การใช้งาน ThingsBoard IoTs Platform เพื่อสร้างและจัดการระบบอัจริยะ

ThingsBoard IoTs Platform for smart system

ชื่อ-สกุล : นายธนพล กาศักดิ์

6/6 -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_101 – ThingsBoard Data Monitor

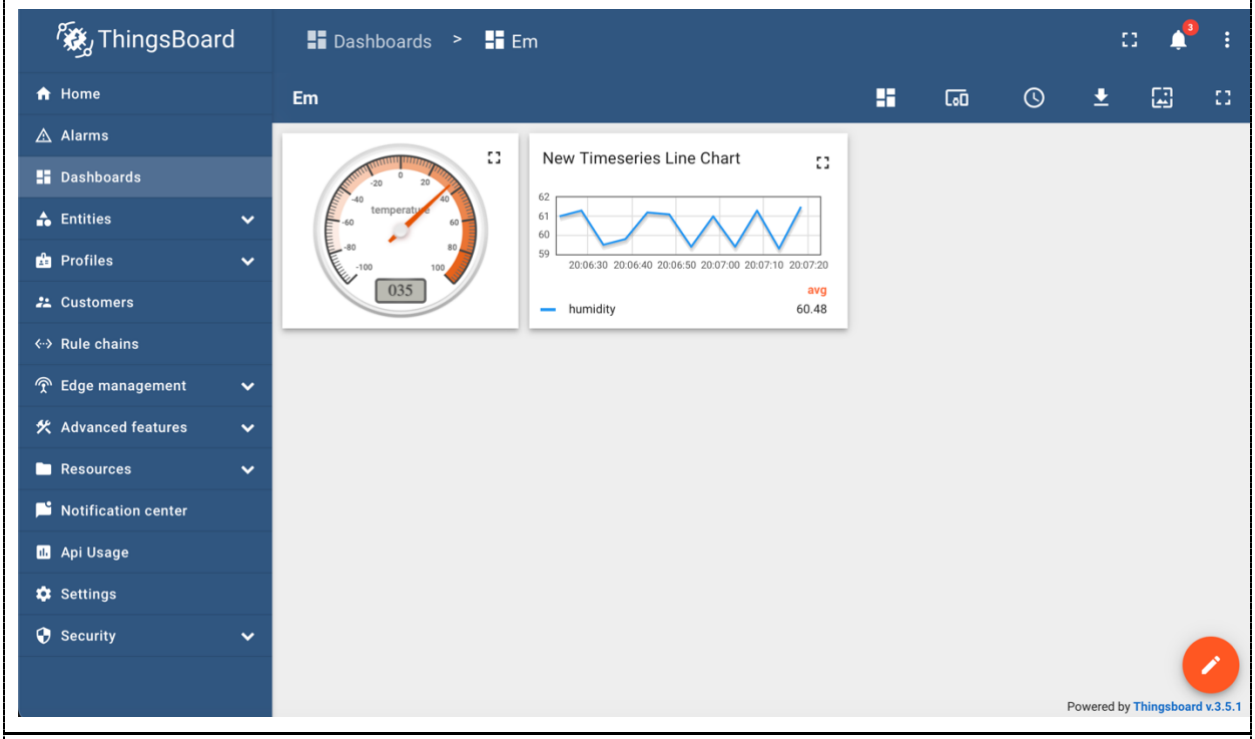
- Mission - 1/4: ให้ส่งข้อมูลค่า Humidity และ Temperatures จากเซ็นเซอร์ DHT-22 ไปยัง Dashboard

```
#include "ThingsBoard.h"
#include "DHTesp.h"
#include <WiFi.h>
#define WIFI_AP "JAEWHON_2.4G" //Your Wifi
#define WIFI_PASSWORD "4144284312" //Your Wifi password
#define TOKEN "bp8fF8oofRkDx2EXU5eL"
#define THINGSBOARD_SERVER "demo.thingsboard.io"
#define Pin_DHT22 15
#define SERIAL_DEBUG_BAUD 115200

WiFiClient espClient;
DHTesp dht;

ThingsBoard tb(espClient);

int status = WL_IDLE_STATUS;

void setup() {
// initialize serial for debugging
  Serial.begin(SERIAL_DEBUG_BAUD);
  dht.setup(Pin_DHT22, DHTesp::DHT22);
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  InitWiFi();
}
void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }
```

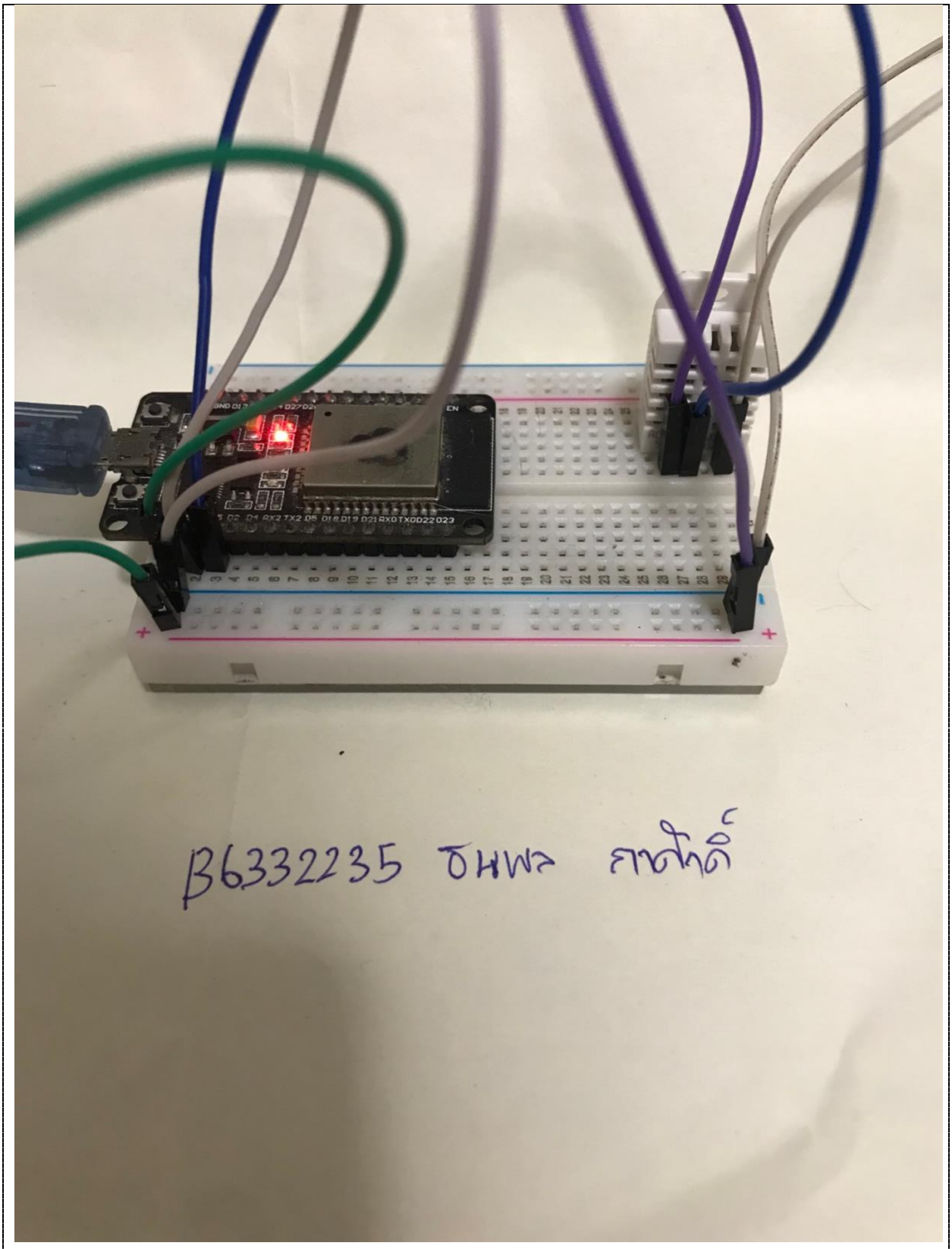
```
  if (!tb.connected()) {
    // Connect to the ThingsBoard
    Serial.print("Connecting to: ");
    Serial.print(THINGSBOARD_SERVER);
    Serial.print(" with token ");
    Serial.println(TOKEN);
    if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
      Serial.println("Failed to connect");
      return;
    }
  }
  Serial.print("Sending data...");float xTempp = dht.getTemperature();
  float xHdmid = dht.getHumidity();
  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"); }
  }
```

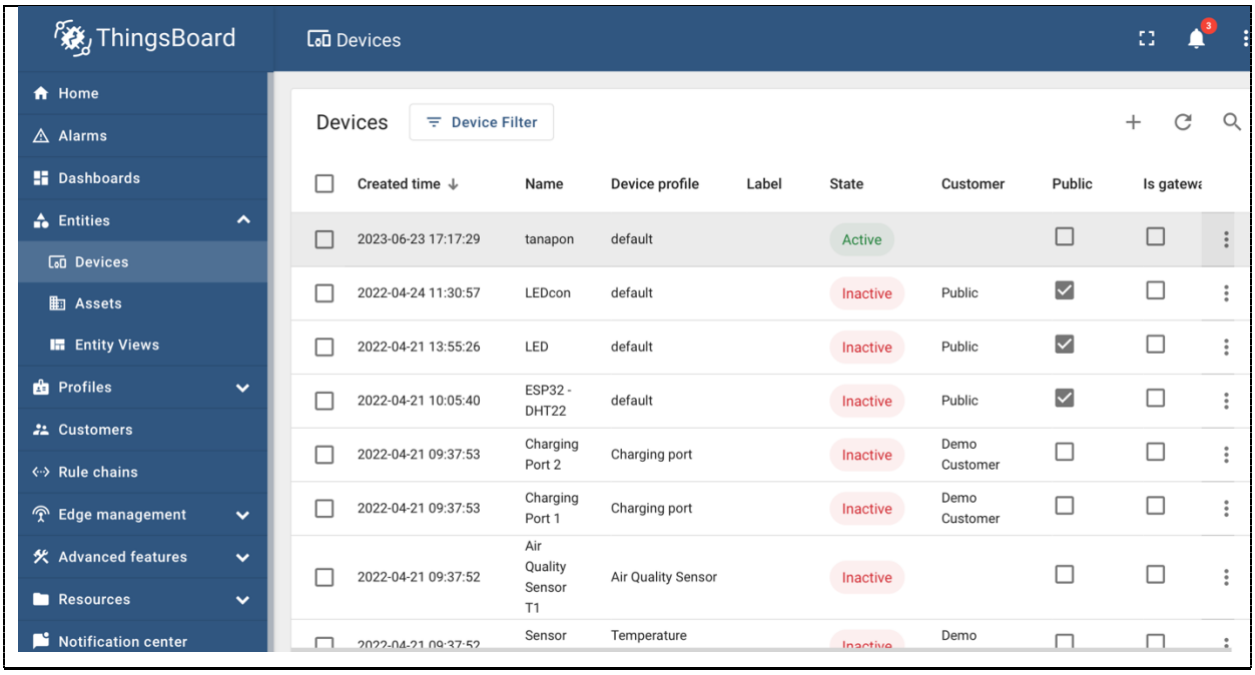
Capture Dashboard



รูปการทดสอบ 1



รูปการทดสอบ 2


ThingsBoard
Devices
Home
Alarms
Dashboards
Entities
Devices
Assets
Entity Views
Profiles
Customers
Rule chains
Edge management
Advanced features
Resources
Notification center
Devices
Device Filter
Created time
Name
Device profile
Label
State
Customer
Public
Is gatewa
2023-06-23 17:17:29 tanapon default Active
2022-04-24 11:30:57 LEDcon default Inactive Public
2022-04-21 13:55:26 LED default Inactive Public
2022-04-21 10:05:40 ESP32 - DHT22 default Inactive Public
2022-04-21 09:37:53 Charging Port 2 Charging port Inactive Demo Customer
2022-04-21 09:37:53 Charging Port 1 Charging port Inactive Demo Customer
2022-04-21 09:37:52 Air Quality Sensor T1 Air Quality Sensor Inactive
Sensor Temperature Demo

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

โปรแกรมที่ใช้ทดสอบ

```
#define COUNT_OF(x) ((sizeof(x)/sizeof(0[x])) / ((size_t)(!(sizeof(x) % sizeof(0[x])))))
#include <WiFi.h>
#include <ThingsBoard.h>
#include "DHTesp.h"
#define WIFI_AP_NAME "JAEWHON_2.4G" //Your Wifi
#define WIFI_PASSWORD "4144284312" //Your Wifi password
#define TOKEN "bp8fF8oofRkDx2EXU5eL"
#define THINGSBOARD_SERVER "thingsboard.cloud"
#define pinLEDBlink 2
#define Pin_DHT22 15
WiFiClient espClient;
DHTesp dht;
ThingsBoard tb(espClient);
int status = WL_IDLE_STATUS;
uint8_t leds_PinControl[] = {19, 21, 22, 23};
int leds_Ststus[] = { 0, 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();
  // Pinconfig
  dht.setup(Pin_DHT22, DHTesp::DHT22);
  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.
    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 temperature = dht.getTemperature();
    float humidity = dht.getHumidity();
    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_Ststus[i]);
      Serial.print(leds_Ststus[i]);
    }
    Serial.println();
    Count_sendDataDelay = 0;
  }

  // Step 6/6 - Process messages
  tb.loop();
}
```

Wifi.h

```
// _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());
  }
}
```

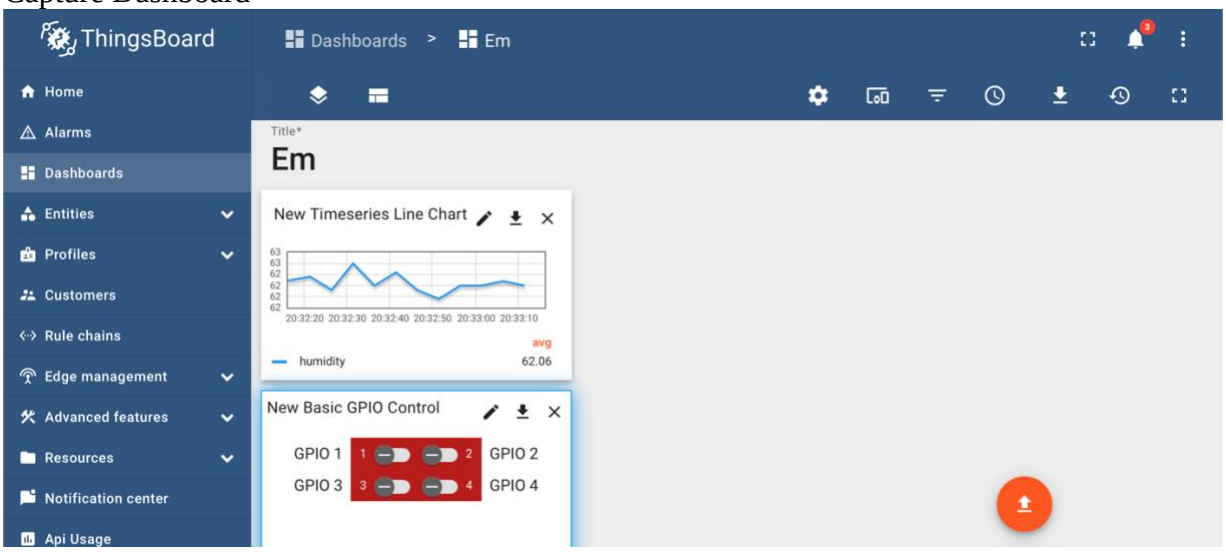
RPC.h

```
// _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_Ststus[pin]);
    leds_Ststus[pin] = 1 - leds_Ststus[pin];
    digitalWrite(leds_PinControl[pin], leds_Ststus[pin]);
  }
  return RPC_Response(data["pin"], (bool)data["enabled"]);
}
//#################################################################
// RPC handlers
//=================================================================
RPC_Callback callbacks[] = {
 { "setValue", processDelayChange },
 { "getValue", processGetDelay },
```

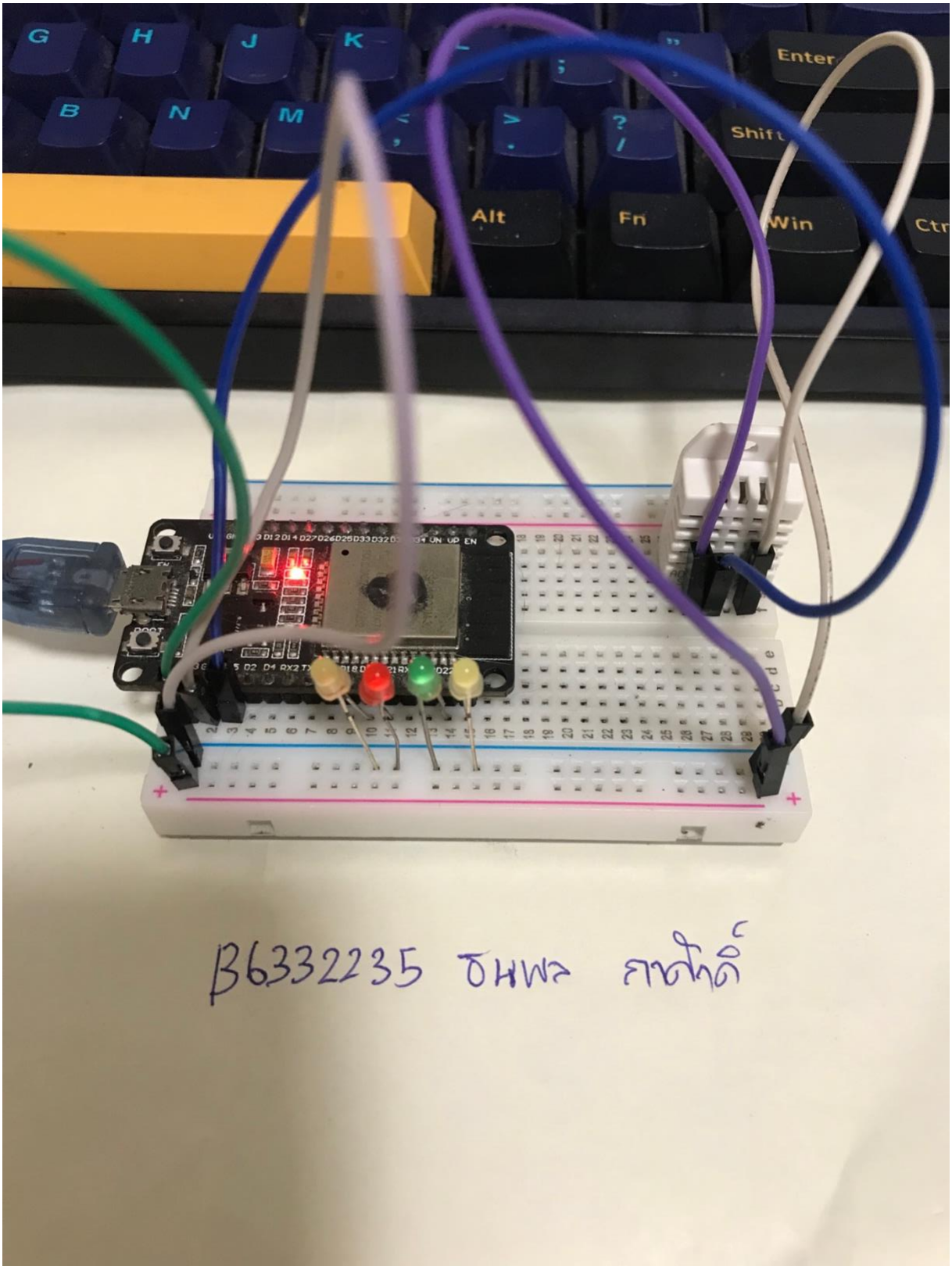
```
{ "setGpioStatus", processSetGpioState },
};
```

Capture Dashboard



รูปถ่ายหน้า Web Broswer

รูปการทดสอบ 1



รูปการทดสอบ 2

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/ }

โปรแกรมที่ใช้ทดสอบ

```
//Arduino Code
// 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#define WIFI_AP_NAME "TRUE-BFFM 2G"
#define WIFI_PASSWORD "7211722142622"
#define Device_Name "new Device 2"
#define Device_Token "HM35zge611h6BIMn78Ia"
#define thingsboardServer "thingsboard.cloud"#define GPIO1_ESP32Pin 19
#define GPIO2_ESP32Pin 21
#define GPIO3_ESP32Pin 22
#define GPIO4_ESP32Pin 23
#define varTemp "temperature"
float temperature;
boolean gpioState[] = {false, false, false, false};
long lastMsgTime = 0;
int status = WL_IDLE_STATUS;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( thingsboardServer, 1883 );
  client.setCallback(on_message);
}void loop() {
  if ( !client.connected() ) {
    reconnect();
  }
  sendTemperature();
  client.loop();
}void sendTemperature() {
  if (millis() - lastMsgTime > 5000)
  { lastMsgTime = millis();
    temperature = random(0, 50);StaticJsonBuffer<200> jsonBuffer;
    JsonObject & data = jsonBuffer.createObject();
    data[String(varTemp)] = temperature;
    char payload[256];
    data.printTo(payload, sizeof(payload));
    String strPayload = String(payload);
    Serial.print("Get GPIO Status: ");
    Serial.println(strPayload);
    client.publish("v1/devices/me/telemetry", strPayload.c_str());
```

```
  }
}
```

MQTT.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];
  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());
 }
}
```

Wifi.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
  }
 }
}
```

Capture Dashboard

รูปถ่ายหน้า Web Broswer
รูปการทดสอบ 1
รูปการทดสอบ 2

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

โปรแกรมที่ใช้ทดสอบ

```
#include "ThingsBoard.h"
#include <WiFi.h>#define WIFI_AP "TRUE-BFFM 2G"
#define WIFI_PASSWORD "7211722142622"
#define TOKEN "z5dfxG3pUlssi9shOjzy"
#define THINGSBOARD_SERVER "thingsboard.cloud"WiFiClient espClient;
ThingsBoard tb(espClient);
int status = WL_IDLE_STATUS;void setup() {
  Serial.begin(115200);
  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.println("\nSending data...");
 float Tempp_A = random(2000,4000)/100.0;
 float Humid_A = random(6000,8000)/100.0;
 Serial.print("Tempp = " + String(Tempp_A, 2) + "'C");
 Serial.print(",");
 Serial.println("Humid = " + String(Humid_A, 2) + "%");
 tb.sendTelemetryFloat("Tempp_A", Tempp_A);
 tb.sendTelemetryFloat("Humid_A", Humid_A);
 tb.loop();
 delay(5000);
}void InitWiFi() {
 Serial.println("Connecting to AP ...");
 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");
 }
}
```

Capture Dashboard

รูปถ่ายหน้า Web Broswer

รูปการทดสอบ 1

รูปการทดสอบ 2