Beágyazott eszközök programozása ZH.

Soft AP mode (hozzáférési pontként)

#include <ESP8266WiFi.h>

- SoftAP mód indításhoz: WiFi.softAP(ssid, password, channel, hidden, max_connection)
  ssid:string
  password:legalább 8 karakter (ha üres akkor publikus)
  channel:1-13 alapértelmezetten 1
  hidden:ssid elrejtése (true/false)
  max connection:0-8, alapértelmezetten 4 (max 5 kliens csatlakozhat)

- Wifi.softAPConfig(local_ip, gateway, subnet)
  local_ip:eszköz ip címe
  gateway:alapértelmezett átjáró
  subnet:alhálózati maszk
  visszatérési értéke a beállításról true/false

EEPROM (valamilyen adat eltárolása/előhívása EEPROM memóriából)

```
#define EEPROM_SIZE 4096          //4kB
#define EEPROM_WATERMARK 197      //8bit érték. [0,255]
#define EEPROM_LOC_WM 0           //vízjel helye az EEPROM-ba
#define EEPROM_DATA   4           //adat helye
void setup()-ba : init_EEPROM();
void init_EEPROM(){
int wm, data;
EEPROM.begin(EEPROM_SIZE);
wm = EEPROM.read(EEPROM_LOC_WM);
if(EEPROM_WATERMARK != wm)
{
//első inicializálás
EEPROM.write(EEPROM_LOC_WM, EEPROM_WATERMARK);
data=42;
EEPROM.put(EEPROM_LOC_DATA, data);
EEPROM.commit();
}
else()
{
//ha már inicializálva van(használva van a vízjel/nem az első használat)
EEPROM.get(EEPROM_LOC_DATA, data);
Serial.printf("DATA was: %d\n", data);
}
```

Nyers (RAW) string literál

Eltárolható vele nyers szöveg. R karakterrel kell jelölni, illetve valamilyen nyitó, záró karaktersorozatal.

Például:
```
  String raw_html = R"(
<div class="city">
<h2>London</h2>
</div>
)";
```

PROGMEM

Használatával a változót a programmemóriába tudjuk betölteni.
(Vagy globális változó, vagy statikus.)
```
  const PROGMEM dataType variableName[] = {};
```

F() MACRO

Debugolásnál hasznos lehet, ha nem az SRAM-ban tárolunk adatot, hanem a programmemóriában.
```
Serial.print(F("Ez egy debug üzenet a fejlesztő számára"));
```

STA mód

Létező WiFi hálózathoz tudunk kapcsolódni a segítségével.

Fetch() - Promise

A Promise egy JS objektum, aszinkron művelet állapotát, eredményét kezeli.
Három állapota lehet(pending,fulfilled,rejected)
Ide tartozik az async/await alkalmazása

SPIFF

Flash memória, lehetőség van a firmware mellett fájlrendszer kialakítására is.
Fájlnevek max 31 karakterből állhatnak, az utolsó karakternek '\0' kell lennie.
(Fontos:Elérési út esetén is érvényes a max. 31 karakter!)
Spiff függvények:

SPIFFS.begin() – fájlrendszer csatolása
SPIFFS.end() – fájlrendszer leválasztása
SPIFFS.format() – fájlrendszer formázása, siker esetén a visszatérési érték „true”;
  SPIFFS.info(FSinfo info) – fájlrendszer tulajdonságainak a lekérése
File f = SPIFFS.open(path, mode) – fájl megnyitása (mode: r – read, w write, a- append)
Dir dir = SPIFFS.openDir(„/”); - gyökérkönyvtár megnyitása!

ESP8266 Filesystem Uploader

https://github.com/esp8266/arduino-esp8266fs-plugin/releases
Kicsomagolod, majd Documents/Programs/arduion-1.8.9/tools (ha ezt a verziót használjuk)
Majd újra kell indítani, Tools menüben ESP8266 Sketch Data Upload.
Ennek a segítségével lehet adatokat menteni, HTML, CSS fájlokat, médiát tárolni.
Használata: elmentjük az Arduino Sketchet(adott projekt), a mappájában a .ino fájl mellé létrehozunk egy data mappát, az ebbe elhelyezett fájlokat tudjuk feltölteni.

SPIFF PÉLDA FELADAT

script.js

```
async function setLed(state) {
try {
const response = await fetch("/led", {
method: "POST",
headers: { "Content-Type":    "application/x-www-form-urlencoded"},
body: `state=${state}`
});
if (!response.ok) {
throw new Error(`Hiba: ${response.status} - ${response.statusText}`);
}
const data = await response.text();
console.log("Szerver válasza:",data);
} catch(error){
console.error("Hiba történt:", error);
```

style.css

```
body {
/*ide kerül a formázás*/
}
```

index.html

html nyitótag, stb (generálod, ha pl. VSCode-ot használsz, vagy valamilyen szerkesztőt)
majd

```
<body>
<button onclick="setLed('on')">LED BE </button>
<button onclick="setLed('off')">LED KI </button>
</body>
```

.ino fájl (Arduino)

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

```
#include <ESPAsyncWebServer.h>
#include <FS.h>

const char* ssid = "BIR_WIFI";
const char* password = "OeBir2019";
AsyncWebServer server(80);


void handleLed (AsyncwebServerRequest* request) {
// Ha a paraméter 'state' létezik a bodyban
if (request->hasParam("state", true)) {
String state= request->arg("state"); // LED vezérlés
if (state == "on") {
        digitalWrite(LED_BUILTIN, LOW); // LED bekapcsolása }
else if (state == "off") {
        digitalWrite(LED_BUILTIN, HIGH); // LED kikapcsolása
}
// Válasz küldése
request->send (200, "text/plain", "LED állapot: + state);
} else {
        // Ha nincs 'state' paraméter, hibát küldünk
        request->send(400, "text/plain", "Hibás kérés");
}
}
void notFound(AsyncWebServerRequest* request){
String message = "Nem található";
request -> send(404, "text/plain", message);
}
void setup(){
Seria.begin(115200);
while (!Serial){
delay(100);
}
pinMode(LED_BUILTIN, OUTPUT);
Serial.println();
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("\nCsatlakozva! IP: " +WiFi.localIP().toString());
if (!SPIFFS.begin()){
```

```
Serial.println("SPIFFS inicializálás sikertelen!");
return;
}
server.serveStatic("/", SPIFFS, "/").setDefaultFile("index.html");
server.on("/led", HTTP_POST, handleLed);
server.onNotFound(notFound);
server.begin();
}
void loop(){}
```

SOFT AP LED VEZÉRLÉS PÉLDA FELADAT

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
String ssid = "SAJATWIFI";
String pass ="jelszo1234";
IPAddress local_ip(192,138,1,1);
IPAddress gateway(192,138,1,1);
IPAdress subnet(255,255,255,0);
ESP8266WebServer server(80); //port beállítás
#define LED1pin LED_BUILTIN
bool LED1status = LOW;
#define LED2pin D6
bool LED1status = LOW;
void setup(){
Serial.begin(115200);
pinMode(LED1pin, OUTPUT);
pinMode(LED2pin, OUTPUT);
WiFi.softAP(ssid, password);
WiFi.softAPConfig(local_ip, gateway, subnet);
delay(100);
server.on("/", handle_OnConnect);
server.on("/led1on", handle_led1on);
server.on("/led1off", handle_led1off);
server.on("/led2on", handle_led2on);
server.on("/led2off", handle_led2off);
server.onNotFound(handle_NotFound);
server.begin();
Serial.println("HTTP server started");
}
void loop(){
server.handleClient();
```

```
if(LED1status)
{digitalWrite(LED1pin, HIGH);}
else
{digitalWrite(LED1pin, LOW);}

if(LED2status)
{digitalWrite(LED2pin, HIGH);}
else
{digitalWrite(LED2pin, LOW);}

void handle_OnConnect(){
LED1status=LOW;
LED2status=LOW;
server.send(200, "text/html", SendHTML(LED1status, LED2status));
}
void handle_led1on(){
LED1status =HIGH;
server.send(200,"text/html", SendHTML(true,LED1status));
}
void handle_led1off(){
LED1status =LOW;
server.send(200,"text/html", SendHTML(false,LED1status));
}
void handle_led2on(){
LED2status =HIGH;
server.send(200,"text/html", SendHTML(true,LED2status));
}
void handle_led2off(){
LED2status =LOW;
server.send(200,"text/html", SendHTML(false,LED2status));
}
void handle_NotFound(){
server.send(404, "text/plain", "Not found")
}
```

Itt a html weboldalt még string változóba hozzuk létre az Arduino projektben, nem úgy mint az előző példában, ahol külön fájlokat kezeltünk.

```
String SendHTML(uint8_t led1stat, uint8_t led2stat)
{
String ptr = "<!DOCTYPE html><html>\n";
ptr +="<head>\n";
ptr +="<title>LED Control </title>\n"
```

```
ptr +="</head>\n";
if (led1stat){
ptr += "<p>LED1 status: OFF </p>";
ptr += "<a href=\"/led1off\"><button>OFF</button></a>\n";
}
else{
ptr += "<p> LED1 status:ON </p><a href=\"/led1on\">";
ptr+= "<button>ON</button></a>\n";
}
if (led2stat){
ptr += "<p>LED2 status: OFF </p>";
ptr += "<a href=\"/led2off\"><button>OFF</button></a>\n";
}
else{
ptr += "<p> LED2 status:ON </p><a href=\"/led2on\">";
ptr+= "<button>ON</button></a>\n";
}
ptr += "</body>\n";
ptr += "</html>\n";
return ptr;
}
```