VALENCIACOLLEGE

Department of Electrical and Computer Engineering Technology (ECET)
Division of Engineering, Computer Programming, and Technology (ECPT)

EET 4950

Senior Design Project

Personal Automated Lawn Mower (P.A.L.M.)

Submitted by

Brian Darling and Gabriel De La Torre

Supervised by

Prof. Notash

July 23, 2021

Abstract

Lawn maintenance can be time-consuming and exhausting, especially for a highly active family. Not to mention the hot summer heat can be brutal on the elderly and those with health conditions.

A fully automated lawn mower that can maintain the yard can help time-restricted families make more time for other things they need to do, reduce weeds growing in the yard, and save households money over time from lawn maintenance services.

This mower will be a fully automated system that will cut the grass on autopilot with the user's preferred cutting consistency. To easily and efficiently control the cutting area, the mower will utilize a specialized GPS module to help the mower stay within its cutting boundaries.

With the addition of solar panels and rechargeable batteries, the mower can also reduce the household's carbon footprint and noise output. The mower will self dock at a charging station and have solar panels on the unit itself and be self-charging.

Acknowledgements

We would like to give thanks to Prof. Notash for helping us with this project every step of the way. His clear and precise guidance allowed us to complete a project that we could be proud of as effectively as possible. This includes his constant feedback during the weekly meetings, as well as his timely response to any separate questions sent to him directly.

We would also like to give thanks to Valencia College and all of its staff for helping us get this far. Thanks to all the great professors we had, we were able to learn the skills necessary to complete this project and to take one step closer to our dream of becoming true professional engineers.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Summary

In this chapter, we present the basic principles of the P.A.L.M., a fully automatic lawn mower designed to remove the effort of mowing the lawn as well as the high cost of continuous payment for lawn mowing services. This chapter will discuss the features, similar products, and the goals of the project.

1.1 Introduction

Mowing the lawn has always been one of the many painstaking tasks that are required to maintain the look of one's home. The two options have always been to either put in the work and mow it yourself or to pay someone to do it for you. The former requires a significant amount of work and the latter can get very expensive over time.

The P.A.L.M. is a fully automated lawn mower that can be set up to mow the lawn with practically no input from the user besides the initial setup. With its ability to be programmed to the requirements of the user and to automatically recharge its own battery, this mower can provide continuous maintenance of the user's lawn for an extended period of time on its own. This would eliminate the need to continuously spend money on lawn maintenance and the inconvenience of having to cut the grass yourself.

1.2 Motivation

The inspiration for this idea came from the fact that many people struggle to maintain their lawn because they either cannot afford to pay for lawn maintenance or because they are unable to cut the grass themselves. This can include people with disabilities, senior citizens, or even those that simply do not have the time to constantly mow their lawn. Having a fully hands-off automatic mower would solve the economic issue by making only one purchase that would pay for itself over time and the physical issue by removing nearly all the effort of mowing the lawn.

1.3 Objectives and Features

The P.A.L.M. requires several features to help it provide the automatic mowing experience. It needs to be able to maneuver the lawn automatically, to recharge its batteries on its own, and to avoid any obstacles it might find while it is cutting. Moreover, the device also features a manual mode where the user can control the mower using a remote controller if they wish to cut the grass themselves while sitting in a chair.

1.3.1 Automatic Lawn Maneuvering

The automatic maneuvering will be achieved by utilizing a specialized GPS and compass navigation to determine the cutting area of the lawn and using this shape to automatically create a path within the designated cutting area to travel.

1.3.2 Automatic Recharging

The mower will have a continuous charging system by using its integrated solar panel that will be mounted on top of the mower. This will help increase the battery life while the system is cutting grass as it will continue to charge while in use.

1.3.3 Manual Control Mode

The manual control will be done through the controller and then simply steering the mower as if it was a remote control car. This would give the user the option to cut outside of the setup barrier if the need should arise.

1.3.4 Obstacle Detection

The obstacle detection will be achieved by placing sensors on the front of the mower. Whenever the device detects an object in its path, it will inform the mower that it needs to change its path in order to avoid an obstacle. The mower will simply keep turning and try to move forward until it no longer bumps into the obstacle and then continues cutting on its programmed path.

1.4 Similar and Existing Products

In our research, we found similar products to our project that were lacking some of the more useful features that we are planning to implement in this project. Our device intends to improve on existing designs to create a more efficient and self-sustaining system.

1.4.1 MowBot

The MowBot is a lawn service that uses mowing robots to cut and maintain a yard [1]. It uses a wire to form a boundary around the lawn that is installed by the service technician.

In order to cut the lawn, the robot moves in an irregular pattern which, given enough time would allow it to mow most of the grass while staying within the boundary wires.

1.4.2 Husqvarna Mowers

The Husqvarna Mowers also use wires to establish boundaries while utilizing GPS modules to help guide its movement patterns [2]. This device can also communicate with the user's phone to send important notifications.

Figure 1.1 Husqvarna AutoMower [2]

1.4.3 MowRo

The MowRo also uses boundary wires for guidance while relying on an algorithm to control its movement. This device also includes safety features that encapsulate the blades in order to prevent accidents and can detect rain in order to prevent water damage to the device.

Figure 1.2 MowRo and Docking Station [3]

1.5 Block Diagram

The block diagram defines the modules needed for the project and shows the flow of communication between the different modules. The block diagram will consist of three assemblies, the mower, docking station, and controller.

The mower module is the main module of this product, it will have two control systems, the mower control module to manage the mower functions and movement direction, as well as a specialized GPS module that will communicate with a similar GPS module in the docking station for accurate positioning. The main source of power for this module will be a rechargeable battery.

The docking station module will be the mower's return to home location as well as the unit that provides real-time GPS corrections via the specialized GPS module. This system can provide accuracy down to 3 cm and allow for accurate pathing [4]. This station will be powered by a rechargeable solar power module and have an inductive charging system to help boost the charging time of the mower after it is done mowing.

The controller will be a simple handheld wireless controller that allows the user to direct the mower for programming GPS coordinates and manual mower control for custom cuts.

During the proposal phase, the design was a little more simplistic as shown in figure 1.3 and we fully anticipated being able to easily create a phone app or use a simple Bluetooth controller to manually control the system. We also had plans of installing a humidity sensor into the design to allow the mower to detect if it is raining or if the ground was too wet to cut, but the team ran out of time to integrate this feature into the final design.

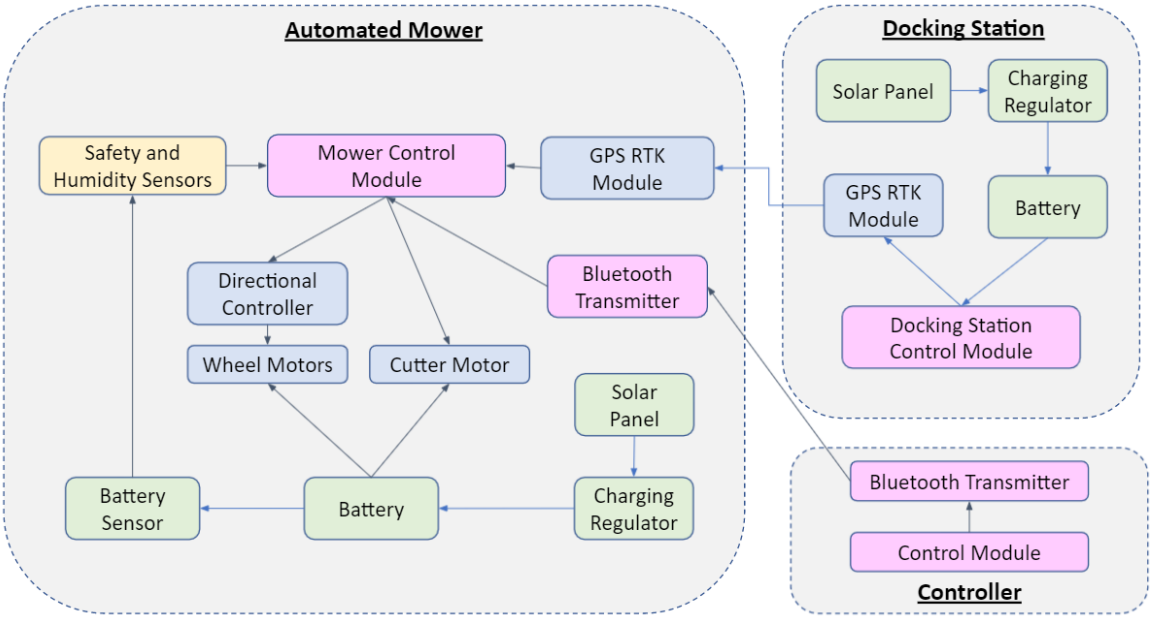

Figure 1.3 Original P.A.L.M. Block Diagram

After the proposal presentation, the industry experts suggested extra safety features to make the mower safer to handle and cut power to the cutter blade. This was implemented into the power system and we added two power switches, the main power switch, and a cutter blade power switch.

In addition, we made updates to the docking station as we found that the GPS module that is providing positional corrections to the mower does not require a Raspberry Pi to control the output of the system, which simplified the design and cut costs. In addition to these changes, our discussions with our adviser requested that we install a way for the docking station to provide power to the mower while docked and to think about users who may not have a smartphone, which leads to us adding an inductive charge system and changing the controller from the pre-planned phone app to a standalone wireless controller as shown in figure 1.4 below.

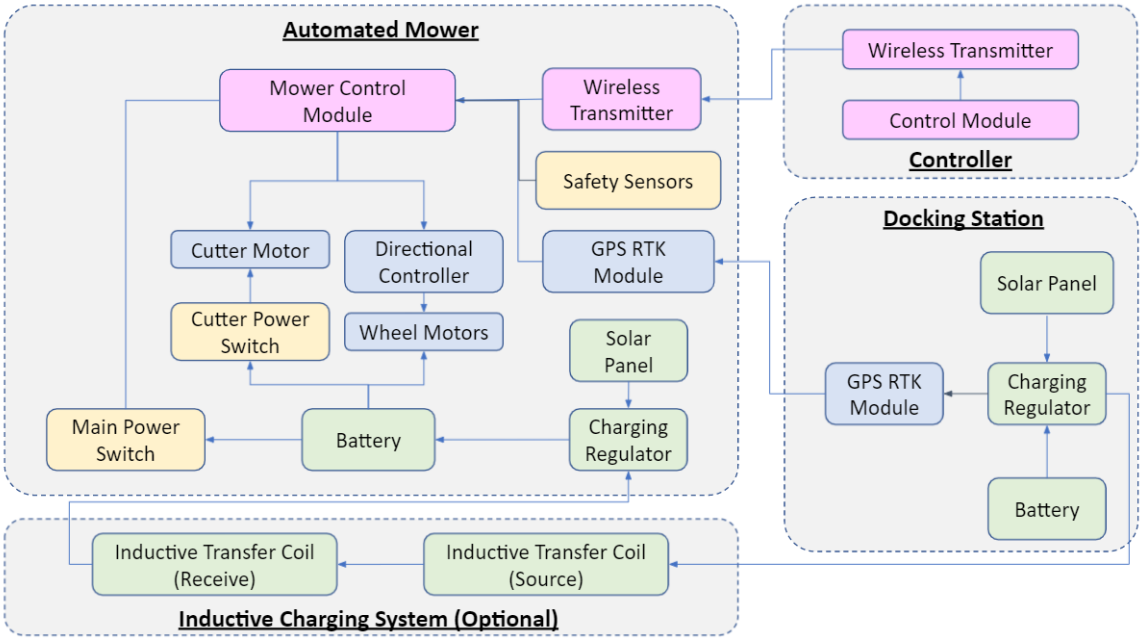

Figure 1.4 Updated P.A.L.M. Block Diagram

1.6 Engineering Requirements and Specifications

In order to confirm the product is a success, the P.A.L.M. shall meet the following Design Engineering Requirements (DER) as shown in Table 1.1 below.

These requirements are separated into high, medium, and low-level requirements and are ranked on importance level for the desired function of the product. We will need at minimum all high-level requirements to be fulfilled for the P.A.L.M. to function at the bare minimum design requirements and function properly.

The medium-level requirements are additional features that we would like implemented in the design to supplement the high-level features and provide the product with additional desirable functions that the intended consumers may find useful.

The low-level requirements are bonus features to further enhance the overall product and create a more desirable overall product.

Table 1.1 Design Engineering Requirements (DER)

Design Engineering Requirements		
Level	**Requirements**	**Verifications and Success Criteria Tests/Trials**
High	1. The user shall be able to teach the mower which sections of the lawn to mow. 2. The mower shall be able to successfully mow a predetermined section of the lawn without further help from the user. 3. The mower shall actively charge by using the attached solar panel to extend battery life. 4. The mower shall be able to return to the docking station once it has done mowing.	● The controller allows the user to interact with the mower and program key locations. ● Once the lawn borders are created the mower will follow a predetermined path until complete. ● Microcontroller(s) will monitor the battery source and pause mowing when power levels are too low. ● Once the mower has completed cutting, it will return to the docking station.
Mid	1. The mower shall be able to automatically avoid obstacles in the yard using sensors. 2. The mower shall remember the mowing path for future use.	● The mower will utilize sensors and move around the object(s) in its path. ● The mower will follow the same repeatable path each iteration.
Low	1. The mower's power source shall provide at least 45 minutes of continuous mow time. 2. The mower shall be able to charge at the docking station.	● Tests will be performed both in direct sunlight and in the shade to determine battery life while in use. ● Tests will be conducted to verify power transfer from the docking station to the mower.

The P.A.L.M. provides automatic maintenance of the user's lawn with the only contact required being the initial setup of the mower. The engineering specifications table shows the project's requirements and product specifications. These specifications ensure ease of use for the user as well as automatically recharging and deploying the system to ensure the lawn is constantly being maintained.

Table 1.2 Design Engineering Specifications (DES)

Mower				
Part	**Specific Component**	**Engineering Specification**	**Justification & Verification**	**Responsibility**
Microcontroller	Raspberry Pi 3B+	Must be able to run off a 12V DC power source. Controls motor and directional controls to determine where to move and what is left to cut. Must be able to detect at least 3 safety and detection sensors. Must be able to use GPS and compass antennas to determine global position.	**Justification:** Needs to be able to apply custom code and respond to GPS, sensor, and pre-mapped paths. The controller also must be able to monitor speed, direction, and cutter speeds. **Verification:** Controller will verify GPS and sensors are active. If a pre-mapped path is not created a new path will be generated by the code or user input.	Gabriel

Car Kit	Robot Smart Car Chassis by XiaoR	Must be able to support the weight of the solar charging kit, batteries, motors, and electronics. Must have enough room to hold all components and allow the cutter blade to move freely. Must be able to traverse multiple terrains and ground conditions.	**Justification:** The frame is capable of carrying the solar panel on top, and electronics within, and a cutter blade, drive motors, and battery on the bottom. **Verification:** Aluminum construction frame and metal track gears will be able to support the weight.	Group
Solar Panels	12V 10W Flexible Monocrystalline	Must be able to charge batteries both while docked and while the mower is moving. Must be able to charge a 12V battery source. Must weigh less than 1 kg	**Justification:** The battery must be able to be recharged within 8 hours of direct sunlight. **Verification:** Solar Panel test will be checked before installation and the microcontroller will verify battery charge before use.	Group
Blades	Stainless Steel	Must be lightweight and durable. Must be able to cut grass efficiently.	**Justification:** The cutter blades must not tax the cutter motor and can be easily replaced if damaged. **Verification:** A test run of the motor will be run to verify that the cutter mower is not experiencing a high current drain.	Brian

Battery	12V 6Ah LiFePo4	Must be able to fit in the designated assembly frame. Must be able to provide 9V power to motors. Must be able to provide at least 45 minutes of run time.	**Justification:** The battery pack must fit within the allotted space on the unit and be able to provide more than the required voltage and current. The battery must be able to power the unit for at least 45 minutes at a time. **Verification:** The microcontroller is able to display power available from the battery.	Brian
Object Detection Sensor	Bumper Sensors	Must be passive and not have an excessive drain on batteries.	**Justification:** Sensors will close a signal loop and notify the microcontroller that an object is in the mower's path. **Verification:** sensors will be tested and verified that they are in working order.	Gabriel

Blade Motor	775 Motor	Must be able to spin cutter blades at 2000 rpm minimum. Must be durable and resilient enough to take impact. Must be able to run on a 12V DC source.	**Justification:** Most commercial lawn mower blades spin at 3000 rpm, but grass can still be cut with a 2000 rpm motor. This will allow for better power efficiency and still allow the blades to cut through most grasses and weeds. **Verification:** Upon startup, the mower will turn on the cutter mower while docked and verify current drain is within specifications without a load.	Brian

GPS	Ardusimple simpleRTK2B	Must have RTK Base station and Rover communication capability. Must be able to communicate with multiple satellites and provide real time corrections for accurate pathing. Must have at least 3cm of positional accuracy.	**Justification:** Since the mower will be GPS-driven, accurate positioning will be required to be at least 5% of the overall cutting area. **Verification:** Upon startup, the mower will calculate and send real-time adjustments between the mower GPS unit, Docking Station GPS unit, and satellites.	Group

Docking Station

Part	Specific component	Engineering Specification	Justification & Verification	Responsibility
Solar Panel Kit	12V 10W Polycrystalline	Must be able to charge batteries while the system is running. Must be able to charge a 12V battery source.	**Justification:** The docking station will have vital electronics necessary to provide real time corrections for GPS accuracy. **Verification:** The charging system will have a voltage controller that will display the supplied voltage and voltage of the battery.	Brian

Inductive Charge System	12V inductive charge coils	Must be able to transmit voltage and current without a physical connection.	**Justification:** System is to help increase battery recharge rate. **Verification:** Tests will be done to verify power transfer.	Brian
GPS	Ardusimple simpleRTK2B	Must have RTK Base station and Rover communication capability Must be able to communicate with multiple satellites and proved real time corrections for accurate pathing Must have at least 3cm of positional accuracy.	**Justification:** Since the mower will be GPS-driven, accurate positioning will be required to be at least 5% of the overall cutting area. **Verification:** Upon startup, the mower will calculate and send real-time adjustments between the mower GPS unit, Docking Station GPS unit, and satellites.	Brian
Controller				
Part	**Specific component**	**Engineering Specification**	**Justification & Verification**	**Responsibility**
Microcontroller	Raspberry Pi 3B+	Must be able to communicate with the microcontroller on the mower for programming and directional controls Must be able to provide real time controls for manual movement.	**Justification:** The controller will allow the user to program and give manual controls. **Verification:** When the mower is set to program mode, the mower and controller will connect.	Gabriel

1.7 Organization of the Report

The remaining portions of the report will be separated into the following 4 chapters outlined in table 1.3 below:

Table 1.3 Report Organization

Chapter 2 - Background Research	● Background research performed ● Explanation of components used ● Programming language used ● Power budget
Chapter 3 - Project Contribution	● Design Integration and Implementation ● Troubleshooting ● Testing Results ● Code Explanation
Chapter 4 - Non-Technical Issues	● Budget and Timeline ● Environmental, health, safety, ethical, and social aspects ● Sustainability
Chapter 5 - Conclusion	● Summary of project and results ● Future recommendations

Chapter 2

Background Research

Summary

In this chapter, we go over research conducted on devices and systems, the hardware used, programming language, and the power budget of the systems.

2.1 Design Research

In order to determine the best possible components to accommodate for size, power, and function, we needed to understand the specifications of key components and how they can differ from the options available to provide the best results for the project.

2.1.1 Lead Acid Batteries vs Lithium Iron Phosphate Batteries

Due to weight restrictions, power consumption of the motors, and the need for faster charging times of the mower, we needed to use a higher quality battery than the standard Sealed Lead Acid (SLA) batteries. This led us to choose Lithium Iron Phosphate (LiFePo4) batteries as our ideal mower battery for longer run time, faster recharge times, and overall weight restrictions.

Due to overall costs, the docking station will be a standard SLA battery, but even then the system would benefit from also using a LiFePo4 battery as well. The key differences between SLA and LiFePo4 batteries that we had to consider for this project are shown below in Table 2.1.

Table 2.1 SLA Batteries vs LiFePo4 Batteries

SLA Batteries vs LiFePo4 Batteries		
	SLA	**LiFePo4**
Weight	~ 0.5 lbs per Ah	~0.27 lbs per Ah
Max Voltage	12.6V	13.4V
Full Charge Time	10 Hours	4 Hours
Recharge Cycles	500	2000 - 4000
Cost	~$2.6 per Ah	~$5.5 per Ah +

Not only is the charge time for both batteries different, but the charging profile is also different as they both require different voltages and currents at the different stages of the charge cycle. Since the batteries are different, they require different chargers or a charger that can handle multiple types of batteries. The solar charge controllers picked for this project are designed to handle both SLA batteries (setting B01) and LiFePo4 batteries (setting B03).

As shown below in figure 2.1, the charging profile of a LiFePo4 battery starts off with high current and low voltage and will continue until 60% to 80% battery charge before it starts to drop off and the battery starts to only absorb voltage until it is at 100% charge. The charging current we used is shown in the formula below:

ƚƀ ϵ ψ∅ǿ ყ ℏ⅃

This shows that the charging current, C, is about 30% of the total current rating of the battery, so the 6Ah battery used for this mower is C = 1.8A, and the maximum charging current applied to the battery is 720mA.

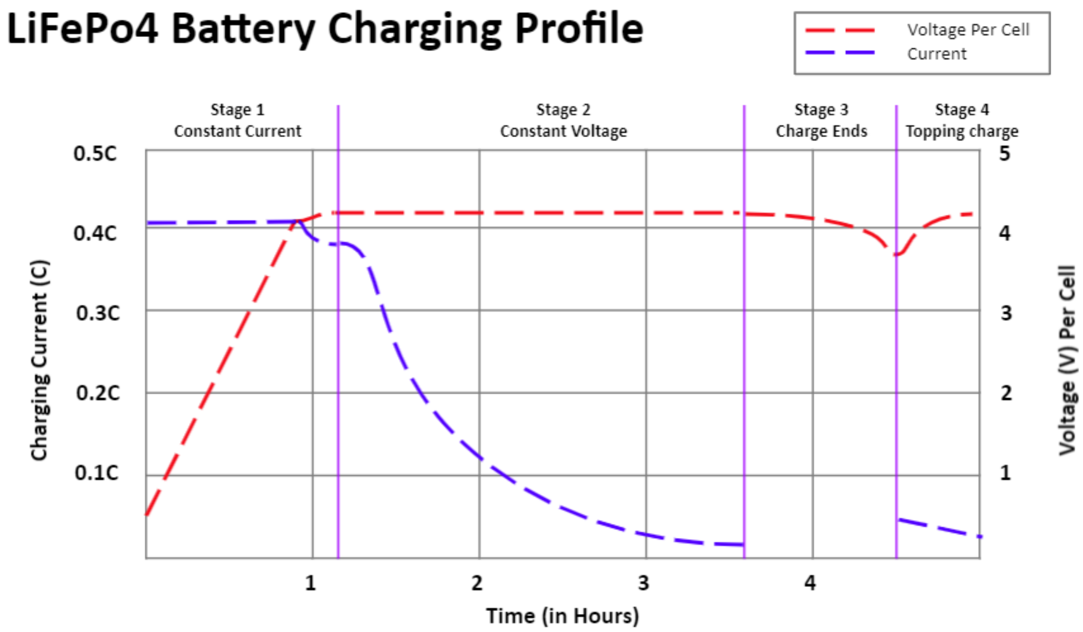

Figure 2.1 LiFePo4 Battery Charging Profile

SLA batteries will typically start off with both the charging current and voltage at higher levels and have the charging current start to fall off once the voltage limit is met. This is illustrated in figure 2.2 below.

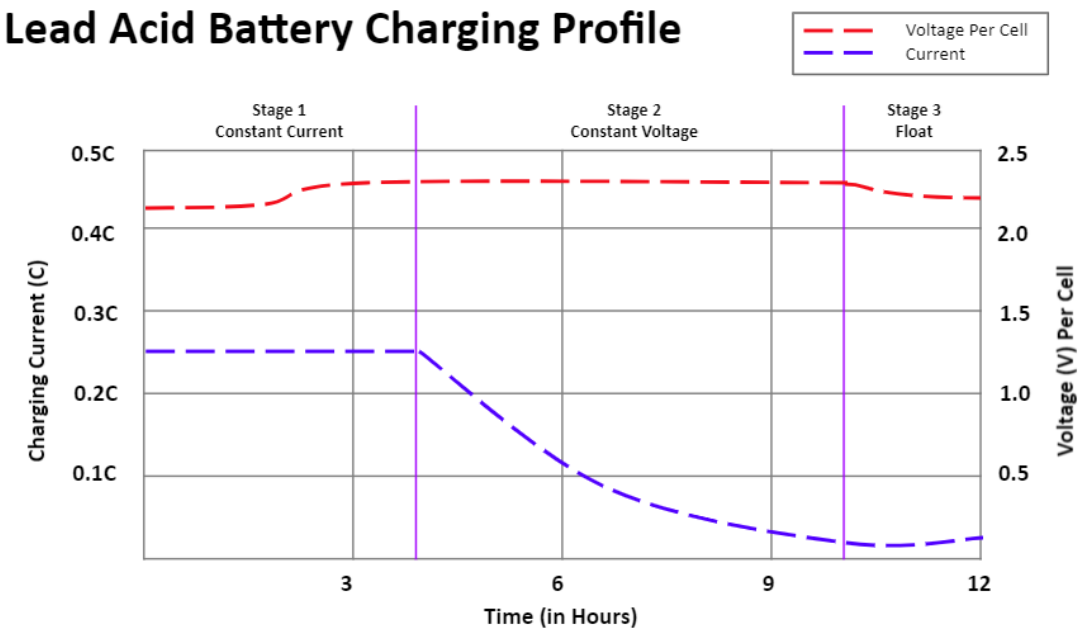

Figure 2.2 SLA Battery Charging Profile

2.1.2 Monocrystalline vs Polycrystalline Solar Panels

Due to the overall size and weight restrictions of the mower, we had to find an acceptable solar panel that had to be within the weight restrictions and overall size dimensions. This required the use of the low-profile flexible monocrystalline solar panels on the mower but still, let us use the more efficient polycrystalline solar panel on the docking station.

Both types of solar panels we used have some advantages and disadvantages over the other in one way or another which are outlined below in Table 2.2.

Table 2.2 Polycrystalline vs Flexible Monocrystalline Solar Panels

Polycrystalline vs Flexible Monocrystalline Solar Panels		
	Polycrystalline	**Flexible Monocrystalline**
Weight	2.55 lbs	0.56 lbs
Size	13.75” x 10” x 0.75”	9.1” x 6.7” x 0.5”
Efficiency	16% - 20%	21% - 24%
Durability	Glass Cover and Metal Frame	No Protection
Cost (10W 12V)	$25	$32

2.1.3 Cutter Blade RPMs

In order for the cutter blade on any lawn mower to cut grass, it must be able to achieve a minimum rotational speed to not only cut through the grass but to continue cutting even while the blade is slowing down. This led us to research the approximate RPMs of both lawn mowers and weed eaters to determine the minimum acceptable cutter blade speed for the system.

In our research, we found that most lawn mowers have an average cutter blade rotational speed of about 3500 RPMs and can still operate at 2000 RPMs when bogged down. Since we will have a much smaller cutting blade, we researched how fast the typical weed eater rotational speed is and found them to average at about 7000 RPMs, with some as low as 3000 RPMs or as high as 13,000 RPMs. Since these systems typically operate with a cord instead of a metal cutting blade, we decided that a minimum rotational speed of 5000 RPMs was an acceptable rotational speed for our system to cut grass.

2.1.4 GPS with Real Time Kinematics

The heart of the pathing system will revolve around the GPS and ordinary GPS modules will typically only have a positional accuracy between 1.5m and 3m, which is well beyond the threshold of accuracy needed for this product. In order to increase the positioning accuracy within the threshold needed by this system, we needed to use a GPS module with Real Time Kinematics (RTK) capability [5]. This will allow our mower to achieve between 1cm and 3cm of positional accuracy, which is exactly what this product will require.

How an RTK capable GPS module works is that the GPS module being tracked will not only communicate with the satellites in the sky but an additional transponder station, either a GNSS tower or an additional RTK capable GPS unit that is acting as a broadcasting tower will work. The stationary tower that is broadcasting will provide real-time corrections based upon its stationary location to the mobile GPS unit, allowing for centimeter-level accuracy.

For this product, we will be using a secondary RTK capable GPS unit housed in the docking station to allow the product to be used independently of any other communication towers or services.

2.1.5 Raspberry Pi Power Conservation and Real Time Clock

We plan on using the Raspberry Pi 3B+ as it was an effective microprocessor for the application of the project. Since we will not be doing high-level computing, machine learning, or video interface, the Raspberry Pi 3B+ offers a cost-effective option for the scope of the project.

Since the Raspberry Pi units are designed cost-effectively with a small footprint, the designers removed some basic functionalities that you would normally see on a typical computer like a real-time clock (RTC) and a low-power sleep mode. Both of these functions are mandatory for a limited power system and creating a cutting schedule. To resolve this, we needed to find a way to integrate an add-on board to the Raspberry Pi that can control the power to the Raspberry Pi and automatically schedule the unit to power on and off at certain times of the day and/or week.

We found two potential modules that fit our Raspberry Pi 3B+, the Sleepy Pi and the Witty Pi 3. Both modules offer a true sleep mode function that will allow the module to both turn on and off the Raspberry Pi through a scheduling script.

This is extremely helpful for conserving battery power over long-term use as the Raspberry Pi 3B+ uses between 250 mA to 350 mA while just powered on and not running any programs or peripherals. When the Raspberry Pi is turned off, it can still consume up to 100 mA of power. The Witty Pi 3 utilizes its RTC to schedule power flow to the Raspberry Pi at certain times and can automatically power the system on or safely shut down the Raspberry Pi when either the schedule times are met or battery power falls too low and recharges to the correct voltage.

Powering the Raspberry Pi through the Witty Pi 3 is easily done either through the 5V USB type C connector or the 6V-24V XH2.54 connector with its onboard voltage regulator. The Witty Pi consumes very little power and while in standby state between Raspberry Pi operation consumes less than 1 mA of power.

2.1.6 Compass Interference

Compasses are made to pick up the earth's natural magnetic fields, which allows them to determine what direction they are pointing. However, magnetic interference either by

other magnets, coils of wire, and even connectors that can be magnetized will cause inaccurate and varying compass results.

While designing the mower layout, magnetic devices like motors and charging coils had to be accounted for. Not only that, magnets that were not being used in the GPS's GNSS antenna had to be removed from the system to prevent interference. Finally, we had to make sure any connectors and pins leading up to the compass were non-ferrous as they would also interfere with the compass readings.

To resolve the issues of interference, the compass was raised far above the mower itself and is far enough from any magnetic interference produced by the cutter blade and drive motors to provide accurate readings.

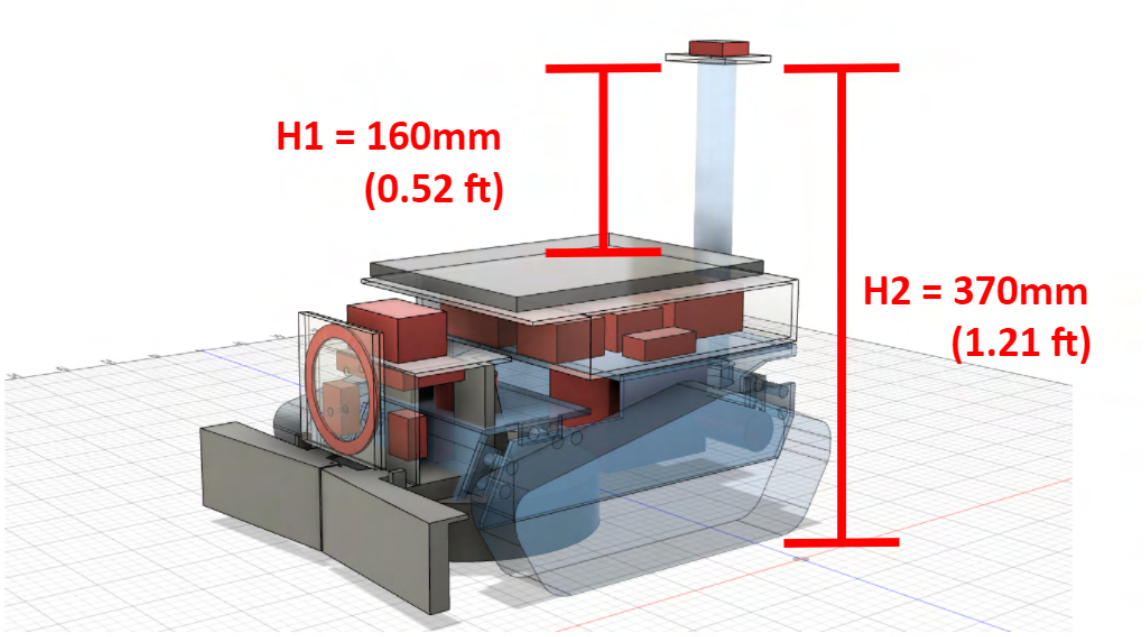

Figure 2.3 Compass Height

2.2 Mower Hardware

Due to the high costs of several key components, like the RTK system, we needed to scale the mower unit down to cut costs but yet provide a functional proof of concept system. While doing so, we needed to keep in mind the overall weight and size of the available components to ensure the system is still able to move and doesn't have components hanging off at odd angles.

The bottom layer will house the drive motors, drive tracks, cutter blade, blade shroud, and power source as shown in the CAD drawing made with Fusion 360 in Figure 2.4 below.

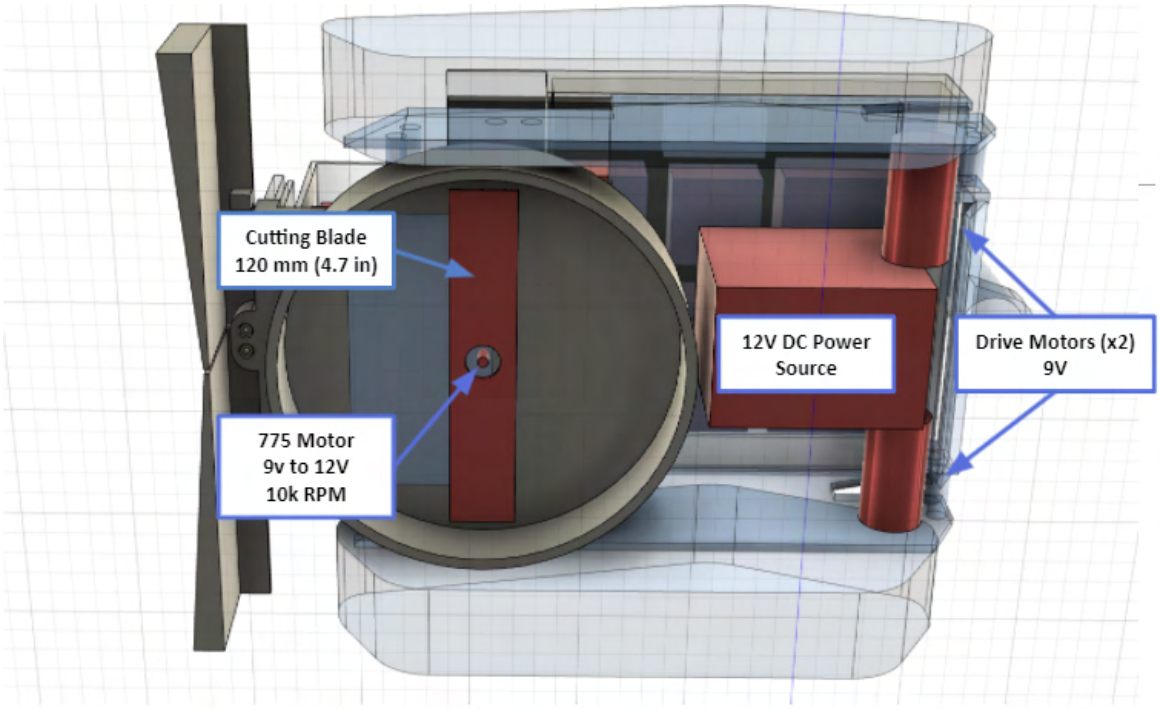

Figure 2.4 Mower Chassis Assembly Bottom

The middle layer will house the majority of the electronic devices within a clear plexiglass box. In this section, the microcontroller, voltage regulators, RTK system, motor controls, and solar charging regulator will be housed as shown in Figure 2.5 below.

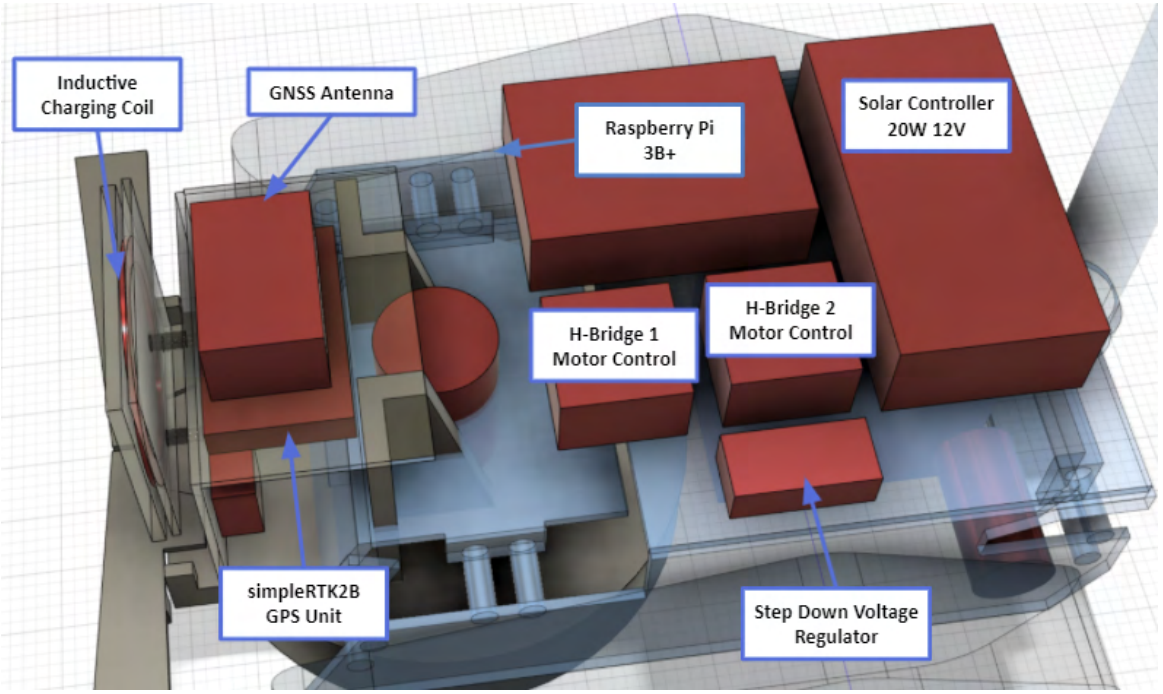

Figure 2.5 Mower Chassis Assembly Middle

The top layer will house the compass and solar panel as shown in Figure 2.6 below.

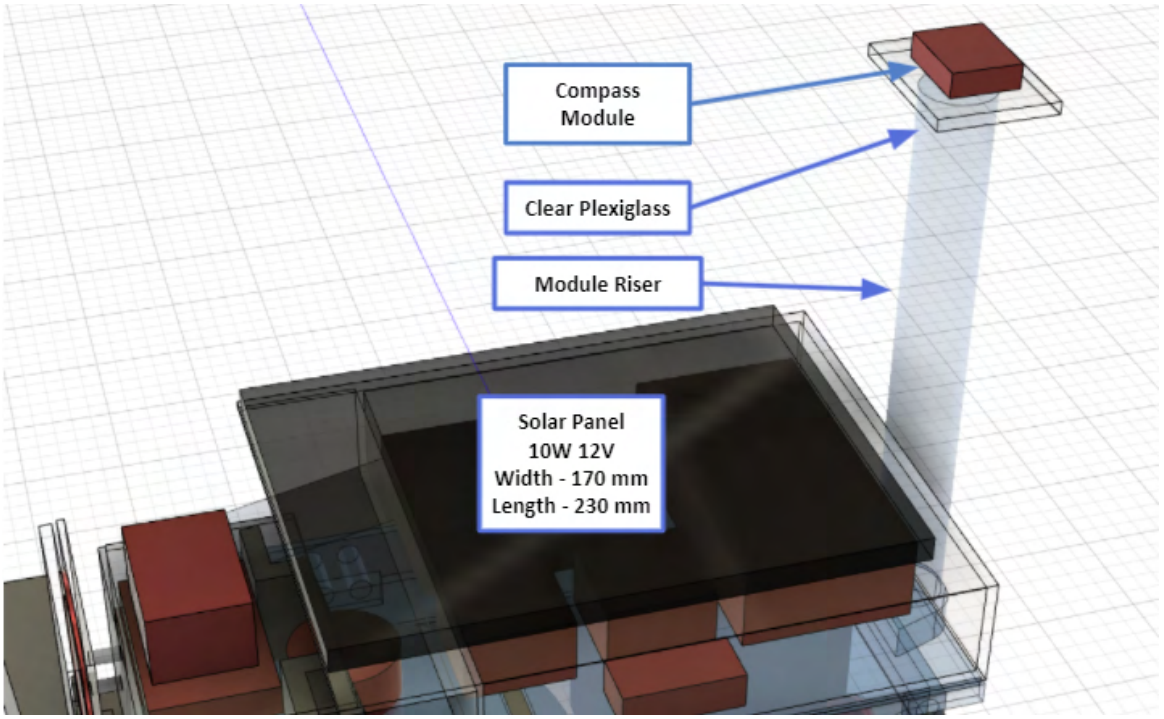

Figure 2.6 Mower Chassis Assembly Top

2.2.1 Mower Chassis with Expansion Kit

As we anticipated the overall mower design to be small in comparison to that of a regular lawn mower, we determined that in order to give the system every possible advantage to move in the grass, it needs to have tank treads to create more traction for movement. This leads us to choose the XiaoR Geek Smart Robot Aluminum Tank Chassis as our frame to hold all of our devices. This was chosen because it had enough surface area to attach all of our components, an all-aluminum frame for durability, the required tank treads we were looking for, 12V brushed DC motors, and had a maximum weight limit of about 15 lbs.

This kit comes with two GM25-270 brushed 12V DC motors that spin at 350 RPMs. These motors are capable of easily moving up to 15 lbs and have a max current draw of 1.2A

The only downside to the system was the available space between the two tracks was less than 5” across, which would severely limit the available cutting space to the mower. This was resolved by making an expansion kit with standoffs, screws, and a new top plate, resulting in an additional 2 inches of cutting space.

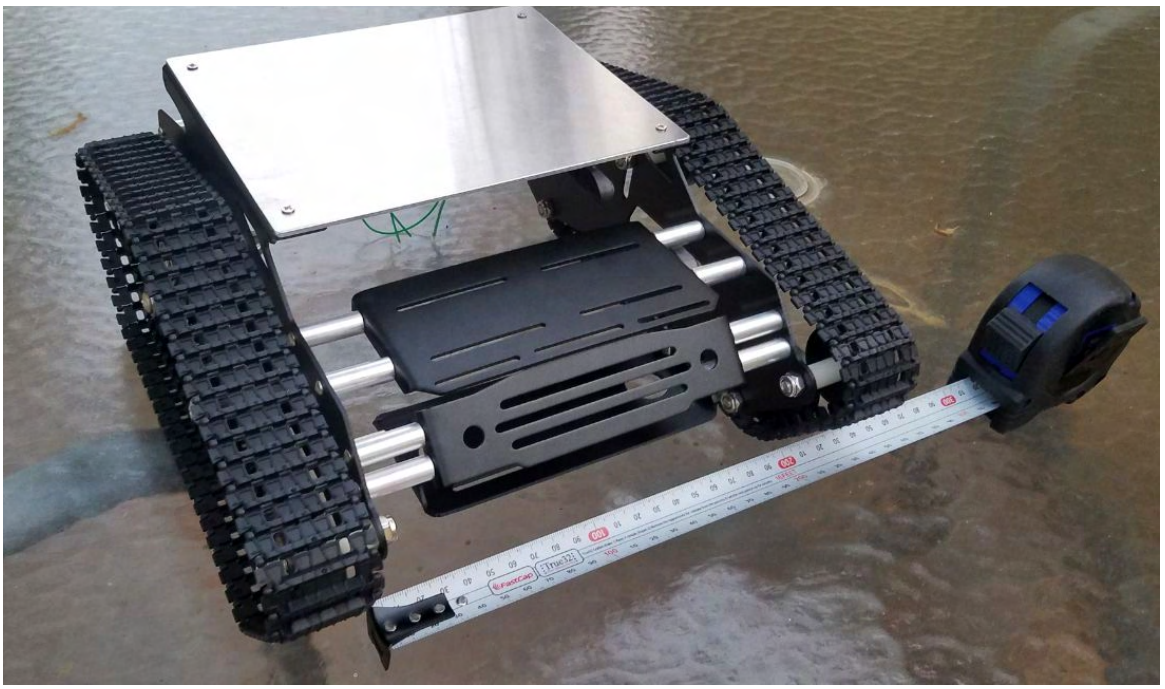

Figure 2.7 Tank Chassis with Expansion Kit

2.2.2 Raspberry Pi 3B+ and Whitty Pi 3

The microprocessor that will be controlling the entire mower system will be the Raspberry Pi 3B+ as it has the adequate processing capability to handle all onboard components, uses less power to function than the updated Raspberry Pi 4 variant, is capable of handling multiple inputs and outputs with its 40 pin header, and has a small overall form-factor that will fit the size profile of the system.

In order to provide the Raspberry Pi with the extra power consumption regulation and RTK functionality the system will require, it will have a Witty Pi 3 module installed on top, connected by the 40 pin header. This will allow us to plug the power straight from the 12V power source into the XH2.54 power connector that will regulate the voltage down to 5V input and provide up to 2A of current to the Raspberry Pi and its peripherals.

The Witty Pi 3 uses GPIO2, GPIO3, GPIO4, and GPIO17 on the Raspberry Pi to communicate between the systems and allows the Witty Pi 3 to initiate power down and monitoring commands. The Witty Pi 3 also uses GPIO14 which is the TXD pin, but only uses it to monitor voltage levels.

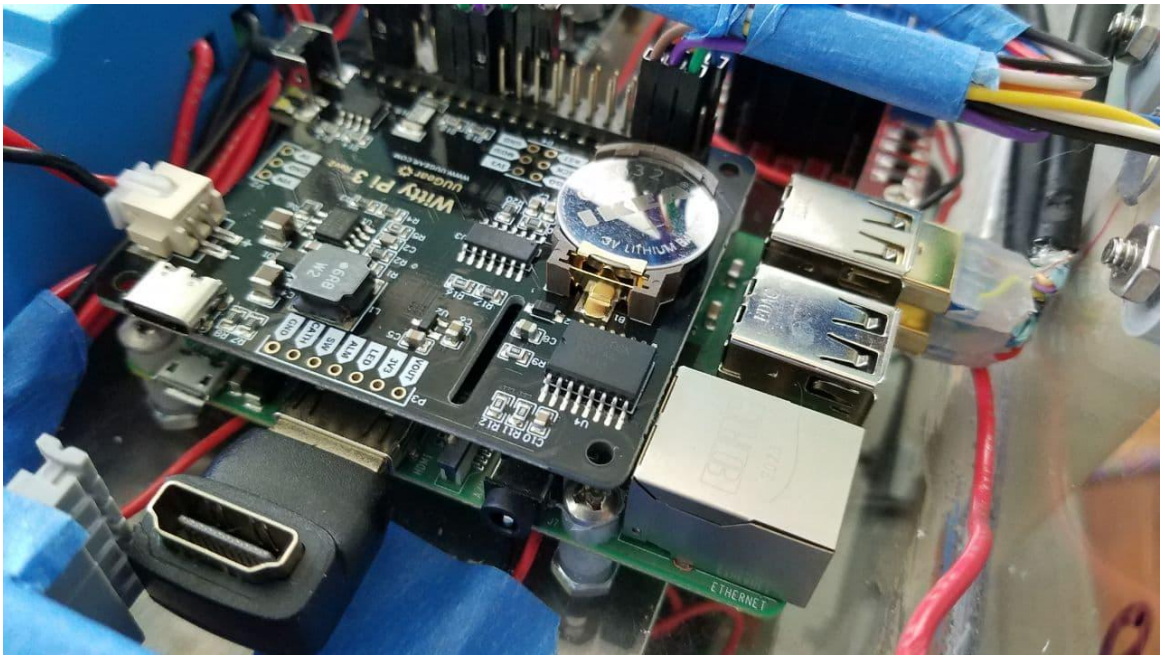

Figure 2.8 Raspberry Pi 3B+ and Witty Pi 3

2.2.3 775 Motor with Cutter Blade

In order to properly cut grass, the system will need a motor that is capable of at least 5000 RPMs, is durable, and can operate on a 12V system. This resulted in us picking a 775 12V DC motor that has a maximum rotational speed of 10,000 RPMs. This motor has ball bearings to help support the drive shaft on impact and has a maximum current draw of 10A under load.

During initial testing, we found that the motor will rotate at about 6,300 RPMs at 7.5V input and will have maximum spikes up to 7.5A under near stall conditions if it hits a stick and spikes to about 5A while cutting grass.

The motor was purchased with a cutting blade and attachment to securely mount to the motor. The Blade is made of stainless steel and is held in place with the motor adapter.

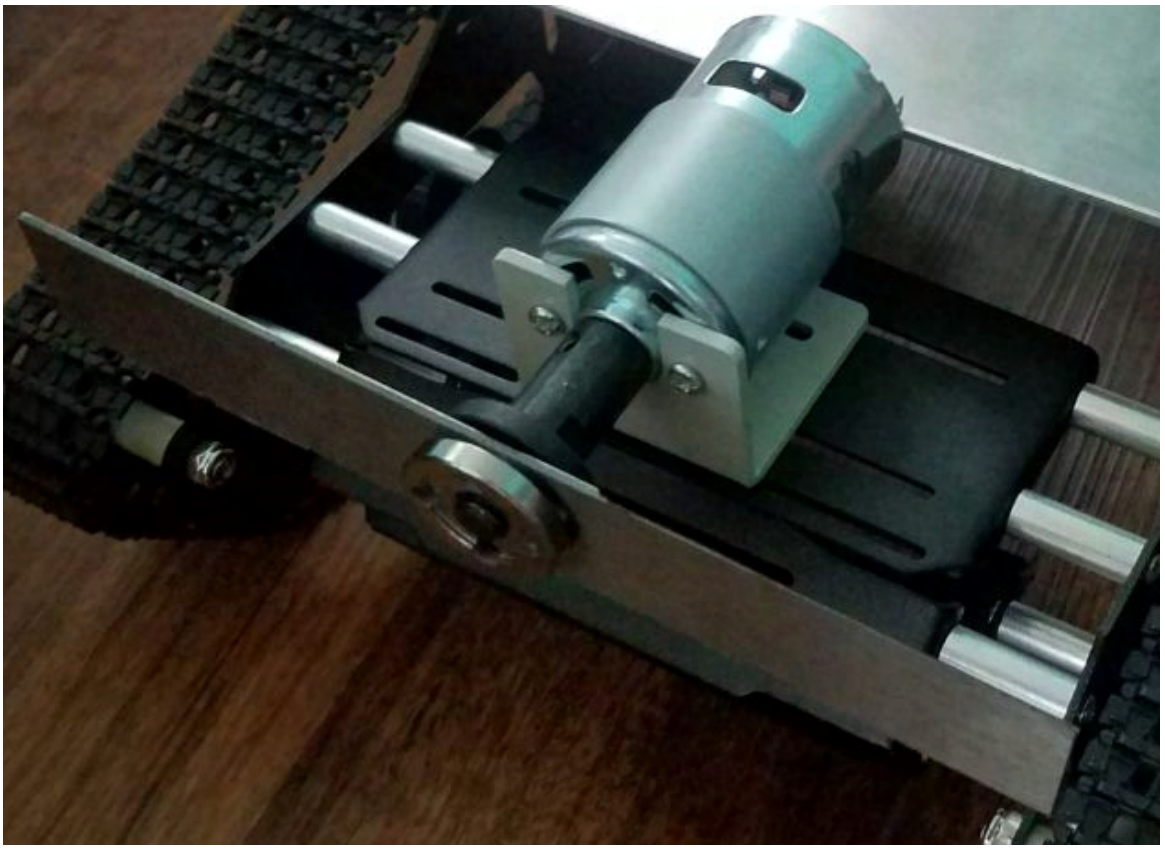

Figure 2.9 775 Cutter Blade Motor and Cutter Blade (not installed)

2.2.4 L298N Motor Drive Controller

The L298N dual h-bridge motor controller is designed to provide both forward and reverse movement operation for two motors on the same board. This module can operate on a 5V to 35V input and has its own 5V voltage regulator on board to handle the logic circuits that control power to the motors. The signals to move the mower forwards and backward are using Pulse Width Modulation (PWM) to control the maximum output voltage levels, which provides some speed controls. The L298N can handle up to 2A on each motor, which is enough power to control the GM25-270 motors which have a max load current of 1.2A.

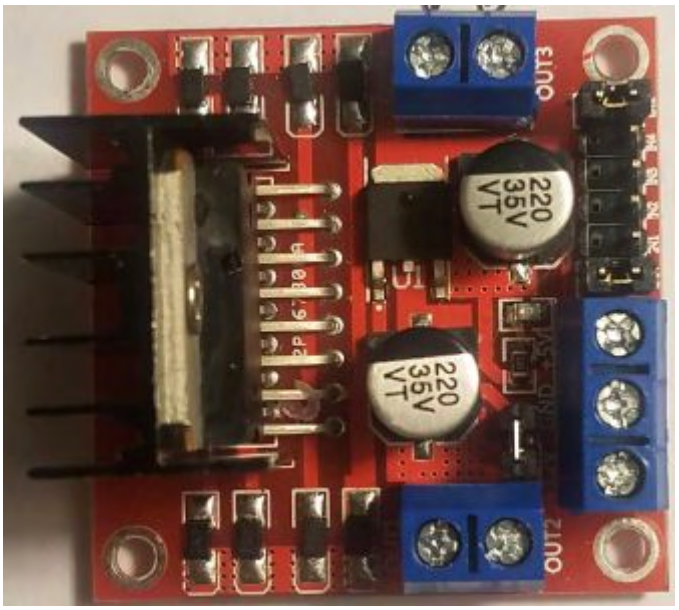

Figure 2.10 L298N Motor Controller

2.2.5 VNH5019 Motor Drive Controller

In order to drive the 775 cutter blade motor, we needed a controller that can handle up to 10A spikes on motor startup and if a stick slows down the cutting blade while running. The VNH5019 motor controller can handle a constant load of 12A and can handle voltages between 5.5V to 24V. Since the module does not have its own 5V regulator to power the logic circuits that turns the motors on and off, we pulled the power from the Raspberry Pi. This is a negligible current draw and has little to no impact on the Raspberry Pi.

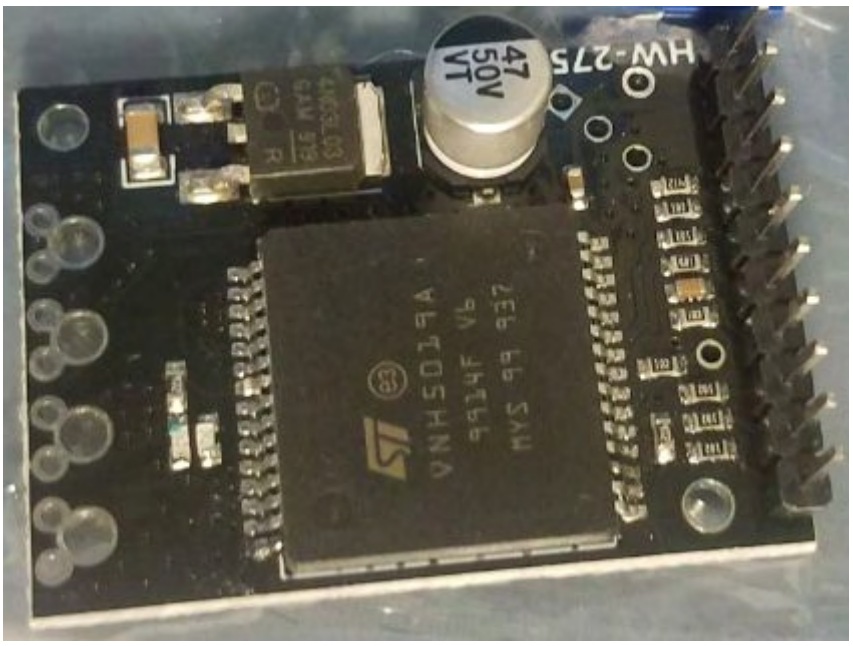

Figure 2.11 VNH5019 Motor Controller

2.2.6 DC to DC Step Down Voltage Regulator

To regulate the cutter blade motor speed, we will be using a DC to DC voltage regulator to drop the voltage to the cutter blade from 12V to 7.5V and can handle up to 10A. This will allow for a slower rotational speed and reduce the chance of the cutter blade vibration shaking the system apart.

Figure 2.12 DC to DC Step Down Voltage Regulator

2.2.7 Ardusimple simpeRTK2B (Rover)

The simpleRTK2B is a GPS system that uses RTK technology to provide centimeter GPS accuracy. It achieves this by having one RTK referred to as Rover on the mower and another RTK referred to as Base in a stationary position on the docking station. Once the Base RTK has a GPS lock it uses it to send correction data to the Rover via the XBee radio modules.

Figure 2.13 SimpleRTK2B

RTK technology has been available for several years but has just recently been integrated into an affordable module that is within the price range for average consumers. The simpleRTK2B uses the U-Blox ZED-F9P module and is fully compatible with the Raspberry Pi.

In order to connect to satellites, the module uses a GNSS antenna that is to be mounted up high and ideally with direct access to the open sky for best results.

2.2.8 12V 6Ah Lithium Iron Phosphate Rechargeable Battery

In order to power the mower and stay within weight parameters set by the chassis, we needed a lightweight power supply that still has enough current to power our system. This led us to pick a Lithium Iron Phosphate (LiFePo4) as our primary power supply.

Due to the overall size restrictions of the mower and the attempt to stay within budget, we opted to pick a 6Ah battery instead of a larger variant. This battery will

provide enough power to operate the system for the required amount of time between charges, but does have a limit to 6A continuous output current with allowable spikes to 20A. During the time of purchase, the next sizes up from 6Ah were 10Ah and 12Ah, but the overall size profile, weight, and price were too large to fit our chassis and in excess of $100 each. These would provide ideal output current levels for the system, but the mower will still be able to operate normally with the 6Ah battery.

Figure 2.14 LiFePo4 Battery

2.2.9 Solar Charge Controller

The solar charge controller is what will direct power from the solar panel to the battery at the correct voltage and current requirements needed for a LiFePo4 battery for each phase of its charge cycle.

Since the charge controller will be handling the power to the system, it will need to be able to handle loads of all the components running at once. The primary components that draw the most power will be the cutter blade motor, drive motors, Raspberry Pi 3B+, and simpleRTK2B devices, which when all ran at the same time at maximum power draw can cause current spikes up to 15A if all motors stall and the

Raspberry Pi is pulling maximum power. This required us to need a charge controller that can handle up to 20A to safely supply power to everything at once.

Figure 2.15 Solar Charge Controller

2.2.10 Flexible Monocrystalline Solar Panel

To provide the mower with the ability to recharge its battery with solar energy, we needed a solar panel that both fit the size and weight restrictions set forth by the mower chassis itself. This required us to locate a small, lightweight solar panel that can still provide 12V and 10W.

After searching through several dozen solar panels, we located a Flexible Monocrystalline 12V 10W solar panel. This solar panel is 9.1” x 6.7”, weighs less than 9oz, and has a claimed conversion efficiency of 21% to 24%. This module fits all parameters for the system requirements.

Figure 2.16 Flexible Monocrystalline Solar Panel

2.2.11 HMC5883 Compass

Since the mower needs directional control to assist the positional control of the GPS unit, we needed to include a compass module that is compatible with the Raspberry Pi. For this, we chose the HMC5883 Compass that was paired with a BN-880 GPS module. This was done purely for the non-ferrous connector system of the module instead of a standard header pin that is known to interfere with the compass. Since we didn't need the GPS module, we simply left its connections disconnected from the Raspberry Pi and only used the compass circuitry.

Figure 2.17 Compass Module

2.2.12 Rocker Power Switches

For safety and functionality reasons, we needed to be able to have two power switches, one to handle power to the entire system and a manual on/off switch for the cutter blade to render it immobile for safe handling and testing. These power switches also had to be robust enough to handle vibrations and potential impact as well as not be a choke point for current. We decided to use a 12V rocker switch that can handle up to 20A to prevent choking of current spikes over 15A that the system could theoretically produce in specific situations.

Figure 2.18 NTE 54-645-B 12V 20A Rocker Switch

2.2.13 Momentary Sensor Switches

Since the mower will be autonomous, it will need a simple way to detect objects in its path. Ideally, you would want some sort of optical sensor in front of the mower to detect objects in its path, but these sensors could easily be obstructed or confused by tall grass. This resulted in us using momentary switches that are activated by a bumper. For this part, we chose a momentary hinge micro switch with a roller lever for sensitivity and to encourage a free range of motion.

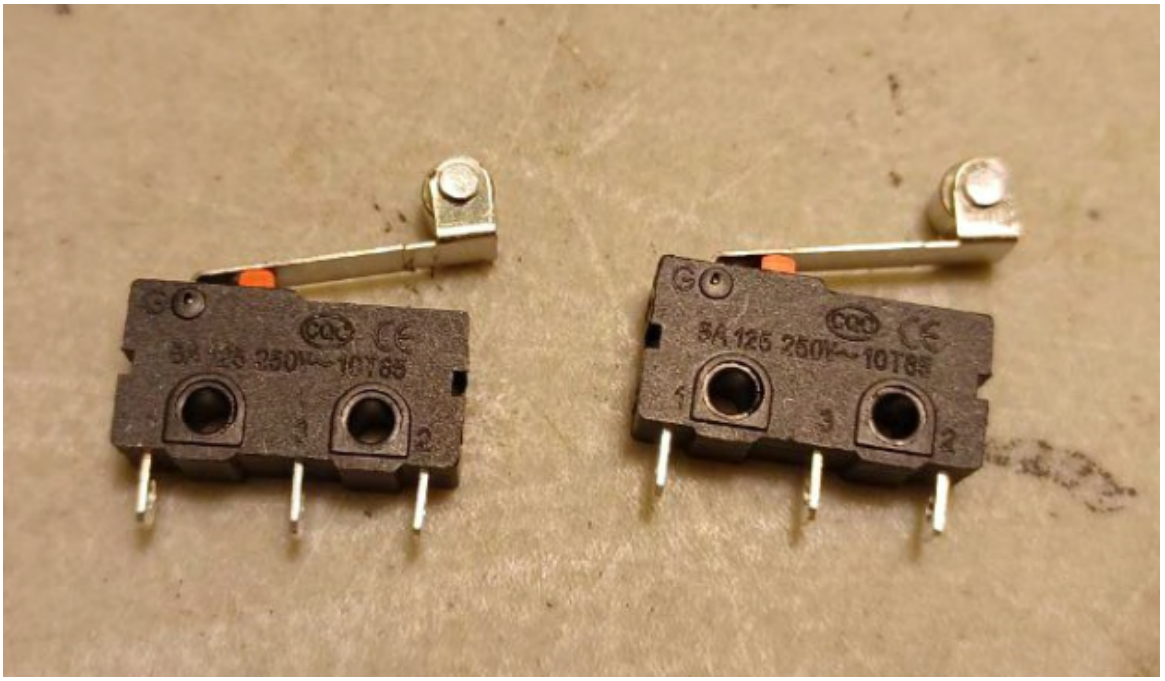

Figure 2.19 Momentary Switch

2.3 Docking Station Hardware

The docking station will be the non-mobile unit that the mower will return to when it is done cutting grass as well as the unit that will be sending RTK corrections to the mower for precise positioning.

2.3.1 Ardusimple simpleRTK2B (Base)

Just like the mower, the docking station will have its own simpleRTK2B module, RF transmitter, and GNSS antenna but will be programmed as the Base unit that is designed to not move and is the key system that will allow for precise positioning. The module is exactly the same as the Rover unit, but will just have different programming.

2.3.2 12V 9Ah Sealed Lead Acid Battery

The primary source of power to the docking station will be a 12V 9Ah SLA Battery. We chose this battery over another LiFePo4 battery purely because this docking station will not have a large continuous power draw on it and is about half the cost. We decided to use a larger 9Ah battery as there are plans on integrating an inductive charge booster to the mower to help keep the mower's battery at a full charge in non-ideal conditions.

Figure 2.20 12V 9Ah SLA Battery

2.3.3 Solar Charge Controller

With any solar charge system, it is ideal to have a power control system that both the input power from the solar panels to the batteries and from the batteries to the load. Since the docking station is relatively low power, a cheaper charge controller was chosen to handle the system.

Since the docking station will also have an inductive charge system in addition to the GPS module, we decided to include a second charge controller with the sole purpose of regulating the voltage from the battery to the mower. This was done because the GPS module needed a power controller to allow it to stay active continuously but wanted the inductive charge system to cut off when voltage drops below 12V. Using the second charge controller was a cheap and simple solution to regulate the power to two different systems with different power needs.

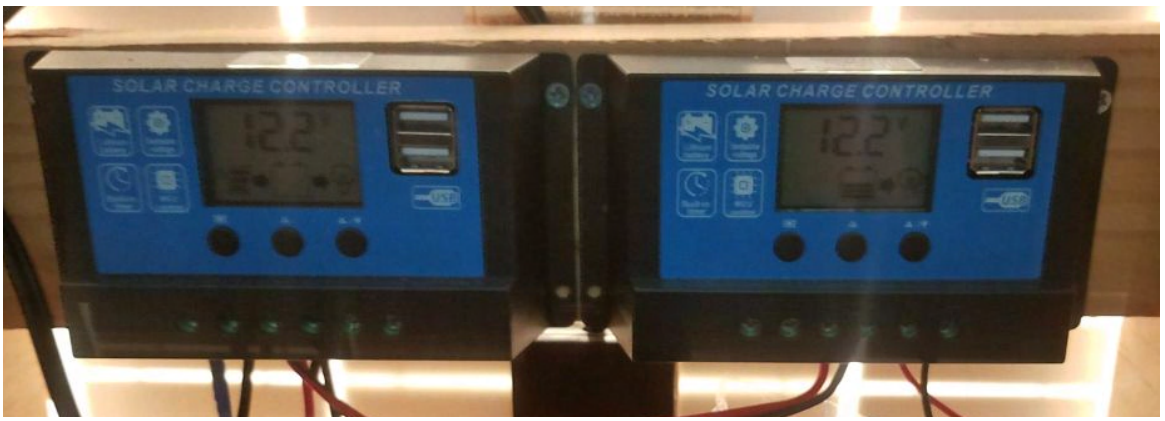

Figure 2.21 12V 10A PWM Solar Charge Controller

2.3.4 Polycrystalline Solar Panel

The weight and size requirements for the docking station were not critical, unlike the mower, which allowed for a wider selection of solar panels. This led us to pick an affordable name-brand solar panel system that was more durable and weatherproof but was larger and heavier than the mower's chosen solar panel. After some research, the Mighty Max 12V 10W Polycrystalline solar panel was chosen as the docking station's solar panel for both cost and matching name brand to the battery.

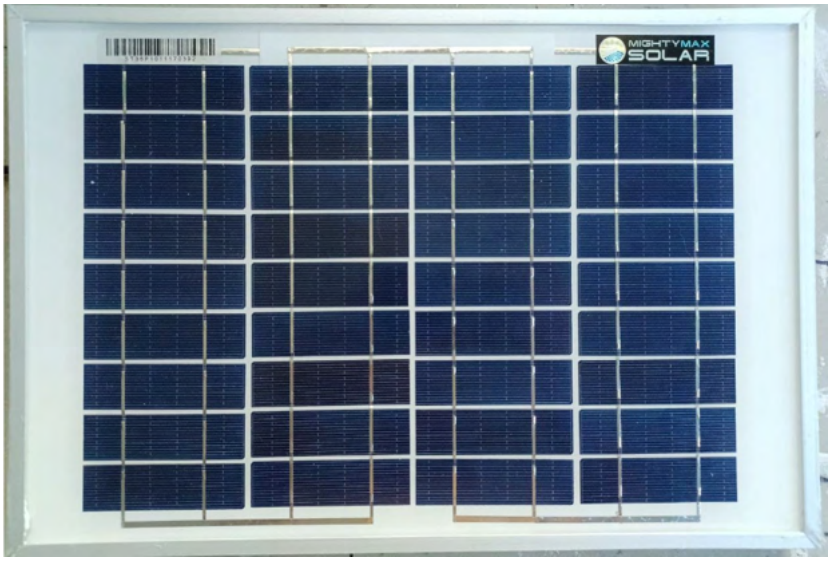

Figure 2.22 12V 10A Polycrystalline Solar Panel

2.3.5 Inductive Charge System

After some extensive research and testing of inductive charge systems that could work with our system, we ended up using a simple 12V 1A inductive charger coil system by Taidacent. This will allow us to transmit current from the docking station and mower to charge the mower's battery without ever having to make physical contact with each other.

There were limitations to the design of the docking station, power transmitted, and physical distance of transmission due to the original 12V power supply choice. Since this was an optional addition to the system that wasn't originally in the proposal phase, we couldn't go back and change the docking station to a 24V system for ideal results, which is covered in chapter 5.2.3.

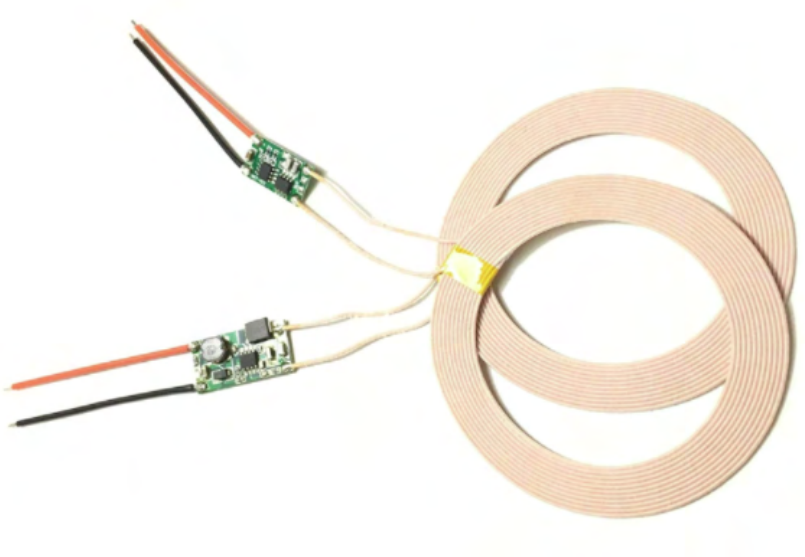

Figure 2.23 Taidacent 12V Wireless Charging Module

2.4 Controller Hardware

In order to communicate with the mower and provide manual controls by the user, the system will need a control device. This system will be able to communicate wirelessly to the mower and provide added functionality and programming to the system.

2.4.1 Raspberry Pi 3B+

The heart of the controller will revolve around the Raspberry Pi 3B+ as it is a simple microprocessor device with built-in wireless connection and added functionality. This

unit will have programming that will allow for direct connection to the mower if it is in range.

2.4.2 7” Touchscreen Kit

The best and most intuitive way that was found to allow the user to have both input and output functionality was to use a simple 7” touchscreen display and case. This provides the user an all-in-one module to control the system.

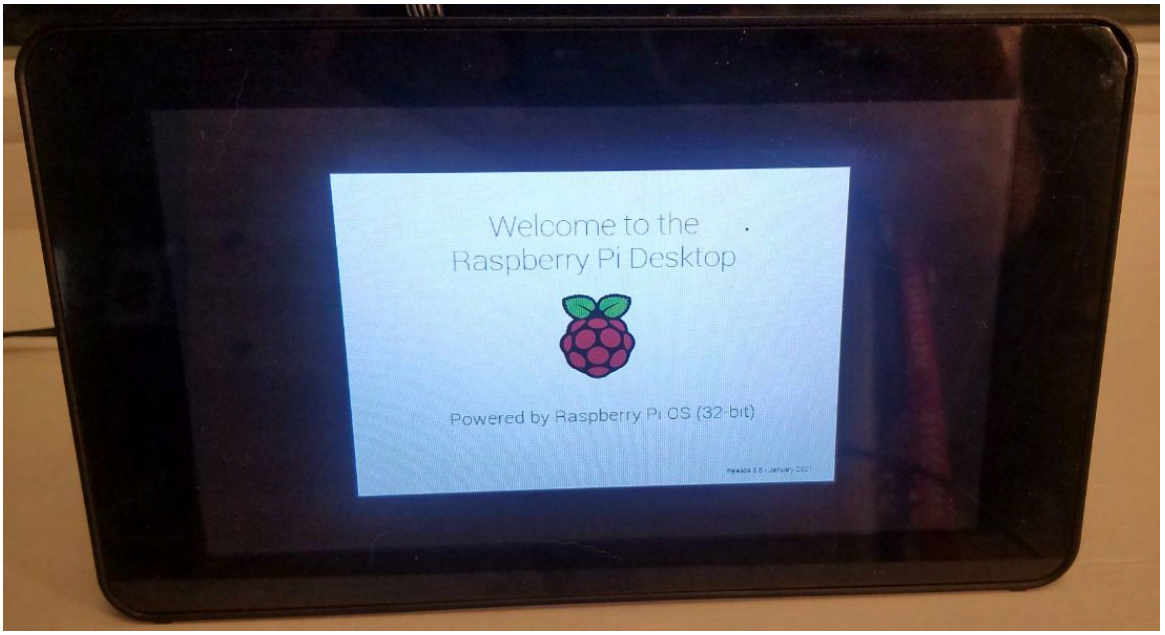

Figure 2.24 Raspberry Pi 7” Touchscreen Kit

2.5 Programming Language

The Raspberry Pi is capable of using many different coding languages, but the language the team is used to using on these systems is Python as it provides the team with an easy-to-understand code with plenty of functions and is compatible with all components communicating with the Raspberry Pi.

2.6 Power Budget

The power budget is used to show how much power is being consumed for each device and allows us to verify that there is enough power supplied to run the devices. Below are the individual power budgets for the mower and docking station.

Table 2.3 Mower Power Budget

Mower						
Part	**Description**	**Qty**	**Current (A)**	**Voltage (V)**	**Power (W)**	**Extended Power (W)**
Control Module	Raspberry Pi	1	3.00	5.10	15.30	15.30
Power Controller	Witty Pi 3	1	1E-03	5V	0.005	0.005
Drive Motor	GM25-270 DC Motor	2	1.20	12	14.4	28.8
Cutter Motor	775 Motor	1	10.00	12.00	120.00	120.00
Radio Antenna	XBee X2C	1	4.00E-02	3.30	0.13	0.13
GPS Module	simpleRTK2B	1	1.80E-01	3.30	0.60	0.60
Charge Controller	20A Solar Charge Controller	1	0.015	12	0.18	0.18
				Total Power (W)		157.64

Table 2.4 Docking Station Power Budget

Docking Station						
Part	**Description**	**Qty**	**Current (A)**	**Voltage (V)**	**Power (W)**	**Extended Power (W)**
Control Module	Raspberry pi	1	3.00	5.10	15.30	15.30
Radio Antenna	XBee X2C	1	4.00E-02	3.30	0.13	0.13
GPS Module	simpleRTK2B	1	1.80E-01	3.30	0.60	0.60
Inductive Charger	Inductive Charge Coils	1	1.00	12.00	12.00	12.00
				Total Power (W)		28.03

Chapter 3

Project Contribution

Summary

This chapter goes over the personal contribution to the design of the project. This includes the design and implementation of the mower, docking station and controller as well as the software, troubleshooting, and testing of the mower.

3.1 Mower Design and Implementation

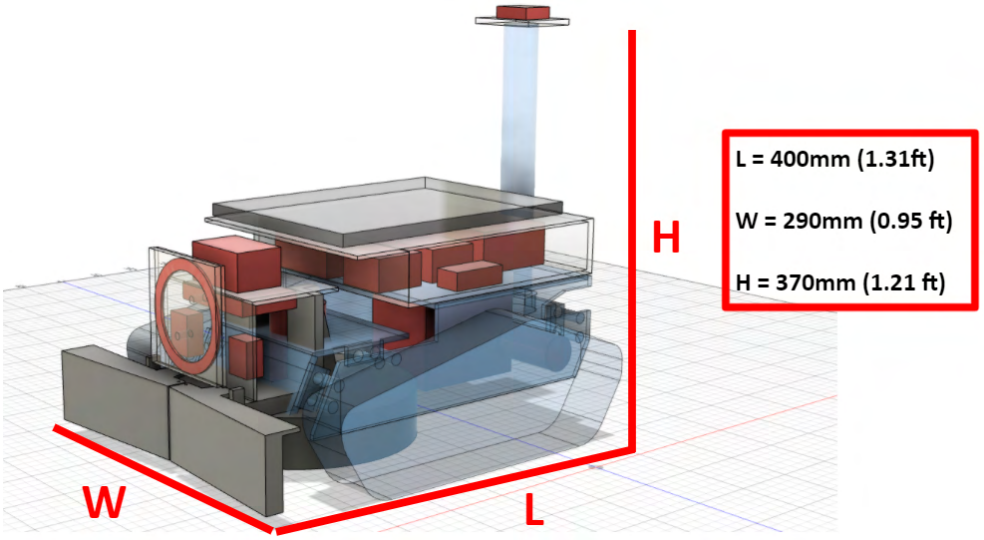

Figure 3.1 Mower Scale

As the mower acts at the central hub of all integrated components, it needs to be able to interact with both the docking station as well as the control device in addition to have the ability to automatically recharge its battery, plot paths, and cut grass without the user actively guiding the system.

3.1.1 Mower Power Supply and Recharging System

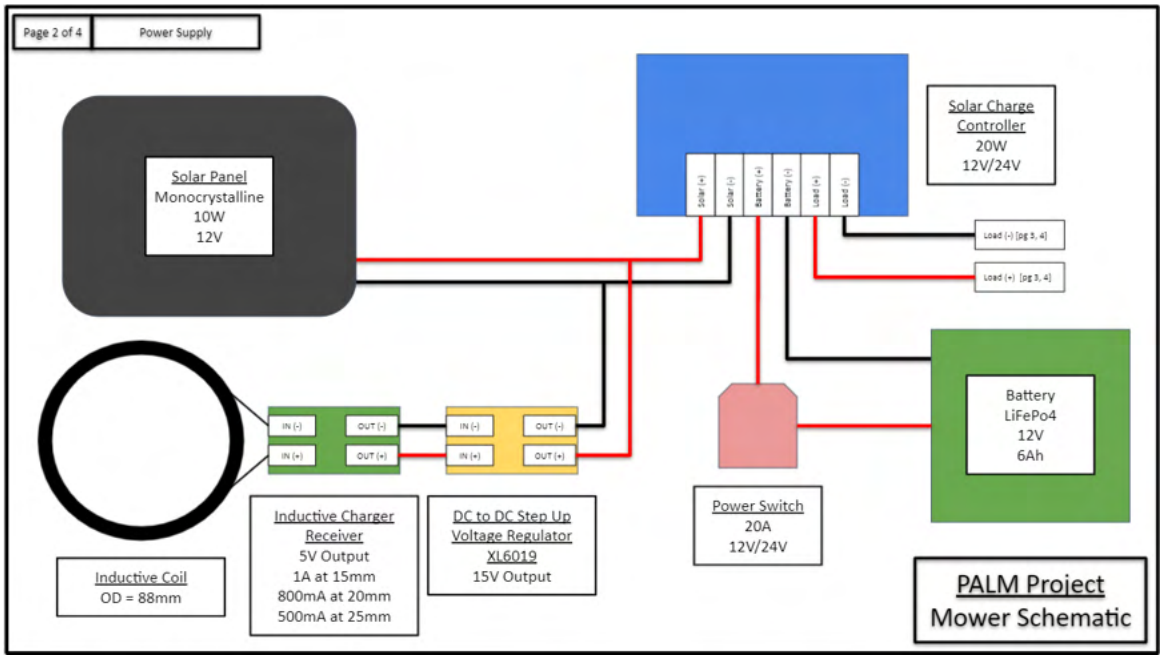

Figure 3.2 Mower Power Supply Schematic

The mower is designed to be self charging and maintain its own power to the system. There will be two methods of recharging the onboard battery; solar power and inductive charging from the docking station.

The power supply has an on/off switch that will allow the system to power on and fully power down for safe transportation. All power sources are controlled and monitored by the solar charge controller which directs power from the power sources (solar and/or inductive) to the battery, and from the battery to the system components.

The solar panel and inductive charge coil were both strategically placed to allow for the best possible chance of collecting solar power and when parked in front of the docking station. Originally, we considered placing the coil under the solar panel and having the mower park under the docking station, but we realized this would prevent the solar panel from collecting sunlight and rendering one of the two charging systems non-functional. This led us to move it to the front and park the mower up next to the docking station.

Figure 3.3 Solar Panel and Charging Coil Placement

The battery itself is mounted on the bottom of the unit behind the cutter blade. We installed a metal protective shield between the cutter blade and the battery to protect it from damage if the blade shroud fails to stop objects from passing through to the battery.

The battery is held in place with a compression craidel that pins it to the bottom side of the chassis. This prevents the battery from moving and holds it completely stationary.

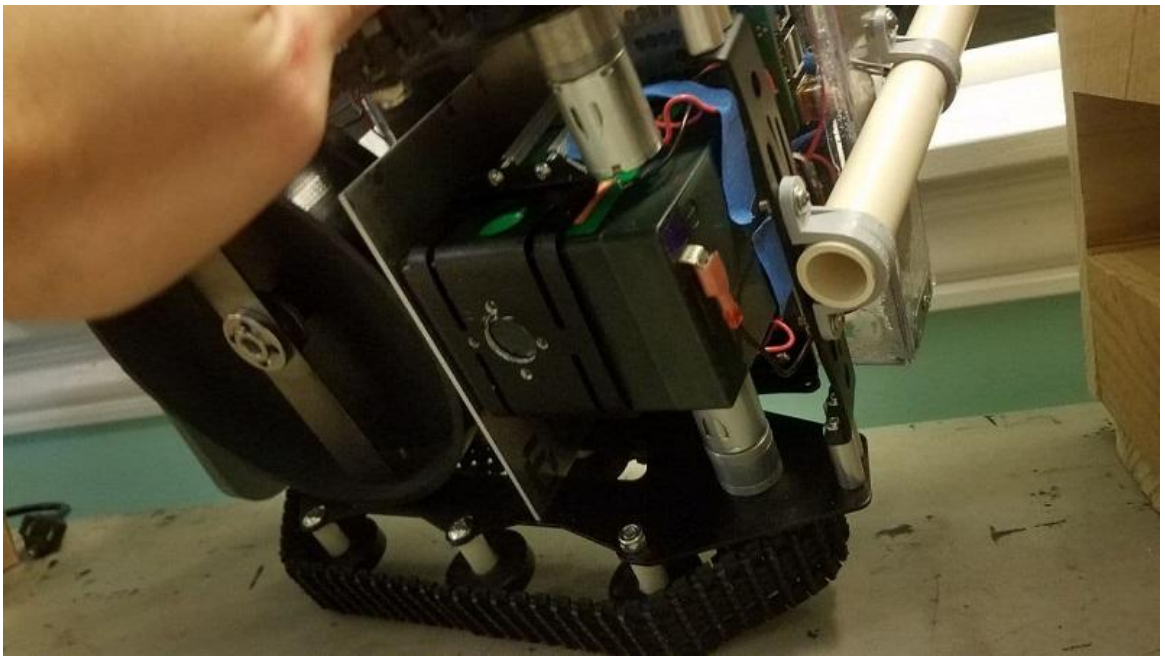

Figure 3.4 Battery Mount and Guard

Figure 3.5 Docked Mower

3.1.2 Raspberry Pi Controller and Peripherals

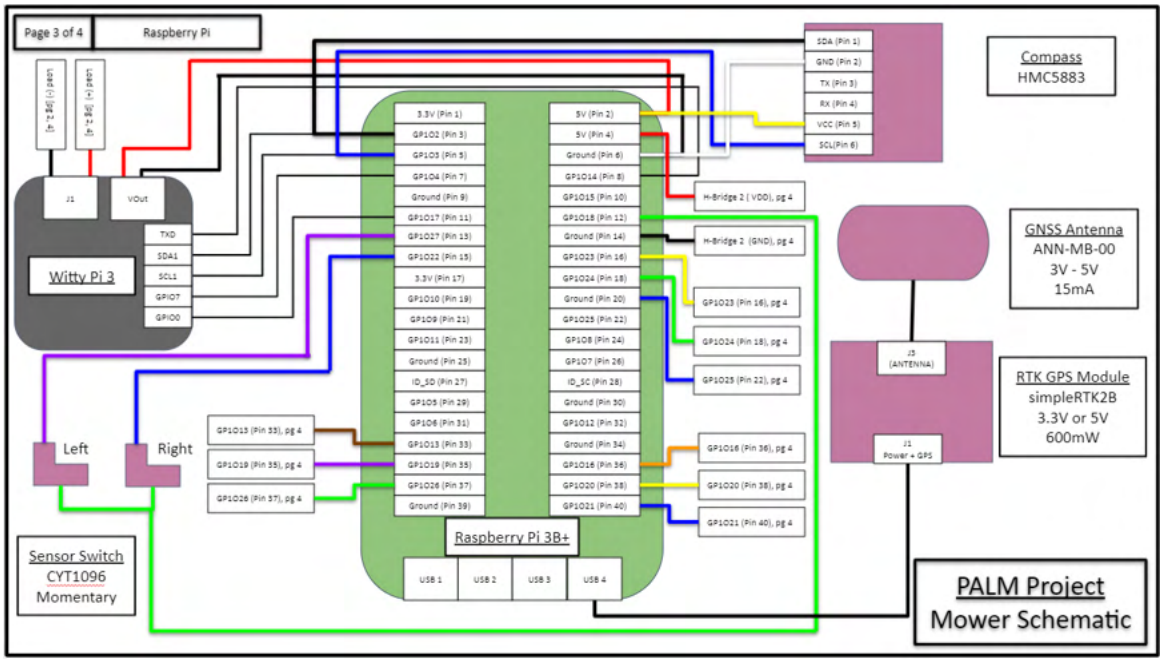

Figure 3.6 Mower Raspberry Pi Schematic

The mower's functionality is all controlled by the Raspberry Pi 3B+ Microprocessor and its peripherals. This is what will provide direction, movement, cutter blade control, and any sensors to the system.

The Raspberry Pi can power a handful of modules and can provide up to 1.2 A of power to its peripherals. The primary source of the current draw will be from the RTK system (600 mW) and XBee radio antenna (215 mW). The compass only uses 50 mA and the Witty Pi 3 will typically use less than 1 mA of power

The mower and docking station will have their own RTK capable GPS modules with GNSS antennas to connect to the satellites and radio communication to communicate with each other. The docking station will utilize the Base RTK module and be the stationary unit that provides correction data to the mower, which will have the Rover RTK module. This setup can achieve up to 1 cm level accuracy in open skies.

These two units will communicate with medium-range XBee onboard radio modules with a line of sight range of 0.7 miles. This will provide enough range for the scope of our project and is powerful enough to reach across a standard yard with plants and obstacles.

As the mower moves around with the cutter blade active, it will need a way to detect if something is in front of the unit to safely move around the object and continue cutting. In this case, we are using two momentary switches that are activated by a front bumper arm that will send a signal to the Raspberry Pi that there is an object in the path and to execute the subroutine to move around the obstacle.

The goal is that the mower will move right or left, depending on what sensor is tripped, to attempt to move clear of the object in its path.

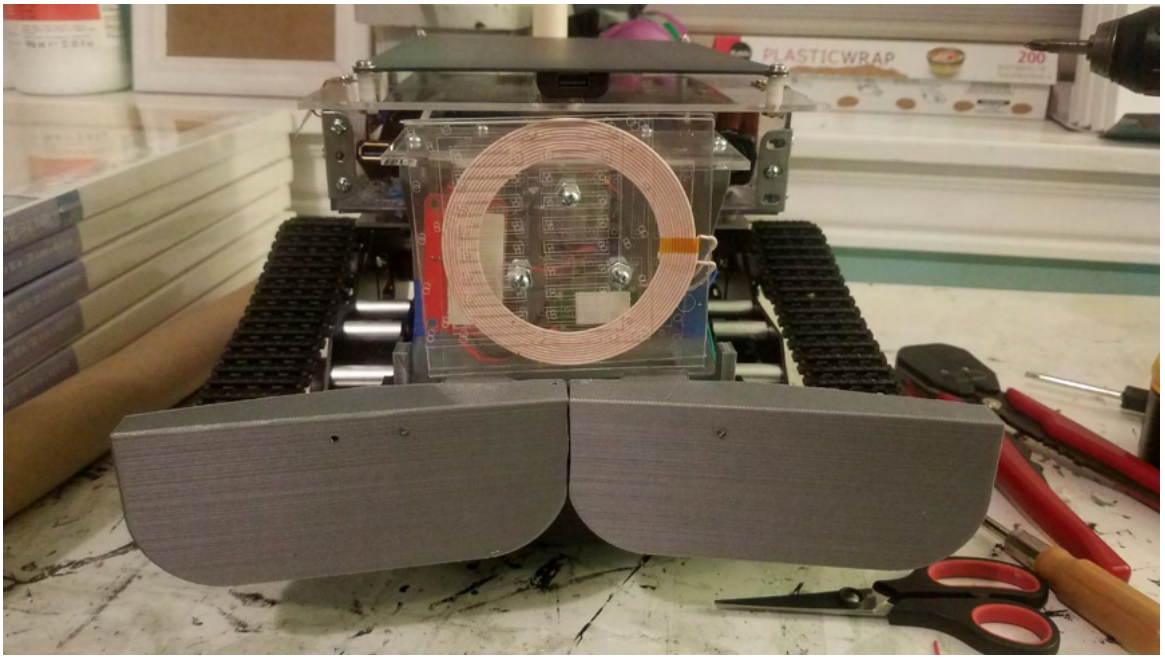

Figure 3.7 Front of Mower with Bumper Sensors

Integration of all the components works well on paper, but installing them into the limited space of the unit required some space management. The Raspberry Pi and Witty Pi are mated together with the Witty Pi plugged directly into the 40 pin header. This allows the boards to share pins and conserve space. The power to the circuit is plugged directly into J1 on the Witty Pi from the charge controller load circuit.

Since we are plugging the simpleRTK2B into the USB slot of the Raspberry Pi, it will let us both power the RTK module and pull coordinate data at the same time. The RTK module will then be mounted near the front of the mower underneath the GNSS antenna module.

We utilized two extra USB extender cables from the Raspberry Pi to both allow for a spot to plug in the RTK system and provide easy access to the Raspberry Pi for programming and data transfer.

To change any power schedule parameters or auto-shutoff settings for the Raspberry Pi, we need to install the Witty Pi's software and utilize its interface settings which is accomplished through the Raspberry Pi's terminal interface.

Since these compasses can easily be thrown off by a magnetic field, we have the compass sitting well above the mower and away from the motor's field of flux. We also removed any magnets that were installed in the unit (like the GNSS antenna) to further reduce magnetic interference.

Figure 3.8 Compass Module

3.1.3 Motor Control

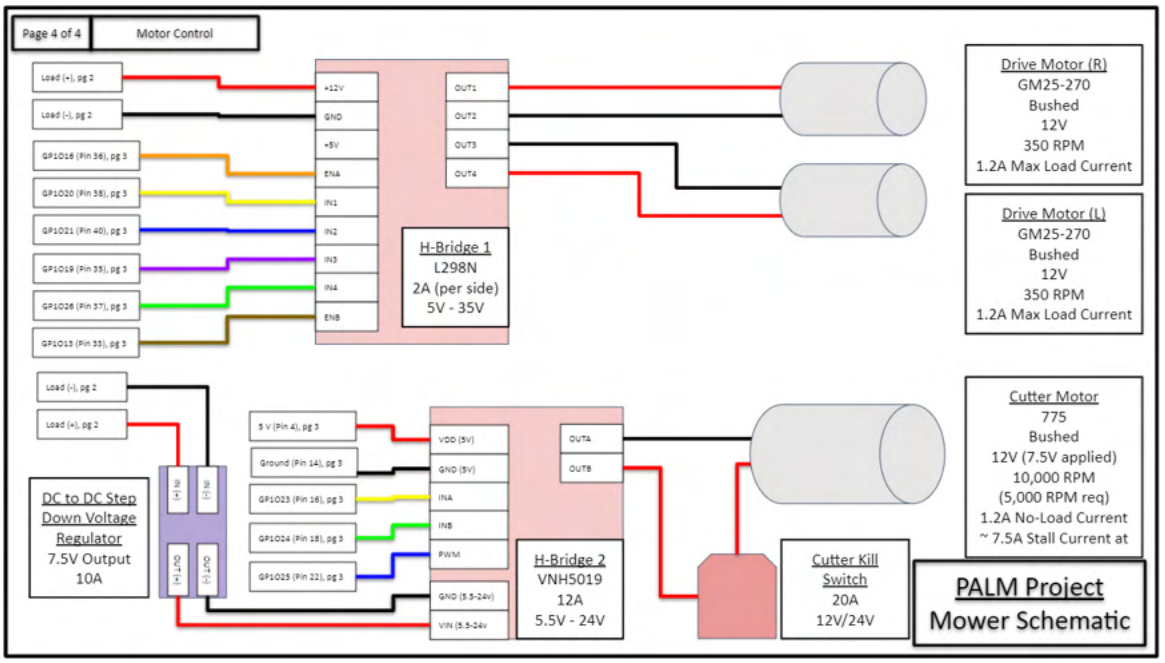

Figure 3.9 Mower Motor Control Schematic

In order to control the power and direction of rotation for both the drive motors and cutter blade motors, there needs to be circuits in place that can both provide enough current to operate the motors as well as directional control for movement. We utilized two different h-bridge motor controllers to accomplish this task; an L298N dual motor controller and a VNH5019 motor controller.

Before hooking up the motor controllers to the Raspberry Pi, we needed to select and designate several GPIO pins as the Input and Enable pins for our motor controller, which can be viewed in Figures 3.6 and 3.9. This is important as they will send high or low input signals to the motor controllers for directional control and power output via PWM.

After these pins are selected, the motor power supply must be fed into the motor controllers so that it may regulate the output supply. Doing this causes some voltage drops to occur and won't have exactly the same output voltage as is supplied to the motor controller. This will be gone over in detail in Chapter 3.9 Testing and Evaluation.

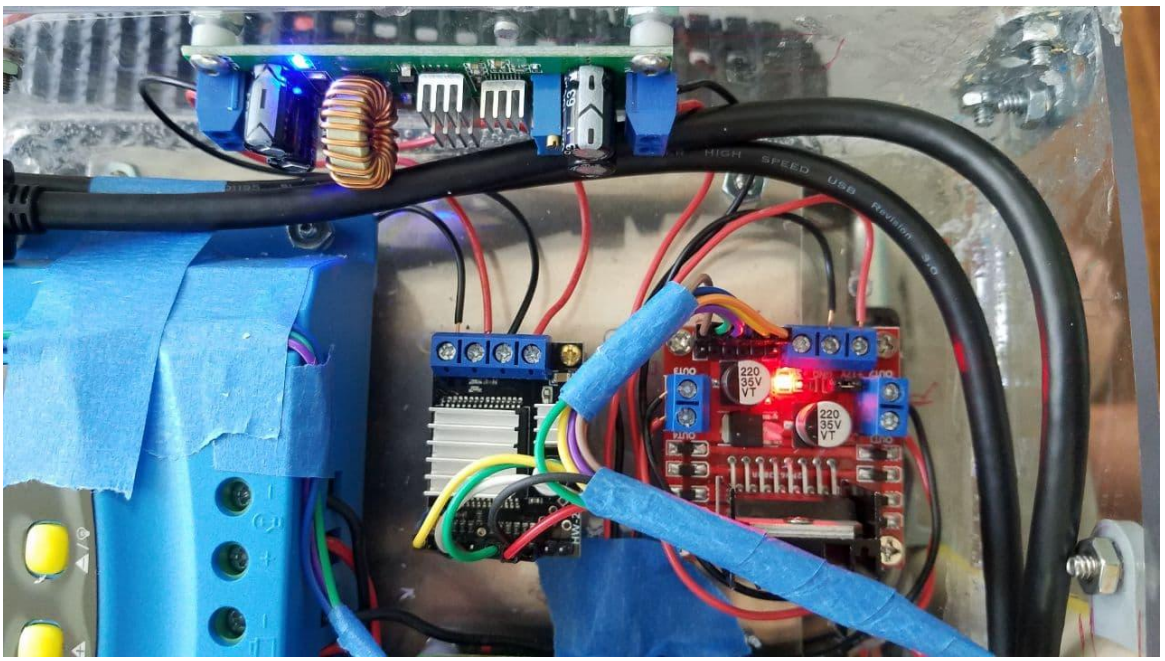

Figure 3.10 H-Bridge controllers and 10A Voltage Regulator

The drive motors are connected directly to the output voltage of the L298N h-bridge for motor 1 and motor 2 outputs. The cutter motor has a power switch installed between the positive VNH5019 output motor and the positive 775 motor pin as a manual cutoff switch for safe testing and manual movement.

3.2 Docking Station Design and Implementation

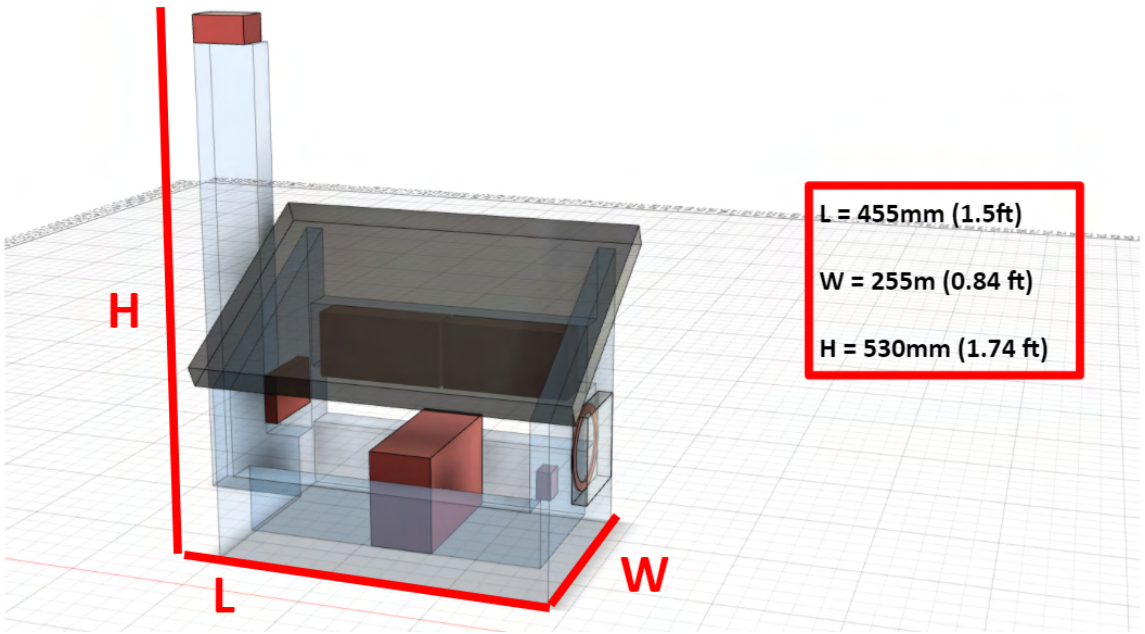

Figure 3.11 Docking Station Scale

In order for the mower to have accurate position data and have a place to return to home, the system needs an immobile docking station that is placed in an area with access to open skies for satellite communication and solar recharging.

This docking station will have the ability to transmit correctional position data to the mower utilizing its onboard smipleRTK2B system, boost the recharging of the mower battery, as well as supplying its own power via the 10W solar panel.

3.2.1 Docking Station Power Supply and Charging System

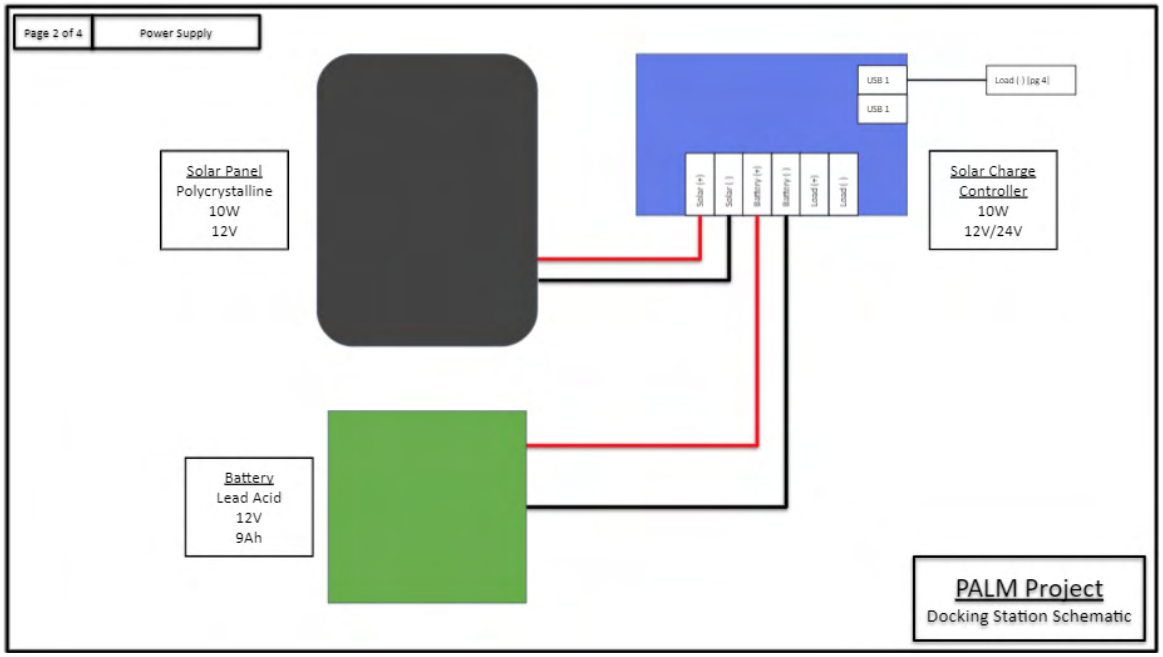

Figure 3.12 Docking Station Power Supply Schematic

The docking station only has a few items it needs to power and is designed to be on continuously, so the power supply has a much lower energy demand when compared to the mower, but still has to be able to supply power via the inductive charging coils.

As there are no size limitations compared to that of the mower, we were able to install the components without much interference with one another. The solar panel sits on top of the unit as a cover, shielding the internal components from direct sunlight and the weather.

The solar panel is connected to the solar charge controller solar inputs and the battery is connected to the charge controller's battery terminals, powering the system.

The primary load in this circuit is driving the GPS module and is powered through USB 1.

3.2.2 Docking Station GPS Module

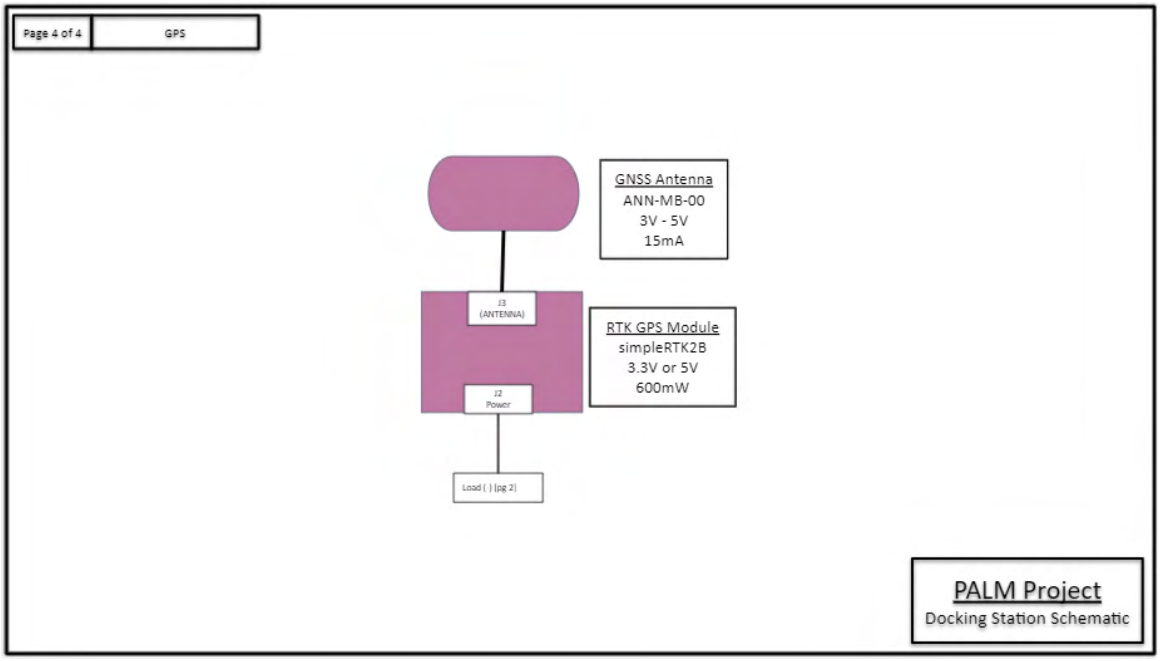

Figure 3.13 Docking Station GPS Module Schematic

The primary role of the docking station is to provide correctional GPS data to the mower via its onboard simpleRTK2B GPS module. This system has an integrated XBee RF transmitter that actively sends corrected positional data to the mower within a line of sight range of 0.7 miles, making it just the right range for a typical backyard with plants and trees. The entire system (simpleRTK2B, GNSS antenna, and XBee module) are all powered by the USB ports on the charge controller.

3.2.3 Inductive Charge System

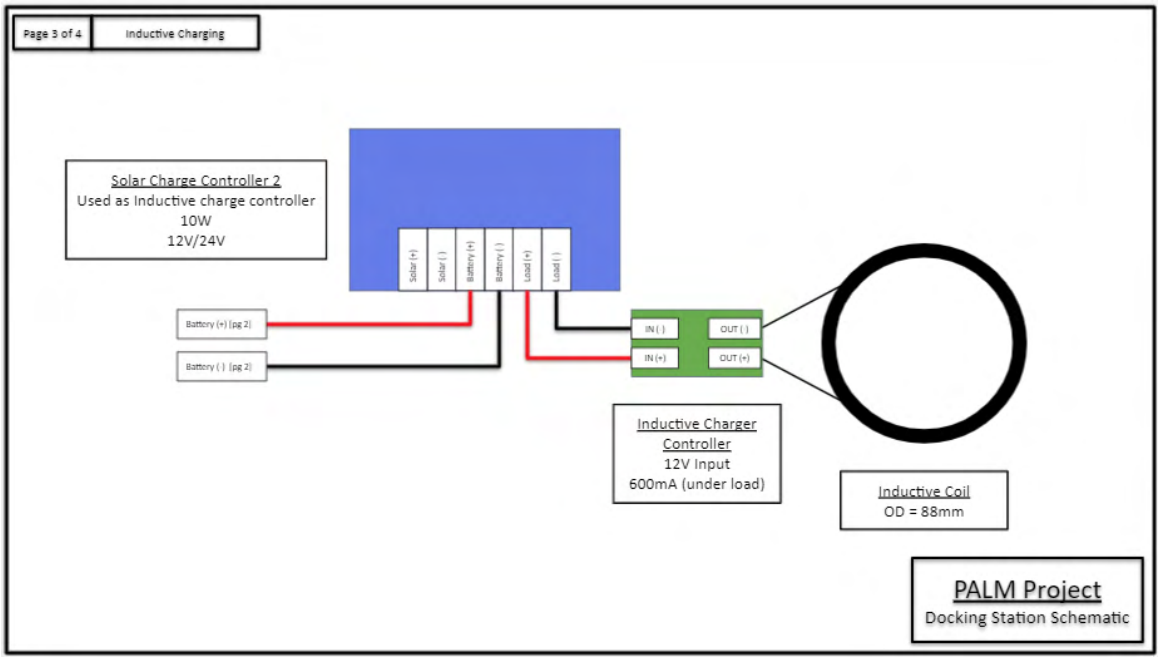

Figure 3.14 Docking Station Inductive Charge Schematic

The secondary role of the docking station is to provide a charge booster to the mower via inductive coils. This allows the docking station to actively transfer power from the docking station to the mower while docked and with no mechanical connections. This will allow the mower to recharge as designed with its solar panel on top and charge assist with the inductive coil on the front of the mower.

To prevent overdraw of the docking station battery, there will be a secondary charge controller connected directly to the battery and controlling the voltage to the inductive coil transmitter. This way, the system doesn't allow the mower to consume too much of the docking station's power and stop the transfer of power until the docking station's battery has enough power to become active again.

3.3 Controller Design and Implementation

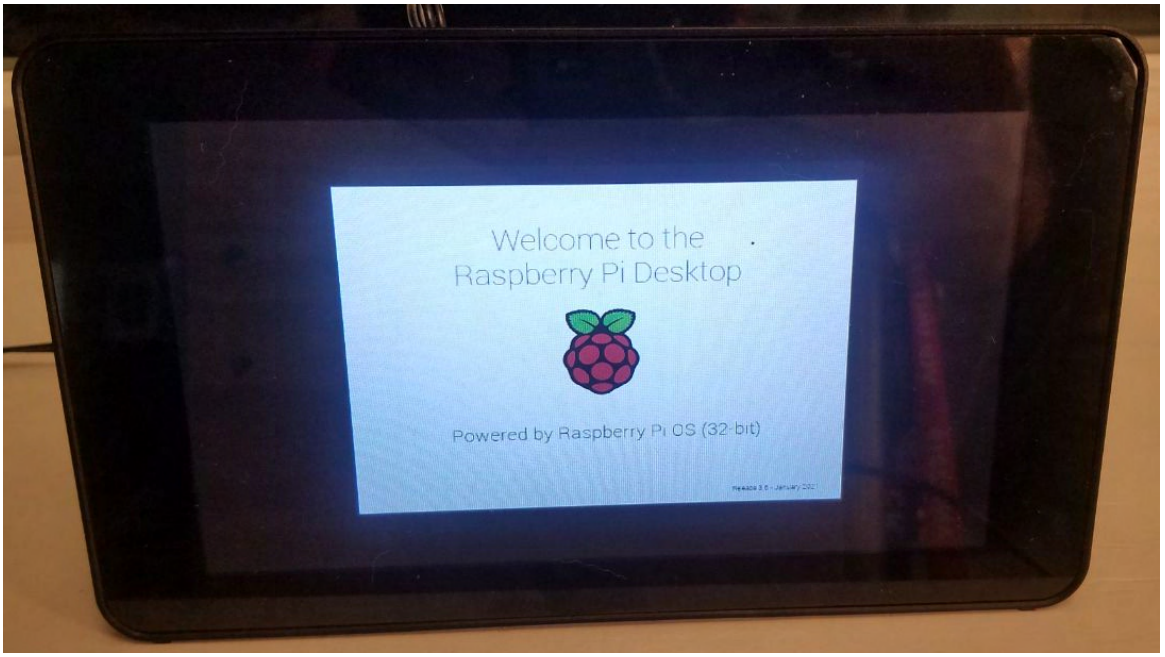

Figure 3.15 Mower Controller

To control the mower, either while programming or for manual cutting, we have a touchpad controller utilizing a Raspberry Pi 3B + and touchscreen interface. This will be a handheld device that allows for direct access to the mower.

3.3.1 Raspberry Pi Interface

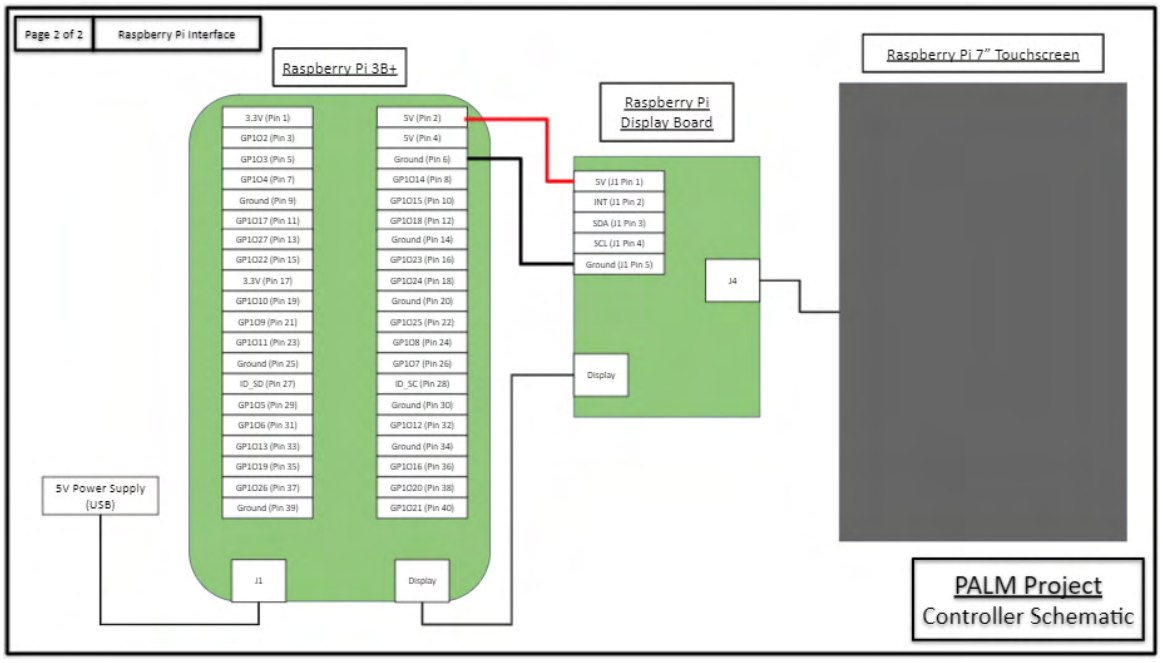

Figure 3.16 PALM Touchscreen Controller Schematic

The Raspberry Pi touchscreen controller is easily assembled, all that needs to be done is supply power to the screen and display board from the 5V pins on the Raspberry Pi and connect the display ribbon cable between the two boards. Due to budget and time limitations, the power supply will be from a Raspberry Pi power cable that plugs into any wall outlet.

Installation is easy and convenient when paired with the Raspberry Pi Tablet case as all the components fit properly in the case and allows all external plugs to be easily accessed. The only thing we had to do was rotate the touchscreen display 180 degrees within the code so that we can utilize the built-in stand and have the plugs in a convenient position for use.

Due to time constraints, we are utilizing the remote desktop function on the Raspberry Pi by installing the xdrp program and using Remmina as our remote desktop portal. Upon powering the controller, Remmina opens up with the mower displayed for remote access. While we could make the system automatically connect, we decided to leave it in list format for selection in case the user has multiple mowers available to choose from.

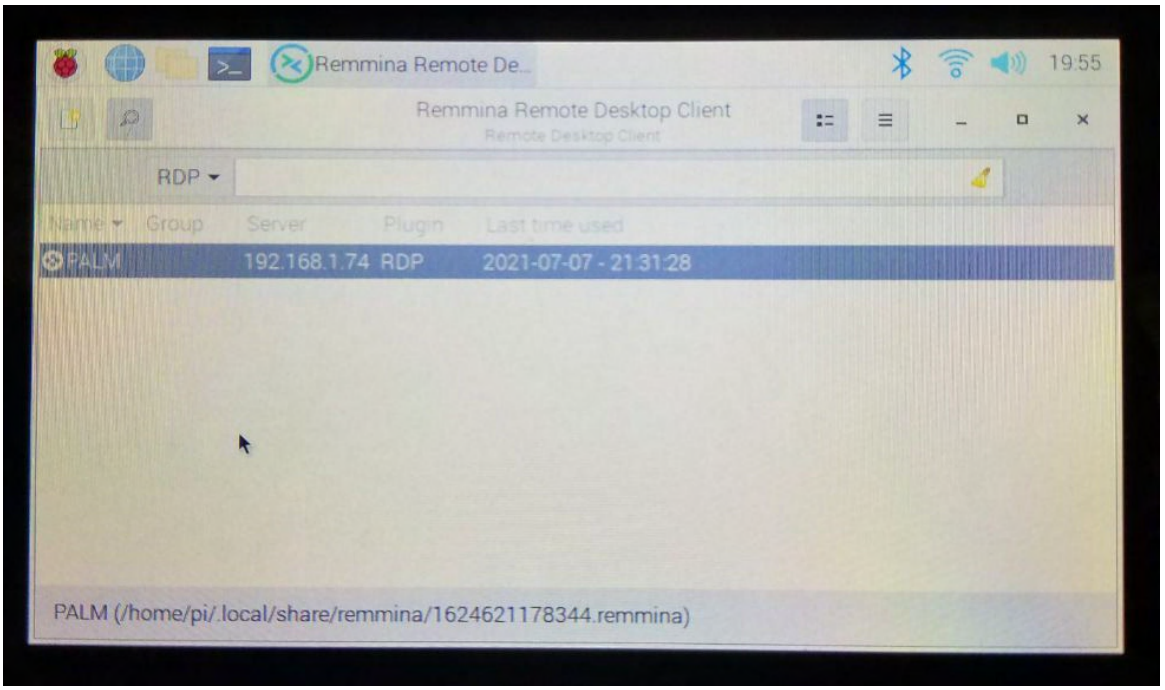

Figure 3.17 Remmina Remote Desktop Interface

3.4 Software

The mower required quite a bit of software to get all of its components working and communicating with each other at the same time. It was important to make sure that each component functioned as expected, especially due to the potential harm that the mower could cause if anything went wrong.

3.4.1 Initial Setup

The first time the mower is used the user must define a few important variables which the device will use to navigate. The first two variables that need to be determined are the longitude and latitude of the starting position of the mower. The starting position of the mower should be its exact spot when the mower is docked to the docking station. This spot will determine where the mower returns to once it is done mowing.

The next four variables required by the mower are the values of the angles when the device is facing "forward", "left", "back", and 'right" in that order. Every user's angles are going to be different depending on where they are in the world which is why it is necessary to specify which angle values correspond to the user's lawn. The final two values that the user needs to determine are the height and width in feet of the square representing their lawn. These two values will be used by the mower to determine the borders around the user's lawn.

3.4.2 Saving User Values

The variables provided by the user during the initial setup need to be saved permanently on the device so that it can reference those values every time it starts up and begins mowing. Having to set up the mower every time it turns on would make for a poor user experience. To avoid this the values set by the user are saved onto a text file within the project folder.

During the initial setup, the user will simply be asked to enter the corresponding values in order, the program will then take the values provided by the user and save them onto the text file. Every time the program starts up it will reference this file and assign the correct values accordingly. With this, the values are permanently saved while also being able to be changed at a later date if the user desires.

3.4.3 Graphical User Interface

The graphical user interface (GUI) is an important part of the user experience. The GUI designed for the mower is a total of ten buttons. The first 8 buttons are directional buttons to control the mower. These include the four standard directions of up, down, left, and right as well as 4 extra directions in between each in case the user wants more fine control of the mower. After this, there is 1 button for control of the cutter blade, 1 for toggling manual mode, and 1 kill switch. The kill switch was an important safety feature to be able to shut down the mower from a distance if need be.

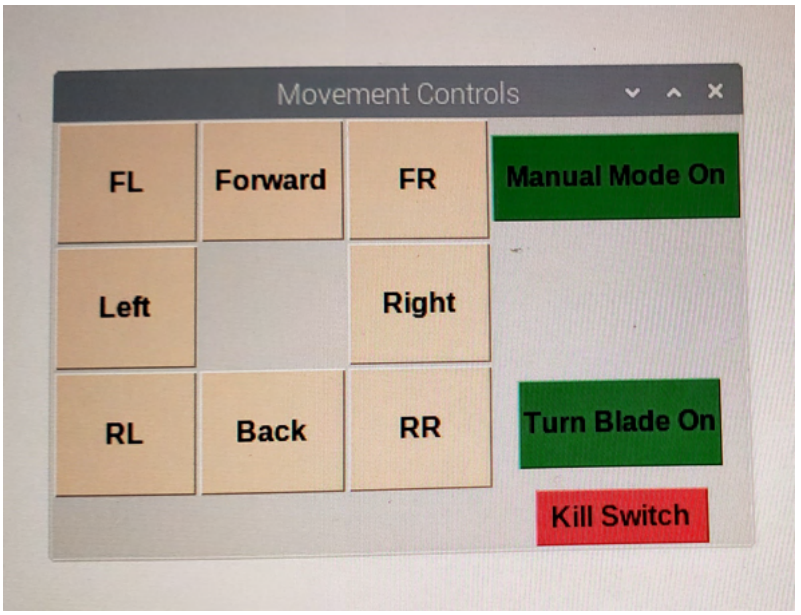

Figure 3.18 GUI

3.4.4 Compass

The compass is essential for the mower to determine which direction it is facing. While the program is running the compass is constantly feeding the mower with its current angle between the values of 0 degrees and 360 degrees. One of the issues encountered when programming the compass was comparing angle values while compensating for angle wrap-around in the coding. If you have an angle of 350 degrees and you add 30 degrees to it the program would interpret that as 380 even though it's actually supposed to be 20 degrees.

To solve this issue the given angle value is taken and decreased by 180 giving us an angle between -180 to 0 and 0 to 180. Next, the code takes the absolute value of both

angles being compared and checks if the difference between them is larger than the established error range of 10 degrees. As long as the current angle of the mower is less than 10 degrees off of the desired angle, the mower accepts that angle and begins to move forward. It was necessary to give the angle an error range due to the poor control offered by the tracks when turning the mower.

3.4.5 Steering

Getting the mower to turn caused significant issues during the coding phase of this project. Ideally, the mower was supposed to continuously turn in one direction as slowly as possible to guarantee that it could acquire the most accurate possible angle.

The problem was that because of the significant amount of friction provided by the tracks the mower had a minimum amount of power required just to begin turning. Any power level below this would simply cause the mower to stay in place unable to turn because it is unable to overcome the friction of the tracks with the ground. The issue was that the instant that the mower had enough power to begin turning it would begin turning at a relatively high speed. So either the mower wasn't turning at all or it was turning too fast to acquire an accurate angle.

To compensate for this problem, two things had to be done while creating the code to control steering. The first was to abandon the idea of continuous turning and instead turn in several short bursts. Each burst was set to turn at the speed required for the mower to overcome friction and begin turning, but it would only do this for a tenth of a second. This allowed the motors to deliver enough power to get the mower moving while moving the mower slowly enough that it had a chance to acquire its desired angle. Any less than a tenth of a second and the motor wouldn't have enough time to get the mower moving.

The second thing that had to be done was to give the mower a certain amount of error room while selecting the angle it needed to turn. Ideally, if the forward angle was 90 degrees, the system could simply wait until the mower was at exactly 90 degrees to begin moving forward, but because the mower had to move in short bursts it would simply take too long for the device to land on exactly 90 degrees. Instead, the mower is allowed to move as long as its current angle is between its desired angle plus or minus 10 degrees. This error range combined with moving in short bursts allowed the mower to acquire an accurate enough angle while compensating for the tracks.

3.4.6 Lawn Borders

The borders of the lawn are determined by four points that are calculated during the initial setup of the mower as well as the selected height and width of the lawn. These four points represent the four corners of the lawn, this assumes that the lawn is either a square shape or a rectangular shape. The mower will always have a predetermined starting position which is ideally right next to the docking station.

Now to determine the right border of the lawn the mower takes the value for width and chooses a point that is that distance away from the starting position towards the east. The same process is repeated using the height value to determine the top border of the lawn. So basically if the starting position of the mower is x and y, then the value for the right border would be x+width and the value for the top border would be y+height.

3.4.7 Mower Pathing

Once the borders of the lawn are determined the mower can begin moving. To navigate around the lawn the mower uses the four angles that were determined during the initial setup. The forward angle represents the direction the mower is facing when traveling from the left border of the lawn to the right border of the lawn. The backward angle represents the direction the mower will be facing when traveling from the right border of the lawn to the left border of the lawn. The left angle represents the direction the mower will be facing when traveling from the bottom border to the top border, and the right angle represents the direction the mower is facing when traveling from the top border to the bottom border.

To begin moving the mower first starts turning slowly until it acquires the forward angle. Once the angle is acquired the mower can move forward until it reaches the right border which then signals the mower to stop. After reaching the right border the mower now begins to turn until it acquires the left angle which is the angle facing the top border. Once it has this angle the mower moves forward just a little bit, this distance that the mower moves forward will be the distance between the side to side paths or the rows of the lawn that the mower will follow. It is important that the distance between these rows needs to be less than the width of the mower otherwise it will leave uncut patches of grass in between the rows.

Next, the mower needs to turn until it acquires the angle facing towards the left border of the lawn. Once this angle is acquired the mower begins to move forward until it detects it has reached the left border. Finally, the mower begins to turn again until it is facing towards the top of the lawn and moves forward just a little bit. At this point, the loop is completed, and the loop will jump back to the beginning where the mower begins to move towards the right border again. This loop of going left to right while moving up slightly will continue forever until the mower detects it has reached the top border. This specific pathing design makes it very simple to integrate whatever width and height the user may pick for their lawn.

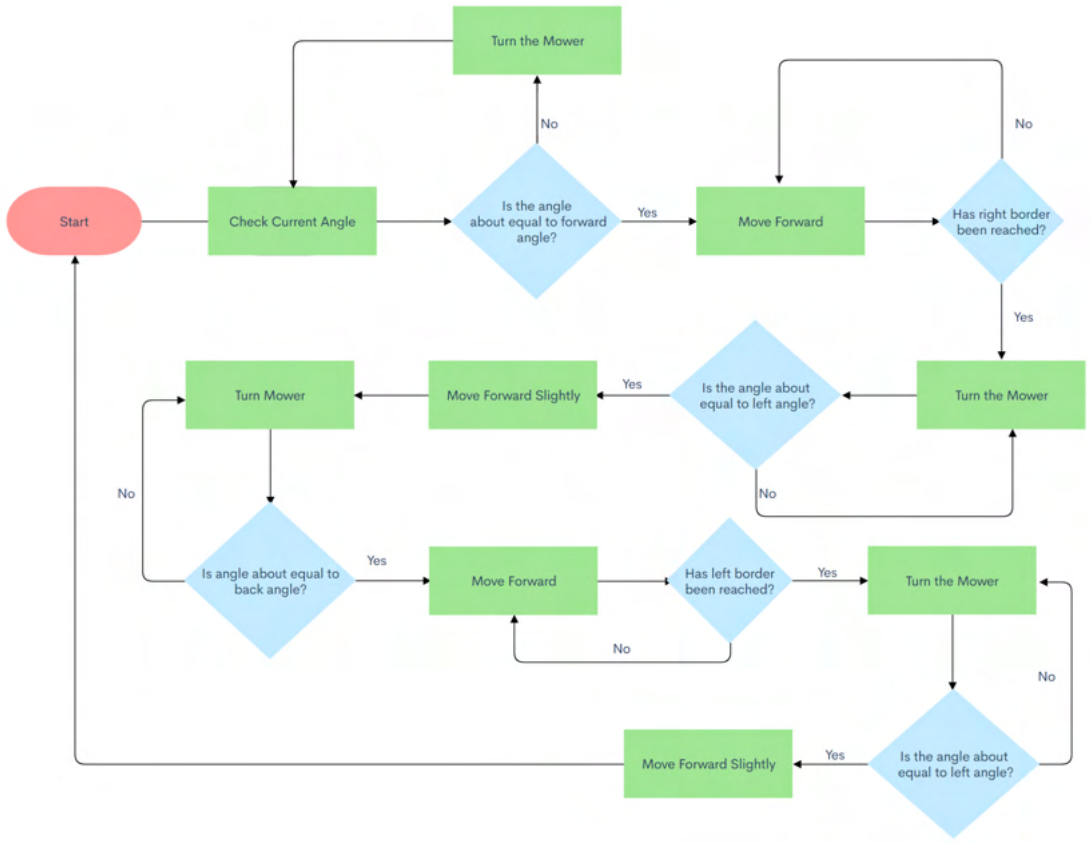

Figure 3.19 Pathing Flow Chart

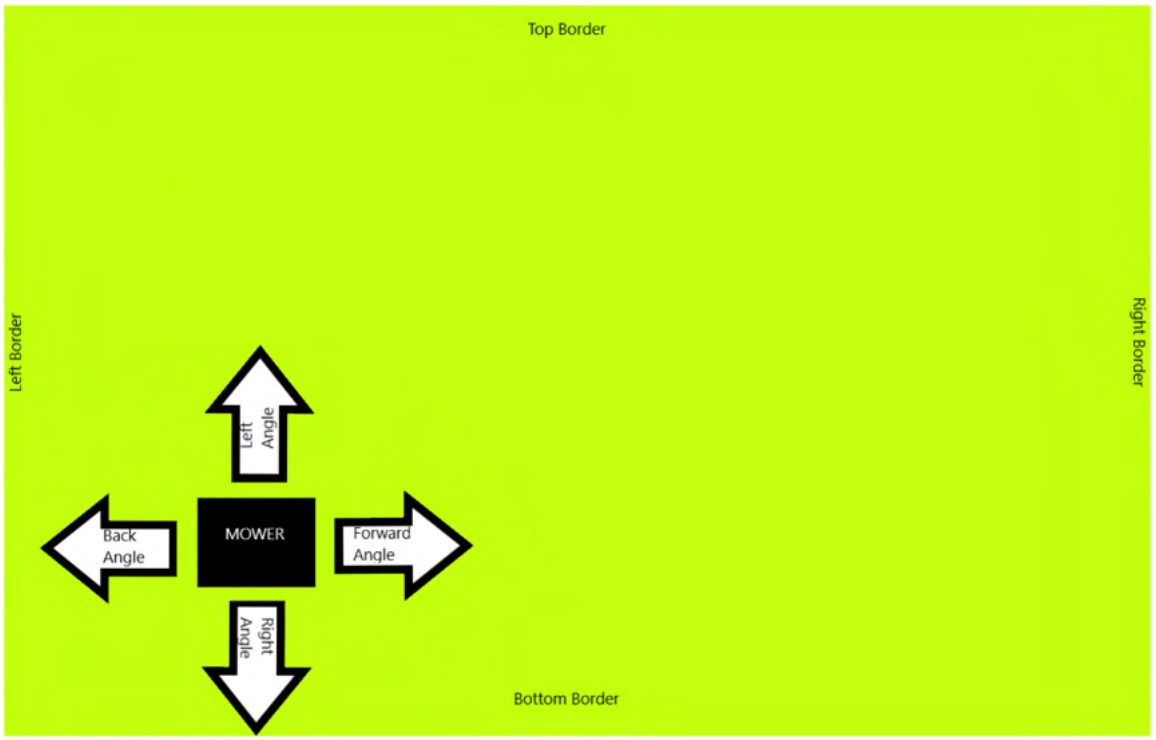

Figure 3.20 Angle to Border Relation

3.4.8 Return Home

Once the mower detects it has reached the top border of the lawn it begins the return to home procedure. The first thing the mower does is turn until it acquires the angle facing towards the right border of the lawn. The mower then moves forward just a little bit unless it detects that it is against the right border. The reason for this is to give the mower a little room between it and the docking station so that it doesn't land on top of it when returning towards the bottom of the lawn.

Now the mower begins to turn until it acquires the angle facing towards the bottom of the lawn. Once it has this angle the mower will begin moving forward until it detects it has reached the bottom border. Now the mower begins to turn until it acquires the angle facing towards the left of the lawn or towards the docking station. At this point, the mower is ready to begin the docking procedure and now begins to move forward slowly and in short intervals until it detects it has reached the starting position where it was docked.

Finally, once the mower has reached the starting position it simply shuts down until the next scheduled lawn maintenance. Unfortunately even with the centimeter accuracy provided by the RTK getting the mower to perfectly dock onto the docking

station was not achievable. This was because even being just a few centimeters off would prevent the mower from receiving a charge from the docking station. The inductive coils on both devices would have to be essentially perfectly lined up which even the RTK couldn't achieve.

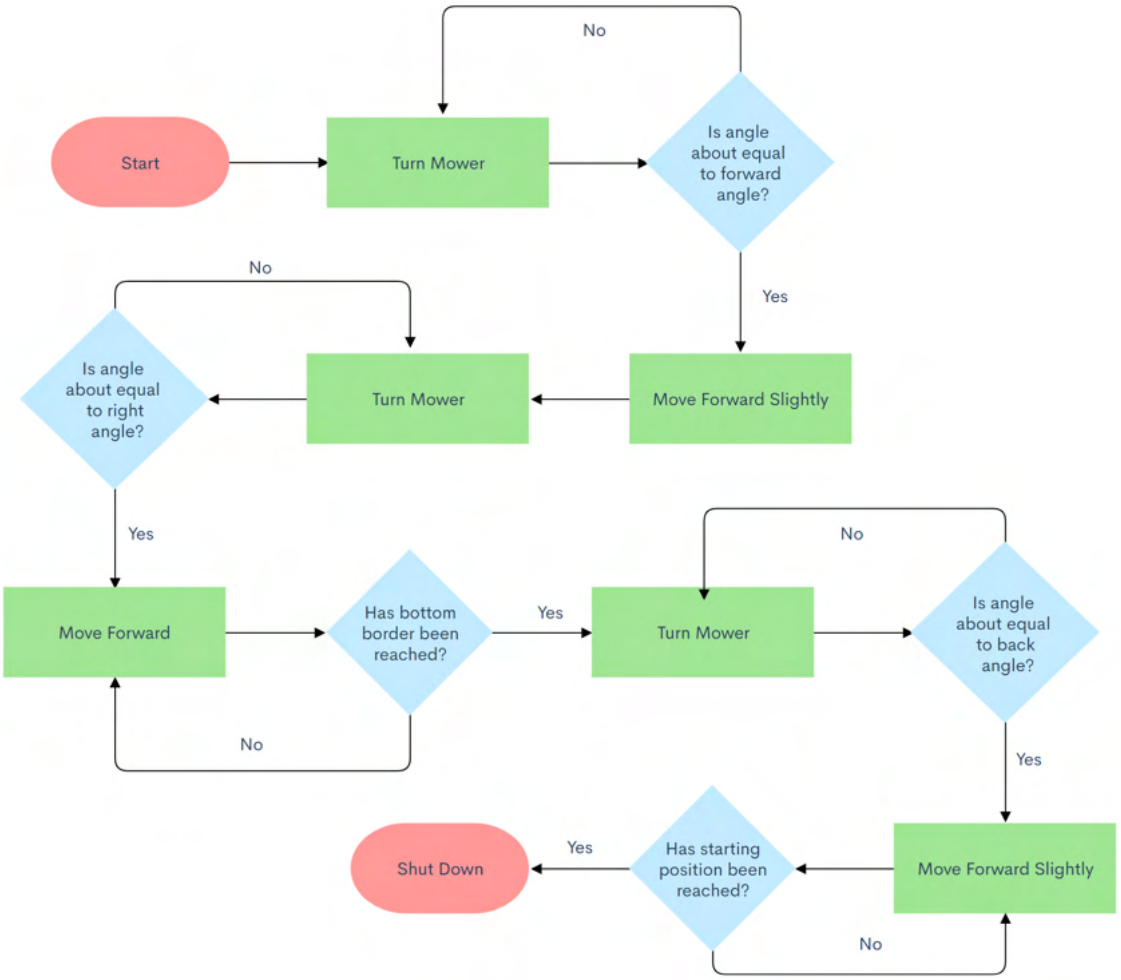

Figure 3.21 Return Home Flow Chart

3.4.9 Progress Counter

The mower keeps track of its current progress by using a section counter. The first section where the mower is moving towards the right border has a section value of 0. Once it reaches the right border it increments the section value and moves on to section 1 which is moving towards the top border ever so slightly. Moving towards the left border is section 3 and moving up towards the top border again is section 4.

Once section 4 is complete if the top border has not been reached then the section counter is reinitialized to 0 and the loop begins again. This section's counter value can

only be changed once the mower has detected that it has reached the end of its current section. The advantage of this is that the mower will always know the last section where it left off. This way in case the mower has to stop for safety reasons or if it's moved by an outside source it would not affect the motor's ability to continue with its path.

Once the mower reaches the top border it arrives at section 5 which signals the device to begin the return to home procedure. Section 5 represents finding the angle towards the right border. Next section 6 represents moving forward after acquiring this angle. Section 7 represents finding the angle towards the bottom border, while section 8 represents moving forward until the bottom border is reached. Section 9 represents finding the angle facing towards the docking station, and finally, section 10 represents moving forward slowly until reaching the starting position. After this, the device is shut down and the section counter is reset once the program re-initializes.

3.4.10 Obstacle Sensor

The obstacle sensor was an important safety feature not only to protect others from being harmed by the mower but also to protect the mower from crashing into objects and harming itself. The obstacle sensor was also important to keep the mower from getting stuck on something preventing it from mowing the lawn. When it came to programming the obstacle sensor the most important thing was deciding in which direction the mower was going to move when traveling around obstacles. It was very important that the mower wouldn't try to move around obstacles in a direction that would cause it to either go past the border or bump into a wall where the borders might be.

The solution to this problem was simply to tie the reaction of the mower to the current section of the mowing procedure. If the mower detects an obstacle while it is moving towards the right border then it is safe to assume that there will be space between the mower and the top border of the lawn, so the mower will dodge to the left. Next, if the mower is moving towards the left border when it detects an obstacle then it will dodge to the right.

If the mower runs into a particularly wide obstacle, it will simply keep bumping into it and moving to the side repeating that process until it finally clears it. If the mower happens to detect an obstacle at the very top of the lawn right by the top border then it will simply dodge towards the top border and begin the return to home procedure once it

detects it has reached the top border. Finally, if the mower detects an obstacle during the return to home procedure it will simply dodge towards the right border and then continue moving down until it reaches the bottom border.

3.4.11 RTK Lock Safety

Unfortunately, the RTK module proved to be less stable than expected. This could be due to several different factors including the time of day, atmospheric conditions, location of the lawn, the height of the docking station, the amount of foliage obstructing the view of the sky around the lawn, and others. Because it was not unlikely for the GPS to lose its RTK lock in the middle of mowing it was important to add some kind of safety feature that would prevent the mower from going past the borderline in case this did happen. Luckily there is a way to determine the current accuracy of the RTK module.

One of the variables that the RTK provides has this specific purpose. The value for this variable can range from 0 to 6. The most important value to the program is 4 because this is the value given when the RTK lock is engaged. Knowing this the mower was programmed to move only when it detected that the RTK lock was engaged. If the mower detected that the RTK lock was lost then all movement would cease and the cutter blade would turn off. The mower will then simply wait until it reacquires an RTK lock and continue mowing where it left off. Unfortunately, this does mean that the mower will occasionally stop in the middle of operation which would increase the amount of time it takes to finish mowing, but this is still a very important and necessary safety feature to prevent possible accidents due to the loss of GPS accuracy.

3.4.12 Stuck Counter

One important thing that had to be addressed was the possibility that the mower could lose the RTK signal and take too long to reacquire it. If the GPS was having a particularly bad day and the mower just kept waiting forever until it reacquired an RTK lock then there was a risk that the mower would keep waiting until it simply ran out of power. This would be a very significant waste of battery life.

To solve this a stuck counter was added to the program. The purpose of this counter is to keep track of how long the mower has been waiting since it last lost its RTK lock. Each second that passes with the mower not having an RTK lock will increment the stock counter by one. For this project, it was decided that the mower will wait a

maximum of 30 minutes (until the counter reaches 1800) to reacquire an RTK lock before automatically shutting down. This way the mower wouldn't have to recharge its battery from zero for the next scheduled lawn maintenance. The mower also has the capability to send an email notification to the user when it's been stuck for too long. Once the mower detects that the RTK lock has been lost for 20 minutes, it sends an email notification to the user to warn them that the mower is stuck. This gives the user 10 minutes to move the mower before it shuts down. Finally, if the mower reacquires an RTK lock in the middle of the countdown then the stuck counter is reset to zero and the mower continues mowing where it left off.

3.4.13 Auto Start

A very important requirement for the mower to be fully automatic was that it needed to be able to shut down and startup on its own. Once the raspberry pi is shut down it requires manual input to turn it back on. The raspberry pi by itself does not have a way to autoboot from a complete shutdown, but here is where the WittyPi comes into play. The WittyPi possesses a real-time clock that can be powered by a separate battery and can be programmed to boot up the raspberry pi according to any schedule that is chosen. In this case, the WittyPi is programmed to turn on the raspberry pi once a week. After turning on the raspberry pi is allowed to run for 2 hours before the WittyPi shuts down the mower and waits until 1 week has passed to turn it back on.

The final requirement for the mower to be fully automatic was to have the mowing program automatically run as soon as the device starts up. Even if the device turned on by itself if the user had to hit run on the program to get it to function then it would no longer be automatic. Luckily the raspberry pi does have several methods for getting a program to run automatically on startup. In this case, the rc.local file was edited to include the full path leading to the mower python file. Any file referenced in the rc.local file will automatically run as soon as the device turns on. With these two things in place, the mower can now operate completely autonomously without the need for the user to worry about having to turn it on or off.

3.5 Troubleshooting

While assembling and doing initial runs with the system, we had to troubleshoot several problems that were either brought up as necessary improvements or limitations of the chosen components that didn’t arise until implementation.

3.5.1 Overall Cutting Space

One of the first hurdles we had to overcome was the limited cutting space available from the chosen mower chassis. The original design only provided the team with about 4.7” of usable space under the unit to install the cutter blade. To remedy this situation, we created an expansion mod using 12 standoffs that are 1” long, and fabricating a new top plate that allowed us to widen the entire unit by 2”, providing a usable cutting area of about 6.7” in diameter.

We chose to only use 1” long standoffs instead of longer ones to help prevent the unit from sagging in the middle or allowing it to twist during use. The result was a very rigid and secure chassis that provided more space for the cutter blade.

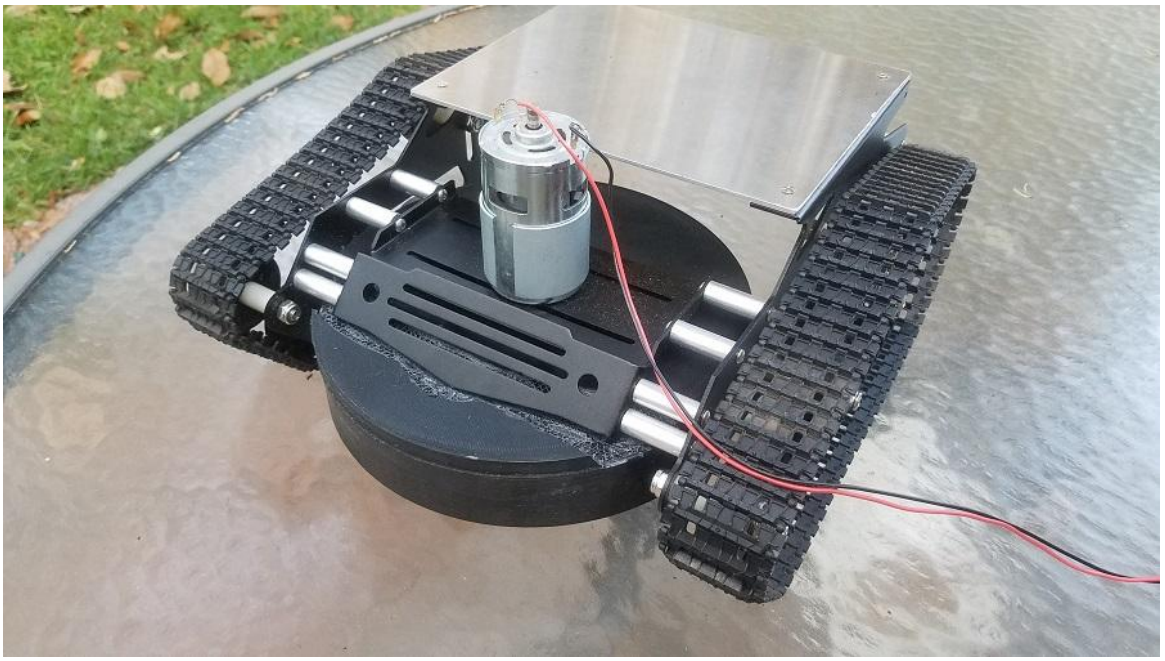

Figure 3.22 Mower Expansion Modification

3.5.2 Cutter Motor Controller

The original design for the 775 cutter mower was to utilize one side of the second L298N h-bridge motor controller, which was the same one used for the drive motors. However, we made a mistake in choosing this motor controller as it only 775 motor pulls up to 10A but mistakenly used the 1.2A of no-load current as what the motor needed to operate. While we had originally calculated the correct current needed by the775 motor in the power budget, we used the wrong figures in determining the right motor controller to run with it.

After this flaw was found, we decided to perform current use tests on the 775 motors to determine the actual load current when the motor starts up, once it reaches max speed, while cutting grass, and by attempting to stall the motor with sticks. These tests were performed 10 times to verify consistency and determine maximum spikes in current, which are shown in table 3.1 below.

Table 3.1 775 Cutter Motor Tests

775 Cutter Motor Current Tests Input Voltage = 7.5V	
Test Performed	**Average Current**
Current Spike at Startup	3.5A
Current Draw at Max Speed	1.7A
Current Spike Cutting Grass	5.5A
Current Spike During Stall Attempt	7.5A

We chose 7.5V as our input voltage as that allowed the cutter blade to achieve more than 6,000 RPMs, which is the desired cutting speed we chose to effectively cut grass for our system.

These results showed us that we need to pick a new motor controller and voltage regulator that can handle between 10A and 12A so that we have enough power going to the cutter blade at all times. After a day of research, we found the VNH5019 DC motor driver that can handle a continuous load of 12A and an effective 10A voltage controller to regulate the input voltage to 7.5V. This has proven during testing to be an effective motor controller for the cutter blade.

3.5.3 Inductive Charge Controller

To help boost the charge time of the mower and maintain the battery overnight for a full charge in the morning, the system will have an inductive charge system for contactless charging. This is an optional design to the system and was added in as we felt that there was enough time to explore the opportunity.

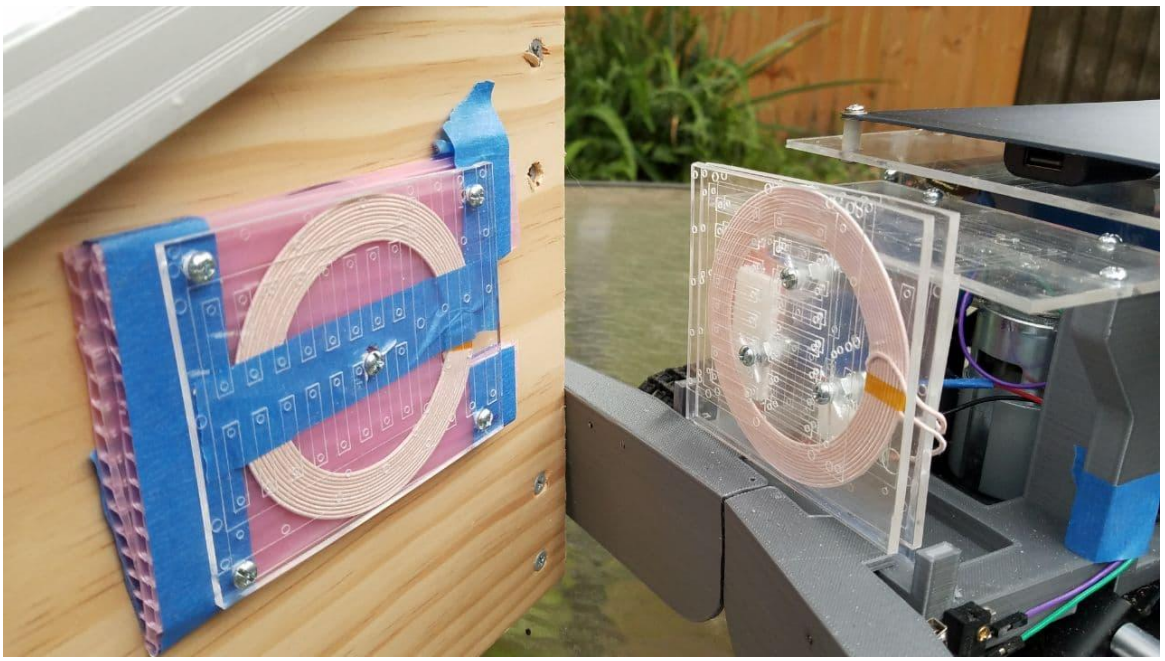

Figure 3.23 Inductive Charge Coils on the Docking Station and Mower

The system has been through two different types of inductive coil systems, one that only needs a 12V input source and outputs at 5V and another version that claims it can use 12V to 24V input and outputs 5V. Each system has its benefits and drawbacks as it relates to the existing system as shown below in table 3.2.

Table 3.2 Inductive Coil Comparison

Inductive Coil Comparison		
Specs	**Taidacent 12V to 24V Input, 5V 2A Output**	**Taidacent 12V Input, 5V 1A Output**
Output Voltage with 12V Input	2.5V	5V
Minimum Usable Distance	15 mm	15 mm
Distance @ 500mA Transferred	40 mm	25 mm
Maximum Power Transfer	2A (only at 24V input)	1A

While the 12V to 24V input inductive coil system provides more transmission distance, the manufacturer failed to mention that this measurement only applies if 24V is

applied to the system. Not only that, the voltage output will only reach 2.5V with a 12V supply and the current will be far less. We attempted to boost this voltage by utilizing voltage regulators, and this allowed us to make use of the system.

The first coil that was used was the 12V input, 5V 1A output system. We were able to get a current draw of 0.6A to the mower at a distance between 15 mm and 20 mm as shown below in Figure 3.24.

Figure 3.24 Current Draw at 40% Battery with the 12V input, 5V 1A Coil

After the system fully charges, the system seems to only pull a small amount of current, about 10mA, but this could be reflective of the way LiFePo4 batteries charge and the charge controller is limiting how much current is being consumed at 80% to 100% charge. More testing will need to be done at various battery levels, but the team did not have time to accomplish these readings.

3.5.4 Tank Treads and Turning

While the tank treads of the mower are very effective at moving forward in tall grass, the issue we ran across is its ability to turn on the spot in tall grass. See Figure 3.25 below for reference to mower size and grass height.

Figure 3.25 Grass height interfering with mower turning

While typical round wheels may only make contact with the ground in a small area, this allows the system to pivot and turn on the spot without the rest of the wheel creating extra friction on the ground and preventing the unit from turning.

Our tank treads will typically have more contact with the ground and will sometimes prevent turning if the treads get stuck on tall clumps of grass further away from their central pivot point. This can lead to issues with turning on the spot and aligning the compass for direction. See figure 3.26 below for a graphical comparison of surface friction areas between tires and tank treads.

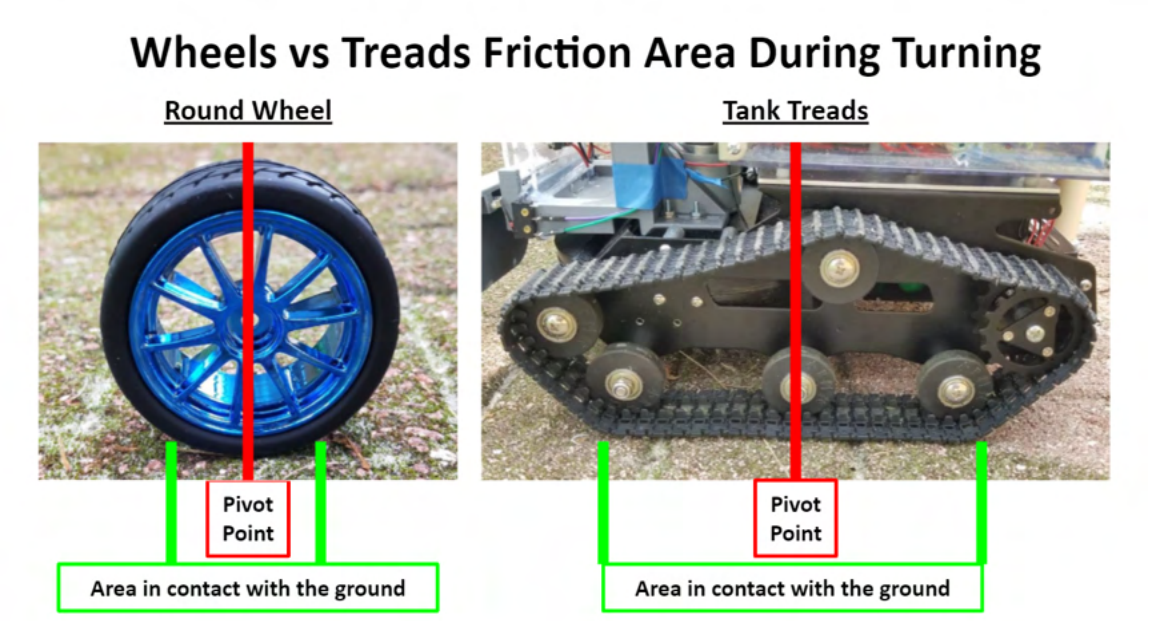

Figure 3.26 Wheels vs Tank Treads friction area

This can be resolved by utilizing PWM on the drive motor controls to turn the tank while moving forward and backward instead of having the right and left treads rotating in opposite directions to turn. Using 100% duty cycle on one side and 50% duty cycle on the other while both are moving forward causes both tracks to rotate at different speeds and will result in the mower turning while moving forward.

Table 3.3 Turn on Grass Success Rate

Test Number	Turning Test Result
1	Failed
2	Passed
3	Failed
4	Failed
5	Failed
6	Failed
7	Failed
8	Passed
9	Passed
10	Failed
Average Turn Success Rate on Grass = 30%	

3.5.5 Compass Interference

Compasses are made to pick up the earth's natural magnetic fields, which allows them to determine what direction they are pointing. However, magnetic interference either by other magnets, coils of wire, and even connectors that can be magnetized will cause inaccurate and varying compass results.

While designing the mower layout, magnetic devices like motors and charging coils had to be accounted for. Not only that, magnets that were not being used in the GPS's GNSS antenna had to be removed from the system to prevent interference. Finally, we had to make sure any connectors and pins leading up to the compass were non-ferrous as they would also interfere with the compass readings.

To resolve the issues of interference, the compass was raised far above the mower itself and is far enough from any magnetic interference produced by the cutter blade and drive motors to provide accurate readings.

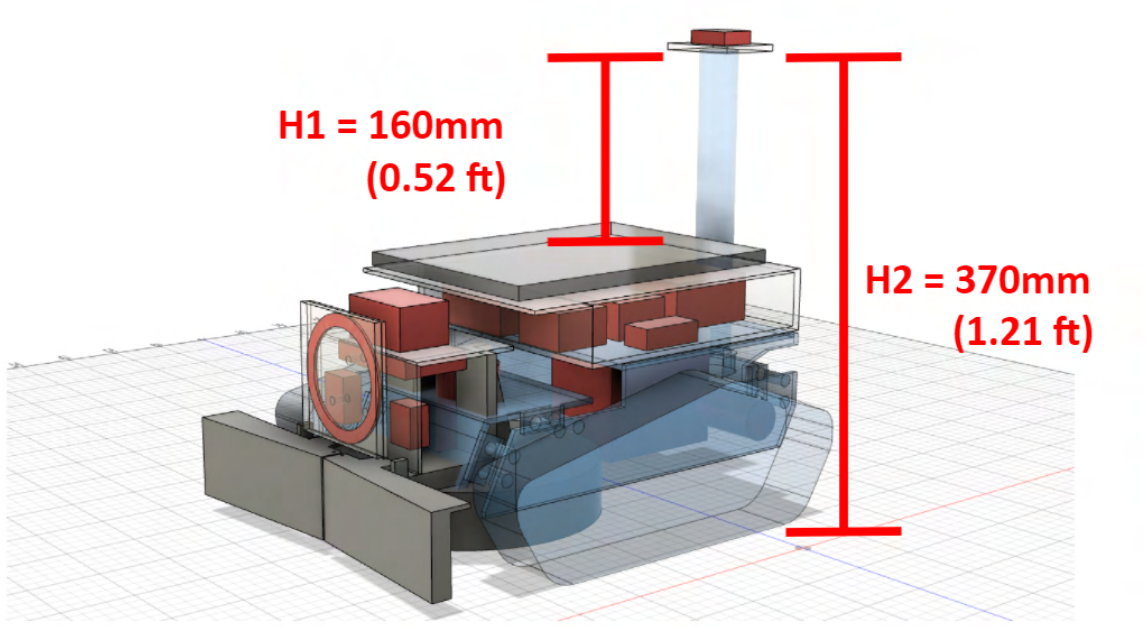

Figure 3.27 Compass Height

3.5.6 Solar Charge Controller Settings

The solar charge controllers are designed to provide an automated solar charging system for the attached batteries. They will supply power to the batteries once power is produced by the solar panels, and will follow the attached batteries charging profile until fully

charged. Not only that, they have the ability to manage current to an attached load if the battery voltage drops too low.

To make the system as efficient as possible, we attempted to utilize another feature provided by the charge controllers; to only supply power for several hours once there is enough sunlight to start charging the battery.

In theory, this is a useful option but the current weather patterns experienced during our last month of testing showed that this isn't a viable option for the system. Due to constant rain and cloudy days, the controller wouldn't receive enough power from the panels to turn on and would inexplicably turn on at random times during the night. This could have been an issue with the controller, the sensitivity settings, or the solar panel, but we decided to not use this feature and have 24-hour power to the system and let the Witty Pi 3 manage the bulk of the power drain.

3.5.7 Charging Times During Bad Weather

Due to solar charging limitations from lack of sun, we had to devise a way to provide some sort of light energy with enough power to test the solar panel charging times and angles. This was resolved by using a 500W halogen work light and held about 12 inches away from the solar panel as shown in figure 3.28 below.

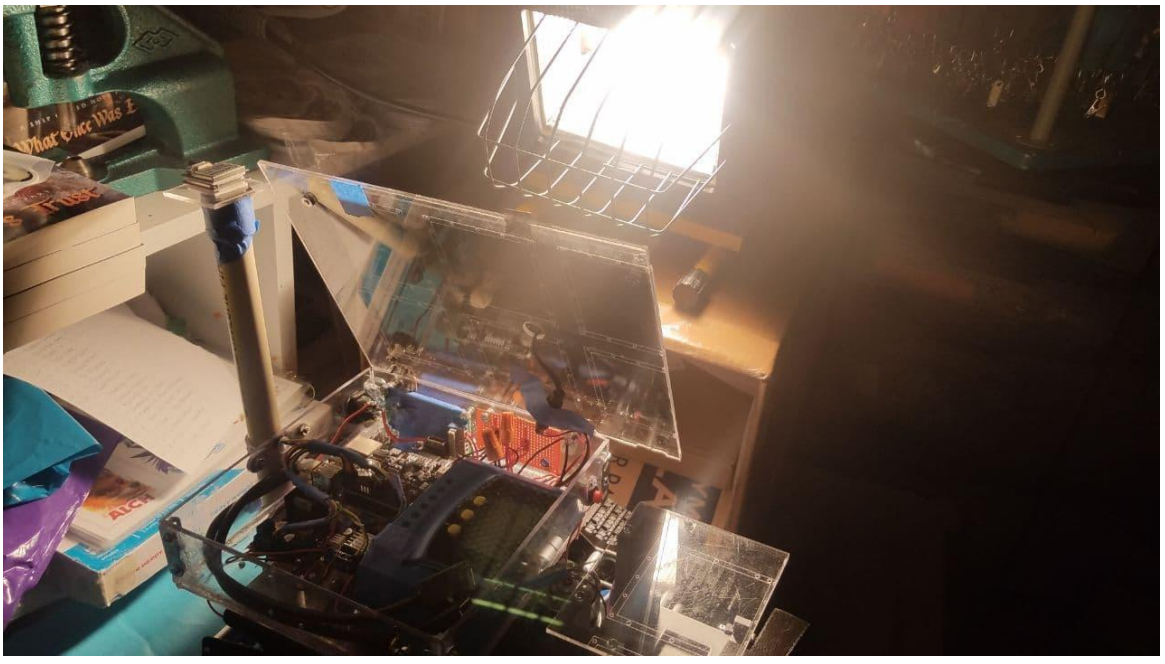

Figure 3.28 Solar Charging Tests with a 500W light

This proved to be an effective temporary replacement as it was able to produce similar results with the solar panels in direct sunlight. The mower was able to take in 0.25A in direct light from the 500W bulb, which was just a little less than the 0.3A max achieved in direct sunlight outside.

This was only good as a temporary solution as the heat generated by the 500W light is substantial and can heat up the mower and docking station. We decided to not run the light in more than 30 min intervals and only if someone is continuously watching it.

3.5.8 Drive Motor Voltage

While the mower was in motion, we noticed that there was a slight pull to the right over long distances. This led us to see how much power the L298N dual h-bridge motor controller was sending to the drive motors. Since the L298N has its own internal load via the voltage regulator to convert power to the logic controls, there is a slight voltage drop. This isn't true for the VNH5019 single bridge motor controller as its logic circuit is powered by the Raspberry Pi and provides a 1:1 input/output voltage ratio.

As you can see in figures 3.29 and 3.30, the right drive motor receives slightly less power to the tracks at the same PWM frequency and duty cycles as the left drive motor, resulting in a small pull to the right.

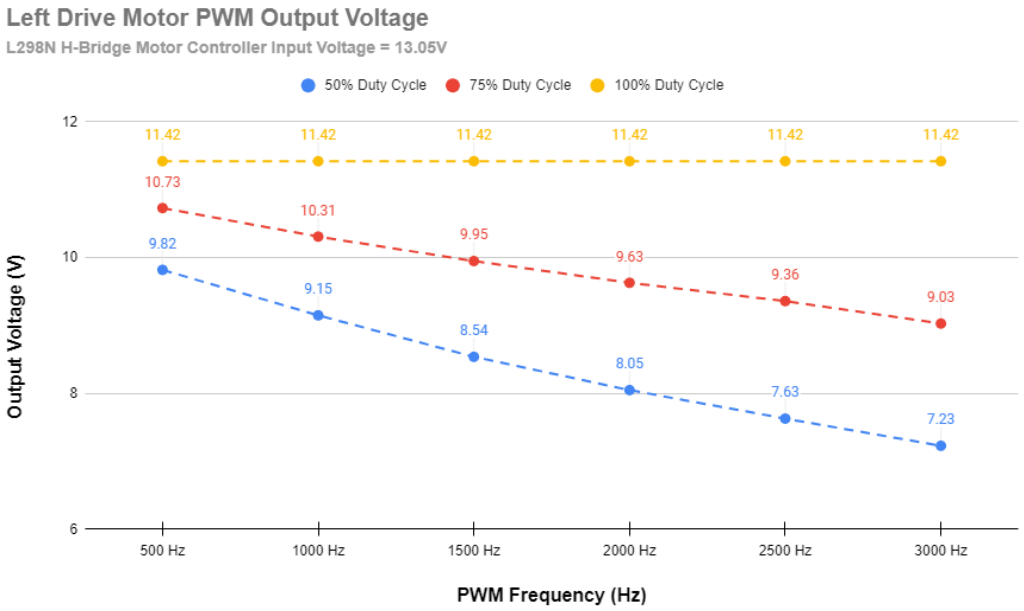

Figure 3.29 Left Drive Motor PWM Output Voltage

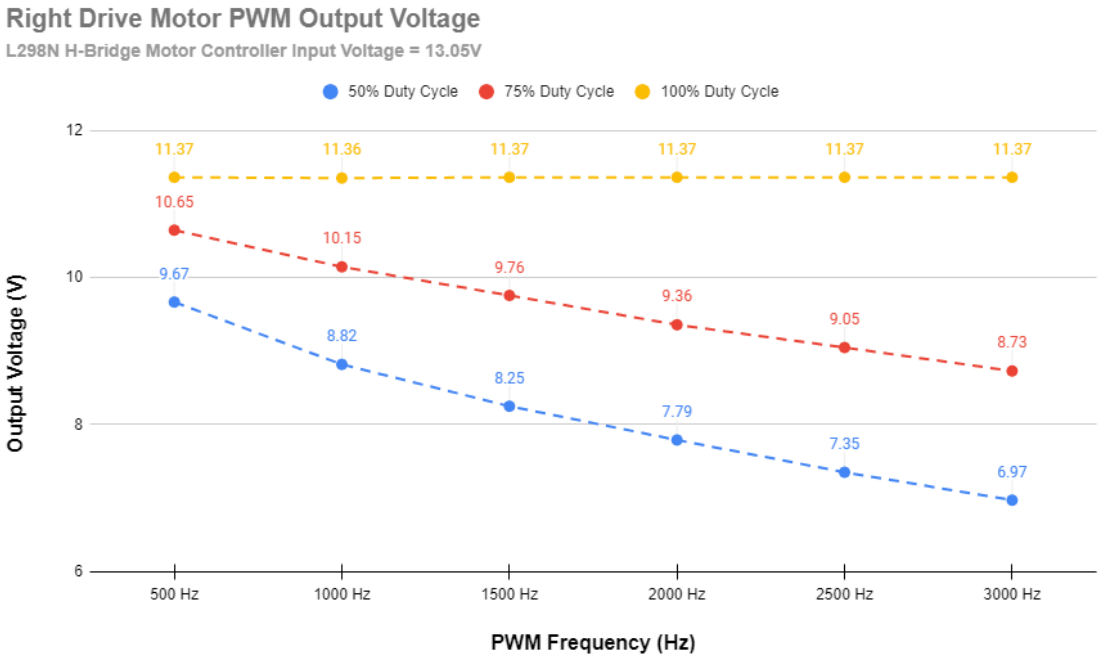

Figure 3.30 Right Drive Motor PWM Output Voltage

3.5.9 Team Testing Limitations due to COVID

One of the biggest hurdles we had to overcome at the beginning of our project was social distancing with COVID until we were able to get vaccinated. To resolve this, we decided a second test tank was needed so team members could remotely test various aspects of the system like the RTK modules. The end result was a smaller test robot that can do the majority of the tasks the mower unit can do. It is a smaller unit and does not have a cutter blade, but has all the mobility of the mower in ideal situations and can be used to verify directional and movement automation.

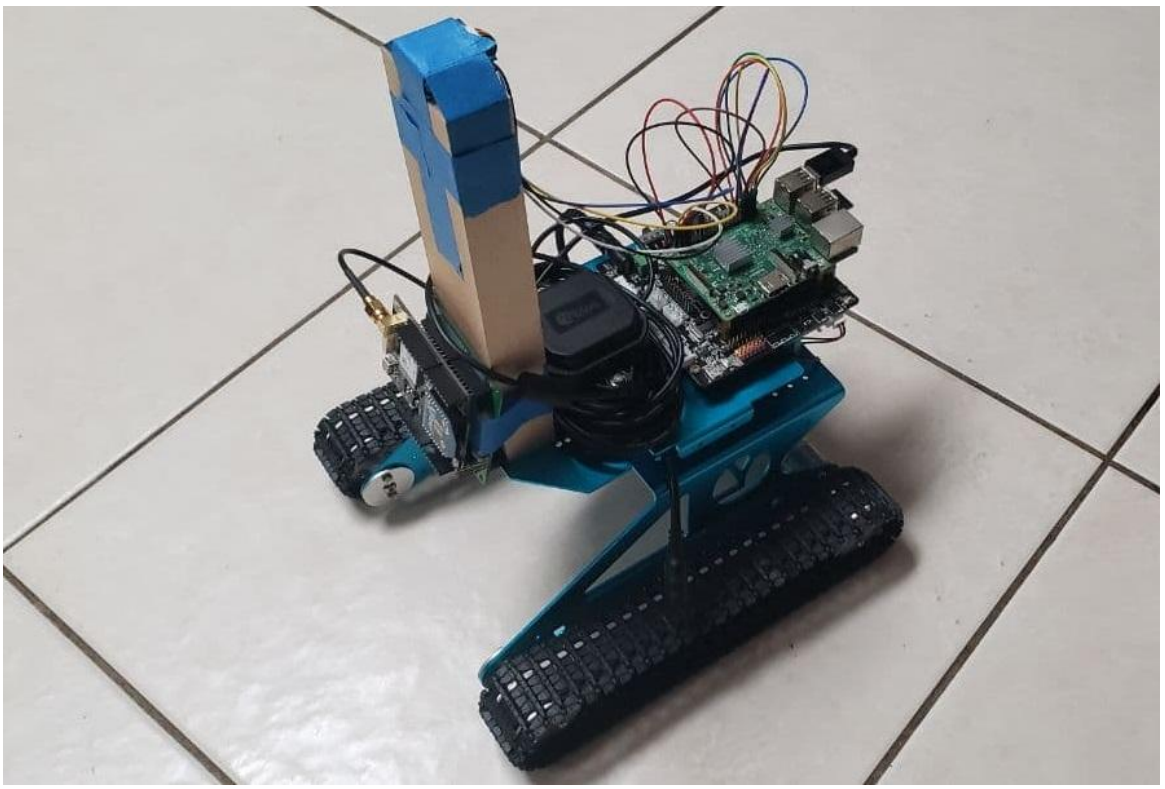

Figure 3.31 Test Tank for Individual Testing at Home

3.6 Testing and Evaluation

All the individual components of the mower were tested in real-world scenarios to determine their reliability and success rates.

3.6.1 Solar Charge Testing

One of the most crucial design elements of the mower is its solar recharging system. However, due to the poor weather and lack of sunlight available due to multiple storms, full testing of the solar recharging system has been limited. The combination of low sunlight and rain potentially damaging the system in its current configuration was a constant hurdle to overcome and reduced nearly 90% of testing availability.

Even with the poor weather, we were able to determine through stop and go testing that the mowers LiFePo4 battery is capable of achieving a full charge from a dead battery state (less than 10V) to a full charge (13.4V) within 3 average days without rain and mostly clear skies. Not only that, the system is able to at least maintain the battery level in poor weather with rain and overcast skies, as shown in figure 3.32 below.

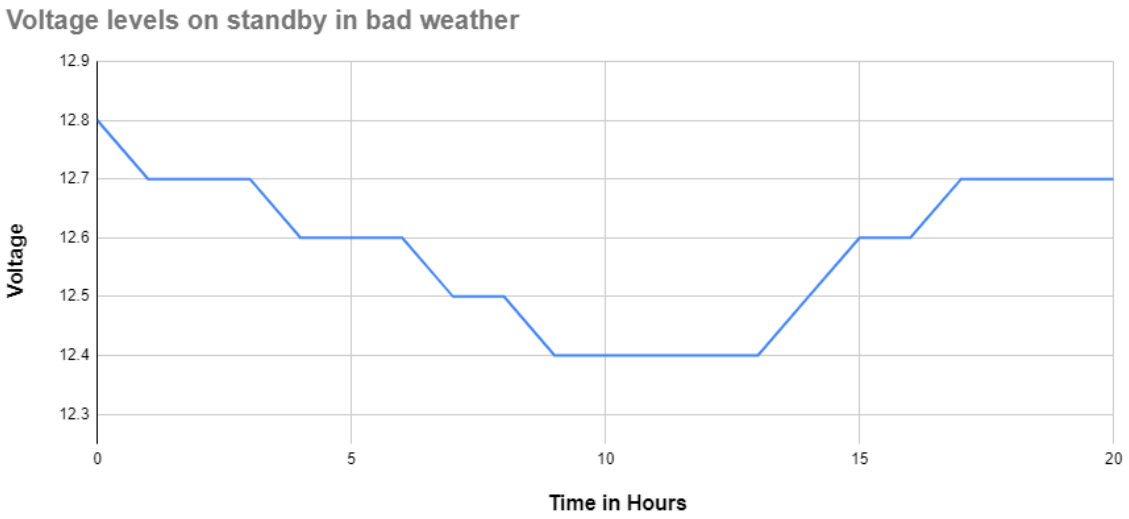

Figure 3.32 LiFePo4 Battery Charging During Poor Weather

Since the solar panels on both the mower and docking station are stationary and do not track the sun, their power output is directly related to the angle of the sun in relation to the face of the solar panel. As you can see in figure 3.33 below, the most power transferred was when the sun was at a direct 90-degree angle to the surface of the solar panel.

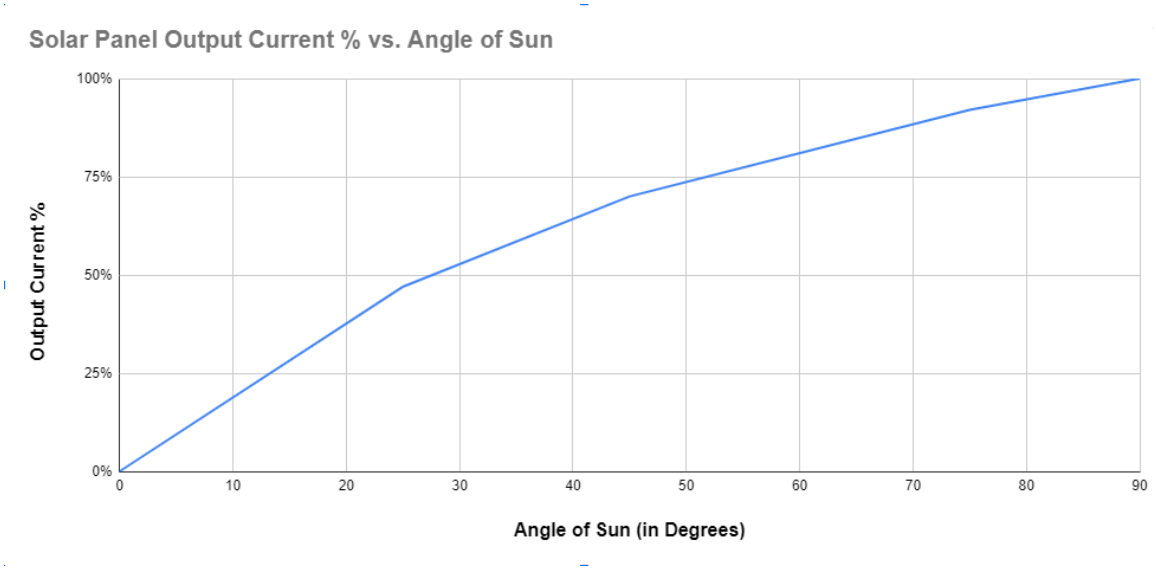

Figure 3.33 Solar Panel Output Current % vs. Angle of the Sun

3.6.2 Grass Cutting Height

The cutting blade voltage was set to 7.5V to the 775 cutter motor, which provided us with a rotational speed of 6,350 RPMs and proved to be an acceptable speed to efficiently cut long grass up to 5” in length. The mower can easily cut through grass under 2.5” without slowing down the cutter blade, making it ideal for weekly or bi-weekly lawn maintenance. Figure 3.34 below shows before and after pictures of grass averaging from 5” cut down to 1.5” after the mower has cut it.

Figure 3.34 5” Grass Before and After Cutting

3.6.3 Battery Life

Both movement tests and cutting while moving tests were done manually several times each to determine both what was going to be the main draw on the power and how long the system could effectively run for starting from a full battery charge (13.4V). The end results showed that without the cutter blade running and the mower in direct sunlight, the system could effectively run for 3-4 hours without depleting the battery below 12V. With the cutting blade active and moving around, the system could run for an hour in ideal conditions, depending on grass height and thickness with the shortest run time being 35

minutes. Due to charging times and limited available trial opportunities due to time availability and weather, only 4 trials were performed.

Table 3.4 Mower Run Times while Cutting

Trial	Active Time	Notes
1	47 minutes	Slightly cloudy skies, tree shade in some spots, tall grass
2	35 minutes	Cloudy skies, tree shade, tall grass
3	62 minutes	Full sunlight conditions, medium grass
4	51 minutes	Full sunlight conditions, medium to tall grass

3.6.4 GPS Measurements

Table 3.5 GPS Distance Measurements

Distance Between two Coordinates	Coordinate 1	Coordinate 2	Difference
1 Foot	-81.23564216	-81.23564524	3.08e-6
4 Feet	-81.23564231	-81.23565471	1.24e-5
10 Feet	-81.23564225	-81.23567248	3.023e-5

Tests were performed to determine how the values of the GPS coordinates would change over specifically measured distances. After performing several tests at different distances it was determined that every foot was changed the coordinate value by about 3e-6. This value was consistent over several tests as moving the GPS by 4 feet yielded three times the value and moving it by 10 feet yielded ten times this value. This information was used in the code to let the user pick any distance in feet for the dimensions of their lawn. Any distance they choose is simply multiplied by 3e-6 and then fed to the GPS.

3.6.5 Return Home

The return home procedure begins when the mower detects that it has reached the top border of the lawn. Without the RTK lock safety feature, the success rate of this procedure was only 4 in every 10 trials. However, once the safety feature was implemented the success rate increased to 10 and every 10 trials. Because the mower was only moving when it confirmed that the RTK lock was active, this guaranteed that it always had precise GPS accuracy to guide it back to the docking station. The only problem was that the return home procedure could be stopped if the RTK lock was lost during the procedure, but trading speed for reliability was definitely worth it. The only part that requires improvement is getting the mower to fully dock. Unfortunately, even with an RTK lock, the mower isn't precise enough to align itself perfectly with the inductive coils. RTK is known for its centimeter accuracy with an error range of about 3 cm, but even being just 3 cm off is too much to achieve perfect docking. There is also the problem of the mower stopping in time, even if the mower detects it is at the correct location to stop, the mower’s momentum still causes it to move slightly forward which can add a few more inches of inaccuracy to the docking.

Table 3.6 Docking Success Rate

Docking Tests	Results
1	Failed
2	Failed
3	Failed
4	Passed
5	Failed
6	Failed
7	Failed
8	Passed
9	Failed
10	Failed
Average Docking Success Rate = 20%	

3.6.6 Border Detection

Border detection is used to move the mower from one side of the lawn to the other. The precision of the RTK is crucial to prevent the mower from going past the borderline. Originally the success rate was very low (3 in every 10 trials) due to the lack of stability from the RTK, but once again thanks to the RTK lock safety feature the reliability increased significantly (9 successes in every 10 trials). The only misses were due to the momentum of the mower causing variation on how close to the border it stopped, but this can be addressed in the initial setup by giving the borders a bit of error room to give the mower time to stop.

Table 3.7 Border Detection Success Rate

Border Detection Tests	Results
1	Passed
2	Passed
3	Passed
4	Passed
5	Passed
6	Passed
7	Failed
8	Passed
9	Passed
10	Passed
Average Border Detection Success Rate = 90%	

3.6.7 Obstacle Sensing

The obstacle sensor was an important safety feature to prevent damage to and from the mower. The reliability of the obstacle sensor was very high due to it being a closed-off system that didn't rely on the GPS. The main concern for the obstacle sensor was if it would avoid obstacles in the correct direction. Because the behavior of the obstacle sensor was coded for every different scenario where it could bump into something, the obstacle sensor was able to avoid all obstacles successfully. The only issue encountered was with lightweight obstacles that wouldn't provide enough force to trigger the bumper sensors but would instead be pushed aside or dragged in front of the mower. That being said this was a very rare occurrence and the obstacle sensor performed as expected in the vast majority of tests (9 successes in every 10 trials).

Table 3.8 Obstacle Avoidance Success Rate

Obstacle Avoidance Tests	Results
1	Passed
2	Passed
3	Passed
4	Failed
5	Passed
6	Passed
7	Passed
8	Passed
9	Passed
10	Passed
Average Obstacle Avoidance Success Rate = 90%	

3.6.8 Mower Autonomy

Testing the autonomy of the mower involved all the previous tests, as well as testing if the mower could start up and shut down on its own while remembering the variables given by the user during the initial setup. Because the user-provided variables are saved onto a text file, the variables cannot be overwritten or erased unless specifically done so by the user. As for starting up and shutting down on its own, the WittyPi was able to start the mower automatically at the specified time without issue which allowed the mower to immediately begin mowing as soon as it acquired an RTK lock. The mower's ability to send an email notification to the user when the RTK lock was lost for over 20 minutes worked as expected. Finally, the mower’s automatic shutdown was triggered successfully during trials whenever it returned to the docking station and whenever it detected an RTK lock loss for over 30 minutes.

Chapter 4

Non Technical issues

Summary

This chapter goes over the budget and timeline of the project, as well as several aspects of the mower including environmental aspects, safety aspects, ethical aspects, social aspects, and sustainability of the project.

4.1 Budget and Timeline

Our original budget didn't account for some of the serious hardware requirements for useability and had some parts that were found to be obsolete as we continued to research and learn more about the capabilities of each component. The bulk of the cost was originally with the RTK GPS kit, Raspberry Pi, and tank chassis as shown below.

Table 4.1 Original Budget

Description	Qty	Price	Total price
Raspberry pi	2	$61.88	$123.76
Car Kit	1	$79.00	$79.00
Solar Panels	2	$25.99	$51.98
Cutting Blade	1	$5.68	$5.68
Battery	2	$21.99	$43.98
Power Converter	2	$14.99	$29.98
Humidity Sensors	1	$11.98	$11.98
Push Button	2	$7.99	$15.98
3D Printed Parts	3	$10.00	$30.00
Blade Motor	1	$29.95	$29.95
RTK Modules	2	$325.00	$650.00
GPS Module	1	$18.59	$18.59
Total	20	$613.04	$1,100.04

After the team started to test equipment and determine real-world functionality and how the systems will fit in the system, the overall budget requirements grew. Some of the additional costs were in hardware and the optional wireless charging modules as seen in the updated budget below.

Table 4.2 Updated Budget

Description	Qty	Price	Total price
Raspberry Pi 3B+	2	$38.18	$76.36
Witty Pi 3	1	$35.00	$35.00
Car Kit	1	$79.00	$79.00
L298N Motor Controller	1	$6.99	$6.99
VNH5019 Motor Controller	1	$11.99	$11.99
DC-DC Buck Converter 10A	1	$10.99	$10.99
Solar Panel 10W 12V (Flexible)	1	$11.39	$11.39
Solar Panel 10W 12V (Polycrystalline)	1	$24.99	$24.99
Solar Charge Controller (Mower)	1	$24.99	$24.99
Solar Charge Controller (Docking Station)	2	$9.99	$19.98
Mower Battery (12V 6Ah, LiFePo4)	1	$39.99	$39.99
Docking Station Battery(12V, 9Ah, SLA)	1	$21.99	$21.99
XH2.54 Connector	1	$1.80	$1.80
ON/OFF Power Button (12V, 20A)	2	$9.16	$18.32
Bumper Sensor	2	$0.40	$0.80
3D Printed Parts	3	$10.00	$30.00
Blade Motor Kit	1	$29.95	$29.95
Compass Module	1	$18.59	$18.59
simpleRTK2B Kit	1	$657.80	$657.80
Female Standoff (Widening Kit)	12	$0.88	$10.56
6-32 Screws qty 100 (Widening Kit)	1	$3.56	$3.56
6” x 12” 6061 Alum Sheet (Widening Kit)	1	$12.20	$12.20
Nylon Spacers qty 100	1	$9.84	$9.84
12” x 12” Polycarbonate Sheet	2	$6.21	$12.42
Wood for Docking Station	1	$10.00	$10.00
Fastening Hardware Kit	1	$12.00	$12.00
Inductive Charging Coil Kit	1	$22.48	$22.48
DC-DC Step Up Buck Converter	1	$3.33	$3.33
Touchscreen Controller Kit	1	$50	$50
		Total =	$1,267.59

We were mostly able to keep on track with the timeline originally submitted. The coding started later in the process due to research required beforehand.

Figure 4.1 Timeline

4.2 Environmental Aspects

The mower and docking station runs on batteries which if disposed of incorrectly could be hazardous to the environment. Over time if the contents of the battery were to leak it could cause damage to water supplies or the soil. Another important factor is that many of the components of the mower are constructed with plastic which does not deteriorate easily over time. To prevent environmental pollution, it is recommended that these plastics be recycled or disposed of appropriately. However, the mower does help the environment on a small scale by generating its own power with renewable resources as well as maintaining the grass health due to regular mowing and by preventing cross contamination of lawn pests from lawn services.

4.3 Health and Safety

To mow the grass, the mower requires a sharp blade rotating at a high speed. Because of its sharpness, this blade could cause bodily harm if the mower were to be mishandled. To combat this the mower was designed with a shield surrounding the blade, the shield prevents anything from slipping under the blade while the mower is active. In addition to this, it was decided that obstacle sensors would be another effective safety feature. This sensor would not only help avoid injury by steering the mower away from any non-grass object but would also protect the blade itself from being damaged against something it wasn't meant to cut. Finally, there is a cutter blade power cutoff switch that will render the cutter blade inoperable if turned off.

Because the blade needs to be rotated at a high speed it was necessary to acquire an adequately powerful battery. The power budget table shows how all the electric

components do not surpass the max capacity of the battery. Surpassing the max capacity of the battery could lead to electrical failures as well as burning and even fires in extreme scenarios. It is recommended that the lawn be as debris-free as possible to avoid unnecessary risks. This device should never be operated by children and should not be allowed near pets.

4.4 Ethical Aspects

The ethical goals of this project are to give an effortless way to mow the lawn to those who are physically incapable of doing so including the elderly and the differently-abled and to help avoid yard pest contamination from lawn services. Having pests puts the individual at risk of contracting different diseases they might carry, and this also applies to anyone living nearby who could be affected by these pests. This also helps improve the look of the neighborhood since nobody wants to live next to the neighbor who ignores his lawn.

4.5 Social Aspects

The social goal of this project is to provide people with a reliable and efficient way to mow their lawn while taking away most if not all the effort typically involved in doing so. This project does not only help people searching for a simple quality of life improvement, but it also helps those who are unable to perform this strenuous physical activity. Having a regularly mowed lawn also helps avoid pests as well as the issues that they bring. Regular mowing is also known to help keep grass healthier and to keep weeds at bay.

4.6 Sustainability

The mower is expected to last at least 45 minutes on a single charge, but because the mower is constantly charging via the solar panel this time can vary depending on the exposure to sunlight. It is recommended that the mower be cleaned from time to time to combat wear and tear, and it is not recommended that the mower operate in high humidity conditions.

Chapter 5

Conclusion

Summary

This chapter discusses the summary and conclusion for the mower project as well as the suggestions for future work which could further improve the project based from our research.

5.1 Summary and Conclusion

All aspects of the mower were tested for reliability and accuracy. Almost all of the components of the device are working as expected and are reliably completing their assigned task. The GPS is able to detect lawn borders to signal to the mower when to turn or when to return home. The compass is able to ensure the mower is facing the correct direction before moving to make sure it stays on the correct path. The obstacle sensor is able to reliably detect when it hits an obstacle and dodge in the correct direction to make sure it doesn't accidentally pass a border. The controller is able to manually steer the mower if the user decides it is necessary.

The only parts of the mower that would require some improvement are the treads, the GPS signal consistency, and the refinement of the user interface. As it stands the mower has no problem moving backward or forward through the grass, but it does struggle to turn in place due to the high level of friction between the treads and the grass. Thanks to the RTK lock safety feature the chances of the mower passing a border due to loss of GPS accuracy are very low, but the RTK lock is lost more often than we would like. This causes the mower to be safe from accidental border crossing but overall less efficient as it takes longer to mow the lawn due to having to wait for an RTK lock before moving.

Finally, while the user interface does allow the user to call control the mower manually and set different variables to determine the mower's pathing, the experience could definitely be improved to be more user-friendly. As it stands the controller options are somewhat limited, only allowing for directional control and a kill switch, and the user setup does require the user to input each variable one by one to get the mower working. Overall the project's functionality is acceptable but a little more refinement would be welcome.

5.2 Suggestions for Future Work

While developing the system, there were several issues or upgrades that we discovered and would need improvement for improved use. Some of these ranged from unseen variables in hardware to design implementations we couldn't integrate in time.

5.2.1 Larger Mower

The overall design of the mower itself was to be made as a proof of concept for the idea of an automated lawn mower. As the project mower has about ¼ the diameter of a standard push mower, a more ideal system for lawn care will need to be larger. This will help compensate for any turning issues with smaller wheels, size restrictions, and cutting time.

5.2.2 Round Wheels

For the mower to be more responsive while turning in the grass, it would be ideal to use large, round wheels with spikes or aggressive treads instead of the track system currently used on the mower. This will help with maneuverability and fine-tuning direction of the system.

5.2.3 24V Docking Station Power Supply

While the docking station's 12V power supply will be sufficient to use the inductive charge system as it sits now, it would be ideal to upgrade the docking stations solar panel(s), charge controller, and battery to operate on a 24V system to allow for a better inductive charge system to transmit power at longer ranges. With the 12V system, we can only expect a maximum transmission range of up to 30 mm to 40 mm and a typical usable range between 20 mm and 25 mm.

5.2.4 Larger Solar Panels and Battery Banks

To help offset the limitations of the weather during weeks with little to no sunshine, it would be ideal to have larger solar panels and batteries on both the mower and docking station. Even though it is unnecessary to cut the grass every day, the system does use power on standby and if there isn't enough sunlight for a week straight, it may result in the mower not having enough power to operate when it comes time to start cutting the grass.

An additional bonus to using larger batteries for the mower is the batteries themselves will allow for a higher level of continuous output current and will allow the cutter blade to more easily maintain its rotational speed while cutting.

5.2.5 Custom User GUI

Our GUI that was created for the mower is very simplistic and needs refinement. While it will get the job done, it could benefit from better organization and more intuitive functionality.

5.2.6 More Yard Size Flexibility

Due to time restraints, we are unable to research and create exceptions for different shaped yards. This will be necessary for those who have curved or custom-shaped yards.

5.2.7 Camera Interface

Having some sort of camera interface will allow the user to control the mower in the comforts of their own home without ever having to go outside. This will require internet connectivity and a better microprocessor than the Raspberry Pi 3B+ may be able to offer.

5.2.8 Machine Learning

Originally we had the idea of making the mower learn new rules and exceptions in the event of continuous obstacle avoidance in an area as well as more efficiently cutting grass in an area. This is enough for a complete project on its own and was set to the side to focus on the primary requirements of the project. Integrating machine learning will require an upgrade to the Raspberry Pi 3B+ to either the Raspberry Pi 4 or NVIDIA Jetson Nano for a more capable processor.

5.2.9 Long Range RF Communication

Due to cost restrictions, the RTK modules are using a shorter range RF module to communicate with each other. This will limit the mower and docking station to only be able to send correction data over the distance of a typical yard without walls or heavy trees blocking reception. There are more powerful RF modules available, but they would substantially increase the cost of the unit.

5.2.10 Humidity Sensors

Our original design had humidity sensors that can be integrated into the system, but due to time limitations, we decided to not install them to focus on the more important system requirements. These sensors will be needed to prevent the mower from cutting during a rainstorm or if the ground is still wet.

5.2.11 Drive Motor RPM

The motors that came with the Tank Chassis are 12V brushed motors that come with a 350 RPM gearbox. This has shown to be a little bit too fast for the intended design. We may want to consider between 150 and 200 RPMs for a slower pace at max input voltage to achieve more precision in turns and allow the cutting blade time to cut the grass without skipping spots.

5.2.12 Nighttime Friendly Cutter Blade

One of the most important aspects of the project is the accuracy of the GPS. The difference in accuracy between nighttime operation and daytime operation was very significant. Therefore it would be recommended to have the mower run at night, but this will require a silent cutting blade to not disturb the neighbors during these hours.

5.2.13 More Accurate Docking Technology

The RTK simply does not possess the accuracy required to complete a successful docking consistently. Infrared technology has been used by other similar projects to aid with docking successfully and would be recommended as a good improvement to the mower.

References

[1] "Getting Started With Mowbot", MowBot.com [Online]. Available at: https://www.mowbot.com/how-does-it-work/ [Accessed: 05/Feb/2021].

[2] "Automower Models", Husqvarna.com [Online]. Available at: https://www.husqvarna.com/ca-en/products/robotic-lawn-mowers/models/ [Accessed 05/Feb/2021].

[3] "MowRo - Easy, Safe, Fully Autonomous Lawn Mower", indiegogo.com [Online]. Available at: https://www.indiegogo.com/projects/mowro-easy-safe-fully-autonomous-lawn-mower#/ [Accessed 05/Feb/2021].

[4] "simpleRTK2B: the first multiband RTK shield based on ZED-F9P", Ardusimple [Online]. Available at: https://www.kickstarter.com/projects/simplertk2b/simplertk2b-the-first-multiband-rtk-shield-based-o [Accessed 20/Mar/2021].

[5] Nathan Seidle, "What is GPS RTK?", [Online] Available at: https://learn.sparkfun.com/tutorials/what-is-gps-rtk/all [Accessed 20/Mar/2021].

[6] wx4cb, "mwp_at", [Online] Available at: https://github.com/wx4cb/mwp_at

[7] Clive Turvey, "TestGPSRTK.py", [Online] Available at: https://github.com/cturvey/RandomNinjaChef/blob/main/TestGPSRTK.py

[8] "Raspberry Pi G1 Tank", [Online] Available at: http://www.yahboom.net/study/G1-T-PI

Appendix A

Equations

Formula for Electric Power:

$$K_{\epsilon} \; \aleph \; \mu \; \Re \qquad \text{(1)}$$

Formula for Charging Current

$$\text{Ib}_{\epsilon} \; \psi\text{é} \; \mu \; \hbar \lrcorner \qquad \text{(2)}$$

Appendix B

Programming Code

Summary

In this Appendix we present the code for the automatic mowing, the GUI, the compass, the GPS, and the saving of user provided variables.

B1 Automatic Mowing Code
B2 GUI Code
B3 Compass Code
B4 GPS Code
B5 Save Variables Code

B1 Automatic Mowing Code

```
import smbus            #import SMBus module of I2C

from time import sleep  #import sleep

import math

import serial

import sys

import math

import RPi.GPIO as GPIO

import time

import smtplib

smtpUser='your email'

smtpPass='your email password'

toAdd= 'your email'

fromAdd=smtpUser


subject= 'PALM Mower Stuck'

header= 'To:' + toAdd+ '\n' + 'From: ' +fromAdd + '\n' +'Subject: ' + subject

body= 'PALM Mower is Stuck Please Help'


st=smtplib.SMTP('smtp.gmail.com',587)
```

```
l=open('/home/pi/Desktop/PALM/save.txt','r')


xf=0

yf=0

fangle=0

langle=0

bangle=0

rangle=0

H=0

W=0


stuck=0


for line in l:
    if xf==0:
        xf=line
    elif yf==0:
        yf=line
    elif fangle==0:
        fangle=line
    elif langle==0:
```

```
            langle=line
        elif bangle==0:
            bangle=line
        elif rangle==0:
            rangle=line
        elif H==0:
            H=line
        elif W==0:
            W=line
    x0=float(xf)
    y0=float(yf)
    afront=int(fangle)
    aleft=int(langle)
    aback=int(bangle)
    aright=int(rangle)
    FEETH=float(H)
    FEETW=float(W)


    #Definition of  motor pin
    IN1 = 20
    IN2 = 21
    IN3 = 19
```

```
IN4 = 26

IN5 = 23 #Cutter blade

IN6 = 24 #Cutter blade

ENA = 16

ENB = 13

ENC = 25 #Cutter blade


#Set the GPIO port to BCM encoding mode

GPIO.setmode(GPIO.BCM)


#Ignore warning information

GPIO.setwarnings(False)


global pwm_ENA

global pwm_ENB

global pwm_ENC

global delaytime

GPIO.setup(18,GPIO.OUT)

GPIO.output(18, GPIO.HIGH)

GPIO.setup(ENA,GPIO.OUT,initial=GPIO.HIGH)

GPIO.setup(IN1,GPIO.OUT,initial=GPIO.LOW)

GPIO.setup(IN2,GPIO.OUT,initial=GPIO.LOW)
```

```
GPIO.setup(ENB,GPIO.OUT,initial=GPIO.HIGH)

GPIO.setup(IN3,GPIO.OUT,initial=GPIO.LOW)

GPIO.setup(IN4,GPIO.OUT,initial=GPIO.LOW)


GPIO.setup(22,GPIO.IN)

GPIO.setup(27,GPIO.IN)


GPIO.setup(ENC,GPIO.OUT,initial=GPIO.HIGH)

GPIO.setup(IN5,GPIO.OUT,initial=GPIO.LOW)

GPIO.setup(IN6,GPIO.OUT,initial=GPIO.LOW)

#Set the PWM pin and frequency is 2000hz

pwm_ENA = GPIO.PWM(ENA, 1000)

pwm_ENB = GPIO.PWM(ENB, 1000)

pwm_ENC = GPIO.PWM(ENC, 1000)

pwm_ENA.start(0)

pwm_ENB.start(0)

GPS_SERIAL = "/dev/ttyACM0" # TTY for your receiver

GPS_BAUD = 38400


running = True

def backward():

    GPIO.output(IN1, GPIO.LOW)
```

```
    GPIO.output(IN2, GPIO.HIGH)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.HIGH)
    pwm_ENA.ChangeDutyCycle(80)
    pwm_ENB.ChangeDutyCycle(80)

def forward():
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
    pwm_ENA.ChangeDutyCycle(80)
    pwm_ENB.ChangeDutyCycle(80)

def right():
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.HIGH)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
    pwm_ENA.ChangeDutyCycle(100)
    pwm_ENB.ChangeDutyCycle(100)
```

```
def left():
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.HIGH)
    pwm_ENA.ChangeDutyCycle(100)
    pwm_ENB.ChangeDutyCycle(100)


def stop():
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.LOW)
    pwm_ENA.ChangeDutyCycle(50)
    pwm_ENB.ChangeDutyCycle(50)


def truncate(number, decimals=0): # trying to remediate some fugly Python formatting issues
    # Returns a value truncated to a specific number of decimal places.
    if not isinstance(decimals, int):
        raise TypeError("decimal places must be an integer.")
    elif decimals < 0:
        raise ValueError("decimal places has to be 0 or more.")
```

```
    elif decimals == 0:

        return math.trunc(number)



    factor = 10.0 ** decimals

    return math.trunc(number * factor) / factor




def decimalDegrees(coordinates, digits, hemi): # crack NMEA (D)DDMM.mmmmm N/S
E/W format

    parts = coordinates.split(".")



    if (len(parts) != 2):

        return coordinates



    if (digits > 3 or digits < 2):

        return coordinates



    left = parts[0]

    right = parts[1]



    degrees = float(left[:digits]) + ((float(left[digits:digits+2])) / 60.0) + (float("." +
str(right)) / 60.0)
```

```
    if (hemi == 'S' or hemi == 'W'):

        degrees = degrees * -1.0



    return truncate(degrees, 8) # max 8 dp, if a minute is 1852m, a degree is 111120m,
roughly, this is about a cm

Register_A     = 0             #Address of Configuration register A

Register_B     = 0x01          #Address of configuration register B

Register_mode  = 0x02          #Address of mode register



X_axis_H    = 0x03             #Address of X-axis MSB data register

Z_axis_H    = 0x05             #Address of Z-axis MSB data register

Y_axis_H    = 0x07             #Address of Y-axis MSB data register

declination = -0.00669         #define declination angle of location where measurement
going to be done

pi          = 3.14159265359    #define pi value

def Magnetometer_Init():

        #write to Configuration Register A

        bus.write_byte_data(Device_Address, Register_A, 0x70)



        #Write to Configuration Register B for gain

        bus.write_byte_data(Device_Address, Register_B, 0xa0)



        #Write to mode Register for selecting mode
```

```
        bus.write_byte_data(Device_Address, Register_mode, 0)


def read_raw_data(addr):


        #Read raw 16-bit value

        high = bus.read_byte_data(Device_Address, addr)

        low = bus.read_byte_data(Device_Address, addr+1)


        #concatenate higher and lower value

        value = ((high << 8) | low)


        #to get signed value from module

        if(value > 32768):

            value = value - 65536

        return value


bus = smbus.SMBus(1)          # or bus = smbus.SMBus(0) for older version boards

Device_Address = 0x1e   # HMC5883L magnetometer device address


Magnetometer_Init()     # initialize HMC5883L magnetometer

print ("Application started!")

# Open GPS COM Port / TTY
```

```
gps = serial.Serial(GPS_SERIAL, baudrate = GPS_BAUD, timeout = 0.5)


s=4

if __name__ == "__main__":


   while running:


      try:

         data = gps.readline().decode('ascii', errors='replace')


         try:

            message = data[0:6]


            x = read_raw_data(X_axis_H)

            z = read_raw_data(Z_axis_H)

            y = read_raw_data(Y_axis_H)


            heading = math.atan2(y, x) + declination


            #Due to declination check for >360 degree

            if(heading > 2*pi):

                  heading = heading - 2*pi
```

```
#check for sign
if(heading < 0):
        heading = heading + 2*pi

#convert into angle
heading_angle = int(heading * 180/pi)

angle=heading_angle-180

if (message == '$GNGGA'):
    #0 $GNGGA
    #1 gpstime
    #2,3 lat
    #4,5 lon
    #6 fixqual
    #7 satcount
    #...
    parts = data.split(',') # separate NMEA on commas
    # should check len(parts)
    latitude = decimalDegrees(parts[2], 2, parts[3]) # DDMM.mmm
    longitude = decimalDegrees(parts[4], 3, parts[5]) # DDDMM.mmm, decimal
places in degree field, hemisphere
```

```
fixtype = parts[6] # 4=RTK Fixed, 5=RTK Float


x=float(longitude)

y=float(latitude)

z=int(fixtype)









#x0=-81.278383

#y0= 28.5003685

#print(y0)


dx=abs(abs(x)-abs(x0))

dy=abs(abs(y)-abs(y0))

height=FEETH*0.000003

width= FEETW*0.000003

#print(stuck)


if z!=4:

    stop()

    GPIO.output(IN5, GPIO.LOW)
```

```
    GPIO.output(IN6, GPIO.LOW)
    stuck=stuck+1
    sleep(1)
    print('NO RTK')
if z==4:
    GPIO.output(IN5, GPIO.LOW)
    GPIO.output(IN6, GPIO.HIGH)
    pwm_ENA.ChangeDutyCycle(50)
    stuck=0
if stuck==1800:
    GPIO.cleanup()
    from subprocess import call
    call("sudo shutdown -h now", shell=True)
elif stuck==1200: #Send Email
    st.ehlo()
    st.starttls()
    st.ehlo()

    st.login(smtpUser,smtpPass)
    st.sendmail(fromAdd, toAdd, header +'\n\n' +body)
    st.quit()
```

```
if s==0 and dy<=height and dx<=width and z==4:  #FORWARD


    forward()
elif s==0 and (GPIO.input(22) or GPIO.input(27)):
    stop()
    sleep(1)
    backward()
    sleep(1)
    stop()
    sleep(1)
    left()
    sleep(0.3)
    stop()
    sleep(1)
    forward()
    sleep(1)
    stop()
    sleep(1)


    s=2
```

```
elif s==0 and dy<=height and dx>=width and z==4:
        stop()
        s=1


if s==1 and abs(aleft-angle)<=10 and dy<=height and z==4:

    stop()
    sleep(1)
    forward()
    sleep(1)
    stop()
    sleep(1)
    s=2


elif s==1 and dy<=height and z==4:

    left()
    sleep(0.1)
```

```
    stop()
    sleep(1)



if s==2 and dy<=height and z==4:



    if s==2 and abs(aback-angle)<=10  and dx>=0.000003 and z==4:

        forward()



    elif s==2 and (GPIO.input(22) or GPIO.input(27)):
        stop()
        sleep(1)
        backward()
        sleep(1)
        stop()
        sleep(1)
        right()
        sleep(0.3)
        stop()
```

```
        sleep(1)


    elif s==2 and dx<=0.000003 and z==4:
        stop()
        s=3


    elif s==2 and z==4:


        left()
        sleep(0.1)
        stop()
        sleep(1)



if s==3 and dy<=height and z==4:
    if s==3 and abs(aleft-angle)<=10 and z==4:


        stop()
        sleep(1)
        forward()
        sleep(1)
        stop()
```

```
            sleep(1)
            s=4


        elif s==3 and z==4:


            right()
            sleep(0.1)
            stop()
            sleep(1)


    if s==4 and dy<=height and z==4:

        if s==4 and abs(afront-angle)<=10 and z==4:


            stop()


            print('forward angle good')
            s=0
```

```
    elif s==4 and z==4:



        right()

        sleep(0.1)

        stop()

        sleep(1)


if dy>=height and s<5 and z==4:

    stop()

    s=5

    print('done')


#DOCKING


if s==5 and z==4: #Docking turn to front

    if s==5 and abs(afront-angle)<=10 and z==4:

        stop()

        sleep(1)

        s=6
```

```
    elif s==5 and abs(afront-angle)>10 and z==4:
        right()
        sleep(0.1)
        stop()
        sleep(1)
if s==6 and dx<=width and z==4: #Go forward
    forward()
    sleep(1)
    stop()
    sleep(1)
    s=7
elif s==6 and z==4:
    stop()
    sleep(1)
    s=7
if s==7 and z==4: #Turn right
    if s==7 and abs(aright-angle)<=10 and z==4:

        stop()
        sleep(1)
        s=8
    elif s==7 and z==4:
```

```
            right()
            sleep(0.1)
            stop()
            sleep(1)
    if s==8 and z==4: #Go right
        if s==8 and (GPIO.input(22) or GPIO.input(27)):
            stop()
            sleep(1)
            backward()
            sleep(1)
            stop()
            sleep(1)
            left()
            sleep(0.3)
            stop()
        if s==8 and dy>=0.000003 and z==4:
            forward()
        elif s==8 and z==4:
            stop()
            sleep(1)
            s=9
```

```
if s==9 and z==4:
    if s==9 and abs(aback-angle)<=10 and dx>=0.000001 and z==4:
        forward()
        sleep(0.1)
        stop()
        sleep(1)



    elif s==9 and dx>=0.000001 and z==4:
        right()
        sleep(0.1)
        stop()
        sleep(1)
    elif s==9 and z==4:
        stop()
        sleep(1)
        s=10
if s==10:
    GPIO.cleanup()
    from subprocess import call
    call("sudo shutdown -h now", shell=True)
```

```
            except Exception:
                print ("GPS:JUNK")
                stop()

        except KeyboardInterrupt:
            # Do a Ctrl-C in Shell Window, not stop signal
            print ("KeyboardInterrupt");
            stop()
            GPIO.cleanup()
            running = False

        except:
            # You should do some error handling here...
            print ("Application error!")
            gps.close()
            print ("Application closed!")
            exit()

gps.close()
print ("Application closed!")
exit()
```

B2 GUI Code

```
from tkinter import *
import tkinter.font

import RPi.GPIO as GPIO
import time

## Definition of  motor pin ##
IN1 = 20 #Right track
IN2 = 21 #Right track

IN3 = 19 #Left track
IN4 = 26 #Left track

IN5 = 23 #Cutter blade
IN6 = 24 #Cutter blade

ENA = 16 #Right track
ENB = 13 #Left track
ENC = 25 #Cutter blade

#Set the GPIO port to BCM encoding mode
GPIO.setmode(GPIO.BCM)

#Ignore warning information
GPIO.setwarnings(False)

#Motor pin initialization operation
global pwm_ENA
global pwm_ENB
global pwm_ENC
global delaytime

#Movement control
GPIO.setup(ENA,GPIO.OUT,initial=GPIO.HIGH)
GPIO.setup(IN1,GPIO.OUT,initial=GPIO.LOW)
GPIO.setup(IN2,GPIO.OUT,initial=GPIO.LOW)
GPIO.setup(ENB,GPIO.OUT,initial=GPIO.HIGH)
GPIO.setup(IN3,GPIO.OUT,initial=GPIO.LOW)
GPIO.setup(IN4,GPIO.OUT,initial=GPIO.LOW)

#Cutter blade control
```

```
GPIO.setup(ENC,GPIO.OUT,initial=GPIO.HIGH)
GPIO.setup(IN5,GPIO.OUT,initial=GPIO.LOW)
GPIO.setup(IN6,GPIO.OUT,initial=GPIO.LOW)

#Set the PWM pin and frequency is 1000hz
pwm_ENA = GPIO.PWM(ENA, 500)
pwm_ENB = GPIO.PWM(ENB, 500)
pwm_ENC = GPIO.PWM(ENC, 1000)
pwm_ENA.start(0)
pwm_ENB.start(0)
pwm_ENC.start(0)

## GUI definitions ##
# win is the window
win = Tk()
win.title("Movement Controls")
myFont = tkinter.font.Font(family = 'Helvetica', size = 12, weight = "bold")


## Event Functions ##
# Forward
def wToggle(event):
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
    pwm_ENA.ChangeDutyCycle(100)
    pwm_ENB.ChangeDutyCycle(100)

# Left
def aToggle(event):
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.HIGH)
    pwm_ENA.ChangeDutyCycle(100)
    pwm_ENB.ChangeDutyCycle(100)

# Right
def dToggle(event):
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.HIGH)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
```

```
        pwm_ENA.ChangeDutyCycle(100)
        pwm_ENB.ChangeDutyCycle(100)

# Reverse
def sToggle(event):
        GPIO.output(IN1, GPIO.LOW)
        GPIO.output(IN2, GPIO.HIGH)
        GPIO.output(IN3, GPIO.LOW)
        GPIO.output(IN4, GPIO.HIGH)
        pwm_ENA.ChangeDutyCycle(100)
        pwm_ENB.ChangeDutyCycle(100)

# Forward Right
def FRToggle(event):
        GPIO.output(IN1, GPIO.HIGH)
        GPIO.output(IN2, GPIO.LOW)
        GPIO.output(IN3, GPIO.HIGH)
        GPIO.output(IN4, GPIO.LOW)
        pwm_ENA.ChangeDutyCycle(40)
        pwm_ENB.ChangeDutyCycle(100)

# Forward Left
def FLToggle(event):
        GPIO.output(IN1, GPIO.HIGH)
        GPIO.output(IN2, GPIO.LOW)
        GPIO.output(IN3, GPIO.HIGH)
        GPIO.output(IN4, GPIO.LOW)
        pwm_ENA.ChangeDutyCycle(100)
        pwm_ENB.ChangeDutyCycle(40)

# Reverse Right
def RRToggle(event):
        GPIO.output(IN1, GPIO.LOW)
        GPIO.output(IN2, GPIO.HIGH)
        GPIO.output(IN3, GPIO.LOW)
        GPIO.output(IN4, GPIO.HIGH)
        pwm_ENA.ChangeDutyCycle(40)
        pwm_ENB.ChangeDutyCycle(100)

# Reverse Left
def RLToggle(event):
        GPIO.output(IN1, GPIO.LOW)
        GPIO.output(IN2, GPIO.HIGH)
        GPIO.output(IN3, GPIO.LOW)
```

```
        GPIO.output(IN4, GPIO.HIGH)
        pwm_ENA.ChangeDutyCycle(100)
        pwm_ENB.ChangeDutyCycle(40)

# Stop moving
def FullStop(event):
        GPIO.output(IN1, GPIO.LOW)
        GPIO.output(IN2, GPIO.LOW)
        GPIO.output(IN3, GPIO.LOW)
        GPIO.output(IN4, GPIO.LOW)
        pwm_ENA.ChangeDutyCycle(0)
        pwm_ENB.ChangeDutyCycle(0)

# Cutter Blade Control
def Cutter():
    if GPIO.input(24):
        GPIO.output(IN5, GPIO.LOW)
        GPIO.output(IN6, GPIO.LOW)
        pwm_ENA.ChangeDutyCycle(0)
        CutButton["text"] = "Turn Blade On"
        CutButton["bg"] = 'green'

    else:
        GPIO.output(IN5, GPIO.LOW)
        GPIO.output(IN6, GPIO.HIGH)
        pwm_ENA.ChangeDutyCycle(50)
        CutButton["text"] = "Turn Blade Off"
        CutButton["bg"] = 'red'

# Manual Mode
## Disables movement buttons
def Manual():
    if wButton["state"] == "normal":
        wButton["state"] = "disabled"
        aButton["state"] = "disabled"
        sButton["state"] = "disabled"
        dButton["state"] = "disabled"
        CutButton["state"] = "disabled"
        GPIO.output(IN5, GPIO.LOW)  #Kills cutter blade if on
        GPIO.output(IN6, GPIO.LOW)  #Kills cutter blade if on
        pwm_ENA.ChangeDutyCycle(0)  #Kills cutter blade if on
        ManualButton["text"] = "Manual Mode Off"
        ManualButton["bg"] = 'red'
    else:
```

```
        wButton["state"] = "normal"
        aButton["state"] = "normal"
        sButton["state"] = "normal"
        dButton["state"] = "normal"
        CutButton["state"] = "normal"
        ManualButton["text"] = "Manual Mode On"
        ManualButton["bg"] = 'green'

def close():
    GPIO.cleanup()
    from subprocess import call
    call("sudo shutdown -h now", shell=True)
#    win.destroy()

####  GPU INTERFACE  ####

## Widgets ##
# Button format (window, text displayed, font defined in GUI defs, Event Toggle,
Background, Height, Width)

# Forward button
wButton = Button(win, text = 'Forward', font = myFont, bg = 'bisque2', height = 3, width
= 6)
wButton.grid (row=0, column=1)
wButton.bind("<ButtonPress>", wToggle)
wButton.bind("<ButtonRelease>", FullStop)

# Left button
aButton = Button(win, text = 'Left', font = myFont, bg = 'bisque2', height = 3, width = 6)
aButton.grid (row=1, column=0)
aButton.bind("<ButtonPress>", aToggle)
aButton.bind("<ButtonRelease>", FullStop)

# Right button
dButton = Button(win, text = 'Right', font = myFont, bg = 'bisque2', height = 3, width = 6)
dButton.grid (row=1, column=2)
dButton.bind("<ButtonPress>", dToggle)
dButton.bind("<ButtonRelease>", FullStop)

# Back button
sButton = Button(win, text = 'Back', font = myFont, bg = 'bisque2', height = 3, width = 6)
sButton.grid (row=2, column=1)
sButton.bind("<ButtonPress>", sToggle)
sButton.bind("<ButtonRelease>", FullStop)
```

```
# Forward Right button
sButton = Button(win, text = 'FR', font = myFont, bg = 'bisque2', height = 3, width = 6)
sButton.grid (row=0, column=2)
sButton.bind("<ButtonPress>", FRToggle)
sButton.bind("<ButtonRelease>", FullStop)

# Forward Left button
sButton = Button(win, text = 'FL', font = myFont, bg = 'bisque2', height = 3, width = 6)
sButton.grid (row=0, column=0)
sButton.bind("<ButtonPress>", FLToggle)
sButton.bind("<ButtonRelease>", FullStop)

# Reverse Right button
sButton = Button(win, text = 'RR', font = myFont, bg = 'bisque2', height = 3, width = 6)
sButton.grid (row=2, column=2)
sButton.bind("<ButtonPress>", RRToggle)
sButton.bind("<ButtonRelease>", FullStop)

# Reverse Left button
sButton = Button(win, text = 'RL', font = myFont, bg = 'bisque2', height = 3, width = 6)
sButton.grid (row=2, column=0)
sButton.bind("<ButtonPress>", RLToggle)
sButton.bind("<ButtonRelease>", FullStop)

# Cutter Blade Toggle
CutButton = Button(win, text = 'Turn Blade On', font = myFont, command = Cutter, bg =
'green', height = 2, width = 10)
CutButton.grid(row=2, column = 4)

# Manual Mode Toggle
ManualButton = Button(win, text = 'Manual Mode Off', font = myFont, command =
Manual, bg = 'green', height = 2, width = 13)
ManualButton.grid(row=0, column = 4)




# Kill Switch
exitButton = Button(win, text = 'Kill Switch', font = myFont, command = close, bg =
'red', height = 1, width = 8)
exitButton.grid (row=3, column=4)

####  END GPU INTERFACE  ####
```

```
#clean exit
win.protocol("WM_DELETE_WINDOW", close)

#Loop forever, keeps GUI running
win.mainloop()
```

B3 Compass Code

```
import smbus            #import SMBus module of I2C
from time import sleep  #import sleep
import math

#some MPU6050 Registers and their Address
Register_A     = 0           #Address of Configuration register A
Register_B     = 0x01        #Address of configuration register B
Register_mode  = 0x02        #Address of mode register

X_axis_H    = 0x03           #Address of X-axis MSB data register
Z_axis_H    = 0x05           #Address of Z-axis MSB data register
Y_axis_H    = 0x07           #Address of Y-axis MSB data register
declination = -0.00669       #define declination angle of location where measurement
going to be done
pi          = 3.14159265359  #define pi value


def Magnetometer_Init():
        #write to Configuration Register A
        bus.write_byte_data(Device_Address, Register_A, 0x70)

        #Write to Configuration Register B for gain
        bus.write_byte_data(Device_Address, Register_B, 0xa0)

        #Write to mode Register for selecting mode
        bus.write_byte_data(Device_Address, Register_mode, 0)



def read_raw_data(addr):

        #Read raw 16-bit value
        high = bus.read_byte_data(Device_Address, addr)
        low = bus.read_byte_data(Device_Address, addr+1)
```

```
        #concatenate higher and lower value
        value = ((high << 8) | low)

        #to get signed value from module
        if(value > 32768):
           value = value - 65536
        return value


bus = smbus.SMBus(1)          # or bus = smbus.SMBus(0) for older version boards
Device_Address = 0x1e   # HMC5883L magnetometer device address

Magnetometer_Init()     # initialize HMC5883L magnetometer

print (" Reading Heading Angle")

c=0

while True:


        #Read Accelerometer raw value
        x = read_raw_data(X_axis_H)
        z = read_raw_data(Z_axis_H)
        y = read_raw_data(Y_axis_H)

        heading = math.atan2(y, x) + declination

        #Due to declination ch     meck for >360 degree
        if(heading > 2*pi):
              heading = heading - 2*pi

        #check for sign
        if(heading < 0):
              heading = heading + 2*pi

        #convert into angle
        heading_angle = int(heading * 180/pi)

        a2=heading_angle-180

        front= 177
        print(a2)
```

B4 GPS Code

```
import serial
import sys
import math
import smbus
from time import sleep
import RPi.GPIO as GPIO

GPS_SERIAL = "/dev/ttyACM0" # TTY for your receiver
GPS_BAUD = 38400

running = True


def truncate(number, decimals=0): # trying to remediate some fugly Python formatting
issues
    # Returns a value truncated to a specific number of decimal places.
    if not isinstance(decimals, int):
        raise TypeError("decimal places must be an integer.")
    elif decimals < 0:
        raise ValueError("decimal places has to be 0 or more.")
    elif decimals == 0:
        return math.trunc(number)

    factor = 10.0 ** decimals
    return math.trunc(number * factor) / factor


def decimalDegrees(coordinates, digits, hemi): # crack NMEA (D)DDMM.mmmmm N/S
E/W format
    parts = coordinates.split(".")

    if (len(parts) != 2):
        return coordinates

    if (digits > 3 or digits < 2):
        return coordinates

    left = parts[0]
    right = parts[1]
```

```
    degrees = float(left[:digits]) + ((float(left[digits:digits+2])) / 60.0) + (float("." +
str(right)) / 60.0)

    if (hemi == 'S' or hemi == 'W'):
        degrees = degrees * -1.0

    return truncate(degrees, 8) # max 8 dp, if a minute is 1852m, a degree is 111120m,
roughly, this is about a cm


def getPositionData(gps):

    data = gps.readline().decode('ascii', errors='replace')

    try:
        message = data[0:6]

        if (message == '$GNGGA'):
            #0 $GNGGA
            #1 gpstime
            #2,3 lat
            #4,5 lon
            #6 fixqual
            #7 satcount
            #...
            parts = data.split(',') # separate NMEA on commas
            # should check len(parts)
            latitude = decimalDegrees(parts[2], 2, parts[3]) # DDMM.mmm
            longitude = decimalDegrees(parts[4], 3, parts[5]) # DDDMM.mmm, decimal
places in degree field, hemisphere
            fixtype = parts[6] # 4=RTK Fixed, 5=RTK Float



            y=float(latitude)
            x=float(longitude)
            z=int(fixtype)


            x0=-81.27842166
            y0= 28.50044166
```

```
            print(x,y,z)

        except Exception:
            print ("GPS:JUNK")
            stop()


def closeout():
    gps.close()

    print ("Application closed!")
    stop()
    exit()

print ("Application started!")

# Open GPS COM Port / TTY
gps = serial.Serial(GPS_SERIAL, baudrate = GPS_BAUD, timeout = 0.5)



if __name__ == "__main__":

    while running:
        try:
            getPositionData(gps)

        except KeyboardInterrupt:
            # Do a Ctrl-C in Shell Window, not stop signal
            print ("KeyboardInterrupt");
            running = False

        except:
            # You should do some error handling here...
            print ("Application error!")
            closeout()

closeout()
```

B5 Save Variables Code

```
f=open('save.txt','w')


xf=input('Enter starting longitude: ')
yf=input('Enter starting latitude')
fangle=input('Align the mower to the desired starting forward angle:')
langle=input('Align the mower to the desired starting left angle:')
bangle=input('Align the mower to the desired starting reverse angle:')
rangle=input('Align the mower to the desired starting right angle:')

H=input('Enter height of lawn in feet: ')
W=input('Enter width of lawn in feet: ')

f.write(xf+'\n')
f.write(yf+'\n')
f.write(fangle+'\n')
f.write(langle+'\n')
f.write(bangle+'\n')
f.write(rangle+'\n')
f.write(H+'\n')
f.write(W+'\n')

f.close()
```

Biography

Brian Darling

- Planning to graduate from Valencia College July 2021 for B.S. in Electrical and Computer Engineering Tech, Electronics Systems Concentration
- A.S. in Electrical Engineering Tech, Electronics Concentration
- A.S. in Electrical Engineering Tech, Lasers and Photonics Concentration
- Experienced Manufacturing Engineer for PCBAs and electronic assemblies
- Experienced Test and Repair Technician for PCBAs and electronic assemblies
- Experienced Process Description Writer for PCBAs and electronic assemblies

Gabriel De La Torre

- Planning to graduate from Valencia College July 2021 for B.S. in Electrical and Computer Engineering Tech, Computer Systems Concentration
- A.S. in Electrical Engineering Tech, Electronics Concentration