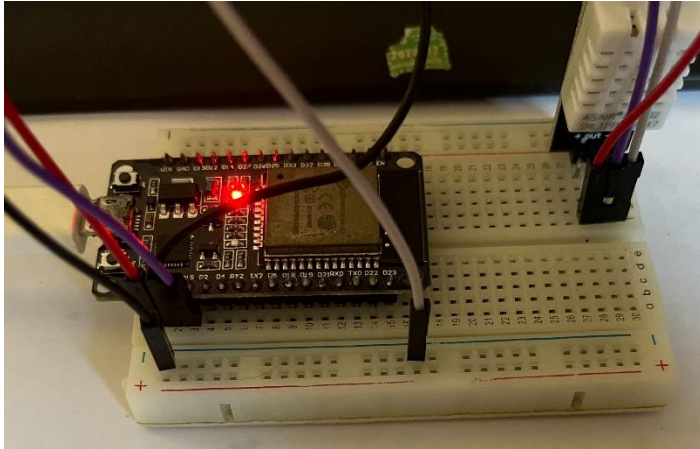


D2 การทดลองที่ 1

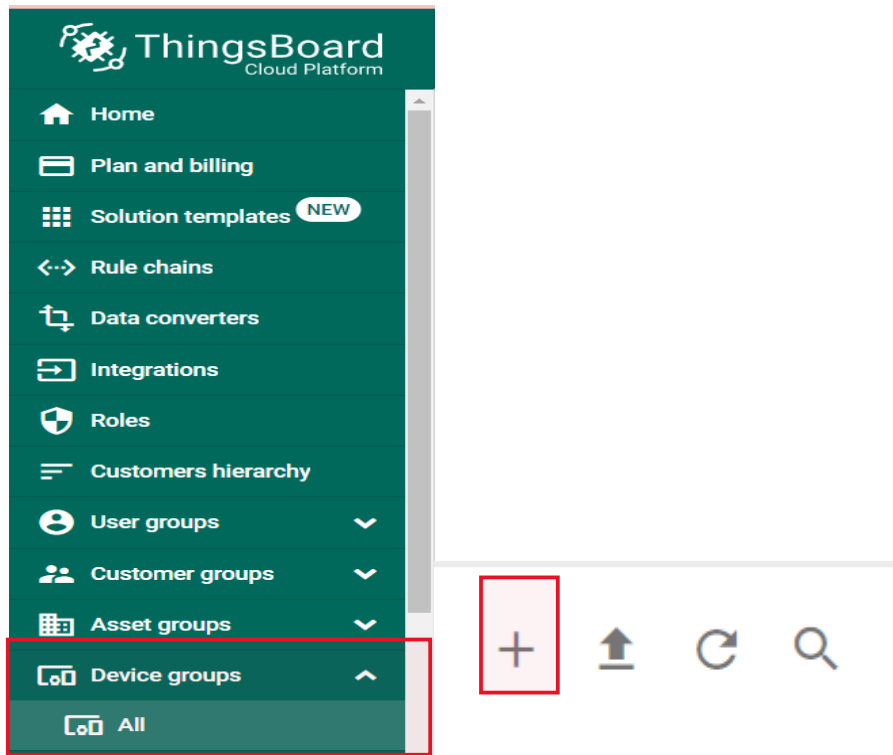
Mission - 1/4: ให้ส่งข้อมูลค่า Humidity และ Temperatures จากเซ็นเซอร์ DHT-22 ไปยัง Dashboard

1. ต่อดังตามรูป



2. สมัครเข้าใช้งาน Thingsboard (<https://thingsboard.cloud/signup>) ใช้งานแบบ cloud platform

3. สร้าง device เพื่อรับข้อมูลจากเซ็นเซอร์



Add new device
?
X

1 Device details
2 Credentials Optional

Name *

Label

☒ Select existing device profile

Device profile *
default
X

☐ Create new device profile

☐ Is gateway

Description

Next: Credentials

Cancel Add

4.เมื่อสร้างแล้วจะได้ Token ที่นำมาใช้งานได้

Temp
Device details

Details Attributes Latest telemetry Alarms Events

Open details page
Manage credentials
Delete device

Copy device Id
Copy access token

5.ติดตั้งไลบรารี ThingsBoard V0.4.0, ArduinoHttpClient V0.4.0, ArduinoJson by Benoit Blanchon V6.18.0 เพิ่มเติม จากนั้นอัปโหลดโค้ดลง ESP32

```
#include "ThingsBoard.h"
#include <WiFi.h>
#include "DHTesp.h"
#define WIFI_AP "network"
#define WIFI_PASSWORD "password"
#define TOKEN "token"
#define THINGSBOARD_SERVER "thingsboard.cloud"
// Baud rate for debug serial
#define SERIAL_DEBUG_BAUD 115200
#define DHT_Pin 15
```

```
// Initialize ThingsBoard client
WiFiClient espClient;
DHTesp dht;
// 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(DHT_Pin, DHTesp::DHT22);
}

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 = dht.getTemperature();
  float xHdmid = dht.getHumidity();
  Serial.print(xTempp, 2);
```

```

Serial.print(",");
Serial.print(xHdmid, 2);
Serial.println();
//tb.sendTelemetryInt("temperature", xTempp);
//tb.sendTelemetryInt("humidity", xTempp);
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");
  }
}
}

```

6. จากนั้นทำการสร้าง dashboard โดยการเลือกที่ latest telemetry → เลือก humidity และ temperature → show on widget → charts → timeseries line chart

Temp

Device details

Details

Attributes

Latest telemetry

Alarms

Events

Relatio

Latest telemetry

<input type="checkbox"/>	Last update time	Key ↑	Value
<input type="checkbox"/>	2022-05-01 16:46:18	humidity	58.29999924
<input type="checkbox"/>	2022-05-01 16:46:18	temperature	31.89999962

Temp

Device details

Details

Attributes

Latest telemetry

Alarms

Events

Relations

Audit Logs

2 telemetry units selected

Show on widget

<input checked="" type="checkbox"/>	Last update time	Key ↑	Value
<input checked="" type="checkbox"/>	2022-05-01 16:46:18	humidity	58.29999924
<input checked="" type="checkbox"/>	2022-05-01 16:46:18	temperature	31.89999962

Temp

Device details

Details

Attributes

Latest telemetry

Alarms

Events

Relations

Audit Logs

Current bundle

Charts

System

Add to dashboard

Timeseries Line Chart

7.เลือกที่ create new dashboard เพื่อสร้าง dashboard ใหม่

Add widget to dashboard

☐
Select existing dashboard

Dashboard

Temp

☒
Create new dashboard

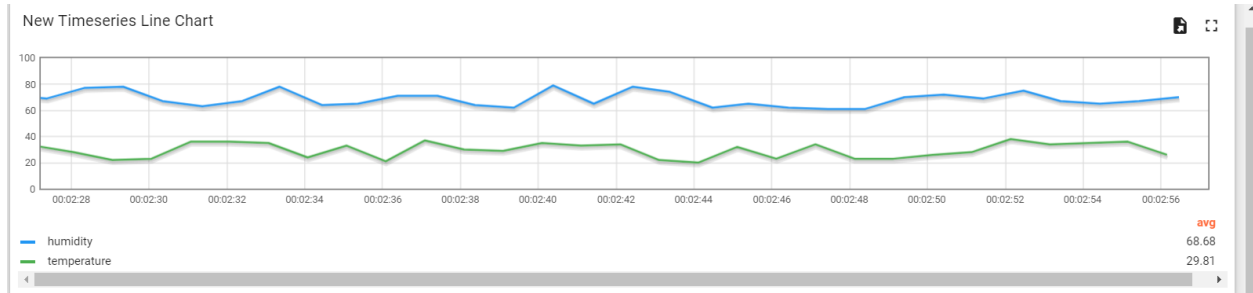
New dashboard title *

☐ Open dashboard

Cancel

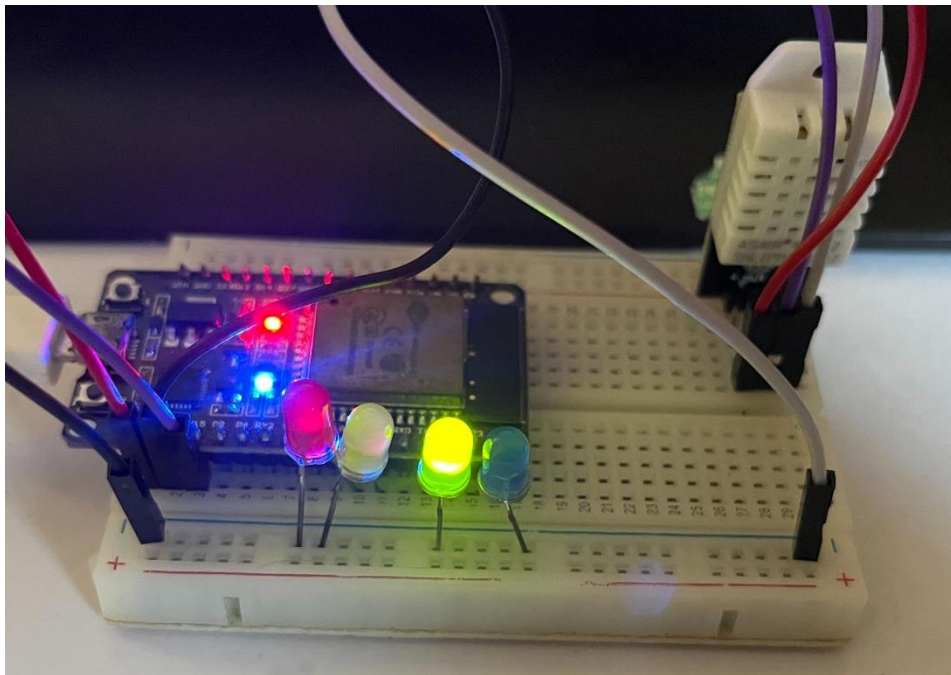
Add

8.เมื่อเปิด dashboard จะได้ดังนี้



D2 การทดลองที่ 2

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



1.ต่อวงจรตามรูป

2.ที่ Thingsboard สร้าง device ใหม่และคัดลอก Token

Device groups > All

Current subscription ThingsBoard Cloud M

Status Trial ends on the May

All: Devices

<input type="checkbox"/>	Created time ↓	Name
<input type="checkbox"/>	2022-05-09 23:59:14	Pk_T423_BoundaryC
<input type="checkbox"/>	2022-05-09 23:05:09	Device_testRules
<input type="checkbox"/>	2022-05-01 23:36:17	My_Kai
<input type="checkbox"/>	2022-05-01 22:08:27	thingsMQTT
<input type="checkbox"/>	2022-05-01 16:34:33	Led
<input type="checkbox"/>	2022-05-01 15:56:06	Temp

Led

Device details

Details

Attributes

Latest telemetry

Alarms

Events

Open details page

Manage credentials

Delete device

Copy device Id

Copy access token

Name

Led

Device profile

default

Label




LED

Assigned firmware

Assigned software

☐ Is gateway

3.สร้างไฟล์ .ino กด save และสร้างไฟล์ .h เพิ่มอีก 2 ไฟล์ ดังรูป

Name	Date modified	Type	Size
 _ConnectWifi.h	5/1/2022 4:38 PM	C Header File	1 KB
 _ThingBoardRPC.h	5/1/2022 4:37 PM	C Header File	3 KB
 2.ino	5/1/2022 4:47 PM	Arduino file	4 KB

4.อัปโหลดโค้ดทดสอบ

```
// ไฟล์ .ino
// Helper macro to calculate array size
#define COUNT_OF(x) ((sizeof(x)/sizeof(o[x])) / ((size_t)(!(sizeof(x) % sizeof(o[x])))))
#include <WiFi.h>
#include <ThingsBoard.h>
#define WIFI_AP_NAME "Network"
```

```

#define WIFI_PASSWORD "Password"
#define TOKEN "Token"
#define THINGSBOARD_SERVER "thingsboard.cloud"
#define pinLEDBlink 2
WiFiClient espClient;
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 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();
// 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.
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 = random(20, 30);
  float humidity = random(40, 50);
  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();
}

```

// ไฟล์ 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 },  
};
```

```
//ไฟล์ _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());  
  }  
}
```

5.สร้าง dashboard เลือก line chart แล้วตั้งค่าเพื่อแสดงข้อมูลอุณหภูมิและความชื้น

Timeseries Line Chart

Timeseries Line Chart

?

×

✓

×

Data

Settings

Advanced

Actions

☒ Use dashboard timewindow

☒ Display timewindow

Timewindow

⌚ Realtime - last minute

Datasources

Type

Parameters

Entity alias *

Led

×

=

●

⌚

humidity: humidity

✎

×

1.

Entity

▼

Filter

×

+ Add

6.สร้างตัวควบคุม LED ด้วย GPIO Control จากนั้นตั้งค่า

เลือกที่แถบ Advanced จากนั้นตั้งค่าดังนี้

- Pin: 0
Label: GPIO1
Row: 0
Column: 0
- Pin: 1
Label: GPIO2
Row: 0
Column: 1
- Pin: 2
Label: GPIO3
Row: 1
Column: 0
- Pin: 3
Label: GPIO4
Row: 1
Column: 1

New Basic GPIO Control

Basic GPIO Control

Data

Settings

Advanced

Actions

1. ✕

Gpio switch

Pin *
0

Label *
GPIO 1

Row *
0

Column *
0

7. สร้างตัวควบคุม LED Blink ด้วย Knob Control

New Widget → Control widget → Knob Control จากนั้นตั้งค่า target device เป็น device ที่สร้างขึ้นในข้อที่ 2

New Knob Control

Knob Control

Data

Settings

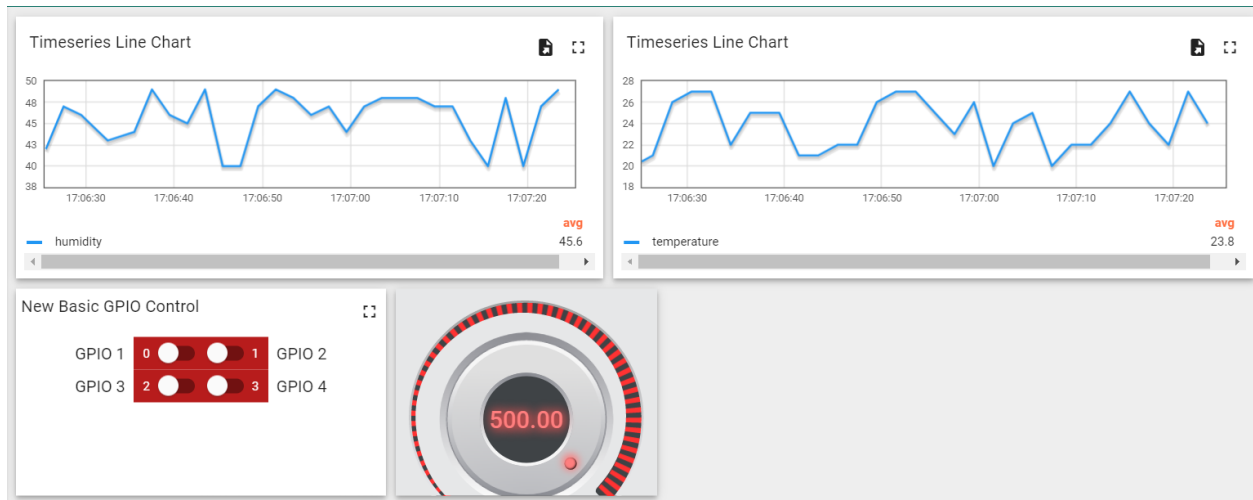
Advanced

Actions

Target device
Led

Data settings

8.ผลลัพธ์ที่ได้จากการทำงาน

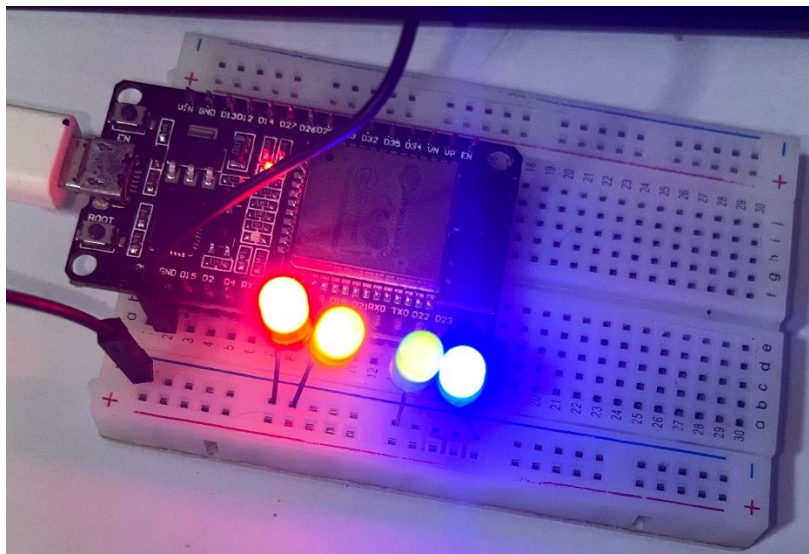


D2 การทดลองที่ 3

Mission 3/4: ให้ใช้ MQTT กับ ThingsBoard

o ปรับปรุงเพื่อให้ทำงานควบคุมการ On/Off - 4 LED

o เพิ่มเติม คือ ทดสอบส่งข้อมูล 1 ค่าแบบสุ่มระหว่าง 00 - 50 ไปแสดงที่ Dashboard ด้วย ได้หรือไม่



1.ต่อวงจรดังรูป

2.ที่ Thingsboard สร้าง device ใหม่และคัดลอก Token

thingsMQTT

Device details

Details

Attributes

Latest telemetry

Alarms

Events

Relations

Open details page

Manage credentials

Delete device

Copy device Id

Copy access token

Name

thingsMQTT

Device profile




default

Label

TM

Assigned firmware

3.สร้างไฟล์ .ino กด save และสร้างไฟล์ .h เพิ่มอีก 2 ไฟล์ ดังรูป

	_HandOnMQTT.h	5/1/2022 11:23 PM	C Header File	3 KB
	_WifiConnect.h	5/1/2022 9:28 PM	C Header File	2 KB
	3.ino	5/1/2022 11:23 PM	Arduino file	2 KB

4.อัปโหลดโค้ดทดสอบ โดย

ไฟล์แรก 3.ino คือโค้ดส่วนของการ กำหนดค่าเครือข่าย โทเคน ไลบรารี กำหนดGPIO ส่ง publish ค่าสู่ไปที่ MQTT

ไฟล์ที่ 2 _HandOnMQTT.h คือส่วนของการ subscribe/publish สถานะของ LED ไปที่ MQTT และควบคุมการเปิด/ปิด LED

ไฟล์ที่ 3 _WifiConnect.h คือส่วนของการเชื่อมต่อกับเครือข่าย

```
// File 1 of 3
// 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 2.8.0
```



```
// replace #ifdef ESP8266
// to #if defined (ESP8266) || defined(ESP32)
#define WIFI_AP_NAME "Network"
#define WIFI_PASSWORD "Password"
#define Device_Name "thingsMQTT"
#define Device_Token "Token"
#define thingsboardServer "thingsboard.cloud"
#define GPIO1_ESP32Pin 19
#define GPIO2_ESP32Pin 21
#define GPIO3_ESP32Pin 22
#define GPIO4_ESP32Pin 23
boolean gpioState[] = {false, false, false, false};
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();
  }
  client.loop();
}
```

```
String msg = "{ Random : "+ String(random(00,50)) +" }";
client.publish("v1/devices/me/attributes", msg.c_str());
Serial.println(msg);
delay(5000);
}
```

```
// _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

```

```

//=====
//=====

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
} } }
```

5. สร้างตัวควบคุม LED ด้วย GPIO Control จากนั้นตั้งค่า

เลือกที่แถบ Advanced จากนั้นตั้งค่าดังนี้

1. Pin: 19
Label: GPIO1
Row: 0
Column: 0
2. Pin: 21
Label: GPIO2
Row: 0
Column: 1
3. Pin: 22
Label: GPIO3
Row: 1
Column: 0
4. Pin: 23
Label: GPIO4
Row: 1
Column: 1

New Basic GPIO Control

Basic GPIO Control

Data

Settings

Advanced

Actions

Gpio switches

1. ✕

Gpio switch

Pin *

19

Label *

GPIO 1

Row *

0

Column *

0

6.สร้าง card เพื่อแสดงสถานะของ LED โดยเลือก New widget → card → Simple card และตั้งค่าตามรูป

New Simple card

Simple card

Data

Settings

Advanced

Actions

Datasources

Maximum 1 datasource is allowed.

Type

Parameters

= 1.

Entity

Entity alias *

LEDcontrol

×

=

ⓘ

19:19

✎

✕

Filter

Maximum 1 timeseries/attribute is allowed.

Data settings

7. สร้าง gauge เพื่อใช้แสดงค่าของเลขที่ส่งมาจาก ESP32

New Mini gauge

Mini gauge

Data

Settings

Advanced

Actions

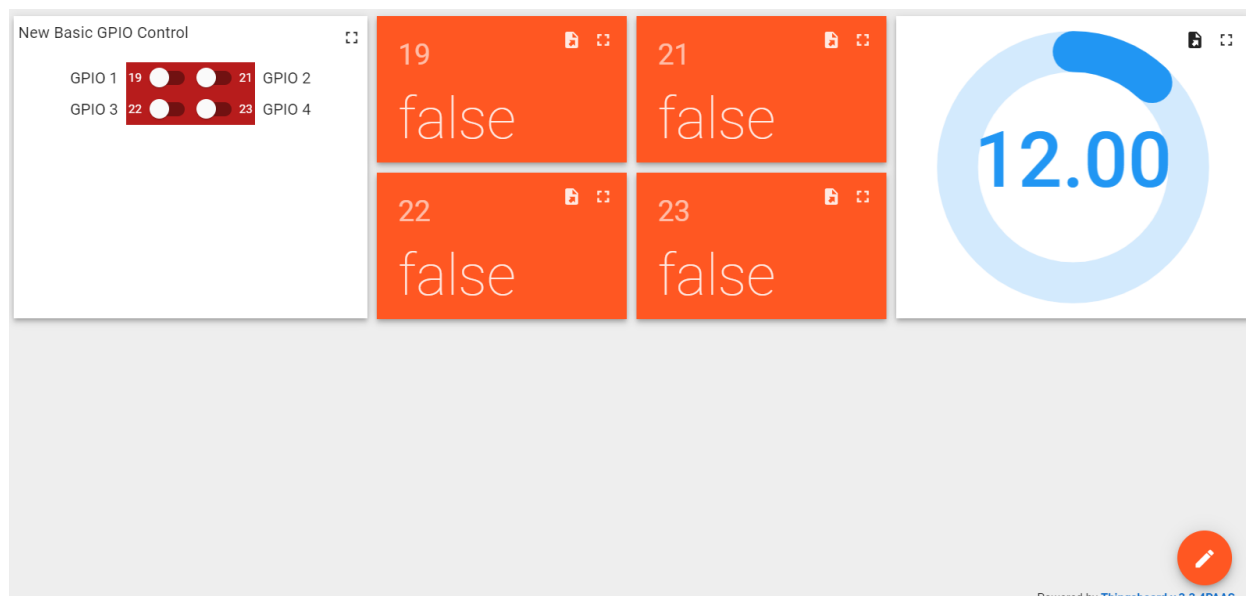
Datasources

Maximum 1 datasource is allowed.

Type	Parameters
<div><div>= 1.</div><div>Entity</div></div>	<div><div>Entity alias *</div><div>LEDcontrol</div><div>×</div></div> <div><div>Filter</div><div></div></div> <div><div>Maximum 1 timeseries/attribute is allowed.</div></div>

Data settings

8. ผลลัพธ์ที่ได้จากการทดลอง

Powered by [Thingsboard v.3.3.4PAAS](#)

D2 การทดลองที่ 4

Mission 4/4: การตรวจสอบและควบคุม อุณหภูมิ-ความชื้น ของโรงเรือนเลี้ยงไก่

o ให้ใช้ ESP32 ส่งข้อมูลแบบสุ่มสองจำนวน คือ

- Tempp_A สุ่มระหว่าง 20-40

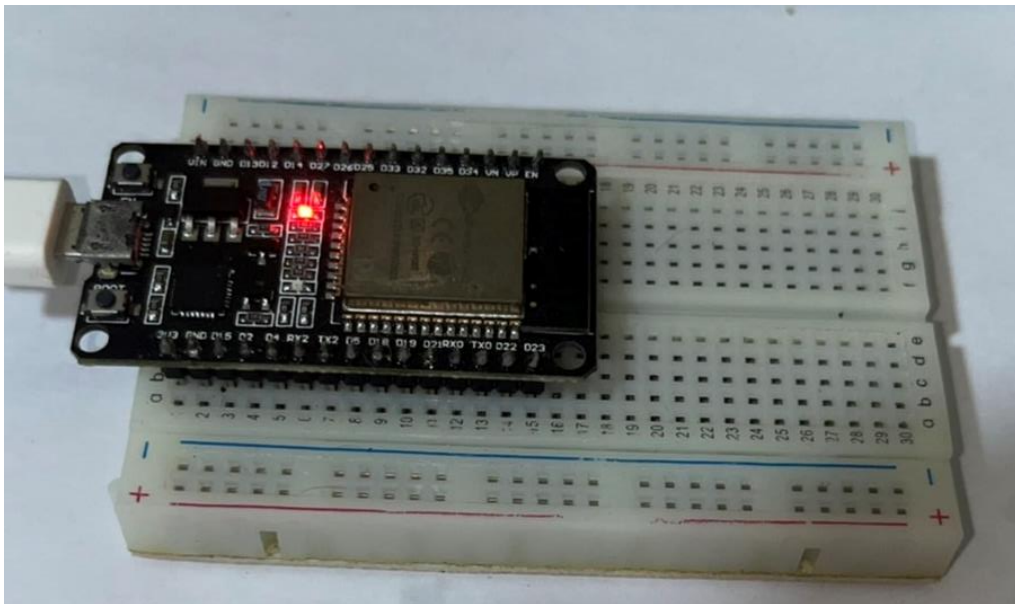
- Hudmid_A สุ่มระหว่าง 60-80

o ข้อมูลทั้งสองค่าจะนำมาแสดงที่ Dashboard

o สร้าง Alarm โดย หาก Tempp_A > 35 หรือ Hudmid_A > 70 ให้ Alarm

o กำหนดรอบการตรวจสอบทุกๆ 20 วินาที

o แชร Dashboard ไปให้ผู้ใช้งาน



1.ต่อวงจรตามรูป

2.ที่ Thingsboard สร้าง device ใหม่และคัดลอก Token

My_Kai

Device details

Details

Attributes

Latest telemetry

Alarms

Events

Relations

Open details page

Manage credentials

Delete device

Copy device Id

Copy access token

Name

My_Kai

Device profile

default

Label

Assigned firmware

3.อัปโหลดโค้ดที่ใช้ในการส่มค่า อุณหภูมิและความชื้น

```
#include "ThingsBoard.h"
#include <WiFi.h>
#define WIFI_AP "Network"
#define WIFI_PASSWORD "Password"
#define TOKEN "Token"
#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() {
  delay(1000);
  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...");
// Uploads new telemetry to ThingsBoard using MQTT.
// See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
// for more details
float Tempp = random(2000, 4000) / 100.0;
float Humid = random(6000, 8000) / 100.0;
Serial.println("Tempp = " + String(Tempp, 2) + "C");
Serial.println("Humid = " + String(Humid, 2) + "%");
tb.sendTelemetryInt("temperature", Tempp);
tb.sendTelemetryInt("humidity", Humid);
tb.loop();
}

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() {

```

```

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");
}
}

```

4.สร้าง dashboard จากนั้น Edit → entity aliases



5. เลือกที่ Add alias

Entity aliases

×

	Alias name	Entity filter	Resolve as multiple entities	
1.	MyDevice	One device	<input type="checkbox"/>	<div>✎</div> <div>×</div>

Add alias

Cancel

Save

6.ตั้งชื่อจากนั้นในหัวข้อ Filter type เลือก Single entity หัวข้อ Type เลือก Device หัวข้อ Device เลือก Device ที่สร้างในข้อที่ 2

8.สร้าง Chart และตั้งค่าตามรูป

New Timeseries Line Chart

Timeseries Line Chart

Data

Settings

Advanced

Actions

☒ Use dashboard timewindow

☒ Display timewindow

Timewindow

Realtime - last minute

Datasources

Type	Parameters
1. Entity	<div>Entity alias * MyDevice</div> <div>Filter</div> <div><div>= blue circle humidity: humidity ✎ ✕</div><div>= green circle temperature: temperature ✎ ✕</div></div>

+ Add

9.สร้าง Alarm widget โดยเลือกที่ New widget → Alarm widget → Alarm Table จากนั้นตั้งค่าตามรูป

New Alarms table

Alarms table

Data

Settings

Advanced

Actions

Alarm type list
Any type

☐ Search propagated alarms

Alarm source

Entity	Entity alias * MyDevice	✕	= blue bell Created time: createdTime ✎ ✕
			= green bell Originator: originator ✎ ✕
			= red bell Type: type ✎ ✕
			= yellow bell Severity: severity ✎ ✕
			= blue bell Status: status ✎ ✕
Filter			

10.ตั้งค่ากำหนดรอบในการตรวจสอบที่ dashboard ดังรูป

The screenshot shows a modal window with two tabs: 'Realtime' and 'History'. The 'Realtime' tab is selected. Under 'Last', there is a radio button, the text 'Last', a dropdown menu showing '1 day', and an 'Advanced' toggle switch. Under 'Interval', there is a radio button, the text 'Timezone', and a text field showing 'Browser Time (UTC+07:00)'. At the bottom are 'Cancel' and 'Update' buttons. Below the modal, a clock icon and the text 'Realtime - last day' are visible.

11.กำหนด Alarm ที่หน้า Device Profile เลือกที่ Alarm rules จากนั้น Edit เลือก add alarm rule จ

The screenshot shows the 'default' Device profile details page. The 'Alarm rules (2)' tab is selected. A red circular button with a pencil icon is visible on the right side of the page.

12.กำหนด Alarm Type กำหนด severity เป็น Critical และเพิ่ม Condition

The screenshot shows the configuration page for a 'High Temp' alarm. The 'Alarm type' is set to 'High Temp'. Under 'Create alarm rules', the 'Severity' is set to 'Critical'. The 'Condition' is 'Please add alarm rule condition'. The 'Schedule' is 'Active all the time'. The 'Add details' button is visible. The 'Mobile dashboard' is set to 'No dashboard selected'. At the bottom, there is a button labeled '+ Add create condition'.

13.เลือกที่ Add key filter

Edit alarm rule condition ✕

Key filters ^

Key name

Key type

No key filters configured

Add key filter

14.ตั้งค่าตามรูปแล้วกด Add (ทำทั้ง temperature และ humidity)

Add key filter ✕

Key type *
Timeseries

Key name *
temperature

Value type
Numeric

Filters ^

Operation

Value

greater than

35

Default value

(x) ✕

Add

Add complex

Cancel

Add

15.ผลลัพธ์ที่ได้จากการทดลอง

