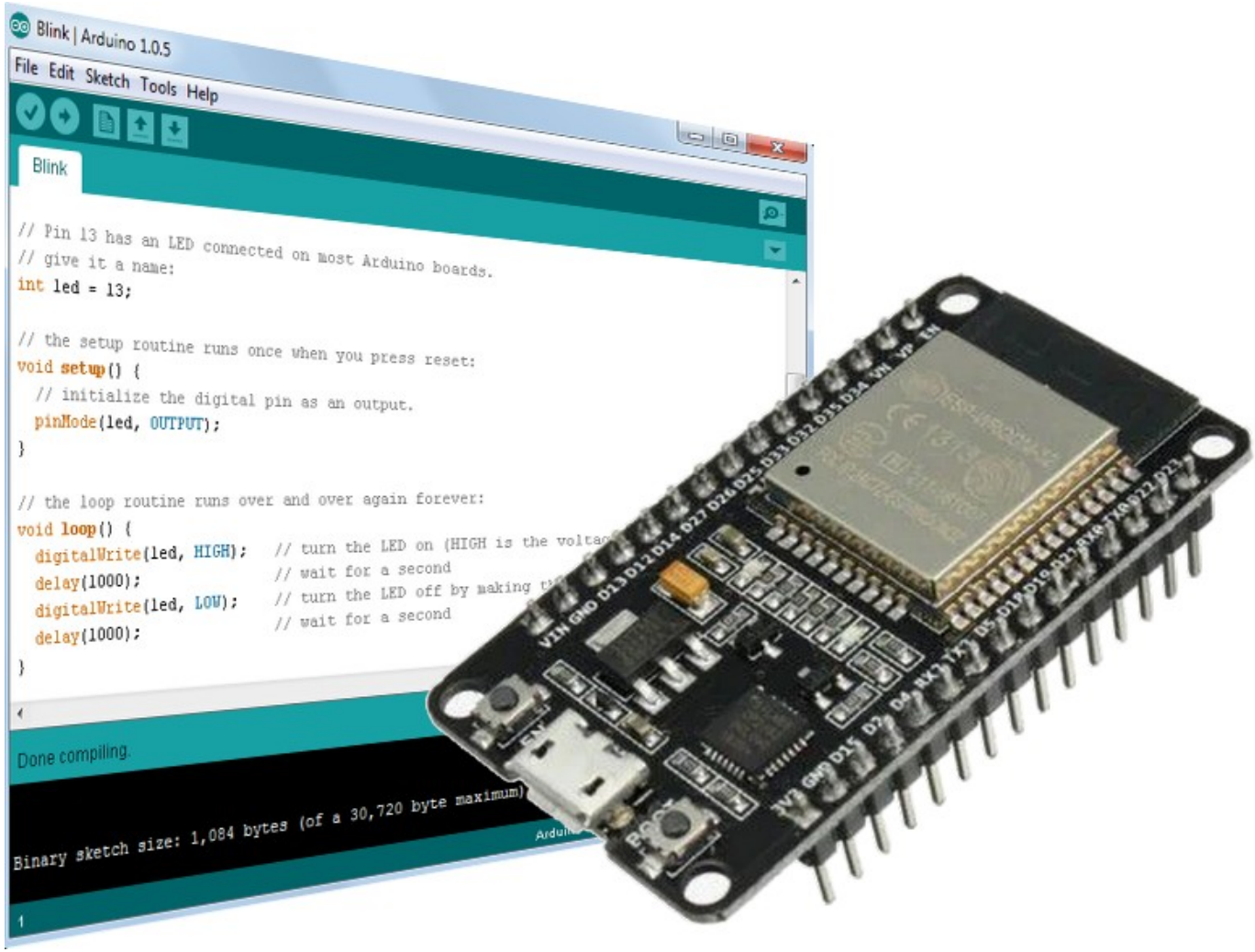
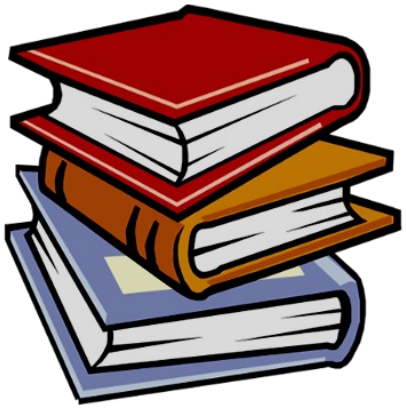
ESP32 mikrovezérlők programozása Arduino környezetben



11. ESP32 webszerverek – 2. rész

Felhasznált és ajánlott irodalom

- **Leírások, dokumentáció**
  - W3Schools: HTML Tutorial
  - W3Schools: CSS Tutorial
  - W3Schools: AJAX Tutorial
  - ESPRESSIF: ESP32 Arduino Core Documentation

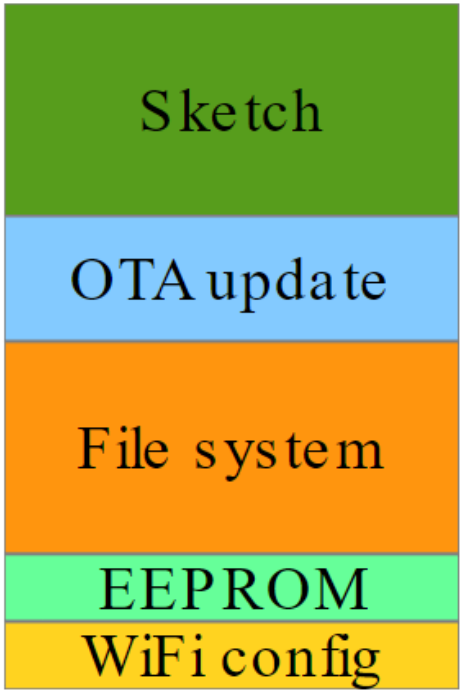


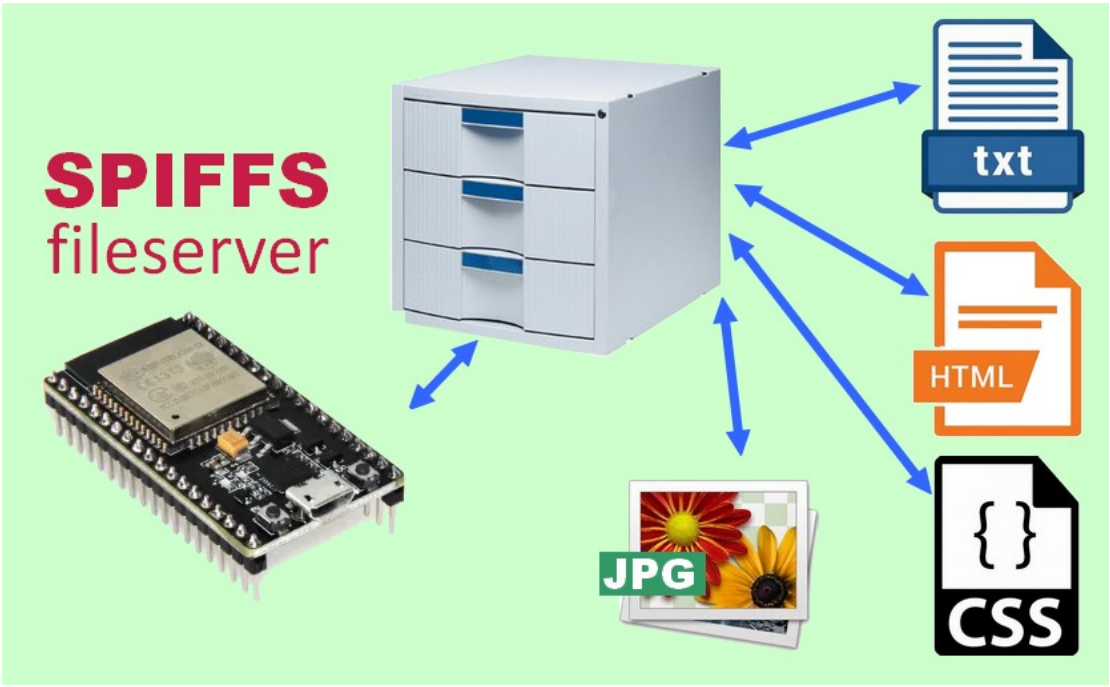
- **Mintaprojektek**
  - Renzo Mischianti: ESP32: integrated SPIFFS FileSystem – Part 2
  - Random Nerd Tutorials: ESP32 Web Server using SPIFFS
  - Last Minute Engineers: ESP32 Weather Station With BME280
- **Arduino programkönyvtárak és kiegészítők**
  - Arduino ESP32FS plugin
  - AsyncTCP
  - ESPAsyncWebServer
  - Adafruit: BME280 sensor library

Az SPIFFS fájlrendszer

- Az **ESP32** mikrovezérlőhöz társított SPI flash memóriában a firmware mellett fájlrendszert is kialakíthatunk, s a webszerver által szolgáltatott állományokat onnan is vehetjük
- Jellemzők: *flat* fájlrendszer (nincsenek alkönyvtárak), minimális RAM memóriaigény, viszonylag lassú elérés, az **FS** könyvtárra épül, melynek tagfüggvényei különböznek az ESP32 és ESP8266 esetében

Memória térkép





ESP32_SPIFFS_seek.ino

- Renzo Mischianti mintaprogramja bemutatja a fájl megnyitását, írását, olvasását és a fájlmutató beállítását (*seek*) adott pozícióba
  - Létrehozunk egy **testCreate.txt** nevű állományt és beleírjuk ezt a szöveget: "*Here is the test text*", majd bezárjuk a fájlt
  - Újra megnyitjuk a fenti fájlt, beolvassuk és kiíratjuk a tartalmát
  - Újra megnyitjuk a fájlt, beállítjuk a fájlmutatót a 9. karakter utáni pozícióba, és onnan kezdve beolvassuk/kiíratjuk a fájl tartalmát
- **A programban használt függvények**
  - **begin**(*formatting*) – a fájlrendszer felcsatolása és szükség estén formázása
  - **open**(*filename, mode*) – fájl megnyitása írásra (w), olvasásra (r)
- **Fájlkezelő tagfüggvények**
  - **print**(*str*) – fájlba kiírja a megadott szöveget
  - **readString**() – szövegként beolvassa a fájl tartalmát
  - **seek**(*n, mode*) – fájlmutató pozicionálása (**SeekSet:** az n-edik karakter, **SeekCur:** az aktuális hely+n, **SeekEnd:** végétől visszafelé n-edik pozíció)
  - **close**() – fájl lezárása

ESP32_SPIFFS_seek.ino

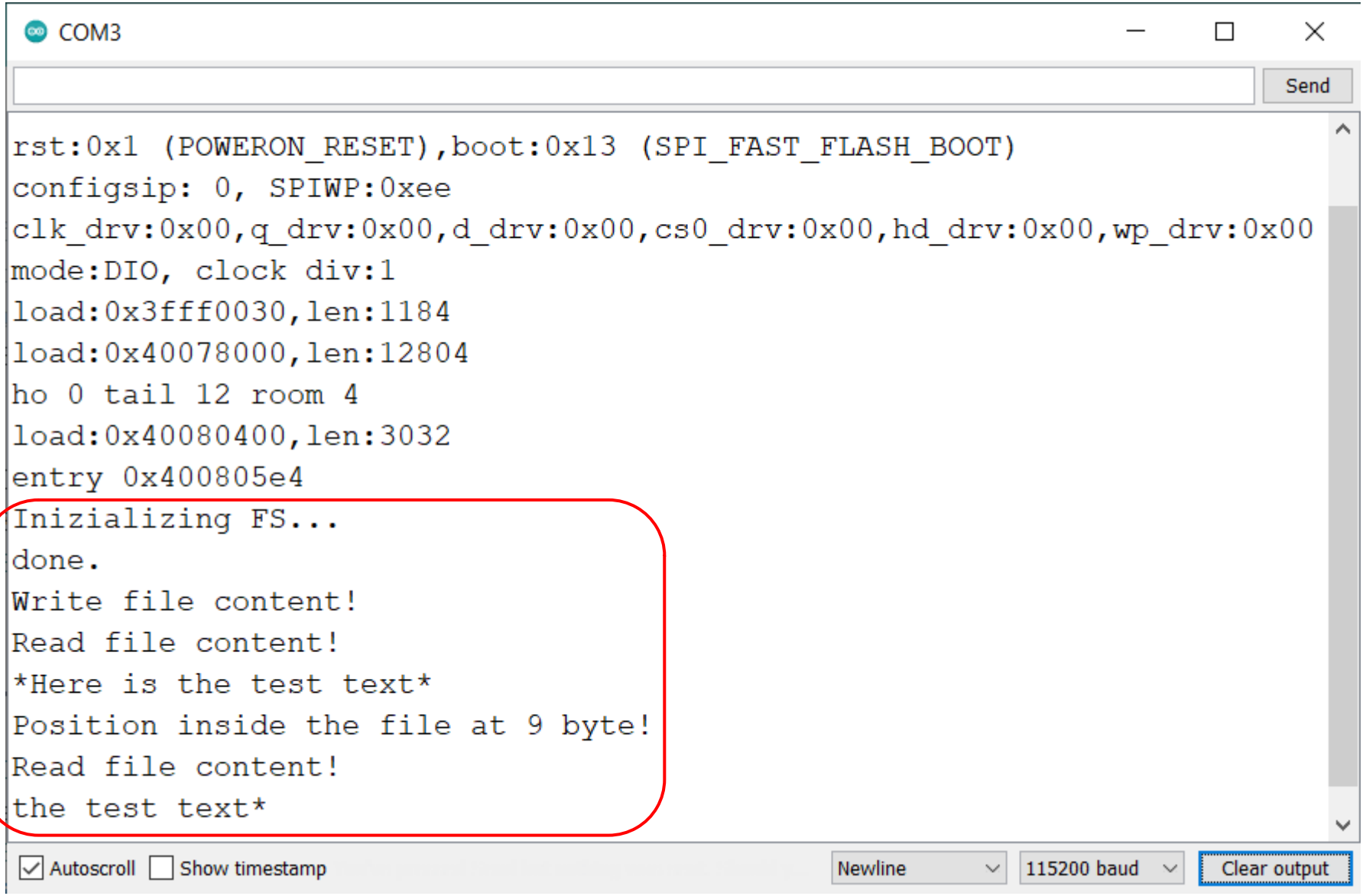
Első alkalommal **true** értékkel hívjuk meg a formázáshoz

```
#include "SPIFFS.h"

void setup() {
    Serial.begin(115200);
    Serial.println(F("Initializing FS..."));
    if (SPIFFS.begin(true)) { Serial.println(F("done.")); }
    else{ Serial.println(F("fail.")); }
    File testFile = SPIFFS.open(F("/testCreate.txt"), "w");
    if (testFile){
        Serial.println("Write file content!");
        testFile.print("*Here is the test text*");
        testFile.close();
    } else{ Serial.println("Problem on create file!"); }
    testFile = SPIFFS.open(F("/testCreate.txt"), "r");
    if (testFile){ Serial.println("Read file content!");
        Serial.println(testFile.readString());
        testFile.close();
    } else{ Serial.println("Problem on read file!"); }
    testFile = SPIFFS.open(F("/testCreate.txt"), "r");
    if (testFile){
        Serial.println("Position inside the file at 9 byte!");
        testFile.seek(9, SeekSet);
        Serial.println("Read file content!");
        Serial.println(testFile.readString());
        testFile.close();
    } else{ Serial.println("Problem on read file!"); }
}
```

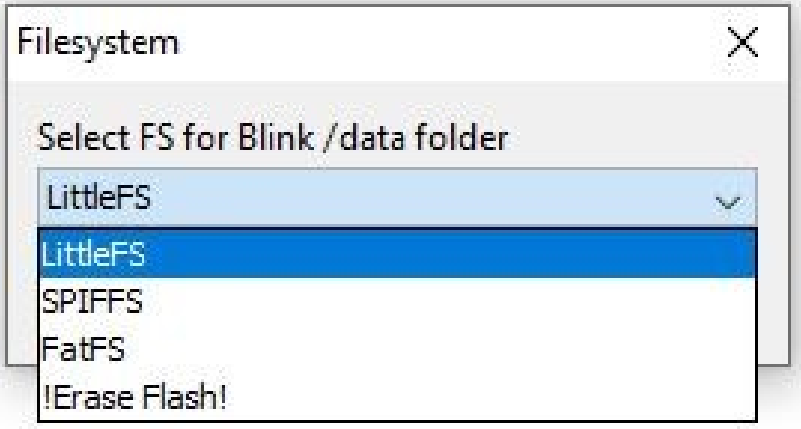
ESP32_SPIFFS_seek.ino futási eredménye

- Az ábrán a program futási eredménye látható (RESET után)



Fájlok feltöltése Arduino IDE alatt

- Az **SPIFFS** (vagy LittleFS, esetlef FatFS) fájlrendszer Arduino IDE alatti kezeléséhez telepíteni kell az Arduino ESP32FS plugin-t:
  - Töltsük le az **esp32fs.zip** csomagot a legfrissebb kiadásból!
  - Bontsuk ki azt a *Vázlatfüzet* mappa ***tools*** almappájába így:
    `/users/<name>/Documents/Arduino/tools/ESP32FS/tool/esp32fs.jar`
- Az Arduino projektünk mappájában egy ***data*** almappába másoljuk be az **ESP32** fájlrendszerébe feltölteni kívánt állományokat
- Csatlakoztassuk az ESP32 kártyát, indítsuk el az Arduino IDE-t, s a ***Tools*** menüben kattintsunk az ***ESP32 Sketch Data Upload*** menüpontra, s válasszuk ki a használni kívánt fájlrenszert (esetünkben SPIFFS)
- Az indítás után (OK gomb) 5-6 másodpercig tartsuk lenyomva az ESP kártya BOOT gombját!



ESP32_SPIFFS_listdir.ino

- Az alábbi minimalista programmal kilistázzuk az SPIFFS fájlrendszer állományainak nevét és méretét (bájtokban)

```
#include "SPIFFS.h"
/* You only need to format SPIFFS the first time you run a
   test or else use the SPIFFS plugin to create a partition
   https://github.com/me-no-dev/arduino-esp32fs-plugin */
#define FORMAT_SPIFFS_IF_FAILED false

void setup() {
  Serial.begin(115200);
  if (!SPIFFS.begin(FORMAT_SPIFFS_IF_FAILED)) {
    Serial.println("SPIFFS Mount Failed");
    return;
  }
  File root = SPIFFS.open("/");
  while (File file = root.openNextFile()) {
    Serial.print("FILE: ");
    Serial.print(file.name());
    Serial.print("\tSIZE: ");
    Serial.println(file.size());
  }
}

void loop() { }
```

```
FILE: 404.html      SIZE: 2802
FILE: style.css     SIZE: 2555
FILE: index.html    SIZE: 483
FILE: 4or3.jpg      SIZE: 27566
FILE: pista.jpg     SIZE: 5858
```

ESP32 webszerver STA módban

- ESP32 webszerverünk STA (station) módban kapcsolódik a helyi hálózatra, s a korábbi előadásokban bemutatott módin fix IP címet rendelünk hozzá
- A helyi hálózat eszözeivel (laptop, tablet, mobil) a fix IP cím alapján könnyen elérhető

A csatlakozások titkos adatai

- Az **ESP32**-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 <WiFi.h>
#include "secrets.h"

void setup_wifi() {
  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"
```

ESP32 webszerverek (emlékeztető)

- Három lehetőség közül választhatunk:
  - A **WiFi** programkönyvtár és a **WiFiServer** osztály használata
  - A **WebServer** programkönyvtár használata
  - Az **ESPAsyncWebServer** programkönyvtár (**AsyncTCP** is kell hozzá)
- A sablonokat (pl. `%STATE%`) küldéskor lecserélhetjük egy függvényben
- HTML oldalak periodikus frissítése:
  - Az üzenet fejlécében elavulási/frissítési idő küldésével
  - A **HTML** dokumentum fejrészében elhelyezett sorral, például:
    `<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

SPIFFS Webszerver

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

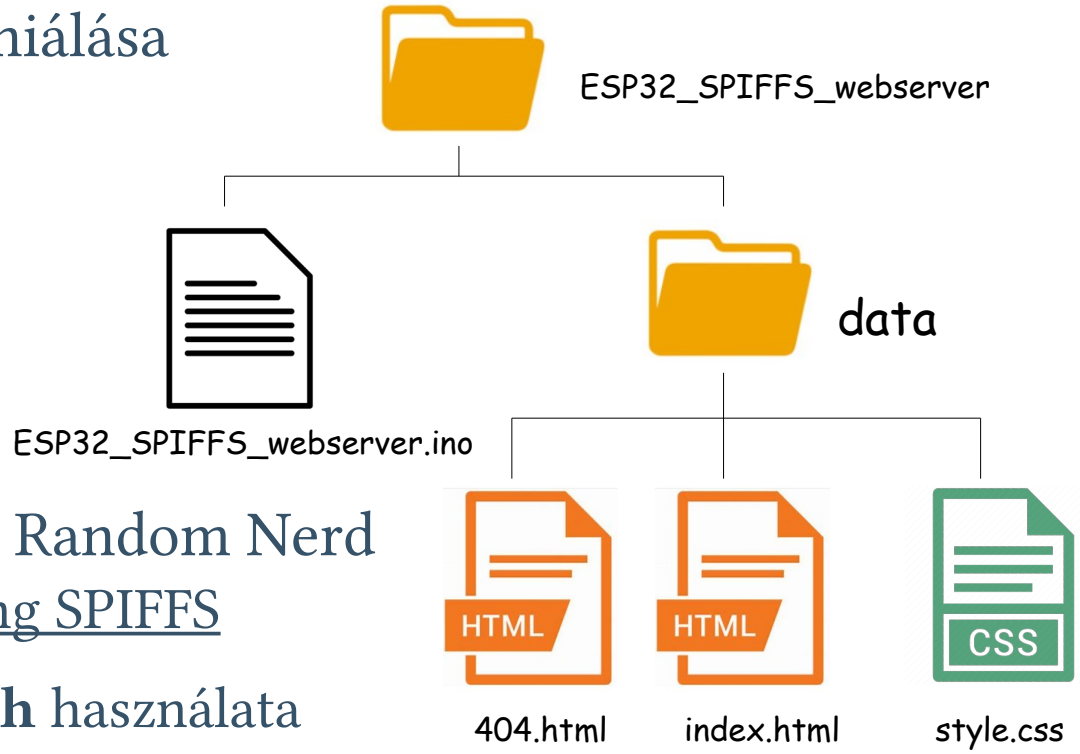
```
#include <WiFi.h>
#include <WebServer.h>
#include <SPIFFS.h>                   // Becsatoljuk az SPIFFS könyvtárat
#include "secrets.h"
WebServer server(80);                 // Webserver objektum példányosítása

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...
}
```

ESP32_SPIFFS_webserver

- A következő példában a beépített LED-et (**GPIO2**) kapcsolgatjuk
- Az **index.html,** a **style.css** és a **404.html** lapokat az **SPIFFS**-ben tároljuk (ezeket a projektünkben egy **data** mappában kell elhelyezni
- **index.html** – LED státusz kijelzése és két nyomógomb megjelenítése
- **404.html** – hibajelző oldal (**onNotFound** eseményhez rendelve)
- **style.css** – CSS stílusok definiálása
- A LED státuszát szövegesen jelezzük ki, ez a **processor**() függvényben történik a `%STATE%` sablon észlelésekor



A program eredetijének forrása: Random Nerd Tutorials – ESP32 Web Server using SPIFFS

Módosítások: **404.html** és **secrets.h** használata

ESP32_SPIFFS_webserver.ino 2/1.

```
#include "WiFi.h"
#include "ESPAsyncWebServer.h"
#include "SPIFFS.h"
#include "secrets.h"

const int ledPin = 2;        // Set LED GPIO
String ledState;             // Store LED state

AsyncWebServer server(80);

// Replaces placeholder with LED state value
String processor(const String& var){
  if(var == "STATE"){
    if(digitalRead(ledPin)){
      ledState = "ON";
    } else { ledState = "OFF";
    }
    Serial.print(ledState);
    return ledState;
  }
  return String();
}

void notFound(AsyncWebServerRequest *request) {
    request->send(SPIFFS, "/404.html", "text/html");
}
```

Ez a függvény fogja lecserélni a **%STATE%** sablont a LED aktuális állapotára (ON vagy OFF)

ESP32_SPIFFS_webserver.ino 2/2.

```
void setup(){
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
  if(!SPIFFS.begin(false)){
    Serial.println("An Error has occurred while mounting SPIFFS"); return;
  }
  setup_wifi();
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/style.css", "text/css");
  });

  server.on("/on", HTTP_GET, [](AsyncWebServerRequest *request){
    digitalWrite(ledPin, HIGH);
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  server.on("/off", HTTP_GET, [](AsyncWebServerRequest *request){
    digitalWrite(ledPin, LOW);
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  server.onNotFound(notFound);
  server.begin();
}
```

index.html és 404.html

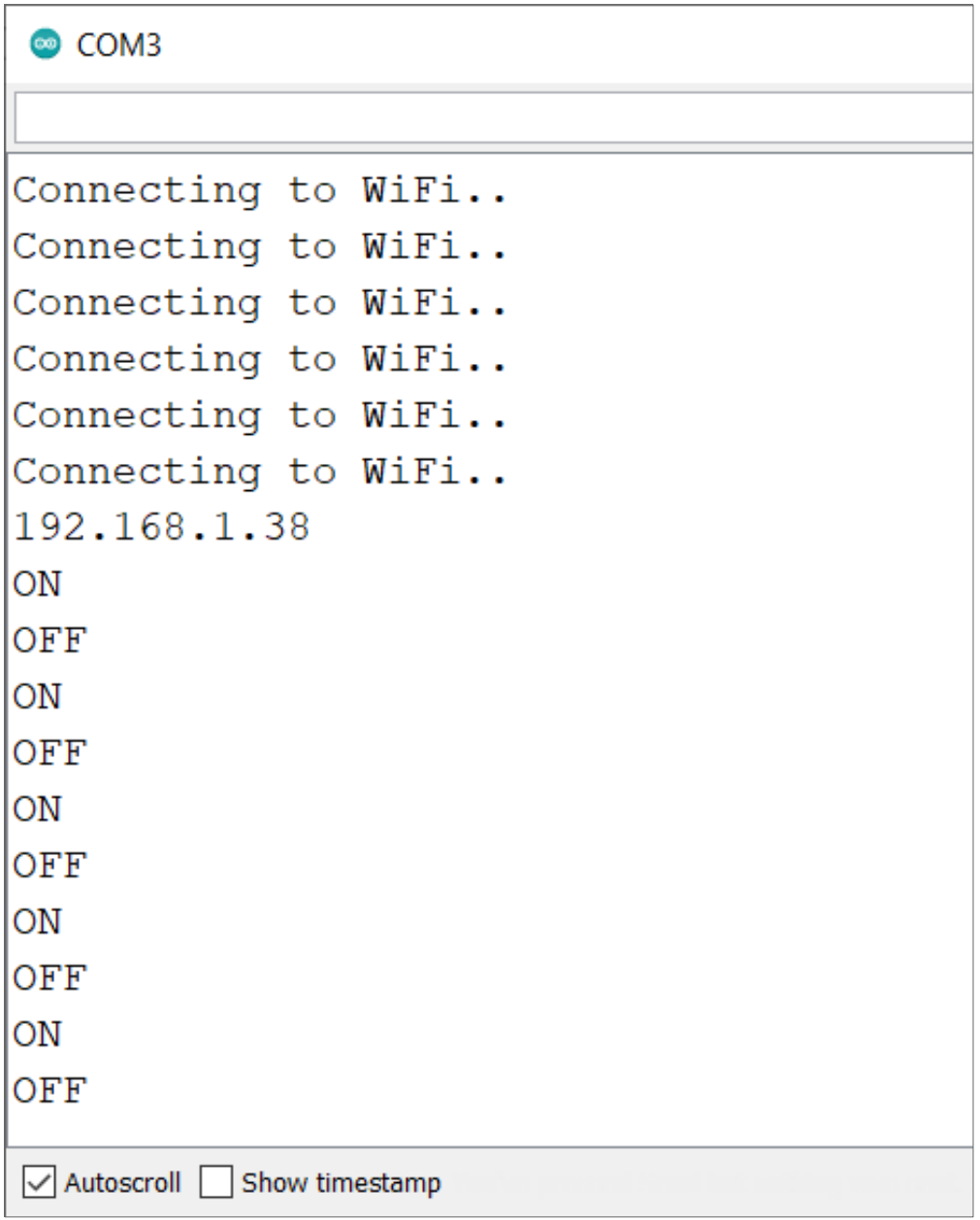
index.html

```
<!DOCTYPE html>
<html><head>
  <title>ESP32 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>ESP32 Web Server</h1>
  <p>GPIO state: <strong>%STATE%</strong></p>
  <p><a href="/on"><button class="button">ON</button></a></p>
  <p><a href="/off"><button class="button button2">OFF</button></a></p>
</body></html>
```

404.html

```
<!DOCTYPE html>
<html><head>
  <title>ESP32 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>404 Page not found</h1>
<h3>Press button to open start page</h3>
<a class="button button2" href="/">RESTART</a>
</body></html>
```

ESP32_SPIFFS_webserver futási eredmény



ESP32_WS_bme280_SPIFFS.ino

- A Last Minute Engineers **ESP32 Weather Station With BME280** projektben egy **ESP32** webszerver egy **BME280** szenzor mérési adatait jeleníti meg. Ezt a mintaprogramot vesszük most elő és alakítjuk át:
  - A `SendHTML()` függvény helyett **SPIFFS** fájlként, `index.html`, ill. `style.css` néven tároljuk a weblap kódját (HTML, stílus és JavaScript)
  - Készítettünk egy `404.html` hibajelző lapot is
  - Megjelöljük a cserélendő adatok helyét (`<span id="...">0</span>`)
  - Módosítjuk a **JavaScript** kódot, hogy az **AJAX** lekérés a a szenzor adatait egyetlen **JSON** üzenetként vegye át és írja be az adatokat a megjelölt helyre
  - **Átírjuk a webszerver programot** úgy, hogy külön kezelje a weblap lekérését és a mérési adatok lekérését (ne küldjük ki minden lekéréskor a teljes weblapot)
  - A **JSON** üzenet összeállításához az **ArduinoJson** könyvtárat használjuk

ESP32_WS_bme280_SPIFFS.ino 3/1.

```
#include <WiFi.h>
#include <WebServer.h>
#include <SPIFFS.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"                 // BME280 library header

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

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 client connections
}
```

ESP32_WS_bme280_SPIFFS.ino 3/2.

- A lekérések kiszolgálása ezekben a függvényekben történik:

Átszámítás tengerszintre (Debrecenben érvényes)

```
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>()); // Serialize and send JSON
}

void handle_NotFound() {
  if (!handleFileRead(server.uri()))                    // send file if it exists
    if(!handleFileRead("/404.html"))                    // otherwise send 404.html
      server.send(404, "text/plain", "File not found"); // or respond with plain text
}

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...
}
```

ESP32_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.

```
<html>
<head>
  <title>ESP32 Weather Station</title>
  <meta name='viewport' content='width=device-width, initial-scale=1.0'>
  <link rel="stylesheet" type="text/css" href="style.css">
  <link rel="icon" href="data:,">
  <link href='https://fonts.googleapis.com/css?family=Open+Sans:300,400,600'
rel='stylesheet'>
  <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:

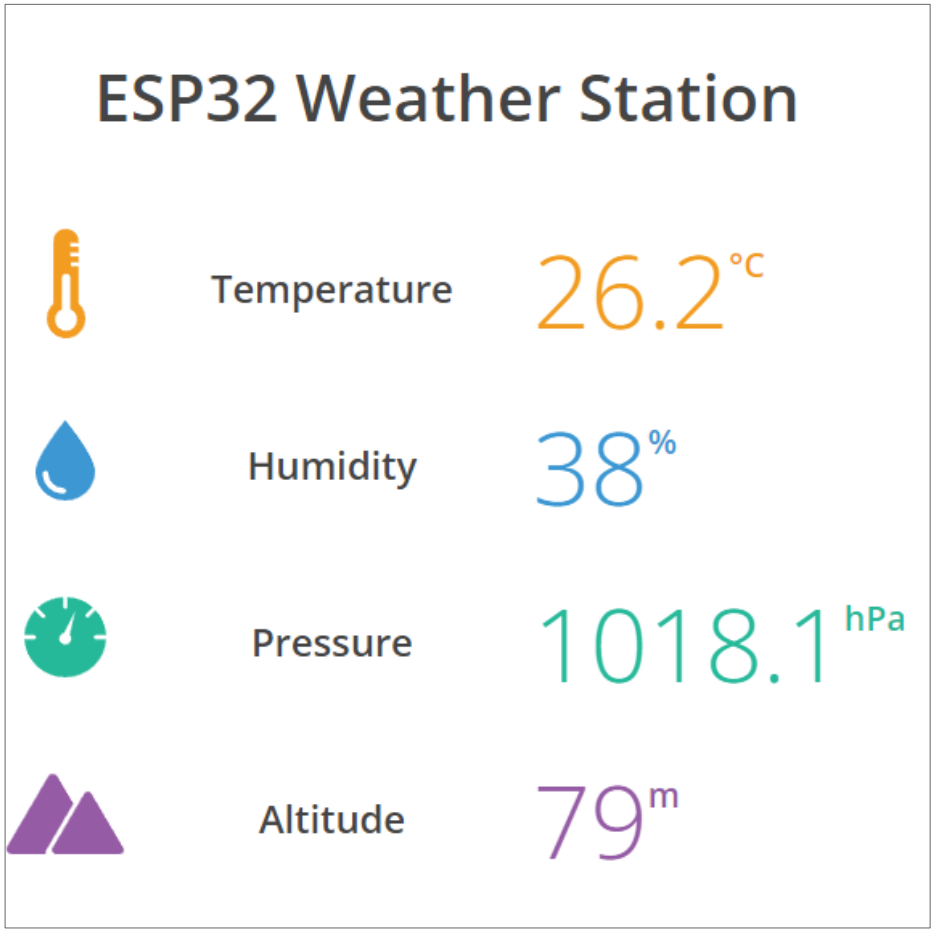
```
document.body.innerHTML = this.responseText;
```

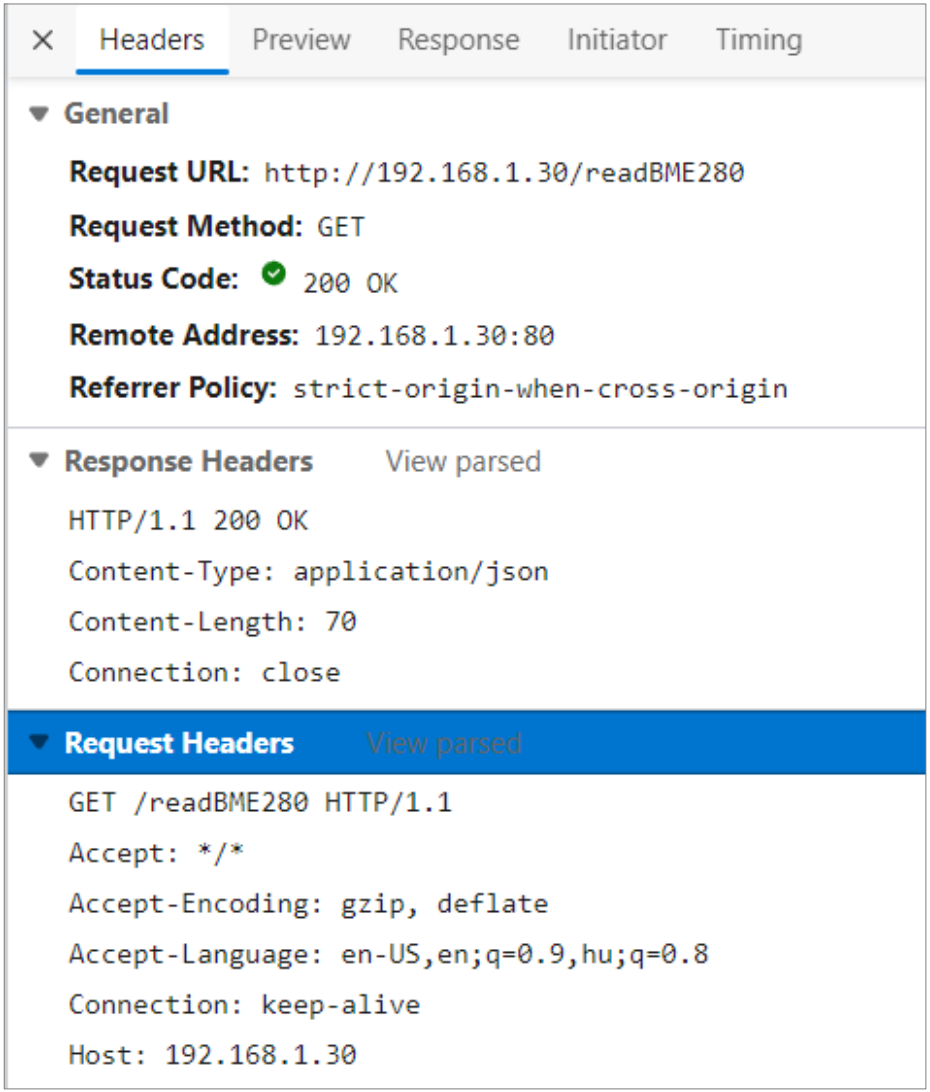
Az index.html állomány tartalma 2/2.

```
<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

{"temperature":"26.2","humidity":38,"pressure":"1018.1","altitude":79}





Az **F12** gomb megnyomásával hozzáférünk a böngésző adataihoz (lekérés és válasz)

A LittleFS fájlrendszer

- Kezdetben volt az ESP32 LITTLEFS programkönyvtár
- Ma már része az ESP32 Arduino Core csomagnak de átkeresztelték **LittleFS**-nek…
- Használata hasonló az SPIFFS-hez, de alkönyvtárakat is kezel

```
#include <Arduino.h>
#include "FS.h"
#include <LittleFS.h>
#define FORMAT_LITTLEFS_IF_FAILED true   // Csak első alkalommal kell

LittleFS.begin(FORMAT_LITTLEFS_IF_FAILED)
LittleFS.mkdir(path);                    // Könyvtár létrehozása
File file = LittleFS.open(filename, FILE_WRITE);     // Fájl létrehozása
file.print("Here is my message");        // Fájlba írás
file.close();                            // Fájl lezárása
File file = LittleFS.open(filename);     // Fájl megnyitása olvasásra
file.read();                             // Olvasás fájlból
file.available();                        // Van még olvasnivaló?
```

- Bővebben lásd az **ESP32_LittleFS_test.ino** gyári mintapéldánál

ESP32_LittleFS_listdir.ino

- Az alábbi program kilistázza a LittleFS fájlrendszer állományait

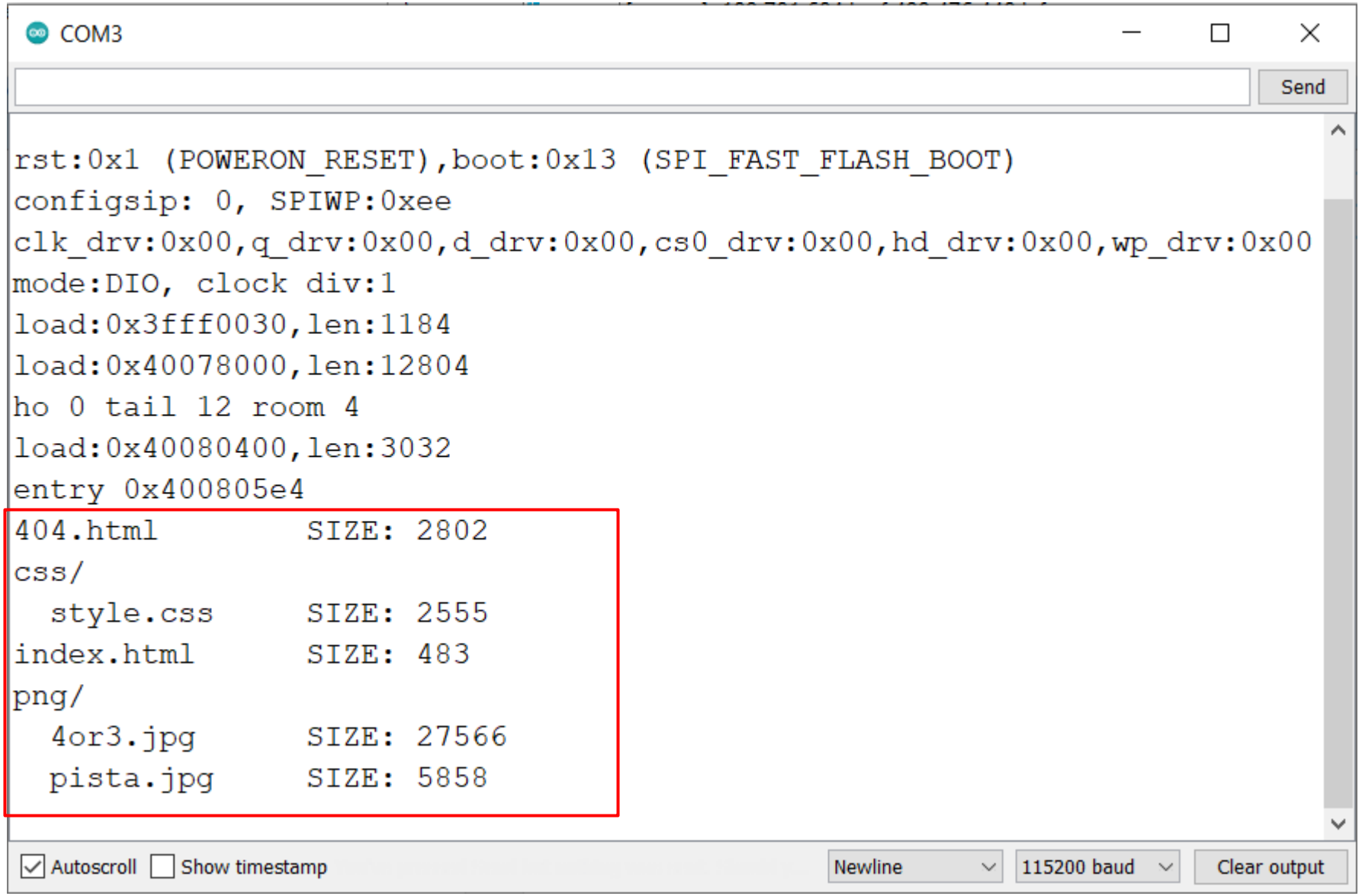
```
#include "LittleFS.h"

void listDir(const char * dirname, uint8_t levels) {
  String prettyprint = "";
  int i =0;
  while(i<levels) { prettyprint += "  "; i++; }
  File root = LittleFS.open(dirname);
  if (!root) { Serial.println("- failed to open directory"); return; }
  if (!root.isDirectory()) { Serial.println(" - not a directory"); return; }
  File file = root.openNextFile();
  while (file) {
    if (file.isDirectory()) {
      Serial.printf("%s%s/\r\n", prettyprint, file.name());
      listDir(file.path(), levels+1);
    } else {
      Serial.printf("%s%s\tSIZE: %d\r\n", prettyprint, file.name(), file.size());
    }
    file = root.openNextFile();
  }
}

void setup() {
  Serial.begin(115200);
  if (!LittleFS.begin(false)) { Serial.println("LittleFS Mount Failed"); return;  }
  listDir("/", 0);
}
```

ESP32_LittleFS_listdir futási eredménye

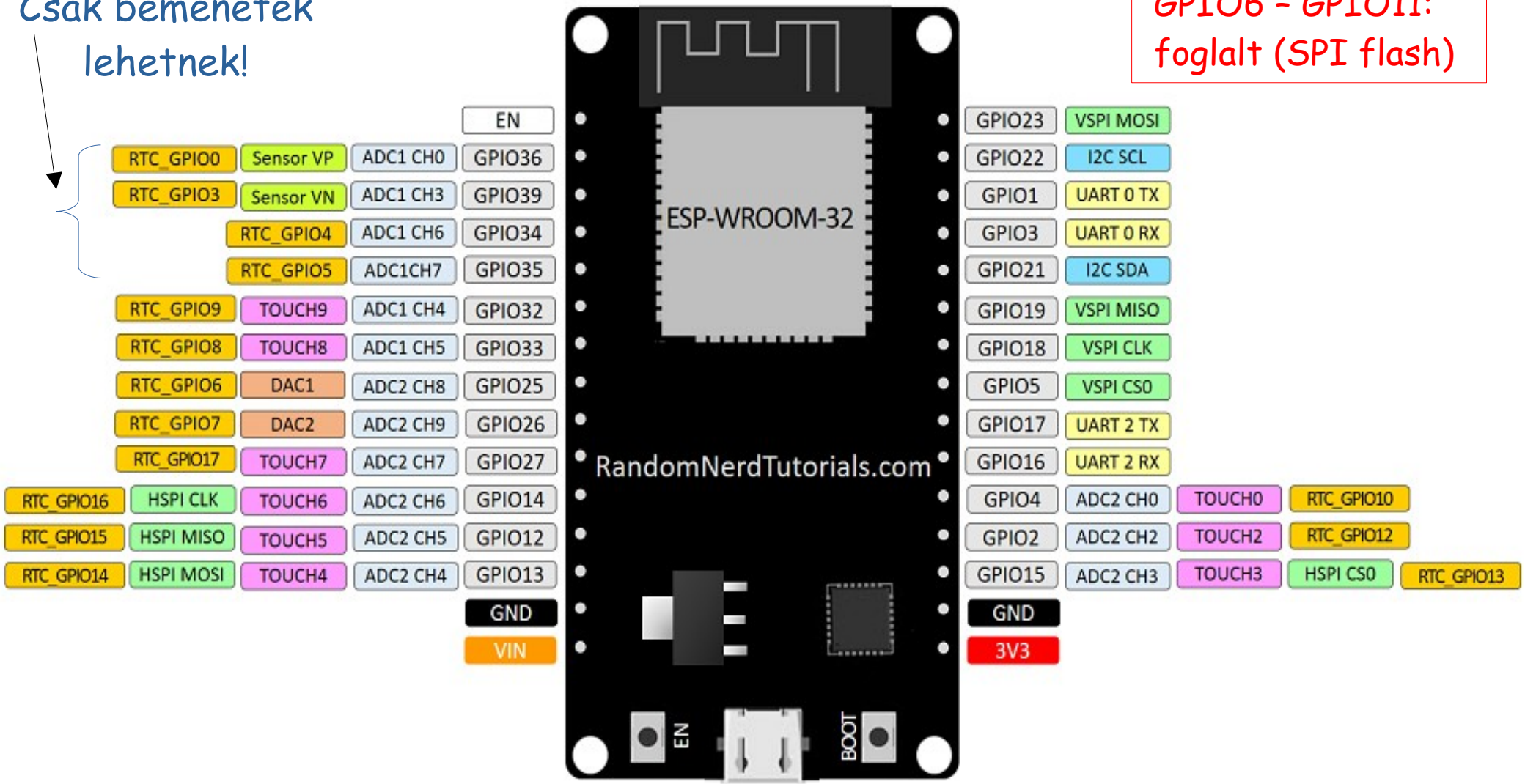
- A program futtatása előtt feltöltöttünk néhány fájlt



A DOIT ESP32 Devkit-1 kártya kivezetései

Csak bemenetek lehetnek!

GPIO6 - GPIO11: foglalt (SPI flash)



- **Forrás:** randomnerdtutorials.com/getting-started-with-esp32/

Ellenállás színkódok