BUILDING A WIRELESS CAR USING ESP32

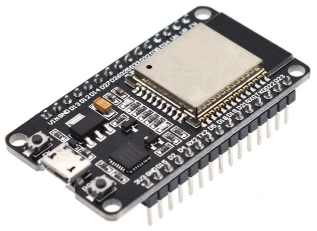
The evolution of microcontroller technology has enabled the development of efficient and scalable robotics projects. This report outlines the design, components, and implementation of a wireless car controlled using the ESP32 development board. The project leverages Bluetooth connectivity for real-time control and employs Pulse Width Modulation (PWM) for precise motor speed adjustments. This wireless car serves as a foundational project for students and professionals looking to explore robotics and IoT integration.

Objective

To design and implement a wireless car that can be controlled using a mobile application via Bluetooth. The project aims to demonstrate:

- **The use of an ESP32 board for wireless communication.**
- **Application of PWM for motor speed control.**
- **Integration of software and hardware components to achieve a functional prototype.**

Components

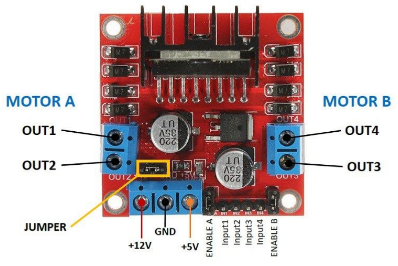


ESP32 Development Board:

- Acts as the central processing unit.
- Features integrated Bluetooth for wireless communication.

4x Gear Box with Wheels:

- Provides motion for the car.
- Ensures balanced and efficient movement.



L298N Motor Driver:

- Controls the motors' direction and speed.
- Supports PWM input for precise speed control.

Bluetooth (Integrated with ESP32):

- Allows for real-time control using a mobile application.

Dabble Mobile App:

- Provides a user-friendly interface for sending commands to the car.

Technical Details

1. ESP32 Setup

- The ESP32 is configured using the Arduino IDE:
  - Select the ESP32 development module from the board manager.
  - Connect the ESP32 to the computer for flashing the program.
  - Upload the code to enable motor control and Bluetooth communication.

2. Pulse Width Modulation (PWM)

PWM is a technique used to simulate analog output using digital signals. It allows for precise motor speed adjustments by modulating the duty cycle:

- **0% Duty Cycle**: Motor is off (analogWrite (0)).
- **25% Duty Cycle**: Quarter speed (analogWrite (64)).
- **50% Duty Cycle**: Half speed (analogWrite (127)).
- **75% Duty Cycle**: Three-quarter speed (analogWrite (191)).
- **100% Duty Cycle**: Full speed (analogWrite(255)).

The duty cycle determines the motor’s operational intensity by alternating the on and off states within a specific time frame.

3. Motor Control Using L298N

The L298N motor driver is critical for controlling the car’s motors. Key functionalities include:

- **Enable Pins (ENA, ENB)**:
  - Control motor speed using PWM signals.
  - Jumper present: Maximum speed.
  - Jumper removed: Speed controlled by PWM input.
- **Input Pins (IN1, IN2, IN3, IN4)**:
  - **Motor A (IN1, IN2):**
    - Forward: IN1=LOW , IN2=HIGH.
    - Backward: IN1=HIGH , IN2=LOW.
    - Stop: IN1=IN2=LOW or HIGH.
  - **Motor B (IN3, IN4):** Similar functionality as Motor A.

Implementation Steps

1. **Hardware Assembly**:
   - Connect the ESP32 to the L298N motor driver.
   - Attach the motors and wheels.
   - Ensure power supply compatibility (5V-25V for L298N).
2. **Programming**:
   - Flash the code onto the ESP32 using Arduino IDE.
   - Verify successful upload and code execution.
3. **Control via App**:
   - Pair the ESP32 with the Dabble app via Bluetooth.
   - Use the app's GamePad module for directional and speed control.

Circuit Diagram

A Fritzing-based circuit diagram illustrates the connections:

- ESP32 pins to L298N inputs (IN1, IN2, IN3, IN4).
- Power supply connections (5V and GND).
- Motors connected to motor outputs of L298N.

This project demonstrates the integration of hardware and software to create a wireless car. The ESP32 board's Bluetooth capabilities, coupled with the L298N motor driver, provide a scalable platform for robotics development. This setup can be expanded with additional sensors for autonomous navigation or enhanced functionality.