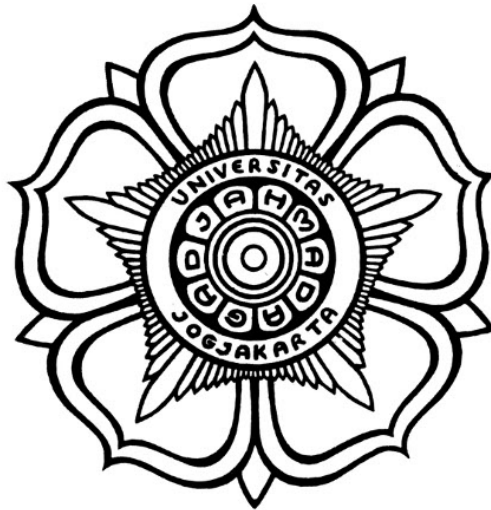


LAPORAN PRAKTIKUM

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

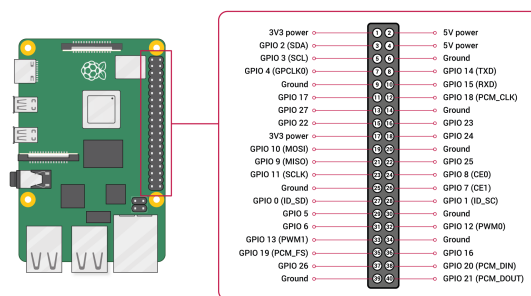
Pengenalan

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 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>
2. #include <PubSubClient.h>
3. #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()
2 {
3     Serial.begin(115200);
4     // Set output mode for all GPIO pins
5
6     delay(10);
7     InitWiFi();
8
9     pinMode(GPIO0, OUTPUT);
10    pinMode(GPIO2, OUTPUT);
11    client.setServer(thingsboardServer, 1883);
12    client.setCallback(on_message);
13 }

```

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

```

4. void loop()
2 {
3     if (!client.connected())
4     {
5         reconnect();
6     }
7
8     client.loop();
9 }

```

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

```

5. void on_message(const char *topic, byte *payload, unsigned int
    length)
2 {
3
4     Serial.println("On message");
5
6     char json[length + 1];
7     strncpy(json, (char *)payload, length);
8     json[length] = '\0';
9
10    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
34     ().c_str());
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
44     ().c_str());
45     client.publish("v1/devices/me/attributes",
46     get_gpio_status().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
2 GPIO_TRIGGER = 18
3 GPIO_ECHO = 24
4 # set GPIO direction (IN / OUT)
5 GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
6 GPIO.setup(GPIO_ECHO, GPIO.IN)

```

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

```

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

```

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

```

4. def distance():
2
3     # set Trigger to HIGH
4     GPIO.output(GPIO_TRIGGER, True)
5     # set Trigger after 0.01ms to LOW
6     time.sleep(0.00001)
7     GPIO.output(GPIO_TRIGGER, False)

```



```

8     StartTime = time.time()
9     StopTime = time.time()    # save StartTime
10    while GPIO.input(GPIO_ECHO) == 0:
11        StartTime = time.time()
12    # save time of arrival
13    while GPIO.input(GPIO_ECHO) == 1:
14        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. Sseed 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 Sseed_Python_DHT.seeed_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
    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.

```

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)

```

```

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.

1. Praktikum Minggu I

Pada praktikum ini, praktikan diminta untuk membuat dashboard yang berfungsi untuk mengontrol pin pada nodemcu.

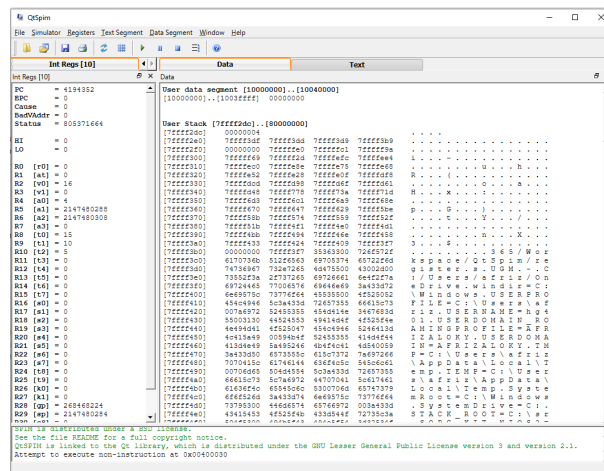
add \$t0,\$t1,\$t2 \$t0 = \$t1 + \$t2

Source Code :

```

1 .text
2 .globl main
3 main:
4
5     li $t1, 10
6     li $t2, 5
7     add $t0, $t1, $t2 # $t0 = $t1 + $t2

```



Gambar 2.1: Simulasi Pengalaman Register

Dapat dilihat pada figure 1, nilai t1 adalah 10, nilai t2 adalah 5 dan nilai t0 adalah hasil penjumlahan t1 dan t2.

2. Pengalaman Base

Pengalaman base merupakan pengalaman yang menunjuk satu pointer sebagai base. Dalam pengalaman ini sebuah register ditunjuk sebagai basis alamat memori Contoh dalam penggunaan pengalaman base :

lw \$s1, 4(\$s0)

perintah diatas digunakan untuk mengambil nilai word pada memori s0 dengan offset 4.

Source Code :

```

1
2 .data
3     str: .asciiz "hello world 1234567890"
4
5 .text
6 .globl main
7 main:
8     la      $s1, str      # s1 points to the string
9
10    lw      $s2, 0($s1)
11    lw      $s3, 4($s1)

```

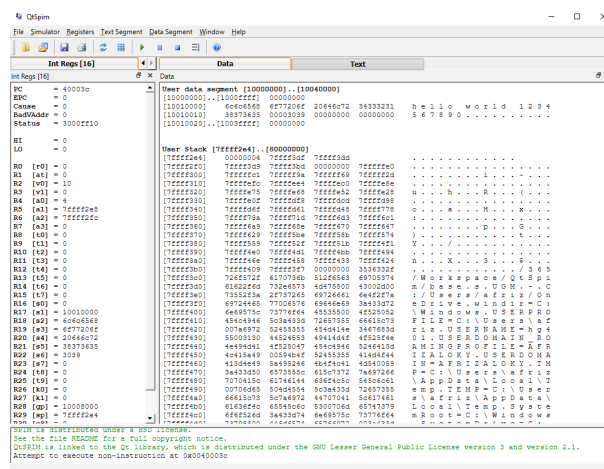
memory	0x100100	0x100101	0x100102	0x100103
value(hex)	6c	6c	65	68
value(ascii)	l	l	e	h

Tabel 2.1: Memory address

```

12      lw      $s4, 8($s1)
13      lw      $s5, 16($s1)
14      lw      $s6, 20($s1)

```



Gambar 2.2: Simulasi Pengalaman Base

Pada source code, nilai register s1 adalah sebuah string "hello world" Dapat dilihat pada figure 2, register s1 memiliki memory address 0x1001000. Register s2 merupakan hasil pointing dari register s1 dengan offset 0. Sehingga register s2 akan memiliki nilai berdasarkan nilai dari memory address 0x1001000-0x1001003. Value dari memory address tersebut ditulis dalam format little endian.

3. Pengalaman Immiediete

Pengalaman immiediete merupakan pengalaman yang membutuhkan satu operand saja dan sebuah konstanta (immiediete value). Pengalaman ini akan mengeksekusi dirinya sendiri Contoh dalam penggunaan pengalaman immiediete :

```
addi    $t0, $t0, 1
```

Dalam bahasa pemrograman C, kode tersebut dapat ditulis dengan $t0 = t0 + 1$;


```

10         la $t2,str    #t2 points to the string
11         li $t1,0      #t1 holds the count
12
13 nextCh: lb $t0,($t2)#get a byte from the string
14         beqz $t0,strEnd #zero means end of string
15         add $t1, $t1,1  #increment count
16         add $t2, 1     #move pointer one character
17         j nextCh      #go round the loop again
18
19 strEnd:
20         la $a0,ans     #System call
21         li $v0,4       #to print out
22         syscall        #the string message
23
24         move $a0,$t1   #copy the count to a0
25         li $v0,1       #System call 1
26         syscall        #to print the length
27         la $a0,endl    #syscall to print out
28         li $v0,4       #a newline
29         syscall
30
31
32         li $v0,10
33         syscall        #Bye!

```

Pada source code tersebut, register t1 merupakan Program Counter.

5. Pengalamatan Pseudodirect

Pengalamatan pseudodirect biasanya tertanam langsung pada instruksinya. Pseudodirect memiliki instruksi format 6bit untuk opcode dan 26bit untuk target. Pseudodirect menggunakan tipe jump instruksi.

Source Code :

```

1
2 .text
3 .globl main
4 main:
5
6     li $t0, 3
7     li $t1, 5
8     li $t2, 0 #counter

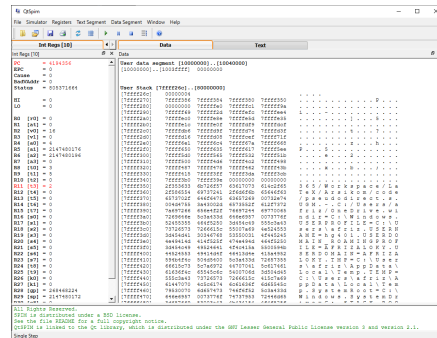
```



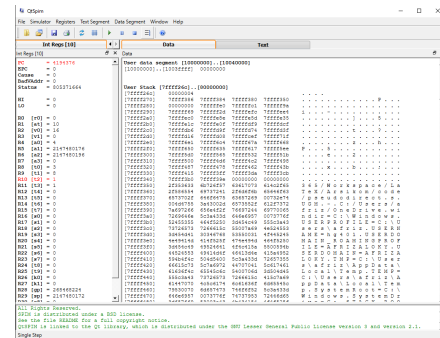
```

9      li $t3, 2
10
11
12 loop:beq $t2,10,End # Looping 10x
13     add $t1, $t1,$t0
14     addi $t3, $t3, -1
15
16     addi $t2,$t2,1 #incnrement
17     j loop
18 End:
19     li $v0,10
20     syscall # Bye!

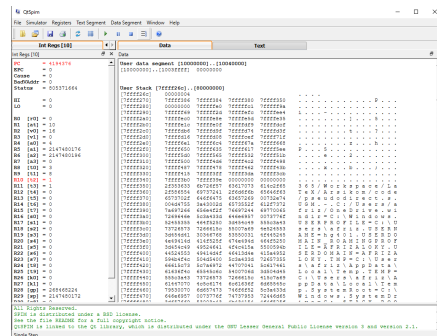
```



(a) step 1



(b) step 2



(c) step 3

Gambar 2.4: Simulasi Pengalaman pseudodirect

Pada Gambar 4.a merupakan proses inisiasi nilai register t0,t1,t2 dan t3. Pada Gambar 4.b merupakan proses iterasi pertama. Pada Gambar 4.c merupakan proses setelah iterasi selesai

iterasi ke-	0	1	2	3	4	5	6	7	8	9	10
t_0	3	3	3	3	3	3	3	3	3	3	3
t_1	5	8	11	14	17	20	23	26	29	32	35
t_2	0	1	2	3	4	5	6	7	8	9	10
t_3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8

Tabel 2.2: Proses Iterasi