MANCHESTER 1824
The University of Manchester

Department of Electrical and Electronic Engineering

Embedded Systems Project 2023-24

Title: Final Report

Group Number: 38

Group members:	ID Number	I confirm that this is the group's own work.
Oh Jie Han	11358882	☒
Jiaming Shang	11317421	☒
Achyuth Nishant	11444769	☒
Naman Goyal	11317658	☒
Hassen Al-Rubei	11392694	☒

Tutor: Andrew Forsyth

Date: 08/05/2025

Contents

1. Executive Summary & Introduction

This report displays the development, challenges and outcomes of our line-following buggy. The primary objective was to design a buggy capable of accurate line-following, incline climbing up to 20 degrees, reliable 180-degree point turns, and stable operation at high speeds. Robust performance in varying levels of lighting and surface conditions was a key requirement, as well as modularity for ease of maintenance and future development.

A compact, lightweight chassis was developed with modular sensor mounting to improve manoeuvrability and adaptability for future upgrades. The mechanical design prioritized stability by distributing physical loads across the chassis, and ease of access to critical components for efficient debugging. The gearbox selection balanced speed and torque to ensure sufficient power for both straight line speed and ramp climbing. Attention was given to cable management and secure mounting of components to minimize vibrations and mechanical failures during testing.

An array of TCRT5000 IR sensors was mounted at the front of the buggy to detect the line. To address challenges with ambient light variability, a custom ambient light filtering algorithm was implemented. This system used multiplexed sampling with synchronised IR LED switching, allowing ambient readings to be subtracted from active IR readings, significantly improving detection accuracy. The sensor board utilized a ULN2003an Darlington Array to manage high-current LED switching without overloading the microcontroller.

A Proportional-Integral (PI) controller calculated steering corrections based on differential sensor readings, with integral clipping applied to maintain stability. The control system used the microcontroller’s Ticker API to synchronize sensor sampling, motor control calculations all at 750 Hz. This high update frequency improved the system’s responsiveness and stability. Bluetooth connectivity was added for remote control and real-time PID tuning during testing, reducing downtime and improving efficiency.

The team adopted a collaborative and flexible approach to development and troubleshooting, addressing both hardware and software challenges through effective communication and iterative testing.

These technical obstacles were addressed through effective teamwork, open communication, and flexible planning. The team maintained a detailed Gantt chart to monitor progress and adjust timelines, which enabled successful completion of the buggy’s core requirements despite unexpected setbacks.

The total cost of the project remained within anticipated limits, with R&D and replacement expenses totalling £52.80. Key expenses included PCB revisions, motor and microcontroller replacements, and materials for testing. While labour and software costs were not directly incurred due to the academic context, a commercial viability assessment projected a per-unit production cost of £548.21. This accounted for component costs, estimated assembly labour, and a contingency margin for rework. A recommended retail price of £1,000 per unit was established to ensure cost recovery and margin sustainability, aligning competitively with similar market products.

The buggy successfully met all three core deliverables, including precise line-following, incline climbing, and Bluetooth-controlled point turns. Despite some challenges during the heats (ending with a 50% completion rate), the system demonstrated reliable performance across multiple timed attempts.

1.2 Introduction

The development of autonomous vehicles has become an important area of research and education, offering opportunities to integrate principles of mechanical engineering, electronics, and control systems. This project aimed to design and construct a small-scale, autonomous line-following buggy capable of navigating a track marked by a white line, climbing inclines up to 20 degrees, and executing controlled 180-degree point turns via Bluetooth input.

The project required the integration of multiple engineering disciplines, including chassis design for stability and accessibility, sensor integration for reliable line detection, and software development for control algorithms and communication interfaces. Particular emphasis was placed on modularity, allowing components to be easily maintained or upgraded for future iterations.

This report details the design process, technical implementations, and key challenges encountered during the project. It also presents the system's performance during testing and competition heats, along with reflections on team collaboration, time management, and budgetary considerations. A diagram of the final buggy design is included below to provide an overview of the integrated system.

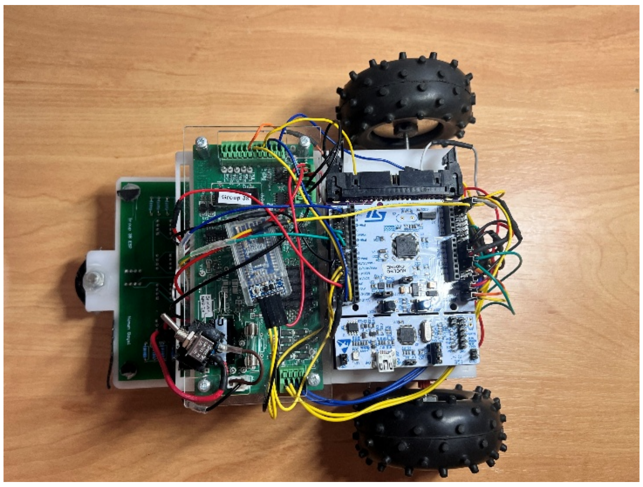


Figure 1: Top View of Completed Buggy

2. Final System Components Summary

2.1 Mechanical

The buggy is constructed with a 2-tier chassis design: the bottom layer houses the battery pack, while the top layer holds the motor drive board and the STM32 microcontroller. This layout balances the buggy's centre of mass and allows easy access to each component. Beneath the bottom chassis layer, the sensor board PCB is mounted using screws, playing a crucial role in detecting track lines for line-following.

The buggy's simple, compact design helps minimise collisions and enhances stability during movement. All wiring was cut and soldered to minimal lengths, with excess wires secured using zip ties to prevent entanglement with the wheels and motors. Additionally, the battery pack is attached to the bottom layer chassis using Velcro, preventing swaying and reducing risk of damage during collisions.

The two figures below show the physical buggy and its CAD model. While no major changes were made between the design and final build, a few minor adjustments were implemented. Specifically, the screw of the castor wheel was shortened to make battery pack removal easier. Additionally, the power LED on the sensor PCB was cut, as it was interfering with the light sensors.

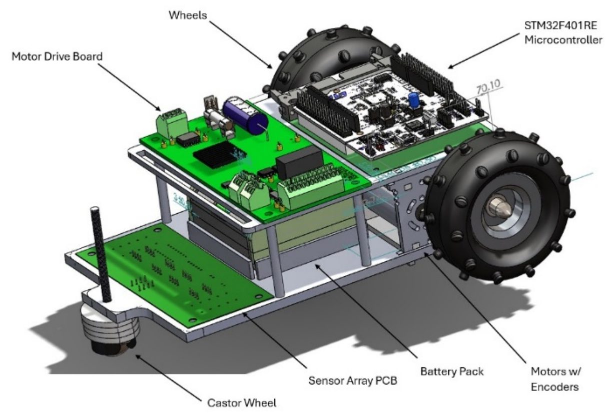


Figure 2: CAD model of buggy

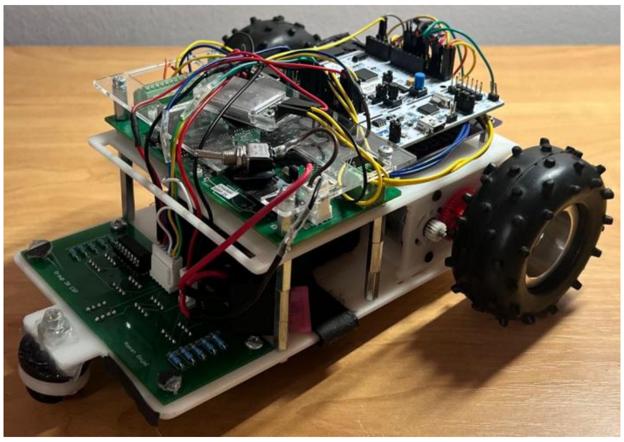


Figure 3: Completed Buggy

2.2 Electronic

To facilitate the buggy's line-following functionality, we used five TCRT5000 sensors, configured as shown in Figure 5. Each sensor consists of two main components: an infrared (IR) LED that continuously emits IR light onto the surface below, and a phototransistor that detects the amount of reflected light. The current flowing through the phototransistor increases proportionally with the intensity of the reflected IR light. When the sensor passes over a white line, more IR light is reflected back, resulting in a higher current. Conversely, when the sensor passes over the black track, less light is reflected, and a smaller current flows. This difference allows the microcontroller to determine the buggy's position relative to the track and guide its movements accordingly.

As shown in Figure 4, each TCRT5000 sensor has its IR LED current-limited by a 75 Ω resistor to prevent excessive current draw and ensure stable operation. All the IR LEDs are connected to two ports on the Darlington buffer which allows us to turn them all on and off simultaneously. The phototransistor side produces an output signal that is pulled up by a 1 kΩ resistor, creating a clear voltage level for detection. Additionally, an LED was connected in parallel with the sensors as a power indicator to confirm proper voltage supply. However, as noted earlier, this LED was later removed because it interfered with the sensors' performance.

Lastly, Figure 6 illustrates the wiring of all the buggy's components. The pins were deliberately selected based on their specific functions, proximity to their respective connections, and port arrangement to prevent shorting. Also, the wiring layout was later organized and categorized to simplify sorting and debugging.

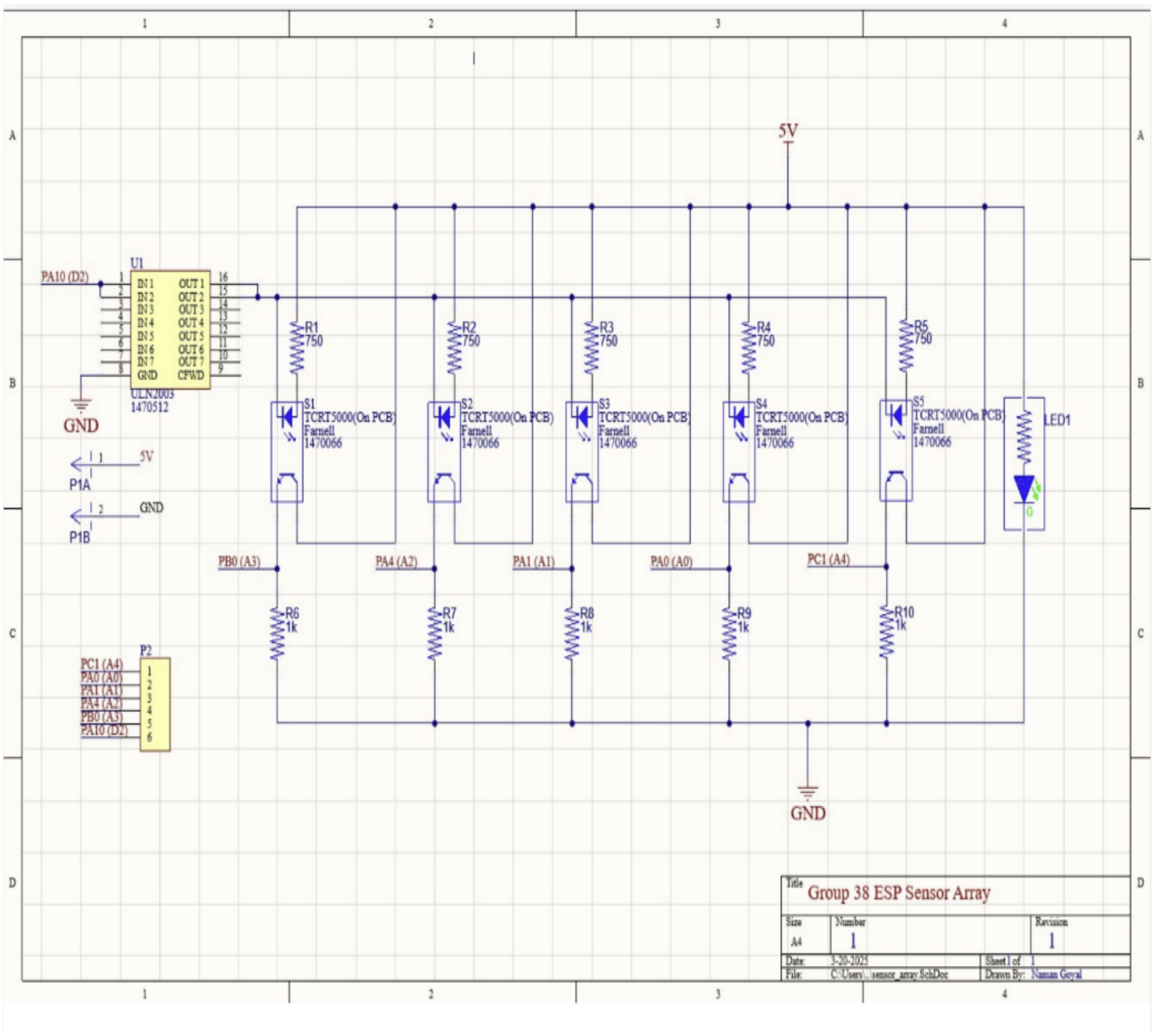

5V
U1
PA10 (D2)
ULN2003
1470512
GND
P1A
P1B
R1 750
R2 750
R3 750
R4 750
R5 750
S1 TCRT5000(On PCB) Farnell 1470066
S2 TCRT5000(On PCB) Farnell 1470066
S3 TCRT5000(On PCB) Farnell 1470066
S4 TCRT5000(On PCB) Farnell 1470066
S5 TCRT5000(On PCB) Farnell 1470066
LED1
PB0 (A3)
PA4 (A2)
PA1 (A1)
PA0 (A0)
PC1 (A4)
R6 1k
R7 1k
R8 1k
R9 1k
R10 1k
P2
Group 38 ESP Sensor Array
Date: 3-20-2025
Drawn By: Naman Goyal

Figure 4: Schematic Diagram of Sensor Circuit

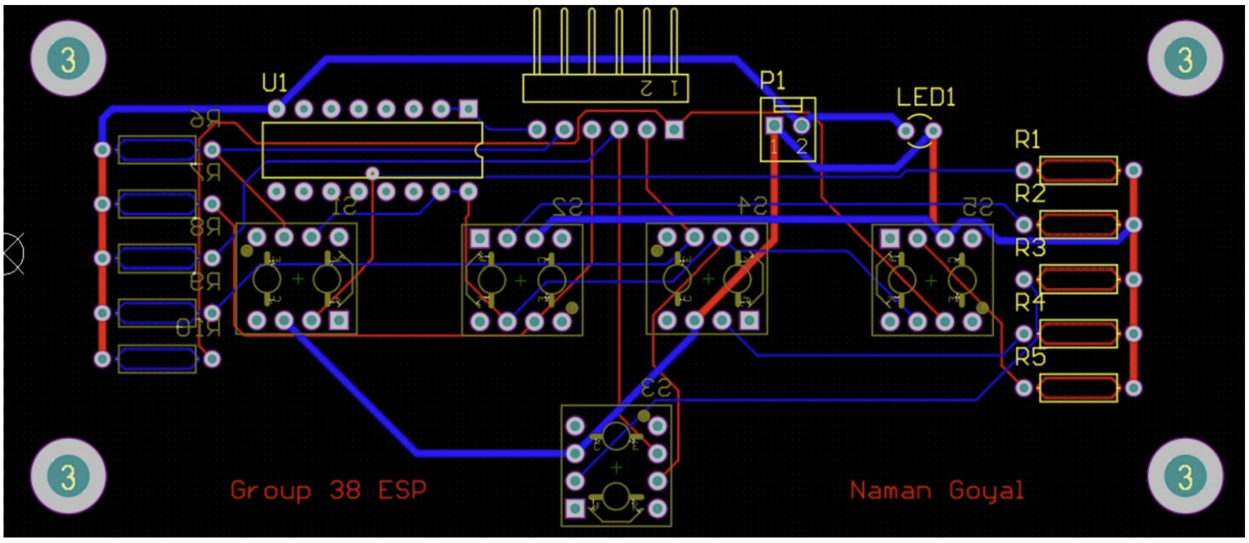


Figure 5: Sensor Circuit PCB

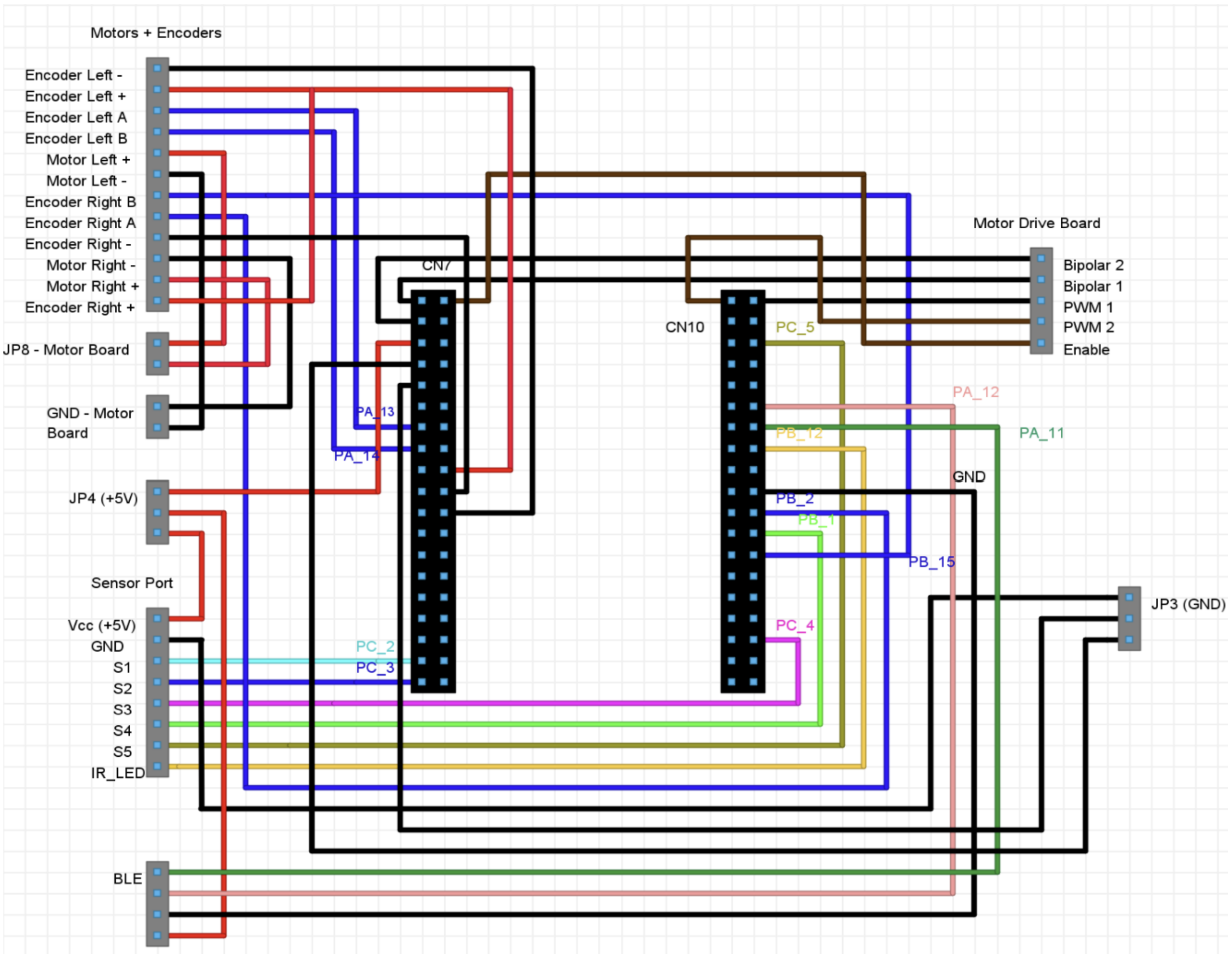


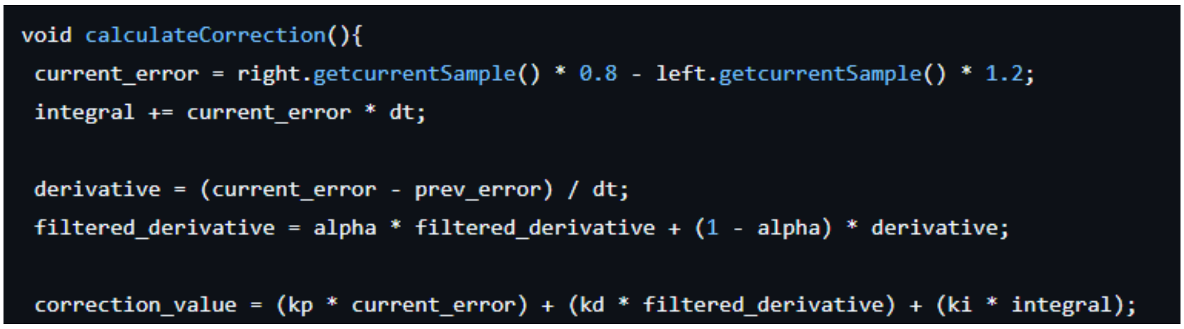
Figure 6: Wiring Diagram

2.3 Control

The buggy's steering control initially aimed to implement full PID. However, practical testing revealed that while proportional and integral terms could be tuned to deliver stable line-following, the derivative term introduced excessive sensitivity and oscillation. As a result, a PI controller was adopted to calculate steering corrections based on differential sensor readings. Speed control was handled independently through proportional feedback using encoder velocity measurements.

The PI controller provided sufficient responsiveness and eliminated steady-state errors (Figure 7). However, testing revealed that the integral term could accumulate excessively, particularly during extended curves, leading to control instability. To mitigate this, a clipping function constrained the integral term to a maximum value of 2.5, determined empirically through testing.

The control system operated at an update frequency of 750 Hz, managed using the microcontroller's Ticker API. This frequency was selected after evaluating various rates to balance responsiveness and computational stability.

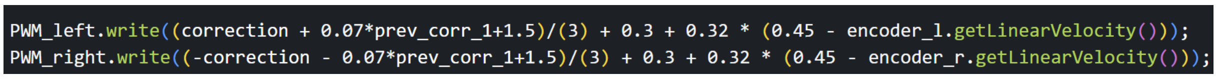


```
void calculateCorrection(){
 current_error = right.getcurrentSample() * 0.8 - left.getcurrentSample() * 1.2;
 integral += current_error * dt;

 derivative = (current_error - prev_error) / dt;
 filtered_derivative = alpha * filtered_derivative + (1 - alpha) * derivative;

 correction_value = (kp * current_error) + (kd * filtered_derivative) + (ki * integral);
```

Figure 7: Code snippet showing correction computation



```
PWM_left.write((correction + 0.07*prev_corr_1+1.5)/(3) + 0.3 + 0.32 * (0.45 - encoder_l.getLinearVelocity()));
PWM_right.write((-correction - 0.07*prev_corr_1+1.5)/(3) + 0.3 + 0.32 * (0.45 - encoder_r.getLinearVelocity()));
```

Figure 8: Code for mapping correction into PWM signals

To ensure reliable line detection under varying lighting conditions, ambient light filtering was implemented. At each sampling cycle, the IR LEDs were turned off to measure ambient light levels. The LEDs were then turned on, and the active IR reading was captured. By subtracting the ambient value, the system isolated the reflected IR signal, improving detection accuracy.

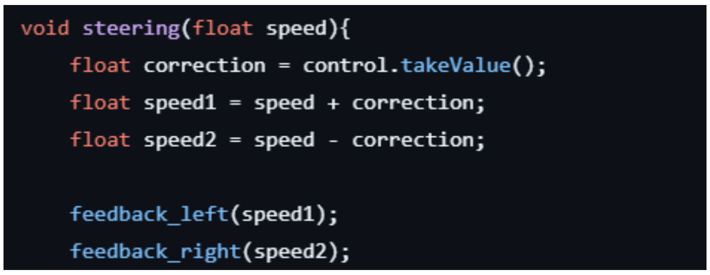
2.4 Software

The overall structure of the code changed completely from the proposed software architecture in Design Report 2 [1]. We opted for a simpler and direct approach to have better code readability, ease of debugging and tuning of the PID constants. To ensure continuous feedback and that the sensors were reactive, we used a ticker to call the steering function from the Motor class (refer to API documentation [2]). This ensures that the buggy is responsive to sensor readings at all times and improves stability.

The line-following algorithm we implemented for steering works by subtracting the readings from the middle two sensors of the IR array. The difference between these two central sensors provides a simple but effective estimate of the lateral deviation from the line's centre. This difference is the error term for PID, which outputs correction. This correction is then fed into the motor control mapping equations and converted into PWM signals (Figure 8). Our previous implementation adjusted the relative speeds of the left and right wheels by adding and subtracting the correction from the buggy target speed (Figure 9) and using full motor speed control. This approach was a lot more complex than our final implementation and proved to be unstable. So, now the motor speed control system basically only has an additive and subtractive effect on the base PWM controller.

Additionally, we later integrated the far-left and far-right sensors as interrupt triggers to detect large deviations (Figure 10) from the line. When activated, these interrupts prompt the buggy to make a strong corrective adjustment in the opposite direction, allowing it to quickly return to the line's centre. We also used the Bluetooth module to add a variety of commands that significantly accelerated our debugging process (as described in more detail in Section 5). Among these was a 180° point-turn command, which enables the buggy to rotate in place until it re-detects the line. Furthermore, Bluetooth was used to handle buggy initialization and provided an emergency stop function for safety during testing.

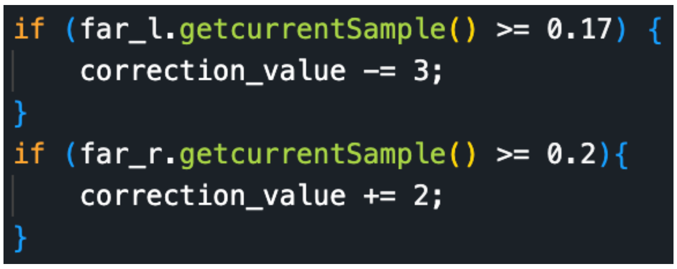
Lastly, the stop module played a crucial role in our system, utilizing input from all five light sensors. Initially, we encountered issues where the buggy would stop at unintended points, particularly at line breaks, where all five sensors would momentarily detect black. To address this, we implemented a timer-based mechanism: the buggy would only stop if all five sensors continuously detected black for the entire countdown period, with the timer duration determined through experimental testing. This approach proved highly effective, and the stop module operated reliably without further issues.



```
void steering(float speed){
    float correction = control.takeValue();
    float speed1 = speed + correction;
    float speed2 = speed - correction;

    feedback_left(speed1);
    feedback_right(speed2);
```

Figure 9: Correction being applied to motor speeds



```
if (far_l.getcurrentSample() >= 0.17) {
    correction_value -= 3;
}
if (far_r.getcurrentSample() >= 0.2){
    correction_value += 2;
}
```

Figure 10: Snippet of far-left and far-right sensors correction

3. Team Organisation and Planning

3.1 Reflecting on Project Objectives

From the project's inception, the objective was to develop a buggy that could follow a lined black track with turns and slopes, as quickly as possible, and using as little energy as possible. First, research and testing were conducted on motors and a variety of sensors and LED's, all of which were potential candidates for the buggy. After picking the TCRT5000 sensor LED array, and gearbox based on calculated torque and back EMF constants, Design Reports 1 and 2 were written, illustrating all the above. Design report 2 further encompassed "software design, sensor selection, control algorithms and chassis design" [2].

After this research phase, code was written in MBED to interface with the motors, encoders and sensors, to develop a line following algorithm, followed by implementing PID speed control (inner sensors) and bang-bang correction (outer sensors). All of these were to be demonstrated via four technical demonstrations. All deliverables were submitted on time. However, the later technical demonstrations showed the buggy lacked some functionality, which will be discussed in section 5.

3.2 Time Management & Gantt Chart

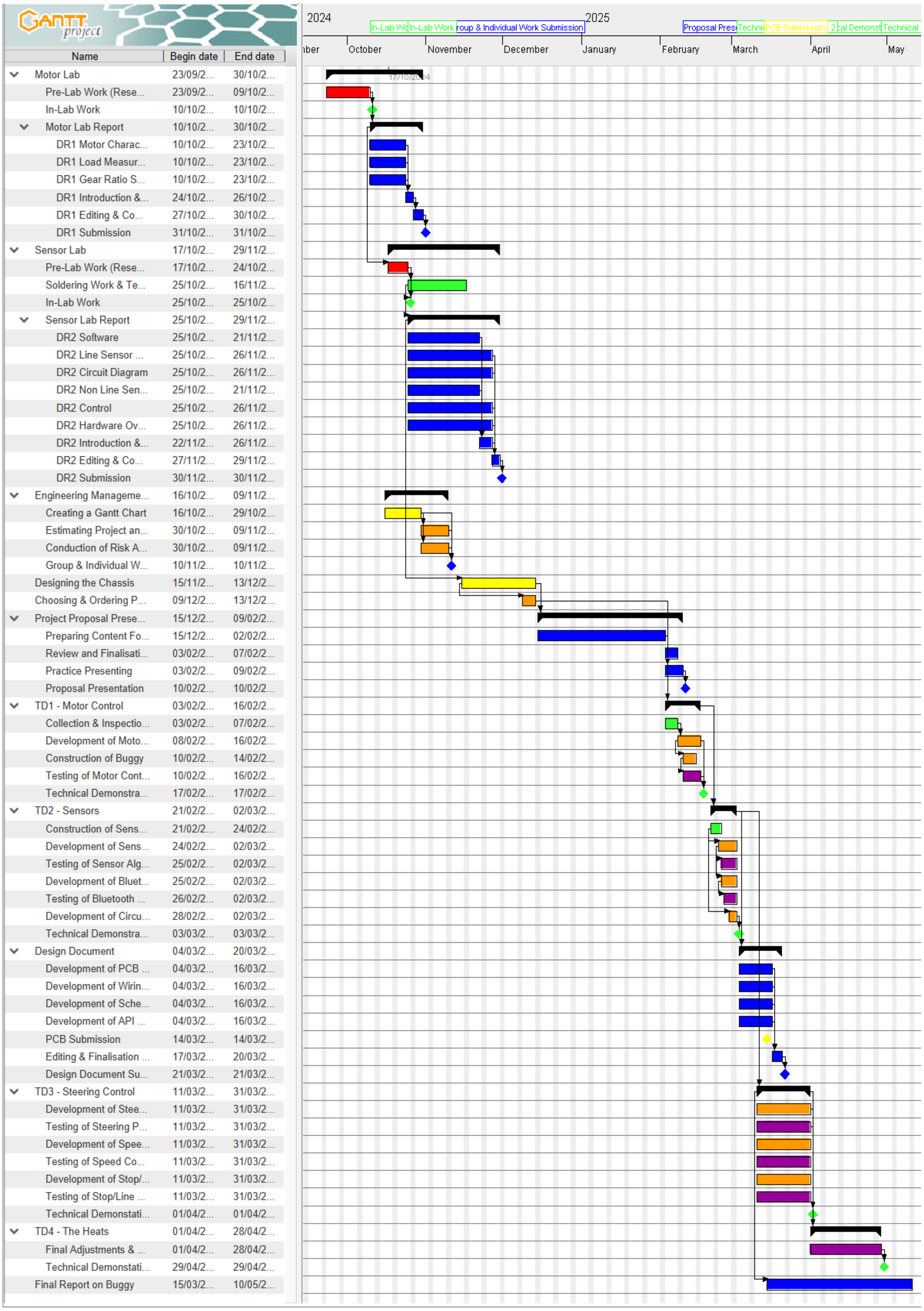
To ensure a smooth roadmap, a Gantt chart was developed, outlining the plan from October to April. As this was well before the project began, tasks were broadly entered, with little information. As the project progressed however, the Gantt Chart was iterated on, filling in specific details on each of the tasks, and splitting them based on which team members tackled them.

Initially, the group had decided to work on all the parts together cohesively, as from initial team building exercises, there weren’t clear distinctions between the group members that set them apart. However, as work progressed, it was realised that the most efficient way to split the work was to do it based on their skillsets, rather than having everyone work on 1 task simultaneously. This in turn gave all members a clear role, with some more focused on programming, and others on dealing with the buggy hardware. This made working on the DR1 and DR2 deliverables easier, as each group member had a clear understanding of 1 specific section.

However, a flaw in both the initial and final Gantt Charts, was that not enough time was allocated to debugging problems. Due to the initial successes of the group, the group was very optimistic about completing the objectives, hence less time was given to debug and replace parts if needed. Often this meant having to backtrack and redo previous tasks, which took up much needed time, and was not accounted for. This proved to be fatal to the group’s success during TD4, when not enough time was allocated to debug the buggy’s line following. This resulted in an incorrect diagnosis of the issue with the buggy and wasted 2 days of time. This is further elaborated on in section 5.

All deliverables were scheduled for completion at least one day in advance, with report drafts prepared three days early to allow time for proofreading and comparison against the mark scheme and previous feedback. For technical demonstrations, the team worked daily to troubleshoot the buggy by testing components, isolating faults, and refining control algorithms through constant tuning and code edits. At least two members regularly used the dry lab or other available spaces outside allocated lab sessions for testing.

The open nature of the work plan allowed the group to address problems flexibly. This led to the development of a system for adjusting PID constants and speed values directly in the code, improving testing efficiency. However, the team assumed hardware was not the source of issues and did not allocate enough time for hardware-level debugging. This became critical during TD3, where persistent upload failures—eventually traced to a faulty microcontroller—delayed progress. Avoiding early hardware replacement resulted in rushed code changes and contributed to marks lost in TD3 and TD4. A similar issue resurfaced during the heats. Future projects will need to prioritise early hardware validation to avoid such setbacks.



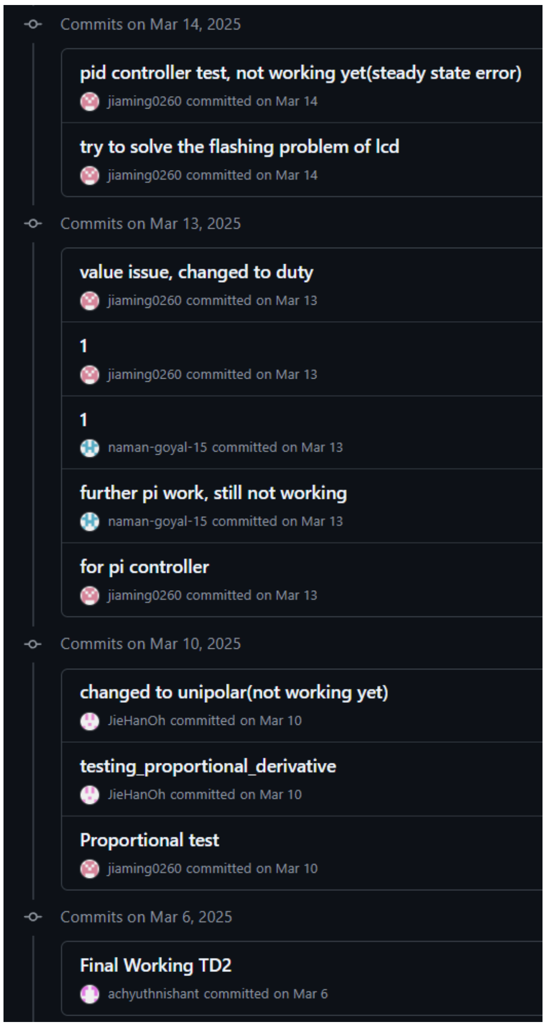
Name	Begin date	End date
Motor Lab	23/09/2...	30/10/2...
Pre-Lab Work (Rese...	23/09/2...	09/10/2...
In-Lab Work	10/10/2...	10/10/2...
Motor Lab Report	10/10/2...	30/10/2...
DR1 Motor Charac...	10/10/2...	23/10/2...
DR1 Load Measur...	10/10/2...	23/10/2...
DR1 Gear Ratio S...	10/10/2...	23/10/2...
DR1 Introduction &...	24/10/2...	26/10/2...
DR1 Editing & Co...	27/10/2...	30/10/2...
DR1 Submission	31/10/2...	31/10/2...
Sensor Lab	17/10/2...	29/11/2...
Pre-Lab Work (Rese...	17/10/2...	24/10/2...
Soldering Work & Te...	25/10/2...	16/11/2...
In-Lab Work	25/10/2...	25/10/2...
Sensor Lab Report	25/10/2...	29/11/2...
DR2 Software	25/10/2...	21/11/2...
DR2 Line Sensor ...	25/10/2...	26/11/2...
DR2 Circuit Diagram	25/10/2...	26/11/2...
DR2 Non Line Sen...	25/10/2...	21/11/2...
DR2 Control	25/10/2...	26/11/2...
DR2 Hardware Ov...	25/10/2...	26/11/2...
DR2 Introduction &...	22/11/2...	26/11/2...
DR2 Editing & Co...	27/11/2...	29/11/2...
DR2 Submission	30/11/2...	30/11/2...
Engineering Manageme...	16/10/2...	09/11/2...
Creating a Gantt Chart	16/10/2...	29/10/2...
Estimating Project an...	30/10/2...	09/11/2...
Conduction of Risk A...	30/10/2...	09/11/2...
Group & Individual W...	10/11/2...	10/11/2...
Designing the Chassis	15/11/2...	13/12/2...
Choosing & Ordering P...	09/12/2...	13/12/2...
Project Proposal Prese...	15/12/2...	09/02/2...
Preparing Content Fo...	15/12/2...	02/02/2...
Review and Finalisati...	03/02/2...	07/02/2...
Practice Presenting	03/02/2...	09/02/2...
Proposal Presentation	10/02/2...	10/02/2...
TD1 - Motor Control	03/02/2...	16/02/2...
Collection & Inspectio...	03/02/2...	07/02/2...
Development of Moto...	08/02/2...	16/02/2...
Construction of Buggy	10/02/2...	14/02/2...
Testing of Motor Cont...	10/02/2...	16/02/2...
Technical Demonstra...	17/02/2...	17/02/2...
TD2 - Sensors	21/02/2...	02/03/2...
Construction of Sens...	21/02/2...	24/02/2...
Development of Sens...	24/02/2...	02/03/2...
Testing of Sensor Alg...	25/02/2...	02/03/2...
Development of Bluet...	25/02/2...	02/03/2...
Testing of Bluetooth ...	26/02/2...	02/03/2...
Development of Circu...	28/02/2...	02/03/2...
Technical Demonstra...	03/03/2...	03/03/2...
Design Document	04/03/2...	20/03/2...
Development of PCB ...	04/03/2...	16/03/2...
Development of Wirin...	04/03/2...	16/03/2...
Development of Sche...	04/03/2...	16/03/2...
Development of API ...	04/03/2...	16/03/2...
PCB Submission	14/03/2...	14/03/2...
Editing & Finalisation ...	17/03/2...	20/03/2...
Design Document Su...	21/03/2...	21/03/2...
TD3 - Steering Control	11/03/2...	31/03/2...
Development of Stee...	11/03/2...	31/03/2...
Testing of Steering P...	11/03/2...	31/03/2...
Development of Spee...	11/03/2...	31/03/2...
Testing of Speed Co...	11/03/2...	31/03/2...
Development of Stop/...	11/03/2...	31/03/2...
Testing of Stop/Line ...	11/03/2...	31/03/2...
Technical Demonstrati...	01/04/2...	01/04/2...
TD4 - The Heats	01/04/2...	28/04/2...
Final Adjustments & ...	01/04/2...	28/04/2...
Technical Demonstrati...	29/04/2...	29/04/2...
Final Report on Buggy	15/03/2...	10/05/2...

Figure 11: Final Iteration of Group 38 Gantt Chart

3.3 Organisation of Work

Multiple systems were put in place to ensure most time was spent progressing on the buggy. A GitHub repository was created, allowing multiple sections of code to be developed in tandem. The code was organised into different header files, each containing classes responsible for specific functions of the buggy: motor and encoder control, sensor input, LED control, speed control, potentiometers, and Bluetooth. Different members were responsible for writing different header files, with those more confident in programming reviewing and verifying them when necessary. This allowed any member to make changes as needed, rather than relying on a single person to store and edit the code. The GitHub repository proved extremely valuable for reverting to or comparing previous versions of code, which aided debugging and restoring working configurations.

All hardware and equipment were stored securely in the dry lab. Since some group members often made progress while the lab was open, the key to the equipment was handed over to whoever was working that day. At times, equipment needed to be taken home for after-hours work, including the black track sections, BLE module, and microcontroller, to allow for testing and coding outside scheduled lab time. This was especially useful when testing the line-following algorithm and developing the Bluetooth command system for tuning PID constants.


Commits on Mar 14, 2025

pid controller test, not working yet(steady state error)
jiaming0260 committed on Mar 14

try to solve the flashing problem of lcd
jiaming0260 committed on Mar 14

Commits on Mar 13, 2025

value issue, changed to duty
jiaming0260 committed on Mar 13

1
jiaming0260 committed on Mar 13

1
naman-goyal-15 committed on Mar 13

further pi work, still not working
naman-goyal-15 committed on Mar 13

for pi controller
jiaming0260 committed on Mar 13

Commits on Mar 10, 2025

changed to unipolar(not working yet)
JieHanOh committed on Mar 10

testing_proportional_derivative
JieHanOh committed on Mar 10

Proportional test
jiaming0260 committed on Mar 10

Commits on Mar 6, 2025

Final Working TD2
achyuthnishant committed on Mar 6

Figure 12: Snapshot of GitHub repository commits by group members

4. Budget vs. Outturn

The total cost of the buggy project can be split into several categories, including component costs, equipment costs, research and development (R&D) costs, software costs and marketing costs. All mechanical components and software licences (e.g. SolidWorks student edition) were supplied at no charge. Consequently, the only direct expenditures are R&D activities and consumable replacement parts. The table below shows the details of these direct costs.

Table 1 R&D costs and replacement costs

Component Name	Price Per Unit (£)	Amount	Total Costs (£)
PCB Boards	1.6	2	3.2
Batteries	1.2	8	9.6
Black Paper	0.12	50	6
White Tape	1.9	1	1.9
Motor	5	1	5
STM32 Nucleo	13.55	2	27.1

The total cost of R&D replacement materials is £52.8. During the prototyping phase, two additional PCB boards were requisitioned to compensate for soldering errors. The broken left motor and two STM32 Nucleo boards had been replaced during the heats. Black paper and electrical tape were bought to construct a testing track at home. Furthermore, some supplementary batteries were also purchased to maintain an uninterrupted testing cycle, ensuring that a charged set is always available.

When considering the sale of buggies on an open market, a comprehensive cost analysis is required to establish a selling price and estimate the revenue. The table below lists all categories of costs in detail.

Table 2 Project total costs

Name	Price Per Unit (£)	Amount	Total Costs (£)
Component costs	354.24	-	354.24
R&D material costs	52.8	-	52.8
R&D labour costs	20.51	1050	21535.5
Software costs	-	-	1920.8
Marketing costs	-	-	6000
Equipment costs	-	-	1015

The total project costs are £30,878.34. The calculation details of component costs, software costs, equipment costs and marketing costs align with the initial estimation in Engineering Management Coursework – predicted project cost [3]. However, the total R&D labour hours were underestimated in the Semester 1 coursework because the technical demonstrations required more time than anticipated. It changed from 131 hours per person to 210 hours per person, and the total working time of the team is 1050 hours with an estimated hourly wage of £20.51.

Although the total production cost of each buggy amounts to £30,878.34 the equipment costs, software costs and marketing costs are assumed to be paid annually. Moreover, now that both the software framework and hardware architecture are fully established, the R&D labour expenses to assemble each buggy are expected to be £102.6 (5 hours), bringing the per-unit cost down to approximately £456.84 by simply adding the component costs to the R&D labour costs within the same fiscal year.

Additionally, assume an outturn of 90%, which means 10% of the buggies require rework or scrap. To cover such costs and deliver a healthy buggy, 20% of total costs need to be added on top of the basic cost. This implies that the minimum price of each unit is about £548.21. A similar product called Husarion-Robot 2.0 Pro [4], which integrals Intel CPU, GPU and a RGBD camera, retail at £2920.3. As a result, our buggy has a lower material cost, and it should have a selling price of about £1000. With an expected annual volume of 100 units, break-even is achieved after 70 sales. The remaining units sold would generate an estimated annual revenue of £14,344, which represents a 15% margin on the full year's output.

Ince all direct and annualized expenses are included, the fully burdened cost per buggy is approximately £548.21. A 20% buffer for quality assurance has also been taken into consideration. To balance optimal value with market competitiveness, a retail price of £1,000 per unit is recommended.

5. Analysis of Heats

5.1 Preparation and testing before Heats

In preparation for the heats, extensive testing was carried out to ensure consistent line-following performance and reliable motor control. Most testing took place in a controlled lab environment, where the buggy successfully completed the course multiple times without failure. Additional testing was also conducted at home on a custom-built track (Figure 14), providing extra opportunities to fine-tune the control systems outside of scheduled lab hours. Preparation focused on three key areas:

5.1.1 Motor Speed Control:

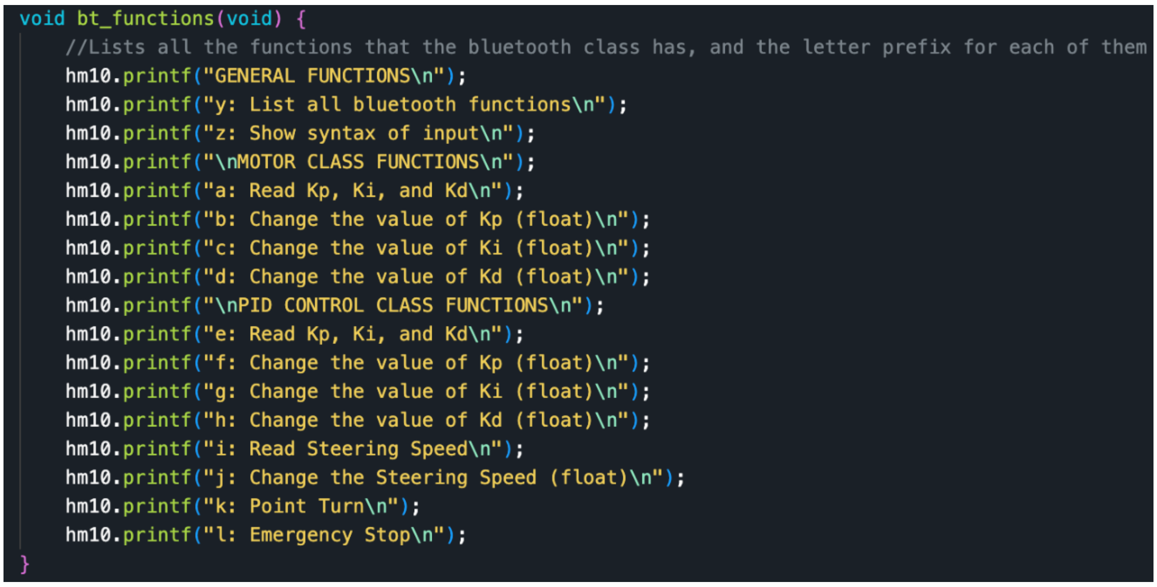
A closed-loop PID-based feedback system was implemented for motor velocity regulation. Iterative tuning helped minimize steady-state error and respond to terrain changes such as inclines. A Bluetooth interface was also implemented to allow real-time tuning of PID coefficients without needing to re-upload code. This significantly accelerated the testing cycle.

5.1.2 Sensor Calibration and Signal Processing:

The IR sensor array, consisting of five analogue light sensors, was improved with a simple ambient light rejection system. Each sensor captured two readings: one with the IR LED turned on and one with it turned off. By subtracting the two, the system obtained a more accurate measure of reflected light, effectively filtering out ambient lighting effects. Under indoor testing conditions, this technique proved reliable and significantly reduced false readings caused by external light sources.

5.1.3 Line-Following Logic:

Several versions of the line-following controller were developed. These used weighted sensor readings to compute a lateral error, which was converted into an angular velocity (omega) command for differential drive. A hybrid control scheme was also tested, which toggled between open-loop PWM and closed-loop feedback based on speed error. This aimed to balance responsiveness and stability but required further tuning to be reliable. Frequent code structure changes were made to improve performance, but these limited the time available for thorough tuning and introduced additional variability.



```
void bt_functions(void) {
    //Lists all the functions that the bluetooth class has, and the letter prefix for each of them
    hm10.printf("GENERAL FUNCTIONS\n");
    hm10.printf("y: List all bluetooth functions\n");
    hm10.printf("z: Show syntax of input\n");
    hm10.printf("\nMOTOR CLASS FUNCTIONS\n");
    hm10.printf("a: Read Kp, Ki, and Kd\n");
    hm10.printf("b: Change the value of Kp (float)\n");
    hm10.printf("c: Change the value of Ki (float)\n");
    hm10.printf("d: Change the value of Kd (float)\n");
    hm10.printf("\nPID CONTROL CLASS FUNCTIONS\n");
    hm10.printf("e: Read Kp, Ki, and Kd\n");
    hm10.printf("f: Change the value of Kp (float)\n");
    hm10.printf("g: Change the value of Ki (float)\n");
    hm10.printf("h: Change the value of Kd (float)\n");
    hm10.printf("i: Read Steering Speed\n");
    hm10.printf("j: Change the Steering Speed (float)\n");
    hm10.printf("k: Point Turn\n");
    hm10.printf("l: Emergency Stop\n");
}
```

Figure 13: List of Bluetooth functions

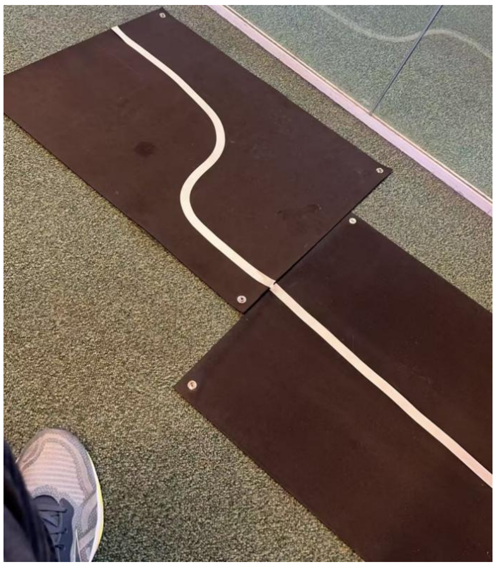


Figure 14: Custom home testing track created using electrical tape and black mats

5.2 Performance During the Heats

The buggy failed to complete the full course in a single uninterrupted run during the final demonstration. Its performance during the heats was significantly poorer than during prior testing, where it had successfully completed full track runs under controlled conditions. Several technical and environmental factors contributed to the outcome across the three competition attempts.

Pre-Attempt Setup: Hardware and Structural Limitations

Before the first attempt, multiple hardware issues were encountered. One of the motor encoders failed unexpectedly during morning setup, disabling the feedback control system and consuming a significant portion of available preparation time. Persistent code uploading failures also necessitated the replacement of the microcontroller shortly before the heats began.

The replacement motor introduced further complications. Its different shaft spacing required recalibration of turning behaviour. However, due to limited time, only partial tuning could be performed. A software change was introduced to increase the turning correction factor, but this was a rushed adjustment with limited impact.

The vertically stacked chassis also presented a challenge during maintenance. Accessing internal components such as wiring and motors required disassembling multiple layers, which slowed down hardware interventions and increased stress during debugging under time pressure.

First Attempt: Turning Failure from Incomplete Tuning

The first attempt ended prematurely due to poor turning performance. The rushed PID adjustments and modified turning routine could not adequately compensate for the mechanical differences introduced by the new motor. As a result, the buggy failed to complete the left turn before the uphill section. To maintain basic movement, parts of the control system were switched to open-loop PWM control, compromising accuracy for simplicity.

Second Attempt: Lighting Conditions and Sensor Disruption

After limited tuning between runs, the buggy demonstrated improved behavior in testing conditions. However, it again failed during the second attempt. The lighting environment on the competition track was significantly different from that used during previous testing. Despite the ambient subtraction method integrated into the sensor system, direct sunlight overwhelmed the IR sensors, resulting in unreliable line detection.

An emergency fix was implemented using black paper to shield the sensor array from ambient light. While this provided some improvement, it was insufficient to overcome the instability caused by glare and resulted in another failed run.

Third Attempt: Understeering and Sensor Geometry Limitations

During the third attempt, the buggy successfully completed the majority of the course but veered off during the downhill segment. The observed failure was characterised by gradual understeering, rather than erratic oscillation or loss of control. The exact cause could not be conclusively identified, but two factors are considered likely contributors.

First, ambient lighting may again have impacted sensor readings despite mitigation efforts. Second, and more significantly, the geometry of the sensor array appeared to limit the system's ability to respond effectively at speed. The two middle sensors were placed too close together to provide stable central tracking, while the outer sensors were spaced too far apart to deliver meaningful corrective influence during sharp turns. This configuration reduced the buggy’s ability to generate smooth and proportional steering corrections.

This issue could potentially have been mitigated by adopting a more balanced sensor layout or by assigning pre-defined turning commands to the outer sensors, rather than treating their input as linear corrections within the existing control scheme.

Ongoing Issues: Program Instability and Upload Complications

Throughout all three attempts, a persistent issue complicated debugging: code could not be uploaded while the encoder wires were connected. This problem remained unresolved even after the microcontroller replacement. Each firmware upload required manual disconnection of the encoders, which introduced time delays and undermined testing consistency. The workaround also made it difficult to identify whether performance issues stemmed from hardware faults, software errors, or instability introduced by the upload process. While this issue was reported during the event, no resolution could be implemented in the available timeframe.

5.3 Performance Evaluation

Based on the performance of the buggy during the heats, distinct strengths and weaknesses in the buggy's design and functionality became evident.

Most Successful Features:

- **Feedback Motor Control System:** When functioning correctly, the PID-based motor control significantly improved performance, especially on inclines and declines, by adapting the motor outputs in real time to track speed commands.
- **Low Centre of Gravity:** The compact chassis, with its centrally mounted battery pack, lowers the buggy's centre of gravity, resulting in smoother turns and improved climbing performance.
- **Switching Logic for Control Modes:** The hybrid logic that toggled between feedback and open-loop PWM based on speed error was an innovative solution. It enabled rapid transitions between accuracy and speed, although it required further tuning and refinement to be fully effective.
- **Design of BLE firmware:** Additional Bluetooth-accessible commands for adjusting PID coefficients and triggering an emergency stop greatly enhance testing and tuning efficiency.

Least Successful Features / Weakest Points

- **Sensor Array Geometry:** As referred before, the two middle sensors in our 5-sensor array were placed too close together. This made it harder for the buggy to detect subtle shifts in position, especially during curves. Additionally, the outer sensors were too far apart to have enough of an effect on the steering unless we implemented a specific controlled turn.
- **Inconsistent Program Behaviour:** The buggy's control software was unpredictable, sometimes functioning correctly and other times failing under similar conditions. This inconsistency could not be definitively attributed to hardware, logic, or environmental factors, making debugging extremely challenging.

- **Inadequate Light Robustness:** Despite the ambient subtraction technique, the sensor system proved to be insufficiently robust against harsh or rapidly changing light conditions, such as direct sunlight or glare.
- **Hardware Accessibility:** The vertical "stacked" chassis design made hardware repairs or part replacements time-consuming and stressful. A modular or side-access design would have been far more practical under race conditions.
- **Lack of Risk contingency plan:** Hardware failures were not anticipated, and no effective or precise contingency exists for such critical situations.

Despite strong pre-race testing, the buggy's performance was compromised by last-minute hardware failures and the absence of a contingency plan. The team's reactive approach fell short under pressure, underscoring the need for early hardware validation, risk planning, and repair-friendly designs. Future versions should feature a more accessible chassis, improved sensor layout, and stricter testing under varied conditions.

6. References

[1] ESP Group 38, "Design Report 2", University of Manchester, Nov 2024

[2] ESP Group 38, "API Documentation", University of Manchester, March 2025

[3] ESP Group 38, "Predicted Project Cost", University of Manchester, Nov 2024

[4] "Husarion - ROSBOT 2.0 PRO," *Robosavvy.co.uk*, 2025. https://robosavvy.co.uk/rosbot2pro.html (accessed May 03, 2025).

Continuing Professional Development Log

Name: *Oh Jie Han*

Current and recent CPD activity:

CPD Activity Title	Description	Dates	CPD Hours
Internship at AMD	-Developed a shell bash script to extract PCIE features from endpoints, enhancing system diagnostics and performance analysis -Created an additional python script to compare register values, streamlining the process of identifying discrepancies and ensuring data integrity -Collaborated with my team to integrate these scripts into existing workflow, improving overall efficiency	June 2024-Sep 2024	450 hours
University of Manchester Hackathon	-Collaborated with my friends on using computer vision to communicate with a Mona Bot to complete an obstacle course -Worked on motor control of the bot and successfully implemented a line-following algorithm, using PID control combined with computer vision	March 2025	24 hours
Mars Rover Project	-Collaborated with students from the university to create a replica of the Mars rover, where it is able to be controlled using a remote controller, pick up objects with a small arm and record its surroundings -Helped decide on the components and design of the rover, using equations to find the required torque, speed, gearbox and weight required -Did research on gear types and watched videos of other people building a similar rover -Helped assembled a 3D test model of the rover	Feb 2025-Aug 2025	30 hours

Planned and Future CPD activity:

CPD Activity Title	Description	Skills addressed	Dates
Internship at Lattice Semiconductor	-IP and System design team -Full time intern for 3 months -Will be working in a professional environment and collaborating with a team on projects -Meeting other likeminded and ambitious interns	- RTL programming -C++ and Python programming -VHDL/Verilog	June 2025-Sep 2025
Investing and stocks	-Planning to take courses on investing and understanding stocks -Making use of Youtube and online materials as well to learn basics on investing	-investing and stocks	June 2025- Sep 2025
Preparations for master's study	-Applying to universities for a masters in engineering -Research on projects different universities are doing and complete applications -Study and prepare for entrance exams	-research -revision	Sep 2025-Sep 2026

Continuing Professional Development Log

Name: *Naman Goyal*

Current and recent CPD activity:

CPD Activity Title	Description	Dates	CPD Hours
Internship at Dtown Robotics	Worked on designing and optimizing a (3+1) Degrees of Freedom manipulator arm for an off-road Unmanned Ground Vehicle (UGV). Responsibilities included actuator control, system layout, performance tuning, and ensuring the arm's functionality under rugged conditions. Gained hands-on experience with Arduino-based systems, embedded programming, and mechanical design.	Jul-Aug 2024	320
RoboSoc Hackabot 2025	Participated in a 24-hour robotics hackathon, where we developed a computer vision model for object and facial recognition using a Sony camera and Raspberry Pi. The team placed runner-up in the competition.	Mar 2025	18

Planned and Future CPD activity:

CPD Activity Title	Description	Skills addressed	Dates
UoM SoE Summer Internship 2025	Assist in integrating a novel excavation tool onto a small planetary rover system as part of a lunar in-situ resource utilisation (ISRU) project. Contribute to mechanical integration, development of control systems, and simulation/testing using Project Chrono.	Robotics integration, control systems, Python, simulation, aerospace engineering	Summer 2025
ROS & Gazebo Self-Project	Plan to explore ROS and Gazebo through online tutorials and a personal simulation project.	ROS basics, simulation, robotics middleware	Jul–Sep 2025
Space Robotics Online Course	Enroll in a recognized online course focusing on robotic systems for extraterrestrial environments.	Space systems, autonomy, control systems	Aug–Oct 2025
Year-Long Placement at Strata	Join Strata’s R&D department for a year-long placement, focusing on manufacturing automation and some robotics. Work on optimizing automation systems and implementing robotic solutions in a manufacturing context.	Manufacturing automation, robotics, process optimization	Sept 2025–July 2026
Computer Vision Model Enhancement	Continue improving the object and facial recognition model developed during the Hackabot event by refining detection accuracy and testing with new datasets. Also plan on adding additional functionality to be able to perform more advanced data analytics.	Computer vision, Python, machine learning basics, data analytics	Jul–Sep 2025

Continuing Professional Development Log

Name: Jiaming Shang

Current and recent CPD activity:

CPD Activity Title	Description	Dates	CPD Hours
University Robotics Hackathon	Took part in a 24-hour hackathon where our team designed and built a robotic buggy that could carry nuclear materials to a safe zone. We used an Arduino board to control the motors and added simple camera modules for obstacle detection. It was intense, hands-on, and a great test of both teamwork and rapid prototyping skills under pressure.	March 29, 2025	24
Robot Arm Summer School	Spent two weeks at a summer school program focused on building and programming a robotic arm for a garbage-sorting task. I helped with hardware assembly—mounting servos, connecting joints—and also worked on the software side, using Python to control movements. We also implemented basic computer vision techniques to detect and classify colors and objects.	July 1–14, 2022	80
SolidWorks Design Seminar	Joined a short seminar held by university engineering faculty that introduced the basics of SolidWorks. We walked through simple modeling exercises and learned how to build 3D parts and basic assemblies. This gave me a solid foundation for exploring CAD design in greater depth.	November 23, 2024	2

Planned and Future CPD activity:

CPD Activity Title	Description	Skills addressed	Dates
Python for Data Science Workshop	Planning to attend a three-day workshop focused on applying Python in data science. It will cover tools like pandas, NumPy, and Matplotlib, and also touch on basic machine learning using scikit-learn. I am especially looking forward to working through real-world datasets during the practical sessions.	This should help strengthen my data handling and visualization skills, and give me a stronger foundation in Python for technical and research projects.	July 15–17, 2025
Academic Writing Seminar	I have signed up for an online one-day seminar focused on academic writing, particularly how to structure papers, write stronger abstracts, and communicate ideas clearly in a research setting. Hoping it'll give me more confidence when working on future publications and reports.	Will improve my academic communication, especially in writing formal research papers and proposals — something I want to get better at before my thesis work ramps up.	July 23, 2025
Leadership and Team Management Course	Planning to take a short course that introduces project management, leadership in engineering teams, and effective communication. It includes group-based exercises and real scenarios, which should be helpful as I take on more responsibility in team settings.	Will help build soft skills like leadership, planning, and team coordination, useful both for university group work and future roles in industry or research teams.	August 2025

Continuing Professional Development Log

Name: Hassen Al-Rubei

Current and recent CPD activity:

CPD Activity Title	Description	Dates	CPD Hours
BMS controller Project for Formula Student Society	Engineered a custom Battery Management System (BMS) controller PCB tailored for the Formula Student Electric Vehicle (FSAE) application. Responsibilities included schematic design, PCB layout, component selection, and design-for-manufacture considerations. Integrated balancing, safety monitoring, and communication protocols to ensure compliance with competition technical standards. Collaborated with interdisciplinary team members throughout the development cycle.	Feb – May 2025	60 hours, including self-teaching, PCB design tools etc
Precharge circuit PCB for Formula Student Society	Designed a dedicated Precharge Circuit PCB to safely manage inrush currents during the power-up sequence of the Formula Student Electric Vehicle (FSAE). Tasks included schematic creation, PCB layout, MOSFET and resistor selection, and integration with the vehicle's high-voltage system architecture. Addressed challenges related to transient suppression, relay control, and compliance with Formula Student electrical safety regulations. Collaborated with team electrical and control subgroups to ensure system-wide compatibility and safety.	Nov – June 2024	40 hours, including self teaching, PCB design tools, testing etc

Planned and Future CPD activity:

CPD Activity Title	Description	Skills addressed	Dates
CS50x course for C programming	I plan on taking a C programming course during the summer of 2025, as a recap and foundational course to enhance my skills in embedded systems and low level software design.	Low-level software design, embedded systems	June 2025
STM32 Project	Following the CS50x course in C programming, I plan on focusing on the HAL library provided by STMicroelectronics for the STM32 microcontroller. This learning journey was inspired by Prof. Liam Marsh after providing me with advice on action plans following the Microcontroller Engineering II course during Y2 SEM1. I hope to focus my projects in the automotive sector, with the specific project to be finalised based on ongoing skills development. I hope to utilize these skills in applying to industrial placements at prestigious engineering companies in the future.	HAL (LL) library, STM32 development, Circuit design	July – Sep 2025