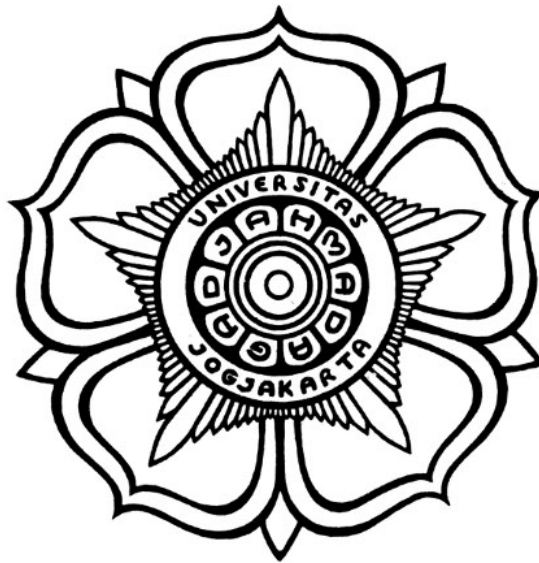


LAPORAN PRAKTIKUM

SISTEM TERTANAM DAN IOT



Disusun oleh:
AFRIZAL DANI SAOQI
17/413500/TK/45940

DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI
FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA
YOGYAKARTA

2020

I

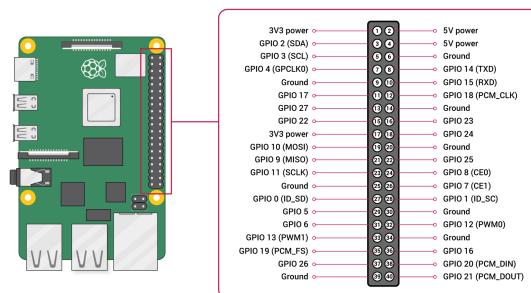
Pengenalan

1.1 Raspberry Pi

Raspberry Pi merupakan SBC (*Single Board computer*) yang dikembangkan oleh Raspberry Pi Foundation. Pada praktikum ini saya menggunakan Raspberry Pi 3 Model B. Berikut spesifikasi Raspberry Pi 3B dan Pinout GPIO pada Raspberry Pi 3B.

<i>SOC</i>	Broadcom BCM2837
<i>CPU</i>	ARM Cortex A53 64Bit @1.2GHz
<i>GPU</i>	VideoCore IV @400MHz
<i>RAM</i>	1GB LPDDR2 @900MHz
<i>Ethernet</i>	10/100Mbps Ethernet
<i>Wifi</i>	2.4GHz IEEE 802.11n
<i>Bluetooth</i>	Bluetooth 4.1 Classic, Bluetooth Low Energy.
<i>Storage</i>	MicroSD
<i>GPIO</i>	40-pin header
<i>MaxPower</i>	2.5A @5V
<i>Ports</i>	4x 2.0 USB Port
<i>VideoOutput</i>	1x HDMI

Tabel 1.1: Spesifikasi Raspberry Pi 3B



Gambar 1.1: Pinout Raspberry Pi 3B

1.2 NodeMCU

NodeMCU adalah mikrokontroler yang dilengkapi dengan modul wifi esp8266. Secara fungsi NodeMCU mirip dengan Arduino, hanya saja NodeMCU sudah dilengkapi dengan wifi.

1.3 Thingsboard

Thingsboard merupakan salah satu IoT platform yang open source. Fitur yang terdapat pada thingsboard dapat mempermudah pengguna dalam pengembangan produk, manajemen maupun *scaling* produk. Terdapat 9 menu pada halaman *home*.

1. HOME

2. RULE CHAINS

3. CUSTOMERS

4. ASSETS

5. DEVICES

Pada menu ini, pengguna dapat mendaftarkan *device* yang akan digunakan.

6. ENTITY VIEWS

7. WIDGETS LIBRARY

8. DASHBOARDS Pada menu ini, pengguna dapat membuat tampilan *dashboard* yang diinginkan

9. AUDIT LOGS

1.4 Protokol Komunikasi

1.4.1 MQTT

MQTT atau *Message Queuing Telemetry Transport* merupakan salah satu protokol komunikasi yang cukup terkenal. MQTT biasanya digunakan dalam komunikasi yang memiliki *bandwidth* terbatas. Hal tersebut karena MQTT didesain untuk komunikasi data kecil. Pada MQTT terdapat dua istilah yang sering digunakan

1. Client/Broker

Client merupakan device yang terkoneksi pada broker

2. Publish/Subscribe

Publish merupakan salah satu metode pada MQTT yang berguna untuk mengirim data ke broker. Subscribe merupakan salah satu metode pada MQTT yang berguna untuk menerima data dari broker.

1.4.2 HTTP

HTTP atau *Hypertext Transfer Protocol* merupakan salah satu protokol komunikasi yang biasa digunakan dalam membuat *web*. Ada 2 metode yang biasanya digunakan pada http

1. GET

Metode ini digunakan untuk meminta data dari server

2. POST

Metode ini digunakan untuk mengirim data ke server

1.4.3 Perbedaan MQTT dan HTTP

<i>Criteria</i>	MQTT	HTTP
<i>Architecture</i>	Client/Broker	Client/Server
<i>Abstraction</i>	Publish/Subscribe	Request/Response
<i>Header Size</i>	2 Byte	Tidak terdefinisi
<i>Message Size</i>	Maksimum 256MB	Bisa sangat besar, bergantung pada teknologi yang digunakan
<i>Methode</i>	Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close	Get, Post, Head, Put, Patch, Options, Connect, Delete
<i>Quality of Service</i>	QoS 0 QoS 1 QoS 2	Terbatas
<i>Transport Protocol</i>	TCP (MQTT-SN)	TCP
<i>Security</i>	TLS/SSL	TLS/SSL
<i>Port</i>	1883/8883	80/443

II

Pembahasan

2.1 Praktikum Minggu I

2.1.1 Percobaan 1

Pada percobaan ini, praktikan membuat *device* led pada Thingsboard. Setelah membuat *device* maka akan mendapatkan akses token. Token tersebut berguna untuk mengakses *device* tersebut.

2.1.2 Percobaan 2

Pada percobaan ini, praktikan menginstall *board* NodeMCU dan beberapa *library* yang akan digunakan. Terdapat 3 *library* yang digunakan, antara lain:

1. ArduinoJSON
2. PubSubClient
3. ESP8266Wifi

Source Code

```
1 #include <ArduinoJson.h>
2 #include <PubSubClient.h>
3 #include <ESP8266WiFi.h>
```

Kode di atas digunakan untuk menambahkan *library* yang akan digunakan.

```
2 #define WIFI_AP "F"
3 #define WIFI_PASSWORD "1234567890"
4 #define TOKEN "303hoMJLhyZfhSbwHZLc"
5 #define GPIO0 D3
6 #define GPIO2 D4
7 #define GPIO0_PIN 1 //ThingsBoard pin
8 #define GPIO2_PIN 2 //ThingsBoard pin
9 char thingsboardServer[] = "demo.thingsboard.io";
```

Kode di atas digunakan untuk mendefinisikan sebuah konstanta dan variable.

```

3 void setup()
4 {
5     Serial.begin(115200);
6     // Set output mode for all GPIO pins
7
8     delay(10);
9     InitWiFi();
10
11     pinMode(GPIO0, OUTPUT);
12     pinMode(GPIO2, OUTPUT);
13     client.setServer(thingsboardServer, 1883);
14     client.setCallback(on_message);
15 }

```

Kode di atas digunakan untuk men-*setup* pin, komunikasi serial, komunikasi MQTT, wifi yang akan digunakan.

```

4 void loop()
5 {
6     if (!client.connected())
7     {
8         reconnect();
9     }
10
11     client.loop();
12 }

```

Kode di atas digunakan untuk menghubungkan kembali komunikasi MQTT jika terputus.

```

5 void on_message(const char *topic, byte *payload, unsigned int length
6 )
7 {
8
9     Serial.println("On message");
10
11     char json[length + 1];
12     strncpy(json, (char *)payload, length);
13     json[length] = '\0';
14
15     Serial.print("Topic: ");
16     Serial.println(topic);

```

```

12     Serial.print("Message: ");
13     Serial.println(json);
14
15     // Decode JSON request
16     StaticJsonBuffer<200> jsonBuffer;
17     JsonObject &data = jsonBuffer.parseObject((char *)json);
18
19     if (!data.success())
20     {
21         Serial.println("parseObject() failed");
22         return;
23     }
24
25     // Check request method
26     String methodName = String((const char *)data["method"]);
27
28     if (methodName.equals("getGpioStatus"))
29     {
30         // Reply with GPIO status
31         String responseTopic = String(topic);
32         responseTopic.replace("request", "response");
33         client.publish(responseTopic.c_str(), get_gpio_status().c_str
34         ());
35     }
36     else if (methodName.equals("setGpioStatus"))
37     {
38         // Update GPIO status and reply
39         set_gpio_status(data["params"]["pin"], data["params"]["
40         enabled"]);
41         String responseTopic = String(topic);
42         responseTopic.replace("request", "response");
43         client.publish(responseTopic.c_str(), get_gpio_status().c_str
44         ());
45         client.publish("v1/devices/me/attributes", get_gpio_status().
46         c_str());
47     }
48 }

```

Kode di atas digunakan untuk mendapatkan pesan dari sebuah topik pada komunikasi MQTT. Pesan tersebut diolah, kemudian didapatkan nilai *boolean*. Nilai *boolean* tersebut digunakan untuk men-set gpio.

2.1.3 Percobaan 3

Pada percobaan ini, praktikan membuat dashboard yang digunakan untuk mengendalikan led. Dashboard tersebut berisi *widget* yang telah diimport

2.1.4 Tugas

Terdapat 2 buah led yang dikendalikan melalui dashboard thingsboard. Ketika tombol on pada dashboard thingsboard ditekan maka led pada rangkaian NodeMCU akan menyala, begitu sebaliknya.

2.2 Praktikum Minggu II

2.2.1 Percobaan 1

Pada percobaan ini, praktikan membuat *device* DHT11 pada Thingsboard. Setelah membuat *device* maka akan mendapatkan akses token.

2.2.2 Percobaan 2

Percobaan ini bertujuan untuk mengirim data dari sensor DHT11 ke thingsboard menggunakan MQTT. Data yang dikirim adalah nilai dari *Humidity* dan *Temperature*.

2.2.3 Percobaan 3

2.2.4 Tugas

2.3 Praktikum Minggu III

2.3.1 Percobaan 1

Pada percobaan ini, praktikan mengirim data dari sensor jarak menggunakan MQTT ke platform thingsboard. Data tersebut dikirim 1 detik sekali.

Source Code

```
1 # Libraries import os import time import sys
2 import paho.mqtt.client as mqtt
3 import json
4 import RPi.GPIO as GPIO
5 import time
6 import sys
7 import os
```


Kode di atas digunakan untuk menambahkan *library* yang akan digunakan.

```
2 GPIO.setmode(GPIO.BCM) # set GPIO Pins
3 GPIO_TRIGGER = 18
4 GPIO_ECHO = 24
5 # set GPIO direction (IN / OUT)
6 GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
7 GPIO.setup(GPIO_ECHO, GPIO.IN)
```

Kode di atas digunakan untuk melakukan konfigurasi pada pin yang akan digunakan.

```
3 THINGSBOARD_HOST = 'demo.thingsboard.io'
4 ACCESS_TOKEN = 'WJUNtDkejLyn9nKhgDov'
5
6 client = mqtt.Client()
7 # Set access token
8 client.username_pw_set(ACCESS_TOKEN)
9
10 # Connect to ThingsBoard using default MQTT port and 60 seconds
    keepalive interval
11 client.connect(THINGSBOARD_HOST, 1883, 60)
12 client.loop_start()
```

Kode di atas digunakan untuk men-*setup* komunikasi MQTT dan komunikasi terhadap platform thingsboard.

```
4 def distance():
5
6     # set Trigger to HIGH
7     GPIO.output(GPIO_TRIGGER, True)
8     # set Trigger after 0.01ms to LOW
9     time.sleep(0.00001)
10    GPIO.output(GPIO_TRIGGER, False)
11    StartTime = time.time()
12    StopTime = time.time() # save StartTime
13    while GPIO.input(GPIO_ECHO) == 0:
14        StartTime = time.time()
15    # save time of arrival
16    while GPIO.input(GPIO_ECHO) == 1:
17        StopTime = time.time()
```

```

15     # time difference between start and arrival
16     TimeElapsed = StopTime - StartTime
17     # multiply with the sonic speed (34300 cm/s) # and divide by 2,
    because there and back
18     distance = (TimeElapsed * 34300) / 2
19     return distance

```

Kode di atas digunakan untuk mendapatkan data dari sensor jarak.

2.3.2 Percobaan 2

Pada percobaan ini, praktikan membuat *device* HCSR04 pada Thingsboard. *Device* tersebut berguna untuk menerima data yang berasal dari HCSR04. Data yang terkirim dapat dilihat *tab LATEST TELEMETRY*. Data tersebut akan digunakan dalam membuat dashboard. Dashboard yang digunakan merupakan sebuah *chart widget*. *Chart widget* tersebut bertipe *time series*.

2.3.3 Tugas

Percobaan ini bertujuan untuk mengambil data dari sensor DHT11 dan mengirimkannya ke Thingsboard. Pengiriman data menggunakan protokol komunikasi MQTT. Source Code

```

1  humidity, temperature = dht.read_retry(dht.DHT22, 4)
2  humidity = round(humidity, 2)
3  temperature = round(temperature, 2)

```

Kode di atas digunakan untuk mendapatkan data *humidity* dan *temperature* sensor DHT11. Data tersebut kemudian dibulatkan.

```

2  client.publish('v1/devices/me/telemetry', json.dumps(
    sensor_data), 1)

```

Kode di atas digunakan untuk mengirim data ke Thingsboard. Data yang dikirim berbentuk JSON.

2.4 Praktikum IV

2.4.1 Percobaan 1

Pada percobaan ini, praktikan menginstall beberapa *library* antara lain :

1. Sced Studio
2. Thingsboard MQTT PubSubClient

Source Code

```

1 import logging
2 import time
3 from tb_device_mqtt import TBDeviceMqttClient, TBPublishInfo
4 from grove.grove_mini_pir_motion_sensor import
    GroveMiniPIRMotionSensor
5 from grove.grove_ultrasonic_ranger import GroveUltrasonicRanger
6 from Sced_Python_DHT.seed_dht import DHT
7 from grove.grove_moisture_sensor import GroveMoistureSensor
8 from grove.button import Button
9 from grove.grove_ryb_led_button import GroveLedButton
10 from grove.grove_light_sensor_v1_2 import GroveLightSensor
11 from grove.grove_servo import GroveServo

```

Kode di atas digunakan untuk menambahkan *library* yang akan digunakan.

```

2 logging.basicConfig(level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(module)s
    - %(lineno)d - %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S')
3
4
5 log = logging.getLogger(__name__)
6
7 thingsboard_server = 'THINGSBOARD_HOST'
8 access_token = 'ACCESS_TOKEN'

```

Kode di atas digunakan untuk melakukan konfigurasi format penulisan dan akses thingsboard.

```

3 # Grove - Servo connected to PWM port
4 servo = GroveServo(12)
5 servo_angle = 90
6
7 # Grove - mini PIR motion pir_sensor connected to port D5
8 pir_sensor = GroveMiniPIRMotionSensor(5)
9

```

```

8     # Grove - Ultrasonic Ranger connected to port D16
9     ultrasonic_sensor = GroveUltrasonicRanger(16)
10
11    # Grove - LED Button connected to port D18
12    button = GroveLedButton(18)
13
14    # Grove - Moisture Sensor connected to port A0
15    moisture_sensor = GroveMoistureSensor(0)
16
17    # Grove - Light Sensor connected to port A2
18    light_sensor = GroveLightSensor(2)
19    light_state = False
20
21    # Grove - Temperature&Humidity Sensor connected to port D22
22    dht_sensor = DHT('11', 22)

```

Kode di atas digunakan untuk mendapatkan data dari masing-masing sensor. Dalam percobaan ini hanya perangkat led dan servo yang digunakan.

```

41 def on_server_side_rpc_request(request_id, request_body):
2     log.info('received rpc: {}, {}'.format(request_id,
request_body))
3     if request_body['method'] == 'getLedState':
4         client.send_rpc_reply(request_id, light_state)
5     elif request_body['method'] == 'setLedState':
6         light_state = request_body['params']
7         button.led.light(light_state)
8     elif request_body['method'] == 'setServoAngle':
9         servo_angle = float(request_body['params'])
10        servo.setAngle(servo_angle)
11    elif request_body['method'] == 'getServoAngle':
12        client.send_rpc_reply(request_id, servo_angle)

```

Kode di atas digunakan untuk menerima permintaan dari pengguna dan mengirim permintaan tersebut ke *device*. Pengguna dapat mengirim perintah untuk menjalankan servo dengan sudut tertentu dan mengontrol led.

2.4.2 Percobaan 2

Pada percobaan ini, praktikan membuat *device* dan *dashboard* pada Thingsboard. *Dashboard* yang digunakan sudah disediakan oleh Screenshot Studio.

2.4.3 Tugas

Percobaan ini merupakan gabungan dari percobaan 1 dan 2. Setelah *dashboard* dibuat, pengguna dapat mengendalikan led dan servo melalui *dashboard* tersebut.

2.5 Praktikum V

2.5.1 Percobaan 1

Pada percobaan ini, pengiriman data sensor DHT11 menggunakan komunikasi HTTP. Data yang akan dikirim merupakan data *dummy* yang dibuat menggunakan fungsi random. Data yang berhasil terkirim dapat dilihat pada *attributes device*. Source Code

```
1 from datetime import datetime
2 import json
3 import random
4 import struct
5 import os
6 import signal
7 import requests
8 import sys
```

Kode di atas digunakan untuk menambahkan *library* yang akan digunakan.

```
2 server_domain = 'https://demo.thingsboard.io/api/v1/'
2 token = 'balbala'
3 url_api_data = '/attributes'
```

Kode di atas digunakan untuk melakukan konfigurasi token dan *endpoint*

```
3 def send_data_to_server():
2     global response
3     print('sending data...')
4
5     url = server_domain + token + url_api_data
```

```

6     headers = {'content-type': 'application/json'}
7     payload = {
8         "temperature" : random.uniform(10.5, 100.5), "humidity" :
9         random.uniform(10.5, 100.5),
10    }
11    response = requests.post(url, data=json.dumps(payload), headers=
12    headers)
13    print(response.status_code)

```

Kode di atas digunakan untuk mengirim data ke Thingsboard. Data tersebut berisi tentang nilai dari *humidity* dan *temperature*. Data dikirim dalam bentuk JSON dengan menggunakan protokol http. Jika data berhasil terkirim akan mendapatkan kode 200 atau 201.

2.5.2 Percobaan 2

Dashboard berisi *cards* yang berisi data *humidity* dan *temperature*.

2.5.3 Tugas

Pada percobaan ini hanya perlu mengubah data yang dikirim dengan data asli dari sensor DHT11. Untuk mendapatkan data dari sensor perlu ditambahkan potongan kode ini

```

1 humidity, temperature = dht.read_retry(dht.DHT11, 4)
2 humidity = round(humidity, 2)
3 temperature = round(temperature, 2)

```

2.6 Praktikum VI

Praktikum kali ini tidak lagi menggunakan platform Thingsboard. Praktikan membuat *server*nya sendiri. Ada 3 komponen penting dalam praktikum ini, antara lain :

1. Server Raspberry digunakan sebagai server dan broker MQTT. Server dibuat menggunakan flask. Flask dipilih karena flask merupakan *framework* pembuat web yang ringan.
2. Client Client di sini merujuk pada NodeMCU. NodeMCU akan menerima data dari server melalui MQTT. Data tersebut digunakan untuk mengendalikan led.
3. komunikasi Komunikasi yang digunakan untuk menghubungkan server dan client adalah MQTT.

2.6.1 Percobaan 1

Pada percobaan ini, praktikan membuat broker. Cukup dengan menginstall *package* mosquitto untuk membuat broker. Setelah mosquitto terinstall, *enable service* mosquitto untuk menjalankan broker. Broker akan berjalan pada port 1883.

2.6.2 Percobaan 2

Pada percobaan ini, praktikan membuat server dengan flask. Berikut *library* yang digunakan :

1. flask
2. paho-mqtt

Source Code

```
1 import paho.mqtt.client as mqtt
2 from flask import Flask, render_template, request
```

Kode di atas digunakan untuk menambahkan *library* yang akan digunakan.

```
2 app = Flask(__name__)
2 mqttc=mqtt.Client()
3 mqttc.connect("test.mosquitto.org",1883,60)
4 mqttc.loop_start()
```

Kode di atas digunakan untuk inisiasi aplikasi flask dan MQTT.

```
3 @app.route("/")
2 def main():
3     # Pass the template data into the template main.html and return it
    to the user
4     return render_template('main.html', **templateData)
```

Ketika pengguna melakukan *request* ke *website* maka server akan mengembalikan "main.html".

```

4 def action(board, changePin, action):
5     # Convert the pin from the URL into an integer:
6     changePin = int(changePin)
7     # Get the device name for the pin being changed:
8     devicePin = pins[changePin]['name']
9     # If the action part of the URL is "on," execute the code indented
10    below:
11    if action == "1" and board == 'esp8266':
12        mqttc.publish(pins[changePin]['topic'], "1")
13        pins[changePin]['state'] = 'True'
14
15    if action == "0" and board == 'esp8266':
16        mqttc.publish(pins[changePin]['topic'], "0")
17        pins[changePin]['state'] = 'False'
18
19    # Along with the pin dictionary, put the message into the template
20    data dictionary:
21    templateData = {
22        'pins' : pins
23    }
24
25    return render_template('main.html', **templateData)

```

Ketika ada *trigger event* dari pengguna, server akan mengirim data 0 atau 1 pada topik yang ditentukan. Tampilan pada *website* juga akan berubah.

2.6.3 Percobaan 3

```

1 #include <ESP8266WiFi.h>
2 #include <PubSubClient.h>

```

Kode di atas digunakan untuk menambahkan *library* yang akan digunakan.

```

2 const char* ssid = "YOUR_SSID";
3 const char* password = "YOUR_PASSWORD";
4 const char* mqtt_server = "YOUR_RPi_IP_Address";

```

Kode di atas digunakan untuk konfigurasi wifi dan MQTT yang digunakan.


```

3  WiFiClient espClient;
2  PubSubClient client(espClient);

```

Kode di atas digunakan untuk inisiasi *class* wifi dan MQTT.

```

4  void setup_wifi() {
2    delay(10);
3    // We start by connecting to a WiFi network
4    Serial.println();
5    Serial.print("Connecting to ");
6    Serial.println(ssid);
7    WiFi.begin(ssid, password);
8    while (WiFi.status() != WL_CONNECTED) {
9        delay(500);
10       Serial.print(".");
11    }
12    Serial.println("");
13    Serial.print("WiFi connected - ESP IP address: ");
14    Serial.println(WiFi.localIP());
15 }

```

Kode di atas digunakan untuk terhubung dengan wifi.

```

5  void callback(String topic, byte* message, unsigned int length) {
2    Serial.print("Message arrived on topic: ");
3    Serial.print(topic);
4    Serial.print(". Message: ");
5    String messageTemp;
6
7    for (int i = 0; i < length; i++) {
8        Serial.print((char)message[i]);
9        messageTemp += (char)message[i];
10    }
11    Serial.println();
12
13    // Feel free to add more if statements to control more GPIOs with
14    MQTT
15
16    // If a message is received on the topic home/office/espl/gpio2,
17    you check if the message is either 1 or 0. Turns the ESP GPIO
18    according to the message
19    if(topic=="esp8266/4"){

```

```

17     Serial.print("Changing GPIO 4 to ");
18     if(messageTemp == "1"){
19         digitalWrite(ledGPIO4, HIGH);
20         Serial.print("On");
21     }
22     else if(messageTemp == "0"){
23         digitalWrite(ledGPIO4, LOW);
24         Serial.print("Off");
25     }
26 }
27 if(topic=="esp8266/5"){
28     Serial.print("Changing GPIO 5 to ");
29     if(messageTemp == "1"){
30         digitalWrite(ledGPIO5, HIGH);
31         Serial.print("On");
32     }
33     else if(messageTemp == "0"){
34         digitalWrite(ledGPIO5, LOW);
35         Serial.print("Off");
36     }
37 }
38 Serial.println();
39 }

```

Ketika ada pesan dari MQTT maka pesan tersebut akan di *parsing* menjadi data. Data tersebut bertipe *boolean* Data tersebut digunakan untuk mengganti *state* dari pin Node-MCU.

```

6. 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:
12     if (client.connect("ESPOffice")) {
13     Then, for the other ESP:
14     if (client.connect("ESPGarage")) {
15     That should solve your MQTT multiple connections problem
16
17     THE SECTION IN loop() function should match your device name
18     */
19     if (client.connect("ESP8266Client")) {
20     Serial.println("connected");
21     // Subscribe or resubscribe to a topic
22     // You can subscribe to more topics (to control more LEDs in
    this example)
23     client.subscribe("esp8266/4");
24     client.subscribe("esp8266/5");
25     } else {
26     Serial.print("failed, rc=");
27     Serial.print(client.state());
28     Serial.println(" try again in 5 seconds");
29     // Wait 5 seconds before retrying
30     delay(5000);
31     }
32     }
33 }

```

Ketika koneksi wifi ataupun mqtt terputus, maka akan dihubungkan kembali.

2.6.4 Tugas

1. Protokol komunikasi yang digunakan

- (a) MQTT
- (b) HTTP
- (c) CoAP