ESP32 LED Brightness Control using a Potentiometer

---

1. Objective

This project controls the **brightness of an LED** based on the **analog input from a potentiometer** using **Pulse Width Modulation (PWM)** on an **ESP32**.

2. Required Components

Component	Quantity
ESP32 Dev Board	1
LED	1
220–330Ω resistor	1
Potentiometer (10kΩ)	1
Breadboard + wires	1 set

3. Pin Configuration

Name	GPIO Pin	Function
`LED_PIN`	GPIO 17	PWM output to control LED
`POTENTIOMETER_PIN`	GPIO 34	Analog input (ADC)

Note:

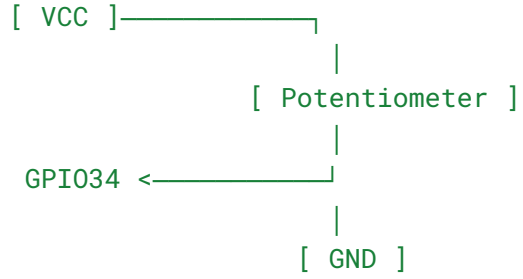
- GPIO 17 is PWM-capable
- GPIO 34 is ADC-capable (analog read)
- GPIO 34 is input-only (can’t be used for output)

4. Wiring Diagram

🔸 **LED Circuit:**

- Connect the **anode** of the LED to **GPIO17** through the **resistor**
  Connect the **cathode** to **GND**

🔸 **Potentiometer Circuit:**



- Connect one side pin of the potentiometer to **3.3V (VCC)**
- Connect the other side pin to **GND**
- Connect the middle pin (wiper) to **GPIO 34**

👩‍💻 5. Code Explanation (Line by Line)

Full code

```
#include<Arduino.h>
#define LED_PIN 17             // Use a real GPIO pin (PWM-capable)
#define POTENTIOMETER_PIN 34   // Use ADC-capable GPIO

void setup() {
  ledcSetup(0, 5000, 8);            // Channel 0, 5kHz, 8-bit resolution
  ledcAttachPin(LED_PIN, 0);        // Attach channel 0 to LED_PIN
  Serial.begin(115200);
}

void loop() {
  int potentiometerValue = analogRead(POTENTIOMETER_PIN);  // 0 to 4095
  int brightness = potentiometerValue / 16;  // Convert to 0-255
  ledcWrite(0, brightness);  // Set PWM brightness
  Serial.printf("Pot value %d, Brightness %d\n",potentiometerValue,brightness);
  delay(10);
}
```

Include Library

```
#include<Arduino.h>
```

- Includes the core functions like `analogRead()`, `ledcWrite()`, etc., used in ESP32-based Arduino programs.

✅ **Define GPIO Pins**

```
#define LED_PIN 17             // Use a real GPIO pin (PWM-capable)
#define POTENTIOMETER_PIN 34   // Use ADC-capable GPIO
```

- `LED_PIN`: GPIO pin for LED output (PWM)
- `POTENTIOMETER_PIN`: GPIO pin for analog input

Setup Function

```
void setup() {
  ledcSetup(0, 5000, 8);             // Channel 0, 5kHz, 8-bit resolution
```

- Configures **PWM Channel 0**
  - `0` = Channel number
  - `5000` = Frequency in Hz (5kHz PWM)
  - `8` = 8-bit resolution (values from 0 to 255)

```
  ledcAttachPin(LED_PIN, 0);        // Attach channel 0 to LED_PIN
```

- Attaches the LED pin (`GPIO17`) to **PWM Channel 0**

```
  Serial.begin(115200);
```

- Initializes the serial monitor for debugging and viewing real-time values.

Loop Function

```
  int potentiometerValue = analogRead(POTENTIOMETER_PIN);  // 0 to 4095
```

- Reads analog voltage from potentiometer (0–3.3V)
- On ESP32, `analogRead()` returns values from **0 to 4095**

```
  int brightness = potentiometerValue / 16;  // Convert to 0-255
```

- Scales down the 0–4095 range to **0–255** (for 8-bit PWM)
- `/ 16` is a shortcut for: `4096 / 256 = 16`

```
  ledcWrite(0, brightness);  // Set PWM brightness
```

- Sends the brightness value to **PWM Channel 0**
- Controls **LED brightness** based on potentiometer position

```
Serial.printf("Pot value %d, Brightness %d\n",potentiometerValue,brightness);
```

- Prints the raw analog value and corresponding brightness level to the Serial Monitor

```
delay(10);
```

- Adds a small delay (10 ms) between readings to avoid flooding the serial monitor

6. Behavior Summary

Potentiometer Position	Analog Value	Brightness (PWM)	LED Behavior
Fully CCW (min)	~0	0	LED OFF
Mid Position	~2048	~128	LED Half Bright
Fully CW (max)	~4095	255	LED Fully Bright

7. Technical Concepts

🔹 **PWM (Pulse Width Modulation)**

- Controls the **average voltage** sent to the LED by **switching ON/OFF rapidly**
- **Duty cycle** determines brightness
  - 0% duty cycle → always OFF
  - 100% duty cycle → always ON

🔹 **ADC (Analog to Digital Conversion)**

- Converts analog voltage (0V to 3.3V) to digital value (0 to 4095)
- ESP32 has **12-bit ADC resolution** by default

✅ **8. Conclusion**

This project demonstrates how to:

- Use **PWM** on ESP32 for brightness control
- Read **analog values** from a potentiometer
  Combine analog input and digital output in real-time
- Understand core embedded concepts like ADC, PWM, and data scaling