical Requirements

Our robot's mechanical system requirements were to remain confined to a 30 inch width in order to allow the robot to pass between corn rows without disturbing the crop. The frame must be easy to prototype, to allow for ease of modification and future adaptations. Additionally, the ratio of the length to the width of the wheel contact surfaces should be considered and remain close to 1:1 to reduce the torque required for skid steering. In order to ensure efficient driving through optimal wheel contact over uneven ground, a suspension system must be used. Additionally, the suspension system used must be able to withstand the forces of skid steering. The transmission system must also be able to provide the torque required for skid steering, as well as fit onto the robot.

For our robot's suspension we chose a rocker suspension system. Our transmission system consists of CIM motors with 48:1 gearboxes connected to a chain drive system. This configuration allows the motors to provide the necessary torque while fitting within the 30 inch maximum width.

3.1.2 Electrical Requirements

Our robot's electrical system allows the mechanical parts to function. These parts include the wheels, soil probe, and fertilizer spray nozzles. Additional control was necessary for non-mechanical parts, such as hub communication and robot navigation. Taking all of these requirements for our electrical system, we organized our complex design into smaller sub-sections. These sub-sections include: power, movement, navigation, fertilization, communication, and soil testing. To control each of these systems, they will interface with a Raspberry Pi 4.

Robot navigation and system control will occur on the Raspberry Pi 4. Additionally, the fertilization, communication, and soil testing sub-sections are all controlled by a separate microprocessor, the Arduino Teensy 4.1. These microprocessors will communicate with the Raspberry Pi through a serial connection, allowing for simple messaging between the controllers. The purpose of this separation is to allow for the building and testing of each sub-system independently from the robot. Once a system has been completed, it can be easily added to the robot with a power line and a serial connection. Power is supplied to the Raspberry Pi and each of the previously mentioned subsystems by the power sub-system. This controls the flow of electricity through the robot, ensuring safe amperage and correct voltages to all components. All of these systems together make up our entire electrical design which can be seen in Figure 3.1.

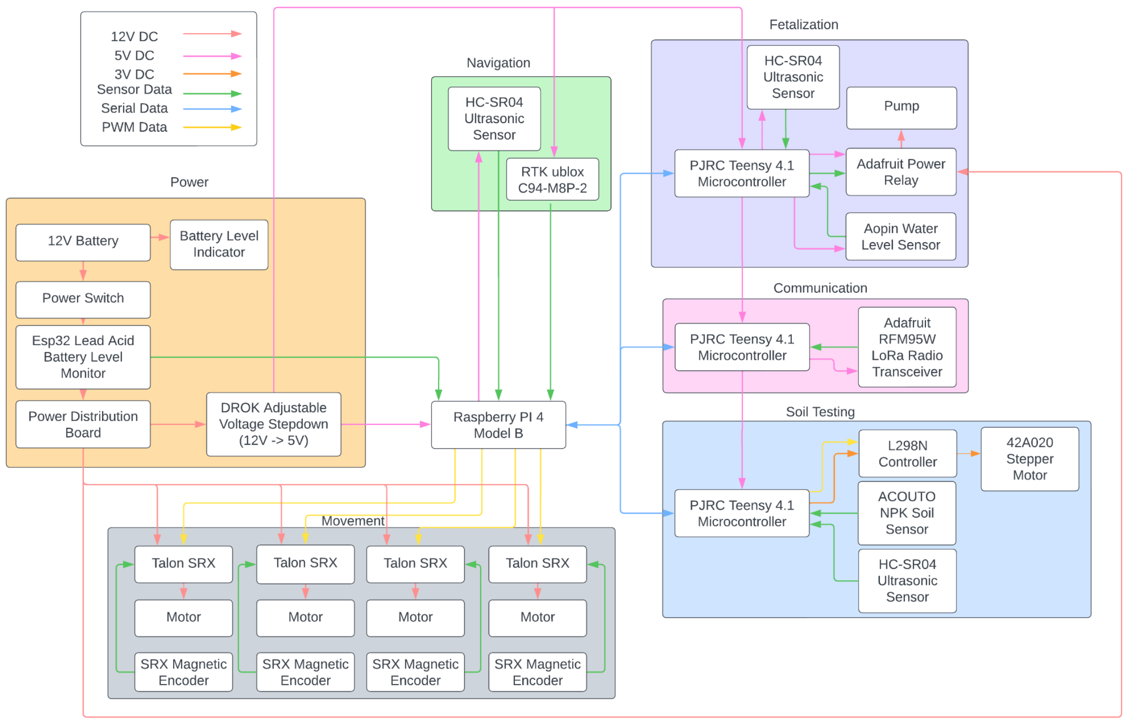


Figure 3.1: Robot Electrical Flow

3.1.2.1 Power in the Robot

Our project goal is to utilize free components provided by the university to design and build a solar powered modular system. The power is first collected at the hub using a solar panel to charge a lead acid battery. The battery can charge the two 17.2 Ah 12 V batteries connected in parallel in order to power the agribot and all of its components. The batteries were then connected to four 40A circuit breakers on power distribution for each motor. This component provided protection of all of our components by ensuring that the current operating in one motor does not exceed 40A. The power monitor design based on ESP32 (the hub and the rover module that communicates over WIFI) is connected to the power distribution board in order to check the battery consumption of the rover battery.

3.1.2.2 Navigation

Navigation was segmented into two parts, essential attributes and additional features. We started off by using an encoder as our main tool for navigation by taking the number of wheel rotations and calculating the resultant distance. Once our first goal was accomplished, we started configuring the rover such that it follows a defined path following RTK GPS coordinates (yielding a displacement between checkpoints) in synchronization with a tilt-compensated IMU magnetometer sensor(compass heading). In a demo, we mapped several checkpoints with the RTK survey mode. In traversing across the terrain, we rotate to the desired pose first then move a displacement calculated from the difference between longitude and latitude of two points.

3.1.2.3 Soil Testing

Our goal was to test the soil NPK values by sampling soil data row by row. In order to get consistent results, it was crucial for the testing probe to penetrate the soil at the same depth. Hence, it was necessary for an Ultrasonic sensor to be attached to the probe vertically to get consistent depth. In addition, using a linear actuator, it was possible to penetrate the soil at different compaction levels,with some errors at different slopes. By collecting pH, moisture and NPK values at different spots, these values were mapped out by rows and columns.

It was important to us to test the soil for NPK values, temperature, humidity, conductivity as well as pH. Research was made in order to find the best sensor for our applications. Our team chose to use the Taidacent RS485 Soil NPK PH Sensor Probe NPK Sensor shown in Figure 3.2. This sensor was able to test the soil for everything that we needed, and was digital as well, making it the best fit for our project. In addition, a current sensor will be added in order to get feedback from the unit step response spike (in a scenario where the probe hits a rock).

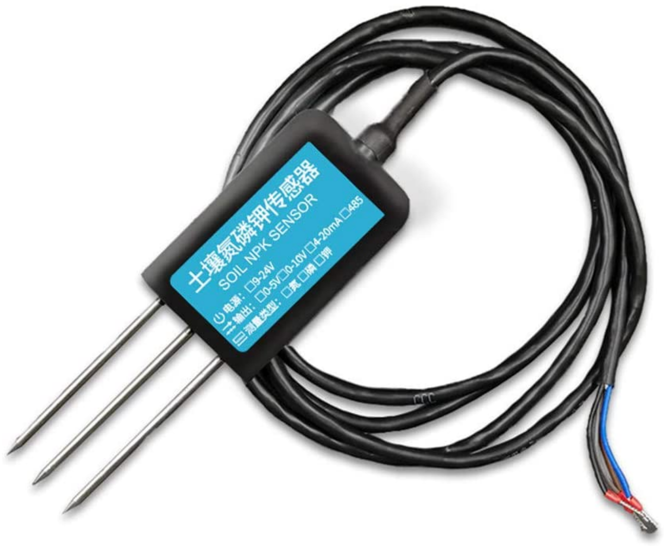


Figure 3.2: NPK Sensor

3.1.2.4 Fertilization

Our robot also has a fertilization system. Once we analyze which parts of the field are nutrient-deficient through the detection data with the NPK sensor, the robot is able to supplement these areas with the liquid fertilizer stored in its half-gallon tank by dispersing them with the water pump shown in Figure 3.3. When the liquid fertilizer in the tank is depleted, the robot will return to the hub for replenishment.

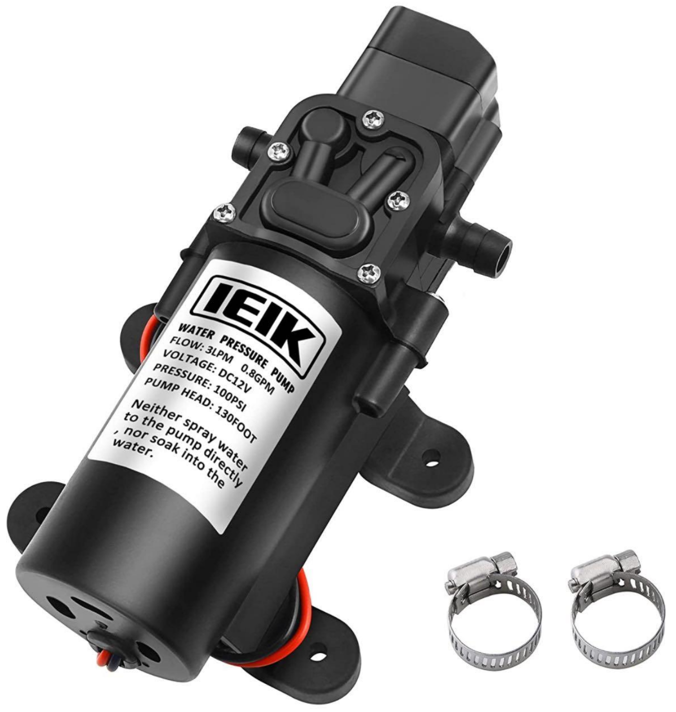


Figure 3.3: Fertilizer Controller-Water Pump

3.1.3 Software Requirements

To build a successful robot our software portions must satisfy a vast amount of goals. These include the abilities to drive, autonomously navigate, take soil samples, fertilizer, and visualize the data for the farmer. Our robot needs several software processes to take place across multiple devices in order to perform all of our robot’s desired goals. For our system we will need four computers, and four microcontrollers to host all of our services, each one performing its own individual actions. These four computers will be called: The Hub, The Web Server, The Website, and The Rover. Each of these devices will communicate with each other consistently in order to ensure a sync of data between devices (Figure 3.4).

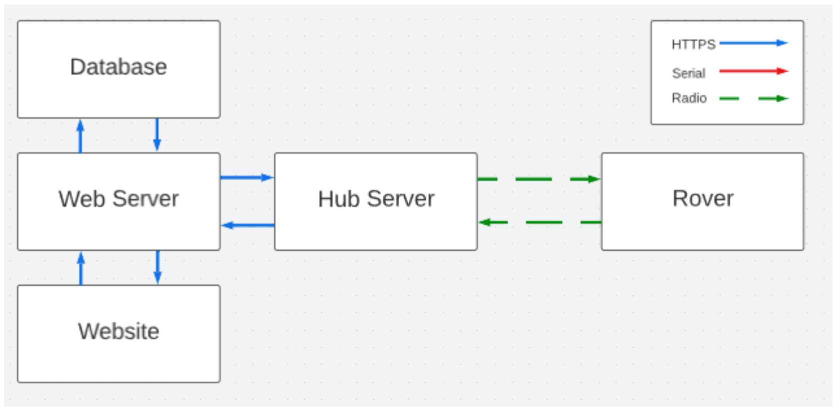


Figure 3.4: Sync of Data

The rover section is responsible for sending soil sample information and listening for commands. The Hub Server is responsible for listening for the rovers messages and then sending them to the web server, additionally it pings the web server to check for potential scheduling updates. The Web Server is responsible for providing an API gateway for the hub server to send data and for the website to retrieve data. The Webs Server needs to properly store and retrieve this data when requested so the web server must also communicate with a database.

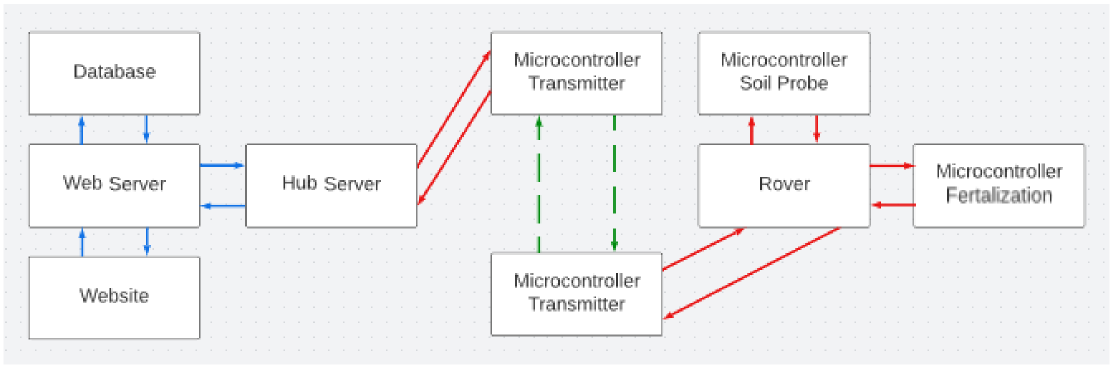

Database
Web Server
Website
Hub Server
Microcontroller Transmitter
Microcontroller Soil Probe
Rover
Microcontroller Fertalization
Microcontroller Transmitter

Figure 3.5: System Interactions

The interactions between all of the computers and microcontrollers can be seen above in Figure 3.5. The transmitter microcontrollers are responsible for creating a long distance information bridge between the rover and the hub. The rover is also connected to the soil probe microcontroller and the fertilization microcontroller over serial. The soil probe microcontroller is responsible for actuating the soil probe and returning the N, P, and K values measured. The fertilization microcontroller is responsible for turning the pump off and on while also monitoring the amount of fertilizer remaining in the tank..

3.2.0 Conceptual designs and prototyping/modeling feasibility studies

In order to have a fully functioning robot there are several electrical, mechanical, and software systems which need to be designed and prototyped before a final implementation can proceed. These sections are carefully designed to ensure time is not wasted during integration. Additionally this allows for time to discover any flaws which might exist within the system.

3.2.1 Chassis

The chassis, suspension, and drivetrain were paid particular attention by the mechanical team due to the main goal of building a rover that could skid steer. Learning from last year, the objectives of the chassis and drivetrain design were to make it lighter weight, incorporate a suspension, and have more than the estimated minimum torque. The first objective was accomplished by setting our maximum rover width at 30 inches, limiting the size, and therefore weight, of our rover. The other two objectives and overall design process are discussed in depth in the following sections.

3.2.1.1 Chassis Preliminary Design

The 2021-2022 agribot chassis frame was composed of welded aluminum, which aided in the goal of carrying a heavy load. Since the 2022-2023 agribot has no such objective, lighter weight materials and easier fabrication methods were considered. A mixture of an aluminum 80/20 frame with laser cut acrylic and plywood panels were chosen as there was 80/20 freely available to us.

The drivetrain of the previous agribot was a four wheel drive skid steering design. This drivetrain was chosen due to its simplicity, compactness, and low cost. Based on our research, this type of drivetrain is common for agribots and other rovers, making it a realistic option.

To compensate for the poor skid turning of the 2021-2022 agribot, the team chose to incorporate a suspension system. Based on other rovers that utilize skid steering such as lunar and Mars rovers, a rocker or rocker bogie suspension was considered. Due to the size constraints of our rover, a four-wheel rocker suspension was the best choice. Most rocker suspensions incorporate a differential bar or gear box to maintain contact with the ground. The team chose to use a 3-part linkage differential inspired by the Beach Swarm MQP due to its simpler design.

The team did research on different kinds of wheels that work well in outdoor environments. From our research, we determined that we needed wheels that have deep, large treads to traverse the farm land so the robot doesn’t lose traction. The team had originally discussed a possibility of manufacturing our own wheels; however, we found wheels online that would be more cost effective towards our project. The team decided that wheels with a diameter of six inches would be ideal for the robot's design, and we found pneumatic wheels that can hold up to 150 lbs. Figure 3.6 shows the wheels chosen this year compared to the wheels that were chosen last year, displaying how the new wheels have a larger width to diameter ratio as well as deeper treads. These chassis design and material choices resulted in the preliminary design shown below in Figure 3.7.

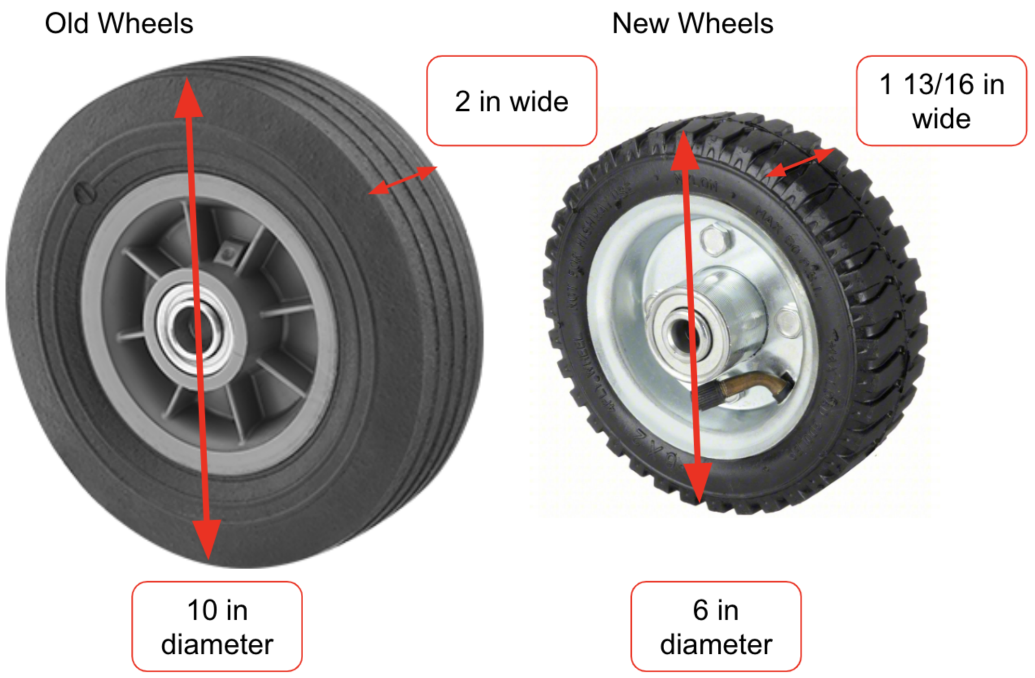


Figure 3.6: Old vs. New Wheels

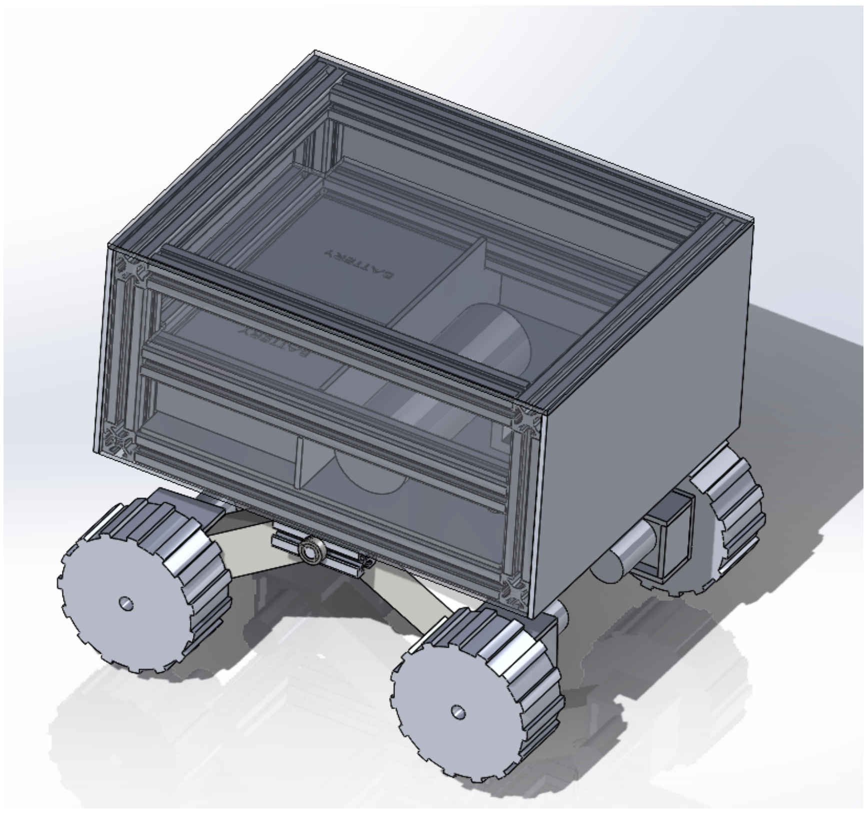


Figure 3.7: Preliminary Chassis Design

3.2.1.2 Torque Calculations

The chassis design informs the torque necessary to drive the robot, and therefore the motor choice and the possible necessity of a gearbox. The following torque analysis estimates the minimum torque required and is based on estimated weight and coefficient of friction between the tires and soil. This analysis is based on the assumptions that the rover is skid turning on even terrain. It also assumes that the center of mass is at the centroid of the rover.

Our process began with obtaining the coefficient of friction values of the wheels. We purchased four 6-inch diameter wheels; bolted them to a wooden frame, assuring any rotary movement was restricted; then weighed the model. Using a force gauge, the wooden model was dragged through the soil forwards and sideways; each was tested by dragging the model in orthonormal and 45 degree angle directions. By dragging the model, the gauge also recorded the force required to move the model from its stationary position. The three steps described can be found in Figure 3.8 below. We determined our friction values by dividing our force gauge measurements, by the measured weight of the figure. Through our testing we obtained both our frontal and lateral friction values μ_R and μ_T. When calculating the resultant force for skid steering, we plugged in μ_R.

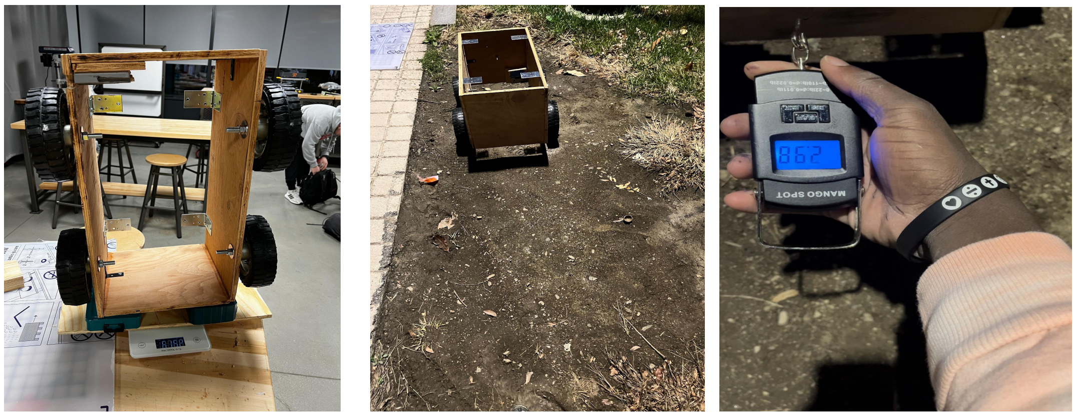

(a) (b) (c)

Figure 3.8: Wooden figure with mounted frame weighed (a), placed in testing environment (b), and pulled with a force gauge (c)

The free-body diagram below (Figure 3.9) depicts a top view of the chassis (pink) and wheels (blue), and includes the factors and forces which influence torque. This free-body diagram represents the external forces that act on the tires when the rover turns to the left. The torque being solved for (T) is the minimum torque for one motor. Each tire has a friction force acting on it from the soil (R), which is broken up into resistance and traction forces (F_r, F_t). The measured width (w) from wheel center to wheel center is 16.5 in. The measured length (l) from wheel center to wheel center is 15.45 in. The estimated weight informs the friction force (R), and is estimated to be less than 100 lbs. The coefficient of friction between the tires and soil (μ) will be estimated once we manufacture our tires and run tests in soil. The wheel radius (r) also determines torque and is equal to 3 in.

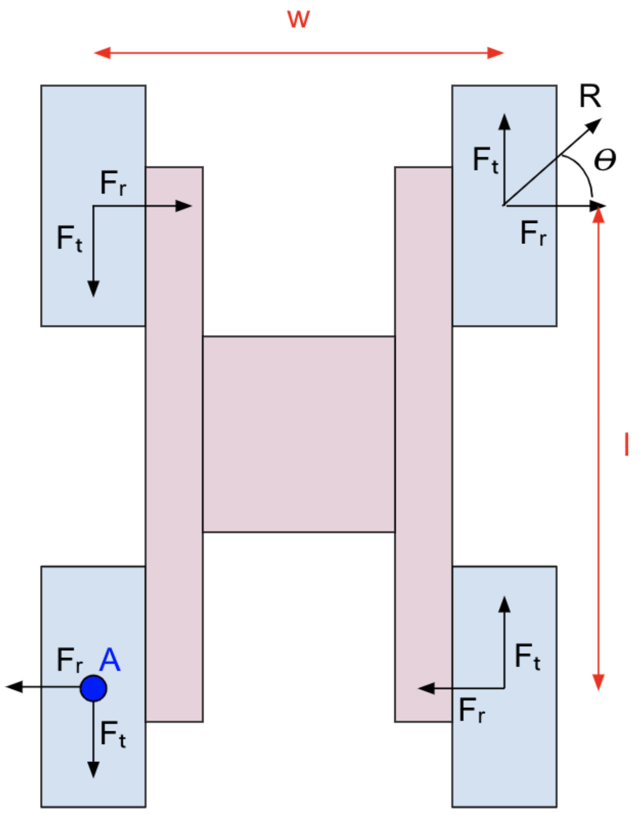


Figure 3.9: Torque Calculations FBD

To find the torque required for one wheel, the sum of the moments about point A is taken. Counter-clockwise is assumed to be the positive direction.

$$\Sigma M_A = 2(F_t * w) - 2(F_r * l) = 0$$

(Equation 3.1)

$$\frac{F_t}{F_r} = \frac{w}{l}$$

(Equation 3.2)

Next, the resultant force magnitude is calculated.

$$R = \frac{1}{4} * \mu * W$$

(Equation 3.3)

From here, theta is determined and used to find the traction force.

$$\theta = tan^{-1}(\frac{F_t}{F_r}) = tan^{-1}(\frac{w}{l})$$

(Equation 3.4)

$$F_t = R * sin(\theta)$$

(Equation 3.5)

Finally, the traction torque is calculated based on the traction force and wheel radius.

$$T = F_t * r$$

(Equation 3.6)

We determined our friction values by dividing our force gauge measurements, by the measured weight of the figure. Through our testing we determined both our frontal and lateral friction values μ_R and μ_T, being .59 and .57 respectively. When calculating the resultant force for skid steering, we plugged in μ_R to get our resultant force, then applied the ratio of the lengths of the width and length to obtain a traction force of 12.3 lb. From there, our required torque was calculated to be 36.9 in-lbf (4.17 N-m).

We were provided CIM motors with an operating torque of .32 N-m, meaning gearboxes were required in order to obtain the necessary torque calculated above. Our calculations revealed that by dividing our required force by the motors output torque, the necessary gear reduction was around 13:1. However, taking losses due to friction in the gearbox into account, as well as any unforeseen forms of loss throughout the transmission sequence, this value was divided by an efficiency value of $1/0.8^3$ being 0.512. The result was a final calculated reduction of about 26:1. Upon selecting the gearboxes, 48:1 gearboxes were the closest available option to our calculated value, so those were selected. The higher gear reduction ratio provides the advantage of nearly double the required amount, taking friction into account.

3.2.1.3 Rocker Redesign

To accommodate the motors we were given and the gearboxes we would need, the rocker design was changed. Our original rocker layout involved the motors being connected to the wheels and driving them directly. This design was not feasible since it had the wheels cantilevered making driving them more difficult. There were also concerns about the motors being cantilevered into the center of the chassis and potentially colliding with the motors on the other rocker.

In an attempt to solve the aforementioned issues, the second iteration of the rocker, shown in Figure 3.10 below, was developed. This design utilizes bevel gears to drive the wheels indirectly. This allows for the motors and gearboxes to be in a space-saving configuration. The output shafts of the gearboxes are hexagonal, so we added hexagonal shafts to drive the wheels. Since the wheel bores are round, we designed inserts made of 3D printed PLA/carbon fiber that will glue into the wheels and have a hexagonal input for the shafts. Other features of this design include quarter inch aluminum rectangular plates that support the wheels on either side so they are not cantilevered, quarter inch gusset plates to connect and support the rectangular plates and are either 3D printed PLA/carbon steel, ¼-20 bolts that connect the rectangular plates, and aluminum standoffs that separate the plates.

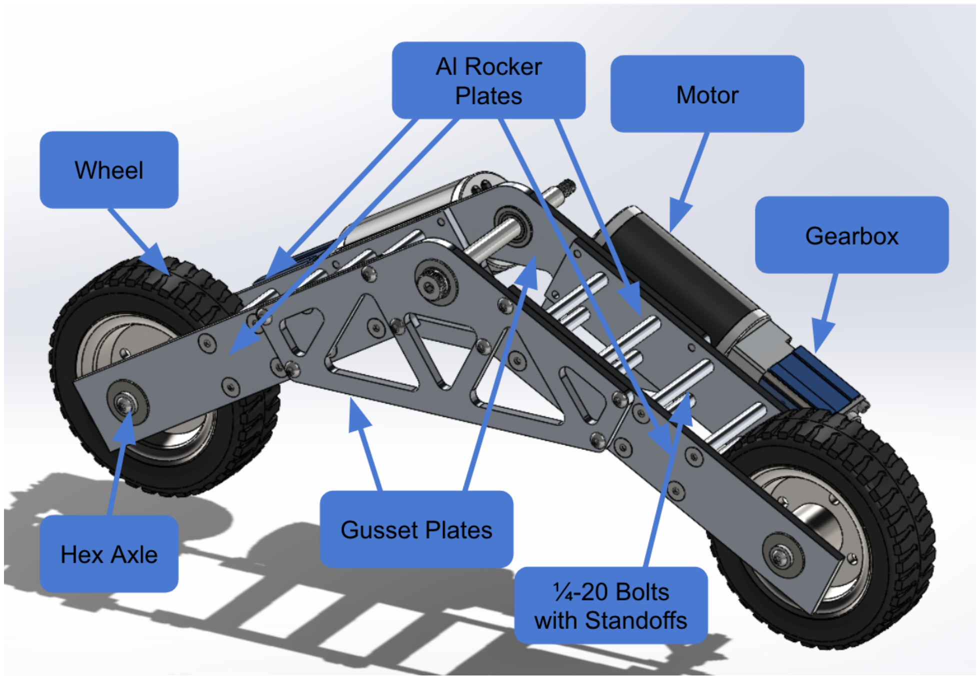


Figure 3.10: Second Rocker Iteration

We realized that the bevel gears could easily have alignment issues, and that the gearbox may not be able to support the radial loads that the bevel gears would apply to it, so we looked for alternative solutions. We first researched worm gearboxes, but quickly determined that they did not meet our space or budget requirements. We also looked into right angle gearboxes, but that did not solve our issue of radial loads. We concluded that a right angle drive was not ideal, and brainstormed ways to support the motors and gearboxes within our space constraints. We revisited the previous team's transmission, and decided to try using a chain and sprocket system so the gearboxes and motors would not have to be cantilevered from the wheels. This resulted in the third iteration of our rocker shown in Figure 3.11 below.

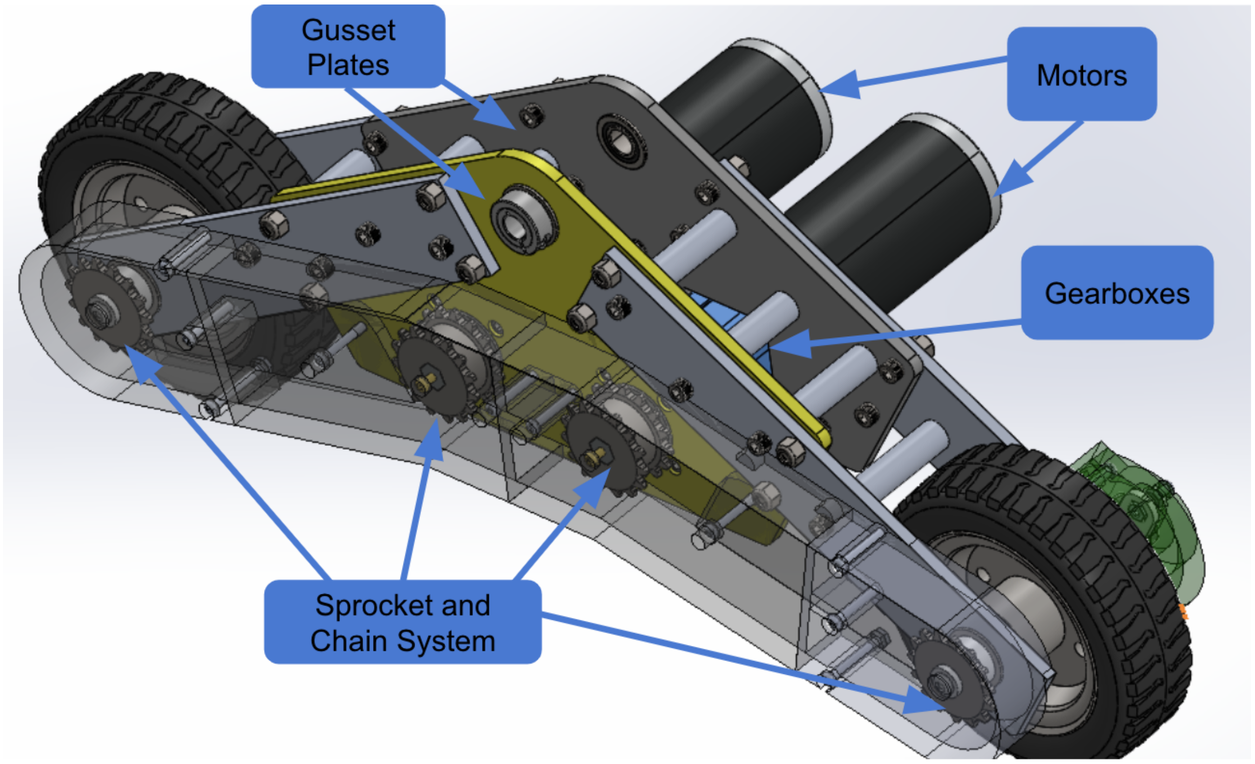


Figure 3.11: Third Rocker Iteration

This version of the rocker drives the wheels with the same hex shafts but has hex input sprockets on the gearboxes and wheel shafts. We chose 35 series chains due to their working strength rating of 480 lbs. We also added a chain connecting the two gearboxes on either rocker to increase the torque for one wheel if it were to get stuck. The sprockets are 15 tooth because of the limited availability of hex input sprockets. To determine the chain length that we needed for the transmission we used Equation 3.7 below:

$$\frac{\text{Length}}{p} = 2\cdot\frac{cc}{p} + \frac{N_1 + N_2}{2} + \frac{(N_2 - N_1)^2}{4\cdot\pi^2\cdot\frac{cc}{p}}$$

Equation 3.7: Chain Length Equation

We positioned the sprockets according to what chain length would allow us to not need a half-link to ensure the structural integrity of the chain. The calculations concluded that we would need 56 links for the 4 long chains and 38 links for the two short chains.The gearboxes are mounted to the front gussets and are supported in the back by the other gusset. We changed the standoffs to wooden ones with a larger diameter for easier machining.

We had to design hex wheel inserts for all four wheels on the robot as well. The wheels we purchased were intended for round shafts, but we needed to use hex shafts for our gearbox. We made multiple 3D printed PLA prototypes to test different lengths and press fit tolerances before printing the final model out of a nylon-carbon fiber mixed filament on the Markforged printers at WPI.

3.2.1.4 Final Chassis Design

After the rocker subassembly design was finalized, the rest of the chassis design was also refined. The first thing changed was the way the rocker is mounted to the chassis frame. We decided to have both rockers connected to one large axle that spans the width of the robot. The axle plate shown in Figure 3.12 was designed to screw into the 80/20 frame and connect to the axle with bearings. We chose to use shaft collars to prevent the rockers and axle from sliding in the axial direction relative to each other and the chassis frame as shown in Figure 3.13.

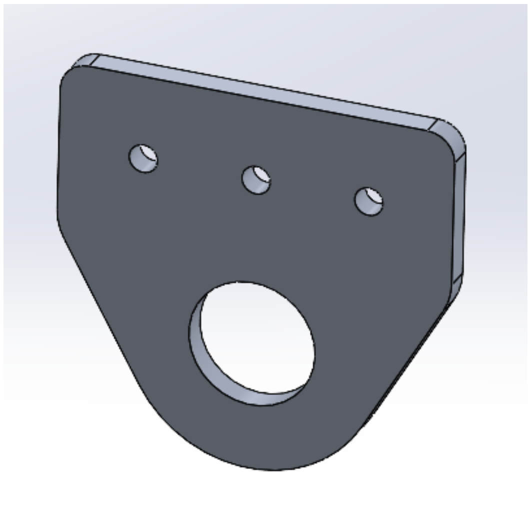


Figure 3.12: Axle Plate

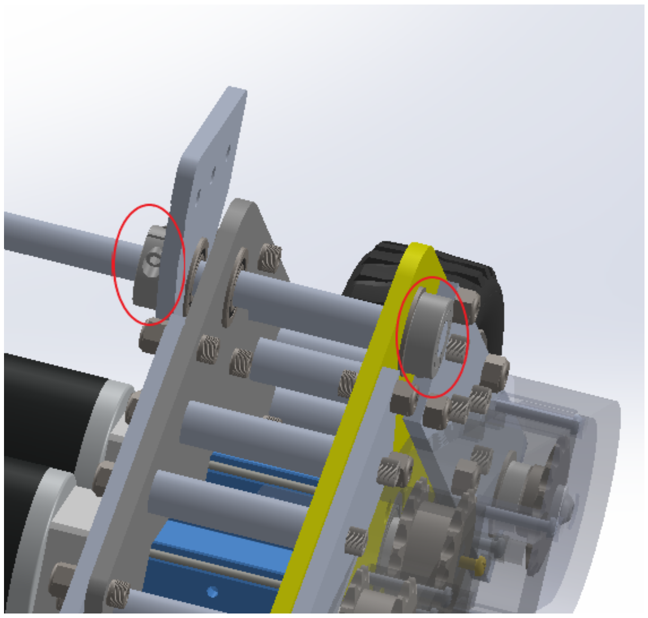


Figure 3.13: Shaft Collars Securing Rocker and Axle

Another subsystem that was redesigned was the differential. A design with two plates that screw to the 80/20 and to each other with standoffs was created to connect the differential linkage to the chassis frame. This subsystem is shown in Figure 3.14. Internally threaded rods and ball joints were sourced from Mcmaster and added to the linkage design. The ball joints connect to the rockers and the differential linkage plate and allow the plate to rotate. The differential linkage plate rotates about a shaft that is supported by the differential mounting plates, and is prevented from moving axially by shaft collars. The final suspension design is displayed in Figure 3.15 below.

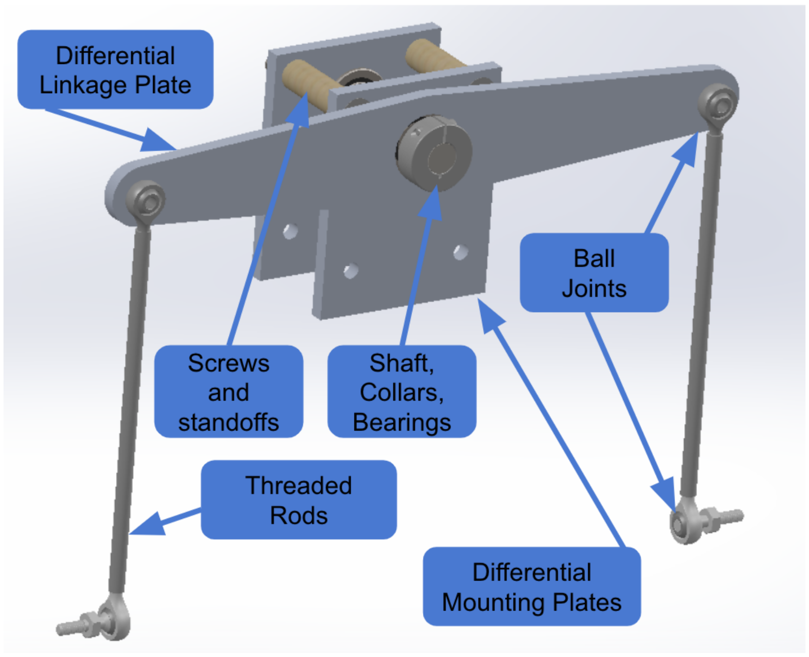


Figure 3.14: Differential Subsystem

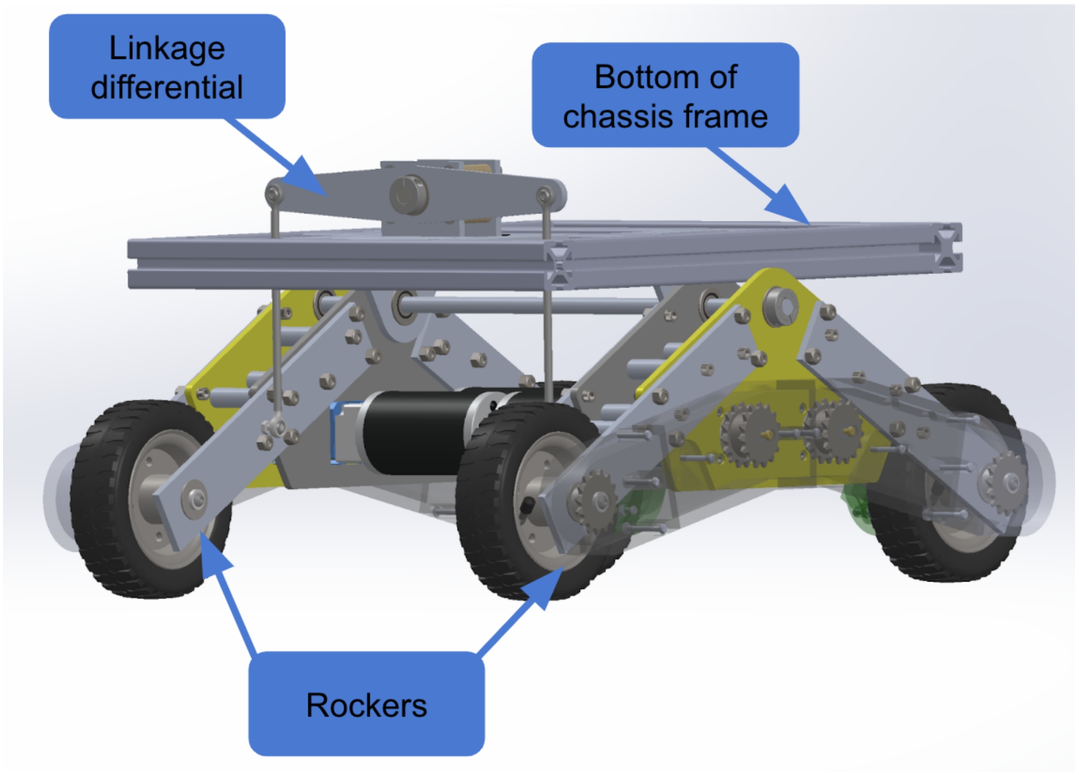


Figure 3.15: Final Suspension Design

3.2.1.5 Manufacturing and Assembly of Final Chassis Prototype

Once the chassis design was finalized and deemed feasible through the use of SolidWorks and Ansys models, manufacturing of a physical prototype began. The development of a full scale prototype was chosen so all of the subsystems could be tested as accurately as possible. The only mechanical component that was simplified for the purpose of prototyping was the gusset plates shown below in Figure 3.16. Although they were originally intended to be waterjet cut or produced using a CNC router, budget, machine availability, and time limitations led to the decision to 3D print them out of PLA for the prototype. The waterjet cutter, which was utilized for other parts that will be discussed, could not be used for the gussets because its cutting area is only 12 x 12 inches. The rest of the chassis components were made of the intended materials.

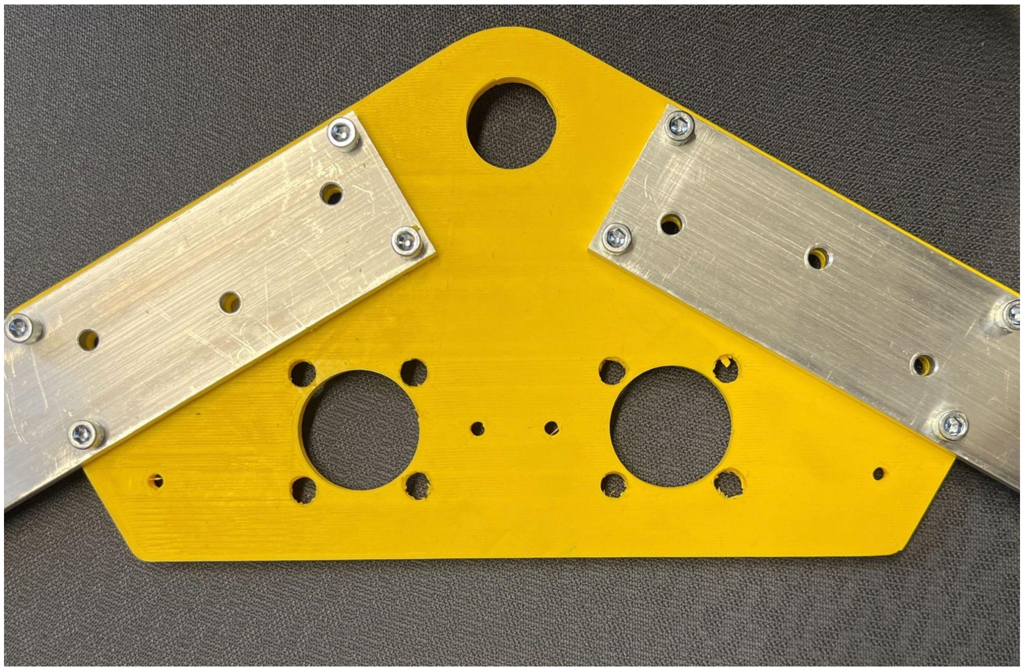


Figure 3.16: 3D Printed Gusset Plate

The first parts produced were the 80/20 pieces for the chassis frame. These pieces were cut to length using an aluminum saw, and the frame was assembled together using corner brackets as shown in Figure 3.17. This method aided in the ease of manufacturing since it eliminates the need for hole operations on the 80/20 pieces.

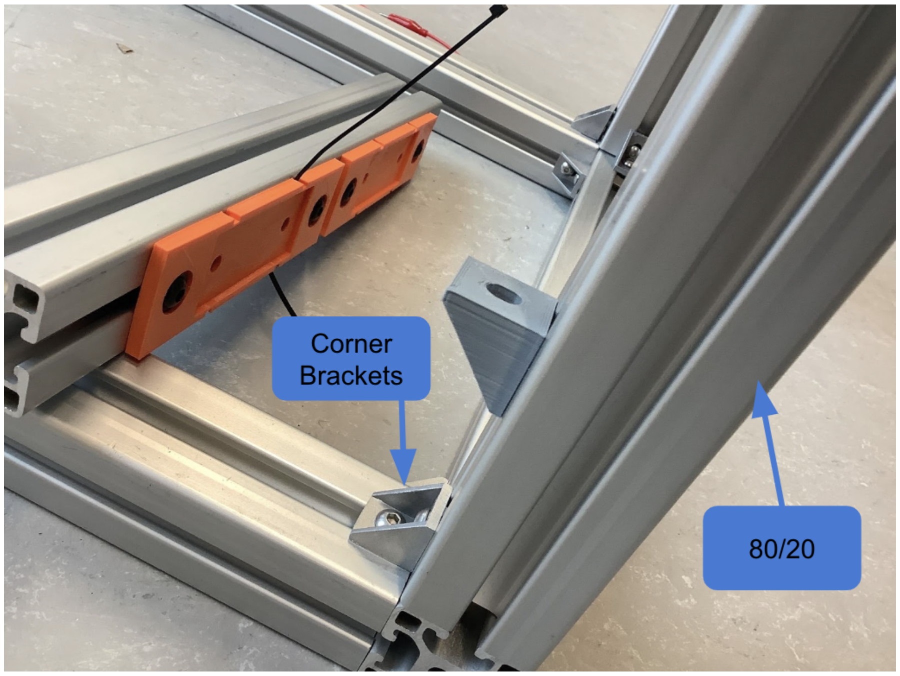


Figure 3.17: 80/20 Fastening Method

Next, the differential linkage plate, and axle mounting plates were cut out of a piece of ¼ in thick aluminum as shown in Figure 3.18 using the waterjet cutter in PracticePoint, a health research facility on campus which has a machine shop. All three of these components contain bearing press fit holes, so they were downsized for the waterjet cutter due to the unknown kerf value. A secondary CNC milling operation was performed to bring the bearing holes to the proper press fit diameters.

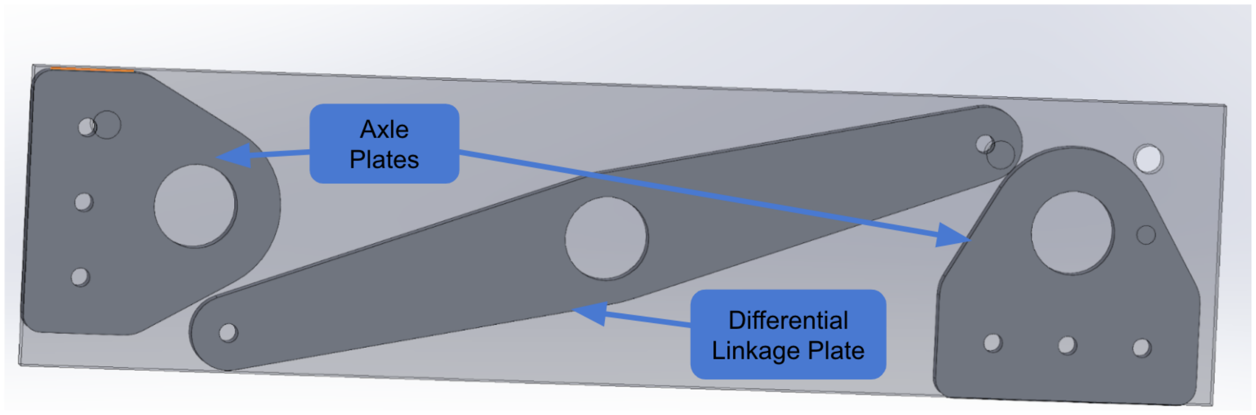


Figure 3.18: Axle Plates and Differential Linkage Plate Cutting Configuration

After the waterjet cut parts were finished, the differential mounting plates were made out of ¼ in thick aluminum bar using a CNC mill. The CAM file created for this part is shown in Figure 3.19. An attempt was made at fabricating the rocker plates shown in Figure 3.20 using a CNC mill. 2 in wide bars of ¼ in thick aluminum were purchased so the plates would only need to be brought to length and have the holes drilled. The CAM toolpath created in Fusion 360 for this operation is shown in Figure 3.21. The vice jaw widths that were available impacted the feasibility of this operation. Vice jaws that were too small had to be used which caused significant deflection during drilling operations. One rocker plate was successfully completed using this method, but the holes on the second one were not drilled all the way through due to deflection. To avoid further complications, it was decided to waterjet cut the remaining rocker plates as well. They also required the same secondary operations to obtain the proper press fit diameters.

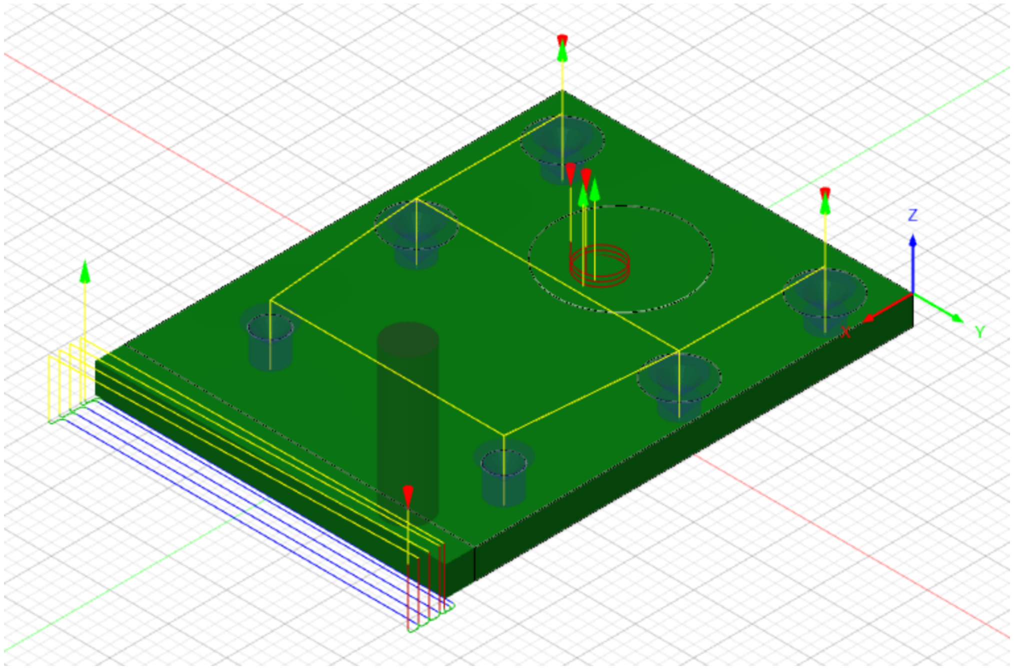


Figure 3.19: Differential Mounting Plate CAM Tool Path

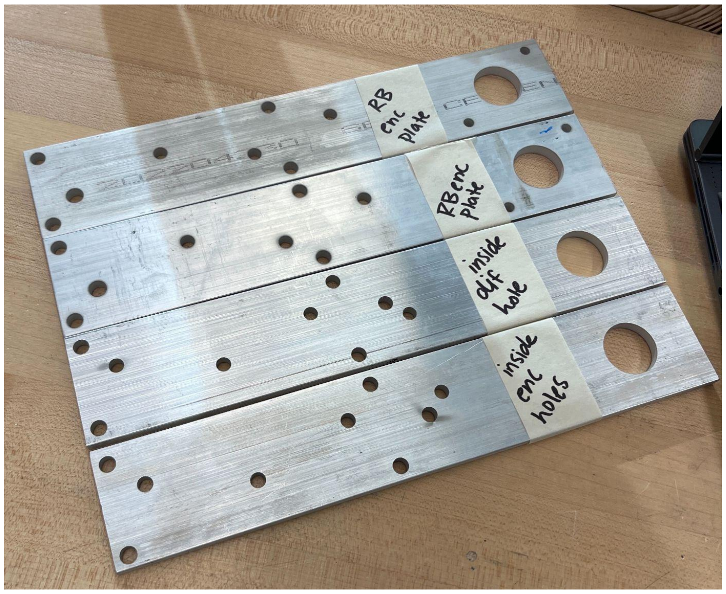


Figure 3.20: Rocker Plates

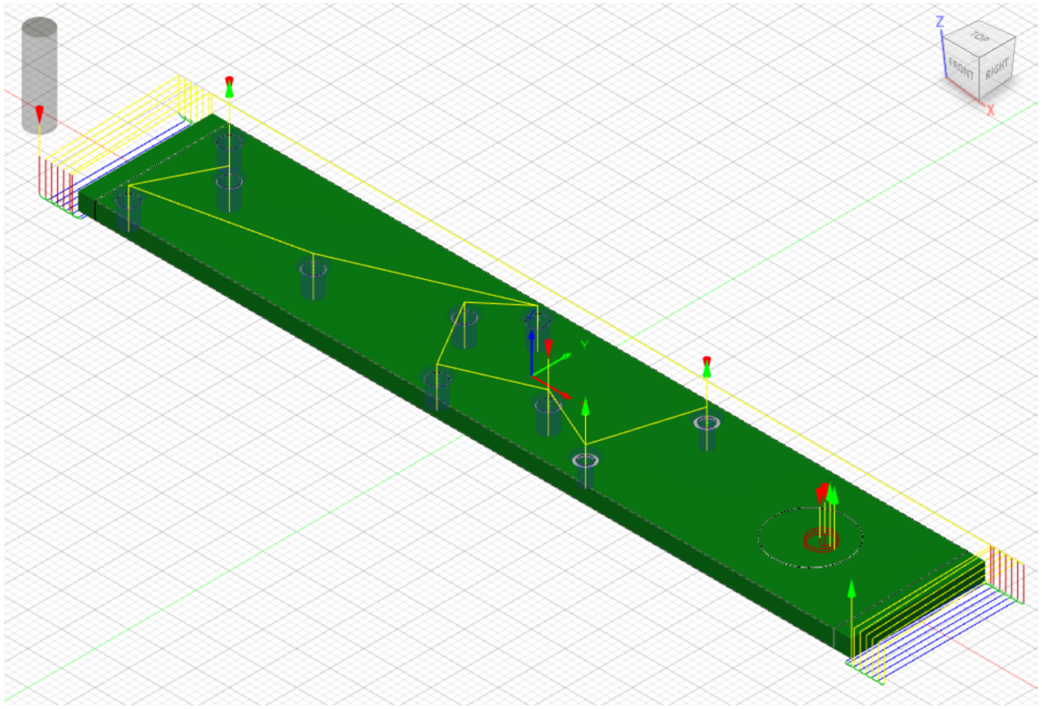


Figure 3.21: CAM Tool Path for Rocker Plate

The final CNC machined parts were the hex axles for the wheels. These axles were made of ½ in steel hex stock due to the output shafts of our gearboxes being ½ in hex shafts, and wanting to keep the sprockets consistent and cheap. This also allowed for the wheel inserts to be produced with internal hex geometry instead of needing a keyway as shown in Figure 3.22. The original plan for manufacturing the axles was to use a CNC lathe. However, since a tapping operation was required to attach the sprockets and encoders to the axles, this made the use of the lathes in Washburn more complicated. The age and consequential inaccuracy of the lathes means that a coaxial indicator should be used for tapping operations to ensure the tap is centered and reduce the risk of tap breakage. Milling was suggested as an alternative because the tap is being rotated instead of the part which reduces the chances of the tap breaking. A pneumatic collet chuck was used to fixture the stock so it could be brought to length, drilled, and tapped on both sides.

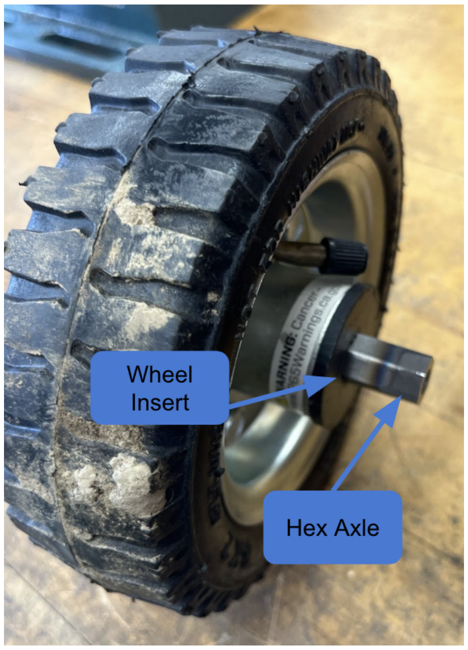


Figure 3.22: Hex Axle in Wheel Insert

With all of the integral components produced, the rockers were assembled. We began by pressing bearings into the top holes of the gusset plates and the rocker plates with an arbor press. The rocker plates were then attached to the gusset plates. Each gusset plate had assigned rocker plates due to the hole placements for each component being different on each side of the robot.

Next, we screwed the gearboxes in place in their respective holes for each of the gussets. The gusset plates were paired together with an inner plate made with a square cutout and an outer plate made with a circular cut out to fit the gear boxes between them. The inner rocker plates needed to have the differential holes aligned on one side of the robot, which was mistakenly overlooked when we were putting the rocker assembly together the first time. We made sure to pair the inner plates correctly in the final assembly.

At this point, we had four different completed plates each consisting of two rocker plates attached to one gusset plate. There were two outside configurations and two inner configurations of the rocker. We then placed the gearboxes in between an inner and outer assembly and fit the wheels on their axles and placed them in between the plates as well. We quickly realized we needed to add spacers between the wheels and the rocker plates on the wheel axles to ensure the wheels could not shift along the axle. These spacers were 3D printed twice. The first round of spacers had a diameter smaller than that of the bearings we used and needed to be printed larger to ensure the spacers would not pop the bearings out of the rocker plates during skid steering. The second and final round of 3D printed PLA spacers were larger in diameter than the bearings and were fitted properly on the axles.

The aluminum spacers and screws were then fitted in between the rocker plates. The two standoffs placed next to the differential holes had to be added after securing the differential to the rocker in the final assembly as the holes were too close together to use the proper tools to connect the differential nuts otherwise. Figure 3.23 below shows the differential attached to the rocker to display how close the holes are.

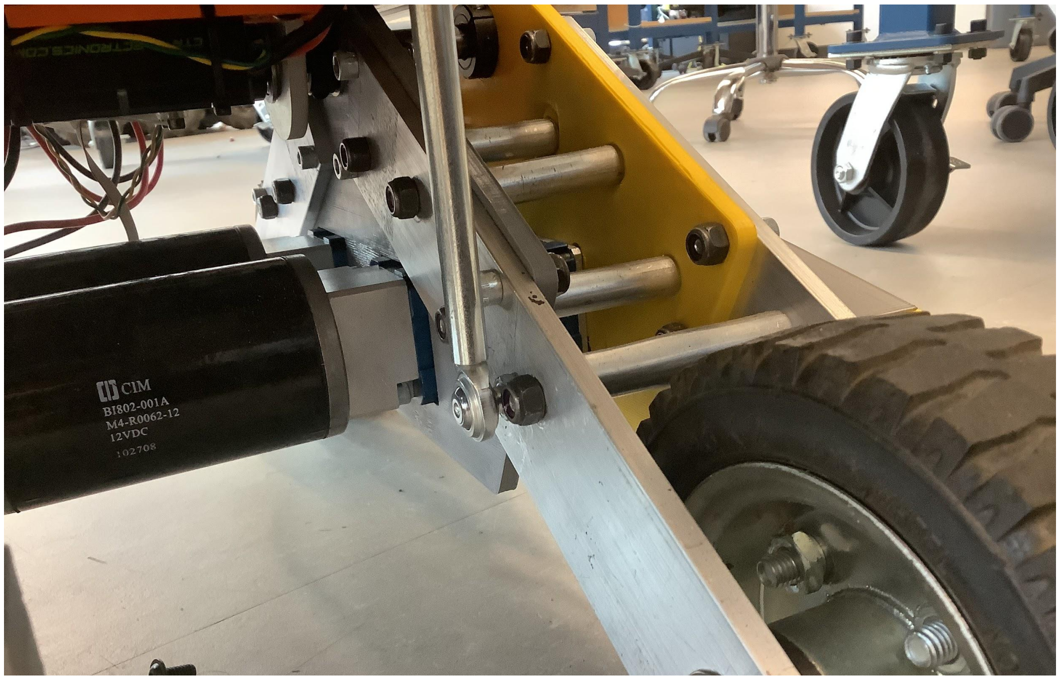


Figure 3.23: Differential Connected to Rocker

The long chains needed to be resized as there was a tolerancing issue during the gusset printing stage which caused some inconsistencies between the actual robot and the solidworks model. The new chain length was slightly longer needing 58.008 links after redoing the calculations, so we used 58 links instead of the original 56. While this added a small amount of slack in the chain, the chains had no slip while the robot was operating. After adjusting the chain lengths, we added the sprockets, PLA chain covers and the encoder mounts to the rocker.

During skid steering testing, we noticed that the wheel inserts were slipping in the wheels, preventing smooth turning, so the team had to disassemble the rockers to epoxy the wheel inserts inside of the wheels. After the epoxy had set for twenty-four hours, the wheels were placed back in the rocker and the robot was reassembled.

Next, the floor panel and batteries were placed into the robot and the electrical components deck was inserted into the robot. The plywood walls of the robot were laser cut and then screwed into the frame of the robot. The front panel was assembled to the robot first, with the addition of assembling the soil probe system after it was placed. The rack and pinion were screwed onto the front panel along with the motor mount.

The fertilizer tank with the pump inside of it was then placed in the back of the chassis, as shown in Figure #: Final Robot Assembled, and a 5/16th inch inner diameter silicone tubing was run from the tip nozzle on the tank to a T-shaped splitter to allow the fertilizer to connect to both sprayers on the front of the robot. The splitter connected the 5/16th inner diameter tubing to two 5 millimeter inner diameter tubes that ran from the splitter to the sprayers. Each 3D printed sprayer mount was screwed in and attached to the 5 millimeter tubing threaded through the robot from the fertilizer tank. The other panels were then assembled to the robot, along with a laser cut acrylic back panel. The top panel was attached with velcro for easy access to the electrical components and wire covers were 3D printed and screwed onto the top panel to protect the battery wires that were sticking out from the top. The final robot is shown in Figure 3.24 below.

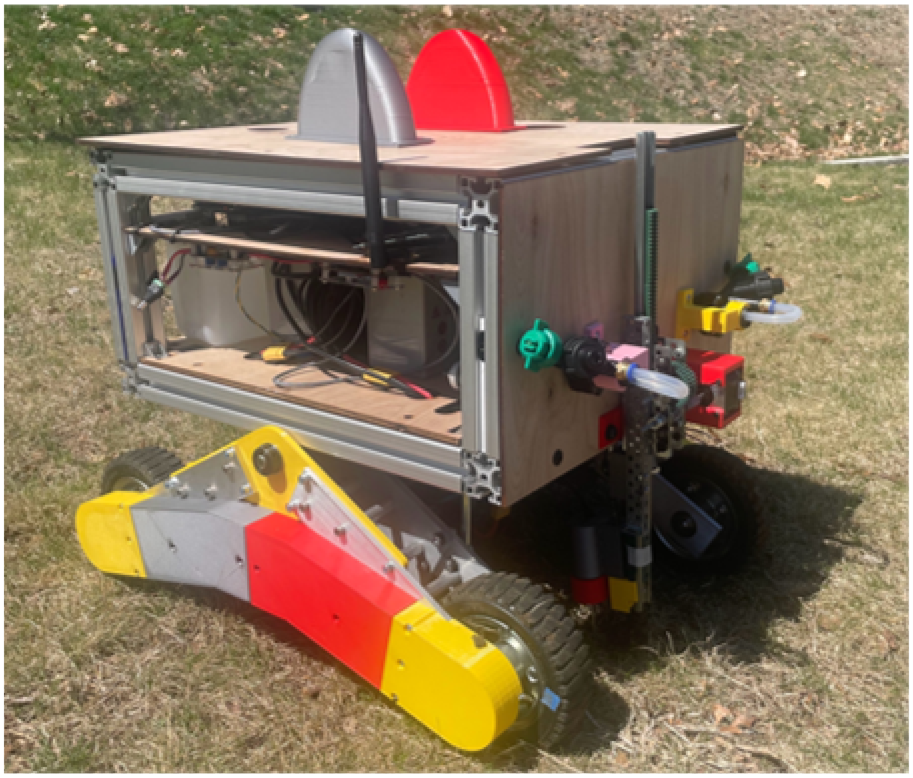


Figure 3.24: Final Robot Assembled

3.2.2 Soil Probe Design

The design for the probe, as shown in Figure 3.25 below, is an application of a rack and pinion mechanism; this converts the rotational motion of a pinion gear to translational motion of a rack. To initiate the rotational portion of the system, the design probe includes a stepper motor that is mounted on the body of the mechanism, and connects to the shaft of the pinion gear. The shape of the circular motor shaft hadn't matched the square shaped shaft for the gears so we designed and 3D printed an adapter. It was apparent that the 12-tooth pinion would not suffice to provide the necessary torque effectively to drive the rack and probe through soil, so we connected the pinion to two 36-tooth spur gears parallel to one another, to provide a reduction ratio of 3:1. Our design needed something more to ensure the repeatability of its function. This was important to consider since the prongs extending from our probe sensor were capable of breaking or even bending with enough force. To address this concern we included a spring in our design that absorbs a portion of the opposing forces of the soil as labeled below in Figure 3.25. Our design included a 3D printed outside slider truck that the probe sensor and spring are mounted on, that freely translates in a vertical motion. As such, when the probe reaches the ground it slides up until it reaches a stationary block, and the load the probe experiences is cushioned by the spring, as it is compressed between the two stationary blocks shown in the figure below. The last consideration taken into account was to include a component that would stop the probe from moving downward if the probe pushed against something too hard to penetrate. This issue was addressed with the second stationary block labeled with a red arrow in the figure below. It was placed such that if half the length of the spring was compressed, then the block would stop the movement of the probe. The design had its drawbacks, which included the limited range of motion of the mechanism. Due to the presence of stationary blocks the probe was limited to less than 2 inches of motion overall. In addition, the bottom stationary block, while performing its duty, limited the amount of depth the probe was able to penetrate soil. The penetration was at a half-inch depth.

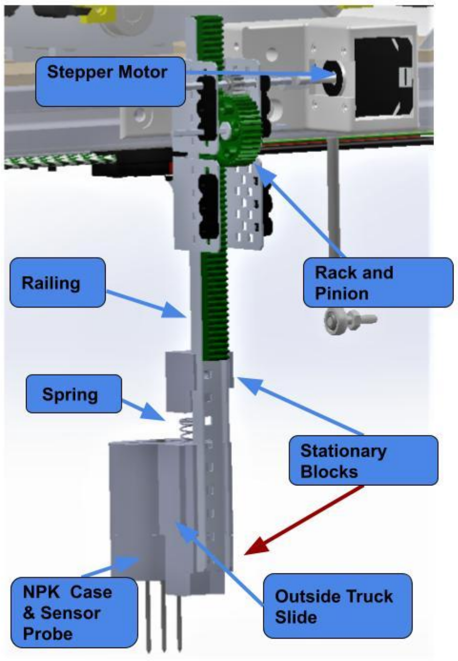


Figure 3.25: Slider Mechanism

3.2.3 Battery Level Indicator Design

3.2.3.1 Circuit Design

Batteries can get damaged if fully discharged or overcharged. A circuit that monitors the battery level of our robot is important in order to properly maintain the battery of our robot. As the battery runs out of power, the voltage level will decrease. This enables us to build a circuit that monitors the battery level by monitoring the voltage across its terminals, and then displaying the battery level to the user in the form of glowing LEDs. The circuit is shown in Figure 3.26 and it consists of:

- A LM3914 chip
- 3 resistors

- A potentiometer
- A switch
- 10 LEDs

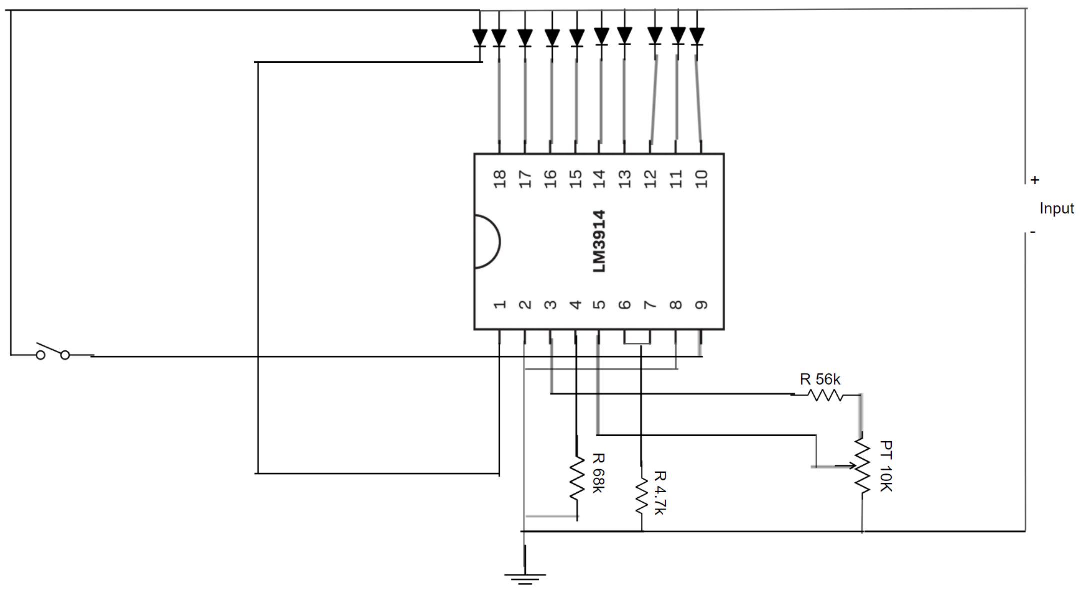


Figure 3.26: Battery level indicator circuit diagram

The LEDs will be used to show the level of the battery. If only 1 LED is turned on, the battery will be at 10%, if 2 LEDs are on then the battery level will be at 20% and so on. As the battery level drops, their voltage will drop as well, which will trigger the LEDs to turn off with each 10% drop in battery level.

3.2.3.2 Selection of resistors

In order to better understand how the resistors for this circuit are selected, it is important to understand the architecture of the LM3914 chip, and how it functions. A picture of what's inside of the LM3914 chip is shown below:

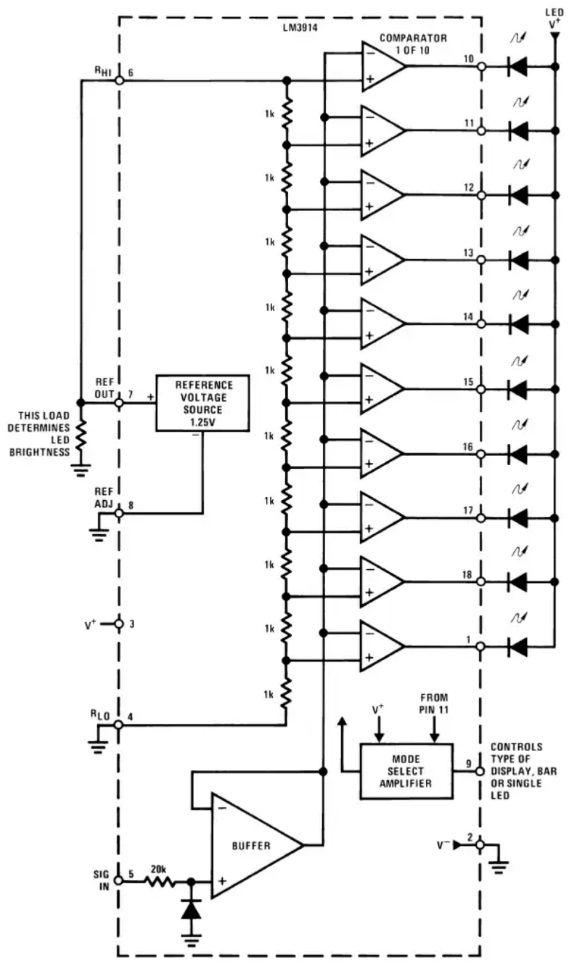


Figure 3.27: LM3914 architecture

As we can see from Figure 3.27, the chip consists of a unity gain amplifier, along with 10 comparators, separated by 1k resistors.

The resistor connected to pin 6 and 7 is responsible for the current across each LED, therefore the brightness of the LEDs. The current through the 10k resistor chain must be added to the current of this particular resistor. To account for the internal reference voltage, the formula used in order to determine the LED current is:

$$10((1.25V/R) + (1.25V/10k))$$

(Equation 3.8)

By selecting a 4.7k resistor, we have a LED current of 3.9mA. This value is important to keep in mind, because it affects the total power dissipation of the chip. Since we want bar mode to be an option, we have to account for the possibility of all 10 LEDs being on at the same time. Also accounting for 10 mA of device standing current, our max current should be:

$$(3.9\text{mA} * 10) + 10\text{mA} = 49 \text{ mA}$$
(Equation 3.9)

The voltage across the chip will be the supply voltage – the LED voltage (typically 2V). This will bring us to a 12V-2V = 10V. Multiplying these values together, we get a power of 490mW dissipated by the chip. This value is well below the max rating of 1365 mW in the datasheet, and it accounts for different factors such as a high environment temperature, which could affect the performance of the chip.

The resistor connected in pin 4 affects the range of voltages that all LEDs can light up. The selection of this resistor is important, because the correct range of voltages is needed in order to correctly measure the battery level. After research, it was determined that the voltage of a lead acid battery should be approx 11.2V when the battery is at 10%, and around 12.7V when the battery is at 100% capacity. These values show us the range of voltages for which LED 1 and LED 10 should light up respectively. Looking at the schematic, the resistor that will be connected at pin 4, will connect right at the beginning of the 10k resistor series that connects to our positive terminal of the comparators. This forms a voltage divider that we will use to determine the voltage range at which LEDs 1-10 turn on. By selecting a 68k resistor, the voltage range of our LEDs will be:

$$12.7V - 12.7V * (69k/(68k + 10k)) = 1.47\text{V}$$
(Equation 3.10)

This value is approximate to the 12.7V - 11.2V = 1.5V range in between LED 1 -10 to light up. Next, we have a voltage divider of a 56k resistor and a 10k potentiometer connected to the signal pin of LM3914 chip. This voltage divider will be used in order to calibrate the circuit. A power supply device will firstly be needed in order to supply the 11.2V signal such as in Figure 3.28. Then, the potentiometer will be adjusted such that the first LED lights up at this value. Next up, the circuit is tested by slowly increasing the voltage level and observing that the LEDs turn on at the desired value, such as in Figure 3.29.

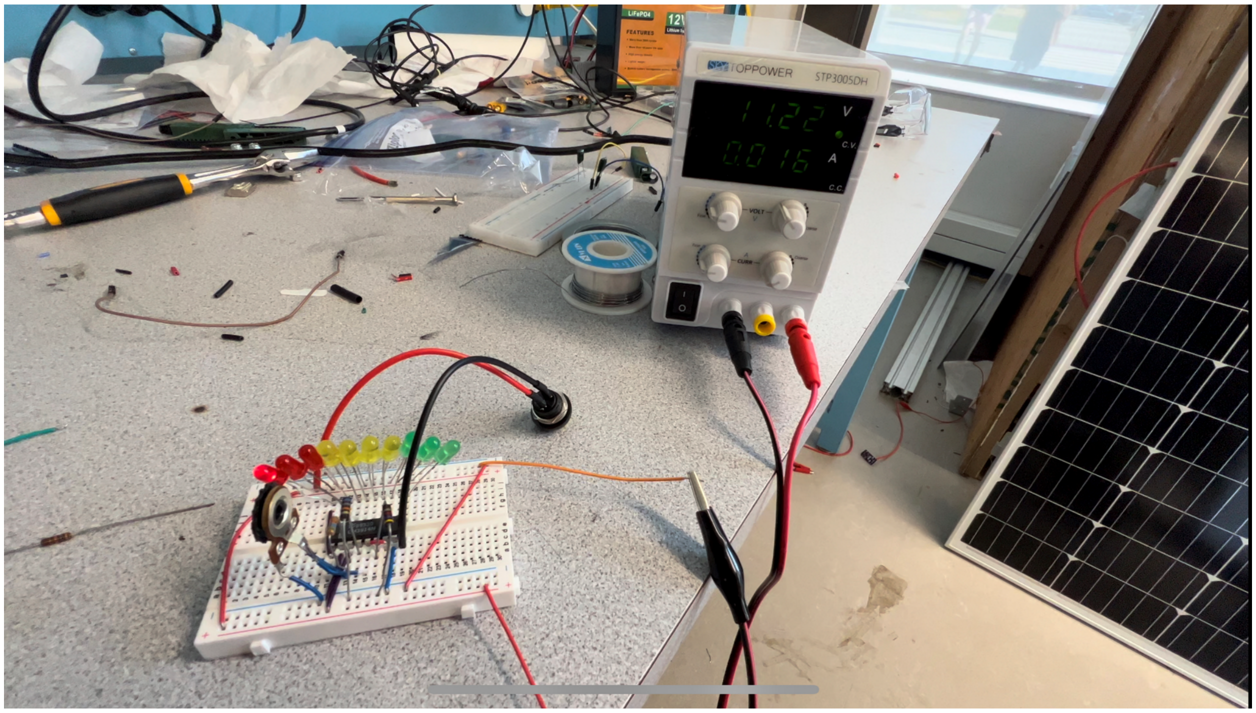


Figure 3.28: LED 1 at 11.22V

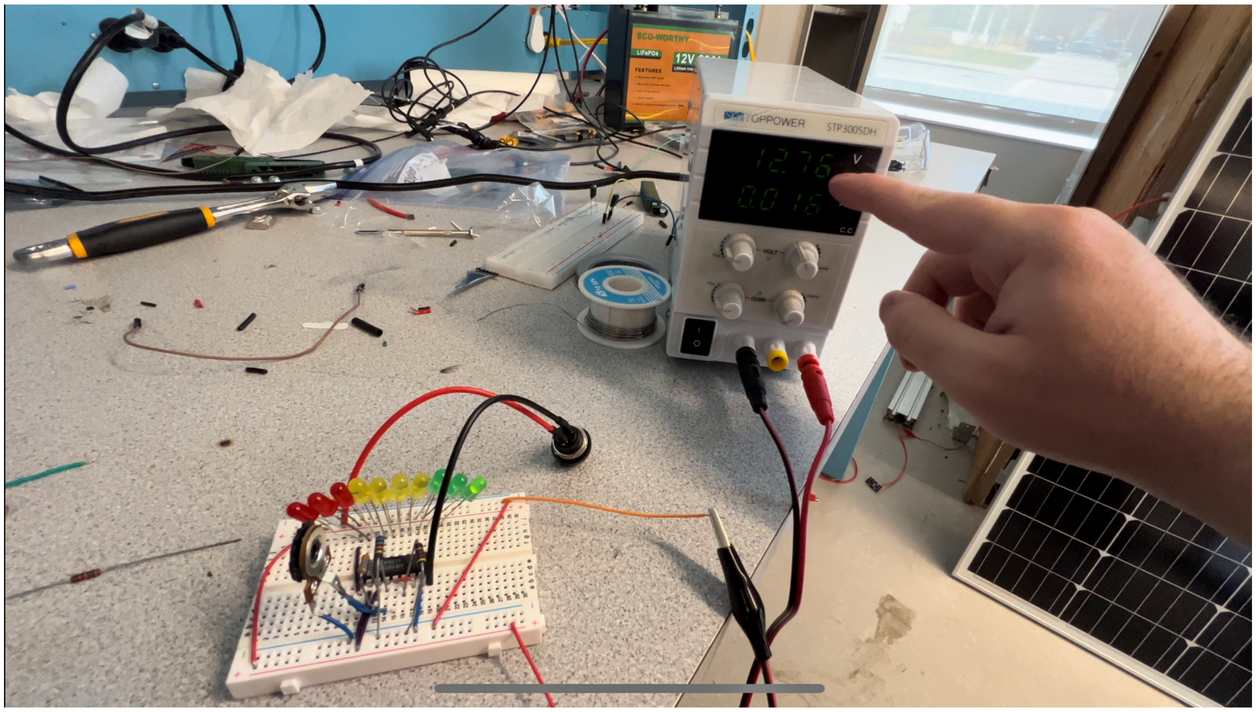


Figure 3.29: LED 10 at 12.76V

3.2.3.3 Implementation

After testing that the LEDs light up at the correct voltages and that the circuit works, the next step was implementing this circuit in our robot. The aim of this circuit was to measure the battery level of our robot, therefore the circuit was connected to the battery of our robot through the use of alligator clips. The next step was to find a space within our robot in order to place the circuit without it interfering with other components inside the robot. Finally, in order to prevent any exposed wires from interfering with other components, the circuit was isolated through the use of insulating tape and a ziplock bag, as shown in Figure 3.30.

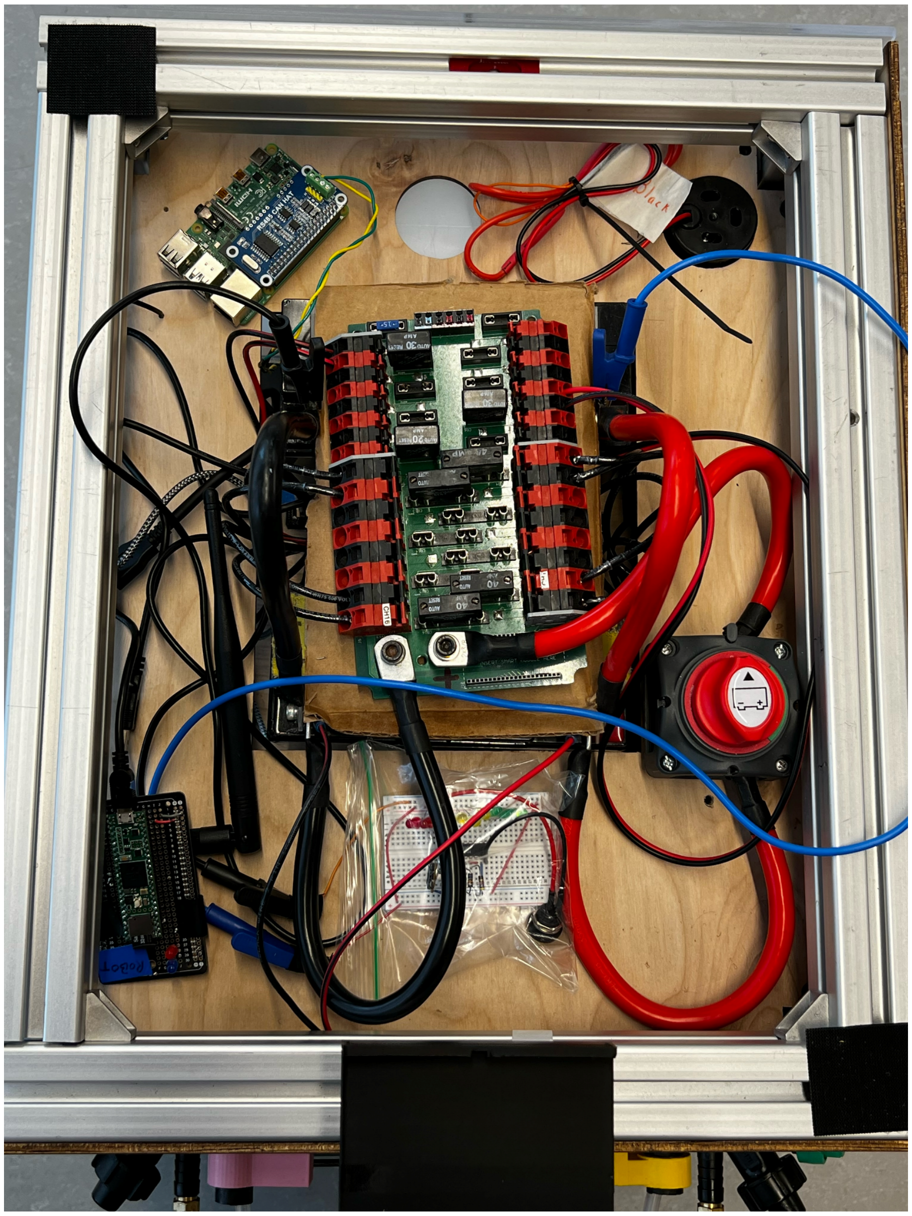


Figure 3.30: Implementation of the circuit

3.2.4 Radio Transmitter Circuit Prototype

Before the creation of a final circuit design, the radio modules needed to be tested to ensure they would work in the final design. This involved making a preliminary circuit and then testing both circuits to ensure that a message could be received and sent from both devices. To make this preliminary design as simple and quick as possible a breadboard was used to place each component onto.

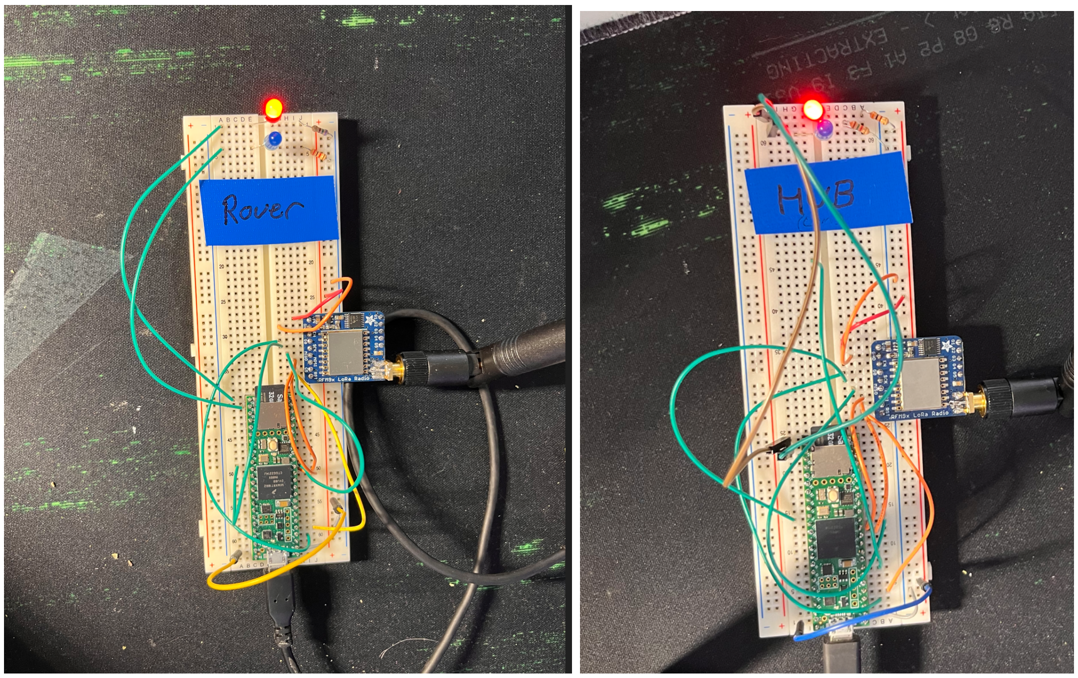
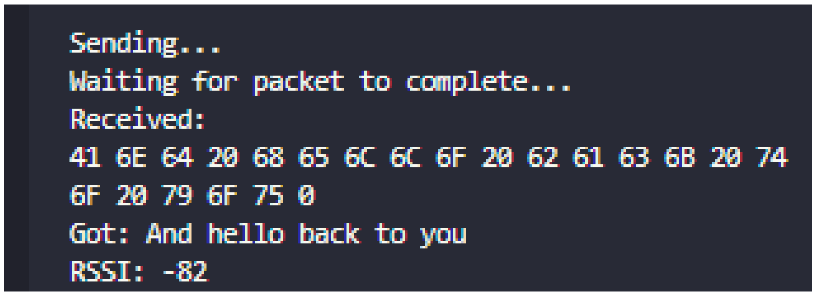


Figure 3.31: Radio Transmitter Prototypes

The prototype of both the hub and rover transceivers can be seen in Figure 3.31. The top of the breadboard contains two indicator LEDs. The red led is used to show that the device has received a signal from the other, confirming a connection. The other LED is used to show when the device is transmitting. The addition of the LEDs were done to improve the programming experience and also to allow for quick analysis of problems without having to dive into system logs, and wire connections.



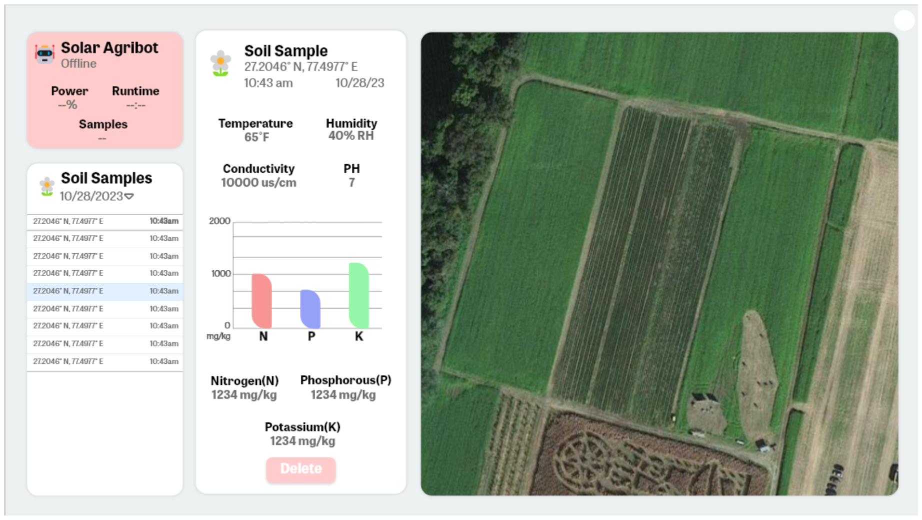
```
Sending...
Waiting for packet to complete...
Received:
41 6E 64 20 68 65 6C 6C 6F 20 62 61 63 6B 20 74
6F 20 79 6F 75 0
Got: And hello back to you
RSSI: -82
```

Figure 3.32: Radio Transmitter Prototype Testing

Once both prototypes were wired up they were tested to ensure a message could be sent back and forth between them. In Figure 3.32 the serial output of the rover microcontroller can be seen. This is the result of sending a “hello” message to the hub and the hub replying with a “And hello back to you”. Since it could be verified that the radio system will work as expected, a final design and more permanent board could be worked on.

3.2.5 Farmer Website Design Prototyping

The farmer website is the main way in which farmers can interact with the robot. Thus a good design is crucial in order for our robot to meet our requirement of being easy to use for the farmer. To ensure a good final design, prototypes were made using AdobeXD, allowing for rapid testing of designs.


Solar Agribot
Offline
Power --%
Runtime --:--
Samples --

Soil Samples
10/28/2023

27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am

Soil Sample
27.2046° N, 77.4977° E
10:43 am 10/28/23
Temperature 65°F
Humidity 40% RH
Conductivity 10000 us/cm
PH 7
Nitrogen(N) 1234 mg/kg
Phosphorous(P) 1234 mg/kg
Potassium(K) 1234 mg/kg

Figure 3.33: Farmer Website First Design

The first design followed a modular design of four columns (Figure 3.33).

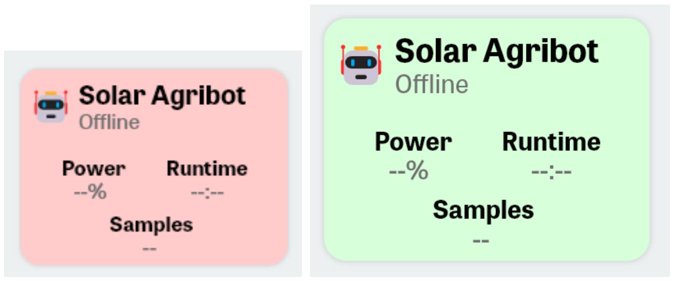
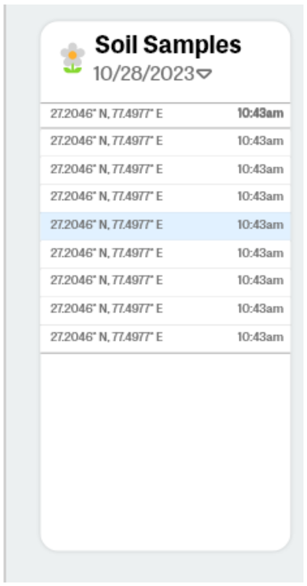


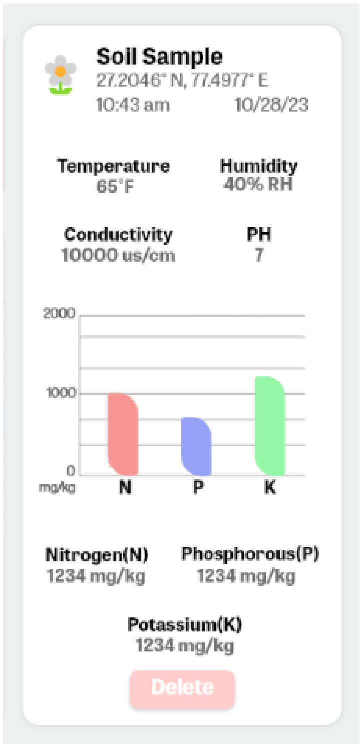
Figure 3.34: Farmer Website First Design - Robot Status


Soil Samples
10/28/2023

27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am
27.2046" N, 77.4977" E	10:43am

Figure 3.35: Farmer Website First Design - Soil Sample List

The first column contains two modules, one for robot status(Figure 3.34) and one for viewing collected soil samples (Figure 3.35). The robot status module contains information such as if the robot is currently online. When the robot is online, this module would turn green and show the farmer the current power level, runtime of the robot and the total samples it has taken in this time. Once the robot has powered off, the module would change to a red color signifying the robot is offline. The second module allows for the viewing of collected soil samples. In this module, the user is shown a list of all the samples on the latest sample date. The user is then able to click on one of the items and view information about it in a different module. Additionally, the user can select the dropdown arrow next to the date to view a list of all other dates samples were taken and view those samples by clicking on the date.


Soil Sample
27.2046° N, 77.4977° E
10:43 am 10/28/23

Temperature
65°F

Humidity
40% RH

Conductivity
10000 us/cm

PH
7

Nitrogen(N)
1234 mg/kg

Phosphorous(P)
1234 mg/kg

Potassium(K)
1234 mg/kg

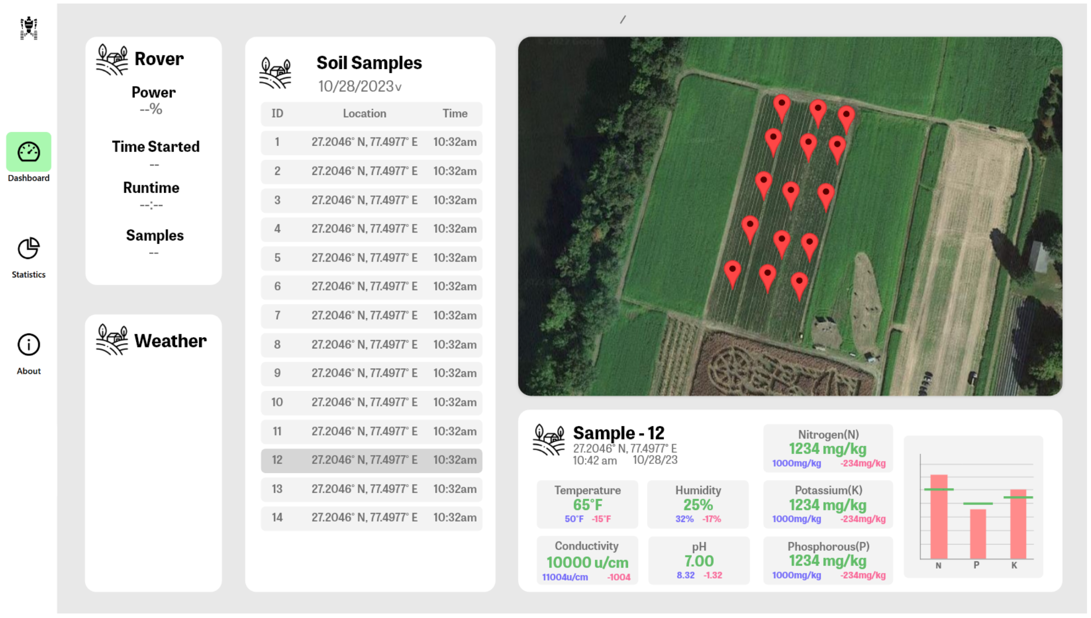
Figure 3.36: Farmer Website First Design - Soil Sample Data

The second column contains the soil sample module (Figure 3.36). This module shows the farmer the soil condition of the selection module in column one. This module starts with the location in longitude and latitude of the soil sample in addition with the time and date it was taken. Next, the temperature, humidity, conductivity, and ph are laid out below it in a 2 x 2 pattern. Below this the N, P, and K soil values are laid out below in a bar chart, a different color being used for N, P and K. Below this chart the exact values for the N, P, and K, values are displayed in mg/kg.. At the bottom of this page the options for the farmer to delete this data entry is also present.

Figure 3.37: Farmer Website First Design - Samples Map

The third and fourth column consisted of a map of the farmer's farm (Figure 3.37). This map would contain pins of each soil sample's collection location. Additionally, the farmer would be able to select samples by clicking on their pin.

This design fell short in a few ways. First the module for robot status and map were taking up way more screen real estate than necessary. Secondly the entire soil samples list module felt too cluttered and awkwardly small, making navigating different samples difficult. Finally the lack of contrast between the white modules and the slightly blue background make the modules difficult to look at for long periods of time.



Rover

Power
--%

Time Started
--

Runtime
--:--

Samples
--

Weather

Soil Samples
10/28/2023v

ID	Location	Time
1	27.2046° N, 77.4977° E	10:32am
2	27.2046° N, 77.4977° E	10:32am
3	27.2046° N, 77.4977° E	10:32am
4	27.2046° N, 77.4977° E	10:32am
5	27.2046° N, 77.4977° E	10:32am
6	27.2046° N, 77.4977° E	10:32am
7	27.2046° N, 77.4977° E	10:32am
8	27.2046° N, 77.4977° E	10:32am
9	27.2046° N, 77.4977° E	10:32am
10	27.2046° N, 77.4977° E	10:32am
11	27.2046° N, 77.4977° E	10:32am
12	27.2046° N, 77.4977° E	10:32am
13	27.2046° N, 77.4977° E	10:32am
14	27.2046° N, 77.4977° E	10:32am

Sample - 12
27.2046° N, 77.4977° E
10:42 am 10/28/23

Temperature	Humidity	Nitrogen(N)
65°F 50°F -15°F	25% 32% -17%	1234 mg/kg 1000mg/kg -234mg/kg

Conductivity	pH	Potassium(K)	Phosphorous(P)
10000 u/cm 11004u/cm -1004	7.00 8.32 -1.32	1234 mg/kg 1000mg/kg -234mg/kg	1234 mg/kg 1000mg/kg -234mg/kg

Figure 3.38: Farmer Website Final Design

The final design (Figure 3.38) also follows a four column design but fixed all of these key issues and also added a weather module and a navigation bar (Figure 3.39).

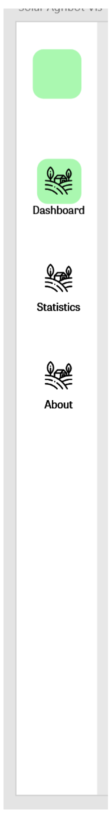


Figure 3.39: Farmer Website Final Design - Navigation Bar

The navigation bar located on the right side of the page allows for routing to future pages like a statistics page that would be used for the farmer to view the health of his field over time. The icon of the current page turns green to symbolize the currently selected page.

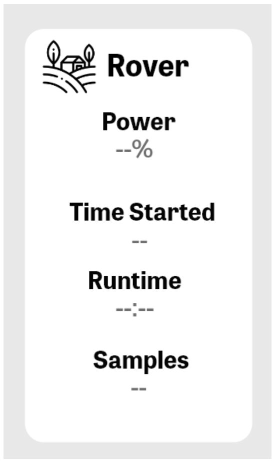
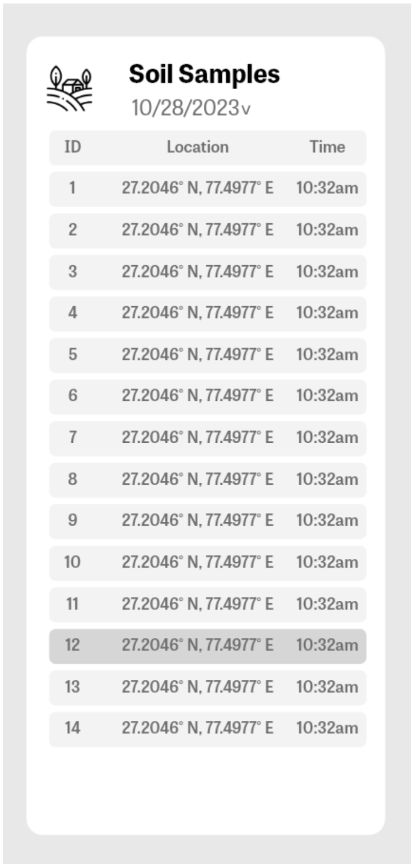


Figure 3.40: Farmer Website Final Design - Rover Information

The robot information module (Figure 3.40) changed slightly from the original but displays the same information. The panel is still located in the first column but now takes up more vertical space and less horizontal space to allow other crucial components to use the screen space.

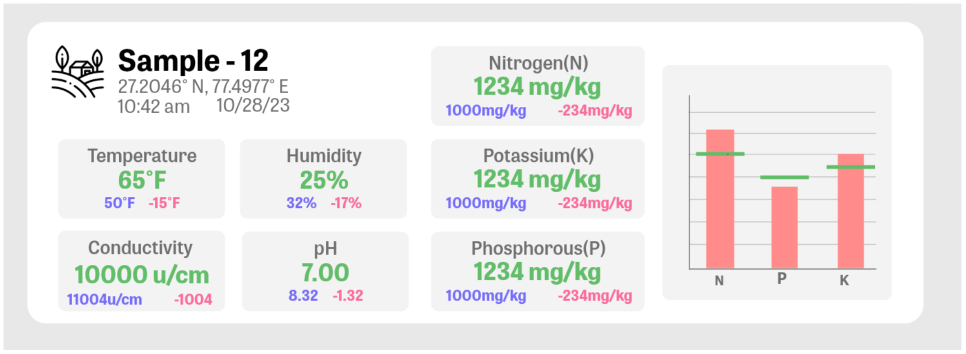


Soil Samples
10/28/2023v

ID	Location	Time
1	27.2046° N, 77.4977° E	10:32am
2	27.2046° N, 77.4977° E	10:32am
3	27.2046° N, 77.4977° E	10:32am
4	27.2046° N, 77.4977° E	10:32am
5	27.2046° N, 77.4977° E	10:32am
6	27.2046° N, 77.4977° E	10:32am
7	27.2046° N, 77.4977° E	10:32am
8	27.2046° N, 77.4977° E	10:32am
9	27.2046° N, 77.4977° E	10:32am
10	27.2046° N, 77.4977° E	10:32am
11	27.2046° N, 77.4977° E	10:32am
12	27.2046° N, 77.4977° E	10:32am
13	27.2046° N, 77.4977° E	10:32am
14	27.2046° N, 77.4977° E	10:32am

Figure 3.41: Farmer Website Final Design - Sample List

The soil samples list (Figure 3.41) has been moved to the second column, where it spans the entire vertical length. This panel still displays all of the information from the previous design but now also shows an ID for each soil sample. The list now also follows a sleeker design allowing simpler reading and navigating between samples. The dropdown array next to the date can still be used to select a previous sample date.



Sample - 12
27.2046° N, 77.4977° E
10:42 am 10/28/23

Temperature
65°F
50°F -15°F

Humidity
25%
32% -17%

Conductivity
10000 u/cm
11004u/cm -1004

pH
7.00
8.32 -1.32

Nitrogen(N)
1234 mg/kg
1000mg/kg -234mg/kg

Potassium(K)
1234 mg/kg
1000mg/kg -234mg/kg

Phosphorous(P)
1234 mg/kg
1000mg/kg -234mg/kg

Figure 3.42: Farmer Website Final Design - Sample Information

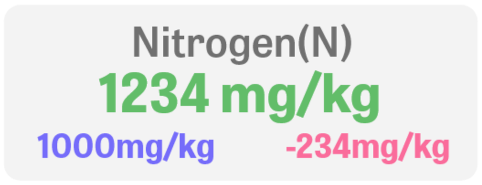


Figure 3.43: Farmer Website Final Design - Soil Data Information Block

The sample information module (Figure 3.42) is now located in the bottom of the third and fourth column. This panel received a major overhaul aesthetically. The module itself follows a modular design of different data entries and a graph. Each data point now shows the measured value, the desired value and the difference between these two numbers (Figure 3.43). Three different colors are used here to allow for easy differentiation between the three numbers. The graph shows tbe measured N, P, and K values, in addition this graph will show lines for the desired value for each of these measurements.

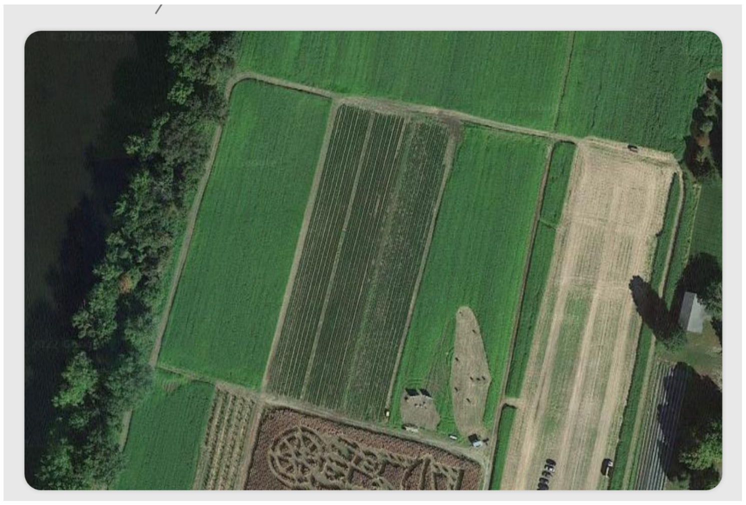
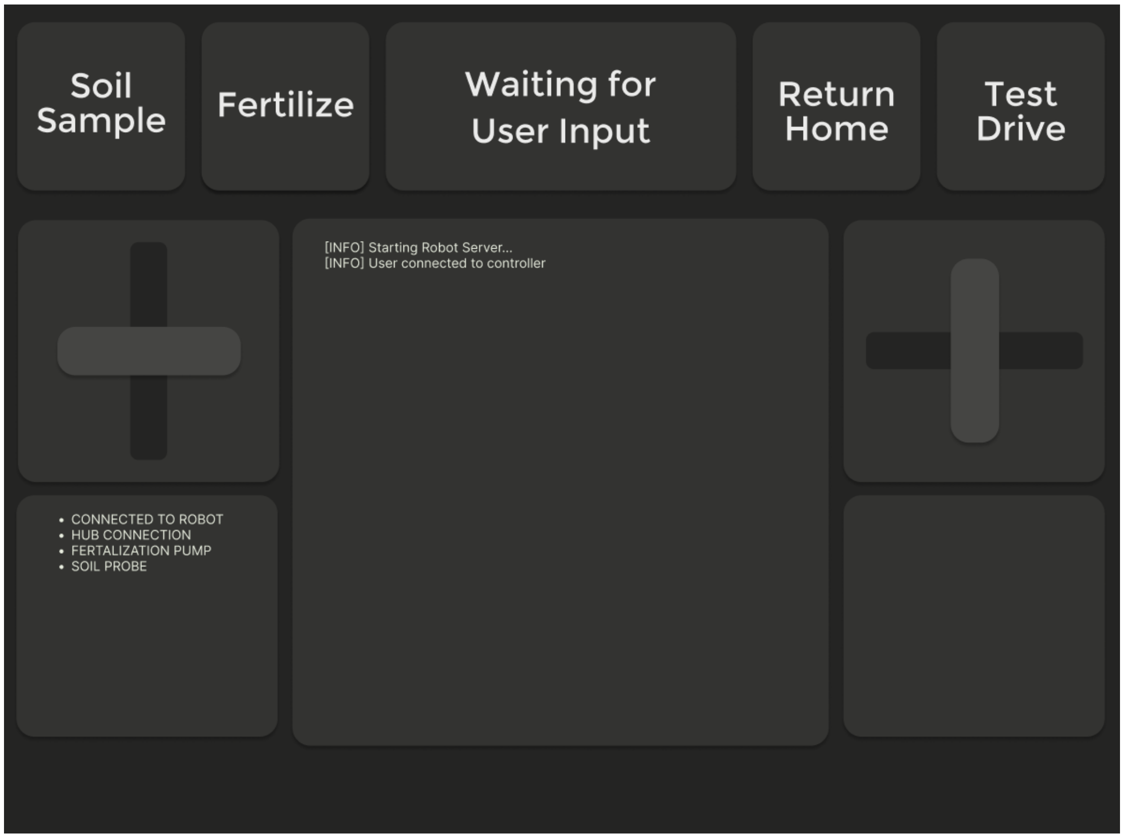


Figure 3.44: Farmer Website Final Design - Samples Map

The final map design (Figure 3.44) has not changed much from the first design, The map still occupies the third and fourth columns but now does not extend to the bottom of the page. This spot is taken up by the sample information module mentioned previously.

3.2.6 Robot Controller Interface Design Prototyping

The robot controller interface is the main way in which we will conduct system testing, such as testing the drive motors. Thus good design is crucial in order to allow development of the robot’s subsystems to proceed smoothly. To ensure a good final design, prototypes were made using Figma, allowing for rapid testing of designs.



```
[INFO] Starting Robot Server...
[INFO] User connected to controller
```

- CONNECTED TO ROBOT
- HUB CONNECTION
- FERTALIZATION PUMP
- SOIL PROBE

Figure 3.45: First Design of Robot Controller

The first design of the robot controller (Figure 3.45) follows a modular design.

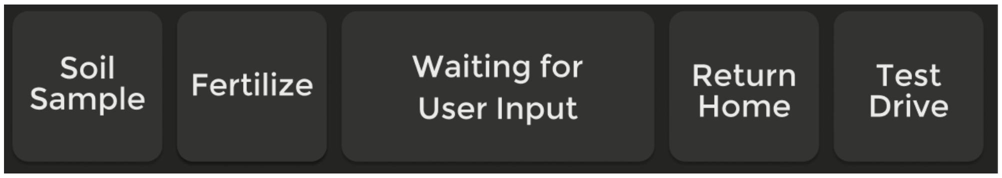
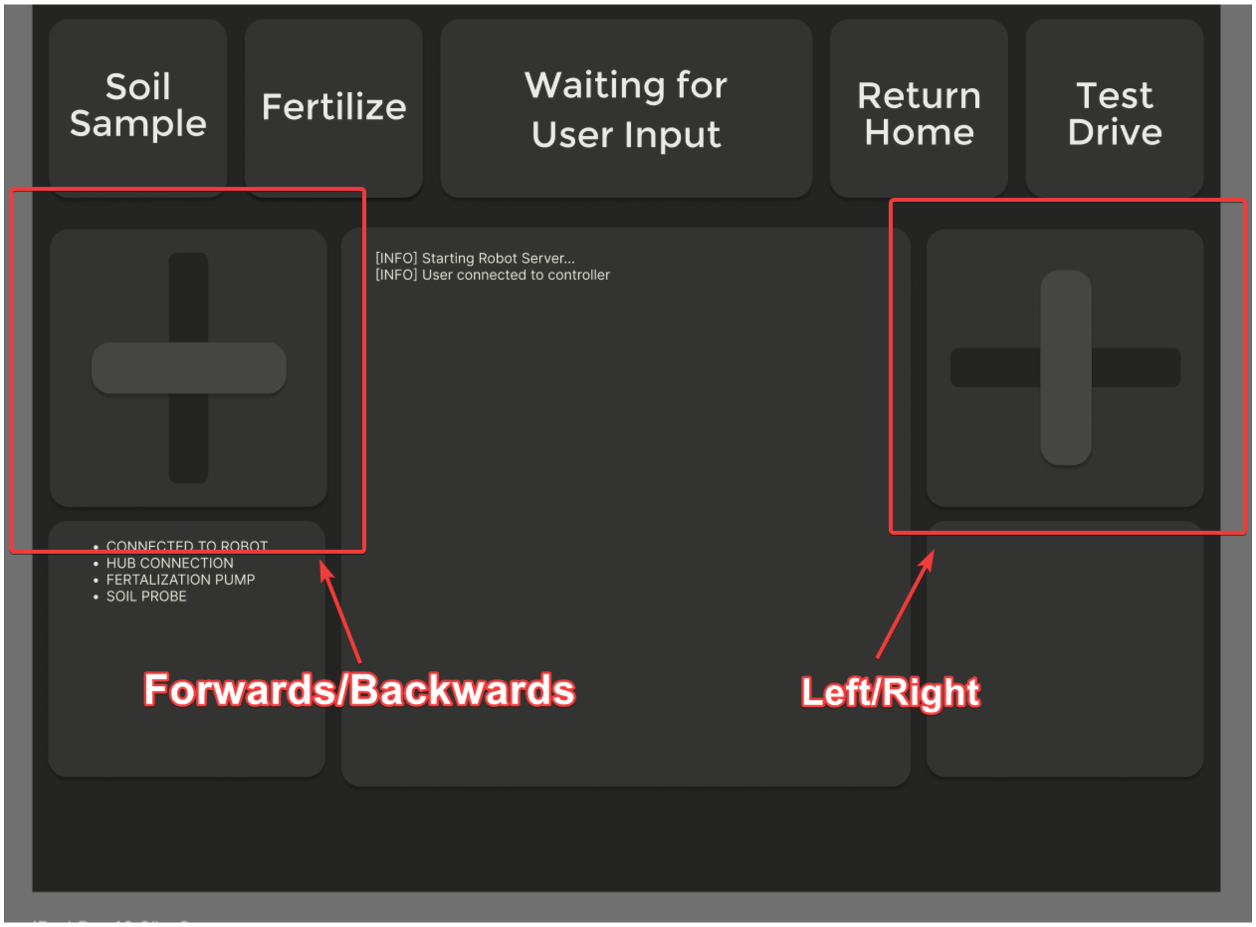


Figure 3.46: First Design of Robot Controller - User Buttons and Status

At the top of the page there are four buttons the user can click to initiate an action on the robot (Figure 3.46). When a button is clicked and the robot begins an action, the status module in the top middle of the page will update to provide feedback to the user on what the robot is doing.

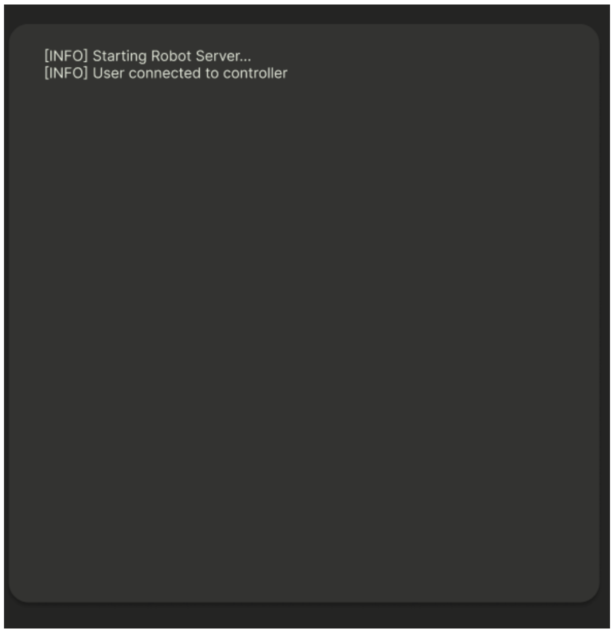


[INFO] Starting Robot Server...
[INFO] User connected to controller

- CONNECTED TO ROBOT
- HUB CONNECTION
- FERTALIZATION PUMP
- SOIL PROBE

Figure 3.47: First Design of Robot Controller - JoySticks

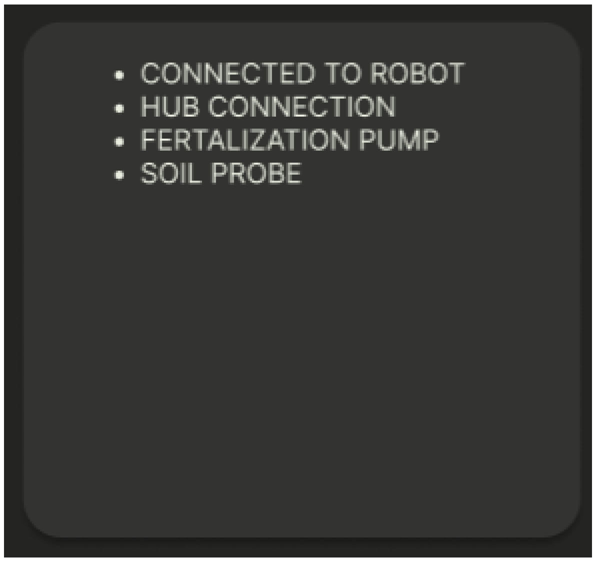
On the left side right side of the page there are two joysticks for the user to interact with (Figure 3.47). The joystick on the left side moves forwards and backwards controlling the robot's velocity. The joystick on the right moves right and left controlling the robot's rotation. When the user lets go of either joystick, the respective joystick will return to its default position.



```
[INFO] Starting Robot Server...
[INFO] User connected to controller
```

Figure 3.48: First Design of Robot Controller - Robot Information Panel

In the center of the page is a box dedicated to important console messages from the robot (Figure 3.48). The Robot Information Panel will populate whenever an information or error is logged on the robot. The Panel will be useful for providing live feedback of what the robot is doing and also for quick debugging when an action does not go to plan.

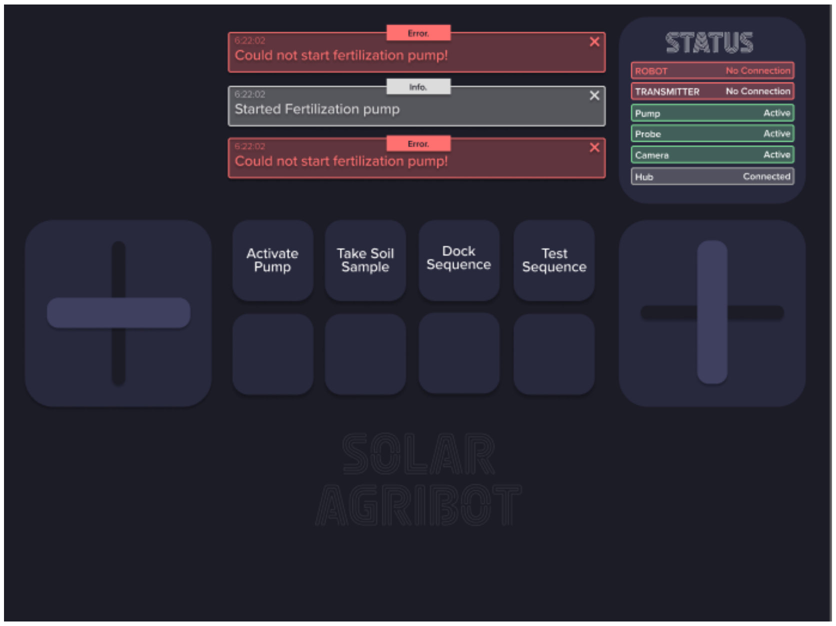


- CONNECTED TO ROBOT
- HUB CONNECTION
- FERTALIZATION PUMP
- SOIL PROBE

Figure 3.49 First Design of Robot Controller - Robot System Status

On the bottom left side of the page there is a module dedicated to displaying information of current robot systems (Figure 3.49). This module will act as a quick look in determining if a system is not connected, lost connection to the main robot controller.

The first design fell short in many ways, the main one being that the design felt flat with its minimal use of colors. Additionally the status bar did not allow for more than one message to be displayed to the user at a time. Finally the location of the buttons made them difficult to interact with on a tablet and did not allow for many buttons to be used during testing.



Error.
6:22:02
Could not start fertilization pump!

Info.
6:22:02
Started Fertilization pump

Error.
6:22:02
Could not start fertilization pump!

STATUS

ROBOT	No Connection
TRANSMITTER	No Connection
Pump	Active
Probe	Active
Camera	Active
Hub	Connected

Figure 3.50: Final Design of Robot Controller

The final design of the controller (Figure 3.50) fixed all of these issues by providing a sleeker design, improved status messages, added four additional buttons and improved the placement of the buttons.

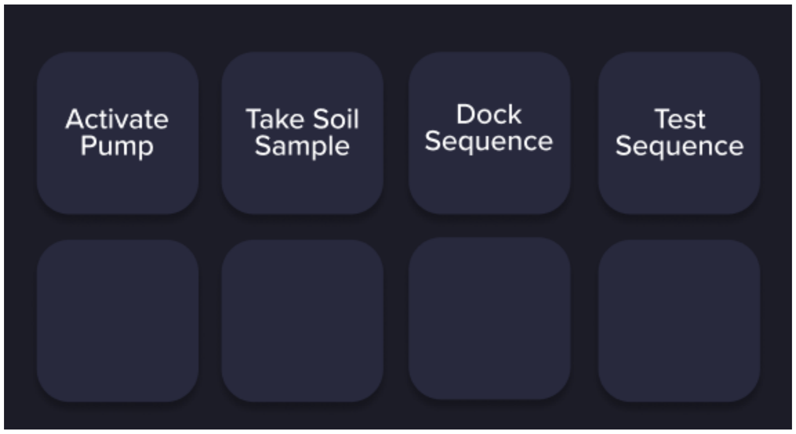


Figure 3.51: Final Design of Robot Controller - Buttons

‘

The controller now contains eight buttons (Figure 3.51) which are located at the center of the controller. These buttons can be used to initiate or terminate pre-programmed robot processes. Half of these buttons are blank to allow for additions of buttons that become relevant during development and testing.

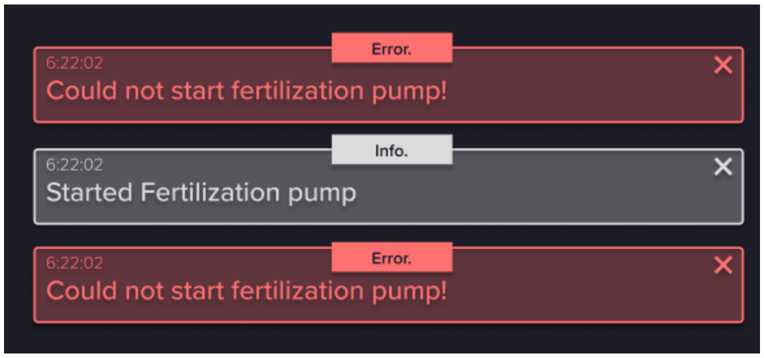
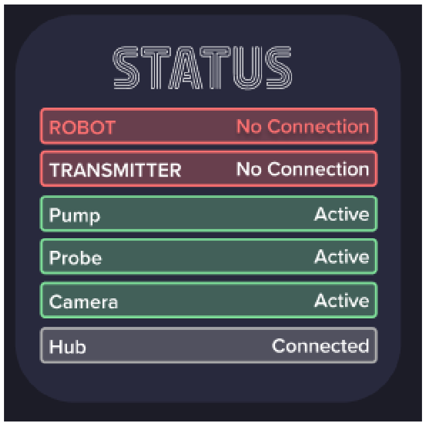

Error.
6:22:02
Could not start fertilization pump!
Info.
6:22:02
Started Fertilization pump
Error.
6:22:02
Could not start fertilization pump!

Figure 3.52: Final Design of Robot Controller - Toasts

The status bar at the top of the page has been removed and is now home to an area of toasts to appear to the user (Figure 3.52). There are two types of toasts, Information and Error. The Information Toasts have a white theme and the Error toasts have a red theme, making them easily distinguishable between the two. These toasts appear typically as feedback for the user when an action begins. Additionally these toasts can be used to show when an action goes wrong so the user is aware that the pre-programmed action needs to be debugged. Finally these toasts will also inform the user when the controllers connect or disconnect from the robot.


STATUS

ROBOT	No Connection
TRANSMITTER	No Connection
Pump	Active
Probe	Active
Camera	Active
Hub	Connected

Figure 3.53: Final Design of Robot Controller - Subsystem Status

The subsystem status module had been moved to the top right of the controller (Figure 3.53). The module now has a better color indication of the current status of a component. A components status block will turn red when not connected, white when connected, and green when actively performing an action. This allows for the user to more quickly understand the current status of a component.

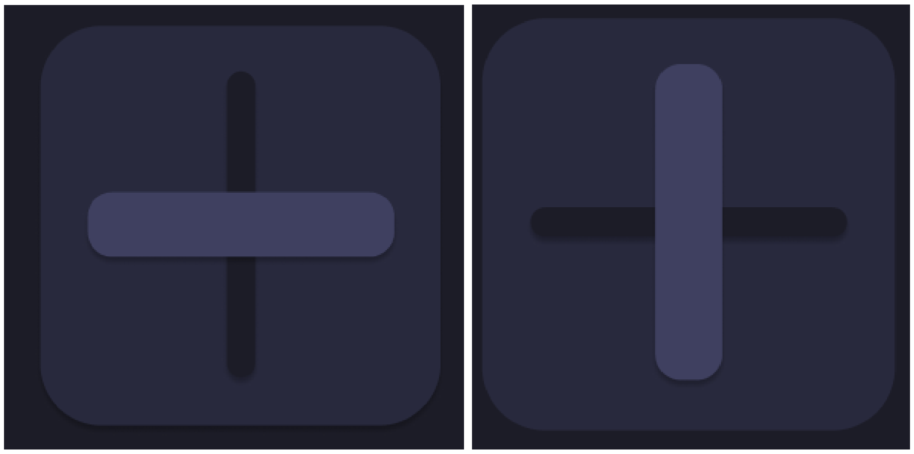


Figure 3.54: Final Design of Robot Controller - Joysticks

The Joystick controllers are still located in the same position as they were in the original design (Figure 3.54). They now have changed color to match the new color scheme of the final design.

3.2.7 Fertilizer Controller Design

In order to achieve our maximum crop yields and save on fertilizer costs. By testing soil NPK values row by row, the rover indicates where fertilization is required. Furthermore, since the demand for NPK is dynamic during the crop growth phase, specific fertilizer ratios are used. Also, fertilize fields after heavy rains by making decisions based on previous weather reports on the internet. Accurate weather forecasts can also avoid fertilization before heavy rains, which can cause soil erosion and fertilizer loss. During days of extreme heat waves, avoid watering and fertilizing to prevent thermostatic shock to the plants. So by combining online weather reports with soil testing probes, targeted fertilization can maximize yields and reduce fertilizer costs. The whole system is connected to the main power supply through a voltage regulator, because the nominal voltage of the pump is 12V, so the voltage is maintained at 12V through this regulator. At the same time, in the use of water pumps, we use double judgment to control the opening and closing of water pumps. I connected two components on Arduino, one is a float switch and the other is a motion sensor. The float switch is placed in the tank. When the fertilizer content in it is sufficient, the water pump can start to work and pump the fertilizer in the tank for spraying. At the same time, a set of motion sensors is installed on the rear of the robot. The sensing range of this set of sensors is 4 meters, which is just outside the range of chemical fertilizer spraying. When humans appear within the range of this sensor, the water pump will stop working , to prevent chemical fertilizers from being sprayed on the personnel's body.

For the selection of the pump, we finally chose the 12V agricultural water pump from the IEIK brand, which is the water pump in Table 3.2. Because this water pump can meet several of our usage criteria: 1. It can be placed outside the tank without occupying the liquid volume of the tank. 2. It has enough power to suck out the liquid in the tank and pump it in the direction of spray. 3. The energy consumption will not be too large to increase the use burden of the entire robot. This water pump has a power of 0.8 GPM. This data can be seen in Table 3.2. This power can spray our 60 ounces of liquid fertilizer on the land we need in about 40 seconds, which meets the demand.

Table 3.2: Data sheet for Fertilizer System

Part Name	Part function	Component parameters
IEIK 12 Volt Diaphragm Pump	Extract the fertilizer into the spray	12V DC 0.8 GPM 100 PSI
HC-SR501 Motion Sensors	Detect moving objects	4.5-20V DC 0.3-200s time range 3-7m detect range 23mm diameter
Omabeta Float Switch	Detect liquid level and control water pump switch	0-200V DC 0.5-1.0A 10 Max power 0-120℃ Work temperature

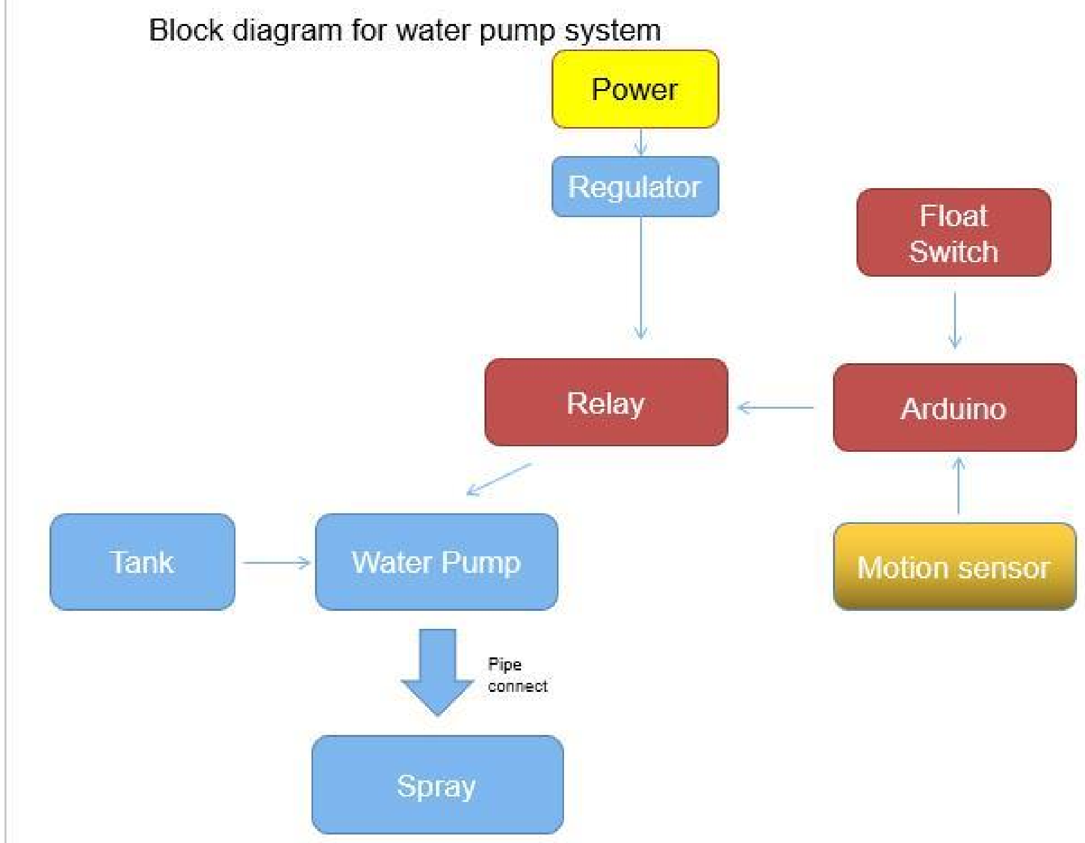


Figure 3.55: Fertilizer Controller-Block diagram for pump system

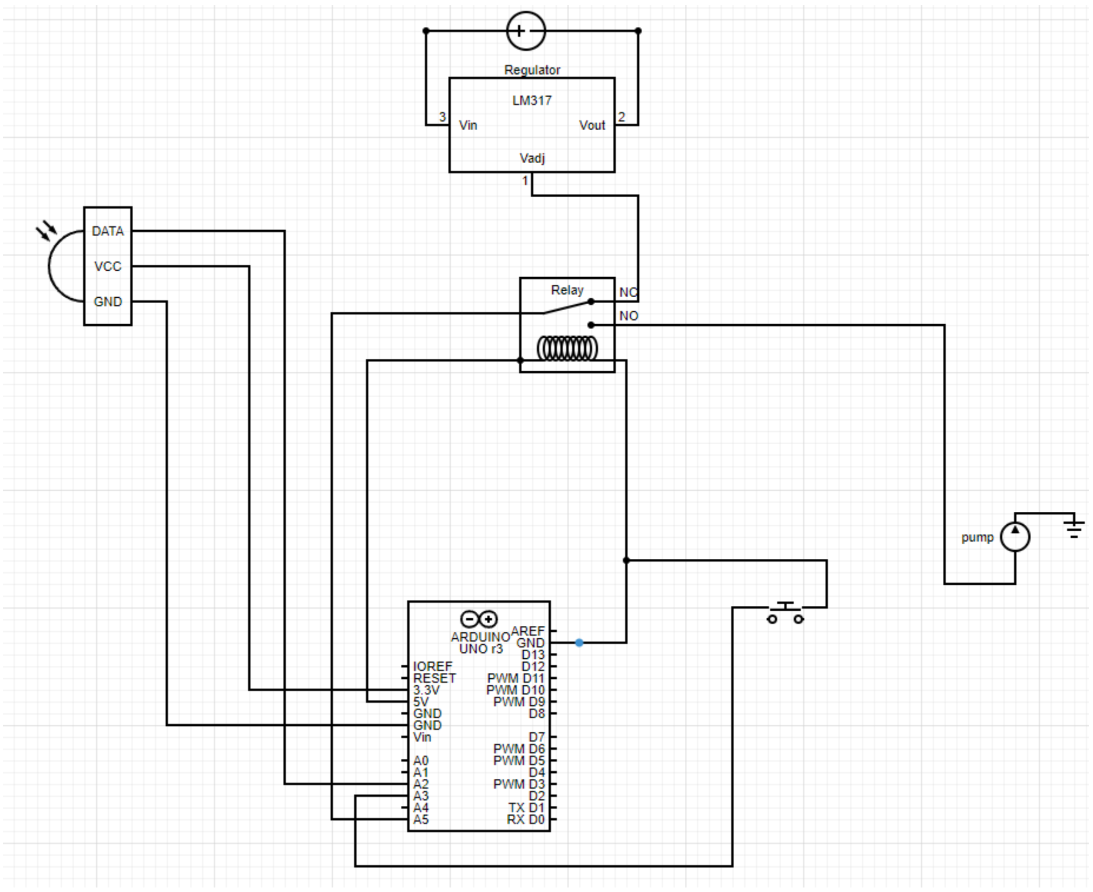


Figure 3.56: Fertilizer Controller-Circuit diagram for pump system

The use of the motion sensor is mainly because we consider that when the robot is in actual use, some mechanical problems may occur and we may need to approach it to solve the problem. When this happens, if the water pump continues to work, there is a high probability that chemical fertilizers will be sprayed on the body of the staff. So I chose HC-SR501 Motion Sensor to complete this job, which is the one in Table 3.2. Through Arduino coding, let HC-SR501 perform repeated detection. If a moving heat source object enters the detection range, it will send a digital signal to make the relay stop supplying power to the pump. If the object continues to be within this range, it will keep refreshing this pause. time until the object leaves the detection range. I set this range to a radius of 3.5m, which is a bit farther than our spray range. This data comes from Table 3.2.

Since we need a sensor in the tank to detect the liquid level of the fertilizer, when the fertilizer is used up, the robot needs to stop the water pump and return to the hub for replenishment, so I chose the Float Switch from Omabeta to accomplish this goal, which is shown in Table 3.2 of them. Since we need a sensor in the tank to detect the liquid level of the fertilizer, when the fertilizer is used up, the robot needs to stop the water pump and return to the hub for replenishment, so I chose the Float Switch from Omabeta to accomplish this goal, which is shown in Table 3.2 of them. When the fertilizer in the tank is sufficient, the float switch will lift up and send a signal to the arduino to make the pump continue to work. When the fertilizer is insufficient and the liquid level drops, the float switch will sink and stop the pump from working. After testing, the pump will stop working only when the liquid level is less than 1 cm, preventing unnecessary waste of fertilizers and ensuring work efficiency. This set of logic is in Figure 3.55 of the block diagram.

Ardunio is used as the main information processing tool in this fertilizer system, as we can see in the circuit diagram Figure 3.56. Arduino is connected to motion sensor, relay and float switch. When the signal of the float switch is sufficient and the signal of the motion sensor is no one is approaching, the Arduino sends the relay signal to continuously power the pump to make it work. When any of the above two signals changes, Arduino will stop the relay from supplying power and the pump will stop working.

3.3.0 Design of Core Functions

The specific designs of each component needed to complete our project goals can be read below. This section will address the relevant requirements, constraints and performance criteria of each core component. In addition the viability of possible options will be addressed along with notes on challenges faced and the solutions found.

3.3.1 Navigation Perception

3.3.1.1 Maneuver Strategy

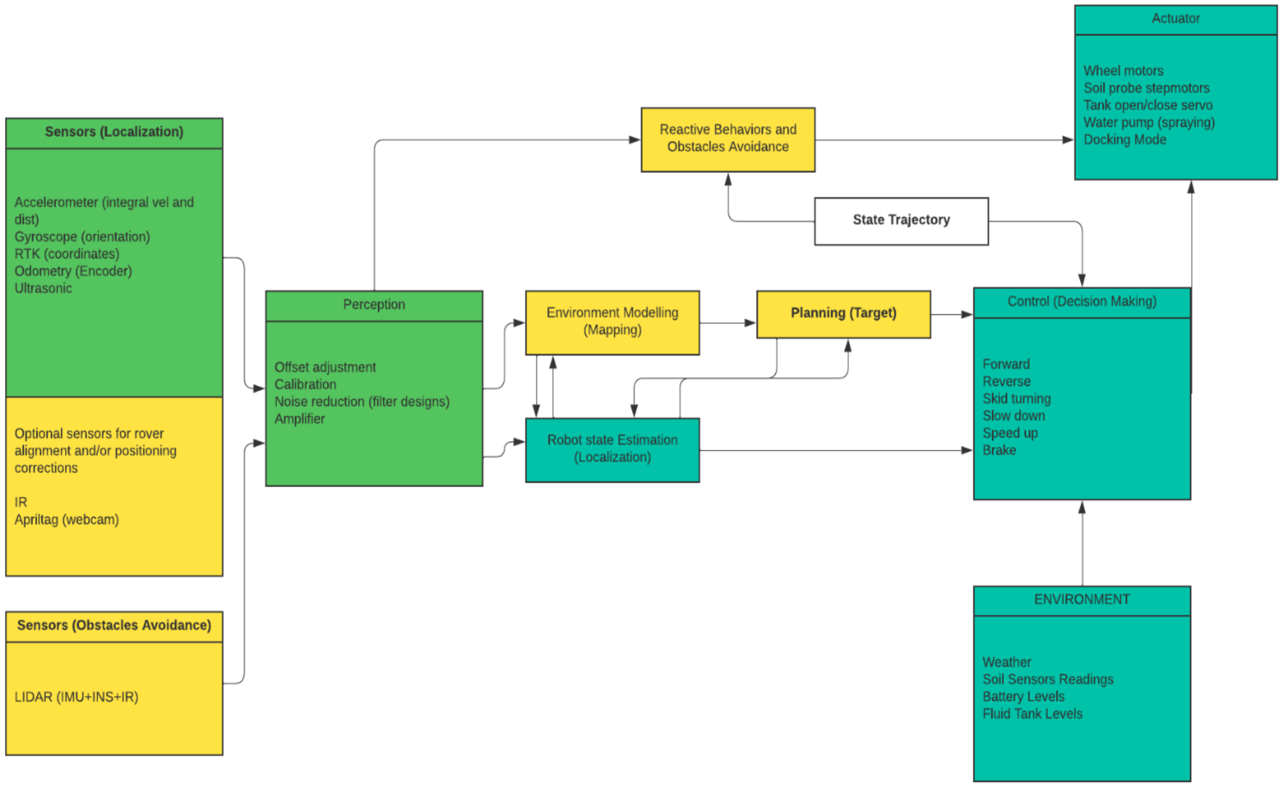
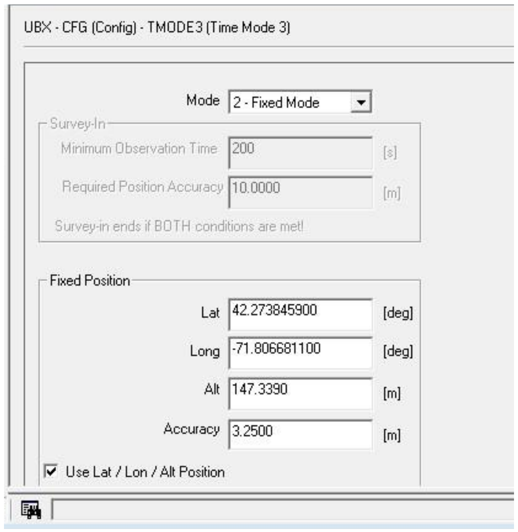


Figure 3.57(a): Navigation Strategy Feedback Loop


UBX - CFG (Config) - TMODE3 (Time Mode 3)

Mode: 2 - Fixed Mode

Survey-In
Minimum Observation Time: 200 [s]
Required Position Accuracy: 10.0000 [m]
Survey-in ends if BOTH conditions are met!

Fixed Position
Lat: 42.273845900 [deg]
Long: -71.806681100 [deg]
Alt: 147.3390 [m]
Accuracy: 3.2500 [m]

☑ Use Lat / Lon / Alt Position

Figure 3.57(b): Navigation Coordinates Survey Example (Exit door bench on UH Boynton Street)

In the requirement of the field test, the base starting point is given as a pre-survey coordinate in Figure 3.57(b) as well as the subsequent checkpoints. Figure 3.57(a) depicts the feedback loop that links the rover's sensor (perception) with the actuators (drive motor, soil probe and pump). The strategy was previously tested in RBE3002 turtlebot3 ROS Kinetic Gazebo/RVIZ simulation. Given the time limit, as the RBE team only has less than three weeks to achieve a certain level of autonomy, the proposed plan is to rotate the rover to be in the same line as the checkpoint (using magnetic compass) and go straight (using RTK) in conjunction with distance feedback (using hall effect encoder).

3.3.1.2 RTK (Real-Time Kinematics or Differential GPS)

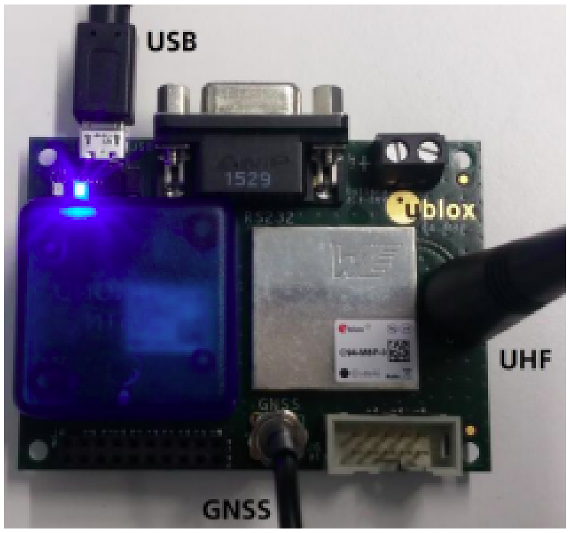


Figure 3.58: The two boards are identical. One is selected as the BASE and one as ROVER

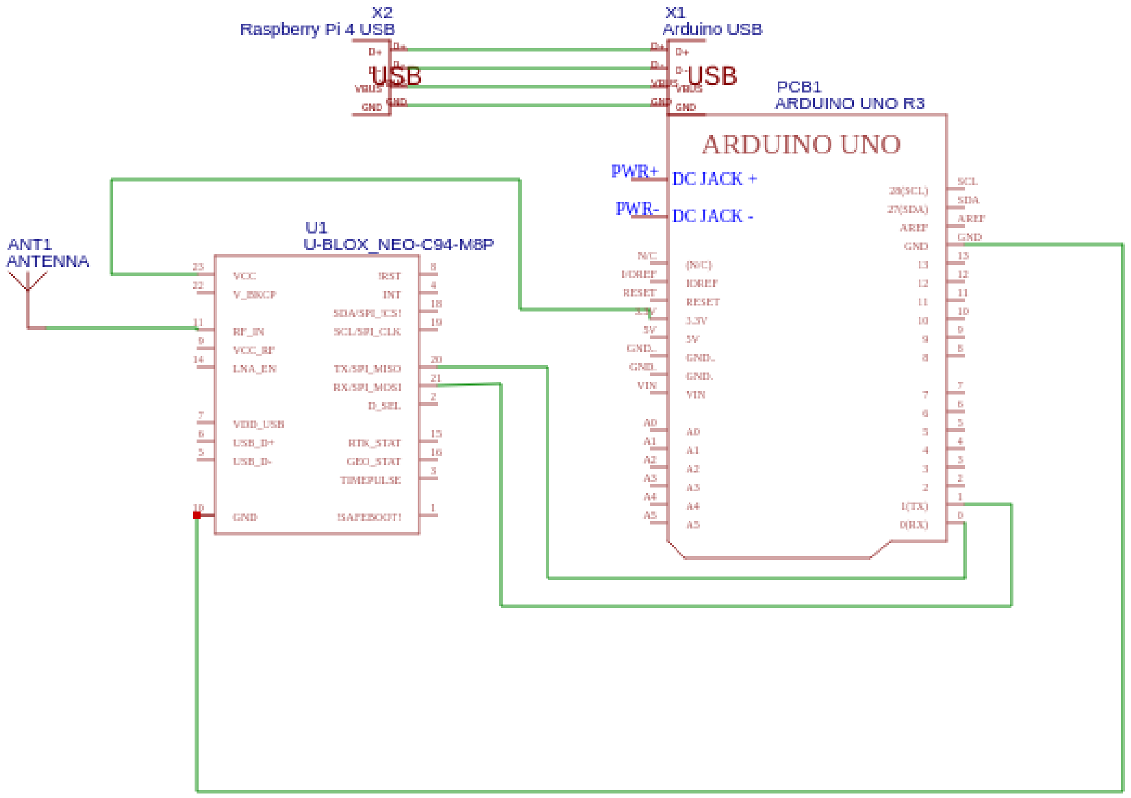
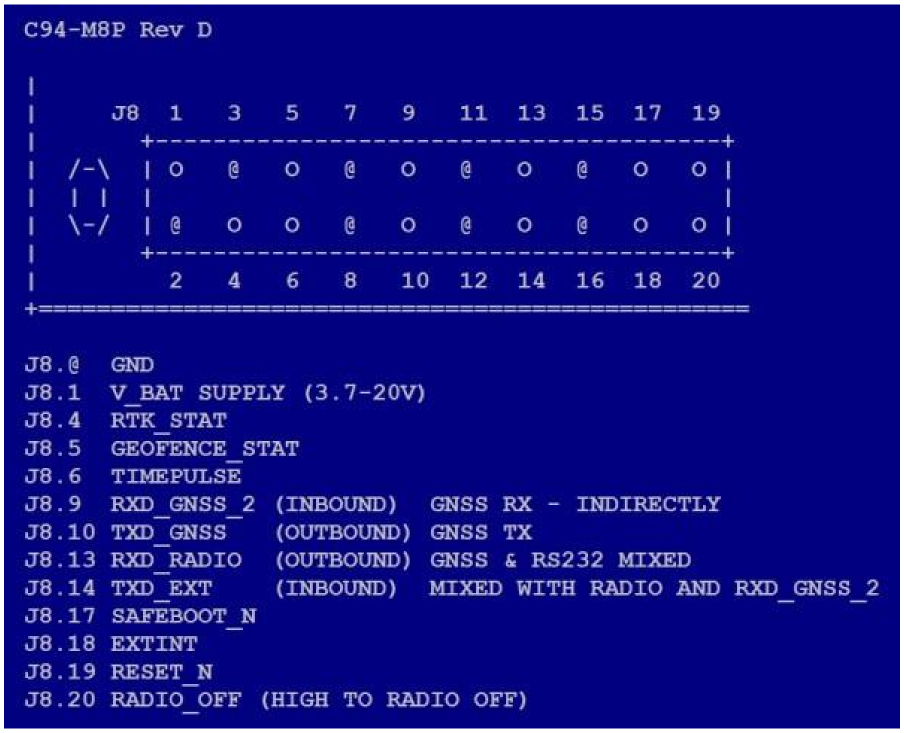


Figure 3.59(a): Final Circuit Design for RTK Interface to Arduino (Output Serial to Raspberry Pi)



```
C94-M8P Rev D

|
|     J8  1   3   5   7   9   11  13  15  17  19
|       +--------------------------------------+
|  /-\  | o   @   o   @   o   @   o   @   o   o |
|  | |  |                                       |
|  \-/  | @   o   o   @   o   @   o   @   o   o |
|       +--------------------------------------+
|         2   4   6   8   10  12  14  16  18  20
+=================================================

J8.@   GND
J8.1   V_BAT SUPPLY (3.7-20V)
J8.4   RTK_STAT
J8.5   GEOFENCE_STAT
J8.6   TIMEPULSE
J8.9   RXD_GNSS_2 (INBOUND)  GNSS RX - INDIRECTLY
J8.10  TXD_GNSS   (OUTBOUND) GNSS TX
J8.13  RXD_RADIO  (OUTBOUND) GNSS & RS232 MIXED
J8.14  TXD_EXT    (INBOUND)  MIXED WITH RADIO AND RXD_GNSS_2
J8.17  SAFEBOOT_N
J8.18  EXTINT
J8.19  RESET_N
J8.20  RADIO_OFF (HIGH TO RADIO OFF)
```

Figure 3.59(b): RTK Interface from U-Center Software (J8 Pin 9 RX (Interface to Uno TX) and Pin 10 TX (Interface to Uno RX))

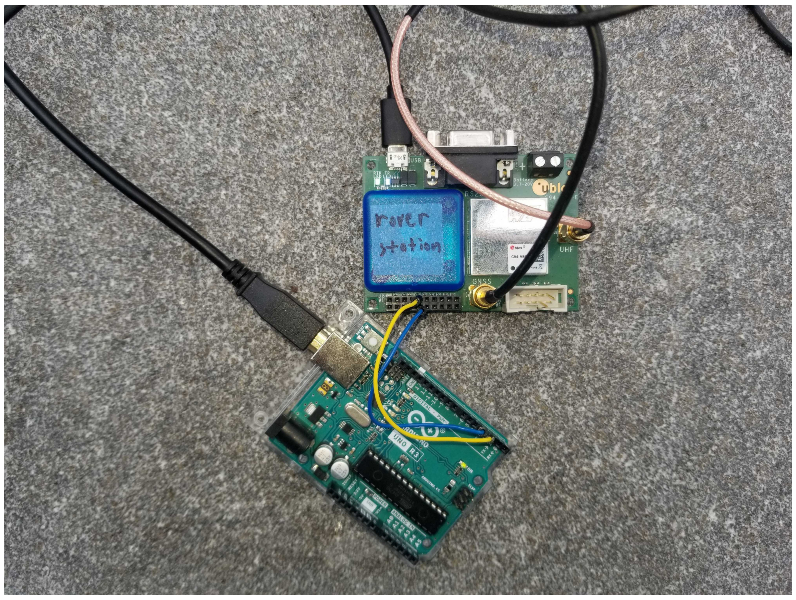


Figure 3.59(c): Sample Rover Setup

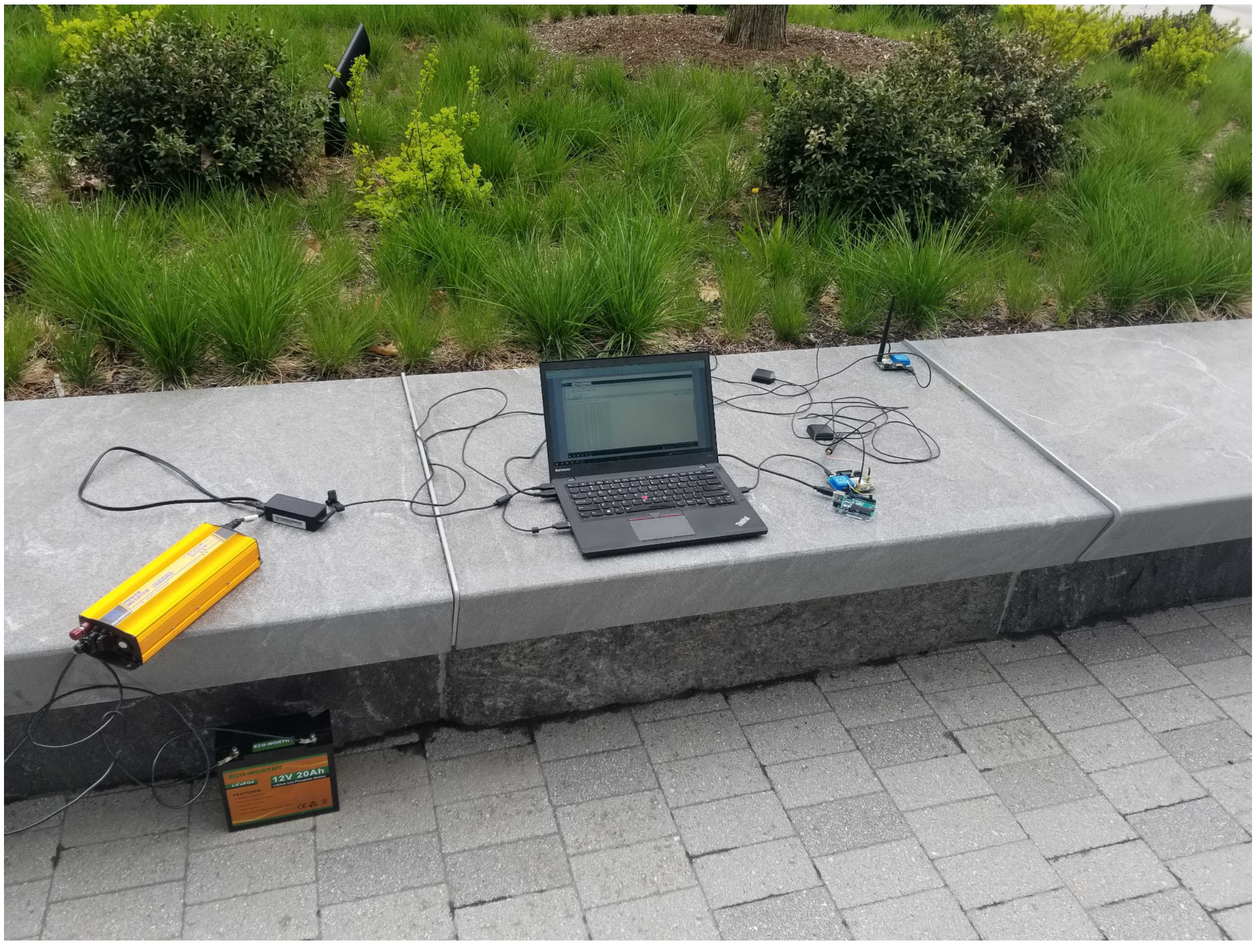


Figure 3.59(d): Setup for sampling stationary rover

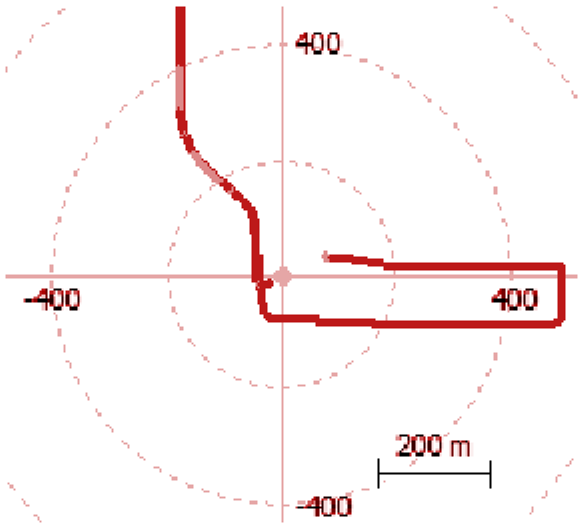


Figure 3.60(a) : RTK Serial Output Plot (Note: the plotter unit should be in foot instead of meter)

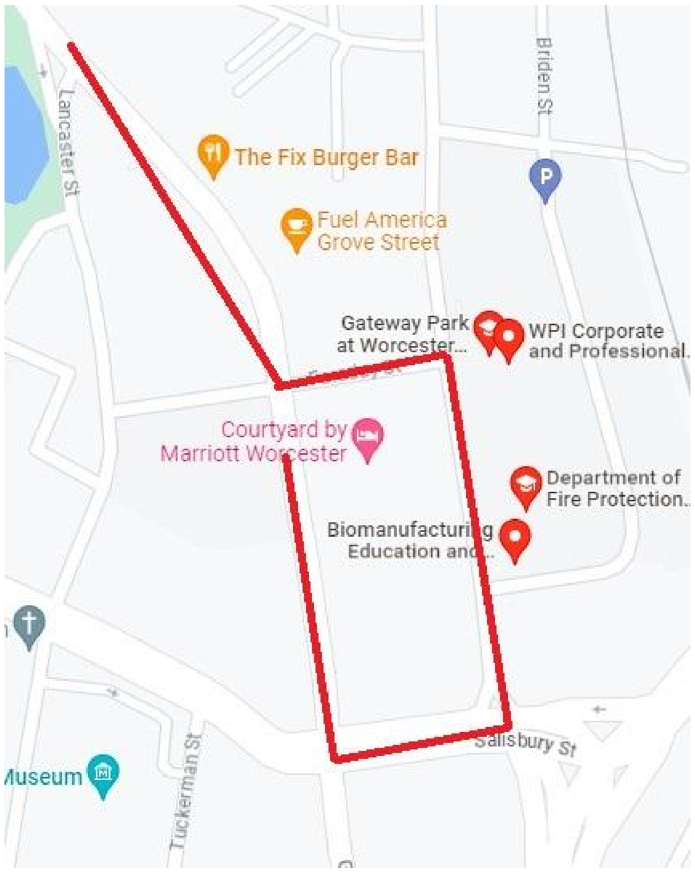
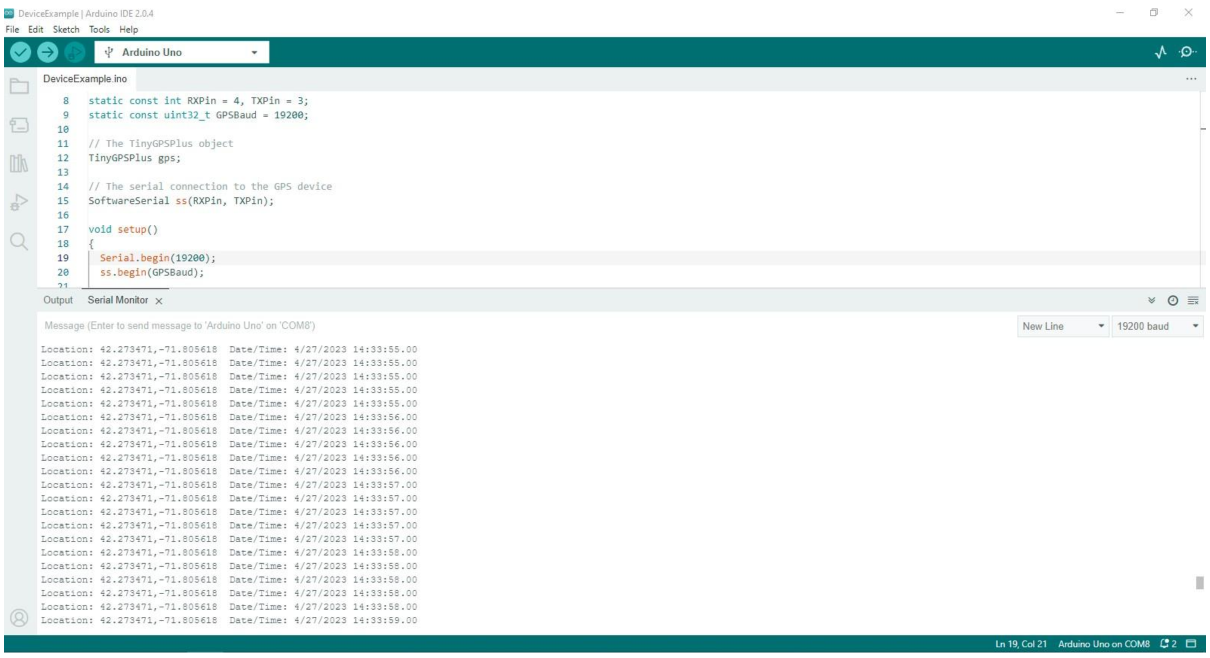

Briden St
Lancaster St
The Fix Burger Bar
Fuel America Grove Street
Gateway Park at Worcester...
WPI Corporate and Professional.
Courtyard by Marriott Worcester
Department of Fire Protection.
Biomanufacturing Education and...
Salisbury St
Tuckerman St
Museum

Figure 3.60(b) : Comparison to Google Map (red line indicates an estimate of the **writer's walking path**)


DeviceExample | Arduino IDE 2.0.4
File Edit Sketch Tools Help
Arduino Uno
DeviceExample.ino
Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM8')
New Line
19200 baud
Ln 19, Col 21 Arduino Uno on COM8 2

```
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 19200;

// The TinyGPSPlus object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

void setup()
{
  Serial.begin(19200);
  ss.begin(GPSBaud);
```

```
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:55.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:55.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:55.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:55.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:55.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:56.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:56.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:56.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:56.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:56.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:57.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:57.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:57.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:57.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:57.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:58.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:58.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:58.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:58.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:58.00
Location: 42.273471,-71.805618  Date/Time: 4/27/2023 14:33:59.00
```

Figure 3.60(c) : Sample reading of a stationary rover (no movement) from Serial Port

The team inherited a set of engineering development GPS modules shown in Figure 3.58 with RTK capability from the previous year. There are two boards; one is used as a base and another as a rover communicating with each other over ultra high frequency radio waves at 915MHz. In order to distinguish between the two boards, it requires legacy firmware flash several times before the chip is usable. In addition, the previous team used a wrong version of (NEO-M8P) firmware on the C94-M8P. To fix this, the safeboot pin on J8 connector has to be pulled low to reset the flash ROM back to factory default shown in Figure 3.59(b). The GNSS receiver is automatically connected upon updating to firmware version 1.40. This setup goal is to pass the data from the RTK module to Arduino Uno via UART interface (TX/RX). The Arduino then passes serial output via USB port to Raspberry Pi 4B shown in Figure 3.59(a). This is because there are two UART ports on the Raspberry Pi 4B but only one is available (the other is used for bluetooth) for RS485 CAN Hat for TalonSRX control.

Because the two boards work in pairs, it is necessary to customize the EEPROM (electrically erasable programmable read-only memory) on each module. Firstly, the Base module has to be set to 19200 baud rate (a level recommended by the manufacturer) and output only RTCM3 messages on UART(where the UHF antenna is attached to). In addition, the antenna UART receiver (RX) has to be disabled to avoid processing unnecessary U Blox binary messages. Moreover, once the configuration is finished, a SPI Flash needs to be saved using the manufacturer software. An interesting result was found that after a week of not powering the RTK module is that it will lose all the saved data. This is due to the extremely tiny charge on the floating gate as well as the magnetized aluminum rover frame. The Base operates UHF transmission at a rate of 1 Hz. In a newer RTK capable GPS module, most manufacturers only require one board (the Rover in this case) while the Base is substituted by signal towers available throughout the country (the closest one to WPI is Auburn tower).

Secondly, to set up the base the EEPROM needs to be set such that it only accepts RTCM3 messages at 1Hz on the UART antenna target(at 19200 baud rate to match the sender) while also sending NMEA (a standard GPS protocol out to the same UART) shown in Figure 3.59(c) and Figure 3.59(d). The reason is that UART has the maximum of one Master and one Slave. Our setup takes advantage of the remaining Slave Frame Protocol as a pick off point to Arduino UART port. In fact, only TX from the RTK module to RX on the Arduino Uno is required for this setup to work. The Arduino receives a stream of NMEA protocol messages and decodes it using the TINYGPS library. The result of the latitude and longitude plot from Microsoft Excel after saving the serial output from Arduino is shown in Figure 3.60(a) in comparison with the writer walking along the edge of the pavement from GoogleMap in Figure 3.60(b). Additionally, a sample Serial Output of the parsed latitude and longitude data is shown in Figure 3.60(c).

One drawback is that after walking past the Gateway high voltage electrical box the RTK receiver shows some interference/signal loss depicted in light red in Figure 3.60(a). The resulting accuracy was given at 6th decimal degree which is equivalent to 111 mm in metric units. Additionally, there were some deviations in the reading due to the survey time. In order to achieve significant improvement on the reading, the survey time (Base absolute coordinate) needs to be run at 24 hours or more. In this setup, however, the survey was run at only 15 minutes (due to weather) which resulted in a base tower standard deviation of 3.25 m.

3.3.1.3 Tilted-Compensated IMU

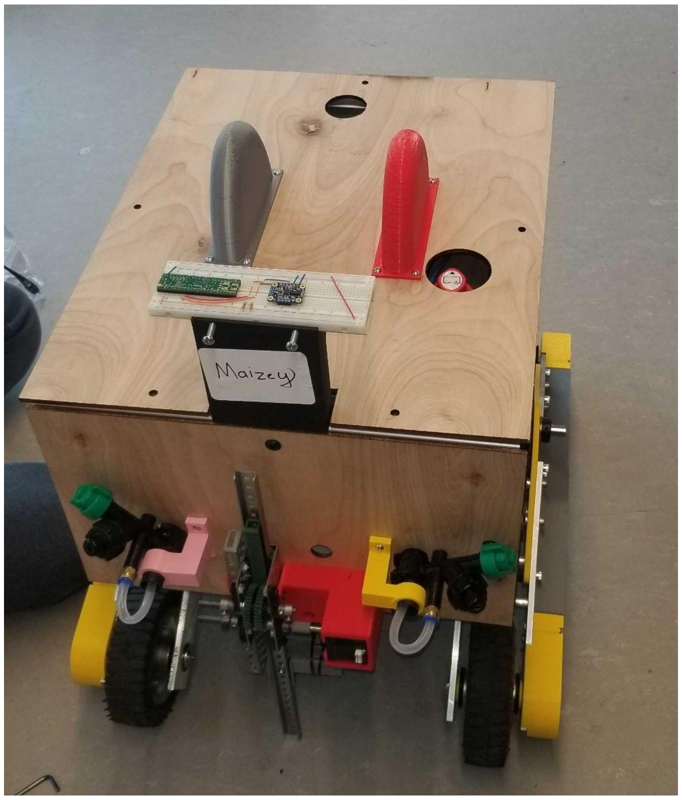


Figure 3.61 : Position of Gyro Module on the Robot

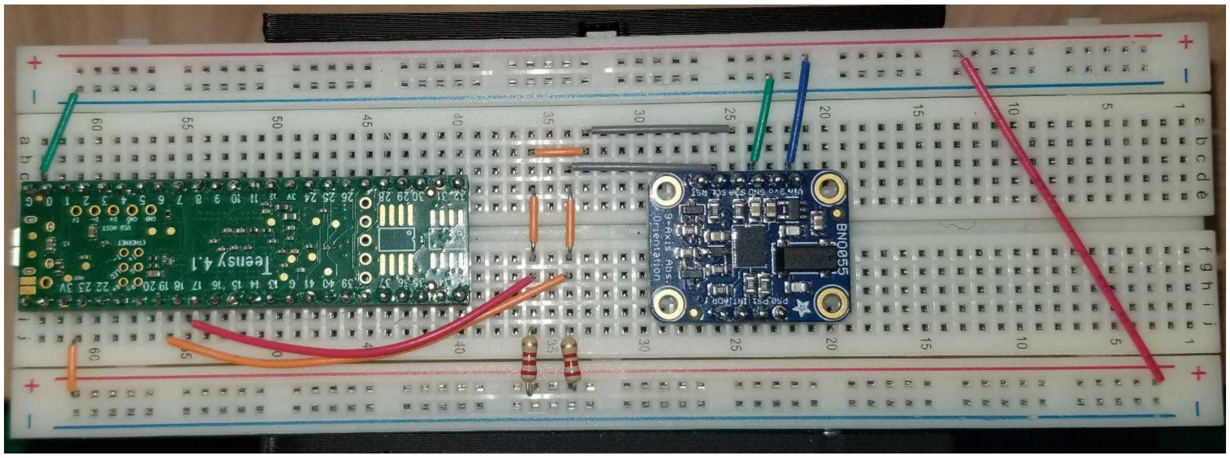


Figure 3.62: Circuit Implementation (cannot be soldered due to return contract)

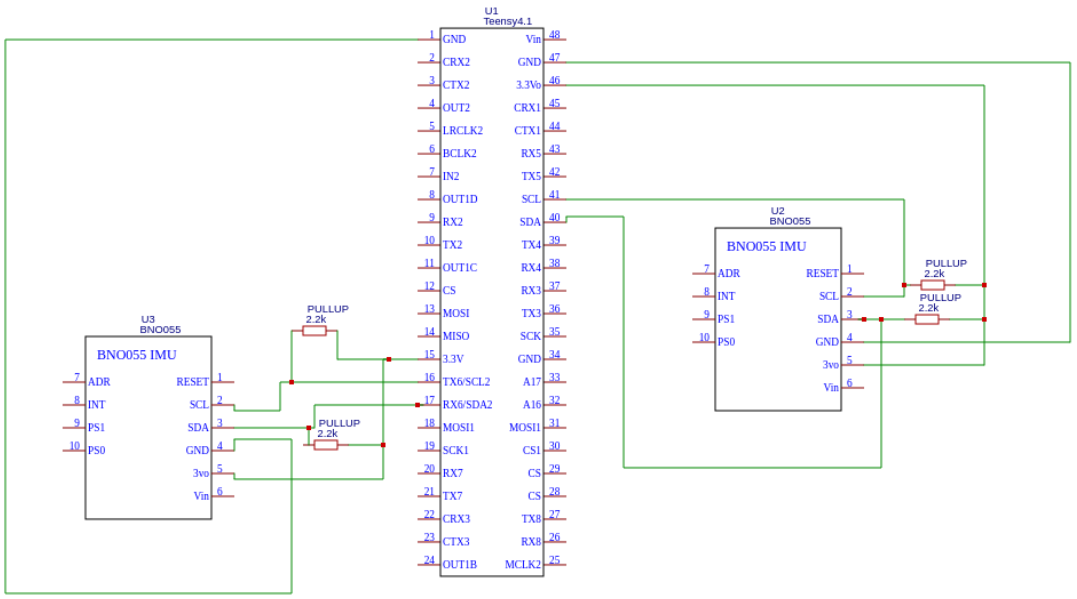


Figure 3.63(a): Final Teensy 4.1 and two IMU BNO055 (for average) Circuit Design

In order to navigate to a desired pose, the rover needs sensory perception to guide which direction its heading. The BNO055 is a 9-axis sensor that includes a magnetometer, accelerometer, and gyroscope, designed to provide orientation information for electronic devices.A magnetic compass on an IMU is used to determine the true heading. There are several challenges that the team faced during the testing process. Even though there is a rocker suspension on the robot, it is imperfect and causes the plane where the IMU is mounted to not be parallel to the ground. Hence, the magnetometer reading will be substantially off when it is titled from the operating plane. Without the tilt compensation, compass heading varies substantially, sometimes up to 100 degrees on a 20 degree tilt angle. The solution is to utilize the gyroscope pitch and roll to trick the IMU as if it is still parallel to the ground using software using the relationship shown in Figure 3.63(b).

Xhorizontal = X*cos(pitch) + Y*sin(roll)*sin(pitch) – Z*cos(roll)*sin(pitch)
Yhorizontal = Y*cos(roll) + Z*sin(roll)

Figure 3.63(b): Tilt Compensation Equation from Gyroscope

In addition, the true north can be achieved using an offset of -13.88 degree(declination rate) from the magnetic north for Worcester, Massachusetts. Another factor to consider is the distortion from hard and soft metal from the frame as well as electronics. This is taken care of by using Bosch's internal sensor fused BNO055 library to dynamically calibrate the surrounding on the rover and the best position where interference is minimal is shown in Figure 3.61. In addition, there are two 2-gauge wires on the top frame which could drive as much as 120A at full skid turning power. The magnetic field generated caused a huge setback which is why two IMU(on the same plane) are used in the signal processing stage to get an average shown in Figure 3.63. Pullup 2.2k resistors are used to bypass the weak internal 10k of the Teensy in order to achieve better reading from the I2C interface. Because these two IMU were borrowed from the department, the team was not allowed to solder it permanently onto the board shown in Figure 3.62. In terms of how well the magnetometer works, it depends on various factors such as the calibration, the placement of the sensor, and the external magnetic interference. When properly calibrated and placed, the BNO055 magnetometer can provide accurate orientation information with minimal drift over time.

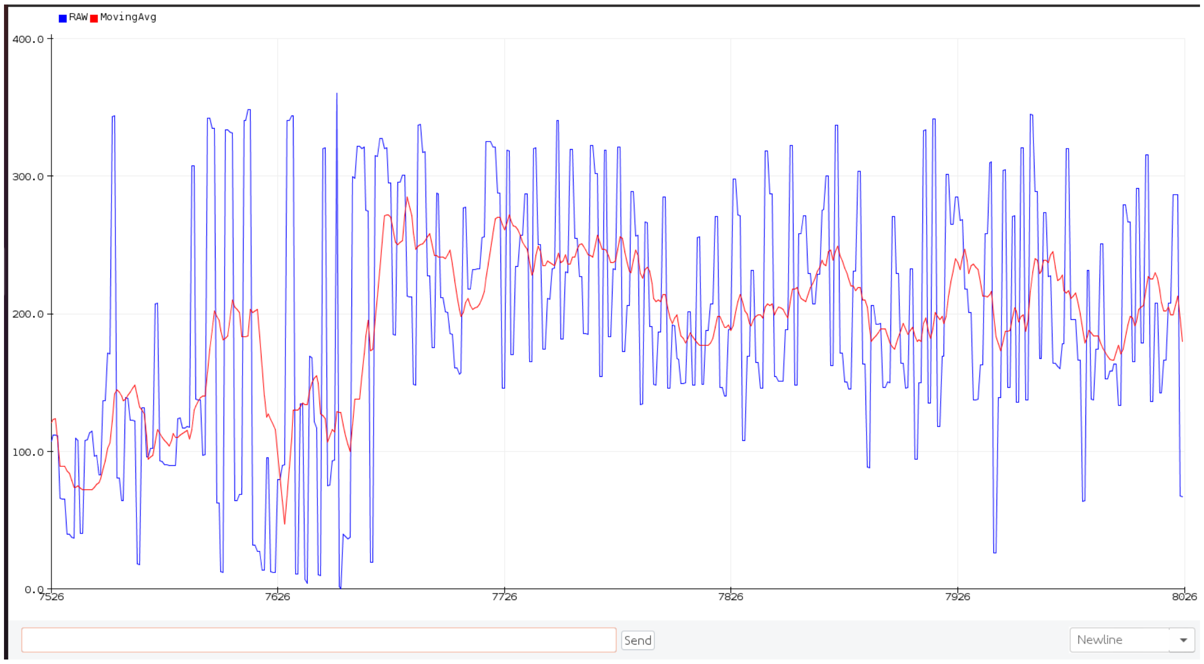


Figure 3.64: Raw and Moving Average of Magnetometer Compass Heading (Pre-signal processing)

For all of the figures, raw data is overlaid with a moving average (20 points) to increase consistency. Figure 3.64 depicts the compass heading (true north) before tilt-compensation and Kalman

Filter. Figure 3.64 is plotted with a hands on test moving the IMU around the a constant axis by hand while tilting between +/- 20 degree from the ground plane. Any tilt will change the reading drastically, sometimes up to 100 degrees. This is because the BNO055 magnetometer measures the earth's magnetic field. The Hall Effect occurs when a current-carrying conductor is placed in a magnetic field perpendicular to the current flow. The resulting voltage produced across the conductor is proportional to the strength of the magnetic field. In a Hall Effect magnetometer, a thin strip of conducting material is placed in a magnetic field and a current is passed through it. The resulting voltage across the strip is proportional to the magnetic field strength and can be measured to determine the field strength and direction. Because the BNO055 magnetometer has a sensitivity of up to 1 µT (micro Tesla), it is necessary to calibrate the sensors on the rover aluminum frame (hard and soft metal) and save the calibration data to the microcontroller to avoid cold start of the sensor.

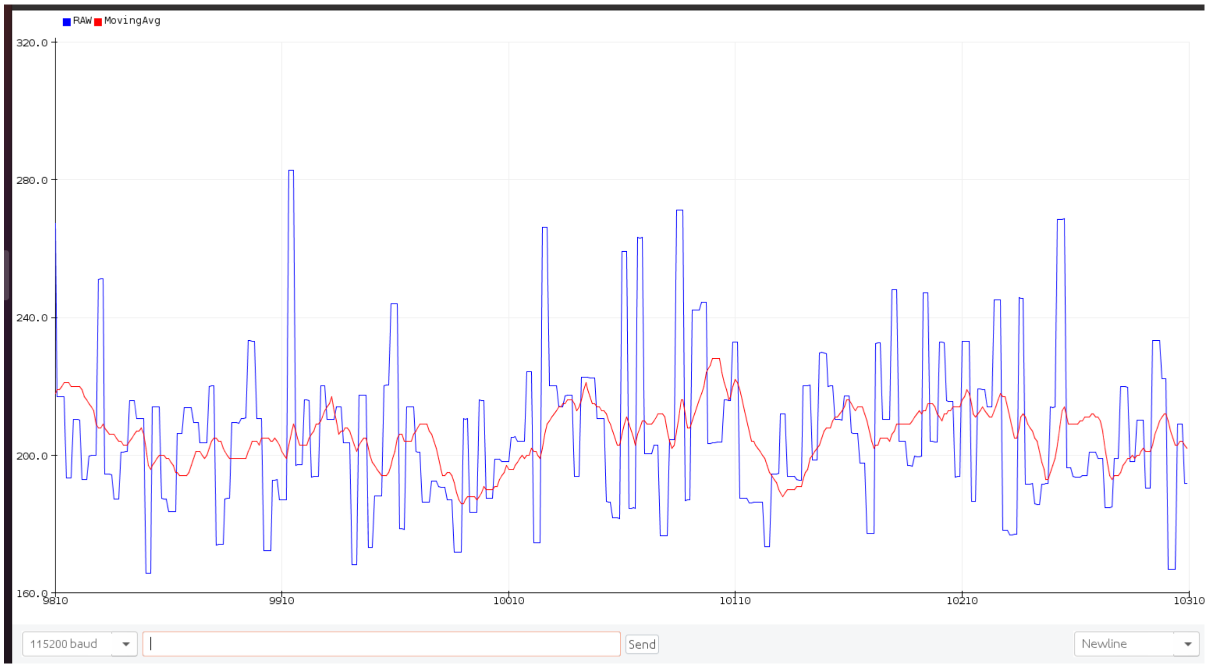


Figure 3.65: Raw and Moving Average of Magnetometer Compass Heading (After tilt-compensation)

After the tilt-compensation algorithm is implemented, the reading is significantly better, reducing 100 degree deviation to only around 10 to 15 degree. However, the issue with magnetic field interference from the two-gauge wire still causes drifting in practice. This is because the calibration data does not take into account the varying current of the exposed wire(therefore varying magnetic interference) which can go up to 120A. This will be fixed using a one-dimension Kalman Filter.

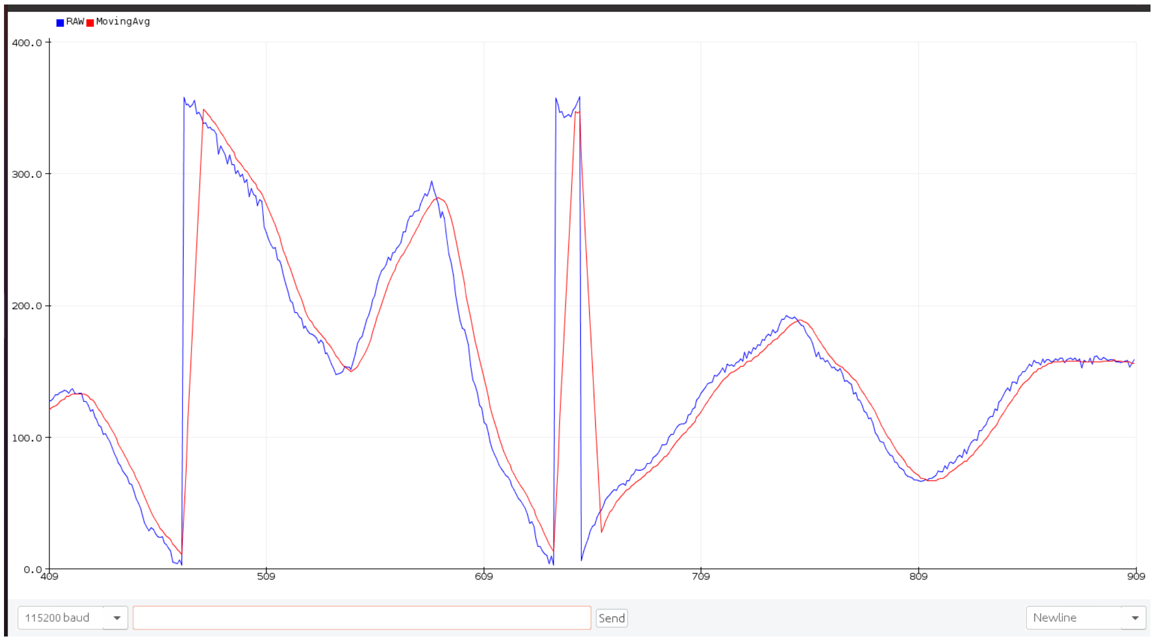
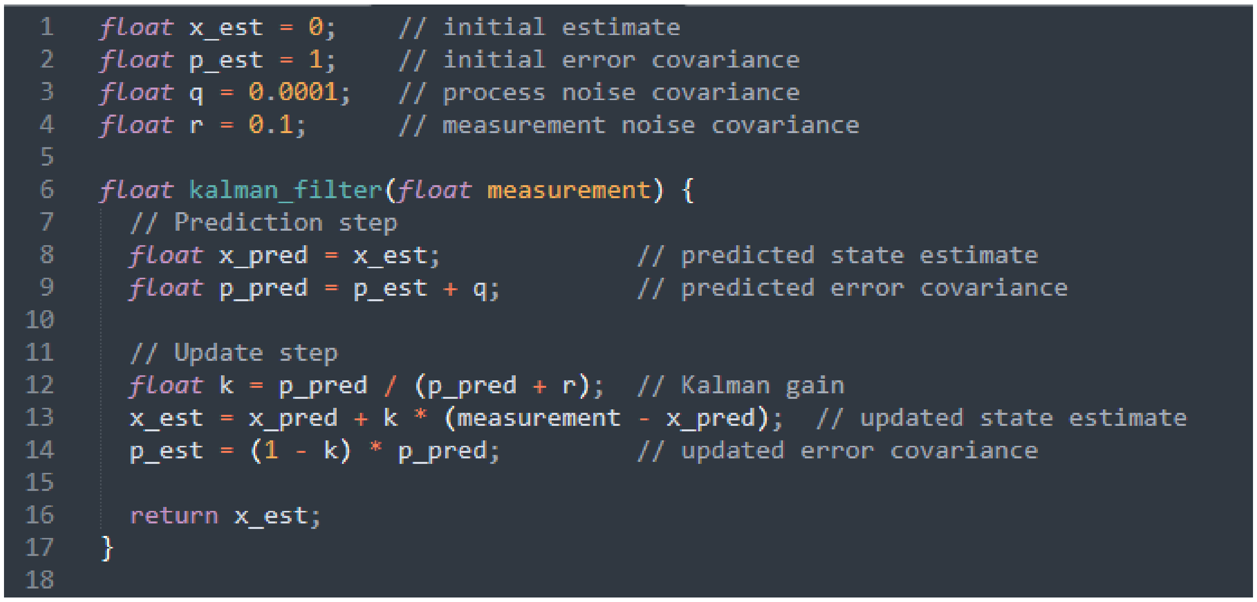


Figure 3.66(a): Raw and Moving Average of Magnetometer Compass Heading (After Kalman Filter)



```
float x_est = 0;     // initial estimate
float p_est = 1;     // initial error covariance
float q = 0.0001;    // process noise covariance
float r = 0.1;       // measurement noise covariance

float kalman_filter(float measurement) {
  // Prediction step
  float x_pred = x_est;             // predicted state estimate
  float p_pred = p_est + q;         // predicted error covariance

  // Update step
  float k = p_pred / (p_pred + r);  // Kalman gain
  x_est = x_pred + k * (measurement - x_pred);  // updated state estimate
  p_est = (1 - k) * p_pred;         // updated error covariance

  return x_est;
}

```

Figure 3.66(b): An oversimplified version of Kalman Filter on Teensy Code

On Teensy 4.1, a filter function implements the Kalman filter algorithm. The function takes the current measurement as input and returns the filtered value as output. Initially, the process noise covariance and measurement noise covariance are set to 0.0001 and 0.1, respectively. The initial state estimate and error covariance are set to 0 and 1, respectively. Through trial and error, it is found that the process noise covariance works best at the value of 0.0015 while the measurement noise covariance works best at 0.3. The function is set inside the loop, the code reads the sensor value, updates the Kalman filter with the new measurement using the function, and prints the filtered value to the serial communication which is passed on to the Raspberry Pi 4B target. A sample output after the Kalman filter is shown in Figure 3.66(a). An oversimplified logic is depicted in Figure 3.66(b). As a result, the compass heading has an accuracy of up to 2 degrees when versus against smartphone apps.

3.3.2 Soil Probe and Slider NPK Sampling

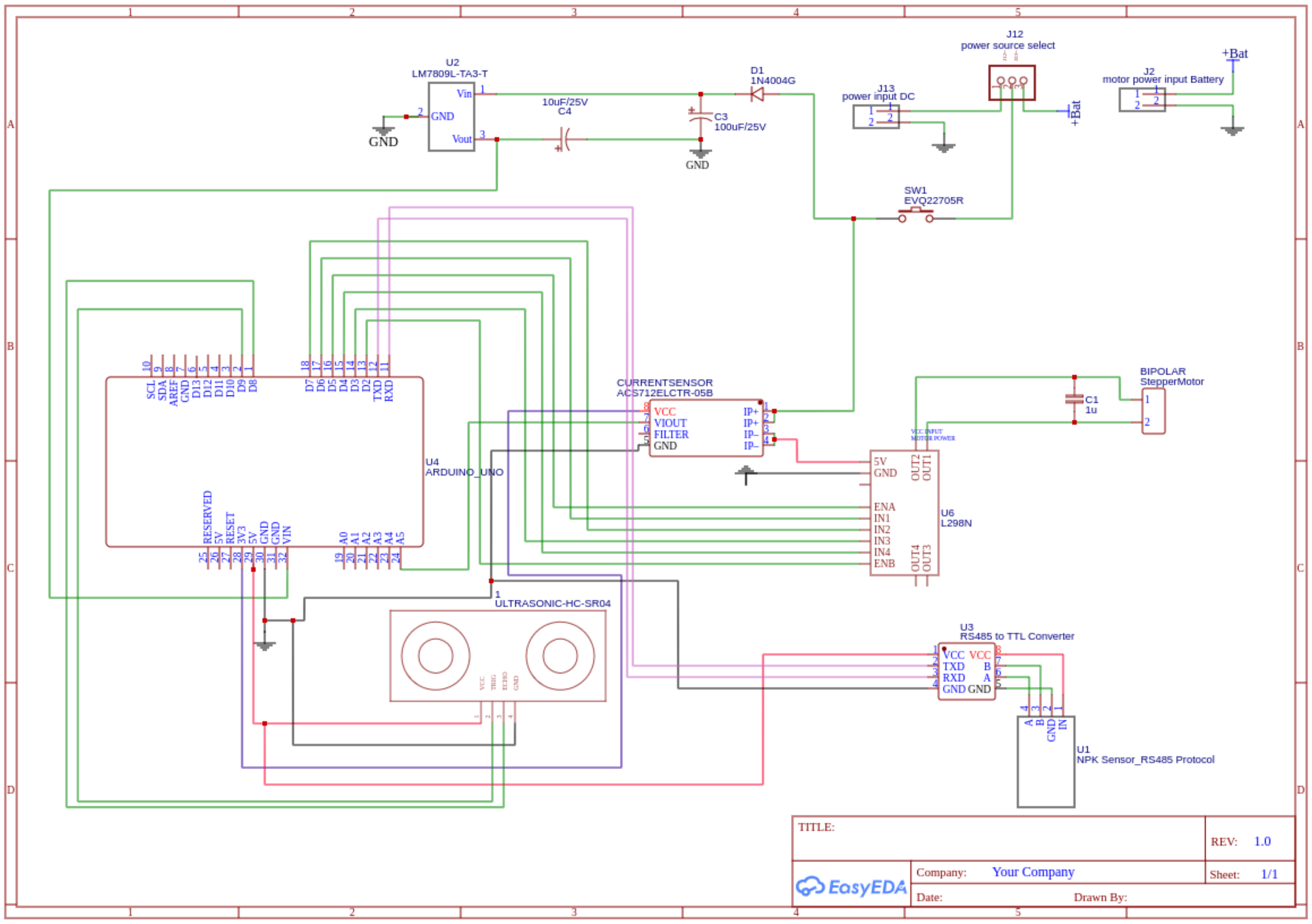


Figure 3.67: Final Circuit Design Slider+NPK+Ultrasonic+Current Sensor

The goal of the slider mechanism is to sample the soil data using the NPK sensor as an end-effector. The design challenge is to control the actuator such that on rough terrain the displacement between the ground and the slider's end-effector varies. Additionally, in case the end effector is directly on hard objects such as rocks or concrete, there should be a sensory perception to inform this module that the stepper has been stalled and should not penetrate the soil any further. Therefore, the feedback loop includes stepper motor encoder, ultrasonic and a current sensor. The final design is shown in Figure 3.68(a). The reason for the team not showing the actual picture is that the poor design of the slider mechanism caused permanent damage to the system during the test drive.

A schematic of the slider sub-system is depicted in Figure 3.67. The role of the stepper motor (bipolar--bidirectional) encoder is to limit the range of motion (200 steps in a revolution). In this case, due to the ineffective mechanical design, the slider itself can only actuate linearly at ½ inch. This indicates that the stepper motor has a 6:1 gear ratio to have a range of motion at only 120 degrees (a third of a full rotation). Given some tolerance (25% or 30 degrees) the stepper limits itself to rotate between (0 degree--slider at its highest point and 90 degree--at its deepest penetration) which is equivalent to 0.375 in range of motion in total.

The role of the ultrasonic sensor is to adjust the range according to the incline of the terrain (0<experimental range< 0.375 in) shown in Figure 3.68(b). There are several limitations to ultrasonic sensors. Firstly, the module our team used only works between 15 cm to 2m. The lower tolerance forced the design to mount the ultrasonic quite high above the ground which introduced another drawback. The sensor is only suitable for detecting at a perpendicular angle to the surface. From our experiment, the ultrasonic only had a detection angle of less than 30 degree to the surface while a perfectly perpendicular surface yielded the best result. In addition, environmental factors such as temperature, high winds and humidity can interfere with accuracy of the sensor readings. Furthermore, the presence of other ultrasonic sensors within the reflecting range can cause interference and unreliable readings. This makes it impossible to mount two ultrasonic sensors and get the average. Lastly, the ultrasonic has difficulty in detecting objects with irregular shapes as well as soft materials such as mushroom or flower.

The current sensor is used to prevent the stepper motor from stalling if it happens to penetrate a rocky surface. From the experiment the 5V stepper motor stalled out at 1.5A. Hence, provided from tolerance, our team set a limit on the current sensor to be 1.2A (around 80% of the stall current) as a breakpoint to terminate the slider operation. One issue with the analog Hall Effect current sensor is that there is random overshoot and excess settling time since there is capacitive resistance as well as interference from the magnetized aluminum frame. Hence, a moving average is used to smooth out the current sensor reading.

The following are the **read only** sensor data to the Arduino Uno microcontroller. The NPK sensor communicates with the Arduino using UART TX/RX interface. However, due to its being a RS485 type(designed for long distance), an adapter is used to convert the 485 protocol to TTL before passing on to the Arduino. The ultrasonic communicates with the Uno via digital pin. The current sensor communicates using an analog pin. Rotary encoders typically have a shaft connection and produce pulses in proportion to the shaft's rotation. The shaft's rotational speed may be determined using these pulses, and it can also be swiftly and accurately regulated using Pulse Width Modulation (PWM) technology. The encoder detects changes in the shaft's position and transforms those changes into electrical signals that may be utilized to determine the rotation's speed and direction.

The following are the **write only** data to the Arduino microcontroller. The actuation of the stepper motor is controlled by PWM. The Raspberry Pi 4B sends and receives the request over serial protocol to the Arduino Uno. The Arduino Uno has four states: 1. Pending mode (handle serial waiting for the Raspberry Pi 4B command --- the slider is at the highest position) 2. Testing mode (moving down to the desired pose based on the ultrasonic reading) 3. Success mode (the slider successfully penetrates the soil and sends data to the Raspberry Pi. Then switch back to Pending mode) 4. Terminate mode (upon entering testing mode and the NPK sensor hits the rock (current sensor on the stepper motor exceeds 1.2A), the switch case terminates and sends the error “Hit the Rock!” code back to the Raspberry Pi).

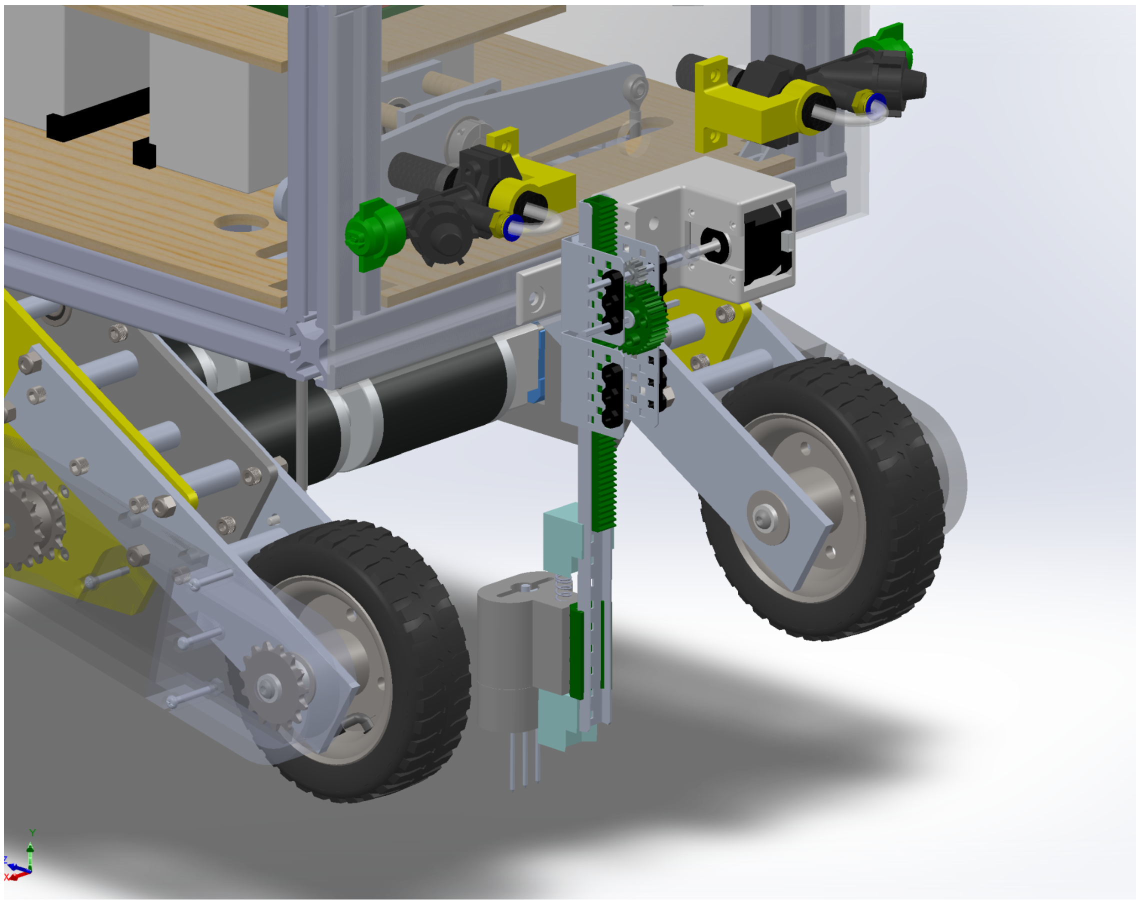


Figure 3.68(a): Soil Probe Model

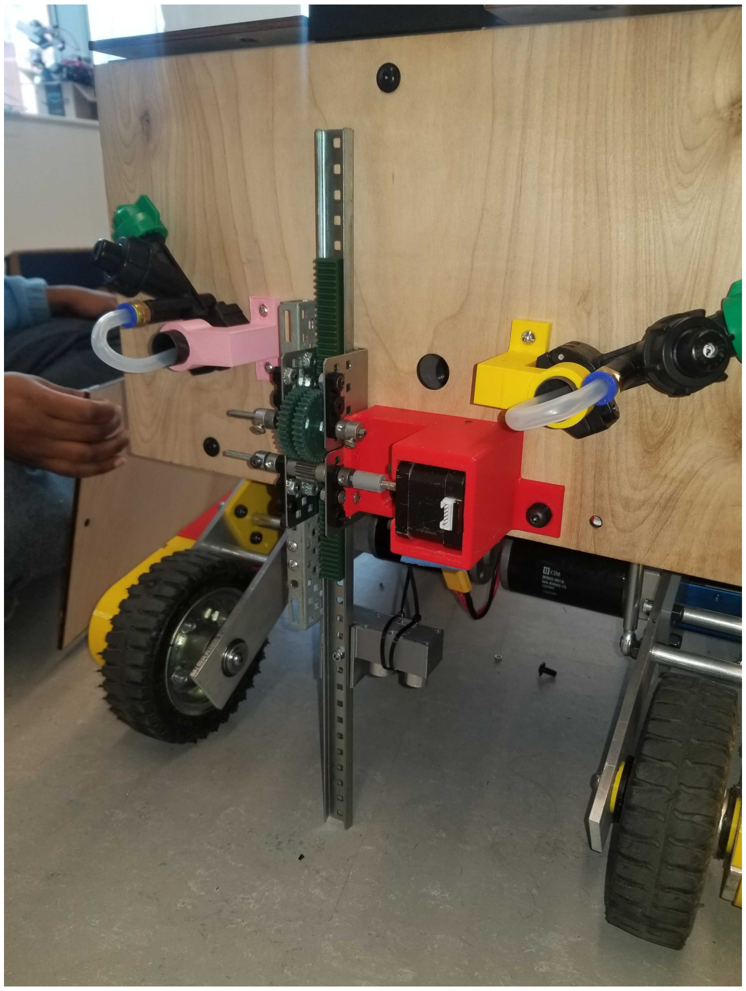
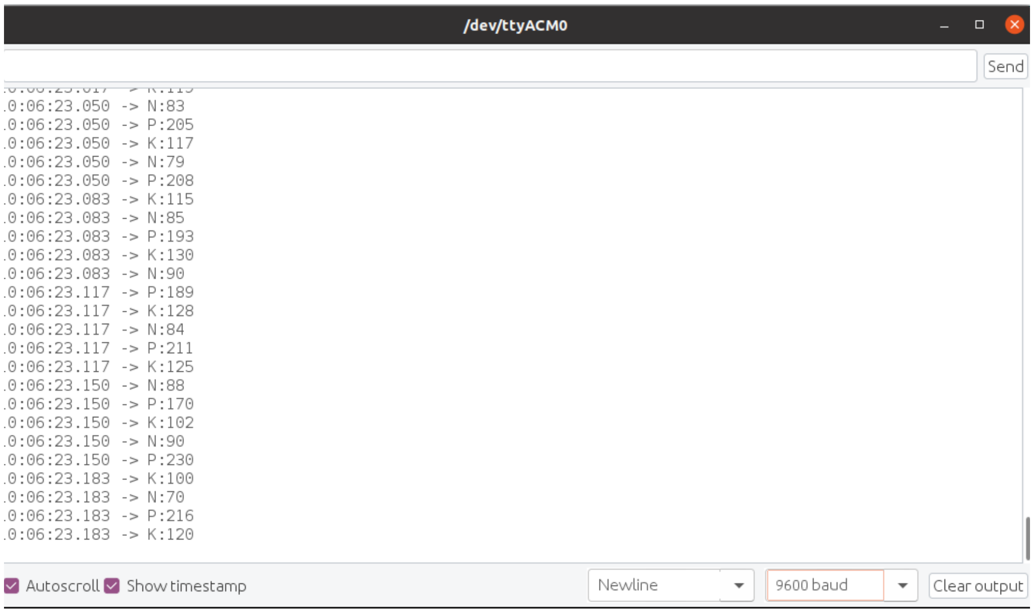


Figure 3.68(b) : Slider Front View

Request (Address code Nitrogen = 0x01 start and end 0x1f)	{0x01,0x03, 0x00, 0x1f, 0x00, 0x01, 0xb5, 0xcc}
Response (the sixth Hex number is content value)	{0x01, 0x03, 0x00, 0x1f, 0x00, **0x20**, 0x79, 0x9F}
Decoding Hex Base 16 to Decimal Base 10	N reading= **0020 H**(hexadecimal) = 32 (Decimal) = 32 mg/kg

Figure 3.69(a): Decoding MODBUS and converting result to float number base 10



```
.0:06:23.050 -> N:83
.0:06:23.050 -> P:205
.0:06:23.050 -> K:117
.0:06:23.050 -> N:79
.0:06:23.050 -> P:208
.0:06:23.083 -> K:115
.0:06:23.083 -> N:85
.0:06:23.083 -> P:193
.0:06:23.083 -> K:130
.0:06:23.083 -> N:90
.0:06:23.117 -> P:189
.0:06:23.117 -> K:128
.0:06:23.117 -> N:84
.0:06:23.117 -> P:211
.0:06:23.117 -> K:125
.0:06:23.150 -> N:88
.0:06:23.150 -> P:170
.0:06:23.150 -> K:102
.0:06:23.150 -> N:90
.0:06:23.150 -> P:230
.0:06:23.183 -> K:100
.0:06:23.183 -> N:70
.0:06:23.183 -> P:216
.0:06:23.183 -> K:120
```

Figure 3.69(b): Soil NPK Sample Reading

To interface Arduino with the NPK sensor, our team used Modbus commands shown in Figure 3.69(a). The commands for Modbus enable the control of Modbus devices to perform different actions, such as changing values in registers, reading data from I/O ports, or requesting the device to send values from its registers. These actions can be performed by sending a Modbus command that includes the address of the intended device (ranging from 1 to 247), also known as an inquiry frame. Only the device with the matching address will respond and act on the command. The NPK Sensor is designed with three inquiry frames to obtain the values of Nitrogen (N), Phosphorous (P), and Potassium (K), as outlined in the manufacturer datasheet. The response is then parsed from the hexadecimal number using the content address to decimal base 10. A sample output for NPK readings only from Arduino IDE Serial Monitor is shown in Figure 3.69(b).

3.3.3 Power System Design

This section aims to perform an in-depth analysis of the power system design for the entire robot and suggest efficient solutions tailored to its specific needs. Considering the diverse range of electrical components our team utilizes in constructing the robot, it is crucial to supply appropriate power to each part, as different components have distinct power requirements.

The power system can be categorized into several segments according to the characteristics of the electrical components used, which include the power source, movement, control system, and sub-functions.

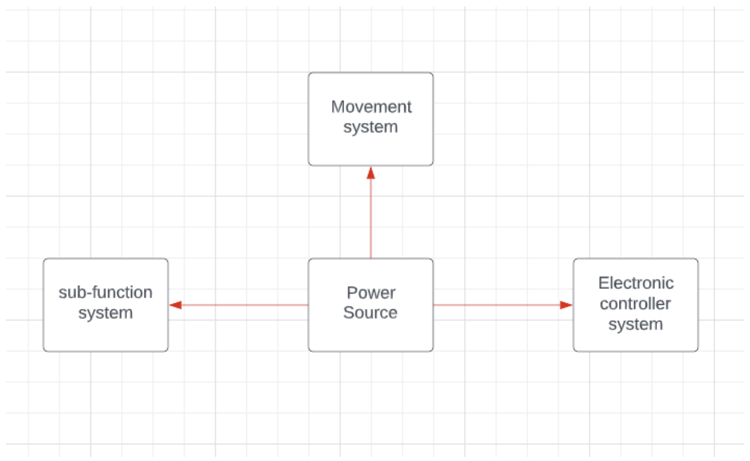


Figure 3.70: concept of rover power system

The figure above shows the basic thoughts of the whole power layout of the robot. In order to produce a reliable power circuit for the robot, the first step is to do a power requirements analysis for different kinds of electrical components. Then, choose the appropriate battery based on the analysis. Factors such as capacity, discharge rate, voltage, size, and weight must be considered to select a battery that can adequately supply the robot's power demands. After the battery selection is completed, a power distribution and management system must be built. An efficient power distribution and management system should be designed to regulate and allocate power to each component as needed. This includes voltage regulation, current limiting, and power monitoring to ensure the proper functioning of all components and prevent potential damage due to over current or voltage fluctuations.

3.3.3.1 Power requirement analysis

This part is to explain the power requirement for different kinds of electrical components. The first part is the movement part of the robot which includes four CIM 2.5 in motors and four Talon SRX motor controllers.

Figure 3.71: 2.5 in CIM 2.5 motor(AndyMark, Inc)

Figure 3.72: Talon SRX motor controller(CTR Electronics)

Based on the datasheet(CTR Electronics) of the manufacturer of the Talon motor controller, the rated voltage of the controller is 12V or 24V. Also, the operating voltage of the CIM motor is 12V which means the movement system is all operated in 12V. Based on the friction test and the datasheet of the sim motor, the current requirement for a single sim motor during skid turning is between 30A and 40A. Taking into account the combined power consumption of all four motors during skid turning, we determined that the total energy consumption range is:

$$P_{max} = 12V * 40A * 4 = 1920W$$
(Equation 3.11)

$$P_{min} = 12V * 30A * 4 = 1440W$$
(Equation 3.12)

The second analysis is about sub-function systems which include the pump and slider motors. Our team selected the IEIK Water Pressure Diaphragm Pump. According to the product datasheet, this pump requires an input voltage of 12V and has a maximum input current of 2.5A. The slider motor our team chose is a stepper motor with 5V with a current of 1.5A power consumption to ensure smooth and accurate movement. The combined power consumption of the sub-system is:

$$P_{pump} = 12V*2.5A = 30W$$
(Equation 3.13)

$$P_{smotor} = 5V*1.5A = 7.5W$$
(Equation 3.14)

$$P_{sub\text{-}total} = 30W + 7.5W = 37.5W$$
(Equation 3.15)

The last analysis is about the control system power assumption. The control system includes one Raspberry Pi 4, two teensy 4.1, two Arduino Uno R3, and one ESP-32 development board. All the control boards are charged through USB. Based on the manufacturer's user guide for Raspberry Pi 4, the max power input for Raspberry Pi 4 is 5V 3A. For the remaining controllers(Team, The Arduino), the operating current depends on the load placed on the boards. Due to the uncertainty of the maximum input current for the controller board, it is advisable to separate the power supply and data transmission for the controllers. All the microcontrollers should be powered independently.

Based on the analysis above, the power consumption of each component has been determined. Since the movement system and the sub-function system will not operate simultaneously, the maximum power system requirement can be based on the movement system's power requirement, which is 1920W.

3.3.3.2 Battery selection and Power Generation

Based on the power analysis from the previous section, the robot's battery should be capable of outputting a minimum of 1920W. Since most systems in our robot operate at 12V, it is more convenient for our team to manage power by using a 12V battery. However, employing a 12V battery in the robot implies that the battery should be able to output a minimum of 160A during skid turning. Taking into account the battery output requirements and the robot's size constraints, our team opted for two 12V 17.2Ah lead-acid batteries as the robot's power source.

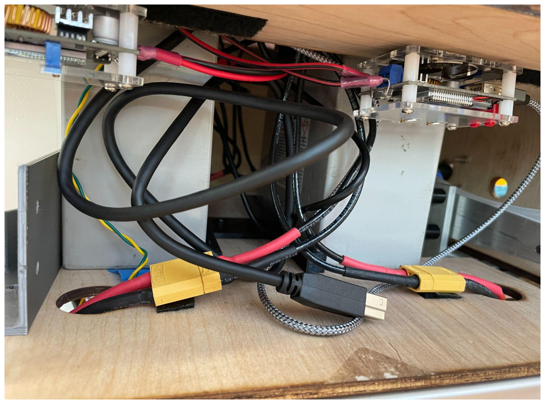


Figure 3.73(a): rover battery

The Figure 3.73(a) illustrates the batteries utilized by our team. We connected two lead-acid batteries in parallel to boost the total capacity, ensuring it met the high discharge current demand. After establishing the parallel connection, the robot's power source became 12V 34.4Ah. According to the datasheets of similar sealed lead-acid batteries, they can output a current five times larger than their capacity without causing an explosion. Given the power consumption in our system, the minimum duration for our battery is:

$$t = (34.4\text{Ah}) / (160\text{A}) * 60 = 12.9 \text{ minutes}$$

(Equation 3.16)

Due to the requirement of the project, the rover battery needs to be powered by solar power. Solar power generation needs to be designed.

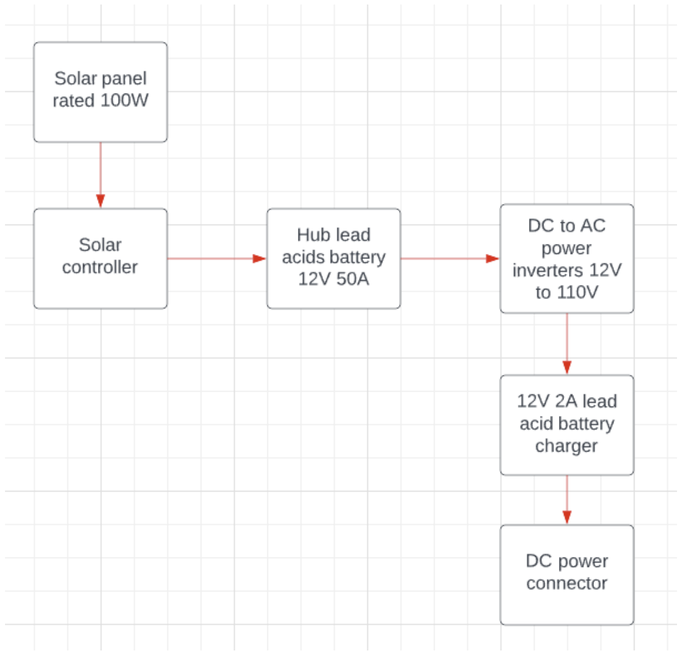

Solar panel rated 100W
Solar controller
Hub lead acids battery 12V 50A
DC to AC power inverters 12V to 110V
12V 2A lead acid battery charger
DC power connector

Figure 3.73(b): Solar power system block diagram

As Figure 3.73(b) shows, the solar panel generates power from sunlight and the solar controller limits the output into 12V. The hub battery can be charged by the solar panel. After storing the energy, the hub battery can charge the rover battery via power inverter and battery charger. The estimation minimum charging time for hub battery during full sunlight is

$$t = 50Ah/(100W/12V) = 6 \text{ hours}$$
(Equation 3.17)

The estimation minimum charging time for rover battery is:

$$t = 34.4Ah/2A = 17.2 \text{ hours}$$
(Equation 3.18)

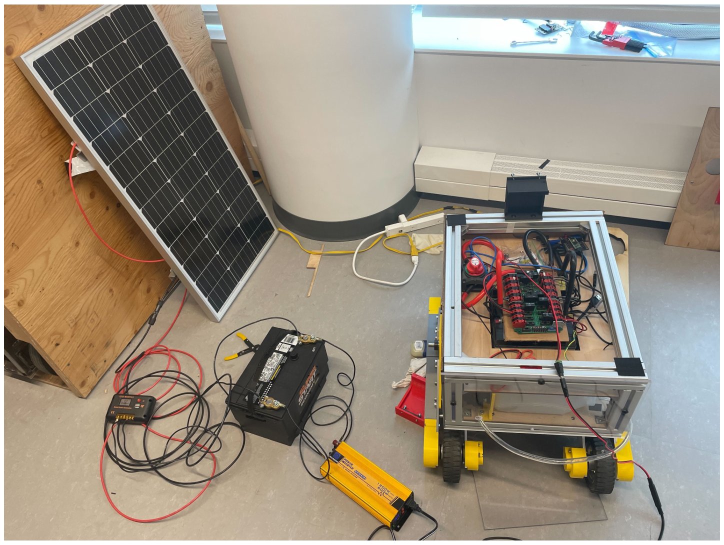


Figure 3.73(c): actual charging circuit

3.3.3.3 Wires and power distribution

The objective of this section is to discuss the power distribution within the robot. Our team incorporated a variety of components in the robot. It is necessary to use a power distribution board to manage the electrical supply efficiently.

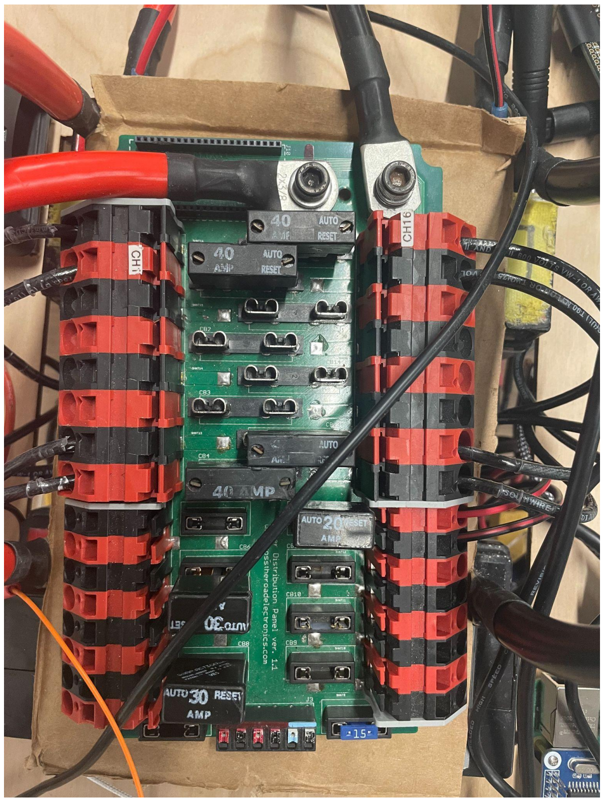
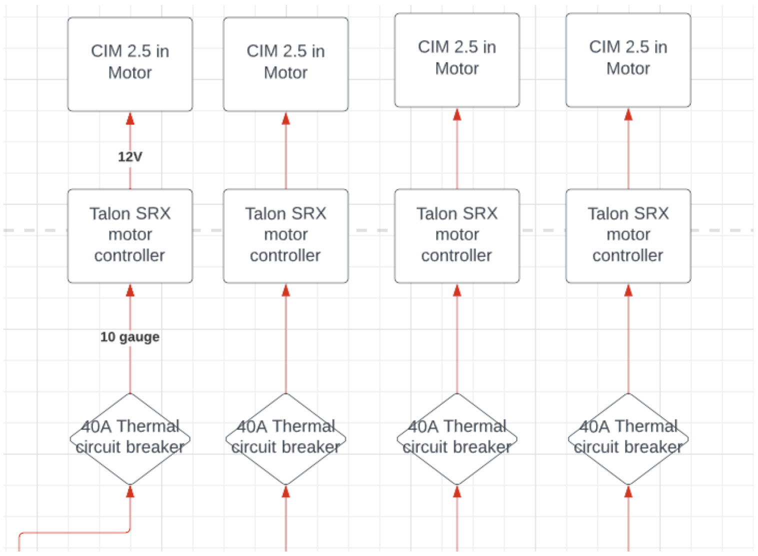


Figure 3.74: power distribution board

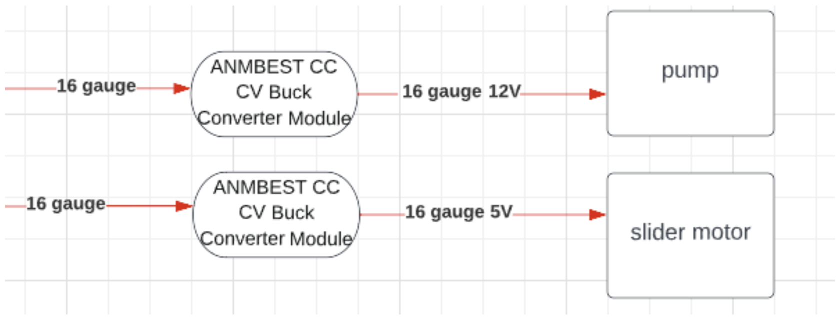
The figure 3.74 shows the power distribution board.

In the movement system, each motor requires independent power and operate in 40A. To match these requirements, our team implemented a 40A circuit breaker and utilized 10-gauge wires. A connection block diagram illustrating this setup is provided in Figure 3.75:


CIM 2.5 in Motor
CIM 2.5 in Motor
CIM 2.5 in Motor
CIM 2.5 in Motor
12V
Talon SRX motor controller
Talon SRX motor controller
Talon SRX motor controller
Talon SRX motor controller
10 gauge
40A Thermal circuit breaker
40A Thermal circuit breaker
40A Thermal circuit breaker
40A Thermal circuit breaker
Figure 3.75: movement power system

The 40A circuit breakers employed in this system are thermal circuit breakers. Their purpose is to ensure that the motor does not operate above 40A, thereby protecting the motor from potential burnout. The 10-gauge wire is the minimum wire gauge suitable for currents ranging from 30 to 40A. Utilizing 10-gauge wires helps prevent circuit overheating and mitigates power losses caused by heat generation.

For the sub-system, the circuit block diagram is illustrated in the Figure 3.76:


16 gauge
ANMBEST CC CV Buck Converter Module
16 gauge 12V
pump
16 gauge
ANMBEST CC CV Buck Converter Module
16 gauge 5V
slider motor
Figure 3.76: sub-functions power system

Our team implemented regulators to manage the input voltage for the pump and slider motors. Given that the lead-acid battery voltage is not consistently stable at 12V, it is essential to employ regulators to adjust the voltage to the appropriate operating levels for the pump and slider motor, preventing potential damage. The pump voltage is regulated to 12V, while the slider motor voltage is regulated to 5V. In this power distribution system, we used 16-gauge wires to ensure efficient and safe power transmission.

For the control system, the power distribution circuit block diagram is shown in figure 3.77:

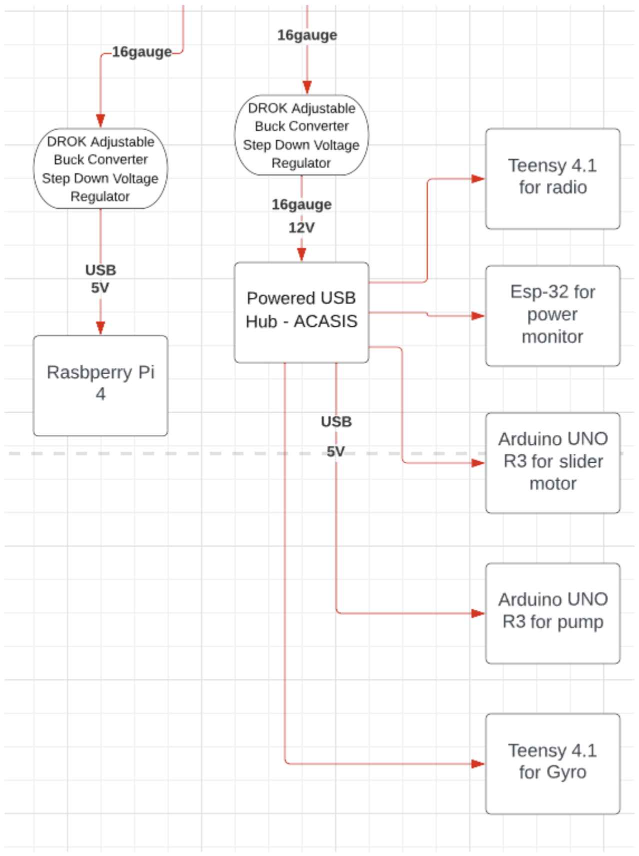


Figure 3.77: control board power system

All the controllers in the system are powered by USB. To accomplish this, two regulators and a powered USB hub are required. To power the Raspberry Pi, one regulator should adjust the voltage to 5V and utilize USB ports to deliver the power. Another regulator is set to output 12V, which is then used to power the USB hub. The powered USB hub distributes 5V power to the rest of the microcontrollers, ensuring they receive the appropriate voltage for their operation. The whole power distribution circuit is shown below:

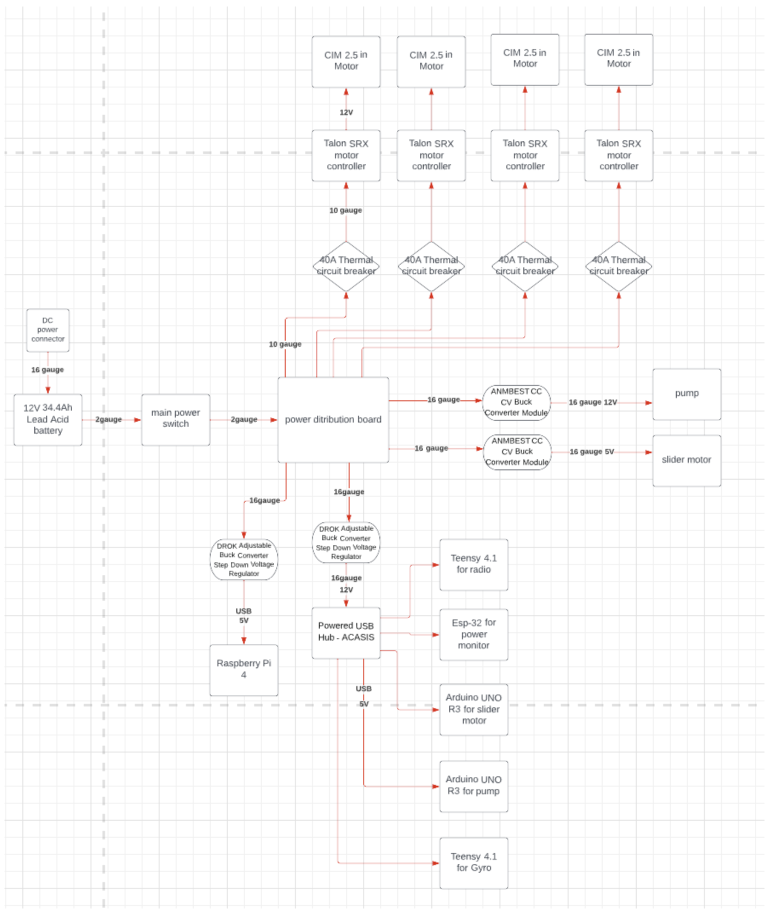

CIM 2.5 in Motor
CIM 2.5 in Motor
CIM 2.5 in Motor
CIM 2.5 in Motor
12V
Talon SRX motor controller
Talon SRX motor controller
Talon SRX motor controller
Talon SRX motor controller
10 gauge
40A Thermal circuit breaker
40A Thermal circuit breaker
40A Thermal circuit breaker
40A Thermal circuit breaker
DC power connector
10 gauge
16 gauge
12V 34.4Ah Lead Acid battery
2gauge
main power switch
2gauge
power ditribution board
16 gauge
ANMBEST CC CV Buck Converter Module
16 gauge 12V
pump
16 gauge
ANMBEST CC CV Buck Converter Module
16 gauge 5V
slider motor
16gauge
16gauge
DROK Adjustable Buck Converter Step Down Voltage Regulator
DROK Adjustable Buck Converter Step Down Voltage Regulator
16gauge
12V
USB
5V
Teensy 4.1 for radio
Powered USB Hub - ACASIS
Esp-32 for power monitor
Raspberry Pi 4
USB
5V
Arduino UNO R3 for slider motor
Arduino UNO R3 for pump
Teensy 4.1 for Gyro

Figure 3.78: total power system layout

3.3.3.4 Power monitor for lead acid battery

This section is the design of power monitoring of the rover and the hub batteries. A battery should have a power monitor to ensure efficient and sustainable usage. Power monitors give precise readings of the battery's current charge level, output voltage, and consumption rate, enabling users to make informed decisions about their energy usage. Figure 3.79 shows the block diagram of the power monitors.

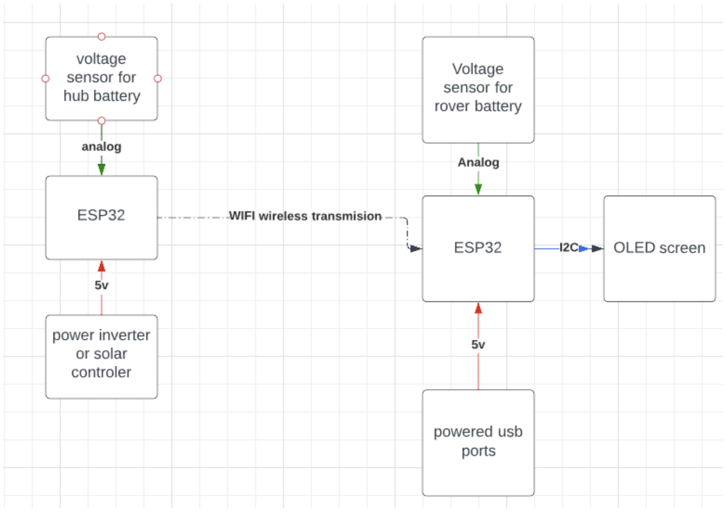

voltage sensor for hub battery
analog
ESP32
5v
power inverter or solar controler
WIFI wireless transmision
Voltage sensor for rover battery
Analog
ESP32
I2C
OLED screen
5v
powered usb ports

Figure 3.79: power monitor block diagram

The power monitors have two parts which is the rover battery monitor and the hub battery monitor. The hub monitor contains a voltage sensor and a ESP-32 development board. The rover monitor contains a voltage sensor, an OLED screen, and a ESP-32 development board same as the hub monitor. Figure 3.80(a) shows the schematics for the rover power monitor. Figure 3.80(b) is the schematics for hub battery monitor.

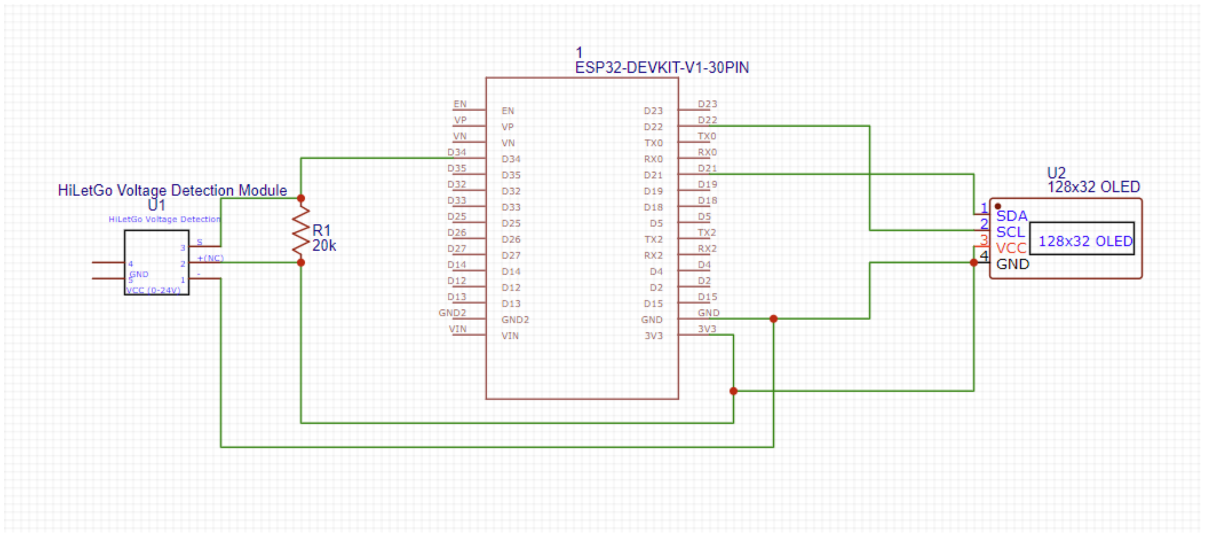


Figure 3.80(a): Schematics for rover battery monitor

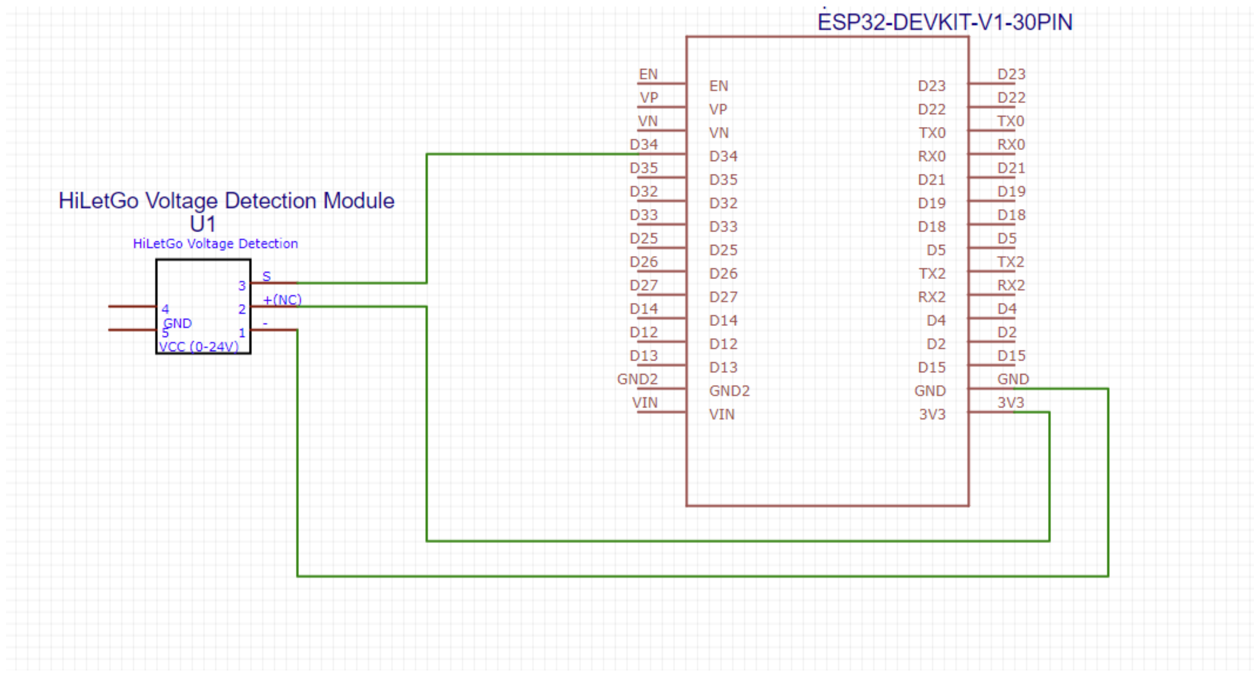


Figure 3.80(b): Schematics for hub battery monitor

For the hub monitor, as shown in Figure 3.80(b), the voltage sensor is powered by ESP32 at 3.3V. The signal that the voltage sensor passes to ESP-32 is analog signal. Analog signals are continuous signals, and they cannot be directly transmitted over digital communication systems. Therefore, before the analog data can be sent over WiFi, it must first be converted into a digital format. This is accomplished using an Analog-to-Digital Converter (ADC). The ESP32 has a built-in ADC that can be used for this purpose. The ADC pin that has been used is Pin 34.

For the rover power monitor as shown in Figure 3.80(a), the OLED screen and voltage sensor were powered by ESP-32 in 3.3V. The connection of the voltage sensor is the same as the hub power monitor. For the OLED screen, our team use I2C(Inter-Integrated Circuit) OLED screen since the power monitor does not require any data feedback from the screen. The I2C Pins on the ESP-32 are pin 21(SDA), and Pin 22(SCL).

For the communication between two monitor modules, they need to be connected to the same WiFi network. The WiFi module in the ESP32 allows it to connect to a WiFi network. Once connected, it can send and receive data over the network. Hub module creates its own wireless network over 2.4 Ghz WiFi. The Rover module connects to the access point. The data transmission between the two ESP32s happens over a TCP (Transmission Control Protocol) protocol. This protocol is responsible for sending the data from the hub ESP32 to the rover ESP32. The data transmitted over the network is broken down into packets. Each packet contains a header and a payload. The header includes information like the source and destination IP addresses, while the payload contains the actual data being transmitted. The Rover module sends a request message to the hub module and the hub module sends the request message (specified by header) back to the client over 12 bits. Rover module decodes the message using objective C and stores the string data in a volatile variable. Finally, the rover outputs the following float number into I2C to OLED screen.

In Figure 3.81(a), a sample test is run based on 2 different batteries. The rover ESP32 is shown to be able to display the battery voltage as well as the battery percentage successfully. A hashset is used to map the voltage to lead acid battery percentage using a commonly reference table shown in Figure 3.82. This test also provides a time delay between voltage feedback time at around 3 seconds. This is the delay between the WiFi communication and data processing between the base and the rover ESP32.

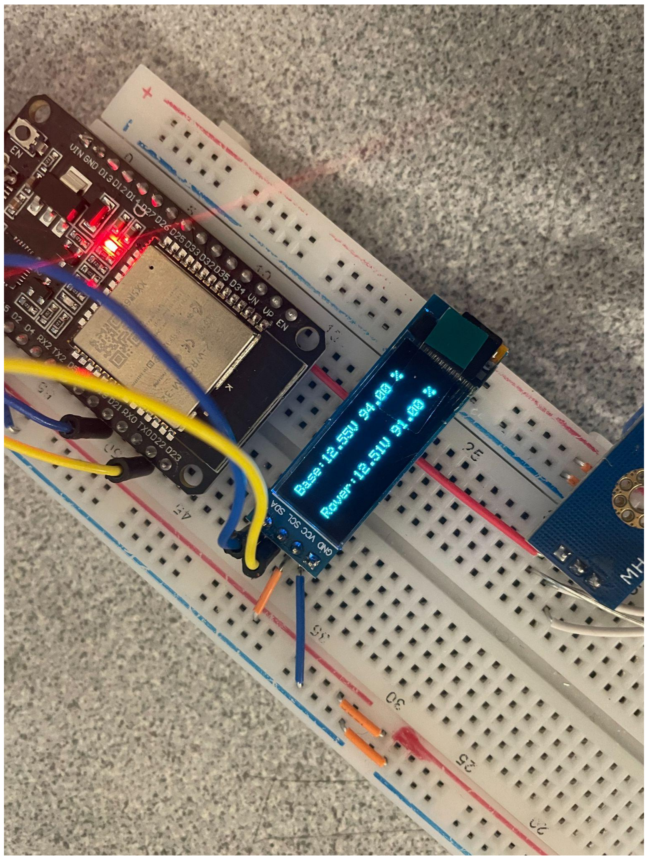


Figure 3.81(a): Implementation of the Battery Level Monitor (hub to rover over WIFI protocol)

The final version of the voltage monitor is shown in Figure 3.81(b). In comparison to the test version of the monitors, the final version has better wire management and they were soldered into the PCB board which can avoid exposing wires.

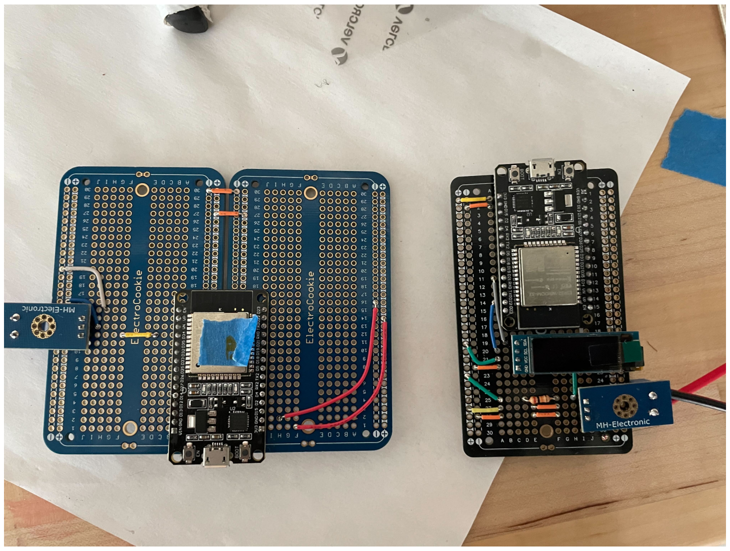
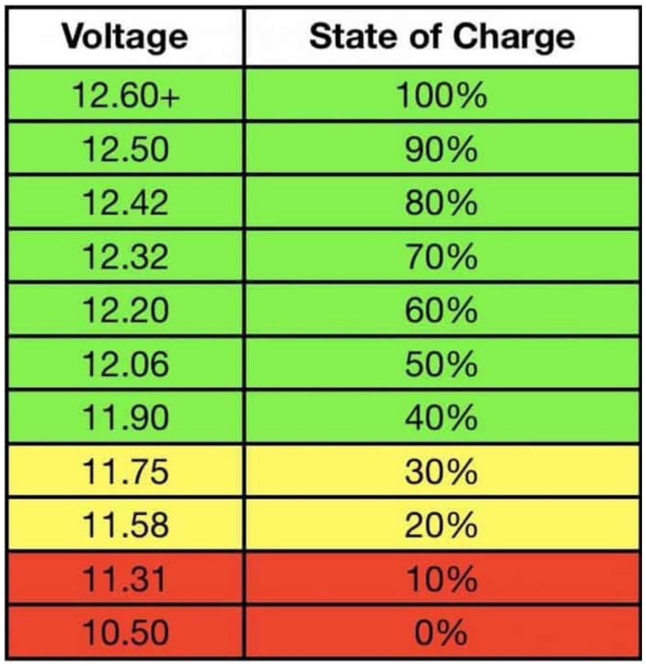


Figure 3.81(b): hub monitor(left), rover monitor(right)



Voltage	State of Charge
12.60+	100%
12.50	90%
12.42	80%
12.32	70%
12.20	60%
12.06	50%
11.90	40%
11.75	30%
11.58	20%
11.31	10%
10.50	0%

Figure 3.82: Voltage versus Lead Acid Battery Level (nemo4all)

3.3.4 Motor Control

The goal of the movement system is to allow the robot to drive and turn. It contains the 2.5in CIM Motors, SRX Mag Encoders, and Talon SRX Motor Controllers. The 2.5in CIM motors have better performance than the motors that the last team used. The raspberry pi can communicate with each of the Talon SRX Motor Controllers using CAN bus. This allows for the minimization of wires that need to be run throughout the system. Since our team decided to use Taylon SRX as our drive controller, SRX

Magnetic Encoder was the best choice for encoder. This kind of encoder is Supported by Talon SRX firmware and could be connected directly to Talon SRX without the need for custom cables.

The motor control section of the robot has a singular goal: enable drive functionality on the robot. Its primary function is to enable the robot to move forward and backward, as well as execute a skid turn. This will allow us to achieve our goal of precise and efficient robot mobility. This section of the robot is composed of the CIM Motors, Talon SRXs, SRX Mag Encoders, and the Power Distribution Board.

3.3.4.1 Component Layout

The robot is designed with a tank drive system. Both of the left motors and both of the right motors are connected together with a chain in order to increase the torque of the side, in the case that one of the wheels begins to slip.

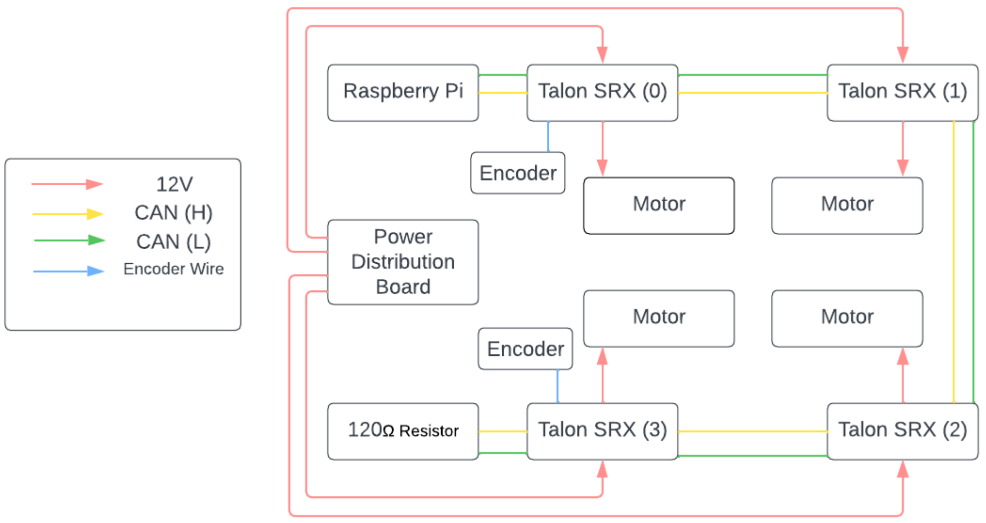


Figure 3.83: Motor Control - Block Diagram of Components

The full layout of each component can be seen above in Figure 3.83. Each motor has its own Talon SRX to control the amount of energy the motor receives. Each of these motor controllers are wired to the 12V power supply, which is regulated for each motor with a 40A fuse. Each side of the robot's drive system is monitored with a singular SRX Mag Encoder. This allows for PID tuning of each side, ultimately allowing precise movement of the robot. Each of the TalonSRX's are connected to each other using CAN bus. The CAN bus is also connected to our Raspberry Pi, which will send out commands to each motor using their own unique id's. The CAN line is terminated after the last motor controller using a 120Ω resistor.

3.3.4.2 Code Structure

The TalonSRX Library is Provided by CTR Electronics, and is called Phoenix V5. This library is written in C++, and Java (Only recommended for First Robotics Teams), causing a compatibility problem with our robot's main controller which is written in Python. The solutions for this problem were to either rewrite the already written robot code in C++, create a wrapper for the library to make it accessible in Python, or to create a separate motor control program that would communicate to our robot's main controller using a PIPE.

Solution	Pros	Cons
Rewrite Robot Code in C++	● Lowest Latency ● Lowest CPU insensitive	● Extremely Time Consuming
Create a Wrapper for the Library	● Low Latency ● Low CPU insensitive	● High Difficulty ● Time Consuming
Write a Separate Motor Control Program	● Quick to implement ● Least prone to unforeseen problems	● Highest latency ● More CPU Intensive

Table 3.3: Motor Control Library Solution - Pros & Cons

To choose which option would be best a pros and cons list was created (Table 3.3). This pros and cons list helped come to the decision of writing a separate motor control program that will communicate over a PIPE. This was done since it was the least error prone solution and could be the quickest to implement. Although this has the highest latency between each option, it should still be small enough to not cause any issues.

To start the motor control program (MCP), the main robot program (MRP) first grants itself permission to access the buid.sh and run.sh files using chmod. Once permissions are granted, the MRP will then rebuild the MCP because it requires building on the system it is running on due to library binaries. Additionally this allows easier testing of new code when pulled to the Raspberry Pi. Once the program is built the program will then call the run.sh file and wait ten seconds to ensure it starts up properly.

To avoid possible collisions, the communication between the MCP and the MRP will be initiated from the MRP. The MRP will send a message to the MCP, and then the MCP will read, parse, and process the command. The MCP will then respond with an acknowledgement to inform the MRP that the command has been processed. This acknowledgement also ensures the program is still functioning. There are some cases where the Phoenix library will print a message, causing unexpected messages to arrive in the PIPE. To unclog the pipe of these messages, the MRP will read through each response in the PIPE until a message that starts with “[agribot-mc]” is discovered.

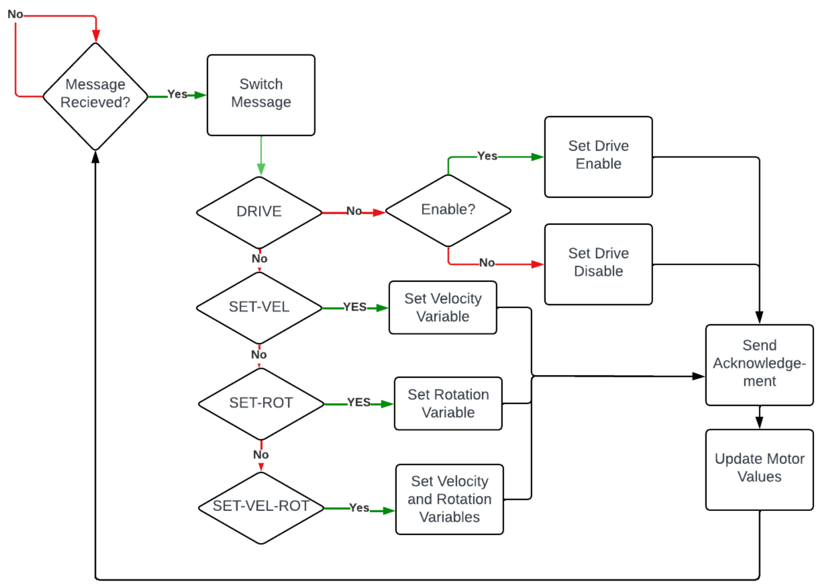


Figure 3.84: Motor Control Program - Code Flow

The MCP flow of code can be seen above in Figure 3.84. The MCP will lay dormant until a message from the MRP is sent. Once a message is received, it will be run through a switch statement that will determine what the message is. It could be one of

- DRIVE, which will either enable the drive system or disable it,
- SET-VEL, which will set the velocity,
- SET-ROT, which will set the rotation velocity, or
- SET-VEL-ROT, which will set the velocity and rotational velocity.

Once these messages are parsed, they will update their respective variables. Then the program will send an acknowledgement to the MRP. Finally the velocity values of each motor will be updated.

3.3.4.3 PID Tuning

The CTRE closed loop PID controller, integrated within their software, offers a straightforward and effective means for achieving accurate wheel control. Utilizing the built in controller instead of implementing our own, increases the speed at which we can implement this and also the simplicity of the code . The SRX Mag Encoders deliver 12-bit precision, equating to 4096 values per rotation, enabling highly precise control of wheel movements. To fully capitalize on this degree of accuracy, proper PID tuning is crucial.

Adjusting the Feed-Forward (F), Proportional (P), Integral (I), and Derivative (D) gains is the core of the PID tuning process, with the aim of attaining optimal control. To create an accurate tuning experience a controlled environment was set up to run a trial by error evaluation to determine the appropriate values. To streamline the process of adjusting each value, the robot controller was modified in order to include four sliders that remotely updated the F, P. I, and D values in the closed loop FPID controller. The system was able to reliably move specified distances having a Feed-Forward value of .06, a Proportional value of .69, an Integral value of .0023, and a Derivative value of .0665.

3.3.5 Radio Transmitter

The Radio Transmitter is crucial for the completion of our goal of having a robot that is able to communicate its collected soil data all the way to the web server. The radio transmitter serves as the bridge between the hub and robot information transfer.

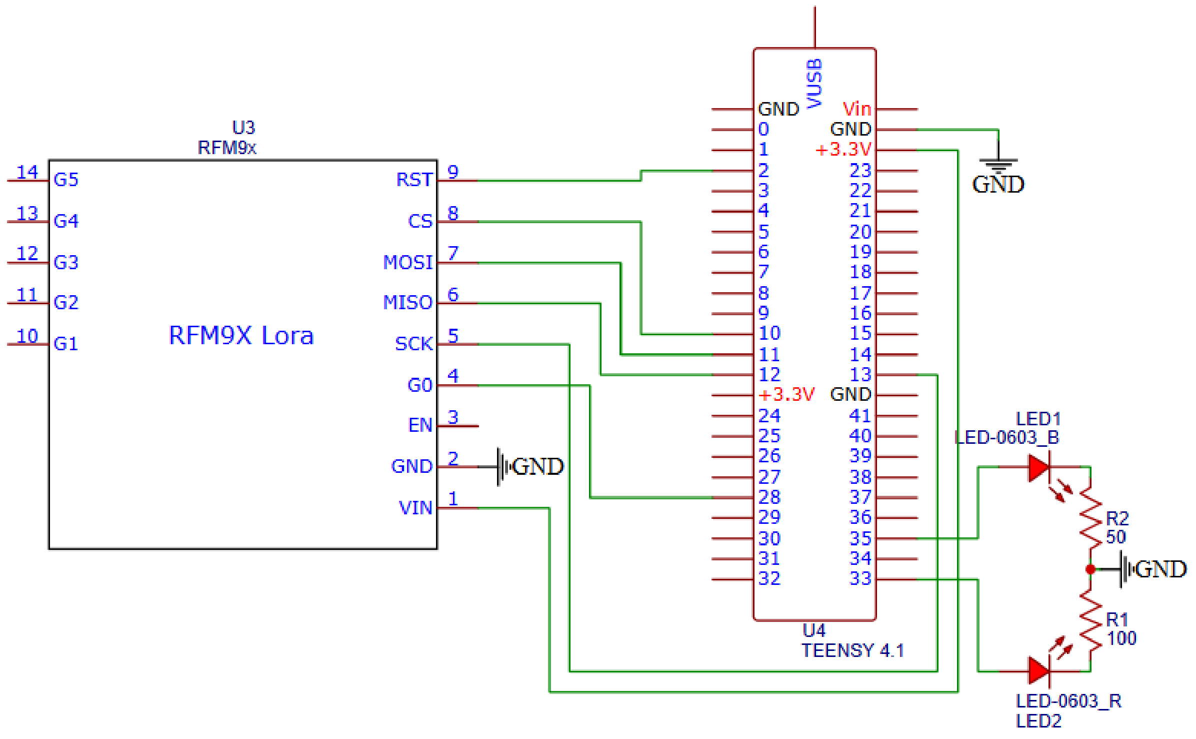


Figure 3.85: Radio Transmitter - Final Circuit Design

The final design of the radio transmitter (Figure 3.85) changed marginally from the original prototype. The main differences occurred in the pins used on the controller. This change allows for an easier installation onto the radio's permanent home, the pera-board. Most pins were selected on the left side of the schematic to minimize wire routing. The SCK pin had to be placed on the right side of the board, since the only SPI Clock Pin was only located on the opposite side of the board. The LED pins were also moved to minimize the amount of wire needed to route on the perma-board.

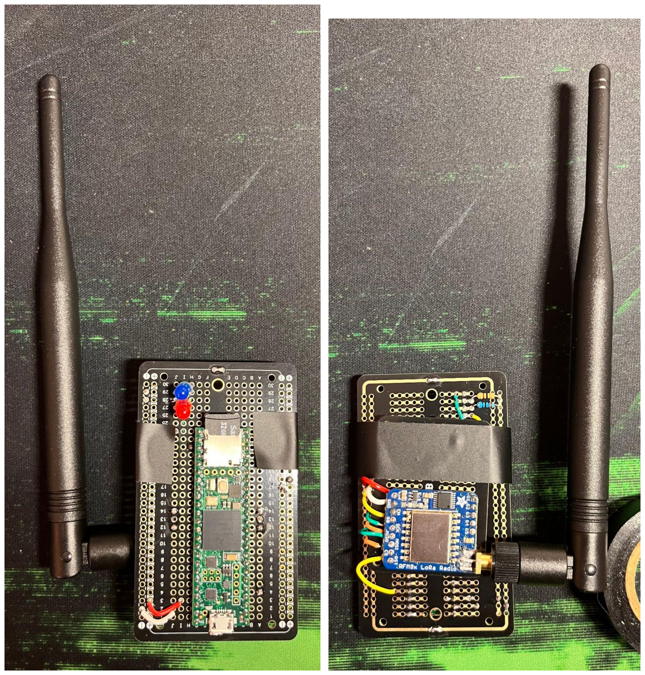


Figure 3.86: Radio Transmitter - Pera-board

After the final redesign, a quick test of the new pin selection was done to verify the changes. Once verified, the components were wired onto the perma-board (Figure 3.86). The perma-boards found were all slightly too small to fit all of our components, and going up in size would result in a perma-board that was much larger than necessary. The fix for this was to place the radio transceiver on its own tiny permaboard and then attach that to the back of the main perma-board. This ultimately led to a sleeker design, but did increase the difficulty of assembly.

3.3.6 Data Management

All of the data for the project is stored in a MongoDB Atlas database. This provides a database setup that is both simple and scalable. Inside the Atlas Database, there is a database for authentication and a database for each robot. When choosing the data storage method a big consideration was scalability, and simplicity. Since MongoDB Atlas was already well known and provides the scalability needed, it was selected for our data storage system.

3.3.6.1 Authentication Database

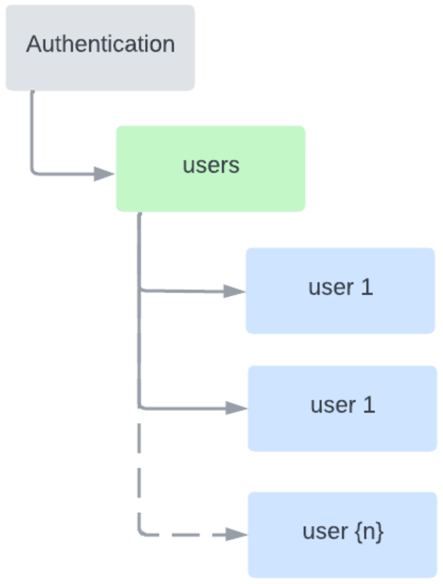
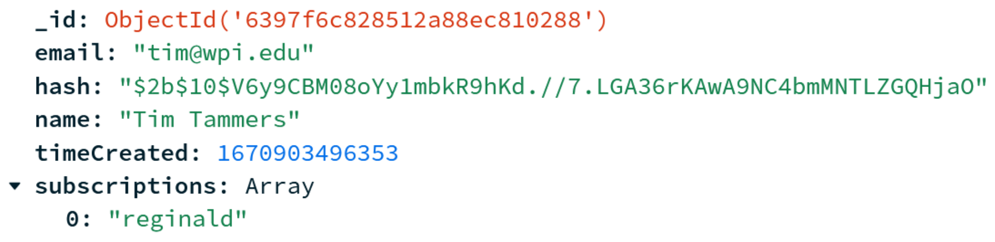


Figure 3.87: Authentication Database - Scheme

The Authentication database consists of a single users collection (Figure 3.87). This collection stores all of the data for each individual user of the website. This method is extremely scalable and good for rapidly changing data due MongoDB's unstructured data storage system. Having the ability to handle changing data is extremely convenient especially during solo development of this system, where some original data points were overlooked for importance.



```
  _id: ObjectId('6397f6c828512a88ec810288')
  email: "tim@wpi.edu"
  hash: "$2b$10$V6y9CBM08oYy1mbkR9hKd.//7.LGA36rKAwA9NC4bmMNTLZGQHjaO"
  name: "Tim Tammers"
  timeCreated: 1670903496353
▾ subscriptions: Array
    0: "reginald"
```

Figure 3.88: Authentication Database - Example User Document

A document is added every time a user is created (Figure 3.88). This document consists of a unique id called _id. Additionally it stores the email of the user and the hash of the user's password. A hash of the user's password was stored instead of the raw password to minimize the severity of a data leak if it were to occur.. The timeCreated of the account is stored to manage potential data issues between old user accounts. The user also stores a list of subscriptions. This list stores the list of robot names that the user has access to the data of.

3.3.6.2 Robot Database

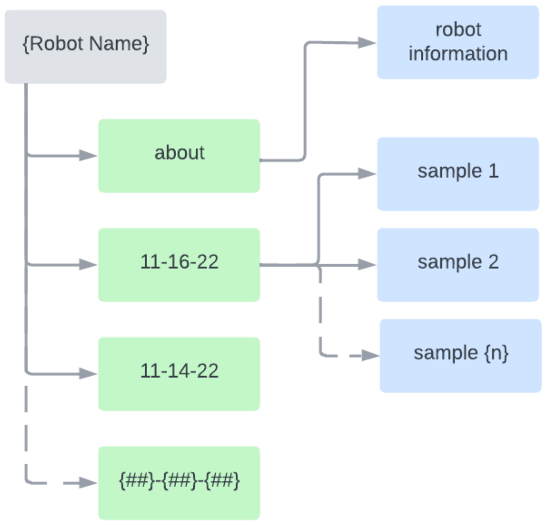
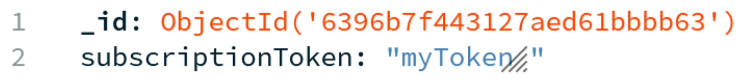


Figure 3.89: Robot Database - Scheme

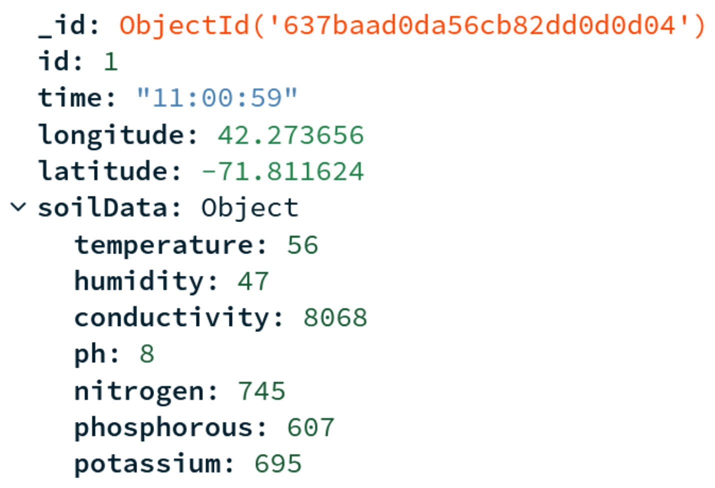
The Robot Database is a unique database titled the name of the robot (Figure 3.89). There will be an individual database for each robot. Inside this database consists a collection called about and a collection for each date that a survey of the farm has taken place. This naming scheme consists of MM-DD-YY.



```
_id: ObjectId('6396b7f443127aed61bbbb63')
subscriptionToken: "myToken"
```

Figure 3.90: Robot Database - Example About Document

Inside a robot's “about” collection stores a single document (Figure 3.90). This document consists of a unique id called _id. Additionally it contains a subscriptionToken. This value is used for when a farmer wants to subscribe to a new robot. This token is used as a password for the robot’s information, given the robot name and this token the user can view all of this robot’s soil-sample data.



```
  _id: ObjectId('637baad0da56cb82dd0d0d04')
  id: 1
  time: "11:00:59"
  longitude: 42.273656
  latitude: -71.811624
⌄ soilData: Object
    temperature: 56
    humidity: 47
    conductivity: 8068
    ph: 8
    nitrogen: 745
    phosphorous: 607
    potassium: 695
```

Figure 3.91: Robot Database - Example Soil-Sample Document

Inside of a survey date document contains many documents, one document for each sample taken on the farm (Figure 3.91). This consists of an _id a unique id and an id which is the index of which the sample was taken. Additionally the longitude and latitude of the sample is stored to know where the sample was taken. There also consists a soilData object which holds all of the information of the soil condition at the sample location. This object olds the soils measured temperature, humidity, conductivity, ph, nitrogen, phosphorus, and potassium level.

3.3.7 Web Server

The web server is responsible for serving two main purposes: providing endpoints for the website and endpoints for a robot to access and manipulate data stored in the database. To ensure that neither the robot nor the website have direct manipulation access to the database, the web server acts as an intermediary layer that controls and restricts access privileges for each user and robot.

3.3.7.1 NodeJS

To build the web server, the NodeJS runtime environment was chosen due to its popularity and versatility for building server-side applications. Additionally it allows for the use of the Node Package Manager which is a big database of libraries that can easily be added and constantly checked for updates. To further simplify and accelerate the development process, several popular frameworks and libraries were utilized.

3.3.7.2 Frameworks and Libraries

Express, a widely used web framework for NodeJS, was used to provide a streamlined and flexible way to define and manage endpoints for the web server. Passport, another important library for web server development, provided middleware and strategies for handling user authentication. Passport-JWT, a strategy for Passport, was used to enable authentication using JSON Web Tokens, which is a secure and scalable method of authentication. Finally, Swagger, a tool used to define and document APIs, was used to ensure consistency and compatibility across different clients and applications by clearly defining the endpoints and their expected input and output.

3.3.7.3 Endpoints

The endpoints can be grouped into 5 distinct categories: index, robot, samples, account, and user. The index route manages page file sharing, the robot route manages any robot request, the samples route manages retrieving soil data from the database, the account route manages user accounts, and the user route manages manipulation that needs to be done to user data.

The index route solely provides the page files for each destination the user can access. This includes:

- GET “/”
- GET “/dashboard”
- GET “/about”
- GET “/statistics”
- GET “/login”

Each of these provide the same page file, since the react app is capable of handling page routing locally, saving the amount of calls needed by the server.

The robot route has one functional endpoint to be used currently. This endpoint is the “/robot/add-soil-data/”, and is a POST. This endpoint takes a json object defined early in the Data Management Section, and sends it to the database, using the robot name and the date it was taken to determine the appropriate collection to store it in.

The samples route provides access to the data collected by a certain robot, and contains two endpoints. These are:

- GET “/samples/:robotName/”
- GET “/samples/:robotName/:date”

The “/samples/:robotName” endpoint is GET and is responsible for returning the dates of each survey date completed by the robot. This is done by listing all the collections for the robot and filtering out the “about” collection which is responsible for storing robot information. Then this list is packaged in a json object containing the robotName and the datesList.

The "/samples/:robotName/:date" endpoint is a GET and is responsible for returning all of the samples which have occurred upon a certain date. This is done by searching for the collection which corresponds to the date entered in the request. The list of soil samples are then return in a JSON object containing the array of soil information.

The account routes provide everything needed for the user to use an account on the website. The route contains the endpoints:

- POST "/login/"
- POST "/register/"
- POST "/logout/"

The "/login/" endpoint is a POST and takes in a JSON request body that contains an email and password. The user is then searched for by the email address and hash of the password. If these are successfully found the server will return a 200 response with an HTTP only authentication cookie which will be attached to every request sent to the web server until the user logs out. If the request fails a 400 request will be sent to the user so they can be informed that the credentials they provided were invalid.

The "/register/" endpoints is a POST and takes in a JSON request body that contains an email, password, rePassword, name, and dob.The password and repassword are first checked to ensure they match. Then the email is then run though the user database to ensure that an account does not exist, if it does a 500 response is sent back. If there is no associated account with that email already then the new user account will be added to the database, replacing the password with a hash of the password.

The "/logout/" endpoint is a POST and takes no request body. The endpoint will clear the "jwt" cookie stored in the user's browser which will log them out of the website.

The user route provides endpoints for the manululation of user data. This is so far implemented for user subscriptions to robots. This feature allows for a user's permission to be checked by robot data search and for loading of the initial page. These endpoints are:

- GET "/user/subscriptions/"
- POST "/user/subscribe/"

The "/user/subscriptions" is a GET and is responsible for returning the list of robots that a user is subscribed to.

The "/user/subscribe" endpoint is a POST and is responsible for adding a robot to the list of subscriptions the user has. To protect access to each robot's data the request must contain the subscription token for the robot. The request body contains the user's email, robotName, and subscriptionToken. The robot name is first verified that it exists and then the subscription token is verified. If there is a match the robot is added to the list of user subscriptions

3.3.7.4 Endpoint Protection

To protect sensitive information, route protection using JSON Web Tokens was implemented. This was implemented by adding in a middleware in express. Middleware can be represented as a stack in which a request needs to follow before a response is sent back. The authentication middleware verifies the request sender has the appropriate JSON Web Token before allowing their request to proceed. This was done on the user routes, samples routes, and the robot route.

3.3.7.5 Hub Server

The Hub Server is responsible for scheduling the robot and relaying soil information sent over radio to the web server. This system aids in achieving the goal of being able to provide a website where soil information can be visualized. This system was created using the Node.js runtime since it allowed for code consistency between the hub and the web server. Additionally since it provides a simple runtime for asynchronous event-driven code. Requests were sent using the axios library since it is an industry standard library that has worked well in the past.

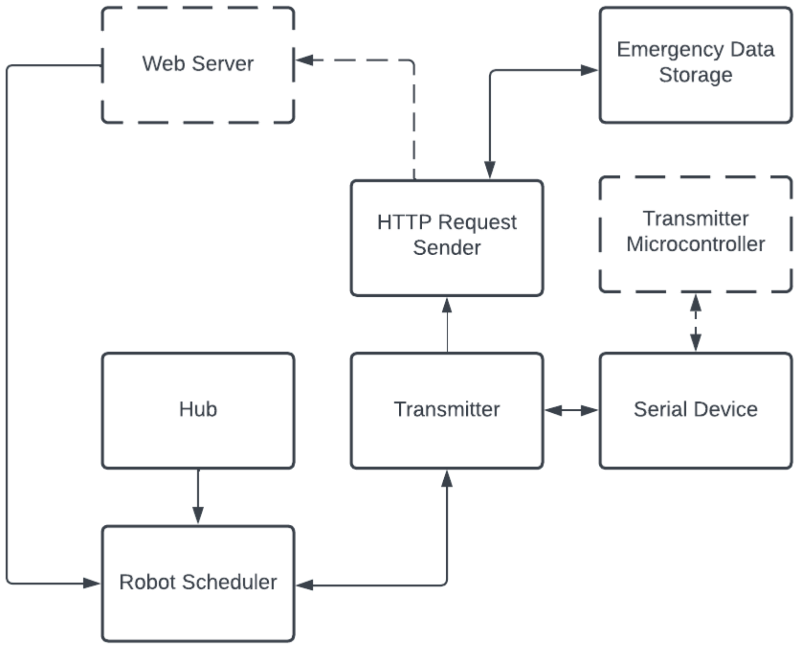
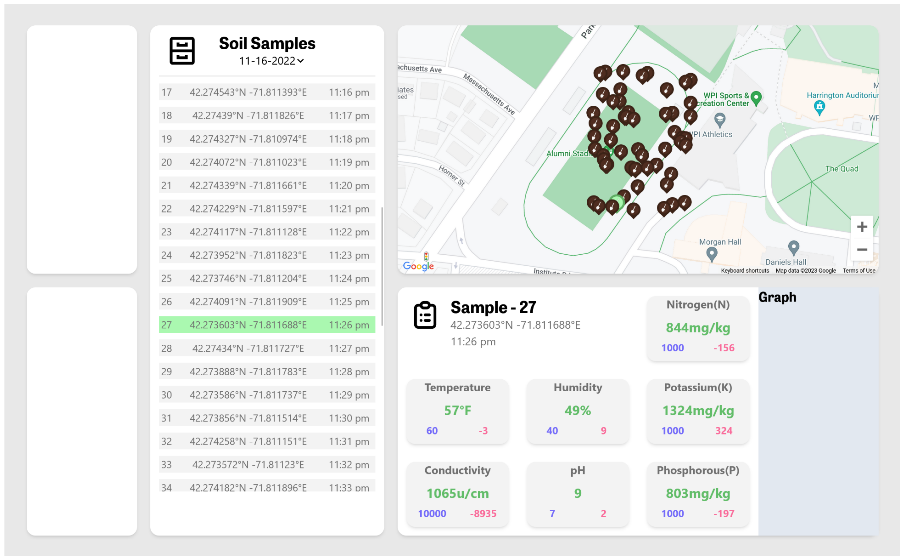


Figure 3.92: Hub Server - Code Flow

The hub server code flow can be seen above in Figure 3.92. The program starts in the Hub section, where the code is configured. The Robot Scheduler is where the code waits for events to trigger from either the transmitter or the web server. If the robot sends information to the hub such as a soil sample it will get sent to the HTTP Request Sender where it will be sent to the web server. If the robot sends data such as action information it will be sent to the robot scheduler. In the scenario where the hub cannot send a successful message to the web server, the messages will be backed up in an emergency data storage system. This system consists of a json file of the missed samples. These samples will be synced to the web server once it is back online.

3.3.8 Farmer Website



Soil Samples
11-16-2022

17	42.274543°N -71.811393°E	11:16 pm
18	42.27439°N -71.811826°E	11:17 pm
19	42.274327°N -71.810974°E	11:18 pm
20	42.274072°N -71.811023°E	11:19 pm
21	42.274339°N -71.811661°E	11:20 pm
22	42.274229°N -71.811597°E	11:21 pm
23	42.274117°N -71.811128°E	11:22 pm
24	42.273952°N -71.811823°E	11:23 pm
25	42.273746°N -71.811204°E	11:24 pm
26	42.274091°N -71.811909°E	11:25 pm
27	42.273603°N -71.811688°E	11:26 pm
28	42.27434°N -71.811727°E	11:27 pm
29	42.273888°N -71.811783°E	11:28 pm
30	42.273586°N -71.811737°E	11:29 pm
31	42.273856°N -71.811514°E	11:30 pm
32	42.274258°N -71.811151°E	11:31 pm
33	42.273572°N -71.81123°E	11:32 pm
34	42.274182°N -71.811896°E	11:33 pm

Sample - 27
42.273603°N -71.811688°E
11:26 pm

Nitrogen(N)	844mg/kg	1000	-156
Temperature	57°F	60	-3
Humidity	49%	40	9
Potassium(K)	1324mg/kg	1000	324
Conductivity	1065u/cm	10000	-8935
pH	9	7	2
Phosphorous(P)	803mg/kg	1000	-197

Figure 3.93: Farmer Website Implementation

The farmer website (Figure 3.93) allows for simple access and viewing of the collected data from the robot. This is crucial for completing our goal of providing a simple and easy access to the data collected. For this dashboard to be successful it must provide access to all samples collected across multiple dates, display this data and also show where on the farm the sample was collected. The Dashboard was made using a tech stack of React, TailwindsCSS, Webpack, and GoogleMapsAPI. The combinations of these libraries allows for quick and easy UI state management, styling, and packaging of code.

React was chosen for the development of this dashboard since it allows for easy state management between components. This was extremely important since the data between the soil samples list, map and sample information are all linked. The selected soil sample will illuminate green on the soil sample list, and on the soil sample map, as well as update the contents of the sample information. Additionally since samples are selectable from both the map and the soil sample list it provides a simple way to update the current selected sample.

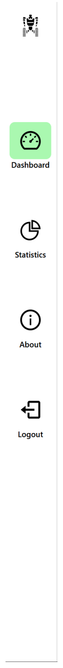


Figure 3.94: Farmer Website Implementation - Navigation Bar

The state of the current page is controlled by the navigation bar on the right side of the page (Figure 3.94). To provide feedback to the user on what page they are currently on, the appropriate icon will illuminate a bright green hue. An icon can be selected to change the current page of the user; currently no other pages are implemented. A logout button is also located at the bottom of this bar and will return the user to the login page when clicked.

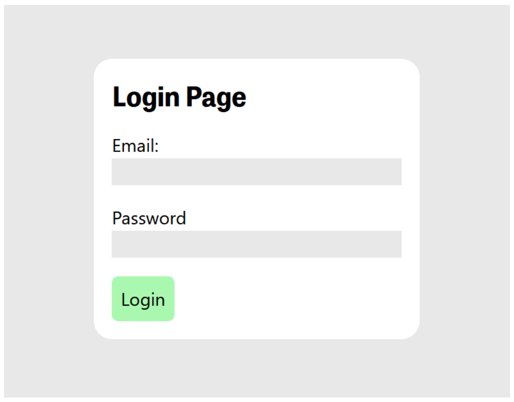
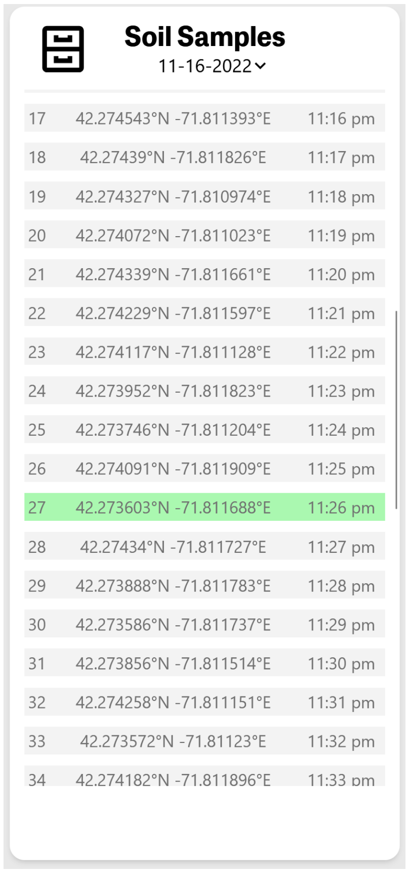


Figure 3.95: Farmer Website Implementation - Login Page

When the user tries to access the dashboard and is not logged in they will be redirected to the login page (Figure 3.95). This page is responsible for providing a prompt for the user to input their credentials. When the user submits valid credentials, the user will be granted access to the dashboard and be automatically redirected. This login state will be held using a JWT authentication token stored in the user's browser as an HTTP cookie.



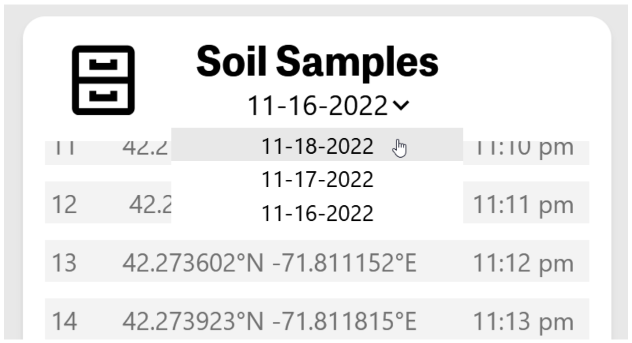
Soil Samples

11-16-2022

17	42.274543°N -71.811393°E	11:16 pm
18	42.27439°N -71.811826°E	11:17 pm
19	42.274327°N -71.810974°E	11:18 pm
20	42.274072°N -71.811023°E	11:19 pm
21	42.274339°N -71.811661°E	11:20 pm
22	42.274229°N -71.811597°E	11:21 pm
23	42.274117°N -71.811128°E	11:22 pm
24	42.273952°N -71.811823°E	11:23 pm
25	42.273746°N -71.811204°E	11:24 pm
26	42.274091°N -71.811909°E	11:25 pm
27	42.273603°N -71.811688°E	11:26 pm
28	42.27434°N -71.811727°E	11:27 pm
29	42.273888°N -71.811783°E	11:28 pm
30	42.273586°N -71.811737°E	11:29 pm
31	42.273856°N -71.811514°E	11:30 pm
32	42.274258°N -71.811151°E	11:31 pm
33	42.273572°N -71.81123°E	11:32 pm
34	42.274182°N -71.811896°E	11:33 pm

Figure 3.96: Farmer Website Implementation - Soil Sample List

On the dashboard the user will see a list of soil samples with their id, location, and time (Figure 3.96). To select a sample a user can click upon any of the samples listed. The selected sample will illuminate a green hue, so show the user which sample is selected. To move up and down the list the user can use their scroll wheel to move across the samples. When hovering over different samples, the sample will illuminate a darker shade of gray so the user can quickly see which sample their mouse is over.



Soil Samples

11-16-2022

11	42.2	11:10 pm
12	42.2	11:11 pm
13	42.273602°N -71.811152°E	11:12 pm
14	42.273923°N -71.811815°E	11:13 pm

Figure 3.97: Farmer Website Implementation - Soil Sample List Date Selection

To view samples from a different point in time, the user can click on the drop down array on the right side of the date (Figure 3.97). This will show a list of dates to select from. When the user hovers over these dates they will illuminate a gray color to show visual feedback to which date is being selected by the mouse. When a user clicks on a date, the menu will automatically close and the samples inside the list will update to the new samples. Additionally, if the user clicks off of the menu, the menu will automatically close, keeping the original survey date.

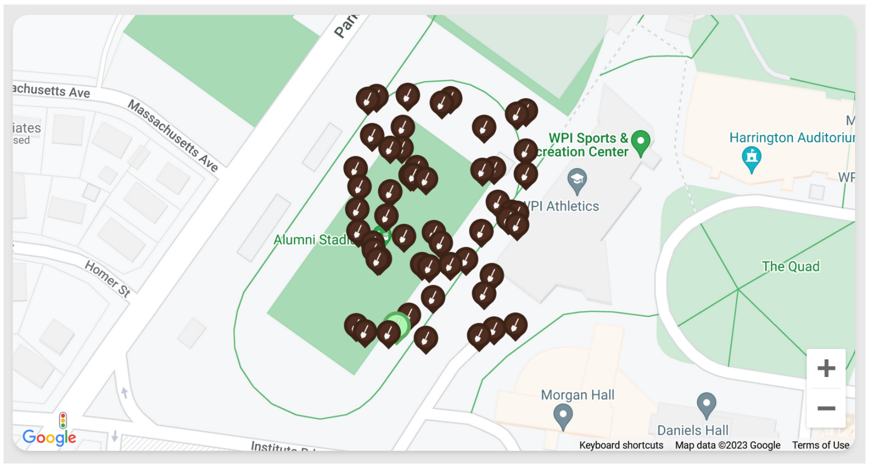
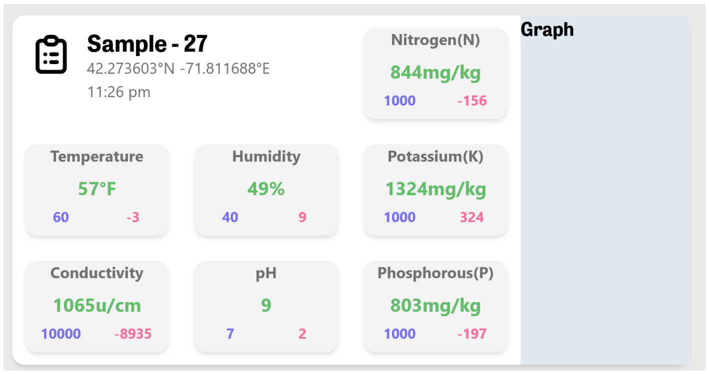


Figure 3.98: Farmer Website Implementation - Soil Sample Map

To view the locations of each sample on the farm, an interactive map of the samples was implemented using GoogleMapsAPI (Figure 3.98). This allowed for the placing of pins on a map given a longitude and latitude. Additionally there was a React module which made the implementation of the maps api with react extremely straightforward. The GoogleMapsAPI is integrated using a cloud service by google which does have a maximum capacity on requests per day, but this capacity is far below the expected requirements of this prototype which is why it was selected.



Sample - 27
42.273603°N -71.811688°E
11:26 pm

Graph

		Nitrogen(N) 844mg/kg 1000 -156
Temperature 57°F 60 -3	Humidity 49% 40 9	Potassium(K) 1324mg/kg 1000 324
Conductivity 1065u/cm 10000 -8935	pH 9 7 2	Phosphorous(P) 803mg/kg 1000 -197

Figure 3.99: Farmer Website Implementation - Soil Sample Data

In order to analyze the conditions of the soil, a sample information component was added to the bottom right of the dashboard (Figure 3.99). This component contains seven individual condition tracks of the soil. Additionally it shows the location and time of the sample collection. There is room for the implementation of a graph on this component to visualize the soil NPK levels, but was dismissed due to its minimal importance for this prototype.

3.3.9 Robot Controller Interface

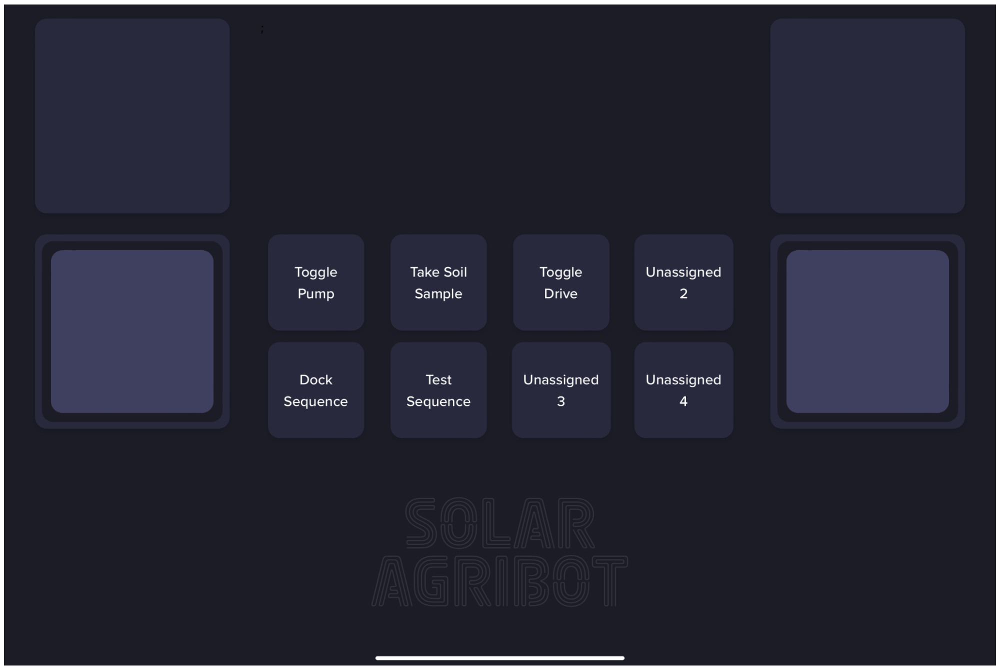


Figure 3.100: Robot Controller Implementation

The robot controller interface (Figure 3.100) is to allow for wireless control of the robot. This interface will be crucial for drive testing, subsystem testing, and routine testing of the robot. For the robot controller interface to be successful, it must be able to control the robot motors, activate and deactivate routes and subsystems, provide feedback to the user, and have low latency communications with the robot. The interface was made using a tech stack of React, TailwindsCSS, and Parcel. The combinations of these libraries allows for quick and easy UI state management, styling, and packaging of code.

There are two separate ways in which the user can interact with the robot. The first way is with the eight buttons placed in the center of the screen. The second way is with each of the two joysticks on the right and left side of the page. The placements of the joysticks allow for comfortable holding of a tablet while driving the robot, while the button placement and naming provided better understanding of each button task when compared to a typical gaming controller.

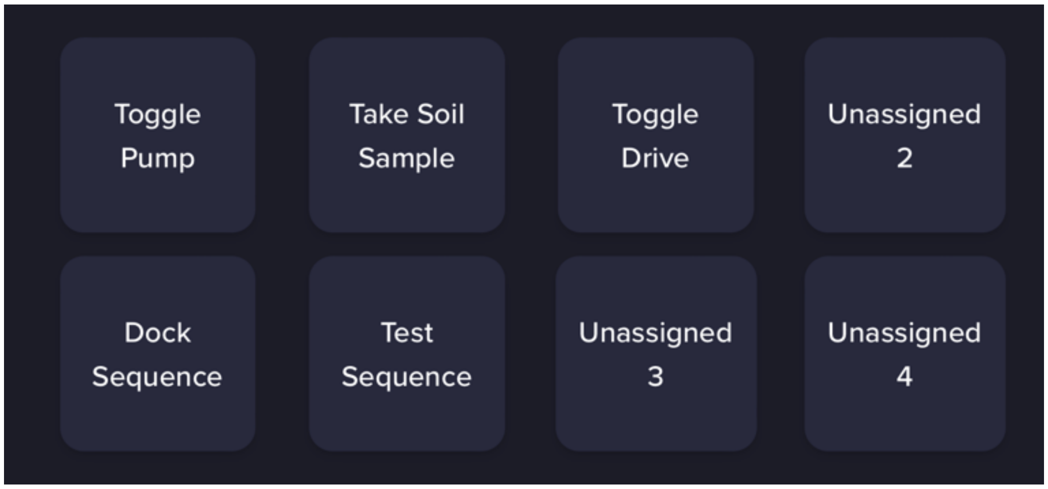


Figure 3.101: Robot Controller Implementation - Buttons

The eight buttons can be found in the center of the controller (Figure 3.101). Each time a button is pressed an event is triggered over the socket connection which the robot will then respond to. Each of these buttons triggers its own individual event. Even though three of the buttons do not have a current purpose, they can be easily updated later by changing the name and event name in an array.

Figure 3.102: Robot Controller Implementation - Joystick

There is a joystick (Figure 3.102) located on both sides of the controller. We implemented the joysticks by utilizing an already made node module called *React Joystick*. React Joystick allows for simple implementation of a joystick component. Since there was a necessity for two of these joysticks, one for forward velocity and one for rotational velocity. Each time one of these joysticks is moved a socket event is called to inform the robot that the joystick has been moved. To avoid overwhelming the robot by sending too many updates. The joystick was capped at sending no more than twenty updates per second. This many requests will allow for a quick response time but also not burn crucial resources needed to run other robot operations.

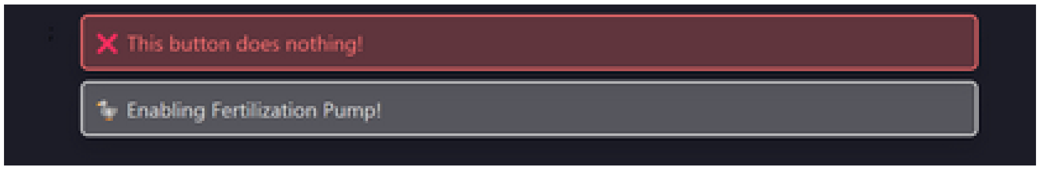

This button does nothing!
Enabling Fertilization Pump!

Figure 3.103: Robot Controller Implementation - Toasts

On the top of the controller is a space reserved for toast notifications for user feedback (Figure 3.103). These toasts inform the user of when an action is taking place upon the robot. Instead of writing a complicated toast management system, a react library was found called *React Hot Toast*. This library allows for simple creation and customization of toasts that auto remove after a few seconds.

3.4.0 System Integration

The integration activities of components are described in this section. This includes changes that arouse due to unforeseen interactions of subsystems. Additionally this section discusses how each core system was integrated into the overall robotic system. This section focused on the changes encountered and how they were addressed.

3.4.1 Ansys Simulations

Simultaneously as the mechanical system was being developed, so too was the electrical system, leading to many changes in the final weight of the system during development due to the battery. It became a concern as to whether the aluminum plates holding the chassis on the center axle would be able to withstand the full force of the chassis, and whether the rocker would be able to withstand the axial loads caused by the skid turning forces as well as the force of the chains. We also wanted to ensure the center axle would be able to carry the load of the chassis. We used ANSYS to simulate each of these situations. Simulation materials were limited to the materials that exist inside ANSYS's libraries for the sake of time.
To simulate the stress of the chassis on aluminum connecting plates, a 60 lb force was applied to the inside surface of the axle holes which connect the chassis to the plate, and the fixed surface boundary condition was applied to the inside of the bearing. The connection between the bearing and the plate is fixed for this simulation, and the material in ANSYS's 'Aluminum alloy'. Figure 3.104 below shows the result of the simulation's calculation of the total deformation of the part. With the minimal deformation, we concluded that only the planned pair of aluminum plates would be needed, instead of any material upgrades or additional plates.

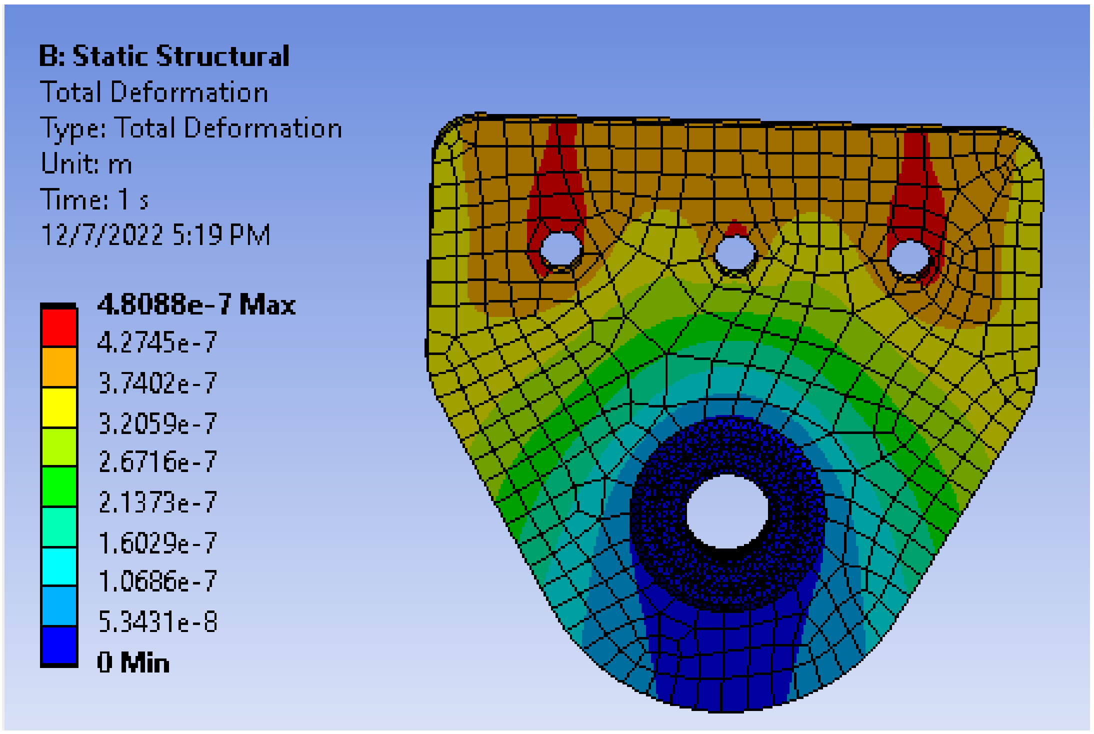


Figure 3.104: Aluminum axle plate ANSYS Simulation

To simulate the stress of the chassis on the center axle, the load of a 120 lb chassis was applied as a downwards force to the outer surface of the chassis bearings, with the fixed surface boundary condition applied to the rocker bearings. The type of connections in this simulation are all bonded, and the material of the center axle is carbon-annealed steel. Figure 3.105 below shows the result of the simulation's calculation of the total deformation of the axle. With this minimal deformation, we confirmed that the axle should hold the weight and withstand the stresses.

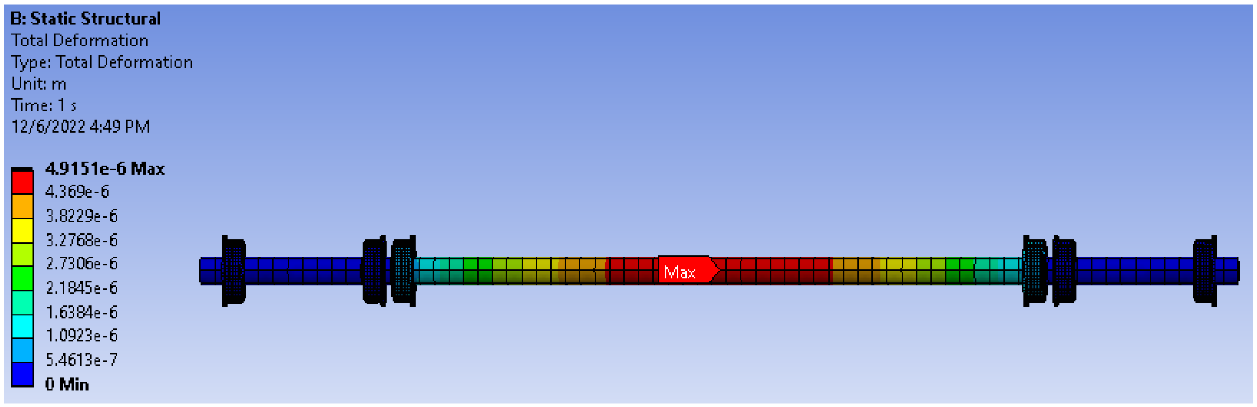


Figure 3.105: Steel Center Axle ANSYS Simulation

To simulate the stresses of skid turning and chain force on the rocker, the rocker model to test first had to be simplified by removing all the parts of the solidworks model that were irrelevant to the simulation(such as the screws, wheels, and gearboxes), and modifying the sprockets to be able to most accurately apply the chain force. It is worth noting therefore that the visual results on the sprockets should be omitted. Only one rocker was simulated at a time, with the inner surface of its center axle bearings acting as the fixed surface boundary condition. The axial and radial skid turning forces of 48 newtons, and 60 newtons respectively were applied to the hex wheel axles, and the calculated chain force was applied as a remote force on the surface of each modified sprocket as a surface effect, at the location where the chain would pull on the sprocket in the direction the chain would pull. The ANSYS materials used in the simulation are Aluminum Alloy, for the four rocker leg plates, and high-impact ABS Plastic for the gusset as a substitute for PLA. The sprockets, wheel axles, gearbox shafts, and gear box front-plate, (which was included in the simulation to transfer the force on the gearbox sprockets to the gusset without including the entire gearbox) used in the simulation were assigned the material of structural steel, since these objects were either made of steel, the deformation of the objects was not of interest ot the simulation, or both. Wood(oak) was used for the rocker standoffs, but additional values which were not in ANSYS were applied to this material in the simulation because the material properties within ANSYS were not sufficient to complete the calculation. At the time of the simulation, the plan was for these standoffs to be made of wood, but due to convenience and the availability of the material which would introduce less variance into the system, hollow aluminum standoffs were used instead. Everything else is structural steel and the connections are all bonded. The material properties of these substances are shown in Table 3.4. Figures 3.106 and 3.107 below show the results of the simulation's calculations of the total deformation of the rocker when turning left, and when turning right.

Table 3.4: Material Properties as recorded by ANSYS (evaluated at 23 °C)

Material	Tensile Yield Strength	Ultimate tensile strength	Density
Aluminum Alloy	2.8E+08 Pa	3.1E+08 Pa	2770 kg/m^-3
Structural Steel	2.5E+08 Pa	4.6E+08 Pa	7850 kg/m^-3
Plastic, ABS (high-impact)	2.7367+07 Pa	3.6387+07 Pa	1030 kg/m^-3

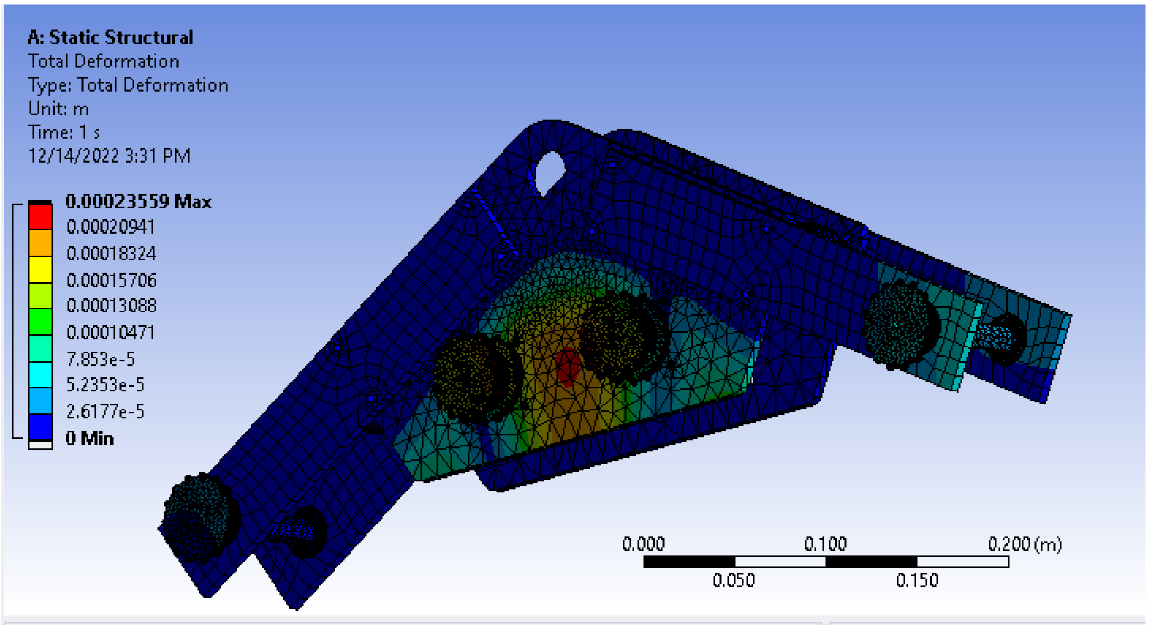


Figure 3.106: Rocker Skid Turning Left ANSYS Simulation

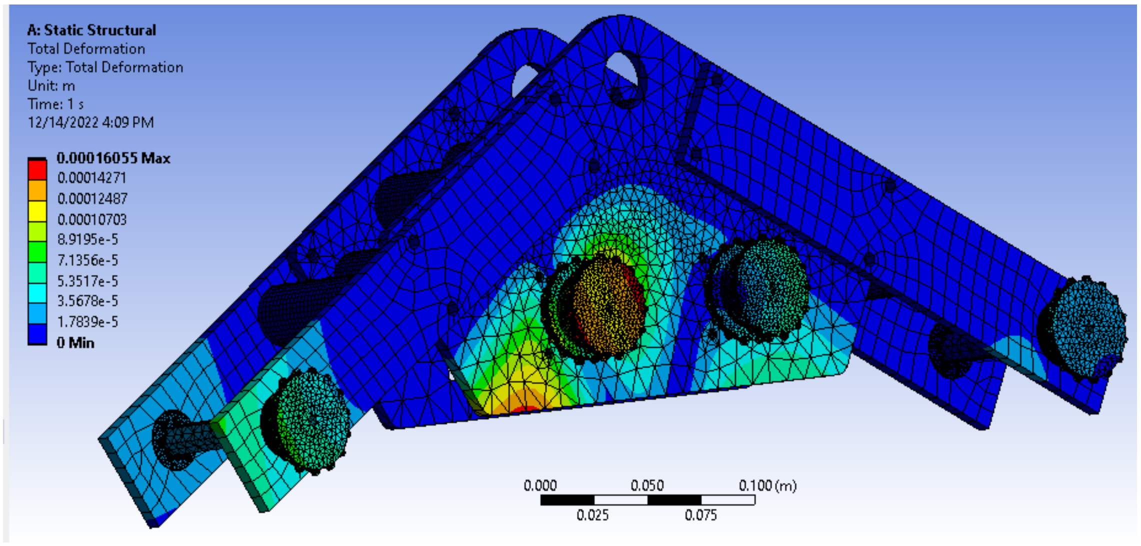


Figure 3.107: Rocker Skid Turning Right ANSYS Simulation

The simulations show that the deformations will be most prominent in the sprockets, which are only present in the simulation to correctly place the forces. The rest of the majority of the deformation appears concentrated in the center of the gusset, and while it is minimal, it is still at a somewhat undesirable magnitude. This contributed to the idea of using an additional 3D printed part to better secure the gearboxes to each other as well as the gusset, while also providing a stabilizing force to the inside of the gusset where this part would not interfere with the chains. The other major contribution to the development of this part was the realization that screws we believed were mounting screws were discovered to not have been mounting screws, making this part immediately essential as the only fixture-based mounting of the gearboxes to the rocker, as the other ‘mounting’ is the situation of the gearboxes through the opposite gusset. This additional piece is pictured in Figure 3.108.

Because our ANSYS analysis does not include key features added later which improve the stiffness of the rocker structure, including the gearbox mount, as well as 3D printed chain covers pictured in Figure 3.109 which are secured directly to the gusset and the rocker plates, the actual deformation is likely less than the deformation shown.

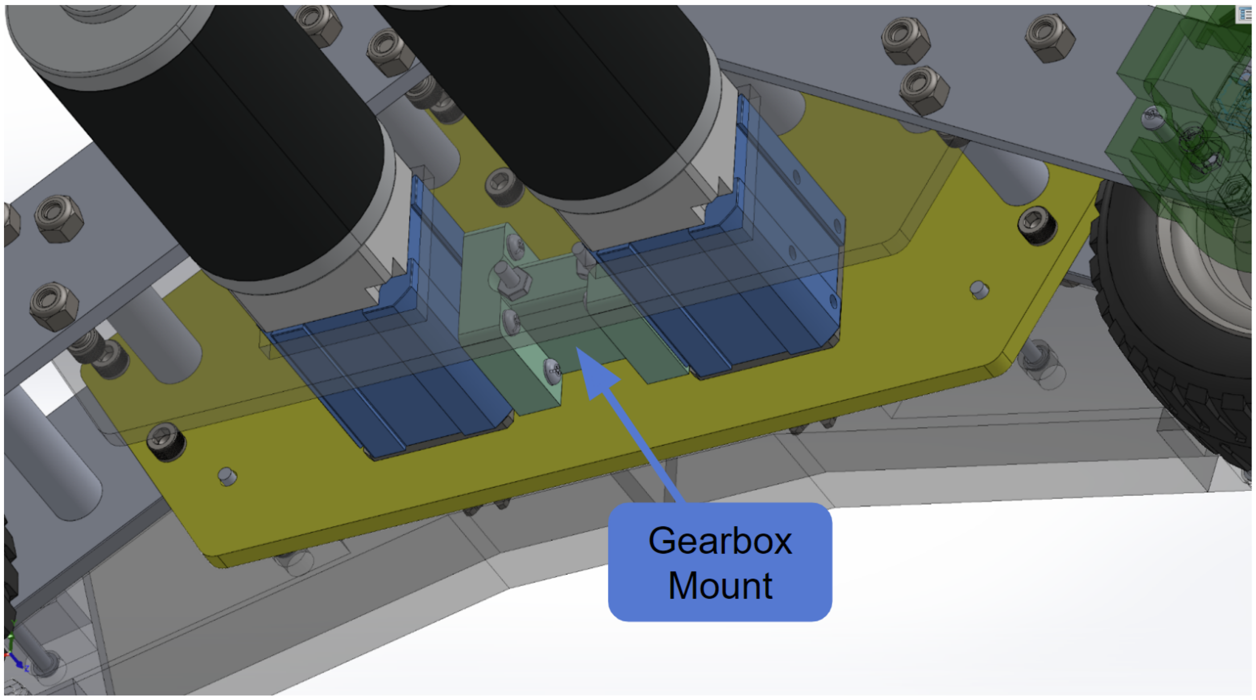


Figure 3.108: Gearbox Mount

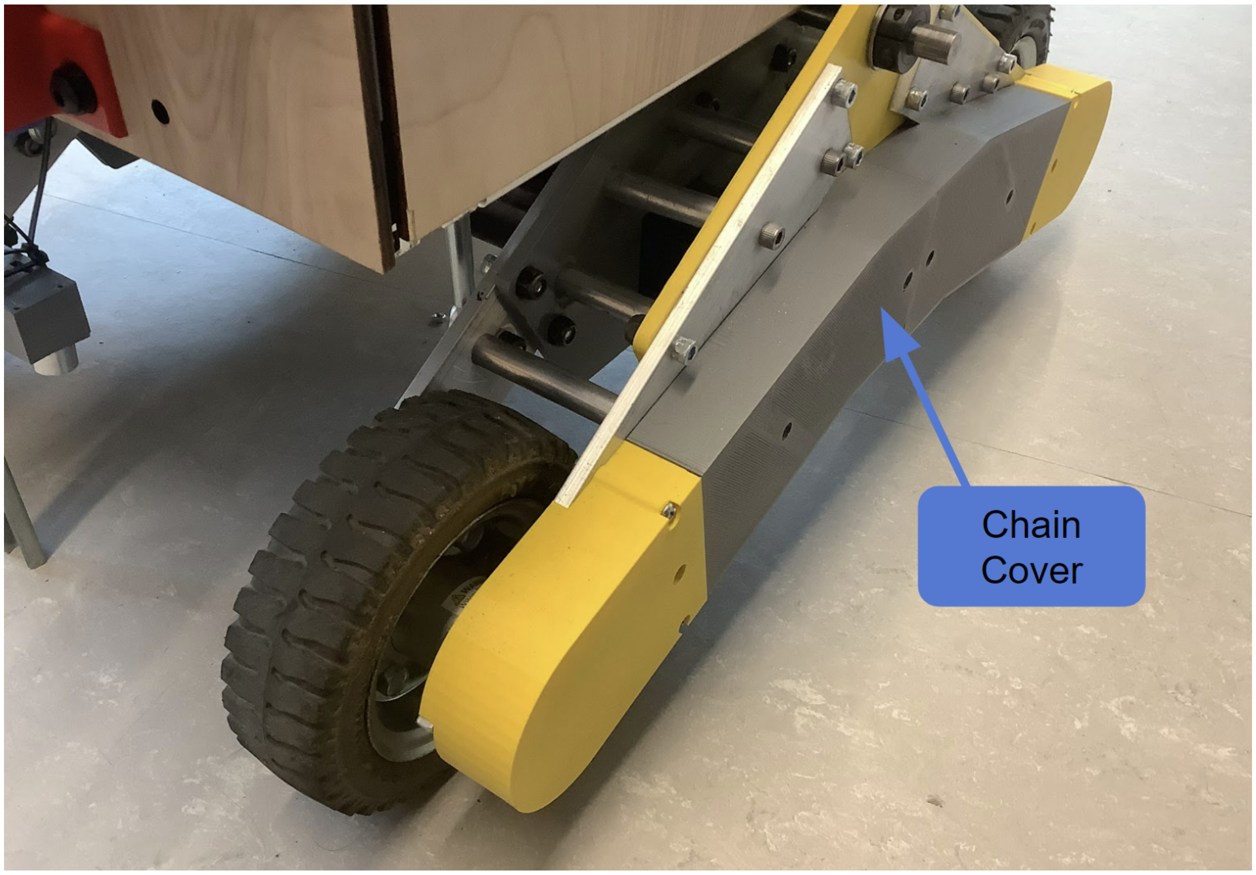
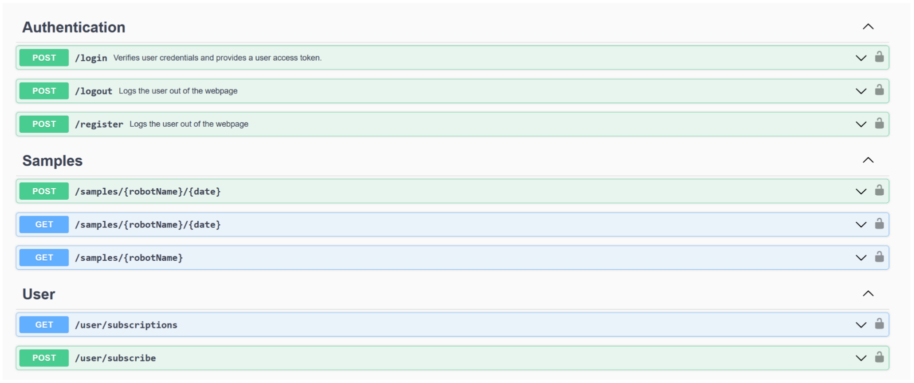


Figure 3.109: Chain cover

Chapter 4: System Testing and Validation

4.1.0 Web Server API Testing

The API endpoints managed by the webserver are crucial for interacting with the database. In order to ensure that these endpoints provide reliable information and respond to the correct requests, Swagger was used to create a testing dashboard for each endpoint(Figure 4.1). Once this dashboard was implemented, it was used to test each endpoint.



Authentication

POST `/login` Verifies user credentials and provides a user access token.

POST `/logout` Logs the user out of the webpage

POST `/register` Logs the user out of the webpage

Samples

POST `/samples/{robotName}/{date}`

GET `/samples/{robotName}/{date}`

GET `/samples/{robotName}`

User

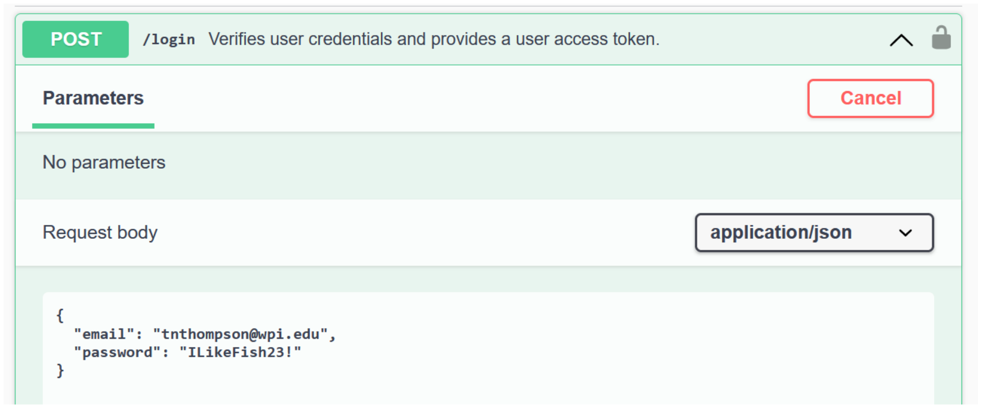
GET `/user/subscriptions`

POST `/user/subscribe`

Figure 4.1: Web Server API Testing - Swagger Dashboard

4.1.1 Account Login Endpoint Testing

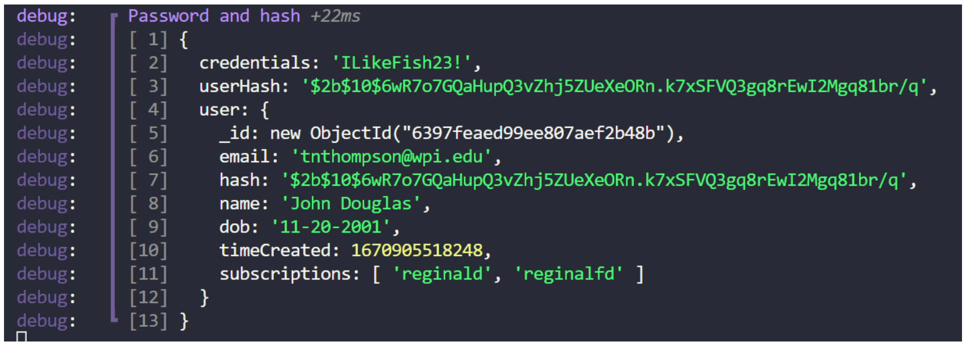
The account login request is responsible for logging a user into the website and returning them a JWT token to store as a cookie. This cookie will be necessary for storing the login state of the user, allowing them to stay logged in across separate sessions.


POST
/login Verifies user credentials and provides a user access token.
Parameters
Cancel
No parameters
Request body
application/json

```
{
  "email": "tnthompson@wpi.edu",
  "password": "ILikeFish23!"
}
```

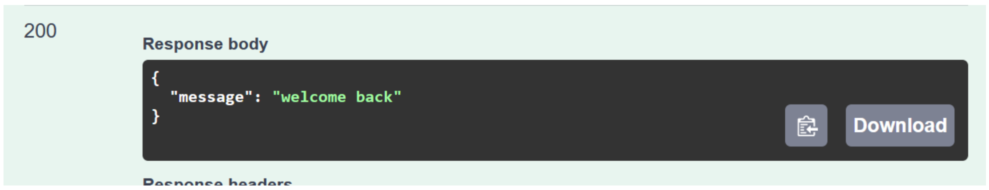
Figure 4.2: Web Server API Testing - Login Request

The login request takes the input of a user email and password, which can be seen in Figure 4.2.



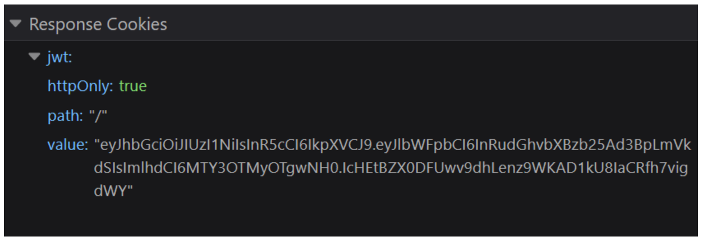
```
debug:   ┏ Password and hash +22ms
debug:   [ 1] {
debug:   [ 2]   credentials: 'ILikeFish23!',
debug:   [ 3]   userHash: '$2b$10$6wR7o7GQaHupQ3vZhj5ZUeXeORn.k7xSFVQ3gq8rEwI2Mgq81br/q',
debug:   [ 4]   user: {
debug:   [ 5]     _id: new ObjectId("6397feaed99ee807aef2b48b"),
debug:   [ 6]     email: 'tnthompson@wpi.edu',
debug:   [ 7]     hash: '$2b$10$6wR7o7GQaHupQ3vZhj5ZUeXeORn.k7xSFVQ3gq8rEwI2Mgq81br/q',
debug:   [ 8]     name: 'John Douglas',
debug:   [ 9]     dob: '11-20-2001',
debug:   [10]     timeCreated: 1670905518248,
debug:   [11]     subscriptions: [ 'reginald', 'reginalfd' ]
debug:   [12]   }
debug:   ┗ [13] }
```

Figure 4.3: Web Server API Testing - Login Account Lookup Development Login



```
{
  "message": "welcome back"
}
```

Figure 4.4: Web Server API Testing - Login Successful Response



```
▼ Response Cookies
  ▼ jwt:
    httpOnly: true
    path: "/"
    value: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InRudGhvbXBzb25Ad3BpLmVk
           dSIsImlhdCI6MTY3OTMyOTgwNH0.IcHEtBZX0DFUwv9dhLenz9WKAD1kU8IaCRfh7vig
           dWY"
```

Figure 4.5: Web Server API Testing - Login Account JWT Cookie

After the request is sent with a valid username and password, the webserver looks up the user by their username and password hash Figure 4.3. Once the account is found a welcome back message is sent to the user (Figure 4.4). In addition to this message, a JWT cookie is also sent in the response, which can be seen by analyzing the request response using the FireFox developer tools (Figure 4.5).

4.1.2 Account Logout Endpoint Testing

The account logout request is a response for logging the user out. This will be crucial for keeping accounts secure on public computers and also for changing accounts.

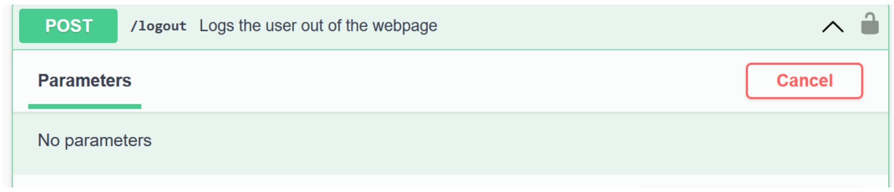
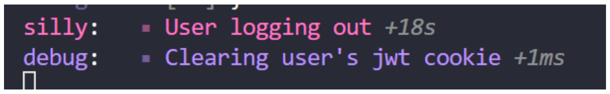


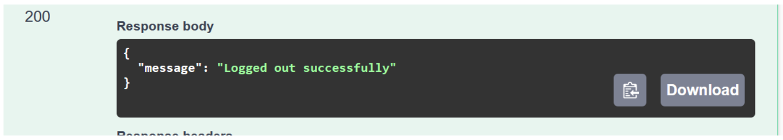
Figure 4.6: Web Server API Testing - Account Logout Request

The logout request takes no parameters, but uses the JWT cookie the user has stored (Figure 4.6).



```
silly:   ▪ User logging out +18s
debug:   ▪ Clearing user's jwt cookie +1ms
```

Figure 4.7: Web Server API Testing - Account Logout Development Log



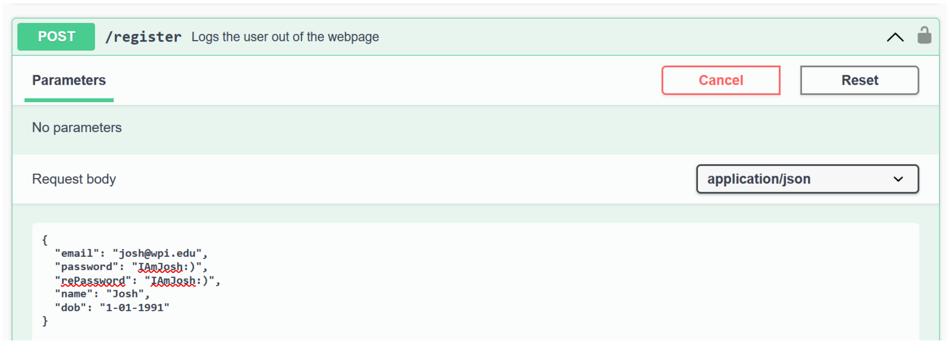
```
200     Response body
{
  "message": "Logged out successfully"
}
```

Figure 4.8: Web Server API Testing - Account Logout Successful Response

When the server receives the request it verifies that the user has a JWT cookie, and then clears it from their browser (Figure 4.7). Once the cookie is cleared, the server will response with a successful logout message (Figure 4.8).

4.1.3 Account Register Endpoint Testing

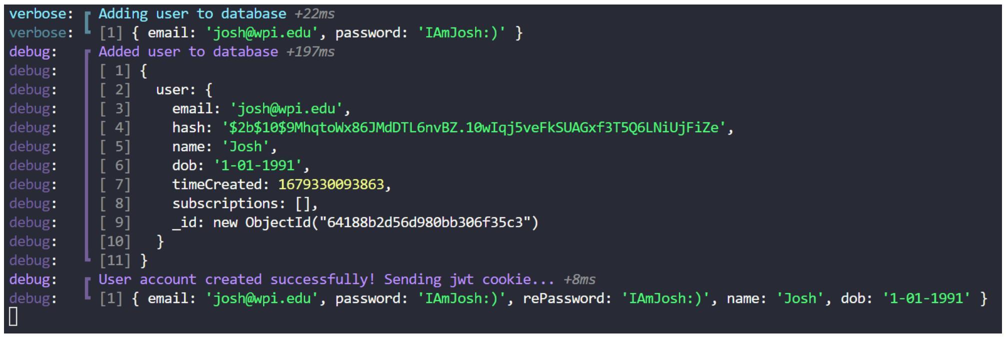
The account register endpoint is responsible for creating user accounts. This is crucial for allowing users to create their own account without an individual having to manually add their credentials to the database.



```
{
  "email": "josh@wpi.edu",
  "password": "IAmJosh:)",
  "rePassword": "IAmJosh:)",
  "name": "Josh",
  "dob": "1-01-1991"
}
```

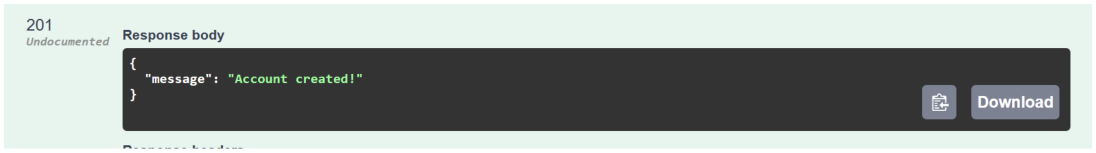
Figure 4.9: Web Server API Testing - Account Register Request

The register request takes the inputs of the user's email, password, repassword(to verify they put in the correct password), name, and date of birth (Figure 4.9).



```
verbose: ┌ Adding user to database +22ms
verbose: └ [1] { email: 'josh@wpi.edu', password: 'IAmJosh:)' }
debug:   ┌ Added user to database +197ms
debug:   │ [ 1] {
debug:   │ [ 2]   user: {
debug:   │ [ 3]     email: 'josh@wpi.edu',
debug:   │ [ 4]     hash: '$2b$10$9MhqtoWx86JMdDTL6nvBZ.10wIqj5veFkSUAGxf3T5Q6LNiUjFiZe',
debug:   │ [ 5]     name: 'Josh',
debug:   │ [ 6]     dob: '1-01-1991',
debug:   │ [ 7]     timeCreated: 1679330093863,
debug:   │ [ 8]     subscriptions: [],
debug:   │ [ 9]     _id: new ObjectId("64188b2d56d980bb306f35c3")
debug:   │ [10]   }
debug:   └ [11] }
debug:   ┌ User account created successfully! Sending jwt cookie... +8ms
debug:   └ [1] { email: 'josh@wpi.edu', password: 'IAmJosh:)', rePassword: 'IAmJosh:)', name: 'Josh', dob: '1-01-1991' }
```

Figure 4.10: Web Server API Testing - Account Register Developer Log



```
{
  "message": "Account created!"
}
```

Figure 4.11: Web Server API Testing - Account Register Successful Response

The web server will then create an object for the user to store in the database (Figure 4.10). Within this user object, the password is hashed in order to maintain a level of security in case the database is ever breached by an unknown party. Once the account has been successfully added to the database, the server responds with an account created message (Figure 4.11).

4.1.4 Robot Sample Dates List Endpoint Testing

The robot sample dates request is responsible for retrieving all of the dates soil samples were taken by a particular robot.

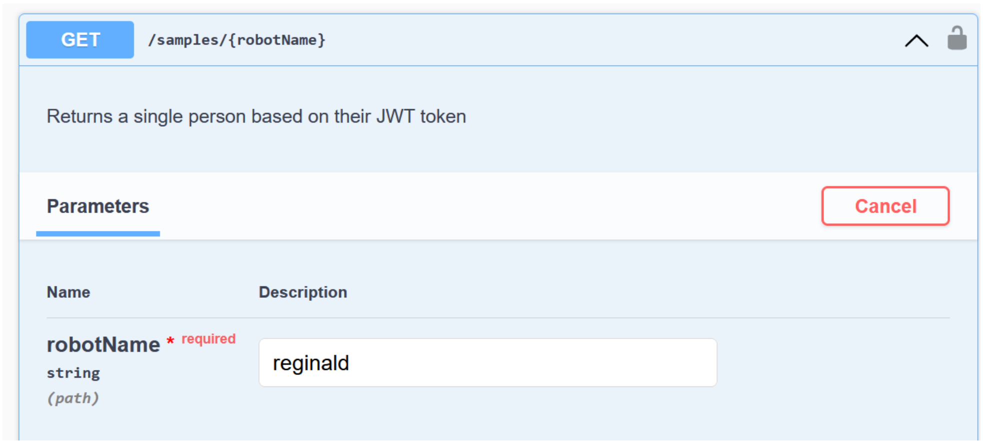
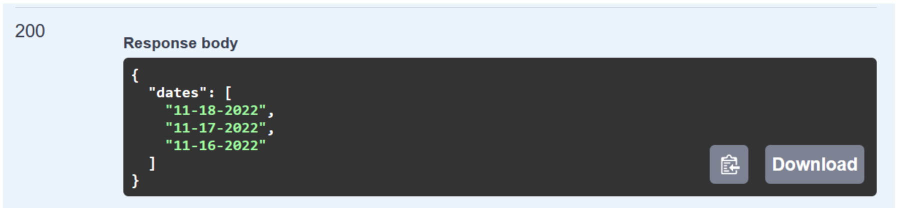


Figure 4.12: Web Server API Testing - Robot Sample Dates Request

This request takes a parameter of the robot's name in the parameter (Figure 4.12). In addition to this, the JWT token of the user is verified to see if the user has access to this data.



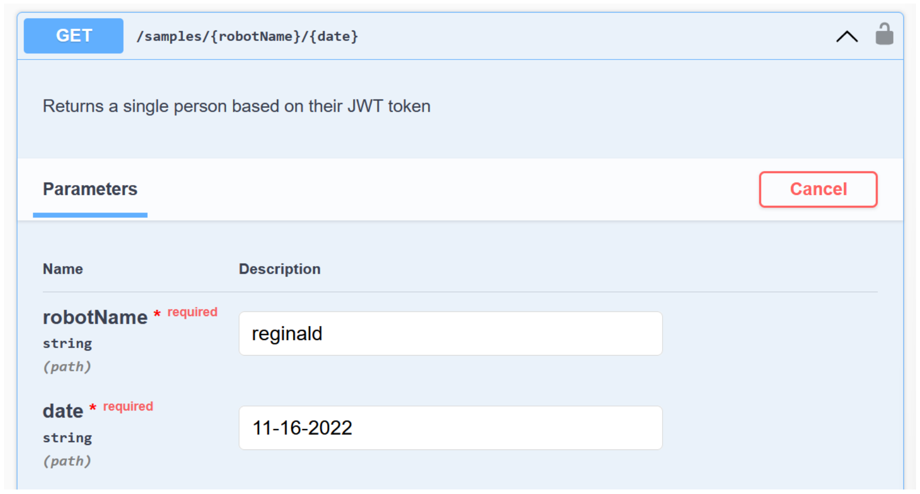
```
{
  "dates": [
    "11-18-2022",
    "11-17-2022",
    "11-16-2022"
  ]
}
```

Figure 4.13: Web Server API Testing - Robot Sample Dates Successful Response

If the robot name is valid and the user has access to this data, the request returns a list of all the dates where samples were collected (Figure 4.13).

4.1.5 Robot Samples from Date Endpoint Testing

The robot samples from the date request returns all of the soil samples that were taken by a specific robot on a particular date. In addition to this, the JWT token of the user is verified to see if the user has access to this data.


GET `/samples/{robotName}/{date}`

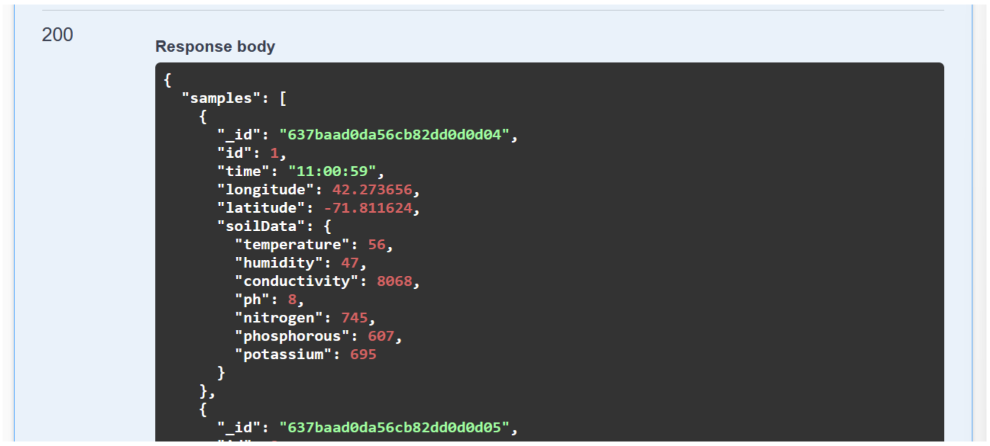
Returns a single person based on their JWT token

Parameters | Cancel

Name	Description
robotName * required string (path)	reginald
date * required string (path)	11-16-2022

Figure 4.14: Web Server API Testing - Robot Sample Request

This request takes the robot's name and date of when the sample was taken as a parameter in the url (Figure 4.14).


200 Response body

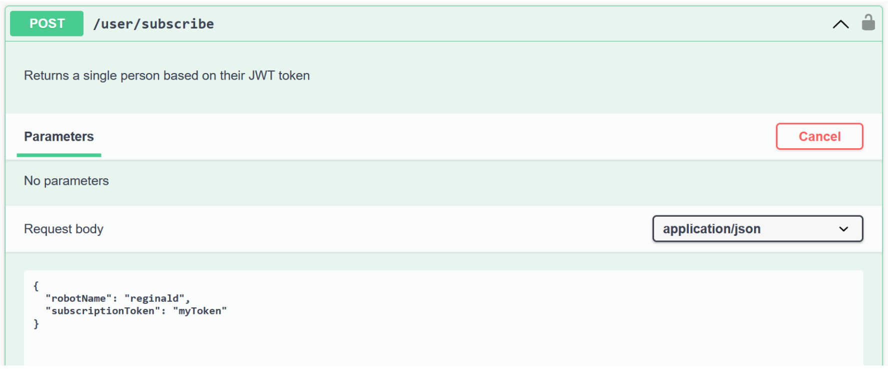
```
{
  "samples": [
    {
      "_id": "637baad0da56cb82dd0d0d04",
      "id": 1,
      "time": "11:00:59",
      "longitude": 42.273656,
      "latitude": -71.811624,
      "soilData": {
        "temperature": 56,
        "humidity": 47,
        "conductivity": 8068,
        "ph": 8,
        "nitrogen": 745,
        "phosphorous": 607,
        "potassium": 695
      }
    },
    {
      "_id": "637baad0da56cb82dd0d0d05",
```

Figure 4.15: Web Server API Testing - Robot Sample Successful Response

If the date and robot name is valid, and the user has access to this data the server will return a list of every sample that was taken for that date (Figure 4.15).

4.1.6 User Subscribe to Robot Endpoint Testing

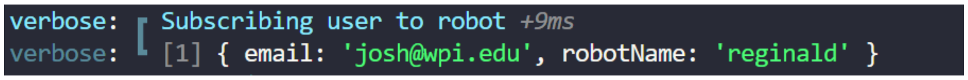
The subscribe to robot request is responsible for subscribing a robot to the data of a particular robot. This is used to ensure that only trusted users have access to the data collected by the robot.



```
/user/subscribe
Returns a single person based on their JWT token
No parameters
Request body
{
  "robotName": "reginald",
  "subscriptionToken": "myToken"
}
```

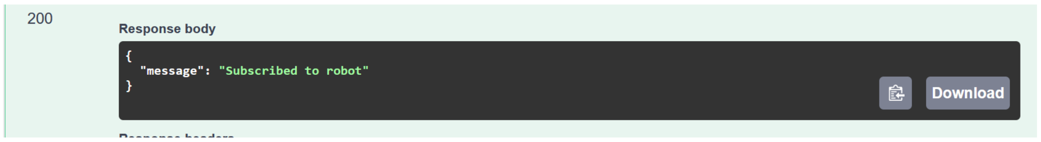
Figure 4.16: Web Server API Testing - Subscribe Request

The request takes in the name of the robot the user wanted to subscribe to, in addition to a subscription token that is used as a password when subscribing to a robot (Figure 4.16). The request also utilizes the JWT token that the user has stored, in order to lookup the user that is sending the request.



```
verbose: ┌ Subscribing user to robot +9ms
verbose: └ [1] { email: 'josh@wpi.edu', robotName: 'reginald' }
```

Figure 4.17: Web Server API Testing - Subscribe Developer Log



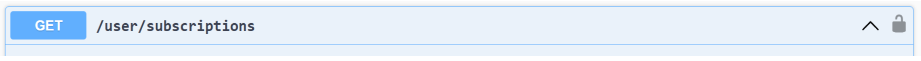
```
200
Response body
{
  "message": "Subscribed to robot"
}
```

Figure 4.18: Web Server API Testing - Subscribe Successful Response

If the robot name and subscription token are correct and the user is logged in, the server will add the robot name into the list of subscriptions stored in the user database (Figure 4.17). After the database is updated the server will respond with a successful subscription message (Figure 4.18).

4.1.7 User Subscription List Endpoint Testing

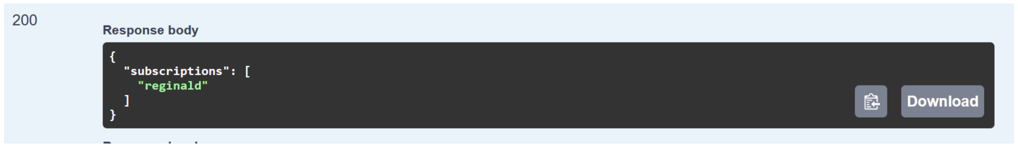
The user subscriptions list request is necessary for determining which robot data should be loaded for the user.



```
/user/subscriptions
```

Figure 4.19: Web Server API Testing - Subscription List Request

This request does not take any parameters, and only uses the JWT token stored in the user's browser (Figure 4.19).


200

Response body

```
{
  "subscriptions": [
    "reginald"
  ]
}
```

Figure 4.20: Web Server API Testing - Subscription List Successful Response

If the user is logged in and possesses the JWT token, the server will return a list of robots the user is subscribed to (Figure 4.20).

4.2.0 Suspension Testing

To verify that the rocker suspension is sufficient for traveling on dirt, the following tests were performed. The total vertical rise that the differential allows was measured, and was found to be 7cm. This value can be adjusted by changing the locations of the hard stop pieces of 8020, shown in Figure 4.21, that prevent the differential linkage plate from rotating past a certain point. These pieces can be moved closer together which would allow the differential linkage plate to rotate further, increasing the maximum vertical rise of the rockers.

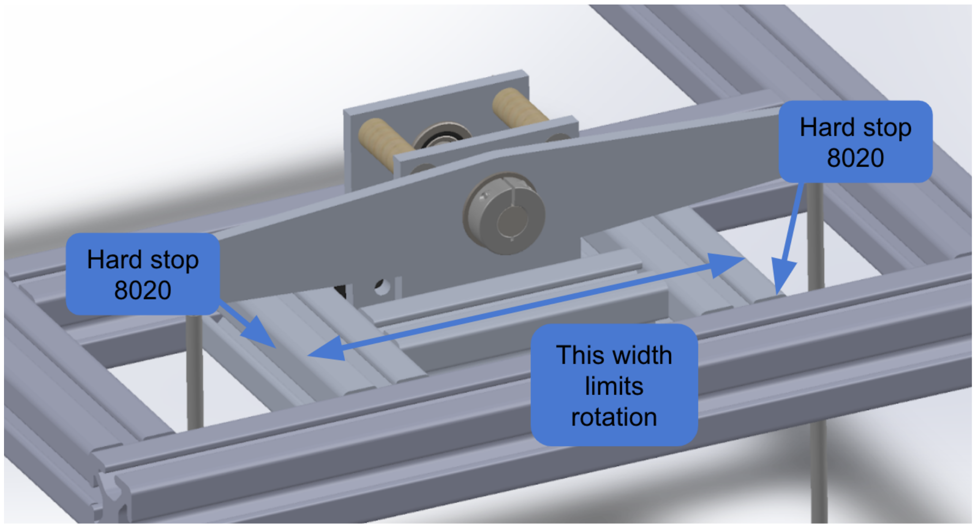


Figure 4.21: Hard Stop 8020 Pieces

The robot was also driven outside on dirt to test if the suspension would allow for skid steering in an environment similar to that of a farm. The robot successfully skid steered, and the rocker suspension allowed the robot to maneuver on uneven terrain.

4.3.0 Radio Testing

To verify the functionality of the radio, a test protocol was designed to simulate the main function of transmitting a soil sample. This was activated through the testing dashboard, where a button was implemented that triggers this protocol. The protocol creates a fake soil sample and sends it to the radio transceiver microcontroller over serial. Then the radio broadcasts this message, where the receiver is listening and then finds this data. The data is then sent over to the web server where it is placed inside the database. This test was run ten times for distances varying from 5 to 30 feet in a controlled indoor experiment. Throughout this trial there were no failed transmissions.

4.4.0 Drive Testing

To verify the functionality of the drive system there were three tests that took place: No Surface, Controlled Surface, and Dirt Surface. Each of these tests verified the functionality of the drive system under varying complexities. The No Surface ca test took place while the robot was upside down to ensure that the electrical, mechanical, and software components were working correctly to obtain the barebone result of having the wheels move. This test can be seen below in Figure 4.22. This test showed perfect results in the robot's ability to drive forwards, backwards, and skid steer.

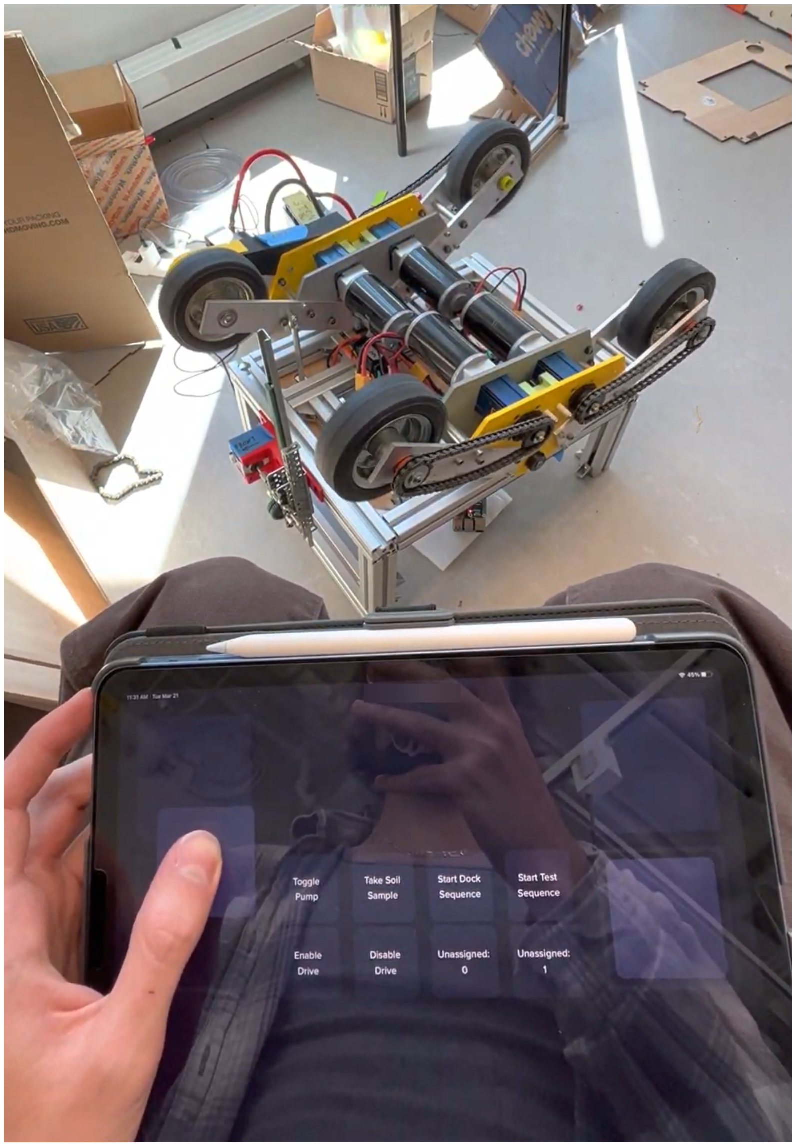


Figure 4.22: No Surface Drive Test

The Controlled Surface test was to ensure that each component worked together in harmony without any unforeseen problems while actually moving. This test which can be seen below in Figure 4.23 took place inside the lab on a tile surface. The test showed perfect results in the robot's ability to drive forwards and backwards. This test did uncover a problem in the robot's ability to skid steer. One of the wheel inserts was not a tight enough press fit for the amount of force acting on the wheel which allowed the wheel insert to spin within the wheel. To fix this issue, all of the wheel inserts were removed and then epoxied into the wheels. After securing them with epoxy, the robot was successful in skid turning on the controlled surface.

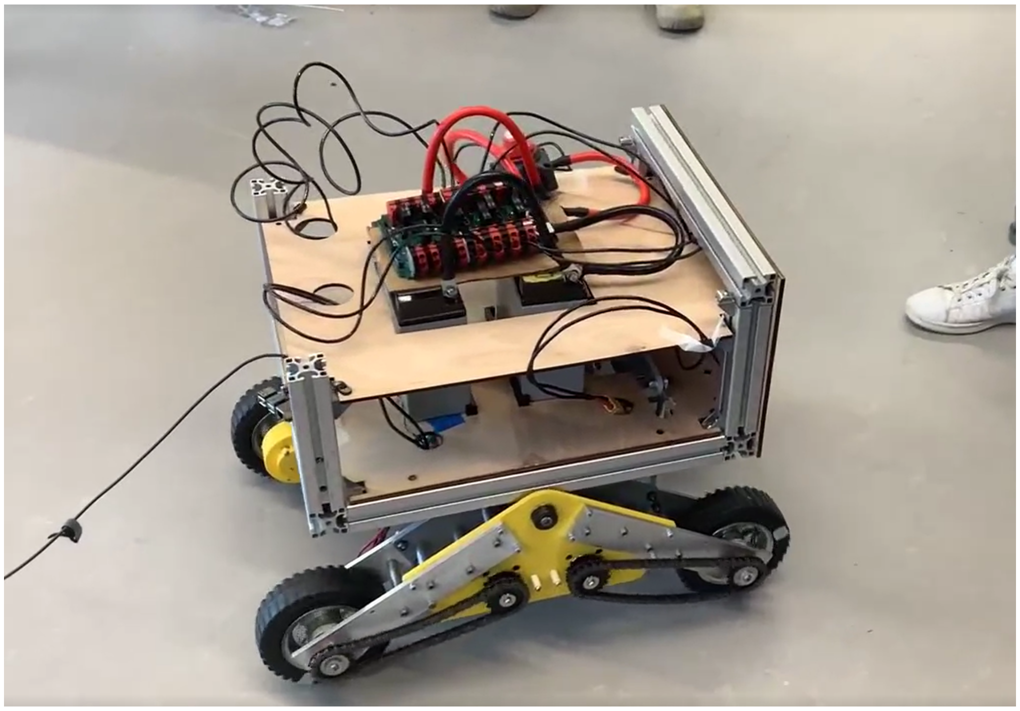


Figure 4.23: Controlled Surface Drive Test

The Dirt Surface test allowed us to verify that the robot could drive on the surfaces it was designed to drive upon. The test can be seen below in Figure 4.24 and took place in the field of campus since we did not have access to a farm. The test verified that the robot could handle moving forwards, moving backwards, and skid steering on uneven terrain.

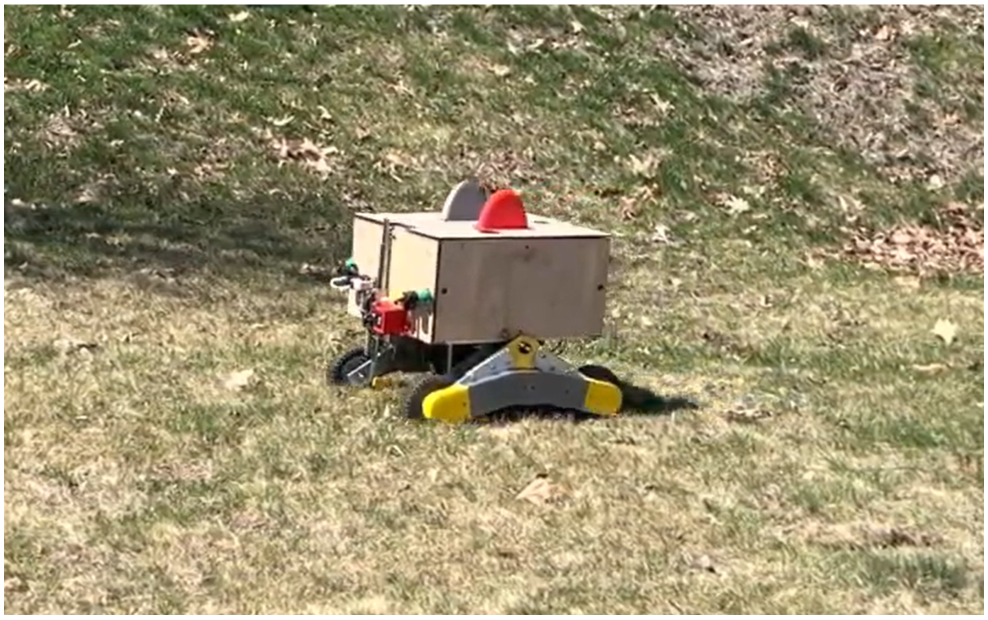


Figure 4.24: Dirt Surface Drive Test

Chapter 5: Conclusions and Recommendations

5.1 Software

The software work done throughout this project has successfully completed the project requirements. It has successfully created a system in which data can be sent from the robot to the web server. Additionally it has created an interactive way for this data to be examined. A fully customizable robot controller was implemented to greatly increase efficiency of running tests. Finally a controller system was implemented to successfully drive the robot and communicate with the devices onboard such as the radio transmitter.

The current software on board has left great room to expand upon and improve the overall functionality of the solar agribot system. The data collected by the robot should be analyzed in future teams to create accurate mappings of farm field conditions. This will help greatly in ensuring the robot fertilizes in only necessary areas. Additionally the necessity of the hub server should be considered by future teams, the final result of this project utilized the hub for less analysis than originally thought. There might be potential to remove it and the radios and replace it with a cellular chip. This would decrease complexity greatly of the overall data transfer system, and provide better ways to accurately track the robot in real time. Finally future teams should focus on improving upon the autonomous systems to create a fully autonomous system that can handle unexpected situations.

5.2 Mechanical

Although the robot chassis was completed and was able to function as intended, there are still some improvements that could be made to the mechanical systems. The first recommendation for future work is to run further testing of the robot in dirt on an actual farm to ensure that the chassis and suspension is appropriate for that setting. Another thing that could be changed for a new agribot design would be to use smaller motors and gearboxes so the robot can be narrower and able to fit between closer crop rows. Smaller batteries would also allow for a lighter weight robot which may be more suited for skid steering in loose dirt. Utilizing a lighter weight material for the frame would also be beneficial. The 80/20 allows for a modular design but a more permanent frame made of a material such as carbon fiber would significantly reduce the weight. The main body should also be waterproofed to protect the electrical components. The fertilizer tank and batteries should be secured more firmly if this design is to be continued. Running analysis tests to compare the values found in ANSYS to better understand the internal forces of the robot would be advisable. Re-printing and re laser-cutting dysfunctionally–modeled parts that had to be modified after fabrication inorder to install them would contribute to a better experience working with the robot.

While the design for the soil probe was complete in its capabilities, there were still some issues that were encountered. One issue involved the limited range the soil probe had in its linear movement. 2 inches was not enough range for the probe to be actuated high enough to prevent from dragging against ground surfaces when the robot encounters an incline. Its range of motion was also limited by the top stationary block labeled in blue from Figure 35. The block hits the bottom of the chassis, preventing any further upward movement. Mounting the soil probe at higher location would allow for greater clearance, however this change would require a longer railing and rack to account for an increased vertical distance.

The vertical distance between the current mounting placement and the next available mounting location was 9.5 inches. Another possible solution is to replace the outside truck labeled in the Figure 35 with one capable of fitting within the railing and sliding. This increases the range of motion by removing the blocks that hit against the chassis. Another issue related to the limited penetration depth of the NPK sensor prongs. The sensor requires a depth of 2 inches within the soil to properly measure soil data. The design hadn't allowed for this. A solution is to replace the bottom block with one that fits within the rail. With this change, there will still need to be a method of stopping the probe from actuating further. This is something that needs to be explored for future designs of the soil probe mechanism.

5.3 Electrical

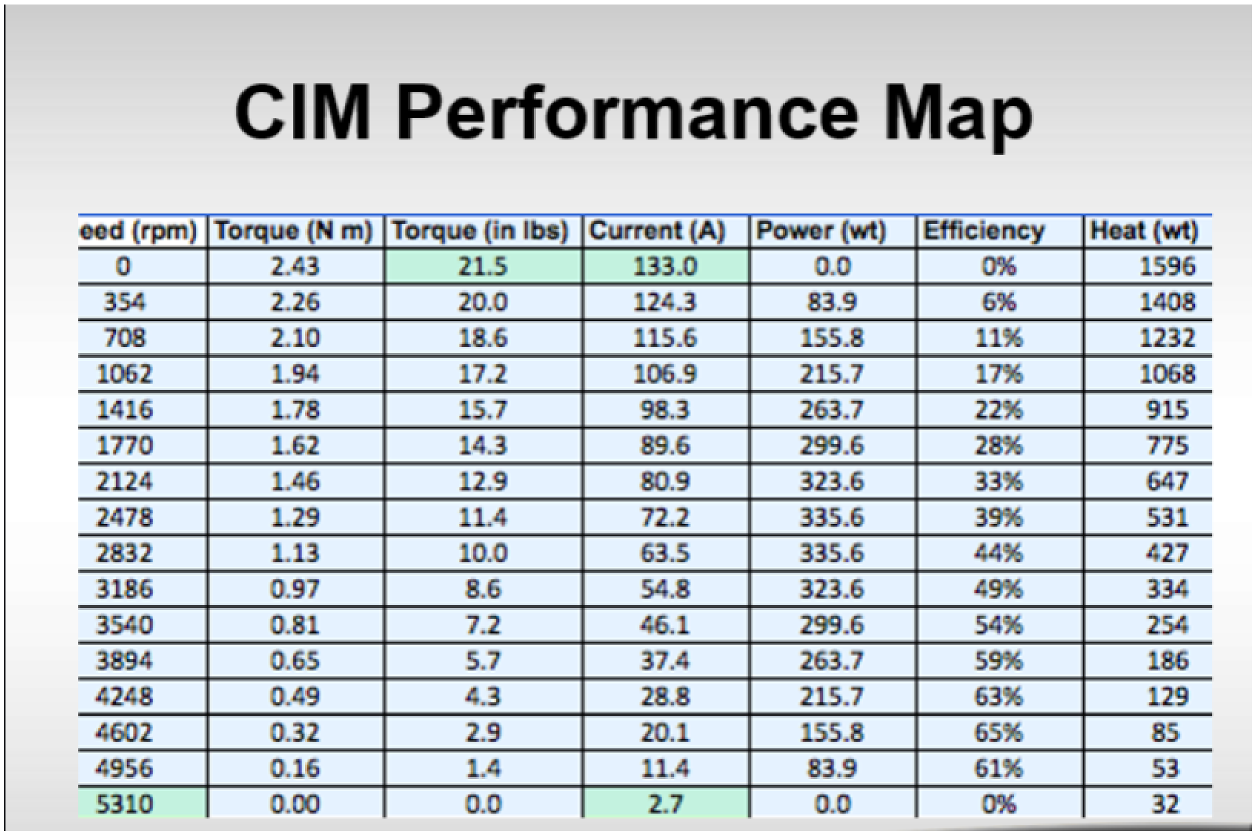
For the electrical components, each core functionality was successfully completed to meet the project requirement as a modular design, meaning that they work independently as subsystems on the rover. The design can be divided into three parts. Firstly, the power system is successfully integrated into the rover providing enough current to the drive motor for skid turning, regulating voltage for microcontrollers and subsystems, and generating feedback via ESP32 over Wifi to detect the battery level between the hub and the rover. Secondly, the navigation system is successfully working as a complete subsystem. The Teensy 4.1 can pass the data from the sensor-fused compass heading IMU to the Raspberry Pi 4B via serial protocol at 2 degrees accuracy deviation . The RTK successfully passes the latitude and longitude data to the Arduino which in turn passes the serial data to the Raspberry Pi 4B at 10 cm accuracy deviation. Lastly, the subsystem of the sliding mechanism is fully functional despite the mechanical design setback. The stepper motor can actuate the slider according to the varying ultrasonic distance feedback as well as the current sensor to prevent stalling. In addition, MODBUS commands were able to request and receive the NPK value via RS485/TTL protocol. This subsystem handles serial commands from the mainframe Raspberry Pi 4B to trigger different state machines and return data.

Most of our subsystem works according to the requirement as an independent module. However, our team had a time constraint that prevented us from testing them as a part of the larger system. In other words, as a whole, the electrical system did not meet the requirements of the robot because not all of the subsystems were fully integrated on the rover. There are a lot of improvements that can be made. For example, the fertilization controller that actuates the water pump using a relay (although works independently) has never been interfaced to handle serial commands from the Raspberry Pi. Moreover, the wire management could be improved by moving the entire 2 and 4 gauge wires inside the rover body frame. This would drastically improve the sensor readings that are heavily affected by magnetic interference. Additionally, adding a current sensor on the CIM motors can be very beneficial for power analysis and control systems. Moreover, setting up an outdoor permanent base to house the RTK so that it can run the entire 48 hour survey(base tower accuracy of up to 10 cm deviation) of the fixed coordinates will immensely improve the results. Additionally, further PID tuning is required for greater flexibility of the maneuver algorithm. The Lidar system that has been tested but not implemented could be a game changer for achieving greater autonomy. Finally, future teams should focus on signal processing, more accurate power analysis, implement subsystems and more robust control designs.

Appendix A: Source Code

The complete source code for the project was combined into one repository can be found below
https://github.com/TravisThomp/solar-agribot-mqp

Appendix B: CIM Motor power consumption datasheet



CIM Performance Map

eed (rpm)	Torque (N m)	Torque (in lbs)	Current (A)	Power (wt)	Efficiency	Heat (wt)
0	2.43	21.5	133.0	0.0	0%	1596
354	2.26	20.0	124.3	83.9	6%	1408
708	2.10	18.6	115.6	155.8	11%	1232
1062	1.94	17.2	106.9	215.7	17%	1068
1416	1.78	15.7	98.3	263.7	22%	915
1770	1.62	14.3	89.6	299.6	28%	775
2124	1.46	12.9	80.9	323.6	33%	647
2478	1.29	11.4	72.2	335.6	39%	531
2832	1.13	10.0	63.5	335.6	44%	427
3186	0.97	8.6	54.8	323.6	49%	334
3540	0.81	7.2	46.1	299.6	54%	254
3894	0.65	5.7	37.4	263.7	59%	186
4248	0.49	4.3	28.8	215.7	63%	129
4602	0.32	2.9	20.1	155.8	65%	85
4956	0.16	1.4	11.4	83.9	61%	53
5310	0.00	0.0	2.7	0.0	0%	32