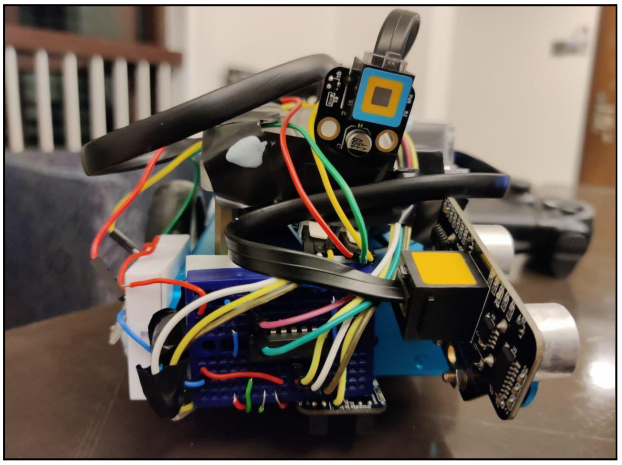
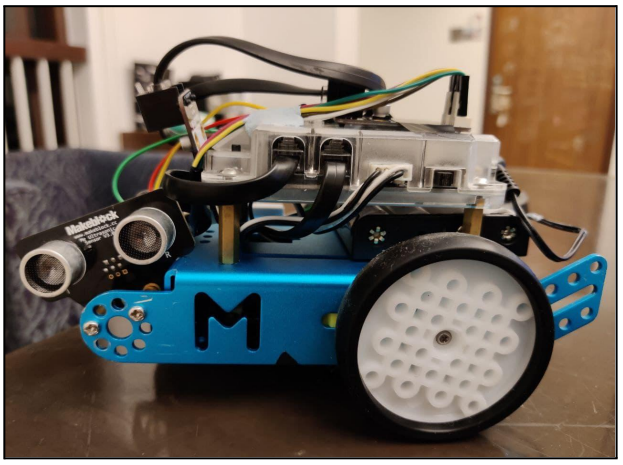
1. Pictures of mBot and sensor breadboard circuits

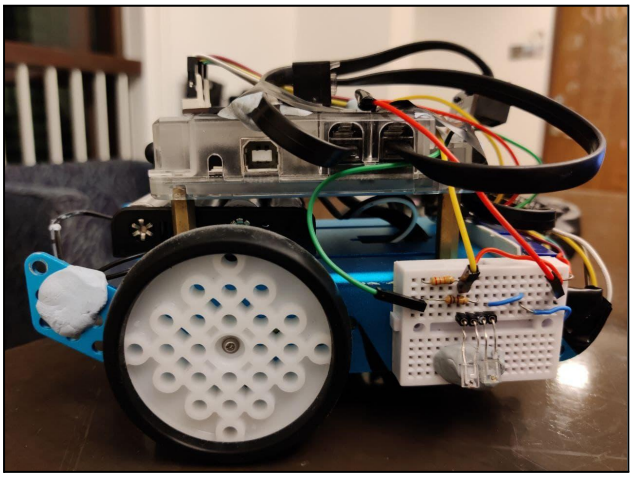
Front View



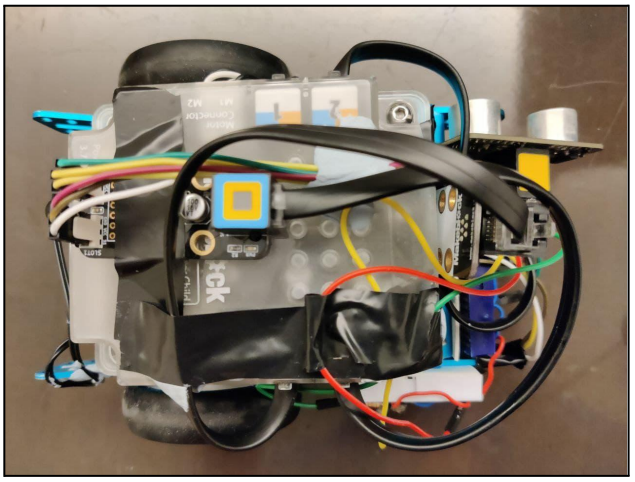
Left side view (Ultrasonic sensor)



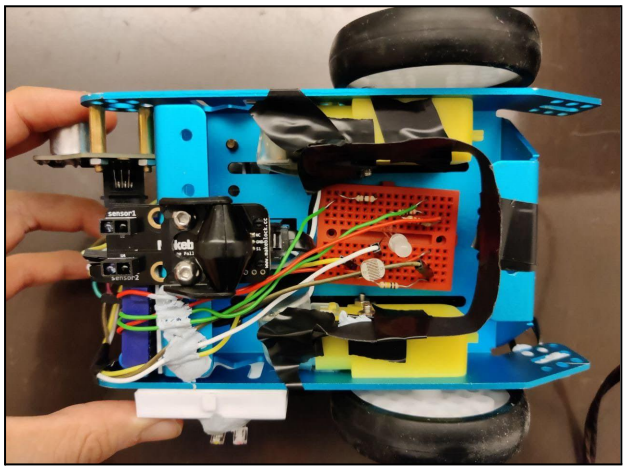
Right side view (IR sensor)



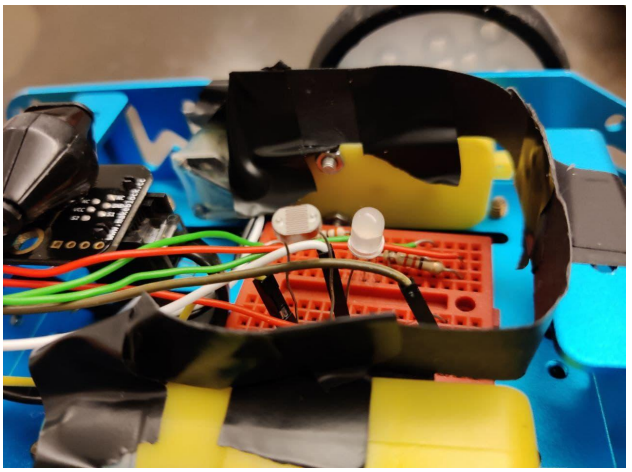
Bird's eye view



Bottom view



Bottom side View



2. Our overall algorithm

We use the ultrasonic and infrared sensors to align the mbot to travel straight in the maze, the line sensor to detect for a black line and the colour sensor to detect the colour of the paper below it.

2.1 Our approach (How does our mBot work?)

- When the MakeBlock Black Line sensor detects the black strip, the mBot will stop. The colour sensor is activated and detects the colour of the paper underneath the mBot.

- The robot will execute the turns or play the celebratory tune depending on the colour.

- If the robot does not detect the black line, it will keep moving forward. Our robot will adjust left if it is too close to the right wall or adjust right if it is too close to the left wall. The adjustment is based on the distances detected by the ultrasonic and infrared sensors.

The flow chart below shows the general approach we used:

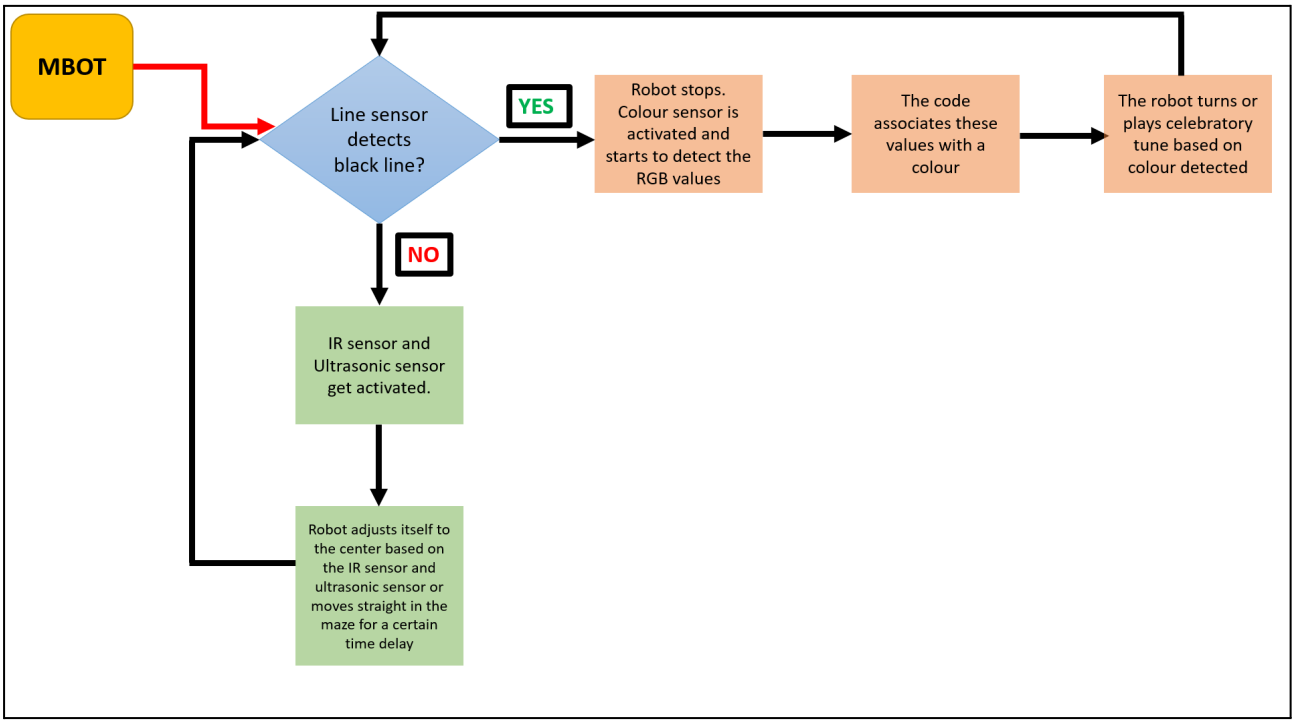


Figure 2.1.1 : Flowchart showing our approach

2.2 Execution of turns

- We decided to code the turning manually to ensure the robot does not hit the walls for all the turns.

- We adjusted the time delay and did many trial-and-errors to get our turns perfect.

- The image of the code below shows the values we used for our motor speeds during different parts of the maze as well as the time delay for each turn.

```
uint8_t motorSpeed = 200;
// motor speed when moving forward and turing left and right 90 degree
uint8_t slow_speed = 70;
// speed for the the side of the motor which the robot is turing while trying to maintain a straight path
uint8_t fast_speed = 150;
//speed for the the side of the motor which the robot is too close to the wall
int right_forward_delay = 770;
// time required for robot to move forward to reach next grid in between the successive right turns
int left_forward_delay = 790;
// time required for robot to move forward to reach next grid in between the successive left turns
int left_turn_delay = 328;
// time required for left turn to complete
int right_turn_delay = 335;
// time required for left turn to complete
int turn_180_delay = right_turn_delay * 2;
// time required for 180 degree turn to complete
```

Figure 2.3.1: Code Excerpt showing motor speeds and delays used for turns

3. Exploring each subsystem and implementation in greater detail

3.1 MakeBlock Black Line Sensor

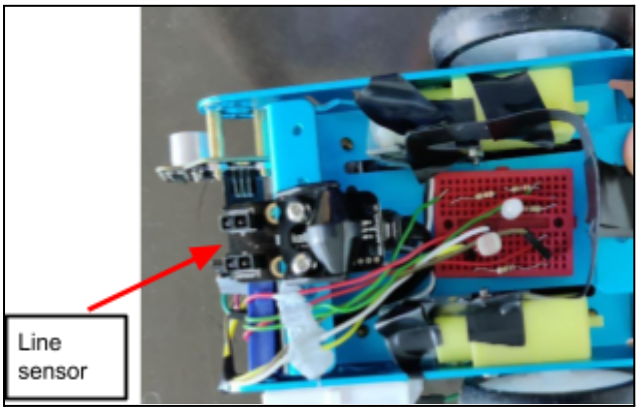


Figure 3.1.1: Location of our line sensor

- The MakeBlock Black Line Sensor is mounted underneath the mBot connected to Digital Port 1.
- With the use of "MeMCore.h" library and declaration of MeLineFollower lineFinder(PORT_1) at the start of the programme, checking the output of the two line sensors can be achieved by calling lineFinder.readSensors().

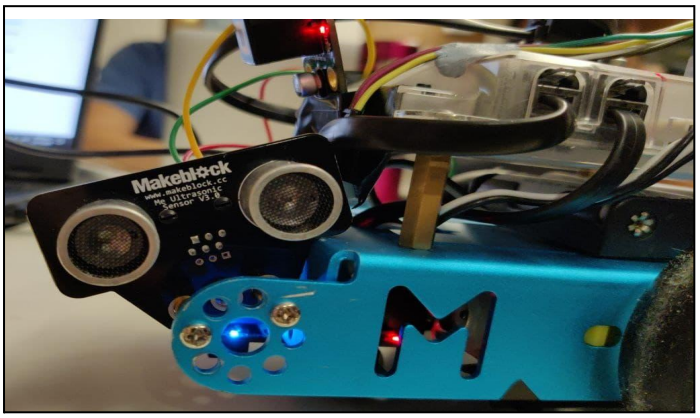
```
lineSensor = lineFinder.readSensors();
record_baseline_voltage(ir_count);

if (lineSensor == S1_IN_S2_IN)
{
  motor_stop();
  read_ldr();
  execute_turn();
}
else
{
  turn_off_led(); //turn on ir emitter.
  ir_val = analogRead(ir_input) ; // ir_input is A0;
  dist = ultraSensor.distanceCm();
```

- If the output from lineFinder.readSensors() == S1_IN_S2_IN, both sensors detect the black strip, the robot stops and detects the colour of the colour paper underneath the mBot. Otherwise, the robot will continue to move straight or adjust itself in the maze.

Figure 3.1.2 : Code Excerpt of actions mBot does upon detecting black line

3.2 Ultrasonic sensor



- The ultrasonic sensor is mounted on the left inner side of the mBot and connected to Digital port 2

- Distance is obtained by calling ultraSensor.distanceCm() from "MeMCore.h" library after declaring MeUltrasonicSensor ultraSensor(PORT_2) at the start of the program.

Figure 3.2.1: Image of Ultrasonic sensor

3.2.1 Calibration

We placed the mbot in the maze in between the 2 walls and measured the actual distance from the robot to the wall and compared it to the value of the distance read by the ultrasonic sensor. This was to check if the ultrasonic sensor was working correctly, and to find out if we need to make any adjustments to the ultrasonic sensor code in the event where the ultrasonic sensor has systematic error.

We also tried different positions and turn angles of the mbot with respect to the walls to troubleshoot and find the threshold distance which it should adjust itself. This way we figured out the optimal distance from the wall at all angles to ensure the robot does not turn too early or too late and hit the wall.

3.3 2-to-4 decoder IC chip

- The decoder chip was used to control the switching on and off of the infrared sensor as well as the 3 LEDs.

- We used the right side of the decoder (the side with 2). Our 2G Enable pin was grounded to ensure that the right side of the decoder was always activated.

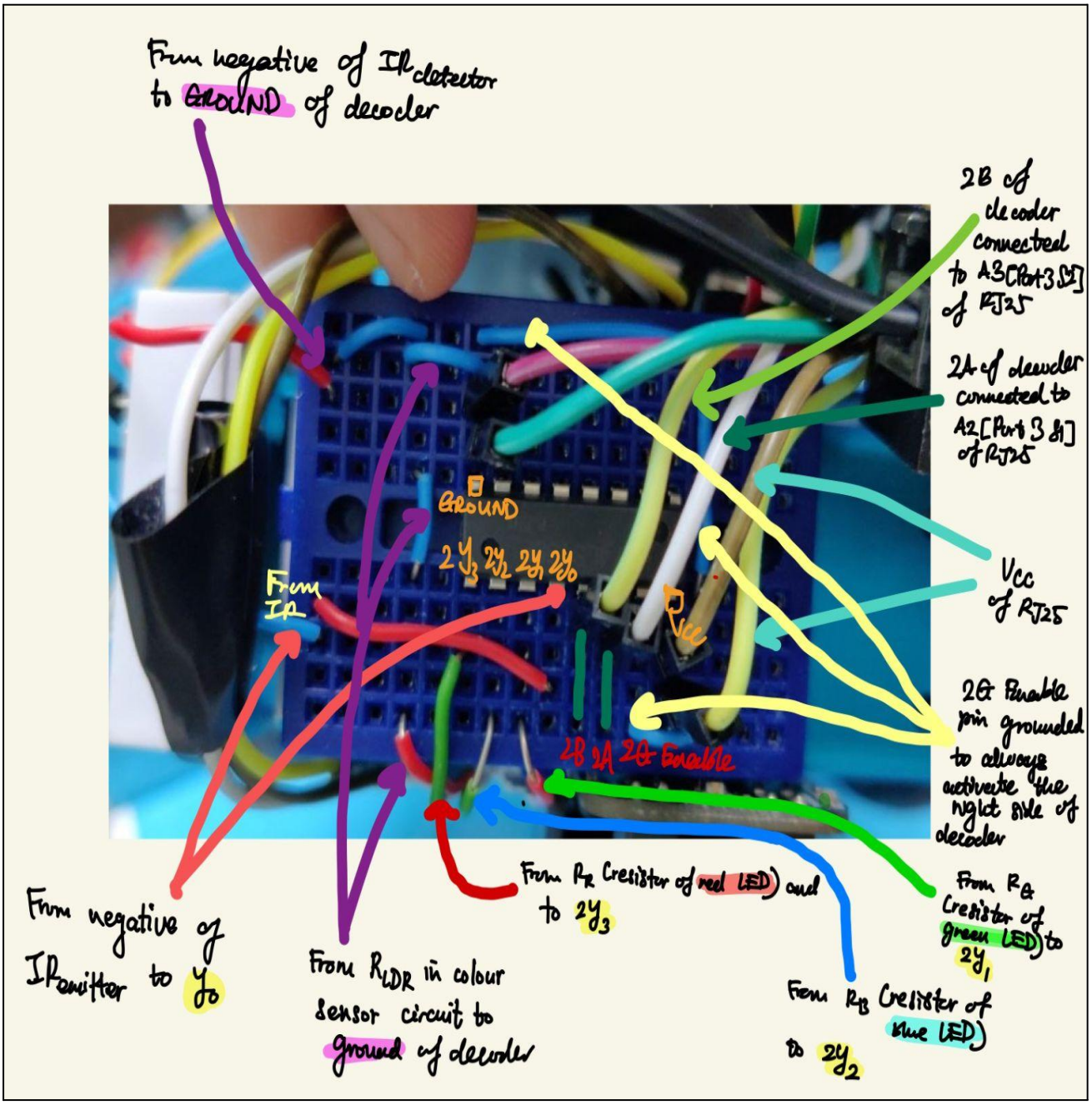


Figure 3.3.1: Annotated diagram showing the circuit of our decoder chip

3.4 Infrared sensor (IR Emitter and Detector)

- The infrared sensor is mounted on the right of the mBot.

- Records the analogue voltage of the infrared sensor when the white tile is placed at a specific distance for a range of distances.

- The data is then used to create an equation converting the values to distance in cm to allow the mBot to determine the distance between the IR sensors and the wall in order to adjust itself and maintain a straight path.

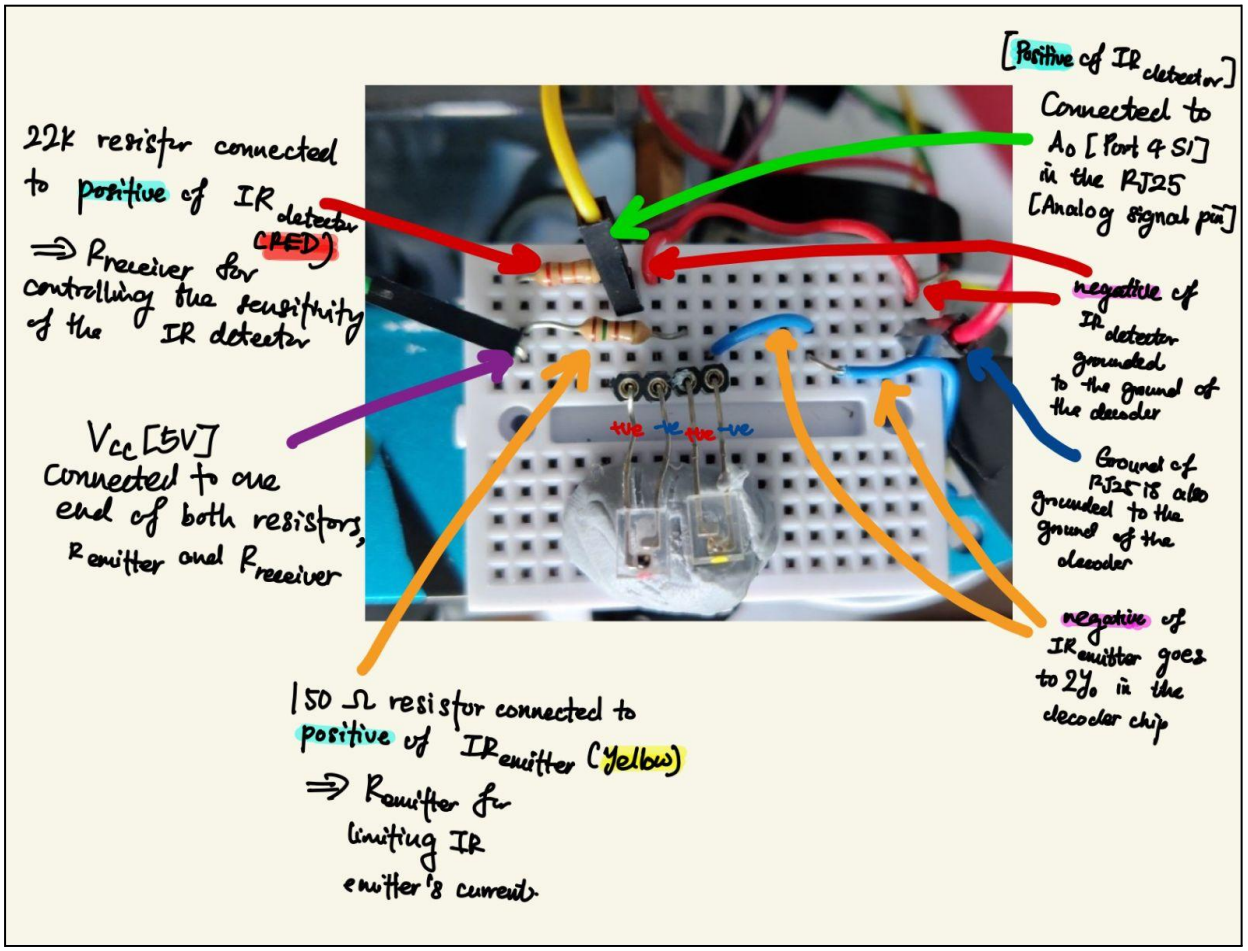


Figure 3.4.1: Annotated diagram showing the circuit of our Infrared Sensor

3.5 Using the ultrasonic sensor and infrared sensor to move straight

- The optimal distance between the wall of the maze and the ultrasonic sensor is set to **7 cm** through experiment.

- IR sensor is used when there is no wall detected by the ultrasonic sensor (distance detected by ultrasonic sensor > **25cm**) while moving forward to increase the reliability of the robot.

- If ultrasonic sensors detect a distance of less than **7 cm** or more than **12cm**, the mbot will adjust itself to the centre of the pathway of the maze.

- Through experiment, we concluded that **6.5cm and 10.5** is the optimal distance away from the wall detected by the IR sensor. Also, the maximum distance that can be detected by our IR sensors is around 12cm. Any further than that, the reading will remain the same. Hence, a condition greater than 10.5 and less than 11.5 cm is used to prevent mBot turns to the right when both sides have no wall, wrongly detected its position is to the left side of the lane.

- If IR sensors detect a distance of less than **6.5 cm** or more than **10.5 and less than 11.5 cm**, the mbot will adjust itself to the centre of the pathway of the maze.

- If the distance detected by the ultrasonic sensor is more than **25cm** and **11.5cm(**Max voltage change) by the IR sensors (no wall on both sides), the robot will move straight.

```
if (dist > 25) { // ultrasonic detects empty wall, use ir to maintain straight path
  if (ir_dist < 6.5) { //ir detects mbot close to right side of the wall
    turn_left_slowly(); // move left
  }
  else if (ir_dist > 10.5 && ir_dist < 11.5) { //ir detects mbot close to left side
    turn_right_slowly(); //move left
  }
  else { // both side detects no wall or mBot moves straight, continue moving forwa
    move_forward();
  }
}
else if (dist < 7) {  // ultrasonic detects mbot close to left side of the wall
  turn_right_slowly();// move right
}
else if (dist > 12) { //ultrasonic detects mbot close to right side of the wall
  turn_left_slowly(); // move left
}
else { // mBot moves straight, continue moving forward
  move_forward();
}
```

Figure 3.5.1: Code excerpt of when our mBot adjusts itself

3.6 Colour Sensor:

- The colour sensor is attached to the bottom of the mBot

- At each waypoint challenge, the colour sensor will detect the colour of the paper placed underneath the mBot.

- To detect colour, voltage reading is obtained each time from shining the red, green, and blue led.

```
void read_ldr()
{
  //turn on one colour at a time and LDR reads 5 times
  for (int c = 0; c <= 2; c++)
  {
    turn_on_led(c);  //turn ON the LED, red, green or blue, one colour at a time.
    delay(RGBWait);

    //get the average of 5 consecutive readings for the current colour and return an average
    colourArray[c] = getAvgReading(5);

    //the average reading returned minus the lowest value divided by the maximum possible range,
    //multiplied by 255 will give a value between 0-255, representing the value for the
    //current reflectivity (i.e. the colour LDR is exposed to)
    colourArray[c] = (colourArray[c] - blackArray[c]) / (greyDiff[c]) * 255;
    turn_off_led(); //turn off the current LED colour
    delay(RGBWait); //time delay before the next RGB colour turns ON to allow LDR to stabilize
  }
}
```

Figure 3.6.1: Code excerpt of how the colour sensor detects RGB values

- The mBot then performs different actions based on the colour of the paper detected.

3.6.1 Our colour sensor connections

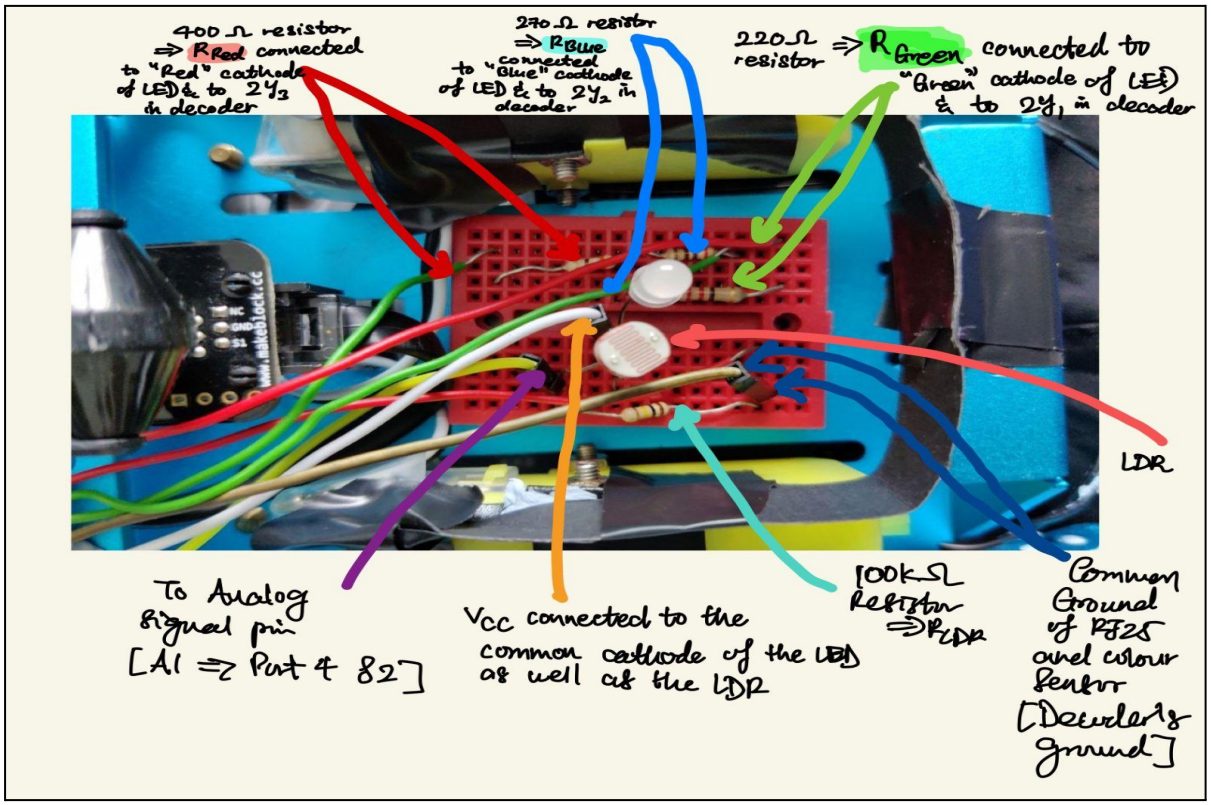


Figure 3.6.1 Annotated diagram showing the circuit of the colour sensor

The resistance of the resistors connected to the LED in the above circuit were approximated and chosen based on the calculated values of the resistances needed to maintain a current of below 0.8mA through the decoder.

3.6.2 Workings of the colour sensor

The code below is used to distinguish the colours. Using if else statements, we separated the RGB values to split the number of colours we are searching and eliminate each colour one by one. The flowchart below shows how the code works.

```
void execute_turn() // detects colour and turns acoordingly
{
  if (colourArray[0] > 220) // red, orange and purple has red vlaue higher than 220.
  {
    if (colourArray[1] > 230 && colourArray[2] > 230) // white has green and blue value higher than 230.
    {
      // robot detects white: stops moving and plays a celebratory tone
      motor_stop();
      play_tune();
    }
    else if (colourArray[1] > 90) //green value of orange is much higher than 90
    {
      turn_180(); //detects orange, turn 180 degree
    }
    else //green value of red is much less than 90
    {
      turn_left(0); // detects red: turn left
    }
  }
  else if (colourArray[1] > 195) //separate blue and green from purple
  {
    if (colourArray[2] > colourArray[1]) // blue value higher than green value for blue
    {
      successive_right_turns(); //detects blue: two successive right turns
    }
    else //detects green
    {
      turn_right(0);
    }
  }
  else if (colourArray[0] < 205 && colourArray[1] < 205) // both green and blue value of purple are much lower than 205
  {
    successive_left_turns(); //detects purple: two successive left turns
  }
  else
  {
    read_ldr(); //colour is not confirm, recheck the rgb reading to determine the colour again
  }
}
```

Figure 3.6.1: Code excerpt on how our colour sensor differentiates the colours

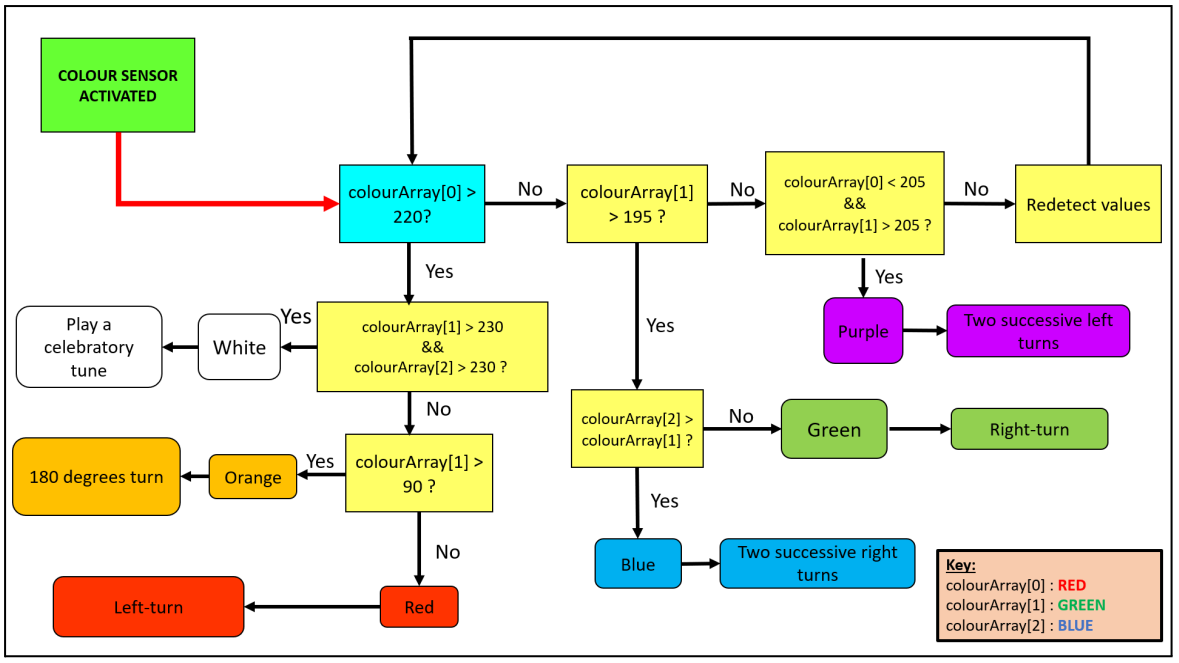


Figure 3.6.2: Flowchart showing how the colours are differentiated

4. Steps taken for calibrating and improving the robustness of custom-built sensors

4.1 Infrared sensor:

- IR sensors are put in close proximity to make the readings more accurate

- To reduce the effect of varying ambient IR, baseline voltage is being read every 10 loops by turning off the IR emitter and used to determine if the mBot is moving straight.

```
void record_baseline_voltage(int ir_count) // record base line voltage every 10 loops
{
  if (ir_count == 0)
  {
    turn_on_led(1);// turn off ir emitter
    base_ir = analogRead(ir_input); // record baseline voltage
  }
  else if (ir_count == 9) // if is 9, restart the cycle.
  {
    ir_count = 0;
  }
  else //else add 1 each time
  {
    ir_count += 1;
  }
}
```

Figure 3.6.3: Code showing how base_ir is recorded every 10 loop

- Baseline voltage and average voltage readings with respect to distance were taken in the actual maze to increase the reliability of the data.

- A graph of distance in cm against the difference between the baseline voltage and the average voltage readings was plotted to determine the formula used to calculate the distance detected by the mBot.

Distance	Baseline voltage	1st IR reading	2nd IR reading	3rd IR reading	Average	Difference in voltage
1	618	38	40	43	40	578
2	618	45	50	53	49	569
3	618	60	67	58	62	556
4	618	158	160	155	158	460
5	618	222	230	228	227	391
6	618	254	240	248	247	371
7	618	283	284	275	281	337
8	618	298	303	307	303	315
9	618	323	320	328	324	294
10	618	348	355	357	353	265
11	618	365	365	366	365	253

Figure 3.6.4: Table with data for IR sensors

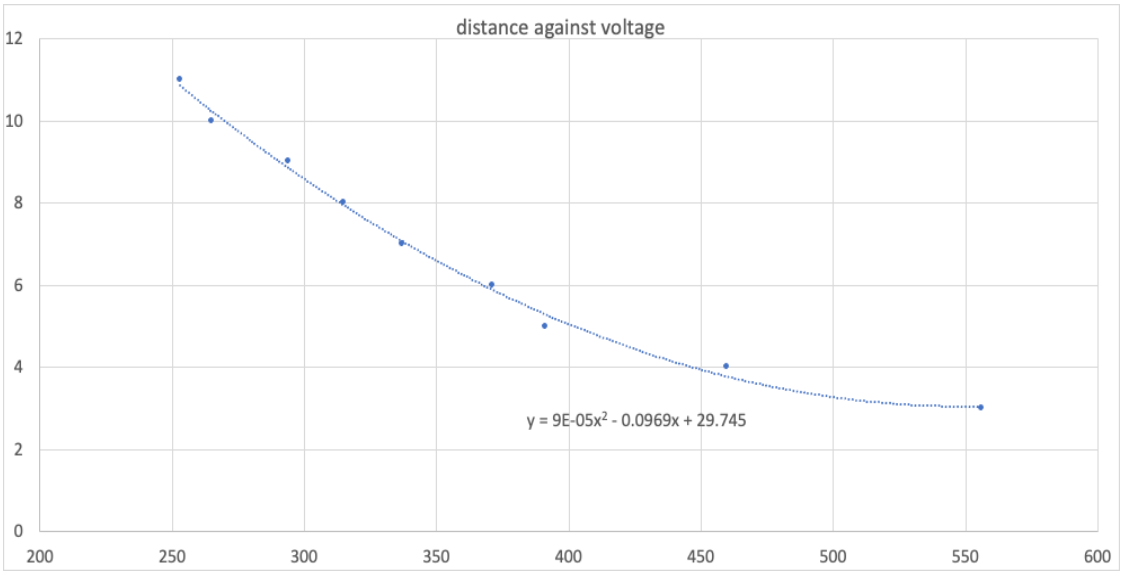


Figure 3.6.5: Graph of distance against voltage in range of 1023

4.2 Colour sensor:

- To ensure consistency in comparison of rgb value for each colour, the reading for each LED colour is then converted to a range of 255 by subtracting the least possible value (obtained from shining RGB led at black paper), dividing by the difference the least possible value and the largest possible value (obtained from shining RGB led at white paper) and multiplying by 255. The value for the least, greatest possible value is taken from the actual maze venue to increase reliability of the data.

```
void setBalance(){
//set white balance
  Serial.println("Put White Sample For Calibration ...");
  delay(5000);           //delay for five seconds for getting sample ready
  digitalWrite(LED,LOW); //Check Indicator OFF during Calibration
//scan the white sample.
//go through one colour at a time, set the maximum reading for each colour -- red, green and blue to the white array
  for(int i = 0;i<=2;i++){
     turn_on_led(i);
     delay(RGBWait);
     whiteArray[i] = getAvgReading(5);         //scan 5 times and return the average,
     turn_off_led();
     delay(RGBWait);
  }
//done scanning white, time for the black sample.
//set black balance
  Serial.println("Put Black Sample For Calibration ...");
  delay(5000);     //delay for five seconds for getting sample ready
//go through one colour at a time, set the minimum reading for red, green and blue to the black array
  for(int i = 0;i<=2;i++){
     turn_on_led(i);
     delay(RGBWait);
     blackArray[i] = getAvgReading(5);
     turn_off_led();
     delay(RGBWait);
//the differnce between the maximum and the minimum gives the range
     greyDiff[i] = whiteArray[i] - blackArray[i];
  }
//delay another 5 seconds for getting ready colour objects
  Serial.println("Colour Sensor Is Ready.");
  print_calibration(); // print to the serial monitor the set of white, black and grey for each led
  delay(5000);
}

void print_calibration()
{
  for (int i = 0; i < 3; i += 1)
  {
     Serial.print("whiteArray: ");
     Serial.print(whiteArray[i]);
     Serial.print(" blackArray: ");
     Serial.print(blackArray[i]);
     Serial.print(" greyDiff: ");
     Serial.println(greyDiff[i]);
  }
}
```

Figure 4.2.1: Code excerpt to obtain greatest and least possible readings from white and black paper

- Average of 5 readings of RGB values was used to increase the accuracy of the readings obtained.
- Black paper was wrapped around the colour sensor to reduce the effect of ambient light affecting the RGB values obtained.
- We also cut the resistor's legs so that they are inserted securely to the breadboard and do not accidentally touch any other metal parts of the circuit and affect the readings.
- LDR is placed higher than the LED to minimize the effect of LED shining on the LDR

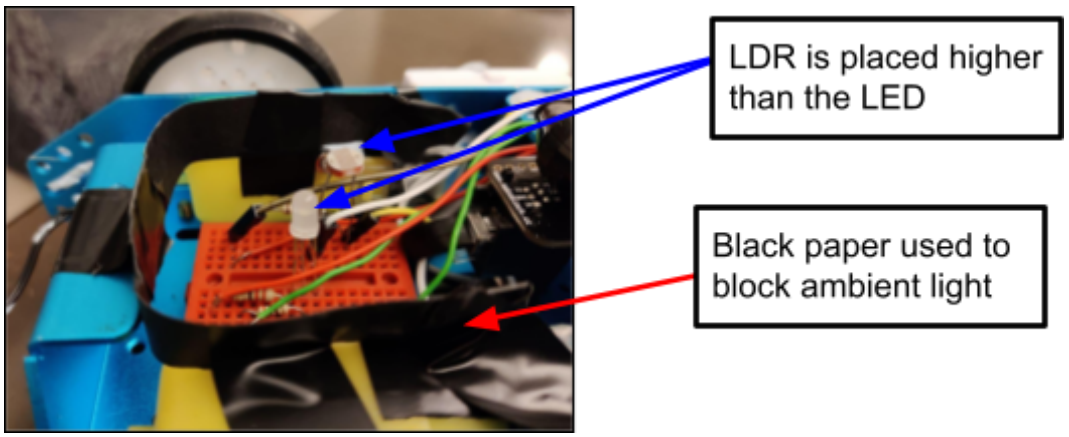


Figure 4.2.2: Image of colour sensor side view

5. Roles and work division

The whole project was done as a team effort. At the beginning of the project, we came together to discuss the overall algorithm as well as the pseudocode on how we could complete the maze. Due to the restriction of having only 3 students per team during the studio session, the 4th person would work on the report as well as the code.

While we helped each other and worked on the mBot together throughout the duration of the project, some of us focused on certain subsystems and tasks.

Parts of mBot/Task	Team Members mainly in-charge
Ultrasonic sensor	Wilson and Yunfan
Motor (Turning and adjusting when too close to the walls)	Wilson and Yunfan
Infrared Sensor	Chaitanya and Jing Wei
Colour sensor	Chaitanya and Jing Wei
Main part of Arduino code	Yunfan
Report	Chaitanya, Jing Wei, Wilson and Yunfan

Overall, this project was a combined team effort and all of us actively participated in group meetings. Everyone put in much hard work and time to make this robot clear the maze.

6. Challenges faced and how we overcome them

6.1 Wiring, connections and equipment

There were a lot of wires in the custom-built circuit
Not only was it difficult to locate the connections, it also caused some of the wires to have loose connections with the breadboard.

To solve this issue, we use the precise-length wires to ensure that the wiring and circuitry is neater and sturdy.

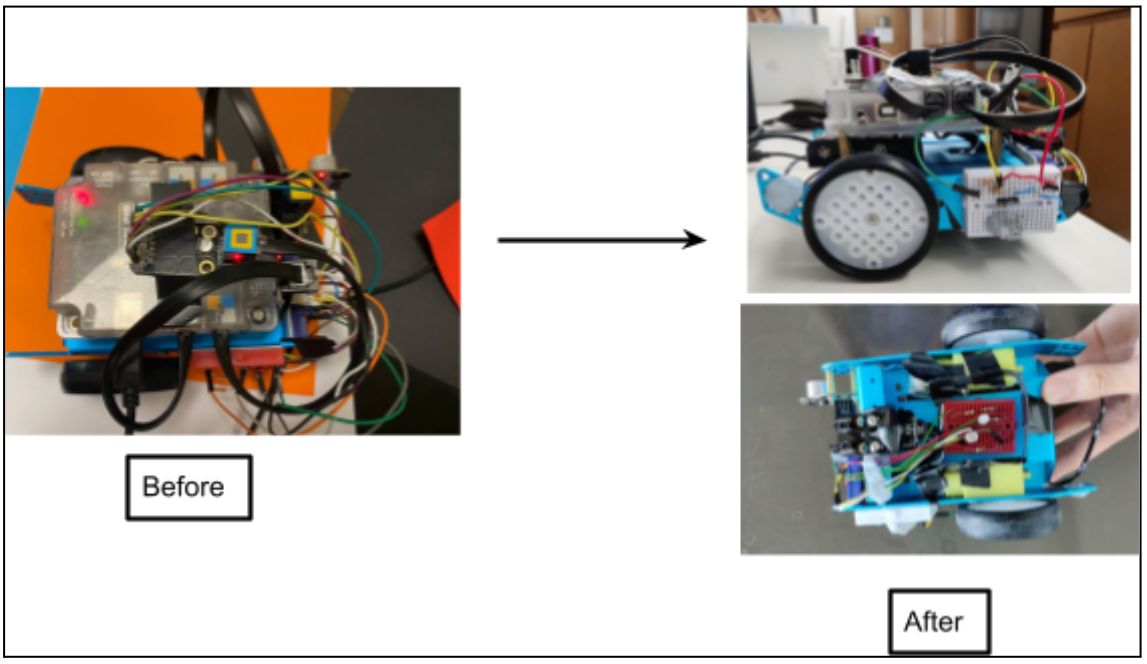


Figure 6.1.1: Images comparing before and after wirings of our circuits

Equipment failures
Our circuits were not working as intended after multiple testings. With the utilisation of the multimeter and lab oscilloscope, we realised that the issues were with the wires and there was no voltage transfer across the wires. We had to change the wires multiple times to ensure that the circuit worked.

We also faced the same issue with the infrared detector. The infrared detector stopped working suddenly during the recording of values and we were unsure why. We replaced our IR sensors multiple times to ensure the sensor could read the values consistently. Many hours and sleepless nights were spent on troubleshooting and narrowing down the possible issues for the faulty circuit.

6.2 Line detector unable to detect black line in time

When we first tested out the mbot, it was unable to detect the black line and stop in time. It would often hit the wall ahead. To solve this, we removed the unnecessary delay in the loop. This increases the frequency of mbot checking for black lines.

6.3 Infrared detector connection and securing it in place

There was a loose connection of the IR detector to the breadboard. Upon bending the IR detector to be parallel to the wall, we realised that the IR detector is no longer connected to the circuit. Therefore, we used pin holders for the IR detectors and emitters to ensure that they remain connected to the circuit.

Furthermore, we realised that the IR sensor was extremely sensitive. Hence, we used blu tack to secure it in place to ensure accurate readings.

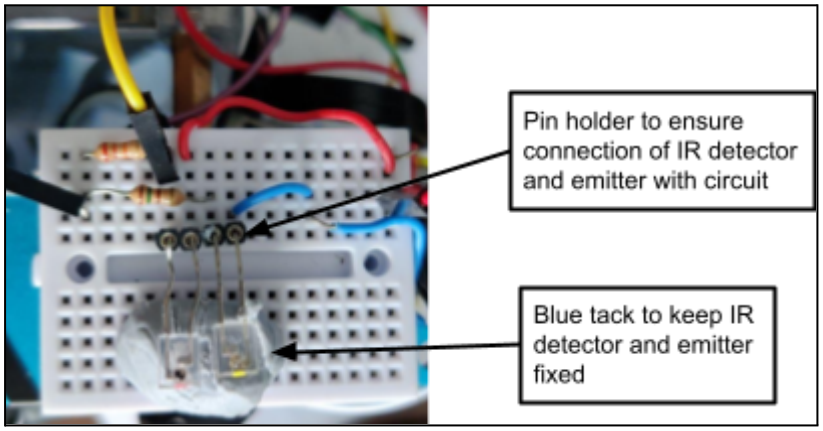


Figure 6.3.1: Image of infrared sensor circuit

6.4 Position of ultrasonic sensor

Originally, our ultrasonic sensor was placed on the outside of the mbot. This increased the likelihood of the distance between itself and the wall being below 3cm (below the range of the ultrasonic sensor). It also caused problems during our turns (especially during the 180 degrees turn) as the sensor would hit the wall and disrupt the turning.

Initially:

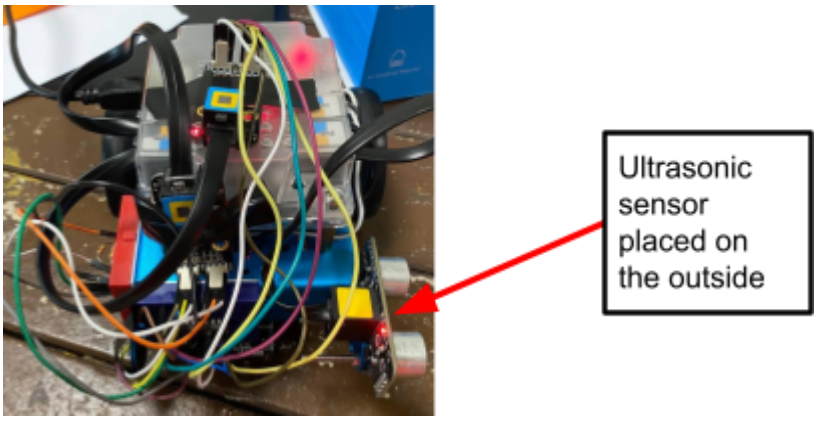
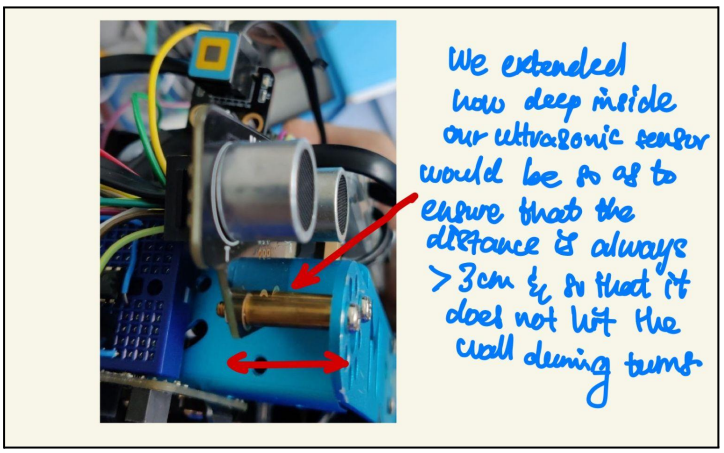


Figure 6.4.1: Image of our initial placement of Ultrasonic sensor



- We shifted the ultrasonic sensor to the inside of the mbot.

- This improved our robot's adjustments and turns.

Figure 6.4.2: Zoomed-in image of our final placement of Ultrasonic sensor

6.5 Colour sensor detecting correctly

We had to overcome various challenges to get our colour sensor to work.
We realised that ambient light made a difference on the calibration as well as the RGB values of each colour. As mentioned above and shown in Figure 4.2.2, we adopted the use of black paper to block off ambient light to the LDR.

Apart from that, we calibrated the white and black papers at the testing maze itself so that the arrays ("white", "black" and "greydiff") would be as accurate as possible.

We also had to adjust the threshold RGB values for each colour in order to safely eliminate a colour one by one as shown in the flowchart in Figure 3.6.2.

These changes allowed us to accurately detect the different colours.

7. Closing remarks

Overall, the A-maze-ing Race Project has been a great experience for the entire team. After 4 weeks of working together and building the robot, we got to know each other better after encountering many challenges as a team. We would like to thank Prof Ravi for his patience and guidance, helping us when we could not figure out why our IR sensor was not working, and also giving us encouragement during the test run. Thank you to the teaching assistants, especially Yuan Bing, for their patience in helping us throughout the entire semester. If not for all of you, our CG1111A journey would not have been as fun and as fruitful! Once again, thank you for all the effort you have put in and our team would like to wish everyone all the best in the future!

8. References

Makeblock-official/Makeblock-Libraries: Arduino Library for Makeblock Electronic Modules, learn more from Makeblock official website (github.com)

Thank you Prof Ravi for his mbot lecture notes

Thank you Dr Henry Tan for his colour sensor code:
 * ColourSensor.txt
 * Designed by Dr Henry Tan
 * For CG1111A Photoelectric Sensor Studio