หน่วยการเรียนรู้ที่ 5 การสร้างแอปพลิเคชันบน NETPIE2020

นายธงชัย ชาบุดศรี

แผนกวิชาเทคโนโลยีสารสนเทศ

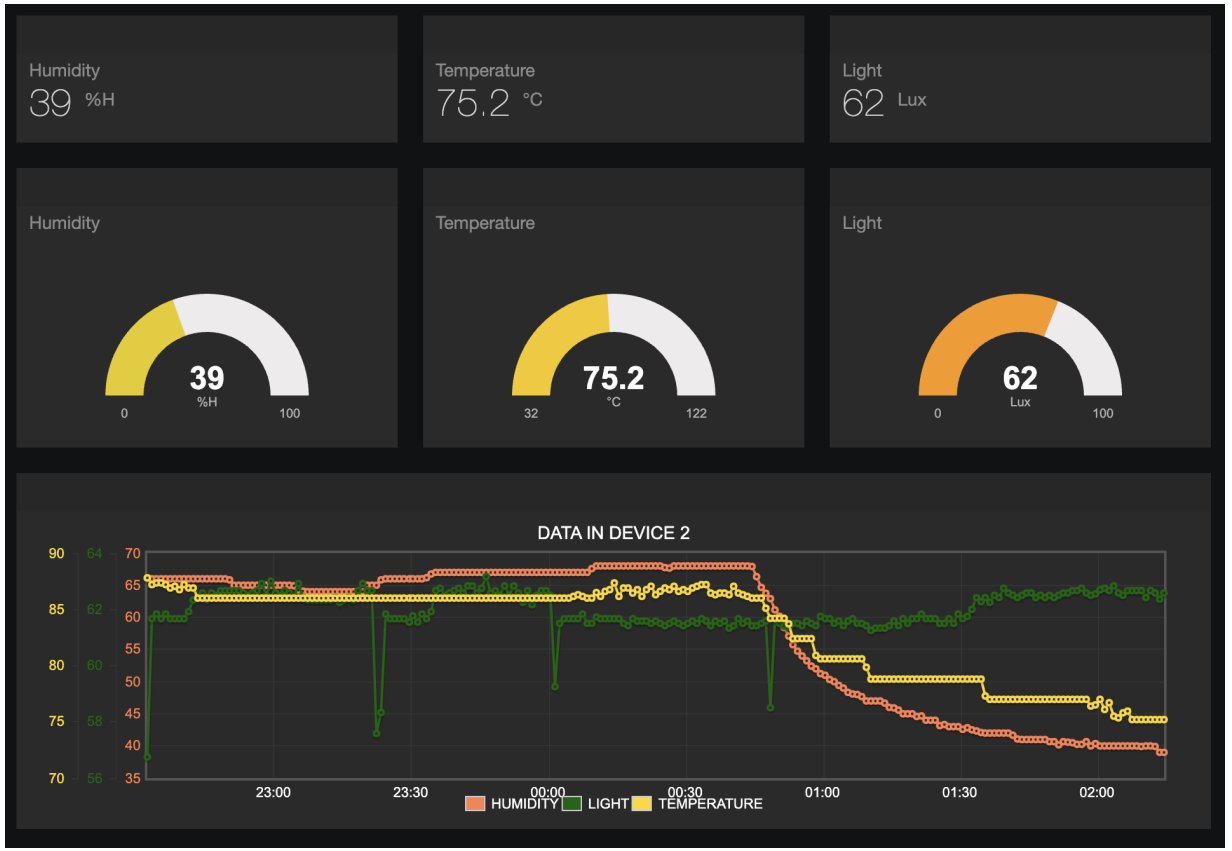
วิทยาลัยเทคนิคชลบุรี

จุดประสงค์การเรียนรู้

1. เรียนรู้การสร้างหน้าจอแสดงผลข้อมูลและควบคุม Dashboard บน NETPIE2020

2. เรียนรู้การสร้างระบบการแจ้งเตือน Line Notify บน NETPIE2020

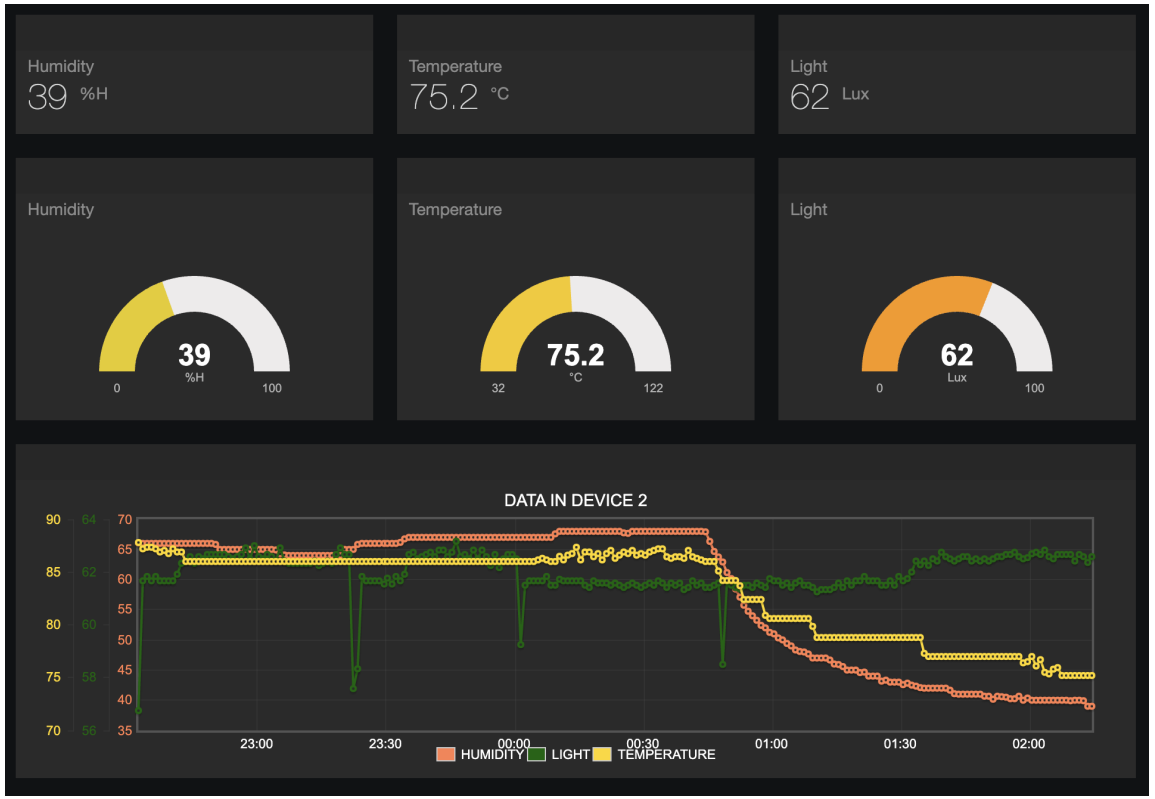
ใบงานที่ 5.1 การแสดงผลข้อมูลจากอุปกรณ์ด้วย Freeboard

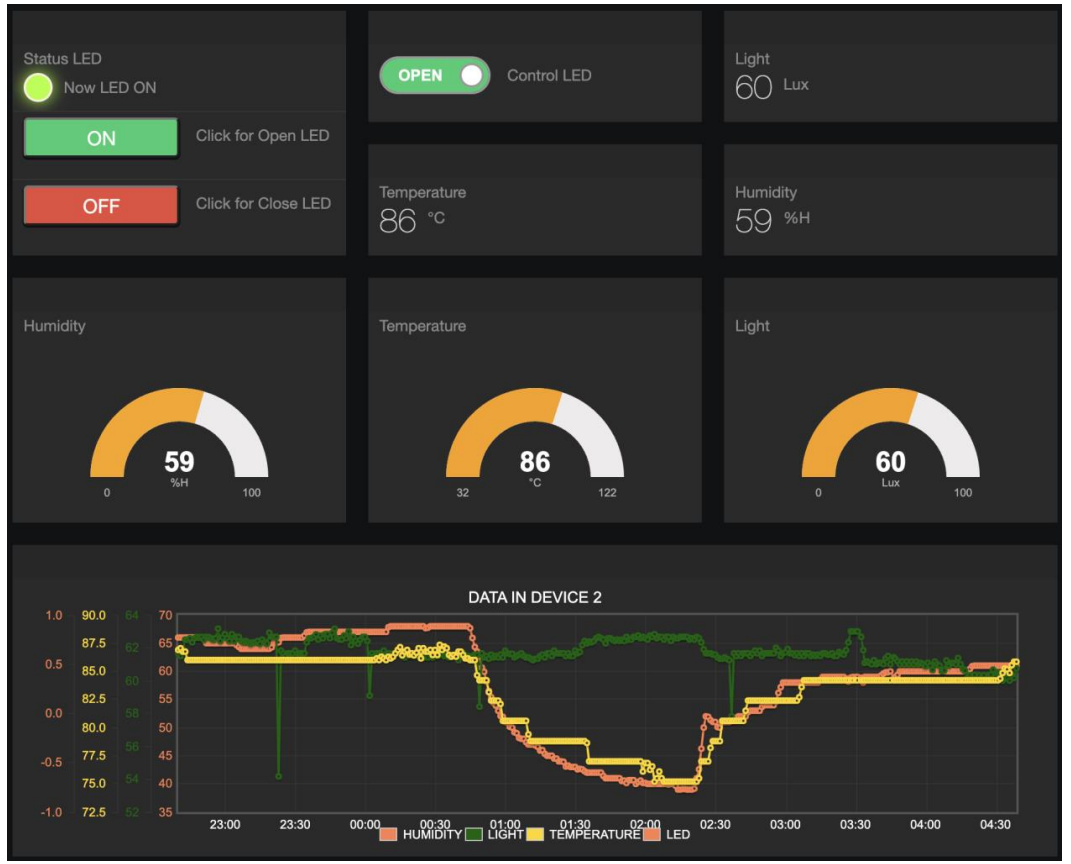


ใบงานที่ 5.1 ทฤษฎีเบื้องต้น

Freeboard คืออะไร

NETPIE Freeboard คือ Dashboard ที่ใช้สำหรับนำข้อมูลที่เก็บอยู่ใน Platform มาแสดงผลในรูปแบบต่างๆ เป็นเหมือนช่องทางให้ผู้ใช้สามารถติดตามหรือควบคุมการทำงานของ Device ของตัวเอง

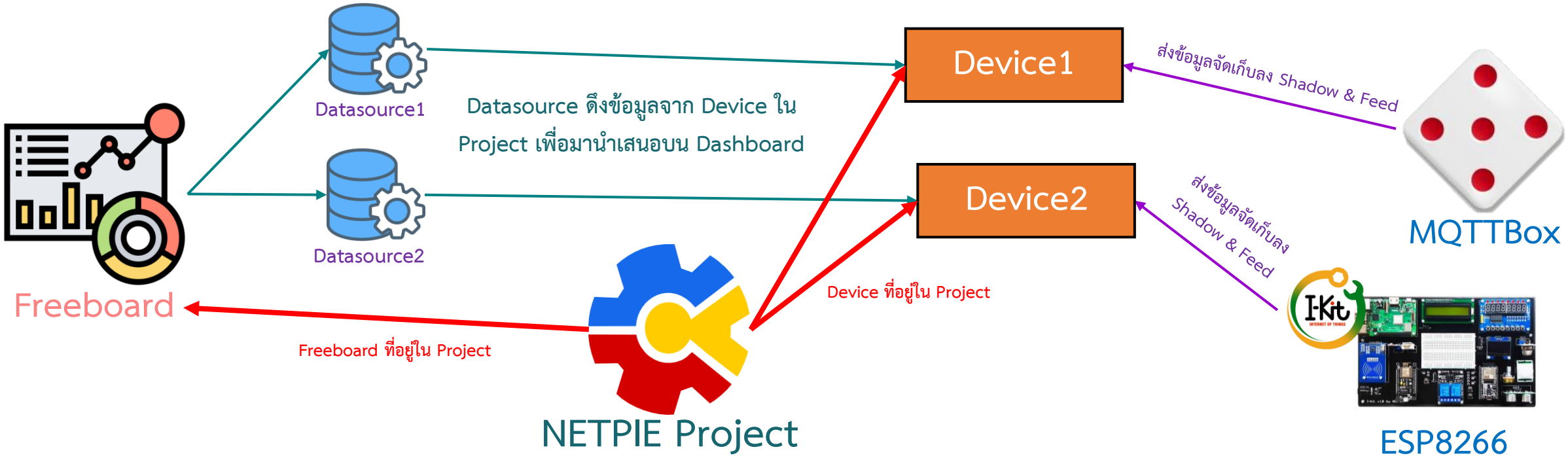




ใบงานที่ 5.1 ทฤษฎีเบื้องต้น

Datasource บน Freeboard

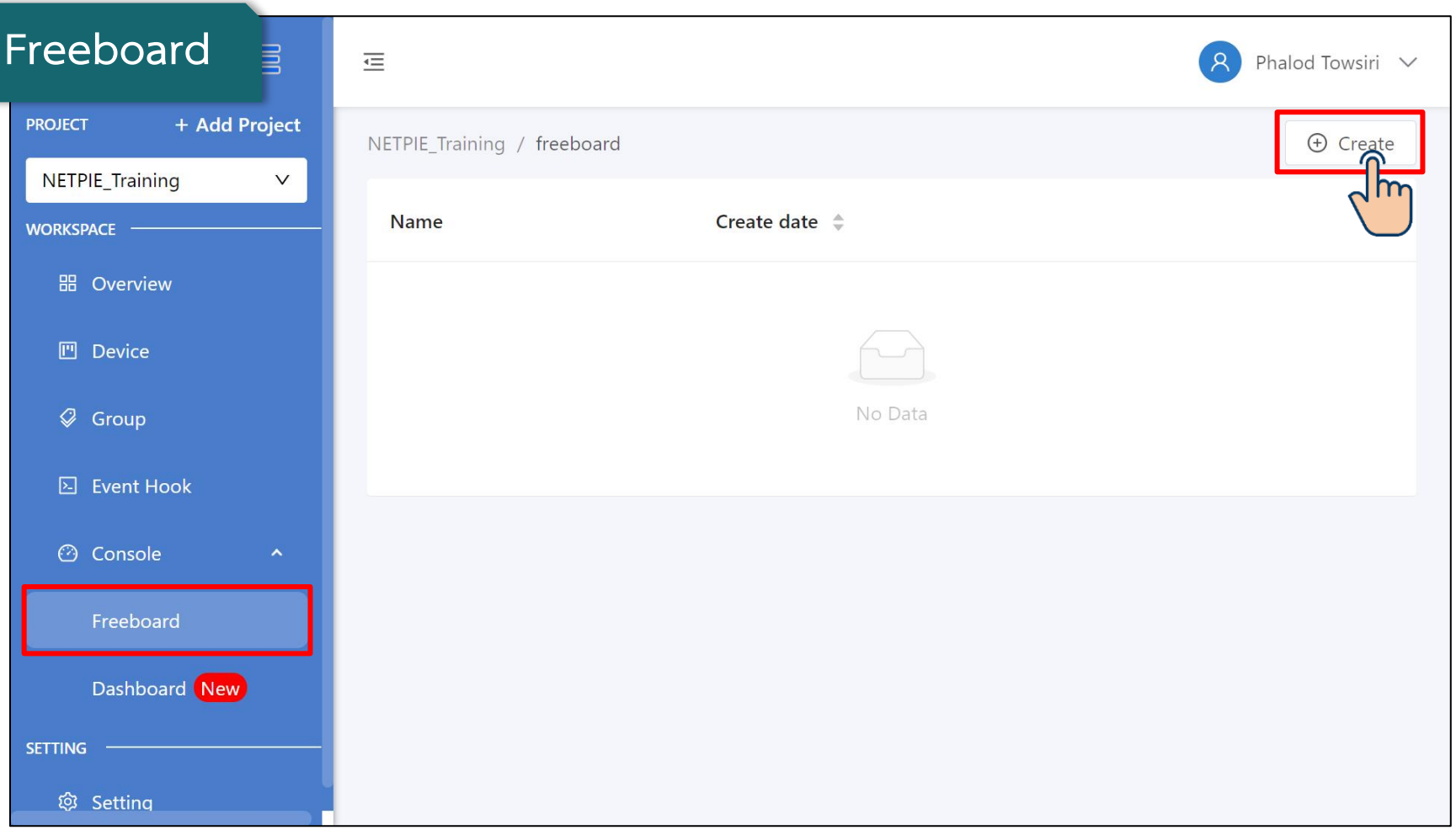
Datasource บน Freeboard คือ ฐานข้อมูลที่ดึงมาจาก Device บน Project เดียวกันกับที่ Freeboard อาศัยอยู่ โดย Datasource นำข้อมูลต่างๆของ Device มารวมอยู่ด้วยกัน เช่น Device Shadow, Device Feed เป็นต้น อีกทั้งบน Freeboard สามารถสร้างจำนวน Datasource ได้มากกว่า 1 Datasource ดังรูปข้างล่าง



ใบงานที่ 5.1 ขั้นตอนการทดลอง

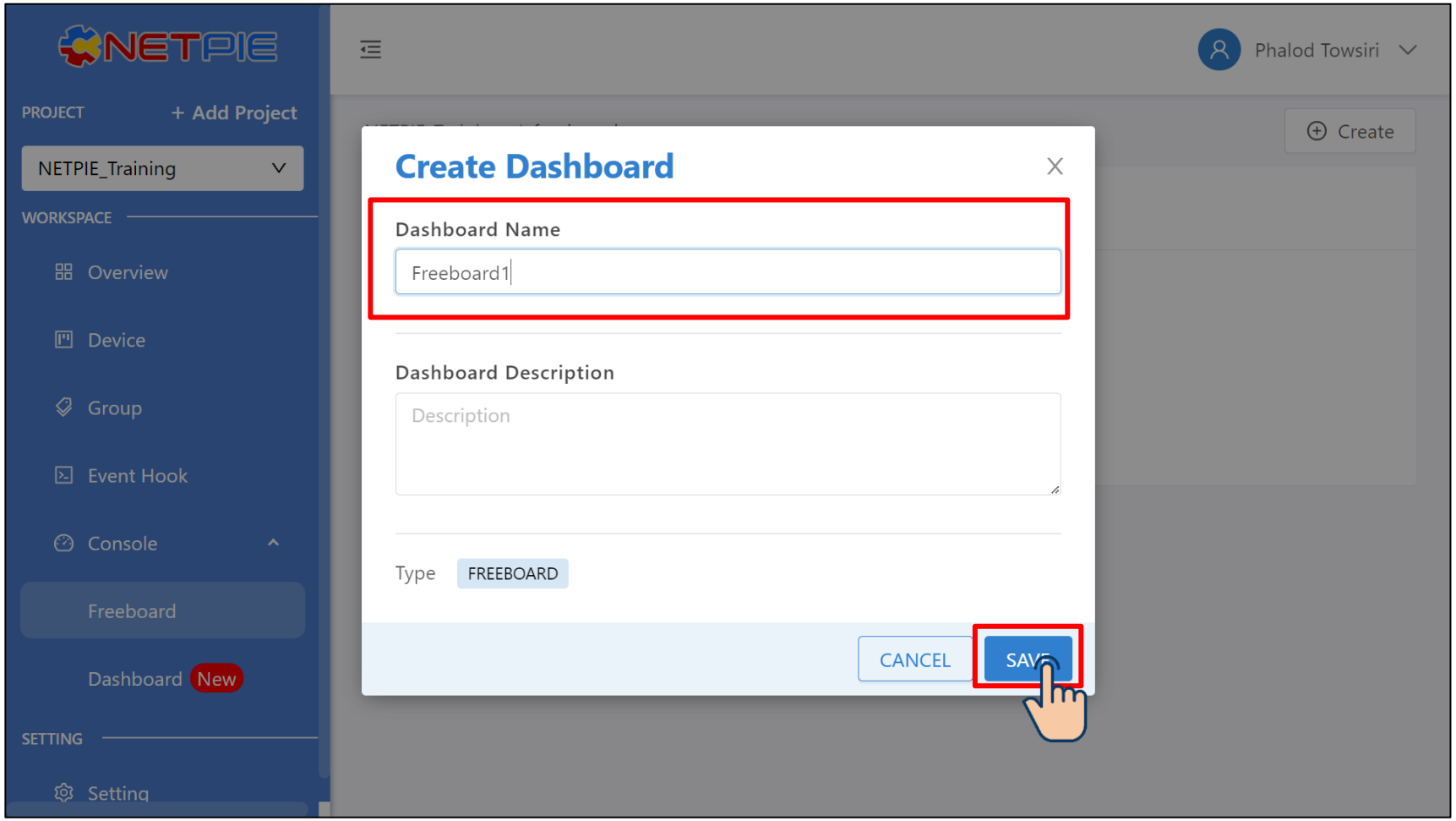
การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource

การสร้าง Freeboard



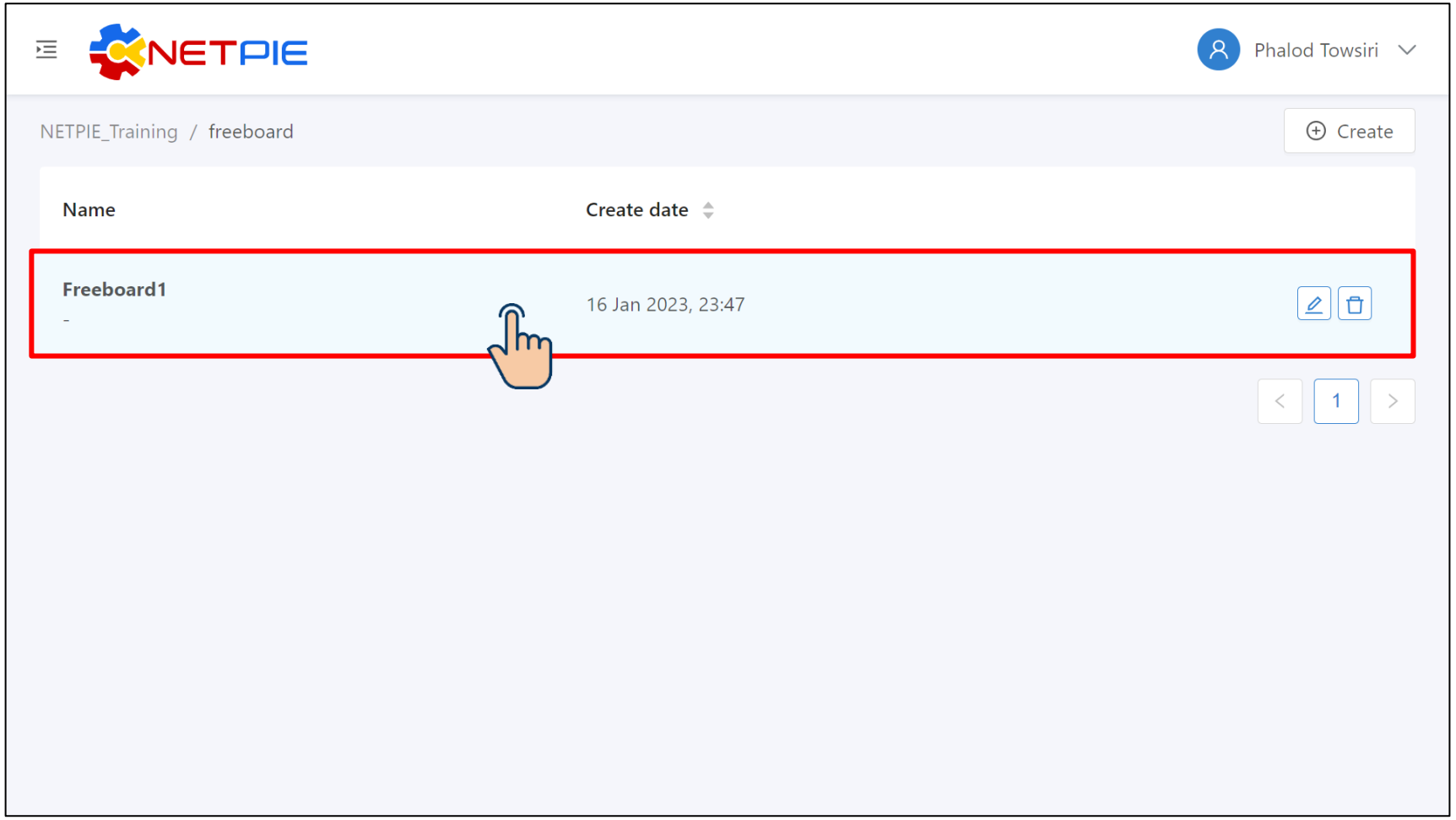
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource



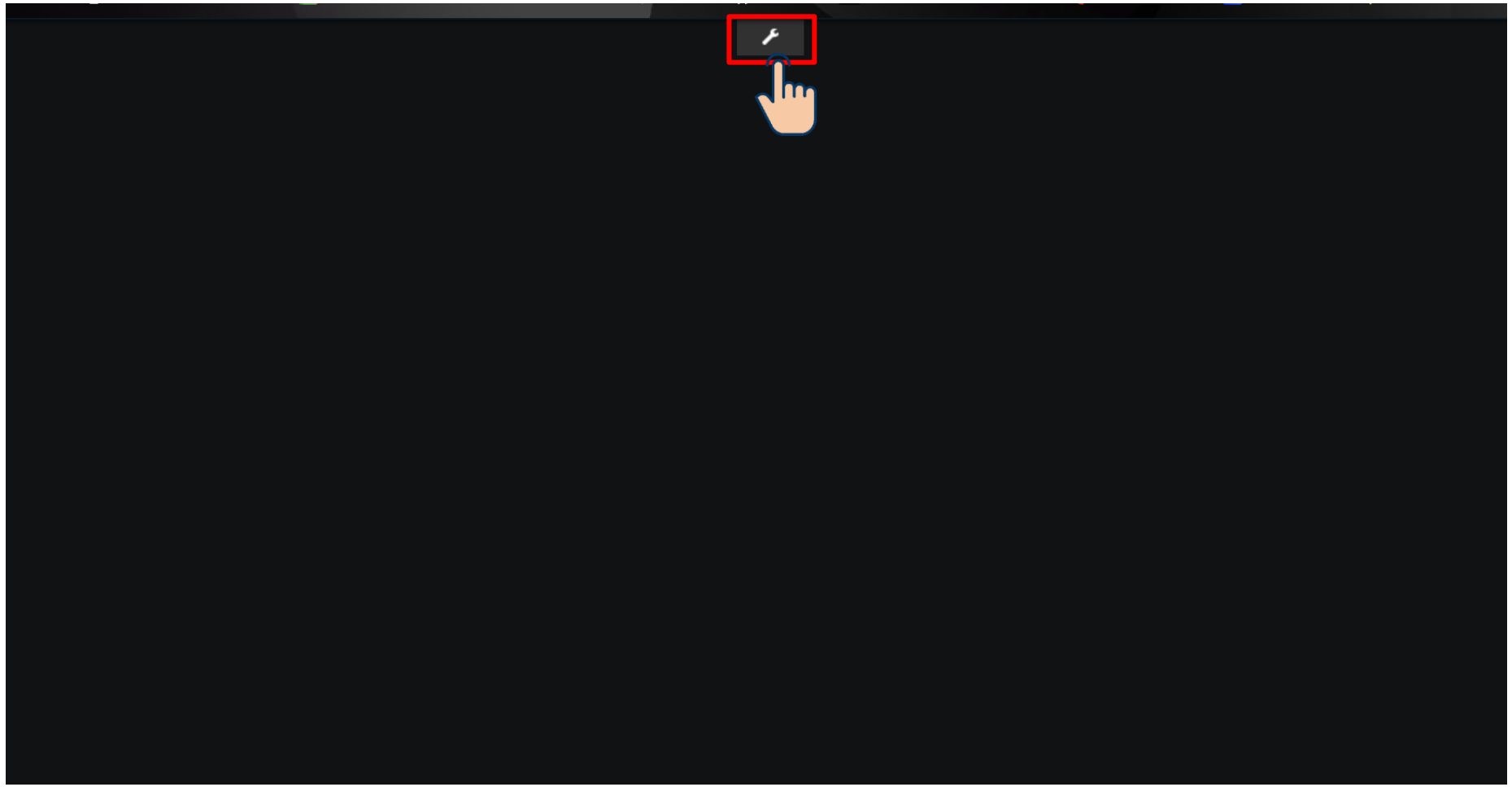
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource



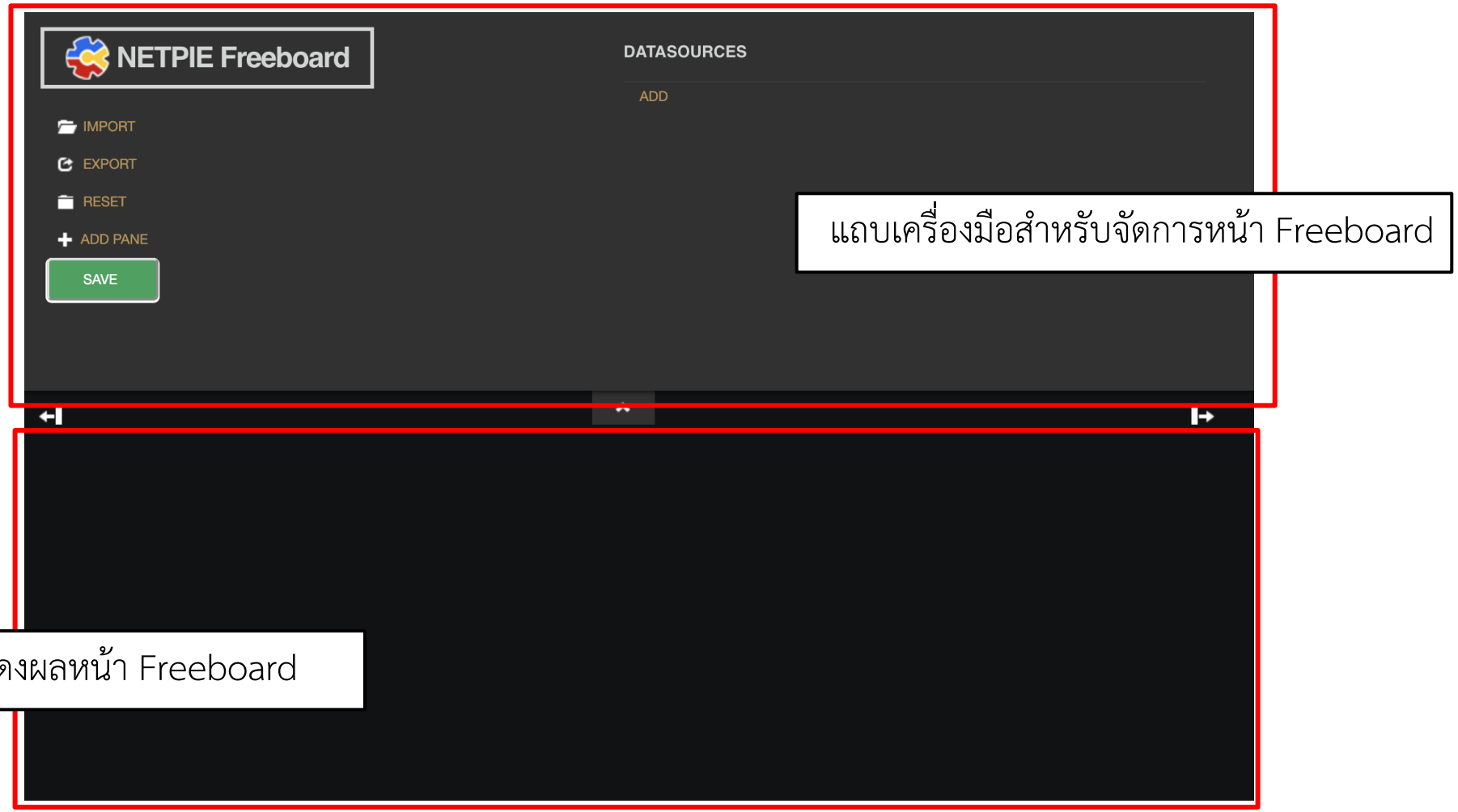
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource



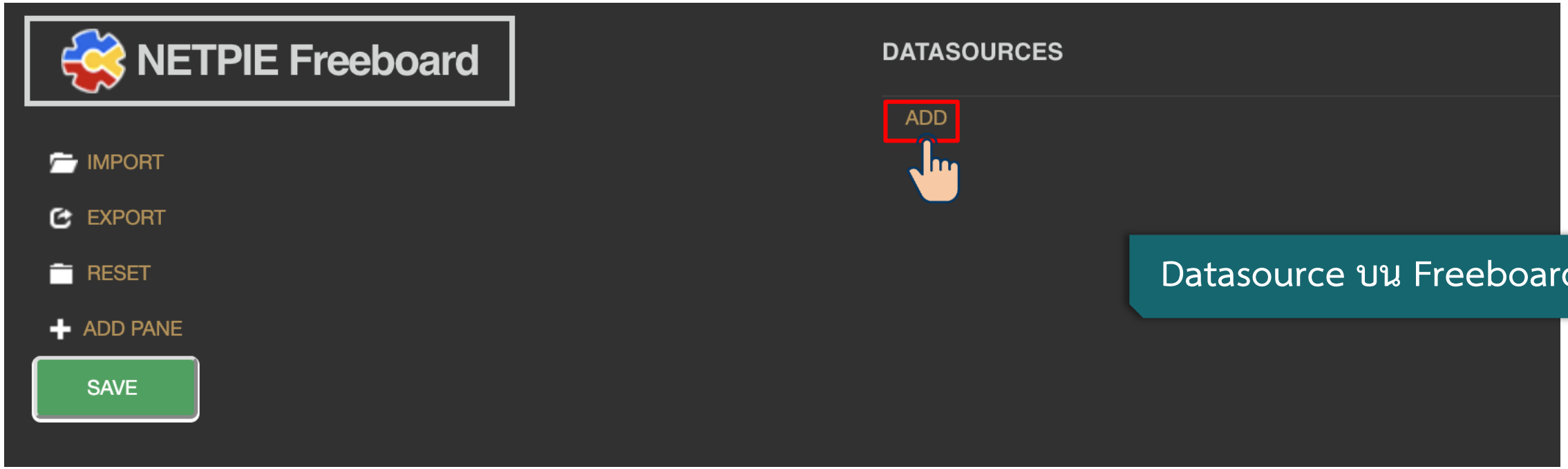
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource



ใบงานที่ 5.1 ขั้นตอนการทดลอง

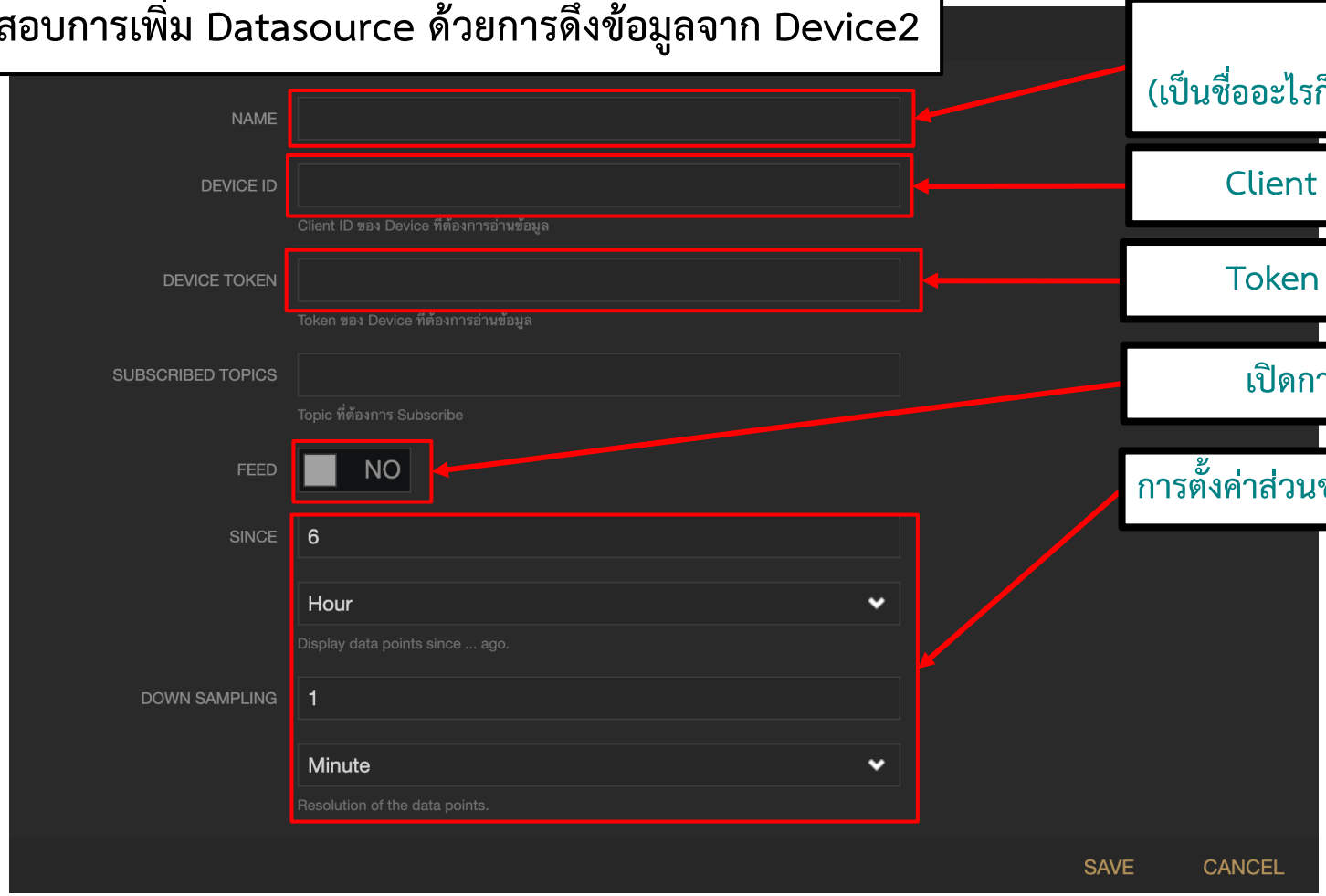
การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource

ทดสอบการเพิ่ม Datasource ด้วยการดึงข้อมูลจาก Device2



ชื่อ Datasource
(เป็นชื่ออะไรก็ได้ แต่ไม่ควรเว้นวรรค ถ้าจำเป็นควรใช้ _ หรือ – แทน)

Client ID ของ Device

Token ของ Device

เปิดการใช้งาน FEED

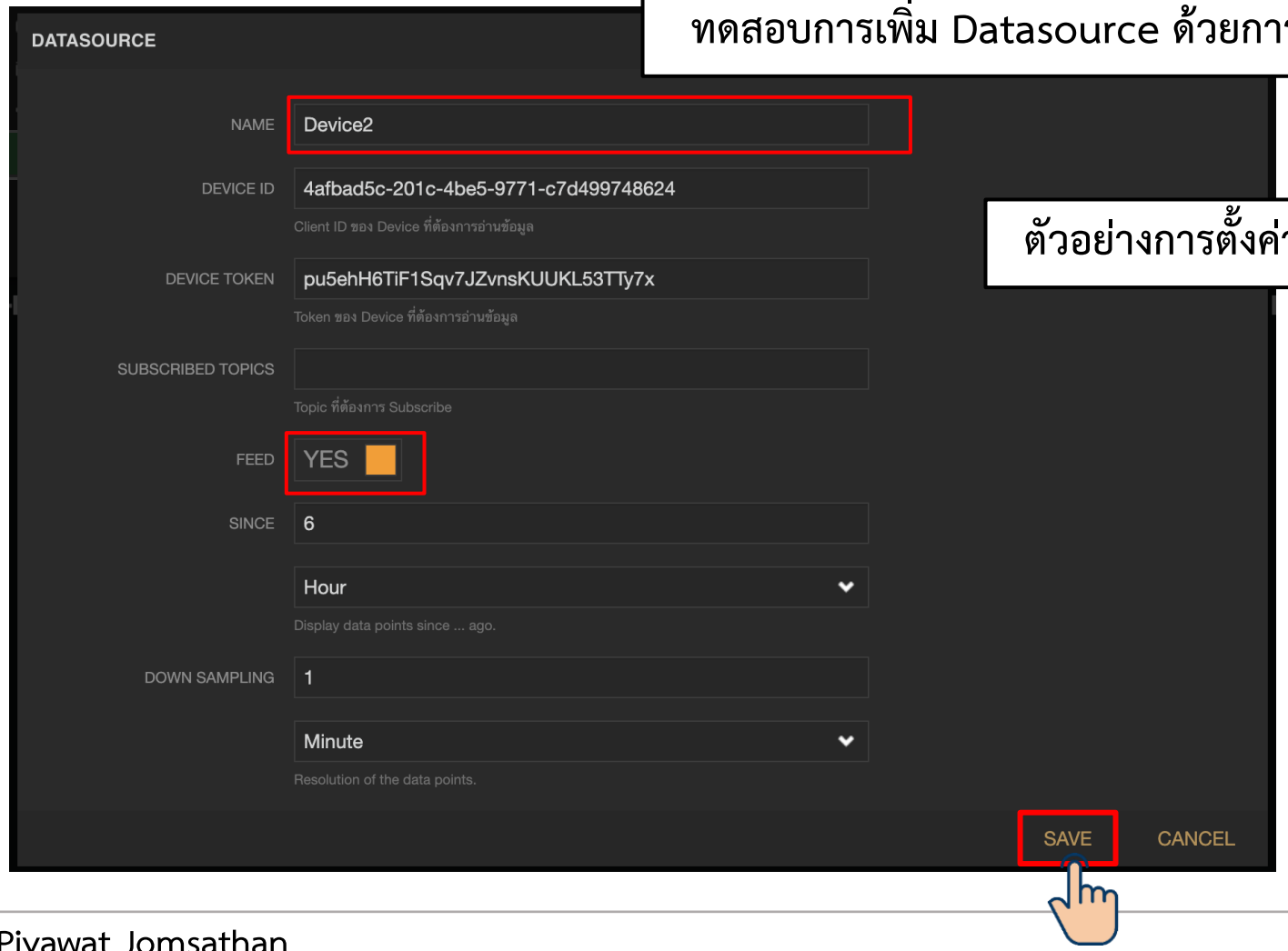
การตั้งค่าส่วนของการแสดงผล Feed

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource

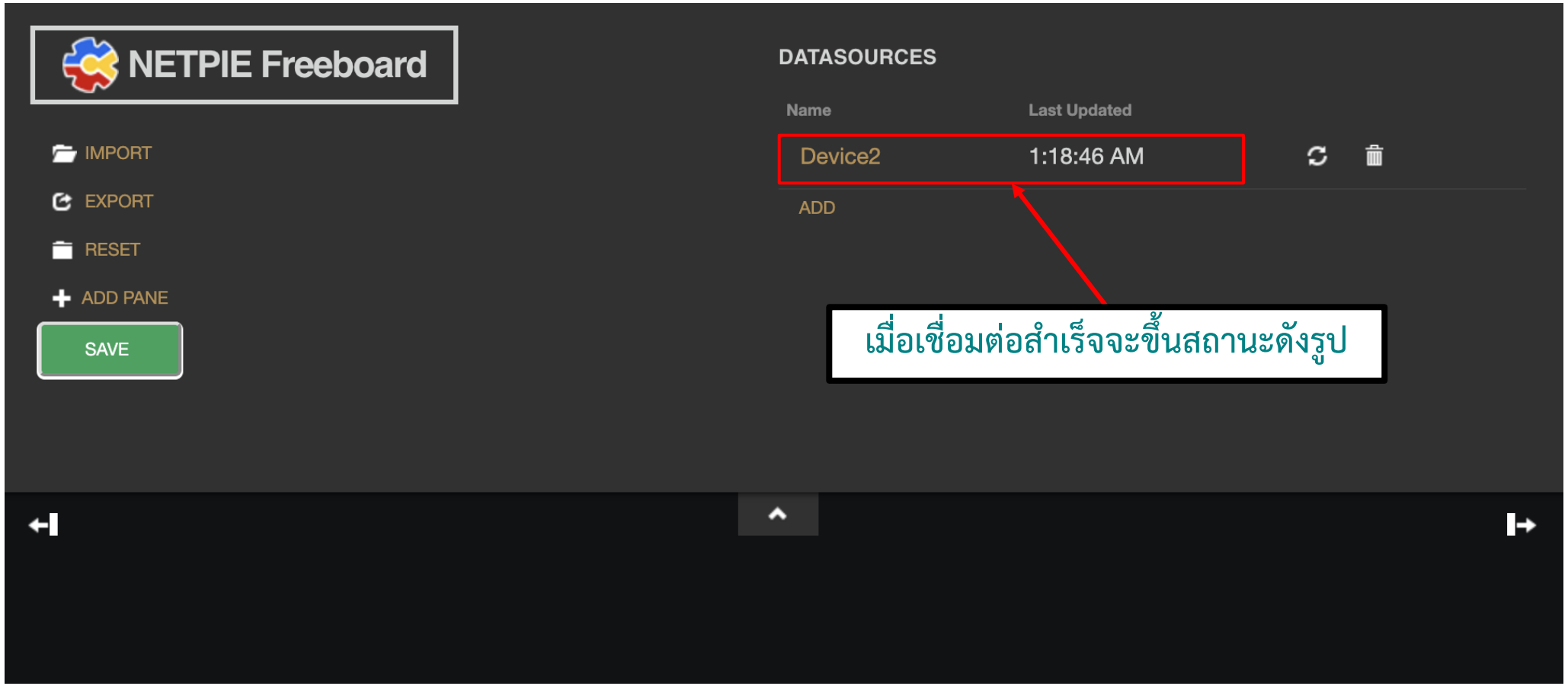
ทดสอบการเพิ่ม Datasource ด้วยการดึงข้อมูลจาก Device2

ตัวอย่างการตั้งค่าต่างๆบน Datasource



ใบงานที่ 5.1 ขั้นตอนการทดลอง

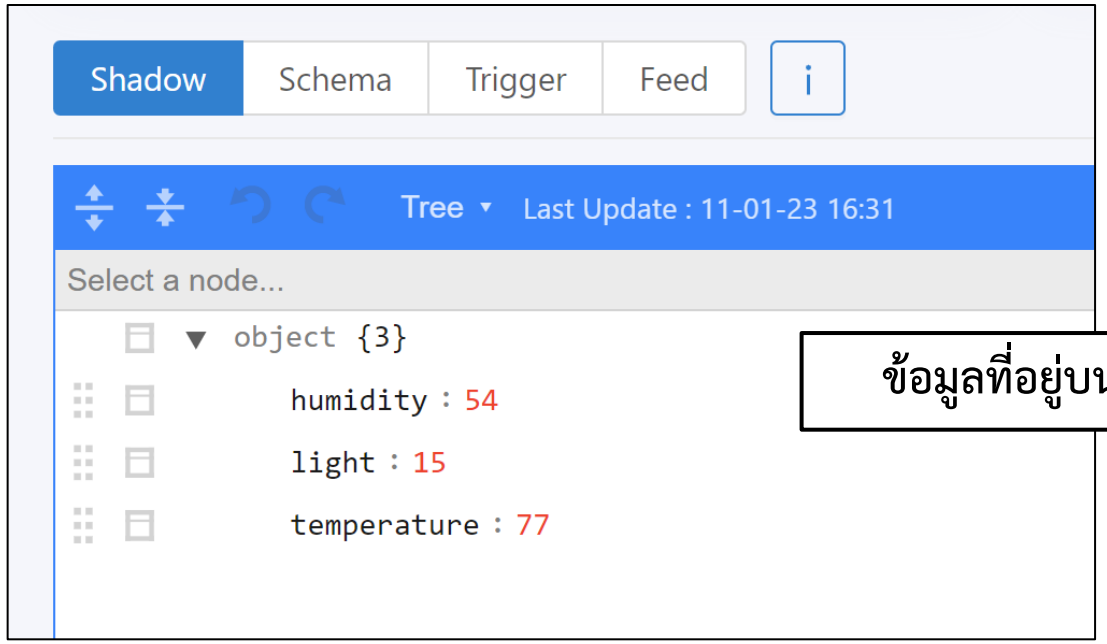
การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource



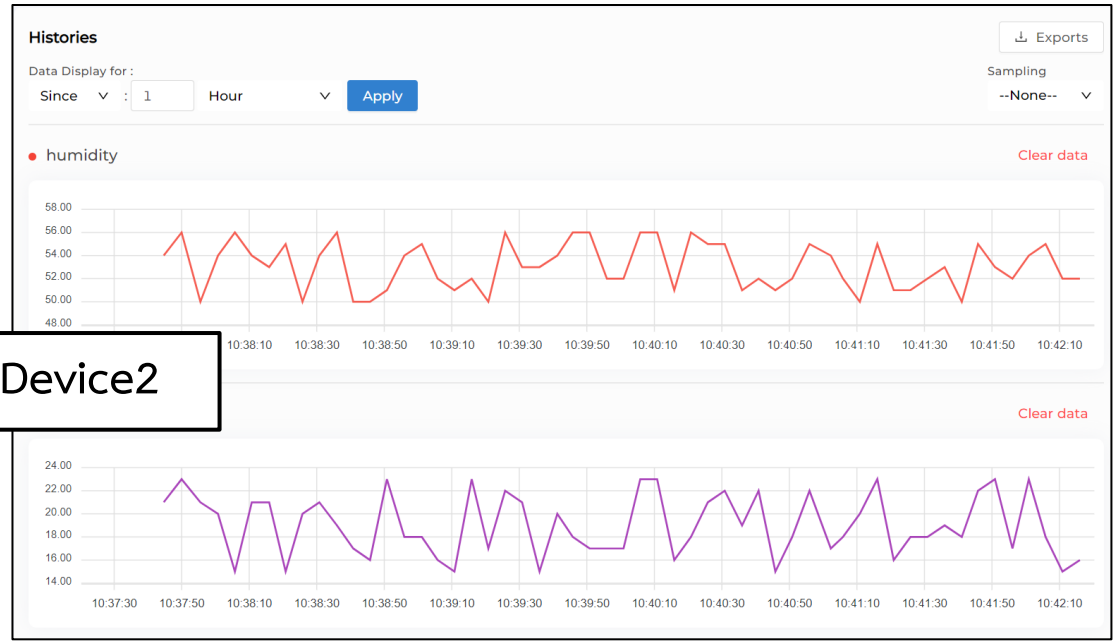
ใบงานที่ 5.1 ทฤษฎีเบื้องต้น

การนำเสนอข้อมูล

เราจะนำข้อมูลต่างๆที่อยู่ใน Device2 ทั้งใน Device Shadow และ Device Feed มานำเสนอบนเครื่องมือนำเสนอต่างๆ ซึ่งเรียกว่า Widget โดยมีอยู่มากบน Freeboard



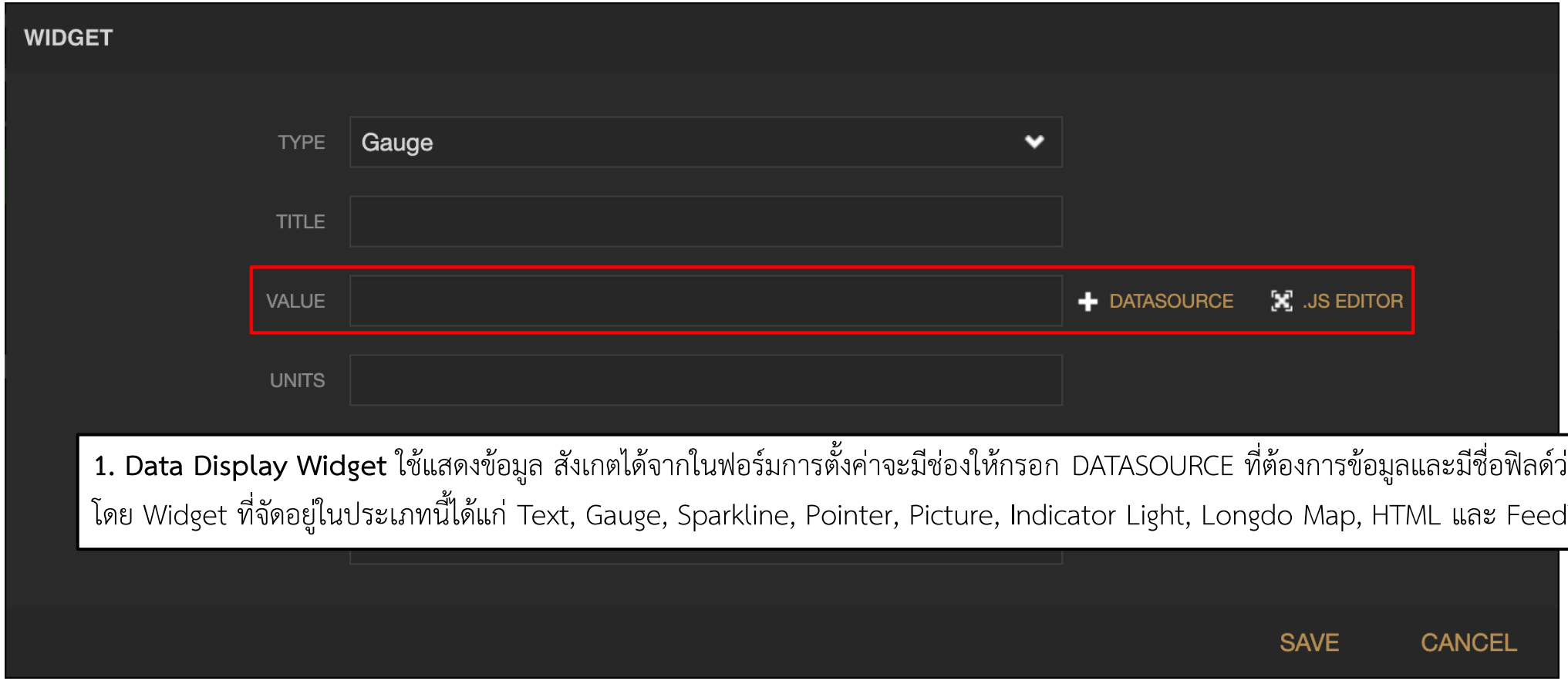
ข้อมูลที่อยู่บน Device2



ใบงานที่ 5.1 ทฤษฎีเบื้องต้น

การใช้งาน Widget

การเลือก Widget ต่างๆ ของ Freeboard เพื่อใช้ในการแสดงข้อมูลหรือควบคุมการทำงานของ Device โดย Widget จะแยกเป็น 2 ประเภท

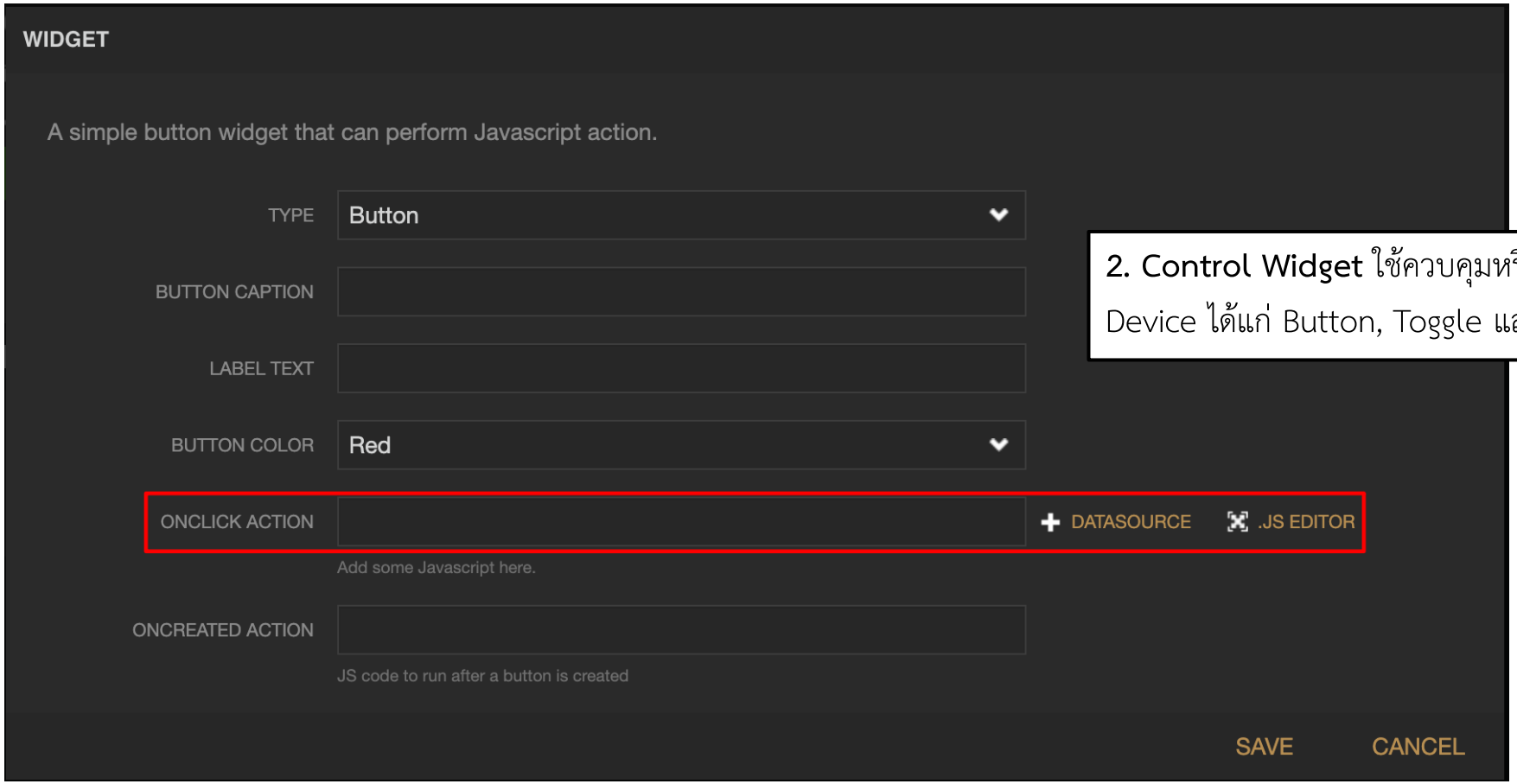


1. Data Display Widget ใช้แสดงข้อมูล สังเกตได้จากในฟอร์มการตั้งค่าจะมีช่องให้กรอก DATASOURCE ที่ต้องการข้อมูลและมีชื่อฟิลด์ว่า “VALUE” โดย Widget ที่จัดอยู่ในประเภทนี้ได้แก่ Text, Gauge, Sparkline, Pointer, Picture, Indicator Light, Longdo Map, HTML และ FeedView

ใบงานที่ 5.1 ทฤษฎีเบื้องต้น

การใช้งาน Widget

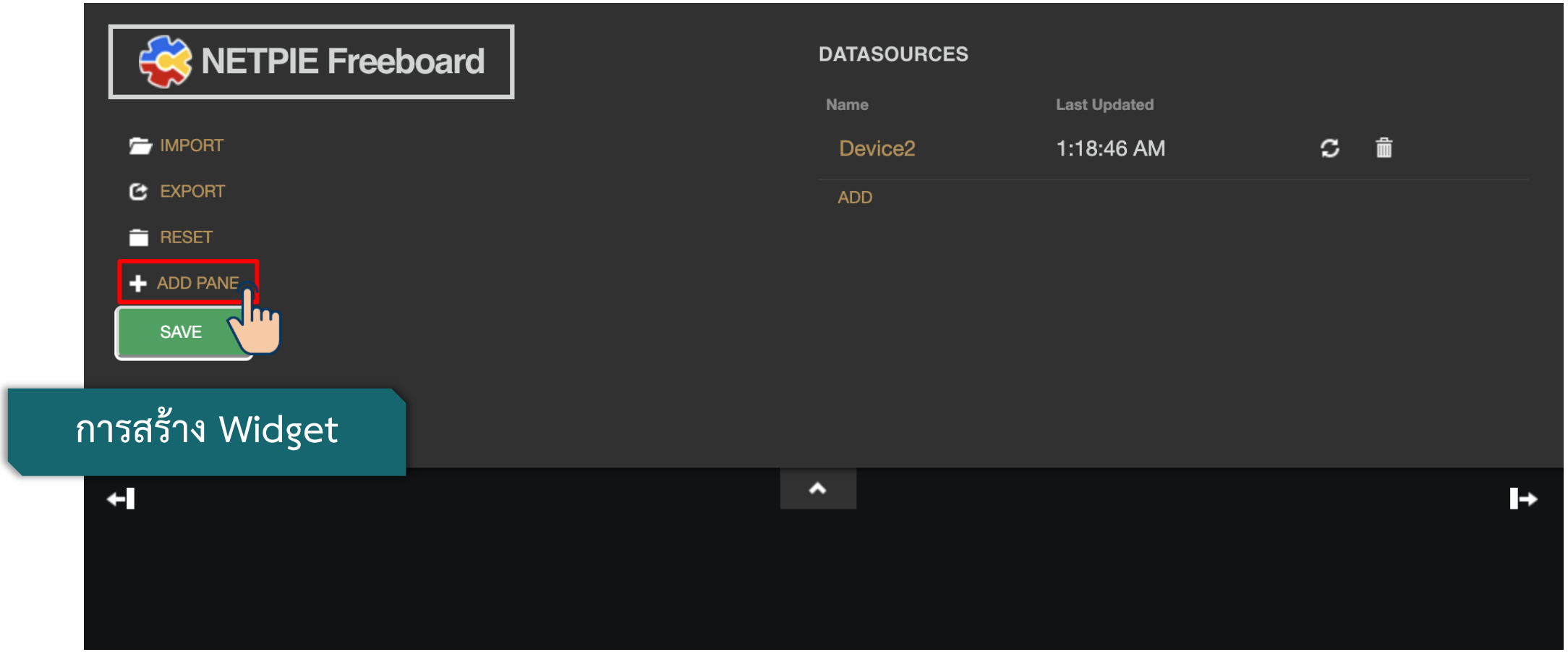
การเลือก Widget ต่างๆ ของ Freeboard เพื่อใช้ในการแสดงข้อมูลหรือควบคุมการทำงานของ Device โดย Widget จะแยกเป็น 2 ประเภท



2. Control Widget ใช้ควบคุมหรือสั่งการทำงานของ Device ได้แก่ Button, Toggle และ Slider

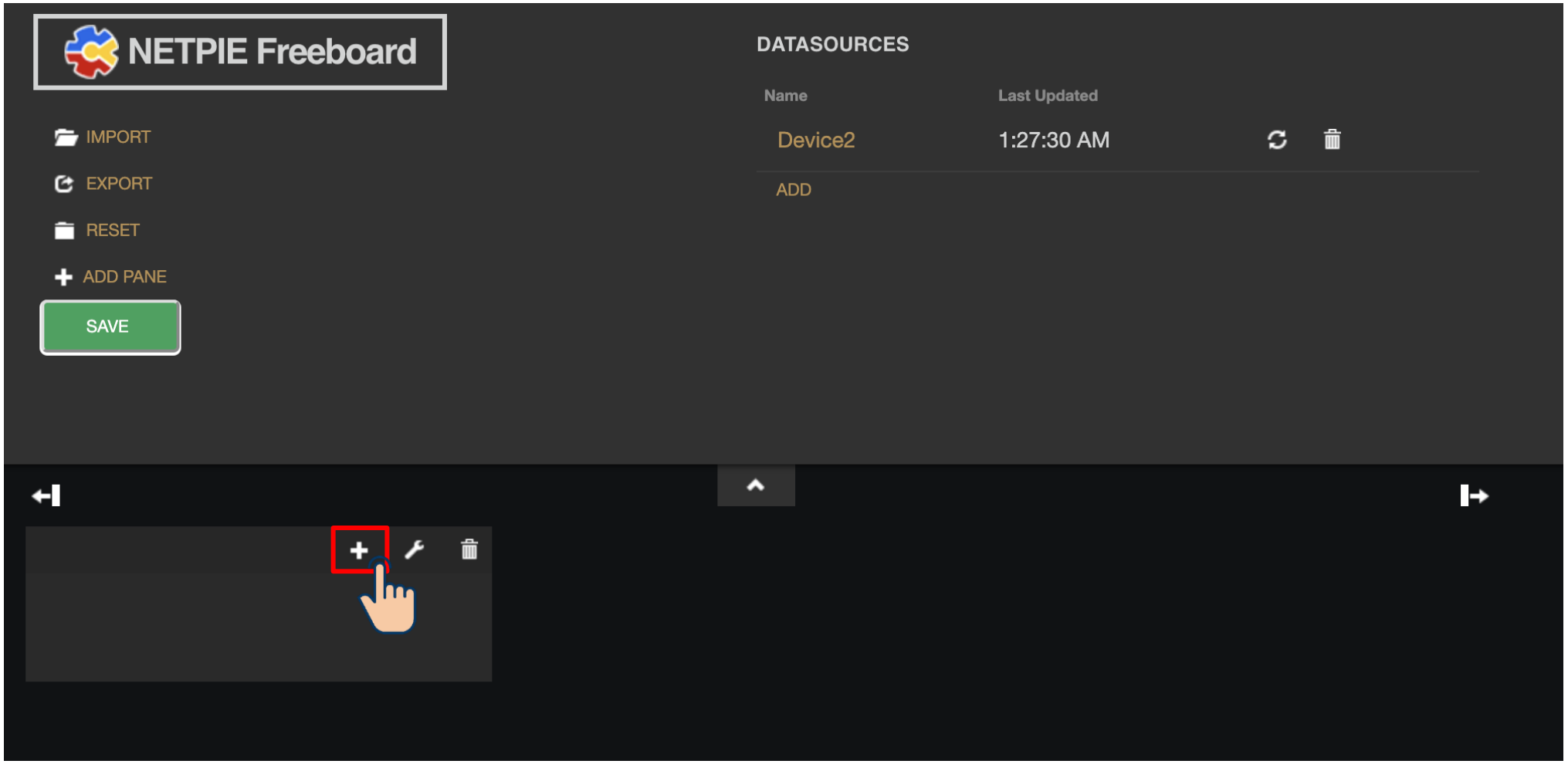
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



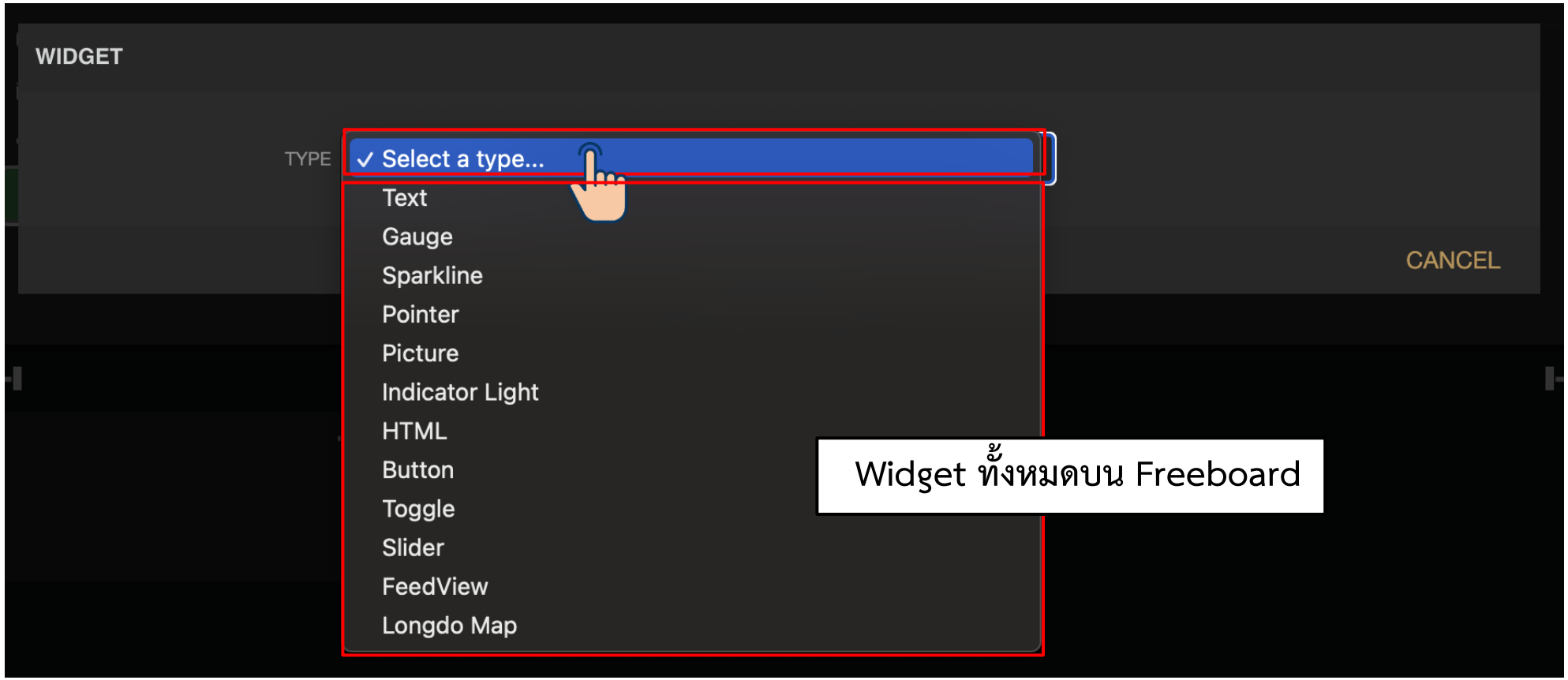
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

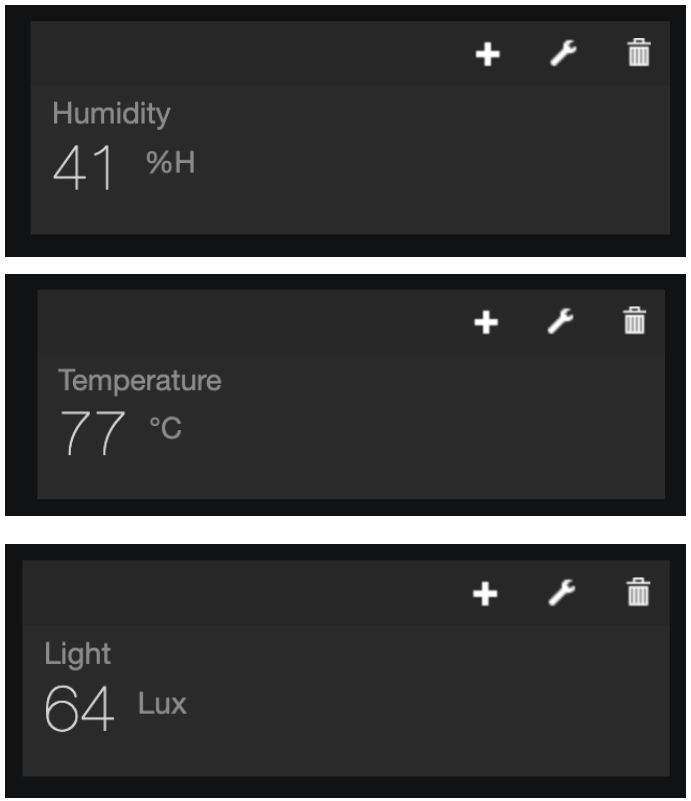


ใบงานที่ 5.1 ขั้นตอนการทดลอง

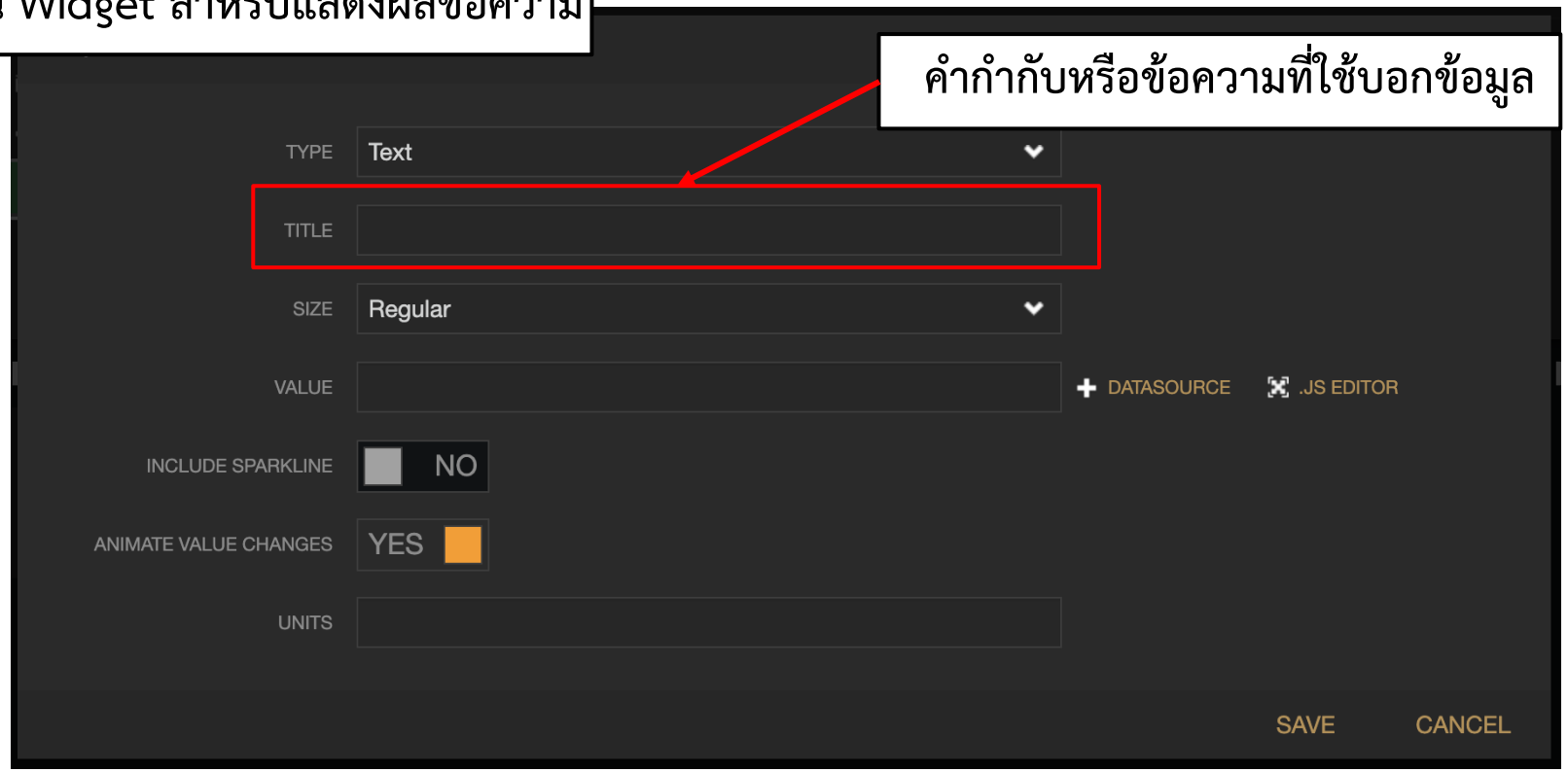
การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text

ตัวอย่าง Widget : Text



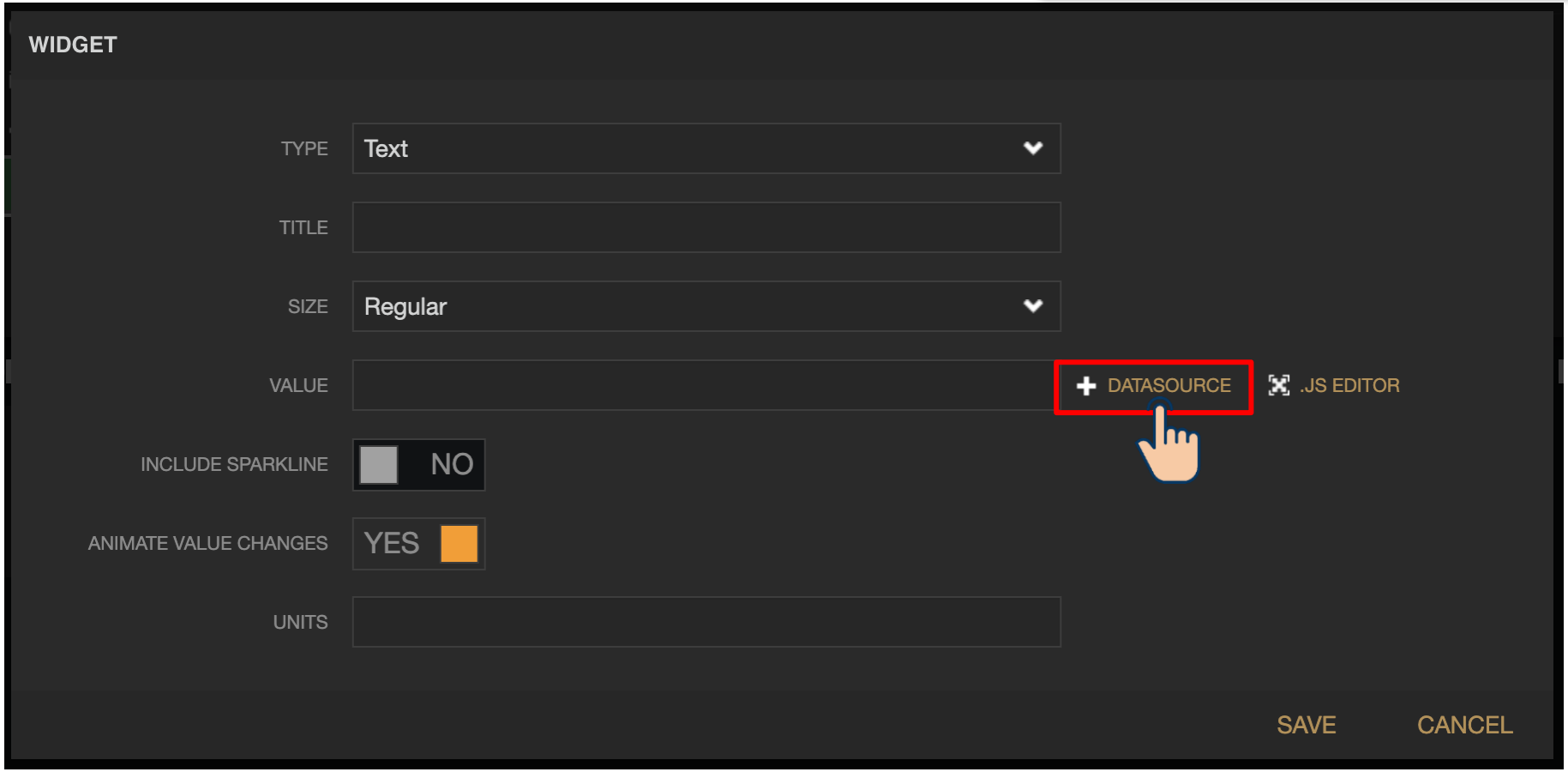
เป็น Widget สำหรับแสดงผลข้อความ



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

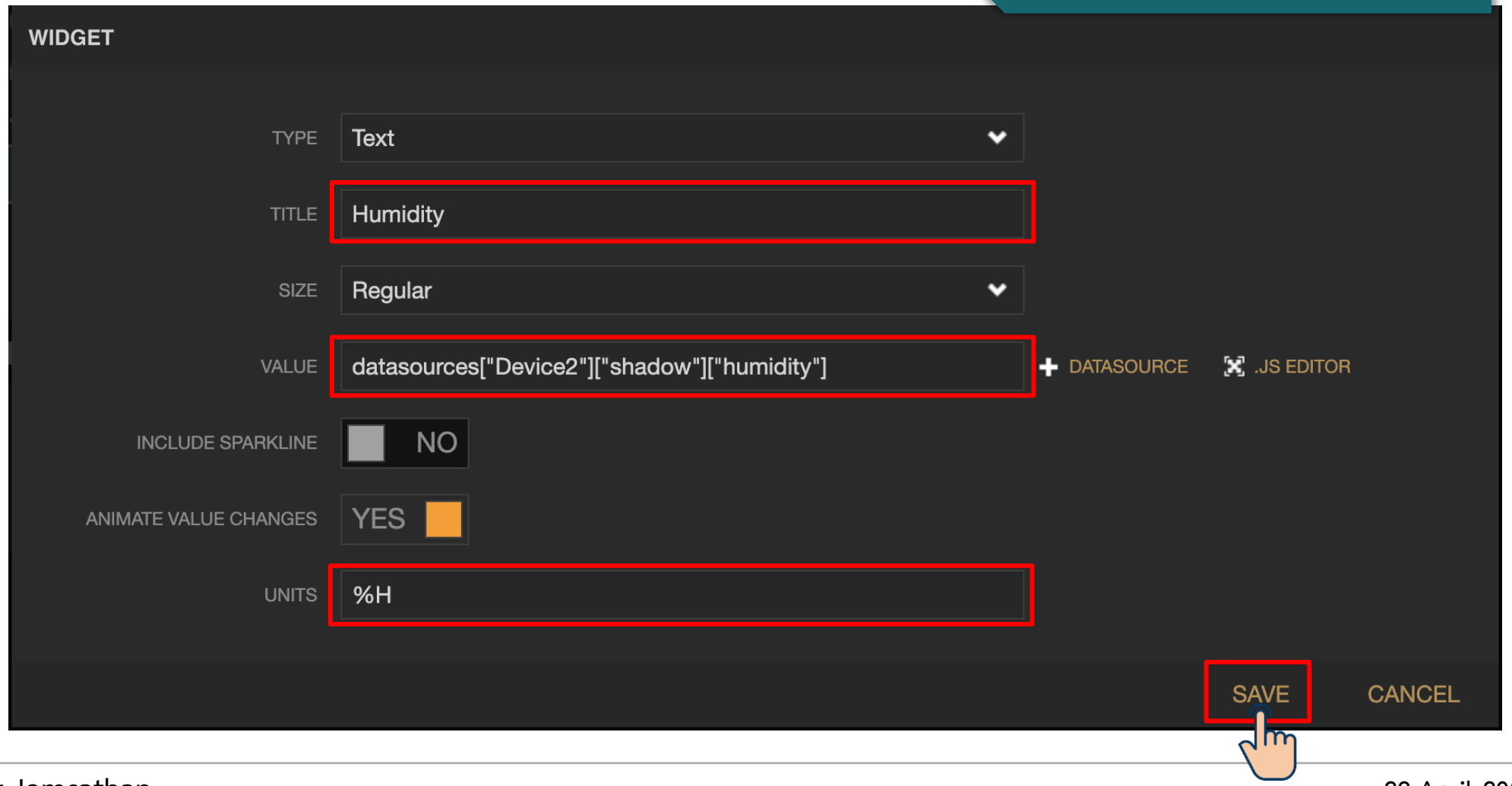
การสร้าง Widget : Text



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

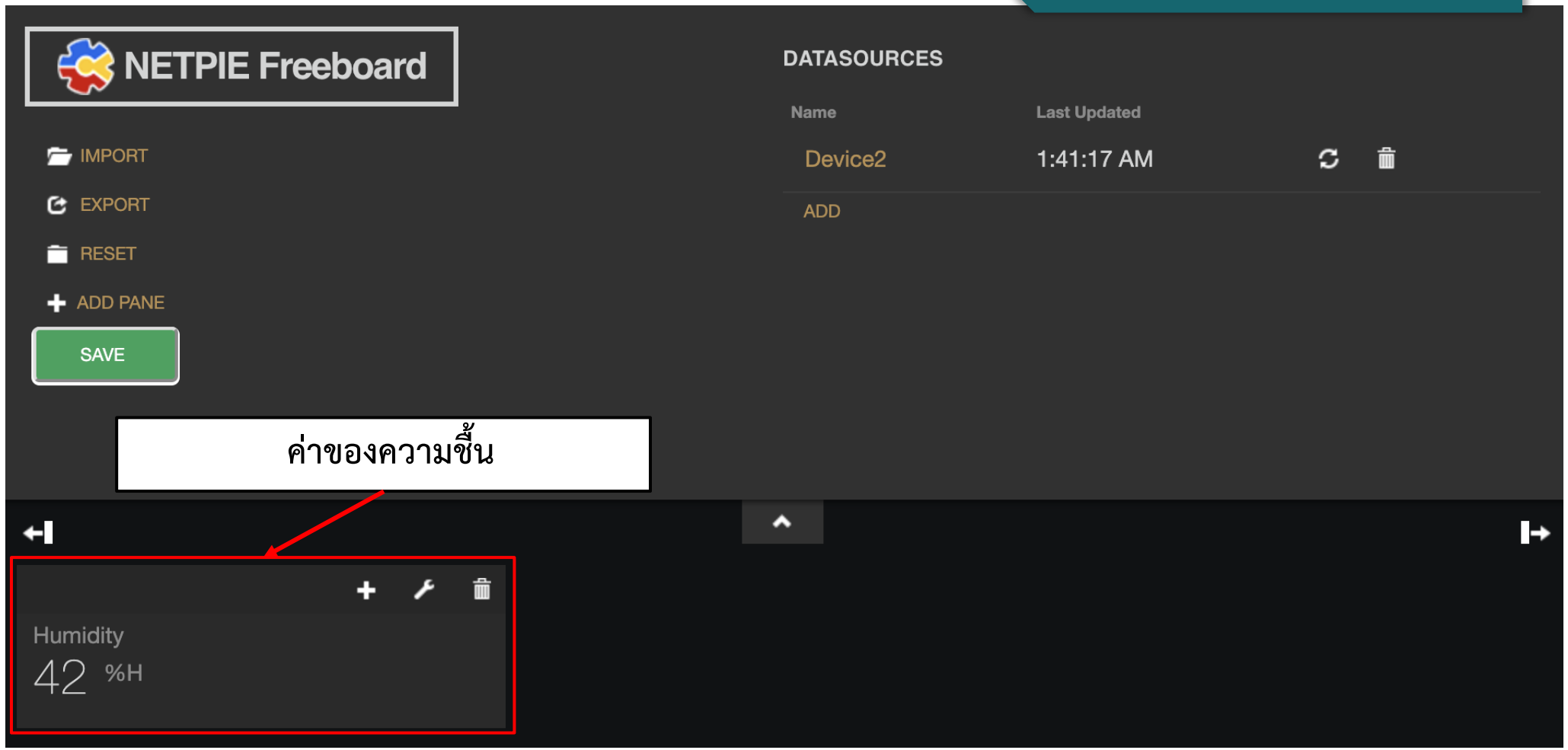
การสร้าง Widget : Text



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

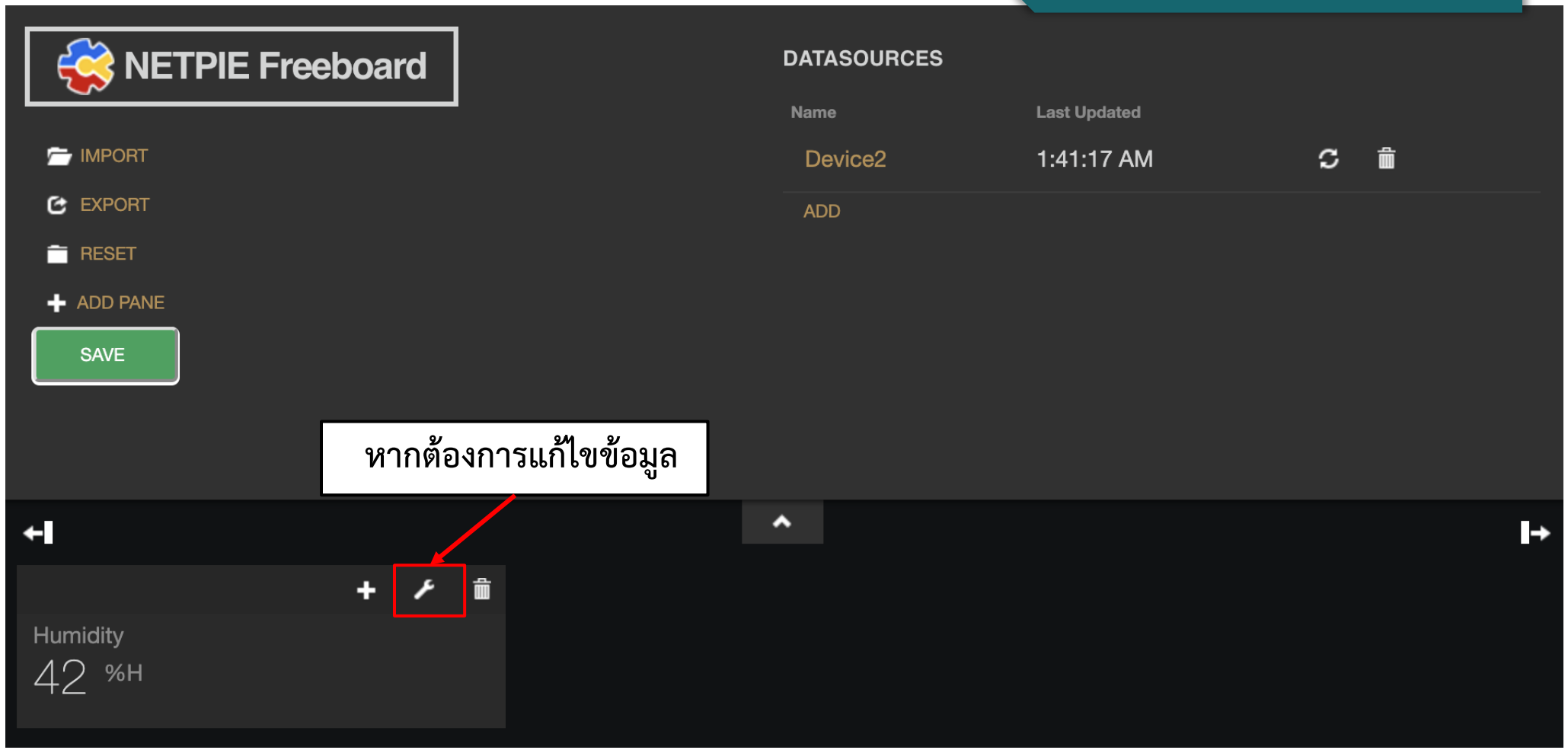
การสร้าง Widget : Text



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

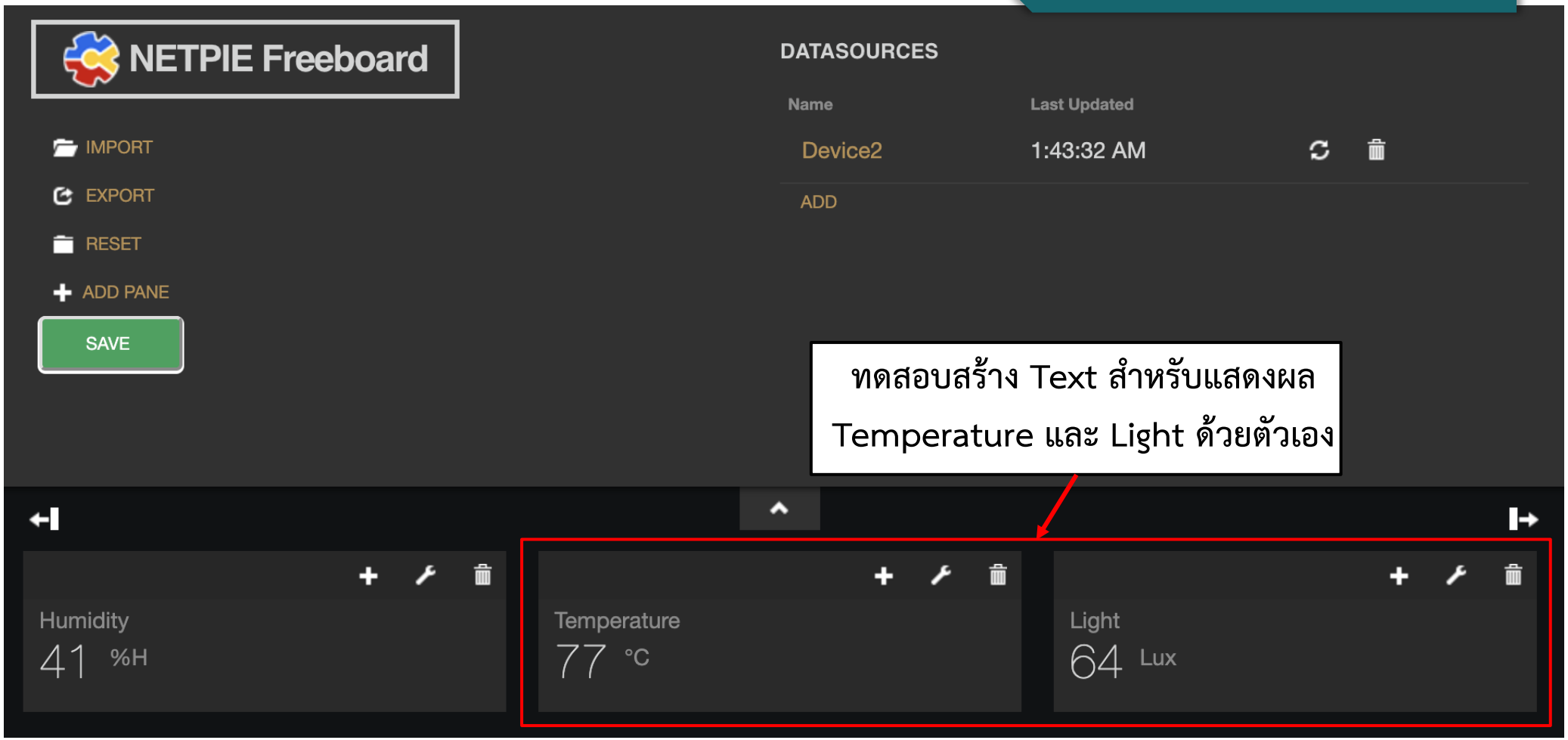
การสร้าง Widget : Text



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text

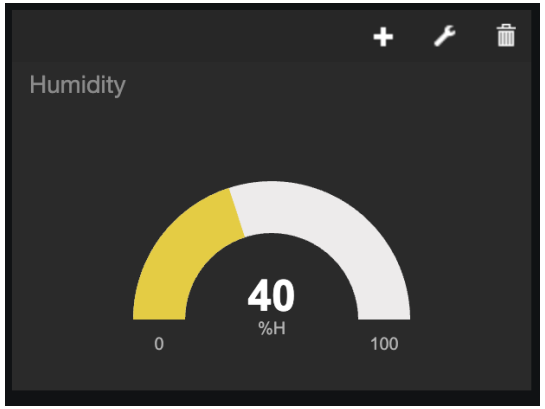


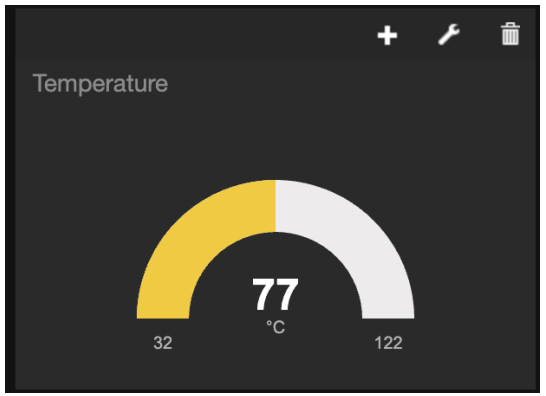
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

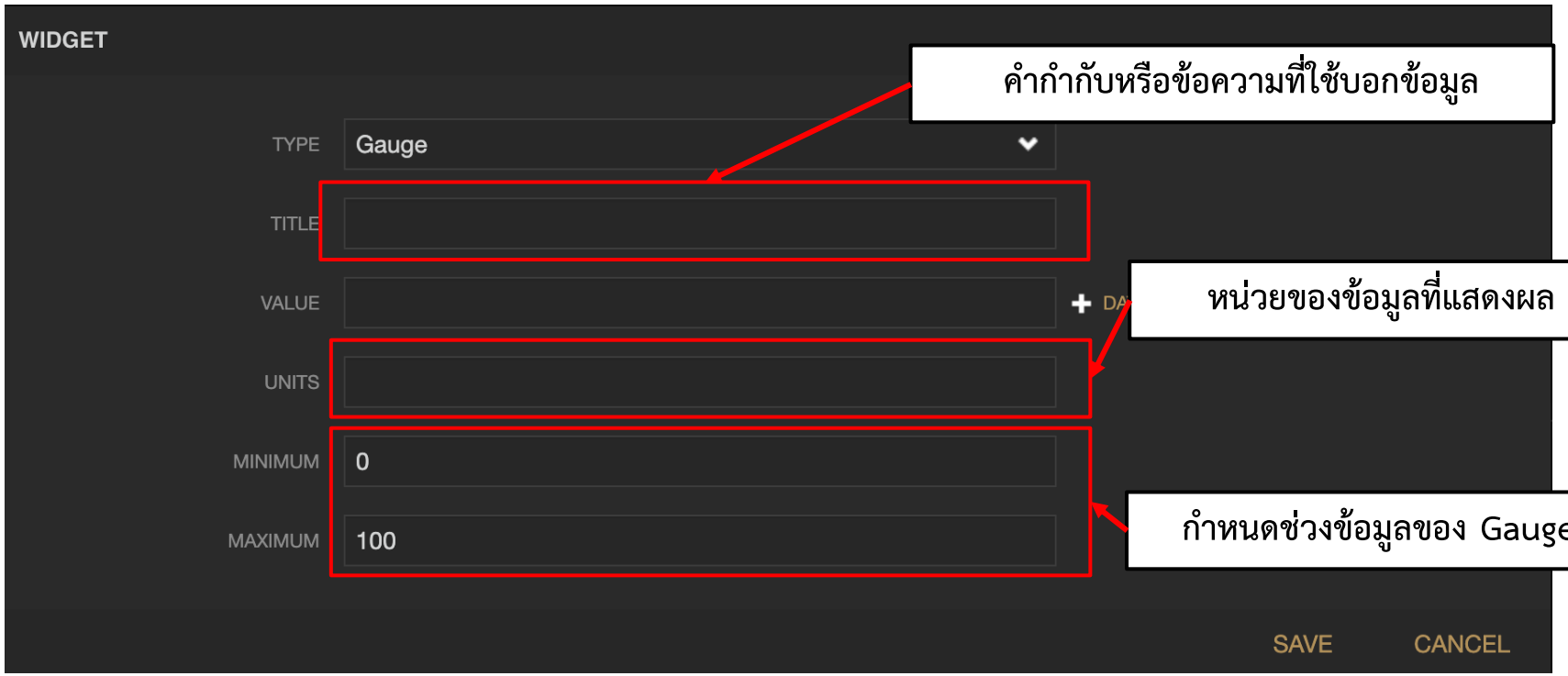
การสร้าง Widget : Gauge

ตัวอย่าง Widget : Gauge





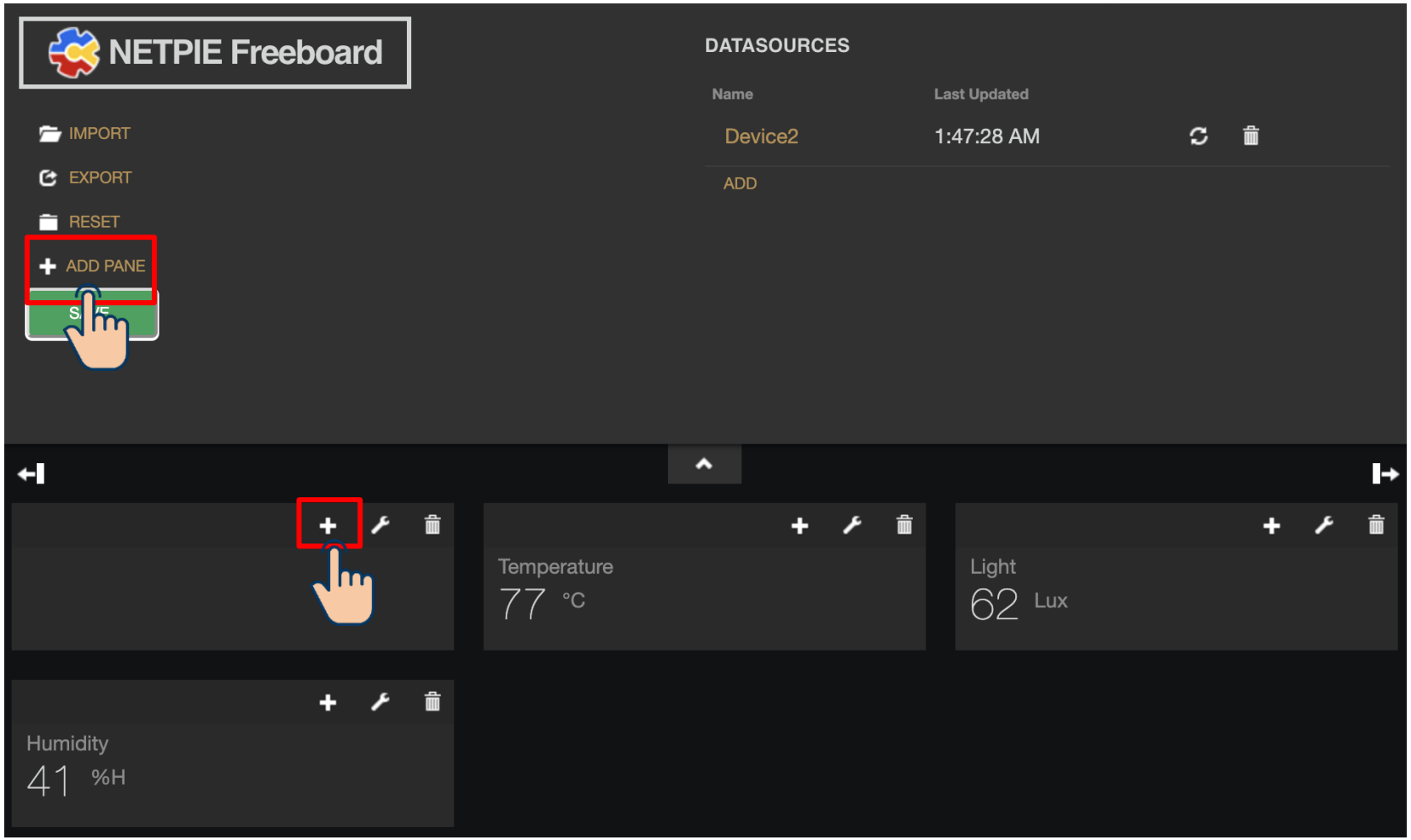
เป็น Widget สำหรับแสดงผลข้อมูลแบบ Gauge



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

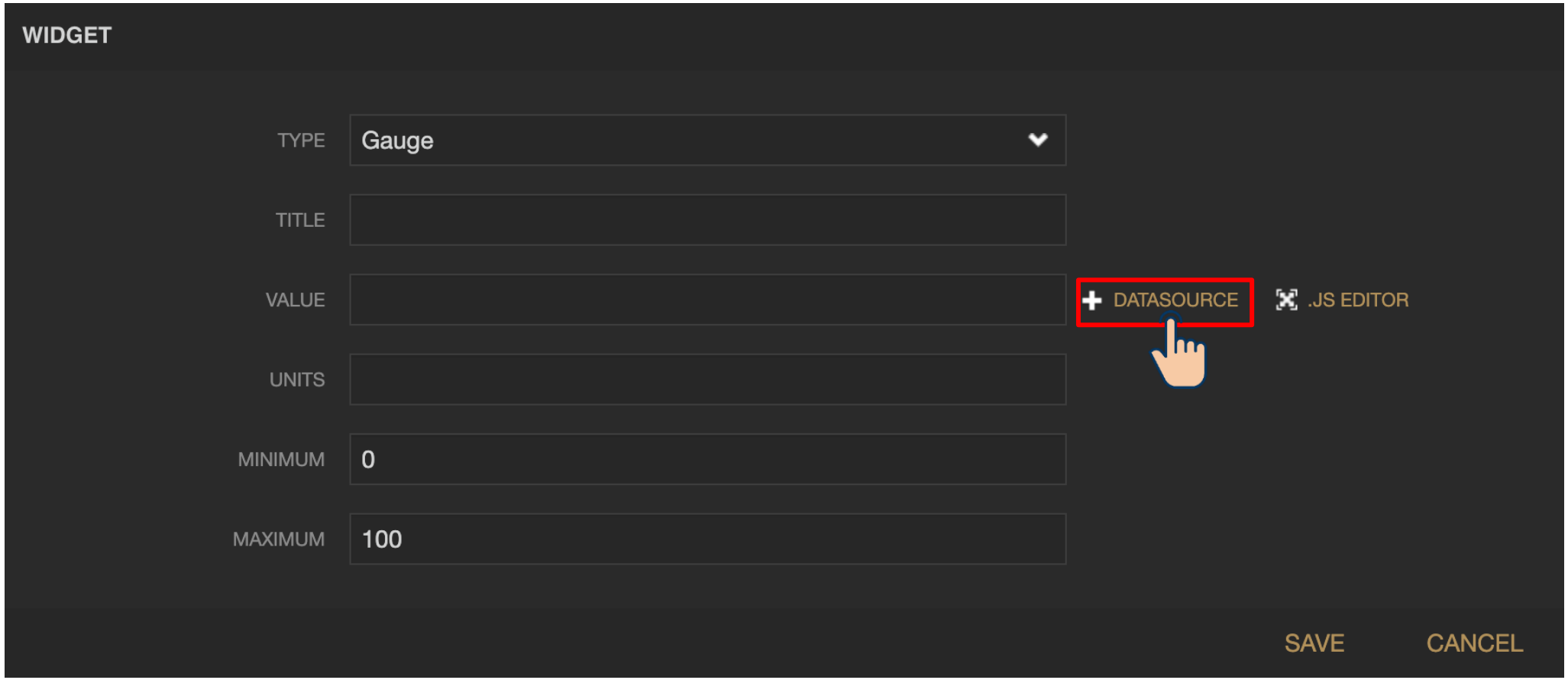
การสร้าง Widget : Gauge



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

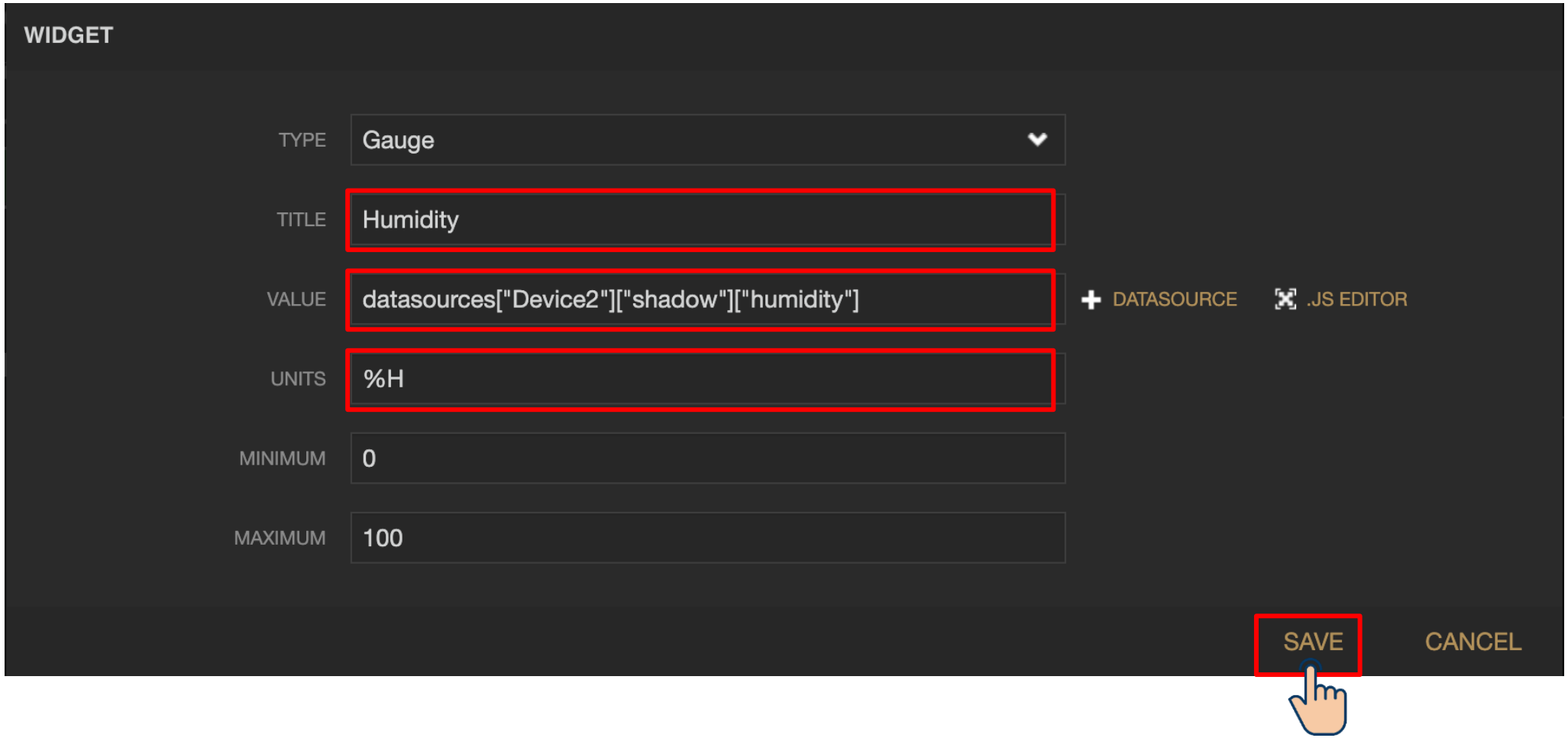
การสร้าง Widget : Gauge



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

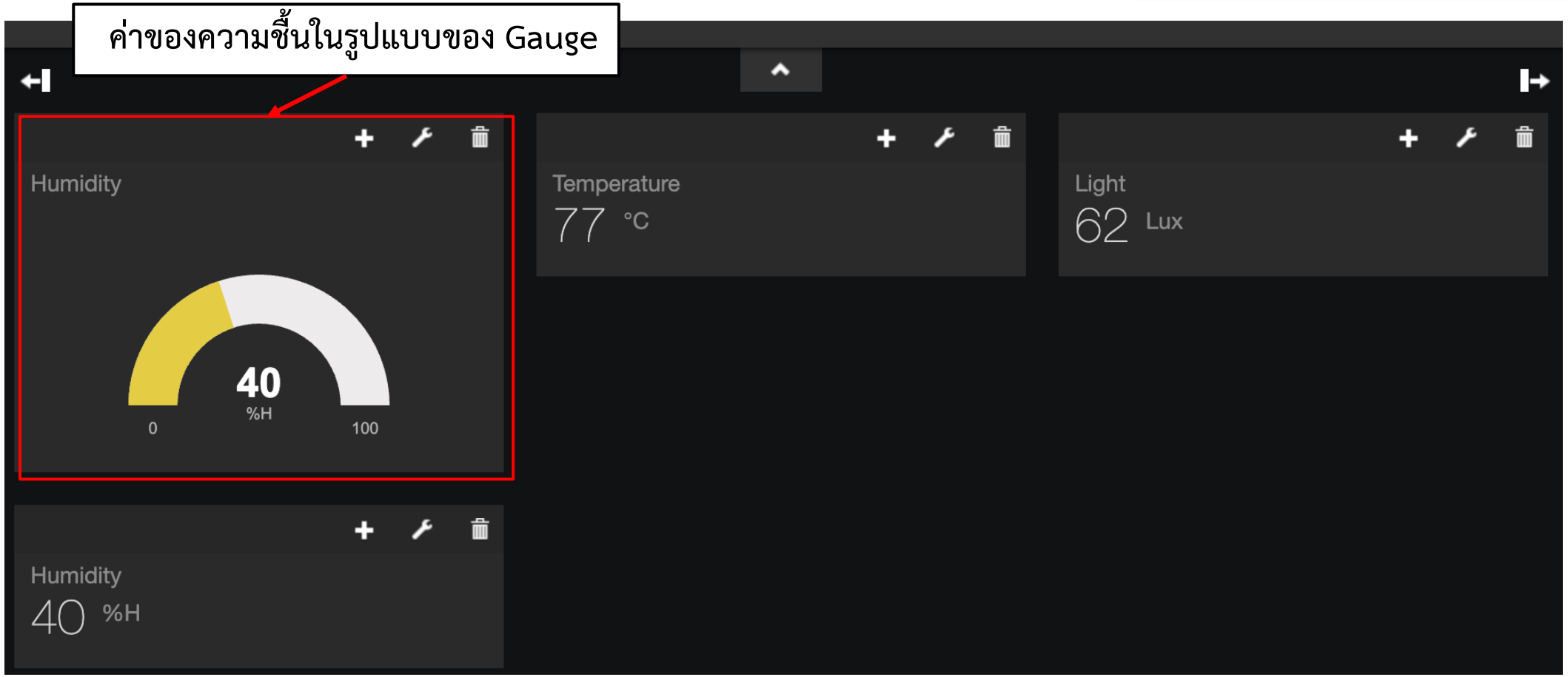
การสร้าง Widget : Gauge



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

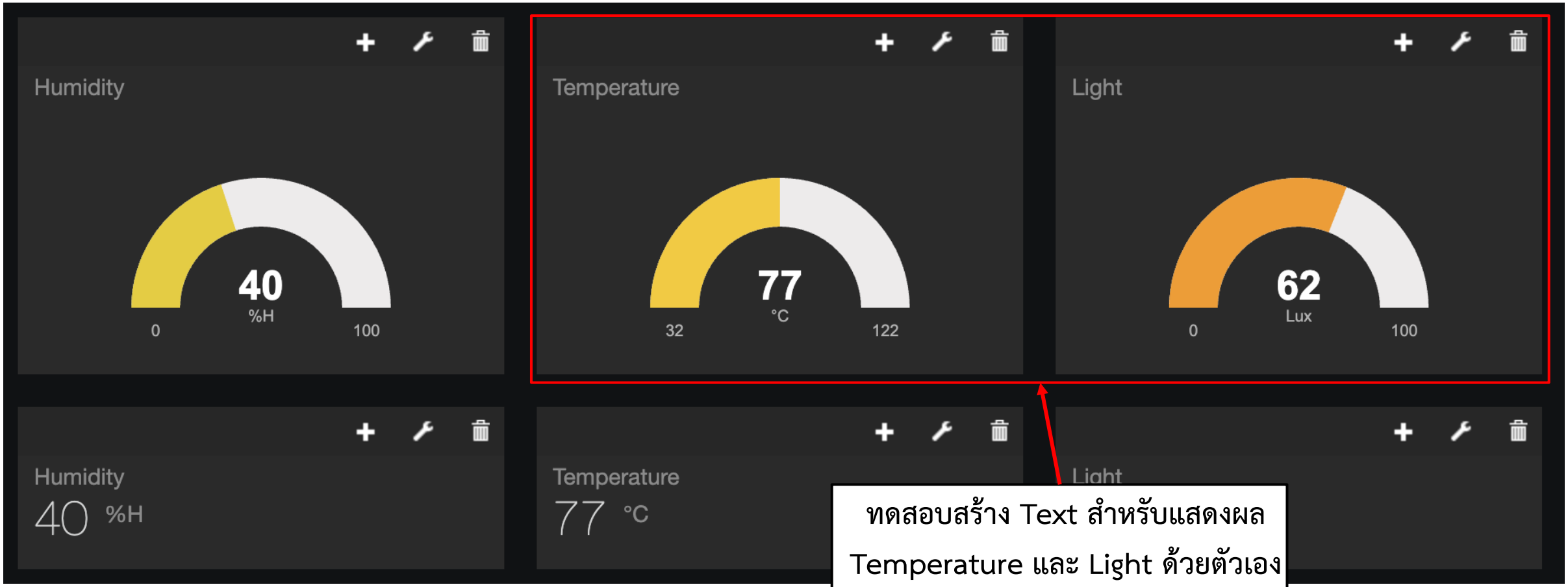
การสร้าง Widget : Gauge



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge

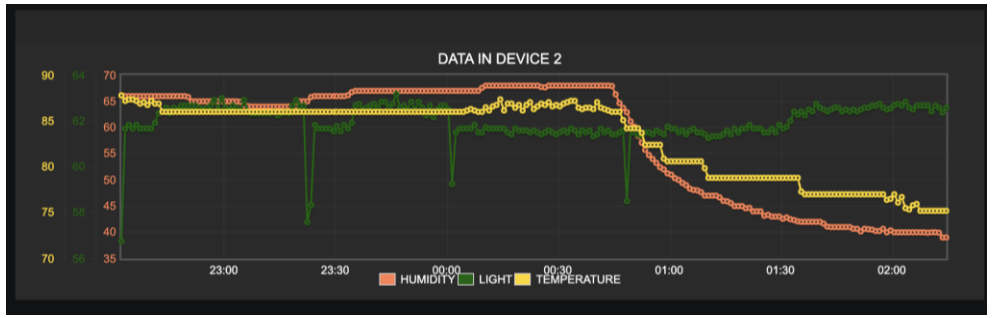


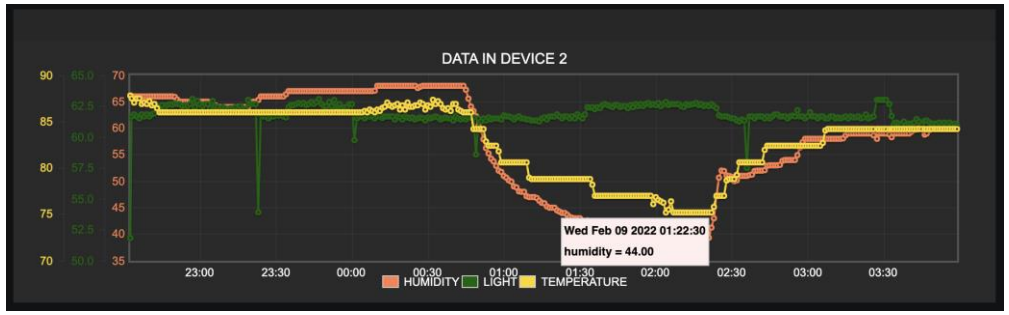
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

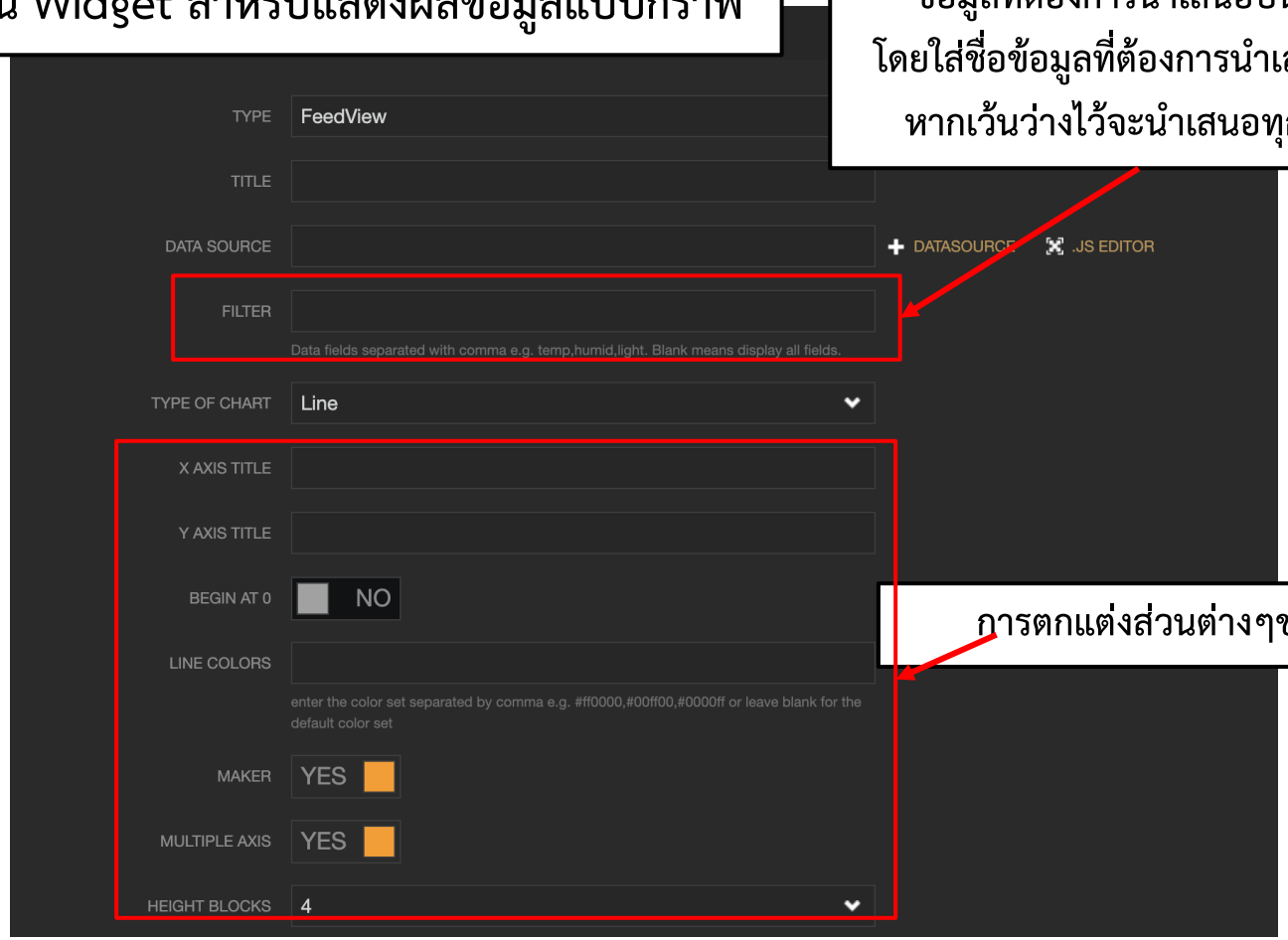
ตัวอย่าง Widget : FeedView





เป็น Widget สำหรับแสดงผลข้อมูลแบบกราฟ

ข้อมูลที่ต้องการนำเสนอบนกราฟ
โดยใส่ชื่อข้อมูลที่ต้องการนำเสนอลงไป
หากเว้นว่างไว้จะนำเสนอทุกข้อมูล

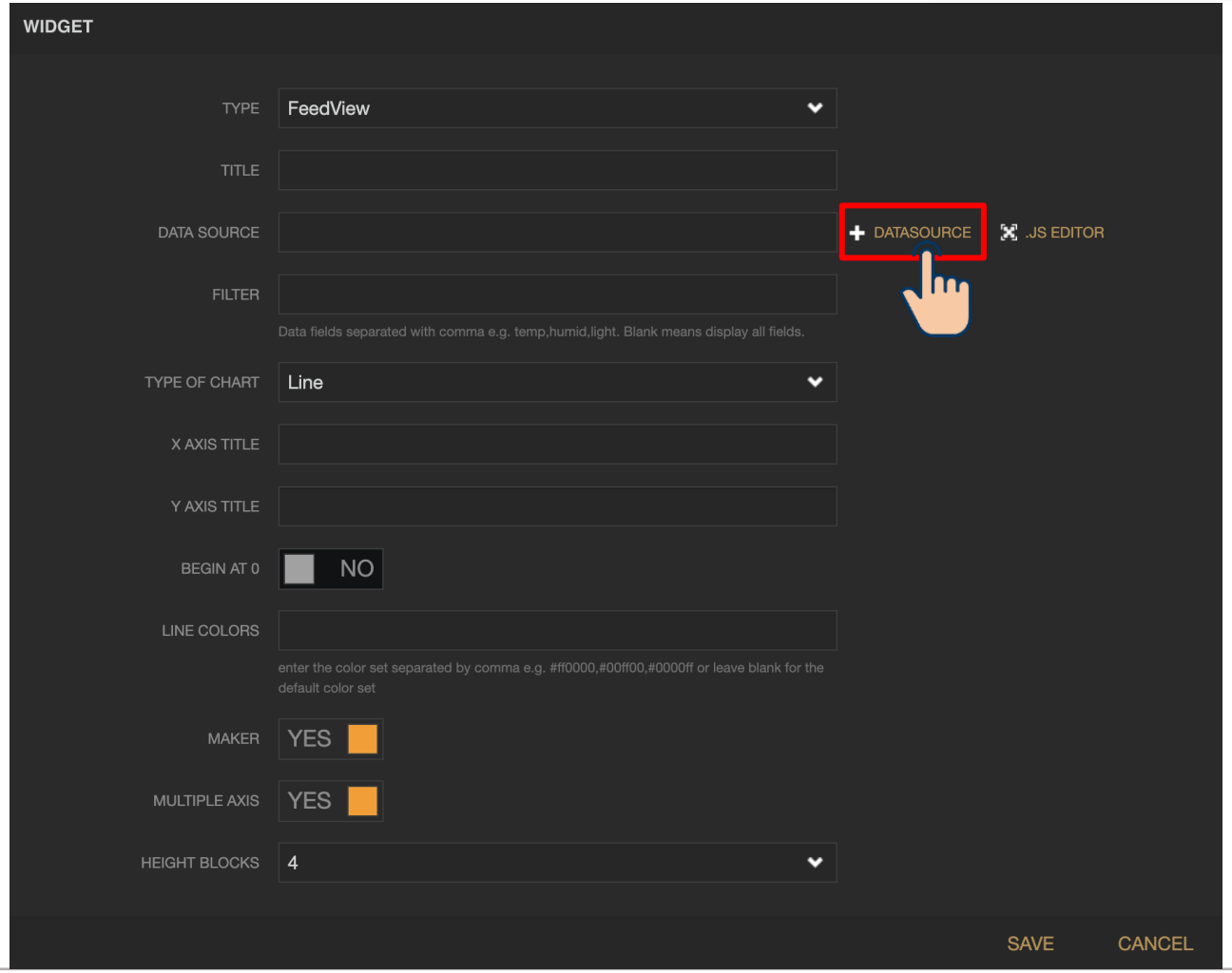


การตกแต่งส่วนต่างๆของกราฟ

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView



ใบงานที่ 5.1 ขั้นตอนการทดลอง

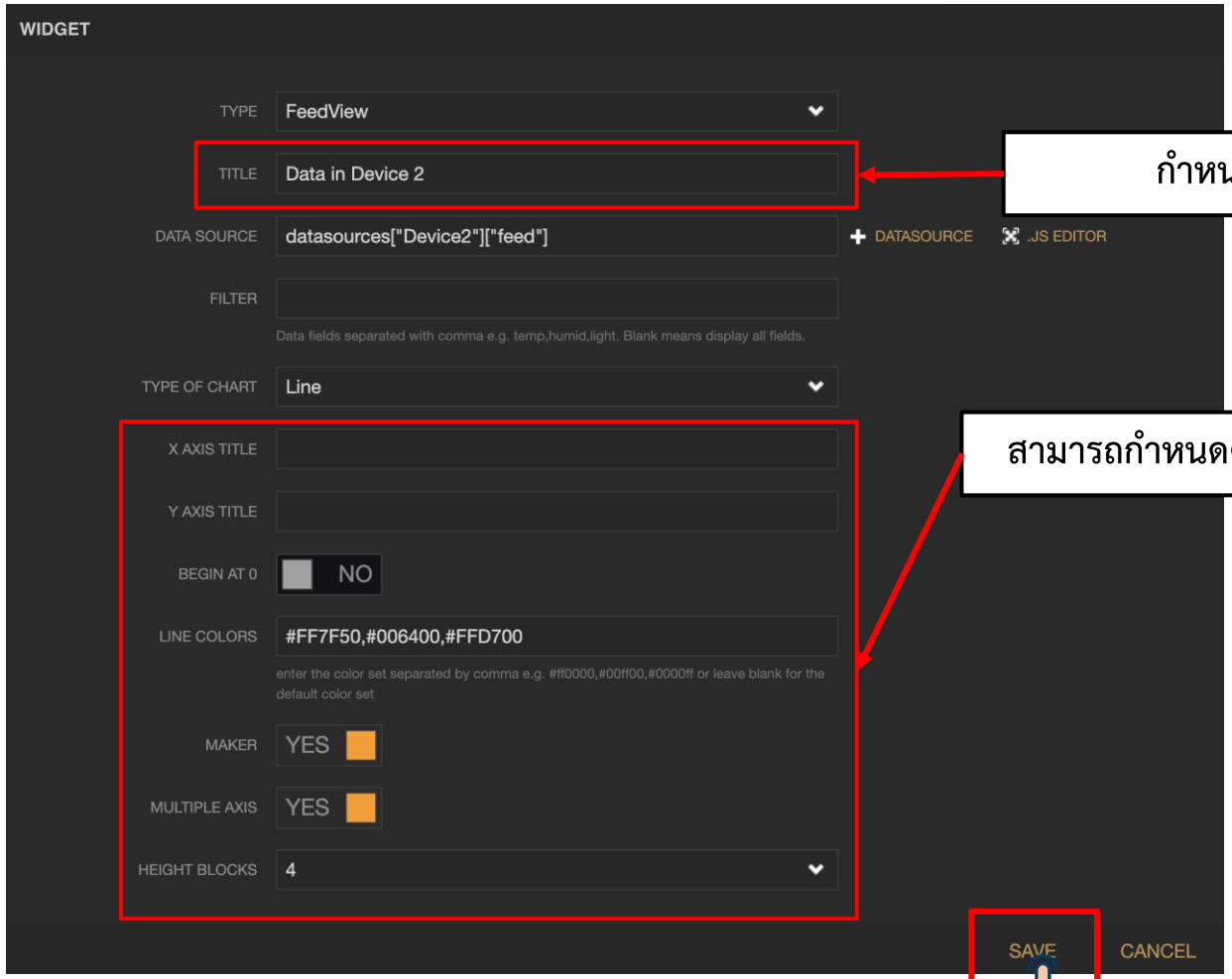
การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView



กำหนดหัวข้อของกราฟ

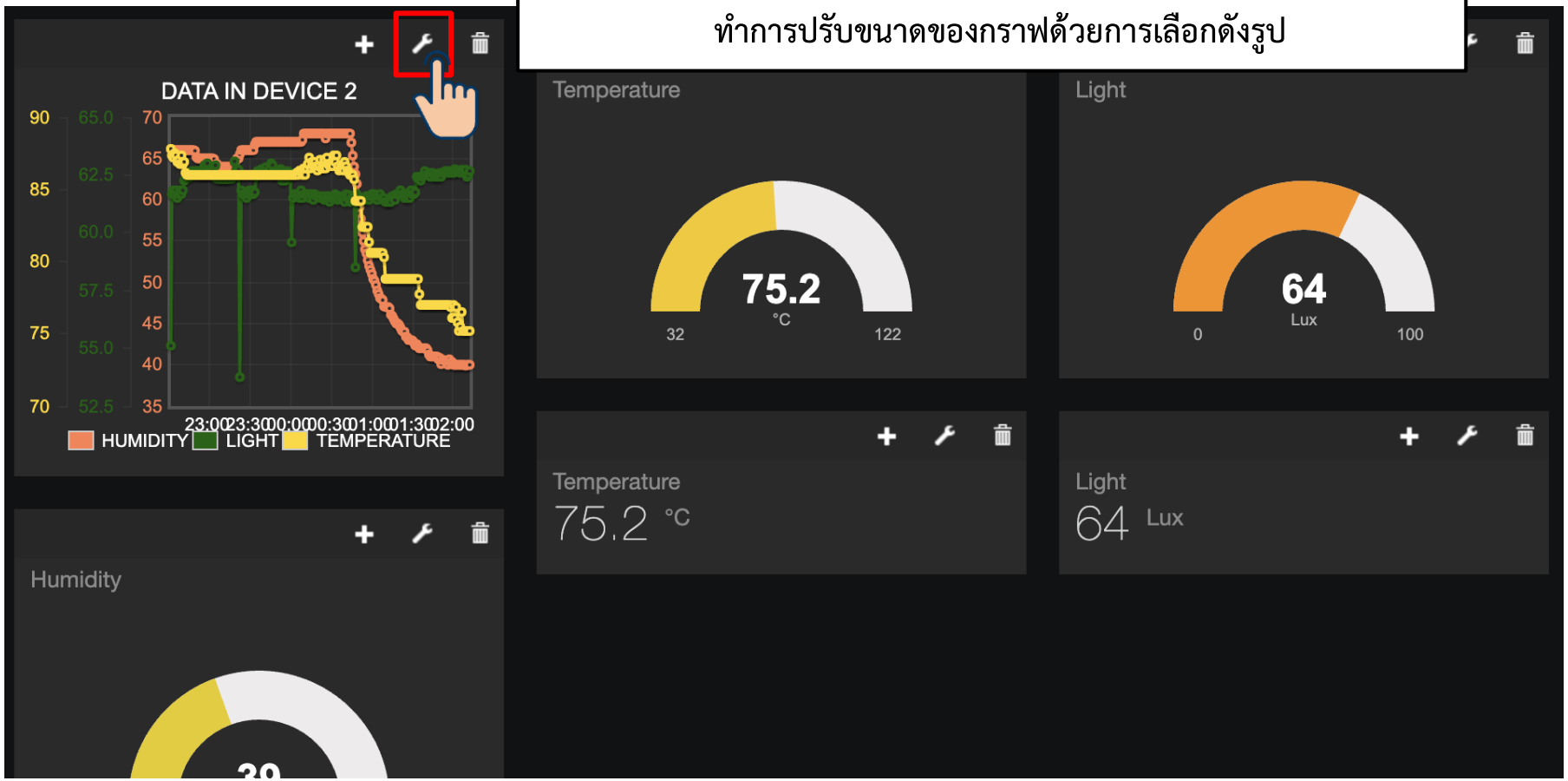
สามารถกำหนดค่าต่างๆได้เพื่อเป็นการตกแต่งกราฟ

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

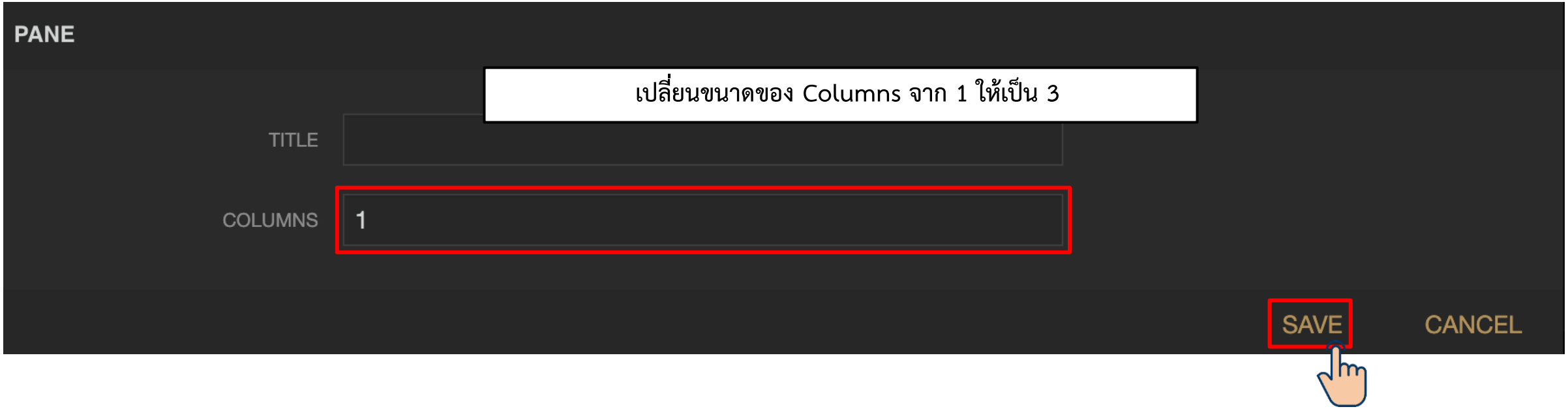
หลังจาก Save แล้วจะได้กราฟมา ซึ่งขนาดของกราฟอาจจะไม่เหมาะสม ทำการปรับขนาดของกราฟด้วยการเลือกดังรูป



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

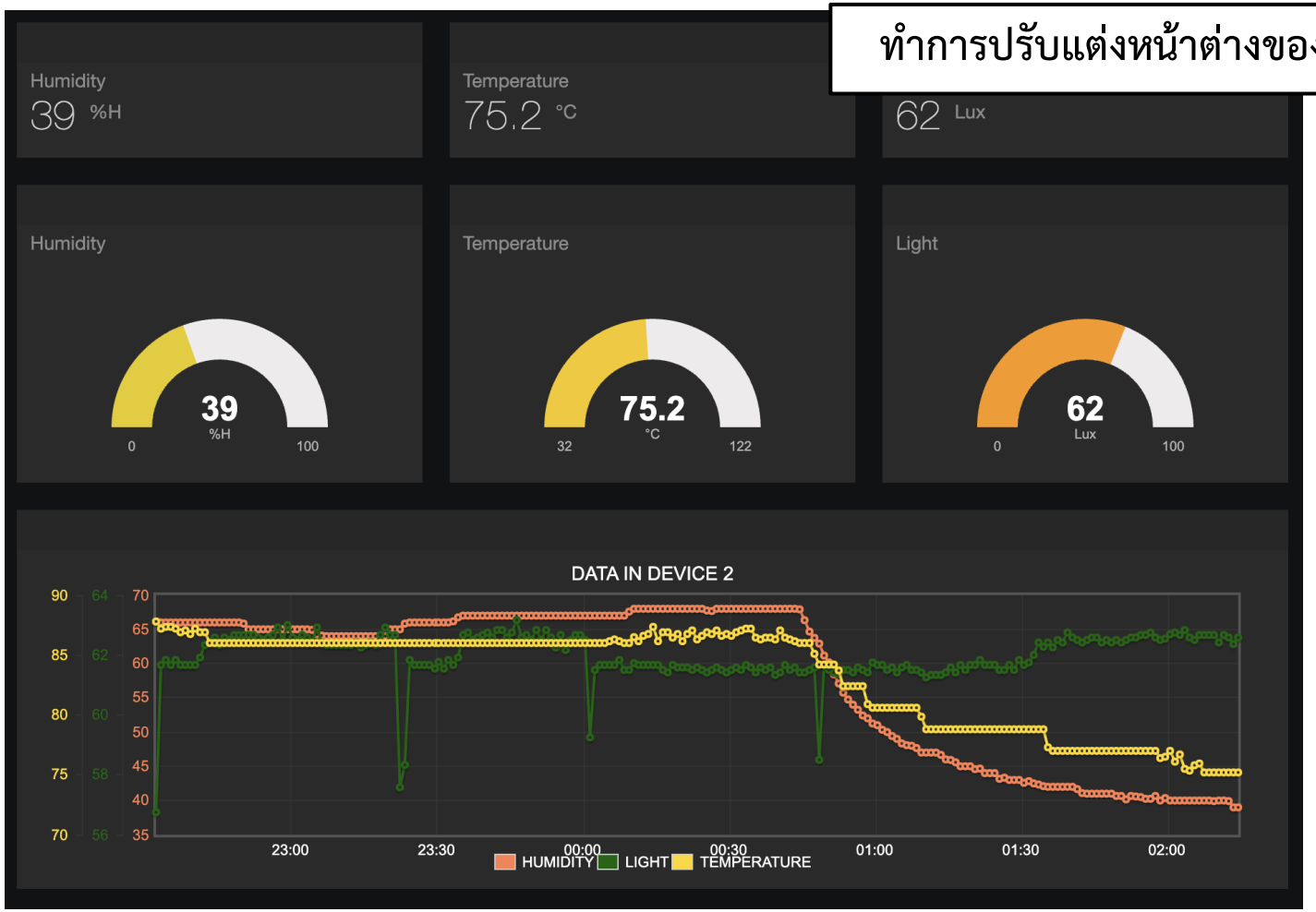


ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

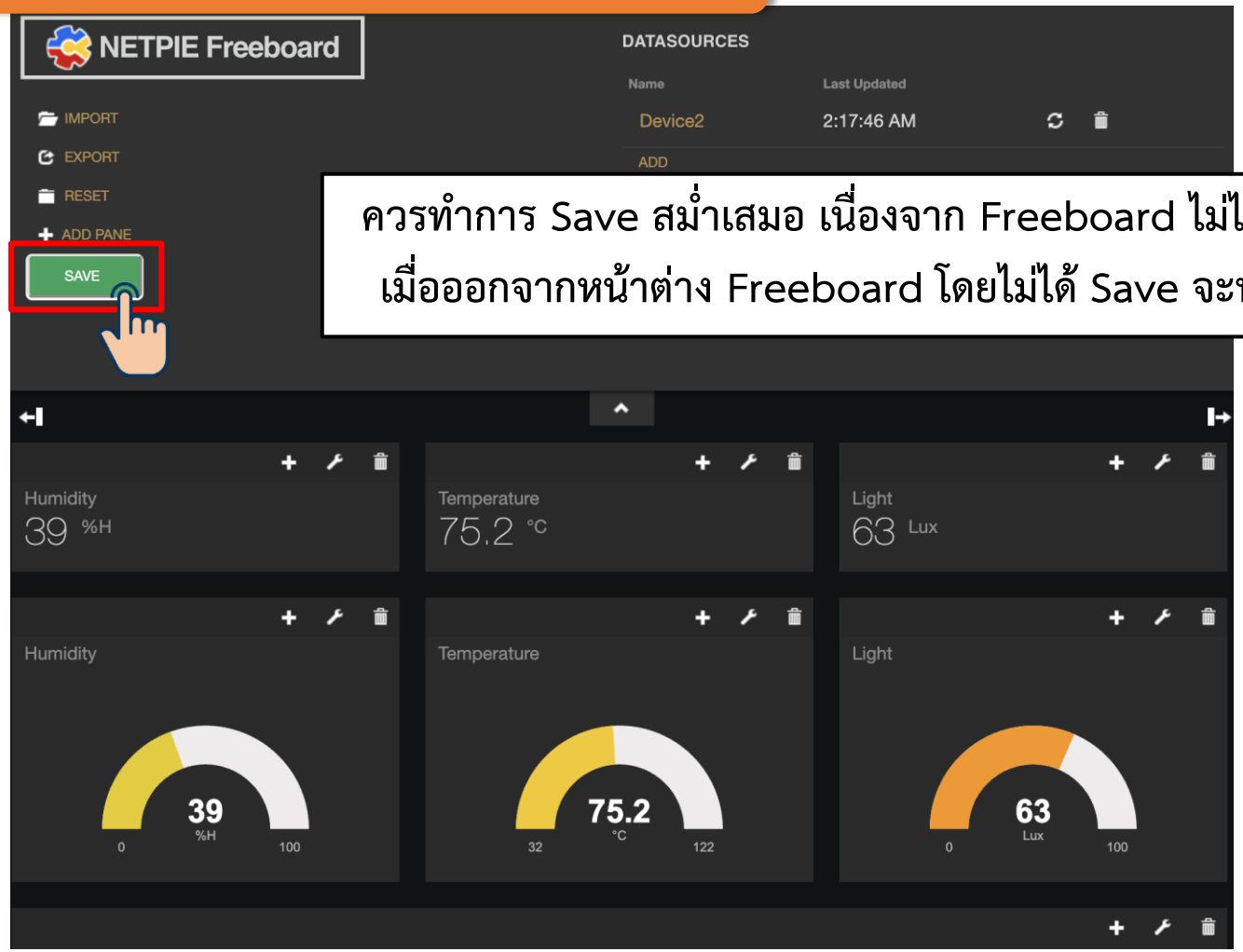
ทำการปรับแต่งหน้าต่างของ Freeboard



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การ Save Freeboard



ควรทำการ Save สม่ำเสมอ เนื่องจาก Freeboard ไม่ได้ทำการ Save ให้อัตโนมัติ เมื่อออกจากหน้าต่าง Freeboard โดยไม่ได้ Save จะทำให้ข้อมูลทั้งหมดหายไป

ใบงานที่ 5.1 คำถามท้ายหน่วยการเรียนรู้

คำถามท้ายหน่วยการเรียนรู้ที่ 1

จงสร้าง Dashboard สำหรับแสดงผลข้อมูลของ ESP32 ที่ถูกจัดเก็บบน Schema จากคำถามท้ายการทดลองในสาระการเรียนรู้ที่ 4 ข้อที่ 11 โดยกำหนดให้มี Widget ในแต่ละข้อมูลดังนี้

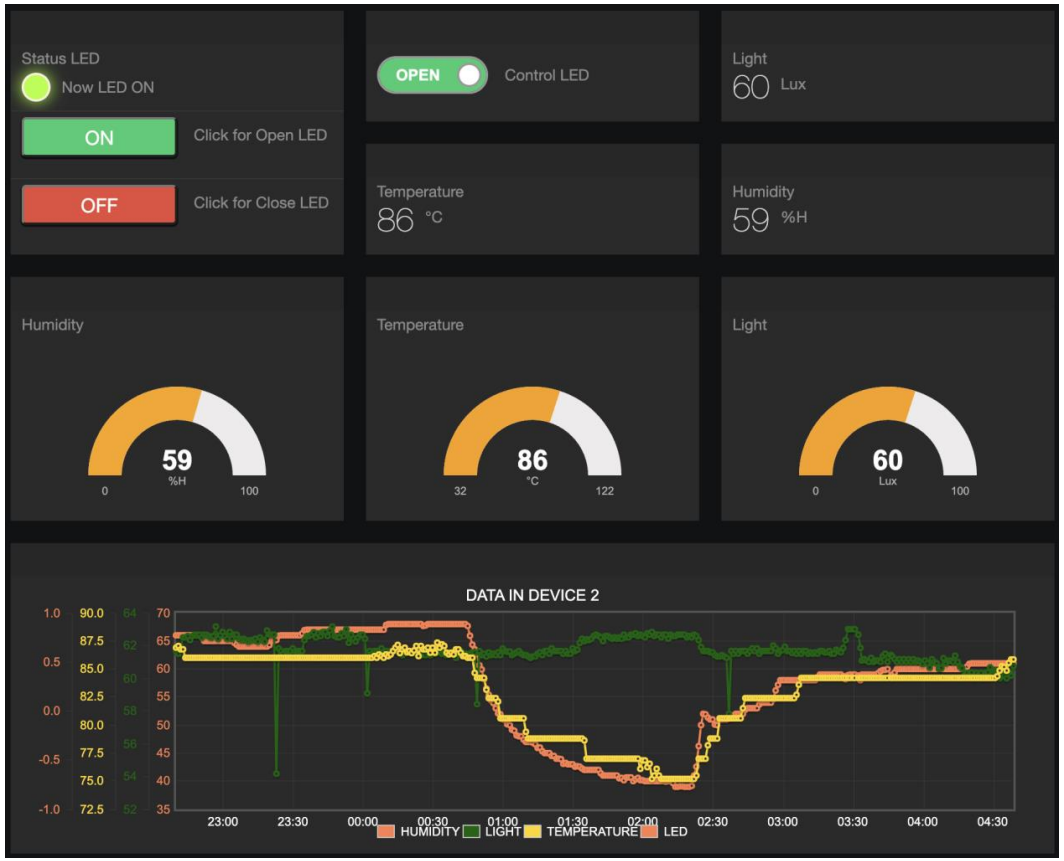
1. Widget : Text แสดงผลข้อมูล voltage, gas, led1, led2 และ led3
2. Widget : Gauge แสดงผลข้อมูล voltage และ gas
3. Widget : FeedView แสดงผลข้อมูล Voltage และ Gas

โดยทุก Widget ของแต่ละข้อมูลกำหนดให้เป็นอิสระจากกัน (นั่นคือ ต้องมี FeedView 2 อัน)

*** เสริมโจทย์ การแปลงข้อมูลบน Widget : Text ตามเงื่อนดังต่อไปนี้ได้

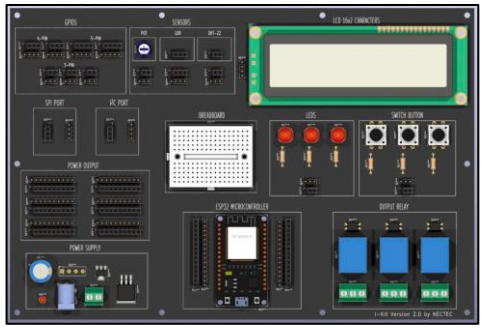
- led1, led2 และ led3 ปกติแล้วข้อมูลจะนำเสนอจาก 0 หรือ false ให้แสดงผลข้อมูลเป็น LED OFF
- led1, led2 และ led3 ปกติแล้วข้อมูลจะนำเสนอจาก 1 หรือ true ให้แสดงผลข้อมูลเป็น LED ON

ใบงานที่ 5.2 การควบคุมอุปกรณ์ด้วย Freeboard



ใบงานที่ 5.2 ทฤษฎีเบื้องต้น

หลักการสื่อสารระหว่างอุปกรณ์และ Freeboard



ESP32

Topic = @shadow/data/update

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

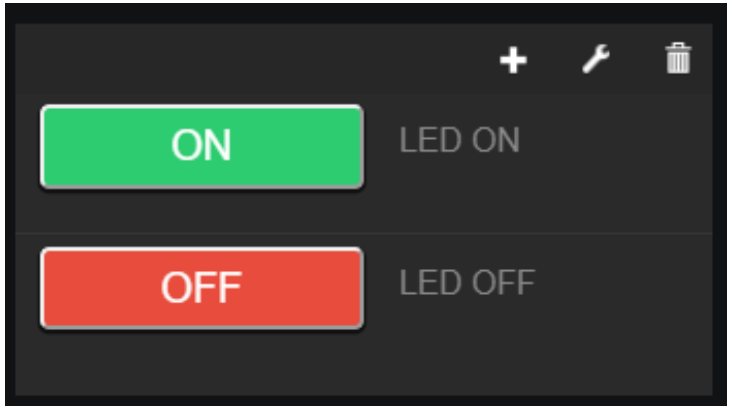
@msg/led : "ON/OFF"

NETPIE

โดยปกติแล้วการส่งข้อมูลจากอุปกรณ์ไปยัง Device บน NETPIE
เราต้องใช้ Shadow Topic เพื่อให้ข้อมูลถูกจัดเก็บลง Device Shadow
แต่ในกรณี ของ Freeboard ไปยัง Device เราจะใช้วิธีการดังนี้

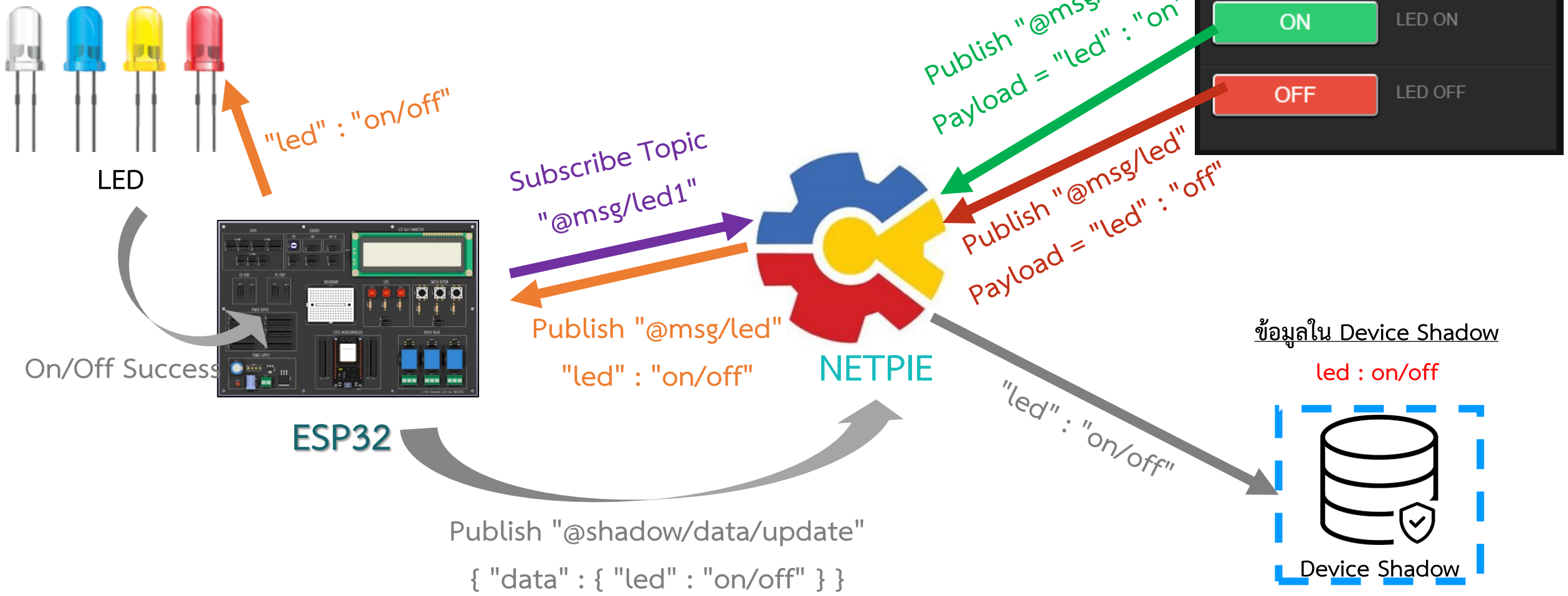
1. Freeboard จะส่งข้อความผ่าน Message Topic ไปยัง ESP32 เพื่อควบคุมอุปกรณ์
2. เมื่ออุปกรณ์ทำงานตามคำสั่งที่ได้รับเสร็จสิ้น อุปกรณ์จะส่งข้อมูลกลับไปผ่าน Shadow Topic เพื่ออัพเดทสถานะการทำงานของอุปกรณ์ที่ควบคุม ดังรูปหน้าถัดไป

Freeboard



ใบงานที่ 5.2 ทฤษฎีเบื้องต้น

หลักการสื่อสารระหว่างอุปกรณ์และ Freeboard

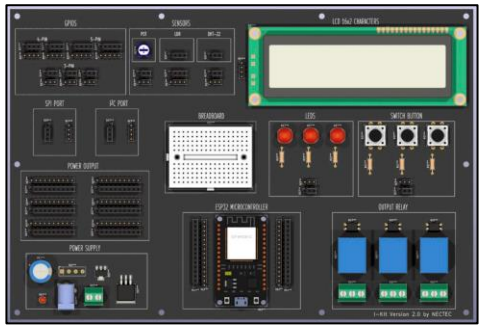


ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

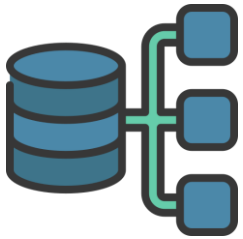
เช็คความพร้อมเพื่อทดสอบทั้งหมด 3 ส่วน

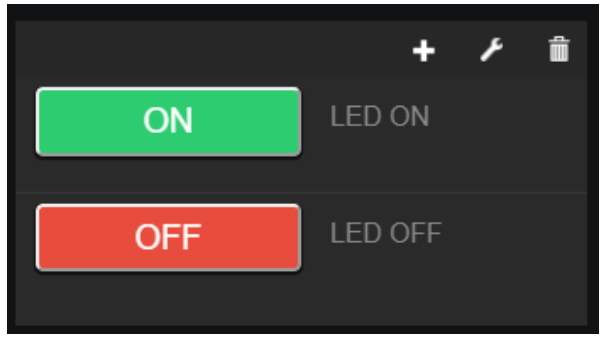
1. เขียน Code บน ESP32 เพื่อ Subscribe และสั่งเปิด/ปิด LED
2. เพิ่ม Device Schema สำหรับข้อมูล LED
3. สร้าง Widget สำหรับควบคุม LED



ESP32

Device Schema





ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

1 ส่วนที่ 1 การเรียกใช้ Library และ ประกาศตัวแปร

ประกาศตัวแปรและขาใช้สำหรับ LED

ประกาศตัวแปรและขาใช้สำหรับการทำ millis เนื่องจากมีการรับส่งข้อมูลอยู่ตลอดการใช้ delay จะไม่เหมาะสม

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.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];
#define DHTPIN 34
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

#define LED1 5

long lastMsg = 0;
int value = 0;
```

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

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

คำสั่ง Subscribe Topic
สำหรับรอรับคำสั่งจาก Freeboard

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

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);

  if (String(topic) == "@msg/led") {
    if (message == "on") {
      digitalWrite(LED1, 0);
      client.publish("@shadow/data/update", "{\"data\" : {\"led\" : \"on\"}}");
      Serial.println("LED ON");
    }
    else if (message == "off") {
      digitalWrite(LED1, 1);
      client.publish("@shadow/data/update", "{\"data\" : {\"led\" : \"off\"}}");
      Serial.println("LED OFF");
    }
  }
}
```

การเขียนเงื่อนไขเพื่อควบคุมการเปิด/ปิด LED
โดยเริ่มจากการเช็ค Topic ก่อนเสมอ
หลังจากนั้นทำการเช็คข้อความที่ได้รับ
เมื่อได้รับข้อความตามที่กำหนดจะทำการ Publish ข้อมูล
กลับไปที่ NETPIE เพื่อทำการอัพเดทสถานะของ LED

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการตั้งค่าต่างๆ

ประกาศการใช้งานฟังก์ชัน callback

กำหนดประเภทขาของ LED1 และสั่งให้ LED1 ดับ

```
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);
  pinMode(ldr, INPUT);
  dht.begin();
  pinMode(LED1, OUTPUT);
  digitalWrite(LED1, 1);
}
```

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

3 ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

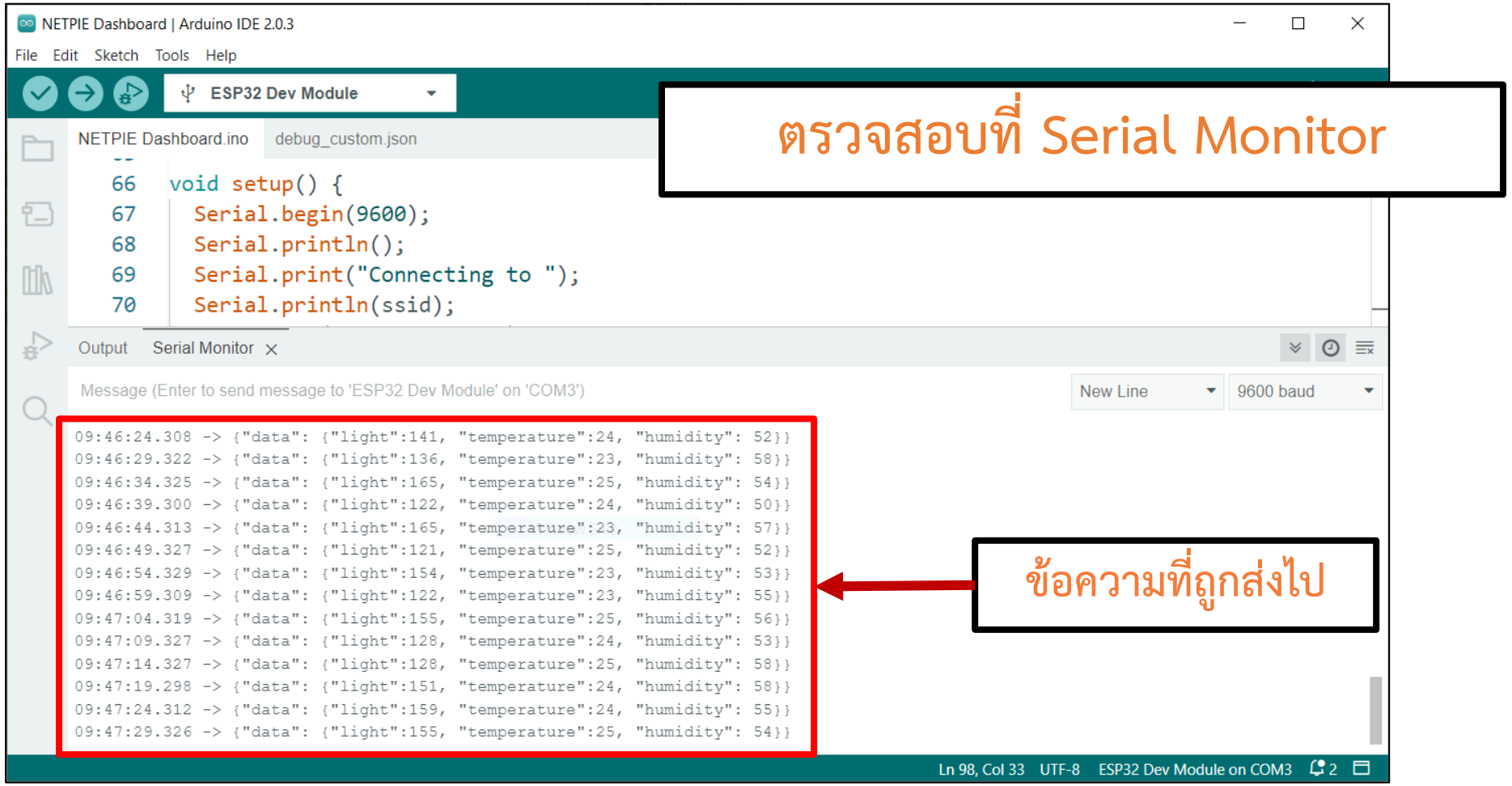
ฟังก์ชันการทำงานหลัก

ใน void loop นั้นส่วนของการส่งข้อมูลจากเซนเซอร์ต่างๆได้นำ millis มาครอบการทำงานทั้งหมดเพื่อไม่ให้เกิดการหน่วงเวลาการทำงานของโปรแกรม

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

  long now = millis();
  if (now - lastMsg > 5000) {
    float ldr_data = analogRead(ldr);
    int lux = int((250.000000 / (ADC_value * ldr_data)) - 50.000000);

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

    lastMsg = now;
    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(1);
}
```

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

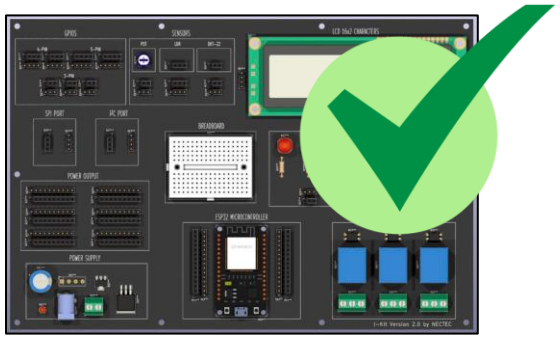


ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

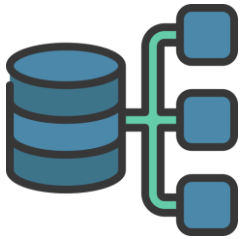
เช็คความพร้อมเพื่อทดสอบทั้งหมด 3 ส่วน

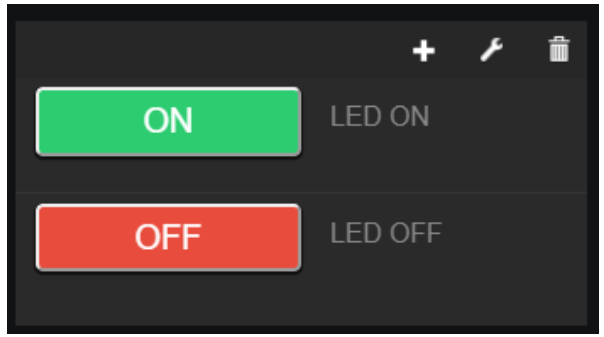
1. เขียน Code บน ESP32 เพื่อ Subscribe และสั่งเปิด/ปิด LED
2. เพิ่ม Device Schema สำหรับข้อมูล LED
3. สร้าง Widget สำหรับควบคุม LED



ESP32

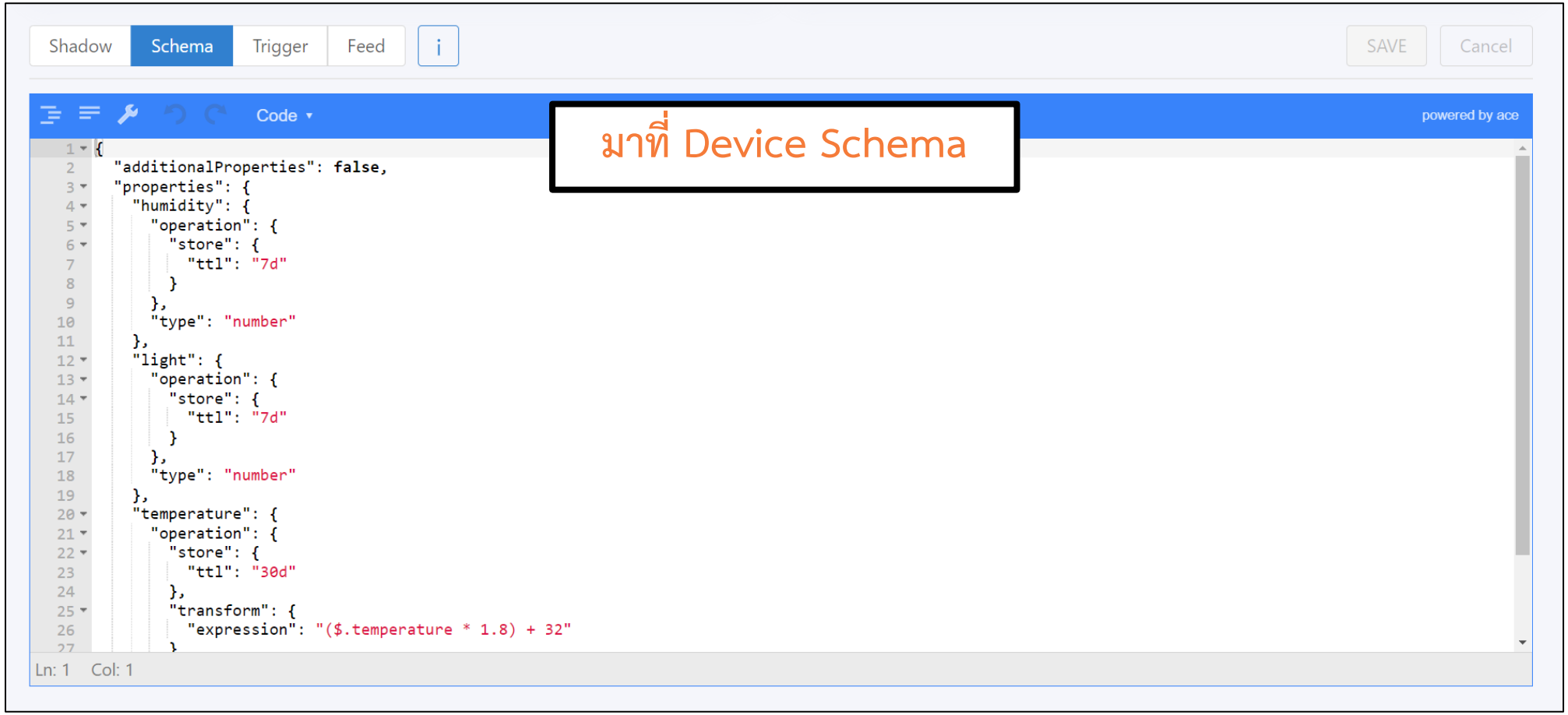
Device Schema





ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button



ใบงานที่ 5.2 ขั้นตอนการทดลอง

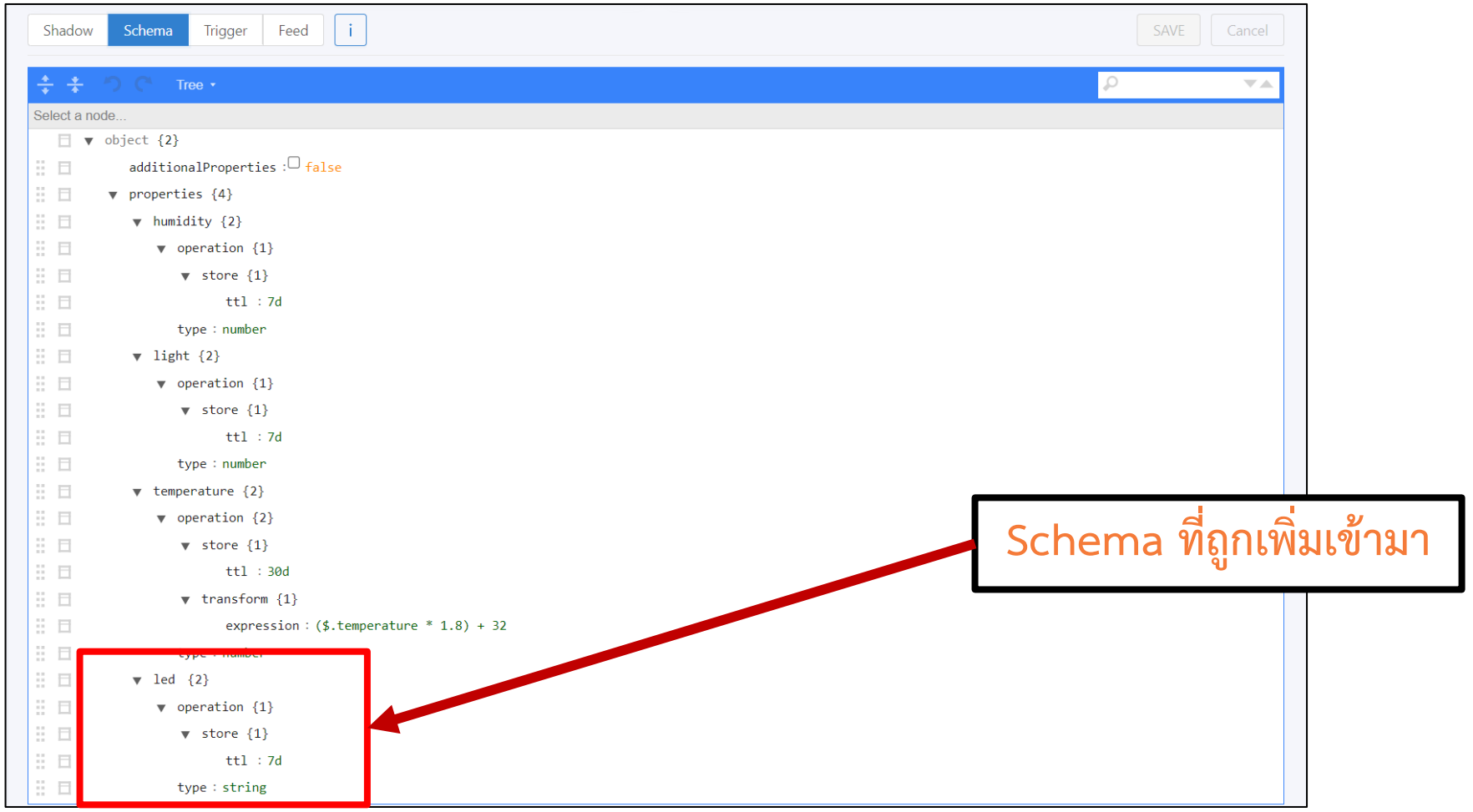
การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

```
{
  "additionalProperties": false,
  "properties": {
    "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"
    },
    "led": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "string"
    }
  }
```

ส่วนของ led ที่ต้องเพิ่มใน Device Schema
สามารถ Copy ทั้งหมดเพื่อนำไป Paste แทนบน Device Schema

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

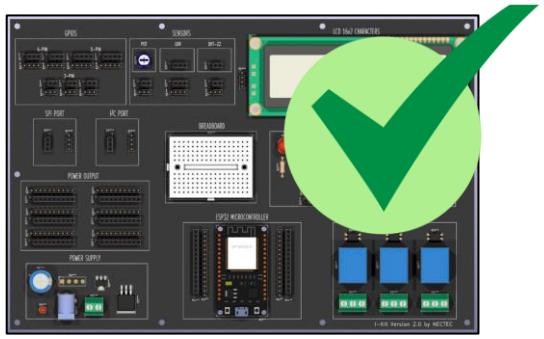


ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

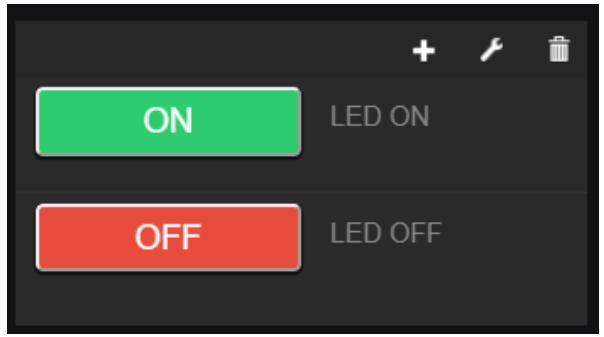
เช็คความพร้อมเพื่อทดสอบทั้งหมด 3 ส่วน

1. เขียน Code บน ESP32 เพื่อ Subscribe และสั่งเปิด/ปิด LED
2. เพิ่ม Device Schema สำหรับข้อมูล LED
3. สร้าง Widget สำหรับควบคุม LED



ESP32

Device Schema



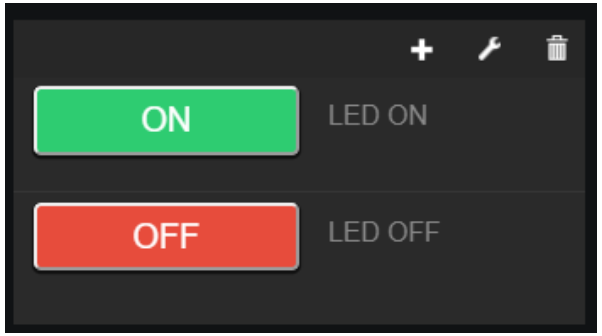
ใบงานที่ 5.2 ขั้นตอนการทดลอง

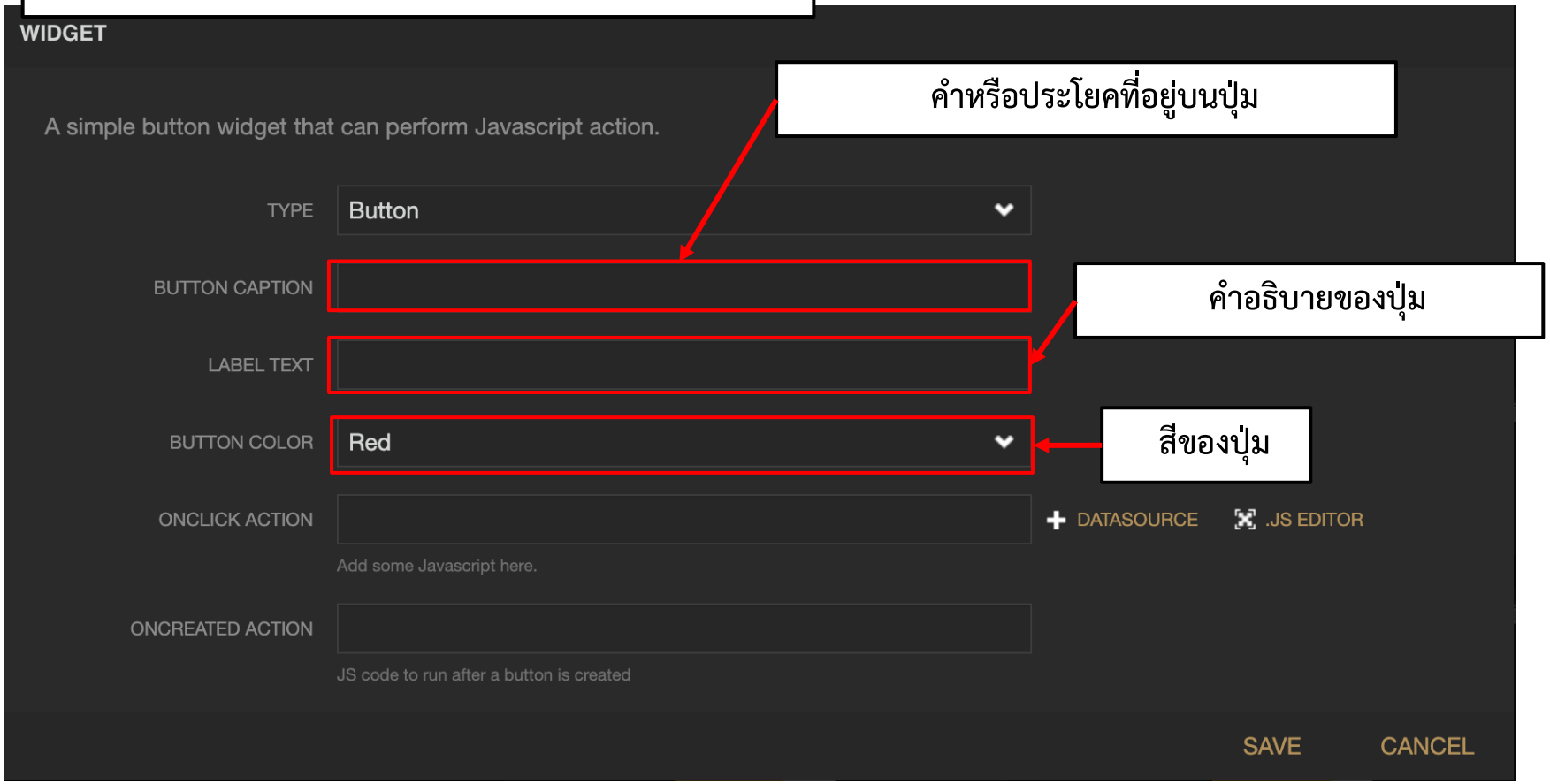
การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

การสร้าง Widget : Button

เป็น Widget สำหรับควบคุมแบบปุ่มกด

ตัวอย่าง Widget : Button





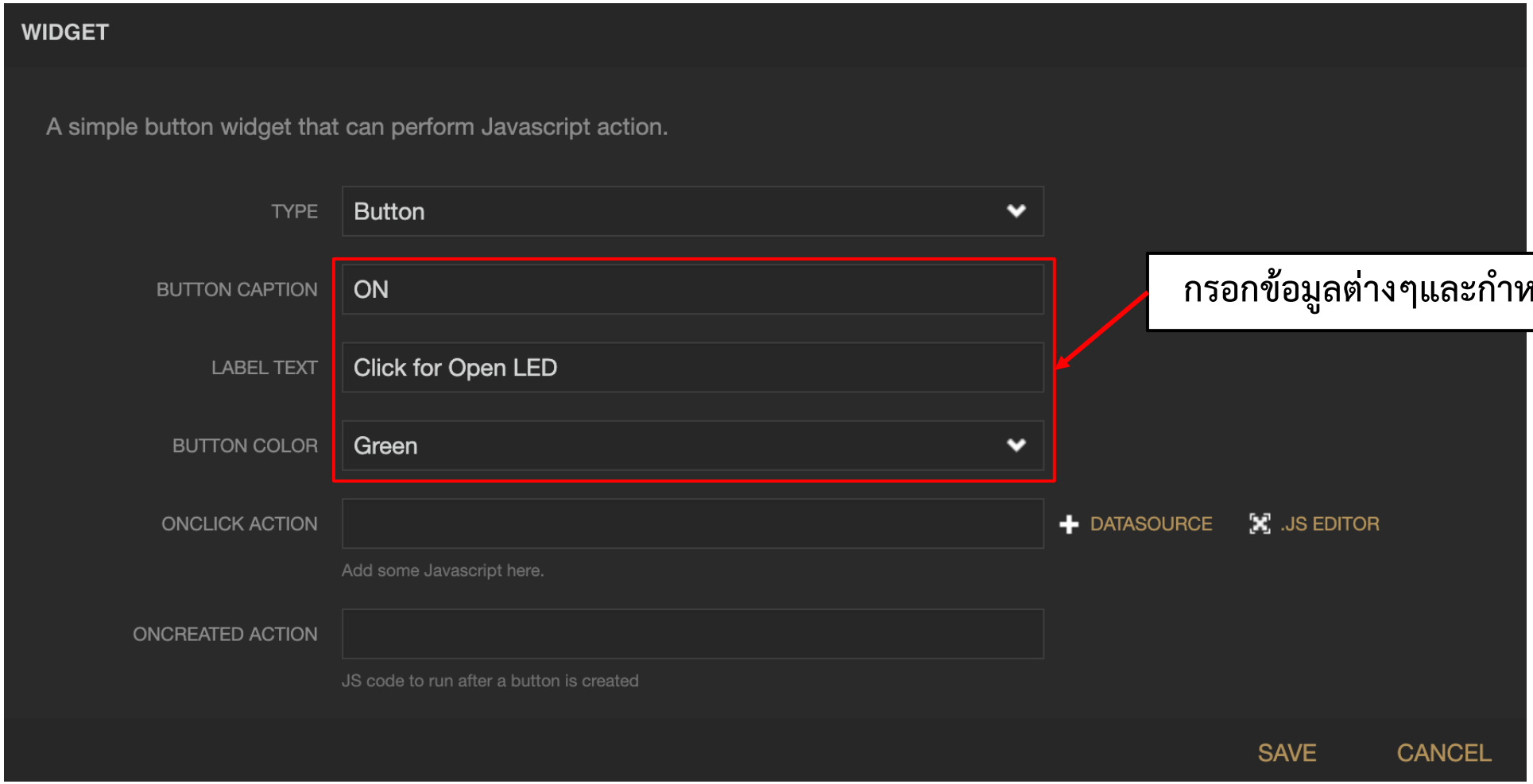
คำหรือประโยคที่อยู่บนปุ่ม

คำอธิบายของปุ่ม

สีของปุ่ม

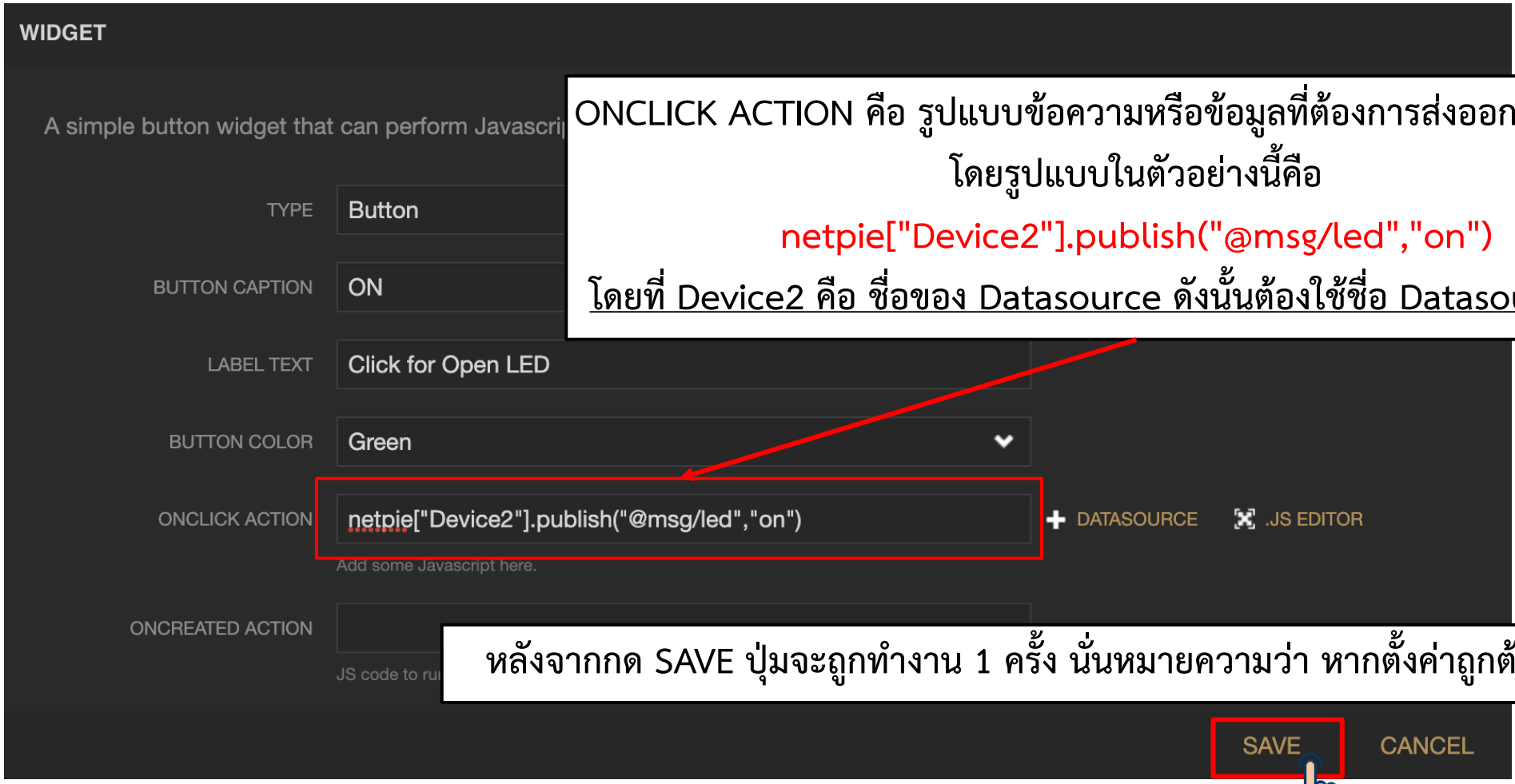
ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button



ONCLICK ACTION คือ รูปแบบข้อความหรือข้อมูลที่ต้องการส่งออกเมื่อคลิกที่ปุ่ม โดยรูปแบบในตัวอย่างนี้คือ

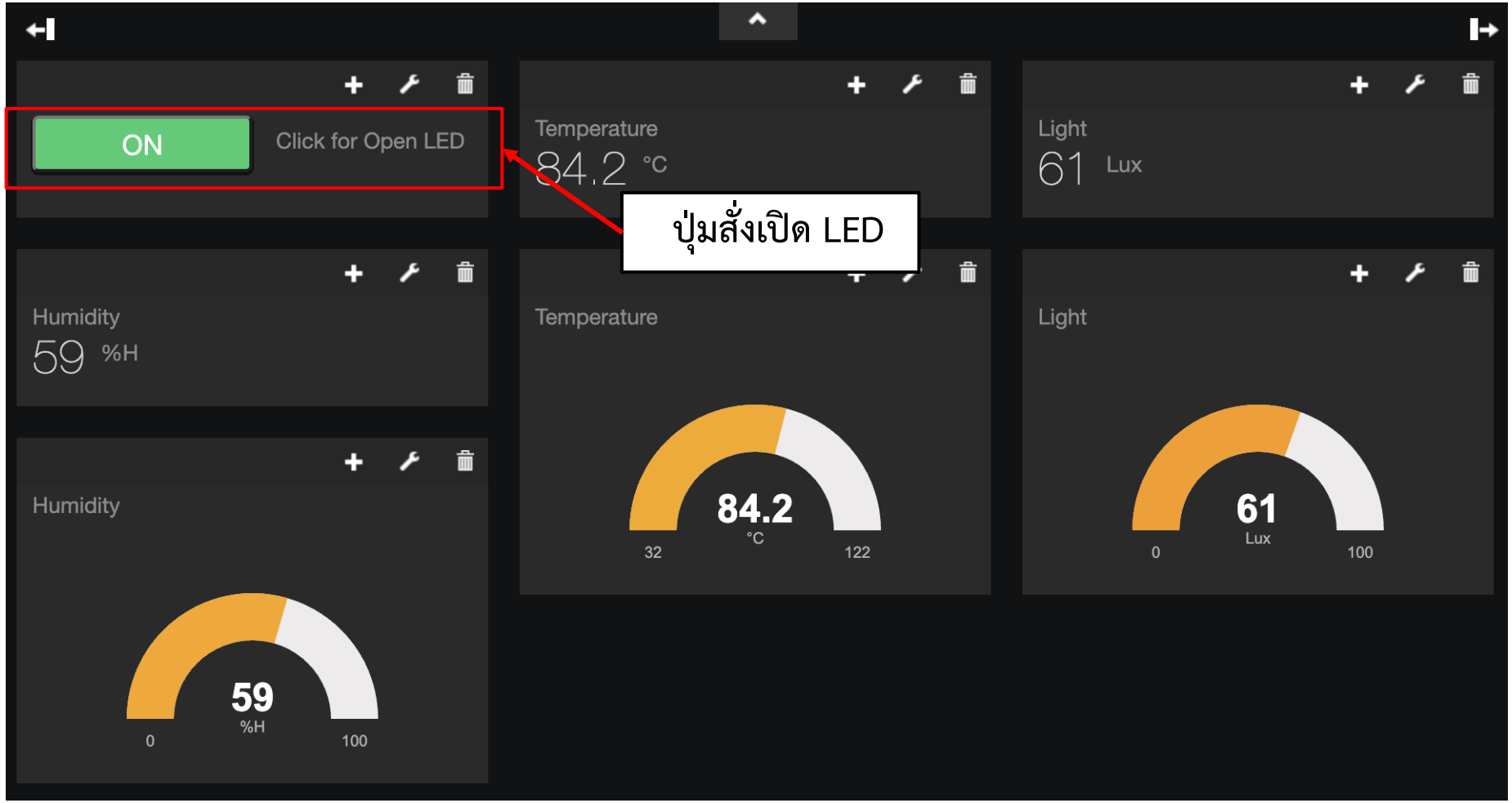
netpie["Device2"].publish("@msg/led","on")

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

หลังจากกด SAVE ปุ่มจะถูกทำงาน 1 ครั้ง นั่นหมายความว่า หากตั้งค่าถูกต้อง LED จะติด

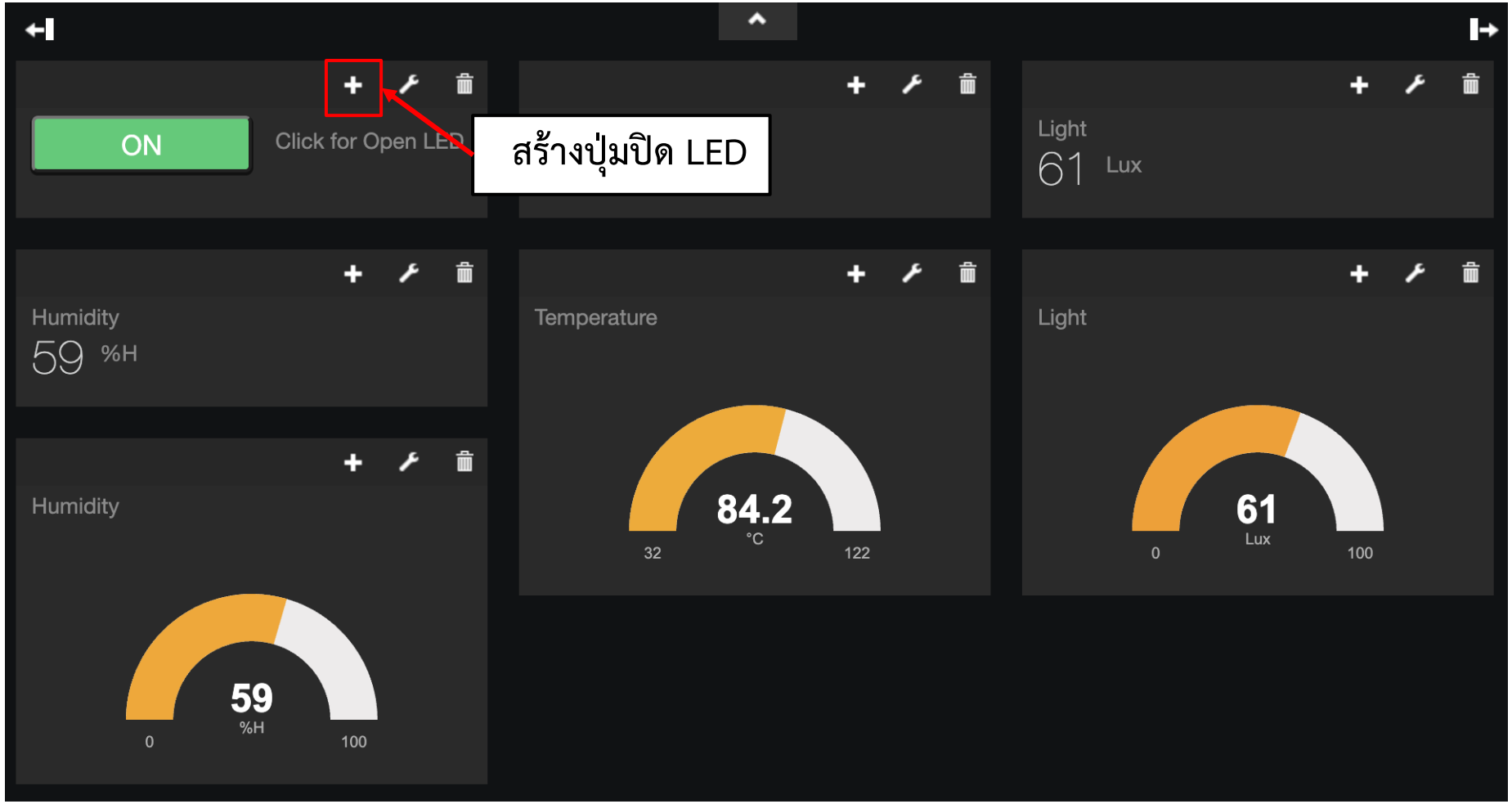
ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button



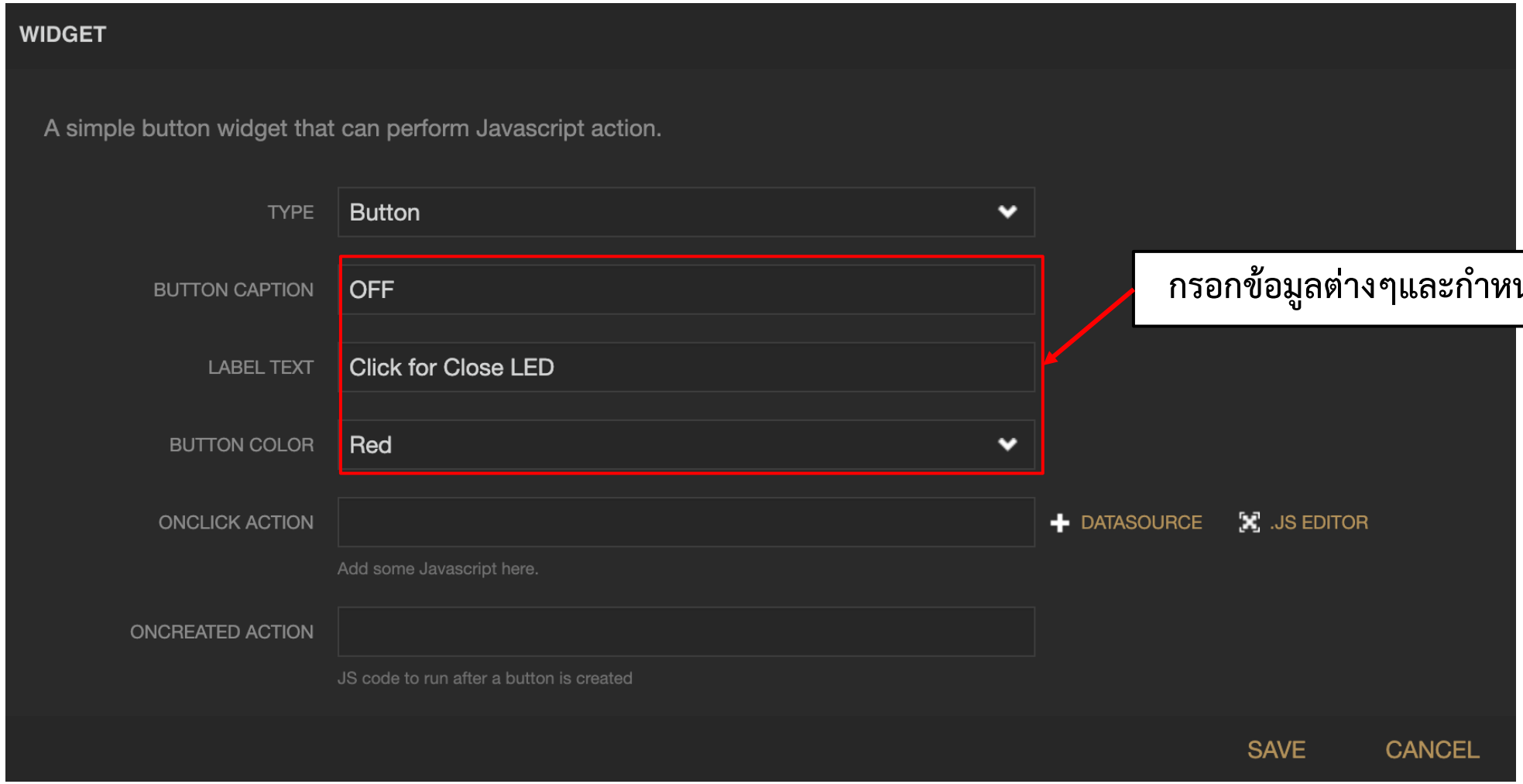
ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button



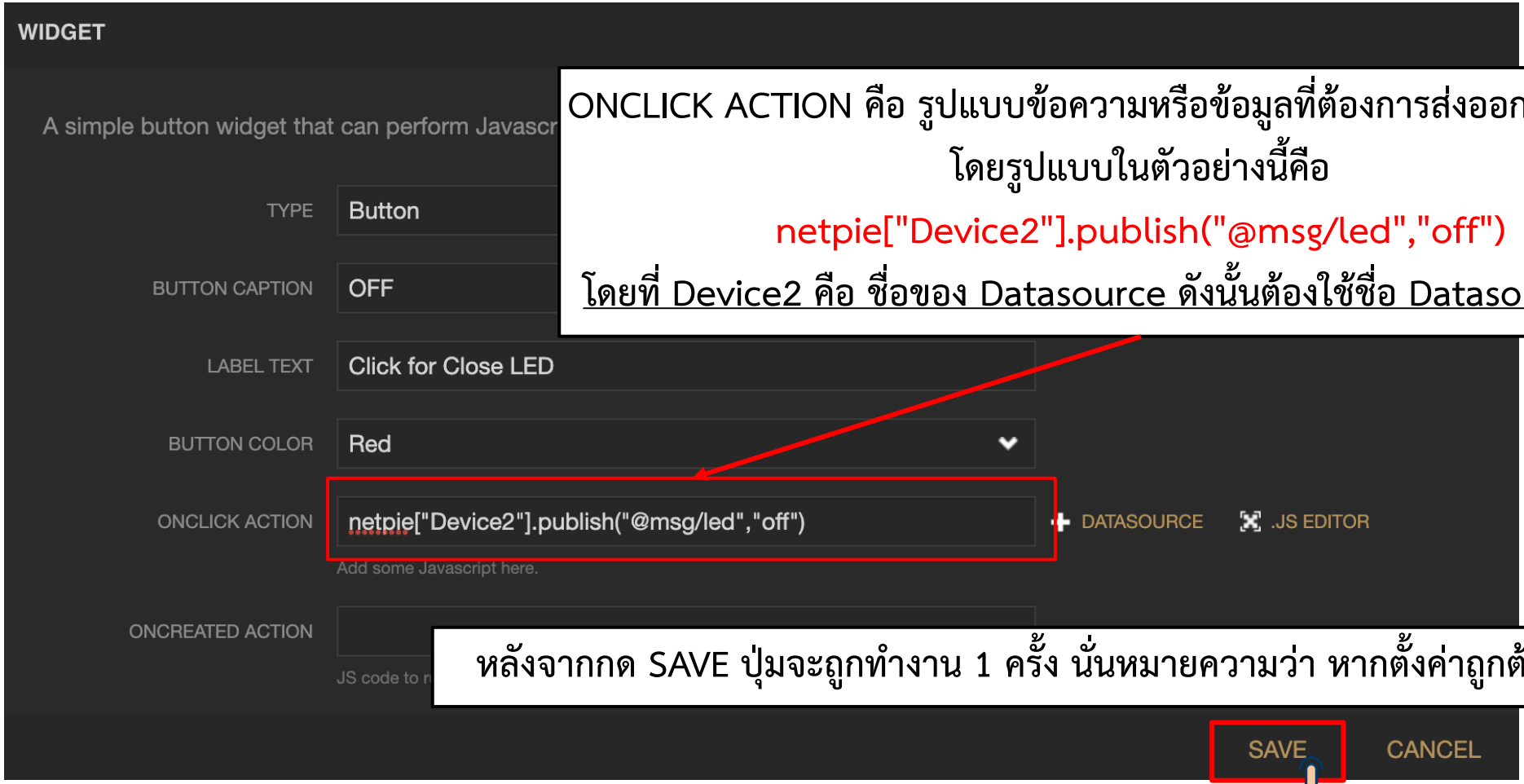
ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button



ONCLICK ACTION คือ รูปแบบข้อความหรือข้อมูลที่ต้องการส่งออกเมื่อคลิกที่ปุ่ม โดยรูปแบบในตัวอย่างนี้คือ

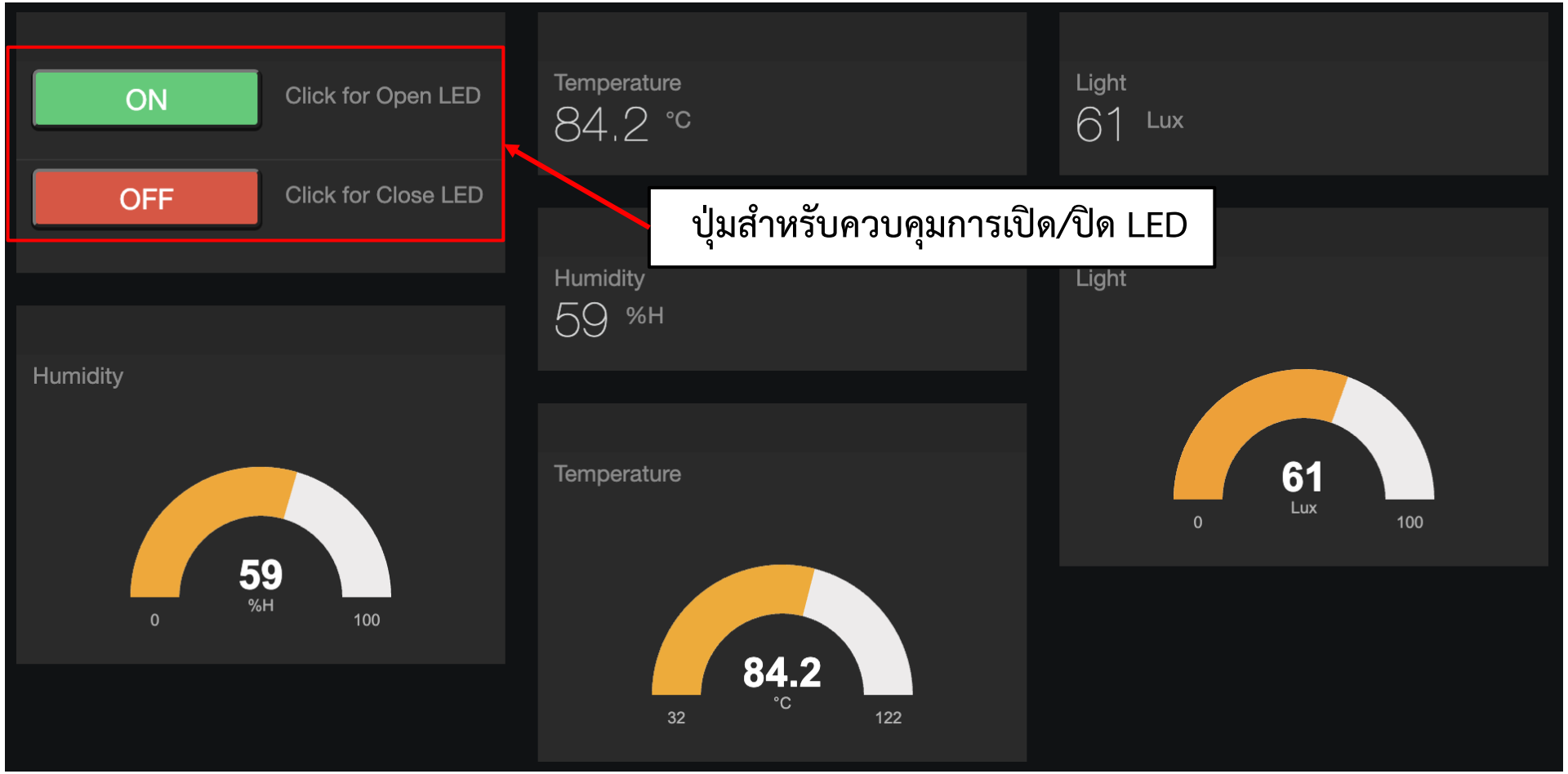
netpie["Device2"].publish("@msg/led","off")

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

หลังจากกด SAVE ปุ่มจะถูกทำงาน 1 ครั้ง นั่นหมายความว่า หากตั้งค่าถูกต้อง LED จะดับ

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button



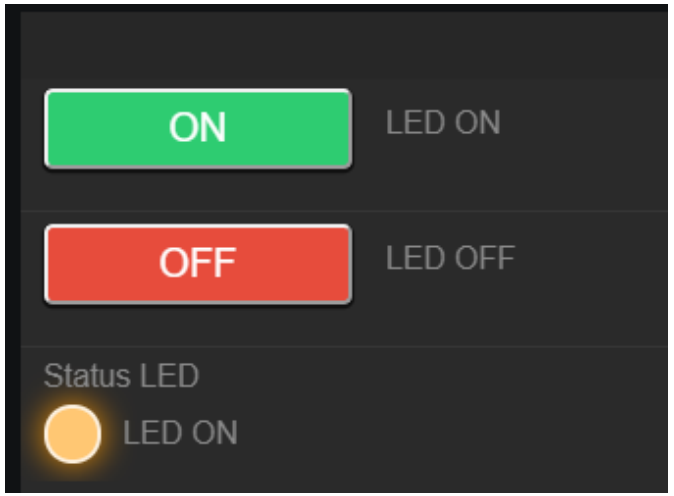
ใบงานที่ 5.2 ขั้นตอนการทดลอง

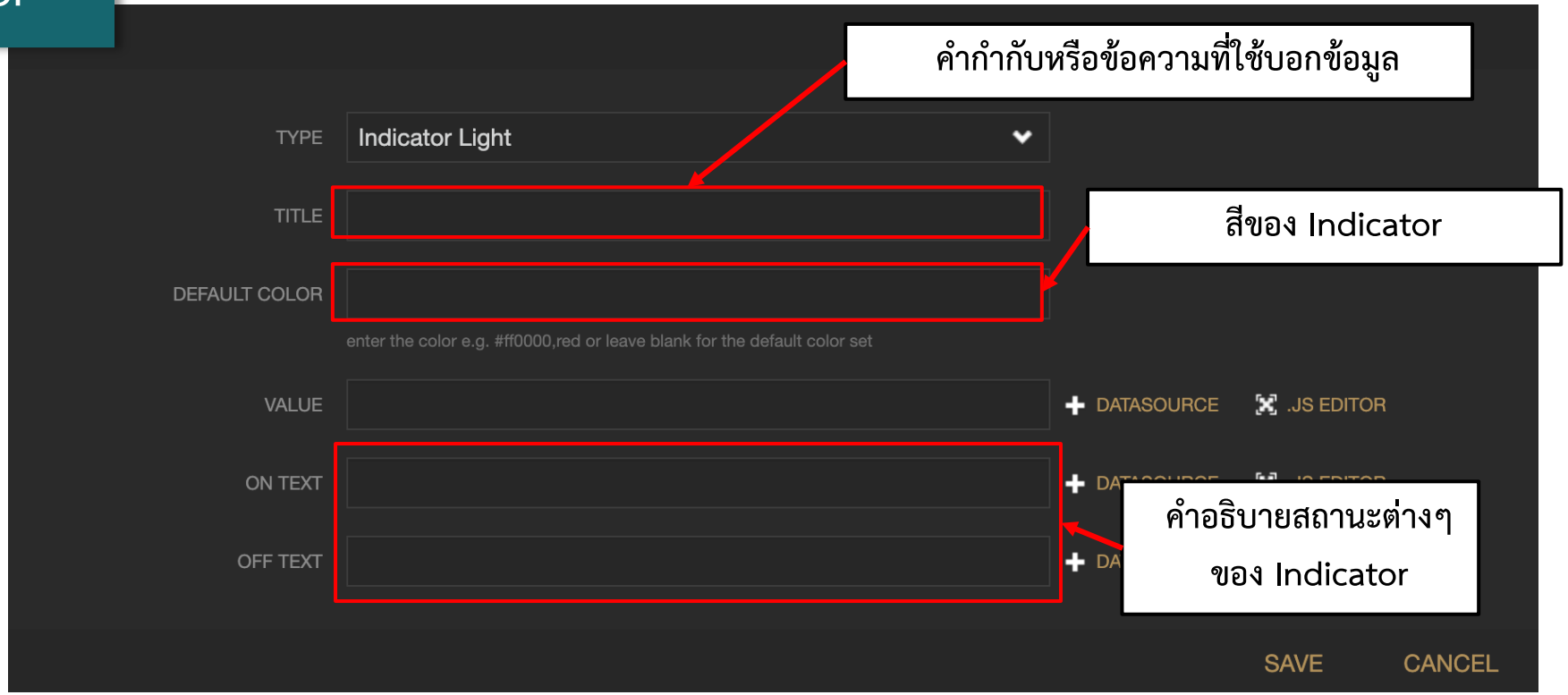
การทดลองที่ 2 สร้าง Widget แสดงสถานะของ LED ด้วย Widget Indicator

เป็น Widget สำหรับแสดงผลสถานะการทำงาน

การสร้าง Widget : Indicator

ตัวอย่าง Widget : Indicator



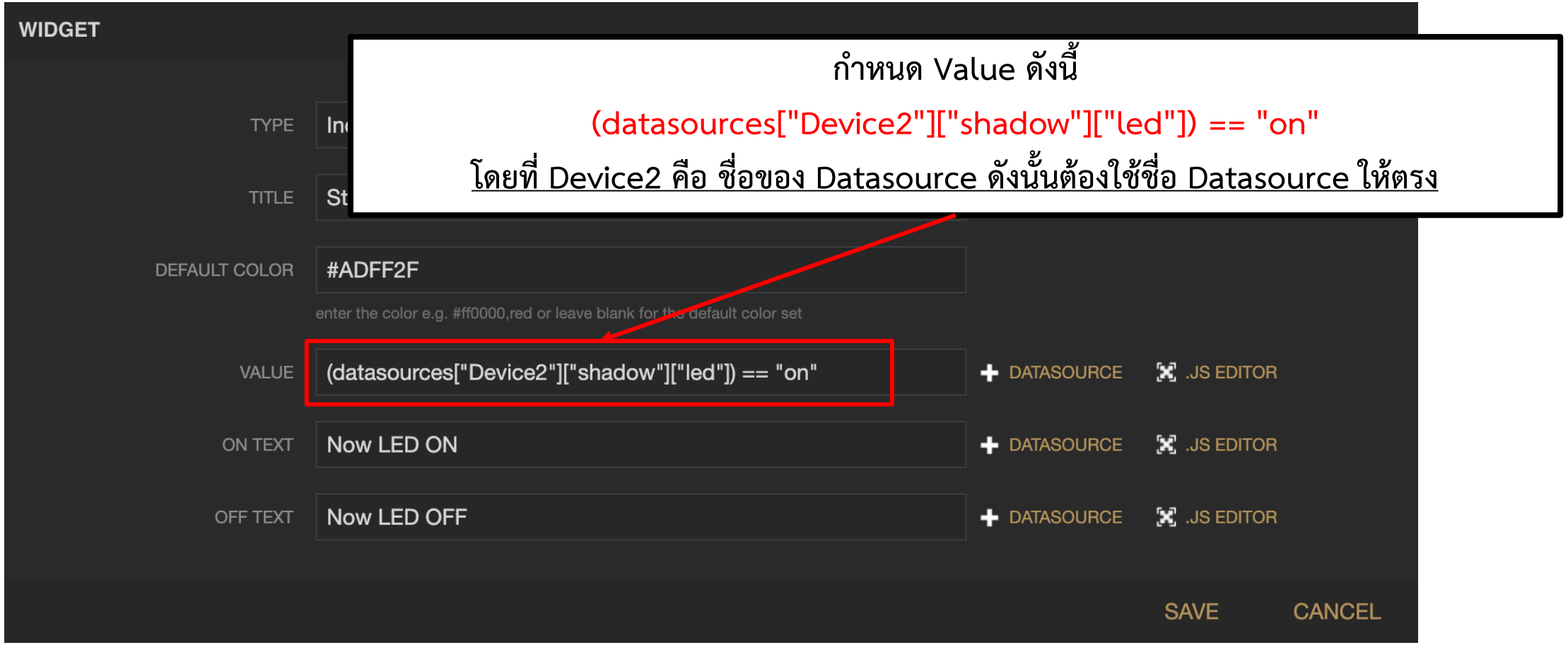


ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 2 สร้าง Widget แสดงสถานะของ LED ด้วย Widget Indicator

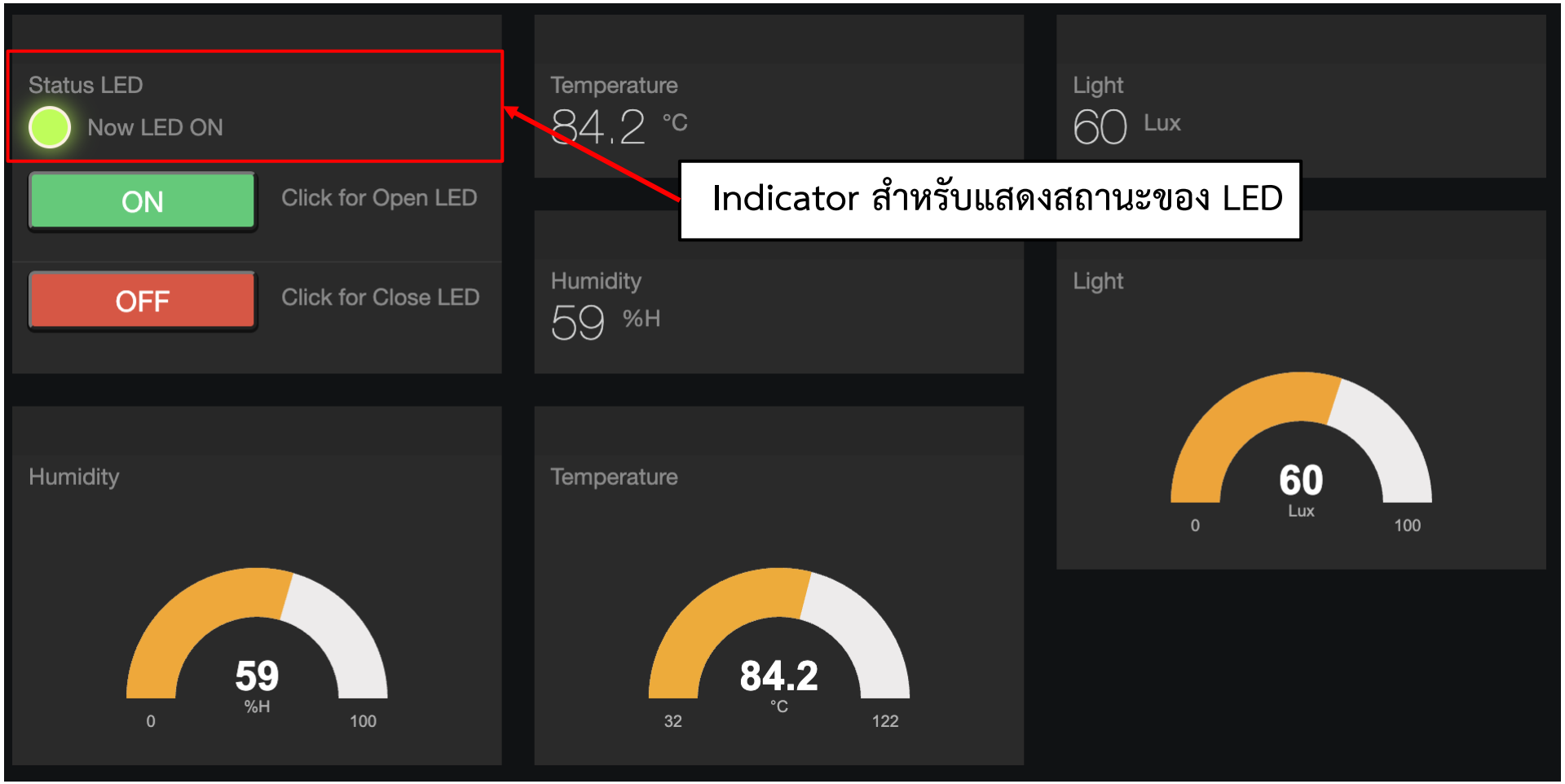
ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 2 สร้าง Widget แสดงสถานะของ LED ด้วย Widget Indicator



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 2 สร้าง Widget แสดงสถานะของ LED ด้วย Widget Indicator

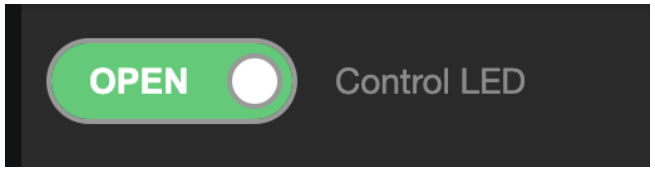


ใบงานที่ 5.2 ขั้นตอนการทดลอง

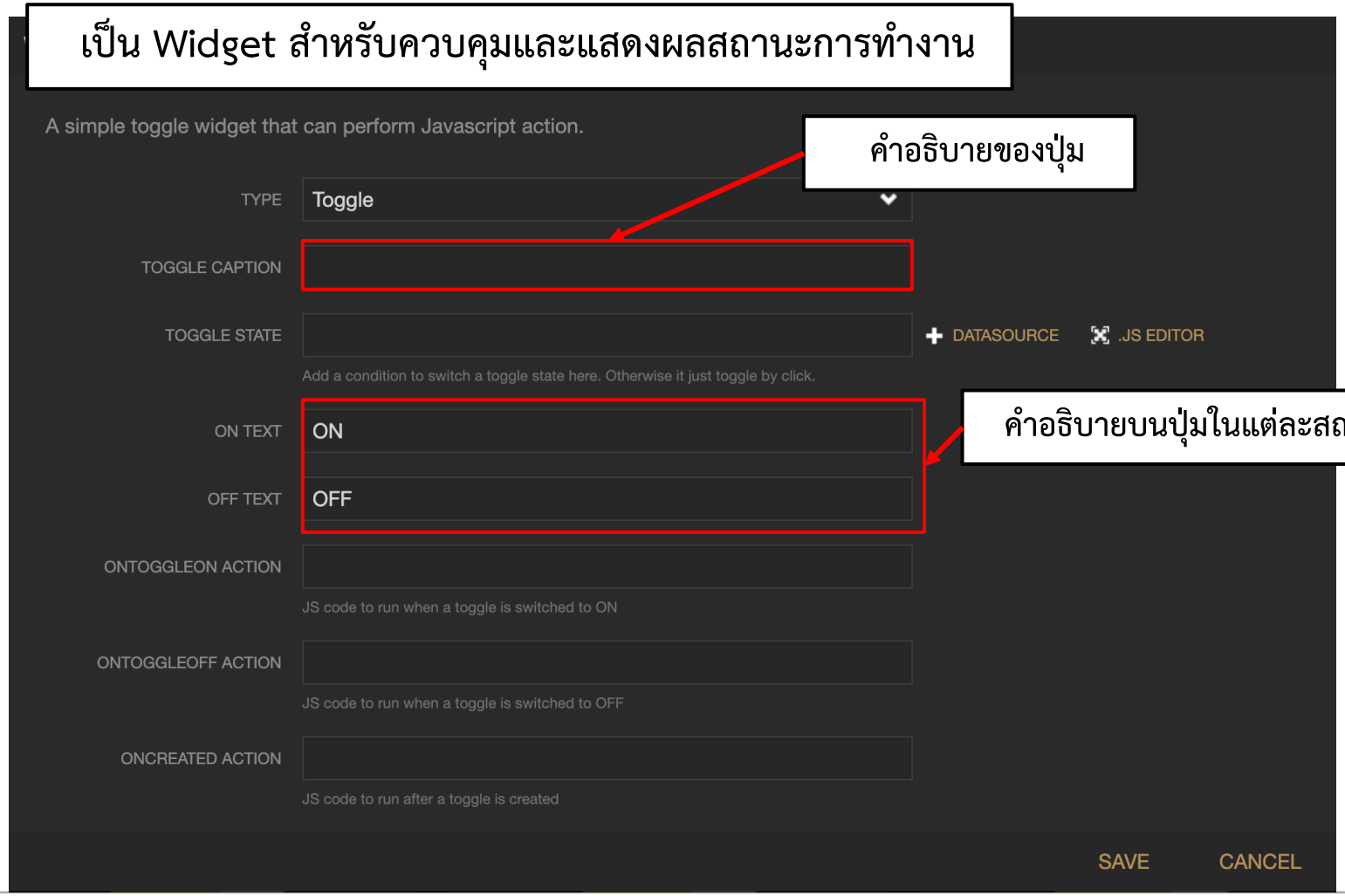
การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle

การสร้าง Widget : Toggle

ตัวอย่าง Widget : Toggle

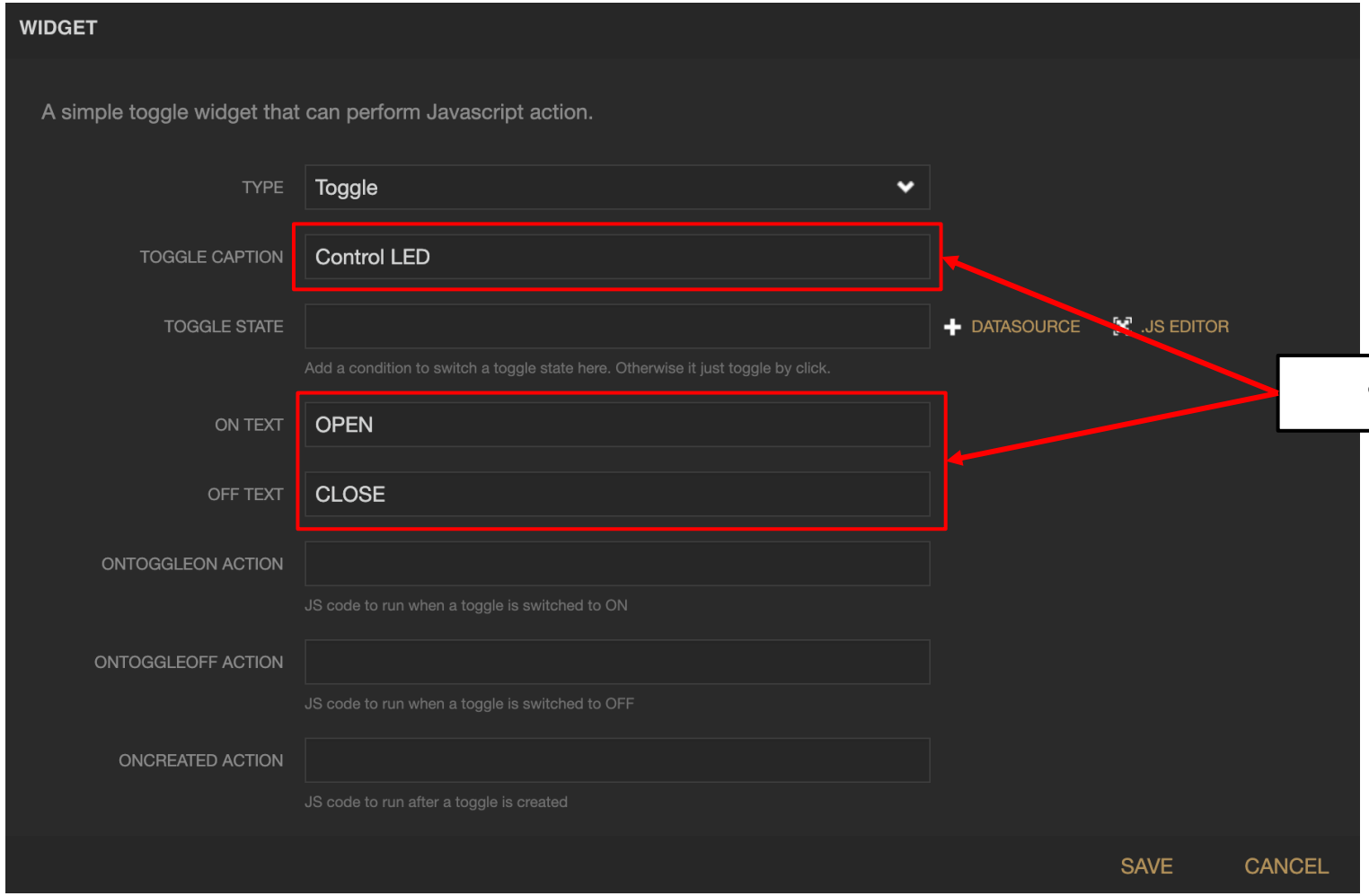


เป็น Widget สำหรับควบคุมและแสดงผลสถานะการทำงาน



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle

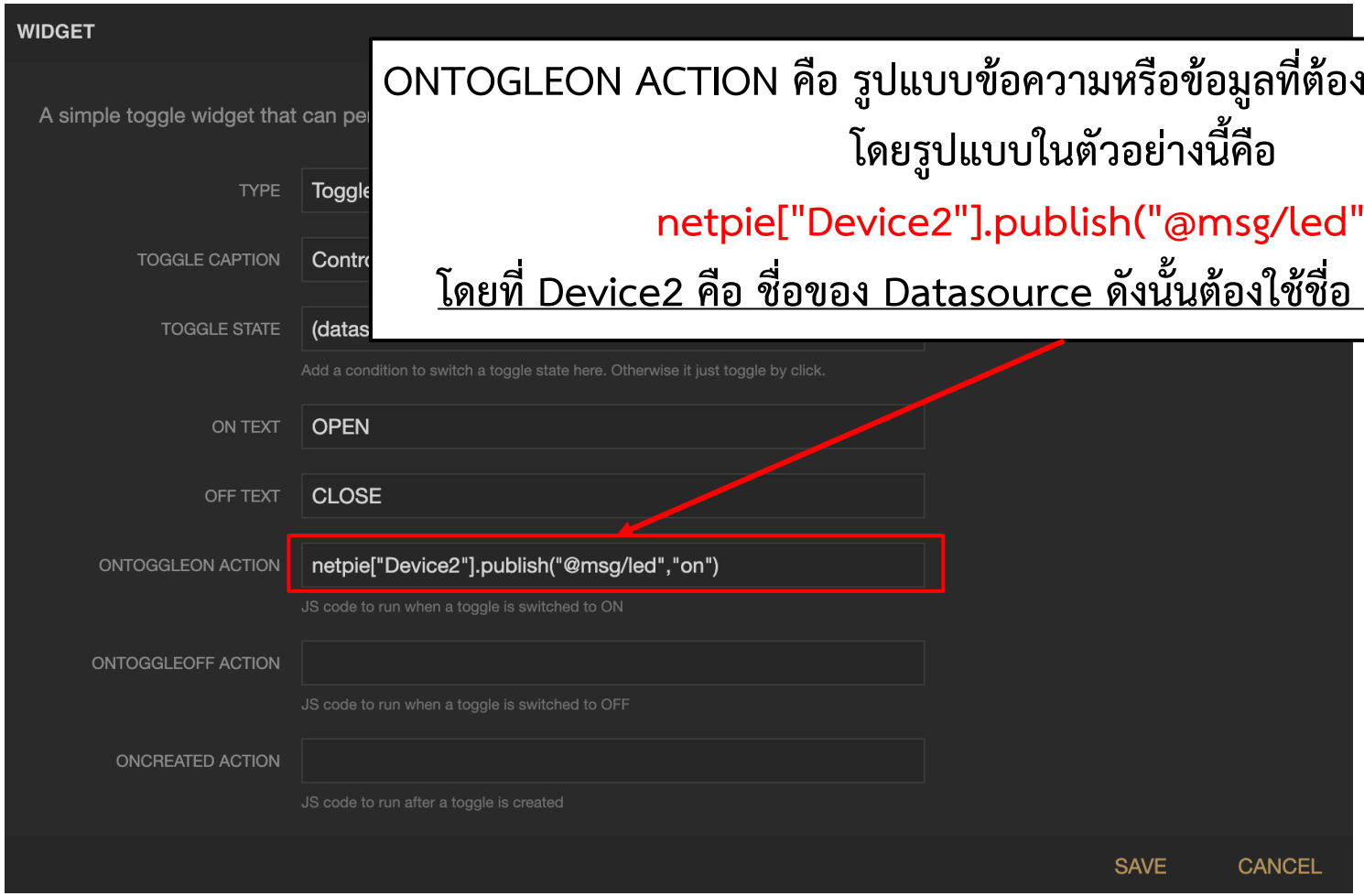
กำหนด Toggle State ดังนี้

(datasources["Device2"]["shadow"]["led"]) == "on"

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle



ONTOGLEON ACTION คือ รูปแบบข้อความหรือข้อมูลที่ต้องการส่งออกเมื่อคลิกที่ปุ่ม โดยรูปแบบในตัวอย่างนี้คือ

netpie["Device2"].publish("@msg/led","on")

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle

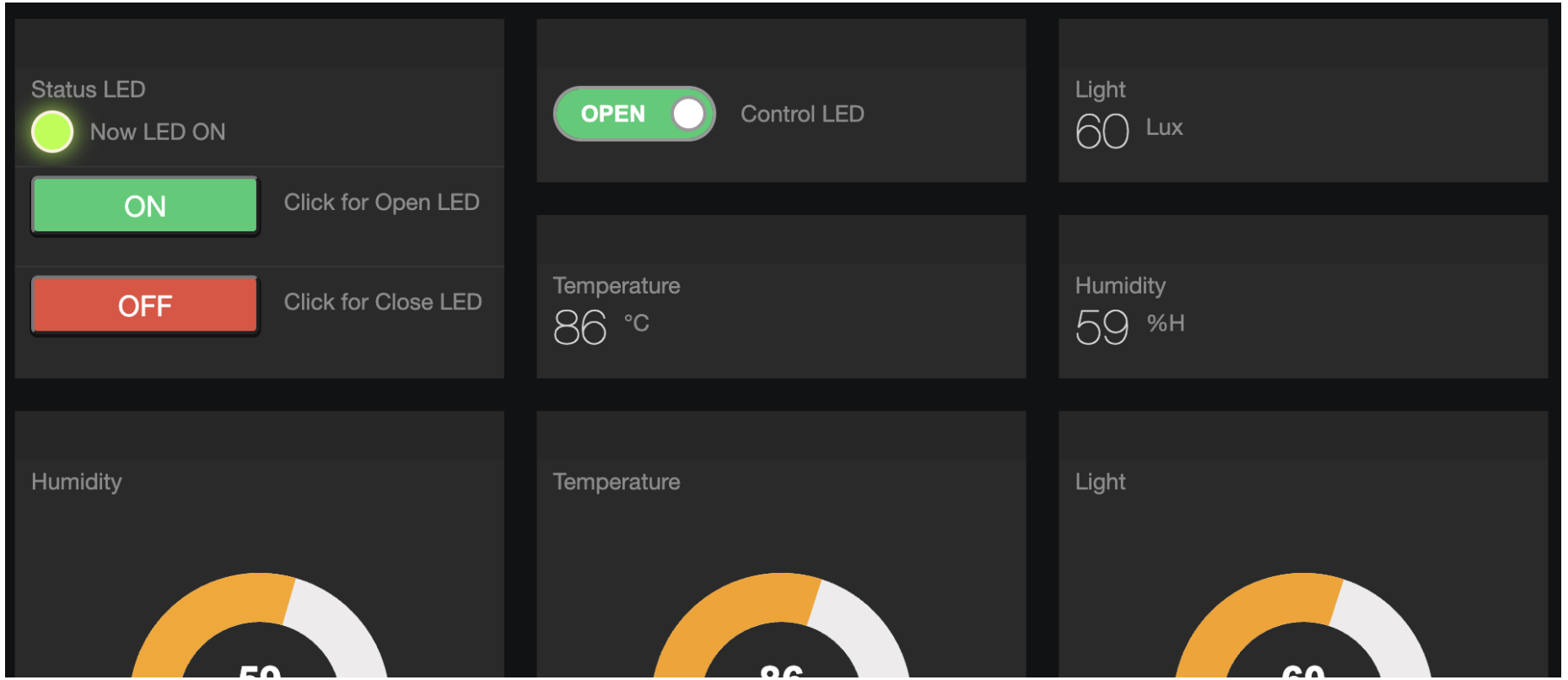
ONTOGGLEOFF ACTION คือ รูปแบบข้อความหรือข้อมูลที่ต้องการส่งออกเมื่อคลิกที่ปุ่ม โดยรูปแบบในตัวอย่างนี้คือ

netpie["Device2"].publish("@msg/led","off")

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

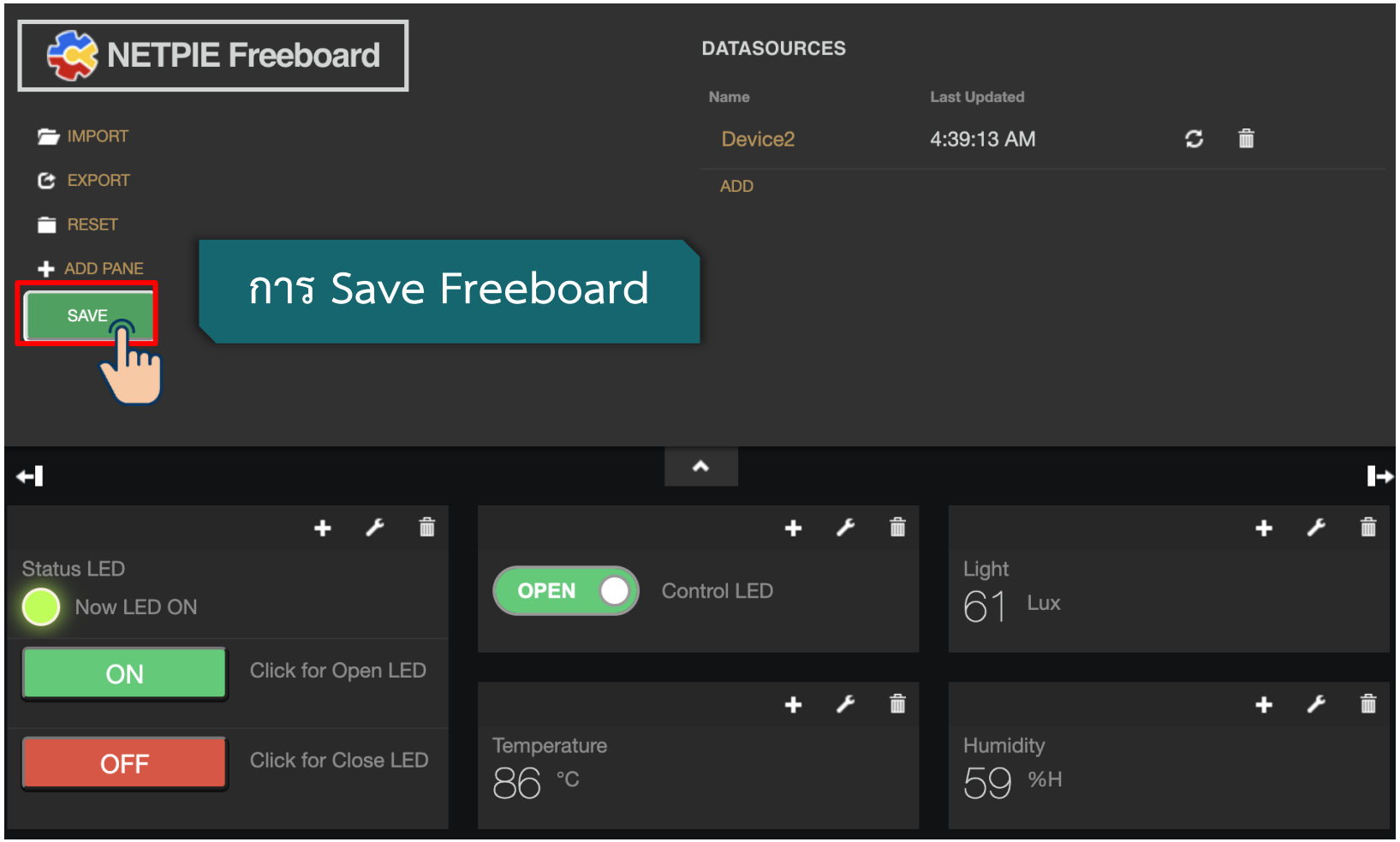
ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle



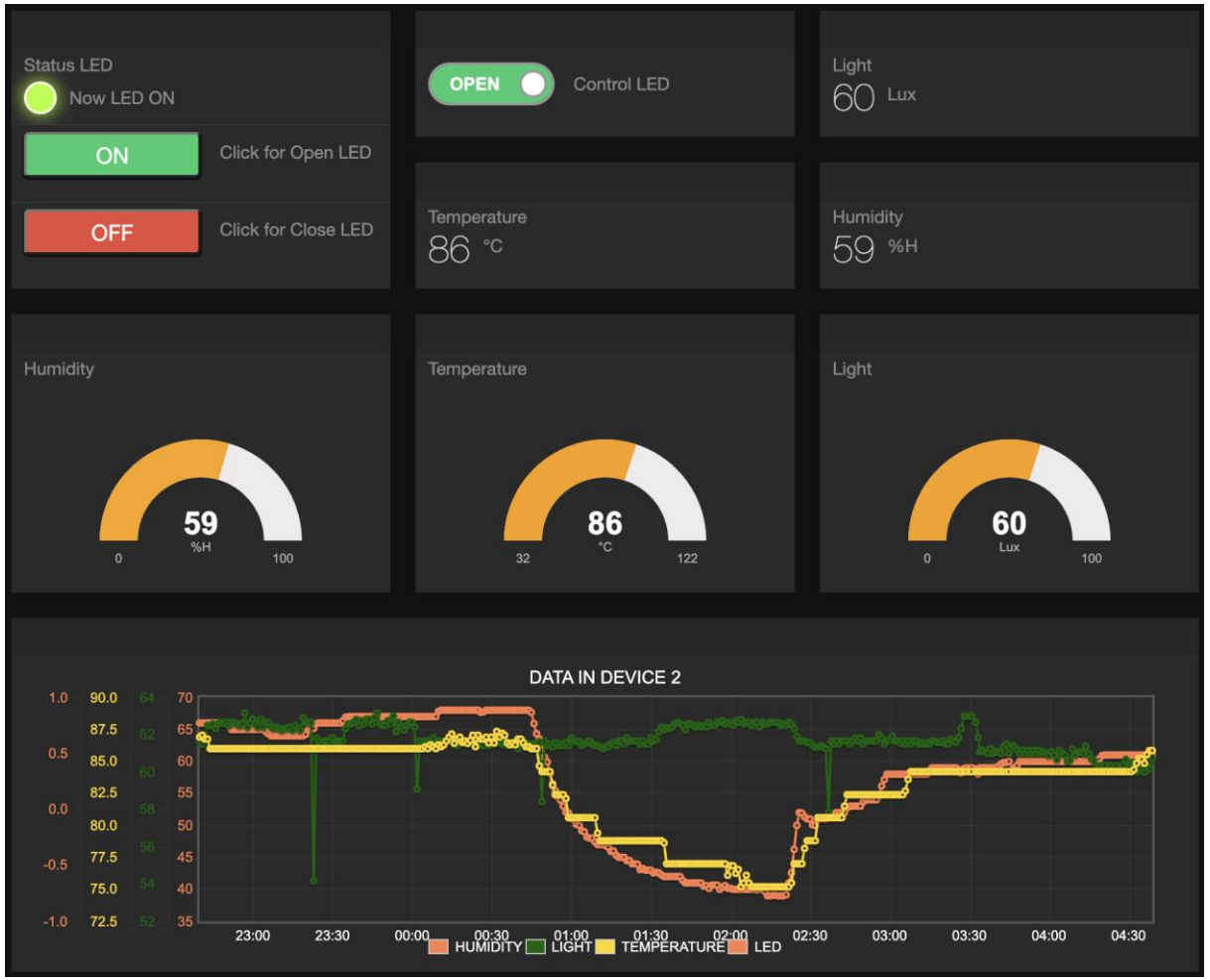
ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle



ใบงานที่ 5.2 คำถามท้ายหน่วยการเรียนรู้

คำถามท้ายหน่วยการเรียนรู้ที่ 4

จงสร้าง Dashboard สำหรับแสดงผลข้อมูลและควบคุมอุปกรณ์ต่างๆบน ESP32 ที่ถูกจัดเก็บบน Schema จาก Exercise 1 โดยกำหนดให้มี Widget ในแต่ละข้อมูลดังนี้

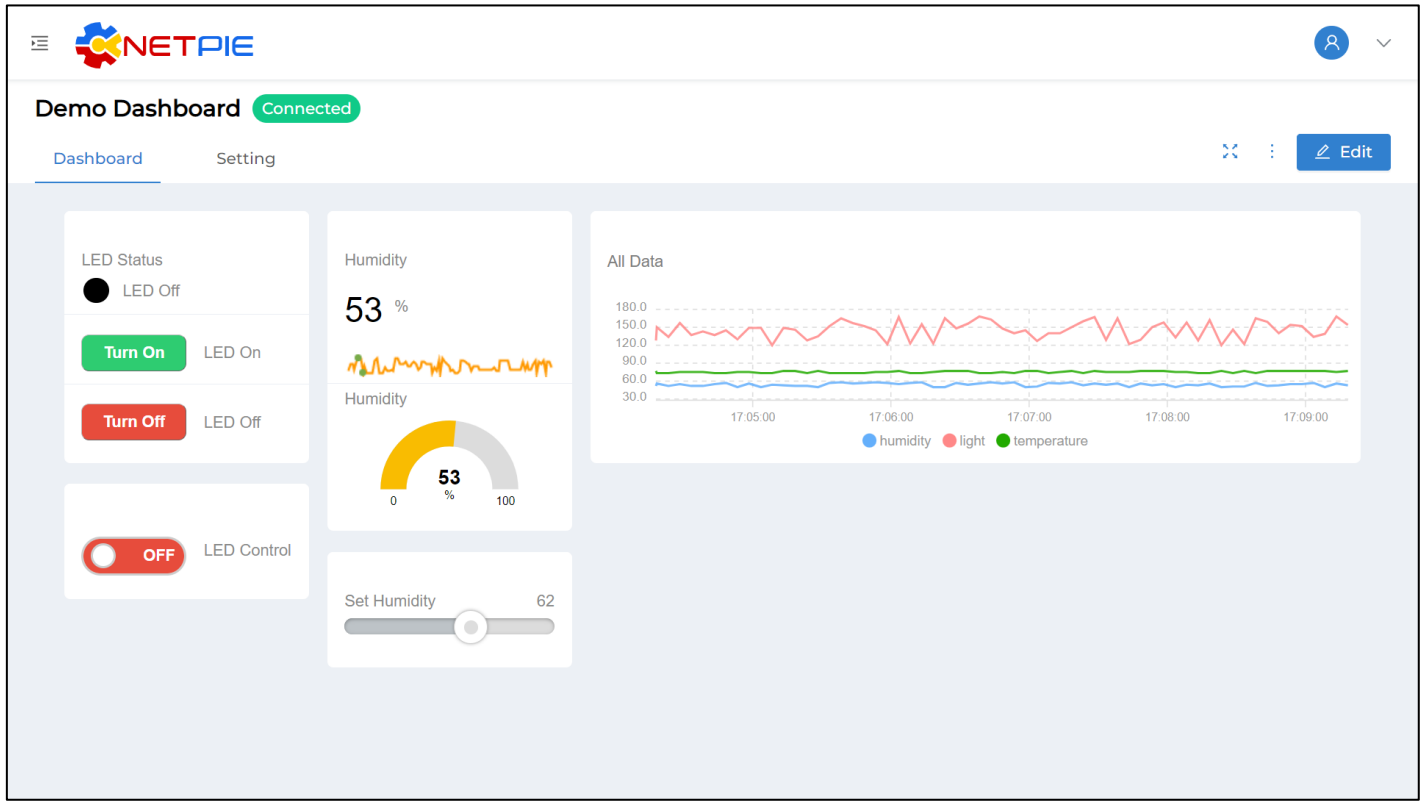
1. Widget : Text แสดงผลข้อมูล voltage และ gas
2. Widget : Gauge แสดงผลข้อมูล voltage และ gas
3. Widget : FeedView แสดงผลข้อมูล voltage และ gas

ข้อ 4 และ 5 เลือกทำเพียงข้อเดียว

4. Widget : Button และ Indicator สำหรับควบคุมและแสดงสถานะ led1, led2 และ led3
5. Widget : Toggle สำหรับควบคุมและแสดงสถานะ led1, led2 และ led3

โดยทุก Widget ของแต่ละข้อมูลกำหนดให้เป็นอิสระจากกัน (นั่นคือ ต้องมี FeedView 2 อัน)

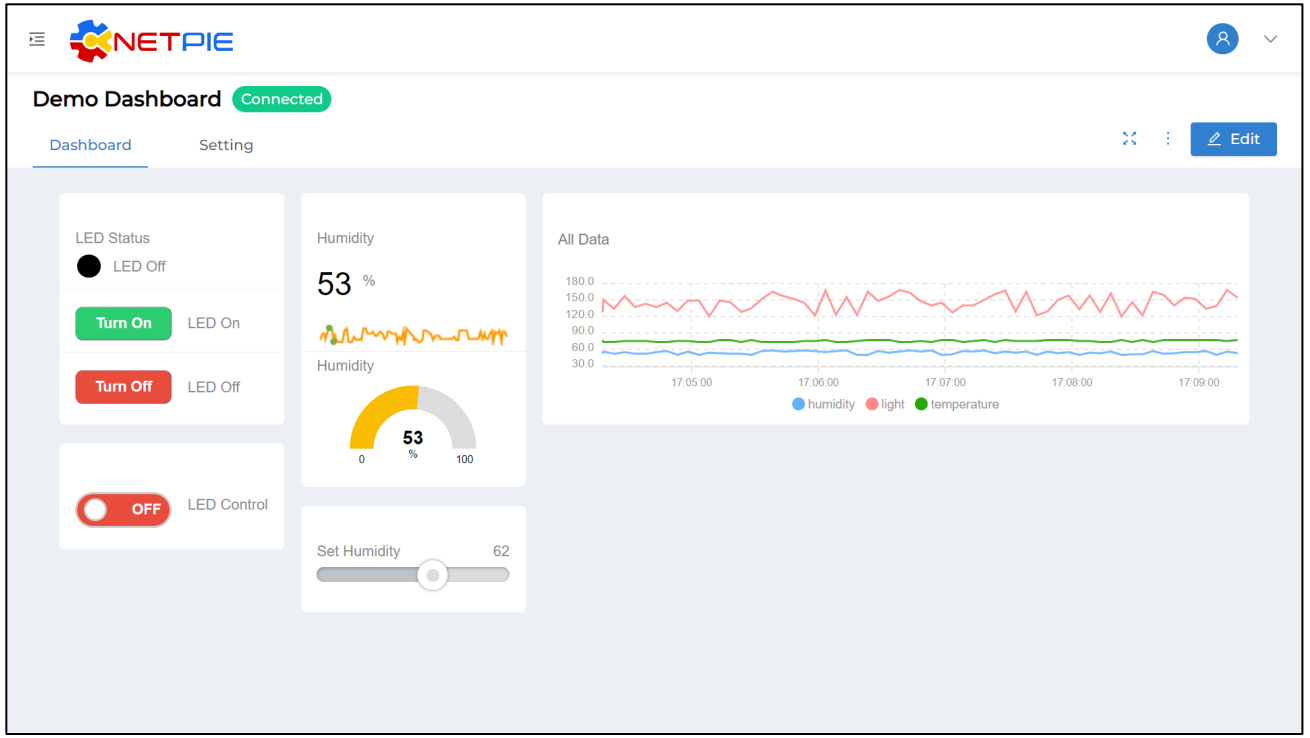
ใบงานที่ 5.3 การแสดงผลและควบคุมอุปกรณ์ด้วย Dashboard



ใบงานที่ 5.3 ทฤษฎีเบื้องต้น

Dashboard คืออะไร

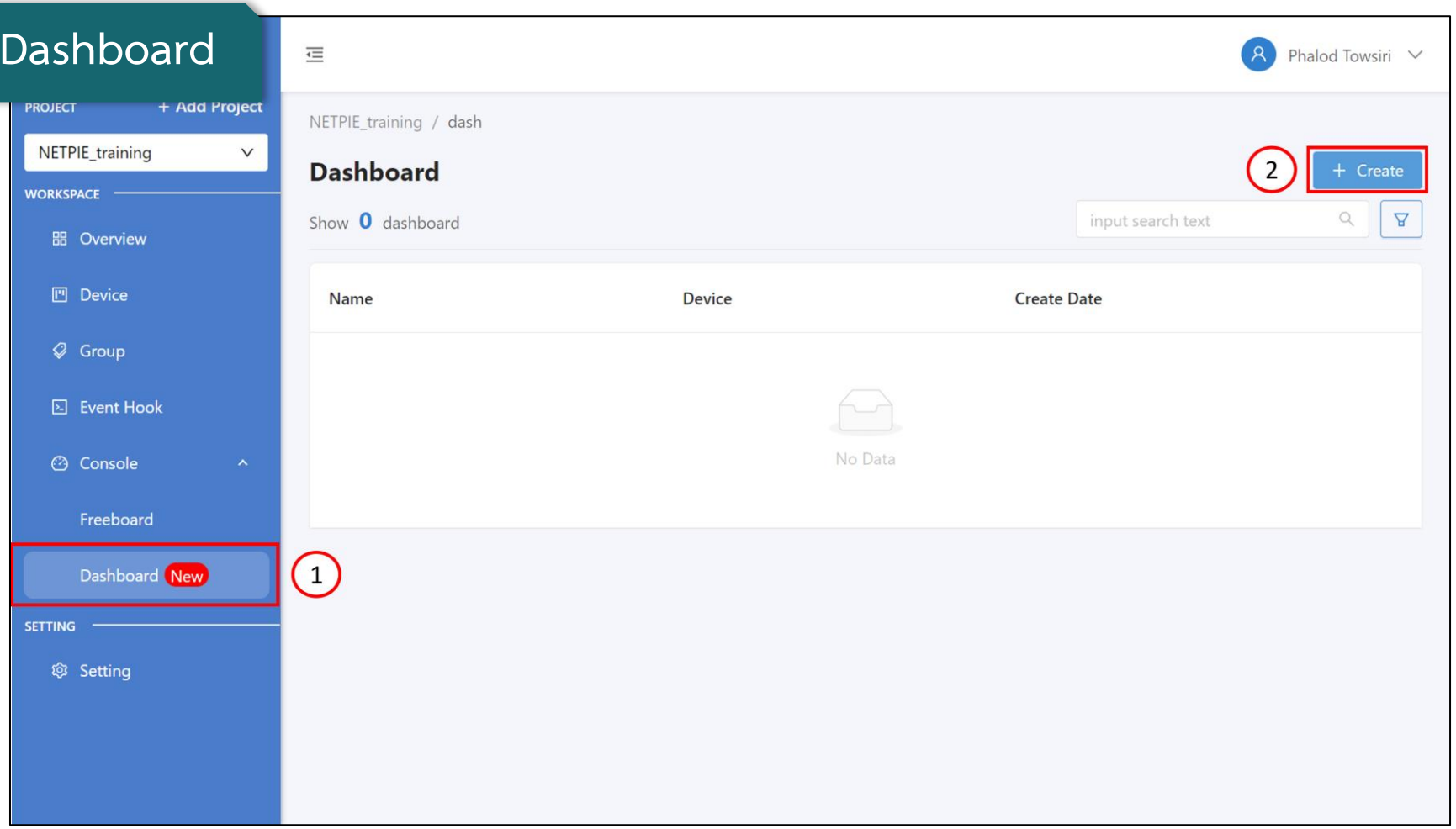
NETPIE Dashboard คือ Dashboard ที่แยกมาจาก NETPIE Freeboard โดยมีหน้าที่ที่ไม่แตกต่างกันกัน คือใช้สำหรับนำข้อมูลที่เก็บอยู่ใน Platform มาแสดงผลในรูปแบบต่างๆ เป็นเหมือนช่องทางให้ผู้ใช้สามารถติดตามหรือควบคุมการทำงานของ Device ของตัวเอง



ใบงานที่ 5.3 ขั้นตอนการทดลอง

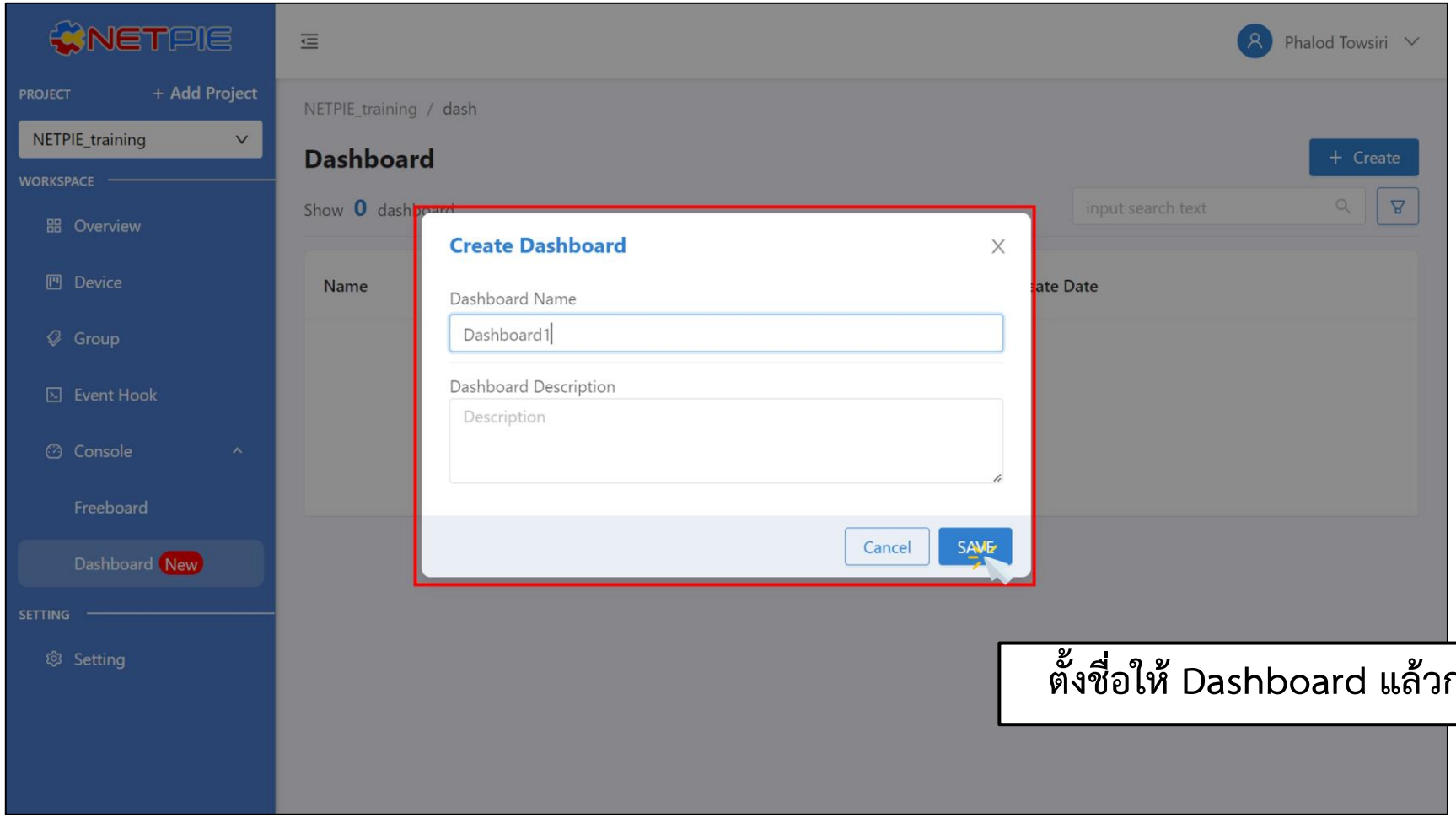
การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource

การสร้าง Dashboard



ใบงานที่ 5.3 ขั้นตอนการทดลอง

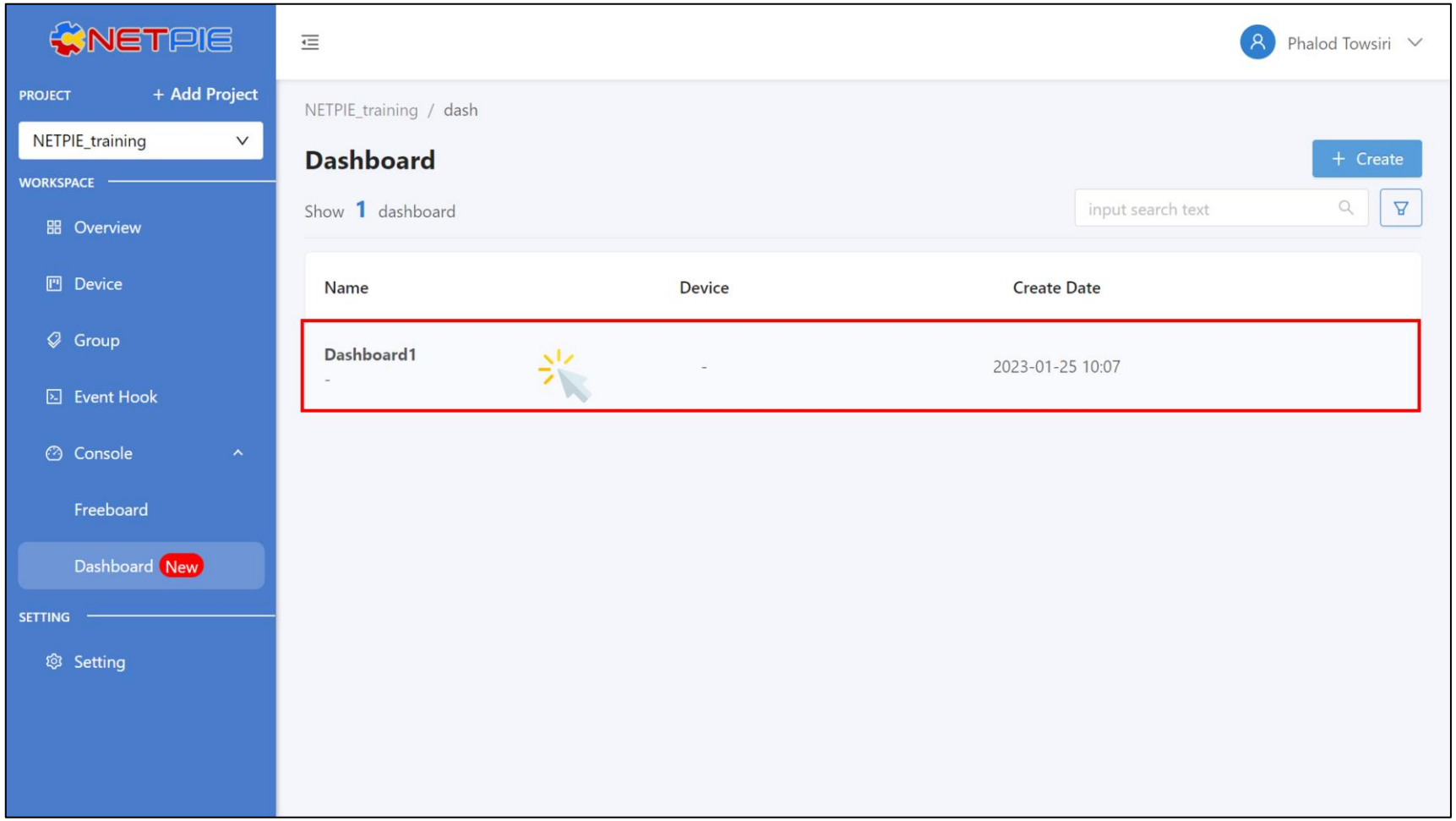
การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource



ตั้งชื่อให้ Dashboard แล้วกด Save

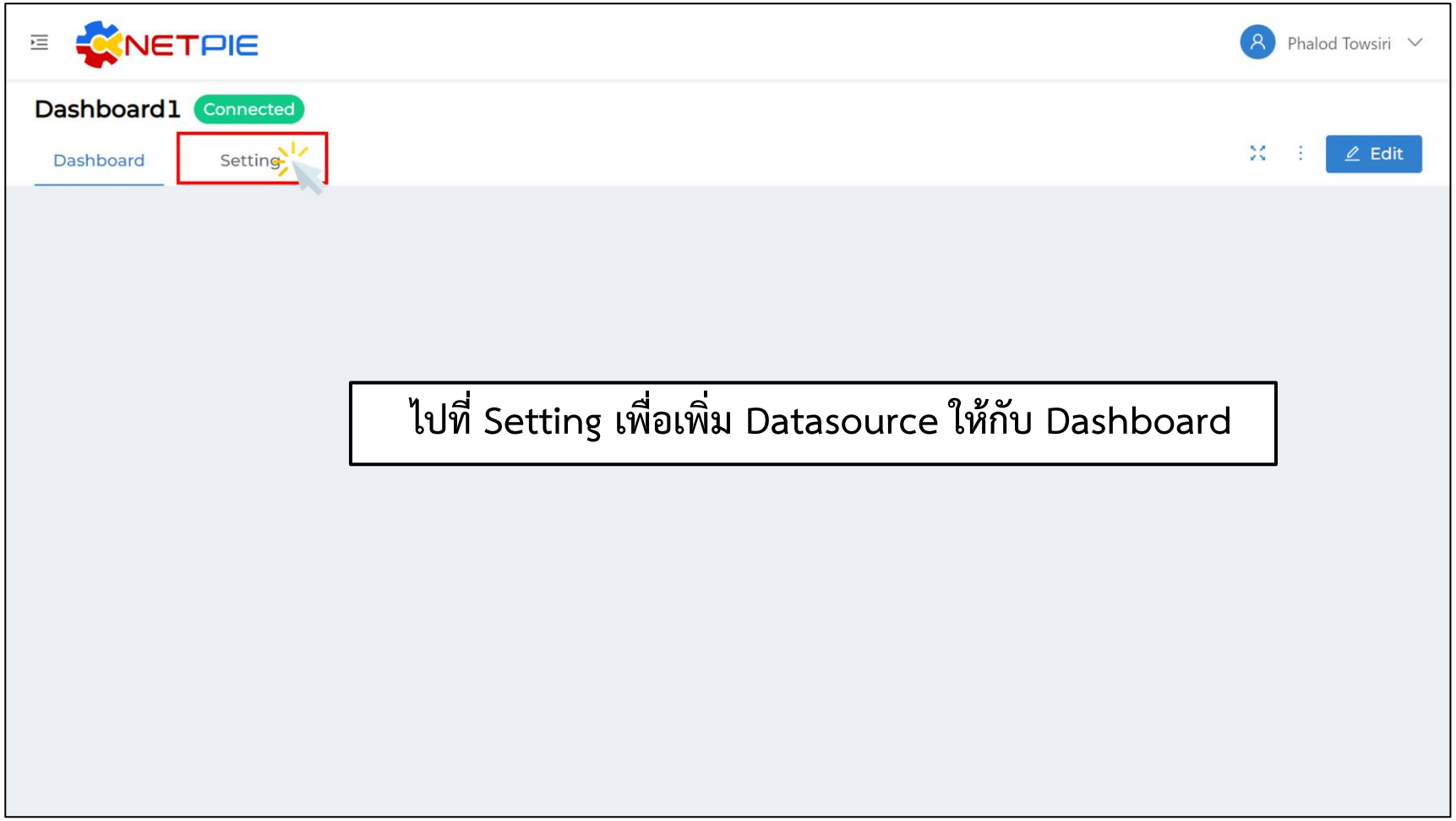
ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource



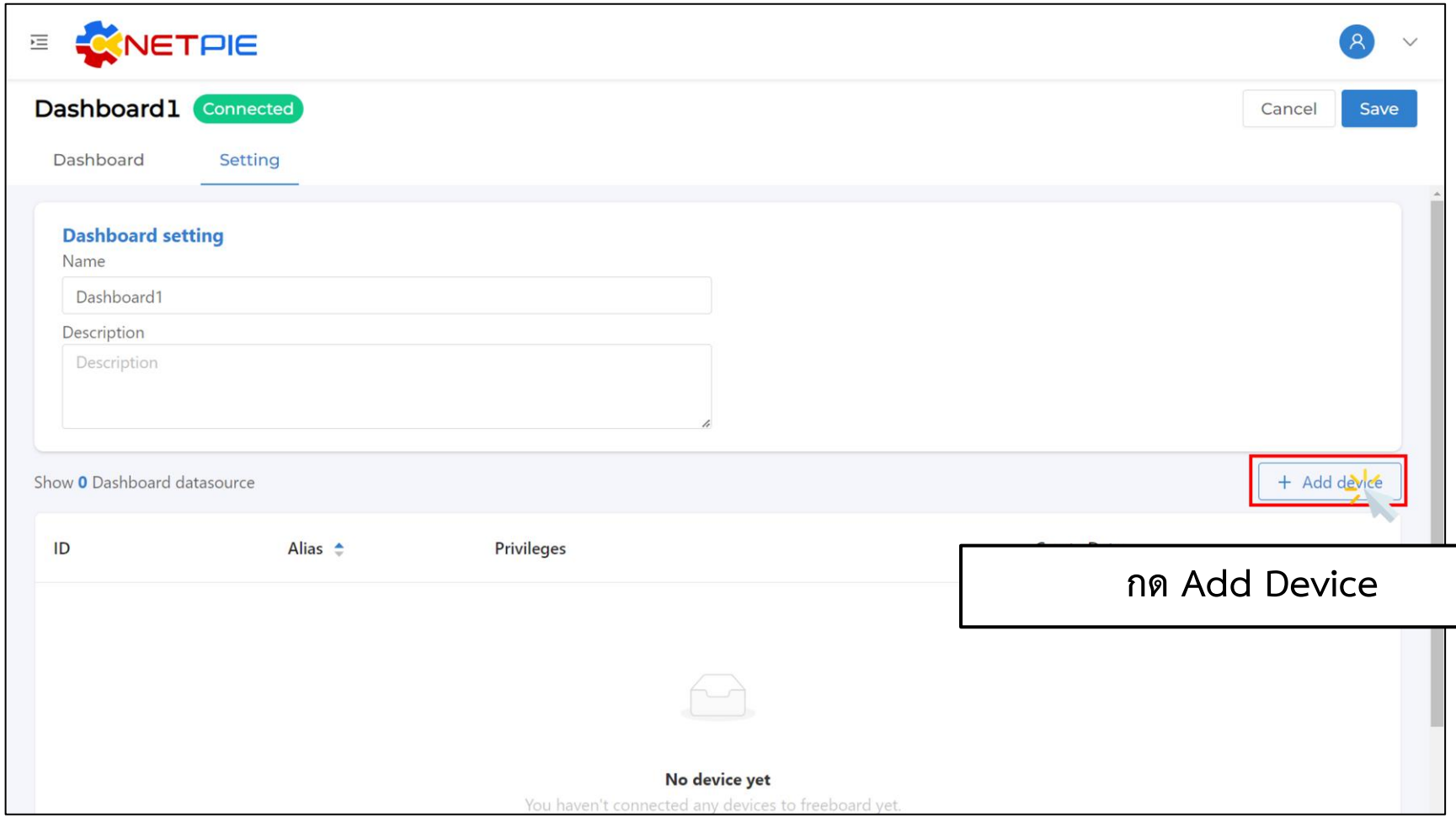
ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource



ใบงานที่ 5.3 ขั้นตอนการทดลอง

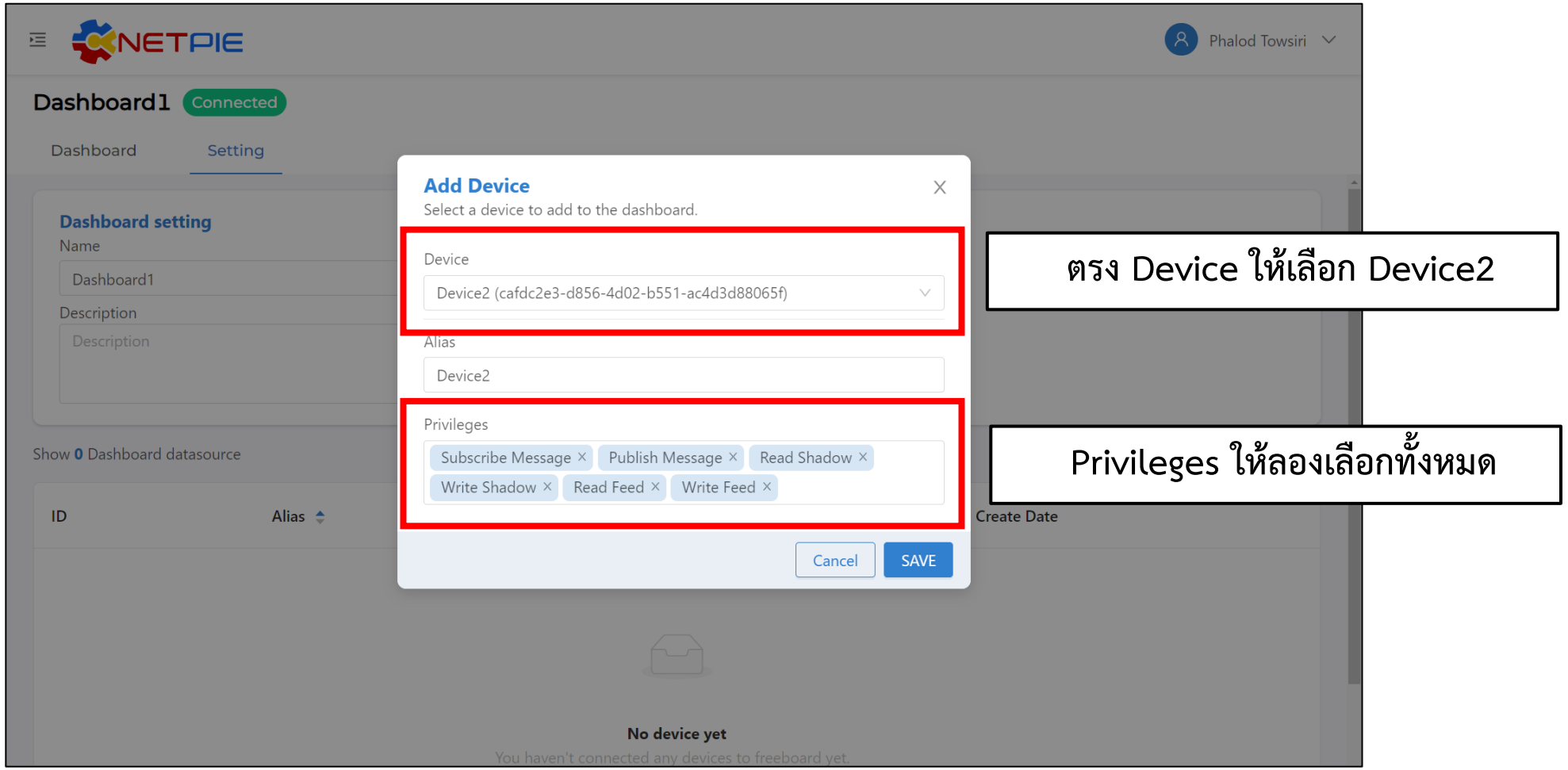
การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource



กด Add Device

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource

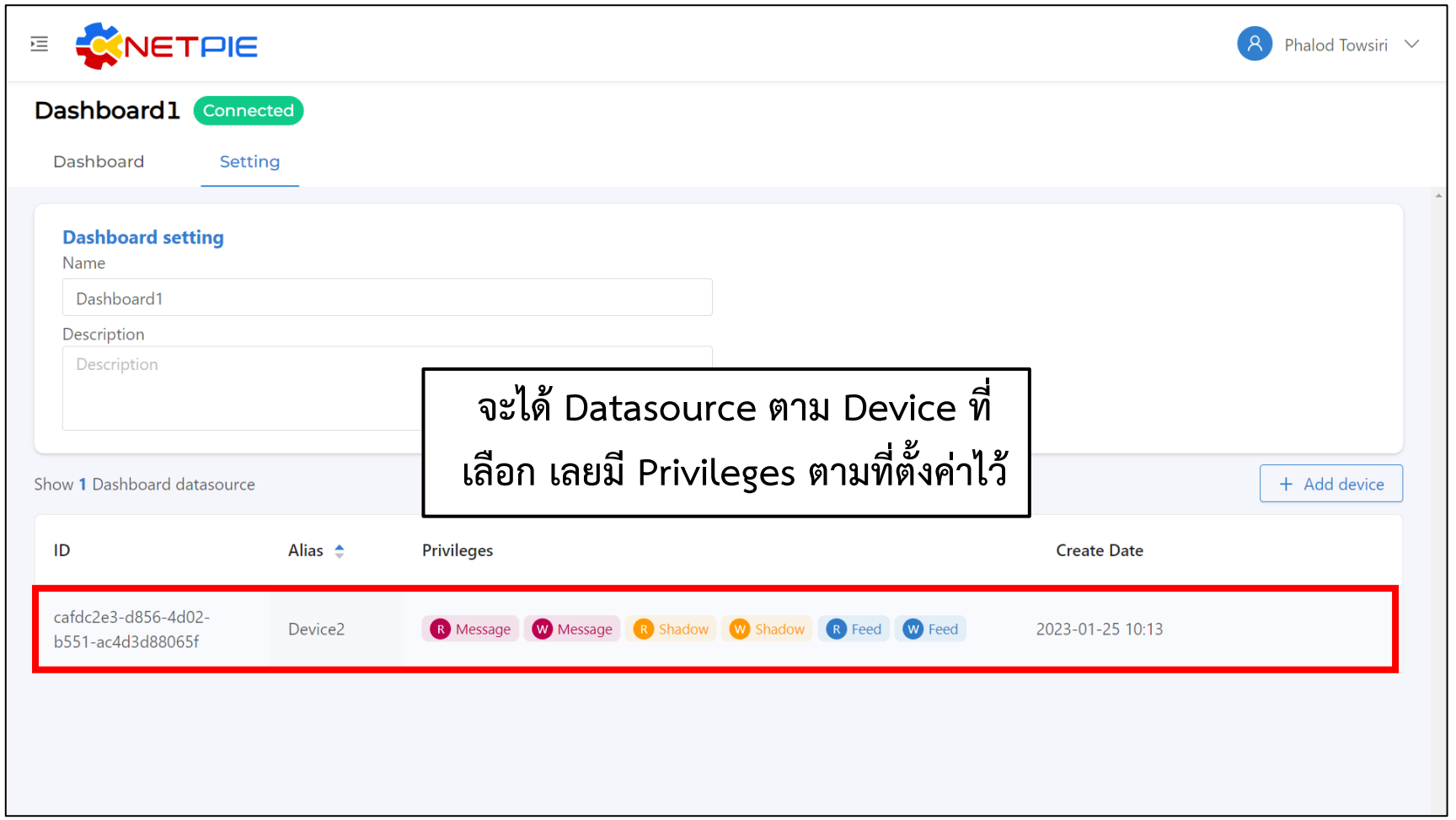


ตรง Device ให้เลือก Device2

Privileges ให้ลองเลือกทั้งหมด

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource



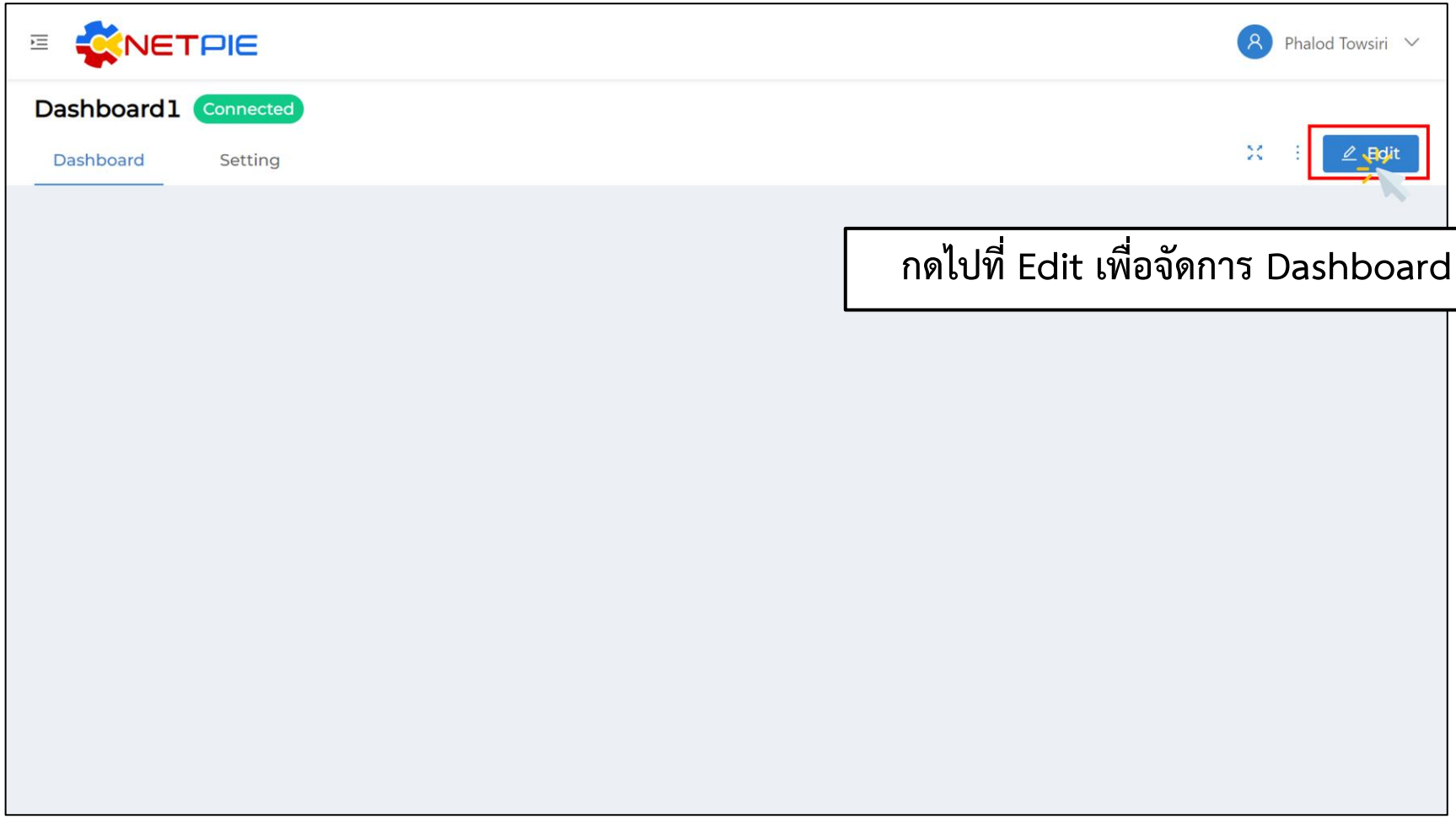
ใบงานที่ 5.3 ทฤษฎีเบื้องต้น

การนำเสนอข้อมูล

Dashboard มีความสามารถที่ไม่ได้แตกต่างไปจาก Freeboard คือ สามารถนำเสนอข้อมูลได้ ผ่าน Widgets ต่างๆ ที่ถูกพัฒนามาสำหรับการแสดงผล โดย Widgets แบ่งออกเป็น 2 ประเภท คือ Data Display Widgets และ Control Widgets ไม่ต่างจาก Freeboard เลย มีเพียงการตั้งค่า Widgets ที่ต่างไปจากเดิม

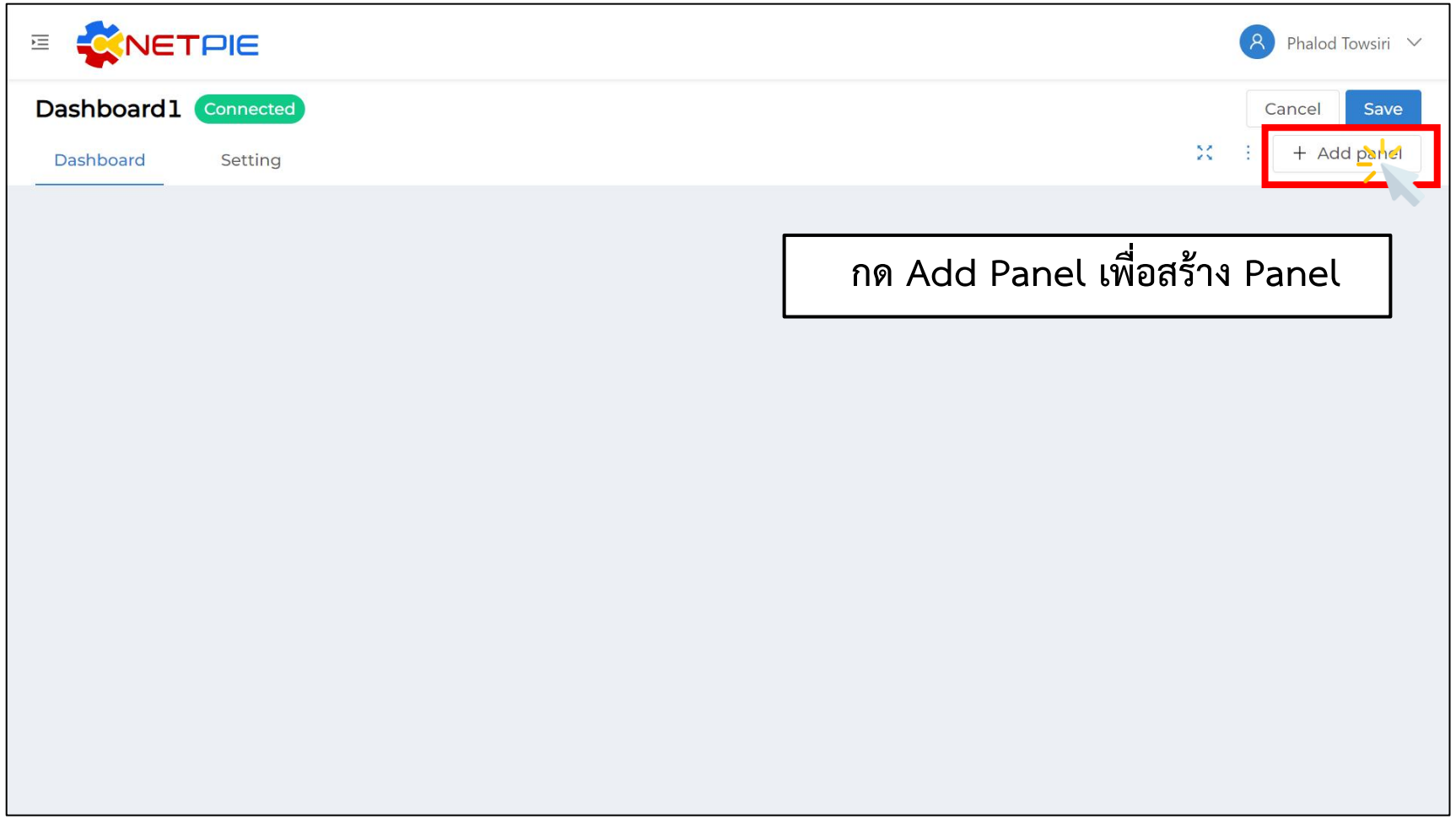
ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



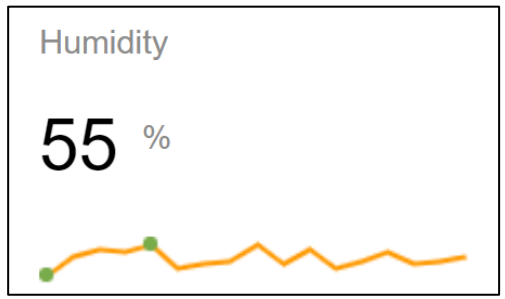
ใบงานที่ 5.3 ขั้นตอนการทดลอง

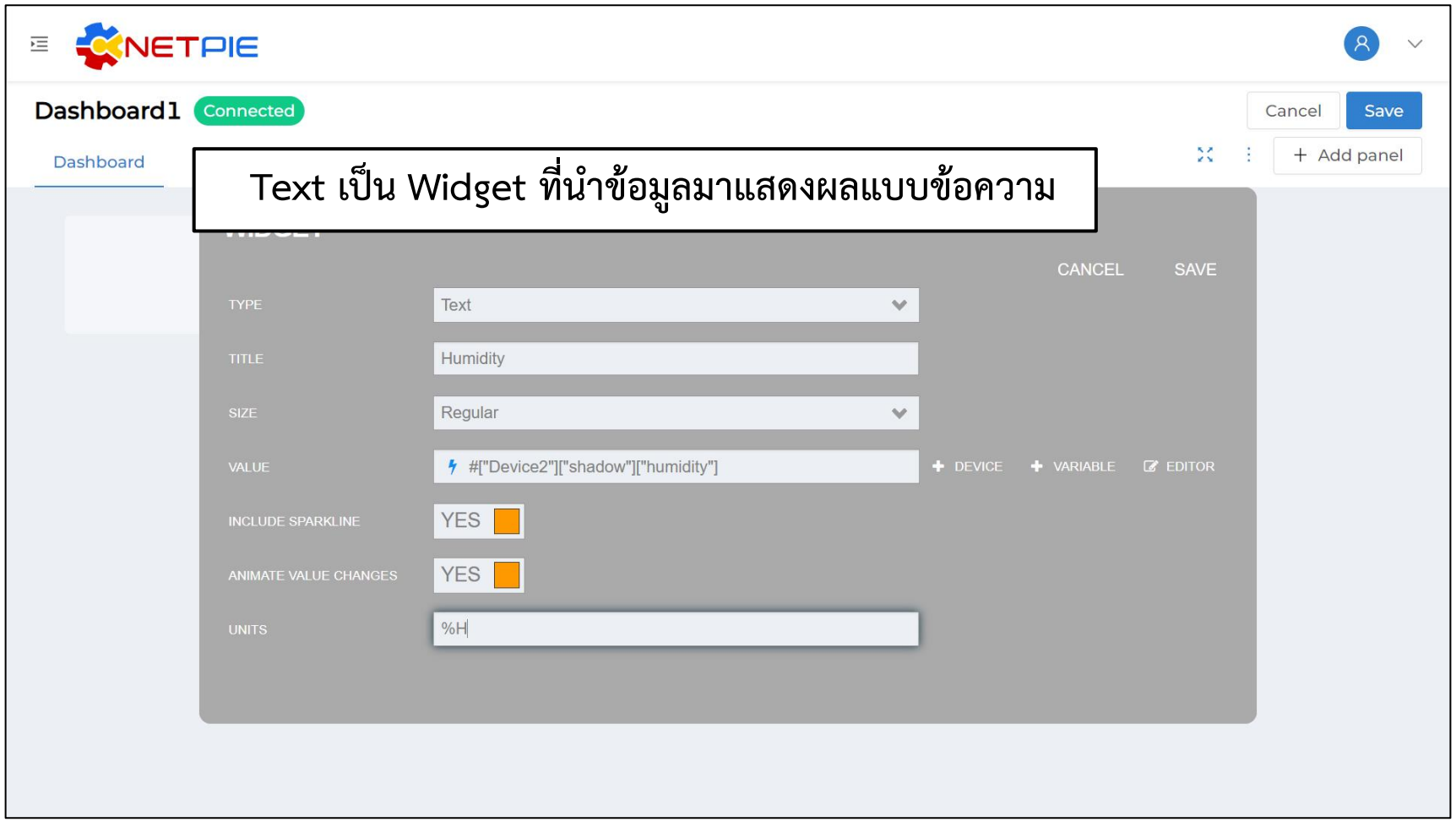
การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text

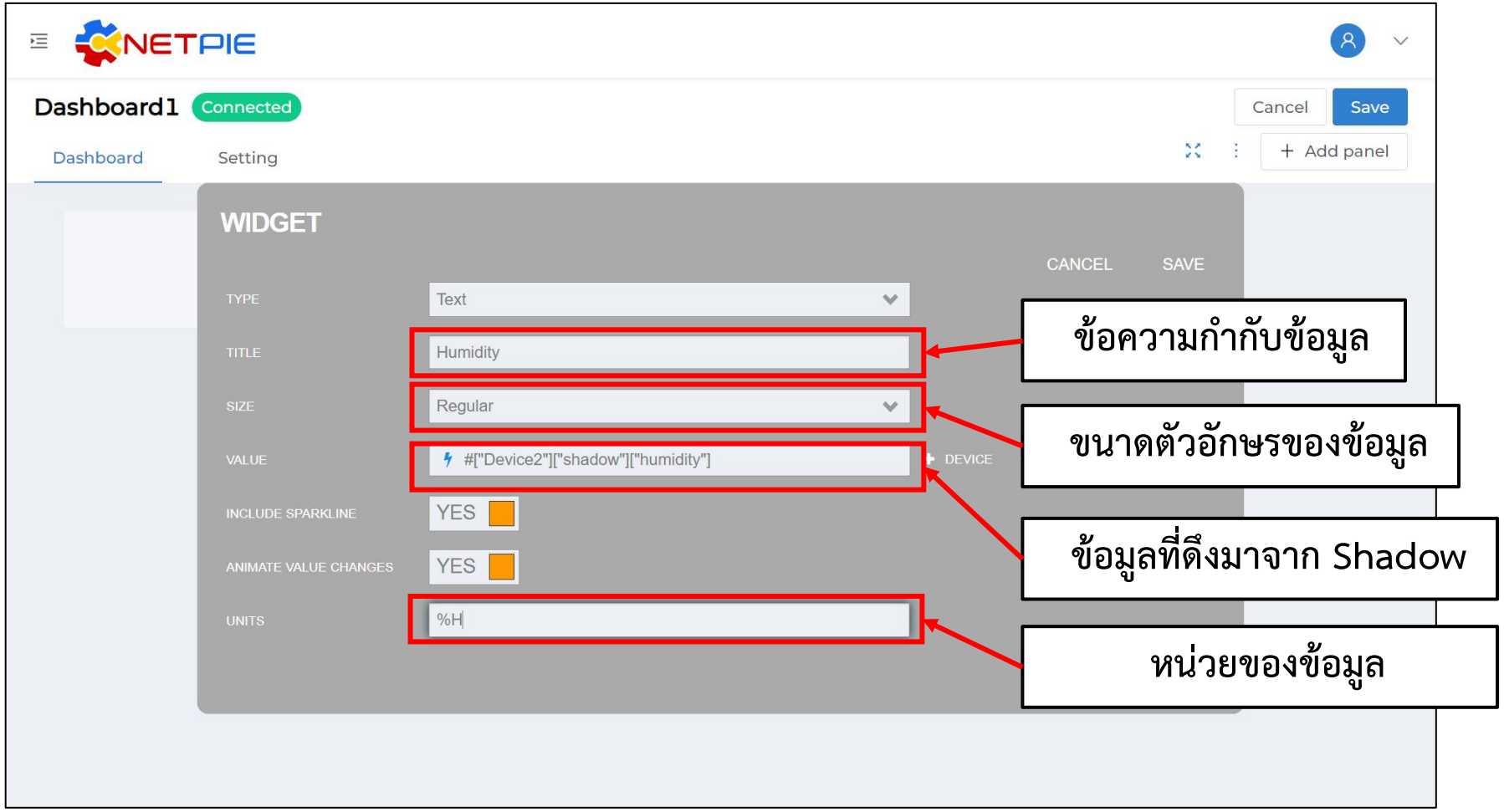




ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text

เมื่อกด Save จะได้ Widget Text ที่แสดงข้อมูล Humidity ของ Device2

ใบงานที่ 5.3 ขั้นตอนการทดลอง

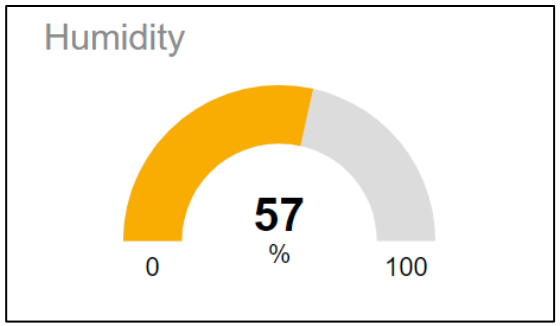
การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

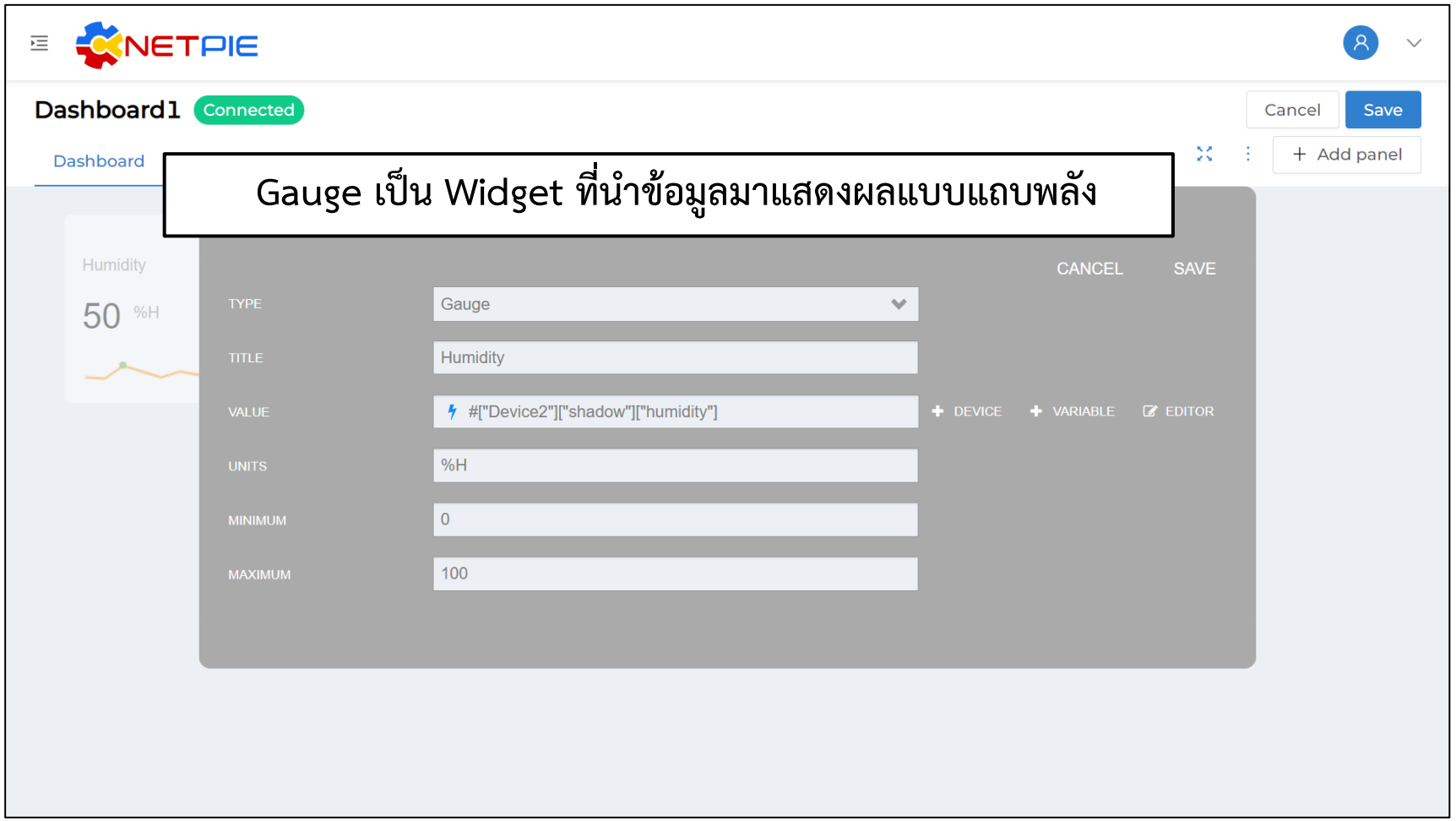
ต่อไป จะเป็นการเพิ่ม Widget Gauge เข้าไปที่ Panel เดิม โดยกดไปที่ปุ่ม +

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge

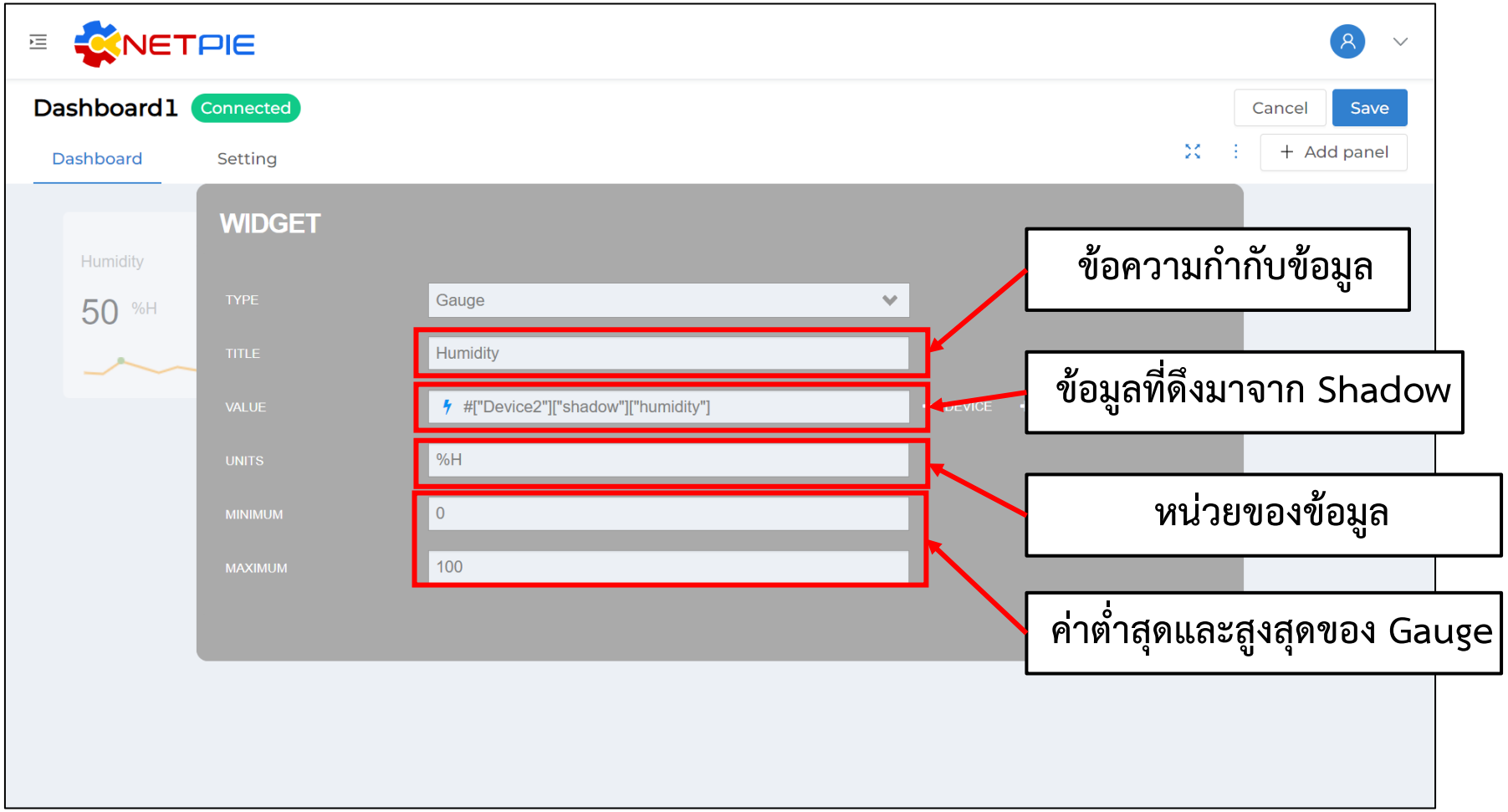




ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge

เมื่อกด Save จะได้ Widget Gauge ที่แสดงข้อมูล Humidity ของ Device2

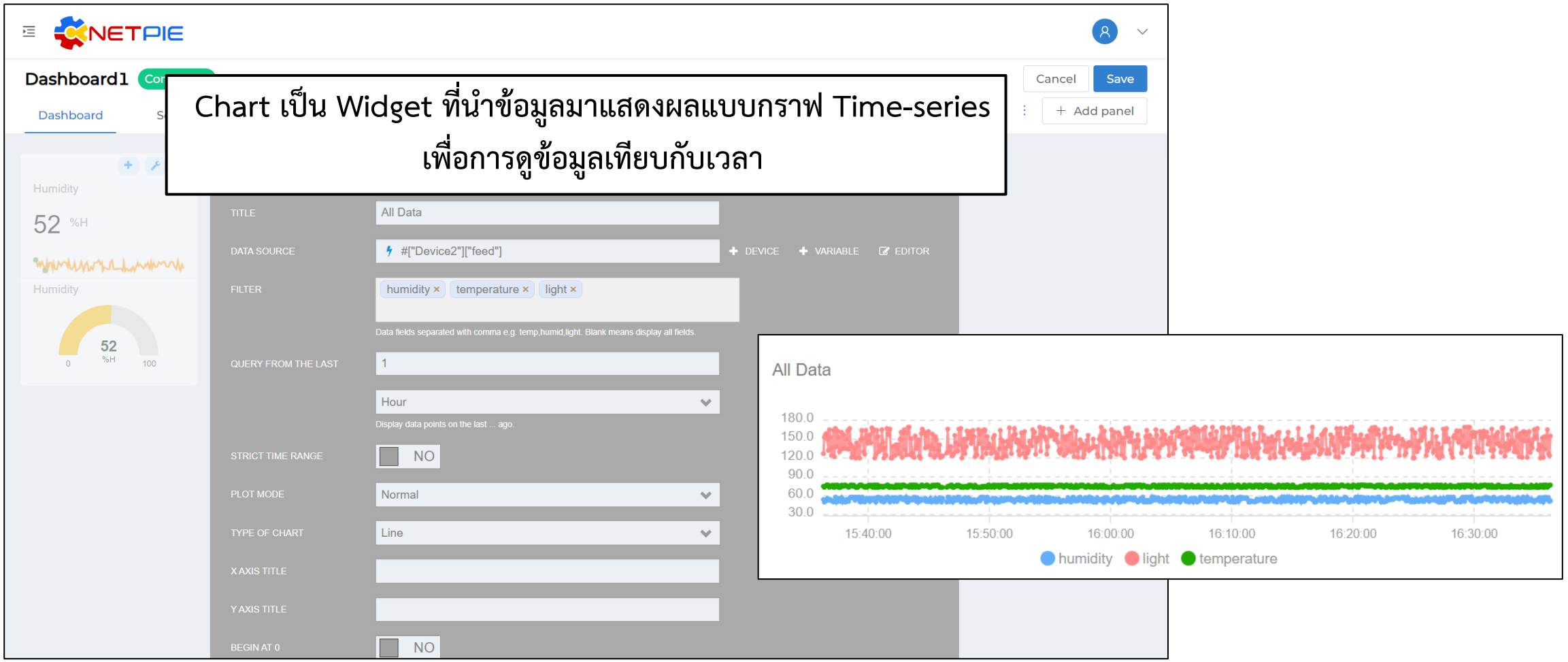
ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

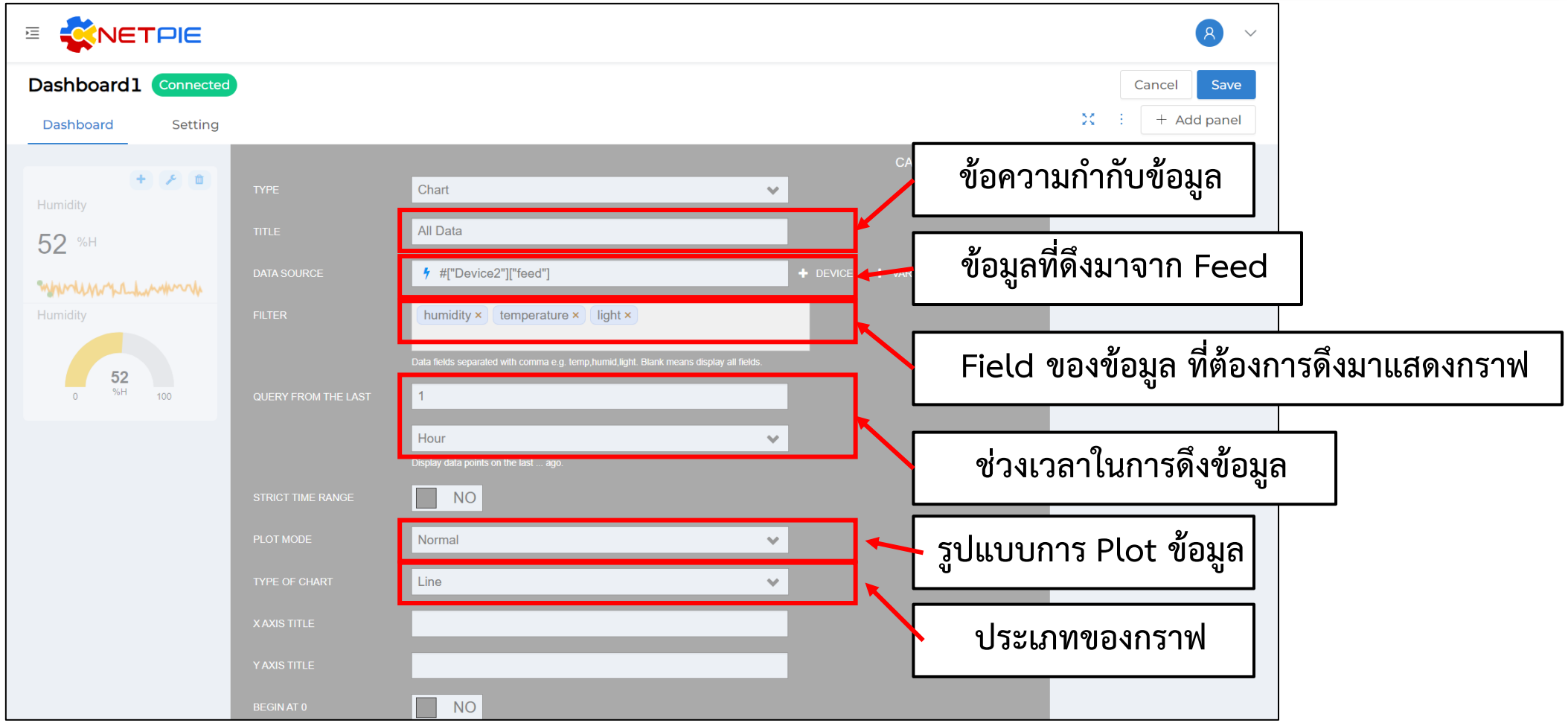
การสร้าง Widget : Chart



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

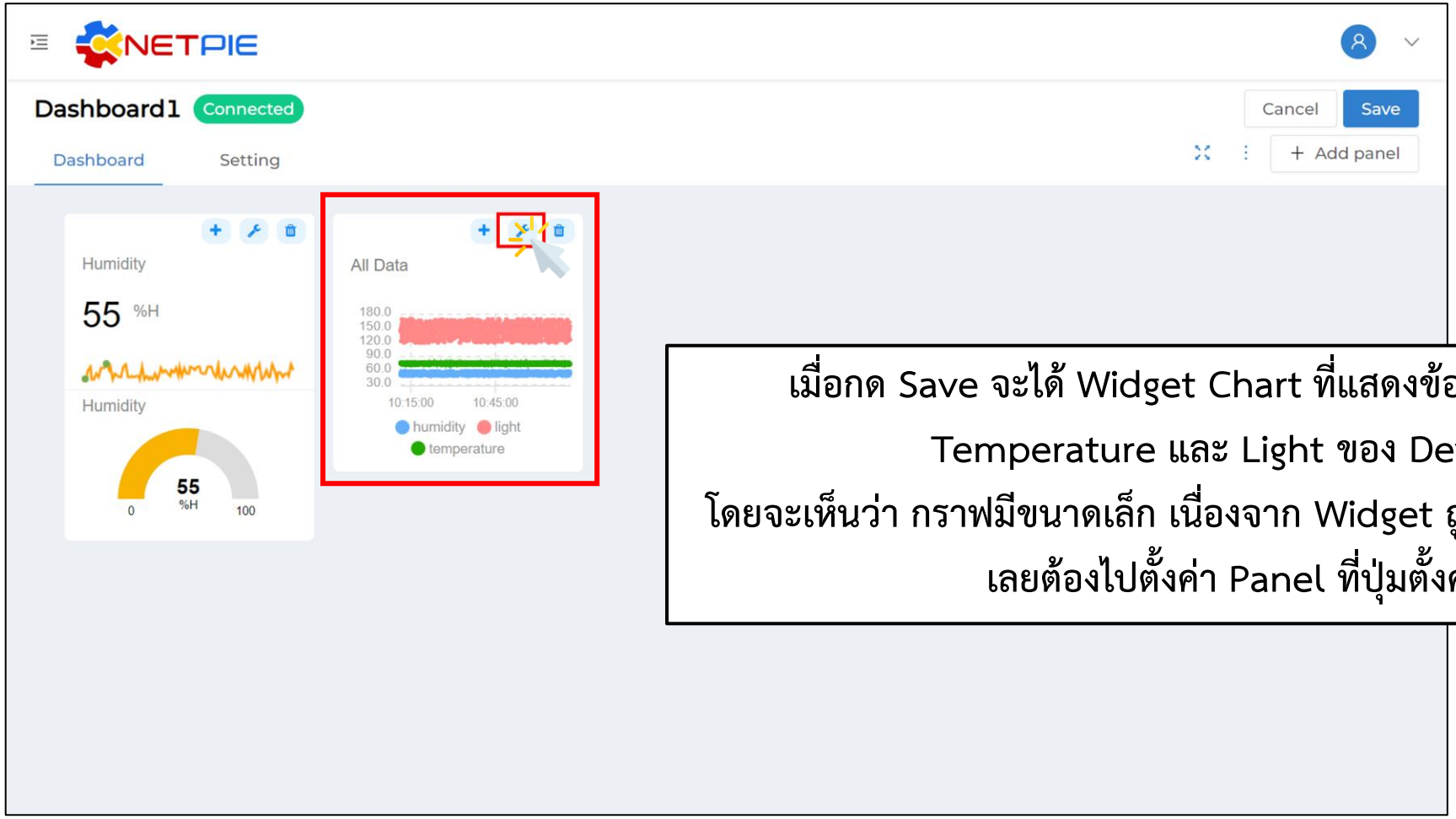
การสร้าง Widget : Chart



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Chart



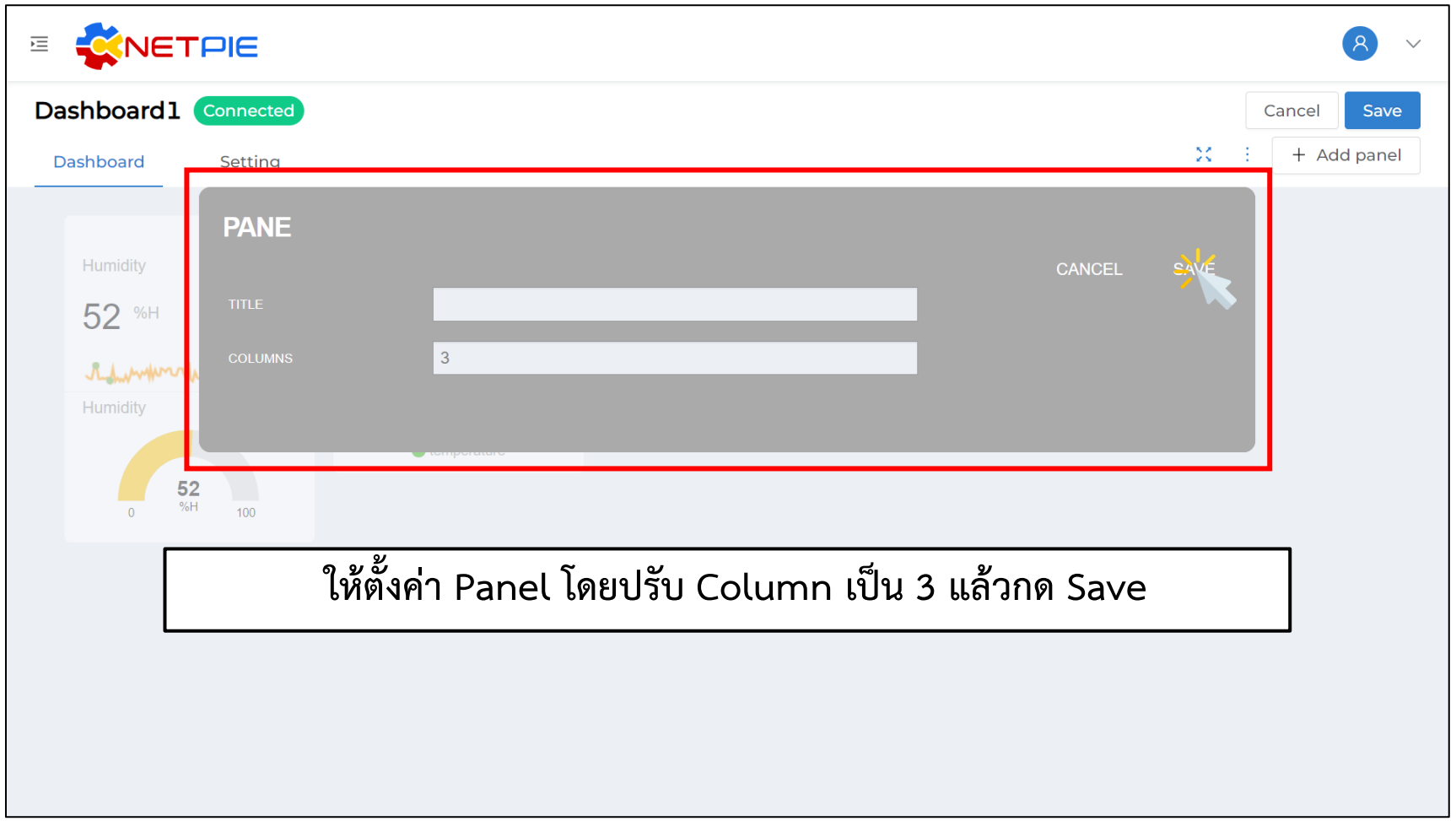
เมื่อกด Save จะได้ Widget Chart ที่แสดงข้อมูล Humidity, Temperature และ Light ของ Device2

โดยจะเห็นว่า กราฟมีขนาดเล็ก เนื่องจาก Widget ถูก Limit ตาม Panel เลยต้องไปตั้งค่า Panel ที่ปุ่มตั้งค่า

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Chart

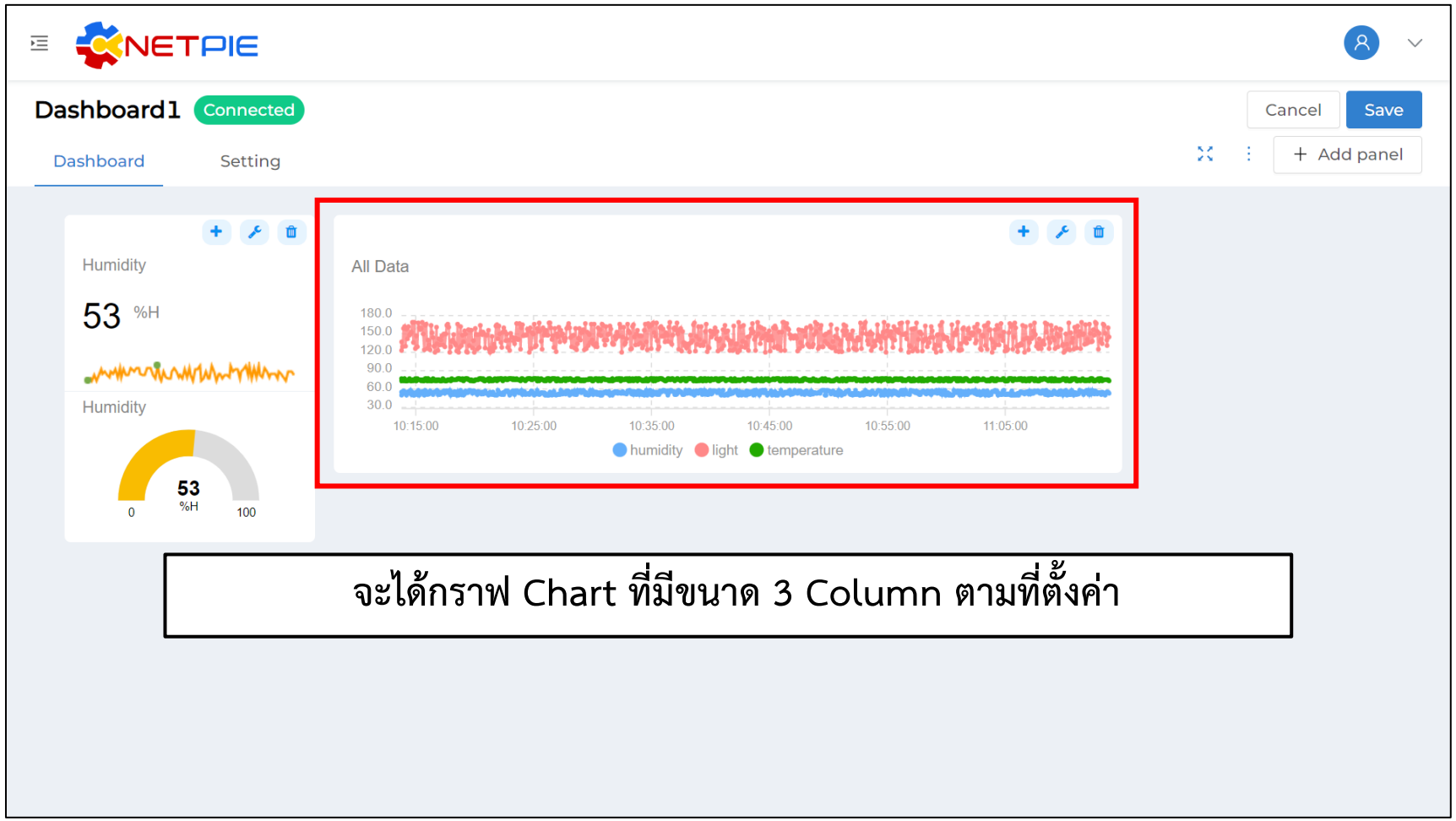


ให้ตั้งค่า Panel โดยปรับ Column เป็น 3 แล้วกด Save

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

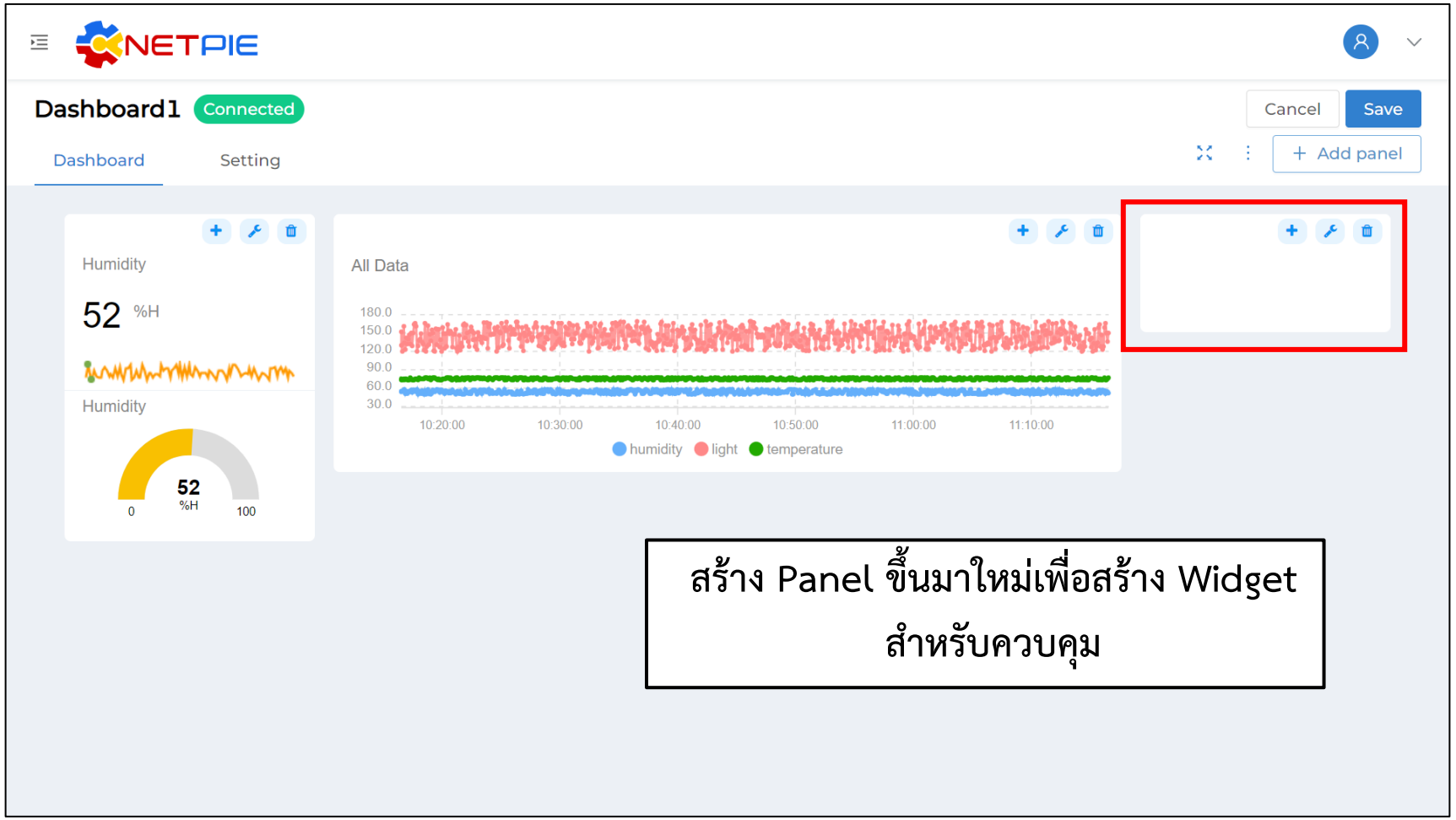
การสร้าง Widget : Chart



จะได้กราฟ Chart ที่มีขนาด 3 Column ตามที่ตั้งค่า

ใบงานที่ 5.3 ขั้นตอนการทดลอง

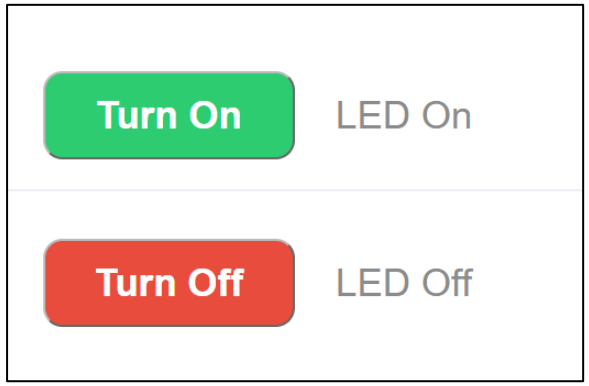
การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

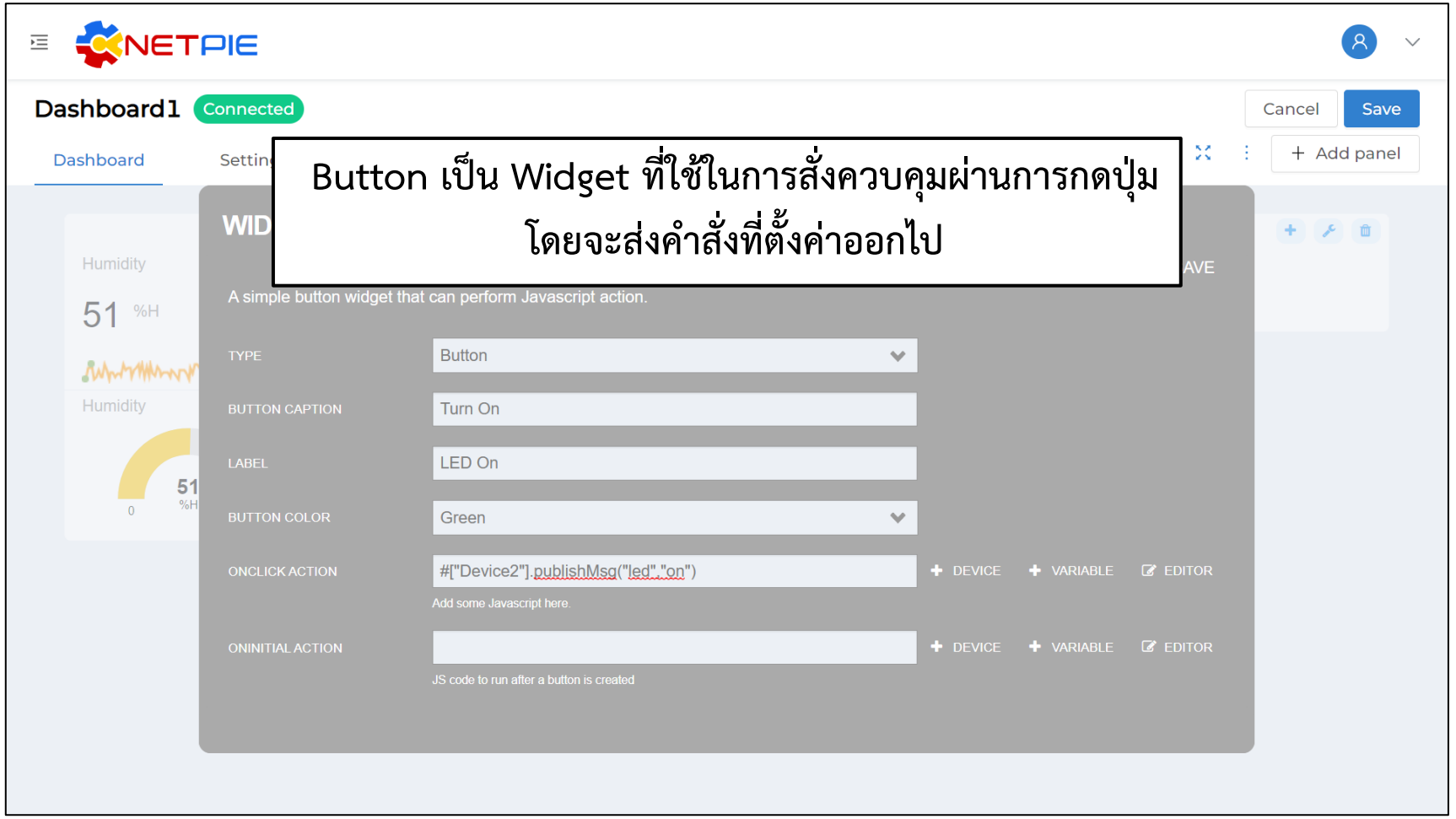


ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Button

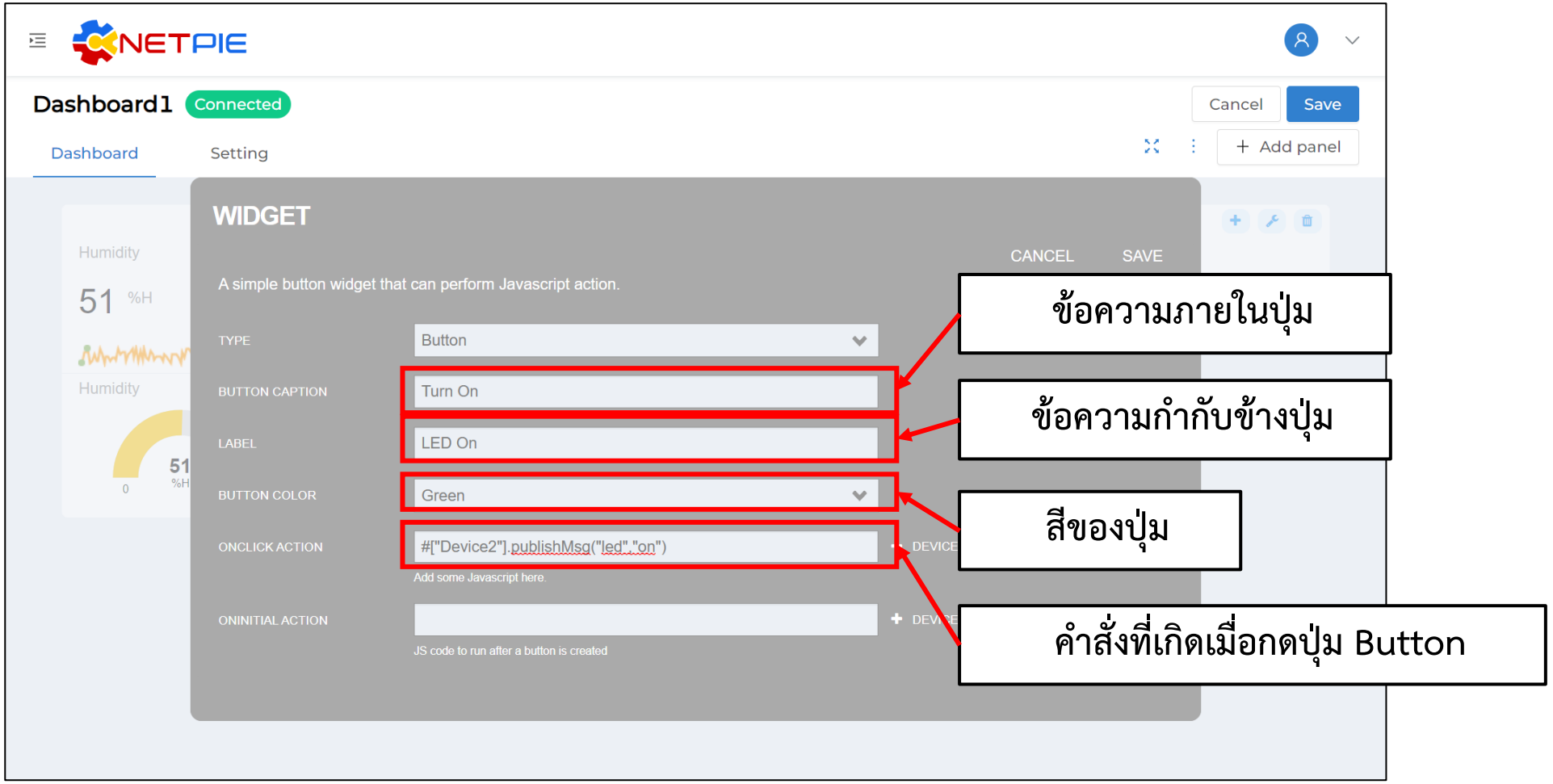




ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

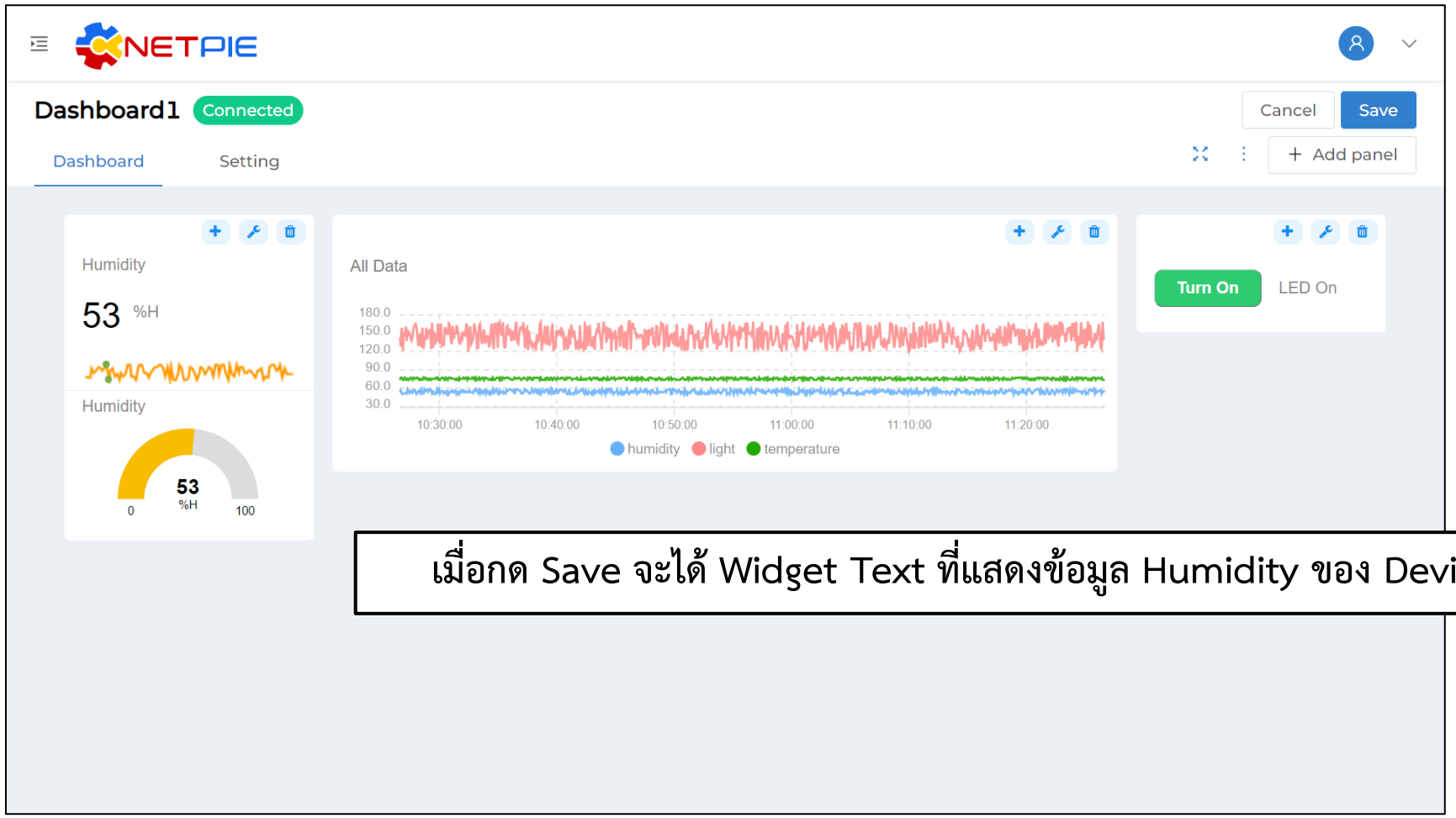
การสร้าง Widget : Button



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Button

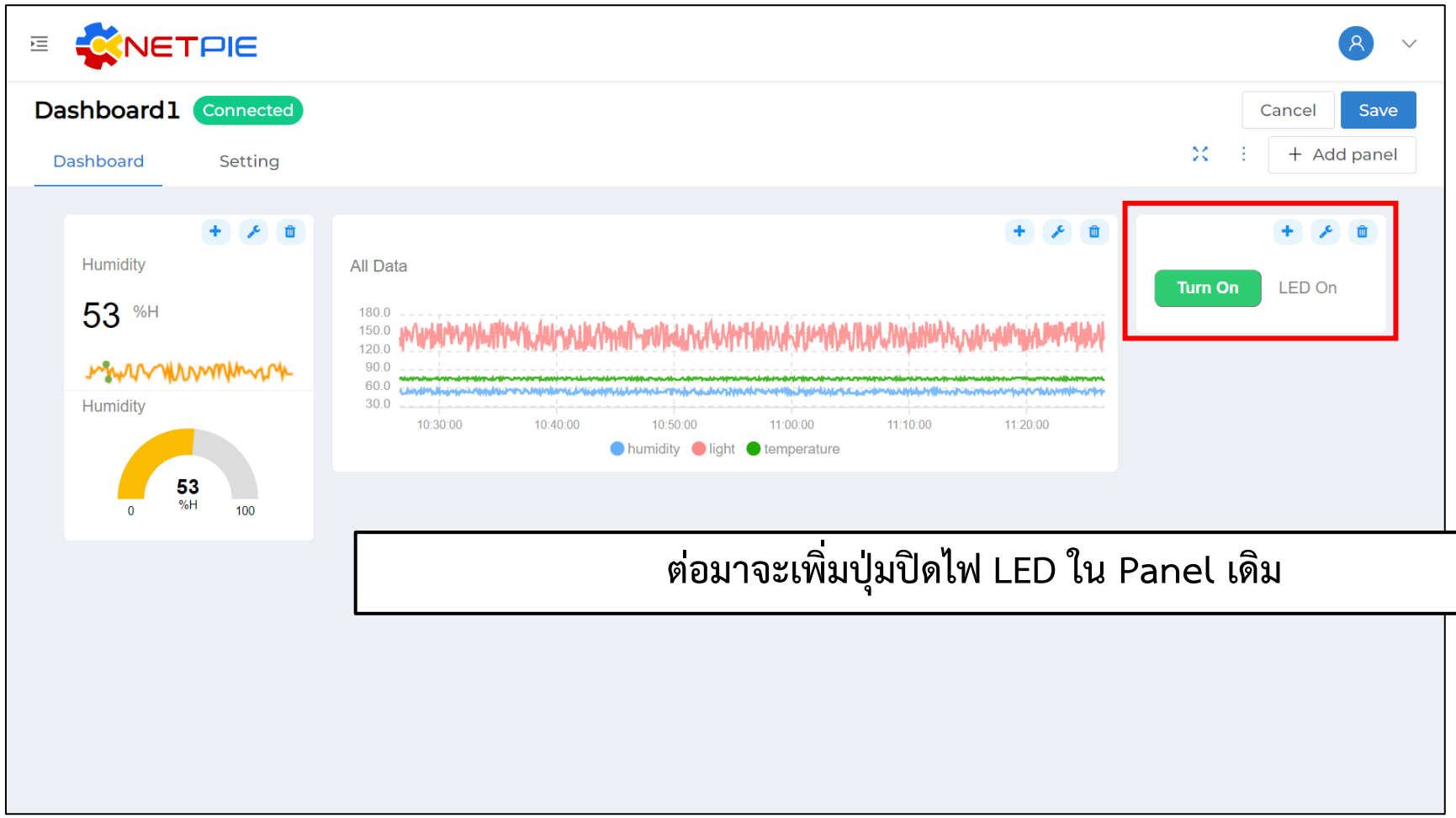


เมื่อกด Save จะได้ Widget Text ที่แสดงข้อมูล Humidity ของ Device2

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Button

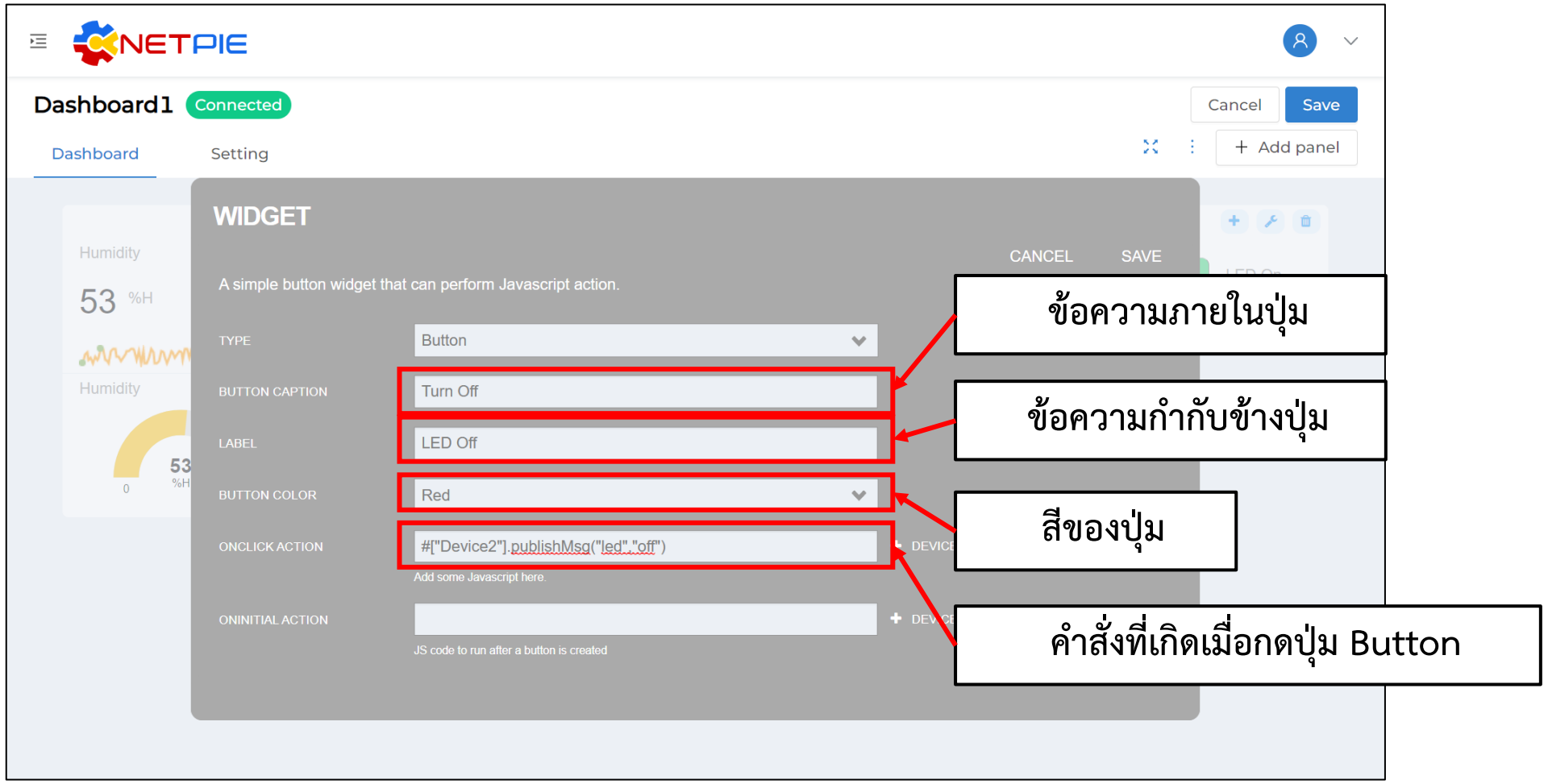


ต่อมาจะเพิ่มปุ่มปิดไฟ LED ใน Panel เดิม

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

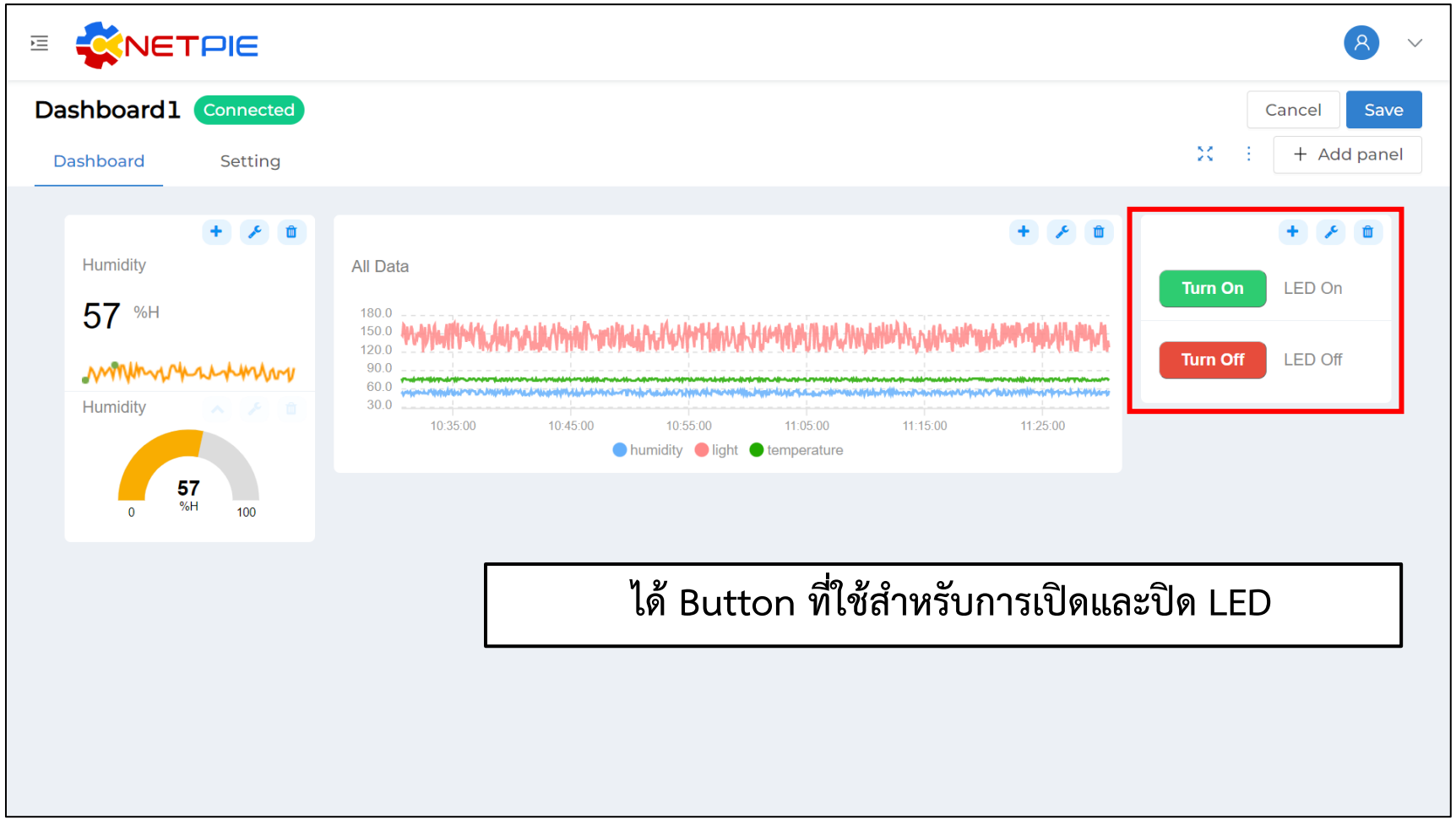
การสร้าง Widget : Button



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

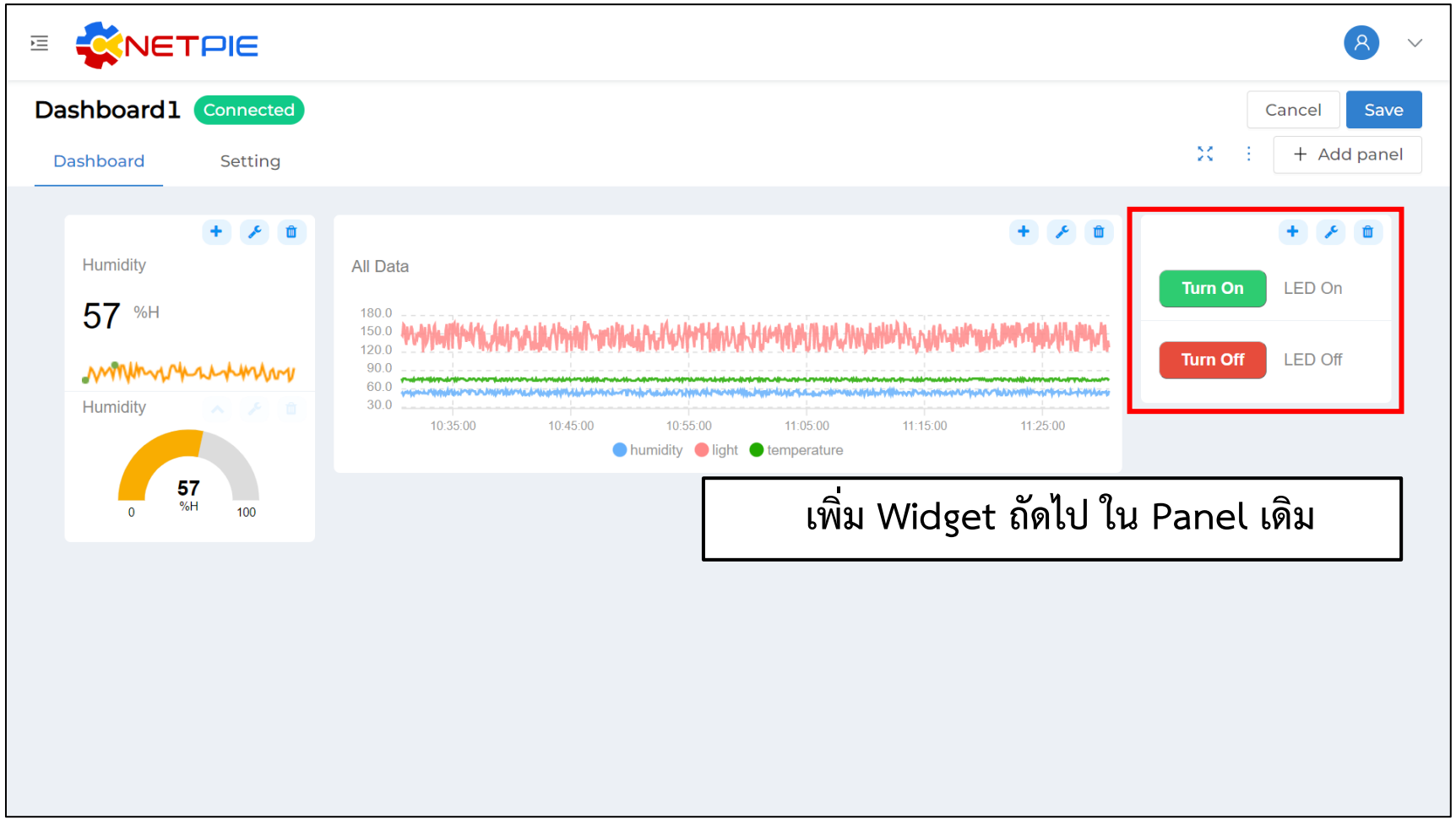
การสร้าง Widget : Button



ได้ Button ที่ใช้สำหรับการเปิดและปิด LED

ใบงานที่ 5.3 ขั้นตอนการทดลอง

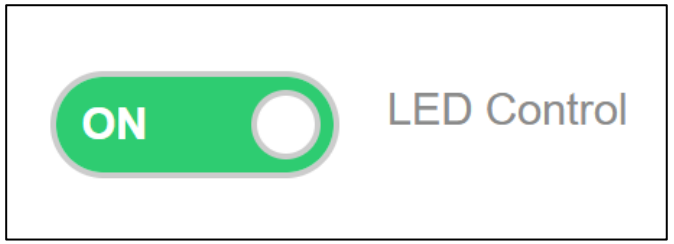
การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

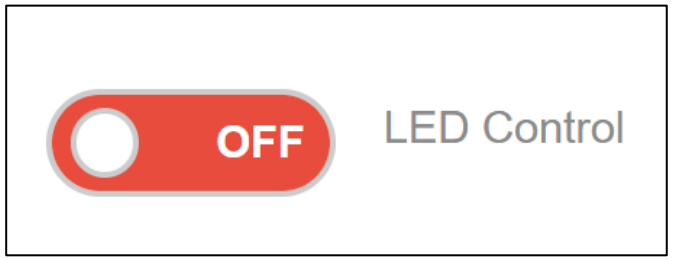


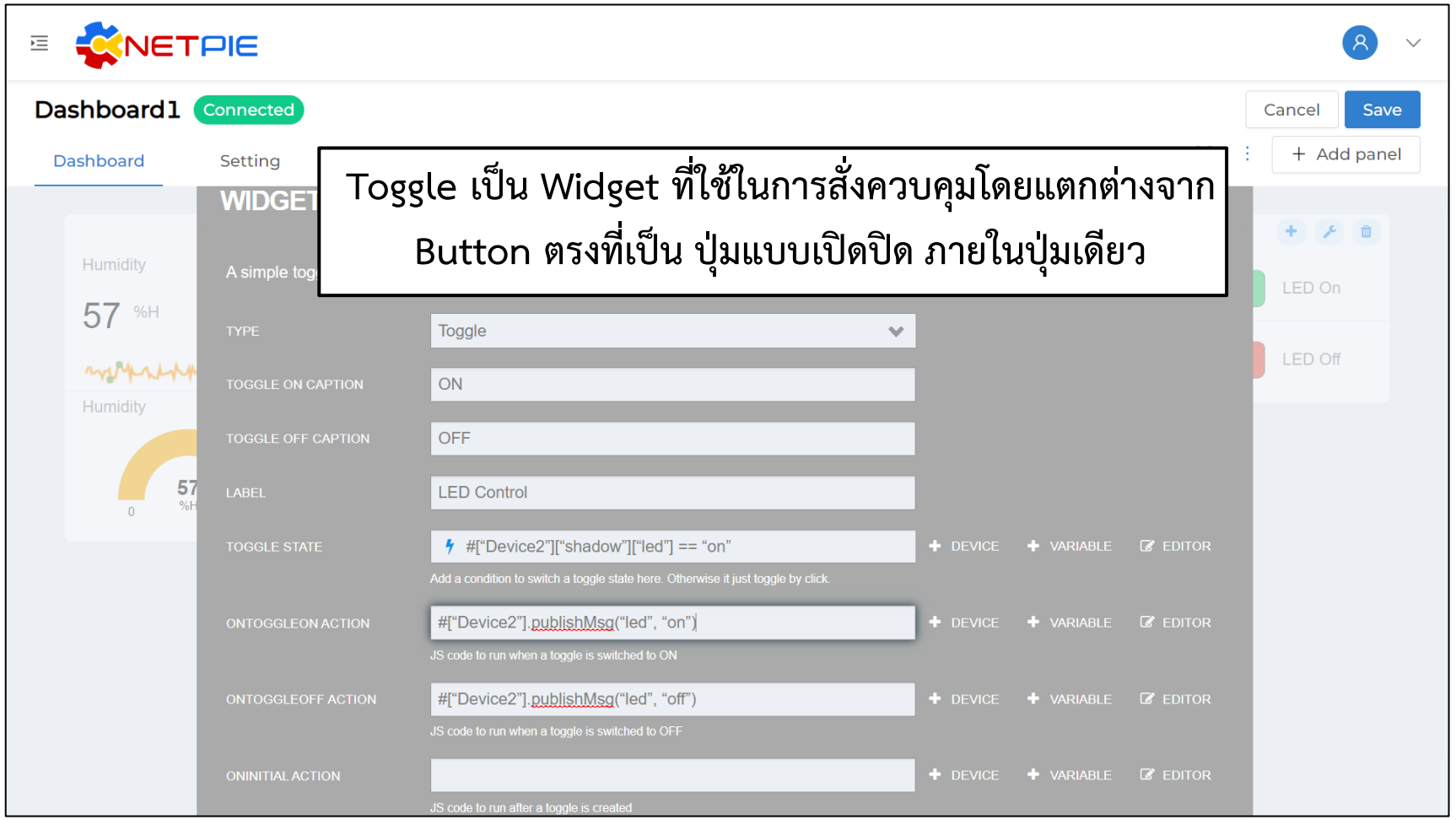
ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Toggle



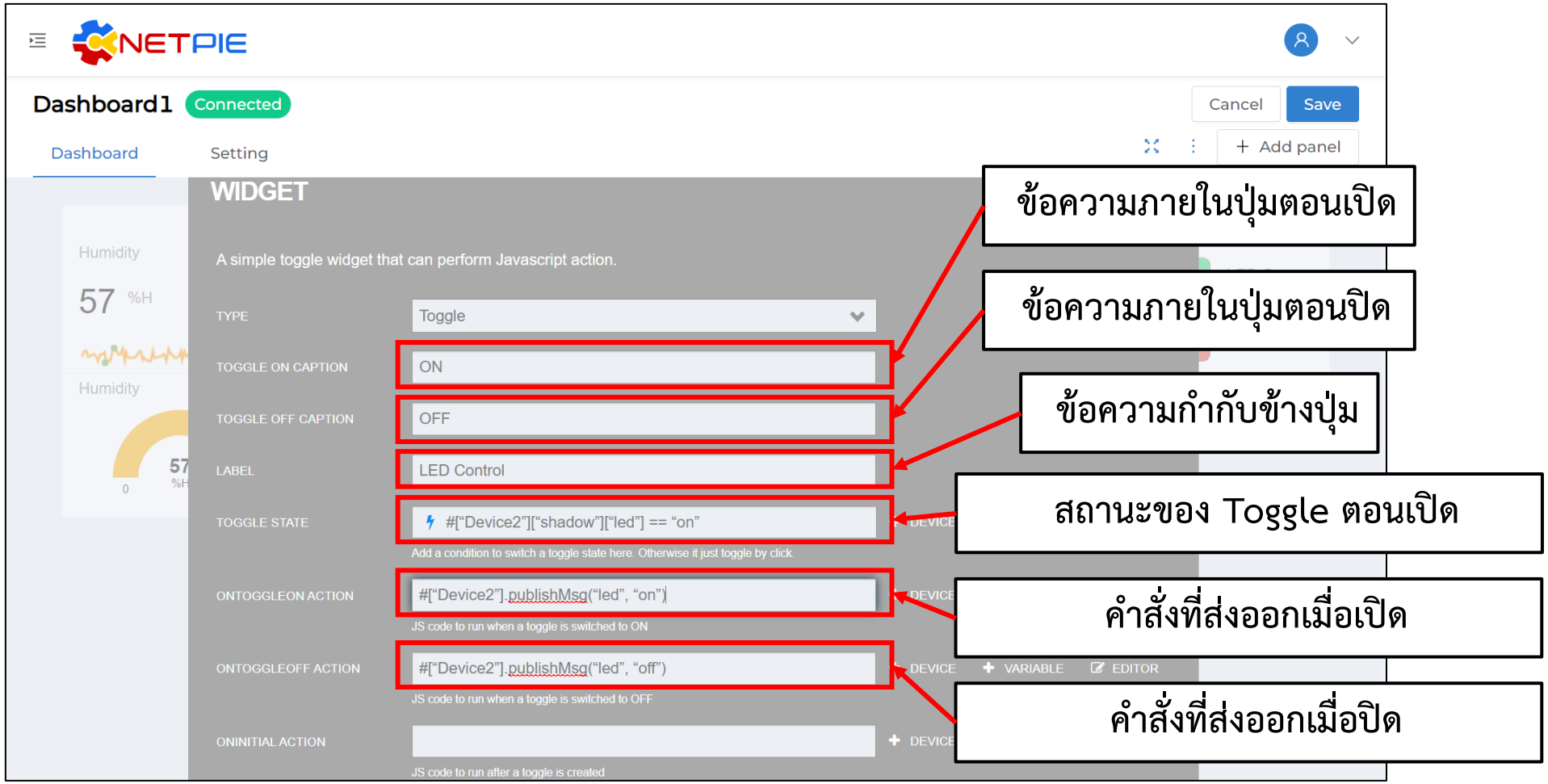




ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

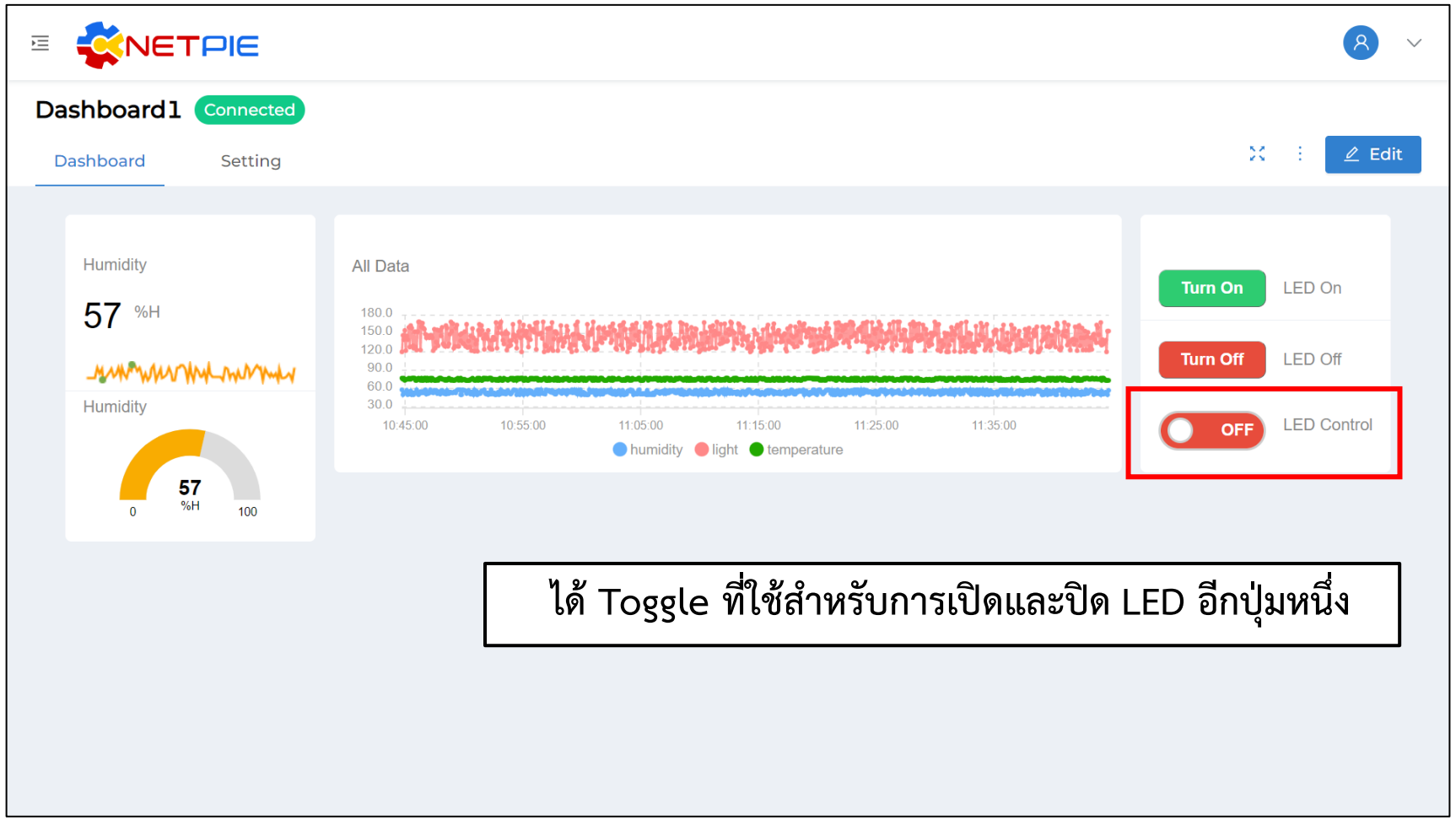
การสร้าง Widget : Toggle



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

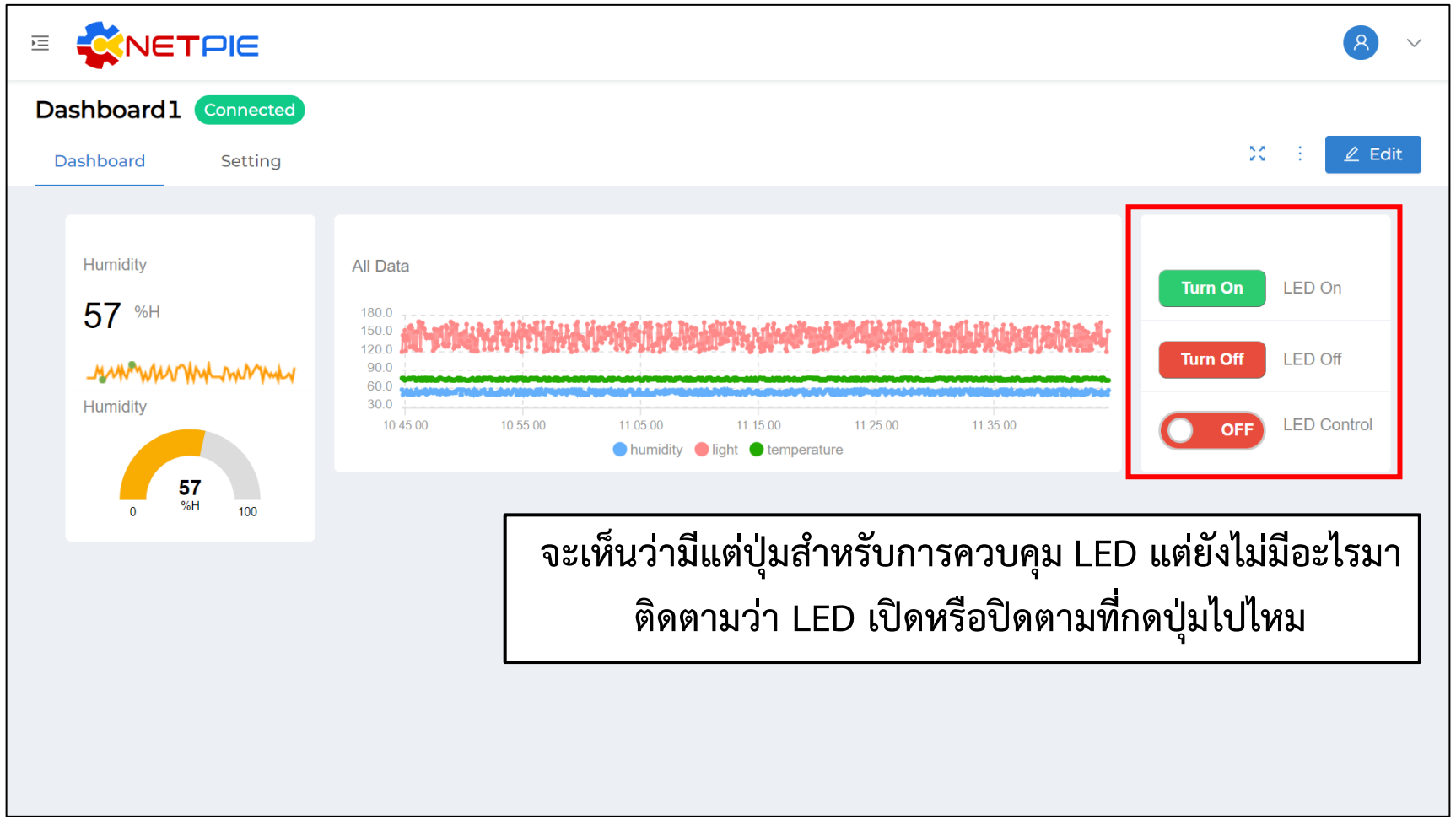
การสร้าง Widget : Toggle



ได้ Toggle ที่ใช้สำหรับการเปิดและปิด LED อีกปุ่มหนึ่ง

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

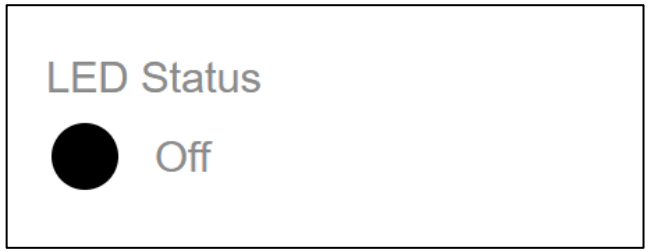


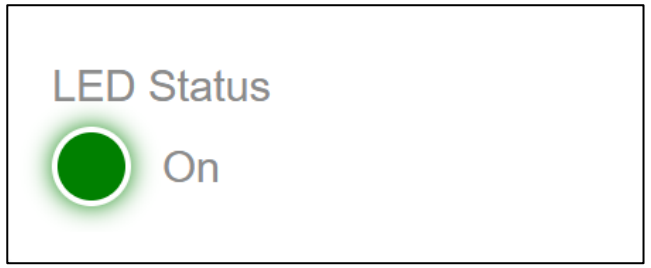
จะเห็นว่ามีแต่ปุ่มสำหรับการควบคุม LED แต่ยังไม่มีอะไรมาติดตามว่า LED เปิดหรือปิดตามที่กดปุ่มไปไหม

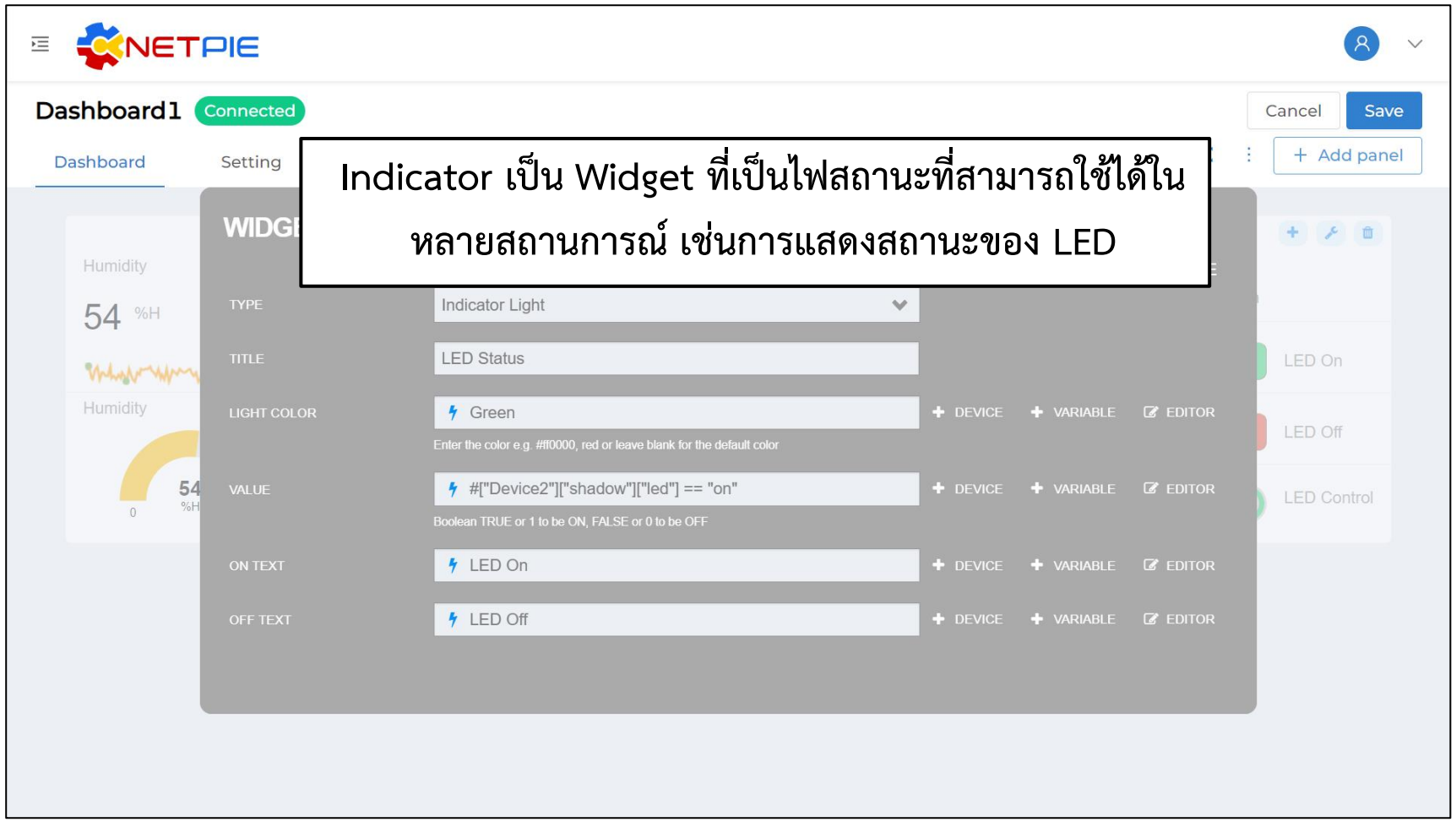
ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Indicator Light



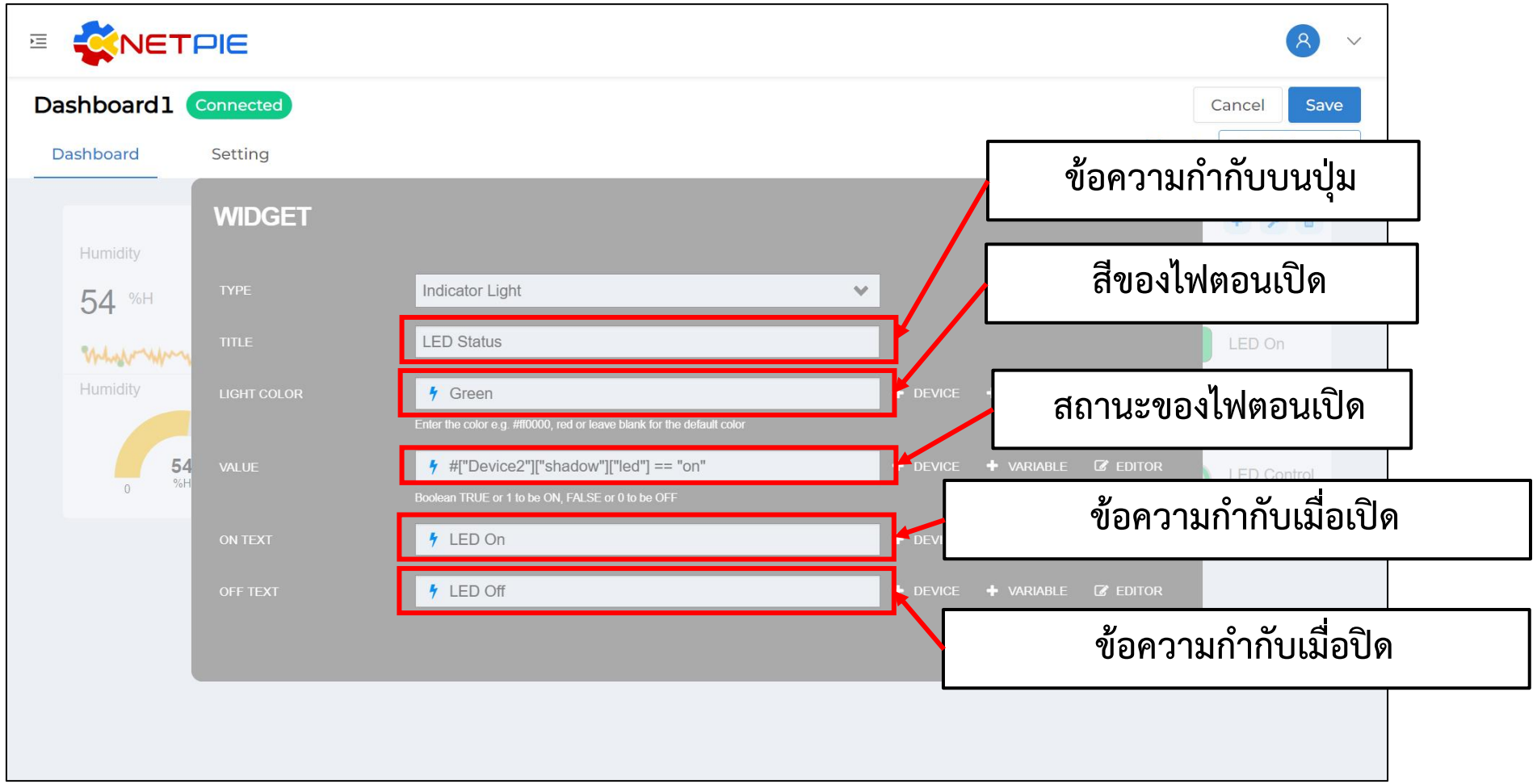




ใบงานที่ 5.3 ขั้นตอนการทดลอง

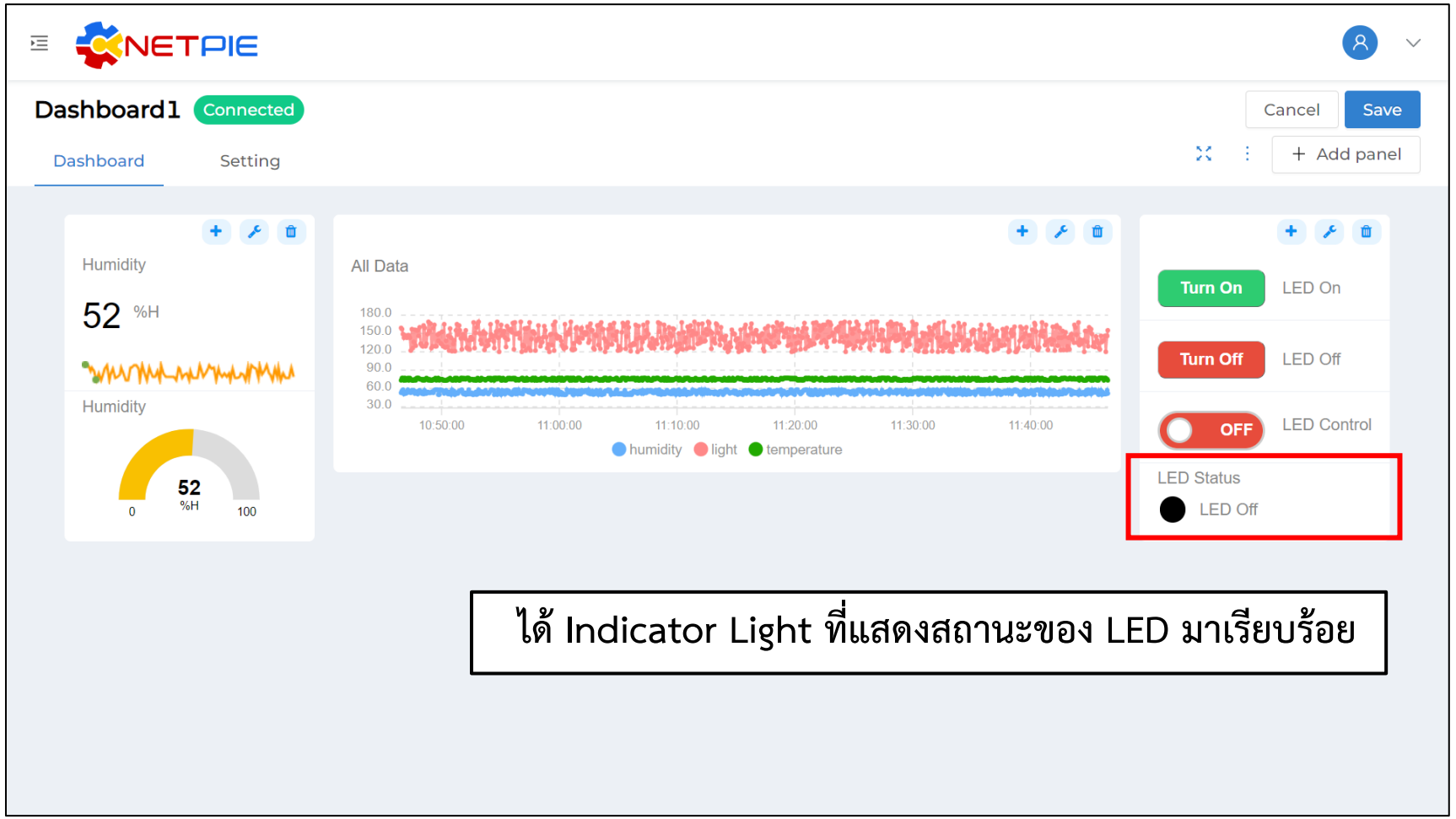
การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Indicator Light



ใบงานที่ 5.3 ขั้นตอนการทดลอง

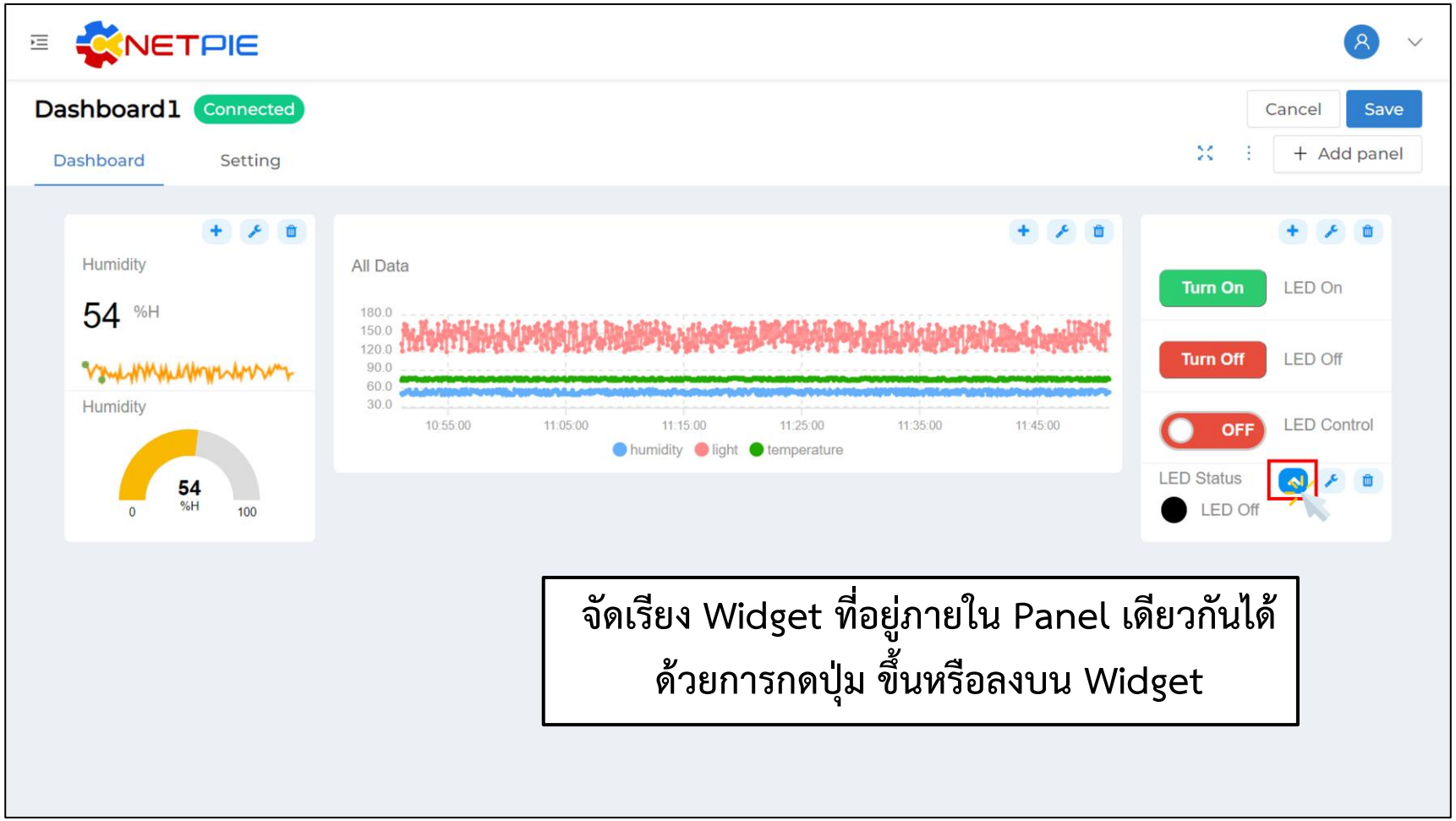
การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม



ได้ Indicator Light ที่แสดงสถานะของ LED มาเรียบร้อย

ใบงานที่ 5.3 ขั้นตอนการทดลอง

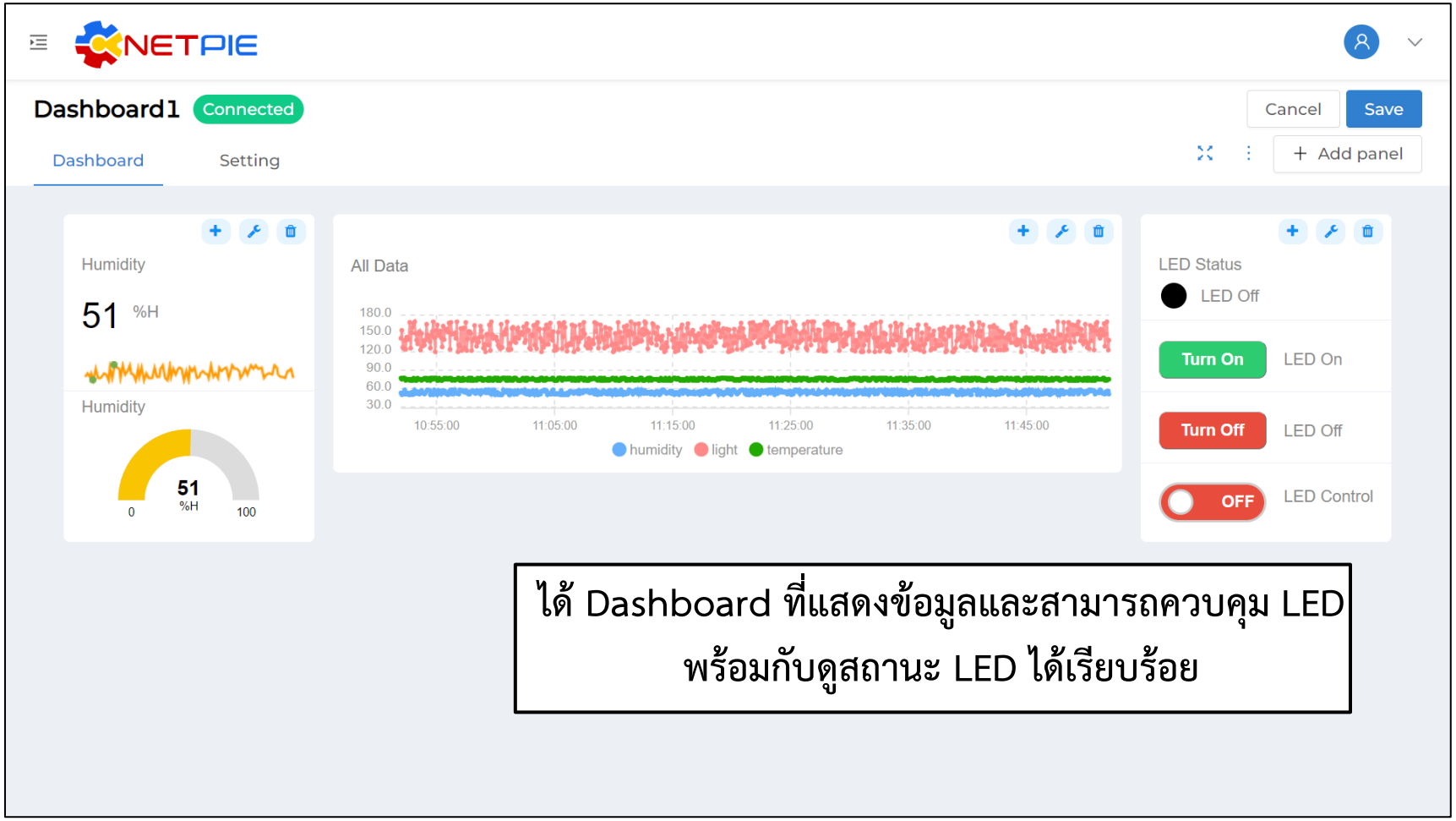
การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม



จัดเรียง Widget ที่อยู่ภายใน Panel เดียวกันได้
ด้วยการกดปุ่ม ขึ้นหรือลงบน Widget

ใบงานที่ 5.3 ขั้นตอนการทดลอง

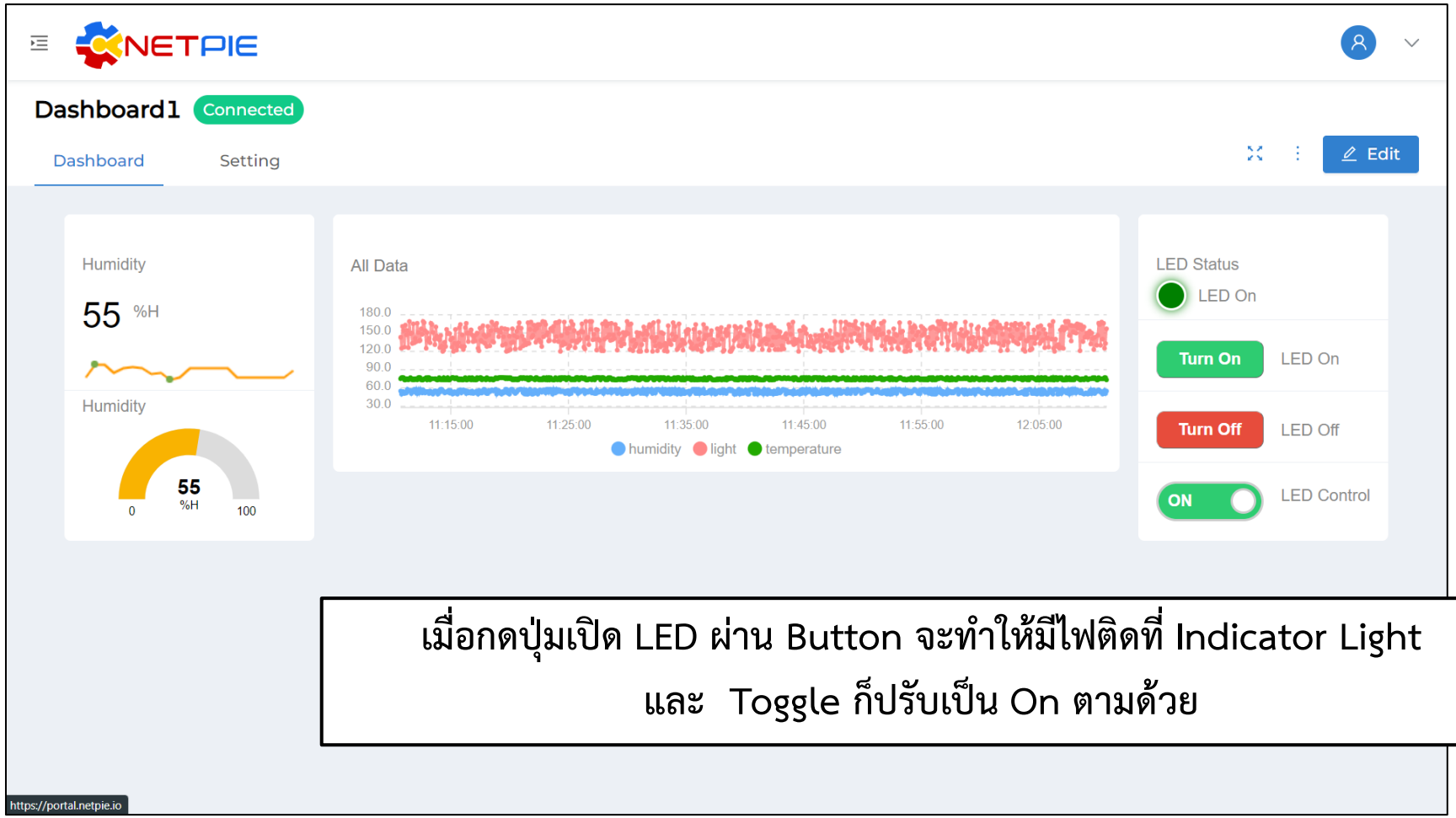
การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม



ได้ Dashboard ที่แสดงข้อมูลและสามารถควบคุม LED พร้อมกับดูสถานะ LED ได้เรียบร้อย

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม



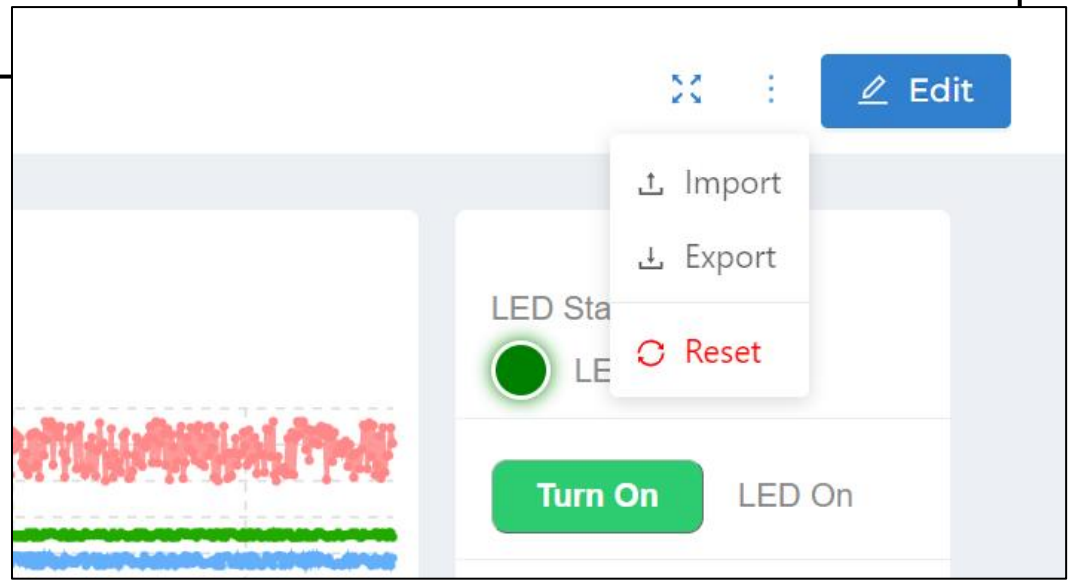
เมื่อกดปุ่มเปิด LED ผ่าน Button จะทำให้มีไฟติดที่ Indicator Light และ Toggle ก็ปรับเป็น On ตามด้วย

ใบงานที่ 5.3 ทฤษฎีเบื้องต้น

การจัดการ Dashboard

ฟังก์ชันที่เพิ่มขึ้นมาของ NETPIE Dashboard ที่ Freeboard ไม่มี คือ การจัดการ Dashboard ที่ผู้ใช้สามารถเพิ่ม Dashboard จากที่อื่น หรือดาวน์โหลด Dashboard ที่สร้างเองเก็บไว้ได้ ผ่านฟังก์ชัน

1. Import คือ การเลือกไฟล์ .json เพื่ออัพโหลด Dashboard อื่นเข้ามาใน Dashboard นี้
2. Export คือ การดาวน์โหลด Dashboard นี้ เก็บเป็นไฟล์ .json
3. Reset คือ การล้าง Dashboard นี้ใหม่เลย



ใบงานที่ 5.3 ขั้นตอนการทดลอง

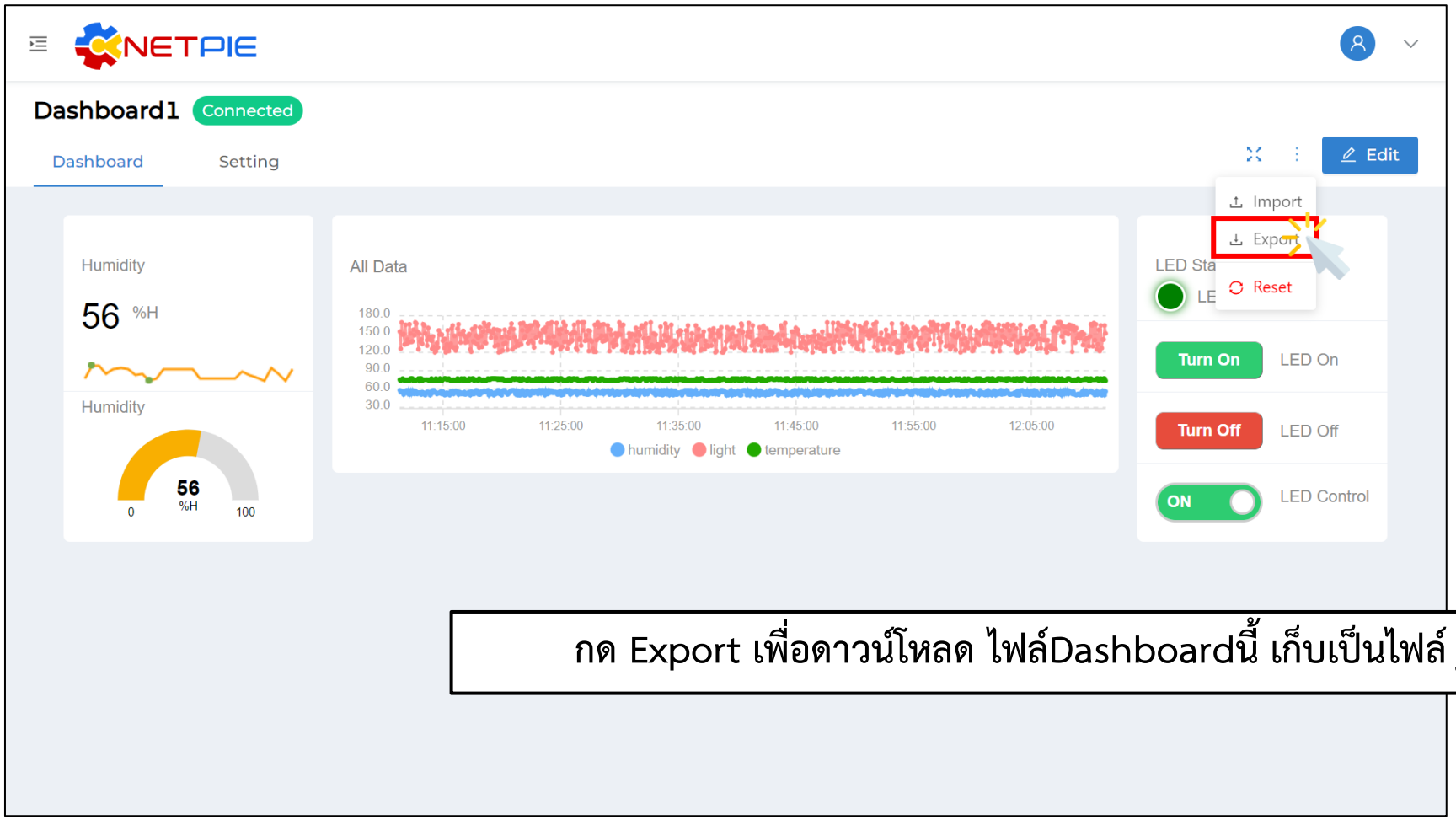
การทดลองที่ 4 การจัดการ Dashboard

ชี้ไปที่จุด 3 จุด ด้านข้างปุ่ม Edit จะมี Import, Export และ Reset ขึ้นมา

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

Export

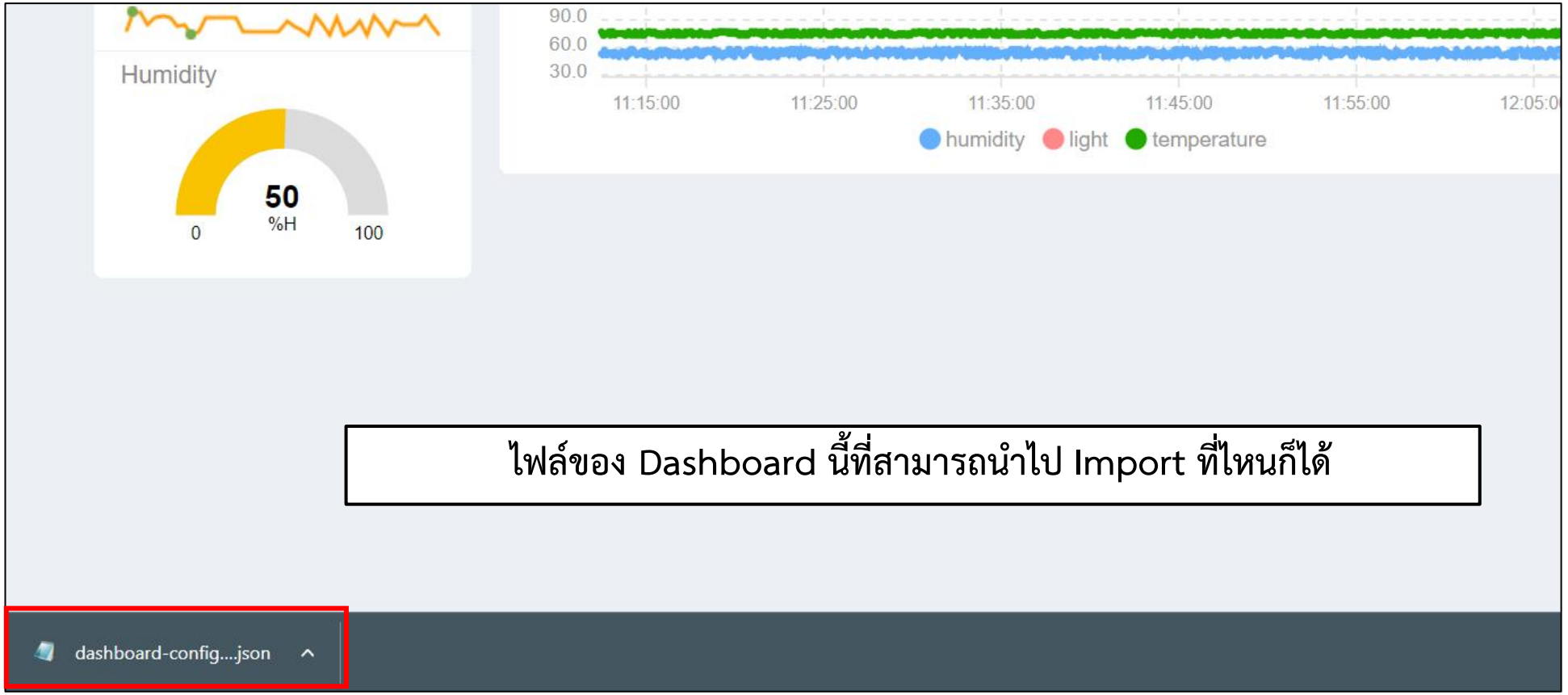


กด Export เพื่อดาวน์โหลด ไฟล์Dashboardนี้ เก็บเป็นไฟล์ json

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

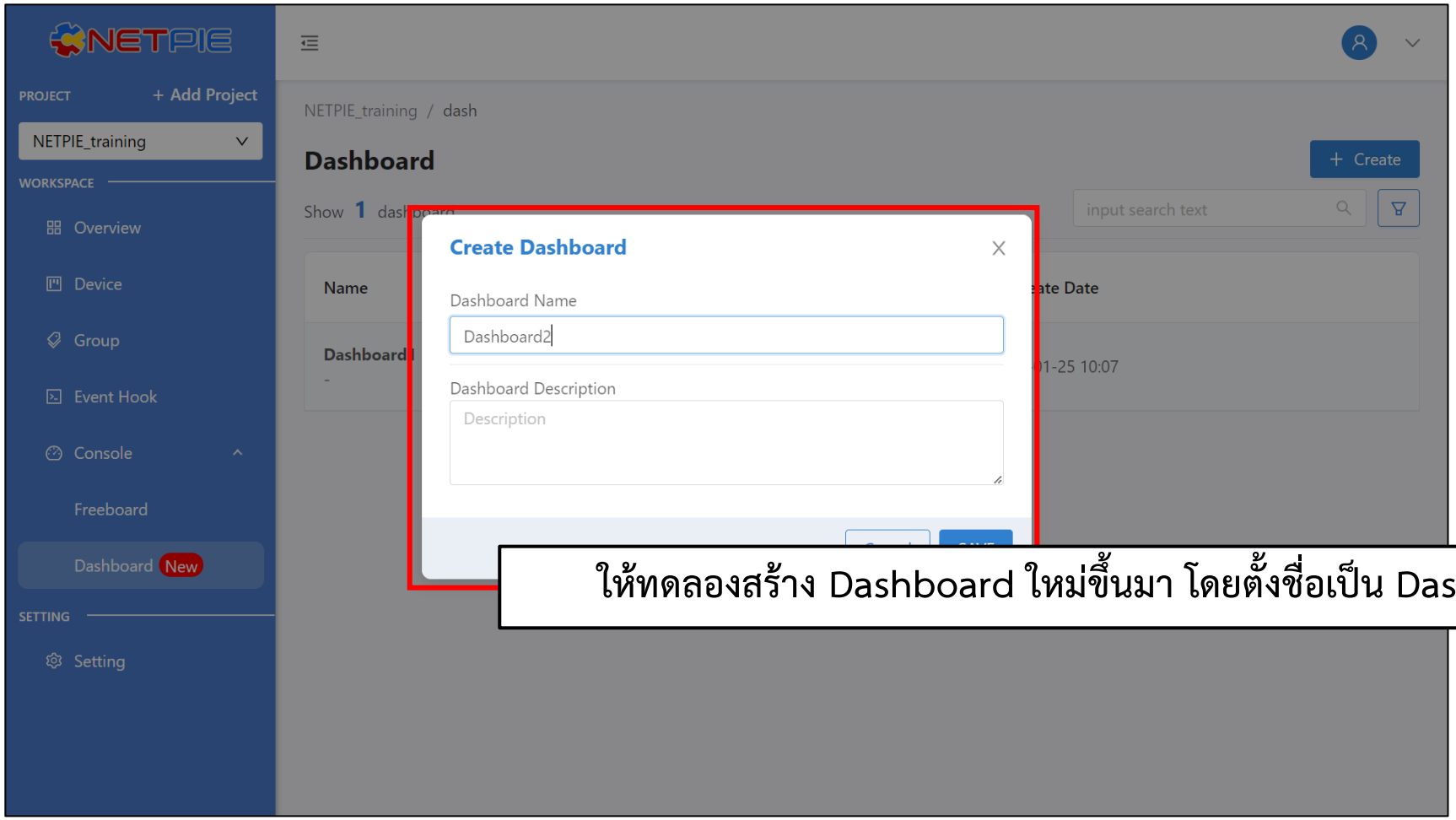
Export



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

Import

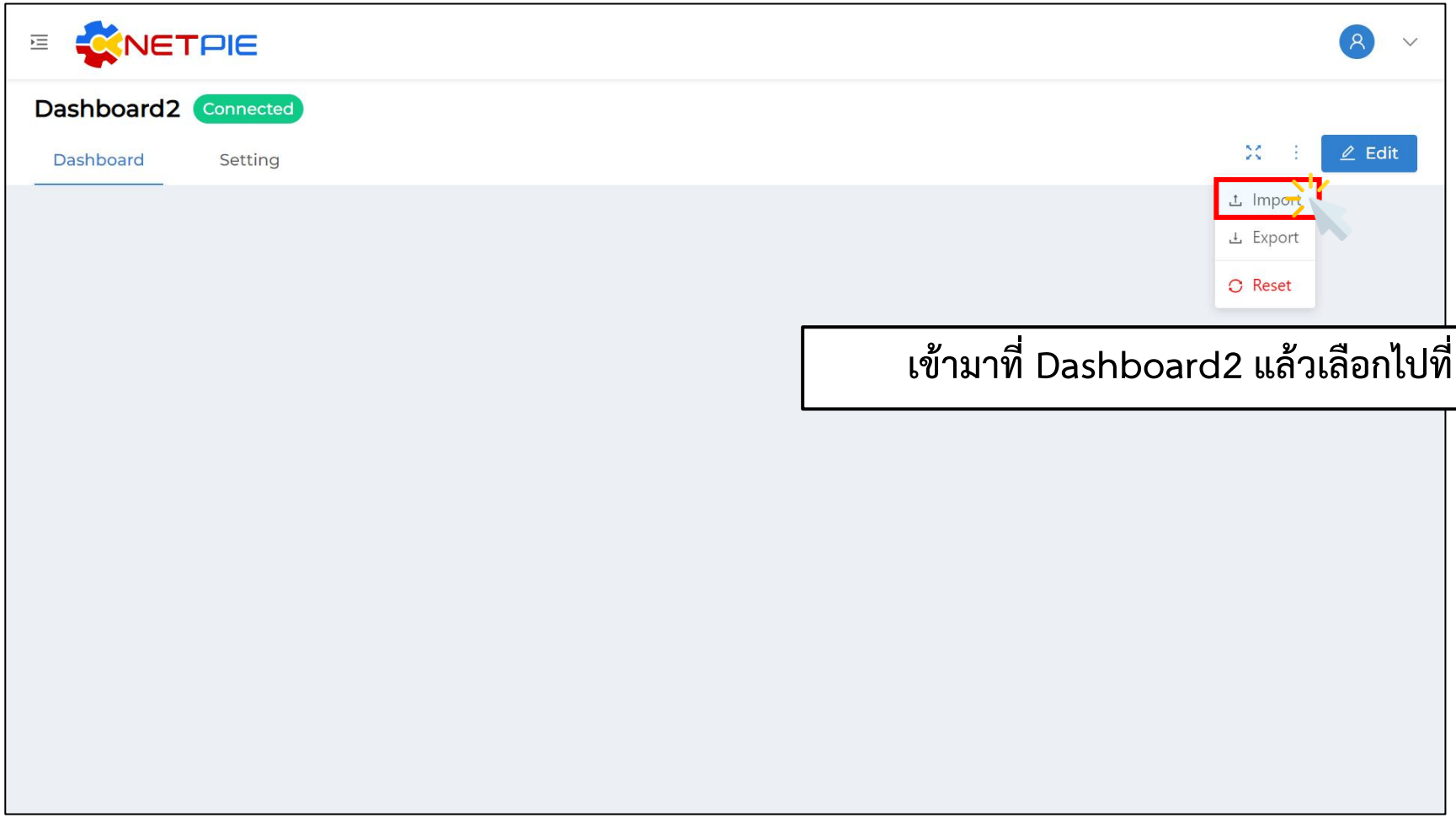


ให้ทดลองสร้าง Dashboard ใหม่ขึ้นมา โดยตั้งชื่อเป็น Dashboard2

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

Import

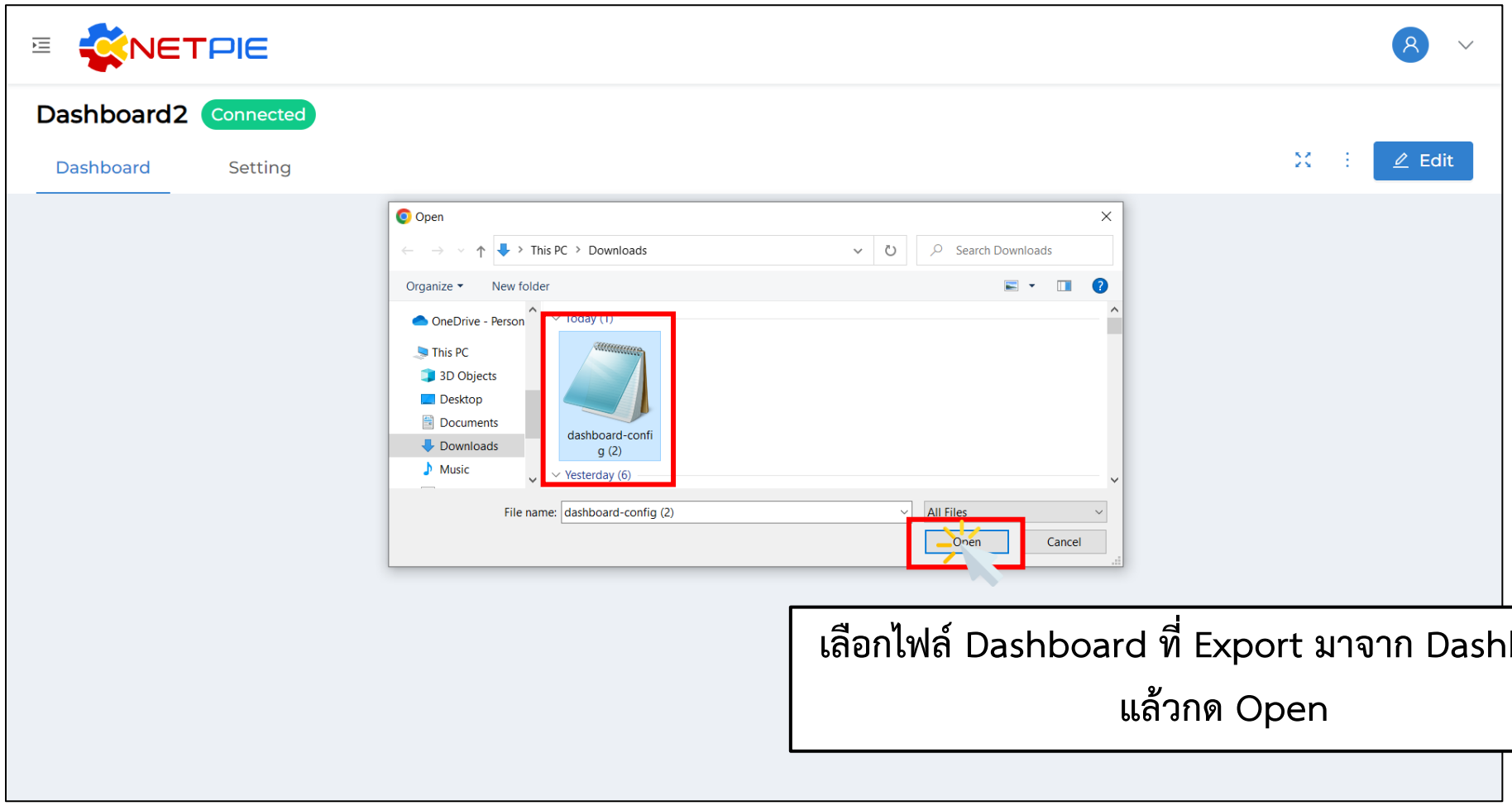


เข้ามาที่ Dashboard2 แล้วเลือกไปที่ Import

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

Import

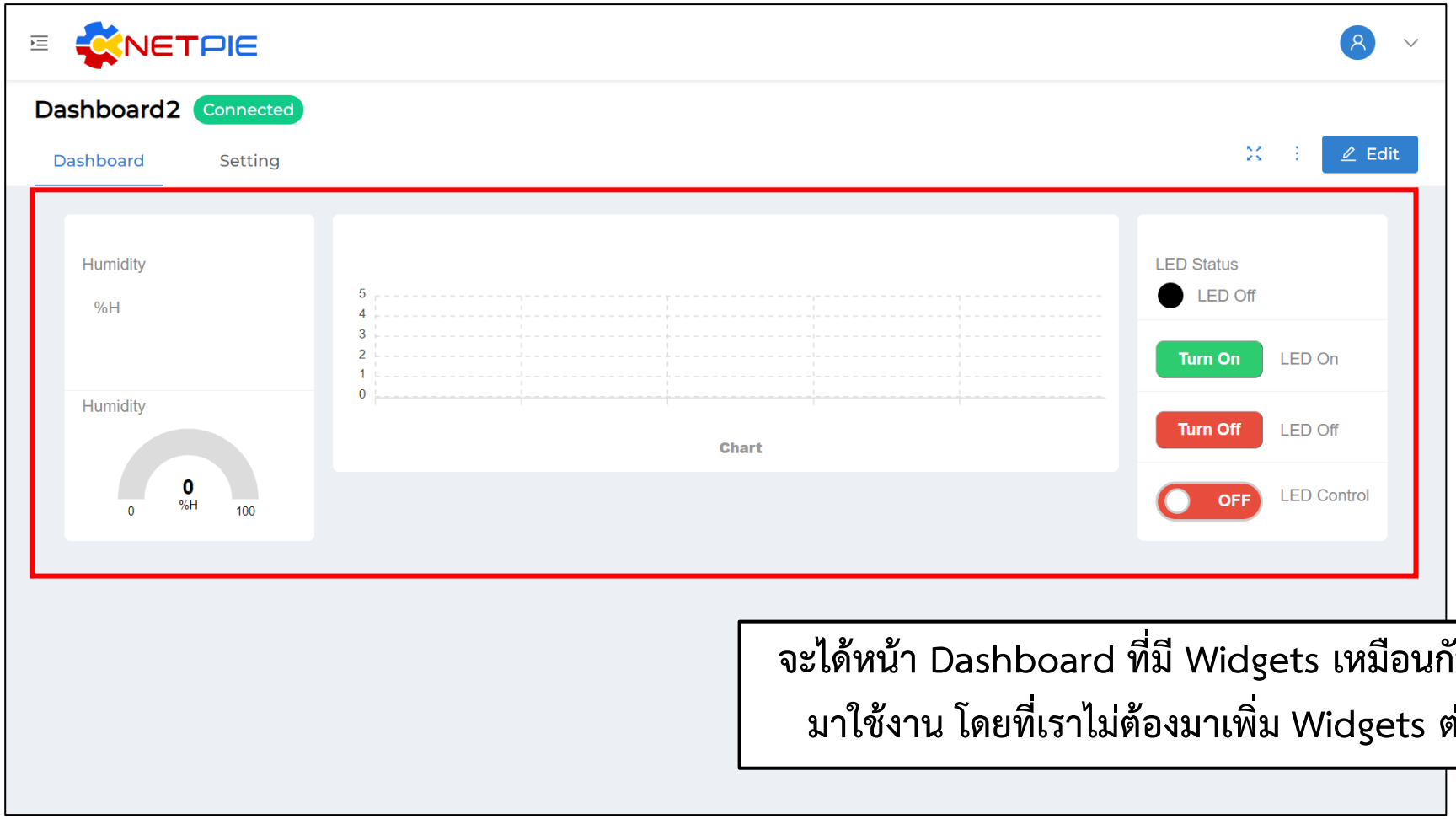


เลือกไฟล์ Dashboard ที่ Export มาจาก Dashboard1 แล้วกด Open

ใบงานที่ 5.3 ขั้นตอนการทดลอง

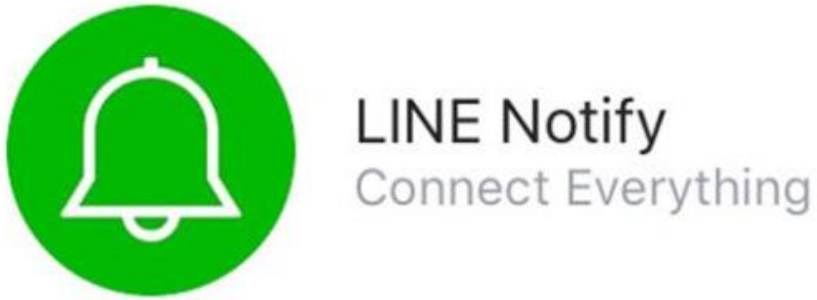
การทดลองที่ 4 การจัดการ Dashboard

Import



จะได้หน้า Dashboard ที่มี Widgets เหมือนกับ Dashboard1 มาใช้งาน โดยที่เราไม่ต้องมาเพิ่ม Widgets ต่างๆ ขึ้นมาเอง

ใบงานที่ 5.4 การสร้างการแจ้งเตือนผ่าน Line Notify



ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

Device Trigger and Event Hook

Device Trigger

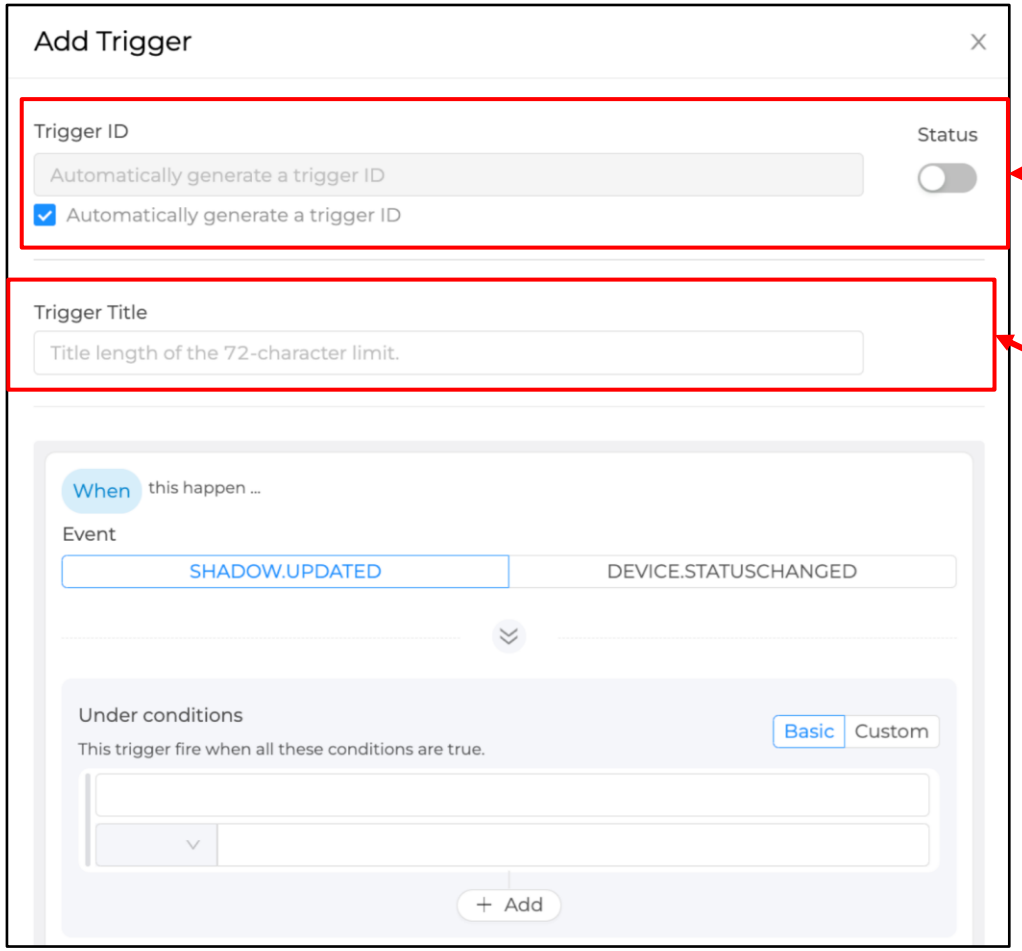
เป็นระบบที่ผูกกับการเปลี่ยนแปลงข้อมูลของ Device Shadow เข้ากับการกระทำภายนอก (Event Hook) เช่น การตั้งค่าแจ้งเตือนตามสถานะต่างๆ ตามเงื่อนไขการทำงานของ Device ที่ถูกตั้งค่าไว้

Event Hook

เป็นตัวกลางที่ใช้กำหนดว่าเมื่อเกิด Trigger จะให้ดำเนินการอะไร ผ่าน 3rd Party หรือระบบอื่นๆที่ต้องการใช้ เช่น การแจ้งเตือนตามสถานะต่างๆผ่าน Line Notify

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

ส่วนต่างๆของ Device Trigger



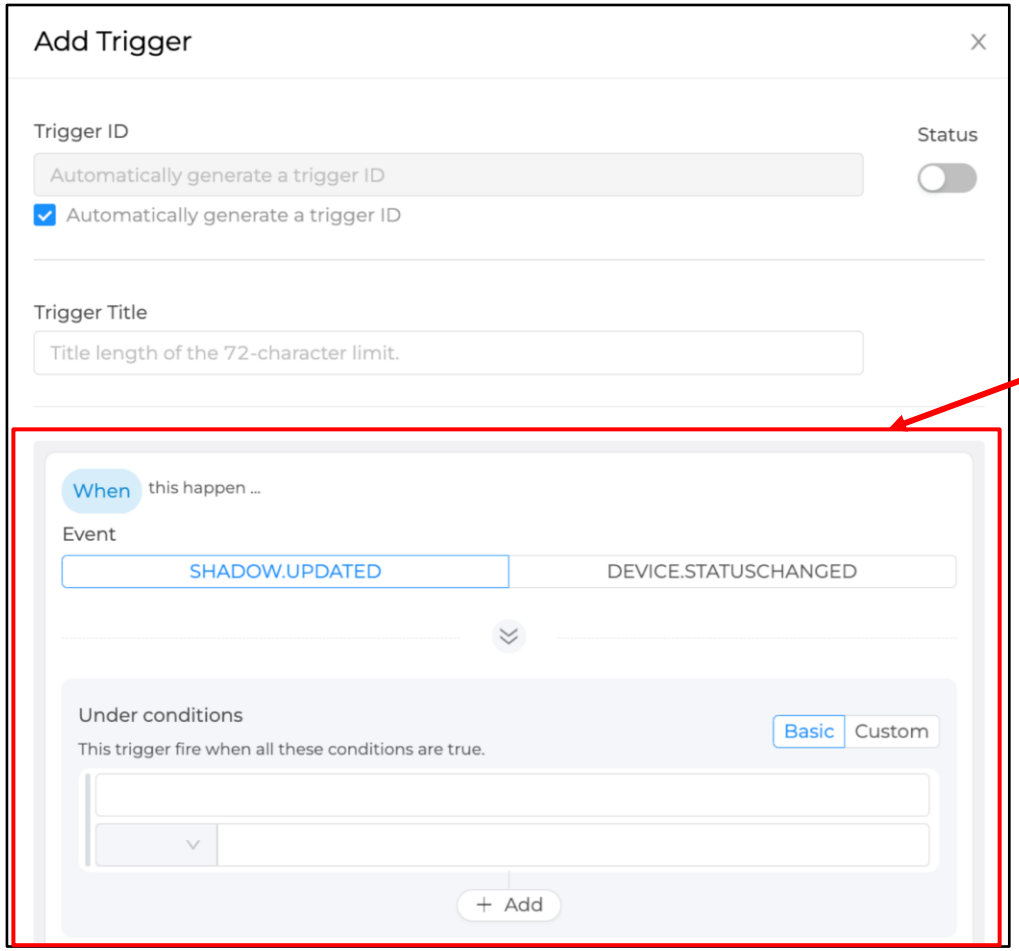
Trigger ID (string) : รหัสของ Trigger ซึ่งระบบจะสร้างให้อัตโนมัติหรือผู้ใช้ต้องการกำหนดเองก็ได้

Status : สถานะเปิด/ปิดการใช้งาน Trigger

Trigger Title (string) : ชื่อหรือคำอธิบายสั้น ๆ เกี่ยวกับ Trigger

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

ส่วนต่างๆของ Device Trigger



Event : ประเภทการเปลี่ยนแปลงข้อมูลของ Device (Device Shadow)
มี 2 ประเภท คือ

1. SHADOW.UPDATED จะเกิด Trigger เมื่อ Device Shadow Data มีการเปลี่ยนแปลงตรงตามเงื่อนไข (Under conditions) ที่กำหนดไว้

2. DEVICE.STATUSCHANGED จะเกิด Trigger เมื่อ Device เปลี่ยนสถานะการเชื่อมต่อ Platform (Online/Offline) และตรงตามเงื่อนไข (Under conditions) ที่กำหนดไว้ ซึ่งการกำหนดเงื่อนไขสำหรับ Trigger Event นี้มีได้ 3 รูปแบบ ดังนี้

1. ต้องการให้ Trigger ทุกครั้งที่สถานะการเชื่อมต่อ Platform เปลี่ยนไม่ว่าจะ Online เป็น Offline หรือ Offline เป็น Online ให้เซ็ตเงื่อนไข (Under conditions) ให้เป็นจริงเสมอ เช่น true == true หรือ 1 == 1 เป็นต้น
2. ต้องการให้ Trigger ในกรณีที่เปลี่ยนสถานะเป็น Online เท่านั้น ให้เซ็ต ให้เซ็ตเงื่อนไข (Under conditions) เป็น $NEW.STATUS == 1
3. ต้องการให้ Trigger ในกรณีที่เปลี่ยนสถานะเป็น Offline เท่านั้น ให้เซ็ต ให้เซ็ตเงื่อนไข (Under conditions) เป็น $NEW.STATUS == 0

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

ส่วนต่างๆของ Device Trigger

Action to : เลือก Event hook ที่ต้องการให้ทำงานต่อเมื่อเกิดการ Trigger โดยรายการใน Dropdown จะได้มาจากการรายการที่ถูกสร้างในเมนู Event Hooks ด้านซ้ายมือ

With this context variables : ประกาศตัวแปรที่จะส่งไปเรียกใช้ใน Event hook โดยทำการประกาศชื่อตัวแปรในช่องฝั่งซ้ายมือ และกำหนดค่าในช่องฝั่งขวามือ ซึ่งค่าที่กำหนดจะเป็นค่าคงที่ ค่าตัวแปรจาก Shadow หรือค่าตัวแปรจากระบบมีให้เรียกใช้ได้ ส่วนการอ้างอิงเพื่อใช้งานที่ Event hook จะใช้เป็น {{context.ชื่อตัวแปร}}

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

การอ้างอิงค่า Shadow ใน Trigger

สำหรับการอ้างอิงค่าตัวแปร Shadow สามารถเรียกใช้ใน Condition หรือ Context Variable ของ Trigger มี 3 รูปแบบ ดังนี้

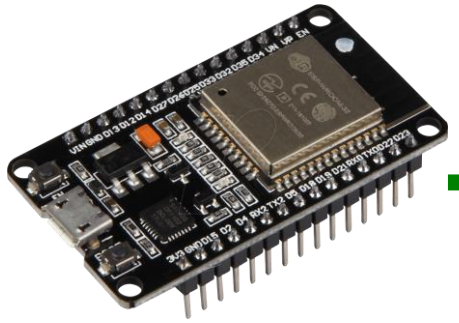
1. **$CUR.พาธของตัวแปร** คือ ค่าปัจจุบันล่าสุดที่ถูกอัพเดท ($NEW merge $PREV) โดยขึ้นต้นด้วย $CUR ตามด้วย Path ตามโครงสร้างใน Shadow
2. **$NEW.พาธของตัวแปร** ค่าใหม่ที่ส่งมาอัพเดทลง Shadow โดยขึ้นต้นด้วย $NEW ตามด้วย Path ตามโครงสร้างใน Shadow
3. **$PREV.พาธของตัวแปร** ค่าก่อนหน้าที่จะถูกอัพเดทลง Shadow โดยขึ้นต้นด้วย $PREV ตามด้วย Path ตามโครงสร้างใน Shadow

ทั้งนี้ Device Trigger มีการอ้างอิงตัวแปรอื่นๆในระบบอีกมากมาย สามารถดูรายละเอียดได้ตามนี้

https://docs.netpie.io/device-config.html#device-trigger-and-event-hook

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

ตัวอย่างการทำงานของ Trigger and Event Hook



ESP8266/ESP32

Publish : @shadow/data/update

Event Hooks

LINE NOTIFY

LINE

NETPIE2020

Device2 Shadow

"temperature" : 74

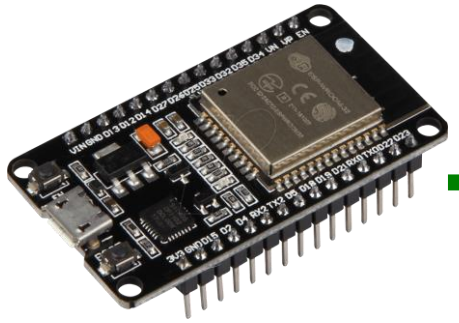
"humidity" : 50

Status : Online

Trigger : DEVICE.STATUSCHANGE

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

ตัวอย่างการทำงานของ Trigger and Event Hook



ESP8266/ESP32

Publish : @shadow/data/update

Event Hooks

LINE NOTIFY

LINE

NETPIE2020

Device2 Shadow

"temperature" : 74

"humidity" : 50

Status : Offline

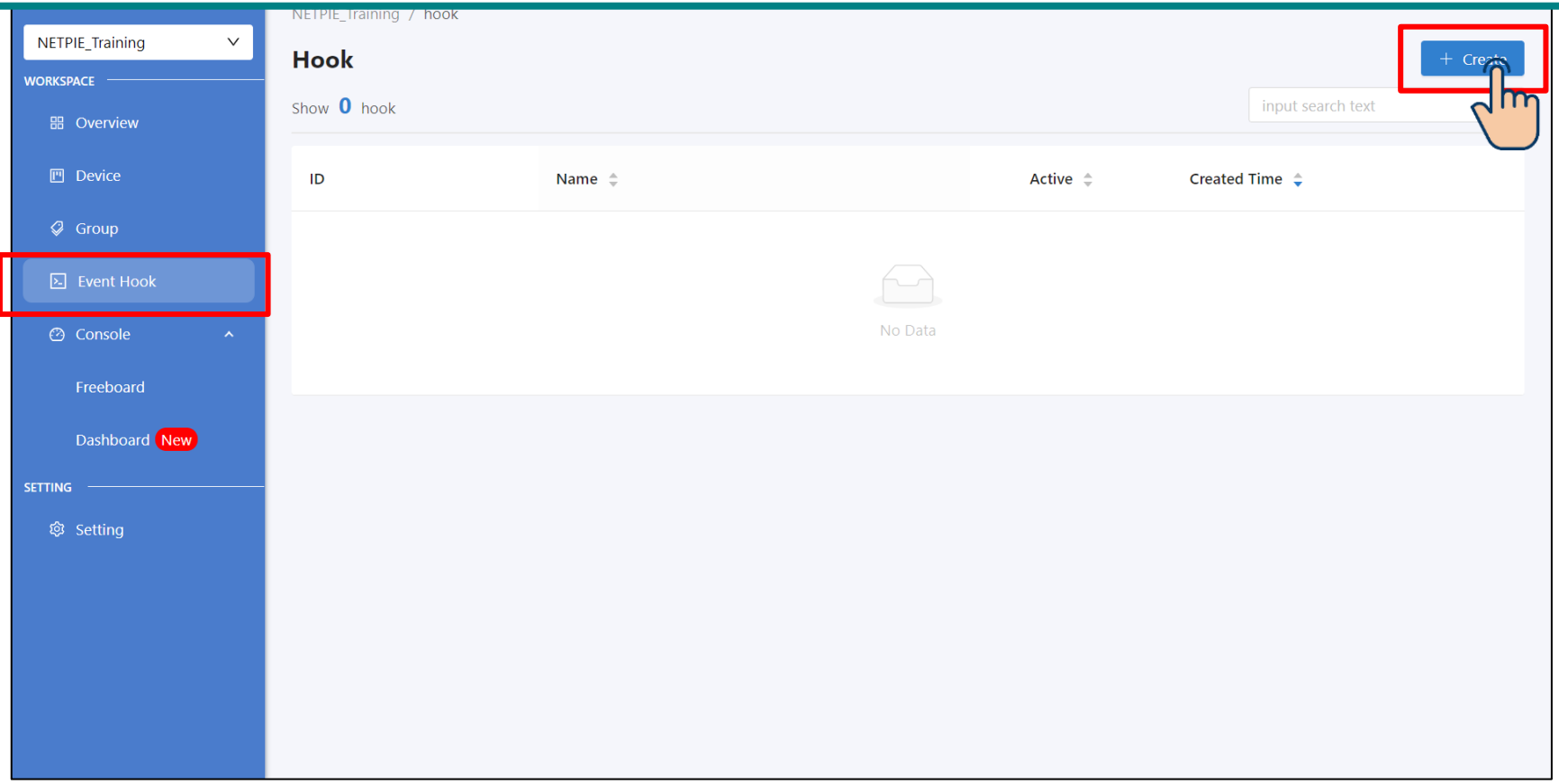
Trigger : DEVICE.STATUSCHANGE

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

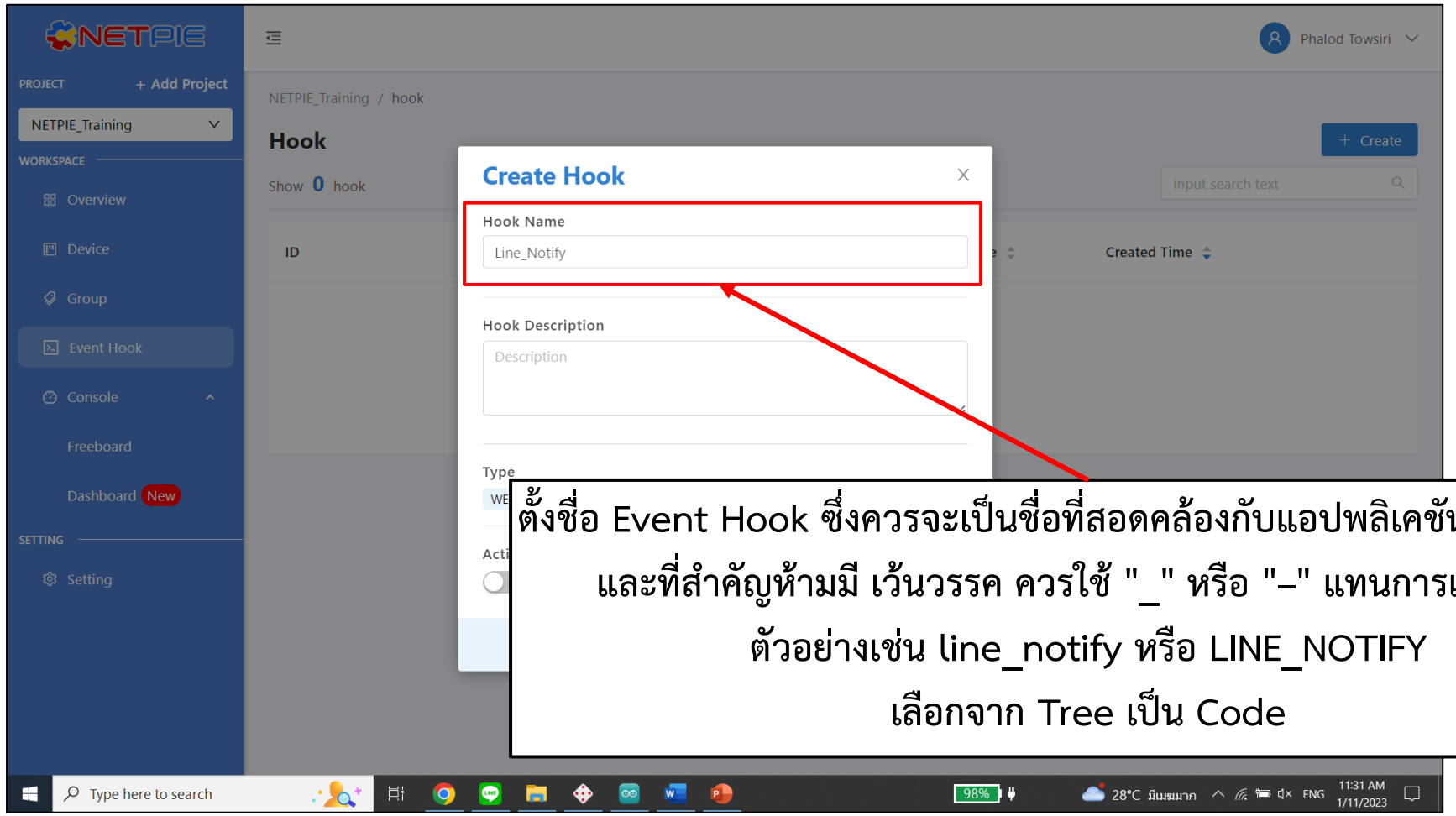
LINE Notify
Connect Everything

เราจะทำการสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify ซึ่งเป็น 3rd Party ที่ได้รับความนิยมสำหรับการทำระบบแจ้งเตือน



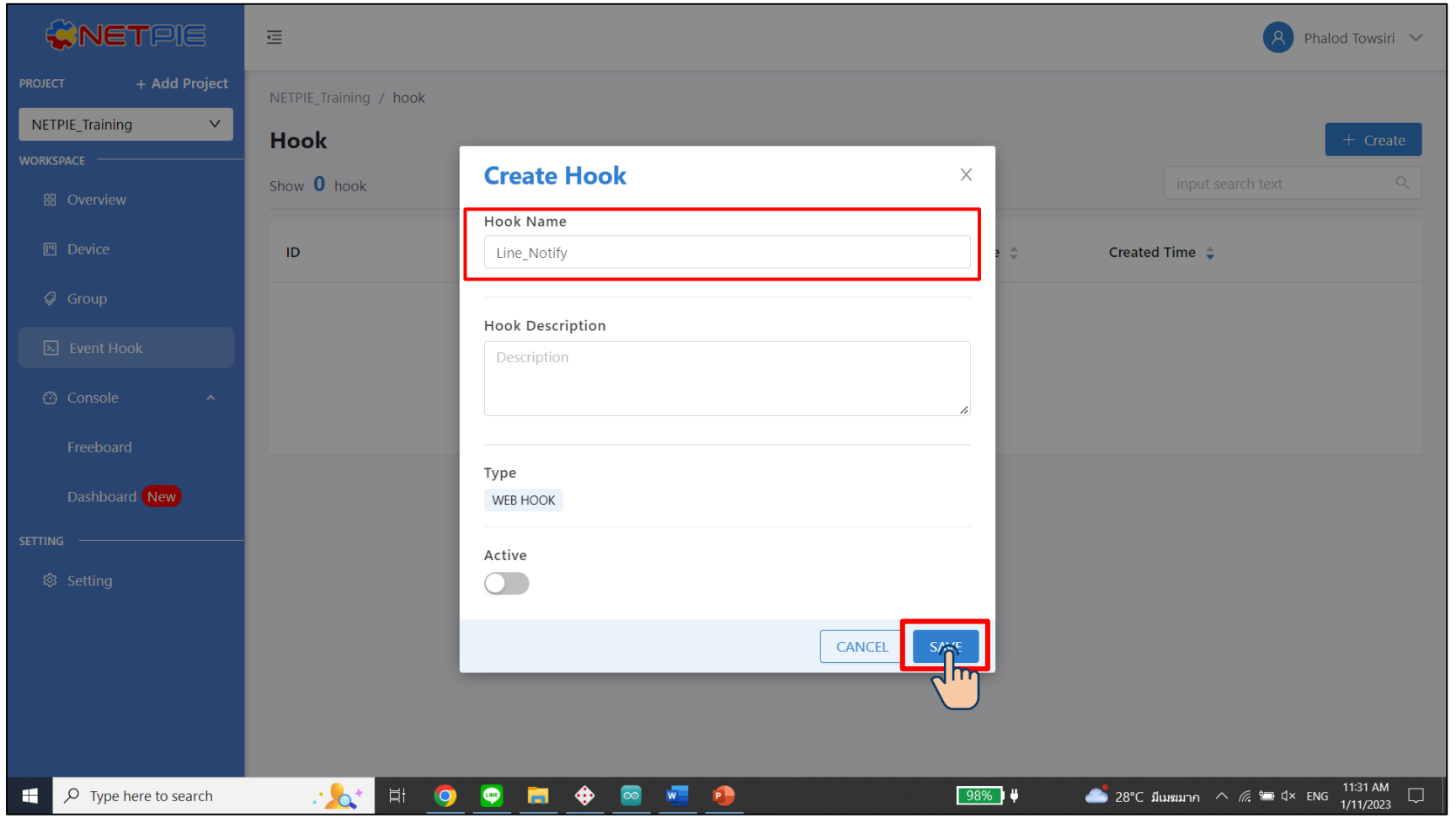
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



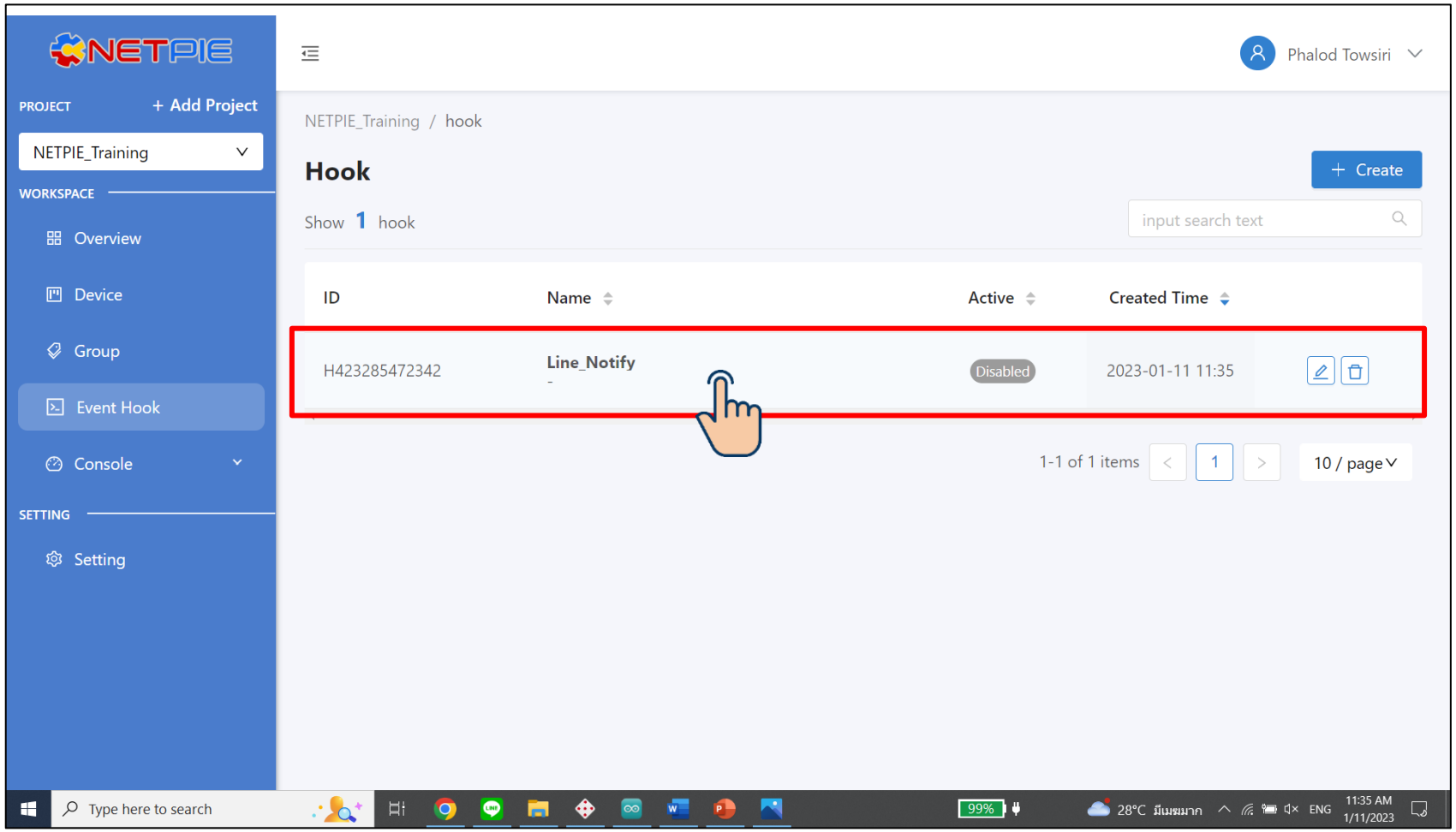
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



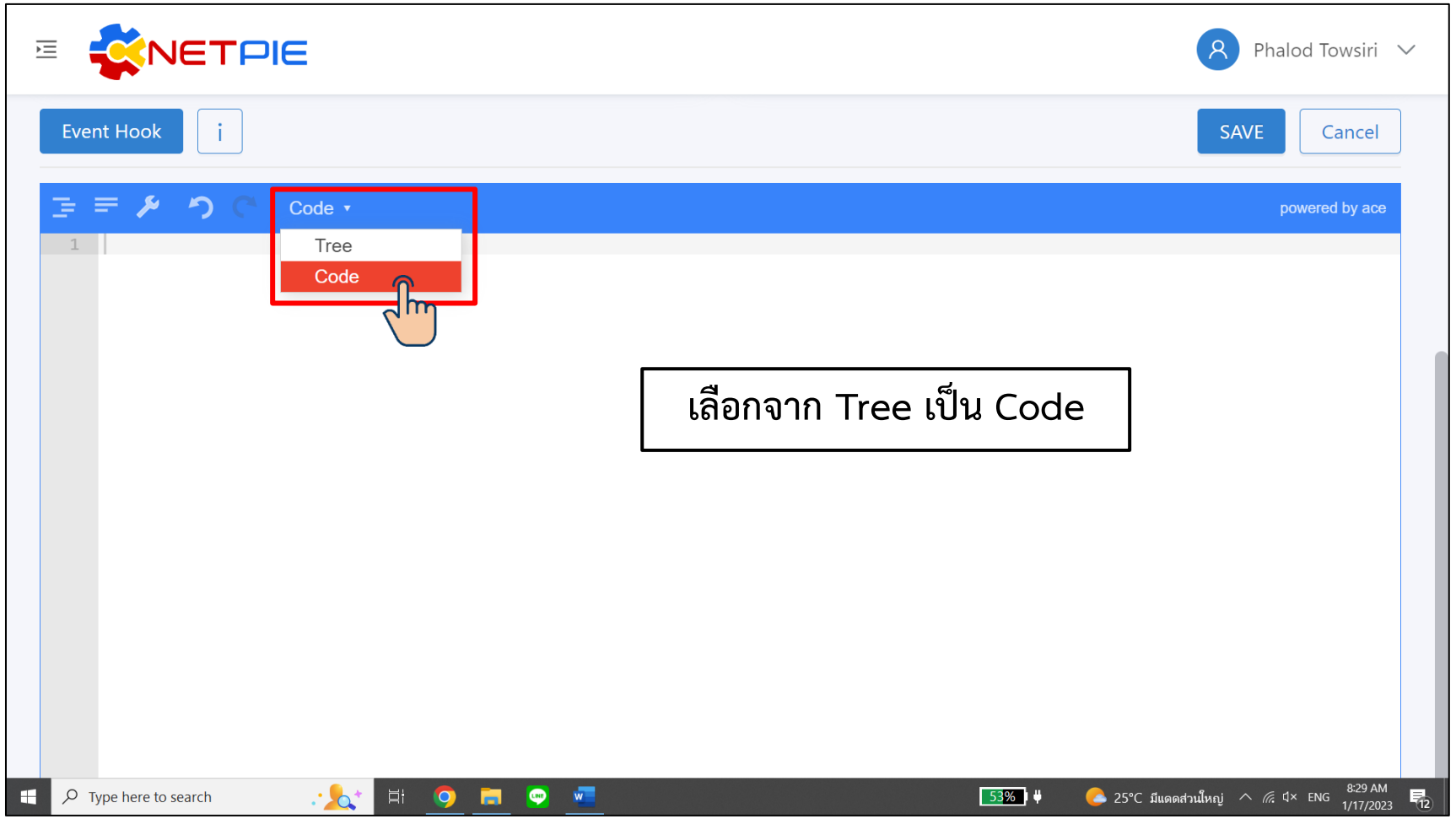
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

Event Hook Example เป็นตัวอย่างการทำ Line Notify ซึ่งกำหนดค่าได้ 4 Attributes

```
{
  "body": "message={{context.msg}}",
  "header": {
    "Authorization": "Bearer {{context.linetoken}}",
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "method": "POST",
  "uri": "https://notify-api.line.me/api/notify"
}
```

คัดลอกทั้งหมดไปไว้ใน Event Hook บน NETPIE2020

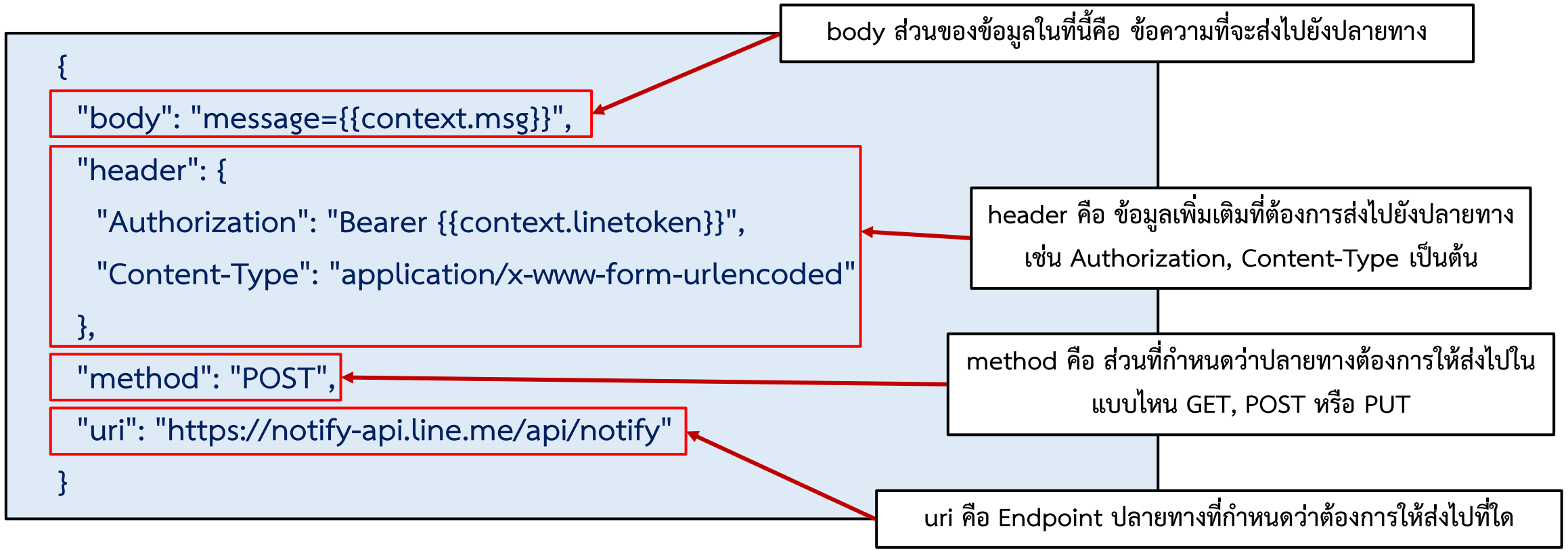
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

ใบงานที่ 5.4 ขั้นตอนการทดลอง

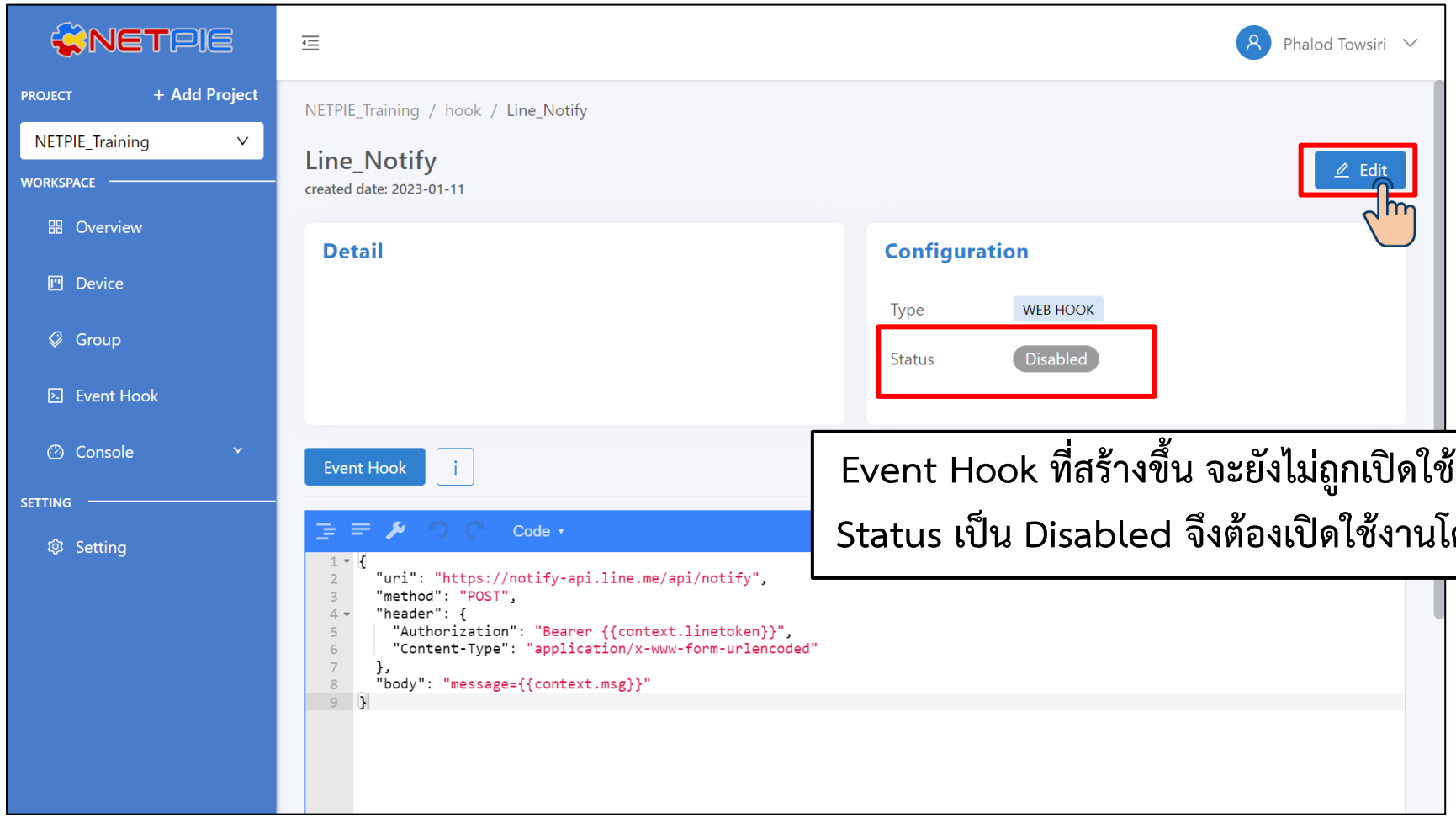
การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

คำอธิบายการใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

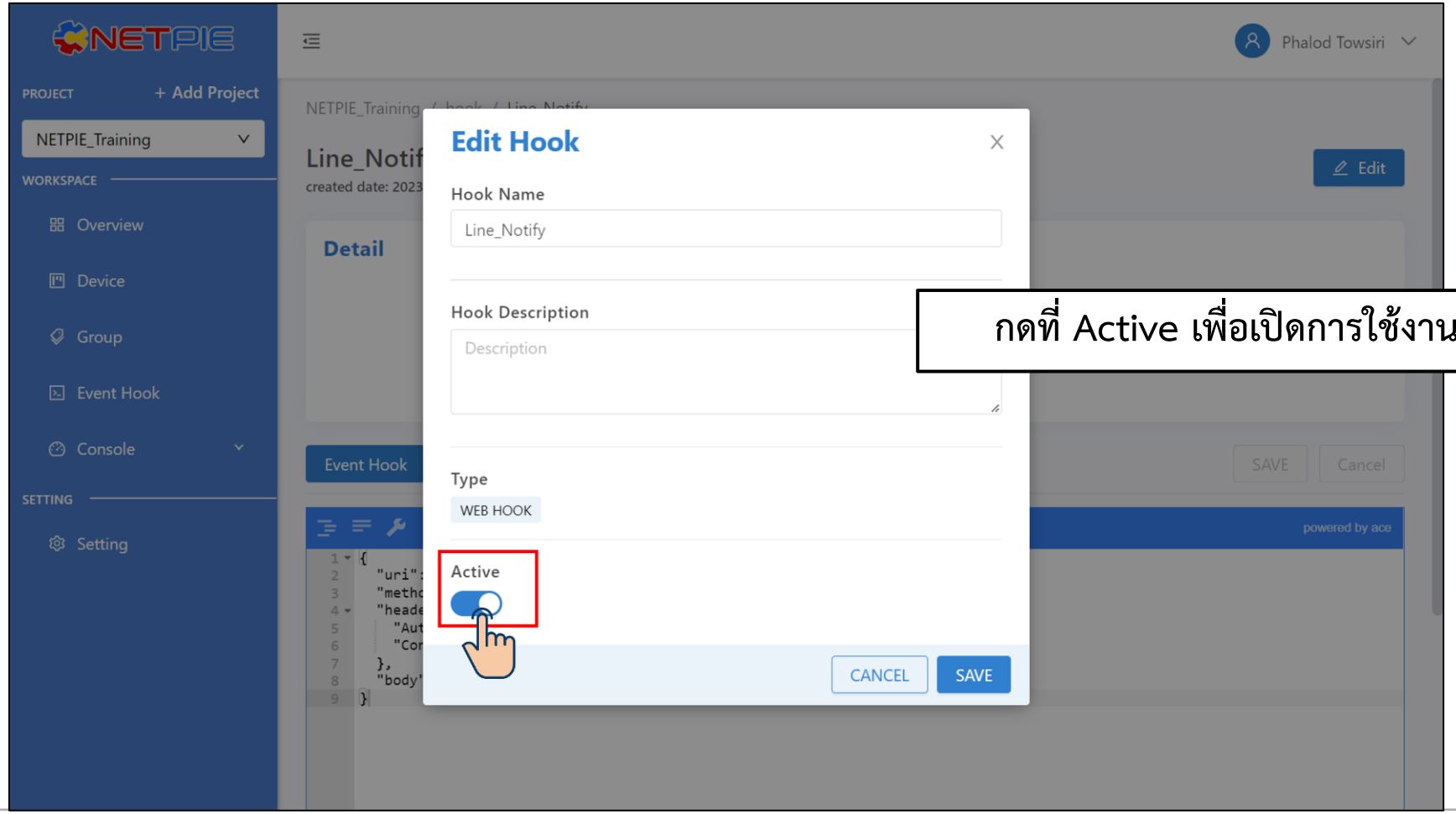
การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



Event Hook ที่สร้างขึ้น จะยังไม่ถูกเปิดใช้งานสังเกตจาก Status เป็น Disabled จึงต้องเปิดใช้งานโดยกดไปที่ Edit

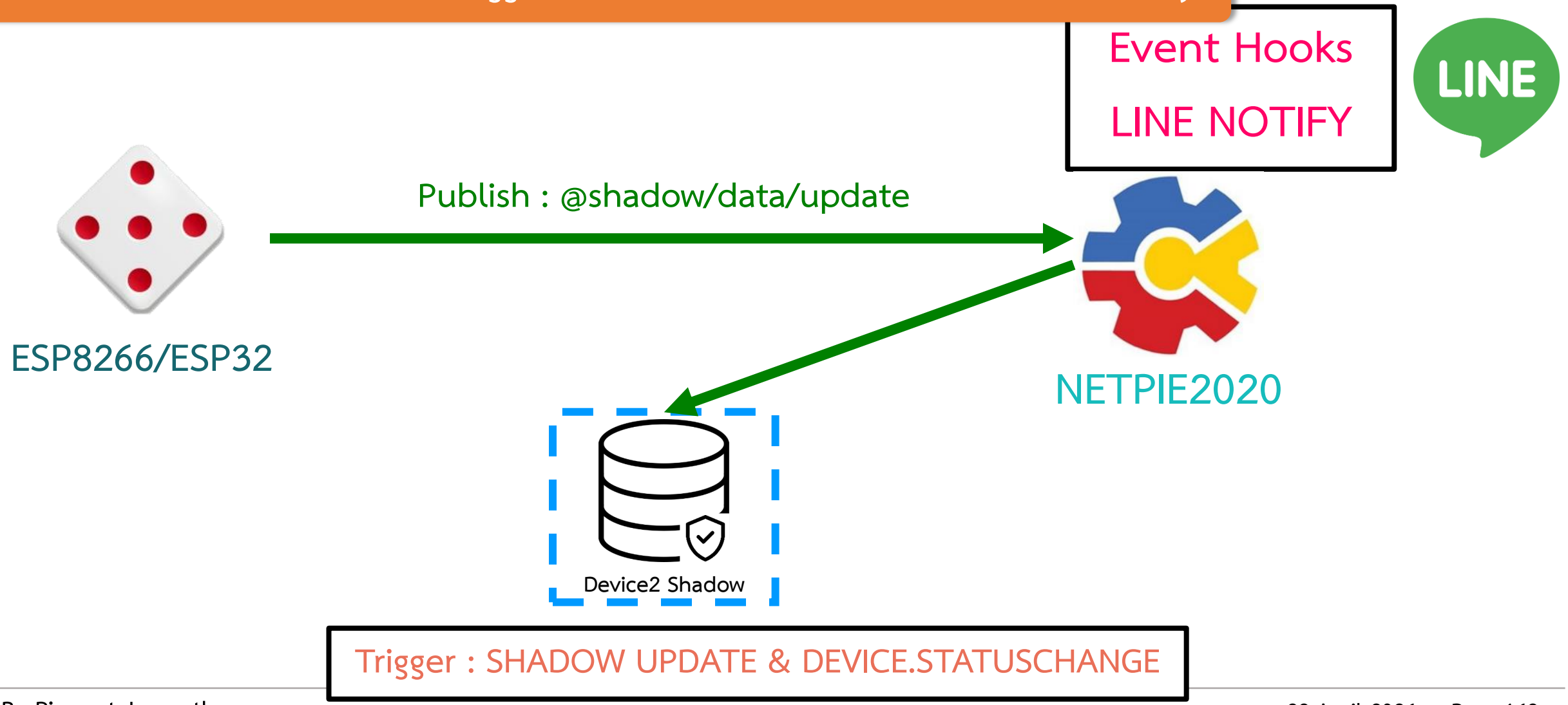
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



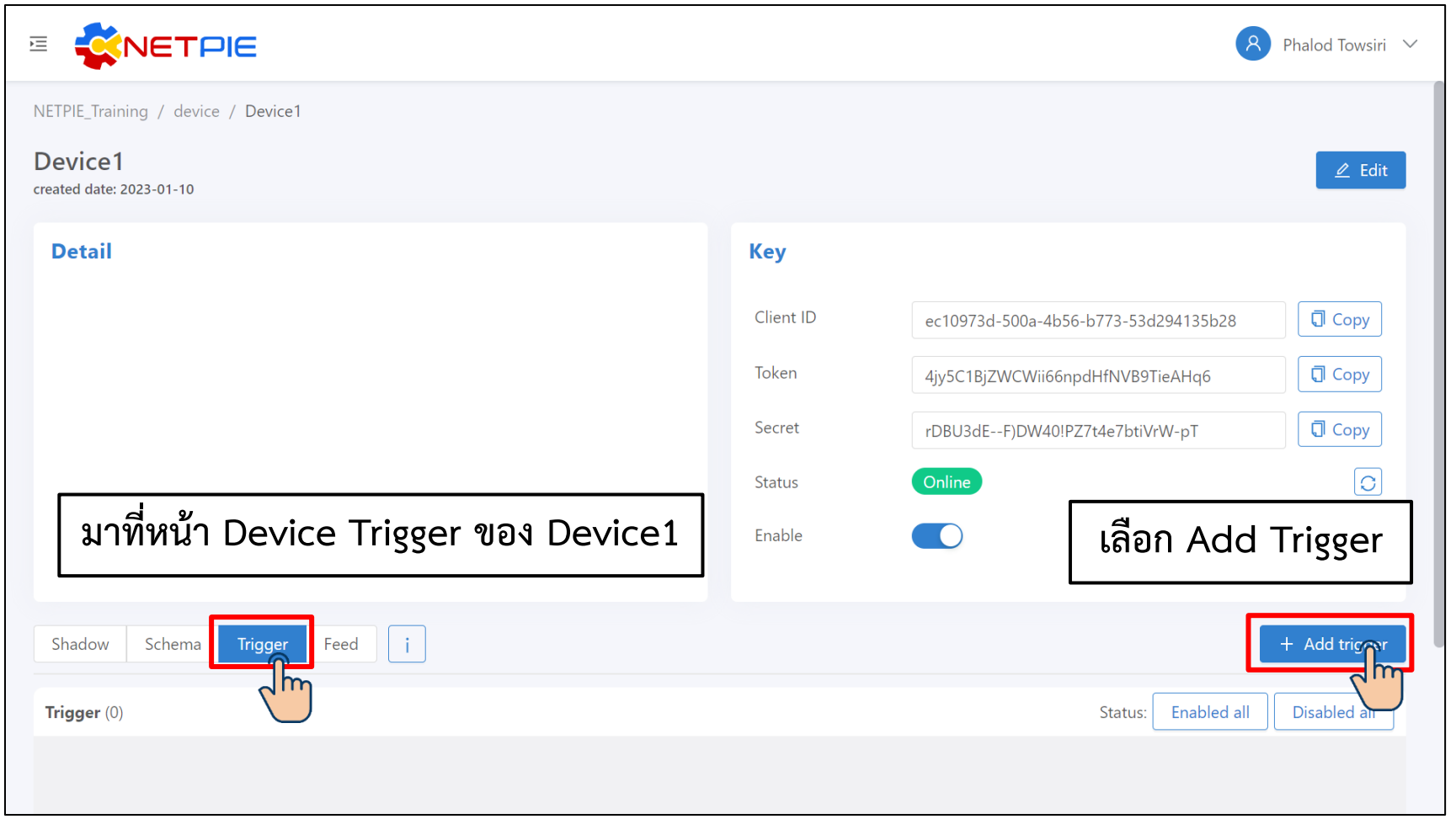
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



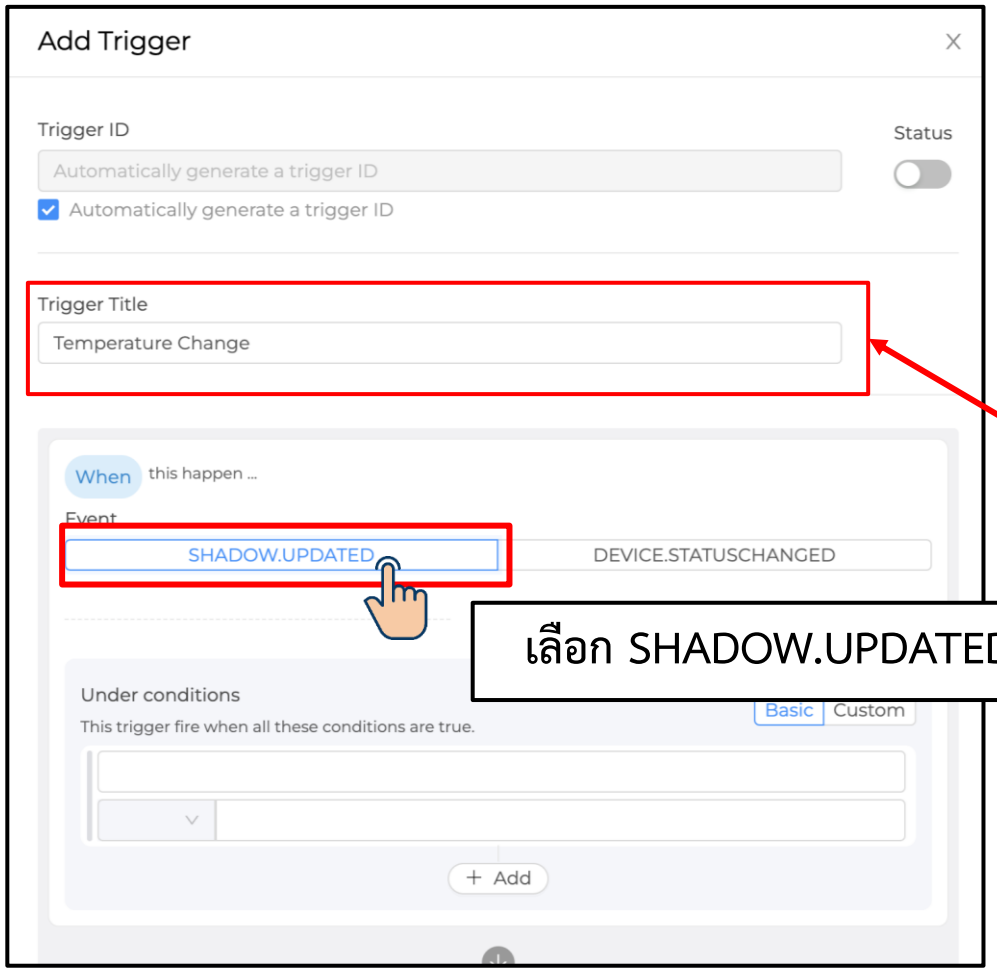
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



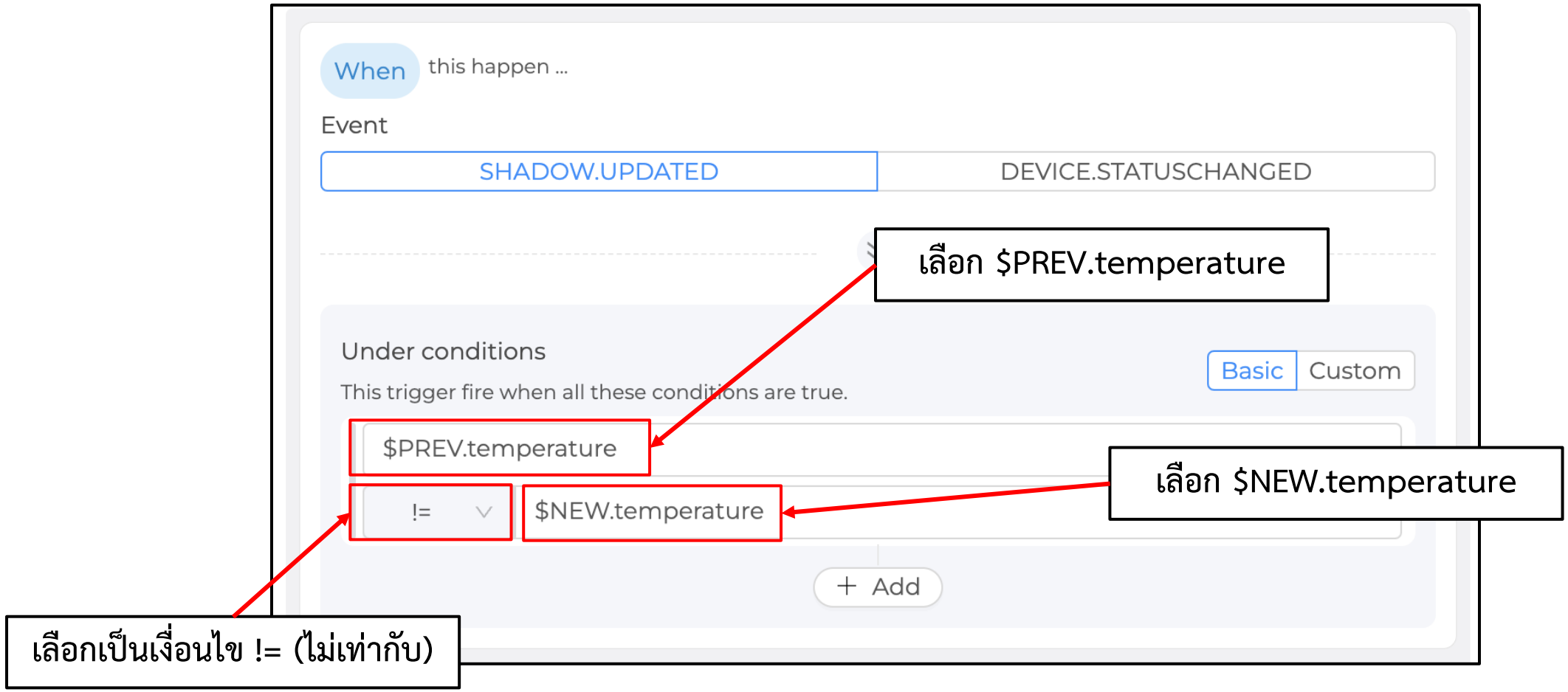
เราจะสร้างเงื่อนไขการแจ้งเตือน คือ เมื่ออุณหภูมิมีการเปลี่ยนแปลง ให้ทำการแจ้งเตือนเข้า Line Notify โดยมีข้อความคือ "Temperature is change from PREV.Temperature °C to NEW.Temperature °C"

ตั้งชื่อ Trigger

เลือก SHADOW.UPDATED

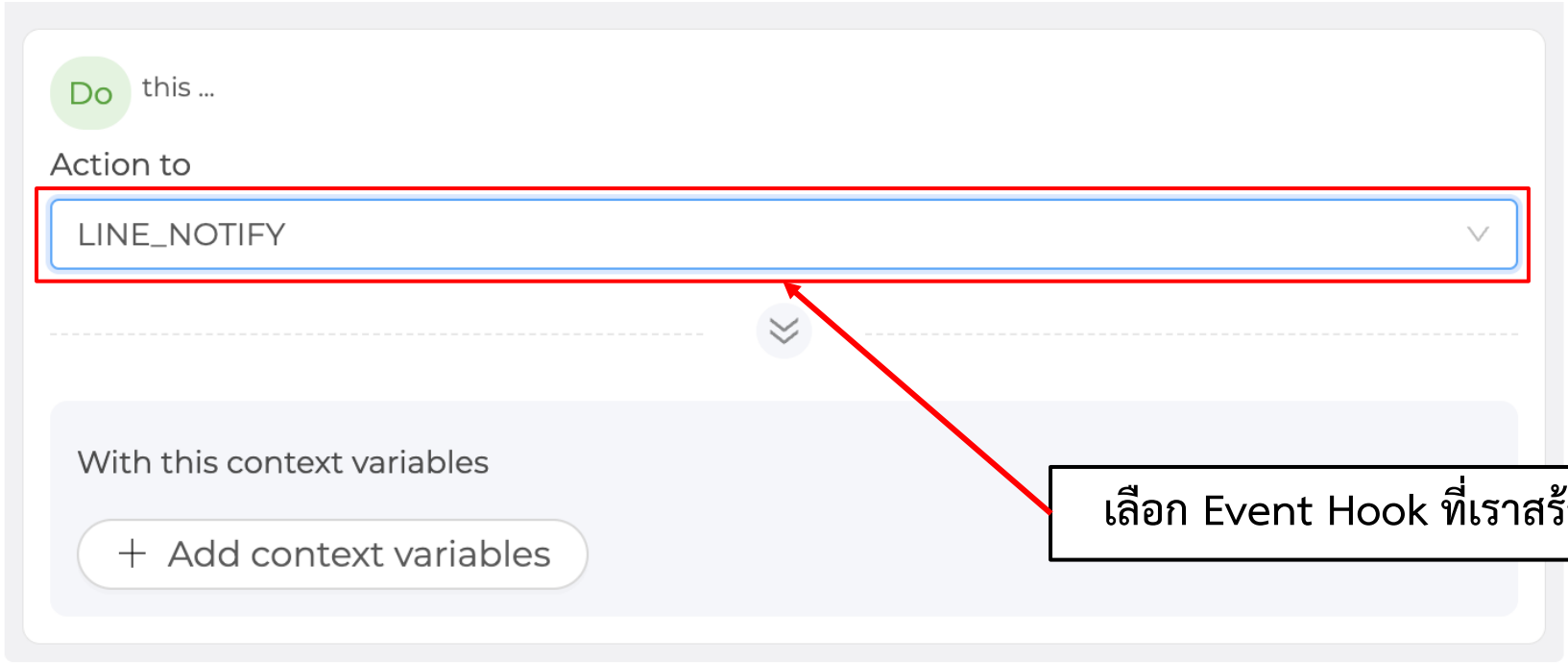
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

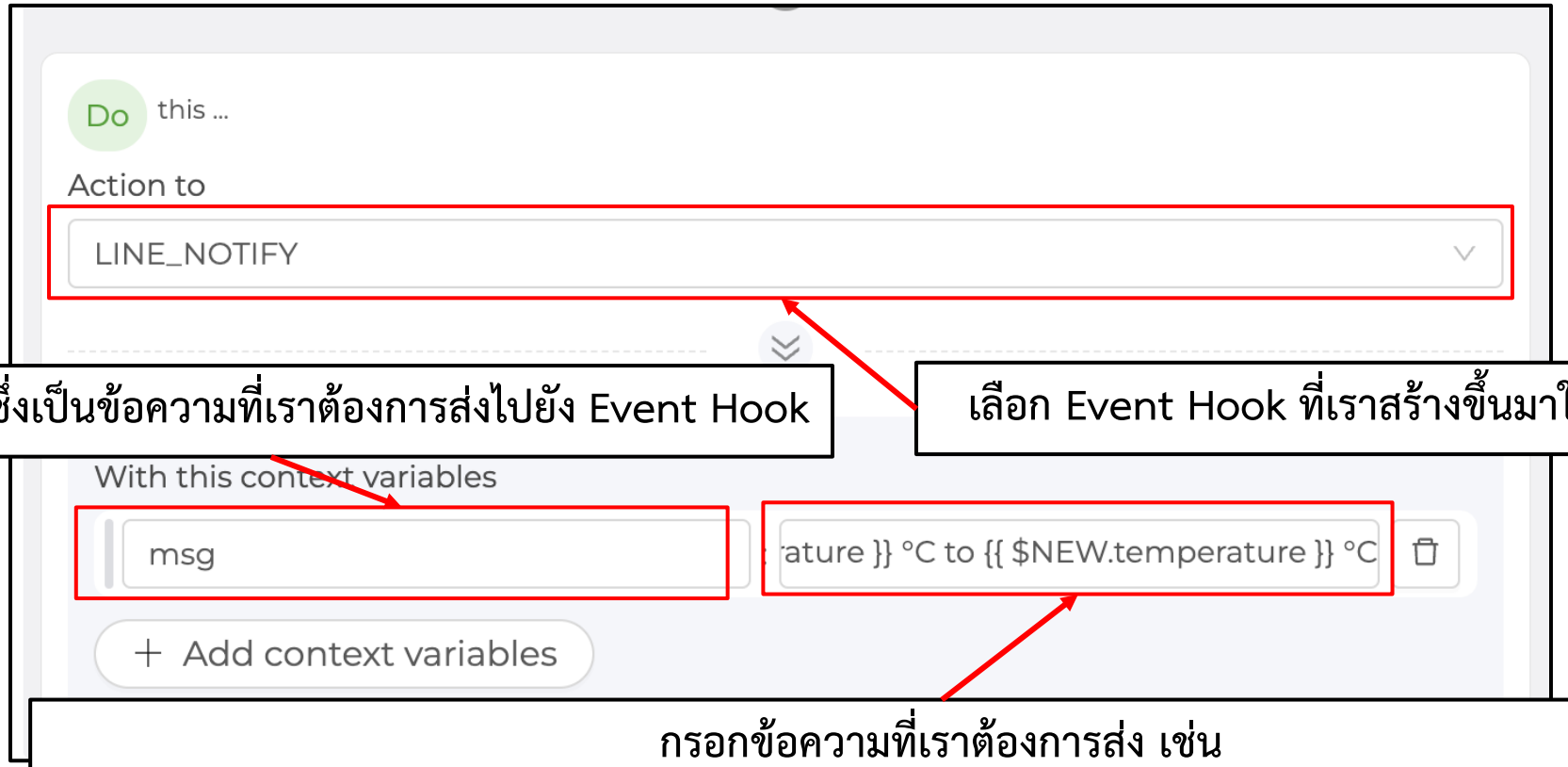
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



เลือก Event Hook ที่เราสร้างขึ้นมาในการทดลองที่ 1

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



เลือกเป็น msg ซึ่งเป็นข้อความที่เราต้องการส่งไปยัง Event Hook

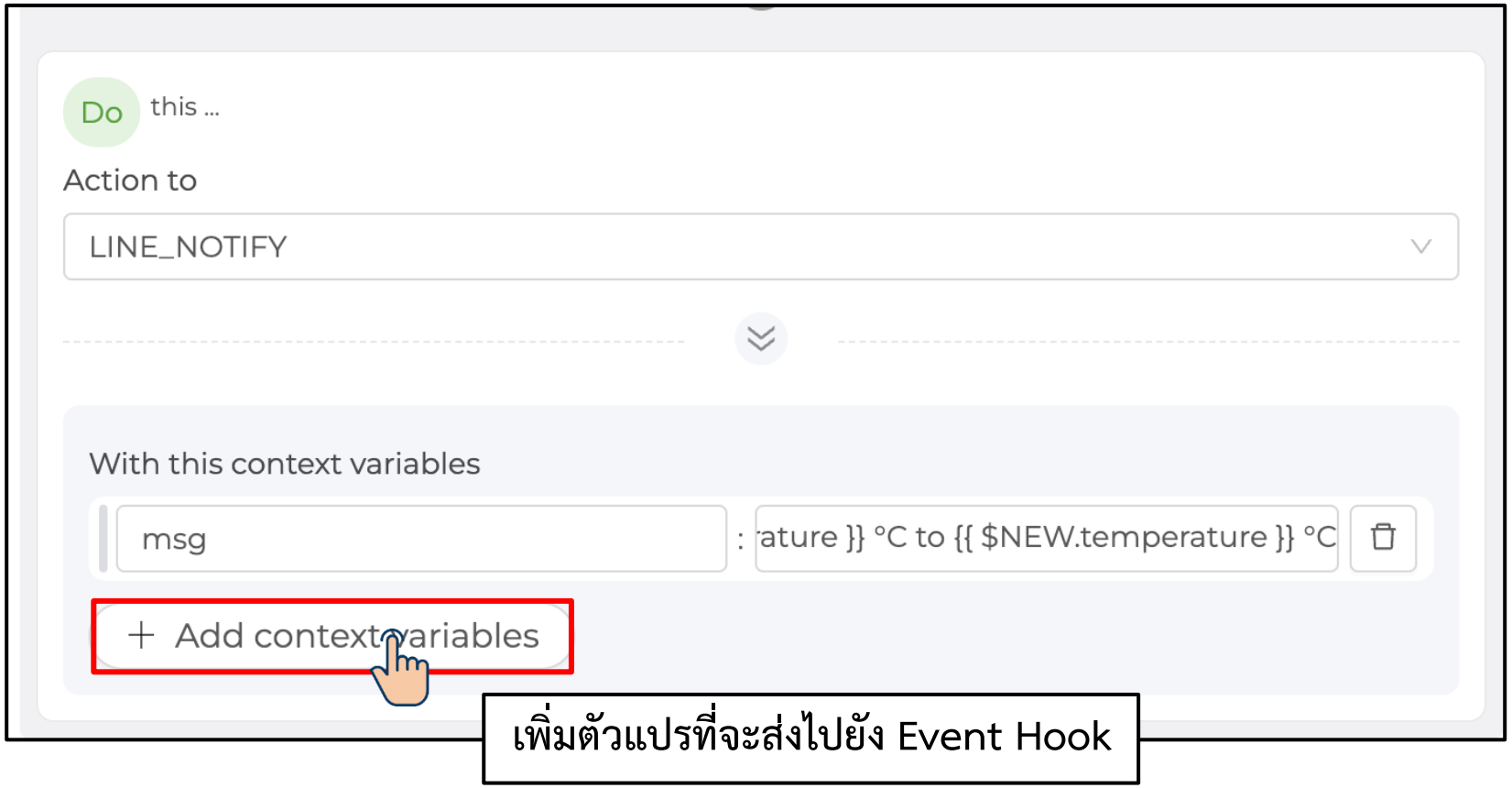
เลือก Event Hook ที่เราสร้างขึ้นมาในการทดลองที่ 1

กรอกข้อความที่เราต้องการส่ง เช่น

Temperature is change from {{ $PREV.temperature }} °C to {{ $NEW.temperature }} °C

ใบงานที่ 5.4 ขั้นตอนการทดลอง

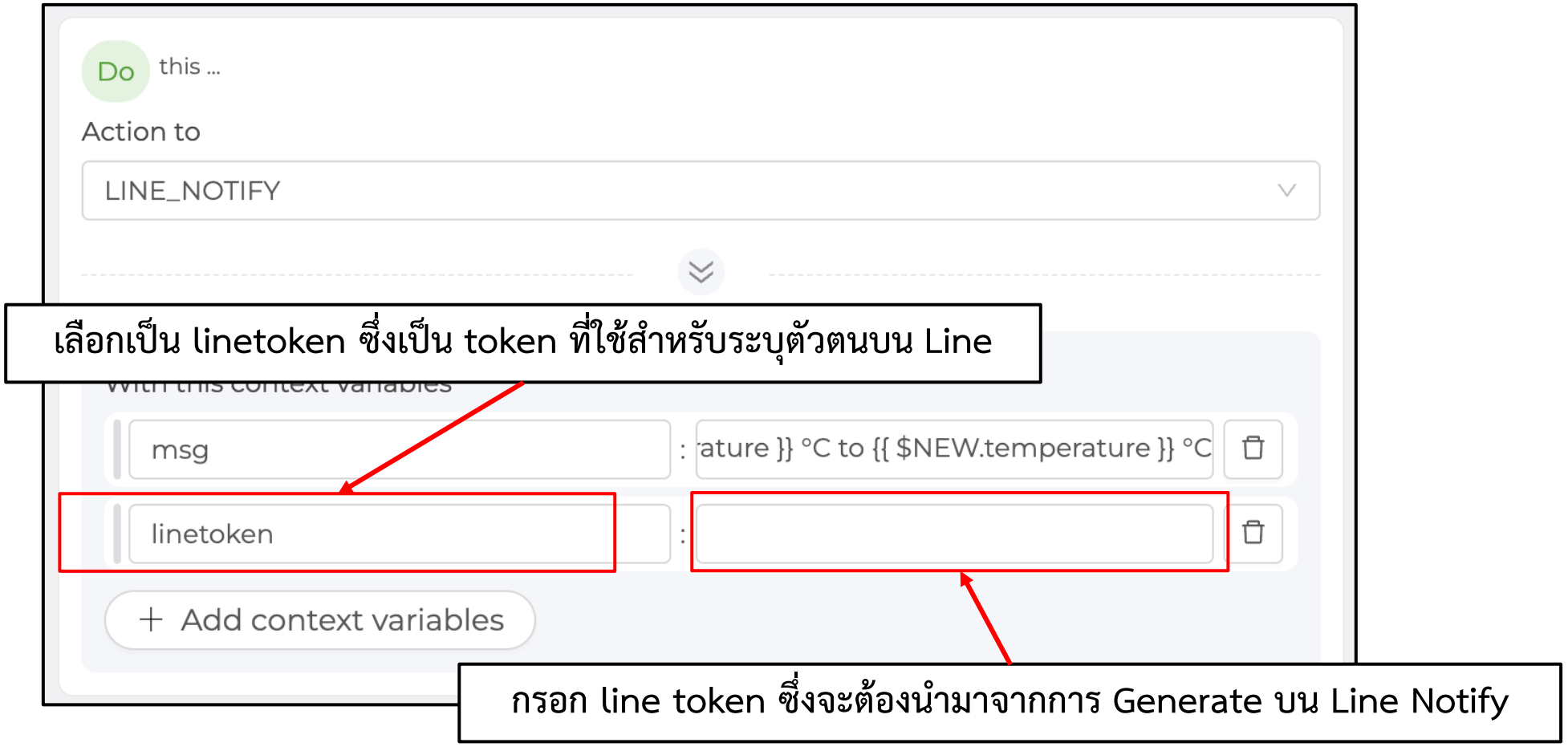
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



เพิ่มตัวแปรที่จะส่งไปยัง Event Hook

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

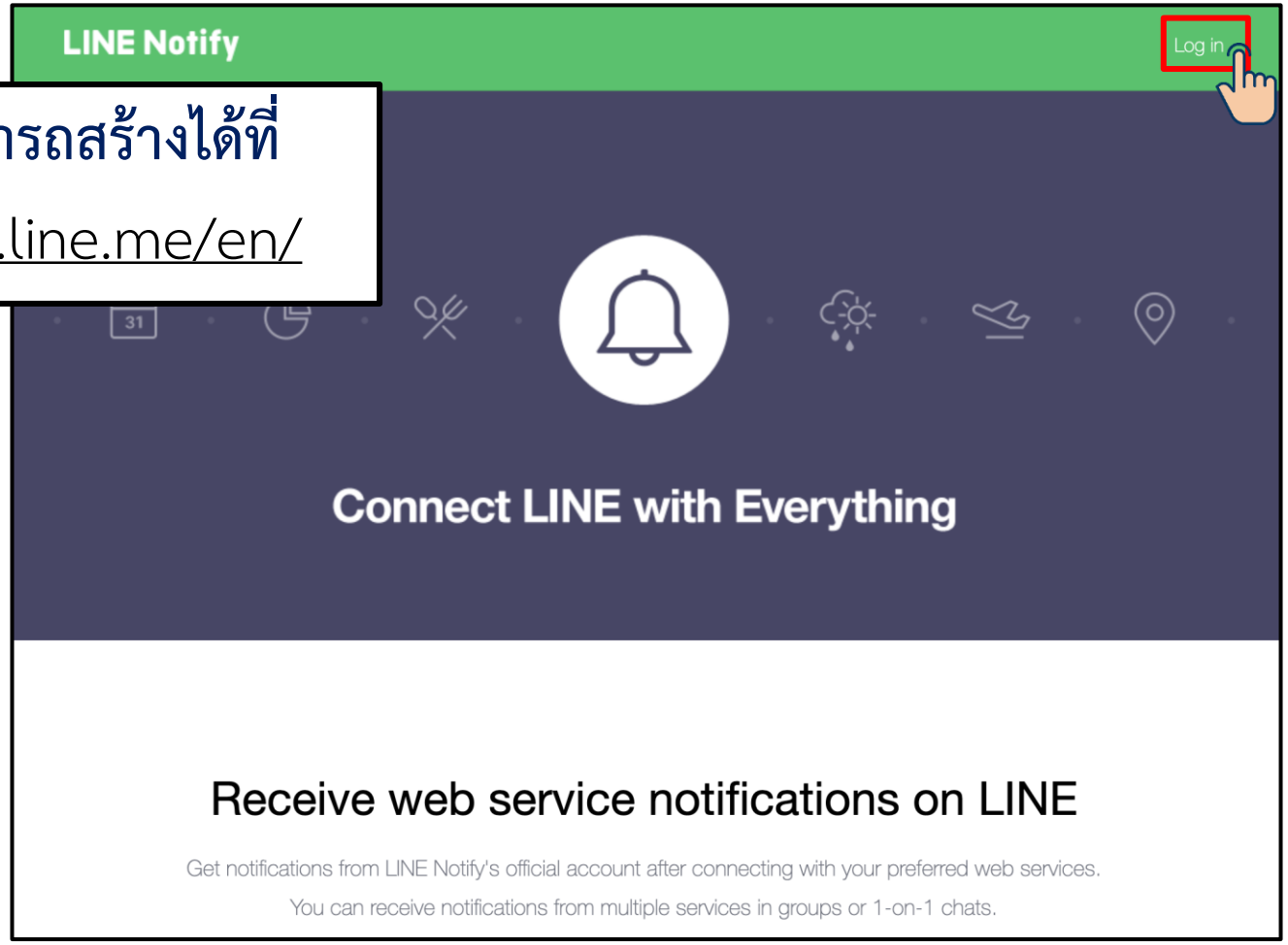


ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

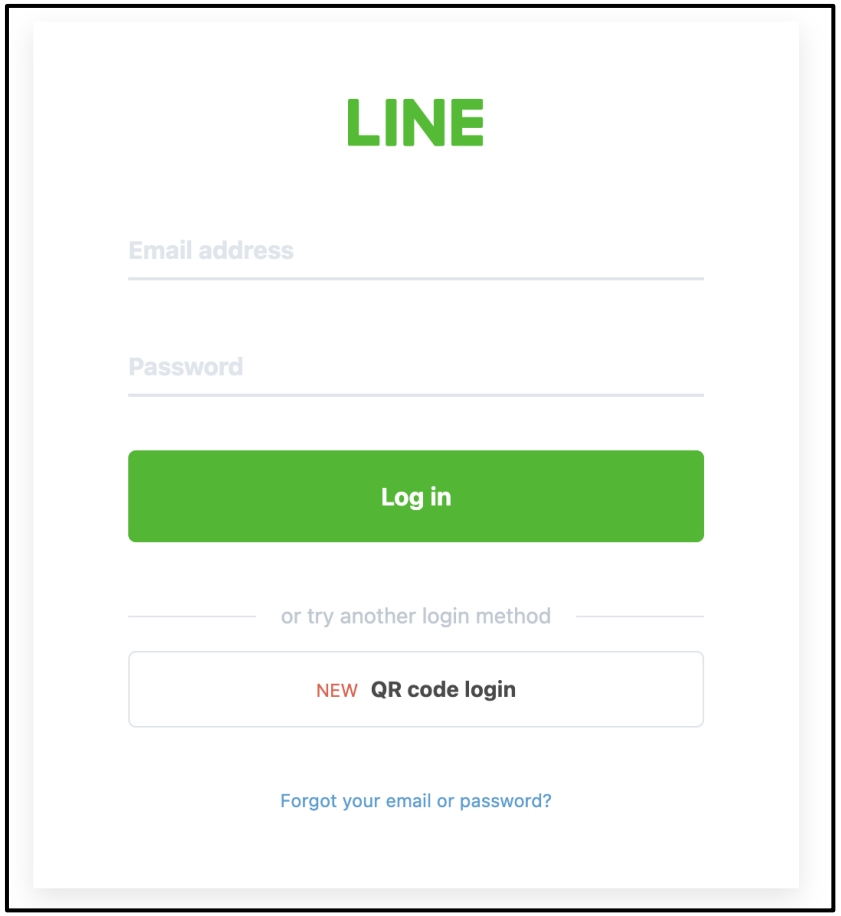
Token Line สามารถสร้างได้ที่

https://notify-bot.line.me/en/



ใบงานที่ 5.4 ขั้นตอนการทดลอง

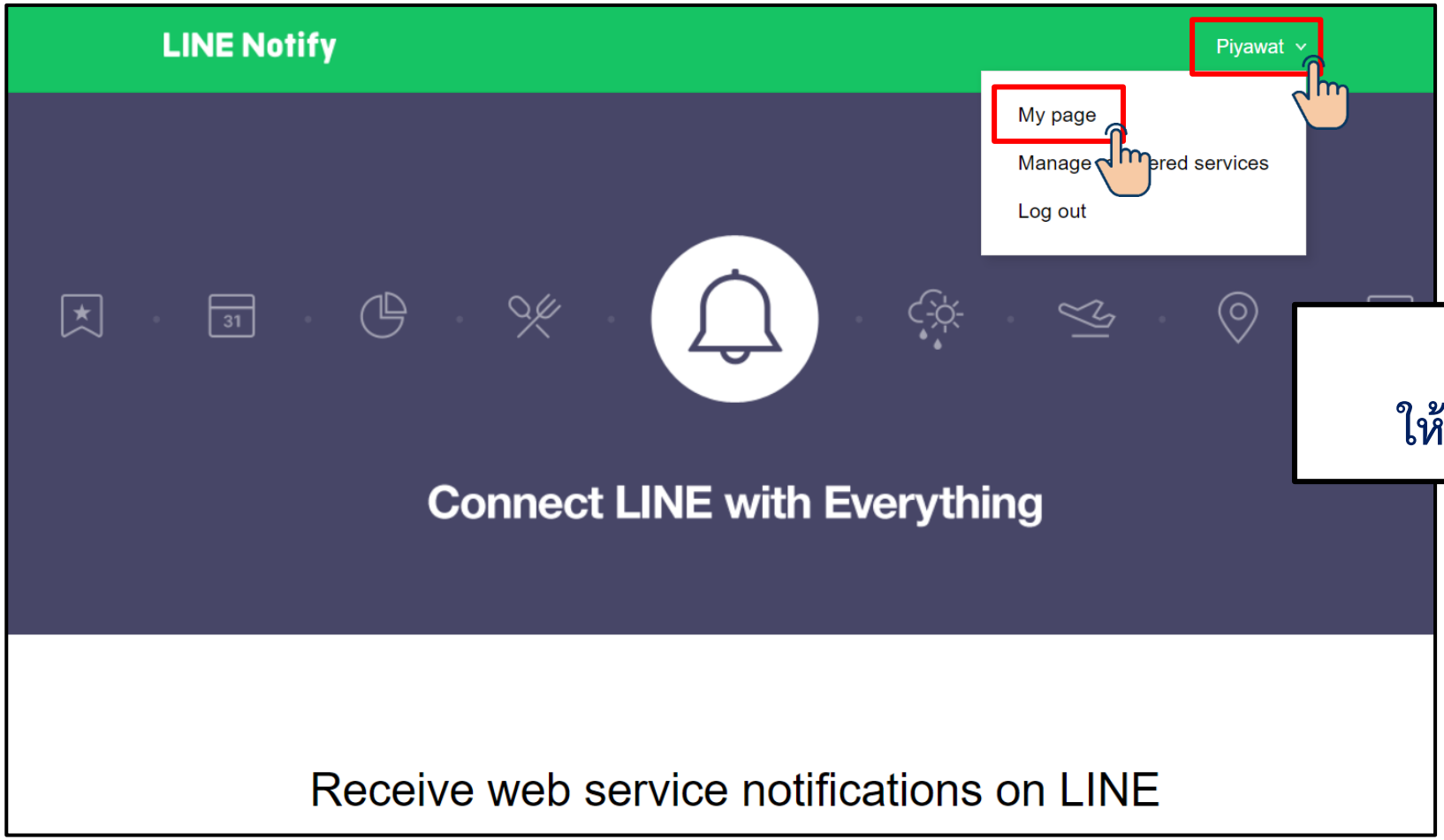
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Login เข้าสู่ระบบโดยใช้
E-mail และ Password ของ LINE ที่ใช้งาน

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

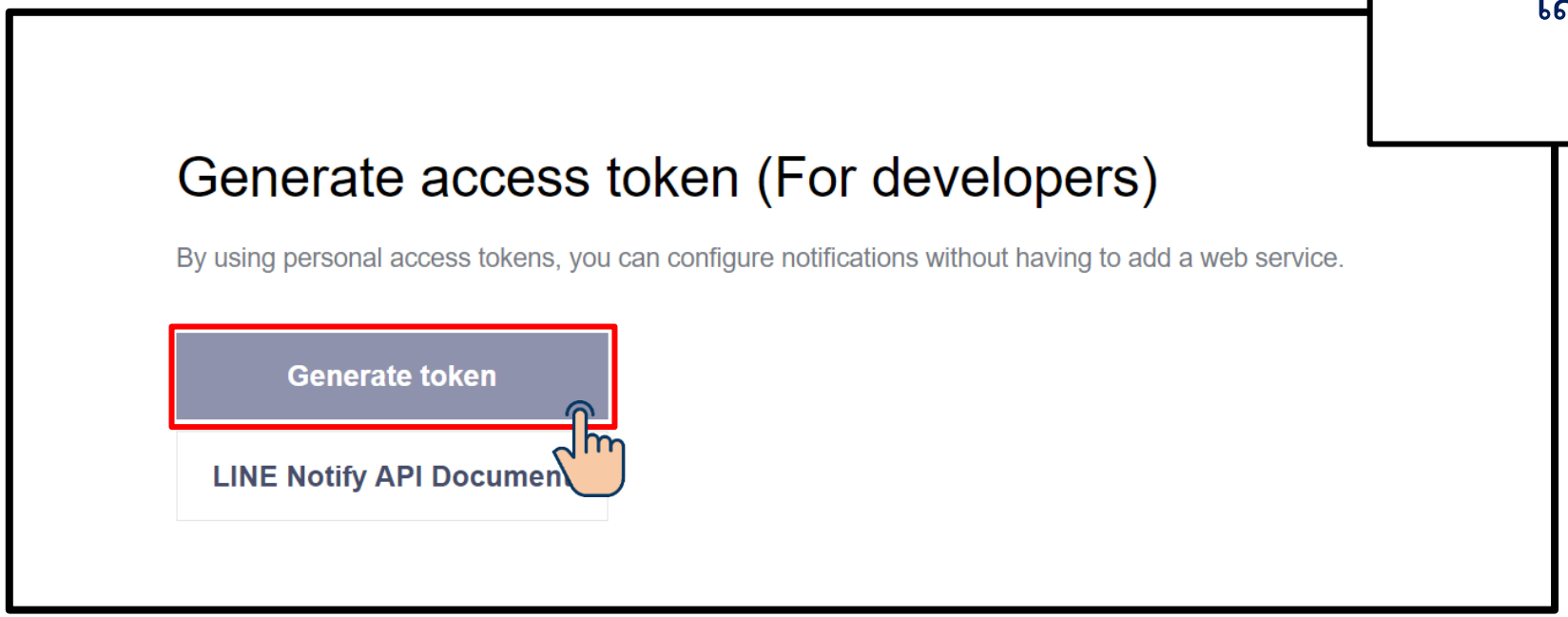


เมื่อเข้าสู่ระบบได้แล้ว
ให้เลือกที่ชื่อ >> My page

ใบงานที่ 5.4 ขั้นตอนการทดลอง

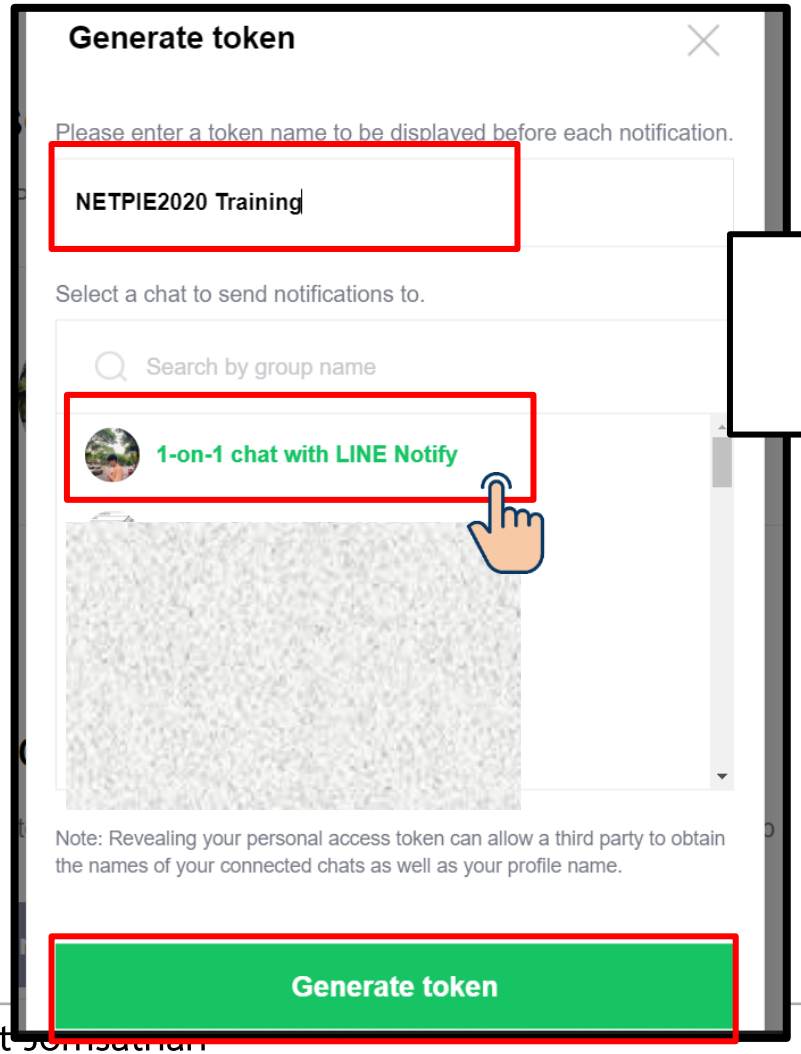
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

เลื่อนลงมาข้างล่างแล้วเลือก Generate Token



ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

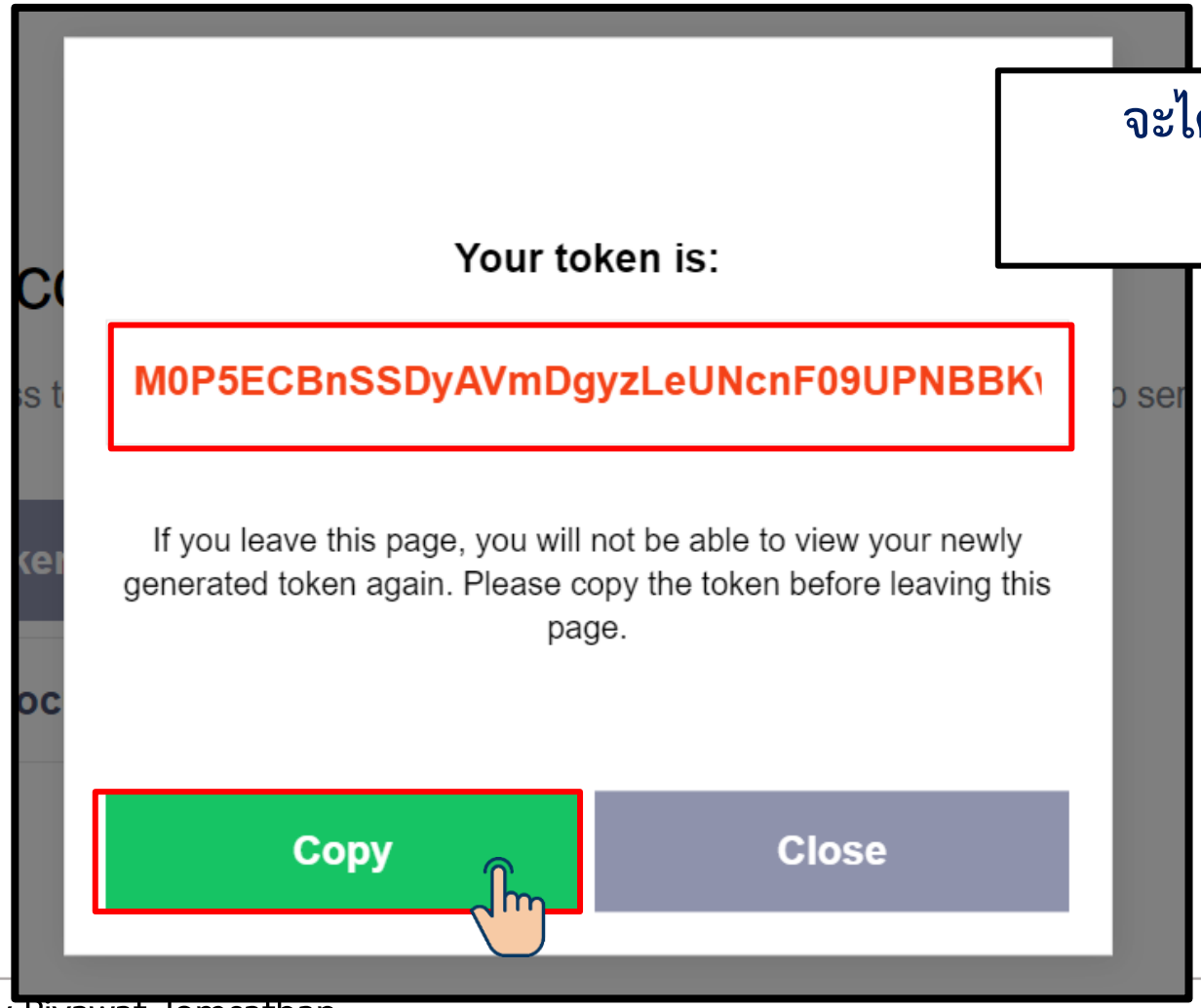


ตั้งชื่อหัวข้อ การสนทนา

แล้วเลือกแชทกับตัวเอง และคลิก Generate Token

ใบงานที่ 5.4 ขั้นตอนการทดลอง

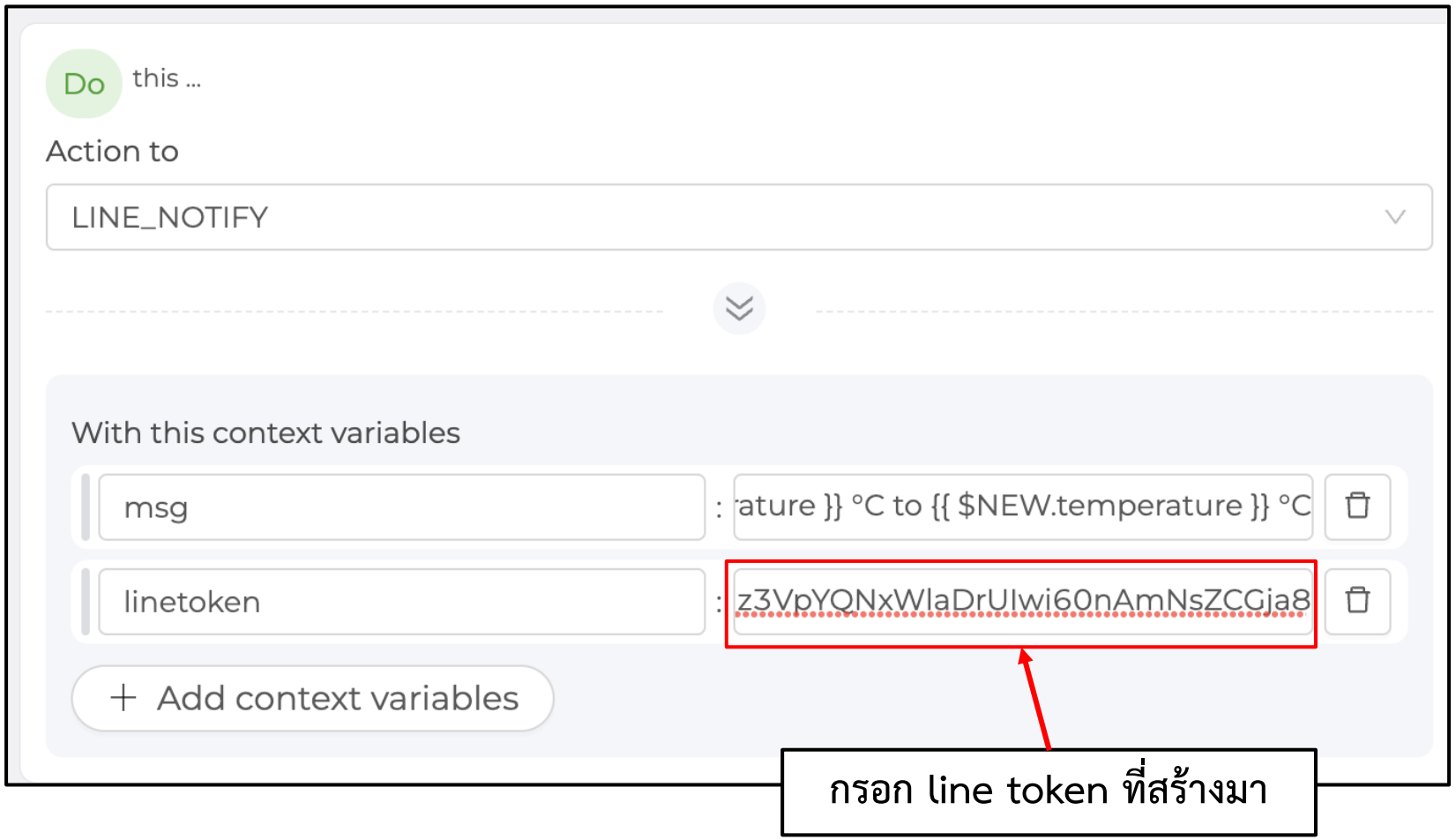
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



จะได้รับ Token ให้ Copy แล้วไปวางใน linetoken ใน Device Trigger

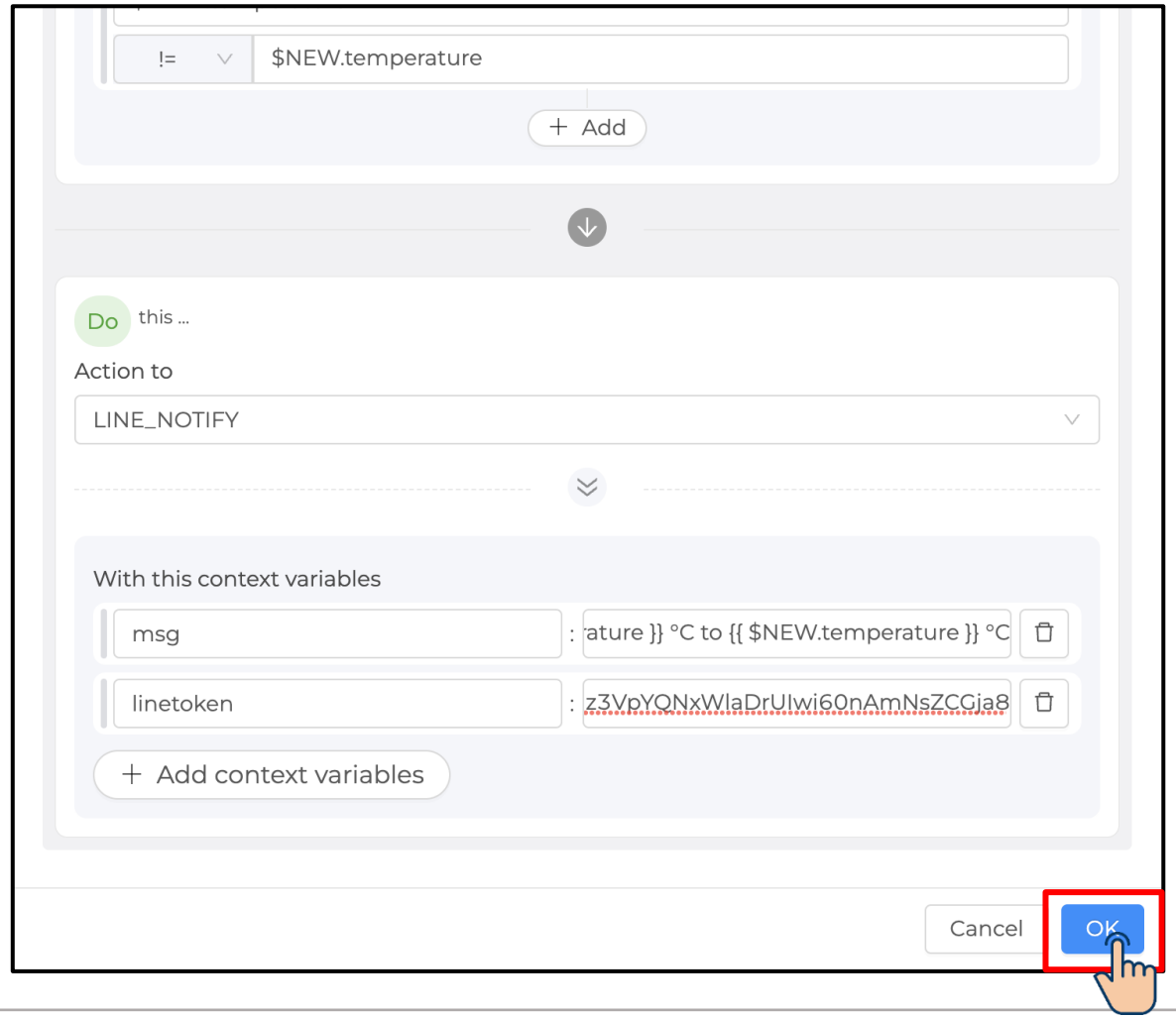
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

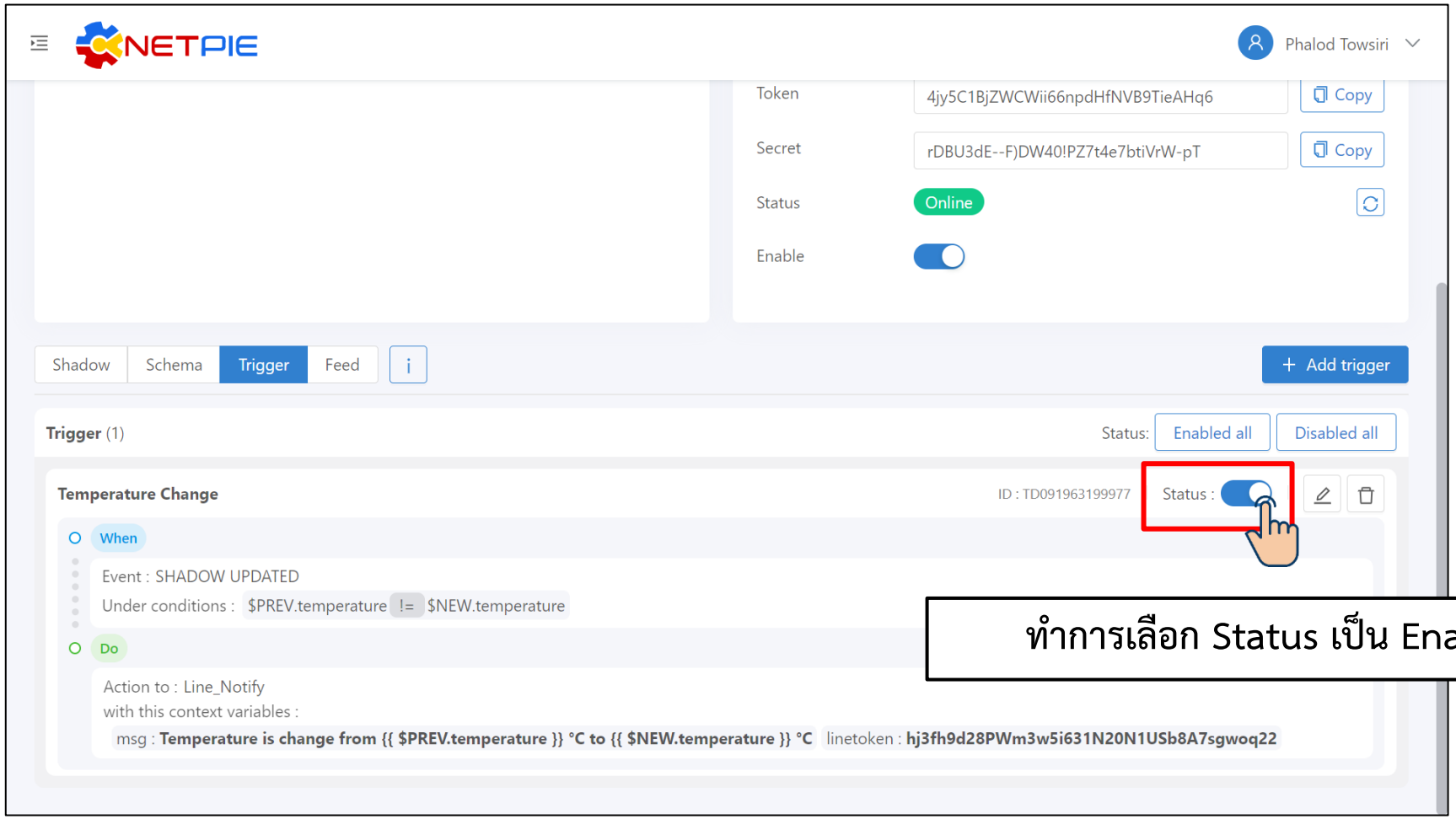
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



เมื่อสร้างทั้งหมดเสร็จแล้วทำการเลือก OK

ใบงานที่ 5.4 ขั้นตอนการทดลอง

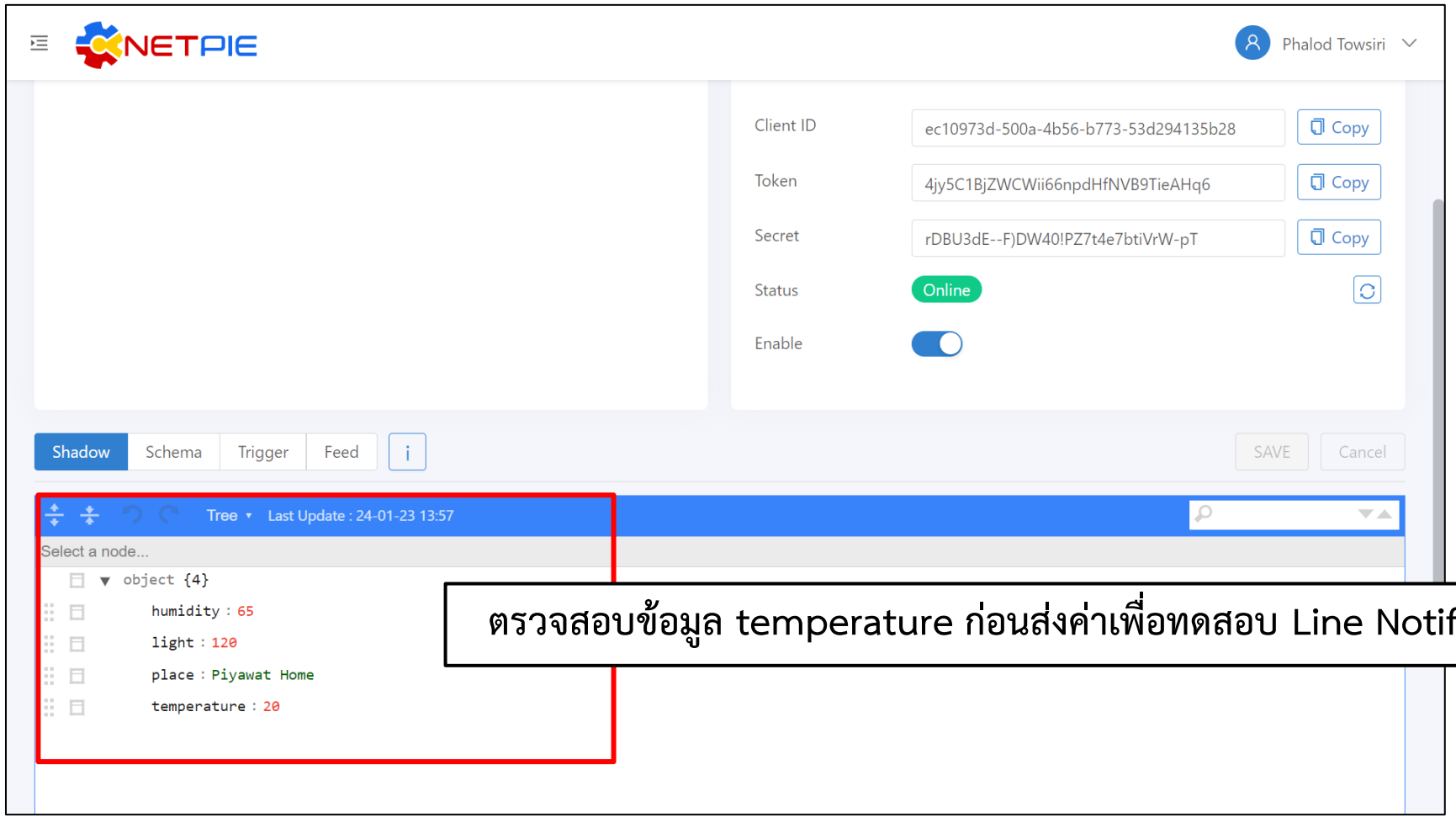
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ทำการเลือก Status เป็น Enable

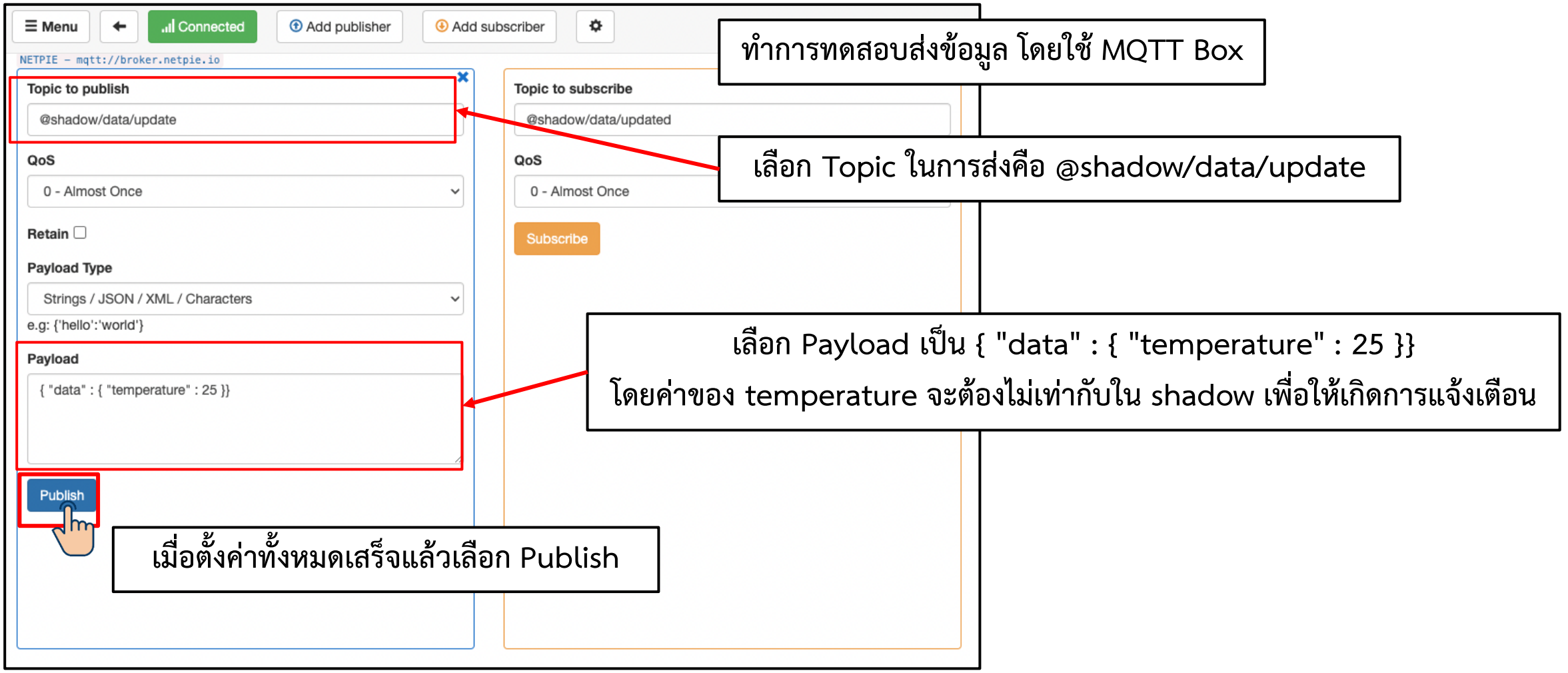
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



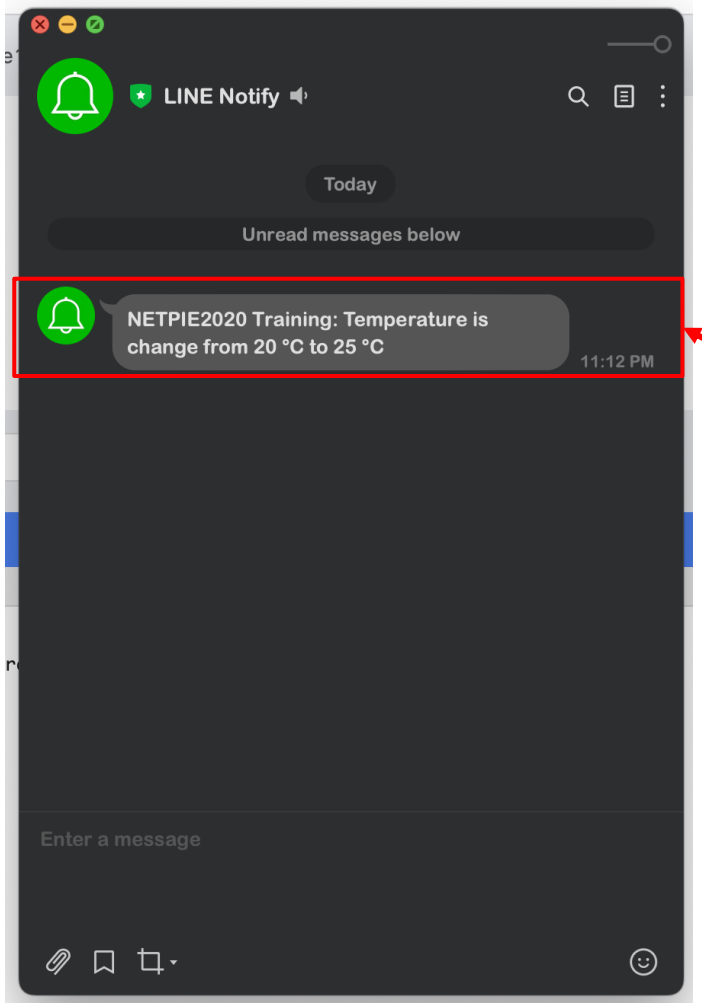
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

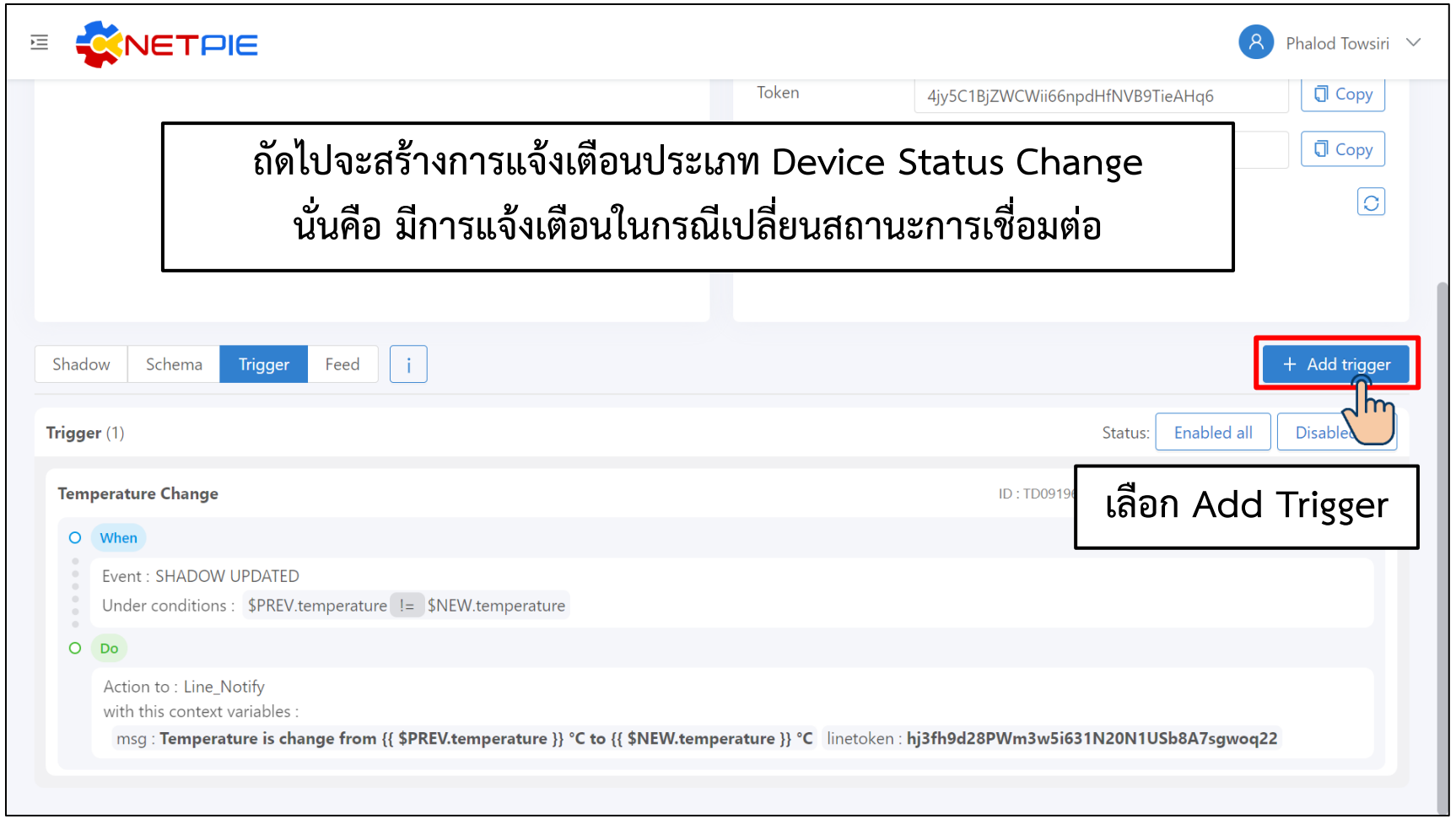
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ข้อความที่ถูกส่งเข้ามาใน Line Notify

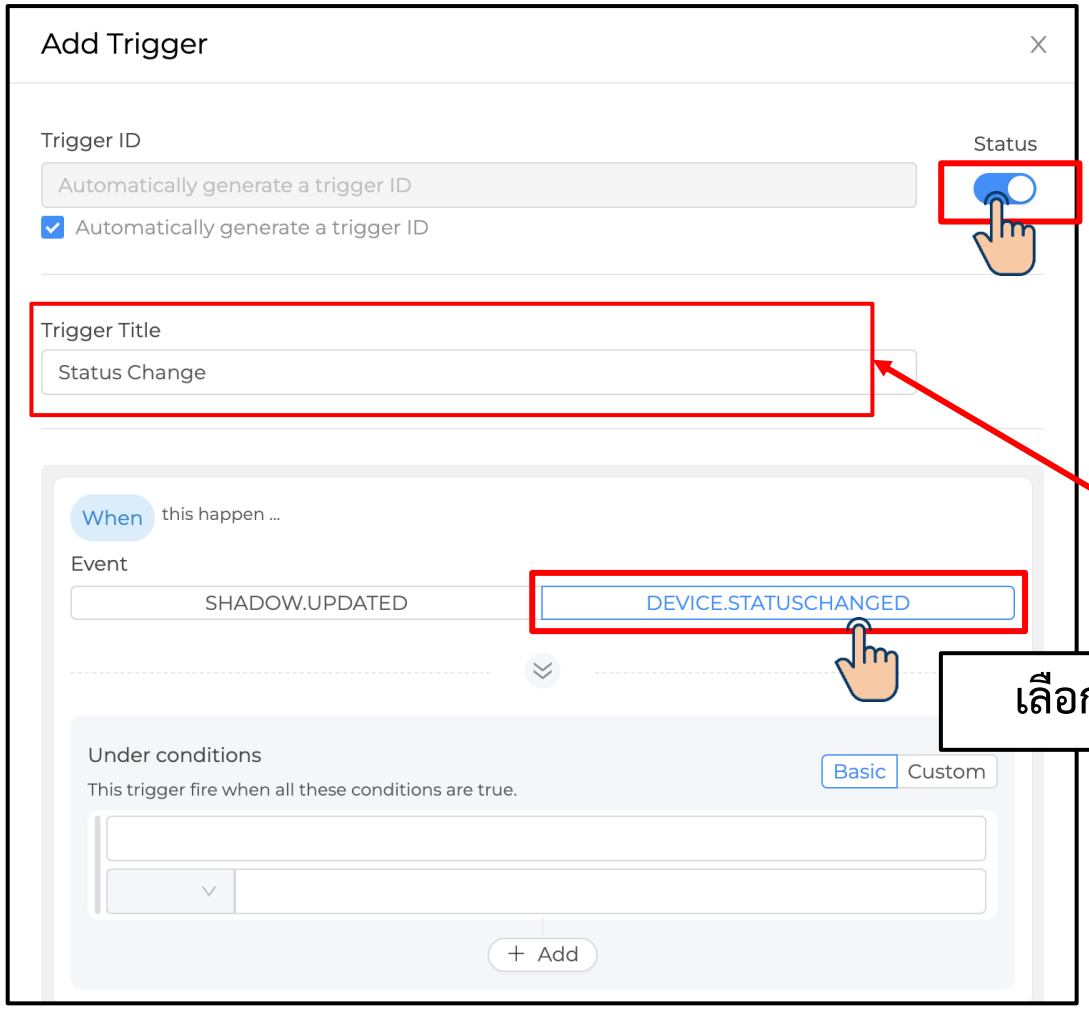
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



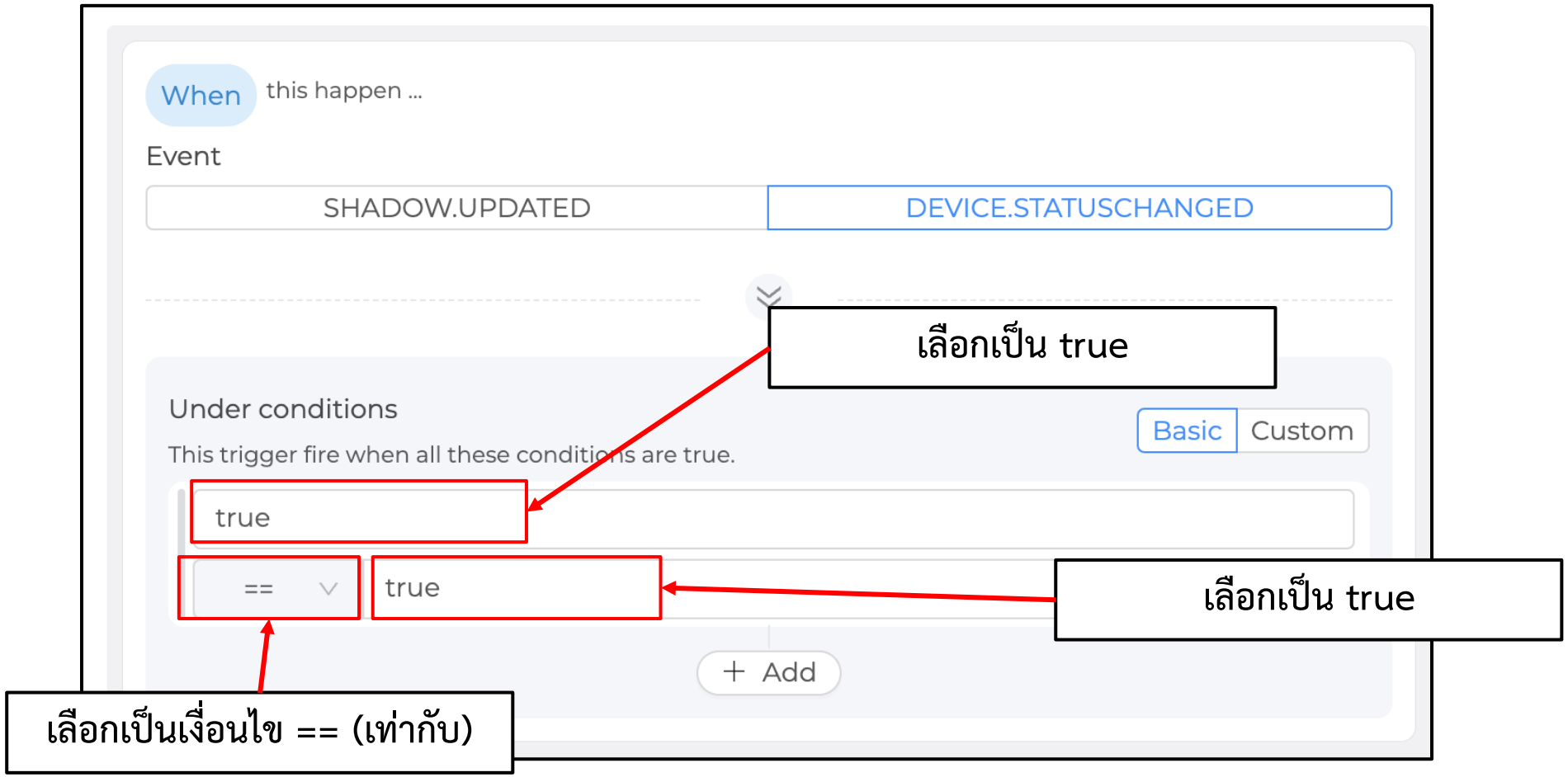
เราจะสร้างเงื่อนไขการแจ้งเตือน คือ เมื่อมีการเปลี่ยนสถานะไม่ว่าจะเป็นรูปแบบไหนก็ตาม ให้แจ้งเตือนไปยัง Line Notify โดยมีข้อความคือ "Status change from OLD.Status to NEW.Status"

ตั้งชื่อ Trigger

เลือก DEVICE.STATUSCHANGED

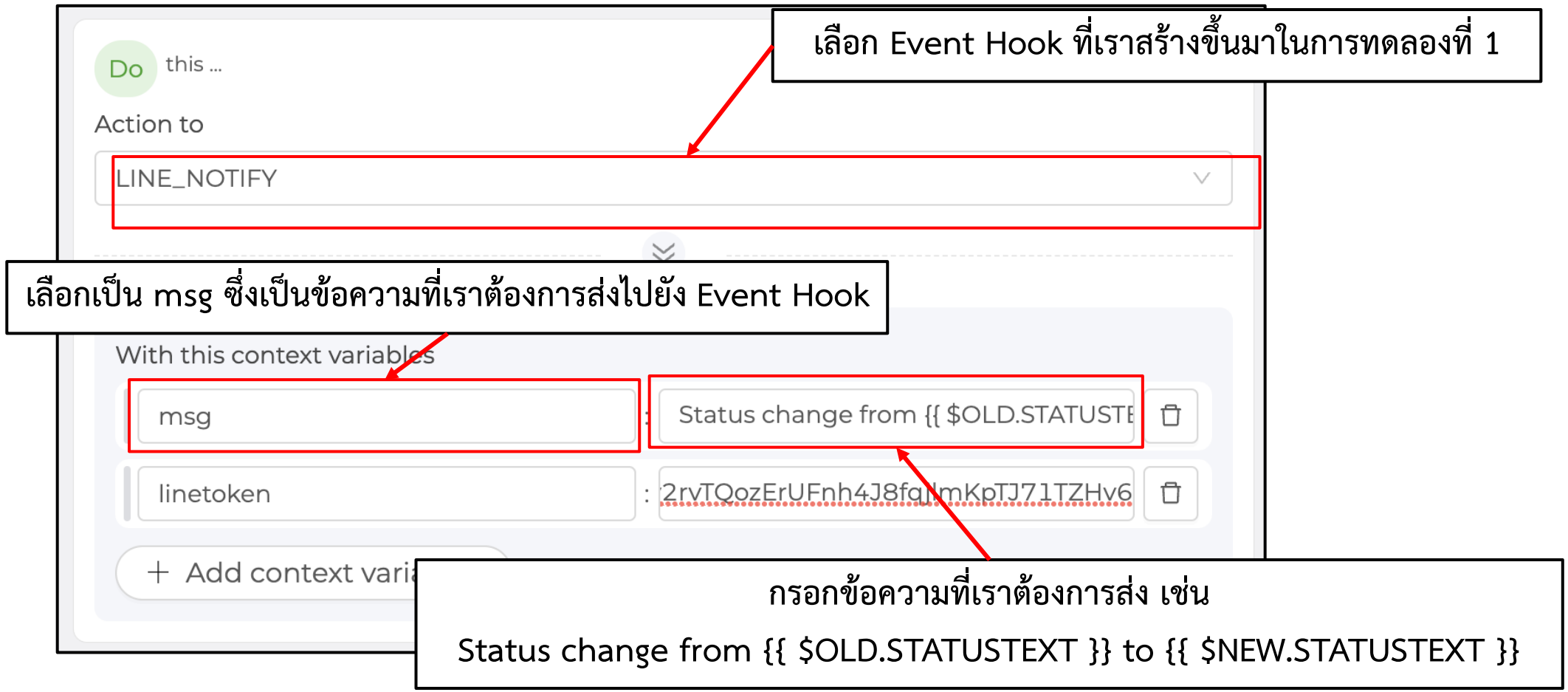
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



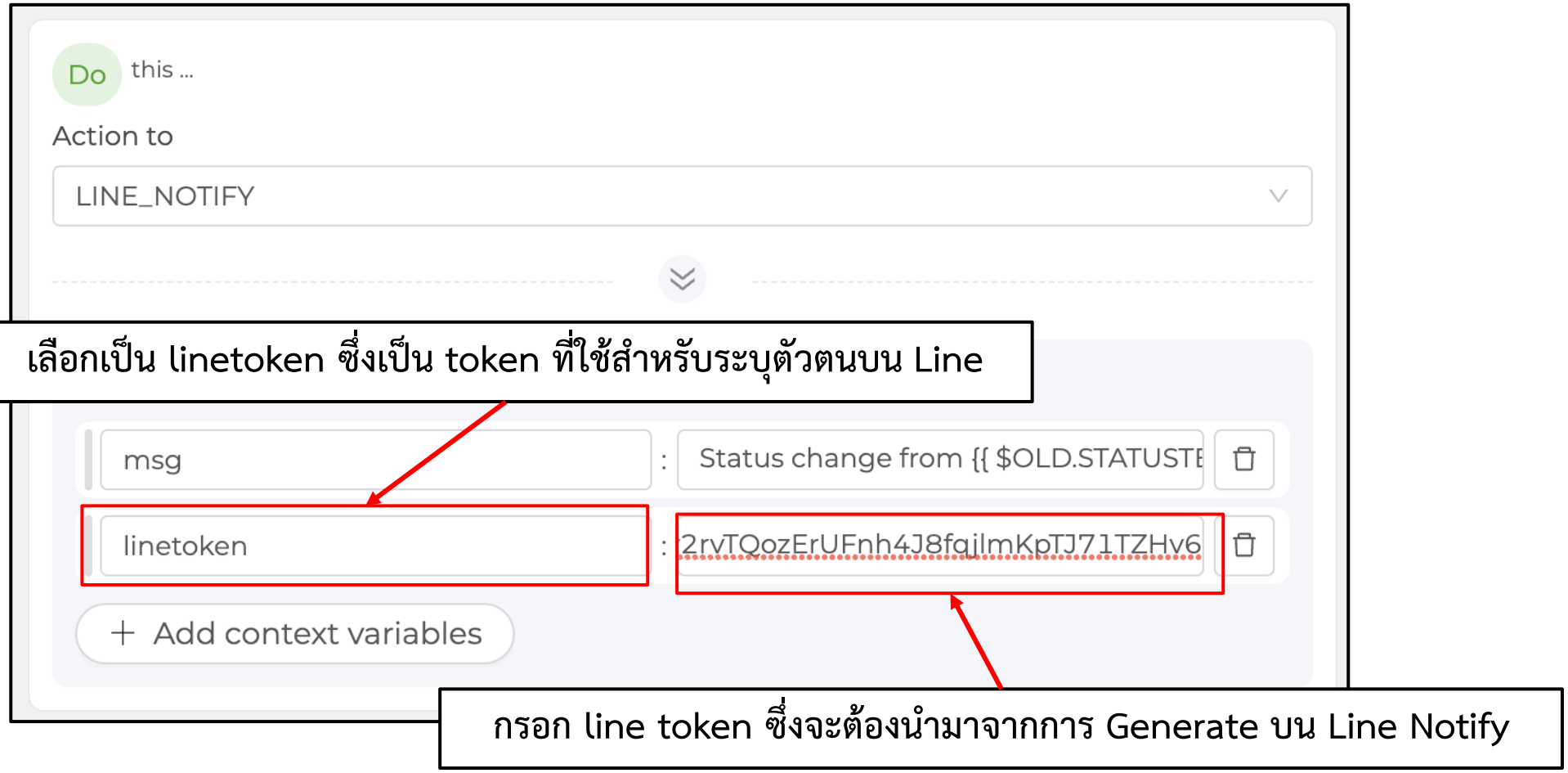
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



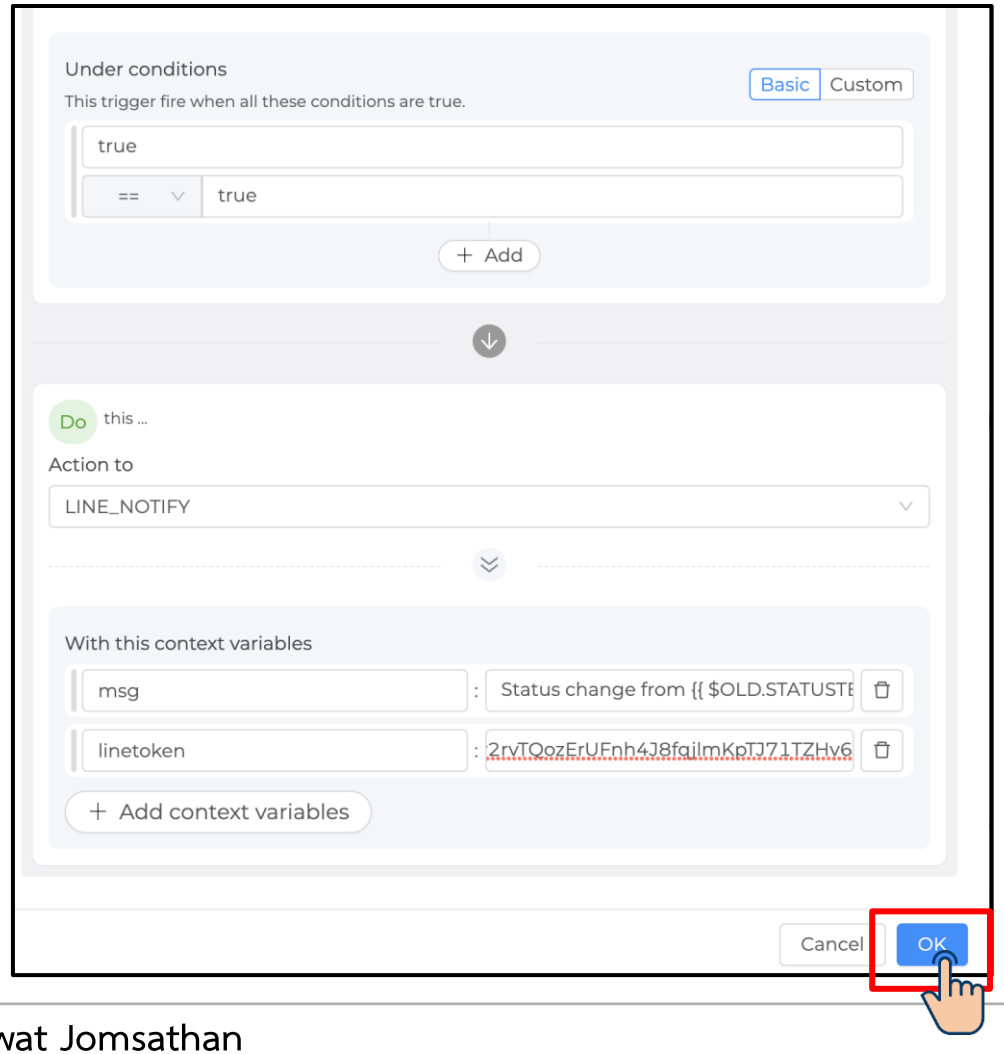
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

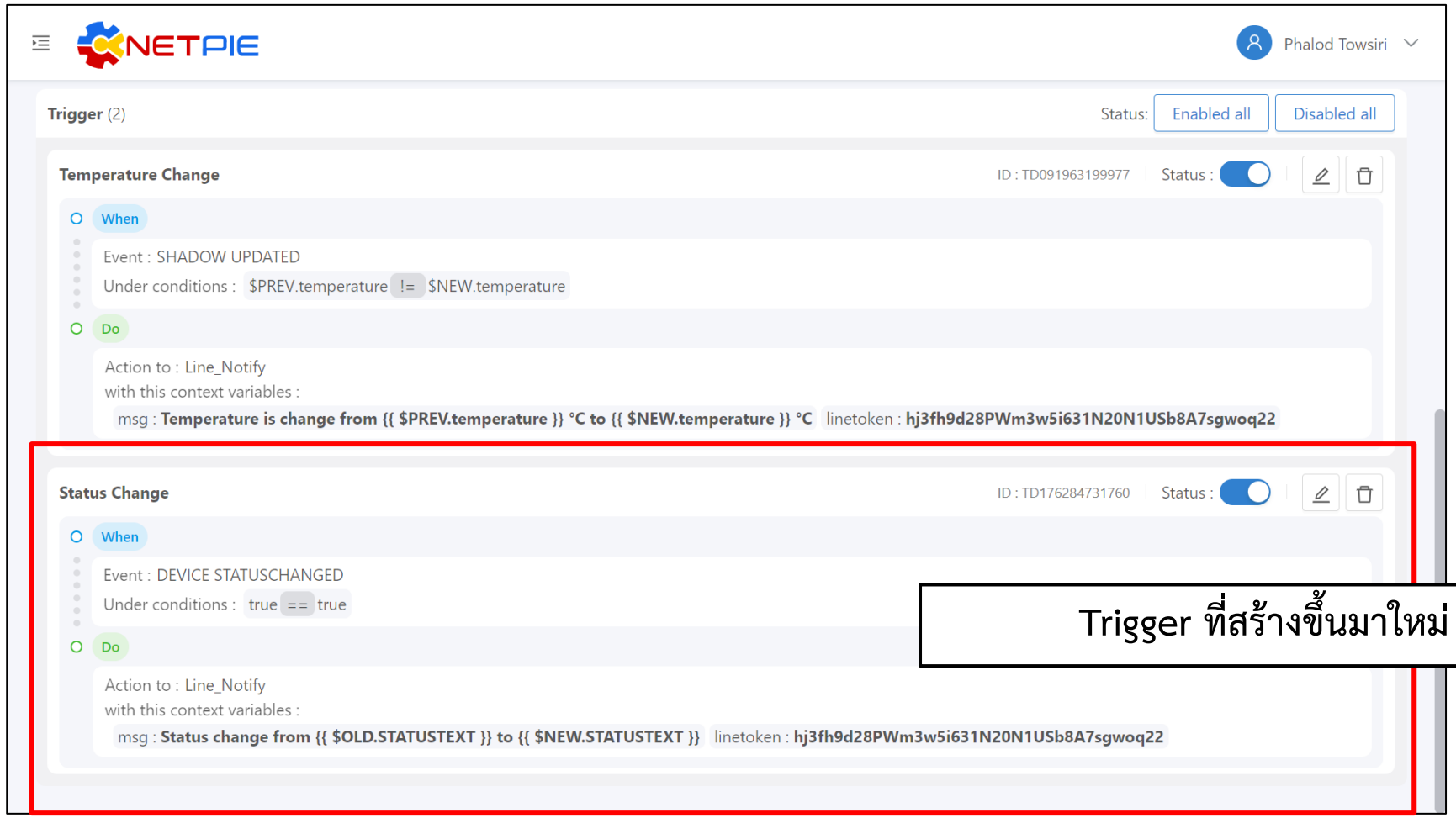
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



เมื่อสร้างทั้งหมดเสร็จแล้วทำการเลือก OK

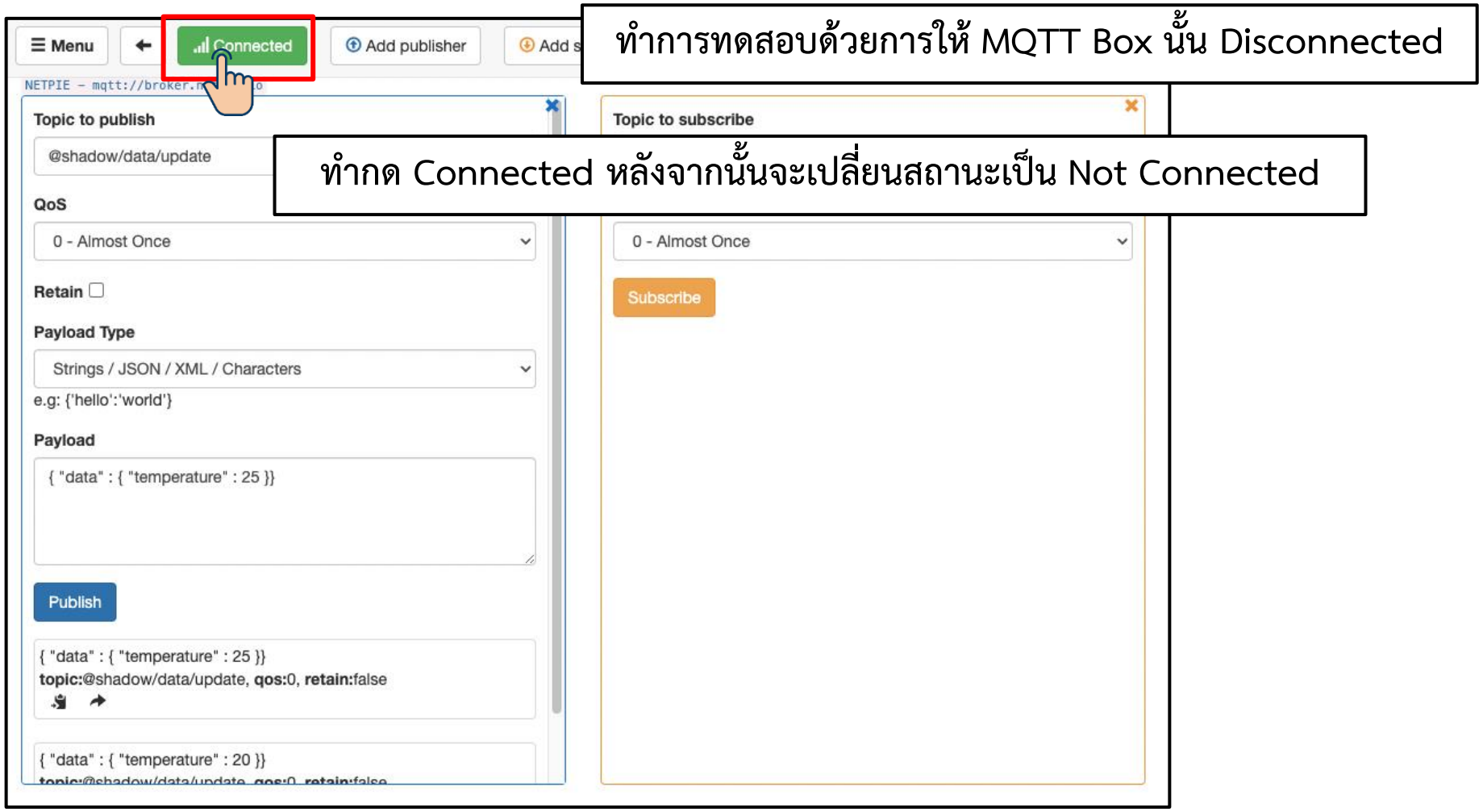
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



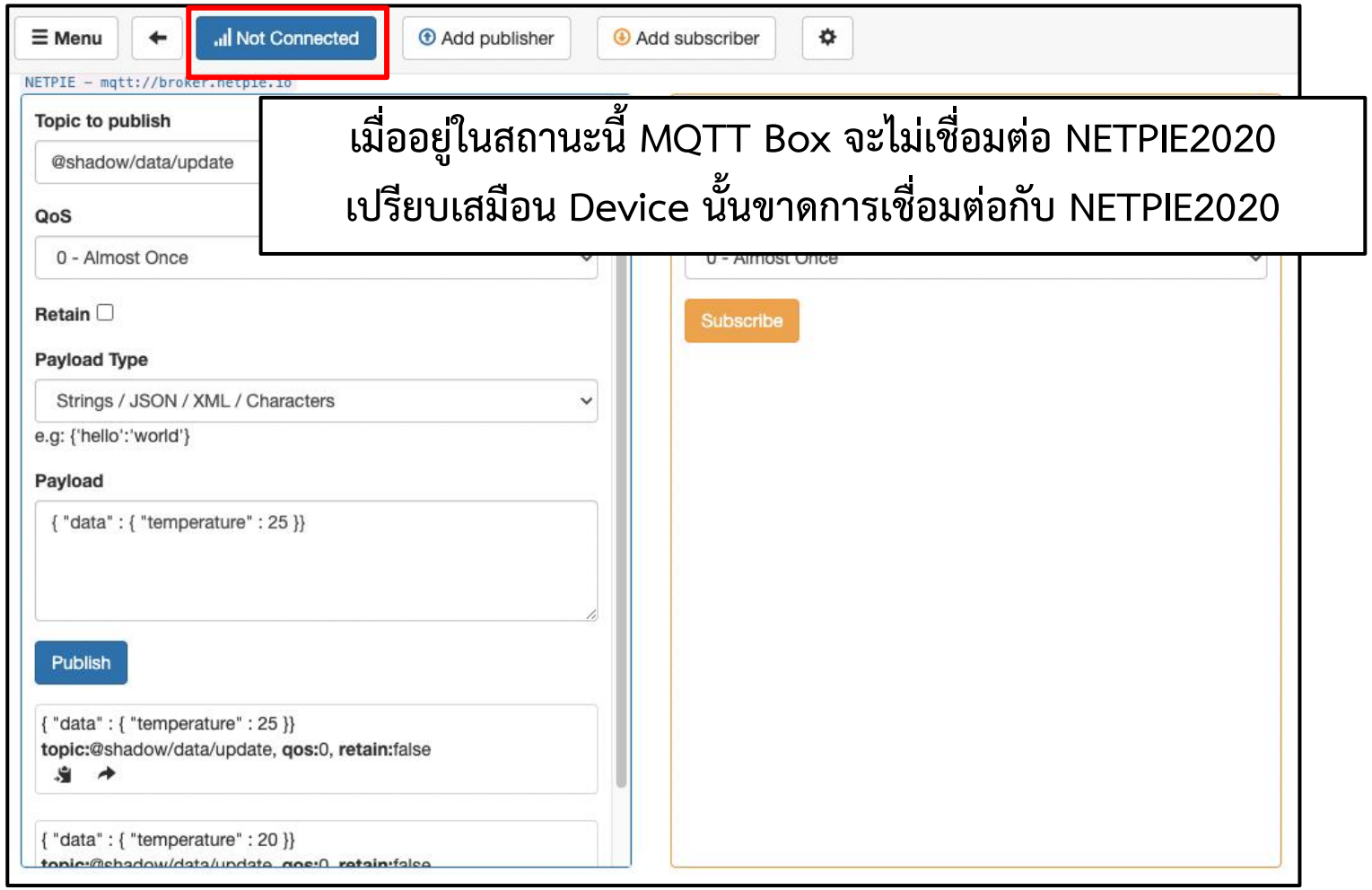
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



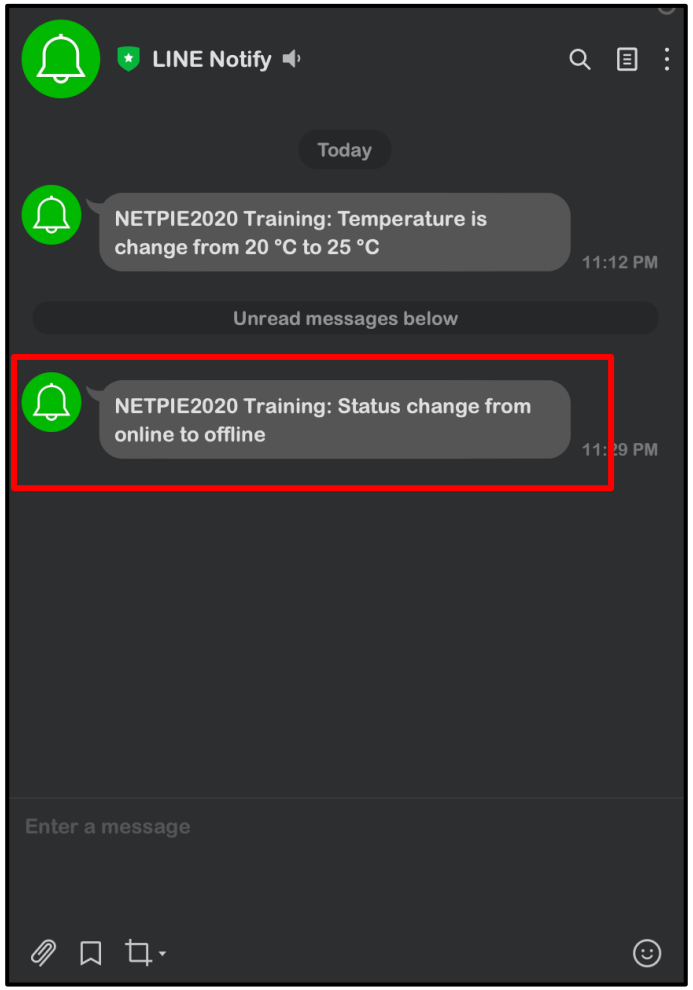
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

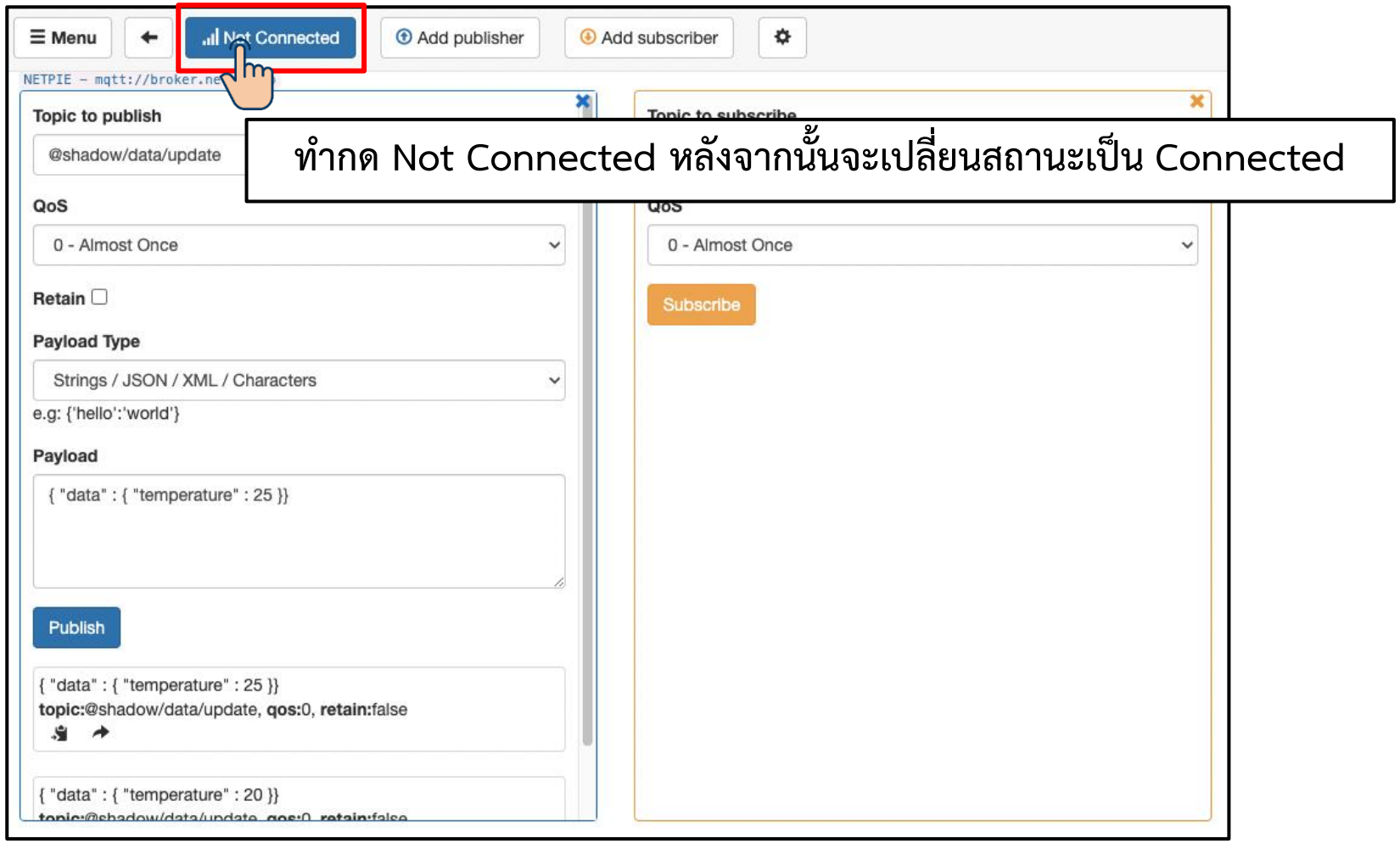
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ข้อความการแจ้งเตือนสถานะที่เปลี่ยนไปถูกส่งมายัง Line

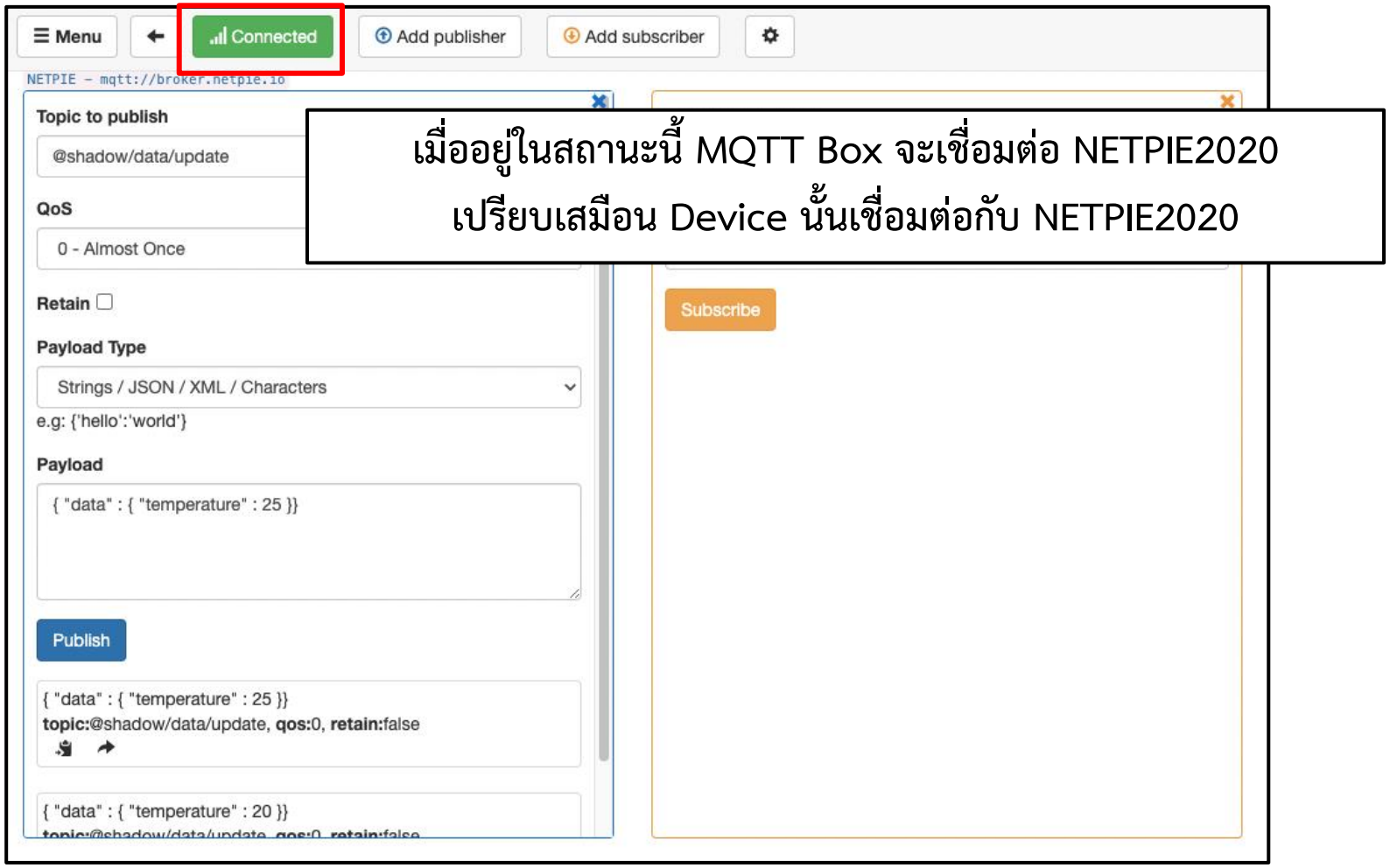
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ใบงานที่ 5.4 ขั้นตอนการทดลอง

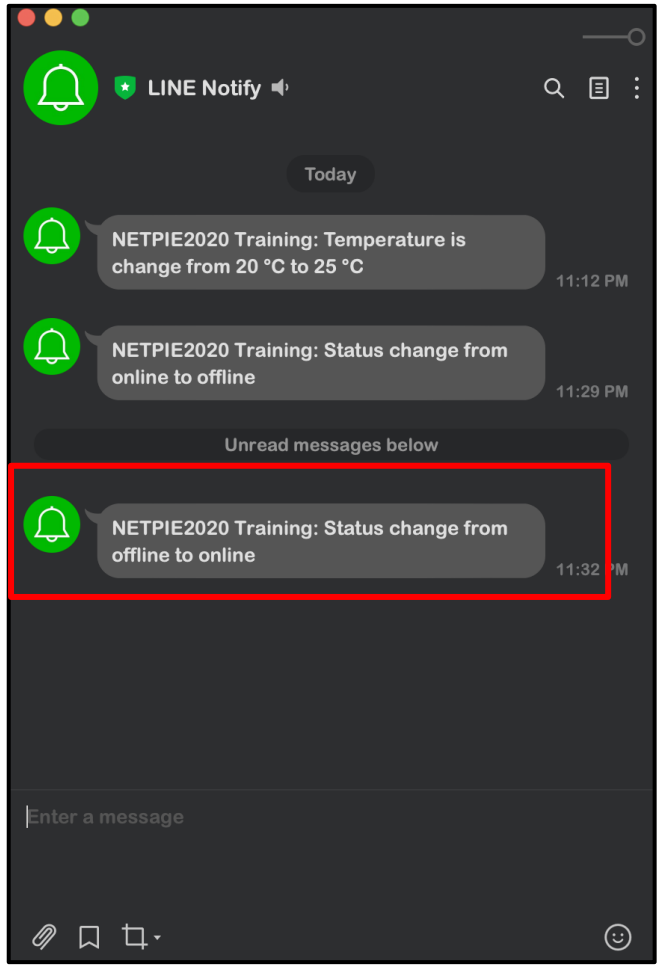
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



เมื่ออยู่ในสถานะนี้ MQTT Box จะเชื่อมต่อ NETPIE2020
เปรียบเสมือน Device นั้นเชื่อมต่อกับ NETPIE2020

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ข้อความการแจ้งเตือนสถานะที่เปลี่ยนไปถูกส่งมายัง Line