

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 IA-33

Заранік Максим

Перевірив:

асистент кафедри ICT

Головатенко І. А.

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 пристройів, 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
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=+6A1CAZkhiAl3cxC90KNZWPoAy2znZN1z0S0qJ+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 серверної частини:

```
app.py
1 import json
2 import time
3 import paho.mqtt.client as mqtt
4 import threading
5
6 id = '9e9e2074-64b1-4dd2-8ee3-ad2ab0359a77'
7 client_telemetry_topic = id + '/telemetry'
8 server_command_topic = id + '/command'
9 client_name = id + 'soil_moisture_sensor_server'
10
11 mqtt_client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2, client_name)
12 mqtt_client.connect('test.mosquitto.org')
13 mqtt_client.loop_start()
14
15 water_time = 5
16 wait_time = 20
17
18 def send_relay_command(client, state):
19     command = {'relay_on': state}
20     print("Sending message:", command)
21     client.publish(server_command_topic, json.dumps(command))
22
23 def control_relay(client):
24     mqtt_client.unsubscribe(client_telemetry_topic)
25     send_relay_command(client, True)
26     time.sleep(water_time)
27     send_relay_command(client, False)
28     time.sleep(wait_time)
29     mqtt_client.subscribe(client_telemetry_topic)
30
31 def handle_telemetry(client, userdata, message):
32     payload = json.loads(message.payload.decode())
33     print("Message received:", payload)
34     if payload['soil_moisture'] > 486:
35         threading.Thread(target=control_relay, args=(client,)).start()
36
37 mqtt_client.subscribe(client_telemetry_topic)
38 mqtt_client.on_message = handle_telemetry
39
40 while True:
41     time.sleep(2)
42
```

Результат роботи:

A screenshot of a Mac OS X desktop. At the top, there's a Dock with various icons. Above the Dock, a browser window is open with multiple tabs. The active tab shows a PowerShell session with some JSON data and command-line output related to an IoT hub. The session includes commands like `az iot hub invoke-device-method` and outputs about soil moisture levels and relay status.

```
origin": "soil-moisture-sensor",
"module": "",
"interface": "",
"component": "",
"payload": "{\"soil_moisture\": 318}"
}
{
"event": {
"origin": "soil-moisture-sensor",
"module": "",
"interface": "",
"component": "",
"payload": "{\"soil_moisture\": 280}"
}
}
^CStopping event monitor...
zaranikxz [ ~ ]$ az iot hub invoke-device-method --device-id soil-moisture-sensor \
--method-name relay_on \
--method-payload '{}' \
--hub-name soil-moisture-sensor-max11189
Error occurred in request., ReadTimeout: HTTPSConnectionPool(host='soil-moisture-sensor-max11189.azure-devices.net', port=443): Read timed out. (read timeout=30)
zaranikxz [ ~ ]$ az iot hub invoke-device-method --device-id soil-moisture-sensor --method-name relay_on --method-payload '{}' --hub-name soil-moisture-sensor-max11189
{
"payload": null,
"status": 200
```

A screenshot of a web-based application titled "CounterFit - Fake IoT Hardware". The interface is divided into two main sections: "Sensors" and "Actuators".

Sensors Section: This section contains a form for adding a sensor. The fields include "Sensor Type" (set to "Button"), "Units" (set to "NoUnits"), "Value range" (set to "0 to 1,023"), "Value" (set to "328"), and "Random" (checkbox checked). Below these are "Min" (234) and "Max" (345) sliders. At the bottom are "Set" and "Delete" buttons.

Actuators Section: This section contains a form for creating an actuator. The fields include "Actuator Type" (set to "LED"), "Pin" (set to "0"), and an "Add" button.

Relay Actuator Detail: A detailed view of a "Relay" actuator connected to "Pin 66". It shows a schematic symbol for a relay coil and contacts.

The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'CounterFit' and displays a task details page from 'classroom.google.com'. The page is for a task named 'Лабораторна робота 7.' under the course 'Інженерія систем IoT 2025'. The task instructions are in Ukrainian and involve using the Azure CLI to invoke a method on an IoT device. A code snippet is provided:

```
az iot hub invoke-device-method --device-id soil-moisture-sensor \
--method-name relay_on \
--method-payload '{}' \
--hub-name <hub_name>
```

Below the code, there is a note: 'Замініть <hub_name> на ім'я, яке ви використовували для свого IoT Hub.'

The task also includes a note about direct methods and a screenshot of a terminal window showing the output: 'Direct method received - relay_on'.

Рисунок 1.1 - інтерфейс Counterfit

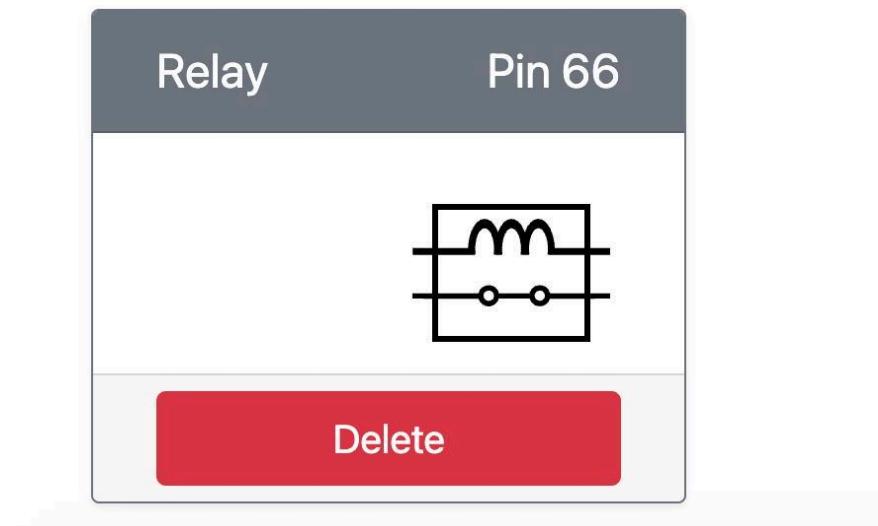


Рисунок 1.2 - стан реле “замкнутий”

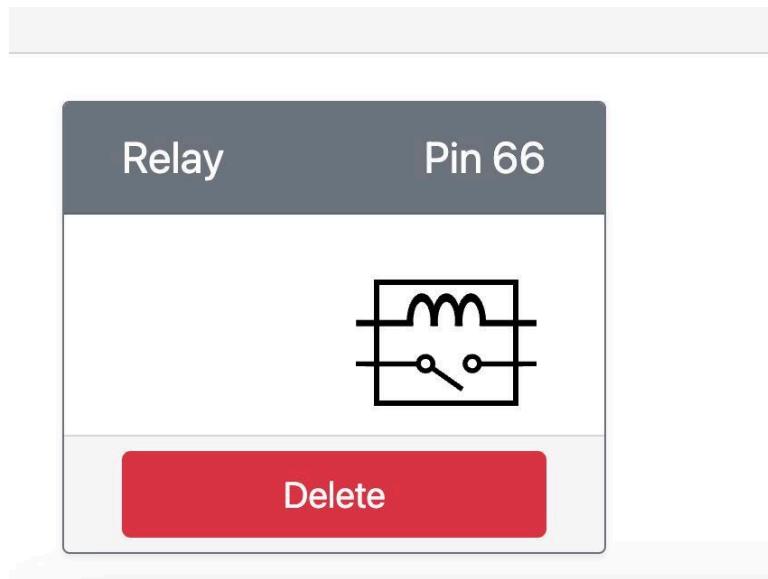


Рисунок 1.3 - стан реле “розімкнущий”

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 пристройв без фізичного обладнання. У середовищі CounterFit було сконфігурковано 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.