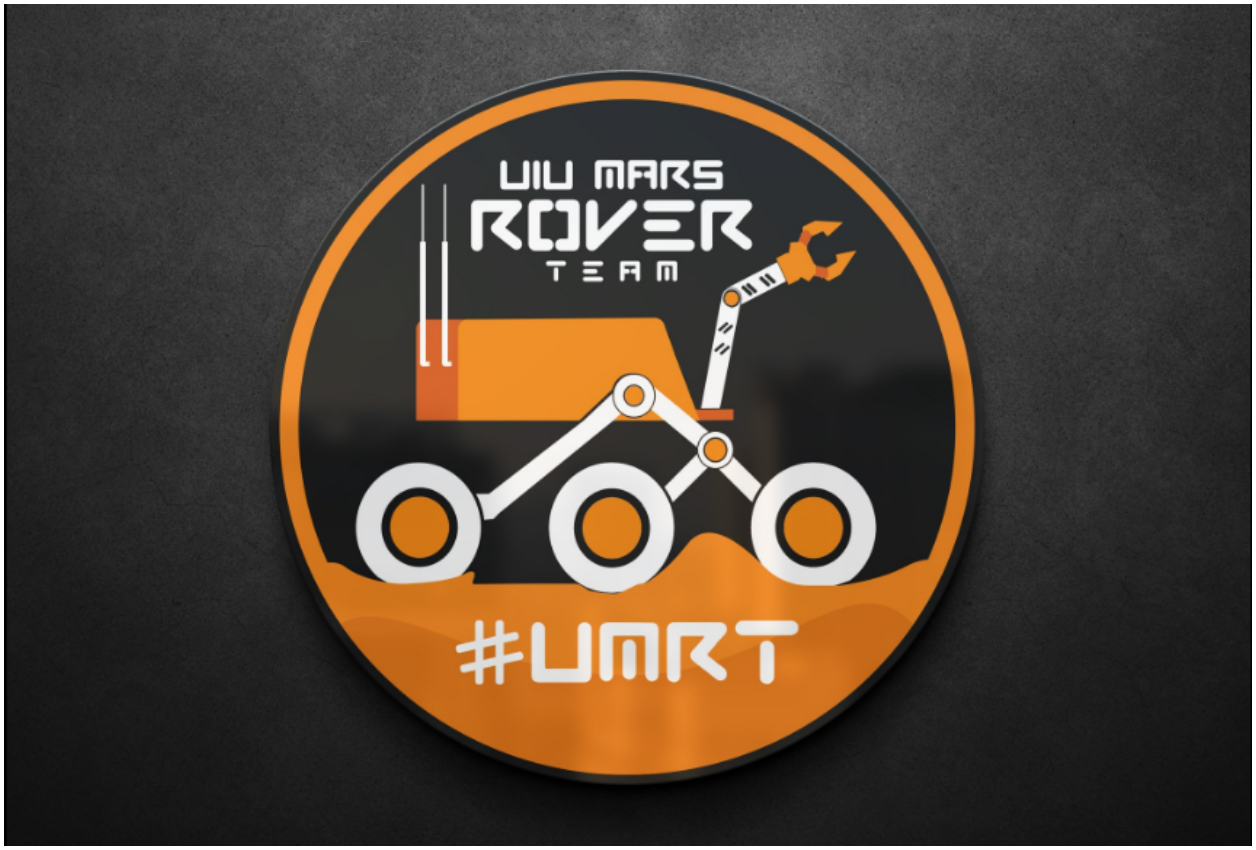



# Project Report of RC Object Avoiding Car

**Prepared By: Team READROCK**

{----------}

MD Habibulla Misba
Nazmul Hasan Athin
Naeem Abdullah Sadik
Abdullah Khan

{----------}

## Introduction:

This project report describes the design and construction of a Bluetooth controlled RC car with obstacle avoidance. The car is powered by two motors that are controlled by an ESP 32 microcontroller. The ESP 32 communicates via Bluetooth to receive commands from a user's smartphone. The car also has an ultrasonic sensor that is used to detect obstacles in its path. If an obstacle is detected within 20 cm of the car, the car will stop automatically.

The purpose of this project was to create a fun and educational project that would teach us about microcontrollers, electronics, programming, and robotics.

## Project Objectives:

The primary objectives of this project are as follows:

- Develop an RC car platform using an ESP32 microcontroller.
- Implement Bluetooth communication for remote control.
- Integrate an ultrasonic sensor for obstacle detection.
- Create control logic to navigate the car while avoiding obstacles.

## Project Methodology:

This project can be completed by the following steps:

**Hardware Setup**

- Assemble the RC car chassis and mount the motors.
- Connect the ESP32 microcontroller, ultrasonic sensor, and Bluetooth module.
- Configure the GPIO pins for motor control and sensor input.

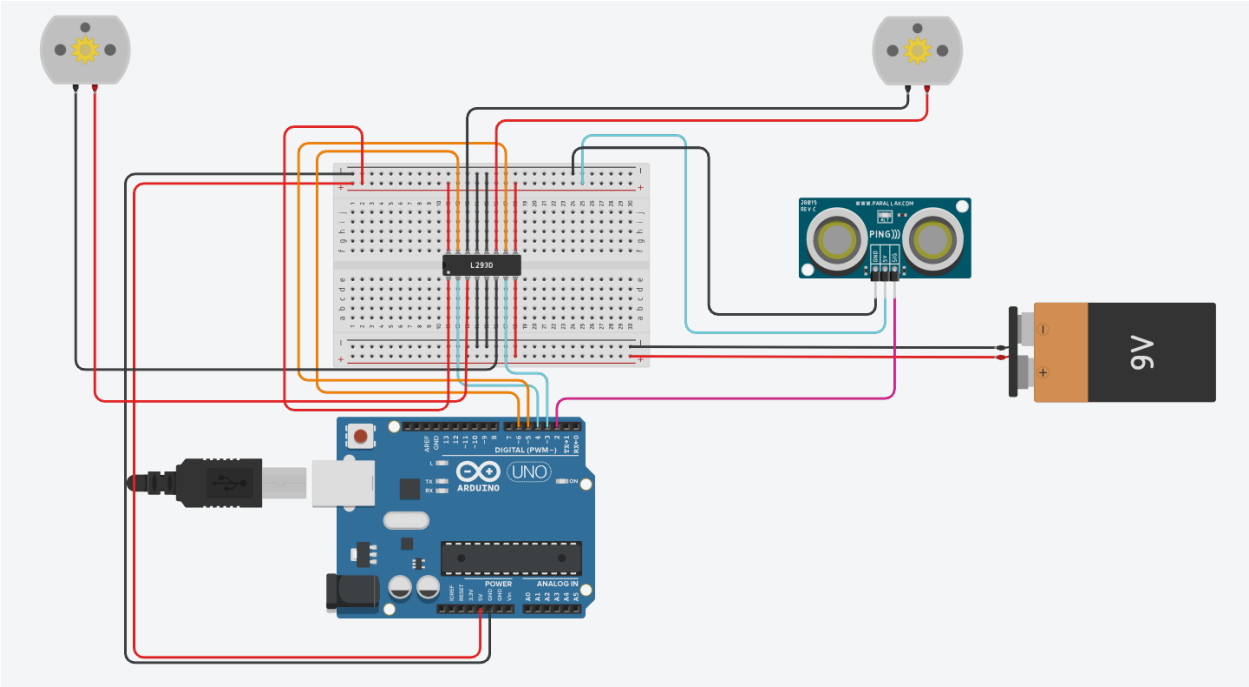
**Software Development**

- Write the code to control the motors in various directions (forward, backward, left, right, and stop).
- Implement Bluetooth communication using the BluetoothSerial library.
- Integrate obstacle detection logic using the NewPing library.
- Create a control loop to continuously monitor Bluetooth commands and obstacle distances.
- Develop a stop mechanism when obstacles are detected within a defined threshold.

**Testing and Validation**

- Conduct extensive testing to ensure that the RC car responds correctly to Bluetooth commands.
- Test the obstacle detection system to verify that the car stops when an obstacle is within range.
- Fine-tune the code and adjust parameters as needed for optimal performance.

## The Circuit Diagram:




## The Source Code:

```
#include "BluetoothSerial.h"
#include <NewPing.h>
#define ENA 25
#define IN1 2
#define IN2 4
#define ENB 26
#define IN3 15
#define IN4 13
#define LED_RED 16
#define LED_GREEN 17
#define TRIGGER_PIN 5
#define ECHO_PIN 18
#define MAX_DISTANCE 200
const char *pin = "1234";
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
String device_name = "TODOS";
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run make menuconfig to and enable
it
```

```
#endif
#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif
BluetoothSerial SerialBT;
void setup() {
  Serial.begin(9600);
  SerialBT.begin(device_name);  //Bluetooth device name
  Serial.printf("The device with name \"%s\" is started.\nNow you can pair it with Bluetooth!\n", device_name.c_str());
//Serial.printf("The device with name \"%s\" and MAC address %s is started.\nNow you can pair it with Bluetooth!\n", device_name.c_str(), SerialBT.getMacString()); // Use this after the MAC method is implemented
#ifdef USE_PIN
  SerialBT.setPin(pin);
  Serial.println("Using PIN");
#endif
  // Motor control pins as outputs
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  // Set motor A and B to stop initially
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  digitalWrite(LED_GREEN, HIGH);
}
void forward() {
  unsigned int distance = sonar.ping_cm();

  if (distance <= 25) {
    stop();
    return;
```

```
  }
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 255);  // Adjust the speed as needed
  analogWrite(ENB, 255);
}
void backward() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENA, 255);  // Adjust the speed as needed
  analogWrite(ENB, 255);
}
void left() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 255);  // Adjust the speed as needed
  analogWrite(ENB, 255);
}
void right() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENA, 255);  // Adjust the speed as needed
  analogWrite(ENB, 255);
}
void stop() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
}
```

```
void loop() {
  if (SerialBT.available() > 0) {
    digitalWrite(LED_GREEN, LOW);
    digitalWrite(LED_RED, HIGH);
    char command = SerialBT.read();  // Read the incoming character
    // Control the car based on the received command
    switch (command) {
      case 'U':
        forward();
        Serial.println("Forward");
        delay(100);
        break;
      case 'D':
        backward();
        Serial.println("Backward");
        delay(100);
        break;
      case 'L':
        left();
        Serial.println("Left");
        delay(100);
        break;
      case 'R':
        right();
        Serial.println("Right");
        delay(100);
        break;
      case 'S':
        stop();
        Serial.println("Stop");
        break;
      default:
        stop();
        break;
    }
  }
}
```

**Description of the code:**

The code begins by including necessary libraries, such as BluetoothSerial for Bluetooth communication and NewPing for ultrasonic sensor functionality. Various constants are defined for pin assignments, including motor control pins (ENA, IN1, IN2, ENB, IN3, IN4), LED pins (LED_RED, LED_GREEN), and ultrasonic sensor pins (TRIGGER_PIN, ECHO_PIN).

A PIN code for Bluetooth pairing is set as "1234". The code initializes the BluetoothSerial module as SerialBT with the device name "TODOS." It also sets the PIN code for pairing if USE_PIN is defined.

Pins for motor control and LEDs are configured as outputs.

Initial motor states are set to stop, and the green LED is turned on.
Several functions (forward, backward, left, right, stop) are defined to control the movement of the RC car. These functions set the motor pins accordingly to achieve specific movements.

Inside the loop function, the code continuously checks for incoming Bluetooth commands and measures the distance to obstacles using the ultrasonic sensor. If an obstacle is detected at a distance less than or equal to 25 cm, the car is stopped (stop function is called).

If no obstacle is detected, the code reads a command character from the Bluetooth connection and performs the corresponding action (e.g., move forward, backward, left, right, or stop) using the motor control functions. The code also updates LED states (LED_GREEN and LED_RED) to indicate whether the car is moving or stopped.

The code listens for Bluetooth commands ('U' for forward, 'D' for backward, 'L' for left, 'R' for right, 'S' for stop).

The sonar.ping_cm() function is used to measure the distance in centimeters to objects in front of the car. The code incorporates obstacle avoidance logic by checking the distance obtained from the ultrasonic sensor. If the measured distance is below the threshold of 25 cm, the car is stopped to prevent collisions.

The code provides feedback via Serial print statements, indicating the distance measured and the actions taken based on received commands.

Small delays (e.g., 100 milliseconds) are added after each movement command to allow the car to execute the action before processing the next command. The code mentions challenges faced, such as motor calibration, Bluetooth communication reliability, and optimization of obstacle detection, without providing specific solutions.

Overall, this code serves as the control logic for an RC car that can move in various directions based on Bluetooth commands while actively avoiding obstacles within a specified range. The code relies on sensor input and motor control to achieve these functionalities.

## The Final Result: