

## Uploading files to the server

There are scenarios where you may want to upload new files to the server from within a browser, without having to connect to the ESP8266 over USB in order to flash a new SPIFFS image.

In this chapter, I'll show you how to use HTML forms and POST requests to upload or edit files to our little ESP server.

### Client: HTML form

The easiest way to upload files is by using an HTML form, just like in the first server examples, where we used forms to turn on/off LEDs, and to send the login credentials back to the server. If you choose a file input, you automatically get a file picker, and the browser will send the right POST request to the server, with the file attached.

```
<form method="post" enctype="multipart/form-data">
  <input type="file" name="name">
  <input class="button" type="submit" value="Upload">
</form>
```

### Server

In the ESP code, we have to add a handler to our server that handles POST requests to the */upload* URI. When it receives a POST request, it sends a status 200 (OK) back to the client to start receiving the file, and then write it to the SPIFFS. When the file is uploaded successfully, it redirects the client to a success page. The relevant new code is found in the `setup` and the `handleFileUpload` function.

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266mDNS.h>
#include <ESP8266WebServer.h>
#include <FS.h> // Include the SPIFFS library

ESP8266WiFiMulti wifiMulti; // Create an instance of the ESP8266WiFiMulti class, called
                              'wifiMulti'

ESP8266WebServer server(80); // Create a webserver object that listens for HTTP request on
                              port 80

File fsUploadFile; // a File object to temporarily store the received file

String getContentType(String filename); // convert the file extension to the MIME type
bool handleFileRead(String path); // send the right file to the client (if it exists)
void handleFileUpload(); // upload a new file to the SPIFFS

void setup() {
  Serial.begin(115200); // Start the Serial communication to send messages to the
  computer
  delay(10);
  Serial.println('\n');

  wifiMulti.addAP("ssid_from_AP_1", "your_password_for_AP_1"); // add Wi-Fi networks you want
  to connect to
  wifiMulti.addAP("ssid_from_AP_2", "your_password_for_AP_2");
  wifiMulti.addAP("ssid_from_AP_3", "your_password_for_AP_3");

  Serial.println("Connecting ...");
  int i = 0;
  while (wifiMulti.run() != WL_CONNECTED) { // Wait for the Wi-Fi to connect
    delay(1000);
    Serial.print(++i); Serial.print(' ');
  }
  Serial.println('\n');
  Serial.print("Connected to ");
  Serial.println(WiFi.SSID()); // Tell us what network we're connected to
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP()); // Send the IP address of the ESP8266 to the computer

  if (!mDNS.begin("esp8266")) { // Start the mDNS responder for esp8266.local
    Serial.println("Error setting up MDNS responder!");
  }
  Serial.println("mDNS responder started");

  SPIFFS.begin(); // Start the SPI Flash Files System
```

```

server.on("/upload", HTTP_GET, []() { // if the client requests the upload
page
    if (!handleFileRead("/upload.html")) // send it if it exists
        server.send(404, "text/plain", "404: Not Found"); // otherwise, respond with a 404 (Not
Found) error
    });

server.on("/upload", HTTP_POST, // if the client posts to the upload page
[]() { server.send(200); }, // Send status 200 (OK) to tell the
client we are ready to receive
handleFileUpload // Receive and save the file
);

server.onNotFound([]() { // If the client requests any URI
    if (!handleFileRead(server.uri())) // send it if it exists
        server.send(404, "text/plain", "404: Not Found"); // otherwise, respond with a 404 (Not
Found) error
    });

server.begin(); // Actually start the server
Serial.println("HTTP server started");
}

void loop() {
    server.handleClient();
}

String getContentType(String filename) { // convert the file extension to the MIME type
    if (filename.endsWith(".html")) return "text/html";
    else if (filename.endsWith(".css")) return "text/css";
    else if (filename.endsWith(".js")) return "application/javascript";
    else if (filename.endsWith(".ico")) return "image/x-icon";
    else if (filename.endsWith(".gz")) return "application/x-gzip";
    return "text/plain";
}

bool handleFileRead(String path) { // send the right file to the client (if it exists)
    Serial.println("handleFileRead: " + path);
    if (path.endsWith("/")) path += "index.html"; // If a folder is requested, send the
index file
    String contentType = getContentType(path); // Get the MIME type
    String pathWithGz = path + ".gz";
    if (SPIFFS.exists(pathWithGz) || SPIFFS.exists(path)) { // If the file exists, either as a
compressed archive, or normal
        if (SPIFFS.exists(pathWithGz)) // If there's a compressed version
available
            path += ".gz"; // Use the compressed version
        File file = SPIFFS.open(path, "r"); // Open the file
        size_t sent = server.streamFile(file, contentType); // Send it to the client
        file.close(); // Close the file again
        Serial.println(String("\tSent file: ") + path);
        return true;
    }
    Serial.println(String("\tFile Not Found: ") + path); // If the file doesn't exist, return
false
    return false;
}

void handleFileUpload(){ // upload a new file to the SPIFFS
    HTTPUpload& upload = server.upload();
    if(upload.status == UPLOAD_FILE_START){
        String filename = upload.filename;
        if(!filename.startsWith("/")) filename = "/" + filename;
        Serial.print("handleFileUpload Name: "); Serial.println(filename);
        fsUploadFile = SPIFFS.open(filename, "w"); // Open the file for writing in SPIFFS
(create if it doesn't exist)
        filename = String();
    } else if(upload.status == UPLOAD_FILE_WRITE){
        if(fsUploadFile)
            fsUploadFile.write(upload.buf, upload.currentSize); // Write the received bytes to the
file
    } else if(upload.status == UPLOAD_FILE_END){
        if(fsUploadFile) { // If the file was successfully created
            fsUploadFile.close(); // Close the file again
            Serial.print("handleFileUpload Size: "); Serial.println(upload.totalSize);
            server.setHeader("Location", "/success.html"); // Redirect the client to the success
page
            server.send(303);
        } else {
            server.send(500, "text/plain", "500: couldn't create file");
        }
    }
}

```

The `handleFileUpload` function just writes the file attached to the POST request to SPIFFS.

If you want to use other file types as well, you can just add them to the `getContentType` function.

## Uploading files

To upload a new file to the ESP, or to update an existing file, just go to <http://esp8266.local/upload>, click the *Choose File* button, select the file you wish to upload, and click *Upload*. You can now enter the URL into the URL bar, and open the new file.

## A note on safety

This example isn't very secure (obviously). Everyone that can connect to the ESP can upload new files, or edit the existing files and insert XSS code, for example. There's also not a lot of error checking/handling, like checking if there's enough space in the SPIFFS to upload a new file, etc.

## Advanced example

The code for these SPIFFS server examples comes (for the most part) from an example written by Hristo Gochkov. You can find it under File > Examples > ESP8266WebServer > FSBrowser. It has a web interface for browsing and editing files in your browser, and has some other nice features as well.