The University of Manchester

Department of Electrical and Electronic Engineering

Embedded Systems Project 2023-24

Final Report

Group Number: 48

Group members:	ID Number	I confirm that this is the group's own work.
Mohamad Amierul Hakeem	11098269	☒
Teethabhumi Tripatarasit	11114360	☒
Jiexi Lu	11488522	☒
Yanhua Zheng	11047457	☒
Sarah El-Mahmoudi	10864355	☒

Tutor: Yin, Wuliang

Date: 08/05/2024

Contents

1. Executive Summary

Throughout the past year the team has worked together to develop a line following buggy. Through several iterations of design and testing, a robust speedy buggy was built and successfully completed the track requirements in the technical demonstrations and the final race. Towards the beginning of the year research and testing was undertaken to design the optimum layout and configuration of the sensor board circuit, select the most suitable gear box and design a functional chassis. After building the buggy, redesigns and adaptations took place to the sensor board and chassis to eliminate any potential sources of error in the hardware early on.

The final circuit included 6 equally spaced TCRT500 sensors, controlled by a Darlington array, outputting to the analogue pins of the microcontroller. This was decided to enable the implementation of a PID algorithm with each of the sensors providing a weighted input to the final position calculation. The final chassis design, as seen below, consisted of two acrylic tiers, with a 3D printed adapter to fixate the upper tier and a network of holes to route the electrical wires through. A15:1 gearbox was selected to meet our weight requirement whilst ensuring the buggy was also competitive. The remaining time was spent developing the software and testing the PID algorithm and the functionality of the Bluetooth module. Separate PD algorithms were used for line following and PI algorithms for motor speed control. Adjustments to the software were made continuously until the maximum stable speed of the buggy was achieved.

A comprehensive plan for team structure and organisation was also developed early on to establish effective channels for communication, sharing ideas and troubleshooting problems. A Gannt chart was used to monitor progress, shared documents were used to collaborate on reports and a shared GitHub repository was used to develop and test the code simultaneously. The successful completion of the technical demonstration can be attributed to the effective teamwork, planning and task delegation throughout the course of the year.

The buggy performed well in all three of the technical demonstrations, achieving full marks in TDA, TDB and TDC. Adjustments were made between the technical demonstrations to tune motor control and sensor algorithms. The main objective of completing the track was achieved. Further improvements to the buggy's performance could be made by further tuning the PID controller as the maximum motor speed could not be achieved with a stable line following. Additionally, once maximum speed is achieved via software modifications, the next limiting factor – the gearbox – can be improved. Improvements to chassis design may allow for a lighter and more mechanically efficient design making a 15:1 ratio gearbox sufficient to meeting the slope requirements.

2. Introduction

The final report aims to summary the engineering design process of the line following buggy and reflect on the overall outcomes. The final system components section discusses the technical aspects of the electronics, motor control, chassis, and software features. Results from experiments that played a decisive role such as sensor and motor tests are reviewed, as well as calculations for torque and turning. Iterations of

the design are evaluated, and justifications are provided for the final design.

The team organisation section discusses the team structure and standards for delegating tasks and meeting deadlines. A review of the real timeline is compared to the original Gannt chart that set out the main deadlines, including setbacks that we faced during the year. The budget section discusses spending on additional features and the total cost of the buggy. Production costs are evaluated if the buggy were to be sold and retail sale price is estimated.

The outcomes section analyses the performance of the heats, testing methods and lessons learnt between technical demonstrations. Data from the testing is used to demonstrate the tuning process to create a smoother performance. Key features of the buggy are highlighted as well as limiting factors and errors in design.

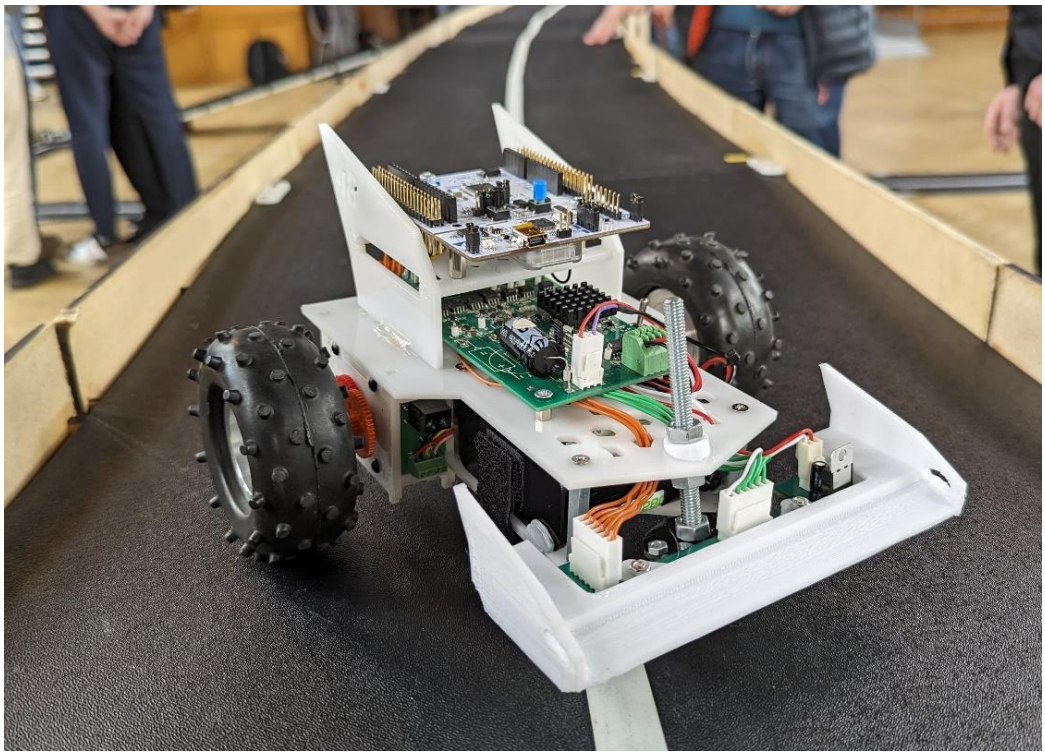

Figure 2.1: Buggy picture on the final race day

3. Final System Components Summary

3.1. Mechanical Components Design and Analysis

Mechanical Components are the basic infrastructure of the buggy. By analysing the load test data [1] of the buggy's motor and the final buggy’s weight, we added 10% safety margin to obtain the minimum required torque and optimal gear ratio for uphill driving, which determining the theoretical maximum speed. Based on this, we chose a 15:1 gearbox to balance speed and torque, ensuring the buggy can effectively handle inclines and flat surfaces.

In terms of chassis design, we considered factors such as the material's strength, tensile stress, and density, then selected acetal as the chassis material. We also optimized the overall weight distribution of the buggy through calculations, reducing the impact of the centre of gravity on buggy sway during turns. Additionally, a Finite

Element Analysis (FEA) simulation was conducted on the chassis to design appropriate thicknesses for different areas of the chassis, aiming to ensure that the acetal has sufficient strength to withstand normal loads while achieving the lightest possible chassis weight. These comprehensive measures have provided the buggy with an excellent mechanical performance.

3.2. Electronic Design and Implementation

Following investigations carried out during the sensor lab [2], an arrangement of six infrared TCRT (5000) sensors with 20mm spacings was chosen and as seen in figure 3.2.1. 100 ohm resistors were used on emitter and 10k ohm resistors were used on the transistor side. This configuration was selected based on the optimal line-detecting parameters to be implemented in a PID line following algorithm. The width of the PCB was designed to be 30mm to allow for the PCB to be as close to ground as possible without hitting the slope. A Darlington array is used switch the sensors on and off when needed. A 3.3V regulator is selected to ensure the analogue output voltage was compatible with the STM32 ADC's range, and to reduce noise on the analogue supply rail. Line sensor schematic is shown in figure 3.2.2.

These designs overall provide the most sensitivity, enabling the buggy to detect quickly and accurately turns, which allows for exceptional performance in the final race's downhill turns and spiral track sections.

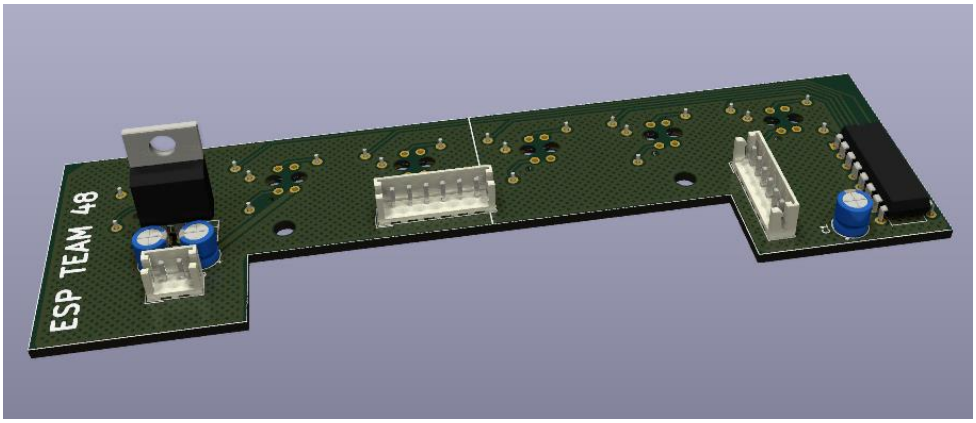

Figure 3.2.1: Line sensor PCB

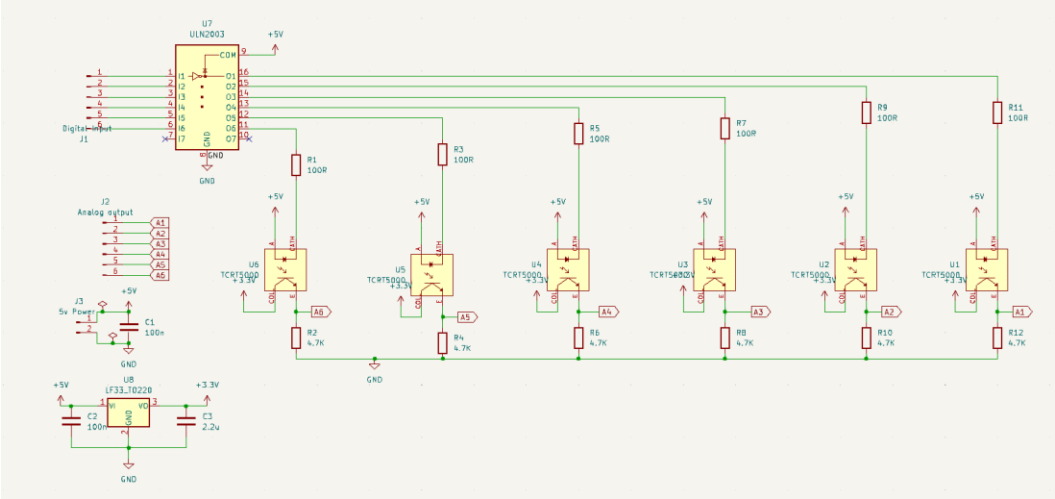

Figure 3.2.2: Line sensor schematic

3.3. Control Algorithm

Various methods were used to enable the buggy to handle diverse track conditions. These methods were tuned using real data from the buggy as shown in analysis of heats section.

A PI (proportional integral) controller was implemented to maintain a controlled regulated speed. The set speed is compared to the speed calculated from the encoder. This means that the PWM signal and therefore the average current is determined by the PI loop to achieve the target speed regardless of the buggy's position. Results from tests showed that the wheel changes were relatively stable during driving and there was no need to use the differential parameter D to achieve ideal results.

Line tracking used PD control. Based on the buggy test results, not using the integral parameter (I) helped reduce overshoot during turns and optimized lateral oscillation during straight-line movement. This allowed the buggy to maintain oscillation-free movement on straight tracks and to accurately recognize and follow track turns at high speeds. In the final race, this controller enabled the buggy to flawlessly navigate each curve at the set maximum speed.

Tuning the PID values was done using a combination of trial and error, and live tracking the individual proportional, integral and derivative values as seen in the “heats section”.

In extreme cases, such as when the buggy significantly deviated from the track due to high-speed sharp turns, additional bang-bang control was implemented. This control method is rapid because it does not consider overshoot and error, directly and effectively reorients the buggy when it strays significantly from the track. This decision was serving as a supplement to the line tracking algorithm. It was proved effective in the final race, the buggy was successfully and quickly reoriented after a sharp turn following a long descent.

3.4. Software

The software part is the decision-making centre. There are four main classes compiled in Mbed C++, which are Control, Sensor Array, Motor, and Bluetooth.

Control class includes the control algorithms previously mentioned. Additionally, a low-pass filter has been added to process the output voltage from sensor Array class, which reducing background noise from track reflections caused by environmental light, thus enhancing the accuracy of the buggy's track detection.

Sensor Array class, after receiving inputs from line sensors, uses a weighting formula to calculate and put these into one output voltage. This output voltage is then used to determine the error, which is sent to the Control class for further processing.

Motor class contains various control functions about motors. We have created a structure to control motor output, which abstracts two signals (direction and PWM) into a single value. This reduces the likelihood of errors, as it only needs to output a value greater than zero, and then the motor structure physically sends the expected signal to the motor driver board.

Bluetooth class is used for wireless communication with the buggy. This is achieved by connecting the HM-10 Bluetooth module as a serial device to the microcontroller. The HM-10 sends and receives bytes to and from the microcontroller through a serial connection. The Bluetooth terminal can not only send turn-around signals to the buggy but also receive real-time buggy speed data sent from the microcontroller. The

displayed real-time speed changes are beneficial for us to observe and adjust for the most suitable PID parameters.

3.5. Change and Overall System Realisation

A redesign of the chassis was carried out in December to improve the buggy's stability, turning and weight distribution. This change is shown in figure 3.5.1 Although this new design was working fine at low speeds, when the buggy would accelerate at startup or up a slope it caused it to tip over. Consequently, through recalculating the weight distribution, we adopted dual measures: moving the battery box forward to slightly shift the centre of gravity, and increasing the time required to reach maximum speed to reduce the initial acceleration rate. These changes successfully resolved the issue without affecting smooth turning.

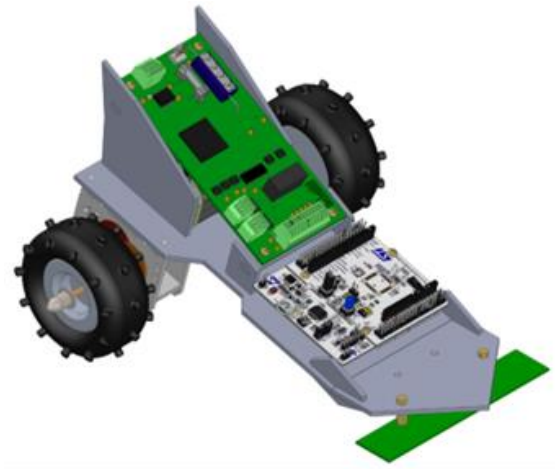

Figure 3.5.1 (a): Previous buggy design

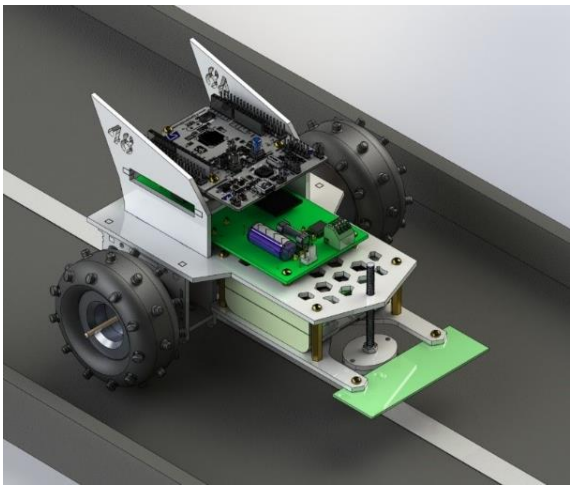

Figure 3.5.1 (b): Current Buggy design

Each part is designed with meticulous thought and shows outstanding performance. The mechanical component significantly lightens the buggy. The electronic component endows the buggy with the highest sensitivity for detection. The software's abstract structural design greatly reduces the time needed for debugging. The real-time speed display via Bluetooth allows us to continuously optimize the control algorithm, ultimately fine-tuning the most suitable PID parameter values, enabling the buggy to navigate all curves at top speed flawlessly. The independent functionality and synergistic interaction of these components were key to our buggy winning the championship in the final race.

4. Team Organisation and Planning

4.1. Aims and Objectives

Team organisation and planning is essential for proper working team to ensure that everyone knows at what stage the project is at so that everyone is on the same page as to what they must do in each week.

Team members regularly communicate with one another to update project objectives and whether they have been achieved yet. This allows every member to know the project status and able to move forward as one cohesive unit.

With an effective communication medium in place, the team has been able to plan and

organise the project successfully, resulting in high standard design reports, full marks in technical demonstrations and eventually winning the final race.

However, it is much easier to split this final objective into smaller more tangible objectives. As such, the milestones the team has set at the start of semester 2 were (in order):

1. Have the buggy run at a constant speed.
2. Have the buggy follow the line smoothly and at a controllable speed.
3. Have the buggy follow the line at the fastest possible speed whilst still maintaining control without too much oscillation.

These milestones have been discussed and shared within the team and agreed upon as necessary steps to achieve the final goal of winning the final race.

4.2. Changes to Plans and Timeline

Although the final goal and milestones never changed ever since it was mutually agreed upon, the planning aspect to achieve them have deviated slightly from the originally planned Gantt chart. Unlike the Gantt chart in the proposal report,

The chassis was re-designed in December to improve the stability of the buggy. however, this was done prior to manufacturing the chassis so it did not affect our overall schedule.

The team decided to design and print the sensor PCB during week 4 which, originally, the team planned to use a stripboard to test the software out [3] and revised the Gantt Chart of semester 2 as seen in figure4.1.1. The reason being that it was determined that testing out the software on the actual component would be more time efficient since the team would also be able to cut off the time that would be spent on soldering the stripboard that would be discarded later anyway. In addition to reducing time on the hardware, the risk of software being incompatible with the final sensor PCB has been reduced to zero if the team used the actual sensor PCB for the software testing.

The revised Gantt chart timeline is reasonable. Based on the requirements of the TD and team's decision, the technical assembly of the buggy is divided into five major sections, in sequence: buggy prototype, motor control, sensor implementation, steering control, and presentation. The timing for each sub-point has also been carefully considered, ensuring that there is sufficient time for testing after assembly to ensure that each part is correctly assembled. This can be seen as an improvement to the plan which was reflected in both TDA and TDB as our group achieved 100% marks in both technical demonstrations.

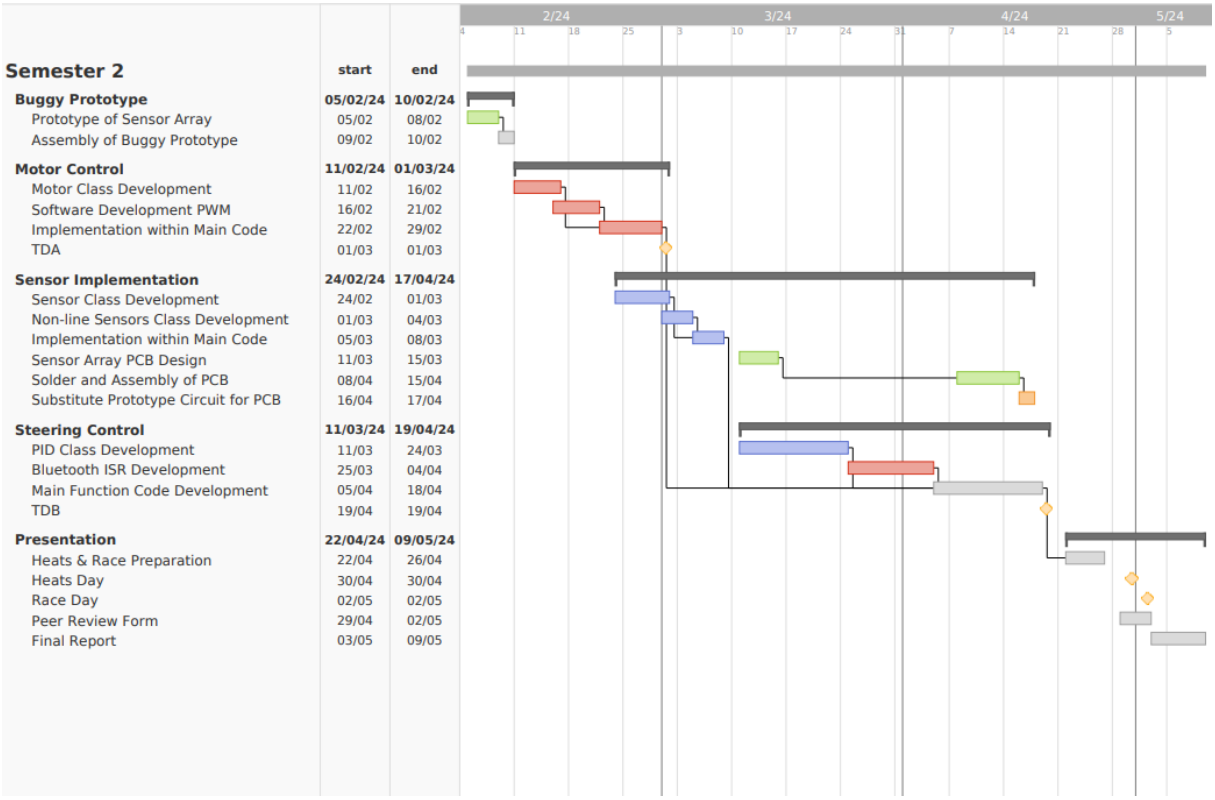

Figure 4.1.1: Revised Gantt Chart

4.3. Deliverables

As for the document type deliverables, there were the technical documentations as well as the proposal report, both of which were completed and submitted on time and achieved high marking grades indicating that our method and group working styles was proven to be effective. The only concrete part of our plan was to submit deliverables before the deadline, with a draft completed several days prior to allow for a team review. The working plan for each member is very flexible and accommodating for everyone's workload, team members regularly assist in each other's tasks to ensure everything is on track. Although as a group, we agreed that submitting reports on time is imperative, the submission of the DR2 report was slightly delayed which is something that we regret.

4.4. Unexpected Setbacks

Other drawbacks to the Buggy's progress occurred that wasn't initially accounted for. An error in importing the TCRT5000 footprint meant the original PCB was not compatible and needed to be redesigned. Because the PCB was designed well in advance this did not affect the overall progress. Issues with the encoder and gearbox were also identified and addressed early on. The original Gannt chart included buffers to account for unpredictable setbacks, which meant that the overall timeline was not affected. Therefore, if we were to redo the project, we would submit the DR2 report on time as well as carefully revise any CAD work before sending them to be manufactured. Apart from these incidents, there is not much else to improve on other than the technical part of the buggy such as increasing the speed even more while maintaining stability, however, that is beyond the scope of team organisation and planning.

4.5. Improvements in the Future

In the future, if the same team members were to work together again, a concise plan that is announced and quickly briefed to all team members at the start of each week could improve the organisation of the team which could result in even higher efficiency than it is in this project since there wouldn't be a time period (usually a day per week) at the start of each week where the team slowly get back into the flow of things.

5. Budget vs. Outturn

5.1. Components Summary

The basic components needed for the buggy includes various components which includes STM32F401RE Nucleo Board, a sensor array PCB, various screws and nuts, front and two rear wheels, batteries, two gearboxes with a motor each, a motor driver board, and a BLE module. The buggy also has a 3D printed part to cover the sensor array PCB but in terms of functionality, it is mostly for aesthetic purposes.

However, there are also a few components that are not included in the list which are used for prototyping which includes strip boards, various jumper wires and more. These components are not present in the final buggy, so they are not considered as the final cost of the product.

5.2. Budget Management

The budget provided for this project was £40. At the beginning of the project, we wanted to purchase an additional 9-axis packaged sensor chip, BNO080, which would have enabled the buggy to achieve more accurate attitude tracking and motion detection, but this would have meant that extra work must be done to implement it. As the project went on, it was decided that the extra sensors are not worth the extra effort of implementing them. Thus, we did not spend the budget on any extra sensors.

The team however, spent £11.81 on two 3.3V regulator used for the sensor array PCB and a Silicone lubricant for the gearboxes. This meant that minimal spending was made and thus the production cost was reduced. The 3D printed part is also included as a part of the budget spending. However, the printing service was provided for free by the university, but this will not be considered free for the final product costs. The £40 budget spending is shown in the table below:

Item	Quantity	Unit Price (£)	Cost (£)
LF33 (3.3V regulator)	2	1.32	2.64
Silicone lubricant	1	9.17	9.17
3D printed PLA	1	Free (University Provided)	0.00
		Total Cost (£)	**11.81**

Table 5.2.1: Budget Spending

5.3. Final Buggy MSRP

After a reasonable and delicate design and many experiments, the buggy achieved full marks in all the tests and won the first place in the final competition, breaking the record of previous years at the same time. During the running process, the buggy was able to track the route very accurately, while keeping minimal vibration throughout the whole

process, and reach a speed of 1.5 m per second. Not only is it the best in terms of functionality, but the buggy also has a design that meets the public's aesthetics while keeping the materials used frugal. All things considered, this is certainly a product that can catch the eyes of customers. After joining the market, the b2asic use of the buggy can be to participate in off-road vehicle races, provide schools with reference templates for related majors, etc. The buggy can also be used as a reference product for schools to use as a template for related majors.

Referring to the function of the trolley, it can also be used for research and production of automatic transport trolley in the workshop, which can help the company to improve the efficiency of transport and reduce the labour cost.

The base cost of the buggy is £226.99, and the price of each component is referenced in table 5.3.1. During the development of the trolley, the programming of instructions, and the design of the trolley including PCB design, form factor design and wiring diagram design require a lot of brain power and energy. After all considerations as well as calculations, a labour cost of £100 needs to be added to the base cost.

In summary, the proposed selling price of a buggy is £326.99. If the buggy were to go into production, additional costs would be incurred based on production requirements such as bespoke millwork, packaging costs, shipping costs, but this would be reduced to a more reasonable price due to mass production, for example, the labour costs and the costs incurred for the software development and design would be spread evenly over a number of buggies.

Item	Quantity	Unit price (£)	Cost (£)
Acetal Sheet	1	42.00	42.00
Front Wheel	1	4.01	4.01
Rubber Tyre	2	1.45	2.90
Motor	2	6.61	13.22
Gearbox Box	2	7.00	14.00
ULN2003	1	0.66	0.66
TCRT5000	6	0.72	4.32
AEAT-601BF06	2	21.40	42.80
HM-10	1	13.53	13.53
NUCLEO-F401RE	1	15.00	15.00
Controller Board	1	30.00	30.00
Battery Holder	1	2.28	2.28
Batteries	8	2.70	21.60
Insulation Tape	1	2.56	2.56
Resistors	12	0.10	1.20
Screws	8	0.08	0.64
Nuts	8	0.07	0.56
Standoffs	8	0.20	1.60
Wires	10	0.15	1.50
3D Printed PLA	1	5.00	5.00
glue	1	4.97	4.97
LF33 (3.3V regulator)	2	1.32	2.64
		Total Cost (£)	**226.99**

Table 5.3.1: Final buggy costs table

6. Analysis of Heats

Before the heats, the buggy is already working and fully tested, especially during TDB where each part of the track was tested individually. However, as a preparation for heats, individual components were tested to ensure that a wrong conclusion wasn't drawn during the testing or tuning of the buggy and knowing that it's not a problem with the hardware. Then, with the buggy known to be working perfectly, the code was developed and tested on it. With the working code and hardware, the buggy's digital low pass filter and PID constants were tuned with the help of Python scripts. Finally, the performance of the buggy in heats was discussed, and the best and worst features of the buggy were discussed.

6.1. Preparation

The crucial preparation for the heats is ensured by devising three stages of testing, consisting of "hardware, software, tuning". Before testing each stage, the preceding one is examined to ensure that any issues are not originating from the previous stage, as each stage is dependent on its predecessor. This method ensures that each stage undergoes testing after the completion of the preceding ones. The utility of the steps devised to test the buggy is particularly evident when troubleshooting the buggy's unexpected malfunctions and identifying the root cause of any issues.

With state machine approach of the code, a state was made to test each component of the buggy by sending various commands via Bluetooth and receiving responses either through Bluetooth or serial to the PC. This code, utilized for TDA and TDB.

However, reliance solely on Bluetooth commands for testing individual components can be laborious. Furthermore, Bluetooth modules are restricted to a baud rate of only 9600, posing a bottleneck if the entire status of the buggy is transmitted solely through Bluetooth, potentially leading to blocking code.

Consequently, an alternative method for the buggy to communicate its status is implemented by transmitting data through serial to a connected PC. This provides the overall status of the buggy, including measurements of each component. The serial data is then interpreted by a serial monitor program on the connected PC. The PC's serial setting is configured to a baud rate of 115200, significantly higher than that of the Bluetooth module, thereby reducing the likelihood of causing a blocking code.

This facilitates swift and straightforward debugging with a single code, as illustrated in the figure below. Although the frequency of status updates can be adjusted, typically 2Hz suffices. However, this method necessitates a USB cable connected to the buggy and is thus only employed when the buggy is stationary, and a PC is available.

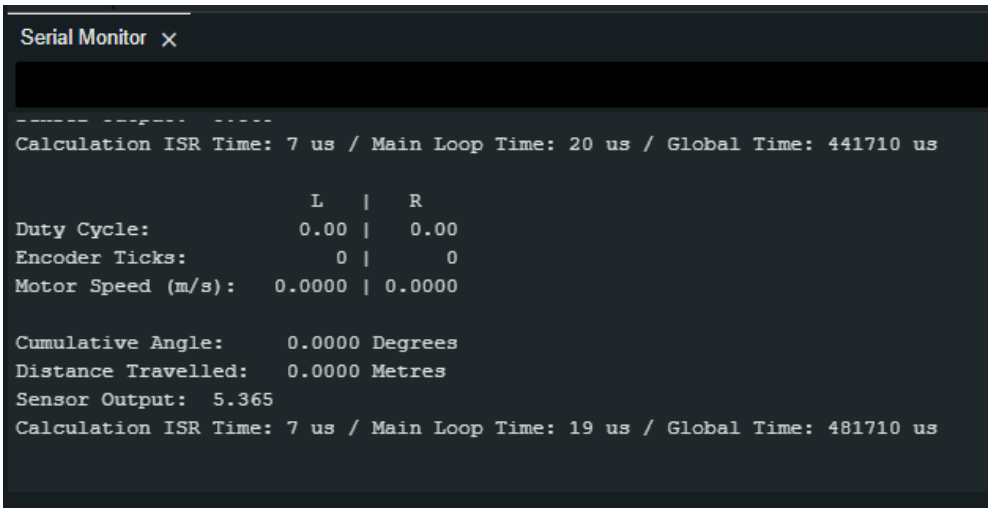

Figure 6.1.1: Arduino IDE used as serial monitor for troubleshooting.

After ensuring the functionality of every component, the subsequent step involved preparing all the algorithm in code required for the heats. The development of the code and algorithms is accomplished by simply appending another state to the existing code. Whenever a different logic or algorithm is required, the code seamlessly alters the buggy's state, initiating the execution of different algorithm.

For instance, a "U-turn" state contains the algorithm for executing a 180-degree turn, while a "line follow" state contains the logic dictating the buggy's movement to follow the white line. A switch-case statement within the continuous loop stores the code for different states, as depicted in the figure below. Notably, the buggy's state is named as "buggy_mode" in the code and the state cases are minimised for a clearer view.

```
239     // Buggy mode transition code
240     if (buggy_mode != prev_buggy_mode)
241     {
242         driver_board.set_enable(buggy_mode != inactive && ···
245         switch (buggy_mode)
246         {···
324         }
325         prev_buggy_mode = buggy_mode;
326     }
327
328     // Buggy mode continous logic
329     switch (buggy_mode)
330     {   ···
504     }
```

Figure 6.1.2: Switch case statements containing different algorithms.

Each state undergoes rigorous trial and error testing on the track multiple times until consistent success is achieved. Subsequently, with the logic of each state validated, the transition between states is tested to ensure smooth transitions. A comprehensive run incorporating all combinations of the buggy's states is then tested until consistent performance is attained.

Following the completion of these steps, the tuning process commences. With all the buggy's configurations functioning impeccably and fully established, only two types of tuning remain necessary: low-pass filter and PID tuning.

6.2. Tuning and Testing

Upon completing the configuration of the buggy, attention turns to specific areas requiring further tuning and testing. These include implementing a Low Pass Filter for Speed and sensor array output, as well as conducting PID tuning for both the motor and sensors. The specific code and formulas of the implementations are not discussed as it will involves a lot of technical details. A more detailed explanation the implementations is discussed in proposal document [3] and DR2 [2]. This section will instead discuss on the results of tuning and testing.

The implementation of a digital low-pass filter is pivotal in minimizing high-frequency noise present in the measurement of speed and line sensor data. This filter ensures that the signals utilized for decision-making remain accurate and reliable.

When focusing on tuning the Speed Low Pass Filter, the control of motor speed becomes a primary consideration. Utilizing a PID controller to regulate the PWM duty cycle sent to the motor driver board facilitates precise speed control. However, the efficacy of this control loop relies heavily on feedback from encoders, which measure speed by detecting differences in ticks and the time intervals between them.

Continuous measurement of speed at high update rates presents challenges, particularly when the time intervals between speed polls are too short for discernible differences in ticks to be detected. This phenomenon can result in erroneous speed readings of 0 despite the wheels' actual rotation, leading to noisy speed measurements.

To address this issue, a low-pass filter is introduced to smooth out the speed measurements and mitigate noise interference. However, implementing such a filter introduces a trade-off between response time and signal smoothness. Achieving an optimal balance is crucial, especially when a smoother output signal is desired without sacrificing responsiveness.

To validate the effectiveness of the digital low-pass filter, UART communication to the PC is employed to print the unfiltered values. These values are then stored and plotted using a Python script with matplotlib library, allowing for comprehensive analysis. This is shown in the figure below:

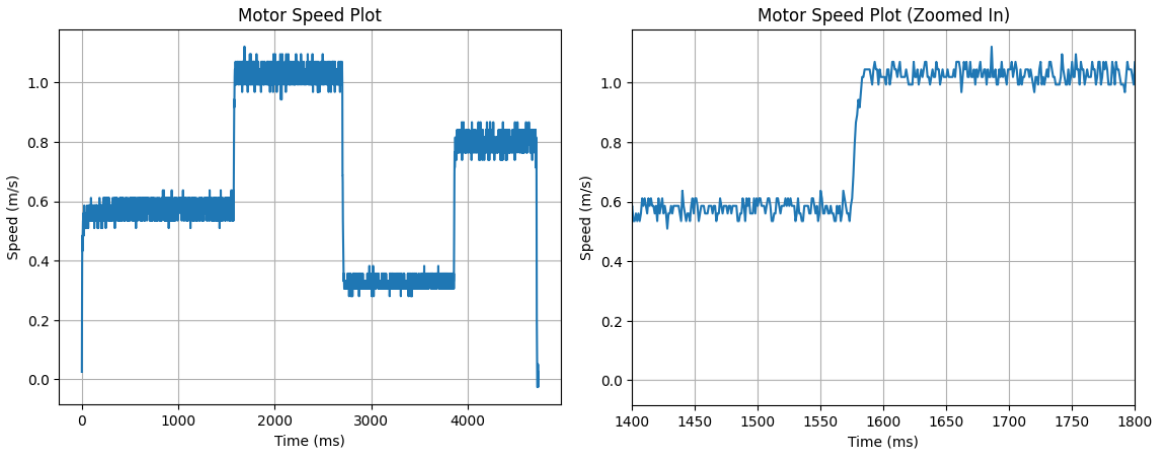

Figure 6.2.1: Raw speed value measured plotted against time.

With the collected data, the analysis is low pass filter can be done without having the buggy connected. The filter can be applied directly on the data and the results can be shown on the same plot for analysis. Thus, with this method, a low pass filter with

different cutoff Frequency, C_f was applied to the raw speed data and the two most notable results was C_f = 7 Hz and C_f = 2 Hz which are shown below.

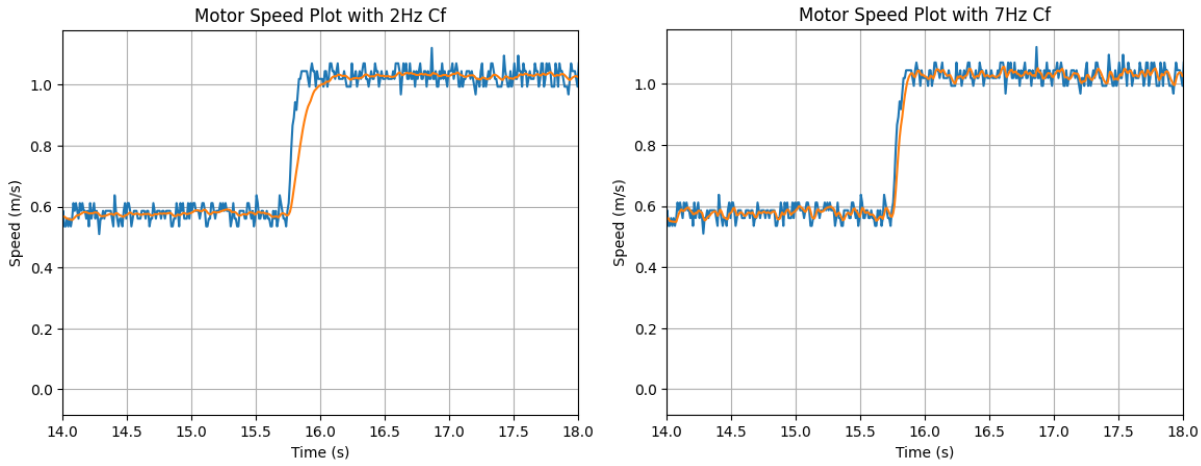

Figure 6.2.2: Low pass filter result (orange line)

At first, the low pass filter with 2 Hz cutoff frequency was implemented as it provides a smooth speed measurement. However, after thorough testing, it was found that it resulted with an overshoot in PID output as the response in the speed measurement was not fast enough. Therefore, it was changed to 7 Hz low pass filter. For the sensor array, tested with similar methodology, 7 Hz C_f was found as the best to filter the output of the sensor array.

Regarding PID tuning, it is important to note that the process is more complex than tuning a low pass filter due to the involvement of three constants: the proportional gain (KP), the integral gain (KI), and the derivative gain (KD). To simplify the tuning process and reduce complexity, the buggy utilizes only two components of the PID controller for each control system, as explained below.

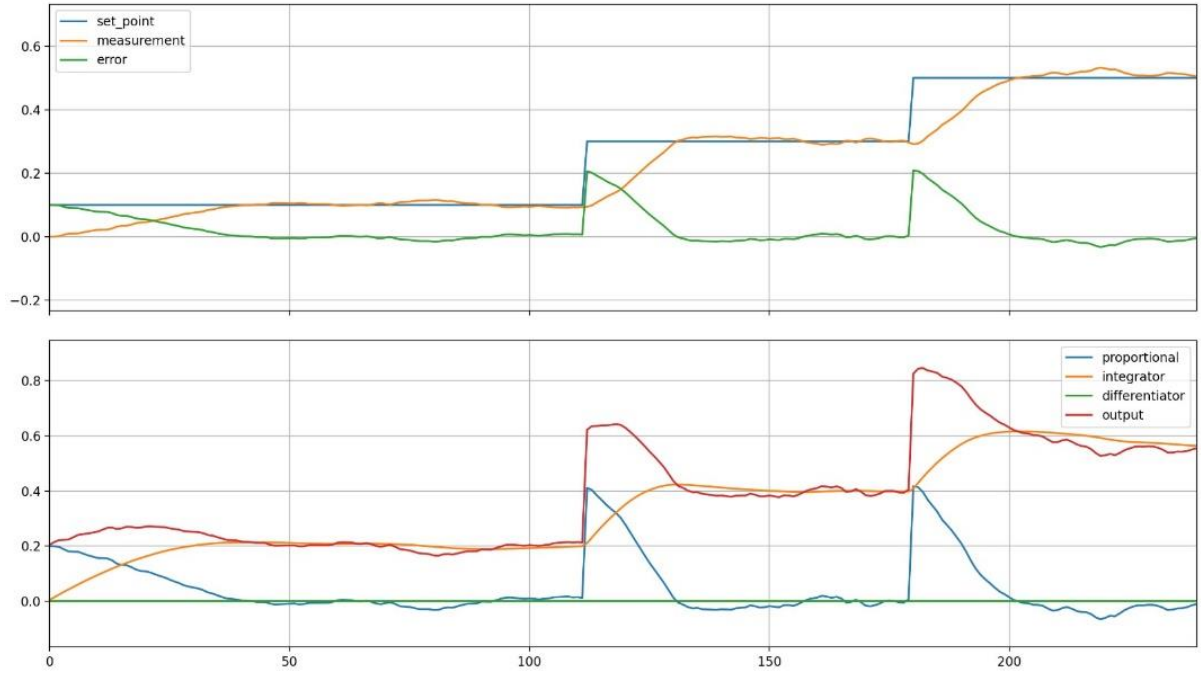

Figure 6.2.3: Example of motor speed control PID terms plotted against time.

For motor speed control, the derivative component (KD) was omitted because it is highly sensitive to noise and can cause instability. The proportional term (KP) provides an immediate response to errors, while the integral term (KI) eliminates steady-state errors, ensuring the motor speed remains close to the desired setpoint. Focusing on the PI terms improves stability and simplifies tuning.

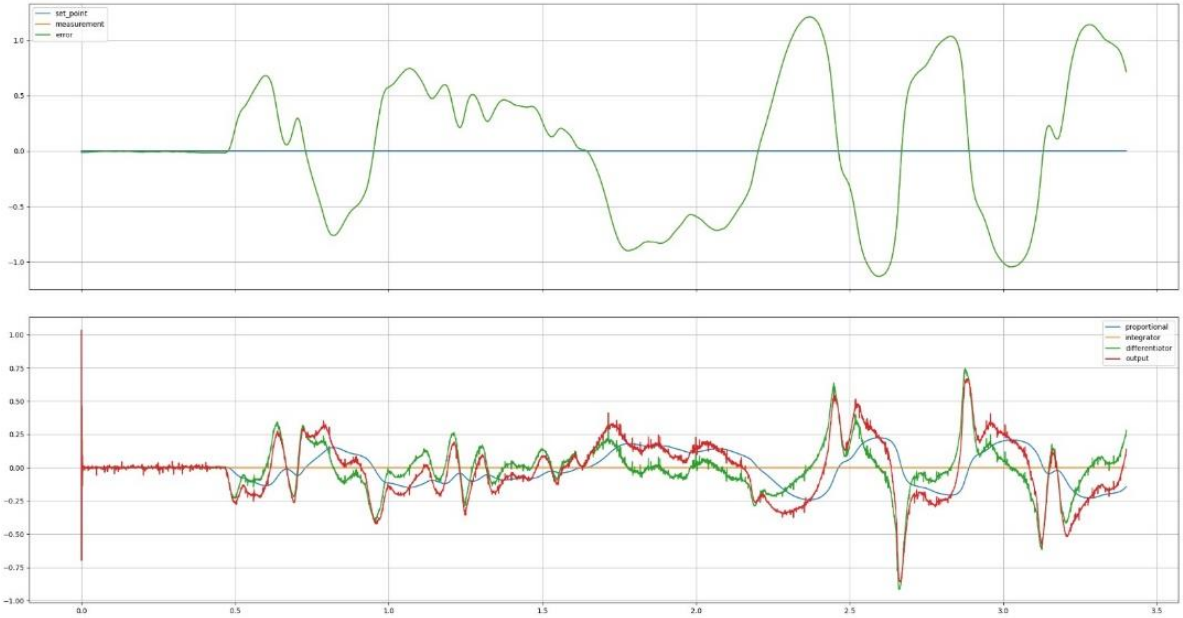

Figure 6.2.4: Example of sensor direction control PID terms plotted against time.

For the sensor array, the integral component (KI) was considered less crucial. The proportional term (KP) quickly corrects deviations from the line, while the derivative term (KD) anticipates future errors based on the rate of change of sensor readings. This combination allows swift reaction to change in the line's position and maintains accurate tracking. Omitting the integral term prevents error accumulation that could lead to overshooting or oscillations, making the sensor PID more responsive and effective.

The decision to use specific PID components for each control system was based on the requirements and characteristics of the motor speed control and sensor array. This approach balances complexity and effectiveness, enabling reliable and efficient operation of the buggy.

In tuning the PID constants, a trial-and-error approach was employed. Although the theory of control systems can provide valuable insights, this method has proven to be the most reliable, albeit the most involved. However, the task of tuning the PID constants was greatly simplified with the help of PID output terms logging. The technical implementation details of this feature are explained in the innovative features document submitted [4].

In essence, the buggy stores the PID output terms in a buffer. When the buggy receives a specific serial command, it prints out all the terms stored in the buffer. The collected data is then plotted using a Python script, with examples shown in figures 6.2.3 and 6.2.4. These plots provide valuable information on how each term affects the output, enabling more informed decisions during the tuning process.

By visualizing the impact of each PID term on the system's response, the trial-and-error approach becomes more guided and efficient. The plots allow for a better

understanding of how adjusting the proportional, integral, and derivative constants influences the buggy's behaviour. This insight facilitates the fine-tuning of the PID controller, leading to improved performance and stability.

6.3. Analysis of Buggy Performance in Heats

During the initial heats, the buggy's base speed was set at 2.0 m/s, a value thoroughly tested the day prior. However, an oversight occurred during testing regarding the buggy's automatic stopping, which needed to halt within 20 cm at the end of the run. This resulted in disqualification despite a swift completion time of around 9 seconds.

Recognizing the issue, we opted to decrease the buggy's speed to 1.2 m/s after the U-turn to reduce the buggy’s momentum during stop. This adjustment minimizes time loss since the initial segment of the track still permits a speed of 2.0 m/s. Additionally, a precautionary measure was implemented in code to further reduce the stopping distance by an additional 10 cm for added safety especially when the wheels are slipping during stopping which makes the measurement by the encoder invalid.

This strategy proved successful, as evidenced by our second run time of 10.86 seconds, securing us the fastest time on our track and the third fastest overall. While further testing of the stopping feature could have potentially yielded a better time, this achievement stands as a testament to the team's efforts.

6.4. Best and Worst Features

Initially, we considered adding extra sensors such as accelerometers and gyroscopes as it would improve the performance of the buggy. However, as we progressed, we discovered that we could achieve high performance without additional sensors. Consequently, decision was made to keep I the buggy simple without unnecessary extra sensors, as it is already challenging even with the existing hardware.

The buggy's primary success lies in its reliability in tasks such as line following during technical demonstrations. Several best key features have greatly contributed to this achievement, as outlined below, significantly aiding the buggy's development.

One of the best features of the buggy is the PID output terms logging. The team encountered a problem where tuning the PID became a guessing game. To address this issue, a PID logging system was developed to plot out each term of the PID, enabling informed decisions when adjusting the gains of each term. This feature proved to be highly useful, as it helped in achieving a really well tuned PID which greatly contributed in achieving a record-breaking time of 41.2 s in the final race.

However, the buggy's worst feature would be its weight distribution. During high-speed testing, the buggy tended to tip over during rapid accelerations, particularly during the initial start. This issue arose because the buggy was accelerating too quickly from a duty cycle of 0 to 1, applying the full voltage of the battery to the motor. Coupled with the buggy's design, where most of the weight is concentrated at the rear, this made it prone to tipping over. To mitigate this problem, a feature was added where the buggy was set to half the desired speed for half a second before reaching the full desired speed, ensuring a slower acceleration. Although this slightly reduced the buggy's overall speed, the impact was minimal. If the buggy were to be redesigned, it is recommended that the batteries be positioned slightly more towards the front to improve weight distribution and stability during acceleration.

7. References

[1] Group 48, Design Report 1, Embedded Systems Project, University of Manchester, 2023.

[2] Group 48, Design Report 2, Embedded Systems Project, University of Manchester, 2023.

[3] Group 48, Proposal Document, Embedded Systems Project, University of Manchester, 2024.

[4] Group 48, Innovative Features Document, Embedded Systems Project 2023, University of Manchester, 2023.

8. Appendix

Continuous Professional Development for each team member:

Continuing Professional Development Log

Name: *Mohamad Amierul Hakeem*

Current and recent CPD activity:

CPD Activity Title	Description	Dates	CPD Hours
Python Programming	I learned to use python and taken CS50 online courses to assist in learning	October 2023	150
Onshape	Learning other CAD software (Onshape) other than what im comfortable which is Solidworks	November 2023	20
PCB Design	Learned basics of kicad through kicad and designed my own pcb	November 2023	30

Planned and Future CPD activity:

CPD Activity Title	Description	Skills addressed	Dates
Python Libraries	Python has many useful libraries such as numpy and scikit-learn that would be very useful for data analysis in the future	Data analysis	June 2024
High speed PCB design	Learning how to design pcb with high speed signals for microcontrollers	PCB design	June 2024
Low level Embedded	Learning to use HAL libraries for stm32 because I think its fun to learn about how MCU works in lower level	Embedded programming	June 2024

Continuing Professional Development Log

Name: *Yanhua zheng*

Current and recent CPD activity:

CPD Activity Title	Description	Dates	CPD Hours
Learning about electricity	Learning about sensors, PCBs, etc. through video sites	November 2023	10
Learn inkscape	Going to youtube to learn how to use vector drawing tools in order to draw wiring diagrams.	March 2024	5
Strengthening of C++ capabilities	Strengthen C++ programming skills by understanding group members' code and ideas.	April 2024	10

Planned and Future CPD activity:

CPD Activity Title	Description	Skills addressed	Dates
Python advanced course	Take an online python course in the summer	Improving my python programming skills	May 2024
Teach myself MATLAB	Learn advanced functions of MATLAB through related materials	Can make me more proficient in using MATLAB for simulations	May 2024
Summer internships	Participate in a summer internship related to my major	Will enhance my practical skills and strengthen social functions	June 2024
Summer programme	Complete a project about Electronic Engineering AI Chip Topics and write a paper through a professor in a related discipline	Enhance the ability to analyse and solve practical problems and understand the cutting-edge applications of chip technology	June 2024

Continuing Professional Development Log

Name: *Jiexi Lu*

Current and recent CPD activity:

CPD Activity Title	Description	Dates	CPD Hours
Machine Learning Basics	Introduction to machine learning concepts and applications.	17/03/2024	20

Planned and Future CPD activity:

CPD Activity Title	Description	Skills addressed	Dates
Name of the CPD activity…	Describe the activity in brief…	What skills will the CPD activity enhance, or what skills-gap you have identified will it address…	When will this happen…
Advanced ML Techniques	Exploring advanced techniques in machine learning, including deep learning and neural networks.	Enhancing data modelling and algorithmic skills.	June 2024
Basics of Finite Element Analysis	Learning the fundamental principles of finite element methods and their applications in engineering problems.	Enhancing structural analysis and material performance analysis skills.	June 2024
Electromagnetic Field Analysis Basics	Introduction to the principles of electromagnetic field theory and its practical applications.	Enhancing skills in electromagnetic modelling and analysis.	June 2024

Continuing Professional Development Log

Name: *Teethabhumi Tripatarasit*

Current and recent CPD activity:

CPD Activity Title	Description	Dates	CPD Hours
GitHub	Setting up a GitHub account and learning how to use GitHub for the buggy code sharing and group development.	February 2024	5
Keil studio code import	Importing and using the imported code along with the existing code and seamlessly combine the 2	March 2024	10
Digital PID controls and implementation in CPP	Watched a YouTube video on how a PID controller can be implanted in CPP get a better understanding and idea on where to begin in the buggy software so that I can give useful insight to my teammates.	March 2024	10

Planned and Future CPD activity:

CPD Activity Title	Description	Skills addressed	Dates
Fourier, Laplace and Z transform	Continue my studies on signals and systems as I am unsatisfied with my current understanding and ability in this topic, especially the mathematical aspects on frequency domain and how they all interlink. The goal would be to get a deeper understanding and origin on how the formulas and equations came to be.	Help me get a better understanding on signals and systems in general up to near second nature level	June
Advance C programming	Continue to improve my skill and understanding of C language in addition to the basics to C programming from year 1 through YouTube tutorials and LeetCode.	My C programming skills and ability to find software solutions	June

Continuing Professional Development Log

Name: *Sarah El-Mahmoudi*

Current and recent CPD activity:

CPD Activity Title	Description	Dates	CPD Hours
Circuit and PCB design	Learning how to design a schematic and PCB for a sensor array using KiCAD software	February 2024	10
Soldering and Testing	Soldering a PCB, testing using lab equipment such a DMM and oscilloscopes, and fault finding	March 2024	5
Mbed platform	Learnt how to use Mbed development environment to program hardware	November 2023	30

Planned and Future CPD activity:

CPD Activity Title	Description	Skills addressed	Dates
Embedded programming	Further develop my skills in programming microcontrollers for different embedded applications. Such as using other serial communication protocols and wireless communications other than Bluetooth.	Programming C/C++, hardware design, microcontrollers	July 2024
Circuit design	Learn more about data acquisition and instrumentation methods including designing relevant circuits	DAQ, electronic circuit design,	October 2024
Machine learning models	Learn about using relevant libraries such as tensor flow to create machine learning models for image processing and image recognition	Machine learning, programming, image processing	August 2024