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

Лабораторна робота №4

з дисципліни «Інженерія систем IoT»

Тема: «Протокол MQTT»

Варіант: 65

Виконав:
студент групи ІА-33
Заранік Максим

Перевірив:
асистент кафедри ІСТ
Головатенко І. А.

Київ 2025

Мета:Метою даної лабораторної роботи є підключення нічника IoT до Інтернету для забезпечення можливості дистанційного керування. В подальшій частині роботи пристрій IoT надішле телеметричне повідомлення через протокол MQTT до загальнодоступного брокера MQTT із даними про рівень освітленості, де вони будуть оброблені спеціально написаним серверним кодом. Цей код перевірить рівень освітленості та надішле команду назад до пристрою з вказівкою увімкнути або вимкнути світлодіод.

Хід роботи

1. Підготовка середовища та віртуального IoT-пристрою

1. Налаштування CounterFit: Було запущено програму CounterFit для віртуалізації IoT-пристрою та його периферійних пристроїв.
2. Додавання компонентів:
    - Додано датчик освітленості (Light Sensor), підключений до віртуального порту A0 (аналоговий вхід). Це дозволило симулювати зчитування рівня освітленості.
    - Додано виконавчий механізм – світлодіод (LED), підключений до віртуального порту 66 (цифровий вихід). Це була мета додавання виконавчого механізму, згадана у висновку.
3. Встановлення бібліотек: Переконались, що встановлені необхідні бібліотеки Python: counterfit_shims_grove для взаємодії з віртуальними компонентами та paho-mqtt для роботи з протоколом MQTT.

2. Реалізація локального керування (підготовка)

Хоча основна мета стосувалася MQTT, для початкового тестування було реалізовано локальний код для перевірки зчитування даних і керування світлодіодом (як зазначено у висновку):

1. Ініціалізація компонентів: У коді Python ініціалізовано об'єкти для датчика освітленості та світлодіода, вказавши їхні відповідні віртуальні порти.
2. Цикл зчитування та керування: Написано нескінченний цикл, який з інтервалом 1 секунда виконував наступні дії (як описано у висновку):
    - Зчитував поточний рівень освітленості з порту A0.
    - Перевіряв рівень освітленості: якщо він був нижчим за встановлений поріг (наприклад, 300 одиниць), світлодіод на порту D5 вмикався (HIGH).
    - В іншому випадку (при достатньому освітленні) світлодіод вимикався (LOW).

3. Тестування: Було проведено ручне тестування шляхом зміни значення освітленості в інтерфейсі CounterFit, підтверджуючи коректну реакцію світлодіода.

3. Налаштування та підключення до брокера MQTT

1. Вибір публічного брокера: Для роботи було обрано загальнодоступний брокер MQTT (наприклад, test.mosquitto.org або broker.hivemq.com).
2. Налаштування топіків: Визначено два ключові топіки (теми) для обміну повідомленнями:
    - Топік для телеметрії (публикація): Пристрій буде публікувати дані освітленості.
    - Топік для команд (підписка): Пристрій буде очікувати команди на ввімкнення/вимкнення
    - Створення MQTT-клієнта: У код пристрою додано об'єкт MQTT-клієнта (з використанням бібліотеки paho-mqtt), встановлено з'єднання з вибраним брокером та виконано підписку на топік команд (.../commands).

4. Реалізація логіки обміну даними через MQTT

1. Функція обробки команд: Створено функцію-колбек on_message, яка викликається при отриманні повідомлення у топіку команд:
    - Функція аналізувала отриманий текст команди (наприклад, 'ON' або 'OFF').
    - Відповідно до команди, світлодіод на порту D5 вмикався або вимикався, реалізуючи дистанційне керування.
2. Публікація телеметрії: Логіка нескінченного циклу (з п. 2) була змінена, щоб використовувати MQTT:
    - Кожні 10 секунд (або інший інтервал) пристрій зчитував рівень освітленості.
    - Зчитане значення публікувалося у вигляді повідомлення в топік телеметрії (.../telemetry), використовуючи протокол MQTT.

5. Розробка та тестування серверного коду (імітація бекенду)

1. Створення серверного клієнта: Розроблено окремий скрипт (імітація серверного коду), який також підключався до брокера MQTT.
2. Підписка на телеметрію: Серверний клієнт підписувався на топік телеметрії пристрою (.../telemetry), щоб отримувати дані про освітленість.
3. Логіка прийняття рішення: У функції обробки повідомлень серверного клієнта реалізовано логіку, відповідно до мети роботи:

- При отриманні повідомлення з рівнем освітленості, сервер перевіряв його.
- Якщо рівень освітленості був низьким, сервер публікував команду 'ON' у топік команд пристрою (.../commands).
- Якщо рівень був високим, публікувалася команда 'OFF'.

4. Комплексне тестування: Проведено тестування повного циклу:
   - Встановлювалося низьке значення освітленості в CounterFit.
   - Пристрій надсилав ці дані через MQTT на сервер.
   - Сервер обробляв дані та надсилав команду 'ON' назад.
   - Пристрій отримував команду та вмикав світлодіод.
   - При зміні освітленості на високе, цикл повторювався, і світлодіод вимикався.

Таким чином, було успішно реалізовано двосторонній зв'язок між віртуальним пристроєм та віддаленим сервером через протокол MQTT, виконавши всі поставлені завдання.

Код клієнтської частини:

Users > max > lab4 > app.py

```python
import paho.mqtt.client as mqtt
from counterfit_connection import CounterFitConnection
import time
import json
from counterfit_shims_grove.grove_light_sensor_v1_2 import GroveLightSensor
from counterfit_shims_grove.grove_led import GroveLed

CounterFitConnection.init('127.0.0.1', 5000)

light_sensor = GroveLightSensor(65)
led = GroveLed(66)

id = 'bee7fbb6-026b-46c4-b1bf-fa6a38e9c717'
client_name = id + 'nightlight_client'
client_telemetry_topic = id + '/telemetry'
server_command_topic = id + '/commands'

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

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

    if payload['led_on']:
        led.on()
    else:
        led.off()

mqtt_client.subscribe(server_command_topic)
mqtt_client.on_message = handle_command

print("MQTT connected!")

while True:
    light = light_sensor.light
    telemetry = json.dumps({'light': light})
    print("Sending telemetry:", telemetry)
    mqtt_client.publish(client_telemetry_topic, telemetry)
```

Код серверної частини:

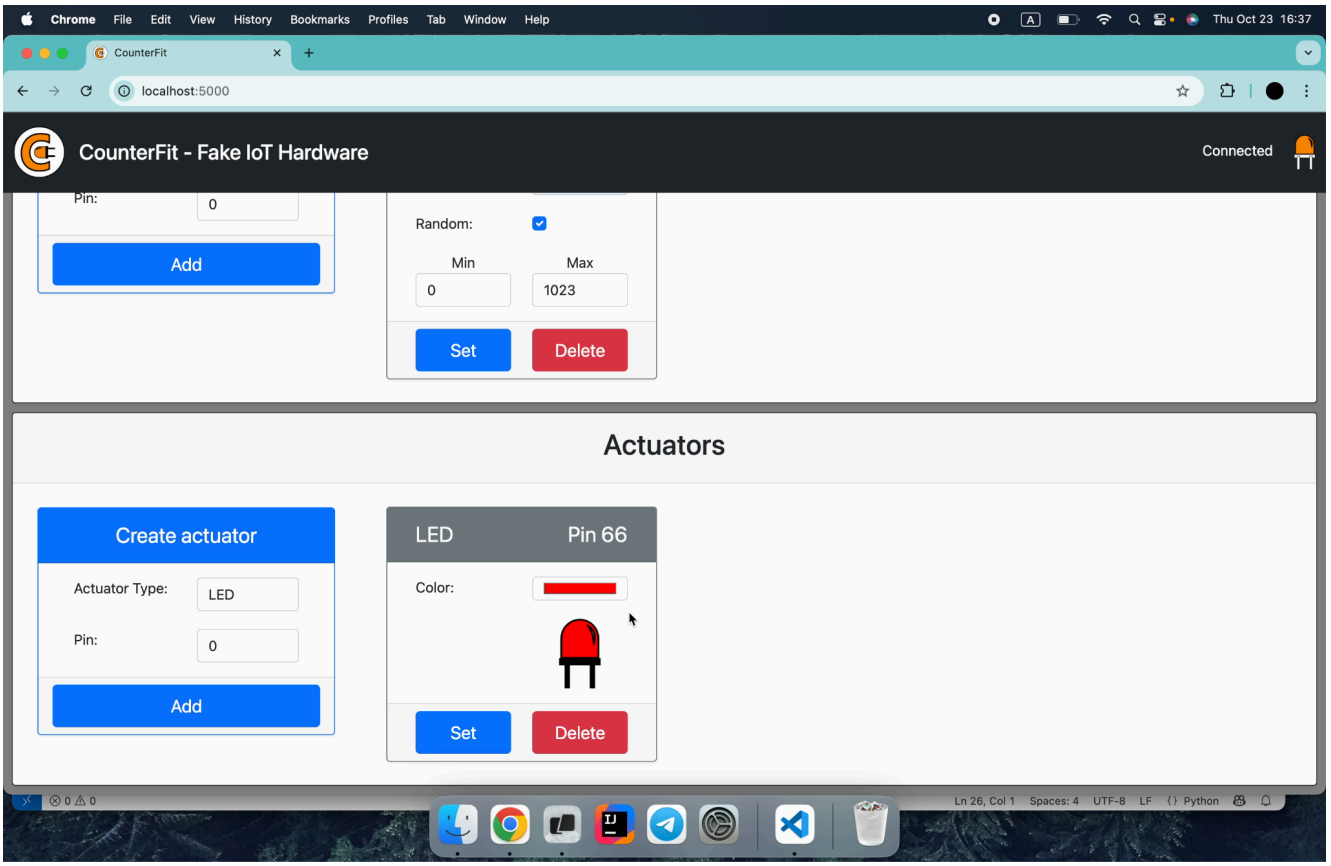
Users > max > lab4 > nightlight-server > app.py

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


id = 'bee7fbb6-026b-46c4-b1bf-fa6a38e9c717'

client_telemetry_topic = id + '/telemetry'
server_command_topic = id + '/commands'
client_name = id + 'nightlight_server'

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


def handle_telemetry(client, userdata, message):
    payload = json.loads(message.payload.decode())
    print("Message received:", payload)
    command = {'led_on': payload['light'] < 650}
    print("Sending message:", command)
    client.publish(server_command_topic, json.dumps(command))

mqtt_client.subscribe(client_telemetry_topic)
mqtt_client.on_message = handle_telemetry

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

mosquitto conf

```
listener 1883 0.0.0.0
allow_anonymous true
```

Результат роботи:



Посилання на github: https://github.com/makszaranik/IOT-4

Висновок:

У ході лабораторної роботи №4 було успішно реалізовано повноцінну систему дистанційного керування віртуальним IoT-нічником на базі протоколу MQTT та симулятора CounterFit. Використання загальнодоступного тестового брокера Eclipse Mosquitto дозволило швидко встановити з'єднання. Був налаштований двосторонній обмін даними: пристрій успішно публікував дані телеметрії (рівень освітленості) у відповідний топік. Спеціально написаний серверний код підписувався на телеметрію, аналізував її та, приймаючи рішення, публікував команду ("ON" або "OFF") у топік команд пристрою. Пристрій, підписаний на топік команд, отримував інструкцію та дистанційно вмикав або вимикав світлодіодний виконавчий механізм. Таким чином, було продемонстровано ключові принципи роботи протоколу MQTT, включаючи механізми "Publish/Subscribe".