

Sistem de udare automata a plantelor

Cuprins

I. Introducere.....	2
Obiective	2
Actuatori	2
Senzori	2
II. Arhitectura.....	3
Componente hardware	3
Software.....	5
Schema electrica.....	6
Topologie	7
III. Implementare	8
Structura Aplicației.....	8
Configurarea hardware-ului	8
Configurarea software-ului	8
Configurarea sistemului de notificare	14
IV. Vizualizare si procesare de date	15
1. Colectarea și Transmiterea Datelor.....	15
2. Procesarea Datelor	15
3. Activarea pompei de apa.....	19
V. Funcționalități de Alertare	20
VI. Securitate	21
1. Criptarea comunicării MQTT	21
2. Asigurarea confidențialității in momentul autentificării Gmail	21
3. Criptarea conexiunii cu serverul de Gmail	22
Pornirea aplicației.....	23

I. Introducere

Obiective

Sistemul IoT pune la dispozitia utilizatorului in timp real informații despre nivelul de umiditate din sol, nivelul apei din rezervorul din care se va uda solul, precum si starea pompei. Acest proiect este destinat unui sistem de irigație automatizat pentru plante.

Sistemul notifica utilizatorul in momentul când nivelul de umiditate din sol scade sub un anumit nivel. Acesta poate oricând sa activeze pompa de apa prin intermediul unei aplicații web. Cloud-ul si modulul Wi-Fi ESP8266 vor comunica prin protocolul de comunicatie MQTT.

Serverul reprezentat de laptop preia datele de pe cloud trimise de ESP folosind tot MQTT si stochează și pune la dispoziție o platforma de vizualizare printr-o interfață web. Clientul final comunica cu serverul web prin protocolul HTTP.

Actuatori

- **pompa de apa**

Aceasta va fi activata de clientul final care acceseaza interfata web si decide pe baza informatiilor daca porneste sau opreste pompa

Senzori

- **nivel de umiditate din sol**

Măsoară umiditatea solului pentru a determina dacă plantele necesită apă.
nivel apa din rezervor

- **nivelul apei din rezervor**

Măsoară nivelul apei din rezervor pentru a preveni epuizarea apei și pentru a asigura o gestionare optimă a resurselor

Acești senzori vor prelua datele necesare utilizatorului.

II. Arhitectura

Componente hardware

- **Senzor de umiditate a solului:** Pentru a detecta nivelul de umiditate din sol; dacă scade sub un anumit nivel, microcontroller-ul trimite o notificare utilizatorului, și el va decide dacă trebuie udate plantele

 Optimus Digital



https://www.optimusdigital.ro/en/humidity-sensors/73-ground-humidity-sensor-module.html?gad_source=1&gclid=CjwKCAiAmfq6BhAsEiwAX1jsZyv3Hw0ktxKrLxnzthy02WAqDhtn-xwH9s1G2VA71xkORrzJyLnQUxoChOUQAvD_BwE

- **Pompă de apă:** Pentru transportul apei din rezervorul de apă către sol

 Optimus Digital



https://www.optimusdigital.ro/ro/altele/4149-mini-pompa-de-apa-submersibila.html?search_query=pompa+de+apa&RESULTS=7

- **Releu:** Pentru a controla pompa de apă



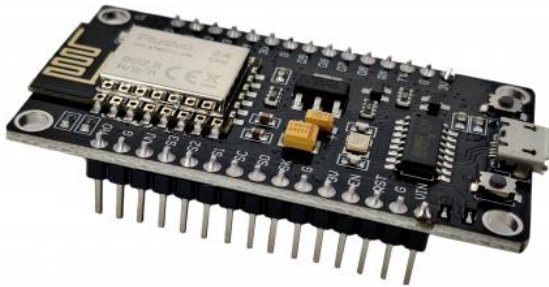
https://www.optimusdigital.ro/ro/electronica-de-putere-module-cu-releu/13084-modul-releu-cu-un-canal-comandat-cu-5-v.html?search_query=releu&results=106

- **Rezervor de apă:** Un recipient pentru a stoca apa pe care pompa o va folosi
- **Senzor de nivel al apei:** Senzor pentru a monitoriza nivelul apei din rezervor



https://www.optimusdigital.ro/ro/senzori-senzori-ultrasonici/9-senzor-ultrasonic-hc-sr04-.html?search_query=senzori+ultrasonici&results=10

- **Modul WiFi ESP8266:** Este responsabil de gestionarea mesajelor între dispozitivele IoT și serverul web. Publică datele pe anumite topicuri, iar serverul web este abonat la aceste topicuri pentru a obține datele și a le vizualiza pe aplicația web.



https://www.optimusdigital.ro/ro/placi-cu-wifi/266-placa-de-dezvoltare-wifi-cu-esp8266.html?search_query=esp8266&results=62

- **Alte componente:** Rezistențe, tranzistoare și fire de conectare: necesare pentru circuitele auxiliare

Software

1. Biblioteci Arduino pentru MQTT și Wi-Fi și pentru conectarea la Cloud:

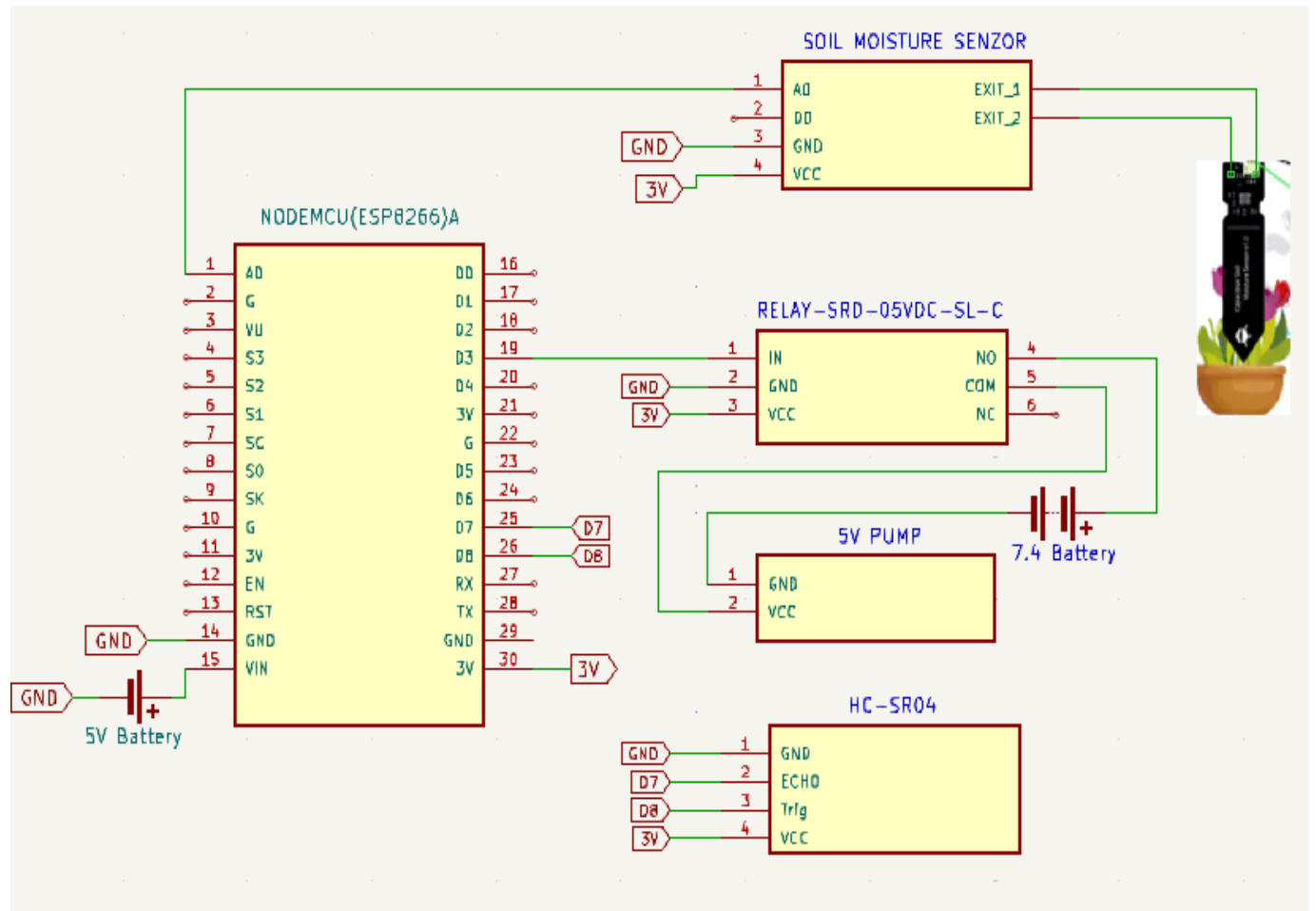
- ESP8266WiFi.h
- WiFiClientSecure.h
- PubSubClient.h
- ArduinoJson.h

2. Broker MQTT: Amazon IOT Core: Cloud-ul în care se stochează datele

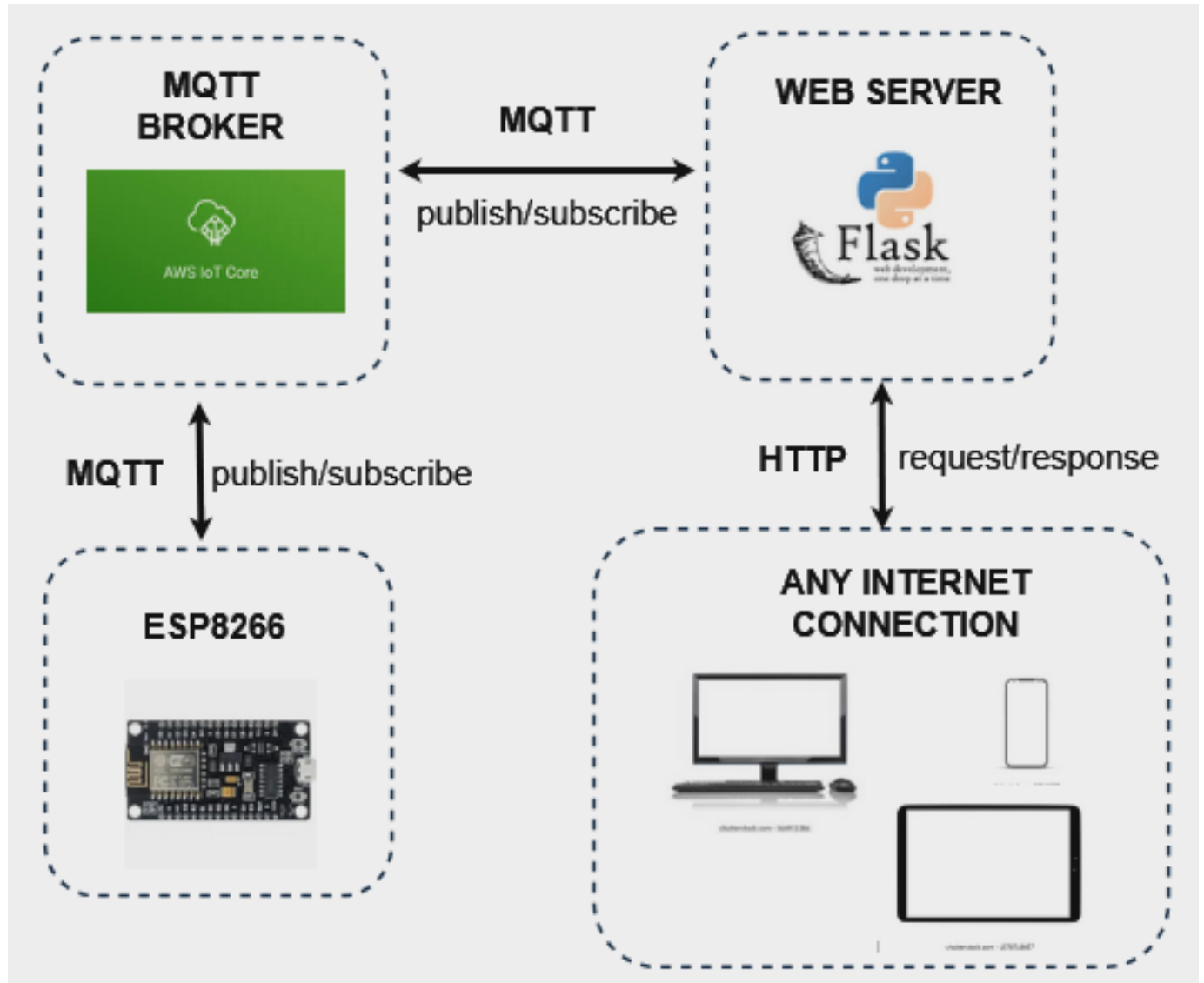
3. Server Web (Python + Flask): Crearea unui server simplu care să primească și să afișeze datele senzorilor; Serverul web se conectează la brokerul MQTT pentru a primi datele senzorilor și pentru a trimite comenzi către dispozitivele IoT (pentru a porni/opri pompa).

4. Grafana și InfluxDB folosind Docker: pentru stocarea și vizualizarea datelor

Schema elettrica



Topologie



Protocoale de comunicatie:

- MQTT (Message Queuing Telemetry Transport) este protocolul utilizat pentru a transmite date între microcontrolerul IoT și serverul web. Este un protocol ușor, eficient, perfect pentru dispozitive IoT care necesită o comunicare rapidă și în timp real.
- HTTP: Protocolul folosit pentru a comunica între utilizator și serverul web. Comenzile de control sunt trimise prin cereri HTTP (pornește/oprește pompa).

III. Implementare

Structura Aplicației

Folderul ESP8266: pentru codul scris pentru ESP8266

Folderul AWS_IOT_CERTIFICATES: păstrarea certificatelor necesare conectării la AWS

Folderul Web_Server: pentru serverul web

- **web_page/templates/index.html:** pagina principala
- **web_page/static/style.css:** aspectul obiectelor din pagina
- **provisioning** -> folder-ul pentru configurările Grafana
- **docker-compose.yml** -> fișierul yml pentru pornirea micro-serviciilor care rulează în backend
- **grafana.ini** -> alt fișier de configurare pentru Grafana
- **run.sh** și **stop.sh** -> fișiere de pornire și deschidere a întregii aplicații

Configurarea hardware-ului

Am folosit schema electrica pentru a realiza conexiunile între componentele hardware. Conexiunile de baza sunt:

- **Senzor de umiditate a solului:** Pin de semnal conectat la un pin analogic al ESP.
- **HC-SR04:** Trig și Echo conectați la pini digitali ai ESP.
- **Releu pentru pompă:** Intrare de comandă conectată la un pin digital configurat ca OUTPUT.
- **Wi-Fi Module:** Incorporat în ESP8266 pentru conectivitate.

Configurarea software-ului

a. Cod pentru ESP32/ESP8266

Pentru implementarea dispozitivului IoT, am utilizat Arduino IDE împreună cu un set de biblioteci folosite pentru conexiunea la internet, criptarea datelor și comunicarea eficientă cu brokerul prin protocolul de comunicare MQTT.

Biblioteci utilizate

1. WiFiClientSecure:

- Permite conexiuni criptate pentru securizarea comunicării cu brokerul MQTT.
- Este esențială pentru transmiterea în siguranță a datelor între dispozitiv și server

2. PubSubClient:

- Folosește pentru comunicarea MQTT între dispozitiv și brokerul Amazon IoT Core.

- Gestionarea publicării și subscrierii la topicurile definite pentru transmiterea și primirea datelor.

3. **NTPClient:**

- Responsabilă pentru sincronizarea ceasului intern al dispozitivului cu un server NTP, asigurând corectitudinea temporală a mesajelor.

4. **ArduinoJson:**

- Utilizată pentru formatarea mesajelor în format JSON, necesar pentru schimbul de date structurate prin MQTT.

b. **Flask & Python**

Pentru gestionarea interfeței utilizatorului și a comunicării cu dispozitivele IoT, am utilizat **Flask**, un framework Python.

- **HTML și CSS:**

Interfața utilizator este realizată folosind HTML și CSS.

Include componente de afișare a datelor (umiditate sol, nivel apă) și butoane pentru pornirea/oprirea pompei.

Datele senzoriale sunt preluate și afișate în timp real utilizând iframe-uri cu link-uri la dashboard-uri Grafana.

- **Rute Definite:**

/: Ruta principală unde este afișată interfața utilizator.

/api/control: Ruta prin care serverul primește cereri pentru controlul pompei.

/api/data: Ruta apelată periodic, inițial pentru actualizarea datelor afișate în interfață, în stadiul actual este folosită pentru Debug.

- **Integrarea cu Grafana:**

Dashboard-urile Grafana sunt integrate în pagină prin iframe-uri, oferind o vizualizare detaliată a datelor colectate.

c. **InfluxDB și Grafana folosind Docker**

InfluxDB:

Este configurată pentru a primi datele primite de server.

Structura bazei de date este definită pentru a include:

- Timestamp.
- Valori senzor (umiditate, nivel apă).
- Starea pompei

Se trimite in baza de date sub forma:

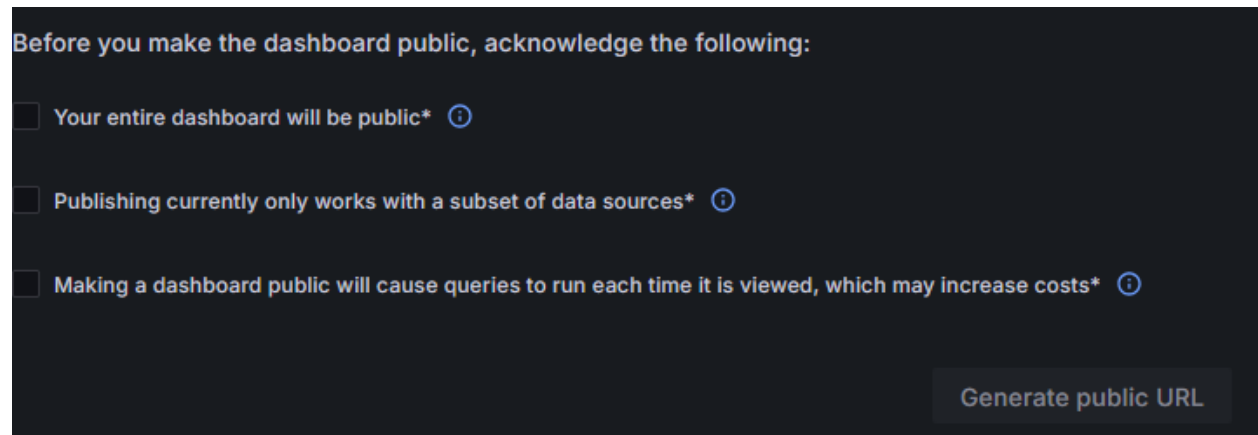
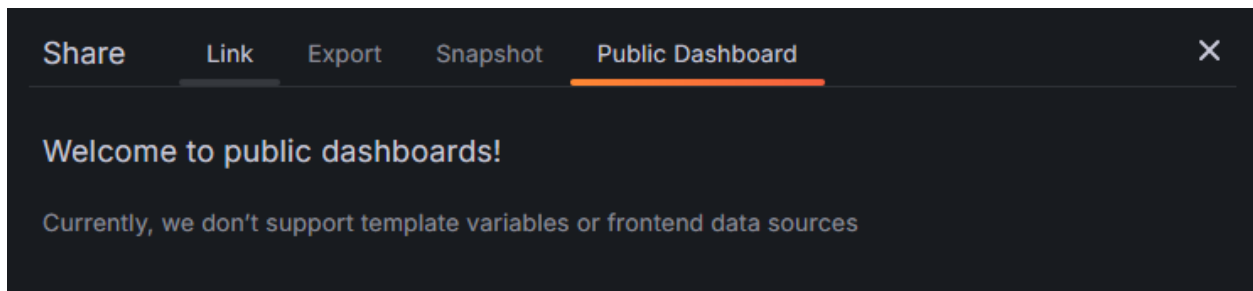
```
"measurement": {station}.{key},
  "tags": {
    "station": station,
    "key": key
  },
  "fields": {
    "value": float(value)
  },
  "timestamp": last_checked
}
```

Grafana:

Dashboard-urile au fost configurate pentru a afișa grafic evoluția în timp a datelor colectate.

Pentru a face publice dashboard-urile si catre alti utilizatori am folosit optiunea **Public**

Dashboards:



- Configurarea automată a fost realizată prin utilizarea folder-ului provisioning:
 - provisioning/dashboards -> setările dashboard-urilor ce se vor adauga automat la pornire
 - provisioning/datasources -> crearea legaturii automat cu baza de date InfluxDB:

```
apiVersion: 1

datasources:
- name: InfluxDB
  type: influxdb
  access: proxy
  url: http://influxdb:8086
  isDefault: true
  jsonData:
    version: Flux
    organization: my-org
    defaultBucket: my-bucket
    tlsSkipVerify: true
  secureJsonData:
    token: token
```

- Am folosit fisierul grafana.ini pentru a defini un timp mai mic de refresh

```
[dashboards]
min_refresh_interval = 1s
```

Configurare Docker:

- Fișierul docker-compose.yml definește serviciile pentru InfluxDB și Grafana.
- Volumele sunt folosite pentru persistența datelor și a configurației Grafana.

Setarea variabilelor de mediu in fișierul docker-compose.yml

Pentru Grafana se setează:

- GF_SECURITY_ADMIN_USER=admin: Definește utilizatorul administrator pentru Grafana
- GF_SECURITY_ADMIN_PASSWORD=password: Definește parola pentru utilizatorul administrator al Grafana
- GF_AUTH_ANONYMOUS_ENABLED=true: Permite accesul anonim în Grafana (fără autentificare). Setarea la true înseamnă că utilizatorii pot accesa Grafana fără a introduce un nume de utilizator și o parolă.

- **GF_SECURITY_ALLOW_EMBEDDING=true:** Permite încorporarea panourilor Grafana în site-ul create cu Python și Flask.

- **GF_AUTH_ANONYMOUS_ORG_ROLE=Viewer:** Definește rolul implicit pentru utilizatorii anonimi (au rol doar de citire).

Pentru InfluxDB se setează:

- **DOCKER_INFLUXDB_INIT_MODE=setup:** InfluxDB va fi configurat automat la prima rulare, creând utilizatori și organizații

- **DOCKER_INFLUXDB_INIT_USERNAME=admin:** Definește utilizatorul administrator pentru InfluxDB

- **DOCKER_INFLUXDB_INIT_PASSWORD=password:** Definește parola pentru utilizatorul administrator al InfluxDB

- **DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=token:** Definește un token de administrator pentru accesul API la InfluxDB

- **DOCKER_INFLUXDB_INIT_ORG=my-org:** Definește organizația implicită creată la inițializarea InfluxDB

- **DOCKER_INFLUXDB_INIT_BUCKET=my-bucket:** Definește bucket-ul (spațiul de stocare) implicit în care vor fi adăugate datele în InfluxDB

d. Brokerul MQTT

Am folosit **Amazon IOT Core**

Crearea și Gestionarea Certificatelor

În AWS IoT Core, pentru fiecare dispozitiv (Thing) s-a creat un set de certificate:

- **Certificatul CA.**
- **Certificatul clientului.**
- **Cheia privată.**

Certificatele sunt descărcate și utilizate local pe dispozitivul ESP32/ESP8266.

Exemplu de configurare în AWS IoT Core:

- Se creează un nou „Thing” (dispozitiv).
- Se generează un certificat pentru „Thing”.
- Certificatul este descărcat și utilizat în codul ESP32/ESP8266.
- Politicile de securitate sunt configurate pentru a controla accesul la topicurile MQTT.

Configurarea acestuia s-a realizat creând un nou „Thing” și un nou „Certificat”:

[AWS IoT](#) > [Manage](#) > Things

Things (1) [Info](#)

Create things

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

< 1 >

<input type="checkbox"/>	Name	Thing type
<input type="checkbox"/>	PR_project	-

[AWS IoT](#) > [Security](#) > Certificates

Certificates [Info](#)

X,509 certificates authenticate device and client connections. Certificates must be registered with AWS IoT and activated before a device or client can communicate with AWS IoT.

[Certificates](#) | Certificates you've transferred

Certificates (1)

< 1 >

<input type="checkbox"/>	Certificate ID	Status	Created
<input type="checkbox"/>	4d7ed773fdc6c79be89e7e1bb605a9fd3d6824432912ba9d211820c3c20a2714	Active	December 11, 2024, 10:38:56

Certificatele se regasesc in folderul AWS_IOT_Certificates si sunt utilizate atat de dispozitivul IOT, cat si de serverul web pentru a comunica si a pastra securitatea datelor.

e. Comunicatia MQTT

Se folosesc 4 topicuri:

A. de la dispozitivul ESP8266 la server

- ESP8266/umiditate_sol
- ESP8266/nivel_apa
- ESP8266/stare_pompa

B. de la server la ESP8266

- web_server/command -> se controleaza pompa printr-o comanda a utilizatorului

Configurarea sistemului de notificare

Se generează o parolă de aplicație Gmail

- Se accesează contul Google: <https://myaccount.google.com>.
- Se navighează la „Securitate”.
- Se activează autentificarea în doi pași.
- Se generează o parolă de aplicație
- Se utilizează această parolă în scriptul python în locul parolei contului principal.

Se folosește serviciul SMTP al Google cu portul 587 pentru a trimite mesaje.

```
sender_password = "HERE"

sender_email = "sender@gmail.com"
recipient_email = "recipient@gmail.com"

message = MIMEMultipart()
message["From"] = sender_email
message["To"] = recipient_email
message["Subject"] = SUBJECT
message.attach(MIMEText(BODY, "plain"))

try:
    with smtplib.SMTP("smtp.gmail.com", 587) as server:
        server.starttls()
        server.login(sender_email, sender_password)
        server.sendmail([sender_email, recipient_email], message.as_string())
        print("Email sent successfully!")
except Exception as e:
    print(f"Failed to send email: {e}")
```

IV. Vizualizare si procesare de date

Pentru stocarea și vizualizarea datelor colectate, am utilizat **InfluxDB** și **Grafana**, două soluții populare pentru monitorizarea și analiza datelor în timp real.

1. Colectarea și Transmiterea Datelor

a. Dispozitiv IoT (ESP32/ESP8266)

- Datele sunt preluate de la senzori (umiditate sol și nivel apă)
- Fiecare citire este însoțită de un timestamp generat prin sincronizarea SNTP pentru a asigura o ordine cronologică precisă.
- Datele sunt formate în JSON și transmise prin MQTT către brokerul MQTT (Amazon IoT Core).

b. Serverul Flask

- Serverul primește datele prin intermediul unui broker MQTT (Amazon IoT Core).
- Aceste date sunt procesate și stocate într-o bază de date InfluxDB pentru persistență și analiză ulterioară.

2. Procesarea Datelor

a. Procesarea la Nivel de Server

Se creează serii temporale specifice în baza de date InfluxDB pentru:

- Umiditatea solului.
- Nivelul apei din rezervor.
- Starea pompei (activată/dezactivată)

b. Utilizarea Interogarilor

Detaliile de construcție a unui dashboard sunt următoarele:

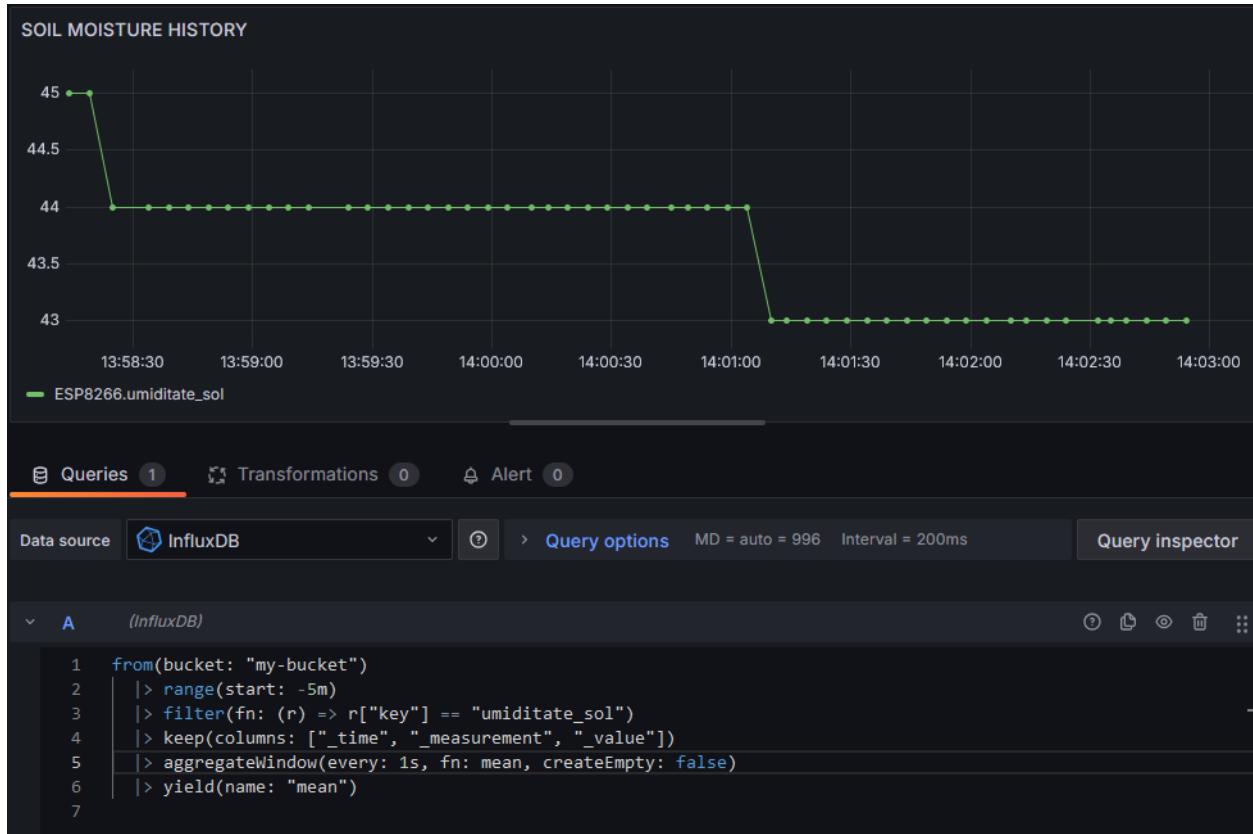
- Numele dashboard-ului
- Formatul numelor seriilor de timp: STATIE.METRICA (exemplu: ESP8266.nivel_apa);
- Interval de grupare a datelor: 1 secundă;
- Politică de grupare a datelor: medie aritmetică;
- Perioada de refresh automat al dashboard-ului: 5 secunde;

Exemplu de interogare influxDB:

```
from(bucket: "my-bucket")
  |> range(start: -5m)
  |> filter(fn: (r) => r["key"] == "umiditate_sol")
```

```
|> keep(columns: ["_time", "_measurement", "_value"])
|> aggregateWindow(every: 1s, fn: mean, createEmpty: false)
|> yield(name: "mean")
```

Se introduce in campul Queries al Grafana:

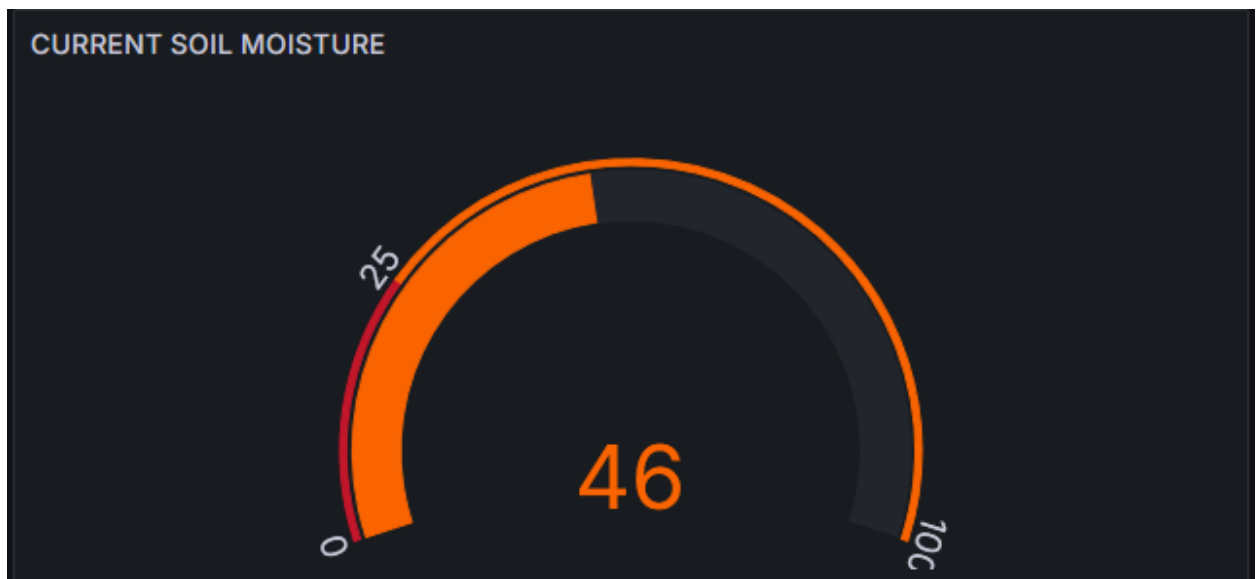
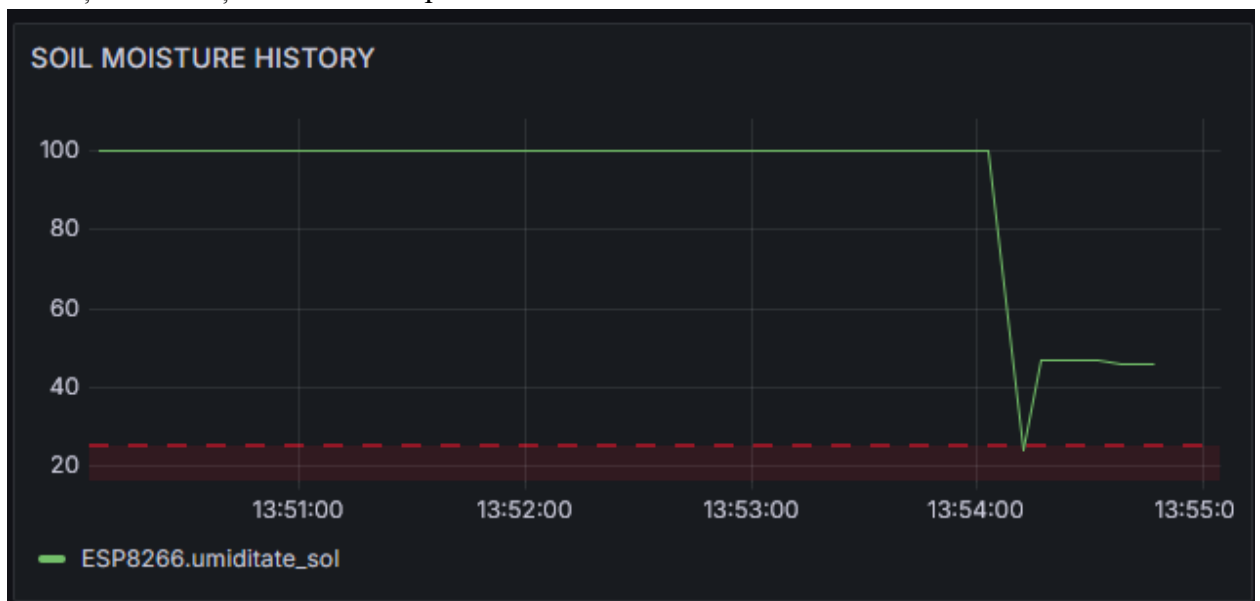


c. Integrarea cu Grafana

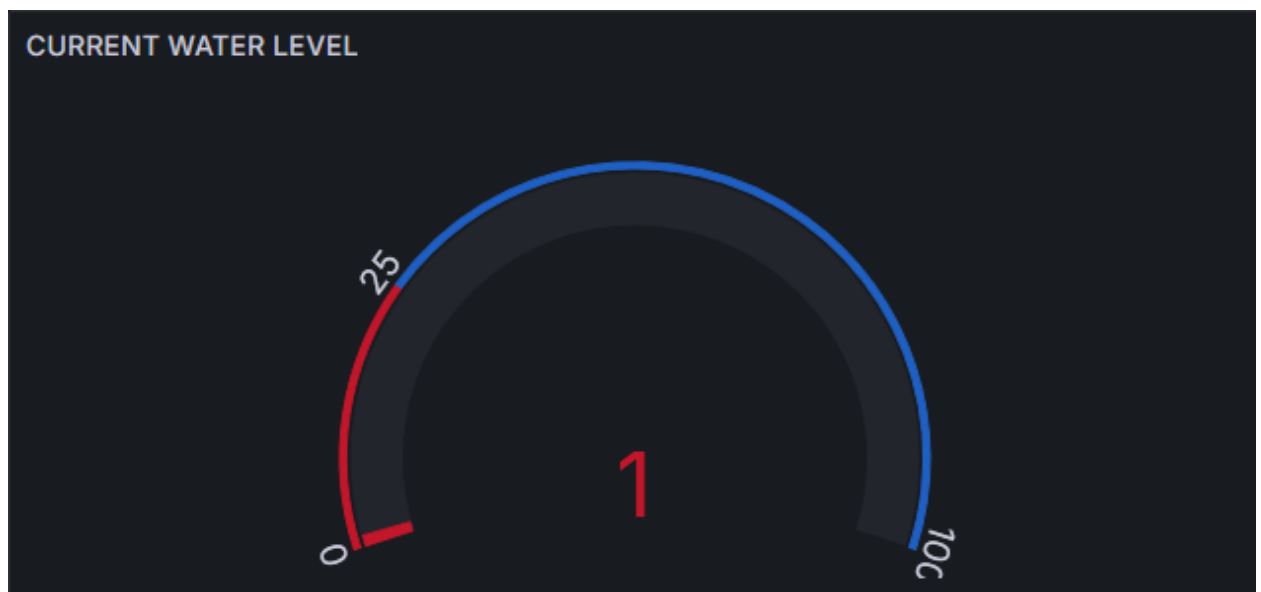
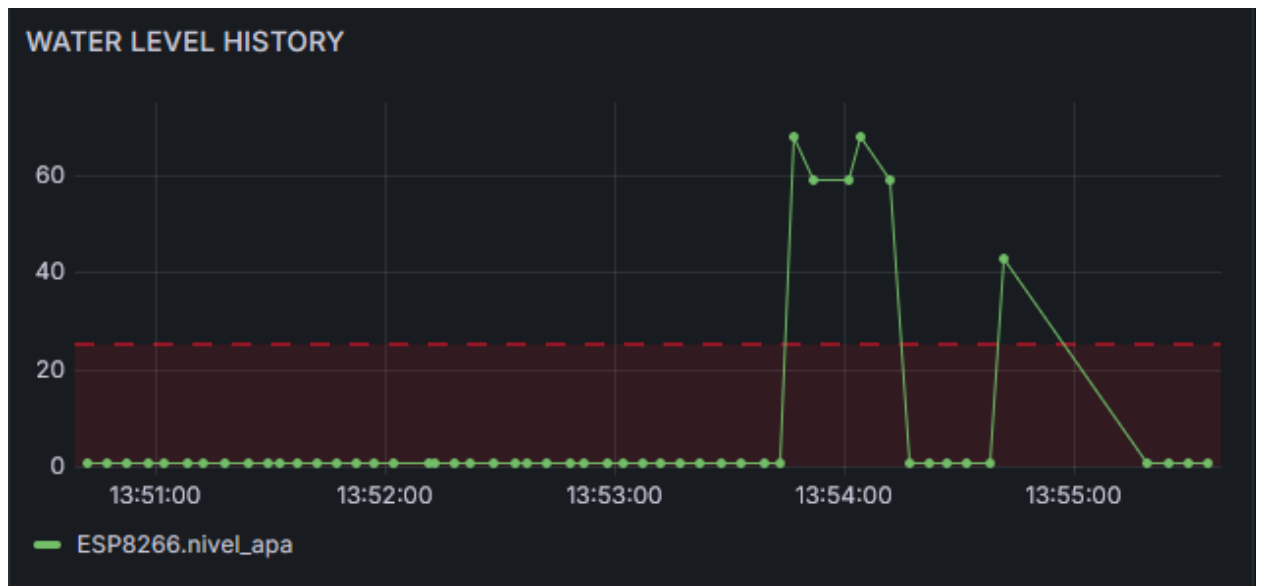
Grafana este utilizată pentru vizualizarea datelor în timp real și istorice.

Dashboard-urile sunt configurate pentru a afișa mai multe informații, printre care:

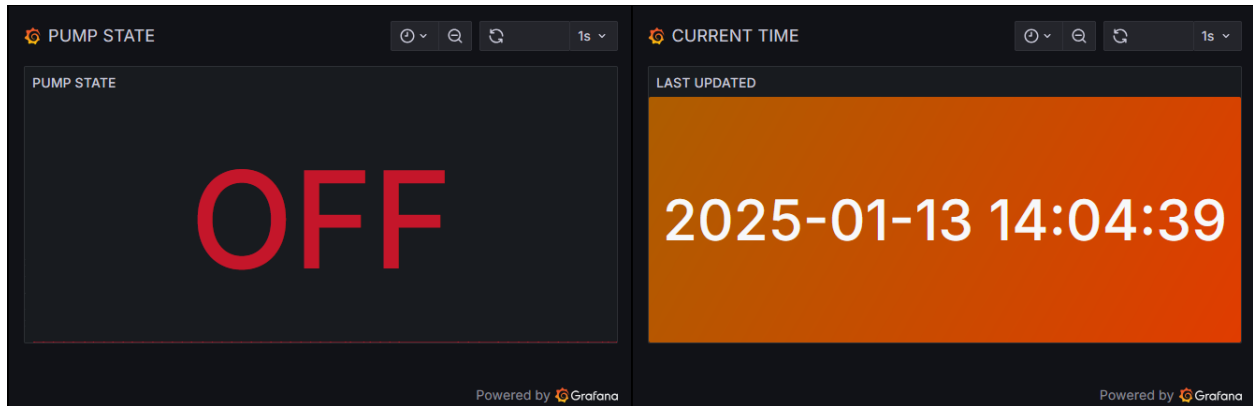
- Evoluția umidității solului în timp și nivelul curent al acestuia



- Evolutia nivelul apei din rezervor in timp si nivelul curent al acestuia



- Status-ul pompei (e pornita sau nu) si ultima actualizare:



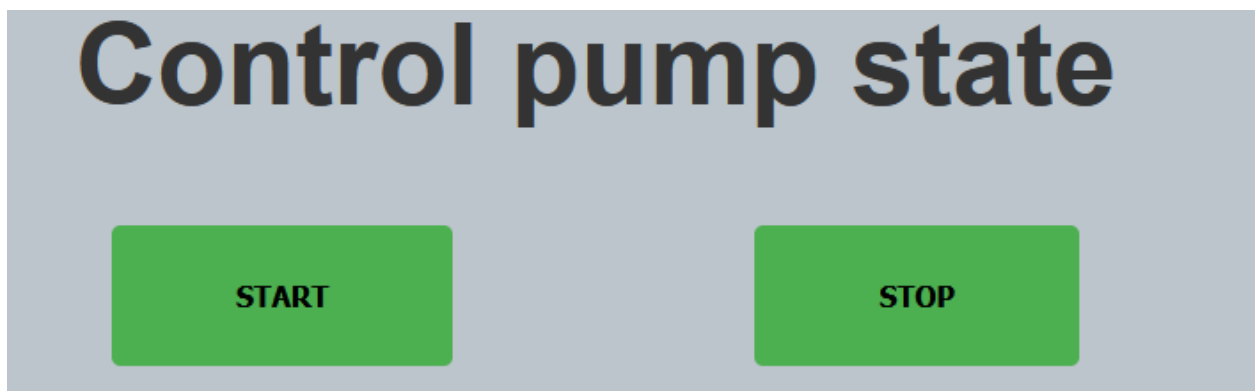
- Tabele cu alte statistici

	Avg	Current	Max	Min
	97.0	43	100	0

Station	Avg	Current	Max
ESP8266	1.17	1	68

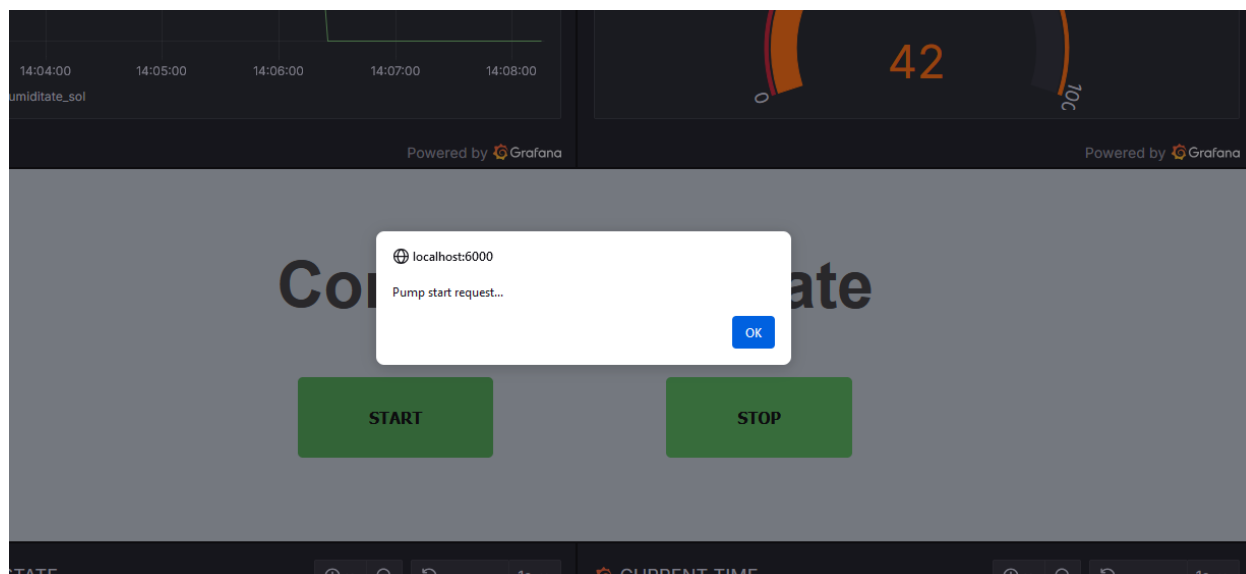
3. Activarea pompei de apa

Se realizează folosind butoanele de START/STOP din interfață:

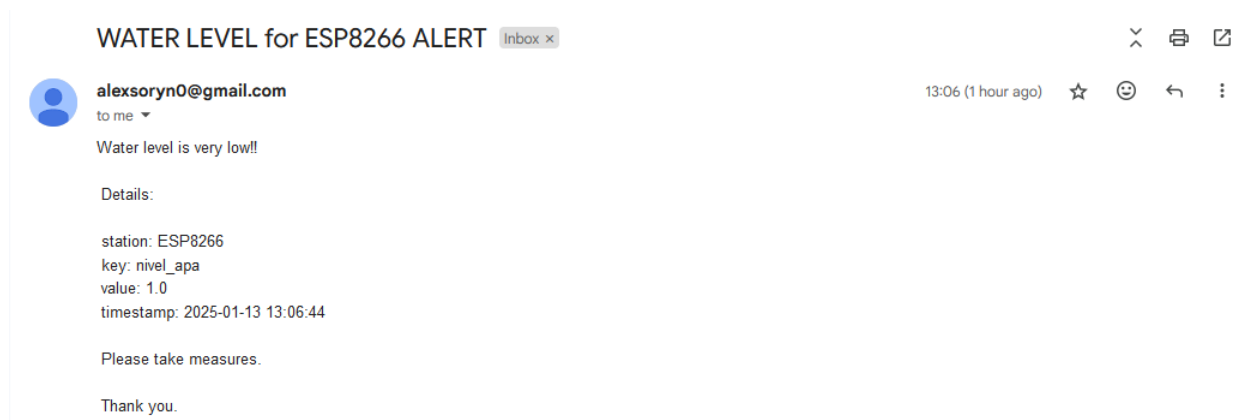


V. Funcționalități de Alertare

Se alertează utilizatorul în momentul când acționează pompa (se pornește / oprește).



În momentul când nivelurile trec sub un anumit prag, se notifica utilizatorul prin Gmail:



VI. Securitate

Comunicarea dintre dispozitive, serverul web și baza de date este securizată folosind criptare și autentificare, astfel încât să fie prevenite atacurile și accesul neautorizat.

1. Criptarea comunicării MQTT

a. Utilizarea WiFiClientSecure pentru Conexiuni Criptate

- Am utilizat biblioteca **WiFiClientSecure** pentru a realiza conexiuni criptate între dispozitivul IoT (ESP32/ESP8266) și brokerul MQTT (Amazon IoT Core).
- **WiFiClientSecure** implementează protocolul TLS (Transport Layer Security), asigurând criptarea end-to-end a datelor transmise.

b. Implementarea Certificatelor X.509

- Pentru a valida identitatea dispozitivelor și a serverelor, s-au utilizat **certIFICATE X.509**.
- Aceste certificate sunt esențiale pentru autentificarea bidirecțională între client și brokerul MQTT.

c. Certificatul CA:

- Este utilizat pentru a valida identitatea serverului MQTT.
- Asigură că dispozitivul se conectează la brokerul MQTT corect și nu la un server intermediar falsificat.

d. Certificatul clientului și cheia privată RSA:

- Sunt utilizate pentru a valida identitatea dispozitivului IoT.
- Acestea sunt generate și gestionate prin AWS IoT Core în timpul configurării dispozitivului („Thing”).

2. Asigurarea confidențialității în momentul autentificării Gmail

Se folosește o parola generată folosind parole pentru aplicații. Aceasta va fi folosită în cod.

← Parole pentru aplicații


Cu ajutorul parolelor pentru aplicații, te conectezi la Contul Google în aplicațiile și serviciile mai vechi, care nu acceptă standardele de securitate moderne.

Parolele pentru aplicații sunt mai puțin sigure decât folosirea aplicațiilor și a serviciilor actualizate care utilizează standarde de securitate moderne.

Înainte de a crea o parolă pentru aplicație, trebuie să verifici dacă aplicația are nevoie de aceasta pentru a te conecta.

[Află mai multe](#)

Parolele pentru aplicații

PYTHON	Creată pe 12:42, folosită ultima dată pe 13:45	
--------	--	---

Ca să creezi nouă parolă specifică aplicației, introdu un nume pentru aceasta mai jos.

Numele aplicației

3. Criptarea conexiunii cu serverul de Gmail

Se folosește starttls pentru a securiza conexiunea:

```
try:
    with smtplib.SMTP("smtp.gmail.com", 587) as server:
        server.starttls()
```

Pornirea aplicației

Pornirea aplicației se face folosind scriptul `run.sh`, care conține comenzile:

- **`sudo docker compose -f docker-compose.yml up --build -d`** -> crearea imaginilor de docker ale bazei de date și aplicației de vizualizare
- **`python3 main.py`** -> pornirea serverului
- accesarea portului **6000** pentru aplicația principală
- InfluxDB rulează pe portul **8086**, iar Grafana pe **3000**