หน่วยการเรียนรู้ที่ 4 พื้นฐานการใช้งาน NETPIE2020

นายธงชัย ชาบุดศรี

แผนกวิชาเทคโนโลยีสารสนเทศ

วิทยาลัยเทคนิคชลบุรี

จุดประสงค์การเรียนรู้

1. เรียนรู้ความหมายและความสำคัญของ IoT Platform
2. เรียนรู้โครงสร้างและการใช้งานพื้นฐานของ NETPIE2020
3. เรียนรู้การใช้งานอุปกรณ์เชื่อมต่อเพื่อรับส่งข้อความผ่าน NETPIE2020
4. เรียนรู้การจัดเก็บข้อมูลจากอุปกรณ์บน NETPIE2020

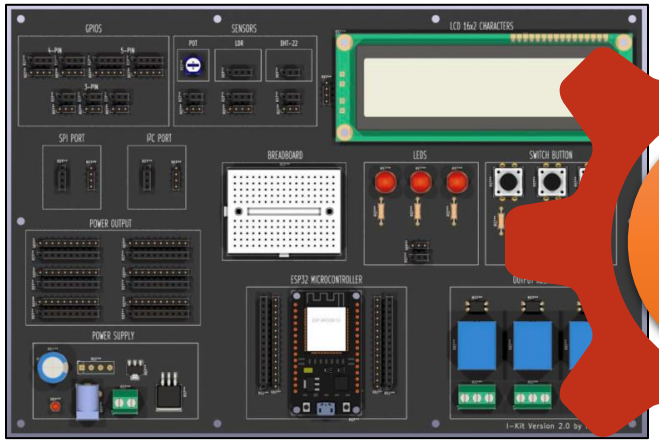
ใบงานที่ 4.1 การใช้งานเบื้องต้น และการเชื่อมต่อ NETPIE2020 ผ่าน MQTT Box

ใบงานที่ 4.1 ทฤษฎีเบื้องต้น

องค์ประกอบของ IoT

ปัจจุบันได้ศึกษาส่วนของ End Nodes และ Connectivity เสร็จสิ้นแล้ว

ลำดับถัดไปเรียนรู้ส่วนของ Servers/Cloud



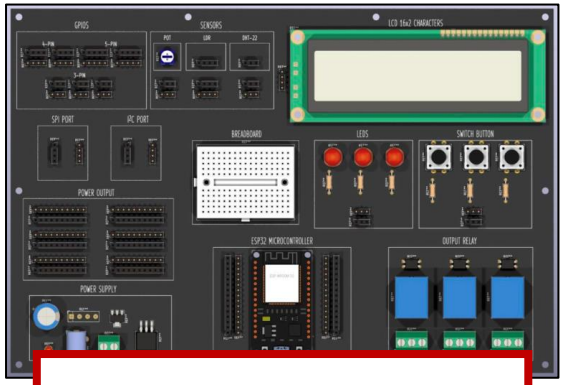
End Nodes

Connectivity

Central Servers/Cloud

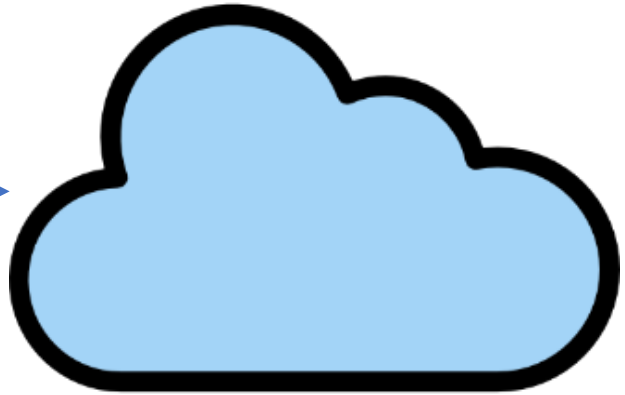
ใบงานที่ 4.1 ทฤษฎีเบื้องต้น

องค์ประกอบของ IoT



I-Kit V.2

วิธีในการส่งข้อมูล/
เส้นทางในการส่งข้อมูล



Server/Cloud

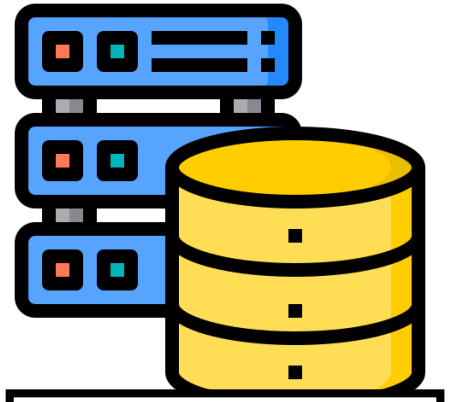
ข้อมูลที่ I-Kit ต้องการส่งไปยัง Cloud

"Temperature" = 25 °C

"Humidity" = 80 Lux

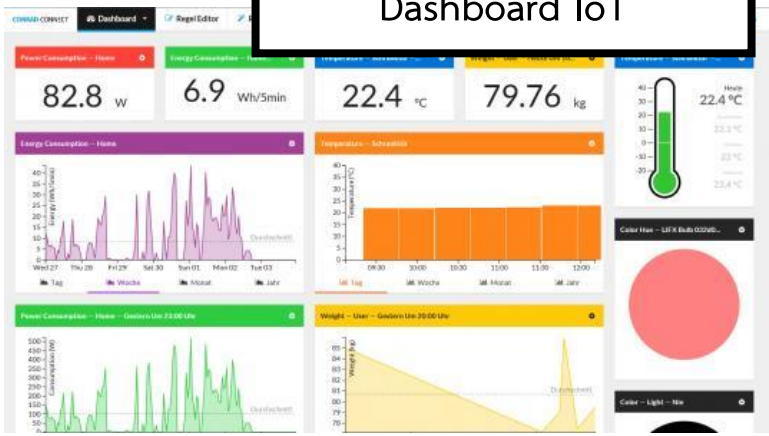
"Light" = 80 Lux

"LED1" = "ON"



Data Storage

Dashboard IoT



ใบงานที่ 4.1 ทฤษฎีเบื้องต้น

องค์ประกอบของ IoT

End Nodes

Connectivity

Central Servers/Cloud

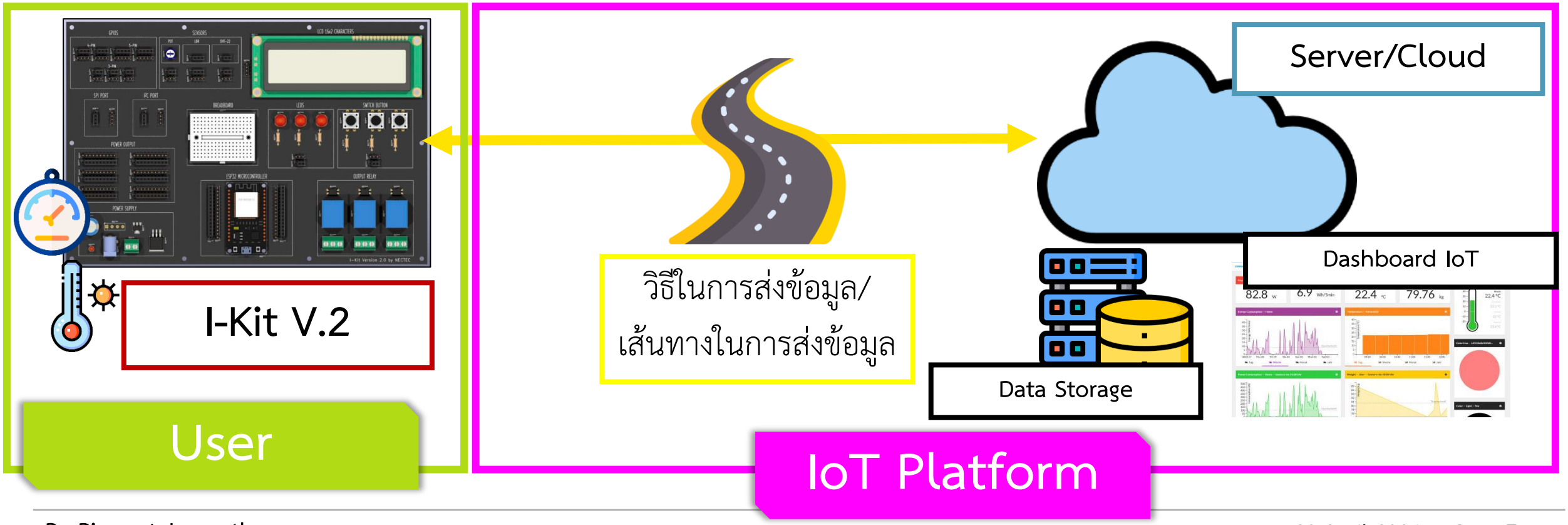
NETPIE
where things chat

ด้วยองค์ประกอบจำนวนมาก และการบริหารจัดการ Server/Cloud ที่ยากจึงเกิดแนวคิดเรื่อง IoT Platform ขึ้นมา

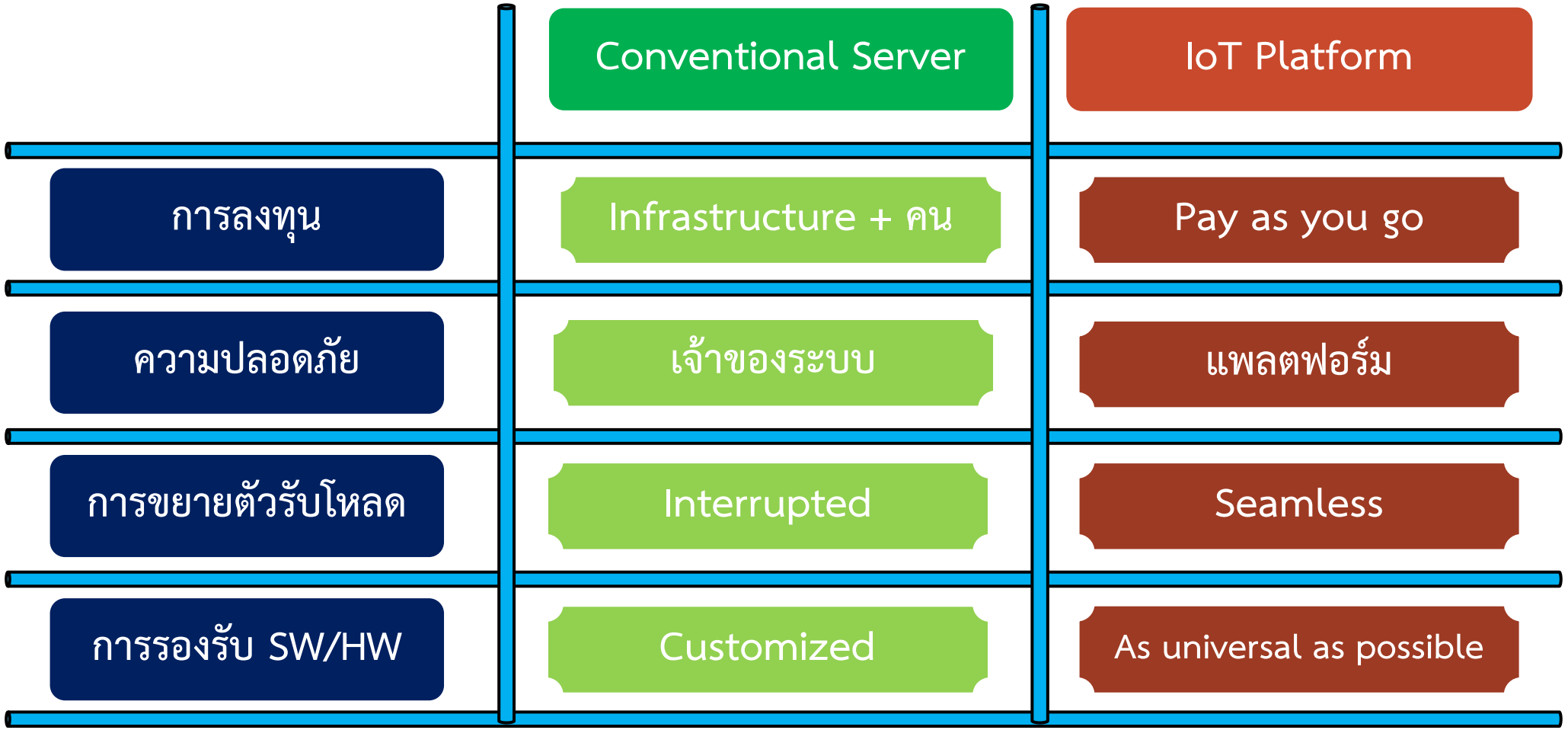
ใบงานที่ 4.1 ทฤษฎีเบื้องต้น

IoT Platform

IoT Platform คือ แพลตฟอร์มที่ช่วยให้ผู้ใช้งานนำเซนเซอร์หรืออุปกรณ์เชื่อมต่อกันได้ โดยมีการจัดการ Protocol, จัดเก็บข้อมูล, รักษาความปลอดภัยของข้อมูล เป็นต้น



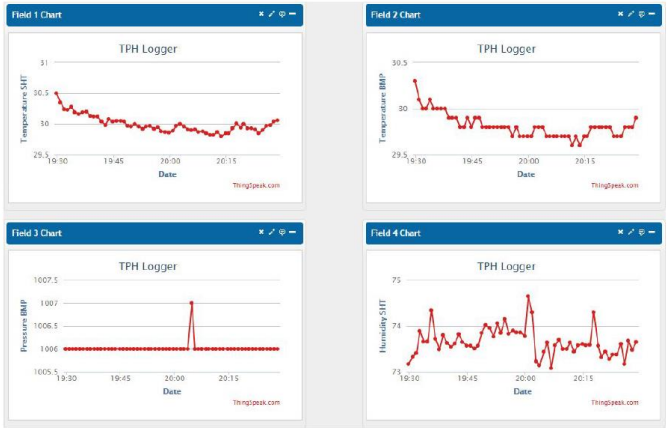
เปรียบเทียบ IoT Platform และการทำ Server ด้วยตัวเอง



ใบงานที่ 4.1 ทฤษฎีเบื้องต้น

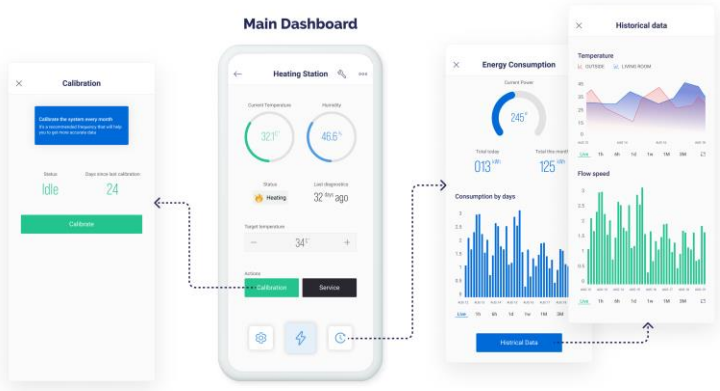
IoT Platform ที่นิยมในประเทศไทยและฟรี

ThingSpeak

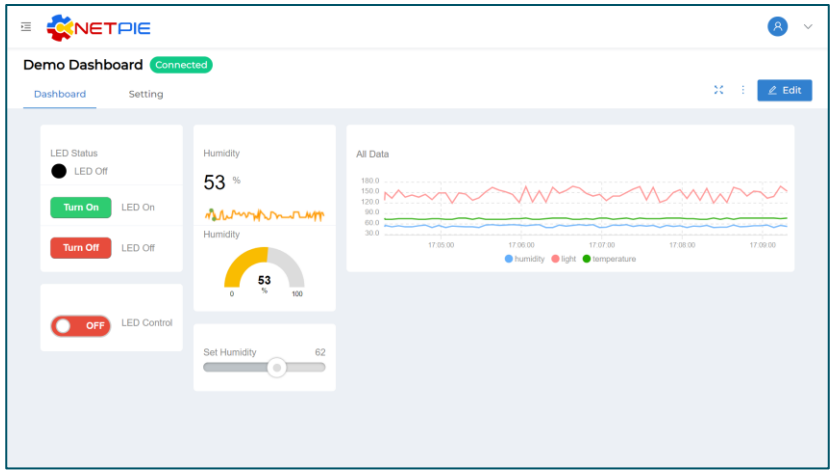


ThingsBoard

Blynk



NETPIE
where things chat



ใบงานที่ 4.1 ทฤษฎีเบื้องต้น

IoT Platform ประเภท Commercial

Cumulocity IoT Platform

Microsoft Azure IoT Suite

Google Cloud's IoT Platform

AWS IoT Platform

Cisco IoT Cloud Connect

Oracle IoT Platform

IBM Watson IoT Platform

ใบงานที่ 4.1 ทฤษฎีเบื้องต้น

NETPIE คืออะไร ??

NETPIE คือ IoT Cloud Platform ที่เปิดให้บุคคลทั่วไปใช้งานได้ โดยแพลตฟอร์มจะช่วยให้อุปกรณ์ต่างๆ สามารถสื่อสารกันได้ เกิดการรับ – ส่งข้อมูลระหว่างอุปกรณ์แบบ real-time ทำให้ผู้ใช้งานทราบถึงข้อมูลของอุปกรณ์ ณ เวลานั้นๆ ไม่ว่าผู้ใช้งานจะอยู่ที่ไหนเวลาใดก็ตาม

NETPIE where things chat

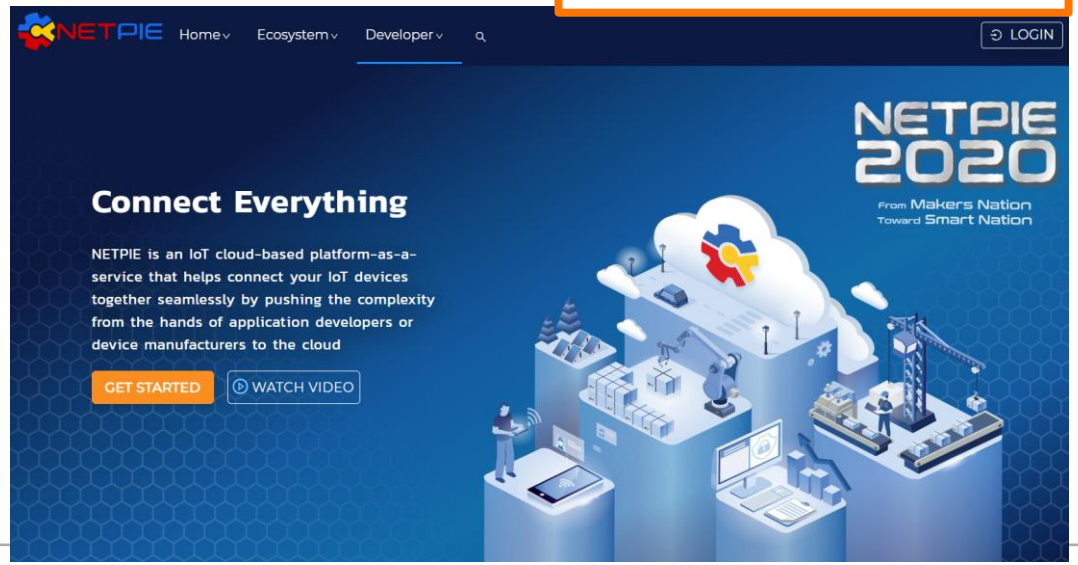
NETPIE 2015

2015.netpie.io

ปัจจุบัน NETPIE2015 ปิดให้บริการแล้ว

NETPIE 2020

netpie.io



ใบงานที่ 4.1 ทฤษฎีเบื้องต้น

คุณสมบัติหลักๆของ NETPIE

Monitoring

การแสดงค่าข้อมูลจากเซนเซอร์หรืออุปกรณ์แบบ Real-Time

Controlling

การควบคุมการทำงานของอุปกรณ์ต่างๆผ่าน Cloud Platform

Data Storage

การเก็บข้อมูลที่ได้จากเซ็นเซอร์หรืออุปกรณ์

Dashboard for monitor & Control

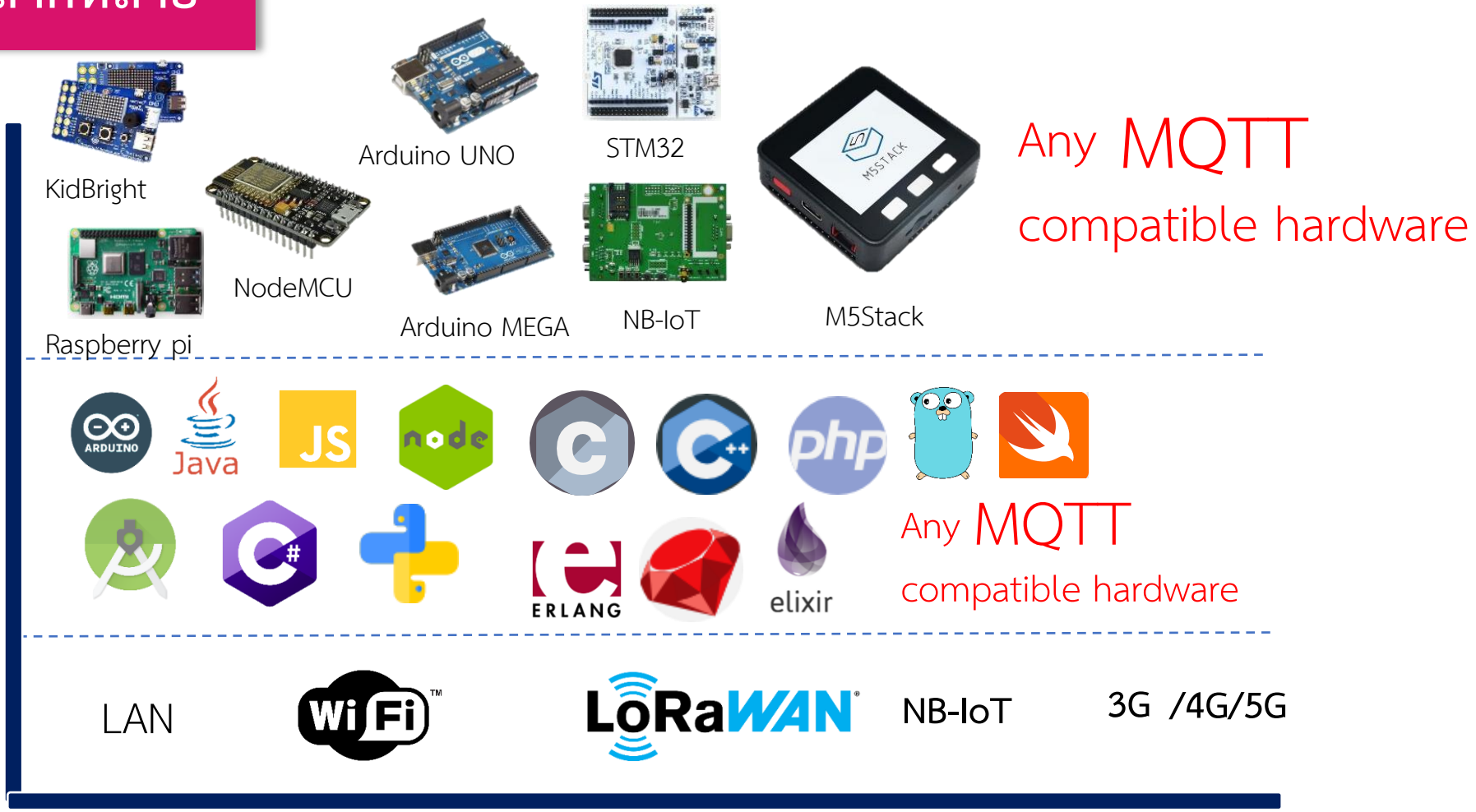
การแสดงผลและควบคุมการทำงานผ่าน Dashboard

Notification

การแจ้งเตือนความผิดปกติของเซนเซอร์หรืออุปกรณ์

ใบงานที่ 4.1 ทฤษฎีเบื้องต้น

รองรับอุปกรณ์ที่หลากหลาย



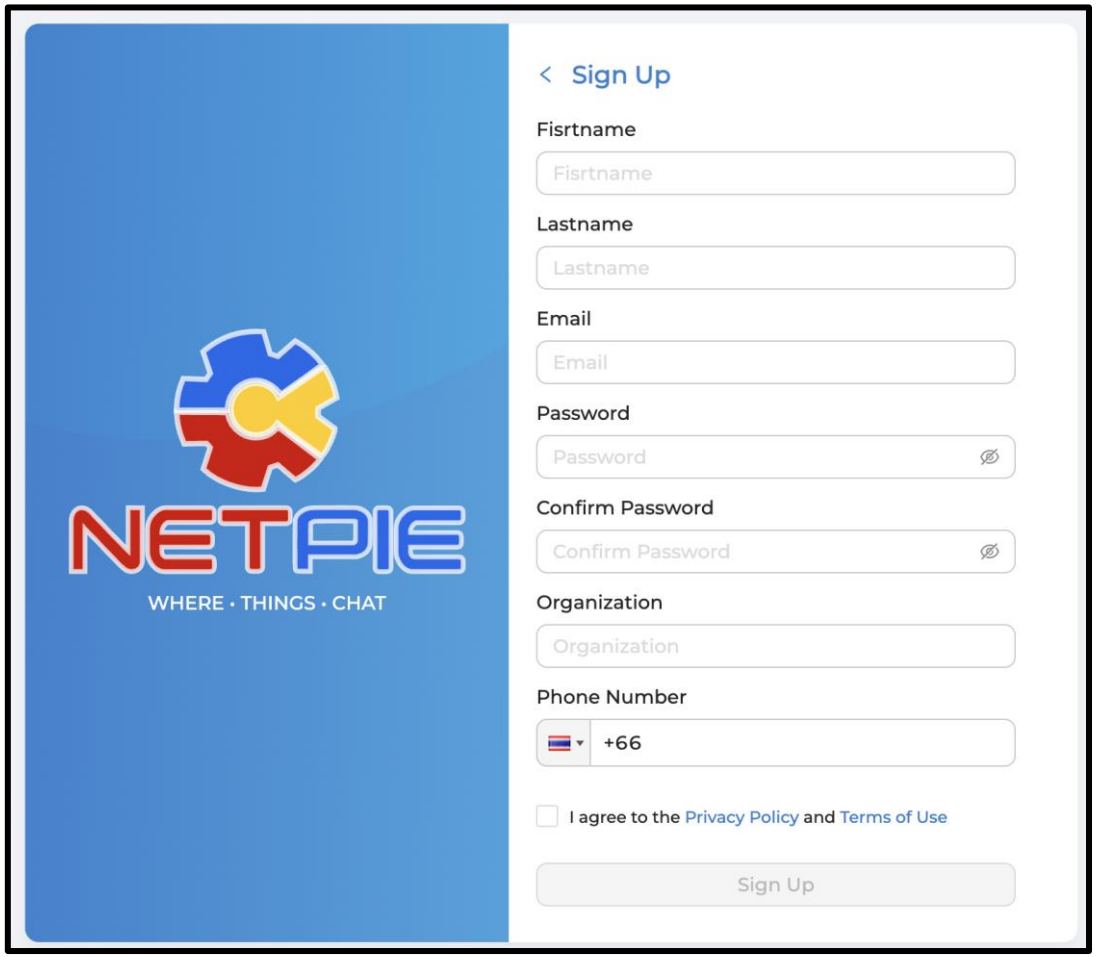
ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device

เริ่มต้นการใช้งาน NETPIE2020

ลงทะเบียน NETPIE2020 ได้ที่

https://auth.netpie.io/signup



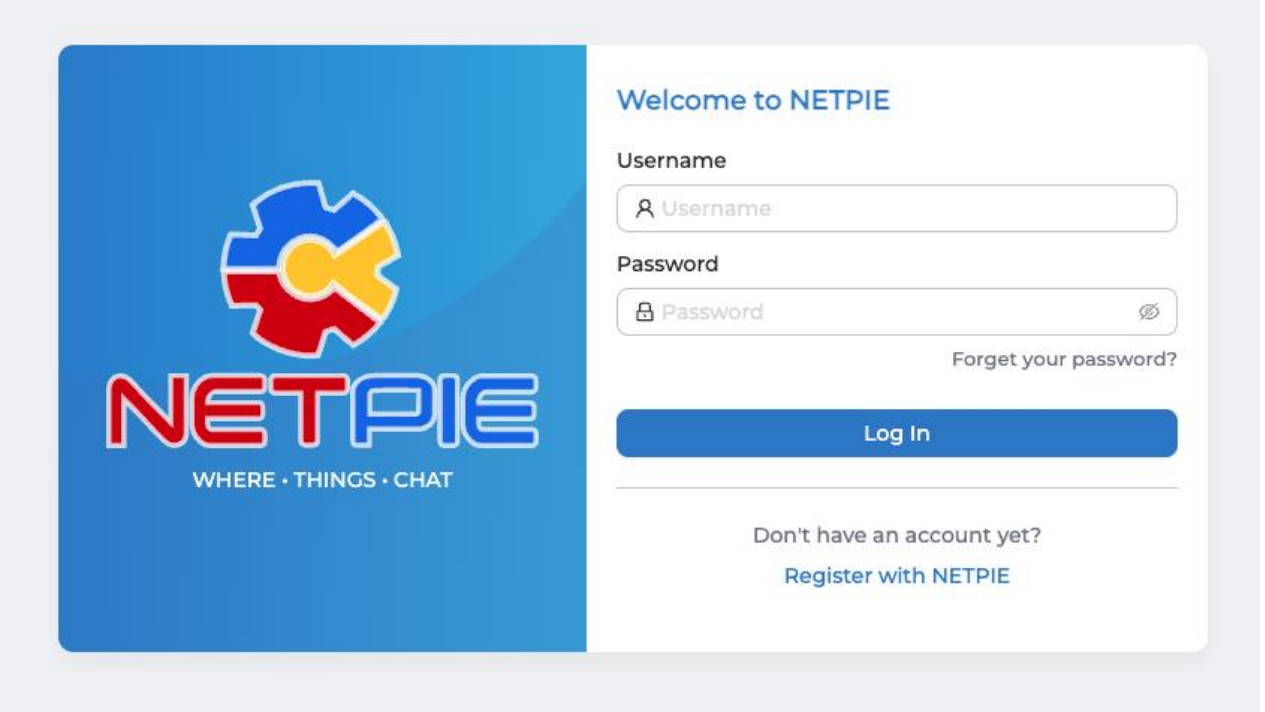
ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device

เมื่อสมัคร NETPIE2020 แล้วทำการเข้าสู่ระบบ

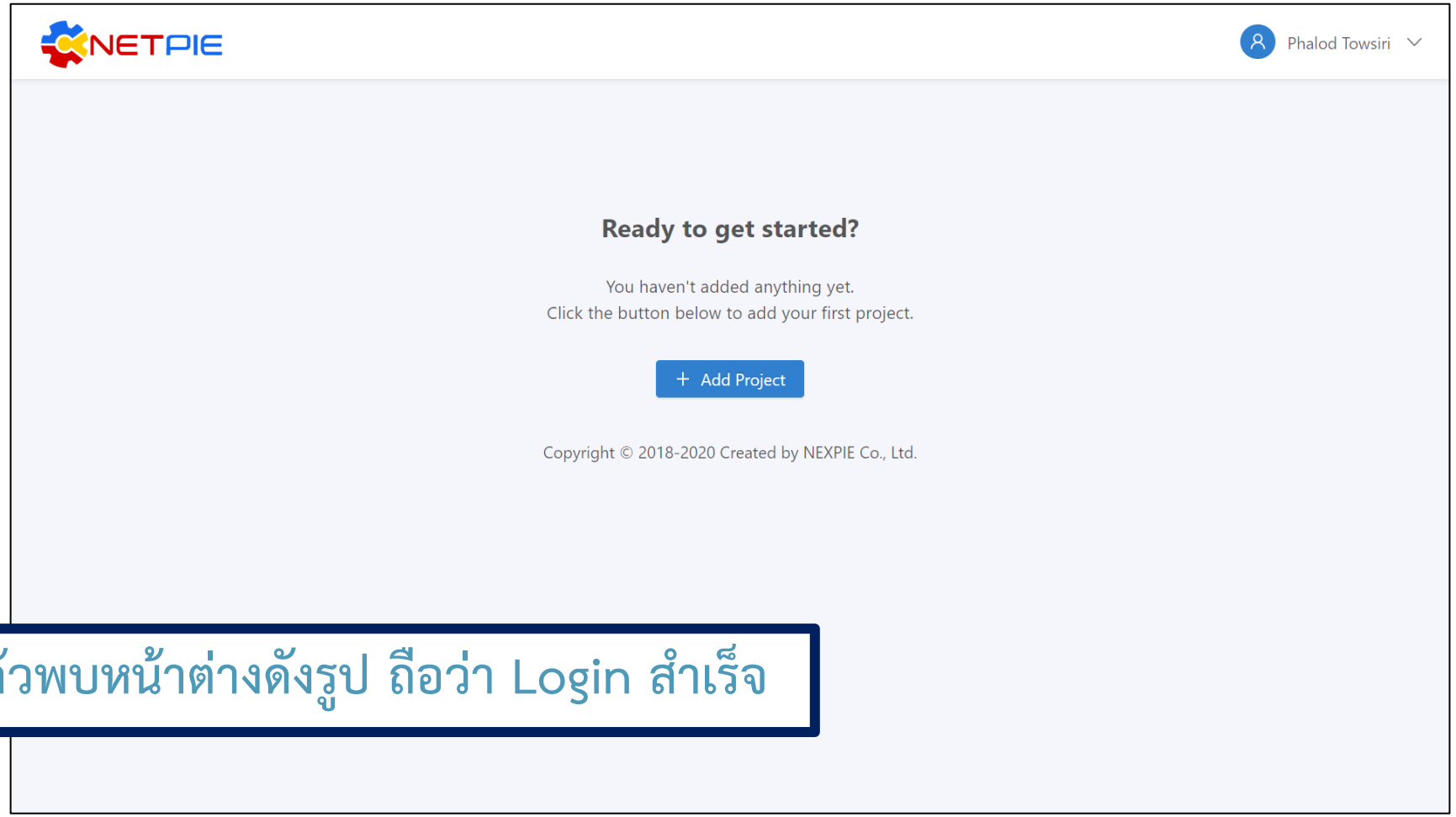
ซึ่งในปกติสามารถเข้าหน้า login ได้ที่

https://auth.netpie.io/login



ใบงานที่ 4.1 ขั้นตอนการทดลอง

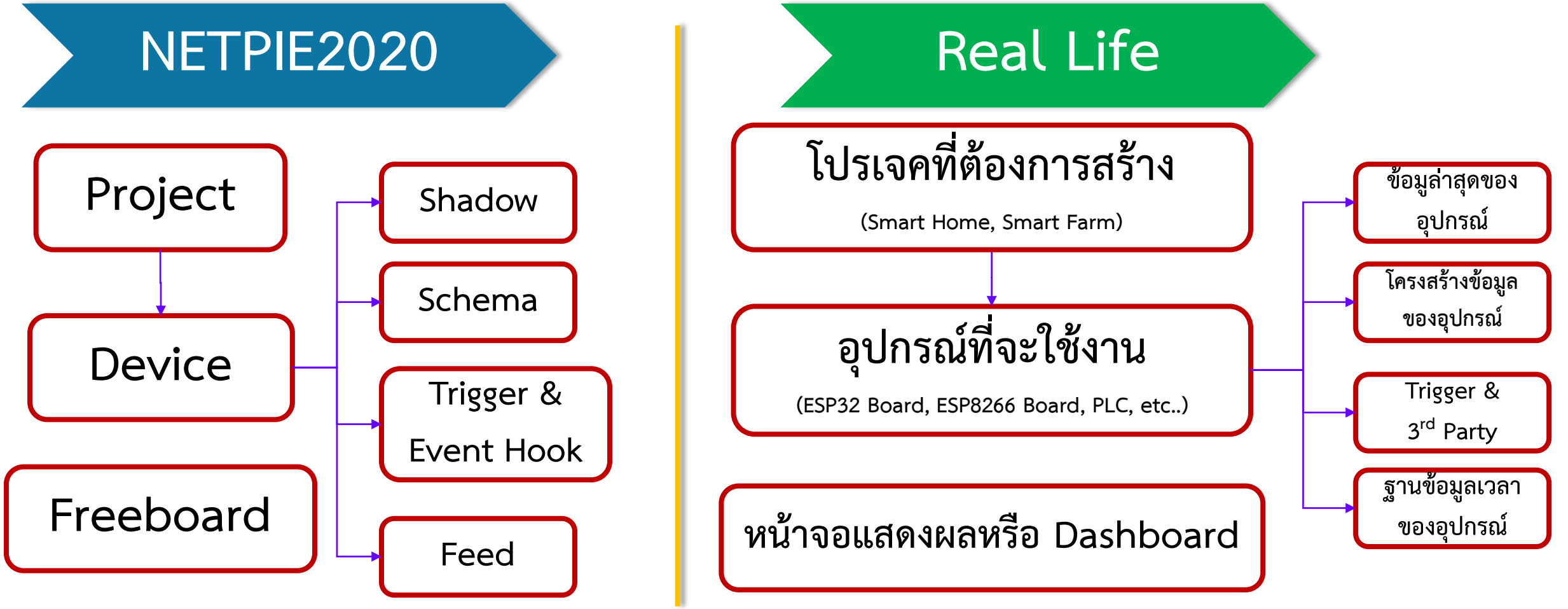
การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device



เมื่อเข้ามาแล้วพบหน้าต่างดังรูป ถือว่า Login สำเร็จ

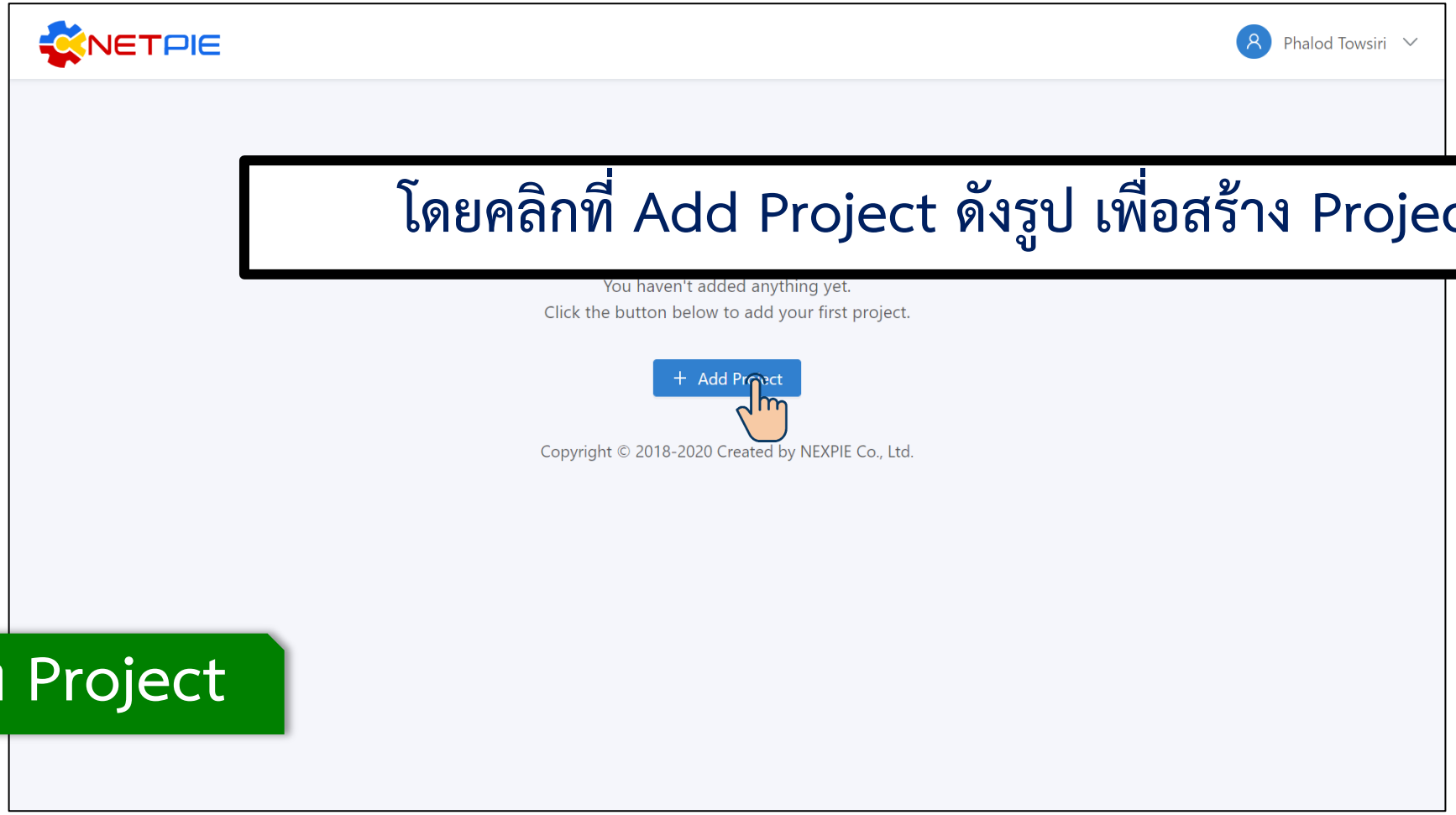
ใบงานที่ 4.1 ขั้นตอนการทดลอง

โครงสร้างของ NETPIE2020



ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device

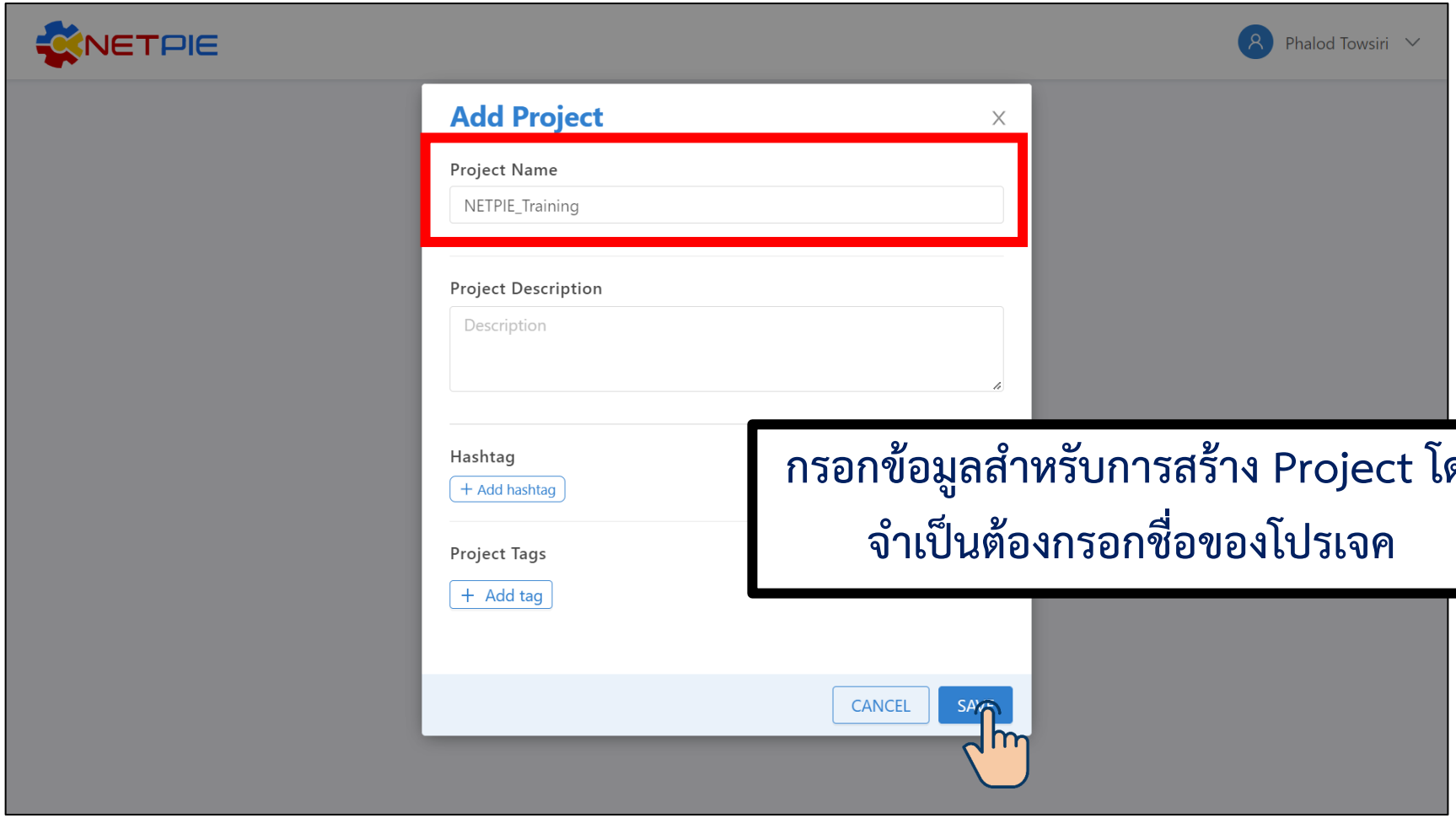


โดยคลิกที่ Add Project ดังรูป เพื่อสร้าง Project

การสร้าง Project

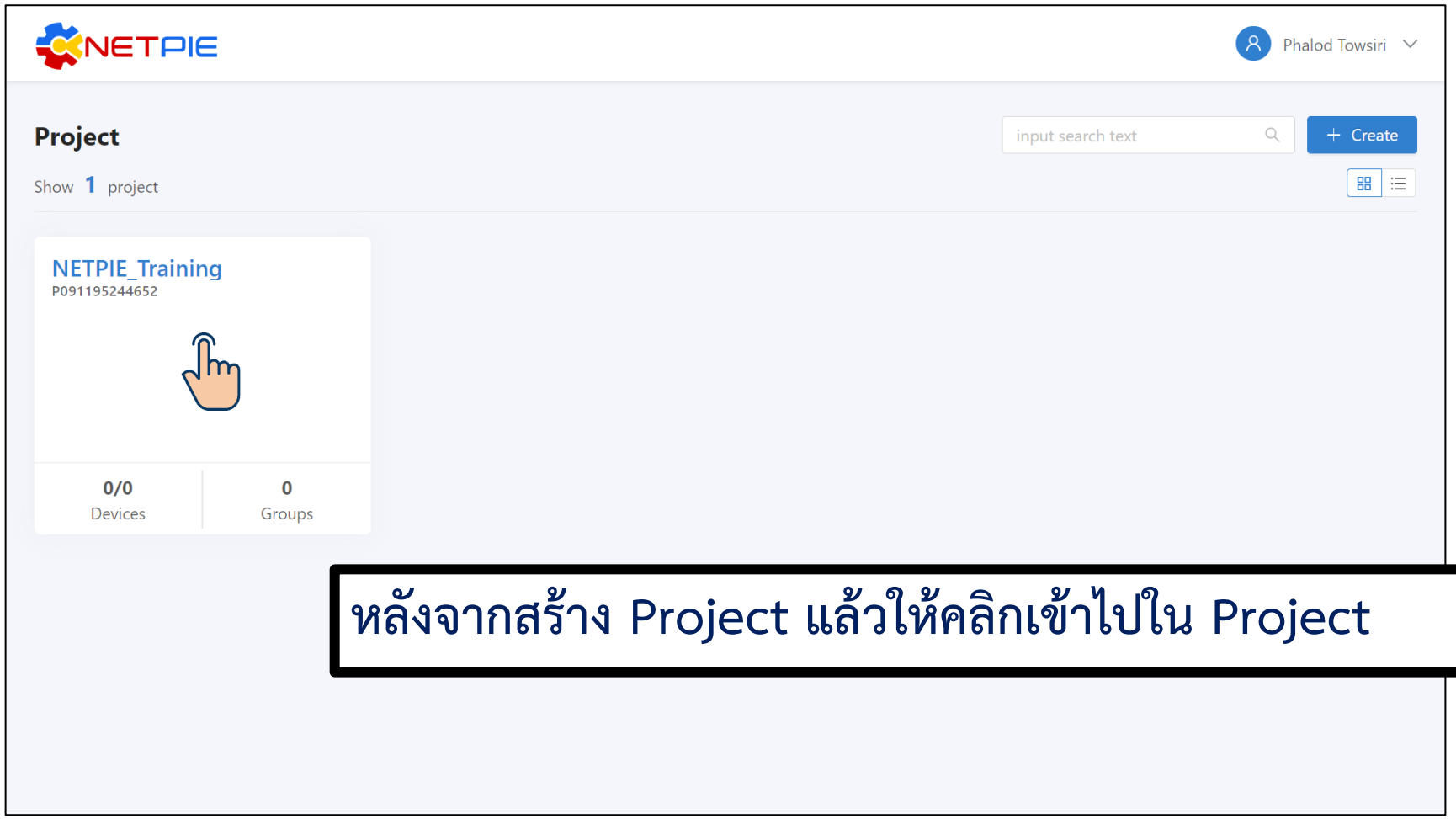
ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device



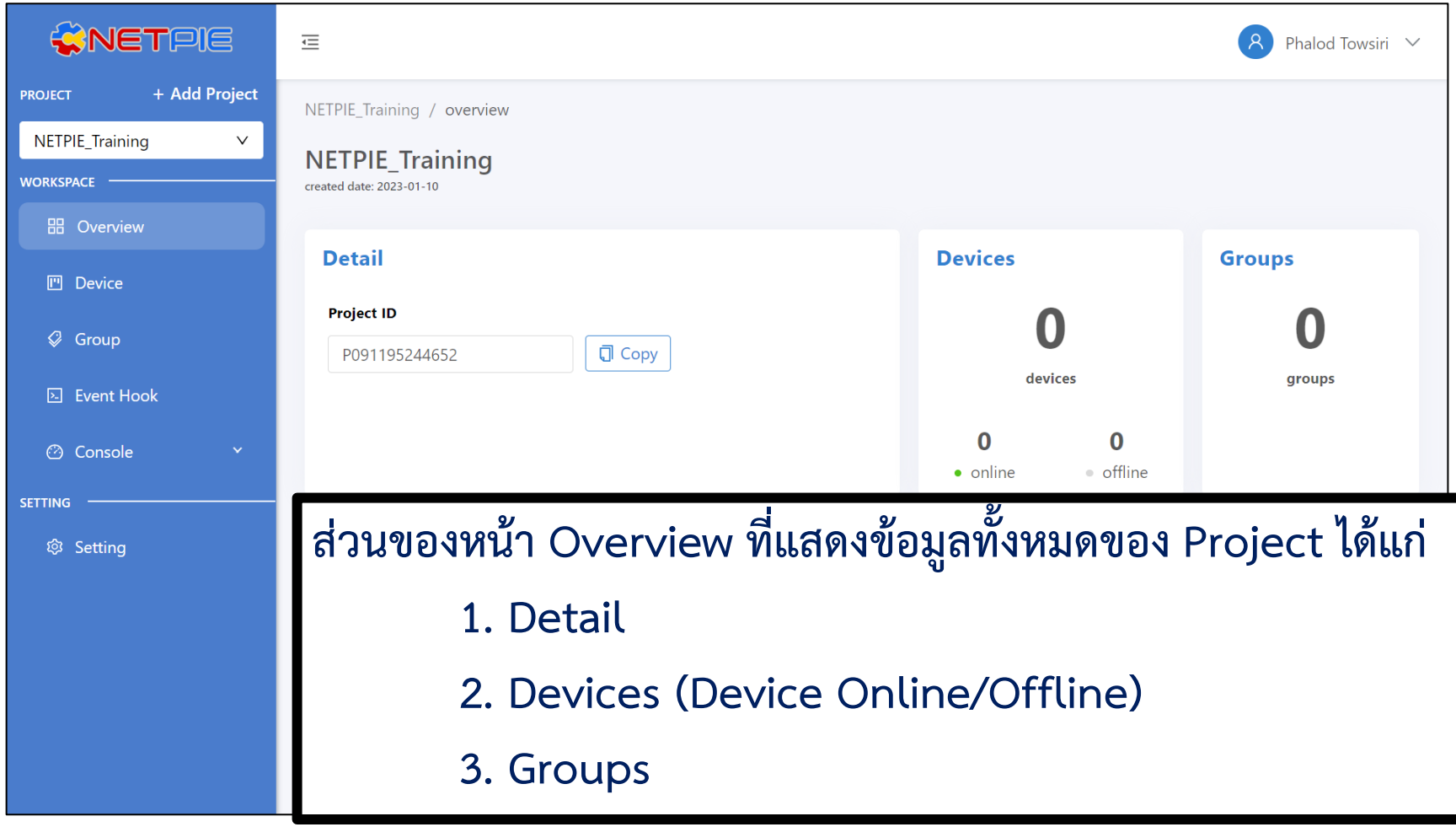
ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device



ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device

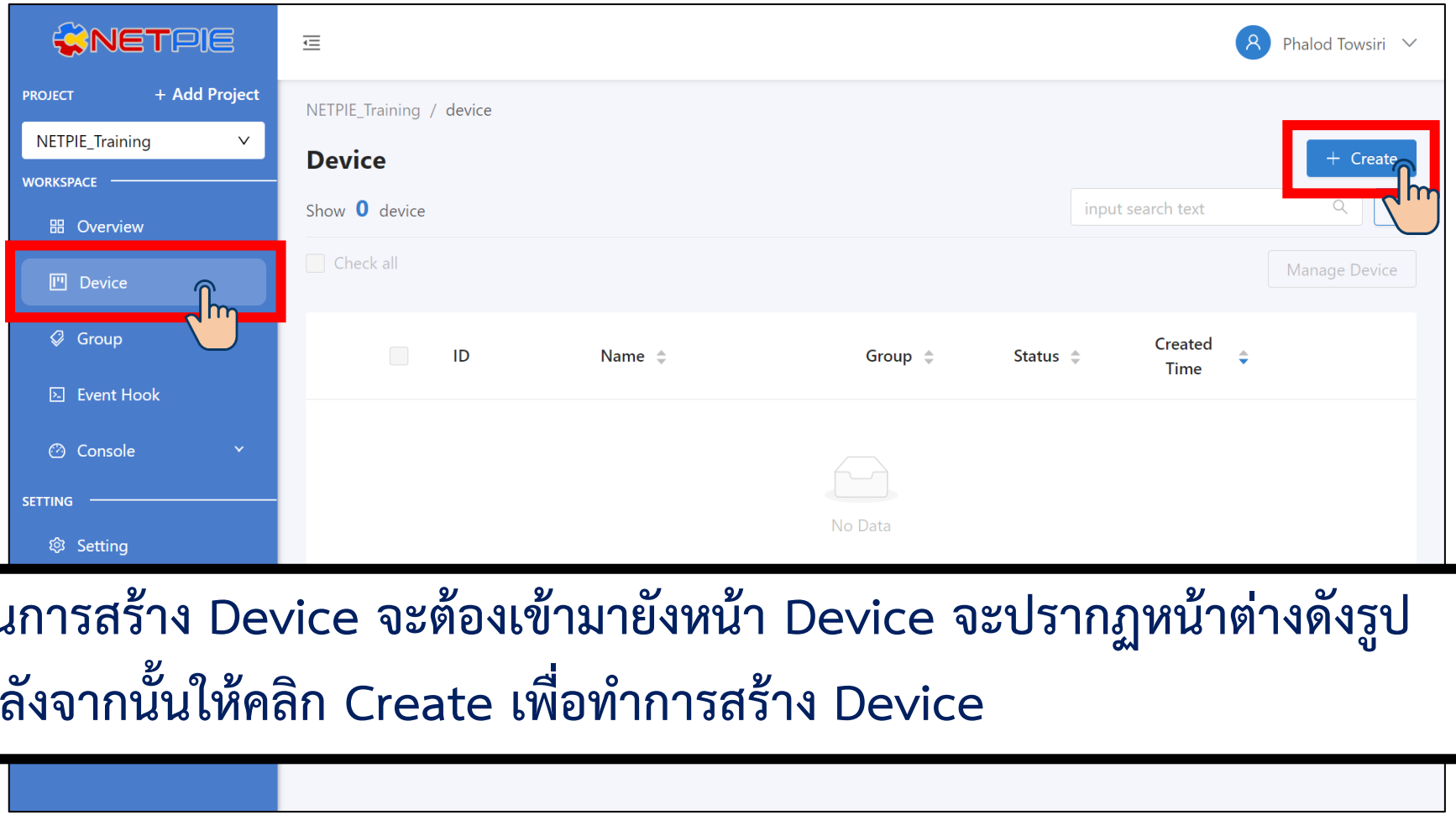


ส่วนของหน้า Overview ที่แสดงข้อมูลทั้งหมดของ Project ได้แก่

1. Detail
2. Devices (Device Online/Offline)
3. Groups

ใบงานที่ 4.1 ขั้นตอนการทดลอง

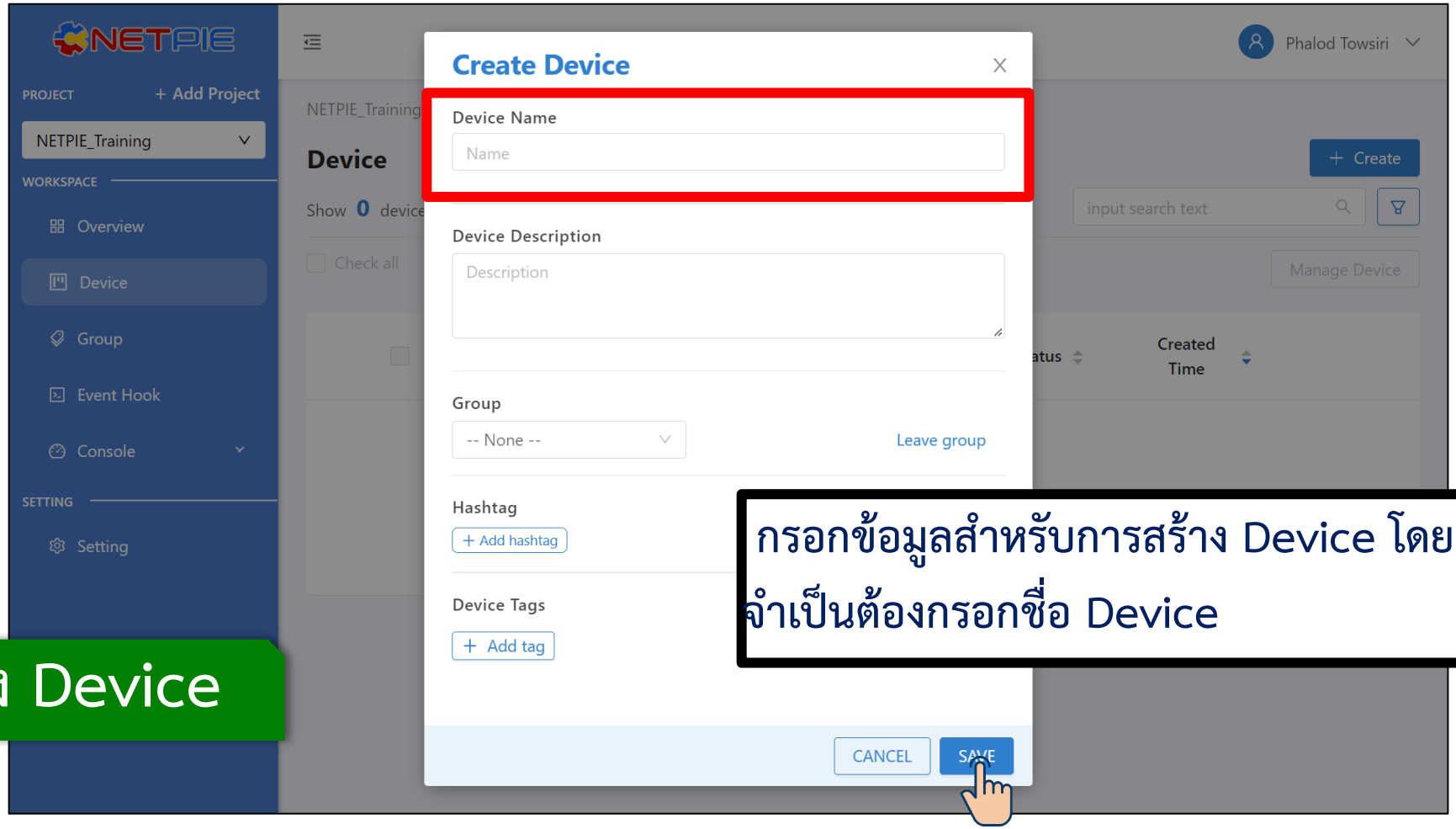
การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device



ในการสร้าง Device จะต้องเข้ามายังหน้า Device จะปรากฏหน้าต่างดังรูป หลังจากนั้นให้คลิก Create เพื่อทำการสร้าง Device

ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device

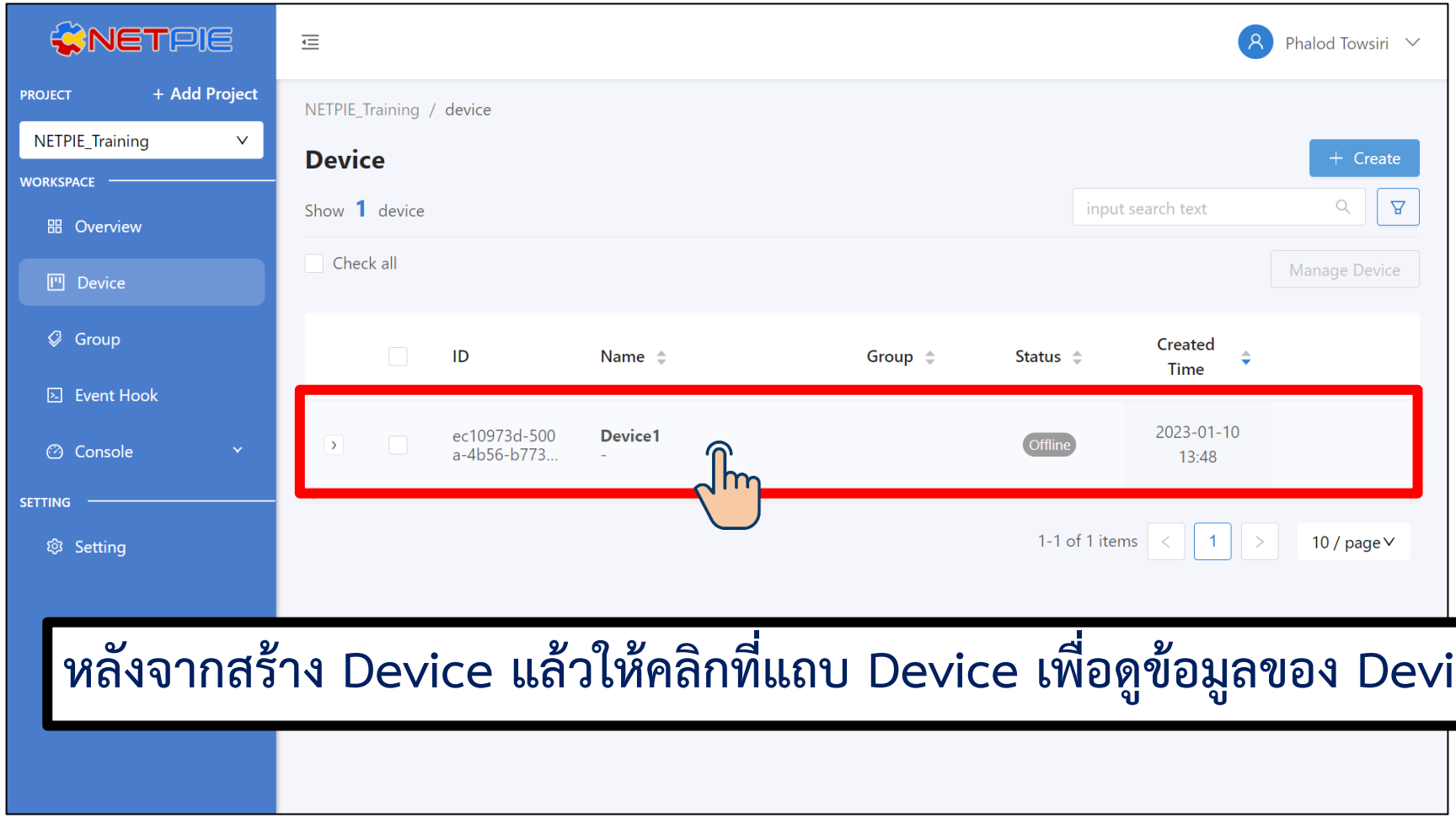


การสร้าง Device

กรอกข้อมูลสำหรับการสร้าง Device โดยจำเป็นต้องกรอกชื่อ Device

ใบงานที่ 4.1 ขั้นตอนการทดลอง

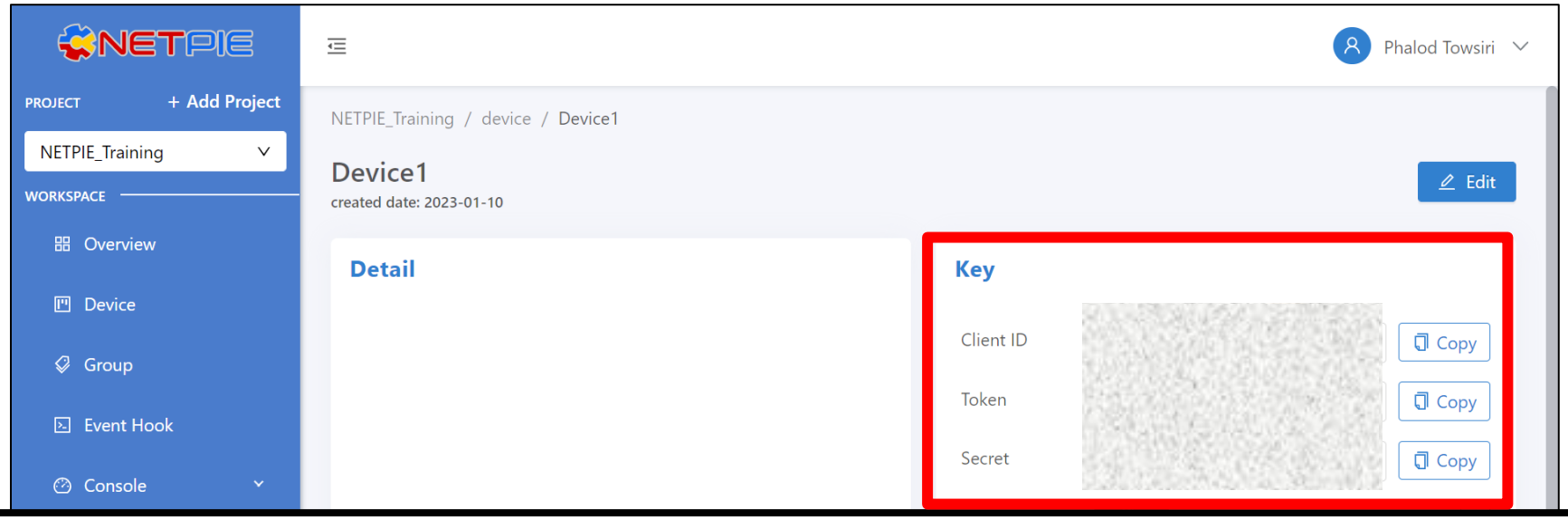
การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device



หลังจากสร้าง Device แล้วให้คลิกที่แถบ Device เพื่อดูข้อมูลของ Device

ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสมัครเข้าใช้งาน NETPIE2020 และการสร้าง Project & Device



หลังจากเข้ามายัง Device จะพบรายละเอียดต่างๆของ Device นั่นคือ

1. Client ID
2. Token
3. Secret

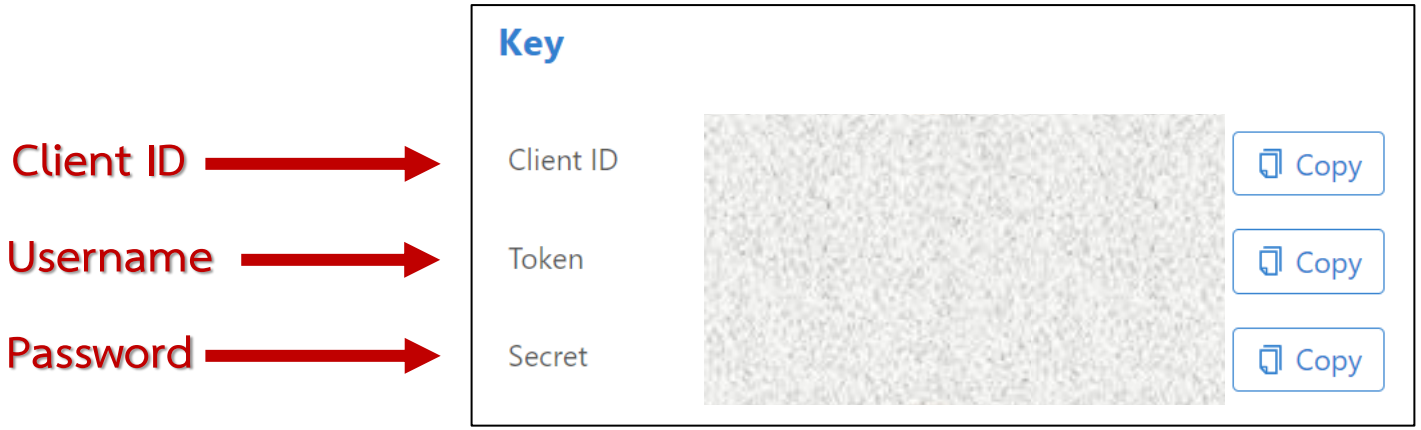
ซึ่งทั้ง 3 Parameter นี้มีส่วนสำคัญในการนำอุปกรณ์ต่อกับ NETPIE ด้วย Protocol ต่างๆ

ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบเชื่อมต่อ NETPIE2020 ด้วย MQTT Box

การเชื่อมต่อ NETPIE จะต้องใช้ 5 Parameters คือ

1. Host คือ mqtt.netpie.io หรือ broker.netpie.io
2. Client ID คือ Client ID ของ Device ที่สร้างขึ้นใน NETPIE
3. Username คือ Token ของ Device ที่สร้างขึ้นใน NETPIE
4. Password (Optional) คือ Secret ของ Device ที่สร้างขึ้นใน NETPIE (ใช้สำหรับกรณีที่ต้องการตรวจสอบที่เพิ่มมากขึ้น)
5. ใช้ Port มาตรฐานของ MQTT คือ Port 1883 (ในกรณีต้องระบุ Port ให้กับอุปกรณ์)



ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบเชื่อมต่อ NETPIE2020 ด้วย MQTT Box

ทำการทดสอบเชื่อมต่อ Device1 บน NETPIE ด้วยการใช้ MQTT Box

Connect

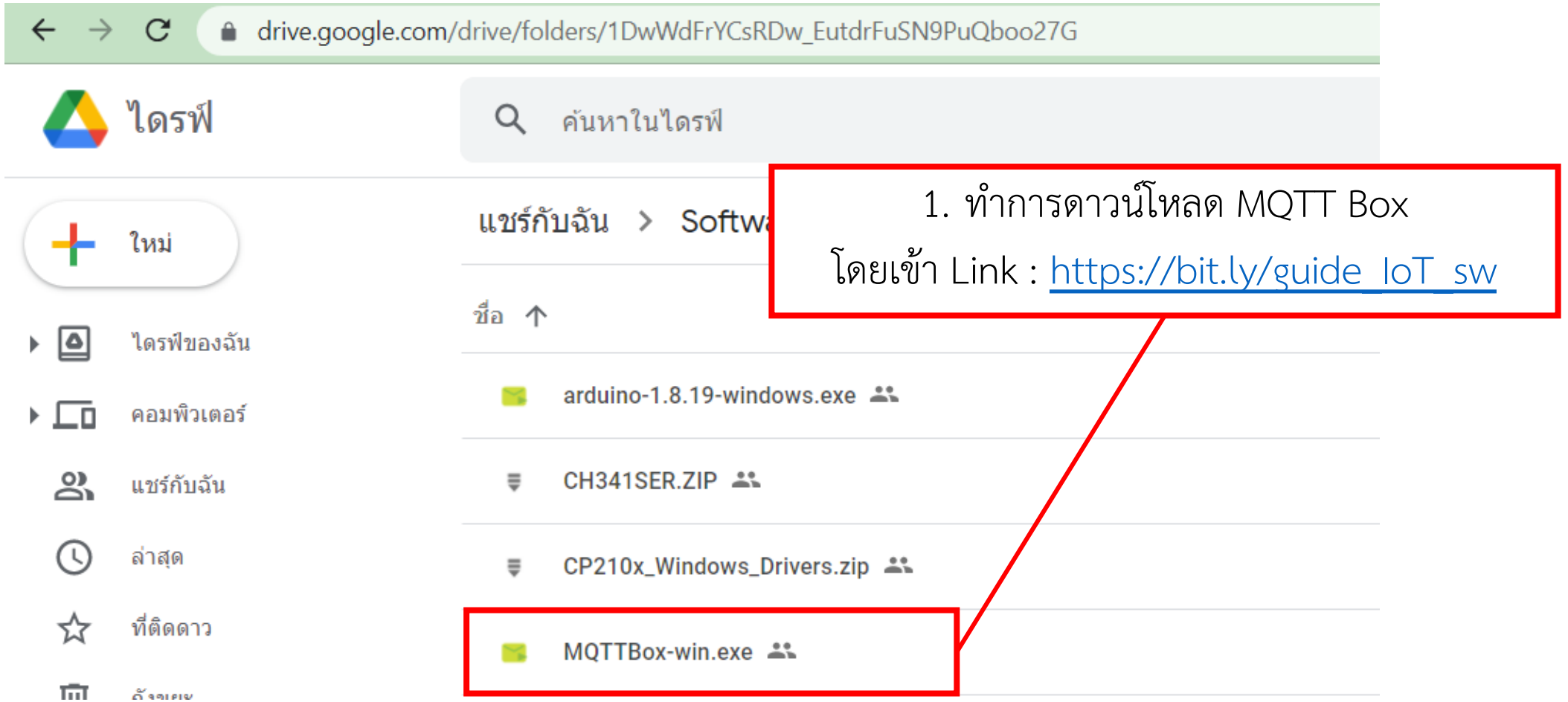
MQTT Box

NETPIE

MQTT เป็น Software ที่ใช้สำหรับเชื่อมต่อและสื่อสารผ่าน MQTT Protocol
โดยใน Example นี้เราใช้ MQTT Box แทนการใช้ ESP8266/ESP32

ใบงานที่ 4.1 ขั้นตอนการทดลอง

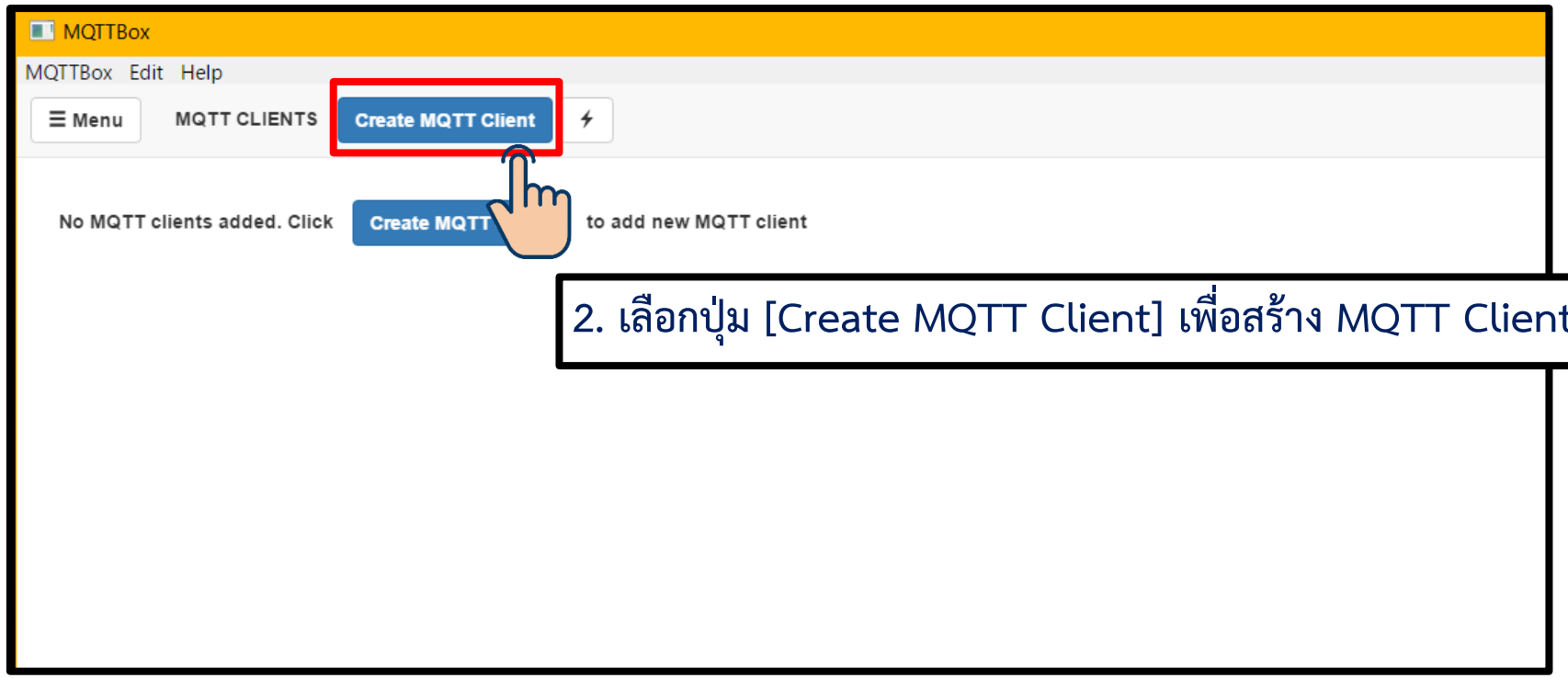
การทดลองที่ 2 ทดสอบเชื่อมต่อ NETPIE2020 ด้วย MQTT Box



ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบเชื่อมต่อ NETPIE2020 ด้วย MQTT Box

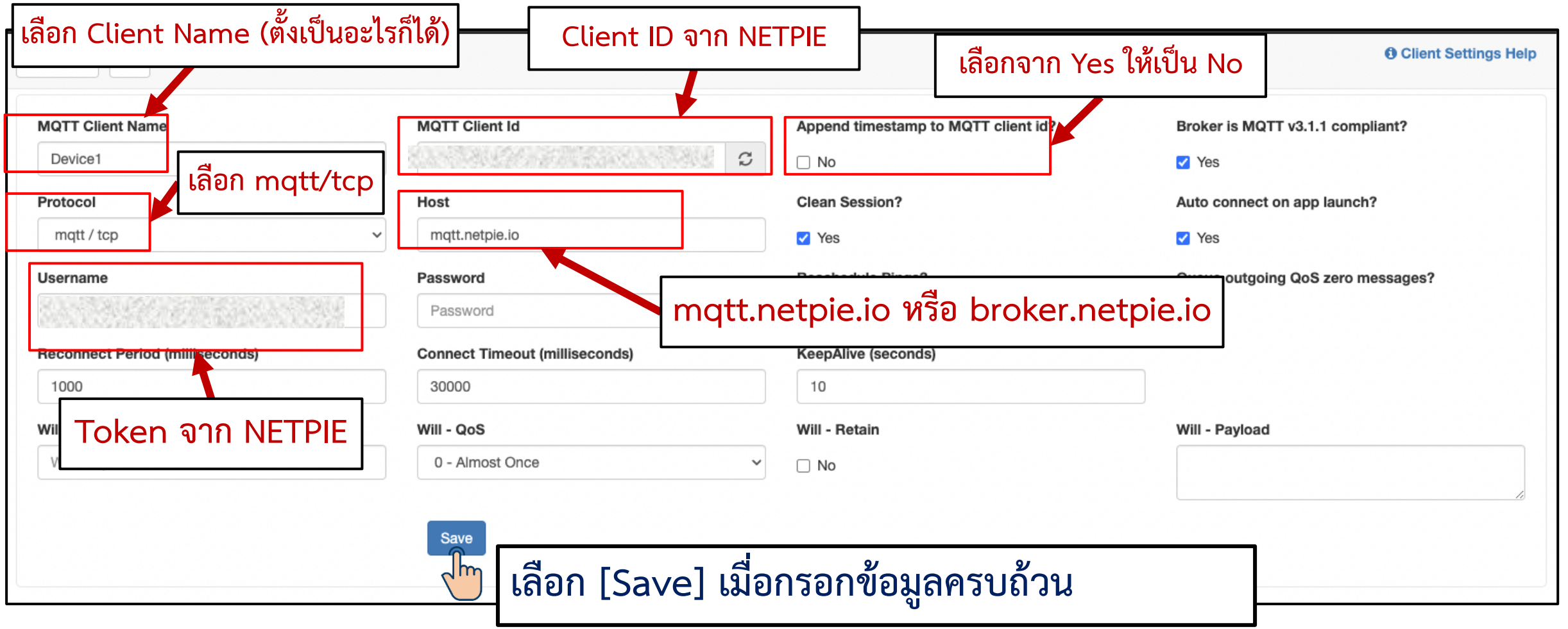
1. เมื่อติดตั้งเสร็จทำการเปิด MQTT Box ขึ้นมาจะได้หน้าต่างดังรูป



2. เลือกปุ่ม [Create MQTT Client] เพื่อสร้าง MQTT Client

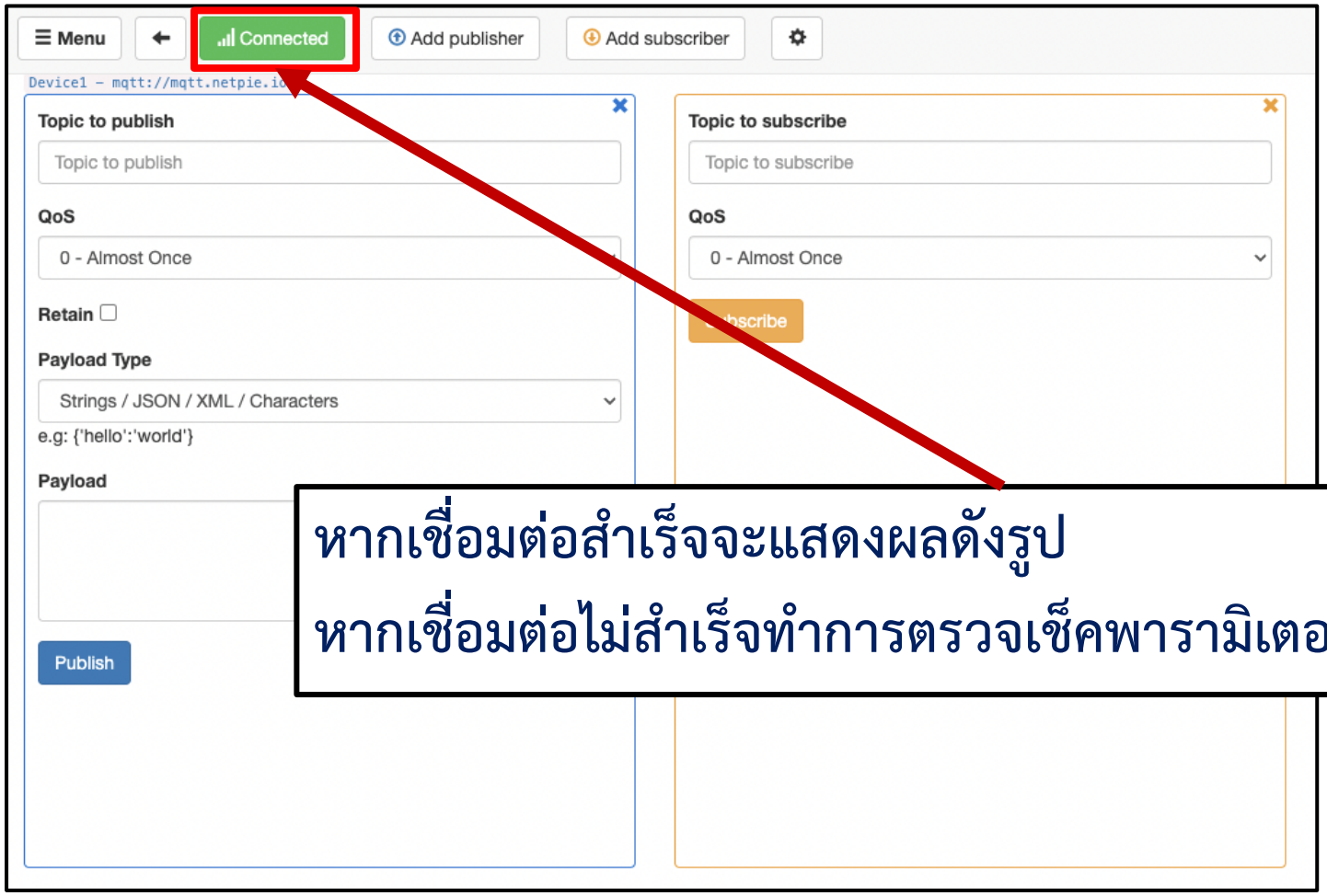
ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบเชื่อมต่อ NETPIE2020 ด้วย MQTT Box



ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบเชื่อมต่อ NETPIE2020 ด้วย MQTT Box

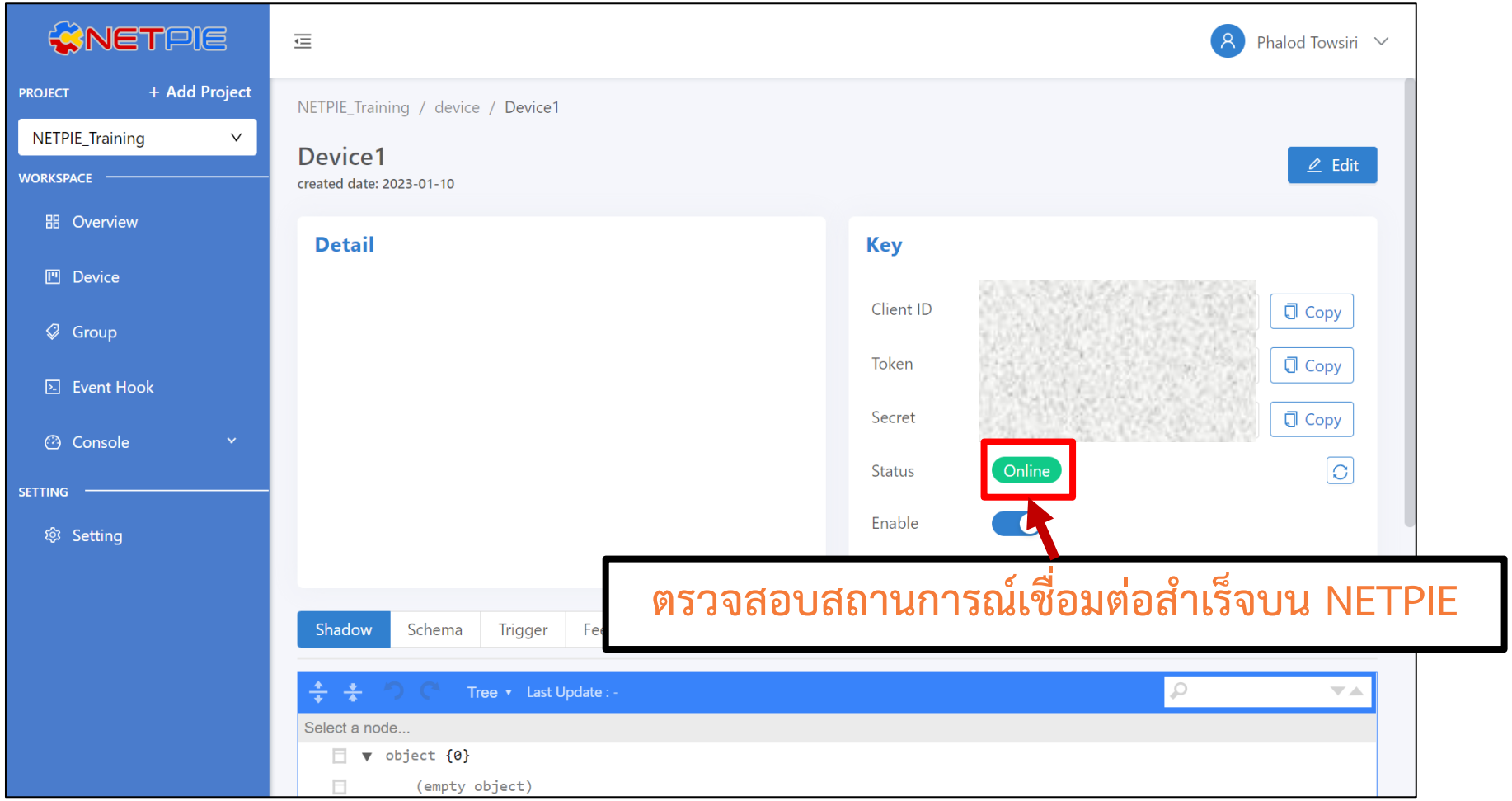


หากเชื่อมต่อสำเร็จจะแสดงผลดังรูป

หากเชื่อมต่อไม่สำเร็จทำการตรวจเช็คพารามิเตอร์ใหม่อีกครั้ง

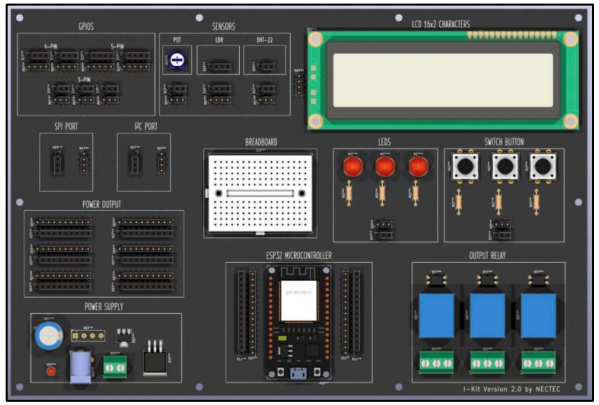
ใบงานที่ 4.1 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบเชื่อมต่อ NETPIE2020 ด้วย MQTT Box



ใบงานที่ 4.2 การเชื่อมต่อ และรับส่งข้อความผ่าน NETPIE2020 ด้วย ESP32 และ MQTT Box

NETPIE
where things chat



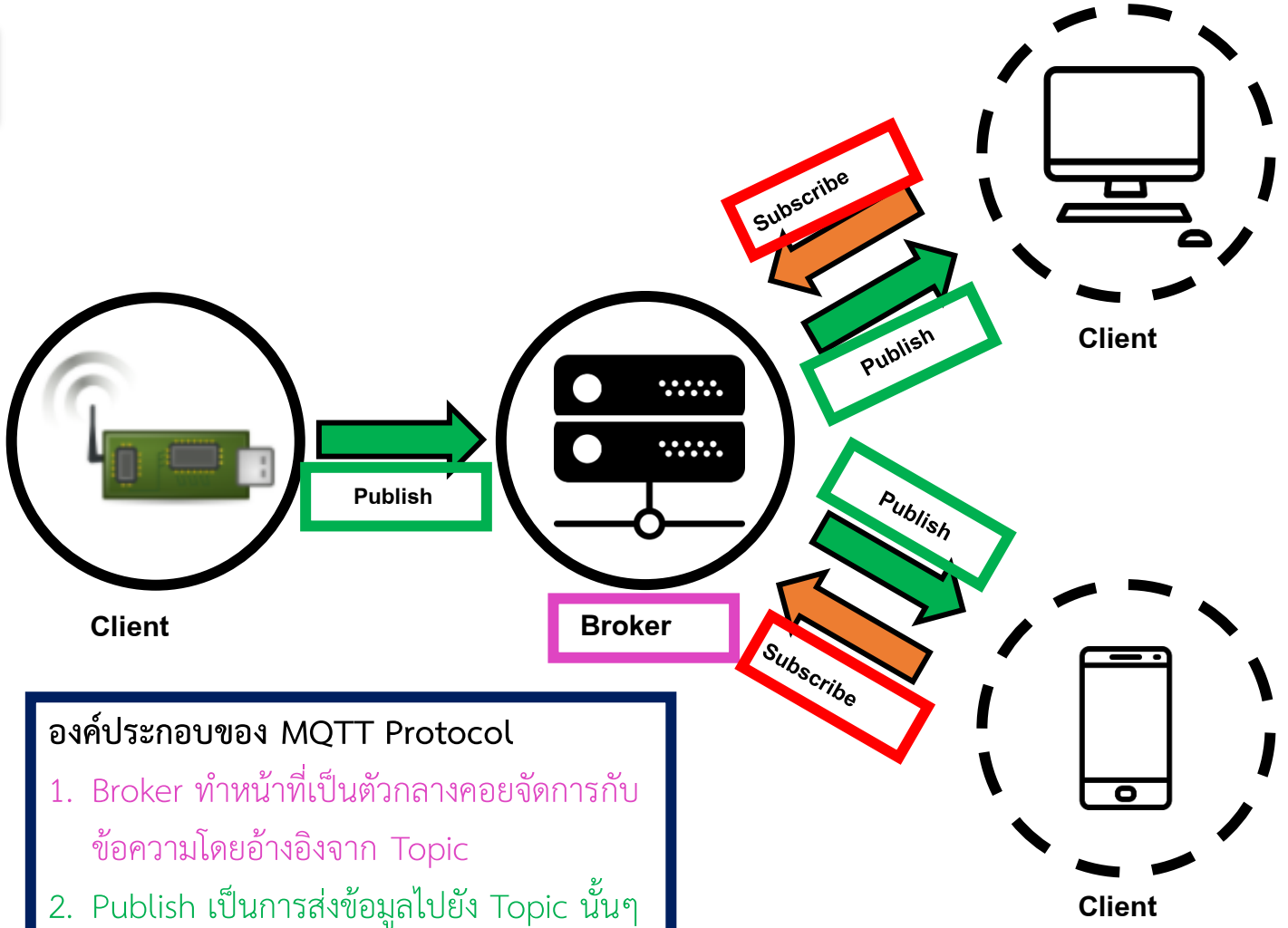
ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

MQTT Publish-Subscribe Model

- MQTT เป็น Protocol ที่ออกแบบมาเพื่อการเชื่อมต่อแบบ M2M หรือ อุปกรณ์กับอุปกรณ์ ซึ่งเป็นการสนับสนุน IoT
- ใช้หลักการการรับส่งข้อมูลแบบ Publish/Subscribe คล้ายกับหลักการที่ใช้ใน Web Service ที่ต้องใช้ Web Server เป็นตัวกลางระหว่างคอมพิวเตอร์ของผู้ใช้งาน
- แต่ MQTT ใช้ตัวกลางที่เรียกว่า Broker ทำหน้าที่จัดการลำดับการรับ – ส่ง ข้อมูลระหว่างอุปกรณ์และทั้งที่เป็น Publish/Subscibe

ข้อดีของ MQTT

- Asynchronous Communication
- Small Code Footprint
- Saving network bandwidth – small header



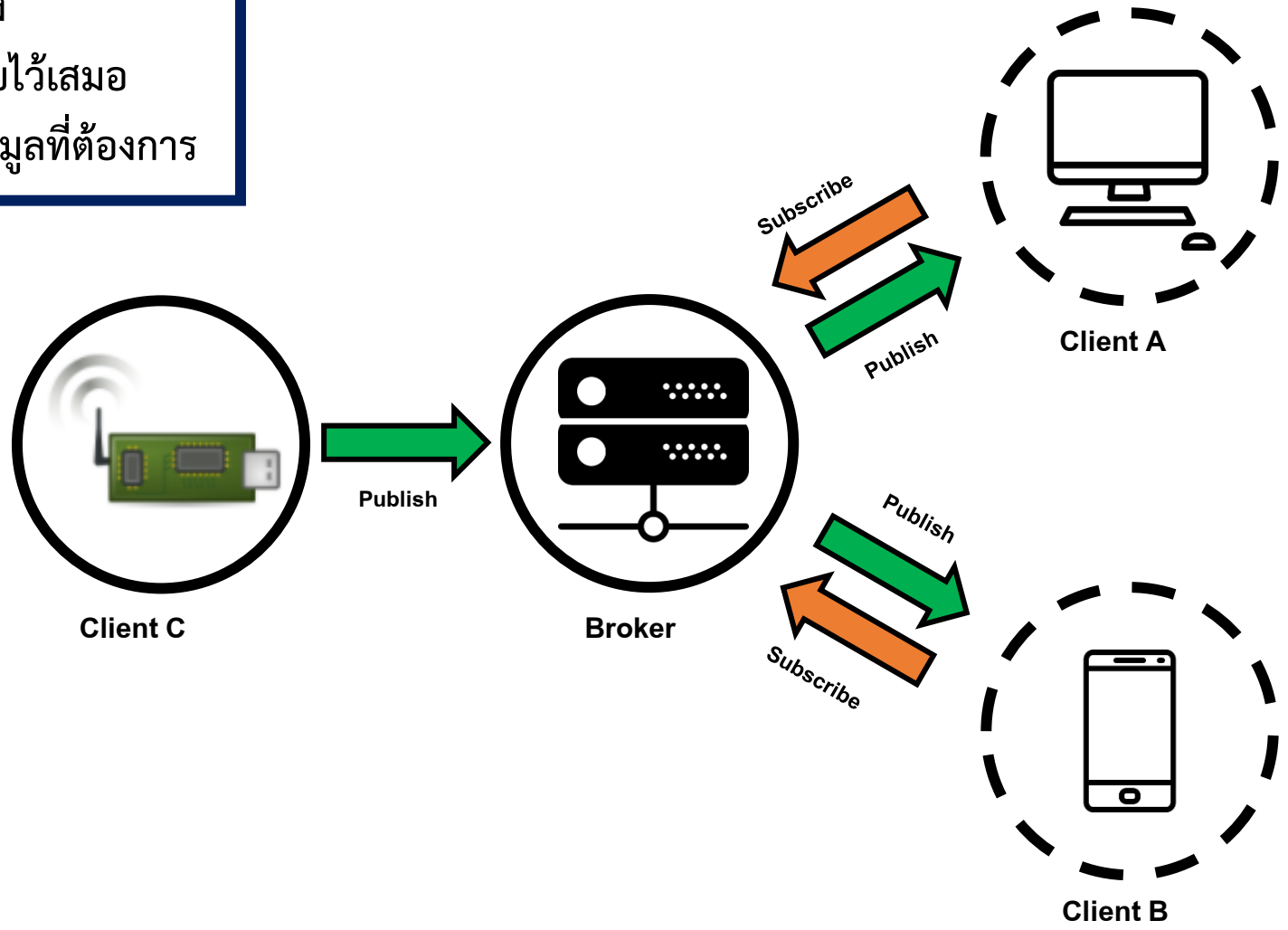
องค์ประกอบของ MQTT Protocol

1. Broker ทำหน้าที่เป็นตัวกลางคอยจัดการกับข้อความโดยอ้างอิงจาก Topic
2. Publish เป็นการส่งข้อมูลไปยัง Topic นั้นๆ
3. Subscribe เป็นการคอยดูการเปลี่ยนแปลง Message ที่อ้างอิงด้วย Topic

ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

การส่งข้อมูลบน MQTT จะมี Topic เป็นตัวอ้างอิงหลัก ซึ่ง

- ข้อมูลที่จะ Publish ไปยัง Broker ต้องมี Topic กำกับไว้เสมอ
- ส่วนของ Subscribe ก็ต้องอ้างอิง Topic เพื่อเรียกข้อมูลที่ต้องการ



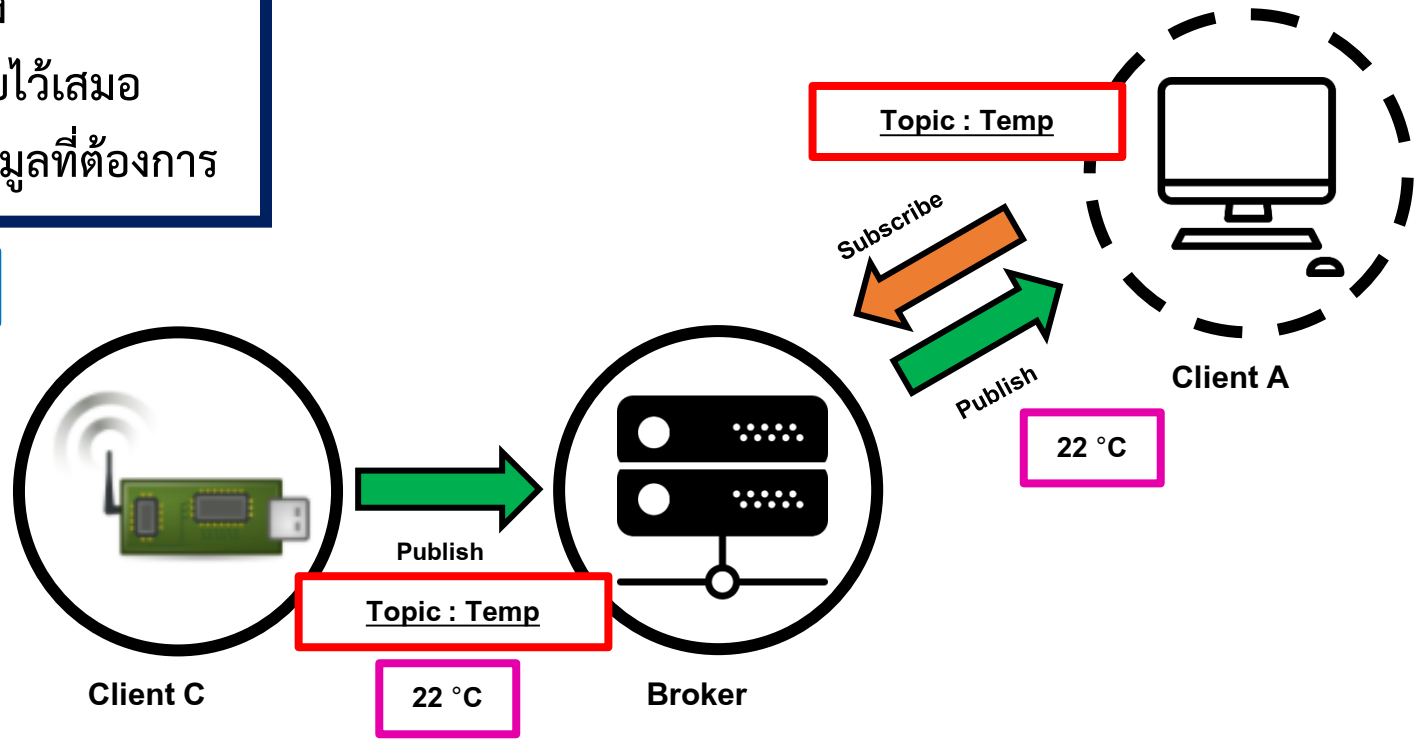
ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

การส่งข้อมูลบน MQTT จะมี Topic เป็นตัวอ้างอิงหลัก ซึ่ง

- ข้อมูลที่จะ Publish ไปยัง Broker ต้องมี Topic กำกับไว้เสมอ
- ส่วนของ Subscribe ก็ต้องอ้างอิง Topic เพื่อเรียกข้อมูลที่ต้องการ

ยกตัวอย่างการส่งข้อมูลแบบ MQTT

- Client C นั้น Publish ข้อมูลบน Topic ที่มีชื่อว่า Temp และมี Client A ที่ Subscribe Topic Temp ไว้อยู่
- โดยข้อมูลจาก Client C นั้นคือ อุณหภูมิ ที่มีค่าเท่ากับ 22 °C ดังนั้น Client A จะได้รับข้อมูลด้วย



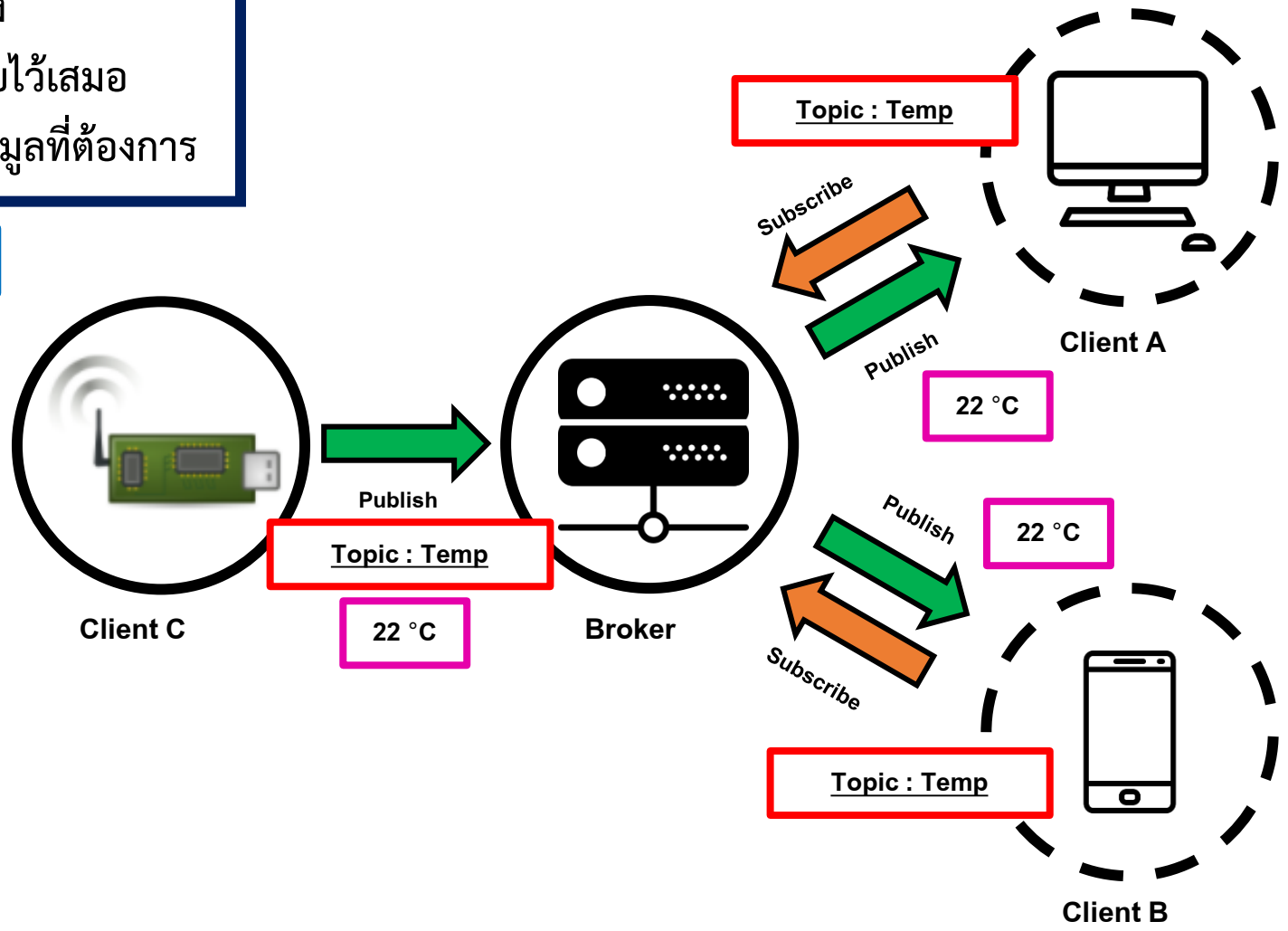
ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

การส่งข้อมูลบน MQTT จะมี Topic เป็นตัวอ้างอิงหลัก ซึ่ง

- ข้อมูลที่จะ Publish ไปยัง Broker ต้องมี Topic กำกับไว้เสมอ
- ส่วนของ Subscribe ก็ต้องอ้างอิง Topic เพื่อเรียกข้อมูลที่ต้องการ

ยกตัวอย่างการส่งข้อมูลแบบ MQTT

- Client C นั้น Publish ข้อมูลบน Topic ที่มีชื่อว่า Temp และมี Client A ที่ Subscribe Topic Temp ไว้อยู่
- โดยข้อมูลจาก Client C นั้นคือ อุณหภูมิ ที่มีค่าเท่ากับ 22 °C ดังนั้น Client A จะได้รับข้อมูลด้วย
- ภายหลัง Client B ทำการ Subscribe Topic Temp ด้วย ดังนั้นข้อมูล อุณหภูมิที่ถูกเก็บไว้จะถูกส่งไปยัง Client B ทันทีหลังจากทำการ Subscribe



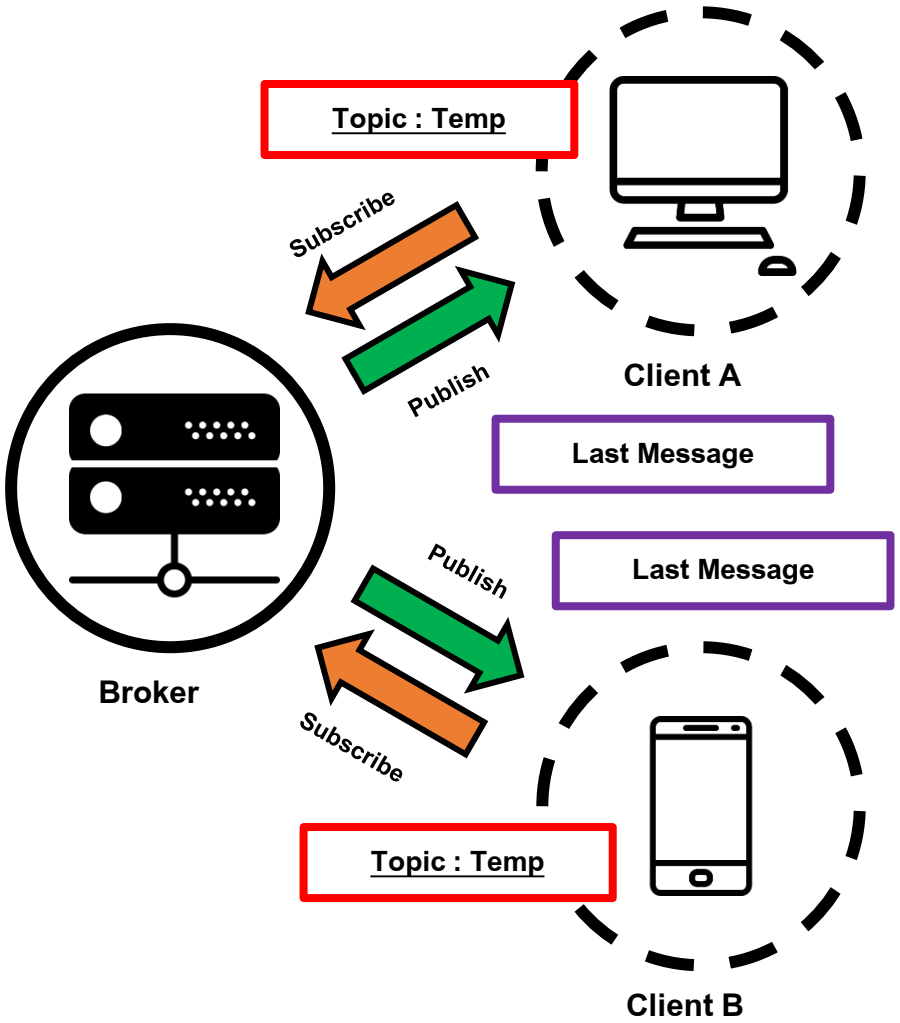
ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

การส่งข้อมูลบน MQTT จะมี Topic เป็นตัวอ้างอิงหลัก ซึ่ง

- ข้อมูลที่จะ Publish ไปยัง Broker ต้องมี Topic กำกับไว้เสมอ
- ส่วนของ Subscribe ก็ต้องอ้างอิง Topic เพื่อเรียกข้อมูลที่ต้องการ

ยกตัวอย่างการส่งข้อมูลแบบ MQTT

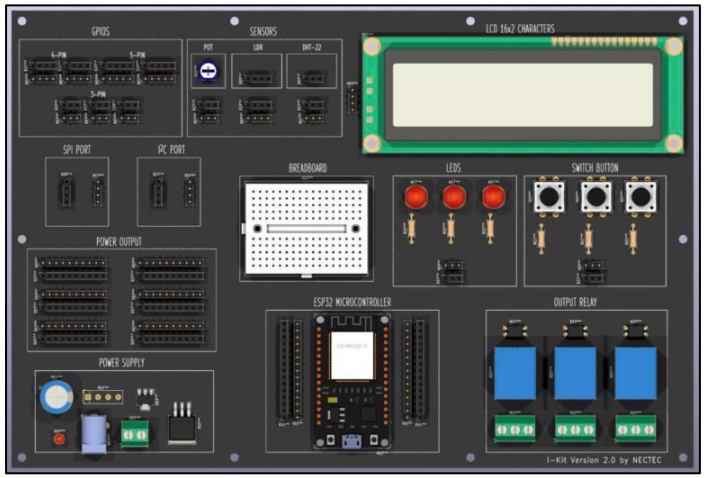
- Client C นั้น Publish ข้อมูลบน Topic ที่มีชื่อว่า Temp และมี Client A ที่ Subscribe Topic Temp ไว้อยู่
- โดยข้อมูลจาก Client C นั้นคือ อุณหภูมิ ที่มีค่าเท่ากับ 22 °C ดังนั้น Client A จะได้รับข้อมูลด้วย
- ภายหลัง Client B ทำการ Subscribe Topic Temp ด้วย ดังนั้นข้อมูล อุณหภูมิที่ถูกเก็บไว้จะถูกส่งไปยัง Client B ทันทีหลังจากทำการ Subscribe
- แต่เมื่อ Client C นั้นขาดการเชื่อมต่อทำให้ไม่มีข้อมูล Publish ไปยัง Broker สิ่งที่ Client A และ B จะได้แสดงผลคือ ข้อความสุดท้ายที่ Client C ส่งไว้



ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32

ทำการทดสอบเชื่อมต่อ Device2 บน NETPIE (ต้องสร้าง Device2 ขึ้นมาใหม่ก่อน)



ESP32 บน I-Kit V.2

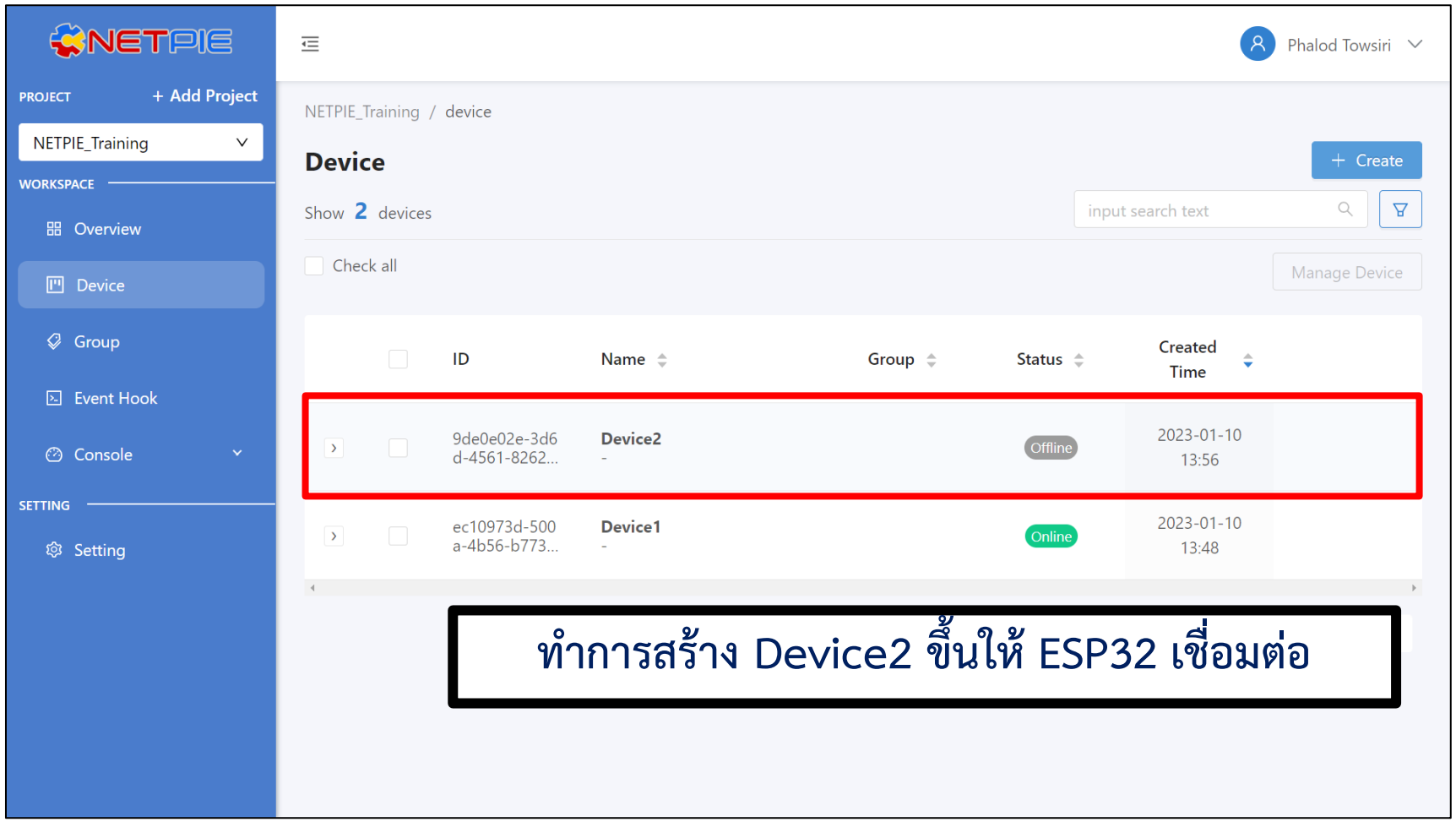
Connect

NETPIE

เชื่อมต่อ MQTT Server บน NETPIE2020 ด้วย ESP32

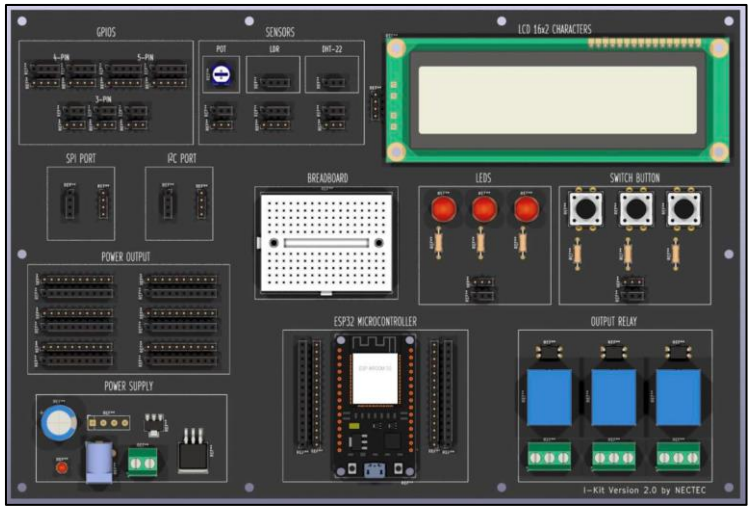
ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32



ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32



ESP32 บน I-Kit V.2

ในการจะให้ ESP32 สามารถเชื่อมต่อกับ NETPIE ได้นั้นจำเป็นต้องใช้ Library คือ

1. WiFi

ใช้สำหรับให้ KidBrightเชื่อมต่อกับอินเทอร์เน็ตผ่านเครือข่าย WiFi

2. PubSubClient

ใช้สำหรับให้ KidBright เชื่อมต่อและสื่อสารบน NETPIE 2020 (MQTT Protocol)

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32

คำสั่งและฟังก์ชันสำคัญใน PubSubClient

void callback

callback เป็นฟังก์ชันสำหรับการรับ payload หรือข้อความต่างๆที่ถูกส่งมาตาม Topic ที่ได้ Subscribe

void reconnect

reconnect เป็นฟังก์ชันสำหรับใช้ทำการเชื่อมต่อกับ MQTT Server ที่ได้ทำการตั้งค่าไว้ หรือเมื่อมีเหตุการณ์ขาดการเชื่อมต่อกับ MQTT Server ฟังก์ชัน reconnect จะถูกเรียกใช้เพื่อทำการเชื่อมต่ออีกครั้ง

client.setServer()

client.setServer() เป็นคำสั่งตั้งค่า MQTT Server รูปแบบคือ

client.setServer(mqtt_server, mqtt_port)

client.connect()

client.connect() เป็นคำสั่งที่ใช้เชื่อมต่อกับ MQTT Broker มีรูปแบบคือ

client.connect(client, username, password)

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Your_SSID";
const char* password = "Your_Password";
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "Client_ID";
const char* mqtt_username = "Token";
const char* mqtt_password = "Secret";

WiFiClient espClient;
PubSubClient client(espClient);

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect(mqtt_Client, mqtt_username,
mqtt_password)) {
      Serial.println("connected");
    }
    else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(9600);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.setServer(mqtt_server, mqtt_port);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32

1 ส่วนที่ 1 การเรียกใช้ Library และ ประกาศตัวแปร

เป็นการเรียกใช้ Library ต่างๆ

เป็นการประกาศตัวแปรสำหรับเชื่อมต่อ WiFi

เป็นการประกาศตัวแปรสำหรับเชื่อมต่อ MQTT Server

การกำหนดและเรียกใช้ชุดคำสั่งสำหรับเชื่อมต่อ MQTT Server

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Your WiFi SSID";
const char* password = "Your Password";

const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "Your Client-ID";
const char* mqtt_username = "Your Token";
const char* mqtt_password = "Your Secret";

WiFiClient espClient;
PubSubClient client(espClient);
```

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการเชื่อมต่อ MQTT Server

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting NETPIE2020 connection...");
    if (client.connect(mqtt_Client, mqtt_username, mqtt_password)) {
      Serial.println("NETPIE2020 connected");
    }
    else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
  }
}
```

เข้าสู่การเชื่อมต่อ MQTT Server

- หากเชื่อมต่อสำเร็จจะแสดงผลว่า “connected”
- หากเชื่อมต่อไม่สำเร็จจะแสดงผลว่า “failed ...” และจะทำการเชื่อมต่อใหม่อัตโนมัติ

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

```
void setup() {
  Serial.begin(115200);
  Serial.println("Starting...");
  if (WiFi.begin(ssid, password)) {
    while (WiFi.status() != WL_CONNECTED) {
      delay(1000);
      Serial.print(".");
    }
  }
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.setServer(mqtt_server, mqtt_port);
}
```

ฟังก์ชันการตั้งค่าต่างๆ

ในฟังก์ชันนี้จะทำการเชื่อมต่อกับ WiFi และ MQTT Server ตามค่าต่างๆที่ได้ตั้งค่าไว้

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32

3 ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

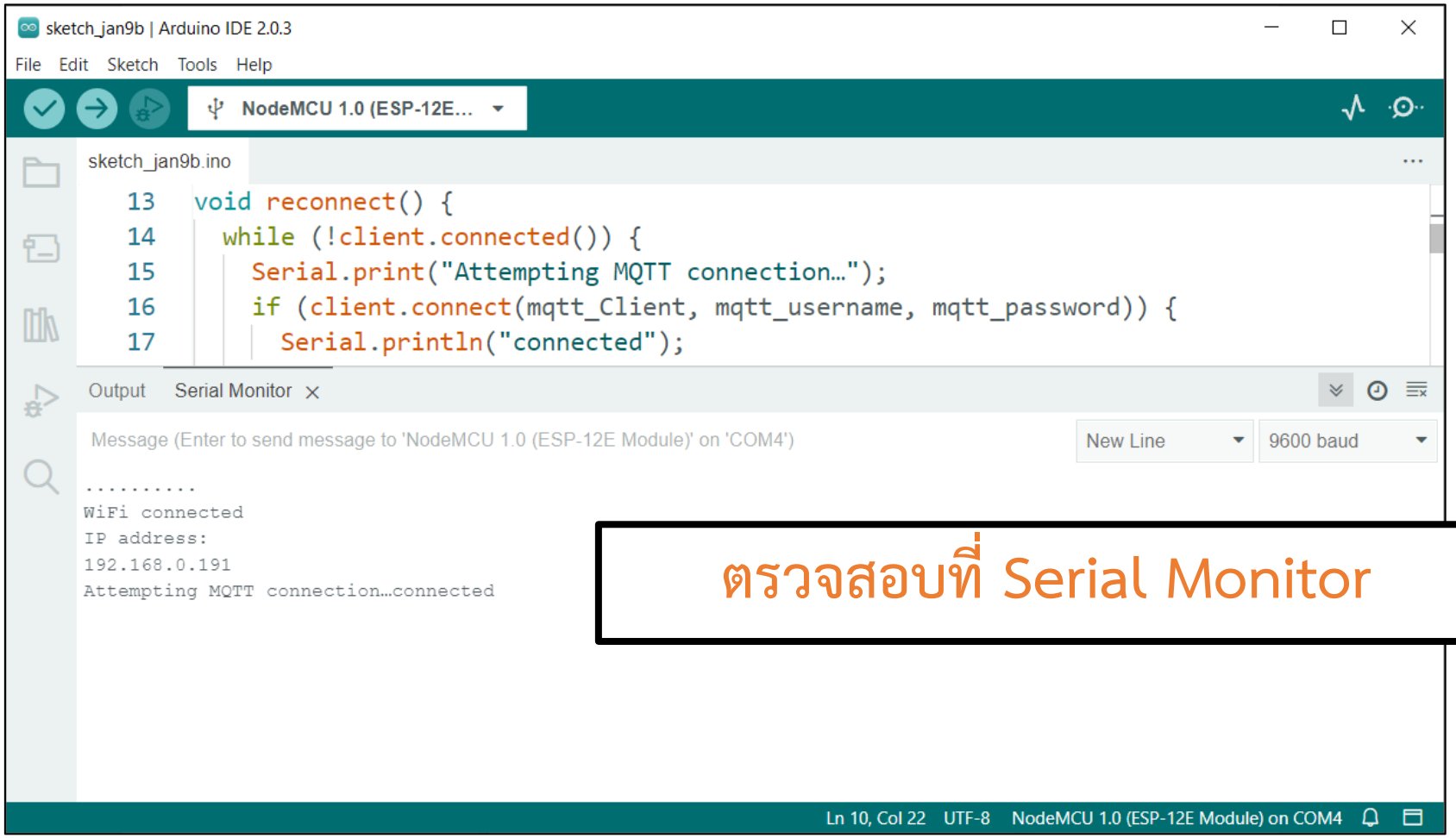
ฟังก์ชันการทำงานหลัก

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

}
```

เป็นชุดคำสั่งคงสถานะของการเชื่อมต่อและการทำงานต่างๆของ MQTT Server

ใบงานที่ 4.2 ขั้นตอนการทดลอง

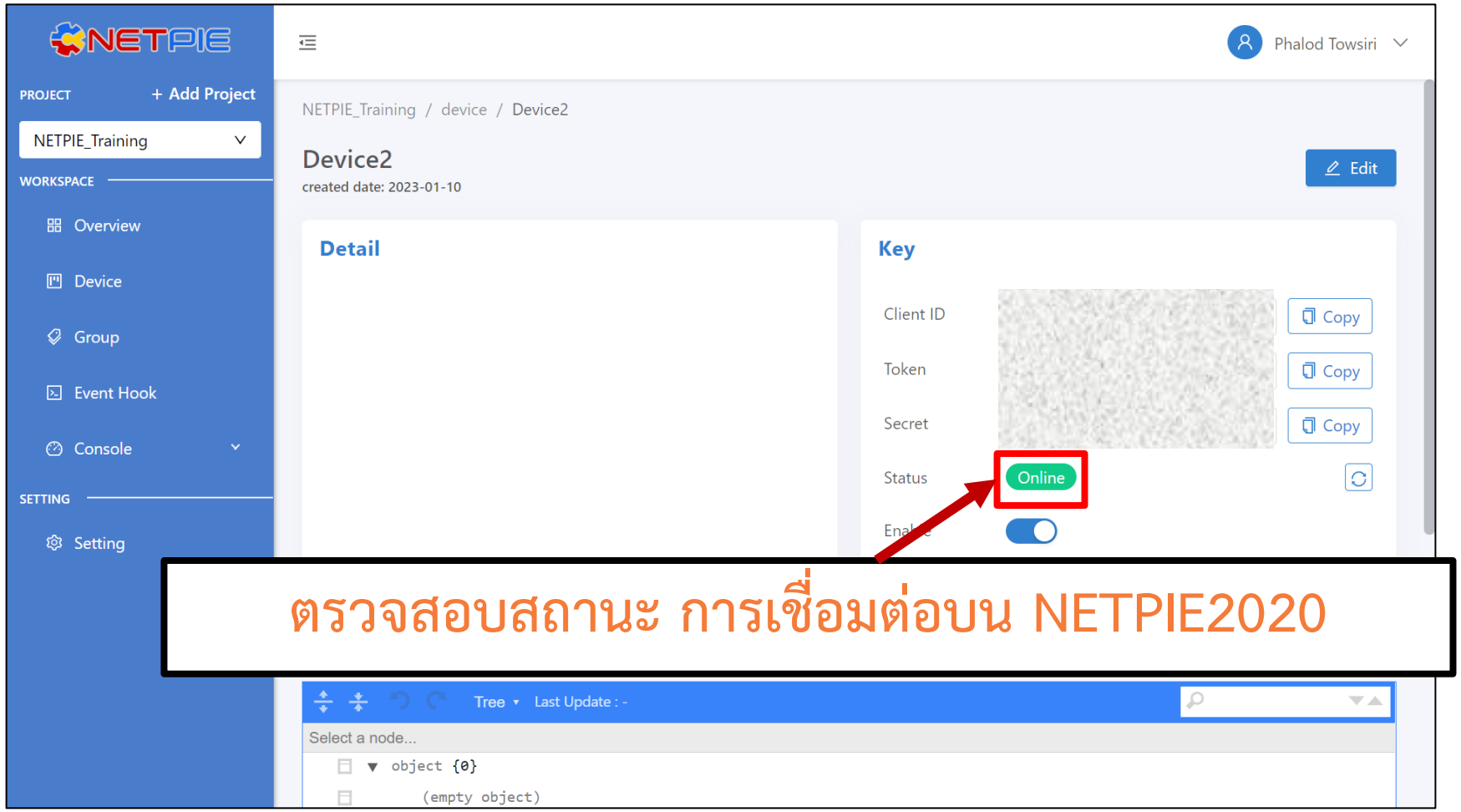
การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32



ตรวจสอบที่ Serial Monitor

ใบงานที่ 4.2 ขั้นตอนการทดลอง

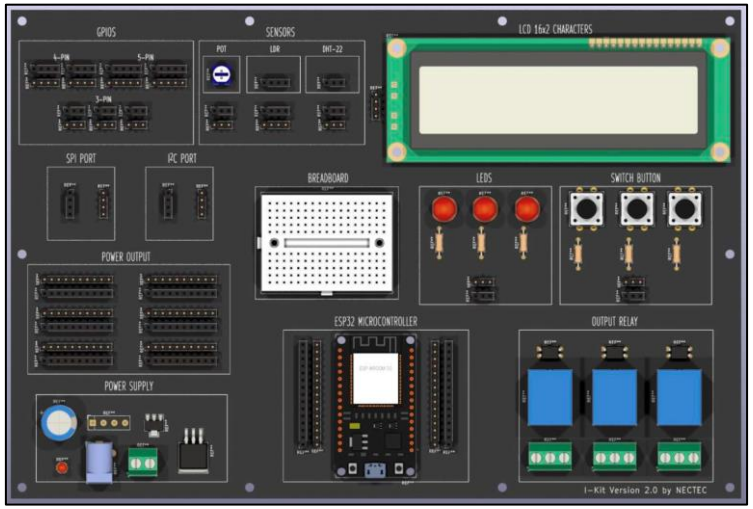
การทดลองที่ 1 ทดสอบการเชื่อมต่อด้วย ESP32



ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อความผ่าน NETPIE2020 ด้วย ESP32 โดยใช้ MQTT Protocol

ทดสอบการสื่อสาร Device2 บน NETPIE



ESP32 บน I-Kit V.2

”Hello NETPIE”

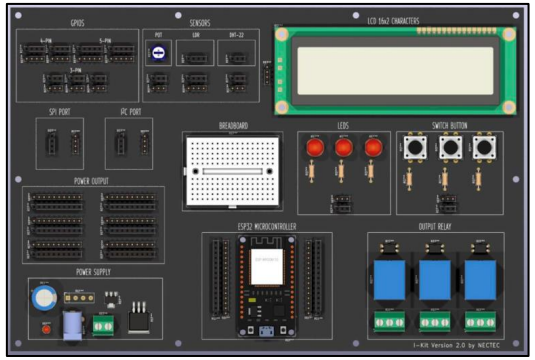
NETPIE

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อความผ่าน NETPIE2020 ด้วย ESP32 โดยใช้ MQTT Protocol

ทำความเข้าใจรูปแบบการสื่อสารบน NETPIE

การจะส่งข้อความไปที่ NETPIE จะต้อง
อ้างอิงหัวข้อเรื่อง ซึ่งเรียกว่า Topic โดยจะต้องขึ้นต้นด้วย @msg/... เสมอ



Topic : @msg/test

"Hello NETPIE"

I-Kit ทำการส่งข้อความไปที่ MQTT Server ด้วยประโยคว่า
"Hello NETPIE"
ซึ่งเรียกว่า การ Publish

NETPIE2020 ทำหน้าที่เปรียบเสมือน
ห้องแชทห้องหนึ่ง
ซึ่งเรียกว่า Broker

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อความผ่าน NETPIE2020 ด้วย ESP32 โดยใช้ MQTT Protocol

คำสั่งและฟังก์ชันสำคัญใน PubSubClient

client.publish()

client.publish() เป็นคำสั่ง publish ไปยัง Topic ที่ต้องการ

client.publish(“topic”, “Message”)

<u>การใช้งานคำสั่ง Publish</u>

1. การจะ Publish ข้อมูลได้นั้น อุปกรณ์จำเป็นจะต้องเชื่อมต่อกับ MQTT Server ก่อนเสมอ
2. คำสั่ง client.publish("topic", "Message") จะต้องอ้างอิง Topic ในการส่งข้อมูลเสมอ
3. คำสั่ง client.publish("topic", "Message") ข้อความที่ถูกส่งไปจะต้องถูกจัดอยู่ในรูปของ String
   <u>หรือหากข้อความมีความซับซ้อนสามารถแปลงข้อมูลจาก String เป็น CharArray เพื่อการจัดการที่ง่ายขึ้น</u>

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อความผ่าน NETPIE2020 ด้วย ESP32 โดยใช้ MQTT Protocol

1 ส่วนที่ 1 การเรียกใช้ Library และ ประกาศตัวแปร

เป็นการเรียกใช้ Library ต่างๆ

เป็นการประกาศตัวแปรสำหรับเชื่อมต่อ WiFi

เป็นการประกาศตัวแปรสำหรับเชื่อมต่อ MQTT Server

การกำหนดและเรียกใช้ชุดคำสั่งสำหรับเชื่อมต่อ MQTT Server

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Your WiFi SSID";
const char* password = "Your Password";

const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "Your Client-ID";
const char* mqtt_username = "Your Token";
const char* mqtt_password = "Your Secret";

WiFiClient espClient;
PubSubClient client(espClient);
```

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อความผ่าน NETPIE2020 ด้วย ESP32 โดยใช้ MQTT Protocol

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการเชื่อมต่อ MQTT Server

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting NETPIE2020 connection...");
    if (client.connect(mqtt_Client, mqtt_username, mqtt_password)) {
      Serial.println("NETPIE2020 connected");
    }
    else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
  }
}
```

เข้าสู่การเชื่อมต่อ MQTT Server

- หากเชื่อมต่อสำเร็จจะแสดงผลว่า “connected”
- หากเชื่อมต่อไม่สำเร็จจะแสดงผลว่า “failed ...” และจะทำการเชื่อมต่อใหม่อัตโนมัติ

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อความผ่าน NETPIE2020 ด้วย ESP32 โดยใช้ MQTT Protocol

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

```
void setup() {
  Serial.begin(115200);
  Serial.println("Starting...");
  if (WiFi.begin(ssid, password)) {
    while (WiFi.status() != WL_CONNECTED) {
      delay(1000);
      Serial.print(".");
    }
  }
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.setServer(mqtt_server, mqtt_port);
}
```

ฟังก์ชันการตั้งค่าต่างๆ

ในฟังก์ชันนี้จะทำการเชื่อมต่อกับ WiFi และ MQTT Server ตามค่าต่างๆที่ได้ตั้งค่าไว้

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อความผ่าน NETPIE2020 ด้วย ESP32 โดยใช้ MQTT Protocol

3 ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

```
void loop() {

 if (!client.connected()) {
    reconnect();
  }
  client.loop();
  client.publish("@msg/test", "Hello NETPIE");
  Serial.println("Send Message Success");
  delay(2000);
}
```

ฟังก์ชันการทำงานหลัก

เป็นชุดคำสั่งคงสถานะของการเชื่อมต่อและการทำงานต่างๆของ MQTT Server

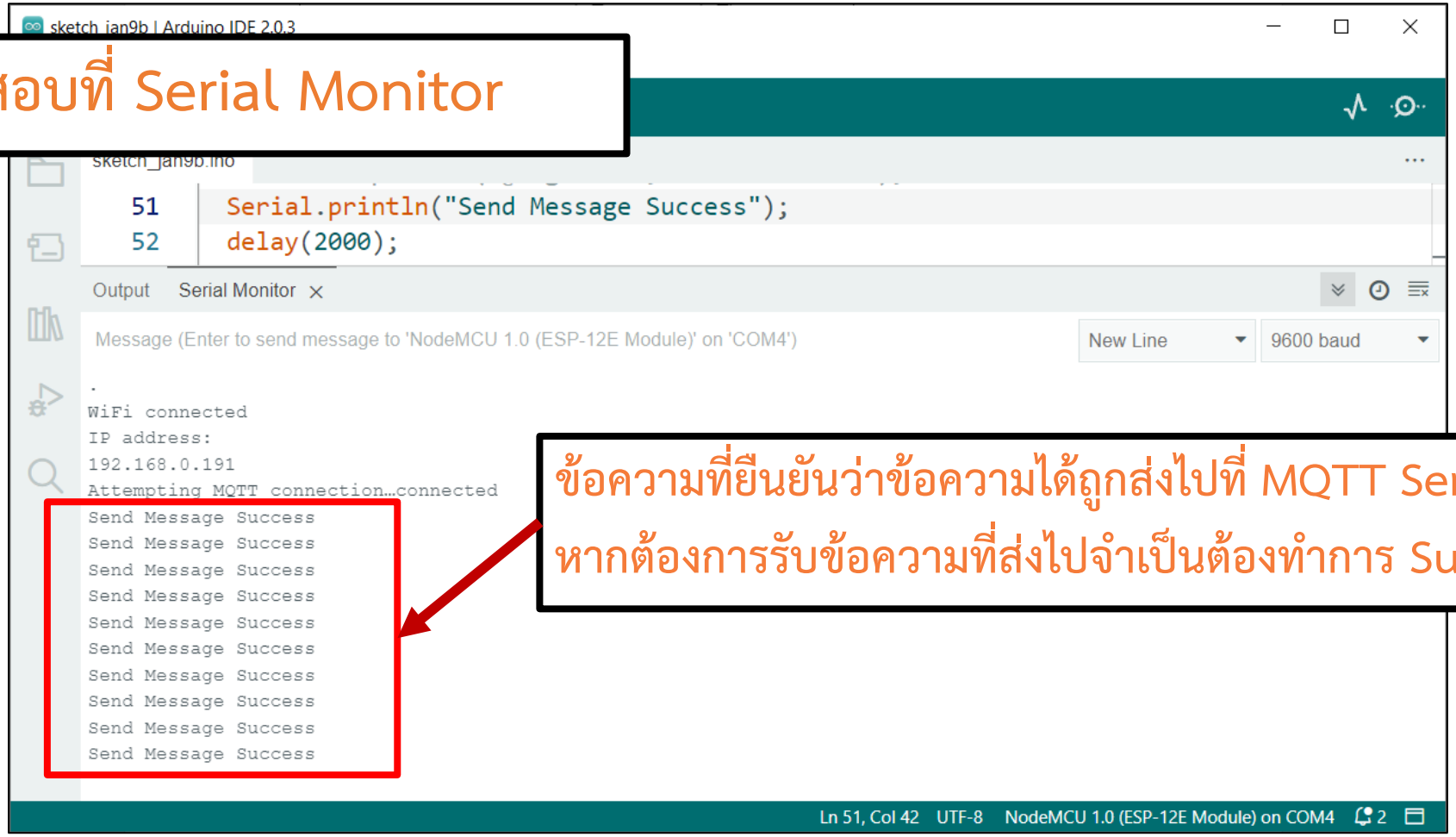
คำสั่งการส่งข้อมูลไปยัง MQTT Server โดยส่งไปที่ Topic คือ @msg/test และข้อความคือ Hello NETPIE

คำสั่งแสดงผลข้อความหลังจาก Publish ข้อมูลสำเร็จ

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อความผ่าน NETPIE2020 ด้วย ESP32 โดยใช้ MQTT Protocol

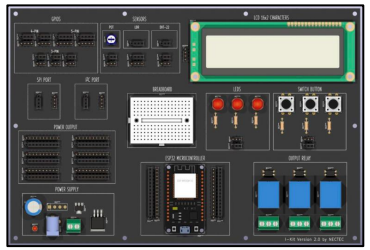
ตรวจสอบที่ Serial Monitor



ข้อความที่ยืนยันว่าข้อความได้ถูกส่งไปที่ MQTT Server แล้ว
หากต้องการรับข้อความที่ส่งไปจำเป็นต้องทำการ Subscribe ข้อมูล

ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

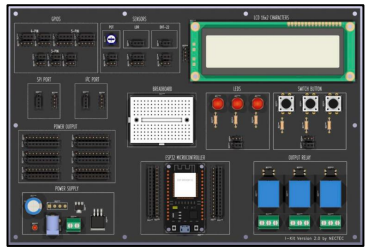
ทำความเข้าใจรูปแบบการสื่อสารบน NETPIE



Publish Topic : @msg/test

"Hello NETPIE"

I-Kit 1 ทำการ Publish ข้อความไปยัง Topic @msg/test



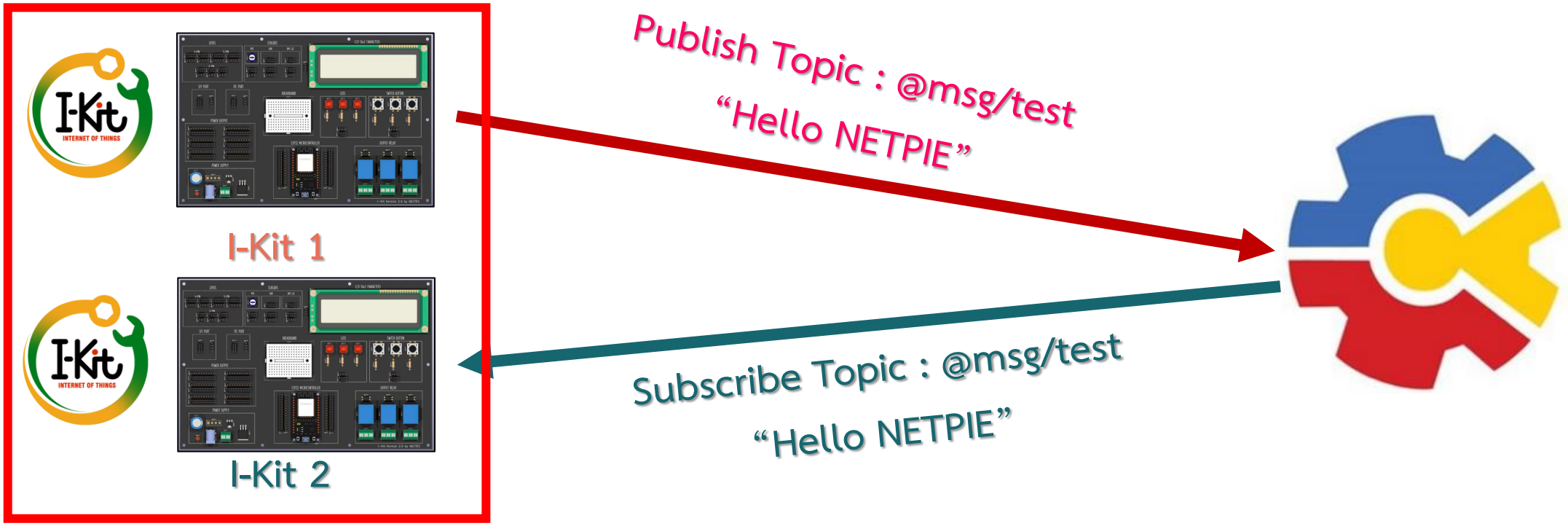
Subscribe Topic : @msg/test

"Hello NETPIE"

I-Kit 2 ต้องการข้อความที่ I-Kit 1 ส่งมาต้องทำการเลือกรับข้อความตาม Topic ซึ่งเรียกว่าการ Subscribe

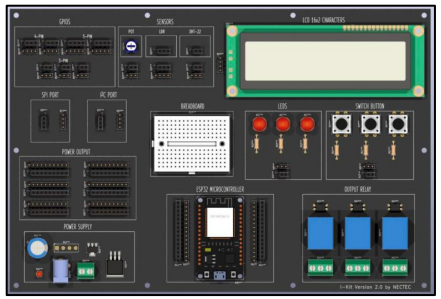
ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

ทำความเข้าใจรูปแบบการสื่อสารบน NETPIE



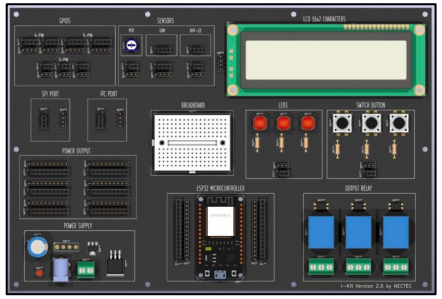
Device ทั้ง 2 จะต้องอยู่ใน Group เดียวกัน

ใบงานที่ 4.2 ทฤษฎีเบื้องต้น



I-Kit 1

“Hello NETPIE”



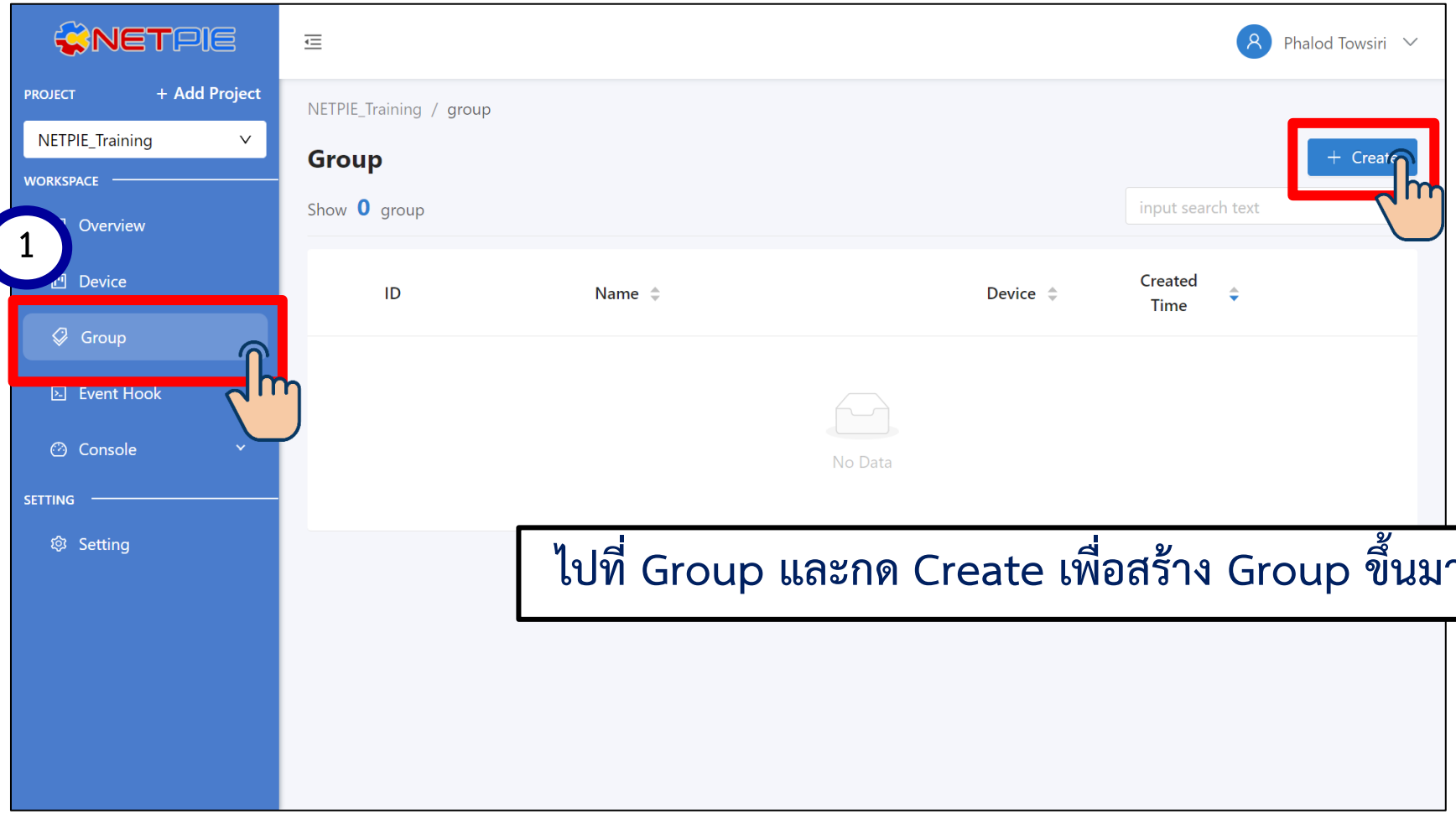
I-Kit 2

ใช้ MQTTBox แทน

“Hello NETPIE”

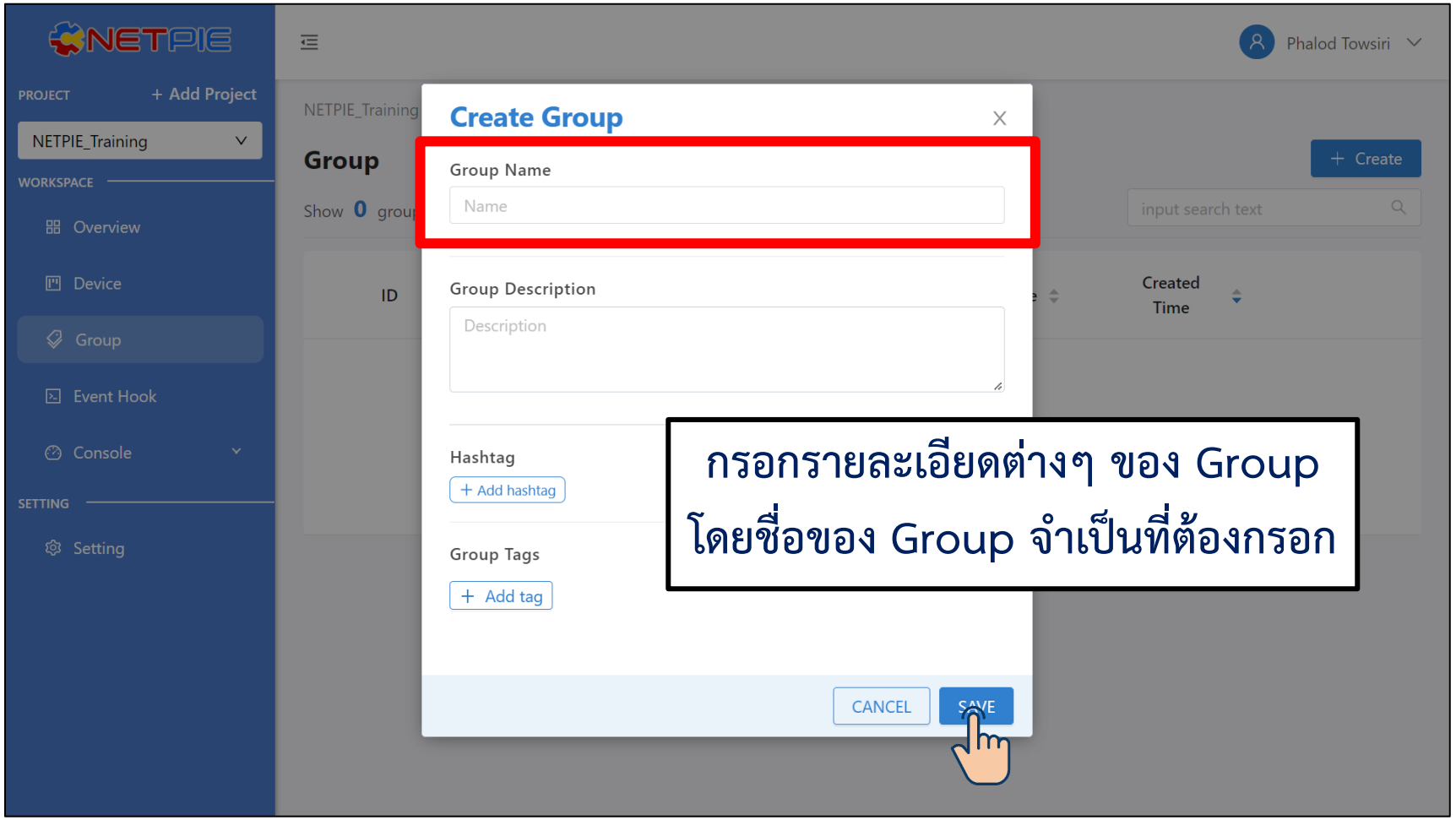
ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



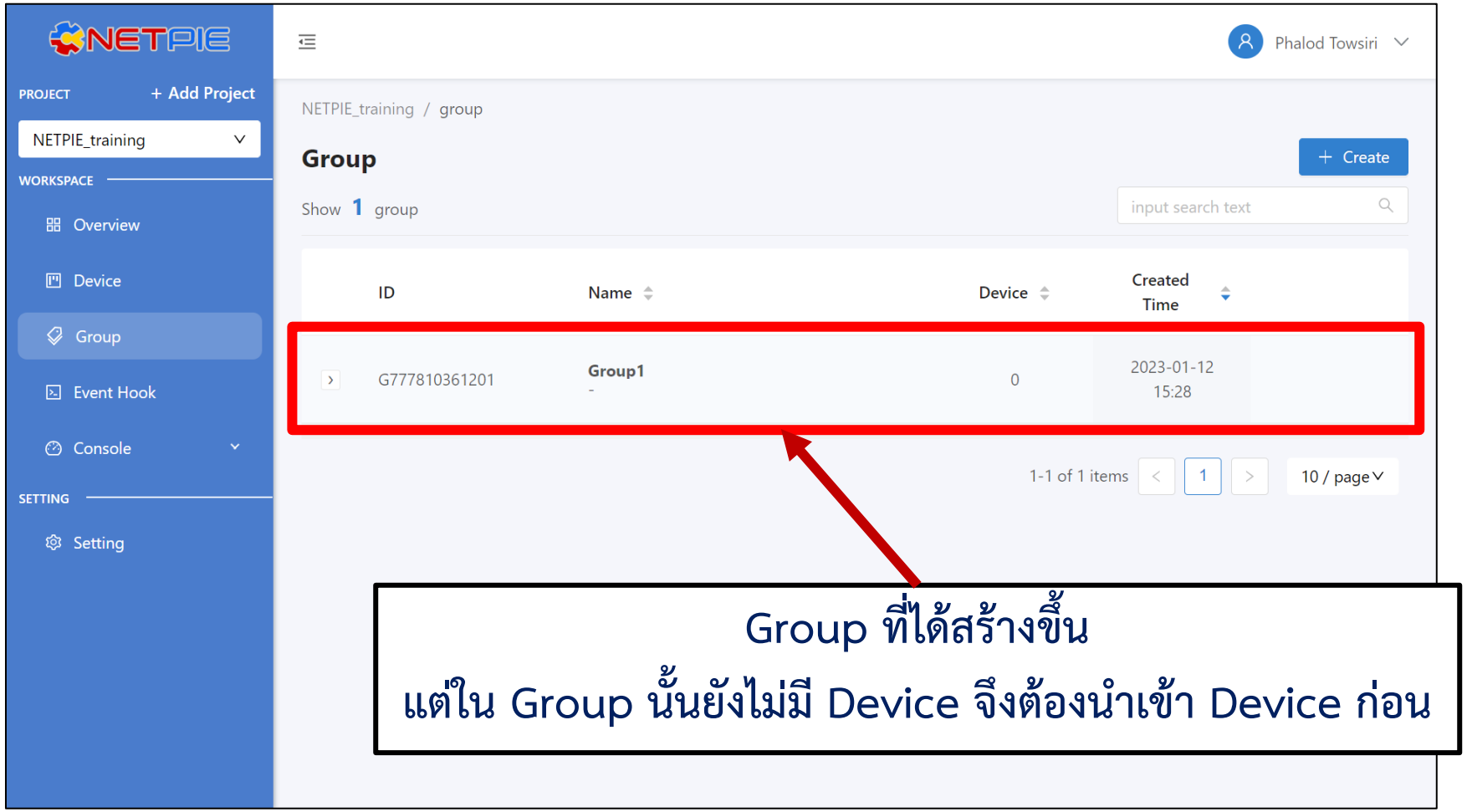
ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



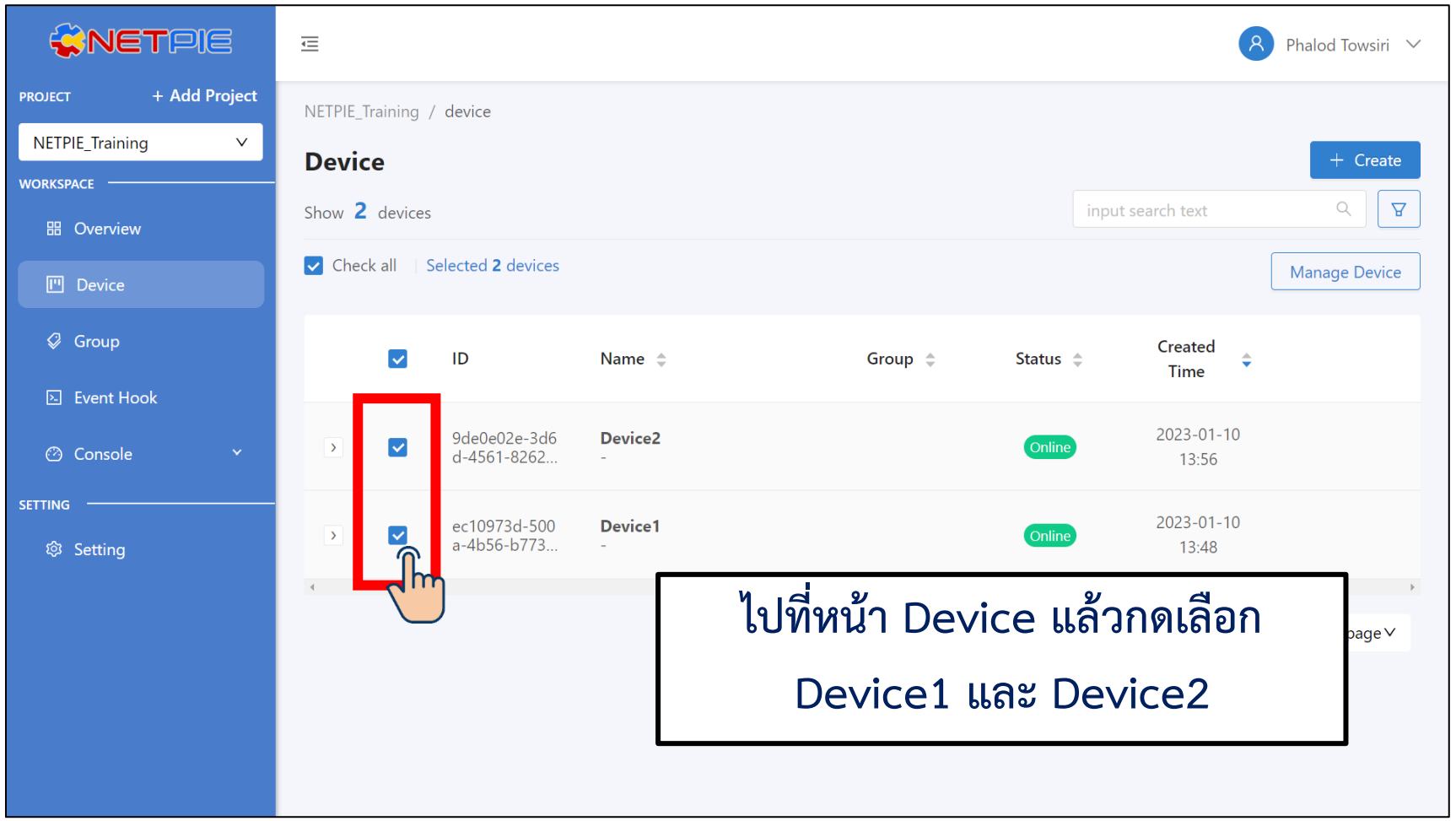
ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



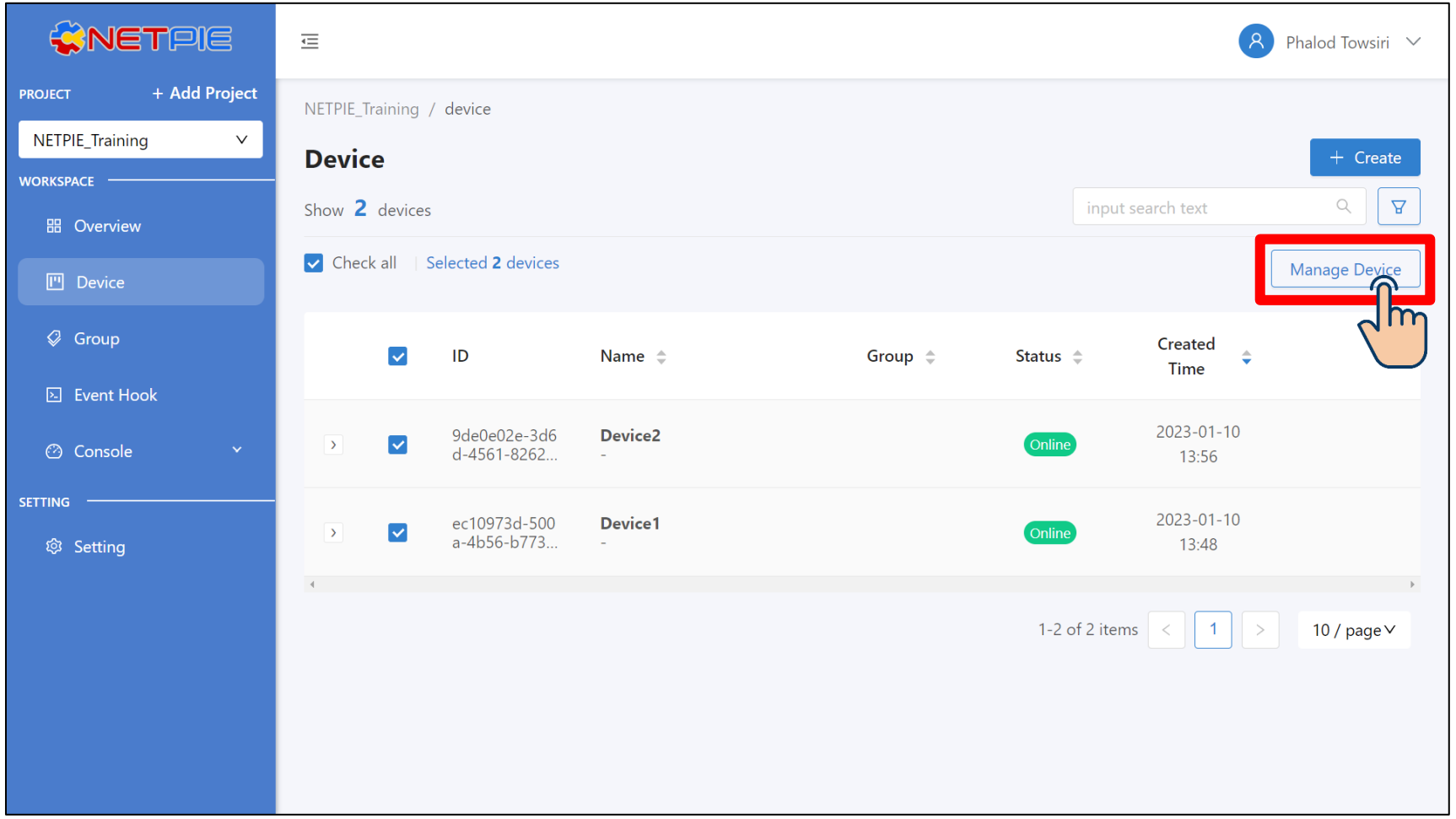
ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



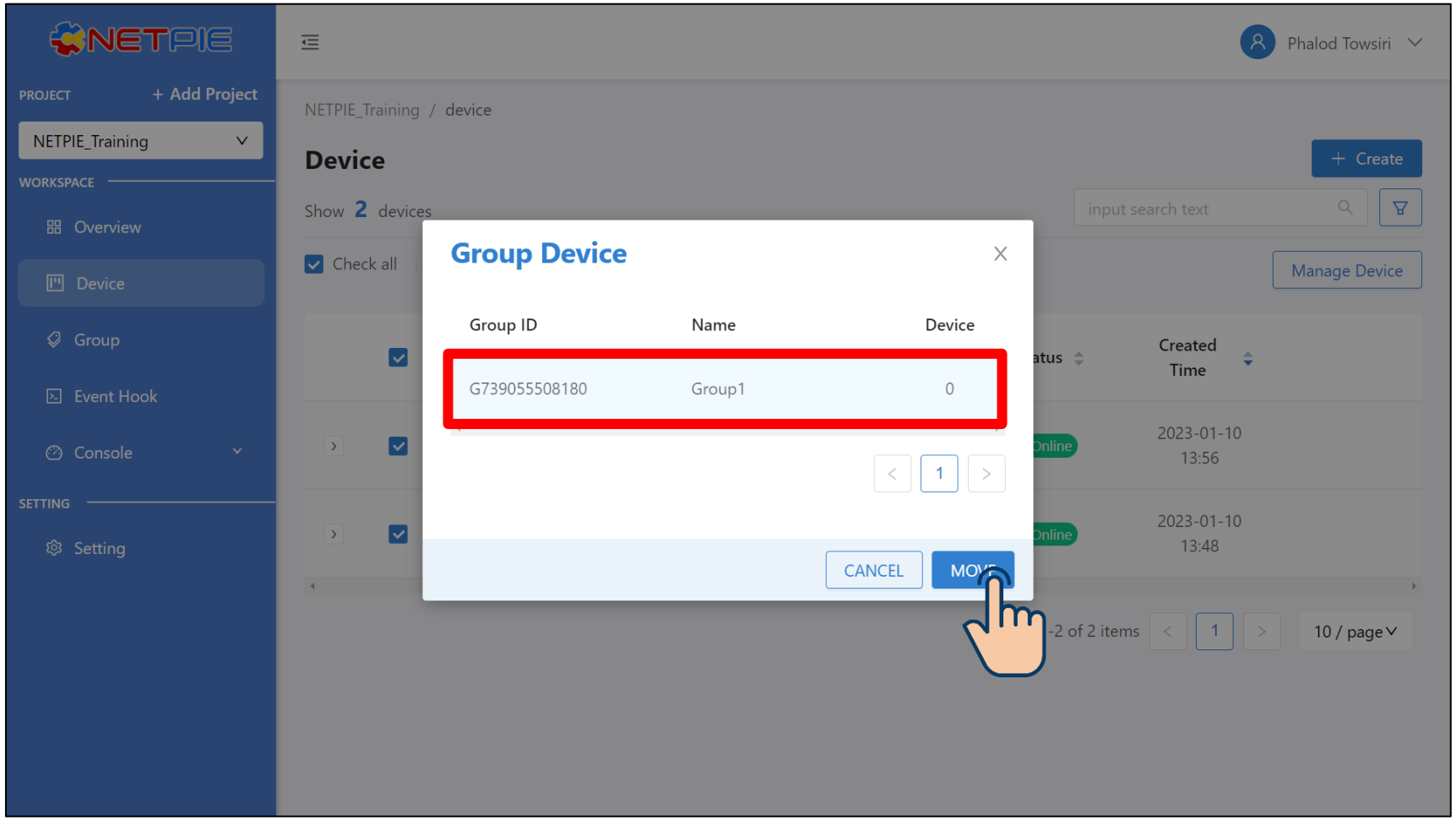
ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



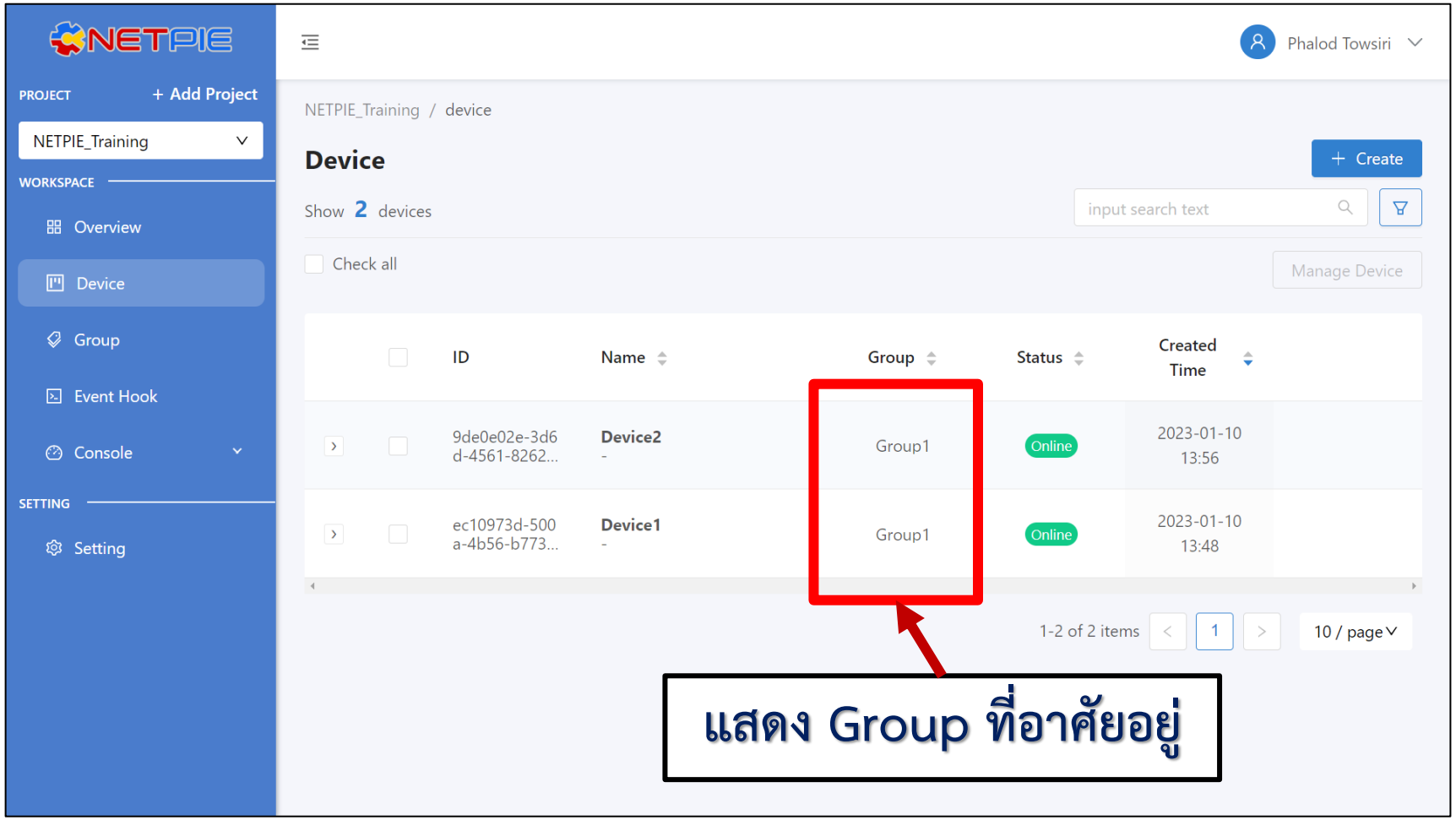
ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



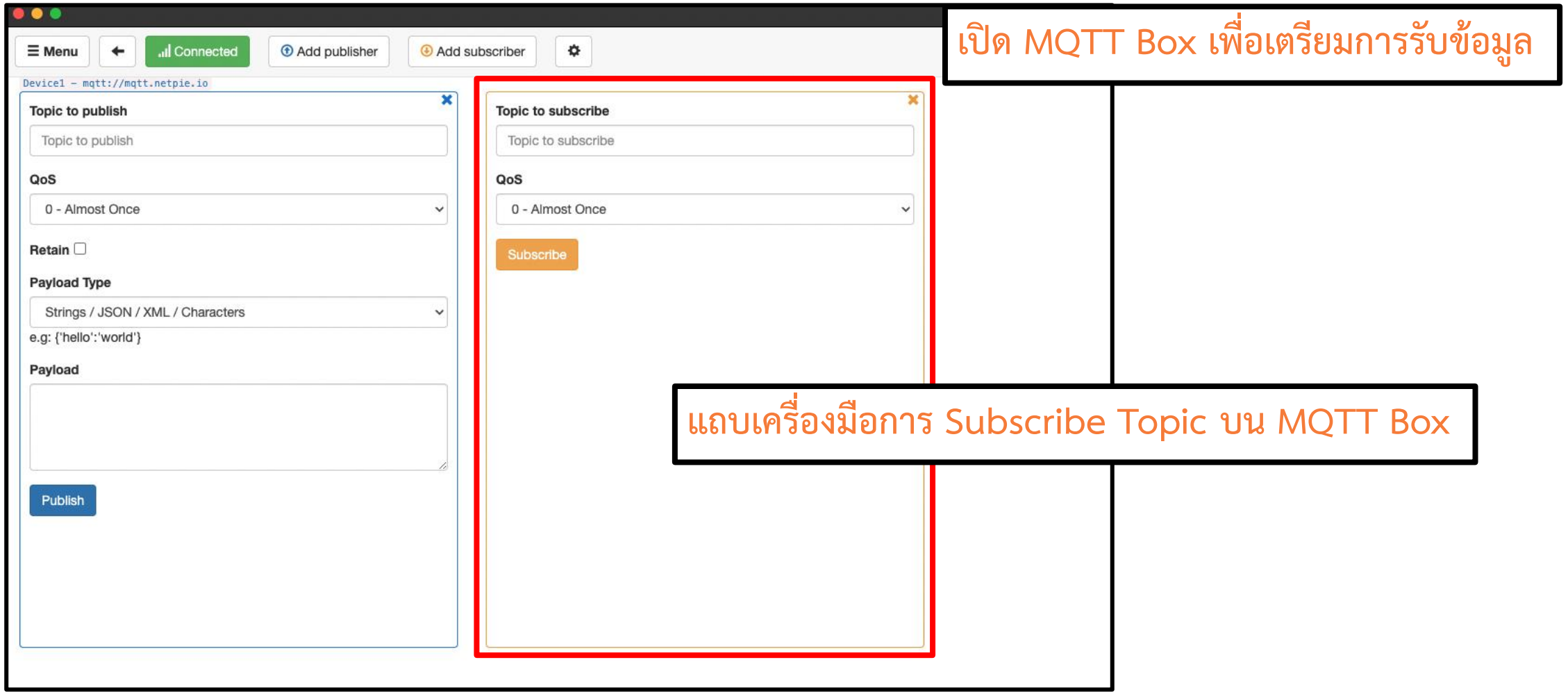
ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



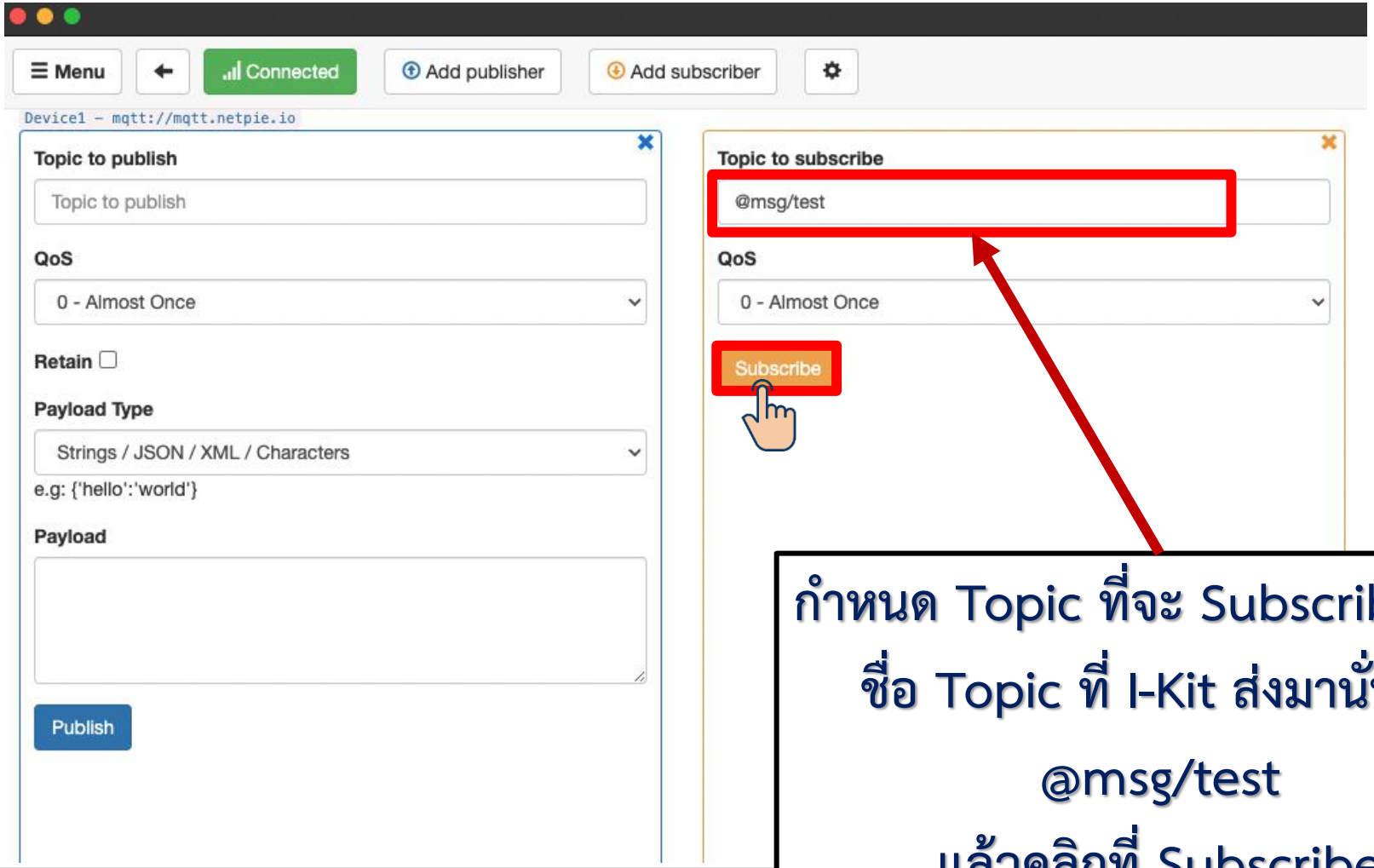
ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



ใบงานที่ 4.2 ขั้นตอนการทดลอง

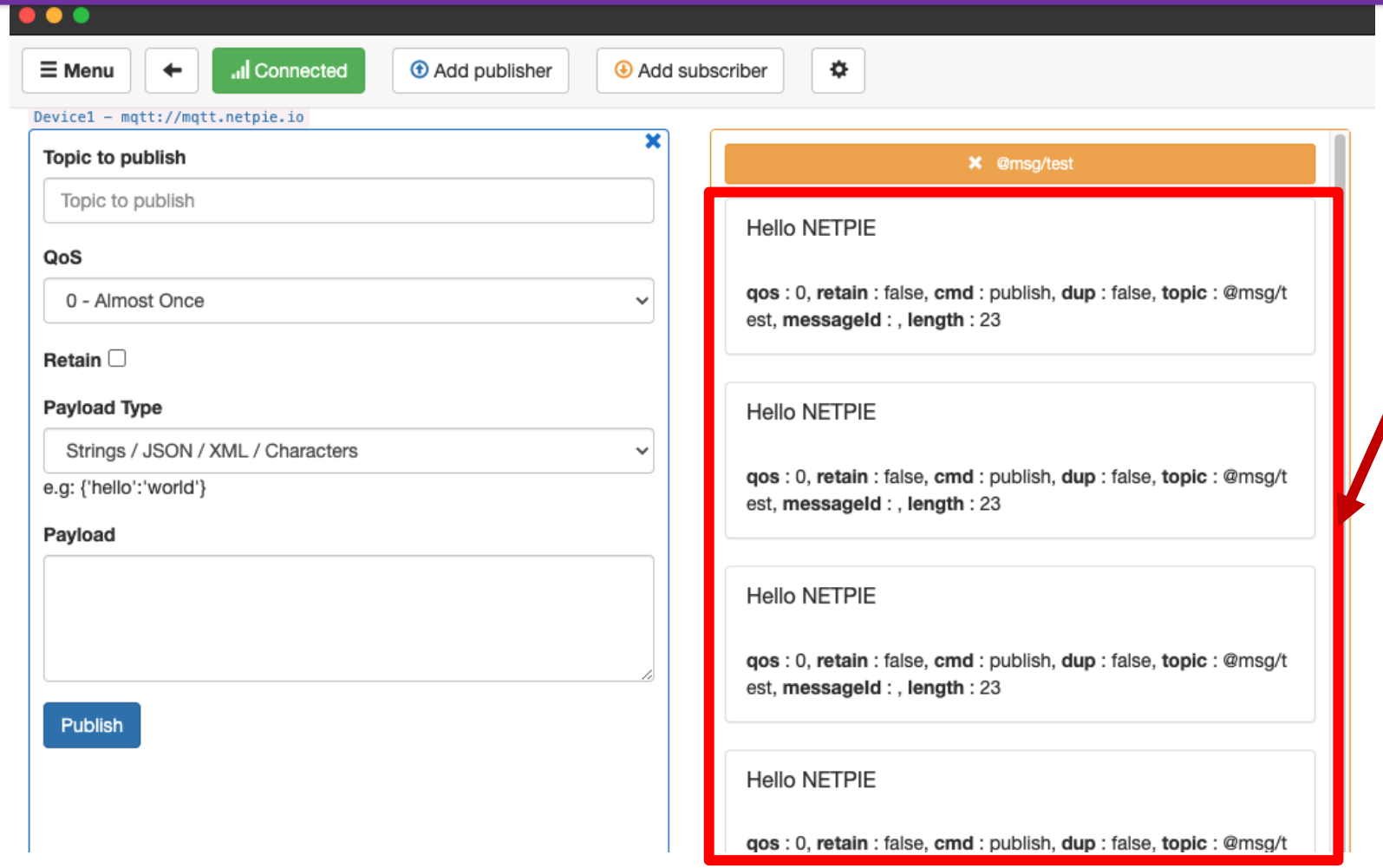
การทดลองที่ 3 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



กำหนด Topic ที่จะ Subscribe ตาม
ชื่อ Topic ที่ I-Kit ส่งมานั่นคือ
@msg/test
แล้วคลิกที่ Subscribe

ใบงานที่ 4.2 ขั้นตอนการทดลอง

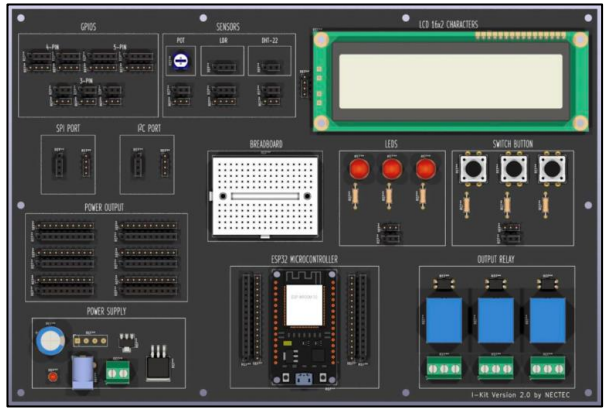
การทดลองที่ 3 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



ข้อความที่ถูกส่งมาจาก ESP8266

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 4 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



I-Kit 1

MQTTBox

ทำการทดสอบส่งข้อมูลจาก MQTT Box
และให้ ESP8266 รับข้อความ

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 4 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box

คำสั่งและฟังก์ชันสำคัญใน PubSubClient

client.subscribe()

client.subscribe เป็นคำสั่ง subscribe Topic ที่ต้องการ ส่วนมากนิยมนำไปไว้ในฟังก์ชัน reconnect รูปแบบคือ

client.subscribe(“topic”)

void callback

callback เป็นฟังก์ชันสำหรับการรับ payload หรือข้อความต่างๆที่ถูกส่งมาตาม Topic ที่ได้ Subscribe

การใช้งานคำสั่ง Subscribe

1. การจะ Subscribe Topic ได้นั้น อุปกรณ์จำเป็นจะต้องเชื่อมต่อกับ MQTT Server ก่อนเสมอ
2. การ Subscribe Topic นั้นนิยมเขียนไว้ในฟังก์ชัน void reconnect เสมอ
   เนื่องจากทุกครั้งที่อุปกรณ์ขาดการเชื่อมต่อจาก MQTT Server อุปกรณ์จะกลับมาทำงานที่ฟังก์ชันนี้เสมอ และทำให้ Topic ที่ Subscribe ขาดการเชื่อมต่อด้วย เมื่ออุปกรณ์เชื่อมต่อใหม่ได้สำเร็จจะได้ทำการ Subscribe Topic ให้ใหม่อัตโนมัติ
3. ข้อความที่ได้รับจากการ Subscribe Topic จะถูกแสดงผลในฟังก์ชัน void callback หากต้องการนำข้อความมาใช้งานมักเขียนที่ฟังก์ชันนี้

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 4 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box

1 ส่วนที่ 1 การเรียกใช้ Library และ ประกาศตัวแปร

การใช้งานส่วนมากจะเหมือนกับตัวอย่างก่อนหน้านี้

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Your WiFi SSID";
const char* password = "Your Password";

const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "Your Client-ID";
const char* mqtt_username = "Your Token";
const char* mqtt_password = "Your Secret";

WiFiClient espClient;
PubSubClient client(espClient);
```

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 4 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect(mqtt_Client, mqtt_username, mqtt_password)) {
      Serial.println("connected");
      client.subscribe("@msg/test2");
    }
    else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
  }
}
```

ฟังก์ชันการเชื่อมต่อ MQTT Server

คำสั่งในการ Subscribe Topic

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 4 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการรับข้อความ void callback

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  String message;
  for (int i = 0; i < length; i++) {
    message = message + (char)payload[i];
  }
  Serial.println(message);
}
```

แสดงผล Topic ที่รับมา

แสดงผลข้อความที่รับมา

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 4 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

```
void setup() {
  Serial.begin(9600);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}
```

ฟังก์ชันการตั้งค่าต่างๆ

คำสั่งการเรียกใช้ฟังก์ชัน callback

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 4 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box

3 ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

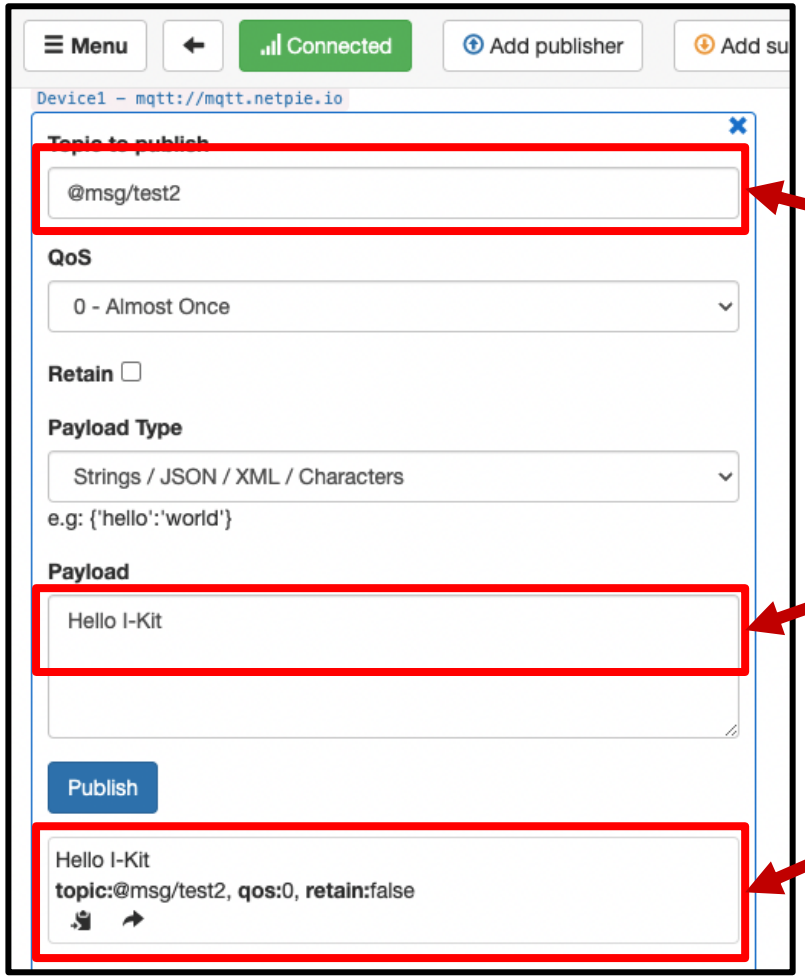
ฟังก์ชันการทำงานหลัก

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

}
```

เป็นชุดคำสั่งคงสถานะของการเชื่อมต่อและการทำงานต่างๆของ MQTT Server

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 4 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box



Topic ที่ต้องการส่งข้อความไป
ซึ่งต้องตรงกับ Topic ที่ ESP8266 Subscribe อยู่

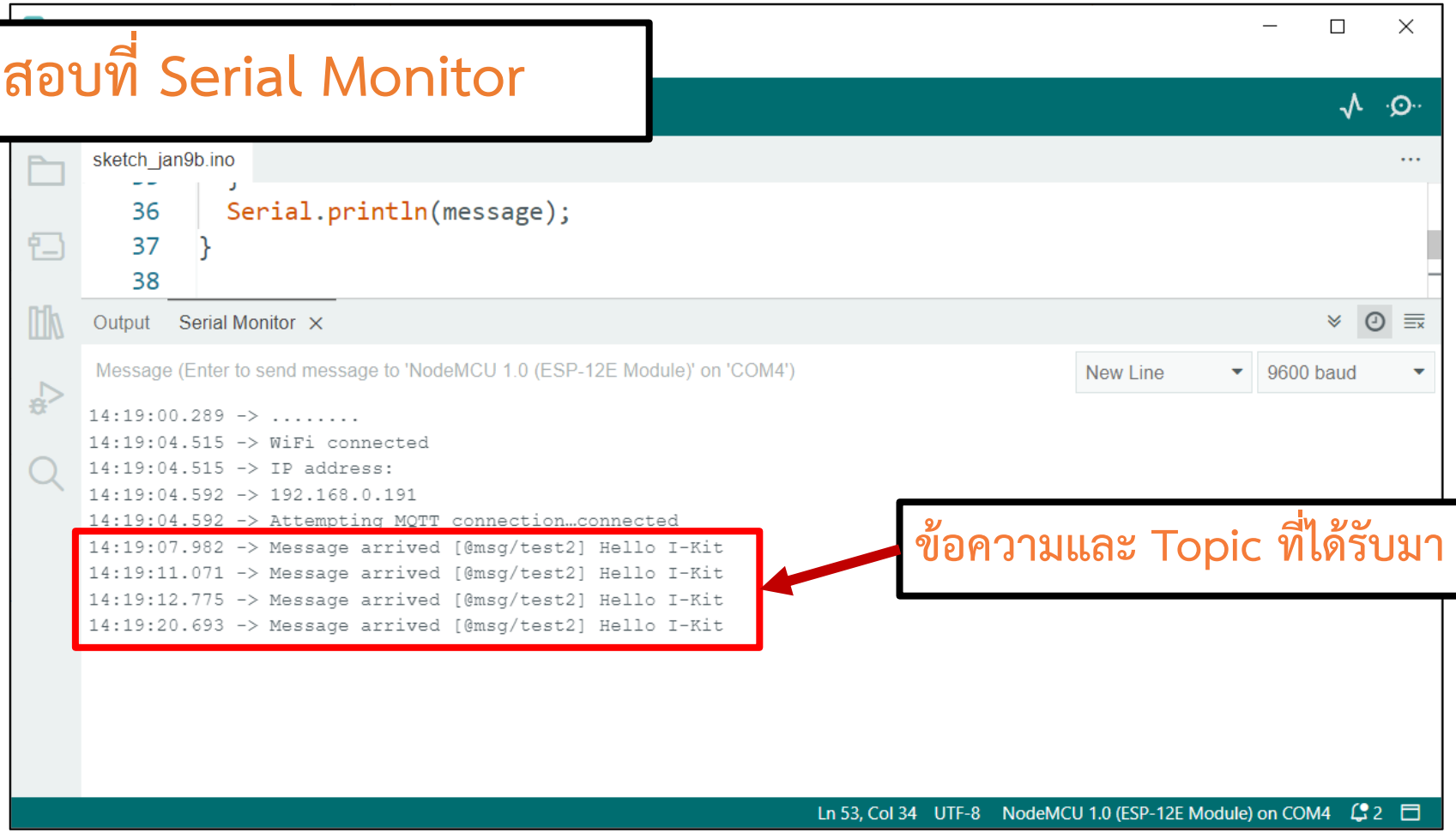
ข้อความที่ต้องการส่ง

สถานะการส่งข้อความ

ใบงานที่ 4.2 ขั้นตอนการทดลอง

การทดลองที่ 4 การสร้าง Device Group และทดสอบการรับข้อมูลจาก ESP32 ผ่าน MQTT Box

ตรวจสอบที่ Serial Monitor



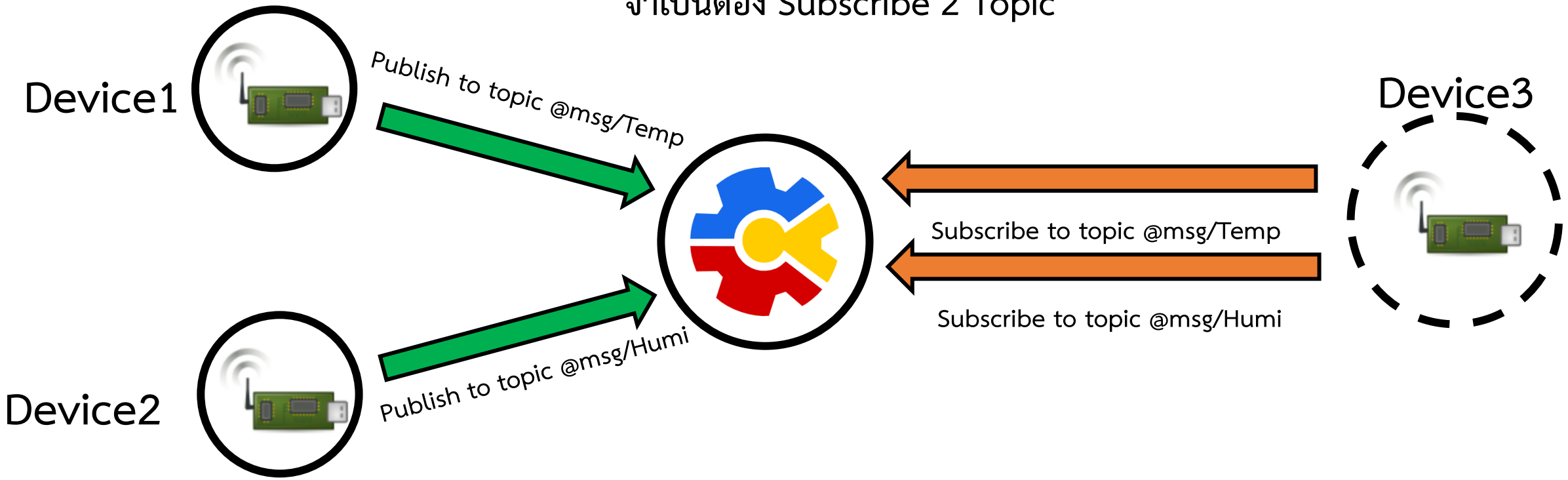
ข้อความและ Topic ที่ได้รับมา

ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

Topic แบบ Wildcard

หากต้องการให้ Device 3 ได้รับข้อความจาก Device1 และ Device2

จำเป็นต้อง Subscribe 2 Topic

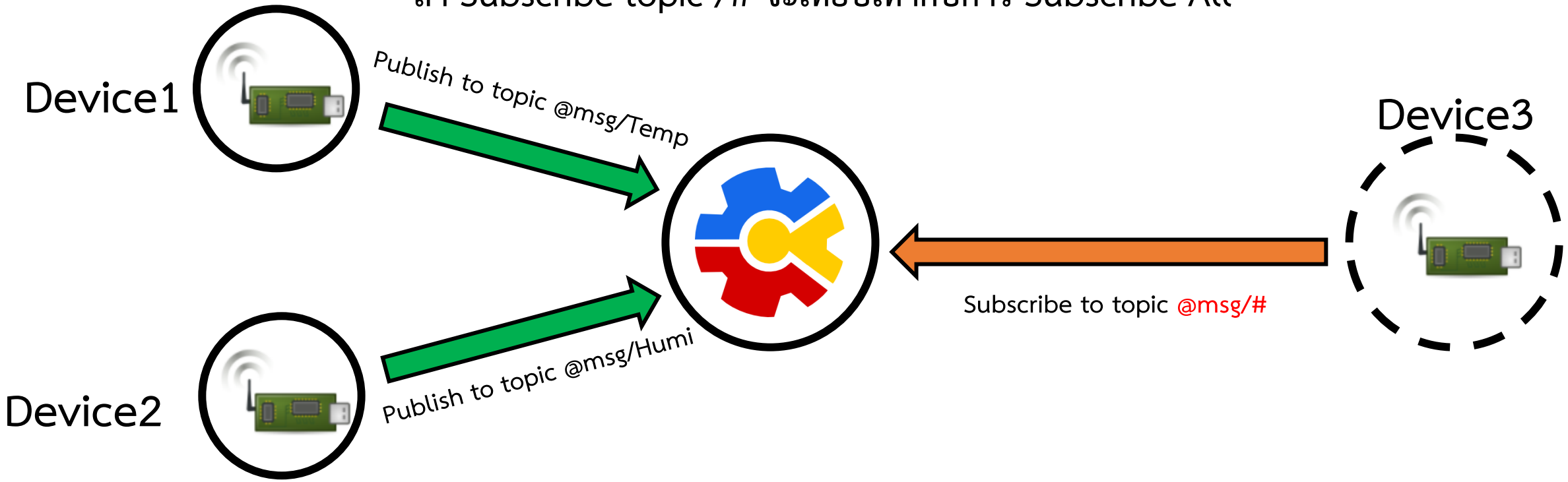


ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

Topic แบบ Wildcard

สามารถใช้สัญลักษณ์ # เป็นส่วนหนึ่งใน topic ซึ่งจะตรงกับคำทุกคำ และทุกจำนวน / ของคำ

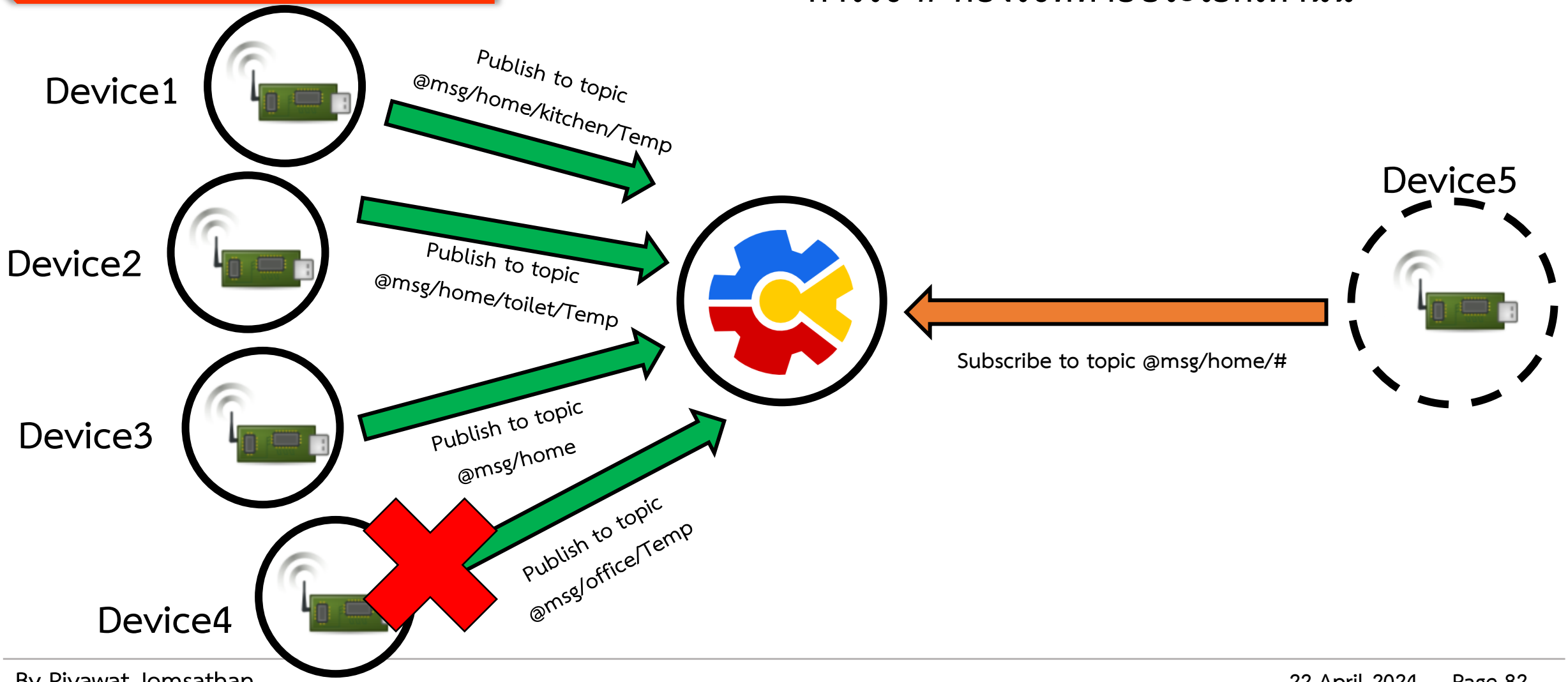
ถ้า Subscribe topic /# จะเทียบเท่ากับการ Subscribe All



ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

Topic แบบ Wildcard

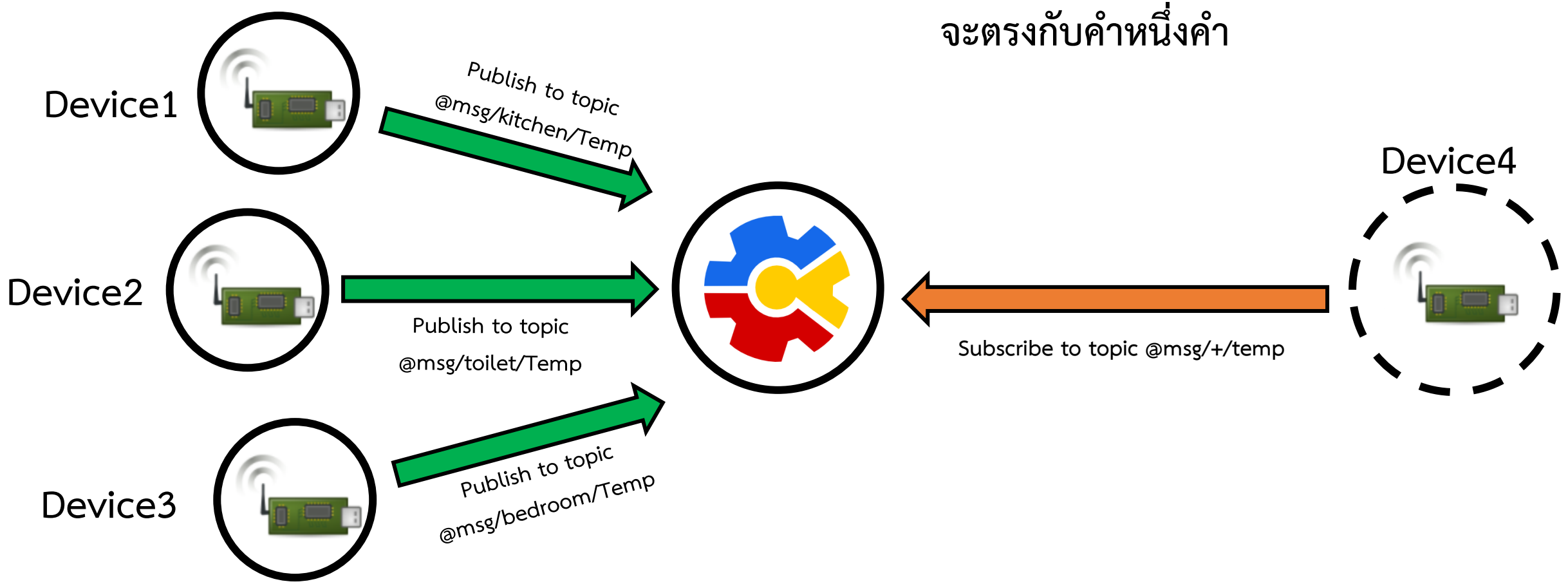
การใช้ # ต้องใช้ที่ท้ายประโยคเท่านั้น



ใบงานที่ 4.2 ทฤษฎีเบื้องต้น

Topic แบบ Wildcard

สามารถใช้สัญลักษณ์ + ใน Topic ซึ่ง
จะตรงกับคำหนึ่งคำ



ใบงานที่ 4.2 คำถามท้ายหน่วยการเรียนรู้

คำถามท้ายหน่วยการเรียนรู้ที่ 4

จงเขียนโปรแกรมควบคุม LED1 ที่เชื่อมต่อกับ ESP32 ผ่าน MQTT Protocol บน NETPIE โดยใช้ MQTT Box เป็นอุปกรณ์ในการส่งข้อความ โดยมีรูปแบบการ Publish – Subscribe ดังนี้

LED1 จะติดก็ต่อเมื่อได้รับข้อความว่า “on” จาก Topic @msg/esp32/led1

LED1 จะดับก็ต่อเมื่อได้รับข้อความว่า “off” จาก Topic @msg/esp32/led1

คำถามท้ายหน่วยการเรียนรู้ที่ 5

จงเขียนโปรแกรมควบคุม LED1, LED2 และ LED3 ที่เชื่อมต่อกับ ESP32 ผ่าน MQTT Protocol บน NETPIE โดยใช้ MQTT Box เป็นอุปกรณ์ในการส่งข้อความ โดยมีรูปแบบการ Publish – Subscribe ดังนี้

LED1 จะติดก็ต่อเมื่อได้รับข้อความว่า “on” จาก Topic @msg/esp32/led1

LED1 จะดับก็ต่อเมื่อได้รับข้อความว่า “off” จาก Topic @msg/esp32/led1

LED2 จะติดก็ต่อเมื่อได้รับข้อความว่า “on” จาก Topic @msg/esp32/led2

LED2 จะดับก็ต่อเมื่อได้รับข้อความว่า “off” จาก Topic @msg/esp32/led2

LED3 จะติดก็ต่อเมื่อได้รับข้อความว่า “on” จาก Topic @msg/esp32/led3

LED3 จะดับก็ต่อเมื่อได้รับข้อความว่า “off” จาก Topic @msg/esp32/led3

ใบงานที่ 4.3 การส่งข้อมูลจาก ESP32 เพื่อจัดเก็บบนฐานข้อมูลเสมือนบน NETPIE2020

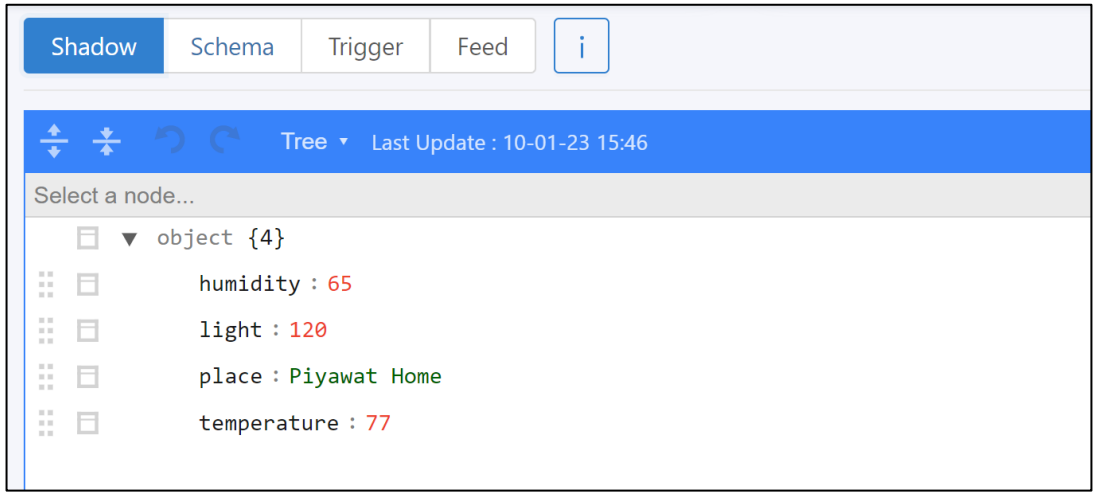
Device Shadow

ใบงานที่ 4.3 ทฤษฎีเบื้องต้น

Device Shadow

คือ ฐานข้อมูลเสมือนของอุปกรณ์ เป็นฐานข้อมูลเล็ก ๆ ที่มีคู่อยู่กับอุปกรณ์ (Device) ทุกตัว ใช้สำหรับเก็บข้อมูลต่าง ๆ เกี่ยวกับอุปกรณ์นั้น ๆ (Device Shadow Data) เช่น ข้อมูลที่เกิดจากเซนเซอร์ ข้อมูลการกำหนดองค์ประกอบต่าง ๆ (Device Configuration)

โดย Device Shadow จะอยู่ในรูปแบบ JSON



JSON (JavaScript Object Notation)

JSON เป็นข้อมูลรูปแบบ text ที่มีรูปแบบที่จะเก็บข้อมูลแบบ key, value โดยการเขียนข้อมูลชนิด JSON มีรูปแบบคือ ชื่อฟิลด์ครอบด้วยเครื่องหมาย “ (double quote), เครื่องหมาย : (colon), value แล้วครอบทั้งหมดด้วยเครื่องหมายปีกกา

ตัวอย่างที่มีข้อมูล 1 อย่างจะเป็นดังนี้

```
{"key": "value"}
```

การเก็บข้อมูลประเภท JSON object สามารถวางซ้อนเข้าไปอีกที ตัวอย่างเช่น

```
{"name": "Tae", "pet": {"dog": "Shiba", "cat": "Persian"}}
```

ประเภทข้อมูลที่ JSON เก็บได้มีดังนี้

string, number, object (JSON object)
array, boolean, null

หรือจะเก็บข้อมูล array ตัวอย่างเช่น

```
{"name": "Piyawat", "age": 28, "car": ["Toyota", "Honda"]}
```

ใบงานที่ 4.3 คำถามท้ายหน่วยการเรียนรู้

คำถามท้ายหน่วยการเรียนรู้ที่ 7

จงแปลงข้อมูลดังต่อไปนี้ให้อยู่ในรูปแบบของ JSON

data1 = { temperature = 25, humidity = 60, light = 120}, data2 = { voltage = 12, id = "ABCDB234", status = true }

จงแปลงข้อมูลดังต่อไปนี้ให้อยู่ในรูปแบบของ JSON

name = "Piyawat", id = "EN57364150",

data = { data1 = { temperature = 25, humidity = 60, light = 120, status = true }, data2 = {voltage = 12, status = false } }

จงแปลงข้อมูลดังต่อไปนี้ให้อยู่ในรูปแบบของ JSON

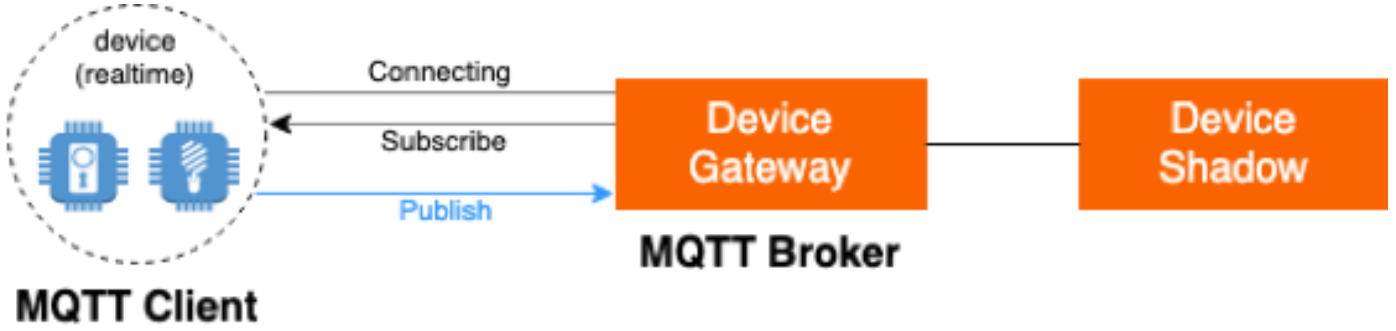
name = "Piyawat", id = "EN57364150", status = "true", date = "19/02/2022", device = [esp8266, esp32]

data = {esp8266 = { temperature = 25, humidity = 60, light = 120, status = true }, esp32 = {voltage = 12, status = false } }

ใบงานที่ 4.3 ทฤษฎีเบื้องต้น

การส่งข้อมูลไปยัง Device Shadow

การส่งข้อมูลไปยัง Device Shadow นั้นทำได้โดยการให้ Device ส่งมูลไปยัง Shadow Topic ซึ่ง MQTT Broker ของ NETPIE ได้ทำการ Subscribe ข้อมูลไว้แล้ว



Shadow Topic

Shadow Topic คือ Topic ที่ใช้สำหรับจัดการ Device Shadow ของตัวเอง Topic ที่เกี่ยวข้องมีดังนี้

Publish Topic	Description
@shadow/data/update	เป็นการอัพเดทค่าใน Shadow Data โดยส่ง Payload ในรูปแบบ JSON มีรูปแบบคือ {"data":{"fieldname1": value1, "field name 2": value 2, ..., "field name n": value n }}

ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย MQTT Box

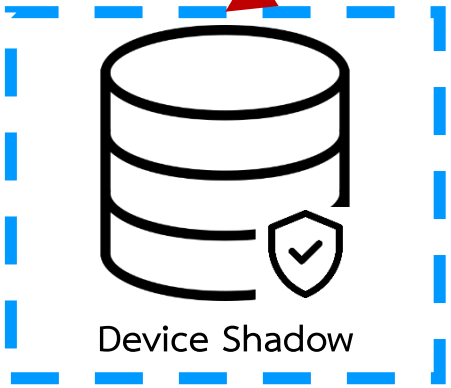
MQTTBox

Topic = @shadow/data/update

Payload = { "data" : { "temperature" : 25, "humidity" : 65, "light" : 120 } }

NETPIE MQTT Broker

ทำการทดสอบส่งข้อมูลจาก MQTT Box เพื่อให้เห็นภาพการจัดเก็บข้อมูลบน Device Shadow



ข้อมูลที่ถูกจัดเก็บลง Device Shadow

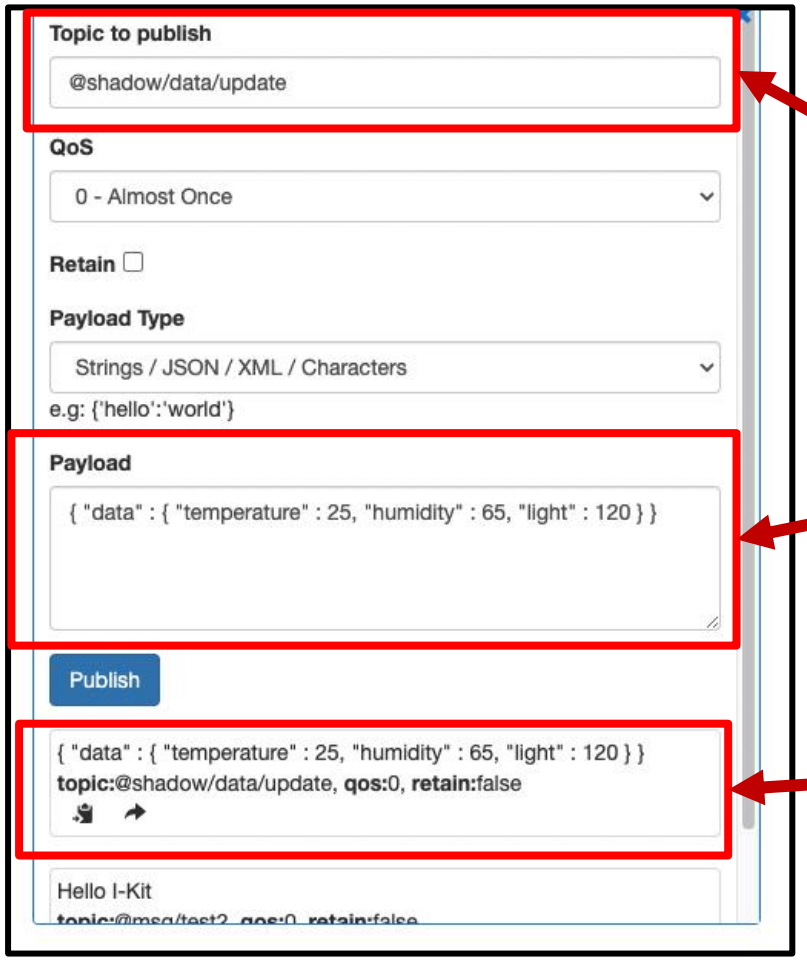
Temperature : 25

Humidity : 65

Light : 120

ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย MQTT Box



ตั้งค่า Topic ให้เป็น @shadow/data/update

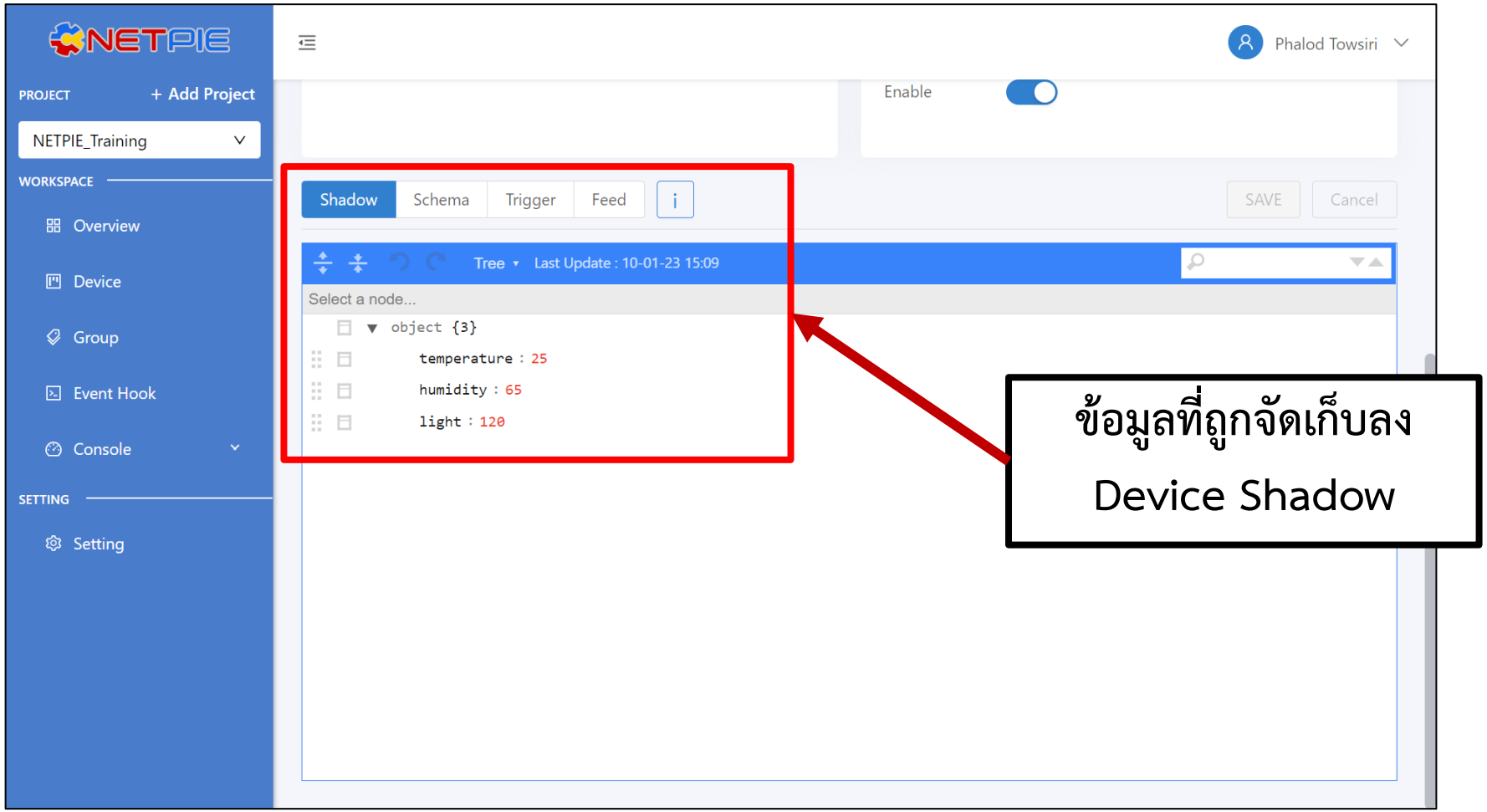
ตั้งค่าข้อความเป็น

{ "data" : { "temperature" : 25, "humidity" : 65, "light" : 120 } }

สถานะการส่งข้อมูล

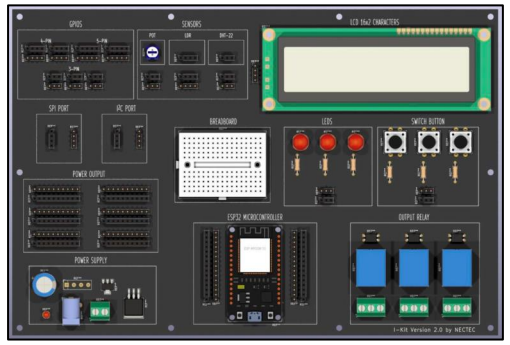
ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย MQTT Box



ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย ESP32



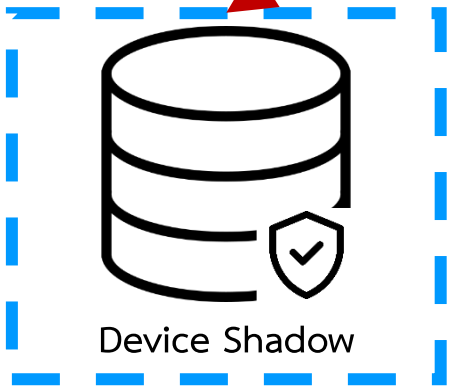
ESP32

Topic = @shadow/data/update

Payload = { "data" : { "light" : 120 } }

NETPIE MQTT Broker

ทดสอบส่งค่าแสงด้วย ESP32



ข้อมูลที่ถูกจัดเก็บลง Device Shadow

Light : 120

ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย ESP32

1 ส่วนที่ 1 การเรียกใช้ Library และ ประกาศตัวแปร

การใช้งานส่วนมากจะเหมือนกับตัวอย่างก่อนหน้านี้

การประกาศขาสำหรับใช้งานกับเซนเซอร์แสง และค่าคงที่

ประกาศตัวแปรใช้สำหรับจัดเก็บข้อมูลก่อนจัดส่งขึ้น MQTT Broker

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Your_SSID";
const char* password = "Your_Password";
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "Client_ID";
const char* mqtt_username = "Token";
const char* mqtt_password = "Secret";

WiFiClient espClient;
PubSubClient client(espClient);

#define ldr 36
float ADC_value = 0.0048828125;
char msg[150];
```

ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย ESP32

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการเชื่อมต่อ MQTT Server

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect(mqtt_Client, mqtt_username, mqtt_password)) {
      Serial.println("connected");
    }
    else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
  }
}
```

ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย ESP32

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการตั้งค่าต่างๆ

```
void setup() {
  Serial.begin(9600);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.setServer(mqtt_server, mqtt_port);
  pinMode(ldr, INPUT);
}
```

คำสั่งประกาศการใช้งานขา ldr เป็น input

ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย ESP32

3 ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  float ldr_data = analogRead(ldr);
  int lux = int((250.000000/(ADC_value*ldr_data))-50.000000);
  Serial.println("Light Value = " + String(lux) + " Lux unit");
  String data = "{\"data\": {\"light\":" + String(lux) + "}}";
  Serial.println(data);
  data.toCharArray(msg, (data.length() + 1));
  client.publish("@shadow/data/update", msg);
  delay(5000);
}
```

ฟังก์ชันการทำงานหลัก

ชุดคำสั่งสำหรับการอ่านค่าและแปลงค่าแสง

ชุดคำสั่งจัดการข้อมูลก่อนส่งขึ้น Shadow Topic มีขั้นตอนดังนี้

1. ทำการจัดรูปข้อมูลให้อยู่ในรูป { "data" : { "light" : 120 } } แล้วจัดเก็บลงตัวแปรที่ชื่อว่า data
2. ทำการแปลงข้อมูลให้อยู่ในรูป CharArray ไว้ในตัวแปร msg เนื่องจากตอนนี้ตัวแปร data เป็นข้อมูลประเภท obj ซึ่งไม่ถูกเงื่อนไขของคำสั่ง client.publish

คำสั่งส่งข้อมูลไปยัง Shadow Topic

ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย ESP32

การส่งข้อความในรูปแบบ JSON บน Arduino IDE

ข้อความที่ต้องการส่งคือ

{ "data" : {"light" : 120 }}

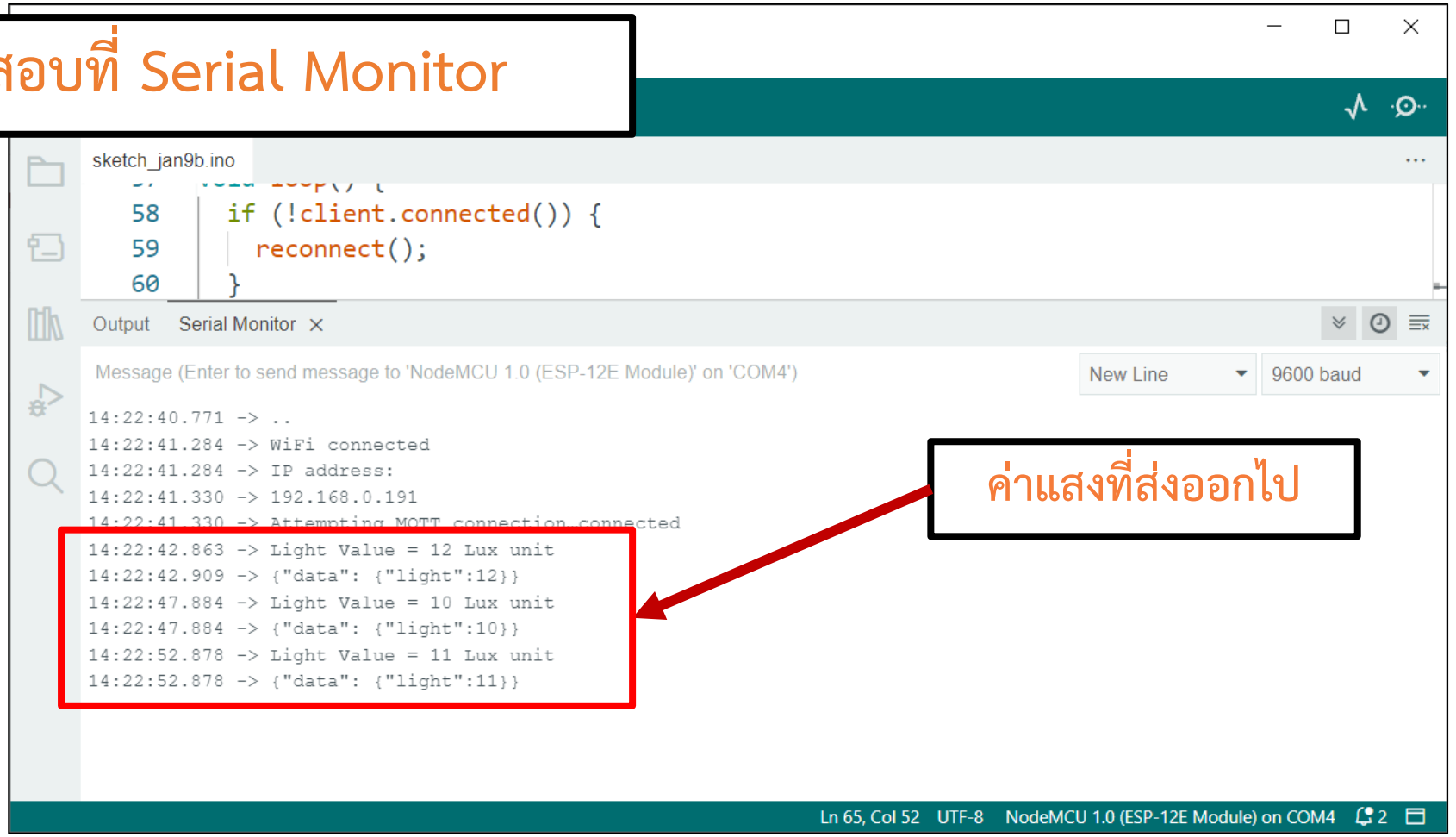
แต่บน Arduino IDE มอง " เป็นการประกาศข้อความหรือ String วิธีแก้คือการนำ \ ไว้ข้างหน้า " เพื่อให้โปรแกรมมอง " เป็นส่วนหนึ่งของข้อความ

```
String data = "{\"data\": {\"light\":" + String(lux) + "}}";
```

ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย ESP8266

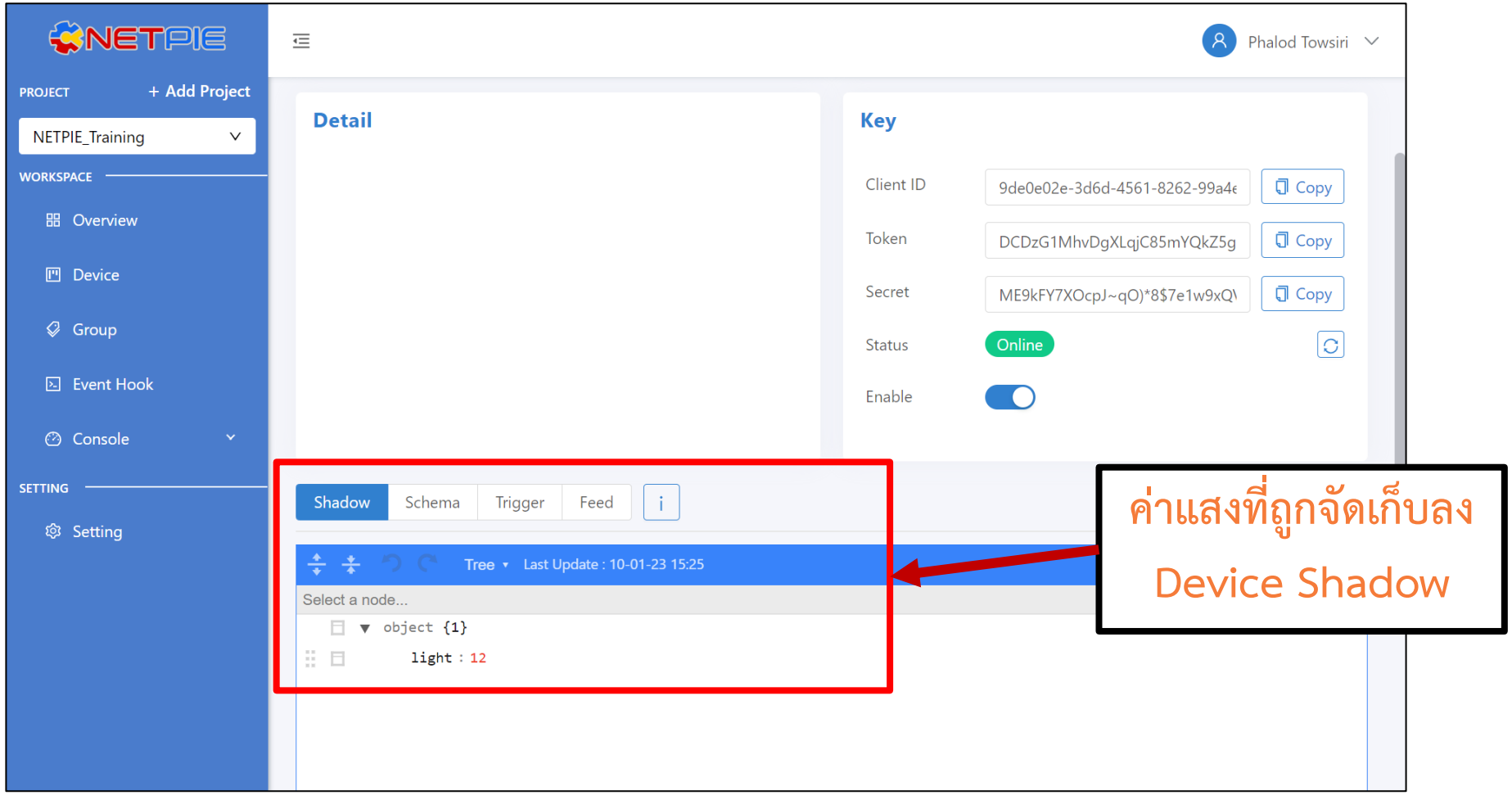
ตรวจสอบที่ Serial Monitor



ค่าแสงที่ส่งออกไป

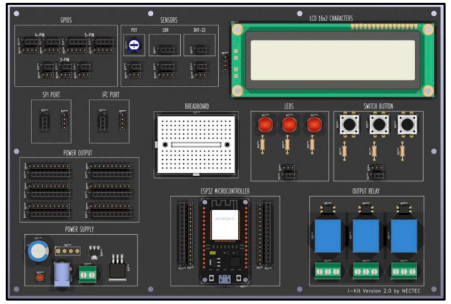
ใบงานที่ 4.3 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการส่งข้อมูลไปยัง Device Shadow ด้วย ESP8266



ใบงานที่ 4.3 ขั้นตอนการทดลอง (เสริม)

การทดลองเสริม



ESP32

Topic = @shadow/data/update

Payload = { "data" : { "temperature" : 25, "humidity" : 65, "light" : 120 } }

NETPIE MQTT Broker

ทดสอบส่งค่าอุณหภูมิ ความชื้น และแสง ด้วย ESP32

Device Shadow

ข้อมูลที่ถูกจัดเก็บลง Device Shadow

Temperature : 25

Humidity : 65

Light : 120

ใบงานที่ 4.3 ขั้นตอนการทดลอง (เสริม)

การทดลองเสริม

1 ส่วนที่ 1 การเรียกใช้ Library และ ประกาศตัวแปร

เรียกใช้งาน library DHT สำหรับอ่านค่าเซนเซอร์ DHT

การประกาศและกำหนดค่าต่างๆสำหรับเซนเซอร์ DHT

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

const char* ssid = "lucifer_wifi";
const char* password = "lucifer69X";
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "4afbad5c-201c-4be5-9771-c7d499748624";
const char* mqtt_username = "pu5ehH6TiF1Sqv7JZvnsKUUKL53TTy7x";
const char* mqtt_password = "xlErH)sca437I3_eN#W8lsUcLZUrkle)";

WiFiClient espClient;
PubSubClient client(espClient);

#define ldr 36
float ADC_value = 0.0048828125;

char msg[150];

#define DHTPIN 34
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

ใบงานที่ 4.3 ขั้นตอนการทดลอง (เสริม)

การทดลองเสริม

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการเชื่อมต่อ MQTT Server

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect(mqtt_Client, mqtt_username, mqtt_password)) {
      Serial.println("connected");
    }
    else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
  }
}
```

ใบงานที่ 4.3 ขั้นตอนการทดลอง (เสริม)

การทดลองเสริม

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการตั้งค่าต่างๆ

```
void setup() {
  Serial.begin(9600);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.setServer(mqtt_server, mqtt_port);
  pinMode(ldr, INPUT);
  dht.begin();
}
```

การเริ่มต้นใช้งานเซนเซอร์ DHT

ใบงานที่ 4.3 ขั้นตอนการทดลอง (เสริม)

การทดลองเสริม

3 ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

ฟังก์ชันการทำงานหลัก

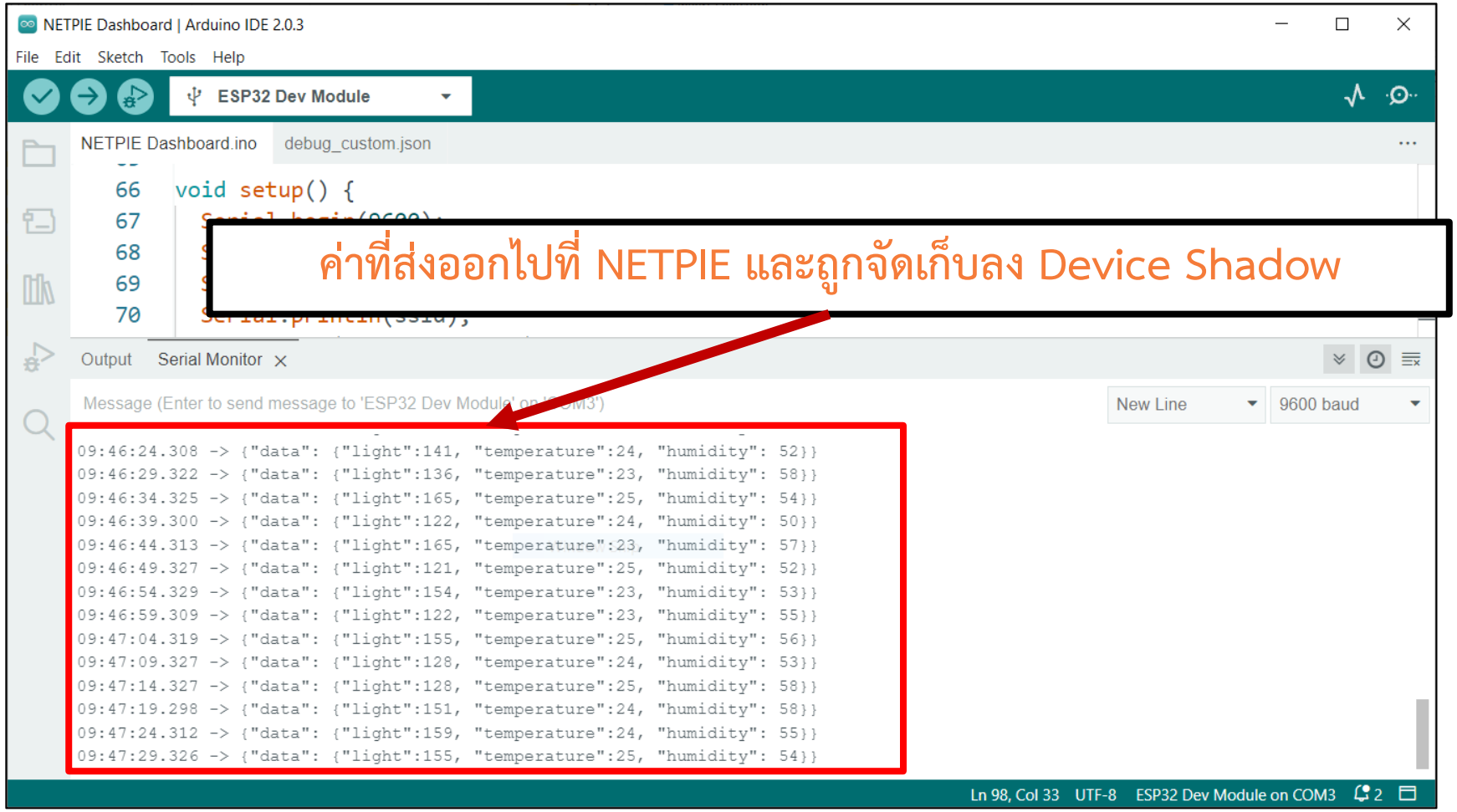
การประกาศตัวแปรและอ่านค่าเซนเซอร์ DHT

นำข้อมูลอุณหภูมิและความชื้นมาเก็บไว้ใน ตัวแปร data ร่วมกับค่าแสง

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  float ldr_data = analogRead(ldr);
  int lux = int((250.000000/(ADC_value*ldr_data))-50.000000);

  int humidity = dht.readHumidity();
  int temperature = dht.readTemperature();

  String data = "{\"data\": {\"light\":" + String(lux) + ", \"temperature\":" + String(temperature) + ",
\"humidity\": " +  String(humidity) + "}}";
  Serial.println(data);
  data.toCharArray(msg, (data.length() + 1));
  client.publish("@shadow/data/update", msg);
  delay(5000);
}
```

ใบงานที่ 4.3 ขั้นตอนการทดลอง (เสริม)

การทดลองเสริม



ใบงานที่ 4.3 คำถามท้ายหน่วยการเรียนรู้

คำถามท้ายหน่วยการเรียนรู้ที่ 8

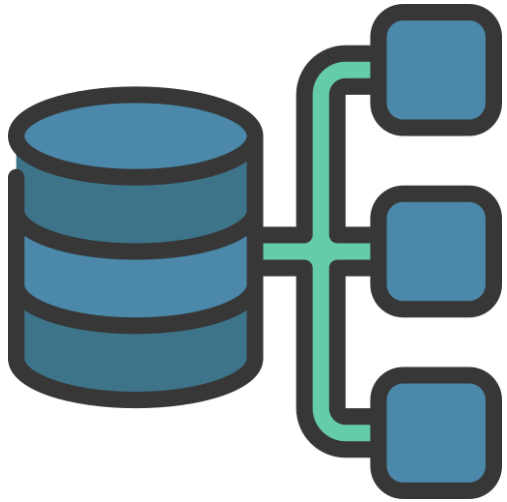
จงเขียนโปรแกรมส่งข้อมูลต่างๆซึ่งอยู่บน ESP32 ไปยัง Device Shadow โดยข้อมูลที่จะถูกส่งขึ้นไปมีดังนี้

1. ค่าแรงดันจากตัวต้านทานปรับค่าได้
2. ค่าอากาศจากเซนเซอร์ MQ2
3. สถานะ LED1
4. สถานะ LED2
5. สถานะ LED3

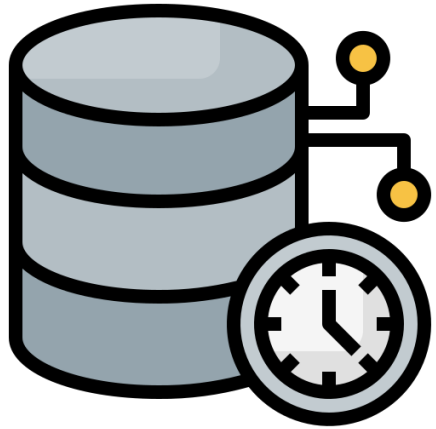
โดย LED1, LED2 และ LED3 นั้นสั่งเปิด/ปิด โดยใช้ MQTT Box

***Tips ปกติแล้วการส่งสถานะของ LED นั้นเราจะนิยมส่งสถานะก็ต่อเมื่อมีการเปลี่ยนแปลงสถานะของอุปกรณ์จริงๆ หรือเมื่ออุปกรณ์ได้รับข้อความแล้วทำการส่งสถานะทันที

ใบงานที่ 4.4 การส่งข้อมูลจาก ESP32 เพื่อจัดเก็บบนฐานข้อมูลเวลาบน NETPIE2020



Device Schema



Device Feed

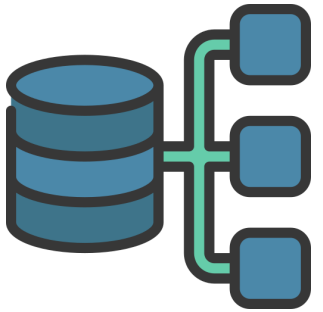
ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

แผงผังข้อมูล (Device Schema)

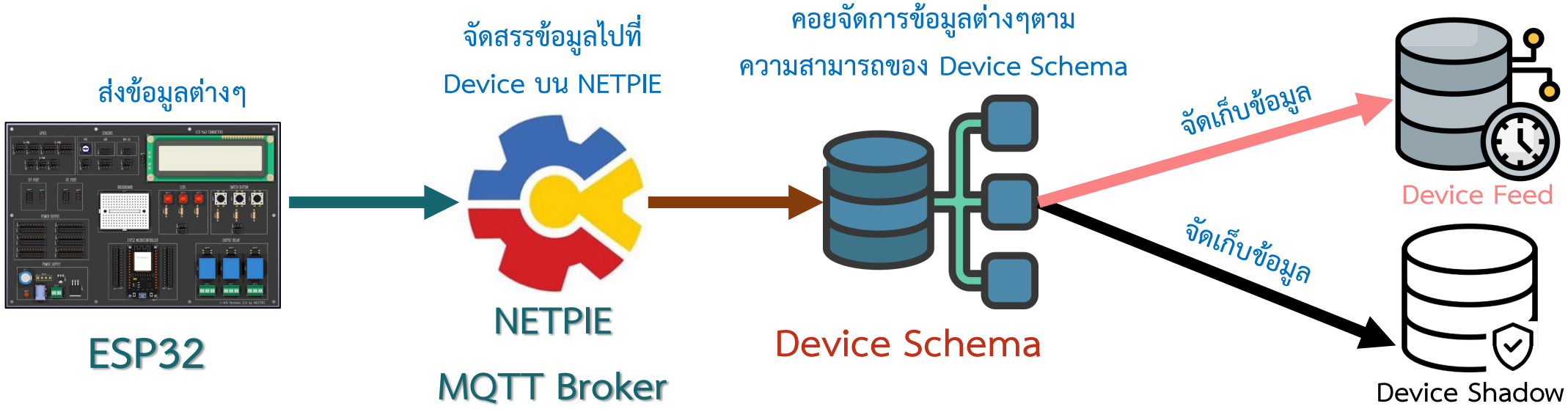
สำหรับอุปกรณ์ที่ต้องมีการจัดการข้อมูล แนะนำให้สร้าง Device Schema ของข้อมูลเตรียมไว้

Device Schema คือ แผงผังข้อมูลที่กำหนดไว้เพื่อใช้กำกับ Device Shadow โดย Device Schema เสมือนเป็น Device Template ทำให้ Server สามารถ

- การตรวจสอบชนิดข้อมูล (Data Validation)
- การแปลงข้อมูล (Data Transformation) เช่น เปลี่ยนหน่วยของข้อมูล เป็นต้น
- การเก็บข้อมูลลงใน Timeseries Database (Device Feed)



Device Schema

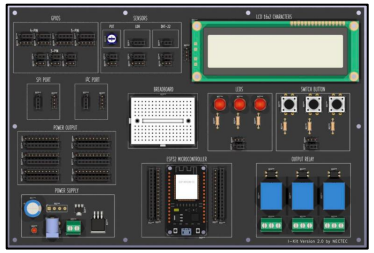


ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

ยกตัวอย่างการทำงานของ Device Schema

หาก Device Schema ทำการตั้งค่าไว้ว่า

1. ข้อมูลที่ต้องการจัดเก็บคือ Temperature (number), Humidity (number) และ Place (string)
2. มีการแปลงหน่วยข้อมูลของ Temperature จาก องศาเซลเซียส (°C) เป็น องศาฟาเรนไฮต์ (°F)
3. กำหนดให้ทุกข้อมูลจัดเก็บเป็นเวลา 7 วัน (ข้อมูลทั้งหมดที่ถูกจัดเก็บใน Device Feed จะมีอายุ 7 วัน)



ESP32

ข้อมูลที่ถูกส่งจากอุปกรณ์

Topic = @shadow/data/update

Payload = { "data" : { "temperature" : 25, "humidity" : 65, "light" : 120, "place" : "home" } }

NETPIE MQTT Broker

Device Schema

ข้อมูลที่จะถูกจัดเก็บลง Device Shadow

temperature : 25

humidity : 65

light : 120

place : "home "

Device Feed

ข้อมูลใน Device Feed

temperature : 77

humidity : 65

~~light : 120~~

place : "home "

Temperature และ Light ถูกจัดการโดย Device Schema ตามค่าที่กำหนด

Device Shadow

ข้อมูลใน Device Shadow

temperature : 77

humidity : 65

~~light : 120~~

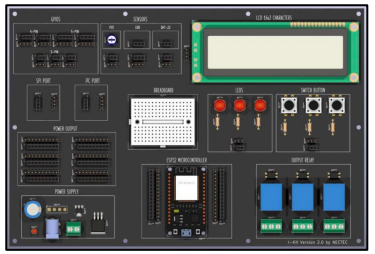
place : "home "

ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

ยกตัวอย่างการทำงานของ Device Schema

หาก Device Schema ทำการตั้งค่าไว้ว่า

1. ข้อมูลที่ต้องการจัดเก็บคือ Temperature (number), Humidity (number) และ Place (string)
2. มีการแปลงหน่วยข้อมูลของ Temperature จาก องศาเซลเซียส (°C) เป็น องศาฟาเรนไฮต์ (°F)
3. กำหนดให้ทุกข้อมูลจัดเก็บเป็นเวลา 7 วัน (ข้อมูลทั้งหมดที่ถูกจัดเก็บใน Device Feed จะมีอายุ 7 วัน)



ESP32

ข้อมูลที่ถูกส่งจากอุปกรณ์

Topic = @shadow/data/update

Payload = { "data" : { "temperature" : 25, "humidity" : 65, "light" : 120, "place" : 1234 } }

ข้อมูลของ place ถูกเปลี่ยนประเภทข้อมูลเป็น number

NETPIE MQTT Broker

Device Schema

ข้อมูลที่จะถูกจัดเก็บลง Device Shadow

temperature : 25

humidity : 65

light : 120

place : "home "

Device Feed

ข้อมูลใน Device Feed

temperature : 77

humidity : 65

~~light : 120~~

~~place : "home "~~

Place จะไม่ถูกจัดเก็บลง Device Shadow และ Device Feed

Device Shadow

ข้อมูลใน Device Shadow

temperature : 77

humidity : 65

~~light : 120~~

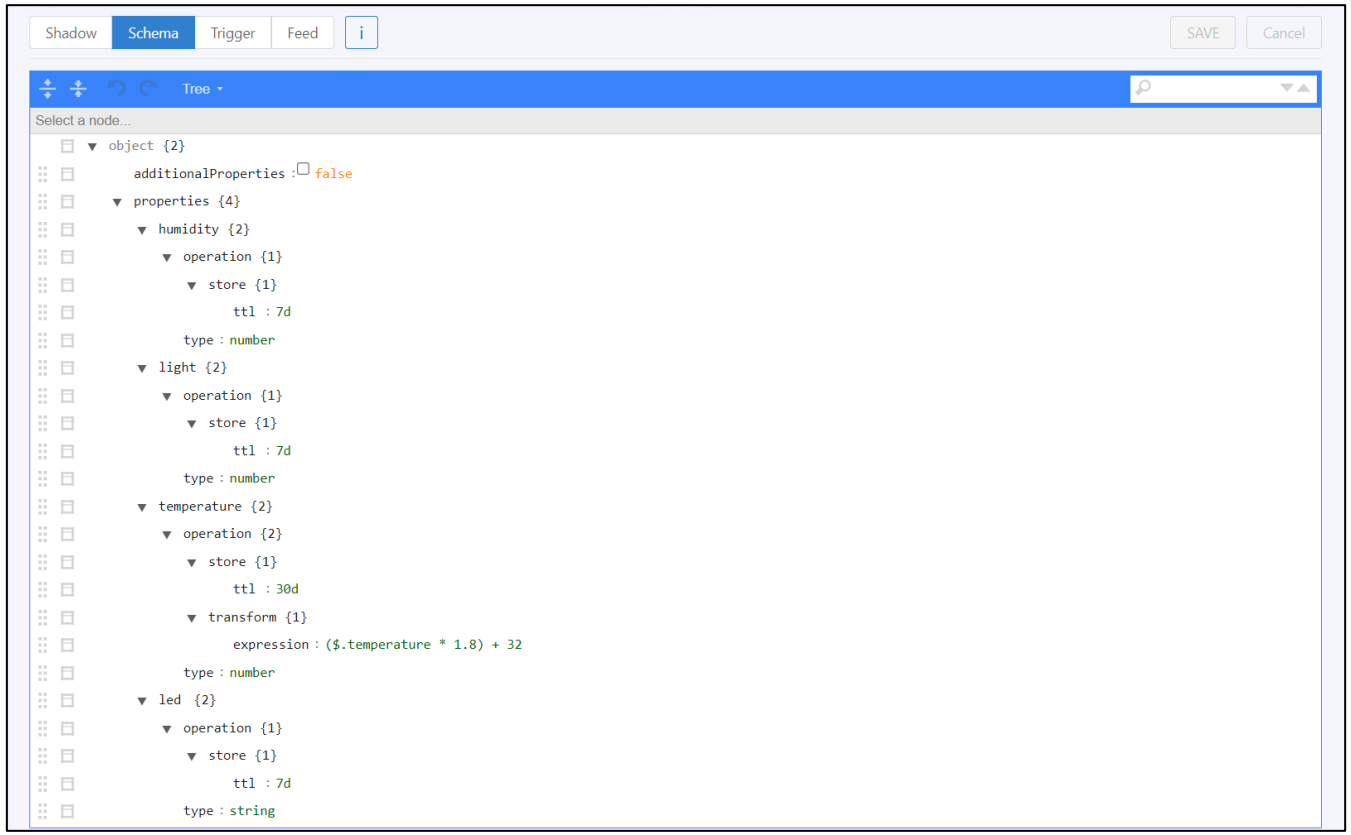
~~place : "home "~~

ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การประกาศ Device Schema

การประกาศ Device Schema จะอยู่ในรูปแบบของ JSON เสมอ

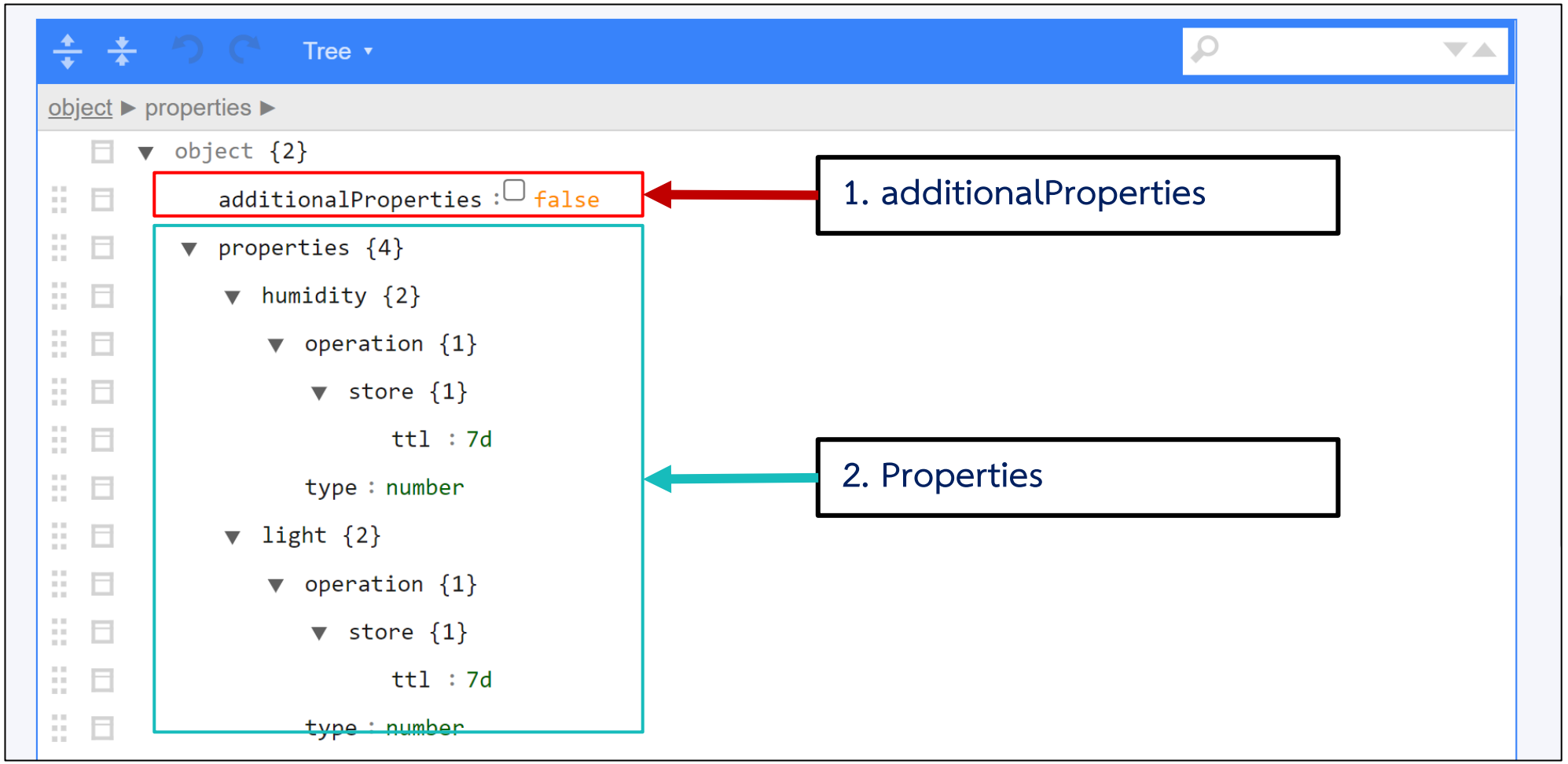
ตัวอย่างการประกาศ Device Schema



ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

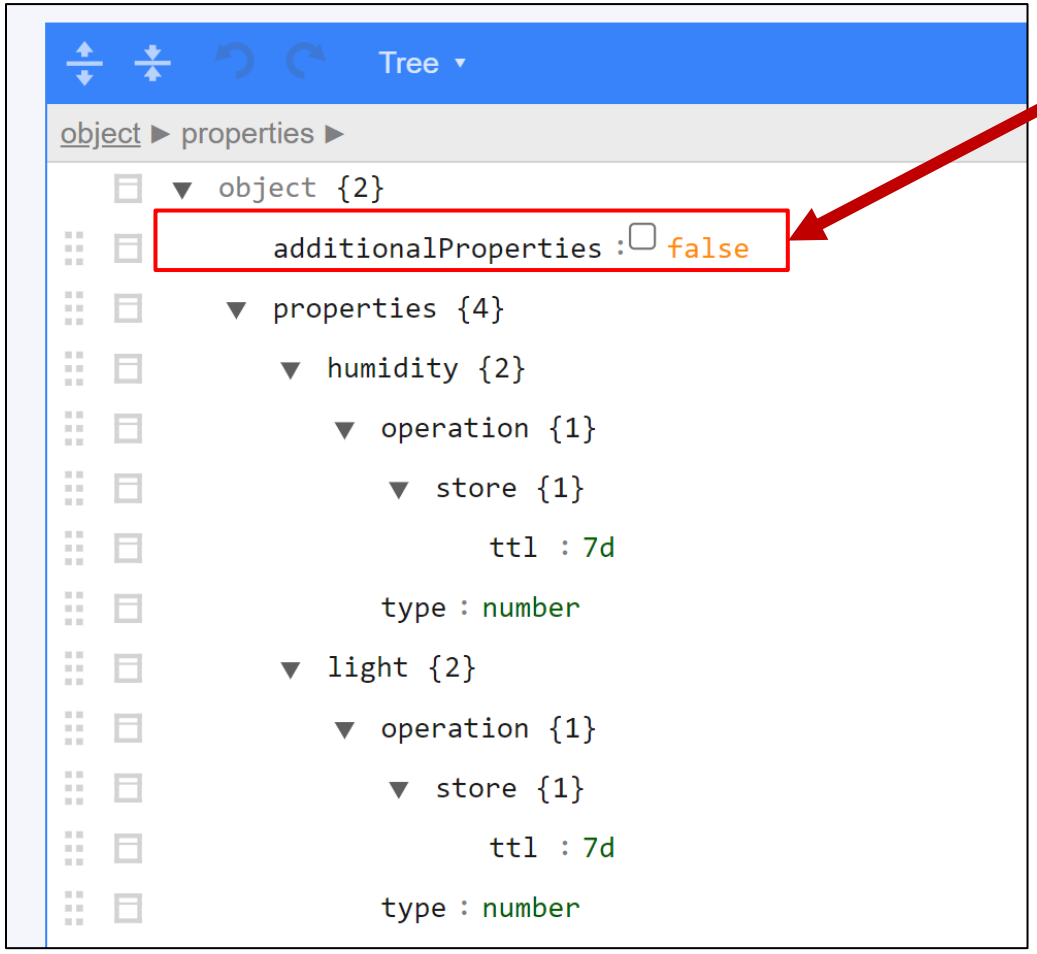
การประกาศ Device Schema

Device Schema จะประกอบด้วย 2 ส่วนหลักคือ



ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การประกาศ Device Schema



1. additionalProperties

คือ สถานะการอนุญาตให้บันทึกข้อมูลลง Device Shadow และ Device Feed ในกรณีที่ข้อมูลไม่ตรงตามที่กำหนด Properties มี 2 สถานะ คือ

true คือ อนุญาตให้บันทึกลง Shadow หรือ Timeseries Database

false คือ ไม่อนุญาตให้บันทึกเฉพาะส่วนที่ไม่ตรงตาม Properties

อย่างเช่นในตัวอย่าง properties 2 ค่าคือ place และ temperature

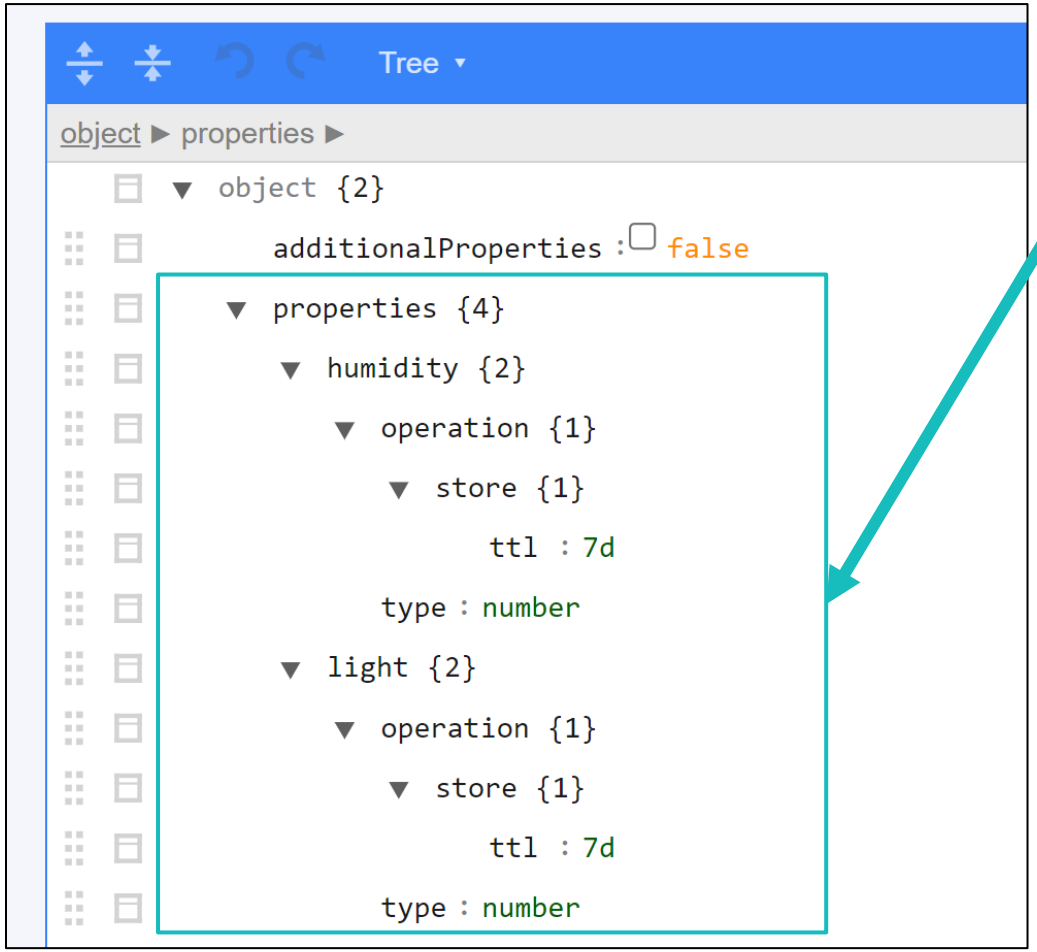
ถ้าข้อมูลที่ส่งมาคือ temperature, place, light

additionalProperties = true จะจัดเก็บทั้ง temperature, humidity และ light

additionalProperties = false จะจัดเก็บเพียง temperature, humid

ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การประกาศ Device Schema

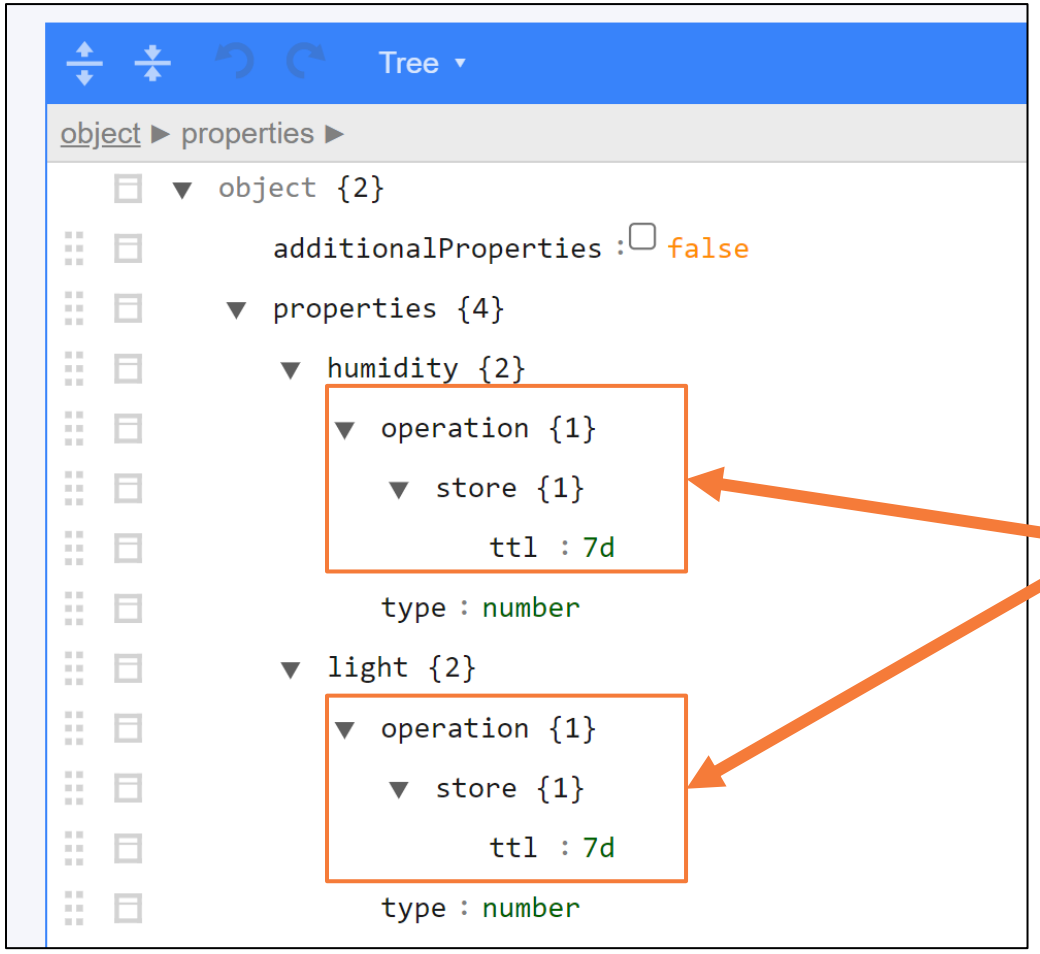


2. Properties

กำหนดชื่อฟิลด์ (ตัวอย่างคือ place และ temperature) และกำหนดคุณสมบัติของแต่ละฟิลด์ซึ่งแบ่งเป็น 2 ส่วนคือ

ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การประกาศ Device Schema



2. Properties

กำหนดชื่อฟิลด์ (ตัวอย่างคือ place และ temperature) และกำหนดคุณสมบัติของแต่ละฟิลด์ซึ่งแบ่งเป็น 2 ส่วนคือ

1. **Operation** สำหรับตั้งค่าการจัดการข้อมูลในฟิลด์นั้นๆ ประกอบไปด้วย

store สำหรับตั้งค่าการเก็บข้อมูลลง Timeseries Database

ttl คือ ระยะเวลาของการเก็บข้อมูลใน Timeseries Database ซึ่งแต่ละข้อมูลมีอายุการเก็บครบตามกำหนดจะถูกลบทิ้งอัตโนมัติ

ถ้าต้องการจัดเก็บข้อมูลระบบจำเป็นต้องกำหนดค่านี้ มีหน่วยเป็น ms(มิลลิวินาที), s(วินาที), m(นาที), h(ชั่วโมง), d(วัน), y(ปี)

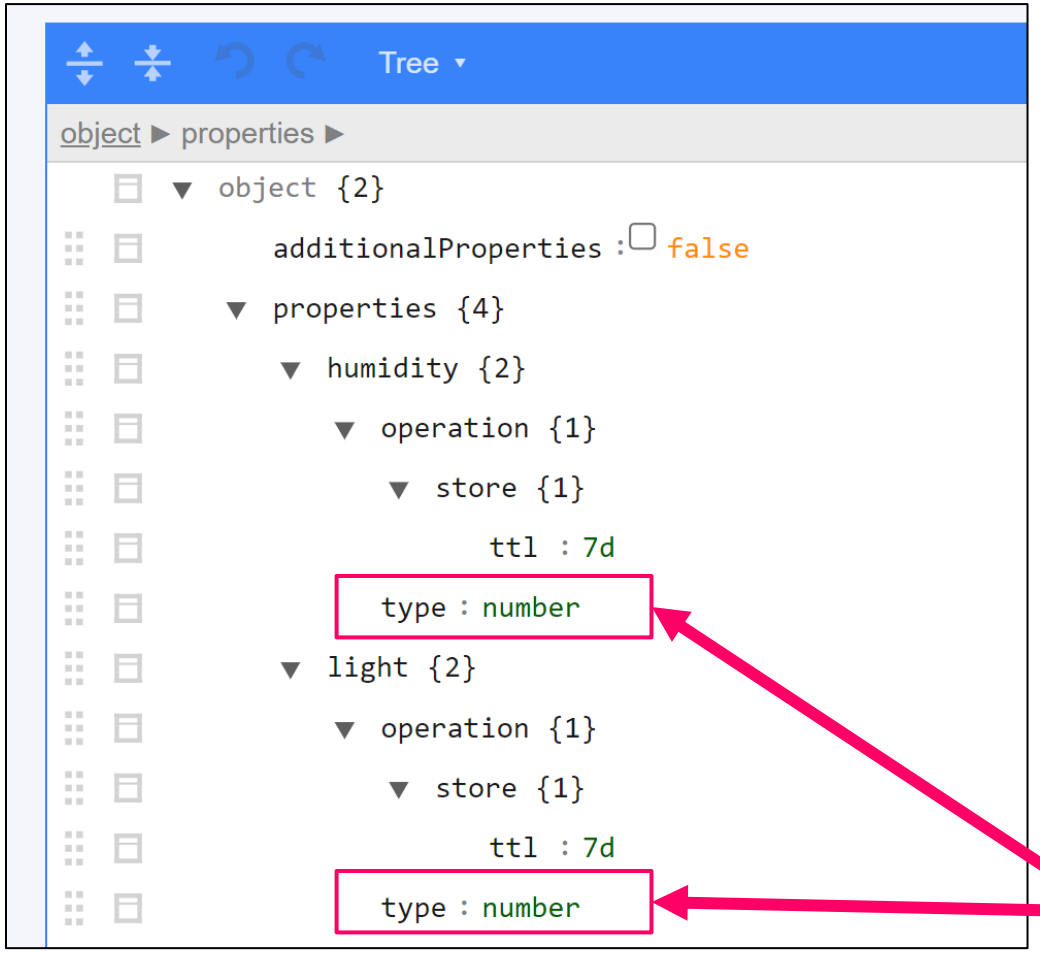
Transform การแปลงข้อมูลก่อนจัดเก็บ

expression คือ สูตรหรือวิธีการแปลงข้อมูลก่อนจัดเก็บ

ตัวอย่าง แปลงหน่วยอุณหภูมิจากเซลเซียสเป็นฟาเรนไฮต์ = ($.temperature*1.8) + 32

ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การประกาศ Device Schema



2. Properties

กำหนดชื่อฟิลด์ (ตัวอย่างคือ place และ temperature) และกำหนดคุณสมบัติของแต่ละฟิลด์ซึ่งแบ่งเป็น 2 ส่วนคือ

1. **Operation** สำหรับตั้งค่าการจัดการข้อมูลในฟิลด์นั้นๆ ประกอบไปด้วย

store สำหรับตั้งค่าการเก็บข้อมูลลง Timeseries Database

ttl คือ ระยะเวลาของการเก็บข้อมูลใน Timeseries Database ซึ่งแต่ละข้อมูลมีอายุการเก็บครบตามกำหนดจะถูกลบทิ้งอัตโนมัติ

ถ้าต้องการจัดเก็บข้อมูลระบบจำเป็นต้องกำหนดค่านี้ มีหน่วยเป็น ms(มิลลิวินาที), s(วินาที), m(นาที), h(ชั่วโมง), d(วัน), y(ปี)

Transform การแปลงข้อมูลก่อนจัดเก็บ

expression คือ สูตรหรือวิธีการแปลงข้อมูลก่อนจัดเก็บ

ตัวอย่าง แปลงหน่วยอุณหภูมิจากเซลเซียสเป็นฟาเรนไฮต์ = ($.temperature*1.8) + 32

2. **Type** คือ ชนิดข้อมูลในฟิลด์นั้นๆ ได้แก่ number, string, array, object

ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ MQTT Box

MQTTBox

กำหนด Device Schema ให้สอดคล้องกับข้อมูลที่ MQTT Box จะส่งมาดังนี้

1. ข้อมูลที่จะถูกจัดเก็บคือ temperature (number), humidity (number), light (number) และ place (string)
2. มีการแปลงหน่วยข้อมูลของ Temperature จาก องศาเซลเซียส (°C) เป็น องศาฟาเรนไฮต์ (°F)
3. กำหนดให้ temperature, humidity, light ถูกจัดเก็บเป็นเวลา 7 วัน และ place ถูกจัดเก็บเป็นเวลา 3 วัน

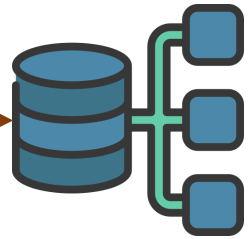
ข้อมูลที่ถูกส่งจาก MQTT Box

Topic = @shadow/data/update

Payload = { "data" : { "temperature" : 25, "humidity" : 65, "light" : 120, "place" : "home" } }

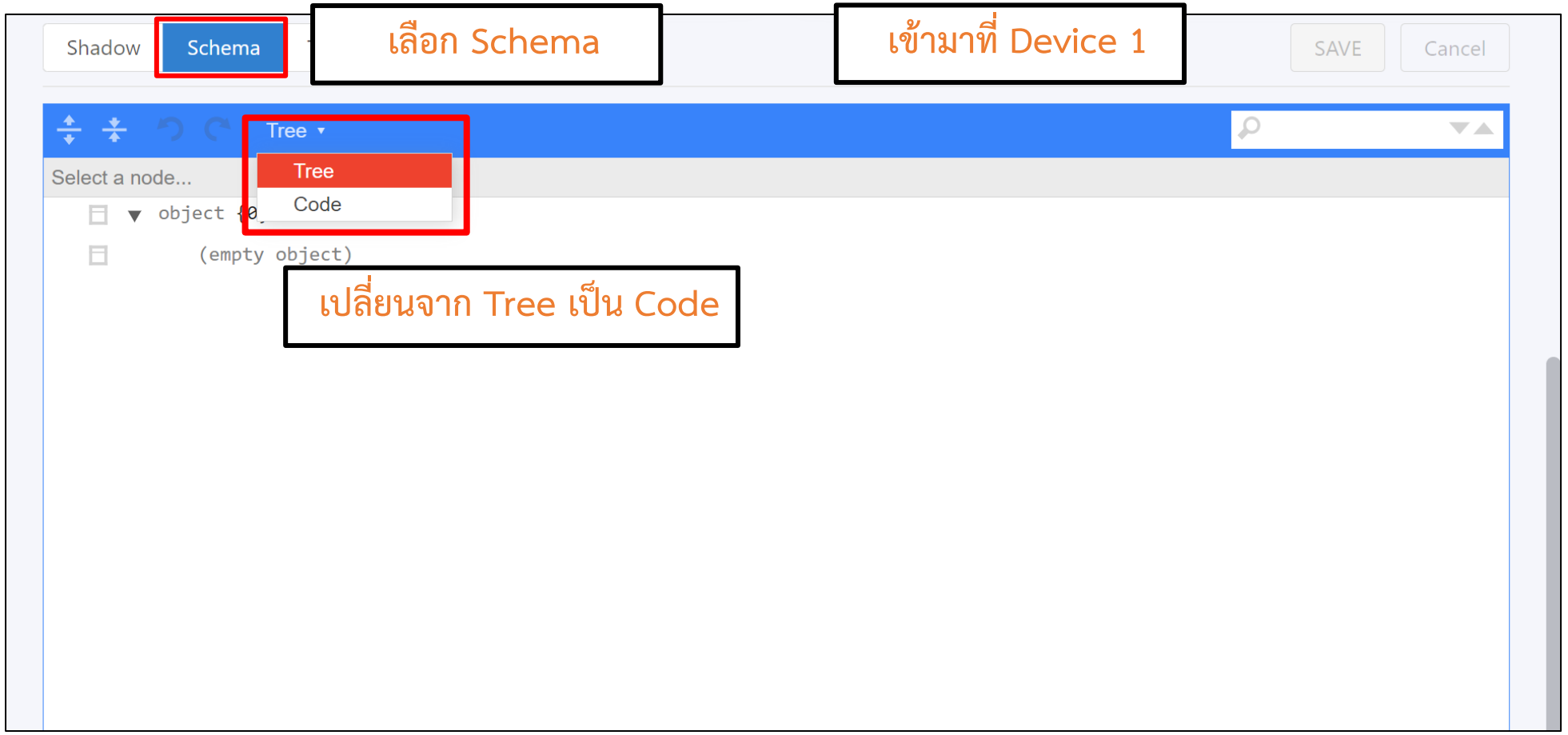
NETPIE MQTT Broker

Device Schema



ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ MQTT Box



ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ MQTT Box

```
{
  "additionalProperties": false,
  "properties": {
    "place": {
      "operation": {
        "store": {
          "ttl": "3d"
        }
      },
      "type": "string"
    },
    "humidity": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "number"
    },
    "light": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "number"
    },
    "temperature": {
      "operation": {
        "store": {
          "ttl": "30d"
        },
        "transform": {
          "expression": "($.temperature * 1.8) + 32"
        }
      },
      "type": "number"
    }
  }
}
```

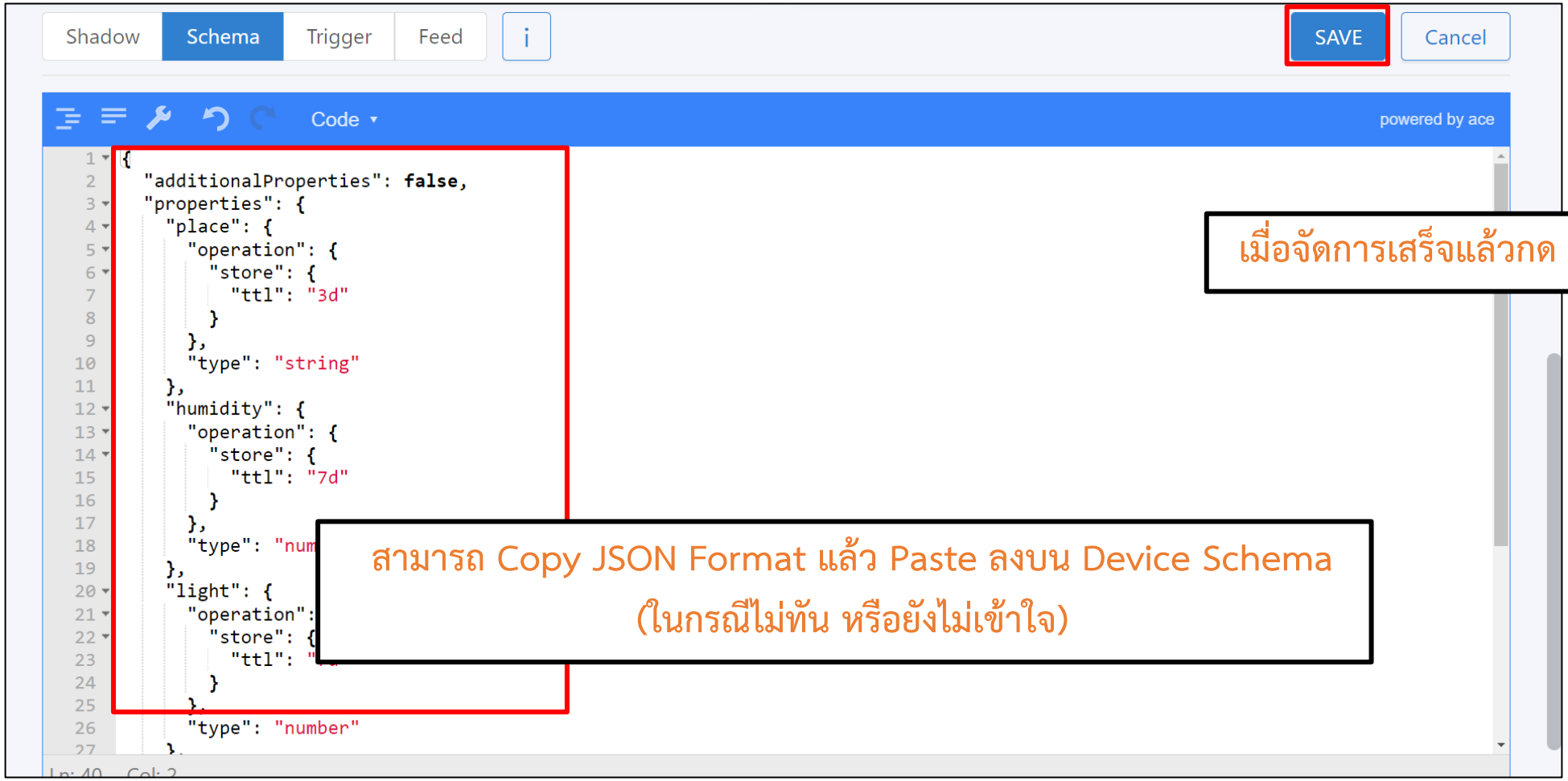
การประกาศฟิลด์ของ place

การประกาศฟิลด์ของ humidity และ light

การประกาศฟิลด์ของ temperature

ใบงานที่ 4.4 ขั้นตอนการทดลอง

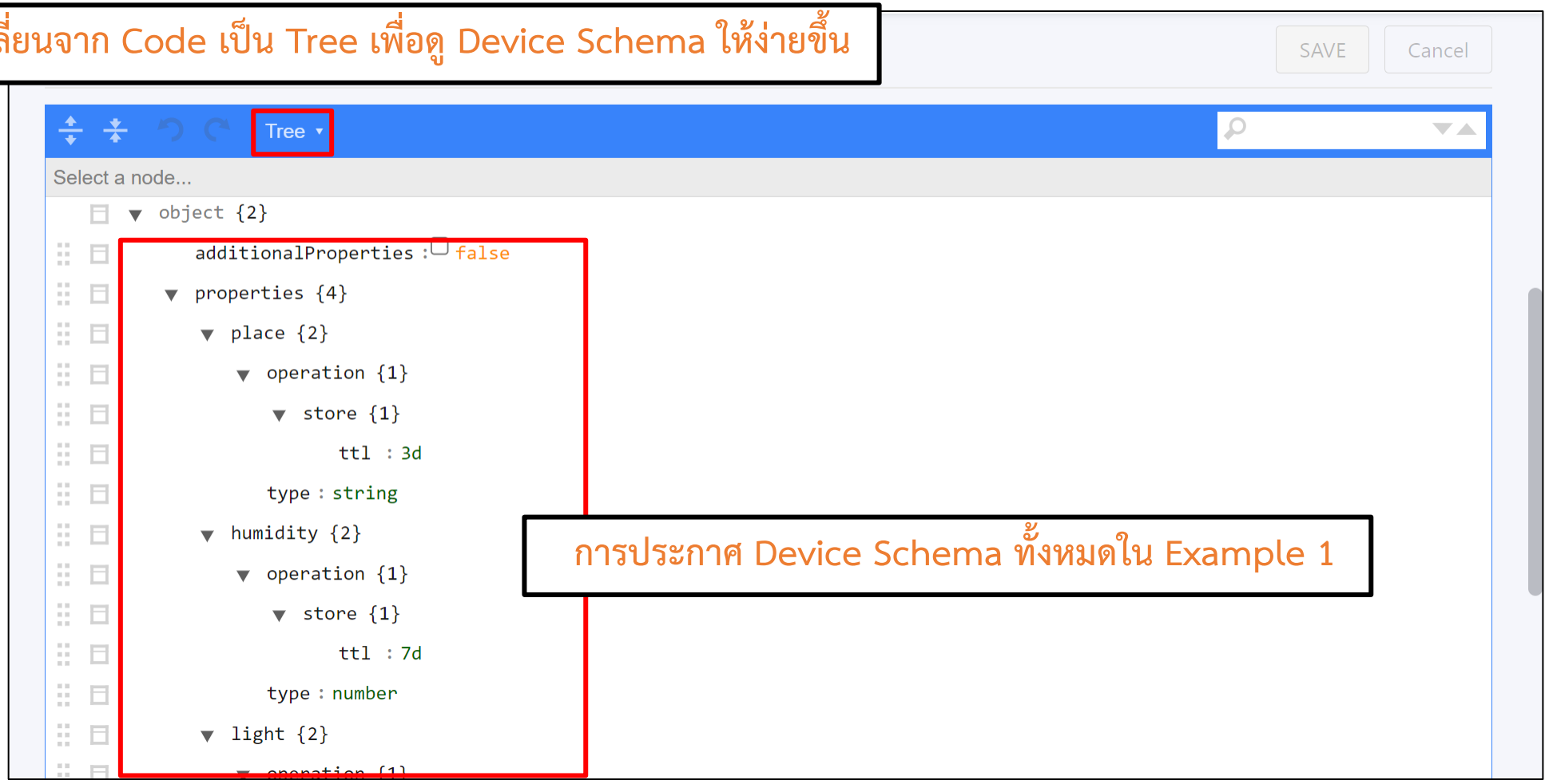
การทดลองที่ 1 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ MQTT Box



ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ MQTT Box

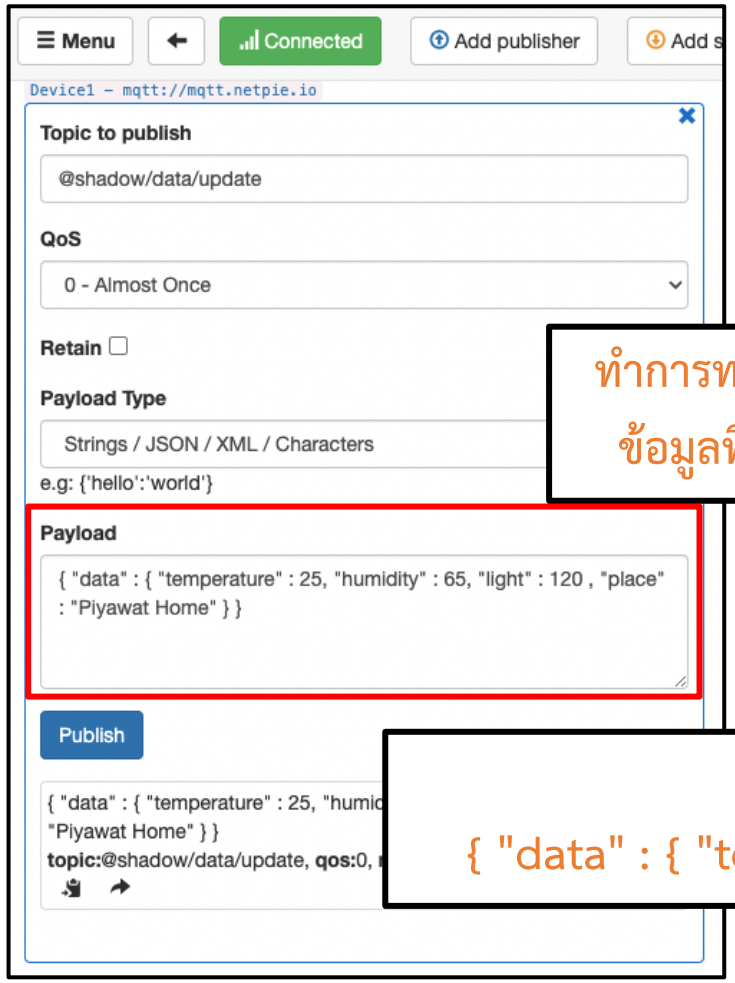
เปลี่ยนจาก Code เป็น Tree เพื่อดู Device Schema ให้ง่ายขึ้น



การประกาศ Device Schema ทั้งหมดใน Example 1

ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ MQTT Box



มาที่ MQTT Box ที่เชื่อมต่อ Device1 ไว้

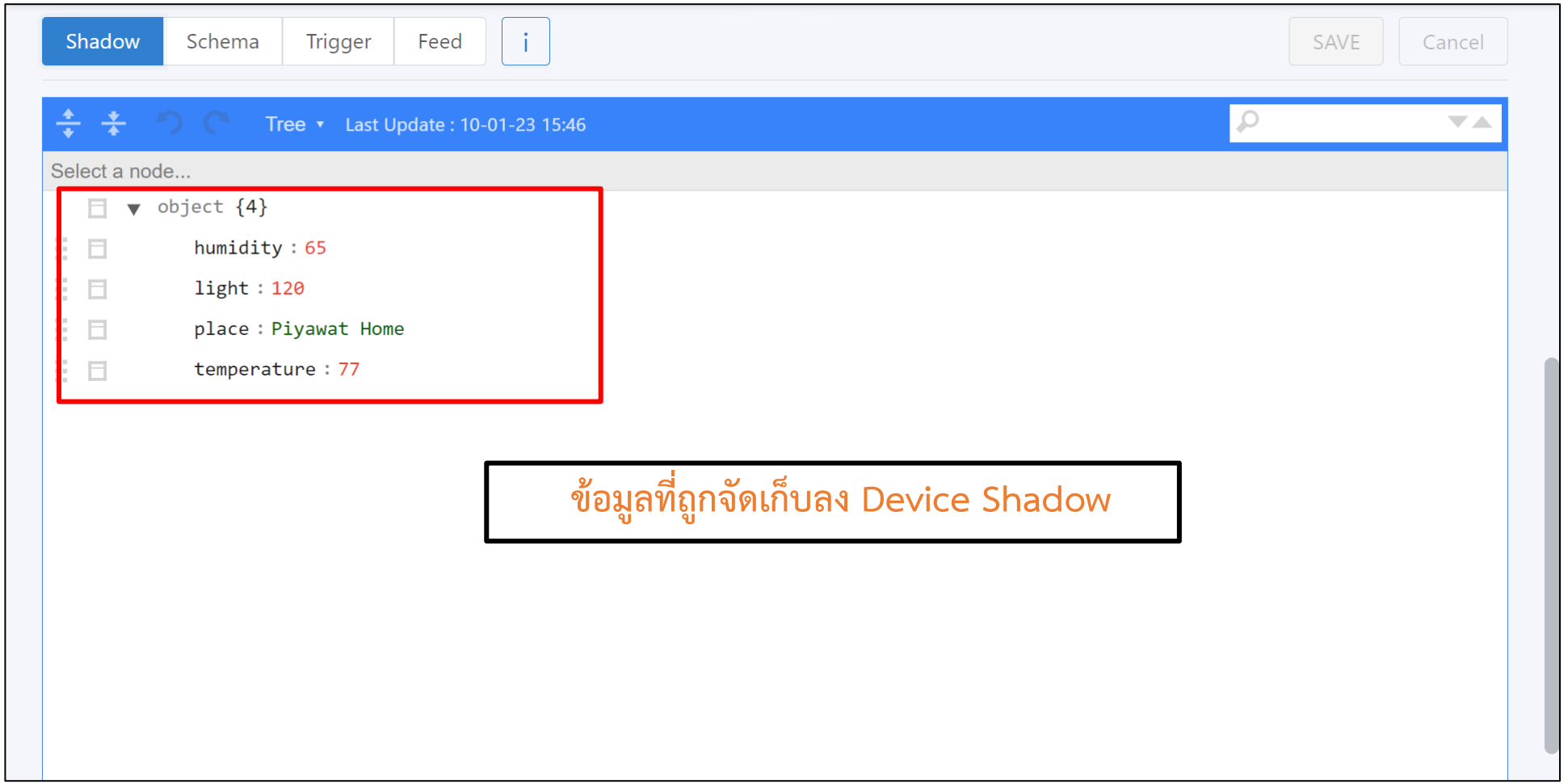
ทำการทดสอบส่งข้อมูลตามตัวอย่างเพื่อดูข้อมูลที่ถูกจัดเก็บลง Device Shadow

Payload สามารถ Copy ได้จากตรงนี้

{ "data" : { "temperature" : 25, "humidity" : 65, "light" : 120 , "place" : "Piyawat Home" } }

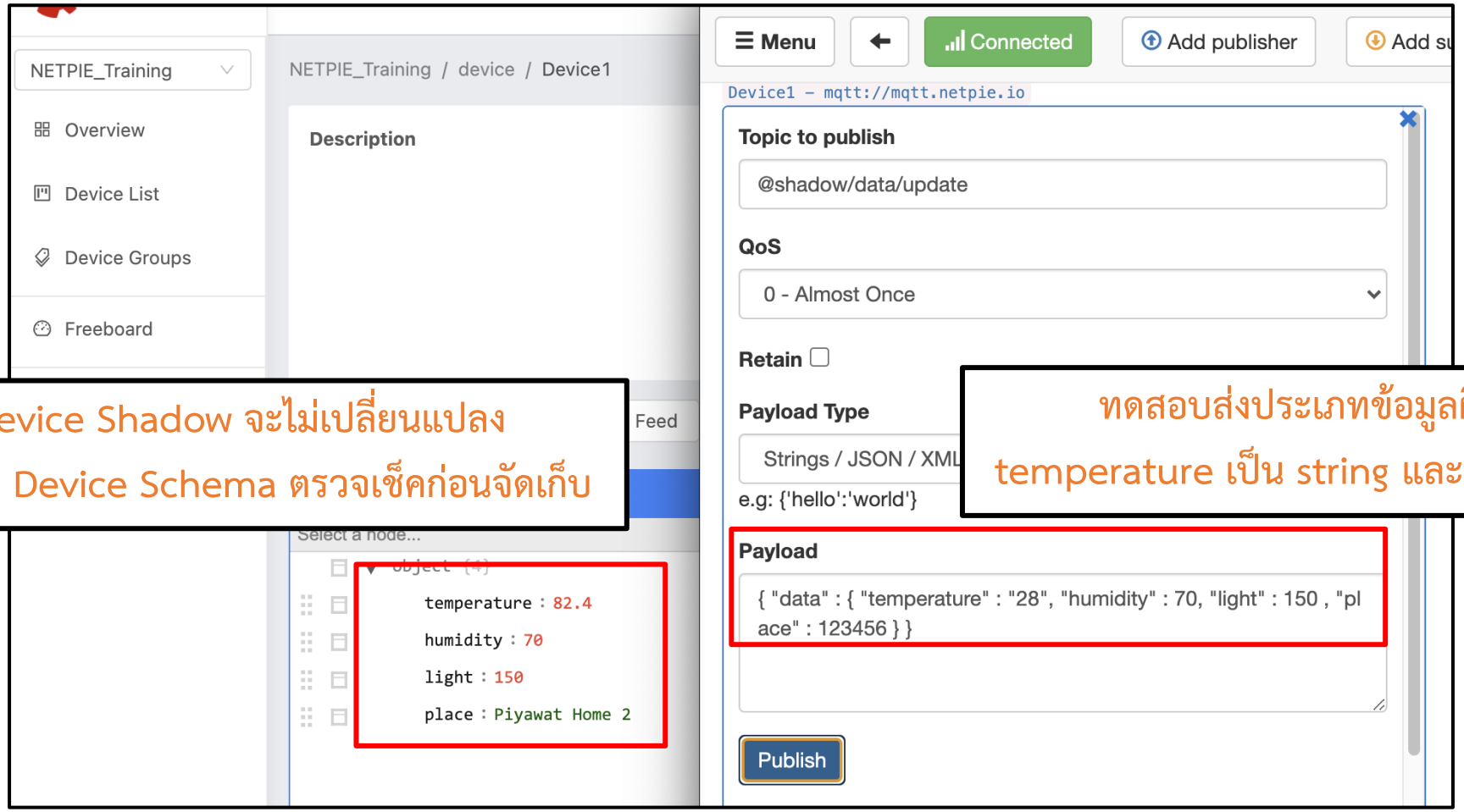
ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ MQTT Box



ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ MQTT Box

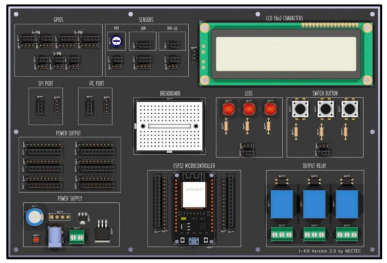


ข้อมูลบน Device Shadow จะไม่เปลี่ยนแปลง
เนื่องจากข้อมูลถูก Device Schema ตรวจเช็คก่อนจัดเก็บ

ทดสอบส่งประเภทข้อมูลผิดขึ้นไป นั่นคือ
temperature เป็น string และ place เป็น number

ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ ESP32



ESP32

กำหนด Device Schema ให้สอดคล้องกับข้อมูลที่ ESP32 จะส่งมาดังนี้

1. ข้อมูลที่จะถูกจัดเก็บคือ temperature (number), humidity (number) และ light (number)
2. มีการแปลงหน่วยข้อมูลของ Temperature จาก องศาเซลเซียส (°C) เป็น องศาฟาเรนไฮต์ (°F)
3. กำหนดให้ temperature, humidity, light ถูกจัดเก็บเป็นเวลา 7 วัน

ข้อมูลที่ถูกส่งจาก MQTT Box

Topic = @shadow/data/update

Payload = { "data" : { "temperature" : 25,

"humidity" : 65, "light" : 120} }

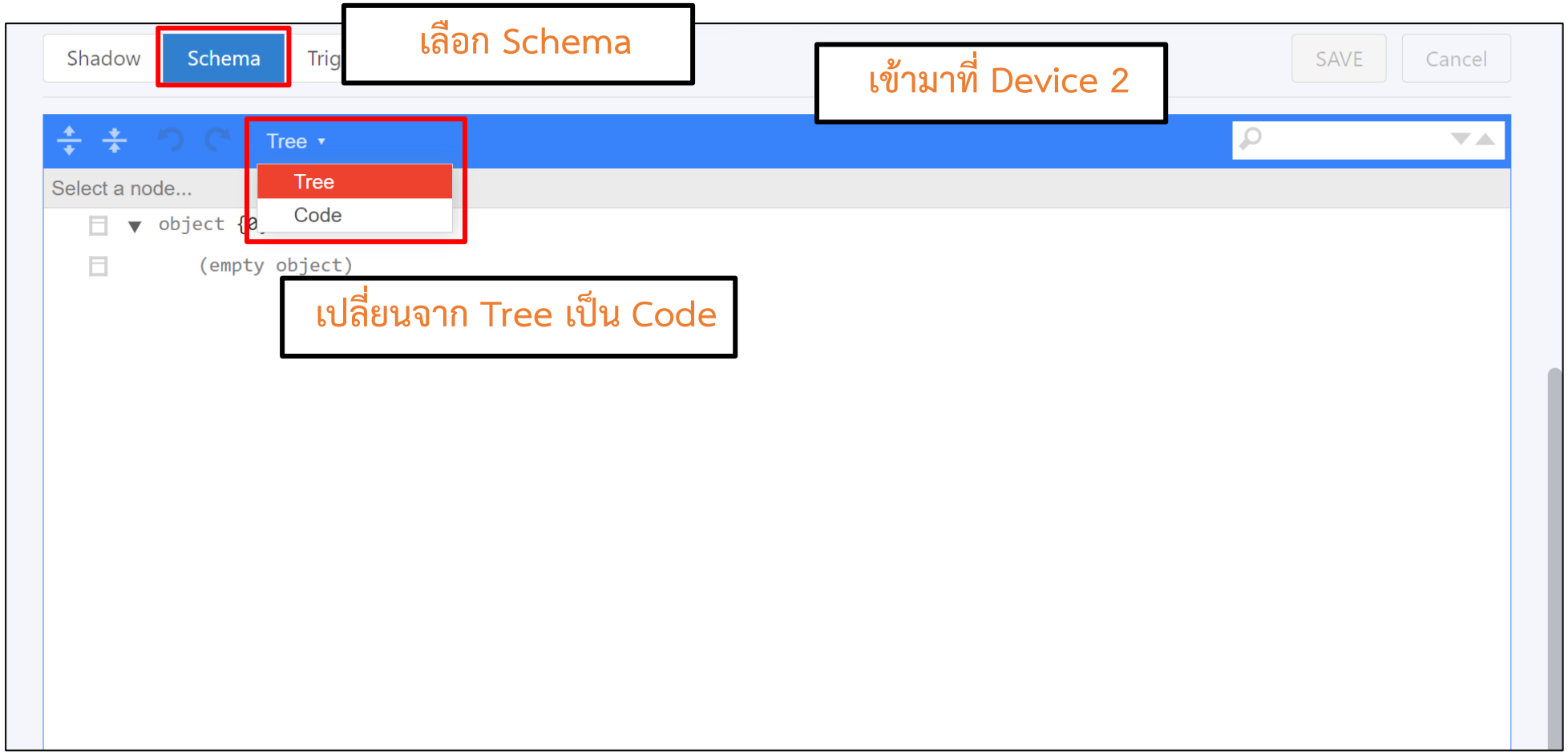
NETPIE MQTT Broker

Device Schema

เราสามารถใช้โค้ด example7 ของ Day3 ในการส่งข้อมูลต่างๆของ ESP8266 ไปยัง Device บน NETPIE

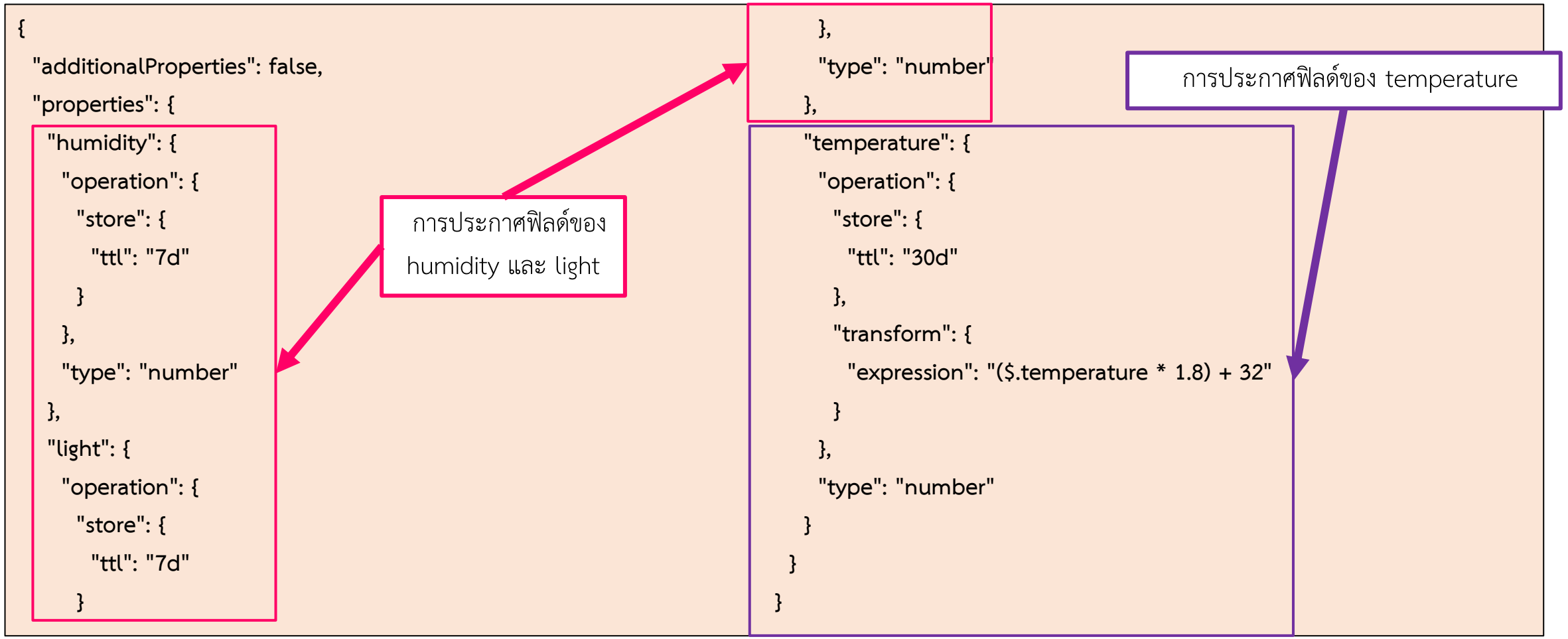
ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ ESP32



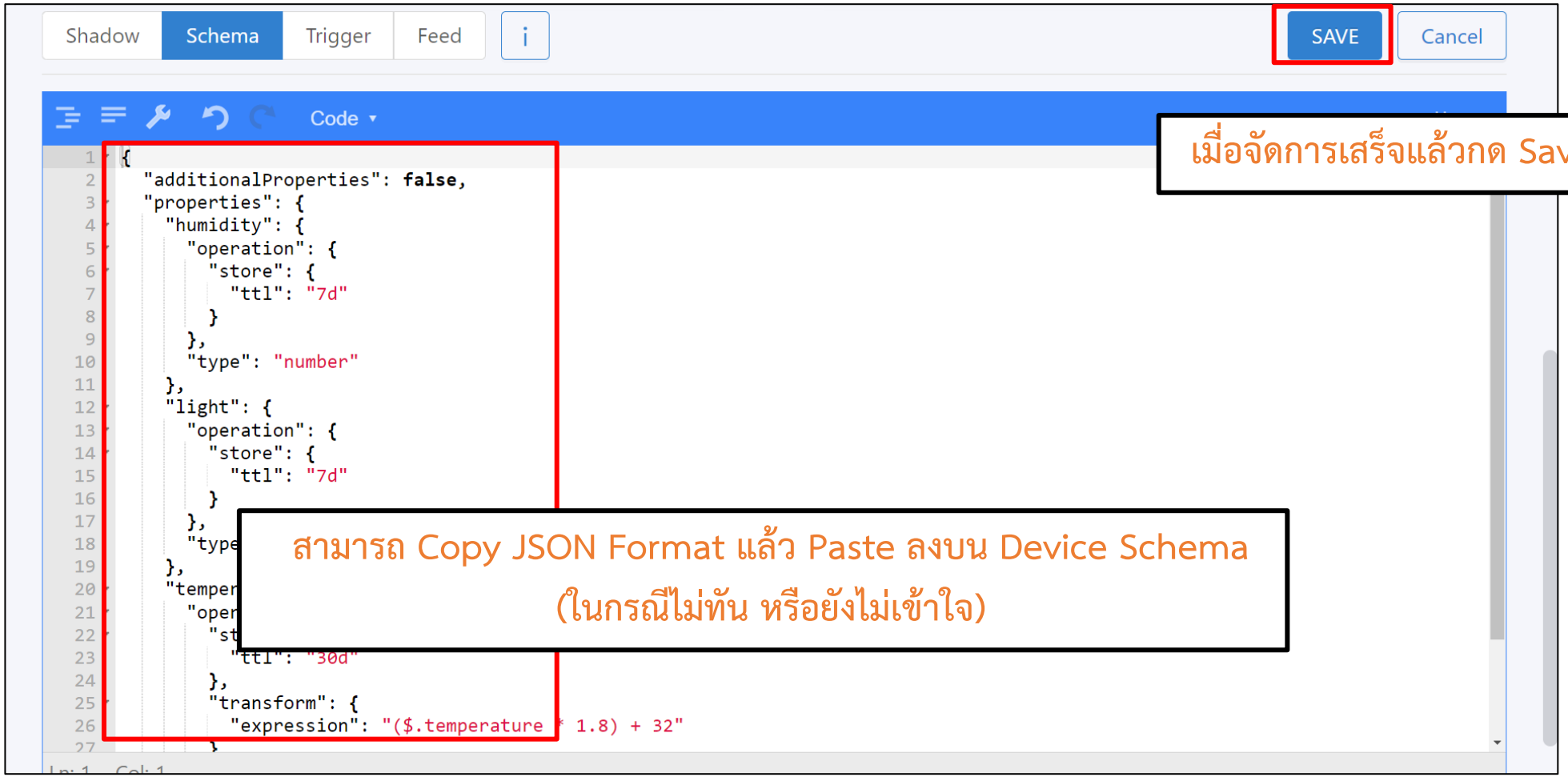
ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ ESP32



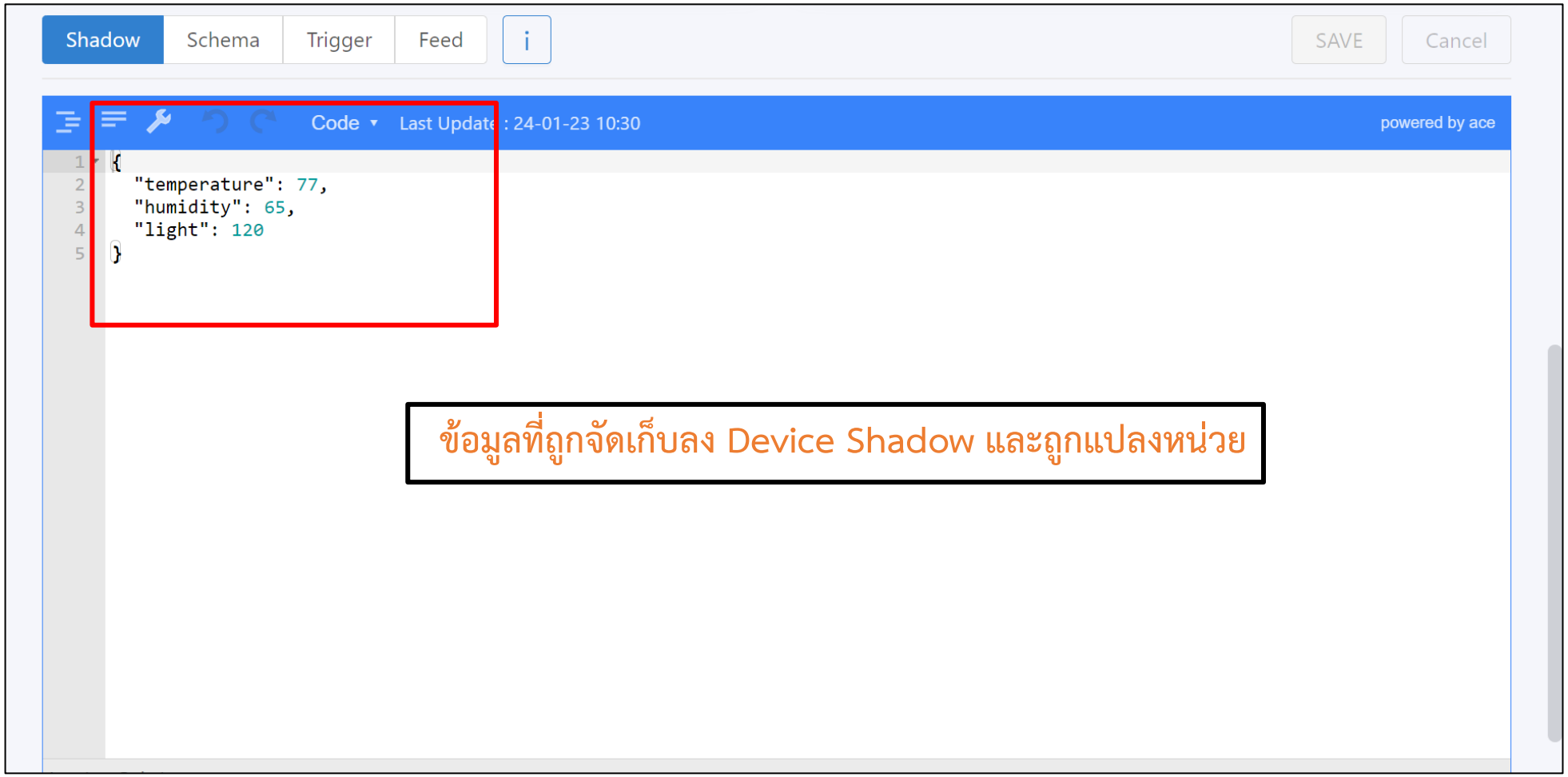
ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ ESP32



ใบงานที่ 4.4 ขั้นตอนการทดลอง

การทดลองที่ 2 ทดสอบการตั้งค่า Schema และส่งข้อมูลโดยใช้ ESP32

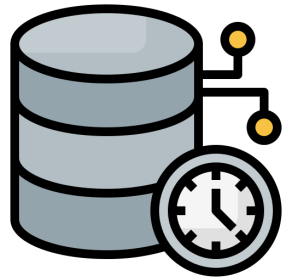


ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

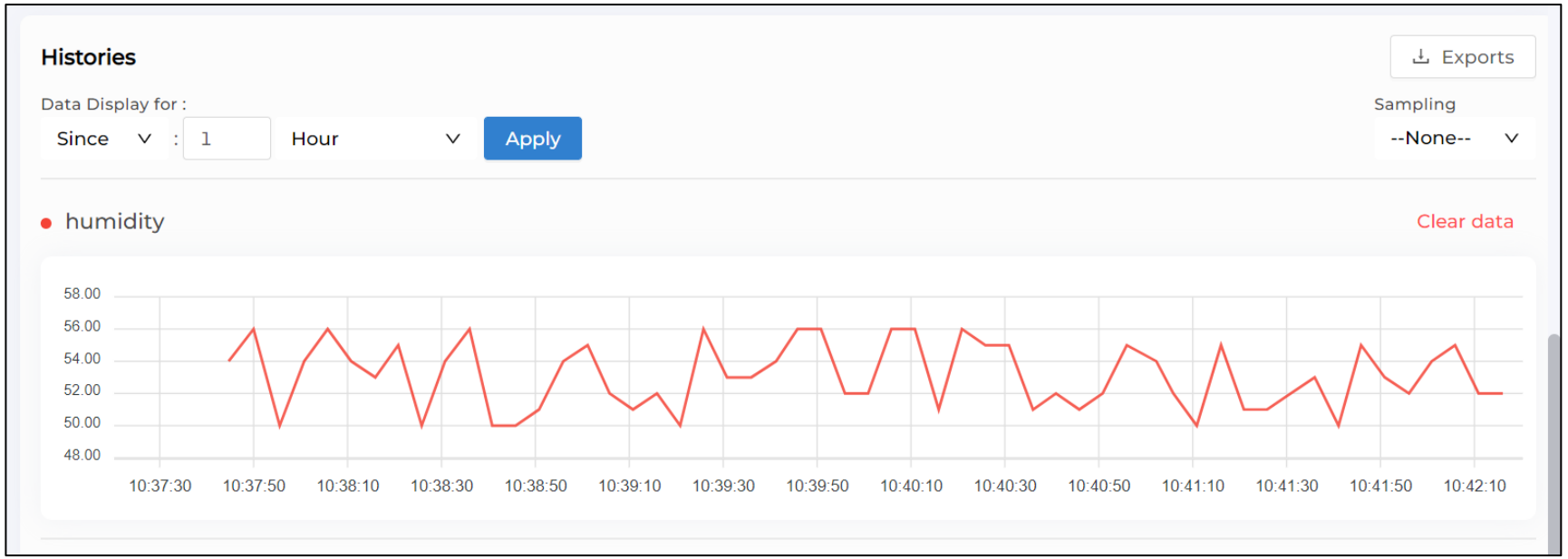
ฐานข้อมูลเวลา (Device Feed)

ข้อมูลบน Device Feed จะถูกจัดเก็บให้อัตโนมัติหลังจากตั้งค่าบน Device Schema สำเร็จ

Device Feed คือ ฐานข้อมูลเวลาบน NETPIE โดยจัดเก็บในรูปแบบของ Timeseries Data ซึ่งมีแถบเครื่องมือในการจัดการและดูข้อมูลเบื้องต้นของแต่ละ Device อีกทั้งจะแสดงในรูปแบบของกราฟเส้น แยกตามฟิลด์ (หรือก็คือ Properties ที่กำหนดอยู่ใน Device Schema) และยังสามารถดาวน์โหลดข้อมูลออกมาเป็นไฟล์ .csv



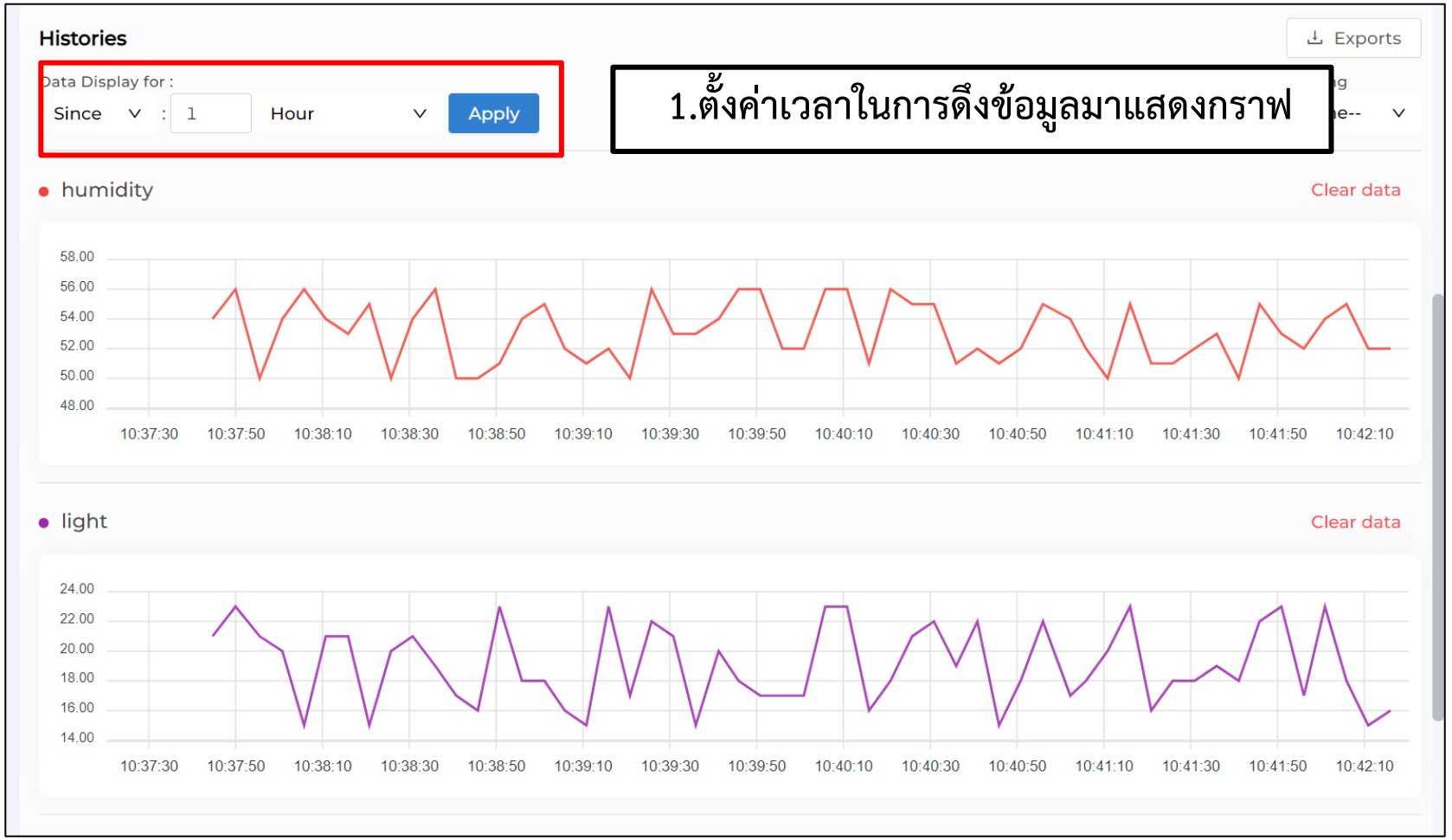
Device Feed



ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การใช้งาน Device Feed

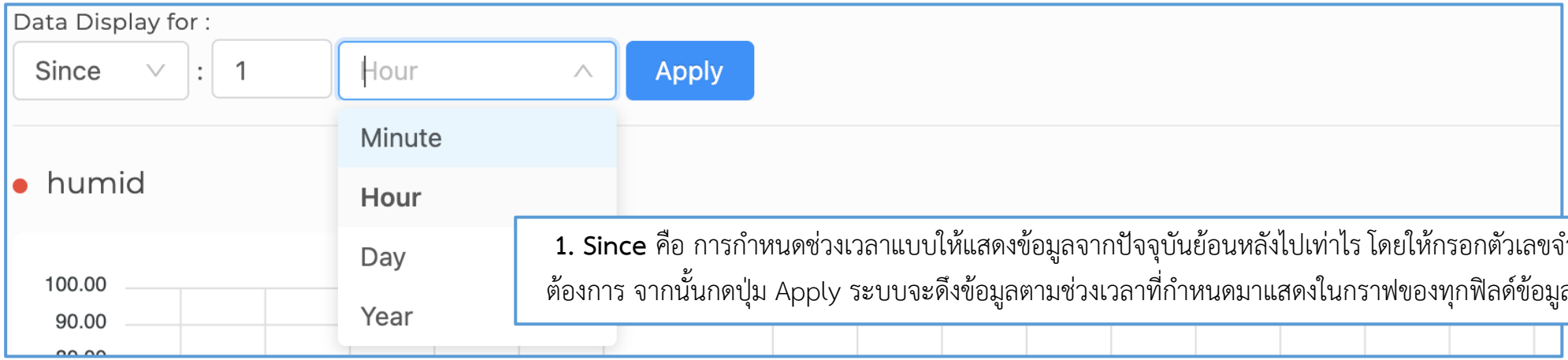
เครื่องมือบน Device Feed มีทั้งหมด 4 เครื่องมือ



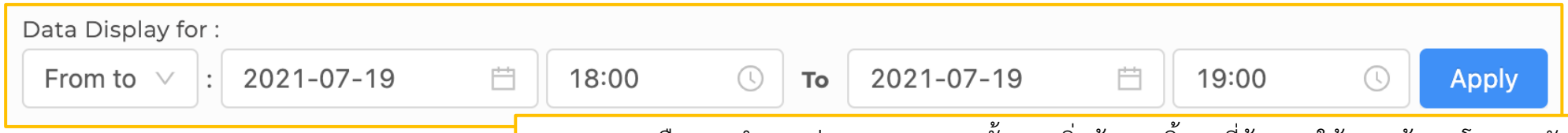
ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การใช้งาน Device Feed

1. การตั้งค่าเวลาในการดึงข้อมูลมาแสดงกราฟ มีอยู่ 2 แบบ



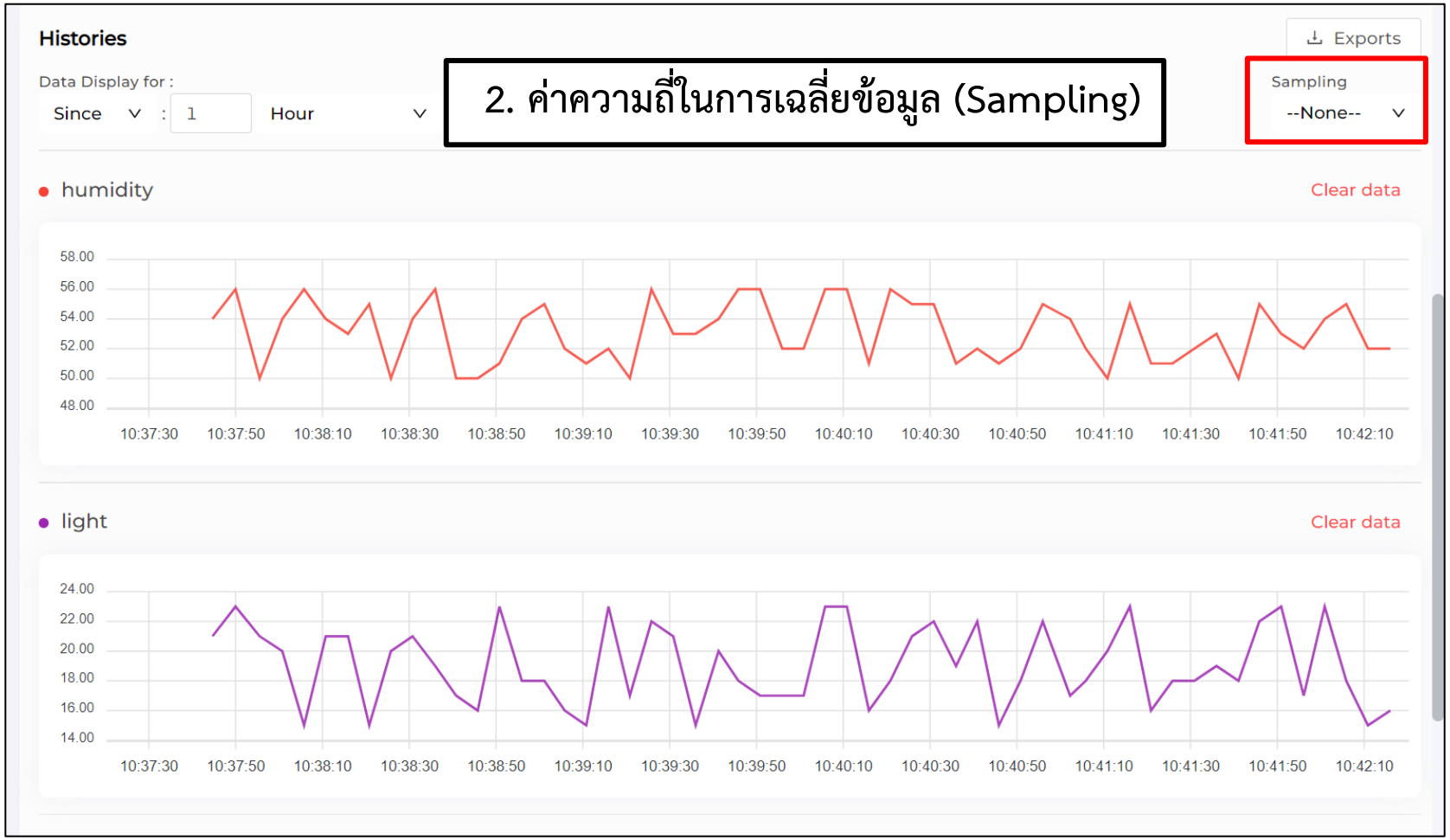
1. Since คือ การกำหนดช่วงเวลาแบบให้แสดงข้อมูลจากปัจจุบันย้อนหลังไปเท่าไร โดยให้กรอกตัวเลขจำนวนเต็มและเลือกหน่วยที่ต้องการ จากนั้นกดปุ่ม Apply ระบบจะดึงข้อมูลตามช่วงเวลาที่กำหนดมาแสดงในกราฟของทุกฟิลด์ข้อมูล การตั้งค่าเป็นดังรูปต่อไปนี้



2. From to คือ การกำหนดช่วงเวลาแบบระบุทั้งเวลาเริ่มต้นและสิ้นสุดที่ต้องการให้แสดงข้อมูล โดยระบุวันเวลาเริ่มต้นและสิ้นสุดตามที่ต้องการ จากนั้นกดปุ่ม Apply ระบบจะดึงข้อมูลตามช่วงเวลาที่กำหนดมาแสดงในกราฟของทุกฟิลด์ข้อมูล การตั้งค่าเป็นดังรูปต่อไปนี้

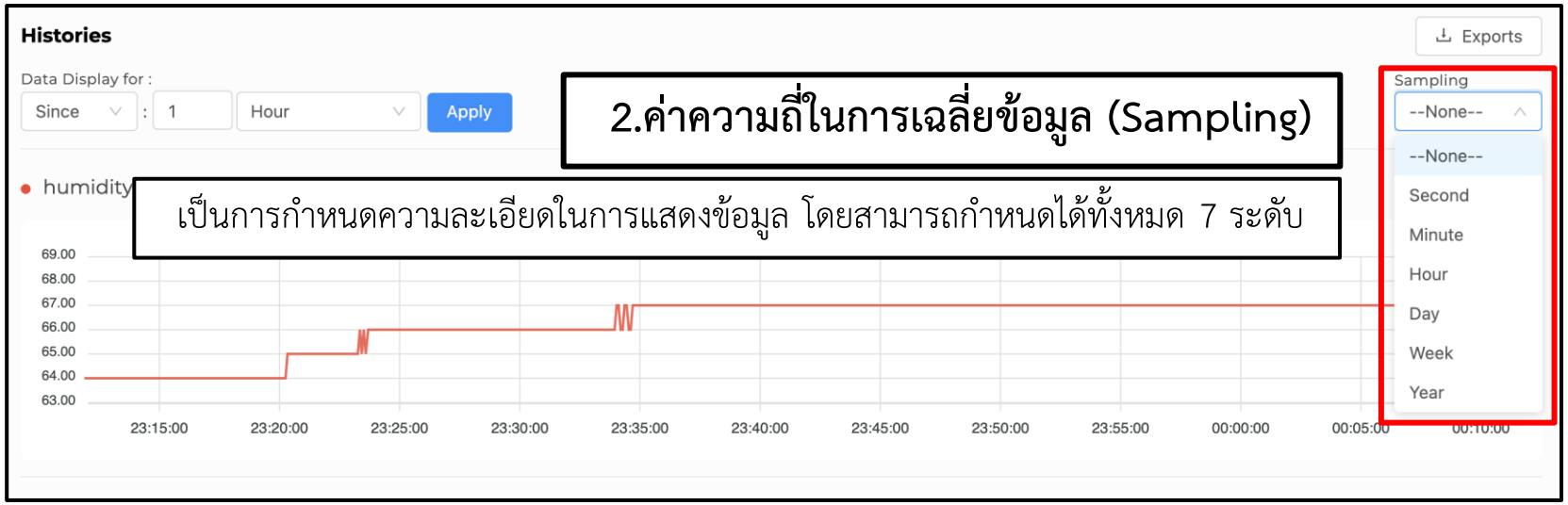
ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การใช้งาน Device Feed



ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การใช้งาน Device Feed

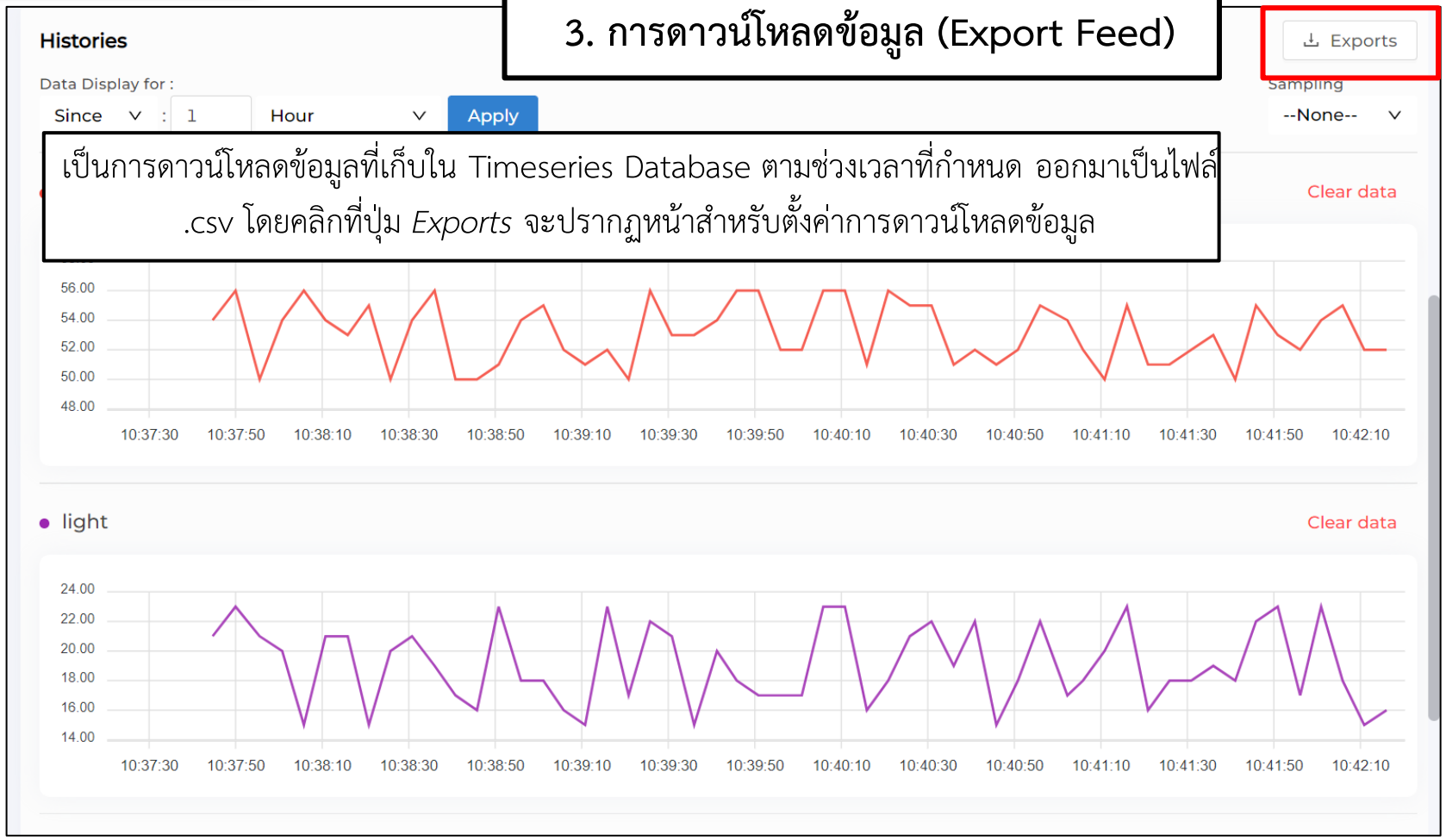


- **None** คือ เป็นการแสดงข้อมูลที่มีความละเอียดสูงสุด โดยข้อมูลที่นำมาแสดงจะเป็นข้อมูลดิบ (Raw Data) ที่ ณ ช่วงเวลานั้น
- **Second** คือ เป็นการนำข้อมูลจริงทั้งหมดในช่วงเวลาที่กำหนดมาประมวลผล โดยทุก 1 วินาทีที่มีข้อมูลมากกว่า 1 จุด จะถูกนำมาหาค่าเฉลี่ยเพื่อให้ได้ค่าเหลือเพียง 1 จุด/วินาที
- **Minute** คือ เป็นการนำข้อมูลจริงทั้งหมดในช่วงเวลาที่กำหนดมาประมวลผล โดยทุก 1 นาทีที่มีข้อมูลมากกว่า 1 จุด จะถูกนำมาหาค่าเฉลี่ยเพื่อให้ได้ค่าเหลือเพียง 1 จุด/นาที
- **Hour** คือ เป็นการนำข้อมูลจริงทั้งหมดในช่วงเวลาที่กำหนดมาประมวลผล โดยทุก 1 ชั่วโมงที่มีข้อมูลมากกว่า 1 จุด จะถูกนำมาหาค่าเฉลี่ยเพื่อให้ได้ค่าเหลือเพียง 1 จุด/ชั่วโมง
- **Day** คือ เป็นการนำข้อมูลจริงทั้งหมดในช่วงเวลาที่กำหนดมาประมวลผล โดยทุก 1 วันที่มีข้อมูลมากกว่า 1 จุด จะถูกนำมาหาค่าเฉลี่ยเพื่อให้ได้ค่าเหลือเพียง 1 จุด/วัน
- **Week** คือ เป็นการนำข้อมูลจริงทั้งหมดในช่วงเวลาที่กำหนดมาประมวลผล โดยทุก 1 สัปดาห์ที่มีข้อมูลมากกว่า 1 จุด จะถูกนำมาหาค่าเฉลี่ยเพื่อให้ได้ค่าเหลือเพียง 1 จุด/สัปดาห์
- **Year** คือ เป็นการนำข้อมูลจริงทั้งหมดในช่วงเวลาที่กำหนดมาประมวลผล โดยทุก 1 ปีที่มีข้อมูลมากกว่า 1 จุด จะถูกนำมาหาค่าเฉลี่ยเพื่อให้ได้ค่าเหลือเพียง 1 จุด/ปี

ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การใช้งาน Device Feed

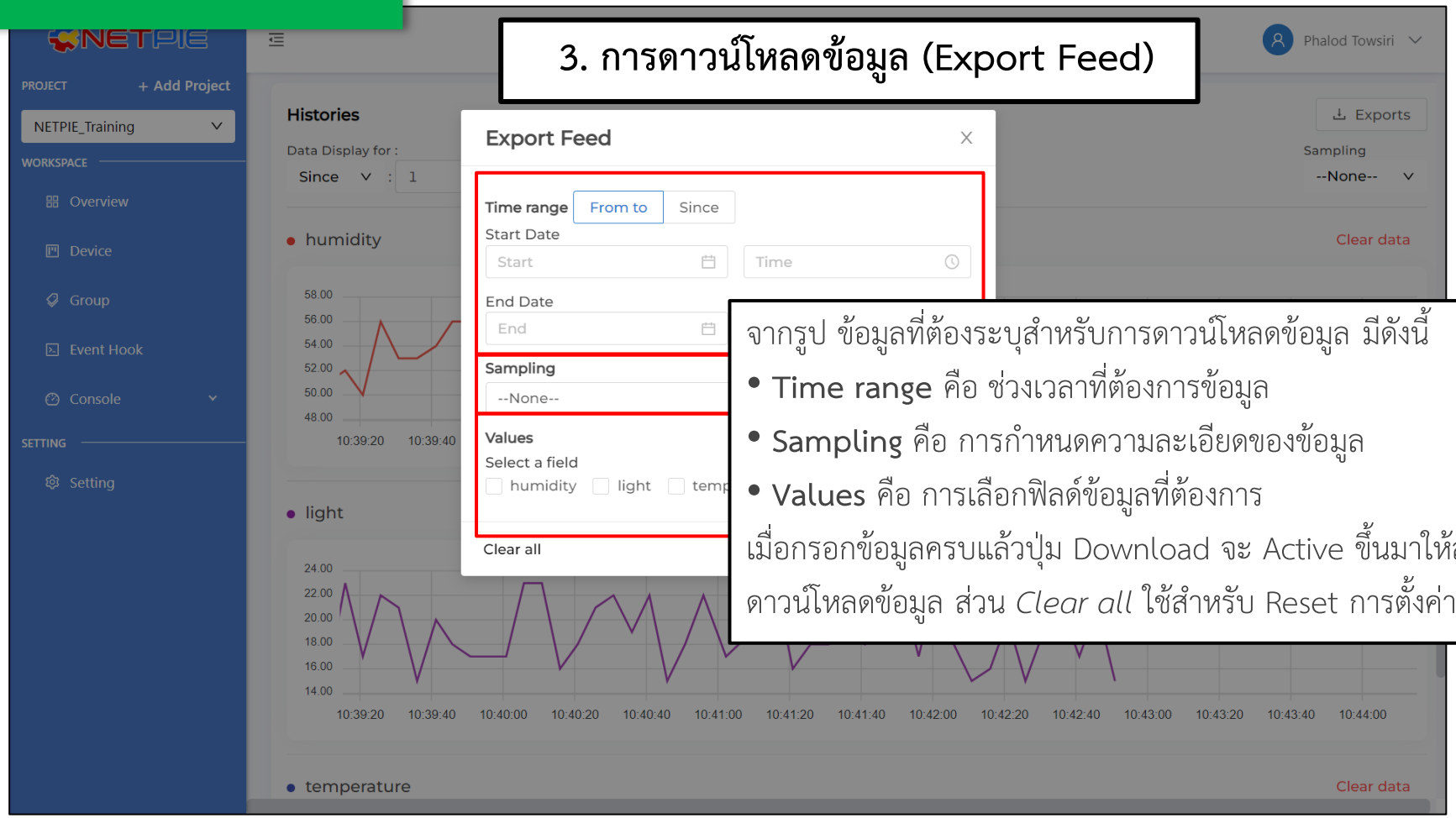
3. การดาวน์โหลดข้อมูล (Export Feed)



เป็นการดาวน์โหลดข้อมูลที่เก็บใน Timeseries Database ตามช่วงเวลาที่กำหนด ออกมาเป็นไฟล์ .csv โดยคลิกที่ปุ่ม *Exports* จะปรากฏหน้าสำหรับตั้งค่าการดาวน์โหลดข้อมูล

ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การใช้งาน Device Feed



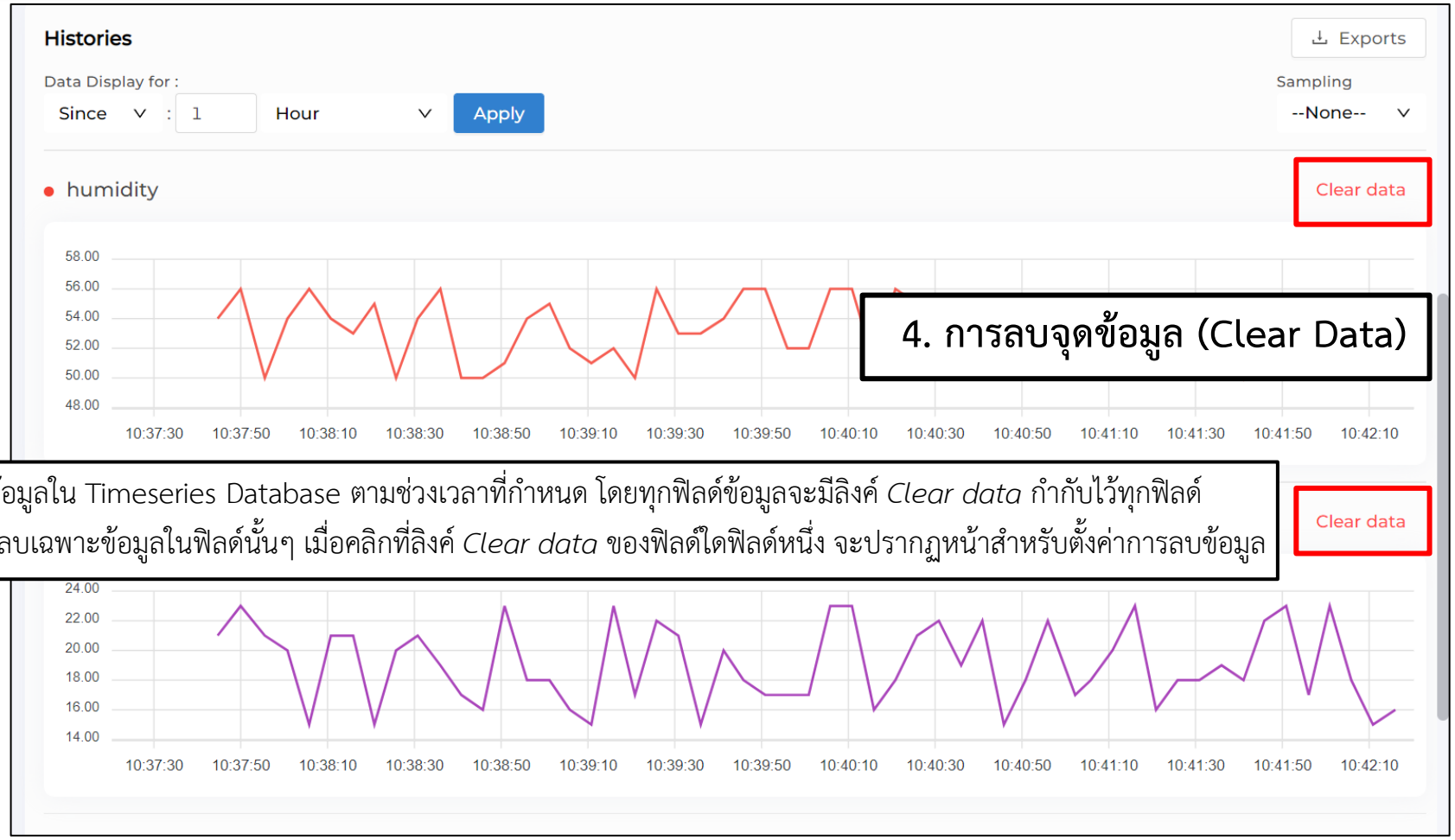
จากรูป ข้อมูลที่ต้องระบุสำหรับการดาวน์โหลดข้อมูล มีดังนี้

- **Time range** คือ ช่วงเวลาที่ต้องการข้อมูล
- **Sampling** คือ การกำหนดความละเอียดของข้อมูล
- **Values** คือ การเลือกฟิลด์ข้อมูลที่ต้องการ

เมื่อกรอกข้อมูลครบแล้วปุ่ม Download จะ Active ขึ้นมาให้สามารถกดได้ ทำการกดเพื่อดาวน์โหลดข้อมูล ส่วน *Clear all* ใช้สำหรับ Reset การตั้งค่าสำหรับดาวน์โหลดข้อมูล

ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

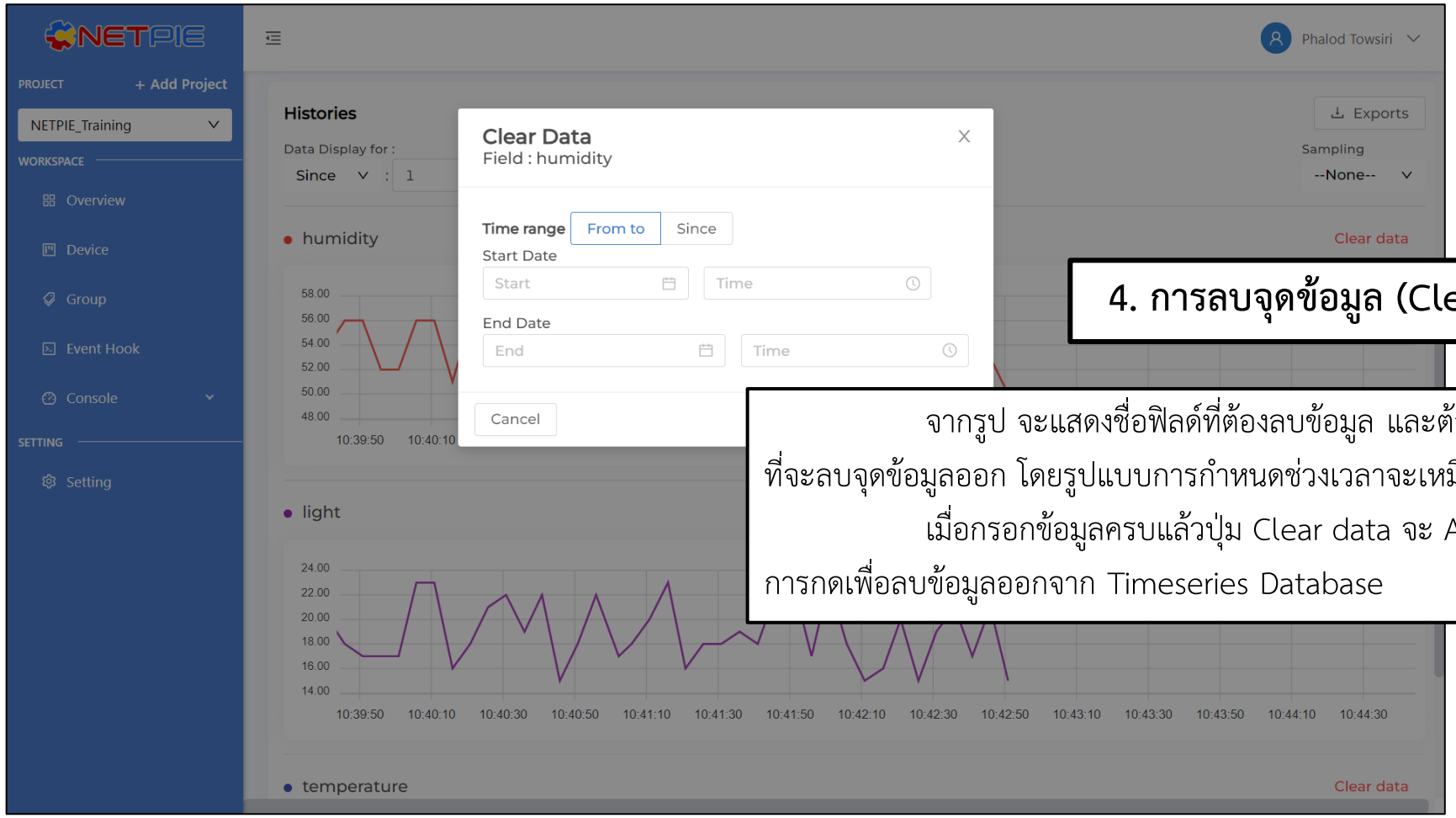
การใช้งาน Device Feed



จะใช้สำหรับลบจุดข้อมูลใน Timeseries Database ตามช่วงเวลาที่กำหนด โดยทุกฟิลด์ข้อมูลจะมีลิงค์ *Clear data* กำกับไว้ทุกฟิลด์ คลิกที่ฟิลด์ใดก็จะเป็นการลบเฉพาะข้อมูลในฟิลด์นั้นๆ เมื่อคลิกที่ลิงค์ *Clear data* ของฟิลด์ใดฟิลด์หนึ่ง จะปรากฏหน้าสำหรับตั้งค่าการลบข้อมูล

ใบงานที่ 4.4 ทฤษฎีเบื้องต้น

การใช้งาน Device Feed



4. การลบจุดข้อมูล (Clear Data)

จากรูป จะแสดงชื่อฟิลด์ที่ต้องลบข้อมูล และต้องกำหนดช่วงเวลา (Time range) ที่จะลบจุดข้อมูลออก โดยรูปแบบการกำหนดช่วงเวลาจะเหมือนในข้อ

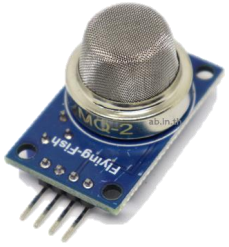
เมื่อกรอกข้อมูลครบแล้วปุ่ม Clear data จะ Active ขึ้นมาให้สามารถกดได้ ทำการกดเพื่อลบข้อมูลออกจาก Timeseries Database

ใบงานที่ 4.4 คำถามท้ายหน่วยการเรียนรู้

คำถามท้ายหน่วยการเรียนรู้ที่ 11

จงเขียนโปรแกรมส่งข้อมูลต่างๆซึ่งอยู่บน ESP32 ไปยัง Device Shadow โดยข้อมูลที่จะถูกส่งขึ้นไปมีดังนี้

1. ค่า Analog (ใช้คำสั่ง AnalogRead ไม่ต้องแปลงค่า) ที่อ่านได้จากตัวต้านทานปรับค่าได้ (voltage)
2. ค่าอากาศจากเซนเซอร์ MQ2 (gas)
3. สถานะ LED1 (led2)
4. สถานะ LED2 (led3)
5. สถานะ LED3 (led4)



และกำหนดค่าบน Device Schema ให้สอดคล้องกับข้อมูลที่ ESP32 ส่งมา โดยมีข้อกำหนดดังนี้

1. ข้อมูลที่ถูกจัดเก็บคือ voltage (number), gas (number), led1 (boolean), led2 (boolean) และ led3 (boolean)
2. มีการแปลงข้อมูลของค่า Analog จากตัวต้านทานปรับค่าได้ ให้เป็นค่าแรงดันไฟฟ้า
3. กำหนดให้ voltage และ gas ถูกจัดเก็บเป็นเวลา 5 วัน และ led1, led2 และ led3 ถูกจัดเก็บเป็นเวลา 2 วัน

โดยส่งส่วนของ Schema ด้วยการ Capture หน้าจอของ Schema แล้วอัพโหลดพร้อมกับ Code ของ Exercise