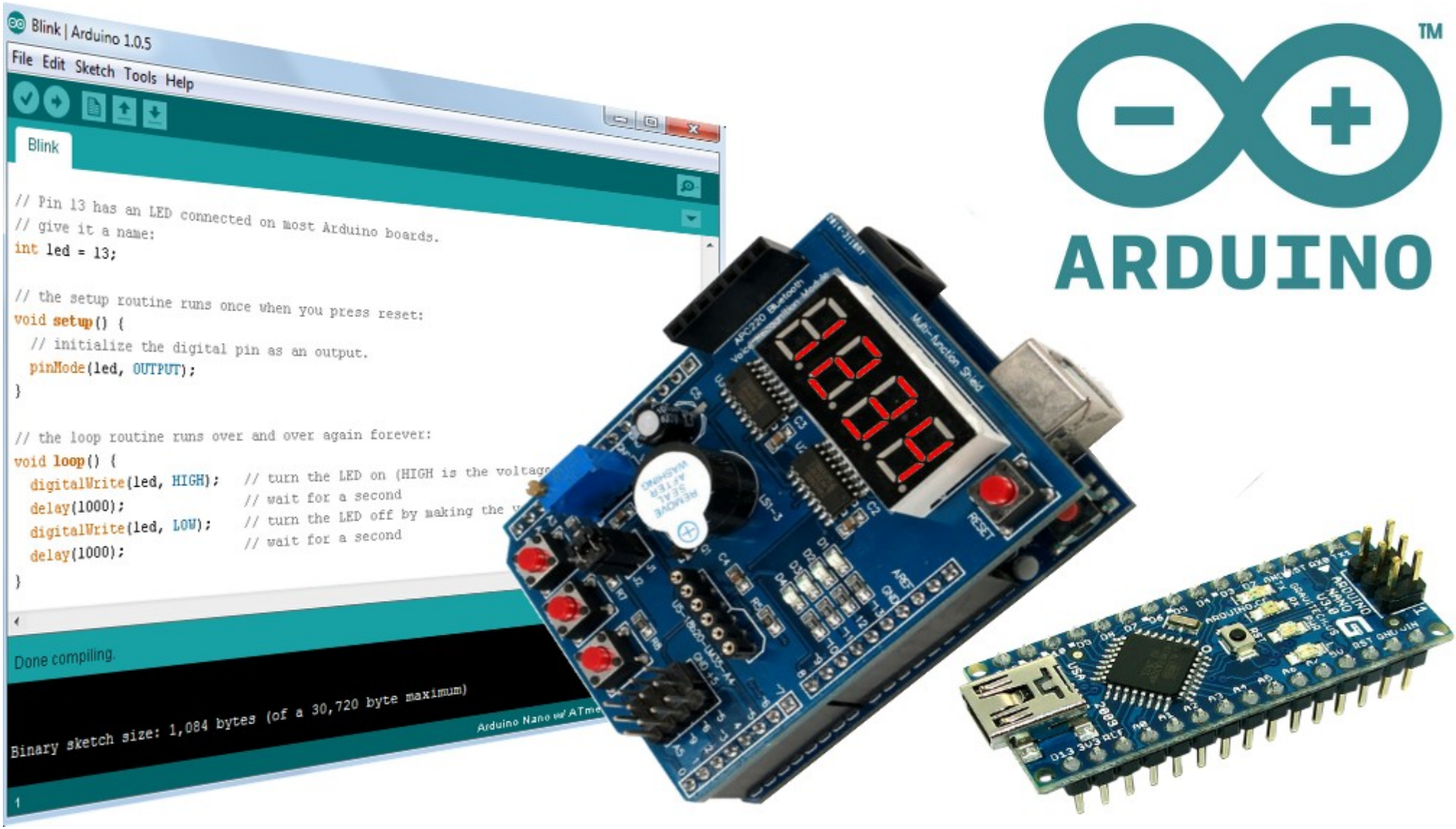
Arduino tanfolyam kezdőknek és haladóknak



17. ESP8266 alapú webszerver – 2. rész

Felhasznált és ajánlott irodalom

- Leírások, dokumentáció:
  - ESP8266 Community: ESP8266 Arduino Core's documentation
  - W3Schools: HTML Tutorial
  - W3Schools: CSS Tutorial
  - W3Schools: AJAX Tutorial
- Arduino programkönyvtárak és kiegészítések:
  - Benoit Blanchon: Mastering ArduinoJson 6
  - Arduino bővítmény: ESP8266FS data uploader
- Mintaprogramok:
  - Last Minute Engineers:
    Create A Simple ESP8266 Weather Station With BME280
  - Pieter P.: A Beginner's Guide to the ESP8266

Az előadásban érintett témák

- Az SPIFFS fájlrendszer használata
- Webszerver SPIFFS fájlrendszerben tárolt fájlokkal
- AJAX technika JSON adatlekéréssel (több adat egy lekéréssel)
- Webszerver SPIFFS fájlrendszerben tárolt tömörített fájlokkal (.gz)
- Fájlok feltöltése webszerverre
- Fájlok kilistázása webszerverrel (dinamikus „index lap” készítés)

A csatlakozások titkos adatai

- Az **ESP8266**-t a helyi hálózatra kliensként csatlakoztatjuk, az ehhez szükséges személyes adatokat kiszerveztük egy **secrets.h** nevű fejléc állományba, amelyet a **Vázlatfüzet** (Sketchbook) mappa **libraries/secrets** almappájában helyeztünk el

```
#include <ESP8266WiFi.h>
#include "secrets.h"

void setup_wifi() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  Serial.print("Connecting to ");
  Serial.print(WIFI_SSID);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Connected! IP address: ");
  Serial.println(WiFi.localIP());
}
```

```
#define WIFI_SSID  "MY_SSID"
#define WIFI_PASS  "MY_PASSWORD"
```

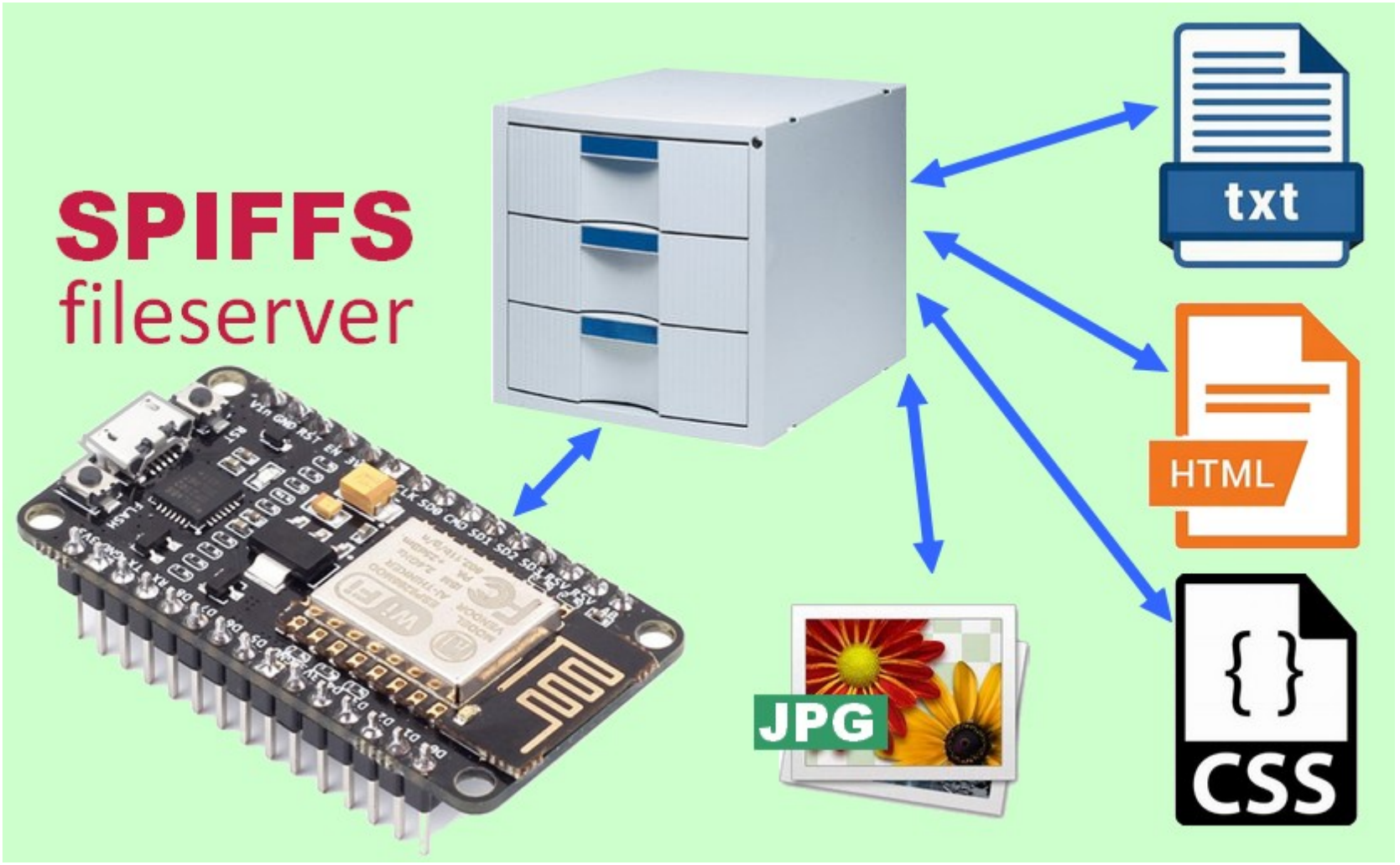
ESP8266 webszerver (emlékeztető)

- A megvalósításnál három lehetőség közül választhatunk:
  - A **ESP8266WiFi** programkönyvtár és a **WiFiServer** osztály használata
  - Az **ESP8266WebServer** programkönyvtár használata
  - Az **ESPAsyncWebServer** programkönyvtár használata
- HTML oldalak frissítése:
  - Az üzenet fejlécében elavulási/frissítési idő küldésével
  - A **HTML** dokumentum fejrészében elhelyezett sorral:
    `<meta http-equiv="refresh" content="5">`
  - **AJAX** technikával (a böngészőbe beépített **`XMLHttpRequest`** objektumosztály és a **JavaScript** lehetőségeinek felhasználásával)
- Az **AJAX** technikával az is megoldható (a **`getElementById()`** metódus használatával), hogy ne az egész dokumentumot, hanem csak annak kijelölt részeit cseréljük le frissítéskor

ESP8266 SPIFFS fájlrendszer

- Az **ESP8266**-hez társított SPI flash memóriában a firmware mellett fájlrendszert is kialakíthatunk, s a webszerver által szolgáltatott állományokat onnan is vehetjük

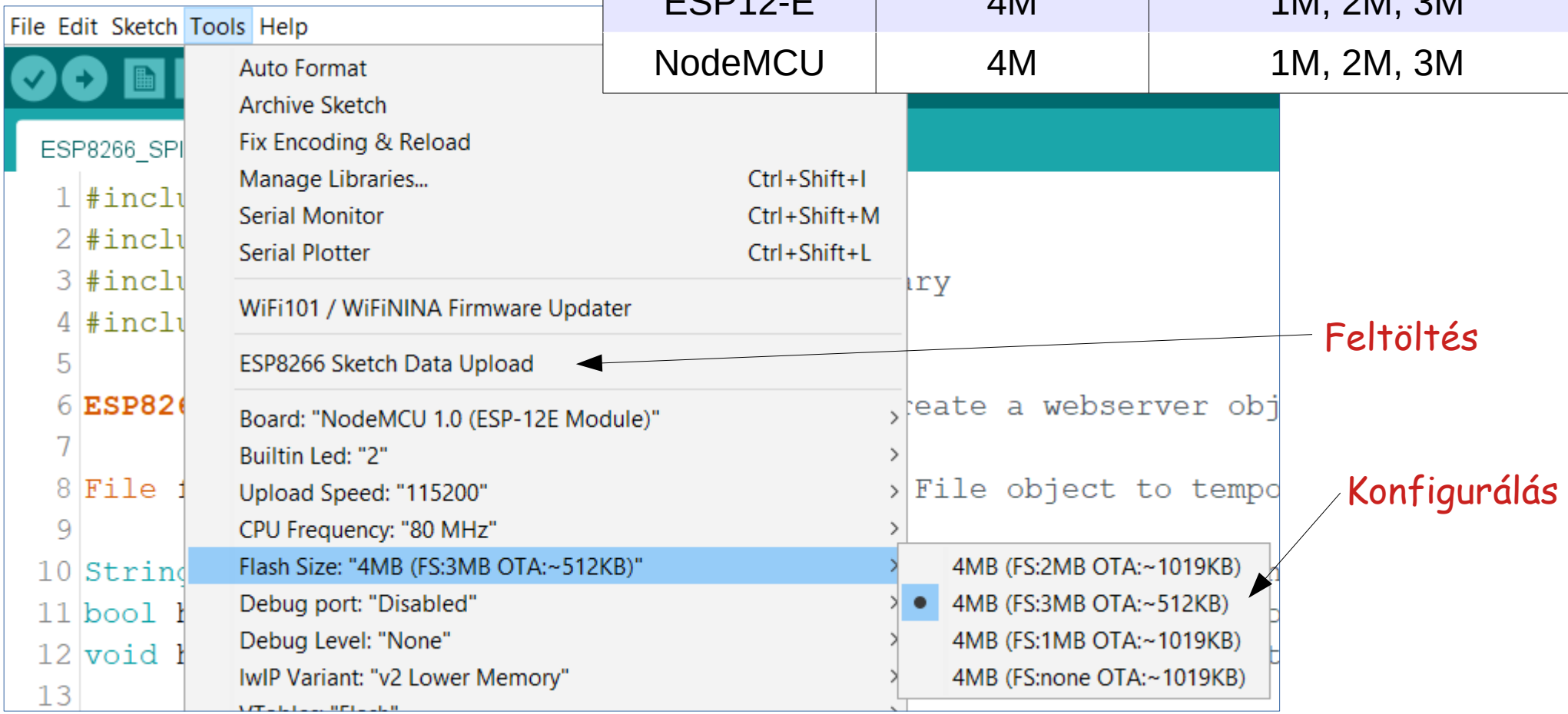
SPI flash



ESP8266 SPIFFS fájlrendszer

A táblázatban láthatjuk a fájlrendszerhez elkülöníthető memória méreteket (Arduino IDE Tools menüjében állítható)

Kártya	Memória	SPIFFS
ESP01 kék	512k	64k, 128k
ESP01 fekete	1M	64k, 128k, 256k, 512k
ESP12-E	4M	1M, 2M, 3M
NodeMCU	4M	1M, 2M, 3M



ESP8266 SPIFFS fájlrendszer

- Az alábbi diagram a memória felosztás sémáját szemlélteti

```
|--------------|-------|---------------|--|--|--|--|--|
^              ^       ^               ^     ^
Sketch    OTA update   File system   EEPROM  WiFi config (SDK)
```

- Az **SPIFFS** már elavultként kezelt, helytakarékos és korlátozott fájlrendszer (korlátja pl. hogy nincs benne könyvtárszerkezet), helyette az újabb **LittleFS** is használható, több-kevésbé hasonló szintaxissal, de annak némileg nagyobb az erőforrás igénye. Mi ebben az előadásban az **SPIFFS**-nél maradunk
- A fájlok feltöltése legegyszerűbben az Arduino IDE bővítményével történhet
- Töltsük le: github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.5.0/ESP8266FS-0.5.0.zip
- Az Arduino **Vázlatfüzet** mappában, a **libraries** almappa mellett hozzunk létre egy **tools** nevű almappát is és ebbe bontsuk ki a letöltött ZIP állományt
- Végeredményben nálam így helyezkedik el a könyvtárfában:
  `c:\Users\<username>\Documents\Arduino\tools\ESP8266FS\tool\esp8266fs.jar`
- A projekt mappában hozzunk létre egy ***data*** almappát, az abba másolt fájlokat lehet feltölteni a **Tools** menü **ESP8266 Sketch Data Upload** menüre kattintva (előtte konfigurálni kell a kártya paramétereket és a soros portot)

Az ESP8266 SPIFFS fájlrendszer használata

- Az **SPIFFS** használatához a program elején be kell csatolni az **FS.h** állományt
  `#include "FS.h"`
- **begin** – becsatolja a fájlrendszert: `SPIFFS.begin()`
- **end** – leválasztja a fájlrendszert: `SPIFFS.end()`
- **format** – (újra)formázza a fájlrendszert: `SPIFFS.format()` hívható **begin()** előtt vagy után is. A visszatérési érték ***true,*** ha sikeres volt a formázás
- **info** – a fájlrendszerre jellemző adatokat adja meg: `SPIFFS.info(fs_info)` ahol ***fs_info*** egy **FSinfo** típusú struktúra
- **open** – fájl megnyitása: `File f = SPIFFS.open(path, mode);` ahol ***mode:*** **r** – read, **w** -write, **a** – append (hozzáfűzés), esetleg r+, w+, a+
- **openDir** – könyvtár megnyitása: `Dir dir = SPIFFS.openDir ("/");` (SPIFFS estén nincsenek mappák, csak gyökérkönyvtár)
- **dir.next** – a visszatérési érték ***true,*** ha volt még fájl az iteráció során. Csak `dir.next()` után hívható `dir.fileName(), dir.fileSize()` és `dir.openFile()`

. . . és még sokan mások! Lásd itt: arduino-esp8266.readthedocs.io/en/stable/filesystem.html

A fájlok kilistázása

- Mit ér egy fájlrendszer fájlok nélkül? S mit tegyünk, ha már nem emlékszünk rá, hogy mit töltöttünk fel? Listázzuk ki a fájlokat!
- Az alábbi program a soros porton a terminálablakba listázza ki a fájlokat
- Az előadás végén a webserverrel fogjuk kilistáztatni a fájlokat

```
#include <ESP8266WiFi.h>
#include <FS.h>                          // Becsatoljuk az SPIFFS könyvtárat

void setup() {
  Serial.begin(115200);
  SPIFFS.begin();                        // Becsatoljuk az SPI Flash fájlrendszert
  Dir dir = SPIFFS.openDir ("/");
  while (dir.next()) {
    Serial.println(dir.fileName());
    Serial.println(dir.fileSize());
  }
}

void loop() {

}
```

Egyszerű SPIFFS Webszerver

- Ha a fájlokat az **SPIFFS** fájlrendszeren tároljuk, a legegyszerűbb webszerver pl. így nézhet ki:

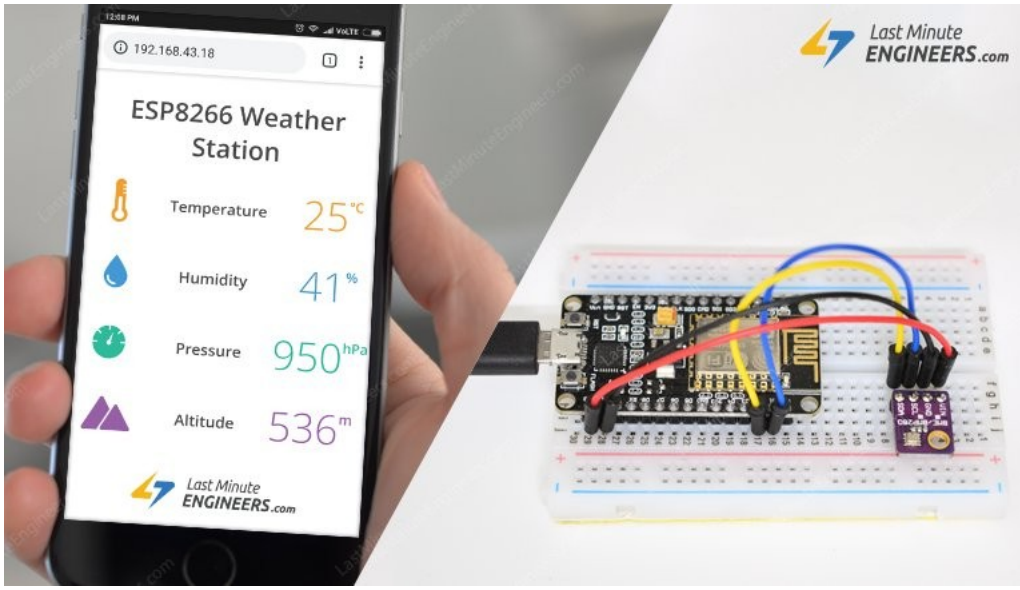
```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <FS.h>                       // Becsatoljuk az SPIFFS könyvtárat
#include "secrets.h"
ESP8266WebServer server(80);          // Webserver objektum példányosítása

void setup() {
  setup_wifi();
  SPIFFS.begin();                     // Becsatoljuk az SPI Flash fájlrendszert
  server.serveStatic("/", SPIFFS, "/index.html");
  server.serveStatic("/style.css", SPIFFS, "/style.css");
  server.onNotFound([]() {
    server.send(404, "text/plain", "404: Not Found");
  });
  server.begin();                     // Elindítjuk a webszervert
}

void loop() {
  server.handleClient();              // A háttérben mozgatjuk a szálakat...
}
```

ESP8266_WS_bme280_SPIFFS.ino

- A Last Minute Engineers ESP8266+BME280 időjárás állomásával már találkoztunk korábban, most a cikkben található utolsó (kidekorált és **AJAX** technikát használó) programváltozatot vesszük elő és alakítjuk át:
  - Az eredeti program **`SendHTML()`** függvényéből kimazsolázzuk és **`index.html`** néven elmentjük a weblap kódját (HTML, stílus és JavaScript)
  - Megjelöljük a cserélendő adatok helyét (**`<span id="...">0</span>`**)
  - Módosítjuk a **JavaScript** kódot, hogy a négy adatot egy **JSON** üzenetként vegye át és írja be az adatokat a megjelölt helyre
  - **Átírjuk a webszerver programot is** úgy, hogy külön kezelje a weblap lekérését és a mérési adatok lekérését (ne küldjük ki minden lekéréskor a teljes weblapot)
  - A mérési adatok kiküldésekor az **ArduinoJson** könyvtár segítségével állítjuk össze az üzenetet



ESP8266_WS_bme280_SPIFFS.ino 3/1.

```
#include <ESP8266WebServer.h>
#include <FS.h>                                   // Include File System Headers
#include <ArduinoJson.h>                          // Include JSON support headers
#include "secrets.h"
#include <Wire.h>                                 // I2C library header
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

#define SEALEVELPRESSURE_HPA (1013.25)
float temperature, humidity, pressure, altitude;
Adafruit_BME280 bme;
ESP8266WebServer server(80);
StaticJsonDocument<200> doc;                      // Create a short JSON document

void setup() {
  setup_wifi();                                   // Connect to local WiFi network
  bme.begin(0x76);                                // Start sensor
  SPIFFS.begin();                                 // Mount filesystem
  server.on("/readBME280", handle_BME280);        // Serve AJAX requests
  server.onNotFound(handle_NotFound);             // Serve file requests
  server.begin();                                 // Start webserver
}

void loop() {
  server.handleClient();                          // Handle clien connections
}
```

ESP8266_WS_bme280_SPIFFS.ino 3/2.

- A lekérések kiszolgálása ezekben a függvényekben történik:

```
void handle_BME280() {
  doc["temperature"] = String(bme.readTemperature(), 1);
  doc["humidity"] = (int)bme.readHumidity();
  doc["pressure"] = String(((bme.readPressure() + 1440) / 100.0F), 1);
  doc["altitude"] = (int)bme.readAltitude(SEALEVELPRESSURE_HPA);
  server.send(200, "application/json", doc.as<String>());
}

void handle_NotFound() {
  if (!handleFileRead(server.uri()))                    // send file if exists
    server.send(404, "text/plain", "404: Not Found");   // or respond with a 404
}

bool handleFileRead(String path) {
  if (path.endsWith("/")) path += "index.html";         // send the index file
  String contentType = getContentType(path);            // Get the MIME type
  if (SPIFFS.exists(path)) {                            // If the file exists
    File file = SPIFFS.open(path, "r");                 // Open it
    size_t sent = server.streamFile(file, contentType); // And send it to the client
    file.close();                                       // Then close the file again
    return true;
  }
  return false;                                         // If file not found...
}
```

ESP8266_WS_bme280_SPIFFS.ino 3/3.

```
String getContentType(String filename){
  if(filename.endsWith(".htm")) return "text/html";
  else 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(".png")) return "image/png";
  else if(filename.endsWith(".gif")) return "image/gif";
  else if(filename.endsWith(".jpg")) return "image/jpeg";
  else if(filename.endsWith(".ico")) return "image/x-icon";
  else if(filename.endsWith(".svg")) return "image/svg+xml";
  else if(filename.endsWith(".xml")) return "text/xml";
  else if(filename.endsWith(".json")) return "application/json";
  else if(filename.endsWith(".zip")) return "application/x-zip";
  else if(filename.endsWith(".gz")) return "application/x-gzip";
  return "text/plain";
}

void setup_wifi() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    Delay(500); Serial.print(".");
  }
  Serial.print("\r\nConnected! IP address: ");
  Serial.println(WiFi.localIP());
}
```

- A Média Típusok leírását lásd pl. itt: Common MIME Types

Az index.html állomány tartalma 3/1.

```
<!DOCTYPE html>
<html>
<head>
  <title>ESP8266 Weather Station</title>
  <meta name='viewport' content='width=device-width, initial-scale=1.0'>
  <link href='https://fonts.googleapis.com/css?family=Open+Sans:300,400,600' rel='stylesheet'>
  <style> ... itt nem részletezzük ... </style>

  <script>
    setInterval(loadDoc, 2000);
    function loadDoc() {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          var obj =JSON.parse(this.responseText);
          document.getElementById("bme_temperature").innerHTML = obj.temperature;
          document.getElementById("bme_humidity").innerHTML = obj.humidity;
          document.getElementById("bme_pressure").innerHTML = obj.pressure;
          document.getElementById("bme_altitude").innerHTML = obj.altitude;
        }
      };
      xhttp.open("GET", "/readBME280", true);
      xhttp.send();
    }
  </script>
</head>
```

Deserialize JSON

Itt az eredetiben ez állt:

```
document.body.innerHTML = this.responseText;
```

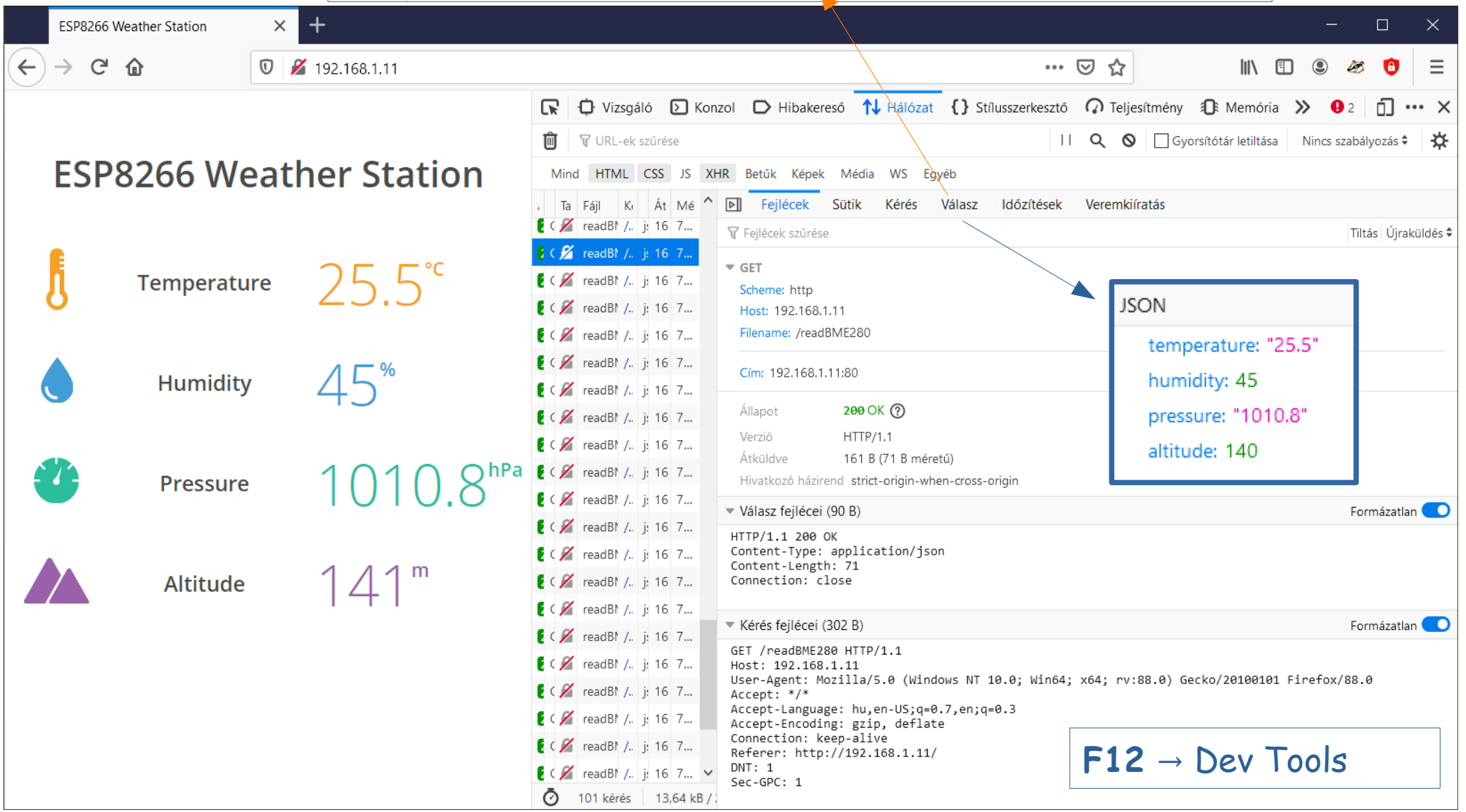
Az index.html állomány tartalma 2/2.

```
<body><h1>ESP8266 Weather Station</h1>
  <div class='container'>
    <div class='data temperature'>
      <div class='side-by-side icon'> <svg>... itt nem részletezzük ...</svg> </div>
      <div class='side-by-side text'>Temperature </div>
      <div class='side-by-side reading'><span id="bme_temperature">0</span>
        <span class='superscript'>&deg;C</span></div>
    </div>
    <div class='data humidity'>
      <div class='side-by-side icon'> <svg>... itt nem részletezzük ...</svg> </div>
      <div class='side-by-side text'>Humidity </div>
      <div class='side-by-side reading'><span id="bme_humidity">0</span>
         <span class='superscript'>%</span></div>
    </div>
    <div class='data pressure'>
      <div class='side-by-side icon'> <svg>... itt nem részletezzük ...</svg> </div>
      <div class='side-by-side text'>Pressure </div>
      <div class='side-by-side reading'><span id="bme_pressure">0</span>
         <span class='superscript'>hPa</span></div>
    </div>
    <div class='data altitude'>
      <div class='side-by-side icon'><svg>... itt nem részletezzük ...</svg> </div>
      <div class='side-by-side text'>Altitude </div>
      <div class='side-by-side reading'><span id="bme_altitude">0</span>
        <span class='superscript'>m</span></div>
    </div>
  </div></body></html>
```

A program futási eredménye

JSON

```
1  {"temperature":"25.5","humidity":45,"pressure":"1010.8","altitude":140}
```



ESP8266_SPIFFS_upload.ino

- Pieter P.: A Beginner's Guide to the ESP8266 12. fejezetében egy **SPIFFS** webserver-t mutat, amely **fájlok feltöltését** is támogatja. A program hasonló az előzőhöz, ezért csak az eltéréseket mutatjuk

```
File fsUploadFile;                                  // a File object for temporarily storage

void setup() {
  setup_wifi();
  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 404 error
  });

  server.on("/upload", HTTP_POST,                   // if the client posts to the upload page
  []() {server.send(200); },                        // Send status 200 (OK)
  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 404 error
  });

  server.begin();                                   // Actually start the server
}
```

ESP8266_SPIFFS_upload.ino

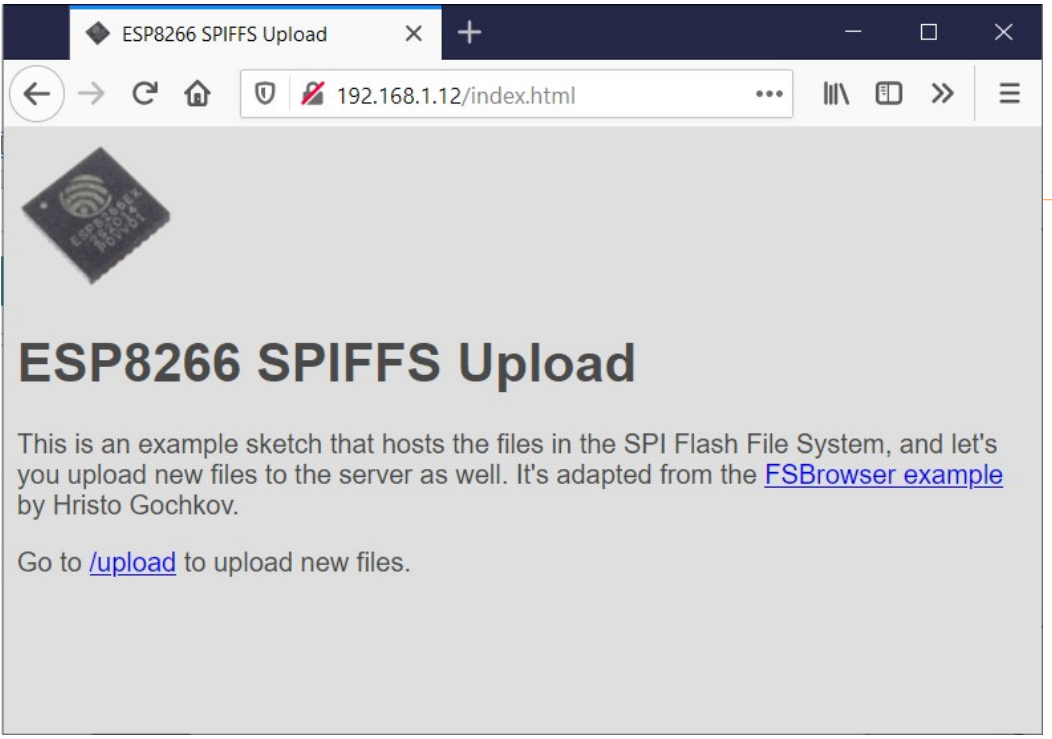
- A **handleFileRead()** függvényt egy kis módosítással arra is rávehetjük, hogy **GNU gzip**-pel tömörített fájlokat kezeljen (ez is az előző oldalon említett leírásban található)
- Ha a kért fájl megtalálható **.gz** kiterjesztéssel, akkor azt küldjük ki
- A **MIME** típus továbbra is ugyanaz marad, mint az eredeti fájl típusa a `Content-encoding: gzip` információt a webszerver fűzi hozzá

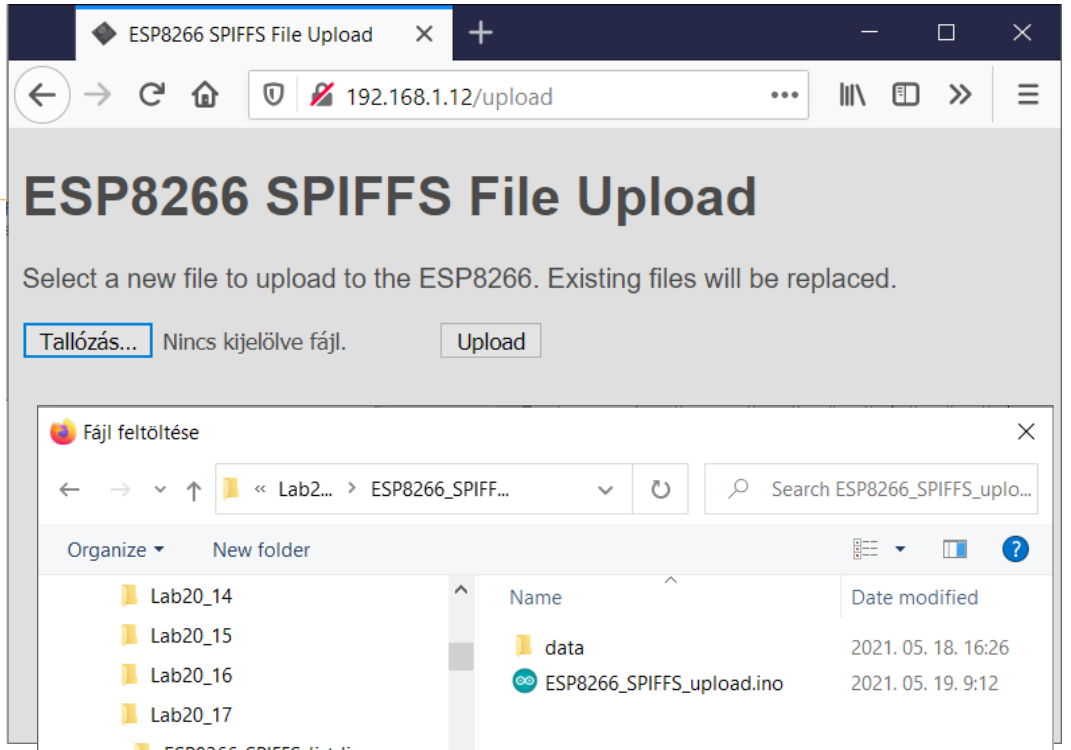
```
bool handleFileRead(String path) {                          // send file to the client
  if (path.endsWith("/")) path += "index.html";             // If a folder is requested
  String contentType = getContentType(path);                // Get the MIME type
  String pathWithGz = path + ".gz";
  if (SPIFFS.exists(pathWithGz) || SPIFFS.exists(path)) {   // If the file exists
    if (SPIFFS.exists(pathWithGz))                          // If there's a compressed
      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
    return true;
  }
  return false;
}
```

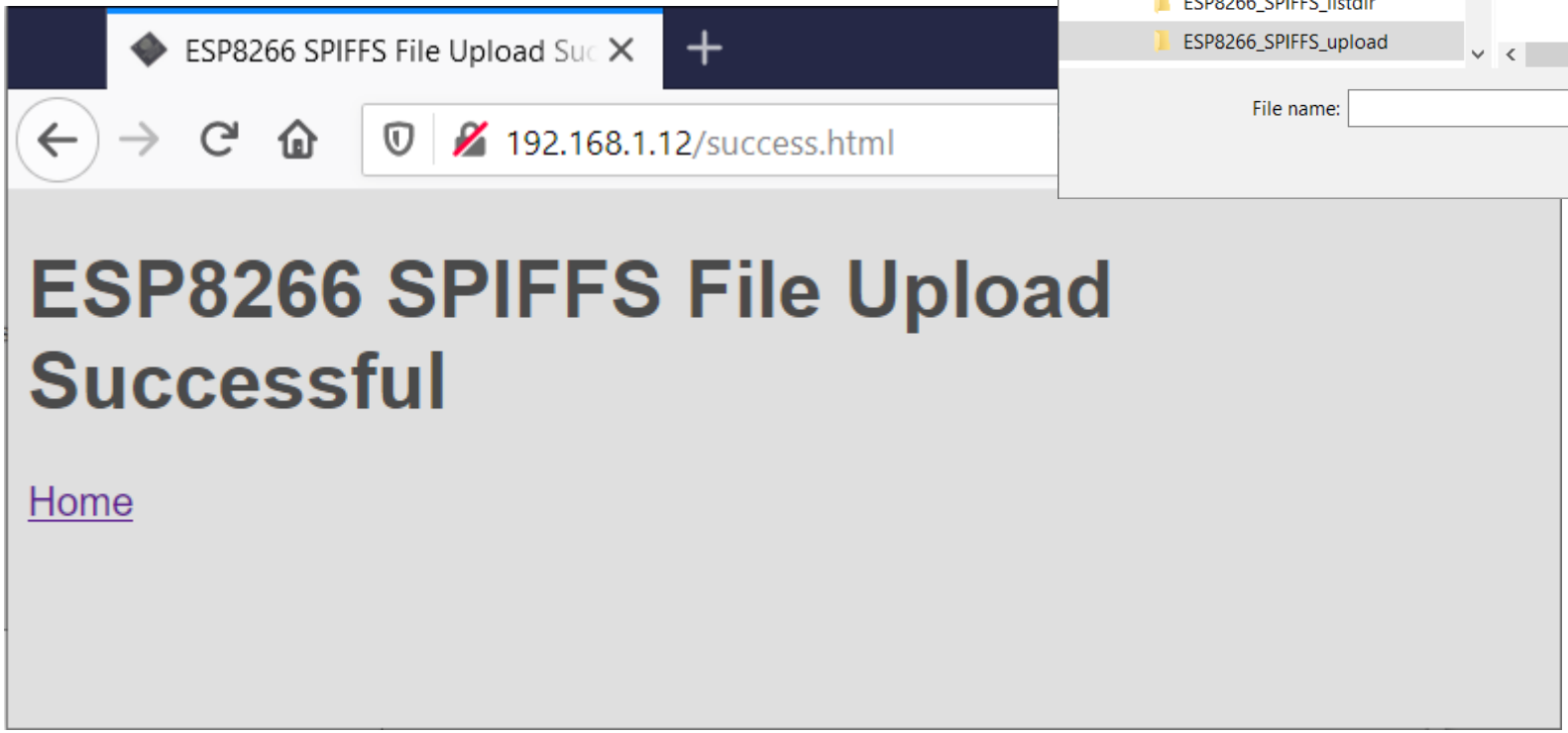
ESP8266_SPIFFS_upload.ino

- A fájl létrehozásáról és a kapott adatok feltöltéséről az alábbi függvény gondoskodik
- Sikeres feltöltés esetén a klienst átirányítjuk a **/success.html** lapra

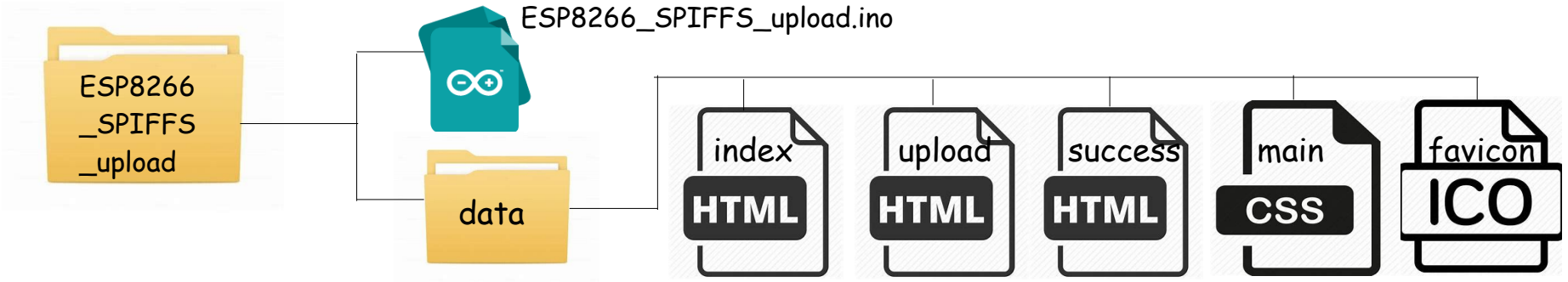
```
void handleFileUpload() {                                   // upload a file to SPIFFS
  HTTPUpload& upload = server.upload();
  if (upload.status == UPLOAD_FILE_START) {
    String filename = upload.filename;
    if (!filename.startsWith("/")) filename = "/" + filename;
    Serial.print("handleFileUpload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


ESP8266 SPIFFS Upload
192.168.1.12/index.html
ESP8266 SPIFFS Upload
This is an example sketch that hosts the files in the SPI Flash File System, and let's you upload new files to the server as well. It's adapted from the FSBrowser example by Hristo Gochkov.
Go to /upload to upload new files.


ESP8266 SPIFFS File Upload
192.168.1.12/upload
ESP8266 SPIFFS File Upload
Select a new file to upload to the ESP8266. Existing files will be replaced.
Tallózás...
Nincs kijelölve fájl.
Upload
Fájl feltöltése
Lab2...
ESP8266_SPIFF...
Search ESP8266_SPIFFS_uplo...
Organize
New folder
Lab20_14
Lab20_15
Lab20_16
Lab20_17
ESP8266_SPIFFS_listdir
ESP8266_SPIFFS_upload
Name
Date modified
data
2021. 05. 18. 16:26
ESP8266_SPIFFS_upload.ino
2021. 05. 19. 9:12
File name:
Minden fájl (*.*)
Open
Cancel


ESP8266 SPIFFS File Upload Su
192.168.1.12/success.html
ESP8266 SPIFFS File Upload Successful
Home

Mi van a data mappában?



index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>ESP8266 SPIFFS Upload</title>
    <link rel="stylesheet" type="text/css" href="main.css">
</head>
<body>
    <img src="favicon.ico">
    <h1>ESP8266 SPIFFS Upload</h1>
    <p>This is an example sketch that hosts the files in the SPI Flash File
    System, and let's you upload new files to the server as well. It's
    adapted from the <a
href="https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WebServ
er/examples/FSBrowser">FSBrowser example</a> by Hristo Gochkov.</p>
    <p>Go to <a href="/upload">/upload</a> to upload new files.</p>
</body>
</html>
```

Mi van a data mappában?

main.css

```css
body {
    font-family: sans-serif;
    color: #444;
    background-color: #ddd;
}
```

success.html

```html
<!DOCTYPE html>
<html>

<head>
    <title>ESP8266 SPIFFS File Upload Successful</title>
    <link rel="stylesheet" type="text/css" href="main.css">
</head>

<body>
    <h1>ESP8266 SPIFFS File Upload Successful</h1>
    <p><a href="/">Home</a></p>
</body>

</html>
```

Mi van a data mappában?

upload.html

```
<!DOCTYPE html>
<html>

<head>
    <title>ESP8266 SPIFFS File Upload</title>
    <link rel="stylesheet" type="text/css" href="main.css">
</head>

<body>
    <h1>ESP8266 SPIFFS File Upload</h1>
    <p>Select a new file to upload to the ESP8266.
       Existing files will be replaced.</p>

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

</html>
```

Készítsünk SPIFFS index lapot!

- Néha jól jön, ha az **SPIFFS** tartalomjegyzékét ki tudjuk íratni. Hogy ne kelljen hozzá külön program, a listázást kombináljuk az előző webszerver programmal
- **Változás:** a `GET "/"` lekéréskor most nem az `index.html` lapot küldjük el, hanem az SPIFFS tartalomjegyzékét
- A tartalomjegyzék összeállításakor **a fájl nevekhez linket is rendelünk,** így rákattintással bármelyik fájlt le lehet kérni
- **Megjegyzés:** ha az adott fájl gzip-pel tömörített formában van tárolva, akkor **a linkben a .gz kiterjesztést levágva kell hivatkozni,** mert csak így lesz helyesen megadva a letöltéskor a fájl **Média Típusa**

ESP8266_SPIFFS_upload_dir.ino

- Mivel a program az előző projekt módosítása, itt csak az eltéréseket mutatjuk

```
File fsUploadFile;                                  // a File object for temporarily storage

void setup() {
  setup_wifi();
  SPIFFS.begin();                                   // Start the SPI Flash Files System
  server.on("/", handleDirList);                    // List files in the SPIFFS
  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 404 error
  });

  server.on("/upload", HTTP_POST,                   // if the client posts to the upload page
  []() {server.send(200); },                        // Send status 200 (OK)
  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 404 error
  });

  server.begin();                                   // Actually start the server
}
```

ESP8266_SPIFFS_upload.ino

- A **handleFileRead()** függvényből távolítsuk el az alábbi sort:

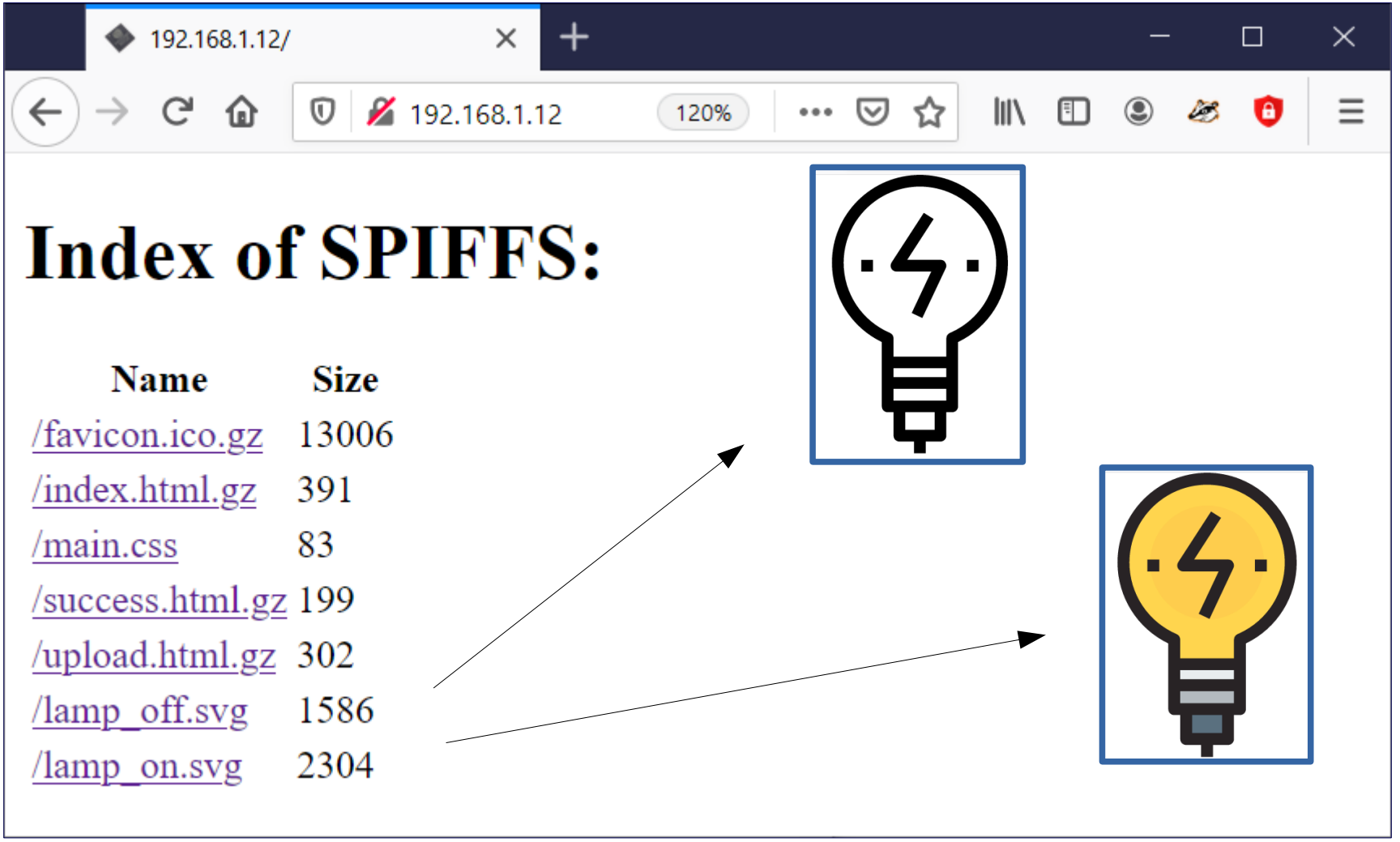
```
if(path.endsWith("/")) path += "index.html"; // If folder requested, send index file
```

- A **handleDirList()** függvény nem előre elkészített fájlt tölt le, hanem „röptében” állítja össze a **HTML** lap tartalmát
- A „nagy” webszerverekhez hasonlóan táblázatos formában listázzuk ki a fájlok nevét és méretét

```
void handleDirList() {
  String page = F("<!DOCTYPE html><html><h1>Index of SPIFFS:</h1>"
                  "<table><tr><th>Name</th><th>Size</th></tr>");
  Dir dir = SPIFFS.openDir ("/");
  while (dir.next ()) {
    String fname = dir.fileName();
    if(fname.endsWith(".gz")) { fname.replace(".gz",""); }
    page += "<tr><td><a href=\"" + fname + "\"</a>" + dir.fileName() + "</td>";
    page += "<td>" + String(dir.fileSize()) + "</td></tr>";
  }
  page += F("</table></html>");
  server.send(200, "text/html", page);
}
```

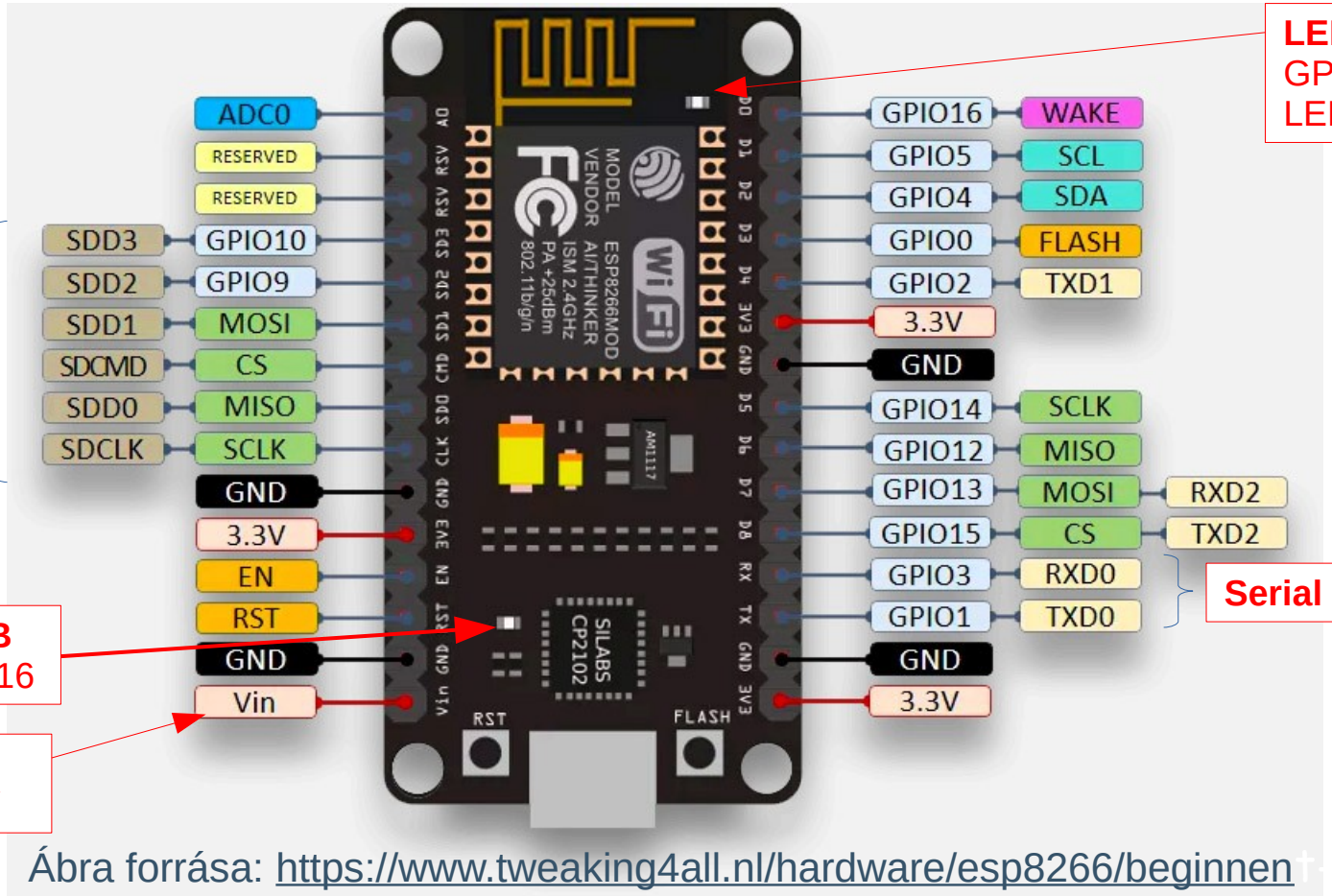
ESP8266_SPIFFS_upload_dir.ino: eredmény

- A program futási eredménye az alábbi ábrán látható
- A lamp_off.svg és lamp_on.svg képek nem tartoznak a projekthez, a fájlfeltöltés próbálgatásakor kerültek fel



NodeMCU GPIO kivezetések

- A ki- és bemenetek jelszintje 3,3 V (ADC0 esetén 1 V lenne, de a **NodeMCU** kártyán van egy 200 k + 100 k előosztó)
- FLASH a jobboldali nyomógombhoz csatlakozik
- Az Arduino számozás a **GPIO*n*** jelölésből leválasztott ***n*** szám



Ábra forrása: https://www.tweaking4all.nl/hardware/esp8266/beginnen

Név	GPIO
D0	16
D1	5
D2	4
D3	0
D4	2
D5	14
D6	12
D7	13
D8	15
A0	19