CG1111A

Final Report

AY 2022/2023 S1

Studio group B04, Section 1, Team 1

Dylan Chia Tian (A0253141U)

Dao Trong Khanh (A0255826W)

Goh Jia Ying Cayenne (A0259078N)

Ella Yovita Suwibowo (A0266763R)

Table of Contents

1. Introduction

1.1 The project

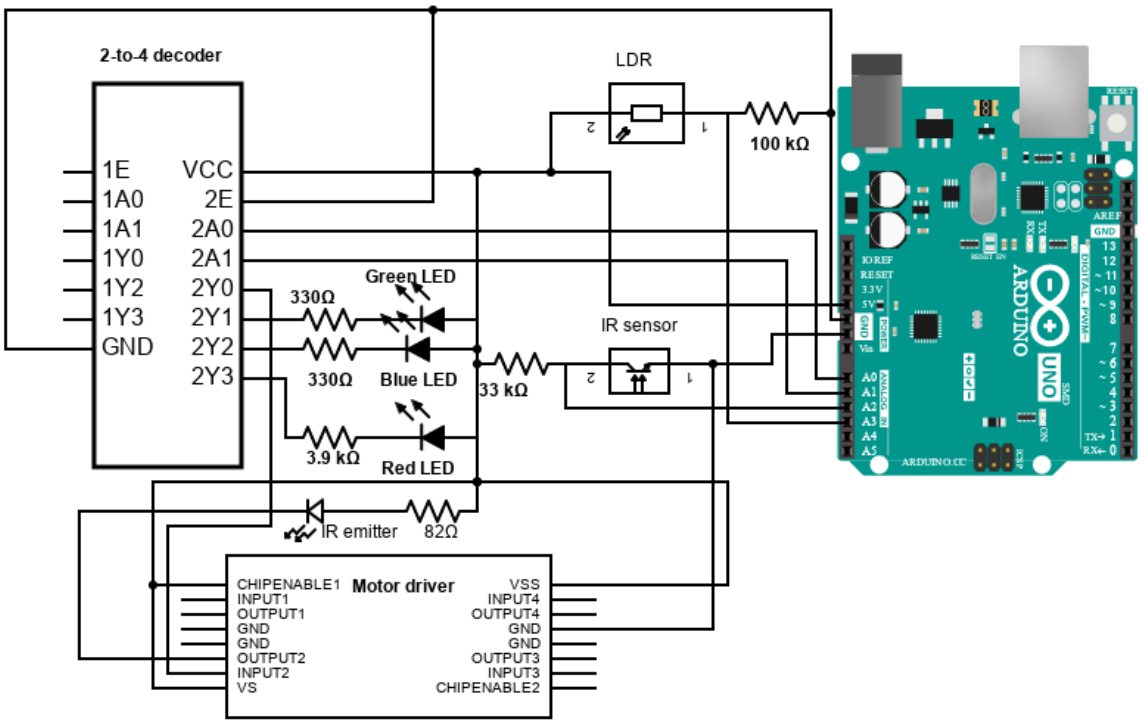
Our project, the A-maze-ing Race is a maze which our robot has to clear in the shortest time. Along the way, there are various colour challenges which the robot would face and the robot needs to react differently according to the various colours it detects in order to complete the race.

1.2 The robot

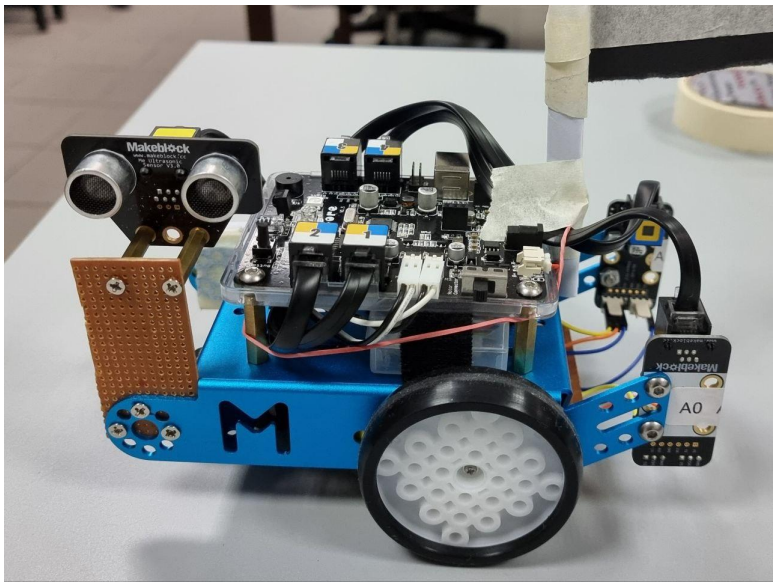
We built our robot with the various components provided, as listed below

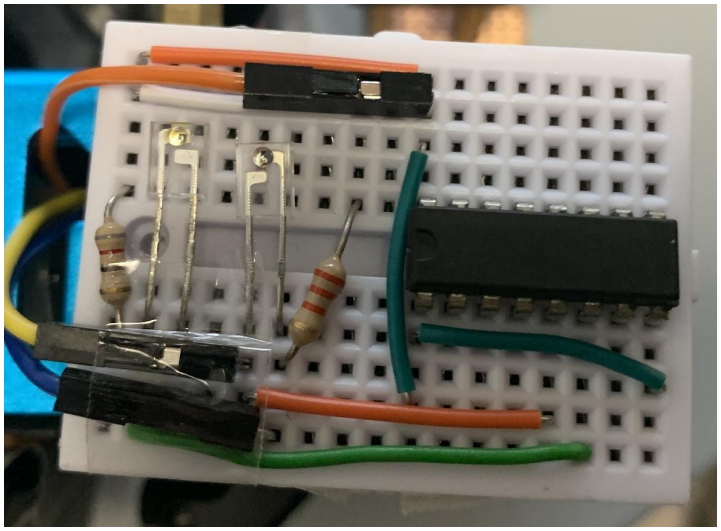
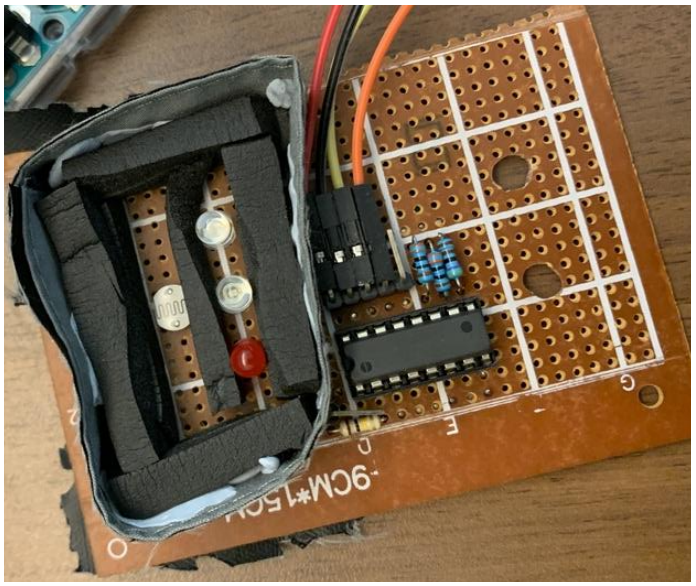
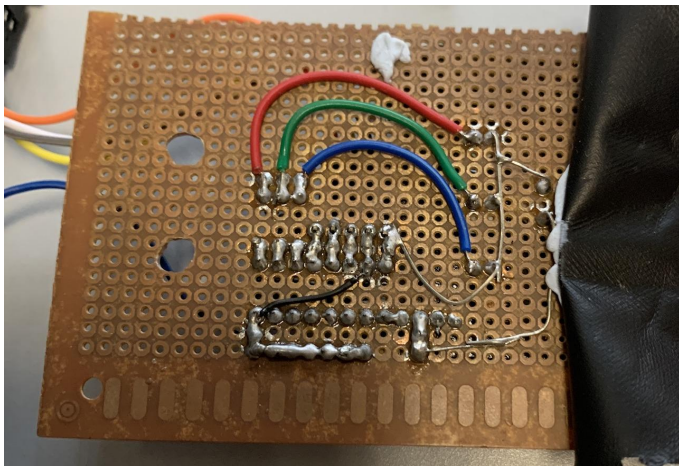
1. 1 Mbot Chassis
2. 2 Motors
3. 1 Mbot ultrasonic sensor
4. 1 Mbot line-follower sensor
5. 2 Wheels
6. 1 Infrared emitter
7. 1 Infrared receiver
8. 1 HD74LS139 2-to-4 decoder chip
9. 1 L293D Motor driver
10. 1 Red LED
11. 1 Blue LED
12. 1 Green LED
13. 2 330 Ohms resistor
14. 1 3900 Ohms resistor
15. 1 100k Ohms resistor
16. 1 33k Ohms resistor
17. 1 82 Ohms resistor
18. 1 Light Dependent Resistor

1.3 CAD diagram of the connections for the infrared and colour sensor



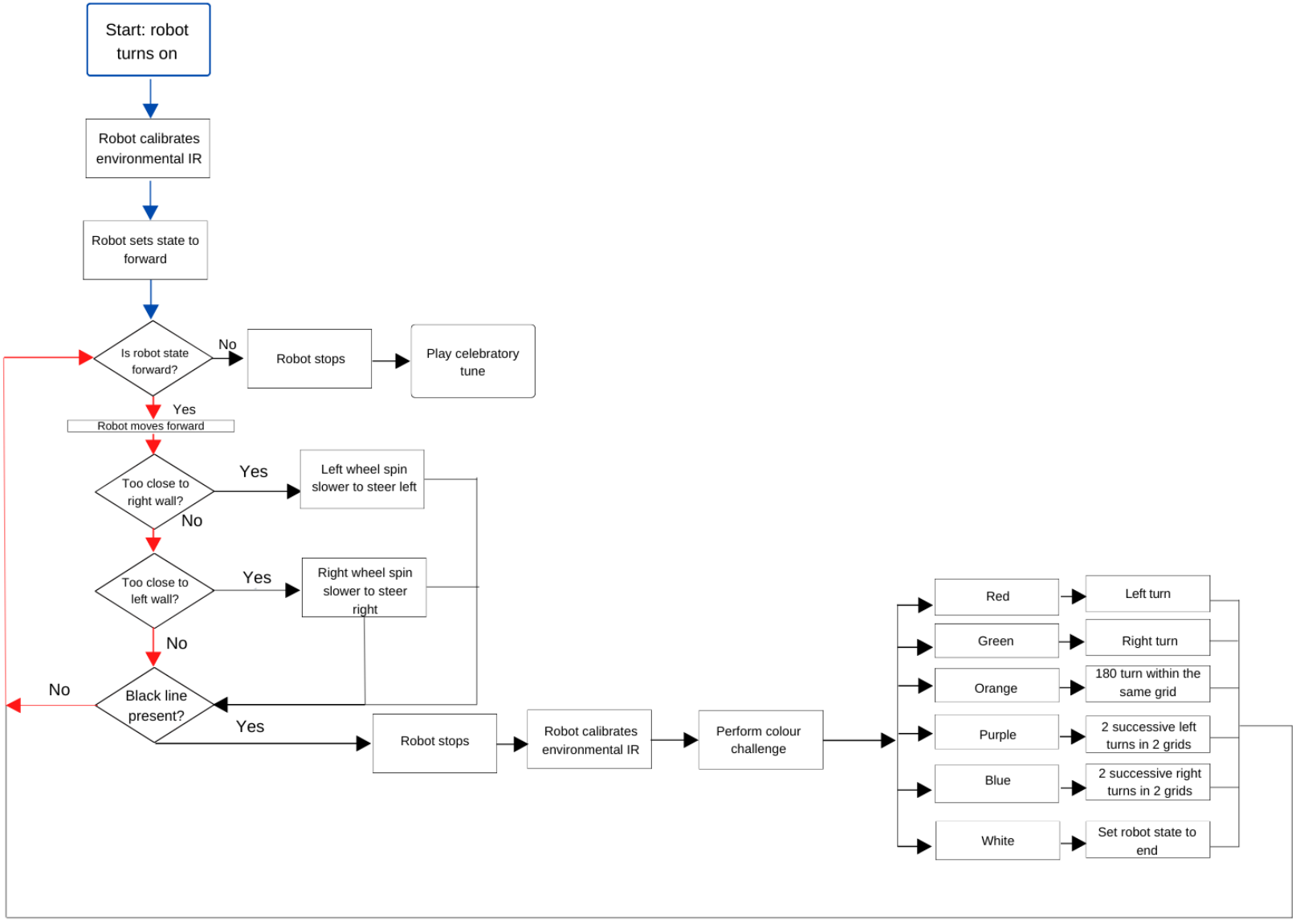
1.4 Pictures of Mbot and Breadboard Circuits

	Overview of Mbot

	Circuit of the IR sensors with cut down wires wired neatly. Secured to the side of Mbot Chassis with double sided adhesive foam and clear tape.
	Front side of the Colour Sensor circuit with soldered IC socket, LEDs, LDR, Resistors and Male connector pins. Holes drilled to secure to Mbot chassis screw holes.
	Reverse side of the Colour Sensor breadboard circuit with clean soldering connections and LED cables colour labelled. Opaque fabric shields LDR from light from top.

2. Overall algorithm

Startup sequence is highlighted in blue which then leads to the main loop which is highlighted in red.



2.1 Startup

Robot performs the usual startup sequence and sets the pins to their correct input and output specifications (not in image). It obtains the environmental IR reading that calibrates the surrounding environmental IR reading and stores it (refer to 3.4). It then sets the robot state to forward and enters the main loop.

2.2 Robot state check

At the start of the loop, it checks for the state of the robot. If the robot state is “forward”, it will enter into the movement loop. Otherwise, if the robot state is “end”, it will stop the robot and play the celebratory tune, indicating the end of the maze.

2.3 Moving forward

To allow the Mbot to move forward in a straight line, the ultrasonic sensor and IR sensor are installed on left and right side of the Mbot respectively to ensure the Mbot is approximately centred and does not veer too close to either wall.

This is done by decreasing the speed of the left wheel if the IR sensor detects that the Mbot is close to the right wall to steer left and decreasing the speed of the right wheel if the Ultrasonic sensor detects that the Mbot is too close to the left wall to steer right.

By having one wheel spin slower than the other, the robot steers towards the side that has the slower wheel.

2.4 Detecting the black strip

The black strip is detected using a line sensor. Once the line sensor detects the black line, the algorithm will stop the Mbot and measure the environment IR values (refer to 3.4). It will then proceed to the Colour checking algorithm.

2.4 Colour checking algorithm

The Mbot checks the colour of the paper. This is done using the colour sensor. This is done by flashing the red, blue, green LEDs one by one to see how much of each light is reflected off the LEDs. Each colour would reflect a different composition of light, which is calibrated with RGB values of coloured paper. For more details refer to (3.2). The Mbot would perform the actions of the colours.

2.4.1 Movement based on colours

For Red, a left turn was executed.

For Green, a right turn was executed.

For Orange, a u-turn was executed by turning left twice.

For Purple, a left turn is executed, followed by advancing forward and turning left again.

For Blue, a right turn is executed, followed by advancing forward and turning right again.

For White, the robot state is set to end, which will result in the check in (2.2) stopping the robot and playing the celebration tune.

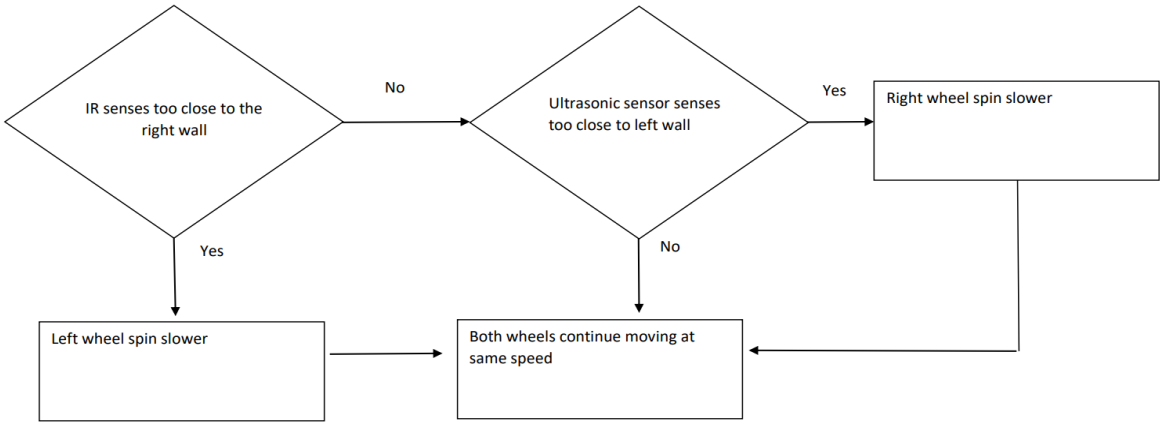
After the colour check, the algorithm goes back to the forward check and continues the loop until a white colour check results in the robot finishing the maze.

3. Implementation details of subsystems

Different subsystems were used to implement the algorithm.
Summarily:

1. Ultrasonic and IR sensor were used to prevent collision of the MBot with the left and right walls and keep the robot moving straight
2. Colour sensing system was used to detect the colour at each waypoint
3. Line detector was used to detect for the presence of the black strip indicating the waypoint being reached

3.1 Keeping robot moving straight



To keep our robot straight, we placed the ultrasonic sensor on the left side of the robot and an infrared sensor on the right side. Both sensors have to maintain a fixed distance from the walls so that the robot would be moving in a straight line.

At the left side of the robot, a Mbot ultrasonic sensor was used. The implementation used is rather straightforward, with the robot steering slightly right by decreasing right motor speed if the Ultrasonic sensor was too close to the left wall. This value is stored as BASELINEULTRA and is used to calibrate the robot.

```
//if the Ultrasonic sensor is too close to the left wall
      if(ultra_sensor.distanceCm() < BASELINEULTRA ){
          //course correct slightly right by decreasing the right
motor speed
          left_motor.run(-FORWARD_SPEED);
          right_motor.run(FORWARD_SPEED - RIGHTCORRECTION);
      }
```

On the right side of the robot, a custom infrared sensor was used. The implementation used is also straightforward, although calibration is required. Calibration of the infrared sensor is required from time to time as the IR reading from the environment is not consistent, therefore at the start of the maze and at every colour waypoint, a reading of the IR detector with the IR emitter switched off will be taken and stored as Env_IR (environment IR). A BASELINEIR reading represents the offset from environmental IR that steers the robot slightly left by decreasing left motor speed if the IR sensor was too close to the right wall.

```
//if the IR sensor is too close to the right wall
      if(infrared_distance() > (BASELINEIR + Env_IR) ){
          //course correct slightly left by decreasing the left
motor speed
          left_motor.run(-FORWARD_SPEED + LEFTCORRECTION);
          right_motor.run(FORWARD_SPEED);
      }
```

3.2 Colour Checking

To check for the colour of the paper at each waypoint, a euclidean distance minimising algorithm is used. This was built on top of the provided library code for the colour sensing lab with additional algorithms to identify the colour closest to the detected colour.

First of all, calibration of the grey and black levels are taken by a separate program. This provides use with the greyCalibArr and the blackCalibArr values. This program is adapted from the lab program of calibrating the black and white sample readings. Details on the algorithm for checking and calibrating is provided in (4.2).

Next, the values of each of the individual colours of paper were taken and defined in the library. This provides the algorithm with a reference point to compare the detected colours with.

```
//COLOR SENSOR CALIBRATION
// Define time delay before the next RGB colour turns ON to allow LDR to
stabilize
#define RGBWait 300 //in milliseconds

// Define time delay before taking another LDR reading
#define LDRWait 10 //in milliseconds


//set these based on calibration values
#define greyCalibArr {356.00,60.00,302.00}
#define blackCalibArr {610.00,948.00,620.00}
#define redArr {255.00,102.00,71.77} //RGB values for Red Paper
#define greenArr {115.32,199.75,146.92} //RGB values for Green Paper
#define blueArr {178.73,230.88,246} //RGB values for Blue Paper
#define whiteArr {255.58,258.45,257} //RGB values for White Paper
#define blackArr {23.64,0.00,1.69} //RGB values for Black Paper
#define purpleArr {228.50,178.50,195.89} //RGB values for Purple Paper
#define orangeArr {258.58,170.00,76.84} //RGB values for Orange Paper
```

Next we have a function to calculate the euclidean distance of the measured colour from the reference colours in the array. Note that in actuality this is the euclidean distance of the colour squared. This is to save on the limited computing resources of the Arduino by ignoring to square root the resulting sum. This does not affect the following euclidean distance minimization algorithm as all values are squared.

```
//function to calculate the euclidean distance squared of colorArray and input
array
float euclideanDist(float colorArray[], float inputArray[]){
  float sum = 0;
  for(int i = 0; i < 3; i++){
    sum += (colorArray[i] - inputArray[i])*(colorArray[i] - inputArray[i]);
  }
  return sum;
}
```

Using the following getColour algorithm, we iterate over all the stored colour arrays and compare them against the measured colour array. The index of the stored colour array with the lowest euclidean distance from the measured colour array is returned.

```
int getColour(){
 //turn on one colour at a time and LDR reads 5 times
 for(int c = 0;c<=2;c++){
   ledArray(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;
   ledArray(3);  //turn off the current LED colour
   delay(RGBWait);
 }
 //iterate over the array of colour arrays and find the closest match with
minimum euclidean distance
 float minDist = euclideanDist(detectArr[0],colourArray);
 int minIndex = 0;
 for(int i = 1; i < 6; i++){
   float dist = euclideanDist(detectArr[i],colourArray);
   if(dist < minDist){
     minDist = dist;
     minIndex = i;
   }
 }
 return minIndex;
}
```

3.3 Checking for waypoint

A line sensor is used such that when the sensor detects a black line, it would stop and execute the colour check algorithm.

```
bool reached_waypoint() {
  int sensor_state = line_finder.readSensors();
  return sensor_state == S1_IN_S2_IN; // S1_IN_S2_IN is the state
when both sensors are on the line
}
```

3.4 Environmental IR calibration reading

As mentioned in (3.1), the environment IR reading is taken. To obtain this reading, a function was written:

```
//update the environment IR reading
void getEnvIR(){
 ledArray(1); //turn on green LED to switch off IR Emitter
 delay(300);
 Env_IR = analogRead(IR);
 ledArray(3); //turn off green LED to switch on IR Emitter
}
```

That sets the global IR environment reading Env_IR:

```
//Environment IR reading
int Env_IR;
```

Which is updated at the start of the maze:

```
void setup(){
  //setup the outputs for the colour sensor and IR sensor
  pinMode(PLEX1,OUTPUT);
  pinMode(PLEX2,OUTPUT);
  pinMode(LDR,INPUT);
  pinMode(IR,INPUT);

  getEnvIR(); //get environment baseline values for IR

  //set the robot state to forward
  robotState state = forward;
}
```

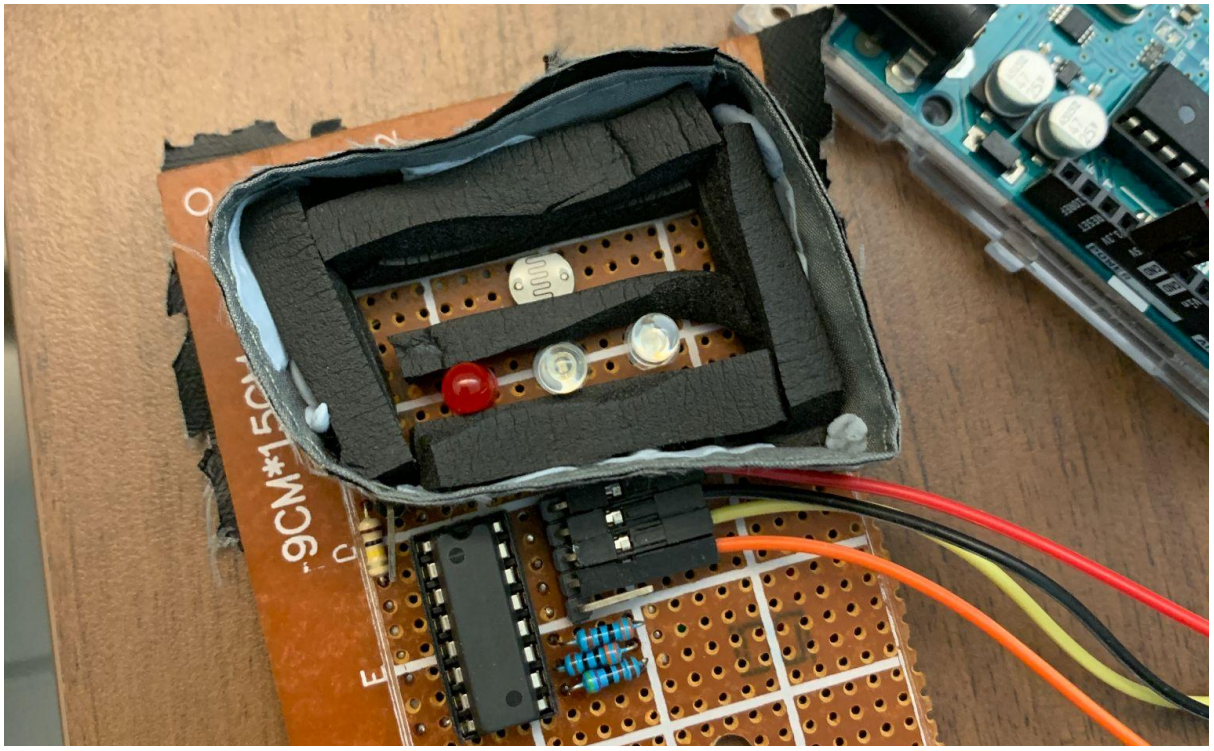
And at every colour checkpoint:

```
//if the line follower sensor detects a black line stop
    if(reached_waypoint()){
      left_motor.stop();
      right_motor.stop();
      getEnvIR(); //get the environment IR reading
      delay(500);
      // run the light challenge
      switch (getColour()){
        case 0: turnLeft(); break; // red : left
        case 1: turnRight(); break; // green : right
        case 2: uTurn(); break; // orange : 180 deg within grid
        case 3: doubleLeft(); break; // purple : 2 left
        case 4: doubleRight(); break; // blue : 2 right
        case 5: state = end; break; // white : End of maze
      }
    }
```

4. Calibration and improving the robustness of sensors

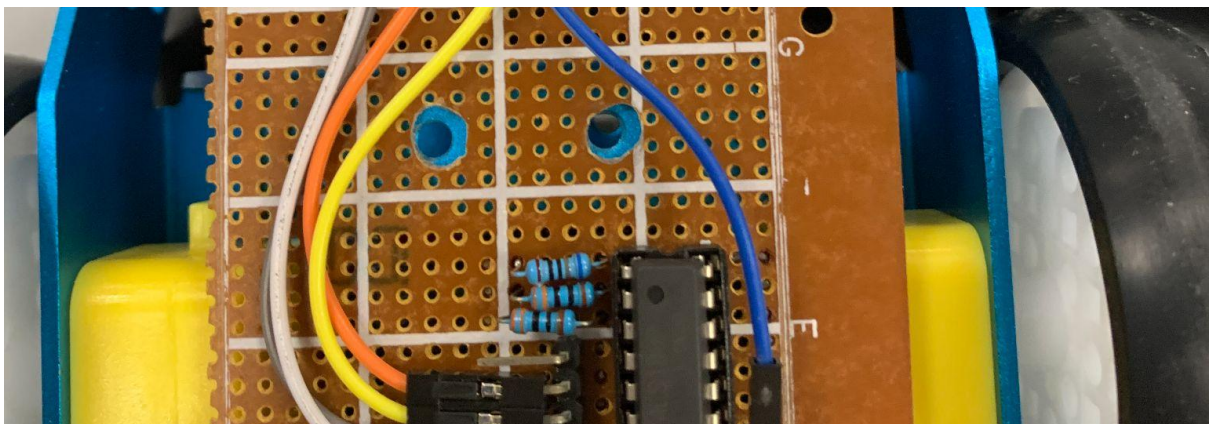
4.1 Colour sensor robustness engineering

Our group has created a customised soldered breadboard design as shown in the image below



This confers a few key advantages to the colour sensor design

1. Easier to mount on the bottom side of the Mbot using screws to the bottom screw holes instead of a suspended breadboard design



The screw holes are shown above and the breadboard can be mounted to the base securely using screws. This is in contrast with other groups that use adhesives to secure the solderless breadboard to the MBot. This may result in the board falling and inconsistent height values causing calibration issues for the LDR.

2. More space to work with for electronics design. The usage of a larger soldered breadboard compared to the usual small solderless breadboard provided allows for easier and more flexible arrangement of electronic components.

3. More secure connection of components. Soldering components and wires to the breadboard prevents any loose connections or wiring from affecting the circuit. This is a more reliable method of connection compared to using a solderless breadboard.

Other colour sensor engineering solutions for reducing light entry are as follows:

1. Dark foam is stacked around the colour sensing main unit to prevent surrounding light from affecting readings
2. Dark foam is also used to barricade the LEDs from the LDR to prevent direct LED light from shining into the LDR and only reflected light for more accurate readings
3. The attachment of an opaque skirt using Blu-Tack further blockades any surrounding light
4. The black skirt is also attached to the other side of the soldered breadboard to prevent any above light from shining into the colour sensing main unit

Apart from these, we have noticed that the LDR is particularly sensitive to red and thus often returns erroneous values that overshoots the actual red value. To counter this, we have increased the resistor value to bias the Red LED from the recommended 2900 Ohms to 3900 Ohms for a more consistent red reading result.

4.2 Colour sensor Calibration

Refer to (3.2) for the basic idea. For more details on the process and program used, they are as follows:

1. Obtain Grey and Black Calibration Array values. This is done through the program GreyBlackCalibration, and attached in the source code file.

From serial monitor:

```
grey calib array: {356.00,60.00,302.00}
black calib array: {610.00,948.00,620.00}
```

2. Transfer the Grey and Black Calibration Array values to the ColourCalibration program:

```
//set these based on calibration values
#define greyCalibArr {356.00,60.00,302.00}
#define blackCalibArr {610.00,948.00,620.00}
```

3. For each paper colour, the RGB values are taken and stored. This is done through the ColorCalibration program, and attached in the source code file. Each colour is measured by resetting the Arduino after each paper colour reading is taken. The Arduino outputs the data in the serial monitor.

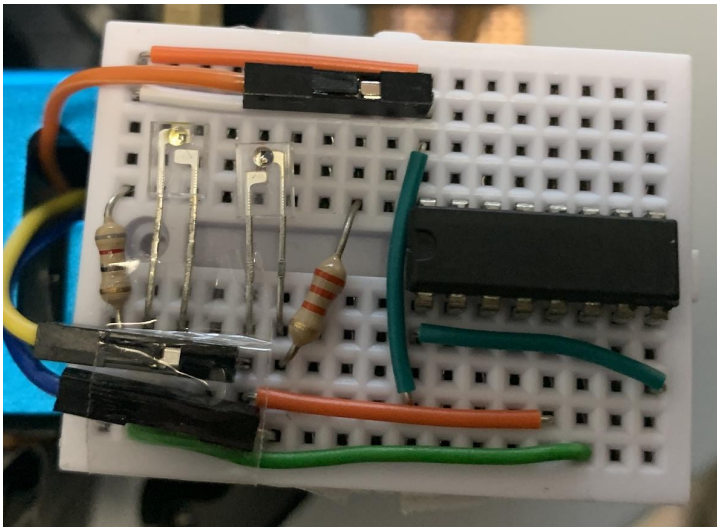
From serial monitor:

```
Color Array: {255.00,102.00,71.77}
//repeated for all 7 colours
```

4. Transfer the Grey and Black Calibration Array values and RGB values of each paper to the main robot program, calibration is now completed.

```
#define redArr {255.00,102.00,71.77} //RGB values for Red Paper
#define greenArr {115.32,199.75,146.92} //RGB values for Green
Paper
#define blueArr {178.73,230.88,246} //RGB values for Blue Paper
#define whiteArr {255.58,258.45,257} //RGB values for White Paper
#define blackArr {23.64,0.00,1.69} //RGB values for Black Paper
#define purpleArr {228.50,178.50,195.89} //RGB values for Purple
Paper
#define orangeArr {258.58,170.00,76.84} //RGB values for Orange
Paper
```

4.3 IR sensor robustness engineering



The IR sensor is installed on the mini breadboard with tight short wires. Care was taken to avoid obstructing the IR emitter and receiver.

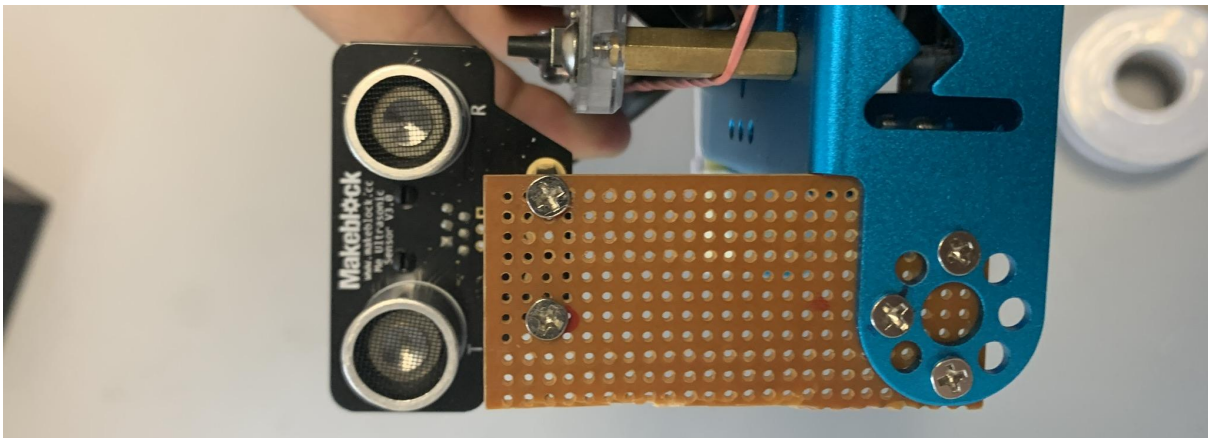
4.4 IR sensor calibration

Initially the sensitivity of the IR sensor was poor as the analog read voltage readings were constantly maxed out at 1024 due to inappropriate resistor values being used. Therefore our team has experimented with different resistor values for the IR emitter and receiver to find the optimum resistor values for the sensor unit. This allows for accurate values of voltage values from the IR sensor.

Our team also had issues dealing with the ambient environmental IR which fluctuated and affected readings. Refer to (3.4) for how we solved this issue.

To obtain the baseline value offset from the IR sensor required, we placed the robot in a central position between two plastic panels and the IR offset required is calculated from the serial monitor output. Some allowance is set for the robot to remain on a relatively stable course.

4.5 Ultrasonic sensor robustness engineering



Our team was having issues with the ultrasonic sensor hitting the walls with tight turns. This is caused by the ultrasonic sensor sticking too far out forwards. Therefore we have made a breadboard based height extension to allow for the sensor to be installed closer to the main body of the robot.

4.6 Ultrasonic sensor calibration

To obtain the baseline ultrasonic distance, we placed the robot in a central position between two plastic panels and the ultrasonic distance required is calculated from the serial monitor output. Some allowance is set for the robot to remain on a relatively stable course.

4.7 Line detector

Since the line detector used was an integrated component and standard library used for measuring output, no engineering and calibration details are required.

5. Significant difficulties and steps taken to solve

1) Wiring on the solderless breadboard is less secure and large difficulty in attaching the colour sensor breadboard to the Mbot securely

Connection of the wiring on the solderless breadboard housing the colour sensing unit easily comes loose when the robot runs on the course. This then causes the robot to not work as intended. Furthermore, there's a difficulty in attaching the breadboard to the bottom of the Mbot. Although we can use the double sided tape, it is not very secure. Thus, we changed from the solderless mini breadboard into a soldered breadboard to connect all the components. Not only that this will prevent a loose connection, but we can also easily use a bolt on the breadboard to attach it to the robot.

2) Colour sensor sensing surrounding light

Our colour sensor was sensing surrounding light as the light was able to enter due to lack of sufficient precautions to prevent the ambient light from reaching the colour sensor through the sides of the robot. As a result, when we tested the colour sensor in different environments, different values of reflected light were returned.

Besides, the colour sensor was not only sensing the reflected light from the various pieces of coloured paper, it was also sensing directly the light produced from the RGB led. We did not want this to happen as it would affect the reliability of our results, since the values we wanted to calibrate should be strictly from the reflection of light off the different coloured paper.

We came up with an idea to solve both issues. We applied opaque black foam in a square shape around the colour sensor and RGB lights, to prevent the surrounding lights from entering the sensor, followed by a piece of foam in the middle barricading the LEDs from the LDR. This ensures that all light detected is from the reflection of LEDs with the colour paper, preventing the light from the LEDs from entering into the LDR directly.

Next, the colour sensor was also detecting light entering through the holes of the soldered breadboard. We used a soldered breadboard instead of a solderless breadboard as mentioned in (1). The soldered breadboard is not opaque unlike the solderless breadboard. As a result, surrounding light was able to enter the sensor through the holes. To resolve this, we added a piece of black fabric in between the top of the breadboard and the robot base.

3) Values on infrared sensor is inconsistent due to fluctuating environmental IR

When we first tested our robot, the robot was seen to move left and right constantly, in a zig zag behaviour which was not ideal as we wanted the robot to move in as straight a line as possible. Hence, we disconnected the IR sensor from the robot and tested its sensitivity in the course.

We found out that at different spots on the course, the IR detected different values even though it was placed at the same distance from the walls of the course, the values output from the serial monitor were different. This resulted in the robot sensing that we are too far from the wall at that particular spot, when actually the robot is of acceptable distance when tested at another spot, due to the difference in environmental IR values, steering unnecessarily. We found out that this is due to the fluctuating ambient environmental IR leading to inconsistent decision making.

At certain spots where it is surrounded by the white pieces of ‘walls’, there is less surrounding light entering from the sides as compared to those places without the ‘walls’, resulting in the values differing. To resolve this, we make the IR emitter turn off and on at different parts of the maze such as at the start and at each colour waypoint. When the IR emitter is off, the environmental IR reading is recorded as Env_IR. The robot will use this calibrated value and offset it to check if it's approaching too close to the right wall. Thus, this will constantly calibrate the sensor with the changing amount of environmental IR.

4) Robot steering sharply due to unsuitable course correction algorithm

To keep our robot straight, we used an ultrasonic sensor on the left side and IR on the right. If the IR value is less than a threshold value, or the ultrasonic sensor senses a value more than the threshold value, the robot would turn in the direction opposite to where the respective sensors are placed.

Initially, the algorithm was such that the speed of the wheel opposite to the side of intended steering direction (left / right) was increased, such that the robot would steer to the left by increasing right motor speed and steer to the right by increasing left motor speed.

However upon testing, we noticed that the robot was steering sharply due to the increased speed. This caused the robot to immediately steer back in the opposite direction as the distance threshold on the other side of the robot was reached immediately. This caused the robot to move in a zig zag behaviour. We realised that the increase in speed of the wheels caused the robot to oversteer.

Hence, we came out with a new algorithm that reduced the speed of the wheel in the intended steering direction instead. This made the robot turn smoothly and significantly reduced the zig zag behaviour.

5) IR sensor cabling and components getting caught in wheels

Initially when we first tested the robot, we constructed the IR sensor with lengthy pin copper wires and jutting out components. However the robot wheels would occasionally get caught with the wiring, causing it to be yanked out and the circuit to disconnect.

Hence we had to shrink the circuit and ensure that there are no exposed lengthy wires. To resolve this, we used solid core wires that are cut and stripped. The trimmed solid core wires were attached close to the breadboard, preventing it from obstructing the robot's movement. Excess length of resistors and IR emitter and receiver is also removed.

6. Evaluation and Further Improvement

Video links of the evaluation attempts are provided:
MBot Test Run 1
MBot Test Run 2

Our MBot is able to complete all the courses well, both during the trial and the test run. Throughout the course, our robot also runs straight, when the sensors on the side detect the walls close, it automatically corrects itself and steers slightly to the other side, and continues to move straight. It also correctly identifies the colour and turns without bumping the walls, even in different places that have different lightings. This reflects an enough blockage for the sensor from the surrounding light. Calibrated by the algorithm we used to determine the colour, our colour sensor can be considered accurate.

However some turns understeer, causing it to be rather close to the wall before it is able to move back to the middle of the track. Thus, further calibration by adjusting the motor turn speed and turn duration will be able to make the robot run more consistently.

Also, the robot can be made to run faster by increasing the forward speed set.

7. Work division

Dylan was in charge of the code, ensuring that both sensors work properly so that the Mbot runs nicely and detects each colour correctly.

Ella was in charge of wiring the robot, ensuring there are no loose connections and fixing wiring problems when they surface.

Cayenne was in charge of reading through the report to ensure that the sentence structure is well developed.

Khanh was in charge of testing the robot on the sample maze provided and changing the code to ensure the turn radius and speed of the robot was suitable.