# Sistem de udare automatizată a plantelor

Cuprins

[I. Introducere 2](#_Toc188127344)

[Obiective 2](#_Toc188127345)

[Actuatori 2](#_Toc188127346)

[Senzori 2](#_Toc188127347)

[II. Arhitectura 3](#_Toc188127348)

[Componente hardware 3](#_Toc188127349)

[Software 5](#_Toc188127350)

[Schema electrica 6](#_Toc188127351)

[Topologie 7](#_Toc188127352)

[III. Implementare 8](#_Toc188127353)

[Structura Aplicației 8](#_Toc188127354)

[Configurarea hardware-ului 8](#_Toc188127355)

[Configurarea software-ului 8](#_Toc188127356)

[Configurarea sistemului de notificare 14](#_Toc188127357)

[IV. Vizualizare si procesare de date 15](#_Toc188127358)

[1. Colectarea și Transmiterea Datelor 15](#_Toc188127359)

[2. Procesarea Datelor 15](#_Toc188127360)

[3. Activarea pompei de apa 19](#_Toc188127361)

[V. Funcționalități de Alertare 20](#_Toc188127362)

[VI. Securitate 21](#_Toc188127363)

[1. Criptarea comunicării MQTT 21](#_Toc188127364)

[2. Asigurarea confidențialității in momentul autentificării Gmail 21](#_Toc188127365)

[3. Criptarea conexiunii cu serverul de Gmail 22](#_Toc188127366)

[VII. Provocări și soluții 23](#_Toc188127367)

[Conectarea la Amazon Iot Core 23](#_Toc188127368)

[Convertirea datelor de la senzorul de distanță într-o distanță corectă 23](#_Toc188127369)

[Vizualizarea link-urilor embedded Grafana pe serverul web 23](#_Toc188127370)

[Micșorarea ratei de refresh de 5 secunde la o secundă 23](#_Toc188127371)

[Pornirea aplicației 24](#_Toc188127372)

[Referințe 24](#_Toc188127373)

# 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. Actuatoripompa de apa Aceasta va fi activata de clientul final care acceseaza interfata web si decide pe baza informatiilor daca porneste sau opreste pompa Senzorinivel 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. ArhitecturaComponente hardware  * **Senzor de umiditate a solului**: Pentru a detecta nivelul de umiditate din sol; daca scade sub un anumit nivel, microcontroller-ul trimite o notificare utilizatorului, si el va decide dacă trebuie udate plantele   Ground Humidity Sensor Module  <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 apa către sol   Mini Pompă de Apă Submersibilă  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   Modul  releu  cu un canal (comandat cu 5 V)  <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   Senzor Ultrasonic de Distanță HC-SR04+  (Compatibil 3.3 V și 5 V)  <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.   Placa de dezvoltare WiFi cu ESP8266  <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 si pentru conectarea la Cloud:   * ESP8266WiFi.h * WiFiClientSecure.h * PubSubClient.h * ArduinoJson.h   **2. Broker MQTT:** Amazon IOT Core: Cloud-ul in care se stocheaza 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** si **InfluxDB** folosind **Docker:** pentru stocarea si vizualizarea datelor Schema electrica |
|  |  |

O imagine care conține text, captură de ecran, diagramă, proiectare

Descriere generată automat

### Topologie

O imagine care conține text, captură de ecran, diagramă, Dreptunghi

Descriere generată automat

**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 configurarile Grafana
* **docker-compose.yml ->** fișierul yml pentru pornirea micro-serviciilor care rulează in backend
* **grafana.ini ->** alt fișier de configurare pentru Grafana
* **run.sh** si **stop.sh ->** fișiere de pornire si deschidere a întregii aplicații

### Configurarea hardware-ului

Am folosit schema electrica pentru a realiza conexiunile intre 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

1. **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 comunicatie 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

1. **PubSubClient**:

* Folosită pentru comunicarea MQTT între dispozitiv și brokerul Amazon IoT Core.
* Gestionarea publicării și subscrierii la topicurile definite pentru transmiterea și primirea datelor.

1. **NTPClient**:

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

1. **ArduinoJson**:

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

1. **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, in stadiul actual este folosita pentru Debug.

* **Integrarea cu Grafana**:

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

1. **InfluxDB si 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:**

O imagine care conține text, Font, captură de ecran

Descriere generată automat

O imagine care conține text, captură de ecran, Font

Descriere generată automat

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

O imagine care conține text, captură de ecran, Font

Descriere generată automat

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

O imagine care conține text, Font, captură de ecran, Grafică

Descriere generată automat

**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 si 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 creand un nou „Thing” si un nou „Certificat”:

O imagine care conține text, Font, număr, software

Descriere generată automat

O imagine care conține text, captură de ecran, software, număr

Descriere generată automat

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

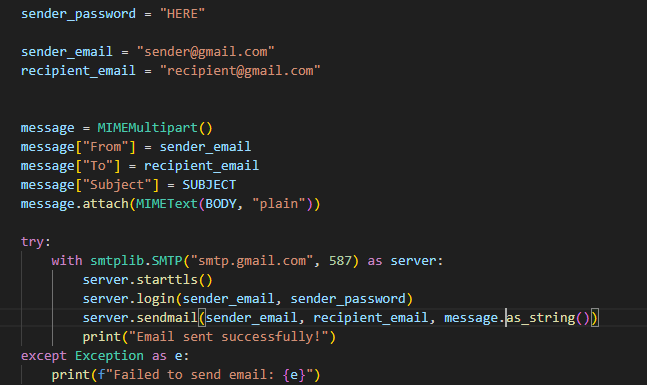
* web\_server/command -> se controleaza pompa printr-o comanda a utilizatorului

### Configurarea sistemului de notificare

Se generează o parola 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.



# 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.

### 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 formatate 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.

### 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 constructie 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;  
• Politica 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:

O imagine care conține text, captură de ecran, software, Software multimedia

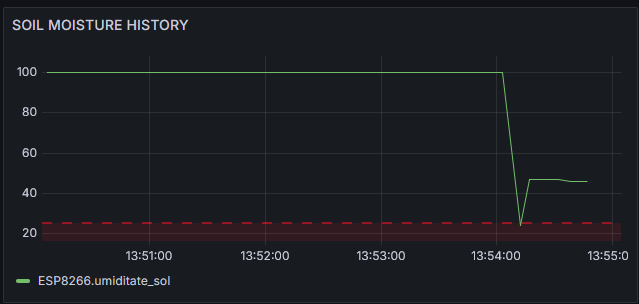
Descriere generată automat

**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 si nivelul curent al acestuia



O imagine care conține captură de ecran, Grafică, design grafic, cerc

Descriere generată automat

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

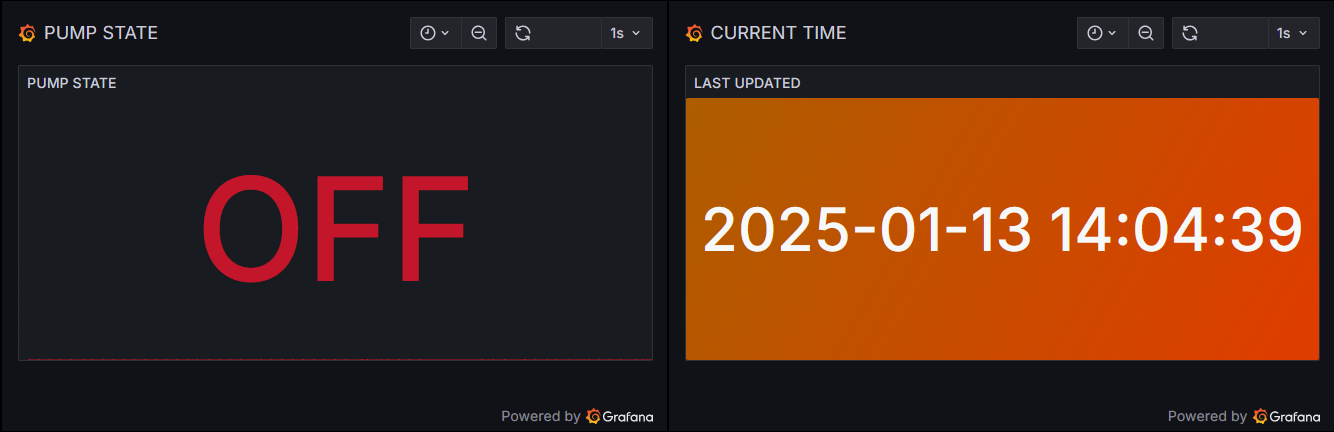
O imagine care conține captură de ecran, linie, Interval, text

Descriere generată automat

O imagine care conține captură de ecran, cerc, Grafică, diagramă

Descriere generată automat

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



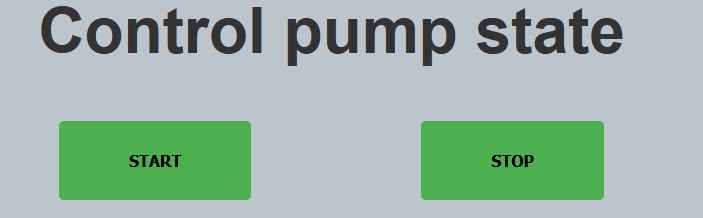
* Tabele cu alte statistici

O imagine care conține captură de ecran, Software multimedia

Descriere generată automat

### Activarea pompei de apa

Se realizează folosind butoanele de START/STOP din interfață:



# V. Funcționalități de Alertare

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

O imagine care conține text, captură de ecran, Software multimedia, software

Descriere generată automat

In momentul cand nivelurile trec sub un anumit prag, se notifica utilizatorul prin Gmail:

O imagine care conține text, Font, captură de ecran

Descriere generată automat

# 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**:

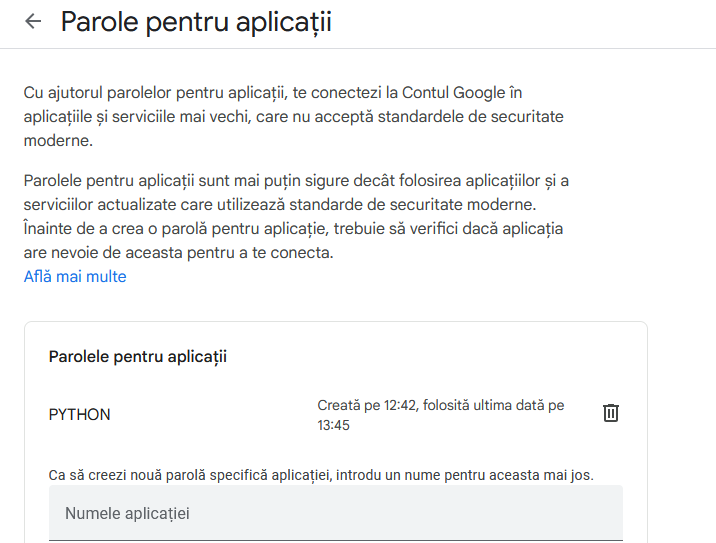
* Este utilizat pentru a valida identitatea serverului MQTT.
* Asigură că dispozitivul se conectează la brokerul MQTT corect și nu la un server intermediar falsificat.

**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”).

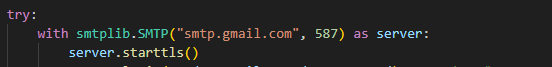
### 2. Asigurarea confidențialității in momentul autentificării Gmail

Se folosește o parola generata folosind parole pentru aplicații. Aceasta va fi folosita in cod.



### 3. Criptarea conexiunii cu serverul de Gmail

Se foloseste starttls pentru a securiza conexiunea:



# VII. Provocări și soluții

## Conectarea la Amazon Iot Core

* Se creează certificate și politici de acces din interfața web. Pentru acest lucru am vizionat un tutorial (1).

## Convertirea datelor de la senzorul de distanță într-o distanță corectă

* Se adaugă codul:

 // Clears the trigPin

    digitalWrite(trigPin, LOW);

    delayMicroseconds(2);

    // Sets the trigPin on HIGH state for 10 micro seconds

    digitalWrite(trigPin, HIGH);

    delayMicroseconds(10);

    digitalWrite(trigPin, LOW);

    // Reads the echoPin, returns the sound wave travel time in microseconds

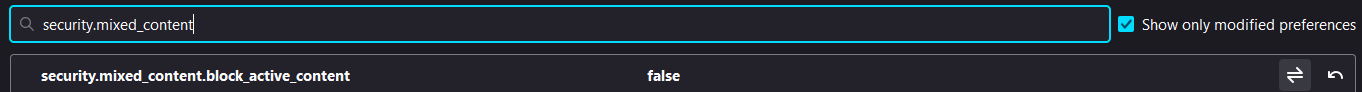
    duration = pulseIn(echoPin, HIGH);

    // Calculating the distance

    distance = duration \* 0.034 / 2;

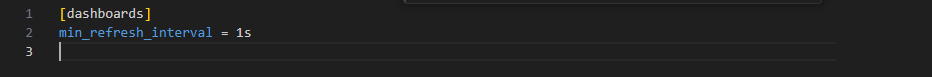
## Vizualizarea link-urilor embedded Grafana pe serverul web

Se activează această opțiune pe browserul web:



## Micșorarea ratei de refresh de 5 secunde la o secundă

Am creat fișierul grafana.ini in care am pus codul:



apoi am schimbat pentru fiecare dashboard rata la o secundă.

# Pornirea aplicației

Pornirea aplicatiei se face folosind scriptul run.sh, care contine comenzile:

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

# Referințe

(1) <https://youtu.be/xZoeJ-osS3g?si=Lw8lMLNcilYTrPPt>

– How to Connect Esp8266 to aws IoT Core - (2024)

(2) <https://github.com/aws/aws-iot-device-sdk-python-v2/blob/main/samples/pubsub.py>

- AWS SDK Python

(3) <https://community.grafana.com/t/how-to-change-refresh-rate-from-5s-to-1s/39008>

- Rata de refresh de la 5 la 1 secundă

# Linkuri

## Youtube:

## Github: https://github.com/sorin0popa/Proiect\_PR.git