List of Figures

List of Tables

1.0 Introduction

1.1 General Background

In a response to human accidents and forest fires occurring in national parks, the startup ‘Tree-80’ was started to create autonomous robots which would be able to patrol parks 24/7. Currently, there does not exist an efficient method to do search and rescue operations, with operations costing upwards of $500 conservatively, and can rocket as high as $1600/hr when helicopter support is required [1].

A search and rescue robot should ideally have several features in order to complete certain tasks. These are:

- Finding and extinguishing fires with passive sensors and extinguishing devices
- Safely and autonomously traversing different types of terrain with sophisticated mapping
- Locating missing persons and having the ability to provide them with survival material (food, etc.)

The company made several internal groups, each tasked with creating a prototype of the machine with the goal of building off the experience gained to fine-tune their final product for release.

After successful test runs at the company's Fred-Church field, employees were tasked with writing a detailed report outlining the successes and failures of the run, as well as lessons learned and steps moving forward.

1.2 Needs Assessment

A method is required to conduct land-based search and rescue missions. The missions entail putting out fires, finding and delivering food, as well as identifying and rescuing survivors. The product must be able to deal with a variety of terrain found in national parks included sand, gravel, steep valleys, and physical obstacles.

1.3 Problem

1.3.1 Problem Definition

To satisfy the needs, an autonomous device must be designed and built that is equipped to traverse the rough and unpredictable terrain of a national park. Additionally, this vehicle should be able to provide status reports to the respective search and rescue teams. As a step towards this solution, multiple teams have been tasked with designing and building prototype vehicles that are able to navigate a randomized search and rescue course, complete the missions stated in the following section, and return to the starting position of the course. Each vehicle must be capable of driving through sandy, rocky and steep terrain.

1.3.2 Objectives

The objectives of the system are defined as the following:

- Dimensions of the device must be limited to 0.2m x 0.2m x 0.2m (8" x 8" x 8")
- Cost of the project should be no more than $265
- Device should reach the initial start position after completing the missions
- Device should be simple in design for ease of modulation and debugging

An initial objective of having the device completely waterproofed was ignored following the removal of water from the course.

1.3.3 Constraints

Due to the environment the robot will be operating in as well as the time frame the project must be completed in, a list of constraints have been summarized as follows:

- The device must run autonomously through the search and rescue course
- The device is not permitted to leave the boundaries of the search and rescue site
- The device is not permitted to climb over walls
- Cameras and vision systems may not be used
- Flying devices are not permitted
- All missions must be completed within 10 minutes.

2.0 Summary of Project

2.1 Overview of Final Design

The final robot is an autonomous search and rescue robot constructed from a laser cut acrylic chassis mated to an array of sensors and equipment. The robot is driven by two motors with encoders coupled to six sorbothane wheels powered by a toothed belt system. A Teensy 3.5 was selected as the microcontroller over an Arduino Mega for processing considerations. The robot is equipped with an Inertial Measurement Unit (IMU), ultrasonic sensor, Infra Red (IR) Range Finder, IR Flame Sensor, Colour Sensor, and a fan with a nozzle. The ultrasonic and IR Range Finder were primarily used for object detection on the board, as well as assisting with robot localization. The combination of the IR Flame Sensor and the fan were used to detect and extinguish the candle flame. The Colour Sensor is used to differentiate the survivors. The device is powered using a 5V line for the sensors (via the Teensy 3.5) and two LiPo batteries; one for the motor drive and one for the fan. The microcontroller was powered with a laptop. Movement and localization relied on a combination of tracking the motor encoder counts, obtaining orientation data from the IMU, and obtaining IR distance readings for correction. Path traversal was achieved with the A* Algorithm. Two LEDs are attached — one is used to perform rudimentary tracking of the microcontrollers health, the other is used to alert everyone when an objective is completed. Figure 1 shows the the complete assembly of the robot.

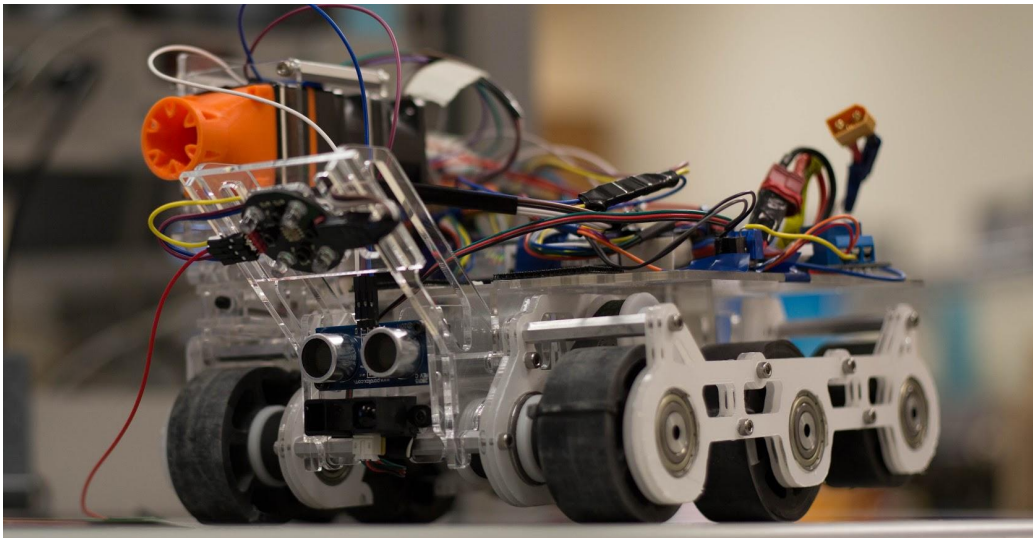


Figure 1: Complete Robot Assembly [2]

2.2 Key Features

The robot was unique in that it was the only drivetrain to feature a 6-wheel drive, which gave it unparalleled turn authority and drive versatility. A manual mode allowed for separate control of the robot without the need for autonomous control, and a Teensy controller gave it compact but powerful processing for software. Custom nozzle geometry with an integrated flame sensor gave the fan 4 ‘tiles’ of range, allowing for greater flexibility in firefighting, and innovative

pivoting sensor housings allowed for a large amount of sensors to be packaged in a small area. To ensure the robot moved in a straight line, motor control was done based on IMU heading. To help with consistent tile positioning, the IR and ultrasonic sensors were used to adjust the robot into the desired position, after most of the movement had been done by the motor encoders. Path planning was done using the A* Algorithm, with slight modifications to avoid hazards and other obstacles.

2.3 Matching of Objectives and Constraints

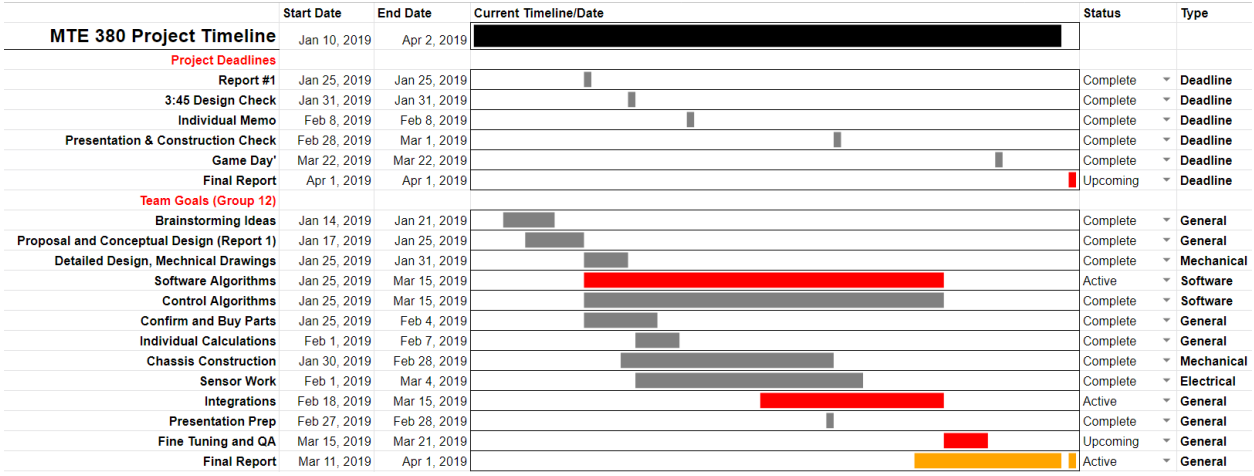
A mapping of the robot's design to the objectives and constraints defined in Sections 1.3.2. and 1.3.3 can be found in Table 1.

Table 1: Mapping of Objectives and Constraints to Final Device

Objective	
Dimensions (0.2m x 0.2m x 0.2m)	8.47in x 7.5in x 5in (0.22m x 0.19m x 0.13m) dimension slightly larger. Confirmed and allowed by Bill Owen. Achieved.
Cost is less than $265	Not achieved. The cost objective is discussed in further detail in Section 2.5.
Device returns to starting location	The robots movement utilized an in-memory map and kept track of its current coordinate to path plan back to start location. Achieved.
Modular and simple in design for debugging	Teensyhub was located in a protoboard where sensors, such as the IMU, could be easily added or removed. Mounts could be added and removed with ease. Power supply and circuit locations were non-static, easy to plug and debug sensors and circuitry. Mechanical redesigns generally involved iterating existing components and could be done quickly and smoothly. Achieved.
Constraints	
Fully Autonomous	The robot was controlled with the Teensy 3.5 Microcontroller. Achieved.
Boundaries	The robots movement utilized an in-memory map and kept track of its current coordinate to not go beyond boundaries. Achieved.
Climb Walls	Capability achieved although objective was ultimately abandoned.
Camera and Vision Systems	Cameras and image processing techniques were not used. Achieved.
Flying	The design did not incorporate any flying components. Achieved.
10 Minutes	All runs accomplishing targeted tasks finished in 10 minutes. Achieved.

2.4 Schedule

The gantt chart shown in Figure 2 was used to keep track of the project deadlines and tasks.



	Start Date	End Date	Current Timeline/Date	Status	Type
MTE 380 Project Timeline	Jan 10, 2019	Apr 2, 2019			
Project Deadlines					
Report #1	Jan 25, 2019	Jan 25, 2019		Complete	Deadline
3:45 Design Check	Jan 31, 2019	Jan 31, 2019		Complete	Deadline
Individual Memo	Feb 8, 2019	Feb 8, 2019		Complete	Deadline
Presentation & Construction Check	Feb 28, 2019	Mar 1, 2019		Complete	Deadline
Game Day'	Mar 22, 2019	Mar 22, 2019		Complete	Deadline
Final Report	Apr 1, 2019	Apr 1, 2019		Upcoming	Deadline
Team Goals (Group 12)					
Brainstorming Ideas	Jan 14, 2019	Jan 21, 2019		Complete	General
Proposal and Conceptual Design (Report 1)	Jan 17, 2019	Jan 25, 2019		Complete	General
Detailed Design, Mechnical Drawings	Jan 25, 2019	Jan 31, 2019		Complete	Mechanical
Software Algorithms	Jan 25, 2019	Mar 15, 2019		Active	Software
Control Algorithms	Jan 25, 2019	Mar 15, 2019		Complete	Software
Confirm and Buy Parts	Jan 25, 2019	Feb 4, 2019		Complete	General
Individual Calculations	Feb 1, 2019	Feb 7, 2019		Complete	General
Chassis Construction	Jan 30, 2019	Feb 28, 2019		Complete	Mechanical
Sensor Work	Feb 1, 2019	Mar 4, 2019		Complete	Electrical
Integrations	Feb 18, 2019	Mar 15, 2019		Active	General
Presentation Prep	Feb 27, 2019	Feb 28, 2019		Complete	General
Fine Tuning and QA	Mar 15, 2019	Mar 21, 2019		Upcoming	General
Final Report	Mar 11, 2019	Apr 1, 2019		Active	General

Figure 2: Snapshot of Gantt Chart Post Construction Check

The first half of the project adhered to the schedule very well. The second half of the project consisted of software and integration tasks which relied on the sensors to be calibrated, finalized, and working without issue. This was not the case. Throughout the process, there were delays in essential components (re-purchases), sensors were completely replaced due to failures and the mechanical build continued to change because of this. As a result, pieces of the software needed to be continuously re-written to accommodate changes. This meant that the software and integration components were never fully complete. Furthermore, there was very little time to perform actual fine tuning and full QA of a device which could perform all the tasks.

2.5 Budget

The initial plan the team had regarding budgeting was to split the costs of each subsection as follows:

- SENSORS: $30 - $60
- CHASSIS: $70 - $100
- BATTERY: $20 - $50
- DRIVE SYSTEM (wheels, treads, etc): $100 - $150
- MISC (tools, extra parts, printing, etc): $0 - $15

Due to unforeseen circumstances, poor initial budget projections, and changing constraints, the final cost of the project exceeded $1000 (see the BOM in the Appendix via Figure A1).

The bulk of the cost came from three main sources: shipping, redesigning, and damaged parts. Due to the time constraints and parts coming from the US, shipping was expensive. For many purchases, shipping was about 30 - 50% of the cost — sometimes even exceeding the cost of the part. Removing shipping alone would account for over $200 of budget.

Due to changing project requirements and unexpected mechanical issues, the physical robot was redesigned and re-cut multiple times, bringing the total mechanical cost to approximately $450. Finally, damaged parts such as the microcontroller had to be replaced, costing approximately $250.

Other common prototyping parts such as solder and header pins, cost about $70. The team spent about $700 due to the hiccups/failures/redesigns and shipping. The best way to reduce cost for future projects would be to to use the MESS lab for parts, plan ahead and with other groups to eliminate shipping costs, and to share the cost of minor parts across the entire class.

3.0 Construction Analysis

The final product differed significantly from the robot initially envisioned by the team. Due to changing problem constraints, unexpected issues, and design changes occurring throughout the build, the robot had to be iterated extremely quickly. This section will examine the final iteration and the modifications along the way to reach the final design.

3.1 Mechanical

A ‘good’ mechanical design would be one in which the chassis does what it was meant to on game day, and is easy to change and adapt to new challenges. The final iteration of the chassis was fairly capable, with a mild drive dynamic, fairly predictable climb, excellent turn authority, and good capacity. However, it was hampered by poor symmetry, and rocky rolling, which made precise autonomous navigation difficult. Mechanical changes to the robot continued well into the build season and final testing days. Several sensor changes, driving dynamics issues, and the constant rule revisions forced significant development to occur fairly late into the timeline. Notable changes included redesign of almost all sensor housings, significant geometry changes to the chassis side plates, wheel diameter revisions, and tire grip modifications. This section discusses the design intent behind the robot and its predicted performance, and explanations as to whether or not these features worked as intended in real life.

3.1.1 Chassis and Drivetrain

The chassis, which includes all drivetrain components, was changed significantly over the course of the project, with almost all changes being driven by a desire for more grip. Grip was the most important challenge, since the correct use of grip determined the robot’s ability to climb, traverse adverse terrain, and turn effectively.

3.1.1.1 Chassis Plates

The chassis plate dictated the geometry of the strengthening standoffs, as well as the wheel locations. Therefore, they had immense importance in determining the efficacy of the chassis strength, scalability, and drive dynamics. It was discovered early on that the chassis did not have sufficient ground clearance to effectively surmount pit obstacles. Therefore, two large arcs of material (Figure 3) were removed from each of the 8 chassis plates to allow the ledges to intrude into the area between the axles, improving its adverse terrain driveability. The chassis was also found to be significantly stiffer than initially expected, and thus several standoffs were removed from the bottom-most row. This further increased the robot’s ability to drive over rocky and sandy terrain without being beached. The subsequent changes to the hole provisions on the plates allowed for a higher robot less affected by micro-elevation changes associated with rough terrain.

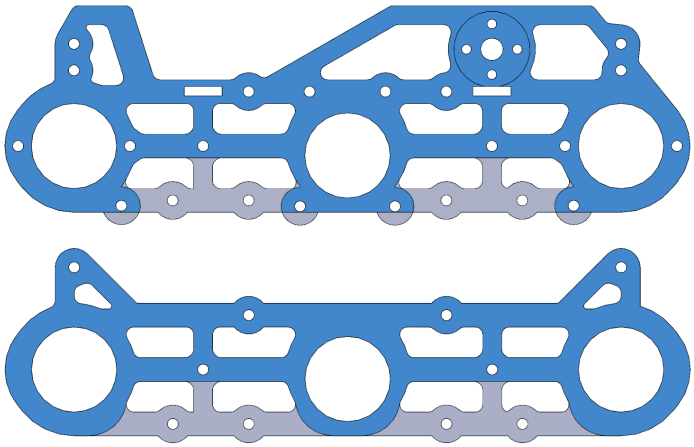


Figure 3: New Chassis Geometry (Blue) Overlaid on Old Chassis Geometry (Grey)

3.1.1.2 Wheel Modifications

The wheels, as a component, make the single greatest contribution to the mechanical grip offered by the drivetrain. As such, their proper operation ensures effective drive dynamics during the robot's operation. Originally, six matching 60A durometer rubber wheels were selected. The compound was the hardest out of three selections (40A, 50A, 60A) and was chosen for its resilience and the initial assumption that there would be no need for a significant amount of tire grip. Harder wheels deform less than softer wheels, and therefore do not 'grab' the terrain as much, firming up the drive characteristics and decreasing drag and power consumption at the cost of less grip. However, it was found quickly that the slick, hard wheels did not offer enough traction for the robot to climb out of pits or traverse quasi-fluid terrain, such as sand. They did not comply on impact, and tended to slip as the contact patch of the tire decreased, such as when on rocks, sand, or the lip of a ledge. Since slick wheels obtain grip by maximizing the amount of contact area they have with the ground, they lose grip quickly and completely when on terrain that is not completely flat.

Therefore, to improve adverse terrain grip, faux obverse treads were added using zipties (Figure 4). Similar in principle to ice chains on car wheels, the zipties allowed for a physical 'ledge' with which the rotation of the wheel could use to directly transfer force to the ground. The robot would therefore 'clamber' over rocks and ledges. However, since the treads were obverse, they compromised ride smoothness every time the wheel rode over them, as they were interruptions in the rotational surface of the wheel. Therefore, too many zipties actually reduced the robot's ability to drive smoothly on smooth terrain, and caused the robot to jerk, often violently, and veer off course as the wheels lost contact with the ground on every bump. Through testing, a suitable compromise between ride security and off-road ability was found to be 2 zipties per wheel, or one interruption per half rotation. This allowed for significantly better traversal of rough terrain while maintaining smooth terrain ride quality.

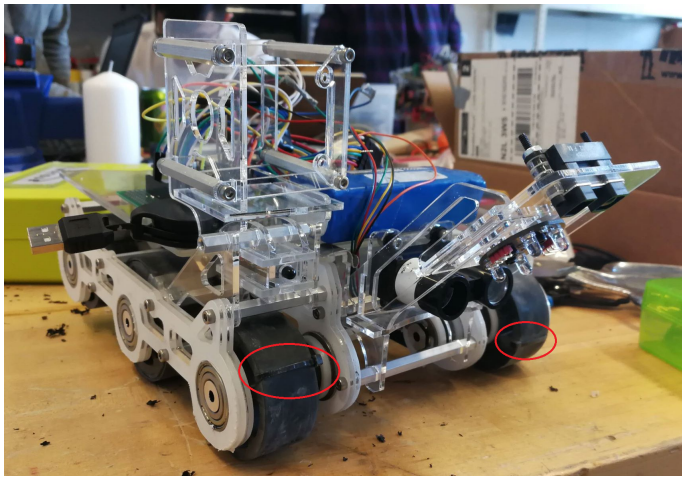


Figure 4: A Late-Model Robot Showing Front-Mounted Zipties, Circled in Red

It was also discovered that a significant reason as to why the robot could not climb ledges was that the front wheel centerline was not over the top line of the ledge. This chassis geometry prevented climbing as the wheel was not able to create positive contact with the top of the ledge. In order to move the centerline of the front wheel above the ledge, the center wheel was enlarged by ¼in. in diameter. This allowed the robot to tilt back enough to rest the front wheels at the desired level, which then permitted the robot to climb.

3.1.1.3 Other Chassis Notes

Other aspects of the chassis, such as the bearings, shafts, rigidity, and belt spacing were not changed due to their great performance and unchanged operational principles. The overall dimensions of the robot did not change, and so the belt spacing and shaft lengths also did not need to change. The shafts were at no point observed to be under excessive stress and thus their diameter was considered to be sufficient. The rigidity of the chassis overall was excellent, and the robot did not bend under load. It also did not bend under torque, which allowed for excellent belt grip and subsequently a very predictable drive behaviour. The bearings, which were lightly shielded, held up well against foreign intrusion. No sand or grit was able to penetrate the bearing race, even under heavy use during testing throughout the term.

3.1.2 Mounting and Periphery

Periphery encompasses sensors and other equipment on the robot, which this section discusses in a mechanical view.

3.1.2.1 Periphery Design

A major driving force for the periphery design was the initial requirement for waterproofing. Motors, sensors, and the top deck were all placed at elevated locations above the chassis, with first-generation enclosures being designed to be totally waterproof (Figure 5). Similarly, the top deck was initially designed to accept a waterproof cover to protect the main boards and batteries. However, upon the removal of water, the top deck was kept open to allow for easier access to the electronics, and all water sealing was removed, as seen in Figure 6.

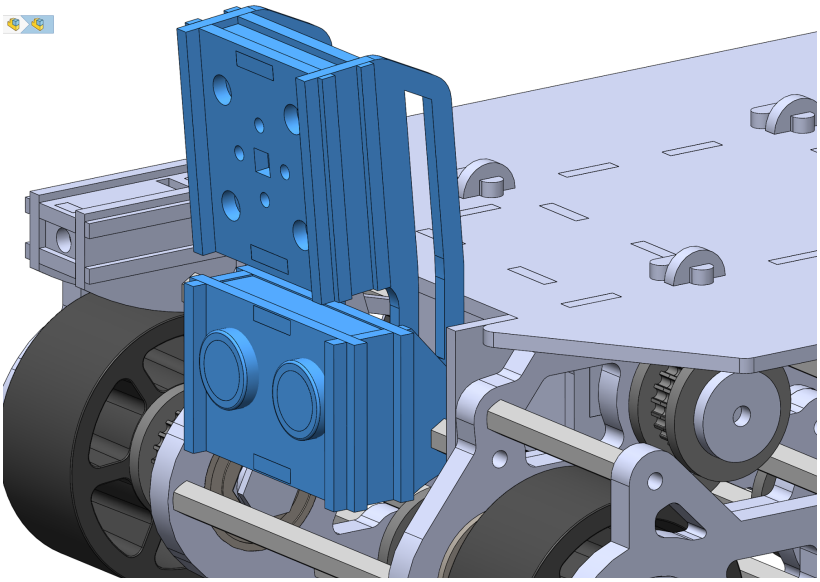


Figure 5: Early Enclosed Sensor Housings, shown in Blue

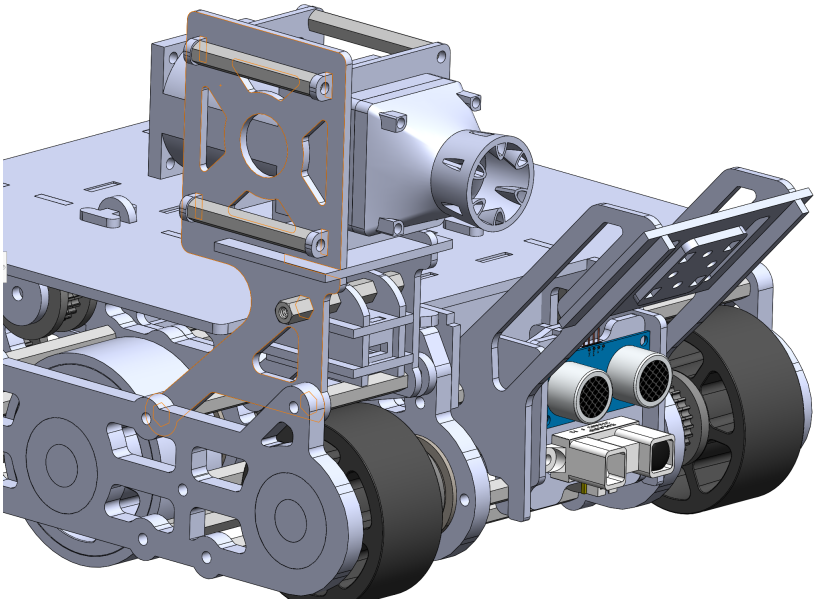


Figure 6: All Final-Generation Production Sensor Mounts and Open Top Deck

3.2.1.2 Sensor Housings

Sensor Housings were created in order to ensure repeatability in sensor operation, as well as to combat shock and operational wear. The sensor mounts were designed to be lightweight and adaptable to each sensor's usage and conditions. Initially, the sensors were designed to be waterproof to allow the robot to traverse any terrain, including water pits. However, following the changes in regulation, later iterations of the sensor housings focused more on repeatable alignment instead of water resistance. Sensor locations were dictated by their operating envelope, and housing design had to follow these requirements. The colour sensor, for example, had to be 4in above and 1.5in forward of the main front-facing sensor package. Since

this would place this outside the permitted starting volume of the robot and make it susceptible to accidental damage from impact, the housing was designed to rotate upwards when pushed.

Some changes were also made due to equipment changes. The main front-facing LIDAR, for example, mysteriously ceased operating the night before competition, which necessitated a total redesign of the front sensor package. Iterative improvements were also made to the sensor housings to allow for effective operation of the sensors. For example, the larger center wheel tipped the robot further than initially calculated, and thus certain sensor mountings had to be rotated backwards to allow them to point straight when at rest.

3.1.2.3 Fan Mounting

The fan mount was raised above the other sensors to keep the fan out of the debris on the ground, ensure that clean air was always available at the intake, and to allow for the output airstream to be at the height of the candle flame. Initial tests showed that the fan put out a significant amount of air, but turbulence and vortex induction quickly reduced the power of the airstream past roughly 2 feet. In response to this, a nozzle and flow straightener pair were designed, to be mated to the output of the fan. The flow straightener consisted of a 1in extruded grid of 4mm hexagonal holes. These holes captured air and reduced large-scale vortices induced from the oscillatory fan blade rotation. Once the flow had been ‘cleaned’, a reduction cone of about 75% was introduced to the airflow. This reduced the net flow of the air, but greatly increased its pressure and its cohesion over long distances. However, the large pressure differential between the ambient air and the high pressure nozzle air produced small eddy vortices, creating noise and further sapping the range of the airflow. Therefore, six corrugated lobes were placed leading from ambient air into the flow stream (Figure 7). These lobes served to draw low pressure air into the high pressure flow via the Venturi Effect, smoothing out the transition between pressure regions and further increasing the range of the nozzle [6].

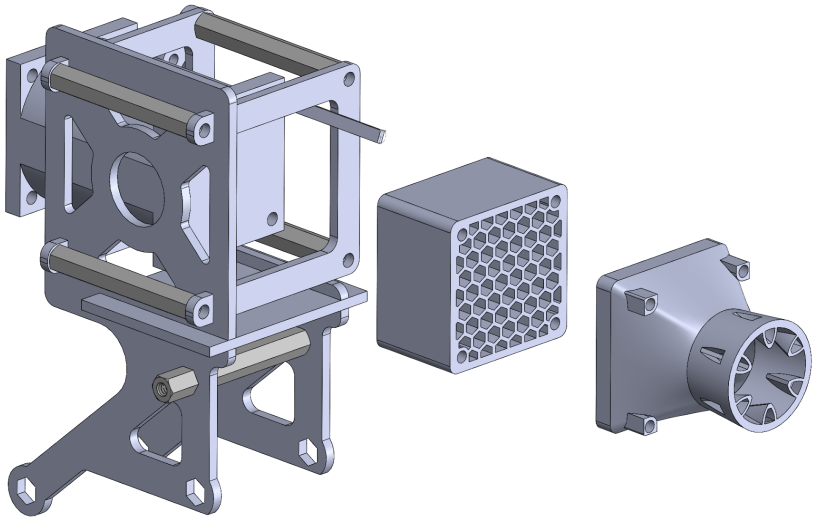


Figure 7: Exploded View of Nozzle Segments

3.2 Electrical

The main setup of the circuit has continuously evolved due to complications with sensors, a fried microcontroller and shifting project specifications. Notable changes include a microcontroller hub that houses the Teensy and the IMU, the addition of a relay to control the fan and the removal of the standalone 5V sensor line circuit. Figure 8 below shows a high level diagram of the final state that was used during game day of the robots circuitry.

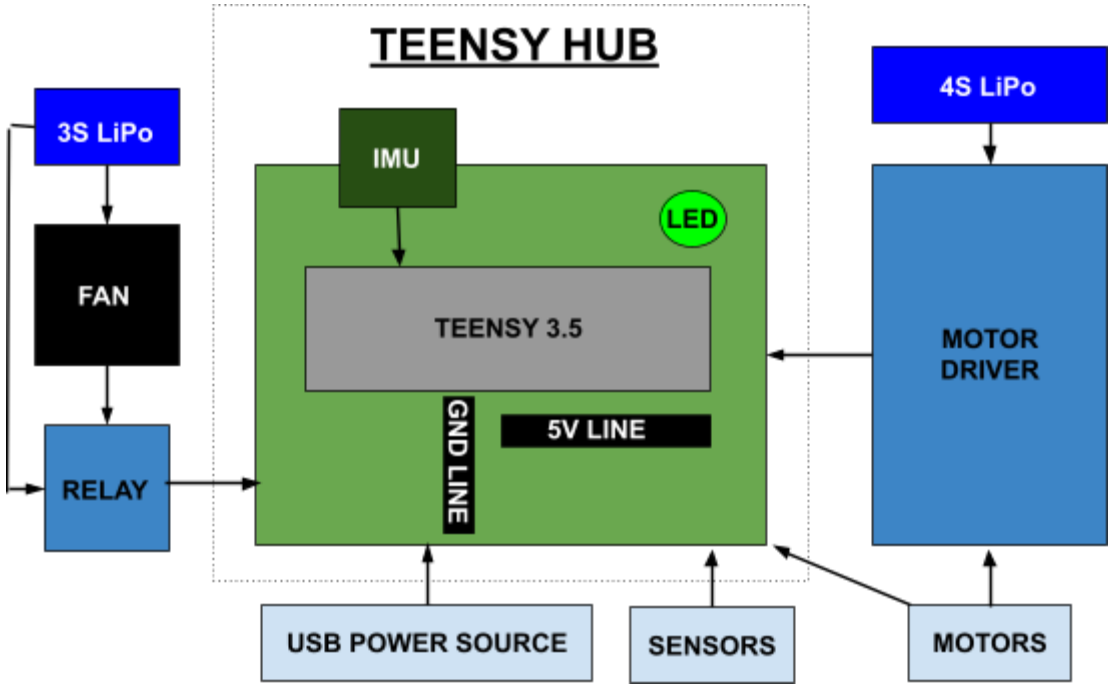


Figure 8: High Level View of Robots' Circuit

This section is split into three parts that delve into the microcontroller setup, sensor hookups and high-power component usage.

3.2.1 Microcontroller Setup

3.2.1.1 Microcontroller Selection

The initial plan regarding the path planning and sensor integration was to use interrupts to perform readings while using sensor fusion with the motor encoders and rangefinders to track the robots path and location. Based on preliminary calculations, the 16 MHz clock speed provided by the Arduino Mega 2560 was deemed insufficient. The Teensy 3.5 was selected as an alternative microcontroller due to its clock speed of 120 MHz, its 5V tolerant pins and the Arduino IDE compatibility provided via the Teeensyduino driver.

3.2.1.2 Teensy Drawbacks

Despite the perks given by the Teensy, there were some drawbacks that had to be resolved using active and passive components in secondary circuits. One such circuit required a level shifter for the flame sensor, due to the fact that the pins & ADC on the controller, although 5V

tolerant, only had a 3.3V resolution. This meant that the Teensy could not read the flame sensor correctly, as the ADC would max out at the sensor's 66% reading. The Teensy also does not have built-in pull-up resistors to the I2C line, meaning that resistors had to be specced and soldered onto the board to act as pull ups to the I2C lines. Figure A2 shows a model of what the required circuits for both the I2C lines and analog signal attenuation would be.

3.2.1.3 Teensy Hub

Unlike the Arduino Mega 2560, the Teensy board does not come with any male or female headers, making sensor integration impossible. As such, the microcontroller had to be soldered onto a protoboard with header pins to allow for said integration. This protoboard was also where the resistors connected to the I2C lines were soldered on. Following a Teensy accidentally being shorted, an LED connected to the 3.3V line was added to the hub as a means to monitor the health of the board, as determining if the previous Teensy was fried or not was a long process.
Figures A3 and A4 show the Teensy protoboard hub, where all sensors and other electronic devices connected to.

3.2.2 Sensors

This section looks at the configuration and design of the five different sensors that were employed. Due to the volume of electronics and how they were configured with the Teensy, careful pin selection was necessary to make the debugging experience of the electronics as smooth as possible. Section 3.3.2 details the sensor configuration and integration. All the sensors were powered off of the Teensy via a 5V DC line.
This section does not look at the laser sensor as it was not used on game day; however, it used I2C communication and was powered with 5V. The sensor is further explored in Section 3.5.3.

3.2.2.1 IR Range Finder

This sensor was used to detect distances within the range of 20cm - 150cm [3]. It took in three pins; 5V, ground and signal. Unlike many analog sensors, the output signal of the sensor was not linear and instead followed a plot [3] whose output never exceeded 3.3V. As such, a level shifter circuit was not required for this sensor. This sensor was used as the long-range sensor.

3.2.2.2 Ultrasonic Transducer

Although this sensor could read distances over 3m [4], due to the height that it was placed in our robot, its effective range was only approximately 40cm. As such it was used as a short range sensor, with the IR taking over when the distances the robot was reading exceeded 30cm. This sensor took in 5V, ground, and one pin for both triggering and echo reads. No level shifting was required.

3.2.2.3 9 D.O.F Inertial Measurement Unit

The IMU is a sensor that incorporates multiple ICs and sensors that each do different things. For the purpose of this project, the magnetometer and orientation sensor of the IMU were used. Unlike many of the other sensors, the IMU is powered with 3.3V, which the Teensy provides. The other pins connected to the hub include ground, SDA and SCL, with the latter pins being signal pins that communicate with I2C. One consideration with I2C is wire length, as long wires may exceed an I2C lines maximum capacitance. Level shifting was not required.

3.2.2.4 Color Sensor

The color sensor was used to detect the lego houses and was also used as a means to identify the different terrains, such as gravel and sand. The Teensy took 7 pins from this sensor: 4 signal pins for each of the phototransistors, 1 for frequency scaling, and 2 for power and ground [5]. Due to all the signal pins being digital, no level shifter was required.

3.2.2.5 IR Flame Sensor

The flame sensor is a linear, analog sensor. Because of this, a unidirectional level shifter was required. In order to reduce the 5V output it has into a 3.3V output that the Teensy can read, the team opted to create a voltage divider circuit within the Teensy hub. Doing the calculations resulted in the following relationship for the resistors: $2\mathrm{R1} = \mathrm{R2}$. Resistors to meet the specs of the relationship above were acquired to do a level shift for the sensor. 1 pin was used for 5V, one for ground and the final pin for the analog signal. The sensor's main purpose was to find the flame.

3.2.3 High Power Components

This section deals with the electrical components that have high power requirements (defined for this application as 10+ V and 100+ mA), such as the motors and fan. For both of these components, complementary circuits that were mostly detached from the Teensy had to be used to power and control them.

3.2.3.1 Motor

The robot used two 75:1 gearmotors, each requiring 12V and 200 mA of non-stall current draw [6]. In order to power them, a 3S LiPo was connected to a dual-channel motor controller. The motor controller allows the Teensy to control the motors using PWM signals to set speed and direction. The driver has signal pins that can connect directly to the Teensy to do the above.

Besides motor control, reading motor encoder values was also important. The selected motors came with built-in encoders, requiring 5V power and two pins for reading. A total of eight pins solely used for the purpose of encoder reading were linked directly with the microcontroller hub.

3.2.3.2 Fan

The selected fan for the project was a tubeaxial 12V DC fan that came with three pins. Two of these related to power and the third was a signal pin for the built-in tachometer. The fan could not be controlled directly from the Teesny using PWM and a MOSFET circuit was initially created to try and provide this control. The circuit was built using an n-channel FET as well as a flyback diode to control the on-off state of the fan. However, due to the tight timeline, not many transistors were readily available that met the required specification. This resulted in the fan signal strength being significantly attenuated when controlled by this MOSFET circuit. To rectify this, the FET configuration was swapped for a prefabricated relay. This relay allowed the fan to be controlled by a digital signal switching its state from normally open to common, operating the fan at full power. Note the fan was powered using an independent 3S LiPo to avoid overdraw.

3.3 Controls

Control systems encompasses tuning the actuated parts of the system, filtering sensor noise, sensor fusion, and damping electrical and mechanical inconsistencies in the system. As such it is a fairly complicated and interdependent problem.

3.3.1 PI Control

In the preliminary configuration there was a strong suspicion that the motors would not have behaviour that was close enough to ideal to operate without correction. The initial characterization tests showed that the motors were too slow to respond to the command input due to the weight of the substantial waterproofing done on the chassis. Figure 9 provides an example measured vs expected result of test conducted to characterize the motors.

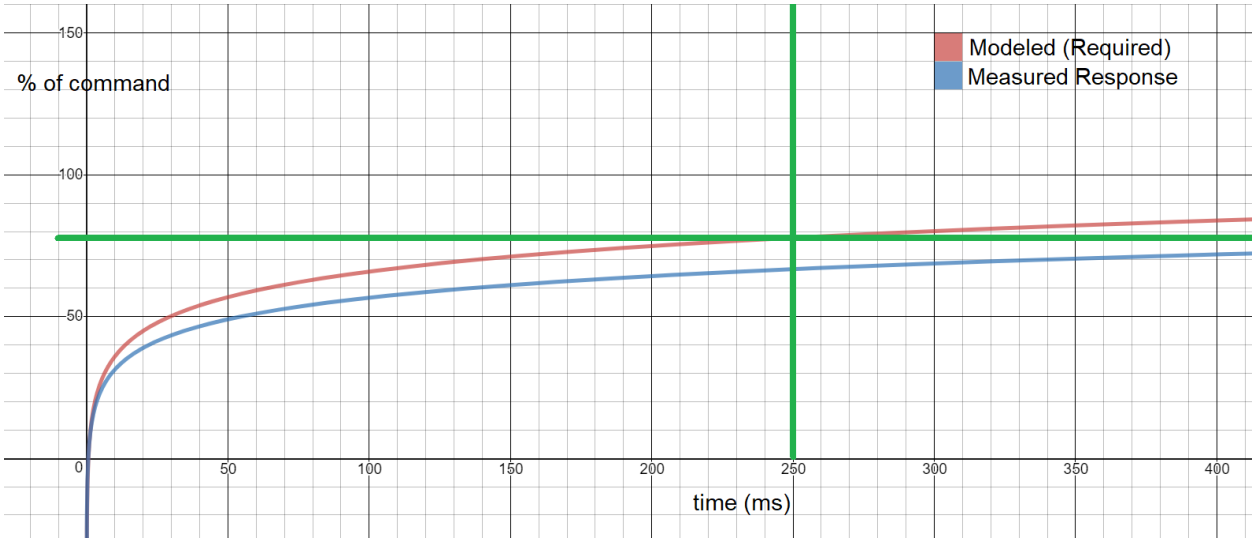


Figure 9: Measure vs Required Response

As seen in Figure 9, the measured response significantly lags the expected response. The bold axis indicates tau or ~77% of command input. To compensate for this, a proportional control package, shown in Figure 10, was fully designed to bring the output to the required response.

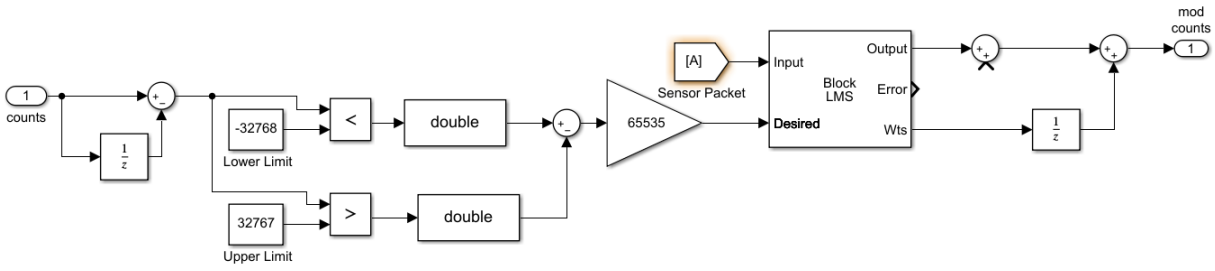


Figure 10: Rollover Quadrature Compensation

This aforementioned compensation model was eventually abandoned when the scope of the project was reduced to exclude water. The robot was stripped of the excess weight to reduce power consumption of the batteries and help reduce effective nonlinearity due to rocking. The existing compensation had to be swapped for a PI compensation model after recharacterizing the motor for the new configuration.

The new configuration had to modify it's assumptions on the key design features; the model had to account for slippage, overdraw and potential drift in the encoders. Overall this was a much more expensive and complicated redesign as shown in Figure 11.

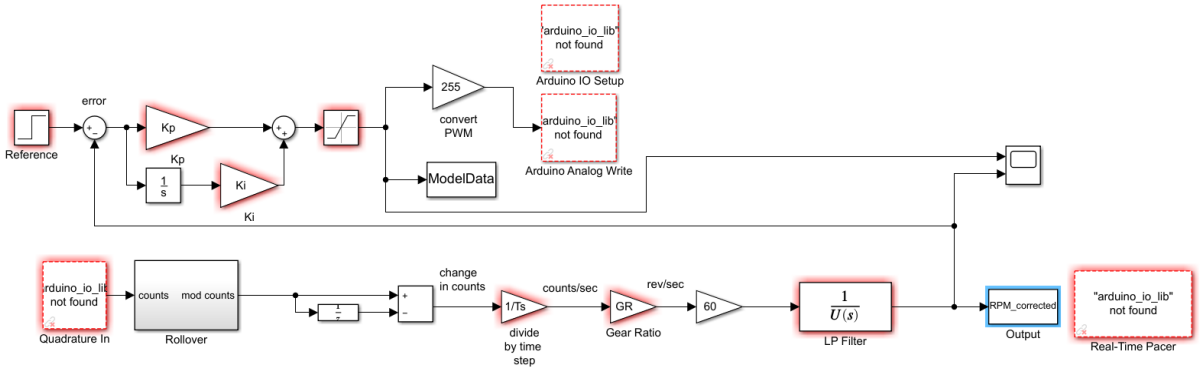


Figure 11: PI Controller + Rollover Compensation
(red blocks due to hardware not being available at time of capture)

The compensator pictured in Figure 11 depicts the final PI controller. Although the tuning resulted in significantly better performance as shown in Figure 12. Due to a lack of time and higher priority tasks, the controller was not re-architected into the software in time for demo day.

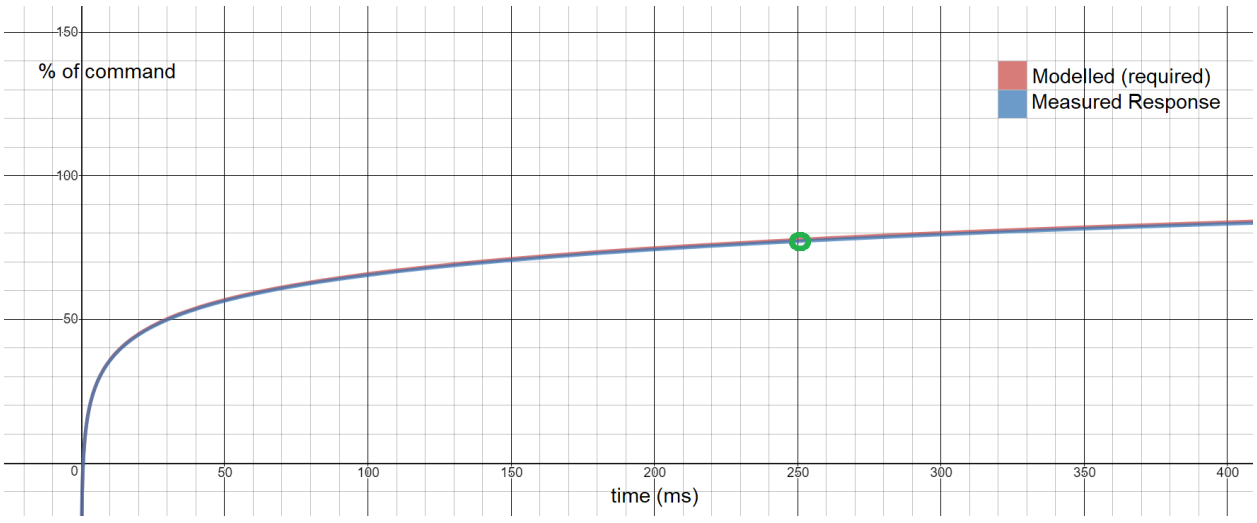


Figure 12: Measure vs Required Response

3.4 Motor and Sensor Integration

While the initial design of the robot depended on using interrupts to do all of the sensor reading, this idea was scrapped when it was discovered that the Teensy did not have enough interrupt timers. Sensor reading moved to periodic polling instead. While some of the sensors could be read with an `analogRead`, many were more complicated. This section will provide an overview of challenges integrating various sensors with the Teensy.

3.4.1 I2C

Both the IMU and the Laser Sensor communicated with the Teensy through I2C. Due to electrical requirements mentioned in Section 3.2.2, the second and third I2C lines needed to be used instead of the default one. Both sensors' libraries needed to be modified and recompiled to support this.

3.4.2 IMU

The IMU had 9 degrees of freedom, and did sensor fusion onboard, providing an output angle in degrees for each of the X, Y, and Z axes. It also self-calibrated, with a heading of 0 degrees each time it was turned on. The IMU was largely consistent, but a filtering function was required to deal with random deviations. IMU readings were ignored if it was more than a certain distance away from the previously calculated angle.

3.4.3 Laser Sensor

The output of the sensor had a large variance, with values being returned that were often up to $\pm$10cm from the average value and occasionally significantly more than that. This variance was dealt with using a combination of filtering and averaging. A function was written to take in a predetermined number of sensor values, calculate the mean of those values, and then recalculate the mean, ignoring values that were more than one standard deviation outside of the initially calculated mean.

Sensor values were also highly nonlinear under 1m of range. Sensor readings were taken at various distances inside of 1m along with their associated expected distances and a function was fit to convert these measured distances into expected ones. The function fitted was an 8th degree polynomial which worked well theoretically, but had issues on the Teensy due to loss of precision errors. The response from the laser (initially in cm) was converted to meters, and then fit to a function that turned it into centimeters to increase coefficient size. The result was then divided by 100 to return the adjusted distance in meters.

This combination of filtering and adjusting the laser sensor values yielded a final variance of $\pm$2cm. Due to the failure of the laser sensor, distance measuring responsibilities were instead taken over by a combination of the IR sensor and the ultrasonic sensor.

3.4.4 Ultrasonic and IR Sensors

While the IR sensor was initially intended and calibrated to work in tandem with the colour sensor to sense holes, sand gravel, rocks, and houses, the IR and ultrasonic sensor needed to be remounted and refitted to replace the laser sensor.

The IR sensor works by shining infrared light into its surroundings and determining distance based on the brightness of the reflection. While this reading was accurate between 20 and 150cm, distances between 10cm and 15cm would return the same voltage as distances between 15cm and 23cm, making any measurement under 23cm unreliable [3]. Additionally, pointing the sensor at a light source such as a candle would result in the sensor returning a voltage equivalent to a distance of 0. A function was fit using the same calibration method as in Section 3.4.3. To deal with the problem of unreliable distance readings below 23cm and when the sensor was pointed at the candle, the IR sensor was used in tandem with the ultrasonic sensor. If the IR sensor read a distance below 30cm, the ultrasonic sensor would be used for distance instead.

The ultrasonic sensor sends a signal and waits for an echo from the target. The initial delay between calling the function and receiving a distance reading was about 700ms, but this was reduced to 400ms after adding a 300ms delay inside the function. The beam also had significant spread, meaning that sound waves emitted could only be treated as a beam for distances under 50cm, limiting the effective range of the sensor to that distance. This ended up being a non-issue when the ultrasonic sensor was used in tandem with the IR sensor.

3.4.5 Movement Corrections

The first challenge encountered with robot movement was the ability for the robot to turn to a specific angle, and then drive in a straight line from there. As mentioned in Section 3.3.4, the robot could only turn to within +-3 degrees of a given angle. Additionally, driving both of the motors at the same power level in software would cause the robot turn slightly. This issue was made significantly worse due to sand and dust from the course getting into wheels and the drivetrain.

To deal with this problem, movement functions were augmented with a check of the current heading with the IMU, and update the motors accordingly. This combination of features allowed for fairly straight and consistent movement.

The second challenge encountered with robot movement was the ability for the robot to drive a specified distance (ex. 1 tile). While this challenge could in general be solved by counting encoder ticks, this solution would end up with a fairly large amount of cumulative error which would get worse the more the robot moved. To deal with this problem, the IR/ultrasonic sensors were used to correct the position of the robot based on expected distance from the nearest object or wall on the robot's current heading.

This combination of solutions allowed the robot to move around the course fairly consistently.

3.5 Software

3.5.1 Teensy

Most sensor drivers and libraries were made for Arduino's and due to the Teensy's similarity to an Arduino, most of it was compatible. For the IMU driver and libraries for additional data structures, individual modifications were required for compilation. Furthermore, the Teensyduino extension for the Arduino IDE was required to properly program the Teensy as mentioned in Section 3.2.1.1.

3.5.2 Path Planning

The path planning was based off of the A* Search Algorithm. By storing a 6 by 6 2D array in memory representing the course, as well as the robot's current 'X, Y' coordinates, the algorithm generated a list of steps (e.g. Move Forward Tile, Rotate To 90) to a given coordinate of interest. The algorithm generated four moves to make next and put them into a priority queue. These four moves were to move forward or rotate to a different directional angle (eg. if the robot was facing 0, it would add movements for rotating to 90, 180, and 270). The move forward action would only be added to the queue if its location was not a detected hazard or outside the grid. The priority queue was sorted based on Manhattan Distance to give a heuristic as to which next movement was the most optimal.

3.5.3 Scanning and Object Detection

Due to the uncertainty of the course, the most efficient idea was to use the long range LIDAR to do a 180 degree scan of the course from the starting position and relate the data to determine the exact location of the three objects of interest (the candle and two houses). An additional program was created to generate the exact angles to target a certain tile. Tiles and their associated angle of rotation relative to the start location can be found in Figure 13. The goal was to have the robot rotate to each angle and perform a filtered scan with the laser sensor and take a flame reading.

```
VISUAL
 329  338  348  00  11  21
 323  333  345  00  14  26
 315  326  341  00  18  33
 303  315  333  00  26  45
 288  296  315  00  45  63
 000  000  000  00  00  00
```

Figure 13: Grid of Specific Target Angles from Starting Location

It was quickly realized that this approach was inaccurate. When this was discovered, wear and tear from the course had caused components from the drivetrain to loosen, meaning that turns could only be accurate to ± 3 degrees of the desired angle. It was anticipated that this issue would worsen with time. Another method would need to be devised.

In the second iteration, a rotating scan mode was implemented. During the rotation, the IMU and LIDAR data were constantly read to match an angle with a LIDAR distance. A limitation of the motors was a minimum speed that the robot needed to be rotating at without risking the motors stalling. The number of samples taken by the laser sensor for each reading needed to be halved from its original calibrated value to be able to take readings fast enough. After this adjustment, this provided enough number of distance readings per angle to obtain a reasonable scan of the map. After the scan, the entire dataset of angle and distance pairs were turned into expected horizontal and vertical tile offsets based on the robot's current position. These were then compared against the expected wall distance to determine the location of objects.

The flame detection in this iteration of the initial scan was very good, but when trying to obtain object positions from this initial scan, a few major issues were encountered. First, the robot rocking when turning made it very difficult to get consistent distances from the laser sensor. The solution to this was to adjust the weight bias of the robot so that it was slightly forward, meaning that the front four wheels would always be on the ground when the robot was stopped. Care was taken to ensure that the weight bias was just enough that the front wheels were touching the ground, but not a source of significant drag when turning the robot. After these modifications, the initial scan was able to determine the location of all objects on the grid, but with a few false positives. Before these could be investigated and fixed, the laser sensor broke.

After the failure of the LIDAR, the IR sensor was used instead. Due to its lower effective range, the robot would need to move towards the center of the course first and do a 360 scan instead. As discussed in Section 3.4.4, it was found that the IR had issues with detecting objects during the initial scan due to the light from the candle flame. While the ultrasonic sensor could take over in most cases, the beam spread and slow responsiveness made it an unacceptable choice for use in the initial scan.

It was ultimately decided that an extra row based scan was required to detect objects and that the rotation scan would be kept due to the significant amount of work put into this method and also its ability to consistently find and put out the flame. The row based scan assumed that there exists some sort of path with no hazards or obstacles from the start position to the other end of the path.

After extensive tuning, the row scan ended up working as desired, and was used on demo day.

4.0 Test Plan

There are two main pieces which needed to be operational in order to successfully accomplish any of the tasks. These would be to ensure that the device's movement is proper and that the device could reach a task destination on the course. The next set of tests were for the completion of the tasks themselves. These included being able to detect magnets, detect and extinguish the flame, distinguish the two houses, and to locate objects on the course. The final integration test was done after all the individual pieces were deemed to be in a working state.

4.1 Turning and Straight Line Movement Testing

The purpose of this test was to determine whether or not the movements of the robot have been tuned well enough to not deviate significantly during a full run through. This test involved setting a heading for the robot, having it turn to that heading, and then move forward indefinitely. The reference "straight line" was a line in the floor tiling that the robot had to follow for two meters. Depending on the deviation from the expected path, the motor tuning parameters were changed.

4.2 Distance Testing

The purpose of this test was to determine if the robot could travel exactly one tile at a time. This test involved commanding the robot to move forward and backward exactly one tile, starting and/or ending ending on any combination of wood, sand, and gravel. Depending on the distance traveled, encoder counts and expected sensor distances were changed.

4.3 Path Planning Test

This test was to ensure the proper function of the path planning algorithm and that the generated optimal path was operational.

The confirmation of this test only needed to be theoretical as the return of this function is a set of movement instructions. As a way to parallelize the development process, a simulation was built and run on the separate Teensy. The simulation was given multiple theoretical grids with hazard tiles and two coordinate locations and returns the path and the corresponding movement instruction. All instructions and the position of the robot on the grid were printed for an easier visual check.

4.4 Magnet Detection Test

The purpose of this test was to determine if the device was able to detect a magnet while on/partially on the tile with a magnet on it. This test involved driving the robot over a sand tile, and printing out the magnetometer values. These magnetometer values could then be used to tune the magnet detection threshold.

4.5 Flame Detection and Fan Test

The purpose of this test was to determine whether or not the heading of the flame can be detected and if the fan can blow out the candle flame.

Due to the offset location of the fan mount and the fan nozzle, the angle range which the air would be affected by the fan was quite narrow. Hence the rotation angle to target the candle needed to be fairly accurate. The effective distance of candle also needed to be determined (ie. 1 tile away vs. the candle at the furthest corner).

The nozzle was very helpful here in directing the air flow towards the candle, while still allowing enough margin to account for inaccuracy in turning. Details about nozzle design can be found in Section 3.1.2.3.

4.6 House Differentiation Test

This test was very tightly coupled with the calibration of the colour sensors. The objective of the test was to determine if the two houses could be differentiated clearly and not confused with a candle or just a blank tile and the effective range to do this. This test involved mounting the color sensor, and moving the robot to various locations around the board, pointing towards various objects. Color Sensor readings would then be recorded, and thresholds would be tuned accordingly.

4.7 Object Detection Test

The purpose of this test was to determine whether the robot could locate objects on the board. This test involved running the initial scan and later the row scan, detailed in Section 3.5.3, and printing out values throughout the process of calculating distances, breaking those distances down into components, and turning those components into tile offsets. Calculation parameters could then be tuned accordingly.

4.8 Integration and Full Run Test

After combining all the required components physically and fitting the component functions into a working main, full tests were planned on the course to make sure all components worked together properly. This test was broken down into three component tests, and one full run test. Due to the number of people that needed to use the course, most of the integration testing was done on a makeshift course created by another group.

The first component test was to determine if the robot could put out the candle. It involved the robot driving to the center of the course, scanning for the fire, and putting it out.

The second component test was to determine if the robot could locate the magnet. It involved the robot starting at a predefined position, moving to each sand pit and attempting to detect the magnet.

The third component test was to determine if the robot could scan the course, locate the houses, and drive to them. It involved the robot moving across the board from its start position, scanning each row as it went. After it got to the other side of the board, the robot would path plan, and drive to the two houses, detecting their color on arrival. It should be noted that this test was never successfully completed, as the color sensor broke shortly before the start of the competition.

The full run test was to determine if the robot could complete all objectives. It involved the robot putting out the fire, locating the magnet, and travelling to the houses. This test was never completed before the presentation.

5.0 Game Day

5.1 Competition Results

The robot required three runs to complete its expected tasks.

During the first run, the robot located and extinguished the fire, but got stuck when trying to exit the first sand pit. In the second run, the robot extinguished the flame, and got through the first two sand pits. It got stuck before reaching the third sand pit. In the third run, the robot found the magnet in the third sand pit, and then travelled to both of the houses. Finally, the robot returned to its starting position to conclude the run. In all three runs, it ran over the second house instead of stopping in front of it.

5.2 Expected vs Actual Performance

The expected execution order of the robot was as follows:

1. Move to the center of the course and do two rotation scans
2. Rotate to the flame
3. Turn on the fan until the flame is extinguished
4. Return to start position
5. Do a full row by row scan
6. Traverse to each sand tile until magnet detected
7. Traverse to each house (delivers food in the process)
8. Return to start position

Steps 1 - 4 executed very well for the runs without human interference. In Step 5, the row scan performed as expected except for at the last row. The red house was placed in a position that was expected to be clear of obstacles. As a result, while the house was detected, the robot ran over the obstacle to perform another row scan instead of stopping in front of it. From the printed output, the objects were detected. In Step 6, the optimal paths to each of the sand tiles were confirmed to be correct with the output. This step encountered two different issues on separate runs. In Step 7, moving towards one of the houses was correct. For the other house, similar to the issue with Step 5, due to the edge case, the robot would drive onto the tile with the house and then signal its location. There was no problems returning to the starting position after the tasks were complete.

Throughout the whole run, the general forward tile movement and rotations were done better than expected. The correction steps done at each tile with the IMU orientation and IR distance data performed well. The major issue requiring human intervention was when the robot moved onto and reversed from the sand tiles.

6.0 Lessons Learned

Over the course of the project, multiple lessons were learned that could be of use during future projects, including the upcoming fourth year design project.

The first lesson learned was to take advantage of discounts and avoid shipping costs via bulk purchases. Many parts purchased by the group were in relatively small quantities with shipping costs that were at times more expensive than the part purchased. If these orders were combined with other groups, shipping costs could be split, and it might even be possible to take advantage of bulk discounts (ex. discounts on orders for connecters).

Another lesson learned was with regard to I2C and the specific and niche knowledge regarding the buses. I2C lines require pullup resistors and short wires to avoid any capacitance errors. Because the team had no prior knowledge regarding the former, both the IMU and LiDAR initially did not work. Finding out what the issue was a time consuming endeavor but eventually, it was found out that resistors were required in order for data readings from the sensors to come through. The main lesson, however, is that communication protocols may have unexpected quirks that a team should research before testing sensors. In general, research should be done for anything a team has limited knowledge on, to reduce delays during production and testing.

A core lesson from the software side was to really understand the problem before making changes. As the deadline approached, brute force, quick fix methods were used to remedy problems without fully reviewing the related components. This eventually led code which was hard to understand and debug. By carefully figuring out the source of the issue and applying a thought out solution, the chances of future problems are reduced and the pain of creating solutions is lowered.

A core mechanical lesson was to put more effort into understanding the CAD rather than just designing. Even after 120 hours of design work, the robot still failed to climb pits when first laid out to run. In those 120 hours, the mechanical team had not caught either of the two largest failures preventing the robot from running effectively. Had a more comprehensive check been done before manufacturing, or more liberal examination of the robot design occurred, much of the rework that dominated the second half of the build season would not have needed to happen.

The most important lesson, however, is setting hard deadlines for tasks. A common event that occured within the team was spending vast amounts of time either trying to fix a component or deciding a strategy (amongst many) to pick from. Because of this, time that could've been used in implementing a new feature or testing was wasted unnecessarily. An example of this was our LiDAR debugging, where the team spent approximately 6 hours trying to fix a broken sensor (researching, debugging, etc).

7.0 Conclusions and Recommendations

The startup ‘Tree-80’ had a goal to create an autonomous device to perform search and rescue missions. Criteria, constraints, and objectives were identified. Possible solutions were brainstormed and the optimal solution was chosen, designed, and constructed. The design was iterated on multiple times to accommodate changes in requirements, overcome challenges, and deal with part failures. Eventually, the robot was able to complete most of the missions set out for it with some human assistance. Overall, the design was partially successful in its ability to perform the search and rescue tasks. The robot performed to the group’s expectations under the constrained and time sensitive conditions. With more time to address the sensor failures and to fine tune the device, it is believed that the robot would be a viable solution to the search and rescue problem.

The design language of the robot should be made different if this challenge is to be completed more successfully. A chassis with larger wheels and grippier tires would be extremely beneficial if pit crawling is desired. A holonomic drivetrain, such as one with omni or mecanum wheels would be suitable for a robot designed to go around all obstacles. Modularity, however, is extremely important regardless of design, and flexibility in how the robot can be equipped to move and complete challenges should be prioritized.

It is recommended that most, if not all, electrical wire ends are crimped with 12-18 gauge wire instead of standard header male-male connectors for more secure connections. Additionally, all wire pairs or groupings be joined or labelled either through heat-shrink or tape, to allow for a smoother debug and plug experience. Doing both of the above also reduces the chance of burning up or shorting circuits, resulting in less potential costs in the long run. One final recommendation is to color-match power, ground and signals.

In the future, it is recommended that the sensor configuration be switched back to using some sort of laser based range finder, especially given the difficulties encountered with ultrasonic sensor update frequency, and the IR sensor being unable to determine distance when pointed at a source of infrared light. Additionally, it is recommended that the PI controller implementation be completed for better control over robot drift.

In retrospect the complexity of this project did not necessarily invite a complex or free thinking challenge; instead, it particularly favoured modularity and ease of iteration. Although clear now at the time of writing, there was no strong indicators this would be a concern early on in the design phase. It is recommended that when faced with tight budgets, timelines, and non concrete targets that modularity be considered the most important design constraint.

8.0 References

[1] "When hikers need help, who foots the rescue bill?", *MNN - Mother Nature Network*, 2019. [Online]. Available: https://www.mnn.com/earth-matters/wilderness-resources/stories/when-hikers-need-help-who-foots-rescue-bill.

[2] R. Yan. MTE 380 2019, 2019. [Online]. Available: https://www.flickr.com/photos/127485946@N06/33570036878/in/album-72157679494977898/

[3] ”Distance Measuring Sensor Unit Measuring distance: 20 to 150 cm Analog output type ”, *Sharp*, GP2Y0A02YK0F Datasheet, Dec 1, 2016, Available: https://www.robotshop.com/media/files/pdf/datasheet-gp2y0a02yk0f.pdf

[4] “Smart Sensors and Applications”, *Parallax Inc*., Chapter 2, Available: https://www.parallax.com/sites/default/files/downloads/28029-Smart-Sensors-Text-v1.0.pdf

[5] "Arduino Color Sensing Tutorial - TCS230 TCS3200 Color Sensor", *HowToMechatronics*, 2016. [Online]. Available: https://howtomechatronics.com/tutorials/arduino/arduino-color-sensing-tutorial-tcs230-tcs3200-color-sensor

[6] "Pololu - 75:1 Metal Gearmotor 25Dx66L mm MP 12V with 48 CPR Encoder (No End Cap)", *Pololu*, 2019. [Online]. Available: https://www.pololu.com/product/3242.

9.0 Appendix

Name	Cost	Vendor	Date	Notes
Jessen	238.14	Pololu	29 Jan	*Mechanical Parts & MISC*
Kelvin	26.99	Amazon (unknown)	29 Jan	*Screw Terminals*
Rahul	121.95	Robot Shop	28 Jan	*Sensors & MISC*
Kelvin	40.5	Robot Shop & Digikey	31 Jan	*Teensy*
Kelvin	27.27	Amazon (Several)	31 Jan	*Level Shifters & Fuses*
Jessen	116.09	VexPro	3 Feb	*Bearings*
Jessen	105	Andymark/Studica	3 Feb	*Motors & MISC*
Kelvin	32	Amazon	20 Feb	*Protoboards*
Kelvin	38	RobotShop	27 Feb	*Dual Channel Motor Driver*
Kelvin	21	Amazon	4 Mar	*Xt60 Connectors - For motors*
Kelvin	50.21	Robotshop	6 Mar	*IR Range Finder*
Kelvin	12	Robotshop	11 Mar	*Sensor Cable*
Kelvin	112	RobotShop	13 Mar	*Replace Broken Teensy. Bought Backup*
Kelvin	15	Sayal	15 Mar	*MOSFET*
Michael	180	RobotShop	28 Mar	*Broken LiDAR - Give to Mess*
TOTAL	1136.15			

Figure A1: Team BOM

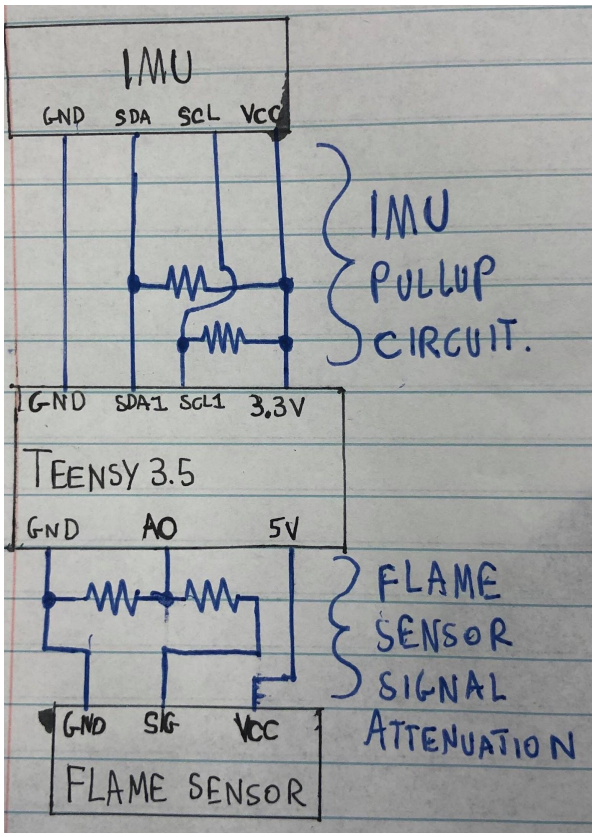


Figure A2: I2C Pull-up & Level Shifting Circuits

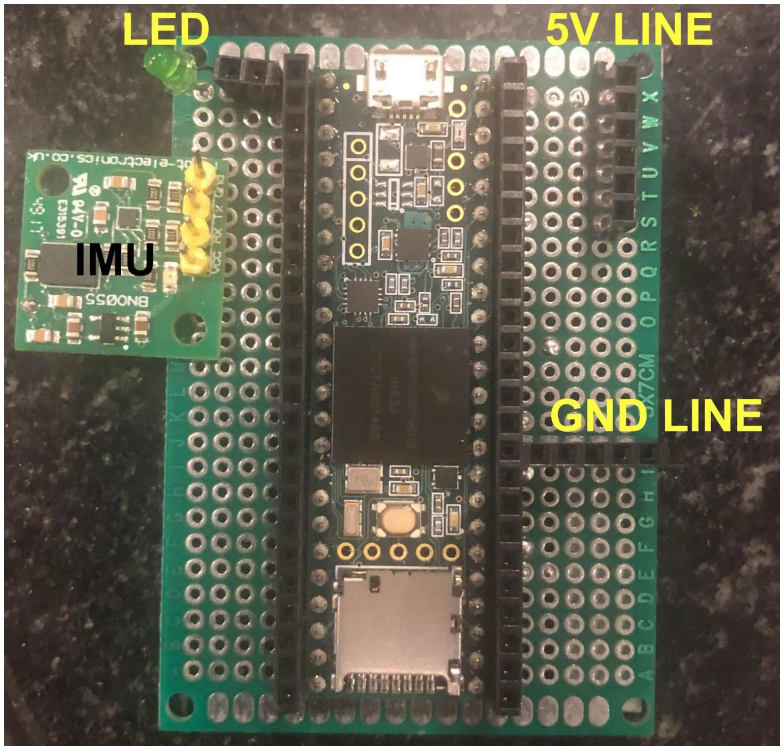


Figure A3: Teensy Hub Top View

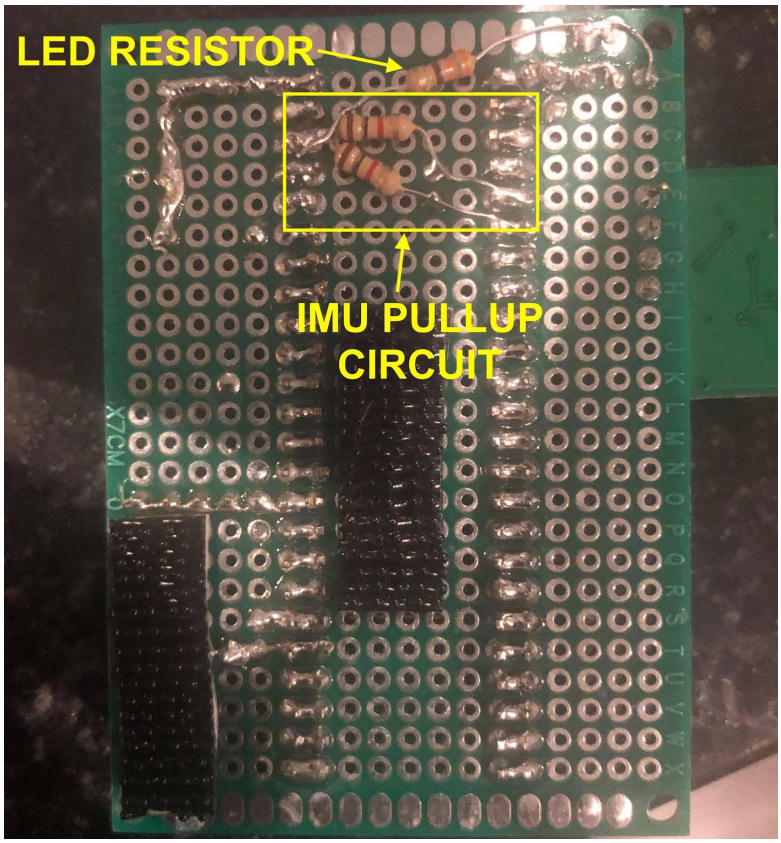


Figure A4: Teensy Hub Bottom View