CG1111A: ENGINEERING PRINCIPLES & PRACTICES I

B02-S4-T1 PROJECT REPORT

THE A-MAZE-ING RACE 2025

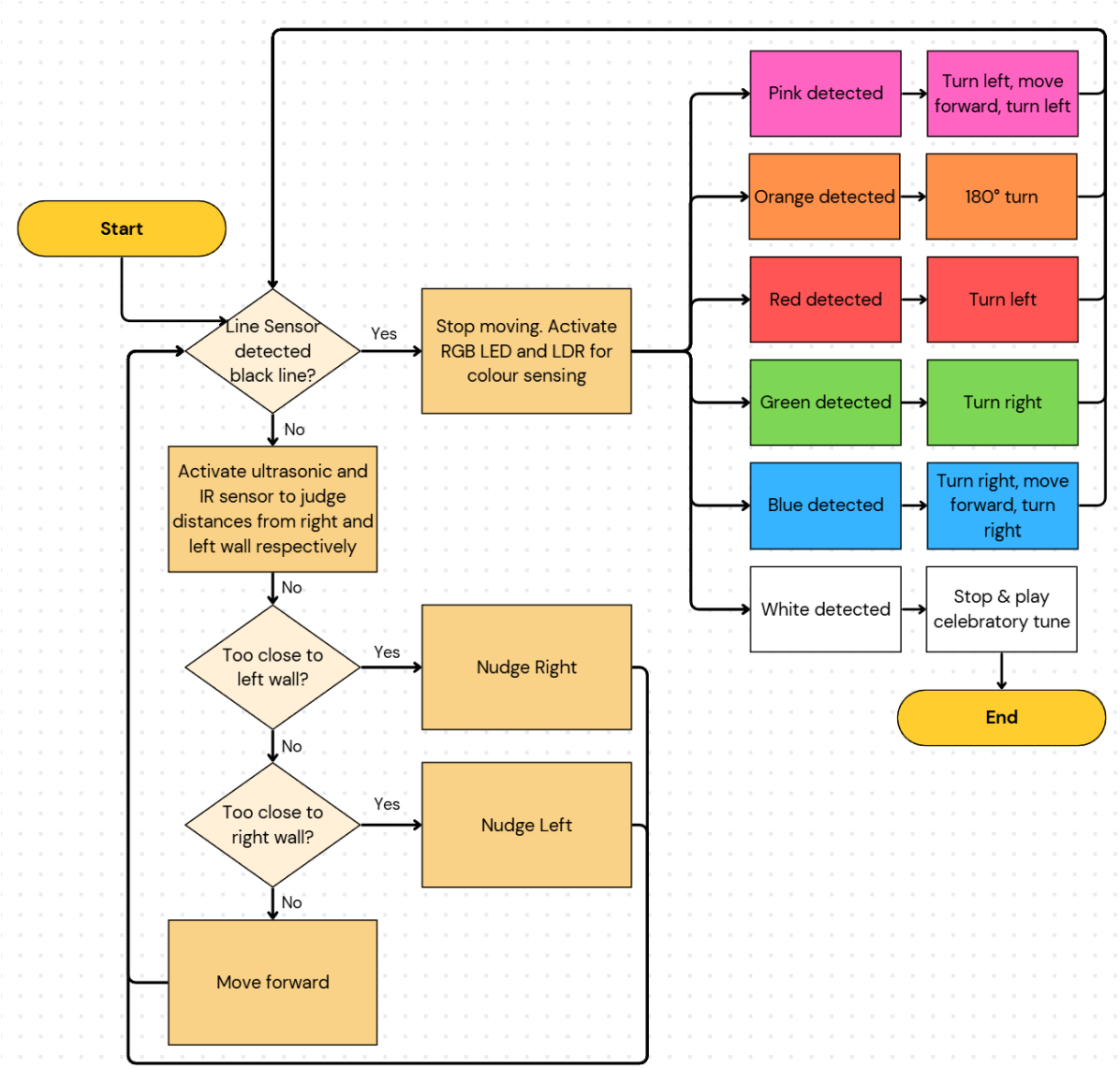
Team Members:	[redacted]
LIM ZHE MING	A0322364L
LOO ZHONG EN SMAUEL	A0322527J
LOW YI JIE SHERYL	A0321940M
MO JIN EN	A0320200L

TABLE OF CONTENTS

SECTION 1:

CODE ALGORITHM

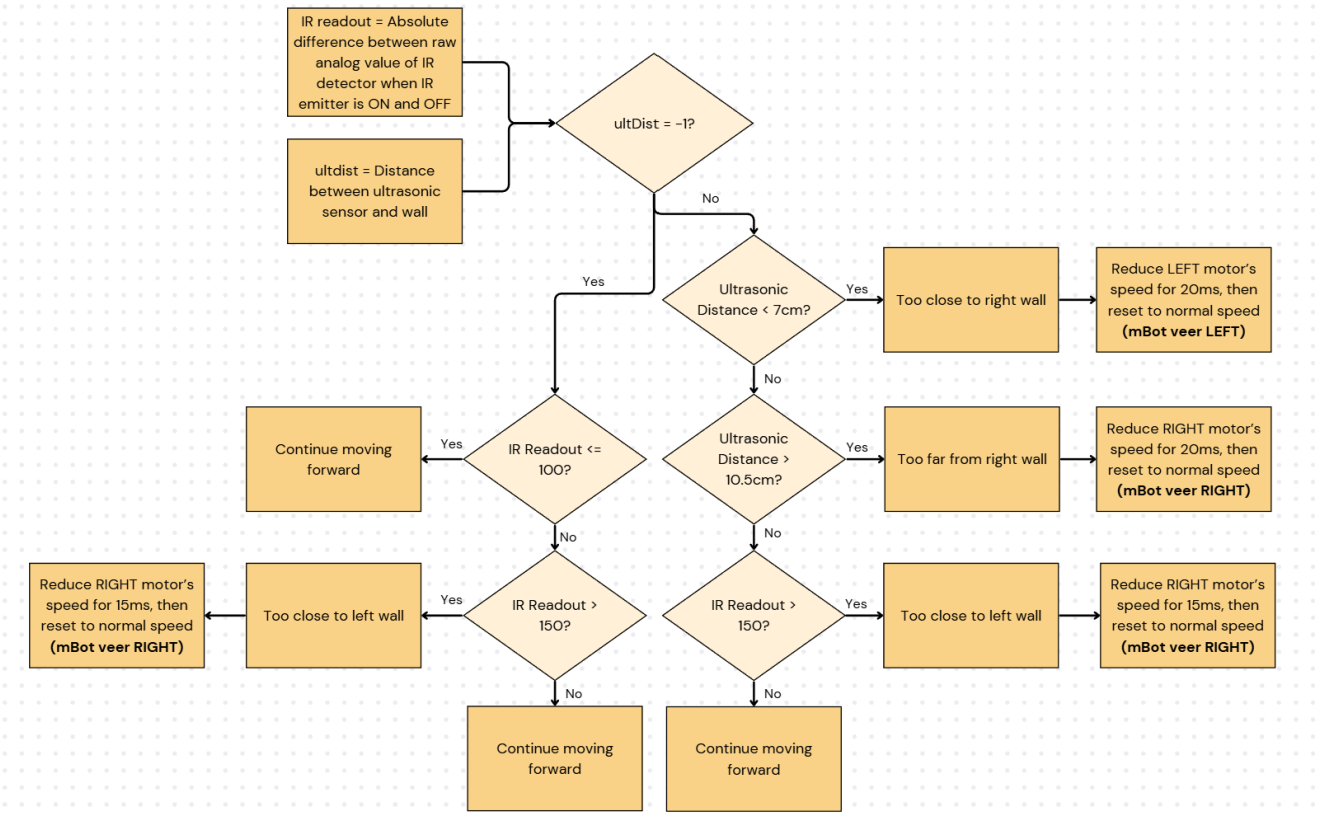
PART 1: OVERALL ALGORITHM



Flowchart 1.1: Simplified Algorithm used to Navigate the Mbot through the Maze

PART 2: CENTERING ALGORITHM

To keep our robot centered and moving straight, our team has used the following algorithm:



Flowchart 1.2: Centering Algorithm

Based on our team's testing, we have determined that a target range of 7-10.5 centimeters is ideal in ensuring that the robot stays centered. To aid in this, we have employed the use of the Ultrasonic Sensor and the Infra-Red (IR) Sensor.

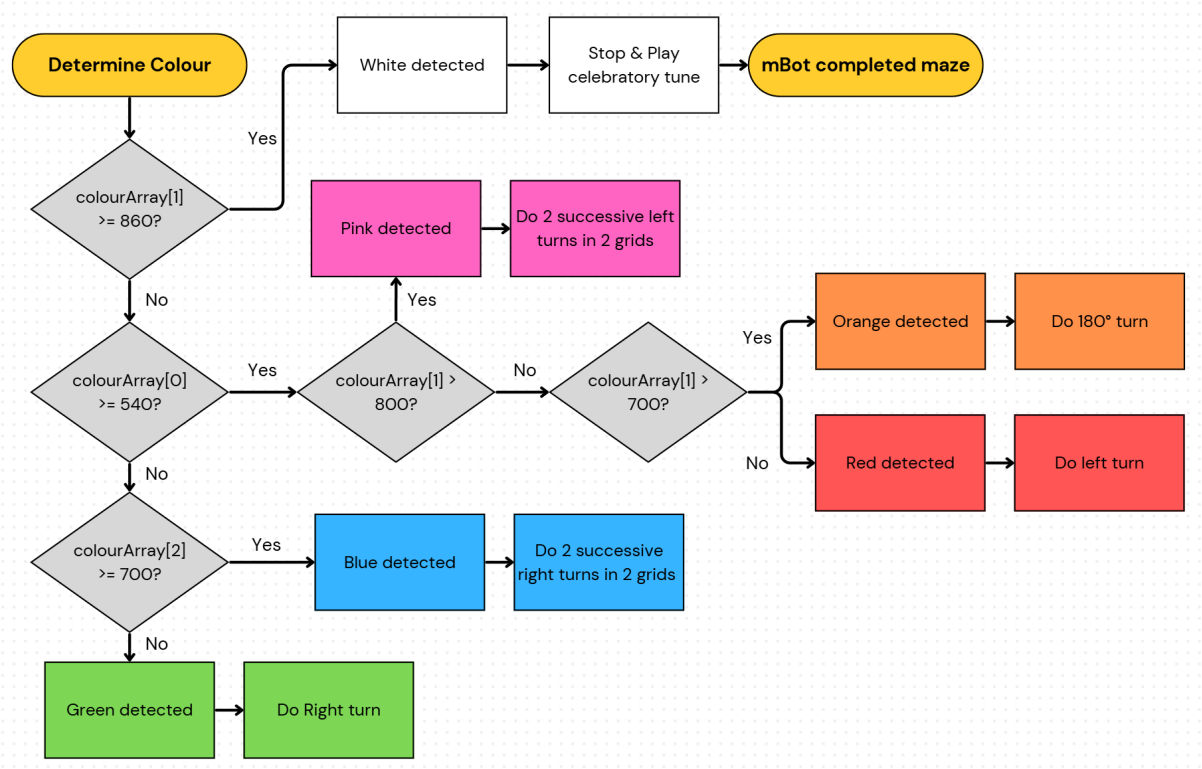
However, due to the limited range of the IR Sensor, our team relies on the Ultrasonic Sensor as the primary sensor for course correction. The IR Sensor is used only in two scenarios:

1. Invalid distance reading of -1 is returned by the Ultrasonic sensor, indicating sensor timeout
2. In the event the robot over-corrects itself based on the data from the Ultrasonic sensor and veers too left.

In both scenarios, should the IR Sensor return a value that is > 150, corresponding to a distance reading of less than 5cm, the mBot will nudge right to avoid hitting the left wall.
The specific values used to determine whether the robot is too close or far from the wall and whether a wall is missing is explained in subsequent sections.

To steer the mBot back, we employ a method of nudging by dropping the speed of the left or right motor from 255 to 50, depending on the given condition. By having a speed difference between the wheels, this causes the mBot to pivot, allowing it to correct its heading. The duration in which the nudging occurs is either 20ms or 15ms depending on the situation. This gives the mBot ample time to reangle itself but not too long such that it overcorrects.

PART 3: COLOUR DETECTION ALGORITHM



Flowchart 1.3: Flowchart of the Color Detection Algorithm

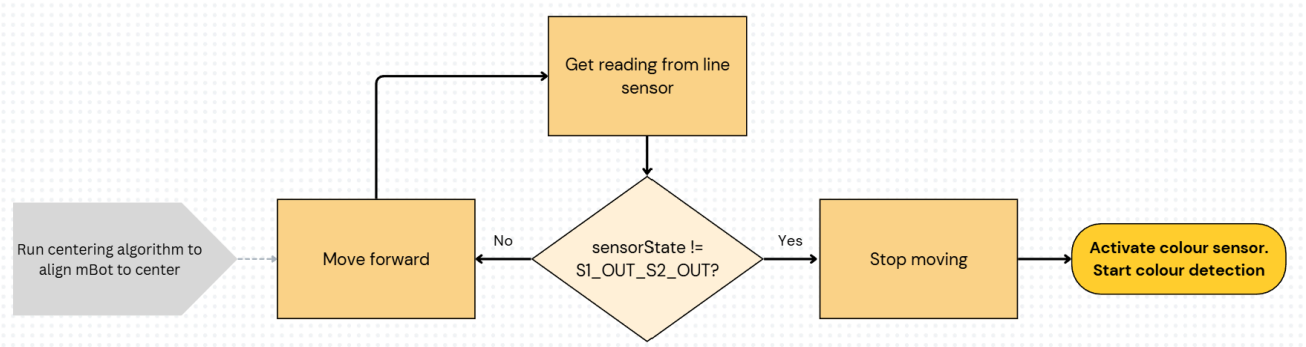
During the colour detection process, Red, Green and Blue LEDs are lit individually and a Light Dependent Resistor (LDR) is used to determine how much light from the LED is reflected by the coloured paper. To store these readings, our code uses an integer array, named colourArray, of 3 elements corresponding to the different LEDs as explained below:

- colourArray[0]: Red LED Reading
- colourArray[1]: Green LED Reading
- colourArray[2]: Blue LED Reading

For colour detection, we implemented a threshold-based approach by comparing RGB sensor readings against calibrated values. Using a systematic elimination process, the program first checks for white, then differentiates red-spectrum colours based on the value stored in colourArray[1], and finally distinguishes blue from green based on the value stored in colourArray[2]. The process of collecting the necessary data is explained further in the next section.

PART 4: LINE DETECTION ALGORITHM

To ensure that the robot is able to stop to decode the different waypoint challenges, we employed the use of the Line Detector of the Mbot to detect the black paper strip at the top of the coloured paper. To ensure that the robot does not over-run the black strip, the conditional used triggers when either one or both of the line sensors detect no reflected light. This is achieved by checking for whether the reading is not equal (!=) to the state S1_OUT_S2_OUT, which indicates that no black strip is detected. The flowchart below shows the control flow of the line detection:



Flowchart 1.4: Line Detection Algorithm

SECTION 2:

IMPLEMENTATION OF SUB-SYSTEMS & CALIBRATION

PART 1: COLOUR SENSOR & TURNING

IMPLEMENTATION

Each LED is activated individually. To turn on the 3 different LED lights, the 2-to-4 Decoder IC Chip was used. Only the left half of the decoder IC (the side labelled with "1") was utilised. The Enable pin 1G was connected to GND, allowing us to control the Select Input pins to switch on the desired LED.

LED Colour	2-4 Decoder Output Pin
Red	Y1
Green	Y2
Blue	Y3

Table 2.1.1: Connections of RGB LED to 2-4 Decoder IC Chip

Taking reference from the Table 2.1.2 below, the decoder operates with active-low outputs, meaning the LED turns on when its corresponding output is set to LOW. For example, as shown in Table 2.1.1, the red LED is connected to Output 1Y1 of the decoder chip. To turn it on, both Select Input pins 1A and 1B must be set to LOW. This causes Output 1Y1 to be set to LOW, illuminating the red LED.

Inputs			Outputs			
Enable	Select					
G	B	A	Y_0	Y_1	Y_2	Y_3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H ; high level, L ; low level, X ; irrelevant

Table 2.1.2: Function Table Summarizing the Behavior of 2-to-4 decoder

To prevent the current through the LEDs from exceeding 8mA, the following resistor values were used:

LED Colour	Resistance of resistors (Ω)
Red	5.5k
Green	8.1k
Blue	8.1k

Table 2.1.3: Final Resistance of Resistor Connected in Series to Each LED

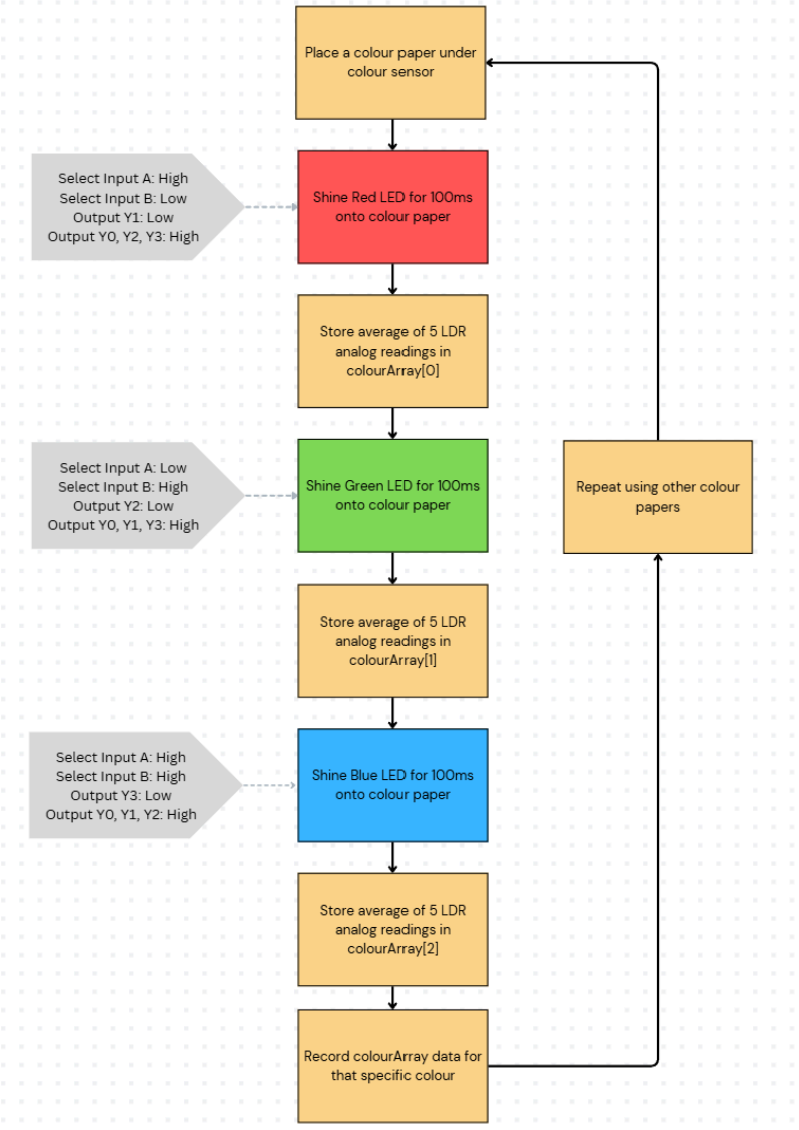
In addition, a 46kΩ resistor is used for the LDR so as to make it sensitive to the differing light output. These resistor values also gave us the most consistent results.

CALIBRATION

To gather the necessary data required to calibrate our colour sensor, we used switch statements to turn on the Red, Green and Blue LED lights separately. The LDR is connected to pin A0 of the mBot as analog INPUT, allowing us to obtain its readings. To store these readings, our code uses an array, named colourArray, of 3 elements corresponding to the different LEDs as explained below:

- colourArray[0]: Red LED Reading
- colourArray[1]: Green LED Reading
- colourArray[2]: Blue LED Reading

To ensure accuracy, 5 analog readings were taken from the LDR per LED. A delay of 100ms is imposed before the we obtain the first reading so as to allow the LDR to adjust to the sudden change. Thereafter, a delay of 10ms is imposed in-between each reading to prevent mis-reading. The average of these readings is then stored into the corresponding array element. We also obtained readings from various maze tables in the lab, and varied the position of the mBot on the colour paper. We did this for all 6 coloured papers. The following flowchart elaborates this process further:



Flowchart 2.1.4: Colour Sensor Data Collection Method

The following table shows a sample of the data collected in our calibration. It is important to note that depending on the lighting conditions in the lab on different days of testing, a degree of minor fine tuning is required to ensure that the colour sensor functions as intended, even with proper shielding. Hence, the exact values used in our final code differs from the values obtained in the table.

PAPER COLOUR																		
	WHITE			RED			PINK			ORANGE			GREEN			BLUE		
	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
TABLE 1	483	833	766	493	566	515	528	793	715	487	681	546	312	742	615	215	719	712
TABLE 2	501	826	759	502	573	521	564	795	715	527	686	556	355	748	626	206	703	697
TABLE 3	514	828	761	515	580	528	545	791	711	532	686	559	322	739	615	190	718	712
AVERAGE	499	829	762	503	573	521	546	793	714	515	684	554	330	743	619	204	713	707

Table 2.1.5: Sample data collected from Colour Sensor Calibration

In our colour detection algorithm, based on the average readings from the different tables, we found which component of the colourArray is the most prominent to determine the paper's colour.

For example, for the case of white, based on the sample data above, we note that it is the only colour for which the reading for the Green LED is above 820. As such, in our conditional check, we implemented a condition to check for this characteristic. If this passes, the colour is determined to be white.

To distinguish Red, Pink and Orange from Green and Blue, we noticed that the RED LED values for Green and Blue were lower than the others. As such, this became our distinguishing factor.

This threshold based system of elimination and the final values used to distinguish the individual colours have been explained in the earlier section on colour detection.

After ensuring that our colour sensor is able to read the paper's colour reliably and accurately, we moved on to calibrating the execution of the various turns required as shown in the table below:

Colour	Interpretation
Red	Left-turn
Green	Right turn
Orange	180° turn within the same grid
Pink	Two successive left-turns in two grids
Light Blue	Two successive right-turns in two grids

Table 2.1.6: Colour Challenge Turns

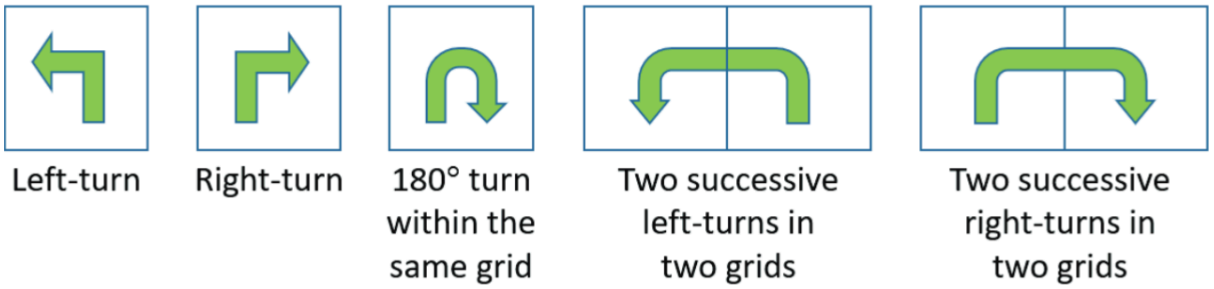


Diagram 2.1.7: Diagram of Turn Execution

We broke the problem into two parts:

1. Single Turns
2. Double Successive Turns

For single turns in the same grid, a process of trial and error was used to determine the optimal delay time imposed for the turning. Based on our testing, a turning time of 340ms was found to be the optimal delay required in ensuring the robot turns approximately 90°. For a 180° turn, we simply multiplied this turning time by 2. As the Ultrasonic sensor is placed on the right of the robot and based on our centering algorithm, the robot is relatively closer to the right wall than the left. As such, we decided that the robot should turn in the anti-clockwise direction for the 180° turn.

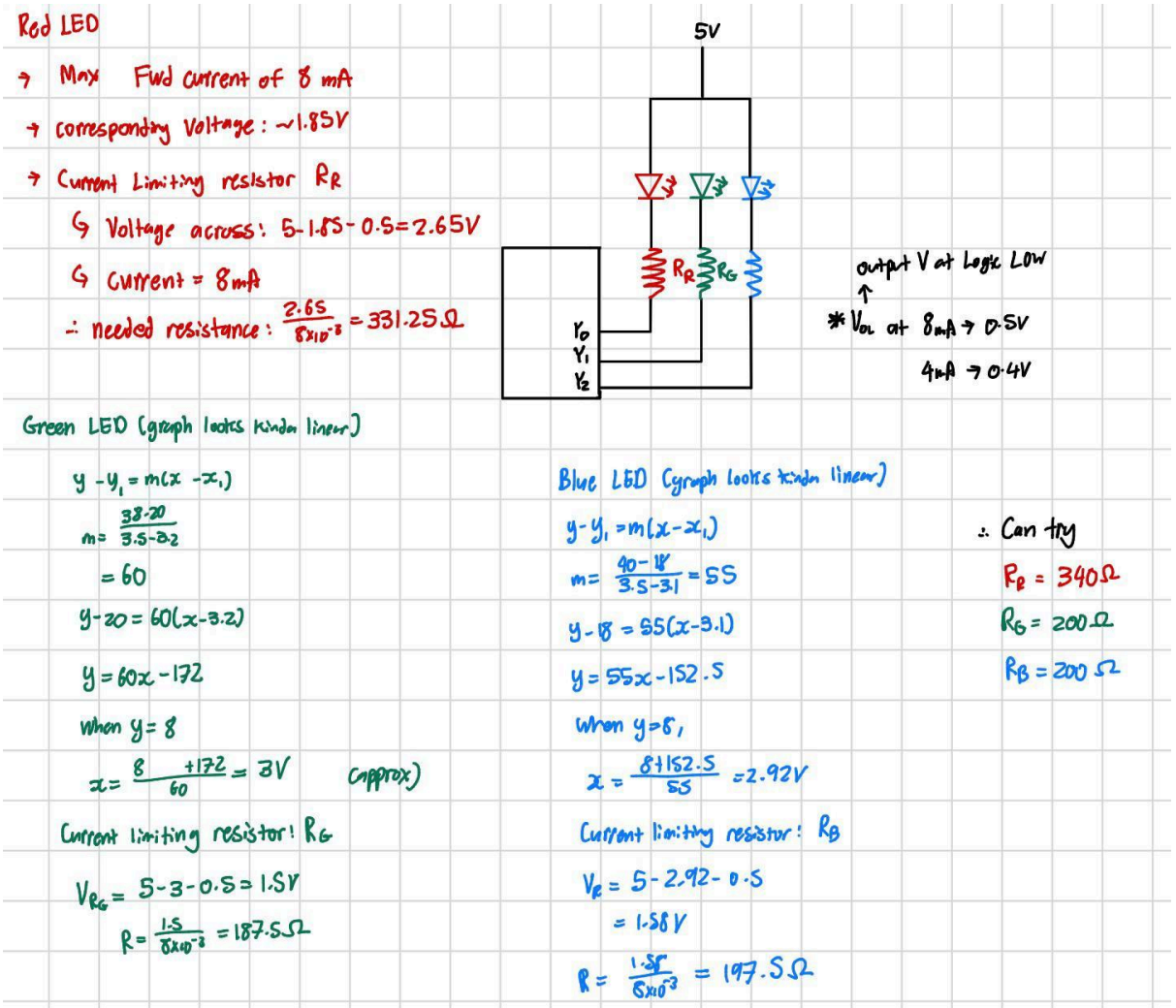
For Double Successive Turns, the main challenge is in determining the delay time required for the robot to move up to the next grid. We also did this through a process of trial and error. A delay time of 800ms was found to be optimal in ensuring sufficient time for the robot to move forward. The same 340ms delay was used in ensuring the robot makes an approximate 90° turn for both turns.

CHALLENGES

Colour Sensor:

After we first built the colour sensor and tested it out for calibration, the colour sensor gave us inconsistent readings for the different coloured papers. We realised that this may be due to the ambient light from the surroundings which was causing this issue as the ambient light will disrupt the reflected light detected by the LDR for each LED. To tackle this issue, we added proper shielding to our robot, adding a chimney around the LDR, Red, Green and Blue LED lights, as well as making sure that the base of the robot is completely covered by black coloured paper so that no ambient light is detected by the LDR (see Annex Photo 5). With that, we were able to get more consistent readings for our colour sensor. In addition, before each lab session, we re-calibrated the colour sensor so as to ensure that the robot is able to adapt to the changing lab environment.

In the early stages of designing the colour-sensor circuit, we initially selected the following resistor values through the following calculations:



Calculation 2.1.8: Calculations done to obtain initial resistance values for the LEDs

LED Colour	Red	Green	Blue
Calculated resistance of resistors in series with LED (Ω)	340	200	200

Table 2.1.9: Table summarising the calculated resistance values for the LEDs

These resistor values were chosen based on the forward current–voltage characteristics of the red, green, and blue LEDs as specified in their datasheets. However, during calibration, we found that these values produced very small differences in the LDR's analog readings—typically less than 100 across the red, green, and blue measurements. This limited separation made it difficult to determine clear threshold values for distinguishing one colour from another, indicating that the sensor was not sufficiently sensitive. To address this, we experimented with various combinations of resistor values to increase the contrast between readings. We also chose not to scale the readings down to 255 as this conversion amplified inconsistencies observed when calibrating with different colour samples.

During experimentation, we also encountered situations where all the LEDs were "too bright". In these cases, the LDR consistently detected a high amount of reflected light for the colour being tested. This issue became especially apparent when we tested the colour sensor using black paper, where the red, green, and blue analog readings were unexpectedly high. Though black should ideally produce very low values for all three since it absorbs the most light. When this happened, the colour sensor became less sensitive because the reflected light readings for that LED did not vary much across different colour samples.

Through multiple trial and error attempts, we settled on the current set of resistor values stated earlier. This is because they gave us a clear separation between the analog readings for the different paper colours, making it easier for us to implement conditional checks for our colour detecting algorithm.

Motor Turning:

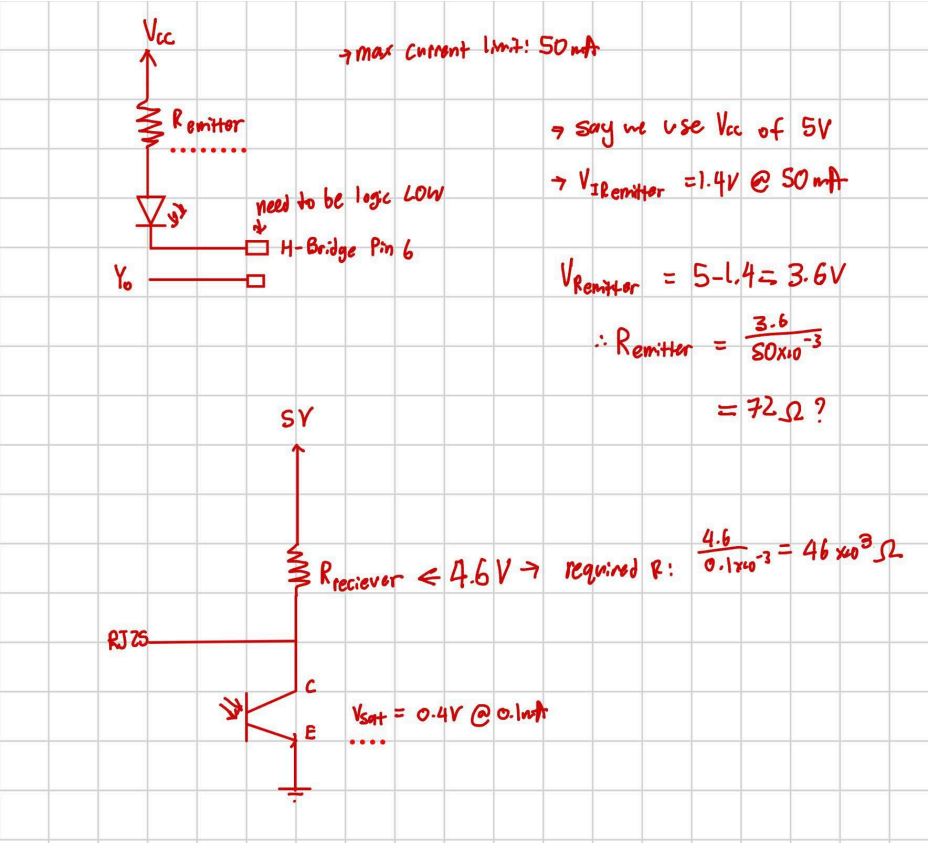
As the battery of the mBot drains over time with repeated testing, it also causes the power output of the motors to decrease over time, resulting in the robot moving slower. This is a major issue as it could mean that a perfect turn executed by the robot may become an over or under turn should the battery level of the robot change. To combat this, our team fully charged the robot before each calibration, ensuring that the changes made are not affected by a lack of power output from the motors, leading to more consistent results.

PART 2: INFRA-RED (IR) SENSOR

IMPLEMENTATION

As the 2-4 Decoder can only output a maximum of 8mA, which is not enough to power our IR Emitter, we connected it to the L293D motor driver chip, which can output a much greater 50mA. To ensure that the current does not exceed 50mA, a resistor is used to limit the current. In addition, a separate resistor is used to ensure that the IR Receiver is sensitive to the incoming IR Ray from the IR Emitter to provide usable data to judge the distance of the mBot to the wall.

We used the following calculations to obtain our required resistance values:



Calculation 2.2.1: IR Emitter and Receiver Calculations

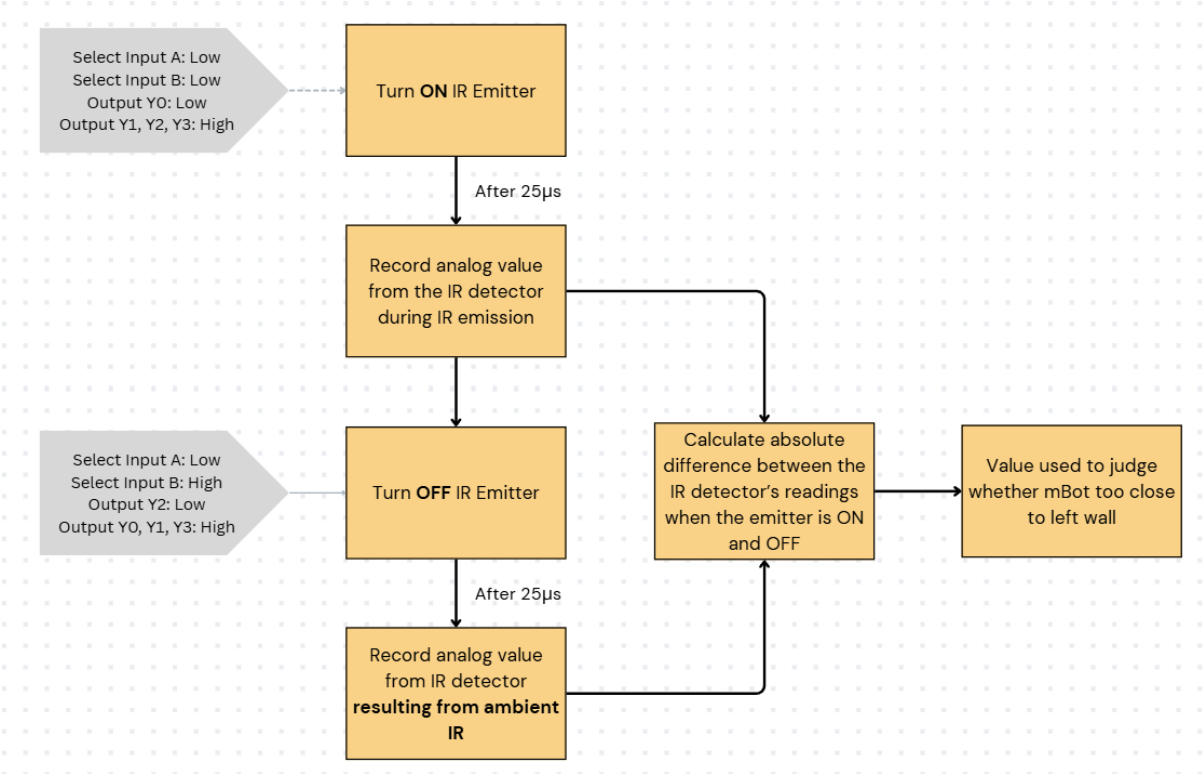
Based on these calculations and trial and error, we determined that the following resistance values gave us the most consistent results:

IR Emitter : 100Ω
IR Receiver: 15kΩ

Since the Ultrasonic Sensor is located on the front right of the mBot, the breadboard containing the IR Circuit is attached to the front left to aid in distance checking.

CALIBRATION

The IR emitter is connected to the 1Y0 pin of the 2-to-4 decoder IC chip. By controlling which Input Pins are activated at any given point in time, we can turn the IR emitter on and off, allowing us to obtain readings taking ambient light into account. A 25 microsecond delay to stabilise the readings. The following flowchart shows the sequence in which we obtain our IR Readings, as reflected in our final code:

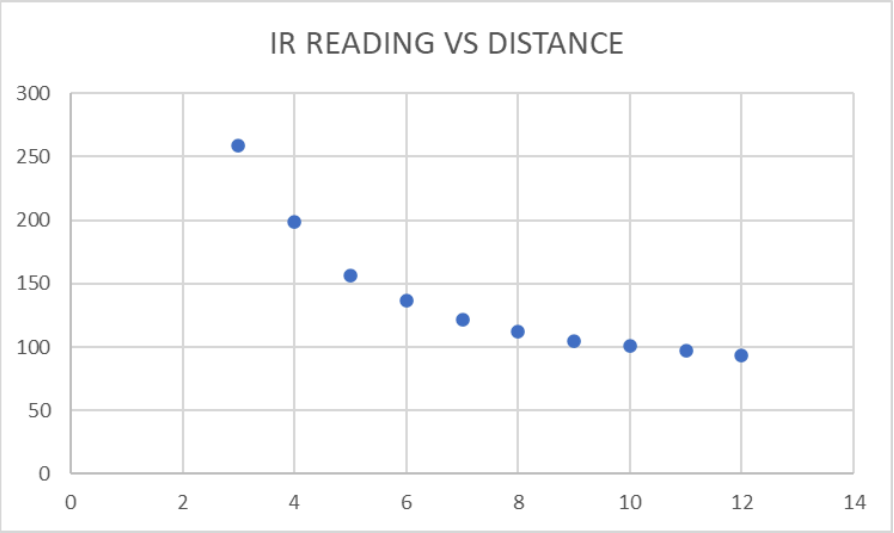


Flowchart 2.2.2: IR Sensor Calibration

To calibrate our IR Sensor, readings were taken at varying distances. The data obtained is then plotted onto a graph in Excel, allowing us to obtain a trendline which gives us the relationship of IR reading against distance. This helped us determine the range of IR values where the mBot is too close to the wall, which we used to activate the nudge right functions of the mBot. In addition, we also took a reading for 'INFINITY', indicating an out-of-range reading (which is the case for missing walls). The following table shows the data collected from our testing & its corresponding graph:

DIST	READING	DIST	READING
MIN	52	MIN	50
3	259	3	218
4	199	4	188
5	156	5	157
6	137	6	132
7	122	7	117
8	112	8	107
9	105	9	101
10	101	10	96
11	97	11	92
INFINITY	84	INFINITY	<90

TABLE 2.2.3: Raw Readings from IR Sensor at different Distances



GRAPH 2.2.4: Graph of IR Reading against Distance, taking ambient light into account

CHALLENGES

The range of the IR sensor is very limited. As shown in the table above, accurate readings can only be taken from 3-11 centimeters. Moreover, the readings from 7-11 centimeters are very close and can be mistaken for 'Infinity'. Hence, it is difficult for us to implement a conditional check in our code to determine if the mBot is drifting towards the left wall and nudge left accordingly.

Hence, in our centering algorithm, we relied heavily on our Ultrasonic Sensor to keep our mBot centered as it is more accurate.

As explained in the above section regarding the centering algorithm, there are only 2 situations whereby the IR sensor is used:

1. Invalid distance reading of -1 is returned by the Ultrasonic sensor, indicating sensor timeout
2. In the event the robot over-corrects itself based on the data from the Ultrasonic sensor

As such, the IR sensor has been relegated to a secondary source of data to gauge the distance of the mBot from the wall, only to be used in emergency situations.

PART 3: ULTRASONIC SENSOR

IMPLEMENTATION

The ultrasonic sensor is mounted on the right inner side of the mBot using bolts and nuts and connected to Port 1 of the mCore. Since the sensor cannot measure distances accurately below 3 centimeters, it is mounted inward by more than 3 centimeters.

To measure distances, the pinMode of the Ultrasonic Sensor is set to OUTPUT. A cycle of 2-10-2 microseconds is triggered, causing the Transmitter of the Ultrasonic sensor to send a corresponding LOW-HIGH-LOW pulse. The pinMode is then set to INPUT, activating the Receiver. The time taken for the pulse to reach the Receiver is obtained through the pulseIn function. To convert the time taken into a usable distance reading, we used the following formula:

$$Distance = Time\,Taken / 2.0 / 1000000 * SPEED\,OF\,SOUND * 100$$

Where SPEED OF SOUND is defined to be 340m/s. Since all our measurements are done in centimeters, we will apply a conversion formula to the value obtained in meters, as given in the formula.

The reason why the time taken is divided by 2 is due to the fact that the pulse needs to travel from the sensor to the object and reflect back to it, hence doubling the distance traveled.

Based on the calibrations below, if the distance read is less than 7.0 centimetres the mBot is too close to the wall and would nudge left, and if it is more than 10.5 centimetres, the mBot is too far from the wall and will nudge right.

In the event that the distance measured is out of the range of the sensor (occurs when there are missing walls in the maze), a timeout of 2 seconds is imposed. Should the time taken exceed the pre-set timeout, an invalid distance reading of -1 is returned.

CALIBRATION

The ultrasonic sensor was calibrated by centralising the mBot in between the wall boundaries and checking the ultrasonic sensor readings on the serial monitor using Serial.println. By doing so, we found that a distance range of 7.0-10.5 centimeters is optimal in ensuring that the robot has sufficient time to make micro-adjustments to keep it centered.

CHALLENGES

During testing, we noticed that the mBot would often veer right when it moves into a grid with missing walls, indicating that it falsely sensed that it was too far away from the 'wall' and attempted to correct itself. This is even more noticeable when the mBot is travelling along the perimeter of the maze where the edge of the table protrudes out. In addition, we also noticed that the robot kept jittering as it moved along even though it was relatively centered and within our target range.

There are two possible reasons for the above behaviour:

1. The Position of the Sensor

If the Sensor is mounted too low, it may detect the protruded edge of the table as a 'wall'. As seen from the Diagram below, the edge is located behind the wall. Thus, when the mBot moves up to a grid where there is a missing right wall, it may falsely perceive it to be 'far away' from the 'wall', which is the table edge in this case, hence veering right to correct its heading.

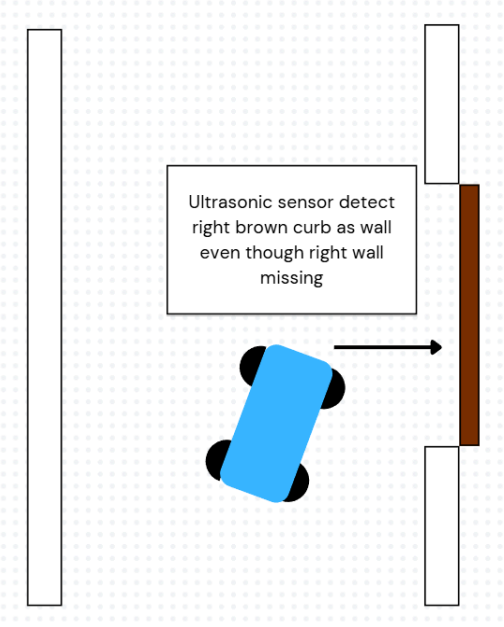


Diagram 2.3.1: Diagram to illustrate the mBot correcting itself to the table edge

2. The constant stream of data from the Ultrasonic Sensor

Our initial centering algorithm did not implement a delay causing the mBot to continually obtain data from the Sensor as it moves. As such, this may cause the mBot to:

a. Make unnecessary adjustments to its heading due to minor variations in the data obtained to keep it centered, leading to the jittering issue

b. Detect the edge of the wall and start to correct its heading towards it as it moves past the edge instead of timing out

To combat these, a series of changes were made.

First, we mounted the ultrasonic sensor to the highest hole available at the front using bolts and screws and positioned it at a slight angle (as shown in Photo 3) to prevent the bottom of the sensor barrel from detecting the right curb.

Second, depending on the scenario, we implemented a delay of either 10 or 50ms in the looped portion of the code:

1. 10ms
    - Imposed after executing a colour challenge to allow the robot to start to correcting itself should a non-ideal turn be executed
2. 50ms
    - Imposed in the Centering Algorithm as the mBot moves forward

The delay imposed allows the data collection process to be spaced out, preventing mis-readings and a constant need to make adjustments based on minor fluctuations.

These two solutions combined, helped to ensure that the mBot continues moving forward even as the wall to the right is missing, and stays relatively centred with minimal adjustments.

SECTION 3:

REFLECTIONS

The A-Maze-ing Race Project 2025 has provided us with an invaluable opportunity to apply the skills and knowledge acquired throughout the semester. Building sensor circuits from scratch and developing algorithms to gather data on the state of the mBot requires a strong foundation in both the Electrical and Programming aspects, both of which our team has had a chance to develop throughout the course and the project.

This project has also improved our problem solving skills as we diagnosed the various issues of the mBot, and taught us to be adaptable should things suddenly change. As the time frame of the project is relatively short, our team learnt how to work under pressure and also, collaboratively in order to successfully deliver the final product.

At the start of the final week leading up to the Project Evaluation, while our colour sensor was well calibrated and worked reliably, we did not have a functioning IR Sensor and the original complex algorithm employed to keep the robot centered did not work as intended. Furthermore, as we had to re-check the wiring for our IR Sensor, we had to remove the bottom shielding, shifting the position of the LEDs and LDR and hence, inadvertently rendering the colour sensor unreliable. As such, this reset the progress of previous weeks.

However, despite these setbacks, we remained resilient. We quickly diagnosed the issue with our IR circuit, and re-gathered the necessary calibration data for both the Colour and IR sensors. In addition, we rethought our entire centering algorithm and made it more simple. The final algorithm used, has been explained in the report. In the end, our team has managed to deliver an mBot that has managed to successfully complete the maze with minimal faults.

Overall, this project has taught us that successful engineering requires not only technical knowledge of circuit design and algorithm development, but also careful planning, systematic problem-solving, data collection, resilience and teamwork. The skills, values and insights gained from this project, from troubleshooting circuits to refining software code, will be extremely valuable in our future engineering endeavours.

ANNEX

SECTION 1: PHOTOGRAPHS OF ROBOT

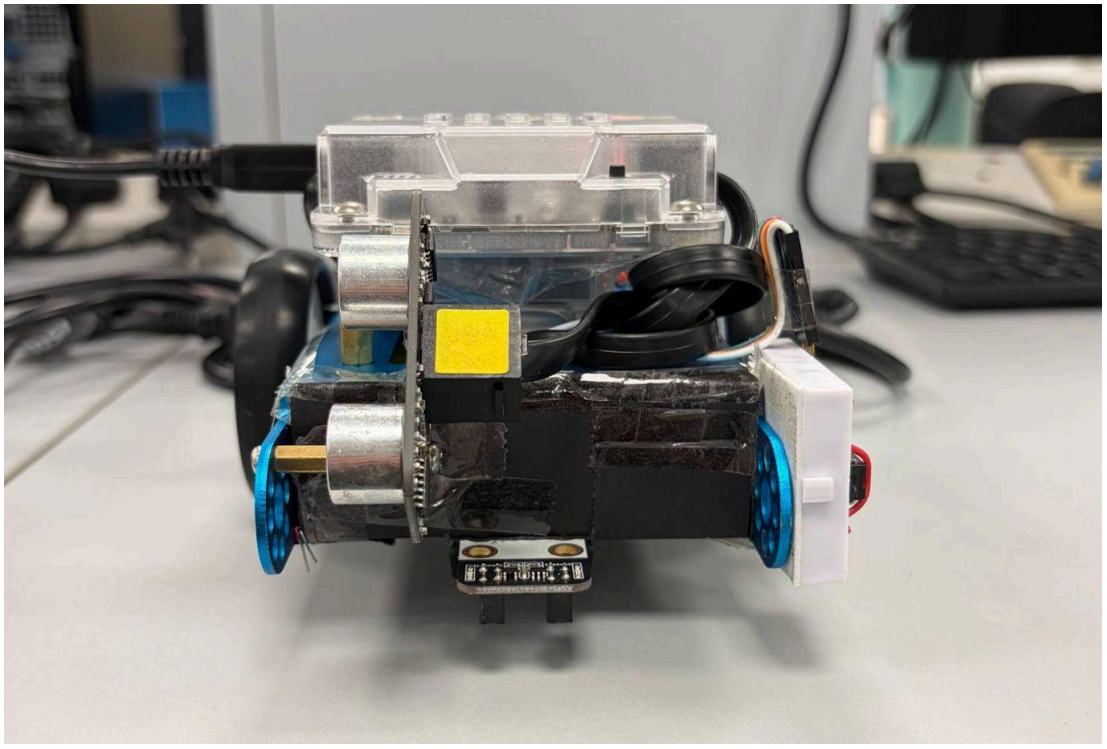


PHOTO 1: ROBOT FRONT VIEW

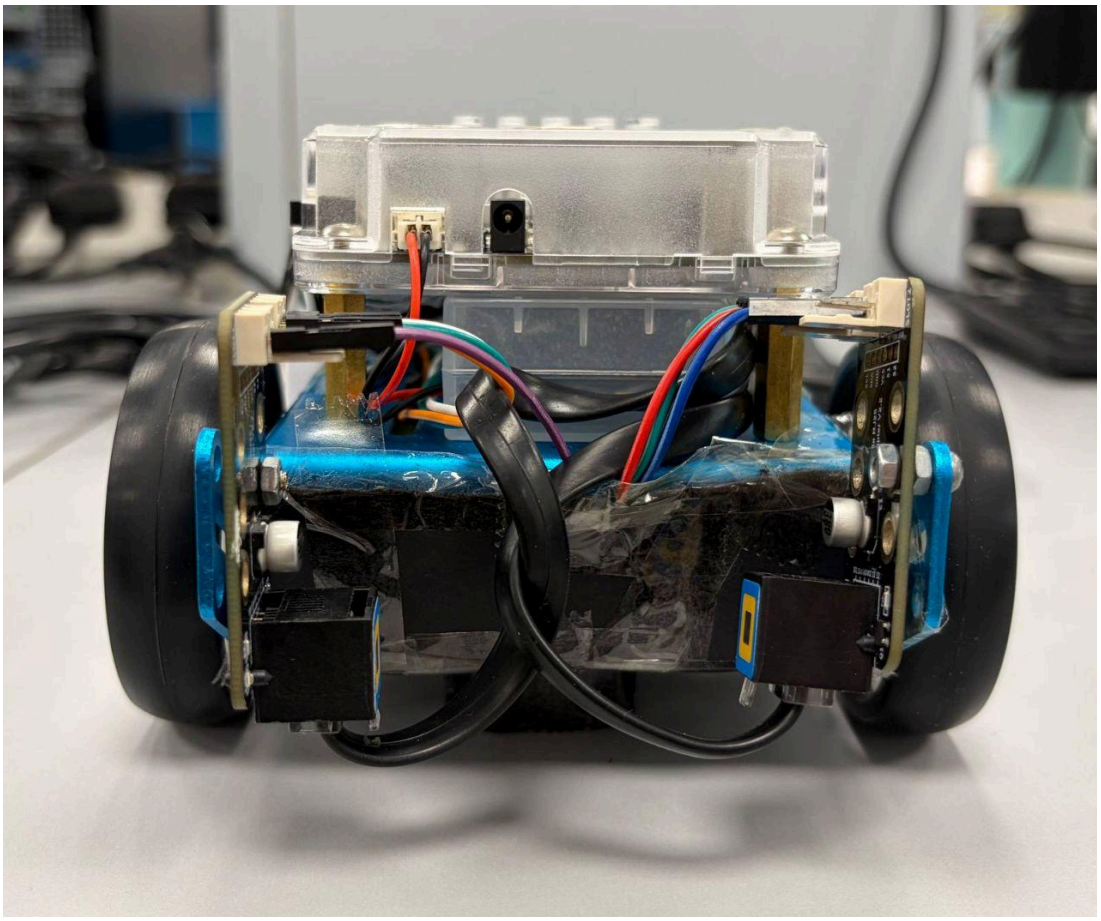


PHOTO 2: ROBOT BACK VIEW

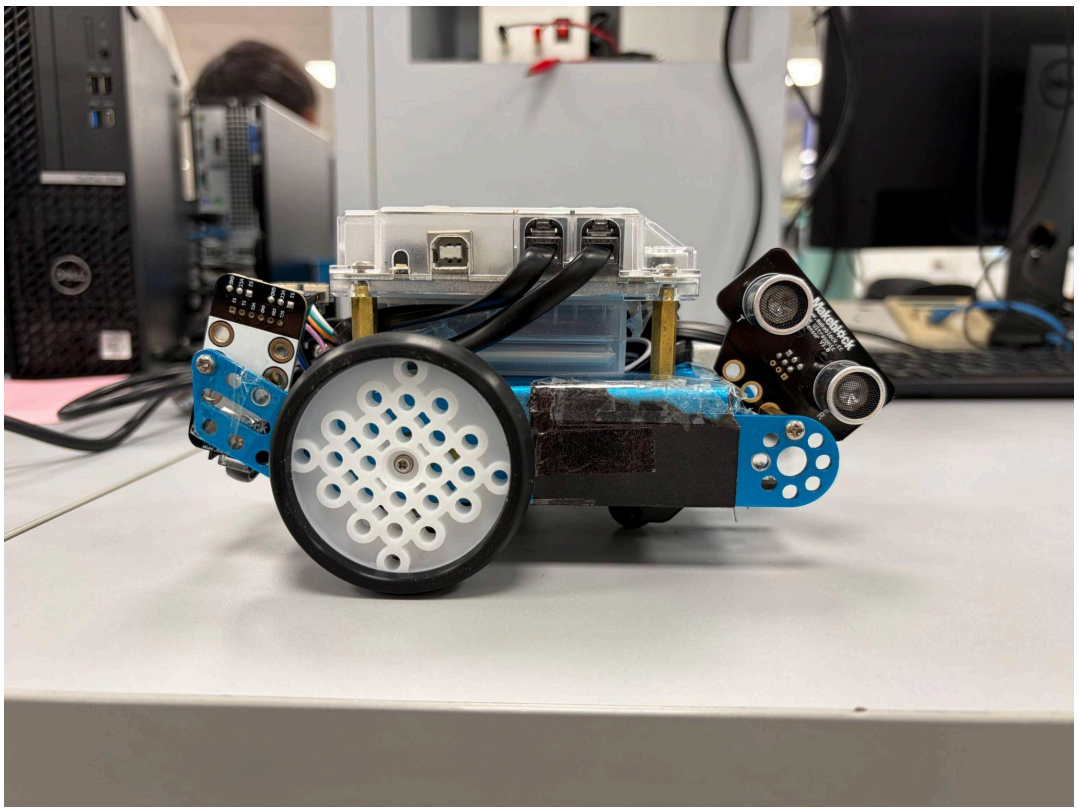


PHOTO 3: ROBOT RIGHT VIEW

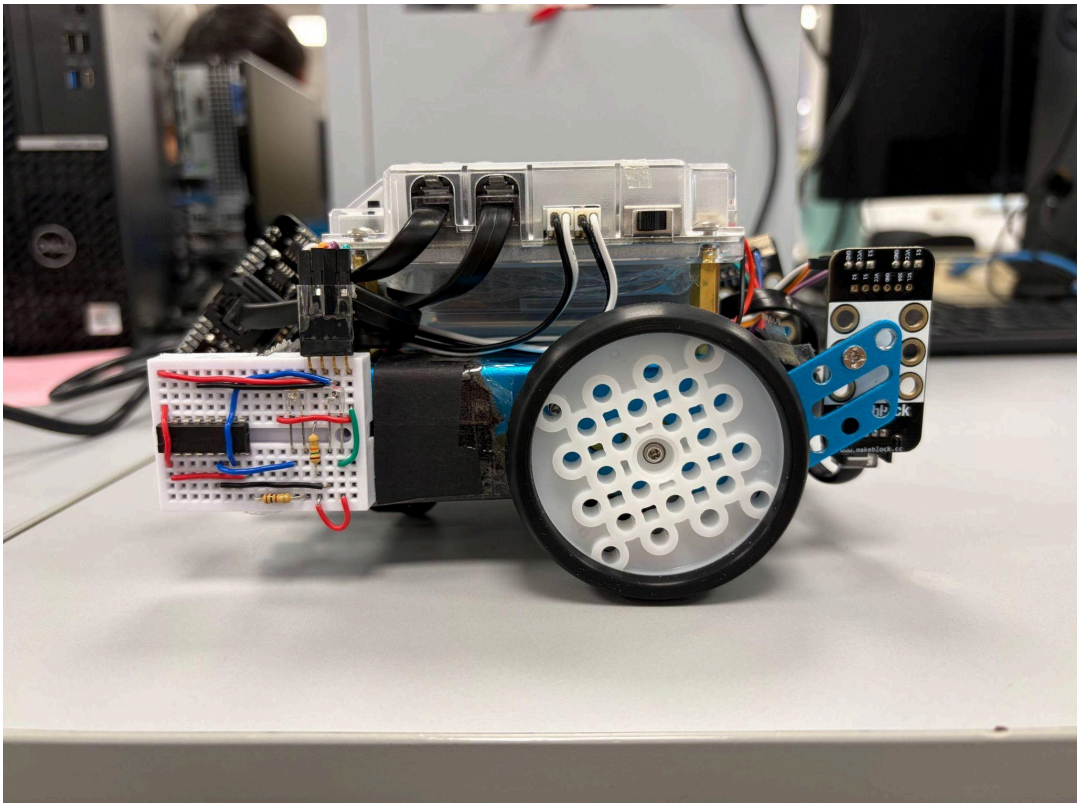


PHOTO 4: ROBOT LEFT VIEW

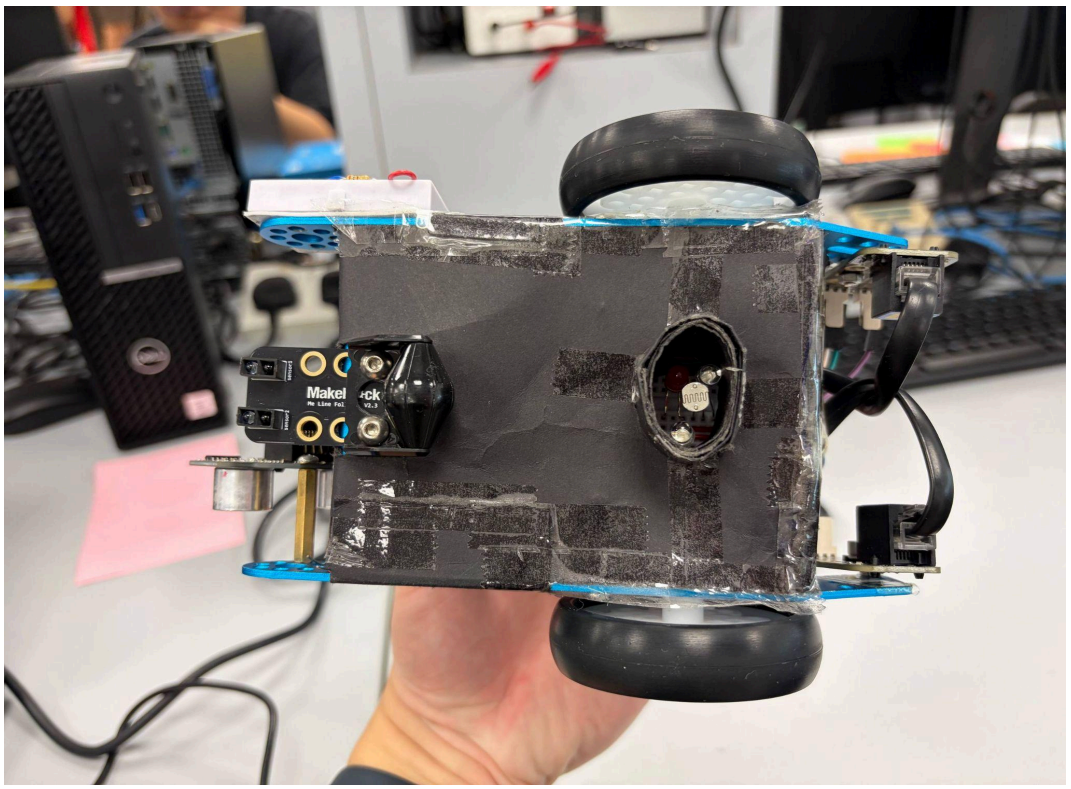


PHOTO 5: ROBOT UNDERSIDE AND COLOUR SENSOR SHIELDING

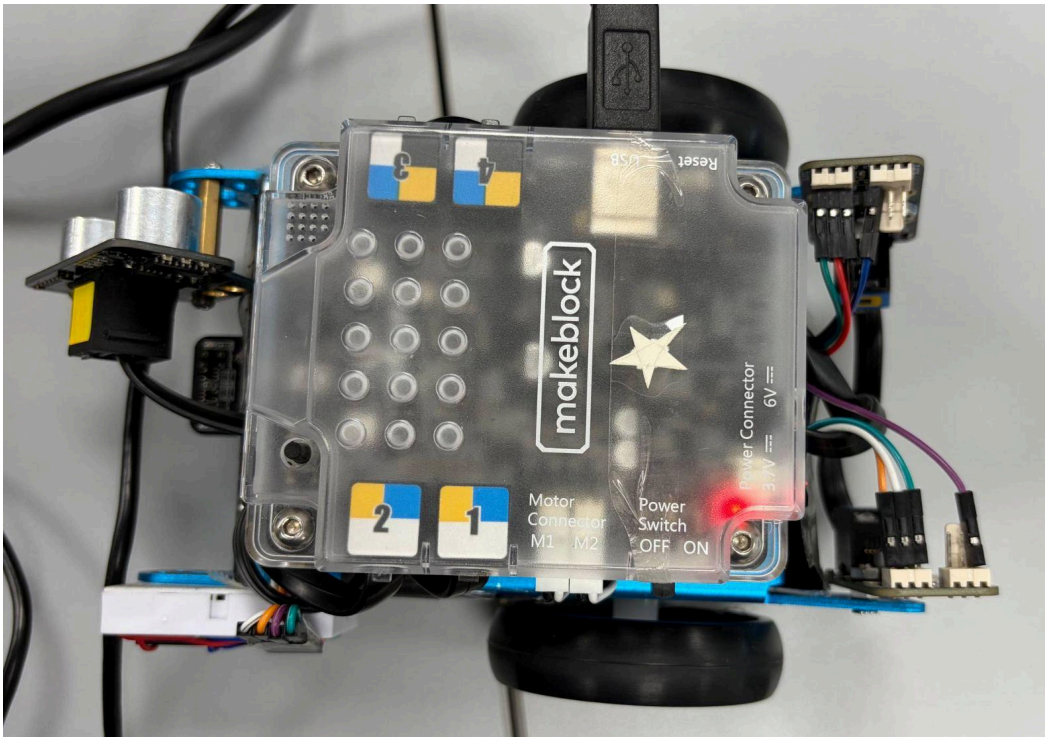


PHOTO 6: ROBOT TOP VIEW

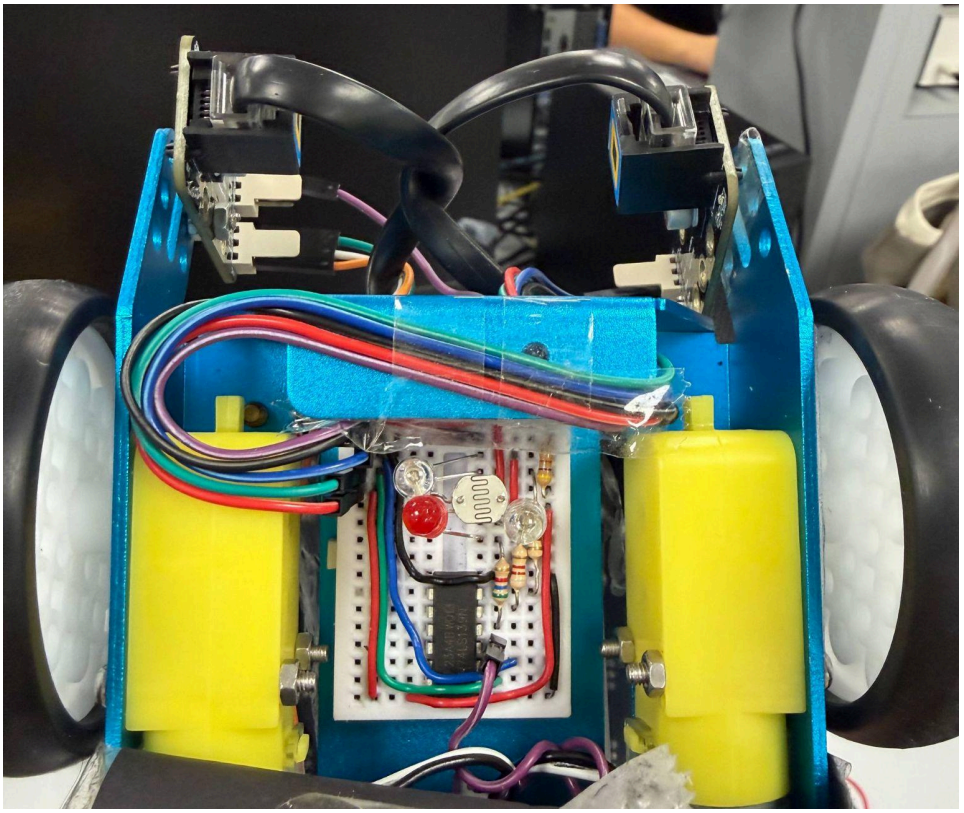


PHOTO 7: WIRING UNDERNEATH THE SHIELDING

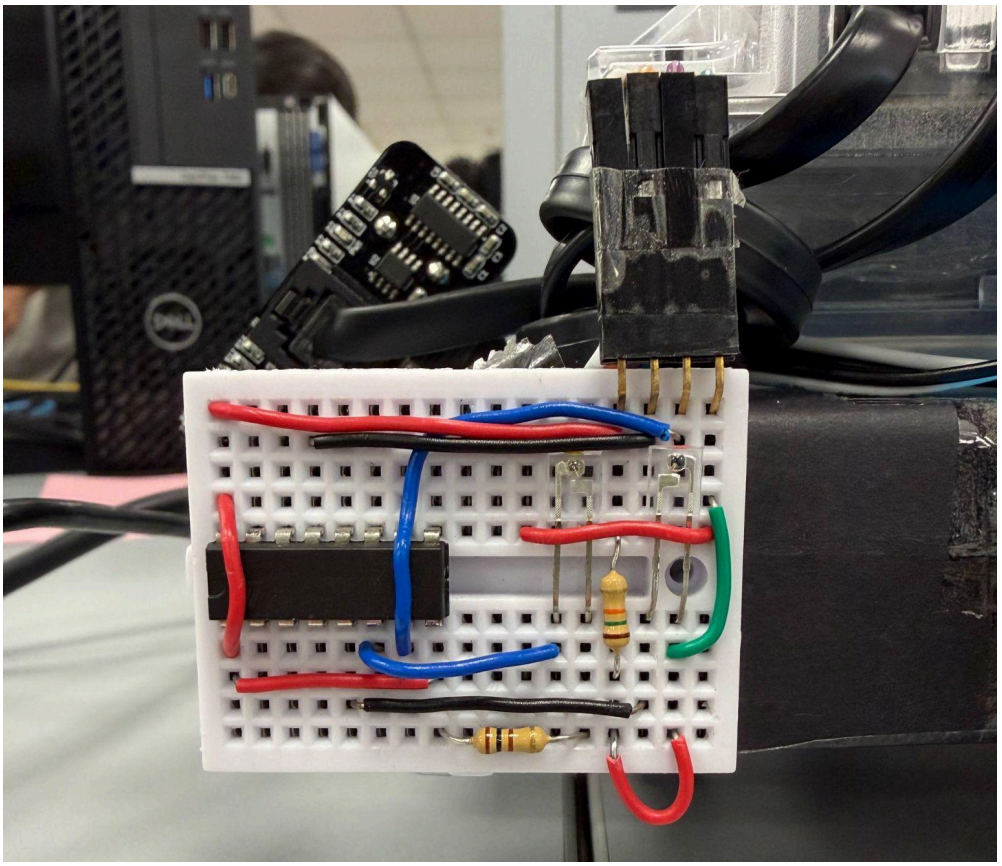


PHOTO 8: IR SENSOR WIRING

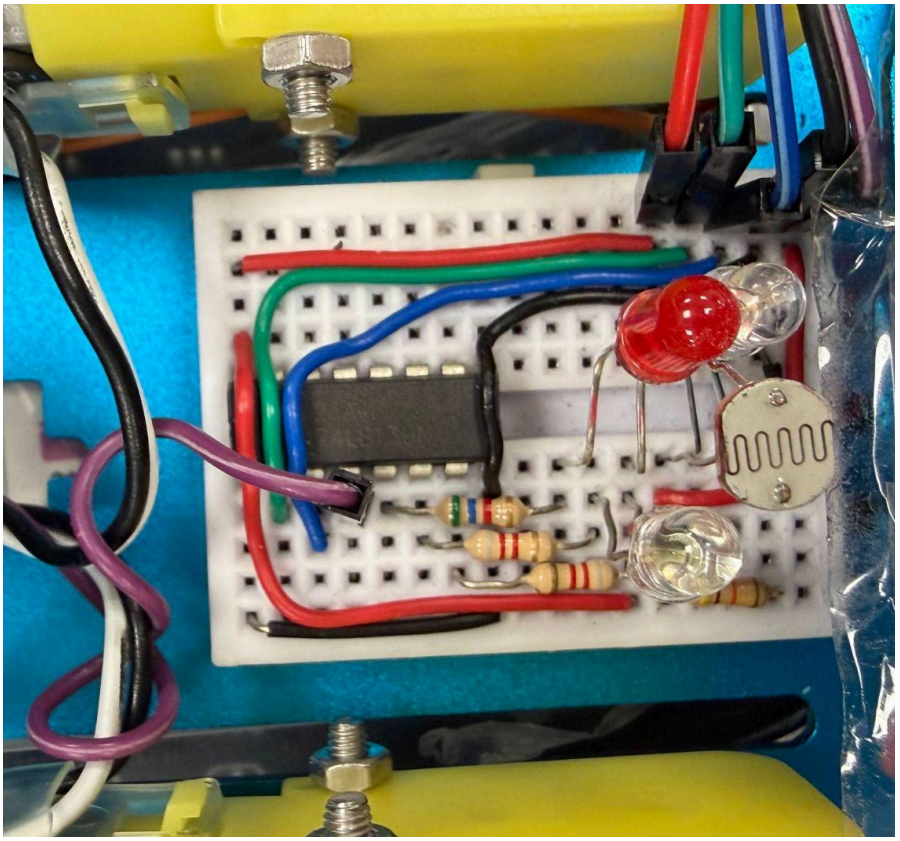


PHOTO 8: COLOUR SENSOR WIRING

SECTION 2: WORKLOAD DISTRIBUTION

Should the need arise, our team members did not hesitate to help each other with the various aspects of the robot. The table below summarises how work was distributed in our team:

Member	Robot Component	Work contributed
Zhe Ming	Infra-red Sensor	1. IR Circuit Design 2. IR Calibration 3. IR Distance Sensing Algorithm
Samuel	Colour Sensor	1. Colour Sensor Circuit Design 2. Robot Shielding 3. Colour Sensor Calibration 4. Colour Detection Algorithm
Sheryl Jin En	Ultrasonic & Line Sensors	1. Ultrasonic Sensor Calibration 2. Ultrasonic Sensor Distance Sensing Algorithm 3. Robot Centering Algorithm 4. Line Detection Algorithm 5. Colour Challenge Turning Calibration

Annex Table 1: Work Distribution Summary

SECTION 3: CREDITS

BUZZER REFERENCE CODE

HiBit. (2022, November 19). Playing popular songs with Arduino and a buzzer. Arduino Project Hub. https://projecthub.arduino.cc/tmekinyan/playing-popular-songs-with-arduino-and-a-buzzer-546f4a

Our Team would also like to acknowledge our Professor, Lab Technicians and TAs. Their guidance and feedback has helped make this project possible.