200 University Ave W.
Waterloo, Ontario, N2L 3G1
April 1, 2019

Prof Bill Owen
Associate Director of First-Year Engineering
Department of Mechanical and Mechatronics Engineering
University of Waterloo
200 University Ave W.
Waterloo, Ontario, N2L 3G1

Dear Professor Owen,

This report, entitled “California Wildfire Search and Rescue Device Final Report”, was prepared by a team of engineers at Lera Inc. as an overview of the final design and results of the search and rescue device prototype, built for the Federal Emergency Management Agency (FEMA).

The report begins with a brief review of the needs of the device and the problem being solved. It continues with modifications made to the defined problem as the project neared completion. The report then describes deviations made from the proposed schedule and budget. Next, the report highlights key features, technical capabilities, construction, and testing of the device’s mechanical, electrical, and software systems. Modifications made to the original design, especially why they were made, is a further key point of discussion in the report. Lastly, the device’s performance and demo day results are highlighted, followed by the team’s learnings and future recommendations.
We are the sole authors of this report and, unless otherwise stated and properly referenced in the report, the entire content of this report is original work done by us. We have all read the report and are aware of the content. The content of this report has not received credit in this or any other course that we have taken in the past or are currently taking at this time.

Best Regards,

Celene (Hsuan Ling) Chen
Hsuan Ling Chen

Ella Rasmussen
Ella Rasmussen

Alexander Rathke
Alexander Rathke

Briar Smith
Briar Smith

Richard (Yangtian) Yan

Yangtian Yan

California Wildfire Search and Rescue Device for FEMA
Final Report

Prepared By:

Celene (Hsuan Ling) Chen
Ella Rasmussen
Alexander Rathke
Briar Smith
Richard (Yangtian) Yan

April 1, 2019

Executive Summary

This report, entitled “California Wildfire Search and Rescue Device for FEMA Final Report” details the results of the project contract between Lera Inc. and FEMA. Specifically, this report talks about deviations from the initial proposal, areas of improvement for future work on this project, should it continue, and the successes and failures of the project.

The goal of this project was to design a prototype that should be an autonomous, scaled-down proof of concept that is capable of extinguishing a fire and navigating terrain such as sand, gravel, and emptied pits, in order to save the lives of at-risk civilians and deliver food. The device must be land-based and be transportable by a human, limiting the size and mass of the prototype.

The project was able to complete 100% of the mechanical and electrical construction, and 96% of software. The remaining 4% of software that was required were the final pieces to tie together the challenges the robot faced, such as localization when objects were present on the field. Thus, the robot was able to complete the objectives individually, but not together, for the March 22 demo deadline.

The project exceeded the $265.00 budget by $572.40, bringing the total spending to $837.49. Areas that substantially exceeded budget include sensors, electrical circuit, MCU, and consumables. The budget was exceeded in these areas due to incompatibility issues of the proposed MCU and sensors, and having to trade cost for time.

It was discovered when trying to tie together all of the challenges into a fully autonomous robot that the mechanical design, while constructed, needed to undergo substantial changes. The wheels had too much static friction and the motors did not provided more speed than torque, which produced unreliable turns and driving. Sensor mounting issues included the need to mount external encoders, which were too heavy at the back of the robot thus shifting the robot’s center of gravity. The chassis design out of hardwood was sufficient for housing all of the electrical equipment.

On the other hand, the electrical system was reliable and robust. The power supply was sufficient for the selected suite of sensors and actuators. The main problems with the electrical system were lack of processing power with the Arduino Uno, incompatibility of the Pi and the Teensy, and the IMU having too much drift and electrical noise, requiring multiple iterations of IMU’s.

The ability to complete the challenges ultimately fell on the software team. The PID controllers were able to perform 90^{o} turns and drive to specified distances with some error due to the static friction in the mechanical system. A grid map gave the robot the ability to localize itself and objects. Path planning was then able to generate optimal paths, with the appropriate priorities for each challenge.

The resulting device met all criteria and constraints, aside from exceeding the budget and limiting reliability due to issues with the drivetrain. Throughout the project, the team at Lera Inc. learned the importance of time management, priority shifting, problem isolation, and frequent communication. In the future, the team will work to appropriately update project scope in a timely fashion and increase the frequency of team meetings with stakeholders.

Table of Contents

List of Figures

List of Tables

1.0 Introduction and Background

1.1 Introduction

Citizens of California are often subjected to hazardous environments and in need of rescue if caught in a wildfire. As such, the Federal Emergency Management Agency (FEMA), which was at a lapse of funding due to government shutdown [1], requires a cost-efficient prototype for a wildfire search and rescue device. A team of engineers at Lera Inc. were tasked with designing, constructing, and testing this prototype. This report is a follow-up to the initial project proposal [2], with the final design and results of this prototype being the main point of discussion.

1.2 Needs Assessment Review

The environments in which California wildfires act contain uneven terrain, bodies of water, gravel, and sand, in addition to flat terrain. For a search and rescue device to be successful it needs to be able to move through such an environment. Thus, it is required to either identify and avoid hazardous terrain or be able to traverse it. Note, bodies of water were removed from the test environment part way through the device's construction and testing phase, and replaced with empty pits.

Procedurally, it is safer to first put out any fires in the vicinity of the survivors before attempting to find them. Furthermore, the successful search and rescue devices needs to be capable of finding any lost survivors, as well as food in the vicinity of the survivors to prevent them from starving.

California wildfires are increasingly putting lives at risk, creating a need for a method of conducting search and rescue operations with the ability to navigate rough terrain, detect and put out a fire, find and deliver food, and locate survivors.

1.3 Problem Formulation Review

1.3.1 Problem Definition

The initial problem definition was to design a device to conduct search and rescue operations with the ability to navigate through and around terrain including flat wood, sand pits, gravel pits, and water pits, detect and put out a fire, find and deliver food, locate survivors, and return to its initial starting position. However, the water pits were deemed too hazardous during device construction and testing, and were replaced with empty pits.

1.3.2 Desired Functions and Goals

The device's area of operation is within a 6x6 tile grid enclosed by cardboard walls. It must be capable of navigating terrain tiles consisting of flat wood, empty pits with 5 cm steps, gravel, and sand.

In addition, the device is to be able to locate and put out a fire, specifically, that emanating from a candle, before carrying out other search and rescue operations. Then, the device should be capable of locating a group of survivors and a lone survivor. Survivors are represented using Lego parts and structures. Food, in the form of a magnet (initially expected to be a metal ball bearing instead) buried in a sand tile, is also to be detected by the device and delivered to the group of survivors.

1.3.3 Objectives

The objectives of the device can be split into physical and non-physical.

Physical:

- Dimensions should be within 0.2 x 0.2 x 0.2 m
- Weigh under 5 kg, such that one person can carry the device
- Able to detect and identify objects such as cardboard walls, Lego structures, and a 12 cm candle within a 2.5 m radius
- Able to identify a flame emitted from a 12 cm tall candle within a 1.8 m radius
- Able to extinguish a flame emitted from a 12 cm tall candle within 15 cm of the device
- Able to navigate in an environment with flat wood terrain, sand mounds, gravel mounds, and 5 cm steps down into empty pits, with an average speed above 0.3 m/s
- Allows for field swapping of major components in under 5 hours
- Capable of detecting a magnet buried underneath 5 cm of sand underneath device (note: it was initially expected that the magnet would be a metal ball bearing)
- Contain an easy to replace power source (i.e. should be field-swappable)
- Track location within search and rescue site to 5 cm radius of accuracy, such that the device is able to localize itself within the area of operation
- Reliably carry out challenges for a minimum of 10 consecutive missions without intervention

Non-physical:

- Cost under $265 CAD

1.3.4 Constraints

The constraints on the final design of the device include:

- Must complete objectives autonomously
- Does not use camera and vision systems
- Does not have flying capabilities
- Stays within boundaries of search and rescue site, marked by cardboard walls

1.3.5 Design Selection Criteria

The proposed design was selected based on the following criteria:

- Cost [$]
- Speed [minutes]
- Size [m x m x m]
- Mass [kg]
- Object detection range and accuracy
- Robustness

- Construction complexity
- Reliability

The final design was selected from alternatives in the initial proposal report [2].

1.4 Key Design Problems That Were Considered

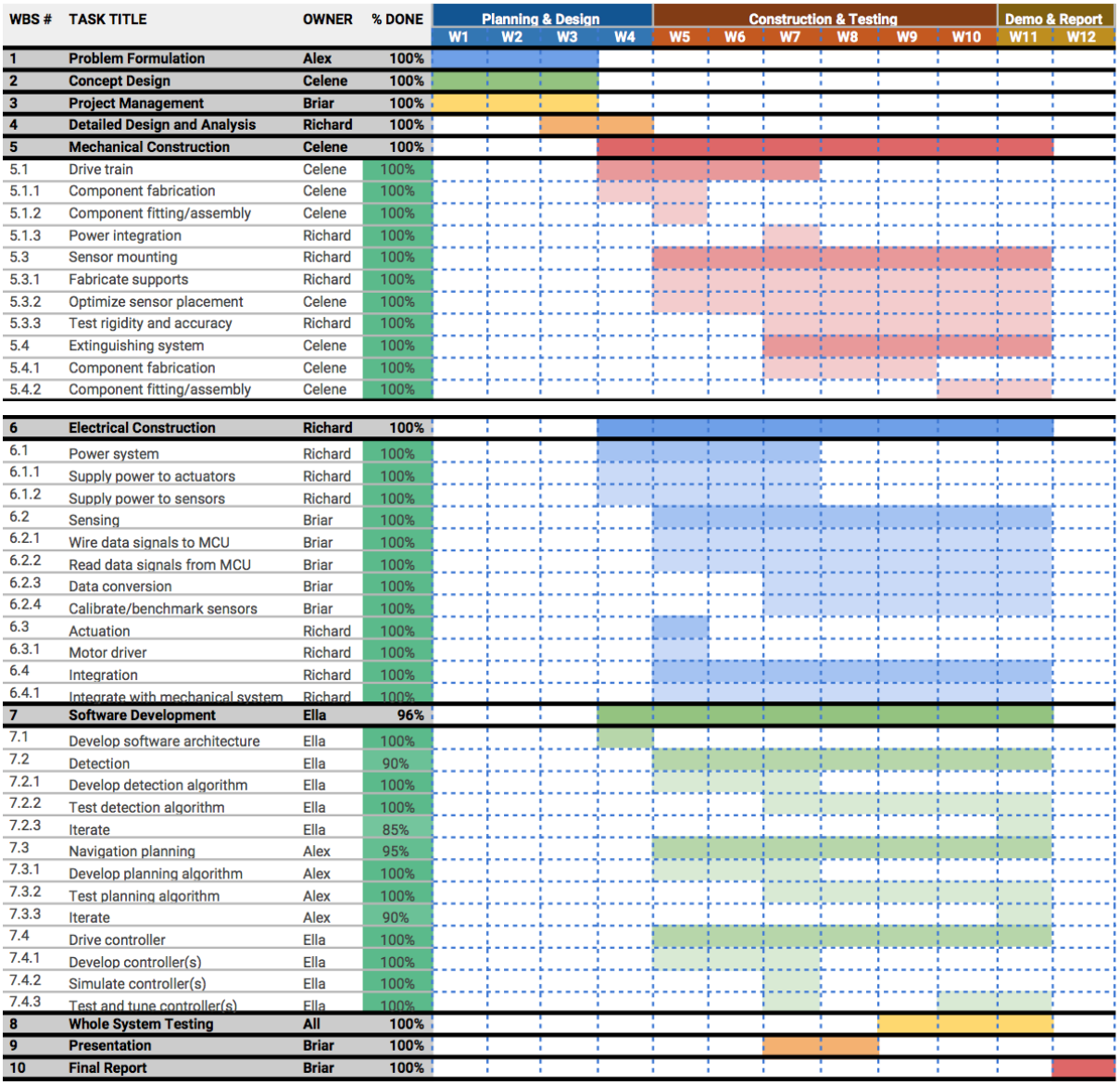
There are three main design problems that needed to be considered when designing and constructing the search and rescue device. First, the device must be able to traverse through gravel, sand, and empty pits, which could cause the device to get stuck, jammed with dirt, or malfunction. Second, the device must be capable of detecting and identifying several different objects, including Lego, a 12 cm lit candle, and a magnet buried in sand tile. Lastly, the device must be able to navigate across the field to complete the missions. To do so, it must localize itself within the field, maintain an internal representation of the field and its objects, and reliably plan and follow a path to its target destination.

2.0 Scheduling Update

In comparison to the schedule proposed in the initial project proposal [2], the project experienced major delays during the construction and testing phase. Table 1 shows an updated Gantt chart illustrating the length of time each subtask took in actuality during the construction process. Note that some subtasks are removed when comparing Table 1 to the original Gantt chart, such as suspension design, since they were deemed to be unnecessary components during the detailed design phase and were not included in the final design.

Key areas that caused delays in scheduling include sensor mounting and sensor bring up, which are reflected in the updated Gantt chart, due to unexpected issues and areas of difficulty that are discussed in more detail in sections 5.0 and 7.0 of the report. These delays in electrical and mechanical construction caused further delays in software development and testing, resulting in some components of the detection and navigation software to not be fully completed in time for demo day. Incomplete components of the software include lack of ability to localize when objects are present on the field and an incomplete field scanning routine (further discussed in section 7.2), hence the software is marked as 96% complete in Table 1. Electrical and mechanical components were completed entirely by demo day, and thus are marked as 100% complete in Table 1.

Table 1: Final Gantt Chart



3.0 Final Budget

The purpose of this section is to present the final cost of the project, and discuss the reasons the project did or did not exceed the budget for each category. Table 2 shows a summarized version of the cost for each category, including all funds used to experiment and iterate on the design and the cost of the final design itself. A full bill of materials can be found in Appendix A, detailing the cost of all items in each category.

Table 2: Search and Rescue Device Final Budget Per Category

Expense	Originally Estimated Cost	Actual Cost
Mechanical System and Machining	$70.00	$55.51
Power System	$30.00	$157.04
Electrical Circuit	N/A	$146.00

Sensors	$100.00	$268.16
MCU	$20.00	$111.23
Consumables	$25.00	$71.99
Miscellaneous	$20.00	$26.63
Total	**$265.00**	**$837.49**

In the project proposal, the expense column did not have the Electrical Circuit category since the main electrical expenses were intended to be under the Power System category. These categories have been separated for clarity.

The Mechanical System and Machining category was under budget because of no unanticipated failures. The power system category was over budget since the team was not provided with a power system, which was originally anticipated. The electrical circuit category, which was originally included in the power system category, as well as the sensors category and the consumables category, were extremely over budget due to more than anticipated failures with the IMU, and the encoders being more expensive since there was not enough time to make our own. The MCU category was over budget because of difficulties involved in using the Raspberry Pi, eventually causing the switch to the Arduino Teensy. The miscellaneous category was only slightly over budget. Overall, the categories that were over budget were mostly due to inexperience using the items, and often needing to trade cost for convenience when prototyping. In conclusion, the cost of building this project was $572.40 more than the originally proposed budget, bringing the total cost of the project to $837.49.

4.0 Mechanical System

4.1 System Overview

The mechanical design of the robot involves a structure that facilitates the transmission of force from the motors to the wheel while providing mounting points for all electronics and sensors. Certain aspects of the mechanical design are defined by objectives and constraints, including the device’s size, mass, and speed, as well as ensuring the design of sensor placement and mounting that maximizes accuracy and robustness of detection and control features. The objective of allowing for field swappable components was not a core focus of the mechanical design due to time limitations. The final mechanical construction prototype is shown in Figure 1.

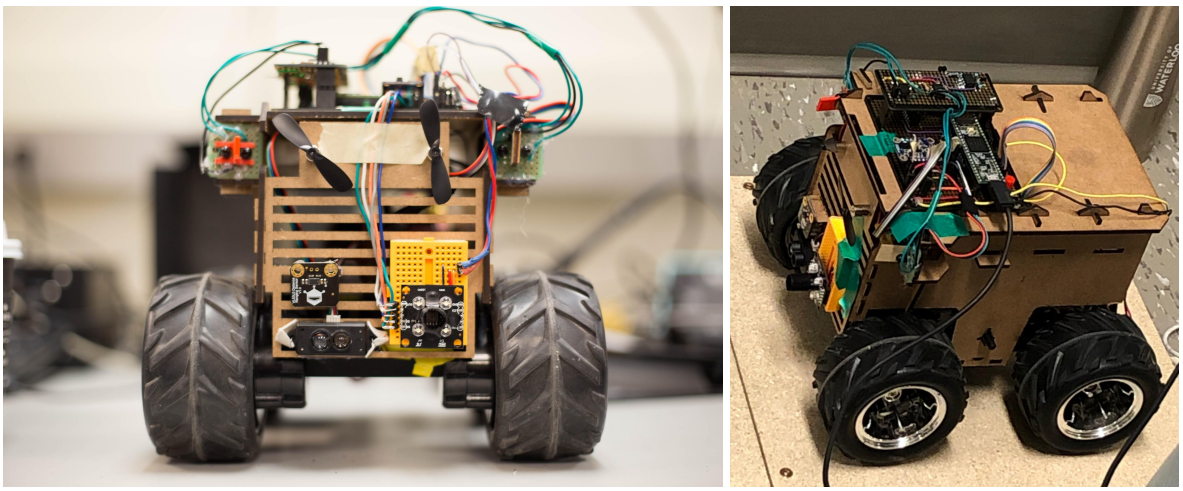


Figure 1: Final Mechanical Prototype Front View (Left) and Side View (Right)

A pre-built drivetrain from an RC car is used with built-in motors and wheels, as well as a fixed gear ratio. This implementation provides a sturdy manufactured drivetrain and allowed for large time savings in the construction and testing phase of the project. The pre-designed chassis has its drawbacks, it limits the amount of modification that can be made to the robot, such as mounting space and modifications to the gear ratio.

4.2 Manufacturability

The initial design proposal of the robot included a protection casing that protects the robot from splash and debris. However, the project definition was adjusted to not include water, and thus a protective seal is no longer required. After the consideration of several factors such as the manufacturability, ease of construction, time, cost and material access, laser cutting hardboard was the selected method of choice for manufacturing. Laser cutting helps to reduce the amount of manual work needed such as hand cutting and drilling to produce the chassis, and it provides more accurate cutting.

3D printing is another method that was considered, specifically for some sensor mounts such as the fire detection sensors. However, due to limited access to the 3D printer and the excessive time required to print the parts some parts are not suitable to be 3D printed, and were handcrafted instead with wood, glue, and wire.

4.3 Chassis Design

The chassis of the robot is designed to rigidly hold all the batteries, sensors, motors and the microcontroller while remaining lightweight. Design and analysis was completed in AutoCad and reviewed by the team prior to construction to verify placement of components and dimensioning.

The final robot chassis was designed symmetrically to ensure even force distribution and stress on both halves of the robot. The base and elevated platforms of the chassis form a box to increase the moment of inertia in all directions and minimize deflection and stress. The chassis is also designed to be lightweight and rigid with the use of the 3 mm hardboard. The mortise and tenon joint technique is a simple and strong technique when adjoining two wooden pieces connected at right angles, and it is

used in the robot to join the top platforms and sides. All platforms were chosen to be through types to keep the design simple [3]. A total of three versions of the chassis were developed and constructed.

The first version of the chassis was printed with lightweight plywood material. The locking of the wood together was not strong enough or rigid, and the parts required glue to be held in place. Hence this design was not able to be taken apart for reassembly when needed.

The second version aimed to solve the issues encountered with the first version. During construction of this version plywood ran out of stock, so hardboard was selected as the next option for the laser cutting material instead. The hardboard option was 1 mm thinner than the original plywood supply which did not add on to a lot of mass on to the robot. Hardwood is also denser and therefore much stronger than plywood, hence a better material choice for the robot's chassis [4]. A majority of the sensors were able to be easily mounted at the frontal area of the robot due to the abundance of mounting space, as seen in Figure 2 below. Holes were also cut in the front mounting plate for cable arrangement and control. Wedges were added to the mortise and tenons to lock the joints in place, fixing the rigidity issues encountered with the first version. With the addition of the wedges, the second version of the chassis design was able to secure the chassis assembly without the use of glue, allowing for disassembly when required. However, a weak point was noticed in the frontal sensor mount area in which was prone to breakage if impacted from the front, such as from bumping into the field boundaries.

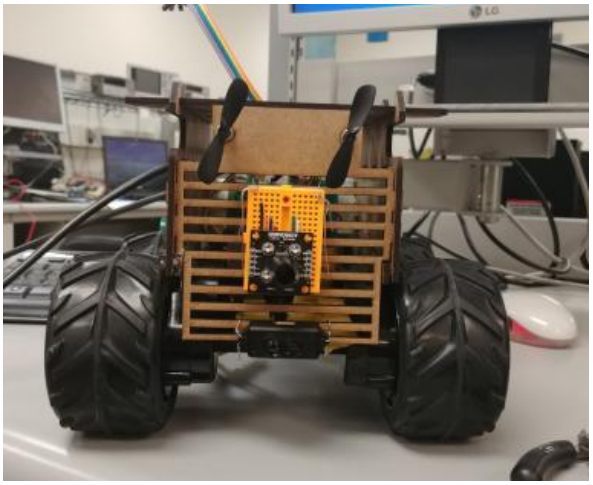


Figure 2: Second Version of the Chassis Construction

The final version of the chassis used on demo day, shown in Figure 1, included a second mounting deck to increase the electronics mounting area and stabilize the angled IR sensor mounts. An additional mortise and tenon joint is added to the front of the chassis for stronger support to fix the weak point from the second version. The CAD drawings for the final design that outline where holes and slots are made for mounting the sensors, fans and circuitry, and can be found in Appendix B along with their dimensions.

4.4 Sensors Placement and Mounting

Some of the smaller sensors, such as the IR sensors for flame detection and the colour sensor, required separate components either for mounting or aiding the functionality of the sensors. For 3D printing,

there is more geometry control and freedom in design than laser cutting. However, only small parts were 3D printed due to long printing times and limited access to the printer. Hence laser cutting was used to manufacture larger sensor mounts. The 3D printed parts are designed in Solidworks. Figures 3 and 4 show the implementation of the divider for the IR sensors. The IR dividers allow for pin-pointing the fire's direction as well as distance from the fan to enable most effective extinguishing. The black film tube to isolate ambient light for the colour sensor.

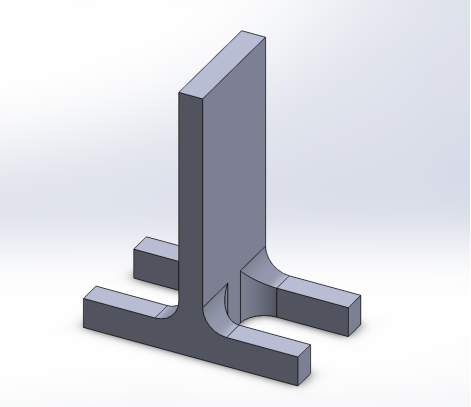


Figure 3: IR Sensor Divider

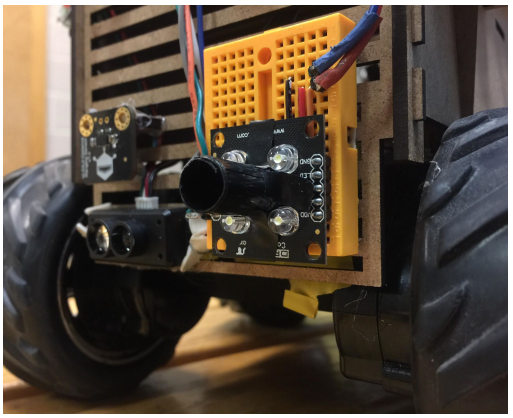


Figure 4: Black film tube on colour sensor

4.5 Encoders

The mounting of the encoders was one of the biggest challenges of the mechanical design. The drivetrain of the robot came from a pre-built RC car which did not have encoders attached internally to the motors. Several ideas were proposed for this problem, such as interfacing the sensor with the chassis's gearbox or extending the shaft on the outside of the wheel in order to connect the encoders. With limited mounting space available on the chassis after all the electronics and power systems were installed, the external encoders were placed at the back of the robot in order to be accessible to the wheels for reading, shown in Figure 5 below. The final decision was to make the center rim of the robot's wheel and the encoder's pinion to be in contact with each other in order to receive the readings from the wheel to the encoder. Electrical tape with rubbery surface was used to wrap around the encoders' pinions to increase friction and constant contact with the wheels. This method was shown to be functionable and reliable when traveling in straight directions after several runs and testings on the robot. However, the performance of the robot with respect to turning and control was greatly affected by the mass of the encoders (0.468kg total) at the rear of the robot, as discussed in section 7.2, thus the encoders were removed prior to demo day.

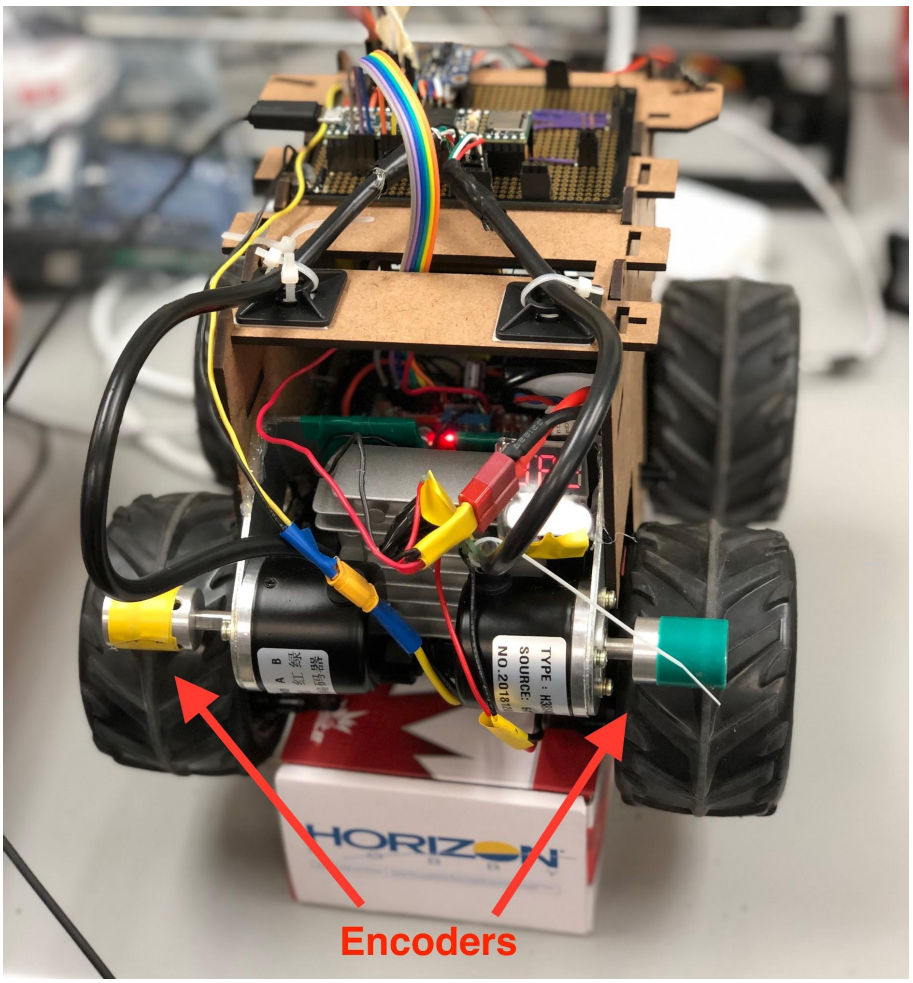


Figure 5: Encoder Mounting Solution at Robot Rear

5.0 Electrical System

5.1 System Overview

The electrical system of the search and rescue device can be divided into the battery supply, as well as three main domains: drivetrain system, fire extinguishing system, and electronics system. The drivetrain system sources power directly from the battery. A H-bridge motor driver IC and two DC brushed motors are its main components. The fire extinguishing system operates using an identical principle, and only uses the motors to spin fans. The electronics system consists of a voltage regulator, a microcontroller unit (MCU), an array of sensors, and supporting circuitry to power and communicate with each other.

The electrical systems are mounted throughout the vehicle. The battery and the motor driver are enclosed between the plastic and laser-cutted chassis, protected from the environment. The voltage regulator is mounted at the rear of the vehicle. The fire extinguishing circuit is mounted in the second deck. For ease of access, all other electronics are mounted on the top of the chassis. For digital components, headers and connection wires are soldered onto the prototype board. This provides the ability to easily remove or replace components using a simple, yet reliable, electrical connection.

Most of the analog electronics are soldered onto a second layer prototype board, which is stacked on the main board. The demo day electronics setup can be seen in Figure 6 below. Note, the analog electronics deck is on the left side of the picture, and the removed encoder connections near the top.

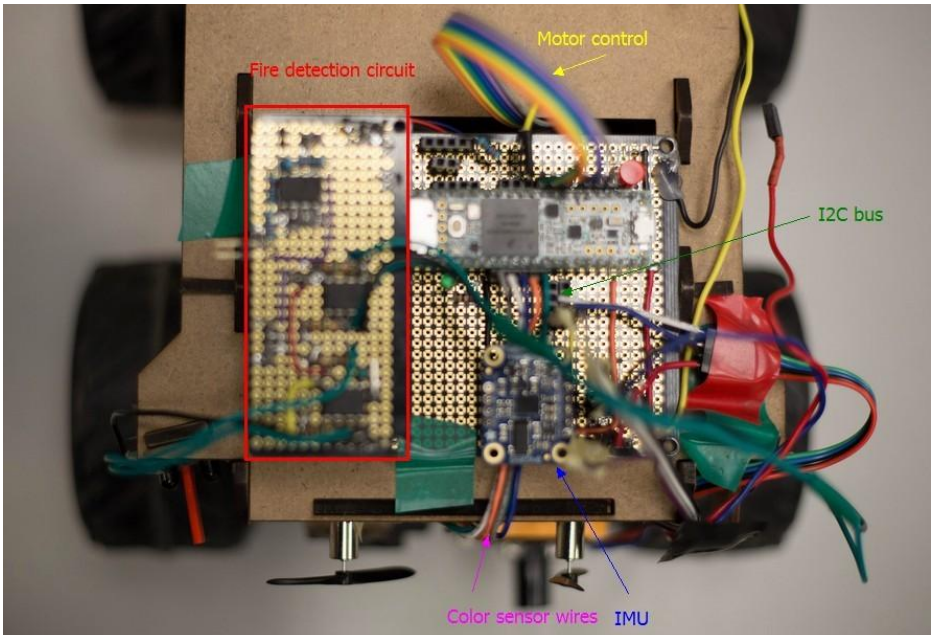


Figure 6: Electronics connections as used on demo day

Figure 7 shows a high level block diagram of the vehicle's electrical systems. Each block is explained in further detail in the following sections. Note the electronics system operates largely independently from the drivetrain and fire extinguishing systems, this is due to the different operating voltages of the components and a need to mitigate electrical and EM interference. Moreover, The sensors with high data flow such as the IMU and the Lidars are connected via an I2C bus.

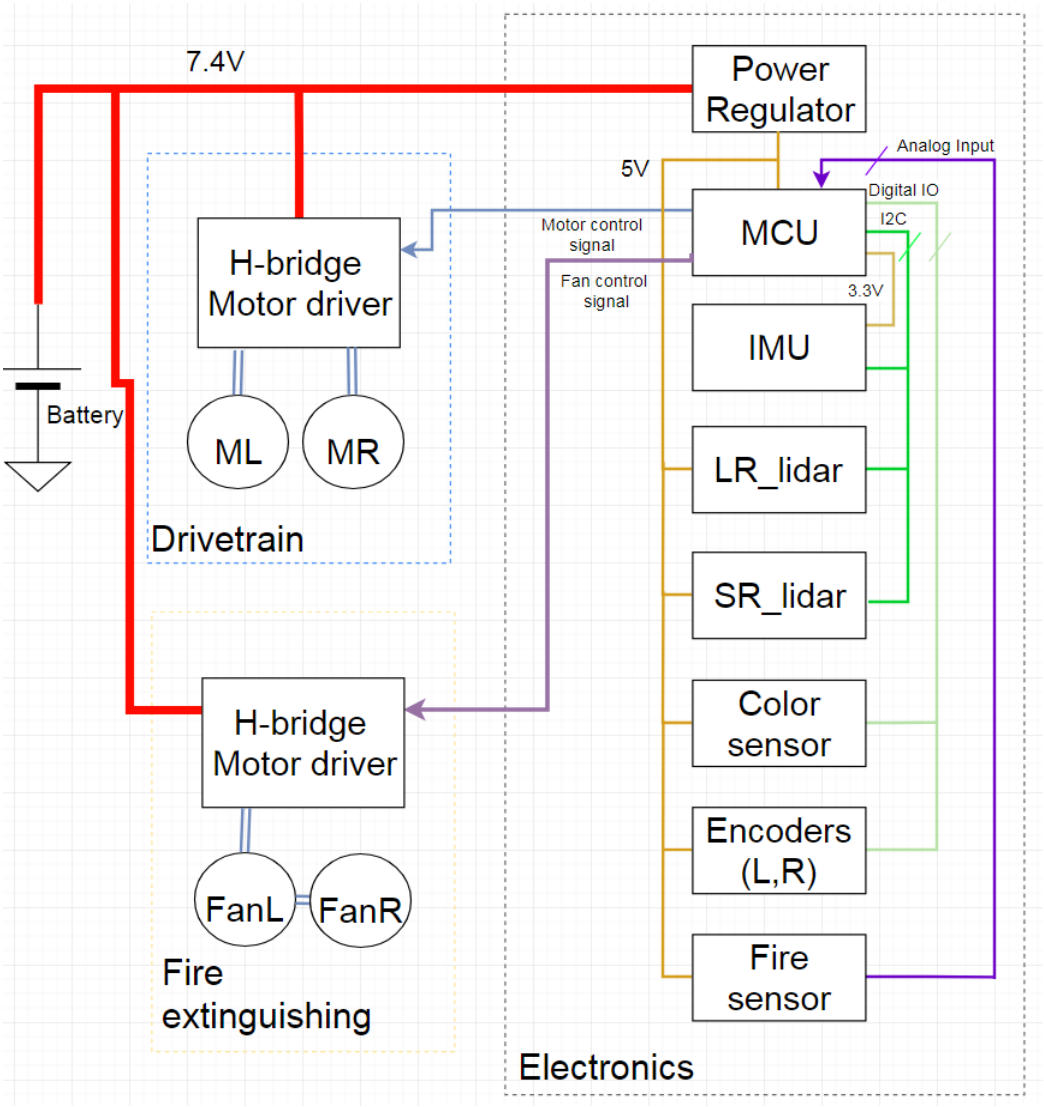


Figure 7: Electrical systems block diagram representation

5.2 Battery

A 7.4V 2000mAh Lithium Polymer (LIPO) battery is used to power the vehicle. This battery configuration was selected due to its appropriate physical dimensions and ability to provide approximately 15 minutes of service time. The battery is connected using Dean connectors which are easily removable. There is extra room in the chassis to carry a second battery in parallel with the first to double the power capacity, but mission testing determined a single battery provides more than 15 minutes of vehicle operation.

5.3 H-bridge Motor Driver and Motors

The prebuilt vehicle drivetrain is equipped with two brushed DC motors, originally operated by a low capacity 9.2V Nickel Cadmium battery. The new motors sourced from a 7.4V LiPo battery with reduced power consumption and better controllability. During early stages of chassis testing, a stalling current of approximately 1.4A per motor was observed. Using this information, a commonly used H-bridge driver, L298N, was selected for the vehicle that is capable of supplying up to 2A of

continuous current to each motor. Figure 8 shows a common application of driving two motors with the L298N, as well as its internal schematics.

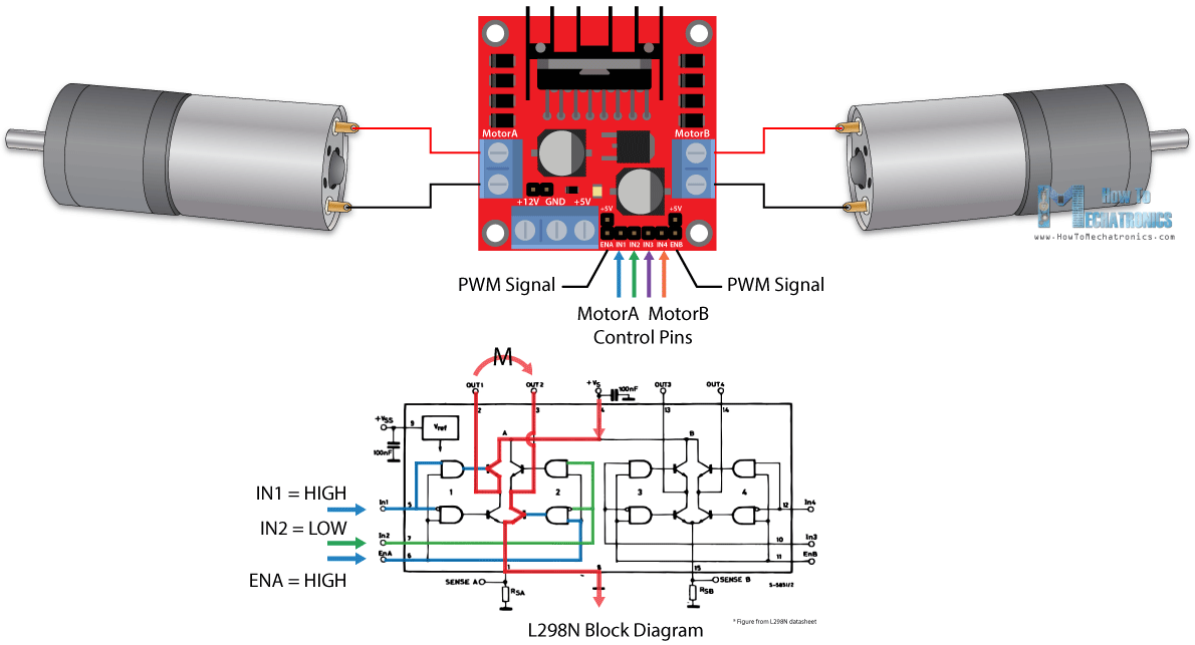


Figure 8: L298N board connections and schematics [5]

The L298N integrated many useful features in a small package, namely:

- Protection diodes to prevent motor inductive kickback damaging the IC
- Onboard 7805 voltage regulator to supply its logic control signals
- User-friendly screw fastened power connectors
- Heatsink for the IC
- Small overall physical footprint

5.4.0 Electronics

5.4.1 Voltage Regulator

A 5V 5A voltage regulator was selected, providing sufficient continuous current to all analog and digital electronics. To ensure a stable supply to the voltage regulator, a low-pass LC filter was implemented near its battery input terminals. To minimize output ripple and noise, capacitors of 470, 47, and 0.1uF were placed between the 5V power rail and ground to stabilize the voltage output.

5.4.2 MCU

The initially proposed MCU, a Raspberry Pi 3 Model B+, was selected as the main processor. The Pi was paired with a USB (UART) connected to an Arduino Uno. The Uno acted as an ADC and a midpoint processor for reading sensor data. However, as the project progressed it became apparent the Arduino Uno's limited processing power and UART bandwidth were becoming limiting factors. Additionally, a lack of off-the-shelf driver libraries for the Raspberry Pi made it difficult to shift processing load from the Arduino Uno to the Pi. The complexity of the hardware and software configuration would have made it difficult to fully utilize the Pi's processing power before the project

deadline, as it required writing many software libraries when compared to Arduino's open source simple interfaces.

A Teensy 3.6 MCU was used as a solution for this issue, as it possesses high processing power, arduino IDE compatibility, and enough physical IO pins to accommodate our project needs, as well as removing the need for UART communication between two MCUs. The Teensey replaced both the Pi and the Uno to become the vehicle's only MCU.

The compact size of the MCU allowed it to be mounted in the center of the electronics prototype board, and permitted easy cable management. The digital connection of the MCU is shown in Figure 9. The implementation of header connectors on the prototype board ensured a reliable electrical connection, modularity for introducing new hardware, as well as ease of troubleshooting.

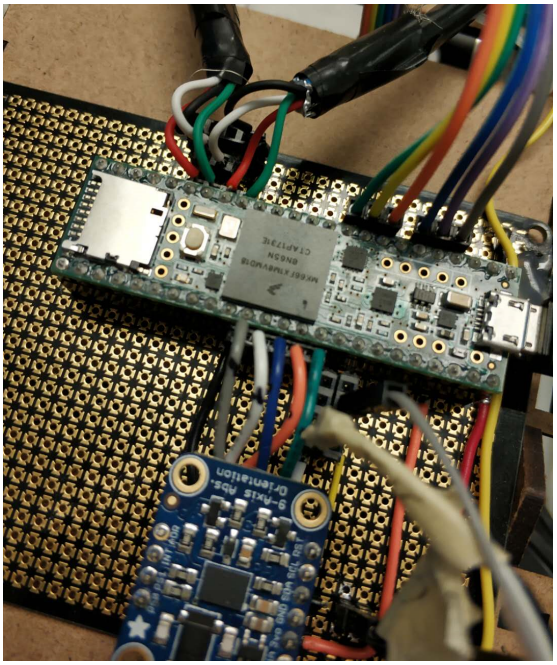


Figure 9: MCU digital connection

5.4.3 IMU

Several different inertial measurement units (IMUs) were tested before a suitable one was finally found. The MPU6050 and its successor, MPU9250, posed difficulties due to sensor drift in the yaw axis and high requirements for processing power. In the end, the BNO055 IMU was chosen as it provided low drift and the Arduino library was easier to interface with. Testing indicated that the BNO055's magnetometer is sensitive enough to detect the food magnets. The magnetometer even outperformed the hall effect sensors designed specifically for the task of detecting magnets. All three tested IMUs were connected to the MCU via I2C bus. However, only the MPU6050 was powered by the 5V power rail. The rest were powered by the Arduino's onboard 3.3V supply. Therefore minimal effort was made to electrically connect each iteration of the IMU.

5.4.4 Long Range Lidar (TFmini)

The TFmini LiDAR is an infrared laser time-of-flight (TOF) distance sensor. The TFmini is able to detect objects up to and beyond the course boundaries. However, it has a lower bound on its detection range of 30cm. Furthermore, it has a narrow 2.3^{o} field of view, allowing for the mapping of objects with a high level of precision. The TFmini is powered by the 5V rail. Its powerful 100Hz detection comes at a cost of a brief 100Hz current draw at 800mA, which required several magnitudes of capacitors ranging from 470uF to 0.1uF to be placed on the power rail and close to the TFmini connection to mitigate power rail ripples. The TFmini also shares the same I2C bus as the IMU.

The TFmini Arduino library made interfacing with this sensor straightforward. However, by default the sensor switches between range modes for different range groups. During range mode switching, approximately 6 cm of error is introduced briefly into the readings. This was avoided by manually switching the sensor to a single range mode ideal for the device's application (detection under 2 m). The TFmini is configured to use mm level precision, which improves performance for the drive PID controller discussed in section 6.4.

5.4.5 Short Range Lidar (Gravity)

The gravity sensor is an evaluation board package of a sharp infrared distance sensor chip. With a 23^{o} wide field of view and accurate distance measurements of up to 80 cm, it is a good compliment to the TFmini LiDAR, as it is able to detect close-up objects, less than than 30 cm, with high accuracy. Testing showed it was able to achieve mm level resolution. The gravity sensor is powered from the 5V rail, and shares the same I2C bus with the IMU and the TFmini lidar, therefore the connection was straightforward.

5.4.6 Color Sensor

The color sensor is used to distinguish between the group and the lone survivors by differentiating red and yellow houses in the course. The sensor produces reliable results under very short range. Therefore it requires the combined effort of the short range gravity sensor and the drivetrain to position the vehicle correctly prior to identification. The color sensor is powered by the 5V power rail. The sensor uses 4 digital input pins to configure its color filters and light sensitivity, and one digital output to deliver the results to the MCU.

5.4.7 Encoders

Two encoders (SEN0230 produced by dfrobot) [6] are mechanically connected to the rear wheels to provide positional feedback. The four input pins were configured as interrupt inputs on the Teensy MCU.

5.4.8 Fire Sensor

The fire sensor system is the only system designed in-house, as a non-integrated analog circuit on the vehicle. The system can be divided into two stages of amplifier circuits, as shown below in Figure 10.

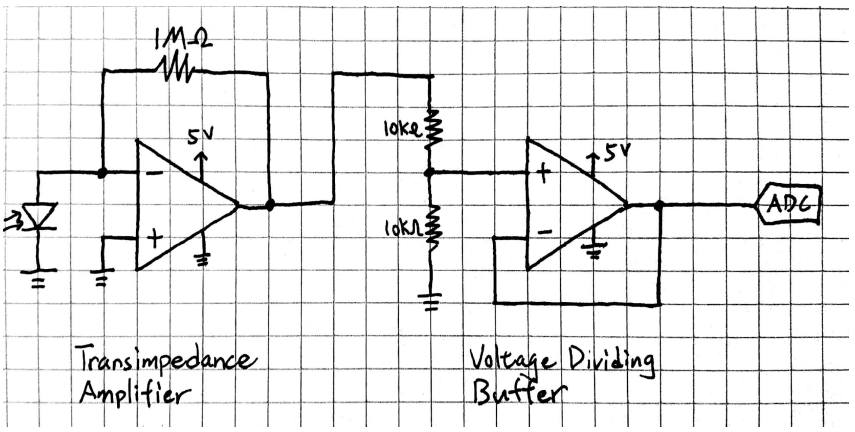


Figure 10: Fire sensor schematics

The fire sensor sensor relies on an IR photodiode to generate current when exposed to direct IR light emitted by the candle's flame. Its sensitivity to light in the 950 nm wavelength range ensures a reliable detection. The current produced from the photodiode is then fed into a transimpedance amplifier to be amplified into an analog voltage.

Since all op amps selected for the vehicle construction are powered and output up to 5V, there is a need to step down the voltage before feeding to the Teensy MCU's 3.6V tolerant ADC ports. A voltage divider is placed onto the output voltage from the first amplifier stage, and two equal 10KΩ resistors ensured a step down to 2.5V maximum input to the ADC. The voltage dividing buffer stage was not required nor planned in the initial design, due to the Arduino Uno's ADC being 5V compatible, but was required after the switch to the 3.3V operating Teensy 3.6 MCU, providing a simple electrical connection.

5.5 Fire Extinguisher

The fire extinguisher circuit operates very similarly to the vehicle motor driver. Two 3.7V drone propeller motors are connected in series to consume the 7.4V full voltage from the 2s-LIPO battery. Although the fans are only required to turn one way, on principle only a single relay or power transistor is required for the motor driver application. The available power transistors require 5V input, thus driving them with a 3.3V digital output from the MCU is sufficiently difficult. As an alternative, a small footprint 3.3V logic H-bridge is used to power the motors. Its package can be seen below in Figure 11. With the directional pins hardwired to 5V and ground, fixing the motor direction, only one digital output pin is required from the Teensy MCU.

Figure 11: Small package H-bridge IC [7]

Note the TO-220-7 footprint of the IC. The fans' current draw are below 1A, and significantly below the 5A rated current supply of the IC, also eliminating the requirement of a heatsink.

6.0 Software System

6.1 System Overview

The final design of the software system involves the core components of detection, planning, and control organized into a state machine. Figure 12 below outlines the software architecture, where some functions are run on every loop cycle (every 50 ms), and some functions are run only when the robot is in a particular state.

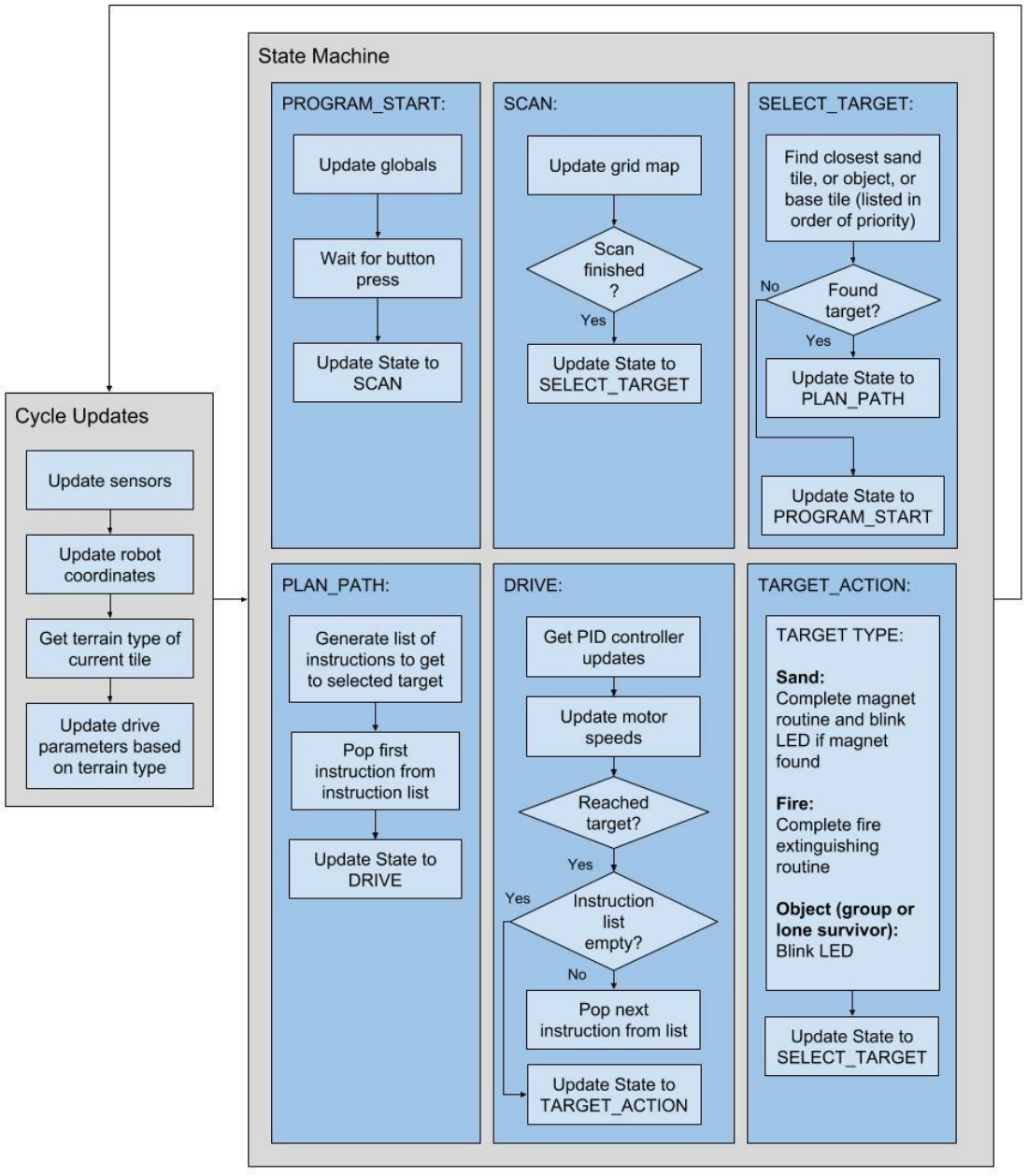


Figure 12: Final Software Architecture

Every cycle, the robot does the following:

1. Updates all sensor values
2. Updates current robot coordinates (i.e. location on the field)
3. Updates drive speeds and constants are updated based on the terrain it is currently driving in

Afterwards, the state machine is entered and the action for the robot's current state is progressed or completed. The completion of a state triggers the progression into the next state. The state machine process is as follows:

1. PROGRAM_START: resets all global variables and waits for a button press to proceed
2. SCAN: the robot captures distances to objects to generate a 6 by 6 grid map that contains the coordinates of all objects and terrain tiles
3. SELECT_TARGET: a target is selected using the generated grid map to select the next closest target coordinate
4. PATH_PLAN: an optimal plan to the selected target is formed and a list of drive and turning instructions is made
5. DRIVE: each instruction is executed from the path planner using PID controllers
6. TARGET_ACTION: the robot will complete an action based on the type of target it has travelled to such as a magnet detection routine, blinking an LED, or running the fire extinguishing routine
7. Return to the SELECT_TARGET state and repeat steps 3 to 6 until there are no more targets of interest left to visit, which returns the state machine to the PROGRAM_START state.

It should be noted that some of the functions of the state machine were demoed separately on demo day due to lack of time to complete a couple components. This is discussed more in section 7.0.

6.2 Grid Map Creation

Grid map creation is the first step in the robot's autonomous functions, and is integral to all processes that follow. The robot uses this grid map to assign itself field coordinates with its current location, keep track of the location of objects and terrain types, and plan paths through the environment.

At the beginning of the program, the grid map is pre-programed with the location of the terrain types, including sand, gravel, water, and flat tiles. Then, using the TFmini LiDAR and IMU, distances and angles are measured to locate the objects and calculate their grid coordinates, as seen in Figure 13.

The x and y coordinates of a detected object are calculated as follows in Equations 1 and 2.

$$x = \frac{(L-o)*sin(\theta)+x_{robot}}{304.8} \quad (1)$$

$$y = \frac{(L-o)*sin(\theta)+y_{robot}}{304.8} \quad (2)$$

Where x is the x-coordinate of the detected object, y is the y-coordinate of the detected object, L is the lidar reading in mm, o is the offset between the lidar and the middle of the tile in mm, θ is the yaw

angle of the robot, x_{robot} is the x-position of the robot on the field in mm, and y_{robot} is the y-position of the robot on the field in mm.

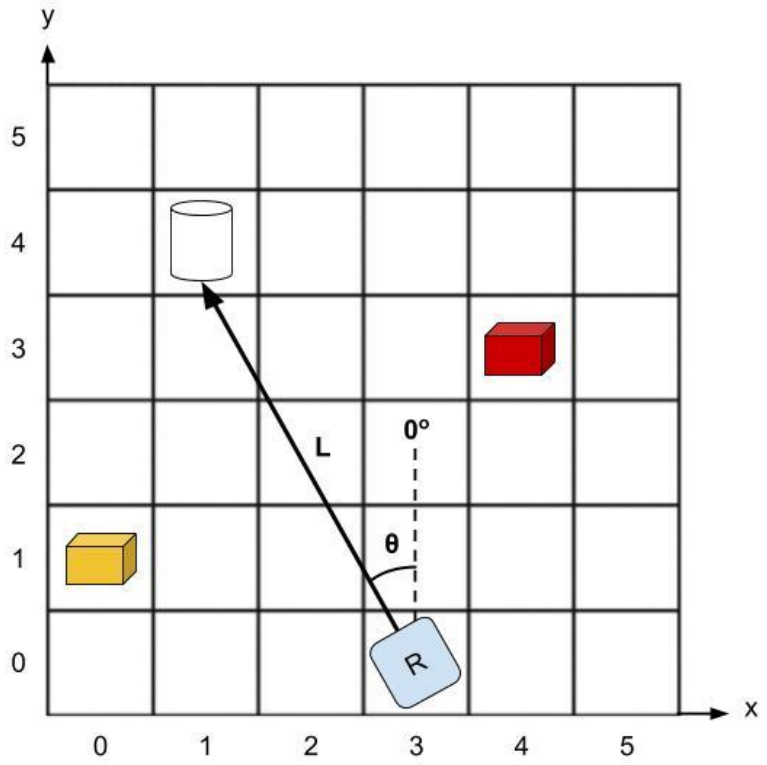


Figure 13: Visualization of Grid Map with Distance and Angle Measurements

Using Equations 1 and 2 above, the grid map is populated with detected objects as the robot rotates to scan the field. Similar concepts are used for localizing the robot on the field relative to the walls and the robot's current heading.

6.3 Path Planning

The objective of path planning is to provide the optimal path for the device to traverse between start and end positions within the tiled grid map. An optimal path is one that takes the least amount of time for the device to complete, but also takes into account the various tile conditions and their effects on device movement. Start and end positions are typically the device's current position, and one of the objectives (survivors, sand tile, candle, or an unidentified object).

It should be noted that the path does not contain the end position/tile. This is because the device will follow the path and stop one tile in front of the end position tile, facing toward the end position tile. The reasoning behind this is that in most cases we do not need to, or cannot fit safely, into the end position tile. For example, in the case of moving to and blowing out a candle, the device should not enter the candle's tile as it would risk tipping over the candle and spreading the fire. The only objective for which the device should enter the end position's tile is when searching sand for a magnet. However, this is handled by a special magnet scan routine which begins one tile over from the sand tile, as discussed in section 6.5.

Through testing and observation, a few constraints on the output path were derived. First, the resulting path must not contain any turns on gravel or sand tiles, as it would significantly reduce turning accuracy or push the robot off track. Second, the path must not contain any empty pits since the robot is not designed to traverse through them, and would likely become stuck. Third, the device must only move horizontally or vertically along the grid in straight lines. Traversing diagonally through the grid introduces too high a risk of getting stuck in a pit, and also makes localization significantly more difficult. Lastly, the output path must not contain any tiles with objects in them, otherwise the robot may collide with the object.

Additionally, it was observed that the device takes significantly longer to make a turn than it takes to go straight through additional tile spaces. Thus, a criteria for the output path is to reduce the number of required turns. This includes taking into consideration the device's initial orientation, since driving backwards is more optimal than turning twice. An example of this case is shown in Figure 14 below. The path on the left of Figure 14 makes an unnecessary turn at the start, causing it to take longer than the path on the right of Figure 14, which achieves the same result.

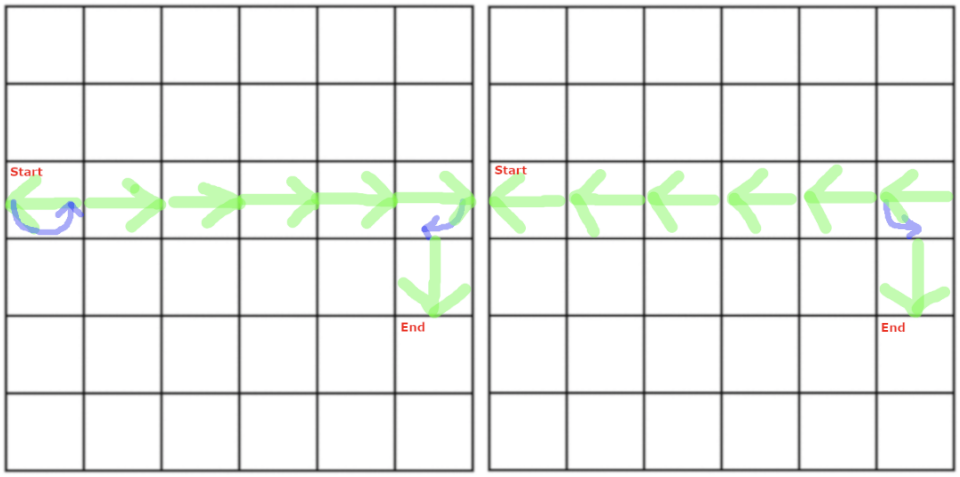


Figure 14: Unnecessary turn at start (left) vs driving backwards (right)

Path planning was implemented using a modified version of the A* algorithm. The algorithm takes as input a 2D array (representing the 6x6 tiled grid), start position, and end position. It outputs a list of steps which move the device from start to one tile beside the end position, facing the end position. A step is either a linear movement (e.g. MOVE 5 to move 5 tiles forward, MOVE -4 to move 4 tiles backward, etc) or an in-place rotation (e.g. ROTATE 180°, ROTATE -90°, etc).

A* works by associating a cost with each position, also known as a node, in the grid. A node's cost contains two cost factors: g(n) - the cost of reaching node n from the starting node, and h(n) - a heuristic estimating the cost of reaching the end node from node n. The algorithm does a best-first search from the start node, and will move to the next unvisited node with the lowest cost until reaching the end node or running out of nodes to visit. More information can be found on the standard A* algorithm's operation in [8].

In order to accomodate the aforementioned constraints and criteria, modifications were made to the standard A* algorithm. These include removing diagonal movement, adding cost penalties to turns,

adding extremely large cost penalties to turns in sand or gravel (to the point where they would never occur), and marking empty pits or pits with objects in them as untraversable.

6.4 PID Control

When executing move and rotation instructions provided by the path planning algorithm, the target distances or angles are fed into one of two PID controllers: drive PID or turning PID. The controllers allow the robot to travel to its target position with high accuracy, ideally eliminating overshooting the target position, and at the very least allowing for correction if overshooting occurs. The concepts for both controllers are the same, the difference is how the output from the PID is used and what PID constants are selected.

A PID controller is made up of three components, namely proportional, integral, and derivative terms, that are summed together to produce a single output based on the current error, the accumulated error, and the error trend respectively. The error being discussed is the difference between the current sensor reading and target sensor reading. These components can be represented as in Equations 3 to 5 below.

$$error = target - feedback \quad (3)$$

$$integral = (integral + error) * \Delta t \quad (4)$$

$$derivative = \frac{error - previous\ error}{\Delta t} \quad (5)$$

The results from Equations 3 to 5 are each multiplied by their corresponding PID constants and summed together to produce the output from the PID controller, shown in Equation 6.

$$output = error * kp + integral * ki + derivative * kd \quad (6)$$

The drive PID controller provides feedback from the TFmini LiDAR, using relative distancing from walls or objects to be able to determine how far the robot has traveled. The output from the PID algorithm is transformed into motor speeds between -100 and 100, where -100 is full speed backwards, 0 is stopped, and 100 is full speed forwards. Originally, the drive PID used feedback from the encoders, and the tuned PID constants were very similar to those calculated in the Position Controller Design Calculations memo. However, the feedback data source now comes from the TFmini LiDAR due to the removal of the encoders discussed in section 7.2.

The turn PID controller, on the other hand, provides feedback from the yaw of the IMU as a relative heading to the robot's starting orientation. The output from the PID algorithm is then transformed into motor speeds in a similar fashion to the drive PID controller, except one motor is given a negative of the other motor's speed to produce a tank turn to turn on around the robot's geometric center.

6.5 Magnet Detection

When arriving in front of a sand tile, the device carries out a routine to detect if the tile contains a magnet (food). This routine uses the device's BNO055 IMU magnetometer readings in the x, y, and z axes to detect the presence of a magnet.

The routine begins by wiggling the robot with two brief forward and backward motions in order to stabilize magnetometer readings. Without the wiggle, magnetometer readings had significant error during testing. Next the routine takes an average of the x, y, and z axes magnetometer values over 100 successive readings on the flat wood tile. It then moves forward onto the sand tile, and after a brief delay, computes a second set of averages over 100 readings.

Next the sum of absolute differences between respective axes averages from the first and second sets of readings is calculated. If the sum is larger than a threshold value, it indicates the magnetometer readings have significantly changed by moving onto the sand, thus indicating the presence of a magnet and notifies the users by turning on a green LED.

7.0 System Construction, Testing, and Modifications

While constructing and testing the device, many modifications were made to the initial design for the mechanical, electrical, and software systems of the device. Some of these changes were touched upon in previous sections, however this section will discuss in detail the most significant modifications and their corresponding ramifications.

7.1 Computation Platform Modifications

One of the largest changes that affected all of the robot's systems was the replacement of the Arduino Uno and Raspberry Pi with the Teensy. As mentioned in section 5.5, this replacement was due to the lack of processing power on the Arduino Uno. When too many sensors were being processed for data, the Arduino Uno was unable to keep up, resulting in the production of bad data. The Teensy has over 10 times the processing power of the Uno, and was more than sufficient for processing the sensors. The Teensy was unable to successfully interact with the Raspberry Pi, as the Pi would fail to detect the serial port that it was supposed to receive the sensor data on. The decision was made to use only the Teensy, as it likely had enough processing capabilities and flash memory to run our software logic and receive the sensor data, in a timely fashion.

The result of this was that all of the software code that was written on the Raspberry Pi in Python needed to be converted to the Arduino's C++ language. After these changes, all sensors and actuators were able to successfully be run simultaneously while receiving clean data from the sensors.

7.2 Testing Software and Mechanical Integration

During integration testing with the software systems, particularly when tuning of the drive and turn PID controllers, a number of issues were encountered with the mechanical design. These problems were due to static friction and center of gravity.

Static friction in the robot chassis caused major problems when trying to complete 90º turns and specified distances at low speeds. This resulted in a lack of ability to control fine movements and complete slow turns for initial scanning of the field. Additionally, PID was very difficult to tune with this amount of static friction since the motors were designed for high speed and low torque. This required a high impulse of speed to move the robot when it was not in motion, but barely any speed to maintain the motion, resulting in high amounts of overshoot or undershoot, depending on the tuning parameters. There was not enough time to change the wheels, motors, gearing, or general methodology of the controllers due to the need to continue testing other components of the software.
Examples of the effects of static friction can be seen in Figures 15 and 16. Figure 15 showcases the response from the drive PID controller to a request position of 1200 mm (i.e. the robot should drive such that it is 1200 mm from the wall in front of it). A small amount of overshoot occurs, about 25 mm, but it takes a long time for the PID to correct this overshoot since it cannot overcome the static friction easily once the robot is at rest, so it must wait for the integral term to build up.

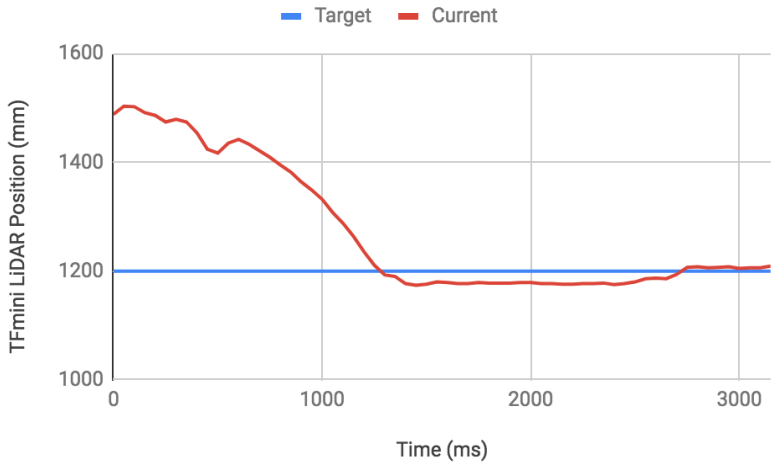


Figure 15: Response of Drive PID Controller With a Drive Request of 1200 mm

The second example, shown in Figure 16, showcases the response from the turn PID controller to a request position of 90º. Static friction is even higher when the robot is turning, since there is also a large friction factor from the rubber wheels on the tile that is not encountered when driving straight. This results in a large overshoot of 7º that is never corrected for. In order to fix this error, it was necessary to implement a “second try” into the software that doubled the PID constants to attempt to overcome static friction. This effect can be seen at around 1400 ms in Figure 16.

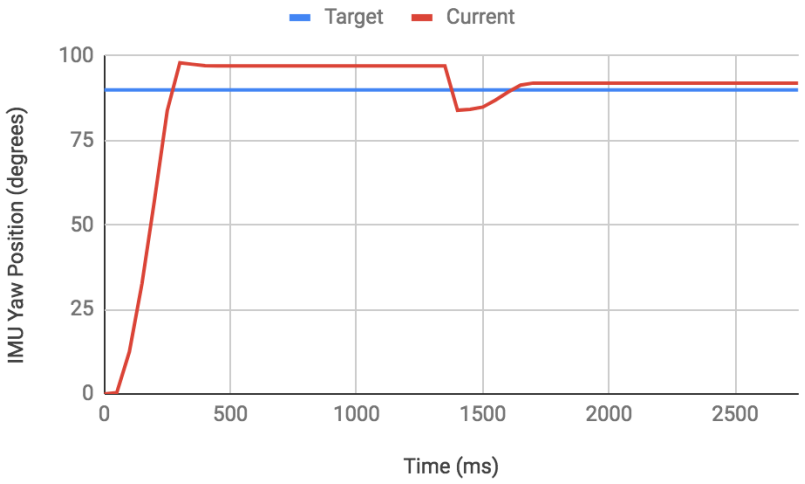


Figure 16: Response of Turn PID Controller With a Turn Request of 90º

Center of gravity was another major issue, specifically relating to accurate turns. Due to the positioning of the encoders at the back of the robot, the center of gravity shifted backwards. This resulted in turns that were not about the robot's geometric center, causing inaccuracies in the robot's motion and localization. The encoders ultimately needed to be removed because of this to redistribute the mass of the robot for more precise turns. After removal, distancing measurements were then taken from the TFmini LiDAR instead of the encoders. Lack of encoders for distancing measurements made it difficult to localize the robot on the field when other objects were obstructing the walls, since the LiDAR could see the wall or an object without being able to differentiate between them. With encoders, both LiDAR and encoder data could have been used to correct for these discrepancies and thus allow the robot to localize with objects on the field.

8.0 Demo Day Results

The final demonstration deadline for FEMA included a fully implemented mechanical and electrical system, with software mostly completed. The prototype was able to go over the required terrain, extinguish a candle, and detect the food in the sand. It was unable to navigate to the lone and group survivors, due to mechanical and software integration problems.

The robot was able to achieve all of the objectives except for navigating to the lone and group survivors due to the lack of ability of the robot to localize itself on the field when objects were present, which implies that the ability to navigate fully autonomously objective was not fully achieved. This was in part due to time constraints in the software and the issues described in section 7.0. However, each core component, including traversing through and around various terrain, searching through each sand tile and detecting whether or not the food was present, generating a map with the locations of all objects from the robot's starting position, and driving towards the candle and extinguishing it, were all successfully completed independently.

None of the constraints of the project were violated. The criteria were mostly met, with the exception of maintaining a reasonable cost, and creating a reliable system in certain aspects. The project exceeded the proposed budget by $572.40, largely due to malfunctioning parts, design changes, and

rapid prototyping. The reliability of the system was not achieved because of the inability of the selected drivetrain to turn consistently. The tested software was reliable, however, incomplete due to time constraints, which results in an overall unreliable system for the main goal of being able to traverse autonomously throughout the entire course and complete each mission sequentially.

9.0 Lessons Learned

There were many lessons to be learned throughout the process of this project. Most of the lessons learned are due to inexperience with creating a robot from the ground up. Creating a robot, as was discovered, is a significant time sink, especially since many things can go wrong that are not anticipated, which effects the timelines for constructing all systems from mechanical to electrical to software. Most of the work cannot be parallelized due to dependencies, which meant that duties had to be shifted promptly. For example, those working on software algorithms helped out with sensor bring up in the beginning, since they could not test their code. This means that constant communication is critical for ensuring that everyone is on track.

Additionally, many issues were encountered sensor debugging was necessary. This introduced learnings that when testing components it is important to isolate them from other components that may introduce error. Isolating components leads to understanding where the problem is occuring quicker, and ensuring that there is no need for repeated tests. During these stages of debugging, a lot of components were purchased with the necessity of saving time. The team learned that prices are higher for components that lend a hand towards convenience and rapid prototyping.

10.0 Conclusions

In conclusion, during the construction of the prototype California wildfire search and rescue device the engineers at Lera Inc. encountered many unexpected problems. This resulted in major delays despite saving time by using a pre-built RC car drivetrain, particularly resulting from problems encountered during sensor mounting and bring up. Consequently, some software components, such as the ability to localize when objects are present on the field, did not have sufficient time to be completed due to mechanical issues relating to static friction and center of mass.

Furthermore, the project constraints were satisfied, however the final design performs poorly by measure of the cost and reliability criteria. The budget exceeded the proposal by $572.40, and the behaviour of the drivetrain was inconsistent, severely limiting reliability when using PID control.

In practice, the prototype was able to navigate through the required terrain, extinguish a candle, and detect food buried in a sand pit. However, it was unable to navigate to the survivors due to mechanical and software integration problems resulting from the aforementioned delays, lack of encoders for distancing measurements, as well as the lack of precise movement offered by the drivetrain. Lastly, given more time to refine or swap the drivetrain and further test and develop software algorithms, it is expected that all of the remaining objectives would be completed.

11.0 Recommendations

Though this project was ambitious, it was a fun experience and the team learned a lot. However, the project was too ambitious for the time constraints, and required approximately 25 to 40 hours per week per team member or more to complete, which is significantly more time than the originally anticipated 8 hours per week per team member. The team believes that the project could have been scoped smaller, making it more reasonable within the given timeframe, and thus recommends working together with stakeholders more to appropriately scope the project from the beginning.

More communication between FEMA and Lera Inc. would have been beneficial, as it would have helped the project stay on track, and perhaps resulted in a higher chance of completing the entire project if the scope had changed correspondingly to the timelines due to frequent progress check-ins. An optional meeting on a biweekly basis between the team and FEMA, for example, would have given a better idea of progress as opposed to one construction check.

12.0 Works Cited

[1] M. Haberman, S. G. Stolberg and J. H. Davis, "Trump to Explore Venue Alternatives for State of the Union," The New York Times, 23 January 2019. [Online]. Available: https://www.nytimes.com/2019/01/23/us/politics/trump-state-of-union-pelosi-letter.html. [Accessed 29 March 2019].

[2] C. Chen et al, “California Wildfire Search and Rescue Device Proposal for FEMA,” 25 January 2019.

[3] K. Taylor, “Basic, Sturdy Wood Joints and When To Use Them”, 06 December 2018, [Online]. Available: https://www.mybluprint.com/article/basic-sturdy-wood-joints-and-when-to-use-them [Accessed 29 March 2019].

[4] Kabinet City, “PLYWOOD, PARTICLE BOARD, MDF, HARDBOARD…”, [Online]. Available: https://www.cabinetcity.net/2017/05/02/plywood-particle-board-mdf-hardboard-where-do-we-go-from-here/ [Accessed 1 April 2019].

[5] D.. Arduino DC Motor Control Tutorial - L298N | PWM | H-Bridge. 08 February 2009, [Online]. Available: https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/ [Accessed 29 March 2019].

[6] “Incremental Photoelectric Rotary Encoder - 400P/R”. 31 March, 2019, [Online]. Available: https://www.dfrobot.com/product-1601.html

[7] Product Overview. [Online]. Available: Available:https://www.digikey.ca/product-detail/en/TLE52062SAKSA1/TLE52062SAKSA1-ND/1283081/?itemSeq=288759322 [Accessed 1 April 2019]

[8] R. Wenderlich, J. Fradj, “Introduction to A* Pathfinding,” raywenderlich.com, 29 September 2011. [Online]. Available: https://www.raywenderlich.com/3016-introduction-to-a-pathfinding. [Accessed 29 March 2019].

Appendix A

Bill of Materials

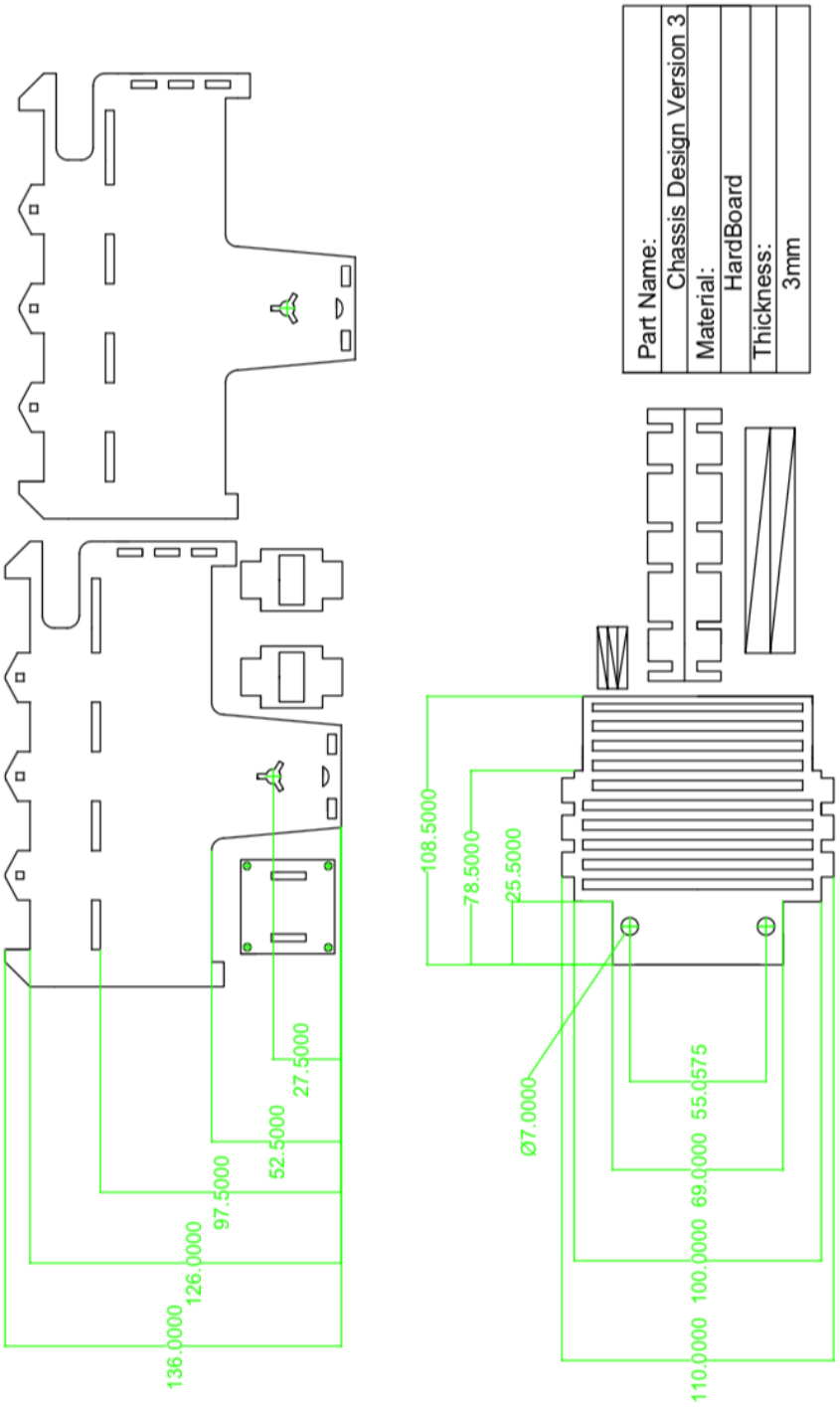
	Unit Cost	Quantity	Total Expense
Consumables			**$71.99**
Cable tie base	$0.23	20	$4.54
Connector	$1.74	3	$5.22
Desolder braid	$6.99	1	$6.99
Logic converter	$4.18	2	$8.36
MicroSD card	$16.99	1	$16.99
Short ethernet cable	$4.95	1	$4.95
Short USB to Micro USB	$4.95	1	$4.95
USB to ethernet adapter	$19.99	1	$19.99
Electrical Circuit			**$146.93**
Deans connector	$1.35	3	$4.05
Micro USB connector	$1.35	2	$2.70
Op amps	$0.56	6	$3.36
Opto isolators	$5.80	6	$34.80
Proto board	$2.87	1	$2.87
Resistors (assorted)	$0.04	45	$1.80
Surface mount to through hole adapter	$6.79	1	$6.79
Switch	$2.99	2	$5.98
USB to USB Mini cable	$9.39	2	$18.78
High gage wire (assorted)	$1.84	7	$12.85
Breadboard	$4.06	4	$16.24
Capacitors (assorted)	$0.12	36	$4.32
Pin headers (assorted)	$1.30	13	$16.90
MOSFETs (assorted)	$0.51	10	$5.13
Various jumper cable set	$10.36	1	$10.36
Microcontroller Units			**$111.23**
Raspberry Pi Model B+	$58.92	1	$58.92
Arduino Teensy 3.6	$52.31	1	$52.31
Mechanical System			**$55.51**

Fastening material	$8.68	1	$8.68
Hardboard	$10.00	1	$10.00
Hardwood	$5.00	1	$5.00
Motor for fan	$17.23	1	$17.23
Plywood	$5.00	1	$5.00
Spacers (assorted)	$0.80	12	$9.60
Miscellaneous			**$26.63**
LIPO bag	$18.99	1	$18.99
Raspberry Pi case	$7.64	1	$7.64
Power System			**$157.04**
Batteries	$46.00	2	$92.00
DC/DC converter	$12.29	1	$12.29
IC Motor driver	$6.91	2	$13.82
LIPO balancing charger	$31.05	1	$31.05
Voltage regulator	$3.94	2	$7.88
Sensors			**$268.16**
BNO IMU	$49.57	1	$49.57
Color sensor	$11.54	1	$11.54
Encoders	$38.89	2	$77.78
Gravity	$20.06	1	$20.06
Hall effect	$3.28	2	$6.56
MPU 6050 IMU	$14.47	1	$14.47
Opto sensor	$1.79	2	$3.58
Photodiode	$1.55	6	$9.30
TFMini	$60.31	1	$60.31
Set of fans (2)	$14.99	1	$14.99
FINAL TOTAL			**$837.49**

Appendix B

Chassis CAD Drawings

Subassembly Parts



Chassis Platform Design