Projekts Nr. VPP-COVID-2020/1-0025

Jaunās tehnoloģijas COVID-19 pacientu tēmētai monitorēšani, testēšanai un terapijai (3-T Project)

Projekta papildu (specifiskie) rezultāti Nr.17

Sensoru datu bāze

Tehnoloģijas apraksts

6. pielikums

Autors:

Kalvis Sūniņš-Biedris

Rīga, 2020

SATURS

APZĪMĒJUMI

AWS – *Amazon Web Services*, Amazon mākoņdatošanas serviss.

MQTT - *Message Queuing Telemetry Transport*, publicēšanas-abonēšanas tīkla protokols.

NoSQL - *Non-relational database*, ne-relāciju datubāze.

PEM - *Privacy-Enhanced Mail*, "de facto" faila formāts.

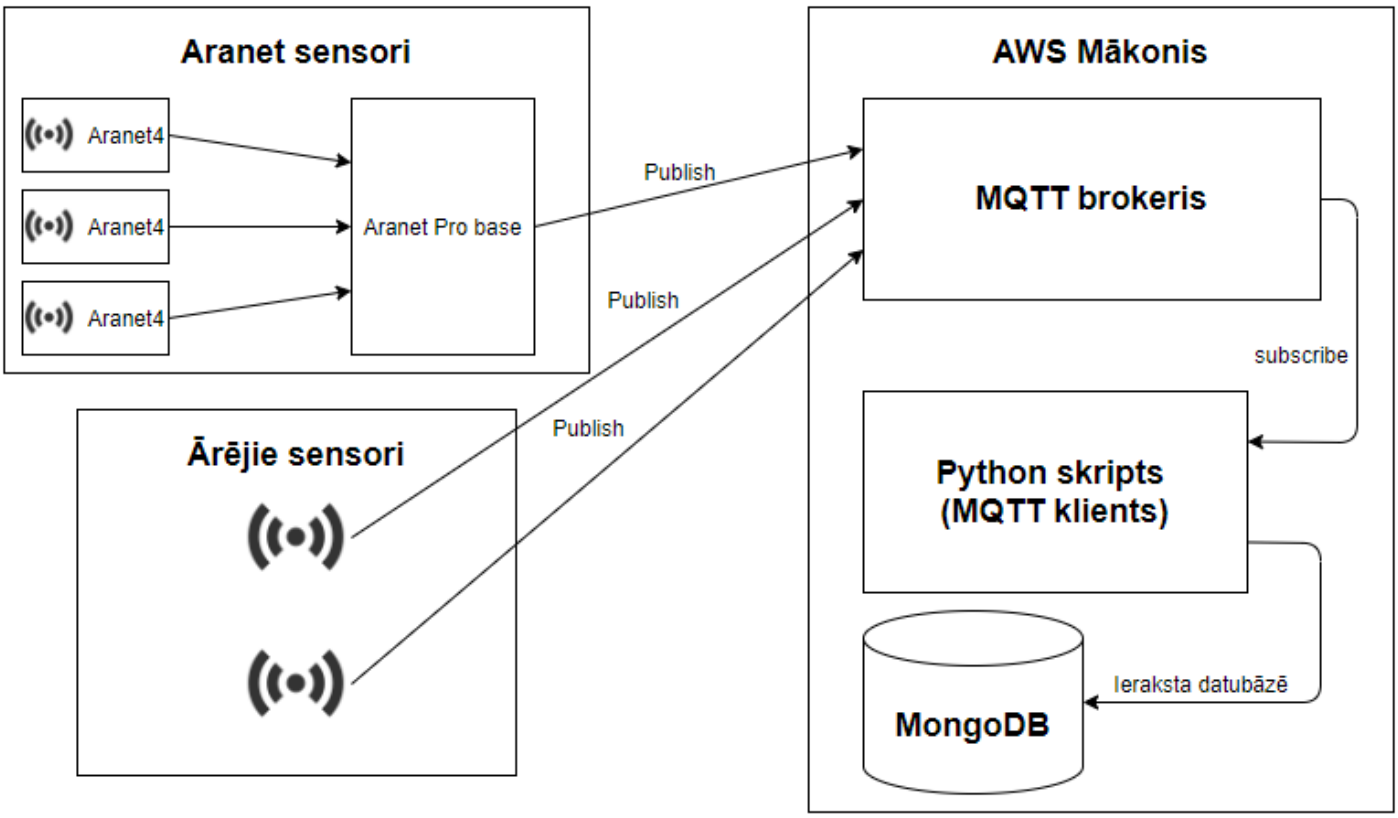
IoT - *Internet of things*, lietu internets.

GUI - *Graphical user interface*, grafiska lietotāja saskarne.

IEVADS

Projektā "Jaunās tehnoloģijas Covid-19 pacientu tēmētai monitorēšani, testēšanai un terapijai (3-T Project)" [1] 1-T darba grupā tiek veikti mērījumi ar sensoru un virtuālu sensoru palīdzību. Šiem sensoriem ir nepieciešams saglabāts datus tālākai apstrādei un nodrošināt komunikāciju starp sensoriem, kā arī nodrošināt komunikāciju ar datubāzi, kur dati tiek saglabāti. Šajā dokumentā tiks aprakstīts risinājums iepriekš minētajai problēmai.

Projektā tiek izmantoti Aranet4 [7] sensori, kas mēra gaisa kvalitāti istabā, un ārējie sensori, kas arī mēra gaisa kvalitāti istabā, kā arī cilvēku skaitu un cilvēka inficēšanās riska varbūtību. Šos datus ir nepieciešams saglabāt, tāpēc tiek izmantota MongoDB datubāze. MongoDB ir NoSQL datubāze, kura iedalās kolekcijās, kurās atrodas JSON tipa dokumenti. Lai šos datus nosūtītu uz datubāzi, tiek izmantots MQTT brokeris, kas saņem ziņas no publicētājiem (sensori) un nosūta abonētājiem (skripti), kas ieraksta datus datubāzē.



AWS MĀKONIS

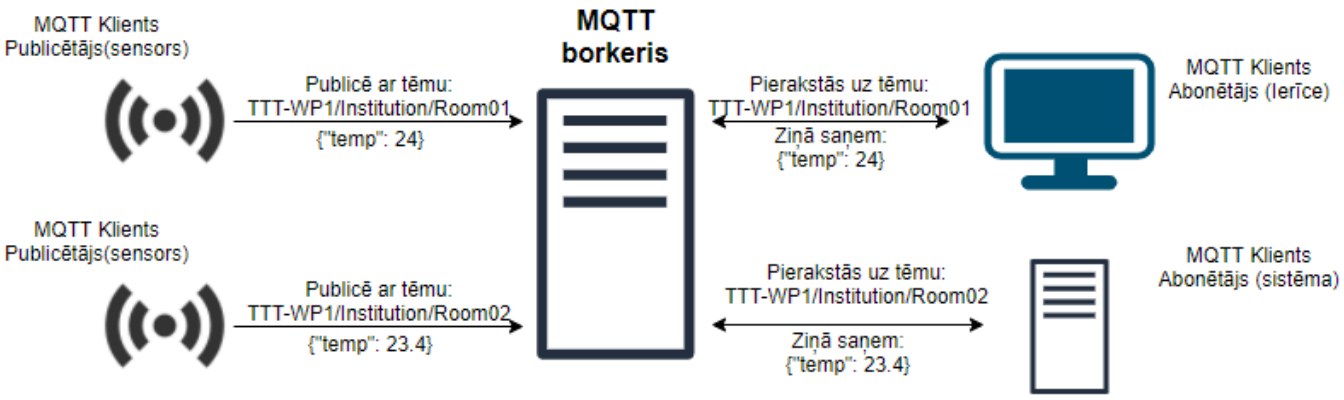
Projektā tiek izmantots Amazon web servisi, lai nodrošinātu ilgtspējīgu un drošu mākoņdatošanu. Uz šī AWS mākoņa jeb AWS instances atrodas Linux Ubuntu serveris, kas ir atvērtā koda Debian bāzes Linux distribūcija.

Serveris klausās 2 portus, kur 27017 ports tiek izmantot datubāzei un 8883 ports tiek izmantots MQTT brokerim. Uz šī servera atrodas 2 galvenās daļas, kas nodrošina sensoru datu ierakstīšanu datubāzē, komunikāciju starp sensoriem, kā arī pati datubāze atrodas uz šī servera.

1. MQTT protokols

Projektā komunikācija ar datubāzi, tai skaitā saziņa starp citām ierīcēm, tiek realizēta ar MQTT protokolu. Šis protokols darbojas pēc publish/subscribe modeļa, kur ir publicētājs (Publisher), kas var būt sensors, dators vai citas elekroniskas ierīces un ir abonētājs (Subscriber) ierīce, kas parakstās uz kādu noteiktu tēmu (Topic). Par komunikāciju starp publicētāju un abonētāju rūpējas MQTT brokeris (Broker), tas parūpēsies, lai abonētājs iegūtu tikai tos datu, uz kuriem tas ir parakstījies.

Projektā MQTT Brokeris atrodas uz AWS mākoņa, tādejādi, ja klientiem (publicētāji/abonētāji) ir interneta pieslēgums, tie var sūtīt datus uz MQTT brokeri.



1.1. Brokeris

Šajā projektā tiek izmantots *Eclipse Mosquitto* [2] atvērtā koda brokeris, kas izmanto MQTT protokola versiju 3.1.1. [3] Brokeris ziņas starp abonētāju un publicētāju šifrē, tādejādi nodrošinot drošu komunikāciju. Brokeris izmanto TLS (Transport Layer Security) [4] šifrēšanas protokolu, kas izmanto paša parakstītu sertifikātu. Mosquitto MQTT vairs neatbalsta TLS 1.1 versiju, tāpēc tiek izmantotas TLS 1.2 un 1.3 versijas.

Lai izmantotu projekta MQTT brokeri ir nepieciešams lietotājvārds un parole, kā arī PEM fails, kas darbojas kā atslēga, ar kuras palīdzību brokeris sapratīs, ka komunicēt vēlas šifrējot ziņas. Ja netiek uzrādīts PEM fails, tad MQTT brokeris ziņas ignorē. MQTT brokerim iespējam pieslēgties norādot AWS instances publisko DNS adresi un 8883 portu. MQTT brokera konfigurācijas fails pieejams pielikumos.

1.2. Klienti

Gan publicētāji, gan abonētāji ir MQTT klienti, vienīgā atšķirības starp tiem ir, vai dotais klients pašlaik publicē vai ir parakstījies uz datu saņemšanu (publicēšanas un abonēšanas funkcionalitāti var ieviest arī tajā pašā MQTT klientā).

1.3. Tēmas (topic)

Tēmas ir teksta virkne (string), kur katrs līmenis tiek atdalīts ar slīpsvītru ("/"), piemēram, RootTopic/2ndLevel/3rdLevel. Aranet sensoru datu saglabāšanai tiek izmantos "VPP-Aranet" tēmas pirmais līmenis, kuru var uzstādīt Aranet Pro Base. Aranet sensoru datu tēmas piemērs:

"VPP-Aranet/BāzesID/sensors/sensoraID/json/measurements"

Protams, ar šādu topic datus var sūtīt vairāki sensori un lai nevajadzētu definēt katra sensora un bāzes stacijas ID var izmantot wildcards, kas palīdz apkopot vairākus topic, piemēram:

a. "VPP-Aranet/BāzesID/sensors/+/json/measurements"

"+" simbols – viena līmeņa aizstājējzīme, kas šajā gadījumā abonēs visiem sensoriem pie noteiktās bāzes stacijas

b. "VPP-Aranet/#"

"#" simbols – vairāku līmeņu aizstājējzīme, kas šajā gadījumā abonēs visā Aranet bāzes stacijām un sensoriem.

2. Python skripti

Pytnon skripti ir klienti (abonētāji), kas klausās Aranet sensoru un ārējo sensoru sūtītos datus un saglabā tos datubāzē. Tiek pieņemts, ka ziņas, kuras ir jāsaglabā datubāzē tiek sūtītas json formātā, ja netiek ievērots šis stils, tad datubāzē netiek saglabāti dati.

Tā kā brokeris tiek izmantots ne tikai komunikācijai starp datubāzi, bet ari komunikācijai starp citām ierīcēm, tad šajā projektā ar tēmas pirmo līmeni (Root Topic) var paziņot, ka šos datus grib saglabāt datubāzē. Attiecīgi norādot TTT-WP1 (Ārējo sonsoru datiem) vai VPP-Aranet (Aranet sensoru datiem). Python skripts, kas raksta datubāzē ārējo sensoru datus iespējams apskatīt pielikumā.

3. MongoDB datubāze

MongoDB pamatā ir NoSQL dokumentu krātuves modelis, kurā datu objekti tiek saglabāti kā atsevišķi dokumenti kolekcijās, nevis kā tradicionālās relāciju datubāzēs tabulu kolonnās un rindās. Dokumenti tiek glabāti kā bināri JSON vai BSON objekti.

MongoDB datubāze atrodas uz AWS mākoņa, kur tai pieslēgties var tikai autentificēti lietotāji. Pieslēgties datubāzei var norādot AWS mākoņa publisko IPV4 adresi, kā arī ir jānorāda MongoDB ports, kas ir mongoDB noklusētais ports :27017. Pieslēgties datubāzei var caur dažādiem MongoDB rīkiem, piemēram, Compass - vieglākai datu vizualizācijai, mongo shell - komandrindā vai caur citu programmu, kur Mongo piedāvā bibliotēkas vairākas valodās, to starpā - Python, Java, C, C++ u.c. Projektā tiek izmantota MongoDB 4.4.1 versija [5]

3.1. Datubāzes struktūra

MongoDB ir NoSQL datubāze, kas nozīmē, ka tajā nav tabulu, bet gan kolekcijas, kurās atrodas dokumenti JSON formātā. Tālāk tiek aprakstīts datubāzē izveidotās kolekcijas. Kolekcija darbojas līdzīgi tabulai tradicionālajā SQL datu bāzē.

3.1.1. Kolekcijas

Datubāzē envMonDB (environment monitoring database) glabā sensoru datus, kur sensori tiek pievienoti istabām un tālāk istabas tiek pievienotas iestādēm, tas vajadzīgs, lai varētu zināt, kur sensori atrodas. Izveidotās kolekcijas:

- Iestādes (institutions) - iestāžu nosaukumi
- Telpas (rooms) - telpas kas ir iestādēs
- Aranet Dati (sensorDataAranet) – dati, kas nāk no Aranet4 sensoriem
- Arenet Alert paziņojumi (aranetAlerts) – kļūdu paziņoji Aranet4 sensoriem
- Ārējo sensoru dati (sensorDataWP1) - dati, kas nāk no ārējiem sensoriem

3.1.2. Struktūra pa kolekcijām

- **Institutions**

```
{
      "_id": {
        "$oid": "hash"
      },
      "name": "LU DF"
}
```

- **Rooms**

```
{
      "_id": {
        "$oid": "hash"
      },
      "name": "310",
      "institution_id": "hash"
}
```

- **sensorDataAranet**

```
{
      "_id": {
           "$oid": "hash"
      },
      "atmosphericpressure": 101170,
      "co2": 817,
      "co2Abc": 0,
      "humidity": 54,
      "temperature": 21.3,
      "rssi": -68,
      "time": 1603717622,
      "battery": 0.95,
```

```
      "sensor_id": "123A5"
}
```


- **aranetAlerts**


```
{
      "_id": {
            "$oid": "hash"
      },
      "sensor_id": "123A5",
      "alarm_type": "packetslost",
      "activesince": "1605177249",
      "timestamp": 1605177249.582793
}
```


- **sensorDataWP1**


```
{
      "_id": {
            "$oid": "hash"
      },
      "person": 0,
      "ts": 1605177216.603715,
      "Topic": "TTT-WP1/SMI/building01/room01"
},
{
      "_id": {
            "$oid": "hash"
      },
      "CLASS": "noise",
      "PROBABILITY": 0.8302324414253235,
      "ts": 1605177249.582793,
      "Topic": "TTT-WP1/SMI/building01/room01"
},
{
      "_id": {
            "$oid": "hash"
      },
      "T": 26.95,
      "RH": 23.3,
      "CO2": 477,
      "PM10": 2.7,
      "PM2,5": 2.5,
      "ts": 1605177446.097504,
      "Topic": "TTT-WP1/SMI/building01/room01"
}
```

ARANET SENSORI

Aranet ir kompānija, kas specializējas IoT risinājumos, kur tā piedāvā noliktavu, lielveikalu un citu iekštelpu monitorēšanas risinājumus un tiem paradzētus sensorus. Šajā projektā tiek izmantoti viņu sensori "Aranet4".

1. Aranet4

Aranet4 ir ar baterijām darbināms autonoms bezvadu CO2, temperatūras, relatīvā mitruma un atmosfēras spiediena sensors, kurš paredzēts iekštelpu monitorēšanai. Ierīce ir viegli pārnēsājama un uzstādāma. Tā kā sensori izmanto e-ink [6] displejus, tie patērē mazāk baterijas, kā arī šiem sensoriem ir iespējams uzstādīt mērījumu biežumu. Šo biežumu var uzstādīt ar Aranet Pro bāzes palīdzību.

2. Aranet PRO

Ir bāzes stacija pie kuras ir iespējams pieslēgt Aranet4 sensorus. Bāzei līdzi nāk programmatūra, kas strādā kā sistēmas GUI. Šajā Aranet bāzē ir iespējams saglabāt sensoru mērījumus, kur Aranet PRO specifikācijā [8] tiek norādīts, ka 100 sensoru mērījumu dati var tikt saglabāti 10 gadu garumā.

Projektā ir nepieciešam saglabāt datus datubāzē, kur tas tiek darīts ar MQTT protokola palīdzību. Lai iespējotu MQTT protokola izmantošanu ir nepieciešama licence no Aranet, kā arī nepieciešama piekļuve pie interneta, lai datu var nosūtīt uz datubāzi. Interneta piekļuvi var nodrošināt ar ethernet kabeli vai pievienot to caur WiFi tīklu. Arenet Pro darbojas arī kā maršrutētājs, pie kura var pieslēgties.

KOPSAVILKUMS

Projekta ietvaros ir izstrādāta datu bāze un pielāgots MQTT brokeris sensoru datu saglabāšanao datu mākonī. Datu bāze ir izvēlēta MongoDB, kas pieļauj elastīgu datu struktūras izmaiņu un līdz ar to tās papildināšanu ar jauna veida sensoriem un notikumiem. Sensoru dati visupirms tiek sūtīti datu brokerim MQTT, kurš spēj tos saņemt no dažādiem avotiem un vietām, kā arī servēt datus klientiem – pieprasītājiem. Šādi klienti ir datu bāze, kur dati tiek saglabāti ilgtermiņā, kā arī programmatūras aģenti, kas, piemēram, izmanto reāla laika sensoru datus lai operatīvi aprēķinātu inficēšanās risku attiecīgajā zonā. Datubāze un brokeris ir testēti vairāku mēnešu garumā ar paralelu pieslēgumu no sensoriem kas izvietoti vairākās organizācijās, tai skaitā LU un PSKUS.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. 3-T Project [tiešsaiste] - [atsauce 15.12.2020]

https://lzp.gov.lv/project/3-t-project/

2. Eclipse Mosquitto [tiešsaiste] – [atsauce 15.12.2020]

https://mosquitto.org/

3. MQTT 3.1.1 [tiešsaiste] – [atsauce 15.12.2020]

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf

4. TLS protocol 1.3 [tiešsaiste] – [atsauce 15.12.2020]

https://tools.ietf.org/html/rfc8446

5. MongoDB Manual 3 [tiešsaiste] – [atsauce 15.12.2020]

https://docs.mongodb.com/manual/

6. E-ink [tiešsaiste] - [atsauce 17.12.2020]

https://www.eink.com/electronic-ink.html

7. Aranet4 Manual [tiešsaiste] - [atsauce 17.12.2020]

https://dl.aranet.com/wp-content/uploads/2020/11/10221825/Aranet4_Sensor_Manual_v2.01__WEB.pdf

8. Aranet PRO specification [tiešsaiste] - [atsauce 17.12.2020]

https://aranet.com/product/wireless-iot-base-station-3/

PIELIKUMI

1. MQTT brokera config fails:

```
# MQTT BROKER CONFIG
listener 8883
protocol mqtt

cafile /etc/mosquitto/certs/cert.pem
certfile /etc/mosquitto/certs/broker.pem
keyfile /etc/mosquitto/certs/private-key.pem

allow_anonymous false
password_file /etc/mosquitto/passwd
```

2. Ārējo sensoru python skripts:

```python
# python3
import paho.mqtt.client as mqtt
import json
from datetime import datetime
from pymongo import MongoClient

# Add topic to message
def add_topic(msg, tString):
    if len(tString) <= 255:
        jsonString = {"topic": tString}
        msg.update(jsonString)

# Replaces dot because MongoDB does not support dot symbol in key names
def replace_dot(jsonMSG):
    for key in list(jsonMSG):
        if "." in key:
            jsonMSG[key.replace(".",",")] = jsonMSG.pop(key)

# Connect to MQTT broker
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("TTT-WP1/#")

# On received message write data in MongoDB
def on_message(client, userdata, msg):
    try:

        # Check if message is in json format
        message = msg.payload.decode("utf-8")
        jsonMSG = json.loads(message)

        if len(msg.topic.split('/')) >= 2:
            if msg.topic.split('/')[1] == "model":
                db = mongoClient.envMonDB_test
                collection = db.sensorDataWP1
                collection.insert_one(jsonMSG)

```

```
            elif len(msg.topic.split('/')) >= 4:
                replace_dot(jsonMSG)
                add_topic(jsonMSG, msg.topic)

                db = mongoClient.envMonDB_test
                collection = db.sensorDataWP1
                collection.insert_one(jsonMSG)
        else:
            print("No Room or Institution defined! Topic: " + msg.topic)
    except:
        message = str(msg.payload)
        now = datetime.now()
        print("Exception!\nTime: " + str(now) + "\nTopic: " + msg.topic + "\nMessage: "
    + message + "\nEnd of exception")

# Set up client for MongoDB
mongoClient = MongoClient('host', username = '', password = '', authSource = 'userDB',
authMechanism = 'SCRAM-SHA-256')

# Initialize the client that should connect to the Mosquitto broker
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.tls_set('path/to/cert.pem', tls_version=2)
client.username_pw_set(username = "", password = ")
client.connect("host", 8883)

# Blocking loop to the Mosquitto broker
client.loop_forever()
```