Bluetooth Controlled Robot Car BY USING ESP32

1. Introduction

The **Bluetooth Controlled Robot Car** is a simple mobile robotic system built using an ESP32 microcontroller with in-built Bluetooth, allowing wireless operation from a smartphone without additional modules. A motor driver controls two DC motors through direction pins only, without enable (PWM) pins, so the car runs at full speed when activated. By sending commands such as forward, backward, left, right, and stop via a Bluetooth terminal app, the user can control the movement of the robot. This project demonstrates wireless communication and basic robotics in an easy, low-cost, and beginner-friendly way, while also serving as a foundation for future enhancements like obstacle avoidance, speed control.

- **Bluetooth Controlled Robot Car**:

The project is a **Bluetooth-controlled 4-wheel robot car** using an ESP32 and an L298N motor driver.
It runs four DC motors (two on each side) controlled directly through direction pins without using enable (PWM) pins.
Commands sent from a smartphone via Bluetooth (`F`, `B`, `L`, `R`, `S`) control the car's movement.
This simple design demonstrates wireless control and robotics, and can be expanded with extra features.

2. Circuit Diagram:

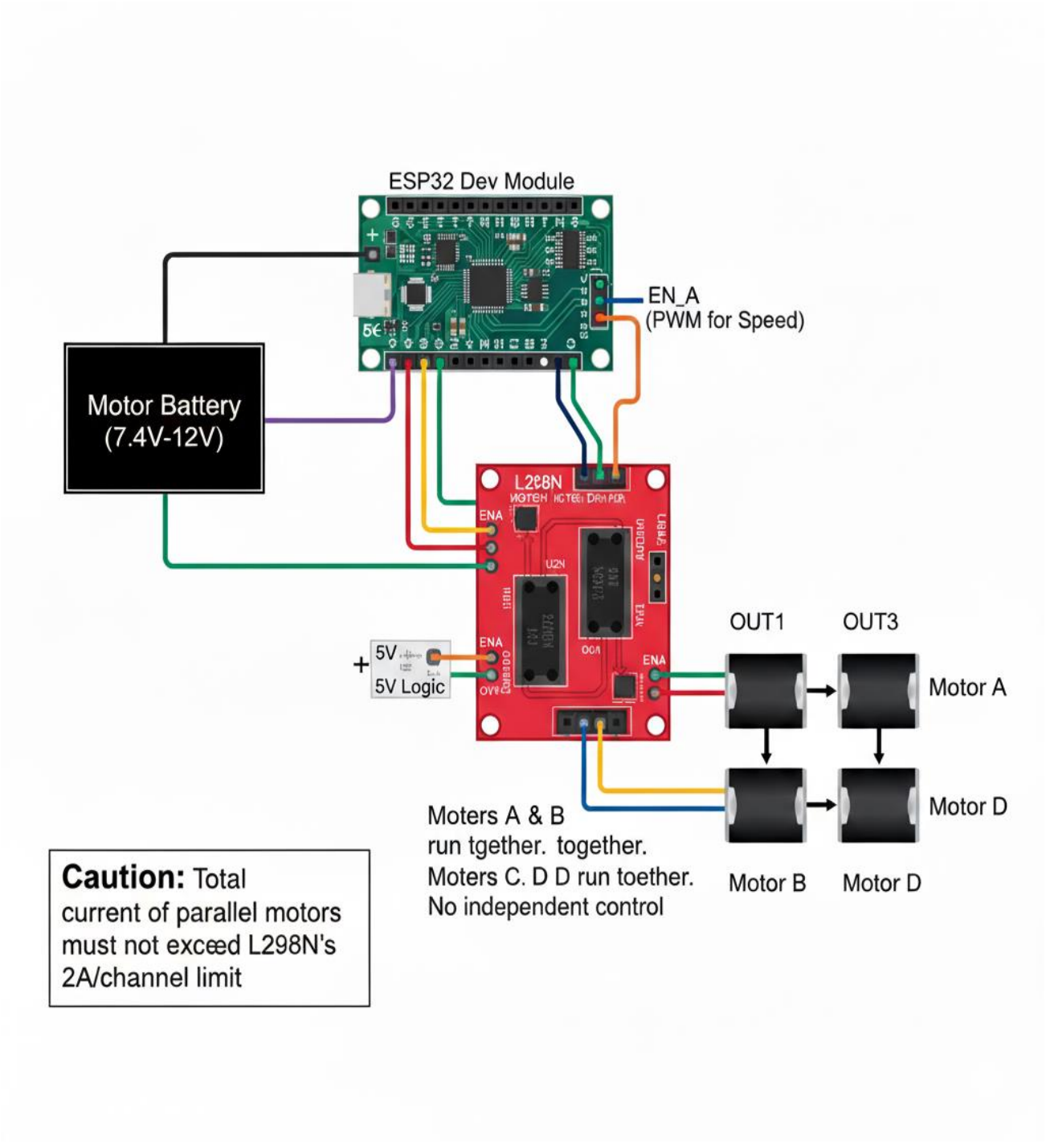


Figure 1: Circuit Diagram

3. Working and Explanation:

The project is a **Bluetooth-controlled robot car** that uses an **ESP32 microcontroller** as its control hub. The user can command the car wirelessly from a smartphone via a Bluetooth terminal app. The car's four **DC motors** are managed by a **single L298N motor driver**. Since the L298N has only two output channels, the motors are wired in two parallel pairs, with each pair acting as a single unit.

Key Components and Their Roles

- **ESP32 Microcontroller:** This is the brains of the car. Its integrated **Bluetooth** functionality allows it to receive commands from a smartphone without needing extra hardware.
- **Single L298N Motor Driver Module:** This acts as the **power amplifier**. The ESP32's outputs are not powerful enough to run motors directly. The L298N takes low-power signals from the ESP32 and uses a separate, high-power battery to drive the motors.
- **Four DC Motors:** These are the wheels of the car. They are wired in two parallel pairs—one pair for the left side and one for the right side. This setup allows the single L298N driver to control all four motors.
- **Motor Battery (7.4V - 12V):** A separate power source that provides the high current required by the motors.
- **Smartphone:** This serves as the remote control, running a Bluetooth terminal app to send commands like 'F' for forward or 'B' for backward.

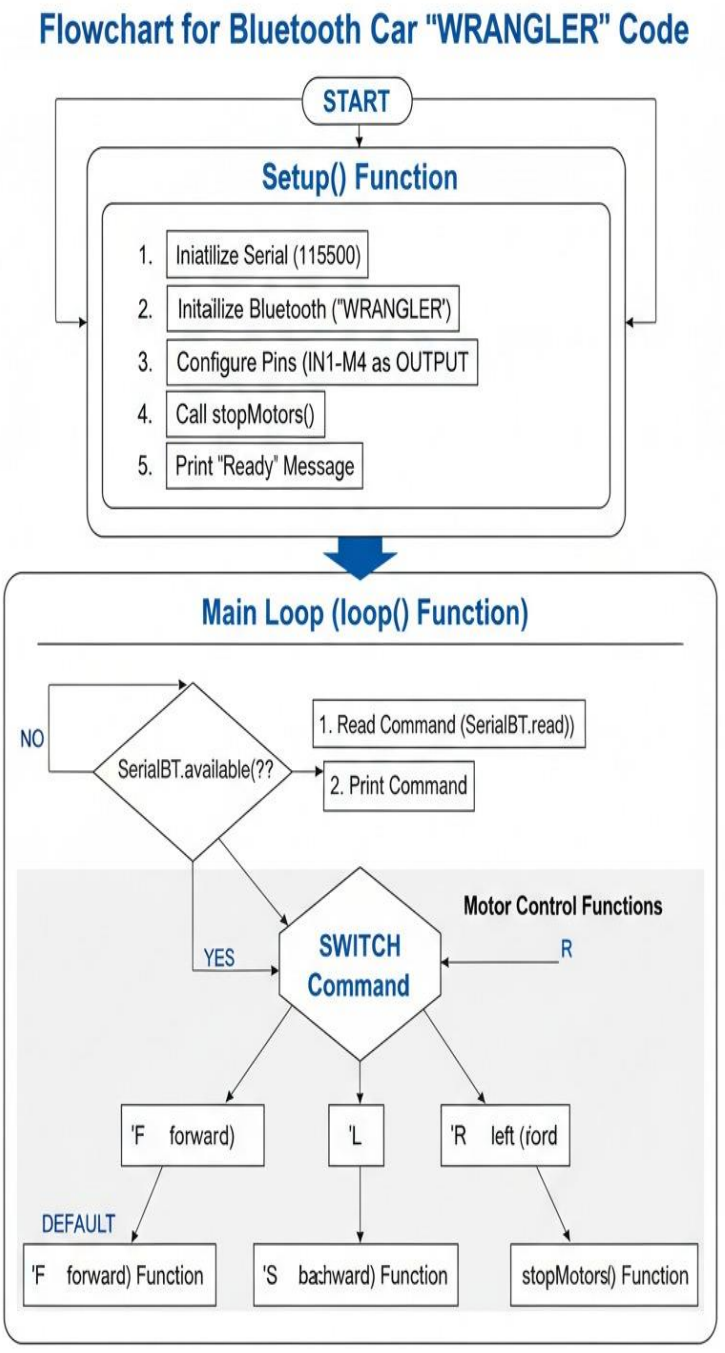
How the System Works

The system operates in a three-stage process:

1. **Command Reception:** A user connects their smartphone to the ESP32 via Bluetooth and sends a single-character command. The ESP32's Bluetooth module receives this command.
2. **Command Processing:** The ESP32's program constantly monitors for incoming data. When a command is received, the code uses conditional statements to match the character to a specific action.
3. **Motor Control:** The ESP32 sends **HIGH/LOW** digital signals to the single L298N driver's input pins. The L298N then controls the two motor pairs. The following table illustrates the logic:

Command	Action	Left Motors (IN1/IN2)	Right Motors (IN3/IN4)
F	Forward	HIGH / LOW	HIGH / LOW
B	Backward	LOW / HIGH	LOW / HIGH
L	Left Turn	LOW / HIGH	HIGH / LOW
R	Right Turn	HIGH / LOW	LOW / HIGH
S	Stop	LOW / LOW	LOW / LOW

4. FLOWCHART :


Flowchart for Bluetooth Car "WRANGLER" Code
START
Setup() Function
1. Iniatilize Serial (115500)
2. Initaïlize Bluetooth ("WRANGLER')
3. Configure Pins (IN1-M4 as OUTPUT
4. Call stopMotors()
5. Print "Ready" Message
Main Loop (loop() Function)
NO
SerialBT.available(??
1. Read Command (SerialBT.read))
2. Print Command
YES
SWITCH Command
Motor Control Functions
R
'F forward)
'L
'R left (rord
DEFAULT
'F forward) Function
'S backward) Function
stopMotors() Function

5. Program:

```
#include "BluetoothSerial.h"

// Create Bluetooth Serial object
BluetoothSerial SerialBT;

// Define motor driver pins (using L298N or similar)
#define IN1  26   // Motor A forward
#define IN2  25   // Motor A backward
#define IN3  33   // Motor B forward
#define IN4  32   // Motor B backward
void setup() {
  Serial.begin(115200);
  SerialBT.begin("WRANGLER");  // Bluetooth device name changed here ✓

 // Set motor pins as outputs
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  stopMotors();  // Initially stop
  Serial.println("Bluetooth Car 'WRANGLER' Ready. Connect and send commands.");
}
void loop() {
  if (SerialBT.available()) {
    char command = SerialBT.read();  // Read incoming char
    Serial.println(command);
    switch (command) {
      case 'F': forward();  break;  // Forward
      case 'B': backward(); break;  // Backward
      case 'L': left();     break;  // Left
      case 'R': right();    break;  // Right
      case 'S': stopMotors(); break; // Stop
      default: stopMotors(); break;
    }
  }
}
// Motor control functions
void forward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
void backward() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
```

```
void left() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
void right() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
void stopMotors() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}
```

6. Conclusion :

The **Bluetooth-controlled robot car** can be successfully built using an **ESP32 microcontroller** and a **single L298N motor driver**. This project demonstrates an effective way to use the ESP32's built-in Bluetooth to establish wireless communication with a smartphone, which serves as the remote control. By creatively wiring the four DC motors in two parallel pairs, the system overcomes the limitation of the single L298N driver, allowing for complete control over basic movements like forward, backward, and turning. This project provides a practical example of how to combine wireless communication, motor control, and power management to create a functional robotic system with a minimal number of components.

MODEL