Laporan UTS IF4051 Pengembangan Sistem IoT

Jusuf Junior Athala -13519174

Sekolah Teknik Elektro dan
Informatika

Institut Teknologi Bandung, Jalan
Ganesha 10 Bandung

E-mail: 13519174@std.stei.itb.ac.id

***Abstract*—Internet of Things (IoT) dapat digunakan untuk mengendalikan perangkat elektronik dengan jangkauan yang luas karena memanfaatkan kekuatan internet yang dapat mencakup wilayah yang luas. Beberapa contoh perangkat elektronik yang dapat dikendalikan secara jarak jauh menggunakan Internet of Things adalah Air Conditioner (AC) dan lampu LED. Pada tugas ini, akan dibuat sebuah perangkat IoT yaitu lampu LED yang dapat dinyalakan dan dimatikan melalui aplikasi yang terhubung dengan internet. Perangkat IoT ini akan membutuhkan microcontroller, salah satu microcontroller yang umum digunakan adalah ESP32 dan memanfaatkan salah satu protokol komunikasi yang digunakan dalam IoT yaitu MQTT.**

Keywords— Internet of Things , LED, internet, aplikasi, ESP32, MQTT

I. INTRODUCTION

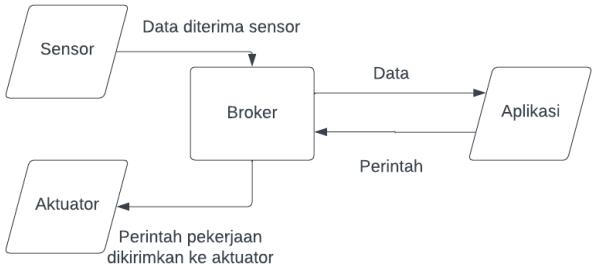
Saat ini, internet berpengaruh besar pada kehidupan sehari-hari. Internet mempermudah komunikasi dari satu tempat ke tempat lainnya dengan cepat dan murah. Namun, internet tidak hanya berperan sebagai perantara pembicaraan antara manusia. Internet juga dapat digunakan sebagai perantara kendali dari perangkat-perangkat elektronik. Inilah yang disebut dengan Internet of Things [1]. Dengan menggunakan internet, perangkat elektronik dapat dikendalikan secara jarak jauh asalkan perangkat elektronik tersebut dapat terhubung dengan internet, bahkan dapat dikendalikan dari luar negeri sekalipun. ESP 32 akan bertanggung jawab sebagai microcontroller yang dapat mengirimkan data hasil tangkapan sebuah sensor dan menerima perintah untuk mengaktifkan sebuah aktuator.

II. DASAR TEORI

Terdapat 3 layer dalam sebuah arsitektur IoT, yaitu :

1. Phyiscal layer. Pada layer ini terdapat sensor dan aktuator. Sensor yang berfungsi untuk mengambil data dari lingkungan dan aktuator yang berfungsi untuk melakukan sebuah pekerjaan berdasarkan data yang telah didapatkan dari sensor. Perangkat pada layer ini dapat berupa lampu LED.
2. Network layer atau dapat disebut broker. Layer ini berfungsi untuk menghubungkan perangkat-perangkat pintar, device, dan server. Disini terjadi pengiriman data dari sensor kepada perangkat lainnya. Salah satu protokol komunikasi yang dapat digunakan yaitu MQTT (Message Queuing Telemetry).
3. Application layer. Layer ini berupa layer yang dapat diakses seorang user. Layer ini dapat menampilkan data atau mengirimkan perintah kepada aktuator. Perangkat pada layer ini dapat berupa aplikasi smart phone.

Dengan mengikuti arsitektur IoT diatas dapat dibuat sebuah diagram yang akan menjelaskan dataflow dari sebuah perangkat ioT.



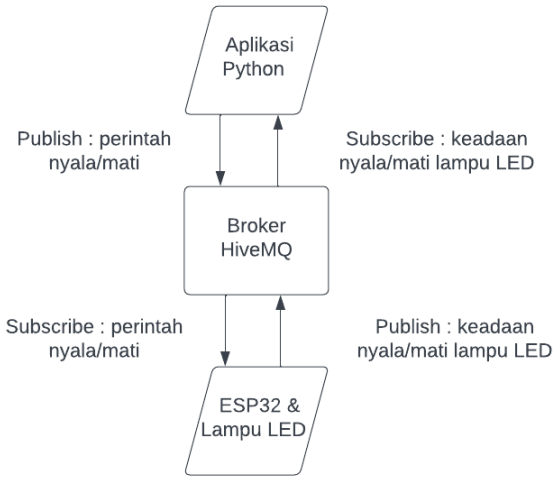
Gambar 1 Diagram dataflow perangkat IoT secara umum

Dalam MQTT, terdapat 4 komponen yaitu :

1. Klien, yang berfungsi sebagai berlangganan pada sebuah topik atau mengirimkan pesan tertentu.
2. Server/Broker, yang berfungsi untuk meneruskan pesan kepada client yang berlangganan pada sebuah topik tertentu.
3. Pesan, merupakan data yang dikirimkan dan diterima antar klien melalui Server.
4. Topik, label dari setiap pesan.

III. RANCANGAN PERANGKAT IOT

Tugas ini akan membuat sebuah perangkat IoT yang dapat menyalakan atau mematikan sebuah lampu LED melalui aplikasi yang terhubung dengan internet, dan aplikasi dapat menampilkan keadaan lampu LED saat itu. Oleh sebab itu, sensor yang ada pada tugas ini adalah interaksi pengguna aplikasi yang mengirimkan perintah nyala dan mati. Aktuator yang akan digunakan adalah lampu LED yang sudah terintegrasi langsung pada perangkat ESP32. Perangkat IoT akan menggunakan protokol MQTT yang memanfaatkan broker online yang sudah disediakan secara gratis yaitu HiveMQ. Aplikasi yang digunakan user untuk mematikan dan menyalakan lampu LED adalah aplikasi berbasis Python dengan memanfaatkan library Flask dan Paho-MQTT.



Gambar 2 Diagram flowchart perangkat IoT lampu LED

A. ESP32 & Lampu LED

Sebuah lampu LED terintegrasi pada perangkat ESP32. Untuk menyalakan atau mematikan lampu LED, dapat mengubah value pin nomor 2 pada ESP32.

Terdapat 2 file di ESP32 yaitu boot.py dan main.py. Berikut adalah kode dari masing-masing file.

```
#boot.py
import esp
import uos, machine
import gc
import network


def connect():
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        sta_if.active(True)
        sta_if.connect('Test', 'wifipass1')
        while not sta_if.isconnected():
            pass # wait till connection
    print('Connection success, network config:', sta_if.ifconfig())


#esp.osdebug(None)
gc.collect()
connect()
```

File boot.py ini berfungsi untuk menyambungkan ESP32 ke internet melalui koneksi Wi-Fi.

```
# main.py
#main.py
import machine
from time import sleep
from umqtt.simple import MQTTClient



sleep(3)
led = machine.Pin(2, machine.Pin.OUT)


CLIENT_NAME = b'esp32'
BROKER_ADDR = b'72f4e7db90564580a07b5ccf36eaab25.s2.eu.hivemq.cloud'
USER_NAME = b'itb13519174'
USER_PASS = b'itB13519174'
mqttc = MQTTClient(CLIENT_NAME,
                   BROKER_ADDR,
                   port=0,
                   user=USER_NAME,
                   password=USER_PASS,
                   keepalive=60,
                   ssl=True,

ssl_params={'server_hostname':'72f4e7db90564580a07b5ccf36eaab25.s2.eu.hivemq.cloud'})
mqttc.connect()


topic_state = 'hivemq/state'
topic_command = 'hivemq/command'
message_received = ""
def callback_prod(topic, msg):
    global message_received
    if msg.decode() == 'on':
        led.value(1)
        message_received = "ON"
    elif msg.decode() == 'off':
        led.value(0)
        message_received = "OFF"
    else : #commmand check
        message_received = "CHECK_STATE"
        pass


# mqtt subscription
mqttc.set_callback(callback_prod)
```

```
    mqttc.subscribe(topic_command)


    while True:
        mqttc.check_msg()
        if (message_received):
            mqttc.publish( topic_state, str(led.value()).encode() )
            message_received = ""
        sleep(0.5)
```

File main.py ini akan dimulai dengan melakukan koneksi pada broker yang telah ditentukan sebelumnya. Pada tugas ini akan digunakan broker online yang disediakan oleh HiveMQ. Ketika koneksi berhasil dilakukan, ESP32 akan subscribe kepada topic ‘hivemq/command’ dan melakukan looping untuk menunggu dan memeriksa jika ada pesan perintah menyalakan atau mematikan lampu LED yang dikirim oleh aplikasi. Jika diterima sebuah pesan perintah, ESP32 akan melakukan publish kepada topic ‘hivemq/state’ berisi pesan yang mengindikasikan keadaan lampu LED sedang menyala atau sedang mati.

B. Aplikasi Python

Aplikasi yang digunakan untuk mengirimkan perintah nyala atau mati lampu LED akan dibuat dengan berbasis Python yang menggunakan library Flask dan Paho-MQTT. Library Flask digunakan untuk dapat menghubungkan ke broker dan Paho-MQTT sebagai library MQTT yang akan digunakan. Berikut adalah kode dari aplikasi.

```
    #app.py
    from flask import Flask,render_template,request
    import paho.mqtt.client as mqtt
    import time


    app = Flask(__name__)


    message_received = ""
    state = ""
    def on_connect(client,userdata,flags,rc):
        print("Connected with result code "+str(rc))
        client.subscribe("hivemq/test")


    def on_message(client,userdata,msg):
        try:
            global message_received
            topic = msg.topic
            # print(topic)
            message_received = msg.payload.decode('utf-8')
            # print(message_received)
        except Exception as e:
            print("error",e)



    client= mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
    client.username_pw_set("itb13519174","itB13519174
")
    client.loop_start()
    client.connect("72f4e7db90564580a07b5ccf36eaab25.
s2.eu.hivemq.cloud",8883,60)


    @app.route('/', methods =["GET", "POST"])
    def index():
        #client.loop_forever()
        if request.method == 'POST':
            global state
            on_button = request.form.get("on_button")
            off_button = request.form.get("off_button")
            check_button = request.form.get("check_button")
            if on_button == 'TURN_ON':
                client.loop_start() #start the loop
                client.subscribe("hivemq/state")
                client.publish("hivemq/command",
payload="on", qos=1)


                time.sleep(3) # wait


                if message_received == "1":
                    state = "ON"
                elif message_received == "0":
                    state = "OFF"
                else:
                    state = "ERROR (Please CHECK_STATE)"
                    pass


            elif  off_button == 'TURN_OFF':
                client.loop_start() #start the loop
                client.subscribe("hivemq/state")
```

```
            client.publish("hivemq/command", payload="off", qos=1)


            time.sleep(3) # wait


            if message_received == "1":
               state = "ON"
            elif message_received == "0":
               state = "OFF"
            else:
               state = "ERROR (Please CHECK_STATE)"
               pass


         elif check_button == 'CHECK_STATE' :
            client.loop_start() #start the loop
            client.subscribe("hivemq/state")
            client.publish("hivemq/command", payload="check", qos=1)


            time.sleep(3) # wait


            if message_received == "1":
               state = "ON"
            elif message_received == "0":
               state = "OFF"
            else:
               state = "ERROR (Please CHECK_STATE)"
               pass


         else :
            pass
         client.loop_stop()
         return render_template('index.html', state=state)


      client.loop_stop()
   return render_template('index.html')
```

```
#index.html
    <!DOCTYPE html>
    <html lang="en">
    <head>
       <meta charset="UTF-8">
       <link rel="stylesheet" href="{{ url_for('static', filename= 'css/style.css') }}">
       <title>ESP</title>
    </head>
    <body>
       <h1>ESP Web App</h1>
       <p>Dibuat oleh 13519174 - Jusuf Junior Athala </p>
       <p>LED state: </p>
       <p> {{state}} </p>
       <form method="post" action="/">
          <input type="submit" value="CHECK_STATE" name="check_button" />
          <input type="submit" value="TURN_ON" name="on_button"/>
          <input type="submit" value="TURN_OFF" name="off_button" />
       </form>
       <br>
       <p>Cara menggunakan</p>
       <p>1. ESP32 harus sudah terhubung dengan internet dan terhubung dengan broker. </p>
       <p>2. Klik Tombol CHECK_STATE untuk memeriksa keadaan nyala/mati lampu LED</p>
       <p>3. Klik Tombol TURN_ON untuk menyalakan lampu LED</p>
       <p>4. Klik Tombol TURN_OFF untuk mematikan lampu LED</p>
       <p>5. Keadaan lampu LED akan diperiksa setiap kali tombol CHECK_STATE, TURN_ON, atau TURN_OFF ditekan </p>
       <p>6. Tombol CHECK_STATE perlu ditekan setiap tombol TURN_ON atau TURN_OFF ditekan agar aplikasi dapat
          menampilkan keadaan lampu LED yang tepat </p>
    </body>
</html>
```

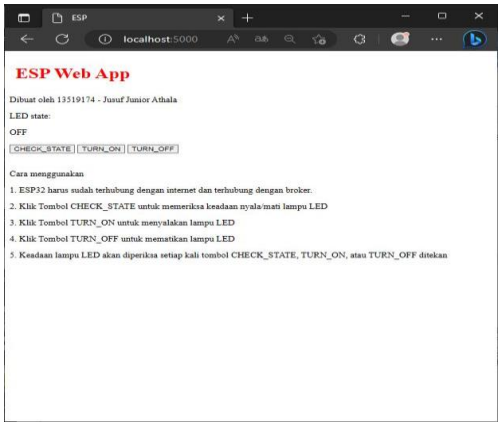
Aplikasi Python ini memiliki 2 file utama yaitu app.py dan index.html. Dengan menggunakan Flask, aplikasi akan diinisialisasi dengan membuat koneksi kepada broker online HiveMQ. Aplikasi akan melakukan subscribe kepada topic ‘hivemq/state’ agar dapat mengetahui keadaan lampu LED. Ketika sebuah tombol perintah ditekan oleh user, aplikasi akan melakukan publish kepada topic ‘hivemq/command’ yang berisi pesan sesuai dengan tombol perintah yang dapat berupa perintah menyalakan, mematikan, atau memeriksa keadaan lampu LED.

IV. Eksperimen dan Pembahasan Hasil

Aplikasi ini berhasil dibuat dan lampu LED dapat dinyalakan/dimatikan melalui aplikasi Python yang terhubung ke internet. Aplikasi juga dapat menampilkan keadaan nyala atau mati lampu LED.

Namun, koneksi internet sangat berpengaruh ketika aplikasi mencoba untuk memeriksa keadaan lampu LED tepat setelah perintah nyala atau mati di publish. Ini disebabkan oleh waktu menunggu yang terlalu cepat dibandingkan waktu untuk menerima pesan keadaan lampu LED yang dikirimkan dari ESP32, sehingga pesan tersebut tidak diterima dan aplikasi gagal menampilkan keadaan lampu LED yang tepat. Oleh sebab itu, Tombol CHECK_STATE perlu ditekan setiap tombol TURN_ON atau TURN_OFF ditekan agar aplikasi dapat menampilkan keadaan lampu LED yang tepat.

Berikut adalah UI dari aplikasi yang digunakan.



Gambar 3 Tampilan UI aplikasi

Berikut adalah link video demo eksperimen

https://drive.google.com/file/d/1PQ4n8hMeADAKrHkUDeYrNAVOmVQ7n9zb/view?usp=sharing

Berikut adalah link github untuk code project

https://github.com/jusufjathala/if4051-uts

V. Kesimpulan dan Saran

Dari tugas ini, dapat disimpulkan bahwa dengan menggunakan internet dan protokol MQTT, sebuah perangkat elektronik, pada khusus ini adalah lampu LED, dapat dinyalakan atau dimatikan melalui aplikasi yang juga terhubung dengan internet. Masih terdapat kelemahan dalam aplikasi ini, yaitu keadaan lampu LED tidak langsung tampil ter-update setelah tombol nyala atau mati ditekan. Hal ini dapat diperbaiki dengan menggunakan library MQTT yang lebih sesuai untuk tugas ini.

References

[1] Kumar, S., Tiwari, P. & Zymbler, M. Internet of Things is a revolutionary approach for future technology enhancement: a review. J Big Data 6, 111 (2019).

[2] Dimitrios Serpanos et. al, Internet-of-Things Internet-of-Things (IoT) Systems: Architectures, Algorithms, Methodologies(IoT) Systems, , Springer, 2018