CSE461

Project Report

Lab Section: 05

Group: 06

Line Following Cleaner Robot

Member Details:

1. Name: MD. FARDEEN ISLAM ID: 22301076
2. Name: MD.TOSLIM UDDIN RAHAT ID: 22301235
3. Name: ASHIQUE RAHMAN SAHIL ID: 22301055

Submission Date: 07/01/2025

Instructors:

1. Dr Md Khalilur Rhaman

2. Rafid Ahnaf

BRAC University

1. Project Title: Line Following, vacuum, and mopping robot.

2. Purpose

Objective: To clean the line that the robot follows

Scope: Cleaning paths for regular use

Significance: Cleaning efficiently, saving time, scheduled cleaning

3. Components

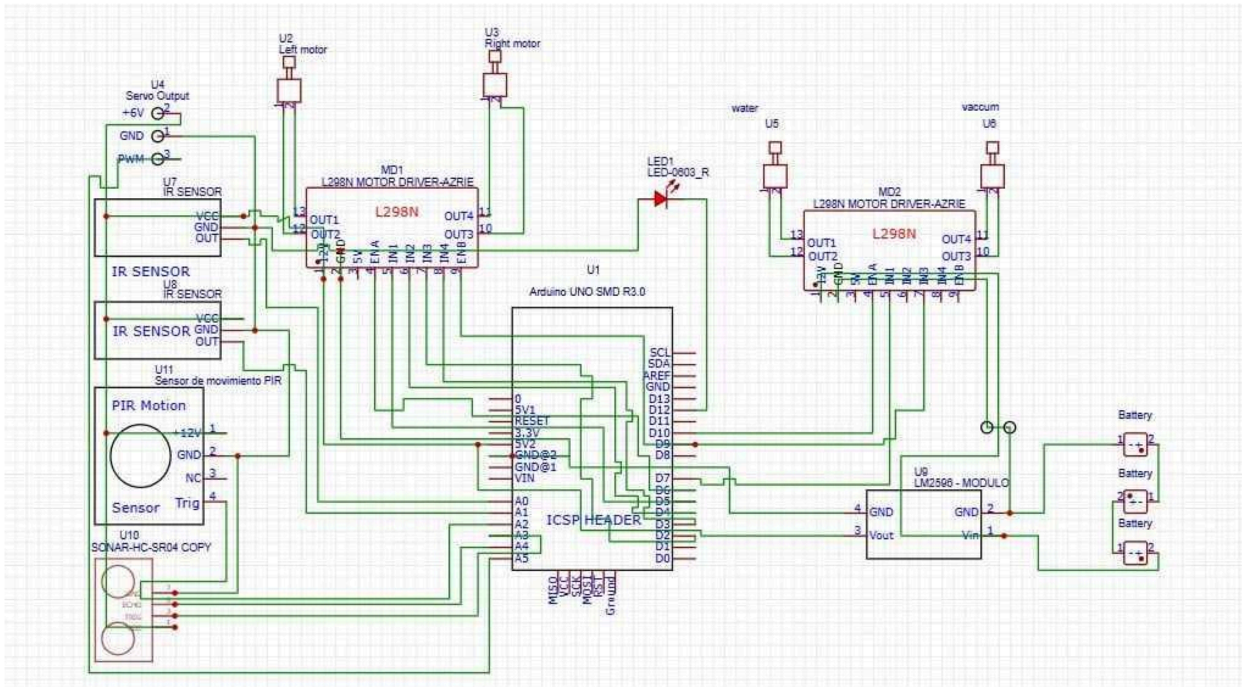
- Microcontroller: Arduino Uno R3
- Sensors:
  - Ultrasonic Sonar Sensor (HC-SR04)
  - Infrared Sensor (IR Sensor)
  - HC-SR501 PIR Motion Sensor

- Actuators:
  - DC 6V Micro Pump Motor

  - BO Motor
  - Servo Motor MG996R

- Body/Chassis: 2WD Wheel Drive Mobile Robot Platform Chassis
- Additional Components: L298N Motor Driver, Breadboard, Jumper wires, Pipe, Battery, Battery holder, LED Red, Arduino chip, Battery charging board

4. Diagram/Circuit Setup:



5. Cost Breakdown

No	Components	Quantity	Unit Cost (BDT)	Total Cost (BDT)
1	Arduino Uno R3	1	1050	1050
2	Breadboard Half Size Bare 400 Tie Points	1	88	88
3	Jumper Wires 20 pcs 20 cm	3	55	165
4	2WD Wheel Drive Mobile Robot Platform Chassis	1	650	650
5	IR Obstacle Sensor	2	70	140

6	LED Red	1	2	2
7	Sonar Sensor HC SR04	1	105	105
8	L298N Motor Driver	2	199	398
9	DC 6V Micro Pump Motor	2	173	346
10	Pipe for Pump Motor	2	20	40
11	PIR Motion Sensor Module	1	116	116
12	14500 Rechargeable Lithium Battery 1200mah 3.7v	5	80	400
13	Servo Motor MG996R	1	445	445
14	3S 10A 18650 Lithium Battery Charging Board, BMS Protection Module 12.6V	1	200	200
15	ATmega328P-PU PDIP-28 Microcontroller	1	550	550
16	Battery Holder	1	100	100
17	XL7015 5V-80V DC-DC 0.8A Dc converter Step-down Module Wide Voltage Input LM2596	1	180	180
Total Cost (BDT)				4325 (excluding delivery fees)

6. Functionality Breakdown

- **Functionality 1:**
  - **Overview:** Follows a black line to travel
  - **Working Procedure:** Two IR Sensors are used to detect the black line by sensing the reflected infrared ray. Black surfaces absorb more light, resulting in low sensor reading. If both sensors detect the line, the robot moves forward. If the left sensor detects the line, the robot moves left. If the right sensor detects the line, the robot moves right. BO motors are controlled by motor driver pins (IN1, IN2, ENA, IN3, IN4, ENB). HIGH and LOW signals are sent to the motor driver to move the motors accordingly.
- **Functionality 2:**
  - **Overview:** Vacuums, Sprays water and Mops the line as it moves
  - **Working Procedure:** Two pump motors (one for vacuum and one for spraying water), and the brush servo are controlled by motor driver pins (VACUUM_RUN, WATER_RUN, VACUUM_WATER_ENA). The brush servo motor moves from 0 to 180 degrees to mimic the sweeping movement. When the sensors detect black line, the vacuum, water pump and brush activates by enableWaterVacuumBrush() function. If the robot stops after motion or object detection, the components deactivate by disableWaterVacuumBrush() function.
- **Functionality 3:**
  - **Overview:** Detects obstacles and humans
  - **Working Procedure:** Sonar sensor (TRIG_PIN and ECHO_PIN) emits ultrasonic waves to measure the distance between robot and obstacles. If the object is within threshold distance (<=20 cm), the robot stops along with the vacuum, water pump and brush and turns on the LED light (MOTION_LED). The PIR Sensor (MOTION_SENSOR) detects human motion by sensing infrared radiation changes. If the PIR sensor detects motion, the robot stops as well and turns on the LED light. Once the obstacle is removed or motion stops, the robot starts to move again after a debounce time (MOTION_CLEAR_TIME or clearDuration).

7. Business Proposal

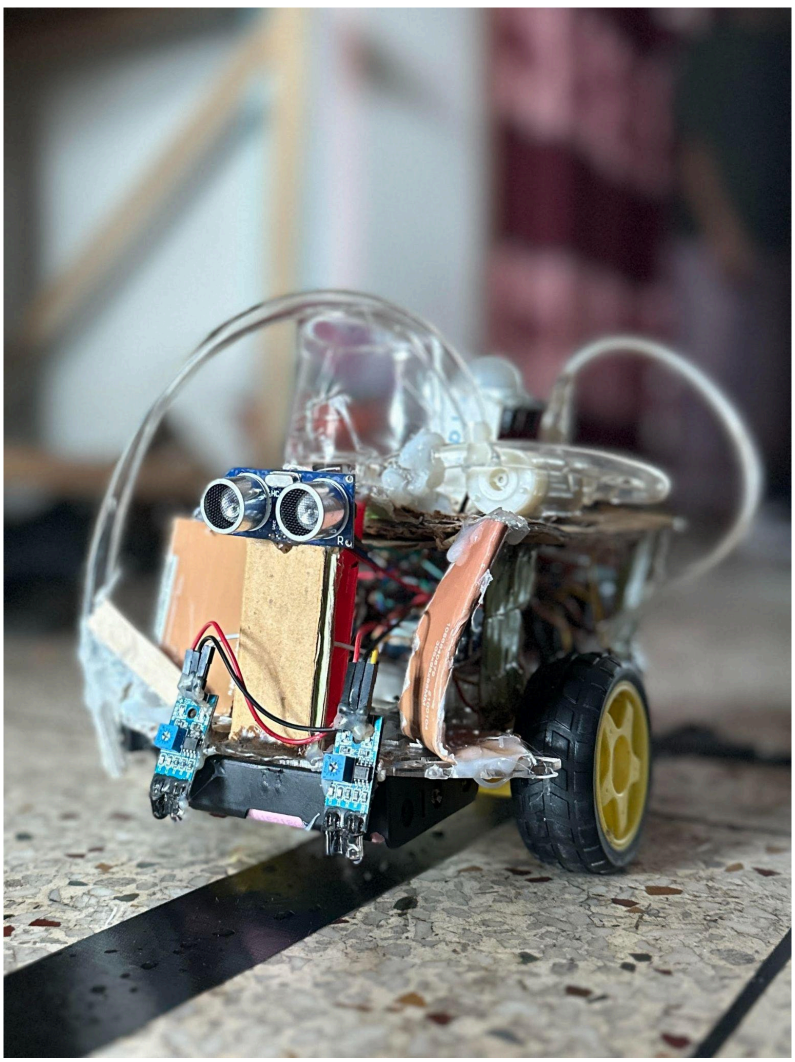
- **Target Audience:** Homeowners looking for automated, time efficient cleaning solutions, specially for physically challenged or sick individuals. Small offices, shops or the hospitality industry for commercial use.
- **Market Analysis and Competitors :** Robot vacuum cleaners is a fairly new concept and it's rapidly growing because of the demand for automation as much as possible in human life. Many companies have their own versions and models for robot vacuum cleaners. iRobot Roomba is the most popular cleaning robot but they only focus on vacuuming more than any other feature. Xiaomi Mi Robot Vacuum is another example, but the low tier models for this have very limited facilities while the high tier models are very costly. Shark IQ Robot is another popular option. But the mopping capabilities are limited while they mainly focus on vacuuming.
- **Revenue Model:** Selling robots directly to the customers. As add-ons, customers can get a bigger water tank and a more powerful vacuum. Customers could take monthly subscriptions for cleaning fluid solutions and vacuum filters. For premium subscriptions, customers can have more advanced cleaning algorithm, better object detections. The company could collaborate with eco-friendly cleaning fluids making the company to get better deals on bundled products. Customers can bring in their robots for servicing every 3-4 months in a cycle.

8. Potential Challenges

- **Technical Challenges:** Ultrasonic sensors will work differently if the robot is on soft or inclined planes. IR Sensors may not detect the dark line if it starts to follow a dark shadow.
- **Design Challenges:** Too many components could result in congestion on the robot, leaving very little space for modification or changes. If the weight is too much, it would put immense pressure on the BO Motors, ultimately damaging the motors.
- **Integration Challenges:** So many wires have been used to connect the components that it becomes hard to keep track of which wire is going into which pin. Installing the components according to the diagram to maintain the direction it is in was also important to quick check after installing everything.
- **Budget Constraints**: As beginners, it's not always easy to know which components are needed and to work carefully and efficiently. That's why there has always been needed some extra pieces of components in case the currently in-use components break down.
- **Risk Mitigation:** Properly measure and place the sensors at the proper places for those to work as you want. Install and run multiple times until the desired values are achieved and keep in mind the limitations of the used sensors. Take time and efficiently use

components that are necessary, without increasing load on the BO Motors. Place components by equally distributing weight for the robot to move smoothly. Make the circuit diagram first, review it and then follow it properly to connect the components to avoid errors in connections. It is good to have spare components for use if necessary, but it would be much more cost effective if some research is done beforehand for exactly which components will be used and checking reviews on which components tend to break down.

Picture →



CODE →

```
#include <Servo.h>

// Motor Driver Pin Definitions (for wheels)
#define IN1 5
#define IN2 4
#define ENA 6
#define IN3 3
#define IN4 2
#define ENB 9

// Motor Driver Pin Definitions (for vacuum and water pump)
#define VACUUM_RUN 7
#define WATER_RUN 8
#define VACUUM_WATER_ENA 10

// IR Sensor Pins
#define IR_LEFT A0
#define IR_RIGHT A1

// Motion Sensor Pin
#define MOTION_SENSOR A2
#define MOTION_LED 12 // LED to indicate motion detection

// Sonar Pins
#define TRIG_PIN A3
#define ECHO_PIN A4
```

```
// Brush Servo Pin (using the sonar servo for brush sweeping)
#define BRUSH_SERVO_PIN A5

// Distance threshold for obstacles (in cm)
#define OBSTACLE_THRESHOLD 20

// Motion Detection Debounce Time (in ms)
#define MOTION_CLEAR_TIME 1000

// Global Variables
Servo brushServo;           // Servo for brush (formerly sonar servo)
unsigned long sweepMillis = 0; // Timing for the brush servo
int sweepAngle = 0;          // Current angle of the brush servo
int sweepDirection = 1;      // Sweep direction: 1 for forward, -1 for
backward
bool isSweeping = false;     // Whether the brush servo is sweeping

bool isObstacleDetected = false; // Obstacle detection flag
bool isMotionDetected = false;   // Motion detection flag
unsigned long motionClearMillis = 0; // Timer for motion to clear
unsigned long clearDuration = 1000;  // Time (in ms) for an obstacle to be
considered cleared

// Function to measure distance using Sonar
float measureDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
```

```
  digitalWrite(TRIG_PIN, HIGH);

  delayMicroseconds(10);

  digitalWrite(TRIG_PIN, LOW);


  long duration = pulseIn(ECHO_PIN, HIGH, 30000); // Timeout at 30ms for
faster reads

  float distance = (duration * 0.0343) / 2;

  return (distance >= 2 && distance <= 400) ? distance : -1; // Return -1
if out of range

}


// Functions to control the robot

void stopRobot() {

  // Stop wheels

  digitalWrite(IN1, LOW);

  digitalWrite(IN2, LOW);

  digitalWrite(IN3, LOW);

  digitalWrite(IN4, LOW);

  analogWrite(ENA, 0);

  analogWrite(ENB, 0);

}


void moveForward() {

  // Move wheels forward

  digitalWrite(IN1, HIGH);

  digitalWrite(IN2, LOW);

  digitalWrite(IN3, HIGH);

  digitalWrite(IN4, LOW);
```

```
  analogWrite(ENA, 255); // Maximum speed
  analogWrite(ENB, 255);
}

void turnLeft() {
  // Turn wheels left
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 255);
  analogWrite(ENB, 0);
}

void turnRight() {
  // Turn wheels right
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 0);
  analogWrite(ENB, 255);
}

void enableWaterVacuumBrush() {
  // Start vacuum, water pump, and brush
  digitalWrite(VACUUM_RUN, HIGH);
```

```
  digitalWrite(WATER_RUN, HIGH);
  analogWrite(VACUUM_WATER_ENA, 255);
  isSweeping = true; // Enable sweeping
}

void disableWaterVacuumBrush() {
  // Stop vacuum, water pump, and brush
  digitalWrite(VACUUM_RUN, LOW);
  digitalWrite(WATER_RUN, LOW);
  analogWrite(VACUUM_WATER_ENA, 0);
  isSweeping = false; // Disable sweeping
}

void setup() {
  // Wheel Motor Driver Pins
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);

  // Vacuum and Water Pump Motor Driver Pins
  pinMode(VACUUM_RUN, OUTPUT);
  pinMode(WATER_RUN, OUTPUT);
  pinMode(VACUUM_WATER_ENA, OUTPUT);
```

```
  // IR Sensors and Motion Detector
  pinMode(IR_LEFT, INPUT);
  pinMode(IR_RIGHT, INPUT);

  // Sonar Pins
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Motion Sensor Pin
  pinMode(MOTION_SENSOR, INPUT);
  pinMode(MOTION_LED, OUTPUT);

  // Brush Servo (using sonar servo)
  brushServo.attach(BRUSH_SERVO_PIN);

  // Initialize Serial Monitor
  Serial.begin(9600);
  stopRobot(); // Start with the robot stopped
}

void loop() {
  // Check Sonar for obstacles
  float distance = measureDistance();
  static unsigned long clearStartMillis = 0; // Timer for continuous clear
detection

  if (distance != -1 && distance <= OBSTACLE_THRESHOLD) {
```

```
    // If obstacle is detected, debounce and stop
    if (!isObstacleDetected) {
      Serial.println("Obstacle detected! Stopping.");
      digitalWrite(MOTION_LED, HIGH); // Turn on LED for obstacle
      isObstacleDetected = true;
      stopRobot(); // Stop robot immediately
      disableWaterVacuumBrush(); // Disable water, vacuum, and brush
    }
    clearStartMillis = 0; // Reset the clear detection timer
  } else {
    // If no obstacle is detected, check if obstacle is cleared for a
while
    if (isObstacleDetected) {
      if (clearStartMillis == 0) {
        clearStartMillis = millis(); // Start timing how long it's clear
      } else if (millis() - clearStartMillis >= clearDuration) {
        Serial.println("Obstacle cleared! Resuming.");
        digitalWrite(MOTION_LED, LOW); // Turn off LED
        isObstacleDetected = false;
        clearStartMillis = 0; // Reset the clear detection timer
      }
    }
  }

  // Check Motion Sensor
  if (digitalRead(MOTION_SENSOR) == HIGH) {
    if (!isMotionDetected) {
```

```
      Serial.println("Motion detected! Stopping.");
      digitalWrite(MOTION_LED, HIGH); // Turn on LED for motion detection
      isMotionDetected = true;
      motionClearMillis = millis(); // Record the time motion was detected
      stopRobot(); // Stop robot
      disableWaterVacuumBrush(); // Disable water, vacuum, and brush
    }
  } else {
    if (isMotionDetected && millis() - motionClearMillis >=
MOTION_CLEAR_TIME) {
      // Motion is cleared for the specified duration
      Serial.println("Motion cleared! Resuming.");
      digitalWrite(MOTION_LED, LOW); // Turn off LED
      isMotionDetected = false;
    }
  }

  // Line-following logic
  if (!isObstacleDetected && !isMotionDetected) {
    int leftIR = digitalRead(IR_LEFT);
    int rightIR = digitalRead(IR_RIGHT);

    if (leftIR == LOW || rightIR == LOW) {
      enableWaterVacuumBrush(); // Ensure water, vacuum, and brush are
running
      if (leftIR == LOW && rightIR == LOW) {
        moveForward(); // Move forward if both sensors detect the line
      } else if (leftIR == LOW) {
```

```
        turnLeft(); // Turn left if only the left sensor detects the line
      } else if (rightIR == LOW) {
        turnRight(); // Turn right if only the right sensor detects the
line
      }
    } else {
      // No IR sensor detects the line
      stopRobot();
      disableWaterVacuumBrush(); // Disable water, vacuum, and brush
    }
  }

  // Sweeping logic for brush servo
  if (isSweeping && millis() - sweepMillis >= 50) {
    sweepMillis = millis();
    sweepAngle += sweepDirection * 10;
    if (sweepAngle >= 180 || sweepAngle <= 0) {
      sweepDirection *= -1; // Reverse direction
    }
    brushServo.write(sweepAngle);
  }
}
```