
**Indian Institute of Technology Guwahati**

# Voice & Obstacle Avoiding with Rain Detector Robot Car

*A Design Lab Project Report Submitted By*

**Malothu Akash**
Roll No: 220108034

**Santanu Kausik Das**
Roll No: 220108052

**Under the Guidance of**
Dr. Gaurav Trivedi

**Department of Electrical and Electronics Engineering**
**Indian Institute of Technology Guwahati**
April 2025

# CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Voice Obstacle Avoiding with Rain Detector Robot Car**" is a bonafide work of **Malothu Akash (Roll No. 220108034)** and **Santanu Kausik Das (Roll No. 220108052)**, carried out in the Department of Electrical and Electronics Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Gaurav Trivedi**

Professor,

Department of Electronics and Electrical Engineering

Indian Institute of Technology Guwahati

April 2025

# Contents

# List of Tables

# Chapter 1

# Introduction

## 1.1 Overview

The evolution of robotics in automation, surveillance, and daily utility has inspired this project, which aims to design a smart robotic vehicle that can be controlled via voice commands and can switch to autonomous navigation when needed. It aims to combine manual control with basic artificial intelligence-like behavior through sensors.

## 1.2 Objective

- To build a Four wheel driven car. Control via voice and obstacle avoidance smart RC Car
- To implement Bluetooth-based manual control through voice commands.
- To automatically switch to obstacle avoidance mode when inactive.
- To integrate a rain detection module to halt or alert during rainy conditions.
- To make the platform expandable for additional features like WiFi control.

# Chapter 2

# Design of the System

## 2.1 System Layout

The robot chassis is a four-wheeled platform with all wheels powered by DC motors via an L298N motor driver sheild . The four wheels are fixed, and turning is achieved by differential motor control.

## 2.2 Functional Modules

- **Manual Control Module**: Accepts voice commands via Bluetooth.
- **Autonomous Module**: Uses ultrasonic sensors to navigate without user input.
- **Rain Detection Module**: Monitors rain presence using a sensor and either halts the robot or provides alert.
- **Control Logic**: Embedded on Arduino UNO and developed in C++ using Arduino IDE.

## 2.3 Component Specifications

### 2.3.1 DC Motor

DC motors are used to drive the all wheels of the robot car. These motors convert electrical energy into mechanical motion to move the robot forward, backward, and turn by differential speed control.

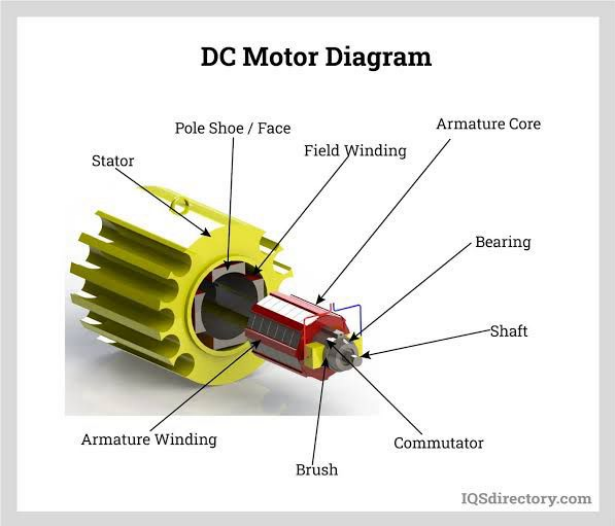


**Fig. 2.1** Standard DC Motor

**Specifications of DC Motor:**

- **Model No.** – BO-type DC Motor
- **Motor Type** – Brushed DC Motor

**Technical Details:**

- Operating Voltage: 6V to 12V
- No-Load Speed: 150–300 RPM (at 6V)
- Stall Torque: 0.8–3 Kgfcm (at 6V)
- Rated Torque: 1 Kgfcm

- No-Load Current: 100–250 mA
- Stall Current: Up to 1.2 A
- Weight: 30 g
- Shaft Diameter: 6 mm

### 2.3.2 Ultrasonic Sensor

The ultrasonic sensor is used to measure the distance to obstacles by sending sound pulses and calculating the time taken for the echo to return.

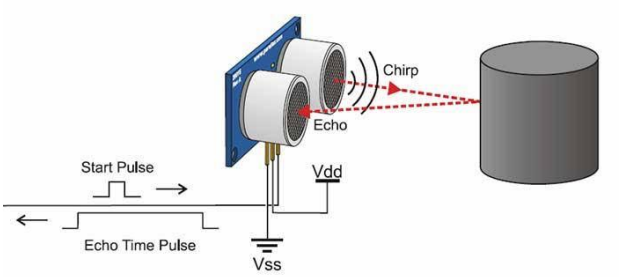


**Fig. 2.2** HC-SR04

**Fig. 2.3** HC-SR04 Ultrasonic Sensor

**Specifications of Ultrasonic Sensor:**

- **Model No.**: HC-SR04
- **Sensor Type**: Ultrasonic Distance Measurement

**Technical Details:**

- Operating Voltage: 5V DC
- Operating Current: ¡15 mA
- Measuring Range: 2 cm – 20 cm

- Accuracy: ±3 mm
- Frequency: 40 kHz
- Trigger Pulse Width: 10 µs
- Measuring Angle: 15°

### 2.3.3 Bluetooth Module

The HC-05 Bluetooth module enables wireless communication between the robot and a mobile phone using serial communication (UART).

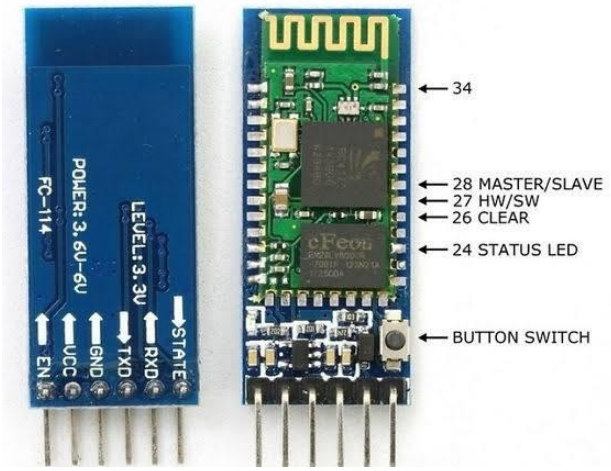


**Fig. 2.4** HC-05

**Fig. 2.5** HC-05 Bluetooth Module

**Specifications of HC-05 Bluetooth Module:**

- **Model No.**: HC-05
- **Module Type**: Serial Port Protocol (SPP)

**Technical Details:**

- Operating Voltage: 3.3V to 5V
- Current Consumption: 30 mA
- Bluetooth Version: 2.0 + EDR
- Frequency: 2.4 GHz ISM band
- Baud Rate: Default 9600 bps (Configurable)
- Communication: UART (TX/RX)
- Transmission Range: Up to 10 meters
- Operating Modes: Master/Slave

### 2.3.4 Rain Detection Sensor

The rain sensor module is used to detect the presence of rain or water droplets. It consists of two parts: the rain detection board and the control module. When raindrops fall on the sensor, the resistance decreases and the analog output voltage changes, which is detected by the Arduino.

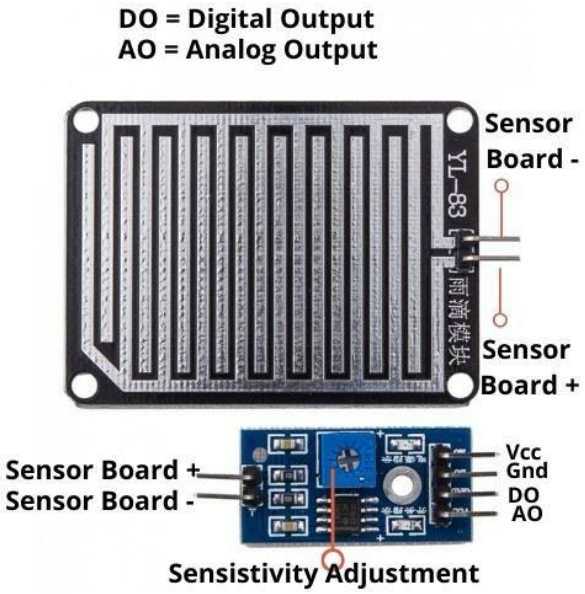


**Fig. 2.6** Rain Detector

**Fig. 2.7** YL-83

YL-83 Rain Detection Sensor Module

**Specifications of Rain Sensor Module:**

- **Model No.**: YL-83 or Raindrop Sensor Module
- **Sensor Type**: Analog + Digital Output

**Technical Details:**

- Operating Voltage: 3.3V – 5V
- Output Type: Analog (AO) and Digital (DO)
- Sensitivity: Adjustable via onboard potentiometer
- PCB Dimensions: 54mm x 40mm
- Detection Area: Corrosion-resistant nickel coating

- Current Consumption: ¡ 20 mA

**Working Principle:** The sensor works based on electrical conductivity. When water (rain) falls on the sensor plate, it forms a conductive path between tracks, reducing resistance. This change is detected as a drop in voltage (analog value) and triggers a digital signal if a threshold is crossed.

# Chapter 3

# Hardware and Software Integration

## 3.1 Components Used

| Component | Description |
|---|---|
| Arduino UNO | Main controller |
| L298N Motor Driver | Controls rear DC motors |
| 2 DC Motors | Rear-wheel propulsion |
| HC-05 Bluetooth Module | Receives voice commands via mobile app |
| Ultrasonic Sensor | Detects obstacles in front or rear |
| Rain Sensor | Detects raindrops on its surface |
| Servo Motor | For rotating ultrasonic sensor |
| Breadboard | For prototyping |
| Battery | 7.4V battery pack |

**Table 3.1** List of components used

## 3.2 Circuit Design

The complete circuit of the voice and obstacle-avoiding robot car is shown below:

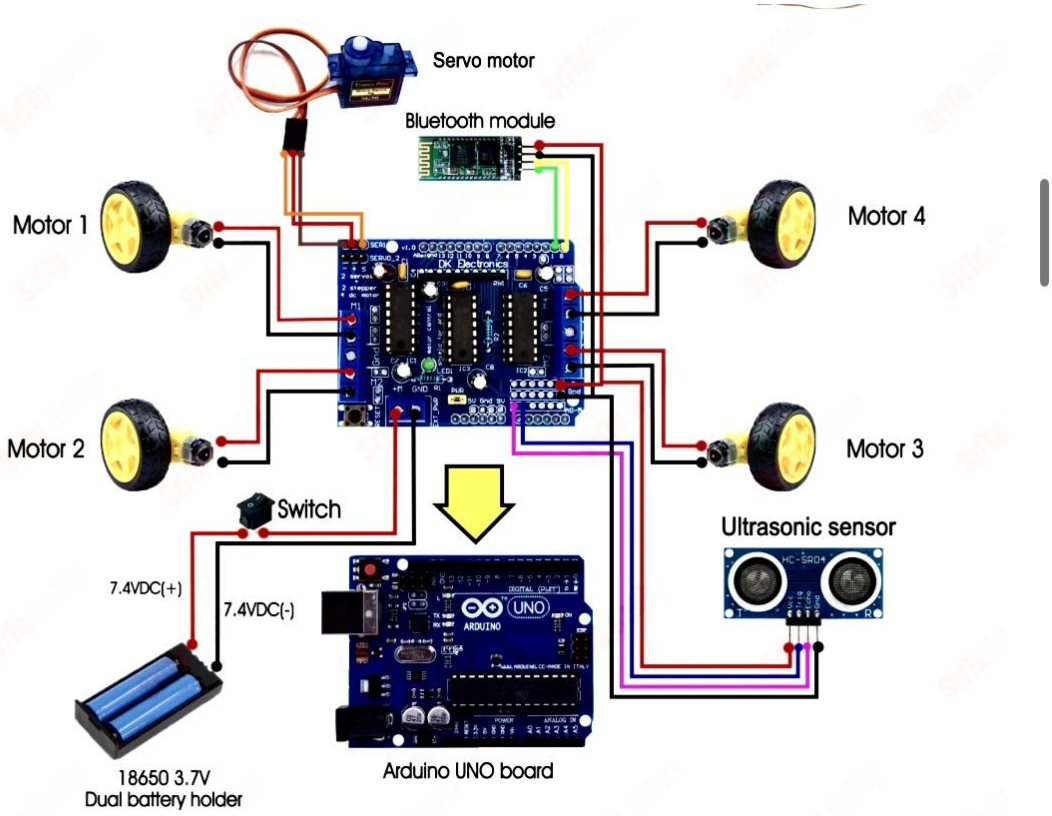


**Fig. 3.1** Complete Circuit Diagram of Voice & Obstacle Avoiding Robot Car

The circuit is built using the following connections:

- **Power Supply:**
  - A dual 18650 battery holder (7.4V DC output) powers the system through a toggle switch.
  - The power is fed to both the Arduino UNO and the motor driver shield.
- **Motor Driver Shield:**
  - Connected to 4 DC motors (Motor 1 to Motor 4) for a 4-wheel drive system.
  - The motor shield is stacked directly onto the Arduino UNO board, using its power and I/O pins.

- **Bluetooth Module (HC-05):**
  - VCC to 5V, GND to GND
  - TX to Arduino RX (D0) via voltage divider
  - RX to Arduino TX (D1)
- **Ultrasonic Sensor (HC-SR04):**
  - VCC to 5V, GND to GND
  - Trig to Digital Pin D10
  - Echo to Digital Pin D11
- **Servo Motor:**
  - Used for rotating the ultrasonic sensor for obstacle scanning.
  - Signal pin connected to Digital Pin D9

This layout allows seamless integration of voice control (via Bluetooth), obstacle detection (ultrasonic), and servo-based scanning, all managed by the Arduino UNO and a motor shield for 4-wheel motion control.

# Chapter 4

# Modes of Operation and Results

## 4.1 Working Modes

### 4.1.1 Manual Mode (Bluetooth Voice Control)

Commands like `forward`, `stop`, `left`, etc., are sent via Bluetooth. Arduino decodes them via Serial and runs corresponding functions.

### 4.1.2 Autonomous Obstacle Avoidance

When no input is received for 10 seconds, robot uses ultrasonic sensor to detect obstacles and move accordingly.

### 4.1.3 Rain Detection Mode

If rain is detected, the car will stop automatically and optionally alert the user via a buzzer or LED.

## 4.2 Software Architecture

- Serial Command Parser

- Motor driver logic (HIGH/LOW signals)
- Ultrasonic distance calculation using pulse duration
- Timer to switch from manual to auto mode
- Rain sensor analog threshold check

## 4.3 Results

| **Test Case** | **Expected** | **Result** |
|---|---|---|
| Voice command: forward | Moves forward | Success |
| Voice command: stop | Stops motors | Success |
| Idle for 10s | Switches to auto mode | Success |
| Obstacle ¡ 12cm | Turns or reverses | Success |
| Rain detected | Car halts and alerts | Success |
| Manual command post-auto | Switches to manual | Success |

**Table 4.1** Functional testing results

# Chapter 5

# Arduino Code 1

### Complete Arduino Source Code

```
#include <Servo.h>

Servo servo;

char val;

int trigPin = 10;

int echoPin = 11;

int servoPin = 9;

int RainSensor = A0;

long duration;

int distance;

unsigned long lastCommandTime = 0;

unsigned long timeout = 10000;


void setup() {

  pinMode(2, OUTPUT);

  pinMode(3, OUTPUT);

  pinMode(4, OUTPUT);
```

```
  pinMode(5, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(RainSensor, INPUT);
  servo.attach(servoPin);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available()) {
    val = Serial.read();
    lastCommandTime = millis();
  }

  if (millis() - lastCommandTime > timeout) {
    autonomousMode();
  } else {
    manualMode();
  }
}

void manualMode() {
  if (analogRead(RainSensor) < 800) {
    stopCar();
    return;
  }
  switch(val) {
```

```
    case 'F': forward(); break;
    case 'B': backward(); break;
    case 'L': left(); break;
    case 'R': right(); break;
    case 'S': stopCar(); break;
  }
}

void autonomousMode() {
  if (analogRead(RainSensor) < 800) {
    stopCar();
    return;
  }
  int distances[3];
  int angles[3] = {0, 90, 180};
  for (int i = 0; i < 3; i++) {
    servo.write(angles[i]);
    delay(500);
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distances[i] = duration * 0.034 / 2;
  }
```

```
  if (distances[1] < 12) {
    if (distances[0] > distances[2]) left();
    else right();
  } else {
    forward();
  }
}

void forward() {
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
}
void backward() {
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
}
void left() {
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
}
void right() {
```

```
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
}
void stopCar() {
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
}
```

# Chapter 6

# Arduino Code 2

### Complete Arduino Source Code

```
#include <Servo.h>
#include <AFMotor.h>

#define Echo A0
#define Trig A1
#define motor 10
#define Speed 170
#define spoint 103

char value;  // Declare 'value' globally to be accessible in all functions
int distance;
int Left;
int Right;
int L = 0;
int R = 0;
int L1 = 0;
```

```
int R1 = 0;

Servo servo;
AF_DCMotor M1(1);
AF_DCMotor M2(2);
AF_DCMotor M3(3);
AF_DCMotor M4(4);

void setup() {
  Serial.begin(9600);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  servo.attach(motor);
  M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed);
  M4.setSpeed(Speed);
}

void loop() {
  Bluetoothcontrol();   // Handle Bluetooth commands
  voicecontrol();      // Handle voice commands
  Obstacle();          // Obstacle avoidance logic
}

void Bluetoothcontrol() {
  if (Serial.available() > 0) {
```

```
    value = Serial.read();  // Read the command from Bluetooth
    value = toupper(value);  // Convert to uppercase to handle both lowercase and uppercas
    Serial.println(value);    // Print received command
  }

  if (value == 'F') {
    forward();
  } else if (value == 'B') {
    backward();
  } else if (value == 'L') {
    left();
  } else if (value == 'R') {
    right();
  } else if (value == 'S') {
    Stop();
  }
}

void voicecontrol() {
  if (Serial.available() > 0) {
    value = Serial.read();  // Read the command from voice control input
    value = toupper(value);  // Convert to uppercase

    Serial.println(value);  // Print the received command

    if (value == '^') {
      forward();
```

```
    } else if (value == '-') {
      backward();
    } else if (value == '<') {
      left();
    } else if (value == '>') {
      right();
    } else if (value == '*') {
      Stop();
    }
  }
}

void Obstacle() {
  distance = ultrasonic();
  if (distance <= 12) {  // If obstacle is detected within 12 cm
    Stop();               // Stop the robot
    backward();           // Move backward
    delay(100);
    Stop();               // Stop the robot
    L = leftsee();        // Check the left side for obstacles
    servo.write(spoint);
    delay(800);
    R = rightsee();       // Check the right side for obstacles
    servo.write(spoint);

    if (L < R) {
      right();            // If the left side is clearer, turn right
```

```
      delay(500);
      Stop();
      delay(200);
    } else if (L > R) {
      left();             // If the right side is clearer, turn left
      delay(500);
      Stop();
      delay(200);
    }
  } else {
    forward();            // If no obstacle is detected, move forward
  }
}

// Ultrasonic sensor distance reading function
int ultrasonic() {
  digitalWrite(Trig, LOW);
  delayMicroseconds(4);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);
  long t = pulseIn(Echo, HIGH);
  long cm = t / 29 / 2; // Time converted to distance
  return cm;
}

void forward() {
```

```
  M1.run(FORWARD);
  M2.run(FORWARD);
  M3.run(FORWARD);
  M4.run(FORWARD);
}

void backward() {
  M1.run(BACKWARD);
  M2.run(BACKWARD);
  M3.run(BACKWARD);
  M4.run(BACKWARD);
}

void right() {
  M1.run(BACKWARD);
  M2.run(BACKWARD);
  M3.run(FORWARD);
  M4.run(FORWARD);
}

void left() {
  M1.run(FORWARD);
  M2.run(FORWARD);
  M3.run(BACKWARD);
  M4.run(BACKWARD);
}
```

```
void Stop() {
  M1.run(RELEASE);
  M2.run(RELEASE);
  M3.run(RELEASE);
  M4.run(RELEASE);
}

int rightsee() {
  servo.write(20);     // Rotate servo to the right position
  delay(800);
  Left = ultrasonic(); // Measure the distance on the left side
  return Left;
}

int leftsee() {
  servo.write(180);    // Rotate servo to the left position
  delay(800);
  Right = ultrasonic(); // Measure the distance on the right side
  return Right;
}
```

# Chapter 7

# Future Work

- Add line-following capability using IR sensors
- ESP8266-based WiFi remote control and telemetry
- Integration with Firebase or Blynk IoT platforms
- Add live camera streaming (ESP32-CAM upgrade)

0pt

# References

**Additional Authors and Publications:**


- T. L. Chen, M. Cakmak, and A. Dragan. *Planning for Socially Assistive Robots*. Springer, 2018.

- R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. MIT Press, 2011.

- A. Kumar and S. Gupta. "Bluetooth-based Control Systems," *International Journal of Embedded Systems*, vol. 7, no. 2, 2020.