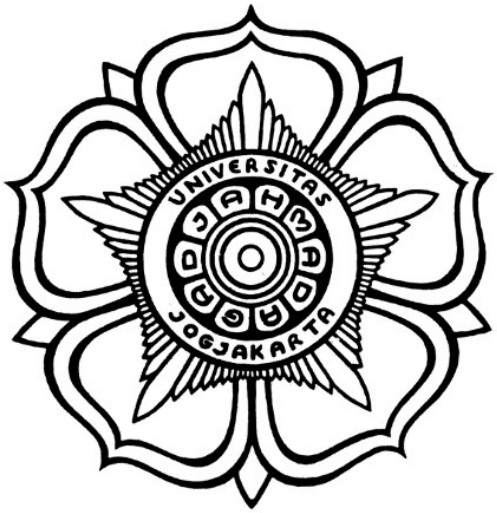
LAPORAN PRAKTIKUM

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

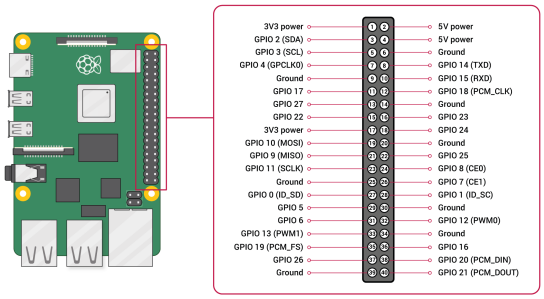
Pengenalan

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.

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

1.5 HTTP

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>
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
```

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

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

Kode di atas digunakan untuk mendefinisikan sebuah konstanta dan variable.

3.
```
void setup()
{
    Serial.begin(115200);
    // Set output mode for all GPIO pins

    delay(10);
    InitWiFi();

    pinMode(GPIO0, OUTPUT);
    pinMode(GPIO2, OUTPUT);
    client.setServer(thingsboardServer, 1883);
    client.setCallback(on_message);
}
```

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

4.
```
void loop()
{
    if (!client.connected())
    {
        reconnect();
    }

    client.loop();
}
```

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

5.
```
void on_message(const char *topic, byte *payload, unsigned int
    length)
{

    Serial.println("On message");

    char json[length + 1];
    strncpy(json, (char *)payload, length);
    json[length] = '\0';

    Serial.print("Topic: ");
```

```
11      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      else if (methodName.equals("setGpioStatus"))
36      {
37          // Update GPIO status and reply
38          set_gpio_status(data["params"]["pin"], data["params
    "]["enabled"]);
39          String responseTopic = String(topic);
40          responseTopic.replace("request", "response");
41          client.publish(responseTopic.c_str(), get_gpio_status
    ().c_str());
42          client.publish("v1/devices/me/attributes",
    get_gpio_status().c_str());
43      }
44  }
```

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
```

```
import RPi.GPIO as GPIO
import time
import sys
import os
```

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

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

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

3.
```
THINGSBOARD_HOST = 'demo.thingsboard.io'
ACCESS_TOKEN = 'WJUNtDkejLyn9nKhgDov'

client = mqtt.Client()
# Set access token
client.username_pw_set(ACCESS_TOKEN)

# Connect to ThingsBoard using default MQTT port and 60
    seconds keepalive interval
client.connect(THINGSBOARD_HOST, 1883, 60)
client.loop_start()
```

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

4.
```
def distance():

    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)
    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)
```

```
    StartTime = time.time()
    StopTime = time.time()  # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()
    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()
    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s) # and divide
    by 2, because there and back
    distance = (TimeElapsed * 34300) / 2
    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)
            humidity = round(humidity, 2)
            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. Seeed Studio
2. Thingsboard MQTT PubSubClient

Source Code

1.
```
import logging
import time
from tb_device_mqtt import TBDeviceMqttClient, TBPublishInfo
from grove.grove_mini_pir_motion_sensor import
    GroveMiniPIRMotionSensor
from grove.grove_ultrasonic_ranger import
    GroveUltrasonicRanger
from Seeed_Python_DHT.seeed_dht import DHT
from grove.grove_moisture_sensor import GroveMoistureSensor
from grove.button import Button
from grove.grove_ryb_led_button import GroveLedButton
from grove.grove_light_sensor_v1_2 import GroveLightSensor
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')

log = logging.getLogger(__name__)

thingsboard_server = 'THINGSBOARD_HOST'
access_token = 'ACCESS_TOKEN'
```

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

3.
```
    # Grove - Servo connected to PWM port
    servo = GroveServo(12)
    servo_angle = 90

    # Grove - mini PIR motion pir_sensor connected to port D5
    pir_sensor = GroveMiniPIRMotionSensor(5)

    # Grove - Ultrasonic Ranger connected to port D16
    ultrasonic_sensor = GroveUltrasonicRanger(16)

    # Grove - LED Button connected to port D18
    button = GroveLedButton(18)

    # Grove - Moisture Sensor connected to port A0
    moisture_sensor = GroveMoistureSensor(0)

    # Grove - Light Sensor connected to port A2
    light_sensor = GroveLightSensor(2)
    light_state = False

    # Grove - Temperature&Humidity Sensor connected to port
    D22
    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.

4.
```
    def on_server_side_rpc_request(request_id, request_body):
        log.info('received rpc: {}, {}'.format(request_id,
    request_body))
        if request_body['method'] == 'getLedState':
            client.send_rpc_reply(request_id, light_state)
        elif request_body['method'] == 'setLedState':
            light_state = request_body['params']
            button.led.light(light_state)
        elif request_body['method'] == 'setServoAngle':
            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 pada 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 Seeed 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

2.5.2 Percobaan 2

2.5.3 Tugas

2.6 Praktikum V

2.6.1 Percobaan 1

2.6.2 Percobaan 2

2.6.3 Percobaan 3

2.6.4 Tugas