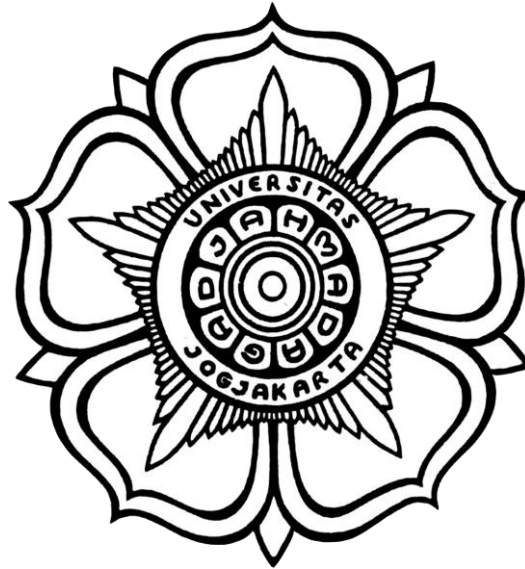


**LAPORAN PRAKTIKUM**  
**ELEKTRONIKA MESIN LISTRIK DAN TEKNIK KENDALI**

*“Internet of Things Using Protokol MQTT”*

Dosen Pengampu: Irfan Bahiuddin, ST, M.Phil., Ph.D.



Disusun Oleh:

Kelompok 4

Chaesar Syaefuddin (19/441195/SV/16547)

Yeyen Karunia (19/441215/SV/16567)

Kelas: ARM 2

**DEPARTEMEN TEKNIK MESIN**

**SEKOLAH VOKASI**

**UNIVERSITAS GADJAH MADA**

**YOGYAKARTA**

**2022**

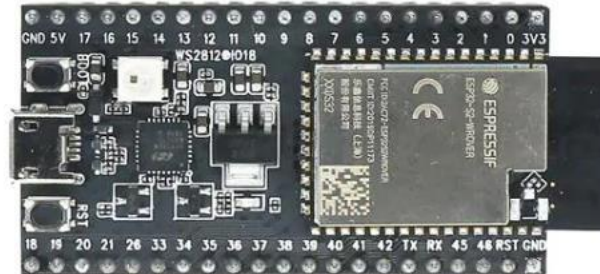
## BAB I. DESKRIPSI KASUS

Pada kasus tugas 2 ini akan membahas terkait penerapan protokol *Internet of Things* berbasis MQTT sehingga dari kasus ini dapat diperoleh beberapa langkah-langkah untuk penyelesaiannya.

## BAB II. KOMPONEN YANG DIGUNAKAN

### 2.1 Mikrokontroler ESP32

ESP 32 adalah mikrokontroler yang dikenalkan oleh Espressif System merupakan penerus dari mikrokontroler ESP8266. Pada mikrokontroler ini sudah tersedia modul WiFi dalam chip sehingga sangat mendukung untuk membuat sistem aplikasi *Internet of Things*. Terlihat pada gambar dibawah merupakan pin out dari ESP32. Pin tersebut dapat dijadikan input atau output untuk menyalakan LCD, lampu, bahkan untuk menggerakkan motor DC. Pada mikrokontroler ini sudah tersedia modul wifi dan bluetooth sehingga sangat mendukung untuk membuat sistem aplikasi Internet of Things. Memiliki 18 ADC (*Analog Digital Converter*), 2 DAC, 16 PWM, 10 Sensor sentuh, 2 jalur antarmuka UART, pin antarmuka I2C, I2S, dan SPI.



Gambar 1. Mikrokontroler Esp32

ESP32 menggunakan prosesor dual core yang berjalan di instruksi Xtensa LX16 [3], ESP32 memiliki spesifikasi seperti yang ditampilkan pada tabel 1.

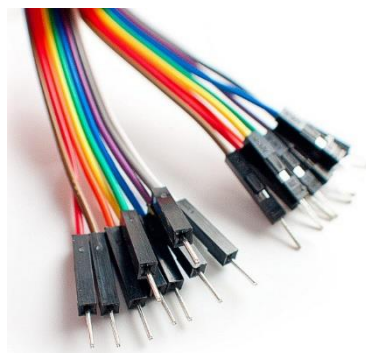
**Tabel 1. Spesifikasi Mikrokontroler Esp32**

Atribut	Detail
CPU	Tensilica Xtensa LX6 32bit Dual-Core di 160/240MHz
SRAM	520KB
FLASH	2MB (max. 64MB)

Tegangan	2.2V sampai 3.6V
Arus Kerja	Rata-rata 80mA
Dapat diprogram	Ya (C, C++, Python, Lua, dll)
Open Source	Ya
<b>Konektivitas</b>	
Wi-Fi	802.11 b/g/n
Bluetooth	4.2BR/EDR + BLE
UART	3
<b>I/O</b>	
GPIO	32
SPI	4
I2C	2
PWM	8
ADC	18 (12-bit)
DAC	2 (8-bit)

## 2.2 Kabel Jumper

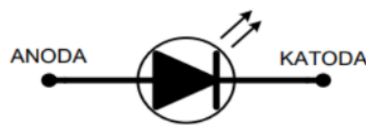
Kabel jumper adalah kabel yang digunakan untuk menghubungkan satu komponen dengan komponen lain ataupun menghubungkan jalur rangkaian yang terputus pada *breadboard*. Umumnya, kabel jumper digunakan pada *breadboard* atau alat prototyping lainnya agar lebih mudah untuk mengutak-atik rangkaian. Konektor yang ada pada ujung kabel terdiri atas dua jenis yaitu konektor jantan (*male connector*) dan konektor betina (*female connector*). Prinsip kerja kabel jumper yaitu menghantarkan arus listrik dari satu komponen ke komponen lainnya yang saling terhubung.



Gambar 2. Kabel Jumper

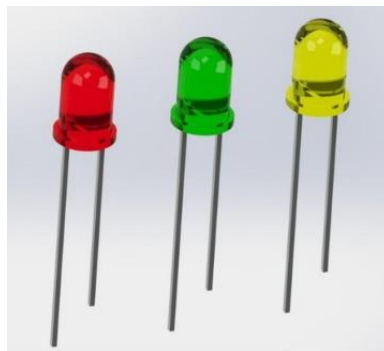
## 2.3 LED

*Light Emitting Diode* (LED) adalah komponen yang dapat memancarkan cahaya. Struktur LED sama dengan dioda. Untuk mendapatkan pancaran cahaya pada semikonduktor, dopping yang dipakai adalah gallium, arsenic, dan phosporus. Jenis dopping yang berbeda akan menghasilkan warna cahaya yang berbeda. Bentuk LED bermacam-macam, ada yang bulat, persegi empat dan lonjong. Simbol LED terlihat pada gambar 3.



Gambar 3. Simbol LED

LED memiliki kaki 2 buah seperti dengan dioda yaitu kaki anoda dan kaki katoda. Pada gambar diatas kaki anoda memiliki ciri fisik lebih panjang dari kaki katoda pada saat masih baru, kemudian kaki katoda pada LED ditandai dengan bagian body yang dipapas rata. Pemasangan LED agar dapat menyala adalah dengan memberikan tegangan bias maju yaitu dengan memberikan tegangan positif ke kaki anoda dan tegangan negatif ke kaki katoda. Konsep pembatas arus pada dioda adalah dengan memasang resistor secara seri pada salah satu kaki LED.

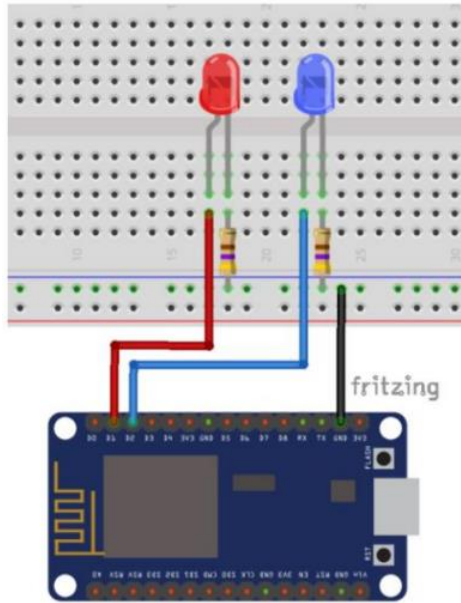


Gambar 4. LED

## BAB III. RANGKAIAN PROGRAM

### 3.1 Rangkaian ESP32

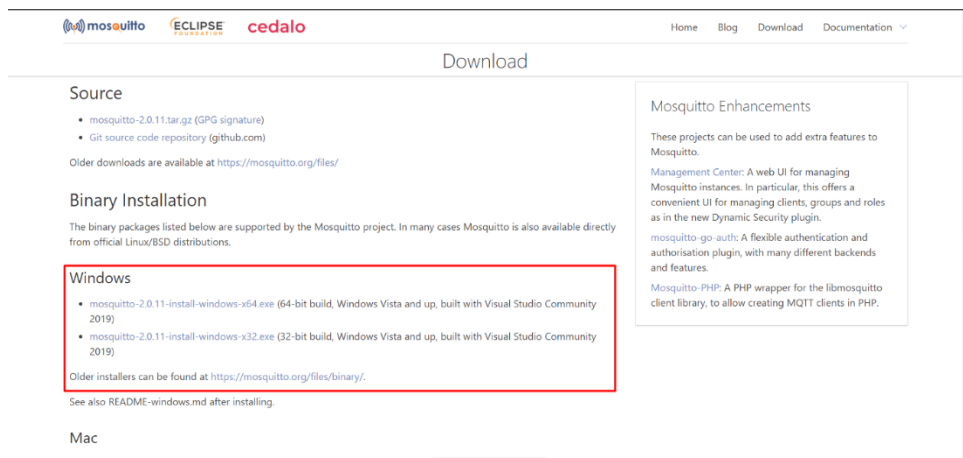
Pada kasus ini, rangkaian mikrokontroller Esp32 akan disusun dengan skema rangkaian seperti dibawah ini.



Gambar 5. Skema Rangkaian

### 3.2 Langkah-Langkah Pengerjaan

1. Pada langkah awal ini, kita harus mengunduh atau menginstall mosquitto pada halaman website <https://mosquitto.org/download/> lalu pilih untuk sistem operasi Windows dan pilih versi 64bi atau 32 bit yang sesuai dengan spesifikasi yang akan ditunjukkan pada Gambar dibawah ini.



Gambar 6. Download MQTT

MQTT singkatan dari *Message Queuing Telemetry Transport* merupakan protokol komunikasi yang berjalan pada stack TCP/IP yang dirancang khusus untuk komunikasi *machine to machine* yang tidak memiliki alamat khusus. Maksud dari kata tidak memiliki alamat khusus ini seperti halnya sebuah arduino, raspberry pi 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. Komunikasi MQTT berfungsi sebagai sistem publish dan subscribe. Perangkat tersebut akan memublikasikan pesan tentang topik tertentu. Dalam hal ini, Broker MQTT bertanggung jawab untuk menerima semua pesan, memfilter pesan, memutuskan siapa yang tertarik padanya, dan kemudian menerbitkan pesan ke semua klien yang berlangganan.

2. Langkah selanjutnya adalah dengan menginstall Phyton Web Server with Flask.

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get upgrade
pi@raspberrypi ~ $ sudo apt-get install python-pip python-flask
```

Gambar 7. Install Phyton Web Server with Flask

3. Langkah terakhir yaitu menginstall Phyton Paho-MQTT dan instalansi aplikasi-aplikasi penunjang. Paket Paho-MQTT menyediakan kelas klien yang memungkinkan aplikasi untuk terhubung ke broker MQTT untuk mempublikasikan pesan, dan untuk berlangganan topik dan menerima pesan yang dipublikasikan. Dalam contoh ini, server web Python akan mempublikasikan pesan ke ESP32 untuk mengaktifkan dan menonaktifkan GPIO. Agar dapat menginstal Paho-MQTT jalankan perintah seperti dibawah ini:

```
pip install paho-mqtt
```

Gambar 8. Install Paho-MQTT

### 3.3 Kode Program

- **ESP32 Code (Arduino)**

```
#include <WiFi.h>
#include <PubSubClient.h>
#define WIFI_TIMEOUT_MS 20000

// Change the credentials below, so your ESP8266 connects to
your router
const char* ssid = "UGM-Hotspot";
```

```
const char* password = "";

// Change the variable to your Raspberry Pi IP address, so it
// connects to
// your MQTT broker
const char* mqtt_server = "10.33.162.50";
// Initializes the espClient
WiFiClient espClient;
PubSubClient client(espClient);
// Connect an LED to each GPIO of your ESP8266
const int ledGPIO5 = 27;
const int ledGPIO4 = 26;
// Don't change the function below. This functions connects your
// ESP8266
// to your router
void connectToWiFi(){
  Serial.print("");
  Serial.println("Connecting to WiFi");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  unsigned long startAttemptTime = millis();
  while (WiFi.status() != WL_CONNECTED && millis () -
    startAttemptTime < WIFI_TIMEOUT_MS){
    Serial.print(".");
    delay(500);
  }
  if(WiFi.status() != WL_CONNECTED){
    Serial.println("Failed!");
  }
  else {
    Serial.print("Connected");
    Serial.println(WiFi.localIP());
  }
}
```

```

}
}

// This functions is executed when some device publishes a
message to a
topic that your ESP8266 is subscribed to
// Change the function below to add logic to your program, so
when a
device publishes a message to a topic that
// your ESP8266 is subscribed you can actually do something
void callback(String topic, byte* message, unsigned int length) {

    Serial.print("Message arrived on topic: ");
    Serial.print(topic);
    Serial.print(". Message: ");
    String messageTemp;
    for (int i = 0; i < length; i++) {
        Serial.print((char)message[i]);
        messageTemp += (char)message[i];
    }
    Serial.println();

    // Feel free to add more if statements to control more GPIOs with
MQTT
    // If a message is received on the topic home/office/esp1/gpio2,
you
check if the message is either 1 or 0. Turns the ESP GPIO
according to the
message
if(topic=="esp32/4"){
    Serial.print("Changing GPIO 4 to ");
    if(messageTemp == "1"){
        digitalWrite(ledGPIO4, HIGH);
        Serial.print("On");
    }
}
}

```



```

    }
    else if(messageTemp == "0"){
        digitalWrite(ledGPIO4, LOW);
        Serial.print("Off");
    }
}

if(topic=="esp32/5"){
    Serial.print("Changing GPIO 5 to ");
    if(messageTemp == "1"){
        digitalWrite(ledGPIO5, HIGH);
        Serial.print("On");
    }
    else if(messageTemp == "0"){
        digitalWrite(ledGPIO5, LOW);
        Serial.print("Off");
    }
}

Serial.println();
}

// This functions reconnects your ESP8266 to your MQTT
broker
// Change the function below if you want to subscribe to more
topics with
your ESP8266

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print(" Attempting MQTT connection...");
        // Attempt to connect

        if (client.connect("ESP32Client")) {

```

```
Serial.println("connected");  
// Subscribe or resubscribe to a topic  
// You can subscribe to more topics (to control more LEDs in  
this  
example)  
client.subscribe("esp32/4");  
client.subscribe("esp32/5");  
} else {  
Serial.print("failed, rc=");  
Serial.print(client.state());  
Serial.println("try again in 5 seconds");  
// Wait 5 seconds before retrying  
delay(5000);  
}  
}  
}  
  
// The setup function sets your ESP GPIOs to Outputs, starts the  
serial  
communication at a baud rate of 115200  
// Sets your mqtt broker and sets the callback function  
// The callback function is what receives messages and actually  
controls  
the LEDs  
void setup() {  
  
pinMode(ledGPIO4, OUTPUT);  
pinMode(ledGPIO5, OUTPUT);  
Serial.begin(115200);  
connectToWiFi();  
client.setServer(mqtt_server, 1883);  
client.setCallback(callback);  
}
```

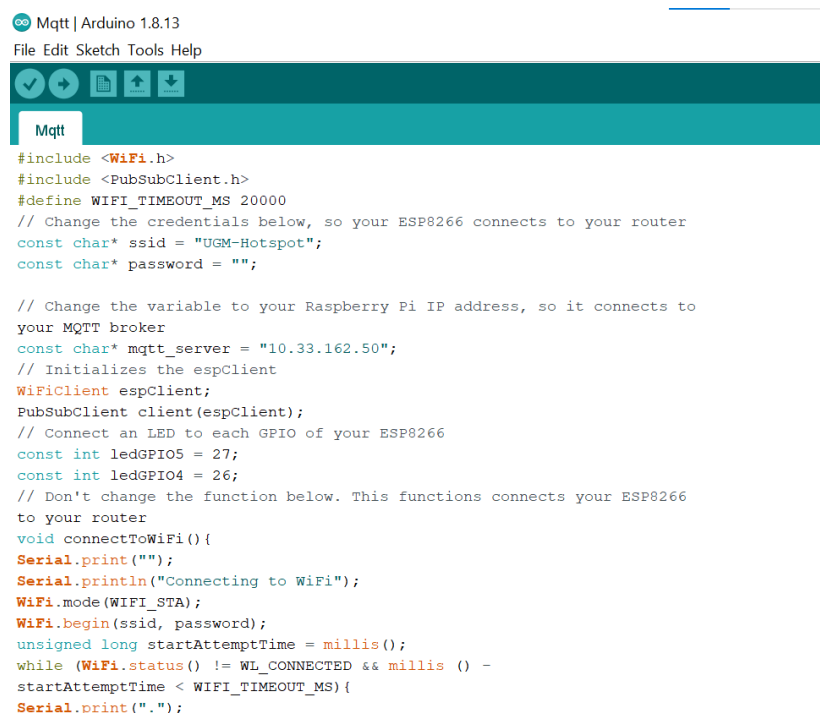
```

// For this project, you don't need to change anything in the loop
function.

// Basically it ensures that you ESP is connected to your broker
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  if(!client.loop())

  client.connect("ESP32Client");
}

```



```

Mqtt | Arduino 1.8.13
File Edit Sketch Tools Help

#include <WiFi.h>
#include <PubSubClient.h>
#define WIFI_TIMEOUT_MS 20000
// Change the credentials below, so your ESP8266 connects to your router
const char* ssid = "UGM-Hotspot";
const char* password = "";

// Change the variable to your Raspberry Pi IP address, so it connects to
your MQTT broker
const char* mqtt_server = "10.33.162.50";
// Initializes the espClient
WiFiClient espClient;
PubSubClient client(espClient);
// Connect an LED to each GPIO of your ESP8266
const int ledGPIO5 = 27;
const int ledGPIO4 = 26;
// Don't change the function below. This functions connects your ESP8266
to your router
void connectToWiFi(){
  Serial.print("");
  Serial.println("Connecting to WiFi");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  unsigned long startAttemptTime = millis();
  while (WiFi.status() != WL_CONNECTED && millis () -
startAttemptTime < WIFI_TIMEOUT_MS){
    Serial.print(".");

```

Gambar 8. Arduino Code

## ■ Python Flask Code

```

#
# Created by Rui Santos
# Complete project details: https://randomnerdtutorials.com
#
import paho.mqtt.client as mqtt

```

```

from flask import Flask, render_template, request
app = Flask(__name__)
mqttc=mqtt.Client()
mqttc.connect("10.33.162.50",1883,60)
mqttc.loop_start()
# Create a dictionary called pins to store the pin number, name,
and pin
state:
pins = {
4 : {'name' : 'GPIO 4', 'board' : 'esp32', 'topic' : 'esp32/4', 'state' :
'False'},
5 : {'name' : 'GPIO 5', 'board' : 'esp32', 'topic' : 'esp32/5', 'state' :
'False'}
}
# Put the pin dictionary into the template data dictionary:
templateData = {
'pins' : pins
}
@app.route("/")
def main():
# Pass the template data into the template main.html and return it
to the
user
return render_template('main.html', **templateData)
# The function below is executed when someone requests a URL
with the
pin number and action in it:

@app.route("/<board>/<changePin>/<action>")
def action(board, changePin, action):
# Convert the pin from the URL into an integer:
changePin = int(changePin)

```

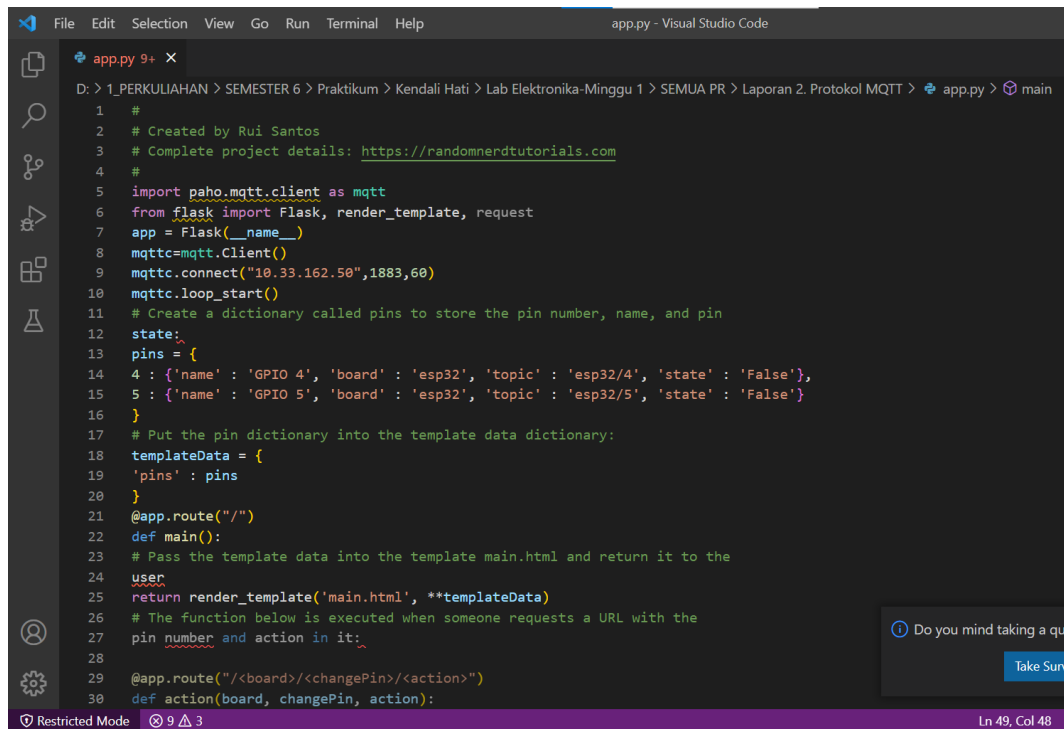
```
# Get the device name for the pin being changed:
devicePin = pins[changePin]['name']

# If the action part of the URL is "on," execute the code indented
below:
if action == "1" and board == 'esp32':
    mqttc.publish(pins[changePin]['topic'], "1")
    pins[changePin]['state'] = 'True'
if action == "0" and board == 'esp32':
    mqttc.publish(pins[changePin]['topic'], "0")
    pins[changePin]['state'] = 'False'

# Along with the pin dictionary, put the message into the
template data
dictionary:
templateData = {
    'pins': pins
}

return render_template('main.html', **templateData)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8080, debug=False)
```



```
1 #
2 # Created by Rui Santos
3 # Complete project details: https://randomnerdtutorials.com
4 #
5 import paho.mqtt.client as mqtt
6 from flask import Flask, render_template, request
7 app = Flask(__name__)
8 mqttc=mqtt.Client()
9 mqttc.connect("10.33.162.50",1883,60)
10 mqttc.loop_start()
11 # Create a dictionary called pins to store the pin number, name, and pin
12 state:
13 pins = {
14 4 : {'name' : 'GPIO 4', 'board' : 'esp32', 'topic' : 'esp32/4', 'state' : 'False'},
15 5 : {'name' : 'GPIO 5', 'board' : 'esp32', 'topic' : 'esp32/5', 'state' : 'False'}
16 }
17 # Put the pin dictionary into the template data dictionary:
18 templateData = {
19 'pins' : pins
20 }
21 @app.route("/")
22 def main():
23 # Pass the template data into the template main.html and return it to the
24 user
25 return render_template('main.html', **templateData)
26 # The function below is executed when someone requests a URL with the
27 pin number and action in it:
28
29 @app.route("/<board>/<changePin>/<action>")
30 def action(board, changePin, action):
```

Gambar 9. Python Flask Code

- **Main HTML**

```
<head>
<title>RPi Web Server</title>
<!-- Latest compiled and minified CSS -->

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.
css" integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fg
jWPGmkzs7" crossorigin="anonymous">

<!-- Optional theme -->
<link rel="stylesheet"

href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css" integrity="sha384-

fLW2N01lMqjakBkx3l/M9EahuwpsFeNvV63J5ezn3uZzapT0u7EYsXMj
```

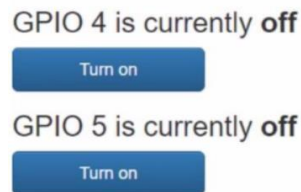
```
QV+0En5r" crossorigin="anonymous">
<!-- Latest compiled and minified JavaScript -->
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-
0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepn
HVP9aJ7xS" crossorigin="anonymous"></script>

<meta name="viewport" content="width=device-width, initial-
scale=1">

</head>
<body>
<h1>RPi Web Server - ESP32 MQTT</h1>
{% for pin in pins %}
<h2>{{ pins[pin].name }}
{% if pins[pin].state == 'True' %}
is currently <strong>on</strong></h2><div class="row"><div
class="col-md-2">
<a href="/esp32/{{ pin }}/0" class="btn btn-block btn-lg btn-default"
role="button">Turn off</a></div></div>
{% else %}
is currently <strong>off</strong></h2><div class="row"><div
class="col-md-2">
<a href="/esp32/{{ pin }}/1" class="btn btn-block btn-lg btn-primary"
role="button">Turn on</a></div></div>
{% endif %}
{% endfor %}
</body>
</html>
```

- **Web Server**

Agar dapat membuka alamat web server yang telah dirancang, diperlukan penyambungan IP Address serta penyesuaian port yang akan dipakai. Pada tampilan web server ini, button “Turn On” mampu dioperasikan untuk menghubungkan rangkaian ESP32 serta memberikan perintah on/off pada kedua LED yang terhubung.



Gambar 10. Tampilan Web Server

## **BAB IV. KESIMPULAN**

Pada project ini, Server Web Python akan mempublikasikan pesan ke ESP32 untuk mengaktifkan dan menonaktifkan GPIO. Fungsi MQTT adalah untuk sistem publish dan subscribe. Perangkat mempublikasikan pesan tentang topik tertentu. Semua perangkat yang berlangganan topik tersebut menerima pesan tersebut.

## **DAFTAR PUSTAKA**

Nusyirwan, D. 2019. “FUN BOOK” RAK BUKU OTOMATIS BERBASIS ARDUINO DAN BLUETOOTH PADA PERPUSTAKAAN UNTUK MENINGKATKAN KUALITAS SISWA. *Jurnal Ilmiah Pendidikan Teknik dan Kejuruan*, 12(2), 94-106.