

# LAPORAN PRAKTIKUM

IoT

MQTT - SIMULATOR

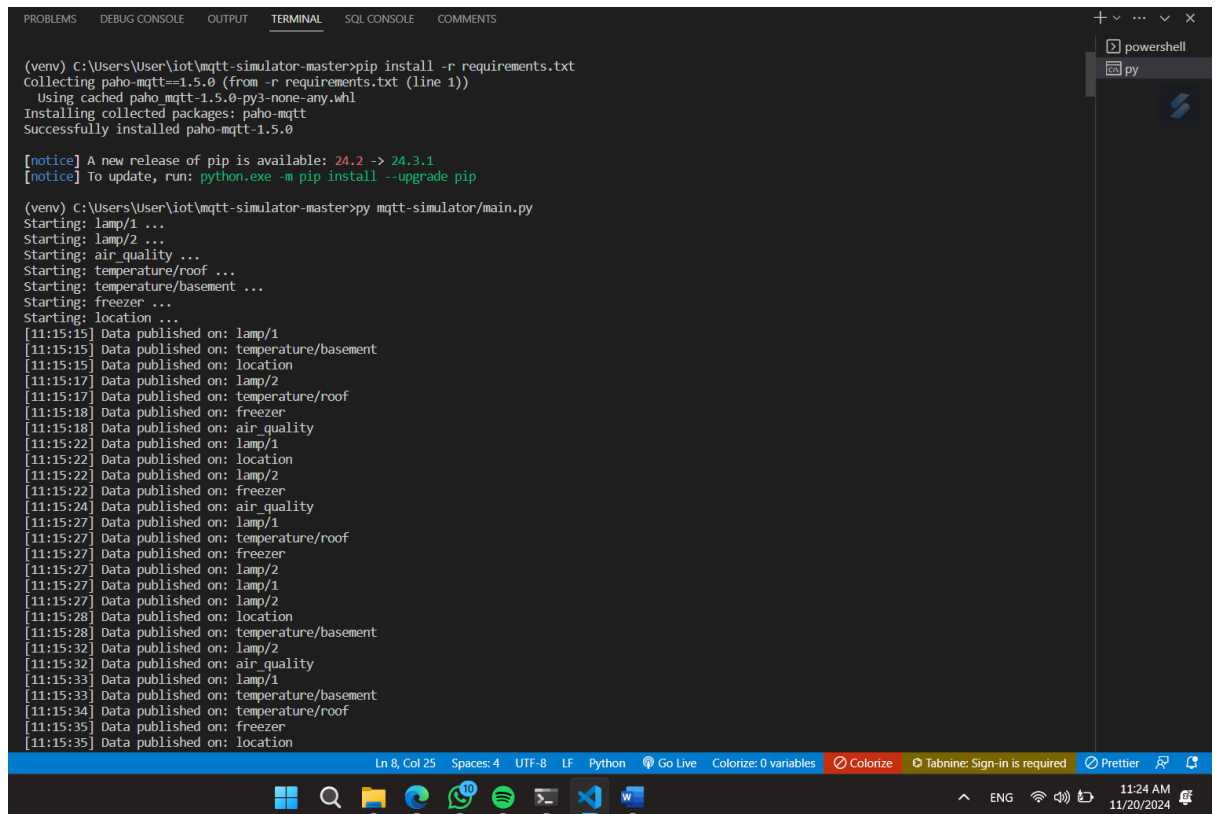


**Marshanda Simangunsong(11323020)**

(D-III Teknologi Informasi)

INSTITUT TEKNOLOGI DEL FAKULTAS VOKASI

## A. Bukti berhasil dijalankan



```
(venv) C:\Users\User\iot\mqtt-simulator-master>pip install -r requirements.txt
Collecting paho-mqtt==1.5.0 (from -r requirements.txt (line 1))
  Using cached paho-mqtt-1.5.0-py3-none-any.whl
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.5.0

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

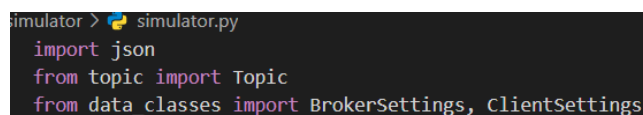
(venv) C:\Users\User\iot\mqtt-simulator-master>py mqtt-simulator/main.py
Starting: lamp/1 ...
Starting: lamp/2 ...
Starting: air_quality ...
Starting: temperature/roof ...
Starting: temperature/basement ...
Starting: freezer ...
Starting: location ...
[11:15:15] Data published on: lamp/1
[11:15:15] Data published on: temperature/basement
[11:15:15] Data published on: location
[11:15:17] Data published on: lamp/2
[11:15:17] Data published on: temperature/roof
[11:15:18] Data published on: freezer
[11:15:18] Data published on: air_quality
[11:15:22] Data published on: lamp/1
[11:15:22] Data published on: location
[11:15:22] Data published on: lamp/2
[11:15:22] Data published on: freezer
[11:15:24] Data published on: air_quality
[11:15:27] Data published on: lamp/1
[11:15:27] Data published on: temperature/roof
[11:15:27] Data published on: freezer
[11:15:27] Data published on: lamp/2
[11:15:27] Data published on: lamp/1
[11:15:27] Data published on: lamp/2
[11:15:28] Data published on: location
[11:15:28] Data published on: temperature/basement
[11:15:32] Data published on: lamp/2
[11:15:32] Data published on: air_quality
[11:15:33] Data published on: lamp/1
[11:15:33] Data published on: temperature/basement
[11:15:34] Data published on: temperature/roof
[11:15:35] Data published on: freezer
[11:15:35] Data published on: location
```

## B. Step by step proses

### Simulator.spy

Simulator.spy adalah sebuah program Python yang digunakan untuk mensimulasikan pengaturan topik MQTT berdasarkan file konfigurasi yang diatur dalam bentuk JSON. Aku akan jelaskan lebih rinci setiap bagian:

#### 1. Mengimpor modul dan kelas



```
simulator > simulator.py
import json
from topic import Topic
from data_classes import BrokerSettings, ClientSettings
```

- json digunakan untuk membaca file JSON yang berisi pengaturan simulasi.
- Topic, BrokerSettings, dan ClientSettings adalah kelas yang diimpor dari modul topic dan data\_classes. Kelas-kelas ini digunakan untuk mengelola topik-topik MQTT dan menyimpan pengaturan broker serta klien.

#### 2. Kelas simulator

```
class Simulator:
    def __init__(self, settings_file):
        self.default_client_settings = ClientSettings(
            clean=True,
            retain=False,
            qos=2,
            time_interval=10
        )
        self.topics = self.load_topics(settings_file)
```

- Simulator adalah kelas utama yang bertanggung jawab untuk mensimulasikan topik-topik MQTT berdasarkan pengaturan dari file JSON.
- Di dalam konstruktor (`__init__`):
  1. `self.default_client_settings` menyimpan pengaturan default untuk klien MQTT.
  2. `self.topics` memuat daftar topik dengan memanggil metode `load_topics` dan mengirim file
- 3. Metode `read_client_settings`

```
def read_client_settings(self, settings_dict: dict, default: ClientSettings):
    return ClientSettings(
        clean=settings_dict.get('CLEAN_SESSION', default.clean),
        retain=settings_dict.get('RETAIN', default.retain),
        qos=settings_dict.get('QOS', default.qos),
        time_interval=settings_dict.get('TIME_INTERVAL', default.time_interval)
    )
```

- Metode ini membaca pengaturan klien dari dictionary yang diambil dari file JSON.
- Menggunakan `get()` untuk mengambil nilai dari dictionary, dengan pengaturan default sebagai cadangan jika pengaturan tertentu tidak ditemukan.
- 4. Metode `load_topics`

```
def load_topics(self, settings_file):
    topics = []
    with open(settings_file) as json_file:
        config = json.load(json_file)
        broker_settings = BrokerSettings(
            url=config.get('BROKER_URL', 'localhost'),
            port=config.get('BROKER_PORT', 1883),
            protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311
        )
        broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
```

- Metode ini membaca file pengaturan dan memuat topik-topik ke dalam daftar topics.
- **Langkah-langkahnya:**

1. Membuka file JSON dan memuat isinya menggunakan json.load.
  2. Membuat objek BrokerSettings yang menyimpan URL, port, dan protokol broker.
  3. Membaca pengaturan klien dengan read\_client\_settings untuk setiap broker.
5. Membaca dan membuat topik

```
broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
# read each configured topic
for topic in config['TOPICS']:
    topic_data = topic['DATA']
    topic_payload_root = topic.get('PAYLOAD_ROOT', {})
    topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)
    if topic['TYPE'] == 'single':
        # create single topic with format: //{PREFIX}
        topic_url = topic['PREFIX']
        topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
    elif topic['TYPE'] == 'multiple':
        # create multiple topics with format: //{PREFIX}/{id}
        for id in range(topic['RANGE_START'], topic['RANGE_END']+1):
            topic_url = topic['PREFIX'] + '/' + str(id)
            topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
    elif topic['TYPE'] == 'list':
        # create multiple topics with format: //{PREFIX}/{item}
        for item in topic['LIST']:
            topic_url = topic['PREFIX'] + '/' + str(item)
            topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
return topics
```

- **Looping melalui setiap topik dalam file JSON:**

1. single: Membuat satu topik dengan URL dari PREFIX.
2. multiple: Membuat beberapa topik dengan format PREFIX/{id} berdasarkan rentang yang ditentukan.
3. list: Membuat beberapa topik dari daftar yang diberikan, dengan URL dari PREFIX/{item}.

6. Metode run dan sytop

```
def run(self):
    for topic in self.topics:
        print(f'Starting: {topic.topic_url} ...')
        topic.start()
    for topic in self.topics:
        # workaround for Python 3.12
        topic.join()

def stop(self):
    for topic in self.topics:
        print(f'Stopping: {topic.topic_url} ...')
        topic.stop()
```

- **Menjalankan Simulasi:**  
start() memulai setiap topik di thread-nya sendiri.  
join() memastikan thread selesai sebelum program berlanjut.
- **Menghentikan Simulasi:**  
Memanggil stop() untuk menghentikan setiap topik dan mencetak pesan.

## IMPLEMENTASI TAMBAHAN

### 1. Kelas BrokerSetting

```
class Simulator:
    def __init__(self, settings_file):
        self.default_client_settings = ClientSettings(
            clean=True,
            retain=False,
            qos=2,
            time_interval=10
        )
        self.topics = self.load_topics(settings_file)
```

### 2. Client setting

```
def read_client_settings(self, settings_dict: dict, default: ClientSettings):
    return ClientSettings(
        clean=settings_dict.get('CLEAN_SESSION', default.clean),
        retain=settings_dict.get('RETAIN', default.retain),
        qos=settings_dict.get('QOS', default.qos),
        time_interval=settings_dict.get('TIME_INTERVAL', default.time_interval)
    )
```

### 3. kela topic

```
def load_topics(self, settings_file):
    topics = []
    with open(settings_file) as json_file:
        config = json.load(json_file)
        broker_settings = BrokerSettings(
            url=config.get('BROKER_URL', 'localhost'),
            port=config.get('BROKER_PORT', 1883),
            protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311
        )
        broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
        # read each configured topic
        for topic in config['TOPICS']:
            topic_data = topic['DATA']
            topic_payload_root = topic.get('PAYLOAD_ROOT', {})
            topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)
            if topic['TYPE'] == 'single':
                # create single topic with format: //{PREFIX}
                topic_url = topic['PREFIX']
                topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
            elif topic['TYPE'] == 'multiple':
                # create multiple topics with format: //{PREFIX}/{id}
                for id in range(topic['RANGE_START'], topic['RANGE_END']+1):
                    topic_url = topic['PREFIX'] + '/' + str(id)
                    topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
            elif topic['TYPE'] == 'list':
```

penjelasna :

Dengan penjelasan ini, kau bisa memahami bagaimana setiap bagian dari kode berfungsi, dan

implementasi tambahan untuk kelas BrokerSettings, ClientSettings, dan Topic yang diperlukan agar kode berjalan dengan baik|

kesimpulan :

Kode ini adalah simulator untuk topik-topik MQTT. Ia membaca konfigurasi dari file JSON, mengatur topik-topik berdasarkan pengaturan yang ada, dan kemudian menjalankan atau menghentikan simulasi topik-topik tersebut. Semua topik dijalankan secara paralel menggunakan thread.

**Kegunaan:** Sangat berguna untuk menguji dan mensimulasikan komunikasi menggunakan protokol MQTT, misalnya untuk pengembangan dan debugging aplikasi IoT.