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

Matei Cristian Costescu 344C1

January 20, 2025

Link Github

Contents

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. Sistemul are ca scop crearea unui mediu ideal pentru creșterea plantelor într-o seră, permițând monitorizarea și optimizarea parametrilor critici, precum temperatura, umiditatea, lumina și umiditatea solului. Acest proiect este o soluție inovatoare pentru agricultura modernă, oferind un sistem automatizat, eficient și economic.

Această implementare poate fi utilizată nu doar pentru serele comerciale mari, ci și pentru serele de dimensiuni mai mici, aducând beneficii semnificative în economisirea resurselor și îmbunătățirea randamentului producției agricole.

1.2 Obiectivele proiectului

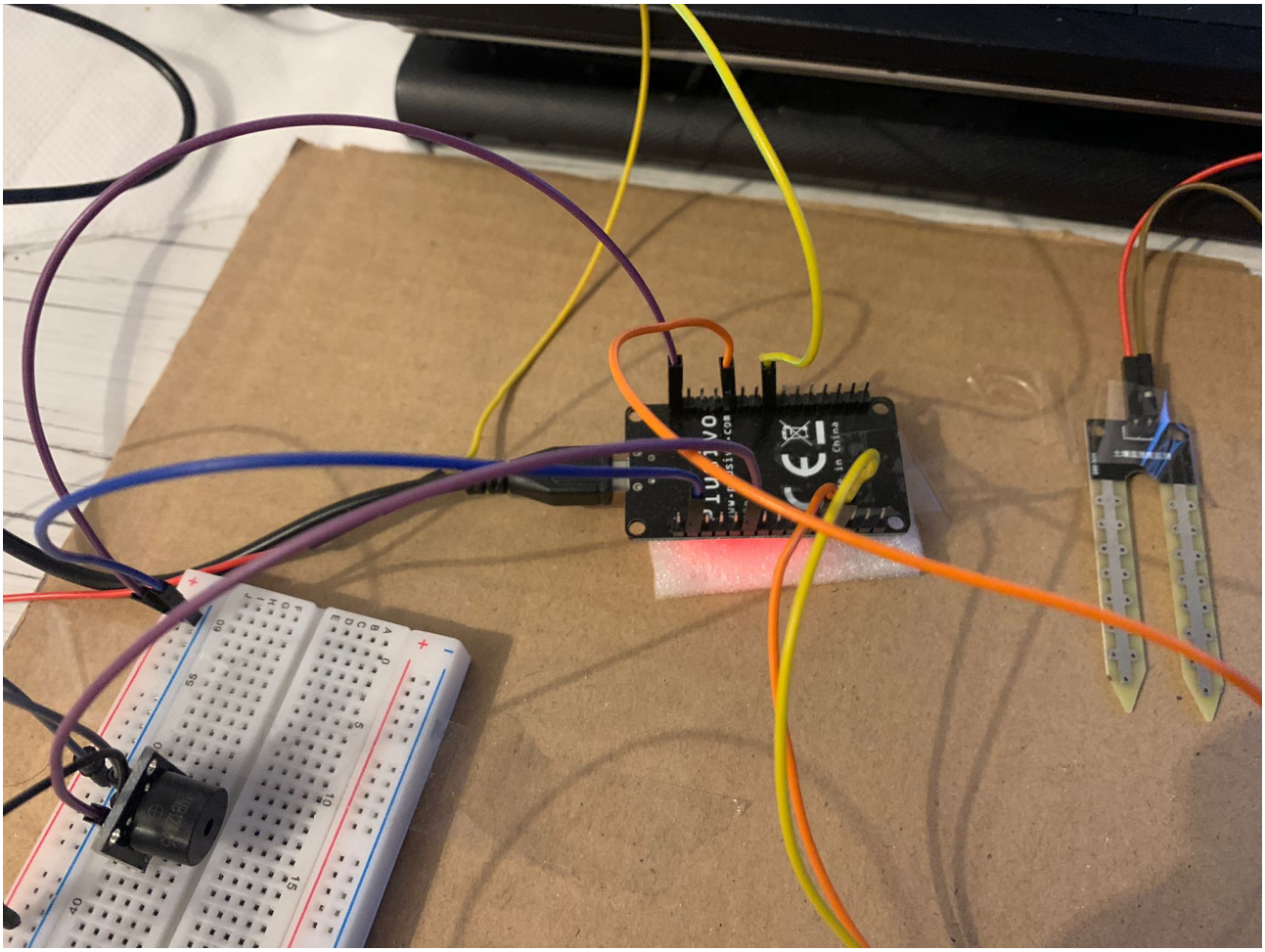
Obiectivele principale ale acestui proiect sunt:

- **Monitorizarea parametrilor ambientali**: Sistemul trebuie să fie capabil să măsoare temperatura, umiditatea aerului, lumina și umiditatea solului, oferind date precise și în timp real.
- **Controlul activităților**: Prin utilizarea unui server Flask, utilizatorii pot controla diverse funcții ale sistemului, inclusiv activarea actuatorilor sau ajustarea pragurilor critice.
- **Alertare eficientă**: Alertele generate de senzori trebuie să fie semnalizate utilizatorilor prin intermediul unui buzzer, LED-uri și notificări pe un display LCD. În plus, notificările prin Telegram permit monitorizarea de la distanță.

Obiectivele sunt orientate către crearea unui sistem scalabil și ușor de utilizat, care să se integreze în diverse medii agricole.

1.3 Tehnologii folosite

- **ESP32**: Acest microcontroller este nucleul sistemului, oferind conectivitate WiFi și capabilități avansate de procesare pentru interfațarea cu senzori și actuatori.



- **Flask**: Serverul Flask este responsabil pentru gestionarea comenzilor și procesarea datelor primite de la senzori. Este utilizat și pentru a oferi o interfață web accesibilă utilizatorilor.
- **HTTP Client**: Permite comunicarea între ESP32 și server, utilizând protocoale REST API pentru transmiterea și primirea datelor.
- **Arduino IDE**: Platforma utilizată pentru dezvoltarea codului care rulează pe ESP32, oferind flexibilitate în implementarea logicii de control.

Aceste tehnologii au fost alese datorită compatibilității lor și a costurilor reduse asociate.

2 Arhitectura Sistemului

2.1 Diagramă de topologie a rețelei

Sistemul utilizează o arhitectură simplă, dar eficientă, pentru a conecta toate componentele necesare. Aceasta include ESP32 ca nod central care comunică cu un server Flask printr-o rețea WiFi locală. Sistemul este conceput pentru a permite extinderea ușoară cu senzori suplimentari sau integrarea unor noi funcționalități.

- **ESP32 → Server Flask**: Transmiterea datelor de la senzori către server și recepționarea comenzilor pentru actuatori.
- **Server Flask → Utilizatori**: Notificări și comenzi gestionate printr-o interfață web și Telegram.

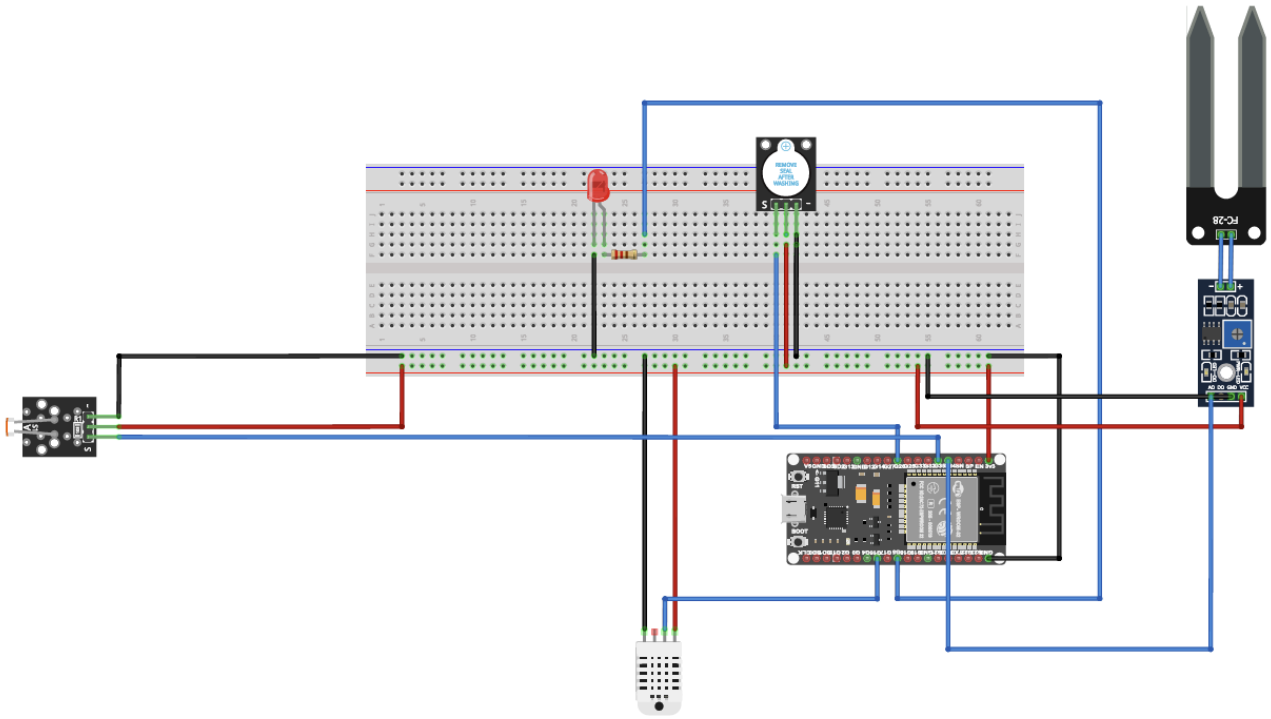
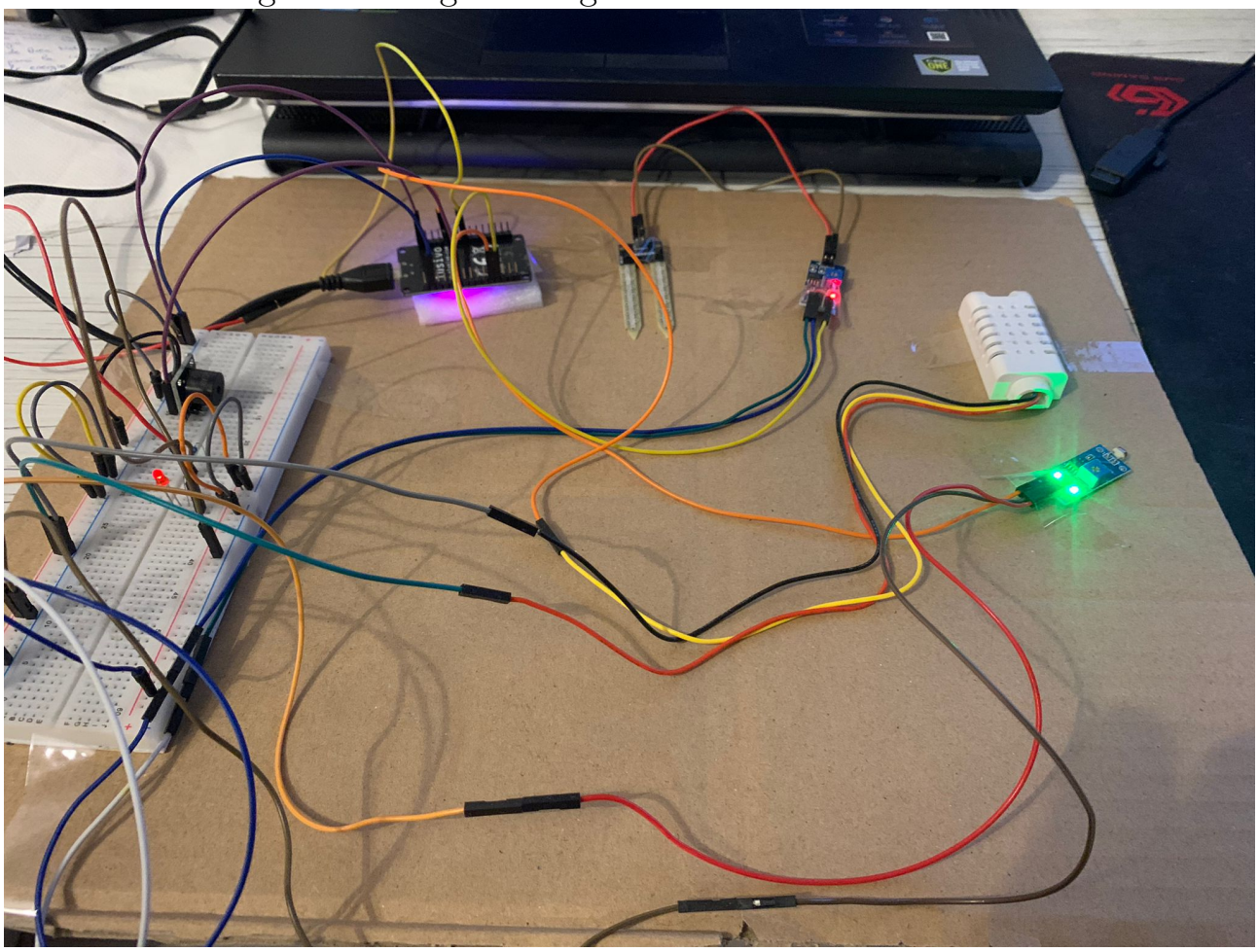


Figure 1: Diagramă logică a arhitecturii sistemului



Această structură simplă, dar robustă, facilitează gestionarea eficientă a resurselor și permite o integrare ușoară a altor module viitoare.

2.2 Senzori conectați la ESP32

- **DHT22**: Acest senzor măsoară temperatura și umiditatea aerului, oferind date precise pentru menținerea unui mediu optim de creștere a plantelor.
- **Senzor de Umiditate Sol**: Un senzor analogic care detectează nivelul de umiditate din sol, permițând utilizatorului să ajusteze irigarea în funcție de nevoile specifice ale plantelor.
- **Fotorezistor**: Utilizat pentru a măsura intensitatea luminii ambientale, acest senzor ajută la determinarea dacă este nevoie de lumină suplimentară în seră.

2.3 Actuatori conectați

- **Buzzer**: Un dispozitiv audio care emite alerte pentru a informa utilizatorii despre valori anormale ale parametrilor monitorizați.
- **LED**: Un indicator vizual pentru a semnaliza condiții specifice sau alerte critice.
- **LCD I2C**: Afișează mesaje informative pentru utilizator, cum ar fi valorile actuale ale senzorilor sau starea sistemului.

Actuatorii contribuie la creșterea interactivității și a eficienței generale a sistemului.

3 Protocoale de Comunicație

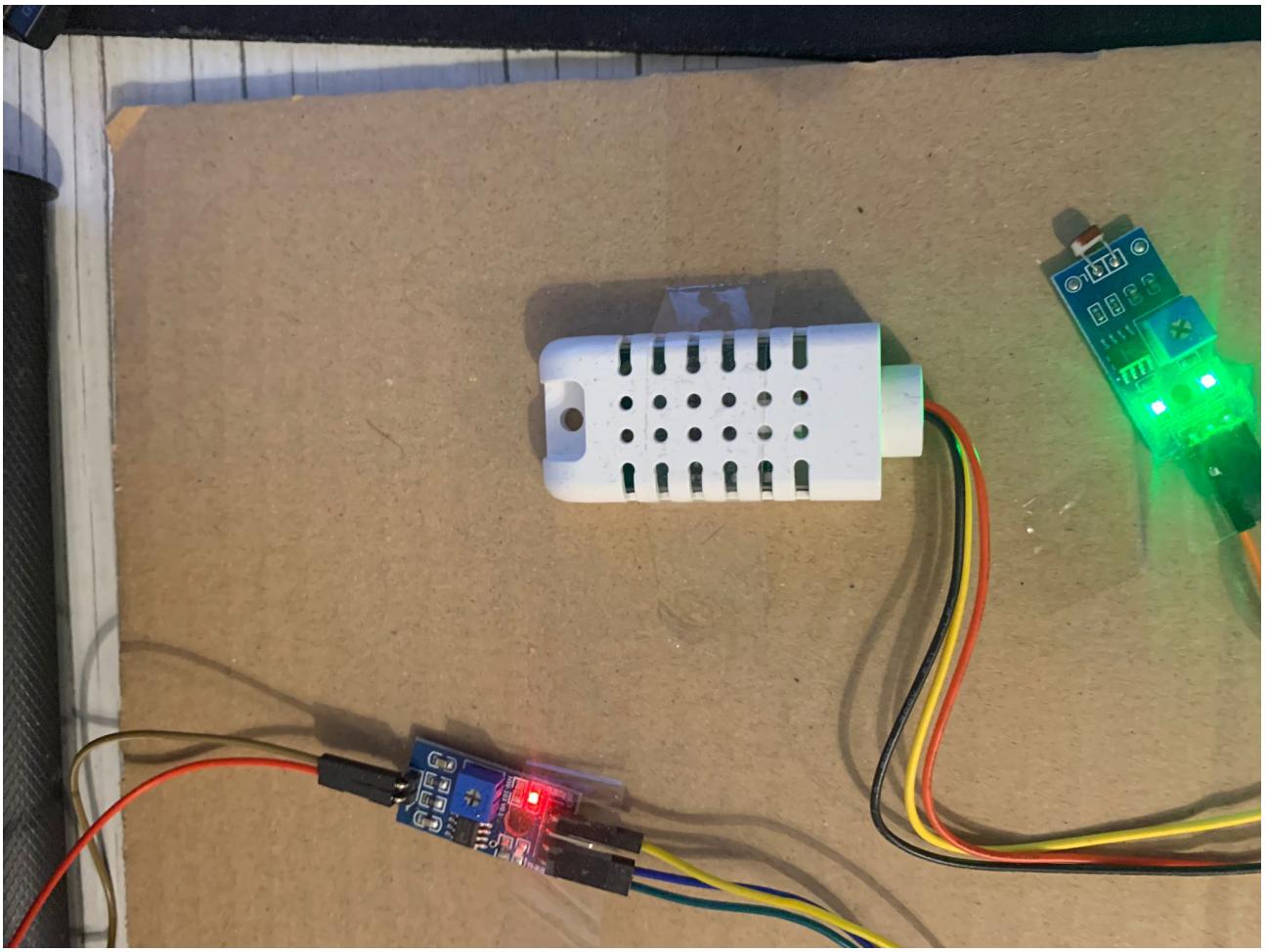
- **WiFi (802.11)**: Este utilizat pentru conectarea ESP32 la serverul Flask printr-o rețea locală, oferind o soluție rapidă și fiabilă pentru transmiterea datelor.
- **HTTP REST API**: Protocolul principal pentru comunicarea dintre ESP32 și server. Acesta include:
  - **Metoda POST**: Transmiterea datelor de la senzori către server. Endpoint: `/api/data`.
  - **Metoda GET**: Preluarea comenzilor de la server către ESP32. Endpoint: `/api/command`.

Utilizarea acestor protocoale asigură o comunicare ușor de implementat și extins în cazul unor cerințe suplimentare.

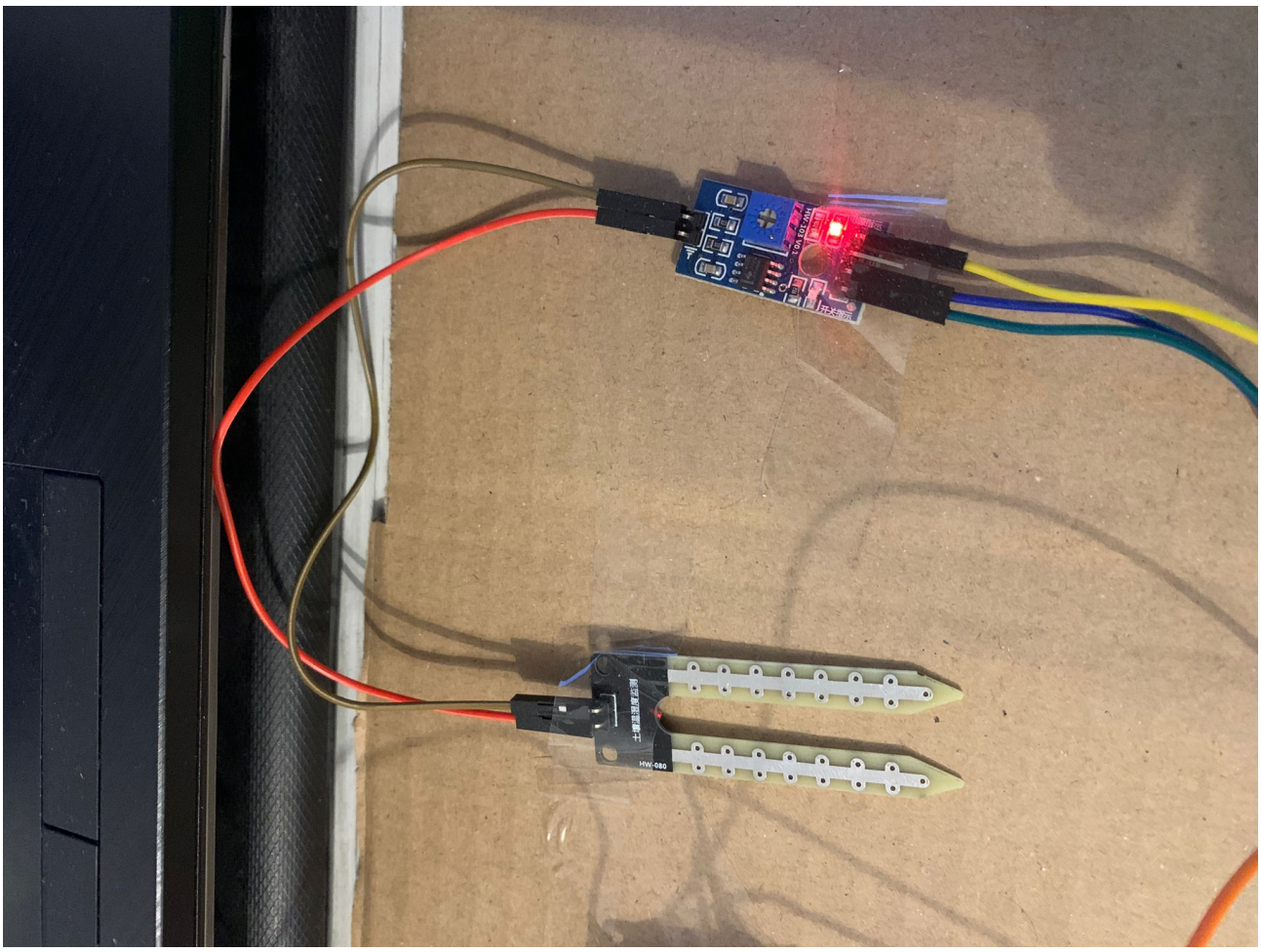
4 Descrierea Componentelor

4.1 Senzori

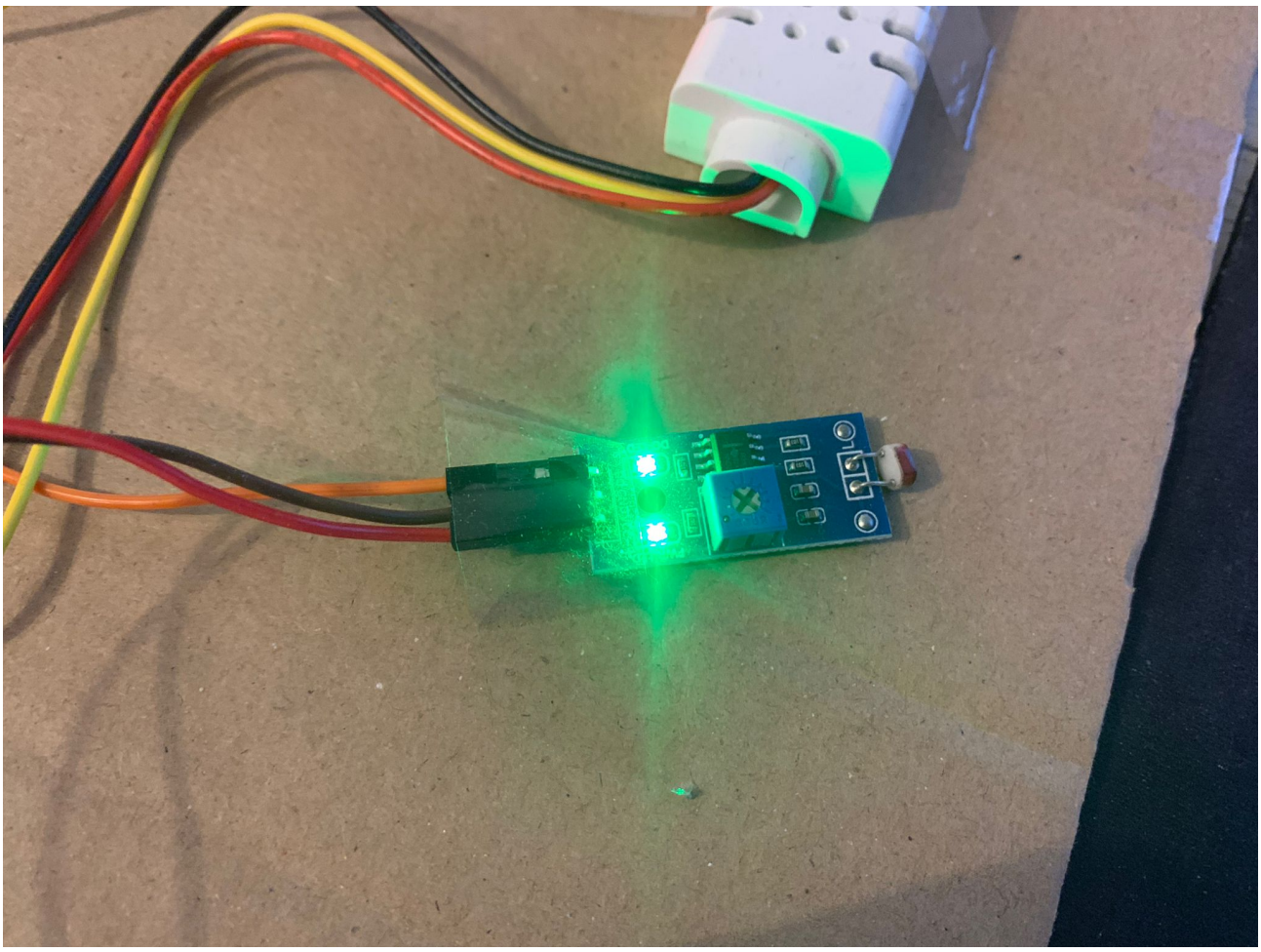
- **DHT22**: Este un senzor digital capabil să măsoare atât temperatura, cât și umiditatea aerului. Acesta are un timp de răspuns rapid și un consum redus de energie, fiind ideal pentru aplicațiile de monitorizare continuă. Datele sunt trimise în format digital către ESP32, eliminând necesitatea conversiei analog-digitale.



- **Senzor de Umiditate Sol**: Acest senzor oferă un semnal analogic care indică nivelul de umiditate din sol. Poate fi utilizat pentru a determina dacă plantele necesită irigare, contribuind astfel la economisirea apei.

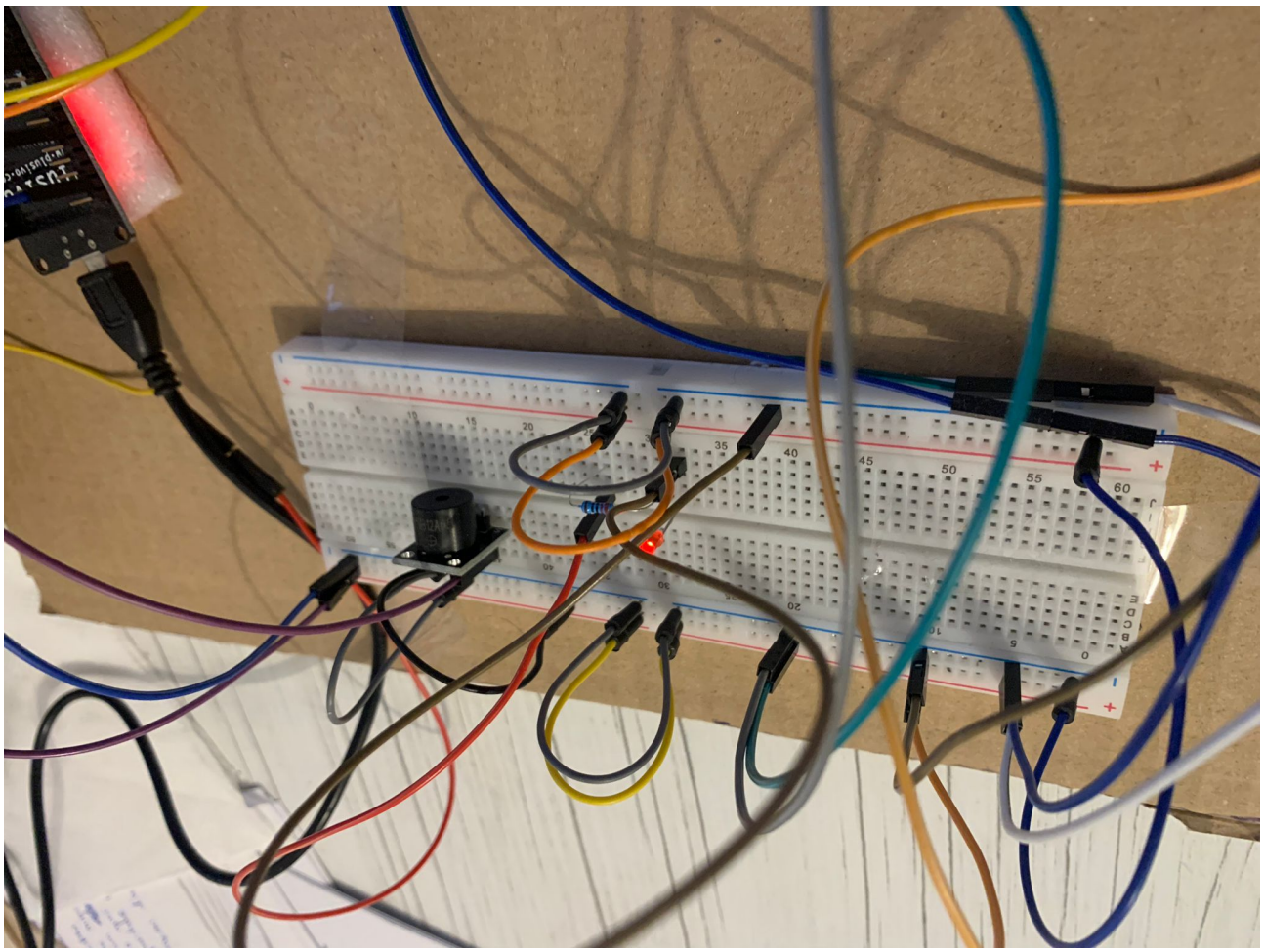


- **Fotorezistor**: Un senzor sensibil la lumină care poate detecta variații ale intensității luminoase. Este folosit pentru a asigura că plantele primesc suficientă lumină pentru fotosinteză.



4.2 Actuatori

- **Buzzer**: Utilizează semnale sonore pentru a alerta utilizatorul în cazul unor condiții critice, cum ar fi niveluri scăzute de umiditate sau temperaturi extreme.
- **LED**: Poate indica starea generală a sistemului sau poate oferi notificări vizuale pentru anumite alerte.



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.

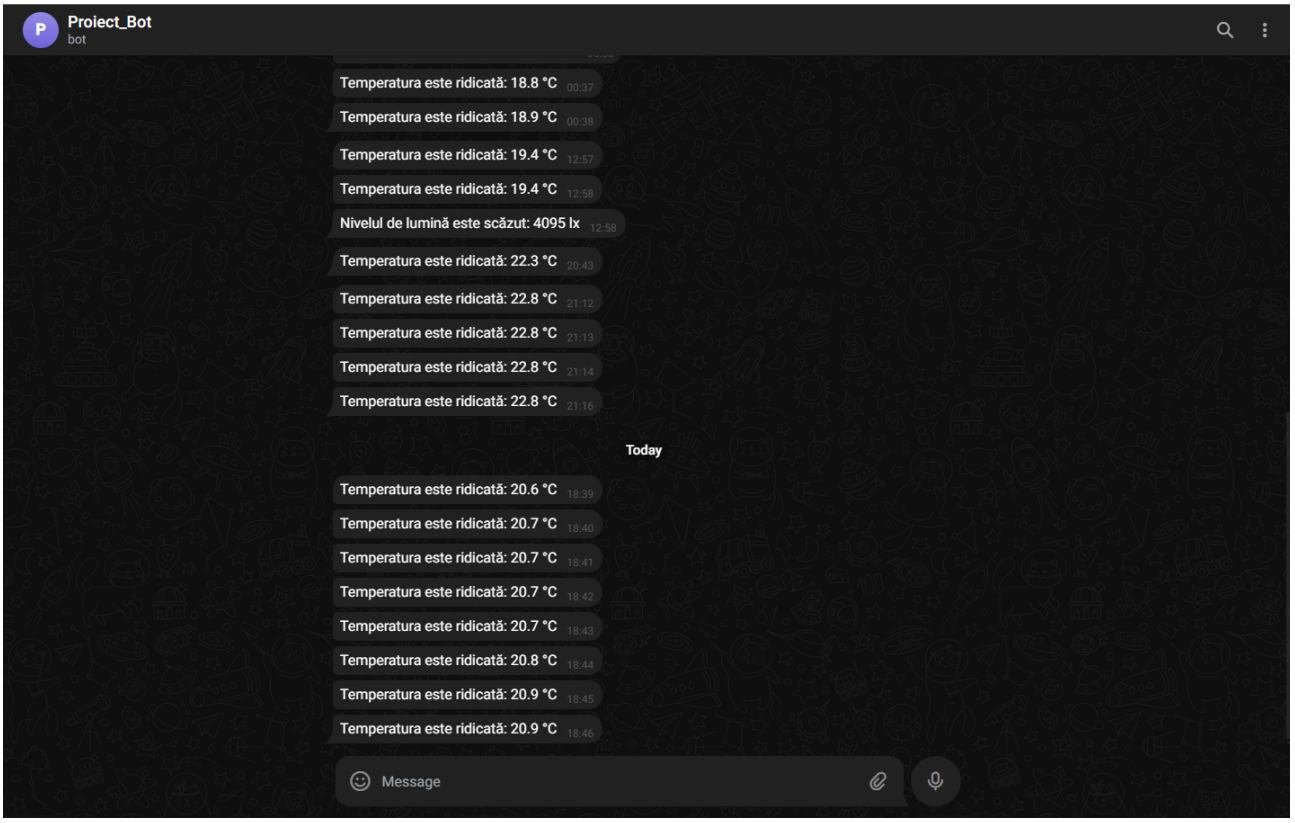
- Testare și Debugging: Am simulat diferite scenarii pentru a verifica funcționalitatea sistemului de alertare, am monitorizat log-urile oferite de serverul de Flask pentru a identifica și corecta erorile.

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 Power 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).

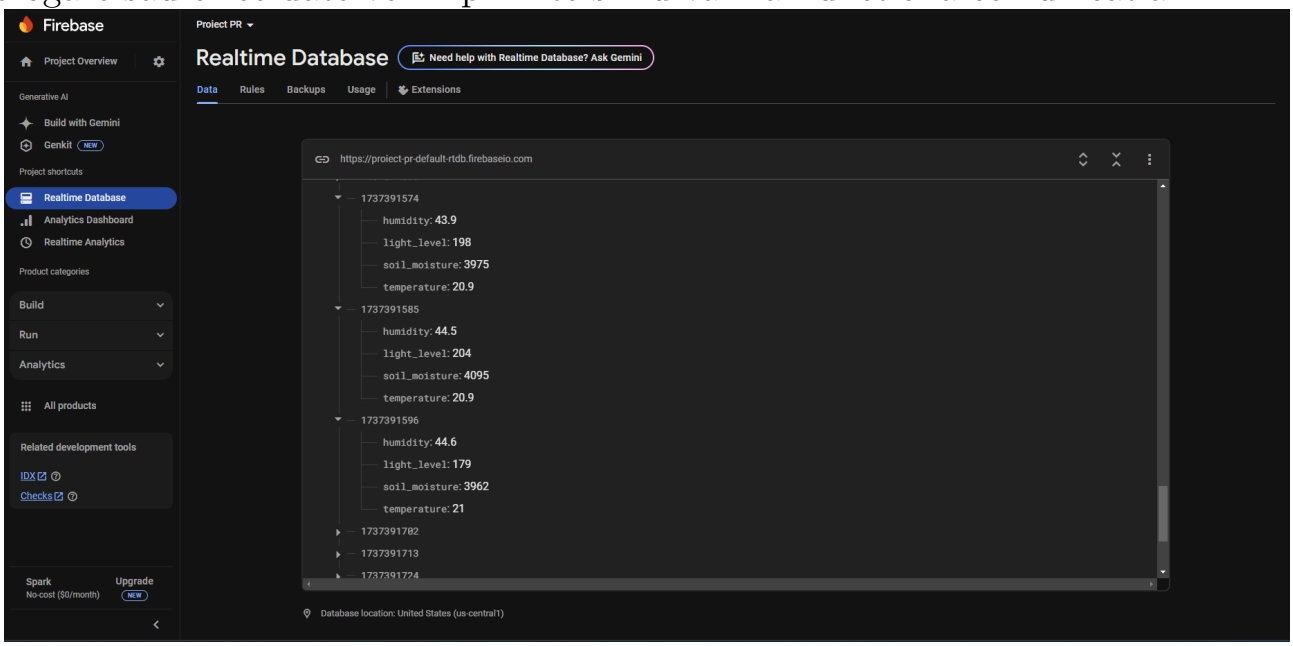
7.3 Compatibilitate cu telefonul

- 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.



9 Provocări și Soluții

Pe parcursul dezvoltării acestui proiect, au fost întâmpinate mai multe provocări, atât pe partea hardware, cât și pe partea software. În continuare, sunt detaliate principalele probleme întâmpinate și soluțiile adoptate pentru a le rezolva.

9.1 Probleme Hardware

În principiu provocările pe partea de hardware au fost legate de cum a trebuit să conectez fiecare senzor la plăcuța ESP32 și cum am reglat fiecare senzor în parte

Soluția aplicată: Am luat datasheet-urile pentru fiecare senzor în parte și am citit ce trebuie să primească la input și la ce fel de pin trebuie conectat (digital sau analogic). Iar pentru partea de reglare a potensiometrelor pot spune doar că a fost nevoie de încercare și testare și tot așa până am găsit niște valori care să se potrivească pentru proiectul meu.

9.2 Probleme Software

- **Întârzieri în transmiterea datelor către server:** Inițial din cauza modului în care am făcut codul pentru plăcuța ESP32, majoritatea datelor erau transmise ori prea târziu ori niciodată și le pierdeam.

  Soluție aplicată: Am modificat complet codul cu ajutorul unor indieni de pe Youtube și unor tutoriale de la alte proiecte pe care le-am găsit pe internet.

- **Crearea unor rapoarte pentru vizualizarea datelor:** Nu știam inițial în ce ar trebui să fac graficele și cum să fac să se vadă mai bine decât pe serverul de Flask, am încercat inițial cu Grafana, dar după ce am stat vreo 2 ore și am văzut că trebuia să folosesc o altă bază de date, nu mergea cu Firebase, am renunțat la idee.

  Soluție aplicată: Mi-am amintit de când am lucrat în vară puțin cu Power BI și cât de ușor să făcea totul dacă aveai salvate datele și într-un Excel/CSV așa că am creat unul și am realizat rapoartele in Power BI.

10 Limitări și Îmbunătățiri posibile

10.1 Limitări identificate

În timpul implementării acestui proiect, au fost observate anumite limitări, atât din punct de vedere hardware, cât și software:

- **Sensibilitatea senzorilor:** Senzorii utilizați, cum ar fi DHT22 sau senzorul de umiditate a solului, pot avea variații în precizie, în special în condiții extreme de temperatură sau umiditate. De exemplu, DHT22 are o toleranță de ± 2
- **Raza de acțiune Wi-Fi:** Conectivitatea ESP32 este limitată la raza de acțiune a rețelei Wi-Fi disponibile, ceea ce poate fi o problemă pentru sere amplasate în locații izolate.
- **Capacitatea de procesare:** Deși ESP32 oferă o putere de procesare impresionantă pentru un microcontroller, procesarea unor cantități mari de date în timp real poate să ducă la întârzieri sau erori.
- **Dependența de server:** Sistemul se bazează pe un server Flask activ pentru gestionarea datelor și comenzilor. În cazul unei întreruperi a serverului, funcționalitatea întregului sistem este compromisă.

10.2 Propuneri de îmbunătățiri

Pentru a depăși limitările identificate și a optimiza performanța sistemului, sunt sugerate următoarele îmbunătățiri:

- **Senzori de calitate mai înaltă:** Utilizarea unor senzori industriali, cum ar fi SHT35 pentru temperatură și umiditate, ar putea crește acuratețea datelor.
- **Extinderea conectivității:** Integrarea unui modul LoRa pentru transmisii pe distanțe mai mari ar permite monitorizarea serei din locații mai îndepărtate.
- **Sisteme redundante:** Implementarea unui sistem de backup, precum un Raspberry Pi local, care să preia temporar funcțiile serverului Flask în caz de avarie.
- **Optimizarea software-ului:** Reducerea complexității codului prin utilizarea unor algoritmi mai eficienți și implementarea unui protocol MQTT pentru comunicare ar putea reduce latența și cerințele de procesare.

11 Concluzii și Perspective viitoare

Proiectul de monitorizare și control al unei sere inteligente reprezintă un exemplu practic de aplicare a tehnologiilor IoT pentru îmbunătățirea eficienței și productivității în agricultură. Sistemul oferă funcționalități variate, de la monitorizarea parametrilor critici până la gestionarea activităților în timp real. Cu toate acestea, există un spațiu considerabil pentru extindere și îmbunătățiri:

- Integrarea unor modele de machine learning pentru a prezice condițiile optime în funcție de tipul de cultură.
- Crearea unei interfețe mobile dedicate pentru accesul rapid la informații și control.
- Extinderea sistemului pentru a gestiona sere multiple dintr-o singură locație centralizată.
- Adăugarea unor funcționalități de automatizare completă, bazate pe praguri predefinite și logici condiționale mai complexe.

Proiectul demonstrează potențialul tehnologiei IoT în agricultură, oferind un sistem modular, extensibil și eficient, care poate fi adaptat nevoilor utilizatorului.