



CPE314 Project I

จัดทำโดย

นาย กิติพัฒน์ เรืองอมรวัฒน์ 63070501006

นาย ชลากร วงษ์ประดิษฐ์ 63070501015

นาย ธนสิทธิ์ โภคพูล 63070501031

นาย นันทมนัส ตั้งประเสริฐ 63070501040

นาย สัณห์ฤทธิ์ พรหมจรรย์ 63070501069

นาย สิริวิชัย ศาสนกุล 63070501072

เสนอ

รศ.ดร.พีรพล ศิริพงษ์วุฒิกกร

รศ.ดร.อัครรัตน์ อมรรักษา

โครงการชิ้นนี้เป็นส่วนหนึ่งของวิชา Computer Networks

ภาคเรียนที่ 2 ปีการศึกษา 2565

คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมคอมพิวเตอร์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

โปรแกรมที่เราเขียนขึ้นมาเป็นการจำลองการทำงานของ Network MQTT(MQ Transmety Transport) ซึ่งเป็น Protocol บน Application Layer ที่ได้รับความนิยมในการนำมาใช้กับ Internet of Things application

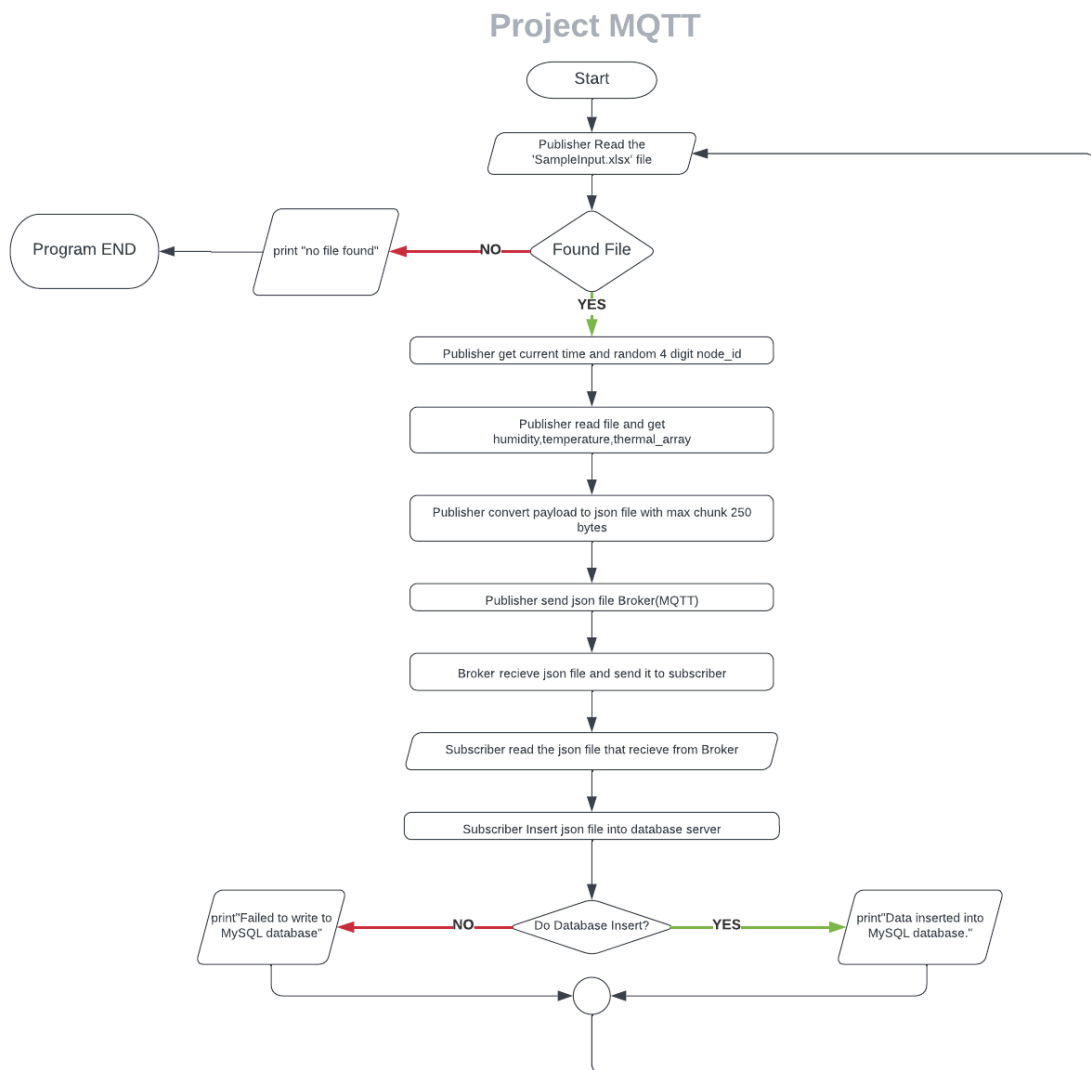
ในโครงงานของเราจะมีองค์ประกอบหลักๆอยู่ 3 ส่วนด้วยกันคือ IoT node(Publisher), Broker, Database Server(Subscriber)

เริ่มโดยการเขียนโค้ดเพื่อจำลอง IoT node ขึ้นมาเพื่อส่ง data ให้ทาง broker โดยให้อ่านข้อมูลจากไฟล์ .xlsx แทนการวัดค่าจริง และสร้าง MQTT Broker ขึ้นมา โดยเครื่องมือที่เราใช้สร้างคือ hivemq และสุดท้ายคือการสร้าง Database Server ขึ้นมาเพื่อเป็น subscriber รับข้อมูลจาก broker ซึ่งทั้งหมดจะถูกจัดการด้วยโค้ด python script

โดยโปรแกรมของเราได้ทำงานอยู่ภายใต้เงื่อนไขต่อไปนี้

- Client จะสามารถส่งข้อความได้ไม่เกิน 250 bytes ต่อการส่ง 1 ครั้ง
- ระบบสามารถรองรับ IoT node และ Database server ได้หลายอัน
- เมื่อมี IoT node(Publisher) และ Database server(Subscriber) เชื่อมต่อมายังระบบ broker จะแสดง IP Address นั้นๆขึ้นมา
- Broker แสดงข้อความที่ส่งมาจาก node ขึ้นบนหน้าจอ
- Database server แสดงข้อความที่ได้รับจาก Broker ขึ้นบนหน้าจอ
- Server สามารถ query data ได้

แผนผังภาพรวมการทำงานของโปรแกรม (Flowcharts)



Client(Publisher)

ในที่นี้คือ IoT node ที่ทำหน้าที่เป็น publisher ส่ง data ไปที่ broker ในส่วนนี้เราได้จำลอง node ขึ้นมา เราใช้โค้ดภาษา python โดยมีการใช้ library ดังนี้

```
import pandas as pd
import paho.mqtt.client as mqtt
import time as times
from datetime import datetime
import random
```

- ใช้ library pandas เพื่ออ่านไฟล์ .xlsx และเก็บไว้ ซึ่งจะเป็นข้อมูลที่จะนำไปใช้ publish

```
# Read sensor data from Excel file
excel_file = ".\Sampleinput.xlsx"
data = pd.read_excel(excel_file, index_col=None)
```

- กำหนดค่าต่างๆ เพื่อเชื่อมต่อไปยังตัว broker ด้วย method connect ซึ่งในที่นี้คือ hivemq โดยใช้ paho mqtt client initialized ตัว client ขึ้นมา

```
# Define MQTT broker settings
broker_address = "broker.hivemq.com"
broker_port = 1883
```

```
# Connect to MQTT broker and publish data every 3 minutes
client = mqtt.Client()
client.connect(broker_address, broker_port)
client.loop_start()
while True:
    publish_data(client)
    times.sleep(10)
client.loop_stop()
client.disconnect()
```

- หลังจากเชื่อมต่อไปยัง broker สำเร็จ เราก็สร้าง loop ขึ้นมาเพื่อใช้ส่งข้อความไปยัง broker ด้วยฟังก์ชัน publish_data ที่เขียนขึ้นมา เรื่อยๆตามระยะเวลาที่กำหนดไว้ด้วย times.sleep(ในที่นี้ 3 นาที) โดยฟังก์ชัน publish_data มีการทำงานดังนี้

```
# Publish sensor data to MQTT broker
def publish_data(client):
    for _, row in data.iterrows():
        # Generate random 4-digit node ID
        node_id = str(random.randint(1000, 9999))
        tor_IP = "tor01"
        # Get current time in the format yyyy-mm-dd hh:mm
        current_time = datetime.now().strftime("%Y-%m-%d %H:%M")

        # Create payload as a dictionary with sensor types as keys and values as lists
        payload = {
            "client_ip" : tor_IP,
            "node_id": node_id,
            "time": current_time,
            "relative_humidity": row["Humidity"],
            "temperature": row["Temperature"],
            "thermal_array": row["ThermalArray"]
        }

        # Convert payload to JSON and split into chunks of max 250 bytes
        payload_json = pd.Series(payload).to_json()
        payload_chunks = [payload_json[i:i+250] for i in range(0, len(payload_json), 250)]

        # Publish payload chunks to MQTT broker
        for chunk in payload_chunks:
            client.publish(topic, chunk)
    print("Sensor data published to MQTT broker.")
```

จากรูป ฟังก์ชันจะทำการ generate node_id ขึ้นมาแบบสุ่มโดยใช้ library random รับเวลาปัจจุบันมาโดยใช้ library datetime และนำ node_id และ current_time ที่ได้มาเอาไปรวมกับข้อมูลที่อ่านได้จากไฟล์ xlsx สร้างเป็น payload หลังจากนั้นจึงนำไปแบ่งเป็น chunk chunkละไม่เกิน 250 bytes แล้วจึง publish ขึ้นไปยัง broker หลังจากทำการ publish ข้อความจนหมดบรรทัดแล้วจะแสดงข้อความว่า “Sensor data published to MQTT broker.”

Broker

ในส่วนของ Broker เราใช้งานเป็นของ hivemq เพื่อใช้เป็นตัวเชื่อมรับข้อมูลมาจาก publisher และส่งต่อไปให้ Subscriber



Link : <https://www.hivemq.com>

Servers(Subscriber)

Server หรือ Subscriber ทำหน้าที่เป็นตัวรับข้อมูลจาก Broker แล้วSubscriberจะนำข้อมูลที่ได้รับมา

นำมารวมกันให้เป็นข้อมูลที่สมบูรณ์แล้วจึงนำไปเขียนลงไปใน Database ที่เราสร้างไว้แล้ว

ในส่วนของโค้ด python script ได้มีการใช้ library ดังนี้

```
import paho.mqtt.client as mqtt
import json
import mysql.connector
```

- กำหนดค่าต่างๆ เพื่อเชื่อมต่อไปยังตัว broker ด้วย method connect ซึ่งในที่นี้คือ hivemq โดยใช้ paho mqtt client initialized ตัว client ขึ้นมา

```
# Define MQTT broker settings
broker_address = "broker.hivemq.com"
broker_port = 1883
```

```
# Create MQTT client and set callbacks
client = mqtt.Client()
```

- ฟังก์ชัน on_message มีการทำงานดังนี้ คือ เมื่อรับข้อความมาจาก broker และ จะทำการ decode

ข้อความและเพิ่มไปยัง buffer เมื่อได้รับข้อความจนจบบรรทัดถึงจะเริ่มทำงานต่อ

โดยในที่นี้ได้ใช้วิธีเช็คคือการดูที่ตัวสุดท้ายว่าใช่ “}” ที่เป็นตัวปิดบรรทัดหรือไม่ และจึงใช้ library json ดึงข้อความมาใส่ใน payload และทำการเรียกฟังก์ชัน insert_to_database ขึ้นมาเพื่อทำการบันทึกข้อมูลลง database และจากนั้นจึงเคลียร์ buffer ใหม่เพื่อเตรียมรับข้อความถัดไป

```
# Define on_message callback function
def on_message(client, userdata, message):
    global buffer
    # Append received data to buffer
    buffer += message.payload.decode()
    # Check if buffer contains entire message
    if buffer.endswith("}"):
        # Parse JSON message
        payload = json.loads(buffer)
        # Insert payload data into MySQL database
        insert_to_database(payload)
        # Clear buffer
        buffer = ""
```

- ฟังก์ชัน `insert_to_database` มีการทำงานดังนี้ คือ เริ่มจากการลองเชื่อมต่อไปยัง database ที่กำหนดเอาไว้ แล้วจึงนำ payload ที่รับมาแบ่งและจัดเก็บเข้าไปยังแต่ละ column โดยการ query เมื่อทำสำเร็จจะแสดงข้อความ "Data inserted into MySQL database." แต่ถ้าเกิด error ขึ้นมา จะแสดงข้อความ "Failed to write to MySQL database."

```
def insert_to_database(payload):  
    try:  
        connection = mysql.connector.connect(  
            host="localhost",  
            user="root",  
            password="",  
            database="input_sensor_server"  
        )  
  
        with connection.cursor() as cursor:  
            sql = "INSERT INTO input_sensor (client,NodeID, Time, Humidity, Temperature, ThermalArray) VALUES (%s,%s, NOW(), %s, %s, %s)"  
            values = (  
                payload["client_ip"],  
                payload["node_id"],  
                payload["relative_humidity"],  
                payload["temperature"],  
                payload["thermal_array"]  
            )  
            cursor.execute(sql, values)  
            connection.commit()  
            print("Data inserted into MySQL database.")  
    except Exception as e:  
        print(f"Failed to write to MySQL database: {e}")  
    finally:  
        connection.close()
```

- ฟังก์ชัน `on_connect` จะมีการทำงานดังนี้ คือ ถ้าเมื่อสามารถเชื่อมต่อไปยัง broker ได้สำเร็จ จะแสดงข้อความว่า "Connected to MQTT broker with result code 0" และทำการ subscribe ไปยัง topic ที่ต้องการ

```
# Define on_connect callback function  
def on_connect(client, userdata, flags, rc):  
    print("Connected to MQTT broker with result code "+str(rc))  
    # Subscribe to topic  
    client.subscribe(topic)
```

- โปรแกรมของเราจะเรียกใช้ `on_connect` และ `on_message` ขึ้นมาก่อนแล้วจึงทำการเชื่อมต่อไปยัง broker และทำการ loop เพื่อเชื่อมต่อกับ broker ไปเรื่อยๆ

- ถ้าการเชื่อมต่อสำเร็จก็จะมี callback กลับมาจากฟังก์ชัน `on_connect` และเมื่อมีข้อความส่งมาก็จะมี callback กลับมาจากฟังก์ชัน `on_message`

```
# Create MQTT client and set callbacks  
client = mqtt.Client()  
client.on_connect = on_connect  
client.on_message = on_message  
  
# Connect to MQTT broker and start loop  
client.connect(broker_address, broker_port)  
client.loop_forever()
```


Database

ใน Database นี้คือ SQL Dump ที่สามารถใช้สร้างฐานข้อมูลและโครงสร้างตารางสำหรับระบบที่เก็บข้อมูล Input Sensor โดยไฟล์ SQL dump นี้สร้างขึ้นโดย phpMyAdmin เวอร์ชัน 5.1.1 และสร้างขึ้นเมื่อวันที่ 5 มีนาคม 2023 เวลา 11:59 น. เวอร์ชันเซิร์ฟเวอร์ MySQL ที่ใช้คือ 10.4.22-MariaDB และเวอร์ชัน PHP ที่ใช้คือ 7.4.27

```
1  -- phpMyAdmin SQL Dump
2  -- version 5.1.1
3  -- https://www.phpmyadmin.net/
4  --
5  -- Host: 127.0.0.1
6  -- Generation Time: Mar 05, 2023 at 11:59 AM
7  -- Server version: 10.4.22-MariaDB
8  -- PHP Version: 7.4.27
9
10 SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
11 START TRANSACTION;
12 SET time_zone = "+00:00";
13
```

สร้าง Database มีชื่อว่า input_sensor_server และมี 1 ตารางชื่อ input_sensor โดยในตารางมี 6 คอลัมน์: client, NodeID, Time, Humidity, Temperature และ ThermalArray


1. คอลัมน์ client เป็นประเภทข้อความและไม่อนุญาตให้เป็นค่า NULL อาจมีสตริงที่ระบุ Client ที่ส่งข้อมูล
2. คอลัมน์ NodeID เป็นประเภท int และไม่อนุญาตให้เป็นค่า NULL อาจมีจำนวนเต็มที่ระบุ Node ที่ส่งข้อมูล
3. คอลัมน์ Time เป็นประเภทวันที่และไม่อนุญาตให้เป็นค่า NULL อาจมีการประทับเวลาที่ระบุเวลาที่ข้อมูลถูกส่ง
4. คอลัมน์ Humidity, Temperature และ ThermalArray เป็นประเภท double ทั้งหมดและไม่อนุญาตให้เป็นค่า NULL

โดยอาจมีข้อมูลตัวเลขที่แสดงถึงการอ่าน Sensor สำหรับ Humidity Temperature และ ThermalArray ตามลำดับ มี index ใน คอลัมน์ NodeID ซึ่งถูกกำหนดให้เป็น Primary Key ของตาราง

```
30 CREATE TABLE `input_sensor` (
31   `client` text NOT NULL,
32   `NodeID` int(11) NOT NULL,
33   `Time` date NOT NULL,
34   `Humidity` double NOT NULL,
35   `Temperature` double NOT NULL,
36   `ThermalArray` double NOT NULL
37 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
38
39 --
40 -- Indexes for dumped tables
41 --
42 --
43 -- Indexes for table `input_sensor`
44 --
45 --
46 ALTER TABLE `input_sensor`
47   ADD PRIMARY KEY (`NodeID`);
48 COMMIT;
49
50 /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
51 /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
52 /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
53
```

DATABASE

Database ที่ใช้ในโครงการครั้งนี้คือ MySQL โดยมี Structure ดังนี้

#	Name	Type
1	client	text
2	NodeID 	int(11)
3	Time	date
4	Humidity	double
5	Temperature	double
6	ThermalArray	double

1. client (text) ชื่อของ client ที่ส่ง data เข้ามา

2. NodeID(PK)(int) node id ของแต่ละการส่ง

3. Time(date) วันที่ที่ทำการรับข้อมูลเข้ามา

อีก 3 columns คือค่าที่วัดได้จากเซ็นเซอร์ จัดเก็บเป็น double

4. Humidity(double)

5. Temperature(double)

6. ThermalArray(double)