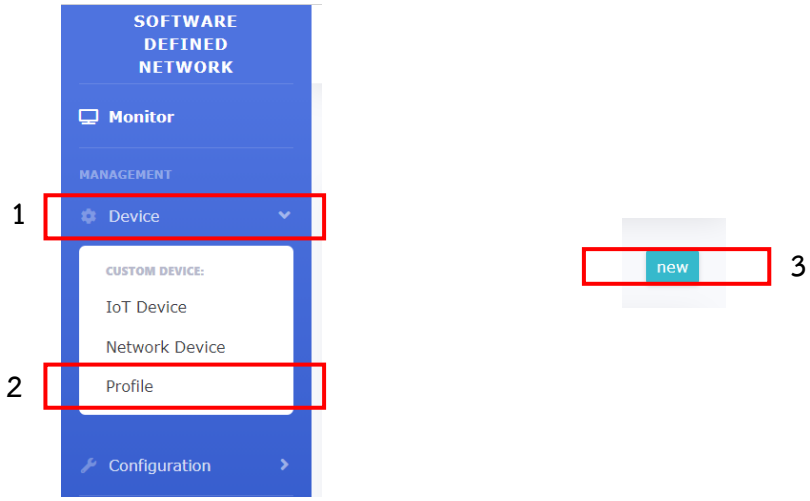
ภาคผนวก ก

คู่มือการใช้งานระบบเครือข่ายที่กำหนดโดยซอฟต์แวร์

การจัดการเครือข่ายที่กำหนดโดยซอฟต์แวร์ โดยใช้เทคโนโลยีอินเทอร์เน็ตของสรรพสิ่ง

ขั้นตอนการใช้งาน

1. ขั้นตอนแรกสร้างโปรไฟล์สำหรับการกำหนดค่าให้กับอุปกรณ์เครือข่าย



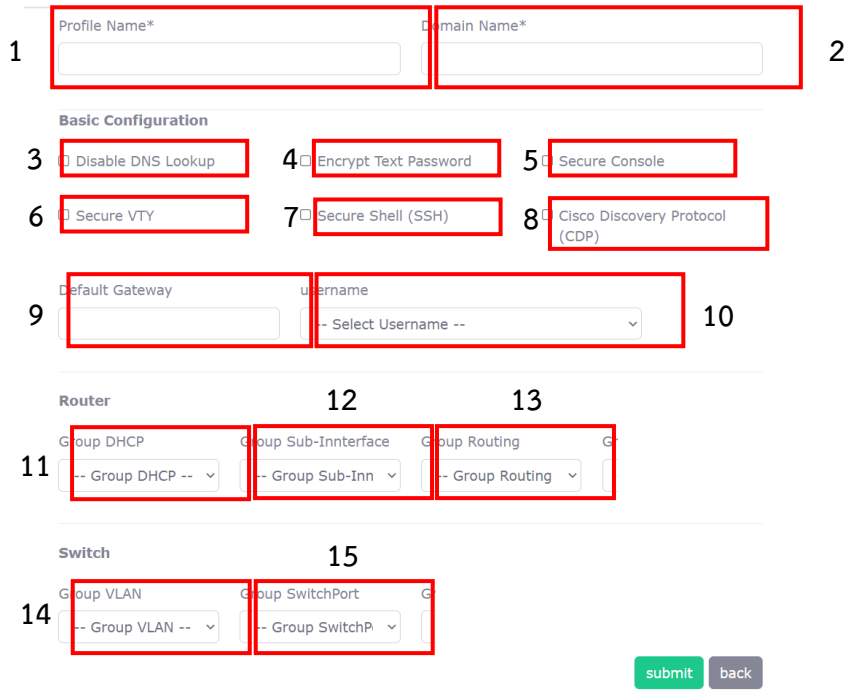
ภาพที่ ก.1 สร้างโปรไฟล์สำหรับการกำหนดค่าอุปกรณ์เครือข่าย

จากภาพที่ ก.1 สร้างโปรไฟล์สำหรับการกำหนดค่าอุปกรณ์เครือข่าย

หมายเลขที่ 1 Device เลือกเมนู Device สำหรับเข้าเมนูย่อยในการจัดการ

หมายเลขที่ 2 Profice เลือกเมนู Profile สำหรับการกำหนดค่ากำหนดค่าให้กับอุปกรณ์เครือข่าย

หมายเลขที่ 3 New กดปุ่ม New สำหรับการสร้างโปรไฟล์



ภาพที่ ก.2 กำหนดค่ากำหนดค่าอุปกรณ์เครือข่าย

จากภาพที่ ก.2 กำหนดค่าอุปกรณ์เครือข่าย

หมายเลขที่ 1 Profile Name	คือการตั้งชื่อโปรไฟล์
หมายเลขที่ 2 Domain Name	คือการกำหนด Domain Name ให้กับอุปกรณ์เครือข่าย
หมายเลขที่ 3 Disable DNS Lookup	คือการปิดคำร้องขอ DNS
หมายเลขที่ 4 Encrypt Text Password	คือการเข้ารหัสให้กับพาสเวิร์ด
หมายเลขที่ 5 Sucure Console	คือการสร้าง Username Password ให้กับการเชื่อมต่อผ่านคอนโซลโดยค่าเริ่มต้นคือ Username : administrator Password : P@ssw0rd@iot
หมายเลขที่ 6 Sucure VTY	คือการสร้าง Username Password ให้กับการเชื่อมต่อผ่านรีโมทโดยค่าเริ่มต้นคือ Username : administrator Password : P@ssw0rd@iot
หมายเลขที่ 7 Sucure SSH	คือการเปิดการใช้งานการเชื่อมต่อโดยรีโมทผ่านโปรโตคอล SSH
หมายเลขที่ 8 CDP	คือการเปิดการใช้งานดูอุปกรณ์ที่มีการเชื่อมต่อด้วย

หมายเลขที่ 9 Default Gateway คือการกำหนด IP Default Gateway ให้กับอุปกรณ์เครือข่าย

หมายเลขที่ 10 Username Password เลือก Username Password สำหรับการ รีโมท ของโปรโตคอล SSH

หมายเลขที่ 11 DHCP เลือกกลุ่มการทำ DHCP สำหรับ Router

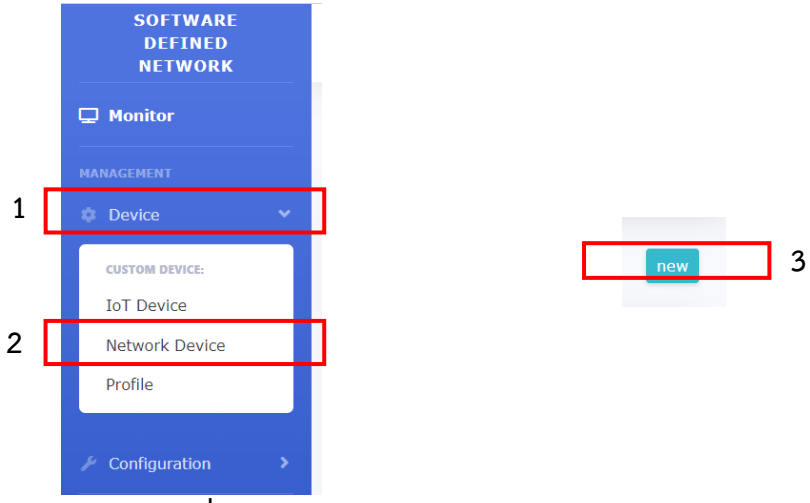
หมายเลขที่ 12 Subinterface เลือกกลุ่มการทำ Subinterface สำหรับ Router

หมายเลขที่ 13 Routing เลือกกลุ่มการทำ Routing สำหรับ Router

หมายเลขที่ 14 VLAN เลือกกลุ่มการทำ VLAN สำหรับ Switch

หมายเลขที่ 15 Etherchannel เลือกกลุ่มการทำ Etherchannel สำหรับ Switch

2. ขั้นตอนการลงทะเบียนอุปกรณ์เครือข่าย



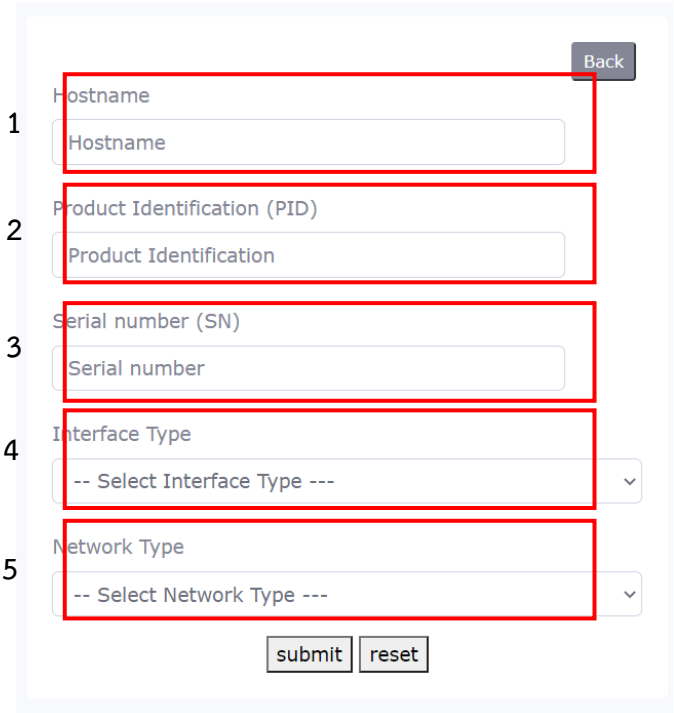
ภาพที่ ก.3 ลงทะเบียนอุปกรณ์เครือข่าย

จากภาพที่ ก.3 ลงทะเบียนอุปกรณ์เครือข่าย

หมายเลขที่ 1 Device เลือกเมนู Device สำหรับเข้าเมนูย่อยในการจัดการ

หมายเลขที่ 2 Network Device เลือกเมนู Network Device สำหรับการลงทะเบียนอุปกรณ์เครือข่าย

หมายเลขที่ 3 New กดปุ่ม New สำหรับการลงทะเบียนอุปกรณ์เครือข่าย



ภาพที่ ก.4 ลงทะเบียนอุปกรณ์เครือข่าย

จากภาพที่ ก.4 ลงทะเบียนอุปกรณ์เครือข่าย

หมายเลขที่ 1 Hostname คือการกำหนดชื่อให้อุปกรณ์เครือข่าย

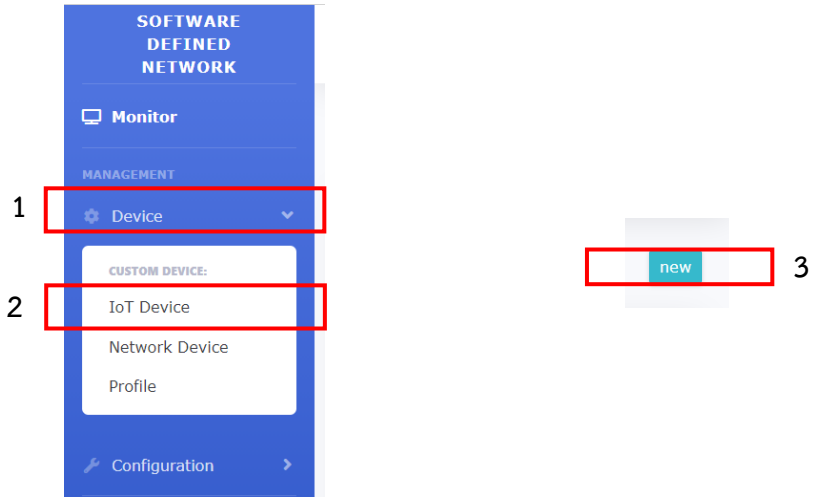
หมายเลขที่ 2 PID คือการกำหนด PID ของอุปกรณ์เครือข่าย

หมายเลขที่ 3 SN คือการกำหนด SN ของอุปกรณ์เครือข่าย

หมายเลขที่ 4 Interface Type เลือกประเภทของขา Interface

หมายเลขที่ 5 Network Type เลือกประเภทของอุปกรณ์เครือข่าย Router,Switch

1. ขั้นตอนการลงทะเบียนอุปกรณ์ IoT



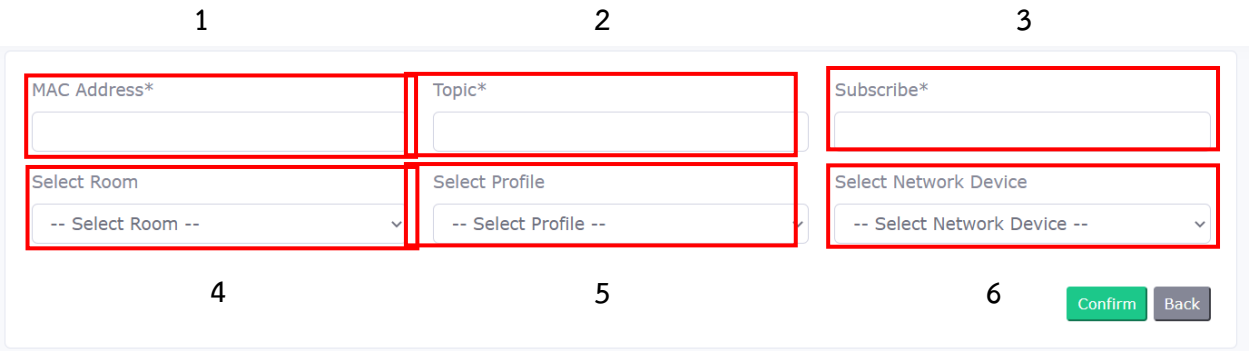
ภาพที่ ก.5 ลงทะเบียนอุปกรณ์ IoT

จากภาพที่ ก.5 ลงทะเบียนอุปกรณ์ IoT

หมายเลขที่ 1 Device เลือกเมนู Device สำหรับเข้าเมนูย่อยในการจัดการ

หมายเลขที่ 2 IoT Device เลือกเมนู IoT สำหรับการลงทะเบียนอุปกรณ์ IoT

หมายเลขที่ 3 New กดปุ่ม New สำหรับการลงทะเบียนอุปกรณ์ IoT



ภาพที่ ก.6 ลงทะเบียนอุปกรณ์ IoT

จากภาพที่ ก.6 ลงทะเบียนอุปกรณ์ IoT

หมายเลขที่ 1 MAC Address คือการใส่ MAC Address ของอุปกรณ์ IoT

หมายเลขที่ 2 Topic คือการใส่ Topic ของอุปกรณ์ IoT สำหรับการสื่อสารส่งข้อมูล

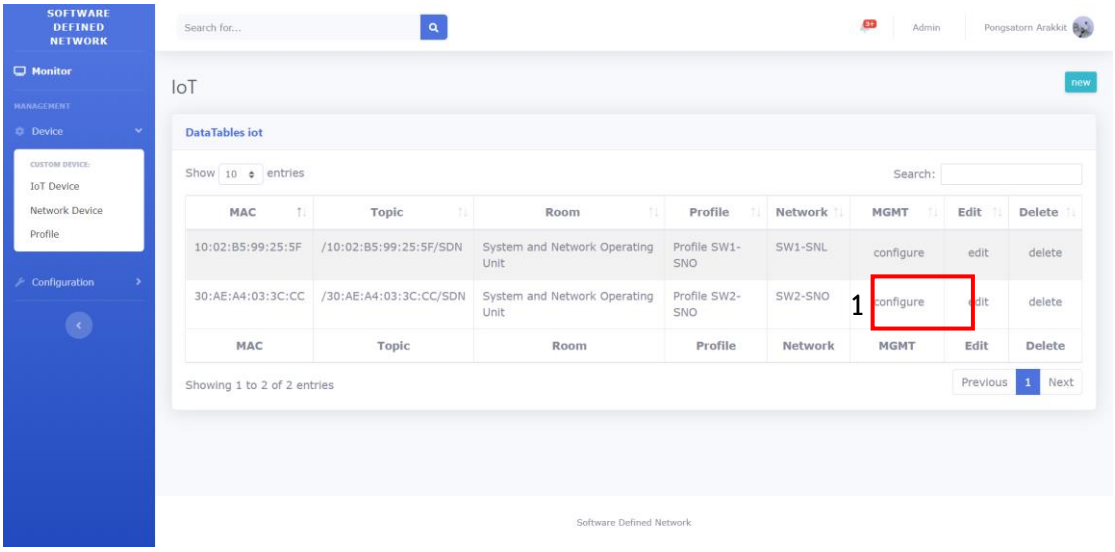
หมายเลขที่ 3 Subscribe คือการใส่ Subscribe ของอุปกรณ์ IoT สำหรับการสื่อสารรับข้อมูล

หมายเลขที่ 4 Room เลือกห้องที่อุปกรณ์เครือข่ายอยู่

หมายเลขที่ 5 Profile เลือกโปรไฟล์สำหรับอุปกรณ์เครือข่าย

หมายเลขที่ 6 Network Device เลือกอุปกรณ์เครือข่าย

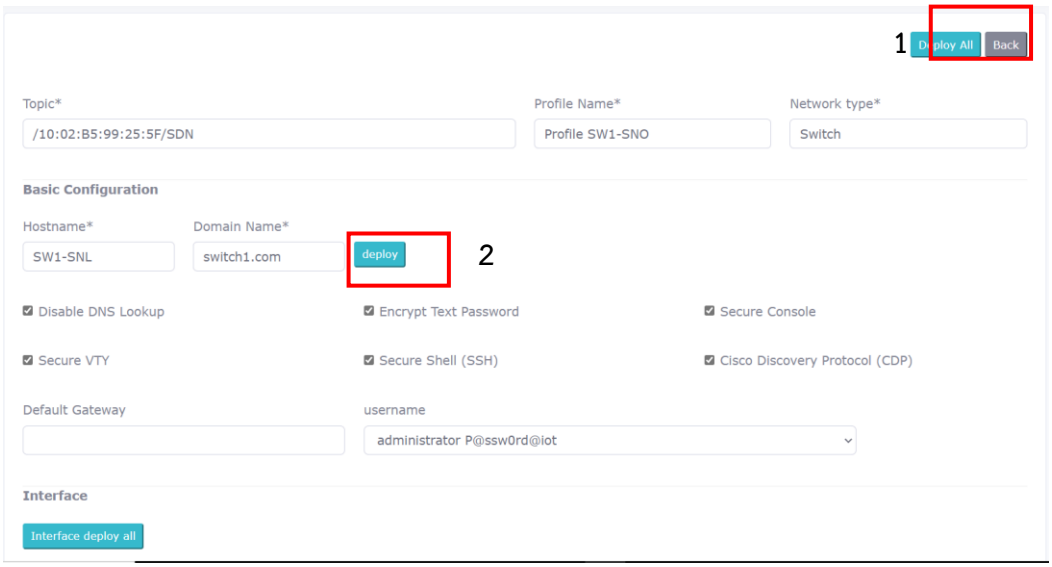
2. ขั้นตอนการส่งคำสั่งสำหรับการกำหนดค่าอุปกรณ์เครือข่าย



ภาพที่ ก.7 ขั้นตอนการส่งคำสั่งสำหรับการกำหนดค่าอุปกรณ์เครือข่าย

จากภาพที่ ก.7 ขั้นตอนการส่งคำสั่งสำหรับการกำหนดค่าอุปกรณ์เครือข่าย

หมายเลขที่ 1 configure เข้าสู่การส่งคำสั่งไปยังอุปกรณ์เครือข่าย

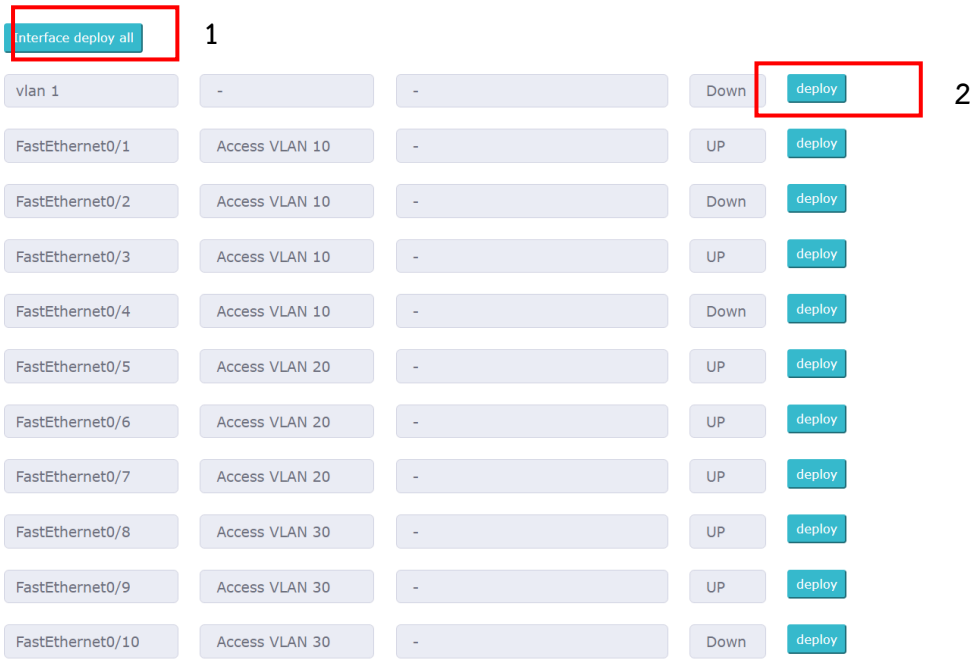


ภาพที่ ก.8 ขั้นตอนการส่งคำสั่งสำหรับการกำหนดค่าอุปกรณ์เครือข่าย

จากภาพที่ ก.8 ขั้นตอนการส่งคำสั่งสำหรับการกำหนดค่าอุปกรณ์เครือข่าย

หมายเลขที่ 1 Deploy All คือการส่งทุกคำสั่งที่อยู่ใน Profile ไปยังอุปกรณ์เครือข่าย

หมายเลขที่ 2 Deploy คือการส่งคำสั่งเฉพาะ Hostname และ Domain Name



ภาพที่ ก.9 ขั้นตอนการส่งคำสั่งสำหรับการกำหนดค่าอุปกรณ์เครือข่าย

จากภาพที่ ก.9 ขั้นตอนการส่งคำสั่งสำหรับการกำหนดค่าอุปกรณ์เครือข่าย

หมายเลขที่ 1 Deploy All คือการส่งทุกคำสั่งเฉพาะส่วนของ Interface

หมายเลขที่ 2 Deploy คือการส่งคำสั่งเฉพาะส่วนของ Interface

ภาคผนวก

ภาคผนวก ข

รายละเอียดชุดข้อมูลโค้ดโปรแกรมการจัดการของอุปกรณ์ IoT และ รายละเอียดชุดข้อมูลโค้ดโปรแกรมการจัดการจัดสมดุลการเชื่อมต่อ สำหรับสแปนนิ่งทรีโปรโตคอล

รายละเอียดชุดข้อมูลโค้ดโปรแกรมการจัดการของอุปกรณ์ IoT

```
#include "M5Atom.h"

#include <WiFiManager.h>

#include <Preferences.h>

#include <PubSubClient.h>

#include <Esp32WifiManager.h>

#include <WiFi.h>

WiFiManager wm;

//Create a wifi manager

WifiManager manager;

String sent;

String topic_str, payload_str;

//// WiFi Setting

//#define WIFI_STA_NAME "CECRRU-WIFI"

//#define WIFI_STA_PASS NULL


// MQTT Setting

#define MQTT_SERVER "172.16.20.12"

#define MQTT_PORT 1883

#define MQTT_USERNAME "administrator"

#define MQTT_PASSWORD "P@ssw0rd@DevNet"

#define MQTT_NAME "DevNet1"
```

```
WiFiClient client;          // Get WiFiClient for connecting to TCP.
PubSubClient mqtt(client); // Send the object from WiFiClient to the active PubSubClient library named mqtt.
// LED
int LED = 13;


void setup()
{
  // Setup M5 library for read RS232 of Network Device
  M5.begin();
  Serial2.begin(9600, SERIAL_8N1, 16, 17); // 16 RXD,17 TXD
  // Serial Baud Rate
  Serial.begin(115200);
  //  Setup_WiFi();
  //Setup ESP32 for Scan Network WiFi
  manager.setupScan();
  //  Serial.print(WiFi.macAddress());
  // Setup Name WiFi
  wm.autoConnect("10:02:B5:99:25:5F");
  // Connect to MQTT
  mqtt.setServer(MQTT_SERVER, MQTT_PORT);
  mqtt.setCallback(callback);
  // LED CHECK STATUS WIFI
```

```
  pinMode(LED, OUTPUT);
}

void loop()
{
  //Callback Funtion Mqtt
  mqtt_loop();
  //Check data from RS232 of M5
  if (Serial2.available())
  {
    sent = Serial2.readString();
    comparison(sent);
    mqtt_pub(sent);
  }
  if (Serial.available())
  {
    Serial2.write(Serial.read());
  }
  //Check WiFi is Connect if connected give LED Write else WiFi is Not Connect give LED Low
  if (WiFi.status() == WL_CONNECTED) {
    digitalWrite(LED, HIGH);
  }
```

```
  else {
    digitalWrite(LED, LOW);
    delay(500);
    digitalWrite(LED, HIGH);
    delay(500);
  }
}

void Setup_WiFi()
{
  // read the serial port for new passwords and maintain connections
  manager.loop();
  if (manager.getState() == Connected) {
    // use the Wifi Stack now connected
  }
  else {
    //reset saved settings
    wm.resetSettings();
  }
}
// Connect MQTT Server
void mqtt_loop()
```

```
{
  if (mqtt.connected() == false) {
    Serial.print("MQTT connection... ");
    if (mqtt.connect(MQTT_NAME, MQTT_USERNAME, MQTT_PASSWORD)) {
      Serial.print("connected");
      mqtt.subscribe("/10:02:B5:99:25:5F/SDN");
    }
    else {
      Serial.println("failed");
      delay(5000);
    }
  }
  else {
    mqtt.loop();
  }
}

// MQTT calback
void callback(char *topic, byte *payload, unsigned int length)
{
  payload[length] = '\0';
  topic_str = topic;
```

```
  payload_str = (char *)payload;
  //    Serial.println(payload_str);
  if (payload_str != NULL) {
    Serial2.write(13);
    Serial2.write(13);
    Serial2.write(payload_str.c_str());
//    Serial.print(payload_str.c_str());
    Serial2.write(13);
  }
}
// MQTT Sender to Software
String mqtt_pub(String mqtt_input)
{
  //  delay(1000);
  mqtt.publish("/10:02:B5:99:25:5F/IOT", mqtt_input.c_str());
}
// String comparion to Network Device
String comparison(String data_input)
{
  Serial.println(data_input);
  if (data_input.indexOf("Would you like to enter the initial configuration dialog? [yes/no]:") >
-1 | data_input.indexOf("System configuration has been modified. Save? [yes/no]:") > -1)
  {
```

```
    Serial2.write("no");
    Serial2.write(13);
  }
  else if (data_input.indexOf("Press RETURN to get started!") > -1 | data_input.indexOf("Press RETURN to get started.") > -1 | data_input.indexOf("%LINEPROTO-5-UPDOWN:") > -1)
  {
    Serial2.write(13);
  }
  else if (data_input.indexOf(">") > -1)
  {
    Serial2.write("enable");
    Serial2.write(13);
  }
  else if (data_input.indexOf("More") > -1 | data_input.indexOf("!") > -1 | data_input.indexOf("--More--") > -1 | data_input.indexOf("export@cisco.com.") > -1) {
    Serial2.write(32);
    Serial2.write(32);
    Serial2.write(32);
  }
  else if (data_input.indexOf("Erasing the nvram filesystem will remove all configuration files! Continue? [confirm]") > -1 | data_input.indexOf("Proceed with reload? [confirm]") > -1) {
    Serial2.write(13);
  }
```

```
  else if (data_input.indexOf("Password:") > -1) {
    Serial2.write("P@ssw0rd@iot");
    Serial2.write(13);
  }
}
```

รายละเอียดชุดข้อมูลโค้ดโปรแกรมการจัดการจัดสมดุลการเชื่อมต่อสำหรับสแปนนิ่งทรีโปรโตคอล

```
from datetime import date
import io
from re import A
import mysql.connector
import paho.mqtt.publish as mqtt
import time
import sched
from datetime import datetime
import statistics as stats
from mypacket.VL_2_G2 import VL_2_MIN, VL_2_MAX, VL_2_AVG
from mypacket.VL_3_G2 import VL_3_MIN, VL_3_MAX, VL_3_AVG
from mypacket.VL_4_G2 import VL_4_MIN, VL_4_MAX, VL_4_AVG
from mypacket.VL_5_G2 import VL_5_MIN, VL_5_MAX, VL_5_AVG
from mypacket.VL_6_G2 import VL_6_MIN, VL_6_MAX, VL_6_AVG
from mypacket.VL_3_G3 import VL_3_MIN_G3, VL_3_MAX_G3, VL_3_AVG_G3
```

```
from mypacket.VL_4_G3 import VL_4_MIN_G3, VL_4_MAX_G3, VL_4_AVG_G3

from mypacket.VL_5_G3 import VL_5_MIN_G3, VL_5_MAX_G3, VL_5_AVG_G3

from mypacket.VL_6_G3 import VL_6_MIN_G3, VL_6_MAX_G3, VL_6_AVG_G3

from mypacket.VL_7_G3 import VL_7_MIN_G3, VL_7_MAX_G3, VL_7_AVG_G3

from mypacket.VL_8_G3 import VL_8_MIN_G3, VL_8_MAX_G3, VL_8_AVG_G3

from mypacket.VL_9_G3 import VL_9_MIN_G3, VL_9_MAX_G3, VL_9_AVG_G3

from mypacket.VL_10_G3 import VL_10_MIN_G3, VL_10_MAX_G3, VL_10_AVG_G3



hostname = "172.16.20.12"

port = 1883

mqtt_auth = {

    'username': 'administrator',

    'password': 'P@ssw0rd@DevNet'

}

mydb = mysql.connector.connect(

    host="172.16.20.63",

    user="administrator",

    password="P@ssw0rd",

    database="devnet"

)

s = sched.scheduler(time.time, time.sleep)
```

```
def status():
    pass
def dictfetchall(cursor):
    columns = [col[0] for col in cursor.description]
    return [
        dict(zip(columns, row))
        for row in cursor.fetchall()
    ]
def round_robin(id, command):
    count1 = 0
    count2 = 0
    range_list1 = len(id)
    range_list2 = len(command)
    event = []
    id_count = {}
    lb = mydb.cursor()
    sql_lb = "INSERT INTO network_load_balance_stp_table
(command,time,date,switch_id,root_bridge) VALUES (%s,%s,%s,%s,%s)"
    for i in id:
        if count2 == range_list2:
            break
        else:
            for j, k in command.items():
```

```
            event.append(i)
            today = date.today()
            now = datetime.now().time()
            val = (k, now, today, i, j)
            lb.execute(sql_lb, val)
            # print(i, j, k)
            count1 += 1
            i += 1
            count2 += 1
            mydb.commit()
            if count1 == range_list1:
                i = id[0]
                count1 = 0
for i in event:
    if i in id_count:
        id_count[i] += 1
    else:
        id_count[i] = 1
for i, j in id_count.items():
    count = mydb.cursor()
    sql_count = "INSERT INTO network_count_table (switch_id,count) VALUES (%s,%s)"
    val_count = (i, j)
```

```
        count.execute(sql_count, val_count)
        mydb.commit()



def dynamic(**kwargs):
    # print("this is funtion dynamic")
    keywork = kwargs
    list = keywork['list_all']
    lb_min = 1
    lb_max = 1
    lb_avg = 1
    if keywork['vlan_range'] == 2 and keywork['iot_num'] == 2 and lb_min ==
keywork['lb_min']:
        VL_2_MIN(list)

    if keywork['vlan_range'] == 2 and keywork['iot_num'] == 2 and lb_max ==
keywork['lb_max']:
        VL_2_MAX(list)

    if keywork['vlan_range'] == 2 and keywork['iot_num'] == 2 and lb_avg == keywork['lb_avg']:
        VL_2_AVG(list)
```

```
    if keywork['vlan_range'] == 3 and keywork['iot_num'] == 2 and lb_min ==
keywork['lb_min']:
        VL_3_MIN(list)

    if keywork['vlan_range'] == 3 and keywork['iot_num'] == 2 and lb_max ==
keywork['lb_max']:
        VL_3_MAX(list)

    if keywork['vlan_range'] == 3 and keywork['iot_num'] == 2 and lb_avg == keywork['lb_avg']:
        VL_3_AVG(list)

    if keywork['vlan_range'] == 4 and keywork['iot_num'] == 2 and lb_min ==
keywork['lb_min']:
        VL_4_MIN(list)

    if keywork['vlan_range'] == 4 and keywork['iot_num'] == 2 and lb_max ==
keywork['lb_max']:
        VL_4_MAX(list)

    if keywork['vlan_range'] == 4 and keywork['iot_num'] == 2 and lb_avg == keywork['lb_avg']:
        VL_4_AVG(list)

    if keywork['vlan_range'] == 5 and keywork['iot_num'] == 2 and lb_min ==
keywork['lb_min']:
```

```
        VL_5_MIN(list)


    if keywork['vlan_range'] == 5 and keywork['iot_num'] == 2 and lb_max ==
keywork['lb_max']:

        VL_5_MAX(list)


    if keywork['vlan_range'] == 5 and keywork['iot_num'] == 2 and lb_avg == keywork['lb_avg']:

        VL_5_AVG(list)


    if keywork['vlan_range'] == 6 and keywork['iot_num'] == 2 and lb_min ==
keywork['lb_min']:

        VL_6_MIN(list)


    if keywork['vlan_range'] == 6 and keywork['iot_num'] == 2 and lb_max ==
keywork['lb_max']:

        VL_6_MAX(list)


    if keywork['vlan_range'] == 6 and keywork['iot_num'] == 2 and lb_avg == keywork['lb_avg']:

        VL_6_AVG(list)


    if keywork['vlan_range'] == 7 and keywork['iot_num'] == 3 and lb_min ==
keywork['lb_min']:

        VL_3_MIN_G3(list)
```

```
    if keywork['vlan_range'] == 7 and keywork['iot_num'] == 3 and lb_max ==
keywork['lb_max']:
        VL_3_MAX_G3(list)

    if keywork['vlan_range'] == 7 and keywork['iot_num'] == 3 and lb_avg == keywork['lb_avg']:
        VL_3_AVG_G3(list)

    if keywork['vlan_range'] == 8 and keywork['iot_num'] == 3 and lb_min ==
keywork['lb_min']:
        VL_4_MIN_G3(list)

    if keywork['vlan_range'] == 8 and keywork['iot_num'] == 3 and lb_max ==
keywork['lb_max']:
        VL_4_MAX_G3(list)

    if keywork['vlan_range'] == 8 and keywork['iot_num'] == 3 and lb_avg == keywork['lb_avg']:
        VL_4_AVG_G3(list)

    if keywork['vlan_range'] == 9 and keywork['iot_num'] == 3 and lb_min ==
keywork['lb_min']:
        VL_5_MIN_G3(list)
```

```
    if keywork['vlan_range'] == 9 and keywork['iot_num'] == 3 and lb_max ==
keywork['lb_max']:
        VL_5_MAX_G3(list)

    if keywork['vlan_range'] == 9 and keywork['iot_num'] == 3 and lb_avg == keywork['lb_avg']:
        VL_5_AVG_G3(list)

    if keywork['vlan_range'] == 10 and keywork['iot_num'] == 3 and lb_min ==
keywork['lb_min']:
        VL_6_MIN_G3(list)

    if keywork['vlan_range'] == 10 and keywork['iot_num'] == 3 and lb_max ==
keywork['lb_max']:
        VL_6_MAX_G3(list)

    if keywork['vlan_range'] == 10 and keywork['iot_num'] == 3 and lb_avg ==
keywork['lb_avg']:
        VL_6_AVG_G3(list)
sw1 = mydb.cursor()
sql_sw1 = "SELECT network.id,vlan.vlan_number AS access FROM network_iot_table AS iot
JOIN network_network_table AS network ON network.id = iot.network_id JOIN
network_interface_table AS interface ON interface.network_id = network.id LEFT JOIN
network_vlan_table AS vlan ON vlan.id = interface.access_id WHERE iot.subscribe =
'/10:02:B5:99:25:5F/IOT' AND interface.status = 1 AND interface.access_id IS NOT NULL GROUP
BY vlan.vlan_number"
```

```
sw1.execute(sql_sw1)

interface_sw1 = dictfetchall(sw1)


sw2 = mydb.cursor()

sql_sw2 = "SELECT network.id,vlan.vlan_number AS access FROM network_iot_table AS iot
JOIN network_network_table AS network ON network.id = iot.network_id JOIN
network_interface_table AS interface ON interface.network_id = network.id LEFT JOIN
network_vlan_table AS vlan ON vlan.id = interface.access_id WHERE iot.subscribe =
'/30:AE:A4:03:3C:CC/IOT' AND interface.status = 1 AND interface.access_id IS NOT NULL
GROUP BY vlan.vlan_number"

sw2.execute(sql_sw1)

interface_sw2 = dictfetchall(sw2)

sw3 = mydb.cursor()

sql_sw3 = "SELECT network.id,vlan.vlan_number AS access FROM network_iot_table AS iot
JOIN network_network_table AS network ON network.id = iot.network_id JOIN
network_interface_table AS interface ON interface.network_id = network.id LEFT JOIN
network_vlan_table AS vlan ON vlan.id = interface.access_id WHERE iot.subscribe =
'/66:96:2C:08:D4:49/IOT' AND interface.status = 1 AND interface.access_id IS NOT NULL
GROUP BY vlan.vlan_number"

sw3.execute(sql_sw3)

interface_sw3 = dictfetchall(sw3)

iot = mydb.cursor()

sql_iot = "SELECT network.id,iot.lb_max,iot.lb_min,iot.lb_avg FROM network_iot_table AS iot
JOIN network_network_table AS network ON network.id = iot.network_id WHERE iot.lb_max
= 1 OR iot.lb_avg = 1 OR iot.lb_min = 1 "

iot.execute(sql_iot)
```

```
iot_list = dictfetchall(iot)

lb_mode = mydb.cursor()

sql_lb_mode = "SELECT iot.lb_max,iot.lb_min,iot.lb_avg FROM network_iot_table AS iot
WHERE iot.lb_max = 1 OR iot.lb_avg = 1 OR iot.lb_min = 1 GROUP BY
iot.lb_max,iot.lb_avg,iot.lb_min"

lb_mode.execute(sql_lb_mode)

list_lb_mode = dictfetchall(lb_mode)

iot_num = 0

list_sw1 = []

list_sw2 = []

list_sw3 = []

load_balance = mydb.cursor()

sql_load_balance = (
    "SELECT stp.id,stp.command,stp.switch_id FROM network_load_balance_stp_table AS stp")

load_balance.execute(sql_load_balance)

load_balance = dictfetchall(load_balance)

row_robin = mydb.cursor()

sql_row_robin = ("SELECT network.id FROM network_network_table AS network JOIN
network_network_type_table AS network_type ON network_type.id =
network.network_type_id JOIN network_interface_table AS interface ON interface.network_id
= network.id LEFT JOIN network_load_balance_stp_table AS lb ON lb.switch_id = network.id
WHERE network_type = 'Switch' AND interface.access_id IS NOT NULL GROUP BY network.id")

row_robin.execute(sql_row_robin)

row_robin = dictfetchall(row_robin)
```

```
list_row_robin = []
for i in row_robin:
    for j in i.values():
        list_row_robin.append(j)
list_command_vlan = {}
resoults = 0
different = 50
############################
network_id = []
lb_min = 0
lb_max = 0
lb_avg = 0
for i in list_lb_mode:
    if i['lb_min'] == 1:
        lb_min = 1
    if i['lb_max'] == 1:
        lb_max = 1
    if i['lb_avg'] == 1:
        lb_avg = 1
for iot in iot_list:
    network_id.append(iot['id'])
    iot_num += 1
```

```
for sw1 in interface_sw1:
    list_sw1.append(int(sw1['access']))
for sw2 in interface_sw2:
    list_sw2.append(int(sw2['access']))
for sw3 in interface_sw3:
    list_sw3.append(int(sw3['access']))
    setA = 0
if iot_num == 3:
    setA = set(list_sw1) & set(list_sw2) & set(list_sw3)
elif iot_num == 2:
    setA = set(list_sw1) & set(list_sw2)
    list_all = list(setA)
    vlan_range = 0

for i in range(len(list_all)):
    vlan_range = i+1
    list_all.sort()
############################
if load_balance == []:
    # print(type(CVL))
    for i in list_all:
        command = "spanning-tree vlan "+str(i)+" root primary"
```

```
        list_command_vlan[i] = command


    round_robin(list_row_robin, list_command_vlan)
    sw_ch = mydb.cursor()
    sql_sw = "SELECT network.id FROM network_network_table AS network JOIN
network_iot_table AS iot ON iot.network_id = network.id WHERE iot.lb_min = 1 OR
iot.lb_max = 1 OR iot.lb_avg = 1"
    sw_ch.execute(sql_sw)
    sw_ch = dictfetchall(sw_ch)
    len_sw_ch1 = 0
    if len(sw_ch) == 2:
        sum_sw1 = 0
        sum_sw2 = 0
        for i in sw_ch:
            differ = mydb.cursor()
            sql_differ = "SELECT stp.root_bridge FROM network_load_balance_stp_table AS stp
WHERE stp.switch_id = %s"
            val_differ = (i['id'],)
            differ.execute(sql_differ, val_differ)
            differ = dictfetchall(differ)
            if len_sw_ch1 < len(sw_ch):
                for j in differ:
                    # print(j)
```

```
                len_sw_ch1 += 1
                sum = mydb.cursor()
                sql_sum = "SELECT COUNT(interface.id) AS num FROM
network_interface_table AS interface JOIN network_vlan_table AS vlan ON vlan.id =
interface.access_id JOIN network_network_table AS network ON network.id =
interface.network_id WHERE vlan.vlan_number = %s AND interface.status = 1"
                val_sum = (j['root_bridge'],)
                sum.execute(sql_sum, val_sum)
                sum = dictfetchall(sum)
                for k in sum:
                    sum_sw1 += int(k['num'])
            elif len_sw_ch1 >= len(sw_ch):
                for j in differ:
                len_sw_ch1 += 1
                sum = mydb.cursor()
                sql_sum = "SELECT COUNT(interface.id) AS num FROM
network_interface_table AS interface JOIN network_vlan_table AS vlan ON vlan.id =
interface.access_id JOIN network_network_table AS network ON network.id =
interface.network_id WHERE vlan.vlan_number = %s AND interface.status = 1"
                val_sum = (j['root_bridge'],)
                sum.execute(sql_sum, val_sum)
                sum = dictfetchall(sum)
                for k in sum:
                    sum_sw2 += int(k['num'])
```

```
        resoults_min = min(sum_sw1, sum_sw2)
        resoults_max = max(sum_sw1, sum_sw2)
        resoults = (resoults_min/resoults_max)*100
        if resoults >= different:
            deploy_stp = mydb.cursor()
            sql_deploy_stp = "SELECT iot.topic,stp.command FROM
network_load_balance_stp_table AS stp JOIN network_network_table AS network ON
network.id = stp.switch_id JOIN network_iot_table AS iot ON iot.network_id = network.id"
            deploy_stp.execute(sql_deploy_stp)
            deploy_stp = dictfetchall(deploy_stp)
            command_stp = []
            topic_stp = []
            command_all = ['end', 'wr']
            count = 0
            for i in deploy_stp:
                topic_stp.append(i['topic'])
                command_stp.append(i['command'])
            topic = set(topic_stp)
            for i in topic:
                cm = "config terminal"
                mqtt.single(i, cm, qos=0, hostname="172.16.20.12",
                            port=1883, auth=mqtt_auth)
            for i in deploy_stp:
```

```
                mqtt.single(i['topic'], i['command'], qos=0, hostname="172.16.20.12",
                            port=1883, auth=mqtt_auth)
                time.sleep(2)
                count += 1
                if count == len(deploy_stp):
                    for i in topic:
                        for j in command_all:
                            mqtt.single(i, j, qos=0, hostname="172.16.20.12",
                                        port=1883, auth=mqtt_auth)
                            time.sleep(2)
            now = datetime.now().time()
            today = date.today()
            diff = mydb.cursor()
            sql_diff = "INSERT INTO network_different_table (different,time,date) VALUES
(%s,%s,%s)"
            val_diff = (resoults, now, today)
            diff.execute(sql_diff, val_diff)
            mydb.commit()
        else:
            dynamic(vlan_range=vlan_range, iot_num=iot_num,
                    lb_min=lb_min, lb_max=lb_max, lb_avg=lb_avg, list_all=list_all)
elif load_balance != []:
    sw_ch = mydb.cursor()
```

```
    sql_sw = "SELECT network.id,count.count FROM network_network_table AS network JOIN
network_iot_table AS iot ON iot.network_id = network.id JOIN network_count_table AS
count ON count.switch_id = network.id WHERE iot.lb_min = 1 OR iot.lb_max = 1 OR
iot.lb_avg = 1 ORDER BY count.id DESC LIMIT 2"
    sw_ch.execute(sql_sw)
    sw_ch = dictfetchall(sw_ch)
    len_sw_ch1 = 0
    change = 0
    for i in sw_ch:
        change += i['count']
    if change < vlan_range:
        dynamic(vlan_range=vlan_range, iot_num=iot_num,
                lb_min=lb_min, lb_max=lb_max, lb_avg=lb_avg, list_all=list_all)
    else:
        if len(sw_ch) == 2 and vlan_range == 4:
            sum_sw1 = 0
            sum_sw2 = 0
            for i in sw_ch:
                differ = mydb.cursor()
                sql_differ = "SELECT stp.root_bridge FROM network_load_balance_stp_table AS
stp  WHERE stp.switch_id = %s ORDER BY stp.id DESC LIMIT 2"
                val_differ = (i['id'],)
                differ.execute(sql_differ, val_differ)
```

```
                differ = dictfetchall(differ)
                if len_sw_ch1 < len(sw_ch):
                    for j in differ:
                        len_sw_ch1 += 1
                        sum = mydb.cursor()
                        sql_sum = "SELECT COUNT(interface.id) AS num FROM
network_interface_table AS interface JOIN network_vlan_table AS vlan ON vlan.id =
interface.access_id JOIN network_network_table AS network ON network.id =
interface.network_id WHERE vlan.vlan_number = %s AND interface.status = 1"
                        val_sum = (j['root_bridge'],)
                        sum.execute(sql_sum, val_sum)
                        sum = dictfetchall(sum)
                        for k in sum:
                            sum_sw1 += int(k['num'])
                elif len_sw_ch1 == len(sw_ch):
                    for j in differ:
                        len_sw_ch1 += 1
                        sum = mydb.cursor()
                        sql_sum = "SELECT COUNT(interface.id) AS num FROM
network_interface_table AS interface JOIN network_vlan_table AS vlan ON vlan.id =
interface.access_id JOIN network_network_table AS network ON network.id =
interface.network_id WHERE vlan.vlan_number = %s AND interface.status = 1"
                        val_sum = (j['root_bridge'],)
                        sum.execute(sql_sum, val_sum)
```

```
                sum = dictfetchall(sum)
                for k in sum:
                    sum_sw2 += int(k['num'])
        resoults_min = min(sum_sw1, sum_sw2)
        resoults_max = max(sum_sw1, sum_sw2)
        resoults = (resoults_min/resoults_max)*100
        if resoults >= different:
            now = datetime.now().time()
            today = date.today()
            diff = mydb.cursor()
            sql_diff = "INSERT INTO network_different_table (different,time,date) VALUES
(%s,%s,%s)"
            val_diff = (resoults, now, today)
            diff.execute(sql_diff, val_diff)
            mydb.commit()
        else:
            dynamic(vlan_range=vlan_range, iot_num=iot_num,
                    lb_min=lb_min, lb_max=lb_max, lb_avg=lb_avg, list_all=list_all)
    elif len(sw_ch) == 2 and vlan_range == 5:
        sum_sw1 = 0
        sum_sw2 = 0
        for i in sw_ch:
            differ = mydb.cursor()
```

```
            sql_differ = "SELECT stp.root_bridge FROM network_load_balance_stp_table AS
stp  WHERE stp.switch_id = %s ORDER BY stp.id DESC LIMIT %s"
            val_differ = (i['id'], i['count'],)
            differ.execute(sql_differ, val_differ)
            differ = dictfetchall(differ)
            # print(i['id'])
            if len_sw_ch1 < len(sw_ch):
                for j in differ:
                    # print(j)
                    len_sw_ch1 += 1
                    sum = mydb.cursor()
                    sql_sum = "SELECT COUNT(interface.id) AS num FROM
network_interface_table AS interface JOIN network_vlan_table AS vlan ON vlan.id =
interface.access_id JOIN network_network_table AS network ON network.id =
interface.network_id WHERE vlan.vlan_number = %s AND interface.status = 1"
                    val_sum = (j['root_bridge'],)
                    sum.execute(sql_sum, val_sum)
                    sum = dictfetchall(sum)
                    for k in sum:
                        sum_sw1 += int(k['num'])
            elif len_sw_ch1 >= len(sw_ch):
                for j in differ:
                    # print(j)
```

```
                    len_sw_ch1 += 1
                    sum = mydb.cursor()
                    sql_sum = "SELECT COUNT(interface.id) AS num FROM
network_interface_table AS interface JOIN network_vlan_table AS vlan ON vlan.id =
interface.access_id JOIN network_network_table AS network ON network.id =
interface.network_id WHERE vlan.vlan_number = %s AND interface.status = 1"
                    val_sum = (j['root_bridge'],)
                    sum.execute(sql_sum, val_sum)
                    sum = dictfetchall(sum)
                    for k in sum:
                        sum_sw2 += int(k['num'])
            resoults_min = min(sum_sw1, sum_sw2)
            resoults_max = max(sum_sw1, sum_sw2)
            resoults = (resoults_min/resoults_max)*100
            if resoults >= different:
                now = datetime.now().time()
                today = date.today()
                diff = mydb.cursor()
                sql_diff = "INSERT INTO network_different_table (different,time,date) VALUES
(%s,%s,%s)"
                val_diff = (resoults, now, today)
                diff.execute(sql_diff, val_diff)
                mydb.commit()
```

```
else:
    dynamic(vlan_range=vlan_range, iot_num=iot_num,
            lb_min=lb_min, lb_max=lb_max, lb_avg=lb_avg, list_all=list_all)
```