

# **Modul Praktikum Jaringan Komunikasi 07**

## **Protokol Komunikasi Berbasis Jaringan Internet (*IoT*)**

Tujuan:

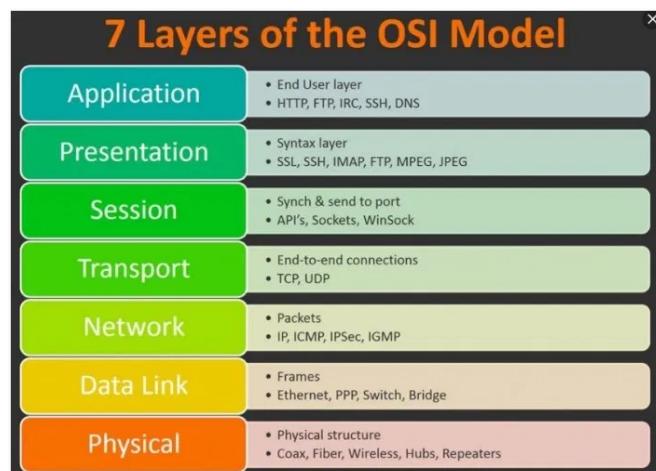
1. Mengenal dasar komunikasi data berbasis internet.
2. Mengenal protokol komunikasi HTTP.
3. Mengenal protokol komunikasi MQTT.
4. Mengenal protokol komunikasi WebSocket.
5. Mengaplikasikan Jaringan Komunikasi Berbasis IoT.

Kompetensi Dasar :

1. Dasar Informatika
2. Praktikum Dasar Informatika
3. Pengolahan Data
4. Jaringan Komunikasi
5. Elektronika Digital

Dasar Teori:

### **OSI Layer**



Model Open Systems Interconnection (model OSI) adalah model konseptual yang mencirikan dan standarisasi fungsi komunikasi dari sistem telekomunikasi atau komputasi tanpa

memperhatikan struktur dan teknologi internal yang mendasarinya. Tujuannya adalah interoperabilitas sistem komunikasi yang beragam dengan protokol komunikasi standar.

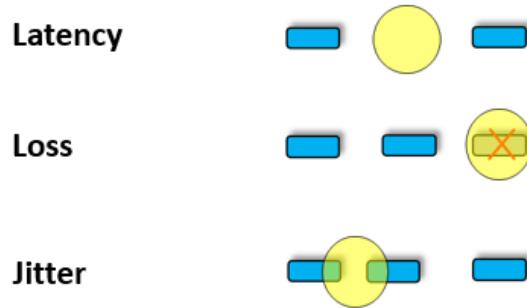
Model OSI membatasi aliran data dalam sistem komunikasi menjadi tujuh lapisan abstraksi, dari implementasi fisik transmisi bit melintasi media komunikasi hingga representasi data tingkat tertinggi dari aplikasi terdistribusi. Setiap lapisan perantara melayani kelas fungsionalitas ke lapisan di atasnya dan dilayani oleh lapisan di bawahnya. Kelas fungsionalitas diwujudkan dalam perangkat lunak dengan protokol komunikasi standar.

## **Pengenalan Internet of Things**

Internet of Things adalah teknologi terkini yang menciptakan jaringan global mesin dan perangkat yang mampu berkomunikasi dan bertukar data satu sama lain melalui Internet. Ada perbedaan antara Internet of Things dan Internet. Internet of Things dapat membuat informasi tentang objek yang terhubung, menganalisisnya, dan membuat keputusan; dengan kata lain, orang dapat mengatakan bahwa Internet of Things lebih pintar daripada Internet. Kamera keamanan, sensor, kendaraan, gedung, dan perangkat lunak adalah contoh benda yang dapat saling bertukar data.

Seiring bertambahnya jumlah aplikasi (perangkat) waktu nyata yang membutuhkan koneksi cerdas satu sama lain, tantangan Internet of Things juga akan meningkat. Tantangan tersebut adalah sebagai berikut.

Sensor dan perangkat yang terhubung dan dikomunikasikan bersama melalui infrastruktur Internet of Things mungkin perlu memperbarui tren atau fiturnya agar sesuai dengan perubahan lingkungan sekitarnya. Internet of Things adalah infrastruktur cerdas yang dapat memproses data yang dikumpulkan dan membuat keputusan yang diperlukan untuk meningkatkan dirinya sendiri dan untuk mengubah tren atau fitur perangkat yang terhubung untuk mengakomodasi perubahan lingkungan sekitarnya. Internet of Things adalah teknologi pintar yang membantu semua perangkat yang terhubung untuk memperbarui diri mereka sendiri sesuai dengan perubahan di lingkungan sekitarnya dan untuk dapat diadopsi dan bekerja di lingkungan aneh lainnya dengan akurasi tinggi. Hasilnya, sistem yang terhubung dengan cerdas dapat diproduksi jika infrastruktur cerdas dirancang dengan baik untuk menangani data yang dikumpulkan dari perangkat dengan benar, untuk membuat keputusan yang diperlukan.



Saat menghubungkan banyak perangkat melalui infrastruktur Internet of Things, data bersama di antara mereka juga akan sangat meningkat. Ini akan menyebabkan penundaan atau latensi pengiriman data di antara perangkat yang terhubung. Ini membuka tantangan baru yang harus dihadapi oleh Internet of Things untuk mengurangi latensi tersebut untuk memastikan infrastruktur Internet of Things yang kuat akan diperoleh.

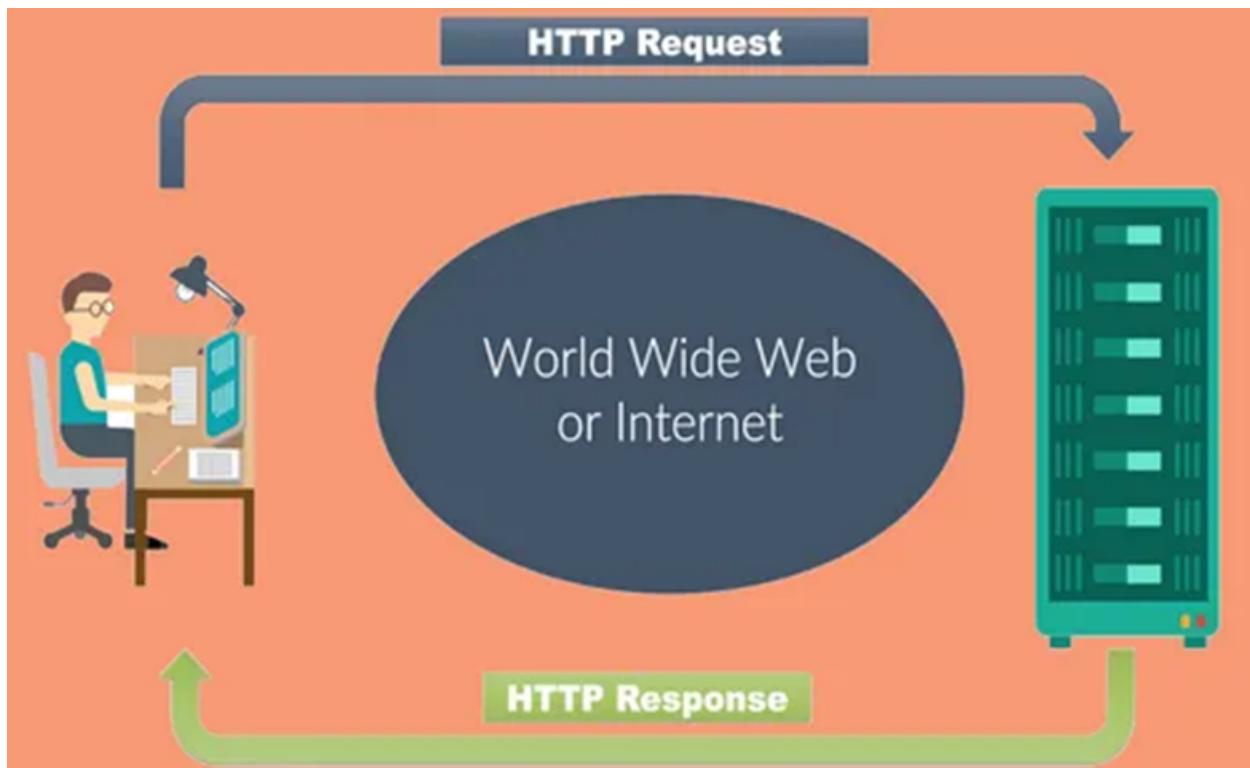
## Protokol

Protokol adalah peraturan atau prosedur untuk mengirimkan data pada perangkat elektronik [1]. Pada sistem komputer, setiap komputer berkomunikasi dengan komputer lainnya menggunakan sebuah protokol. Protokol komunikasi berbasis Internet of Things (IoT) adalah serangkaian aturan dan standar yang digunakan untuk mengatur pertukaran data antara perangkat IoT yang terhubung ke jaringan. IoT merupakan konsep di mana berbagai jenis perangkat fisik, seperti sensor, aktuator, dan perangkat elektronik lainnya, dapat terhubung dan berkomunikasi melalui jaringan internet. Protokol komunikasi IoT menjadi sangat penting karena berperan dalam memungkinkan interoperabilitas dan pertukaran data yang efisien antara perangkat dari berbagai produsen.

Dalam keseluruhan, protokol komunikasi berbasis IoT adalah elemen kunci dalam ekosistem IoT yang memungkinkan perangkat untuk berkomunikasi, berbagi data, dan berinteraksi dengan infrastruktur jaringan yang lebih luas. Pemilihan protokol yang sesuai tergantung pada kebutuhan spesifik dari aplikasi IoT, seperti ketersediaan sumber daya, keamanan, dan jenis data yang akan ditransmisikan.

## HTTP

*Hypertext Transfer Protokol* (HTTP) adalah protokol komunikasi data yang bekerja dengan sistem request-response antara perangkat client dan server [2]. Terdapat berbagai macam metode request. Metode request meminta server untuk melakukan sesuatu terhadap resources server sesuai request dari client. Resources tersebut dapat berupa berita, email, komentar, atau data di database [3].



Gambar 1: Salah satu contoh cara Kerja HTTP di Internet

Terdapat beberapa 6 metode HTTP yang dapat digunakan yaitu sebagai berikut [4]:

1. Delete Method: metode yang memberikan izin untuk menghapus konten tujuan URI dari server dan metode ini adalah kebalikan dari metode get.
2. Get Method: digunakan untuk meminta sumber dari server. Metode get membuat sebagian besar perintah HTTP yang dikirim ke server dan biasanya dipicu dengan “mengklik” pada sebuah tautan, menuliskan URI ke baris alamat browser dan menekan tombol “enter”, atau menekan tombol **kirim** format / kondisi / keadaan / susunan. Saat sumber daya dari server diminta, badan permintaan HTTP selalu kosong. Ketika metode get dipicu dari format / kondisi / keadaan / susunan, maka seluruh parameter kondisi /

keadaan / susunan ditambahkan (append) ke URI, dimulai dari simbol "?" dan diakhiri dengan simbol "&".

3. Head Method: metode *head* mirip dengan metode *get*, selain tidak mengembalikan sumber daya apa pun dari server, karena tidak mengembalikan badan respons. Semua tajuk respons yang berisi data tentang sumber daya akan dikembalikan dan dapat diinterogasi. Ini dapat efisien berdasarkan ukuran file yang diminta, sebelum memutuskan apakah akan menggunakan metode GET untuk pengambilan sumber daya. Saat informasi diminta dari server, badan permintaan HTTP selalu kosong.
4. Option Method: metode *option* mirip dengan metode *get*, selain tidak mengembalikan sumber daya apa pun ke server, karena tidak mengembalikan badan respons. Hal ini mirip dengan metode *head* tetapi lebih minimalis karena hanya mengembalikan *header* respons yang menunjukkan metode HTTP yang dapat digunakan pada sumber daya. Saat informasi dari server diminta, badan permintaan HTTP kosong. Meskipun tidak seperti metode *head* dan *get*. Metode ini dapat berubah di bagian tengah rilis perubahan.
5. Post Method: metode yang paling sering digunakan setelah metode *get* dan dapat dipicu dari bentuk HTML yang diatur menjadi *post*. Terdapat beberapa alasan digunakan metode *post* dibandingkan metode *get*, yaitu:
  - a. ketika penggunaan metode *get*, parameter informasi dari format / kondisi keadaan / susunan dapat dilihat oleh pengguna dalam bentuk string URI dan untuk informasi sensitif, diharapkan untuk disembunyikan.
  - b. Mungkin terdapat informasi yang ingin disampaikan ke server dan menggunakan URI untuk penambahan yang terbukti tidak praktis.

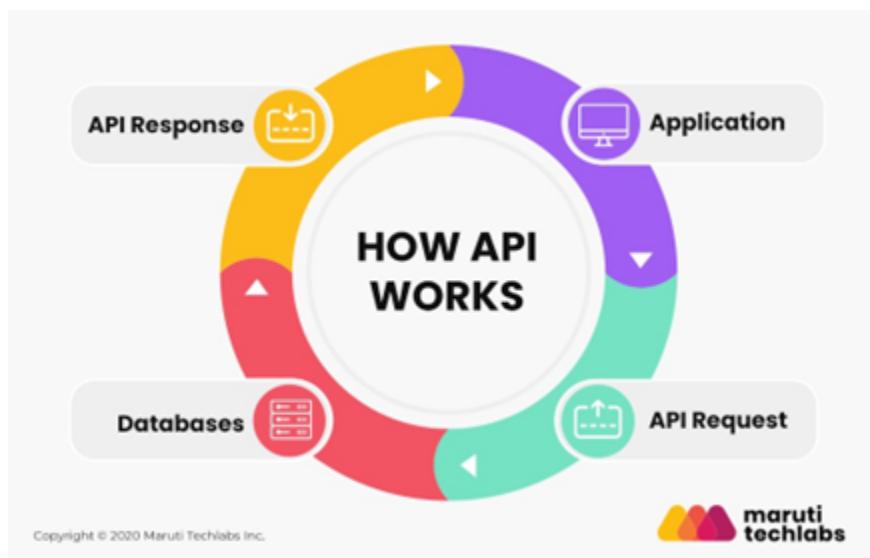
Semua data pada metode *post* ditempatkan di dalam badan permintaan sehingga untuk menjaganya tetap "pribadi" dan juga menghentikan URI agar tidak tersumbat dengan parameter kondisi / keadaan / susunan. Penggunaan metode *post* adalah untuk membuat / memperbarui / menghapus informasi pada server, sebagai contohnya adalah untuk pembaharuan *database* informasi dari parameter yang diberikan.

6. Put Method: metode yang dapat digunakan untuk memasukkan konten, khususnya dalam bentuk dokumen, ke URI tujuan pada server dan metode ini adalah lawan dari metode *delete*. Bagian kepala **Request-URI** digunakan untuk meletakkan nilai ke dalam URI tujuan, untuk responnya itu harus ditambahkan sendiri. Ketika sumber daya

ditempatkan ke server, badan permintaan HTTP yang berisi data untuk dimasukkan ke dalam sumber daya yang telah di *header Request-URI*. Variasi metode *put* adalah dari metode *post* karena sumber daya selalu dimasukkan ke dalam URI tujuan yang ditentukan di *header Request-URI*. Saat menggunakan metode *post*, pemrogram memiliki lebih banyak kendali atas lokasi sumber yang dikirim. Metode ini perlu perhatian lebih karena sumber daya yang ada dan cocok dengan URI tujuan dapat ditimpak tanpa adanya peringatan.

7. Metode *trace*: berguna untuk melihat keadaan permintaan ketika data mencapai server. Karena metode ini melibatkan transversal yang melalui banyak koneksi internet, metode HTTP ini dapat berguna bagi para insinyur sistem untuk keperluan *debugging* dan merupakan metode yang paling tidak berguna bagi para pemrogram.

Pada HTTP, kita juga akan mengenal istilah API. Apa itu API? API atau *Application Programming Interface* adalah sebuah interface yang dapat menghubungkan aplikasi satu dengan aplikasi lainnya. Jadi, API berperan sebagai perantara antar berbagai aplikasi berbeda, baik dalam satu platform yang sama atau lintas platform.



## MQTT

*Message Queuing Telemetry Transport* (MQTT) adalah salah satu protokol pengiriman data berbasis *wireless* yang berjalan diatas stack TCP/IP dan dirancang khusus untuk machine to machine yang tidak memiliki alamat khusus. Maksud dari kata tidak memiliki alamat khusus

ini seperti halnya sebuah arduino, raspi atau device lain yang tidak memiliki alamat khusus. Sistem kerja MQTT menerapkan **Publish** dan **Subscribe** data. Dan pada penerapannya, device akan terhubung pada sebuah **Broker** dan mempunyai suatu **Topic** tertentu. Pada protokol pengiriman ini, akan dikenal 4 istilah yang telah disebutkan.

- **Publish**

Merupakan cara suatu device untuk mengirimkan datanya ke subscribers. Biasanya pada publisher ini adalah sebuah device yang terhubung dengan sensor tertentu.

- **Broker**

Berfungsi untuk menghandle data publish dan subscribe dari berbagai device, bisa diibaratkan sebagai server yang memiliki alamat IP khusus. Beberapa contoh dari Broker yang ada seperti Mosquitto, HiveMQ dan Mosca. Pada PJK 06 yang akan digunakan sebagai broker adalah Mosquitto.

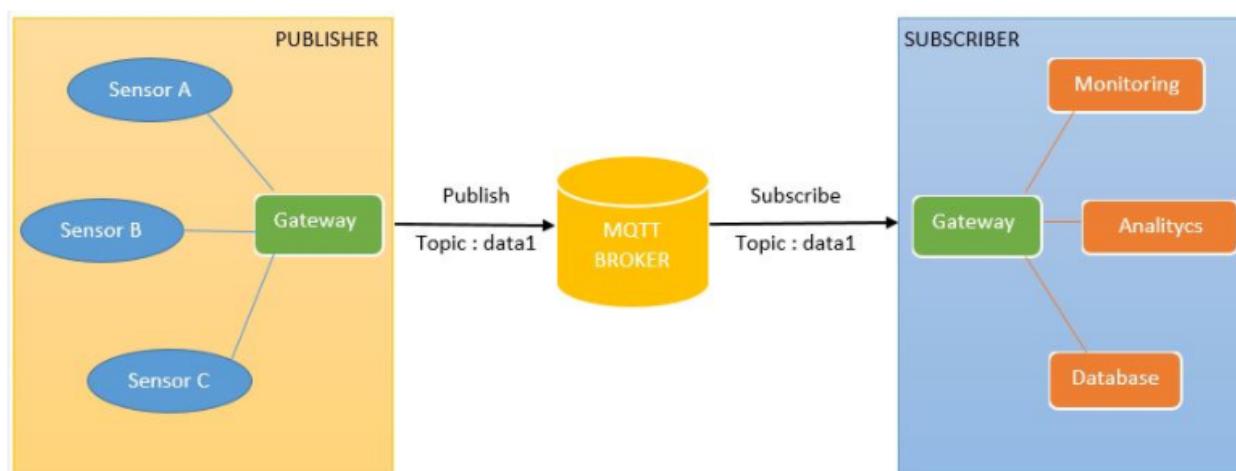
- **Subscribe**

Merupakan cara suatu device untuk menerima berbagai macam data dari publisher. Subscriber dapat berupa aplikasi monitoring sensor dan sebagainya, subscriber ini yang nantinya akan meminta data dari publisher.

- **Topic**

Seperti halnya pengelompokan data di suatu kategori tertentu. Pada sistem kerja MQTT protokol ini, topic bersifat wajib hukumnya. Pada setiap transaksi data antara Publisher dan Subscriber harus memiliki suatu topic tertentu.

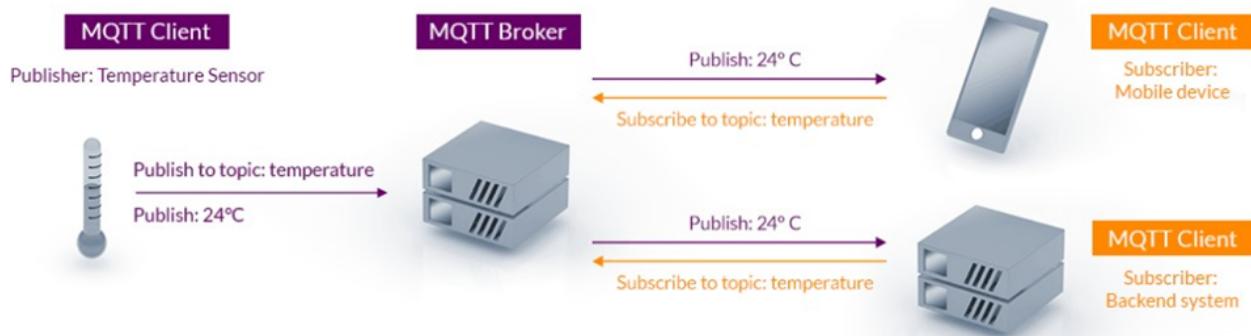
Untuk mempermudah gambaran mengenai pengiriman data dengan protokol MQTT perhatikan blok diagram berikut.



Pada blok diagram diatas dapat dilihat pada protokol MQTT terdapat 3 poin penting yaitu Publisher, Broker, dan Subscriber.

1. **Blok Publisher** disini memiliki tugas yaitu mengirimkan data yang terbaca pada sensor menuju Broker. Pada blok ini terdapat gateway. Biasanya antara gateway dengan sensor terdapat controller yang kita kenal seperti Arduino, PSoC, Raspberry Pi, dan sejenisnya. Data yang dikirim dari sensor menuju Broker disini memiliki **Topic** yaitu data1.
2. **Blok Broker** disini berfungsi untuk penghubung transaksi antara Publisher dengan Subscriber. Broker memiliki alamat yang dapat diakses oleh Publisher ataupun Subscriber. Broker mengenali pengelompokan data melalui **Topic**. Inilah mengapa penamaan topic menjadi penting. Hal ini karena ketika Subscriber ingin mendapatkan data yang sama dengan yang terbaca oleh sensor pada blok diagram diatas, maka Subscriber mengambil data dengan Topic yaitu data1.
3. **Blok Subscriber** bertugas untuk menerima data dari Broker dengan **Topic** data1. Data yang diterima berdasarkan pembacaan sensor pada publisher dapat disimpan dalam database dan diolah atau dianalisis yang kemudian selanjutnya dapat menjadi sistem monitoring yang terstruktur.

Selain blok diagram diatas, berikut kami lampirkan sebuah contoh implementasi dari penggunaan MQTT.



Berdasarkan gambar diatas, dapat dilihat MQTT terbagi menjadi *Client* dan *Broker* dimana *Client* terbagi menjadi *Publisher* dan *Subscriber* [5]. Sesuai dengan yang telah dijelaskan sebelumnya pada blok diagram diatas, Pada *Publisher* terdapat sebuah sensor yang akan mendeteksi temperatur (ruangan), kemudian data ini diolah oleh mikrokontroler (Bisa Arduino, PSoC, atau Raspberry Pi) yang selanjutnya dikirimkan menuju MQTT Broker. Topic

pada proses ini adalah ***temperature***. Kemudian *Subscriber* pada proses ini adalah *mobile device* atau *Backend system* yang menerima data dari Broker dengan topic yang sama yaitu ***temperature***. Sehingga pada *mobile device* dapat dilihat data yang terbaca oleh *temperature sensor* sebagai *publisher*.

## WebSocket

WebSocket adalah sebuah protokol komunikasi berbasis *client* dan *server*. Protokol WebSocket memungkinkan komunikasi *full-duplex* dilakukan pada protokol HTTP melalui koneksi TCP. WebSocket dalam hal ini mengatasi kelemahan dari protokol HTTP, di mana setiap pengiriman data oleh *server* ke *client* harus didahului oleh *request* dari *client*. (<https://sookocheff.com/post/networking/how-do-websockets-work/>)

WebSocket menggunakan HTTP sebagai mekanisme transport awal. Lalu, *client* melakukan *request* ke *server* untuk meng-*upgrade* protokol menjadi WebSocket. Jika *server* mampu dan setuju untuk melayani protokol WebSocket, koneksi TCP akan dipertahankan untuk komunikasi antara *client* dan *server*.

Ketika *client* mengirimkan *request* protokol WebSocket ke *server*, *header* HTTP yang dikirimkan akan serupa seperti pada gambar di bawah (<https://datatracker.ietf.org/doc/html/rfc6455>), yang mana terdapat *header* Connection yang bernilai Upgrade dan *header* Upgrade yang bernilai websocket.

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

Selanjutnya, jika *server* menyetujui koneksi WebSocket, *server* akan memberi *response* kepada *client* dengan *header* HTTP yang serupa seperti pada gambar di bawah, yang mana terdapat *header* Connection dan Upgrade seperti *header* pada *request* dari *client*, serta *status code* 101 yang menandakan *server* mengubah protokol komunikasinya.

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzhZRbK+xOo=
```

## Aplikasi yang digunakan Praktikum Jaringan Komunikasi

### 1. Python

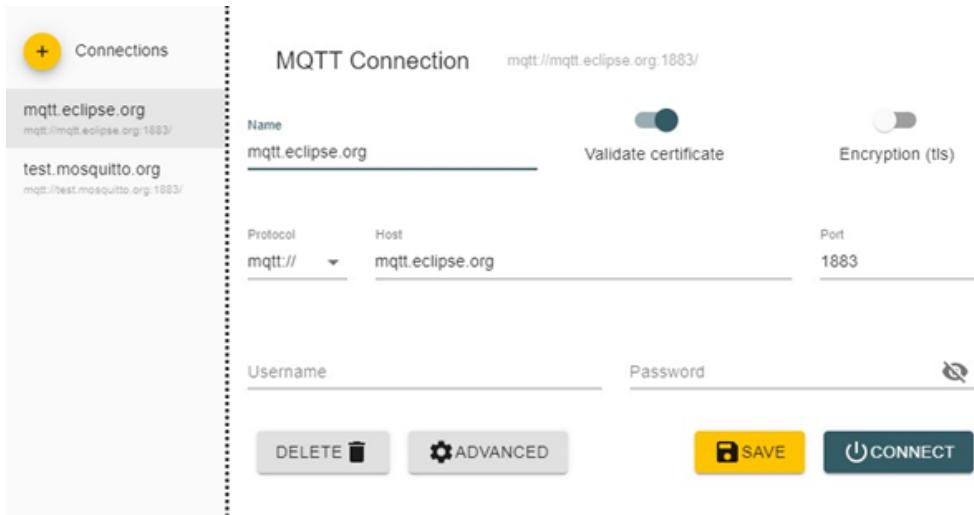


Python digunakan untuk membuat program HTTP dan MQTT. Pada PJK 06, Python digunakan untuk *running server*, melakukan *request* dari server, dan membuat API untuk HTTP. Sedangkan untuk protokol MQTT, Python digunakan untuk melakukan publish topic menuju MQTT Broker dan mengambil data selaku Subscriber dari Broker yang ditunjukan melalui aplikasi MQTT Explorer. Flask adalah kerangka kerja web. Flask menyediakan alat, libraries, dan teknologi yang memberikan akses untuk membuat aplikasi web. Aplikasi web ini dapat berupa halaman web, blog, wiki, atau seperti aplikasi kalender *web-based* atau *website* komersial [6].

### 2. MQTT Explorer



Pada protokol MQTT, kita akan menggunakan MQTT Explorer untuk memantau Topic pada Broker, dilihat dari sudut pandang broker. Broker yang akan digunakan adalah Mosquitto. Berikut tampilan awal pada MQTT Explorer.



Pada laman ini, terdapat beberapa hal yang harus diperhatikan seperti Connection, Port, dan Host. Untuk melakukan uji coba ini, pastikan perangkat yang digunakan berada pada jaringan yang sama dengan broker.

### 3. Postman



Postman adalah sebuah aplikasi yang berfungsi sebagai REST CLIENT untuk uji coba REST API. Postman biasa digunakan oleh para pengembang pembuat API sebagai *tools* untuk menguji API yang telah dibuat. Pada PJK-06, Postman digunakan untuk melihat parameter API, *header*, atau *preview* yang biasa ditampilkan dalam bentuk json. Hal ini serupa ketika melakukan running menggunakan browser, namun pada Postman, tampilan lebih tertata dan membuat API dapat menguji API-nya.

#### 4. Mosquitto



Mosquitto adalah sebuah perangkat lunak broker MQTT (Message Queueing Telemetry Transport) yang open-source dan tersedia secara gratis. MQTT adalah protokol komunikasi ringan yang digunakan untuk pertukaran data antara perangkat, terutama dalam lingkungan Internet of Things (IoT).

Mosquitto berfungsi sebagai perantara atau broker yang mengelola aliran pesan antara perangkat yang berkomunikasi melalui MQTT. Broker ini menerima pesan yang dikirimkan oleh perangkat pengirim dan mengirimkannya ke perangkat penerima yang berlangganan topik tertentu. Ini memungkinkan perangkat IoT untuk berkomunikasi secara efisien dan dengan overhead yang rendah.

Beberapa fitur utama Mosquitto meliputi dukungan untuk QoS (Quality of Service) yang dapat dikonfigurasi, keamanan melalui autentikasi dan enkripsi, dan kemampuan untuk mengelola banyak klien secara bersamaan. Mosquitto dapat diinstal di berbagai platform, termasuk Linux, Windows, dan macOS, sehingga memudahkan integrasi MQTT dalam proyek-proyek IoT dan aplikasi lainnya.

Langkah percobaan :

Percobaan 01: HTTP dengan Postman

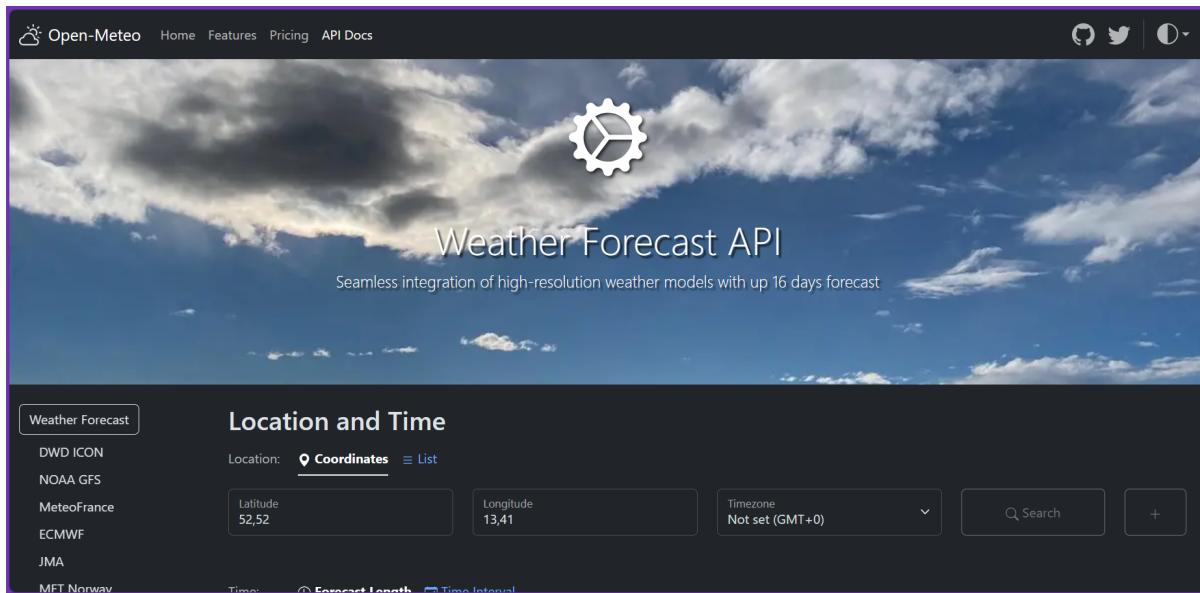
1. Silahkan cari website yang sudah API (selain yang dicontohkan di modul ini), dibebaskan kepada masing-masing praktikan. Contohnya adalah NASA, Solar Atlas, Openweather

2. Misalkan website yang digunakan adalah open meteo, tampilan awalnya adalah seperti gambar dibawah ini.

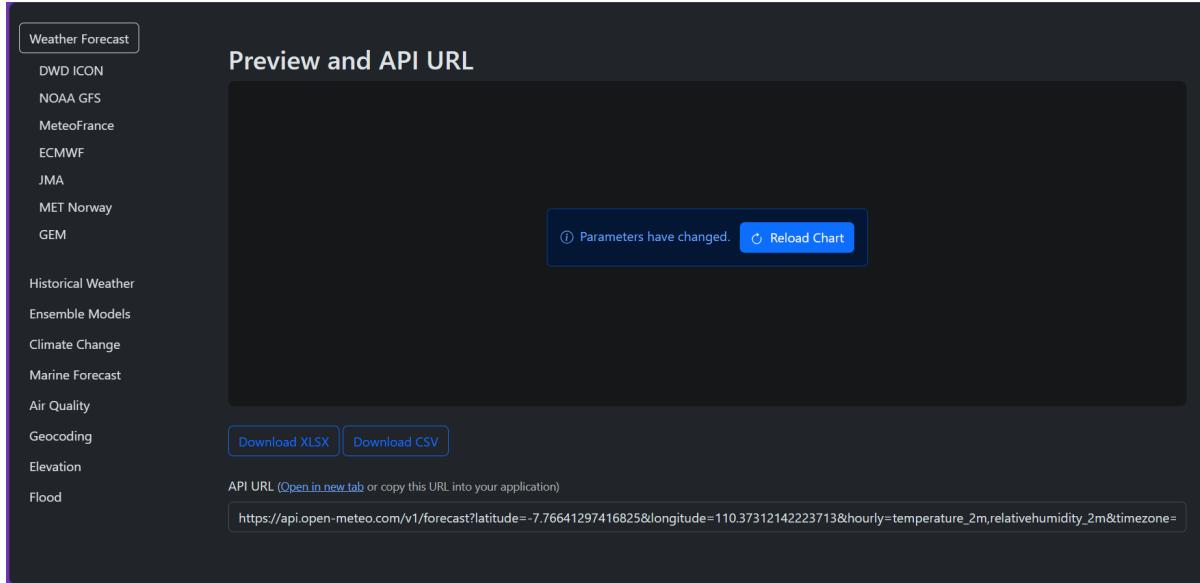


Gambar 1: Tampilan awal website Open-Meteo

3. Masuk ke bagian Documentation sampai muncul tampilan seperti gambar di bawah ini.



4. Masukan koordinat latitude dan longitude salah satu kota di Indonesia pada bagian pencarian lokasi.
5. Masukkan variabel temperatur dan kelembaban relatif untuk mendapatkan contoh API dengan format json.
6. Cari bagian API URL seperti pada gambar di bawah ini.



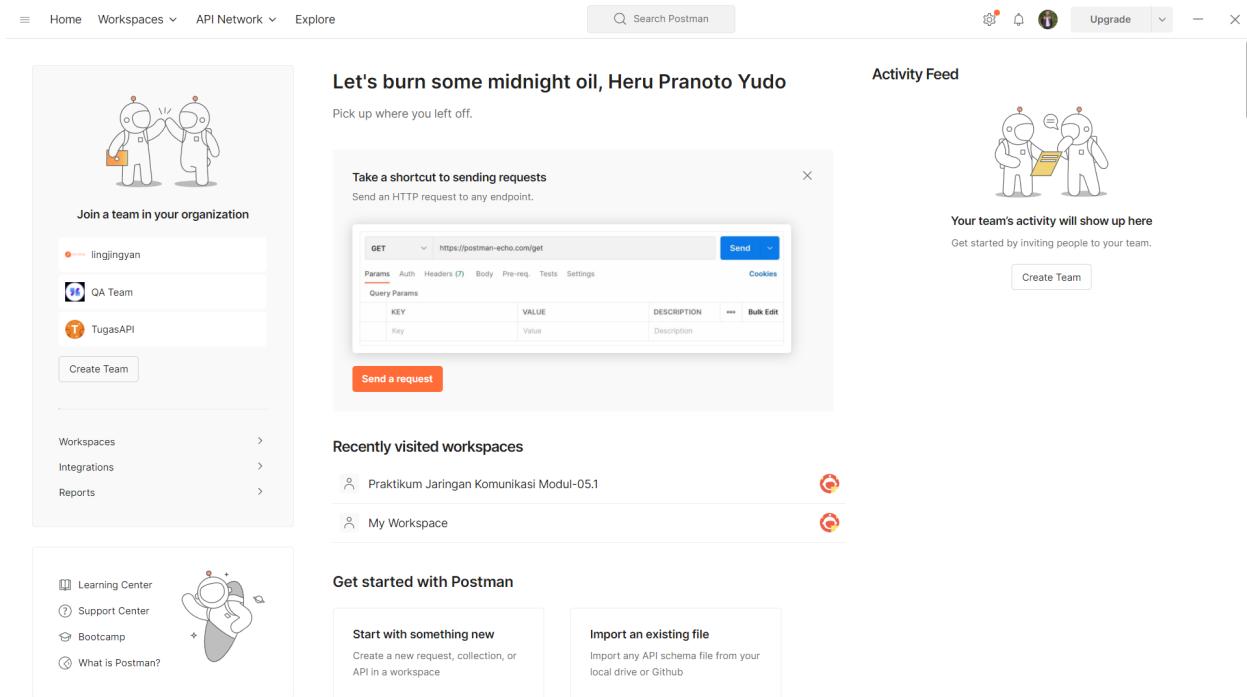
7. Salin dan tempel url tersebut pada taskbar baru jendela browser komputer Anda.
8. Maka akan ditampilkan data kondisi cuaca terbaru pada lokasi dengan parameter temperatur dan kelembaban relatif, seperti pada gambar di bawah ini.

```

1 {
2   "latitude": -7.75,
3   "longitude": 110.375,
4   "generationtime_ms": 0.05996227264404297,
5   "utc_offset_ms": -18000,
6   "timezone": "Asia/Bangkok",
7   "timezone_abbreviation": "+07",
8   "elevation": 139,
9   "hourly": {
10     "time": "2023-11-01T00:00",
11     "temperature_2m": "°C",
12     "relativehumidity_2m": "%"
13   },
14   "hourly": [
15     "time": [
16       "2023-11-01T00:00",
17       "2023-11-01T01:00",
18       "2023-11-01T02:00",
19       "2023-11-01T03:00",
20       "2023-11-01T04:00",
21       "2023-11-01T05:00",
22       "2023-11-01T06:00",
23       "2023-11-01T07:00",
24       "2023-11-01T08:00",
25       "2023-11-01T09:00",
26       "2023-11-01T10:00",
27       "2023-11-01T11:00",
28       "2023-11-01T12:00",
29       "2023-11-01T13:00",
30       "2023-11-01T14:00",
31       "2023-11-01T15:00",
32       "2023-11-01T16:00",
33       "2023-11-01T17:00",
34       "2023-11-01T18:00",
35       "2023-11-01T19:00",
36       "2023-11-01T20:00",
37       "2023-11-01T21:00",
38       "2023-11-01T22:00",
39       "2023-11-01T23:00",
40       "2023-11-02T00:00",
41       "2023-11-02T01:00",
42       "2023-11-02T02:00",
43       "2023-11-02T03:00",
44       "2023-11-02T04:00",
45       "2023-11-02T05:00",
46       "2023-11-02T06:00",
47       "2023-11-02T07:00",
48       "2023-11-02T08:00",
49       "2023-11-02T09:00",
50       "2023-11-02T10:00",
51       "2023-11-02T11:00",
52       "2023-11-02T12:00",
53       "2023-11-02T13:00",
54       "2023-11-02T14:00"
      ]
    ]
  }
}

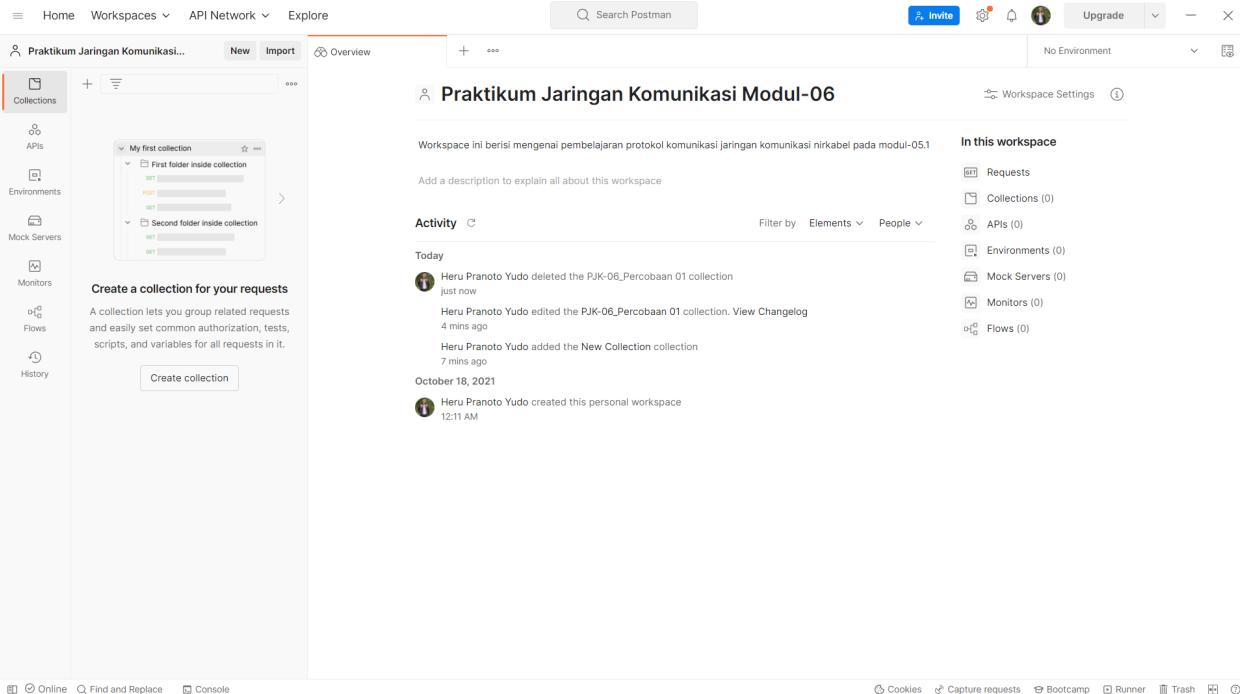
```

9. Buka aplikasi Postman yang telah Anda pasang pada komputer Anda.
10. Masuk ke bagian Halaman Utama. Tampilan awal aplikasi Postman adalah seperti gambar di bawah ini.



11. Buat Workspaces baru dengan menekan taskbar Workspace di bagian atas tampilan awal Postman

12. Jika sudah, maka akan tertampil tampilan awal Workspaces seperti gambar di bawah ini.



13. Buat project baru dengan menekan tanda plus (+) sebelah bagian overview, seperti pada gambar di bawah ini.

14. Salin tautan pada langkah 06 kemudian tempel pada kotak tautan untuk dilakukan uji coba API.

15. Pilih metode HTTP yang ingin yang sesuai dengan API-nya

16. Tekan Send untuk melihat isi API-nya, seperti pada gambar di bawah ini.

```

1
2   "latitude": -7.76,
3   "longitude": 110.375,
4   "generationtime_ms": 0.050067901611328125,
5   "utc_offset_seconds": 25200,
6   "timezone": "Asia/Bangkok",
7   "timezone_abbreviation": "+07",
8   "elevation": 139.0,
9   "hourly_units": [
10     {
11       "time": "iso0601",
12       "temperature_2m": "+C",
13       "relativehumidity_2m": "%"
14     }
15   ]
  
```

17. Catat dan amati kode status, waktu respon API, dan ukuran file json-nya.

18. Masuk ke bagian Pretty dan Preview pada bagian body API-nya

19. Amati dan catat parameter dan payload setiap parameternya.

## Percobaan 02: Olah Data Json menggunakan Python dari API HTTP

1. Buka tautan berikut ini, <https://data.bmkg.go.id/gempabumi/>
2. Salin dan tempel data dalam format json pada tab baru browser Anda
3. Berikut ini adalah contoh ketika salah satu tautan berhasil terbuka.

```
1 {  
2     "Infogempa": {  
3         "gempa": {  
4             "Tanggal": "01 Nov 2023",  
5             "Jam": "00:37:23 WIB",  
6             "DateTime": "2023-10-31T17:37:23+00:00",  
7             "Coordinates": "-0.16,123.84",  
8             "Lintang": "0.16 LS",  
9             "Bujur": "123.84 BT",  
10            "Magnitude": "5.1",  
11            "Kedalaman": "112 km",  
12            "Wilayah": "61 km BaratDaya BOLAANGUKI-BOLSEL-SULUT",  
13            "Potensi": "Tidak berpotensi tsunami",  
14            "Dirasakan": "-",  
15            "Shakemap": "20231101003953.mmi.jpg"  
16        }  
17    }  
18 }
```

4. Buka VsCode komputer Anda dan buat kode program berikut ini.

```
import requests  
import json  
from flask import Flask, json, request  
  
api = Flask(__name__)  
  
# url = 'http://10.46.10.128:3001/site/indoor7/temperature'  
url = 'tautan yang telah Anda salin sebelumnya'  
response = requests.get(url)  
a = response.json()  
b = json.loads(response.text)  
#print(response.json())  
c = b["parameter-01"]["parameter-02"][[indeks ke-n]]  
print(a)
```

5. Pasang modul flask dan requests.
6. Jalankan kode program di atas langsung dari terminal VsCode.
7. Amati dan catat hasil pengamatan yang didapatkan
8. Ubah kode program di atas hingga Anda hanya mendapatkan nilai dari parameter terpenting pada file jsonnya
9. Variasikan percobaan sesuai tabel percobaan sesuai kesepakatan bersama asisten praktikum Anda.
10. Amati dan catat hasil percobaan pada tabel percobaan

### Percobaan 03: Berbagai Metode pada HTTP

1. Buka VsCode pada komputer Anda
2. Buat kode program seperti pada gambar di bawah ini.

```
from flask import Flask, jsonify, abort, render_template, make_response, request

app = Flask(__name__)

# dummy database array of dictionaries
tasks = [
    {
        'id': 1,
        'title': u'Buy Groceries',
        'desc': u'Milk,Chese,Brocoli',
        'done': False
    },
    {
        'id':2,
        'title': u'Go to the Gym',
        'desc': u'leg day,arm day',
        'done': False
    }
]

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/todo/api/v1.0/tasks', methods=['GET']) #get JSON data 'tasks'
def get_tasks():
    return jsonify({'task':tasks})

@app.route('/todo/api/v1.0/tasks/<int:task_id>', methods=['GET']) #get JSON data by task_id
def get_tasks_by_id(task_id):
    task = [task for task in tasks if task['id']== task_id]
    if len(task)==0:
        abort(404)
    return jsonify({'task': task[0]})
```

```

@app.route('/todo/api/v1.0/tasks', methods=['POST']) #post new JSON task
def create_task():
    if not request.json or not 'title' in request.json:
        abort(404)
    task = {
        'id': tasks[-1]['id'] + 1,
        'title': request.json['title'],
        'desc' : request.json.get('desc',''),
        'done' : False
    }
    tasks.append(task)
    return jsonify({'task': task}), 201

@app.route('/todo/api/v1.0/tasks/<int:task_id>', methods=['PUT']) #update task by task_id
def update_task(task_id):
    task = [task for task in tasks if task['id'] == task_id]
    if len(task) == 0:
        abort(404)
    if not request.json:
        abort(400)
    if 'title' in request.json and type(request.json['title']) != str:
        abort(400)
    if 'desc' in request.json and type(request.json['desc']) is not str:
        abort(400)
    if 'done' in request.json and type(request.json['done']) is not bool:
        abort(400)
    task[0]['title'] = request.json.get('title', task[0]['title'])
    task[0]['desc'] = request.json.get('desc', task[0]['desc'])
    task[0]['done'] = request.json.get('done', task[0]['done'])
    return jsonify({'task': task[0]})

@app.route('/todo/api/v1.0/tasks/<int:task_id>', methods=['DELETE']) #delete task by task_id
def remove_task(task_id):
    task = [task for task in tasks if task['id'] == task_id]
    if len(task) == 0:
        abort(404)
    tasks.remove(task[0])
    return jsonify({'result':True})

@app.errorhandler(404)
def not_found(error):
    return make_response(jsonify({'error':'Not found'}), 404)

@app.errorhandler(400)
def bad_request(error):
    return make_response(jsonify({'error':'Bad request'}), 400)

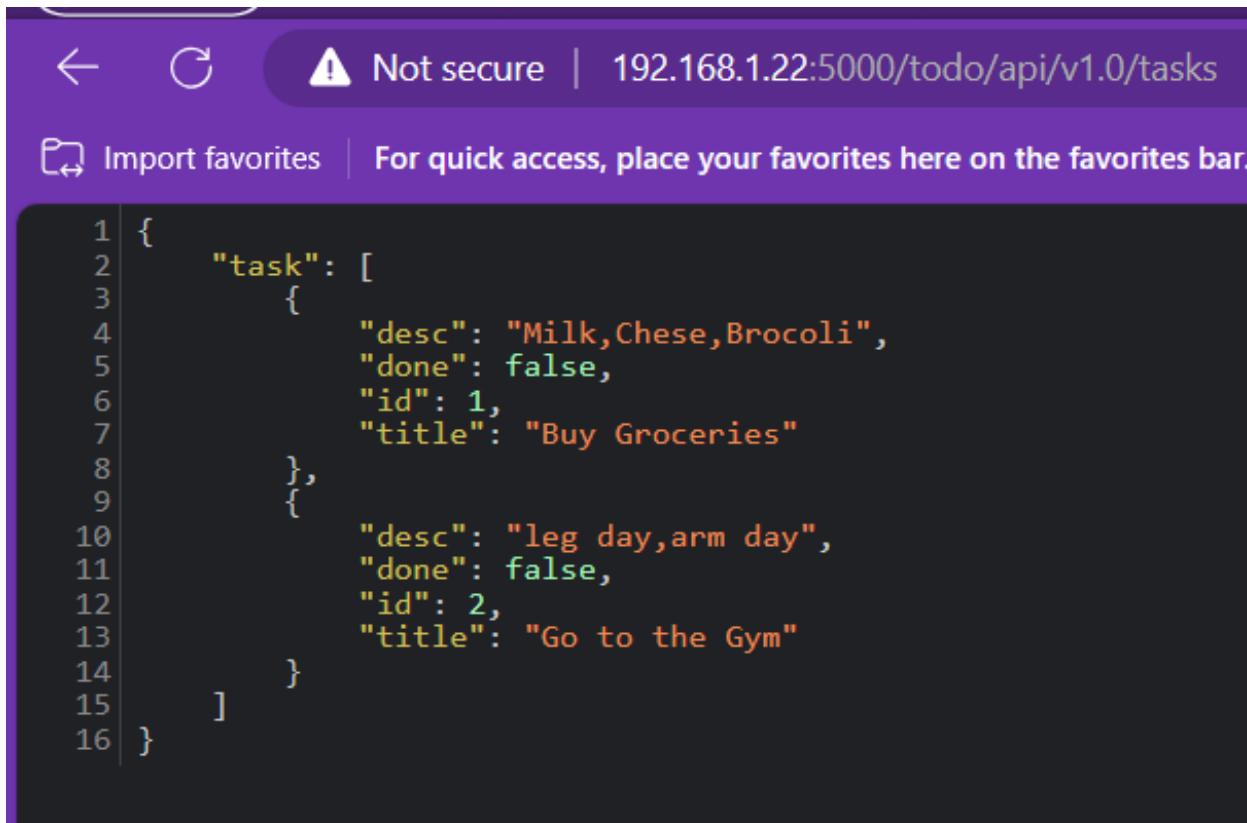
if __name__ == '__main__':
    app.run(host="0.0.0.0" ,debug=True)

```

3. Jalankan kode program di atas
4. Salin alamat tautan yang muncul pada terminal komputer Anda, seperti pada gambar di bawah ini.

```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on all addresses (0.0.0.0)  
* Running on http://127.0.0.1:5000  
* Running on http://192.168.1.22:5000  
Press CTRL+C to quit  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 406-239-060
```

5. Salin, buka, dan tambahkan parameter alamat tautan pada tab browser Anda agar tertampil data yang diinginkan.



The screenshot shows a web browser window with the following details:

- Address bar: Not secure | 192.168.1.22:5000/todo/api/v1.0/tasks
- Import favorites bar: Import favorites | For quick access, place your favorites here on the favorites bar.
- Main content area:

```
1 {  
2     "task": [  
3         {  
4             "desc": "Milk,Chese,Brocoli",  
5             "done": false,  
6             "id": 1,  
7             "title": "Buy Groceries"  
8         },  
9         {  
10            "desc": "leg day,arm day",  
11            "done": false,  
12            "id": 2,  
13            "title": "Go to the Gym"  
14        }  
15    ]  
16 }
```

6. Ulangi langkah 4 dan 5 hingga seluruh metode pada HTTP diuji coba
7. Amati dan catat hasil percobaan yang didapatkan pada tabel percobaan

## Percobaan 04: Broker MQTT

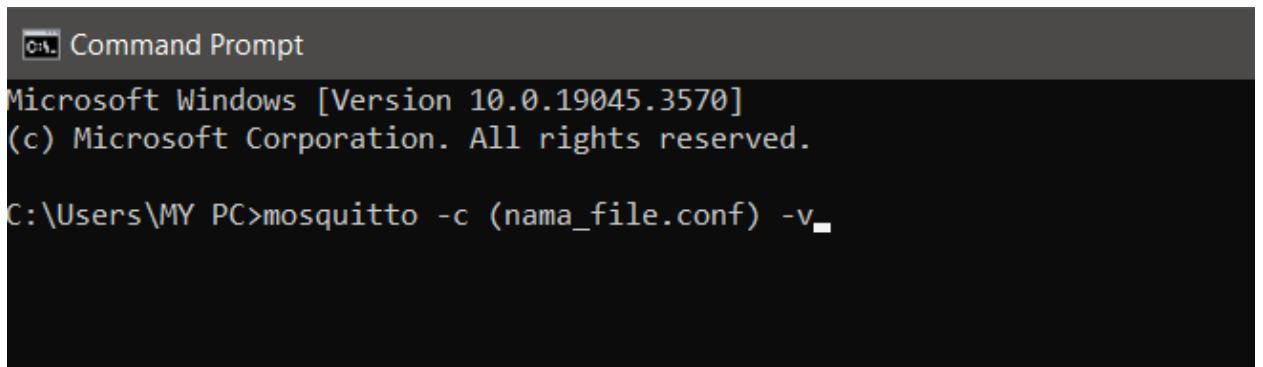
1. Buat file dengan ekstensi .conf pada notepad yang berisikan listener 1883 dan allow\_anonymous true

---

File Edit Format View Help

listener 1883  
allow\_anonymous true

2. Simpan file .conf pada folder penginstalan *mosquitto*
3. Jalankan file .conf yang sudah anda buat dengan cara membuka terminal (cmd/Powershell) dengan perintah “mosquitto -c (nama file .conf) -v”

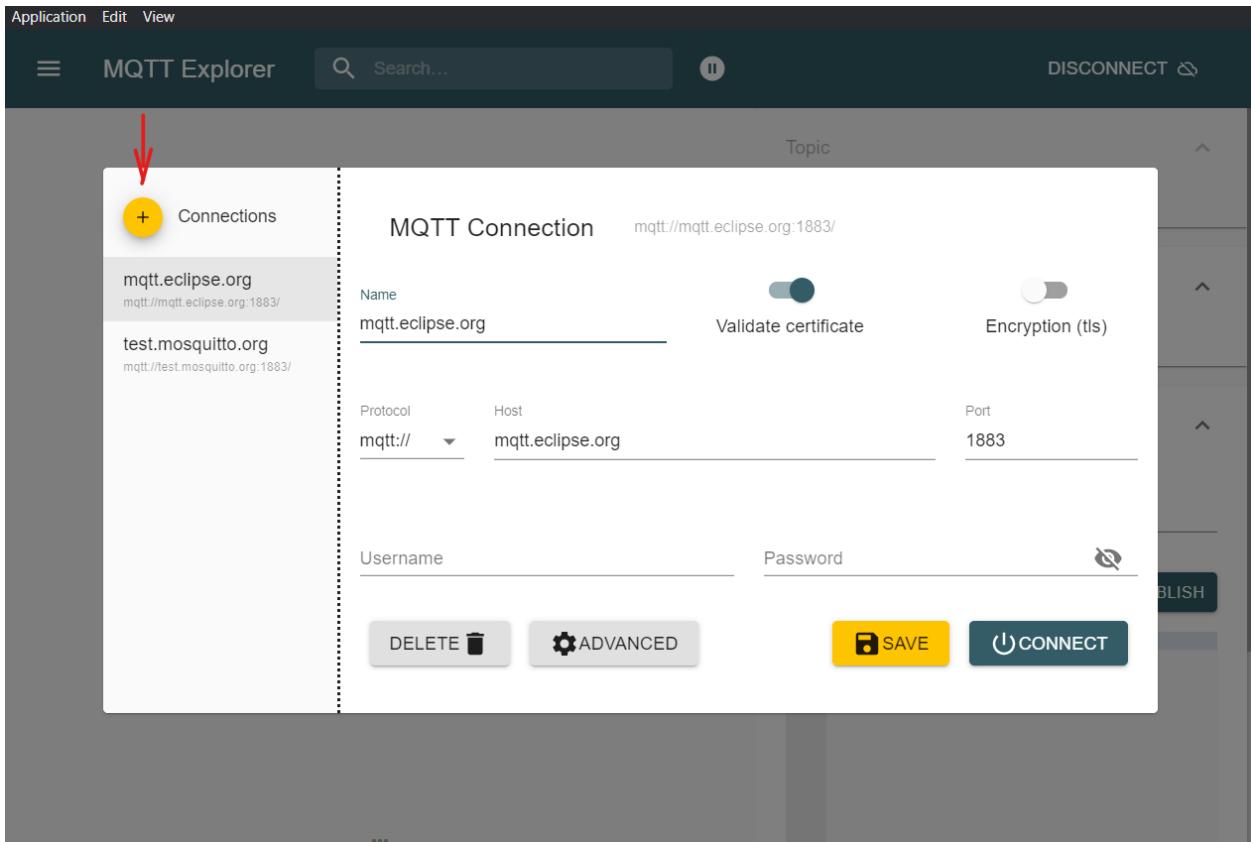


Command Prompt

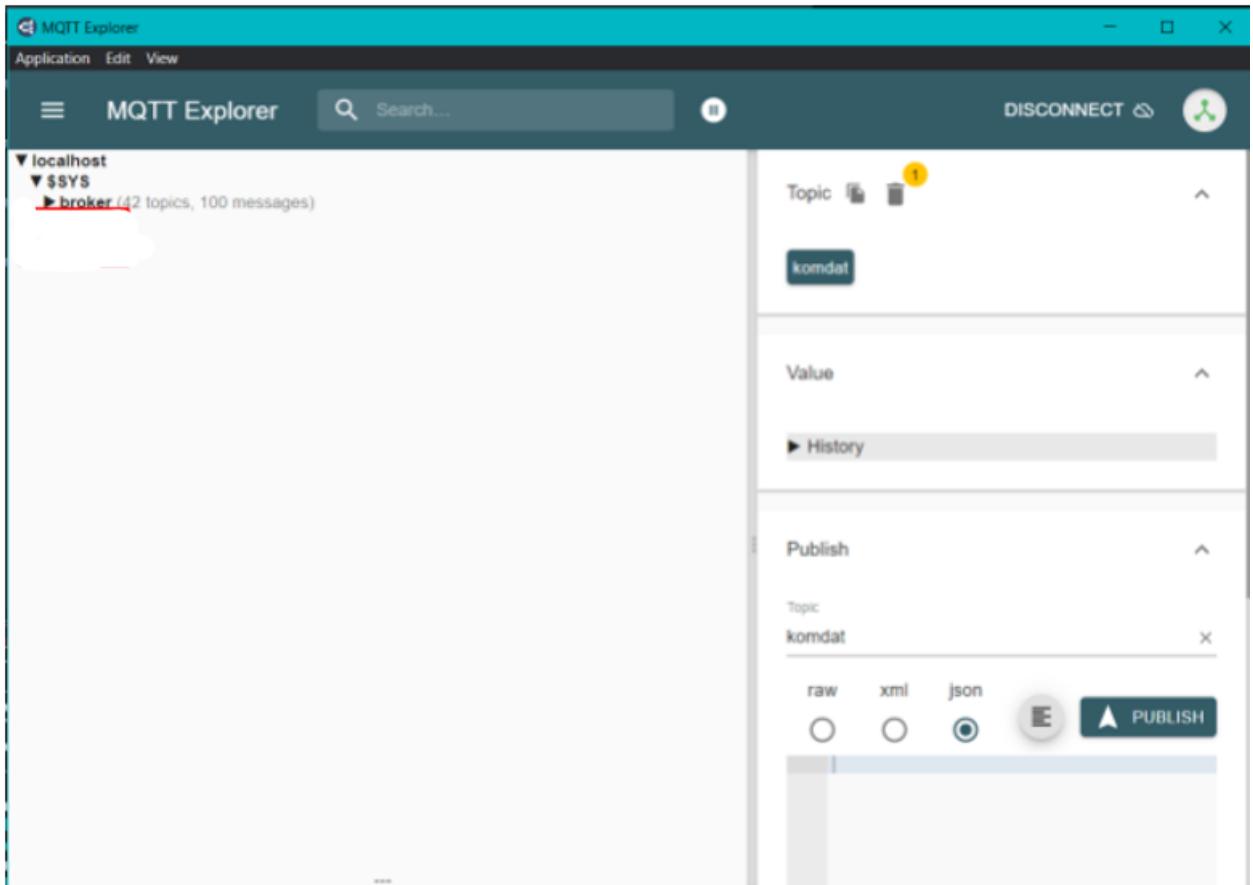
Microsoft Windows [Version 10.0.19045.3570]  
(c) Microsoft Corporation. All rights reserved.

C:\Users\MY PC>mosquitto -c (nama\_file.conf) -v

4. Buka aplikasi MQTT Explorer lalu buat *connection* baru pada tombol (+)



5. Isi *port* sesuai dengan listener yang sudah kalian buat pada file .conf, isi *host* dengan alamat IP laptop atau koneksi internet anda menggunakan *localhost*
6. Tekan tombol *connect* untuk membuat koneksi
7. Broker bekerja apabila pada MQTT Explorer menunjukkan jendela komunikasi



## Percobaan 05: Publisher MQTT

1. Buka VScode dan install library paho
2. Buat kode program *publisher MQTT* seperti gambar berikut

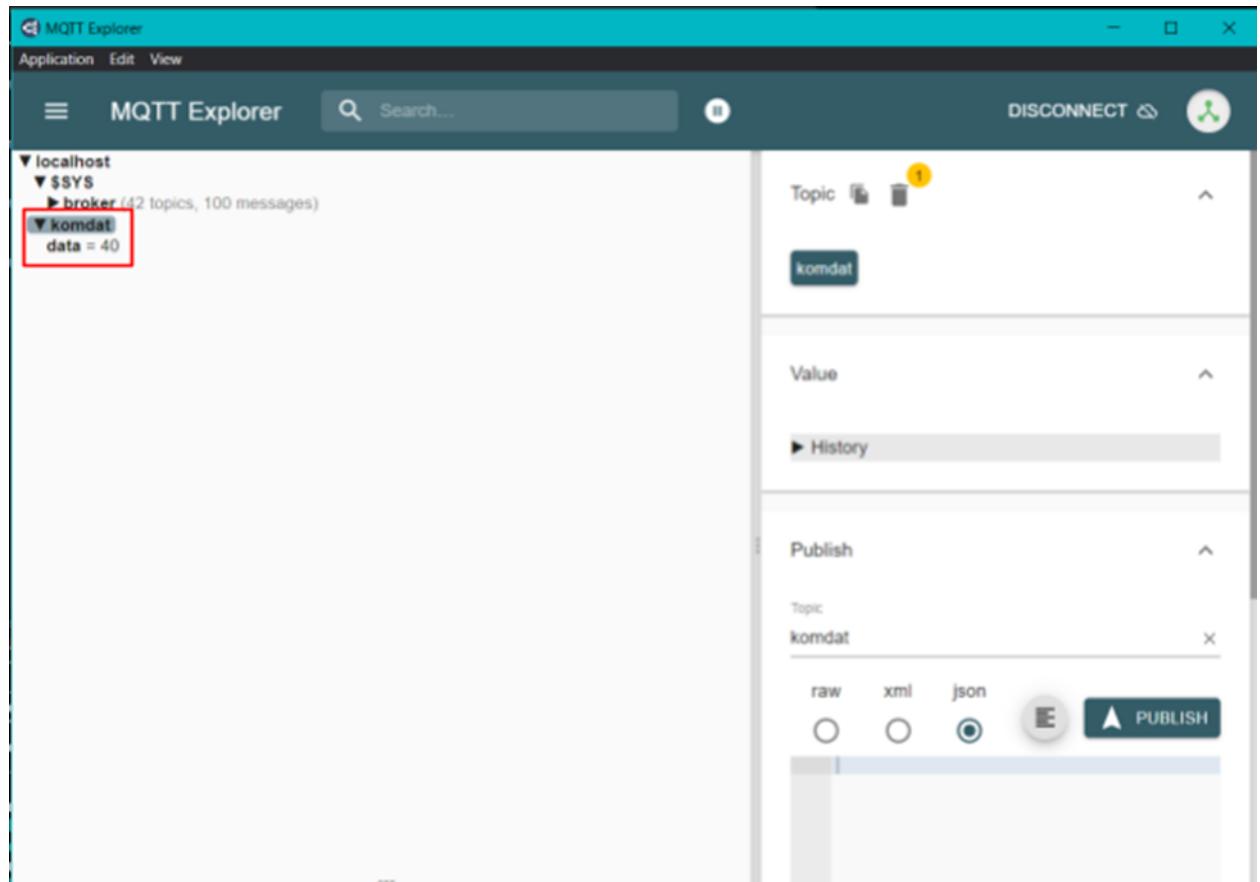
```
import time
import paho.mqtt.client as paho

broker="192.168.43.140"
dd = []

def on_message(client, userdata, message):
    print("topik",message.topic)
    print("data" ,str(message.payload.decode("utf-8")))

data = 40 #dummy data
client= paho.Client("langganan")
client.on_message=on_message
client.connect(Broker)
client.loop_start()
ta = time.time()
client.Publish("komdat/data",dd)
client.Subscribe("komdat/data") #Topic
time.sleep(0.001)
client.loop_stop()
client.disconnect()
```

3. Jalankan program *publisher MQTT* dan apabila tidak terjadi *error*
4. Buka *MQTT Explorer* dan pastikan *host* yang digunakan pada *MQTT Explorer* sama dengan alamat IP laptop atau koneksi internet anda menggunakan *localhost*
5. Protokol komunikasi *publisher* berhasil apabila jendela *MQTT Explorer* berhasil memunculkan data yang dikirimkan pada program *Python* yang anda buat



## Percobaan 06: Subscriber MQTT

1. Buka VScode dan buat kode program *subscriber MQTT* seperti gambar berikut

```

import time
import paho.mqtt.client as paho
#from kalkulasi import hitung

broker="192.168.43.140"

def on_message(client, userdata, message):
    print("topik ",message.topic)
    print("data " ,str(message.payload.decode("utf-6")))

#data=hitung(IbuKota,"Padang","Yogyakarta")
client= paho.Client("langganan")
client.on_message=on_message
client.Connect(Broker)
client.loop_start()
client.Subscribe("komdat/data")
#ta = time.time()
#client.publish("komdat/data",data)
#print((time.time()-ta)*1000)
time.sleep(10)

```

2. Jalankan kode program subscribe lalu lakukan hal yang sama seperti percobaan sebelumnya dengan membuka MQTT Explorer dan pastikan host dan port sama dengan yang digunakan pada broker
3. Protokol komunikasi subscriber berhasil apabila jendela MQTT Explorer berhasil memunculkan data yang dikirimkan pada program *Python* dari publisher

## Percobaan 07: Server WebSockets

1. Pastikan Anda telah menginstal python di sistem Anda.
2. Sekarang gunakan pip untuk menginstal paket WebSocket menggunakan perintah di bawah ini

```
pip install websockets
```

3. Buat file server misalnya “server.py”

4. Di dalam file server, import module yang diperlukan—dalam hal ini, `asyncIO` , dan `WebSockets` .
5. Tulis kode seperti dibawah:

```
1 import websockets
2 import asyncio
3
4 PORT = 5555
5 print("Server listening on Port " + str(PORT))
6
7 connected = set()
8
9 async def echo(websocket, path):
10     print("A client just connected")
11     connected.add(websocket)
12     try:
13         async for message in websocket:
14             print("Received message from client: " + message)
15             for conn in connected:
16                 if conn != websocket:
17                     await conn.send("Someone said: " + message)
18     except websockets.exceptions.ConnectionClosed as e:
19         print("A client just disconnected")
20     finally:
21         connected.remove(websocket)
22
23 start_server = websockets.serve(echo, "localhost", PORT)
24 asyncio.get_event_loop().run_until_complete(start_server)
25 asyncio.get_event_loop().run_forever()
```

6. Jalan kan kode yang telah dibuat
7. Server websocket siap digunakan

## Percobaan 08: Client WebSockets

1. Buat file baru untuk client.
2. Jalankan program python pada komputer

```
1 import asyncio
2 from websockets.sync import client
3
4 async def connect_to_server():
5     with client.connect('ws://127.0.0.1:5555') as websocket:
6         while True:
7             message = input("Enter message: ")
8             websocket.send(message)
9             response = websocket.recv()
10            print(response)
11
12
13 # Connect the WebSocket client
14 asyncio.get_event_loop().run_until_complete(connect_to_server())
```

### 3. Komunikasi antara komputer dengan komputer lainnya

#### Percobaan Bonus: WebSockets dengan modul socket

```
1 import socket, threading
2 nickname = input("Nama Kamu: ")
3
4 client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 client.connect(('localhost', 1405))
6
7 def receive():
8     while True:
9         try:
10             message = client.recv(1024).decode('ascii')
11             if message == 'NICKNAME':
12                 client.send(nickname.encode('ascii'))
13             else:
14                 print(message)
15         except:
16             print("Ada Kesalahan Teknis!")
17             client.close()
18             break
19 def write():
20     while True:
21         message = '{} ngomong: {}'.format(nickname, input(''))
22         client.send(message.encode('ascii'))
23
24 receive_thread = threading.Thread(target=receive)
25 receive_thread.start()
26 write_thread = threading.Thread(target=write)
27 write_thread.start()
```

```
● ● ●

1 import socket, threading
2
3 host = 'localhost'
4 port = 1405
5
6 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7 server.bind((host, port))
8 server.listen()
9
10 clients = []
11 nicknames = []
12
13 def broadcast(message):
14     for client in clients:
15         client.send(message)
16
17 def handle(client):
18     while True:
19         try:
20             message = client.recv(1024)
21             broadcast(message)
22         except:
23             index = clients.index(client)
24             clients.remove(client)
25             client.close()
26             nickname = nicknames[index]
27             broadcast('{} Keluar Coy!'.format(nickname).encode('ascii'))
28             nicknames.remove(nickname)
29             break
30
31 def receive():
32     while True:
33         client, address = server.accept()
34         print("Nyambung sama {}".format(str(address)))
35         client.send('NICKNAME'.encode('ascii'))
36         nickname = client.recv(1024).decode('ascii')
37         nicknames.append(nickname)
38         clients.append(client)
39         print("Nama dia {}".format(nickname))
40         client.send('Kamu telah koneksi ke server!'.encode('ascii'))
41         broadcast("{} bergabung ke tongkrongan!".format(nickname).encode('ascii'))
42         thread = threading.Thread(target=handle, args=(client,))
43         thread.start()
44
45 receive()
```

## Daftar Pustaka

- [1] RESLAB, “Mengenal Mqtt Protokol untuk IOT,” RESLAB, 23-Oct-2018. [Online]. Available:[http://reslab.sk.fti.unand.ac.id/index.php?option=com\\_k2&view=item&id=229%3Amengenal-mqtt-protokol-untuk-iot&Itemid=303](http://reslab.sk.fti.unand.ac.id/index.php?option=com_k2&view=item&id=229%3Amengenal-mqtt-protokol-untuk-iot&Itemid=303). [Accessed: 11-Oct-2021].
- [2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee,

- “Hypertext Transfer Protocol -- HTTP/1.1.” The Internet Society1, Jun. 1999, Accessed: May 01, 2020. [Online]. Available: <https://tools.ietf.org/html/rfc2616>
- [3] R. Fielding, J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.” Internet Engineering Task Force (IETF), Jun. 2014, Accessed: May 03, 2020. [Online]. Available: <https://tools.ietf.org/html/rfc7231>.
- [4] LJS, “What is HTTP?S2C home ” what is HTTP?,” LJS, 2020. [Online]. Available: <https://server2client.com/servlets/whatishttp.html>. [Accessed: 11-Oct-2021].
- [5] MQTT, “The standard for IOT messaging,” MQTT, 2020. [Online]. Available: <https://mqtt.org/>. [Accessed: 11-Oct-2021].
- [6] M. R. Adani, "Memahami Konsep Penggunaan Xampp untuk Kebutuhan Development," Sekawan Media, 26 April 2021. [Online]. Available: <https://www.sekawanmedia.co.id/apa-itu-xampp/>. [Accessed 19 Oktober 2021].
- [7] K. Das, "Introduction to Flask," pymbook, [Online]. Available: <https://pymbook.readthedocs.io/en/latest/flask.html>. [Accessed 19 Oktober 2021].