

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №8

з дисципліни «Інженерія систем IoT»

Тема: «Міграція IoT коду в хмару»

Варіант: 65

Виконав:

студент групи IA-33

Заранік Максим

Перевірив:

асистент кафедри ICT

Головатенко І. А.

Київ 2025

Мета: Дослідити етапи розгортання інфраструктури IoT застосунку в хмарі.

Хід роботи

- 1. 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 пристрій, приймання телеметрії та надсилання команд.
- 2. 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.
- 3. 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 JM X
app.py
1  from counterfit_connection import CounterFitConnection
2  import time
3  from counterfit_shims_grove.adc import ADC
4  from counterfit_shims_grove.grove_relay import GroveRelay
5  import json
6  from azure.iot.device import IoTHubDeviceClient, Message, MethodResponse
7
8 connection_string = "HostName=soil-moisture-sensor-max11189.azure-devices.net;DeviceId=soil-moisture-sensor;SharedAccessKey=+6A1CAZkhA13cxC90K0lZWPoAy2znZN1z0S0qJ+0dbw="
9
10 device_client = IoTHubDeviceClient.create_from_connection_string(connection_string)
11 print('Connecting')
12 device_client.connect()
13 print('Connected')
14
15
16 CounterFitConnection.init('127.0.0.1', 5001)
17 adc = ADC()
18 relay = GroveRelay(66)
19
20
21 def handle_command(client, userdata, message):
22     payload = json.loads(message.payload.decode())
23     print("Command received:", payload)
24
25     if payload.get('relay_on'):
26         print("Turning relay ON")
27         relay.on()
28     else:
29         print("Turning relay OFF")
30         relay.off()
31
32 def handle_method_request(request):
33     print("Direct method received - ", request.name)
34     if request.name == "relay_on":
35         print("> Turning relay ON")
36         relay.on()
37     elif request.name == "relay_off":
38         print("> Turning relay OFF")
39         relay.off()
40
41     method_response = MethodResponse.create_from_method_request(request, 200)
42     device_client.send_method_response(method_response)
43
44 device_client.on_method_request_received = handle_method_request
45
46 while True:
47     soil_moisture = adc.read(65)
48     print("Soil moisture:", soil_moisture)
49     telemetry = json.dumps({'soil_moisture': soil_moisture})
50     print("Sending telemetry:", telemetry)
51     message = Message(json.dumps({'soil_moisture': soil_moisture}))
52     device_client.send_message(message)
53     time.sleep(10)
```

Код функцій:

```

import azure.functions as func
import logging

import json
import os
from azure.iot.hub import IoTHubRegistryManager
from azure.iot.hub.models import CloudToDeviceMethod

app = func.FunctionApp()

@app.event_hub_message_trigger(arg_name="azeventhub",
                                event_hub_name="iothub-ehub-soil-moist-66140085-c8e8ee0eee",
                                connection="EventHubConnectionString")
def eventhub_trigger(azeventhub: func.EventHubEvent):
    body = json.loads(azeventhub.get_body().decode('utf-8'))
    device_id = azeventhub.iothub_metadata['connection-device-id']
    logging.info(f'Received message: {body} from {device_id}')

    soil_moisture = body['soil_moisture']
    if soil_moisture > 238:
        direct_method = CloudToDeviceMethod(method_name='relay_on', payload='{}')
    else:
        direct_method = CloudToDeviceMethod(method_name='relay_off', payload='{}')

    logging.info(f'Sending direct method request for {direct_method.method_name} for device {device_id}')

    registry_manager_connection_string = os.environ['REGISTRY_MANAGER_CONNECTION_STRING']
    registry_manager = IoTHubRegistryManager(registry_manager_connection_string)
    registry_manager.invoke_device_method(device_id, direct_method)

    logging.info('Direct method request sent!')

```

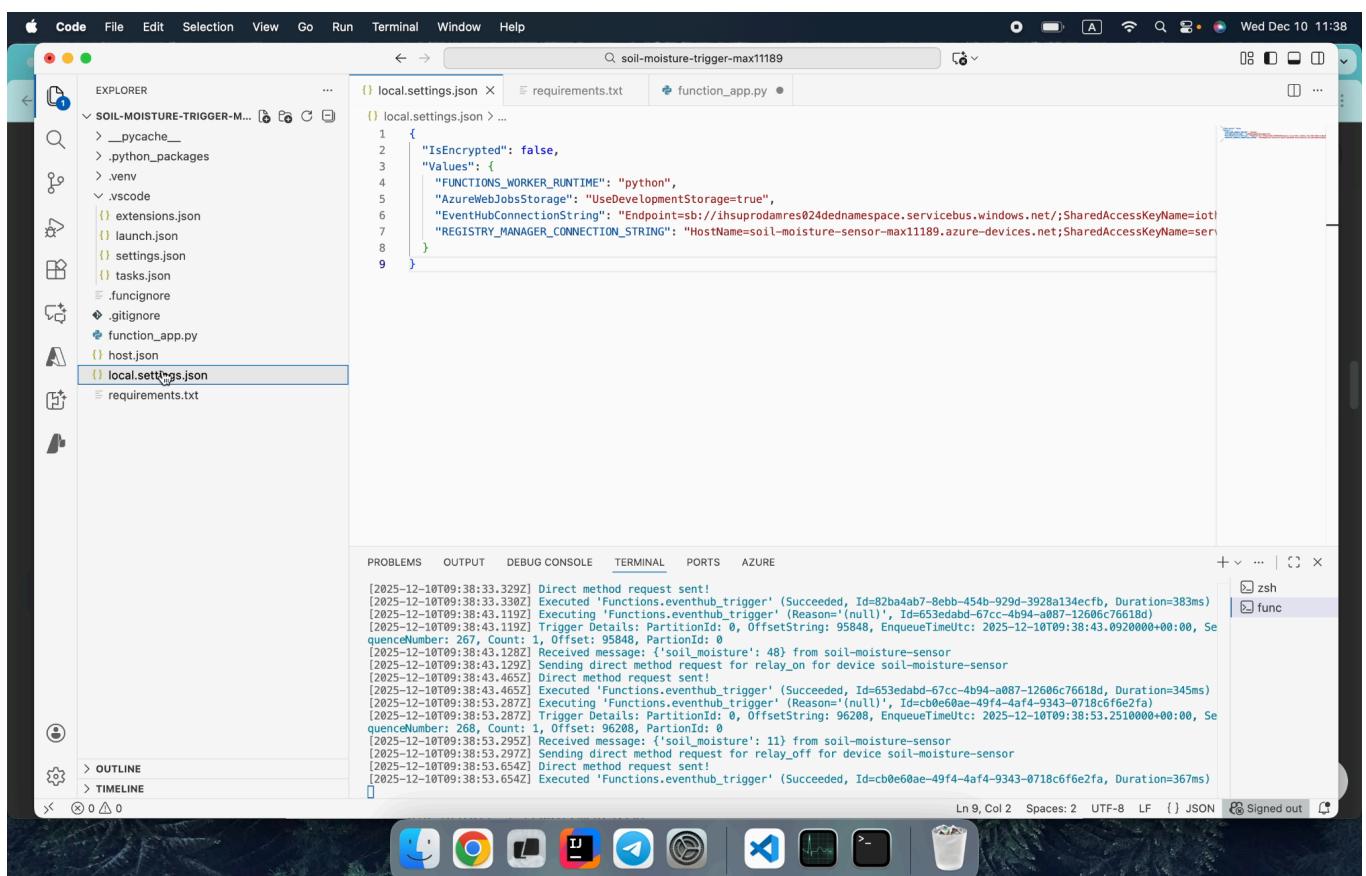


Рисунок 1.1 - Локальна конфігурація Azure Function та журнали успішного виклику direct method для пристроя.

The screenshot shows a Mac OS X desktop environment. A terminal window is open with the title "soil-moisture-trigger-max11189". The terminal output displays logs from an Azure Function named "Functions.eventhub_trigger". The logs show multiple executions of the function, each triggered by a message from an event hub. The logs include details such as the trigger reason ("soil_moisture"), the partition ID, offset, and enqueue time. The code editor in the background shows the "requirements.txt" file and the "function_app.py" file, which contains Python code for the Azure Function.

Рисунок 1.2 - Моніторинг подій у консолі та значення датчика з інтерфейсу CounterFit під час роботи функції.

The screenshot shows a web browser window with multiple tabs. The active tab is titled "CounterFit". The page displays a "Fake IoT Hardware" configuration section where users can set up a sensor. It includes fields for "Units" (with a dropdown menu), "Pin" (set to 0), and a "Value range" slider from 0 to 1,023. A value of 47 is currently selected. Below these are "Random" and "Min/Max" settings, with "Min" set to 9 and "Max" set to 70. There are "Set" and "Delete" buttons. Below this section is a large "Actuators" heading. Under "Actuators", there is a "Create actuator" form for an "LED" on "Pin 0" and a "Relay" component labeled "Pin 66" with a schematic diagram. There are "Add" and "Delete" buttons for the actuators. The browser's address bar shows "localhost:5001".

Рисунок 1.2 - Інтерфейс CounterFit із сенсором вологості та актуатором Relay.

```

import json
import os
from azure.iot.hub import IoTHubRegistryManager
from azure.iot.hub.models import CloudToDeviceMethod

app = func.FunctionApp()

@app.event_hub_message_trigger(arg_name="azeventhub",
                                 event_hub_name="iothub-ehub-soil-moist-66140085-c0e0ee0eee",
                                 connection="EventhubConnectionString")
def eventhub_trigger(azeventhub: func.EventHubEvent):
    body = json.loads(azeventhub.get_body().decode('utf-8'))
    device_id = azeventhub.iothub_metadata['connection-device-id']
    logging.info(f'Received message: {body} from {device_id}')

    soil_moisture = body['soil_moisture']
    if soil_moisture > 40:
        direct_method = CloudToDeviceMethod(method_name='relay_on', payload='{}')
    else:
        direct_method = CloudToDeviceMethod(method_name='relay_off', payload='{}')

    logging.info(f'Sending direct method request for {direct_method.method_name} for device {device_id}')

    registry_manager_connection_string = os.environ['REGISTRY_MANAGER_CONNECTION_STRING']
    registry_manager = IoTHubRegistryManager(registry_manager_connection_string)
    registry_manager.invoke_device_method(device_id, direct_method)

    logging.info('Direct method request sent!')

```

Рисунок 1.3 - Редагування function_app.py у VS Code та відображення логів виконання direct-method.

Службы Azure

- Создать ресурс
- Центр быстрого...
- Azure AI Foundry
- Службы Kubernetes
- Виртуальные машины
- Службы приложений
- Учетные записи...
- Базы данных SQL

Ресурсы

Недавние	Избранное	
soil-moisture-sensor-max11189	Центр Интернета вещей	Последний просмотр 8 мин назад
5a2d35e8-482c-4c05-bf15-8e20f053cca7	Группа ресурсов	29 мин назад

Перейти

- Подписки
- Группы ресурсов
- Все ресурсы
- Панель мониторинга

Средства

Рисунок 1.4 - Панель Azure Portal із відкритим ресурсом IoT Hub.

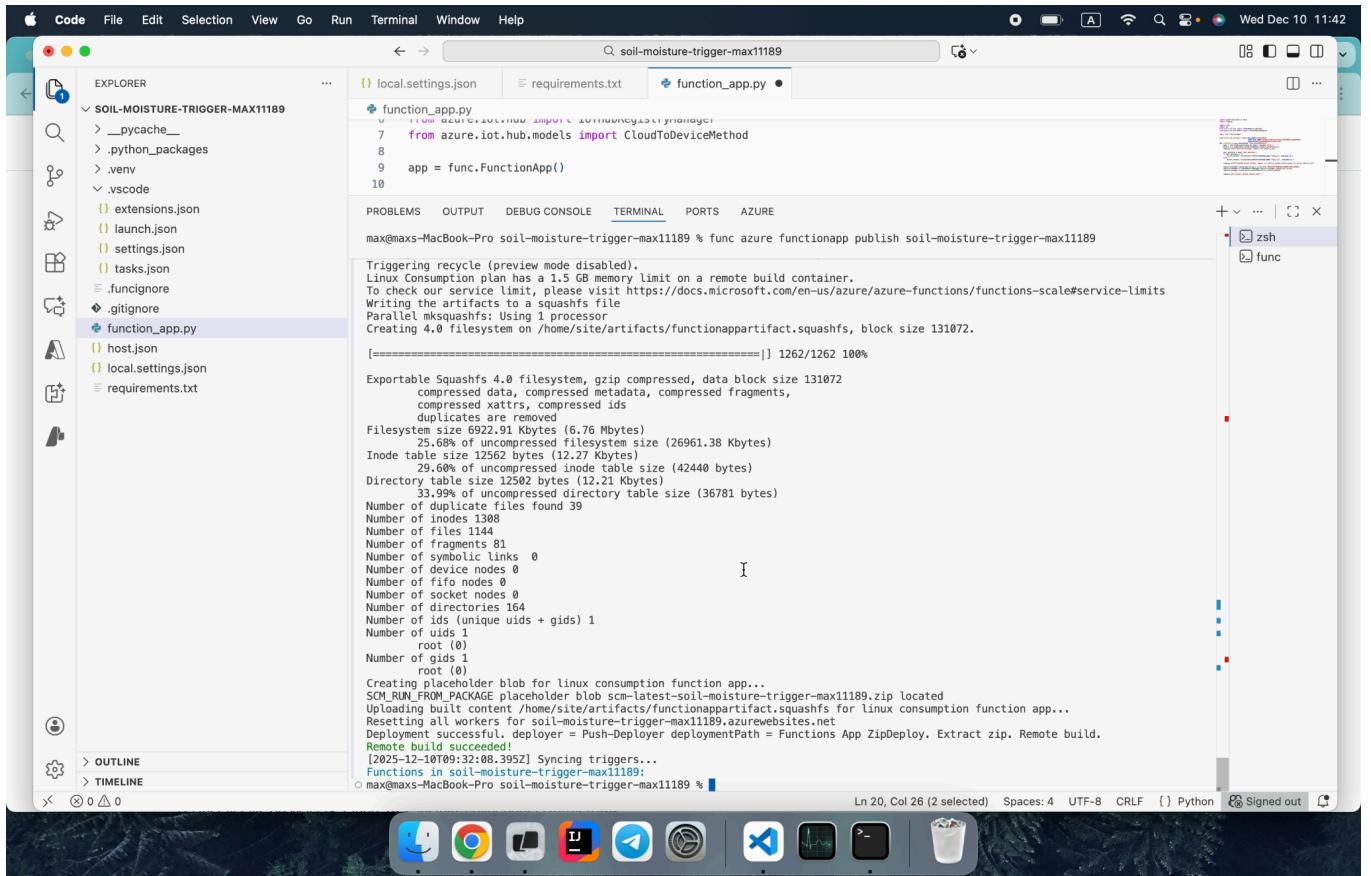


Рисунок 1.5 - Процес публікації Azure Function у терміналі та повідомлення про невідповідність Python-версії.

2. Результат роботи:

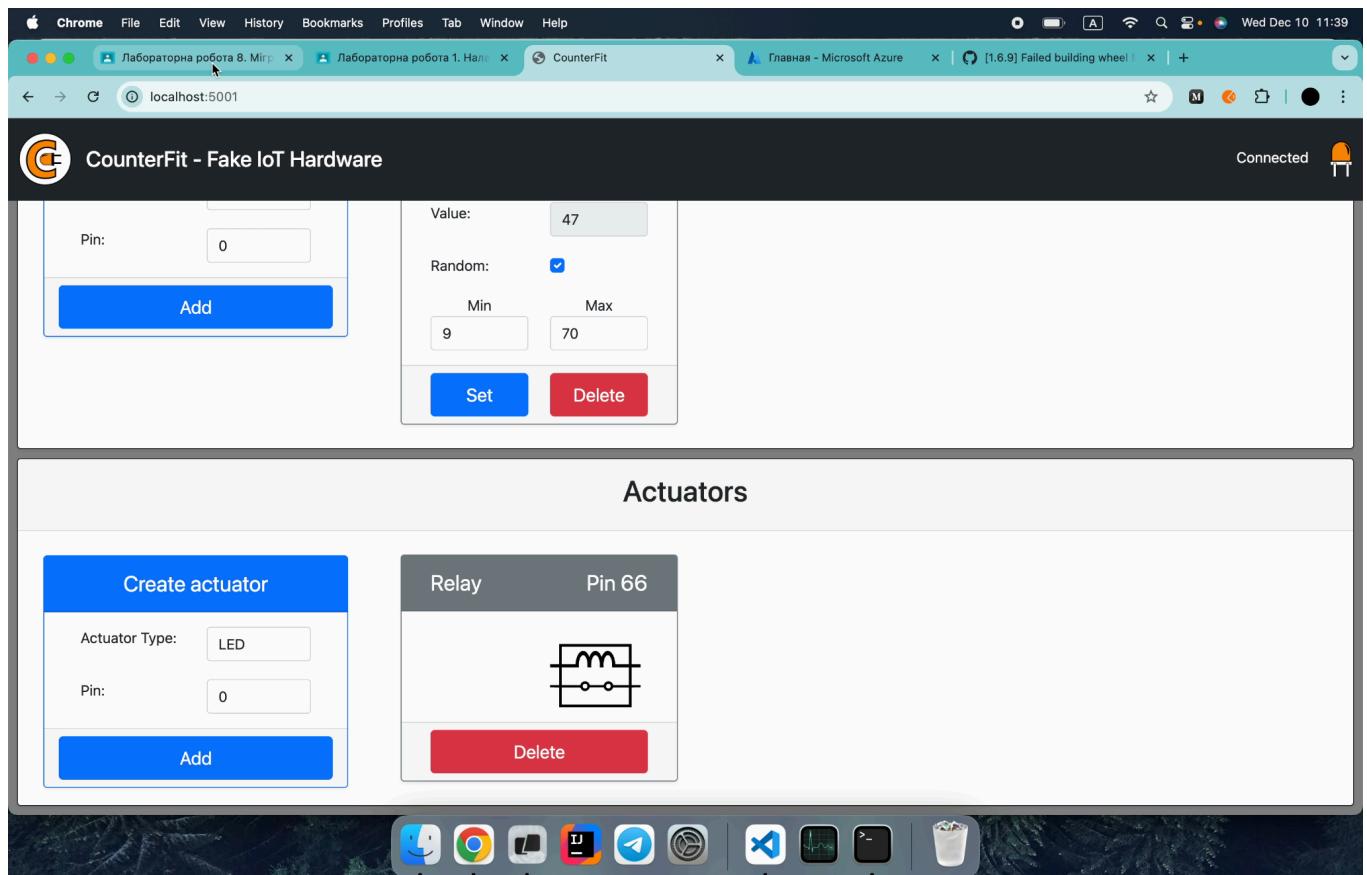
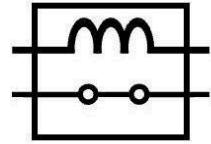


Рисунок 2.1 - інтерфейс Counterfit

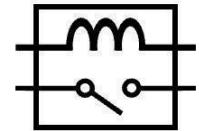
Relay Pin 66



Delete

Рисунок 2.2 - стан реле “замкнутий”

Relay Pin 66



Delete

Рисунок 2.3 - стан реле “розімкнутий”

Посилання на github: <https://github.com/makszaranik/IOT-8>

Висновок:

У ході виконання лабораторної роботи №8 було отримано практичні навички розроблення та розгортання безсерверних функцій у середовищі Microsoft Azure Functions, а також налаштування подієво-орієнтованої обробки телеметрії, що надходить від IoT-пристроїв через Azure IoT Hub. Локальна інфраструктура була розгорнута із використанням емулятора CounterFit, що дало можливість тестувати сценарії роботи IoT-пристроїв без реального фізичного обладнання.

Було створено функцію з Event Hub Trigger, яка автоматично активується при надходженні повідомлення від пристрою. На основі телеметрії датчика вологості система виконує обробку отриманих даних та приймає рішення щодо подальших дій. Для взаємодії з виконавчим пристроєм реалізовано виклики Cloud-to-Device Direct Method, за допомогою яких реле (актуатор) може бути увімкнено або вимкнено безпосередньо з хмари.