

Dunaújvárosi Egyetem Bánki Donát Technikum

Projekt feladat dokumentáció

Az ötlet rövid leírása:	2
Arduino kód	2
Index.html kód	2
Program működésének ábrázolása:	3
Serial Monitor:	4
Interneten keresztül (telefonon):	4
Továbbfejlesztési lehetőségek:	4
Önreflexió:	5

Tantárgy neve: IoT

Projekt címe: Hőmérséklet-páratartalom Web Plotter

Osztály: 13.B

Dátum: 2025.11.19

Az ötlet rövid leírása:

Ebben a projektben egy ESP8266 alapú eszközt fejlesztettem, amely a környezeti paraméterek (hőmérséklet és páratartalom) mérésére és távoli, valós idejű monitorozására szolgál. A rendszer egy webes felületen keresztül, dinamikus grafikonok segítségével teszi lehetővé az adatok elemzését.

Arduino kód

```
1  #include <ESP8266WiFi.h>
2  #include <ESP8266WebServer.h>
3  #include <WebSocketsServer.h>
4  #include "index.h"
5  #include <DHT.h>
6
7
8
9  const char* ssid = "Igen";
10 const char* password = "12345678";
11 #define DHTPIN 2
12 #define DHTTYPE DHT22
13 ESP8266WebServer server(80);
14 WebSocketsServer plotter = WebSocketsServer(81);
15 DHT dht(DHTPIN, DHTTYPE);
16 unsigned long last_update = 0;
17 const unsigned long UPDATE_INTERVAL = 2000;
18 void setup()
19 {
20   Serial.begin(115200);
21   delay(500);
22   dht.begin();
23   WiFi.begin(ssid, password);
24   while (WiFi.status() != WL_CONNECTED) {
25     delay(1000);
26     Serial.println("Connecting to WiFi...");
27   }
28
29   Serial.println("Connected to WiFi");
30   Serial.print("IP: ");
31
32   Serial.println(WiFi.localIP());
33   plotter.begin();
34   server.on("/", HTTP_GET, []() {
35     server.send(200, "text/html", HTML_CONTENT);
36   });
37   server.begin();
38
39   Serial.println("HTTP server started");
40
41 }
42
43 void loop() {
44   server.handleClient();
45   plotter.loop();
46   unsigned long now = millis();
47   if (now - last_update > UPDATE_INTERVAL) {
48     last_update = now;
49     float humidity = dht.readHumidity();
50     float temperature = dht.readTemperature();
51
52     // Hiba olvasás esetén
53     if (isnan(humidity) || isnan(temperature)) {
54       Serial.println("Failed to read from DHT sensor!");
55       return;
56     }
57
58     // Serial Plotter
59     Serial.print(temperature, 2);
60     Serial.print(" ");
61     Serial.println(humidity, 2);
62
63     // Serial Monitor
64     Serial.print("Temp: "); Serial.print(temperature, 2);
65     Serial.print(" °C\tHum: "); Serial.println(humidity, 2);
66
67     // Websocket szerver
68
69     String payload = String(temperature, 2) + "\t" + String(humidity, 2);
70     plotter.broadcastTXT(payload);
71   }
72 }
```

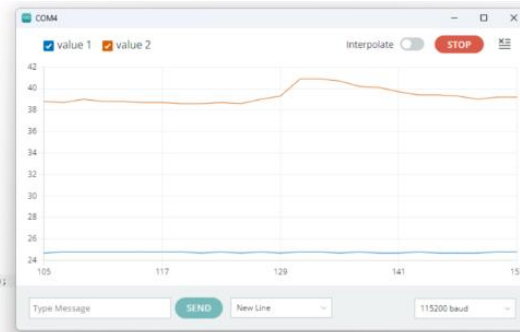
Index.html kód

```
1  const char *HTML_CONTENT = R"=====(
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <title>ESP8266 Temp & Humidity Plotter</title>
6      <meta name="viewport" content="width=device-width, initial-scale=0.7">
7      <style>
8          body { text-align: center; }
9          #graph { width: 400px; height: 200px; border: 1px solid #000; }
10         button { font-size: 14pt; margin-top: 10px; }
11     </style>
12 </head>
13 <body>
14     <h1>ESP8266 Web Plotter</h1>
15     <canvas id="graph"></canvas><br>
16     <div>
17         <button id="btn_connect">Connect</button>
18         <div id="ws_state">CLOSED</div>
19     </div>
20 <script>
21     var websocket;
22     var canvas = document.getElementById("graph");
23     var ctx = canvas.getContext("2d");
24     var data = [[], []]; // two lines: [0] = temp, [1] = humidity
25     var MAX_SAMPLE = 50;
26     function draw() {
27         ctx.clearRect(0, 0, canvas.width, canvas.height);
28         // draw grid
29         ctx.strokeStyle = "#eee";
30         for (var i = 0; i <= 5; i++) {
31             ctx.beginPath();
32             ctx.moveTo(0, i * canvas.height / 5);
33             ctx.lineTo(canvas.width, i * canvas.height / 5);
34             ctx.stroke();
35         }
36         var colors = ["#FF0000", "#0000FF"]; // red = temp, blue = hum
37         for (var line = 0; line < data.length; line++) {
38             ctx.strokeStyle = colors[line];
39             ctx.beginPath();
40
41             data[line].forEach(function(val, i) {
42                 var x = (i / (MAX_SAMPLE - 1)) * canvas.width;
43                 // scale Y: assume 0-100 for simplicity
44                 var y = canvas.height - (val / 100) * canvas.height;
45                 if (i == 0) ctx.moveTo(x, y);
46                 else ctx.lineTo(x, y);
47             });
48             ctx.stroke();
49         }
50     }
51     function connectWS() {
52         if (!websocket) {
53             websocket = new WebSocket("ws://" + window.location.hostname + ":81");
54             document.getElementById("ws_state").innerHTML = "CONNECTING";
55             websocket.onopen = function() {
56                 document.getElementById("ws_state").innerHTML = "<span style='color: green;'>CONNECTED</span>";
57                 document.getElementById("btn_connect").innerText = "Disconnect";
58             };
59             websocket.onclose = function() {
60                 document.getElementById("ws_state").innerHTML = "<span style='color: grey;'>CLOSED</span>";
61                 document.getElementById("btn_connect").innerText = "Connect";
62                 websocket = null;
63             };
64             websocket.onmessage = function(evt) {
65                 var msg = evt.data;
66                 // parse lines
67                 msg = msg.replace(/\r\n/g, "\n");
68                 var lines = msg.split("\n");
69                 lines.forEach(function(l) {
70                     if (l.trim().length == 0) return;
71                     var parts = l.split("\t");
72                     if (parts.length >= 2) {
73                         var t = parseFloat(parts[0]);
74                         var h = parseFloat(parts[1]);
75                         data[0].push(t);
76                         data[1].push(h);
77                         if (data[0].length > MAX_SAMPLE) data[0].shift();
78                         if (data[1].length > MAX_SAMPLE) data[1].shift();
79                     }
80                 });
81                 draw();
82             } else {
83                 websocket.close();
84             }
85         }
86         document.getElementById("btn_connect").onclick = connectWS;
87         canvas.width = 400;
88         canvas.height = 200;
89         draw();
90 </script>
91 </body>
92 </html>
93 )=====";
```

Program működésének ábrázolása:

Serial Monitor:

```
189 if (now - last_update > UPDATE_INTERVAL) {
190   last_update = now;
191
192   float humidity = dht.readHumidity();
193   float temperature = dht.readTemperature();
194
195   // Check if readings failed
196   if (isnan(humidity) || isnan(temperature)) {
197     Serial.println("Failed to read from DHT sensor!");
198     return;
199   }
200
201   // --- Serial Plotter (numeric for plotting) ---
202   Serial.print(temperature, 2); // temperature as float
203   Serial.print(" "); // space separator
204   Serial.println(humidity, 2); // humidity as float
205
206   // --- Serial Monitor (human-readable) ---
207   Serial.print("Temp: "); Serial.print(temperature, 2);
208   Serial.print(" "); Serial.print("Hum: "); Serial.print(humidity, 2);
209
210   // --- Websocket / web plotter (with labels) ---
211   String payload = String(temperature, 2) + "\t" + String(humidity, 2);
212   plotter.broadcastTXT(payload);
213
214 }
215 }
```



Interneten keresztül (telefonon):

ESP8266 Web Plotter



Továbbfejlesztési lehetőségek:

- SD-kártyás modul: Adatok mentése helyben, CSV formátumban.
- Adatbázis integráció: Az adatok küldése egy távoli szerverre (pl. InfluxDB vagy MySQL) és megjelenítése Grafana dashboardon.
- EEPROM használata: A szélsőséges értékek (minimum/maximum) elmentése a memóriába, hogy áramszünet után is megmaradjanak.

Önreflexió:

A projekt kidolgozását kifejezetten élveztem, mivel lehetőséget adott a beágyazott rendszerek és a webes technológiák összehangolására. Különösen izgalmas volt látni, ahogy a szenzor által mért fizikai környezet változásai a WebSocket technológiának köszönhetően szinte azonnal, látványos grafikonok formájában megjelentek a böngészőben.

Ez a fajta valós idejű webes megjelenítés nemcsak esztétikailag tette vonzóbbá a rendszert, hanem funkcionálisan is magasabb szintre emelte a projektet a hagyományos, statikus megoldásokhoz képest. A fejlesztés során sikerült elmélyítenem a tudásomat az adatátviteli protokollok és az interaktív felületek kialakítása terén, ami a jövőbeli IoT-munkáimhoz is kiváló alapot ad.