Microcontrollers and Interfacing Project

Final Milestone Report

Group Members

Afsah Hyder
Syed Muhammad Qamar Raza
Zaryan Ahmed Siddiqui

Supervisor

Ms. Areeba Rajput

Friday, December 8, 2023

Habib University

Contents

1 Introduction

For the final milestone, we have successfully integrated and implemented various sensors, achieving multifaceted functionality for our robot. The utilization of the IR sensor is dedicated to the task of line following, enabling the robot to autonomously navigate along predefined paths. The ultrasound sensor, on the other hand, serves the crucial purpose of obstacle detection, ensuring the robot can detect and respond to barriers in its environment. Additionally, we have incorporated a Bluetooth module for manual control of the robot, providing a versatile and user-friendly means of directing its movements. Furthermore, the robot's functionality was completed by adding the circumnavigation and serial monitor data display functionality where the obstacle count, the ultrasonic distance and the number of times the perpendicular line was detected were displayed on the serial monitor.

2 Sensors functionality and integration

In this section, we will discuss the functionality of different sensors and their integration.

2.1 Line Following using IR sensor

IR sensors emit infrared light, detecting reflected levels to identify black lines for line-following robots. Positioned on the robot's chassis, it measures surface reflectivity changes as the robot moves, recognizing the line. Continuous monitoring enables prompt adjustments if deviation is detected, ensuring the robot stays on track. In a closed-loop control system, IR sensors provide real-time feedback for rapid, autonomous line-following.

To implement this functionality, the system checks if either IR sensor ceases to detect the black line. If the robot veers off course on the right side, corrective action involves slightly moving it to the left. This is achieved by rotating the left wheel backward while the right wheel continues to move forward, causing the robot to turn left. This process iterates continuously to ensure that the robot consistently follows the line.

We also have an additional condition, if all the IR sensors detect black line, that is true for the perpendicular line, we keep moving forward.

The IR sensor attached to our robot is shown below. We are using the external two sensors for now, we'll be utilizing the internal two in later stages.

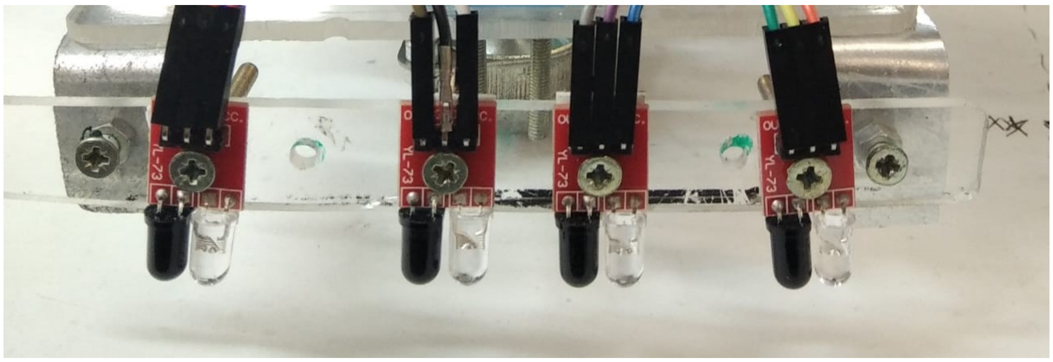

Figure 1: IR Sensor

The YL-70 IR sensor needs an average of 3.3-5V, and a current of less than 100 mA. So, the typical power consumption at 5V is 0.5 W. No specific communication protocol is being used.

2.2 Ultransonic Sensor

For obstacle detection, the HC-SR04 ultrasonic sensor is strategically positioned at the front of the robot. The robot immediately halts upon receiving ultrasonic sensor readings that fall below a predefined threshold. Polling is employed to continuously monitor for potential obstacles.

The ultrasonic sensor is only active during the automated line-following mode, with the option for manual obstacle avoidance in manual mode. The sensor consumes approximately 0.075W of power at 5V with a peak current requirement of 15mA. The sensor communicates distance readings to the controller via timed pulses, through its echo pin. The duration of the pulse is proportional to the distance of the robot from some obstacle in front of it.

The sensor placement is illustrated below. It is placed below the chassis for optimum space management.

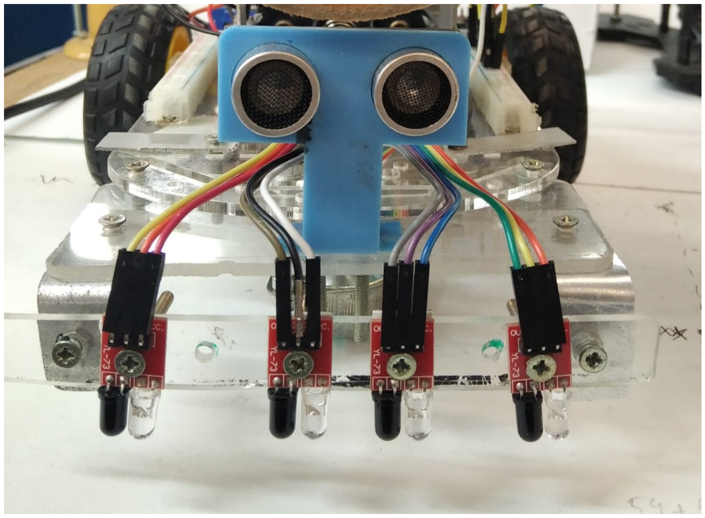

Figure 2: Ultrasonic and IR Sensor placement

2.3 Ball Dropping Mechanism

As explained and briefed earlier by the MCI Team, the robot is expected to have an automatic ball release mechanism but the ball could be loaded manually on to the robot. Keeping this in view, we have designed and implemented the ball release mechanism using a servo motor controlled by Arduino.

The servo arm was extended using the acrylic sheet and screws and was placed at angle cut into the tube. This ensures that the ball stays inside the tube unless the servo is lifted up to release it.

The servo motor operates at the end of the line-following part when all the IR sensors detect white line. It checks for two other conditions i.e if the obstacle count is equal or greater than 1 and an obstacle (the box) is detected.

The Servo motor (SG90) consumes about 4.8 to 6V, the current is typically about 100 mA, and the maximum can go to 500 mA. So, the typical power required is (5) (0.1) = 0.5W. It has a single-wire half-duplex asynchronous serial communication via PWM.

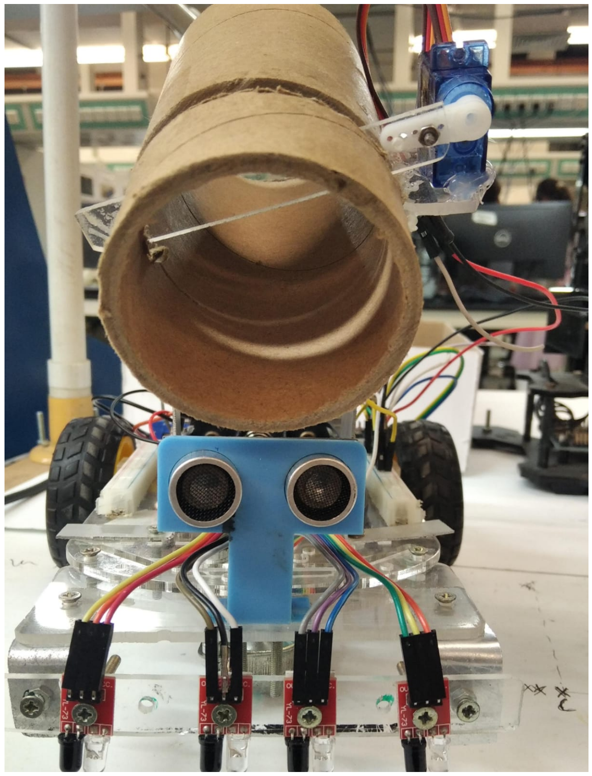

Figure 3: Servo mechanism

2.4 Bluetooth

The robot is maneuvered via a Bluetooth module, allowing control over various movements such as forward, backward, left, right, and diagonal. In manual mode, obstacle avoidance is seamlessly managed through the Bluetooth module, rendering the ultrasonic sensor unnecessary during this mode.

The HC-05 bluetooth module consumes around 3.3 to 6V, and a current of about 5mA. So, the overall power consumption is (5) (0.005) = 0.025W. The communication protcol being used is RS-232 implemented through UART, which is an asynchronous serial protocol that employs two wires for data transmission and reception.

The sensor placement is illustrated below. It is placed below the chassis for optimum space management.

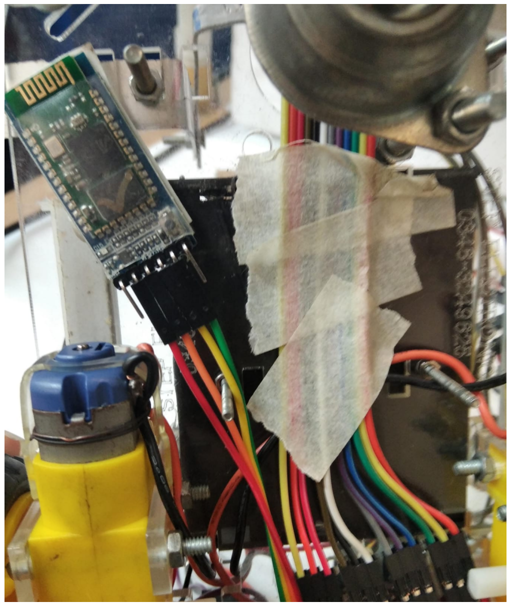

Figure 4: Bluetooth module placement

2.5 Circumnavigation

Circumnavigation involves a sequence of actions for our robot. Upon detecting an obstacle, the robot initiates a left turn until all infrared (IR) sensors register a white surface. The duration of this left turn is recorded in a variable. Subsequently, the robot travels in a straight line for a specific duration before executing a right turn for the same recorded time. Following the right turn, it moves forward until one of the IR sensors identifies a black line, at which point our line-following algorithm takes over.

2.6 Serial Monitor Output

The data about the obstacle count, the distance readings from the ultrasonic sensor and the detection of the perpendicular line is displayed using code on the Serial Monitor and the screenshot from the bluetooth terminal on phone is attached.

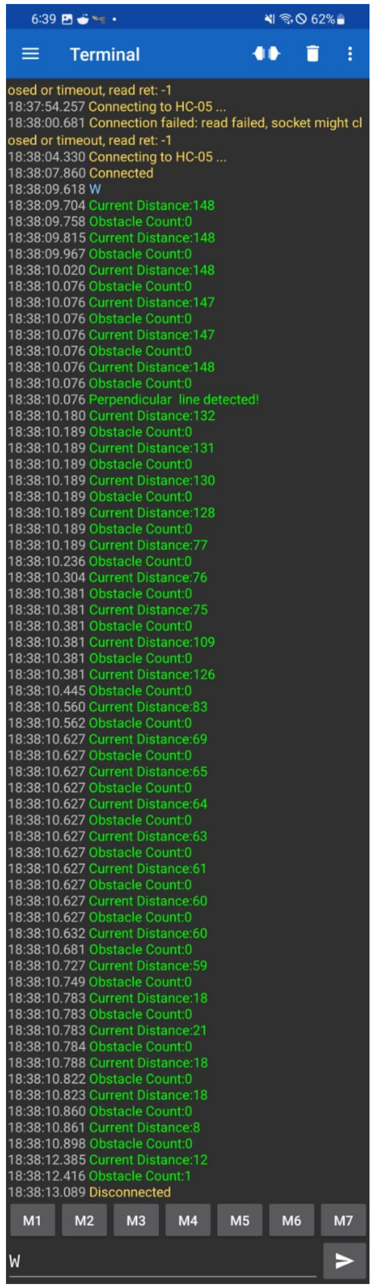

Figure 5: Serial Monitor Output

3 Sustainability

By working closely with Lab Research Assistants in the MCI Faculty and adopting a design thinking approach right from the start of the project, we have incorporated sustainable practices into our line-following robot project. Our core objective in taking this approach is to minimize costs wherever feasible, all the while taking into account the essential value derived from the expenditures.

A key element of our sustainable approach revolves around the practice of repurposing materials. We have made a conscious choice to utilize acrylic sheets and various other materials that were readily available from previous projects conducted at Habib University. This deliberate choice serves multiple purposes: it significantly reduces costs, aligns with our objective of minimizing waste within the Habib community, and encourages a more active and thoughtful engagement with the materials at hand.

For instance, as part of our commitment to sustainable engineering, we adopted innovative solutions like reusing a simple tissue roll in our ball dropping mechanism rather than acquiring new materials. Similarly, we leveraged an existing holder for the ultrasound sensor that was already present in the workshop from prior projects. This decision not only saved resources but also showcased the importance of utilizing what's available to us. Furthermore, we made a conscious effort to maximize the use of the robot's chassis, minimizing the need for additional acrylic. By doing so, we minimized waste and ensured that our project had a smaller environmental footprint. All these practices minimized the need for additional manufacturing and resource consumption.

In our line-following robot project, maintenance plays a pivotal role in ensuring sustainability. To facilitate easy maintenance and repairs, we've strategically positioned sensors and drivers for accessibility, streamlining troubleshooting and component replacement. This minimizes downtime and enhances maintenance efficiency. Additionally, we prioritize efficient space and resource utilization in our design. Our layout optimizes available space, reducing the need for extra materials like acrylic. By fully utilizing the chassis provided, we maintain a compact and resource-efficient robot design, minimizing unnecessary additions.

In summary, our line-following robot project embraces sustainability by reusing materials and prioritizing maintenance efficiency, minimizing both costs and environmental impact.

4 Task Distribution

The work was divided amongst the members as follows:

- **Afsah:** Circumnavigation, Bluetooth, Sensor Integration, Ball dropping
- **Zaryan:** Line following, Bluetooth, Sensor Integration, Ball dropping
- **Qamar:** Arena, Bluetooth, Sensor Integration, Ball dropping, Troubleshooting

5 Code Appendix

Below is the code we used for the Milestone IV

```
#include<Servo.h>
#define enA 5           //RIGHT
#define enB 6           //LEFT
#define IN1 7
#define IN2 9
#define IN3 13
#define IN4 12

#define ECHO A4
#define TRIG A5

#define SERVO_PWM 11

#define LS  3
#define RS  2
#define ML  4
#define MR  A3
```

```
enum Direction {FORWARD, BACKWARD};
int state_mode;
char command;
uint32_t dist = 0;

enum DriveModes {AUTO, MANUAL};

uint32_t obstacle_count=0;
bool armUp = false;
uint32_t time_elapsed;
uint32_t time_start;
uint32_t right_time_start;
uint32_t leftturn_duration;
uint32_t left_time_start;
uint32_t back_on_line;
uint32_t forward_clk;

DriveModes driveMode = MANUAL;
Servo servo;

void setup()
{
  Serial.begin(9600);

  // Initial direction of the robot is forward.
  ML_direction(FORWARD);
  MR_direction(FORWARD);

  // Starting time of the robot.
  time_start = millis();

  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
```

```

  pinMode(ECHO, INPUT);
  pinMode(TRIG, OUTPUT);
//  pinMode(SERVO_PWM, OUTPUT);
  servo.attach(SERVO_PWM);
  pinMode(LS,INPUT);
  pinMode(RS,INPUT);
  pinMode(MR,INPUT);
  servo.write(180);
}


void loop()
{
  command = receiveData();
  // Serial.println(command);
  //
  // bool RS_val= digitalRead(RS);
  // bool LS_val= digitalRead(LS);
  // bool MR_val= digitalRead(MR);

  // Serial.print("MR_val = ");
  // Serial.println((MR_val));
  // Serial.print("RS_val = ");
  // Serial.println((RS_val));
  // Serial.print("LS_val = ");
  // Serial.println((LS_val));

  // delay(500);
  if (command == 'W')
    driveMode = AUTO;
  if (command == 'w')
    driveMode = MANUAL;
//  Serial.println(command);
  // Serial.println(driveMode);

  if (driveMode == MANUAL)
    manualMode(command);
```

```
  else if (driveMode == AUTO)
    autoMode();
}

void MR_direction(Direction dir)
{
  if (dir == FORWARD)
  {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
  }
  else if (dir == BACKWARD)
  {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
  }
}

void ML_direction(Direction dir)
{
  if (dir == FORWARD)
  {
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
  }
  else if (dir == BACKWARD)
  {
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
  }
}

uint32_t check_distance()
{
  uint32_t duration;
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
```

```

  duration = pulseIn(ECHO, HIGH);

  uint32_t distance = 0.034 * duration * 0.5;

  return distance;
}

char receiveData()
{
  //Following Condition is true whenever we send a command from Bluetooth Terminal App
  char rxData;

  if (Serial.available()>0)
  {
    //Read the bluetooth data and store it in colorDetect variable using Serial.read() command here
    rxData = Serial.read();
  }

  return rxData;
}

void manualMode(char inpCommand)
{
  switch (inpCommand)
  {
    break;
    case 'F':
      analogWrite(enA, 114);//right
      analogWrite(enB, 115);//left
      ML_direction(FORWARD);
      MR_direction(FORWARD);
    break;
    case 'B':
      analogWrite(enA, 114);//right
```

```
      analogWrite(enB, 115);//left
      ML_direction(BACKWARD);
      MR_direction(BACKWARD);
    break;
    case 'L':
      analogWrite(enA, 114);//right
      analogWrite(enB, 115);//left
      ML_direction(BACKWARD);
      MR_direction(FORWARD);
    break;
    case 'R':
      analogWrite(enA, 114);//right
      analogWrite(enB, 115);//left
      ML_direction(FORWARD);
      MR_direction(BACKWARD);
    break;
    case 'G':             //forward left
      analogWrite(enA, 114);//right
      analogWrite(enB, 80);//left
      ML_direction(FORWARD);
      MR_direction(FORWARD);
    break;
    case 'I':           //forward right
      analogWrite(enA, 80);//right
      analogWrite(enB, 115);//left
      ML_direction(FORWARD);
      MR_direction(FORWARD);
    break;
    case 'H':           //back left
      analogWrite(enA, 115);//right
      analogWrite(enB, 80);//left
      ML_direction(BACKWARD);
      MR_direction(BACKWARD);
    break;
    case 'J':     //back right
      analogWrite(enA, 80);//right
      analogWrite(enB, 115);//left
      ML_direction(BACKWARD);
```

```
      MR_direction(BACKWARD);
    break;
    default:
      analogWrite(enA, 0);//right
      analogWrite(enB, 0);//left
    break;
  }
}

void turnSpeed()
{
  analogWrite(enA, 100);//right
  analogWrite(enB, 100);//left
}

void straightSpeed()
{
  analogWrite(enA, 114);//right
  analogWrite(enB, 115);//left
}

void halt()
{
  analogWrite(enA,0);
  analogWrite(enB,0);
}


void autoMode()
{
  //Read IR Sensors. if HIGH (BLACK Line) or LOW (
     WHITE Line).
  bool RS_val= digitalRead(RS);


  bool LS_val= digitalRead(LS);
  bool MR_val= digitalRead(MR);
  // Read Ultrasonic sensor.
```

```
uint32_t current_distance = check_distance();
Serial.print("Current Distance:");
Serial.println(current_distance);
Serial.print("Obstacle Count:");
Serial.println(obstacle_count);

// Line following.
if(current_distance < 15)
{
  if (current_distance<10){
    halt();
    if (obstacle_count == 0) // If distance is less than 10, start obstacle avoidance sequence.
    {
      obstacle_avoidance();
      obstacle_count++;
    }
  }

  else if (obstacle_count >= 1 && (RS_val==LOW && MR_val==LOW && LS_val==LOW))
  {
    halt();
    servo_motion();
    obstacle_count++;
    }
}
else if(RS_val == HIGH && LS_val == LOW)
{
  //Turn RIGHT
  turnSpeed();
  ML_direction(FORWARD);
  MR_direction(BACKWARD);
}

else if (RS_val == LOW && LS_val == HIGH)
{
```

```
    //Turn LEFT
    turnSpeed();
    ML_direction(BACKWARD);
    MR_direction(FORWARD);
  }

  else if ((RS_val==LOW && LS_val==LOW)) //for the perpendicular line that will be in the middle
  {
    straightSpeed();
    ML_direction(FORWARD);
    MR_direction(FORWARD);
  }
  else if (RS_val==HIGH && MR_val==HIGH && LS_val==HIGH)
  {
    straightSpeed();
    ML_direction(FORWARD);
    MR_direction(FORWARD);
    Serial.print("Perpendicular  line detected!\n");
  }
  // else if ((RS_val==LOW && LS_val==LOW) || ((RS_val==HIGH && MR_val==HIGH) && LS_val==HIGH)) //for the perpendicular line that will be in the middle
  // {
  //   straightSpeed();
  //   ML_direction(FORWARD);
  //   MR_direction(FORWARD);
  // }
}



void obstacle_avoidance()
{
  left_time_start=millis();
  //turn left
```

```
do
{
  turnSpeed();
  ML_direction(BACKWARD);
  MR_direction(FORWARD);
}
while(digitalRead(RS)==HIGH || digitalRead(LS)==
   HIGH || digitalRead(MR)==HIGH);    //until the
   obstacle is in path of the sensor, it keeps
   turning
delay(140);
leftturn_duration=millis()-left_time_start;

halt();
delay(100);

straightSpeed();
ML_direction(FORWARD);
MR_direction(FORWARD);
delay(800);
//turn right again to get back on line
right_time_start=millis();
do{
  turnSpeed();
  ML_direction(FORWARD);
  MR_direction(BACKWARD);
 }
while(millis()-right_time_start<leftturn_duration)
   ;

halt();
delay(100);

do
{
  ML_direction(FORWARD);
  MR_direction(FORWARD);
  straightSpeed();
```

```
  }while(digitalRead(RS)==LOW && digitalRead(LS)==
     LOW && digitalRead(MR)==LOW);

  // delay(1000);

  // delay(10);
  // back_on_line=millis();
  // do{
  //    analogWrite(enA,70);
  //    analogWrite(enB, 70);
  //    ML_direction(BACKWARD);
  //    MR_direction(FORWARD);
  // }
  // while(millis()-back_on_line<1000);     //this
     value needs to be foudnd through trial n test,
     to see how long it takes to get back on line
}


 void servo_motion(){
  servo.write(90);
  delay(1000);
  servo.write(180);
 }
 }
```

Listing 1: Arduino Code for Milestone IV

References

[1] https://www.electronicsforu.com/technology-trends/learn-electronics/ir-led-infrared-sensor-basics

[2] https://robocraze.com/blogs/post/what-is-ultrasonic-sensor

[3] https://einstronic.com/product/yl-70-infrared-line-tracking-sensor-module/

[4] https://www.sgbotic.com/index.php?dispatch=products.view&product_id=3028

[5] https://www.linkedin.com/pulse/introduction-serial-bus-servo-feetech-servo-abby: :text=The%20communication%20protocol%20used%20by,series%20on%20a%20serial%20bus.