Robotics Challenge - Final Report
Team 404
Robot: Passive-Adaptive Terrain Object Navigator (PAXTON)

Morgan Zhang – 24007835

Ian Yin – 24025142
Xavier Parker – 24077390
Helitha Cooray – 24161798

06/06/2025

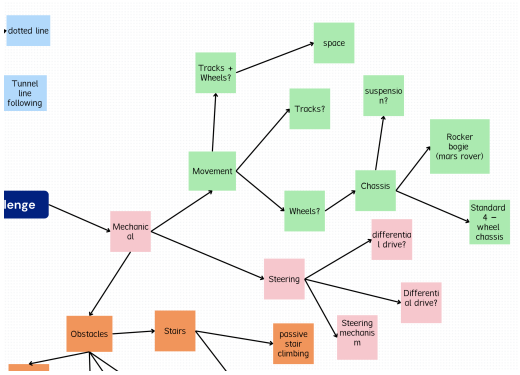
Contents

1 Mechanical Design

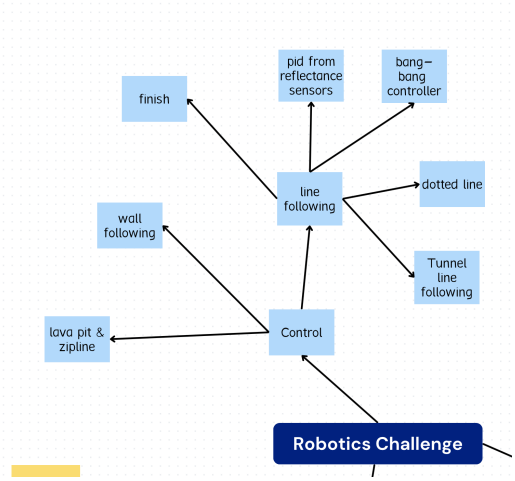
1.1 General Project Outline

1.1.1 Mind Map

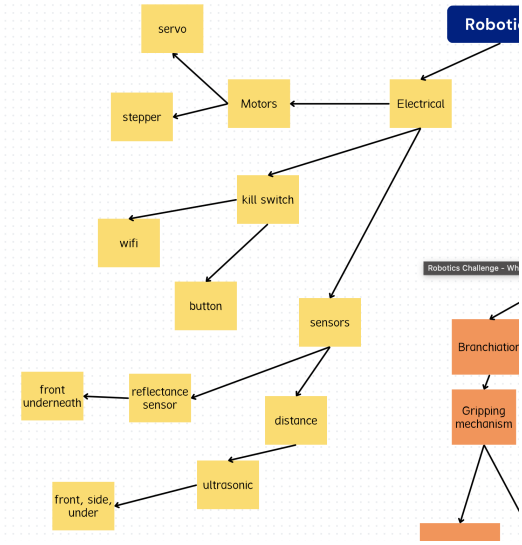
After the launch presentation on Monday, we did our research overnight and drew up a mind map with four key areas: Control/Programming, Electrical, Mechanical, and Obstacles.



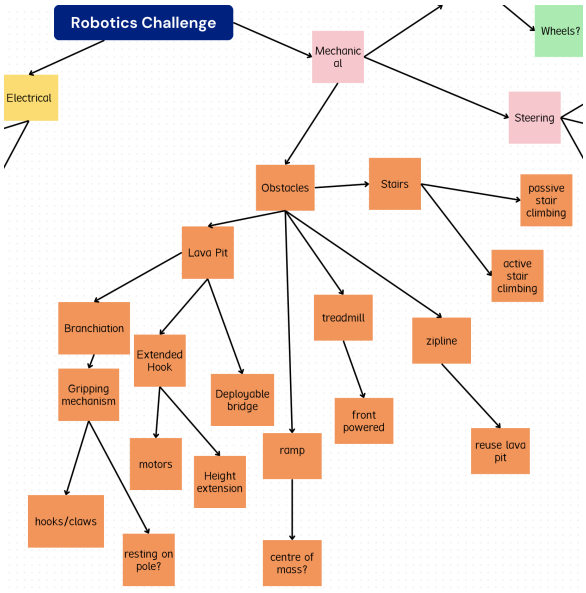
(a) Mechanical Component of the Mind Map



(b) Control and Programming component of the Mind Map



(c) Electrical Design Mind Map



(d) Obstacle Considerations Mind Map

Figure 1: Sections from the overall mind map for each core design area of the project

1.1.2 Timeline

Week 1

- Considered two initial mobility concepts: tracks and rocker-bogie.
- Prototyped both using lab materials.
- Attended the launch of the **Handbook on Soft Robotics** and received advice from Dr. Angus B. Clark (Imperial College London) to investigate the PaTS wheel by Godden et al [3].
- Laser-cut a cardboard version of the PaTS wheel and evaluated all three mechanisms.
- Chose to proceed with the PaTS wheel based on performance and versatility.

Week 2

- Team split into subgroups to work on electronics, sensors, and chassis design.
- Prototyped the PaTS wheel in different materials and settled on TPU.
- Main mechanism mostly built; overall robot structure clearly defined.
- Active steering mechanism constructed and tested.
- Planned layout of all components on the chassis.

Week 3

- Objective: have a working prototype ready for the trial run.
- Built and tested a robot capable of crossing the lava pit and engaging the zip line.
- The Assembly process is understood to prepare for a more robust final build.

Week 4

- Began working on the line following control.
- Re-designed the gripper mechanism to its final form.
- Completed perforated board soldering for the final robot.
- Began testing the line following code on simple tracks.

Week 5

- Verified that the final assembly is fit for the trial runs and the challenge
- Finalized the line following and wall following control tuning.
- Began wall following testing with turns.

Week 6

- Observe the track layout for each section and make any code adjustments for line and wall following, both for the trial runs and the challenge.
- Debugged hardware or software issues and made last-minute adjustments based on the trial performance.
- Additional preparation for the final run.

1.1.3 Milestones

Task	Date
Team Formation	2025-04-28
Brainstorming	2025-04-29
Prototyping – 3 mechanical variants	2025-04-30
Confirmed the mechanical approach	2025-05-02
Completed chassis modelling	2025-05-04
Soldering sensors and perforated board	2025-05-05
Testing the sensors	2025-05-06
Finished printing the TPU wheels	2025-05-07
Chassis Assembly	2025-05-13
Testing the robot on the obstacles	2025-05-14
Lifting mechanism and first trial run	2025-05-15
Finished basic line following and wall following control	2025-05-21
Final gripper model design and manufacturing	2025-05-22
Final Tuning and testing of wall following and line following logic	2025-05-22

Table 1: Project Milestones

1.2 Introduction

Meet **PAXTON**, the **P**assive-**A**daptive **T**errain **O**bject **N**avigator. This robot is based on the course assignment for this semester and aims to design and implement a mobile robot that can perform a variety of typical subtasks, including line tracking, obstacle avoidance, tunnel navigation, stair climbing, driving on uneven surfaces, and pole traversing, etc. However, this multi-terrain and multi-task mobile robot could not only be used in the term project, but also has broad application prospects in the fields of industrial AGV, warehouse inspection, and urban search and rescue.

As a mobile robot capable of completing multiple tasks which could be shown in the above paragraph in complex environments, the core requirements include:

- Line-Following: The robot needs to drive steadily along the predetermined route and avoid obstacles on the road. In terms of this predetermined route, there are straight lines, curves, dashed lines, forks, right-angle turns, and end points. The most challenging part is the right-angle turns. Because the two right-angle bends are very close to each other, a slight mistake can cause the robot to miss the black line or for the sensor to pick up multiple lines, leading to program confusion. Therefore, we need to design a robot that can nimbly navigate the competition field while quickly traversing the course.
- Stair Climbing: When faced with obstacles of varying heights or standard steps, the robot must have sufficient traction and climbing capability, while remaining stable on soft or

uneven terrain. Since in these situations the robot's chassis is very likely to make contact with the ground or obstacles, we need to place special emphasis on its design. And since we have size constraints, we must either use a simple structure or an extendable one.

- Energy Efficiency: Since we have limited voltage – only an 11.1V lithium battery can be used, it is challenging for us to design such an intensive but powerful robot that could tackle everything. Besides, limited voltage also represents we have to minimize the weight of the robot and do not use excessive components to divide the power supply.
- Safety: Even though this is just a small robot, improper measures can still be very dangerous. For example, take the battery's placement: if we mount wood or metal next to the battery, then in the event of a fire, it would spread extremely quickly. Furthermore, if the battery isn't positioned in a location that allows for easy removal, it becomes difficult to stop a developing battery fault before disaster strikes. For the electronic components, we need to solder in a power switch so that, in special circumstances, we can immediately cut off the power.

Based on the above considerations, we initially had many ideas. Although a tracked vehicle offers excellent obstacle cross capability and low ground pressure, its steering is dependent on differential braking, resulting in a large turning radius, low efficiency under repeated maneuvers, and increased wear and energy waste during long-term operation. Omni-directional or Mecanum wheels can achieve zero-radius rotation and an extremely small turning circle, but their complex structure drives up manufacturing and maintenance costs, and lateral slippage on rough terrain degrades positioning accuracy while increasing energy consumption. After a one-week testing process, we eventually picked an Ackermann chassis robot with four passively transformable single-part wheels.

1.3 Mechanical Design and Manufacturing

1.3.1 Prototyping

Our prototyping process began with selecting a chassis capable of reliably traversing the varied terrain of the obstacle course, which included ramps, stairs, narrow paths, and 90-degree turns. It was essential that the robot maintained stability, traction, and ground clearance across all sections of the challenge.

During the initial design phase, we considered two options: a tracked chassis and a rocker-bogie mechanism inspired by the Mars rover. To evaluate their performance in the context of the challenge, we developed scaled-down prototypes of each system using laser-cut acrylic and plywood, along with wheels available in the lab.

Rocker-bogie The rocker-bogie mechanism was inspired by Mars rovers launched by the NASA Jet Propulsion Laboratory [7], because of its proven ability to operate in extreme terrain. The simplified version we developed featured a six-wheel configuration (three on each side) and a two-part body. The front section consisted of a rocker arm carrying four wheels, which was connected to the bogie via a revolute joint, allowing it to swing freely. The bogie itself supported two wheels and functioned as a load-distributing linkage, similar to those used in semi-truck suspensions.

During testing, the rocker arm did not behave as expected; instead of moving upward, it swung backward, compromising stability. We considered restricting the backwards-motion of the arm,

but further analysis revealed that the spacing between the two front wheels significantly affected the system's ability to handle sudden elevation changes. Our initial prototype had insufficient spacing, leading us to redesign the rocker arm with a longer shape to improve ground clearance.

This adjustment proved effective in navigating the steps; however, the increased length introduced new challenges. The extended chassis made it difficult to maneuver through sharp turns and tight sections, such as the line-following tunnel.

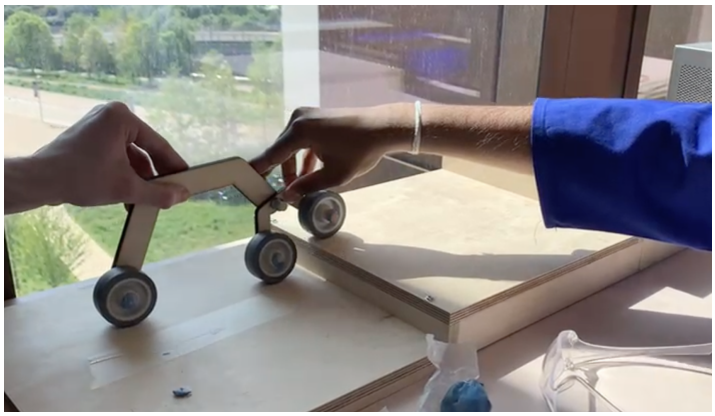
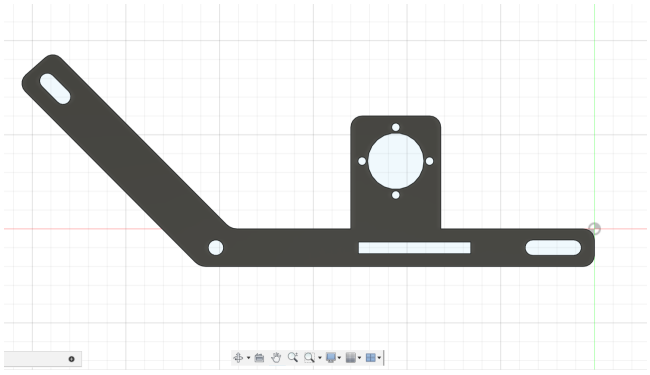
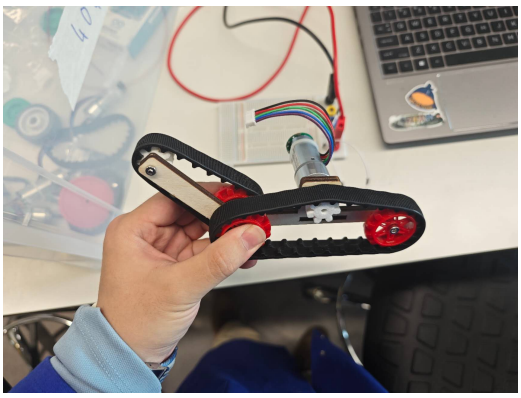


Figure 2: Mars Rover Prototype

Tracked Chassis As for the second strategy, we could build a tank chassis for mobility needs. Since its inception, the tank has been able to adapt to all kinds of roads and environments [6]. Wide tracks increase the force area and reduce pressure while allowing the robot to easily cross small bumps. Two active wheels and multiple passive wheels provide the robot with constant power and solid suspension support. [2].



(a) 3D CAD model of length-adjustable double-layer tank track structure



(b) Prototyped tank tacked chassis

Figure 3: Tack mechanism design

However, tracked chassis also have several drawbacks. There are only two common materials that can be used for a track: one-piece produced rubber tracks and removable plastic track plates. For rubber tracks, the soft material and surface pattern bring stronger adhesion to the robot, helping it to achieve fast climbing. However, due to the fixed length, the size of the robot is limited and cannot be extended freely. With plastic track plates, we can indeed quickly

adjust the length of the tracks and adapt them to different ground surfaces. Nevertheless, due to the properties of the plastic itself, the traction on climbs is reduced to some extent. Another problem that tank tracks can't solve is steering. Since the tracks can only steer and turn in place with different power sources on each side, the lateral pressure created by the grinding and friction of the tracks makes turning more difficult. Considering that our mission requires high agility of the robot, which needs to change direction quickly to patrol lines and avoid obstacles, we finally abandoned the tank mode.

Conclusion Although both the tracked and rocker-bogie systems demonstrated potential in navigating uneven surfaces, neither prototype was able to overcome the stair section of the obstacle course. The rocker-bogie mechanism, in particular, climbed the steps but did not have a high success rate. Likewise, the tracked chassis did not struggle with the first step but did not perform well with the final step. At this stage, the stair section remained the most difficult obstacle to overcome.

1.3.2 Mechanical Analysis

Our robots were designed to have three mechanical parts. The movement part, in other words, the chassis, is most essential and fundamental. Our requirement for the chassis is handling the first part of The Grand Escape, which the line following requires rapid steering capabilities, giving the robot the ability to follow lines on the ground and turn quickly, even making continuous 90-degree turns. Moreover, the treadmill and the ramp acquire sufficient power to overcome the resistance force and keep moving forward.

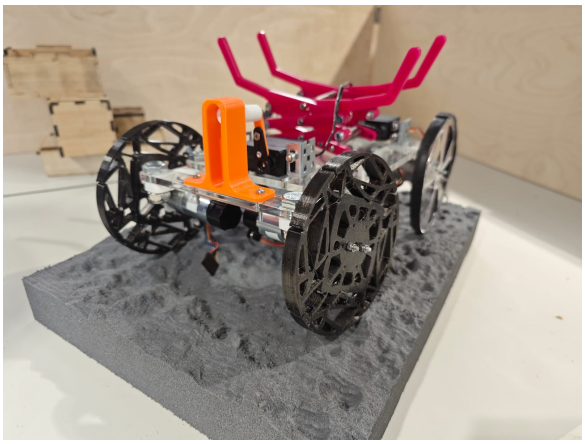


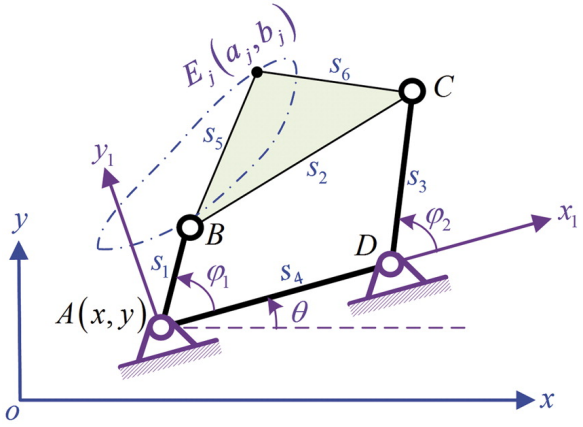
Figure 4: Our prototyped robot

The chassis should also be capable of dealing with the lunar surface and stair climbing, which requires a robot with good off-road capabilities that can easily go over assorted obstacles.
The second part is a gripper, helping the robot to go over the lava pit using the beam hanging from above. It is noteworthy that the gripper should not only be steady enough to take the majority of the weight of the robot while hanging, but also be precisely controlled to catch the zip line and support the robot with only the work done by gravity. The third component of the robot is various attachments and support for electronic equipment, including several servo and motor holders, a battery case, and an Arduino holder.
Last but not least, a tunnel limits the dimensions of the robot, especially the width and height, to below 30 centimeters. Using controlling methods, the robot should have the capability to make an autonomous 90-degree turn in a narrow tunnel.

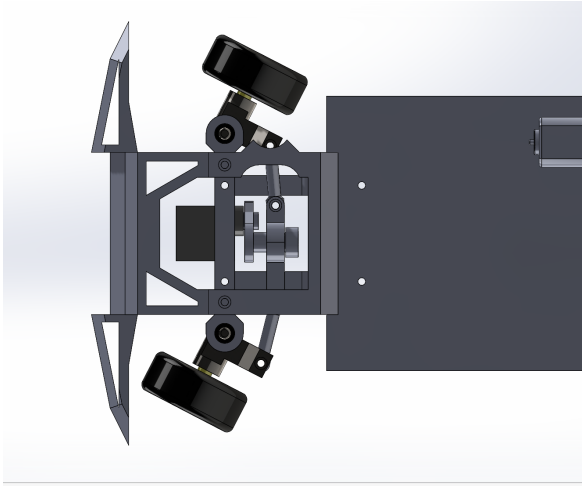
Steering Mechanism

The Steering Mechanism provides the robot capability to smoothly change its direction without significant friction between the ground and the wheels. Using differential speed control at the same time allows the robot to do a good job of patrolling and obstacle avoidance.

Initially, we discussed the application of Ackermann steering mechanisms. It is constructed by a four-bar linkage, which is usually arranged to create a parallelogram or trapezoid structure. One steering power, usually a servo or motor, can twist the parallelogram structure, causing the wheels on both sides to turn simultaneously. As a result, the Ackermann steering mechanism decreases the difference from the robotic centres over the desired steering angle, while efficiently using the limited volume on the robot and fitting into a reasonable space [10].



(a) Trapezoidal Ackermann steering structure [10]



(b) Paralleled Ackermann steering structure

Figure 5: Ackermann steering structure

After considering modelling, 3D printing, and fabrication accuracy, we ultimately chose to use a parallel Ackermann structure. Although the standard Ackermann structure has been popularly used in the industry for many decades, it is inevitably overpopulated with parts and requires very precise connections and tuning because of the need to make room for the servos that control the steering. The robotic chassis also needs to be redesigned in order to accommodate the whole Ackermann steering structure.

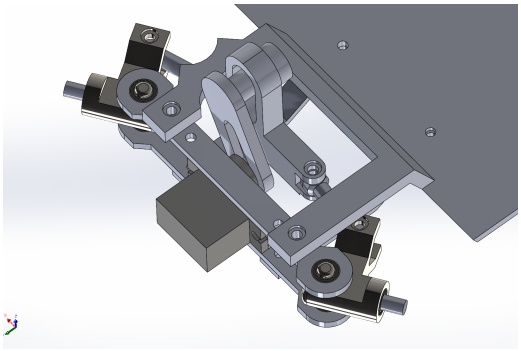


Figure 6: Sophisticated Ackermann structure

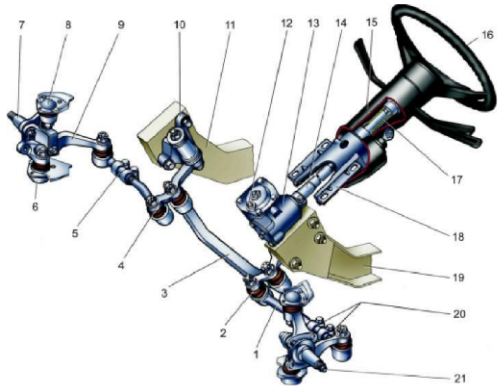
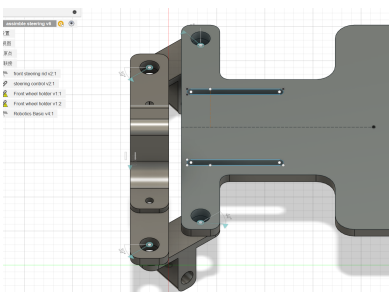


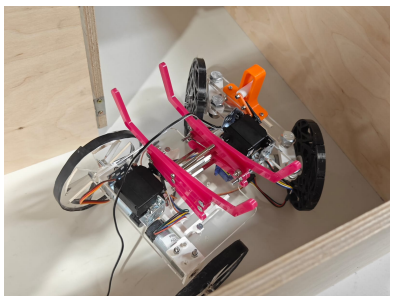
Figure 7: Steering Components

Therefore, we chose to simplify the steering structure, referring to the Ackermann steering mechanism. We connected the two small steering levers that were originally connected to the steering control levers into one. At the same time, the servo, which follows the steering device and slides from side to side, was fixed to the chassis of the robot for better circuit control. While the servo drives the steering wheel axle to slide left and right in the steering control slot, the wheel axle also slides back and forth in the steering control slot because the height of the parallelogram is also reforming as it is twisted.

We found the range of backward and forward sliding by binding and simulation in CAD, so that the steering wheel axle can be made precisely. On top of that, we added two motors to the front wheels for simultaneous drive and differential control of all four wheels. At present, the maximum turning angle of the robot can reach 55 degrees, and it can easily make a right-angle turn in the tunnel.



(a) Simplified Ackermann steering structure [10]



(b) The servo powers the steering mechanism

Figure 8: Ackermann steering structure

Lifting Mechanism

Throughout the whole competition field, there are two sections that requires this lifting mechanism. The first one is a platform with a circular bar suspended above it, whose lower segment is missing. We have to use the pole, the wall, or an extendable part of the robot itself to get though. The second part is the final one – A rope running from a high point down to the ground. The robot must use this rope to slide down from a high platform to the ground.

If we are to use the wall and an extendable part, it would be more stable than using a pole, but the structure would also become more complex. A more complex design requires many more components, which increases the robot's weight. More weight negatively impacts the robot's agility, especially in line-following and obstacle-avoidance scenarios.

Furthermore, since the final section requires the robot to use a hanger-like part to slide down from a height, we decided to use the lifting mechanism directly and integrate the hanger with it. This not only eliminates an additional structure to design but also lightens the overall robot.

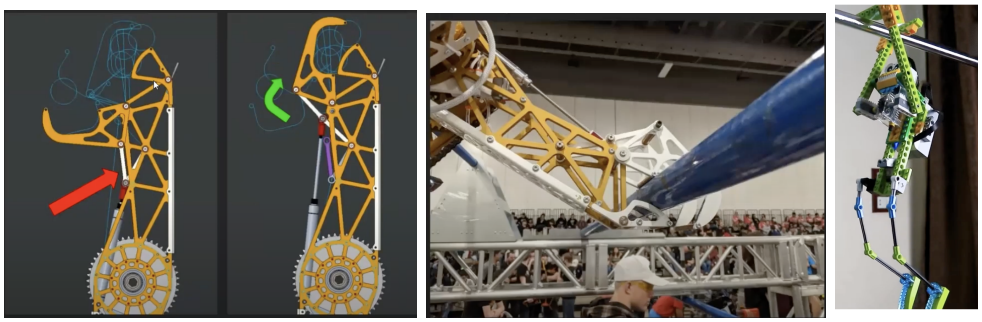


Figure 9: 1st version of the claw

The image above shows our first claw design, taken from a YouTube video by an FRC robotics team [8]. Our original concept was to have two servo-driven claws that extend from the sides and alternately "walk" up the pole, just like the right-most illustration, reminiscent of a LEGO monkey climbing a tree. Thus, the robot could propel itself forward.

But we ultimately ruled out this idea. First, it requires two servos to drive and has a bulky structure that consumes a lot of material, adding a significant portion to the robot's weight. Second, the claw in the video doesn't actually move. If we tried to make it climb like a monkey, it probably wouldn't be very secure. Even if it grips without slipping, we'd still have to keep the robot's centre of mass directly under the pole; otherwise, it would tilt and get jammed at that spot.

Secondly, after taking on-site measurements, we found that the central trench isn't actually very long. Because our wheels are of very large diameter, even if the robot is suspended right in the middle of the trench, the wheels can still reach the edges. That makes it simple: as long as there's a contact point, there's a force point, so we can drive the robot across by wheel rotation. In this case, we need a structure that extends upward to first reach the pole, then

hook onto the pole with a hanger, and finally use speed and the robot's body to pass through quickly.

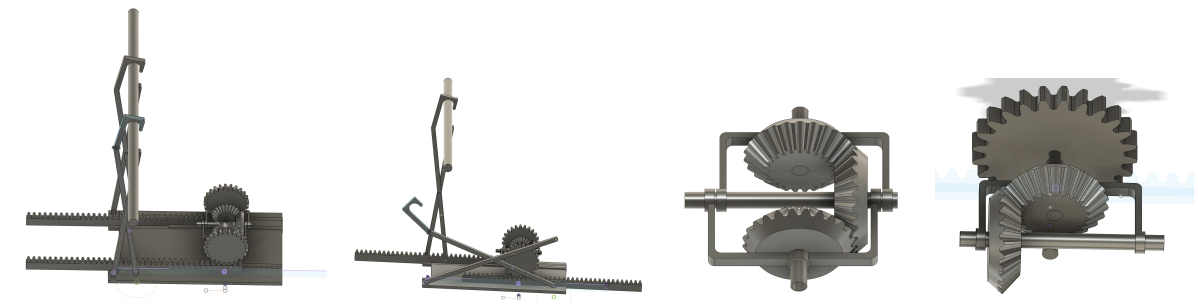


Figure 10: Lifting Mechanism – 2nd version

This is the second version of the CAD Model. A motor or a servo drives the whole mechanism: the motor/servo turns a bevel gear, which in turn drives two adjacent 90-degree bevel gears. The shafts of those bevel gears connect to the shafts of two spur gears, and the spur gears' rotation drives the lateral movement of two racks. On each side, there are two linkages – one linkage's lower end is fixed to the base, and the other's lower end is fixed to its rack. In this way, the rack's sideways motion causes the two linkage assemblies to act as a lift mechanism, which could be shown in the above diagram for the detailed layout.

Additionally, the other ends of the linkages are the claws, so at any given moment, there are four claws on the pole. In this case, even if one or two fail to grip, the robot won't fall. There's one more detail not shown in the CAD model: the bevel gears rotate in opposite directions. During assembly, we'll add an extra gear next to or above the gear and mesh it with that bevel gear. This one will not change speed via gear ratio since it only reverses the rotational direction of the original gear.

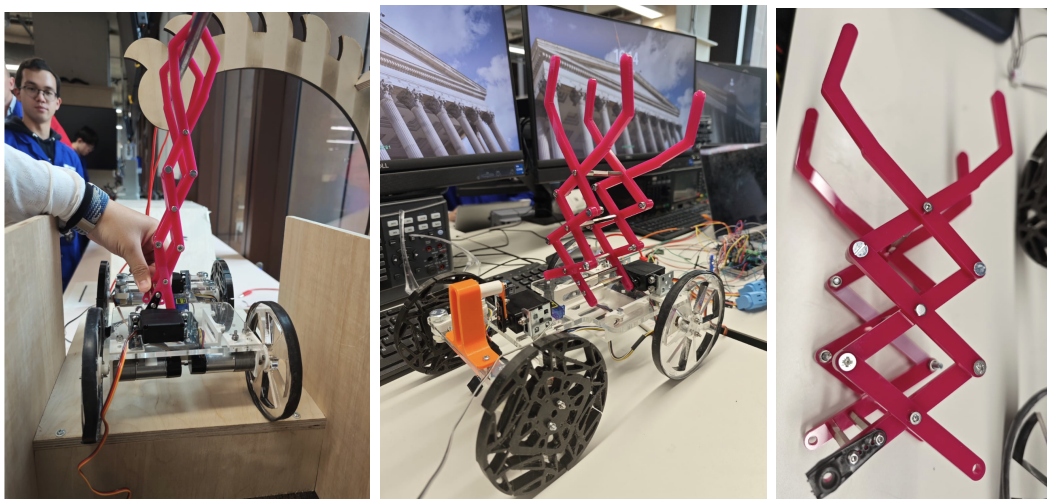
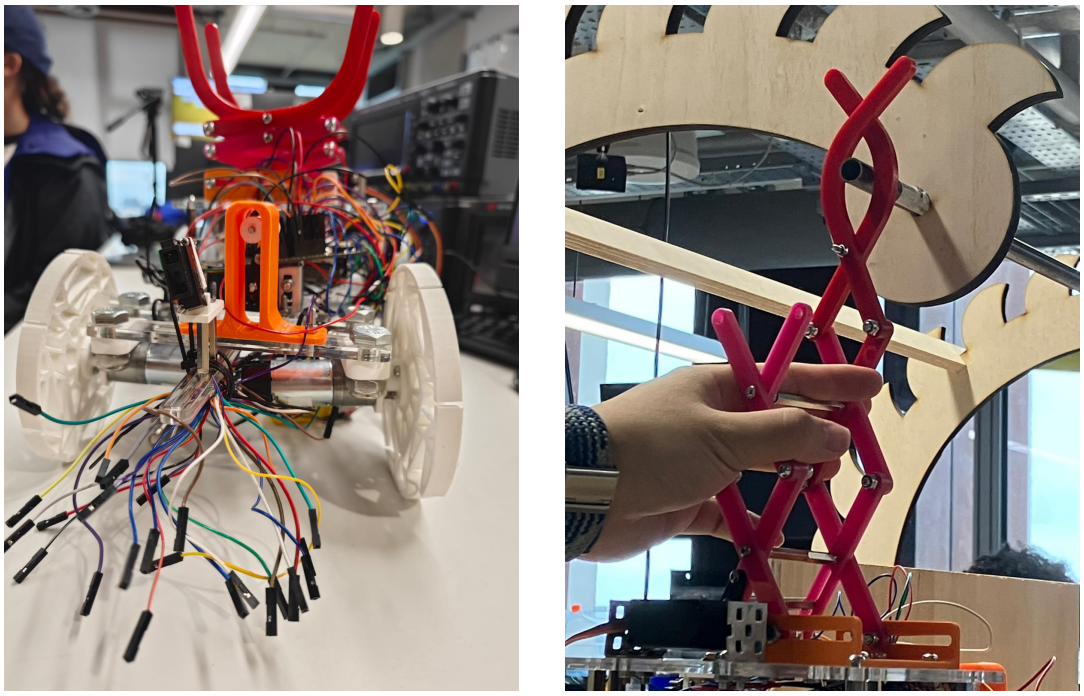


Figure 11: Lifting Mechanism – 3rd version

This is the third version of the lifting mechanism. In this iteration, we use a scissor-style lift: by stacking an odd number of stages, the top two linkages retract as they rise, performing a gripping action. The lower ends of the linkages mirror the second version's design – one end fixed, the other translating laterally – and ride on a guide rail that limits their movements and provides two supporting points. Notably, one of the fixed pivots is mounted to a servo, enabling the vertical motion, and both ends are secured with screws so that a single motor can drive the entire assembly.

Our fourth and final iteration of the lifting mechanism largely resembled the gripper in the tried iteration, however, we altered the the gripping linkages to be arc shaped which results in a tighter fit around the pole, therefore giving a more secure grip, allowing the robot to make it past the lava pit obstacle with ease. Another key difference was that this gripper was mounted higher up, on an acrylic sheet layer above the chassis, in order to accommodate space for the perforated board, electronics and battery slot, which we moved to the top as it was initially too close to the motors which would have introduced a fire risk due to the motors heating up.



(a) Fully assembled gripper (b) Testing the curved linkages

Figure 12: 4th iteration of Paxton's gripper

Passively-Transformable Single-Part Wheel

After evaluating and discarding both the rocker-bogie and tracked chassis configurations due to their respective limitations in maneuverability and adaptability, the improved performance of the rocker-bogie system led us to delve further into passive stair climbing mechanisms. We believed that the requirement of an active exertion of an input force to actuators would aid us in making the robot as simple as possible. Additionally, the lack of such an active mechanism would significantly reduce the weight of the robot, as this also omits the requirement of any electronic components involved, which would be particularly useful for the lava pit section and the final escape zipline, cases in which the robot will be suspended mid-air.

Upon meeting **Dr. Angus B. Clark** at **Imperial College London**, we were introduced to the **PaTS-Wheel, or Passively-Transformable Single-part Wheel**, which is an innovative wheel, by researchers **Thomas Godden, Barry W. Mulvey, Ellen Redgrave, and Professor Thrishantha Nanayakkara** at Imperial College London's Morph Lab. The wheel, arguably Paxton's main highlight, cleverly combines the efficiency of traditional wheels with the obstacle-negotiating capabilities of legged systems, which make use of "whegs" or "wheel-legs" without the need for active control systems. Its structure allows passive transformation in response to uneven surfaces, and additionally provide significant leverage when climbing stairs.

The core structure of the PaTS-Wheel is composed of **repeating flexible segments**, which are patterned around the wheel's circumference. As detailed in Godden et al. [3], each segment consists of a **"claw"** and **"pad"**. When the wheel comes across terrain with sudden elevation changes, these segments deform passively under load. Therefore, as the pad presses against the vertical surface of a step, the claw hooks onto the top, enabling the wheel to transition between rolling and climbing without requiring active control. The wheel has also been designed with **bi-directional symmetry**, allowing it to function even in reverse.

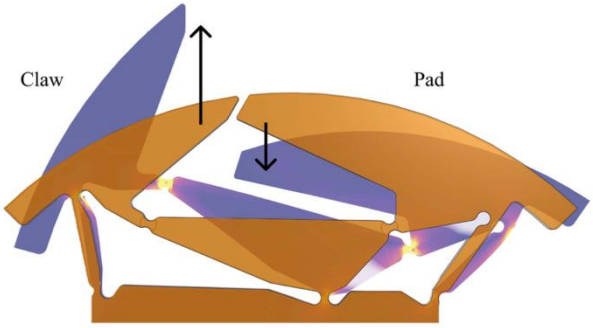
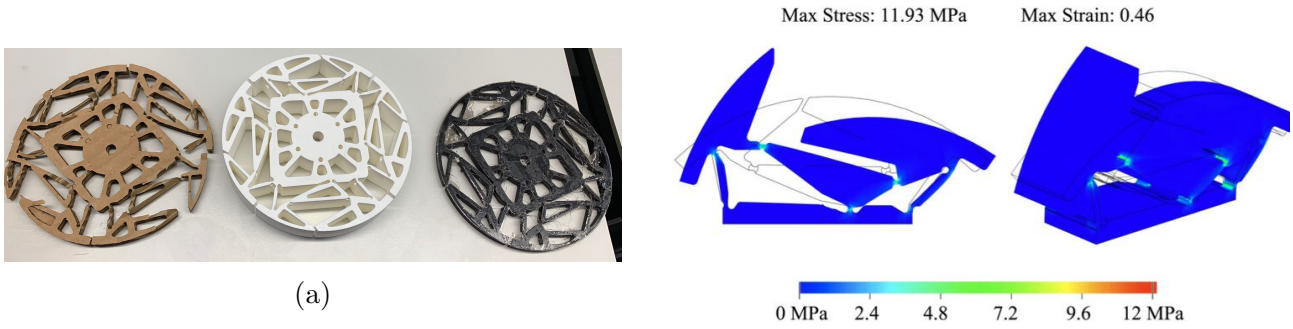


Figure 13: An individual claw-pad linkage section, with motion depicted once it has reached a step. [3]

In the initial prototyping phase of the wheel, a laser-cut cardboard model was developed. Cardboard provided sufficient flexibility for the linkages to articulate as intended, and initial tests on the obstacle course confirmed its suitability for functional evaluation. However, due to its limited structural integrity, alternative materials were investigated for subsequent iterations. Acrylic and PLA were selected for further testing; however, both proved unsuitable. Acrylic's rigidity and the limitations of laser cutting prevented the integration of functional joints. PLA, on the other hand, allowed joint movement only when printed with no infill, which significantly compromised mechanical strength and led to frequent failures under load. Ultimately, thermoplastic polyurethane (TPU) was adopted, aligning with previous research in the field. TPU offered an optimal balance between flexibility and structural resilience, enabling reliable performance across varied obstacle surfaces.



(a) (b)

Figure 14: (a)-prototypes of the PaTS wheel (b) - FEA of a single claw/pad mechanism (from research paper)

Wheel Kinematics (from the research paper)

The relationship between the deformation of the pad and the extension of the claw would be given as follows:

$$c_y \approx -c_{p1} + c_{p2}p_y - c_{p3}p_y^2 + c_{p4}p_{rt} - c_{p5}p_y p_{rt} \tag{3}$$

Summary

In addition, we have designed flanges to fasten the wheels to the motor with screws. We

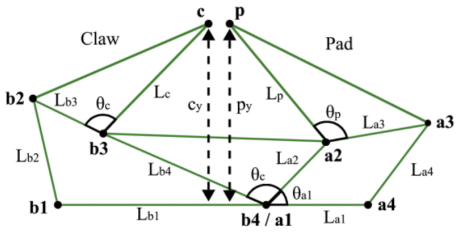
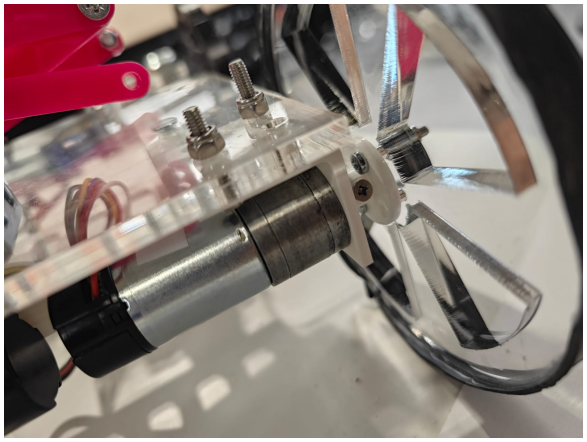
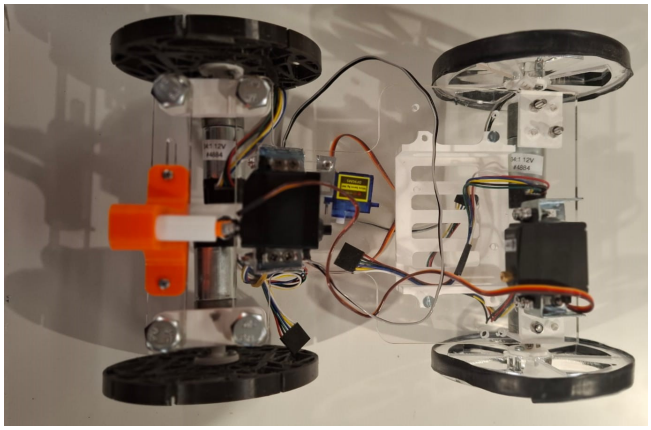


Figure 15

added a battery slot to store the battery safely. It is worth noting that around the battery compartment, we made a safety treatment to prevent any combustible or puncture objects.



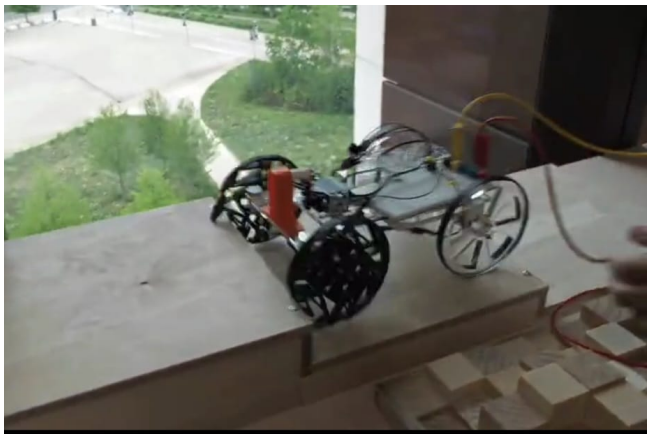
(a) flanges to connect the wheels to the motors



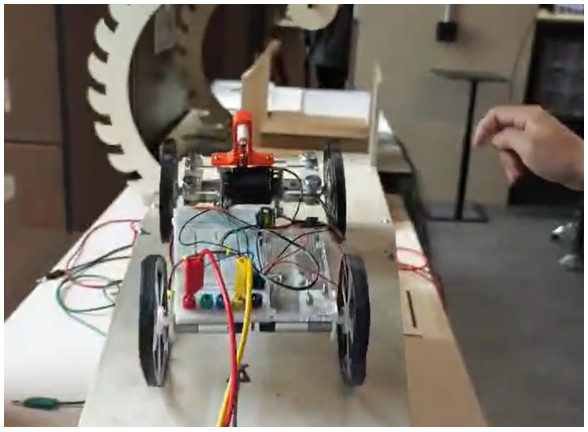
(b) Robot chassis with the battery slot

Figure 16: Ackermann steering structure

After completing the construction of the whole robot, we tested it on the field, and the final result was satisfactory. The robot's TPU-printed wheels can pass through obstacles and stairs well, and the steering structure can also adapt to the black lines on the ground.



(a) Paxton going up the stairs with the PaTS wheels



(b) Paxton going up the ramp

1.4 Bill of Mechanical Components

Mechanical Item	Quantity
TPU PaTS wheels	2
Acrylic back wheels	2
Custom PLA servo mount (steering)	1
Acrylic gripper linkages	6
PLA gripper slider	1
Vex Robotics aluminium frames	4
Screws (M2, M3, M4, M10)	20+
Acrylic chassis	1
PLA wheel flanges	4
PLA reflectance sensor mount	1

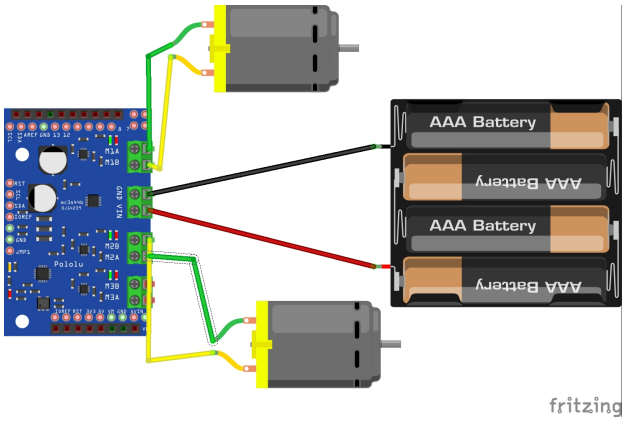
Table 2: Mechanical component list

2 Electrical design

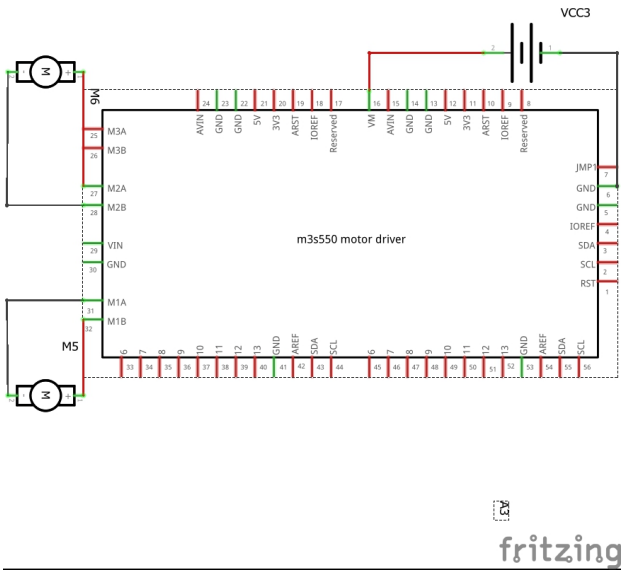
The challenge required using very specific components and often more than one of the same components. The full list of components can be found in Section 2.2.1.

The MCU we are used was an Arduino Giga R1 because we needed the extensive digital pins that the Giga provides. Each subsystem is connected to the MCU but has no direct connection to other subsystems. The subsystem diagrams are presented in isolation with the MCU to better understand how they connect. The only difference is that all grounds and 5V go to their respective row of headers on the PCB, which are then connected to the MCU.

The first subsystem is the dual motor shield connected to the Giga, these motor shields are each connected to two independent motors, however the encoders on the motors are not in use. Each motor shield has the same subsystem, Figure 18, and stacks on top of each other on top of the Arduino.



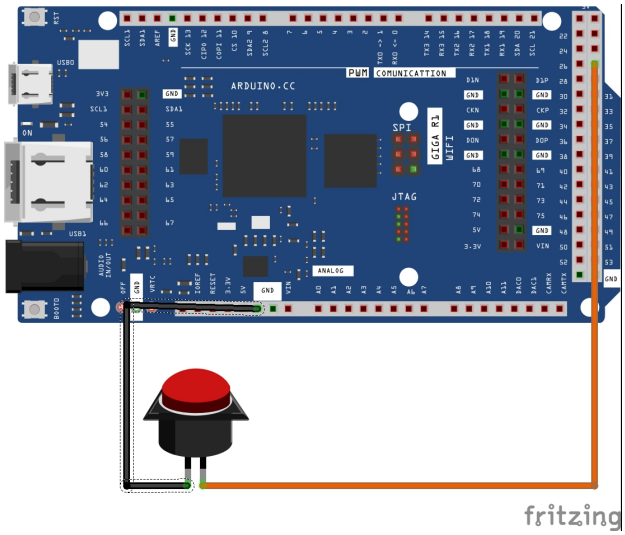
(a) Diagram of motor subsystem



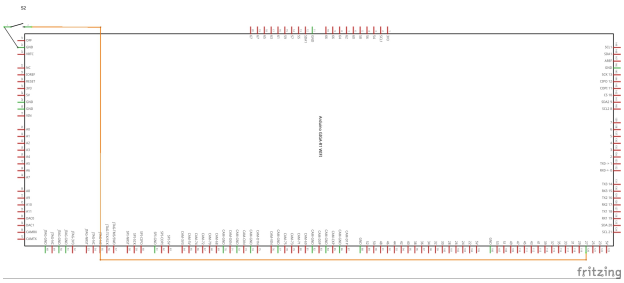
(b) Schematic diagram of motor subsystem

Figure 18: Schematic diagrams of a motor shield subsytem. There are two of them stacked on each other

The second subsystem is the mechanical kill switch, for this we used a button which directly connects to a digital pin and MCU ground, Figure 19.



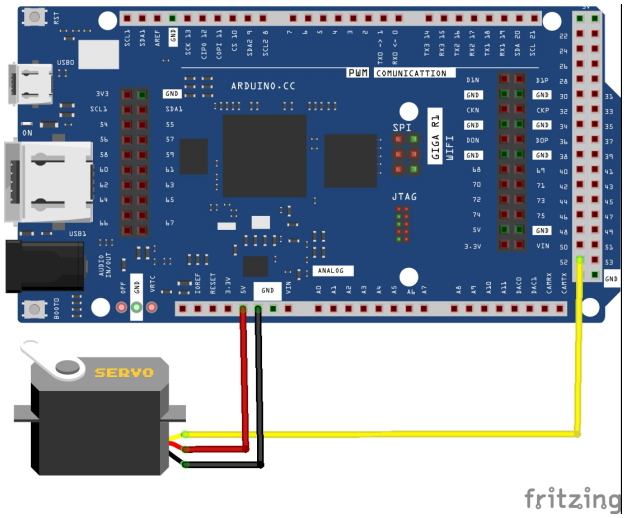
(a) Diagram of button subsystem



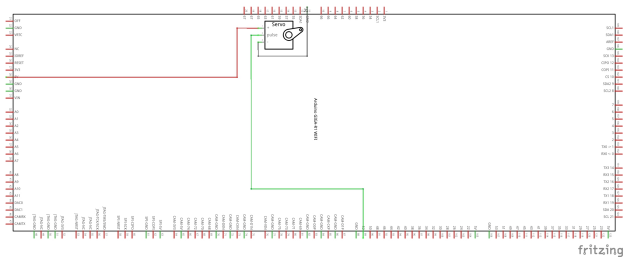
(b) Schematic diagram of button subsytem

Figure 19: Schematic diagrams of the button subsystem

The servo subsystem consisted of three servos, all connected independently of each other. The subsystem for one can be seen in Figure 20, all three servos follow the same design.



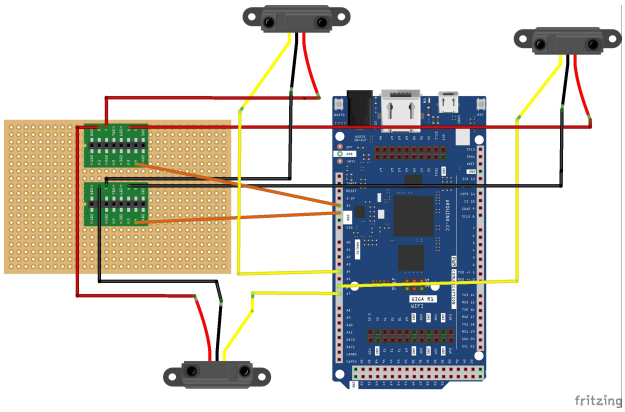
(a) Diagram of servo subsystem



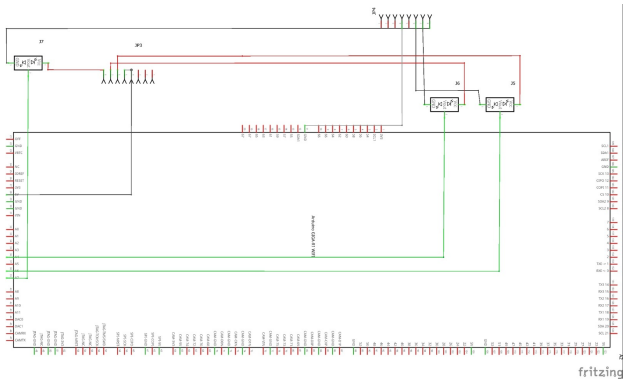
(b) Schematic diagram of servo subsystem

Figure 20: Schematic diagrams of servo subsystem. There will be three of these.

The IR-LED subsystem consisted of three IR-LED sensors that were all connected independently of each other. The connection of these three sensors can be seen in Figure 21.



(a) Diagram of IR-LED sensor subsystem



(b) Schematic diagram of IR-LED subsystem

Figure 21: Schematic diagrams of IR-LED distance sensor subsystem. More can be added following the template of the current three as each is independent of the others.

Finally, the reflectance sensor subsystem consists of one physical component that contains nine sensors that all connect independently to digital pins but share a ground and VCC. This can be seen in Figure 22.

2.1 Conceptual Design

2.1.1 Line detection

To detect the black lines, we decided to go with the reflectance sensor given to us, the QTR-HD-09RC, Figure 23. The nine sensors on the board must be connected to separate digital pins. The reason we stuck with the Arduino Giga R1 is that the smaller boards were limited in

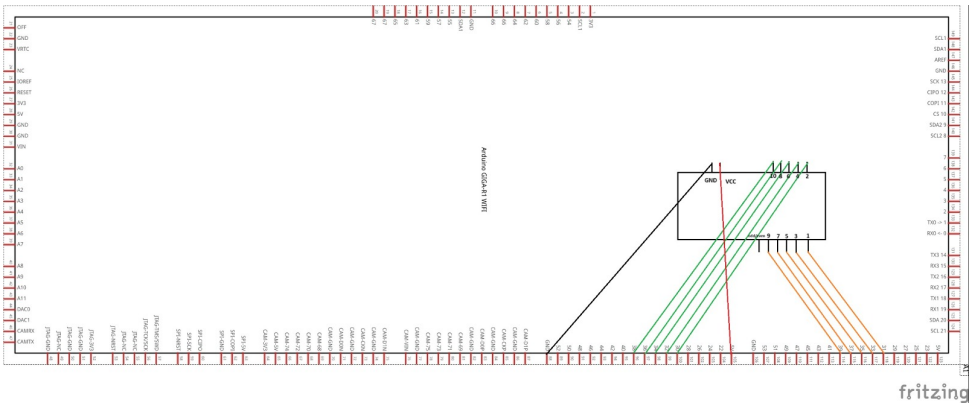


Figure 22: Schematic diagram of the reflectance sensor

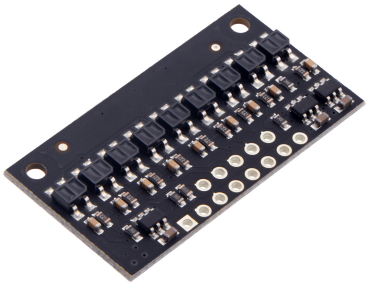
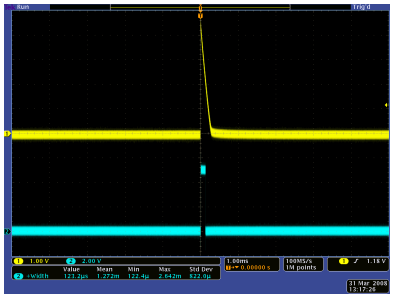


Figure 23: QTR-HD-09RC reflectance sensor [5]

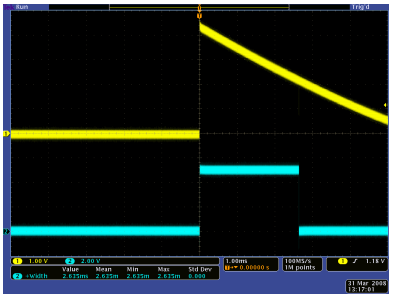
the number of digital pins they provided, allowing lower flexibility when we try to implement control algorithms and want to adjust the number of sensors.

To obtain data from the sensors, we first switch the pins to output, then send a high voltage to the pins. Wait ten microseconds, then turn the pins to input and time how long for each pin to reach a low voltage. From the data sheet [5] and the example RC readings, Figure **??**, we know that when the recorded time is low, the surface is reflective (white), Figure 24a, and when the time takes longer to decay at the same physical height, the surface is darker (black), Figure 24b, this . This data is stored in an array and can be passed to control algorithms. Each sensor requires a unique I/O pin to communicate with the MCU.

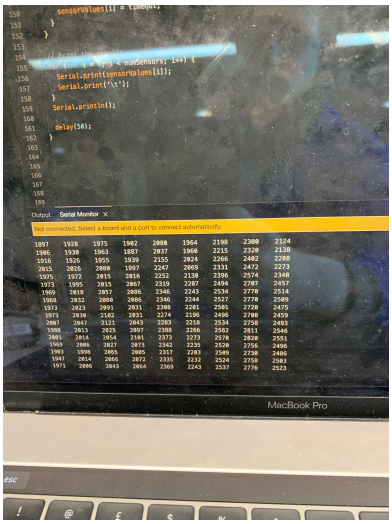
The reflectance sensor has a maximum sensing distance of 40mm, however the chassis was too high, so a lever was designed to lower and raise the sensor when needed. An example reading can be seen in in Figure 23(c).



(a) Digital Pin output from 1/8" from white surface



(b) Digital Pin output from 1/8" from black surface



(c) Readings from the nine sensors on the reflectance sensor

Figure 24: Reflectance sensor outputs: digital signals from white and black surfaces, and sensor array readings [5]

To do line following, the robot must first be able to detect the walls. This would be done by detecting the distance to the nearest object, which should be the wall or an obstacle. We decided to use three IR-LED distance sensors (GP2Y0A51SK0F), Figure **??**, to perform this task. The range of the sensor is between 2 and 15cm. We think this will be acceptable at the lower end because we would not want the robot to be closer than 2cm to the walls at any given moment, as turning could become more challenging. 15cm as the upper bound should be sufficient, as that is more than half of the length of the robot along its longest axis, allowing plenty of time and space to perform a correct action dictated by the control logic.

The output produced by the sensor is analogue, providing greater accuracy without relying on the Arduino's internal clock counter. The resolution of the reading is 12 bits. To get the reading, Figure 25, it sends an infrared signal and times how long until the light returns after bouncing against an object. Upon returning to the sensor, the IR light passes through a photodiode, which releases a voltage to the MCU. This process happens continuously while power is being supplied through the VCC pin; the diagram can be seen in Figure 25.

2.1.2 Motors

Our design requires 4 DC motors and 3 servo motors. We wanted to use servo motors instead of dc motors in the steering, the reflectance sensor drop down and the gripper mechanisms because we wanted them to have a limited range of movement otherwise they could break the whole robot with undesirable movement ranges, additionally they will draw less Amps from the microcontroller resulting in a safer robot.

The use of 4 DC motors forced us to utilise two motor shields (Pololu M3S550), Figure 26. To have all four motors be independent of each other, the address of one of the motor shields had to be changed from the base 16, which they are automatically set to. To change the I2C address of one of the motor shields, we followed the guide on the pololu website [4], the method involved purposefully creating a short-circuit between the JMP1 and ground pins on one motor

■Timing Chart

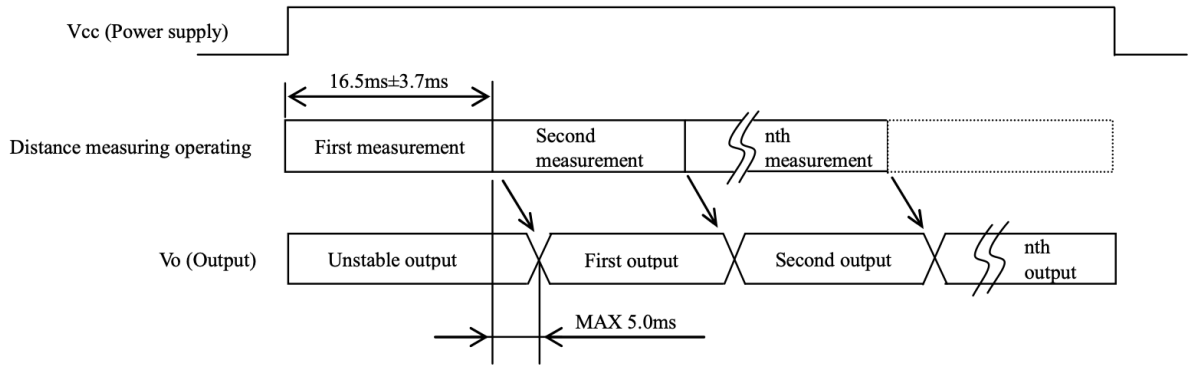


Figure 25: The timing chart of the GP2Y0A51SK0F distance sensor [9]

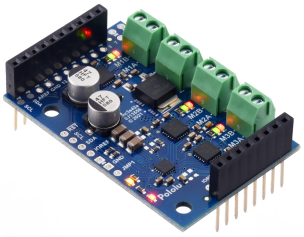


Figure 26: Pololu M3S550 motor shield [1]

shield and then with the serial monitor send the message "a17" to set that motor shield's I2C address to 17. These Motor Shields both communicate with the MCU using the I2C protocol. The diagrams for each motor shield can be seen in Figure 18.

The three servo motors were needed for specific actions outlined earlier in the report: lava pit, steering, and reflectance sensor. These mechanisms required specific constraints that guided the choice of the servo. The lever is powered by a microservo because the movement range is not too high, and the torque required is low. This does not add much mass to the robot. The other two servos required considerable torque as one holds the robot (lava pit) and one controls the steering. This forced us to go with the high torque servo for both, which does come with more mass and more power required, but is necessary. The subsystems for all three are the same, and the diagram for one can be seen in Figure 20

2.1.3 Mechanical Kill Switch

The mechanical kill switch is controlled by a button. When the button is pressed, the robot is required to completely shut down its motors immediately, this is why we needed a debounce feature. The debounce creates a timer that does not allow the Arduino to register the button being pressed again for a certain duration after the initial press. The button communicates through one I/O pin.

2.2 Practical Implementation

2.2.1 Electrical components

The full list of electrical components used:

Name	ID	Quantity
Reflectance sensor 4×9	QTR-HD-09RC	1
IR-LED distance sensor	GP2Y0A51SK0F	3
Motor shield	M3S550	2
Arduino Giga	Arduino Giga R1	1
Button	–	1
Motor	HB7545GS	4
Feetech Microservo	FS90	1
High torque servo	900-00008	2
Perforated board	–	1

Table 3: Component list for the project

2.2.2 Manufacturing

Firstly, every component was tested individually to confirm that it worked, and we could get reproducible readings from it. Once we confirmed all subsystems worked as intended, we built the system up with a breadboard. Once that all worked to our satisfaction, we soldered all the connections onto a perforated board. The complete system can be seen in Figure 27. The whole circuit worked as intended for the trial run, the only problem was a software issue with the Wi-Fi where it could not detect the stop message although it was connected to the Wi-Fi.

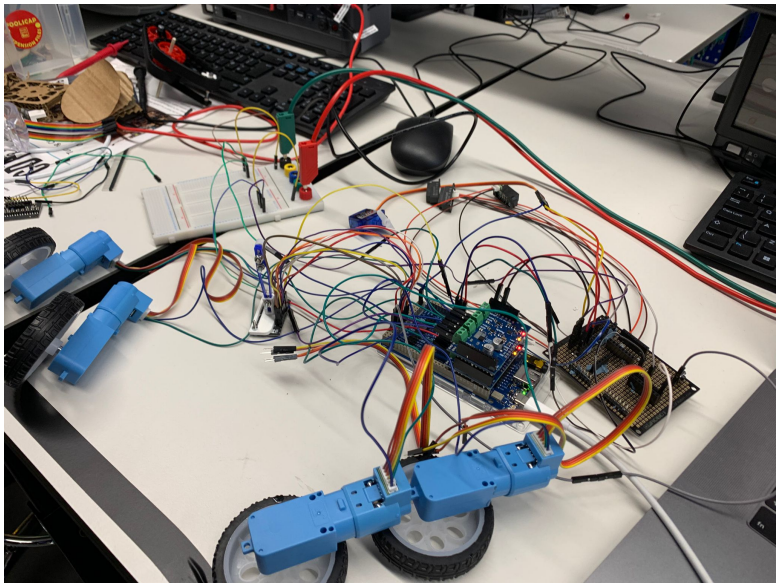


Figure 27: The complete electrical system with soldering. The only change is the number of servos, and the motors will be different as well, but all connections will remain the same.

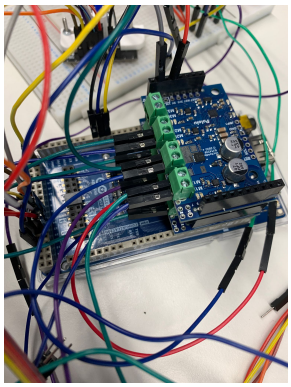


Figure 28: All the connections going into the Arduino from the components and the perforated board.

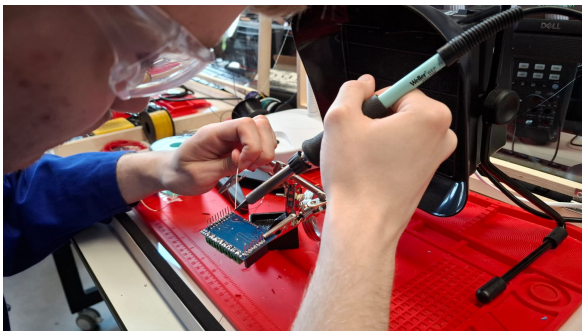


Figure 29: The headers and terminals had to be soldered onto both motor shields.

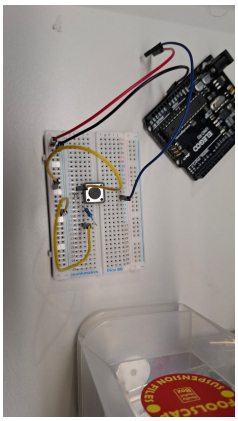


Figure 30: The breadboard working prototype of the button kill switch subsystem.

3 Programming

A GitHuB with the code can be found at: MightyBrushwagg/RoboticsChallenge
Alternative repository link: ItzSmudge/RoboticsChallenge

3.1 Algorithms for Line Following

The line tracking algorithm is designed to control the robot along a predefined black line using an array of 19 reflectance sensors. As illustrated in the below flowchart, the robot continuously reads sensor data determines its current mode based on the readings, and adjusts steering accordingly through a PID control loop.

At the beginning of each cycle, the robot obtains the reflectance sensor reading. Each sensor returns a discharge time value, which is binarized based on a calibrated threshold to indicate whether the sensor is detecting a blck or white surface. These binary values are used to determine the mode of operation:

- if all sensor return black, such as black Count equals the number of sensors, the robot considers this the end of the track and switches to the `MODE_DEAD`, bringing the motors to a stop.
- if all sensor return white, it may indicate a dashed or missing segment. In this case, the robot moves lightly forward and rechecks. If still all white, it transitions into wall following mode.
- If the readings match the pattern of a Y shaped fork, the robot enters `MODE_CROSSROAD_TAKING` and turns based on a predefined direction array.
- Otherwise, the robot remains in `MODE_LINE_FOLLOWING`.

In `MODE_LINE_FOLLOWING`, the robot computes a weighted centroid from the active sensors, which represents the error from the center of the track. This error is then processed by a PID controller to calculate a correction value that adjusts the servo angle of the front steering wheels, as well as dynamically modulates the motor speeds to smoothen tight turns.

This logic ensures the robot maintains a centered position on the line while being able to handle forks, gaps, and terminal marters. The decision making process is implemented in the `check_crossroad()` and `auto_tracking()` functions in the source code, and is executed continuously inside the `loop()` function.

Key Functions:

- `Reflectance_Sensor_Reading()`: Reads and binarizes the sensor data.
- `check_crossroad()`: Determines the current mode based on sensor patterns.
- `auto_tracking()`: Executes PID-based steering and speed control.

This algorithm proved highly effective under most lighting and surface conditions, with responsive steering and reliable mode switching.

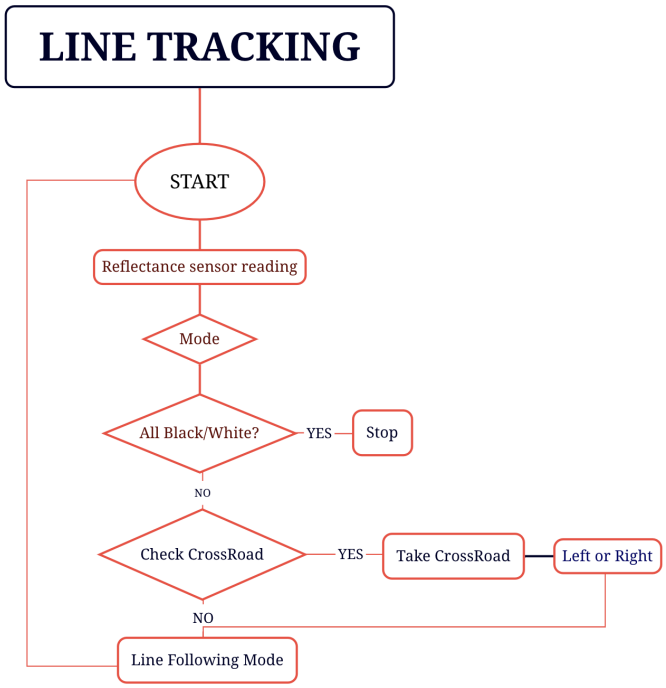


Figure 31: Flowchart for line tracking

3.2 Algorithms for Wall Following

The wall following is implemented to allow the robot to navigate along a wall located on its left hand side using two distance sensors: one facing left and one facing forward. The flow of the algorithm is illustrated in the corresponding flowchart below.

At the beginning of each cycle, the robot measures the distance to the left wall using the side facing sensor. If the distance is too far beyond the ideal threshold, the robot initiates a correction by steering left to maintain the desired offset. If the left distance is within an

acceptable margin, the robot proceeds to check the distance in front using the front facing sensor.

If an obstacle is detected ahead, such as the front distance is below a critical threshold, the robot executes a 90 degree turn to avoid collision and prepare to follow the next wall segment. If no obstacle is present, the robot continues driving foward while constantly adjusting its lateral position relative to the wall.

Key Points:

- Maintain a consistent distance from the left wall using a proportional controller (error = ideal distance - actual distance)
- Perform a right angle turn when the front obstacle is detected
- Continuously drive forward when clear, while maintaining parallel to the wall

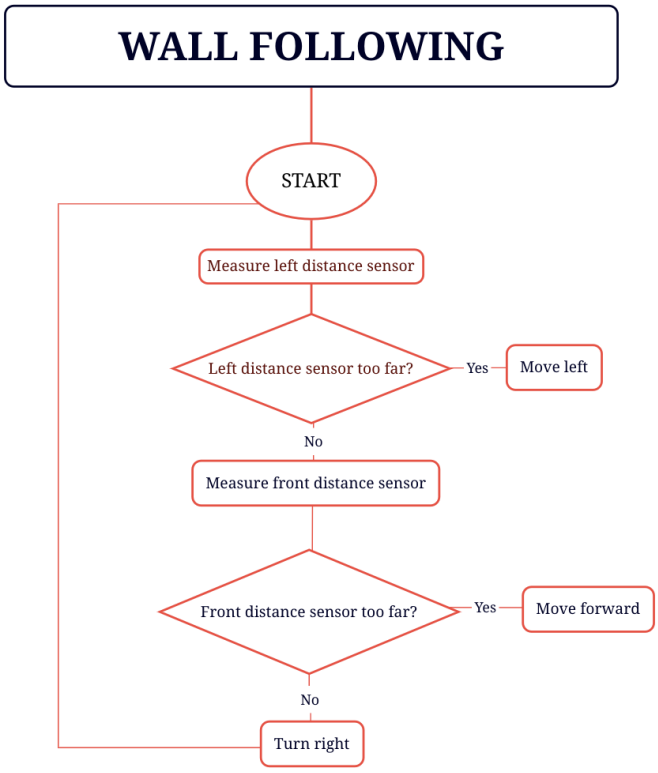


Figure 32: Flowchart for wall following

3.3 Algorithms for Lifting Mechanism

The lifting mechanism is activated when both front facing distance sensors detect that the robot has reached a predefined target distance.

At first, the robot moves forward at a reduced speed to ensure precise alignment with the target object. After covering a short distance, the gripper is raised using a servo to prepare for object capture or lifting. Then, the robot accelerated forward at full speed, completing the gripping action. Once the sprint is completed, the gripper is lowered. The robot transitions back to its standard line following mode and continues navigation.

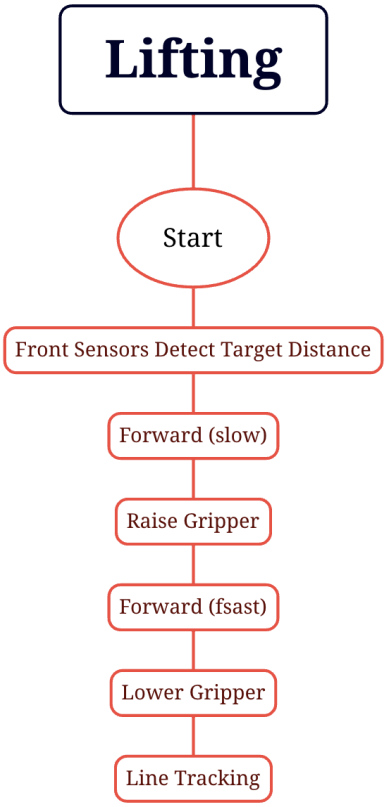


Figure 33: Flowchart for lifting

3.4 Algorithm Performance Evaluation and Reflections

In this section, we assess the performance of these algorithms in our robot control. We discuss which algorithms behaved as expected, which did not, and propose possible improvements during testing.

3.4.1 Line Following Algorithm

The line following algorithm, driven by reflectance sensors and a front-servo-based PID control system, performed reliably throughout our tests. It successfully detected and followed a single black line by calculating a weighted centroid and using proportional control for steering.

Additionally, the robot correctly identified the following conditions:

- **All-white and all-black sensor states**, which triggered transitions into alternative behaviours (stop).
- **Y-shaped forks (crossroads)**, where a hardcoded path selection strategy (e.g., turning left or right) was applied effectively.

Improvements:

- The current PID controller uses only the K_p gain. Incorporating K_i and K_d terms could enhance stability, especially on sharper curves.

- The 90-degree turn logic was commented out. This could be resolved by introducing a temporary state with turn-specific conditions.
- Currently, it is not possible to switch from line tracking to wall following. More robust trigger conditions may be needed to enable this transition.

3.4.2 Wall Following Algorithm

The wall following routine utilized left and front distance sensors to maintain a constant offset from the left wall while detecting obstacles ahead. In fact, we didn't fully debug the wall following program during the competition, but we were able to anticipate some potential issues, which we can keep in mind for similar competitions in the future

- **Sensor Noise**, Due to sensor reading noise or uneven wall surfaces, the robot may perform sudden corrections, which can occasionally cause it to deviate from its intended path.
- **90 Degree Turn**, Executing a hardcoded turn immediately after meeting the turning condition can potentially lead to downstream path issues. Even a slight angular deviation at the starting point of the turn can significantly affect the robot's final position, potentially preventing the left side sensors from aligning parallel to the wall after the turn.

Improvements:

- Applying Kalman filter to the distance sensor readings before further processing can improve accuracy and reduce noise.
- Replacing the current hardcoded turning logic with angle tracking using encoders or an IMU cam enable precise 90 degrees turns, ensuring the robot consistently reaches the correct position after each turn.

3.4.3 Lifting Algorithm

The lifting procedure was designed to trigger when both front sensor detected a target object. After that, gripper performs the grasping action while the lifting mechanism rises, then locks onto the pole. This allows the robot to hang from the pole.

Challenges:

- The forward sprint sometimes caused the robot to overshoot the target area, falling down to the gap.
- It is difficult for the sensors to reliably trigger the activation of the lifting mechanism

Improvements:

- Use encoder calibrate the estimated distance, instead of relying on fixed delays to control the forward sprint.
- Introduce a brief alignment pause after lifting the gripper to improve positioning accuracy.
- use other conditions, such as camera, for helping triggering the activation of the lifting mechanism instead of only using two distance sensors at the front of the robot. Camera enables recognize any objects and then we can do any actions that we want based on this.

4 Discussion and Conclusion

4.1 Biggest Takeaways

Takeaways: From A Team Perspective As a team, we cannot stress enough the fact of how important proper planning and management area. Dividing the workflow, from the start of week 1 itself, it was a significantly more robust approach towards project completion in comparison to the previous term projects. Having dedicated time towards organizing a project dashboard and a mind map in the beginning meant that we were clear on what we had to achieve by the end of each week, rather than simply attending to tasks required whenever they seemed important. When it came to splitting tasks, we mostly got them done in pairs, with the confidence that at least one member of each pair was fully coherent in the domain of the task, be it mechanical, electronics, programming, or control.

Takeaways: From A Project Perspective Setbacks and unforced were expected but did not hinder our progress for the most part; our mechanical prototyping hardly came across any hurdles, and we were able to effectively decide on our approach to make the robot traverse the obstacles in sections 2 and 3 effectively. We think that it would be safe to assume that we made all the right decisions from a mechanical perspective, as despite having around three to four iterations for most of the components by week 2, the choices we made resulted in us having one of the oldest robots by week 6, which proved to be capable of traversing the physical obstacles exceptionally well. By the time of the preliminary trial runs, we had managed to get all electronic components to individually function as required, excluding the WiFi kill switch. Our success here provided us with the confidence that they would perform equally as good when they were all brought and operated together on the robot. However, when it came to implementing control for the robot, we were quite far from reaching the targets we had set. For line following, our steering mechanism seemed to be quite useful, but thinking about the bigger picture, pure differential drive would have resulted in smoother progress in PID tuning and navigating turns in the obstacle course. We regretted not having good cable management early on as debugging electrical problems took longer than they should have, which cost us some points for the second trial run and section 1 on the obstacle course. Having a short-circuit that rendered a reflectance sensor dysfunctional and loose wires falling apart while changing power sources were some instances of mishaps that could have been avoided. Regardless of the issues, we are glad to have experienced them early during our course, as these are clearly some points to be kept in mind for future projects.

4.2 What worked and what didn't

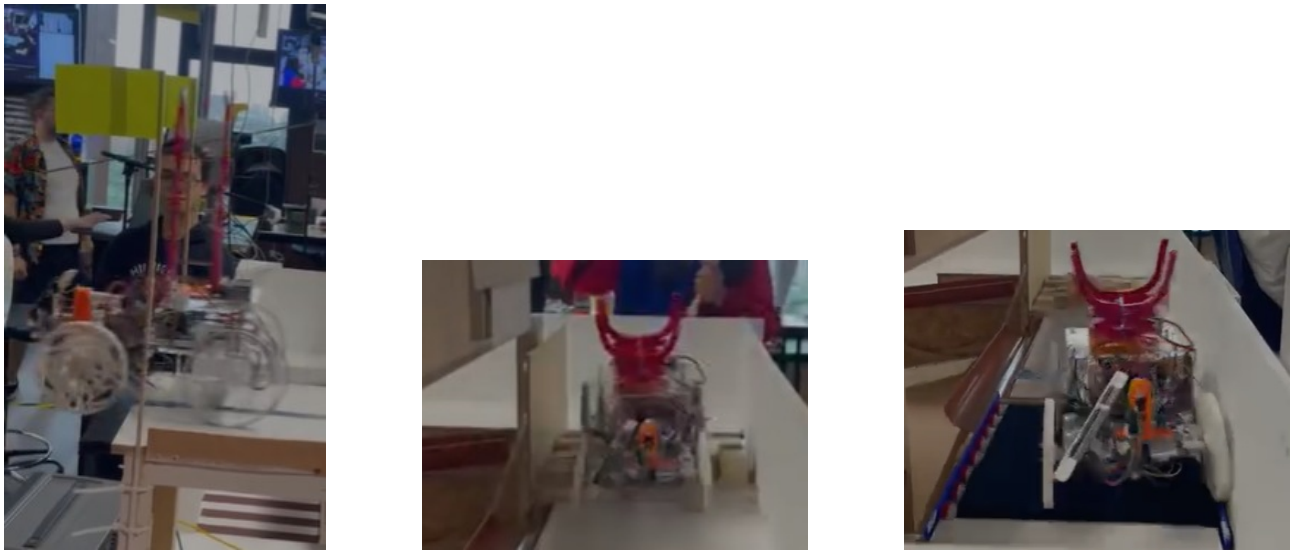
4.2.1 Mechanical

What worked: We were really pleased with all of Paxton's mechanical aspects. Having successfully cleared the stairs, Giant's Courseway, ramp and the sharp turns in the trial runs, Paxton was able to repeat these performances on the day of the Challenge, being among the quickest to traverse the obstacles in sections 2 and 3. The revised gripper design also allowed Paxton to become the **very first robot** to clear the **lava pit** and **zipline**, both of which were done successfully with the robot being able to land on the finish line upright and unbroken.

What did not work: We went with a single chassis for the entirety of the project after finalizing on the movement method, and since it was made of acrylic, we lost the flexibility of moving components and sensors around on the chassis which hindered our cable management and overall organization.

4.2.2 Electronic & Control

What worked: The main success is that the robot moved and could follow the line. This meant that the reflectance sensors were working, the dual motor shield control worked as intended, and the servo controlling the Ackerman was smooth and accurate. All sensors were able to read and got meaningful consistent data. We were able to fully power the robot from the battery with nothing going wrong and the kill switches worked. The robot correctly followed the line and took the correct crossroads. The motors were given the correct powers, enabling traversal over all obstacles.



(a) Paxton on the zipline

(b) Clearing the Giants' Courseway

(c) Clearing the treadmill

Figure 34: Performance of the robot on key obstacles: the zipline(a), Giants' Courseway(b), and the treadmill(c)

What did not work: On demonstration day and the days before, we ran into some problems, the main problem being that the cable management should have been better and thought of from the start as debugging faulty readings became impossible at times. In addition, our reflectance sensors all died the day before the demonstration and we had to replace them. During the trial runs, the button kill switch worked perfectly, however during the demonstration, the button stopped working, forcing us to rely on the WiFi kill switch. Because of poor time management and oversight, we did not get to test the wall following to a high enough standard, resulting in that not working as intended. During the demonstration, the robot would follow the line and then randomly stop, and we do not know why as that bug never occurred during testing and we could not recreate it before a rerun. Another issue we encountered were that the wires that connect to the IR-distance sensors were too flimsy and would snap off frequently from the PCB board.

4.3 Fascinating Discoveries

We were absolutely delighted with our decision to integrate the PaTS wheel into our project. It brought immense terrain adaptability through simple mechanical means through passive motion. This idea came to light during week 1 through a combination of an interest in soft robotics and reaching out to academics for suggestions. Being the only group to make use of both differential drive and an Ackermann steering mechanism, we were able to make the robot navigate sharp turns which we initially thought was not possible with the turning radius of the mechanism. An important we noted about control is that having simulations of tasks such as centering, wall and line following that work perfectly does not at all guarantee that it will work well with the actual system, which we came to realize. The PID controller, parameters and code we used for the robot had significant differences from the ideal PID controller in theory.

References

[1] Motoron m3s256 triple motor controller shield for arduino (connectors soldered), 2025. Date Accessed: 2025-05-05.

[2] Dick L Gebhart, Thomas A Hale, and Kim Michaels-Busch. Dust control material performance on unsurfaced roadways and tank trails. 1996.

[3] Thomas Godden, Barry W. Mulvey, Ellen Redgrave, and Thrishantha Nanayakkara. Patswheel: A passively-transformable single-part wheel for mobile robot navigation on unstructured terrain. *IEEE Robotics and Automation Letters*, 9(6):5512–5519, 2024.

[4] Pololu. Pololu - 3.5. setting i²c addresses with an arduino, 2025. Date Accessed: 2025-05-15.

[5] Pololu. Qtr-hd-09rc reflectance sensor array: 9-channel, 4mm pitch, rc output, 2025. Date Accessed: 2025-05-02.

[6] Douglas V Prose and Howard G Wilshire. The lasting effects of tank maneuvers on desert soils and intershrub flora. Technical report, US Geological Survey, 2000.

[7] Eunho Seo, Jooyoung Chun, Hunkeon Ko, Sangin Park, and Dong Jin Hyun. Study of rocker-bogie typed mobile robot for driving, climbing and standing upright. *IEEE Robotics and Automation Letters*, 2021.

[8] Shagoof. Wedo 2 0 instructions + code climbing monkey ii lego education, 2021. Date Accessed: 2025-04-29.

[9] SHARP. *GP2Y0A51SK0F*, N/A. Accessed: 2025-05-13.

[10] Jing-Shan Zhao, Xiang Liu, Zhi-Jing Feng, and Jian S Dai. Design of an ackermann-type steering mechanism. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 227(11):2549–2562, 2013.