Remote Weather Station - NIŢU Gabriel

Introducere

Proiectul consta intr-o statie meteo remote care aduna informatii despre temperatura, umiditate, presiunea atmosferica, nivelul de intensitate luminoasa si predictie a ploii si transmite toate aceste date prin wifi catre un webserver. Webserverul le afiseaza intr-un mod interactiv si usor de vizualizat si analizat pe un dashboard.

Varianta finala si functionala a proiectului se poate vedea pe urmatorul link https://youtu.be/XK-8px-DiMA?feature=shared

Descriere generală

Schema bloc

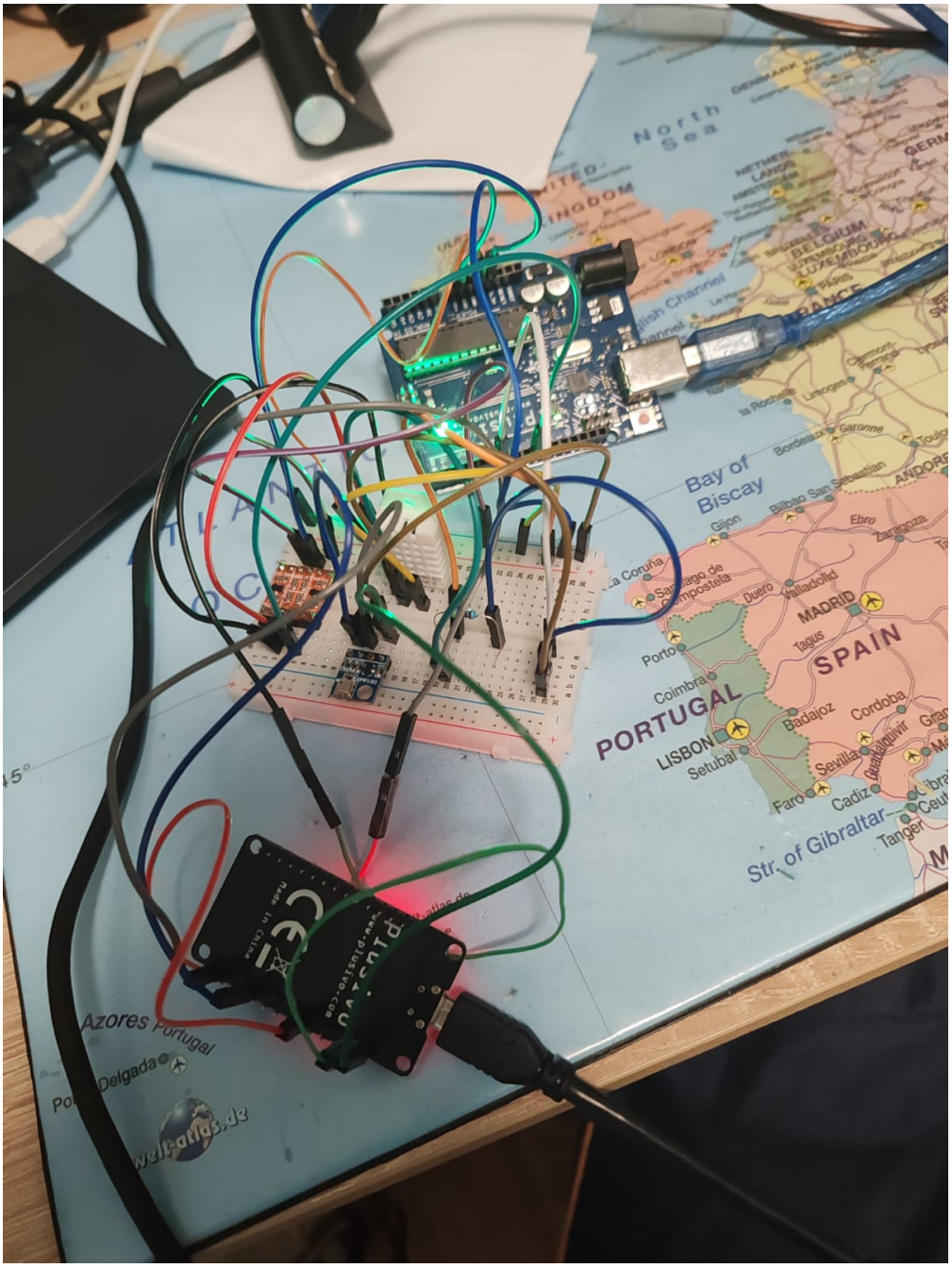
Hardware Design

- Listă de piese

Componenta	Cantitate	Link
Placa de Dezvoltare ESP32 cu WiFi și Bluetooth 4.2	1	https://www.optimusdigital.ro/ro/placi-cu-bluetooth/4371-placa-de-dezvoltare-esp32-cu-wifi-i-bluetooth-42.html?search_query=Placa+de+Dezvoltare+ESP32+cu+WiFi+%C8%99i+Bluetooth+4.2+&results=8
Placa de Dezvoltare Compatibila cu Arduino UNO R3	1	https://www.optimusdigital.ro/ro/placi-avr/4561-placa-de-dezvoltare-compatibila-cu-arduino-uno-r3-atmega328p-atmega16u2-cablu-50-cm.html?search_query=arduino+uno&results=129
Modul senzor de Temperatura si Presiune BMP180	1	https://www.optimusdigital.ro/ro/senzori-senzori-de-presiune/149-modul-senzor-de-temperatura-si-presiune-bmp180.html?search_query=bmp180&results=7
Senzor de Temperatura și Umiditate DHT22	1	https://www.optimusdigital.ro/ro/senzori-senzori-de-temperatura/1199-senzor-de-temperatura-i-umiditate-dht22.html?search_query=dht+22&results=25
Level Shifter cu 2 canale	2	https://www.optimusdigital.ro/ro/interfata-convertoare-de-niveluri/12562-convertor-de-nivel-logic-cu-2-canale-33v-5v-ttl.html?search_query=convertor+de+nivel&results=32
Breadboard	1	https://www.optimusdigital.ro/ro/prototipare-breadboard-uri/44-breadboard-400-points.html?search_query=breadboard&results=125

Schema electrica

Cablajul



Software Design

Mediul de dezvoltare:

- Arduino IDE
- VS Code

Pe Arduino am implementat citirea senzorului DHT si trimiterea informatiilor catre ESP32.

```
SoftwareSerial espSerial(rxPin, txPin);

void setup() {
  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);
```

```
  Serial.begin(9600);
  espSerial.begin(9600);
  dht.begin();
}

void loop() {
  Serial.println("Sending to ESP32...");
  int16_t  value = 1234;


  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  espSerial.println(temperature);
  espSerial.println(humidity);

  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" *C");

  delay(2000);
}
```

Pe ESP32 citesc informatiile de la senzorul bmp, primesc datele de la Arduino si le trimit pe toate prin Wifi catre un server implementat de mine.

```
void setup() {
  Serial.begin(9600);  // For Serial Monitor
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2); // UART2
  int counter = 0;

  delay(2000);  // Wait for power to stabilize
  WiFi.disconnect(true);  // Clear previous settings
  delay(1000);

  Serial.println("Scanning Wi-Fi...");
  int networks = WiFi.scanNetworks();

  if (networks == 0) {
    Serial.println("No networks found.");
```

```
  } else {
    Serial.println("Networks found:");
    for (int i = 0; i < networks; ++i) {
      Serial.println(WiFi.SSID(i));
    }
  }

  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" Connected!");


   Wire.begin(21, 22); // SDA = 21, SCL = 22
   if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP180 sensor, check wiring!");
    while (1) {}
  }

}

void loop() {
    while (Serial2.available() > 0) {
        Serial2.read();
    }
    delay(2000);
    String temperature = "";
    String humidity = "";

    temperature = Serial2.readStringUntil('\n');  // Read first line
    temperature.trim();

    humidity = Serial2.readStringUntil('\n');  // Read second line
    humidity.trim();


    float p = bmp.readPressure();
    String pressure = String(p);

    float a = bmp.readAltitude();
    String altitude = String(a);

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverURL);
    http.addHeader("Content-Type", "application/json");

    String jsonPayload = "{\"temperature\":" + temperature +
```

```
                          ",\"humidity\":" + humidity +
                          ",\"pressure\":" + pressure +
                          ",\"altitude\":" + altitude +
                          "}";
    Serial.println(jsonPayload);
    int httpResponseCode = http.POST(jsonPayload);
    Serial.println("HTTP Response code: " + String(httpResponseCode));

    String response = http.getString();
    Serial.println("Server response: " + response);


    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println(httpResponseCode);
      Serial.println(response);
    } else {
      Serial.print("Error in sending POST: ");
      Serial.println(http.errorToString(httpResponseCode).c_str());
    }

    http.end();
  }

}
```

Am implementat un server in Python folosind Flask, aceasta fiind partea de backend.

```


from flask import Flask, request, jsonify, render_template
from datetime import datetime

app = Flask(__name__)

latest_data = {
    "temperature": None,
    "humidity": None,
    "pressure": None,
    "timestamp": None
}

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/update', methods=['GET'])
def get_data():
    return jsonify(latest_data)

@app.route('/data', methods=['POST'])
```

```
def receive_data():
    data = request.get_json(force=True)
    if data:
        latest_data["temperature"] = float(data.get("temperature", 0))
        latest_data["humidity"] = float(data.get("humidity", 0))
        latest_data["pressure"] = float(data.get("pressure", 0))
        latest_data["timestamp"] = datetime.now().strftime("%H:%M:%S")
        return jsonify({"status": "success"}), 200
    return jsonify({"status": "fail"}), 400

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Pentru a vizualiza datele in mod real intr-un mod cat mai placut si interactiv, am imple,mentat si partea de frontend a serverului, in HTML si JavaScript.

HTML:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Sensor Dashboard</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/chartjs-plugin-streaming"></script>
    <script
src="https://cdn.jsdelivr.net/npm/@bernaferrari/gauge-chart"></script>
    <script defer src="/static/script.js"></script>
    <style>
        body { font-family: sans-serif; text-align: center; padding: 20px;
background: #f5f5f5; }
        h1 { color: #333; }
        .container { display: flex; flex-wrap: wrap; justify-content: center;
gap: 20px; }
        .card { background: white; padding: 20px; border-radius: 10px;
box-shadow: 0 4px 10px rgba(0,0,0,0.1); }
        canvas { max-width: 600px; }
        .gauge-container { width: 250px; height: 150px; margin: auto; }
    </style>
</head>
<body>
    <h1>□□ Real-Time Sensor Dashboard</h1>

    <div class="container">
        <div class="card">
            <h3>Temperature Gauge</h3>
            <div id="gauge-temp" class="gauge-container"></div>
            <p>Temp: <span id="temp">--</span> °C</p>
        </div>
```

```
        <div class="card">
            <h3>Humidity Gauge</h3>
            <div id="gauge-hum" class="gauge-container"></div>
            <p>Humidity: <span id="hum">--</span> %</p>
        </div>
    </div>

    <div class="container">
        <div class="card">
            <h3>Temperature - Last 10 Minutes</h3>
            <canvas id="chart-temp" height="100"></canvas>
        </div>

        <div class="card">
            <h3>Humidity - Last 10 Minutes</h3>
            <canvas id="chart-hum" height="100"></canvas>
        </div>
    </div>

    <div class="card">
        <h3>Pressure</h3>
        <p> Pressure: <span id="pres">--</span> Pa</p>
        <p> Time: <span id="time">--:--:--</span></p>
    </div>
</body>
</html>
```

JavaScript:

```
const tempSpan = document.getElementById('temp');
const humSpan = document.getElementById('hum');
const presSpan = document.getElementById('pres');
const timeSpan = document.getElementById('time');

const gaugeTemp =
GaugeChart.gaugeChart(document.getElementById("gauge-temp"), {
  arcDelimiters: [0.3, 0.6, 1],
  arcColors: ["#00bfff", "#ffa500", "#ff0000"],
  arcLabels: ["Low", "Medium", "High"],
  rangeLabel: ["0°C", "50°C"],
  centralLabel: ""
});

const gaugeHum = GaugeChart.gaugeChart(document.getElementById("gauge-hum"),
{
  arcDelimiters: [0.3, 0.6, 1],
  arcColors: ["#add8e6", "#00ced1", "#007bff"],
  arcLabels: ["Dry", "Moderate", "Humid"],
  rangeLabel: ["0%", "100%"],
  centralLabel: ""
});
```

```
const ctxTemp = document.getElementById('chart-temp').getContext('2d');
const ctxHum = document.getElementById('chart-hum').getContext('2d');

const tempChart = new Chart(ctxTemp, {
  type: 'line',
  data: {
    datasets: [{
      label: 'Temperature (°C)',
      borderColor: 'rgba(255, 99, 132, 1)',
      backgroundColor: 'rgba(255, 99, 132, 0.2)',
      data: []
    }]
  },
  options: {
    plugins: {
      streaming: {
        frameRate: 30
      }
    },
    scales: {
      x: {
        type: 'realtime',
        realtime: {
          duration: 600000,
          refresh: 2000,
          delay: 2000,
          onRefresh: async chart => {
            await fetchAndUpdate(chart, 'temperature');
          }
        }
      },
      y: {
        beginAtZero: true
      }
    }
  }
});

const humChart = new Chart(ctxHum, {
  type: 'line',
  data: {
    datasets: [{
      label: 'Humidity (%)',
      borderColor: 'rgba(54, 162, 235, 1)',
      backgroundColor: 'rgba(54, 162, 235, 0.2)',
      data: []
    }]
  },
  options: {
    plugins: {
      streaming: {
```

```
        frameRate: 30
      }
    },
    scales: {
      x: {
        type: 'realtime',
        realtime: {
          duration: 600000,
          refresh: 2000,
          delay: 2000,
          onRefresh: async chart => {
            await fetchAndUpdate(chart, 'humidity');
          }
        }
      },
      y: {
        beginAtZero: true
      }
    }
  }
});

async function fetchAndUpdate(chart, type) {
  try {
    const res = await fetch('/update');
    const data = await res.json();
    const now = Date.now();

    if (type === 'temperature') {
      chart.data.datasets[0].data.push({ x: now, y: data.temperature });
    } else if (type === 'humidity') {
      chart.data.datasets[0].data.push({ x: now, y: data.humidity });
    }

    // Update UI
    tempSpan.textContent = data.temperature?.toFixed(1) ?? '--';
    humSpan.textContent = data.humidity?.toFixed(1) ?? '--';
    presSpan.textContent = data.pressure ?? '--';
    timeSpan.textContent = data.timestamp ?? '--';

    if (data.temperature != null) {
      gaugeTemp.update({ percent: Math.min(data.temperature / 50, 1) });
    }
    if (data.humidity != null) {
      gaugeHum.update({ percent: Math.min(data.humidity / 100, 1) });
    }
  } catch (err) {
    console.error("Fetch error:", err);
  }
}
```

Rezultate Obţinute

Rezultatele obtinute in acest proiect se pot vedea pe urmatorul link:
https://youtu.be/XK-8px-DiMA?feature=shared

Concluzii

Proiectul Remote Weather Station demonstreaza o solutie eficienta si accesibila pentru monitorizarea conditiilor meteorologice in timp real. Am invatat foarte multe lucruri lucrand la acest proiect, de la cum legi toate perifericele si le conectezi intre ele, pana la ce cod scrii pentru a face ceea ce iti propui.

Bibliografie/Resurse

Resurse hardware

- Datasheet BME280 (Bosch): https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/
- Datasheet DHT22: https://cdn.sparkfun.com/assets/f/7/d/9/c/DHT22.pdf
- Datasheet ESP32: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

Resurse software

- Documentație oficială Arduino: https://www.arduino.cc/
- Bibliotecă Adafruit Sensor: https://github.com/adafruit/Adafruit_Sensor
- Connect ESP32 to wifi: https://www.electronicwings.com/esp32/esp32-wi-fi-basics-getting-started
- Biblioteca Wifi: https://docs.arduino.cc/libraries/wifi/