The A-maze-ing Race Project Report

CG1111A Team B02-S4-T1

Lee Hao Zhe

Li Mengyan

Li XiaoNa

Li Zekuan

Table of Contents

1. Overall Algorithm and Implementation

1.1 Overall Algorithm

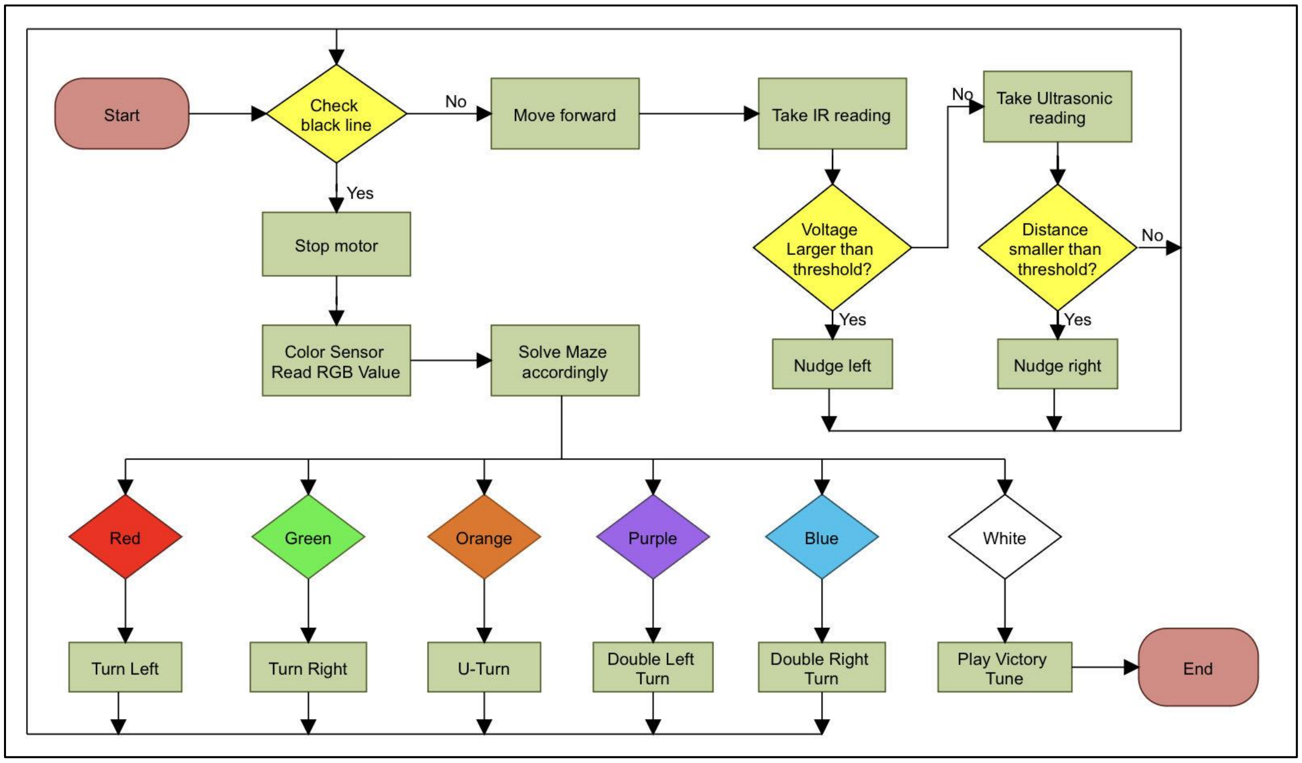


Figure 1: Overall algorithm

To navigate through the maze, we employed a combination of ultrasonic and infrared sensors to maintain the mBot's straight trajectory. The line sensor is utilized for detecting black lines, while the color sensor is employed to identify the color of the surface beneath the mBot.

While the mBot is in motion, the ultrasonic sensor and IR sensor continuously assess the distances from both walls. If the ultrasonic sensor identifies the mBot as being excessively close to the left wall, it adjusts the mBot towards the right. Similarly, when the IR sensor on the right side detects a voltage reading difference beyond the specified threshold, signifying proximity to the right wall, the mBot makes a corrective movement to the left.

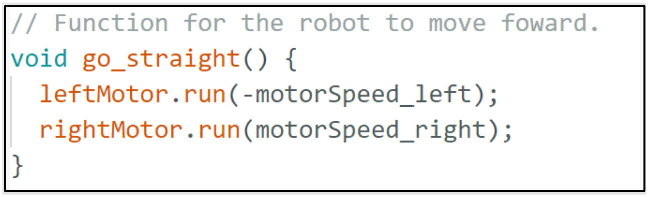
Upon detection of a black line by the line detector, the mBot will come to a halt. Subsequently, the colour sensor activates the LEDs sequentially (red, green, blue) for 10 iterations each, during which the LDR measures the reflected light intensity off the paper. The colour determination is then derived from the intensity values obtained by the LDR, guiding the mBot in making appropriate turns.

1.2 Basic Movement of the mBot

The fundamental movement functionality is facilitated by two PMDC motors situated beneath the chassis. The mBot is propelled by the two rear wheels driven by these motors, complemented by a single unpowered front pivot wheel. Basic movement operations essential for maze traversal include moving straight, executing left or right turns, and nudging left or right. More intricate movements, like U-turns, are accomplished through various combinations of these fundamental movements. The behaviour of these movements is influenced by the sensors, which will be elaborated upon in the subsequent sections.

1.2.1 Going Straight

To commence forward motion, the two motors receive equal power, and the motor speed is configured to 220 out of 255 (4.3V), considering the possibility of nudging adjustments later. The wheels rotate in opposing directions due to their opposite orientations.



```
// Function for the robot to move foward.
void go_straight() {
  leftMotor.run(-motorSpeed_left);
  rightMotor.run(motorSpeed_right);
}
```

Figure 2: Function for moving forward

1.2.2 Turns

For basic turns, we programmed the mBot to rotate its two motors in the same direction during the turn. Specifically, both motors rotate clockwise when turning left and anticlockwise when turning right.

In the case of more complex turns, involving combinations of left or right turns and straight movement, the challenge was to precisely calibrate the duration of each turn. The goal was to ensure that the robot completes a 180-degree rotation and faces the correct direction after the turn without colliding with the wall. Our team successfully addressed this challenge through the process of trial and error, gradually adjusting the motor power duration to guarantee the robot's correct orientation following each turn.

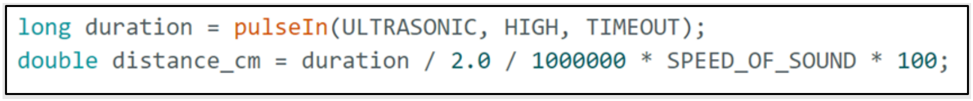
1.2.3 Nudging

To maintain the robot’s straight trajectory, we designed it to nudge whenever it gets too close to a wall. When nudging to the left, the speed of the left wheel briefly decreases to 185, while the right wheel's speed increases to 255. Similarly, for nudging to the right, the method remains consistent, but the roles are reversed. The right wheel will now slow down to 185 and the left wheel will accelerate to 255 for a short duration.

1.3 Keeping the Path of the mBot Straight

1.3.1 Ultrasonic sensor

The ultrasonic sensor is mounted on the left inner side of our mBot and detects the distance of the wall from the left side of the mBot. The sensor emits an ultrasonic wave and measures the time taken for the wave to be bounced back from the wall and gets detected by the receiver. The distance is then computed using the formula between the distance and the duration.



```
long duration = pulseIn(ULTRASONIC, HIGH, TIMEOUT);
double distance_cm = duration / 2.0 / 1000000 * SPEED_OF_SOUND * 100;
```

Figure 3: Formula for distance

The distance will then be compared to our set value - 8cm. If the distance is less than 8cm, the mBot will nudge right and prevent itself from colliding with the walls on its left.

1.3.2 InfraRed (IR) Proximity Sensor

To navigate the path of mBot, an InfraRed (IR) proximity sensor is used. The diagrams below (Figure 4 and 5) show the design of circuit of IR sensor.

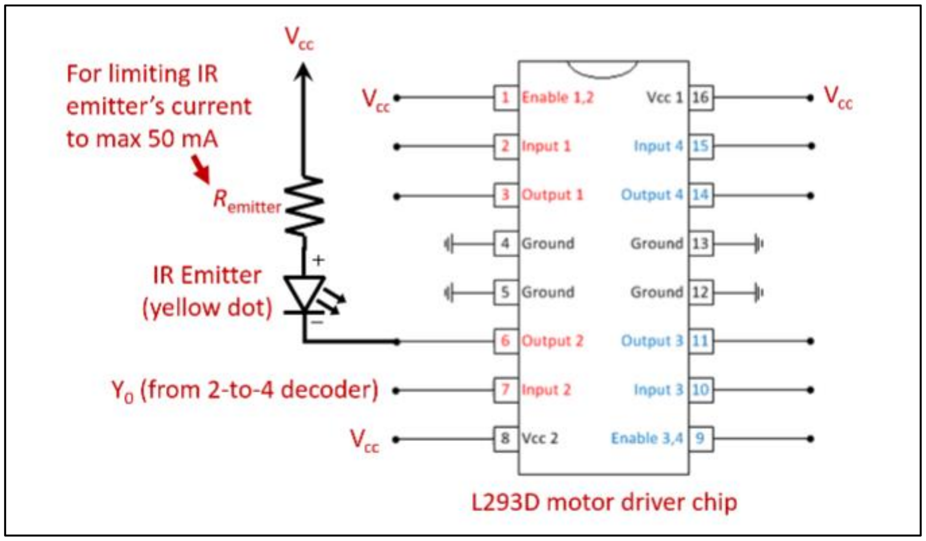


Figure 4: IR emitter

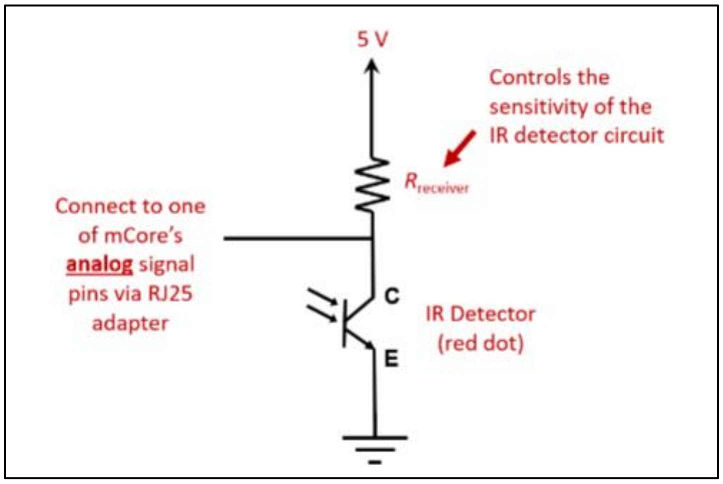

5 V
Controls the sensitivity of the IR detector circuit
$R_{receiver}$
Connect to one of mCore's analog signal pins via RJ25 adapter
C
E
IR Detector (red dot)

Figure 5: IR detector

The resistance values that we chose for the IR emitter and IR detector are 150Ω and 12kΩ respectively. This combination ensures maximized balance of sensitivity and consistency of IR sensor. The IR sensor is mounted on the right side of the mBot (Figure 6) to detect the distance between the right wall and the mBot. The circuit is designed and made to optimize neatness and stability of the wiring. The negative polarity (black wire) of IR emitter is connected to the ground pin of RJ25. The circuit is powered up by connecting the L293D chip to the VCC pin of RJ25 (red wire). The green wire crossing IR emitter and IR detector is not part of the circuit but serves as a securing purpose to ensure stability of the sensor.

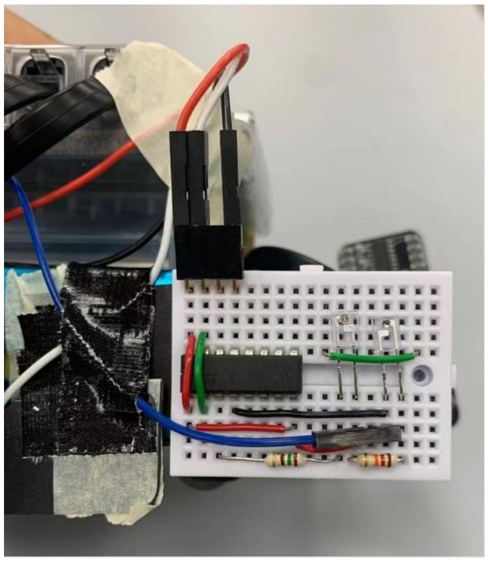
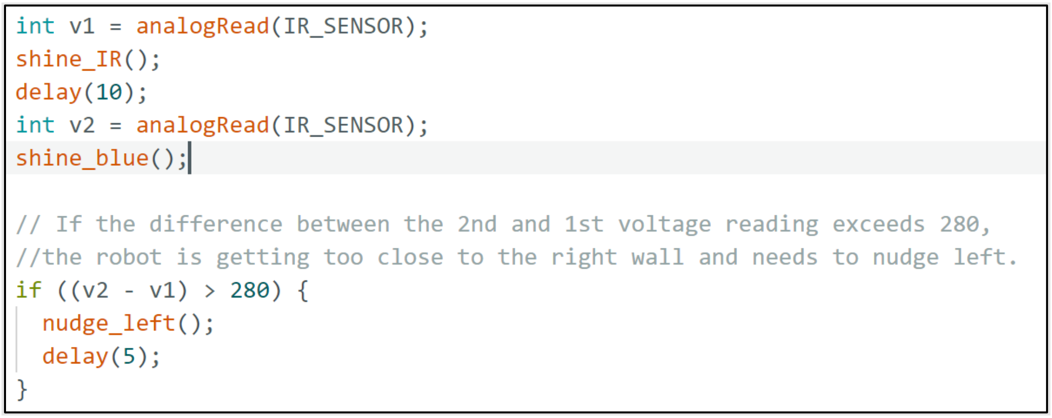


Figure 6: IR sensor

To improve the robustness of IR sensor, the code is modified to reduce the effect of ambient IR on distance detection. As shown in Figure 7, instead of reading IR voltage directly, the difference between IR voltage readings is used for detecting the distance from the right wall and executing nudging command.



```
int v1 = analogRead(IR_SENSOR);
shine_IR();
delay(10);
int v2 = analogRead(IR_SENSOR);
shine_blue();

// If the difference between the 2nd and 1st voltage reading exceeds 280,
//the robot is getting too close to the right wall and needs to nudge left.
if ((v2 - v1) > 280) {
  nudge_left();
  delay(5);
}
```

Figure 7: Fast Looping to counter ambient IR

1.4 Colour Sensing Algorithm

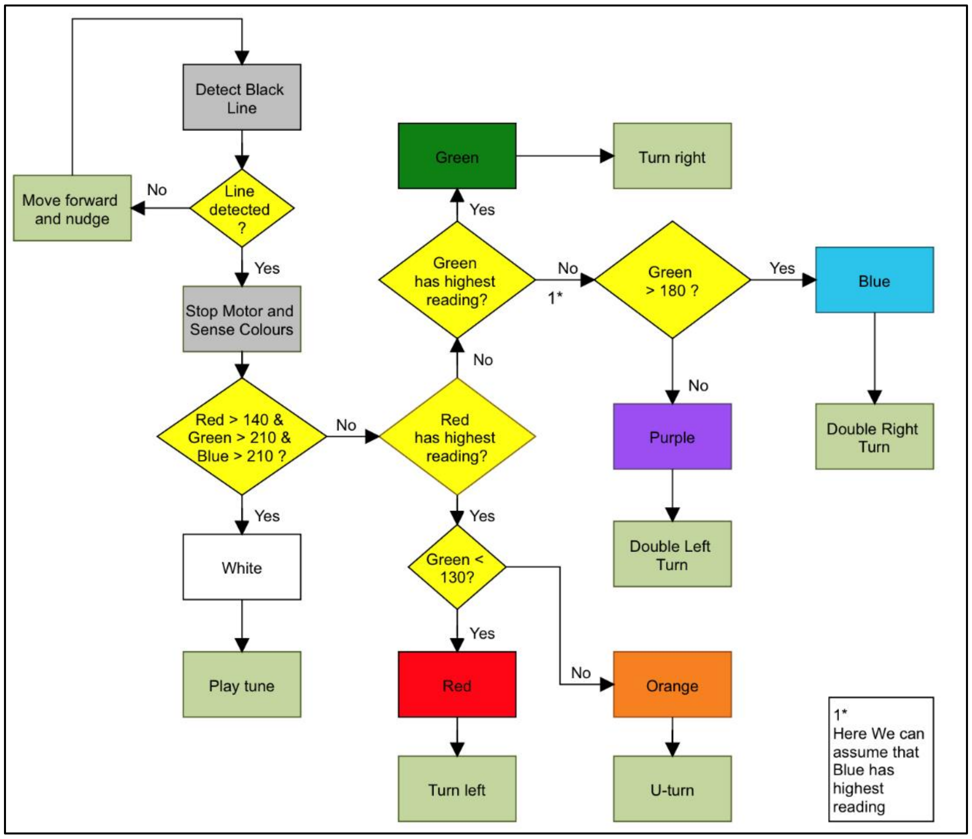


Figure 8: Colour sensing algorithm

As shown in the flowchart (Figure 8), upon detecting a black line, the robot will come to an immediate stop and activate the colour sensor. The colour sensor will systematically switch on the Red, Green, and Blue LEDs one by one for a total of 10 readings each, measuring the voltage across the Light Dependent Resistor (LDR). The resistance, and hence voltage, of the LDR varies according to the amount of light received. Therefore, the different amounts of light reflected into the LDR can be translated into the characteristics of specific colours and are read as 10-bit values. Then, the average of the last 5 readings is stored in the colourArray[]. Since taking readings is faster and more efficient than using delay(), we chose to take 10 readings for each colour. However, the first 5 readings tend to be inaccurate, so only the last 5 readings out of the total 10 are used to compute the average reading for each colour.

The values stored in the colourArray[] are then compared to the pre-set values, as depicted in the flowchart. This comparison aids the mBot in identifying the colour of the paper underneath, enabling it to execute appropriate turns and successfully complete the colour challenge.

1.5 End-of-Maze Detection

Throughout the maze, the mBot will keep checking to see if there is a black strip and the white-coloured paper underneath to indicate the end of the maze. When it does detect this point, it will play the victory tune and stop the movement of the mBot.

To keep the theme in line with the futuristic nature of this project, we have decided on the Star Wars theme to signal the end of the maze. In this part, we searched for the frequency of each note and arranged them to form the well-known melody.

2. Steps taken for calibration

2.1 Distance Sensing

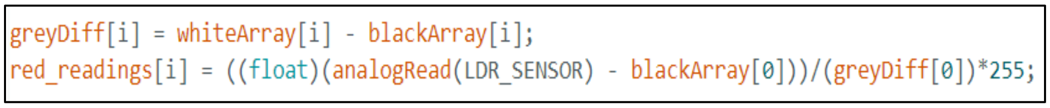
To improve the sensitivity of IR sensor in detecting the distance between the right side of mBot body and the wall, we underwent a vigorous trial and error process to determine the optimal resistance value to connect to the IR detector. The table below (table 1) shows a brief comparison between several resistor values we have chosen. The degree of sensitivity is determined by the effectiveness to adjust distance from the right wall and the degree of consistency is determined by the smoothness of the movement of mBot.

Table 1: Comparison of resistor values

Resistance	Performance
8.2 kΩ	Low sensitivity and high consistency
12 kΩ	Optimal balance of sensitivity and consistency
15 kΩ	High sensitivity and low consistency

From the results shown above, we have chosen 12kΩ as the resistor value connected to the IR detector although it did not give the most sensitive outcome. This is because, after carrying out the testing, we have found that this resistance was a perfect balance between outputting smoothly changing values and navigating the mBot to move in a straight line.

2.2 Colour Sensing



```
greyDiff[i] = whiteArray[i] - blackArray[i];
red_readings[i] = ((float)(analogRead(LDR_SENSOR) - blackArray[0]))/(greyDiff[0])*255;
```

Figure 9: GreyDiff & Readings Conversion Calculation

Our colour sensor circuit consists of 3 LEDs (Red, Green and Blue) and a Light Dependent Resistor (LDR). The shining of each LED is controlled via an HD74LS139P 2-to-4 Decoder IC Chip, where LOW at both 1A and 1B shines the red LED, a LOW at 1A and a HIGH at 1B shines the blue LED and a HIGH at 1A and a LOW at 1B shines the green LED.

With the 2-to-4 decoder, we can now use just two signal pins from the mCore to control the turning on/off of the three colour LEDs (i.e., red, green, blue) as well as one IR emitter. This leaves our mCore with two remaining signal pins for reading the LDR circuit's voltage and the IR detector's voltage.

The LEDs are shine in the order of red, green then blue. The light reflected off the surface beneath the robot is then detected by the LDR and an analog value is sent to the Arduino based robot. For calibration of the colour sensor, we conducted our base measurements of the whiteArray[] and blackArray[] using the white and black coloured papers respectively in our environment. The white colour readings stored in the whiteArray[] represent scenarios where all light is reflected, while the black colour readings stored in the blackArray[] indicate situations where most light is absorbed.

By subtracting the black values from the white values, the greyscale can be determined (Figure 9). Since the analog pin values of the Arduino span from 0 to 1023 (representing 0V to 5V), a conversion is necessary to align these values with the RGB colour code (e.g., 255, 0, 0 for red). This conversion is achieved by employing the ratio of the difference between the measured value of the LDR and the black value to the greyscale (Figure 9).

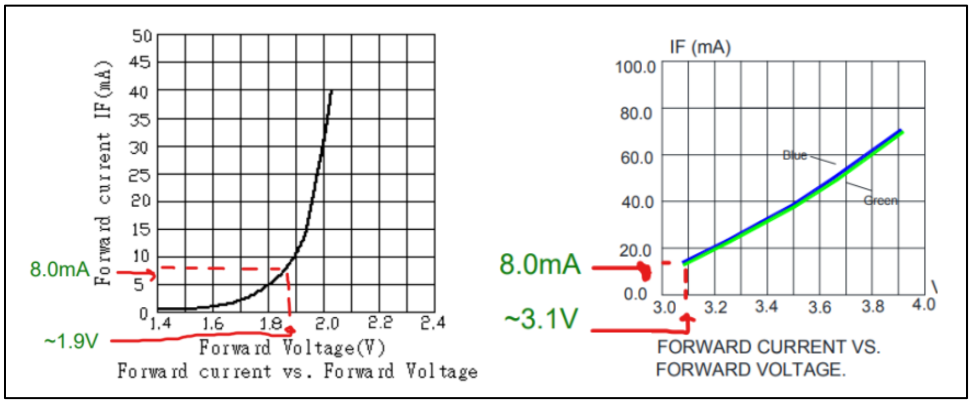


Figure 10: Current vs Voltage graph for Red (Left), Blue and Green (Right) LEDs

As for the resistance values for the red, green and blue LEDs, based on Figure 10, we decided to use a resistor value that can achieve the maximum voltage of ~1.9V for the red LED at 8.0mA which is (5.0 - 1.9)V / (8mA) = 387Ω. As for the blue and green LEDs the ideal resistor value to use would be (5.0 - 3.1)V / (8mA) = 238Ω. On the other hand, the LDR uses a 100kΩ resistor.

However, upon multiple trials and error, we concluded that the readings of each individual LEDs are too high as most of the light is reflected off the surface due to the light being too bright. Thus, we decided to increase the resistance values of all 3 LEDs and after many repeated experimentations, we arrived at our final resistance values of 1.2kΩ for the red LED and 560Ω for both green and blue LEDs.

To prevent the LDR from directly receiving the light shone by the LEDs, we went ahead and shorten the legs of the LEDs such that they are shorter than the LDR so that the light received by the LDR would solely be from above it. To further ensure this, we made a cylinder out of black paper and wrapped it around the LDR (Figure 11). (More on skirting in Section 4.1.)

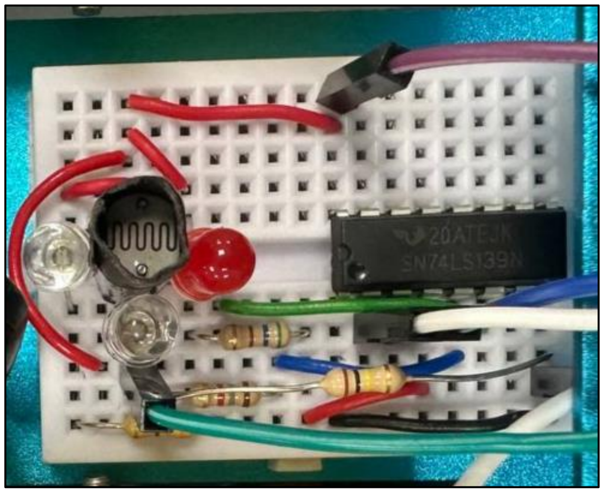
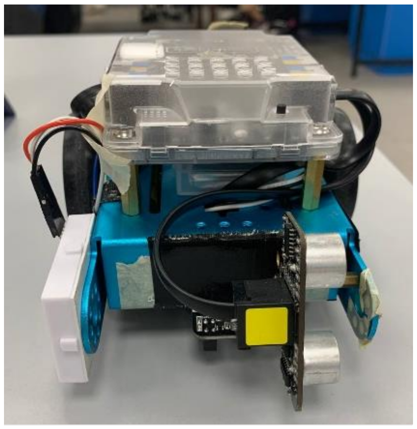
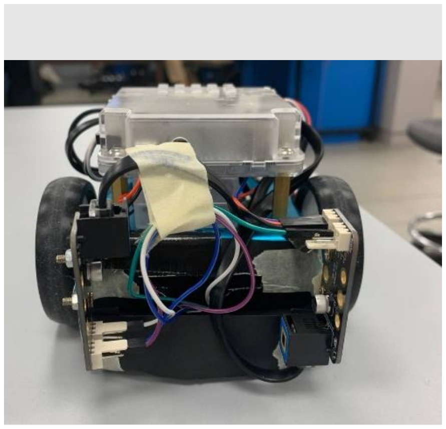


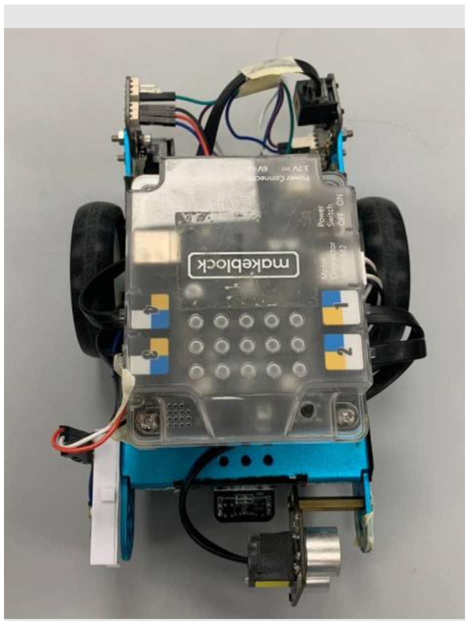
Figure 11: Colour sensor circuitry

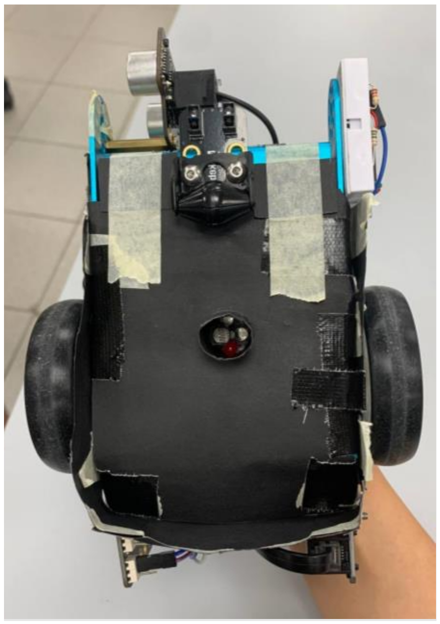
The other end of the purple wire (Figure 11) is connected to the VCC pin of a RJ25 adaptor, and it provides the 5V supply to the circuit. The green wire is connected to Pin S2 of port 3 and it is used to read the voltage readings of our LDR sensor. The white and blue wire that are adjacent to each other are connected to Pin S1 and Pin S2 of port 4 respectively and it is used to control the turning on of the red, green and blue LEDs. The 1G Enable pin is connected to ground to ensure its active LOW and not “floating” around. Y3 output of the chip is reserved for the IR sensor where an output of LOW will result in the input of the L293 motor driven chip in the IR circuit to be LOW and hence its output thus turning on the IR emitter.

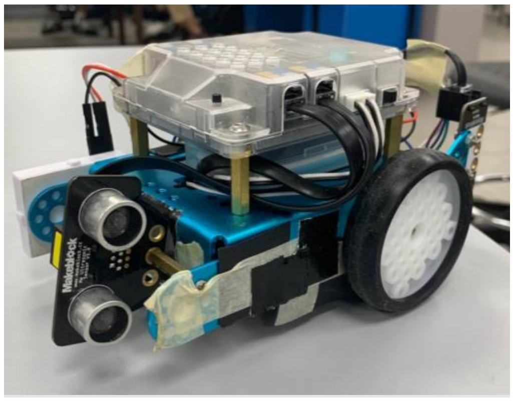
3. Final Design of mBot

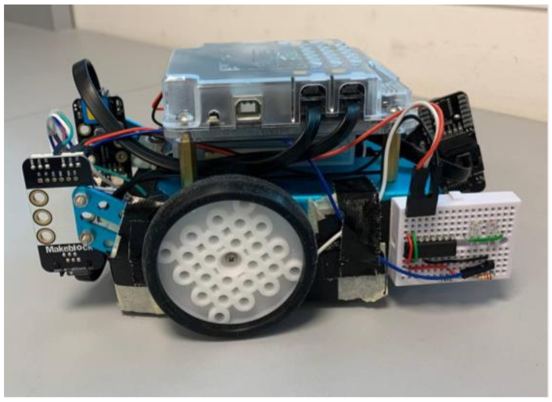

Front View

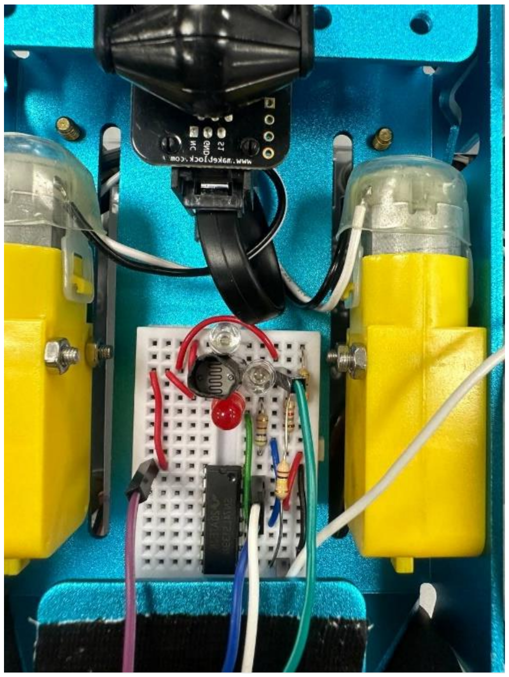

Back View

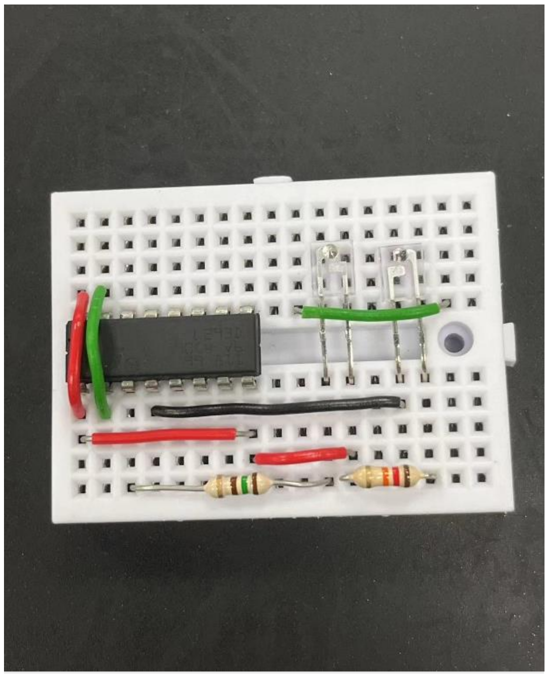

Top View


Bottom View


Left View


Right View


Inner connection


IR sensor connection

4. Significant Difficulties and Steps to Overcome

4.1 Lighting Changes

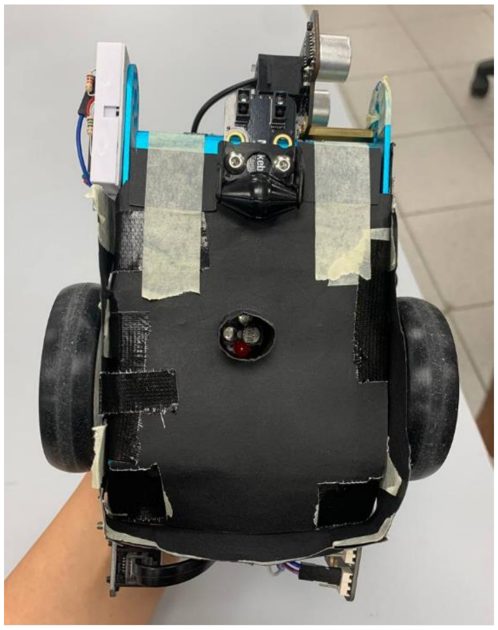


Figure 12: Skirting of mBot

Due to the ever-changing lighting conditions faced by the mBot, the colour sensor often detected the wrong colour when it stopped on top of a piece of coloured paper. This is because the mBot's colour sensor is calibrated in a specific lighting condition at a specific period which may not be replicable after that period. Therefore, even though the mBot may work correctly right after the colour sensor is calibrated, it may start malfunctioning after some time due to fluctuating lighting conditions. To overcome this issue, we decided to use black paper (Figure 12) to block off ambient light to the LEDs and the LDR.

Furthermore, in the algorithm section, rather than comparing the values in the colourArray[] to specific numbers, we opted to compare the RGB values to each other. To differentiate colours with similar characteristics, we identify them based on specific readings. Take red and orange, for instance. Since both exhibit the highest reading in red among the three colours, we use the green reading to distinguish between them. Specifically, since the green reading for red is less than 130 and for orange exceeds 130, we can effectively discern between the two. This approach enhances colour distinction and contributes to improved accuracy.

4.2 Corner-Turning of the mBot

As the mBot navigated through the maze, a common issue we faced was the mBot overturning and consequently, crashing into the side walls. This issue surfaced more when the mBot sensed the colours purple or light blue and had to execute two consecutive left-turns or right-turns respectively. We also noticed that the mBot had the tendency to crash into the wall usually only after the first turn was executed successfully. Therefore, this required us to undergo rigorous trial and error testing so that we could figure out the optimal amount of time for the mBot to initiate its turns and the distance it should travel forward after initiating the first turn. After these two values were calibrated correctly, the mBot was able to pass all the colour challenges without crashing into any of the obstacles side walls.

4.3 IR Sensor

The IR sensor connected to the mBot generated significant difficulties for our group. The IR sensor was installed on the right side of the mBot, and it was supposed to work in harmony with the ultrasonic sensor installed on the left side of the vehicle to keep the mBot moving in a straight-line trajectory until it reached a piece of colour paper. During our first attempt with a fully constructed mBot, we deliberately tilted the mBot towards the walls on the vehicle's right side to test the functionality of the IR sensor and it was able to reorientate the mBot to its optimal straight path. However, after a few more attempts in the obstacle course, the IR sensor began to fail. The vehicle would travel in a curved path towards the right and crash into the walls on the right side of the mBot. Initially, we believed that this behaviour was due to the mBot having low battery, and it possibly affecting the power output level to the IR sensor circuit of the mBot. However, even after fully charging the mBot, the problem with the IR sensor persisted.

We decided to take a closer look at the individual electrical components within the IR sensor circuit since the mBot itself was not identified as the issue. We proceeded to replace the IR emitter and detectors within the circuit as we hypothesised that it may have stopped working after incurring external damages from hitting the walls of the obstacle course. However, the IR sensor continued failing to work. Then, after some careful testing, we determined that the defective L293 motor-driven chip was the issue in the IR sensor circuit.

This was probably due to the multiple collisions that our mBot had with the walls during trials. These collisions might have led to some form of contact between protruding metallic components and the L293 chip, resulting in a short circuit. To resolve this issue, we decided to replace the L293 chip, tidy up the IR circuit by shortening the legs of resistors and hold wires in place using tape. We then conducted a trial run, and the IR sensor was able to detect the spike in IR emitted by the IR emitter thus allowing the robot to nudge correctly.

5. Work Division

This project's success was only possible with everyone's hard work and dedication to making our mBot traverse past all the obstacles. The table below documents the work distribution among the members in the group.

Table 2: Work Division

Task completed	Team Members in-charge
Colour sensor	Hao Zhe
Infrared sensor	Mengyan and XiaoNa
Ultrasonic sensor	Hao Zhe and Zekuan
Wiring of Components	Zekuan and Hao Zhe
Arduino Code	Hao Zhe
Skirting for mBot	XiaoNa and Mengyan
Report	Mengyan, XiaoNa and Zekuan

6. Conclusion

This project was a fun conclusion to our CG1111A module, and it required the application of many of the concepts we have learned throughout this semester. After weeks of working together, we are proud to have been able to present a robot that could complete the maze in 30 seconds without incurring any penalty deductions. Although our group may have faced a few difficulties in ensuring the mBot would function as we intended it to, we are thankful for the valuable lessons we have learned as we corrected the errors we faced. Finally, we would like to thank Professor Sangit Sasidhar for his guidance and patience in teaching us the fundamental principles of engineering and we would also like to thank the teaching assistants who have aided us greatly in refining our mBot.