April 6, 2025 Team #5

Dr. Behrad Khamesee
Professor, Mechatronics Engineering
Department of Mechanical and Mechatronics Engineering
University of Waterloo
200 University Ave W.
Waterloo, Ontario, N2L 3G1

Dear Professor Khamesee,

We are pleased to submit the attached report titled “Autonomous Search and Rescue Robot” as the final requirement for the MTE 380 course. This report outlines the design and development process of our autonomous search and rescue robot, which is capable of navigating a course using computer vision, retrieving a LEGO figure, and returning it to a designated safe zone. We are a team of 5 Mechatronics engineering students with a diverse background in mechanical, electrical and software engineering. Our varied experiences allowed for strong collaboration throughout the project. Over the past three months, our team has worked diligently to meet the project requirements as outlined in the MTE380 course. The design process included both technical challenges and rewarding learning experiences, and we are proud of the outcome.

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.

We would like to thank the MTE 380 teaching team, including you, Professor Khamesee, Jeffrey Lee, and Yufeng Lee for the guidance and support throughout the term. Additionally, we would like to thank Paul Groh and Eugene Li for their support on this project and for the space and resources in the MME workspace.

We have thoroughly enjoyed the learning process throughout this project and are excited to present our results.

Kind regards,

Lili Strong, Chelsea Dmytryk, Nathan Rowe, Nicholas Drazso, Natalie Tsang, Nathan Rowe,
3B Mechatronics Engineering
University of Waterloo

Department of Mechanical and Mechatronics Engineering

MTE 380 Final Project Report: Autonomous Search and Rescue Robot

A Report Prepared For:
Professor Behrad Khamesee
The University of Waterloo

Prepared By:
Natalie Tsang
Chelsea Dmytryk
Lili Strong
Nathan Rowe
Nicholas Drazso

April 6, 2025

Executive Summary

The objective of this project was to design and build an autonomous search and rescue robot capable of following a line using computer vision to rescue a LEGO figure and transport it to one of the designated safety zones.

The development process involved several key steps, beginning with the design of the robot's locomotion system, followed by the integration of the Pixy vision module, and ultimately the full system integration. Over the past three months, our team worked on the design, manufacturing, and integration of the robot, adhering to the timeline provided by the MTE380 course. We evaluated various design iterations based on the constraints provided by the teaching team and the objectives defined by our team to be successful on the final competition day. The final design evolved through multiple iterations to create a fully operational robot. The finished design included a custom PCB chassis, a single-arm gripper, a Pixy vision module for computer vision, and a Teensy microcontroller for robot control.

Throughout the development process, we encountered several challenges. These included errors in the initial PCB design, which required fly-wiring to make the robot operational, as well as difficulties with the gripper's ability to hold the LEGO figure, necessitating several iterations. Additionally, the software required extensive tuning, particularly due to environmental factors such as varying lighting conditions, which affected the Pixy module's ability to reliably detect different parts of the course. However, the team's collaboration, leveraging each member's expertise, enabled us to successfully troubleshoot and iterate on the final design.

The final cost of the robot came to $278.15. The main costs included the Pixy module, motors, and the custom PCB. Most of the electronic components were bought off-the-shelf or manufactured externally, while many of the custom mechanical parts were 3D printed by team members, and the PCB was assembled in-house.

The robot was tested on Game Day, March 28th, 2025. Game Day was a great success: the final robot weight came to 131g, with the fastest run time of 13 seconds. The robot completed 3 out of 3 successful runs, achieving a reliability factor of 0.75, and meeting all performance objectives of the competition.

Table of Contents

List of Figures

List of Tables

1 Introduction and Background

1.1 MTE 380 Course Description

The MTE 380 course is a project-based course designed for third year Mechatronics engineering students at the University of Waterloo. Over the term, students are exposed to the design process of an open-ended problem requiring a mechatronics-based solution. This design process is then applied and documented as outlined in the course. This includes applying the design process from a needs analysis, problem formulation, preliminary designs, creating detailed designs, and then fully implementing this design, and then communicating the design through reports and presentations. Additionally, it develops skills such as working on a multi-discipline project and developing leadership and membership in team efforts.

1.2 Autonomous Line Following Robot Challenge

For the Winter 2025 term, the given problem was to design and build a device to execute a search and rescue mission to save a LEGO figure. The course is designed with sand, gravel, and empty traps if the robot goes off the course. The project cumulates to the final competition day where each group is given three attempts to complete the course. A successful completion run is where the device can traverse the course, pick up the LEGO figure, deliver it to a safe-zone, and then return to the start tile. There is a performance index used to determine bonus marks during the competition, used to calculate the device's performance. It is a function of the device's mass, reliability, and the run time as given below.

$$PI = \frac{F_f}{Rmt^2} \tag{1}$$

Where F_f is a fudge factor used by the teaching team to make the result look nice, m is mass in kg, t is time and is a measure of the fastest time it takes to complete the course in seconds, and R is the reliability factor which is based on the number of successful attempts out of 3. If all attempts are successful, the reliability is 0.75, then 0.9 for 2 successful attempts, and 1.0 for 1 successful attempt.

2 Needs Analysis and Problem Formulation

2.1 Needs Analysis and Problem Statement

The rising number of search and rescue incidents in recent years has created an urgent need for more trained responders. In addition to increasing the number of search and rescue volunteers, there is also a critical need for faster response times. Delays in emergency situations can mean the difference between life and death, making it essential to enhance both personnel capacity and response efficiency.

2.1.1 Needs Analysis

By designing and building an autonomous search-and-rescue device for navigating an obstacle course to rescue a LEGO figure, this project aims to contribute to the development of reliable, low-cost autonomous systems for search and rescue operations.

2.1.2 Problem Definition

Design an autonomous mobile robot capable of performing a search and rescue operation. The robot must navigate a designated obstacle course to locate and retrieve a LEGO figure from a danger zone, transport it to a safe zone, and return to the starting point without human intervention.

2.2 Objectives, Constraints and Criteria

In this section the various objectives, constraints and criteria developed both by the teaching team and group will be discussed.

2.2.1 Objectives

The objectives for the final robot design are outlined below in Table 1.

Table 1: Objectives of Final Robot Design

Objective	Success Criteria
Autonomous Navigation	The robot must effectively follow the red tape line to and from the bullseye while minimizing oscillations, overshoot, and steady-state error.
Use computer vision	The robot must accurately identify the bullseye, LEGO figure and the green box for successful completion.
Lightweight and liftable	The robot needs to be under 200 grams.
Minimize Time to Rescue	The robot should accelerate at 5 meters per second squared and reach a velocity greater than 1 meter per second. Complete the full course in under 30 seconds.

2.2.2 Constraints

The project has various constraints restricting the design of the robot as outlined by the teaching team. These constraints are listed below.

1. The device is not permitted to leave the boundaries of the search and rescue site.
2. The device is not permitted to climb over the walls.
3. The device must adhere to the designated path; no shortcuts allowed.
4. The maximum size of the robot is 10”x10”x10”.
5. The total cost of the robot must be under $300.
6. Flying devices, projectiles, LiDAR, and devices that separate into multiple pieces are not permitted.
7. The course cannot be destroyed.

2.2.3 Criteria

The criteria our group focused on when developing possible designs and selecting the final design are outlined in Table 2.

Table 2: Criterion for Selection of Final Robot Design

Criteria	Description
Integration	Having certain features on the robot can have negative effects on other components or systems. Designs minimizing this potential negative impact were prioritized.

Weight	The weight of the robot is critical for the performance index. It is a value that is fixed prior to game day but has a big range. Therefore, minimizing weight is extremely important.
Cost	The actual cost of the robot was important to minimize since marks would be deducted on robot designs that exceeded the $300 budget.
Complexity	The complexity of the design impacts the overall time commitment of the team. Minimizing the level of difficulty to implement the design while ensuring a well-designed robot was considered for this criterion.
Reliability	The reliability of the robot is extremely important. It is a criterion that is critical for the performance index, as the course needs to be completed at least

2.3 Overview of Final Design

This section will outline the components of our final robot design including the mechanical components, electrical system, and software architecture. Figure 1 shows the final completed robot design with numbers indicating important components. These components are listed in Table 3.

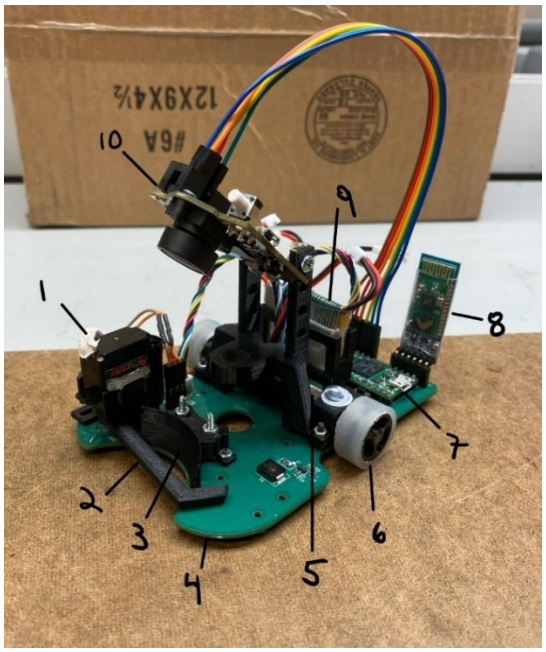

Figure 1: Full robot with numbered components

Table 3: Component Legend

Number	Component
1	Servo Motor
2	Arm
3	Backstop + Roller Ball
4	PCB
5	DC Motor
6	Wheel
7	Teensy
8	Bluetooth Module
9	Battery
10	Pixy Camera

2.3.1 Mechanical System Design

The mechanical system was comprised of four main components. The motors, gripper, wheels, and the suction system.

2.3.1.1 Motors

The motors are two 6V Pololu Micro metal gear motors with a 10:1 gear ratio. These motors had a maximum speed of 3100rpm which translates to 324.6 radians per second. This gives us a maximum possible velocity of 7 meters per second. They are also capable of providing 0.02157463 Newton meters of torque, allowing us to achieve an acceleration far greater than our intended 5 meters per second squared specification.

2.3.1.2 Gripping Mechanism

The gripping mechanism was designed to scoop the minifigure into a cut out space in our chassis. The assembly was made of a servo motor for actuation, a 3D printed arm, and a backstop to provide a locating face. The servo was a DSM44 servo with a max range of 145 degrees. The arm was a curved piece of PLA with foam wrapped around the center to increase the area that gripped the figure, as well as cushioning for vibrations. Finally, the backstop was also 3D printed. The arm was positioned to contact the LEGO figure on the upper chest area, since the chest was the largest flat surface and high enough to effectively use the backstop. The arm also was required to be above the legs of the minifigure while in the seated position so the arm could act like a seatbelt and reduce the chance of the minifigure being released. The backstop was designed to lift the minifigure off the ground slightly to avoid getting caught on large bumps on the track. This was accomplished by tilting the backstop so when the figure came in contact with it, the arm would push the top of the figure up the slope.

2.3.1.3 Fan Design

To improve the robot's performance, particularly its cornering speed, a suction system was incorporated into the design. Based on early calculations, it was clear that slipping would become a limiting factor for the robot's speed and ultimately for its performance index.

This limitation arises from the fact that the maximum cornering speed is governed by the balance of frictional force and the required centripetal force for turning. Starting from Newton's Second Law applied to circular motion, the frictional force, F_f must equal the centripetal force F_C leading to equation 2:

$$F_C = F_f = \mu F_N = ma = m\left(\frac{v^2}{r}\right) \tag{2}$$

Without additional forces, the normal force, F_N, is just mass times gravity, mg, where m is the mass of the robot and g is the gravitational acceleration. Here, v is the velocity of the robot and r is the radius of the corners on the course. Then, solving for maximum velocity results in equation 3.

$$v_{max} = \sqrt{\mu r g} \tag{3}$$

Where μ is the tire to ground coefficient of friction and r is the turn radius. Since μ, r, and g are fixed properties, the maximum cornering speed is capped at:

$$v_{max} = \sqrt{(1.2)(0.15[m])\left(9.81\left[\frac{m}{s^2}\right]\right)}$$

$$v_{max} = 1.32 \left[\frac{m}{s}\right]$$

Based on initial calculations of driving speed requirements and motor capabilities, it was clear that the robot was capable of much more than 1.32 m/s meaning this velocity was a clear limitation to the overall performance of the robot.

However, by introducing an additional downward force through suction, the normal force becomes, as described by equation 4:

$$F_N = mg + F_d \tag{4}$$

Where F_d is the downforce created by the suction system. This increases the effective frictional force and raises the maximum velocity according to equation 5:

$$v_{max} = \sqrt{\frac{\mu r(mg + F_d)}{m}} \tag{5}$$

Therefore, by adding suction, the robot can corner at significantly higher speeds without increasing its mass.

Research into similar small-scale robotics competitions, such as Micromouse, revealed that suction systems are commonly used to overcome this challenge with some robots achieving downforce as high as 5x their mass allowing for cornering velocities of up to 4 m/s. The suction system creates downforce, allowing the robot to achieve higher cornering speeds without the drawback of increasing mass, which is advantageous considering the performance index favors reduced mass. The design consisted of two main subcomponents: an impeller and a skirt. The impeller is responsible for creating a pressure differential across the chassis while the skirt is responsible for sealing against the ground to generate the required suction effect needed for increased traction.

Given the tight project timeline, the complexity of accurately simulating airflow, and uncertainties about other component selection and design, the suction system design was driven primarily by hands on experimentation rather than computational methods such as CFD. Experimentation was prioritized over calculation to save time, rapidly iterate on designs, and adapt quickly based on observed results. This approach allowed the suction system to be developed within the constraints of the project timeline.

A powerful 8520 coreless drone motor was selected for preliminary testing for driving the impeller since it is commonly used in the hobby drone sector.

2.3.1.4 Wheel Design

To meet the design requirements, custom tires were cast in silicone and paired with 3D printed PLA rims. Custom making the tires allowed for customizability of material and in turn the coefficient of friction, and the size. Custom tires also reduced costs compared to off the shelf options since it was assumed iterations would be required.

The rims were designed with minimal spokes to reduce weight while maintaining strength. During manufacturing, it was found that the silicone did not naturally bond to the PLA, so super glue was used to attach the tires to the rims.

The rim was chosen to be 16mm in diameter to account for a 2mm thick silicone tire which was enough to be durable yet lightweight.

2.3.2 Electrical System Design

The electrical system had two primary focuses: power distribution and enabling communication between the various components. An overview of power distribution and communication as they relate to critical components are shown in Figure 2.

.

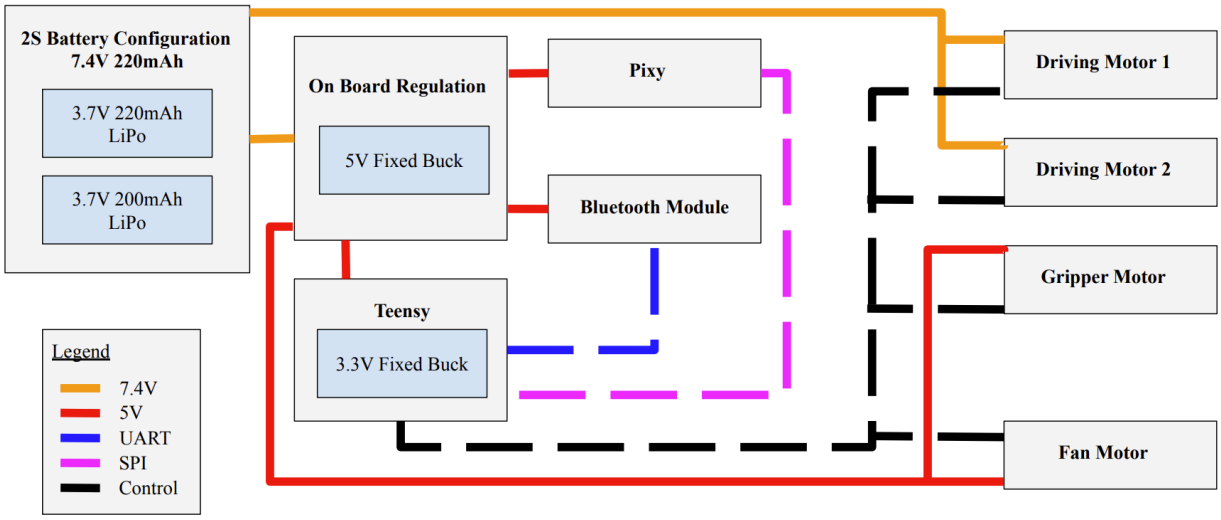

Figure 2: High Level Electrical System Power Distribution and System Communication

2.3.2.1 Power Distribution

The robot is powered by two 3.7V, 220mAh LiPo batteries in a 2S configuration. This allows the robot to receive an equivalent 7.4V, 220mAh power source. The battery voltage is used to power the driving motors and provide an input to the fixed 5V buck. This buck then provides a regulated 5V output to power critical components such as the Pixy, Teensy, Bluetooth module, motor drivers, gripper motor, and fan motor. For this reason, the selected buck can provide up to 5A on the 5V rail. The Teensy additionally has an onboard 3.3V voltage regulation system that is used to power the button circuit. This was done since the pins on the Teensy cannot have a voltage above 3.3V applied, so the 3.3V ensures no damage to the Teensy.

2.3.2.2 System Communication

The Teensy is the main control board of the system. It communicates with the Pixy through SPI and receives feedback that allows it to drive various motors on the system based on the robot's surroundings. The Bluetooth module communicates with the Teensy through UART, allowing the serial output of the Teensy to be viewed without physically connecting to the Teensy. The driving motors are driven by motor drivers mounted on the board.

2.3.3 Software System Architecture

2.3.3.1 Teensy

The Teensy microcontroller was selected for its ability to run the main control and state arbitration loop. Given its small form factor, reasonable price, and adequate computation ability, the Teensy was a superior choice over other microcontrollers considered such as the Arduino Nano. The main software logic running on the Teensy included the PD controller, motor, and encoder output, as well as a state arbitrator which was used to switch the state of the robot's software program. Table 4 provides a brief description of each state within the main control loop's switch-case logic running on the Teensy.

Table 4: Description of all states and functionality in the main control loop.

State	Description
LINE_FOLLOW_PICKUP	- Follow the red line. - Stop if the bullseye has been detected within a specific threshold.
LINE_FOLLOW_DROPOFF	- Follow the red line. - Stop when the green box has been detected within a specific threshold.
LEGOMAN_ALIGN	- Identify the LEGO figure in the frame. - Center the robot so that the LEGO figure is in ideal location for gripping based on coordinates in the frame. - If the LEGO figure is not detected, pulse the robot forward until the LEGO figure enters the frame.
PICKUP_LEGOMAN	- Close the gripper around the LEGO man. - Turn 180 degrees with the LEGO man. - Look for the red line to follow. - If the red line is not in view, turn a small amount until the line comes in view.
UNLOAD_LEGOMAN	- Turn to face the green box straight-on. - Move forward and open the gripper to release the LEGO man. - Reverse the robot and turn back to re-find the red line.
GO_HOME	- Follow the red line. - Stop when there is no longer a red line detected.

2.3.3.2 Pixy

The Pixy camera module was chosen for its built-in line tracking and object detection capabilities, along with its user-friendly and well-documented API, which made integration into the system straightforward. By combining a camera and a system-on-chip (SoC) in a single package, the Pixy significantly simplified the alternative approach of pairing a standalone camera, such as a web camera, with a microcomputer like a Raspberry Pi. Its largely open-source API also provided extensive documentation, making it easy to interface the camera's computer vision features with control algorithms on a microcontroller.

The Pixy camera's color detection algorithms were used to detect the red line, as well as all other objects of importance, such as the bullseye, LEGO figure and the green box. Since the Pixy API provides crucial data such as block size and the X and Y coordinates of detected objects within the frame, this output could be directly integrated into the proportional-derivative (PD) controller to dynamically adjust the robot's steering in real time.

The Pixy camera performs object detection using a color filtering algorithm that can be trained to recognize specific objects based on their color properties, known as ‘signatures’. Using the PixyMon software (provided with the Pixy camera), the camera could be programmed to detect up to seven different color signatures.

To configure a signature, the Pixy camera is first connected to a compatible external laptop running the PixyMon software. A color within the camera’s frame is then selected and assigned to a signature. The Pixy records the hue and saturation range of that color for future recognition.

As the Pixy captures images at a high frame rate (up to 60 frames per second) it checks each pixel to determine whether it matches any of the saved hue/saturation ranges. It then applies the Color Connected Components algorithm to identify groups of adjacent pixels that share the same color signature. These groups are treated as individual blocks. The algorithm filters out noise and false detections, ensuring it will only return well-defined blobs.

PixyMon also offers a variety of adjustable parameters, allowing users to fine-tune the inclusiveness of the CCC algorithm, set limits on how many blocks are returned, and compensate for different lighting conditions to improve detection reliability. For Game Day, the Pixy camera was specifically configured to return only one block per color signature. This setting was chosen to minimize the risk of the robot selecting an incorrect block, ensuring that only the most prominent for each signature was used during operation.

2.4 Mapping the Design to the Objectives and Constraints

A summary on how the constraints were met in the design are shown in Table 5.

Table 5: How Criteria was met in the Design

Constraint	Solution
Maximum size of robot is 10”x10”x10”	The final robot dimensions were 3.3”x4.5”x3.7”, well within the limit.
Total cost of robot must be under $300	The total cost of the robot came to $278.15
LiDAR cannot be used	The robot relied solely on input from the Pixy camera module.
The course cannot be destroyed	The robot was designed to navigate the course without causing any damage.
The device is not permitted to leave the boundaries of the search and rescue site	The robot was designed to complete the course within the boundaries.
The device is not permitted to climb over the walls	The robot was designed to not climb over the walls and solely move using wheels.
The device must adhere to the designated path; no shortcuts allowed	The robot was designed to follow the designated path without any shortcuts.
Flying devices, projectiles, LiDAR, and devices that separate into multiple pieces are not permitted.	No flying devices, projectiles, LiDAR, and devices that separate into multiple pieces were used.

The project objectives were also fully met, as summarized in Table 6.

Table 6: How Objectives were met in the Design

Objective	Criteria Met

Autonomous Navigation	The robot demonstrated fully autonomous behavior, including line following, Lego figure identification, gripping, and precise drop off.
Use computer vision	Computer vision algorithms powered the robot's autonomous capabilities, enabling accurate detection and decision making.
Lightweight and liftable	Weighing only 131 grams, the robot was highly portable and easy to handle.
Minimize Time to Rescue	The robot's motors were selected for rapid response. Its speed was limited only by its ability to track the line accurately, making it especially fast on straight sections.

2.5 Project Schedule

The Gantt chart used to track each team's progress throughout the term is show in Figure 3. The projected timelines are shown in purple, with a red outline showing the actual timeline that was followed.

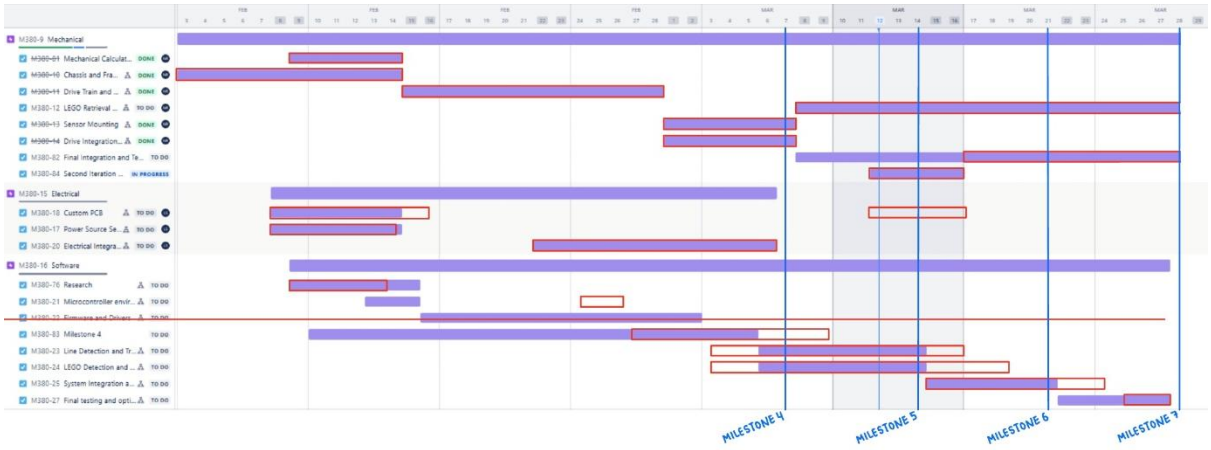

Figure 3: Gantt Chart with Team Progress and Milestones

Each milestone from the construction check onwards is identified on the Gantt chart as these milestones required substantial preparation.

2.6 Project Budget

Table 5 outlines the all components included in the assembly of the robot are included. It should be noted that although a Bluetooth module was integrated into the PCB during development for debugging purposes, it was removed before racing and therefore is not included in the budget of our final system.

Table 5: Project budget for the purchased bill of materials

Item	Vendor	Unit Cost ($)	Quantity	Total Cost ($)
Driving Motors	Pololu	37.12	2	74.24
Driving Motor Cables	Pololu	3.13	2	6.26
Gripping Motor	MESS	0	1	0
Pixy 2	DigiKey	105	1	105
Teensy	DigiKey	33	1	33
Battery	Amazon	4.30	2	8.60
PCB	JLCPCB	7.34	1	7.34

PCB Parts	DigiKey	28.96	1	28.96
Foam	Amazon	0.22	1	0.22
Roller Ball	Pololu	3.66	1	3.66
Tires	CUSTOM	0.75	2	1.50
Misc. 3D Prints	CUSTOM	1	3	3
			TOTAL	278.15

3 Construction and Testing

3.1 Final Built Model

Figure 4 shows the final Game Day assembly with the second iteration of the PCB and the Bluetooth module, which were improvements from Design Check 2.

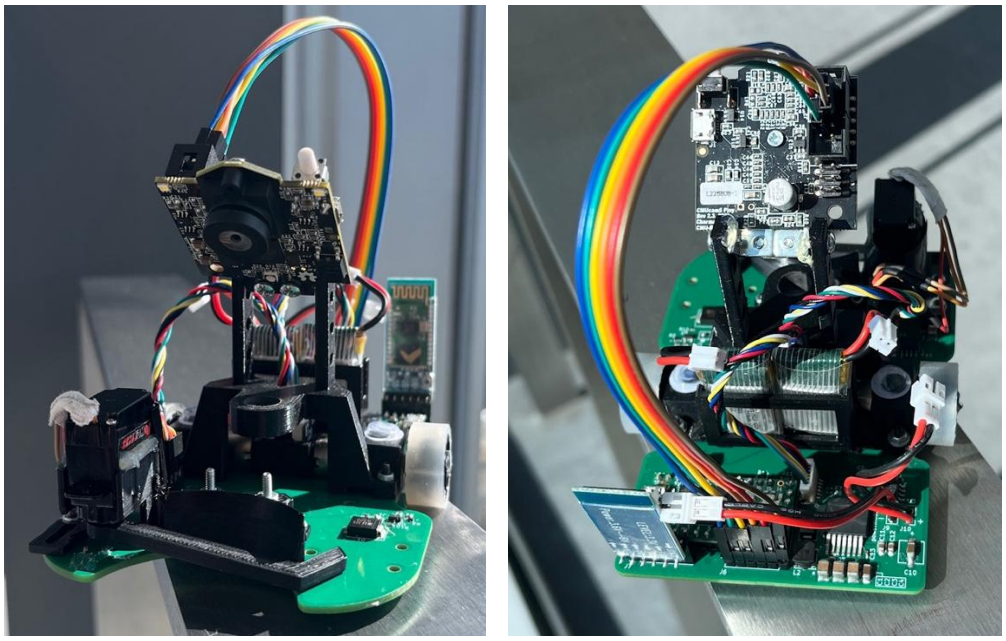

Figure 4: Final built model on game day.

To improve the organization of the wiring, the jumper cables connecting to the Pixy were used in their original ribbon configuration and the motor connector cables were twisted. Keeping the wires bundled together allowed for cleaner routing, minimized potential for disconnection or miswiring, and improved the overall tidiness of the system.

3.2 Design Modifications

3.2.1 Mechanical System Modifications

3.2.1.1 Arm Iterations

The robot gripper arm went through many iterations. The initial iteration featured a straight arm with a small hook on the end to prevent the minifigure from slipping out. This performed well when the LEGO figure was positioned but struggled to acquire the target in non-ideal cases. It was then decided to increase

the length of the arm to create a larger possible area. After testing again, it was seen that the design did not properly corral the LEGO figure into the slot, so the arm was curved to force the LEGO figure into the cutout. Finally, it was observed that the LEGO figure was occasionally gripped too loosely, so foam was added to the arm to fill any space and increase the gripping area.

3.2.1.2 Backstop Iterations

The backstop also experienced multiple iterations throughout the project. Initially the backstop was a small flat piece that did not protrude from the chassis at all. This did not effectively locate the LEGO figure. The backstop was then curved to match the shape of the chassis cutout. This provided a much larger locating face and increased the reliability of the gripping process. Finally, after observing the LEGO figure getting caught on the track, the backstop was tilted backwards to provide a small amount of LEGO figure off the ground.

3.2.1.3 Suction System – Impeller Iterations

Impeller testing was conducted to determine the optimal impeller design for generating downward suction against the course once it was realized that the robot was able to reach a high enough speed for the downforce to matter. Figure 5 illustrates the mock 3D printed robot chassis that was used to hold the different impellers while timing how long it took the chassis to slide down a wooden tile that matched the course surface. The impeller was powered off an external power supply with the voltage set to 5V and the current limited to 500mA. Although the motor could draw up to 1.7A, this was not realistic for the robot's power system, so the current was limited during testing.

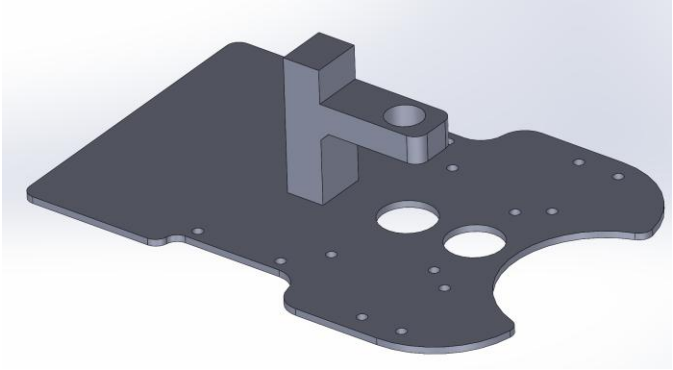

Figure 5: 3D printed mock chassis.

Figure 6 illustrates the test setup which involved a 1' piece of plywood inclined to 25 degrees.

Figure 6: Impeller test setup.

Testing varies impeller diameter, height, blade pitch, blade count, blade curvature angle, and impeller style, comparing compression design against full radial designs. Figure 7 illustrates approximately 20% of the impeller designs that were tested, the remaining ones were discarded or lost during the iterative design process.

Figure 7: 20% of tested impellers.

Table 6 summarizes the best performer for each style of impeller, showing that the radial design is approximately 2.6x more effective than the compression style.

Table 6: Impeller performance summary

Impeller Setup	Average time down ramp for 5 runs [s]
Control – motor off	0.844
5 blade, 60 degree curvature, 25mm pitch, 30mm diameter, 15mm height, compression	21.13
5 blade, 60 degree curvature, 30mm diameter, 15mm height, radial	55.12

Blade count was identified as the most significant factor influencing performance, with 4 or 5 blades yielding optimal results. Increasing the blade count from 5 to 7, or decreasing from 5 to 2, both resulted in approximately a 50% reduction in performance, approximately 21s to 9s for both cases meaning 7 blades performed equally as good as 2.

Figure 8 illustrates the best performing impeller was a full radial style with 5 blades, 60 degrees of convex curvature, 25mm blade pitch, 30mm diameter, and 15mm height, with a 13mm hole in the PCB.

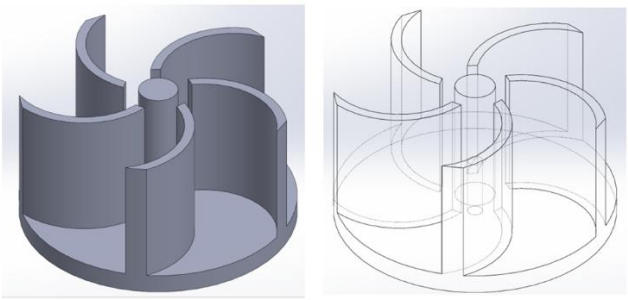

Figure 8: Optimal impeller design.

Further improvements could be made with higher quality manufacturing or CFD optimization, but instead physical testing was prioritized due to time constraints. All impellers were FDM printed using a Prusa MK4S with 0.2mm layer height. It must be noted that with a smoother impeller, results should improve.

It should also be noted that the final design was restricted to a 28mm maximum diameter tapering down to 22mm at the base, since the 2nd PCB revision was dimensioned for a compression style fan rather than a radial style. This modification was required, which affected performance negatively. Figure 9 illustrates the final impeller that was prepared for use.

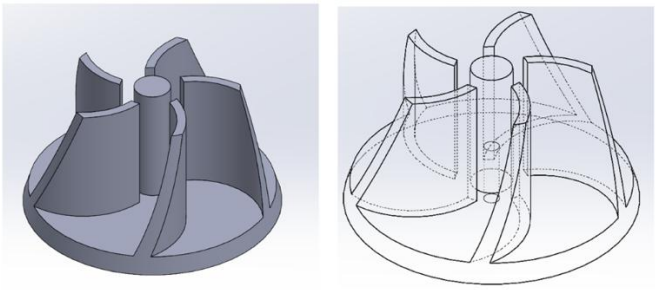

Figure 9: Adjusted optimal impeller design.

3.2.1.4 Suction System – Skirt Iterations

Skirt testing was conducted to determine which material, and design provided the best suction effect against the course surface using a similar technique as described for impeller testing. Dynamic skirts were created using tape, thin PLA 3D printed skirts, and a rail system. Foam and rigid 3D prints were also tested. Figure 10 illustrate all the skirt designs that were evaluated.

Figure 10: Example skirt designs.

The same test setup was used for the skirt as the impellers except 2.5mm spacers were added to simulate the distance off the ground that the wheels would provide, and the same 7 bladed impeller was used for all skirts. Table 7 summarizes the differences between skirt performances.

Table 7: Skirt performance summary

Skirt Description	Average time down ramp for 5 runs [s]
Control, impeller motor on but no skirt	0.98
Tape and thick foam	4.52
Felt	2.23
Felt, thin foam, thin plastic	12.8
3D printed rail system	51.3

Figure 11 illustrates the most effective skirt design. The design used the same tape concept as the skirt shown on the far right of Figure 10. The design combined 3D printed side rails with a thin 3D printed skirt that lightly contacted the ground. To prevent the skirt from catching on surface ridges, a ramp feature was added at the front of the thin skirt. This design allowed for movement in the driving direction while using tape to seal and the rail walls to constrain and partially seal in the non-driving direction.

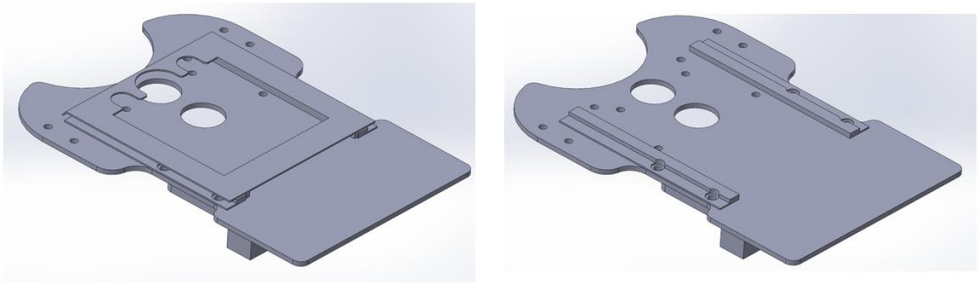

Figure 11: Optimal skirt design.

3.2.1.5 Custom Wheel Iteration

During the initial wheel design, ribs were incorporated into the 3D printed PLA rims with the intention that the molded silicone would bond mechanically to the outer shell and the ribs would help secure the material in place during the torsional loading. However, during manufacturing it was observed that silicone did not adhere to the PLA. As a result, the silicone tires had to be super glued to the rims. This approach introduced a new issue. The preens of the ribs caused the silicone to sit unevenly around the rim, leading to the wheels not being perfectly circular. This non-uniformity resulted in highly inconsistent driving behavior when the robot was commanded to drive straight, without any controller

implemented. This natural instability made PD controller tuning more difficult, especially at higher speeds.

To address this issue, the rims were remade with smooth, perfectly circular surfaces instead of a ribbed structure. This change worked extremely well. The wheels molded cleanly and produced a more consistent result after gluing. The robot was naturally driving more straight which made PD tuning much easier. Figure 12 illustrates the old and new rim design.

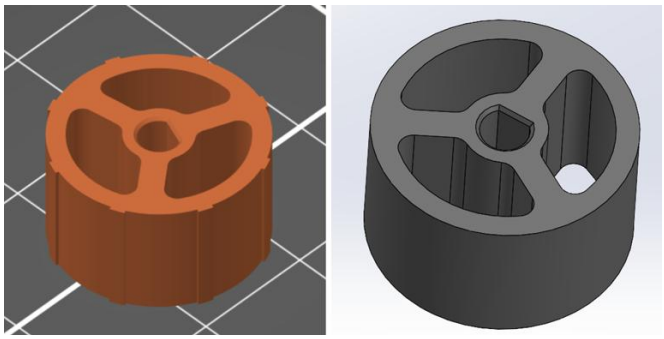

Figure 12: Rim iteration from ribbed (left) to smooth (right)

The rim geometry was not further optimized, such as reducing spokes or thinning sections, since the performance was adequate and given the timeline, there were other challenges to solve first. It should be noted that the manufacturing process remained the same for the silicone tire.

3.2.2 Electrical System Modifications

During the debugging and bring-up phase of the board, several design issues were identified within the electrical system that required modifications. Figure 13 shows the first iteration of the PCB.

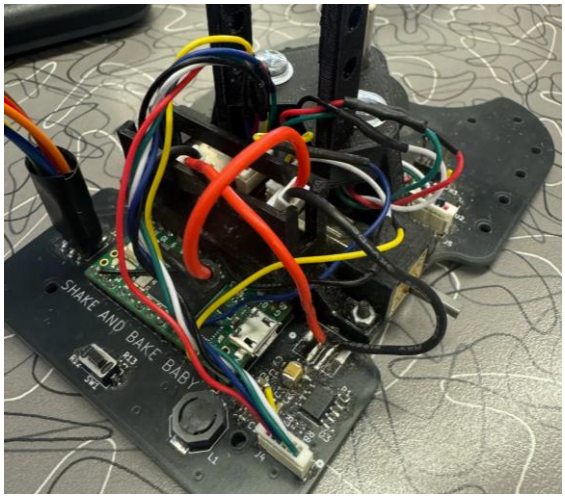

Figure 13: Rewiring of the first PCB design iteration.

The first issue involved the direction of the driving motor connectors on the PCB. During the design phase, the pins were incorrectly labeled, leading to the pins being attached in reverse. This mistake

resulted in the need to cut and re-solder all the wires from the motor to the PCB in the correct configuration.

The second issue that emerged was the limited output current of the 5V buck converter, which could only supply 2A. The major components powered by the 5V rail – including the Teensy, Pixy, Bluetooth module, gripper motor, and fan motor – were drawing more current than initially calculated. The power requirements of the gripper and fan motors were underestimated, and the Bluetooth module was not included in the original design since it was a later request by the software team. After recalculating the power demands, it was clear that a new buck converter with a higher current rating would be required. This change was implemented in the revised iteration of the PCB.

Another oversight involved powering the motor encoders with 5V instead of 3.3V. While the encoders can technically operate on 5V, the Teensy is not 5V-tolerant, so the encoder powering wires had to be fly wired to a 3.3V rail off the Teensy to ensure compatibility.

To improve system diagnostics and prevent further issues, a battery monitoring circuit was also added in the second revision. This circuit used a voltage divider to scale down the battery voltage into the ADC range of the Teensy, allowing a software algorithm to estimate battery levels in real-time. Previously, battery-related issues could go unnoticed and took using a multimeter to diagnose. This new monitoring feature ensures that low battery conditions can be detected and addressed during operation. A dedicated function was implemented in the software to constantly monitor the battery level. When the voltage dropped below a defined critical threshold, the system triggered a visual warning by flashing an LED on the Pixy camera and initiated a full system shutdown to prevent damage to the hardware or unstable operation.

3.2.3 Software System Modifications

During the initial design phase of the custom PCB, several factors were considered to ensure the board would meet all software requirements, with extra care taken to ensure the correct components were selected. However, a significant oversight occurred in this initial design – it failed to consider a debugging tool.

This oversight likely occurred due to the assumption that debugging individual sub-functions could be completed prior to their integration with the main software algorithm. It was also presumed that, once the robot was operational on the course, serial output would no longer be necessary. However, during the tuning of the line following algorithm and development of functions involving object detection, it was evident that a constant stream of serial output would accelerate the development process.

Once the second PCB arrived, compatible with an HC-05 Bluetooth module, it was simple to connect an external device to the Bluetooth module and stream the serial output of print statements to the terminal. Specifically, it greatly simplified the tuning process for object detection thresholds – particularly in defining the minimum object size required in the frame for it to be considered a valid detection.

In the original software system plans, and early development process, the Pixy 2.1 camera built-in line tracking algorithm was used to perform line following. This worked extremely well in early testing, where a white tape line was stuck on the dark linoleum floor in the shape of a box with rounded corners.

With the combination of white tape and dark floor, and the consistent nature of the taped box, the robot could smoothly follow the tape and not veer off course or extensively oscillate.

Once testing started on the actual course though, the inconsistencies in the course became significantly problematic for the Pixy's line tracking algorithm. This was likely due to the slight gaps between wooden tiles, where the tape would not smoothly follow from one tile to the next. In addition, red tape against the medium wood grain was not a significant enough contrast for the Pixy's line tracking algorithm to always select the red tape as the main vector to follow, and it would sometimes select other vectors from random edges on the course. When the main vector was 'lost' or selected incorrectly, the robot's behaviour would become unpredictable, often either overshooting a corner, not turning at all, or following the line with constant small oscillations which could not be reduced through tuning the proportional-derivative (PD) controller.

To mitigate this issue, several adjustments were taken, the most significant being switching from using the Pixy line tracking algorithm to the hue-based colour detection algorithm, called Colour Connected Components (CCC) mode in the Pixy documentation. As opposed to detecting lines via edges and shapes, based on canny edge detection, the colour detection algorithm can detect the red line by learning a specific colour and identifying the red line based on hue and saturation filtering. This proved to be much more effective under a wider range of lighting conditions, surfaces, and inconsistencies in the tiles.

When switching to the Color Connected Components (CCC) mode, the code was configured to query the x-position of the center of the detected red block, as opposed to the x-position at either the head or the tail of the vector in line tracking mode. This x-position was then compared to the known center of the camera's frame to calculate an error value. This error served as the input to the PD controller, which adjusted the robot's steering to maintain alignment with the center of the red block.

An additional advantage of using red line tracking through the Pixy's Color Connected Components (CCC) algorithm was that it allowed the software to operate entirely within CCC mode. This eliminated a need to switch between the Pixy's line tracking and colour detection modes during runtime, ultimately avoiding any potential issues related to time latency and frame-to-frame data inconsistencies. If the Pixy's line tracking API had been used instead, it would have been challenging to programmatically switch to color detection mode without resorting to hard-coded workarounds. This is due to a key limitation of the Pixy camera; it cannot run both the line tracking and color detection algorithms in parallel. By using CCC mode to detect the red line, bullseye, LEGO figure and green box, the software implementation was simpler and more effective.

Another significant change to the overall software system involved a major refactor of the codebase to adopt an object-oriented programming (OOP) paradigm and improve upon organization in the main control loop. Further details regarding the changes in programming style and their impact on system design can be found in Section 3.5.1.

3.3 Testing Plan and Design Verification Results

3.3.1 Mechanical System Test Plan

3.3.1.1 Suction System

Before any on-course testing, the fan motor was tested with the selected batteries. It was found that the motor drew too much power, rapidly depleting the battery capacity preventing reliable operation of other systems. Although the fan and skirt showed very promising results individually as described in Section 3.2.1.3 and 3.2.1.4, the battery limitations made it impractical to reliably operate the robot while the suction system was active. Therefore, the suction system was abandoned to preserve battery life for critical functions.

The test and verification plan for the suction system was to measure the maximum stable cornering speed with and without the suction system active. Given that the suction system added approximately 10g to the robot's overall mass, the robot would need to corner approximately 5% faster with the suction system on to justify the added mass according to the performance index. However, this test was never conducted due to the battery limitations, budget constraints, and timeline pressures.

Although the fan system was abandoned before verification could be completed, it should be noted that the systems performed very well individually. At this point, the team decided to prioritize more essential performance index improvements including reliable LEGO figure acquisition and green box detection.

3.3.1.2 Gripper

The testing of the gripper system was done through observational experiments. The testing was split into two parts, the grabbing, and then the holding process. First, the grabbing was tested by placing the LEGO figure in a variety of places in front of the robot and observing how the figure was clamped when the arm closed. Based on the observations, the shape of the arm was changed to improve the acquisition of the figure. Next, the holding was tested by placing the figure in the ideal spot, closing the arm, and driving the robot over simulated bumps. The video was taken of these tests to review and observe where and how the LEGO figure was shaking or being released. These tests led to the formation of the different backstops. To pass this test, the robot had to grip the figure successfully ten times in a row from the same spot, and then not release the figure when driving over the bumps.

3.3.2 Electrical System Testing

The electrical system test plan focused on verifying that the PCB operated as expected. This process began with inspecting the PCB assembly and carefully bringing it up to ensure the proper functionality of all components. Initially, the board was powered on with a limited current to minimize risk, and the current was gradually increased once it was confirmed that the board was correctly assembled and not drawing excess power. The following electrical testing was conducted in parallel with software development. As various Teensy firmware was executed, the functionality of physical components such as motors was tested to ensure the electrical system was properly configured.

3.3.3 Software System Testing

The software system was developed and tested in stages, beginning with low-level hardware validation and progressing toward full integration of high-level behavior. This staged testing approach ensured that

individual components were functioning correctly before introducing complexity, helping to isolate issues early and reduce debugging overhead later in the process. Testing began with low-level component verification, progressed through subsystem functionality, and concluded with full-system state machine validation.

3.3.3.1 Low-Level Testing

Before implementing any behavioral logic such as line following or turning, each basic hardware component was validated through simple software tests. Each motor was connected to the microcontroller and driven independently at incrementally increasing PWM values in both forward and reverse directions. This test was performed with the chassis elevated, before the wheels were installed, to eliminate ground friction as a factor. It was confirmed that both motors responded appropriately to control signals and were able to spin freely in both directions. This test helped determine the deadband threshold of each motor, and also helped discover that this deadband varied depending on the state of charge of the batteries.

Following motor tests, the encoders were examined. Each encoder was manually rotated, and serial monitor output was used to verify pulse counts. During this stage, one encoder failed to return any values, leading to the discovery of a wiring issue at the signal connection point. This issue was corrected, allowing accurate speed feedback from both motors. Identifying and resolving this before more complex control logic was added saved considerable troubleshooting time later.

Thirdly, testing was conducted to characterize the motor speeds. The step response of each motor was tested using the encoders. We found that, despite using the same PWM inputs, the motors did not rotate at identical speeds (one was naturally less powerful). This was critical in understanding the limitations of our drivetrain and informed our later calibration strategies.

With encoders functioning, additional tests were conducted to characterize the motor speeds and symmetry. The step response of each motor was tested using the encoders. By applying identical PWM inputs and logging encoder feedback over a fixed time interval, it was determined that the motors operated at noticeably different speeds. This discrepancy was attributed to inherent hardware variation. Compensation was attempted via PWM offsetting, where the slower motor was given a higher duty cycle to match the speed of the faster one.

Despite offsetting PWM values of the motors, straight-line driving tests on the ground revealed that the robot still veered consistently to one side. Further investigation determined that the source of the deviation was mechanical, not electrical. The rubber tire on one wheel, which are manufactured in-house using a custom mold, was discovered to be uneven and sloped. This created a lateral force imbalance, causing persistent drift regardless of software compensation. The defective wheel was replaced, after which the robot demonstrated consistent straight-line movement under identical motor control conditions.

3.3.3.2 Mid-Level Testing

Upon successful completion of low-level hardware testing, attention was directed toward subsystem-level software testing. A significant portion of this testing involved configuring and refining the Pixy vision system, which served as the primary sensor for navigation and task execution. The camera was programmed using the PixyMon software to track four distinct colour signatures: red for line following, blue for the bullseye rescue zone, orange for the LEGO figure, and green for the safe drop-off zone.

To reduce false positives and enhance detection reliability, pixel thresholding and region filtering were applied. These filters constrained detection based on the size and position of the coloured blocks within the camera's field of view. For example, blue detection (used to identify the bullseye) was initially limited to the lower-central portion of the frame. This tight region constraint ensured that the robot only triggered the pick-up routine when the bullseye was positioned directly beneath the camera. This improved spatial accuracy and reduced false detections from nearby objects like the rocky terrain or shadows.

However, during extended field testing, it became clear that detection performance was strongly influenced by the robot's speed. At lower speeds, the original narrow region of interest was effective. The robot was able to approach the bullseye slowly and stop precisely when the blue block entered the designated lower-third of the frame. In contrast, at higher speeds, this same configuration caused performance degradation. The robot often overran the bullseye before detection occurred because the blue signature did not enter the narrow detection region until it was too late to react. To address this, the vertical detection bounds for the blue signature were expanded. Specifically, at high-speeds, detection was allowed as soon as the blue block appeared in the middle third of the frame. This increased the reaction time available to the robot, enabling earlier braking and smoother alignment over the target. While this adjustment slightly reduced positional precision, it significantly improved task success at higher velocities by compensating for momentum and potential lateral drift. This observation highlighted the importance of co-tuning sensing parameters alongside dynamic system behavior.

In addition to spatial filtering, lighting conditions were identified as another major factor affecting detection accuracy. Early testing on the track revealed that the robot's behaviour varied significantly depending on the time of day. This inconsistency was attributed to the large windows in the room, which allowed natural light to alter the brightness and exposure levels captured by the Pixy camera. As a result, the colour detection thresholds configured in PixyMon (particularly for the red, blue, and green signatures) would perform differently depending on whether the system was operated during daylight hours or under artificial lighting at night.

To address this inconsistency, an attempt was made to introduce controlled lighting conditions using the built-in bright white LEDs on the Pixy camera module. The hypothesis was that by using the LED lights, a more consistent lighting environment could be maintained regardless of ambient daylight conditions. However, testing with the LEDs revealed several drawbacks. Due to the proximity of the camera to the ground, the intense light output caused the field of view to become overexposed, washing out colour contrasts and degrading the effectiveness of colour block recognition. Additionally, the surface of the LEGO figure was highly reflective, which led to unpredictable reflections and further interfered with accurate detection.

Given these issues, the use of the LED lights was abandoned. Instead, a dual-configuration approach was adopted within PixyMon. Two distinct sets of exposure and colour threshold settings were created: one optimized for daytime use when natural light was present, and another tailored for nighttime or low-light conditions relying solely on overhead lighting. This strategy allowed the robot to maintain reliable performance across varying environmental conditions by manually loading the appropriate configuration prior to operation.

3.3.3.3 High-Level Testing

Higher-level software functions were developed modularly and tested independently before integration. High-level testing focused on confirming that the robot could execute the complete search and rescue sequence reliably: following the red line from the start position, detecting, and retrieving the LEGO figure from the bullseye, identifying a green safe zone for drop-off, and returning to the starting location. These tasks relied not only on the correct operation of each subsystem, but also on the accurate coordination and timing between them.

The PID controller subsystem, while validated earlier in controlled environments, required additional tuning during full system testing to ensure smooth performance when integrated into the full search and rescue routine. For example, the PID controller for line-following had been tested prior to assembling the gripper arm on the servo motor and without the LEGO figure gripped in front of it. Therefore, further testing was conducted to ensure neither of these were blocking the camera's view of the track.

During testing of the LEGO figure pick-up routine, an initial approach was implemented in which the robot used a proportional-only control scheme to align itself with the target. This method employed a standard PID control structure with the integral and derivative gains set to zero. The goal was to allow the robot to adjust its heading and drive slowly into the optimal gripping position, using feedback from the Pixy camera to minimize horizontal offset between the LEGO figure and the camera's centerline.

A primary source of this limitation was the large deadband inherent in the DC motors. The motors required a minimum PWM signal before any movement occurred, and this threshold was sufficiently high that low-speed, fine-grained adjustments were not possible. As a result, when the robot attempted to perform precise movements at close range to the LEGO figure, it often moved too abruptly.

Because the robot stopped very close to the target before initiating alignment, the sudden movements caused by the deadband frequently led to overshooting. In some cases, the robot would collide with the LEGO figure, knocking it out of view of the camera. In other instances, the steering correction would overshoot the target alignment, causing the robot to lose visual contact with the figure altogether. These outcomes significantly reduced the reliability of the pick-up routine.

To address this issue, a more controlled technique was developed. Instead of issuing continuous motor commands through the P controller, a pulsed control method was implemented. In this scheme, the motors were still driven based on proportional error, but rather than maintaining sustained motion, motor output was pulsed in short intervals. Specifically, the motors were commanded to spin for approximately 100 milliseconds, followed by a 75-millisecond braking period. This pattern allowed the robot to "jitter" forward in small increments toward the LEGO figure.

This intermittent motion preserved the basic structure of the P controller, allowing the robot to adjust its heading and position based on real-time visual feedback, but added a critical layer of control suited for fine movements at close range. The braking interval between pulses prevented the system from building up too much momentum, reducing the risk of overshooting or displacing the LEGO figure. As a result, this method provided a more precise and stable alignment routine, significantly improving the consistency of successful pick-up actions during testing.

The gripper mechanism was also tested independently. Manual triggers were used to simulate pick-up and drop-off events, allowing verification of timing and position consistency. The gripper was tested using the

LEGO figure in different orientations such as sitting flat on its side, or face down, to ensure that gripping performance was not dependent on precise alignment. Since the robot occasionally knocked the LEGO figure over while attempting to align itself for the pick-up, it was crucial to have a gripper arm that could work with the LEGO figure in various orientations. Additionally, the 180-degree turning function was tested in isolation. The robot was placed on a track segment where it performed the turn, realigned with the path, and resumed tracking. Adjustments were made to motor speed and turn duration to improve repeatability and prevent over-rotation.

3.3.4 Full System Test Plan

Full system testing was conducted through repeated test runs on the obstacle course. The software system was deemed successful when it demonstrated stable and responsive line tracking, accurate LEGO pick-up only within the bullseye zone, reliable drop-off within the designated green area, and correct transitions between all operational states without freezing or erratic behavior. Each run was closely observed, and results were used to iteratively refine software logic and sensor tuning. This comprehensive testing strategy ensured that the final software and hardware implementation was both reliable and adaptable under competition conditions.

3.4 As-Built Drawing of Final Design

Below are two drawings of the overall final design. Figure 14 illustrates three views and an orthographic view of the assembly.

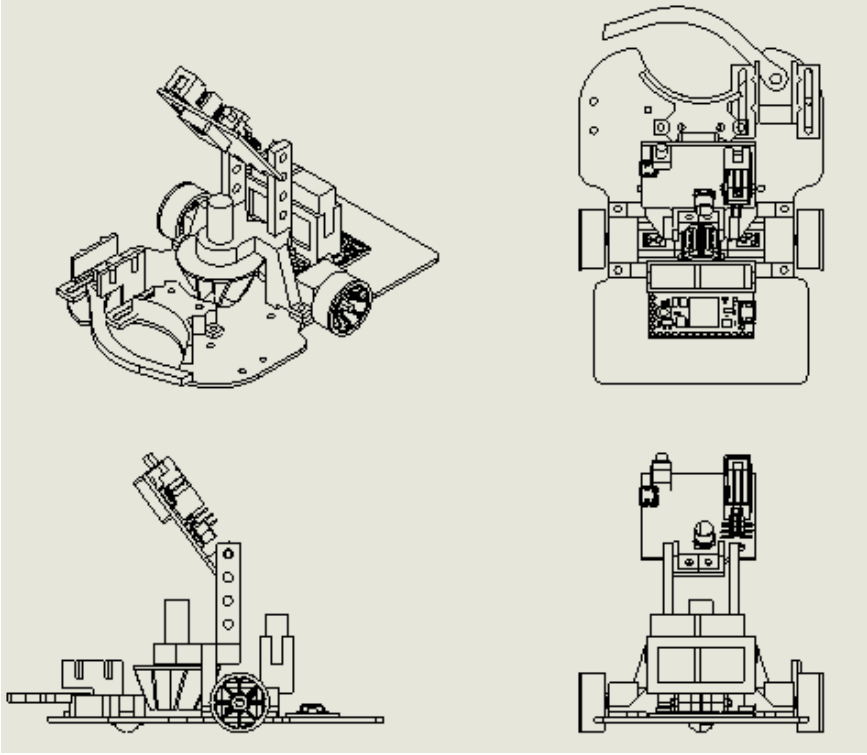

Figure 14: Orthographic Drawing

Figure 15 illustrates the exploded view of all components in the assembly.

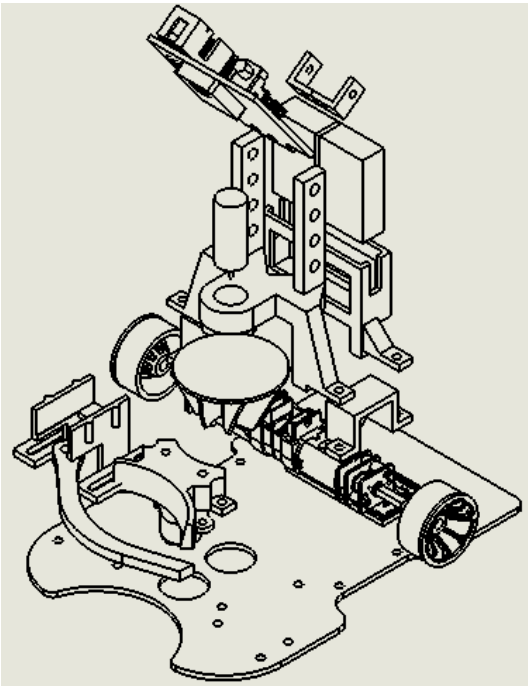

Figure 15: Exploded View

Finally, Figure 16 illustrates the overall dimensions of the robot.

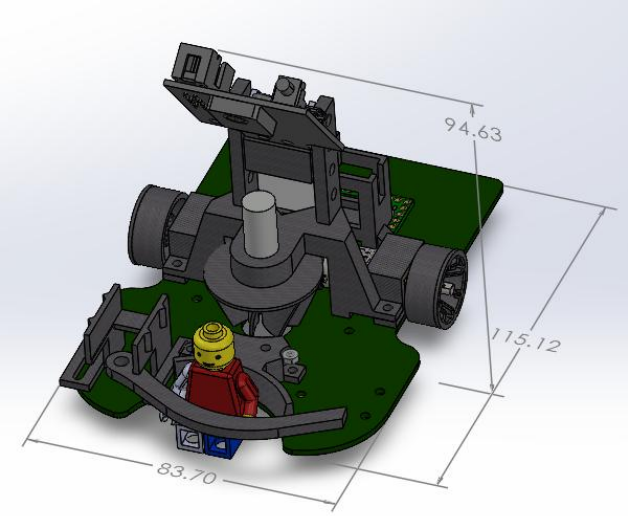

Figure 16: CAD

3.5 Discussion on Design Calculations vs Actual Results

3.5.1 Software Algorithm Design

The initial software architecture of the robot was planned with a combination of flowchart diagrams and pseudocode. In parallel, a hardware architecture block diagram was developed to show the division of responsibilities between the Teensy 4.0 microcontroller (MCU) and the Pixy 2.1 module, which integrates a system-on-chip (SoC) and camera. This hardware architecture diagram also outlined the design

objectives corresponding to specific hardware functions and identified the communication protocols used for data exchange between the Teensy and the Pixy. Overall, the tasks split between the MCU and the SoC did not deviate from the original hardware architecture block diagram. The Teensy MCU executed the main software logic, including the proportional-derivative (PD) controller and motor control, while it queried the Pixy module for color block data. The Pixy handled object detection using its onboard hue-based color filtering algorithm.

While the general structure of the flow-chart diagram was followed in the final code, some details were finalized in the development process that were initially overlooked in the primary design calculations. Specifically, in the pseudo code, the main loop executed a long chain of if-else statements to switch the robot between its main states; line following, LEGO figure pickup and green box detection. To start, the main software program was written in one single Arduino program with this structure, which quickly became very lengthy and difficult to maintain and debug. As a solution for streamlining the code, the decision to switch to Platform IO was made. Platform IO allowed the code to follow object-oriented programming practices, which organized the code around main objects helping to organize the large code base and make the code more reusable and modular.

In the codebase, an object was created for all the main components of the robot, such as the encoders, motors, turn controller and color detection. While this modular design improved clarity and separation, debugging, and updating the main control loop became increasingly difficult as additional robot functions were integrated. To address this, the control flow logic was refactored from a series of if-else statements into a switch statement structure, further improving readability, scalability, and maintainability within the Platform IO environment.

In addition, a few more states were added to the state diagram that were not part of the original design. Initially, the design calculations included a single state for line following, intended to handle all scenarios in which the robot was tracking the red line. However, this was later expanded into three distinct line-following states: one for navigating to the bullseye, one for travelling to the green box, and one for returning to the home position. This design choice was made specifically for the switch case approach, as it improved code clarity and made it easier to determine which segment of the course the robot was currently executing. This also helped clearly define the conditions required for the robot to switch from line following to another task, such as LEGO figure pick-up or drop-off.

3.5.2 PID Controller Design

The initial control strategy for the robot was based on using the Pixy's vector tracking mode to extract a line vector from the red tape path. The PID control system was to calculate the steering error based on the horizontal deviation of the vector's head (x_1) from the camera's center ($x_{\text{center}} = 39.5$ pixels). However, during implementation, this method proved unreliable due to inconsistent vector detection at various lighting angles and partial occlusions during turns. As a result, vector tracking was abandoned in favour of colour block detection, which provided a more stable and consistent response from the Pixy.

Instead of using the vector's x-coordinate, the error was calculated as the distance between the center pixel of the detected red block and the frame's horizontal midpoint. This change introduced several benefits such as reduced false detections and jitter in the error signal and faster response due to more consistent frame-by-frame data.

To further improve performance, a deadband threshold was implemented to ignore small, noisy deviations. This was implemented by setting the error to zero if the absolute value of the error was calculated to be below 10. This prevented the PID controller from making unnecessary corrections in cases where the block was nearly centered, thereby avoiding overcorrection and jitter at low error levels, which improved stability on straight sections of the course.

Initially, the Ziegler-Nichols closed-loop tuning method was used, as outlined in the design calculations memo. Specifically, K_i and K_d were set to zero, then gradually increased the proportional gain, K_p, until the robot exhibited sustained oscillations. This was determined qualitatively through repeated test runs on the obstacle course. At this stage, the ultimate gain (K_u) and the oscillation period (T_u) were attempted to be measured in order to compute the full PID gains using the standard Ziegler-Nichols formulas. However, this part of the method was not successful in practice. The calculated gains led to unstable or sluggish behavior, likely due to external disturbances from surface inconsistencies, sensor noise from the Pixy, or unmodeled dynamics like motor lag and drivetrain asymmetry.

Instead, $0.6K_u$ was used as the initial K_p and a small K_d, then fine-tuned from there. The K_d value was added to reduce overshoot and smooth the response during turns, but Ki was not used at all. Since line tracking did not suffer from consistent steady-state error, the integral term was deemed unnecessary.

The final tuning process was entirely empirical, relying on iterative testing and visual performance assessments. Despite these deviations from the original tuning methodology, the controller ultimately provided reliable, responsive tracking that allowed the robot to complete the Game Day challenge effectively.

3.5.3 Battery Calculations

During the design phase, calculations were performed to select the optimal LiPo battery for the robot. Key factors considered included voltage compatibility, capacity, and discharge rate. Additionally, because there is a positive correlation between battery size and capacity, the goal was to select a battery that met the necessary specifications with the minimum capacity required.

Initially, the calculations compared various battery capacities ranging from 500 mAh to 2000 mAh. The reason for considering these high capacities was the difficulty in finding batteries with a sufficiently high C-rating. Maximum current draw is influenced by both battery capacity and C-rating, so in order for the robot to draw the significant current expected, the battery needed either a higher capacity or a higher C-rating (most batteries found did not exceed 4C). As higher capacities were initially easier to find, this option was initially prioritized.

Based on these considerations, the calculations led to the selection of a 1000 mAh LiPo with a 2C rating. However, after further analysis, a 220 mAh 3.7 V battery with a 25C rating was found. Despite its lower capacity, the high C-rating of this battery allowed it to meet the required power output. When plugged into the original calculations, it was determined that this battery could still fulfill the requirements.

The new batteries were purchased, and as predicted by the calculations, they were able to provide sufficient power for the robot to operate throughout the course. However, the battery ultimately fell short when powering the fan motor, which turned out to be significantly more inefficient than expected. This impacted on the ability to properly integrate the fan on test day, as there is a correlation between motor

speed and battery voltage. With the high current draw from the fan, the battery voltage dropped rapidly, reducing performance and preventing the robot from reaching high speeds. As a result, the fan could not be reliably used, and overall system performance was negatively affected. This highlighted the need to account not just for peak current requirements, but also for sustained power delivery and total energy capacity. In future iterations, selecting batteries with a higher overall capacity, even at the same C-rating, will be critical to ensure stable voltage rails and reliable integration of all components.

3.5.4 Gripper Design

Torque calculations were performed during the design phase to ensure the selected servo motor could securely grip and transport a LEGO figure without slipping. Using a conservative assumption for plastic-to-plastic contact, $\mu = 0.3$, and maximum arm length, 0.08m, the minimum required gripping torque was calculated to be 0.00785 Nm and gripping closing rate to be a maximum of 0.005 s/degree.

The FEETECH 0403 micro servo was selected because it exceeded both the torque and speed requirements, 0.05886 Nm and 0.0015 s/degree respectively. These values are 7.5x larger and 3x faster than required, meaning there is a large factor of safety. Testing confirmed that the servo could reliably grip the LEGO figure under static conditions. However, during driving tests across the bumpy course, vibrations and inconsistencies in the course occasionally caused the LEGO figure to eject from the gripping mechanism. To improve stability, foam was added to the gripper surfaces to increase friction and reliability since the foam would wrap around the LEGO figure.

Later in testing, the original servo failed. A replacement servo with similar specifications was found and installed. After replacing the servo, the gripping system worked reliably through multiple pickups and drop-offs, confirming that the design calculations met the project requirements.

3.5.5 Driving Motor Calculations

To properly select motors, calculations for the required torque and revolutions per minute were performed. To calculate the torque required, an acceleration of five meters per second squared was selected as a goal and used the equivalent inertia of our system to find the torque required. Then, it was determined that our max velocity would be 1 meter per second. After converting the linear velocity to angular velocity in revolutions per minute a motor that fits both of our requirements with a factor of safety of 30% was selected.

4 Review and Reflection

4.1 Challenges and Lessons Learned

The development of the robot presented a variety of challenges across mechanical, electrical, and software domains. Many of these challenges provided valuable insight into system-level design and highlighted the importance of collaboration and iteration. This section outlines key lessons learned over the course of the project.

4.1.1 Lessons Learned from Design Process

Several lessons were learned throughout various phases of the design process. One of the most critical lessons was the importance of iteration, not just within subsystems, but across them. The robot was composed of mechanical, electrical, and software subsystems, and a system-based design verification process was followed: mechanical systems were verified first, followed by electrical, and finally software. While this sequential approach made it easier to isolate issues within a specific domain, it ultimately slowed the identification of cross-system issues due to the interconnected nature of the platform. It became clear that iterating across subsystems, rather than within them in isolation, was more effective for diagnosing and resolving complex problems.

For example, during the software integration phase, repeated efforts were made to tune the gripper's actuation through software. However, progress was limited until the root of the issue was reconsidered from a mechanical perspective. After making slight adjustments to the gripper design, the functionality improved significantly. This reduced the total time spent debugging and highlighted the value of reallocating troubleshooting efforts across subsystems, rather than assuming a single domain was at fault.

Another lesson learned was understanding when and where to over-design for robustness. A specific example of this occurred in the electrical subsystem. Through testing, it was found that the initial selected buck did not have a high enough max current rating based on experimental values. Although, theoretically, it should have been able to provide enough power, it was found that the buck could not supply enough power to the 5V rail. As a result, a different buck had to be chosen for the new PCB. This time, the selected buck was chosen to be more robust, ensuring it could handle the power requirements with plenty of margin for safety. Going forward, approaching design in this way will reduce the amount of redesign needed. Especially for critical parts of the system, such as power.

Another critical lesson involved the risks of treating third-party APIs as "black boxes" without fully understanding their internal behaviour. Early in the software development process, the Pixy API was integrated without a clear understanding of the performance implications of its functions. Specifically, the getBlocks() function (used to retrieve color block data from the camera in Color Connected Components (CCC) mode) was assumed to be lightweight and non-blocking. As a result, it was called independently within multiple functions throughout the main loop. However, the function is actually a blocking call that retrieves only the most recent frame of vision data and overwrites the internal buffer each time it is executed. Because of this, multiple calls within the same loop iteration caused each subsystem to operate on a different frame of visual data, resulting in inconsistent perception and erratic behaviour.

The unintended consequence of this design choice was a form of data desynchronization, where line-following, bullseye detection, and drop-off zone detection logic were reacting to different views of the environment within a single control cycle. This mismatch introduced instability into the steering response, as corrections were often based on outdated or partial frame data. Compounding the issue was the cumulative delay introduced by calling the blocking function multiple times. While the delay of a single call was minor, calling it redundantly in the same loop significantly increased the loop's execution time, reducing the frequency of control updates. This delay caused the robot to react sluggishly to dynamic changes in its environment, particularly at higher speeds, and contributed to a "chasing" effect where the robot continuously overcorrected its position in response to delayed or inconsistent error inputs, manifesting as oscillations during line following.

To resolve the issue, the software architecture was restructured to include only a single call to getBlocks() at the start of each loop iteration. The resulting data was stored and passed to all relevant functions as a parameter. This ensured that all subsystems operated on a consistent and synchronized frame of visual input while also minimizing unnecessary communication delays. Once implemented, this change eliminated the oscillation problem entirely and significantly improved system responsiveness and stability. This experience emphasized the importance of understanding the performance characteristics of external libraries, particularly in time-sensitive embedded systems. Going forward, greater care will be taken to analyze and document the behaviour of third-party functions before integrating them into critical control loops.

Another valuable lesson was the importance of considering hardware-related causes for behaviors that initially appeared to be software issues. One such example occurred during testing, where the robot exhibited inconsistent line-tracking behaviour and unpredictable navigation patterns. At first, extensive time was spent debugging the software, particularly the PID controller and Pixy data-handling logic. However, after exhausting potential software-based explanations, it was discovered that the Pixy camera was had physically shifted in its mount, preventing it from detecting the line in the same manner that it had been tuned in. After testing the robot with reliable working versions of the code, and still observing abnormal line following behaviour, the root cause was identified through physical inspection of the robot's setup. This experience reinforced the importance of not treating sensor inputs as infallible, and of validating hardware configuration before investing extensive effort into software-side debugging. From that point forward, more rigorous alignment checks, and calibration protocols were put in place and will be incorporated during the integration phase to catch such issues earlier in the process.

4.1.2 Lessons Learned from Project Planning

A number of important lessons emerged from the planning and coordination aspects of the project. While the milestone structure provided by the course was effective for guiding the overall timeline, it quickly became apparent that detailed daily or weekly planning was not always feasible. The unpredictable nature of hardware testing, the frequency of integration issues, and the tight coupling between subsystems often led to non-linear progress, making it difficult to adhere to rigid schedules. As a result, flexibility became an essential aspect of day-to-day planning, and the team adopted a more adaptive and iterative approach to time management.

One of the clearest takeaways was that project planning always takes longer than expected. Whether tuning the line-following controller or verifying gripper design, tasks regularly required more time than initially budgeted. Problems also tended to emerge during the most time-sensitive phases, reinforcing the need to build in buffer time and avoid assuming anything would "just work" during final integration.

Team dynamics also played a key role. Initially, responsibilities were divided between mechanical, electrical, and software roles, but this structure became inefficient during system integration. The team worked far more effectively when everyone collaborated in the same space, enabling real-time debugging and faster iteration. Each member contributed across disciplines wearing "every hat" as needed, which improved both efficiency and overall system understanding.

4.1.3 Lessons Learned from Game Day

On game day, a significant learning experience for the team was determining how to race the robot in a way that maximized all elements of the performance index. This involved balancing consistency and speed. By starting with 2 conservative runs to secure a solid baseline score, followed by a final high-speed run aimed at achieving the best possible time result the highest reliability factor and time was obtained.

For the very first run of the robot, the motor speed during line-following was set to a base speed of 66 PWM. The proportional (P) and derivative (D) control values had been extensively tuned and tested beforehand to ensure a smooth and reliable performance at this speed. This speed was also chosen for a safe first run because most of the testing had been conducted at this level, as it was the slowest PWM at which the motors could consistently operate without stalling.

For the next run of the robot, the same speed was chosen to ensure that there were at least two successful completions. While the initial intention was to run the second attempt at a slightly faster speed, it had been observed that the slightly higher PWM value, 71, that had been extensively tested did not actually produce a faster result as the stopping was ability of the robot at the bullseye was less consistent and often forced the robot to take longer to secure the LEGO figure in the perfect gripping position. As predicted, the robot successfully completed this attempt, increasing the reliability score to 0.9 from 0.75, but took the exact same amount of time, 15 seconds, to complete the course.

In the final run of the robot, a strategic decision was made to prioritize speed over reliability. This was based on the competition scoring criteria, where only the fastest of the three runs is used to calculate the performance index score, rather than an average. To ensure that the robot would still have the best chance of a successful attempt, as this was needed for the time to count towards the fastest time, additional test runs were performed using significantly higher base PWM values (90 – 125). By this stage, extensive testing had already been completed for the robot's proportional and derivative (P and D) control coefficients, making the process of adjusting and fine-tuning these values through trial and error relatively straightforward.

After a few more tests, the optimal base PWM value for line-following towards the bullseye was updated to 95. To accommodate the increased speed and reduce risk of slippage, slight modifications were made to the bullseye stopping criteria. Once the robot had successfully picked up the LEGO man, and began its traversal toward the green box, the base PWM was reduced back to 66. These adjustments were made to improve accuracy during approach of object detection, ensuring that the robot could reliably locate both the green box and the bullseye. Despite the modifications made to ensure secure retrieval and drop-off of the LEGO man, the total time to complete the course on the third attempt was still reduced to just 13 seconds. This demonstrated that the adjustments for accuracy did not compromise the increase in speed and instead contributed to a successful final run.

Overall, some key takeaways from Game Day included learning to trust personal intuition when tuning proportional and derivative coefficients on the spot, as well as developing the ability to quickly iterate and apply insights gained from test runs to improve performance in real time.

4.2 Recommendations

The next steps for the mechanical team would be to further develop and optimize our systems. First, there would be mass reductions through using a smaller servo motor, as the servo motor used provided too much torque, and was unnecessarily large. Next, the 3D prints would be optimized to further reduce mass by cutting away any material that is not integral to the robot's operation.

Next, the fan assembly would be further developed to increase our speed around corners. The fan assembly did not work as intended on the final design, so the next step would be to find the source of the issue and create an iteration that solves it. This would improve our time as the added downforce would allow the robot to reach a higher top speed.

The next step for software would be to swap the camera for a lighter camera, reducing the mass, and then writing a new line detecting software that is more reliable than the Pixy API software. This would allow the robot to be adapted to different situations and allow for more customization of the program.

Finally, all Dupont style connectors would be replaced by a more permanent solution like solder or glue. These connectors are unreliable as they often slip off pins, severing important connections. A more secure solution will increase the reliability of the robot since the wires would no longer slip out.

4.3 Conclusions

This term, the team was challenged by the MTE380 teaching staff to design and build an autonomous line-following robot. Key design constraints included minimizing weight and cost, ensuring ease of integration, managing design complexity, and maximizing reliability.

Over the course of the term, the team developed multiple prototypes and continually iterated on the design as challenges emerged and improvements were identified. Notable issues included insufficient power delivery on the 5V rail of the PCB, a gripper design that struggled with consistent pickup and drop-off, and unexpected software behavior from various third-party APIs. Addressing these challenges required a mix of strategies, including a full redesign of the PCB, close cross-functional collaboration on the gripper mechanism, and a deep dive into the open-source software to better understand and debug its behavior.

Despite these hurdles, the project was a success. On competition day, the robot placed first in both time and weight categories, achieving a fastest run of 13 seconds, a weight of 131 grams, and a perfect reliability score.

This experience offered valuable lessons in iterative design, system integration, and team-based problem solving. The team looks forward to applying these skills in the upcoming MTE481/482 capstone projects.

MTE 481/482 Proposal

1 Needs Analysis and Problem Statement

With the rapid advancement of battery technology, adoption across multiple sectors including electric vehicles (EVs), grid energy storage, and home battery backup systems are just a few.

With the rise in high energy batteries, this has led to an increase in fire related incidents involving these systems. Fires caused by battery failures are self-sustaining, extremely toxic, and difficult to extinguish using traditional methods.

Fire fighters responding to EV and battery fires face a few critical challenges:

1. Limited visibility into thermal runaway hotspots within the battery packs.
2. Unpredictable fire behaviour leading to increased risk of explosion and re-ignition.
3. Inefficient suppression tactics that waste resources, like water, and time (initial studies say 10-60x more water as an internal combustion engine vehicle).

Therefore, there is a need for an autonomous device that maps thermal and gas hotspots and strategically targets suppression efforts to improve firefighter safety and effectiveness.

1.1 Needs Analysis

By designing and building a low-profile autonomous robot capable of mapping heat and gas distributions across EV battery packs, this project aims to contribute to the development of a reliable, low cost, and field deployable technology that will improve modern firefighting operations.

This robot will allow firefighters to:

1. Rapidly locate critical hotspots.
2. Visualize dangerous conditions beneath burning vehicles.
3. Target cooling efforts precisely, improving fire suppression and reducing risk.

1.2 Problem Definition

Design an autonomous mobile robot capable of mapping thermal and gas concentrations under electric vehicle battery packs during fire emergencies. The robot must navigate under damaged vehicles, general live data maps, and optionally deploy water to the most critical hot spot areas.

1.3 Objectives/Constraints/Criteria

Objective	Success Criteria
Map thermal hotspots	Accurately identify areas above 80 (C) under EV body
Detect hazardous gases	Detect battery pack off gas concentrations
Autonomous navigation	Robot maps 90% of EV underside within 30 seconds
Portable deployment	Robot deployment in under 30 seconds
Manual control override	Operator can manually operate robot if autonomy fails

Constraints	Description
Size	Must not exceed 20cm in height and 60cm in length and width
Weight	Less than 30 lbs
Temp Resistance	10 min exposure to 1000 (C)
Cost	Less than $2000
Battery life	Minimum 30 min under full load

Criteria	Description
Thermal mapping accuracy	Compare designs based on resolution and accuracy of hot spot detection
Gas detection sensitivity	Compare designs based on minimum gas concentration reliably detected
Traversal reliability	Compare designs based on how reliability each design covers the underside of the EV without manual intervention
Deployment speed	Time to fully deploy and activate the robot at scene
Weight	Lighter designs are preferred
Cost	Lower cost designs are preferred
Durability	Robot survival under high temperature environments

1.4 Overview

Hot Spot Robotics will aim to develop a small, rugged, autonomous mobile robotic system that improves firefighter response to battery fires by mapping and locating critical hot spots which will help firefighters act smarter, faster, and safer.