Proiect PR IoT

Sistem de casă inteligent pentru controlul intensității luminii dintr-o încăpere

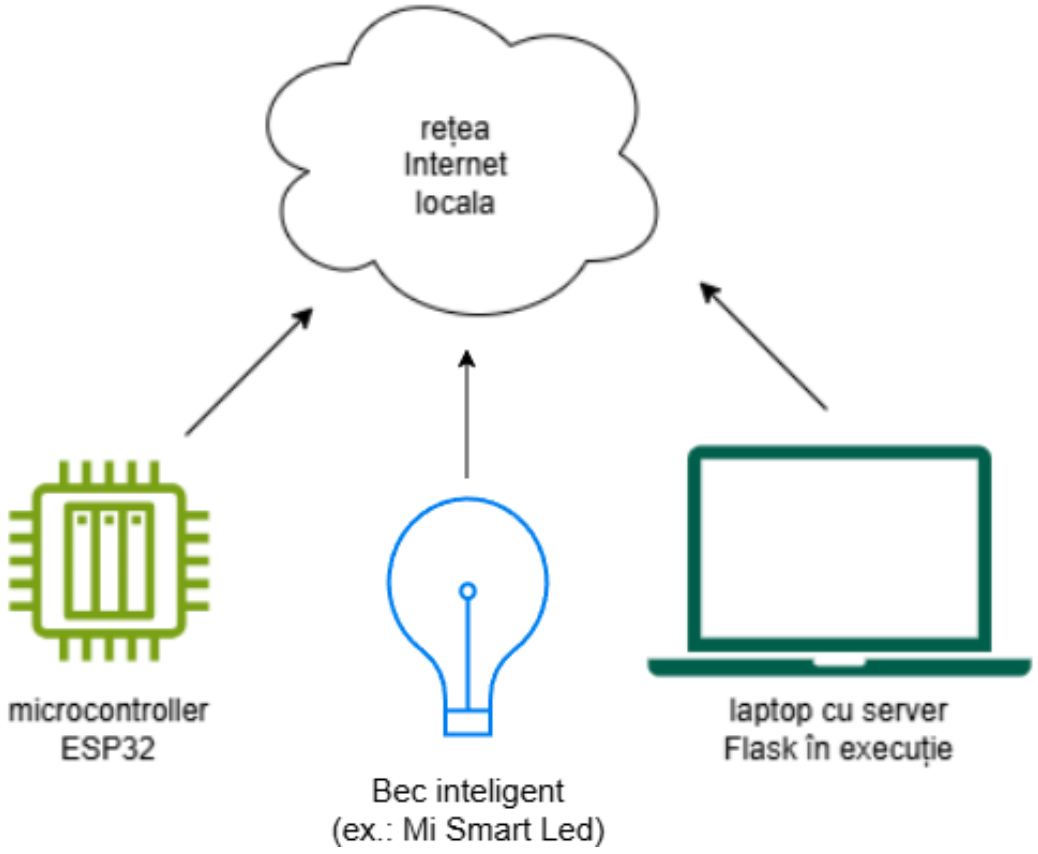
Realizator: Arpășanu Emilia-Oana 341C1

Scopul proiectului

- rețea capabilă să analizeze intensitatea luminii din mediul ambiant
- prelucrare de date pentru vizualizare
- statistici relevante trimise utilizatorului
- control intuitiv al actuatorului
- comunicație securizată între senzor și server
- protocol de transmisie Wi-Fi, ușor de gestionat

Arhitectura rețelei

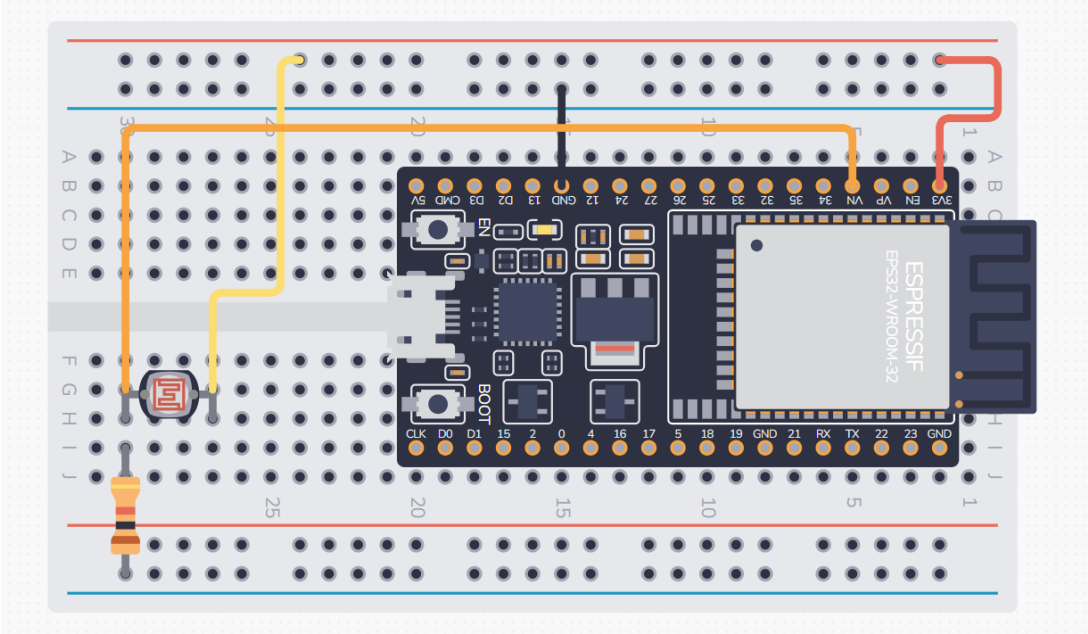
- microcontroller-ul preia datele de la senzor și le transmite server-ului
- server-ul trimite comenzi actuatorului
- toate componentele sunt conectate la aceeași rețea Wi-Fi pentru a asigura comunicarea corectă între ele



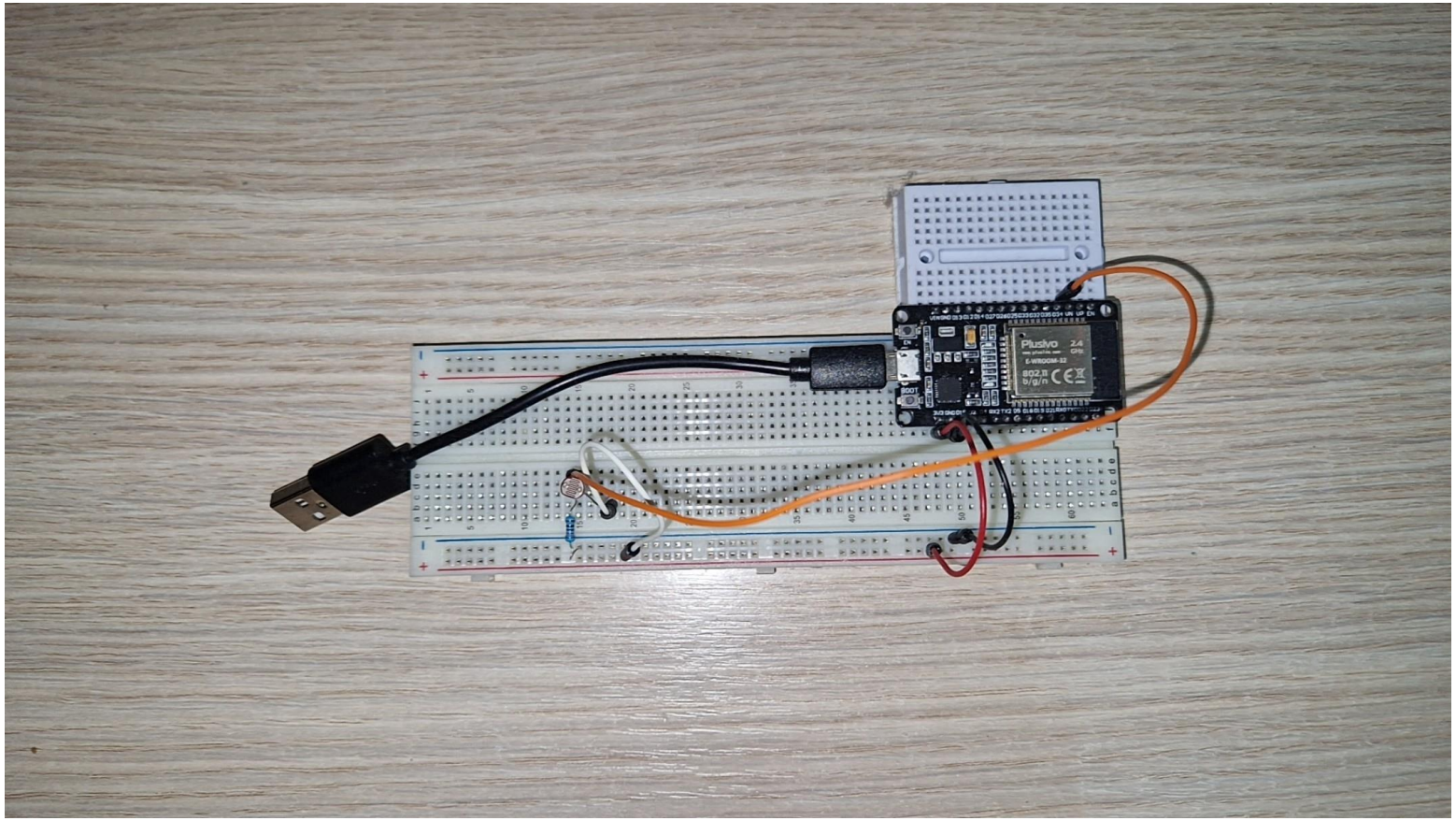
Implementare hardware

Pentru implementarea circuitului sunt necesare următoarele:

- microcontroller de tip ESP32
- rezistență 10kΩ
- fotorezistență
- breadboard (830 tie-points)
- 4 fire de legătură (male-to-male)
- cablu USB pentru alimentare
- + bec inteligent (aici Mi Smart Led) pus la o lampă



Implementare hardware



- Circuitul fizic rezultat

Programarea microcontroller-ului

Pentru a putea citi datele de la senzor și a le trimite către server se execută un program scris în MicroPython în cadrul căruia:

- se realizează conexiunea microcontroller-ului la rețeaua Wi-Fi locală
- se setează pinul GPIO la care este conectat fotorezistorul
- se setează atenuarea valorilor citite cu ajutorul ADC-ului corespunzătoare tensiunii circuitului
- într-o buclă infinită se citește valoarea curentă de la senzor, se construiește payload-ul ce o conține (obiect JSON), se trimite către server și se așteaptă 1 minut între citiri

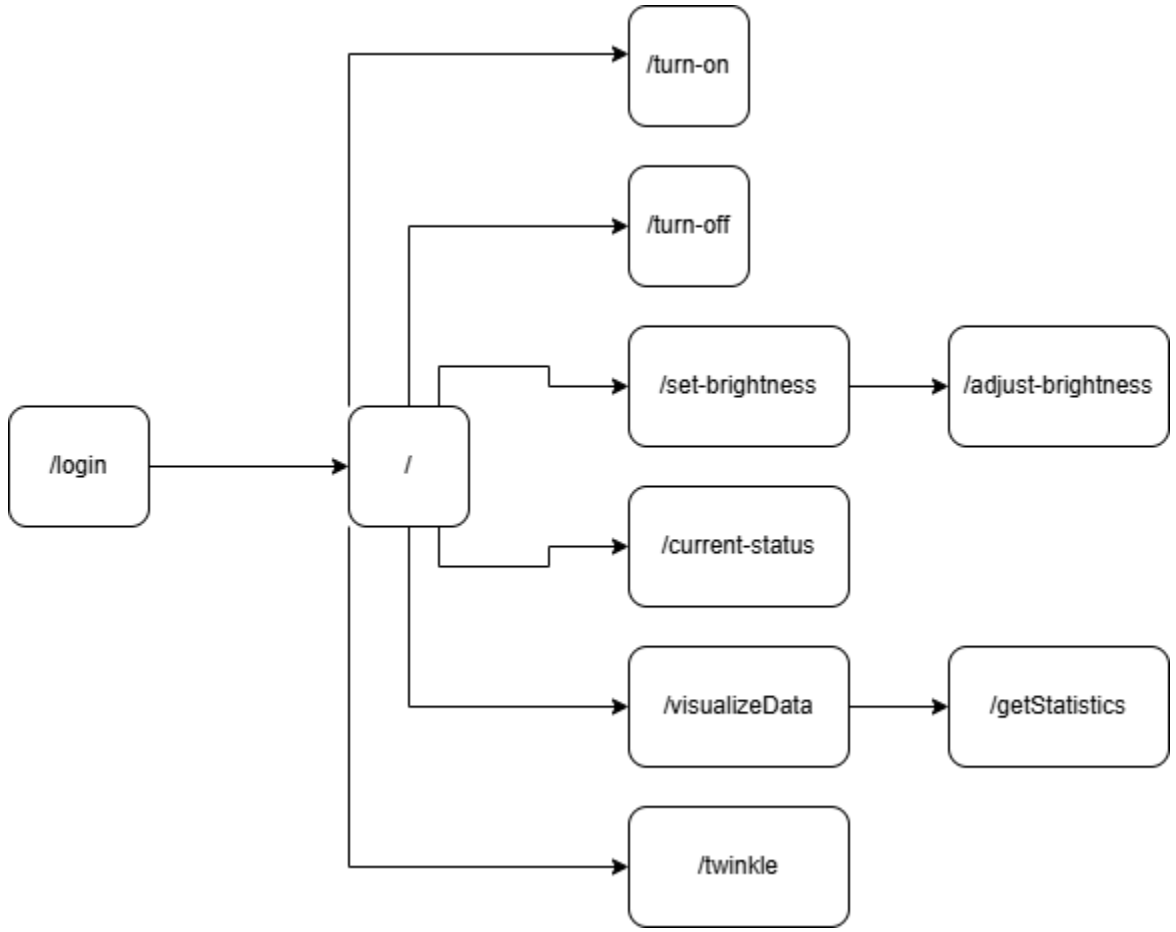
Implementare software a interfeței expuse utilizatorului

- server realizat cu ajutorul framework-ului Flask
- stocare a datelor în baza de date SQLite3
- fiecare pagină web are o rută corespunzătoare pentru expunerea operațiilor efectuate asupra actuatorului
- primirea datelor de la microcontroller – cereri POST

Operațiile aplicate actuatorului

- /turn-on: se trimite comanda de pornire a becului
- /turn-off: se trimite comanda de stingere a becului
- /set-brightness: se trimite comanda de actualizare a luminozității becului conform valorii introduse de utilizator în pagină
- /adjust-brightness: se trimite comanda de actualizare a luminozității becului conform ultimei valori citite de la senzor
- /current-status: se trimite comanda de determinare a stării curente a becului (dacă este pornit/oprit și procentul de luminozitate)
- /twinkle: se trimite comanda de creare a unui efect luminos pentru bec: clipire de 3 ori

Structura server-ului



- navigarea prin paginile web

Sistem de alertare și notificare

- utilizatorul primește constant feedback în timpul navigării printre paginile web puse la dispoziție de server, prin afișarea de mesaje de informare în caz de succes sau de eroare (pentru a ști cum se pot remedia acestea)

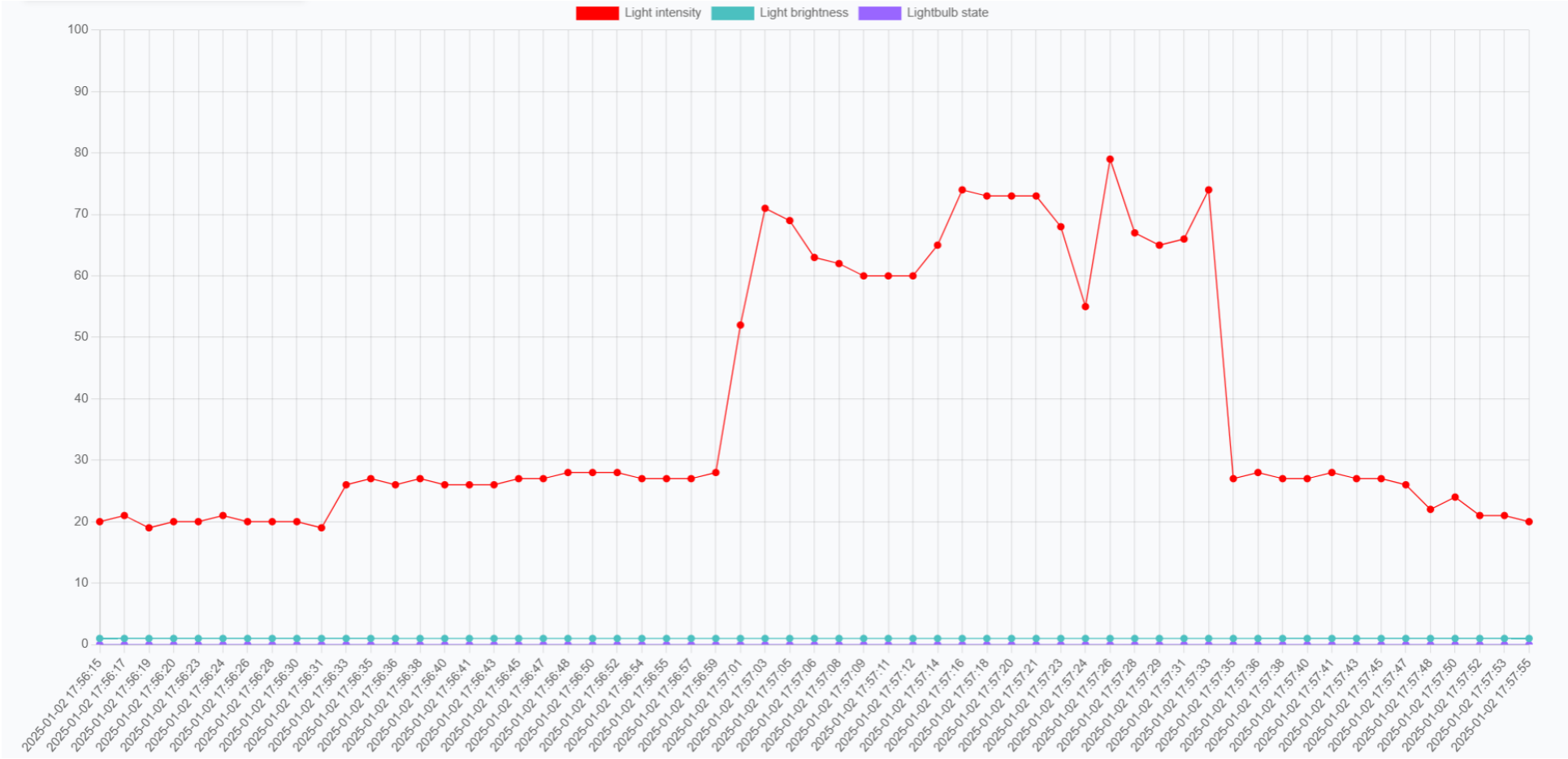
- în cazul în care acesta dorește să fie notificat cu privire la statisticile determinate la un moment dat se pot introduce la logare datele necesare unui bot de Telegram pentru a primi mesajele și a putea ține ușor evidența valorilor intensității luminoase din încăpere

Vizualizarea și procesarea datelor

- graficul rezultat în urma prelucrării datelor (cele primite de la senzor cât și cele determinate prin aflarea stării curente ale actuatorului) este construit cu ajutorul framework-ului Chart.js
- graficul se actualizează în mod automat o dată la 5 secunde
- datele primite de către metoda ce construiește obiectul de tip Chart provin de la un end-point auxiliar /getDBData
- se trasează evoluția intensității luminoase din mediu, a gradului de iluminare a becului și a stării acestuia (oprit/pornit)
- statisticile se determină pe baza datelor aflate la un moment dat în tabelă (media valorilor intensităților luminoase preluate de la senzor și cele unice înregistrate); de asemenea, se estimează următoarea valoare a intensității cu ajutorul metodei ARIMA

Vizualizarea și procesarea datelor

- exemplu de grafic rezultat



Securitate

- transmiterea datelor de la senzor către server se realizează prin protocolul HTTP securizat, payload-ul conținând headerele specifice necesare, care descriu tipul operației, ruta la care ajung datele, tipul și lungimea lor
- paginele web asociate server-ului au certificate generate care atestă siguranța conexiunii
- microcontroller-ul și server-ul comunică prin intermediul socketilor, care dispun de criptare SSL
- în cadrul programului uploadat pe plăcuță se deschide câte un socket pentru fiecare pachet de date care se trimite

Raport detaliat și demo

- modul de funcționare este prezentat aici:

Demo proiect PR IoT - YouTube

- detaliile de implementare sunt descrise aici:

https://drive.google.com/drive/folders/1ruRRRpS9j56gKC-UrDg87xuvwMTC82Lo?usp=sharing

Bibliografie

- How To Use Web Forms in a Flask Application – GeeksforGeeks
- How to Create an ARIMA Model for Time Series Forecasting in Python - MachineLearningMastery.com
- CSS Tutorial
- How To Use an SQLite Database in a Flask Application | DigitalOcean
- Chart.js
- html - Application not picking up .css file (flask/python) - Stack Overflow
- ESP32 In MicroPython: SSL Sockets
- Flask Rendering Templates – GeeksforGeeks
- Python Post JSON using requests library
- Python Tutorial: ssl — TLS/SSL Wrapper for Socket Objects - USAVPS.COM