SISTEM IOT PENTRU MONITORIZAREA CALITATII AERULUI

Proiect realizat de:

Ciorgan Andrei-Florian, 345C1

Cuprins:

1. Date generale..... 3
2. Arhitectura..... 4
3. Implementare..... 5
4. Vizualizare si procesare de date..... 6
5. Sistem de notificari..... 8
6. Provocari si solutii..... 8
7. Unelte necesare..... 9
8. Mod de rulare..... 9
9. Concluzii..... 10

1. Date generale

Proiectul propus are in vedere creare unui sistem inteligent de monitorizare a diversilor parametrii ai aerului cu scopul monitorizarii continue a calitatii acestuia prin integrarea a mai multi senzori si actuatori cu roluri diferite. Acest sistem va oferii detalii despre temperatura, umiditate si calitate a aerului cu referinta la numarul de particule de gaz din jurul acestuia.

Sistemul va avea in componenta un minim de 2 senzori: un senzor DHT11 ce va colecta date despre temperatura si umiditate, si un senzor MQ135 care va sintetiza date despre cantitatea de compusi chimici din atmosfera (NH3, Nox, alcool, benzen, CO2, etc).

Datele de la senzori vor fi transmise cu ajutorul protocolului MQTT catre un server local hostat pe un laptop in aceasi retea cu senzorii. Acestia vor trimite date pe diverse topicuri la care un client (o aplicatie pe telefon) o sa isi dea subscribe ca sa le afiseze intr-un dashboard ce va cuprinde diverse grafice si metode de vizualizare a datelor pentru a asista utilizatorul in inteleagerea informatiilor colectate.

Totodata, aplicatia ii va pune la dispozitie utilizatorului o metoda de pornire si de oprire a senzorilor in vederea optimizarii consumului de baterie utilizata.

Detectarea temperaturii si umiditatii: Senzorul DHT11 este capabil sa colecteze date precise referitoare la temperatura si umiditate din mediul inconjurator. Aceste informatii sunt esentiale in evaluarea conditiilor climatice interne sau externe, fiind principalele caracteristici utiliuzate in optimizarea comfortului ambiental.

Detectarea gazelor din aer: Senzorul MQ135 este un dispozitiv sensibil la o gama larga de gaze, inclusiv amoniac, noxe, alcool, benzen si CO2. Acesta furnizeaza date valoroase despre poluarea

aerului, oferind o imagine clara a concentratiilor de compusi chimici in atmosfera. Astfel, sistemul va ajuta utilizatorii sa monitorizeze si sa gestioneze mai eficient calitatea aerului, oferind posibilitatea de a interveni in cazurile in care nivelul poluantilor depaseste limitele acceptabile.

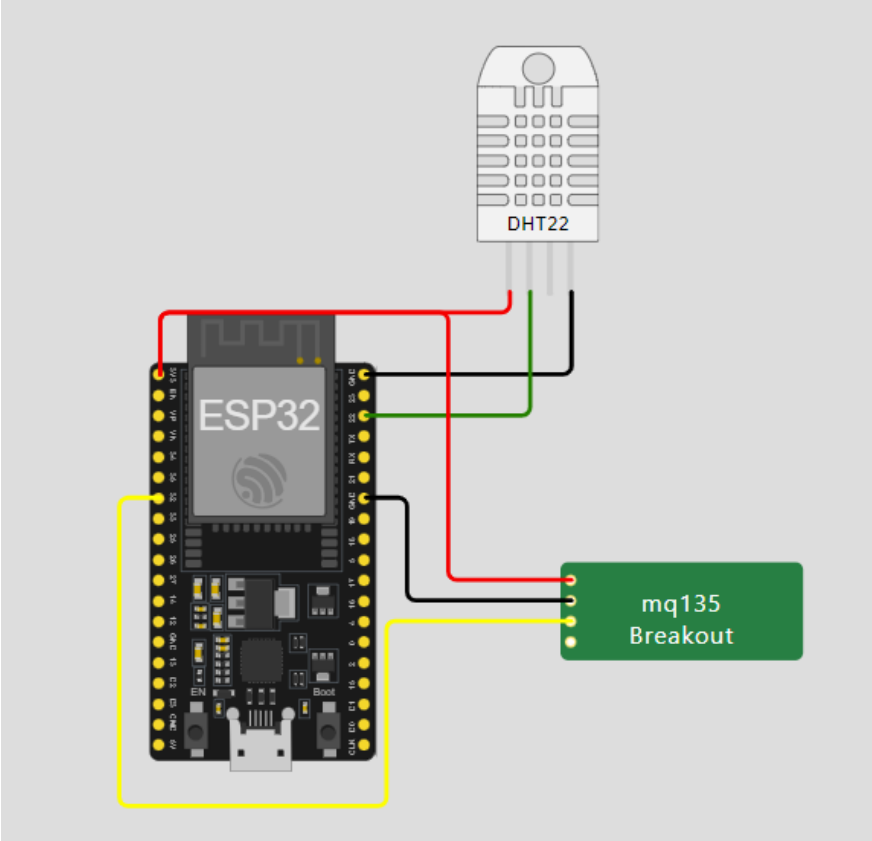
Monitorizarea datelor: Datele colectate de senzori vor fi transmise continuu catre serverul local prin protocolul MQTT, asigurand o comunicare rapida si eficienta. Clientul (aplicatia pe telefon) va permite utilizatorului sa acceseze informatiile in timp real, vizualizand grafice interactive si date istorice, astfel incat sa poata evalua tendintele in calitatea aerului. Aceasta monitorizare activa va contribui la luarea unor decizii informate privind sanatatea si siguranta mediului in care se afla utilizatorul.

2. Arhitectura

Arhitectura proiectului descris mai sus se va baza pe o retea IoT de tip star, in care senzorii cat si utilizatorii se vor conecta la un nod central, serverul MQTT, care se va ocupa de impartirea mesajelor pe topicuri si notificarea utilizatorilor la updatarea acestora. Datele, fiind procesate in apropierea locului in care sunt preluate, determina topologia clasica de Edge computing.

In imaginea de mai jos este prezentata o schema simplificata in care avem un singur dispozitiv de preluare al datelor (un singur microcontroler Raspberry Pi Pico W) si un singur client de subscribe pentru un utilizator. Arhitectura permite, in mod evident, un numar mai mare de conexiuni de ambele tipuri.

Conexiunea exacta dintre microcontroler si cei doi senzori poate fi vazuta mai in detaliu in imaginea urmatoare.



Protocoale utilizate: Sistemul va folosi protocolul MQTT (Message Queuing Telemetry Transport) pentru transmiterea eficienta a datelor intre senzori si serverul local. MQTT este ideal pentru aplicatii IoT, datorita utilizarii reduse a latimii de banda si fiabilitatii in transmiterea datelor in timp real. Comunicarea intre componentele sistemului se va realiza prin intermediul unei retele WiFi, asigurand o conectivitate stabila si acces facil la informatiile transmise. Alegerea acestor protocoale contribuie la implementarea unui sistem eficient, scalabil si usor de utilizat.

Senzori si actuatori utilizati: Sistemul va include senzorul DHT11 pentru masurarea temperaturii si umiditatii, precum si senzorul MQ135 pentru detectarea nivelului de compusi chimici din aer, oferind o monitorizare cuprinzatoare a parametrilor de calitate a aerului. Totodata, utilizatorul va avea in aplicatie o metoda de a opri / porni transmiterea de date de la microcontroler (un buton sau o lista de selectie a starii de transmisie).

3. Implementare

Proiectul este alcatuit din trei parti principle: un server MQTT, senzorii conectati la un microcontroler cu capabilitati de comunicare wireless (in cadrul proiectului s-a utilizat un ESP32, detaliile pentru aceasta alegere urmand sa fie precizate mai jos) si o aplicatie Android folosita pentru vizualizarea datelor trimise de agentul de colectare de date. Broker-ul MQTT este reprezentat de un server standard MQTT rulat local cu ajutorul aplicatiei mosquitto pe portul 1883.

Agentul de colectare de date am decis sa fie rulat intr-un simulator pentru a ne permite testarea si salvarea progresului mai usor decat daca implementam proiectul pe o arhitectura reala

(schimbarea de la mediul simulat la cel real se reduce la incarcarea fisierelor de lucru pe placuta utilizata). In cadrul acestui proiect, am folosit simulatorul Wokwi, accesat din VSCode prin extensia cu acelasi nume.

Sistemul de colectare de date utilizeaza doi senzori: un DHT11/DHT22, componenta ce se afla deja in plaja de elemente valabile in cadrul simulatorului, si un senzor MQ135 care a fost creat prin intermediul chipurilor customizabile, o posibilitate in cadrul aplicatiei de a defini componente specializate ce nu au o implementare oficiala. Simulatorul Wokwi permite crearea unor astfel de componente customizabile prin definirea functionalitatii acestora in libajul C, Rust sau Verilog. Codul sursa specific senzorului proiectat este valabil in cadrul proiectului, insa simulatorul se foloseste doar de varianta precompilata (.wasm).

Senzorul MQ135 proiectat simuleaza colectarea si trimiterea de date pentru un senzor de acest tip. In cadrul simularii, valoarea trimisa este configurata cu ajutorul unui slider ce cuprinde valori de la 0 la 1000. Pentru modificari in cadrul functionalitatii acestui chip este necesara recompilarea acestuia (cel mai usor mod de a face asta este prin suprascrierea exemplului oficial de compilare a unui circuit inversor, lucru datorat docker-uli specific).

Microcontrolerul utilizat in simulator este ESP32, rulat pe baza firmware-ului inclus in proiect. Puterea de procesarea a acestui microcontroler este relativ mare comparativ cu functionalitate ce este necesara pentru acest sistem, insa simulatorul Wokwi are integrata parte de comunicare Wifi doar pe aceasta placuta. In realitate, un microcontroler Raspberry Pi Pico W este suficient.

Comunicarea dintre sistemul de colectare de date si serverul brokerului se realizeaza pe baza de MQTT, functionalitate trimiterii si primirii de mesaje fiind implementata in cadrul fisierului simple.py, fisier ce va trebui incarcat pe placuta pentru functionarea corecta a programului principal. Comunicarea cu sistemul local se face pe baza de port forwarding, datele specifice fiind incluse in wokwi.toml.

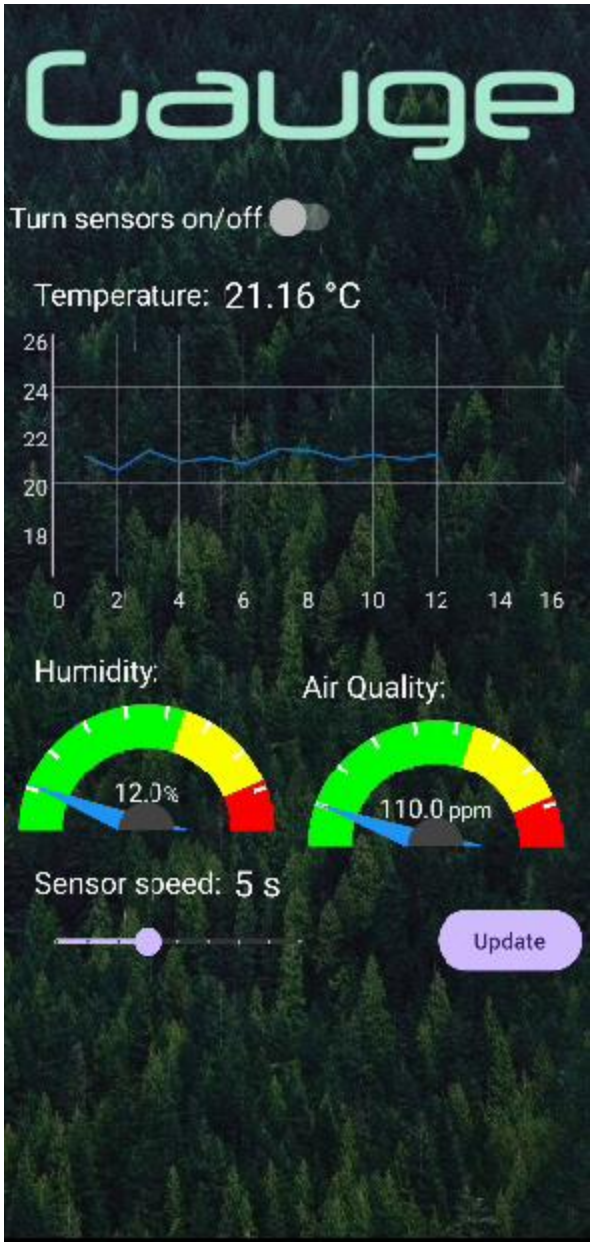
4. Vizualizare si procesare de date

Initial, atat sistemul de colectare de date cat si clientul de vizualizare (aplicatia de telefon) isi initiaza conexiune la serverul mosquitto si isi declara preferintele despre senzorii pe care vor publica sau ii vor astepta pentru procesare.

Odata la 5 secunde, sunt preluate de la senzori datele monitorizate (temperatura, umiditate si ppm) si apoi sunt trimise catre server. Ininte ca datele sa fie trimise de la simulator, am decis sa aplic un zgomot artificial care sa ne permita sa analizam datele intr-o forma mai realista, o forma posibila a datelor daca acestea ar fi fost colectate dintr-o zona reala.

Odata ajunse, broker-ul se va ocupa de trimiterea datelor catre aplicatia de vizualizare.

Vizualizarea datelor se face pe baza unei aplicatii Android creata in Android Studio. Aplicatia este scrisa in mare parte in Java, partea de afisarea grafica fiind realizata in XML. Pentru temperatura, am decis sa afisez datele in cadrul unui time series, reprezentate de un grafic in timp. Pentru umiditate si ppm, am folosit o reprezentare grafic cu ajutorul unei librarii de vitezometre.

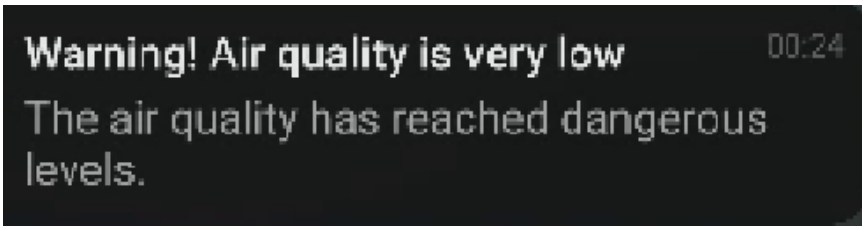


Un utilizator are si posibilitatea de a interactiona cu sistemul de colectare de date. In cadrul aplicatiei, am adaugat posibilitate de a specifica starea agentului de colectare de date: pornit, in cadrul caruia acesta continua sa trimita date din 5 in 5 secunde, sau oprit, in cazul in care vrem sa minimizam consumul bateriei, factor important in cadrul oricarui sistem IoT. Acest lucru se realizeaza prin trimiterea unui mesaj special MQTT pe care agentul il monitorizeaza constant in cadrul unui thread separat de cel de colectare de date de la senzori.

Totodata, daca dorim o viteza specifica de trimitere a mesajelor, un utilizator poate ajusta sliderul de viteza si sa apese butonul Update. Acesta va trimite un mesaj cu valoarea selectata catre agentul de colectare de date ce va schimba rata de transmitere de mesaje, un mesaj la cate secunde sunt specificate.

5. Sistem de notificari

Datorita importantei calitatii aerului pe care il respiram de zi cu zi, a fost imperativa adaugarea unei functionalitati de notificare care sa anunte un utilizator in cazul unor probleme ce ar putea afecta negativ sanatatea acestuia.



Senzorul MQ135 este cel care ofera date despre numarul de compusi chimici din atmosfera. Astfel, daca valoarea returnata de acest senzor depaseste 600 (prag ales arbitrar be baza analizei valorilor posibile si plauzibile specifice unui astfel de senzor), aplicatia va genera o notificare in care se specifica calitatea proasta a aerului si necesitatea utilizatorului de a lua actiune.

6. Provocari si solutii

De-a lungul proiectului am intampinat diverse provocari, in particular in cadrul asigurarii unui mediu propice pentru simularea senzorilor. Problema principala in cadrul agentului de colectare de date se refera la faptul ca nu am reusit initial sa creez o conexiune in afara retelei default utilizate de simulator (o retea Wifi fictiva numita ‚Wokwi-Guest' fara parola). Problema a fost cauzata de lipsa functionalitaii de comunicare wireless pe placuta cu care lucram initial, un Raspberry Pi Pico W. Odata aflata cauza problemei, am ajuns repede la o solutia prin inlocuirea microcontrolerului cu unul mai avansat, ESP32, desi daca dorim sa refacem proiectul in viata reala, acest microcontroler va fi prea puternic relativ la puterea de procesare utilizata.

Alte probleme intalnite includ modul de compilare al binarului specific chipului custom, MQ135. Initial am incercat compilarea lui locala, atat pe Windows cat si pe WSL. Tentativele au esuat datorita nefunctionarii adecvate a diverselor programe necesare pentru acest task. Solutia s-a rezumat la reanalizarea exemplului oficial si utilizarea aceluiasi docker.

In fine, au mai aparut niste probleme legate de incarcarea de fisiere pe placuta simulata. Aceste probleme, din pacate, nu au putut fi solutionate intrucat eroarea este generata de simulator in sine. Detaliile specifice rularii proiectului si evitarii acestei probleme sunt specificate mai jos.

7. Unelte necesare

Desi proiectul reprezinta un simplu sistem de monitorizare a calitatii aerului, pentru functionalitatea acestuia sunt necesare mai multe componente specifice. Lista cu extensii si aplicatii necesare cuprinde:

- Extensia Wokwi – aceasta extensie este utilizata pentru rularea circuitului simulat
- Mosquitto – aplicatie care ne permite rularea unui broker MQTT
- Android Studio – folosit pentru generarea executabilului aplicatiei Android

Pe langa aceste, am utilizat si niste fisiere speciale preluate din laboratoare trecute si din exemple de chipuri customizate:

- Simple.py – implementarea functionalitailor de baza MQTT
- Wokwi-api.h – utilizat pentru rularea binarelor chipurilor customizate
- ESP32_GENERIC-20240602-v1.23.0.bin – firmware-ul specific unui ESP32 cu functionalitate Wifi inclusa

8. Mod de rulare

Pentru rularea corecta a proiectului, trebuie urmariti pasii de mai jos:

1. Initiati serverul MQTT (rulat pe mosquitto)
2. (se poate realiza in paralel cu 5) Setati ip-ul brokerului si porniti simulatorul Wokwi din VSCode (F1->Wokwi:Start simulator).
    a. In cazul in care primiti o eroare Missing Licence, este necara crearea unui cont Wokwi si activarea unei licente temporare
3. Odata pornit, simulatorul trebuie tinut activ ca sa se poata trimite date
    a. Daca simulatorul este minimizat, acesta va fi oprit
    b. Pentru functionarea corecta, atata timp cat se doreste functionarea acestuia, simulatorul trebuie sa fie vizibil
4. Intr-un terminal normal, trebuie incarcate cele doua fisiere (simple.py si main.py).
    a. Prima data, incarcam siple.py utilizand comanda:

python -m mpremote connect port:rfc2217://localhost:4000 fs cp src/simple.py :simple.py

    b. Daca nu ruleaza imediat, opriti executia (Ctrl+C) si rulati din nou. Aceasta eraore este cauzata de simulator si se datoreaza nefinisarii stadiului de initializarea a microcontrolerului.
    c. Apoi, trebuie rulat codul main.py cu ajutorul comenzii:

python -m mpremote connect port:rfc2217://localhost:4000 run src/main.py

5. (se poate realiza in paralel cu 2) Porniti Android Studio si setati ip-ul brokerului.
6. Rulati aplicatia generata, Gauge Enjoyer.
    a. Daca este prima data cand rulati proiectul, va trebui conectat telefonul la dispozitiv si construit executabilul.

9. Concluzii

In cadrul proiectului, am utilizat un singur agent de colectare de date, insa putem genera mai multi, fapt datorat de natura usor replicabiala a circuitului simulat, astfel oferind redundanta in retea si sporind calitatea valorilor observate.

Totodata, putem sa conectam mai multi clienti aplicatie, permitand unui numar mai mare de utilizatori sa se foloseasca de acest sistem.

Acest proiect reprezinta un simplu model de baza de la care putem pleca pentru a creea sisteme mai complexa de monitorizare a aerului, aici fiind implementate principalele functionalitati necesare unui astfel de aplicatie, lasand loc atat pentru imbunatatirii cat si pentru extinderi ulterioare.