Stație meteorologică inteligentă

Introducere

Proiectul constă în realizarea unei stații meteorologice inteligente avansate care:

- măsoară și afișează temperatura, umiditatea și calitatea aerului în timp real
- înregistrează datele colectate și afișează data și ora curentă
- utilizează o placă Arduino, senzori de temperatură și umiditate, un senzor de calitate a aerului, un breadboard, un display și funcționalități de conectivitate wireless și înregistrare a datelor
- ideea de la care am pornit a fost să creez o stație meteorologică care să ofere date precise și în timp real despre condițiile de mediu dintr-o locație anume
- proiectul este util atat pentru mine cat si pentru altii, deoarece poate fi utilizat pentru monitorizarea mediului înconjurător și pentru a lua măsuri adecvate în cazul depășirii limitelor acceptabile ale parametrilor măsurați

Descriere generală

Schemă bloc a proiectului:

- Placa Arduino controlează senzorii, modulul Wi-Fi, modulul RTC, precum și afișajul LCD
- Senzorii de temperatură, umiditate și calitatea aerului trimit datele măsurate către placa Arduino
- Modulul Wi-Fi trimite datele colectate către un server Node.js
- Modulul RTC furnizează data și ora curentă pentru afișare pe ecranul LCD
- Ecranul LCD afișează valorile măsurate de senzori, data și ora curentă
- Serverul trimite datele catre o baza de date PostgreSQL
- Vom genera 3 grafice care monitorizeaza datele prinmite de la Arduino, pe un site care primeste datele de la server printr-un request GET

Hardware

```
+---------------------+
|                     |
|      Arduino        |
|                     |
+----------+---------+
           |
           | I2C, Analog, Digital
           |
+----------+---------+
|                     |
```

```
|    Sensori + RTC   |
|                    |
+----------+---------+
           |
           | Wireless (ESP32)
           |
```

Software

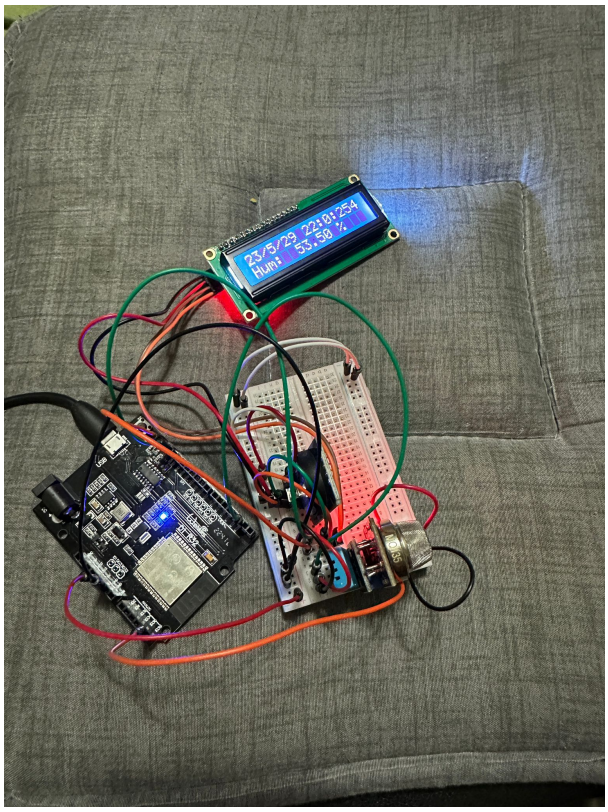
```
+----------+---------+
|                    |
|    Cod Arduino     |
| (C++ cu librarii)  |
|                    |
+----------+---------+
           |
           | HTTP POST
           |
+----------+---------+      +--------------------+
|                    |      |                    |
|     Server Node    |      |     Baza de        |
|  (JavaScript cu    |      |  date PostgreSQL   |
|   Express și Prisma)      |                    |
|                    |      |                    |
+----------+---------+      +--------------------+
           |                         ^
           | HTTP GET                |
           |                         | Query SQL
+----------+---------+               |
|                    |               |
|     Frontend       |               |
|   (React, Vue etc) |---------------+
|                    |
+--------------------+
```

Hardware Design

Listă de piese:

```
1. Placă Arduino Wemos R32 (cu modul wi-fi implementat):
2. Senzor de temperatură și umiditate: DHT11
3. Senzor de calitate a aerului: MQ135
4. Breadboard
5. Display 16x2
7. Fire de conexiune
10. Modul RTC (Real Time Clock)
```

In schema de pe thinkercad nu am inclus modulul RTC, de asemenea senzorul de calitatea aerului a fost reprezentat printr-un senzor de gaz.



Software Design

Arduino:

Software-ul proiectului constă intr-un program încărcat pe placa Arduino. Acesta citește datele de la senzorii conectați la placa, le afișează pe display-ul LCD și le trimite pe un server prin intermediul modulului WiFi.

Librării utilizate în acest proiect:

Wire.h: O bibliotecă care oferă comunicare I2C, utilizată pentru interacțiunea cu display-ul LCD și cu RTC.

LiquidCrystal_I2C.h: Biblioteca care controlează display-ul LCD.

DHT.h: Biblioteca utilizată pentru a comunica cu senzorul DHT11.

MQ135.h: Biblioteca folosită pentru a comunica cu senzorul de calitate a aerului MQ135.

RTClib.h: O bibliotecă pentru a lucra cu Real-Time Clock.

WiFi.h: O bibliotecă care permite conectarea la o rețea WiFi și comunicarea prin aceasta.

HTTPClient.h: Biblioteca pentru efectuarea de cereri HTTP (POST, în cazul nostru).

Structura codului este simplă și se bazează pe două funcții principale ale Arduino IDE:

setup(): Aceasta este funcția unde toate modulele hardware sunt inițializate. Se inițializează modulul LCD, senzorul de temperatură și umiditate (DHT11), senzorul de calitate a aerului (MQ135) și Real Time Clock (RTC).

De asemenea, se inițializează modulul WiFi și se conectează la rețeaua specificată prin SSID și parola. În cazul în care conexiunea la WiFi nu reușește, Arduino începe un proces de reîncercare la fiecare secundă. După ce s-a realizat conexiunea, adresa IP alocată modulului WiFi este afișată pe LCD.

loop(): Aceasta este funcția unde se realizează citirea datelor de la senzorii conectați, afișarea lor pe LCD și trimiterea lor pe server.

La începutul funcției loop, se inițializează clientul HTTP care va fi folosit pentru a trimite cereri HTTP la serverul nostru. Se adaugă apoi un header pentru a specifica că tipul de conținut care va fi trimis va fi text.

Se creează apoi trei șiruri de caractere, unul pentru fiecare dată pe care dorim să o trimitem la server: temperatura, umiditatea și calitatea aerului.

Următoarea buclă for trece prin toate aceste șiruri de caractere și pentru fiecare dintre ele, actualizează display-ul LCD cu ora curentă și cu valoarea datelor, apoi trimite o cerere HTTP POST către server cu datele. Această buclă se repetă la fiecare secundă.

După ce toate datele au fost trimise la server, se închide clientul HTTP. De asemenea, există funcții auxiliare pentru conversia datelor de la senzori în șiruri de caractere, pentru afișarea timpului și pentru ștergerea unui rând de pe display.

Server:

Serverul Node.js, scris în JavaScript și folosind Express.js și Prisma pentru interacțiunea cu baza de date, primește aceste date, le parsează și le stochează în baza de date. Acest server este și punctul de plecare pentru solicitările GET ale frontend-ului, răspunzând cu datele cerute.

Frontend:

Site-ul este scris în React.js și conține două componente principale: App și AirChart.

Componenta App este componenta principală a aplicației. În interiorul acesteia, folosim Hook-urile useState și useEffect pentru a obține și actualiza datele primite de la serverul Node.js.

useState: Este un Hook în React care ne permite să adăugăm stare în componentele noastre funcționale. În acest caz, starea este utilizată pentru a stoca datele primite de la server.

useEffect: Acest Hook acceptă o funcție care conține efecte secundare - operațiuni pe care le putem efectua în componenta noastră funcțională. În acest caz, se folosește pentru a face o cerere HTTP la server și pentru a actualiza starea cu datele primite. Acest efect va rula după fiecare randare și la un interval de 5 secunde datorită apelului setInterval.

Datele obținute sunt apoi filtrate pe baza tipului lor (Air, Temp, Hum) și fiecare set de date este trimis ca prop la componenta AirChart pentru a fi afișat în graficul corespunzător.

Componenta AirChart este responsabilă pentru afișarea datelor într-un grafic de linii, folosind biblioteca Recharts. Aceasta primește datele ca prop și calculează valoarea minimă, maximă și media a datelor. Acestea sunt folosite pentru a configura axa Y a graficului și pentru a trasa o linie de referință care reprezintă media datelor. În final, datele sunt afișate într-un grafic de linii, cu timpul pe axa X și valorile pe axa Y.

În concluzie, această aplicație React.js preia date de la serverul Node.js și le afișează în trei grafice de linii, unul pentru fiecare tip de date (Air, Temp, Hum), actualizându-se în timp real la un interval de 5 secunde.

cod arduino si server:

https://github.com/bubu798/Intelligent_Meteo_Station

cod frontend:

https://github.com/bubu798/Intelligent_Meteo_Station_Frontend

Rezultate Obținute

Rezultatele obtinute sunt 3 grafice care monitorizeaza temperatura, umiditatea, respectiv calitatea aerului dintr-un anumit loc, pe un site.

Acestea sunt actualizare in timp real, aici: https://meteo-stats.vercel.app/, atata timp cat statia meteorologica este in functiune si conectata la WiFi.

Concluzii

Acest proiect ilustrează eficient cum tehnologiile digitale pot fi folosite pentru a monitoriza și a înțelege mediul înconjurător. De la senzori și microcontrolere la servere și vizualizări de date,

proiectul reprezintă un exemplu concret de aplicare a tehnologiei în rezolvarea problemelor reale.

- Înțelegerea senzorilor și a datelor: Proiectul ne ajută să înțelegem cum funcționează senzorii - cum colectează datele și cum aceste date pot fi folosite pentru a interpreta starea mediului înconjurător. Ne oferă o perspectivă asupra modului în care datele pot fi prelucrate și interpretate pentru a ne oferi informații utile.
- Conectivitate și comunicații: Ne arată cum dispozitivele pot fi conectate între ele și cum pot comunica datele prin intermediul rețelei WiFi. Acest lucru este esențial în lumea conectată de azi, unde Internetul lucrurilor (IoT) devine din ce în ce mai prevalent.
- Programare și dezvoltare software: Proiectul necesită cunoștințe de programare atât pentru microcontroler, cât și pentru server și frontend. Acest lucru ne poate ajuta să ne îmbunătățim abilitățile de programare și să învățăm cum să dezvoltăm aplicații software complete.
- Vizualizarea datelor: Prin aplicația web, învățăm despre importanța vizualizării datelor și a modului în care aceasta poate facilita înțelegerea și interpretarea datelor. Acest lucru este esențial în lumea datelor de astăzi, unde abilitatea de a prezenta și interpreta datele într-un mod ușor de înțeles este o abilitate valoroasă.

În concluzie, acest proiect oferă o oportunitate excelentă de învățare și dezvoltare a abilităților în mai multe domenii cheie ale tehnologiei digitale. De asemenea, reprezintă un exemplu concret de modul în care tehnologia poate fi folosită pentru a ne ajuta să înțelegem și să interacționăm cu mediul nostru într-un mod mai informat și eficient.

Download

cod arduino si server:

https://github.com/bubu798/Intelligent_Meteo_Station

cod frontend:

https://github.com/bubu798/Intelligent_Meteo_Station_Frontend

Jurnal

07.05 - Creat pagina de proiect și adăugat documentația inițială

21.05 - Hardware design

28.05 - Sofware design, Rezultate obtinute, Concluzie

29.05 - Retusare

Export to PDF