HCI - 01

Cao Shiqi, Zhu Wenxin

EE1111B

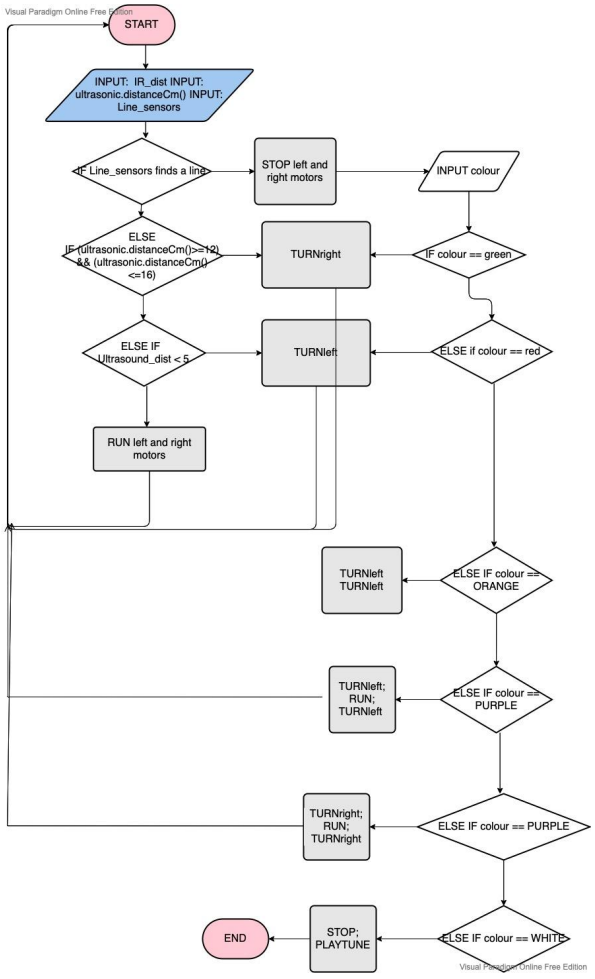
Final report

Table of Contents

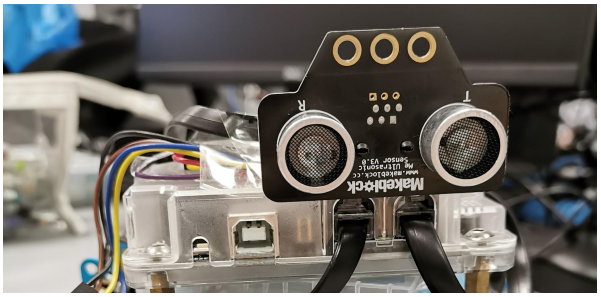
1. Algorithm Logic

Our overall algorithm attempts to keep the mBot in a straight line by turning it away from the walls of the maze when the walls are detected to be too close. For this, we made use of the ultrasonic sensor that was provided.

Using code from the mBot library (MeLineFollower lineFinder()), we were able to make it stop when a black line is sensed. The colour detection mechanism then tells the mBot which way to turn from the colour detected. The flowchart on the above displays the sequence of events in the loop.



2. Ultrasonic Sensors



2.1. Configuration

We had configured the ultrasonic sensor to be placed at the top of the mBot, to avoid it detecting the protruding edge of the table when there is an open wall and turning right to compensate for the long distance that it senses.

2.2. Algorithm

As we were unable to get our IR sensors working, we had to rely on the ultrasonic sensors as our only way to maintain a relatively straight path for our mBot. Therefore, we had written our code in a way that the mBot distinguishes when it is too far right, too far left, or when the reading from the ultrasonic sensor should be ignored as it is pointing at an open wall. To elaborate, we had considered the following scenarios.

- Scenario 1: Too close to the right wall
    - The ultrasonic sensor returns a reading that is smaller than 5 cm
    - mBot turns left for 10 ms to divert itself away from the wall

- Scenario 2: Too close to the left wall
    - The ultrasonic sensor returns a reading that is between 12 and 16 cm
    - mBot turns right for 10 ms to divert itself away from the wall

- Scenario 3: Open wall on the right side
    - The ultrasonic sensor returns a reading that is larger than 16 cm
    - mBot ignores the reading and continues to go straight

2.3. Calibration

To find out the distance that should make the mBot adjust its path, we simulated the previous 3 scenarios by pointing the mBot in different directions in the maze and gathering the distance the ultrasonic detector reads from the serial monitor. This process is repeated when the Mbot bumps into a wall or when we feel that the mBot is not adjusting itself at the correct time.

2.4. Difficulties

2.4.1. mBot turning to the right when there is an open wall

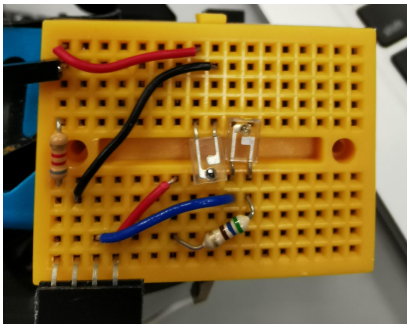
Originally, it was assumed that when the mBot is travelling from a closed to an open wall region, there is a segment where the mBot detects that it is too far from the right wall, so it turns right to adjust itself, resulting in an inaccurate turn. Therefore, to counteract this problem, the mBot was instructed to turn left when the ultrasonic sensor detects a very large distance.

However, this method is unreliable and sometimes the mBot overcompensates, making it too far left instead. We felt that our original assumption was wrong, so further testing was done and we realised that the ultrasonic sensor placed near the table would occasionally detect the protruding edge instead of an open wall, resulting in wrong path adjustments.

3. IR sensor

3.1. Configuration

Our IR sensor was placed on the left side of the mBot, opposite our ultrasonic sensor for it was intended to be supplementing the other sensor in keeping the mBot straight.



```
#include <MeMCore.h>
#define ADJUST_TIME_MS 10
#define IR A1
MeDCMotor leftMotor(M1); // assigning leftMotor to port M1
MeDCMotor rightMotor(M2); // assigning RightMotor to port M2
uint8_t motorSpeed = 255;
int longDis = -100;

void setup() {
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
  Serial.begin(9600); // Setup serial monitor for debugging purpose
}

void calibrateIR(){
  //digitalWrite(11, HIGH); digitalWrite(12, LOW); // turn off IR
 // ambient = analogRead(IR);
  digitalWrite(11, LOW); digitalWrite(12, LOW); // IR
  //IRreading=analogRead(IR)-ambient;
}

void loop() {
  // put your main code here, to run repeatedly:
calibrateIR();
if(IRreading>=longDis){ //copy code from IR sensors
  leftMotor.run(-motorSpeed);
  rightMotor.run(-motorSpeed); // Positive: wheel turns clockwise
  delay(ADJUST_TIME_MS); // Keep turning left for this time duration
    }
else{
  leftMotor.run(-motorSpeed);
  rightMotor.run(motorSpeed);
}
}
```

Our previously written code to check the functionality of the IR sensor

3.2. Algorithm

We had written our code in a way that the IR emitter would be turned off at first, to enable the IR detector to detect the surrounding ambient light. After that, the emitter would turn on, and the detector would measure the reading once again. The mBot would then take the difference between the two readings to be the true IR reading and react accordingly.

As the IR proximity sensor is less accurate than the ultrasonic sensor, we planned for it to only measure long distances and for the mBot to travel towards the left to correct for it. The measurements of the ultrasonic sensor would precede that for the IR sensor in the case of an open wall on the side with the IR sensor mounted.

3.3. Difficulties

3.3.1. Negative and unpredictable readings

When we first tried to run our testing code for the IR, we realised that the difference between the ambient lighting and the reading with the emitter turned on was negative. We just assumed that it is a natural occurrence and proceeded with the measured values for calibrating the distance at which the mBot should turn.

However, when we attempted to use the IR sensor, the mBot kept turning in circles. Upon further investigation, it was realised that the readings from our table and for different parts of the maze were all very different from each other, even if we had ensured that the sensor is the same distance from the obstacle.

We found out that the negative readings resulted from us not understanding how the IR emitter works and inserted it the wrong way, with the emitting side facing the breadboard instead of outwards. We could not find a definite answer for the unpredictable readings, yet we believe that it should be due to the same reason as the negative readings.

3.3.2. Little difference with the emitter turned on or off

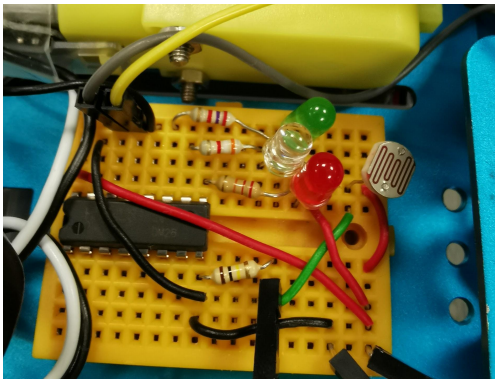
After we had corrected our previous mistake, the IR detector did return positive readings. However, we realised that the difference between the ambience and the reading with the emitter turned on was too small (ranging from -2 to 4) to be meaningful for determining the distance. We tried changing the resistor connected to the emitter to one with a smaller resistance, for we assumed that perhaps the current was too small for the emitter to turn on, yet this method yielded no results.

At this point, we believed that there might be some fault in the hardware itself, but we had run out of time to test the components.

Due to all of these difficulties stated above, we believed that if we continue and try to ensure that our IR sensor works correctly, it would take up too much time and affect our ability to improve the robustness of other subsystems. Therefore, our IR proximity sensor was abandoned and we worked on adapting our ultrasonic sensor instead.

4. Colour Sensor

4.1. Configuration



This is the final version of our colour sensing circuit that was glued to the bottom of the mBot.

The resistor values used for the LEDs were obtained by testing them with a constant 5V from AD2 and various resistors. Using two multimeters, we were able to obtain a set of voltage and current values to plot an V-I graph. The best fit curve equation was then keyed into a graphing calculator, enabling us to check for voltage for different currents.

4.2. Algorithm

4.2.1. Version 1 : Range of values

```
void SenseColours(){
  ReadValues();
   if( (120 < red ) && (red < 280) && (-5 < green) && (green < 10) && (275 < blue) && (blue < 480)){
    colour = 1;
    //Serial.println("orange");
  }
  else if((130 < red) && (red < 260) && (-30 < green) && (green < 5) && (310 < blue) && (blue < 470)){
    colour = 2;
   // Serial.println("red");
  }
  else if((-30 < red) && (red< 90) && (-5 < green) && (green < 20) && (530 < blue) && (blue < 720)){
    colour = 3;
   // Serial.println("blue");
  }
  else if((60 < red) && (red < 170) && (-20 < green) && (green <= 5) && (420 <= blue) && (blue < 550)){
    colour = 6;
 // Serial.println("green");
  }
  else if((30 <= red) && (red <= 140) && (-20 < green) && (green <= 10) && (460 < blue) && (blue < 680)){
    colour = 5;
   // Serial.println("purple");
  }
  else{
    colour = 4;
   // Serial.println("white");
  }
```

Our original version with the values used

At first, we relied on an algorithm that had a range of set values to determine the colour.

The values were obtained by placing the mBot in various places on the table that ran the final evaluation, detecting various colours and getting the range from a set of at least 20 readings. A headroom of ±20 was given and ambience reading was taken before the bulbs were turned on to reduce the effect of lighting conditions.

It performed relatively well on the test run. However, its inability to adapt to various lighting conditions meant that it required copious calibration to work.

4.2.2. Version 2 : Difference of values

Therefore, we changed our mind and created the current system, that made use of one standard reading of all six colours, then taking the one with the smallest difference from the obtained set of readings as the detected colour. This method meant that small disturbances due to ambient light are less likely to affect the colour detected.

As we ended up changing our black paper chimney at the very last moment, we had to obtain a new set of standard reading very quickly, which in addition to the crowding affecting the light, resulted in the mBot not being able to distinguish the red and the orange.

4.3. Difficulties

4.3.1. Errors in circuitry

We had some difficulties getting the bulbs to light up at first, and it turned out to just be us not understanding properly how the 2-to-4 decoder IC worked. There was also some error in our circuitry that made it very easy to short the circuit. This is due to the wrong placement of our LDR detector that resulted in a male-to-female wire used in detection protruding between the LEDs and the LDR. In the end, we redesigned the circuit by moving the LDR to the right instead of it being on the left of the LEDs, and moving the troublesome wire from before to the side and bending the tip so it lies flat.

5. Work Division

- Wenxin

  As she was unable to come for some lab sessions, she wrote the algorithm beforehand then handed the draft over to Shiqi to refine using collected data. Later, when we made drastic changes to the code, especially the colour sensing one, she was also the one who dealt with them. Wenxin also designed the circuit for the colour sensor and neaten all the circuits on board by cutting the wires. The groundwork for our report was also laid by her.

- Shiqi

  She mostly worked on the calibration, testing of components and refinement of the algorithm when Wenxin was absent. She also wrote the bulk of the report.