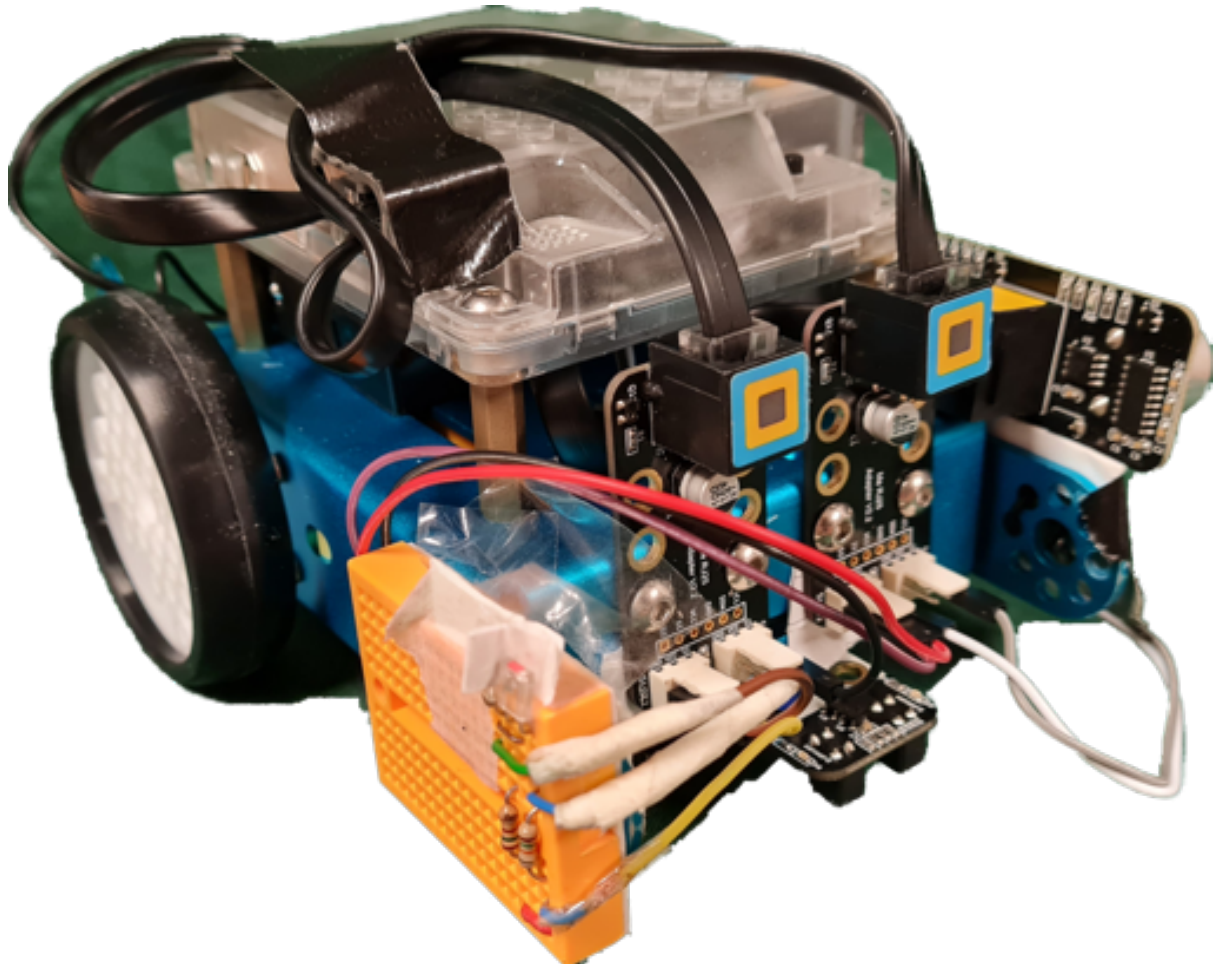


CG1111A MBot Project Report



Studio Group: B02

Section: 3

Team: 2

- **Koh Jing Jie Marcus**
- **Koh Ngiap Hin**
- **Lee Jun Hao Bryan**
- **Lee Jun Hui Ansenn**

Table of Contents

Code and Algorithm	3
Ultrasonic Sensor	8
IR(Infra - Red) Sensor	9
Building our Sensors	10
Difficulties Encountered	10
Work Delegation	16

Code and Algorithm:

The Mbot is programmed to move in a straight line using the IR sensor and ultrasonic distance sensor placed on the right and left side respectively to adjust its course if it is too close to one side of the wall.

When the Mbot detects a black strip (via the black line sensor located at the bottom of the Mbot), it would stop moving and start the color sensing algorithm to detect what is the most likely color of the colored paper underneath it.

The resistance of the LDR increases when the intensity of the light decreases. For each object whose color is to be detected, the three colored LEDs (red, blue and green) are turned on sequentially and data is recorded as an analog input. Our code is implemented such that the LDR would take in the voltage values when the R, G, B LED shines on the colored paper then adjust the value based on the white balance and black balance of our circuit and converts the value to between the range 0 to 255 and turn accordingly (Figure.1) to the values obtained.

For the fine tuning of the colors, we made use of the color-test code to measure the RGB values of each color and calibrate accordingly to the range that falls within the color.

After decoding the color of the paper and turning, the Mbot will continue moving in a straight line. This loop will continue until it detects a white paper underneath it.

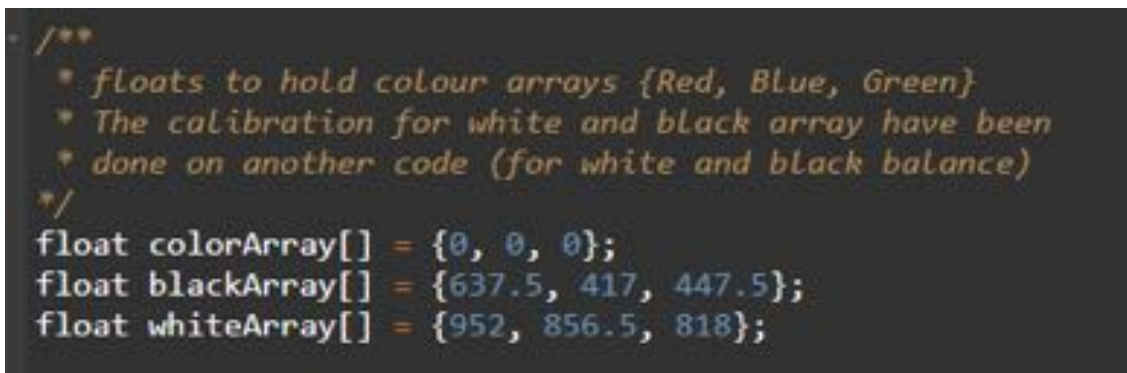
Upon detection of white paper, a short tune would be played, signalling that the Mbot has successfully reached the end of the maze.

Color	Direction to turn
Red	Left-turn
Green	Right-turn
Orange	U-turn
Purple	Two left turn in two grids
Blue	Two right turns in two grids
White	Plays a tune (Final tile)
Black line	stops

Figure 1: color and its respective commands

Pseudocode of the algorithm we used:

```
void loop() {  
    Left and right motors move forward;  
    if (ultrasonic distance sensor on left too close) {  
        Adjust course slightly to the right;  
    }  
    if (IR sensor on right too close) {  
        Adjust course slightly to the left;  
    }  
    if (Black line detector detects black strip) {  
        Check color;  
        if (color == red/orange/green/purple/light blue) {  
            Turn according to the table below;  
        }  
        if (color == white) {  
            Play victory music; //Maze completed  
        }  
    }  
}
```



```
/**  
 * floats to hold colour arrays {Red, Blue, Green}  
 * The calibration for white and black array have been  
 * done on another code (for white and black balance)  
 */  
float colorArray[] = {0, 0, 0};  
float blackArray[] = {637.5, 417, 447.5};  
float whiteArray[] = {952, 856.5, 818};
```

Figure 2: Initialisation of an Array to store the R,G,B values respectively

- colorArray[0] == Red Value
- colorArray[1] == Blue Value
- colorArray[2] == Green Value

```

void colorTurn()
{
    //for turning, motor speed for the two motors are not the same
    //so the delay for each turn and adjustment are not the same
    if (colorArray[0] > 240
        && colorArray[1] < 200
        && colorArray[2] < 200
        && colorArray[2] > 135) //(red)
    {
        //turn left
        motor1.run(motorSpeed);
        motor2.run(motorSpeed);
        delay (400);
    } else if (colorArray[0] < colorArray[2]
        && colorArray[0] > 200
        && colorArray[1] < 200
        && colorArray[2] > 200) //(green)
    {
        //turn right
        motor1.run(-motorSpeed);
        motor2.run(-motorSpeed);
        delay (425);
    } else if (colorArray[0] > colorArray[2]
        && colorArray[0] > 200
        && colorArray[1] < 200
        && colorArray[2] > 200) //(orange)
    {
        //reverse slightly to not hit left wall
        motor2.run(motorSpeed);
        delay(400);
        //turn 180 to the left in the same grid
        motor1.run(motorSpeed);
        motor2.run(motorSpeed);
        delay (600);
    }
}

```

Figure 3: code for turning of Mbot (red, green, orange)

```

} else if (colorArray[0] > 280
  && colorArray[0] < 240
  && colorArray[1] < 125
  && colorArray[2] < 135) //(purple)
{
  //turn left
  motor1.run(motorSpeed);
  motor2.run(motorSpeed);
  delay (420);
  motor1.stop();
  motor2.stop();
  //move straight for one grid
  motor1.run(motorSpeed);
  motor2.run(-motorSpeed);
  delay (925);
  motor1.stop();
  motor2.stop();
  //turn left again
  motor1.run(motorSpeed);
  motor2.run(motorSpeed);
  delay (420);
}

```

Figure 4: Code for turning of Mbot (Purple)

```

} else if (colorArray[0] < 200
    || colorArray[1] < 200
    || colorArray[2] < 200 ) //(light blue)
{
    //turn right
    motor1.run(-motorSpeed);
    motor2.run(-motorSpeed);
    delay (400);
    motor1.stop();
    motor2.stop();
    //move straight for one grid
    motor1.run(motorSpeed);
    motor2.run(motorSpeed);
    delay (1000);
    motor1.stop();
    motor2.stop();
    //turn right again
    motor1.run(-motorSpeed);
    motor2.run(-motorSpeed);
    delay (400);
} else if (colorArray[0] > 255
    || colorArray[1] > 255
    || colorArray[2] > 255) //(white)
{
    //play victory music
    for (int i = 0; i < 3; i += 1)
    {
        note = random (100, 1500); // Freq range of numbers
        duration = random (50, 300); // Duration for each notes
        buzzer.tone (note, duration);
        delay(1000);
    }
}
}
}

```

Figure 5: Code for turning of Mbot (light blue) and end of maze (white)

Ultrasonic Sensor:

The ultrasonic sensor of our Mbot is placed on the left side of our Mbot. The ultrasonic sensor is also optimised such that if it detects an object (the walls of the maze) within 5.5cm, the Mbot would correct its course to prevent collision with the wall on the left side. To prevent collision with the wall on the left side, the Mbot would shift slightly to the right. We decided to set the distance as 5.5(cm) after much trial and error. The distance of 5.5(cm) allowed for just enough adjustment to the course of the Mbot while preventing over-adjustment.

```
//Ultrasonic distance sensor
Serial.print("distance (cm) = ");
int leftDist = ultrasonic.distanceCm ();
Serial.println(leftDist);

if (leftDist < 5.5) //if its too close to the left wall
{
    //correct to the right
    motor2.run(-motorSpeed);
    motor1.run(motorSpeed - 80);
}
```

Figure 6: Code for the ultrasonic sensor

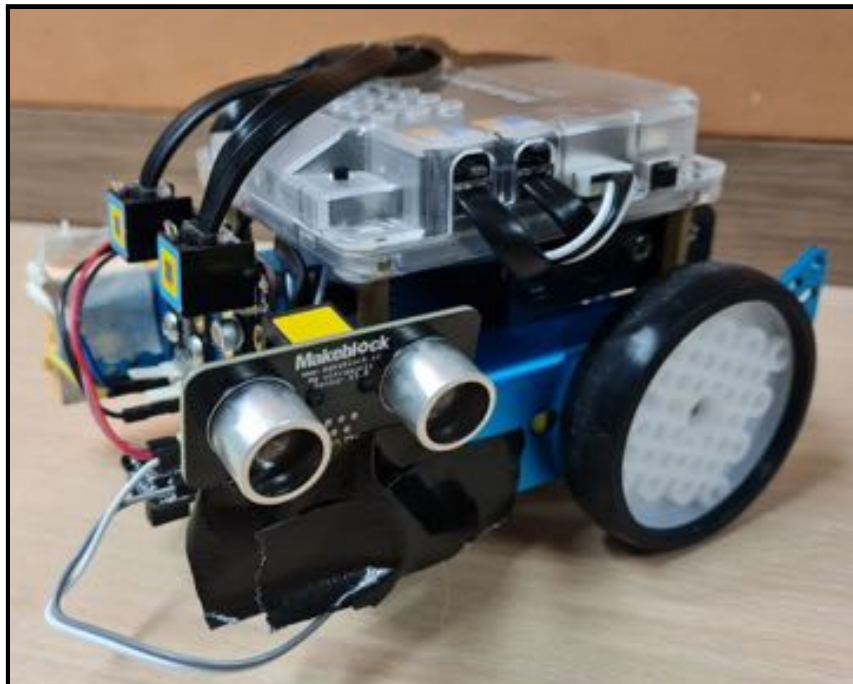


Figure 7: Placement of Ultrasonic Sensor

IR (Infra - Red) Sensor:

As for the IR sensor, we placed it on the right side of our Mbot. This is used to detect light intensity between the IR sensor and the right side of the wall and converts to the voltage measured. As seen from the code below, if the voltage measured is lower than 330 (3.3V), the Mbot would then correct its course to prevent collision with the left side of the wall by turning to the left slightly.

```
} else if (rightDist < 330) //if its too close to right wall
{
    //correct to the left
    motor1.run(motorSpeed);
    motor2.run(-motorSpeed + 130);
}
```

Figure 8: Code for IR sensor

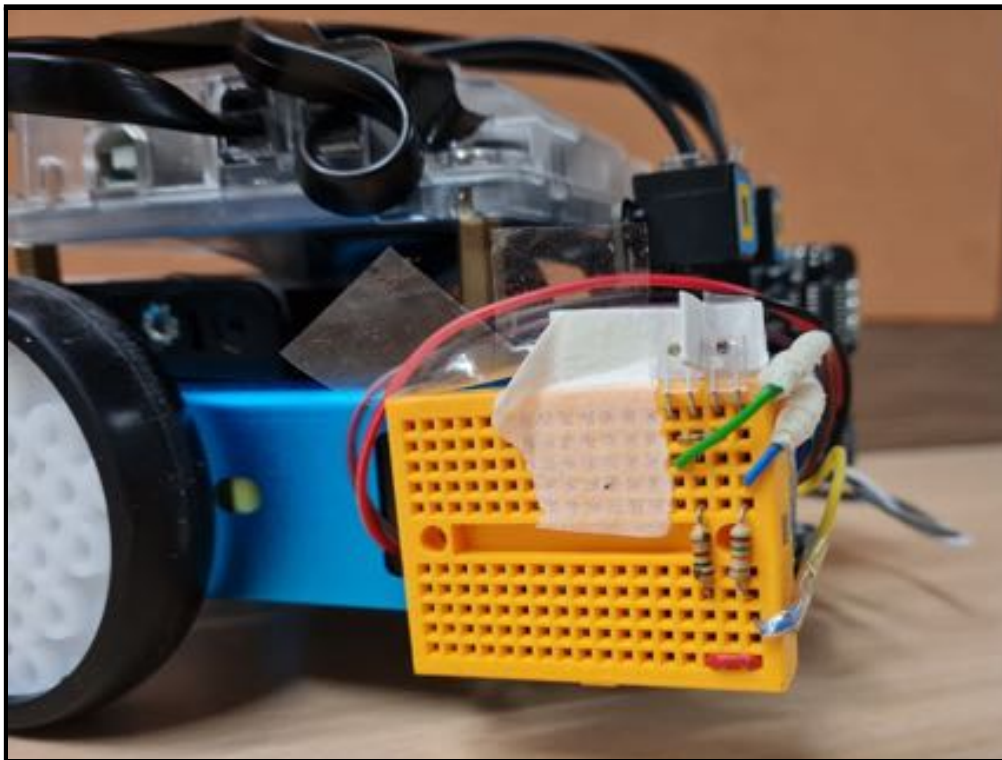


Figure 9: Placement of IR Sensor Circuit

1) Steps taken for calibrating and improving the robustness of our custom-built sensors

- We tried using different resistor values to see the maximum range at which our IR sensor can work. We then added paper in between the IR detector and IR emitter to prevent IR leaking directly from IR emitter to IR detector.
- We shortened all our connecting wires so that they would not interfere with the measurements taken, so that there would be no loose wires that might be caught on parts of the maze or the moving wheels.
- We added a ring of opaque black paper around the LDR to reduce the ambient light that could affect the LDR reading. However, we did not execute this correctly, as explained further in the next section below.

2) Significant difficulties faced

1. Fluctuating RGB values from the LDR

We had trouble doing the color calibration for the mbot, as we found it difficult to adjust the RGB values returned by the LDR as they kept fluctuating. Initially, we did not realise that we had to wrap (cover up the sides of the LDR) to prevent the LED from directly shining onto the LDR. Only after going through many rounds of testing did we realise that the LDR readings were being affected by the light from the LED being directly sensed by the LDR instead of getting the readings via the colored papers. The LDR is supposed to measure the RGB light bouncing off the colored paper. However, we were oblivious and went to wrap the LED instead of the LDR. It was only on the day of the final run, after we had done our first graded run, that we were told that the paper we used to wrap the LED to block or minimise any ambient light, should have been used to wrap the LDR instead.

- After correcting our mistake, we went straight to work on calibrating our RGB values that the LDR senses, as the previous values were no longer accurate with the changes we made .
- We found that the LDR now gave us much more consistent readings, as compared to when we wrapped the LED instead.

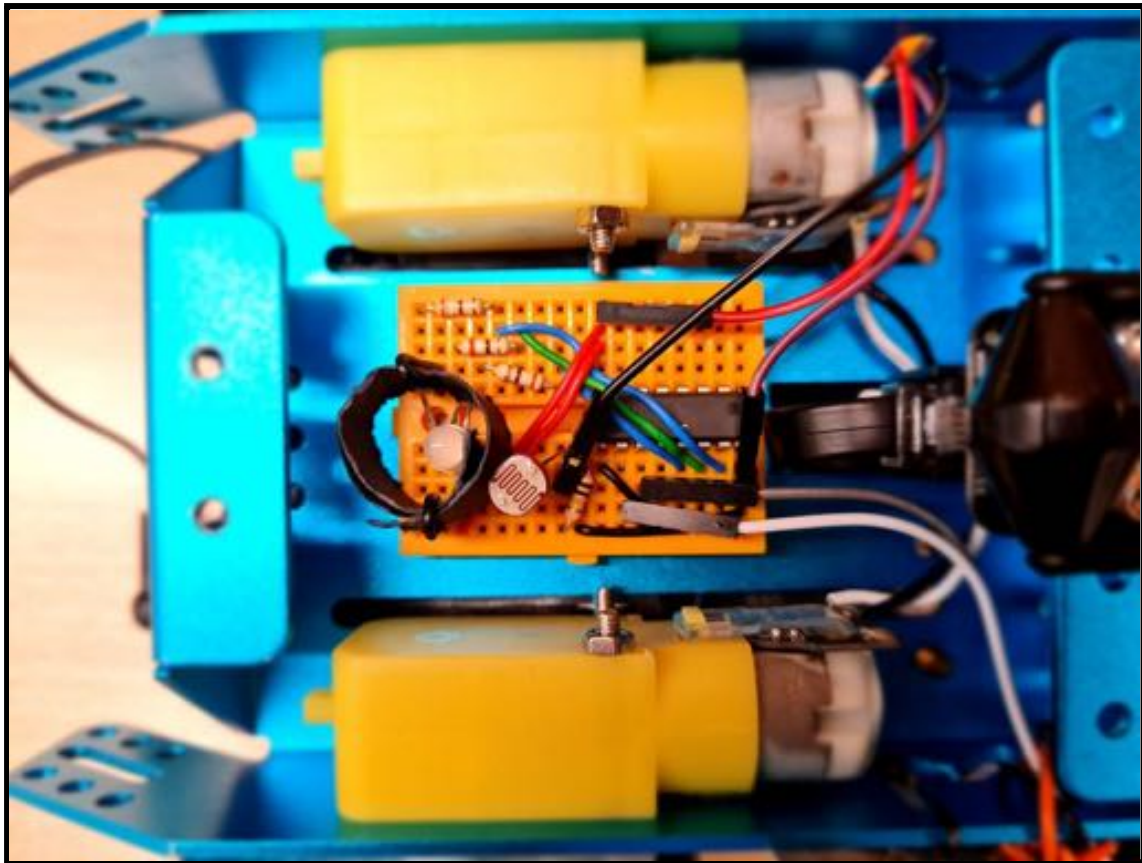


Figure 10: Before shifting the “chimney” from the LED to the LDR

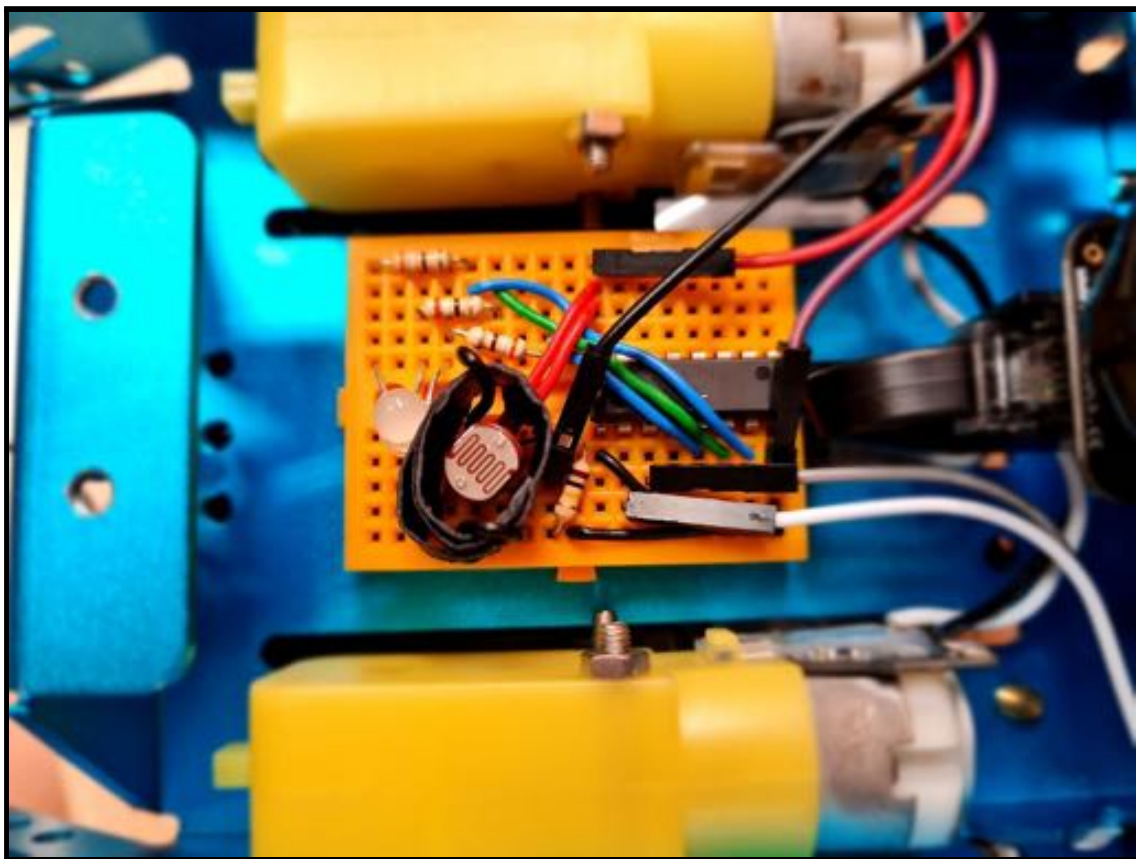


Figure 11: After feedback from the TAs and Professor

2. Potential shorting of the given sensors/adapters

Another difficulty faced was that because the ultrasonic sensor is attached to the side of the chassis instead of the usual front, we had to be careful that the ultrasonic sensor did not get shorted as the chassis is also made of metal.

- Thus, we layered some paper as insulating material in between the pins of the ultrasonic sensor and the body of the Mbot. (Figure 12)
- We repeated this for the 2 RJ25 Adapters that we mounted on the front of the Mbot. (Figure 13)

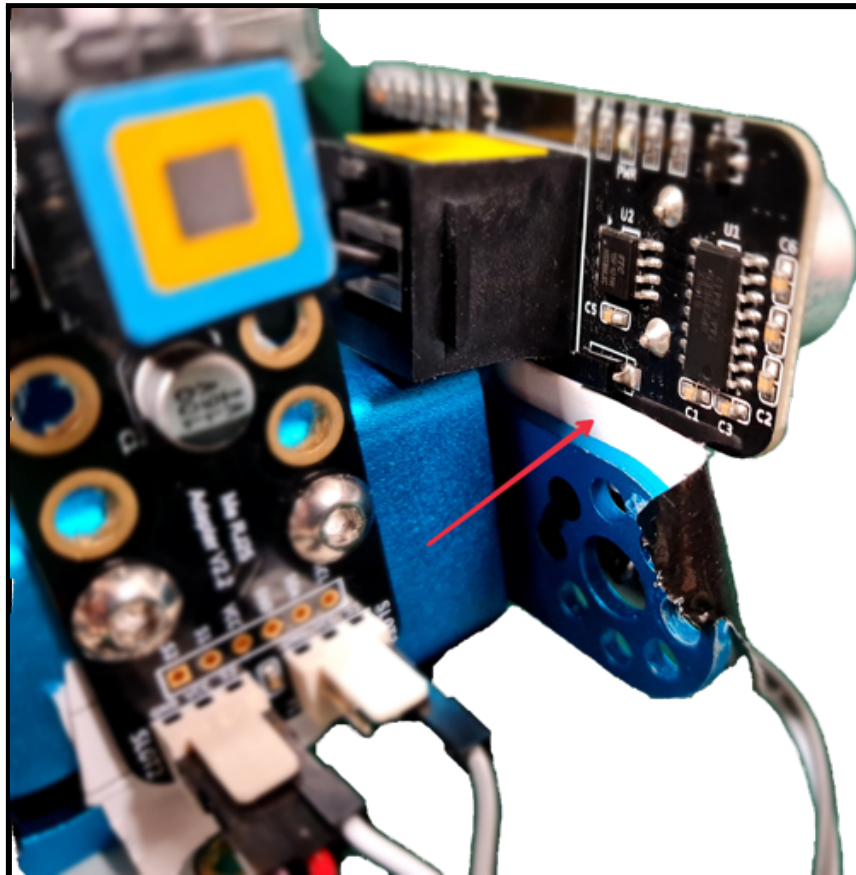


Figure 12: The piece of insulating material (paper) placed as shown

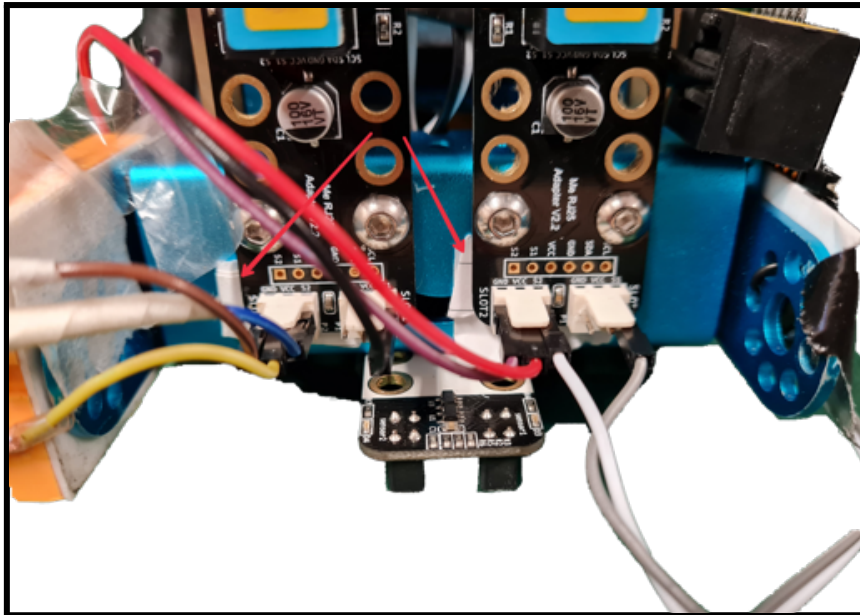


Figure 13: More insulating material used for the 2 RJ25 Adapters

3. Wires provided were too long

The normal breadboard wires provided, albeit convenient for setting up and testing the circuit, made our circuits messy as they were far too long and hindered the movement of the Mbot. Thus, we had to shorten the wires and resistors to make the circuit as compact as possible, as seen in Figure 14 and 15.

A model of the IR sensor circuit can be seen in Figure 16.

In addition, we also shortened the “legs” of the resistors so they could be slotted into the holes of the breadboard more compactly as seen below.

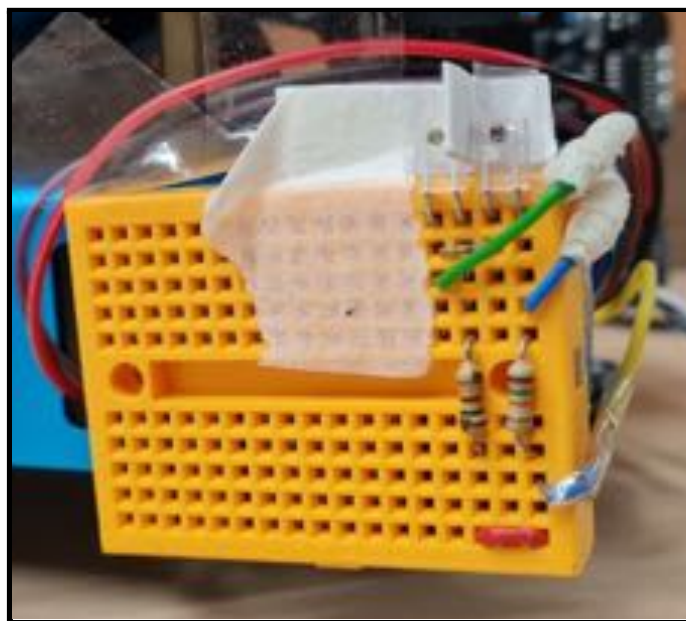


Figure 14: IR sensor breadboard

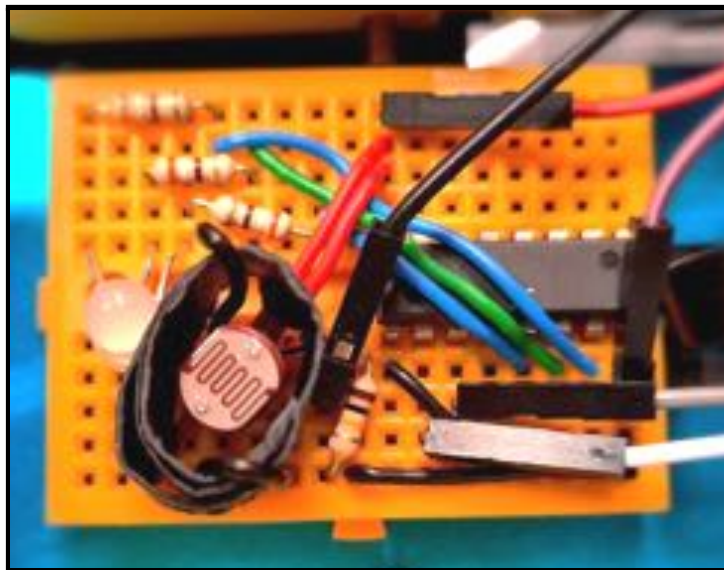


Figure 15: LDR breadboard

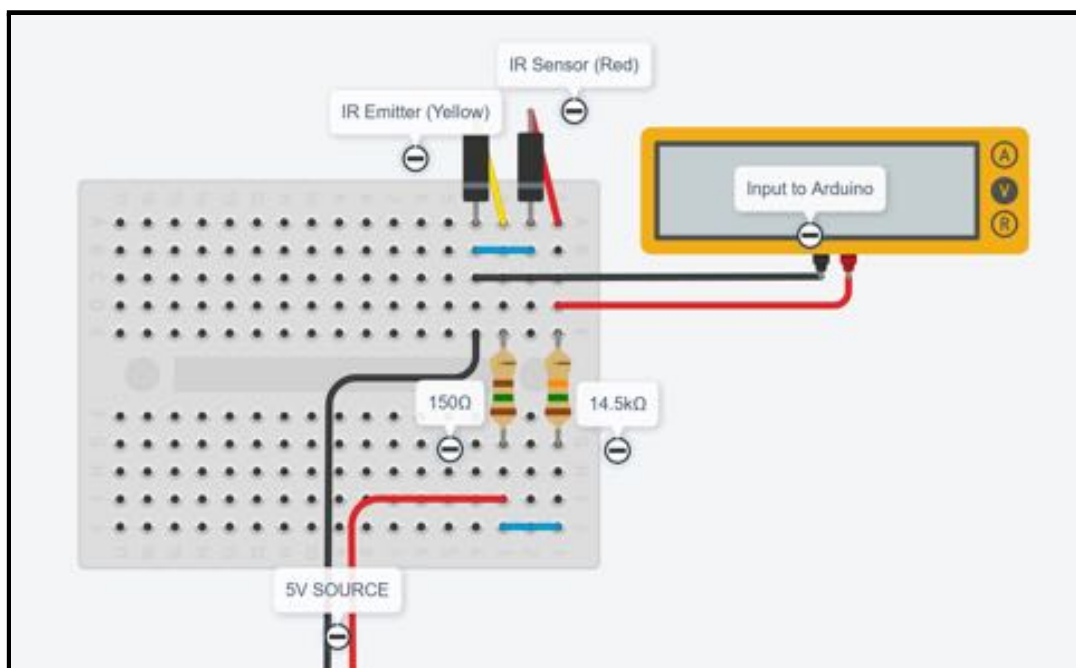


Figure 16: Model of the IR Sensor Circuit

4. Faulty Wires

We also found out that the wire used to connect the black line sensor to the Arduino was faulty and would only work when the wire was angled at certain positions. To deal with this issue, we decided to tape the wire in place using masking tape at first. We ultimately used clear tape to secure the wires on the underside of the Arduino, to ensure the wire was working as reliably as it can. (Figure 17)

We also used duct tape to secure the loose wires on the exterior of the Mbot, so that the wires do not come into contact with the wheels. (Figure 18)

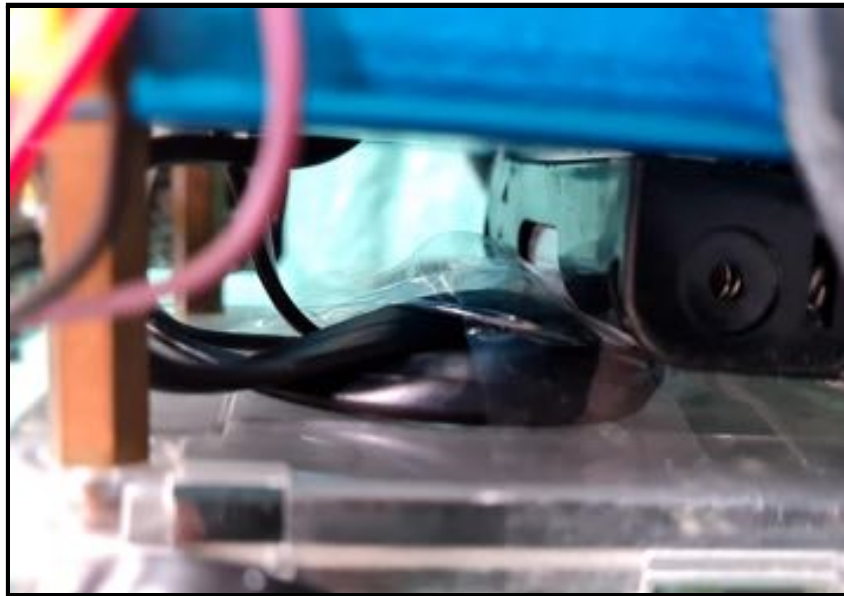


Figure 17: Tape used to hold wire in working position



Fig 18: Duct tape used to hold wires together

Work Delegation:

Due to the restrictions in place and the limited capacity the lab can hold, it was rather disruptive for the project, with only three of us being able to attend the lab sessions at any time.

As such, we decided to split the workload in a way that the one who is attending the online lab session would focus on writing up the final report while the rest of the members work on the Mbot.

The Coding for this project was mainly helmed by Ngiap Hin and Marcus, while the rest of us provide additional inputs on how to improve the code.

Circuit designs, calibration of the different colors, as well as the report was done as a group.

We repeatedly met up outside as a group to work on the Mbot, doing the calibrations together (IR sensor, Ultrasonic Sensor, RGB sensor).

Code: Marcus, Ngiap Hin

Assembling of Mbot: Marcus, Ansenn, Bryan, Ngiap Hin

LDR circuit: Ansenn

IR sensor circuit: Bryan

Report: Marcus, Ansenn, Bryan, Ngiap Hin

END