Habib University

Dhanani School of Science and Engineering

Microcontrollers & Interfacing

EE 375L-T1/T2

Milestone 5 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: **Finalizing the robot and display of information:** In this final milestone, we upgraded the functionality we previously implemented, to provide some finalizing touches to our robot. As the hardware implementation and functionality was done already, we can add more information e.g. dropping count to our code and provide a more presentable robot. With slight changes, our RoboRace Royale Robot is now completely ready.

Stance: Involving presentation and more interaction with the arena, our robot now runs in the ground.

Images of updated robot

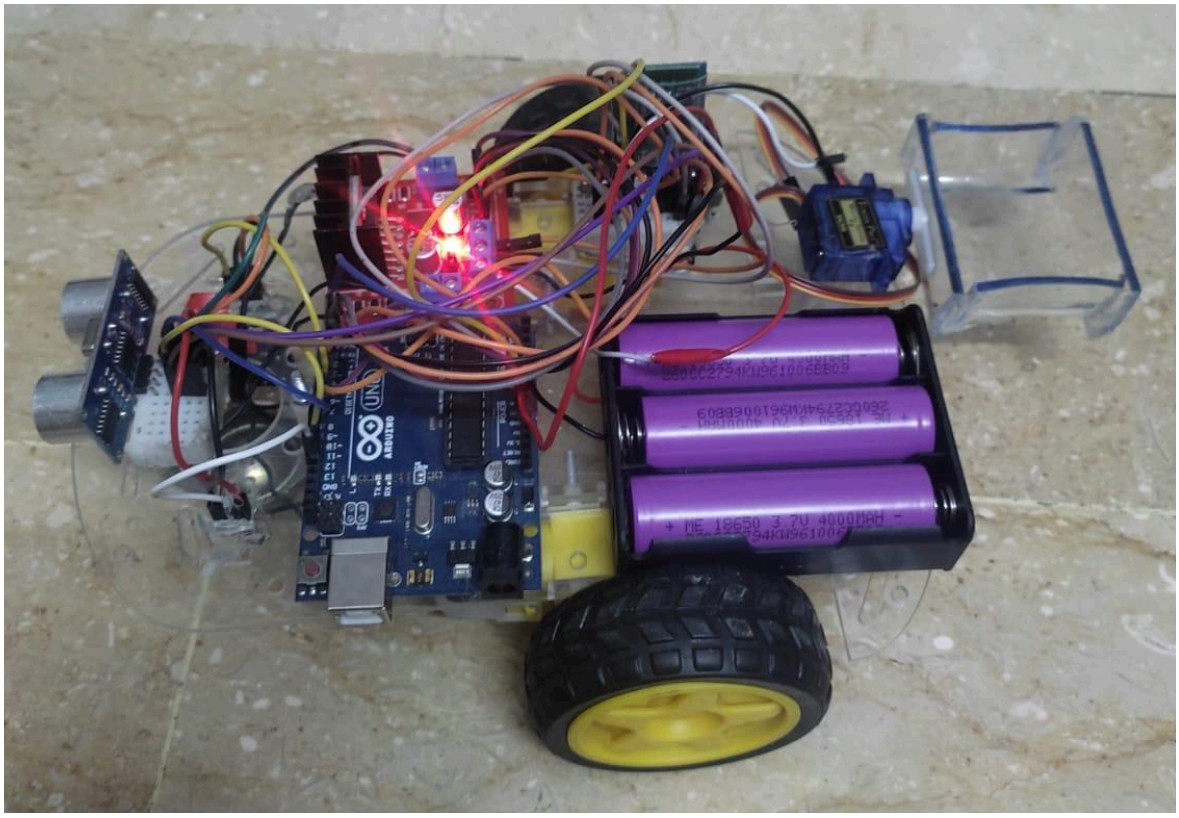


Figure 1: Robot side view

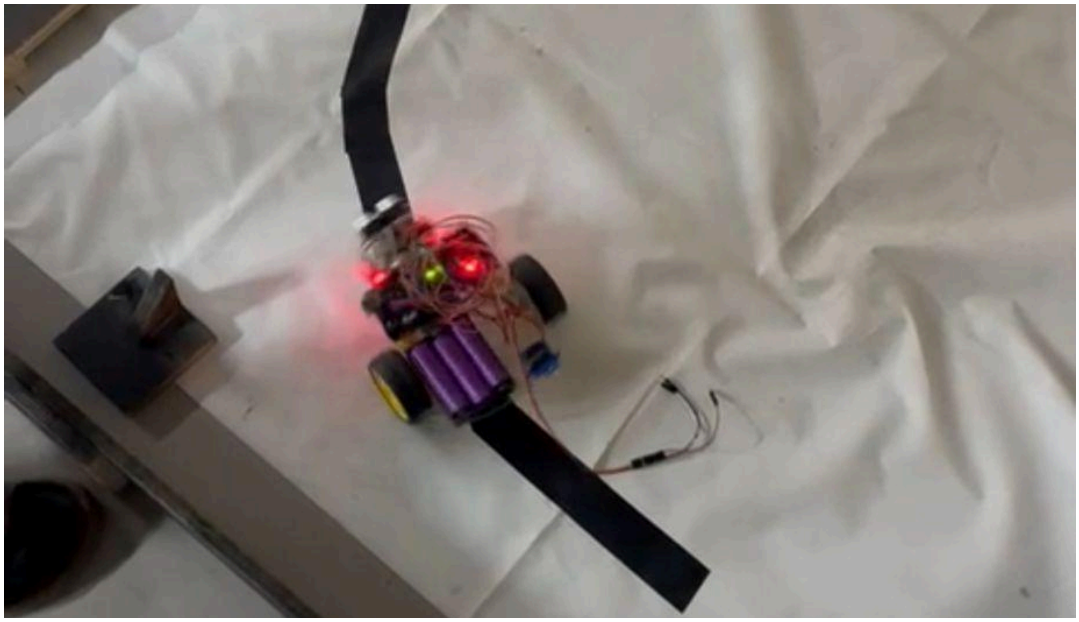


Figure 2: Robot in action in *arena*

Explanation of functionality

Perpendicular line detection

In case of a perpendicular line in the arena, the robot moves towards it. When requested with command ‘P’, it will also display the information. The following function can be referred to, in order to understand the functionality:

```
void DetectPerpendicularLines() {
  bool perpendicularDetected = false;

  bool leftLine = updateIR(IR1); // Check left IR sensor for line
  bool rightLine = updateIR(IR2); // Check right IR sensor for line

  if (leftLine && rightLine) {   //if on both sides,
    perpendicularDetected = true; //there's a perpendicular line
  }

  if (perpendicularDetected) {
      moveForward(1000); // Move forward
      Serial.println("Proceeding forward")

  } else {
    Serial.println("No Perpendicular line")
    Stop_Motors(); // Stop
  }
}
```

(Note: Distance calculation in rubric: Our code already prints the distance detected between the sensor and an object, hence there was no need to update the code for this part).

Dropping count, obstacle count

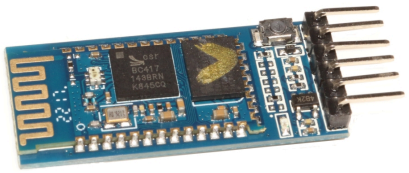
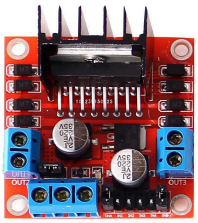
Using a servo, we implemented a ball dropping mechanism. Everytime the servo rotated and rotated back, we knew that an object was dropped. So we can add and display a counter in that part of the code to show the number of times the mechanism was activated:

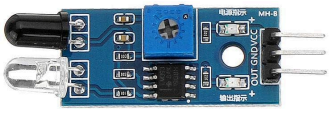
```
void Unload()
int unloadCount=0;
{
  unloadCount++; //increment the count everytime this function is trigerred
  myservo.write(360);              // tell servo to go to position in variable 'pos'
    delay(1000);
    myservo.write(360);              // tell servo to go to position in variable 'pos'
    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
    Serial.print("Ball dropped successfully."); //after executing the function, print the message.
    Serial.print("Number of balls dropped: ");
    Serial.println(unloadCount);
  }
}
```

Component selection justification

As each sensor or basic electronic was integrated throughout the project, earlier reports explained their functionality and implementation. Each component we used in the robot was a requirement; please refer to the table below for the justification.

Component	Picture	Use
Arduino		The main microcontroller used to drive the robot. Arduino provides an interface between our code and the hardware functioning of the robot. It communicates with other components attached to the robot, like servo motor, ultrasonic sensor etc and brings them into integration with the robot.
Lithium batteries		To Power the robot (Vcc)
HC-05		Input interface and driver. The signals we send to the robot on which it acts in accordance are through bluetooth. HC-05 provides us a way to communicate with the robot wirelessly; the robot executes functions based on our choice.
DC motors		Mobility: Two DC motors are attached on either side of the robot chassis. With the rotational motion it provides, we attach wheels to the motor which then provides the robot with basic motion like moving forward, moving in rotation, etc.
L298N		To integrate DC motors into our robot, we use L298N (interface provider), which can control the motion of two DC motors at a time.

Servo motor		The rotation that the servo motor provides is used to meet the ball dropping requirement of the project. As we attached the head of the servo to a drawer, it unloads an object using that.
Ultrasonic sensor		This sensor is used to measure distance from the robot to a nearby object, and hence implement the obstacle avoidance mechanism.
IR sensor		With its ability to distinguish colors, this sensor is used to implement line following mechanism in the robot.
breadboard		A mini breadboard is required for maintaining connections between the components of the robot via jumper wires.

Task division

Abdullah worked on the hardware and testing part of the robot, Amena mainly contributed more into making video for submission, Fatima worked on the code and the report.

Milestone 5 summary

Progress: Our robot is a professional project now.

Key achievement: Besides updating some things in the robot, a functional video of our robot running in the arena and performing its wanted tasks is submitted in this milestone. This is equivalent to the presentation.

Feedback and challenges: Although the line following of the robot was tested at home with a simple line, the robot made some mistakes following the line in the arena; in the end, however, it did follow it correctly. A lot of care needs to be taken in order to carry and run the robot everytime, since the jumper wires may move a little and cause disconnection in important parts.

Remarks: This marks the end of our project, and we have built it in small steps throughout.

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;
    case 'P':
      DetectPerpendicularLines();
      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 DetectPerpendicularLines(){
// bool perpendicularDetected = false;
// bool leftLine = updateIR(IR1); // Check left IR sensor for line
// bool rightLine = updateIR(IR2); // Check right IR sensor for line
// if (leftLine && rightLine) { //if present on both the sensors,
perpendicular line is detected
//     perpendicularDetected = true;
//   }
//   if (perpendicularDetected) {

//     if (leftLine && rightLine) {
//       moveForward(1000); // Move forward for 1 second (adjust time as
needed)
```

```
//     }
//     else {
//     Stop_Motors(); // Stop
//   }
//   }
// }

void DetectPerpendicularLines() {
  bool perpendicularDetected = false;

  bool leftLine = updateIR(IR1); // Check left IR sensor for line
  bool rightLine = updateIR(IR2); // Check right IR sensor for line

  if (leftLine && rightLine) {   //if on both sides,
    perpendicularDetected = true; //there's a perpendicular line
  }

  if (perpendicularDetected) {
      moveForward(1000); // Move forward
      Serial.println("Proceeding forward")

  } else {
    Serial.println("No Perpendicular line")
    Stop_Motors(); // Stop
  }
}

void Unload()
int unloadCount=0;
{
  unloadCount++; //increment the count everytime this function is
trigerred
  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
    Serial.print("Ball dropped successfully."); //after executing the
function, print the message.
    Serial.print("Number of balls dropped: ");
    Serial.println(unloadCount);
  }
}
```

References

(Pictures)

Sciencestore.pk
Majju.pk
components101.com
Amazon
electrobes.com
Embeddedstudio.com