**Documentation for ESP32 Motor Control Setup with L298N**

### ****Overview****

This document outlines the connections and usage of the ESP32 microcontroller to control a 12V DC motor using the L298N motor driver module. The setup allows for direction control, speed adjustment, and the potential use of additional GPIO pins for other features.

### ****Components Used****

1. ESP32 (WROOM variant)
2. L298N Motor Driver Module
3. 12V DC Motor
4. 12V Battery (or other power source)
5. Jumper wires
6. Breadboard (optional)

### ****Wiring Diagram****

Refer to the wiring diagram provided earlier. The key connections are as follows:

| **ESP32 Pin** | **L298N Pin** | **Functionality** |
| --- | --- | --- |
| GPIO4 | IN1 | Motor Direction Control 1 |
| GPIO5 | IN2 | Motor Direction Control 2 |
| GPIO26 | ENA | Motor Speed Control (PWM) |
| GND | GND | Common Ground |
| 5V | +5V | Power for L298N logic |

### ****Updated Pin Usage****

If GPIO14 (G14), GPIO27 (G27), and GPIO26 (G26) are available, their functionality is as follows:

#### ****GPIO26 (ENA / PWM for Speed Control)****

* Connect to the **ENA** pin of the L298N for speed control using PWM.

#### ****GPIO14 and GPIO27 (IN1/IN2 for Direction Control)****

* GPIO14 can be connected to **IN1** for direction control.
* GPIO27 can be connected to **IN2** for direction control.

#### ****Repurposing GPIO Pins****

If all motor driver pins are already connected, GPIO14, GPIO27, and GPIO26 can be repurposed for:

1. Additional digital control (e.g., another motor or relay).
2. Reading sensor data (e.g., ultrasonic sensor or buttons).
3. Controlling LEDs for visual feedback.

### ****Sample Code****

The following code provides basic functionality for controlling the motor's speed and a

### ****Additional Features****

#### ****Using GPIO14, GPIO27, and GPIO26 for Other Purposes****

**GPIO14 (LED Brightness Control)**

* + Connect to an LED for brightness control using PWM.

**GPIO27 (Sensor Input)**

* + Use as a digital or analog input for a sensor (e.g., ultrasonic sensor or push button).

**GPIO26 (PWM for Another Motor)**

* + Use to control another motor or device requiring PWM.

#### ****Example: LED Brightness Control****

const int LED\_PIN = 14; // GPIO14 for LED brightness control

#include <WiFi.h>

#include <ESPAsyncWebServer.h>

// Define motor control pins

const int IN1 = 14; // IN1 connected to GPIO14

const int IN2 = 27; // IN2 connected to GPIO27

const int ENA = 26; // ENA (Enable pin) connected to GPIO26

// WiFi credentials

const char \*ssid = "Nensi Batra";

const char \*password = "waheguru.";

// Create an asynchronous web server object on port 80

AsyncWebServer server(80);

// Function prototypes

void moveForward();

void moveBackward();

void stopMotor();

void setup() {

  Serial.begin(115200);

  Serial.println("Setup started...");

  // Set motor control pins as outputs

  pinMode(IN1, OUTPUT);

  pinMode(IN2, OUTPUT);

  pinMode(ENA, OUTPUT);

  // Start with motor off

  digitalWrite(IN1, LOW);

  digitalWrite(IN2, LOW);

  analogWrite(ENA, 0); // PWM duty cycle set to 0

  Serial.println("Motor pins initialized.");

  // Connect to Wi-Fi

  WiFi.begin(ssid, password);

  Serial.print("Connecting to Wi-Fi");

  while (WiFi.status() != WL\_CONNECTED) {

    Serial.print(".");

    delay(1000);

  }

  Serial.println("\nWi-Fi connected.");

  // Print IP address

  Serial.println("IP Address: ");

  Serial.println(WiFi.localIP());

  // Define HTTP endpoints

  server.on("/forward", HTTP\_GET, [](AsyncWebServerRequest \*request){

    moveForward();

    request->send(200, "text/plain", "Motor moving forward");

  });

  server.on("/backward", HTTP\_GET, [](AsyncWebServerRequest \*request){

    moveBackward();

    request->send(200, "text/plain", "Motor moving backward");

  });

  server.on("/stop", HTTP\_GET, [](AsyncWebServerRequest \*request){

    stopMotor();

    request->send(200, "text/plain", "Motor stopped");

  });

  // Start server

  server.begin();

}

void loop() {

  // No loop processing is needed as AsyncWebServer handles requests asynchronously

}

// Function to move motor forward

void moveForward() {

  Serial.println("Motor moving forward");

  digitalWrite(IN1, HIGH);

  digitalWrite(IN2, LOW);

  analogWrite(ENA, 255); // Full speed

}

// Function to move motor backward

void moveBackward() {

  digitalWrite(IN1, LOW);

  digitalWrite(IN2, HIGH);

  analogWrite(ENA, 125); // Full speed

  Serial.println("Motor moving backward");

}

// Function to stop the motor

void stopMotor() {

  digitalWrite(IN1, LOW);

  digitalWrite(IN2, LOW);

  analogWrite(ENA, 0); // Turn off motor

  Serial.println("Motor stopped");

}

// this code is working for backward and stop api calls on which the motor is moving in one direction otherwise the motor is stopping on forward api calls

### ****Power Considerations****

* Use a 12V battery to power the L298N's motor power input (**+12V and GND**).
* Ensure that the ESP32 and the L298N share a common ground.
* For the ESP32, use a 5V regulated power supply if powering directly from the battery.

### ****Troubleshooting****

**Motor Not Spinning**

* + Verify the connections between the ESP32, L298N, and the motor.
  + Check the power supply (12V) for the motor.

**Incorrect Direction**

* + Swap the connections for IN1 and IN2.

**ESP32 Resetting/Rebooting**

* + Ensure the ESP32 is powered using a stable 5V source.
  + Avoid powering the ESP32 directly from the motor's power source without proper regulation.

### ****Conclusion****

This setup provides a simple and effective way to control a 12V DC motor using the ESP32 and the L298N motor driver module. The additional GPIO pins can be repurposed for extended functionalities, enhancing the project's capabilities.