LAPORAN PRAKTIKUM TEKNIK KENDALI DAN MESIN LISTRIK

“ Python Flask Publishing MQTT Message to ESP32 ”

DISUSUN OLEH:

Kelompok 2

ARM 2

Ade Surya Pramudya	(19/441188/SV/16540)
Mahesa Audriansyah Agatha	(19/441203/SV/16555)

TEKNOLOGI REKAYASA MESIN

DEPARTEMEN TEKNIK MESIN SEKOLAH VOKASI

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2022

I. Deskripsi Kasus

Dalam proyek ini kami akan membuat server web mandiri dengan microframework Phyton Flask yang dapat mengoperasikan dua LED dari ESP32 menggunakan protokol MQTT.

II. Komponen dan Ekstensi aplikasi yang digunakan

1. Komponen yang digunakan
   - ESP32 board dengan chip ESP-WROOM-32
   - Resistor
   - Kabel jumper
   - LED 2 buah
2. Persiapan Ekstensi aplikasi yang digunakan
   A. Install dan jalankan Mosquitto Broker

   MQTT adalah singkatan dari Message Queuing Telemetry Transport. Mosquitto MQTT adalah protokol pesan sederhana, yang dirancang untuk perangkat terbatas dengan bandwidth rendah. Jadi, ini adalah solusi sempurna untuk bertukar data antara beberapa perangkat IoT.

   Komunikasi MQTT berfungsi sebagai sistem publish dan subscribe. Perangkat memublikasikan pesan tentang topik tertentu. Semua perangkat yang berlangganan topik tersebut menerima pesan tersebut.

   Broker MQTT bertanggung jawab untuk menerima semua pesan, memfilter pesan, memutuskan siapa yang tertarik padanya, dan kemudian menerbitkan pesan ke semua klien yang berlangganan.

   B. Python Web Server with Flask

   Untuk menginstal Flask, kami harus menginstal pip.

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

   Kemudian, kami menggunakan pip untuk menginstal Flask dan dependensinya

```
pi@raspberrypi ~ $ sudo pip install flask
```

   C. Install Python Paho-MQTT

   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.

   Untuk menginstal paho-mqtt jalankan perintah berikut

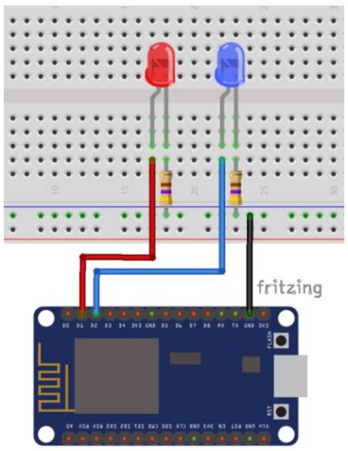
```
pip install paho-mqtt
```

D. Instal Aplikasi Penunjang

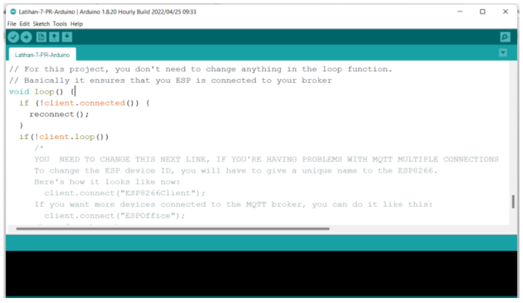
Kami menggunakan Visual Studio Code untuk menunjang pembuatan coding python dan html yang akan dibuat.

III. Tahap Pengerjaan

1. Pembuatan Skema Rangkaian



2. Pembuatan ESP32 code di Arduino IDE


Latihan-7-PR-Arduino | Arduino 1.8.20 Hourly Build 2022/04/25 09:33
File Edit Sketch Tools Help
Latihan-7-PR-Arduino

```
// 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())
     /*
     YOU  NEED TO CHANGE THIS NEXT LINE, IF YOU'RE HAVING PROBLEMS WITH MQTT MULTIPLE CONNECTIONS
     To change the ESP device ID, you will have to give a unique name to the ESP8266.
     Here's how it looks like now:
       client.connect("ESP8266Client");
     If you want more devices connected to the MQTT broker, you can do it like this:
       client.connect("ESPOffice");
```

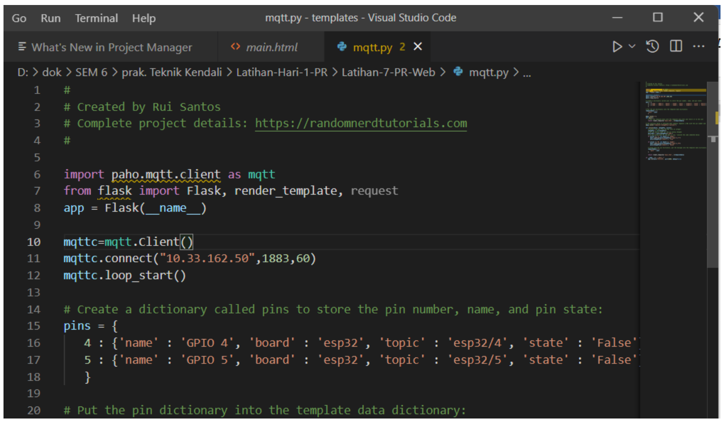
```
// Loading the ESP32 WiFi library and the PubSubClient library
#include <WiFi.h>
#include <PubSubClient.h>
#define WIFI_TIMEOUT_MS 20000

// 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
     /*
     YOU  NEED TO CHANGE THIS NEXT LINE, IF YOU'RE HAVING
PROBLEMS WITH MQTT MULTIPLE CONNECTIONS
     To change the ESP device ID, you will have to give a unique name to the
ESP8266.
```

```
     Here's how it looks like now:
       if (client.connect("ESP8266Client")) {
     If you want more devices connected to the MQTT broker, you can do it like
this:
       if (client.connect("ESPOffice")) {
     Then, for the other ESP:
       if (client.connect("ESPGarage")) {
      That should solve your MQTT multiple connections problem

     THE SECTION IN loop() function should match your device name
    */
    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())
     /*
     YOU  NEED TO CHANGE THIS NEXT LINE, IF YOU'RE HAVING
PROBLEMS WITH MQTT MULTIPLE CONNECTIONS
     To change the ESP device ID, you will have to give a unique name to the
ESP8266.
     Here's how it looks like now:
       client.connect("ESP8266Client");
     If you want more devices connected to the MQTT broker, you can do it like
this:
       client.connect("ESPOffice");
     Then, for the other ESP:
       client.connect("ESPGarage");
      That should solve your MQTT multiple connections problem

     THE SECTION IN recionnect() function should match your device name
    */
    client.connect("ESP32Client");
}
```

3. Pembuatan Python Script MQTT di Visual Studio Code
   Ini adalah skrip inti dari aplikasi kami. Ini mengatur server web dan ketika tombol-tombol ini ditekan, ia menerbitkan pesan MQTT ke ESP32.

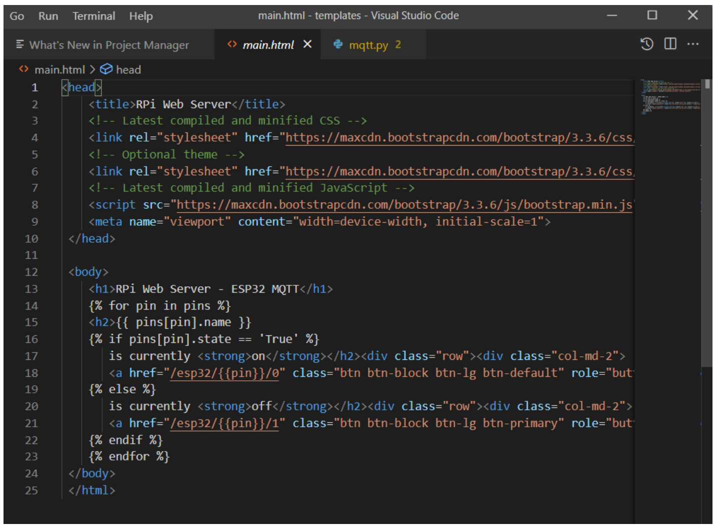

Go
Run
Terminal
Help
mqtt.py - templates - Visual Studio Code
What's New in Project Manager
main.html
mqtt.py 2
D: > dok > SEM 6 > prak. Teknik Kendali > Latihan-Hari-1-PR > Latihan-7-PR-Web > mqtt.py > ...

```
#
# 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:
```

```
#
# 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)
```

4. Pembuatan File HTML

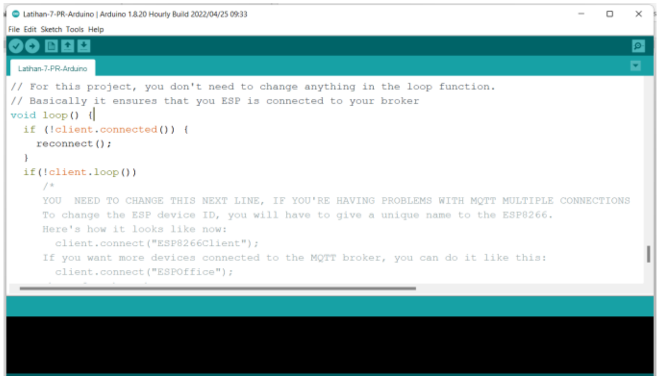


```
<head>
    <title>RPi Web Server</title>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css
    <!-- Optional theme -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css
    <!-- Latest compiled and minified JavaScript -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js
    <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="but
    {% 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="but
    {% endif %}
    {% endfor %}
</body>
</html>
```

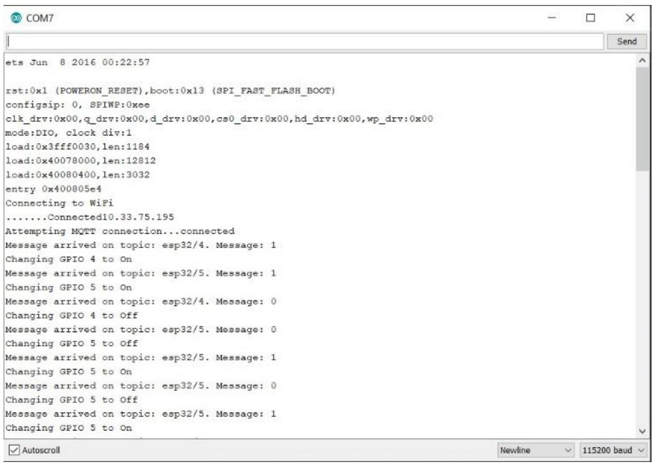
```
<head>
    <title>RPi Web Server</title>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjW
PGmkzs7" crossorigin="anonymous">
    <!-- Optional theme -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css" integrity="sha384-
fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV
+0En5r" crossorigin="anonymous">
    <!-- Latest compiled and minified JavaScript -->
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-
0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHV
P9aJ7xS" 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>
```

IV. Upload Program dan Launch the Web Server

1. Upload ESP32 Program Code di Arduino IDE

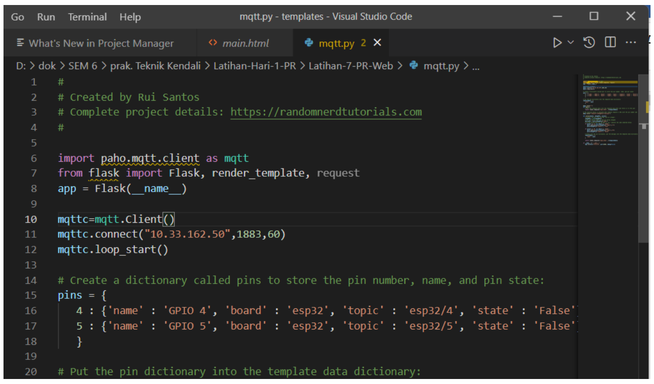

Latihan-7-PR-Arduino | Arduino 1.8.20 Hourly Build 2022/04/25 09:33
File Edit Sketch Tools Help
Latihan-7-PR-Arduino

```
// 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())
    /*
    YOU  NEED TO CHANGE THIS NEXT LINE, IF YOU'RE HAVING PROBLEMS WITH MQTT MULTIPLE CONNECTIONS
    To change the ESP device ID, you will have to give a unique name to the ESP8266.
    Here's how it looks like now:
      client.connect("ESP8266Client");
    If you want more devices connected to the MQTT broker, you can do it like this:
      client.connect("ESPOffice");
```

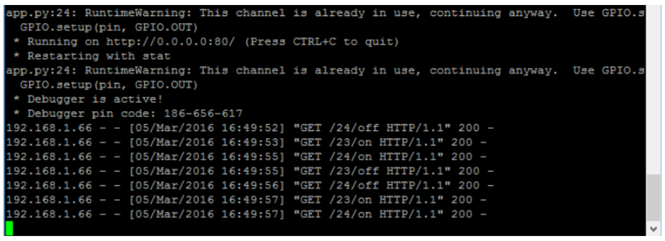


```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1184
load:0x40078000,len:12812
load:0x40080400,len:3032
entry 0x400805e4
Connecting to WiFi
.......Connected10.33.75.195
Attempting MQTT connection...connected
Message arrived on topic: esp32/4. Message: 1
Changing GPIO 4 to On
Message arrived on topic: esp32/5. Message: 1
Changing GPIO 5 to On
Message arrived on topic: esp32/4. Message: 0
Changing GPIO 4 to Off
Message arrived on topic: esp32/5. Message: 0
Changing GPIO 5 to Off
Message arrived on topic: esp32/5. Message: 1
Changing GPIO 5 to On
Message arrived on topic: esp32/5. Message: 0
Changing GPIO 5 to Off
Message arrived on topic: esp32/5. Message: 1
Changing GPIO 5 to On
```

2. Run mqtt program with pyhton app.py

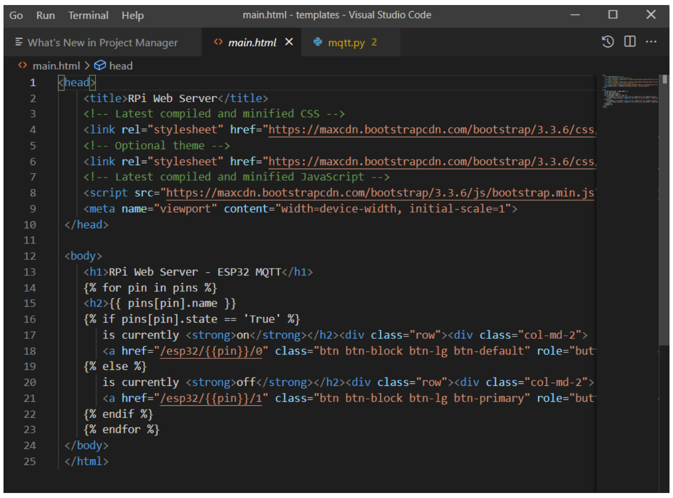


```
#
# Created by Rui Santos
# Complete project details: https://randomnerdtutorials.com
#

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:
```



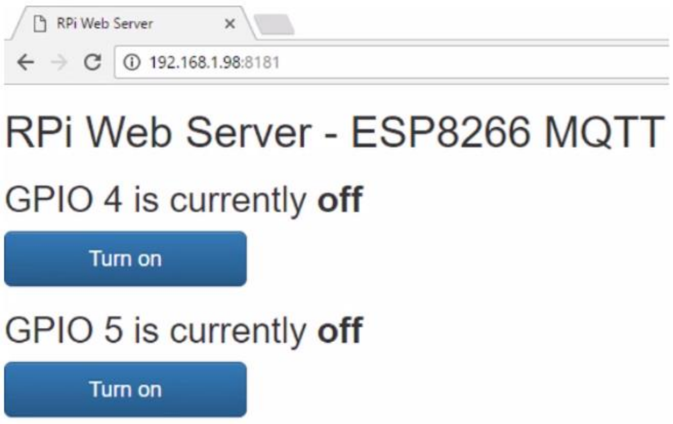
```
app.py:24: RuntimeWarning: This channel is already in use, continuing anyway.  Use GPIO.s
  GPIO.setup(pin, GPIO.OUT)
 * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
 * Restarting with stat
app.py:24: RuntimeWarning: This channel is already in use, continuing anyway.  Use GPIO.s
  GPIO.setup(pin, GPIO.OUT)
 * Debugger is active!
 * Debugger pin code: 186-656-617
192.168.1.66 - - [05/Mar/2016 16:49:52] "GET /24/off HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:53] "GET /23/on HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:55] "GET /24/on HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:55] "GET /23/off HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:56] "GET /24/off HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:57] "GET /23/on HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:57] "GET /24/on HTTP/1.1" 200 -
```

3. Run HTML code di Visual Studio Code



```
<head>
    <title>RPi Web Server</title>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css
    <!-- Optional theme -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css
    <!-- Latest compiled and minified JavaScript -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js
    <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="but
    {% 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="but
    {% endif %}
    {% endfor %}
 </body>
 </html>
```

4. Demonstrasi Web Server



RPi Web Server
192.168.1.98:8181

RPi Web Server - ESP8266 MQTT

GPIO 4 is currently **off**

GPIO 5 is currently **off**

Untuk membuka alamat web server yang dibuat, kami menggukan ip addres dan penyesuaian port yang dipakai.

kedua button “**Turn on**” bisa dioperasikan untuk connect ke rangkaian ESP32 yang memberikan perintah on/off pada kedua LED yang tersambung.
Status “**Turn on**”dan “**off**” pada web akan berubah menjadi “**Turn Off**”dan “**on**”jika kami tekan tombol **Turn on.**