Habib University

Dhanani School of Science and Engineering

Microcontrollers & Interfacing

EE 375L-T1/T2

Milestone 4 report

Team: Circuit Cruiser

Group Members:	**Student ID:**
Aamaina Mukarram Tahir Ali	08391
Abdullah Khalid	08428
Fatima Binte Aijaz	08519

Table of contents

Intro

Objectives: **Sensor interfacing and computational demo:** In this milestone, we furthered our sensor interfacing as the robot now used the integrated sensors implemented in Milestone 3 to follow a line, detect a wall and act accordingly, and drop a ball at point C in the arena. We have also worked on the arena, the ‘playground’ where we will be running our robot to test its various functionalities, including line following, obstacle avoidance, unloading. This report aims to explain some aspects of the functionality as implemented with arduino IDE.

Stance: Most of the work for the project is done now. Our robot is one with multiple functionalities. We are closer to the function we proposed in Milestone1.

Images of updated robot

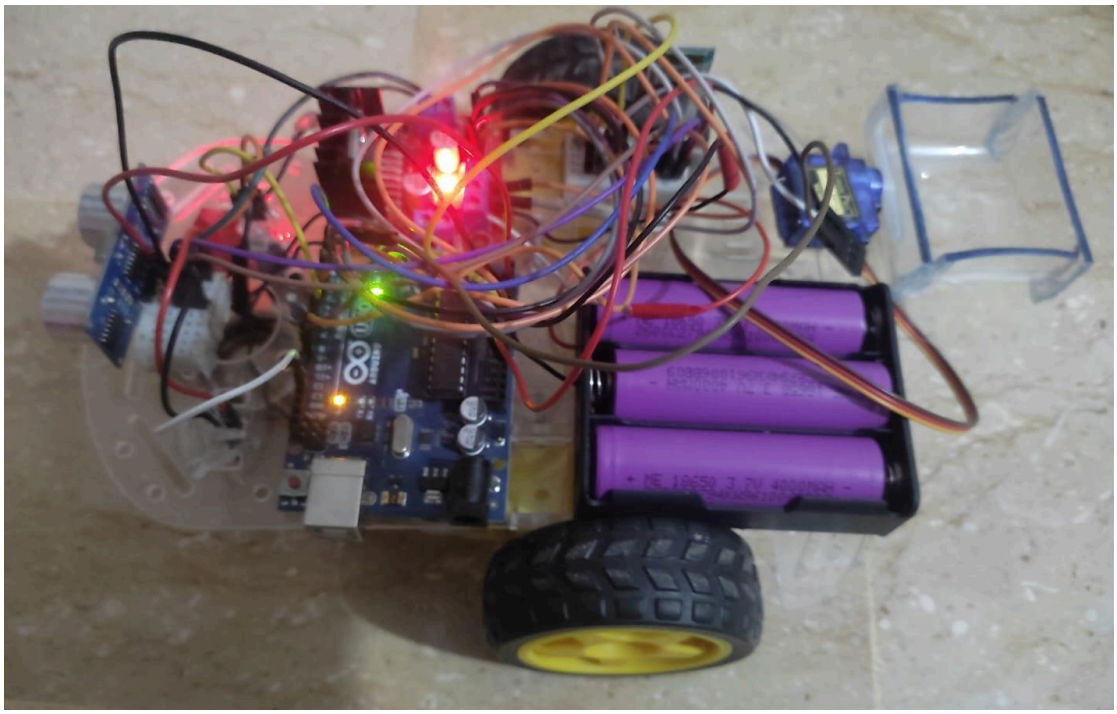

Figure 1: Robot top view

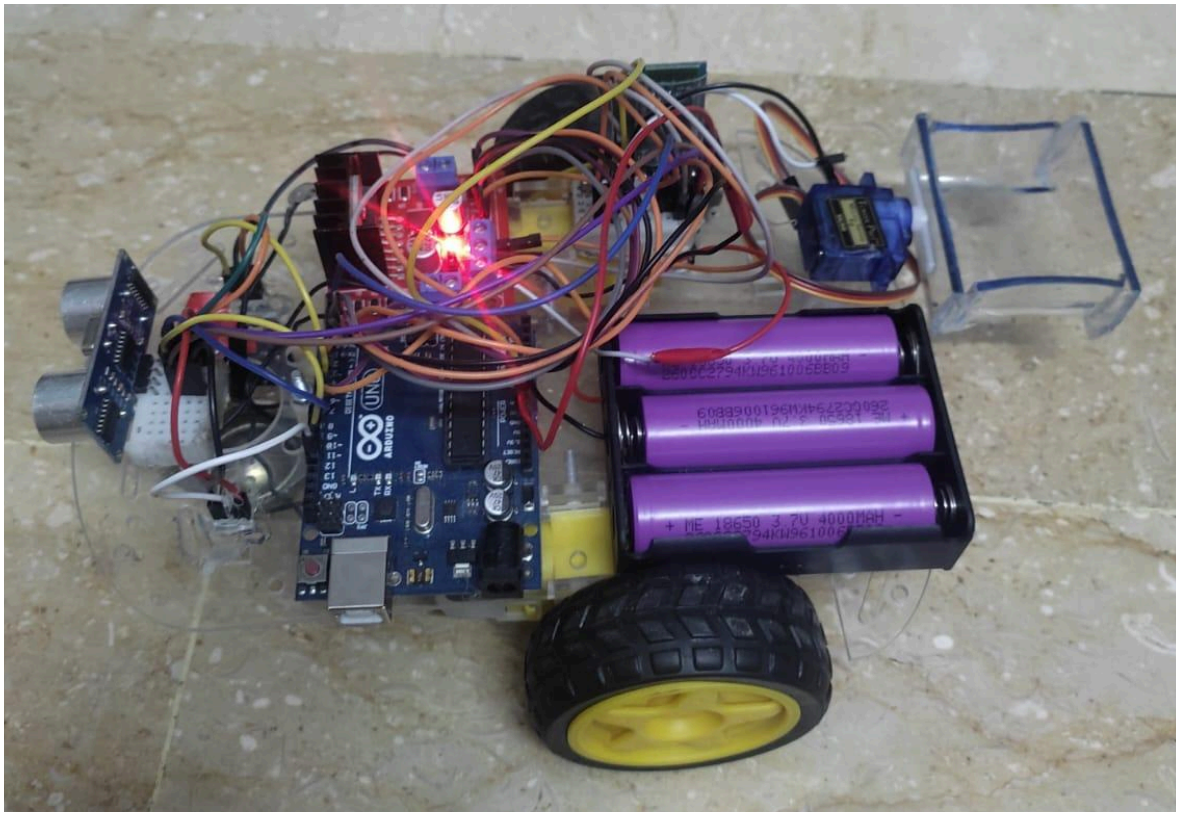

Figure 2: Robot side view

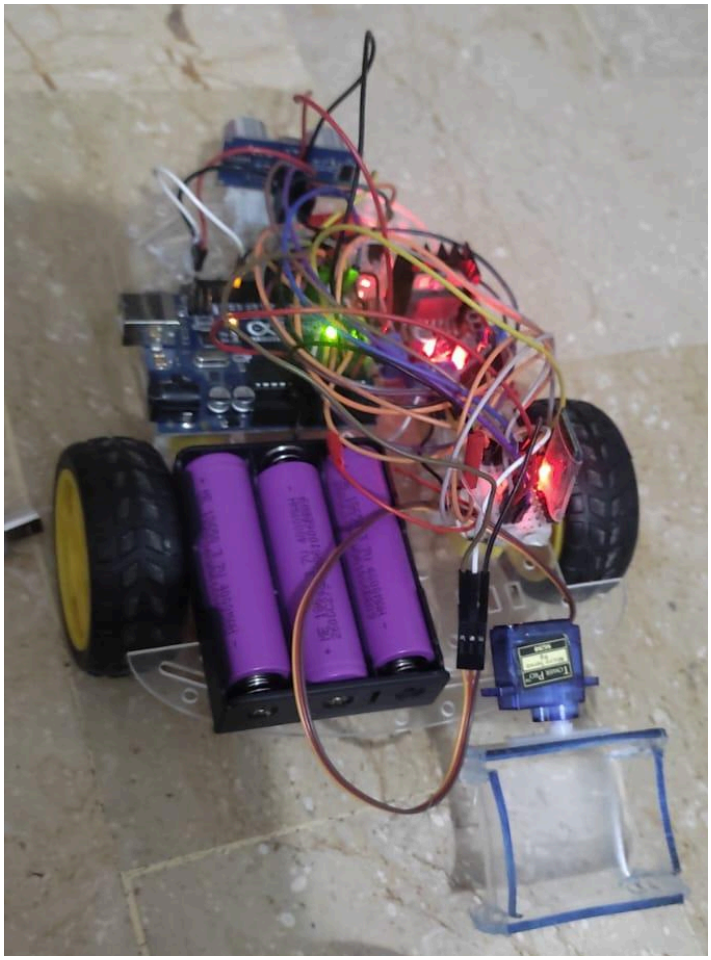

Figure 3: Robot front view

Explanation of functionality

Line following and corresponding action

In the previous milestone, we integrated an IR sensor into the bottom of our robot and made it functional. The IR emitters send out an infrared light which is reflected on the surface below. The receiver then detects the light. Different objects and colors have different capacities for absorbing IR radiation. As our arena has a white background and the line is with black, there's a clear distinction in the colors and so the difference between the absorption of infrared light enables the robot to correctly detect a line. In case of a ‘followline’ command in the main loop:

```
else if(Followline) // line following mode
{
 right = updateIR(IR1);
left = updateIR(IR2);
 LineFollower();
```

Variables left and right are updated, and a LineFollower function is called.

Updating the IR (and hence variables left and right) is as follows:

```
bool updateIR(const int PIN)
{
  int i;
  //if (PIN == 13){}
  for (i=0; i<5 ; i++)
  {
    if(digitalRead(PIN) == HIGH)
    {
      //Serial.println("OBJECT IN FRONT");
      if (PIN == 13){Serial.println("BLACK LINE on LEFT IR");}
      else{Serial.println("BLACK LINE on RIGHT IR");}
      return true;
    }
    delay(1) ; // take 5 accurate readings
  }
  return false;
}
```

We check if the corresponding IR pin is high 5 times. If true, it sets the value true. It also prints a message for debugging. Then,

```
void LineFollower()
{
  if(right)
  {
    rotate360(70,speed);
  }
  else if(left)
  {
    antirotate360(70,speed);
  }
  else
  {
    moveForward(0);
  }
}
```

The line follower function directs the robot to rotate in the corresponding directions based on the boolean values of ‘left’ and ‘right’. If the line is on the right, the robot moves right and if line’s on

the left the robot moves left, using rotate360 and antirotate360 respectively. (For implementation of movement functions please refer to the ‘All Codes’ section of this report).

Wall detection and corresponding action

We use an ultrasonic sensor to detect obstacles. Using print statements we can also see the distance between the robot and an object. In the ‘auto’ mode of our robot in the main function,

```
if(Auto) // detect collisions
{
 obstacle = is_Obstacle(40);
   Automovement();
   delay(100);
}
```

A variable ‘obstacle’ is updated and ‘AutoMovement’ is called.

```
void Automovement()
{
  if(obstacle)
   {
      rotate360(0,70);
   }
   else{
      moveForward(0);
   }
}
```

It (automovement) basically just rotates the robot incase of an obstacle (70 used to represent the time for which the robot rotates) otherwise does not affect the ongoing motion of the robot.

Unloading Mechanism (servo)

For unloading of an object (in this project a ball), we have attached a basket to the front of the robot.

Figure 4: Basket attached to servo.

As the object is placed in the basket, it drops down when the servo rotates. A rotation of 180 degrees is used as follows:

```
void Unload()

{
  myservo.write(360);              // tell servo to go to position in variable 'pos'
    delay(1000);
    myservo.write(360);              // tell servo to go to position in variable 'pos'
    delay(1000);

  int pos = 0;
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(150);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(150);                       // waits 15 ms for the servo to reach the position
  }
```

The servo rotates from 0 to 180 degrees, then from 180 degrees to 0 to its initial position. (This unload function is a case when input ‘U’ is given to bluetooth, which is in turn a case of the main function: command for handling movement).

Task division

We collectively worked on the project.

Milestone 4 summary

Progress: After a lot of hard work, our robot has now acquired more upgrades and now runs in the arena. It’s independent and has most of the functionality we aimed for.

Key achievements: Updated hardware functionality

Feedback and challenges: Working on the arena required a careful setting up of backgrounds. As all the sensors were already on the robot and functional, we just had to look into the coding aspect of the robot for its functions. Due to the countless wires on the robot, we still have to be extremely precautious about moving our robot; if it runs too fast, if it collides accidentally, etc.

Remarks: Seeing the robot finally run on the arena gave us a sense of achievement in the fact that we have a fully running robot that runs on a ground. From when the project guidelines were initially released and we saw videos of others’ projects, we now feel our project has also reached that level.

All code used

```
//#include<SoftwareSerial.h>
#include <Servo.h>
//==============TODO's=================
//Acceleration mechanism for better line following
//=====================================

// Operationmodes
bool Auto;
bool Followline = true;

// L298 pins
int motor1pin1 = 2;
int motor1pin2 = 3;

int motor2pin1 = 4;
int   motor2pin2 = 5;

//Ultrasonic
const int echoPin = A0;
const int trigPin = A1;

// HC_05
char Input; // will have values zero onewards
bool obstacle;

// IR sensor
const int IR1 = 7;
const int IR2 = 13; // pin 8 conflicting with echo pin of ultrasonic
bool left;
bool right;

int speed;
// int RX = 0;
// int TX = 1;
// SoftwareSerial bt(RX,TX);

//Servo
Servo myservo;
```

```
const int servopin = 11;

void setup() {
  Serial.begin(9600);
  init_L298N();
  initUltrasonic();
  InitIR();
  //Initservo();
  Auto = false ;
  Followline = false ;
  speed = 100;
  //L298N_SampleTest();
}

void loop() {
   //L298N_SampleTest();
   Receive_Bluetooth_Data();

    // Sensors

   // CHeck if Input is for automovement or manual movement
   if(Input == 'A'){Auto = true; Followline = false; }
   if(Input == 'S'){Auto = false; Followline = false; }
   if(Input == 'L'){Followline = true; Auto = false; }

  //Handle_movement() ;// for interupts

   if(Auto) // detect collisions
   {
    obstacle = is_Obstacle(40);
      Automovement();
      delay(100);
   }
   else if(Followline) // line following mode
   {
    right = updateIR(IR1);
   left = updateIR(IR2);
    LineFollower();
    delay(20) ;
   }
```

```
   else{ // manual movement mode
      Handle_movement();
      //moveForward(0);
   }
  delay(20) ;
}


//====================== Motor driver management ======================

// initialize motor
void init_L298N()
{
  pinMode(motor1pin1, OUTPUT);
  pinMode(motor1pin2, OUTPUT);
  pinMode(motor2pin1,   OUTPUT);
  pinMode(motor2pin2, OUTPUT);

  //(Optional)
  pinMode(9,   OUTPUT);
  pinMode(10, OUTPUT);

  // initialize motor speeds
  resetspeed(speed);
}

void resetspeed(int sp)
{
  int newspeed = sp;
  if (sp < 0)
  {
    newspeed = 0;
  }

  if (sp>250)
  {
    newspeed = 250;
  }

  analogWrite(9, newspeed); //ENA   pin
```

```
  analogWrite(10, newspeed); //ENB pin
}

// dry run motor
void L298N_SampleTest()
{

  digitalWrite(motor1pin1,   HIGH);
  digitalWrite(motor1pin2, LOW);

  digitalWrite(motor2pin1, HIGH);
   digitalWrite(motor2pin2, LOW);
  delay(3000);

  digitalWrite(motor1pin1,   LOW);
  digitalWrite(motor1pin2, HIGH);

  digitalWrite(motor2pin1, LOW);
   digitalWrite(motor2pin2, HIGH);
  delay(3000);

  // turn off both motors after that
  Stop_Motors();
}

//============ bluetooth management ==========================

void Receive_Bluetooth_Data()
{
  int data;
  //char[] string ;
  if(Serial.available()>0)
  {
    while(Serial.available()>0)
    {
      data = Serial.read();
      Input = char(data);
      Serial.print(Input);
      break;
    }
```

```
  }
  else {
  Input = '0';
  }
}

// ====================== movement ============================

void moveForward(int time) {
  resetspeed(speed);
  digitalWrite(motor1pin1, HIGH);
  digitalWrite(motor1pin2, LOW);
  digitalWrite(motor2pin1, HIGH);
  digitalWrite(motor2pin2, LOW);
  delay(time); // Adjust delay for desired duration of forward movement
  //Stop_Motors();
}

// Rotate 360 degrees
void rotate360(int time, int sp) {

  resetspeed(sp);
  // Implement rotation logic by running motors appropriately for
360-degree turn
  digitalWrite(motor1pin1, HIGH);
  digitalWrite(motor1pin2, LOW);
  digitalWrite(motor2pin1, LOW);
  digitalWrite(motor2pin2, HIGH);
  delay(time); // Adjust delay for desired duration of rotation
  //Stop_Motors();
}

// rotate anticlockwise
void antirotate360(int time,int sp)
{
  resetspeed(sp);
  digitalWrite(motor1pin1, LOW);
  digitalWrite(motor1pin2, HIGH);
  digitalWrite(motor2pin1, HIGH);
  digitalWrite(motor2pin2, LOW);
```

```
  delay(time);
 // delay(100); // Adjust delay for desired duration of rotation
}

// Rotate right by 45 degrees
void rotateRight45() {
  // Implement rotation logic by running motors appropriately for right
turn
  resetspeed(100);
  digitalWrite(motor1pin1, HIGH);
  digitalWrite(motor1pin2, LOW);
  digitalWrite(motor2pin1, LOW);
  digitalWrite(motor2pin2, HIGH);
  delay(200); // Adjust delay for desired duration of rotation
  Stop_Motors();
}

// Rotate left by 45 degrees
void rotateLeft45() {
  // Implement rotation logic by running motors appropriately for left
turn
    resetspeed(100);
  digitalWrite(motor1pin1, LOW);
  digitalWrite(motor1pin2, HIGH);
  digitalWrite(motor2pin1, HIGH);
  digitalWrite(motor2pin2, LOW);
  delay(200); // Adjust delay for desired duration of rotation
  Stop_Motors();
}

// istantly stop both motors
void Stop_Motors()
{
  digitalWrite(motor1pin1,   LOW);
  digitalWrite(motor1pin2, LOW);

  digitalWrite(motor2pin1, LOW);
   digitalWrite(motor2pin2, LOW);

    //speed = 50;
```

```
   Auto = false;
   Followline = false;
}
void Handle_movement()
{
switch (Input) {
    case 'F':
      moveForward(1000);
      Stop_Motors();
      //Accelerate(2);
      break;
    case 'R':
      rotate360(1000,80);
      break;
    case 'r':
      rotateRight45();
      break;
    case 'l':
      rotateLeft45();
      break;
    case 'S':
      Stop_Motors();
      break;
    case 'a':
      Accelerate(5);
      break;
    case 'd':
      Decelerate(5);
      break;
    case 'U':
      Unload();
      break;

    default:
      //Stop_Motors();
      //Serial.println("Decelaration");
      //Decelerate(5);
      //resetspeed(50);
      // No action for other inputs
      break;
```

```
}
}

//=======================Ultrasonic=============================

void initUltrasonic()
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

// take Inputs from ultrasonic and return a true if object found within
the desired distance in cm
bool is_Obstacle(long range)
{
  long duration;
  long cm;
  //Setting Trig Signal HIGH for 10us to produce burst of 8 pulses at
40KHz
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  //digitalRead(anypin/switch) //Use this function to read the state of
any Pin/Switch i.e. SW1 and SW2

  duration = pulseIn(echoPin, HIGH);   // Reads duration (microseconds)
for which Echo pin reads HIGH till wave is reflected
  cm = microsecondsToCentimeters(duration); // Handle Average Reading
mechanism Separately

  Serial.print("Distance=");
  //Serial.print("\t");
  Serial.print(cm);
  Serial.print("\n");

  if(range > cm){return true;}
  else {return false;}
```

```
}

long microsecondsToCentimeters(long microseconds)
{
long distance;
// The speed of sound is 340 m/s or 29 microseconds per centimeter.
distance = microseconds / 29 ;
return distance / 2;
}

void Automovement()
{
  if(obstacle)
   {
      rotate360(0,100);
   }
   else{
      moveForward(0);
   }
}

void LineFollower()
{
  if(right)
  {
    rotate360(70,speed);
  }
  else if(left)
  {
    antirotate360(70,speed);
  }
  else
  {
    moveForward(0);
  }
}
//======================IR sensor=================
void Accelerate(int acc)
```

```
{
  if (speed > 220)
  {
    return;
  }
  speed += acc;
  Serial.print("new speed is : ");Serial.print(speed);
}
void Decelerate(int dec)
{
  if (speed < 50)
  {
    return;
  }
  Serial.print("new speed is : ");Serial.print(speed);
  speed -= dec;
}
void InitIR()
{
  pinMode(IR1,INPUT);
  pinMode(IR2,INPUT);
}

bool updateIR(const int PIN)
{
  int i;
  //if (PIN == 13){}
  for (i=0; i<5 ; i++)
  {
    if(digitalRead(PIN) == HIGH)
    {
      //Serial.println("OBJECT IN FRONT");
      if (PIN == 13){Serial.println("BLACK LINE on LEFT IR");}
      else{Serial.println("BLACK LINE on RIGHT IR");}
      return true;
    }
    delay(1) ; // take 5 accurate readings
  }
  return false;
}
```

```
// Servo

void Initservo()
{
  myservo.attach(servopin);
  myservo.write(10);
  //delay(20);
}

void Unload()

{
  myservo.write(360);              // tell servo to go to position in
variable 'pos'
    delay(1000);
    myservo.write(360);              // tell servo to go to position in
variable 'pos'
    delay(1000);

  int pos = 0;
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180
degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in
variable 'pos'
    delay(150);                       // waits 15 ms for the servo to
reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0
degrees
    myservo.write(pos);              // tell servo to go to position in
variable 'pos'
    delay(150);                       // waits 15 ms for the servo to
reach the position
  }
```

References

Arduino IDE