MANCHESTER 1824
The University of Manchester

Department of Electrical and Electronic Engineering

Embedded Systems Project 2020-21

Title: ESP Final Report

Group Number: 39

Group members:	ID Number	I confirm that this is the group's own work.
Mohammad Arab	10446769	☒
Ionut Buciuman	10455641	☒
Ahmad Huneiti	10528746	☒
Danial Murshed	10245339	☒
Yuwen Peng	10789777	☒
Liangchao WU	10573442	☒

Tutor: Wuqiang Yang

Date: 13/05/2021

Contents

1. Executive Summary 1

2. Introduction 2

3. Final system components summary 3

4. Team Organisation and Planning 9

5. Analysis of Heats (TD4) 12

6. References 15

1. Executive Summary

In order to finalise the embedded systems project, the final report is the last assessment of the overall project. The report mainly focuses on the overall work that has been done over the semester. By the end of the report, the main aspects of the report will be discussed such as the buggy assembling and components used in the final buggy version, team organisation and an explanation of the programming and testing part on the buggy.

To summarise the whole semester 2, the semester started with writing the proposal document which mainly focuses on what did the group accomplish in the last semester and how did the group organise work. The proposal documents also did focus on the goals of the group in the upcoming assessments. Technical demonstrations took place in the second semester, but due to the COVID-19 situation and on-campus restrictions both TD1 and TD2 took place online using MATLAB simulation. In TD1 and TD2, the main assessment was to achieve a line following robot that follows the white line with bends and detect certain obstacles like the walls.

The puzzlebot was distributed to each ESP group. Due to the ongoing COVID-19 situation, the puzzlebot was an alternative for the desired buggy that each ESP group worked on. The puzzlebot was used to test the programming abilities of each group on the final track. TD3 was the first test for the puzzlebot as each task required a C/C++ code that will be uploaded to test on the buggy to see if the buggy functioned as required.

TD4(Heats) was the last technical demonstration. The buggy should follow the white line on the track with right and left bends and an uphill which leads to a wall that the buggy should detect and take a turnaround to the downhill and follow the line until the line ends the buggy should stop. Correct implementation of code and wiring each component to its pin was crucial to accomplish the final task.

The final report is the last group assessment that each group should work on. It is important to point out that the final report summarises the work done throughout the second semester of the year. The main objective of the final report is the evaluation of how the overall project went and how did the buggy perform. An honest and clear evaluation of the project must be considered to build on the correct work done and fix the mistakes that occurred during the project planning and fabrication in future work.

2. Introduction

The last embedded systems project deliverable is the final report. The final report of the built white line following robot is considered as an evaluation and description of the system that has been built over the year. In the final report, discussion and analysis of the overall performance of the built buggy as shown in figure 1 should be stated.

The final report is separated into three main sections regardless of the executive summary and the introduction. The first section is the final system component. Moreover. The components section concentrates on the performance of the electronic, control, and software components used in the buggy and what differences have been made between the first and second semesters especially with the presence of the puzzlebot. Team organisation and planning is the second section of the final report. furthermore, the second section focuses on how the team members dealt with unexpected changes and if there were any communication problems between the group members. The team organisation section also discusses the overall project objectives. The last section is the analysis of the heats which mainly discusses the preparations of the final technical demonstration (TD4) from an evaluation point of view. In the TD4 analysis part, the performance of the buggy on track is evaluated where the most successful and worst features parts of the puzzlebot buggy will be considered.

Finally, the final report is an evaluation document that should describe and state the purpose of the project which is a line following robot as shown in Figure 1.

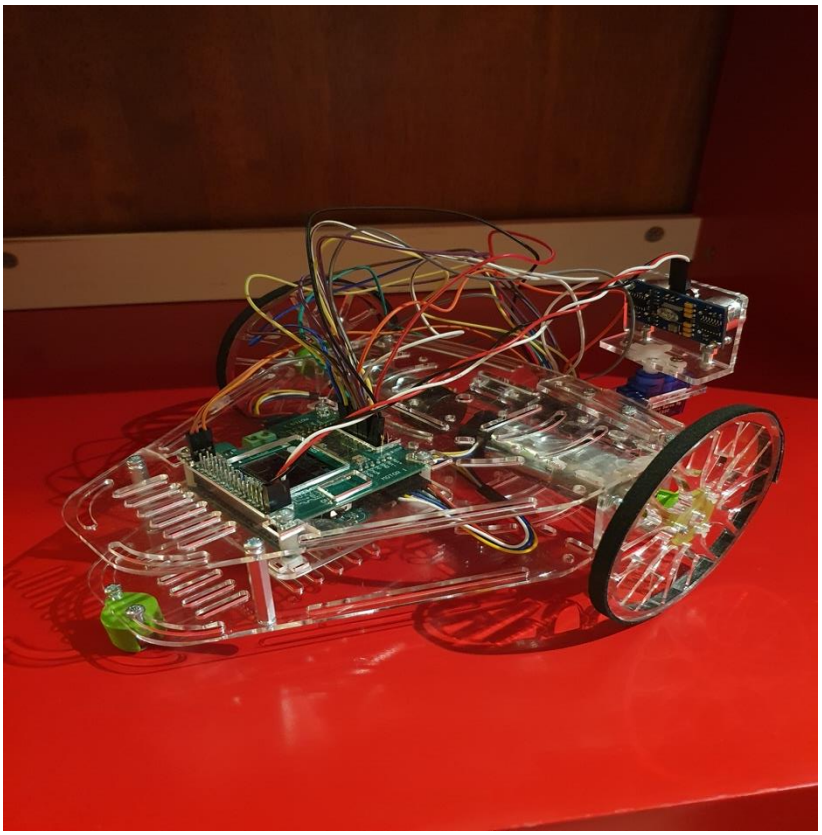

Figure 1: The line following robot

3. Final system components summary

Electronic components

Figure 2 and Figure 3 are pinout diagrams of puzzlebot control system. In the puzzlebot user manual it is shown how to assemble each component and how to connect each of them together to the baseboard.

The reflectance sensor is also an important component in the buggy. It connects to the analogue output. These reflectance sensors feature a linear array of infrared emitter/phototransistor pair modules which makes them well suited for applications that require detection of changes in reflectivity. [3]

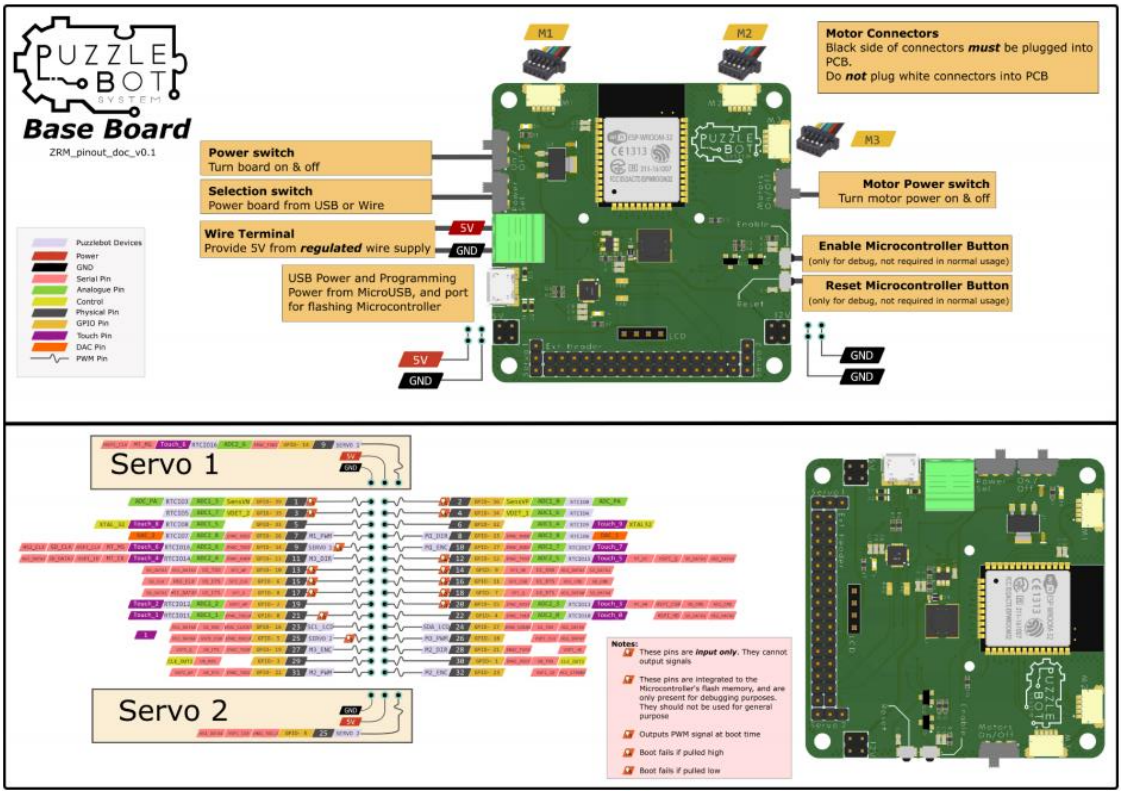

Figure 2: Puzzle-bot control module (bottom PCB)

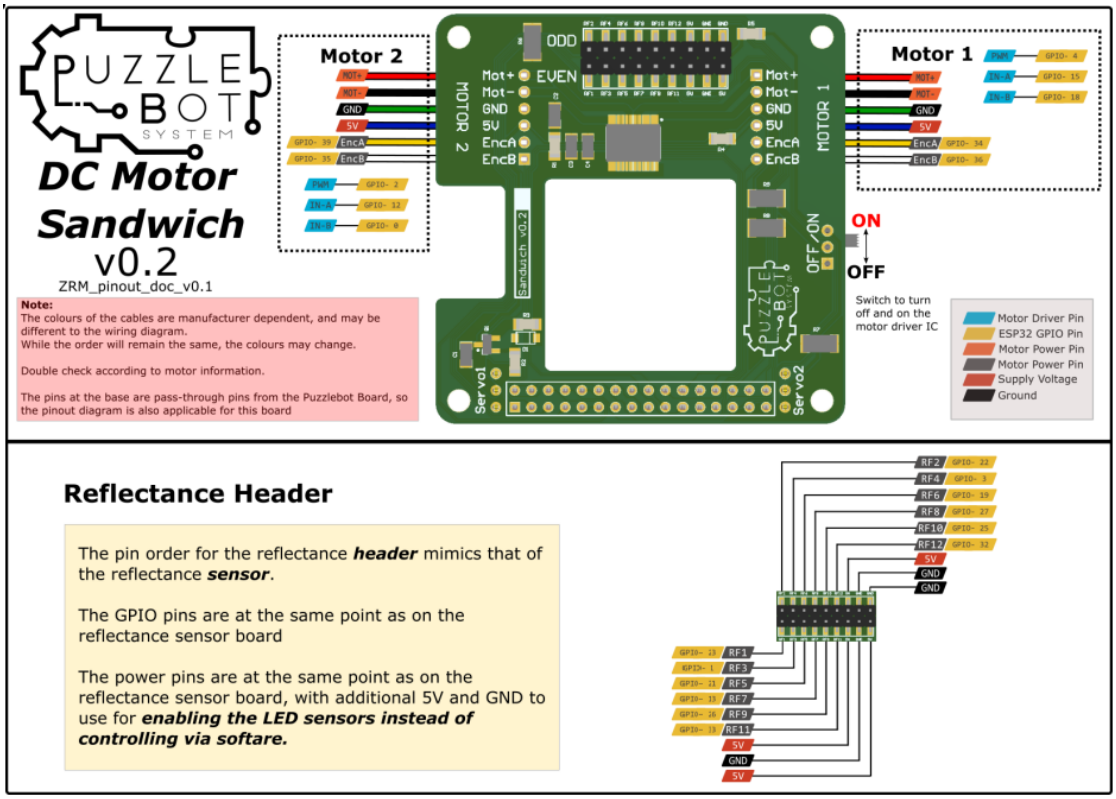

Figure 3: Puzzle-bot control module extension (top PCB)

The servo and sonar sensor connections are shown in Figure 4. When plugging the cable onto the sonar sensor, connect brown to GND, red to 5 V, and orange/yellow to SIG. the servo motor has three wires, each a different color (brown, red, and orange/yellow). The brown color wire is a ground wire (GND, battery negative terminal). The red one is connected to servo power (Vservo, battery positive terminal). The orange/yellow wire is the servo control signal line. It is required to check the specifications for the servo to determine the proper power supply voltage before connection, and take care to plug the servo into the device in the proper orientation (plugging it in backwards could break the servo or device) [1].
For interfacing ultrasound sensor, a single I/O pin is used to trigger an ultrasonic burst and then detect the echo of the return pulse. The sensor measures the time required for the echo return and returns this value to the microcontroller as a variable-width pulse via the same I/O pin. The sonar sensor module's GND pin connects to Vss, the 5 V pin connects to Vdd, and the SIG pin connects to I/O pin P15. This circuit is the same as Figure 4 shown below [2].

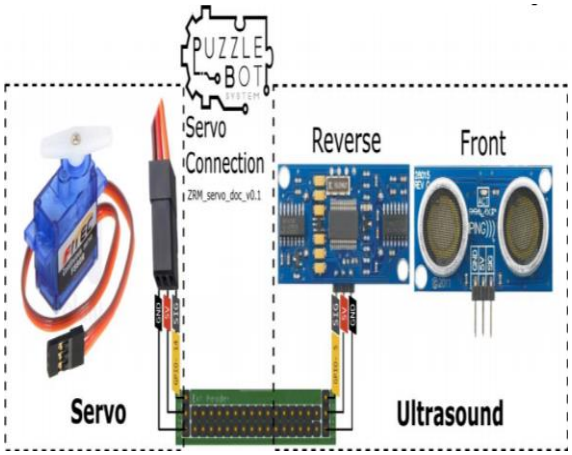

Figure 4: Servo and sonar connection

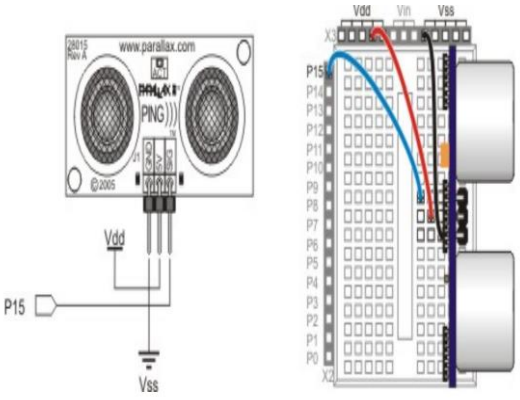

Figure 5: Sonar sensor schematic and wiring diagram

Control components

In this semester, at the very beginning, bang-bang control was used for the wheel control part of TD1 in the MATLAB codes. But when the bang-bang control method was applied to the mobile robot platform, the drawback of this sort of control emerged. Because bang-bang depended on the width of the hysteresis gap and inertia in the process, there was an oscillating error signal around the desired setpoint value. As a result, the buggy in the mobile platform zigzagged across the drawn line which was deliberately unstable, and the speed of the buggy was variable.

After noticing this serious problem, the bang-bang control was replaced by the PID control in MATLAB codes. A PID controller consists of three basic coefficients: proportional (K_P), integral (K_I) and derivative (K_D). The PID control in MATLAB is shown in Figure 6. The most difficult part in writing control codes was determining the most appropriate values of these three parameters which were varied to achieve an optimal system response reflecting by buggy running stability and detection sensitivity in the mobile platform. The result was an error value that was used in calculating proportional, integral, and derivative responses. By summing the three responses it was possible to obtain the output which is used as an input to control the mobile robot and keep the robot running stable.

```
if dt>my_alg('outer_sampling')   % execute code when desired sampling time is reached
    my_alg('outer_loop') = tic;

    %% Add your loop code here %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  refl = my_alg('reflectance');
  S=[my_alg('reflectance_raw')];
  my_alg('reaflectance_values')=[ my_alg('reaflectance_values'),refl ];
  Kp = 0.00005;
  Ki = 0.01;
  Kd = 0.0065;    %0.0065
  err = Kp*refl + my_alg('prev_refl') + Ki*refl*dt + Kd*(refl-my_alg('prev_refl'))/dt;
  my_alg('prev_refl') = refl;
```

Figure 6: MATLAB codes of PID control in TD3

However, for the Arduino code, the PD control replaced the PID control in control programming. The derivative action could provide damping to the system, so it could

make the vehicle move around more smoothly [4]. Compared with PID control, PD control was simpler and more practical applying to the buggy. When PD control was applied in Arduino codes and implemented in the buggy, it turned out that the buggy became more stable in speed as well as sensor detection and the response to the expected value/point was faster and more accurate.

```
float uR = Kpp*errorWR + Kdd*(errorWR-dR);
float uL = Kpp*errorWL + Kdd*(errorWL-dL);
dR=errorWR;
dL=errorWL;
```

Figure 7: Arduino code of PD control in TD4

Software components

In semester 2, it was required to use the simulation environment of the Puzzlebot platform in MATLAB for TD1 to TD4. Puzzle-bot was a good platform for developing the robot buggy and constructing a control system. Some basic controller libraries, sensors and actuators libraries had already constructed by the platform designer. Running situations (Figure 8) and barriers (lines, line breaks, walls and slopes etc.) could also be drawn in the simulation platform which enables the codes in MATLAB to ensure the buggy to solve all the situations in reality successfully. In the simulation environment, it programmed the microcontroller to set up the PWM outputs and implemented the functionality of line sensors and sonar sensors.

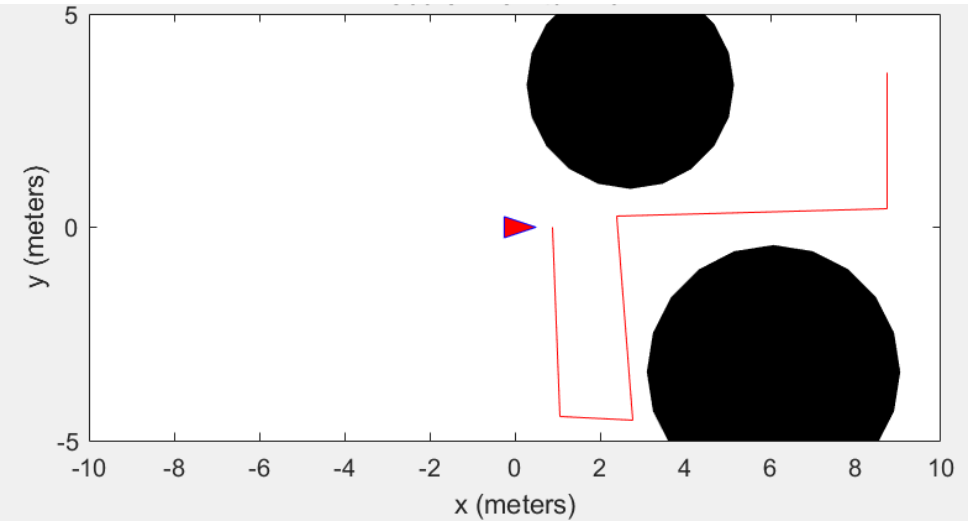

Figure 8: Environment in simulation platform

After simulating in MATLAB, it was also required to transform the MATLAB codes into Arduino codes for implementation in the real buggy. It was also one of the most difficult parts in TD3 and TD4 as even though the libraries for each component were given it was hard to learn. The Arduino codes needed to be adjusted over and over again for a better performance in the presentation and race according to how the real buggy behaved in the testing.

Overall, among these three parts, the control part worked the best. The performance of the buggy, such as speed, direction and sensor detection, was directly affected by the control part. The choice of control method was varied due to writing complexity,

practicability, and operating efficiency. Furthermore, the controlling coefficients were the key factors that determined the buggy's performance when following the line. As each coefficient had an influence on the sensitivity of changing the velocity and detecting the white line, it was necessary to change the control method repeatedly to find the most appropriate one and try each controlling coefficients over and over again, aiming for a buggy with brilliant behaviour for the presentations and the final race.

In terms of software, in S1 design, it has been planned to programme the STM32 microcontroller based on the MBed platform. Due to remote study, MBed was replaced by the Puzzle-bot system using MATLAB for the simulation and using Arduino to implement the program in the real buggy. Learning how to program in MATLAB and how to transfer MATLAB codes into Arduino codes caused some issues.

In the aspect of controller choosing, in S1 design, PID control was chosen as the most appropriate controller for the buggy. In S2 design, PID control was used in MATLAB simulation codes to ensure the mobile robot stable running and accurate detection. However, by many trials in the real robotic buggy, PD control replaced PID control which was not needed in Arduino codes for the reason that when the buggy was implemented with PD control, it became more stable and the speed was faster. PD control, which was used in the process where offset value is acceptable and should have some noise, seemed to be more practical and simpler in the physical buggy.

Furthermore, in S2 design, speed control was used to make sure that the speed was maintained at the desired speed that the programmer set up. This helped the buggy go up and go down the slope because without speed control the buggy would not have enough speed to move up the slope and also the speed would increase more than expected when the buggy went down the slope, as a result, the buggy would deviate and move out of the line. The codes in Arduino relating to speed control is shown in Figure 9.

```
float errorWR=0.5-(encR.getreading()/14);
float errorWL=0.5-(encL.getreading()/14);

 float uR = Kpp*errorWR + Kdd*(errorWR-dR);
 float uL = Kpp*errorWL + Kdd*(errorWL-dL);
 dR=errorWR;
 dL=errorWL;

 float uU=(uR+uL)/2;

float error1=uU + err;
float error2=uU - err;

if (error1<0){
  error1=0;
}
if (error2<0){
  error2=0;
}

Mr.MotorWrite(error1);
Ml.MotorWrite(error2);
```

Figure 9: Speed control codes

The puzzlebot offered lots of extra features which were out of the S1 design, such as a sonar sensor, much bigger wheels, a new reflectance sensor and, a different microcontroller.

The ultrasonic sensor provided an easy method of precise distance measurement. The sonar sensor measured the time required for the echo return and returns this value to the microcontroller as a variable-width pulse. And it could work in any lighting condition, making this a good choice to supplement infrared object detectors [2]. Because in S1 design, there were no obstacles, the sonar was not included in S1 design. However, it not only detected obstacles but also enabled the buggy to follow the line track and prevented the buggy from rushing to the wall of the racetrack. It seemed to be a greatly useful and safe component implemented in the robotic buggy.

The radius of wheels in S2 design is about 0.0505 meters which is much bigger than the wheels that were offered in S1 designing project. Though it may have caused increased instability than if the smaller wheels were used, the big wheels were better for the motor by reducing the required power for buggy to move the same distance. Moreover, the bigger wheels were made up of plastic which reduced the weight of the buggy as a result the buggy could have a faster speed.

The line detection sensors are located individually in a line in the front of the PCB board in S1 design. However, in S2 design, it offered a new reflectance sensor (Figure 10) which featured a linear array of infrared emitter/phototransistor pair modules enabling them well suited for applications that required detection of changes in reflectivity. This change in reflectivity could be due to a white line on a black background [3]. Moreover, the outputs were all independent which helped the buggy run along the white track in the right direction. This kind of reflectance sensor was more practical and simpler than the one chosen in S1.

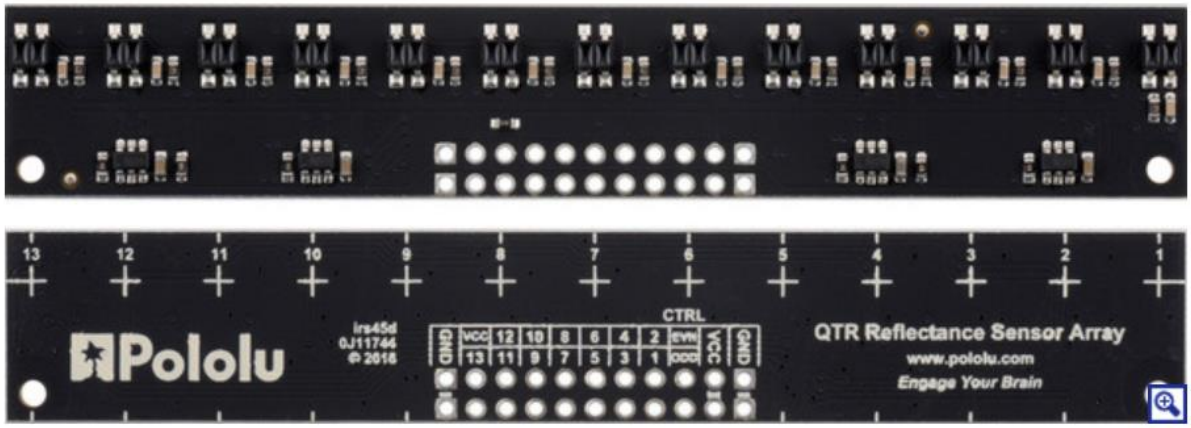

Figure 10: Reflectance sensor array, front back view [3]

4. Team Organisation and Planning

Table of deliverables and deadlines

Project	objectives	Deadline
Design report 1	• Select the appropriate gear size for the buggy based on the experimental data and calculation. • Research on the motor characterisation and analyse the laboratory data.	13/11/2020
Design report 2	• Learn and figure out the theoretical knowledge, including control system, software and etc. • Draw chassis outlines • Research on the line sensor and non-line sensor	11/12/2020
Proposal document	• State assumptions and summarise the design choices in S1, as well as the reasons for these designs. (This report serves as a summary and conclusion of S1 which is also the blueprint of S2 design.)	19/2/2021
TD1	• Apply PWM to motor control • Accomplish speed control and wheel control	5/3/2021
TD2	• Implement functionality for the line sensors and the sonar • Write the robustness of the algorithms to factors such as changes in the track, gaps in the line, and obstacles such as walls	19/3/2021
TD3	• Using speed control enable the buggy go up and down the slope • Control return and stop algorithm of the buggy • Transfer the MATLAB codes to Arduino codes	16/4/2021
TD4	• Mainly focus on implementing the Arduino codes in the real buggy	30/4/2021

In semester 1, considering the mass of the buggy, gearbox 2 was chosen to overcome the required torque and ensure a stable balance between the buggy's speed and torque to go over the incline and make the buggy move on the surface. The maximum velocity that the buggy was able to produce was 0.958 m/s on a flat surface and 0.361 m/s on the incline. In semester 2 TD1, speed can be controlled by using the code to change the PWM output. Speed control and wheel control can be accomplished by applying different PWM signals to the wheel.

For the control algorithm, in the beginning the PID control algorithm was chosen as it helps reduce the overshoot and decrease the oscillations around set points. However, when it was tested in MATLAB the group wanted to first change it to bang bang control. But it was suggested that bang bang control will make less overall stability of the buggy which led the code in MATLAB to be changed back to a PID control. When implementing the Arduino codes in the real buggy, PD control seems to be more efficient and can make the buggy more stable as it has simpler programs that can run quicker.

The sensors are crucial components for the ideal buggy design therefore, choosing suitable sensors will have a great effect on the project. In semester1 the TCRT5000 sensor was chosen to be the most suitable sensor for the buggy design as it contains a daylight filter and has a good sensitivity when exposed to the white line. As in TD2, it requires implementing functionality for the line sensors and the sonar (exteroceptive sensors). Therefore, the sonar is added in the front. It is an ultrasonic sensor for distance which is not considered before to detect the wall that is very important for the turnaround point. For the power of the buggy, the power bank is finally chosen instead of classical batteries as it is easier to work with.

In semester 2, our group used the same project plan as the one used in semester 1 and that is every two people focus on one task and help each other. More than one meeting for all of the group members was usually scheduled to update project progress in real-time. As soon as work was done it would be sent to Dropbox to make sure that all group members are able to access it at any time. This method worked well for report writing, but for the technical demonstrations it was realized that some tasks are continuous and dependent on one another. Therefore, separating the tasks equally between the group members was not as effective and efficient. In other words, more cooperation and coordination was needed between the team members. Also, each group member has his own skillset and preferred to work on specific tasks. Therefore, it was important to communicate with all team members before assigning the tasks to each member to make sure that everyone is comfortable with the work assigned. Not all group members participated in the coding of the buggy as it was not efficient to do so. Instead, the other members that were not coding were busy researching the types of control algorithms that may be implemented on the buggy. After performing an extensive research, the ideas are shared between the members and a certain type of action is agreed upon by everyone.

During the second semester the team had multiple objectives, and most of them were achieved on time and some not. In the first part of the semester, during the MATLAB coding, everything went as expected, because everyone had the software and could write code and test it very easily. However, some problems started to appear in preparation for TD3. The group members didn't manage their time well

during the easter holiday and this resulted in the group not having much time left to transfer the programs from MATLAB to Arduino. As a result, the group's grade for TD3 was not as expected and was certainly the lowest of all technical demonstrations. For TD4, there were some issues with mechanical aspects of the puzzlebot as the wheels were not fixed properly and this caused some delays in creating the necessary program and the group was not able to attend the testing slot. Fortunately, the group had planned for such events to occur and a backup plan was developed to mitigate the effects of such delays. At the end the robot was able to perform every task required except for the turn-around point.

Communication between group members is vital in group projects such as this one and since this year was done remotely not all group members were available in the same place and at the same time which meant that the time difference between the members living in different areas was one of the biggest challenges. Constant communication and continuous updates was a key factor in developing an efficient method of communication between the team members. To achieve such communication a group chat was created on Wechat. Also, a shared folder was created on Dropbox to allow filesharing between the team members. By communicating using the mentioned platforms the group was able to meet all the deadlines involved in semester 1 and 2. A Gantt chart was also created to make sure that the group's progress was being continuously tracked and updated.

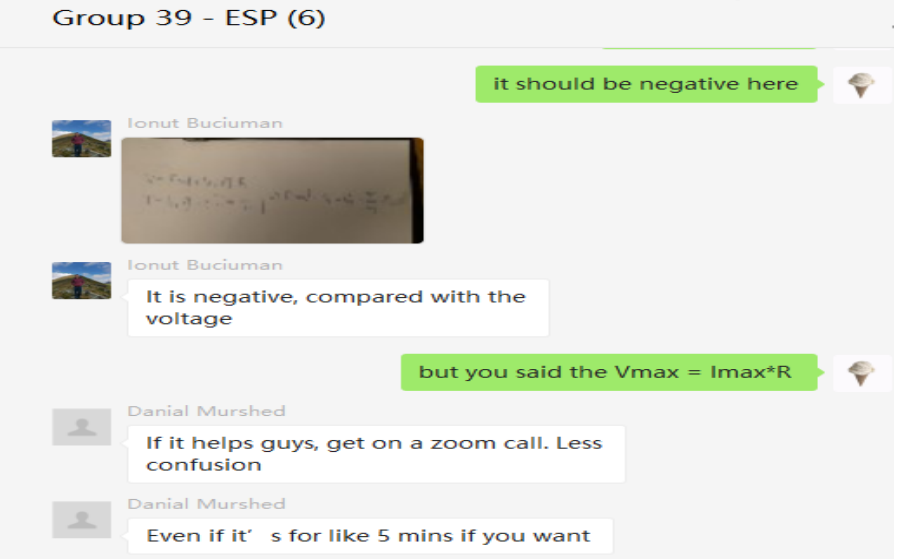

Figure 11: Screenshot of group chat on WeChat

In semester 2, the platform for group meetings was changed from Zoom to Discord which seemed to cause some sort of confusion and the communication of information sometimes appeared to be messy. For example, sometimes after the meeting instructions like the division of the tasks among the members are sent on Discord instead of WeChat which can be a problem when trying to look for information and makes it inefficient. Another way to improve cooperation is having more frequent group meetings. One group meeting before the deadline is commonly held by all the group members which is sometimes not enough.

Unexpected situations are inevitable. This year, the final draft was always submitted one day before the deadline and luckily it worked throughout the whole year without any problems, but it should be considered to arrange more time for unexpected problems that may arise.

5. Analysis of Heats (TD4)

Technical demonstration 4, also known as the heats, consists of a track assembled in the main campus that includes a start point, a ramp, a wall at the top of the ramp that is also the turn-around point, and an endpoint. The heat was the final practical assessment and was basically a combination of all the tasks that the group has worked on in the previous technical demonstrations. It was important to make use of all the buggy's features such as the encoders, sonar sensor, and reflectance sensor. To prepare for the heats the group had to build a track that was similar to the one on campus that will be used in the assessment. Creating the bends and the track on which the buggy had to move was quite simple; however, building a ramp was not as simple. To make use of time the group decided to test the buggy without the ramp and only see if it follows the line at first. An adequate amount of time was spent on making sure the buggy follows the line properly and that it quickly responds to any changes in the track.

As previously mentioned, the heats contained a combination of all the previous tasks which meant that all group members had to put in some work to make sure that the buggy is ready before the submission deadline. It was difficult for all group members to work together simultaneously and therefore as a group, it was decided that the work should be split into two groups of three which was convenient since each group had access to two buggies. This did not mean that the groups worked on different tasks but instead worked on the same tasks and this was done to make sure that the buggy operates in the best possible way as this meant that every idea was tested. This may not seem like the best method as not enough time was left to implement the turn-around, but the group certainly thought it would produce the best outcome as every other task was successfully implemented to the finest of the buggy's ability. Having access to the same puzzlebot as the one used for the assessment helped the group understand how to program the buggy in the most optimum way. Also, the MATLAB simulation played a major role in helping the group understand the main aspects of the buggy such as the PID control and speed control. An important thing to look out for was to make sure that the buggy is assembled exactly in the same way as the buggy on the main campus because even if one pin is connected differently the buggy will not operate as intended.

Figure 12: Group 39 test track

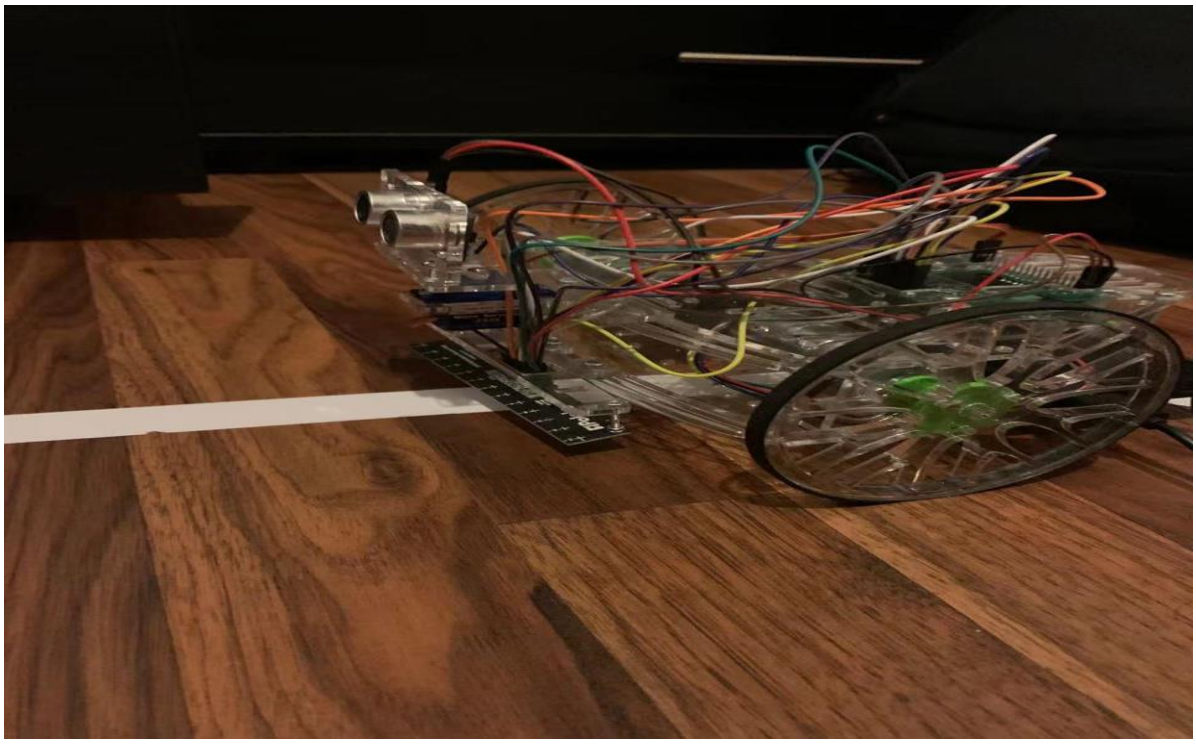

Figure 13: Puzzlebot moving on the track

Figures 12 and 13 demonstrate the track used by the group to test the buggy. As is shown the track consists of white tape attached to a brown floor. According to the handbook, the surface underneath the white tape should be black [4] and therefore the reflectance values for the sensor will be different than when the buggy is tested on the track in the main campus and had to be calibrated accordingly. It can be seen from figure 12 that the track built by the group contains bends that are similar to the ones in the track that will be used for the assessment.

During the first attempt of the heats the buggy's performance was much better than expected. The buggy followed the line accurately and was able to move up and down the slope all while still following the white line and it stopped when the line was not detected. There was only one flaw in the firmware submitted and that was the buggy could not perform the turn-around when faced by the wall. The reason behind this was that the sonar sensor was not implemented as the group did not have enough time to add it. The group's decision was to submit a working code for the first attempt that all group members were confident would perform most of the tasks instead of risking it and trying to add the sonar sensor which could have a bad effect on the whole program and potentially stop it from doing the other tasks. The group decided to use the same code for the second attempt but to perform the tasks separately to allow the buggy to reach the endpoint without having to implement the turn-around. Surprisingly even though the same code was used; however, this time the buggy did not detect the line and could not move up the slope. The second attempt was about 4 hours after the first attempt meaning that the time difference between the first and second attempt may explain why the buggy did not detect the line. The value of the reflectance sensor was manually adjusted according to the environment that was present during the testing slots; therefore, if in the second attempt the lighting in the room slightly changed then the reflectance sensor algorithm might have been affected. Determining the minimum value of reflectance that will be read by the sensor was quite challenging since it is dependent on the environment surrounding the buggy such as the surface below the buggy and light exposure. Therefore, the minimum value when the buggy was being tested during the first attempt was not the same as when it was being tested during the second attempt. To prevent this from happening again a calibration function may be added to the code which allows the reflectance values to be updated continuously according to the environment surrounding the buggy.

Fortunately, each group had a testing slot before the heats that could be used to test the group's buggy on the same track that will be used during the assessment. It was very important for the group to attend its slot as it helped the group identify the buggy's strengths and weaknesses. For example, the buggy was not following the line when it was tested during the testing slot and the reason was that the reflectance sensors were calibrated according to the surface that was used in the track that the group built at home. As a result, the sensors had to be calibrated again to make sure that the buggy successfully follows the line during the assessment. Also, since the group did not build a ramp the speed that was required by the buggy to move up the slope was unknown, and a random speed was chosen at first. After testing the buggy, it was clear that the speed chosen was not enough to move the buggy up the slope and had to be increased. After increasing the speed, the buggy was able to move up the slope but started to deviate whenever it went down the slope. A speed control was then programmed using the motor encoders which allowed the buggy to maintain a constant desired speed when moving down the slope to prevent it from

moving out of the line. To make sure that the buggy follows the line accurately the PD controller used had to be fine-tuned. The group had an idea about what values to use for the proportional gain (K_p) and the derivate term (K_d) since it was previously calculated in the MATLAB simulation. However, the previous values had to be adjusted slightly by trial and error as the environment has changed and the buggy has become more susceptible to noise from the surroundings.

One of the most successful features of the buggy was its response to the white line. The buggy was able to follow the line as accurate as possible while still maintaining a relatively fast response to sudden changes in the track such as sharp turns. Another successful feature was the ability to change the distance between the line sensors and the ground simply by adjusting the positions of the screws. This feature may have not been very important for the heats race but is certainly helpful for other applications when the distance has to be changed to avoid damaging the line sensor board. One feature that the group considered to be a weakness was that the buggy did not have a battery pack in which the power bank could be placed and held tightly. The buggy that was designed by the group in semester 1 [5] contained a battery support that was placed under the chassis unlike the puzzlebot which did not have one and therefore it was difficult to attach the power bank and prevent it from moving as the buggy accelerated.

6. References

[1] Pololu.com. 2021. Pololu - FEETECH FS90 Micro Servo. [online] Available at: <https://www.pololu.com/product/2818> [Accessed 13 May 2021].

[2] Parallax. 2021. PING))) Ultrasonic Distance Sensor - Parallax. [online] Available at: <https://www.parallax.com/product/ping-ultrasonic-distance-sensor/> [Accessed 13 May 2021].

[3] Pololu.com. 2021. Pololu - QTR-MD-13RC Reflectance Sensor Array: 13-Channel, 8mm Pitch, RC Output. [online] Available at: <https://www.pololu.com/product/4153> [Accessed 13 May 2021].

[4] Marsh, L., Embedded Systems Project Technical Handbook 2020-21. University of Manchester, UK, 2020.

[5] “Design Report 2 - Technical Characterisation - Group 39” University Of Manchester, Manchester, 2020.