

Quiz_101 – ThingsBoard Data Monitor

- The screenshot shows the Arduino IDE interface during a code upload. The main editor window displays C++ code for a ThingsBoard IoT application. The code includes headers for ThingsBoard.h, WiFi.h, and DHTesp.h, and defines various constants like WIFI_AP, WIFI_PASSWORD, TOKEN, THINGSBOARD_SERVER, and pin numbers. It sets up a serial debug port and initializes a WiFi client and a DHT sensor. The setup() function initializes the serial port and connects the DHT sensor. The loop() function contains a comment indicating the connection of the DHT sensor to GPIO 15.

```
#include "ThingsBoard.h"
#include <WiFi.h>
#define WIFI_AP "BOOK"
#define WIFI_PASSWORD "bookl017"
#define TOKEN "J5f7M15UGGLkXNXCasRS"
#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() {
```

A secondary window titled 'COM3' displays the real-time output from the microcontroller. It consists of multiple lines of the text 'Sending data...32.40 , 99.90', indicating that the program is successfully sending temperature and humidity data to the ThingsBoard server via the serial interface.

At the bottom of the IDE, the status bar indicates 'Done uploading.' followed by a progress bar. Below this, it says 'Leaving...' and 'Hard resetting via RTS pin...', which are standard messages after a successful upload.

```
#include "ThingsBoard.h"

#include <WiFi.h>

#define WIFI_AP "BOOK"

#define WIFI_PASSWORD "book1017"

#define TOKEN "J5f7MI5UGGLk3NXCsHS"

#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...");

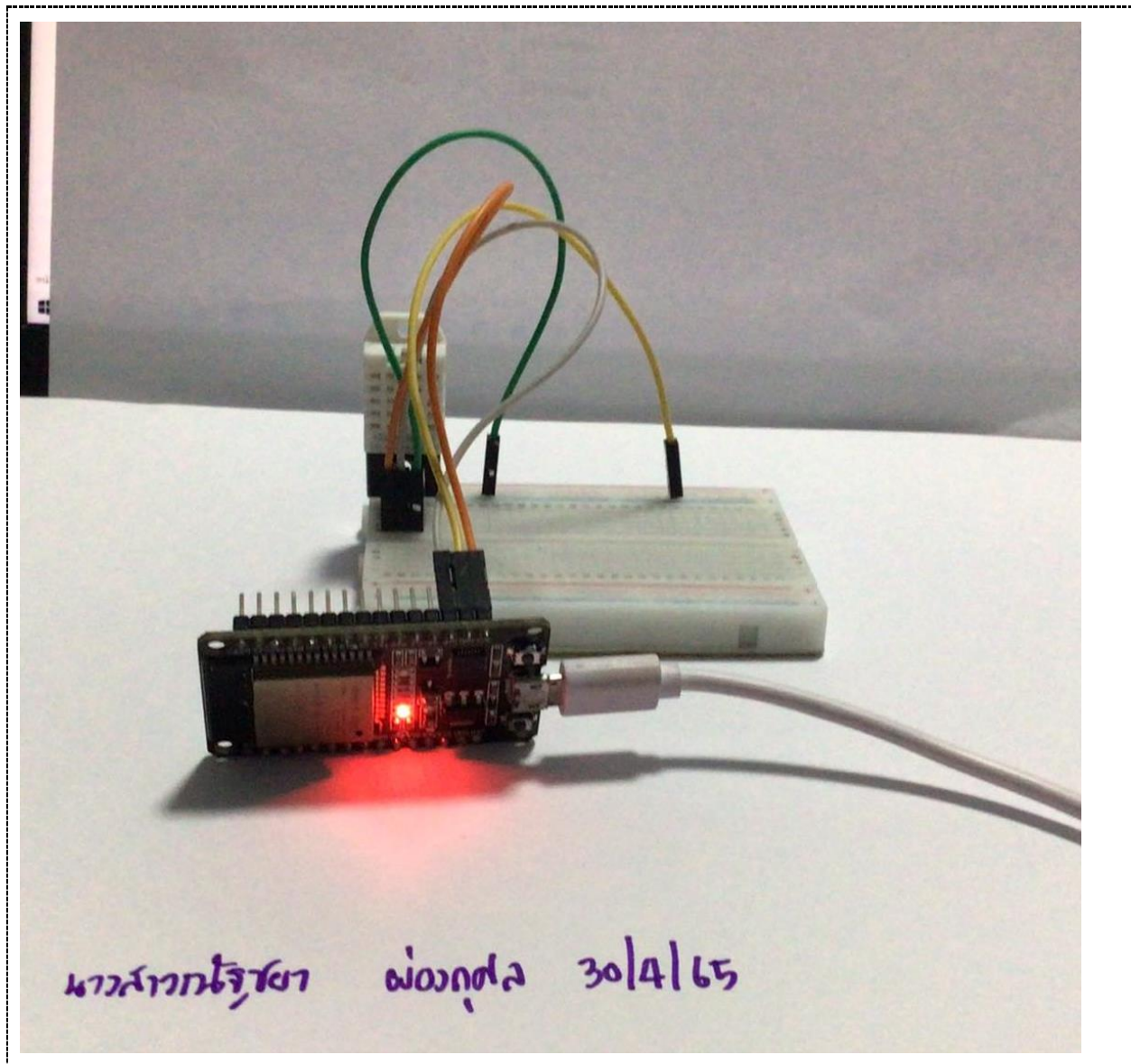
  // Uploads new telemetry to ThingsBoard using MQTT.
  // See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
  // for more details
  //tb.sendTelemetryInt("temperature", xTempp);
  //tb.sendTelemetryInt("humidity", xTempp);

```

```
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");  
}  
}
```

The top section is a screenshot of the ThingsBoard Cloud Platform dashboard. The interface includes a left sidebar with navigation options like Home, Plan and billing, Solution templates, Rule chains, Data converters, Integrations, Roles, Customers hierarchy, User groups, Customer groups, Asset groups, Device groups, and Device profiles. The main area displays two circular gauges: 'temperature' with a value of 032 and 'humidity' with a value of 100. The dashboard is titled 'TPhappy' and shows a subscription for 'ThingsBoard Cloud Maker' with a trial ending on May 29, 2022. The user is identified as Natchaya Phongkuson, a Tenant administrator.

The bottom section is a photograph of the physical hardware. It shows a breadboard with several colored wires (green, yellow, orange, grey) connected to a small electronic module. A red LED indicator on the module is illuminated. A grey cable is plugged into the module. Handwritten Thai text at the bottom of the photo reads 'การตั้งค่าบอร์ด 30/4/65'.



Quiz_102 – ThingsBoard Data Monitor and Control

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

The image shows the Arduino IDE interface with a C++ sketch for a ThingsBoard IoT project. The sketch includes headers for WiFi, ThingsBoard, and DHT sensors, and defines variables for pin control and status. The COM3 serial monitor shows a continuous stream of data being sent to the ThingsBoard cloud, including temperature, humidity, and LED status. The IDE interface includes the File menu, a toolbar, and a status bar at the bottom.

```

D1  _ConnectWifi.h  _ThingBoardRPC.h

1 #define COUNT_OF(X) ((sizeof(X)/sizeof(0[X])) / ((size_t)(!(sizeof(X) % sizeof(0[X])))))
2 #include <WiFi.h>
3 #include <ThingsBoard.h>
4 #include <Arduino.h>
5 #define WIFI_AP_NAME "BOOK"
6 #define WIFI_PASSWORD "book1017"
7 #define TOKEN "J5f7M15UGGLk3XNcCsHs"
8 #define THINGSBOARD_SERVER "thingsboard.cloud"
9 #define DHT22_Pin 15
10 #include "DHTesp.h"
11 DHTesp dht;
12 #define pinLEDBlink 2
13
14 WiFiClient espClient;
15 ThingsBoard tb(espClient);
16 int status = WL_IDLE_STATUS;
17 uint8_t leds_PinControl[] = {21, 22, 23};
18 int leds_Status[] = { 0, 0, 0 };
19 char StringEcho[] = "stsLED_1";
20 int loopDelay = 20; // Main loop delay(ms)
21 int sendDataDelay = 2000; // Period of Sending Temp/Humid.
22 int BlinkLEDDelay = 500; // Initial period of LED cycling.
23 int Count_BlinkLEDDelay = 0; // Time Counter Blink period
24 int Count_sendDataDelay = 0; // Time Counter Sending Temp/Humid
25 bool Subscribed_Status = false; // Subscribed_Status for the RPC messages.
26 int stsus_BlinkLED = 0; // LED number that is currently ON.
  
```

COM3

```

Sending data...T=23.20, H=99.90, LED=001
Sending data...T=23.20, H=99.90, LED=001
Sending data...T=23.20, H=99.90, LED=001
Sending data...T=23.20, H=99.90, LED=001
Sending data...T=23.20, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
Sending data...T=23.20, H=99.90, LED=001
Sending data...T=23.20, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
Sending data...T=23.20, H=99.90, LED=001
Sending data...T=23.20, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
Sending data...T=23.10, H=99.90, LED=001
  
```

☒ Autoscroll ☐ Show timestamp Carriage return 115200 baud Clear output

leaving...
Hard resetting via RTS pin...

```
#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 currentlty ON.
#include "_ThingBoardRPC.h"
#include "_ConnectWifi.h"

//=====================================================
void setup() {
    // Initialize serial for debugging
    Serial.begin(115200);
    WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
    WiFi_Init();
    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) {

```



```

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) {

```

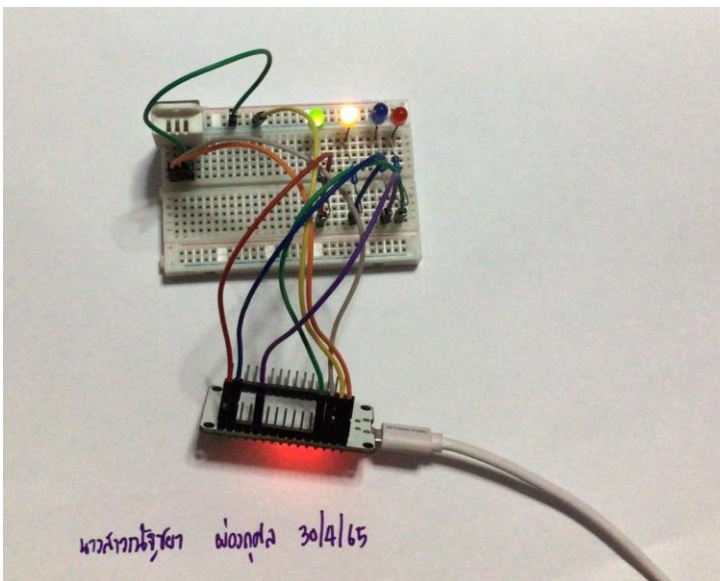
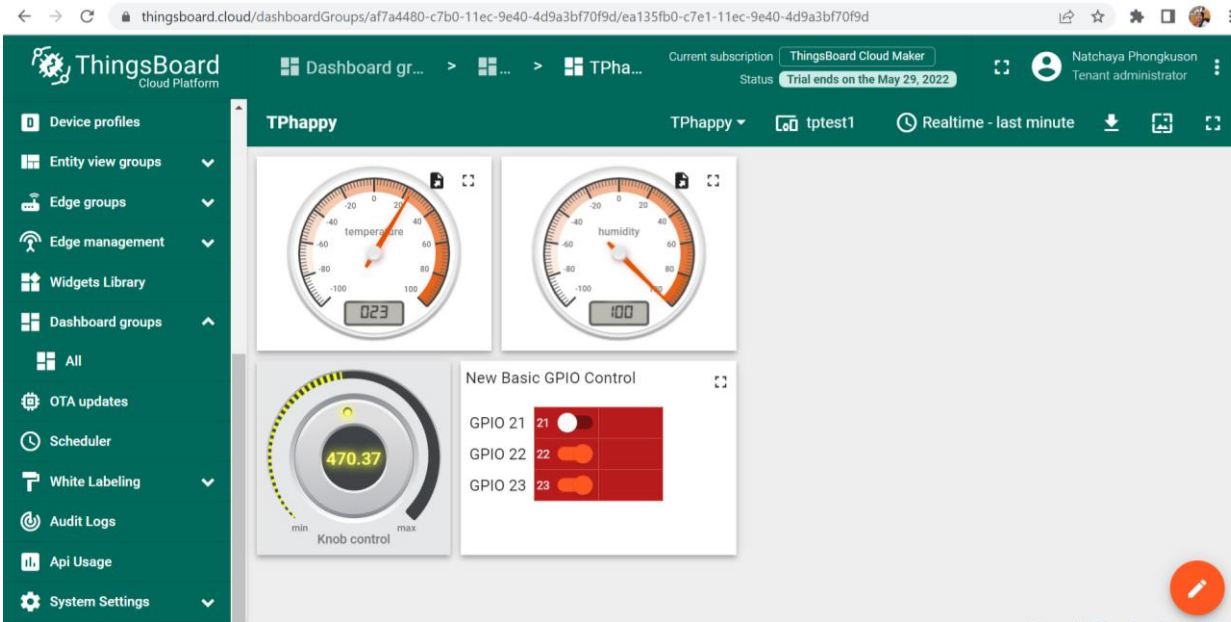
```

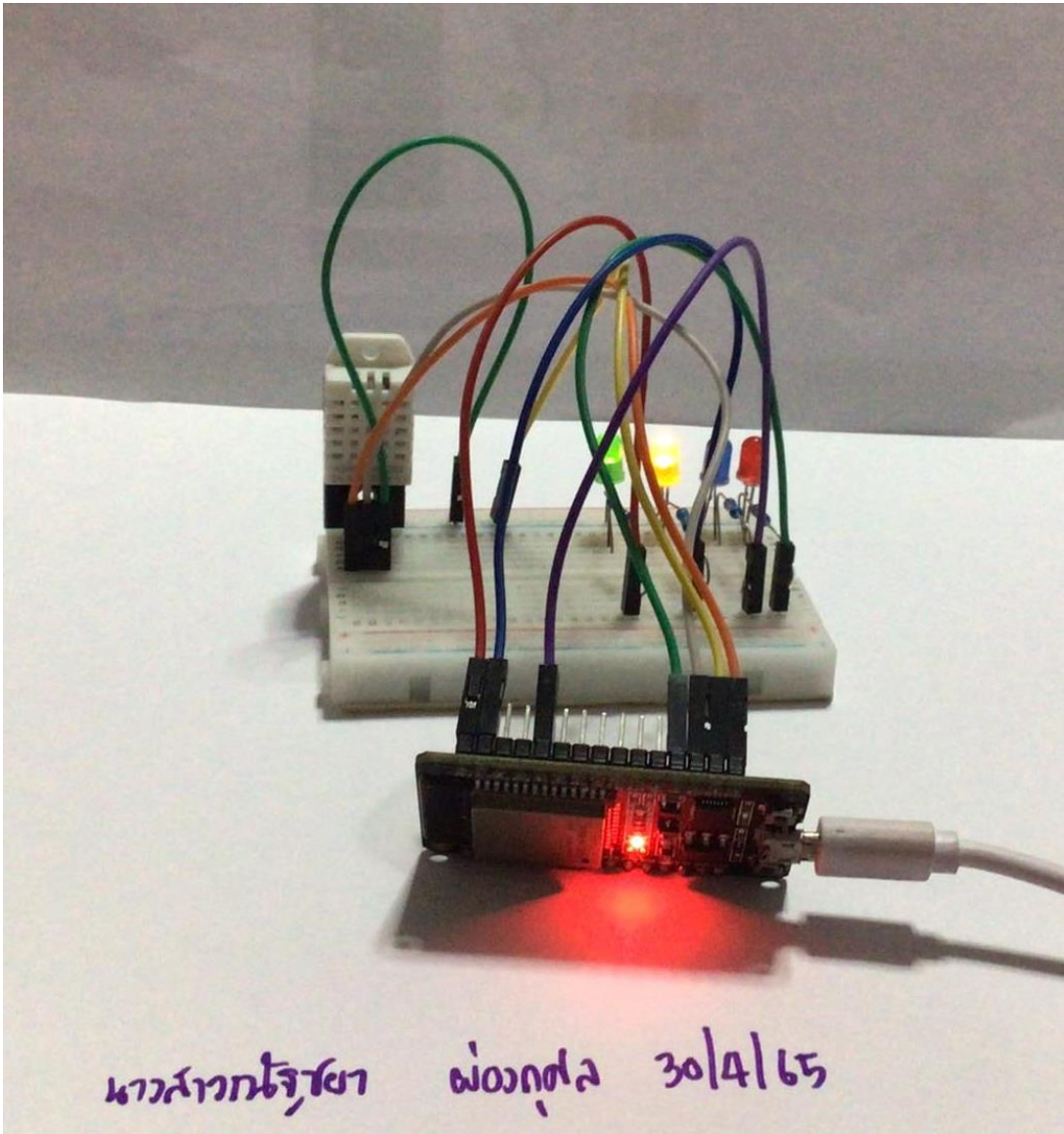
Serial.println("Received the get value method");
return RPC_Response(NULL, BlinkLEDDelay);
}
//#####
// Processes function for RPC call "setGpioStatus"
// RPC_Data is a JSON variant, that can be queried using operator[]
// See https://arduinojson.org/v5/api/jsonvariant/subscribe/ for more details
//=====
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

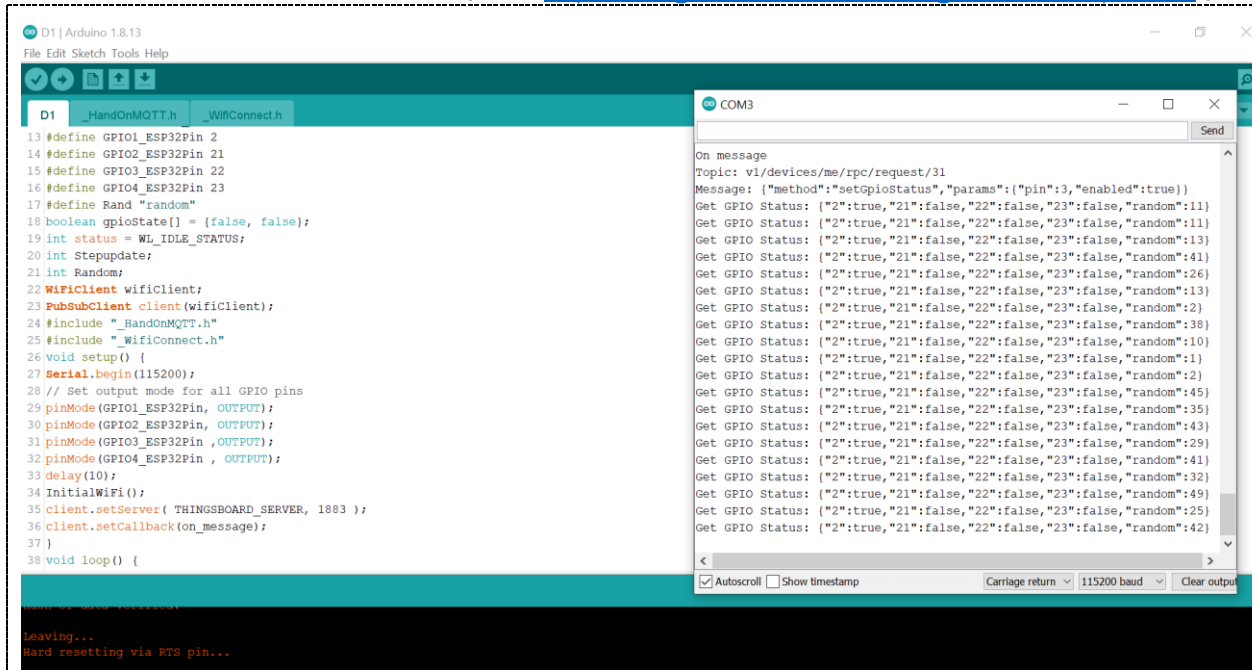
- ThingsBoard Arduino SDK Ver 0.2.0
- PubSubClient by Nick O'Leary Ver 2.7.0
- ArduinoHttpClient libraries. Ver 0.3.2
- ArduinoJSON library Ver 6.9.1





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
}}}

```

Library Manager

Type: Topic:

ArduinoJson
by **Benoit Blanchon** Version **5.13.0** **INSTALLED**

A simple and efficient JSON library for embedded C++. ArduinoJson supports ✓ serialization, ✓ deserialization, ✓ MessagePack, ✓ fixed allocation, ✓ zero-copy, ✓ streams, ✓ filtering, and more. It is the most popular Arduino library on GitHub ♥♥♥♥♥. Check out arduinojson.org for a comprehensive documentation.

[More info](#)

Select version:

ThingsBoard Cloud Platform

Dashboard gr... > TPha...

Current subscription: **ThingsBoard Cloud Maker**
Status: Trial ends on the May 29, 2022

User: Natchaya Phongkuson, Tenant administrator

TPhappy

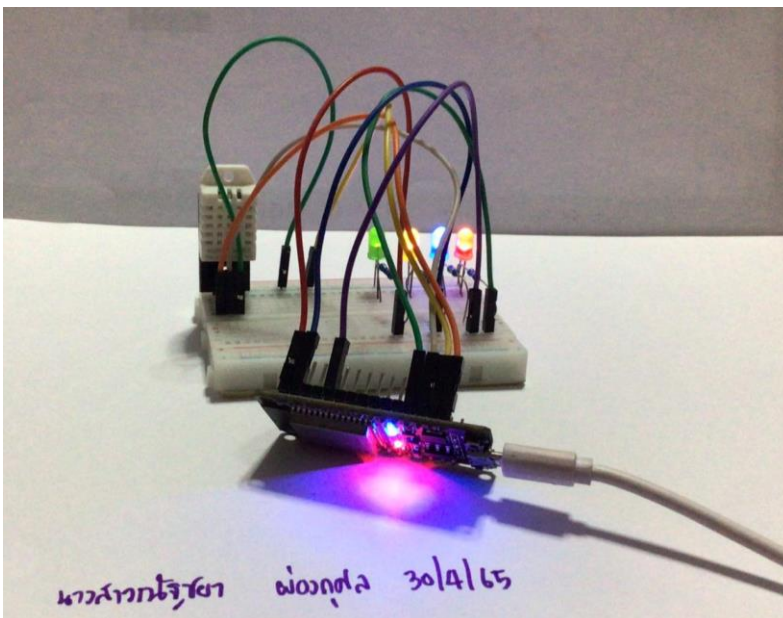
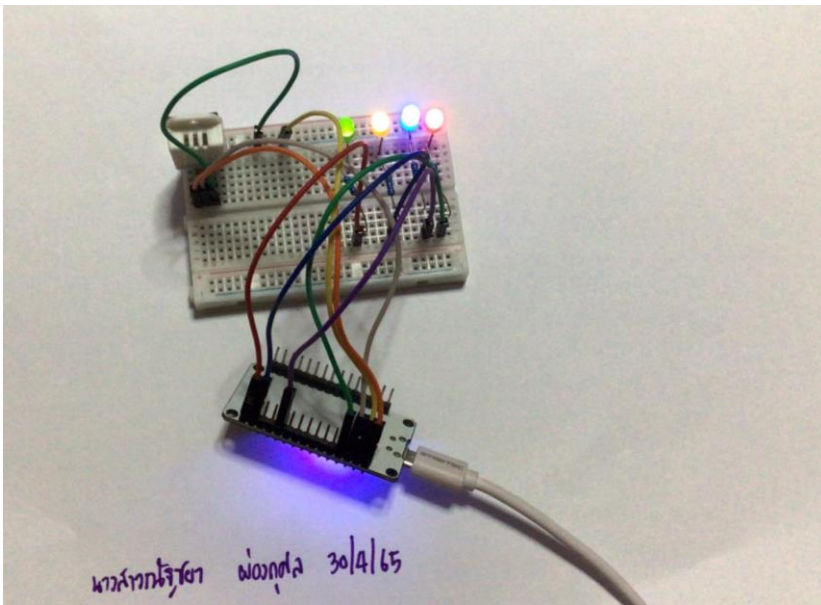
Widgets: TPhappy, tptest1, Realtime - last minute

random
2 °C

New Basic GPIO Control

GPIO 21	21	<input type="checkbox"/>	<input type="checkbox"/>
GPIO 22	22	<input type="checkbox"/>	<input type="checkbox"/>
GPIO 23	23	<input type="checkbox"/>	<input type="checkbox"/>
GPIO 2	2	<input type="checkbox"/>	<input type="checkbox"/>

Knob control
470.37

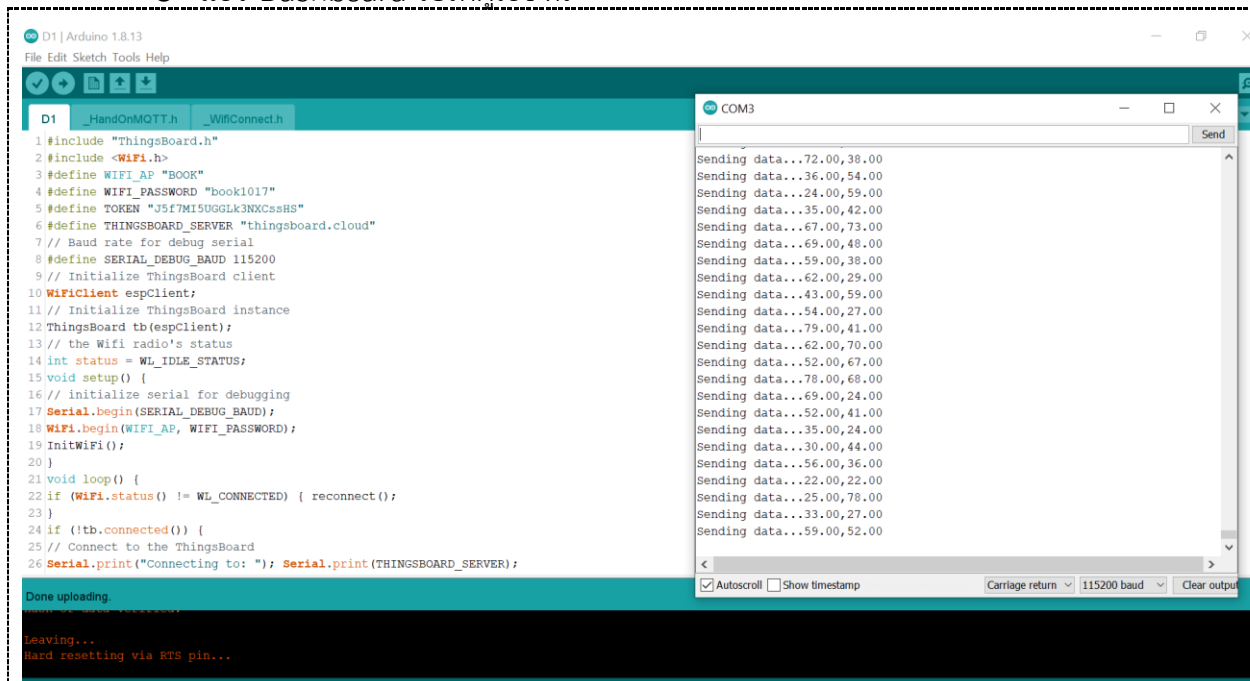


เพิ่มเติม คือ ทดสอบส่งข้อมูล 1 ค่าแบบสุ่มระหว่าง 00 – 50 ไปแสดงที่ Dashboard ด้วย ได้หรือไม่ ทำอย่างไรบ้างให้อธิบาย
สามารถทำได้ โดย

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

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
 - แชร Dashboard ไปให้ผู้ใช้งาน



```

#include "ThingsBoard.h"

#include <WiFi.h>

#define WIFI_AP "BOOK"

#define WIFI_PASSWORD "book1017"

#define TOKEN "J5f7M15UGGLk3NXCssHS"

#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");
  }
}

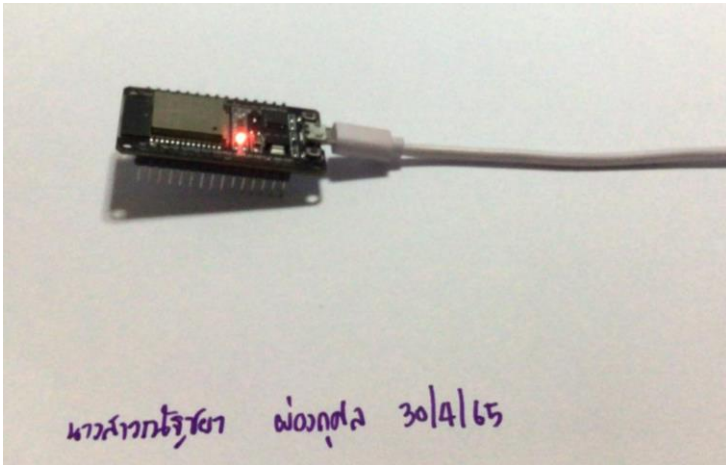
```

The screenshot displays the ThingsBoard Cloud Platform interface. The left sidebar contains navigation options: Asset groups, Device groups, Device profiles, Entity view groups, Edge groups, Edge management, Widgets Library, Dashboard groups, All, OTA updates, Scheduler, White Labeling, and Audit Logs. The main dashboard area is titled 'TPhappy' and features two circular gauges: 'temperature' showing 064 and 'humidity' showing 073. To the right, an 'Alarms' section shows a table of active alarms. The table has columns for Created time, Originator, Type, Severity, and Status. One alarm is listed: '2022-04-30 03:25:21' from 'tptest1' with a 'High Temperature' severity, marked as 'Critical' and 'Active Unacknowledged'. The bottom right corner indicates the platform is 'Powered by Thingsboard v3.3.4PAAS'.

Created time	Originator	Type	Severity	Status
2022-04-30 03:25:21	tptest1	High Temperature	Critical	Active Unacknowledged

The top screenshot displays the ThingsBoard Cloud Platform interface. The left sidebar contains navigation options: Asset groups, Device groups, Device profiles, Entity view groups, Edge groups, Edge management, Widgets Library, Dashboard groups, All, OTA updates, Scheduler, White Labeling, and Audit Logs. The main content area shows a list of dashboards under 'All: Dashboards'. A table lists dashboards with columns for 'Created time' and 'Dashboard'. One dashboard, 'TPhappy', is highlighted. To the right, the 'TPhappy' dashboard details are shown, including tabs for Details, Attributes, Latest telemetry, Alarms, Events, and Recent. The details section includes fields for Title (TPhappy), Description, Mobile application settings, and Dashboard image (No image). Buttons for 'Open dashboard', 'Export dashboard', and 'Delete dashboard' are visible.

The bottom screenshot shows the 'User details' form for a user named Natchaya Phongkuson. The left sidebar contains navigation options: Home, Plan and billing, Solution templates (NEW), Rule chains, Data converters, Integrations, Roles, Customers hierarchy, User groups, Customer groups, All, Asset groups, and Device groups. The main content area shows the 'User details' form for the user 'B6226718@g.sut.ac.th'. The form includes fields for Email, First Name (Natchaya), Last Name (Phongkuson), and Description. There are checkboxes for 'Always fullscreen' and 'Hide home dashboard toolbar' (checked). Buttons for 'Save' and 'Cancel' are visible.



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

The image displays two screenshots of the ThingsBoard Cloud Platform interface.

The top screenshot shows the 'Create Password' dialog box. It has a dark green header with the title 'Create Password'. Below the header, there are two input fields: 'Password' with the value '12345678' and 'Password again' with the value '*****'. Both fields have a lock icon on the left and an eye icon on the right. At the bottom of the dialog, there are two buttons: 'Create Password' (orange) and 'Cancel' (teal).

The bottom screenshot shows the 'Home' dashboard. The top navigation bar is dark green with the 'ThingsBoard Cloud Platform' logo on the left, a 'Home' button in the center, and a user profile 'Natchaya Phongkuson Customer' on the right. A left sidebar contains a menu with items: Home, Roles, Customers hierarchy, User groups, Customer groups, Asset groups, Device groups, Entity view groups, Edge groups, Dashboard groups, Scheduler, White Labeling, and Audit Logs. The main content area is divided into two columns. The left column contains two circular gauges: 'temperature' with a value of '052' and 'humidity' with a value of '067'. The right column contains an 'Alarms' section with a 'Realtime - last day' filter. Below the filter is a table with columns: 'Created time', 'Originator', 'Type', 'Severity', and 'Status'. The table contains one row of data: '2022-04-30 03:25:21', 'tptest1', 'High Temperature', 'Critical', and 'Active Unacknowledged'. At the bottom of the table, there is a pagination bar showing 'Items per page: 10' and '1 - 1 of 1'.