**Laborator 12 - IoT – documentatie**

**Aplicatia propusa**

Pornind de la etapele descrise anterior adaptati serviciul REST dezvoltat în cadrul laboratorului anterior astfel încât sa realizati urmatoarele functionalitati:

1. **apel de metoda GET** peste ID-ul unui senzor se va citi o valoare furnizata de la un senzor; optional, puteti simula mai multi senzori
2. **apel de metoda POST** peste ID-ul unui senzor se va crea un fisier de configurare care va permite, de exemplu, modificarea scalei de reprezentare a valorii masurate:
   1. daca fisierul nu exista, atunci se creeaza cu nume implicit
   2. daca fisierul exista, nu se va permite recrearea acestuia prin metoda POST si se va întoarce un raspuns de tip 400 Bad Request, 406 Not Acceptable sau 409 Conflict (pentru toate cele 3 coduri se recomanda ca raspunsul sa includa un corp de mesaj care sa ofere detalii cu privire la eroarea generata)
3. **apel de metoda PUT** peste ID-ul unui senzor/numele fisierului de configurare se va înlocui fisierul de configurare (daca fisierul nu exista, nu se va permite crearea acestuia si se va întoarce unul dintre codurile de raspuns mentionate anterior)

**Modul de abordare:**

1. **Utilizand ESP**

**Structura proiect:**

project/

├── src/

│   └── main.c <-- codul principal pt ESP

├── platform.ini

├── partitions.csv

Prima data am pornit construirea acestei aplicatii prin citierea efectiva a datelor unui sensor fizic. Asadar cu ajutorul ESP Wroom 32 am si prin intermediul platofrmei PlatformIO am reusit sa citesc datele de la un senzor PIR de la GPIO23. Mai jos se poate vedea o imagine cu placuta si senzorul.

A circuit board with wires

AI-generated content may be incorrect.

Ceea ce face proiectul bazat pe o placa ESP32 și un senzor de mișcare PIR este capabil pana acum să expună datele senzorului printr-o interfață web accesibilă în rețeaua locală Wi-Fi.

* Detectează mișcare printr-un senzor PIR conectat la unul dintre pinii săi digitali.
* ESP32 se conectează la o rețea Wi-Fi specificată (SSID și parolă). După conectare, IP-ul local este disponibil pentru accesul la server.
* Rulează un server HTTP local, care permite accesarea datelor senzorului în timp real printr-un browser.
* Stochează un fișier de configurare pentru senzor în memoria internă (SPIFFS).
* Expune o interfață web minimalistă care:
  + Afișează valoarea curentă a senzorului (0 sau 1). (GET)
  + Permite crearea unui fișier de configurare pentru senzor. (POST)
  + Actualizează automat valoarea senzorului la fiecare câteva secunde.
  + Permite actualizarea fisierului de configurare pentru senzor. (PUT)

In imaginea de mai jos se poate observa interfata web:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Metodele REST folosite cu ESP:

|  |  |  |
| --- | --- | --- |
| Metodă HTTP | Endpoint | Scop |
| GET | /sensor/pir1 | Citește starea senzorului (valoare 0/1) |
| POST | /sensor/pir1 | Creează un fișier de configurare pentru senzor |
| PUT | /sensor/pir1 | Actualizeaza fisierul de configuratie pentru senzor |

Dar fiindca nu am fost multumit de rezultate si din lipsa de informatii am trecut la o alta abordare si anume utilizand Flask.

1. **Utilizand Flask**

**Structura proiect:**

project/

├── app.py                <-- codul principal Flask

├── sensors/              <-- folder cu date senzori

│   └── sensor\_temp.cfg

│   └── sensor\_light.cfg

| ...

├── index.html <-- interfata web

* 1. *Listare senzori existenti (GET)*

Acest endpoint listează toți senzorii existenți, adică toți cei care au un fișier .cfg în folderul sensors/.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. *Citeste valoare senzor (GET)*

Acest endpoint GET /sensors/<sensor\_id>/value citește tipul unui senzor din fișierul său de configurare și returnează o valoare simulată corespunzătoare acelui tip, sub formă de răspuns JSON. In caz contrar, daca nu exista un fisier de configuratie corespunzator id-ului senzorului, va returna un mesaj de eroare.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. *Adaugare senzor nou (POST)*

Endpointul POST /sensors/<sensor\_id> creează un nou fișier de configurare pentru senzorul specificat, dacă acesta nu există deja, și salvează în el tipul senzorului și data creării. Dacă fișierul există deja, returnează o eroare și sugerează folosirea metodei PUT pentru actualizare.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. *Actualizare senzor existent (PUT)*

Endpointul PUT /sensors/<sensor\_id> actualizează fișierul de configurare al unui senzor existent cu un nou tip de senzor și data actualizării. Dacă fișierul nu există, returnează eroare și nu permite crearea lui prin această metodă.

A screenshot of a computer

AI-generated content may be incorrect.