Proiect-PR-IOT: Sistem de monitorizare pentru o seră inteligentă

Matei Cristian Costescu 344C1

January 13, 2025

Link Github

Abstract

Proiectul presupune dezvoltarea unui sistem de monitorizare și control bazat pe microcontroller-ul ESP32, care colectează date despre condițiile ambientale ale unei sere și trimite informații prin actuatori pentru a menține condițiile optime de creștere. Proiectul folosește tehnologii precum ESP32, Flask, HTTP REST API și Telegram pentru a crea un sistem eficient și ușor de utilizat.

1 Introducere

1.1 Descriere generală

Proiectul presupune dezvoltarea unui sistem de monitorizare și control bazat pe microcontroller-ul ESP32, care colectează date despre condițiile ambientale folosind senzori și trimite informații prin actuatori. Proiectul este reprezentat de creearea unui sistem pentru o seră inteligentă, care să măsoare temperatura, umiditatea, factorul de lumină și umiditatea solului.

1.2 Obiectivele proiectului

- Monitorizarea temperaturii, umidității aerului, luminii ambientale și umidității solului.
- Controlul activităților printr-o aplicație de control bazată pe un server Flask.
- Afișarea alertelor folosind un buzzer, un LED și un display LCD pentru mesaje informative.

1.3 Tehnologii folosite

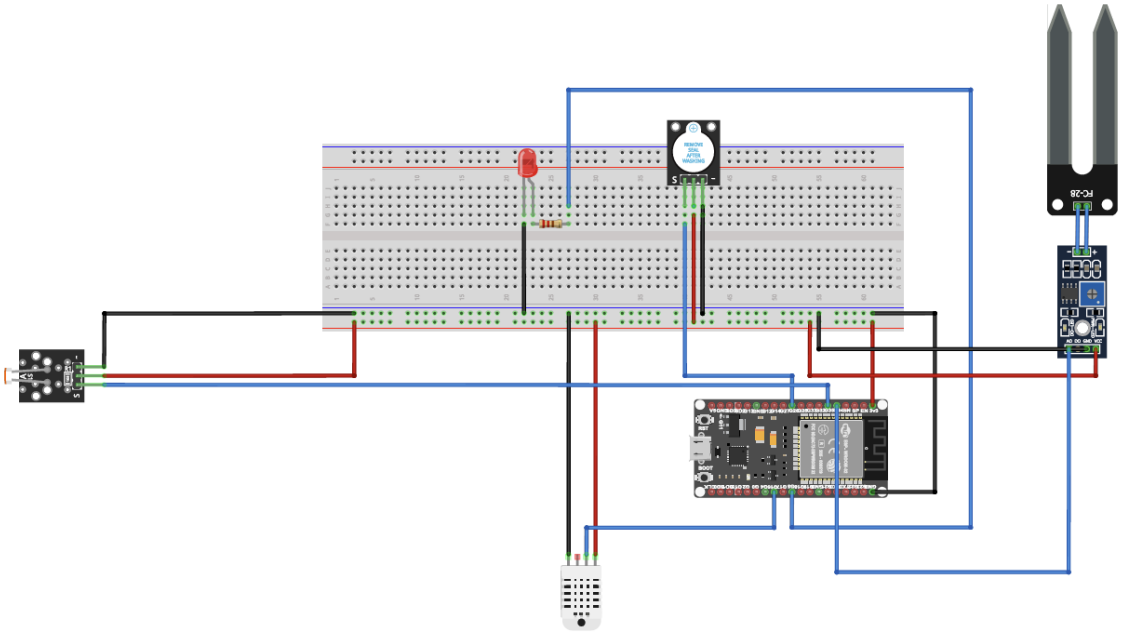
- ESP32 pentru conectivitate WiFi și interfațarea cu senzori.
- Flask pentru backend și gestionarea comenzilor.
- HTTP Client pentru comunicarea între ESP32 și server.
- Arduino IDE pentru dezvoltarea codului pe placuța ESP32.

2 Arhitectura Sistemului

2.1 Diagramă de topologie a rețelei

Diagramă logică ce arată conexiunile între componente:

- ESP32 → Server Flask (rețea WiFi locală)



2.2 Senzori conectați la ESP32

- DHT22: Temperatură și umiditate aer.
- Senzor analogic pentru umiditatea solului.
- Fotorezistor pentru măsurarea nivelului de lumină.

2.3 Actuatori conectați

- Buzzer pentru alerte.
- LED pentru a simula ”aducerea de lumină”.
- LCD I2C (opțional) pentru afișarea informațiilor.

3 Protocoale de Comunicație

- **WiFi (802.11)**: Utilizat pentru conectarea ESP32 la serverul Flask.
- **HTTP REST API**:
  - Metoda POST: Transmiterea datelor de la senzori către server. Endpoint: /api/data.
  - Metoda GET: Preluarea comenzilor de la server. Endpoint: /api/command.

4 Descrierea Componentelor

4.1 Senzori

- **DHT22**: Măsoară temperatura și umiditatea aerului, conectat la pinul digital al ESP32.
- **Senzor de Umiditate Sol**: Măsoară umiditatea solului în mod analogic, conectat la pinul analogic (A0) al ESP32.
- **Fotorezistor**: Măsoară intensitatea luminii, conectat la un pin analogic al ESP32.

4.2 Actuatori

- **Buzzer**: Avertizează utilizatorul în funcție de comanda primită.
- **LED**: Avertizează utilizatorul în funcție de comanda primită.
- **LCD I2C**: Afișează mesaje informative (opțional).

4.3 Server Flask

- Rol: Gestionarea datelor de la senzori și trimiterea comenzilor către ESP32.
- Endpoints:
  - /api/data (POST): Primește date de la ESP32.
  - /api/command (GET): Trimite comenzi către ESP32.

5 Funcționarea Sistemului

5.1 Inițializare ESP32

- Se conectează la rețeaua WiFi.
- Inițializează senzorii și actuatorii.

5.2 Transmiterea datelor

- ESP32 colectează datele senzorilor.
- Trimite datele către server printr-o cerere HTTP POST.

5.3 Primirea comenzilor

- Serverul trimite comenzi (ex: humidity, temperature) în funcție de nevoile definite..
- ESP32 reacționează activând buzzerul sau afișând mesaje pe LCD.

5.4 Comunicarea bidirecțională

Sistemul oferă feedback în timp real între server și microcontroller.

6 Implementare

6.1 Pașii de Configurare a Hardware-ului

- Asamblarea componentelor: Am conectat ESP32 la senzorii de umiditate a solului, modulul cu fotorezistor, senzorul de temperatură și umiditate, buzzerul activ conform schemei electronice. Apoi am verificat conexiunile pentru ma asigura ca se face bine citirea datelor de catre senzori si ca nu se produce niciun scurt-circuit sau ceva asemanator.
- Configurarea ESP32: Am deschis Arduino Uno si am selectat placa de ESP32 si am configurat acolo pinii GPIO corespunzatori pentru fiecare senzor, apoi am testat functionarea fiecărui senzor individual utilizând coduri de test predefinite.

6.2 Pașii de Configurare a Software-ului

- Dezvoltarea firmware-ului: Am scris codul ESP32 utilizând Arduino IDE, implementand logica de citire a senzorilor și transmiterea datelor către serverul Flask prin Wi-Fi. Am configurați un sistem de alertare pe baza pragurilor definite pentru parametrii senzorilor, sisteml de alerte de la nivelul esp32-ului activeaza buzzerul activ pentru a atentiona utilizatorul ca ceva nu merge cum ar trebui.
- Setarea serverului Flask: Am implementat serverul Flask și dependințele necesare, creand rutele pentru primirea și procesarea datelor de la ESP32. Am configurat notificări prin telegram cu ajutorul unui bot pe care mi l-am creat. Notificarile se trimit in caz ca una dintre valorile citite de oricare senzor nu se afla in parametrii (trece de o valoare de prag), caz care se verifica la fiecare minut, tot atunci cand se si primesc datele de la placuta ESP32.
- Testare și Debugging: Am simulat diferite scenarii pentru a verifica funcționalitatea sistemului de alertare, am monitorizat log-urile oferite de serverul de Flask pentru a identifica și corecta erorile.

6.3 Configurarea Sistemului de Alertare și Notificare

- Setarea Pragurilor: Am definit praguri pentru parametrii senzorilor (acest lucru nu este foarte ine pus la punct intrucat nu am atasat inca senzorii la un ghiveci cu plante pentru a vedea exact valorile de care are nevoie planta si a vedea exact care ar trebui sa fie pragurile finale, totusi am trecut niste praguri la care m-am gandit eu avand in vedere valorile citite de senzori in camera mea, iar pentru senzorul de temperatura am lasat intentionat un prag extrem de mic pentru a putea verifica trimiterea alertelor e telegram).
- Implementarea Notificărilor: Am utilizat un bot de telegram pentru trimiterea alertelor, lucru pe care l-am facut cu ajutorul informatiilor din laboratorul 8.
- Testare Finală: Am testat sistemul cu valori simulate pentru a verifica că notificările funcționează corect.

7 Vizualizare și Procesare de Date

7.1 Metoda de Procesare

- Preluarea Datelor: Datele sunt transmise de ESP32 către serverul Flask prin HTTP POST. Serverul validează și stochează datele primite. Datele au fost prelucrate o parte inca de la nivelul ESP32-ului pentru a ma asigura ca au o forma corecta si adecvata cu care sa lucrez mai departe, apoi s-a facut o extra procesare a lor la nivelul serverului de Flask pentru a putea sa creez acele praguri pentru alerte.
- Stocarea Datelor: Am folosit Firebase pentru a stoca valorile senzorilor și timestamp-urile asociate.

7.2 Metoda de Afișare

- Interfața Web: Am utilizat serverul Flask pentru a imi genera o interfata grafica pentru valori, astfel ca am adaugat pe pagina serverului 4 grafice care stocheaza fiecare ultimele 20 de valori citite si trimise de catre senzori pentru a le afisa intr-un mod frumos.
- Raport Poer BI: Pe langa asta am creat un raport in Power BI pentru a prezenta graficele pentru fiecare senzor si am adaugat dependinte intre ele pentru a putea selecta spre exemplu valoarea citita candva pentru o temperatura si a imi afisa totodata valorile senzorilor de umiditate, lumina si umiditatea solului care au fost primite in acelasi timp cu acea temperatura. Astfel am adaugat si un filtru dupa temperatura pentru a fi mai usor sa gasim anumite valori (pentru a se face o verificare mai simpla a senzorilor) si pe langa asta am adaugat un panou de cereri, care prelucreaza o cerere data de la tastatura si iti afiseaza date in concordanta cu cererea primita. Astfel poti descoperi usor valoarea medie a temperaturii spre exemplu doar tastand "Average temperature" sau sa sortezi datele intr-o anumita ordine (exista si niste comenzi sugerate/predefinite pentru vizualizarea datelor).

7.3 Compatibilitate cu telefonul

- Partea de notificari fiind pe Telegram e pot observa usor de pe telefonul mobil, primind notificari de fiecare data cand se depisteaza o valoare care trece de anumite praguri predefinite.

8 Securitate

Pentru partea de securitate am stocat datele in Firebase, iar aceasta baza de date criptează datele în tranzit folosind HTTPS și le stochează criptat pe serverele Google, oferind protecție împotriva interceptării datelor. Firebase oferă integrări cu Google Cloud pentru monitorizarea activităților și auditarea accesului, ceea ce poate ajuta la detectarea accesului neautorizat. Am adaugat reguli de securitate la nivelul bazei de date pentru a nu putea toata lumea sa citeasca sau sa modifice datele.

Pe langa partea de Firebase am creat o mini securitate a datelor intre serverul Flask si placuta ESP32 prin realizare unei autentificari, am creat un token privat pe care il cunosc atat serverul de Flask cat si placuta, iar toate mesajele trimise intre acestea doua se realizarea utilizand acest token. Daca serverul nu recunoaste tokenul trimis de placuta ESP32 va da abort la orice interogare sau orice date vor fi primite si nu va mai functiona comunicatia.