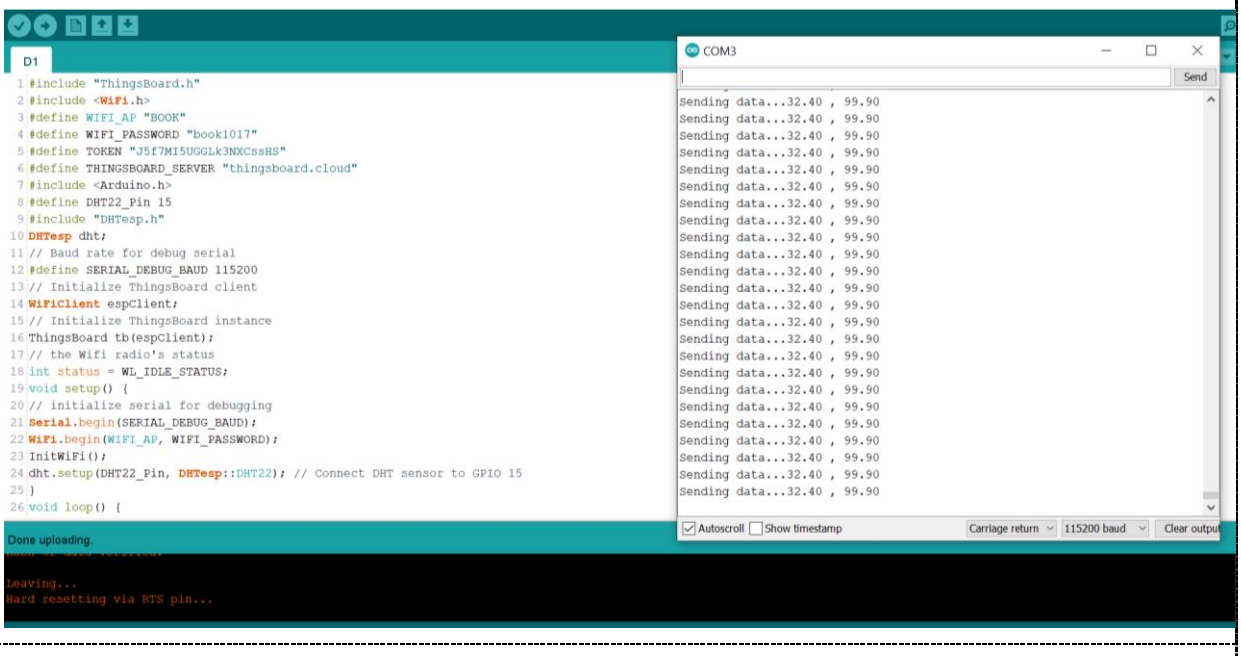
การใช้งาน ThingsBoard IoTs Platform เพื่อสร้างและจัดการระบบอัจริยะ
ThingsBoard IoTs Platform for smart system

ชื่อ-สกุล : นางสาวณัฐชยา ผ่องกุศล B6226718

6/6 -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_101 – ThingsBoard Data Monitor

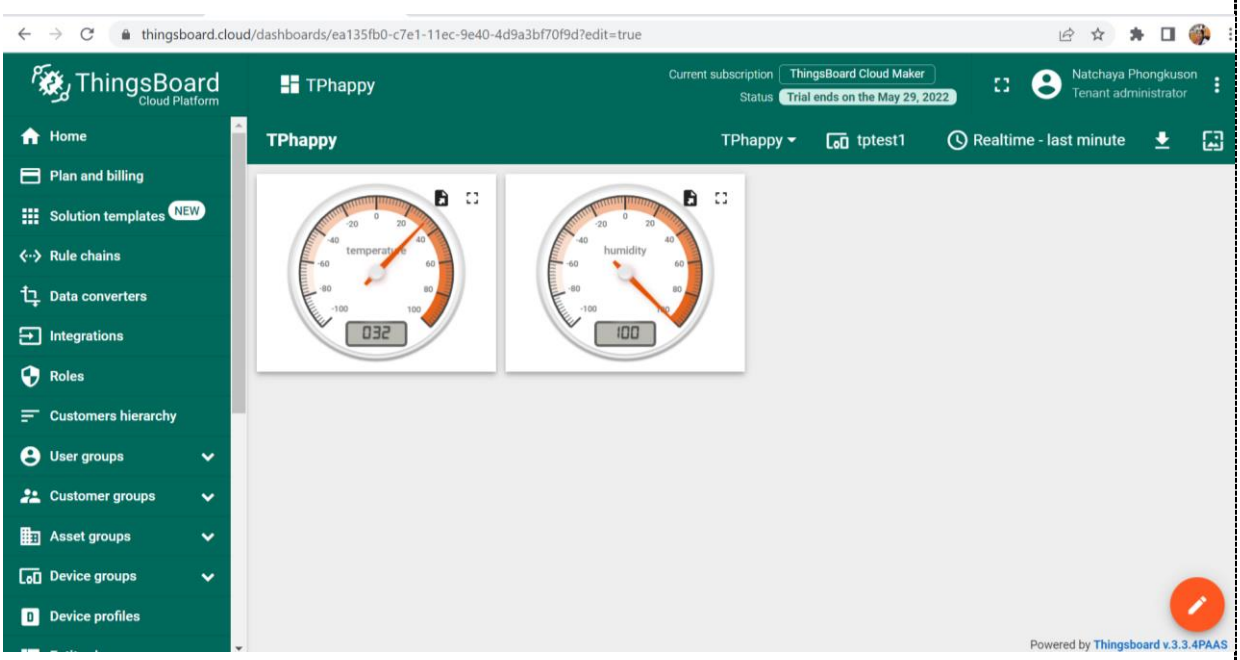
- Mission - 1/4: ให้ส่งข้อมูลค่า Humidity และ Temperatures จากเซ็นเซอร์ DHT-22 ไปยัง Dashboard

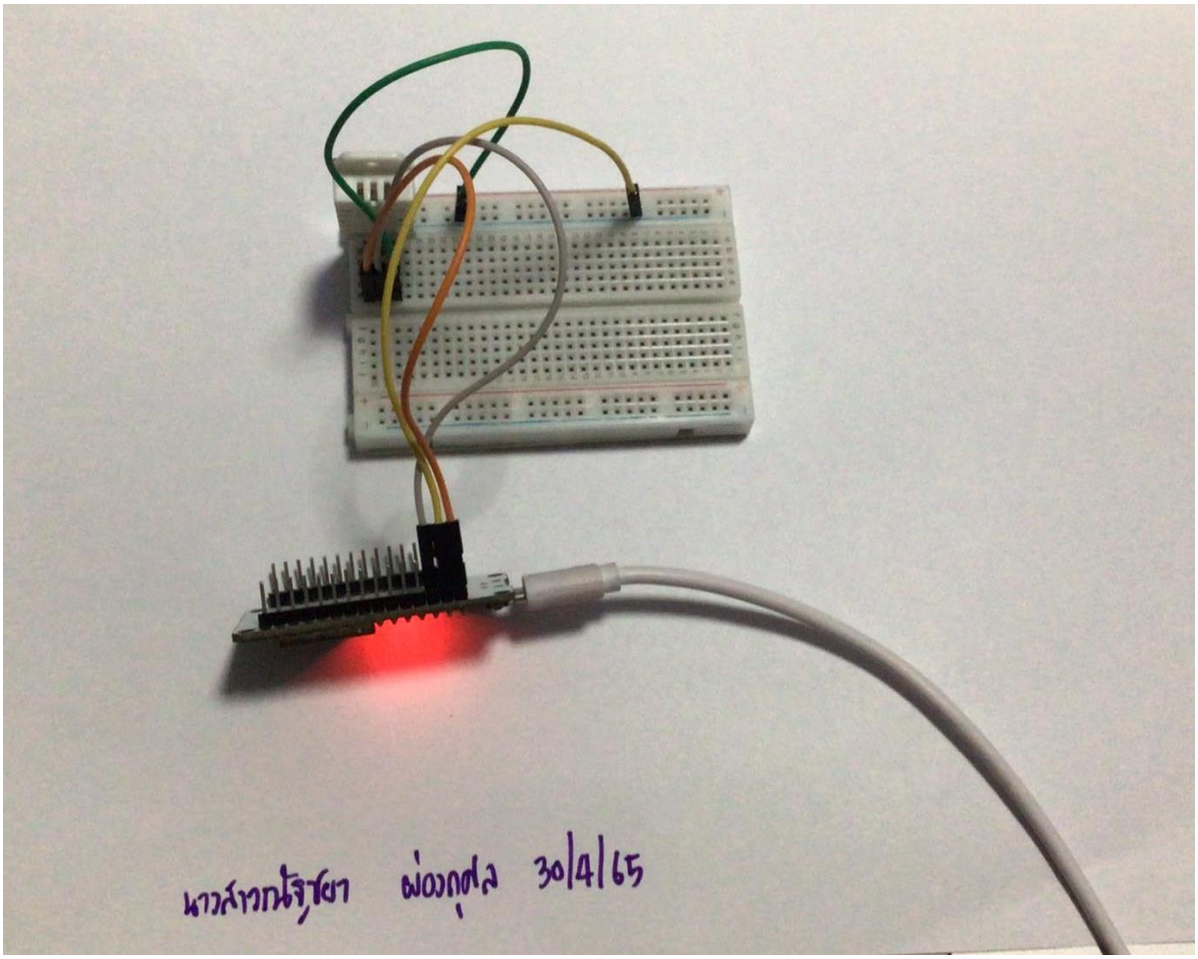


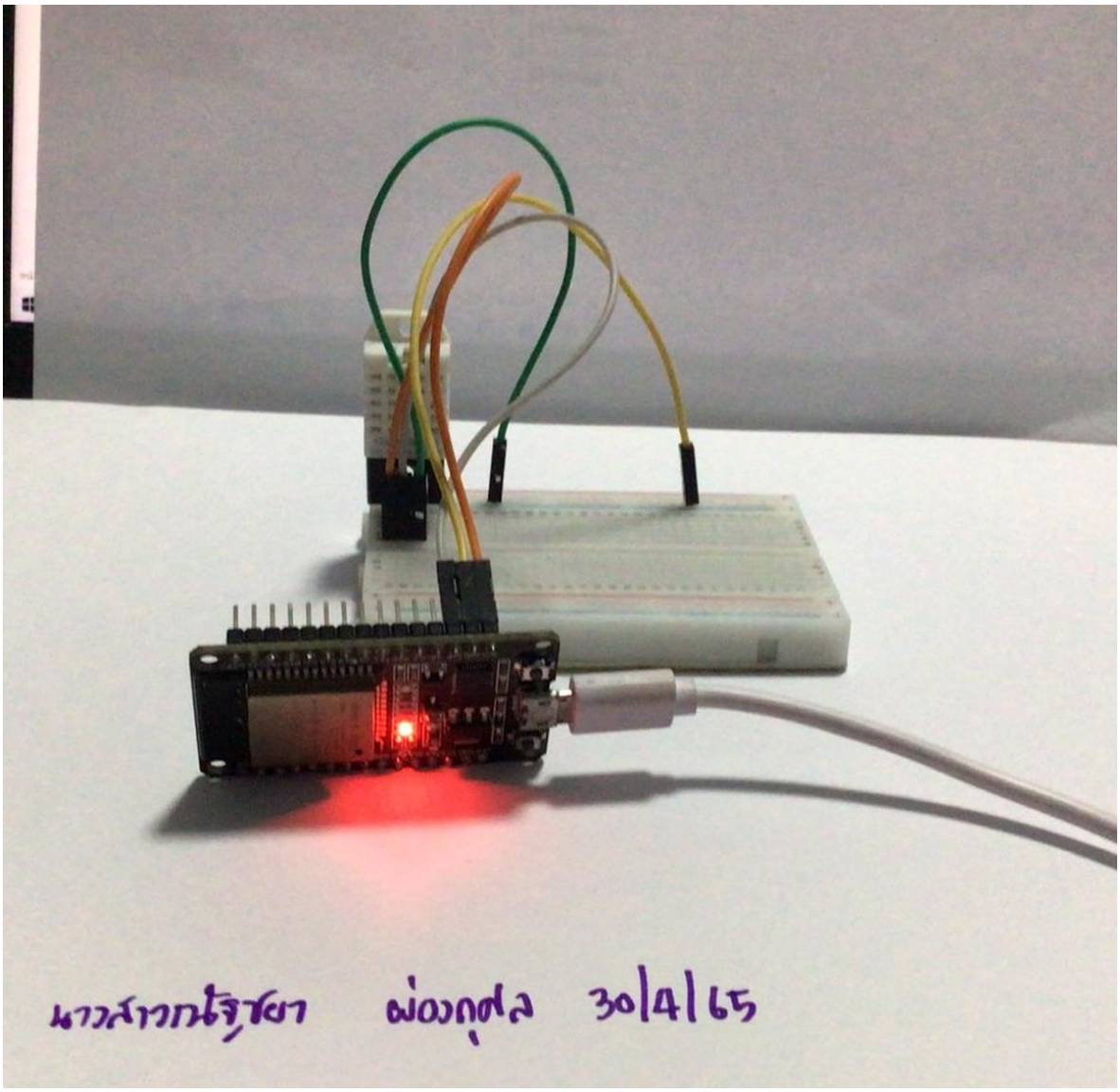

```
#include "ThingsBoard.h"
#include <WiFi.h>
#define WIFI_AP "BOOK"
#define WIFI_PASSWORD "book1017"
#define TOKEN "J5f7MI5UGGLk3NXCssHS"
#define THINGSBOARD_SERVER "thingsboard.cloud"
#include <Arduino.h>
#define DHT22_Pin 15
#include "DHTesp.h"
DHTesp dht;
// Baud rate for debug serial
#define SERIAL_DEBUG_BAUD 115200
// Initialize ThingsBoard client
```

```
WiFiClient espClient;
// Initialize ThingsBoard instance
ThingsBoard tb(espClient);
// the Wifi radio's status
int status = WL_IDLE_STATUS;
void setup() {
// initialize serial for debugging
Serial.begin(SERIAL_DEBUG_BAUD);
WiFi.begin(WIFI_AP, WIFI_PASSWORD);
InitWiFi();
dht.setup(DHT22_Pin, DHTesp::DHT22); // Connect DHT sensor to GPIO 15
}
void loop() {
if (WiFi.status() != WL_CONNECTED) {
reconnect();
}
if (!tb.connected()) {
// Connect to the ThingsBoard
Serial.print("Connecting to: ");
Serial.print(THINGSBOARD_SERVER);
Serial.print(" with token ");
Serial.println(TOKEN);
if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
Serial.println("Failed to connect");
return;
}
}
Serial.print("Sending data...");
// Uploads new telemetry to ThingsBoard using MQTT.
// See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
// for more details
//tb.sendTelemetryInt("temperature", xTempp);
//tb.sendTelemetryInt("humidity", xTempp);
```

```
Serial.print(dht.getTemperature() );
Serial.print(" , ");
Serial.println(dht.getHumidity());
tb.sendTelemetryFloat("temperature", dht.getTemperature() );
tb.sendTelemetryFloat("humidity", dht.getHumidity());
tb.loop();
delay(5000);
}
void InitWiFi()
{
Serial.println("Connecting to AP ...");
// attempt to connect to WiFi network
WiFi.begin(WIFI_AP, WIFI_PASSWORD);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("Connected to AP");
}
void reconnect() {
// Loop until we're reconnected
status = WiFi.status();
if ( status != WL_CONNECTED) {
WiFi.begin(WIFI_AP, WIFI_PASSWORD);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("Connected to AP");
}
}
```

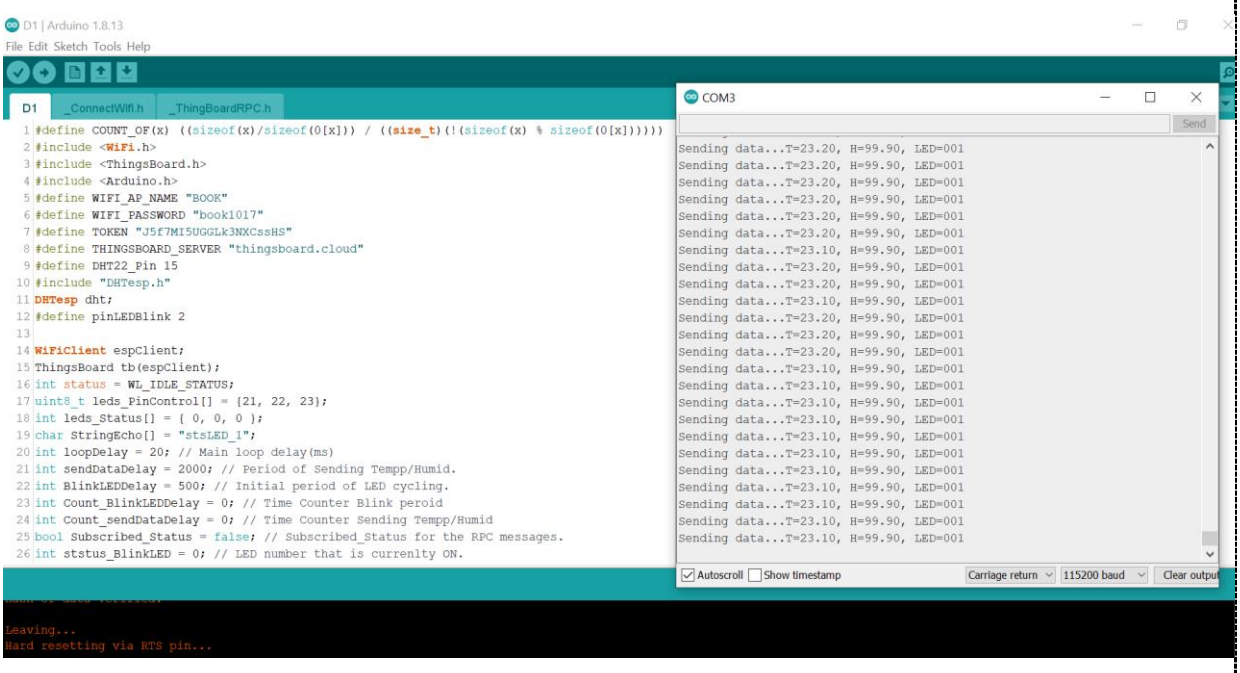

thingsboard.cloud/dashboards/ea135fb0-c7e1-11ec-9e40-4d9a3bf70f9d?edit=true
ThingsBoard
Cloud Platform
TPhappy
Current subscription
ThingsBoard Cloud Maker
Status
Trial ends on the May 29, 2022
Natchaya Phongkuson
Tenant administrator
Home
Plan and billing
Solution templates
NEW
Rule chains
Data converters
Integrations
Roles
Customers hierarchy
User groups
Customer groups
Asset groups
Device groups
Device profiles
TPhappy
TPhappy
tptest1
Realtime - last minute
temperature
032
humidity
100
Powered by Thingsboard v.3.3.4PAAS


30/4/65


นางสาวกชธิชา ผ่องกุล 30/4/65

Quiz_102 – ThingsBoard Data Monitor and Control

- Mission 2/4: ให้ส่งข้อมูลค่า Humidity และ Temperatures จากเซ็นเซอร์ DHT-22 ไปยัง ThingsBoard พร้อมทั้งควบคุม On/Off - 4 LED และ Blink Speed สำหรับอีก 1 LED




```
#define COUNT_OF(x) ((sizeof(x)/sizeof(0[x])) / ((size_t)(!(sizeof(x) % sizeof(0[x])))))
#include <WiFi.h>
#include <ThingsBoard.h>
#include <Arduino.h>
#define WIFI_AP_NAME "BOOK"
#define WIFI_PASSWORD "book1017"
#define TOKEN "J5f7MI5UGGLk3NXCssHS"
#define THINGSBOARD_SERVER "thingsboard.cloud"
#define DHT22_Pin 15
#include "DHTesp.h"
DHTesp dht;
#define pinLEDBlink 2

WiFiClient espClient;
ThingsBoard tb(espClient);
int status = WL_IDLE_STATUS;
uint8_t leds_PinControl[] = {21, 22, 23};
```

```
int leds_Status[] = { 0, 0, 0 };
char StringEcho[] = "stsLED_1";
int loopDelay = 20; // Main loop delay(ms)
int sendDataDelay = 2000; // Period of Sending Tempp/Humid.
int BlinkLEDDelay = 500; // Initial period of LED cycling.
int Count_BlinkLEDDelay = 0; // Time Counter Blink peroid
int Count_sendDataDelay = 0; // Time Counter Sending Tempp/Humid
bool Subscribed_Status = false; // Subscribed_Status for the RPC messages.
int ststus_BlinkLED = 0; // LED number that is currenlty ON.
#include "_ThingBoardRPC.h"
#include "_ConnectWifi.h"
//=====================================================
void setup() {
// Initialize serial for debugging
Serial.begin(115200);
WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
WiFi_Initial();
dht.setup(DHT22_Pin, DHTesp::DHT22); // Connect DHT sensor to GPIO 15
// Pinconfig
pinMode(pinLEDBlink, OUTPUT);
for (size_t i = 0; i < COUNT_OF(leds_PinControl); ++i) {
pinMode(leds_PinControl[i], OUTPUT);
}
}
//=====================================================
void loop() {
// Step0/6 - Loop Delay
delay(loopDelay);
Count_BlinkLEDDelay += loopDelay;
Count_sendDataDelay += loopDelay;
// Step1/6 - Check if next LED Blink
if (Count_BlinkLEDDelay > BlinkLEDDelay) {
digitalWrite(pinLEDBlink, ststus_BlinkLED);
```

```
ststus_BlinkLED = 1 - ststus_BlinkLED;
Count_BlinkLEDDelay = 0;
}
// Step 2/6 - Reconnect to WiFi, if needed
if (WiFi.status() != WL_CONNECTED) {
reconnect();
return;
}
// Step 3/6 - Reconnect to ThingsBoard, if needed
if (!tb.connected()) {
Subscribed_Status = false;
// Connect to the ThingsBoard
Serial.print("Connecting to: "); Serial.print(THINGSBOARD_SERVER);
Serial.print(" with token "); Serial.println(TOKEN);
if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
Serial.println("Failed to connect");
return;
}
}
// Step 4/6 - Subscribe for RPC, if needed
if (!Subscribed_Status) {
Serial.println("Subscribing for RPC...");
// Perform a subscription. All consequent data processing will happen in
// callbacks as denoted by callbacks[] array. Page 14 of 23
if (!tb.RPC_Subscribe(callbacks, COUNT_OF(callbacks))) {
Serial.println("Failed to subscribe for RPC");
return;
}
Serial.println("Subscribe done");
Subscribed_Status = true;
}
// Step 5/6 - Check if it is a time to send Tempp/Humid
if (Count_sendDataDelay > sendDataDelay) {
```

```
Serial.print("Sending data...");
float humidity = dht.getHumidity();
float temperature = dht.getTemperature();
tb.sendTelemetryFloat("temperature", temperature);
tb.sendTelemetryFloat("humidity", humidity);
Serial.print("T=" + String(temperature, 2) + ", ");
Serial.print("H=" + String(humidity, 2) + ", ");
Serial.print("LED=");
for (size_t i = 0; i < COUNT_OF(leds_PinControl); ++i) {
StringEcho[7] = 0x30 + i; // Set 0 to "0"
tb.sendTelemetryInt(StringEcho, leds_Status[i]);
Serial.print(leds_Status[i]);
}
Serial.println();
Count_sendDataDelay = 0;
}
// Step 6/6 - Process messages
tb.loop(); }
```

```
//_ConnectWifi.h
//=====================================================
void WiFi_Initial() {
Serial.println("Connecting to AP ..."); // attempt to connect to WiFi network
WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("\nConnected to AP");
Serial.print("Local IP = ");
Serial.println(WiFi.localIP());
}
//=====================================================
```

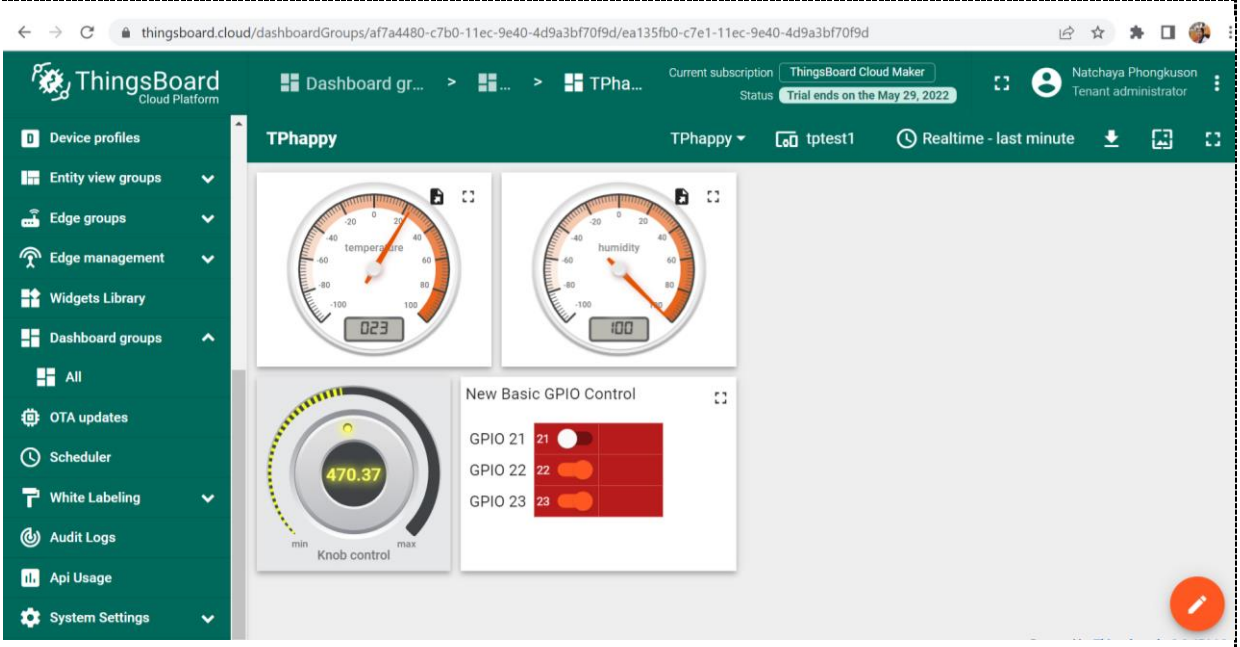
```
void reconnect() {
status = WiFi.status(); // Loop until we're reconnected
if ( status != WL_CONNECTED) {
WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("\nConnected to AP");
Serial.print("Local IP = ");
Serial.println(WiFi.localIP());
}
}
```

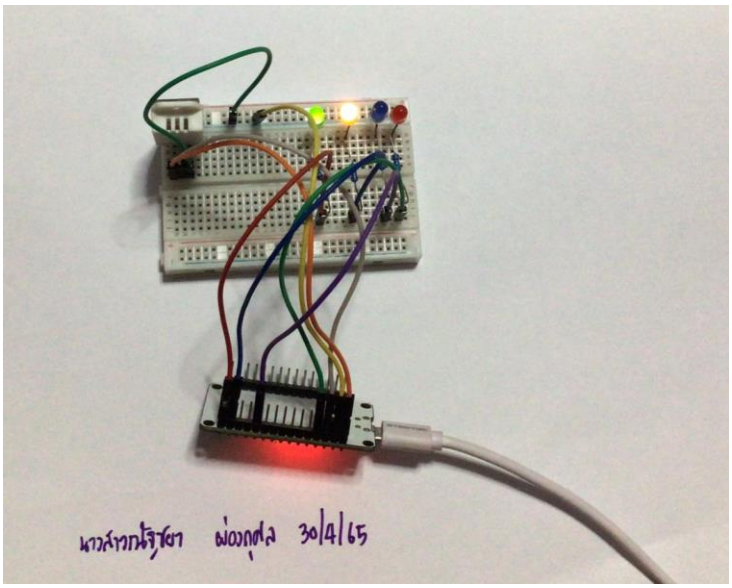
```
// _ThingBoardRPC.h
//#############################################################
// Processes function for RPC call "setValue"
// RPC_Data is a JSON variant, that can be queried using operator[]
// See https://arduinojson.org/v5/api/jsonvariant/subscript/ for more details
//===========================================================
RPC_Response processDelayChange(const RPC_Data &data)
{ Serial.println("Received the set delay RPC method");
BlinkLEDDelay = data;
Serial.print("Set new delay: ");
Serial.println(BlinkLEDDelay);
return RPC_Response(NULL, BlinkLEDDelay);
}
//#############################################################
// Processes function for RPC call "getValue"
// RPC_Data is a JSON variant, that can be queried using operator[]
// See https://arduinojson.org/v5/api/jsonvariant/subscript/ for more details
//===========================================================
RPC_Response processGetDelay(const RPC_Data &data) {
```

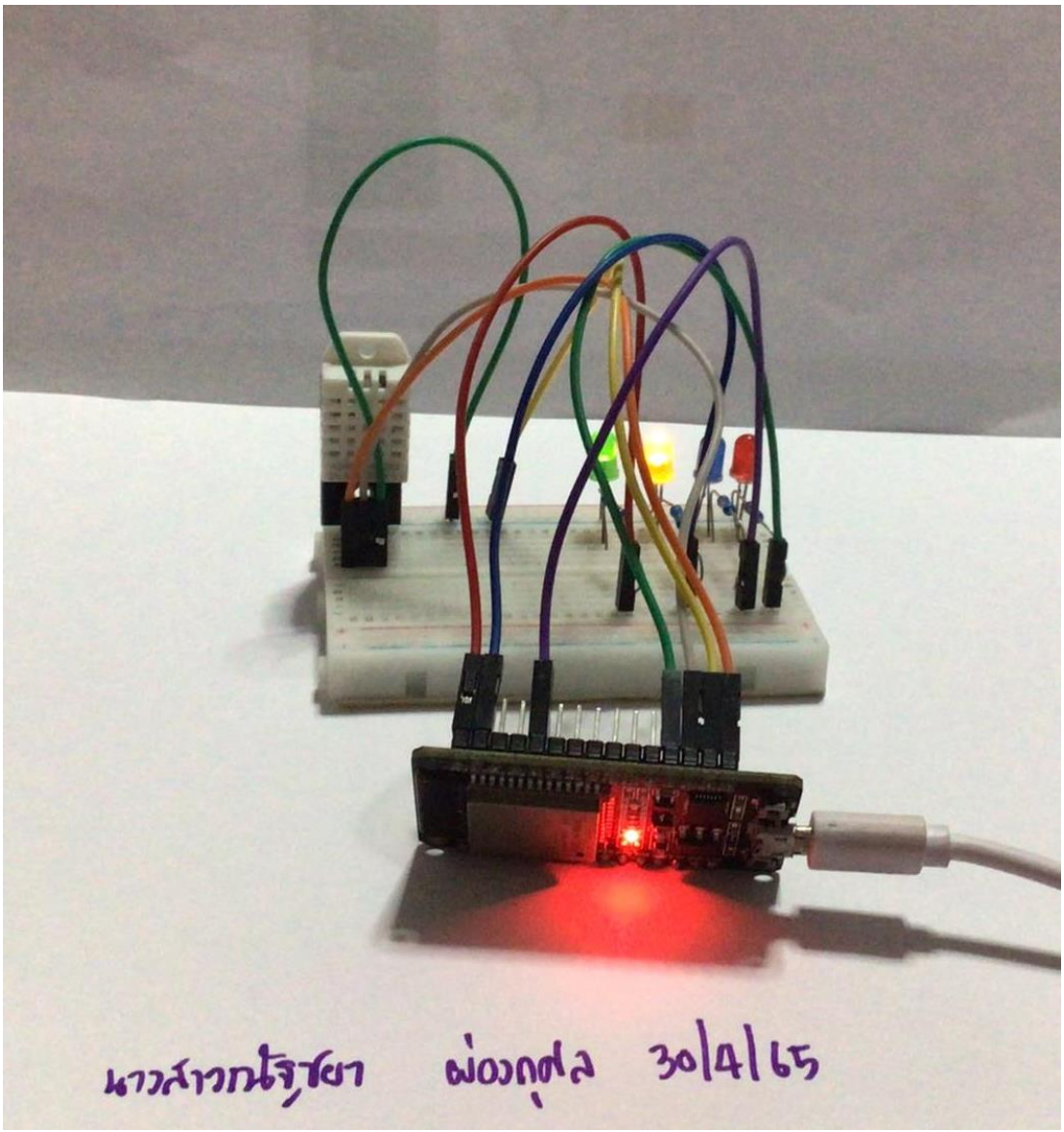
```
Serial.println("Received the get value method");
return RPC_Response(NULL, BlinkLEDDelay);
}
//############################################################
// Processes function for RPC call "setGpioStatus"
// RPC_Data is a JSON variant, that can be queried using operator[]
// See https://arduinojson.org/v5/api/jsonvariant/subscript/ for more details
//===========================================================
RPC_Response processSetGpioState(const RPC_Data &data) {
Serial.println("Received the set GPIO RPC method");
int pin = data["pin"];
bool enabled = data["enabled"];
if (pin < COUNT_OF(leds_PinControl)) {
  Serial.print("Setting LED ");
  Serial.print(pin);
  Serial.print(" to state ");
  Serial.println(leds_Status[pin]);
  leds_Status[pin] = 1 - leds_Status[pin];
  digitalWrite(leds_PinControl[pin], leds_Status[pin]);
}
return RPC_Response(data["pin"], (bool)data["enabled"]);
}
//############################################################
// RPC handlers
//===========================================================
RPC_Callback callbacks[] = {
{ "setValue", processDelayChange },
{ "getValue", processGetDelay },
{ "setGpioStatus", processSetGpioState },
};
```

16. ปรับปรุง Arduino Library

- ThingsBoard Arduino SDK Ver 0.2.0
- ~~PubSubClient by Nick~~ O'Leary Ver 2.7.0
- ArduinoHttpClient libraries. Ver 0.3.2
- ArduinoJSON library Ver 6.9.1

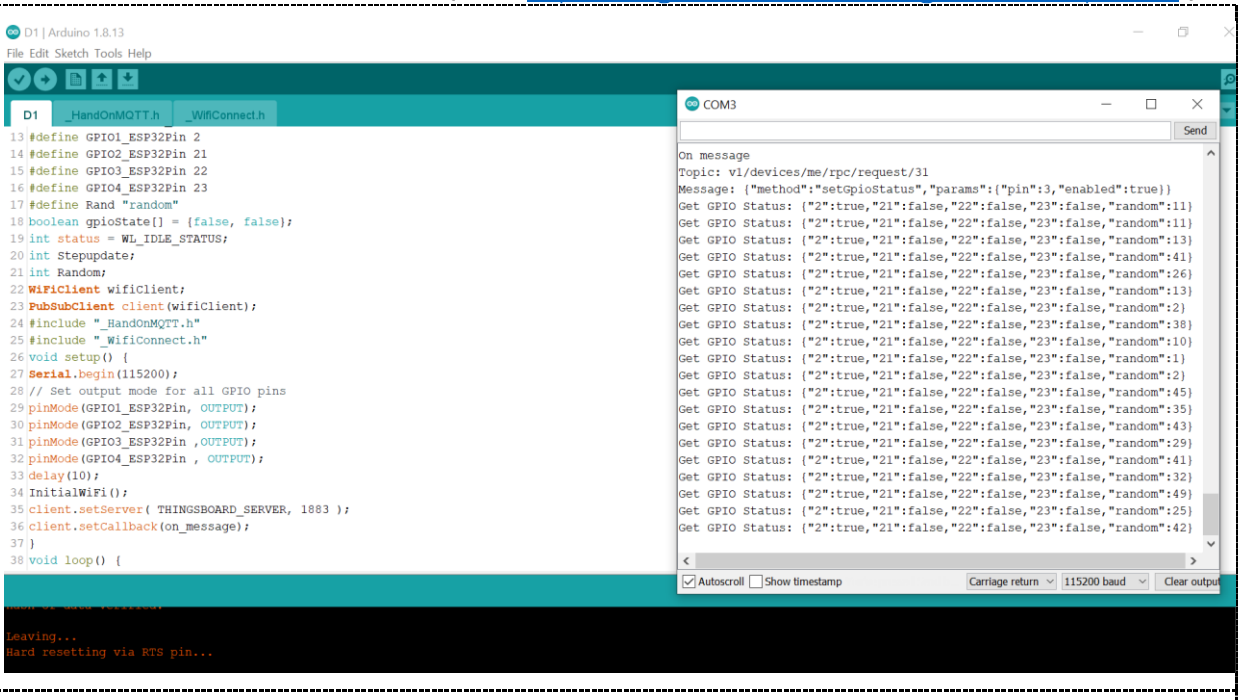





นางสาวกนิษฐา ผ่องกุศล 30/4/65

Quiz_103 – ThingsBoard Data Monitor and control with MQTT Protocol

- Mission 3/4: ให้ใช้ MQTT กับ ThingsBoard
  - ปรับปรุงเพื่อให้ทำงานควบคุมการ On/Off - 4 LED
  - เพิ่มเติม คือ ทดสอบส่งข้อมูล 1 ค่าแบบสุ่มระหว่าง 00 – 50 ไปแสดงที่ Dashboard ด้วย ได้หรือไม่ ทำอย่างไรบ้างให้อธิบาย {Read https://thingsboard.io/docs/user-guide/device-profiles/ }




```
// https://thingsboard.io/docs/samples/esp8266/gpio/
// https://blog.thingsboard.io/2017/01/esp8266-gpio-control-over-mqtt-using.html
#include <WiFi.h>
#include <ArduinoJson.h> // by Benoit Blanchon >> Ver 5.8.0
#include <PubSubClient.h> // by Nick O'Leary. >> Ver 2.6 and Update PubSubClient.h
// replace #ifdef ESP8266
// to #if defined (ESP8266) || defined(ESP32)
#define WIFI_AP_NAME "BOOK"
#define WIFI_PASSWORD "book1017"
#define Device_Name "TPhappy"
#define Device_Token "J5f7MI5UGGLk3NXCssHS"
#define THINGSBOARD_SERVER "thingsboard.cloud"
#define GPIO1_ESP32Pin 2
#define GPIO2_ESP32Pin 21
#define GPIO3_ESP32Pin 22
```

```
#define GPIO4_ESP32Pin 23
#define Rand "random"
boolean gpioState[] = {false, false};
int status = WL_IDLE_STATUS;
int Stepupdate;
int Random;
WiFiClient wifiClient;
PubSubClient client(wifiClient);
#include "_HandOnMQTT.h"
#include "_WifiConnect.h"
void setup() {
Serial.begin(115200);
// Set output mode for all GPIO pins
pinMode(GPIO1_ESP32Pin, OUTPUT);
pinMode(GPIO2_ESP32Pin, OUTPUT);
pinMode(GPIO3_ESP32Pin ,OUTPUT);
pinMode(GPIO4_ESP32Pin , OUTPUT);
delay(10);
InitialWiFi();
client.setServer( THINGSBOARD_SERVER, 1883 );
client.setCallback(on_message);
}
void loop() {
delay(20);
Stepupdate += 20;
if(Stepupdate > 5000){
Random = random(00 , 50);
client.publish("v1/devices/me/telemetry", get_gpio_status().c_str());
Stepupdate = 0;
}
if ( !client.connected() ) {
reconnect();
}
```

```
client.loop();
}
```

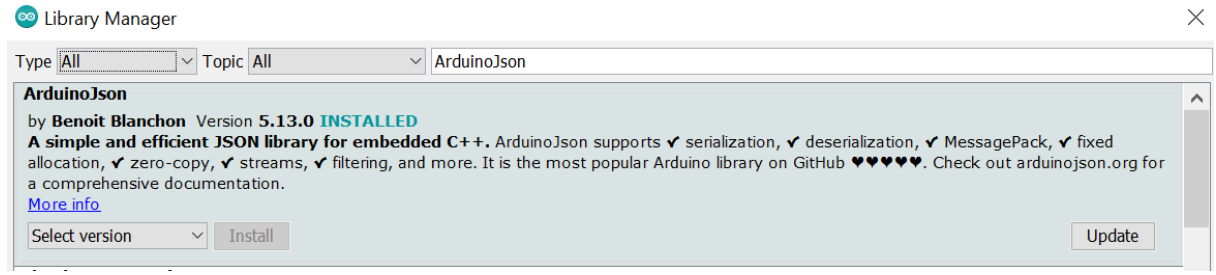
```
// _HandOnMQTT.h
String get_gpio_status() {
// Prepare gpios JSON payload string
StaticJsonBuffer<200> jsonBuffer;
JsonObject & data = jsonBuffer.createObject();
data[String(GPIO1_ESP32Pin)] = gpioState[0];
data[String(GPIO2_ESP32Pin)] = gpioState[1];
data[String(GPIO3_ESP32Pin)] = gpioState[2];
data[String(GPIO4_ESP32Pin)] = gpioState[3];
data[Rand] = Random;
char payload[256];
data.printTo(payload, sizeof(payload));
String strPayload = String(payload);
Serial.print("Get GPIO Status: ");
Serial.println(strPayload);
return strPayload;
}
//=======================================================
//=======================================================
void set_gpio_status(int pin, boolean enabled) {
if (pin == GPIO1_ESP32Pin) {
gpioState[0] = 1 - gpioState[0];
digitalWrite(GPIO1_ESP32Pin, gpioState[0]);
}
if (pin == GPIO2_ESP32Pin) {
gpioState[1] = 1 - gpioState[1];
digitalWrite(GPIO2_ESP32Pin, gpioState[1]);
}
if (pin == GPIO3_ESP32Pin) {
gpioState[2] = 1 - gpioState[2];
```

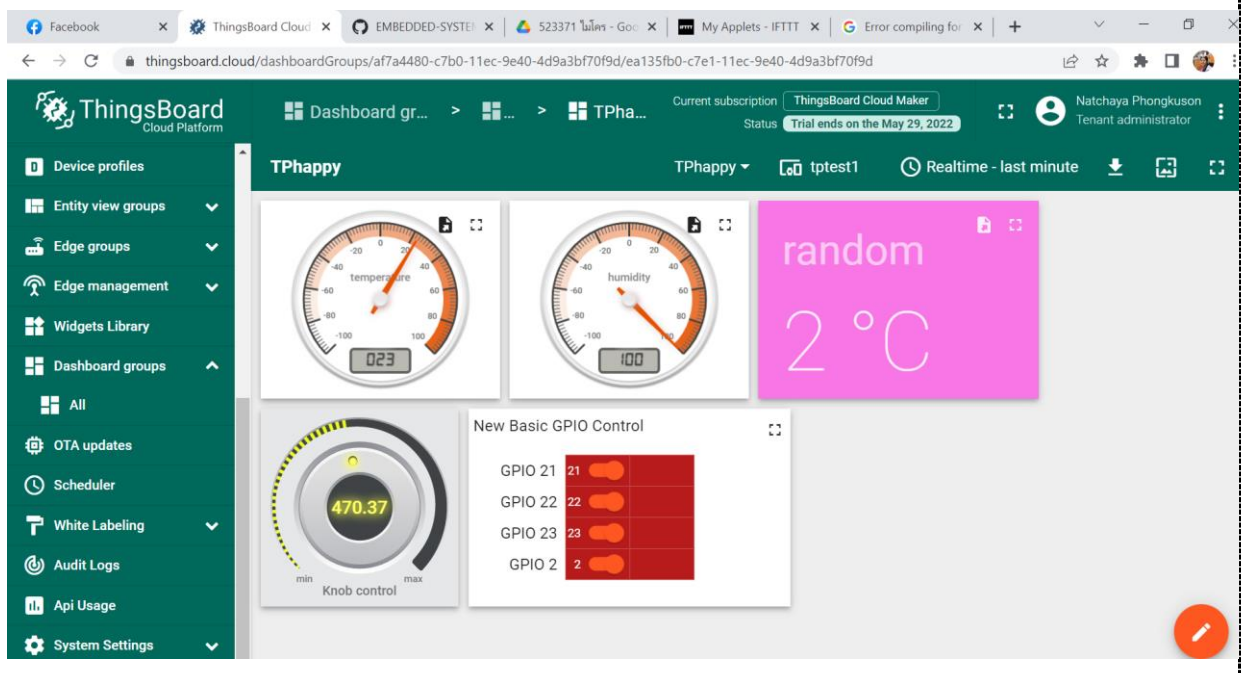
```
digitalWrite(GPIO3_ESP32Pin, gpioState[2]);
}
if (pin == GPIO4_ESP32Pin) {
gpioState[3] = 1 - gpioState[3];
digitalWrite(GPIO4_ESP32Pin, gpioState[3]);
}
}
//=====================================================
//=====================================================
// The callback for when a PUBLISH message is received from the server.
void on_message(const char* topic, byte* payload, unsigned int length) {
Serial.println("\nOn 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 Reply with GPIO status
if (methodName.equals("getGpioStatus")) {
String responseTopic = String(topic);
responseTopic.replace("request", "response");
client.publish(responseTopic.c_str(), get_gpio_status().c_str());
}
// If Update GPIO status and reply
```

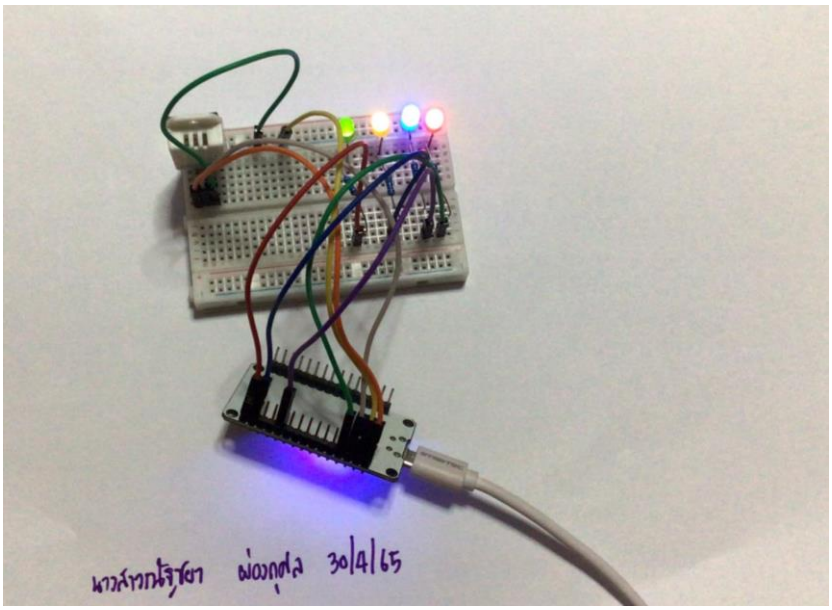
```
if (methodName.equals("setGpioStatus")) {
set_gpio_status(data["params"]["pin"], data["params"]["enabled"]);
String responseTopic = String(topic);
responseTopic.replace("request", "response");
client.publish(responseTopic.c_str(), get_gpio_status().c_str());
client.publish("v1/devices/me/attributes",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

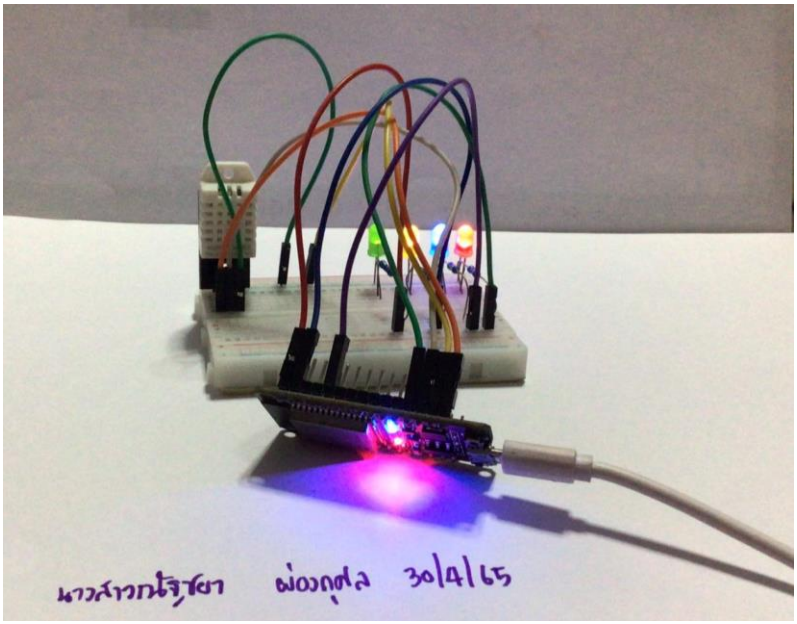
```
// _WifiConnect.h
void InitialWiFi() {
Serial.println("Connecting to AP ...");
WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("Connected to AP");
}
//=========================================================
//=========================================================
void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
status = WiFi.status();
if ( status != WL_CONNECTED) {
InitialWiFi();
}
Serial.print("Connecting to ThingsBoard node ...");
// Attempt to connect (clientId, username, password)
if ( client.connect(Device_Name, Device_Token, NULL) ) {
Serial.println( "[DONE]" );
// Subscribing to receive RPC requests
```

```
client.subscribe("v1/devices/me/rpc/request/+");
// Sending current GPIO status
Serial.println("Sending current GPIO status ...");
client.publish("v1/devices/me/attributes", get_gpio_status().c_str());
} else {
Serial.print( "[FAILED] [ rc = " );
Serial.print( client.state() );
Serial.println( " : retrying in 5 seconds]" );
delay( 5000 ); // Wait 5 seconds before retrying
} } }
```









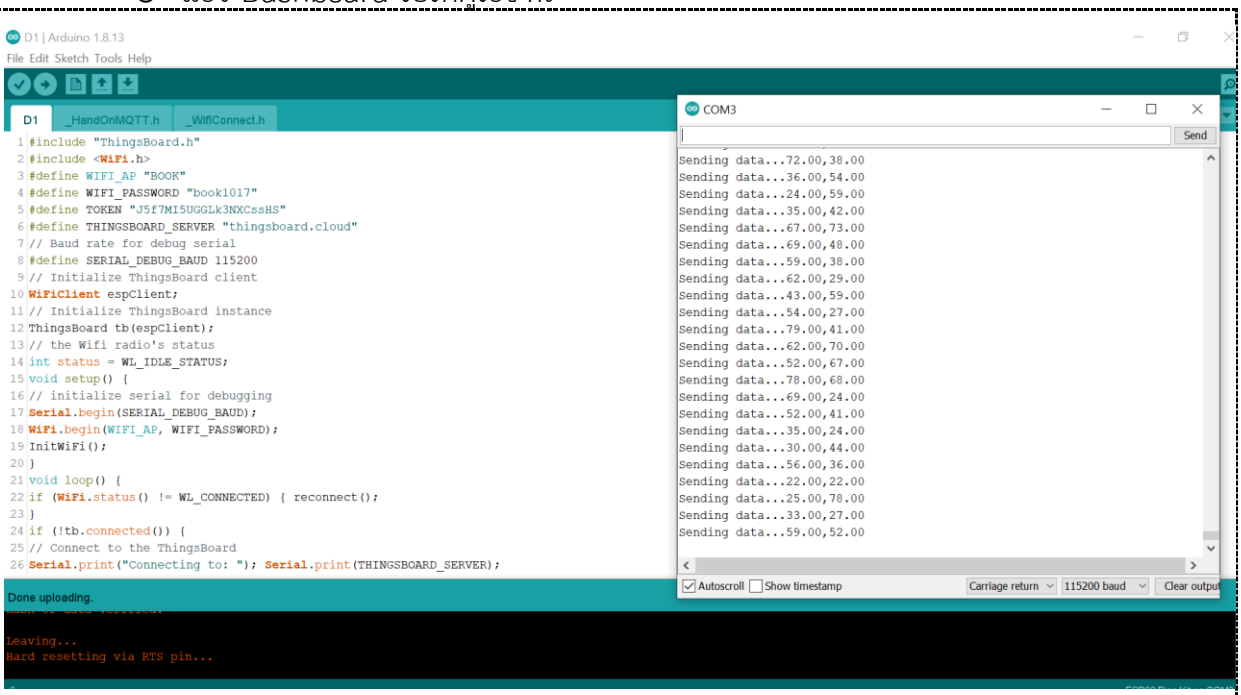
เพิ่มเติม คือ ทดสอบส่งข้อมูล 1 ค่าแบบสุ่มระหว่าง 00 – 50 ไปแสดงที่ Dashboard ด้วย ได้หรือไม่ ทำอย่างไรบ้างให้อธิบาย

สามารถทำได้ โดย

1. ข้อมูลที่เข้ามาที่ dashboard จะเป็นข้อมูลที่เราส่งค่ามาจาก Arduino ไปที่ Device
2. เมื่อได้ค่า random ที่ส่งมา ก็เลือกรูปแบบที่จะแสดงคือ Card โดยค่าที่แสดงก็จะออกมาตามข้อมูลสุ่มนั่นเอง
3. ข้อมูลที่ส่งมาจะไม่มี องศาเซลเซียส เนื่องจากเราเลือกให้แสดงผ่าน Card จึงสามารถใส่องศาเซลเซียสเข้าไปเพิ่มได้

Quiz_104 – Web Control 4 LED and Monitor Humid/Temperature

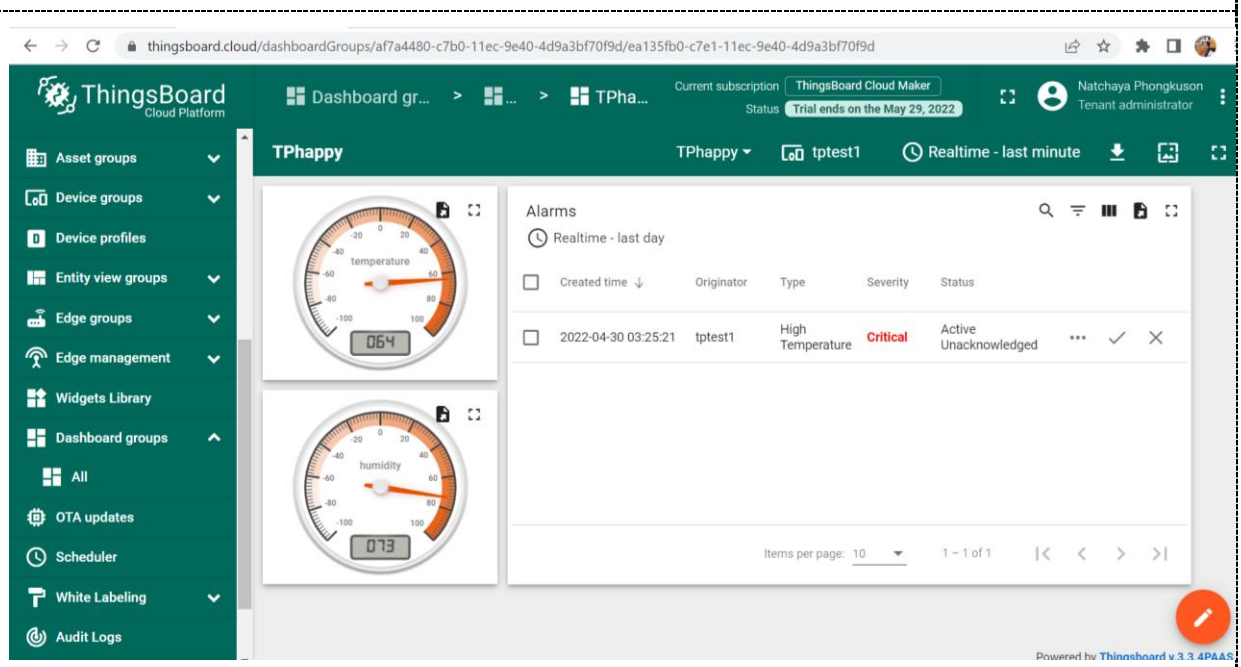
- Mission 4/4: การตรวจสอบและควบคุม อุณหภูมิ-ความชื้น ของโรงเรือนเลี้ยงไก่
  - ให้ใช้ ESP32 ส่งข้อมูลแบบสุ่มสองจำนวน คือ
    - Tempp_A สุ่มระหว่าง 20-40
    - Hudmid_A สุ่มระหว่าง 60-80
  - ข้อมูลทั้งสองค่าจะนำมาแสดงที่ Dashboard
  - สร้าง Alarm โดย หาก Tempp_A > 35 หรือ Hudmid_A > 70 ให้ Alarm
  - ศึกษาการตั้ง Alarm - https://thingsboard.io/docs/user-guide/alarms/
  - กำหนดรอบการตรวจสอบทุกๆ 20 วินาที
  - แชร์ Dashboard ไปให้ผู้ใช้งาน

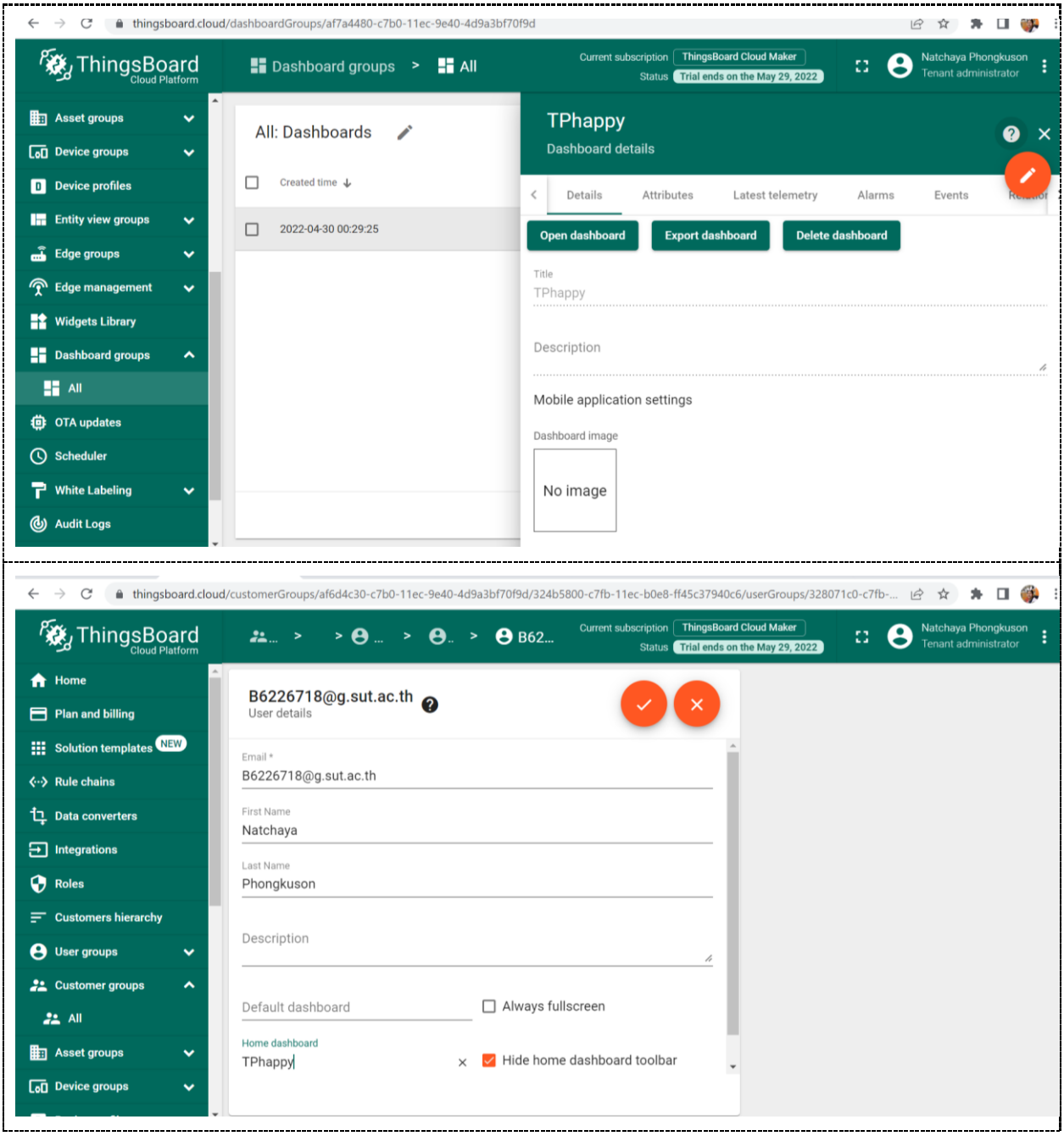


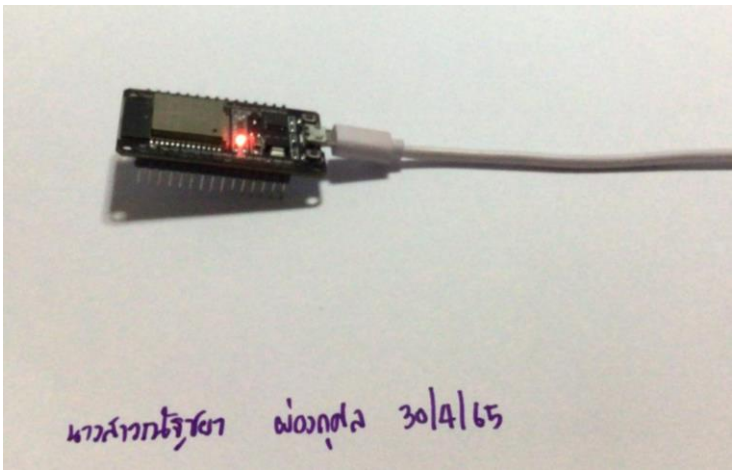

```
#include "ThingsBoard.h"
#include <WiFi.h>
#define WIFI_AP "BOOK"
#define WIFI_PASSWORD "book1017"
#define TOKEN "J5f7MI5UGGLk3NXCssHS"
#define THINGSBOARD_SERVER "thingsboard.cloud"
// Baud rate for debug serial
#define SERIAL_DEBUG_BAUD 115200
// Initialize ThingsBoard client
WiFiClient espClient;
```

```
// Initialize ThingsBoard instance
ThingsBoard tb(espClient);
// the Wifi radio's status
int status = WL_IDLE_STATUS;
void setup() {
// initialize serial for debugging
Serial.begin(SERIAL_DEBUG_BAUD);
WiFi.begin(WIFI_AP, WIFI_PASSWORD);
InitWiFi();
}
void loop() {
if (WiFi.status() != WL_CONNECTED) { reconnect();
}
if (!tb.connected()) {
// Connect to the ThingsBoard
Serial.print("Connecting to: "); Serial.print(THINGSBOARD_SERVER);
Serial.print(" with token "); Serial.println(TOKEN);
if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
Serial.println("Failed to connect"); return;
}
}
Serial.print("Sending data...");
// Uploads new telemetry to ThingsBoard using MQTT.
// See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
// for more details
float xTempp = random(20, 80);
float xHdmid = random(20, 80);
Serial.print(xTempp, 2);
Serial.print(","); Serial.print(xHdmid, 2); Serial.println();
tb.sendTelemetryFloat("temperature", xTempp);
tb.sendTelemetryFloat("humidity", xHdmid);
tb.loop(); delay(5000);
}
```

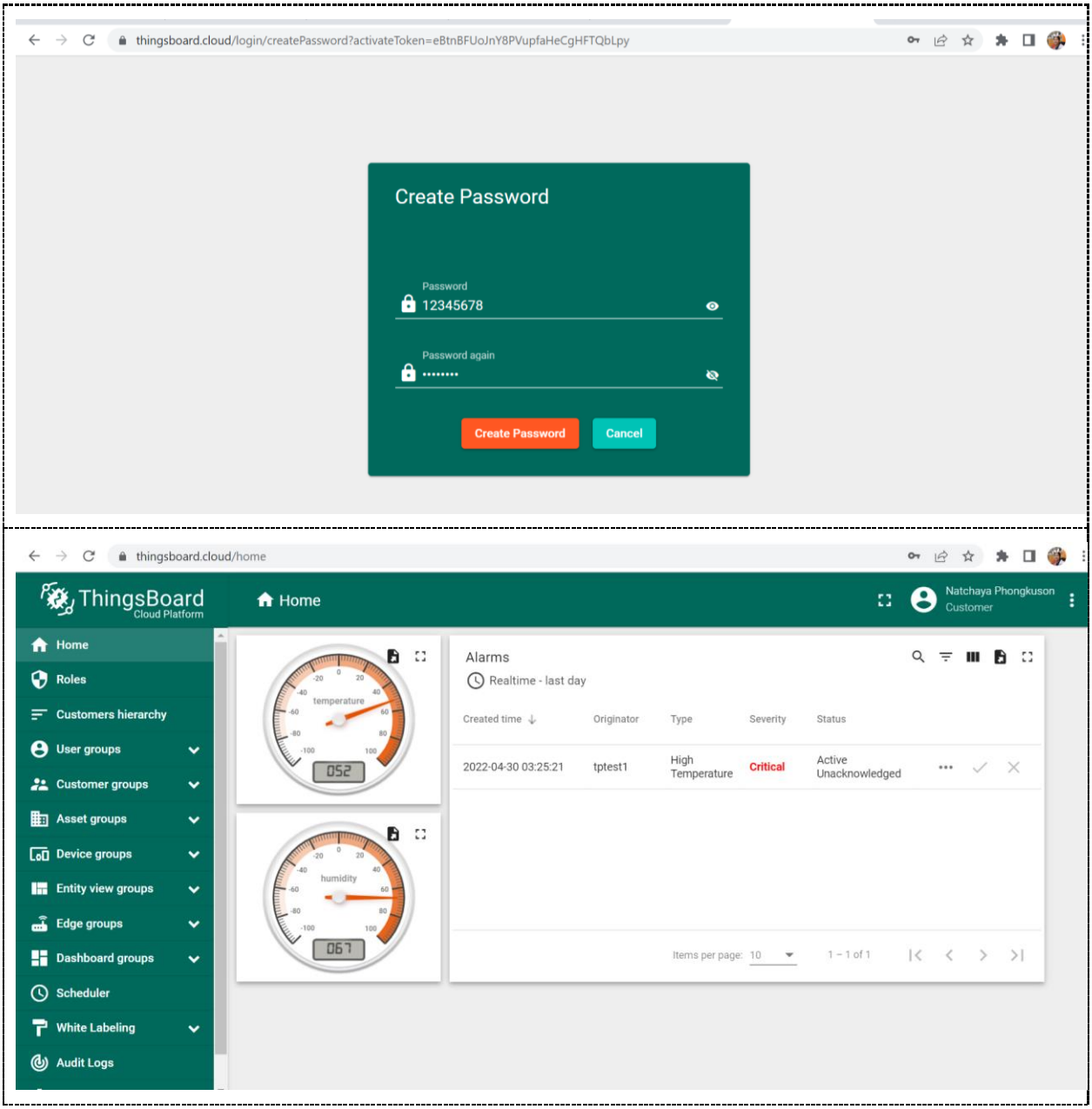
```
void InitWiFi(){
Serial.println("Connecting to AP ...");
// attempt to connect to WiFi network
WiFi.begin(WIFI_AP, WIFI_PASSWORD);
while (WiFi.status() != WL_CONNECTED) { delay(500);
Serial.print(".");
}
Serial.println("Connected to AP");
}
void reconnect() {
// Loop until we're reconnected
status = WiFi.status();
if ( status != WL_CONNECTED) {
WiFi.begin(WIFI_AP, WIFI_PASSWORD);
while (WiFi.status() != WL_CONNECTED) { delay(500);
Serial.print(".");
}
Serial.println("Connected to AP");
}
}
```




thingsboard.cloud/dashboardGroups/af7a4480-c7b0-11ec-9e40-4d9a3bf70f9d
ThingsBoard Cloud Platform
Dashboard groups > All
Current subscription ThingsBoard Cloud Maker
Status Trial ends on the May 29, 2022
Natchaya Phongkuson
Tenant administrator
Asset groups
Device groups
Device profiles
Entity view groups
Edge groups
Edge management
Widgets Library
Dashboard groups
All
OTA updates
Scheduler
White Labeling
Audit Logs
All: Dashboards
Created time
2022-04-30 00:29:25
TPhappy
Dashboard details
Details
Attributes
Latest telemetry
Alarms
Events
Open dashboard
Export dashboard
Delete dashboard
Title
TPhappy
Description
Mobile application settings
Dashboard image
No image
thingsboard.cloud/customerGroups/af6d4c30-c7b0-11ec-9e40-4d9a3bf70f9d/324b5800-c7fb-11ec-b0e8-ff45c37940c6/userGroups/328071c0-c7fb-...
ThingsBoard Cloud Platform
B62...
Current subscription ThingsBoard Cloud Maker
Status Trial ends on the May 29, 2022
Natchaya Phongkuson
Tenant administrator
Home
Plan and billing
Solution templates
NEW
Rule chains
Data converters
Integrations
Roles
Customers hierarchy
User groups
Customer groups
All
Asset groups
Device groups
B6226718@g.sut.ac.th
User details
Email *
B6226718@g.sut.ac.th
First Name
Natchaya
Last Name
Phongkuson
Description
Default dashboard
Always fullscreen
Home dashboard
TPhappy
Hide home dashboard toolbar



https://thingsboard.cloud/api/noauth/activate?activateToken=eBtnBFUoJnY8PVupfaHeCgHFTQbLpy


thingsboard.cloud/login/createPassword?activateToken=eBtnBFUoJnY8PVupfaHeCgHFTQbLpy
Create Password
Password
12345678
Password again
Create Password
Cancel
thingsboard.cloud/home
ThingsBoard
Cloud Platform
Home
Natchaya Phongkuson
Customer
Home
Roles
Customers hierarchy
User groups
Customer groups
Asset groups
Device groups
Entity view groups
Edge groups
Dashboard groups
Scheduler
White Labeling
Audit Logs
temperature
052
humidity
067
Alarms
Realtime - last day
Created time
Originator
Type
Severity
Status
2022-04-30 03:25:21
tptest1
High Temperature
Critical
Active Unacknowledged
Items per page: 10
1 – 1 of 1