THE AMAZING RACE PROJECT BY B03 SECTION 5 TEAM 1

Ong Yong Chein (A0235421R), Ong Zhi Hong (A0231087J), Oong Jin Rong Jared (A0180023X), Ow Yong Jin Xuan (A0240051B)

Table of Contents

1. Introduction

1.1. The A-maze-ing Race Project

The grand project of CG1111A: The A-maze-ing Race, is a maze challenge in which our group is tasked to design and build a wheeled robot that can navigate a maze in the shortest time possible, solving challenges at numerous waypoints in the maze. Given the set of requirements that can be found in Appendix A at the end of the report, our group has designed a mBot over the course of three weeks that managed to successfully navigate the maze. In this report, we will show and explain the working principles behind our mBot and algorithm.

1.2. Our mBot

Our mBot consists of:

1. 1 x Metallic structure for body of mBot
2. 1 x Ultrasonic sensor
3. 1 x Infrared (IR) emitter and detector
4. 1 x Common anode RGB LED
5. 1 x Light Dependent Resistor (LDR)
6. 1 x HD74LS139P 2-to-4 Decoder IC Chip
7. 7 x Resistors of varying resistance
8. 1 x Line sensor
9. 2 x DC Motors and wheels
10. 1 x mCore

Figures 1.2.1 to 1.2.4 show the completed mBot and the position of the various components. Figure 1.2.5 shows the schematic for circuit logic of our mBot, excluding line sensor, ultrasonic sensor, and DC motors.

Figure 1.2.1: Top view of mBot.

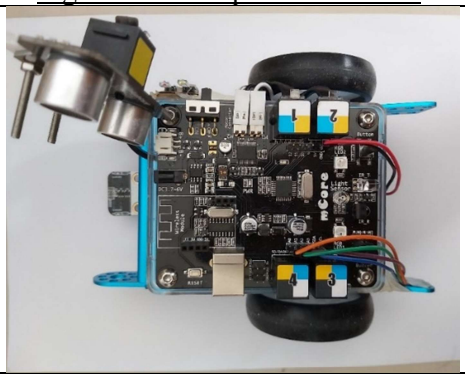


Figure 1.2.2: Bottom view of mBot.

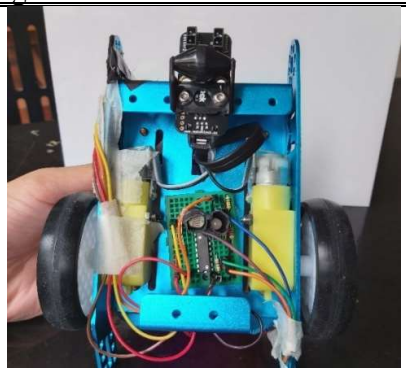


Figure 1.2.3: Right side view of mBot.

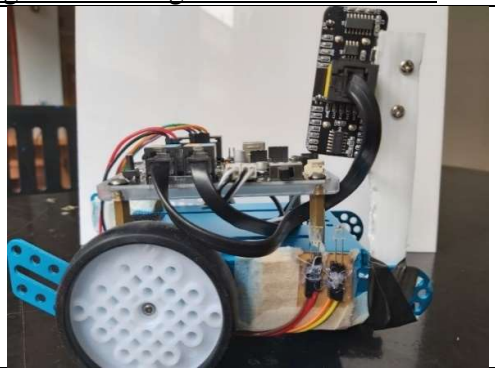


Figure 1.2.4: Left side view of mBot.

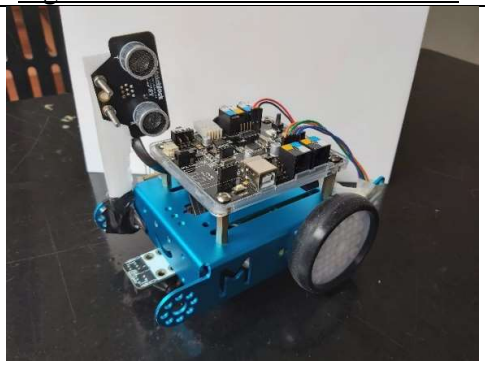
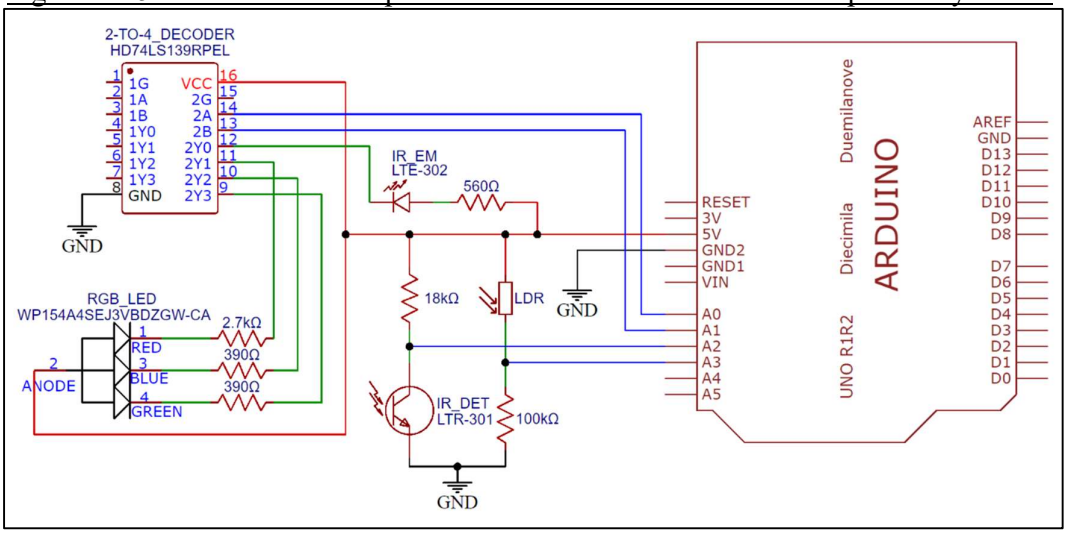


Figure 1.2.5: Schematic of components used for colour sensor and IR proximity sensor.

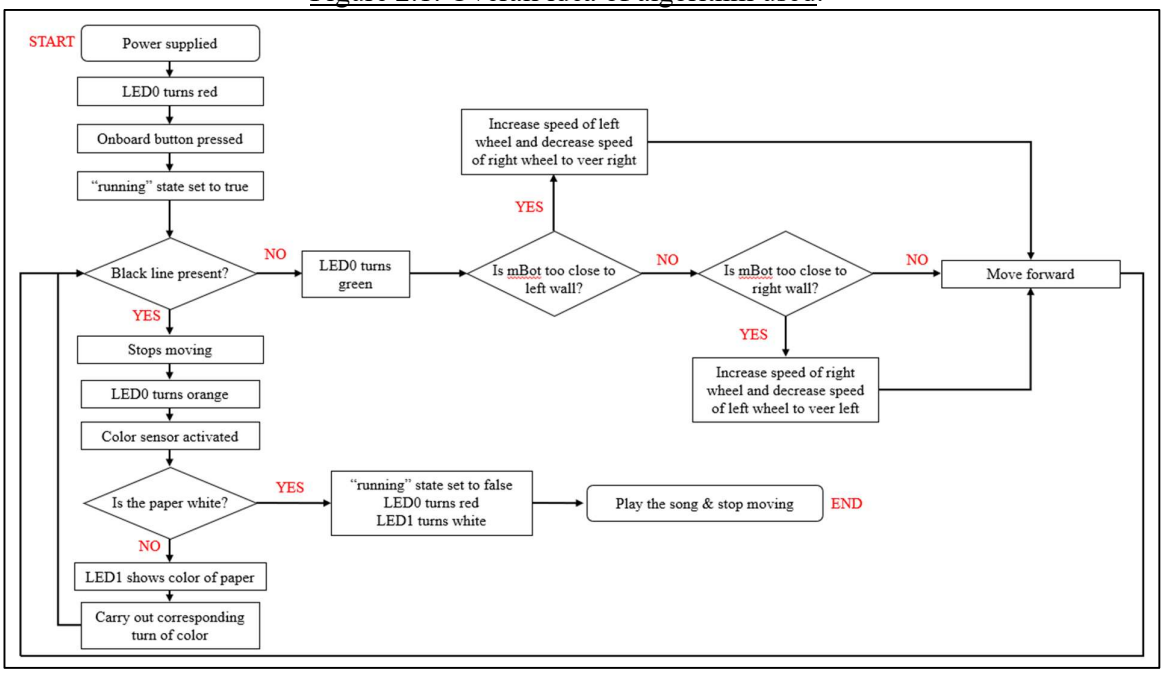


547Ω resistor was used to bias IR emitter. actual current: 7.2mA < 8mA

2. Overall Algorithm of mBot

Figure 2.1 shows the main idea of the algorithm used by the mBot to navigate the maze and clear the different challenges. The code corresponding to this flowchart can be found in the loop function of mBot.ino.

Figure 2.1: Overall idea of algorithm used.



2.1. Initial Power On

When the mBot is powered on, LED0 displays red to indicate waiting status. The mBot then waits for the onboard button to be pressed, at which point "running" variable is set to true.

2.2. Moving Forward

To indicate that the mBot is running, LED0 turns green. The mBot then enters a loop starting with a conditional check to see if a black strip is present. At the start of the loop, the mBot checks if it is in the middle, and if not, which wall it is then closer to. We steer right if the ultrasonic sensor deems that the mBot is too close to the left by increasing the speed of the left wheel and decreasing the speed of the right wheel. We steer left if the IR proximity sensor deems that the mBot is too close to the right by increasing the speed of the right wheel and decreasing the speed of the left wheel. If the mBot is in the middle, no adjustments are made and the mBot will move with both wheels at same speed. This process will continue while the black strip is not detected.

2.3. Colour Detection

Once a black strip is detected, the mBot stops, and LED0 displays orange to indicate that the mBot is currently reading the colour paper. The mBot checks what the colour of the paper beneath it is using the colour sensor. The corresponding colour is then shown on LED1. If white is detected, the mBot has reached the end of maze. The "running" variable is then set to false, LED0 shows red to indicate that it is not running, then the mBot will play a celebratory song and remain on the spot. Otherwise, the mBot will execute the corresponding turn based on the colour. Once the mBot has executed the corresponding turn, it will restart the loop function described in the section above.

3. Implementation Details of Subsystem

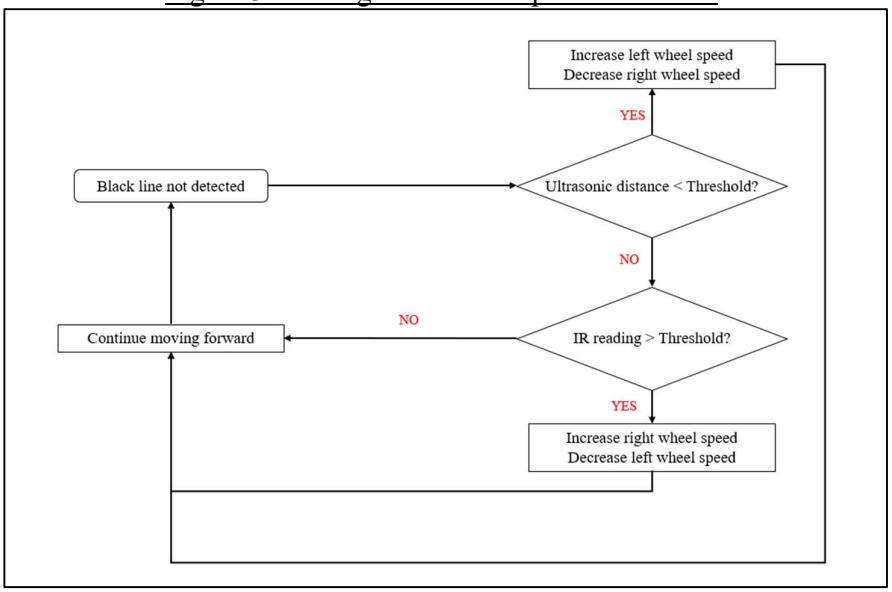
Subsystems were utilized to meet the project requirements. Specifically,

1. Ultrasonic and IR sensors were used to keep the mBot moving straight
2. Colour sensor was used to allow the mBot to detect the colour of the paper at each waypoint
3. Line detector was used to detect presence of a black strip.

3.1. Keeping the mBot centred

To keep the mBot in the centre of the track, both the ultrasonic and IR sensor were used. Figure 3.1.1 below shows the looping algorithm used specifically for this portion. The code corresponding to this flowchart can be found in mBot.ino, lines 135-157 and 03_helper.ino, lines 8-26.

Figure 3.1.1: Algorithm to keep mBot centred.



3.1.1. The Left Side

An ultrasonic sensor was used to check the distance between the mBot and left wall. Since each grid has a width of 27cm, while the mBot is only 13cm wide, it leaves the about 7cm on each side of the mBot when the mBot is perfectly centred. Considering the minimum working distance of 3cm of the ultrasonic sensor, placing the sensor on the left edge of the mBot can result in inaccurate readings in scenarios where the mBot gets too near the wall, especially during left turns. To avoid this issue, the sensor was mounted on the right of the robot while facing left. This ensured that the distance between the ultrasonic sensor and the left wall would always be more than 3cm. The built-in library function of the mBot was used to read in the distance as detected by the ultrasonic sensor. If the distance detected was less than the threshold distance of 15cm, we steer the mBot to the right by increasing the speed of the left wheel and decreasing the speed of the right wheel.

Figure 3.1.2 below shows the pseudo code the ultrasonic sensor. The threshold value is a defined constant of 15cm away from the wall.

Figure 3.1.2: Code snippet for using ultrasonic sensor (02_sensors.ino, lines 6-9).

```
float get_ultrasonic_distance() {
  return ultrasonic reading
}
```

Figure 3.1.3: Code snippet for left steer (mBot.ino, lines 137-146).

```
if mBot is closer to the left wall {
  increase speed of left motor
  decrease speed of right motor
}
```

3.1.2. The Right Side

An IR proximity sensor was used to check if the mBot was too near the right wall. Since the limitation of the effective range was not applicable to the IR sensor, it was positioned on the right side of the mBot and facing the right wall. Due to the fluctuation of ambient IR at different points of the maze, we had to ensure that the proximity sensor was making decisions based purely on the reflected IR. The elimination of the influence of ambient IR would be elaborated in Section 4. When the sensor reading exceeds our threshold value, the mBot is too close to the right wall. The speed of its right wheel would be increased, and the speed of the left wheel will be decreased to veer it to the left. Like the ultrasonic sensor, the threshold value is a pre-determined constant which is the reading for IR when the mBot is 5 cm away from the wall.

3.2. Checking for waypoint

To determine whether the mBot should continue moving forward, a line sensor was used. Since there would be a black strip at every waypoint challenge, the line sensor was used together with the inbuilt library function to check for the presence of the black strip while moving forward. In the case that mBot reaches the black strip at an angle, the mBot is coded to stop when either, or both sensors detect a black strip. Stopping at the earliest time possible minimises the chances of the mBot crashing into the wall ahead. The colour sensor is then used to solve the waypoint challenge, by turning in the corresponding direction of the colour identified and continue moving forward. This process repeats until white is detected, which marks the end of the maze. The pseudo code for the line sensor is shown in Figure 3.2.1.

Figure 3.2.1: Code snippet for line sensor (mBot.ino, lines 131-158).

```
while(running && !reached_black_line()) {
  check if mBot is too close to the sides

  if mBot is too close to left:
    steer right
  else if mBot is too close to right:
    steer left

  move forward
}
stop moving
```

Figure 3.2.2: Code snippet for detecting black line (02_sensors.ino, lines 17-20).

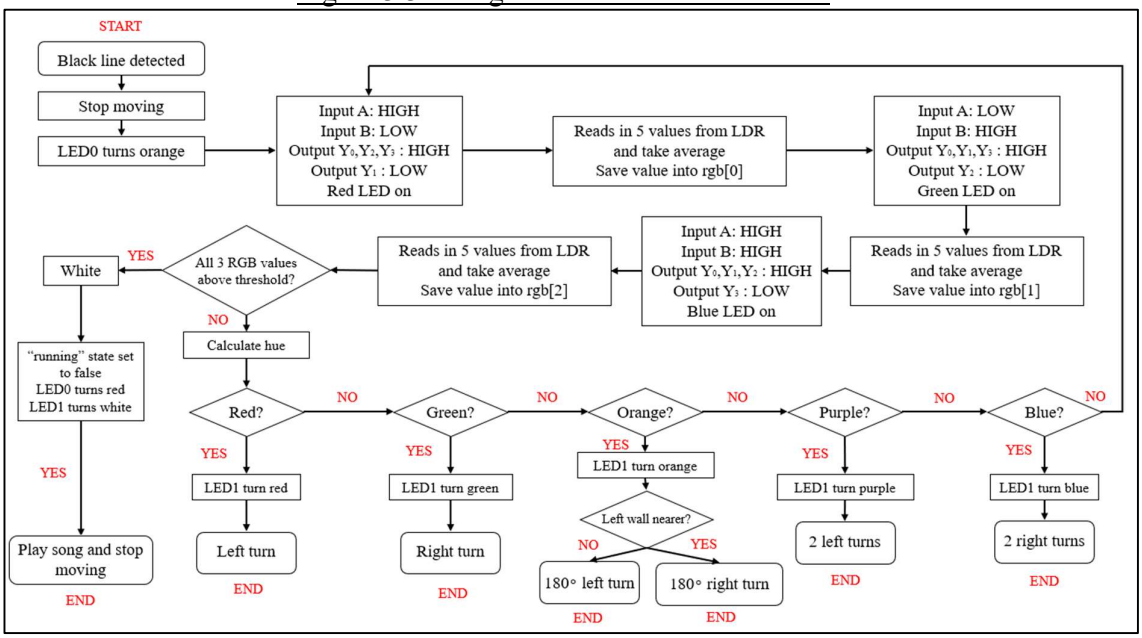
```
bool reached_black_line() {
  return true when black line is reached, false otherwise
}
```

3.3. Solving Waypoint Challenge

To solve the waypoint challenges, a colour sensor was used. Using the HD74LS139P 2-to-4 Decoder IC Chip, we control when we want to switch on the red, green, and blue components of the RGB LED. A delay is added between each colour to account for the response time of the LDR so that the readings stabilise before taking the reading. As such, we ensure that the reading is not affected by the previous colour.

We calculated the hue of the colour paper from the average measured RGB values, then compared it to the hardcoded hue values for each individual colour. Whenever the mBot detects a black strip, it would proceed to carry out the algorithm shown in Figure 3.3.2. The code corresponding to this flowchart can be found in 01_motor_control.ino, lines 91-111, 02_sensors.ino, lines 40-99, 03_helper.ino, lines 1-139

Figure 3.3.2: Algorithm for colour detection.



If the colour identified is not white, the colour_turn function executes the different turns, where the motors are adjusted to different speed and direction. To make a single left or right turn, the motors run at the same speed but in opposite direction. To turn around in the same grid, the robot uses the ultrasonic and IR sensor to determine which wall is closer. The mBot rotates 180° away from the wall it is closer to by using the twice the time taken to rotate 90°. For the two successive left and right turns, the mBot would turn once in the given direction first before moving forward one grid, followed by another turn in the same direction.

Since there is no sensor for the mBot to know its distance travelled or whether there is a wall in front of the mBot, this moving forward by one grid is done by moving forward for a fixed period before coming to a stop. After the turn, the array holding the RGB values from the LDR would be cleared in preparation for the next waypoint challenge and the mBot will continue moving forward. The code snippet for the calibration of the colours, the identification of colours and the turning of the mBot is shown in Figure 3.3.3.

Figure 3.3.3: Code snippet for turning mBot (01_motor_control.ino, lines 91-111).

```
void colour_turn(char colour) {
  if colour is red
    turn left
  if colour is green
    turn right
  if colour is orange
    turn 180 degrees away from the side that is nearer to a wall
  if colour is purple
    wide turn left
  if colour is blue
    wide turn right
}
```

Figure 3.3.4: Code snippet for solving waypoint challenge (mBot.ino, lines 160-167).

```
After black line is detected.
mBot stops moving
read rgb values
identify color using hue calculated from the rgb values
turn the mBot based on identified color
```

3.4. Reaching end of the maze

To finish the maze, the mBot had to be able to identify the white paper and play a celebratory tune after that. For the portion on detecting the white paper, the algorithm for detecting the colour of the paper as mentioned in the above section was used. Once the white paper is detected, "running" is set to false and LED0 is set to red and then it will play a song. We play a sequence of notes using the library function of the buzzer, which plays a specific frequency corresponding to each note for a set amount of time.

Figure 3.4.1: Code snippet for end of maze (mBot.ino, lines 169-175).

```
if identified color is white {
  change LED0 to display red
  set running to false
  play celebratory tune
}
```

4. Calibration of Custom-Built Sensors

4.1. IR Sensor

Due to the sensitive nature of the IR sensor, the values read in by the sensor can differ greatly even with a small change in distance. There was a need to ensure the robustness of the IR sensor and eliminate the influence of the ambient IR. When the ambient IR increases, the current I will increase, thus voltage drop across $R_{receiver}$ will increase, thus the voltage value measured will be lowered.

Figure 4.1.1: Circuit Diagram of IR sensor.

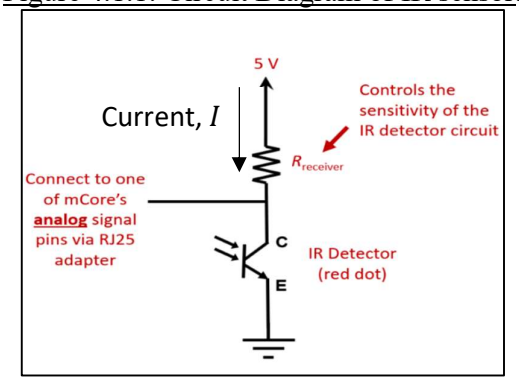
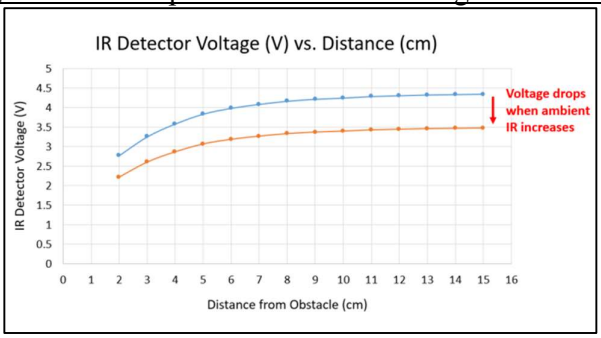


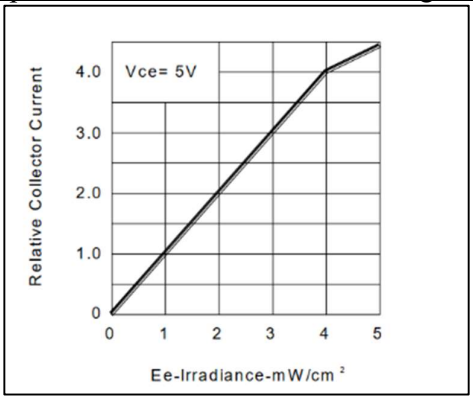
Figure 4.1.2 below illustrates the effect of ambient IR on the voltage readings for different distance from obstacle.

Figure 4.1.2: Graph of IR Detector Voltage vs Distance.



As we can see from the data sheet, as irradiance increases, the current increases linearly. As ambient IR increases, and current increases, the voltage drop across $R_{receiver}$ will be linear.

Figure 4.1.3: Graph of Relative Collector Current against Ee-irradiance.



To remove the interference from the ambient IR, we implemented the following logic shown in Figure 4.1.4. Firstly, we turn off the IR emitter and take the reading from the IR sensor. Let Baseline V_{IR} be the reading when IR emitter is off (Ambient IR only). Once the ambient IR is calculated, the IR emitter is then switched on. We then take a second reading from the IR sensor to obtain the reflected IR value. Let Reflected V_{IR} be the reading with IR emitter on. To account for ambient IR. We calculate

$$V_{Adjusted} = Reflected\, V_{IR} - \text{Baseline}\, V_{IR}$$

Since it is negative, we take the absolute value and compare it to a pre-determined value. As the distance between the mBot and the wall decreases, $Reflected\, V_{IR}$ decreases, thus $|V_{Adjusted}|$ increases. When the mBot gets too close to the wall, $|V_{Adjusted}|$ exceeds our threshold. We increase the right motor speed and decrease the left motor speed to help steer the mBot slightly to the left. This threshold value for the IR detector is measured when we placed the mBot 5 cm away from the right wall. Figure 4.1.4 shows the pseudo code for the IR sensor.

Figure 4.1.4: Pseudo code for getting IR reading (02_sensors.ino, lines 27-37).

```
int get_ir_distance_reading() {
  before IR emitter is turned on
  get ambient reading
  turn on IR emitter
  get actual reading
  turn off IR emitter
  return absolute value of ambient reading - actual reading
}
```

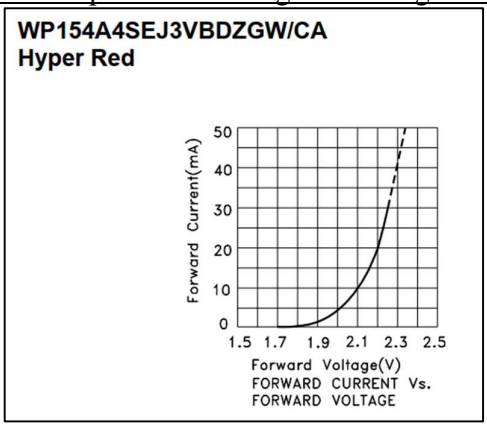
Figure 4.1.5: Code snippet for steering left (mBot.ino, lines 146-154).

```
if mBot is closer to the right wall {
  increase speed of right motor
  decrease speed of left motor
}
```

4.2. Colour Sensor

To keep below the 8mA output current limit of the chip, we limit the forward current of the LED to 5mA. Using the datasheet, we determined the forward voltage for a 5mA forward current and used it to calculate the resistor value.

Figure 4.2.1: Graph for current against voltage for red LED.



At 5mA, the forward current is 2V. Thus, the voltage drop across the biasing resistor is 5 – 2 = 3V. Resistor value required is 3V / 5mA = 600Ω.

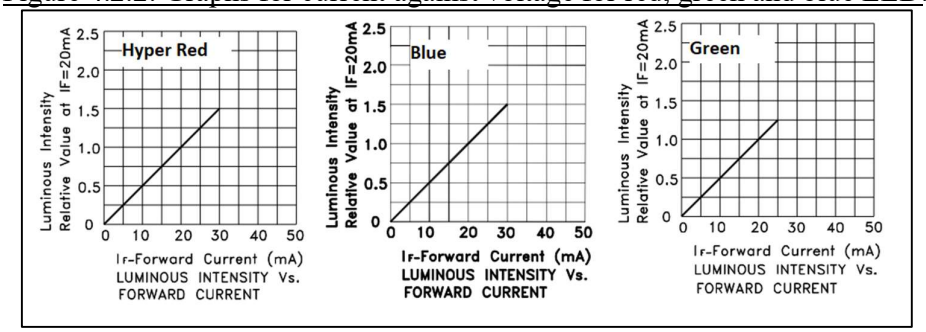
After obtaining the appropriate biasing resistors for each colour, we measured the current, they were all within the 8mA limit and are roughly the same.

Table 1: Initial values of resistor used and theoretical current drawn.

Colour	Biasing Resistor (Ω)	Current (mA)
Red	545	5.06
Blue	378	5.08
Green	378	5.18

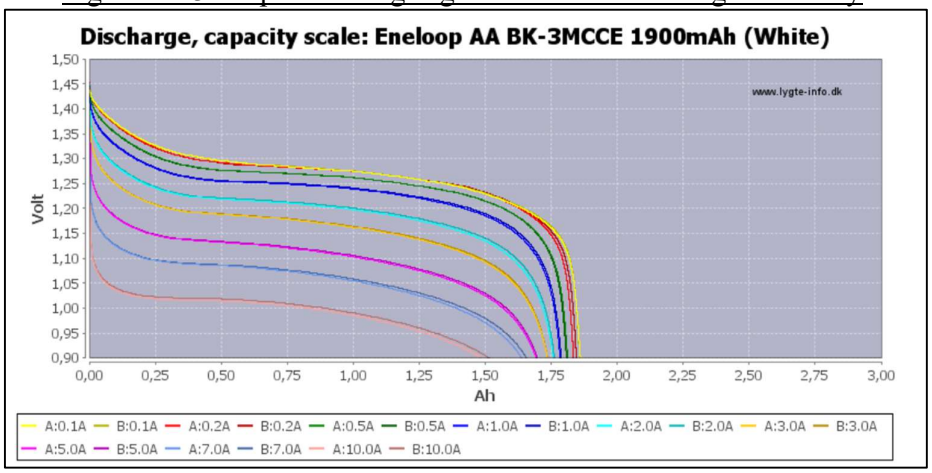
Based on the Luminous Intensity vs Forward Current graph for the three LEDs, we expect that as the current is roughly similar the intensity would be similar.

Figure 4.2.2: Graphs for current against voltage for red, green and blue LED.



However, when colour calibration is done, the values read in for when red LED is turned on is always inflated. As such, the biasing resistance the red LED was increased to 2.7k ohms to limit the current further, so that for white colour, the values read in for all three LEDs, are in similar ranges.

Figure 4.2.3 Graph of voltage against measure of charge in battery.



Since the battery voltage drops significantly when discharging it near full charge, it can lead to inconsistent turns and movement. This causes less power to be supplied to the DC motors, resulting in less mechanical power and slower wheel speeds. Hence, we calibrated our mBot turn timing and one cell move timing for the case when the batteries supply 5.24V, or 1.31V per cell, which is the point where the voltage of each cell is fairly stable even with changes in measure of charge in battery.

5. Work Division

Generally, our team got together often to work on various parts of the projects together so that we can make sure everybody has a clear understanding of every aspect of the project and everybody contributes to every aspect of the project. However, we do have assigned members to oversee various aspects of the project and be responsible for it. The four aspects are the code, report, wiring and testing.

Jin Xuan oversees documenting the code and cleaning up, making sure it is bug-free, readable and that no redundant work is done.

Jared oversees proofreading and formatting of the report, correcting any Grammar mistakes and word choices, making sure it flows logically, uses good English and is properly formatted for a reader-friendly experience.

Zhi Hong oversees the wiring and structure of the mBot, making sure components and wires are properly connected, so that no errors due to loose wiring or shorting of electrical component occurs.

Yong Chein oversees the testing of the mBot, ensuring that the code runs as intended, spotting any erroneous behaviour of our mBot, especially for edge cases so that we can correct it in the code promptly.

6. Challenges Faced

6.1. Calibration of speed of wheel to stay on track

To steer the mBot back on track, the motor speed was varied to produce a turning effect. Since the mBot would generally be in the centre with only some deviation, we reasoned that a small change in the speed should be sufficient to effectively correct the course of the mBot. However, the initial value that was added was too little to produce any visible change in the direction of the mBot. Unaware of our problem, we assumed that the problem lied with our IR sensor and tried to print the values being read in from the IR sensor. Seeing that the values were following a reasonable trend and that the IR sensor was working as intended, the problem could be pinpointed to the motors. The delay given to the motors was also important as the mBot would not be straightening itself during the delay. Hence, we varied the adjustment of the speed of motor several times before sticking to our current value which produces just enough turn to prevent the mBot from crashing into the wall and at the same time prevent it from over-correcting the turns and driving in a zigzag manner.

6.2. Preventing the IR reading from being affected by ambient IR

For our initial prototype, we read in the absolute value from the IR sensor without accounting for the ambient IR. As a result, when the mBot moved to different parts of the maze, there was a possibility that the ambient IR could fluctuate, causing the mBot to steer unnecessarily. This was a significant problem as the high sensitivity of the IR sensor meant that even a slight change in the ambient IR could cause the mBot to steer wrongly. To overcome this problem, we decided to adopt the idea of turning on and off the IR emitter to calculate a value that is affected only by the emitted IR reflected off the wall. Having this system significantly reduced the effects of the ambient IR.

6.3. Unresponsive colour sensor

For the initial prototype of the colour sensor, we used resistors that had too low resistance. This resulted in the LED being too bright, which caused voltage across LDR to be close to 5V when tested against paper of any colour due to the high intensity of light from the LED that was being reflected. We replaced the original resistors with resistors of higher resistance and the LED became dimmer, resulting in an increase in the range of values being read in by the mCore. However, the difference in RGB values between the white and black paper was still relatively small, hence the colour sensor was still unreliable.

Upon comparing our LDR to that of other groups, we realised that our LDR was slightly different and was less sensitive. The LDR was swapped with a new one, which allowed us to yield a wider range of readings with the same setup. We could now identify the paper with higher accuracy, but we decided that the system was not sufficiently robust and could be further improved. We felt that the ambient light was affecting our reading and we wrapped the LDR and LED individually with a black paper. This ensured that the reflected light incident on the LDR was mainly reflected from the colour paper and not the surrounding, hence effectively increasing the accuracy of our colour sensor.

However, there still times where the mBot detected the wrong colour. Though it did not occur frequently, we did not want to take the risk of the robot detecting the wrong colour, hence we decided to change to using the calculated hue values from the voltage values returned by the LDR. Based on whether the red, green, or blue light returned the largest value, a different formula is used to calculate the corresponding hue. Each coloured paper is then tested to gauge its hue value, and these values are then placed into a function to allow the mBot to identify the colour of the paper. If all three red, green, and blue values are above a threshold value, the

paper would be determined to be white. Since hue reflects the relationship between the individual RGB values, the calculated values differ distinctly for each different coloured paper and hence the colour sensor was much more reliable.

6.4. Not detecting the waypoint

In our loop, there was a chance that the robot might not detect the black line indicating that it had reached a way point and thus kept moving forward. There are two possible instances for this problem to occur. The first possibility happens when the robot crosses the black line during the delay function called for it to correct its direction. To solve the issue, we broke up the delay and intermittently checked whether a black line has been detected (mBot.ino, lines 140-144, 149-153). Hence with the speed of our bot and the short delays between checking for black line, there is insufficient time for the mBot to fully cross the black line and we would not overshoot the black line. The second possibility happens when the robot crosses the black line while it is measuring the average ambient IR. As the process of taking the average took too long, there was a chance that the mBot overshoots the black line and crashes into the wall. Since taking just a single reading proved to be sufficiently accurate and ensured that the mBot had enough time to sense the black line, we decided to take individual readings for the final design of the mBot.

7. Appendix A

7.1. Key Project Requirements

1. The mBot must not bump into any wall. Your mBot shall accomplish this with the help of one ultrasonic sensor on one side, and one infrared (IR) proximity sensor on the other side (no restriction on which of these two sensors to place on the left or right). You need to come up with your own algorithms to meet this requirement. Note that there will be penalty points for bumping into walls, even if your mBot doesn't get stuck.

2. When your mBot is not making a turn, it must travel as straight as possible. (It must not drive in a zig-zag manner like a car driven by a drunken driver.)

3. All turns in the maze are dictated by "waypoint challenges". Your mBot must not make any automatic turn without decoding a waypoint challenge.

4. When making a turn, your mBot must not over- or under-manoeuvre too much.

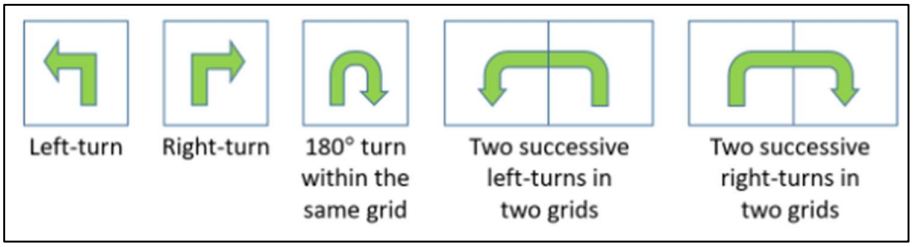
5. At each waypoint challenge, there will be a black strip (about 4 cm by 21 cm) on the maze floor. (Please refer to the video "Black Strip's Position.mp4" in LumiNUS for an illustration of how the black strip will be placed.) Your mBot needs to detect the black strip, stop, solve the waypoint challenge directly underneath it, and act according to the turn instruction decoded from the waypoint challenge.

6. Waypoint Challenge: Colour-sensing

(Sound-sensing challenge has been removed due to COVID-19 pandemic so as to reduce the amount of time students need to spend in the lab calibrating their setup.)

You need to build your own colour-sensing circuit on a mini-breadboard and place it underneath your mBot. At each waypoint challenge grid, besides the black strip, there will be a colour paper directly underneath your mBot. Depending on the colour of the paper, your mBot needs to execute one of the following five types of turns:

Figure 7.1: The five types of turns that your mBot needs to execute.



Note:

For the "two successive left-turns in two grids" and the "two successive right-turns in two grids", there will not be any black strip in the second grid to guide the mBot to execute the second turn. Your mBot needs to be hard coded to make these successive turns.

Table 2 summarizes how the colours are to be interpreted. Colour paper samples for each type of turn will be given to every project team.

Table 2: Colour Interpretation for the Colour-sensing Challenge.

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

7. End of Maze:

At the end of the maze, there will also be a black strip. The colour of the paper underneath the mBot at this grid will be **white**. Upon decoding that it has reached the end of the conquest, the mBot must **stop moving**, and **play a celebratory tune** of your choice (Yay!).