Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №7

з дисципліни «Інженерія систем IoT»

Тема: «Міграція IoT інфраструктури в хмару»

Варіант: 65

Виконав:
студент групи ІА-33
Заранік Максим

Перевірив:
асистент кафедри ІСТ
Головатенко І. А.

Київ 2025

Мета: Дослідити етапи розгортання інфраструктури IoT застосунку в хмарі

Хід роботи

Розгортання IoT інфраструктури в Azure.
Відповідно до завдання було активовано хмарну підписку Azure та підготовлено середовище для роботи з IoT Hub. Створено групу ресурсів, яка стала базовим контейнером для усіх компонентів хмарної інфраструктури. У межах цієї групи ресурсів було створено екземпляр IoT Hub безкоштовного тарифного рівня, що забезпечує можливість підключення пристроїв, приймання телеметрії та надсилання команд.

Реєстрація IoT-пристрою.
У створеному IoT Hub було зареєстровано пристрій “soil-moisture-sensor”. Під час реєстрації сформовано індивідуальний рядок підключення, який містить інформацію про хаб, ідентифікатор пристрою та секретний ключ. Цей рядок використовувався програмою датчика для автентифікації та встановлення захищеного з’єднання з хмарним сервісом.

Розробка програмного забезпечення пристрою.
Було створено Python-скрипт, який імітує роботу датчика вологості ґрунту. Програма встановлює з’єднання з IoT Hub, періодично обчислює поточне значення вологості та надсилає його у хмару у вигляді телеметричного повідомлення. Додатково у програму додано обробку прямих методів, що дозволяють з хмари керувати реле, вмикаючи або вимикаючи його за отриманою командою.

Код клієнтської частини:

app.py

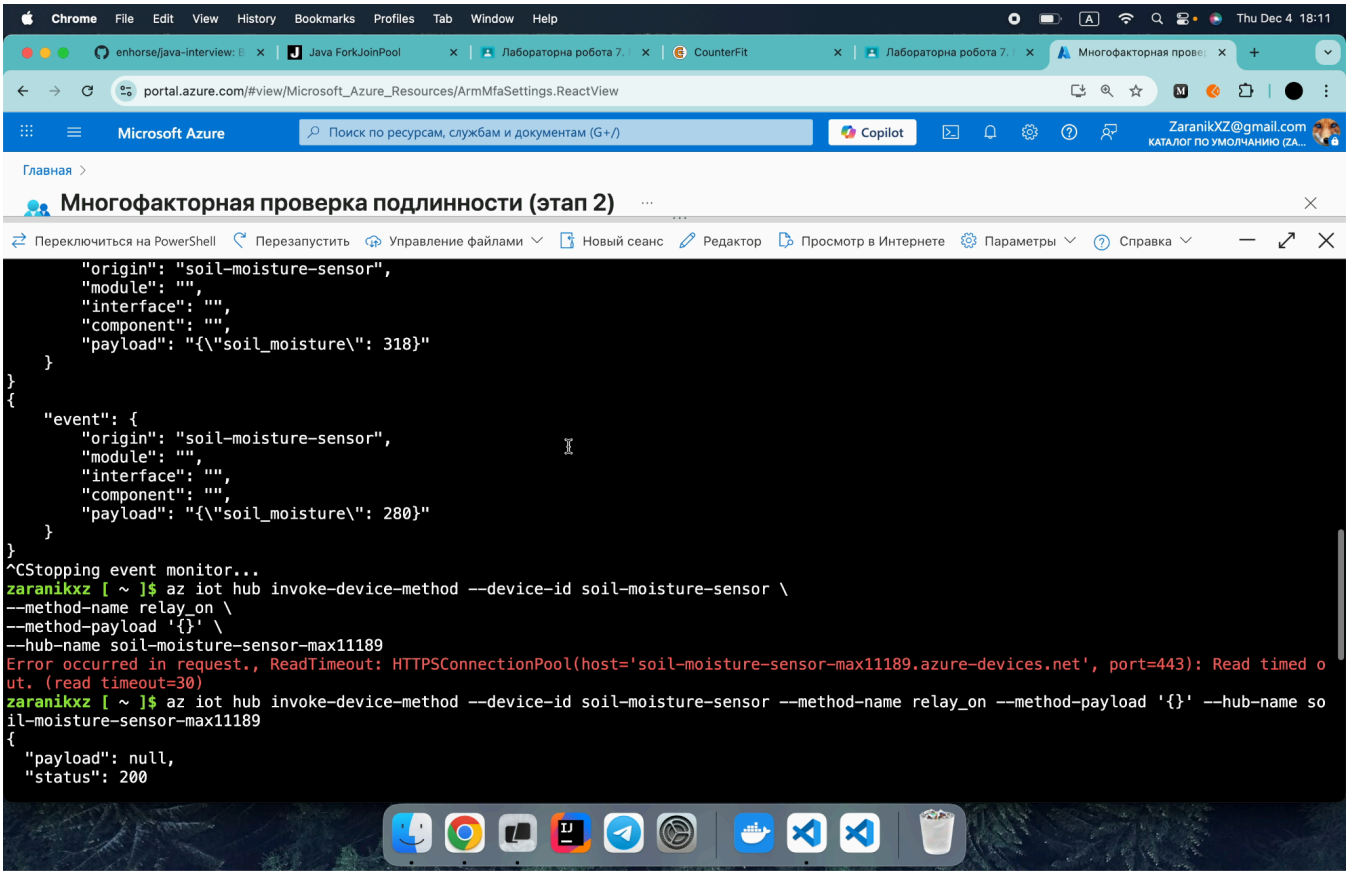
```python
from counterfit_connection import CounterFitConnection
import time
from counterfit_shims_grove.adc import ADC
from counterfit_shims_grove.grove_relay import GroveRelay
import json
from azure.iot.device import IoTHubDeviceClient, Message, MethodResponse

connection_string = "HostName=soil-moisture-sensor-max11189.azure-devices.net;DeviceId=soil-moisture-sensor;SharedAccessKey=+6A1CAZkhiAl3cxC90KNZWPoAy2znZN1z0SOqJ+0dbw="

device_client = IoTHubDeviceClient.create_from_connection_string(connection_string)
print('Connecting')
device_client.connect()
print('Connected')


CounterFitConnection.init('127.0.0.1', 5001)
adc = ADC()
relay = GroveRelay(66)


def handle_command(client, userdata, message):
    payload = json.loads(message.payload.decode())
    print("Command received:", payload)

    if payload.get('relay_on'):
        print("Turning relay ON")
        relay.on()
    else:
        print("Turning relay OFF")
        relay.off()

def handle_method_request(request):
    print("Direct method received - ", request.name)
    if request.name == "relay_on":
        print("-> Turning relay ON")
        relay.on()
    elif request.name == "relay_off":
        print("-> Turning relay OFF")
        relay.off()

    method_response = MethodResponse.create_from_method_request(request, 200)
    device_client.send_method_response(method_response)

device_client.on_method_request_received = handle_method_request

while True:
    soil_moisture = adc.read(65)
    print("Soil moisture:", soil_moisture)
    telemetry = json.dumps({'soil_moisture': soil_moisture})
    print("Sending telemetry:", telemetry)
    message = Message(json.dumps({ 'soil_moisture': soil_moisture }))
    device_client.send_message(message)
    time.sleep(10)
```

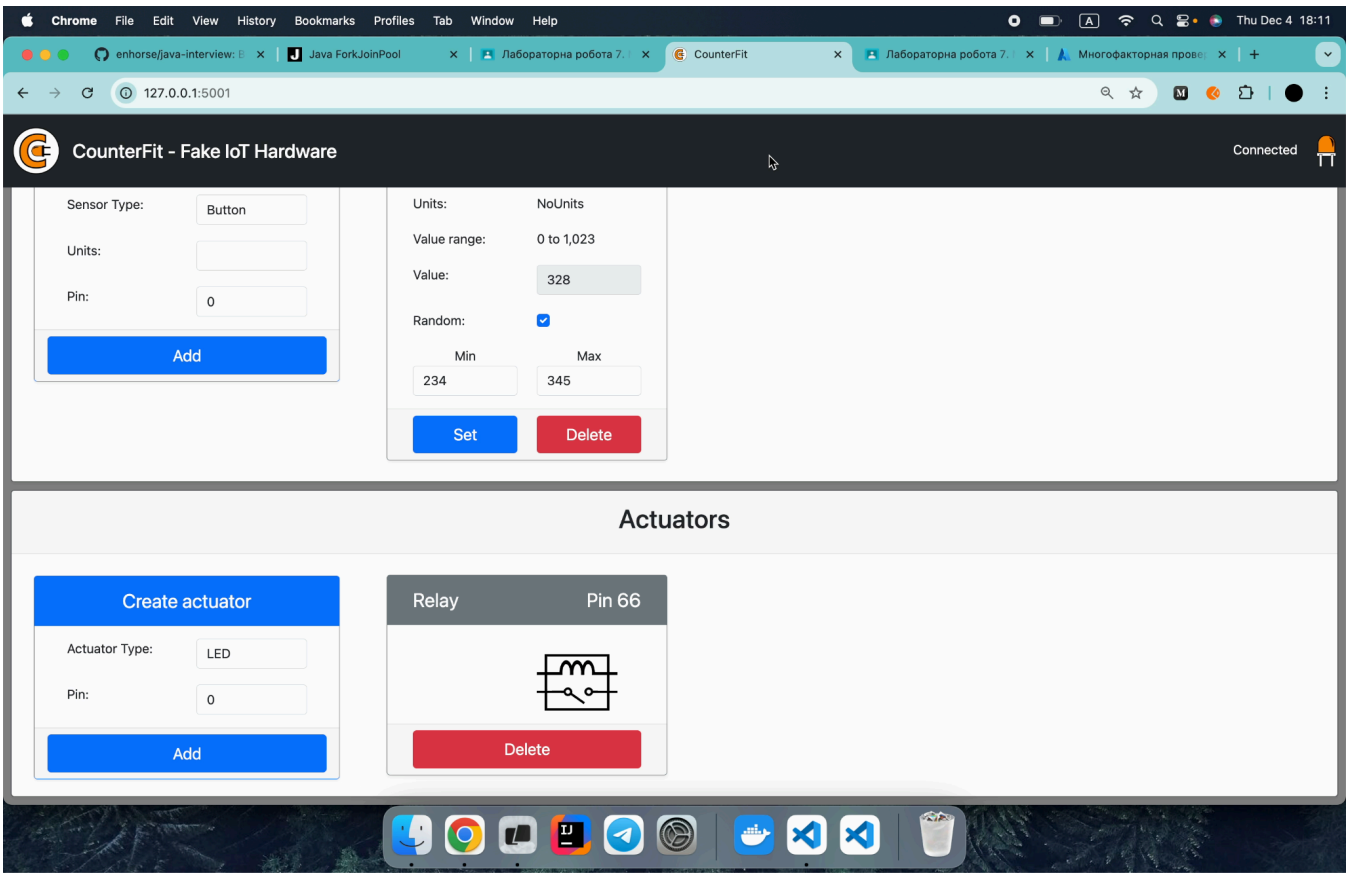
Код серверної частини:

app.py

```python
import json
import time
import paho.mqtt.client as mqtt
import threading

id = '9e9e2074-64b1-4dd2-8ee3-ad2ab0359a77'
client_telemetry_topic = id + '/telemetry'
server_command_topic = id + '/command'
client_name = id + 'soil_moisture_sensor_server'

mqtt_client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2, client_name)
mqtt_client.connect('test.mosquitto.org')
mqtt_client.loop_start()

water_time = 5
wait_time = 20

def send_relay_command(client, state):
    command = {'relay_on': state}
    print("Sending message:", command)
    client.publish(server_command_topic, json.dumps(command))

def control_relay(client):
    mqtt_client.unsubscribe(client_telemetry_topic)
    send_relay_command(client, True)
    time.sleep(water_time)
    send_relay_command(client, False)
    time.sleep(wait_time)
    mqtt_client.subscribe(client_telemetry_topic)

def handle_telemetry(client, userdata, message):
    payload = json.loads(message.payload.decode())
    print("Message received:", payload)
    if payload['soil_moisture'] > 486:
        threading.Thread(target=control_relay, args=(client,)).start()

mqtt_client.subscribe(client_telemetry_topic)
mqtt_client.on_message = handle_telemetry

while True:
    time.sleep(2)
```

Результат роботи:





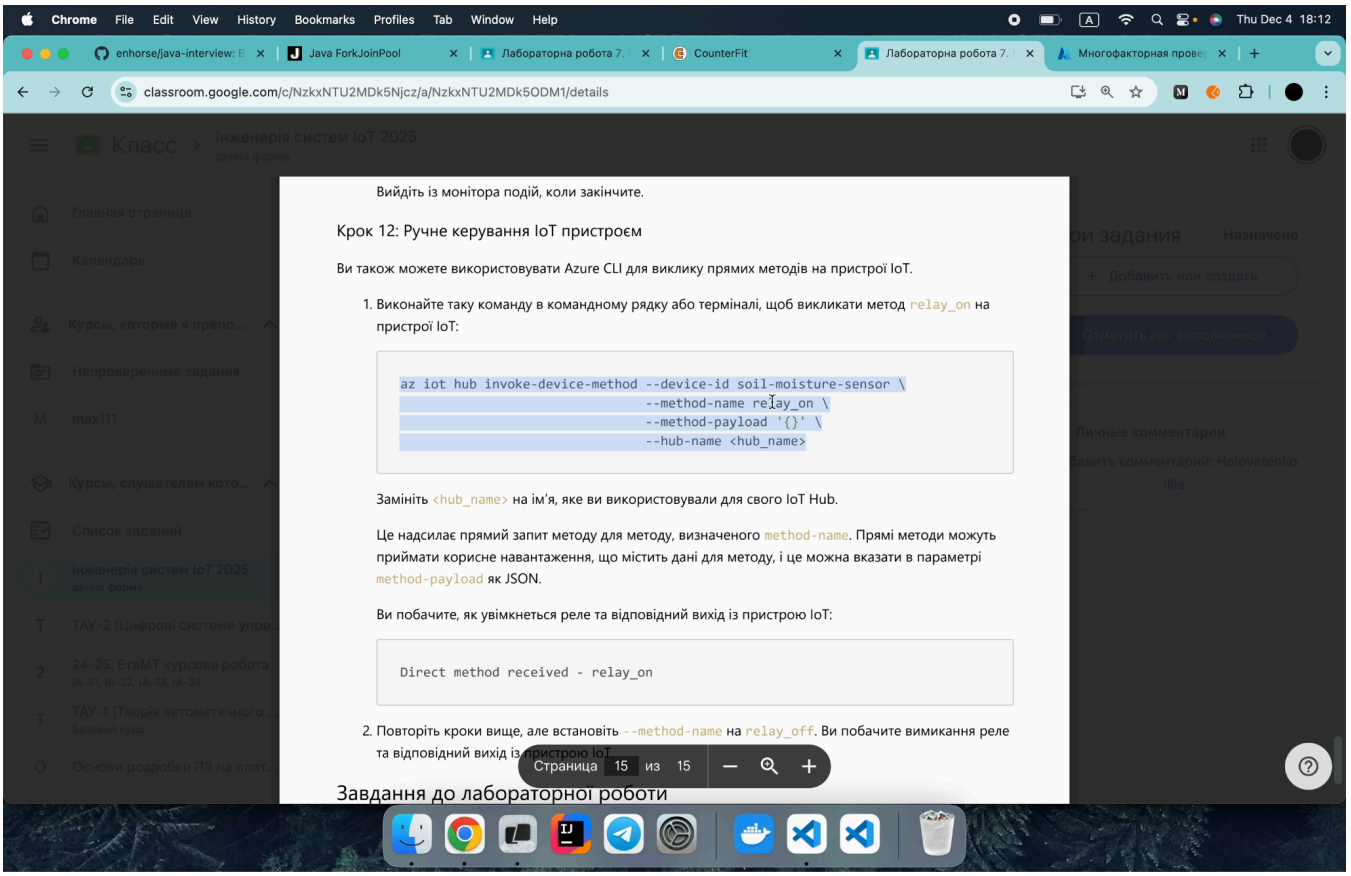


Рисунок 1.1 - інтерфейс Counterfit

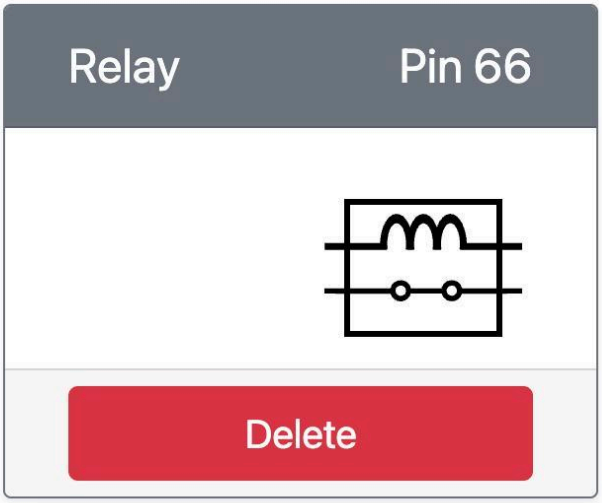


Рисунок 1.2 - стан реле “замкнутий”

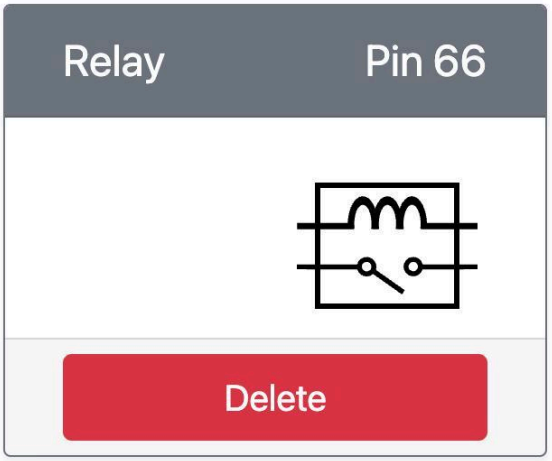


Рисунок 1.3 - стан реле “розімкнутий”

Посилання на github: https://github.com/makszaranik/IOT-7

Висновок:

У ході виконання лабораторної роботи №7 було набуто практичних навичок створення та налаштування хмарної інфраструктури для IoT-рішень на базі Microsoft Azure, а також використання емуляційного середовища CounterFit для тестування роботи пристроїв без фізичного обладнання. У середовищі CounterFit було сконфігуровано аналоговий датчик вологості ґрунту та цифрове реле, що дозволило промоделювати роботу типового IoT-сценарію.
У хмарі було створено ресурсну групу та розгорнуто сервіс Azure IoT Hub, який виступає центральною точкою взаємодії між пристроями та хмарною платформою. На основі Python-скрипта реалізовано підключення емулятора до IoT Hub, зчитування аналогових даних із датчика та передавання телеметрії. Окремо було налаштовано логіку прийняття рішень на основі порогових значень вологості, що дозволяє автоматично активувати керувальний елемент (реле) при досягненні критичних показників.
Розроблена система імітує роботу «розумного» поливу та демонструє типовий життєвий цикл IoT-рішення: збір даних, хмарну обробку, генерацію керівного впливу та взаємодію з виконавчим пристроєм. Виконання лабораторної роботи дозволило закріпити практичні навички роботи з Azure CLI, IoT Hub, віртуальними пристроями CounterFit та принципами побудови розподілених IoT-систем.