Sistem de agricultura inteligent cu monitorizarea solului si vremii

Lățea Mihai Alexandru – 341C1

Cuprins

I. Introducere 3
II. Arhitectura 3
III. Implementare 4
IV. Vizualizare de date 12
V. Securitate 14
VI. Provocari si Solutii 15

I. Introducere

Proiectul constă în realizarea unui sistem de agricultură inteligent care să monitorizeze solul și vremea. Sistemul va fi capabil sa monitorizeze temperatura, umiditatea aerului si a solului, luminozitatea, daca ploua sau nu si va putea sa controleze irigarea solului.

Datele vor putea fi vizualizate de catre un utilizator de la distanta, prin intermediul unei interfete web (Grafana). In functie de anumite valori primite de la senzorii de umiditate a solului si ploaie, serverul poate activa sau dezactiva pompele de irigare, in funtie de situatie. De asemenea, utilizatorul poate trimite manual comenzi in cazul in care este nevoie interventia umana.

II. Arhitectura

Microcontroller-ele folosite sunt Raspberry Pi Pico W. Acestea fac parte din 2 tipuri de dispozitive: colectoare si cele de control. Dispozitivele colectoare vor prelua datele de la senzori si le vor trimite mai departe catre server. In functie de aceste date sau de interactiunea utilizatorului, serverul va trimite comenzi catre dispozitivele de control care vor porni/opri relee, motoare, etc.

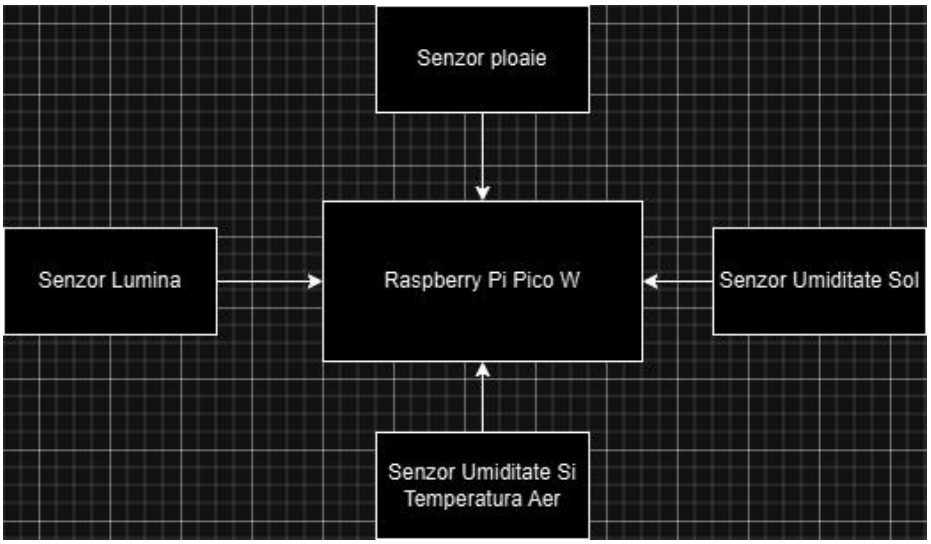


Figure 1 Diagrama dispozitiv colector

Din punct de vedere software, dispozitivele vor transmite date folosind protocolul MQTT catre server. Pentru interactiunea utilizatorului din interfata cu serverul, se va folosi un Rest API. Datele vor si salvate intr-o baza de date pentru 30 de zile si afisate folosind Grafana.

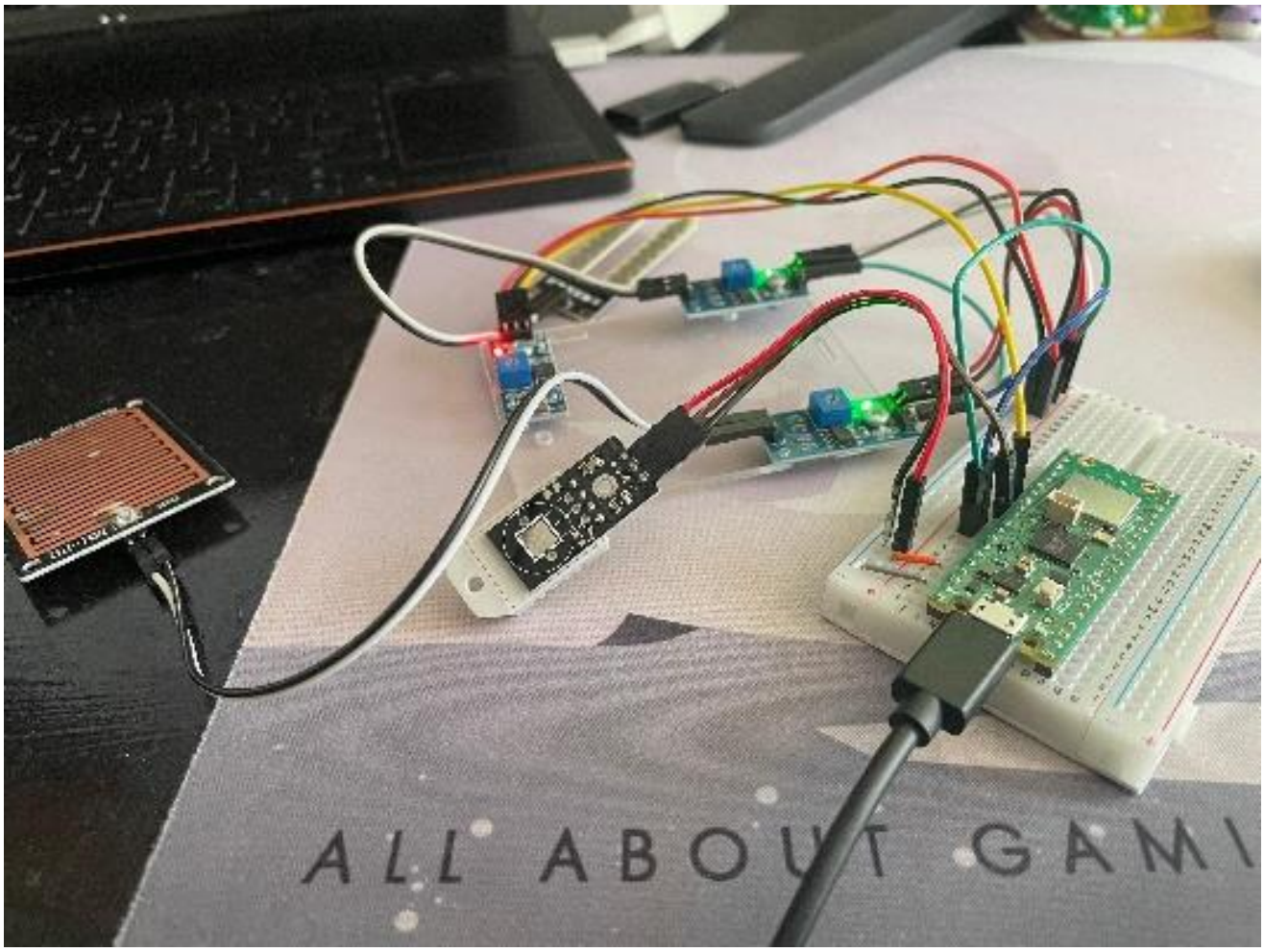
III. Implementare

Hardware

Am folosit placute Raspberry Pi W la care am conectat senzorii:

- Senzorul de umiditate si temperatura - pin 22;
- Senzorul de ploaie - pin 26;
- Senzorul de umiditate a solului – pin 27;
- Senzorul de lumina – pin 21;
- Releul pentru activarea pompei de apa – pin 16 (pe o placuta diferita de cea cu senzorii).

Circuitul fizic



Software

Pentru placuta de colectare de date este nevoie de biblioteca mqtt (simple.py), biblioteca pentru senzorul de umiditate si temperatura DHT22 (PicoDHT22.py).

Codul se afla in /data-producer/main.py. Placuta incearca intai sa se conecteze la wifi prin functia ‘wifi_connect()’ iar apoi la serverul mqtt (mqtt_connect()). Aceasta citeste datele de la senzori, prin functii dedicate: check_light_status(), check_soil_status() si check_rain_status(). Apoi, datele sunt trimise la serverul mqtt pe topicuri diferite pentru fiecare senzor: /temperature, /humidity, /rain, etc. Acest lucru se intampla pentru a fi mai usor de vizualizat datele in platforma web.

Excemplu cod

```python
def wifi_connect():

    ssid = ""
    password = ""

    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)

    mac =  wlan.config('mac')
    global client_id
    client_id = "pico-" + "".join("{:02x}".format(x) for x in mac)

    wlan.connect(ssid, password)

    while not wlan.isconnected():
        time.sleep(1)
        print("Connecting to WiFi...")

    print("Connected to WiFi")
```

```python
def mqtt_connect():
    mqtt_server = "35.239.200.114"

    client = mqtt.MQTTClient(client_id, mqtt_server)
    print("Client ID:", client_id)
    client.connect()
    print("Connected to MQTT")
    return client
```

```python
def check_rain_status():
    raw_value = rain_sensor.read_u16() // 64

    if 0 <= raw_value < 256:
        return "Ploaie puternica"
    elif 256 <= raw_value < 512:
        return "Ploaie"
    elif 512 <= raw_value < 768:
        return "Ploaie inceata"
    elif 768 <= raw_value <= 1023:
        return "Nu ploua"
    else:
        return "Valoare necunoscuta"
```

Pentru sistemul de alertare am folosit un bot de Telegram. Placuta care colecteaza date trimite informatii la interval de o ora (pentru a nu incarca inbox-ul).

Pentru placuta care se ocupa de actuatoare, trebuie importata doar biblioteca mqtt(simple.py).

Codul sursa poate fi gasit in /action-producer/main.py. Aceasta se conecteaza la serverul mqtt, la topicul ‘pump’ de unde va primi mesaje de pornire sau oprire a pompei de apa.

```
def mqtt_callback(topic, msg):
    msg = msg.decode()
    topic = topic.decode()

    if (topic == "pump"):
        if (msg == "ON"):
            pump_motor.value(1)
        elif (msg == "OFF"):
            pump_motor.value(0)
```

Proiectul dispune si de un server REST, realizat folosind Flask. Api-ul este folosit in platforma web pentru a activa sau dezactiva manual pompa de apa si de a vedea statusul acesteia. Sunt expuse 3 endpoint-uri,

/api/pump/activate – activeaza pompa;

/api/pump/deactivate – dezactiveaza pompa;

/api/pump/status – verifica daca pompa este pornita sau oprita.

```
@app.route('/api/pump/activate', methods=['GET'])
def activatePump():
    global pump_activated

    if not pump_activated:
        mqtt_client.publish(PUMP_TOPIC, "ON")
        pump_activated = True
    else:
        response = jsonify({"message": "Pump already activated!"})
        response.headers.add('Access-Control-Allow-Origin', '*')
        return response

    response = jsonify({"message": "Pump activated!"})
    response.headers.add('Access-Control-Allow-Origin', '*')
    return response
```

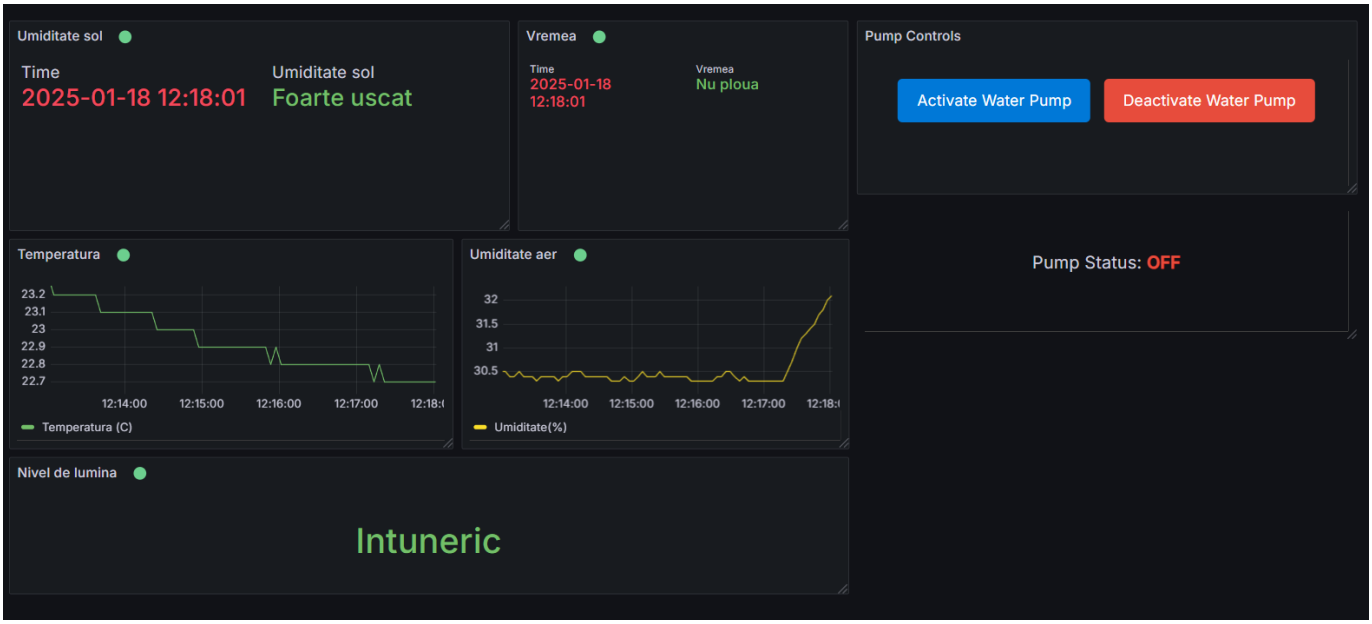
Cloud

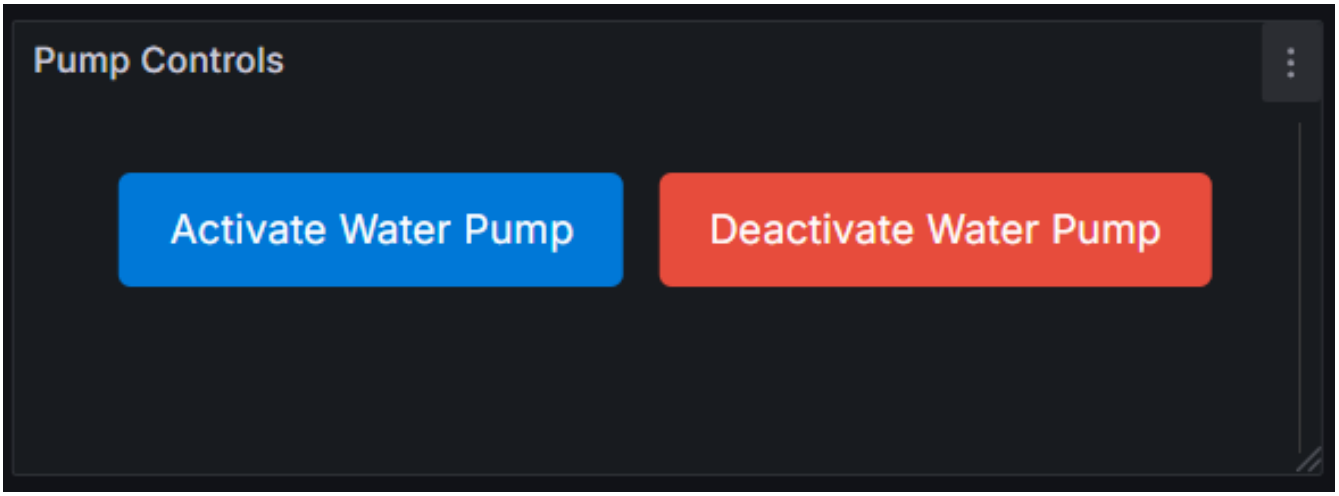
Atat serverul Grafana cat si backendul in Flask si serverul MQTT sunt lansate in Cloud, folosind serviicile celor de la Google Cloud. O masina cu o distributie Ubuntu contine toate servele pe porturi diferite:

- Grafana: https://35.239.200.114:3000/
- Flask: https://35.239.200.114/api/
- MQTT: tcp://35.239.200.114:1883

IV. Vizualizare de date

Ca tool de vizualizare a datelor, a fost folosit grafana cu plugin-ul de MQTT pentru conectarea la server si prelucrarea datelor.





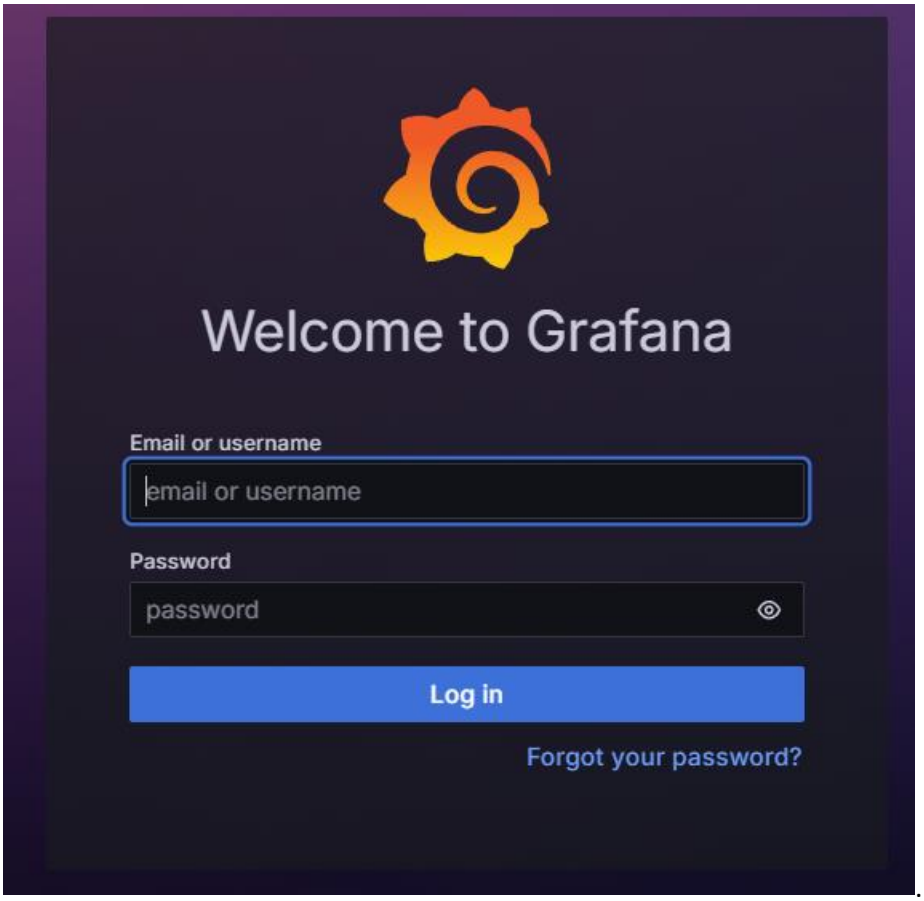
Cele 2 butoane trimit request catre api pentru a porni sau opri pompa de apa.

Fiecare cadran de vizualizare este abonat la un topic pentru senzorul specific. Panoul cu butoane este un panou simplu, te tip text, iar butoanele sunt create de la 0 folosind html si javascript. La fel este create si panoul pentru statusul pompei, acesta interogheaza la fiecare secunda API-ul pentru a vedea daca s-a schimbat starea in care se afla pompa.

V. Securitate

Atat serverul de grafana cat si cel backend transmit datele prin https, folosind un certificat SSL. Acest certificat este self signed, iar de aceea, browserele vor continua sa afiseze conexiunea ca fiind nesigura, chiar daca pachetele sunt criptate.

Accesul la serverul grafana se realizeaza prin autentificare cu user si parola. Asa ca, desi server-ul este public si poate fi accesat de oricine, doar cei cu credentiale pot accesa datele.



VI. Provocari si Solutii

Provocarile au venit din toate partile: atat hardware, software cat si datorita tehnologiilor noi.

Hardware

Din punct de vedere hardware, cea mai mai provocare a fost calitatea proasta a pieselor. Placute care la a doua rulare a codului nu se mai conectau la wifi, se conectau la wifi dar nu se mai contectau la serverul mqtt si multe altele. Contacte imperfecte, fire defecte... Singura solutie a fost rabdarea deoarece avand un numar limitat de piese la dispozitie, a trebuit sa ma folosesc de acestea.

Placutele nu prea faceau fata sa trmita date la mqtt, sa proceseze requesturi de la api si sa trimita mesaje catre botul de telegram. Am incercat sa limitez cat mai mult interactiunea cu internetul, pastrand doar functionalitatile esentiale

Software

Pe partea software, a fost o provocare sa ma obisnuiesc cu Grafana. Desi este un tool popular, pare ca are un User Experience destul de primitiv. Plugin-urile sunt sarace si a trebuit sa implementez diferte hack-uri pentru a ajunge la functionalitatea dorita.

Hostarea in cloud a reprezentat un proces greoi deoarece nu greu se gaseste documentatie sau tutoriale pentru providerii de astfel de servicii. Am ajuns sa folosesc google cloud pentru ca parut cea mai rapida solutie (avand putina experienta cu AWS si configurarea masinilor in cloud) ++ era varianta gratis.

A durat ceva pana sa realizez ca firewall-ul implicit generat de google cloud bloca porturile pentru servere, fiind nevoit sa creez reguli noi pentru a putea accesa interfata web sau serverul api.

O alta provocare a fost securizarea serverelor fara un domeniu. Acest lucru l-am realizat printr-un certificat self signed. Dorinta mea era de a scapa de atentionarea din browser a faptului ca nu este sigura conexiunea. Oricand de mult am incercat sa import pe browserul local certificatul creat de mine, acesta tot nu era sigur. Conexiunea a ramas securizata, dar si atentionarea prezenta.

Resurse

https://ocw.cs.pub.ro/courses/priot/laboratoare/04

https://ocw.cs.pub.ro/courses/priot/laboratoare/05

https://ocw.cs.pub.ro/courses/priot/laboratoare/06

https://ocw.cs.pub.ro/courses/priot/laboratoare/08

https://grafana.com/tutorials/

https://flask.palletsprojects.com/en/stable/

https://www.youtube.com/watch?v=VSktsnRoDsc

https://community.grafana.com/t/basic-button-panel/124536