<https://randomnerdtutorials.com/esp8266-web-server-spiffs-nodemcu/>

|  |  |
| --- | --- |
|  | * The web server controls an LED connected to the ESP8266 GPIO 2. This is the ESP8266 on-board LED. You can control any other GPIO; * The web server page shows two buttons: ON and OFF – to turn GPIO 2 on and off; * The web server page also shows the current GPIO state;   You’ll also use a BME280 sensor to display sensor readings (temperature, humidity, and pressure). |

**Installing Libraries**

One of the easiest ways to build a web server using files from the filesystem is using the [ESPAsyncWebServer](https://github.com/me-no-dev/ESPAsyncWebServer)and **ESPAsyncTCP** libraries, they are not available through the Arduino IDE libraries manager. Follow the next steps to install the libraries:

1. [Click here to download the ESPAsyncWebServer library](https://github.com/me-no-dev/ESPAsyncWebServer/archive/master.zip).
2. Unzip the .zip folder and you should get ESPAsyncWebServer-master folder
3. Rename your folder from  to ESPAsyncWebServer
4. Move the ESPAsyncWebServer folder to your Arduino IDE installation libraries folder

Alternatively, you can go to **Sketch** > **Include Library** > **.zip Library** and select the previously downloaded library.

**Installing the ESPAsyncTCP**

1. [Click here to download the ESPAsyncTCP library](https://github.com/me-no-dev/ESPAsyncTCP/archive/master.zip). You should have a .zip folder in your Downloads folder
2. Unzip the .zip folder and you should get ESPAsyncTCP-master folder
3. Rename your folder from  to ESPAsyncTCP
4. Move the ESPAsyncTCP folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

Alternatively, you can go to **Sketch** > **Include Library** > **.zip Library** and select the previously downloaded library.

**Installing BME280 libraries**

You can install these libraries through the Arduino IDE Libraries Manager. Go to **Sketch**> **Include Libraries** > **Manage Libraries**. Then, search for the libraries’ name to install them.

* [Adafruit\_BME280\_Library](https://github.com/adafruit/Adafruit_BME280_Library)
* [Adafruit\_Sensor](https://github.com/adafruit/Adafruit_Sensor)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **BME280** | **ESP8266** | | Vin | 3.3V | | GND | GND | | SCL | GPIO 5 | | SDA | GPIO 4 |  * [BME280 sensor module](https://makeradvisor.com/tools/bme280-sensor-module/) ([Guide for BME280](https://randomnerdtutorials.com/esp8266-bme280-arduino-ide/)) * 330 Ohms |

**Organizing Your Files**

|  |  |
| --- | --- |
| To build the web server you need three different files. The Arduino sketch, the HTML file and the CSS file. The HTML and CSS files should be saved inside a folder called **data** inside the Arduino sketch folder |  |

**Filesystem Uploader Plugin**

To upload files to the ESP8266 SPI Flash Filesystem (SPIFFS), we’ll use the Filesystem Uploader Plugin. Install the plugin in your Arduino IDE:

* [Install ESP8266 Filesystem Uploader in Arduino IDE](https://randomnerdtutorials.com/install-esp8266-filesystem-uploader-arduino-ide/)

**Creating the HTML File**

Create an *index.html* file

<!DOCTYPE html>

<html>

<head>

<title>ESP8266 Web Server</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" href="data:,">

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<h1>ESP8266 Web Server</h1>

<p>GPIO state<strong> %STATE%</strong></p>

<p>

<a href="/on"><button class="button">ON</button></a>

<a href="/off"><button class="button button2">OFF</button></a>

</p>

<p>

<span class="sensor-labels">Temperature</span>

<span id="temperature">%TEMPERATURE%</span>

<sup class="units">&deg;C</sup>

</p>

<p>

<span class="sensor-labels">Humidity</span>

<span id="humidity">%HUMIDITY%</span>

<sup class="units">&#37;</sup>

</p>

<p>

<span class="sensor-labels">Pressure</span>

<span id="pressure">%PRESSURE%</span>

<sup class="units">hPa</sup>

</p>

</body>

<script>

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("temperature").innerHTML = this.responseText;

}

};

xhttp.open("GET", "/temperature", true);

xhttp.send();

}, 10000 ) ;

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("humidity").innerHTML = this.responseText;

}

};

xhttp.open("GET", "/humidity", true);

xhttp.send();

}, 10000 ) ;

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("pressure").innerHTML = this.responseText;

}

};

xhttp.open("GET", "/pressure", true);

xhttp.send();

}, 10000 ) ;

</script>

</html>

## Creating the CSS File

Create the style.css file

html {

font-family: Arial;

display: inline-block;

margin: 0px auto;

text-align: center;

}

h1 {

color: #0F3376;

padding: 2vh;

}

p {

font-size: 1.5rem;

}

.button {

display: inline-block;

background-color: #008CBA;

border: none;

border-radius: 4px;

color: white;

padding: 16px 40px;

text-decoration: none;

font-size: 30px;

margin: 2px;

margin: 2px;

cursor: pointer;

}

.button2 {

background-color: #f44336;

}

.units {

font-size: 1.2rem;

}

.sensor-labels {

font-size: 1.5rem;

vertical-align:middle;

padding-bottom: 15px;

}

= = = = =

#include <ESP8266WiFi.h>

#include <ESPAsyncTCP.h>

#include <ESPAsyncWebServer.h>

#include <FS.h>

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BME280.h>

Adafruit\_BME280 bme; // I2C

//Adafruit\_BME280 bme(BME\_CS); // hardware SPI

//Adafruit\_BME280 bme(BME\_CS, BME\_MOSI, BME\_MISO, BME\_SCK); // software SPI

// Replace with your network credentials

const char\* ssid = "REPLACE\_WITH\_YOUR\_SSID";

const char\* password = "REPLACE\_WITH\_YOUR\_PASSWORD";

// Set LED GPIO

const int ledPin = 2;

// Stores LED state

String ledState;

// Create AsyncWebServer object on port 80

AsyncWebServer server(80);

String getTemperature() {

float temperature = bme.readTemperature();

// Read temperature as Fahrenheit (isFahrenheit = true)

//float t = dht.readTemperature(true);

Serial.println(temperature);

return String(temperature);

}

String getHumidity() {

float humidity = bme.readHumidity();

Serial.println(humidity);

return String(humidity);

}

String getPressure() {

float pressure = bme.readPressure()/ 100.0F;

Serial.println(pressure);

return String(pressure);

}

// Replaces placeholder with LED state value

String processor(const String& var){

Serial.println(var);

if(var == "STATE"){

if(digitalRead(ledPin)){

ledState = "ON";

}

else{

ledState = "OFF";

}

Serial.print(ledState);

return ledState;

}

else if (var == "TEMPERATURE"){

return getTemperature();

}

else if (var == "HUMIDITY"){

return getHumidity();

}

else if (var == "PRESSURE"){

return getPressure();

}

}

void setup(){

// Serial port for debugging purposes

Serial.begin(115200);

pinMode(ledPin, OUTPUT);

// Initialize the sensor

if (!bme.begin(0x76)) {

Serial.println("Could not find a valid BME280 sensor, check wiring!");

while (1);

}

// Initialize SPIFFS

if(!SPIFFS.begin()){

Serial.println("An Error has occurred while mounting SPIFFS");

return;

}

// Connect to Wi-Fi

WiFi.begin(ssid, password);

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

delay(1000);

Serial.println("Connecting to WiFi..");

}

// Print ESP32 Local IP Address

Serial.println(WiFi.localIP());

// Route for root / web page

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

request->send(SPIFFS, "/index.html", String(), false, processor);

});

// Route to load style.css file

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

request->send(SPIFFS, "/style.css", "text/css");

});

// Route to set GPIO to HIGH

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

digitalWrite(ledPin, HIGH);

request->send(SPIFFS, "/index.html", String(), false, processor);

});

// Route to set GPIO to LOW

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

digitalWrite(ledPin, LOW);

request->send(SPIFFS, "/index.html", String(), false, processor);

});

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

request->send\_P(200, "text/plain", getTemperature().c\_str());

});

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

request->send\_P(200, "text/plain", getHumidity().c\_str());

});

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

request->send\_P(200, "text/plain", getPressure().c\_str());

});

// Start server

server.begin();

}

void loop(){

}