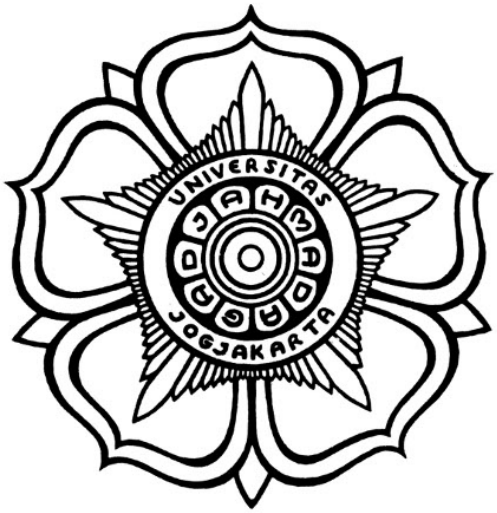
LAPORAN PRAKTIKUM

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

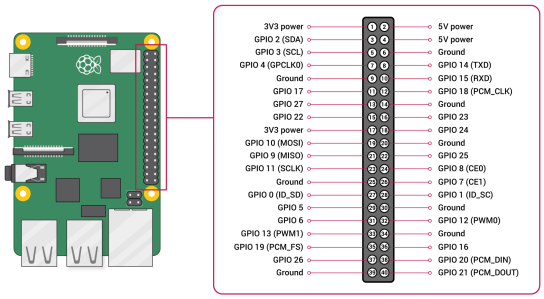
Pengenalan

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

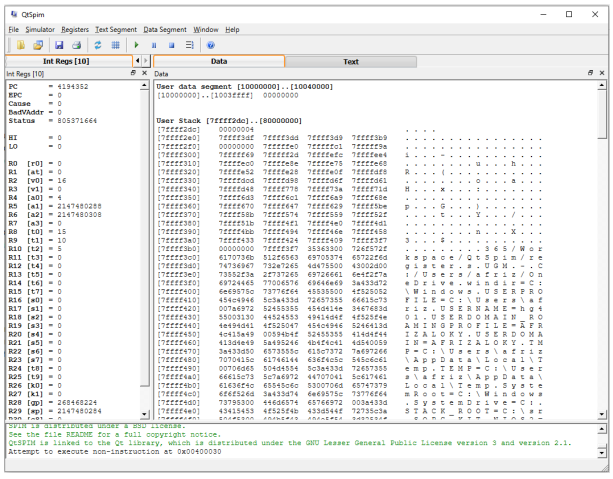
1. Praktikum Minggu I

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

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

Source Code :

```
.text
.globl main
main:

    li $t1, 10
    li $t2, 5
    add $t0, $t1, $t2 # $t0 = $t1 + $t2
```



Gambar 2.1: Simulasi Pengalamatan Register

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

2. Pengalamatan Base

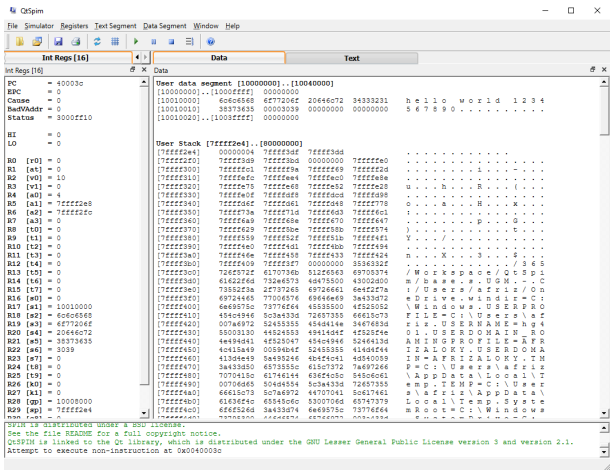
   Pengalamatan base merupakan pengalamatan yang menunjuk satu pointer sebagai base. Dalam pengalamatan ini sebuah register ditunjuk sebagai basis alamat memori Contoh dalam penggunaan pengalamatan base :

$$lw \quad \$s1, 4(\$s0)$$

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



Gambar 2.2: Simulasi Pengalamatan Base

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

Tabel 2.1: Memory address

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. Pengalamatan Immediete

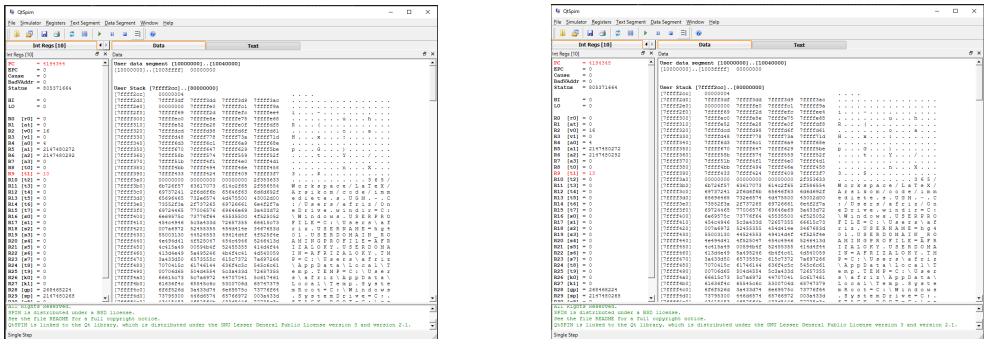
   Pengalamatan immediete merupakan pengalamatan yang membutuhkan satu operand saja dan sebuah konstanta (immediete value). Pengalamatan ini akan mengeksekusi dirinya sendiri Contoh dalam penggunaan pengalamatan immediete :

   $$addi \quad \$t0, \$t0, 1$$

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

   Source Code :

```
1 .text
2 .globl main
3 main:
4
5     li      $t1, 10
6     addi    $t1, $t1, 3 #$t1 = $t1 + 3
```



(a) step 1 **(b)** step 2

Gambar 2.3: Simulasi Pengalamatan Immediete

Pada gambar Figure 3.a nilai dari register t1 adalah 10. Dengan menggunakan single step pada QtSpim, setelah opcode dieksekusi maka nilai register t1 menjadi 13. Hal tersebut dapat dilihat pada gambar Figure 3.b

4. Pengalamatan PC-relative

   Data dari pengalamatan ini spesifik pada offset tertentu. Pengalamatan ini sering digunakan pada fungsi kondisional. Offset value dapat berupa *immediete value* atau sebuah label value.

   Source Code :

```
.data
  str:  .asciiz "hello world"
  ans:  .asciiz "Length is "
  endl: .asciiz "\n"

        .text
        .globl main
main:                 #execution starts here

        la $t2,str  #t2 points to the string
        li $t1,0    #t1 holds the count

nextCh: lb $t0,($t2)#get a byte from the string
  beqz $t0,strEnd #zero means end of string
  add $t1, $t1,1  #increment count
  add $t2, 1  #move pointer one character
  j nextCh  #go round the loop again

```

```
strEnd:
  la $a0,ans  #System call
  li $v0,4  #to print out
  syscall   #the string message

  move $a0,$t1 #copy the count to a0
  li $v0,1   #System call 1
  syscall    #to print the length
  la $a0,endl  #syscall to print out
  li $v0,4   #a newline
  syscall


  li $v0,10
  syscall     #Bye!
```

Pada source code tersebut, register t1 merupakan Program Counter.

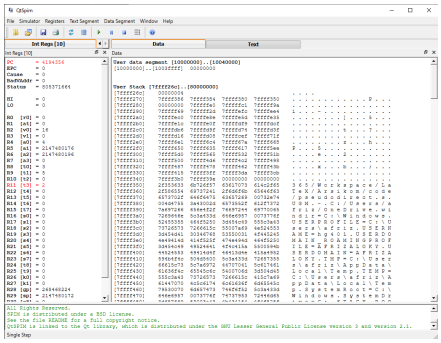
5. Pengalamatan Pseudodirect

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

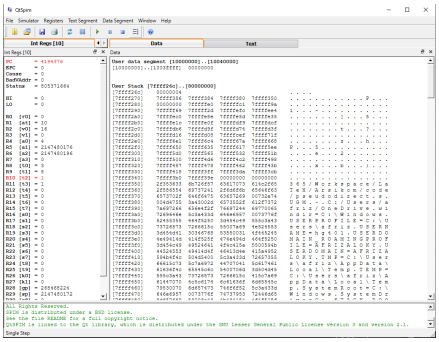
   Source Code :

```

.text
.globl main
main:

    li $t0, 3
    li $t1,5
    li $t2, 0 #counter
    li $t3, 2


loop:beq $t2,10,End # Looping 10x
  add $t1, $t1,$t0
    addi $t3, $t3, -1

  addi $t2,$t2,1  #increnment
  j loop
```

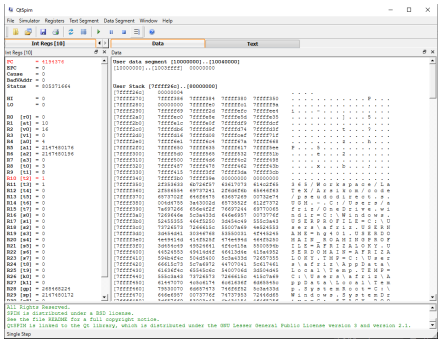
```
18 End:
19   li $v0,10
20   syscall # Bye!
```



(a) step 1



(b) step 2



(c) step 3

Gambar 2.4: Simulasi Pengalamatan 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
$t0$	3	3	3	3	3	3	3	3	3	3	3
$t1$	5	8	11	14	17	20	23	26	29	32	35
$t2$	0	1	2	3	4	5	6	7	8	9	10
$t3$	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8

Tabel 2.2: Proses Iterasi