DHANANI
SCHOOL OF SCIENCE
AND ENGINEERING

Microcontrollers & Interfacing

Spring 2025

EE/CE 376L/332L

T1

Yousuf Aijaz

Sufiyan Aman

Asad Siddique

Taqi Shah

Table of Contents

Introduction:

The wall-following robot is designed to autonomously navigate a rectangular arena by adjusting its movement based on the distance from both the left and right walls. It uses ultrasonic sensors to detect nearby obstacles and follows the wall while avoiding collisions. When a front wall is detected, the robot performs a 90-degree left turn to stay on track. A Tiva C board handles sensor input and motor control, using a straightforward decision-making algorithm to ensure efficient and systematic navigation**.**

System Overview:

The robot is driven by a Tiva C microcontroller and controlled via an L298N motor driver. It operates based on a predefined logic to detect obstacles and navigate accordingly, executing precise turns to maneuver efficiently within a rectangular environment.

Hardware Components:

- **Microcontroller:** Tiva C **TM4C123GH6PM**
- **Motor Driver:** L298N
- **Communication:** Bluetooth Module HC-05
- **Motors:** 2 x DC motors
- **Power Supply:** 3 x 3.7V batteries
- **Sensors:**
    - Three Ultrasonic sensors (Front, left and right)

Sensor Integration and Functionality:

Ultrasonic Sensor (Left): Primarily responsible for monitoring the robot's distance from the left wall. It plays a dual role:

- When the robot is **too close or too far from the left wall**, it adjusts its position to maintain a consistent gap and ensure straight-line movement.

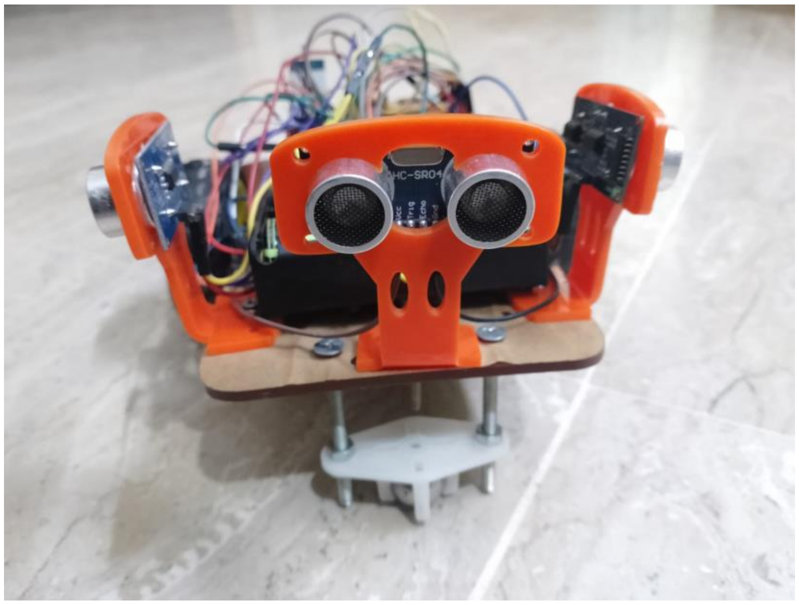
- If the **left wall suddenly disappears** (i.e., the distance exceeds a certain threshold), the robot identifies this as an opportunity to turn and executes a **90-degree left turn** to continue following the new wall.

Ultrasonic Sensor (Right): Helps maintain balance and alignment by detecting the distance to the right wall. If the robot drifts too close to the right side, it adjusts slightly to the left to re-center itself in the lane.

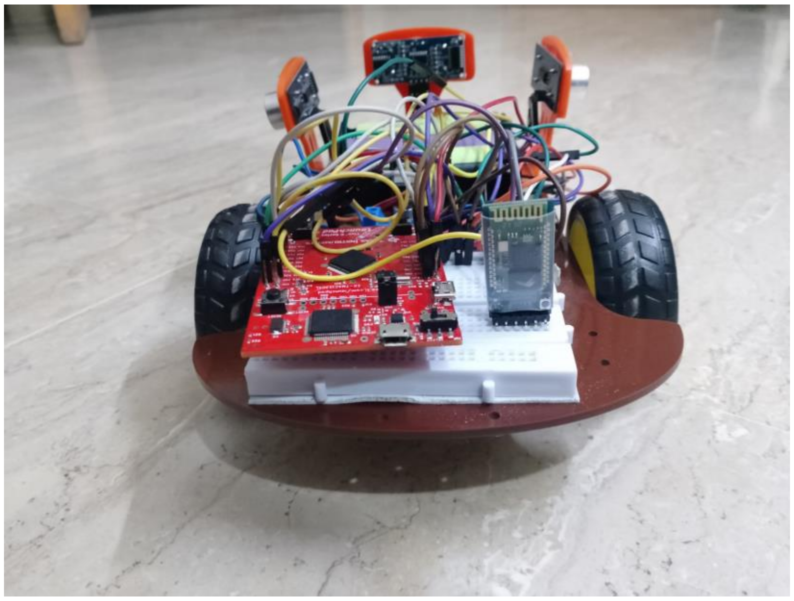
Ultrasonic Sensor (Front): Used to detect obstacles directly in front of the robot. If a wall is detected within a close range and both side paths are blocked, the robot interprets it as the end of the path and **comes to a complete stop**.

Robot Pictures:

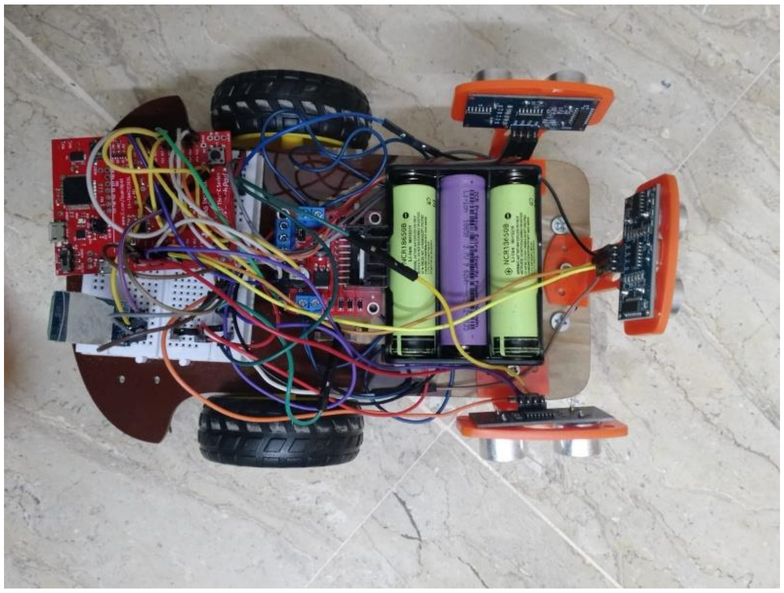
Front:



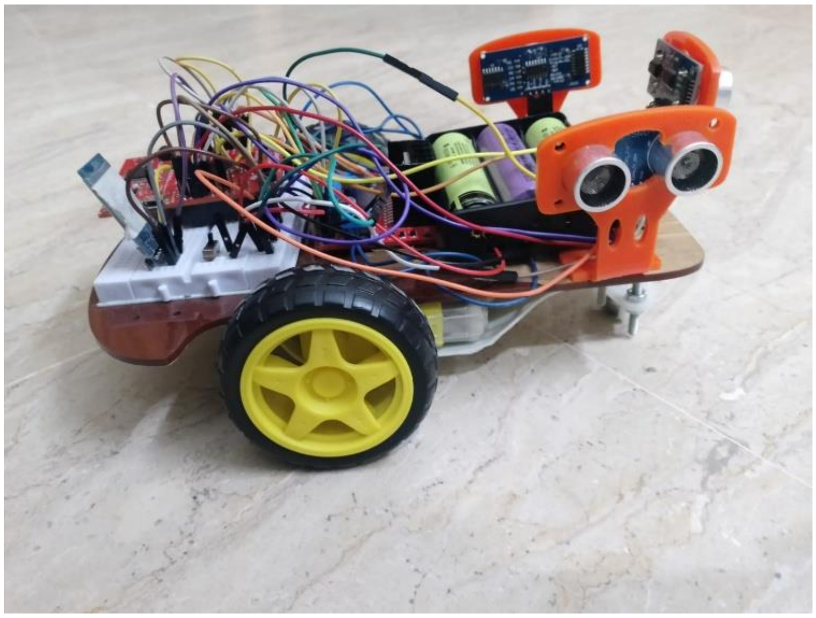
Back:



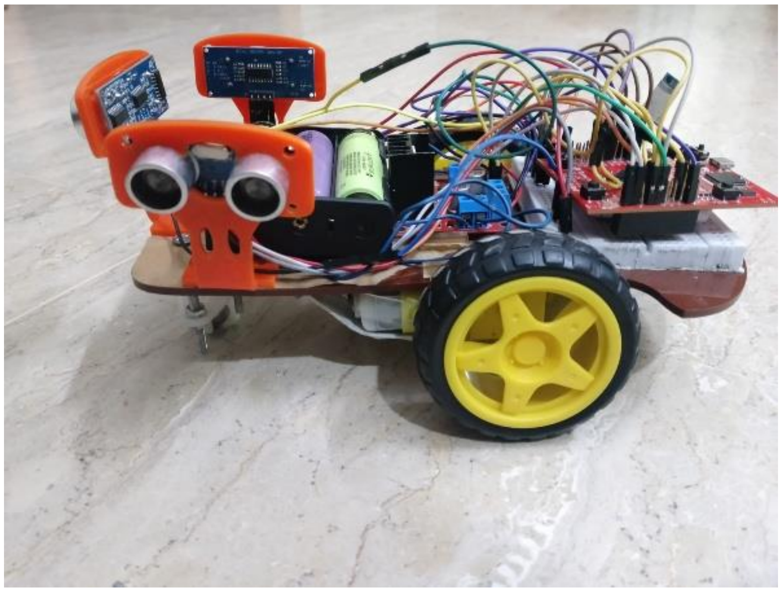
Aerial:



Right:



Left:



GitHub Link:

https://github.com/Yousuf7788/MCI_Project_sp25.git

Code:

```
const int LEFT_PWM  = 31;  // PC5 (M0PWM0) - Left Motor Speed
const int RIGHT_PWM = 36;  // PC4 (M0PWM1) - Right Motor Speed
const int LEFT_IN1  = 11;  // PA2 - Left Motor Direction 1
const int LEFT_IN2  = 25;  // PD2 - Left Motor Direction 2
const int RIGHT_IN3 = 37;  // PC4 - Right Motor Direction 1
const int RIGHT_IN4 = 38;  // PB3 - Right Motor Direction 2

const int trigFront = 7, echoFront = 27;  // Front Sensor
const int trigLeft = 32, echoLeft = 12;    // Left Sensor
const int trigRight = 3, echoRight = 4;  // Right Sensor

long cmFront, cmLeft, cmRight;
int stop_distance = 20;    // Stop when front wall is within 20 cm
int wall_distance = 9;
int turn_speed = 160;       // Speed for minor adjustments
int correction_speed = 120; // Speed for adjusting to maintain wall gap

char bluetoothCmd = 'S';  // Default to stopped
bool isRunning = false;  // Flag to control movement

void setup() {
  Serial.begin(9600);
  Serial3.begin(9600);
  pinMode(LEFT_IN1, OUTPUT);
  pinMode(LEFT_IN2, OUTPUT);
  pinMode(RIGHT_IN3, OUTPUT);
  pinMode(RIGHT_IN4, OUTPUT);
  pinMode(LEFT_PWM, OUTPUT);
  pinMode(RIGHT_PWM, OUTPUT);

  pinMode(trigFront, OUTPUT);
  pinMode(echoFront, INPUT);
  pinMode(trigLeft, OUTPUT);
  pinMode(echoLeft, INPUT);
  pinMode(trigRight, OUTPUT);
  pinMode(echoRight, INPUT);
}

void loop() {
  cmFront = getDistance(trigFront, echoFront);
  cmLeft = getDistance(trigLeft, echoLeft);
  cmRight = getDistance(trigRight, echoRight);
  Serial3.print("cmFront: ");
  Serial3.print(cmFront);
  Serial3.print("\n");
  Serial3.print("cmLeft: ");
  Serial3.print(cmLeft);
  Serial3.print("\n");
  Serial3.print("cmRight: ");
  Serial3.print(cmRight);
  Serial3.print("\n");
if (Serial3.available() > 0) {
    bluetoothCmd = Serial3.read();
    if (bluetoothCmd == 'G' || bluetoothCmd == 'g') {
      isRunning = true;
      Serial3.println("Received: G (Go)");
    } else if (bluetoothCmd == 'S' || bluetoothCmd == 's') {
        isRunning = false;
        Stop_Motors();
        Serial3.println("Received: S (Stop)");
    }
}
if (isRunning==true){

if (cmFront<20 && cmLeft<15 && cmRight<15){  // Stop at the end
  Stop_Motors();
}
else if (cmLeft>30) {       //Turn left if no wall on left
  delay(100);
  Stop_Motors();
delay(500);
  Move_Left();


}
// Adjust right if too close to left wall
else
if (cmLeft > 0 && cmLeft < cmRight) {
    //Serial3.println("Too close to left wall, adjusting right.");
    Adjust_Right();
}

// Adjust left if too close to right wall
else if (cmRight > 0 && cmRight < cmLeft) {
    //Serial3.println("Too close to right wall, adjusting left.");
    Adjust_Left();
}
// Move forward normally
else {
        Move_Forward();
    }

    delay(5);  // Small delay for sensor stabilization
    }
}

void Move_Right() {
    digitalWrite(LEFT_IN1, HIGH);
    digitalWrite(LEFT_IN2, LOW);
    digitalWrite(RIGHT_IN3, HIGH);
    digitalWrite(RIGHT_IN4, LOW);
    analogWrite(LEFT_PWM, 144);
    analogWrite(RIGHT_PWM, 120);
    delay(600);
    Stop_Motors();
    delay(200);
    Move_Forward();
    delay(300);



}
void Adjust_Right() {
    digitalWrite(LEFT_IN1, HIGH);
    digitalWrite(LEFT_IN2, LOW);
    digitalWrite(RIGHT_IN3, LOW);
    digitalWrite(RIGHT_IN4, HIGH);
    analogWrite(LEFT_PWM, 192);
    analogWrite(RIGHT_PWM, correction_speed);
}
```

```
// Move Slightly Left (Correct right wall proximity)
void Adjust_Left() {
  digitalWrite(LEFT_IN1, HIGH);
  digitalWrite(LEFT_IN2, LOW);
  digitalWrite(RIGHT_IN3, LOW);
  digitalWrite(RIGHT_IN4, HIGH);
  analogWrite(LEFT_PWM, 144);
  analogWrite(RIGHT_PWM, turn_speed);
}
// Move Left (Turn left to avoid obstacle)
void Move_Left() {
  digitalWrite(LEFT_IN1, LOW);
  digitalWrite(LEFT_IN2, HIGH);
  digitalWrite(RIGHT_IN3, LOW);
  digitalWrite(RIGHT_IN4, HIGH);
  analogWrite(LEFT_PWM, 144);
  analogWrite(RIGHT_PWM, 120); // Slow down right wheel for turning
  delay(600);  // Adjust experimentally for a 90° turn
  Stop_Motors();
  delay(500);
  Move_Forward();
  delay(300);
}
// Move Forward
void Move_Forward() {
  digitalWrite(LEFT_IN1, HIGH);
  digitalWrite(LEFT_IN2, LOW);
  digitalWrite(RIGHT_IN3, LOW);
  digitalWrite(RIGHT_IN4, HIGH);
  analogWrite(LEFT_PWM, 130);
  analogWrite(RIGHT_PWM, 125);
}

// Stop Motors
void Stop_Motors() {
  analogWrite(LEFT_PWM, 0);
  analogWrite(RIGHT_PWM, 0);
}
// Convert Time to Distance
long microsecondsToCentimeters(long microseconds) {
  return (microseconds * 0.0343) / 2;  // More accurate conversion factor
}

// Get Distance from Ultrasonic Sensor
long getDistance(int trigPin, int echoPin) {
  long total = 0, count = 0;

    digitalWrite(trigPin, LOW);
    delay(2);
    digitalWrite(trigPin, HIGH);
    delay(10);
    digitalWrite(trigPin, LOW);

    long duration = pulseIn(echoPin, HIGH, 10000); // 20ms timeout
    long distance = microsecondsToCentimeters(duration);

    if (distance > 1) { // Ignore noise (less than 1 cm is unrealistic)
      total = distance;

    }
   return total;
}
```

Explanation of Code:

Robot Behavior Based on Sensor Input

1. **Obstacle Detection in Front (End Condition):**
   If the front sensor detects a wall closer than 20 cm, and both the left and right walls are less than 15 cm away, the robot stops by calling Stop_Motors(). This indicates a likely end of a corridor or path.
2. **Turning Left When No Left Wall:**
   If the left distance exceeds 30 cm, the robot assumes an opening on the left and executes a left turn using Move_Left(). This movement includes a short stop before and after the turn to stabilize the motion.
3. **Wall Following - Adjusting Right:**
   If the robot is closer to the left wall than the right (i.e., cmLeft < cmRight), it slightly turns right by calling Adjust_Right(). This helps maintain a balanced distance from the left wall.
4. **Wall Following - Adjusting Left:**
   If the robot is closer to the right wall than the left (i.e., cmRight < cmLeft), it adjusts slightly left using Adjust_Left() to correct its path.
5. **Normal Forward Motion:**
   If none of the above conditions are met, the robot moves forward by calling Move_Forward().
6. **Bluetooth Communication:**
   The robot receives commands over UART (Serial3) from a Bluetooth module (e.g., HC-05).
   - Sending 'G' or 'g' starts the robot.
   - Sending 'S' or 's' stops the robot and calls Stop_Motors().
   -

Motor Control Functions

- **Move_Right():**
  Turns the robot 90 degrees to the right by reversing the direction of the right motor and running the left motor forward. Includes timed delays for accurate rotation and resumes forward motion after the turn.
- **Move_Left():**
  Turns the robot left by reversing the direction of the left motor and running the right motor forward. Like Move_Right(), this includes timed delays and resumes forward motion after the turn.
- **Adjust_Right():**
  Performs a slight right adjustment by increasing the speed of the left motor and slowing down the right motor.
- **Adjust_Left():**
  Slightly turns the robot left by increasing the speed of the right motor and reducing the speed of the left motor.
- **Move_Forward():**
  Moves the robot forward at moderate speeds using both motors.
- **Stop_Motors():**
  Stops the robot by setting the motor speeds to zero.

Ultrasonic Sensor Functions

- **microsecondsToCentimeters():**
  Converts the time measured by the ultrasonic sensor into distance in centimeters using the speed of sound.
- **getDistance():**
  Sends a pulse from the trigger pin and measures the time until the echo returns. Uses a timeout to prevent blocking if no object is detected. Returns the calculated distance, ignoring unrealistic values under 1 cm.

Task Distribution:

- **Hardware and integration:** Muhammad Asad Siddique
- **Programming and documentation**: Yousuf Aijaz
- **Programming and navigation logic:** Sufiyan Aman
- **Arena setup and navigation testing:** Taqi Shah