

# LAPORAN PRAKTIKUM

## Internet Of Things



Romian A. Tambunan

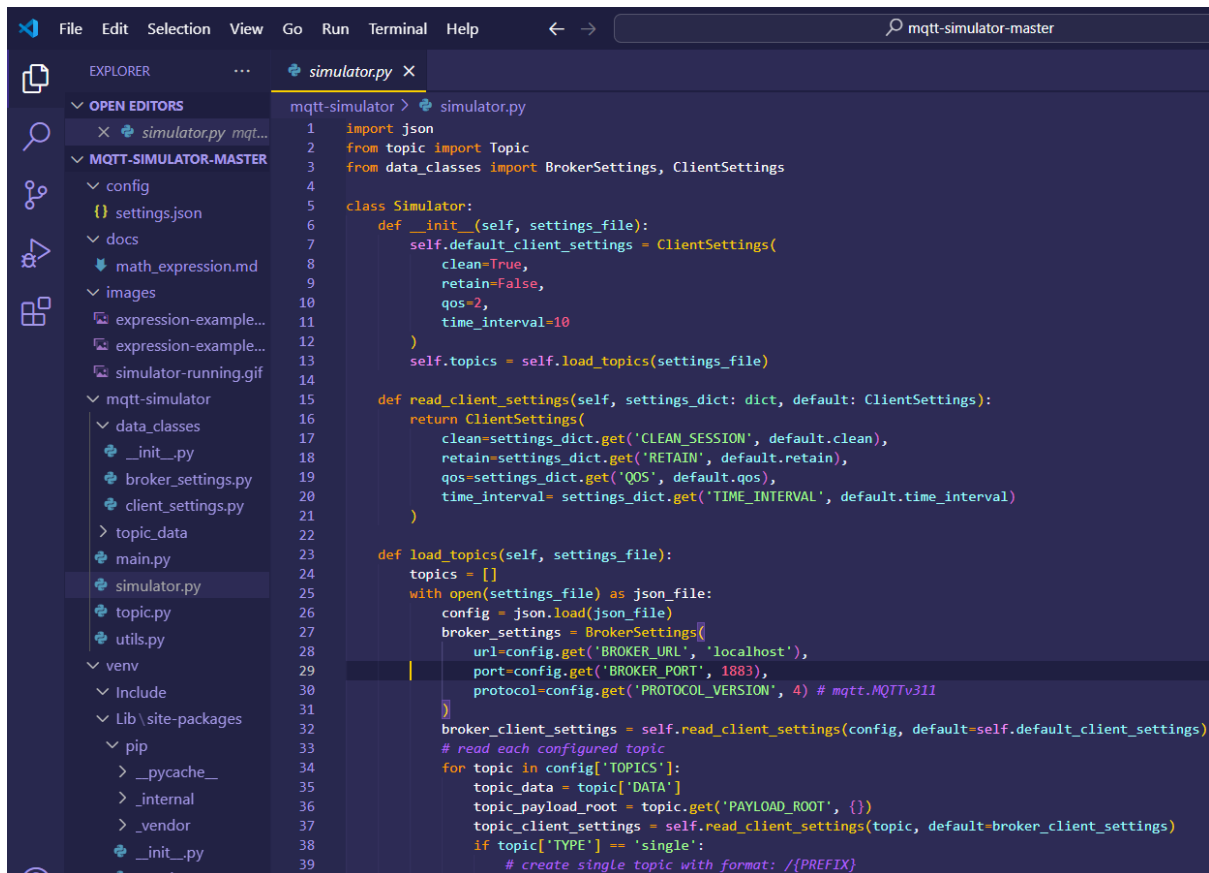
11323007

D3 Teknologi Informasi

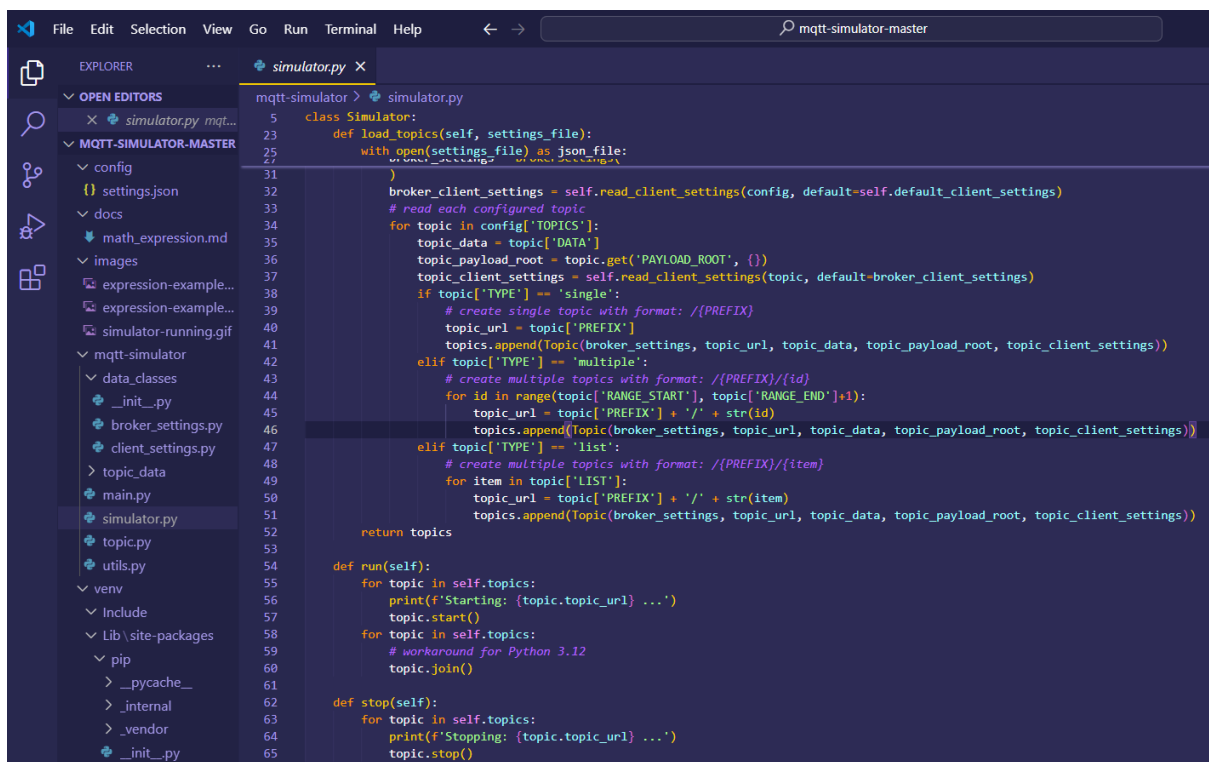
INSTITUT TEKNOLOGI DEL

FAKULTAS VOKASI

TAHUN AJARAN 2024/2025



```
1 import json
2 from topic import Topic
3 from data_classes import BrokerSettings, ClientSettings
4
5 class Simulator:
6     def __init__(self, settings_file):
7         self.default_client_settings = ClientSettings(
8             clean=True,
9             retain=False,
10             qos=2,
11             time_interval=10
12         )
13         self.topics = self.load_topics(settings_file)
14
15     def read_client_settings(self, settings_dict: dict, default: ClientSettings):
16         return ClientSettings(
17             clean=settings_dict.get('CLEAN_SESSION', default.clean),
18             retain=settings_dict.get('RETAIN', default.retain),
19             qos=settings_dict.get('QOS', default.qos),
20             time_interval=settings_dict.get('TIME_INTERVAL', default.time_interval)
21         )
22
23     def load_topics(self, settings_file):
24         topics = []
25         with open(settings_file) as json_file:
26             config = json.load(json_file)
27             broker_settings = BrokerSettings(
28                 url=config.get('BROKER_URL', 'localhost'),
29                 port=config.get('BROKER_PORT', 1883),
30                 protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311
31             )
32             broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
33             # read each configured topic
34             for topic in config['TOPICS']:
35                 topic_data = topic['DATA']
36                 topic_payload_root = topic.get('PAYLOAD_ROOT', {})
37                 topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)
38                 if topic['TYPE'] == 'single':
39                     # create single topic with format: /(PREFIX)
```



```
40                 topic_url = topic['PREFIX']
41                 topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
42             elif topic['TYPE'] == 'multiple':
43                 # create multiple topics with format: /(PREFIX)/(id)
44                 for id in range(topic['RANGE_START'], topic['RANGE_END']+1):
45                     topic_url = topic['PREFIX'] + '/' + str(id)
46                     topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
47             elif topic['TYPE'] == 'list':
48                 # create multiple topics with format: /(PREFIX)/(item)
49                 for item in topic['LIST']:
50                     topic_url = topic['PREFIX'] + '/' + str(item)
51                     topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
52         return topics
53
54     def run(self):
55         for topic in self.topics:
56             print(f'Starting: {topic.topic_url} ...')
57             topic.start()
58         for topic in self.topics:
59             # workaround for Python 3.12
60             topic.join()
61
62     def stop(self):
63         for topic in self.topics:
64             print(f'Stopping: {topic.topic_url} ...')
65             topic.stop()
```

Kode ini adalah implementasi dari simulator untuk menangani pengiriman data melalui protokol MQTT, yang umum digunakan untuk komunikasi antar perangkat

## CLASS 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)

    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)
        )
```

Penjelasan:

**\_\_init\_\_(self, settings\_file):**

menginisialisasi simulator dengan file pengaturan (settings\_file).

self.default\_client\_settings: Mengatur pengaturan klien default misalnya clean=True, retain=False

self.topics: Memuat semua topik yang ada

**read\_client\_settings(self, settings\_dict, default):**

Fungsi ini membaca pengaturan klien dari file JSON dan menggabungkannya dengan pengaturan default jika ada nilai yang tidak ditemukan.

## FUNGSI LOAD\_TOPICS

Fungsi load\_topics adalah tempat utama yang membaca file JSON dan memproses pengaturan topik-topik MQTT

```
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':
                # 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
```

Penjelasan:

broker\_settings: Membaca pengaturan broker seperti URL, port, dan versi protocol

broker\_client\_settings: Membaca pengaturan klien untuk broker yang diset secara global

## MENJALANKAN SIMULATOR metode run dan stop

```
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()
```

Penjelasan

run(self): menjalankan semua topik yang ada di dalam self.topics

Pesan Starting: {topic.topic\_url} dicetak untuk menunjukkan bahwa simulasi dimulai untuk setiap topik.

stop(self): Menutup atau menghentikan komunikasi pada semua topik yang ada di dalam self.topics.

Memanggil topic.stop() untuk setiap topik untuk menghentikan pengiriman data atau koneksi

Pesan Stopping: {topic.topic\_url} dicetak untuk menunjukkan bahwa topik dihentikan