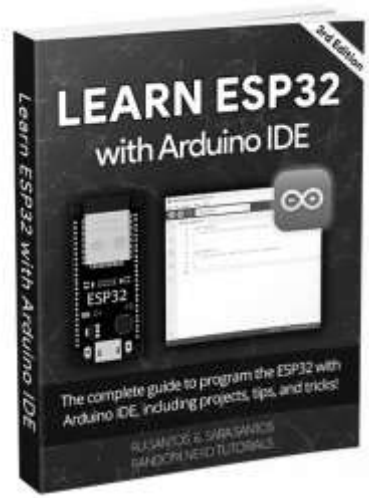
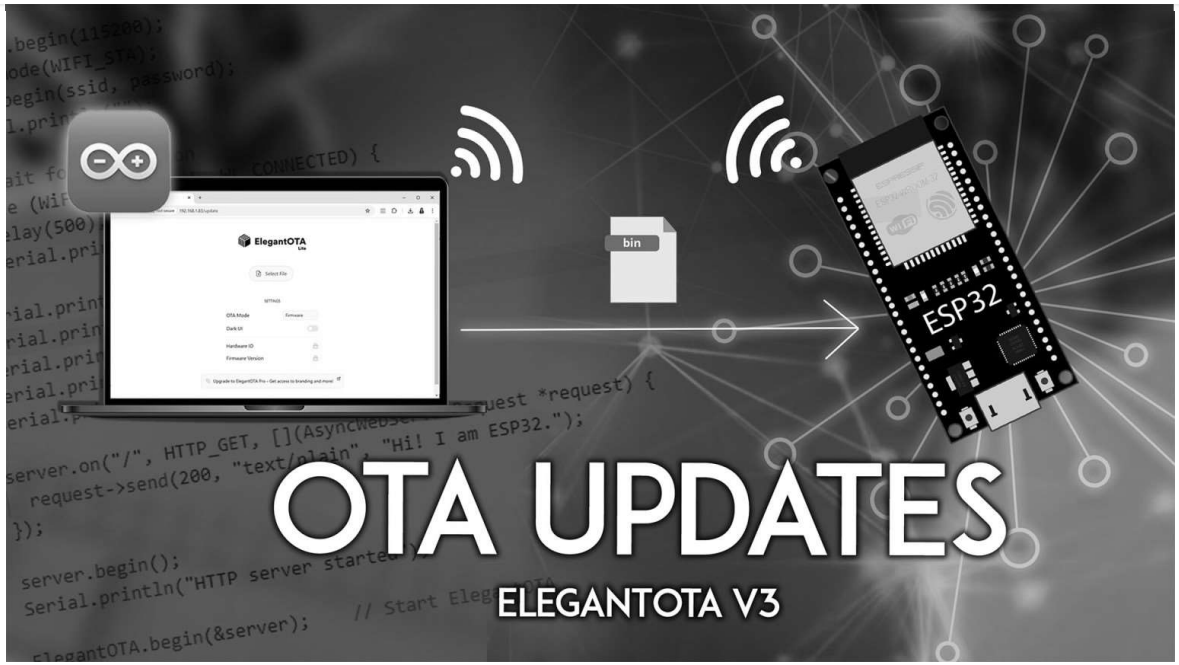
ESP32 OTA (Over-the-Air) Updates – ElegantOTA Library with Arduino IDE

This tutorial shows how to do OTA (over-the-air) updates to your ESP32 boards using the ElegantOTA library (V3 version) with Arduino IDE. This library sets up a web server that lets you update the firmware (a new sketch) on your board wirelessly. This way, you don't need a connection between the ESP32 and your computer to upload a new sketch. This library also allows uploading files to the filesystem (LittleFS or SPIFFS) wirelessly.





By the end of this tutorial, you'll be able to easily add OTA capabilities to your web server projects with the ESP32 to upload new firmware and files to the filesystem wirelessly in the future.

Table Of Contents

Throughout this tutorial, we'll cover:

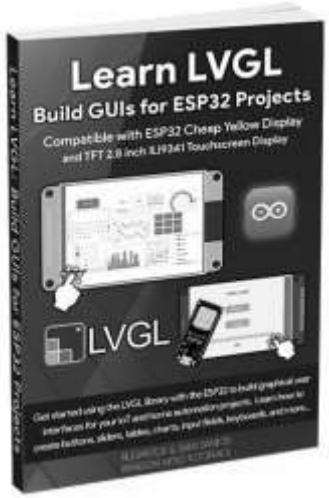
- Add the ElegantOTA feature to your web server
- Upload new firmware via OTA to the ESP32 board
- Upload files to LittleFS via OTA to the ESP32 board

We recommend that you follow all the tutorial steps to understand how the ElegantOTA library works and how you can use it in your projects. To demonstrate how to do this, we'll upload files to build different web server projects.

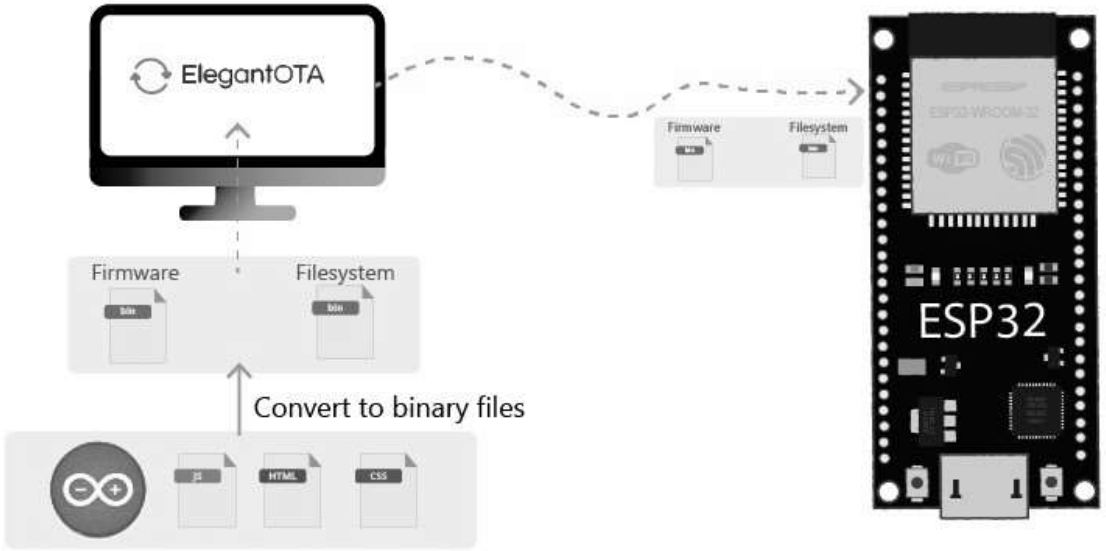
ESP32 OTA (Over-the-Air) Programming

OTA (Over-the-Air) update is the process of loading new firmware to the ESP32 board using a Wi-Fi connection rather than a serial communication. This functionality is extremely useful in case of no physical access to the ESP32 board. You don't need a connection between your computer and the board to upload new code.

Learn Raspberry Pi Pico/Pico W with MicroPython » The complete getting started guide to get the most out of the the Raspberry Pi Pico/Pico W (RP2040) microcontroller board using MicroPython programming language.



🔥 **Learn LVGL: Build GUIs for ESP32 Projects »** Learn how to build Graphical



User Interfaces (GUIs) for ESP32 Projects using LVGL (Light Versatile Graphics Library) with the Arduino IDE.

There are different ways to perform OTA updates. In this tutorial, we'll cover how to do that using the ElegantOTA library (version V3 — this is the successor of the deprecated AsyncElegantOTA library). In our opinion, this is one of the best and easiest ways to perform OTA updates.

The ElegantOTA library creates a web server that you can access on your local network to upload new firmware or files to the filesystem (SPIFFS or LittleFS). The files you upload should be in *.bin* format. We'll show you later in this tutorial how to convert your files to *.bin* format.

The only disadvantage of OTA programming is that you need to add the code for OTA in every sketch you upload so that you're able to use OTA in the future. In case of the ElegantOTA library, it consists of just three lines of code.

ElegantOTA Library

Here are some great features of this library:

- It is compatible with the built-in `WebServer.h` library and with several forks of the `ESPAsyncWebServer` library.
- You just need to add three lines of code to add OTA capabilities to your “regular” web server;
- It allows you to update not only new firmware to the board but also files to the ESP32 filesystem (LittleFS or SPIFFS);
- It provides a beautiful and modern web server interface;
- It is available as a pro paid version that adds more features.

OTA Updates with the ElegantOTA Library – Quick Summary

To add OTA capabilities to your projects using the ElegantOTA library, follow these steps:

1) Install the ElegantOTA, AsyncTCP, and ESPAsyncWebServer libraries;

2) Include ElegantOTA library at the top of the Arduino sketch:

```
#include <ElegantOTA.h>;
```

3) Add the following line in the setup before `server.begin();`

```
ElegantOTA.begin(&server);
```

4) In the `loop()`, add the following line:

```
ElegantOTA.loop();
```

5) Open your browser and go to `http://<IPAddress>/update`, where `<IPAddress>` is your ESP32 IP address to access the web server page for the OTA updates.

Continue reading the tutorial for more detailed steps.

How does OTA Web Updater Work?

The first sketch must be uploaded using a serial connection. This sketch should include the code to set up the OTA Web Updater, allowing you to upload new sketches through a web browser.

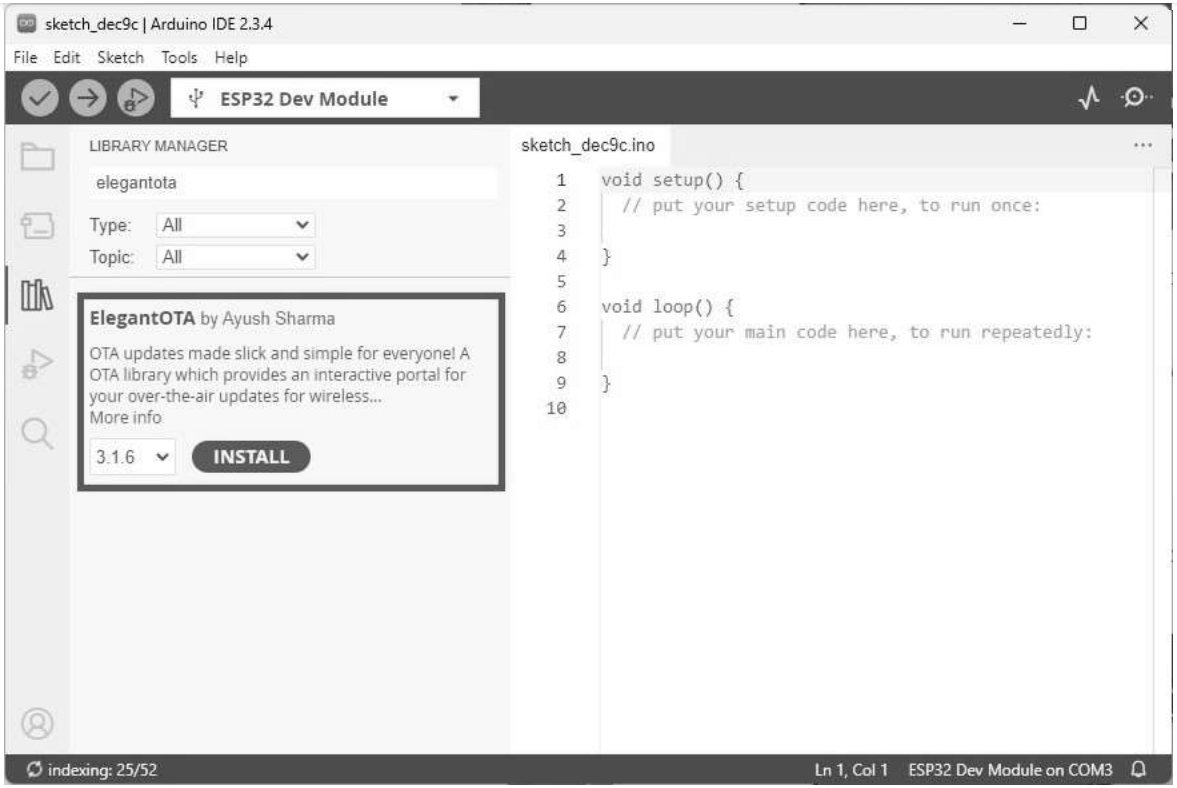
The OTA Web Updater creates a web server where you can upload sketches wirelessly.

If your code doesn't include an OTA routine, you won't be able to use the web server to upload new sketches wirelessly.

After that, make sure every sketch you upload includes OTA routines so you can continue updating the board wirelessly in the future.

Installing the ElegantOTA Library

Having the ESP32 Board installed in your Arduino IDE, you also need to install the ElegantOTA library in the Arduino IDE go to **Sketch** > **Include Library** > **Manage Libraries**, search for **ElegantOTA** and install the *ElegantOTA library by Ayush Sharma*.



Enabling Async Mode

For the ElegantOTA library to work in async mode (with the ESPAsyncWebServer library), you need to do the following procedure.

1) Go to your Arduino *libraries* directory.

2) Open the *ElegantOTA* folder and then open the *src* folder.

3) Locate the `ELEGANTOTA_USE_ASYNC_WEBSERVER` macro in the

ElegantOTA.h file, and set it to `1` as follows:

```
#define ELEGANTOTA_USE_ASYNC_WEBSERVER 1
```

4) Save the changes to the *ElegantOTA.h* file.

5) Now you can use the ElegantOTA in async mode for your OTA updates and with the ESPAsyncWebServer library.

Install AsyncTCP and ESPAsyncWebServer Libraries

To test the examples in this tutorial, you also need to install the AsyncTCP and the ESPAsyncWebServer libraries.

Note: the ElegantOTA library works with different forked versions of the ESPAsyncWebServer and the AsyncTCP web server libraries, so you don't necessarily need to use the same ones that we're using.

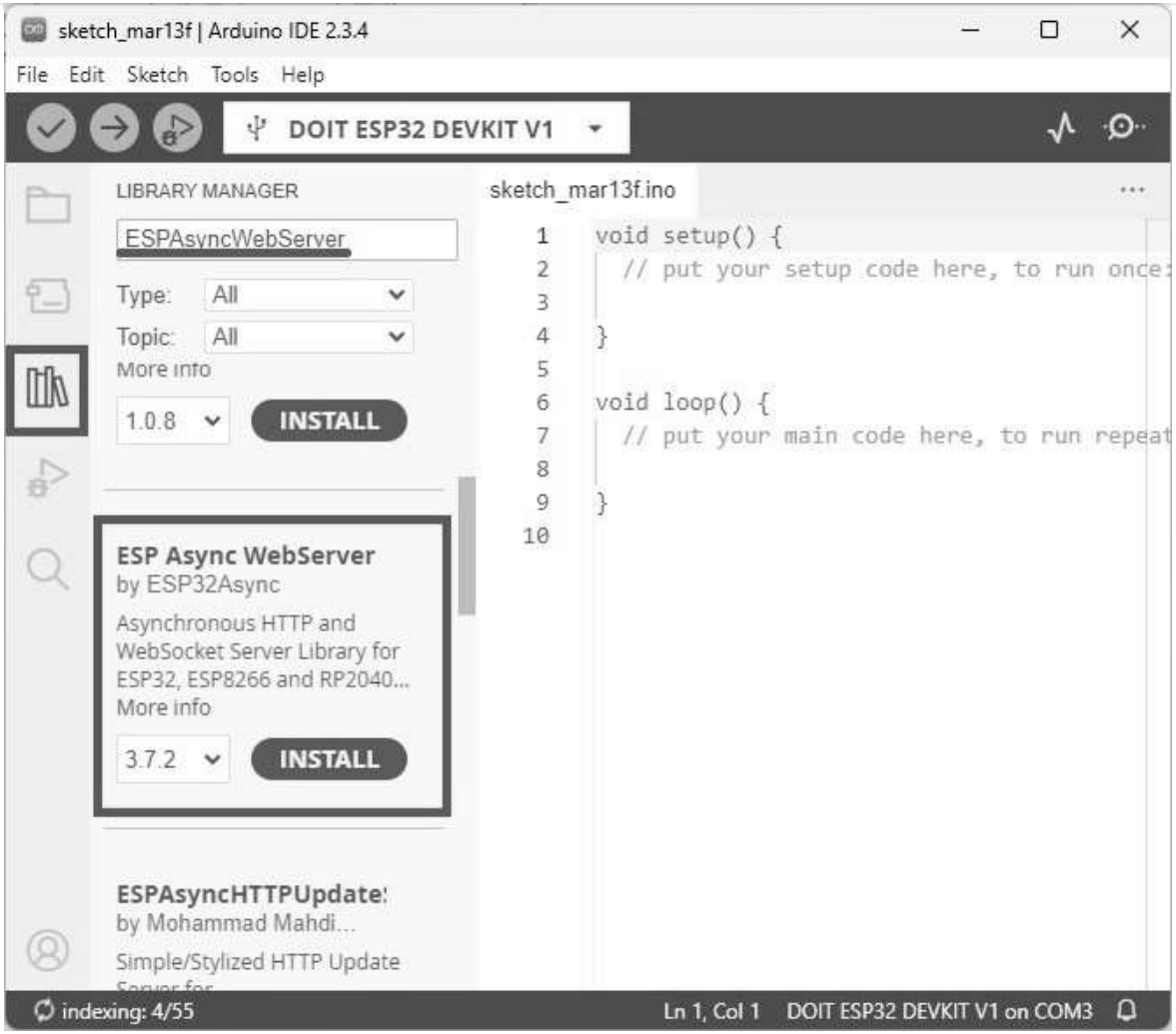
Async Web Server Libraries

We'll build the web server using the following libraries:

- ESPAsyncWebServer
- AsyncTCP

You can install these libraries in the Arduino Library Manager. Open the Library Manager by clicking the Library icon at the left sidebar.

Search for `ESPAsyncWebServer` and install the **ESPAsyncWebServer by ESP32Async**.



Then, install the AsyncTCP library. Search for `AsyncTCP` and install the **AsyncTCP by ESP32Async**.

ElegantOTA ESP32 Basic Example

Let’s start with a basic example (based on one of the library’s examples). The following code creates a simple web server with the ESP32. The

root URL displays some text, and the **/update** URL displays the interface to update the firmware and the filesystem.

Copy the following code to your Arduino IDE.

```
/*********
  Rui Santos & Sara Santos - Random Nerd Tutorials
  Complete project details at https://RandomNerdTutorials
  Permission is hereby granted, free of charge, to any pe
*********/

#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <ElegantOTA.h>

const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

AsyncWebServer server(80);

void setup(void) {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
```

```
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

View raw code

Insert your network credentials and the code should work straight away.

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

How Does the Code Work?

Start by including the required libraries.

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

Insert your network credentials in the following variables so that the ESP32 can connect to the Wi-Fi on your local network.

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

Create an `AsyncWebServer` object on port 80:

```
AsyncWebServer server(80);
```

In the `setup()`, initialize the Serial Monitor:

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

Initialize Wi-Fi:

```
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
```

Then, handle the client requests. The following lines, send some text `Hi! I am ESP32.` when you access the root (/) URL:

```
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *reques
    request->send(200, "text/plain", "Hi! I am ESP32.");
  });
```

If your web server needs to handle more requests you can add them

(we'll show you in the next example).

Initialize the server:

```
server.begin();
```

Then, add the next line to start ElegantOTA:

```
ElegantOTA.begin(&server);    // Start ElegantOTA
```

In the `loop()`, add the following line:

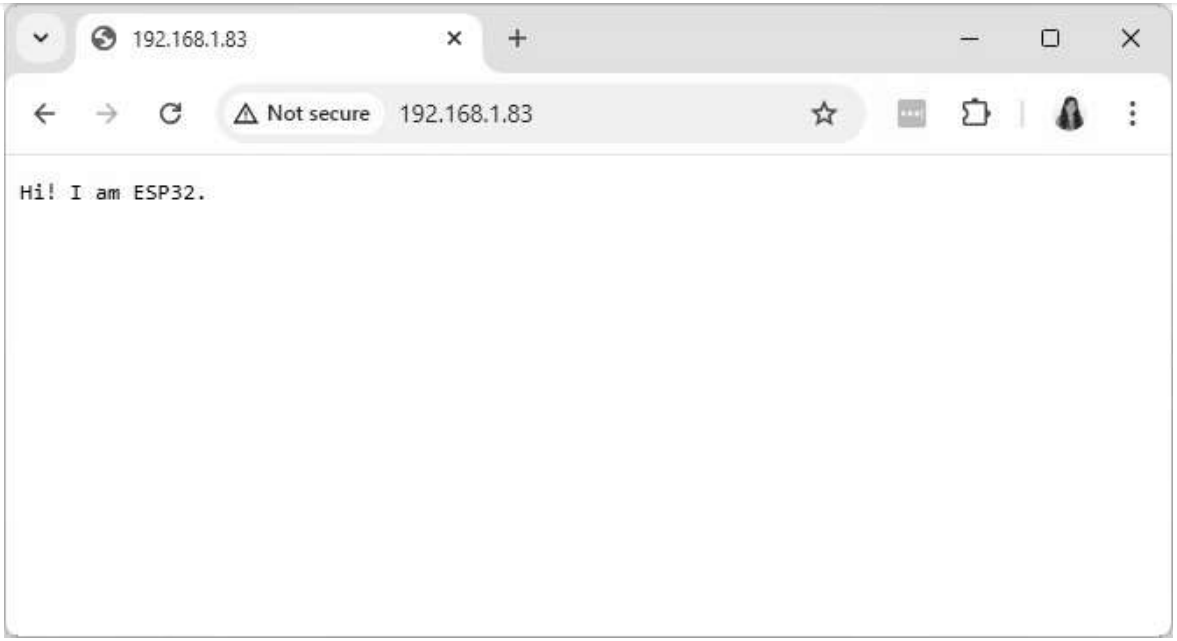
```
ElegantOTA.loop();
```

Accessing the Web Server

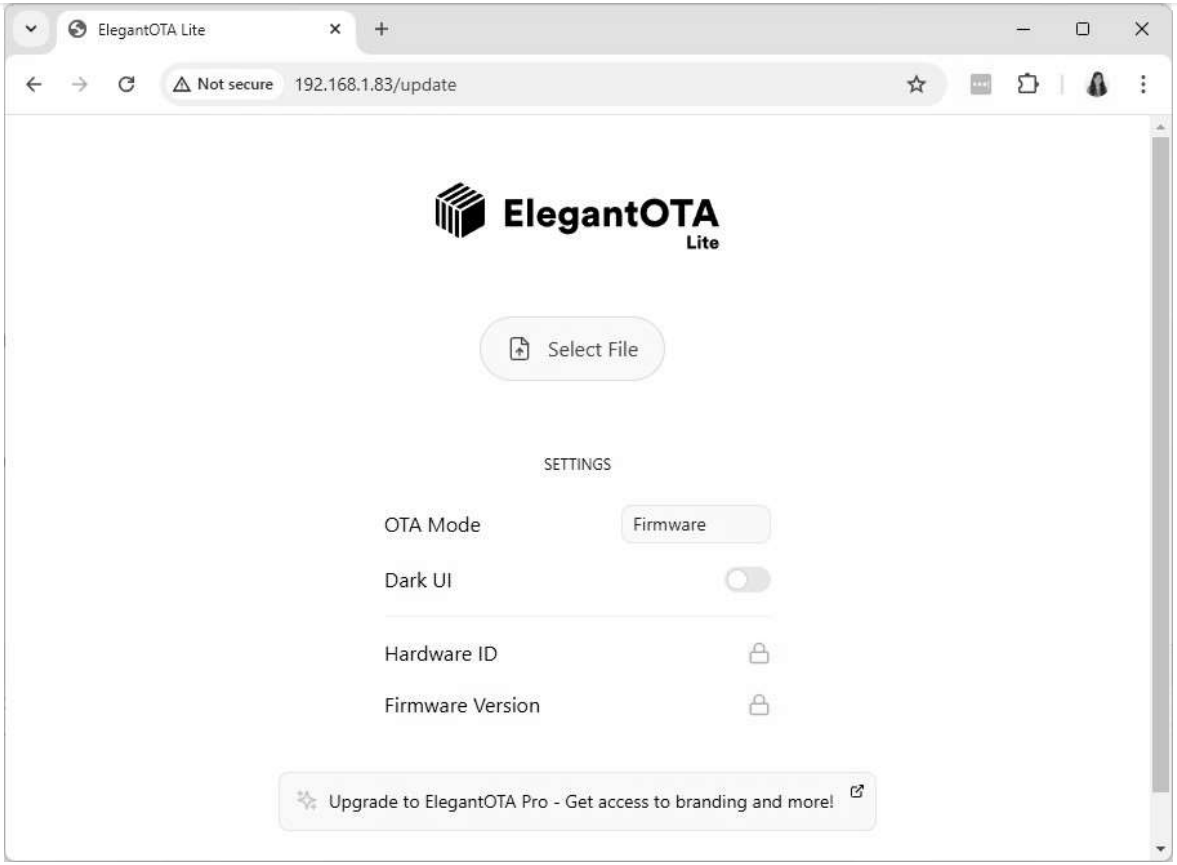
After uploading code to the board, open the Serial Monitor at a baud rate of 115200. Press the ESP32 on-board RST button. It should display the ESP IP address as follows (yours may be different):

Note: when uploading the code, if you have the Partition Scheme options for your particular board in the Tools menu, make sure to select a Partition Scheme with space for the filesystem. For example: *Partition Scheme: 16M Flash (2MB APP/12.5MB FATFS)*. If you don't have that option for the particular board you selected, don't worry about it.

In your local network, open your browser and type the ESP32 IP address. You should get access the root (/) web page with some text displayed.



Now, if you want to modify your web server code via OTA, go to the ESP IP address followed by `/update`. The following web page should load.

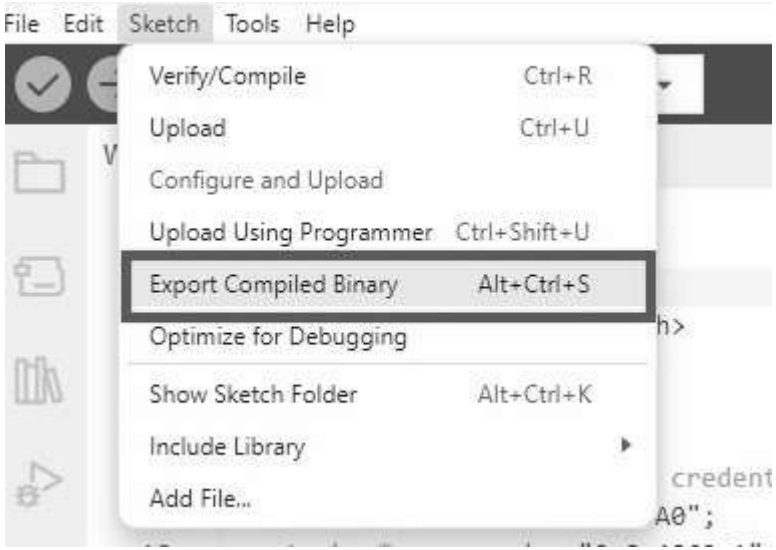


To upload new code to your board or files to the filesystem, you need to upload them in *.bin* format. Follow the next section to learn how to generate a *.bin* file from your sketch in Arduino IDE.

Upload New Code to the ESP32 via OTA

Every file that you upload via OTA should be in *.bin* format. You can generate a *.bin* file from your sketch using the Arduino IDE.

With your sketch opened, you just need to select the ESP32 board you're using in **Tools** > **Board**, and then go to **Sketch** > **Export Compiled Binary**. A *.bin* file will be generated from your sketch.



The generated file will be saved under your project folder inside a series of other folders. The file with the *.ino.bin* extension is the one you should upload to your board using the ElegantOTA web page.

Upload a New Web Server Sketch via OTA – Example

Imagine that after uploading the previous sketch, you want to upload a new one that allows you to control an LED via a web interface like this project. Here's the steps you need to follow:

1. Copy the following code to your Arduino IDE. Don't forget to insert your network credentials.

```
/*********
  Rui Santos & Sara Santos - Random Nerd Tutorials
  Complete project details at https://RandomNerdTutorials
  Permission is hereby granted, free of charge, to any pe
*********/

// Import required libraries
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <ElegantOTA.h>

// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

bool ledState = 0;
```

```
const int ledPin = 2;

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);
AsyncWebSocket ws("/ws");

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
```

View raw code

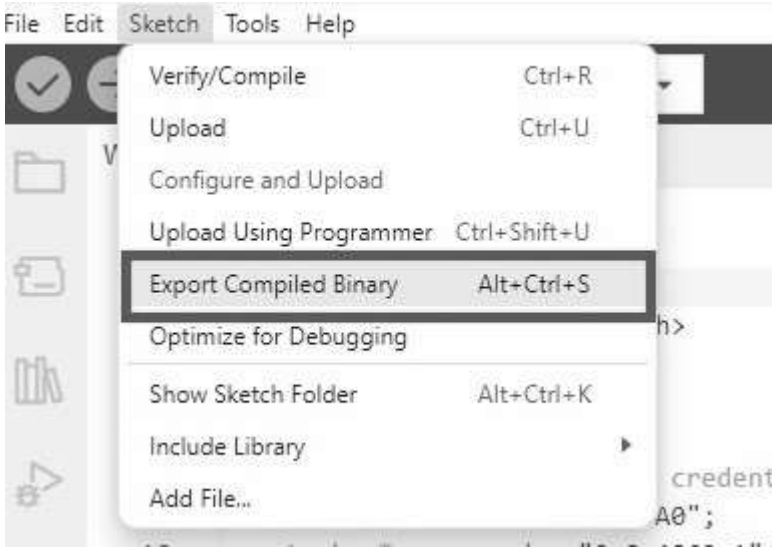
This is the same code used in this project, but it contains the needed lines of code to handle ElegantOTA:

```
#include <ElegantOTA.h>


ElegantOTA.begin(&server);


ElegantOTA.loop();
```

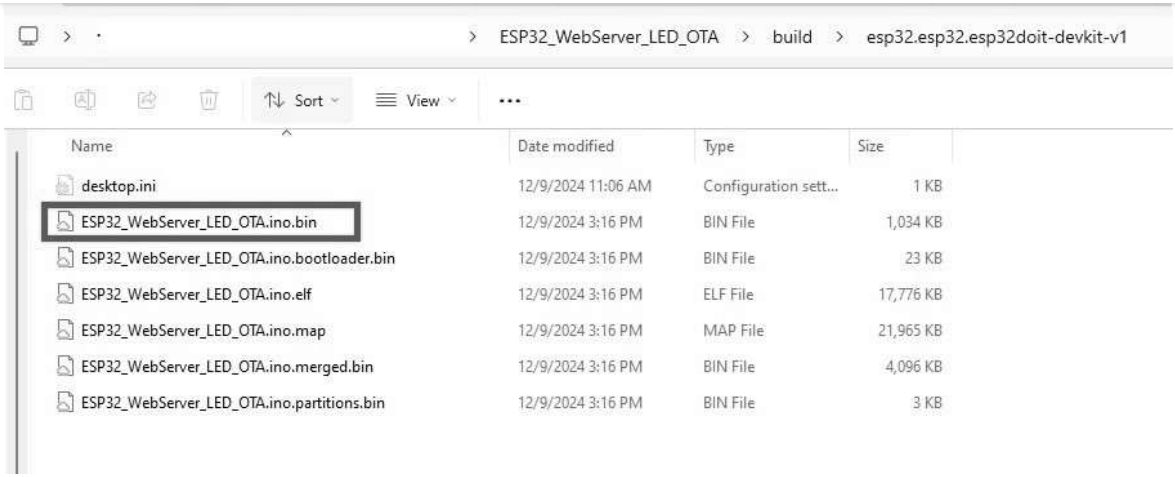
2. Save your sketch: **File** > **Save** and give it a name. For example:

Web_Server_LED_OTA_ESP32.

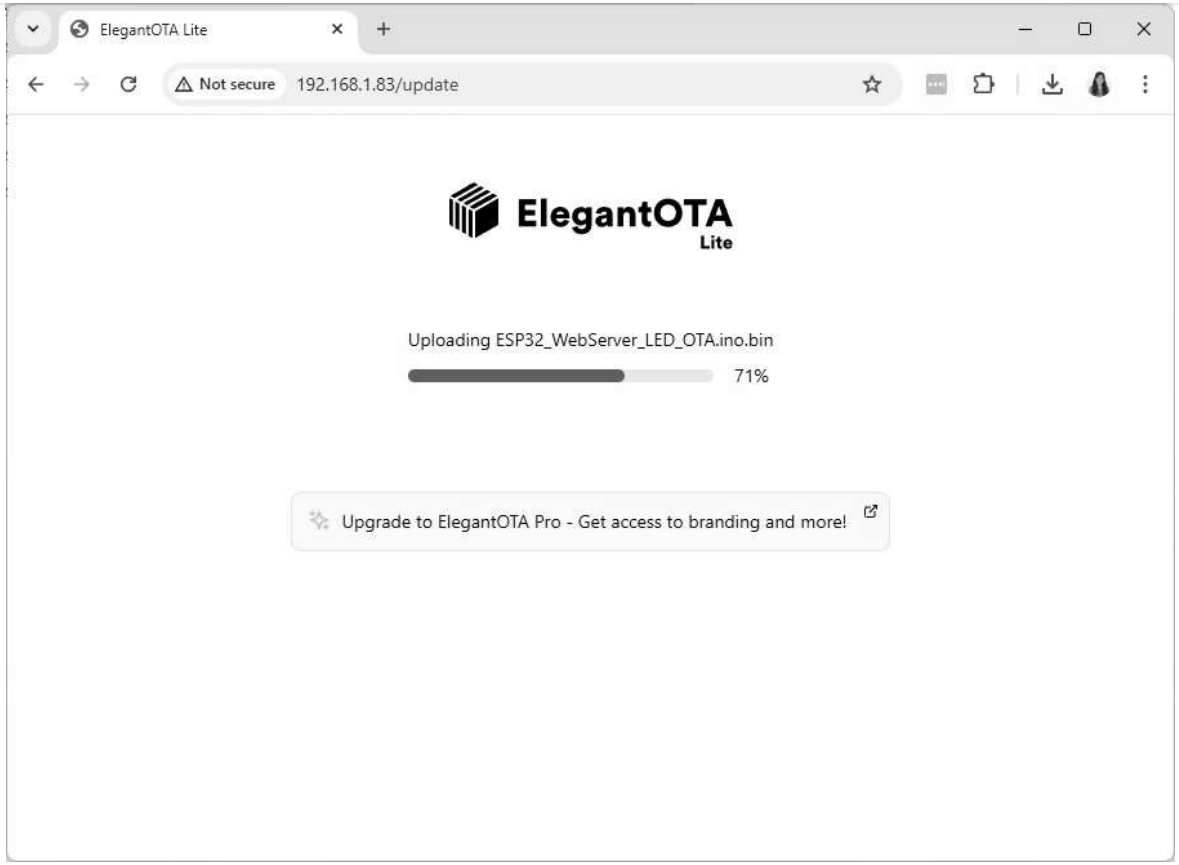
3. Generate a *.bin* file from your sketch. First, select the board model you're using in **Tools** > **Board**. Then, go to **Sketch** > **Export Compiled Binary**.



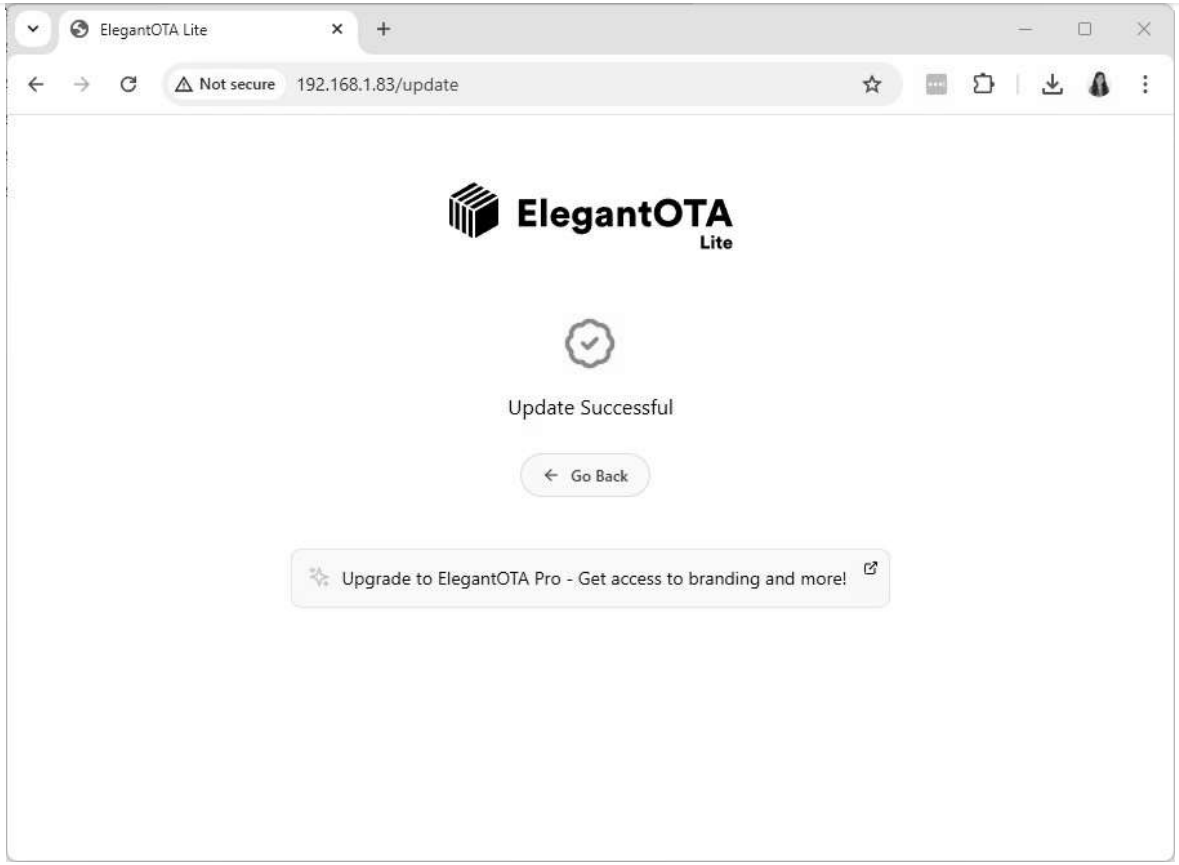
Go to your sketch folder. You should have a *build* folder. Inside that folder, you'll have another folder related to your board model. Open that folder. There'll be several files. You should upload the file with the *ino.bin* extension.



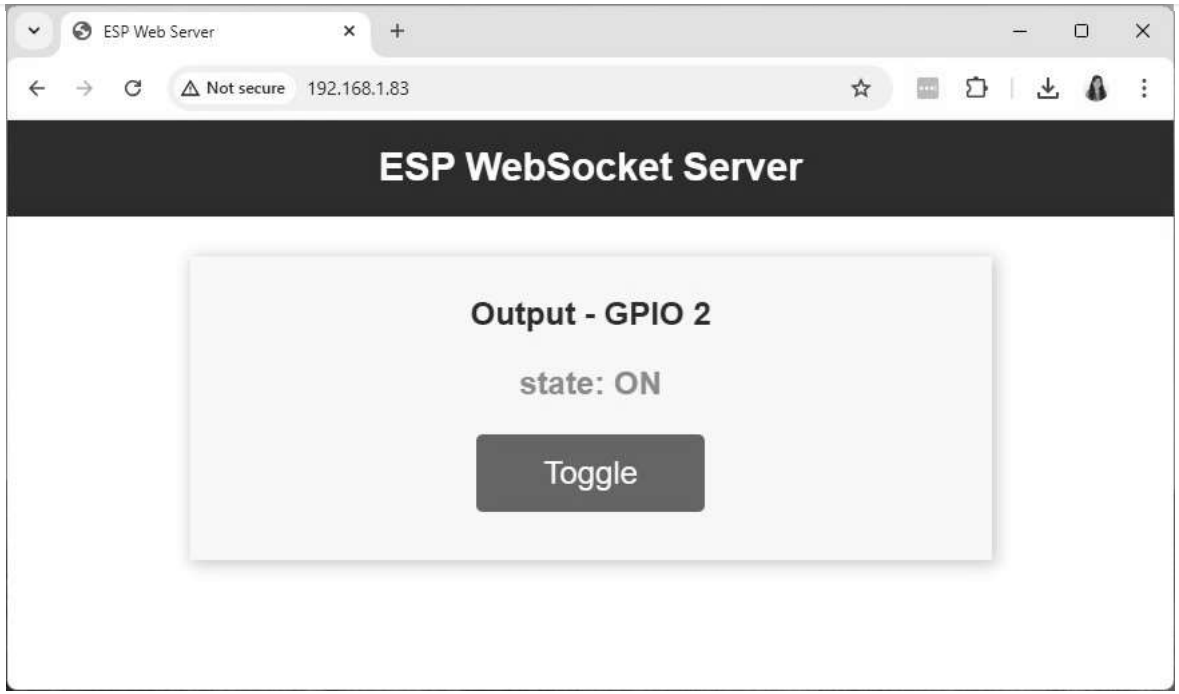
4. Now, you need to upload that file using the ElegantOTA page. Go to your ESP IP address followed by `/update`. Make sure you have the **firmware** option selected. Click on **Choose File** and select the *.ino.bin* file you've just generated.



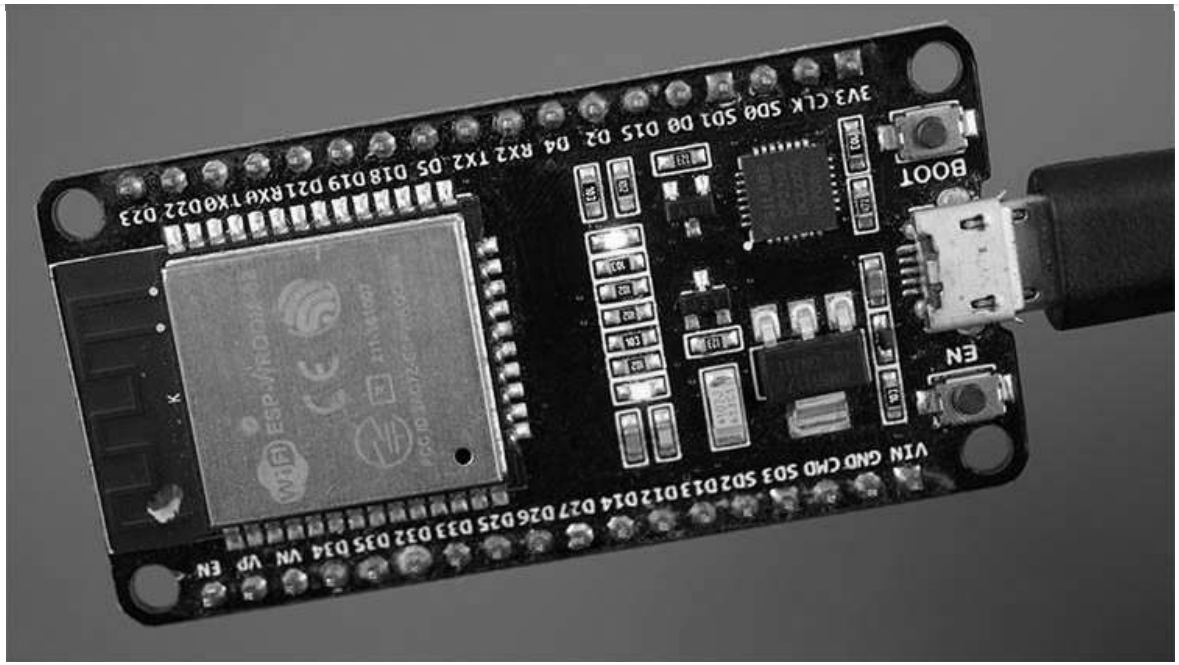
5. After a few seconds, you should get a success message. Then, click on the **Back** button.



6. Now, you can go to the root (/) URL to access the new web server. This is the page you should see when you access the ESP IP address on the root (/) URL.



You can click on the button to turn the ESP32 on-board LED on and off.



Because we've also added OTA capabilities to this new web server, we can upload a new sketch in the future if needed. You just need to go to the ESP32 IP address followed by `/update`.

Congratulations, you've uploaded new code to your ESP32 via Wi-Fi using ElegantOTA.

Continue reading if you want to learn how to upload files to the ESP32 filesystem (LittleFS) using the ElegantOTA library.

Upload Files to Filesystem via

OTA to the ESP32

In this section you'll learn to upload files to the ESP32 filesystem (LittleFS) using the ElegantOTA library.

ESP32 Filesystem Uploader Plugin

Before proceeding, you need to have the ESP32 Filesystem Uploader Plugin installed in your Arduino IDE. Follow the next tutorial before proceeding:

- Arduino IDE 2: Install ESP32 LittleFS Uploader (Upload Files to the Filesystem)

Web Server with Files from LittleFS

Imagine the scenario that you need to upload files to the ESP32 filesystem, for example: configuration files; HTML, CSS and JavaScript files to update the web server page; or any other file that you may want to save in the LittleFS filesystem via OTA.

To show you how to do this, we'll create a new web server that serves files from LittleFS: HTML, and CSS files to build a web page that controls the ESP32 built-in LED remotely using a different interface.

Copy the following code to your Arduino IDE.

```
/*********
  Rui Santos & Sara Santos - Random Nerd Tutorials
  Complete project details at https://RandomNerdTutorials
  Permission is hereby granted, free of charge, to any pe
*********/

#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "LittleFS.h"
#include <ElegantOTA.h>

// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

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

// Set LED GPIO
const int ledPin = 2;
```

```
// Stores LED state
String ledState;

// Initialize LittleFS
```

View raw code

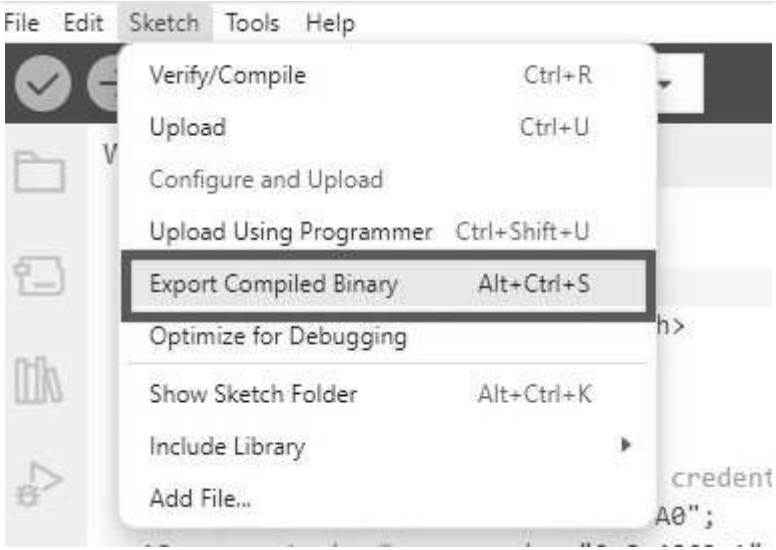
Note: this example is covered and explained in our eBook: Build Web Severs with the ESP32 and ESP8266.

Insert your network credentials in the following variables and save the code.

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

Update Firmware

Create a *.ino.bin* file from this sketch as shown previously (this sketch also includes the needed lines of code to provide OTA capabilities).



Go to the ESP32 IP address followed by `/update` and upload the new firmware.

Next, we'll see how to upload the files.

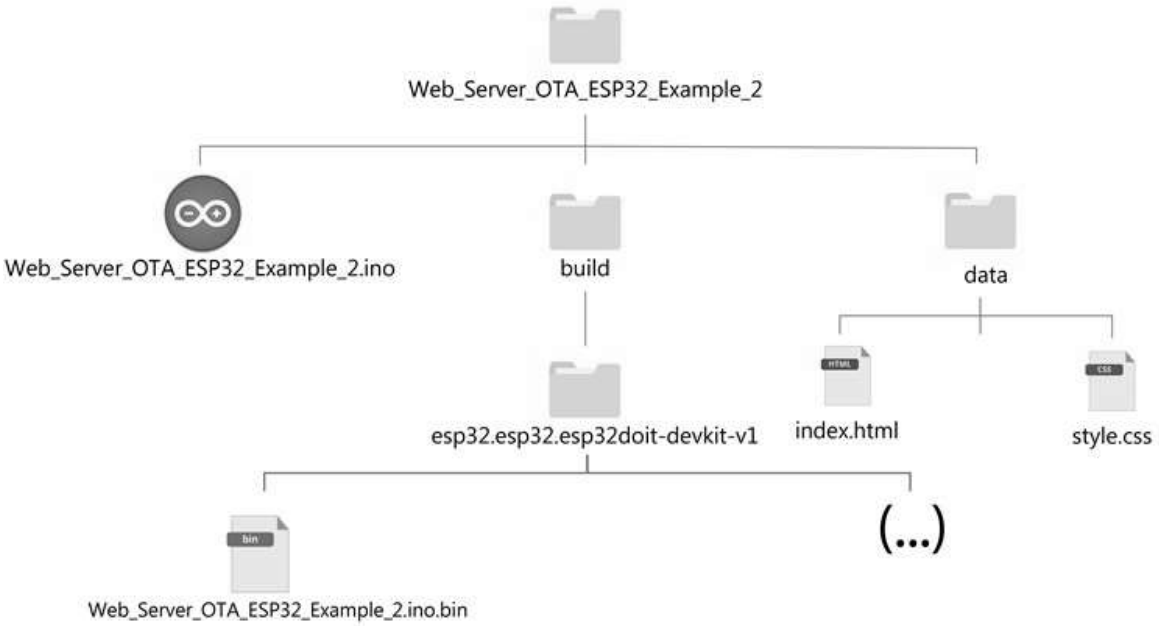
Update the Filesystem

Under the project folder, create a folder called ***data*** and move the following HTML and CSS files (click on the links to download the files):

- HTML file: *index.html*
- CSS file: *style.css*
- **Click here to donwload all project files**

To find your project folder, you can simply go to **Sketch** > **Show Sketch Folder**.

This is the folder structure of your project and where the data folder should be located:



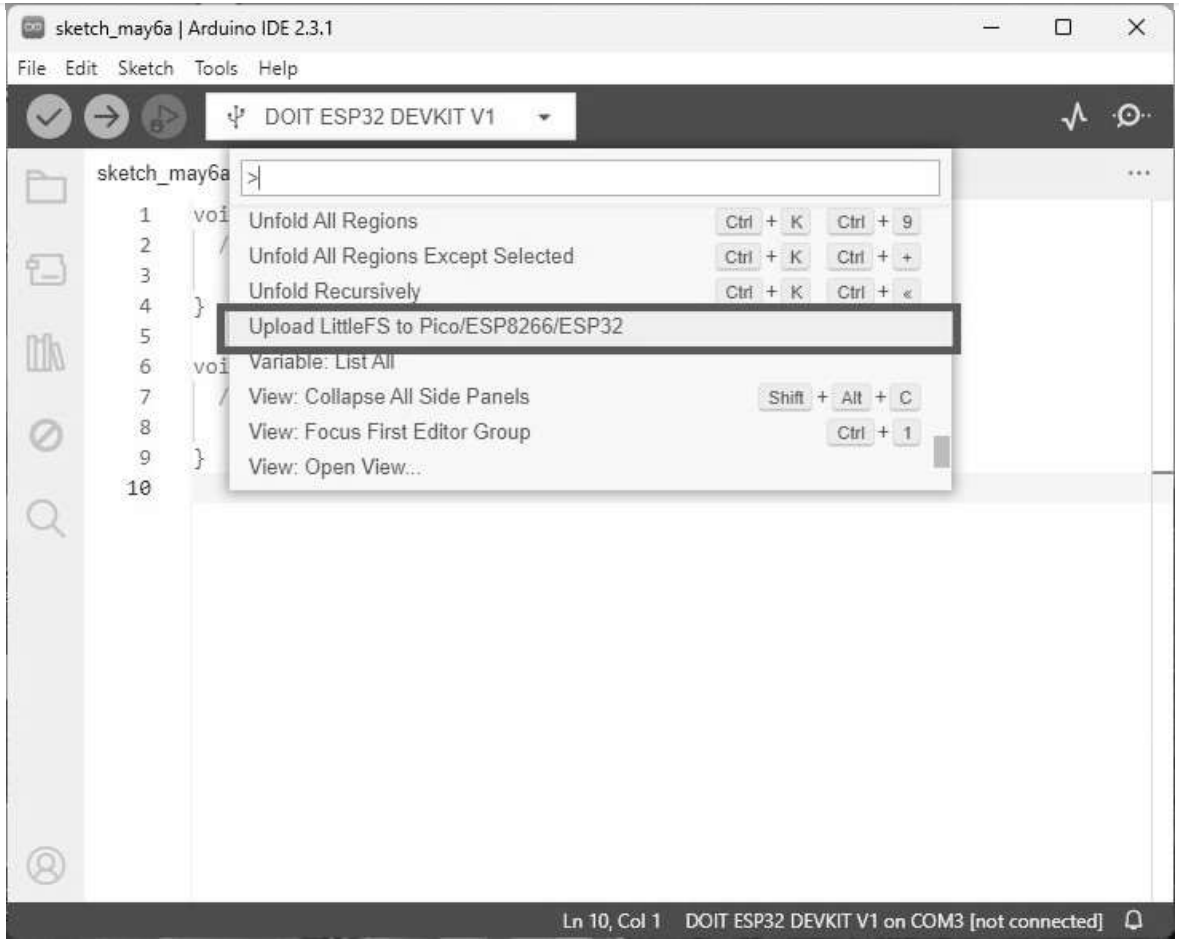
- Web_Server_OTA_ESP32_Example_2
  - **data**
    - ***style.css***
    - ***index.html***
  - *Web_Server_OTA_ESP32_Example_2.ino*
  - build
    - esp32.es32.esp32doit-devkit-v1 (or similar depending on

the selected board)

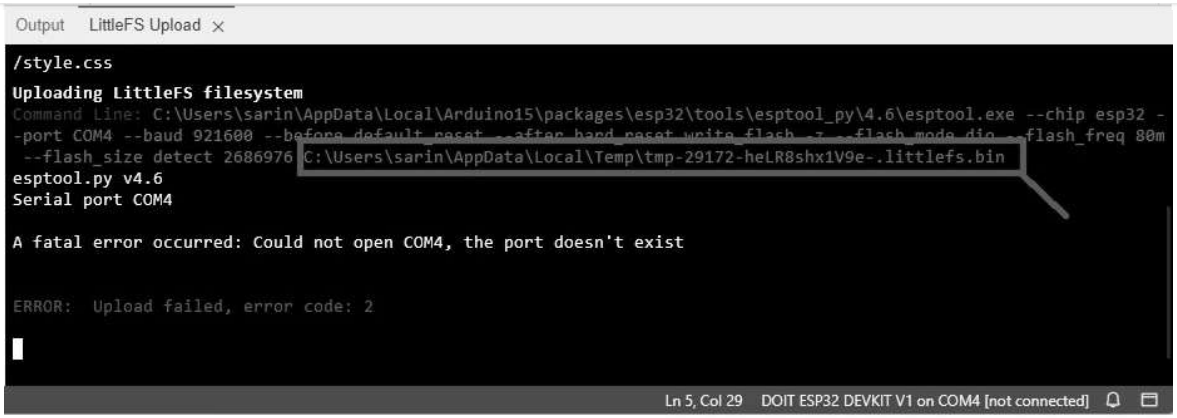
- *Web_Server_OTA_ESP32_Example_2.ino.bin*

After this, with the **ESP32 disconnected from your computer** (that's the whole purpose of OTA), let's pretend we'll upload the files to the filesystem. Press [**Ctrl**] + [**Shift**] + [**P**] to open the command palette. An instruction called '**Upload Little FS to Pico/ESP8266/ESP32**' should be there (just scroll down or search for the name of the instruction).

If you don't have that option, it means you don't have the Filesystem Uploader Plugin installed in your Arduino IDE. Follow the next tutorial to install it: Arduino IDE 2: Install ESP32 LittleFS Uploader (Upload Files to the Filesystem).



You'll get an error because there isn't any ESP32 board connected to your computer – don't worry. It will still create a *.bin* file from the *data* folder.

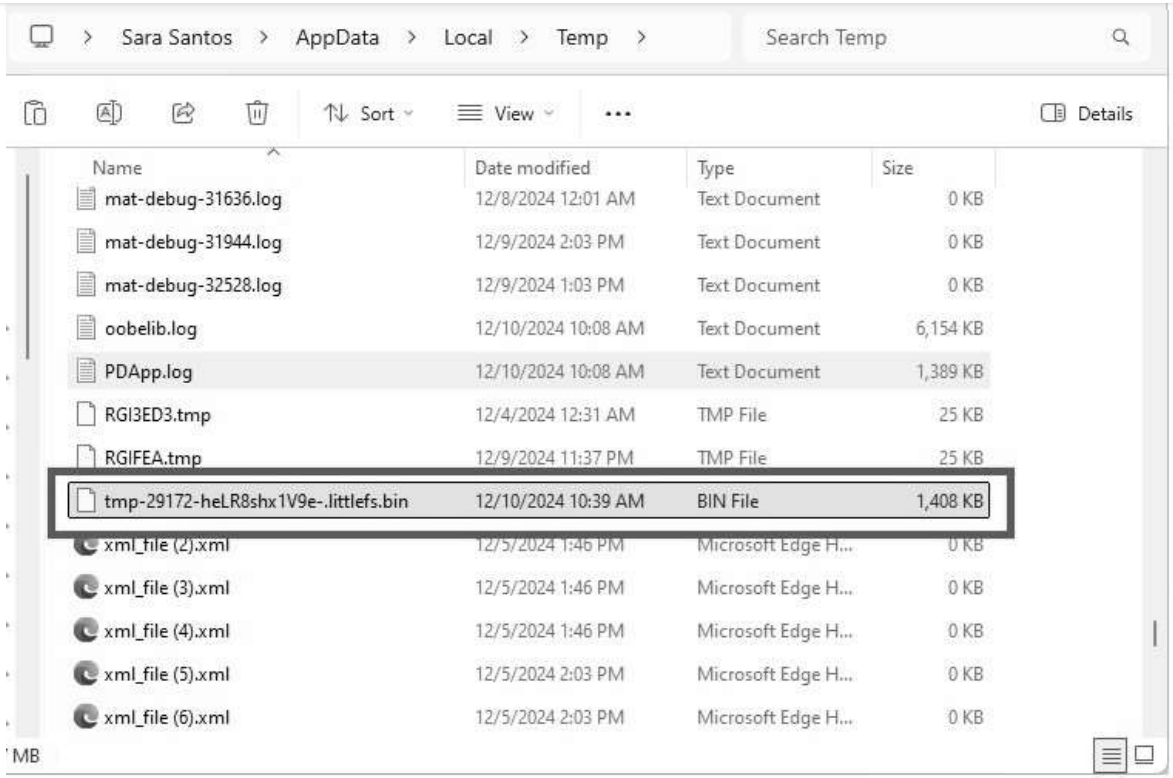


On the debugging window you'll see the *.littlefs.bin* file location. That's that file that you should upload (in our case the file is called *tmp-29172-heLR8chx1V9e-.littlefs.bin*.

In our case, this is the path where that file is located:

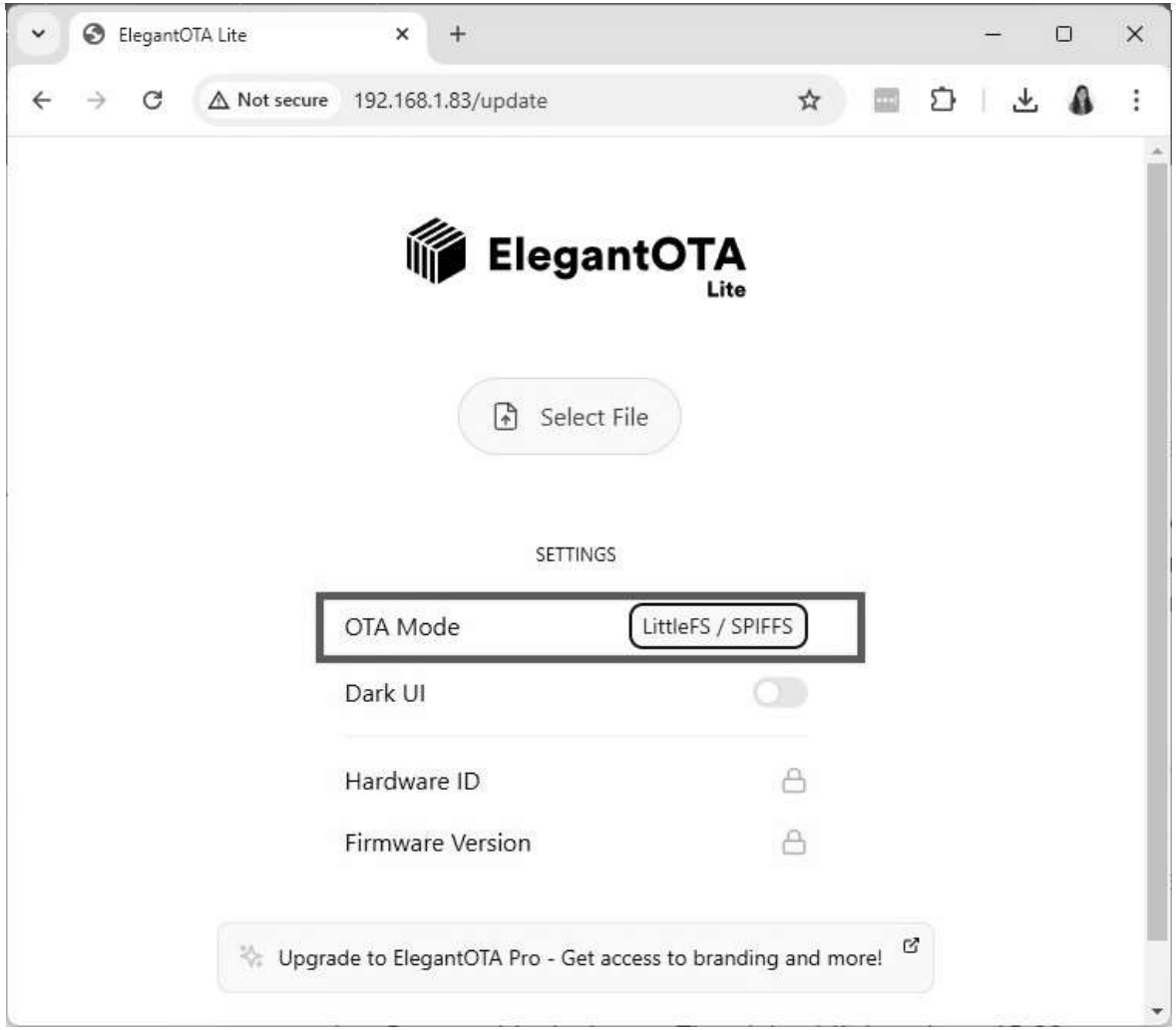
```
C:\Users\sarin\AppData\Local\Temp\tmp-29172-
heLR8chx1V9e-.littlefs.bin.
```

Find that file in your computer.



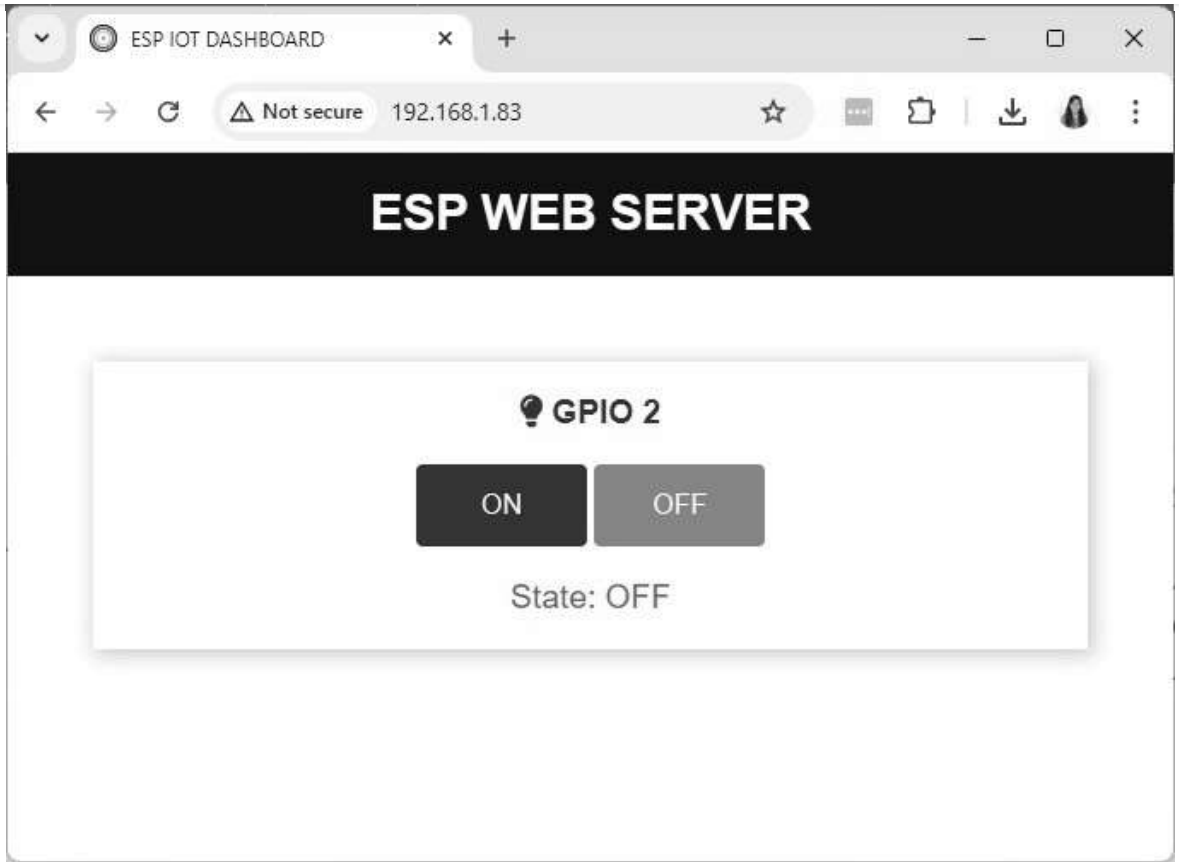
To make things easier, you can copy that file to your project folder.

Now that we have a *.litlefs.bin* file from the *data* folder, we can upload that file. Go to your ESP32 IP address followed by `/update`. Make sure you have the **Filesystem** option selected in the OTA Mode option.



Then, select the file with the *.littlefs.bin* extension.

After successfully uploading, click the **Back** button. And go to the root (/) URL again. You should get access to the following web page that controls the ESP32 built-in LED with two buttons.



If you need to update something on your project, you just need to go to your ESP32 IP address followed by `/update`.

Congratulations! You've successfully uploaded files to the ESP32 filesystem using the ElegantOTA library.

Wrapping Up

In this tutorial you've learned how to add OTA capabilities to your Async Web Server projects using the ElegantOTA library. This library is straightforward to use—you just need to add three lines of code to your project.

Additionally, this library also allows you to upload new firmware or files to the LittleFS filesystem effortlessly using a web page.

We hope you've found this tutorial useful.

Learn more about the ESP32 with our resources:

- Learn ESP32 with Arduino IDE (eBook)
- Build ESP32 Web Servers with Arduino IDE (eBook)
- More ESP32 Projects and Tutorials…

Thanks for reading.