ELEKTRONIKOS FAKULTETAS
KOMPIUTERIJOS IR RYŠIŲ TECHNOLOGIJŲ KATEDRA

LABORATORINIO DARBO ATASKAITA

Laboratorinis darbas nr. 3

MQTT IR EXPO INTEGRACIJA

Studentas: Eimantas Repšys

Dėstytojas: **Tomas Cuzanauskas**

Vilnius, 2025

Ataskaitos turinys

1 Įvadas

Šio laboratorinio darbo tikslas buvo sujungti pateiktą EXPO mobiliosios aplikacijos programą su MQTT serveriu ir papildomai integruoti Python backend serverį, kuris klausytųsi pranešimų iš EXPO aplikacijos per MQTT brokerį. Darbo metu buvo įgyvendinti šie uždaviniai:

- MQTT serverio (Mosquitto) įdiegimas ir konfigūravimas savo virtualioje mašinoje su Ubuntu.
- EXPO programos pritaikymas prisijungimui prie Mosquitto serverio per WebSocket protokolą.
- Python backend serverio, naudojančio orų paieškos funkcionalumą, integracija.
- MQTT brokerio veikimo skirtumų tarp 1883 ir 8000 portų analizė

2 Darbo eiga

2.1 Virtualios mašinos paruošimas

- Naudota jau turima virtuali mašina su Ubuntu operacine sistema, sukonfigūruota pagal laboratorinio darbo reikalavimus.
- Nustatytas tiltinis (bridge) tinklo režimas, kad virtuali mašina būtų pasiekiama iš išorinio tinklo.
- Prisijungta per SSH naudojant terminalą ir atlikti sistemos atnaujinimai:
  sudo apt update && sudo apt upgrade -y
- Įdiegtas Mosquitto serveris ir pagalbiniai įrankiai:
  sudo apt install mosquitto mosquitto-clients vim -y
  sudo systemctl enable mosquitto
  sudo systemctl start mosquitto

2.2 Mosquitto konfigūravimas

Mosquitto konfigūracinis failas /etc/mosquitto/mosquitto.conf buvo atnaujintas, kad palaikytų tiek standartinį MQTT protokolą (1883 portas), tiek WebSocket protokolą (8000 portas):

listener 1883

listener 8000

protocol websockets

allow_anonymous true

persistence true

persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

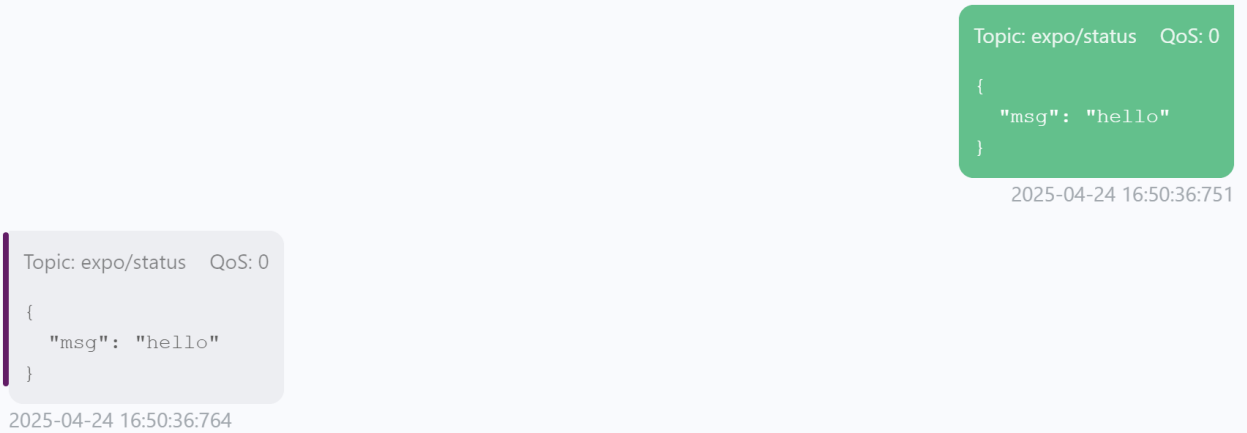
include_dir /etc/mosquitto/conf.d

Patikrinta, ar Mosquitto serveris klausosi nurodytų portų:

```
root@lab2:/home/reimis# sudo netstat -tulnp | grep mosquitto
tcp        0      0 0.0.0.0:1883            0.0.0.0:*               LISTEN      674/mosquitto
tcp6       0      0 :::1883                 :::*                    LISTEN      674/mosquitto
tcp6       0      0 :::8000                 :::*                    LISTEN      674/mosquitto
```

2.3 EXPO programos integracija

- Atsisiųsta EXPO programa iš Moodle platformos ir išarchyvuota.
- Programos kode surastas Paho MQTT kliento prisijungimo adresas ir atnaujintas, nurodant Mosquitto serverio IP adresą ir 8000 portą (WebSocket protokolas).
- Programa paleista naudojant komandą: *npx expo start*
- Naudojant MQTTX įrankį patikrinta, ar EXPO aplikacija sėkmingai siunčia ir gauna pranešimus per tą pačią MQTT temą.

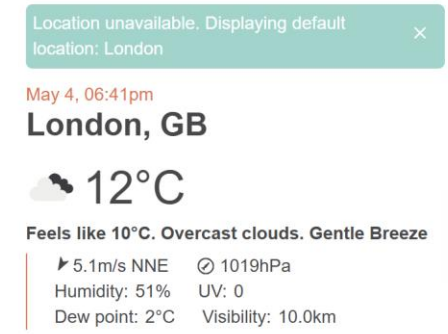


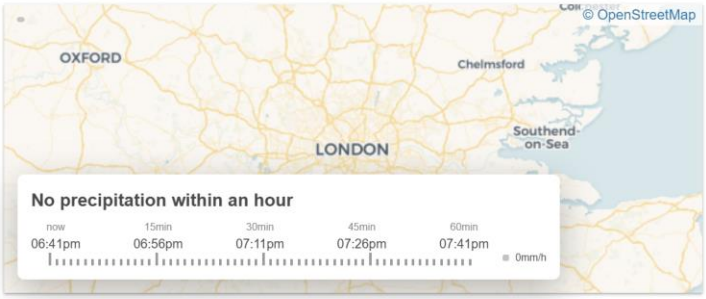
3 Python backend integracija

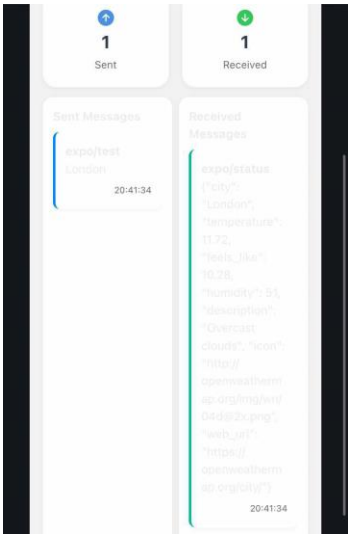
Buvo sukurti weather.py ir mqtt_sub.py failai, kurie realizuoja backend serverio funkcionalumą, skirtą orų informacijos paieškai:

- weather.py atsakingas už orų duomenų gavimą iš OpenWeatherMap API pagal pateiktą miesto pavadinimą.
- mqtt_sub.py sukonfigūruotas klausytis MQTT pranešimų per 1883 portą ir, gavus pranešimą, atlikti šiuos veiksmus:
    - Gauti orų informaciją naudojant weather.py.
    - Atidaryti naršyklės puslapį su OpenWeatherMap miesto orų informacija.
    - Grąžinti orų duomenis į MQTT temą expo/status.

```
PS C:\Users\ninje\desktop\lab-backend> python mqtt-sub.py
C:\Users\ninje\desktop\lab-backend\mqtt-sub.py:30: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
  client = mqtt.Client(client_id="WeatherSubscriber", protocol=mqtt.MQTTv311)
Connected with result code 0
Received city query: London
Weather result: {'city': 'London', 'temperature': 11.72, 'feels_like': 10.28, 'humidity': 51, 'description': 'Overcast clouds', 'icon': 'http://open
weathermap.org/img/wn/04d@2x.png', 'web_url': 'https://openweathermap.org/city/'}
```







4 Rezultatai

- Sėkmingai įdiegtas ir sukonfigūruotas Mosquitto MQTT brokeris, palaikantis 1883 (standartinis MQTT) ir 8000 (WebSocket) portus.
- EXPO programa sėkmingai prijungta prie Mosquitto serverio per WebSocket protokolą, užtikrinant stabilų pranešimų siuntimą ir gavimą.
- Python backend serveris sėkmingai apdoroja MQTT pranešimus, gauna orų informaciją, atidaro naršyklės puslapį ir grąžina rezultatus į EXPO aplikaciją.
- Demonstracija atlikta naudojant vietinį tinklą, sukurtą per telefono hotspotą.

4.1 1883 ir 8000 portų skirtumai

- **1883 portas**: Naudojamas standartiniam MQTT protokolui, veikiančiam per TCP. Šis portas dažniausiai taikomas įrenginiams, palaikantiems tiesioginį MQTT ryšį, pavyzdžiui, IoT įrenginiams ar desktop aplikacijoms. Protokolas yra lengvas ir efektyvus, tačiau reikalauja MQTT bibliotekų palaikymo.

- **8000 portas**: Naudojamas MQTT per WebSocket protokolą, kuris leidžia MQTT pranešimus siųsti per HTTP/WS. Šis portas būtinas naršyklinėms aplikacijoms, tokioms kaip EXPO, veikiančioms mobiliosiose platformose. WebSocket prideda šiek tiek papildomos apkrovos, tačiau užtikrina suderinamumą su naršyklių aplinkomis.

5 Išvados

Laboratorinio darbo metu sėkmingai įgyvendinta EXPO aplikacijos ir Mosquitto MQTT serverio integracija, papildomai prijungiant Python backend serverį su orų paieškos funkcionalumu. Darbas leido praktiškai išbandyti MQTT protokolo veikimą, suprasti skirtumus tarp standartinio MQTT ir WebSocket ryšių bei įgyti patirties dirbant su virtualiomis mašinomis ir tinklo konfigūracija. Sukurtas sprendimas yra funkcionalus ir gali būti pritaikytas realiose aplikacijose, kur reikalingas realaus laiko pranešimų perdavimas tarp mobiliosios aplikacijos ir serverio