

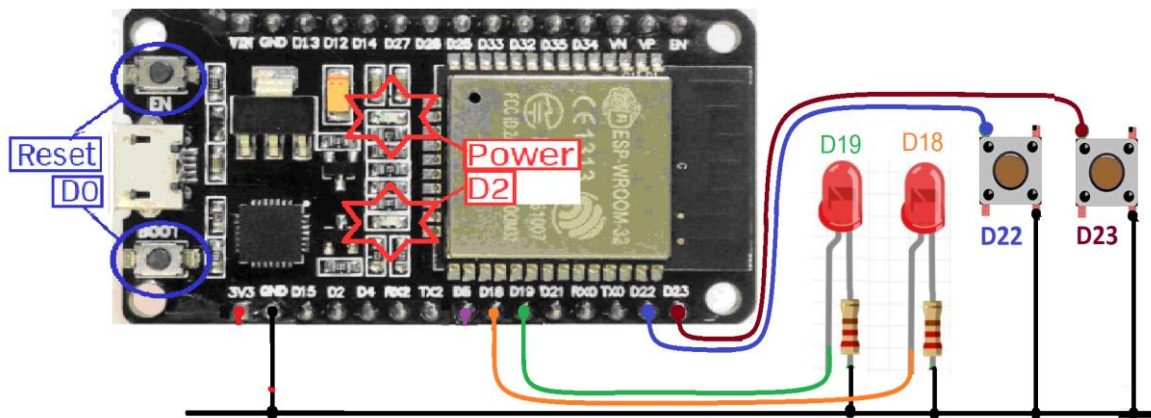
การใช้งาน ThingsBoard IoTs Platform เพื่อสร้างและจัดการระบบอัจฉริยะ ThingsBoard IoTs Platform for smart system

ชื่อ-สกุล : วราสิริ ลิ้มประเสริฐ B6214005

6/6 -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_101 – กดติด กดดับ 2 ชุด

- หากต้องการให้ใช้ 1 สวิตช์ ควบคุม 1 LED แบบกดติด-กดดับ จำนวน 2 วงจรจะต้องวงจรและเขียนโปรแกรมอย่างไร {SW-D22 -- LED-D19, SW-D23 -- LED-D18}



```
#define pushButton1 22
#define pushButton2 23
#define LEDPin1 18
#define LEDPin2 19
int buttonState1 = 0;
int buttonState2 = 0;

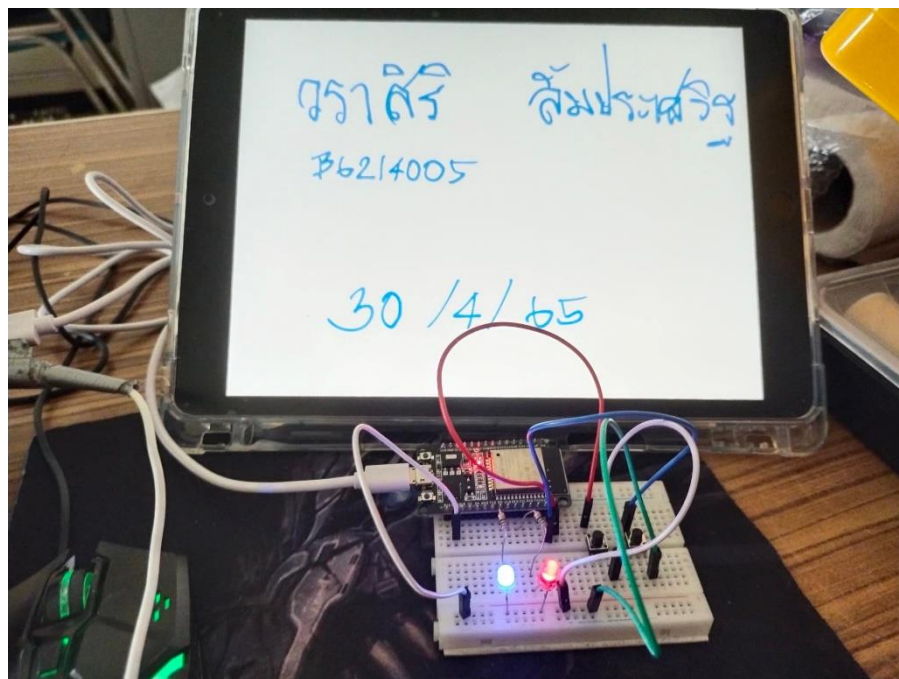
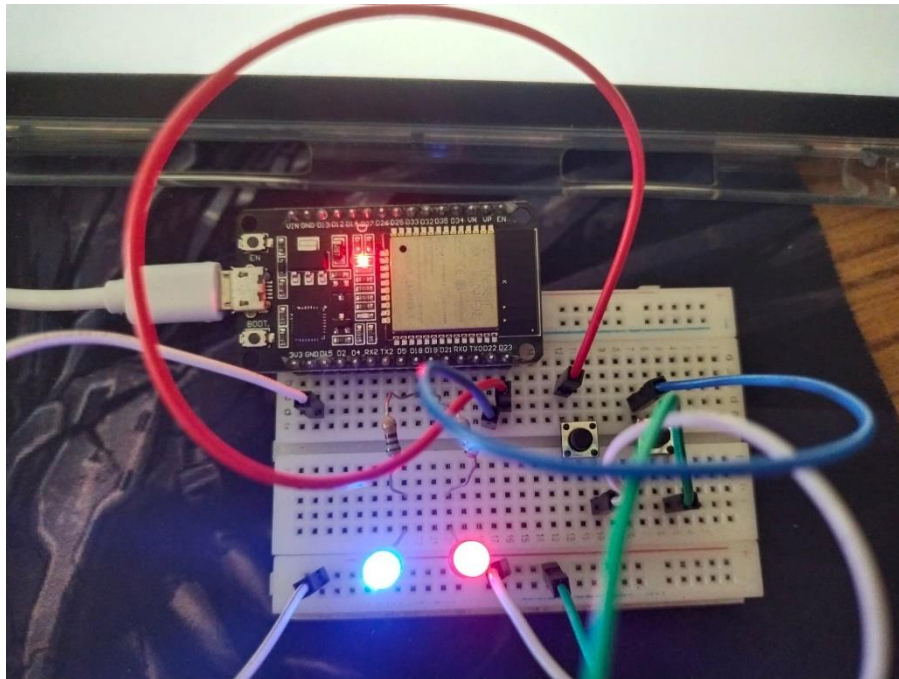
void setup() {
  Serial.begin(115200);
  pinMode(pushButton1, INPUT_PULLUP);
  pinMode(pushButton2, INPUT_PULLUP);
  pinMode(LEDPin1, OUTPUT);
  pinMode(LEDPin2, OUTPUT);
}

void loop() {
  if (digitalRead(pushButton1) == LOW) {
    delay(20);
    buttonState1 = 1 - buttonState1;
    digitalWrite(LEDPin1, buttonState1);
    while (digitalRead(pushButton1) == LOW);
    delay(20);
  }
  if (digitalRead(pushButton2) == LOW) {
```

```

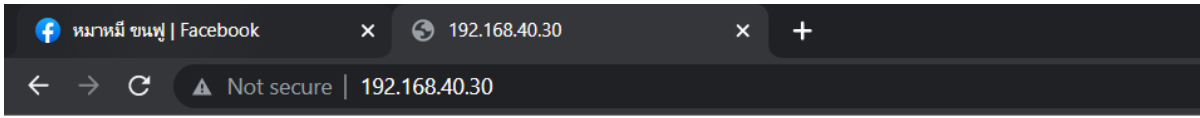
delay(20);
buttonState2 = 1 - buttonState2;
digitalWrite(LEDPin2, buttonState2);
while (digitalRead(pushButton2) == LOW);
delay(20);
}
}

```



Quiz_102 – Web Control 4 LED and Monitor Humid/Temperature

- เพิ่มเติมจาก Q202 อยากได้ปุ่มสำหรับคุมปิด-เปิด หลอดไฟ LED 4 ดวง
- อยากมีกด Link ไปที่หน้า FB ของตัวเอง
- https://www.colorhexa.com/008cba?fbclid=IwAR3dIZ_gRgDWmREmnzuknLbMxV3pOHY4YIPuLEz8-ZzTOX2VhWxcH2QjLGk



The screenshot shows a web browser with the address bar displaying '192.168.40.30'. The page title is 'The ESP-32 Update web page without refresh'. The interface features two rows of buttons: the top row has four red buttons labeled 'LED1 ON', 'LED2 ON', 'LED3 ON', and 'LED4 ON'; the bottom row has four blue buttons labeled 'LED1 OFF', 'LED2 OFF', 'LED3 OFF', and 'LED4 OFF'. Below the buttons, the text reads: 'State of [LED1, LED2, LED3, LED4] is >> ON, ON, ON, ON'. Further down, it shows 'DHT-22 sensor : Temp = 31.90 C, Humidity = 64.50 %'. At the bottom, it says 'By Varasiri Limprasert B6214005'.

Code

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include "DHTesp.h"
#include "index.h" //Our HTML webpage contents with javascripts
#define DHT_Pin 4

#define testLED1 18
#define testLED2 19
#define testLED3 22
#define testLED4 23

//SSID and Password of your WiFi router
const char* ssid = "V2036";
const char* password = "fnafchica";
WebServer server(80); //Server on port 80
DHTesp dht;
String ledState1 = "OFF";
String ledState2 = "OFF";
String ledState3 = "OFF";
```

```

String ledState4 = "OFF";
//=====
// This routine is executed when you open its IP in browser
//=====
void handleRoot() {
    String s = MAIN_page; //Read HTML contents
    server.send(200, "text/html", s); //Send web page
}
void handleADC() {
    float h = dht.getHumidity();
    float t = dht.getTemperature();
    String tmpValue = "Temp = ";
    tmpValue += String(t) + " C, Humidity = ";
    tmpValue += String(h) + " %";
    server.send(200, "text/plain", tmpValue); //Send value to client ajax request
}
void handleLED() {
    String t_state = server.arg("LEDstate"); //Refer xhttp.open("GET", "setLED?LEDstate="+led, true);
    Serial.println(t_state);
    if (t_state == "11") {
        digitalWrite(testLED1, HIGH); //Feedback parameter
        ledState1 = "ON";
    }
    if (t_state == "10") {
        digitalWrite(testLED1, LOW); //Feedback parameter
        ledState1 = "OFF";
    }
    if (t_state == "21") {
        digitalWrite(testLED2, HIGH); //Feedback parameter
        ledState2 = "ON";
    }
    if (t_state == "20") {
        digitalWrite(testLED2, LOW); //Feedback parameter
        ledState2 = "OFF";
    }
    if (t_state == "31") {
        digitalWrite(testLED3, HIGH); //Feedback parameter
        ledState3 = "ON";
    }
    if (t_state == "30") {
        digitalWrite(testLED3, LOW); //Feedback parameter
        ledState3 = "OFF";
    }
    if (t_state == "41") {
        digitalWrite(testLED4, HIGH); //Feedback parameter
        ledState4 = "ON";
    }
}

```

```

if (t_state == "40") {
    digitalWrite(testLED4, LOW); //Feedback parameter
    ledState4 = "OFF";
}
server.send(200, "text/plain", ledState1 + ", " + ledState2 + ", " + ledState3 + ", " + ledState4); //Send web page
}

void setup(void) {
    Serial.begin(115200);
    dht.setup(DHT_Pin, DHTesp::DHT22); // DHT_Pin D4, DHT22
    pinMode(testLED1, OUTPUT);
    pinMode(testLED2, OUTPUT);
    pinMode(testLED3, OUTPUT);
    pinMode(testLED4, OUTPUT);
    Serial.print("\n\nConnect to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500); Serial.print(".");
    }
    Serial.print("\nConnected "); Serial.println(ssid);
    Serial.print("IP address: "); Serial.println(WiFi.localIP());
    server.on("/", handleRoot);
    server.on("/setLED", handleLED);
    server.on("/readADC", handleADC);
    server.begin();
    Serial.println("HTTP server started");
}

void loop(void) {
    server.handleClient(); //Handle client requests
}

```

Index.h

```

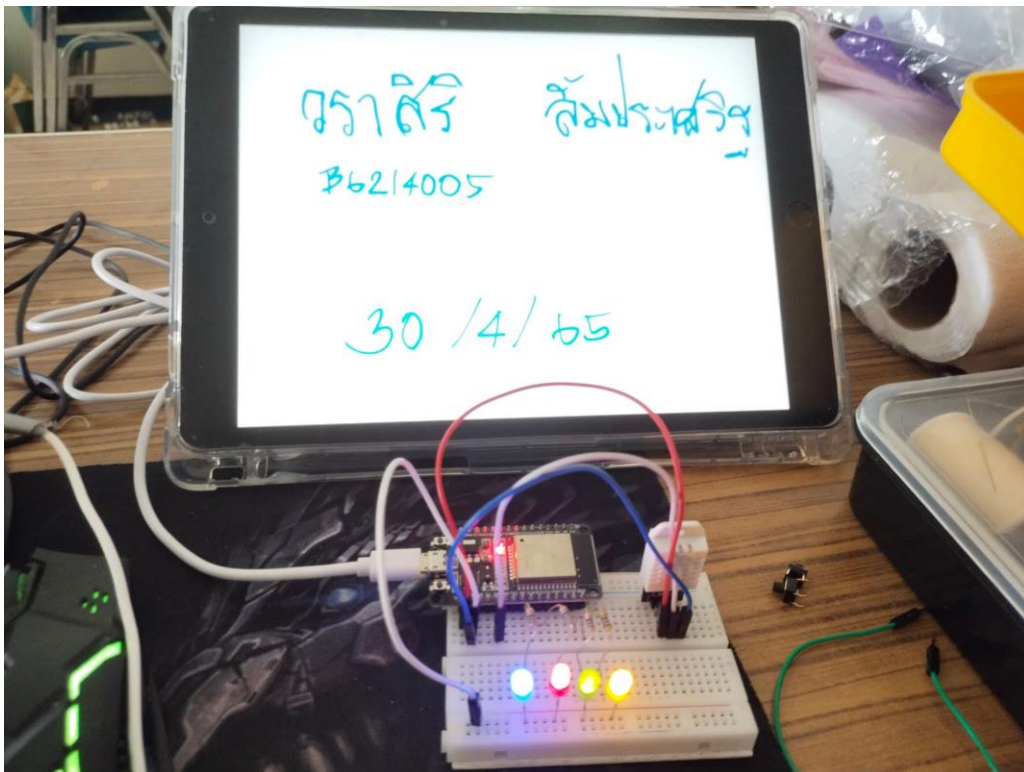
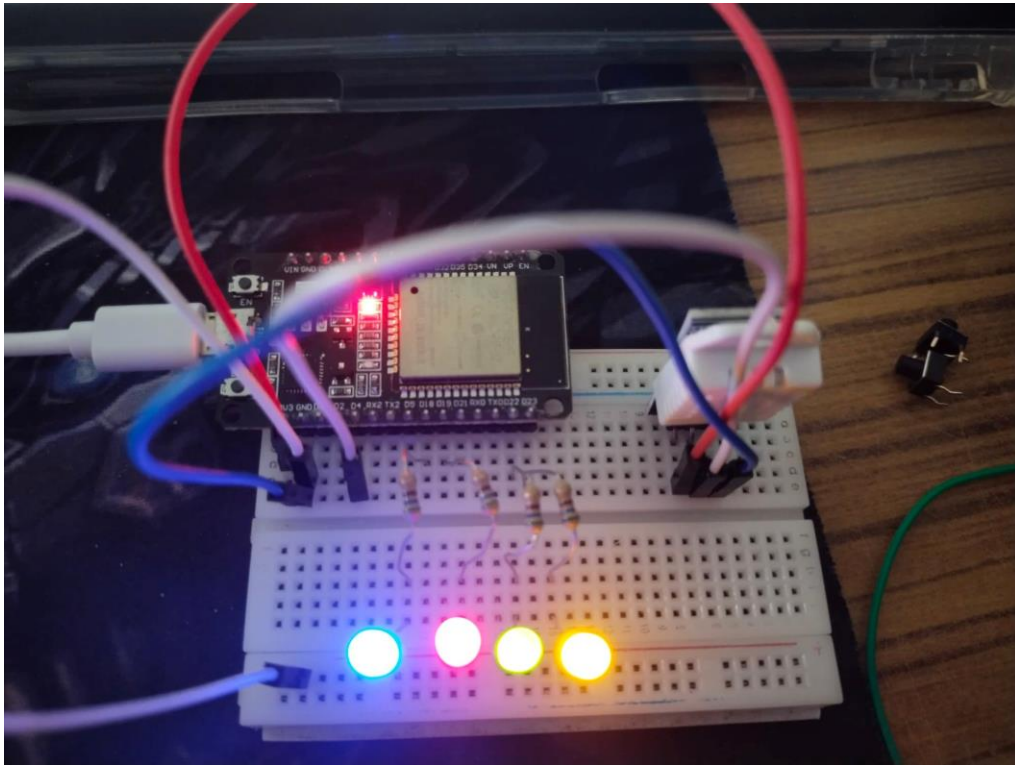
const char MAIN_page[] PROGMEM = R"====(
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h1>The ESP-32 Update web page without refresh</h1>
<button type="button" onclick="sendData(11)" style="background: rgb(202, 60, 60);width:100px;height:30px">LED1
ON</button>
<button type="button" onclick="sendData(21)" style="background: rgb(202, 60, 60);width:100px;height:30px">LED2
ON</button>
<button type="button" onclick="sendData(31)" style="background: rgb(202, 60, 60);width:100px;height:30px">LED3
ON</button>

```

```

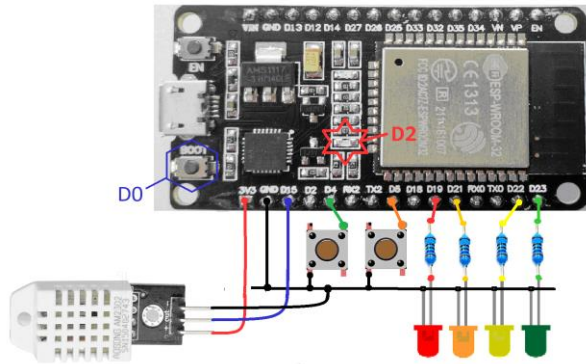
<button type="button" onclick="sendData(41)" style="background: rgb(202, 60, 60);width:100px;height:30px">LED4
ON</button><br><br>
<button type="button" onclick="sendData(10)" style="background:
rgb(100,116,255);width:100px;height:30px">LED1 OFF</button>
<button type="button" onclick="sendData(20)" style="background:
rgb(100,116,255);width:100px;height:30px">LED2 OFF</button>
<button type="button" onclick="sendData(30)" style="background:
rgb(100,116,255);width:100px;height:30px">LED3 OFF</button>
<button type="button" onclick="sendData(40)" style="background:
rgb(100,116,255);width:100px;height:30px">LED4 OFF</button><br><br>
State of [LED1, LED2, LED3, LED4] is >> <span id="LEDState">/span><br>
</div>
<div>
<br>DHT-22 sensor : <span id="ADCValue">0</span><br>
</div>
<script>
function sendData(led) {
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
document.getElementById("LEDState").innerHTML =
this.responseText;
}
};
xhttp.open("GET", "setLED?LEDstate="+led, true);
xhttp.send();
}
setInterval(function() {
// Call a function repetatively with 2 Second interval
getData();
}, 2000); //2000mSeconds update rate
function getData() {
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
document.getElementById("ADCValue").innerHTML =
this.responseText;
}
};
xhttp.open("GET", "readADC", true);
xhttp.send();
}
</script>
<br><a href="https://www.facebook.com/chi.sweethome.50/">By Varasiri Limprasert B6214005</a>
</body>
</html>
)=====";

```

Quiz_103 – Pub/Sub Data from (DHT22 + 4 LED + 2 Switch)

- อ่านค่า DHT-22 แล้วส่งไปยัง MQTT Broker ทุกๆ 5 วินาที
- กำหนดให้ใช้ mqtt.eclipse.org เป็น Broker
- ควบคุมการปิดเปิด 4 LED
- รับคำสั่งที่กำหนด SW1 แจ้ง Overheat Alarm, SW2 แจ้ง Intruders Alarm



```
#include <WiFi.h>
#include <Wire.h>
#include <PubSubClient.h>
#include "DHTesp.h"
DHTesp dht;

#define testLED1 18
#define testLED2 19
#define testLED3 22
#define testLED4 23
#define DHT22_Pin 15

const char* ssid = "V2036";
const char* password = "fnafchica";
const char* mqtt_server = "test.mosquitto.org";
const char* topic1 = "bearish";

String ledState1 = "NA";
int pushButton1 = 2;
int pushButton2 = 4;

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
```



```

Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
}
randomSeed(micros());
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
pinMode(testLED1, OUTPUT);
pinMode(testLED2, OUTPUT);
pinMode(testLED3, OUTPUT);
pinMode(testLED4, OUTPUT);
}

void callback(char* topic, byte* payload, unsigned int length)
{ char myPayload[50];
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  { Serial.print((char)payload[i]);
    myPayload[i] = payload[i];
    myPayload[i + 1] = '\0'; // End of String
  }
  Serial.print("\n ---> "); Serial.println(myPayload);
  myPayload[4] = '\0'; // String less than 4 characters
  if ((String)myPayload == "ON1") digitalWrite(testLED1, HIGH);
  if ((String)myPayload == "OFF1") digitalWrite(testLED1, LOW);
  if ((String)myPayload == "ON2") digitalWrite(testLED2, HIGH);
  if ((String)myPayload == "OFF2") digitalWrite(testLED2, LOW);
  if ((String)myPayload == "ON3") digitalWrite(testLED3, HIGH);
  if ((String)myPayload == "OFF3") digitalWrite(testLED3, LOW);
  if ((String)myPayload == "ON4") digitalWrite(testLED4, HIGH);
  if ((String)myPayload == "OFF4") digitalWrite(testLED4, LOW);
}

void reconnect()
{ while (!client.connected()) // Loop until we're reconnected
  { Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX); // Create a random client ID
    if (client.connect(clientId.c_str())) // Attempt to connect
    { Serial.println("connected"); // Once connected, publish an announcement...
      client.publish(topic1, "Hello World Pk007"); // ... and resubscribe
    }
  }
}

```

```

    client.subscribe(topic1);
  } else
  { Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
    delay(5000);
  }
}
}

void setup()
{ Serial.begin(115200);
  setup_wifi();
  dht.setup(DHT22_Pin, DHTesp::DHT22);
  pinMode(pushButton1, INPUT_PULLUP);
  pinMode(pushButton2, INPUT_PULLUP);
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  pinMode(testLED1, OUTPUT);
  pinMode(testLED2, OUTPUT);
  pinMode(testLED3, OUTPUT);
  pinMode(testLED4, OUTPUT);
}

void loop()
{
  if (!client.connected()) reconnect();
  client.loop();
  long now = millis();
  if (now - lastMsg > 5000)
  { lastMsg = now;
    ++value;
    float h = dht.getHumidity();
    float t = dht.getTemperature();
    sprintf (msg, "TempC: %.2f C, Humidity: %.2f %%", t, h);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topic1, msg);
  }
  if (digitalRead(pushButton1) == 0) {
    sprintf (msg, "Overheat Alarm");
    Serial.println(msg);
    client.publish(topic1, msg);
    delay(500);
  }
  if (digitalRead(pushButton2) == 0) {
    sprintf (msg, "Intruders Alarm");

```

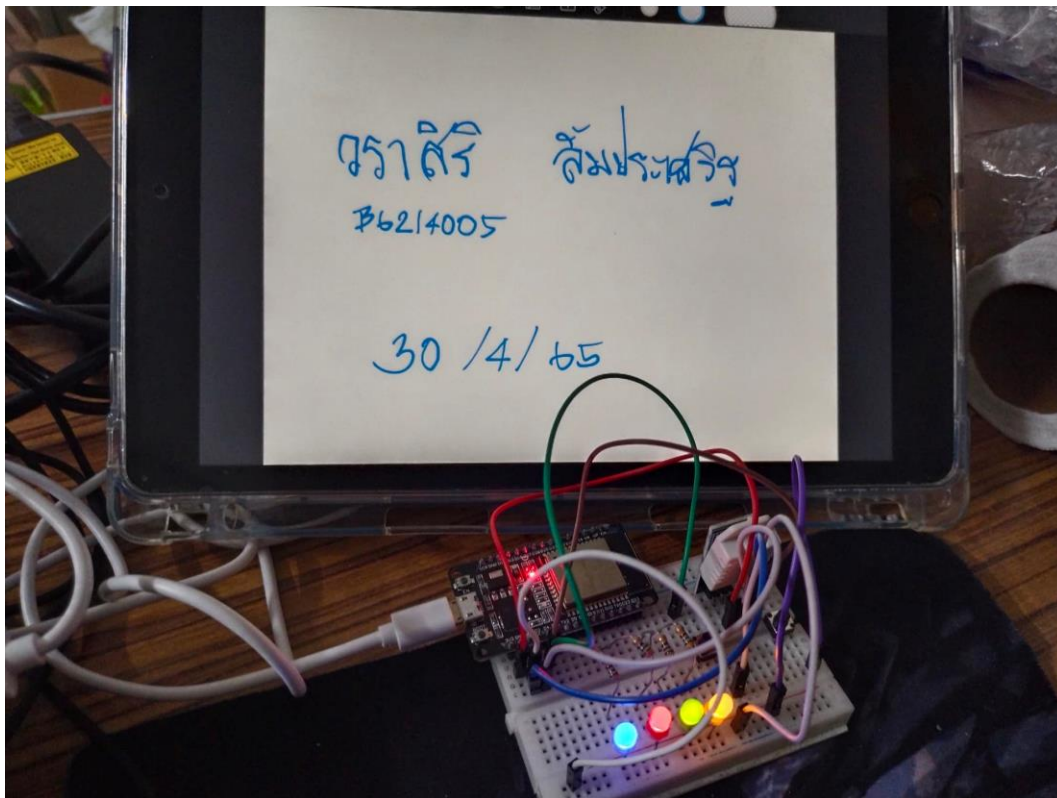
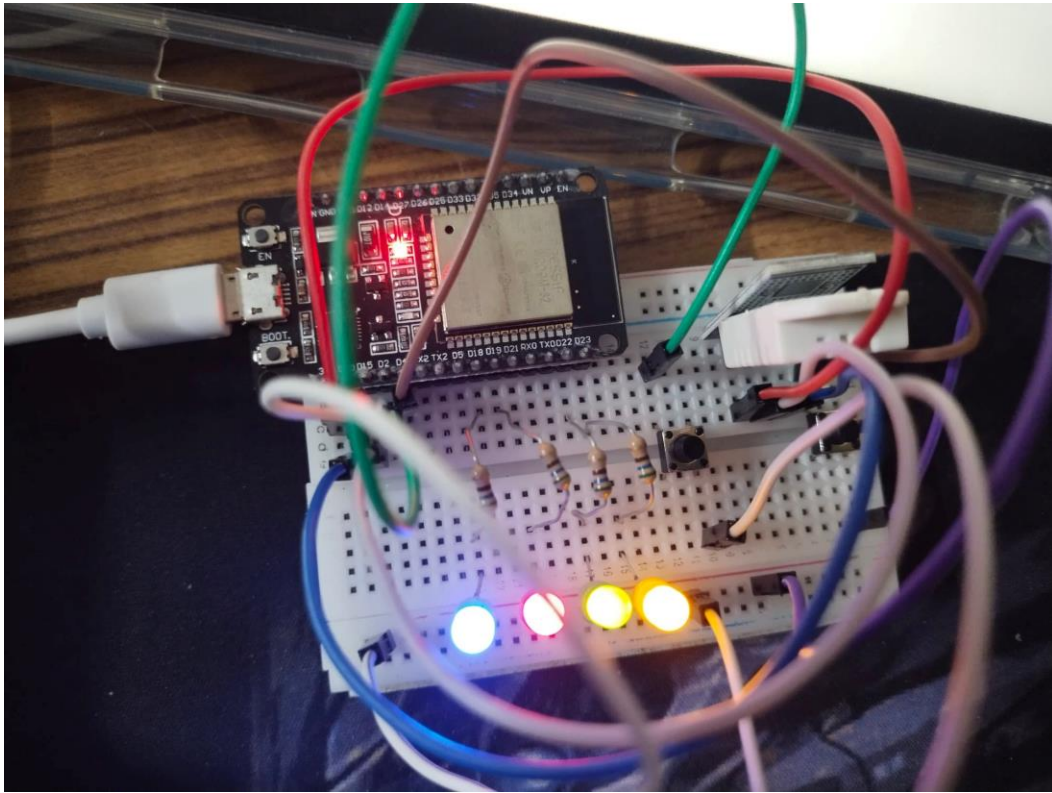
```
Serial.println(msg);  
client.publish(topic1, msg);  
delay(500);  
}  
}
```

The screenshot displays the ThingsBoard IoT Platform interface. On the left, a sidebar shows a connection to 'bearish'. The main panel is divided into several sections:

- Subscribe:** A section with a text input field containing 'bearish', a dropdown menu set to '0 - at most once', and a green 'SUBSCRIBE' button.
- Publish:** A section with a text input field containing 'bearish', a dropdown menu set to '0 - at most once', a checkbox for 'Retained', and a green 'PUBLISH' button.
- Message:** A section with a text input field containing 'ON4'.
- Subscriptions:** A section titled 'Topic: "bearish"' showing the last 5 messages. It includes a trash icon and a message count 'Messages: 0/390'.

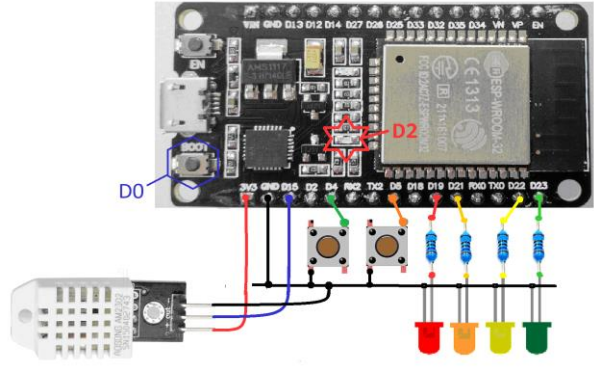

The messages list shows the following details for each entry:

#	Time	Topic	QoS	Message
385	8:42:27	bearish	0	TempC: 31.10 C, Humidity: 69.40 %
386	8:42:27	bearish	0	Overheat Alarm
387	8:42:28	bearish	0	Overheat Alarm
388	8:42:28	bearish	0	Overheat Alarm



Quiz_104 – Blynk and LINE from (DHT22 + 4 LED + 2 Switch)

- ควบคุมการปิดเปิด 4 LED
- อ่านค่า DHT-22 แล้วส่งไปยัง Blynk ทุกๆ 5 วินาที
- บันทึกค่าไปยัง Google Sheet
- หากอุณหภูมิเกิน 28°C ให้แจ้งไปยัง LINE
- รับคำสั่งวัดที่กำหนด SW1 แจ้ง Overheat Alarm, SW2 แจ้ง Intruders Alarm ไปยัง LINE

```

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <HTTPClient.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include "DHTesp.h"
#define DHT22_Pin 15
#define sw1 2
#define sw2 4

#define WebHooksKey "oXSQX-hS7mc2o1bIAA3UlubXBXN2WlrMliheoCkvYQI"
#define WebHooksEventName "Test_Key"
char auth[] = "Y1ccpnuLjmwpmQ1n_ZqSVxraOe88oHp";
char ssid[] = "V2036";
char pass[] = "fnafchica";

DHTesp dht;
WidgetLED LED1(V2);
WidgetLED LED2(V3);
BlynkTimer timer;

void setup() {
  Serial.begin(115200);
  dht.setpin(DHT22_Pin, DHTesp::DHT22); // Connect DHT sensor to GPIO 15
  pinMode(sw1, INPUT_PULLDOWN);
  pinMode(sw2, INPUT_PULLDOWN);
  Blynk.begin(auth, ssid, pass);
  timer.setInterval(1000L, myTimerEvent);
}

```

```

void myTimerEvent() {
  float humidity = dht.getHumidity();
  float temperature = dht.getTemperature();
  Blynk.virtualWrite(V0, temperature);
  Blynk.virtualWrite(V1, humidity);
  if (digitalRead(sw1)) LED1.on();
  else LED1.off();
  if (digitalRead(sw2)) LED2.on();
  else LED2.off();
  Serial.print(" Temp('C) >> "); Serial.print(temperature, 1);
  Serial.print(", Humidity(%) >> "); Serial.println(humidity, 1);
}

void loop()
{
  Blynk.run();
  if (digitalRead(sw1) == LOW) {
    String serverName = "http://maker.ifttt.com/trigger/" +
      String(WebHooksEventName) + "/with/key/" + String(WebHooksKey);
    String httpRequestData = "value1=" + String("Door Open Alarm");
    Serial.println("Server Name : " + serverName);
    Serial.println("json httpRequestData : " + httpRequestData);
    if (WiFi.status() == WL_CONNECTED) {
      HTTPClient http;
      http.begin(serverName);
      http.addHeader("Content-Type", "application/x-www-form-urlencoded");
      int httpResponseCode = http.POST(httpRequestData);
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      http.end();
      if (httpResponseCode == 200)
        Serial.println("Successfully sent");
      else
        Serial.println("Failed!");
    }
  }
  else {
    Serial.println("WiFi Disconnected");
  }
}

if (digitalRead(sw2) == LOW) {
  String serverName = "http://maker.ifttt.com/trigger/" +
    String(WebHooksEventName) + "/with/key/" + String(WebHooksKey);
  String httpRequestData = "value1=" + String("Intruders Alarm");
  Serial.println("Server Name : " + serverName);
  Serial.println("json httpRequestData : " + httpRequestData);
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;

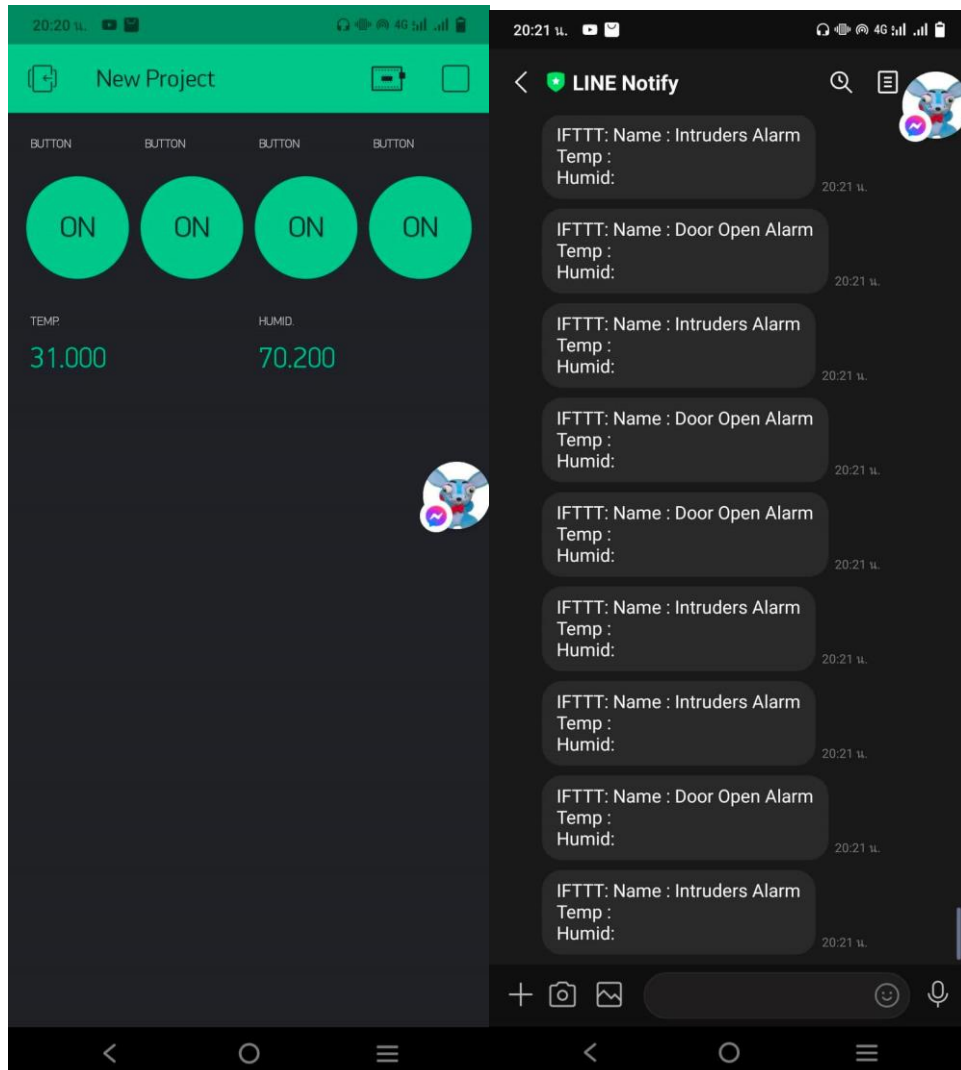
```

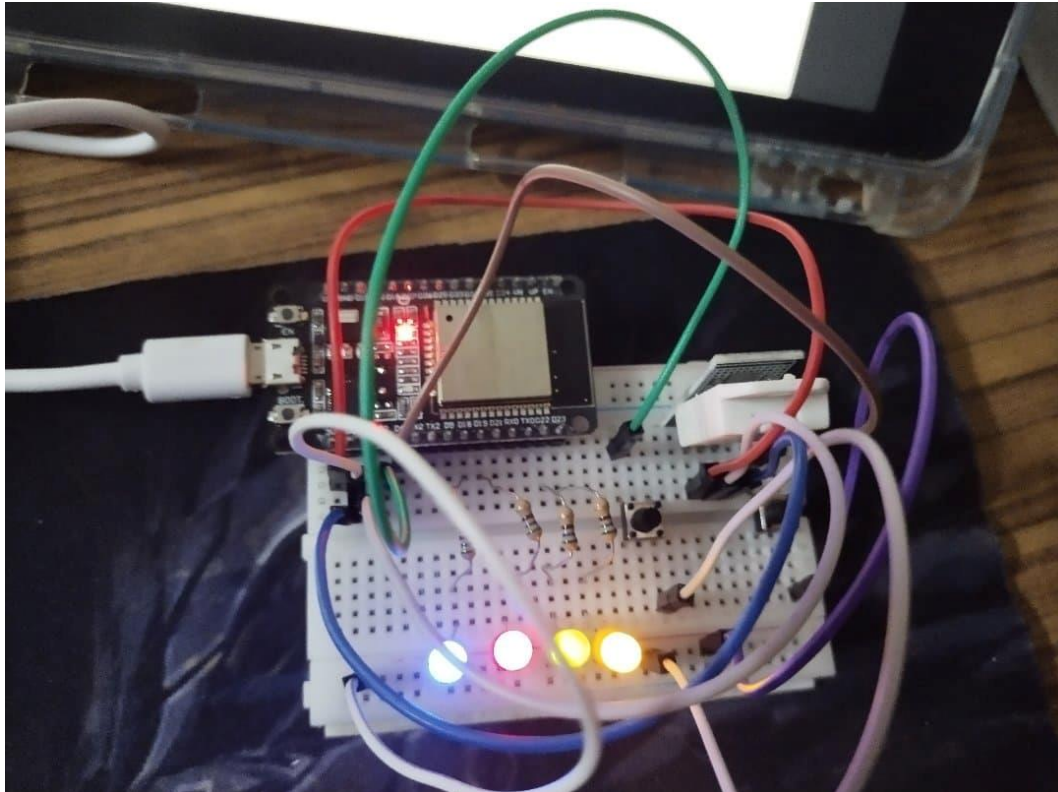
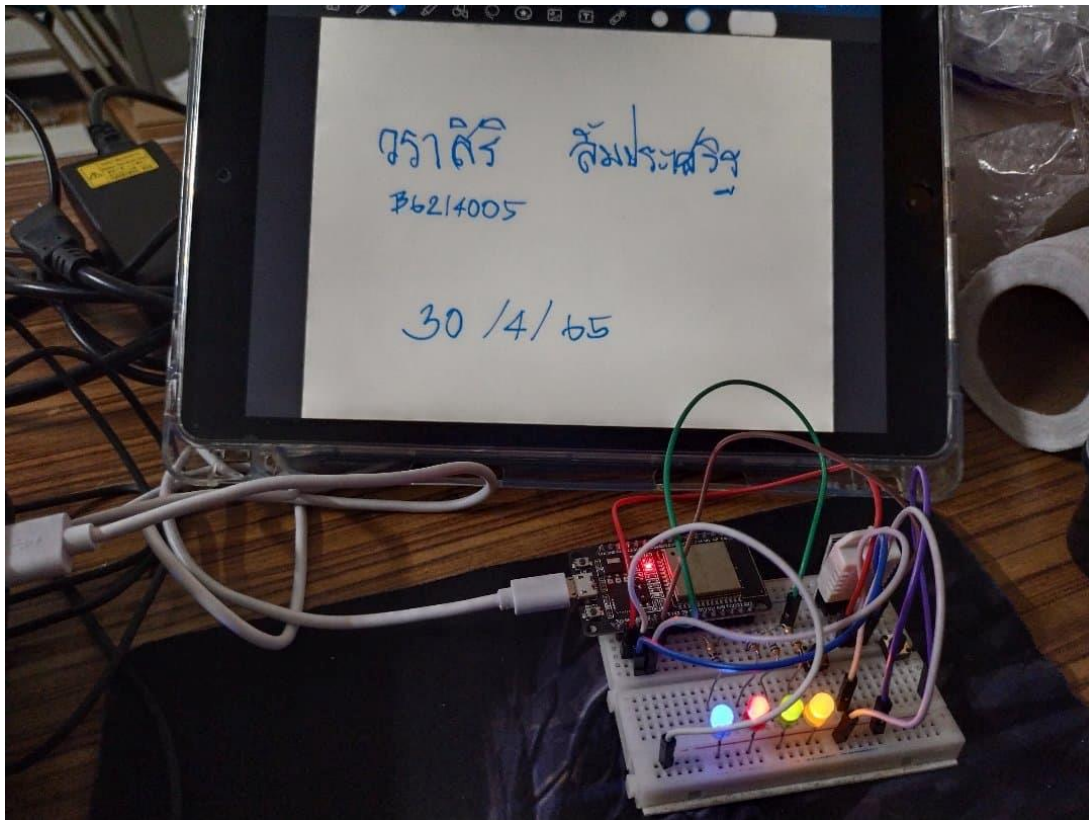


```

http.begin(serverName);
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
int httpResponseCode = http.POST(httpRequestData);
Serial.print("HTTP Response code: ");
Serial.println(httpResponseCode);
http.end();
if (httpResponseCode == 200)
    Serial.println("Successfully sent");
else
    Serial.println("Failed!");
}
else {
    Serial.println("WiFi Disconnected");
}
timer.run(); // running timer every 250ms
}
}

```





การใช้งาน ThingsBoard IoTs Platform เพื่อสร้างและจัดการระบบอัจฉริยะ
ThingsBoard IoTs Platform for smart system

ชื่อ-สกุล : วราสิริ ลิ้มประเสริฐ B6214005

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 "V2036"
#define WIFI_PASSWORD "fnafchica"
#define TOKEN "wcbt10HzC54MQwRO6DeA"
#define THINGSBOARD_SERVER "demo.thingsboard.io"
#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)) {
```

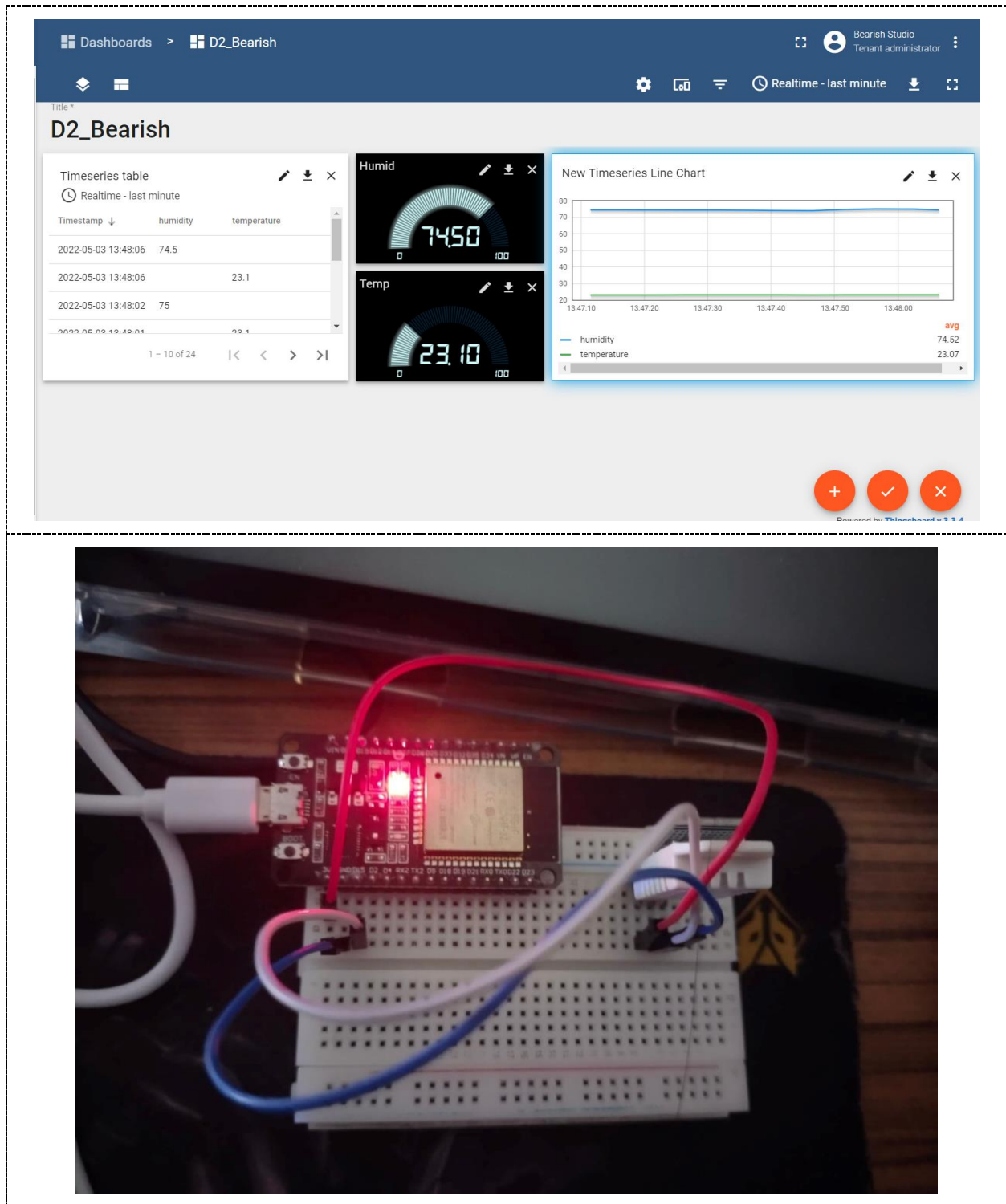
```

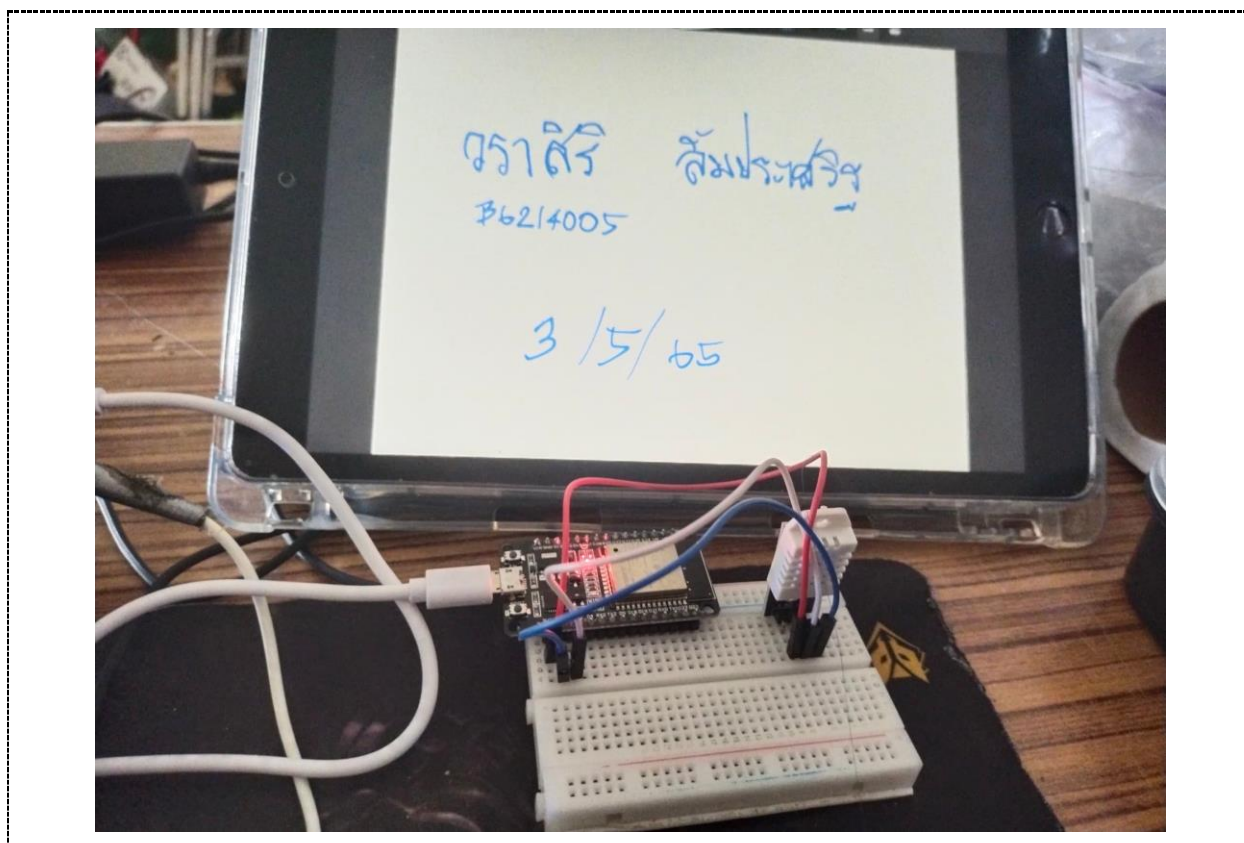
        Serial.println("Failed to connect");
        return;
    }
}
Serial.print("Sending data...");
// Uploads new telemetry to ThingsBoard using MQTT.
// See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
// for more details
//tb.sendTelemetryInt("temperature", xTempp);
//tb.sendTelemetryInt("humidity", 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");
    }
}
}

```





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

Code

```

#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 "V2036"
#define WIFI_PASSWORD "fnafchica"
#define TOKEN "wcbt10HzC54MQwRO6DeA"
#define THINGSBOARD_SERVER "demo.thingsboard.io"

#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[] = {18, 19, 22, 23};

int leds_Status[] = { 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 currently 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 (!Itb.connected()) {
        Subscribed_Status = false;
        // Connect to the ThingsBoard
        Serial.print("Connecting to: "); Serial.print(THINGSBOARD_SERVER);
        Serial.print(" with token "); Serial.println(TOKEN);
        if (!Itb.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 (!Itb.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

```

//_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());
    }
}
}

```

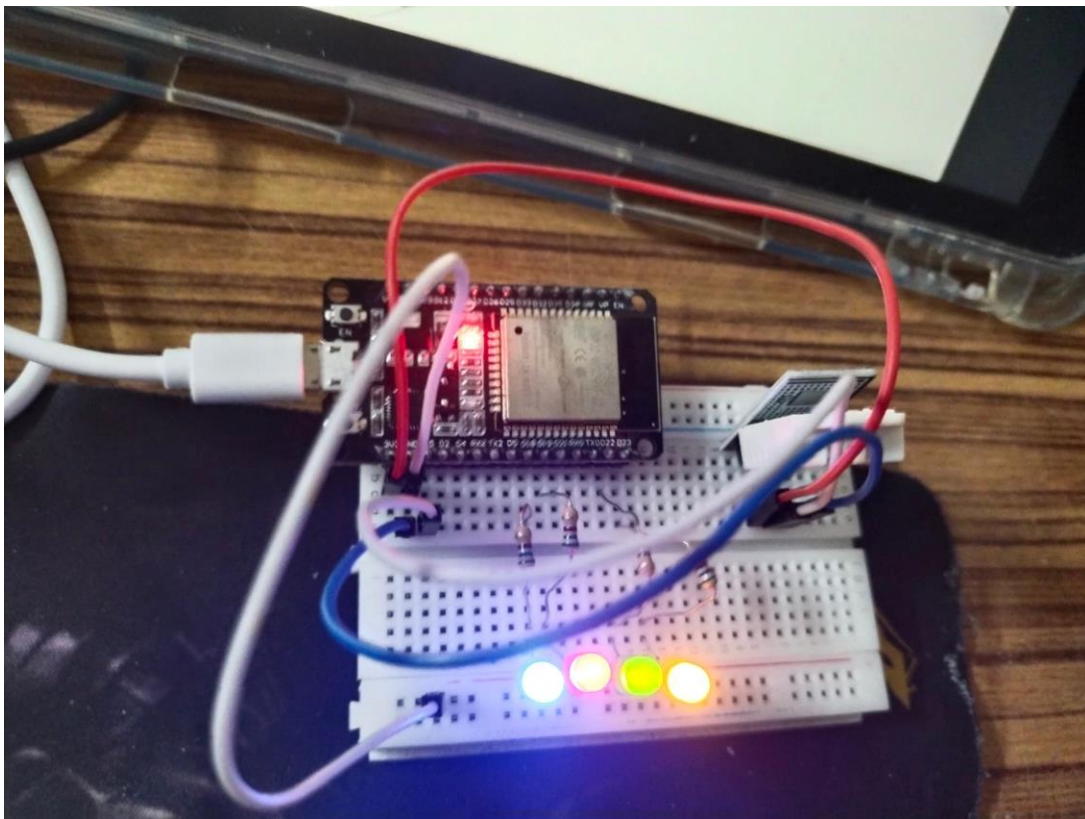
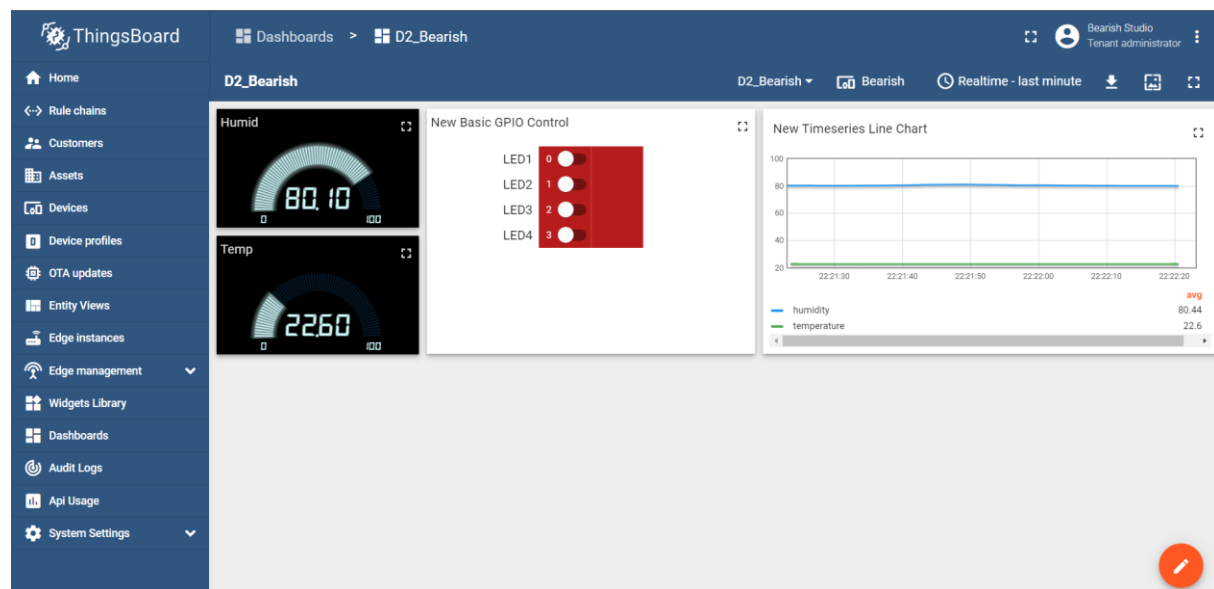
_ThingBoardRPC.h

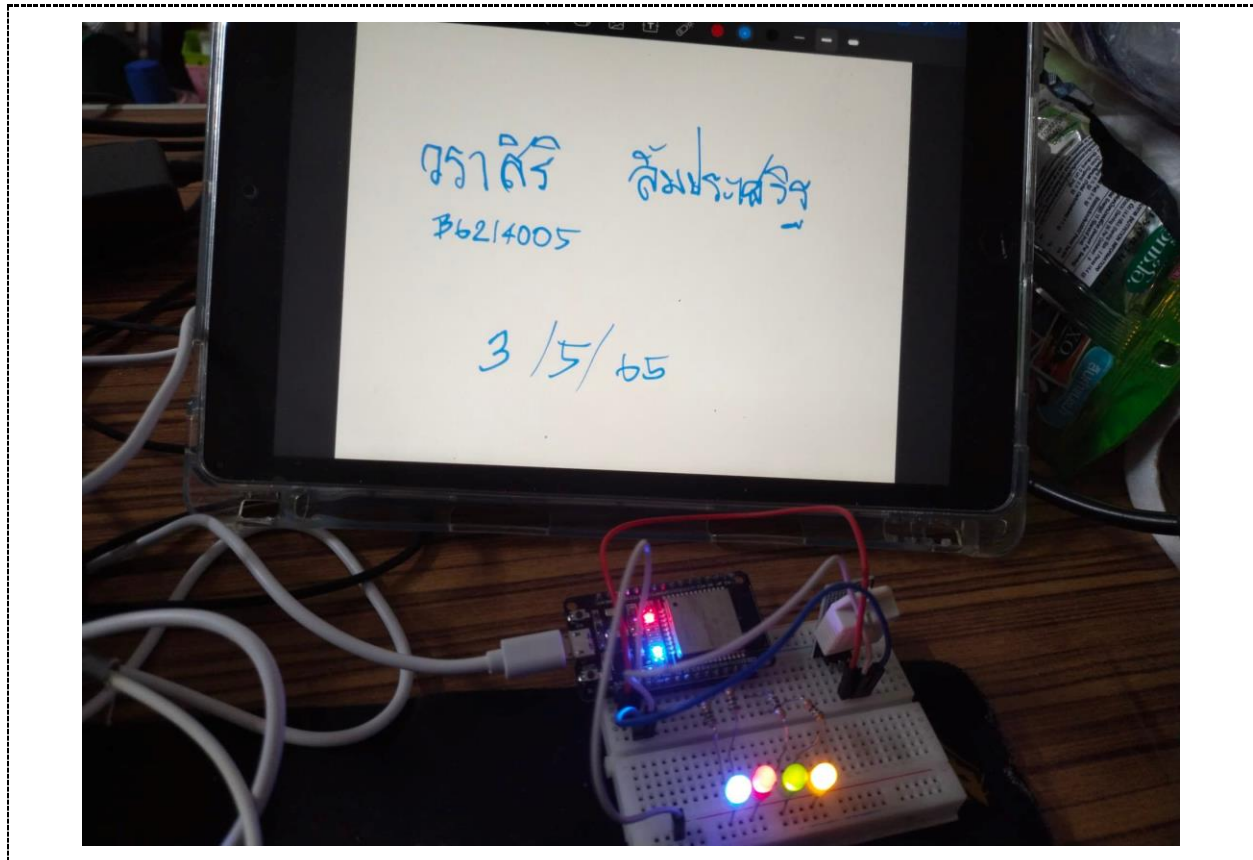
```

#####
// 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 },  
};
```





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/> }

Code

```
#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 "V2036"
#define WIFI_PASSWORD "fnafchica"
#define Device_Name "Bearish"
#define Device_Token "wcbt10HzC54MQwRO6DeA"
#define THINGSBOARD_SERVER "demo.thingsboard.io"
#define GPIO1_ESP32Pin 18
#define GPIO2_ESP32Pin 19
#define GPIO3_ESP32Pin 22
#define GPIO4_ESP32Pin 23

#define Rand "random"
boolean gpioState[] = {false, false, 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

```

// File 2 of 3
// _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];
    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());
    }
}
}

```

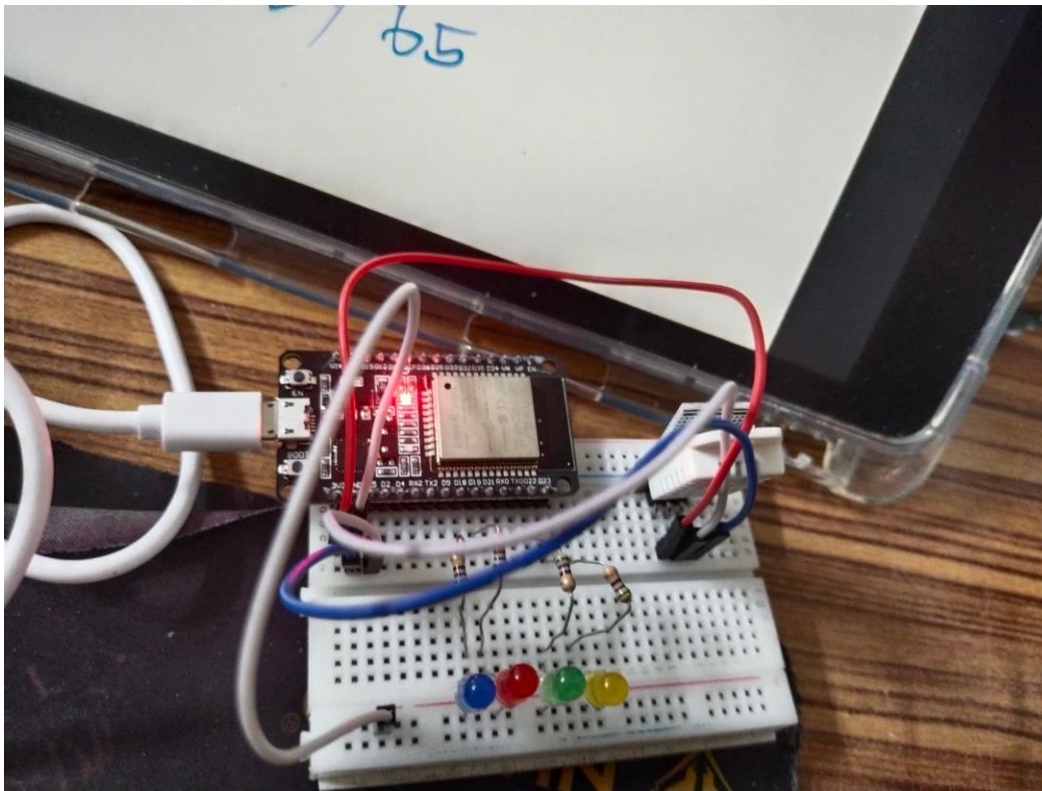
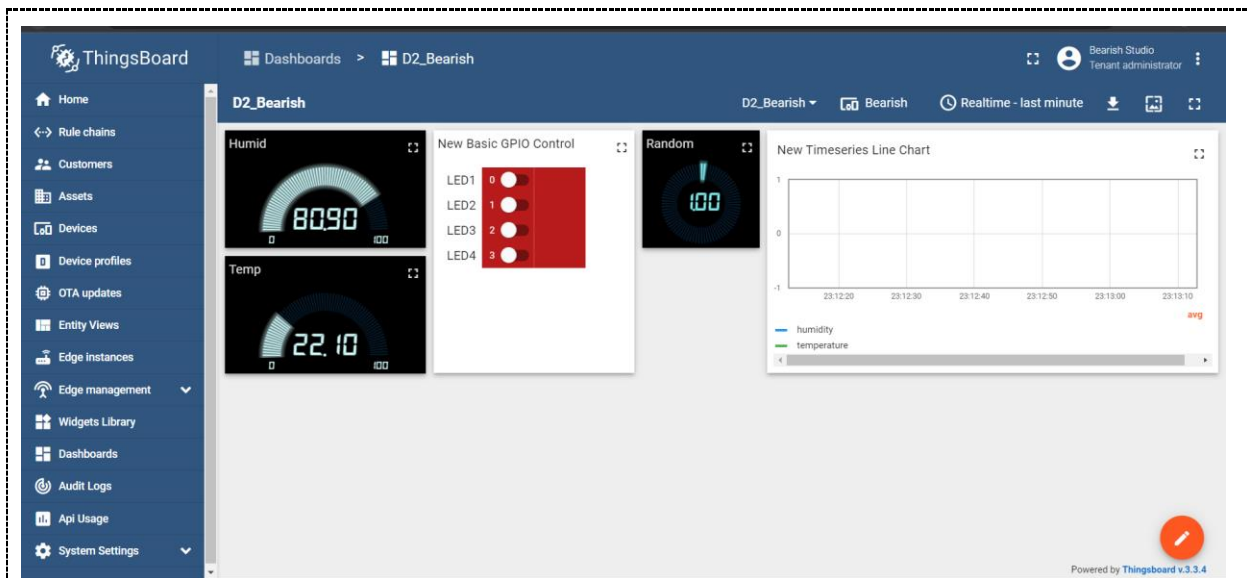
_WifiConnect.h

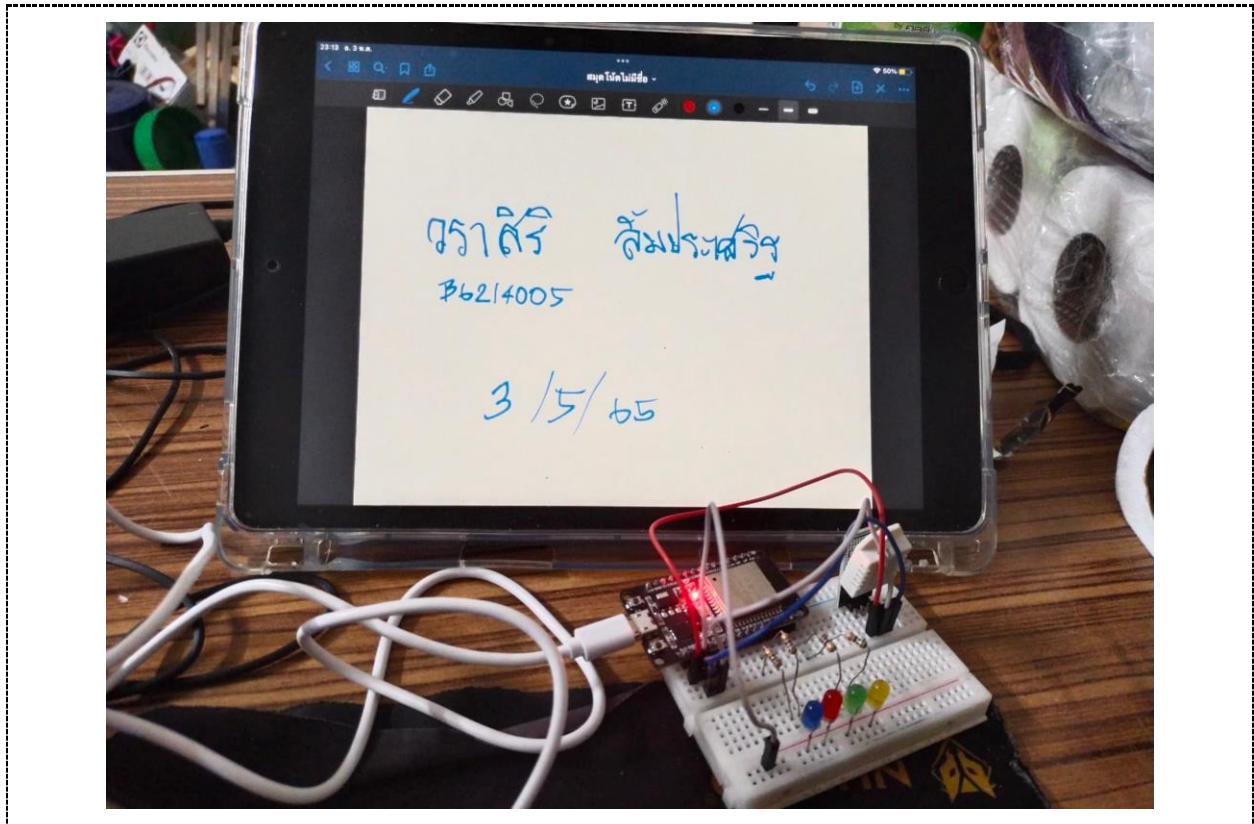
```
// File 3 of 3
// _WifiConnect.h
//=====
//=====

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
        }
    }
}
}
```





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 "V2036"
#define WIFI_PASSWORD "fnafchica"
#define TOKEN "wcbt10HzC54MQwRO6DeA"
#define THINGSBOARD_SERVER "demo.thingsboard.io"
#include <Arduino.h>
#include "ArduinoJson.h"

#define SERIAL_DEBUG_BAUD 115200
WiFiClient espClient;
ThingsBoard tb(espClient);
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;
    }
  }
}
```

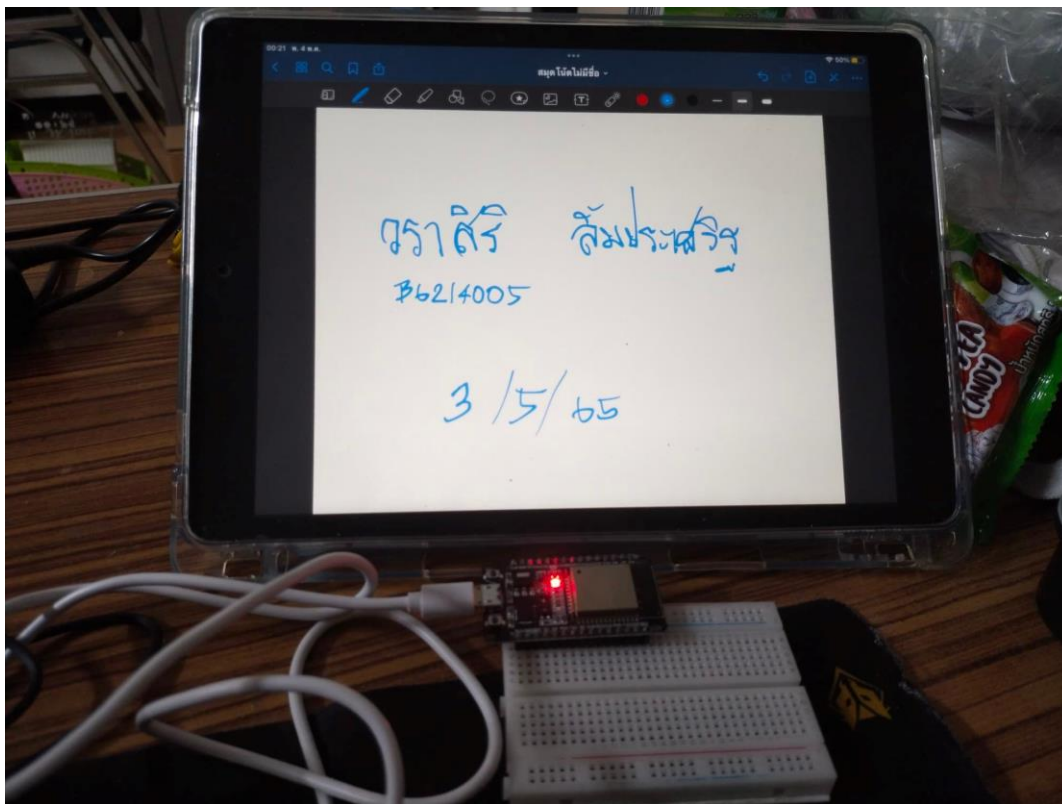
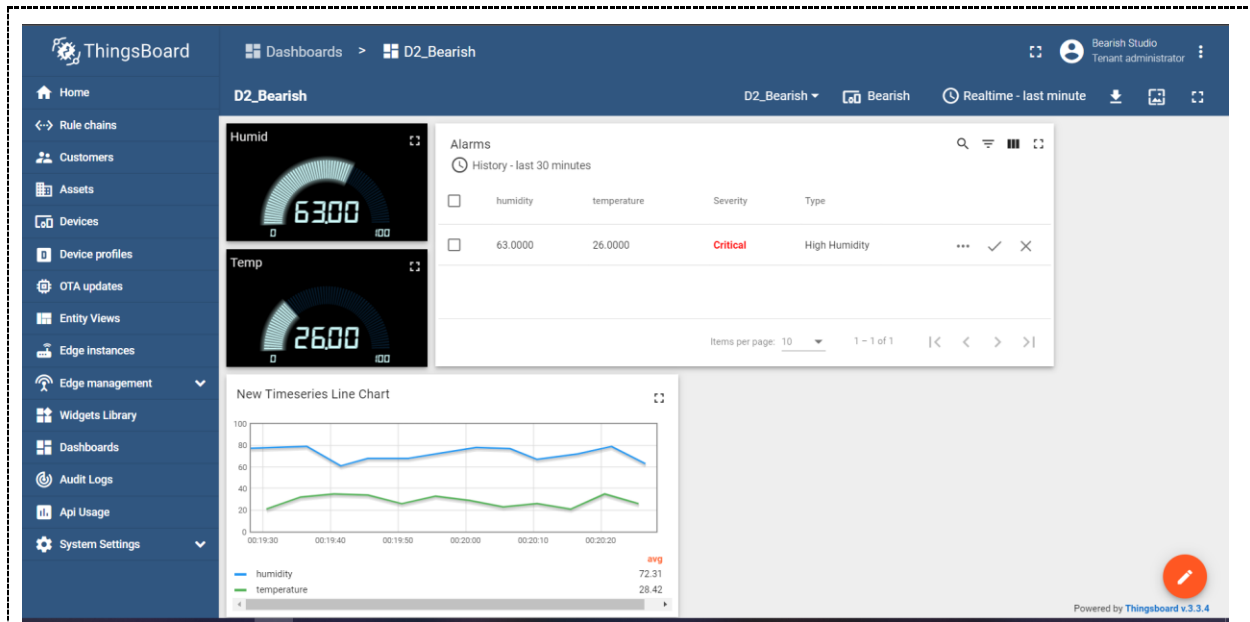
```

    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, 40);
    float xHdmid = random(60, 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");
    }
}

```



ลิงค์ Dashboard :

<https://demo.thingsboard.io/dashboard/01e00640-caab-11ec-9a68-6b50da95566e?publicId=7fa73190-c037-11eb-8f11-41ff5faa9969>