ESP32 Web Server with Slider

Will guide you to build a web server-controlled PWM in ESP32.

Difficulté **Facile** | Durée **1 heure(s)** | Catégories **Électronique** | Coût **15 USD ($)**

Sommaire

Introduction

In this blog post, I will show you how to create a web server with an ESP32 board that allows you to control the brightness of an LED using a slider on a web page. This is a fun and easy project demonstrating how to use the ESP32's PWM (Pulse Width Modulation) feature to vary the intensity of an LED. You will also learn how to use the ESPAsyncWebServer and AsyncTCP libraries to handle asynchronous web requests and responses, which can improve the performance and responsiveness of your web server.

Matériaux

Hardware components:
DFRobot FireBeetle ESP32 IOT Microcontroller (Supports Wi-Fi & Bluetooth)

Software apps and online services:
Arduino IDE

Outils

Étape 1 - What You Will Need:

To follow along with this tutorial, you will need the following components:

- An ESP32 board (I'm using the ESP32 DevKitC)
- The Arduino IDE installed on your computer
- The ESPAsyncWebServer and AsyncTCP libraries installed on your Arduino IDE

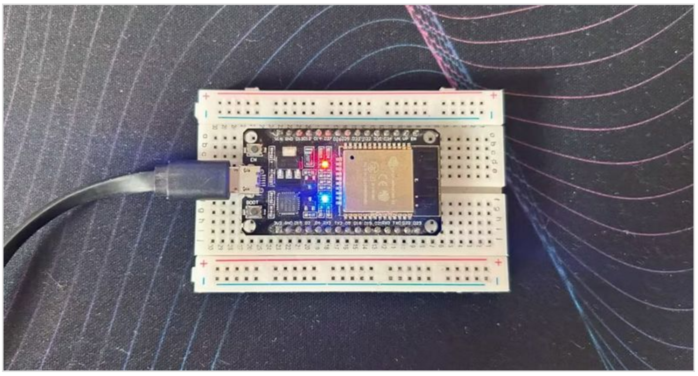
You can find the links to download the libraries and the complete code for this project at the end of this post.

Étape 2 - Get PCBs for Your Projects Manufactured

PCBWay 6th Project Design Contest
Electronic & Mechanical Design
PARTICIPATE NOW
PCBWay
Instant Quote
$5.00
$30

You must check out PCBWAY for ordering PCBs online for cheap! You get 10 good-quality PCBs manufactured and shipped to your doorstep for cheap. You will also get a discount on shipping on your first order. Upload your Gerber files onto PCBWAY to get them manufactured with good quality and quick turnaround time. PCBWay now could provide a complete product solution, from design to enclosure production. Check out their online Gerber viewer function. With reward points, you can get free stuff from their gift shop. Also, check out this useful blog on PCBWay Plugin for KiCad from here. Using this plugin, you can directly order PCBs in just one click after completing your design in KiCad.

Étape 3 - How It Works:



The basic idea of this project is to use a slider on a web page to send an HTTP request to the ESP32 with a value between 0 and 255, representing the LED's desired brightness. The ESP32 will then use that value to adjust the duty cycle of a PWM signal, which is a digital signal that switches between high and low states at a certain frequency. By changing the ratio of the high and low states, we can change the average voltage applied to the LED, and thus its brightness.

The inbuilt LED is connected to GPIO2, which is one of the pins that supports PWM (Pulse Width Modulation). PWM is a technique that allows us to vary the brightness of the LED by changing the duty cycle of a digital signal. The ESP32 has 16 PWM channels that can be configured to different frequencies and resolutions. By using the `ledcSetup()` `ledcWrite()` and functions, we can set up a PWM channel for the inbuilt LED and write a value between 0 and 255 to adjust its brightness. We can also use `Blink` the example sketch from the Arduino IDE to make the inbuilt LED blink with a certain interval. The inbuilt LED is useful for testing and debugging purposes, as well as for creating simple projects that involve lighting effects.

To create the web server, we will use the ESPAsyncWebServer library, which is an asynchronous web server library for the ESP32 and ESP8266. This library allows us to handle multiple web requests and responses without blocking the main loop of the ESP32, which can improve the performance and responsiveness of our web server. We will also use the AsyncTCP library, which is a dependency of the ESPAsyncWebServer library and provides low-level TCP communication between the ESP32 and the web browser.

To create the web page, we will use HTML, CSS, and JavaScript. HTML is the markup language that defines the structure and content of the web page.

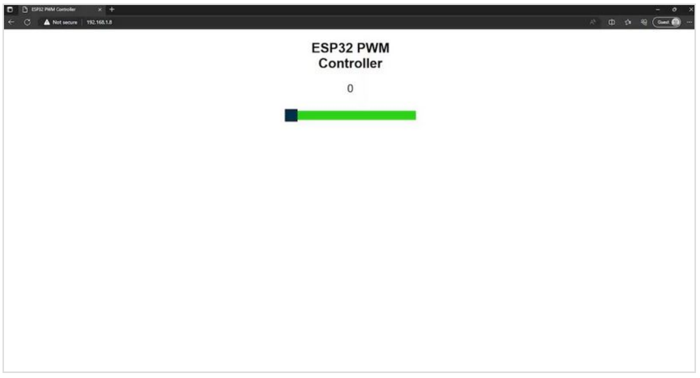
```
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1"
>
  <title>ESP32 PWM Controller</title>

  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    h2 {font-size: 2.3rem;}
    p {font-size: 1.9rem;}
    body {max-width: 400px; margin:0px auto; padding-bottom: 25px
;}
    .slider { -webkit-appearance: none; margin: 14px; width: 360px; h
eight: 25px; background: #2ad713;
      outline: none; -webkit-transition: .2s; transition: opacity .2s;}
    .slider::-webkit-slider-thumb {-webkit-appearance: none; appearan
ce: none; width: 35px; height: 35px; background: #003249; cursor:
pointer;}
    .slider::-moz-range-thumb { width: 35px; height: 35px; backgroun
d: #05abf8; cursor: pointer; }
  </style>

</head>
<body>
  <h2>ESP32 PWM Controller</h2>
  <p><span id="textSliderValue">%SLIDERVALUE%</span></p>
  <p><input type="range" onchange="updateSliderPWM(this)" id="pw
mSlider" min="0" max="255" value="%SLIDERVALUE%" step="1" clas
s="slider"></p>
<script>
function updateSliderPWM(element) {
  var sliderValue = document.getElementById("pwmSlider").value;
  document.getElementById("textSliderValue").innerHTML = sliderValue
;
  console.log(sliderValue);
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/slider?value="+sliderValue, true);
  xhr.send();
}
</script>
</body>
</html>
```

CSS is the style sheet language that defines the appearance and layout of the web page. JavaScript is the scripting language that adds interactivity and functionality to the web page. In this project, we will use JavaScript to capture the slider's value and send it to the ESP32 via an HTTP GET request.

```
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>

// Replace with your network credentials
const char* ssid = "ELDRADO";
const char* password = "amazon123";

const int output = 2;

String sliderValue = "0";

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

const char* PARAM_INPUT = "value";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1"
>
  <title>ESP32 PWM Controller</title>

  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    h2 {font-size: 2.3rem;}
    p {font-size: 1.9rem;}
    body {max-width: 400px; margin:0px auto; padding-bottom: 25px;}
    .slider { -webkit-appearance: none; margin: 14px; width: 360px; heig
ht: 25px; background: #2ad713;
      outline: none; -webkit-transition: .2s; transition: opacity .2s;}
    .slider::-webkit-slider-thumb {-webkit-appearance: none; appearance
: none; width: 35px; height: 35px; background: #003249; cursor: pointe
r;}
    .slider::-moz-range-thumb { width: 35px; height: 35px; background: #
05abf8; cursor: pointer; }
  </style>

</head>
<body>
  <h2>ESP32 PWM Controller</h2>
  <p><span id="textSliderValue">%SLIDERVALUE%</span></p>
  <p><input type="range" onchange="updateSliderPWM(this)" id="pwm
Slider" min="0" max="255" value="%SLIDERVALUE%" step="1" class
="slider"></p>
<script>
function updateSliderPWM(element) {
  var sliderValue = document.getElementById("pwmSlider").value;
  document.getElementById("textSliderValue").innerHTML = sliderValue
;
  console.log(sliderValue);
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/slider?value="+sliderValue, true);
  xhr.send();
}
</script>
</body>
</html>
)rawliteral";

// Replaces placeholder with button section in your web page
String processor(const String& var){
  //Serial.println(var);
  if (var == "SLIDERVALUE"){
    return sliderValue;
  }
  return String();
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);

  // configure LED PWM functionalitites
  ledcSetup(ledChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(output, ledChannel);

  ledcWrite(ledChannel, sliderValue.toInt());

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

  // Print ESP Local IP Address
  Serial.println(WiFi.localIP());
  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });

  // Send a GET request to <ESP_IP>/slider?value=<inputMessage>
  server.on("/slider", HTTP_GET, [] (AsyncWebServerRequest *request
) {
    String inputMessage;
    // GET input1 value on <ESP_IP>/slider?value=<inputMessage>
    if (request->hasParam(PARAM_INPUT)) {
      inputMessage = request->getParam(PARAM_INPUT)->value();
      sliderValue = inputMessage;
      ledcWrite(ledChannel, sliderValue.toInt());
    }
    else {
      inputMessage = "No message sent";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
  });

  // Start server
  server.begin();
}

void loop() {

}
```

Étape 4 - Wrap-Up:

I hope you enjoyed this introduction to the ESP32 web server with the Slider project. In the next section, I will show you how to wire the circuit and program the ESP32. Stay tuned!


ESP32 PWM Controller
Not secure
192.168.1.8
ESP32 PWM
Controller
0