WOJSKOWA AKADEMIA TECHNICZNA

**im. Jarosława Dąbrowskiego**

**WYDZIAŁ CYBERNETYKI**



STUDIA I°

Temat: **Sprawozdanie z realizacji ćwiczenia laboratoryjnego**

**INFORMATYKA**

…………………………………………………………………………

(kierunek studiów)

**INŻYNIERIA SYSTEMÓW**

…………………………………………………………………………

(specjalność)

|  |  |
| --- | --- |
| Wykonał: | Prowadzący: |
| **Radosław Relidzyński** | **mgr inż. Tomasz Gumowski** |

**Warszawa 2023**

Spis treści

[Rozdział I. Treść zadań 4](#_Toc150800895)

[Rozdział II. Implementacja zadania 5](#_Toc150800896)

[Rozdział III. Opis działania zadania 7](#_Toc150800897)

[III.1. Deklaracja sensora 7](#_Toc150800898)

[III.2. Deklaracja listy sensorów (dla każdego jest jego nazwa i odpowiadająca funkcja sensora) 7](#_Toc150800899)

[III.3. Wyświetlenie wiadomości powitalnej 7](#_Toc150800900)

[III.4. Wybór rodzaju sensora 7](#_Toc150800901)

[III.5. Monitorowanie sensora (monitorowanie kliknięć dla joysticka lub cykliczne wypisywanie dla pozostałych) 7](#_Toc150800902)

[III.6. Wyświetlenie wiadomości pożegnalnej 7](#_Toc150800903)

[III.7. Zakończenie programu 7](#_Toc150800904)

[Rozdział IV. Prezentacja działania programu 7](#_Toc150800905)

[IV.1. Wyświetlenie wiadomości powitalnej 7](#_Toc150800906)

[IV.2. Wybór sensora 8](#_Toc150800907)

[IV.3. Monitorowanie sensora cyklicznie wypisującego bez manipulacji wartością 8](#_Toc150800908)

[IV.4. Monitorowanie sensora cyklicznie wypisującego z manipulacją wartości (najpierw do maksymalnej wartości, potem do minimalnej) 9](#_Toc150800909)

[IV.5. Monitorowanie joysticka bez klikania 10](#_Toc150800910)

[IV.6. Monitorowanie joysticka z klikaniem 10](#_Toc150800911)

[Rozdział V. Podsumowanie 11](#_Toc150800912)

1. Treść zadań

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

1. Implementacja zadania
   1. Struktura projektu

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

* 1. Kontener z brokerem MQTT

Dockerfile

FROM eclipse-mosquitto:2.0  
  
COPY mosquitto.conf /mosquitto/config/mosquitto.conf

mosquito.conf

allow\_anonymous true  
listener 1883

* 1. Kontener z usługami FastAPI, usługi dostępne również z hosta

Dockerfile

FROM python:3.11  
  
WORKDIR /app  
  
COPY requirements.txt /app/  
RUN pip install --no-cache-dir -r requirements.txt  
  
COPY . /app/  
  
CMD ["python", "-m", "uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]

requirements.txt

fastapi  
uvicorn

main.py

from fastapi import FastAPI  
import random  
  
app = FastAPI()  
DATA = list()  
  
  
@app.get("/get\_first\_names")  
def get\_data1():  
 result = DATA  
 return [row['first\_name'] for row in result]  
  
  
@app.get("/get\_emails")  
def get\_data2():  
 result = DATA  
 return [row['email'] for row in result]  
  
  
@app.get("/get\_all")  
def get\_data3():  
 result = DATA  
 return result  
  
  
@app.post("/add\_person/")  
def add\_person(first\_name: str, last\_name: str, age: int, email: str):  
 person = {  
 'first\_name': first\_name,  
 'last\_name': last\_name,  
 'age': age,  
 'email': email,  
 }  
 DATA.append(person)  
 return "Successfully added a person"  
  
  
@app.post("/add\_random\_person")  
def add\_random\_person():  
 first\_names = ["A", "B", "C"]  
 last\_names = ["AA", "BB", "CC"]  
  
 first\_name = random.choice(first\_names)  
 last\_name = random.choice(last\_names)  
 age = random.randint(18, 60)  
 email = f"{first\_name}.{last\_name}@gmail.com"  
  
 person = {  
 'first\_name': first\_name,  
 'last\_name': last\_name,  
 'age': age,  
 'email': email,  
 }  
 DATA.append(person)  
 return "Successfully added a random person"  
  
  
@app.delete("/delete\_people")  
def delete\_people():  
 DATA.clear()  
 return "Successfully deleted people"  
  
  
# if \_\_name\_\_ == "\_\_main\_\_":  
# DATA.clear()  
# import uvicorn  
#  
# uvicorn.run(app, host="0.0.0.0")

* 1. Kontener z aplikacją pobierającą dane z FastAPI po otrzymaniu pewnej wiadomości (MQTT), dane są następnie publikowane na inny temat

Dockerfile

FROM python:3.11  
  
WORKDIR /app  
  
COPY requirements.txt /app/  
RUN pip install --no-cache-dir -r requirements.txt  
  
COPY . /app/  
  
CMD ["python", "-u", "subscriber.py"]

requirements.txt

paho-mqtt

subscriber.py

import sys  
  
import paho.mqtt.client as mqtt  
  
  
def connect\_handling(client, userdata, flags, rc):  
 print("Connected with result code " + str(rc))  
 client.subscribe("parse\_add\_person")  
  
  
def message\_handling(client, userdata, msg):  
 print(f"Received message on topic: {msg.topic}: {msg.payload.decode()}")  
  
  
client = mqtt.Client()  
client.on\_connect = connect\_handling  
client.on\_message = message\_handling  
  
  
if client.connect("mqtt\_broker", 1883, 60) != 0:  
 print("Couldn't connect to the mqtt broker")  
 sys.exit(1)  
  
client.loop\_forever()

* 1. Kontener z klientem MQTT subskrybującym na temat, na który publikuje inny klient

Dockerfile

# Dockerfile.mqttApp  
FROM python:3.11  
  
WORKDIR /app  
  
COPY requirements.txt /app/  
RUN pip install --no-cache-dir -r requirements.txt  
  
COPY . /app/  
  
CMD ["python", "-u", "mqtt\_app.py"]

requirements.txt

paho-mqtt  
requests

mqtt\_app.py

import sys  
import paho.mqtt.client as mqtt  
import requests  
  
  
def connect\_handling(client, userdata, flags, rc):  
 print("Connected with return code:" + str(rc))  
 client.subscribe("add\_person")  
  
  
def message\_handling(client, userdata, msg):  
 print(f"Received message on topic: {msg.topic}: {msg.payload.decode()}")  
  
 response = requests.post("http://fastapi:8000/add\_random\_person")  
 data = response.json()  
 client.publish("data\_from\_fastapi", str(data), qos=0)  
  
 response = requests.get("http://fastapi:8000/get\_all")  
 data = response.json()  
 client.publish("parse\_add\_person", str(data), qos=0)  
  
  
mqtt\_client = mqtt.Client()  
mqtt\_client.on\_connect = connect\_handling  
mqtt\_client.on\_message = message\_handling  
  
if mqtt\_client.connect("mqttBroker") != 0:  
 print("Couldn't connect to the mqtt broker")  
 sys.exit(1)  
  
mqtt\_client.loop\_forever()

* 1. Plik docker-compose.yml

version: '3'  
  
services:  
 mqtt\_broker:  
 container\_name: mqttBroker  
 build:  
 context: mqttBroker  
 dockerfile: Dockerfile  
 ports:  
 - "1883:1883"  
  
 fastapi\_service:  
 container\_name: fastapi  
 build:  
 context: fastapiService  
 dockerfile: Dockerfile  
 ports:  
 - "8000:8000"  
  
 mqtt\_app:  
 build:  
 context: mqttApp  
 dockerfile: Dockerfile  
 depends\_on:  
 - fastapi\_service  
 - mqtt\_broker  
  
 mqtt\_subscriber:  
 build:  
 context: ./mqttSubscriber  
 dockerfile: Dockerfile  
 depends\_on:  
 - mqtt\_broker

1. Opis działania zadania
   1. Kontener z brokerem MQTT

Przechwytuje połączenia i zapytania mqtt

* 1. Kontener z usługami FastAPI, usługi dostępne również z hosta

Posiada mechanizm zarządzania oraz przechowywania danych

* 1. Kontener z aplikacją pobierającą dane z FastAPI po otrzymaniu pewnej wiadomości (MQTT), dane są następnie publikowane na inny temat

Przejmuje zapytania i przekierowuje na ich podstawie informację do innego wątku

* 1. Kontener z klientem MQTT subskrybującym na temat, na który publikuje inny klient

Przejmuje zapytania od klienta i przekazuje do innego wątku

1. Prezentacja działania programu
   1. Zbudowanie i uruchomienie kontenerów poleceniem „docker-compose up ”:

[+] Building 1.7s (17/17) docker:default

=> [mqtt\_broker internal] load .dockerignore 0.0s

[+] Building 2.2s (31/31) FINISHED docker:default

=> [mqtt\_broker internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> [mqtt\_broker internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 121B 0.0s

=> [fastapi\_service internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> [fastapi\_service internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 258B 0.0s

=> [mqtt\_broker internal] load metadata for docker.io/library/eclipse-mosquitto:2.0 1.5s

=> [mqtt\_app internal] load metadata for docker.io/library/python:3.11 0.0s

=> [mqtt\_app 1/5] FROM docker.io/library/python:3.11 0.0s

=> [fastapi\_service internal] load build context 0.0s

=> => transferring context: 94B 0.0s

=> CACHED [mqtt\_app 2/5] WORKDIR /app 0.0s

=> CACHED [fastapi\_service 3/5] COPY requirements.txt /app/ 0.0s

=> CACHED [fastapi\_service 4/5] RUN pip install --no-cache-dir -r requirements.txt 0.0s

=> CACHED [fastapi\_service 5/5] COPY . /app/ 0.0s

=> [fastapi\_service] exporting to image 0.0s

=> => exporting layers 0.0s

=> => writing image sha256:5cd0a25029fec62eee618a5829faa38c19cc4311cdf9bd13339ed8cf80f5c0db 0.0s

=> => naming to docker.io/library/project-fastapi\_service 0.0s

=> [mqtt\_broker internal] load build context 0.0s

=> => transferring context: 35B 0.0s

=> [mqtt\_broker 1/2] FROM docker.io/library/eclipse-mosquitto:2.0@sha256:09bf8853ef2c5cf5356f0924aec318dce2b304cd63ece091c2cf9973aba7258e 0.0s

=> CACHED [mqtt\_broker 2/2] COPY mosquitto.conf /mosquitto/config/mosquitto.conf 0.0s

=> [mqtt\_broker] exporting to image 0.0s

=> => exporting layers 0.0s

=> => writing image sha256:aa4f7bdd0d9e967196196e32053ebfd3dda845f0bd161fb8751dab4fa66d86ee 0.0s

=> => naming to docker.io/library/project-mqtt\_broker 0.0s

=> [mqtt\_subscriber internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 213B 0.0s

=> [mqtt\_subscriber internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> [mqtt\_subscriber internal] load build context 0.0s

=> => transferring context: 281B 0.0s

=> [mqtt\_app internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 231B 0.0s

=> [mqtt\_app internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> CACHED [mqtt\_subscriber 3/5] COPY requirements.txt /app/ 0.0s

=> CACHED [mqtt\_subscriber 4/5] RUN pip install --no-cache-dir -r requirements.txt 0.0s

=> [mqtt\_subscriber 5/5] COPY . /app/ 0.0s

mqttBroker | 1700931584: New connection from 172.29.0.4:49713 on port 1883.

mqttBroker | 1700931584: New client connected from 172.29.0.4:49713 as auto-D78F50A0-71C9-AE74-B3F7-94EDADBBF1BC (p2, c1, k60).

project-mqtt\_subscriber-1 | Connected with result code 0

fastapi | INFO: Started server process [1]

fastapi | INFO: Waiting for application startup.

fastapi | INFO: Application startup complete.

fastapi | INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)

project-mqtt\_app-1 | Connected with return code:0

mqttBroker | 1700931584: New connection from 172.29.0.5:46293 on port 1883.

mqttBroker | 1700931584: New client connected from 172.29.0.5:46293 as auto-CAEAB77C-AA15-015E-C292-1A20AA56971A (p2, c1, k60).

* 1. Wysłanie zapytania do aplikacji klienta poprzez mqtt.fx (przyciskiem „publish”):

Obraz zawierający tekst, oprogramowanie, Ikona komputerowa, Strona internetowa

Opis wygenerowany automatycznie

* 1. Przechwycenie informacji od publishera

mqttBroker | 1700931727: New connection from 172.29.0.1:60762 on port 1883.

mqttBroker | 1700931727: New client connected from 172.29.0.1:60762 as 4cc1aa5f953b42598f0ba159abdc4656 (p2, c1, k60).

project-mqtt\_app-1 | Received message on topic: add\_person:

fastapi | INFO: 172.29.0.5:40776 - "POST /add\_random\_person HTTP/1.1" 200 OK

fastapi | INFO: 172.29.0.5:40782 - "GET /get\_all HTTP/1.1" 200 OK

project-mqtt\_subscriber-1 | Received message on topic: parse\_add\_person: [{'first\_name': 'B', 'last\_name': 'CC', 'age': 20, 'email': 'B.CC@gmail.com'}]

* 1. Kilkukrotne wysłanie wiadomości

project-mqtt\_subscriber-1 | Received message on topic: parse\_add\_person: [{'first\_name': 'B', 'last\_name': 'CC', 'age': 20, 'email': 'B.CC@gmail.com'}]

project-mqtt\_app-1 | Received message on topic: add\_person:

fastapi | INFO: 172.29.0.5:44006 - "POST /add\_random\_person HTTP/1.1" 200 OK

fastapi | INFO: 172.29.0.5:44022 - "GET /get\_all HTTP/1.1" 200 OK

project-mqtt\_subscriber-1 | Received message on topic: parse\_add\_person: [{'first\_name': 'B', 'last\_name': 'CC', 'age': 20, 'email': 'B.CC@gmail.com'}, {'first\_name': 'B', 'last\_name':

'CC', 'age': 34, 'email': 'B.CC@gmail.com'}]

project-mqtt\_app-1 | Received message on topic: add\_person:

fastapi | INFO: 172.29.0.5:44026 - "POST /add\_random\_person HTTP/1.1" 200 OK

fastapi | INFO: 172.29.0.5:44042 - "GET /get\_all HTTP/1.1" 200 OK

project-mqtt\_subscriber-1 | Received message on topic: parse\_add\_person: [{'first\_name': 'B', 'last\_name': 'CC', 'age': 20, 'email': 'B.CC@gmail.com'}, {'first\_name': 'B', 'last\_name':

'CC', 'age': 34, 'email': 'B.CC@gmail.com'}, {'first\_name': 'C', 'last\_name': 'AA', 'age': 33, 'email': 'C.AA@gmail.com'}]

project-mqtt\_app-1 | Received message on topic: add\_person:

fastapi | INFO: 172.29.0.5:44048 - "POST /add\_random\_person HTTP/1.1" 200 OK

fastapi | INFO: 172.29.0.5:44052 - "GET /get\_all HTTP/1.1" 200 OK

project-mqtt\_subscriber-1 | Received message on topic: parse\_add\_person: [{'first\_name': 'B', 'last\_name': 'CC', 'age': 20, 'email': 'B.CC@gmail.com'}, {'first\_name': 'B', 'last\_name':

'CC', 'age': 34, 'email': 'B.CC@gmail.com'}, {'first\_name': 'C', 'last\_name': 'AA', 'age': 33, 'email': 'C.AA@gmail.com'}, {'first\_name': 'C', 'last\_name': 'BB', 'age': 60, 'email': 'C.B

B@gmail.com'}]

1. Podsumowanie

Zadanie to dostarczyło praktycznej wiedzy na temat implementacji mechanizmów technologii Internetu rzeczy. Celem ćwiczenia było zbudowanie aplikacji pozwalającej na przechwytywanie i odpowiednie przekazywanie wiadomości od użytkownika i dokonywanie na jego podstawie działań wewnątrz systemu. Udało się stworzyć komunikację z każdym wymienionym w treści zadania elementem zaimplementowanym jako niezależny kontener Dockera.

Ćwiczenie pozwoliło zrozumieć w jaki sposób należy konstruować systemy w ramach technologii Internetu rzeczy oraz jak one się zachowują.