CG1111A Engineering Principles and Practice I

The A-maze-ing Race
Project Report

Team Number: B04-S2-T2
Team Members:

Name	Matriculation Number
Roderick Kong Zhang	A0286550Y
Rohan H	A0276403H
Roma Shrikant Joshi	A0821973R
Rishi Moorthy	A0237618X

Table of Contents

1 | Description of Overall Algorithm

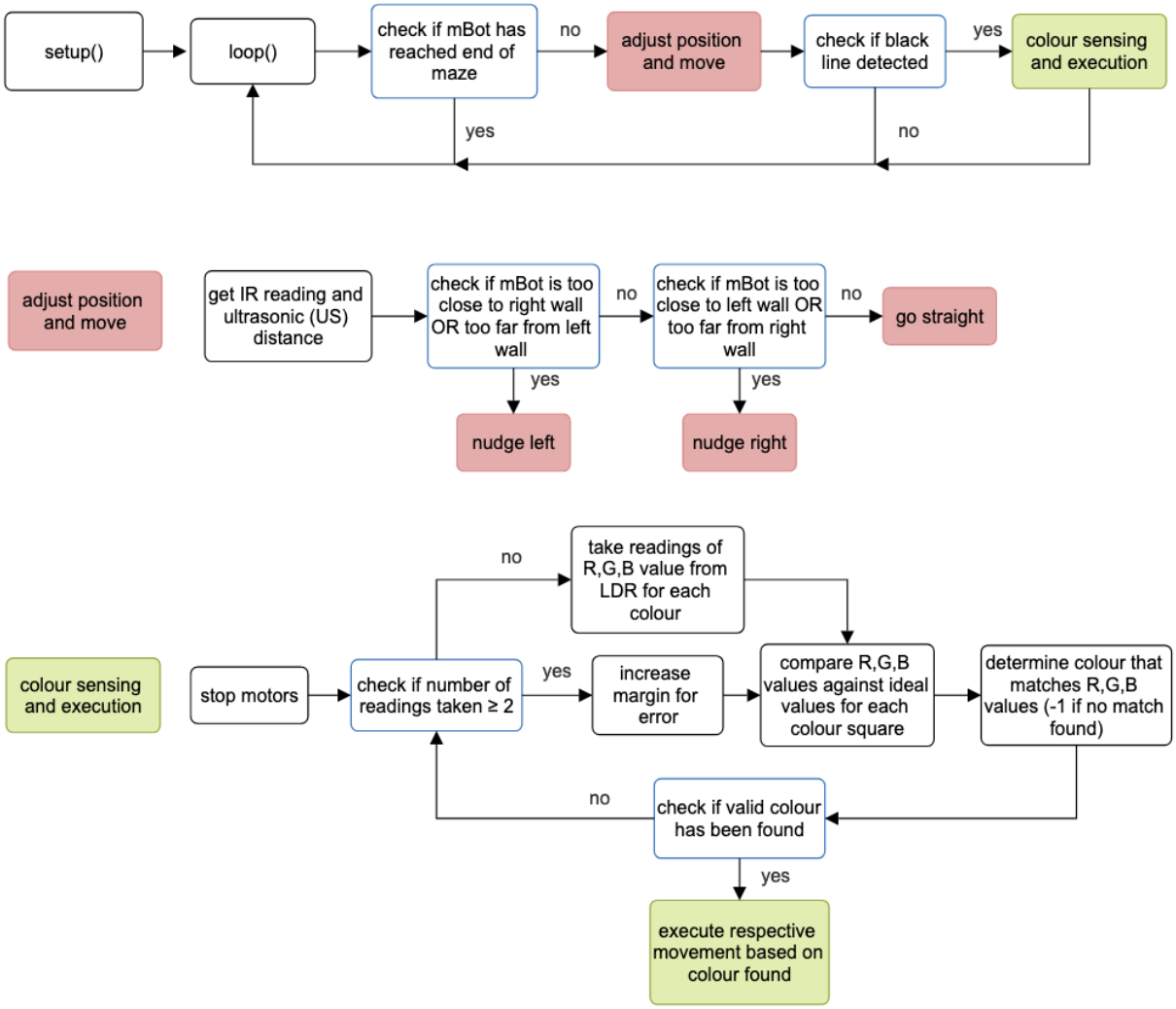

Figure 1. Flowchart of overall maze-solving algorithm

1. The mBot first checks whether it has celebrated (i.e., reached the end of the maze). If it has celebrated, stop indefinitely by returning prematurely from loop(), otherwise, continue with loop().
2. The mBot moves forward in a straight line until it detects a black strip underneath it, using a line follower. It constantly adjusts its movement by nudging left or right to move in a straight line and avoid walls. It decides whether to nudge left or right based on the distance measured using the ultrasonic sensor, and the reading obtained from the IR sensor (see *2.2 | Moving in a Straight Line* for more details).
3. If it detects a black strip, it stops and reads the colour of the paper underneath it, using a colour sensor.

4. The built-in LED display at the top of the mBot lights up to indicate which colour has been detected (e.g., if the colour red is detected, red is shown on the LED display).
5. The mBot then moves according to the colour detected (see *Figure 2*). These movements are hard-coded, and the mBot does not measure the distances and readings from the ultrasonic sensor and the IR sensor respectively during these hard-coded movements, meaning no nudging occurs.
6. Steps 1 to 5 are then repeated indefinitely.

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

Figure 2. Colour interpretation for the colour sensing challenge

2 | Implementation Details and Calibration of the Various Subsystems

2.1 | Motion

To code the turning motions of the mBot, values provided in *mBot introduction.pdf* were used initially. However, it was discovered through preliminary testing that these values were not suitable for all the mazes, due to variation in the maze surfaces (see *3.2 | Variation in Maze Surfaces*). Hence, different values for turning delays and motor speeds were experimented with, and the final values used are: a delay value of 340 milliseconds per 90° turn, and a motor speed of 220.

The following motion-related functions were defined in the code:

Function	**Corresponding Colour**	**Description**
stopMotors()	-	Causes the mBot to halt immediately. Both motors are stopped by this function.
goStraight()	-	Causes the mBot to move in a perfectly straight line when no nudging is required, until the mBot detects a black strip. Both motors move in the forward direction at the same speed to achieve this.
goForward()	-	Causes the mBot to move one grid forward. This function is only invoked in other functions, such as doubleLeft(). This is achieved by calling goStraight() and delaying for 800 milliseconds.
turnLeft()	Red	Causes the mBot to make a 90° left turn. The left motor moves backwards while the right motor moves forwards, with both motors running at the same speed, to achieve this stationary turn. The turning delay used is 340 milliseconds.
turnRight()	Green	Causes the mBot to make a 90° right turn. The left motor moves forwards while the right motor moves backwards, with both motors running at the same speed, to achieve this stationary turn. The turning delay used is 340 milliseconds.

doubleLeft()	Purple	Causes the mBot to turnLeft(), goForward(), then turnLeft() again.
doubleRight()	Blue	Causes the mBot to turnRight(), goForward(), then turnRight() again.
spin()	Orange	Causes the mBot to make a 180° turn on the spot. The left motor moves backwards while the right motor moves forwards, with both motors running at the same speed, to achieve this stationary turn. The turning delay used is 680 milliseconds (340 milliseconds * 2).
nudgeLeft()	-	Causes the mBot to veer leftwards to avoid colliding with the right wall. The right motor moves faster than the left motor by 69 units to nudge left. This function is used in the algorithm that ensures that the mBot moves in a straight line, by utilising the ultrasonic sensor and the IR sensor (to be discussed in *2.2 | Moving in a Straight Line*).
nudgeRight()	-	Causes the mBot to veer rightwards to avoid colliding with the left wall. The left motor moves faster than the right motor by 69 units to nudge right. This function is used in the algorithm that ensures that the mBot moves in a straight line, by utilising the ultrasonic sensor and the IR sensor (to be discussed in *2.2 | Moving in a Straight Line*).

Figure 3. Table illustrating each motion-related function, their related colour, and purpose

2.2 | Moving in a Straight Line

The mBot is able to move in a straight line as it constantly adjusts its movement by nudging left or right based on the distance measured using the ultrasonic sensor, and the reading obtained from the IR sensor. The specifics of the IR sensor, ultrasonic sensor, and nudging algorithm are mentioned below.

2.2.1 | Ultrasonic Sensor

The ultrasonic sensor detects the distance between the sensor and the wall on its right, by measuring the time taken for an emitted ultrasonic pulse to bounce off the wall, back to the sensor. The following formula was used to calculate the distance:

$$distance\ (cm)\ =\ \frac{0.0345\ (cm/\mu s) \times time\ taken\ (\mu s)}{2}$$

where 0.0345 cm/μs is the speed of sound measured at 23℃.

2.2.2 | IR Sensor

2.2.2.1 | Explanation of Components

The IR emitter and IR detector are placed side by side on the mBot breadboard circuit, facing the left wall (see *Figure 4*). The emitted IR light is reflected by the left wall, and the distance to the left wall affects the amount of IR light detected. The IR detector's output voltage decreases as the distance from the wall also reduces.

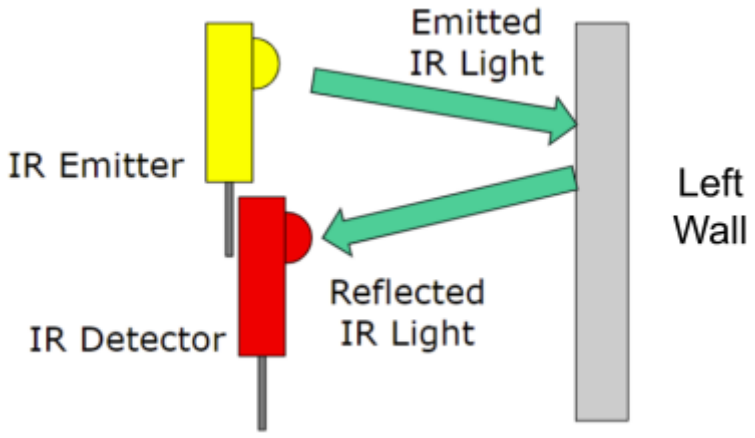

Figure 4. Diagram illustrating the positions of the IR emitter, IR detector, and left wall

2.2.2.2 | Circuit Design

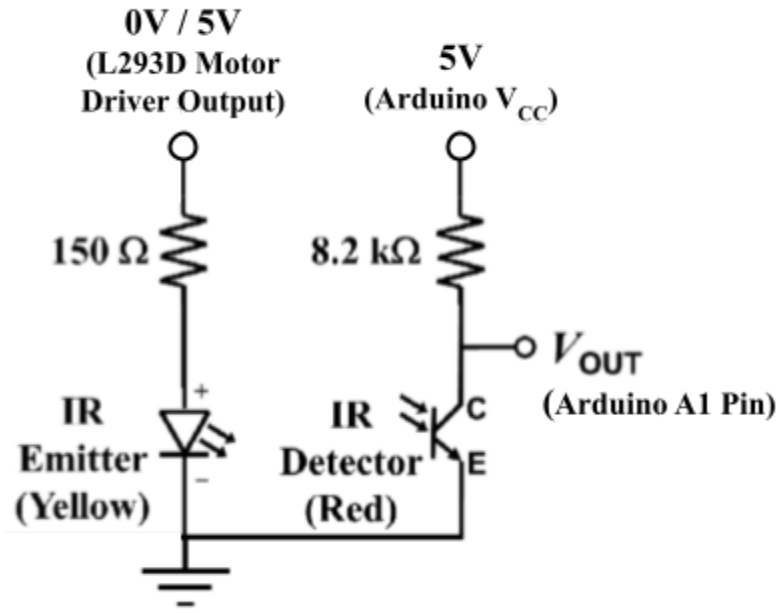

Figure 5. Circuit diagram of IR sensor

The IR sensor (V_{out}) terminal is connected to an individual analog pin (A1) on the Arduino to measure the voltage drop across the detector.

The circuit with the IR detector is wired directly to the 5V pin of the Arduino, so that the IR detector is always turned on and is detecting IR light.

The circuit with the IR emitter is wired to the output pin of the L293D motor driver. This allows the IR emitter to be switched on and off by toggling the input pin of the motor driver between high and low voltages. The goal of this circuit configuration is for the IR detector to be able to detect both the ambient IR reading (when the IR emitter is turned on), and the wall IR reading (when the IR emitter is turned off). The difference between the ambient IR reading and the wall IR reading is used for the nudging algorithm, to determine the mBot's position relative to wall (see *2.2.3.2 | getIRDetectorReading()*).

2.2.3 | Nudging Algorithm

2.2.3.1 | getUltrasonicDistance()

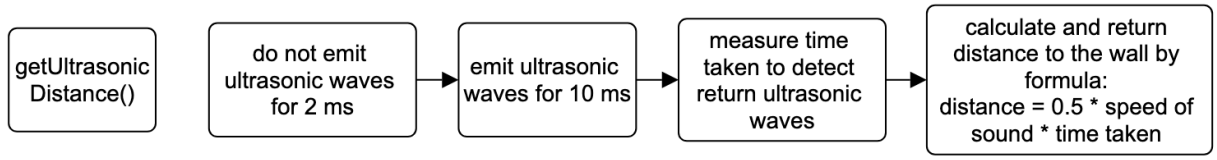

Figure 6. Flowchart of ultrasonic sensor function

This function returns the distance measured by the ultrasonic sensor, between the sensor and the wall.

2.2.3.2 | getIRDetectorReading()

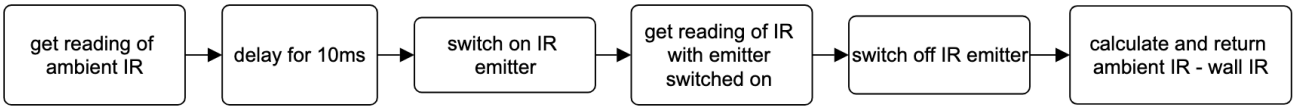

Figure 7. Flowchart for IR sensor function

This function returns the distance measured by the infrared detector. First, a reading of ambient infrared radiation is taken. Then, after a 10-millisecond delay, the infrared emitter is switched on for some time and then switched off. Some of the infrared light it emits will reach the wall and be reflected back to the infrared detector. A reading of the infrared detector is taken. The final IR reading is calculated by finding the difference between ambient IR readings and the detected IR reading. This increases the accuracy of the IR sensor, as it will be able to detect distances regardless of the ambient lighting conditions.

2.2.3.3 | Sensor Calibration

A ruler was used to measure what the ideal distance should be between the ultrasonic sensor and the wall of the maze (calibrated to be 10 cm). The maximum distance that the ultrasonic sensor should detect based on the length of a grid was also measured with a ruler (calibrated to be 20 cm).

To calibrate the IR sensor, the value of the ideal IR reading was varied until the ideal IR reading was detected when the mBot is equidistant from the left and right walls (calibrated to be 6 units).

2.2.3.4 | Decision-Making Based on Ultrasonic Sensor Distance and IR Reading

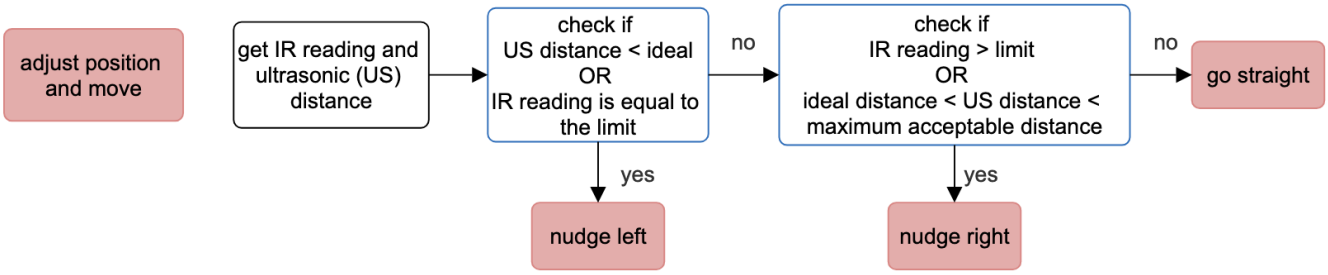

Figure 8. Flowchart of algorithm that ensures that the mBot moves in a straight line

The nudging algorithm only takes into account reasonable values obtained from the ultrasonic and IR sensors. This is implemented so that the algorithm will not consider readings that are taken by the IR and ultrasonic sensors when the left and right walls are absent respectively. For the ultrasonic sensor, only distances obtained that are below the maximum acceptable distance are considered. For the IR sensor, only IR readings that are more than or equal to the IR reading limit are taken into consideration.

If the distance returned by the ultrasonic sensor is below the ideal distance (i.e., mBot is too close to the right wall), or the IR reading is equal to the limit (i.e., mBot is too far from the left wall), the mBot nudges left.

Else, if the IR reading is above the limit (i.e., mBot is too close to the left wall), or the distance returned by the ultrasonic sensor is above the ideal distance but below the maximum acceptable distance (i.e., mBot is too far from the right wall), the mBot nudges right. Therefore, this algorithm ensures that our mBot does not collide with the walls of the maze.

2.3 | Black Strip Detection

Each IR emitter of the line follower continuously shines IR rays onto the surface below the mBot. Black surfaces reflect much less infrared light compared to other coloured surfaces, hence the line follower can determine whether the surface below it is a black strip by detecting the amount of reflected IR light. The mBot checks whether there is a black strip in every iteration of the main loop function. Once a black strip is detected, the mBot stops and reads the colour beneath it (see *Figure 9*).

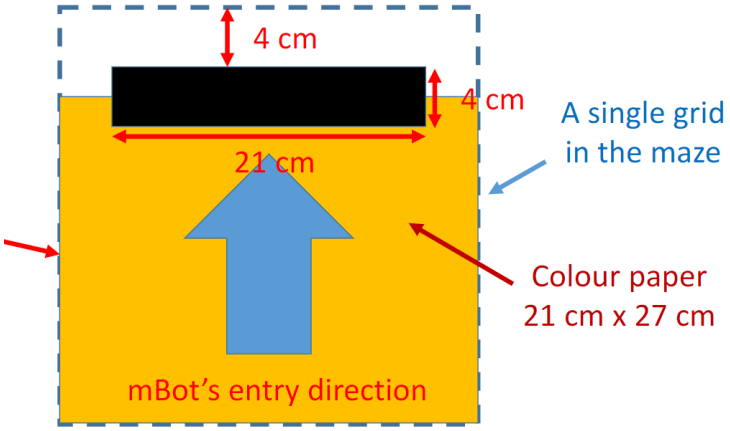

Figure 9. Position of black strip relative to coloured paper on the maze

To determine whether the black strip has been detected, the line follower only checks whether Situation 1 has occurred. This is achieved by checking if the sensor state is *S1_IN_S2_IN*, and the mBot will be stopped if this condition is true.

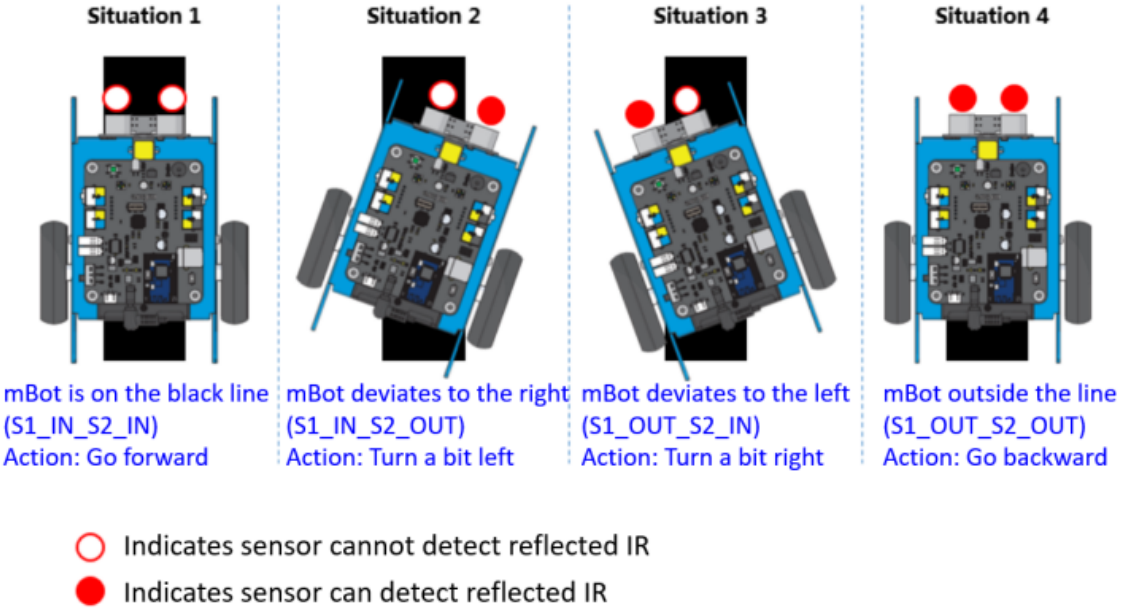

Figure 10. Four possible situations reported by the line follower
(Picture credit: http://mBlock.cc)

2.4 | Colour Sensing

2.4.1 | Basics of Colour Sensing

When the colour sensing function is triggered, the mBot circuitry lights up the red, green and blue LEDs via the 2-to-4 converter. When each LED is lit, the light reflected off of the colour paper is read by the LDR and the R, G, B values for the colour square are obtained. These values are then compared against the ideal R, G, B values for each of the colour squares to get the closest match.

2.4.2 | Colour Changing Algorithm

The three LEDs were wired to the outputs Y_1, Y_2, Y_3 of the 2-to-4 converter respectively. The A2 and A3 pins of the mBot are connected to the 2 input pins of the converters. By outputting different combinations of HIGH and LOW through the two input pins, it is possible to light one LED at a time.

2.4.3 | Colour Sensing Circuit

As can be seen from Figure 11, the green and blue wires are connected to the pins A2 and A3, and go into the A and B inputs of one side of the converter. The enable pin of that side, as well as the GND pin of the converter, are connected to the GND of the mBot. The three outputs Y_1, Y_2, and Y_3 are connected to the cathodes of the red, green, and blue LEDs respectively, and their anodes are connected to a common resistor and then to 5V. This is because the 2-to-4 converter lets V_{CC} be the default value of the output pins. Only one pin can be grounded at a time. Hence, when Y_1, Y_2 or Y_3 are grounded from the combination of A and B inputs, current flows from V_{CC} through the respective LED into Y_1, Y_2 or Y_3.

The output Y_0 is connected to the VCC pin of the motor driver that operates the IR sensor.

The LDR is also installed next to the 3 LEDs in series with another resistor. The A0 pin of the mBot is connected between the LDR and the normal resistor with the brown wire and measures the voltage drop across the LDR, which is used to calculate its resistance.

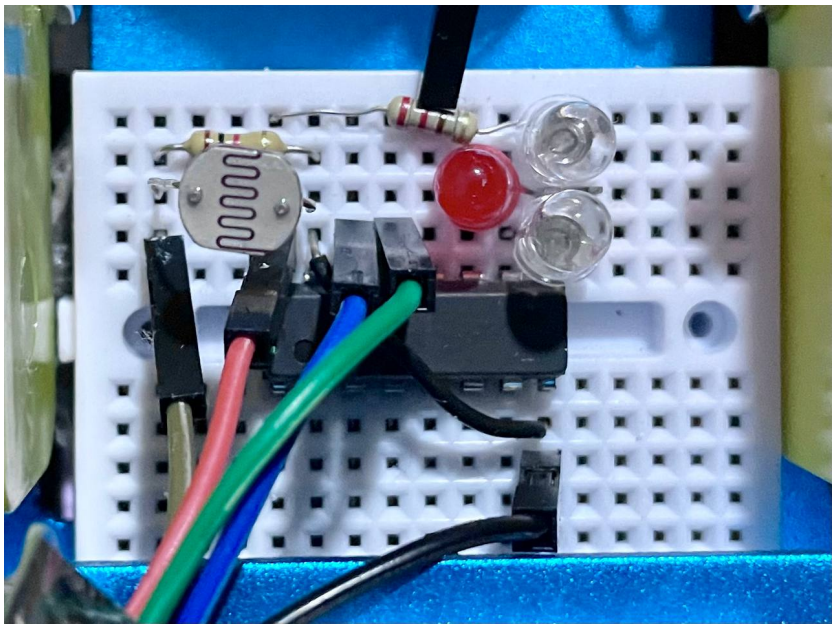

Figure 11. Exposed colour sensor circuit

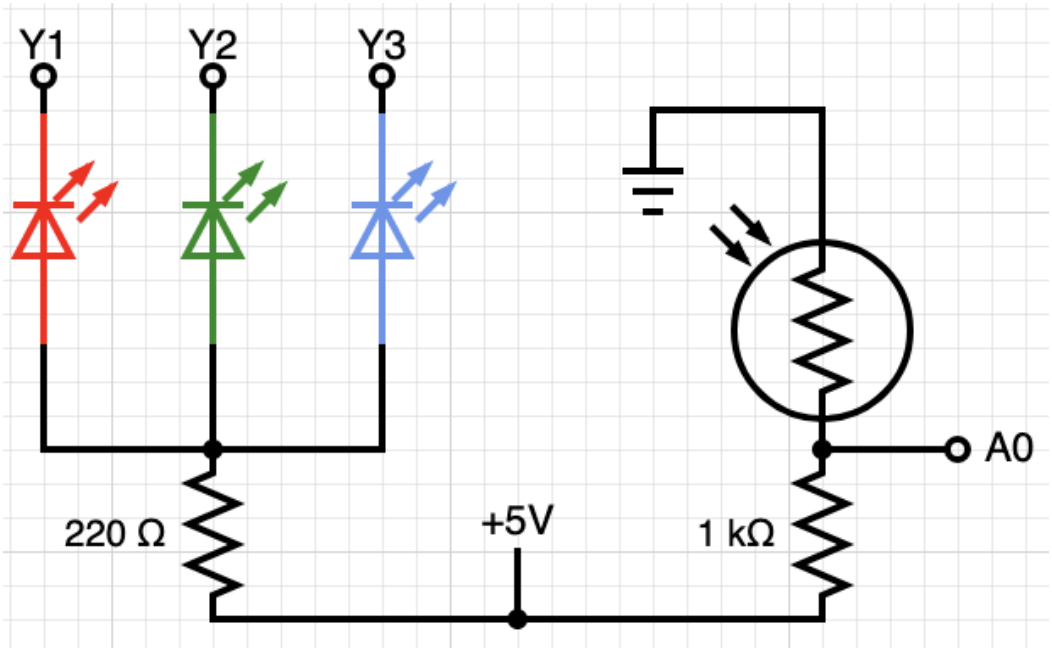

Figure 12: Circuit diagram for the colour sensing circuit

2.4.4 | Colour Detection Algorithm

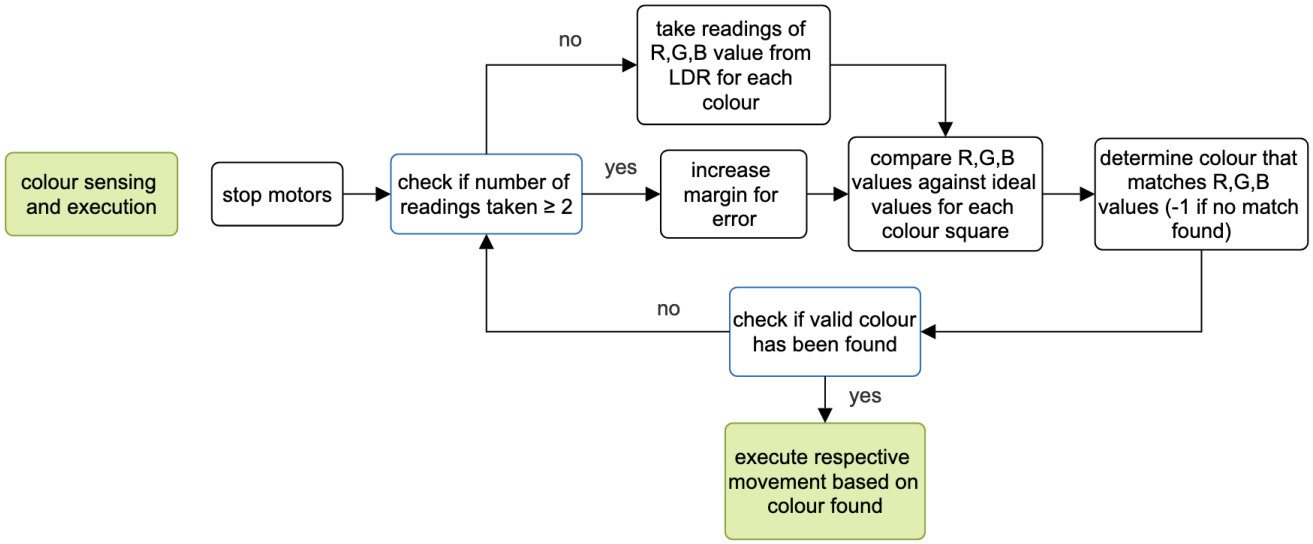

Figure 13. Flowchart for colour sensing algorithm

When the colour detection function is triggered, the code enters a while loop that runs until a valid colour is found. The mBot will only take a maximum of 2 readings in order to save time; the second reading is taken in case the first reading is an outlier. Once a reading of the R, G, B values is taken, they are compared against the ideal values for every colour square within a certain error margin. After the second reading is taken, the margin of error is increased with every iteration such that the code will eventually determine the colour that matches the closest with the measured values. Once a valid colour is found, the mBot will execute the command that corresponds to that colour.

2.4.5 | Colour Decision Calibration

The ideal RGB values for each colour, which were taken from the coloured sheets from the final maze area, are given below:

Colour/RGB	RED (R)	GREEN (G)	BLUE (B)	RGB Code Colour
Red	225	57	42	
Orange	249	116	54	
Green	53	112	59	
Blue	70	166	223	
Purple	107	107	150	
White	251	255	255	

Figure 14. RGB values detected by colour sensor for each coloured sheet

The raw R, G and B values that are measured by the LDR are first processed by accounting for colour balance, using the formula

$$processed\ value\ =\ 255\ \times\ \frac{raw\ value\ -\ value\ for\ black}{grey\ diff}$$

where the grey *diff* of a colour is defined as the (*value for white – value for black)* for that colour. The values of *grey diff* and *value for black* for each colour are shown below, based on the values the LDR read for pure black and pure white under lighting conditions of the final maze.

Colour/RGB Values	RED (R)	GREEN (G)	BLUE (B)
Value for white	223	287	218
Value for black	85	114	97
Grey diff	138	173	121

Figure 15. RGB values detected by colour sensor for pure white, pure black, and grey diff

2.5 | Celebratory Tone

Each note was defined as its frequency in Hertz (e.g., A4 as 440Hz), and each note duration as its duration in milliseconds (e.g., a semiquaver as 125) in the code, as the buzzer.tone() function takes in two parameters; the note frequency in Hz, and the duration in milliseconds.

The entire chorus of Rick Astley's "Never Gonna Give You Up" was then transcribed into our code, to be used as our celebratory tone.

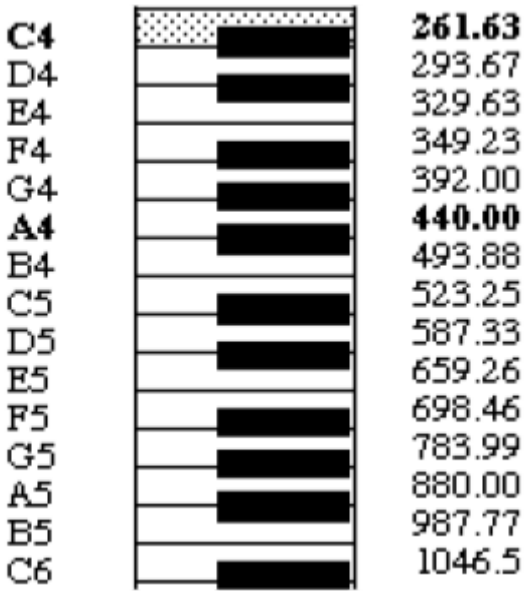

Figure 16. Diagram showing musical notes and their corresponding frequencies in Hz

3 | Difficulties Faced and How They were Overcome

3.1 | Lighting Conditions

Since the lighting in both mazes was different, the colour detection process was affected. Hence, we were prone to detecting the wrong colour in some instances. To overcome this, we fully shielded the underside of the mBot with black paper so that no external light could reach the LDR, increasing the accuracy of the results.

3.2 | Variation in Maze Surfaces

The friction between the wheels and the table surface varied between the mazes. This meant that the angles of the turns (especially the double-lefts and the double-rights) varied too much between each maze, and sometimes even led to the mBot crashing into the wall. Since we were not allowed to modify the code in the mBot between mazes, we could not modify the turn delays for each maze. Hence, we attempted to strike a balance between the turning delays of the different mazes, made the nudge-checking more frequent, and increased the nudging angle so that it could correct the over/underturning of the mBot.

3.3 | Positioning of the LDR

The LDR is rather sensitive with respect to its position, and the mBot's collisions with the walls would jar it, which affected the amount of light of each colour it received. To counter these offsets, we secured it in place using a mounting putty so that our readings became more accurate.

3.4 | Debugging Colour Read by Colour Sensor

To better debug what colour the mBot sensed, we used the LED display to showcase the colour detected by the LDR, to determine the colour being sensed. Since we shielded the robot very well, it was not possible to see when the mBot was reading colour and when it was not. To overcome this we used the built-in LED display to show a light pink colour whenever the colour was being read. This made it easy for us to know how many checks it took for the colour sensor to detect the colour.

4 | Pictures of the mBot

4.1 | Top and Bottom Views

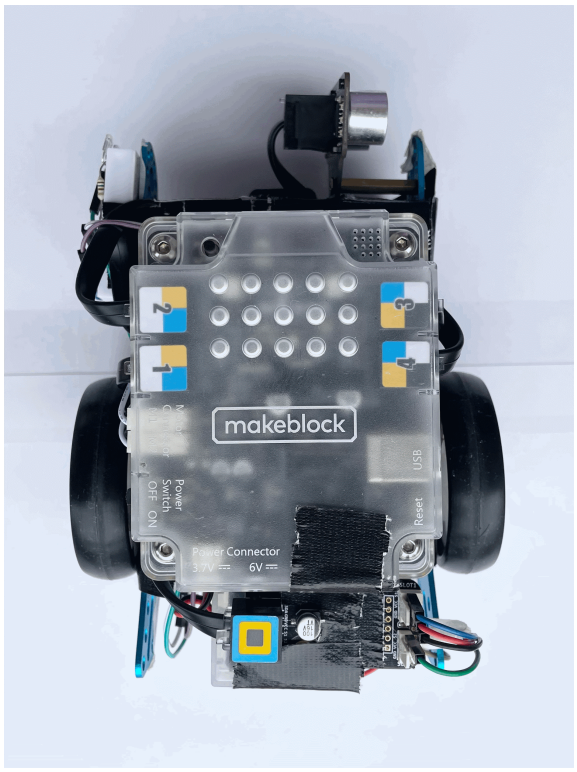

Figure 17. Top view of mBot

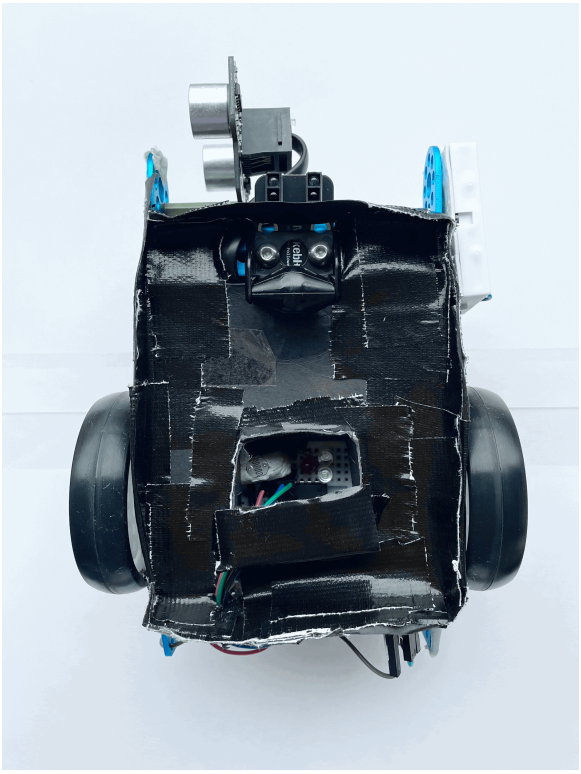

Figure 18. Bottom view of mBot

4.2 | Front and Back Views

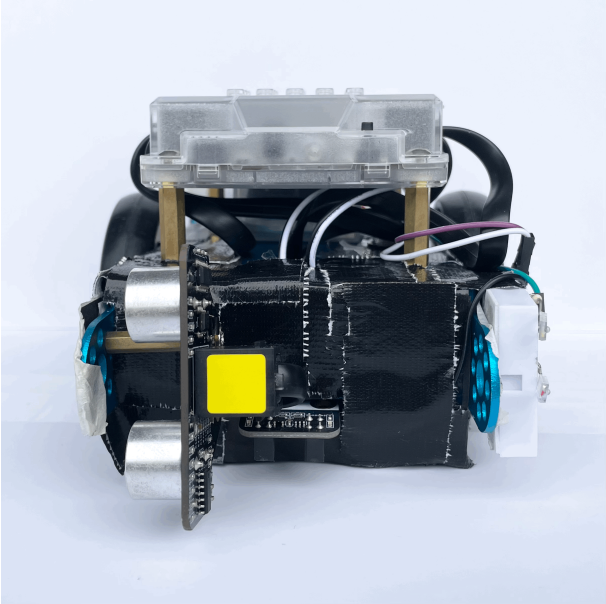

Figure 19. Front view of mBot

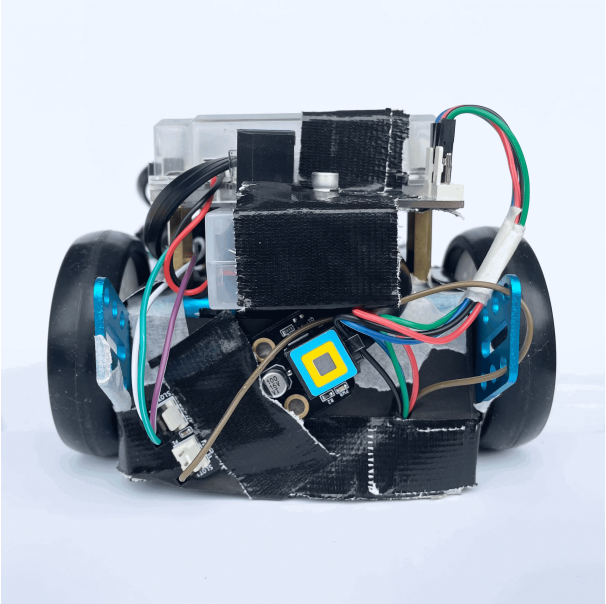

Figure 20. Back view of mBot

4.3 | Side Views

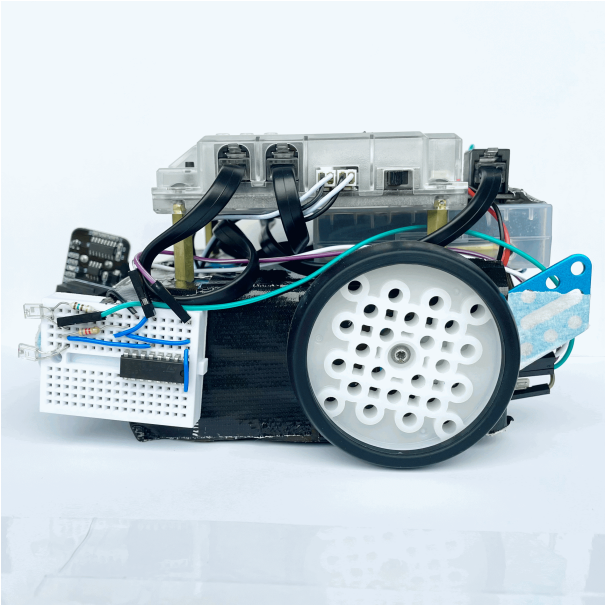

Figure 21. Left side view of mBot

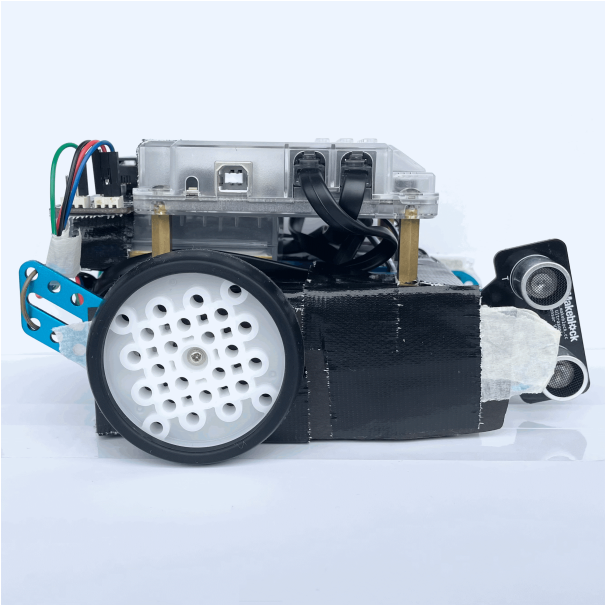

Figure 22. Right side view of mBot

5 | Pictures of Sensor Breadboard Circuits

5.1 | Colour Sensor

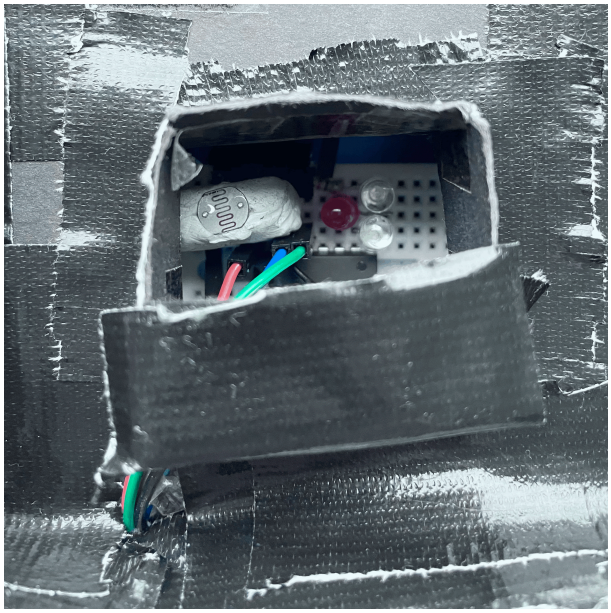

Figure 23. Shielded colour sensor circuit with mounting putty on LDR for stability

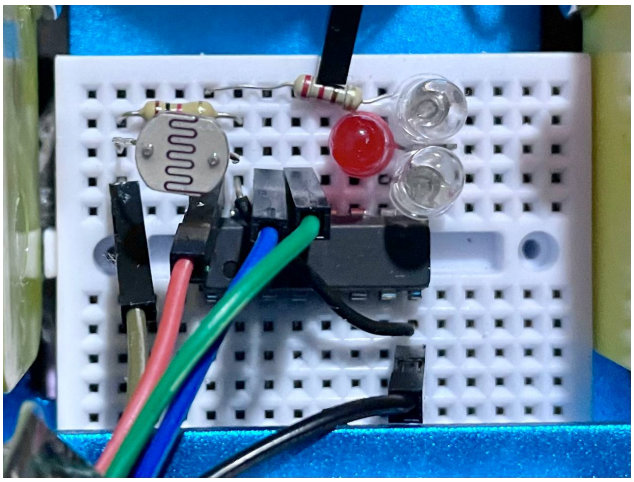

Figure 24. Exposed colour sensor circuit

5.2 | Picture of the IR sensor circuit

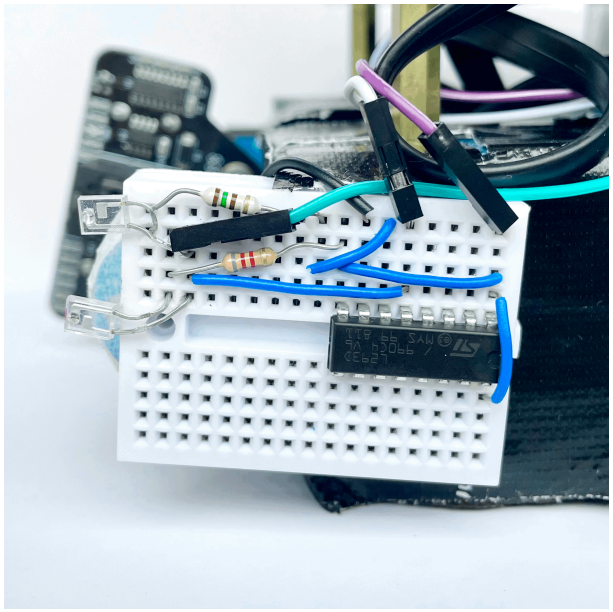

Figure 25. IR sensor circuit

6 | Team Workload Division

Name	Category	Work Allocated
Roderick Kong Zhang	Breadboard Circuits	• Editing the breadboard circuit for the IR sensor, ensuring neatness of wiring

	Code and Algorithms	• Nudging code and algorithm for mBot to move in a straight line
		  ○ Code to get the current distance from sensor to the wall using the ultrasonic sensor
		  ○ Code and algorithm to get the current IR reading from the IR detector (difference between ambient IR reading and wall IR reading)
		• Code for black strip detection
		• Code for celebratory tone
		• Code and algorithm for colour sensor
		• Documented source code with comments

Rohan H	Code and Algorithms	● Code, algorithms for all motion functions of mBot (decreasing delays and sharpening turns) ● Code for mBot LED display to debug colours read by the colour sensor in the maze
	External Hardware	● Full physical assembly of mBot
Roma Shrikant Joshi	Breadboard Circuits	● Assembling the breadboard circuit for the colour sensor and IR, ensuring neatness of wiring
	Code and Algorithms	● Code and algorithm for colour sensor
	External Hardware	● Creating the colour sensor skirting and shielding to block it from external light
Rishi Moorthy	External Hardware	● mBot skirting and shielding ● Procuring materials needed for the shielding (black coloured paper, tape, etc.).
	Code and Algorithms	● Code for the optimisation of motions of the mBot (decreasing delays and sharpening turns)

Figure 26. A table of our contributions to the project