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

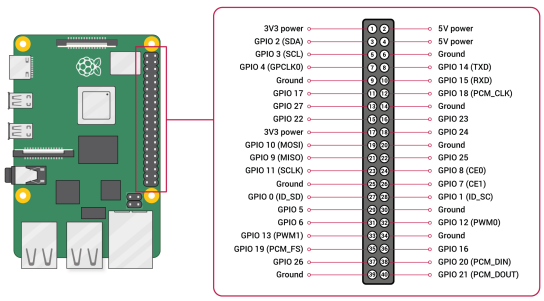
Pengenalan

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: ");
    Serial.println(topic);
```

```
    Serial.print("Message: ");
    Serial.println(json);

    // Decode JSON request
    StaticJsonBuffer<200> jsonBuffer;
    JsonObject &data = jsonBuffer.parseObject((char *)json);

    if (!data.success())
    {
        Serial.println("parseObject() failed");
        return;
    }

    // Check request method
    String methodName = String((const char *)data["method"]);

    if (methodName.equals("getGpioStatus"))
    {
        // Reply with GPIO status
        String responseTopic = String(topic);
        responseTopic.replace("request", "response");
        client.publish(responseTopic.c_str(), get_gpio_status().c_str
    ());
    }
    else if (methodName.equals("setGpioStatus"))
    {
        // Update GPIO status and reply
        set_gpio_status(data["params"]["pin"], data["params"]["
    enabled"]);
        String responseTopic = String(topic);
        responseTopic.replace("request", "response");
        client.publish(responseTopic.c_str(), get_gpio_status().c_str
    ());
        client.publish("v1/devices/me/attributes", get_gpio_status().
    c_str());
    }
}
```

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
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'])
            servo.setAngle(servo_angle)
        elif request_body['method'] == 'getServoAngle':
            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

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
import json
import random
import struct
import os
import signal
import requests
import sys
```

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

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

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

3.
```
def send_data_to_server():
    global response
    print('sending data...')

    url = server_domain + token + url_api_data
```

```
    headers = {'content-type': 'application/json'}
    payload = {
        "temperature" : random.uniform(10.5, 100.5), "humidity" :
    random.uniform(10.5, 100.5),
    }

    response = requests.post(url, data=json.dumps(payload), headers=
    headers)
    print(response.status_code)
```

Kode di atas digunakan untuk mengirim data ke Thingsboard. Data tersebut berisi tentang nilai dari *humdity* 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

Dasboard 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

```
    humidity,temperature = dht.read_retry(dht.DHT11, 4)
    humidity = round(humidity, 2)
    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
from flask import Flask, render_template, request
```

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

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

Kode di atas digunakan untuk inisiasi applikasi flask dan MQTT.

3.
```
@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)
```

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

4.
```
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 == 'esp8266':
        mqttc.publish(pins[changePin]['topic'],"1")
        pins[changePin]['state'] = 'True'

    if action == "0" and board == 'esp8266':
        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)
```

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>
#include <PubSubClient.h>
```

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

2.
```
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
const char* mqtt_server = "YOUR_RPi_IP_Address";
```

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

3.
```
WiFiClient espClient;
PubSubClient client(espClient);
```

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

4.
```
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected - ESP IP address: ");
  Serial.println(WiFi.localIP());
}
```

Kode di atas digunakan untuk terhubung dengan wifi.

5.
```
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=="esp8266/4"){
```

```
      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=="esp8266/5"){
      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();
}
```

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 NodeMCU.

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:
       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("ESP8266Client")) {
      Serial.println("connected");
      // Subscribe or resubscribe to a topic
      // You can subscribe to more topics (to control more LEDs in
   this example)
      client.subscribe("esp8266/4");
      client.subscribe("esp8266/5");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

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