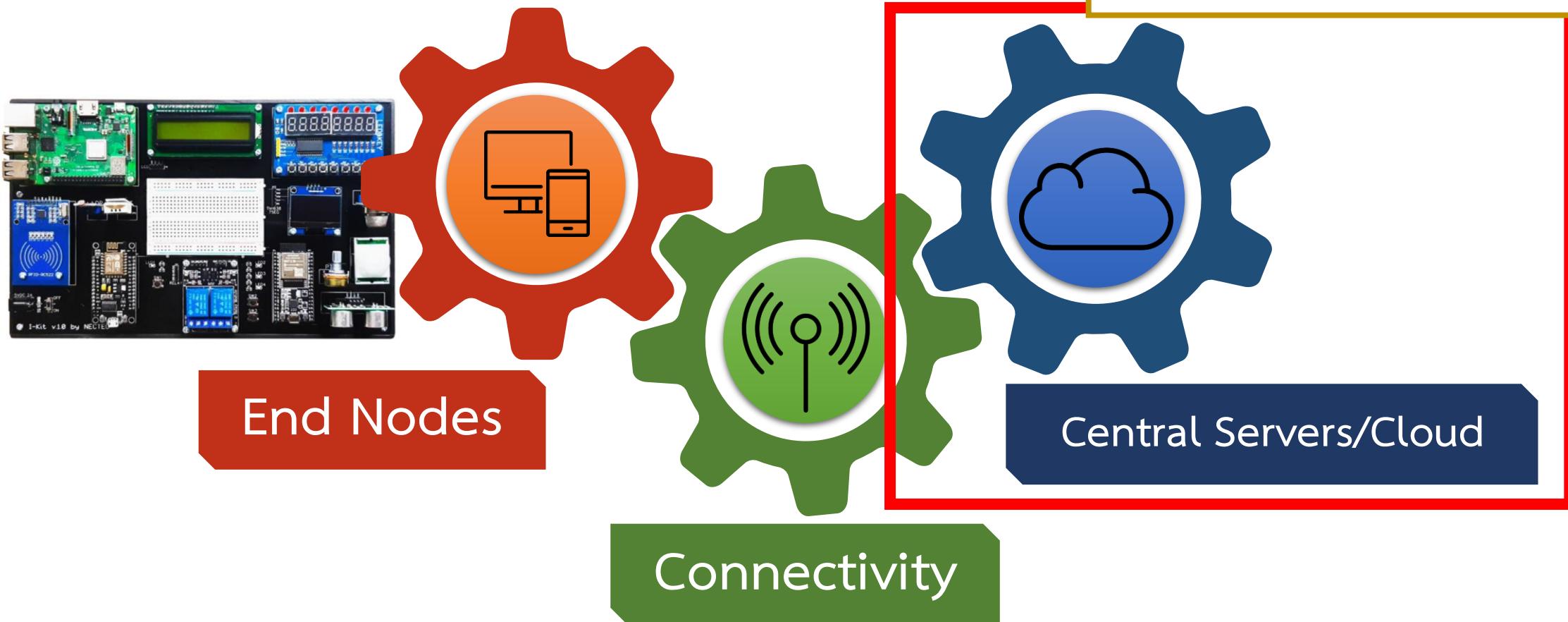


4

NETPIE2020

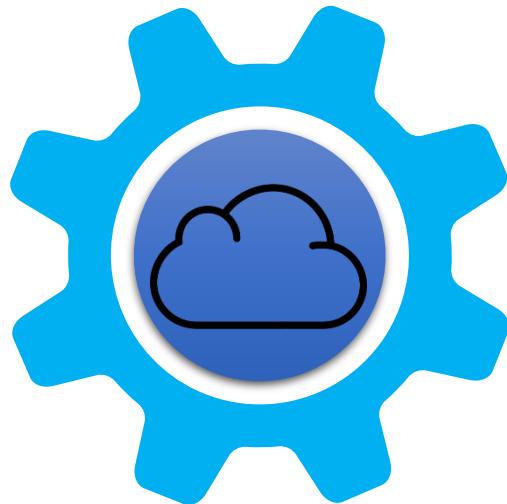


องค์ประกอบของ IoT



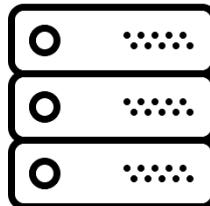
องค์ประกอบของ IoT

Central Servers/Cloud



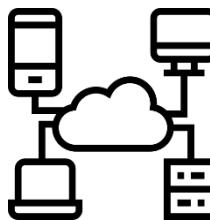
Central Servers/Cloud คือ พื้นที่จัดเก็บข้อมูลเพื่อนำมาสร้างเป็น **Web Server**, **Application Server** เป็นต้น โดยปัจจุบันสำหรับระบบ IoT นิยมใช้ Cloud เป็นพื้นที่จัดเก็บข้อมูลโดยที่ Server และ Cloud มีความแตกต่างกันคือ

Server



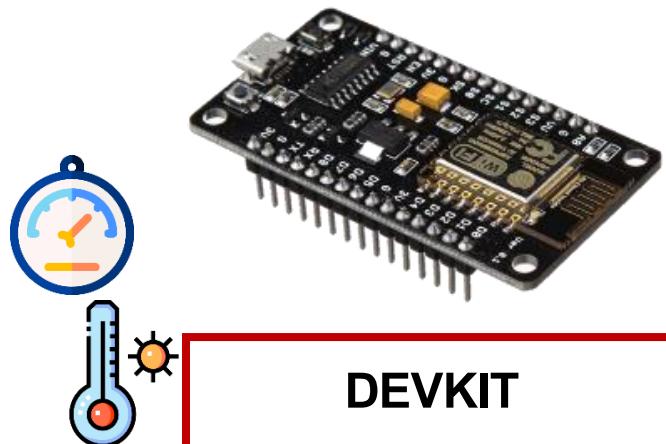
คือ เครื่องหรือโปรแกรมคอมพิวเตอร์ซึ่งทำงานให้บริการในระบบเครือข่ายแก่ลูกข่าย เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นเซิร์ฟเวอร์นี้ควรมีประสิทธิภาพสูง มีความเสถียร สามารถให้บริการแก่ผู้ใช้ได้เป็นจำนวนมาก

Cloud



คือ การนำ Server หลายๆเครื่องมาทำงานด้วยกันโดยมีหน่วยประมวลผลระดับสูง โดย Cloud สามารถสร้าง Service ขึ้นมาทำงานได้หลากหลาย ในปัจจุบัน Cloud มีผู้ให้บริการหลากหลายทั้งแบบมีและไม่มีค่าใช้จ่าย

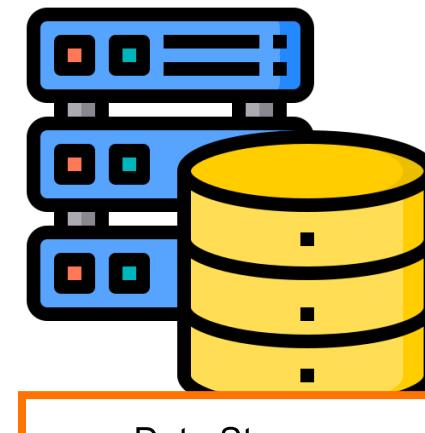
องค์ประกอบของ IoT



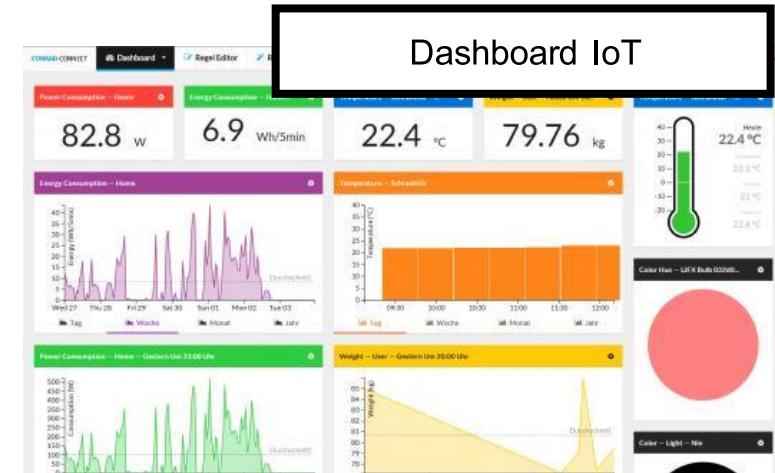
DEVKIT
(ESP8266/32)

ข้อมูลที่ DEVKIT ต้องการส่งไปยัง Cloud
"Temperature" = 25 °C
"Humidity" = 80 %H
"LED Status" = "OFF"

วิธีในการส่งข้อมูล/เส้นทางใน
การส่งข้อมูล

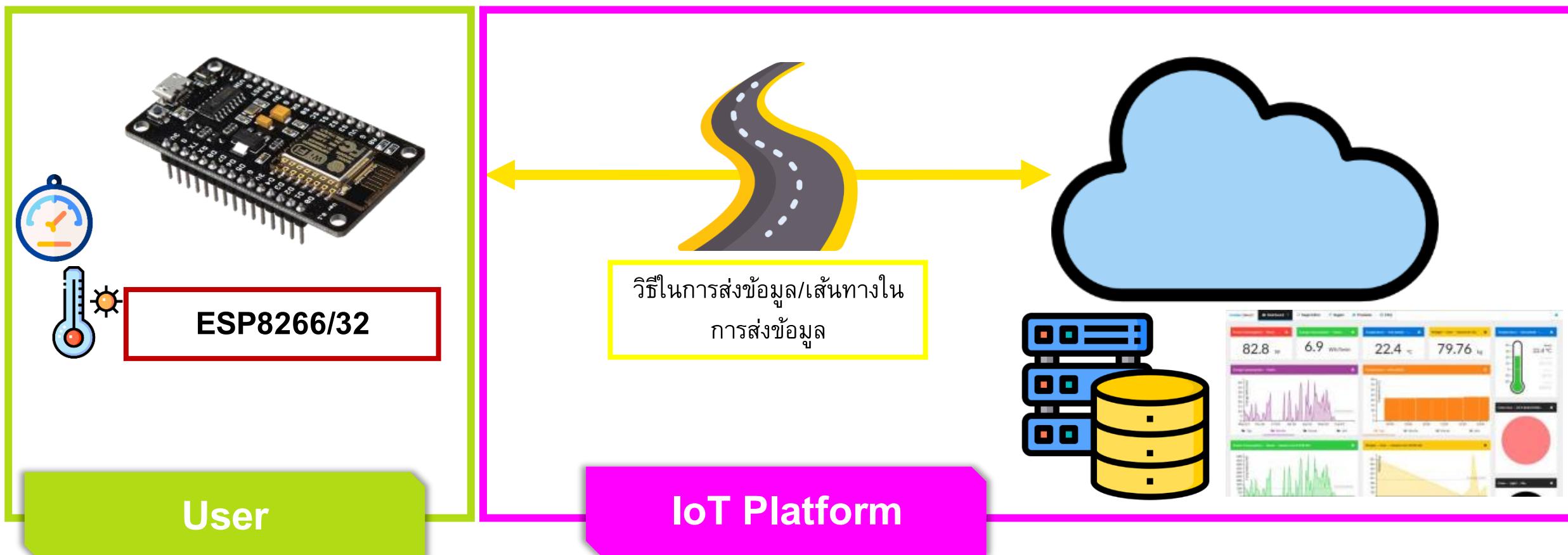


Server/Cloud



IoT Platform

IoT Platform คือ แพลตฟอร์มที่ช่วยให้ผู้ใช้งานนำเซนเซอร์หรืออุปกรณ์เชื่อมต่อกันได้ โดยมีการจัดการ Protocol, จัดเก็บข้อมูล, รักษาความปลอดภัยของข้อมูล เป็นต้น



IoT Platform vs Conventional Server

ข้อกำหนดต่างๆ	Conventional Server	IoT Platform
การลงทุน 	Infrastructure + คน	Pay as you go
ความปลอดภัย 	เจ้าของระบบ	แพลตฟอร์ม
การขยายตัวรับ荷重 	Interrupted	Seamless
การรองรับ SW/HW 	Customized	As universal as possible

IoT Platform ที่นิยมในประเทศไทย

 ThingSpeak



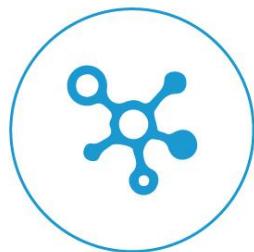
 ThingsBoard



 NETPIE
where things chat



IoT Platform ປະເກມ 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

4 – NETPIE2020

NETPIE คืออะไร ??

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



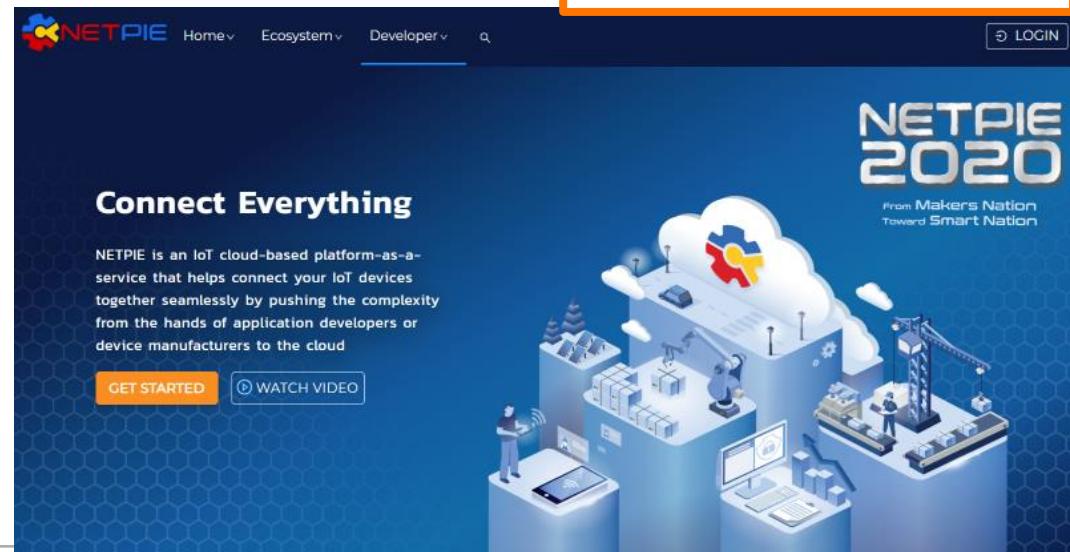
NETPIE 2015

2015.netpie.io



NETPIE 2020

netpie.io



คุณสมบัติหลักๆ ของ NETPIE



Monitoring

การแสดงค่าข้อมูลจากเซนเซอร์หรืออุปกรณ์แบบ Real-Time



Controlling

การควบคุมการทำงานของอุปกรณ์ต่างๆ ผ่าน Cloud Platform



Data Storage

การเก็บข้อมูลที่ได้จากเซ็นเซอร์หรืออุปกรณ์



Dashboard for monitor & Control

การแสดงผลและควบคุมการทำงานผ่าน Dashboard

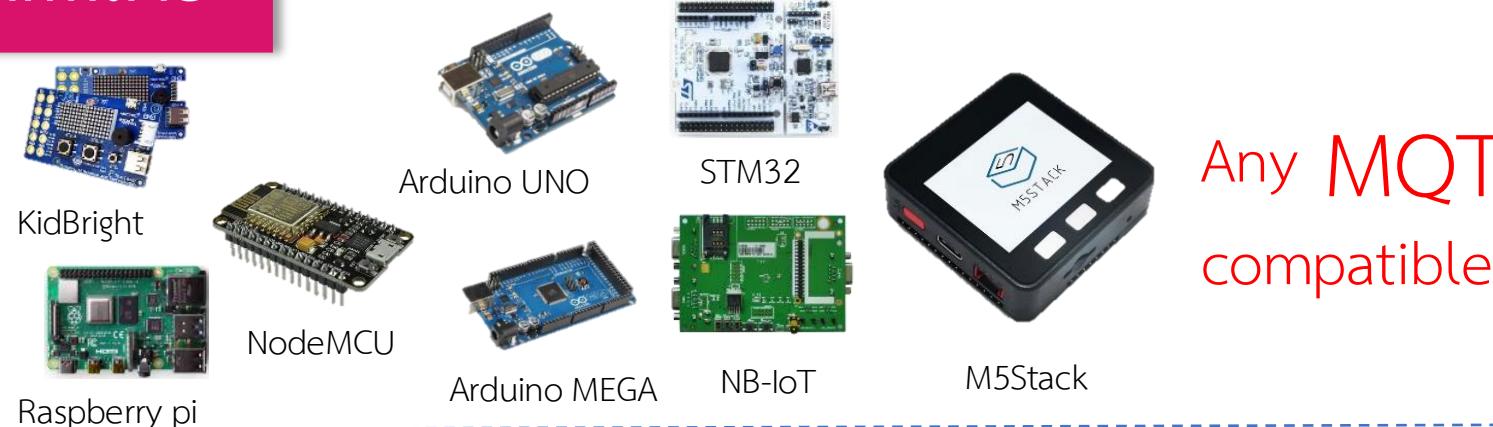


Notification

การแจ้งเตือนความผิดปกติของเซนเซอร์หรืออุปกรณ์

รองรับอุปกรณ์ที่หลากหลาย

Hardware



Any MQTT
compatible hardware

Programing Language



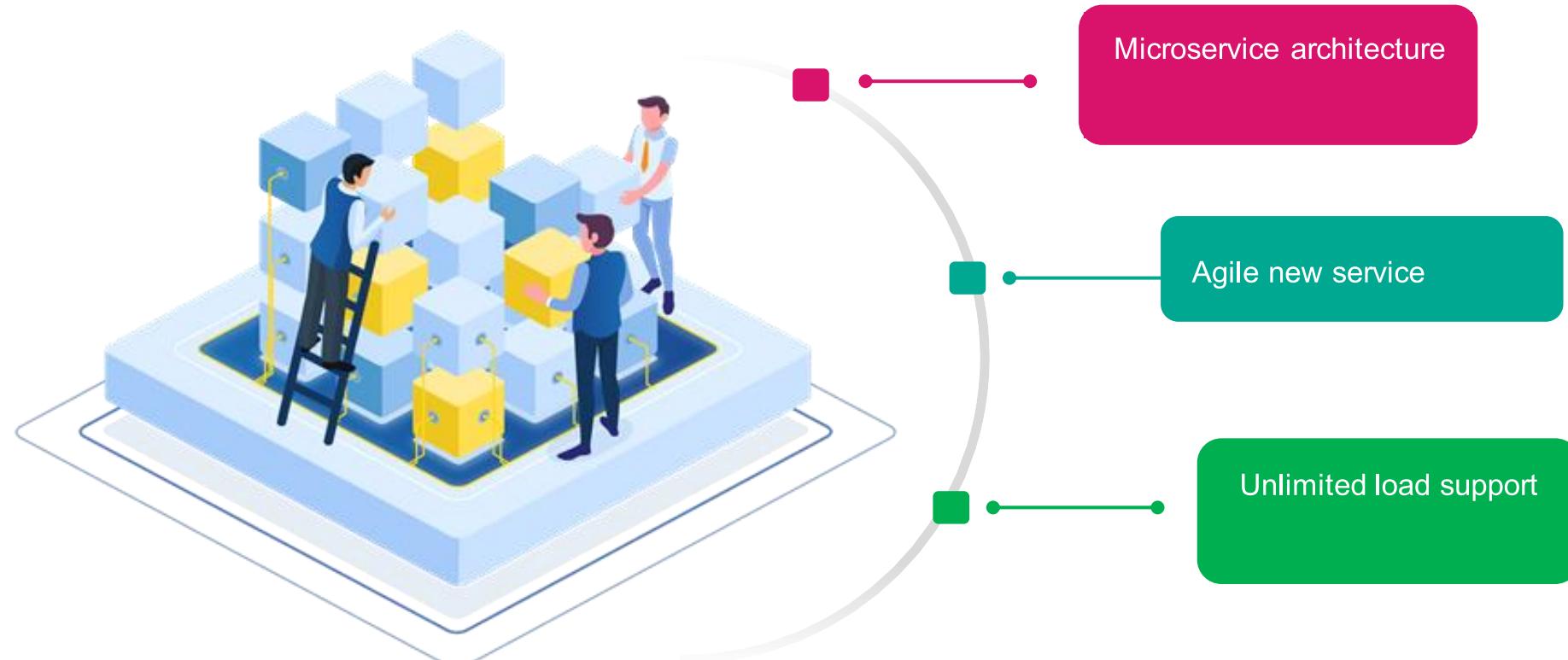
Any MQTT
compatible hardware

Network



ความยืดหยุ่นสูง

ด้วยสถาปัตยกรรมแบบ Microservices ที่ทำงานร่วมกันอย่างอิสระ พลิกโฉมรองรับการปรับแต่ง การเพิ่ม/ลดทรัพยากรที่ใช้แบบอัตโนมัติ (Auto scaling) ซึ่งใช้ Kubernetes ในการจัดการและควบคุมการทำงานของทุก Microservices



พร้อมเชื่อมต่อระบบที่มีอยู่

รองรับการเชื่อมต่อไปยังระบบอื่นๆ ที่มีอยู่เดิม (3rd Party) ด้วย APIs ที่ถูกออกแบบไว้สำหรับ
หลากหลายรูปแบบ

แพลตฟอร์มเป็นศูนย์กลาง

Device Shadow



Device Shadow

Status : On

Brightness : 3,200



Device Schema

Device Schema

What : Brightness

When : Every 10 min

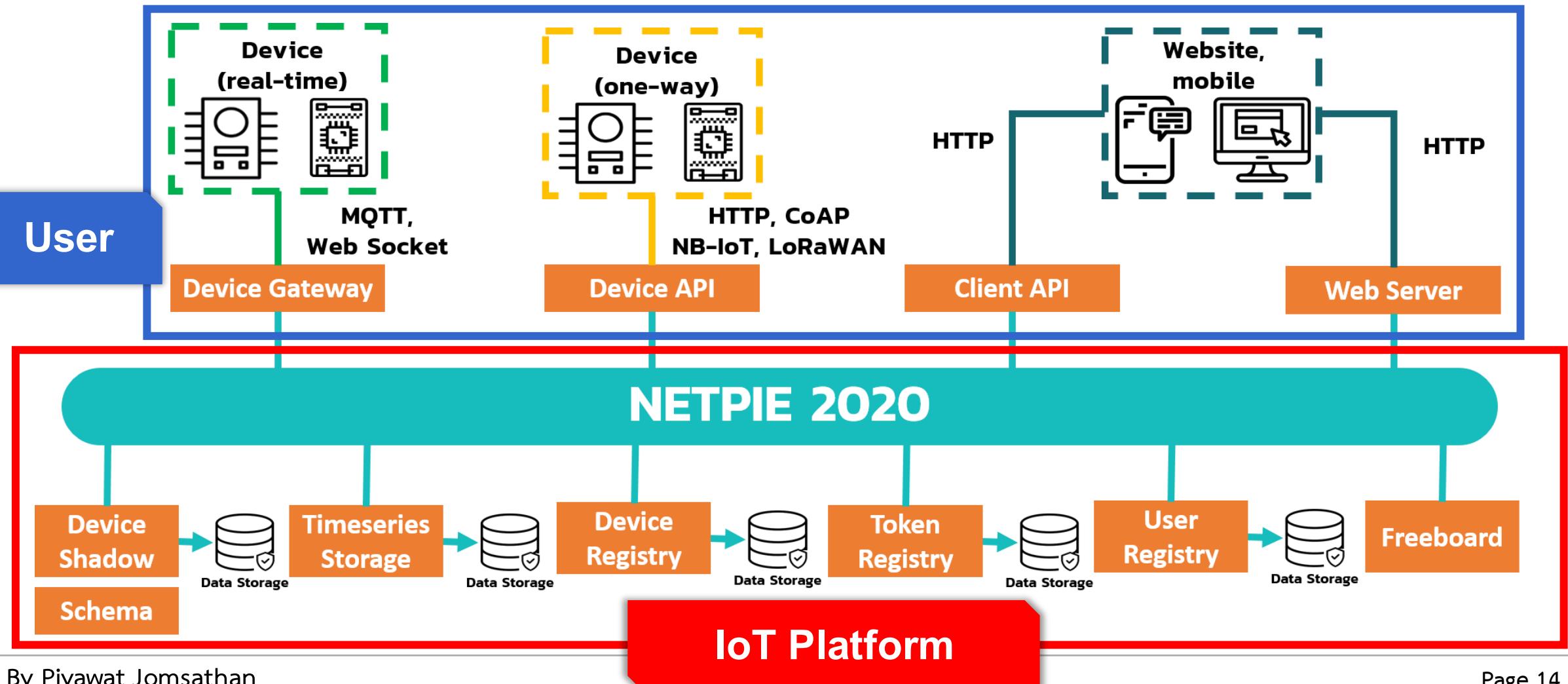
Where : Bedroom

$$\text{How : } E = \frac{\text{Brightness}}{\text{area(m)}^2}$$

Trigger



Architecture ຂອງ NETPIE2020



เริ่มต้นการใช้งาน NETPIE2020

ลงทะเบียน NETPIE2020 ได้ที่
<https://auth.netpie.io/signup>



NETPIE
2020

EMAIL required

NAME required

ORGANIZATION required

COUNTRY CODE

MOBILE PHONE NUMBER* (NO COUNTRY CODE) required and number only

I agree to the [Privacy Statement](#) and [Terms of Use](#)

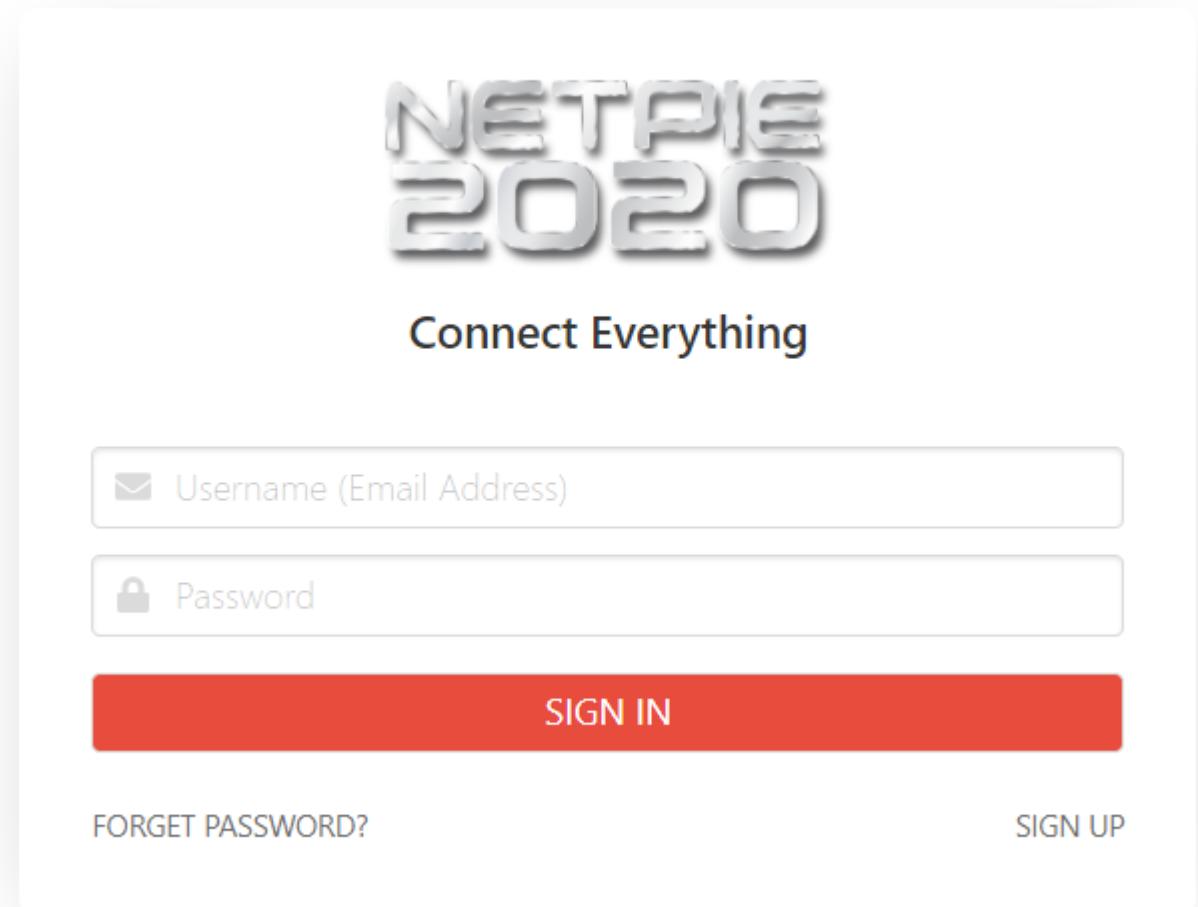
SIGN UP

*Password will be sent to your mobile phone number.

เริ่มต้นการใช้งาน NETPIE2020

เมื่อสมัคร NETPIE2020
แล้วทำการเข้าสู่ระบบ

ซึ่งในปกติสามารถเข้าหน้า login ได้ที่
<https://auth.netpie.io/login>



เริ่มต้นการใช้งาน NETPIE2020

The screenshot shows the main interface of the NETPIE2020 web application. At the top left is the NETPIE logo. At the top right is a user profile icon labeled "Piyawat NECTEC". The central area features the text "Ready to get started?" followed by "You haven't added anything yet." and "Click the button below to add your first project.", with a blue "Add Project" button. At the bottom, it says "Copyright © 2018-2020 Created by NEXPIE Co., Ltd." A blue callout box at the bottom left contains the text "เมื่อเข้ามาแล้วพบหน้าต่างดังรูป ถือว่า Login สำเร็จ".

เมื่อเข้ามาแล้วพบหน้าต่างดังรูป ถือว่า Login สำเร็จ

โครงสร้างของ NETPIE2020

NETPIE2020

Project

Device

Freeboard

Shadow

Schema

Trigger &
Event Hook

Feed

Real Life

โปรเจคที่ต้องการสร้าง

(Smart Home, Smart Farm)

อุปกรณ์ที่จะใช้งาน

(ESP32 Board, ESP8266 Board, PLC, etc..)

หน้าจอแสดงผลหรือ Dashboard

ข้อมูลล่าสุดของ
อุปกรณ์

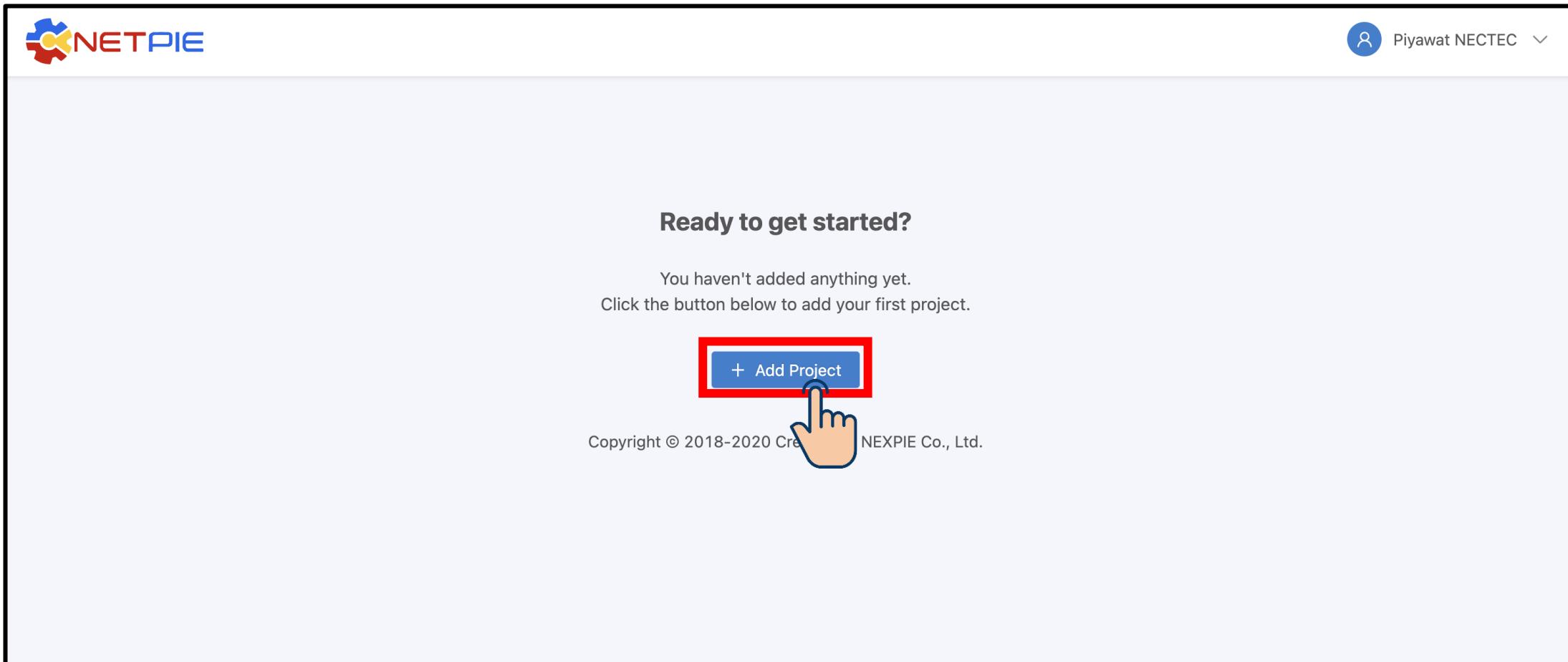
โครงสร้างข้อมูล
ของอุปกรณ์

Trigger &
3rd Party

ฐานข้อมูลเวลา
ของอุปกรณ์

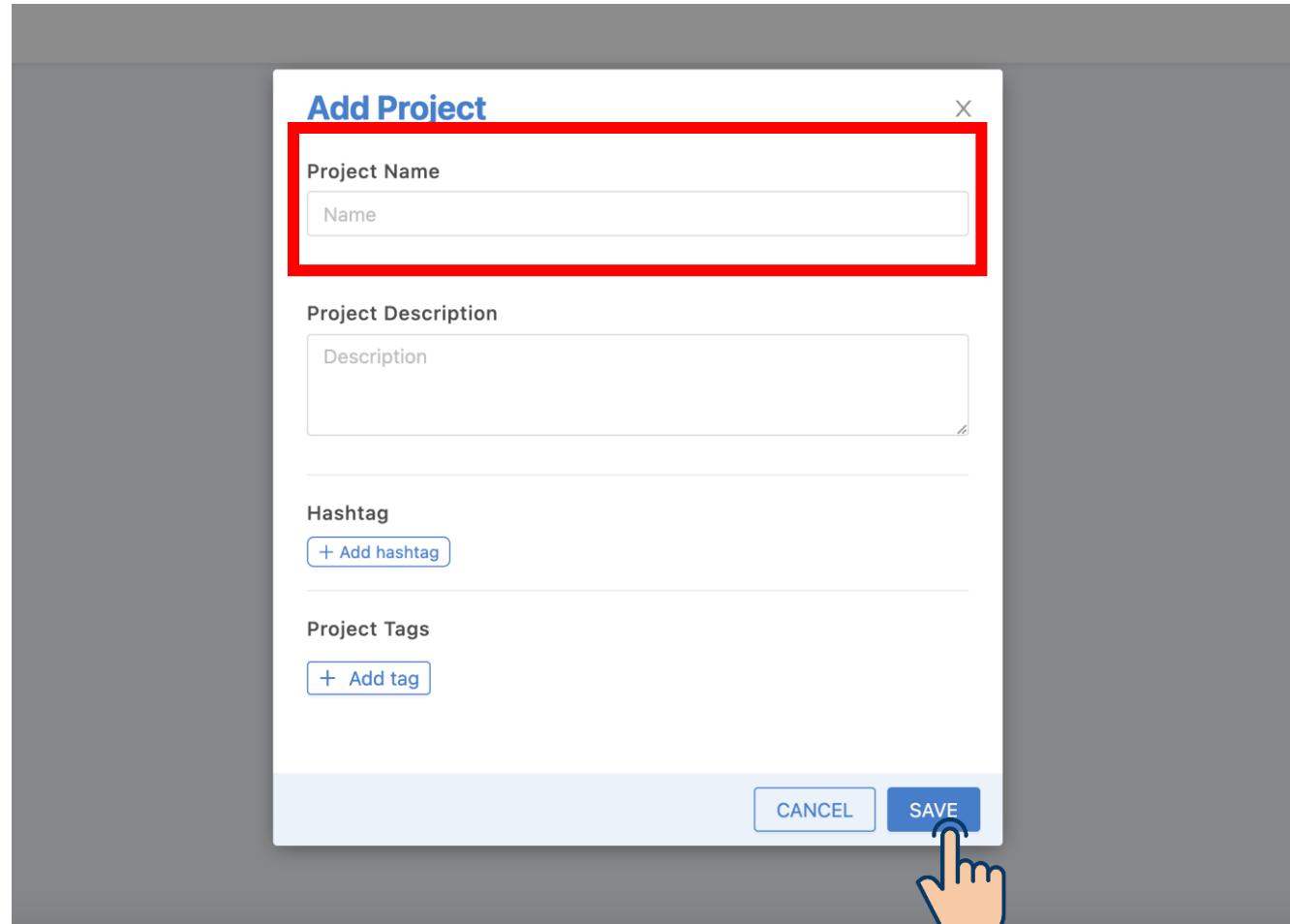
การสร้าง Project

โดยคลิกที่เครื่องหมาย + Add Project ดังรูป เพื่อสร้าง Project



การสร้าง Project

กรอกข้อมูลสำหรับการสร้าง Project

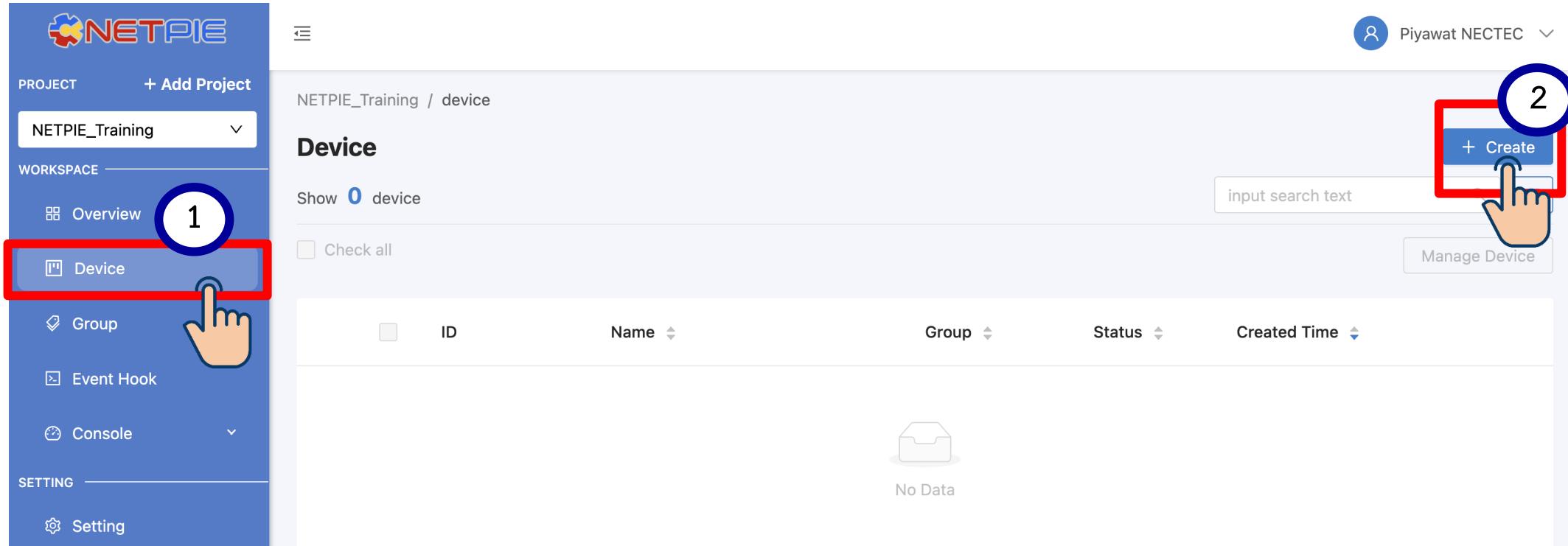


การสร้าง Project

หน้า Overview ของ Project

The screenshot shows the NETPIE platform interface. On the left, there is a sidebar with the NETPIE logo at the top. Below it, the 'PROJECT' section shows a dropdown menu set to 'NETPIE_Training'. Under 'WORKSPACE', the 'Overview' tab is selected, indicated by a blue background. Other tabs include 'Device', 'Group', 'Event Hook', and 'Console'. Under 'SETTING', there is a 'Setting' option. The main content area is titled 'NETPIE_Training / overview'. It displays the project name 'NETPIE_Training' and the creation date 'created date: 2023-01-19'. A 'Detail' section shows the 'Project ID' as 'P065181832677' with a 'Copy' button. To the right, there are three summary cards: 'Devices' (0 devices, 0 online, 0 offline), 'Groups' (0 groups), and another 'Devices' card with the same information.

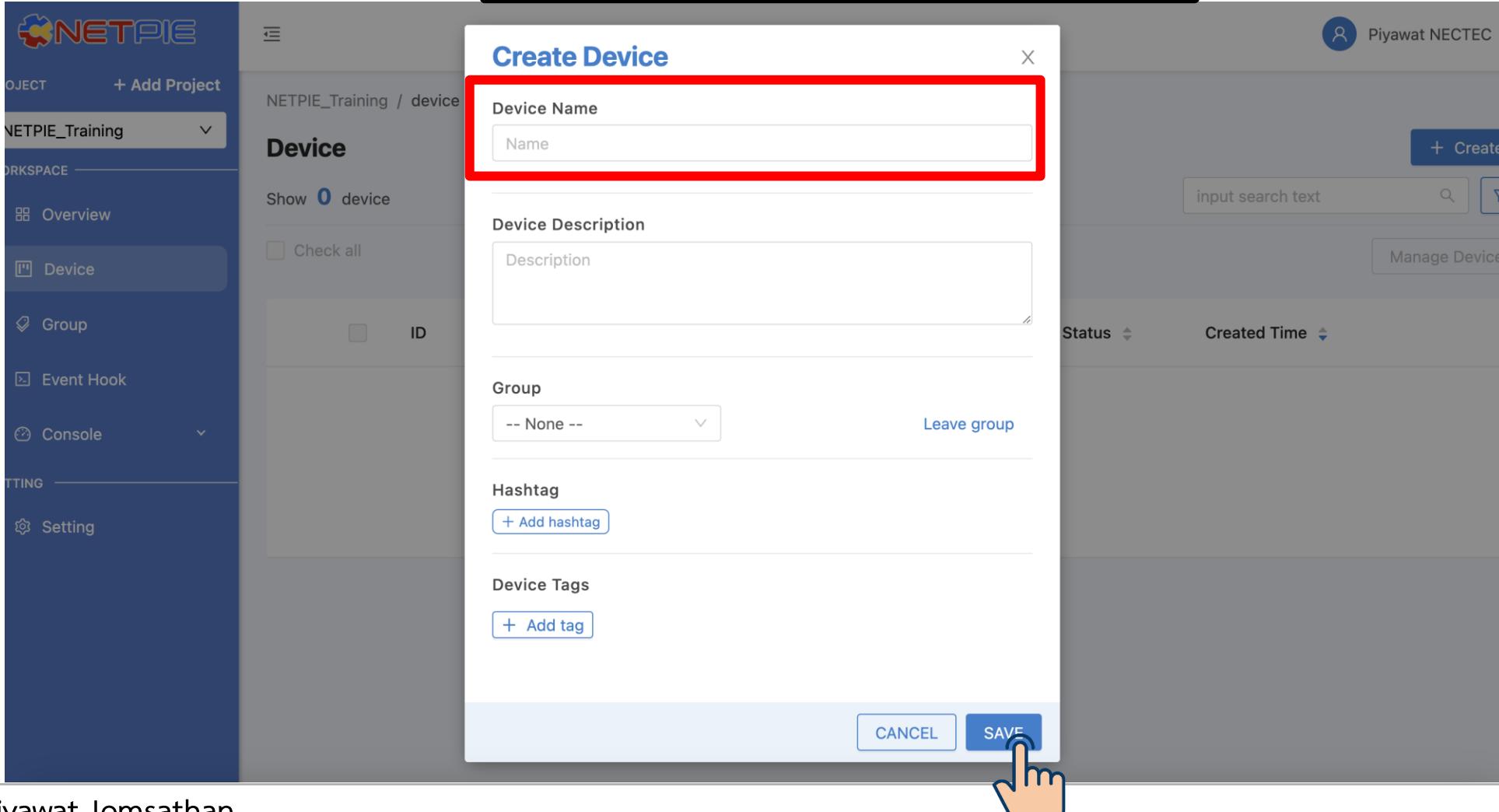
การสร้าง Device



ในการสร้าง Device จะต้องเข้ามายังหน้า Device List จะปรากฏหน้าต่างดังรูป
หลังจากนั้นให้คลิก Create เพื่อทำการสร้าง Device

การสร้าง Device

กรอกข้อมูลสำหรับการสร้าง Device



การสร้าง Device

The screenshot shows the NETPIE Device management interface. On the left, there's a sidebar with 'PROJECT' (NETPIE_Training selected), 'WORKSPACE' (Device selected, highlighted in blue), and 'SETTING'. The main area is titled 'Device' and shows 'NETPIE_Training / device'. It displays one device entry:

ID	Name	Group	Status	Created Time
4391e694-8897-4358-be1d-e7...	Device1	-	Offline	2023-01-19 11:47

A red box highlights the entire row for 'Device1'. A hand cursor icon is positioned over the 'Device1' name. Below the table, there are pagination controls: '1-1 of 1 items' and '10 / page'.

หลังจากสร้าง Device แล้วให้คลิกที่แบบ Device เพื่อดูข้อมูลของ Device

การสร้าง Device

The screenshot shows the NETPIE web interface for creating a device. On the left, there's a sidebar with 'PROJECT' (NETPIE_Training), 'WORKSPACE' (Overview, Device, Group, Event Hook, Console), and a 'Detail' section. The main area shows 'Device1' created on 2023-01-19. A red box highlights the 'Key' section, which contains 'Client ID', 'Token', and 'Secret' fields, each with a 'Copy' button. The top right shows a user profile (Piyawat NECTEC) and a 'Edit' button.

หลังจากเข้ามา�ัง Device จะพบรายละเอียดต่างๆของ Device นั้นคือ

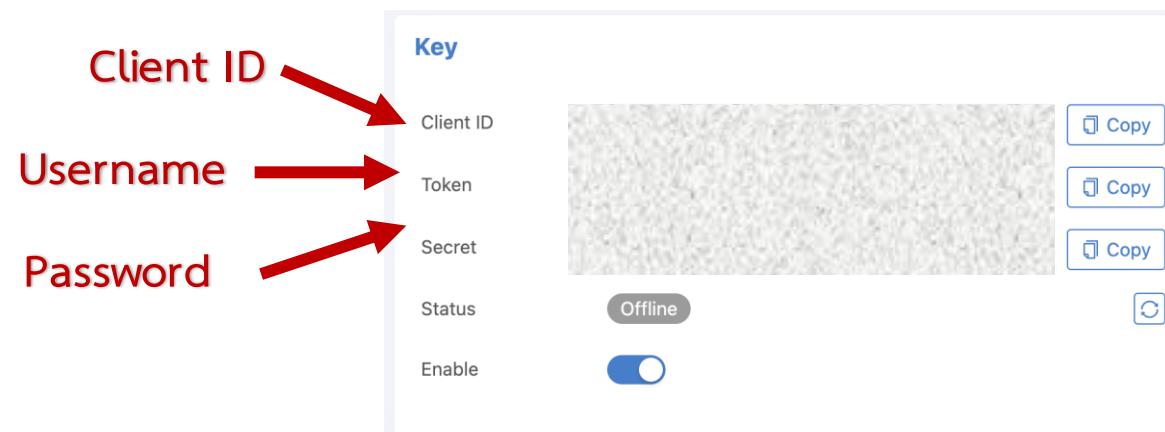
1. Client ID
2. Token
3. Secret

ซึ่งทั้ง 3 Parameter นี้มีส่วนสำคัญในการนำอุปกรณ์ต่อ กับ NETPIE ด้วย Protocol ต่างๆ

การเชื่อมต่อ Device บน NETPIE2020

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)



Example TEST : เชื่อมต่อ NETPIE2020 ด้วย MQTT Box

ทำการทดสอบเชื่อมต่อ Device1 บน NETPIE ด้วยการใช้ MQTT Box



MQTT Box

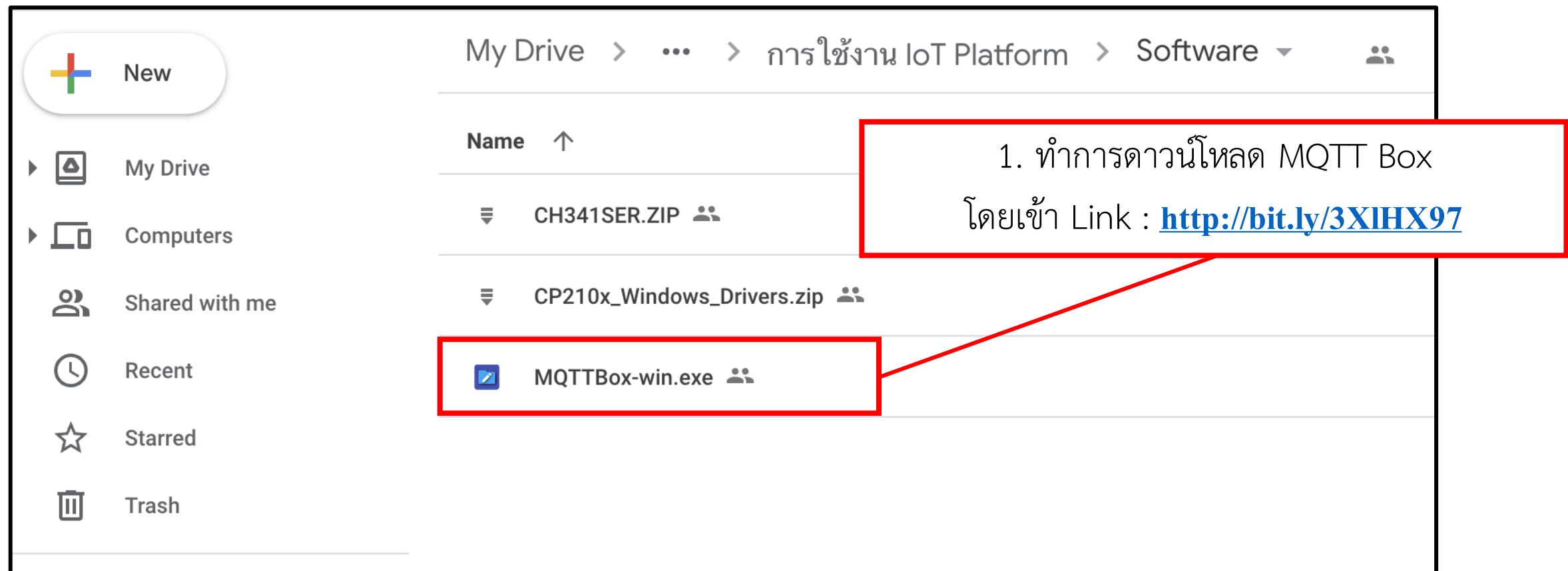
Connect



NETPIE

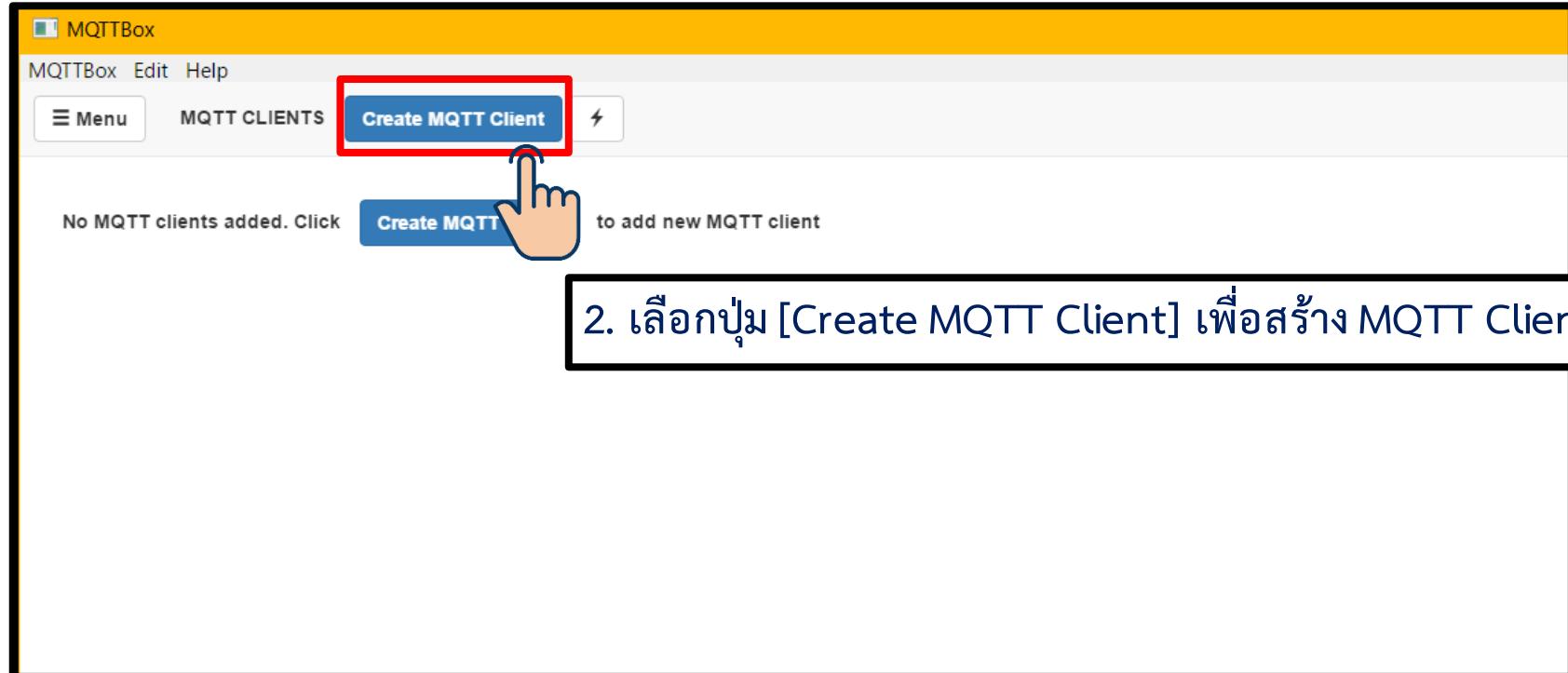
MQTT เป็น Software ที่ใช้สำหรับเชื่อมต่อและสื่อสารผ่าน MQTT Protocol
โดยใน Example นี้เราใช้ MQTT Box แทนการใช้ DEVKIT

Example TEST : เชื่อมต่อ NETPIE2020 ด้วย MQTT Box



Example TEST : เชื่อมต่อ NETPIE2020 ด้วย MQTT Box

1. เมื่อติดตั้งเสร็จทำการเปิด MQTT Box ขึ้นมาจะได้หน้าต่างดังรูป



2. เลือกปุ่ม [Create MQTT Client] เพื่อสร้าง MQTT Client

Example TEST : เชื่อมต่อ NETPIE2020 ด้วย MQTT Box

เลือก Client Name (ต้องเป็นอะไรก็ได้)

Client ID จาก NETPIE

เลือกจาก Yes ให้เป็น No

MQTT Client Name
Device1

Protocol
mqtt / tcp

เลือก mqtt/tcp

Host
mqtt.netpie.io

Username

Reconnect Period (milliseconds)
1000

Token จาก NETPIE

Append timestamp to MQTT client id?

No

Clean Session?
 Yes

Broker is MQTT v3.1.1 compliant?
 Yes

Auto connect on app launch?
 Yes

outgoing QoS zero messages?

Connect Timeout (milliseconds)
30000

KeepAlive (seconds)
10

Will - QoS
0 - Almost Once

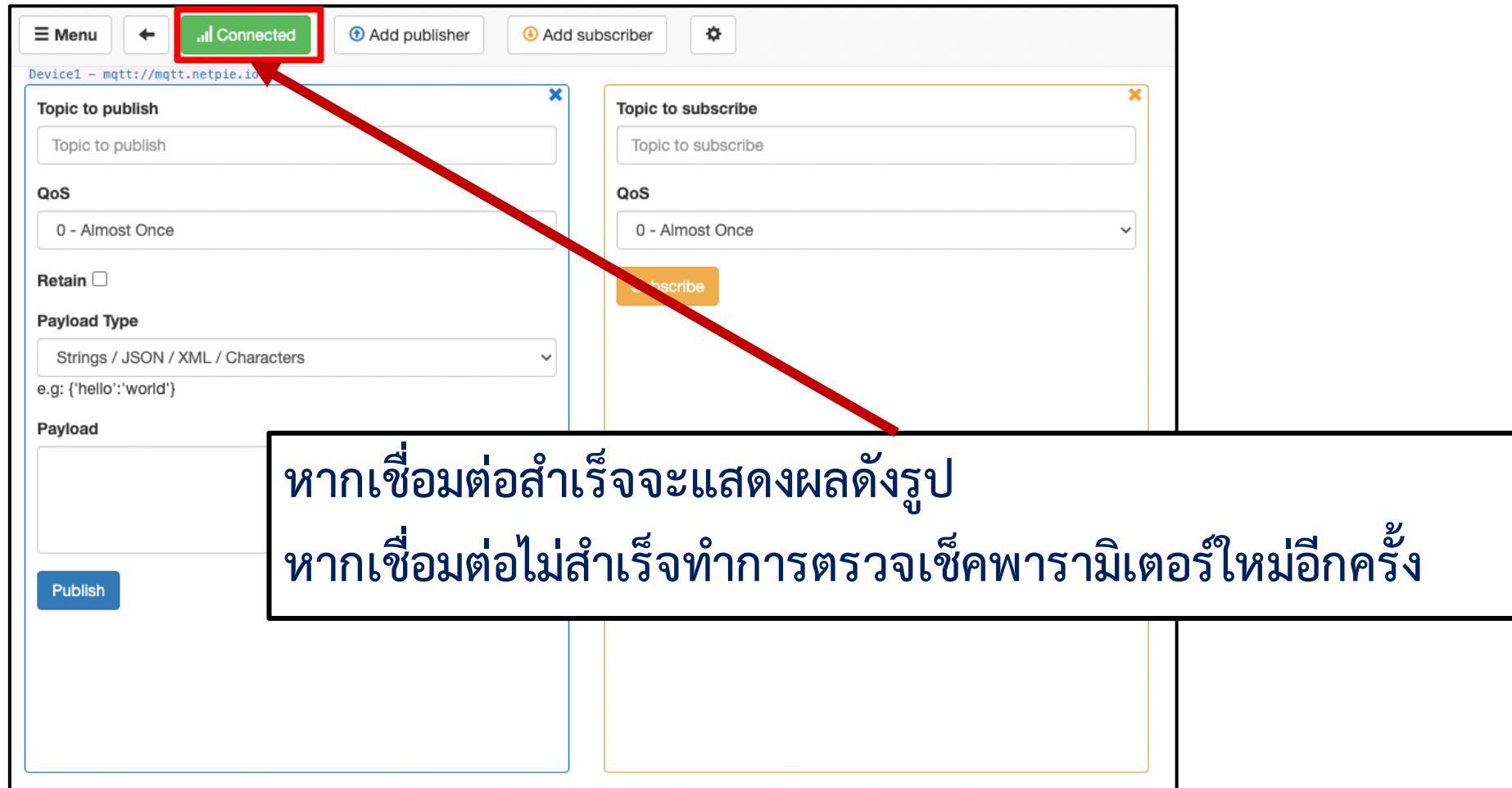
Will - Retain
 No

Will - Payload

Save

เลือก [Save] เมื่อกรอกข้อมูลครบถ้วน

Example TEST : เชื่อมต่อ NETPIE2020 ด้วย MQTT Box



4 – NETPIE2020

Example TEST : เชื่อมต่อ NETPIE2020 ด้วย MQTT Box

The screenshot shows the NETPIE web interface. On the left, there's a sidebar with 'PROJECT' (NETPIE_Training selected), 'WORKSPACE' (Overview, Device, Group, Event Hook, Console), and 'SETTING' (Setting). The main area shows 'NETPIE_Training / device / Device1'. The 'Device1' card has a 'Detail' section and a 'Key' section. The 'Key' section lists 'Client ID', 'Token', 'Secret', 'Status' (which is highlighted with a red box and a red arrow points to it), and 'Enable'. To the right of the 'Status' field are three 'Copy' buttons and a refresh icon. At the bottom of the page, there's a callout box with the text 'ตรวจสอบสถานการณ์เชื่อมต่อสำเร็จบน NETPIE'.

ตรวจสอบสถานการณ์เชื่อมต่อสำเร็จบน NETPIE

การสื่อสารบนระหว่าง Device และ Cloud
บนระบบ Internet of Things (IoT Protocol)



IoT Protocol คืออะไร

IoT Protocol คือ ข้อกำหนดหรือข้อตกลงในการสื่อสารระหว่างอุปกรณ์ หรือภาษาสื่อสารที่ใช้เป็น ภาษากลางในการสื่อสารระหว่างอุปกรณ์ด้วยกัน การที่อุปกรณ์ที่ถูกเชื่อมโยงกันไว้ในระบบจะสามารถติดต่อสื่อสารกันได้นั้น จะเป็นจะต้องมีการสื่อสารที่เรียกว่า โปรโตคอล (Protocol) เช่นเดียวกับคนเราที่ต้องมีภาษาพูดเพื่อให้สื่อสารเข้าใจกันได้ ซึ่งอยู่หลักหลายประเภท ดังนี้

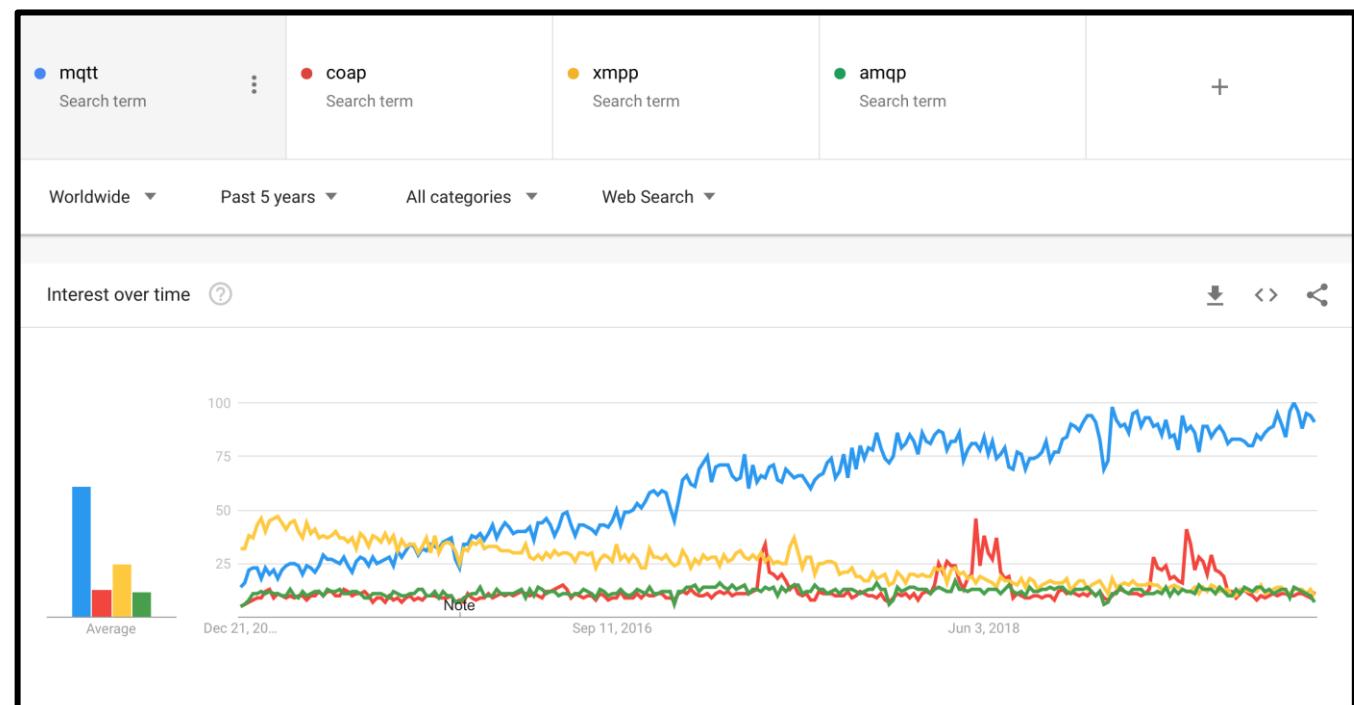
HTTP (Hyper Text Transfer Protocol)

MQTT (Message Queue Telemetry Transport)

CoAP (Constrained Application Protocol)

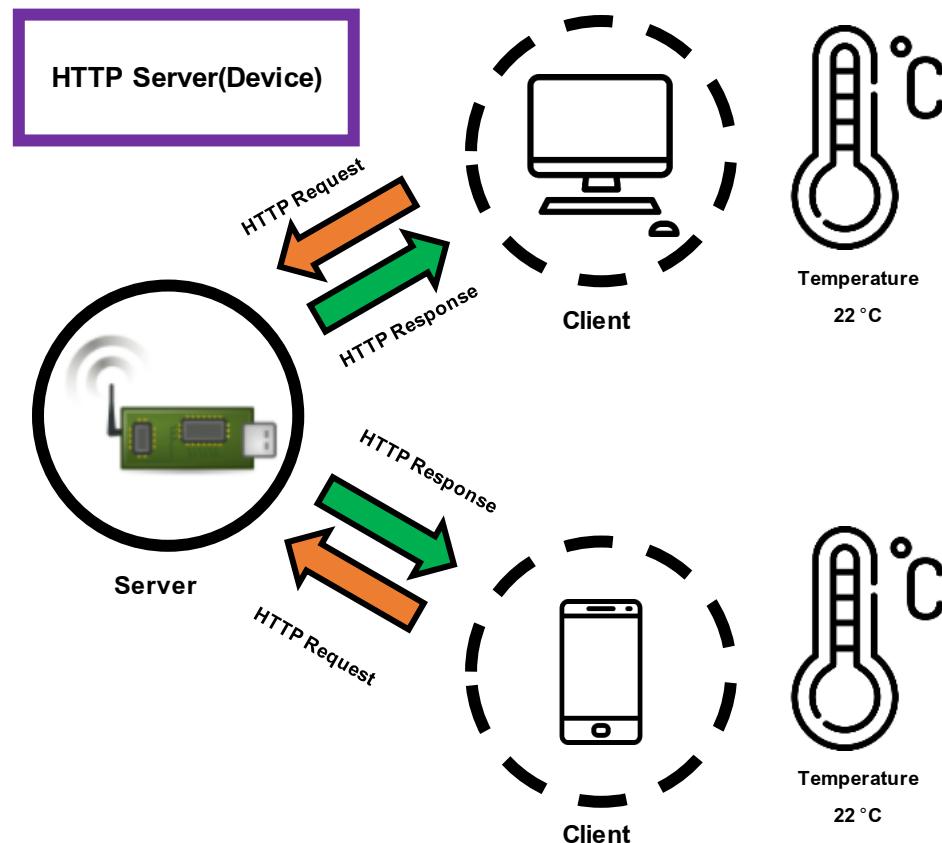
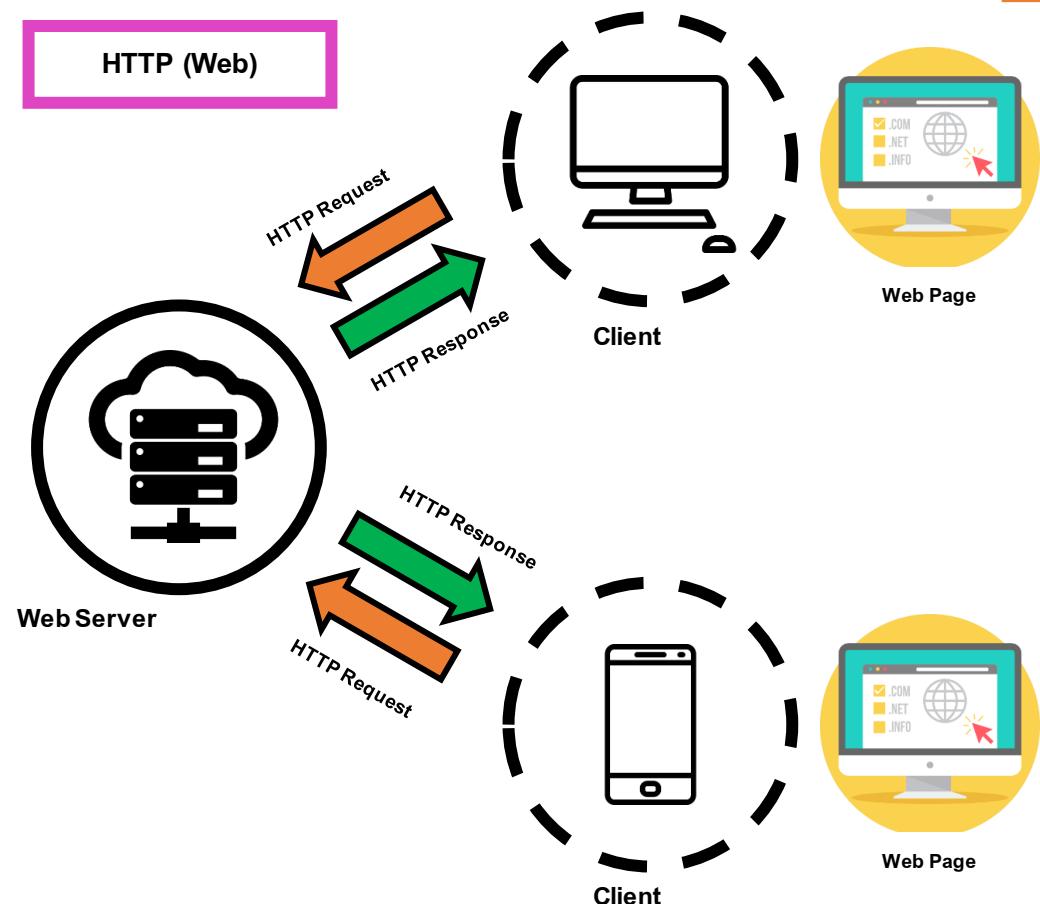
XMPP (Extensible Messaging and Presence Protocol)

AMQP (Advanced Message Queuing Protocol)



HTTP Client-Server Model

HTTP เป็น Protocol ที่ใช้ในการติดต่อกันระหว่างอุปกรณ์ที่รองรับการเชื่อมต่อ ซึ่งมีส่วนประกอบหลัก ๆ คือ Client กับ Server ซึ่งทำการสื่อสารกันเรียกว่า HTTP Request/Response

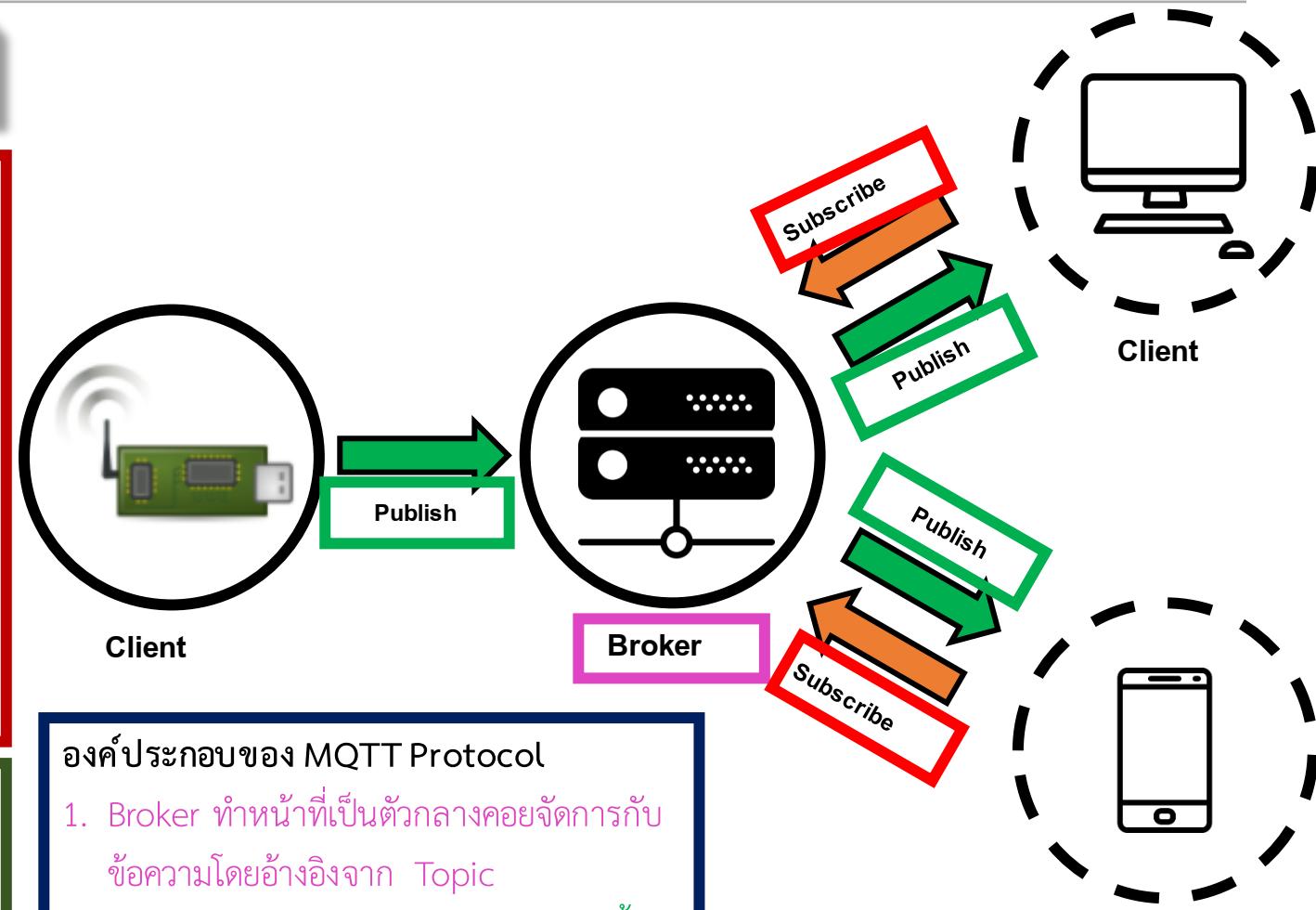


MQTT Publish-Subscribe Model

- MQTT เป็น Protocol ที่ออกแบบมาเพื่อการเชื่อมต่อแบบ M2M หรือ อุปกรณ์กับอุปกรณ์ ซึ่งเป็นการสนับสนุน IoT
- ใช้หลักการการรับส่งข้อมูลแบบ Publish/Subscribe คล้าย กับหลักการที่ใช้ใน Web Service ที่ต้องใช้ Web Server เป็นตัวกลางระหว่างคอมพิวเตอร์ของผู้ใช้งาน
- แต่ MQTT ใช้ตัวกลางที่เรียกว่า Broker ทำหน้าที่จัดการ ลำดับการรับ – ส่ง ข้อมูลระหว่างอุปกรณ์และทั้งที่เป็น Publish/Subscribe

ข้อดีของ MQTT

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

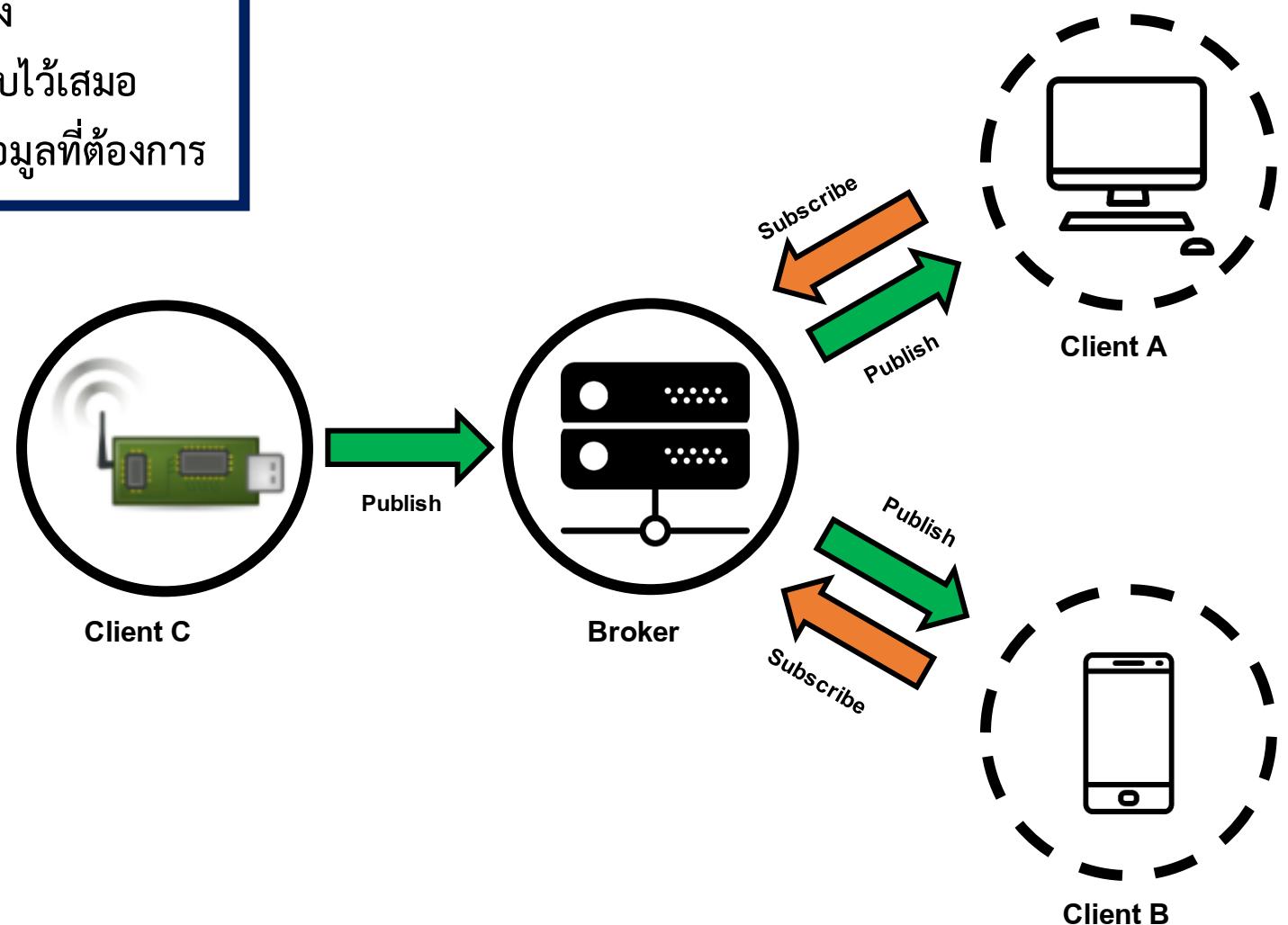


องค์ประกอบของ MQTT Protocol

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

การส่งข้อมูลบน MQTT จะมี Topic เป็นตัวอ้างอิงหลัก ซึ่ง

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

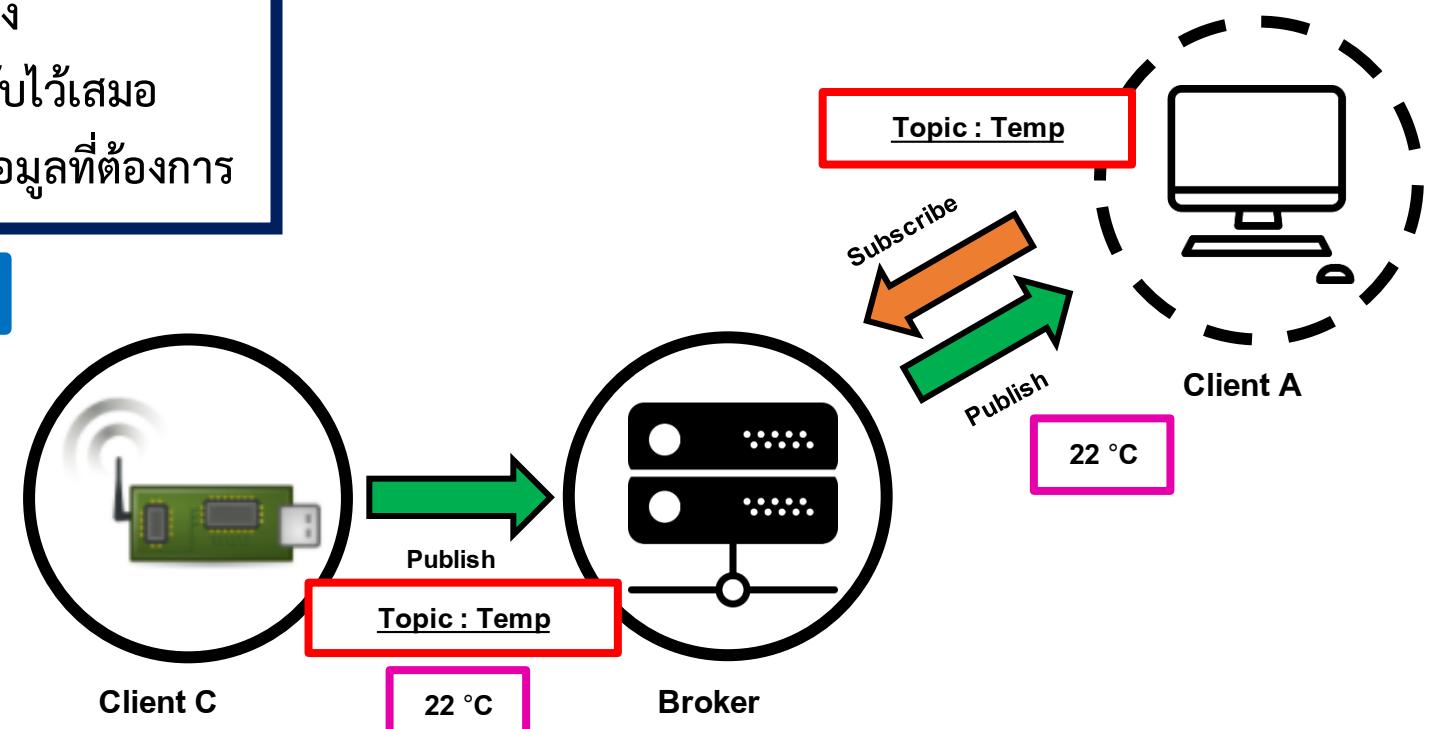


การส่งข้อมูลบน MQTT จะมี Topic เป็นตัวอ้างอิงหลัก ซึ่ง

- 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

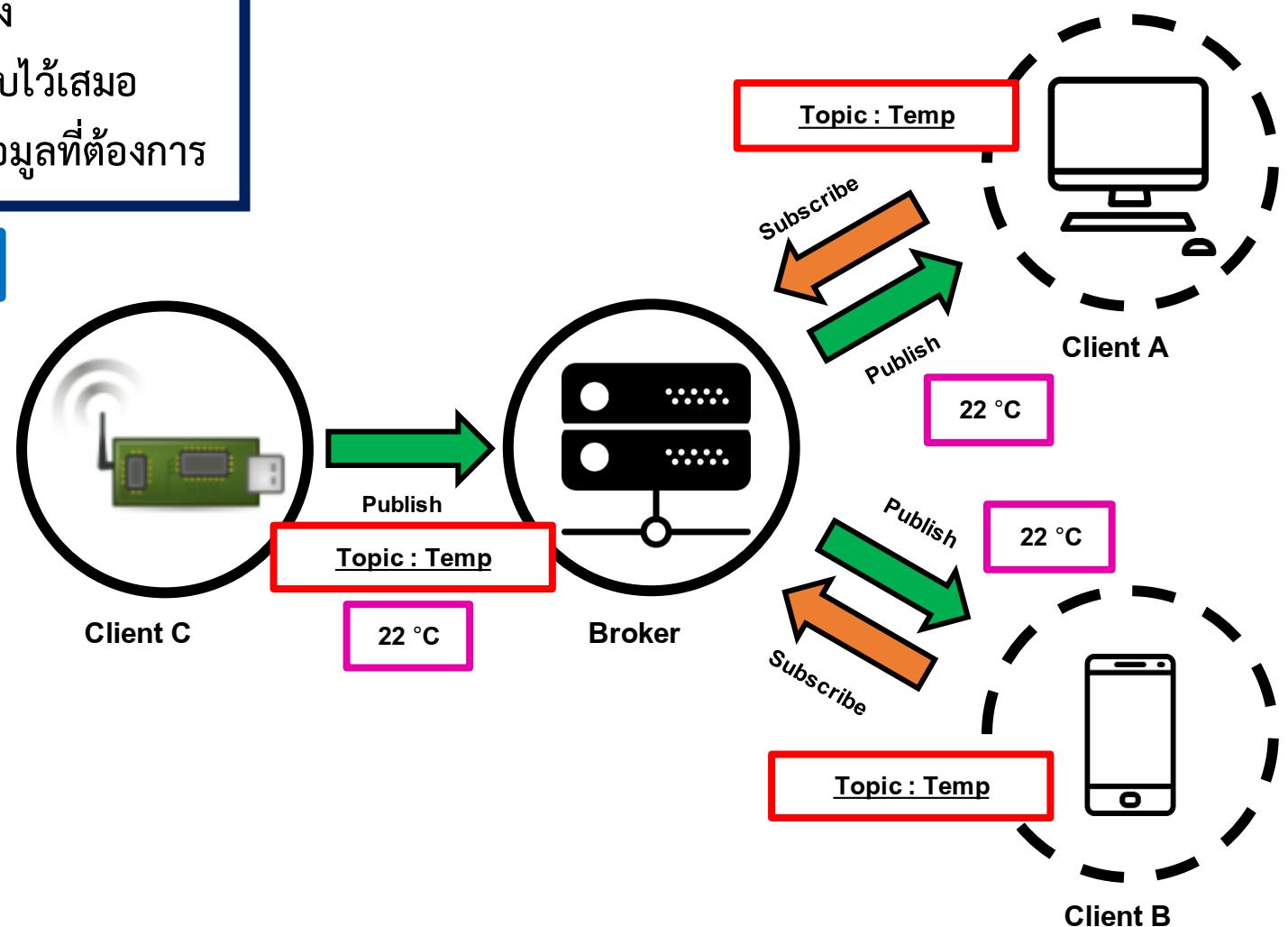


การส่งข้อมูลบน MQTT จะมี Topic เป็นตัวอ้างอิงหลัก ซึ่ง

- 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 ทันทีหลังจากการ Subscribe



4 – NETPIE2020

การส่งข้อมูลบน MQTT จะมี Topic เป็นตัวอ้างอิงหลัก ซึ่ง

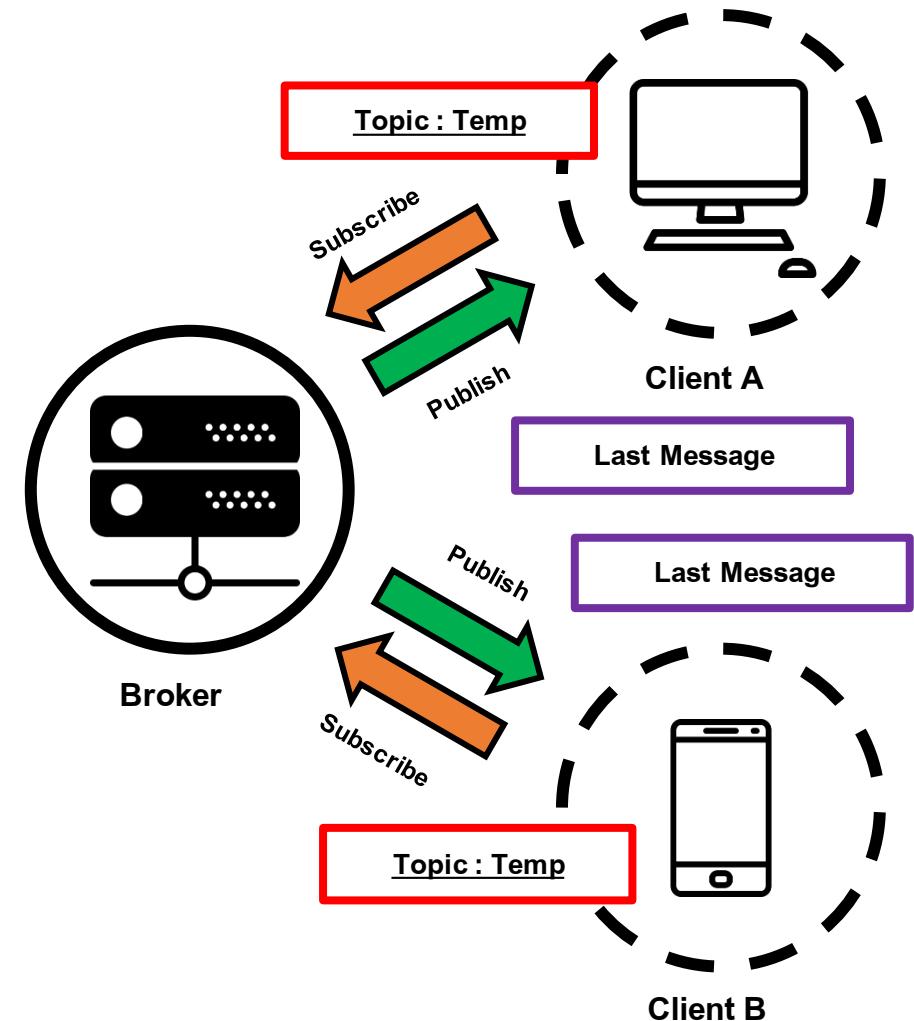
- ข้อมูลที่จะ Publish ไปยัง Broker ต้องมี Topic กำกับไว้เสมอ
- ส่วนของ Subscribe ก็ต้องอ้างอิง Topic เพื่อเรียกข้อมูลที่ต้องการ

ยกตัวอย่างการส่งข้อมูลแบบ MQTT

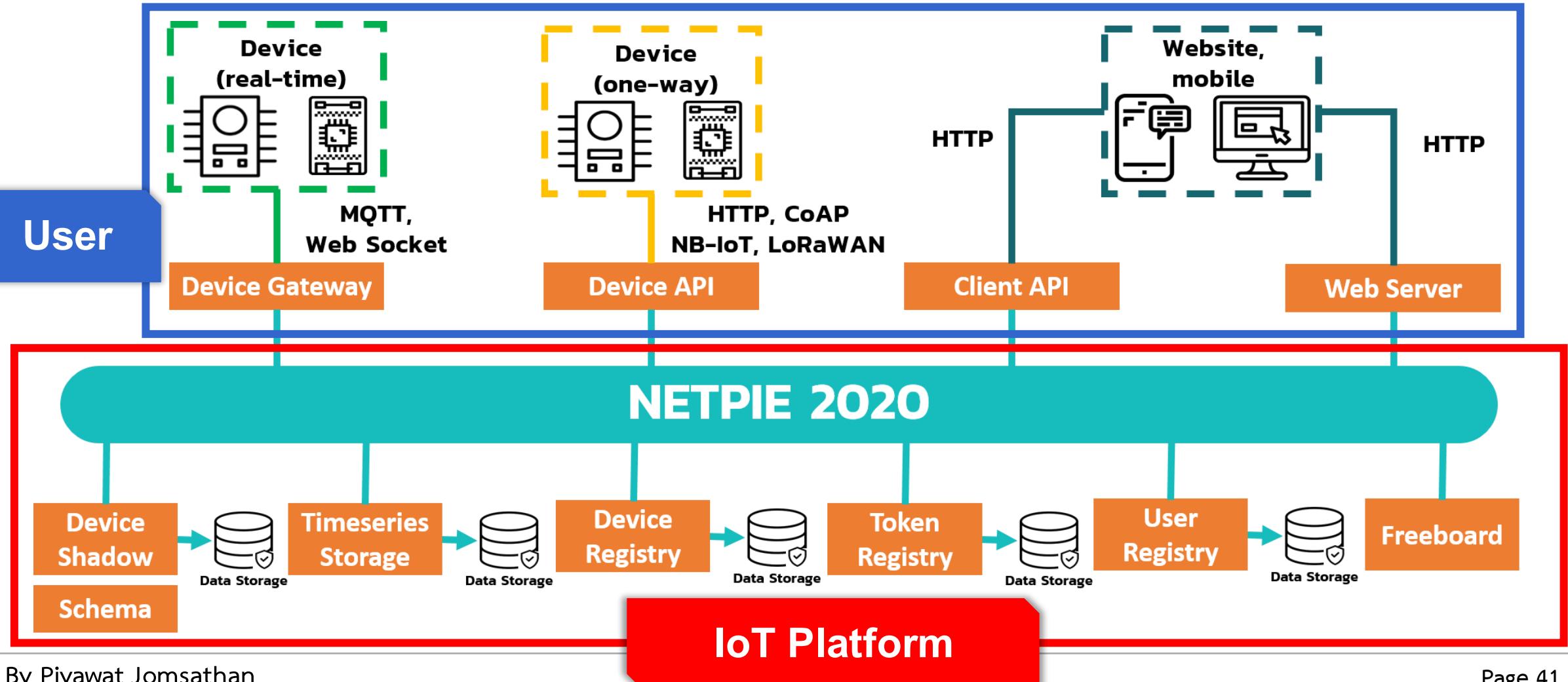
- Client C นั้น Publish ข้อมูลบน Topic ที่มีชื่อว่า Temp และมี Client A ที่ Subscribe Topic Temp ไว้อยู่
- โดยข้อมูลจาก Client C นั้นคือ อุณหภูมิ ที่มีค่าเท่ากับ 22°C ดังนั้น Client A จะได้รับข้อมูลด้วย
- ภายหลัง Client B ทำการ Subscribe Topic Temp ด้วย ดังนั้นข้อมูล อุณหภูมิที่ถูกเก็บไว้จะถูกส่งไปยัง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



Client C

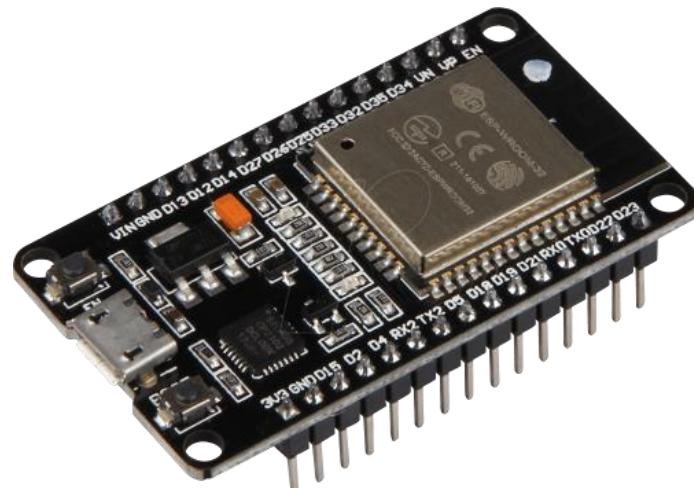


Architecture ຂອງ NETPIE2020



Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT

ทำการทดสอบเชื่อมต่อ Device2 บน NETPIE (ต้องสร้าง Device2 ขึ้นมาใหม่ก่อน)



DEVKIT

เชื่อมต่อ MQTT Server บน
NETPIE2020 ด้วย DEVKIT

Connect



NETPIE

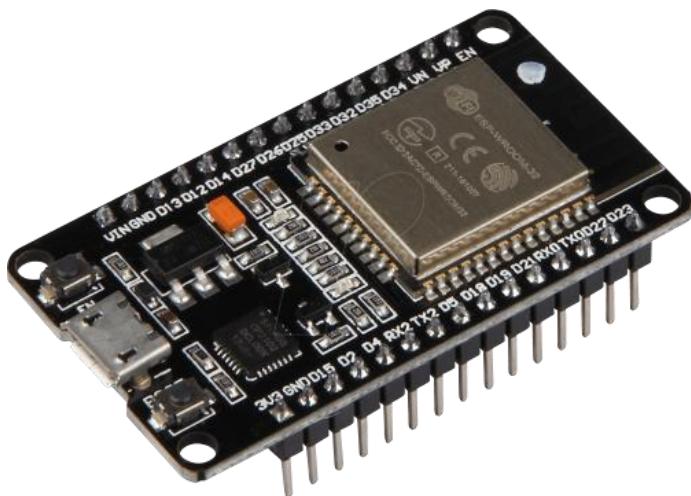
Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT

The screenshot shows the NETPIE Device management interface. On the left, there's a sidebar with 'PROJECT' (NETPIE_Training), 'WORKSPACE' (Overview, Device selected), 'Group', 'Event Hook', 'Console', and 'SETTING' (Setting). The main area is titled 'Device' and shows 'NETPIE_Training / device'. It displays 'Show 2 devices' with a search bar and a 'Create' button. A table lists devices with columns: ID, Name, Group, Status, and Created Time. The first device, 'Device2' (ID: 04368041-2e05-41e2-881e-500d8d4...), is highlighted with a red box. A callout box over this row contains the text 'ทำการสร้าง Device2 ขึ้นให้ DEVKIT เชื่อมต่อ'. The second device has an ID starting with 4391e694-8897-58-be1d-e797c3. At the bottom, there are navigation links for page 1 of 2 items, page 1, page 2, and a '10 / page' dropdown.

ID	Name	Group	Status	Created Time
04368041-2e05-41e2-881e-500d8d4...	Device2	-	Offline	2023-01-19 13:04
4391e694-8897-58-be1d-e797c3				2023-01-19 11:47

ทำการสร้าง Device2 ขึ้นให้ DEVKIT เชื่อมต่อ

Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT



DEVKIT

ในการจะให้ DEVKIT สามารถเชื่อมต่อกับ NETPIE ได้นั้นจำเป็นต้องใช้ Library คือ

1. WiFi

ใช้สำหรับให้ DEVKIT เชื่อมต่อกับอินเทอร์เน็ตผ่าน WiFi

2. PubSubClient

ใช้สำหรับให้ DEVKIT เชื่อมต่อและสื่อสารบน NETPIE (MQTT Protocol)

Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT

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)

Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT

```
#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 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);
        }
    }
}

void setup() {
    Serial.begin(9600);
    Serial.println("Starting...");
    if (WiFi.begin(ssid, password)) {
        while (WiFi.status() != WL_CONNECTED) {
            delay(1000);
            Serial.print(".");
        }
    }
}

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

Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT

Coding ใน Example17.ino แบ่งออกเป็น 3 ส่วน

1

ส่วนที่ 1 การเรียกใช้ Library และ ประกาศตัวแปร

เป็นการเรียกใช้ Library ต่างๆ

เป็นการประกาศตัวแปรสำหรับเชื่อมต่อ WiFi

เป็นการประกาศตัวแปรสำหรับเชื่อมต่อ MQTT Server

การกำหนดและเรียกใช้ชุดคำสั่งสำหรับ
เชื่อมต่อ 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);
```

Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT

2

ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

```
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);  
        }  
    }  
}
```

ฟังก์ชันการเชื่อมต่อ MQTT Server

เข้าสู่การเชื่อมต่อ MQTT Server
- หากเชื่อมต่อสำเร็จจะแสดงผลว่า “connected”
- หากเชื่อมต่อไม่สำเร็จจะแสดงผลว่า “failed ...” และจะทำการเชื่อมต่อใหม่อัตโนมัติ

Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT

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

Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT

3

ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

ฟังก์ชันการทำงานหลัก

```
void loop()
```

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

เป็นชุดคำสั่งคงสถานะของการเชื่อมต่อและ
การทำงานต่างๆของ MQTT Server

Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT



The screenshot shows a 'Serial Monitor' window with the following interface elements:

- Top bar: 'Output' tab is selected, followed by 'Serial Monitor' and a close button.
- Message input field: 'Message (Enter to send message to 'ESP32 Dev Module' on '/dev/cu.usbserial-0001')'
- Bottom right: 'New Line' button and '9600 baud' dropdown.

The log output is as follows:

```
14:39:58.601 ->      ,000 0 000000Starting...
14:39:59.983 -> .WiFi connected
14:40:00.015 -> IP address:
14:40:00.015 -> 172.20.10.2
14:40:00.047 -> Attempting NETPIE2020 connection000NETPIE2020 connected
```

A callout box with an orange border and text 'ตรวจสอบที่ Serial Monitor' is overlaid on the bottom left of the log area.

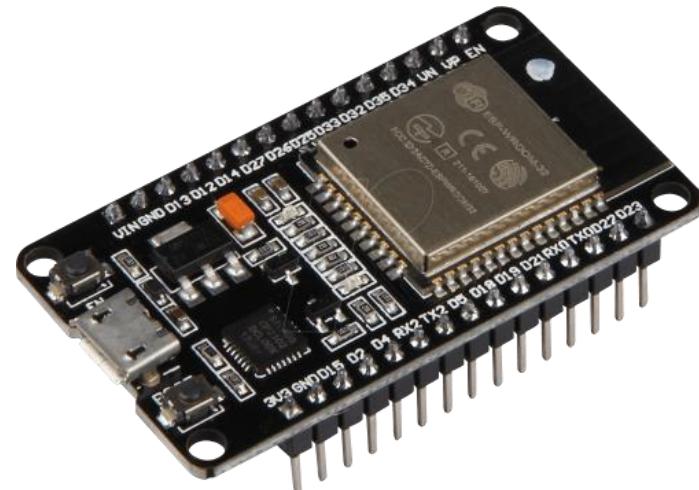
Example 17 : เชื่อมต่อ NETPIE ด้วย DEVKIT

The screenshot shows the NETPIE web interface. On the left, there's a sidebar with 'PROJECT' (NETPIE_Training), 'WORKSPACE' (Overview, Device, Group, Event Hook, Console), and 'SETTING' (Setting). The main area shows 'NETPIE_Training / device / Device2' created on '2023-01-19'. A 'Detail' card is open. To the right is a 'Key' section with fields: Client ID (disabled), Token (disabled), Secret (disabled), Status (green 'Online' button), and Enable (switch). A red arrow points from an orange text box at the bottom to the 'Status' button.

ตรวจสอบสถานะ การเชื่อมต่อนetpie2020

Example 18 : การสื่อสารผ่าน MQTT Protocol บน NETPIE ด้วย DEVKIT

ทดสอบการสื่อสาร Device2 บน NETPIE



DEVKIT

“Hello NETPIE”



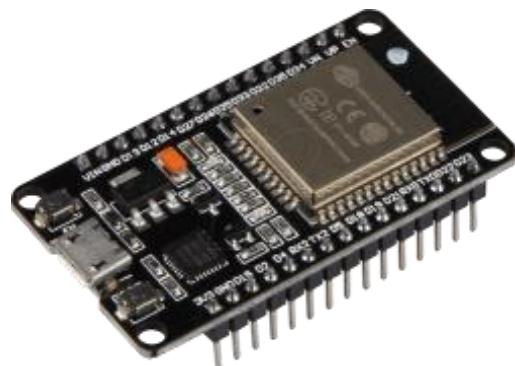
NETPIE

Example 18 : การสื่อสารผ่าน MQTT Protocol บน NETPIE ด้วย DEVKIT

ทำความเข้าใจรูปแบบการสื่อสารบน NETPIE

การจะส่งข้อความไปที่ NETPIE จะต้อง

อ้างอิงหัวข้อเรื่อง ซึ่งเรียกว่า Topic โดยจะต้องขึ้นต้นด้วย @msg/... เช่น



Topic : @msg/test

“Hello NETPIE”



DEVKIT ทำการส่งข้อความไปที่ MQTT Server ด้วยประโยคว่า
“Hello NETPIE”
ซึ่งเรียกว่า การ Publish

NETPIE2020 ทำหน้าที่เปรียบเสมือน
ห้องเช็คห้องหนึ่ง
ซึ่งเรียกว่า Broker

Example 18 : การสื่อสารผ่าน MQTT Protocol บน NETPIE ด้วย DEVKIT

คำสั่งและฟังก์ชันสำคัญใน PubSubClient

client.publish()

client.publish() เป็นคำสั่ง publish ไปยัง Topic ที่ต้องการ
client.publish("topic", "Message")

การใช้งานคำสั่ง Publish

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 หรือหากข้อความมีความซับซ้อนสามารถแปลงข้อมูลจาก String เป็น CharArray เพื่อการจัดการที่ง่ายขึ้น

Example 18 : การสื่อสารผ่าน MQTT Protocol บน NETPIE ด้วย DEVKIT

Coding ใน Example18.ino แบ่งออกเป็น 3 ส่วน

1

ส่วนที่ 1 การเรียกใช้ Library และ ประกาศตัวแปร

```
#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);
```

Example 18 : การสื่อสารผ่าน MQTT Protocol บน NETPIE ด้วย DEVKIT

2

ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

```
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);
        }
    }
}
```

Example 18 : การสื่อสารผ่าน MQTT Protocol บน NETPIE ด้วย DEVKIT

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);
}
```

Example 18 : การสื่อสารผ่าน MQTT Protocol บน NETPIE ด้วย DEVKIT

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);
```

```
}
```

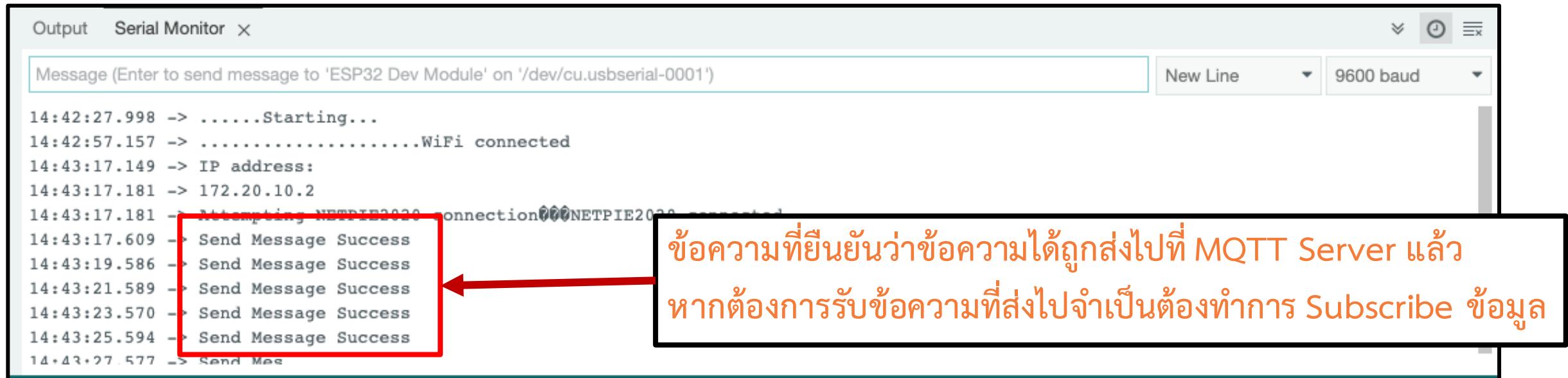
ฟังก์ชันการทำงานหลัก

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

คำสั่งแสดงผลข้อความหลังจาก Publish ข้อมูลสำเร็จ

Example 18 : การสื่อสารผ่าน MQTT Protocol บน NETPIE ด้วย DEVKIT

ตรวจสอบที่ Serial Monitor



The screenshot shows a Serial Monitor window with the following details:

- Output**: Serial Monitor
- Message (Enter to send message to 'ESP32 Dev Module' on '/dev/cu.usbserial-0001')**: An empty input field.
- New Line**: A dropdown set to "New Line".
- 9600 baud**: A dropdown set to "9600 baud".
- Logs (Text Area):**
 - 14:42:27.998 ->Starting...
 - 14:42:57.157 ->WiFi connected
 - 14:43:17.149 -> IP address:
 - 14:43:17.181 -> 172.20.10.2
 - 14:43:17.181 -> Attempting NETPIE2020 connection 000NETPIE2020 connected
 - 14:43:17.609 -> Send Message Success
 - 14:43:19.586 -> Send Message Success
 - 14:43:21.589 -> Send Message Success
 - 14:43:23.570 -> Send Message Success
 - 14:43:25.594 -> Send Message Success
 - 14:43:27.577 -> Send Mes

A red box highlights the "Send Message Success" log entries from 14:43:17.609 to 14:43:25.594. A red arrow points from this highlighted area to a callout box containing the following text:

ข้อความที่ยืนยันว่าข้อความได้ถูกส่งไปที่ MQTT Server แล้ว
หากต้องการรับข้อความที่ส่งไปจำเป็นต้องทำการ Subscribe ข้อมูล

ทำความเข้าใจรูปแบบการสื่อสารบน NETPIE



DEVKIT 1 ทำการ **Publish** ข้อความไปยัง
Topic @msg/test

*Publish Topic : @msg/test
“Hello NETPIE”*

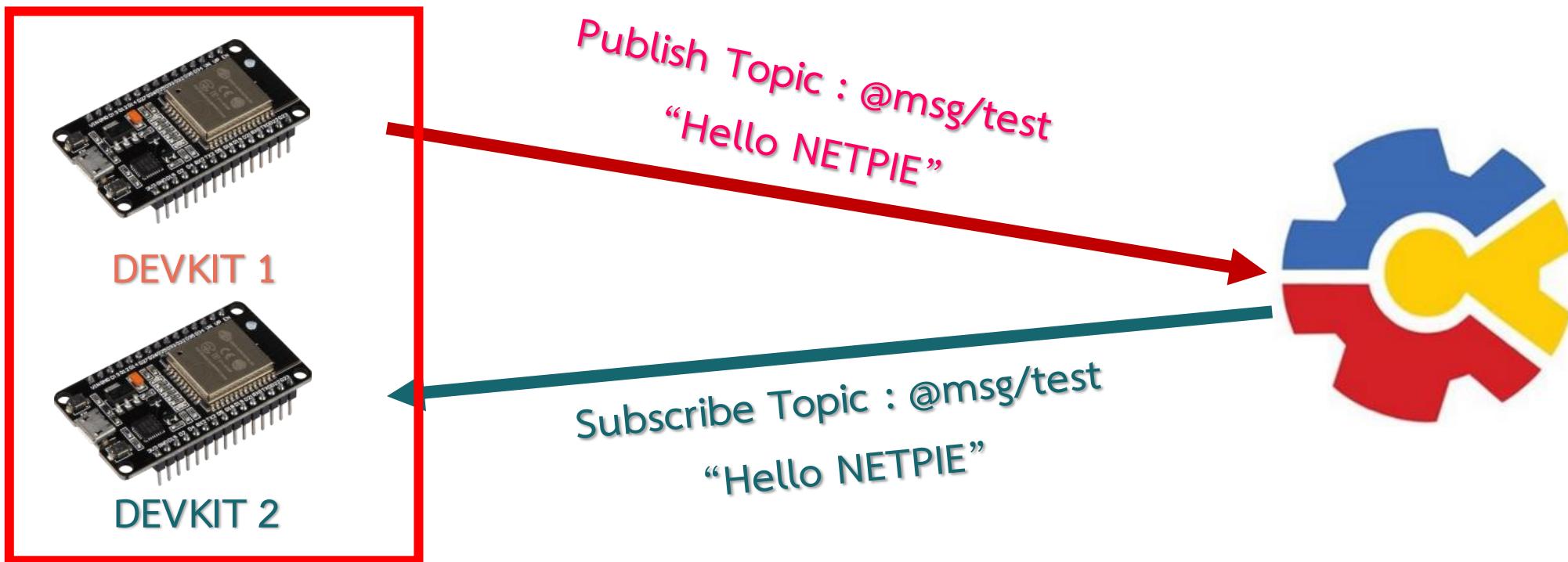


DEVKIT 2 ต้องการข้อความที่ DEVKIT 1 ส่งมาต้องทำการ
เลือกรับข้อความตาม Topic ซึ่งเรียกว่าการ **Subscribe**

*Subscribe Topic : @msg/test
“Hello NETPIE”*



ทำความเข้าใจรูปแบบการลือสารบัน NETPIE



Device ทั้ง 2 จะต้องอยู่ใน Group เดียวกัน

4 – NETPIE2020



DEVKIT 1



DEVKIT 2

ใช้ MQTTBox แทน

“Hello NETPIE”

“Hello NETPIE”



4 – NETPIE2020

The screenshot shows the NETPIE web interface. On the left sidebar, under the WORKSPACE section, the 'Group' option is highlighted with a red box and a circled '1'. A hand cursor is shown clicking on it. At the top right, a user profile for 'Piyawat NECTEC' is visible. In the main content area, the title 'NETPIE_Training / group' is displayed above a table header for 'Group'. The table has columns for ID, Name, Device, and Created Time. Below the table, it says 'No Data'. On the far right, a red box highlights the '+ Create' button, which is also circled with a blue circle and has a hand cursor pointing at it. An input search bar is also present.

4 – NETPIE2020

The screenshot shows the NETPIE web interface. On the left, there's a sidebar with 'PROJECT' (NETPIE_Training selected), 'WORKSPACE' (Overview, Device, Group selected, Event Hook, Console), and 'SETTING' (Setting). The main area is titled 'Group' under 'NETPIE_Training / group'. It shows one group named 'Group1' with ID G655194967146, created on 2023-01-19 14:44, and 0 devices. A red box highlights the first row of the table, and a red arrow points from it to a callout box containing the text: 'Group ที่ได้สร้างขึ้นแต่ใน Group นั้นยังไม่มี Device จึงต้องนำเข้า Device ก่อน'.

ID	Name	Device	Created Time
G655194967146	Group1	0	2023-01-19 14:44

Group ที่ได้สร้างขึ้น
แต่ใน Group นั้นยังไม่มี Device จึงต้องนำเข้า Device ก่อน

4 – NETPIE2020

The screenshot shows the NETPIE Device management interface. On the left, there is a sidebar with sections for PROJECT (+ Add Project), WORKSPACE (NETPIE_Training), and SETTING (Setting). The main area is titled "Device" and shows a table of devices. The table has columns for ID, Name, Group, Status, and Created Time. Two devices are listed: "Device2" (ID: 04368041-2e05-41e2-881e-500d8d4..., Name: Device2, Group: -, Status: Online, Created Time: 2023-01-19 13:04) and "Device1" (ID: 4391e694-8897-4358-be1d-e797c3d..., Name: Device1, Group: -, Status: Online, Created Time: 2023-01-19 11:47). A checkbox labeled "Check all" is checked, and "Selected 2 devices" is displayed. A blue button labeled "+ Create" and a white button labeled "Manage Device" are at the top right. Below the table, there is a search bar with "input search text" and a filter icon. At the bottom, there is a page navigation section with "1-2 of 2 items", page number "1", and "10 / page". A large callout box with the text "เลือก Check Box ของ Device ที่ 2" (Select the Check Box of Device 2) is overlaid on the second device row.

ID	Name	Group	Status	Created Time
04368041-2e05-41e2-881e-500d8d4...	Device2	-	Online	2023-01-19 13:04
4391e694-8897-4358-be1d-e797c3d...	Device1	-	Online	2023-01-19 11:47

เลือก Check Box ของ Device ที่ 2

4 – NETPIE2020

NETPIE

PROJECT + Add Project

NETPIE_Training

WORKSPACE Overview Device Group Event Hook Console

SETTING Setting

NETPIE_Training / device

Device

Show 2 devices

input search text

+ Create

Check all Selected 2 devices

ID	Name	Group	Status	Created Time
04368041-2e05-41e2-881e-500d8d4...	Device2	-	Online	2023-01-19 11:47
4391e694-8897-4358-be1d-e797c3d...	Device1	-	Online	2023-01-19 11:47

เลือก Manage Device

1-2 of 2 items < 1 > 10 / page

Manage Device

4 – NETPIE2020

The screenshot shows the NETPIE Device management interface. On the left, there's a sidebar with 'PROJECT' and 'NETPIE_Training' selected. Under 'WORKSPACE', 'Device' is highlighted. In the main area, it says 'Show 2 devices'. A modal window titled 'Group Device' is open, showing a table with two rows. The first row has a red border and a large orange hand cursor pointing to the 'Name' column, which contains 'Group1'. The second row has a blue checkmark icon in the first column. The modal has 'CANCEL' and 'MOVE' buttons at the bottom. A black callout box with white text 'เลือก Group ที่สร้างขึ้น' (Select the group created) points to the 'Group1' entry. The background shows a list of devices with columns for ID, Name, Device, Status, and Created Time.

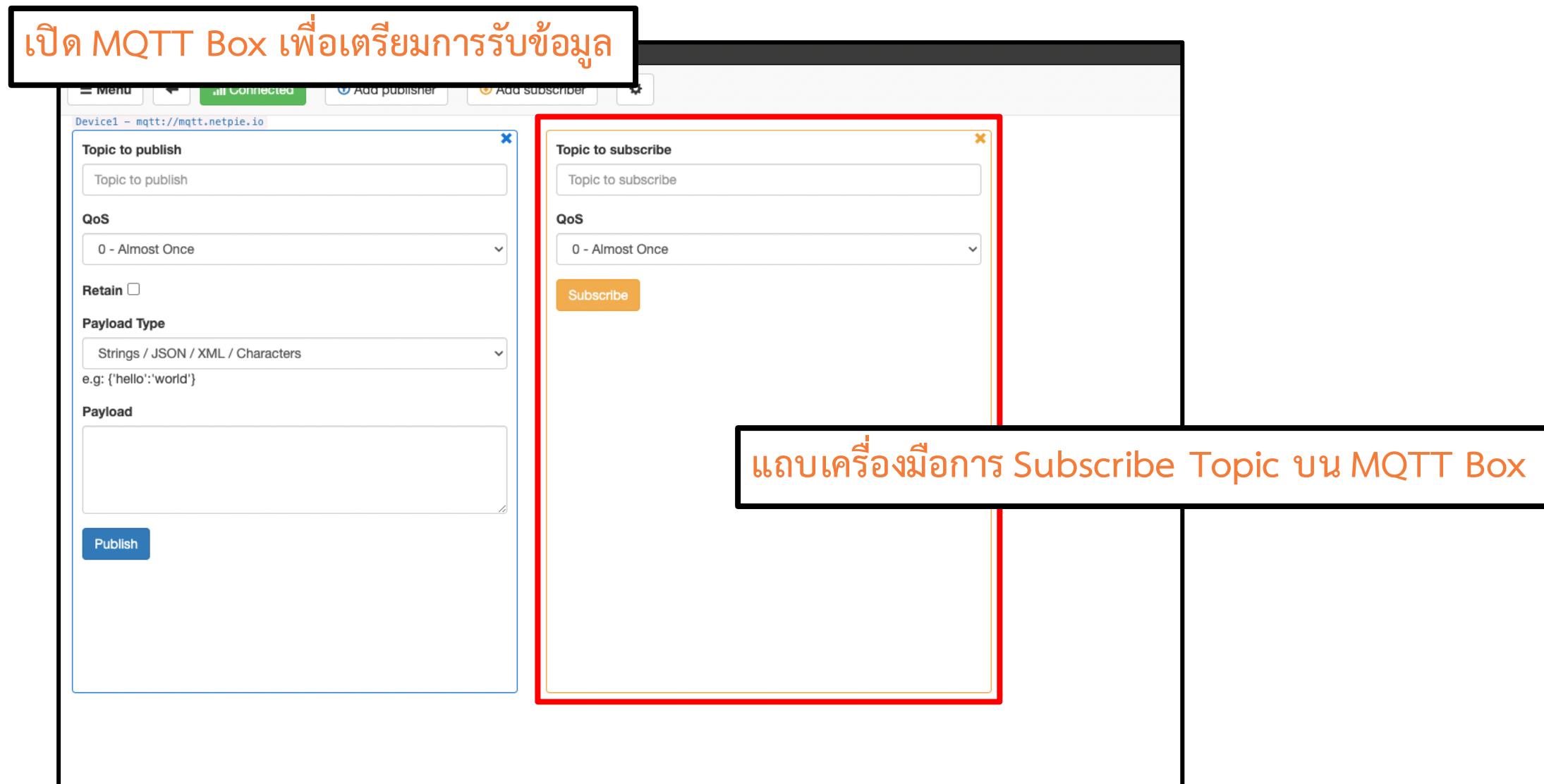
ID	Name	Device	Status	Created Time
04368041-2e05-e2-881e-500d8	Group1	0	Online	2023-01-19 13:04
4391e694-8897-58-be1d-e797c3				

4 – NETPIE2020

The screenshot shows the NETPIE Device management interface. On the left, there's a sidebar with 'PROJECT' (NETPIE_Training selected), 'WORKSPACE' (Overview selected), and 'SETTING' (Setting selected). The main area is titled 'Device' and shows 'NETPIE_Training / device'. It displays 'Show 2 devices' with a checkbox for 'Check all'. There are columns for 'ID', 'Name', 'Group', 'Status', and 'Created Time'. Two devices are listed: 'Device2' (ID: 04368041-2e05-41e2-881e-500d8d4..., Name: -, Group: Group1, Status: Online, Created Time: 2023-01-19 13:04) and 'Device1' (ID: 4391e694-8897-4358-be1d-e797c3d..., Name: -, Group: Group1, Status: Online, Created Time: 2023-01-19 11:47). A red box highlights the 'Group' column for both devices, and a red arrow points from this box to a callout at the bottom. The callout contains the text 'แสดง Group ที่อาศัยอยู่' (Display the group that resides).

ID	Name	Group	Status	Created Time
04368041-2e05-41e2-881e-500d8d4...	Device2 -	Group1	Online	2023-01-19 13:04
4391e694-8897-4358-be1d-e797c3d...	Device1 -	Group1	Online	2023-01-19 11:47

แสดง Group ที่อาศัยอยู่



4 – NETPIE2020

The screenshot shows the NetPie 2020 MQTT interface. On the left, the 'Topic to publish' section is visible, containing fields for 'Topic to publish' (@msg/test), 'QoS' (0 - Almost Once), 'Retain' (unchecked), 'Payload Type' (Strings / JSON / XML / Characters), and a payload area. A 'Publish' button is at the bottom. On the right, the 'Topic to subscribe' section is shown with the topic '@msg/test' highlighted by a red box. A red arrow points from this box to a callout box containing the following text:

กำหนด Topic ที่จะ Subscribe ตาม
ชื่อ Topic ที่ DEVKit ส่งมานั่นคือ
@msg/test
แล้วคลิกที่ Subscribe

4 – NETPIE2020

The screenshot shows the NETPIE2020 MQTT interface. On the left, there is a configuration panel for publishing a message to the topic `@msg/test`. The configuration includes:

- Topic to publish:** `@msg/test`
- QoS:** 0 - Almost Once
- Retain:**
- Payload Type:** Strings / JSON / XML / Characters
- Payload:** (Empty text area)

At the bottom of this panel is a blue **Publish** button.

On the right, a list of received messages is shown, each with a red border around its content area. The messages are identical:

- Topic:** `@msg/test`
- Message Content:** `Hello NETPIE`
- Message Details:** `qos : 0, retain : false, cmd : publish, dup : false, topic : @msg/test, messageId : , length : 23`

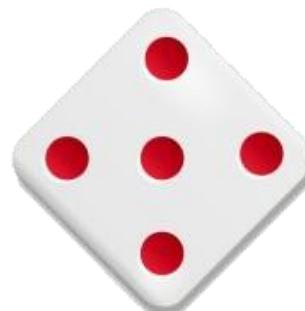
A large red arrow points from the right side of the interface towards a callout box containing the text:

ข้อความที่ถูกส่งมาจาก DEVKIT

Example 19 : การรับข้อมูลจาก MQTTBox ด้วย DEVKIT



DEVKIT 1



MQTTBox

“Hello DEVKIT”

“Hello DEVKIT”



ทำการทดสอบส่งข้อมูลจาก MQTT Box
และให้ DEVKIT รับข้อมูล

Example 19 : การรับข้อมูลจาก MQTTBox ด้วย DEVKIT

คำสั่งและฟังก์ชันสำคัญใน PubSubClient

client.subscribe()

void callback

client.subscribe เป็นคำสั่ง subscribe Topic ที่ต้องการ
ส่วนมากนิยมนำไปไว้ในฟังก์ชัน reconnect รูปแบบคือ

client.subscribe("topic")

callback เป็นฟังก์ชันสำหรับการรับ payload หรือข้อมูล
ต่างๆที่ถูกส่งมาตาม Topic ที่ได้ Subscribe

การใช้งานคำสั่ง Subscribe

1. การจะ Subscribe Topic ได้นั้น อุปกรณ์จำเป็นจะต้องเชื่อมต่อกับ MQTT Server ก่อนเสมอ
2. การ Subscribe Topic นั้น นิยมเขียนไว้ในฟังก์ชัน void reconnect เสมอ
เนื่องจากทุกครั้งที่อุปกรณ์ขาดการเชื่อมต่อจาก MQTT Server อุปกรณ์จะกลับมาทำงานที่ฟังก์ชันนี้เสมอ และทำให้ Topic ที่ Subscribe ขาดการเชื่อมต่อด้วย เมื่ออุปกรณ์เชื่อมต่อใหม่ได้สำเร็จจะได้ทำการ Subscribe Topic ให้ใหม่อัตโนมัติ
3. ข้อมูลที่ได้รับจากการ Subscribe Topic จะถูกแสดงผลในฟังก์ชัน void callback หากต้องการนำข้อมูลมาใช้งานมากเขียนที่ฟังก์ชันนี้

Example 19 : การรับข้อมูลความจาก MQTTBox ด้วย DEVKIT

Coding ใน Example19.ino แบ่งออกเป็น 3 ส่วน

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);
```

Example 19 : การรับข้อมูลความจาก MQTTBox ด้วย DEVKIT

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

Example 19 : การรับข้อมูลจาก MQTTBox ด้วย DEVKIT

2

ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

```
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);  
}
```

ฟังก์ชันการรับข้อมูล void callback

แสดงผล Topic ที่รับมา

แสดงผลข้อมูลที่รับมา

Example 19 : การรับข้อมูลจาก MQTTBox ด้วย DEVKIT

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

Example 19 : การรับข้อมูลความจาก MQTTBox ด้วย DEVKIT

3

ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

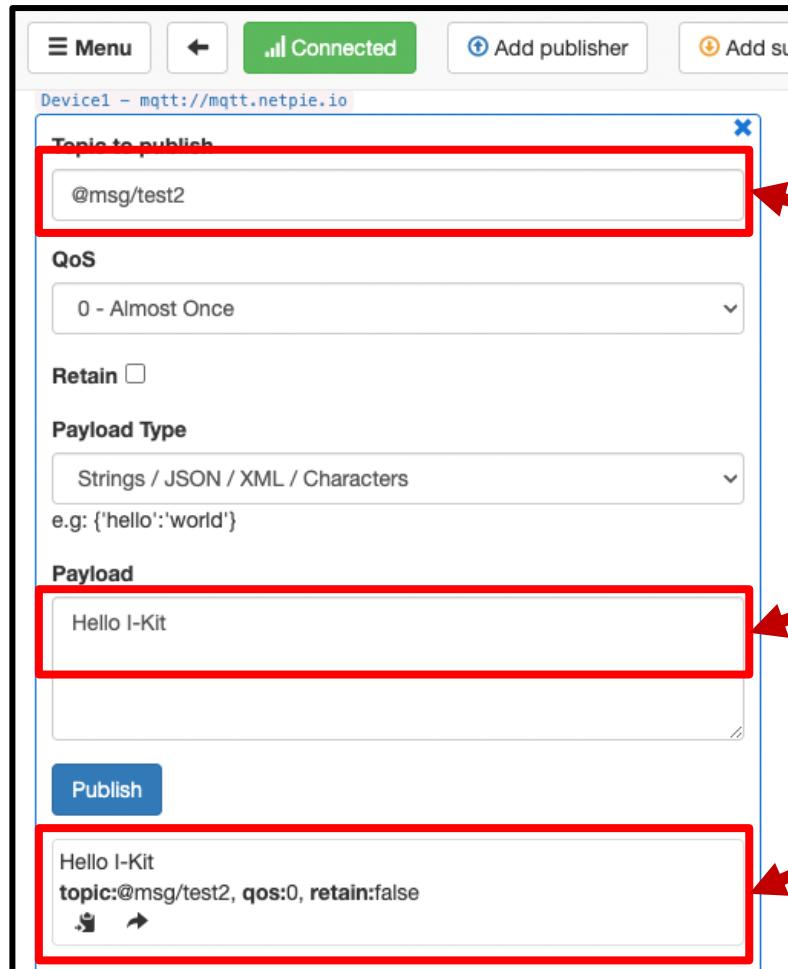
ฟังก์ชันการทำงานหลัก

```
void loop()
```

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

เป็นชุดคำสั่งคงสถานะของการเชื่อมต่อและ
การทำงานต่างๆของ MQTT Server

Example 19 : การรับข้อมูลความจาก MQTTBox ด้วย DEVKIT



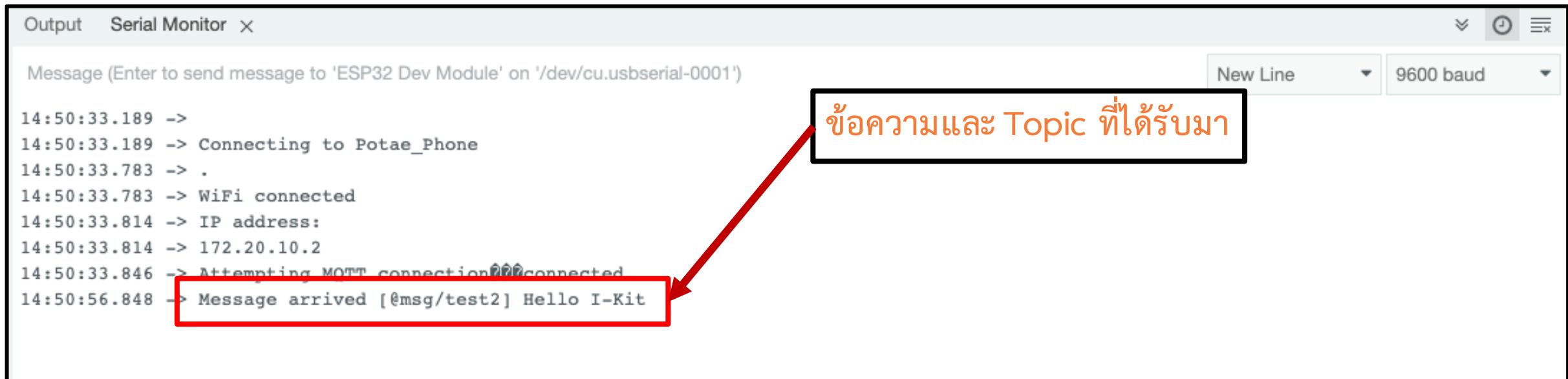
Topic ที่ต้องการส่งข้อมูลไป
ซึ่งต้องตรงกับ Topic ที่ DEVKIT Subscribe อยู่

ข้อมูลที่ต้องการส่ง

สถานะการส่งข้อมูล

Example 19 : การรับข้อมูลจาก MQTTBox ด้วย DEVKIT

ตรวจสอบที่ Serial Monitor



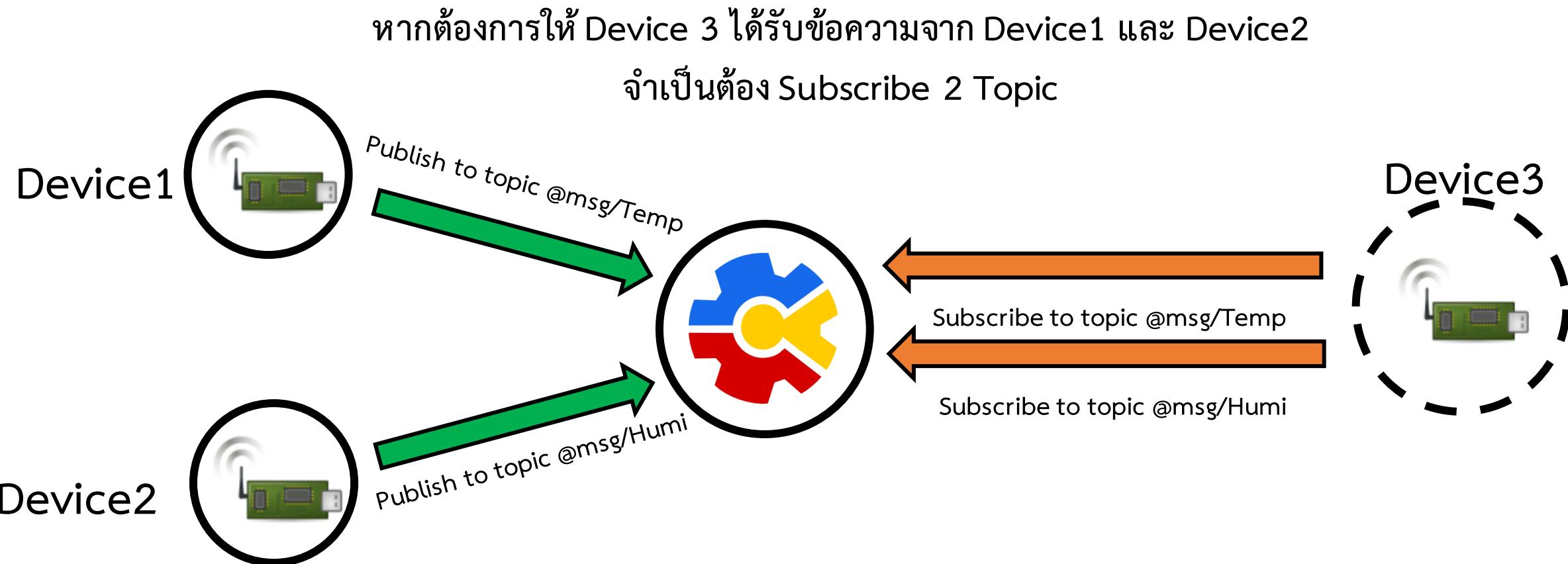
Output Serial Monitor New Line ▾ 9600 baud ▾

Message (Enter to send message to 'ESP32 Dev Module' on '/dev/cu.usbserial-0001')

```
14:50:33.189 ->
14:50:33.189 -> Connecting to Potaе_Phone
14:50:33.783 -> .
14:50:33.783 -> WiFi connected
14:50:33.814 -> IP address:
14:50:33.814 -> 172.20.10.2
14:50:33.846 -> Attempting MQTT connection?connected
14:50:56.848 -> Message arrived [@msg/test2] Hello I-Kit
```

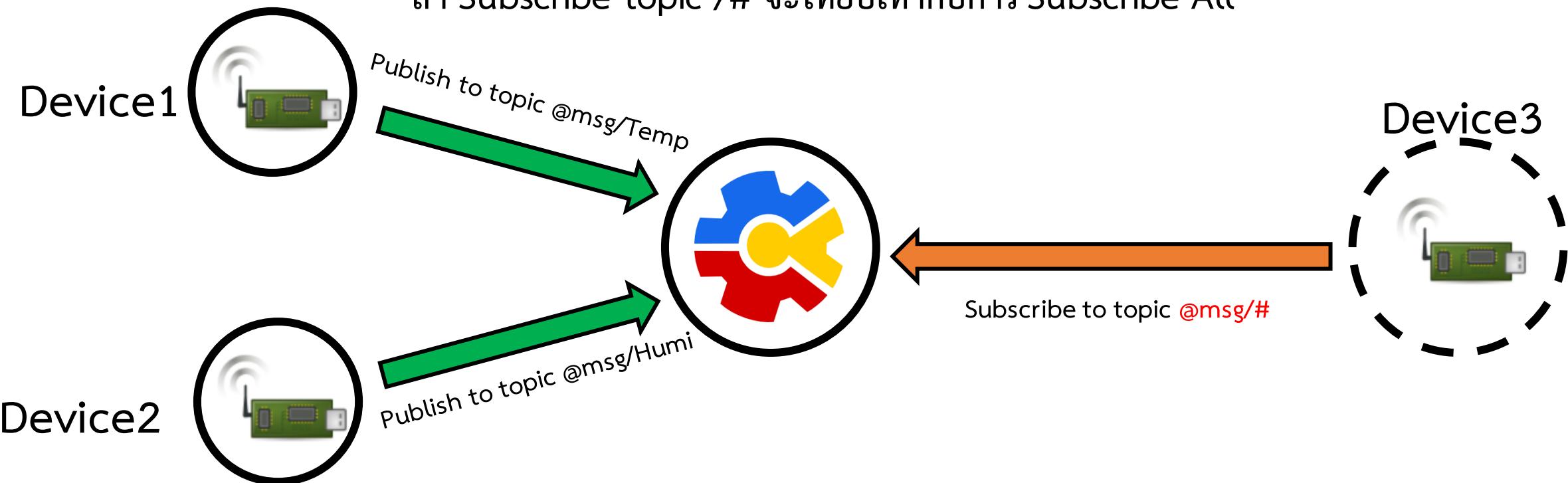
ข้อมูลและ Topic ที่ได้รับมา

Topic แบบ Wildcard

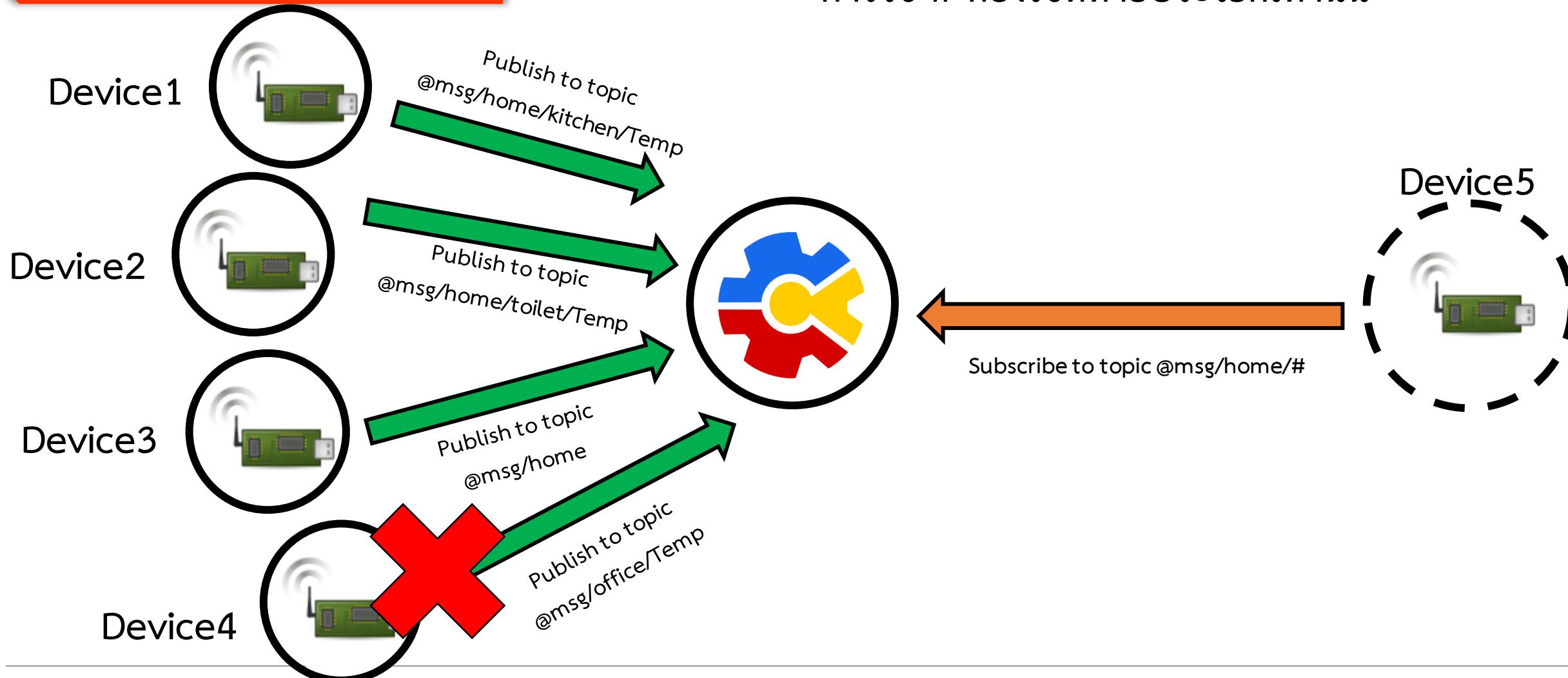


Topic แบบ Wildcard

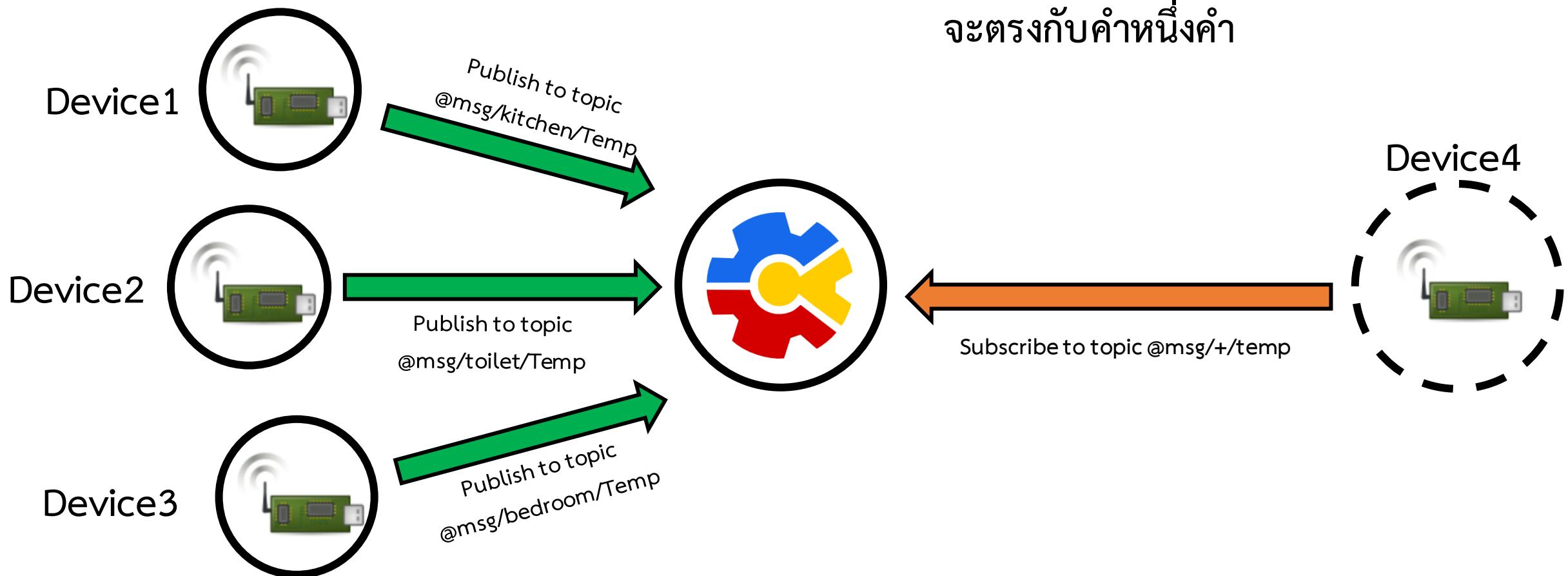
สามารถใช้สัญลักษณ์ # เป็นส่วนหนึ่งใน topic ซึ่งจะตรงกับคำทุกคำ และทุกจำนวน / ของคำ
ถ้า Subscribe topic /# จะเทียบเท่ากับการ Subscribe All



Topic แบบ Wildcard



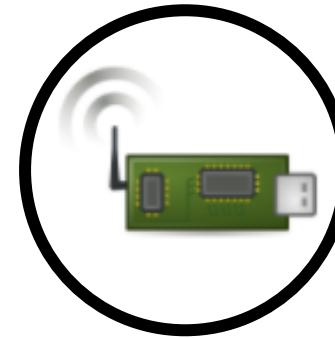
Topic แบบ Wildcard



การสื่อสารระหว่าง DEVICE และ NETPIE2020 ผ่าน MQTT แบ่งเป็น 2 ชนิด

1

Message

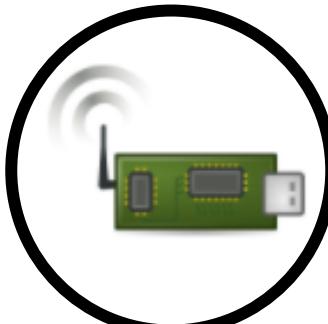


MQTT Protocol



2

Data



MQTT Protocol

MQTT บน NETPIE2020 แบ่งการส่งเป็น 2 แบบคือ

1

Message

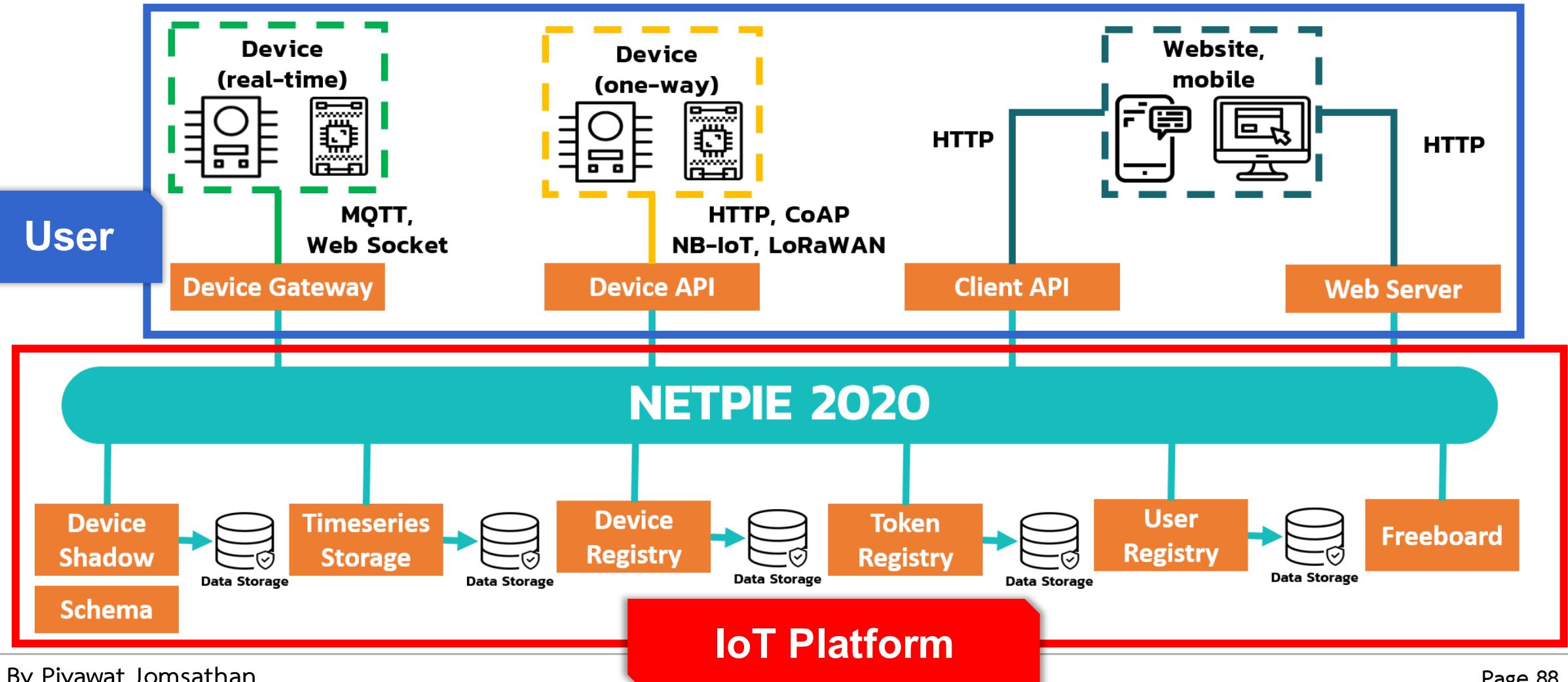
เป็นการส่งข้อความผ่าน MQTT โดยอ้างอิง Topic ชึ่งรูปแบบ topic คือ **@msg/topic**

2

Shadow (Data)

เป็นการส่งข้อความผ่าน MQTT ไปยัง Device Shadow เพื่อบันทึกข้อความล่าสุด มีรูปแบบ topic คือ **@shadow/data/update**
ชึ่งรูปแบบข้อความเป็น JSON ตัวอย่างเช่น **{data:{temp:24}}**

Architecture ຂອງ NETPIE2020



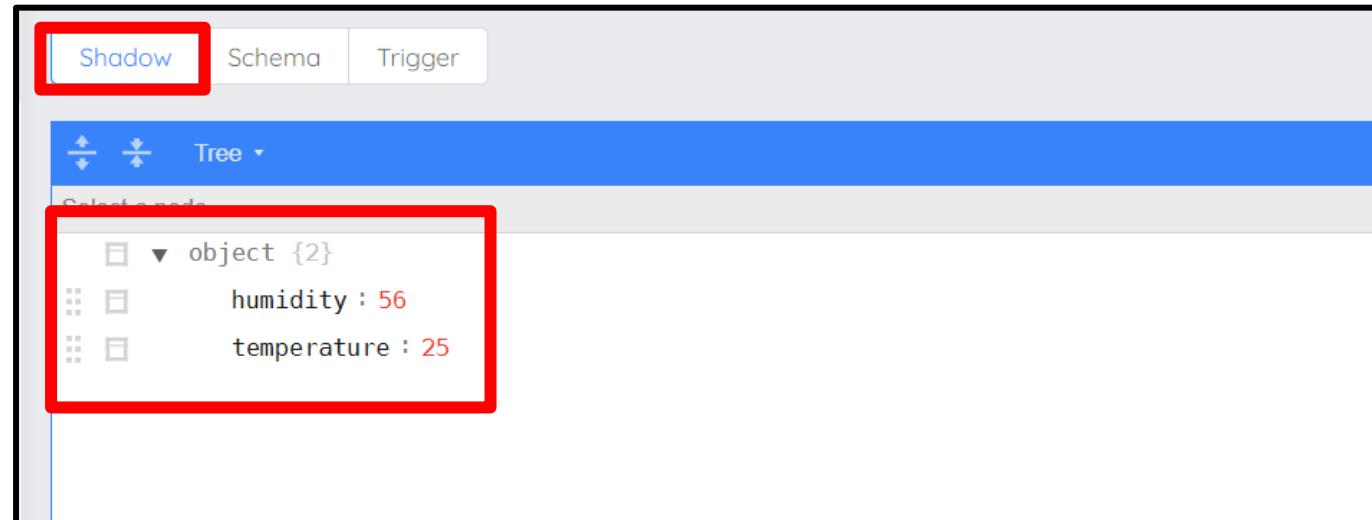
ฐานข้อมูลเสมือน (Device Shadow) บน NETPIE



Device Shadow

Device Shadow

คือ ฐานข้อมูลเสมือนของอุปกรณ์ เป็นฐานข้อมูลเล็ก ๆ ที่มีคู่ออยู่กับอุปกรณ์ (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 และครอบทั้งหมดด้วยเครื่องหมายปีกๆ

ตัวอย่างที่มีข้อมูล 1 อย่างจะเป็นดังนี้
`{"key": "value"}`

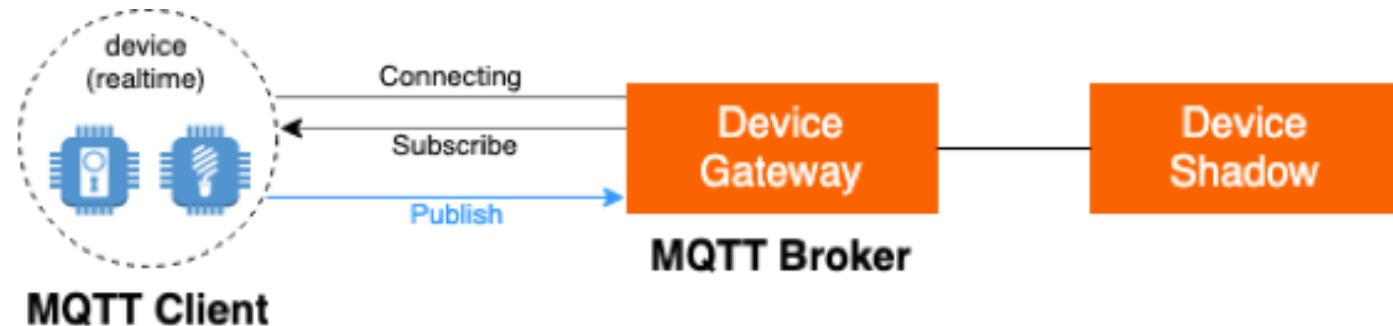
ประเภทข้อมูลที่ JSON เก็บได้มีดังนี้
string, number, object (JSON object)
array, boolean, null

การเก็บข้อมูลประเภท JSON object สามารถวางซ้อนเข้าไปอีกที ตัวอย่างเช่น
`{"name": "Tae", "pet": {"dog": "Shiba", "cat": "Persian"}}`

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

การส่งข้อมูลไปยัง Device Shadow

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

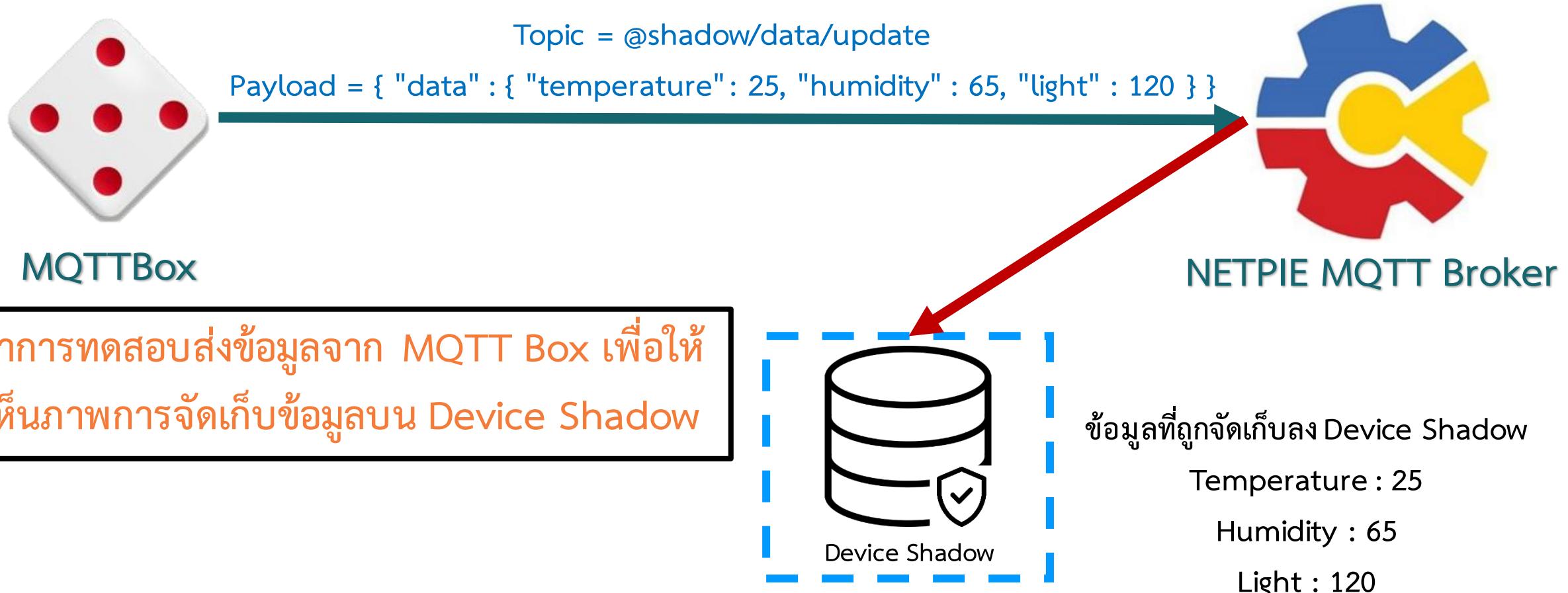


Shadow Topic

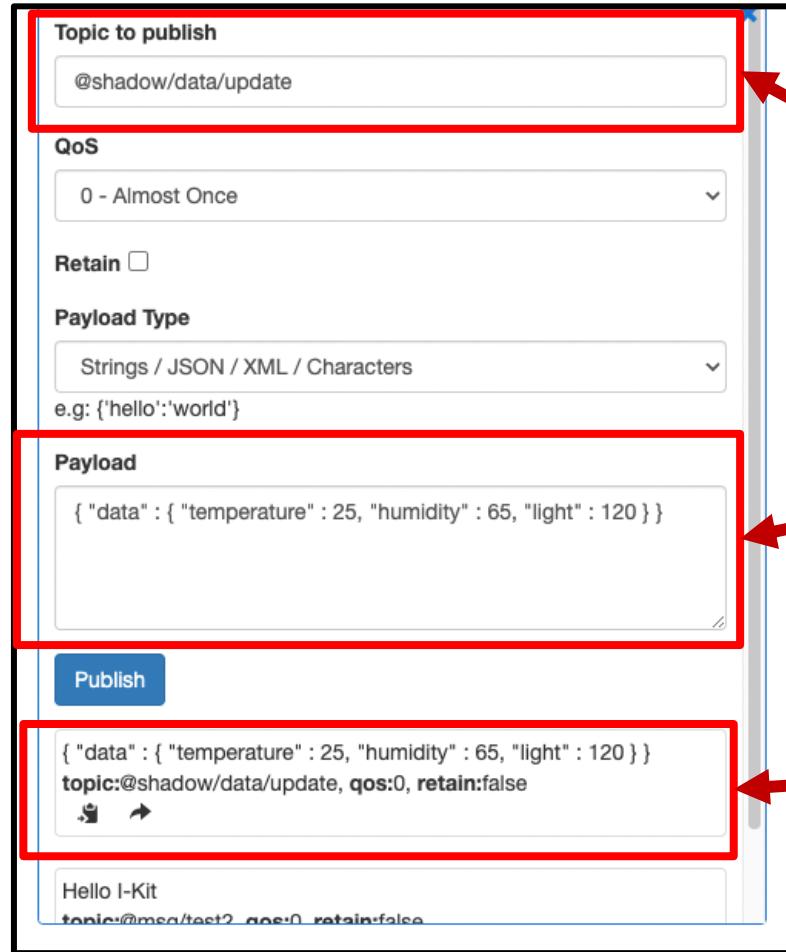
Shadow Topic คือ Topic ที่ใช้สำหรับจัดการ Device Shadow ของตัวเอง Topic ที่เกี่ยวข้องมีดังนี้

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 }}

Example TEST : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย MQTT Box



Example TEST : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย MQTT Box



Example TEST : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย MQTT Box

The screenshot shows the NETPIE platform interface. On the left, the sidebar includes 'PROJECT' (NETPIE_Training), 'WORKSPACE' (Overview, Device, Group, Event Hook, Console), and 'SETTING' (Setting). The main area displays 'Device1' (created date: 2023-01-19) with an 'Edit' button. A 'Detail' panel is open, showing 'Key' information: Client ID (4391e694-8897-4358-be1d-e797c3d0b826), Token (531jXLkuDpMfwzEkGtSqj2HjVi1KX7eQ), Secret (Fgl5kS4olpM!9Nx!68)n-1q-UuJ~s)eZ, Status (Online), and Enable (switch). Below the detail panel is a 'Shadow' tab showing a tree structure with nodes: object (2), temperature : 25, humidity : 65, and light : 120. A red box highlights the 'temperature : 25' node. To the right, a callout box contains the text 'ข้อมูลที่ถูกจัดเก็บลง Device Shadow'.

NETPIE

PROJECT + Add Project

NETPIE_Training

WORKSPACE

- Overview
- Device
- Group
- Event Hook
- Console

SETTING

Setting

Device1

created date: 2023-01-19

Detail

Key

Client ID: 4391e694-8897-4358-be1d-e797c3d0b826 Copy

Token: 531jXLkuDpMfwzEkGtSqj2HjVi1KX7eQ Copy

Secret: Fgl5kS4olpM!9Nx!68)n-1q-UuJ~s)eZ Copy

Status: Online Online

Enable:

Shadow Schema Trigger Feed i

Last Update : 19-01-23 14:52

Select a node...

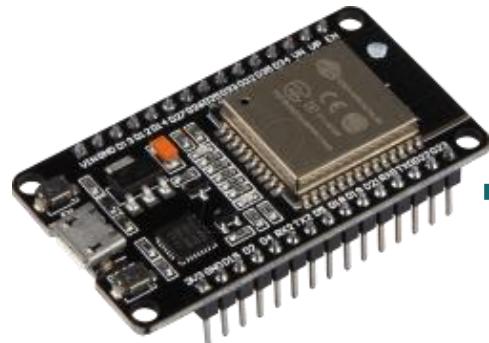
object (2)

- temperature : 25
- humidity : 65
- light : 120

SAVE Cancel

ข้อมูลที่ถูกจัดเก็บลง
Device Shadow

Example 20 : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย DEVKIT



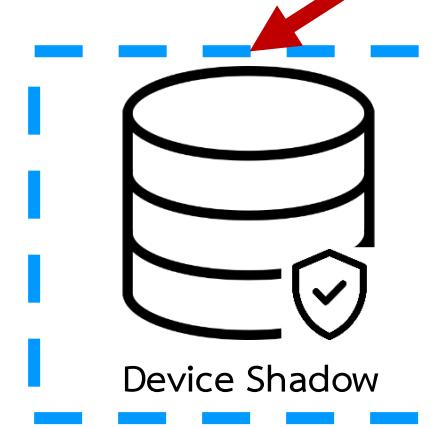
DEVKIT

ทดสอบส่งค่าอุณหภูมิ และความชื้น
ด้วย DEVKIT

Topic = @shadow/data/update
Payload = { "data" : { "temperature": 25, "humidity": 65} }



NETPIE MQTT Broker



ข้อมูลที่ถูกจัดเก็บลง Device Shadow

Temperature : 25

Humidity : 65

Example 20 : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย DEVKIT

Coding ใน Example20.ino แบ่งออกเป็น 3 ส่วน

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 = "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);

char msg[150];

#define DHTPIN 0
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

Example 20 : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย DEVKIT

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

ฟังก์ชันการเชื่อมต่อ MQTT Server

Example 20 : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย DEVKIT

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);  
    dht.begin();  
}
```

ฟังก์ชันการตั้งค่าต่างๆ

การเริ่มต้นใช้งานเซนเซอร์ DHT

Example 20 : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย DEVKIT

3

ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

```
void loop() {  
    if (!client.connected()) {  
        reconnect();  
    }  
    client.loop();  
  
    int humidity = dht.readHumidity();  
    int temperature = dht.readTemperature();  
  
    String data = "{\"data\": {\"temperature\": " + String(temperature) + ",  
    \"humidity\": " + String(humidity) + \"}}";  
    Serial.println(data);  
    data.toCharArray(msg, (data.length() + 1));  
    client.publish("@shadow/data/update", msg);  
    delay(5000);  
}
```

ฟังก์ชันการทำงานหลัก

การประกาศตัวแปรและอ่านค่าเซนเซอร์ DHT

ชุดคำสั่งจัดการข้อมูลก่อนส่งขึ้น Shadow Topic มีขั้นตอนดังนี้

- ทำการจัดรูปข้อมูลให้อยู่ในรูป { "data" : { "temperature" : 25 } } แล้วจัดเก็บลงตัวแปรที่ชื่อว่า data
- ทำการแปลงข้อมูลให้อยู่ในรูป CharArray ไว้ในตัวแปร msg เนื่องจากตอนนี้ตัวแปร data เป็นข้อมูลประเภท obj ซึ่งไม่ถูกจ่อให้ของคำสั่ง client.publish

คำสั่งส่งข้อมูลไปยัง Shadow Topic

Example 20 : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย DEVKIT

การส่งข้อความในรูปแบบ JSON บน Arduino IDE

ข้อความที่ต้องการส่งคือ

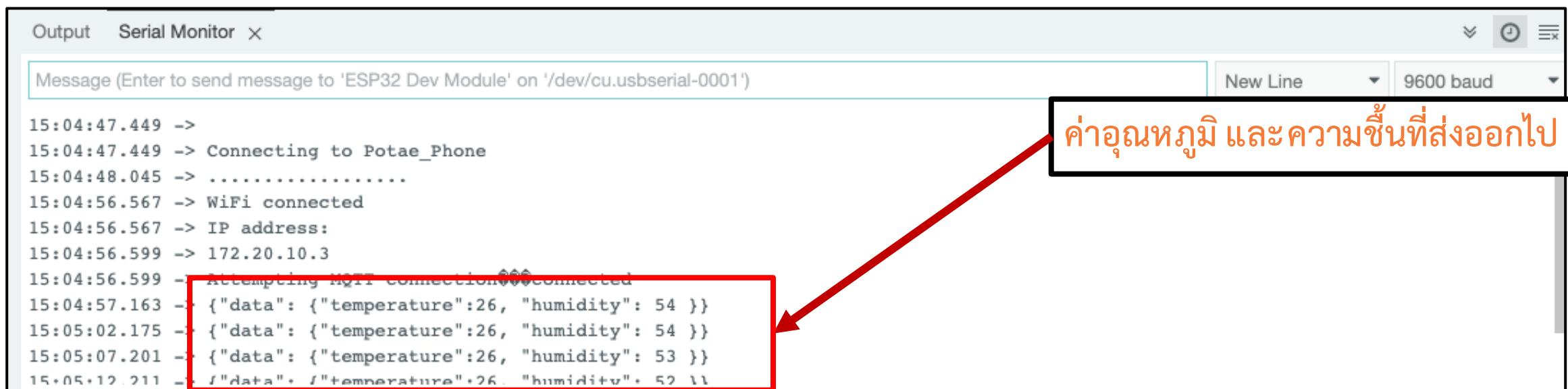
```
{ "data" : { "temperature" : 25 } }
```

แต่บน Arduino IDE มอง " เป็นการประกาศข้อความหรือ String วิธีแก้คือการนำ \ ไว้ข้างหน้า " เพื่อให้
โปรแกรมมอง " เป็นส่วนหนึ่งของข้อความ

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

Example 20 : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย DEVKIT

ตรวจสอบที่ Serial Monitor



Output Serial Monitor ×

Message (Enter to send message to 'ESP32 Dev Module' on '/dev/cu.usbserial-0001')

15:04:47.449 ->

15:04:47.449 -> Connecting to Potaе_Phone

15:04:48.045 ->

15:04:56.567 -> WiFi connected

15:04:56.567 -> IP address:

15:04:56.599 -> 172.20.10.3

15:04:56.599 -> Attempting MQTT connection

15:04:57.163 -> {"data": {"temperature": 26, "humidity": 54 }}

15:05:02.175 -> {"data": {"temperature": 26, "humidity": 54 }}

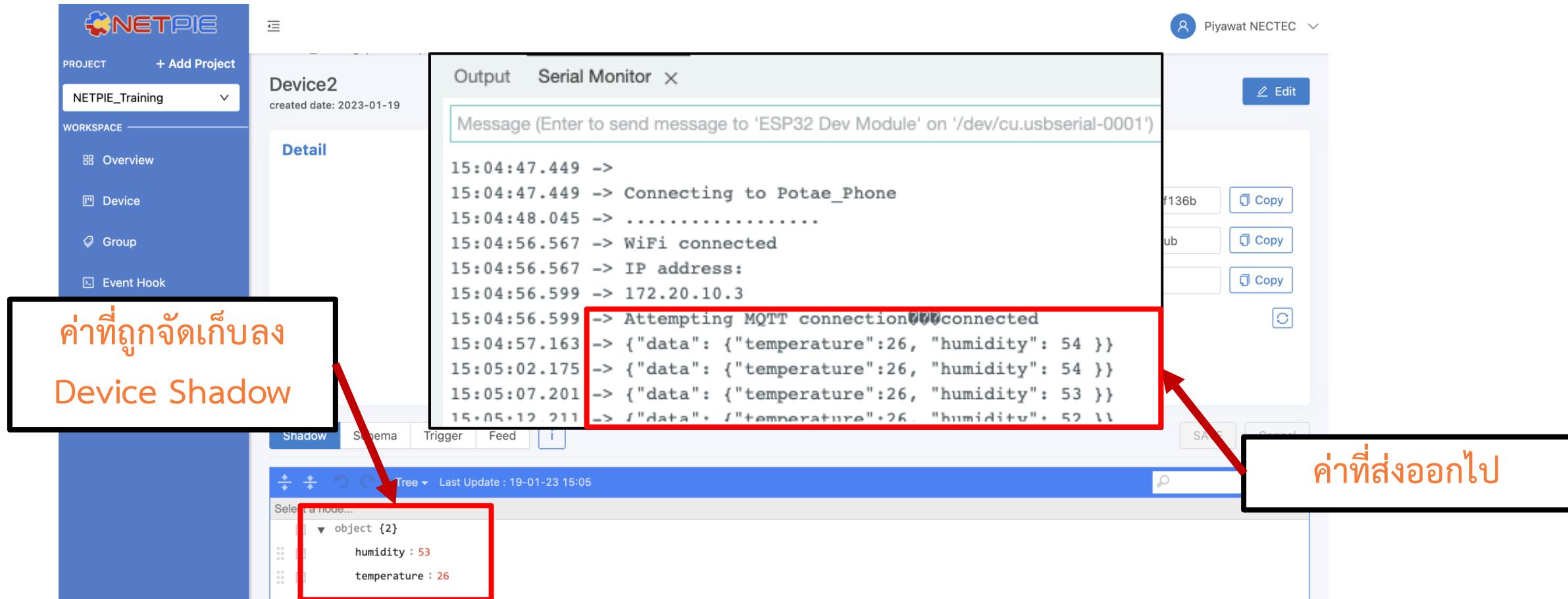
15:05:07.201 -> {"data": {"temperature": 26, "humidity": 53 }}

15:05:12.211 -> {"data": {"temperature": 26, "humidity": 52 }}

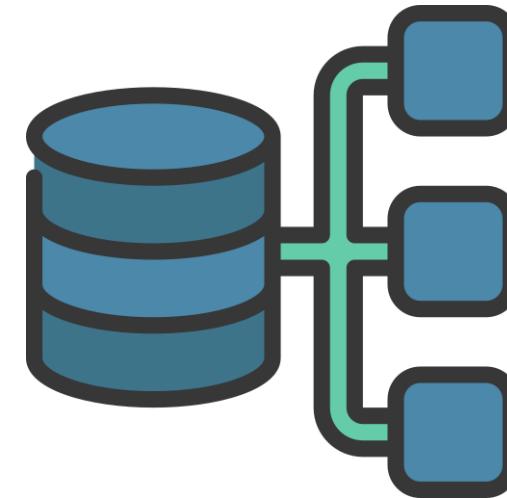
New Line 9600 baud

ค่าอุณหภูมิ และความชื้นที่ส่งออกไป

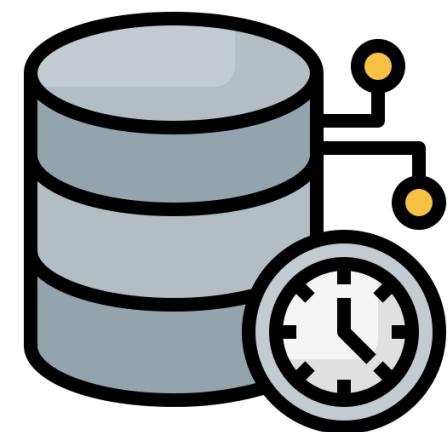
Example 20 : ทดสอบส่งข้อมูลไปยัง Device Shadow ด้วย DEVKIT



ແຜັດຂໍ້ມູນ (Device Schema) ແລະ ຮູນຂໍ້ມູນເວລາ (Device Feed) ບນ NETPIE



Device Schema



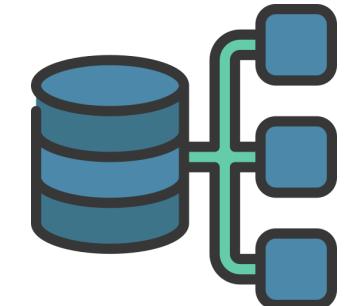
Device Feed

ແພັດຂໍ້ມູນ (Device Schema)

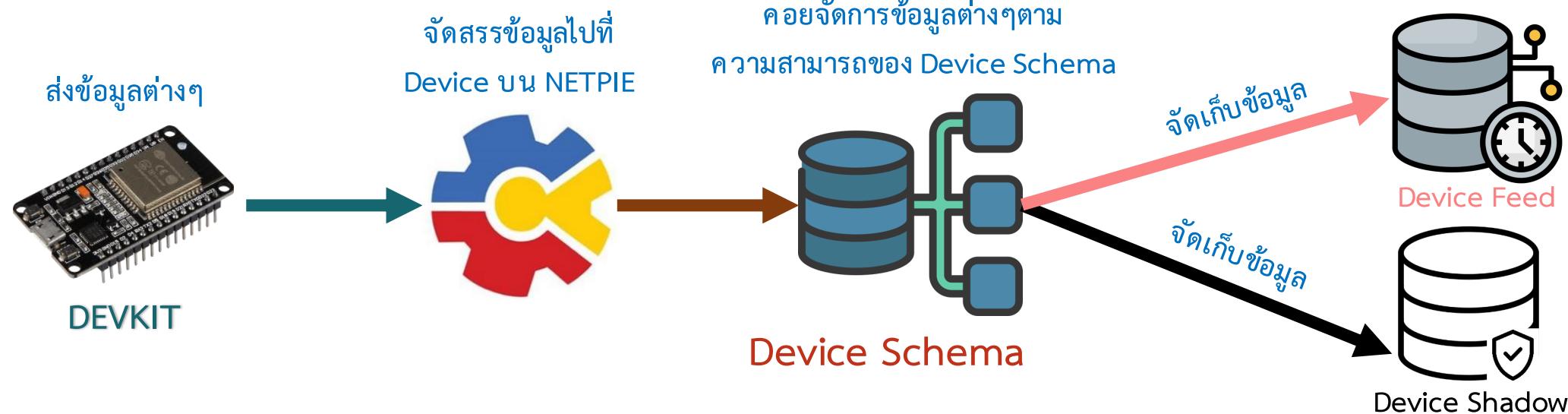
ສໍາຮັບອຸປະກນົມທີ່ຕ້ອງມີການຈັດການຂໍ້ມູນ ແນະນຳໃຫ້ສ່ວນ Device Schema ຂອງຂໍ້ມູນເຕີຍມໄວ້

Device Schema ຄື່ອ ແພັດຂໍ້ມູນທີ່ກໍາທັນໄວ້ເພື່ອໃຊ້ກໍາກັບ Device Shadow ໂດຍ Device Schema ເສີ່ອນເປັນ Device Template ທຳໃໝ່ Server ສາມາດ

- ການຕຽບສອບໜິດຂໍ້ມູນ (Data Validation)
- ການແປລັງຂໍ້ມູນ (Data Transformation) ເຊັ່ນ ເປີ່ຍໜ່ວຍຂອງຂໍ້ມູນ ເປັນຕົ້ນ
- ການເກີບຂໍ້ມູລັງໃນ Timeseries Database (Device Feed)



Device Schema



ยกตัวอย่างการทำงานของ Device Schema



DEVKIT

ข้อมูลที่ถูกส่งจากอุปกรณ์

Topic = @shadow/data/update

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

NETPIE MQTT Broker



หาก Device Schema ทำการตั้งค่าไว้ว่า

1. ข้อมูลที่ต้องการจัดเก็บคือ Temperature (number), Humidity (number) และ Place (string)
2. มีการแปลงหน่วยข้อมูลของ Temperature จาก องศาเซลเซียส ($^{\circ}\text{C}$) เป็น องศาฟาร์นไฮต์ ($^{\circ}\text{F}$)
3. กำหนดให้ทุกข้อมูลจัดเก็บเป็นเวลา 7 วัน (ข้อมูลทั้งหมดที่ถูกจัดเก็บใน Device Feed จะมีอายุ 7 วัน)

Device Schema

ข้อมูลที่จะถูกจัดเก็บลง Device Shadow

temperature : 25

humidity : 65

light : 120

place : "home "

ข้อมูลใน Device Feed

temperature : 77

humidity : 65

light : 120

place : "home "

Temperature และ Light ถูกจัดการโดย
Device Schema ตามค่าที่กำหนด

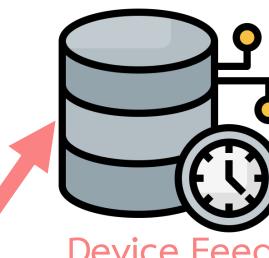
ข้อมูลใน Device Shadow

temperature : 77

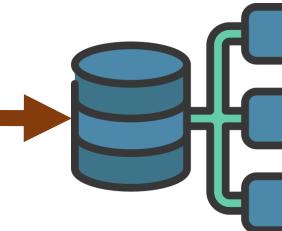
humidity : 65

light : 120

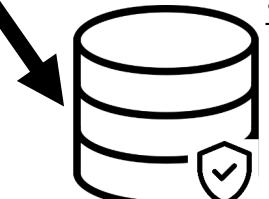
place : "home "



Device Feed



Device Schema



Device Shadow

ยกตัวอย่างการทำงานของ Device Schema



DEVKIT

ข้อมูลที่ถูกส่งจากอุปกรณ์

Topic = @shadow/data/update

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

ข้อมูลของ place ถูกเปลี่ยนประเภท
ข้อมูลเป็น number

NETPIE MQTT Broker



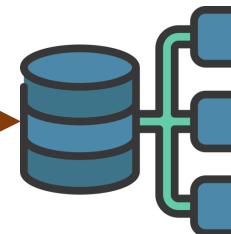
หาก Device Schema ทำการตั้งค่าไว้ว่า

1. ข้อมูลที่ต้องการจัดเก็บคือ Temperature (number), Humidity (number) และ Place (string)
2. มีการแปลงหน่วยข้อมูลของ Temperature จาก องศาเซลเซียส ($^{\circ}\text{C}$) เป็น องศาฟาร์เรนไฮต์ ($^{\circ}\text{F}$)
3. กำหนดให้ทุกข้อมูลจัดเก็บเป็นเวลา 7 วัน (ข้อมูลทั้งหมดที่ถูกจัดเก็บใน Device Feed จะมีอายุ 7 วัน)

ข้อมูลที่จะถูกจัดเก็บลง Device Shadow

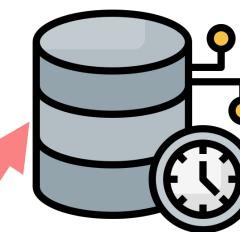
temperature : 25
humidity : 65
light : 120
place : "home "

Device Schema



ข้อมูลใน Device Feed

temperature : 77
humidity : 65
light : 120
place : "home "

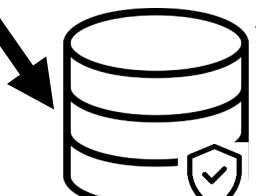


Device Feed

Place จะไม่ถูกจัดเก็บลง
Device Shadow และ Device Feed

ข้อมูลใน Device Shadow

temperature : 77
humidity : 65
light : 120
place : "home "

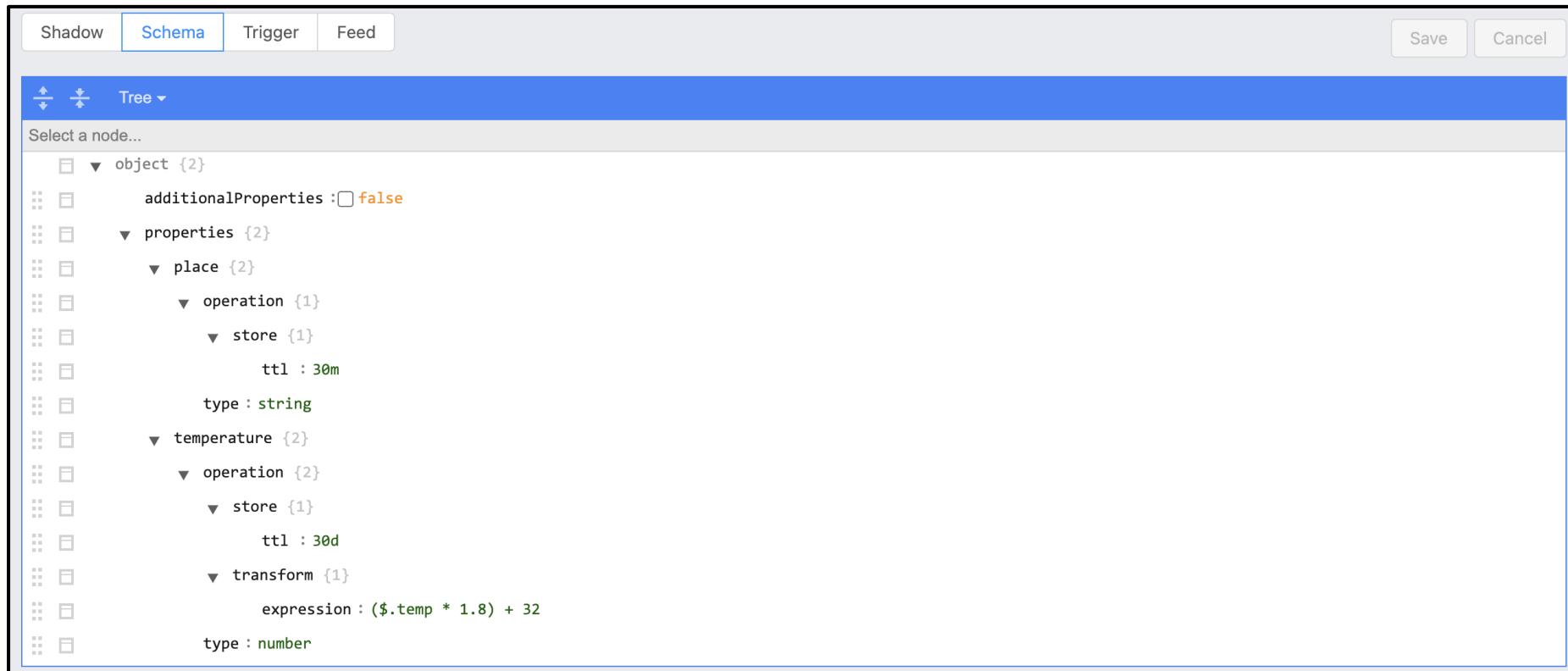


Device Shadow

การประกาศ Device Schema

การประกาศ Device Schema จะอยู่ในรูปแบบของ JSON เสมอ

ตัวอย่างการประกาศ Device Schema



การประกاث Device Schema

Device Schema จะประกอบด้วย 2 ส่วนหลักคือ

```
Code ▾  
1 {  
2   "additionalProperties": false,  
3   "properties": {  
4     "place": {  
5       "operation": {  
6         "store": {  
7           "ttl": "3d"  
8         }  
9       },  
10      "type": "string"  
11    },  
12    "temperature": {  
13      "operation": {  
14        "store": {  
15          "ttl": "30d"  
16        },  
17        "transform": {  
18          "expression": "($.temperature * 1.8) + 32"  
19        }  
20      },  
21      "type": "number"  
22    }  
Ln: 1 Col: 1
```

1. additionalProperties

2. Properties

การประกาศ Device Schema

```
1  {
2   "additionalProperties": false, ←
3   "properties": {
4     "place": {
5       "operation": {
6         "store": {
7           "ttl": "3d"
8         }
9       },
10      "type": "string"
11    },
12    "temperature": {
13      "operation": {
14        "store": {
15          "ttl": "30d"
16        },
17        "transform": {
18          "expression": "($.temperature * 1.8) + 32"
19        }
20      },
21      "type": "number"
22    }
23  }
```

Ln: 1 Col: 1

1. additionalProperties

คือ สถานะการอนุญาตให้บันทึกข้อมูลง Device Shadow และ Device Feed
ในการณ์ที่ข้อมูลไม่ตรงตามที่กำหนด 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

การประกاث Device Schema

```
1  {
2     "additionalProperties": false,
3     "properties": {
4         "place": {
5             "operation": {
6                 "store": {
7                     "ttl": "3d"
8                 }
9             },
10            "type": "string"
11        },
12        "temperature": {
13            "operation": {
14                "store": {
15                    "ttl": "30d"
16                },
17                "transform": {
18                    "expression": "$.temperature * 1.8 + 32"
19                }
20            },
21            "type": "number"
22        }
23    }
24 }
```

Ln: 1 Col: 1

2. Properties

กำหนดชื่อฟิลด์ (ตัวอย่างคือ place และ temperature) และกำหนดคุณสมบัติของแต่ละฟิลด์ซึ่งแบ่งเป็น 2 ส่วนคือ

การประยุกต์ Device Schema

```

1 "additionalProperties": false,
2 "properties": {
3     "place": {
4         "operation": {
5             "store": {
6                 "ttl": "3d"
7             }
8         },
9         "type": "string"
10    },
11    "temperature": {
12        "operation": {
13            "store": {
14                "ttl": "30d"
15            },
16            "transform": {
17                "expression": "($.temperature * 1.8) + 32"
18            }
19        },
20        "type": "number"
21    }
22 }

```

Ln: 1 Col: 1

2. Properties

กำหนดชื่อฟิลด์ (ตัวอย่างคือ place และ temperature) และกำหนดคุณสมบัติของแต่ละฟิลด์ซึ่งแบ่งเป็น 2 ส่วนคือ

- 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

การประยุกต์ Device Schema

```

1  {
2   "additionalProperties": false,
3   "properties": {
4     "place": {
5       "operation": {
6         "store": {
7           "ttl": "3d"
8         }
9       },
10      "type": "string"
11    },
12    "temperature": {
13      "operation": {
14        "store": {
15          "ttl": "30d"
16        },
17        "transform": {
18          "expression": "$.temperature * 1.8 + 32"
19        }
20      },
21      "type": "number"
22    }
23  }

```

Ln: 1 Col: 1

2. Properties

กำหนดชื่อฟิลด์ (ตัวอย่างคือ place และ temperature) และกำหนดคุณสมบัติของแต่ละฟิลด์ซึ่งแบ่งเป็น 2 ส่วนคือ

1. Operation สำหรับตั้งค่าการจัดการข้อมูลในฟิลด์นั้นๆ ประกอบไปด้วย
store สำหรับตั้งค่าการเก็บข้อมูลลง Timeseries Database
ttl คือ ระยะเวลาของการเก็บข้อมูลใน Timeseries Database ซึ่งแต่ละข้อมูลมีอายุการเก็บครบตามกำหนดจะถูกลบทิ้งอัตโนมัติ
ถ้าต้องการจัดการเก็บข้อมูลระบบจำเป็นต้องกำหนดค่านี้ มีหน่วยเป็น ms(มิลลิวินาที), s(วินาที), m(นาที), h(ชั่วโมง), d(วัน), y(ปี)

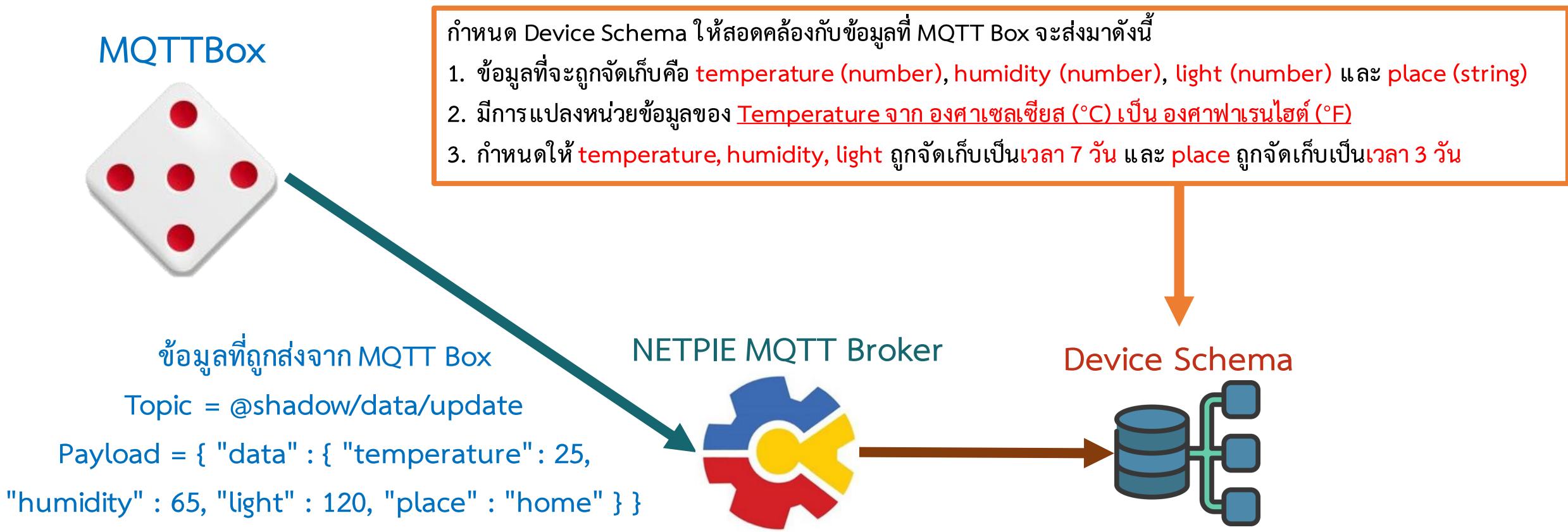
Transform การแปลงข้อมูลก่อนจัดเก็บ

expression คือ สูตรหรือวิธีการแปลงข้อมูลก่อนจัดเก็บ

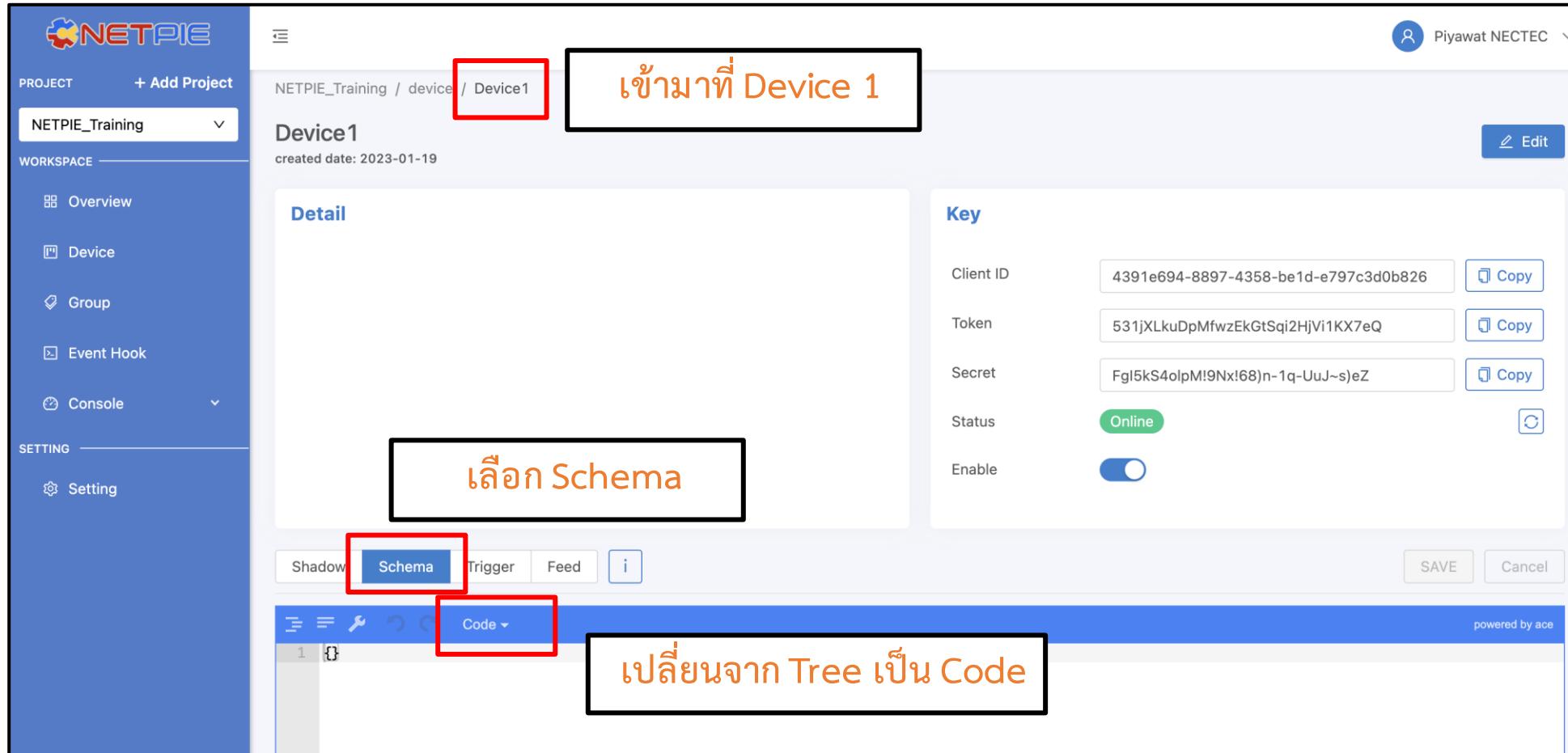
ตัวอย่าง แปลงหน่วยอุณหภูมิจากเซลเซียสเป็นฟาเรนไฮต์ = (\$.temperature*1.8) + 32

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

Example TEST : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device1 (MQTT Box)



Example TEST : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device1 (MQTT Box)



Example TEST : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device1 (MQTT Box)

```
{  
  "additionalProperties": false,  
  "properties": {  
    "place": {  
      "operation": {  
        "store": {  
          "ttl": "3d"  
        }  
      },  
      "type": "string"  
    },  
  },  
}
```

การประกาศฟิลด์ของ place

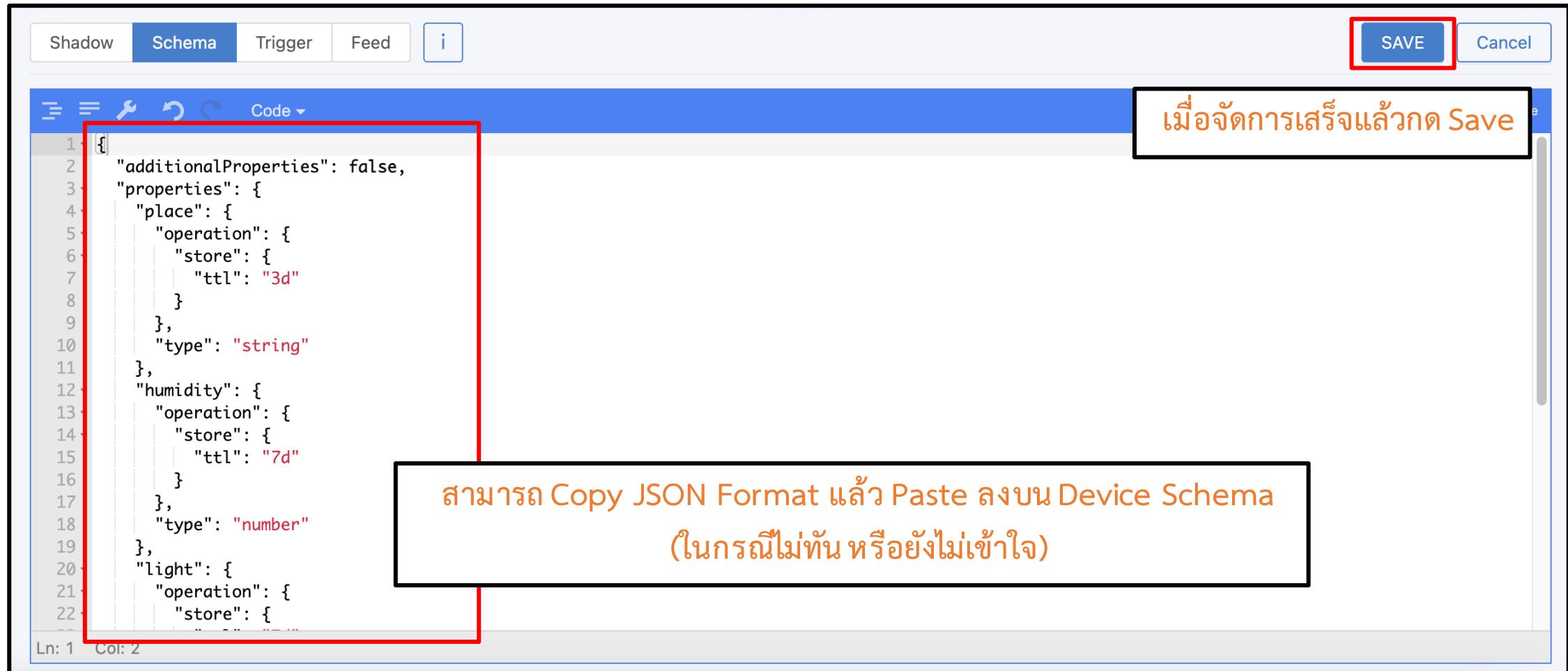
```
"humidity": {  
  "operation": {  
    "store": {  
      "ttl": "7d"  
    },  
    "type": "number"  
  },  
  "light": {  
    "operation": {  
      "store": {  
        "ttl": "7d"  
      },  
      "type": "number"  
    },  
  },  
}
```

การประกาศฟิลด์ของ humidity และ light

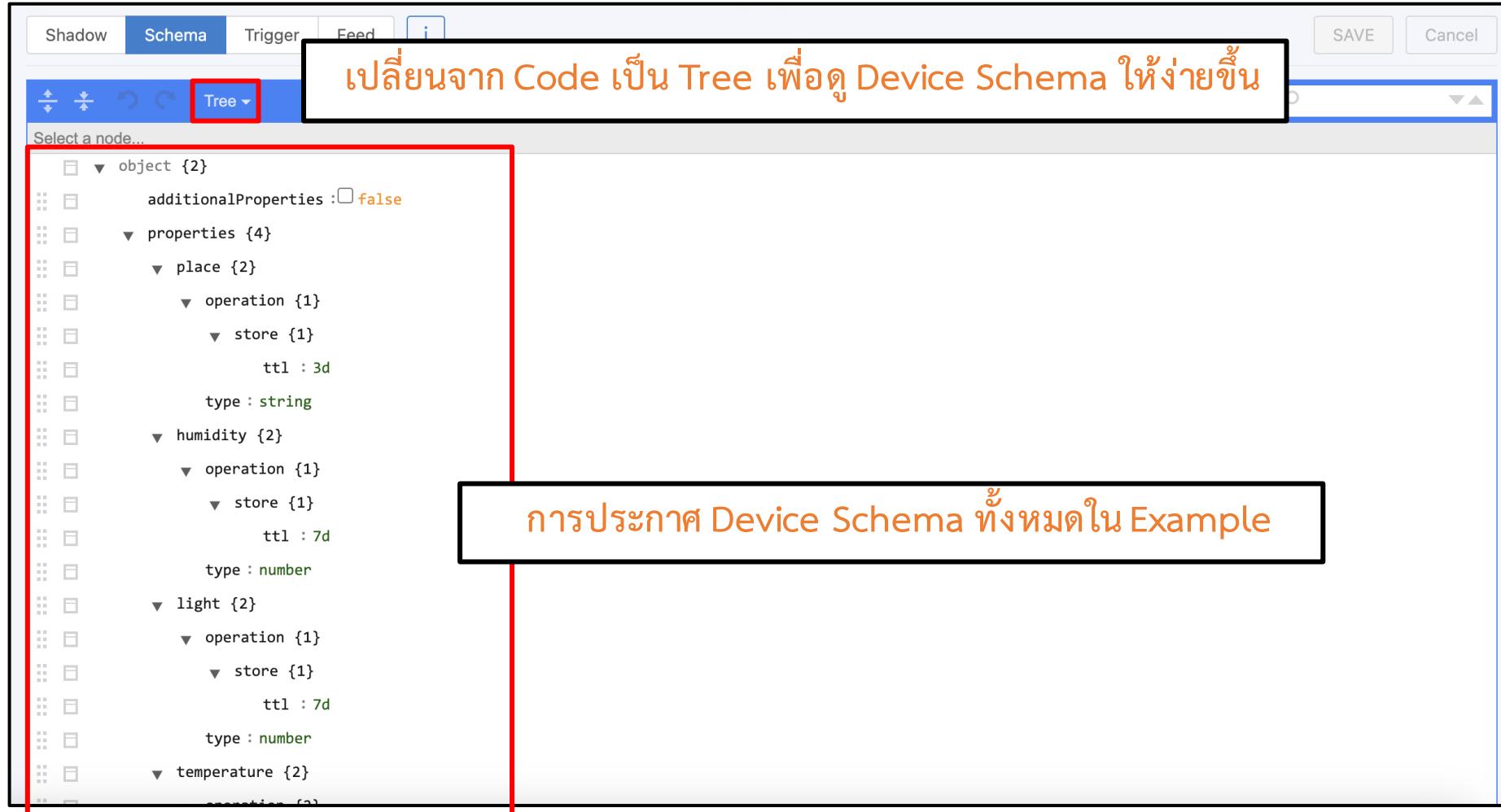
```
"temperature": {  
  "operation": {  
    "store": {  
      "ttl": "30d"  
    },  
    "transform": {  
      "expression": "($.temperature * 1.8) + 32"  
    }  
  },  
  "type": "number"  
}
```

การประกาศฟิลด์ของ temperature

Example TEST : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device1 (MQTT Box)



Example TEST : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device1 (MQTT Box)



Example TEST : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device1 (MQTT Box)

The screenshot shows the NetPie MQTT publisher interface for Device1. The payload field contains the following JSON:

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

Annotations in orange text provide instructions:

- มาที่ MQTT Box ที่เชื่อมต่อ Device1 ไว้
- ทำการทดสอบส่งข้อมูลตามตัวอย่างเพื่อตัด
ข้อมูลที่ถูกจัดเก็บลง Device Shadow
- Payload สามารถ Copy ได้จากตรงนี้
- { "data" : { "temperature" : 25, "humidity" : 65, "light" : 120 , "place" : "Piyawat Home" } }

Example TEST : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device1 (MQTT Box)

The screenshot shows the NETPIE Platform interface. On the left, the sidebar includes 'PROJECT' (NETPIE_Training selected), 'WORKSPACE' (Overview, Device, Group, Event Hook, Console), and 'SETTING' (Setting). The main area displays 'Device 1' (created date: 2023-01-19) with a 'Detail' tab and a 'Key' section containing Client ID, Token, Secret, Status (Online), and Enable toggle. Below this is a 'Shadow' tab in the Device Shadow editor, which shows a tree structure with nodes: object (4) containing temperature (77), humidity (65), light (120), and place (Piyawat Home). A red box highlights the 'object' node, and an orange callout box points to it with the text 'ข้อมูลที่ถูกจัดเก็บลง Device Shadow'. The editor also has tabs for Schema, Trigger, Feed, and an info icon.

Example TEST : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device1 (MQTT Box)

The screenshot shows the NETPIE IDE interface. On the left, the 'PROJECT' sidebar shows 'NETPIE_Training' selected. The 'WORKSPACE' section has 'Overview' selected. In the center, there's a preview area and a 'Trigger' / 'Feed' switch. Below it, a 'Last Update : 19' message is displayed. A red box highlights the 'Device Shadow' state, which shows:

ข้อมูลบน Device Shadow จะไม่เปลี่ยนแปลง
เนื่องจากข้อมูลถูก Device Schema ตรวจสอบก่อนจัดเก็บ

The 'Topic to publish' field contains '@shadow/data/update'. The 'QoS' dropdown is set to '0 - Almost Once'. The 'Payload' dropdown is set to 'Strings' with the example 'e.g: {"hello": "world"}'. The 'Payload' text input field contains the JSON object:

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

A red box highlights the 'Publish' button at the bottom of the configuration panel.

On the right, a large orange box contains the following text:

ทดสอบส่งประเภทข้อมูลผิดขึ้นไปนั่นคือ
temperature เป็น string และ place เป็น number

Example 21 : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device2 (DEVKIT)



DEVKIT

- กำหนด Device Schema ให้สอดคล้องกับข้อมูลที่ DEVKIT จะส่งมาดังนี้
1. ข้อมูลที่จะถูกจัดเก็บคือ temperature (number), humidity (number) และ light (number)
 2. มีการแปลงหน่วยข้อมูลของ Temperature จาก องศาเซลเซียส ($^{\circ}\text{C}$) เป็น องศา华เรนไฮต์ ($^{\circ}\text{F}$)
 3. กำหนดให้ **temperature, humidity, light** ถูกจัดเก็บเป็นเวลา 7 วัน

ข้อมูลที่ถูกส่งจาก MQTT Box

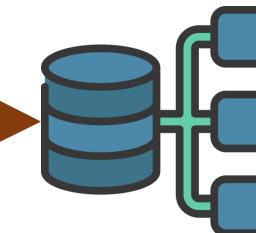
Topic = @shadow/data/update

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

NETPIE MQTT Broker

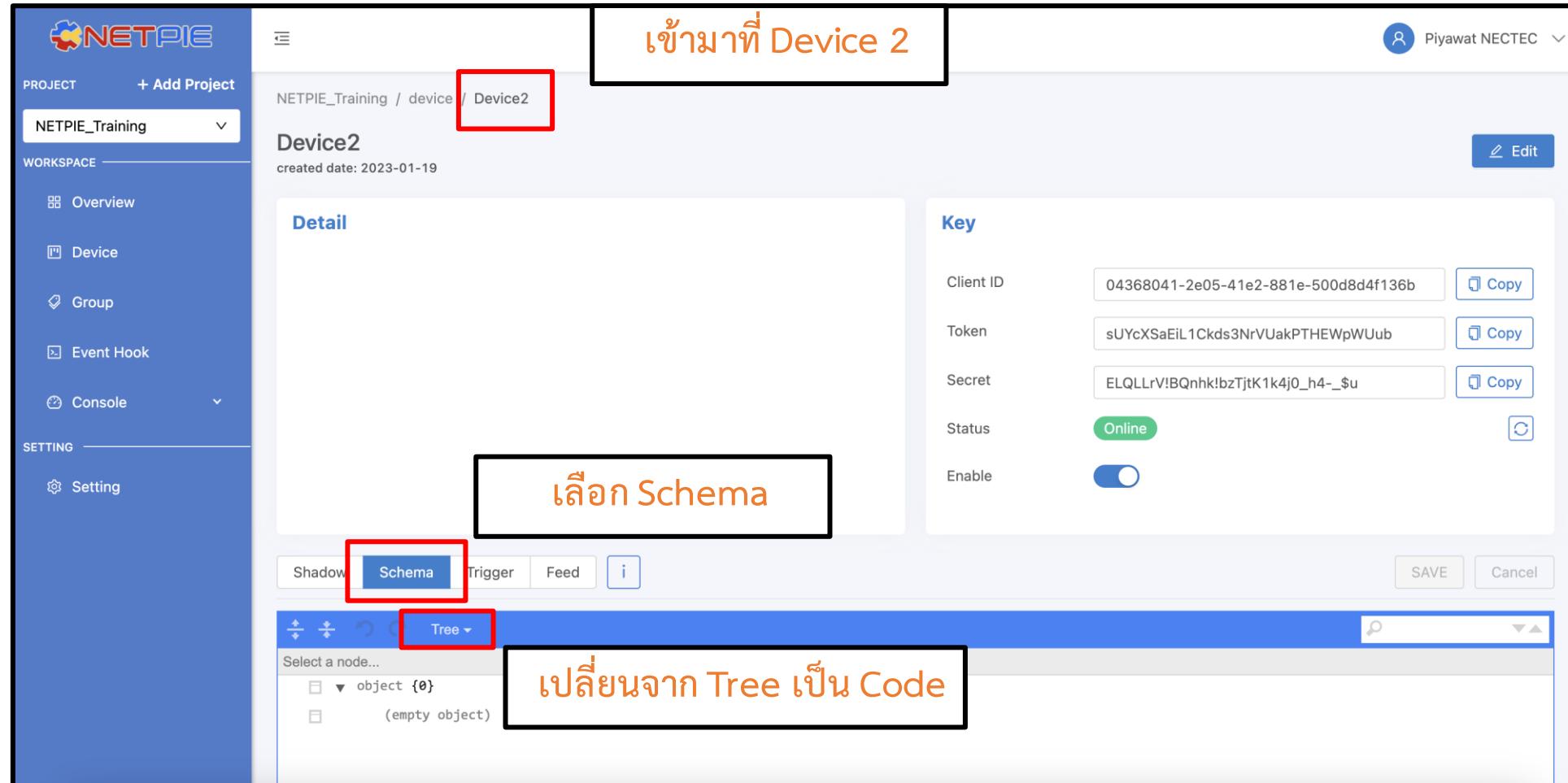


Device Schema



เราสามารถใช้โค้ด example20 ในการส่งข้อมูลต่างๆ ของ DEVKIT ไปยัง Device บน NETPIE

Example 21 : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device2 (DEVKIT)



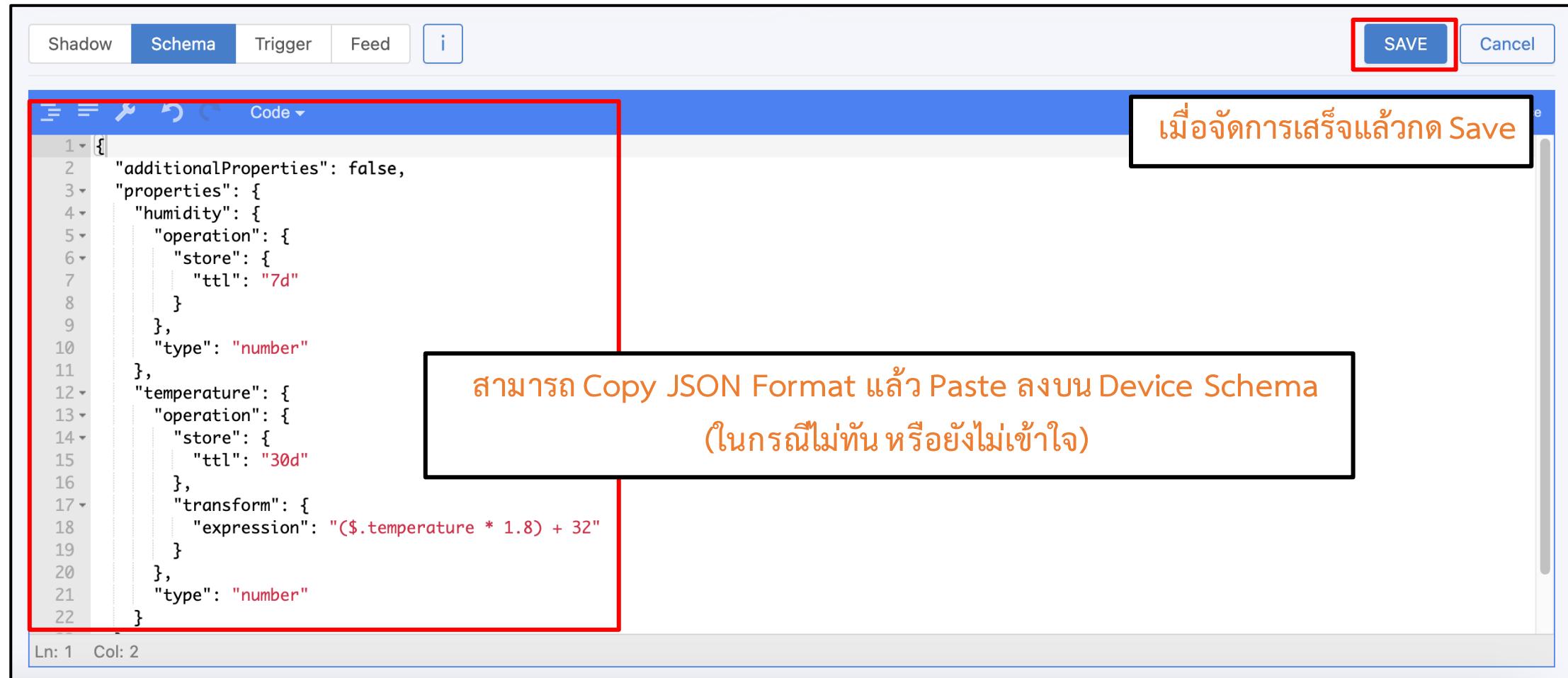
Example 21 : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device2 (DEVKIT)

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

การประมวลผลของ humidity

การประมวลผลของ temperature

Example 21 : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device2 (DEVKIT)



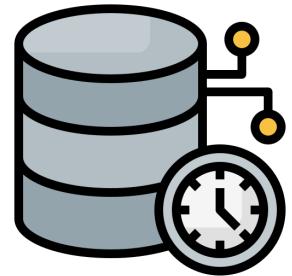
Example 21 : ทดสอบการตั้งค่า Schema และส่งข้อมูลบน Device2 (DEVKIT)

The screenshot shows the NETPIE Platform interface. On the left, there's a sidebar with 'PROJECT' and 'WORKSPACE' sections. Under 'WORKSPACE', 'Device' is selected. In the main area, 'Device2' is shown with its creation date '2023-01-19'. A 'Detail' panel is open. To the right, there's a 'Key' section with Client ID, Token, Secret, Status (Online), and Enable toggle. Below these is a schema editor titled 'Shadow'. It shows a tree structure with an object containing 'humidity: 54' and 'temperature: 77'. A red box highlights this schema. An orange callout box points to it with the text 'ข้อมูลที่ถูกจัดเก็บลง Device Shadow และถูกแปลงหน่วย' (Data stored in Device Shadow and converted to units).

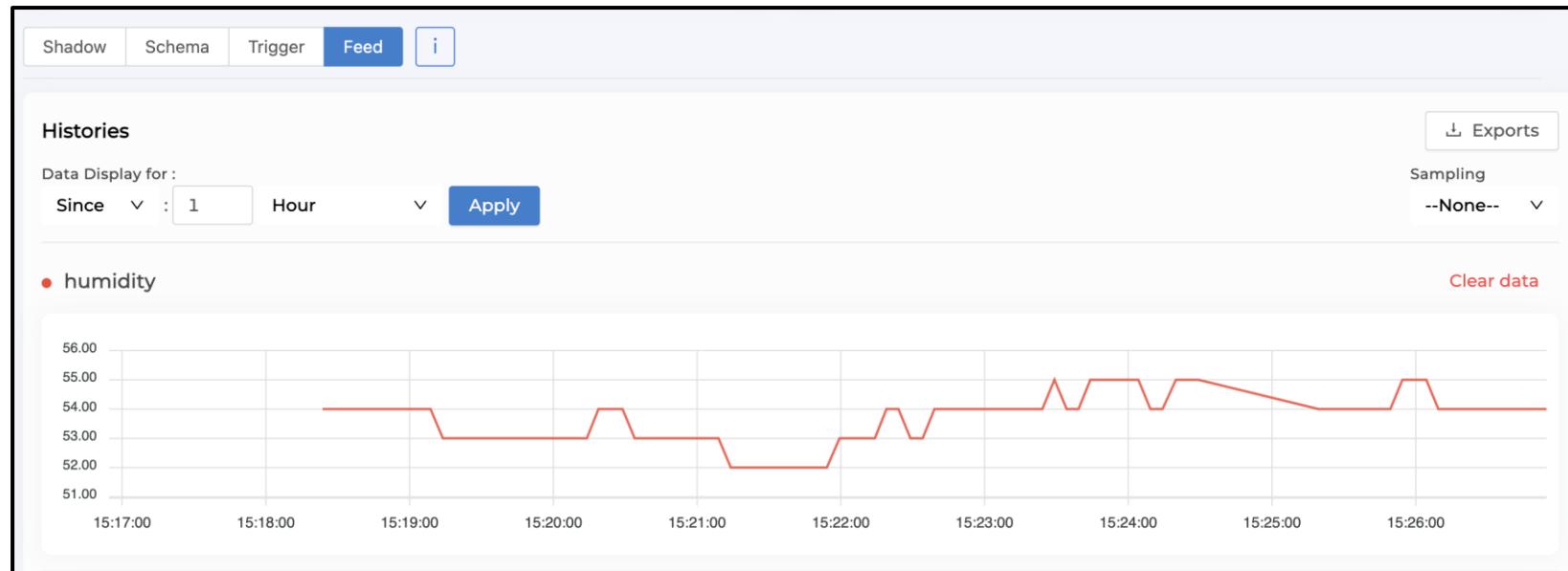
ฐานข้อมูลเวลา (Device Feed)

ข้อมูลบน Device Feed จะถูกจัดเก็บให้อัตโนมัติหลังจากตั้งค่าบน Device Schema สำเร็จ

Device Feed คือ ฐานข้อมูลเวลาบน NETPIE โดยจัดเก็บในรูปแบบของ Timeseries Data ซึ่งมีແນບเครื่องมือในการจัดการและดูข้อมูลเบื้องต้นของแต่ละ Device อีกทั้งจะแสดงในรูปแบบของ กราฟเส้น แยกตามพิลต์ (หรือ ก็คือ Properties ที่กำหนดอยู่ใน Device Schema) และยังสามารถ ดาวน์โหลดข้อมูลออกมาเป็นไฟล์ .csv

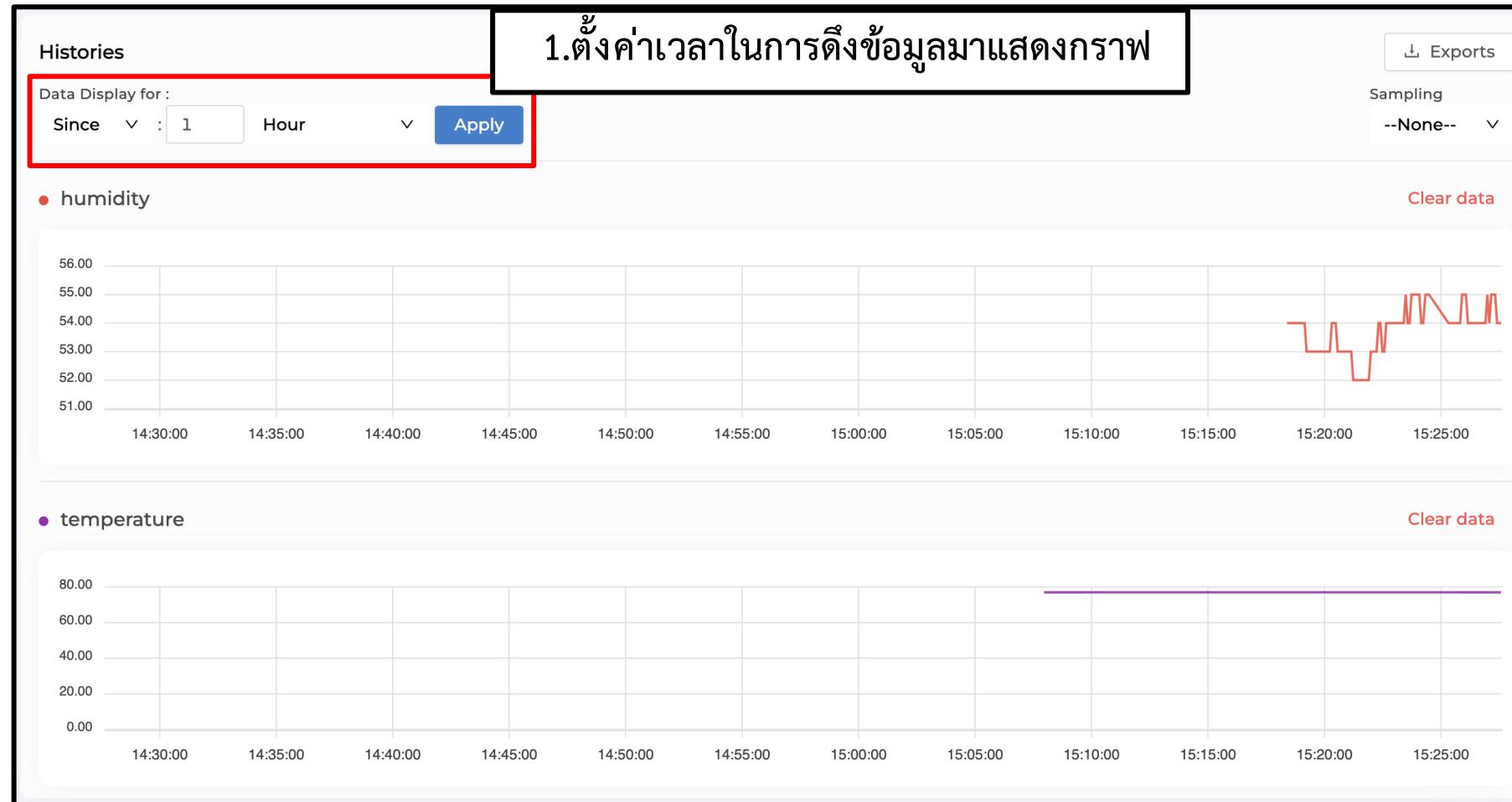


Device Feed



การใช้งาน Device Feed

เครื่องมือบน Device Feed มีทั้งหมด 4 เครื่องมือ



การใช้งาน Device Feed

1. การตั้งค่าเวลาในการดึงข้อมูลมาแสดงกราฟ มีอยู่ 2 แบบ

Data Display for :

Since : 1 Hour Apply

Minute

Hour

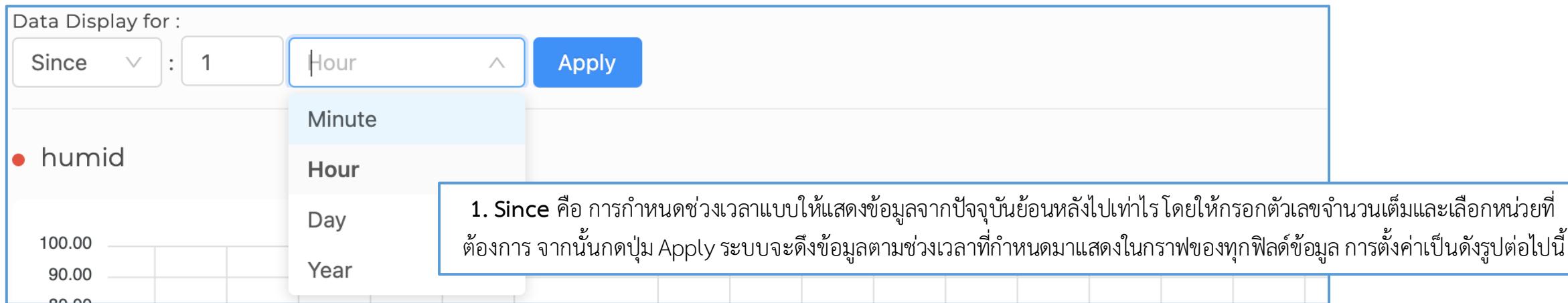
Day

Year

● humid

100.00
90.00
80.00

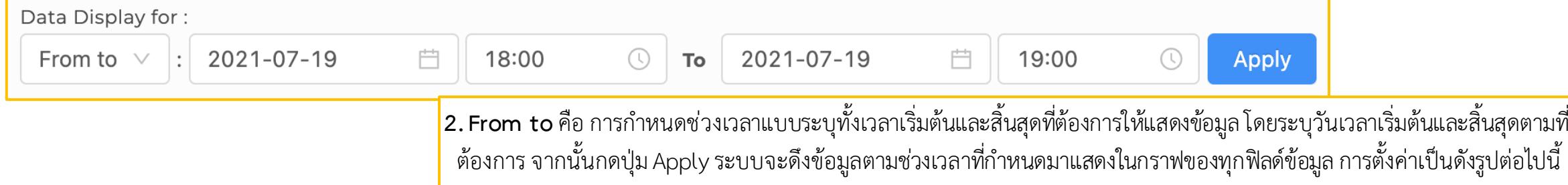
1. Since คือ การกำหนดช่วงเวลาแบบให้แสดงข้อมูลจากปัจจุบันย้อนหลังไปเท่าไร โดยให้กรอกตัวเลขจำนวนเต็มและเลือกหน่วยที่ต้องการ จากนั้นกดปุ่ม Apply ระบบจะดึงข้อมูลตามช่วงเวลาที่กำหนดมาแสดงในกราฟของทุกฟิลด์ข้อมูล การตั้งค่าเป็นดังรูปต่อไปนี้



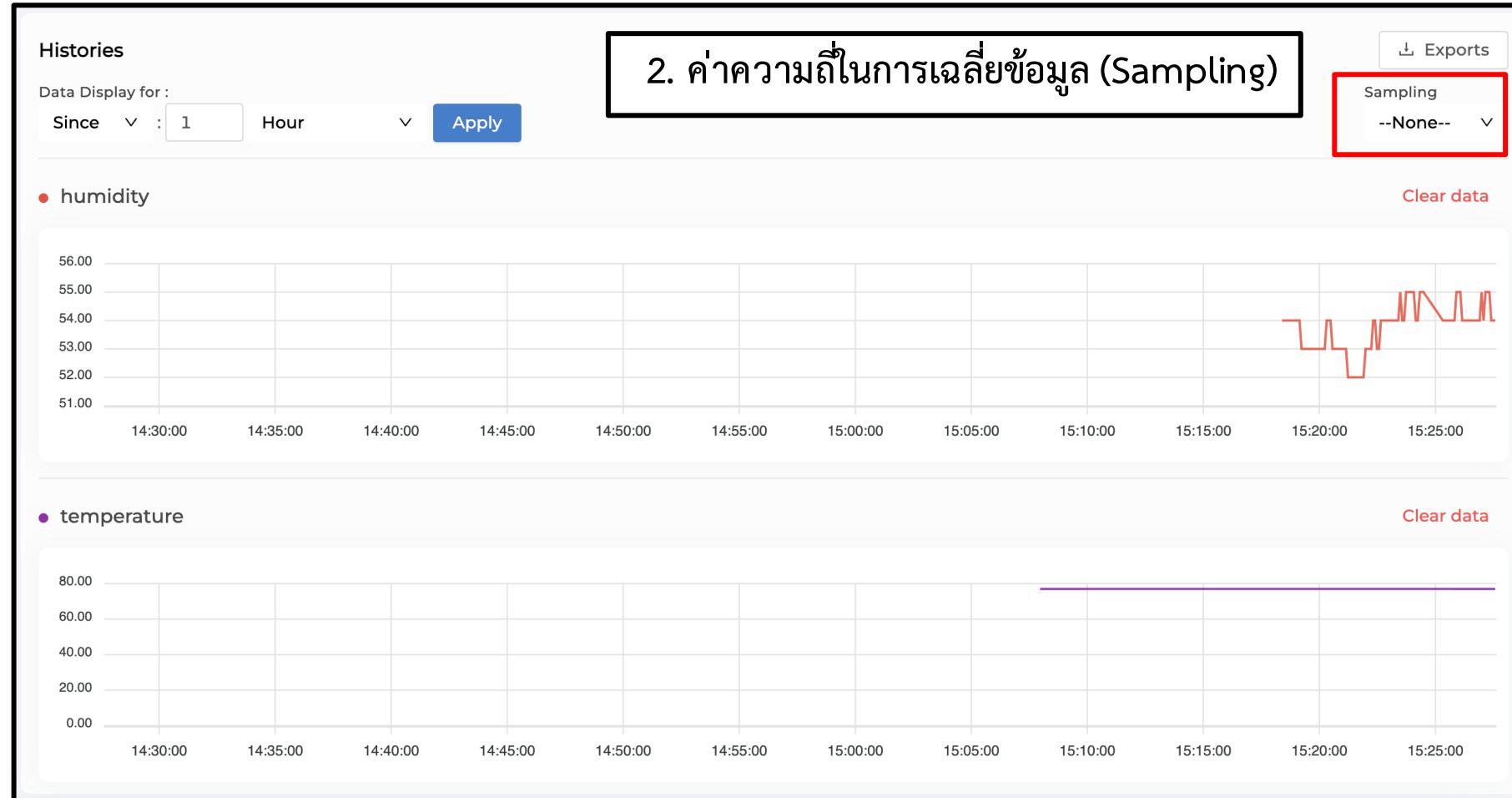
Data Display for :

From to : 2021-07-19 18:00 To 2021-07-19 19:00 Apply

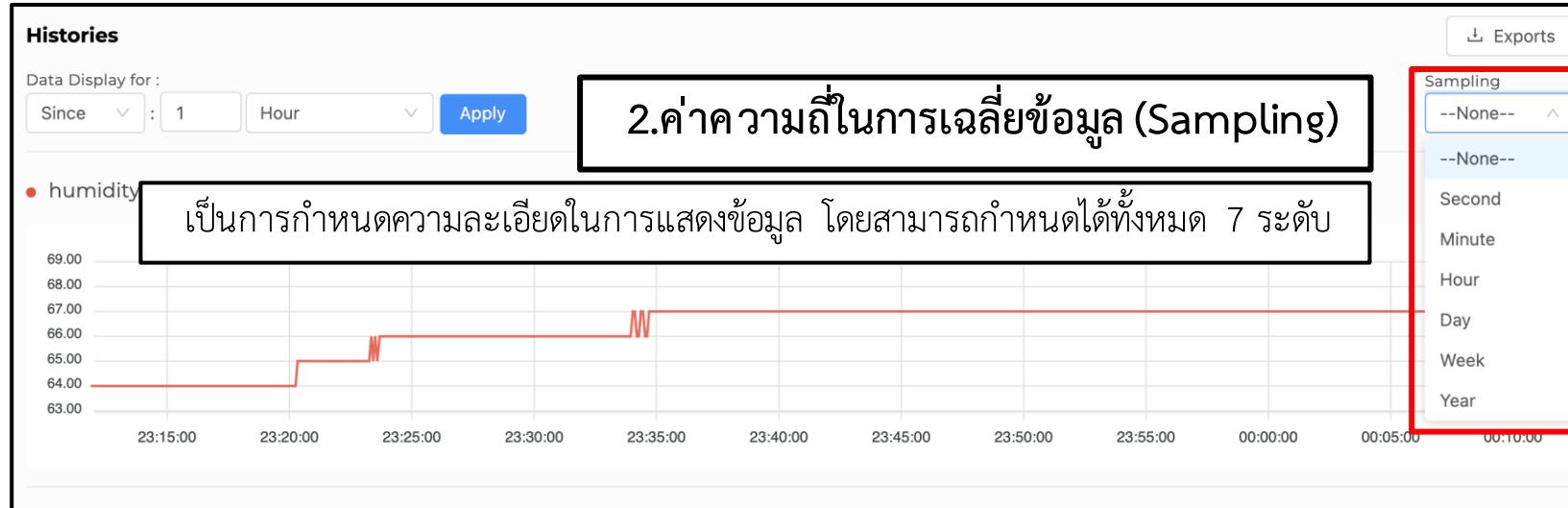
2. From to คือ การกำหนดช่วงเวลาแบบระบุทั้งเวลาเริ่มต้นและสิ้นสุดที่ต้องการให้แสดงข้อมูล โดยระบุวันเวลาเริ่มต้นและสิ้นสุดตามที่ต้องการ จากนั้นกดปุ่ม Apply ระบบจะดึงข้อมูลตามช่วงเวลาที่กำหนดมาแสดงในกราฟของทุกฟิลด์ข้อมูล การตั้งค่าเป็นดังรูปต่อไปนี้



การใช้งาน Device Feed



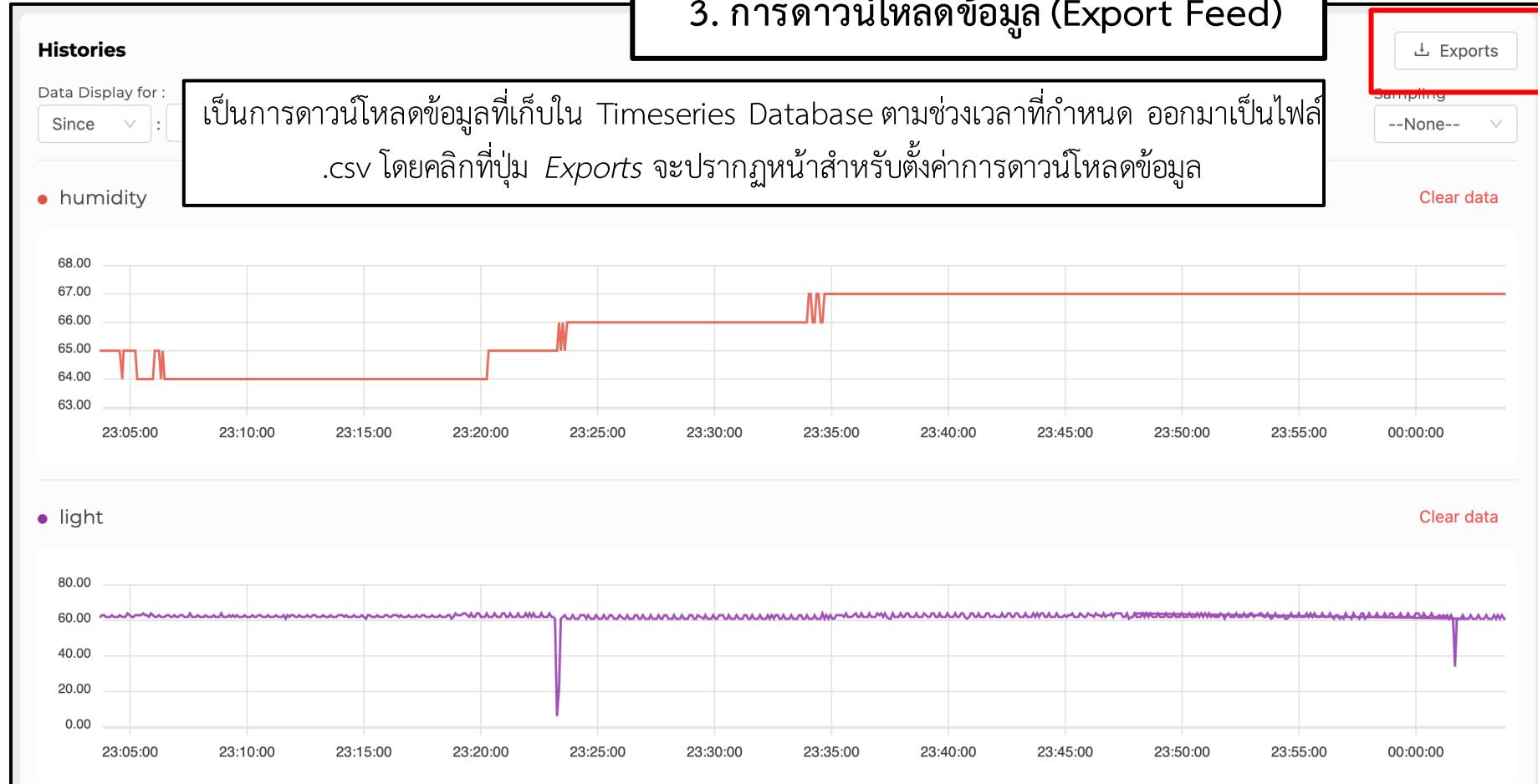
การใช้งาน Device Feed



- None คือ เป็นการแสดงข้อมูลที่มีความละเอียดสูงสุด โดยข้อมูลที่นำมาแสดงจะเป็นข้อมูลต้น (Raw Data) ที่ ณ ช่วงเวลาที่
- 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

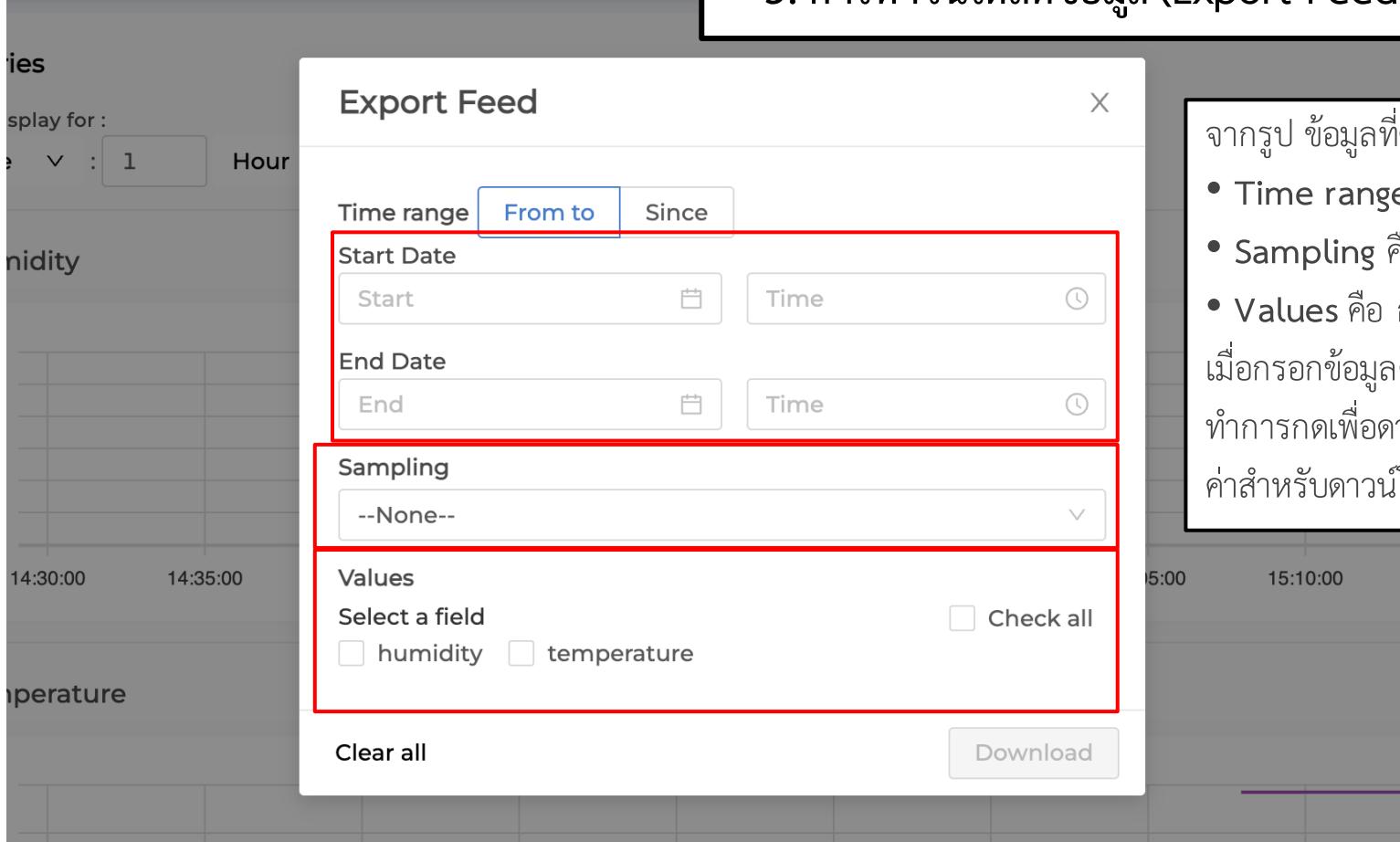
การใช้งาน Device Feed

3. การดาวน์โหลดข้อมูล (Export Feed)



การใช้งาน Device Feed

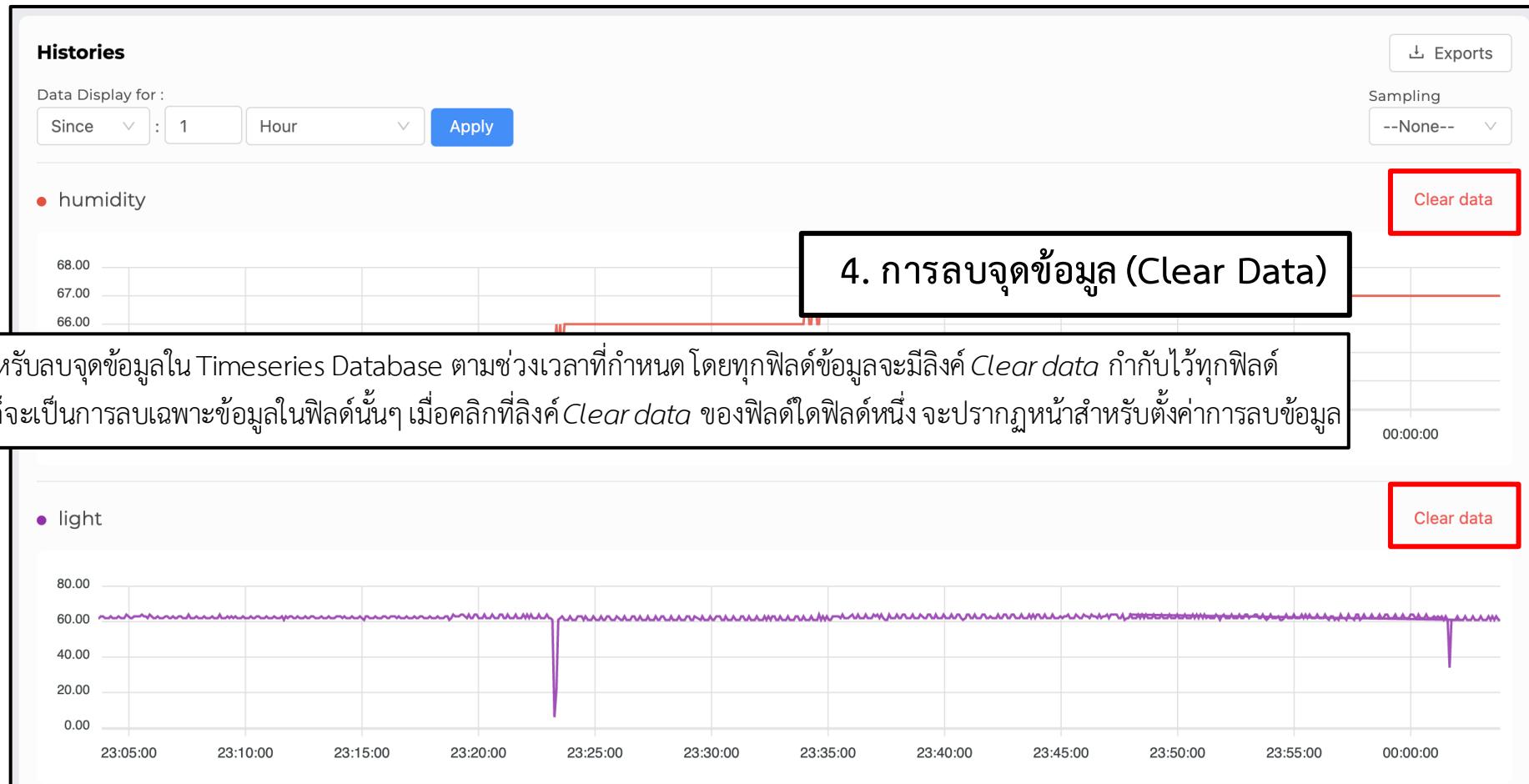
3. การดาวน์โหลดข้อมูล (Export Feed)



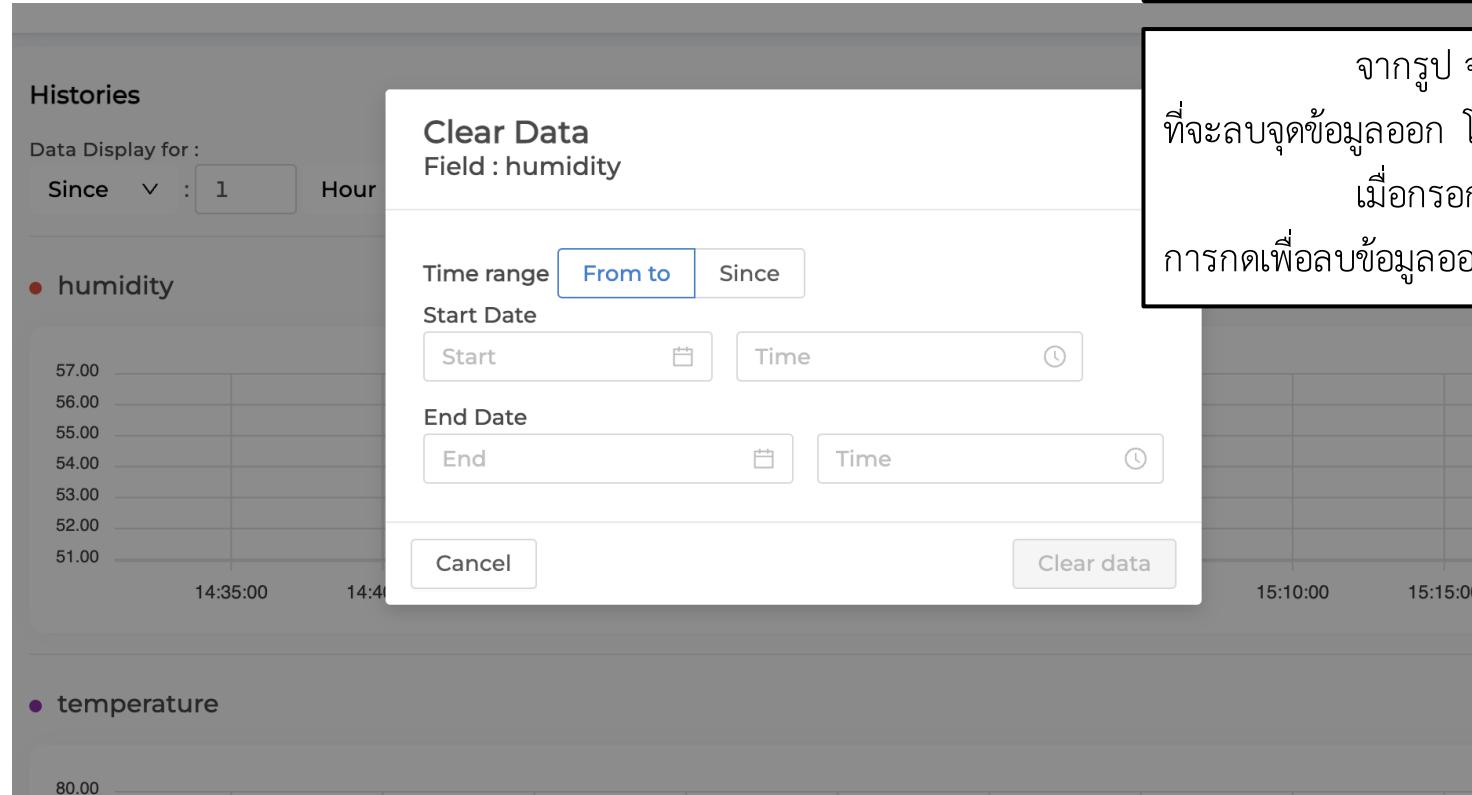
จากรูป ข้อมูลที่ต้องระบุสำหรับการดาวน์โหลดข้อมูล มีดังนี้

- **Time range** คือ ช่วงเวลาที่ต้องการข้อมูล
 - **Sampling** คือ การกำหนดความละเอียดของข้อมูล
 - **Values** คือ การเลือกฟิลเตอร์ข้อมูลที่ต้องการ
- เมื่อกรอกข้อมูลครบแล้วปุ่ม Download จะ Active ขึ้นมาให้สามารถกดได้ ทำการกดเพื่อดาวน์โหลดข้อมูล ส่วน Clear all ใช้สำหรับ Reset การตั้งค่าสำหรับดาวน์โหลดข้อมูล

การใช้งาน Device Feed



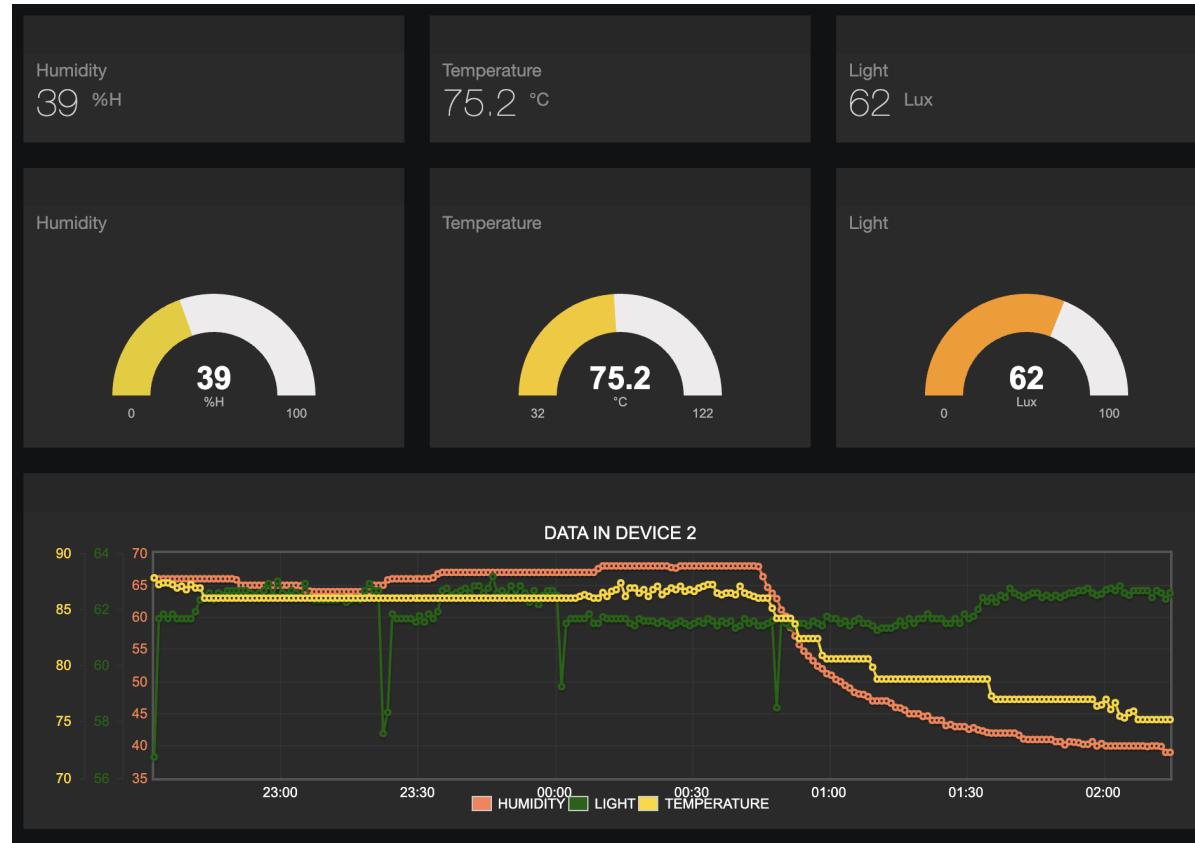
การใช้งาน Device Feed



4. การลบจุดข้อมูล (Clear Data)

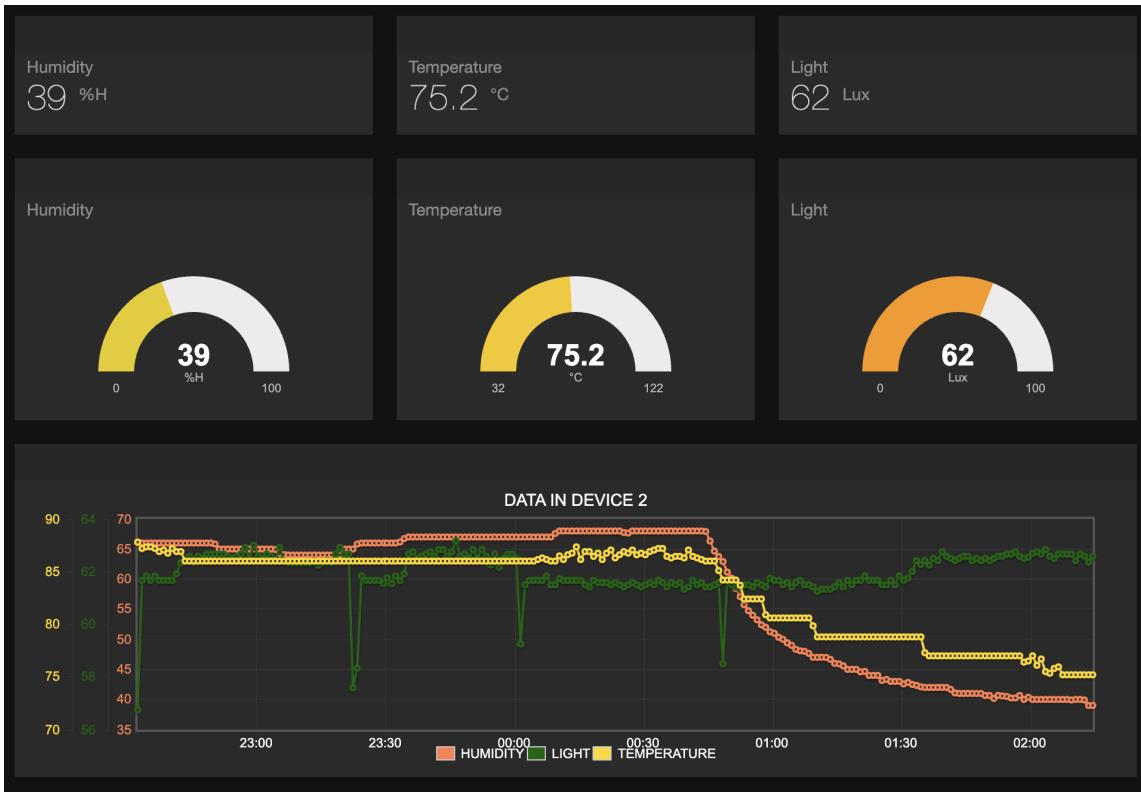
จากรูป จะแสดงชื่อฟิล์ตที่ต้องลบข้อมูล และต้องกำหนดช่วงเวลา (Time range) ที่จะลบจุดข้อมูลออก โดยรูปแบบการกำหนดช่วงเวลาจะเหมือนในข้อ เมื่อกด 'Clear data' จะ Active ขึ้นมาให้สามารถกดได้ ทำ การกดเพื่อลบข้อมูลออกจาก Timeseries Database

การสร้างและใช้งาน Dashboard ด้วย NETPIE Freeboard



Freeboard คืออะไร

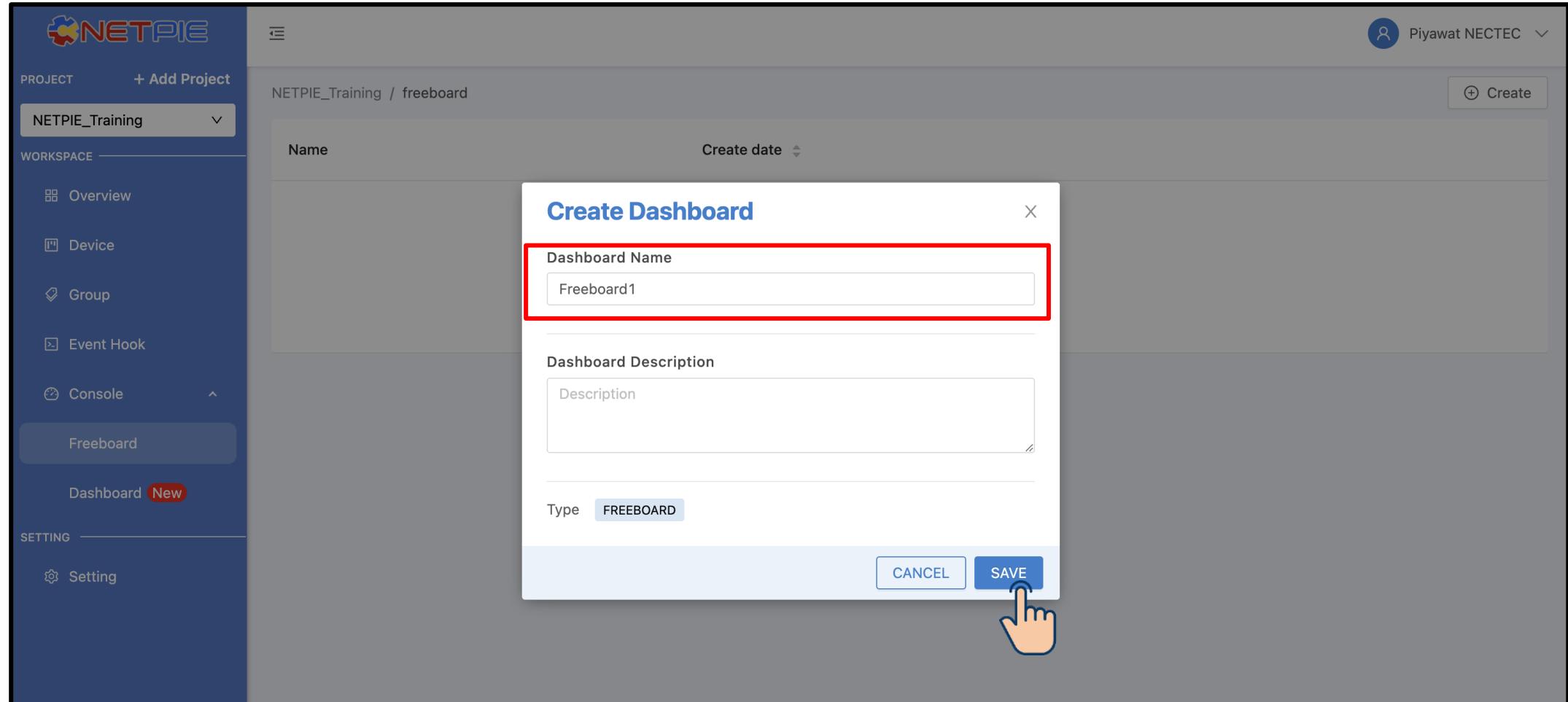
NETPIE Freeboard คือ Dashboard ที่ใช้สำหรับนำข้อมูลที่เก็บอยู่ใน Platform มาแสดงผลในรูปแบบต่างๆ เป็นเหมือนช่องทางให้ผู้ใช้สามารถติดตามหรือควบคุมการทำงานของ Device ของตัวเอง



การสร้าง Freeboard

The screenshot shows the NETPIE web interface. On the left, a sidebar menu includes 'PROJECT' (selected), '+ Add Project', 'NETPIE_Training' (selected), 'WORKSPACE' (selected), 'Overview', 'Device', 'Group', 'Event Hook', 'Console' (selected), 'Freeboard' (highlighted with a red box), and 'Dashboard' (New). On the right, the main area shows the 'NETPIE_Training / freeboard' workspace. It has a header with a user icon and 'Piyawat NECTEC'. A 'Create' button is highlighted with a red box and a hand cursor. Below it, there's a table with columns 'Name' and 'Create date' (sorted by Create date). The table shows 'No Data' with an icon of a folder.

การสร้าง Freeboard



การสร้าง Freeboard

The screenshot shows the NETPIE platform interface. On the left, a sidebar menu is visible with the following sections and items:

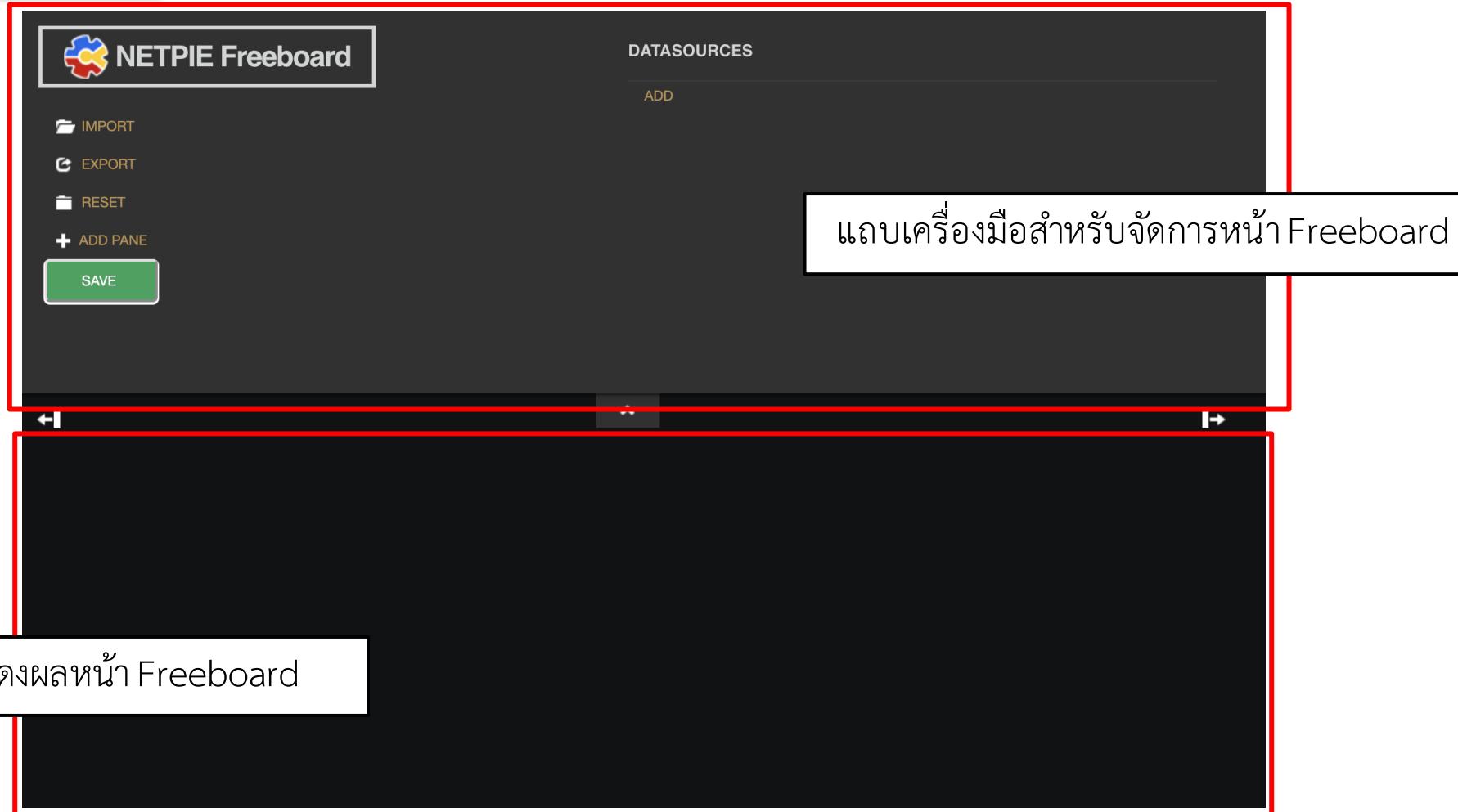
- PROJECT**: + Add Project, NETPIE_Training (selected)
- WORKSPACE**: Overview, Device, Group, Event Hook, Console, Freeboard (selected), Dashboard (New)
- SETTING**: Setting

The main workspace area displays a list of Freeboards under the project "NETPIE_Training / freeboard". The list includes:

Name	Create date
Freeboard1	19 Jan 2023, 15:33

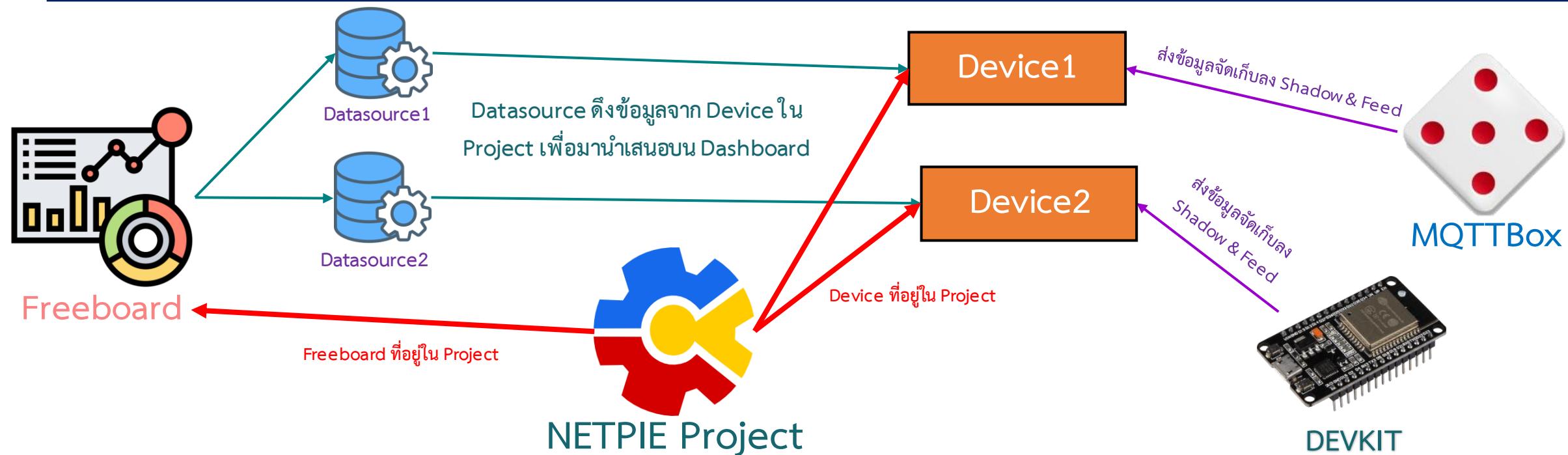
A red box highlights the "Freeboard1" row, and a hand cursor icon is positioned over the "Freeboard1" link.

การสร้าง Freeboard

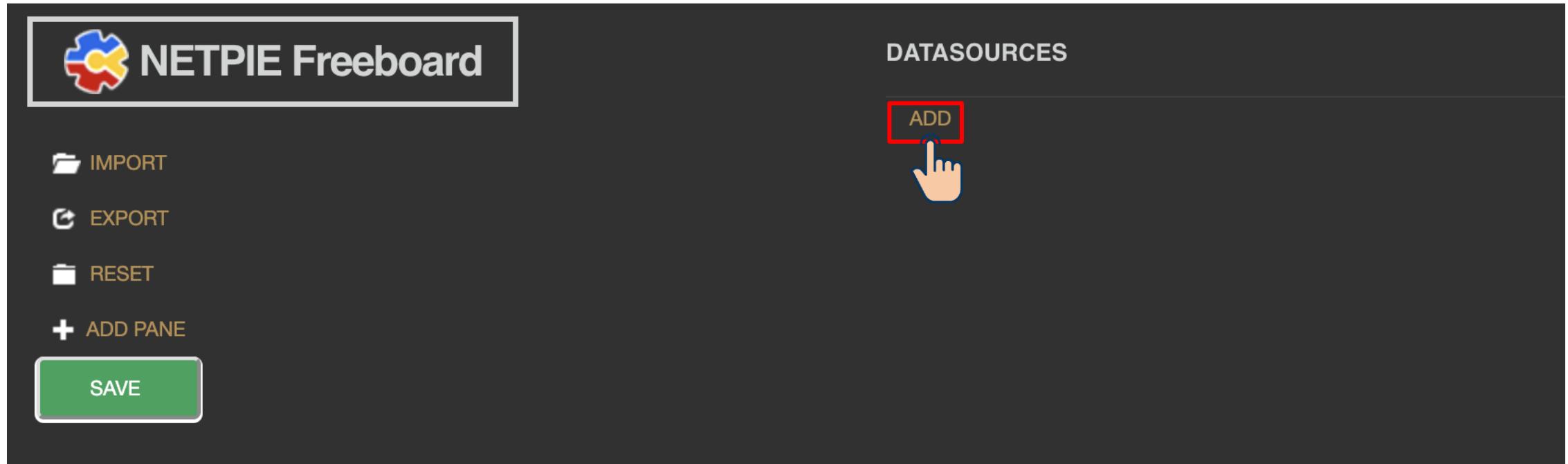


Datasource บน Freeboard

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



Datasource عن Freeboard



Datasource บน Freeboard

ทดสอบการเพิ่ม Datasource ด้วยการดึงข้อมูลจาก Device2

DATASOURCE

NAME ชื่อ Datasource
(เป็นชื่ออะไรก็ได้ แต่ไม่ควรเว้นวรรค ถ้าจำเป็นควรใช้ _ หรือ - แทน)

DEVICE ID Client ID ของ Device

Client ID ของ Device ที่ต้องการอ่านข้อมูล

DEVICE TOKEN Token ของ Device

Token ของ Device ที่ต้องการอ่านข้อมูล

SUBSCRIBED TOPICS เปิดการใช้งาน FEED

Topic ที่ต้องการ Subscribe

FEED NO การตั้งค่าส่วนของการแสดงผล Feed

SINCE 6

Hour

Display data points since ... ago.

DOWN SAMPLING 1

Minute

Resolution of the data points.

SAVE CANCEL

The diagram illustrates the configuration of a Datasource in Freeboard. It maps various fields to their meanings:

- NAME: ชื่อ Datasource (Name of the Datasource, can be anything but should not be empty)
- DEVICE ID: Client ID ของ Device (Client ID of the Device required to read data)
- DEVICE TOKEN: Token ของ Device (Token of the Device required to read data)
- SUBSCRIBED TOPICS: เปิดการใช้งาน FEED (Enable FEED usage)
- FEED: การตั้งค่าส่วนของการแสดงผล Feed (Feed configuration settings)
- SINCE: Display data points since ... ago. (Display data points from ... ago)
- DOWN SAMPLING: Resolution of the data points. (Resolution of the data points)

Datasource บน Freeboard

ทดสอบการเพิ่ม Datasource ด้วยการดึงข้อมูลจาก Device2

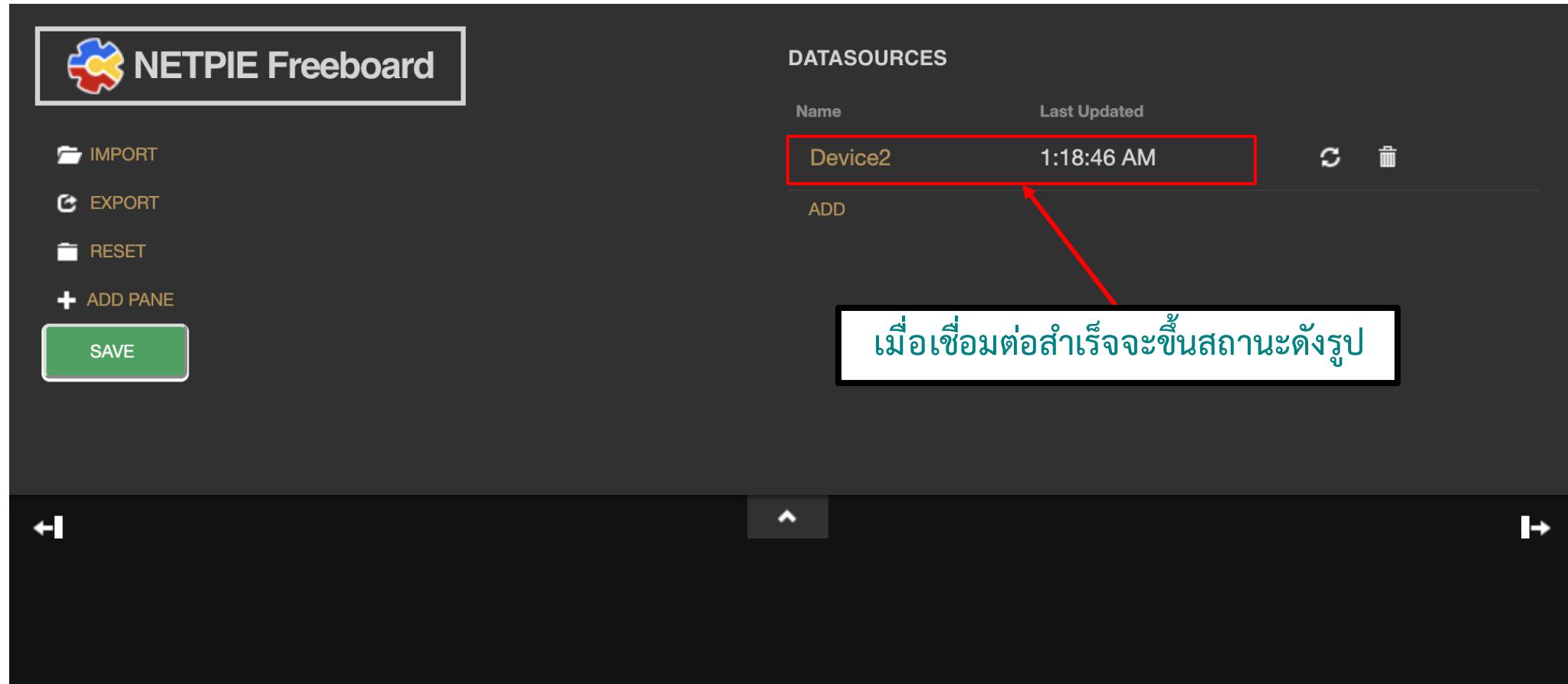
DATASOURCE

NAME	Device2
DEVICE ID	4afbad5c-201c-4be5-9771-c7d499748624 Client ID ของ Device ที่ต้องการอ่านข้อมูล
DEVICE TOKEN	pu5ehH6TiF1Sqv7JZvnsKUUKL53TTy7x Token ของ Device ที่ต้องการอ่านข้อมูล
SUBSCRIBED TOPICS	
FEED	YES <input checked="" type="checkbox"/>
SINCE	6 Hour Display data points since ... ago.
DOWN SAMPLING	1 Minute Resolution of the data points.

SAVE CANCEL

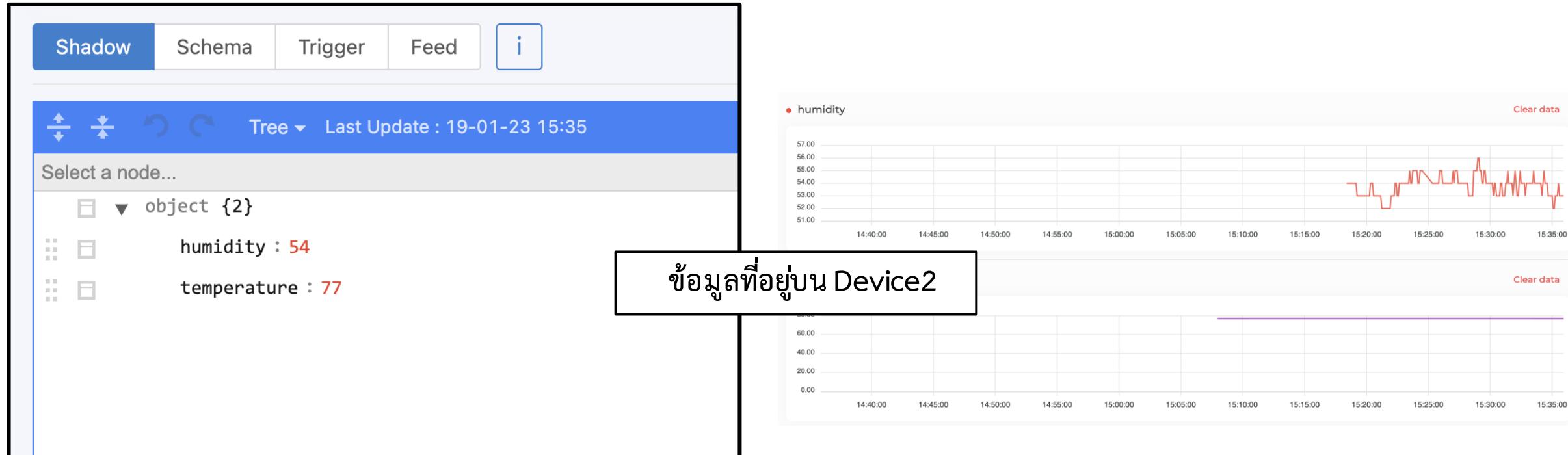
ตัวอย่างการตั้งค่าต่างๆ บน Datasource

Datasource บน Freeboard

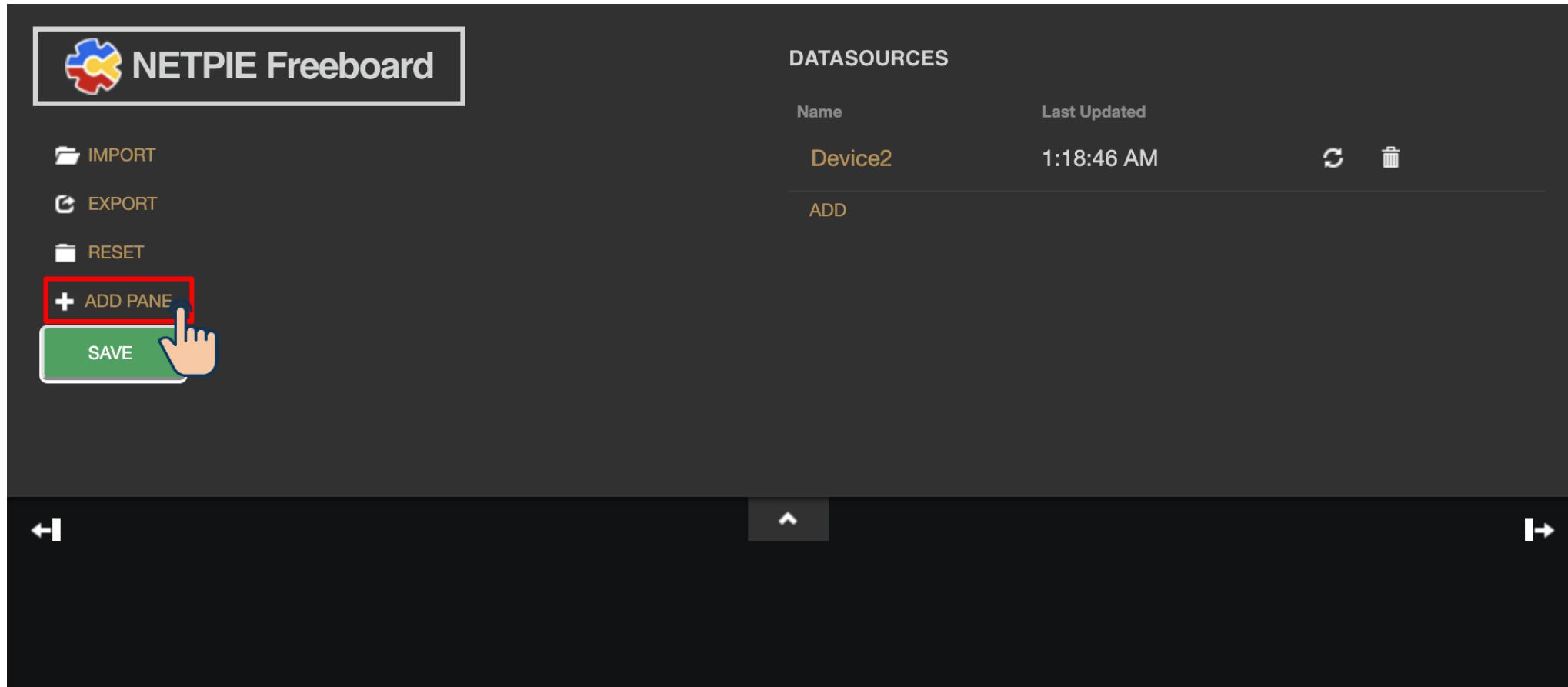


การนำเสนอดанны่

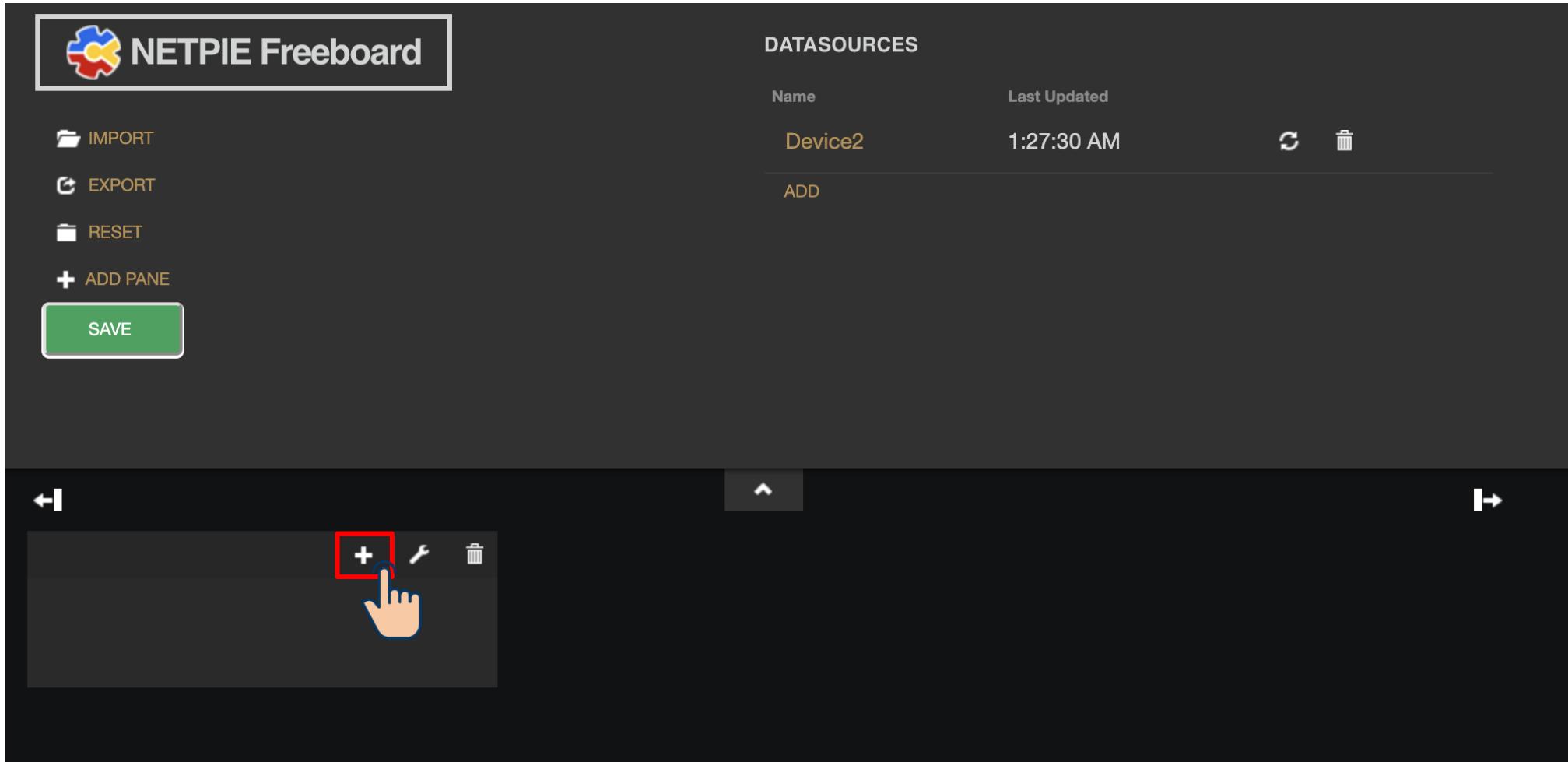
เราจะนำข้อมูลต่างๆที่อยู่ใน Device2 ทั้งใน Device Shadow และ Device Feed มานำเสนอบนเครื่องมือนำเสนอต่างๆ ซึ่งเรียกว่า Widget โดยมีอยู่มากบน Freeboard



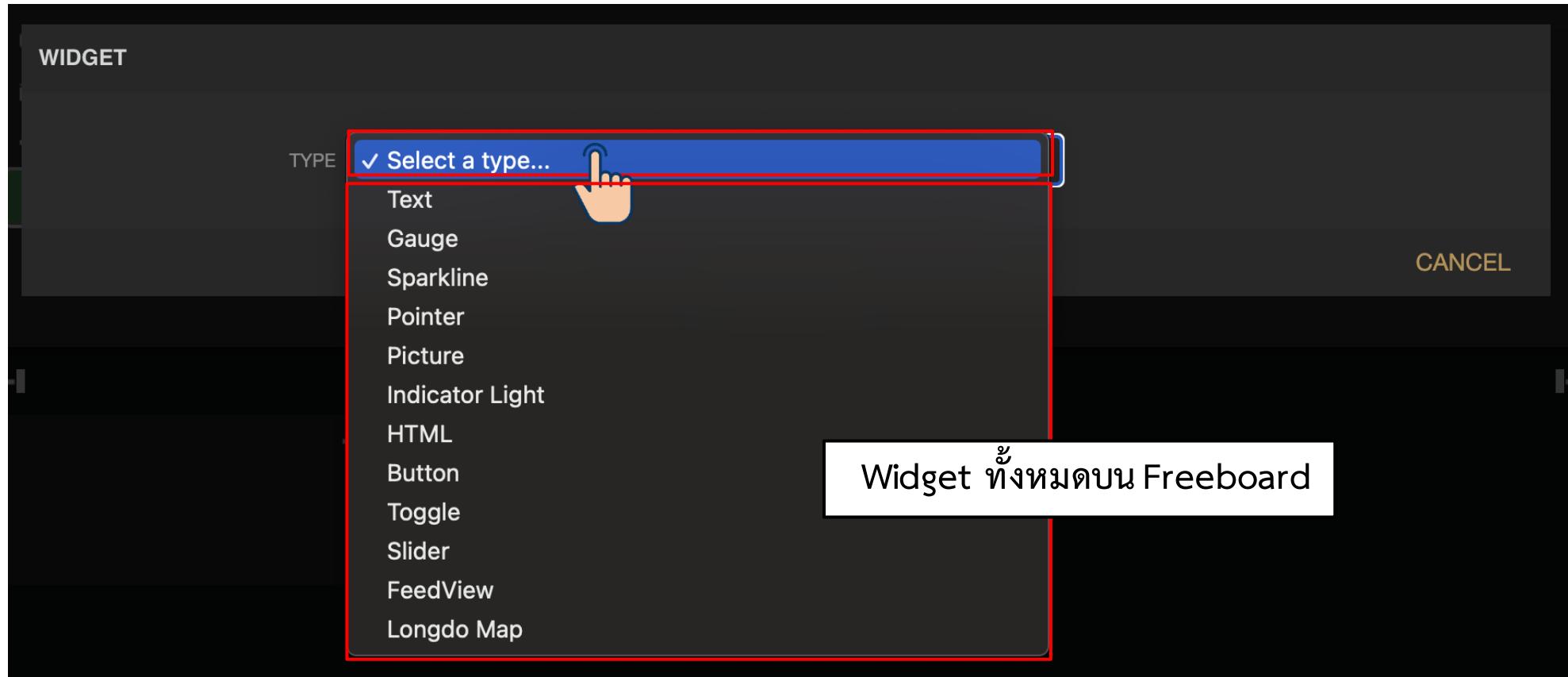
การสร้าง Widget



การสร้าง Widget



การสร้าง Widget



การสร้าง Widget

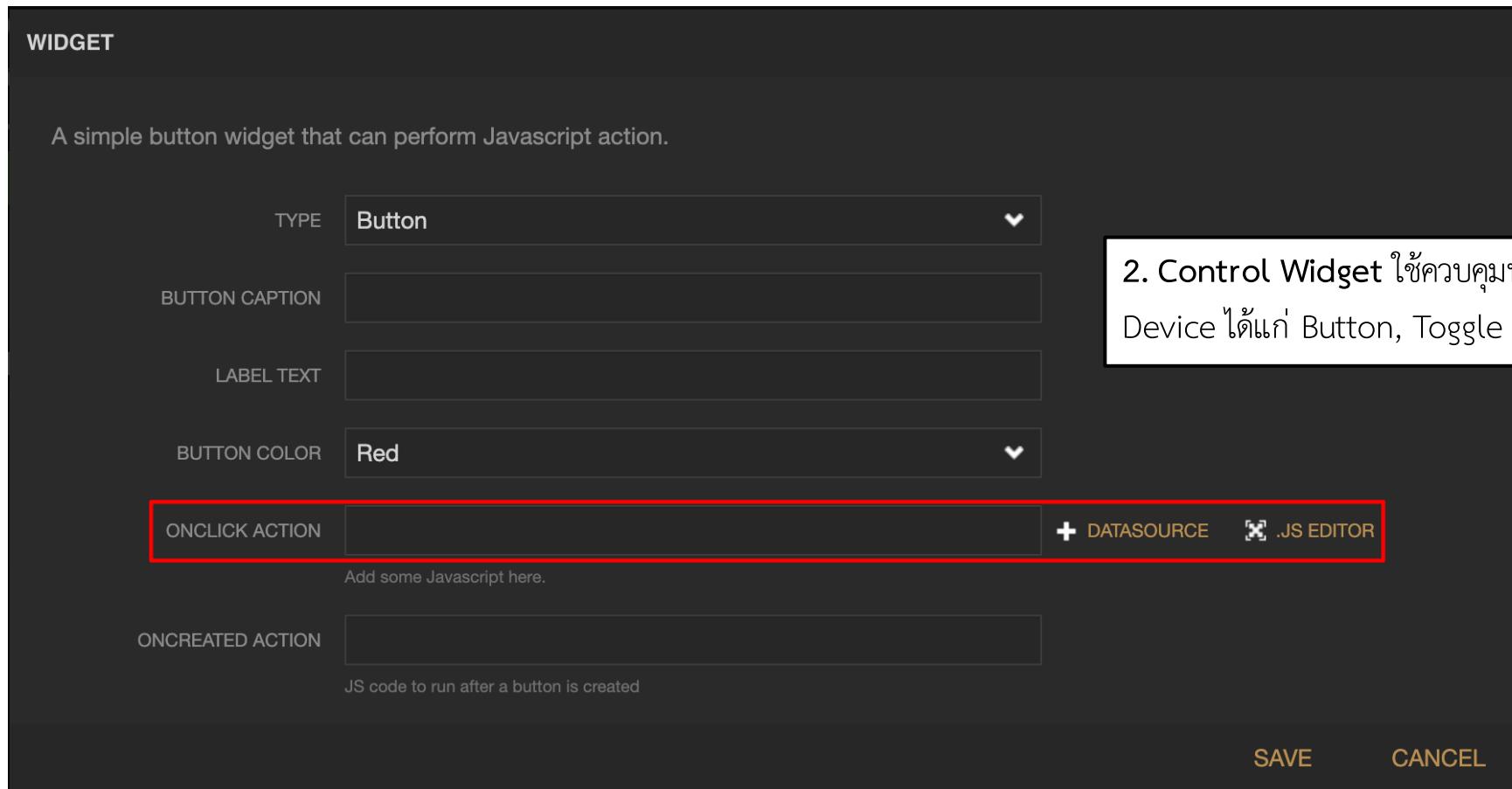
การเลือก Widget ต่างๆ ของ Freeboard เพื่อใช้ในการแสดงข้อมูลหรือควบคุมการทำงานของ Device โดย Widget จะแยกเป็น 2 ประเภท



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

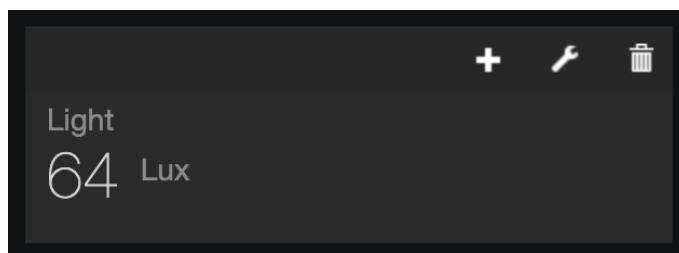
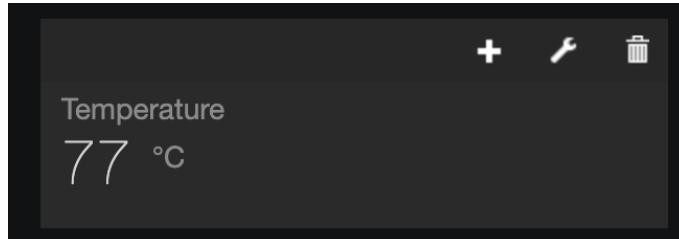
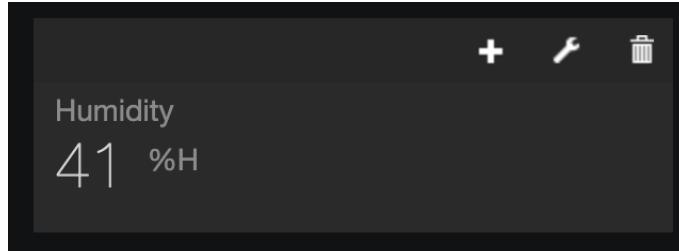
การสร้าง Widget

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

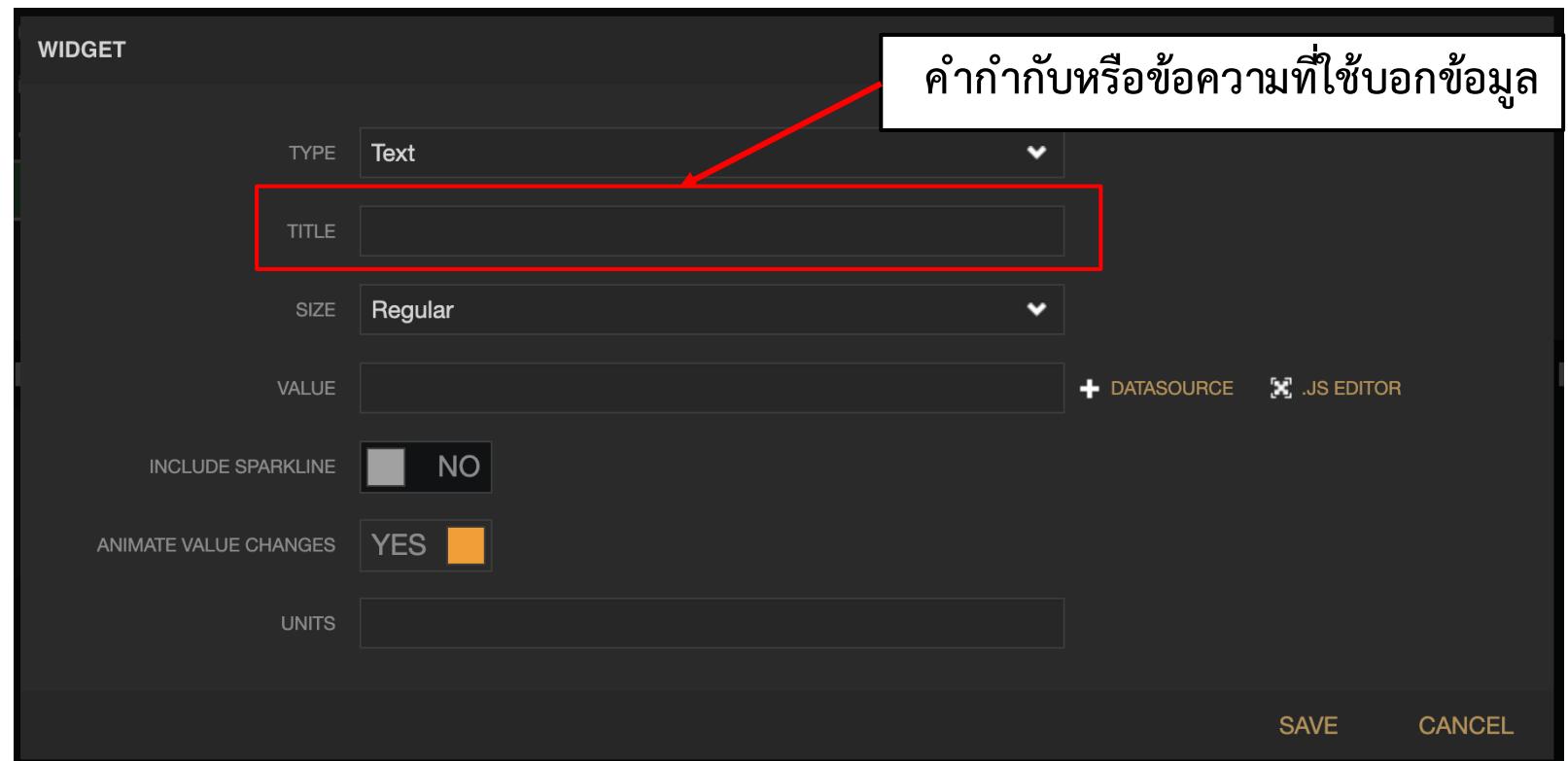


การสร้าง Widget : Text

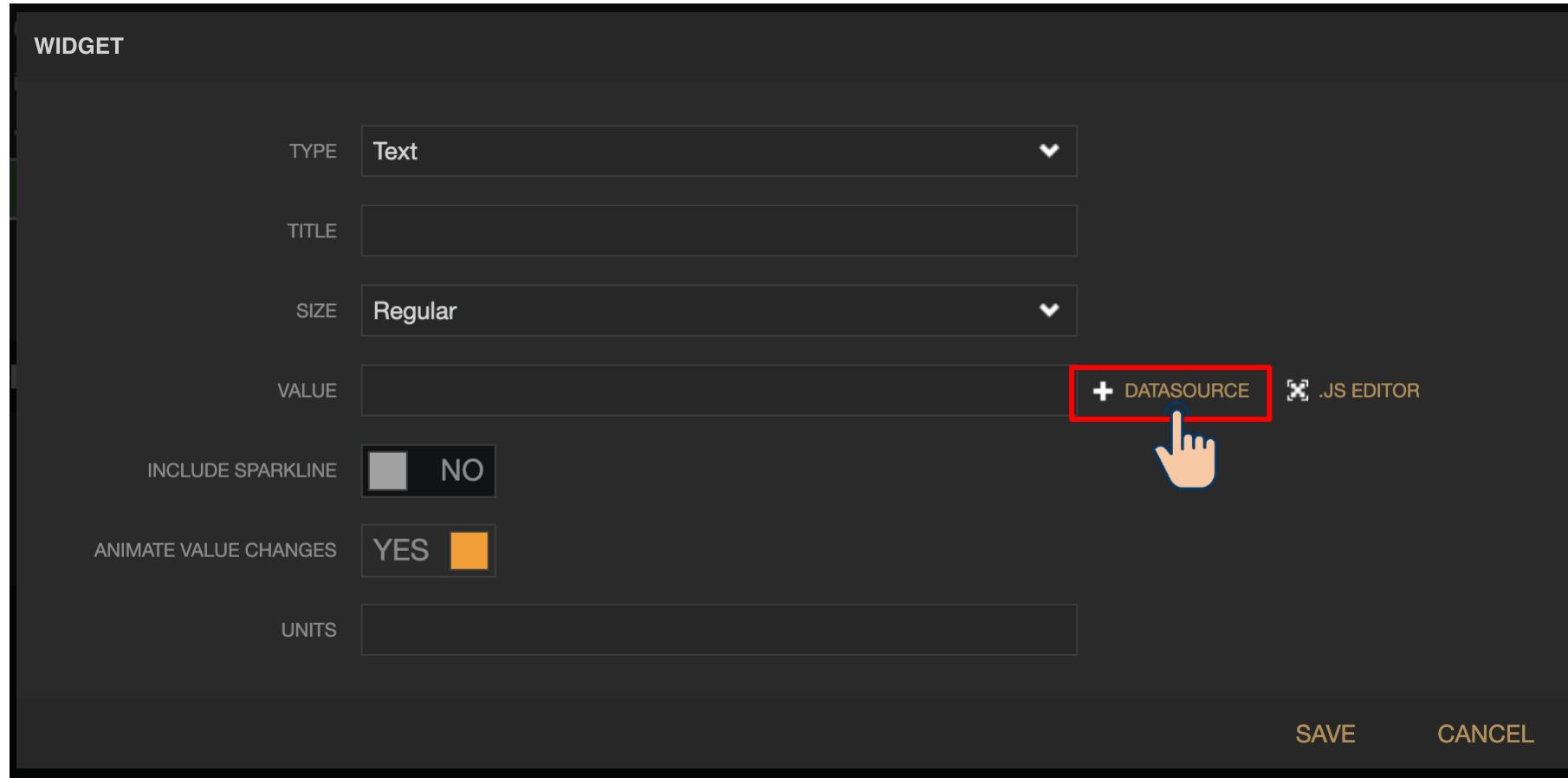
ตัวอย่าง Widget : Text



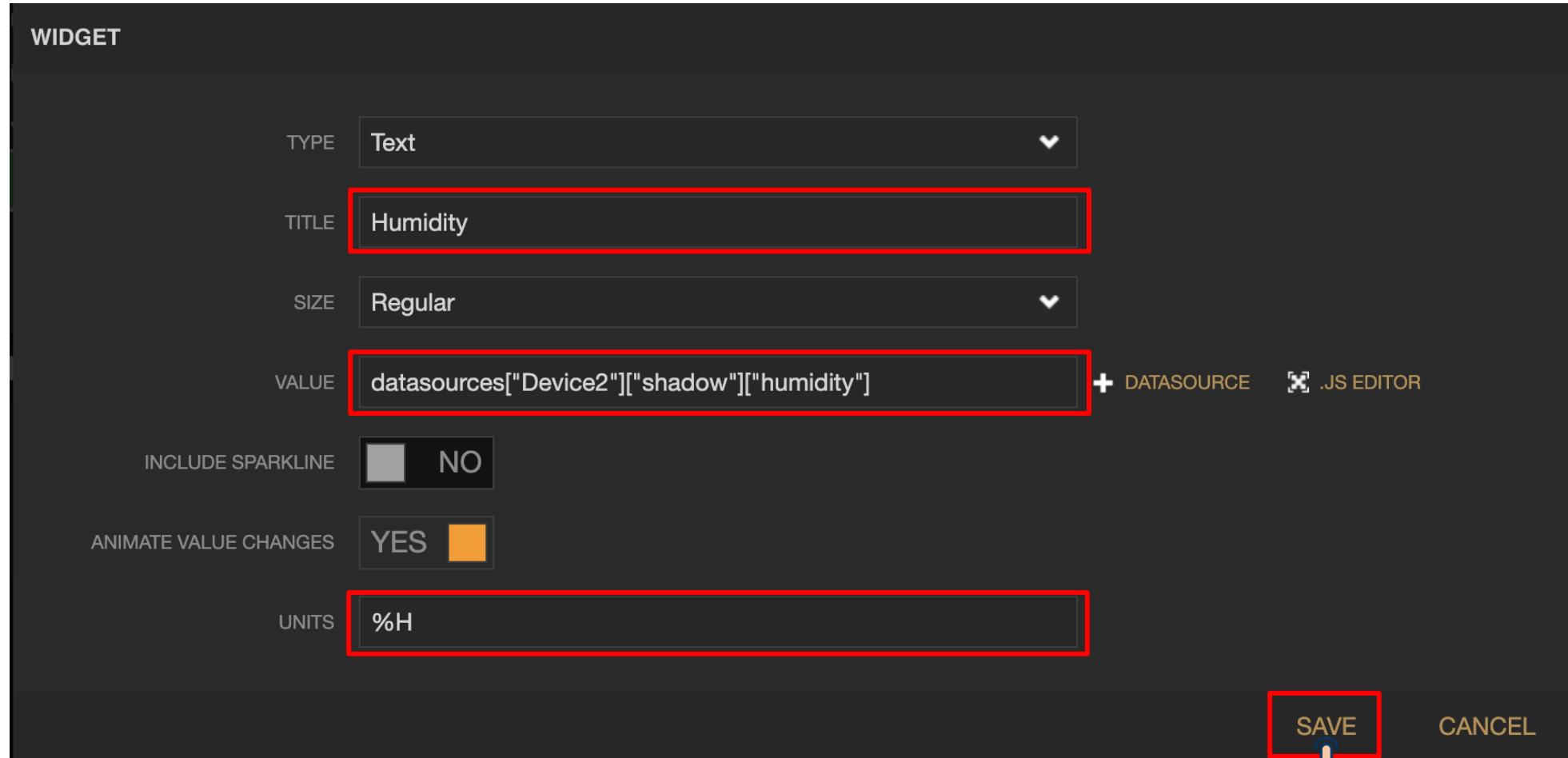
เป็น Widget สำหรับแสดงผลข้อความ



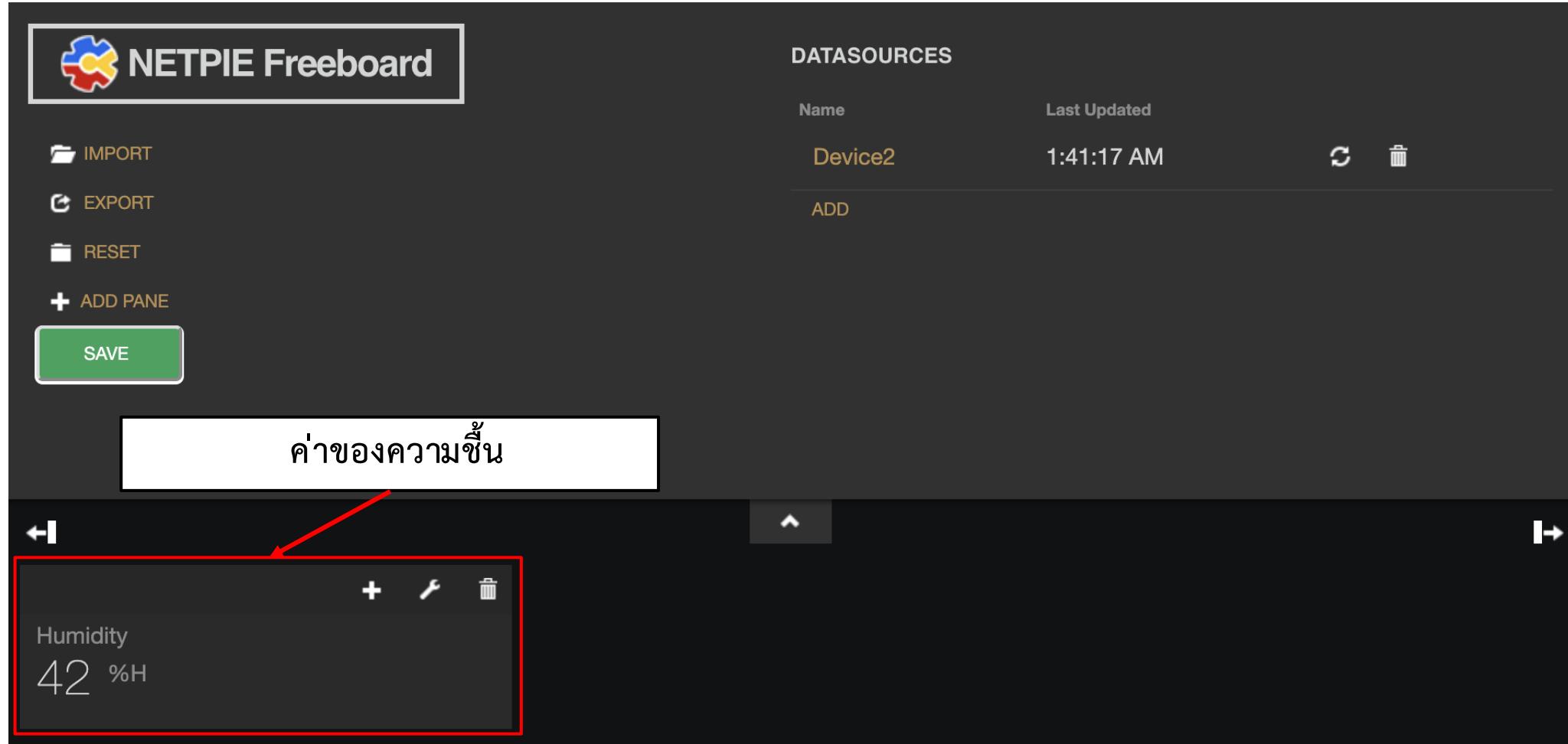
การสร้าง Widget : Text



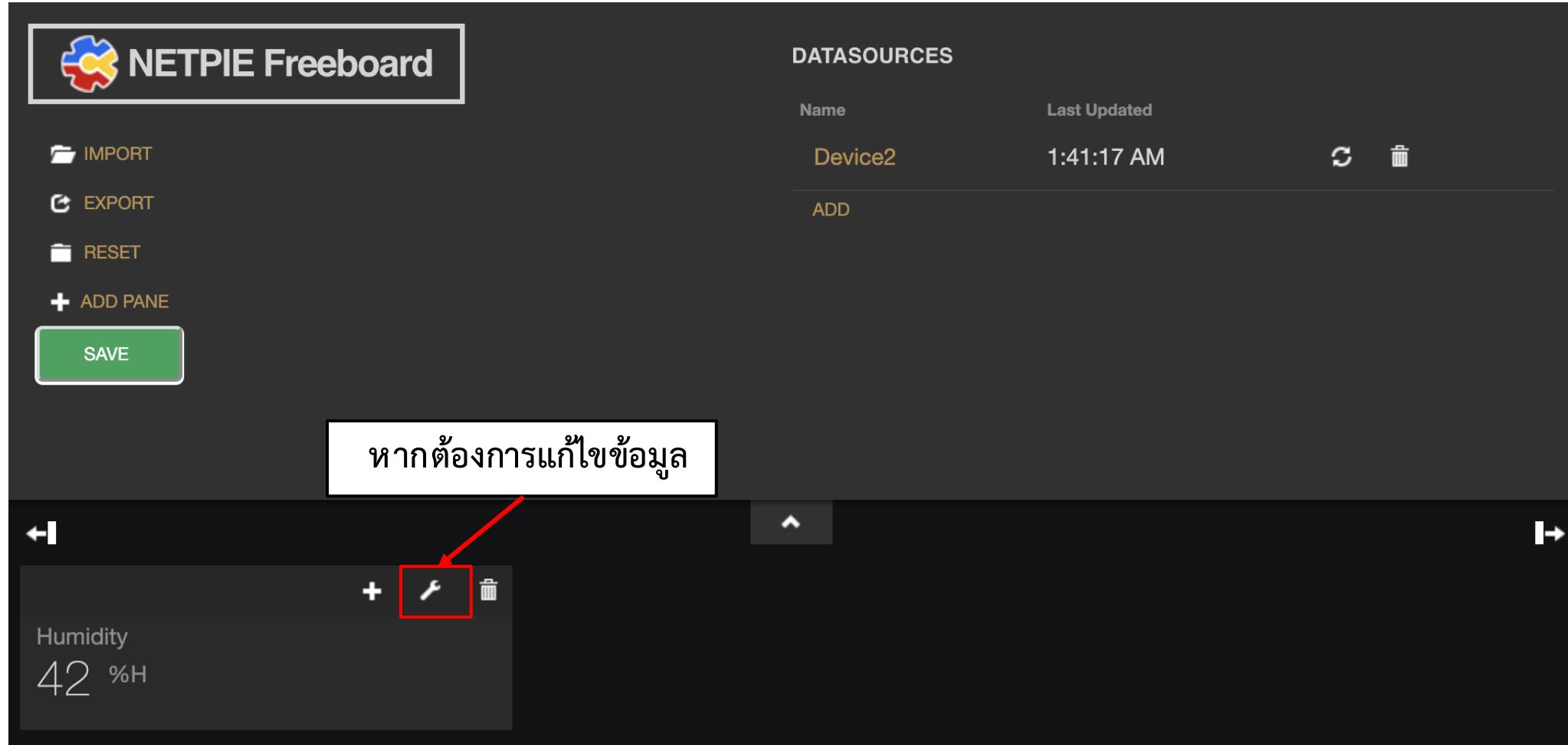
การสร้าง Widget : Text



การสร้าง Widget : Text



การสร้าง Widget : Text



การสร้าง Widget : Text

The screenshot shows the NETPIE Freeboard interface. On the left, there's a sidebar with buttons for IMPORT, EXPORT, RESET, ADD PANE, and a large green SAVE button. The main area has a header 'DATASOURCES' with a table showing one entry: 'Device2' last updated at '1:43:32 AM'. Below the table is an 'ADD' button. In the center, there's a text box containing the message: 'ทดสอบสร้าง Text สำหรับแสดงผล Temperature และ Light ด้วยตัวเอง'. Below this text box, there are three data cards: 'Humidity' (41 %H), 'Temperature' (77 °C), and 'Light' (64 Lux). The 'Temperature' card is highlighted with a red border and a red arrow points from the explanatory text above it to this card.

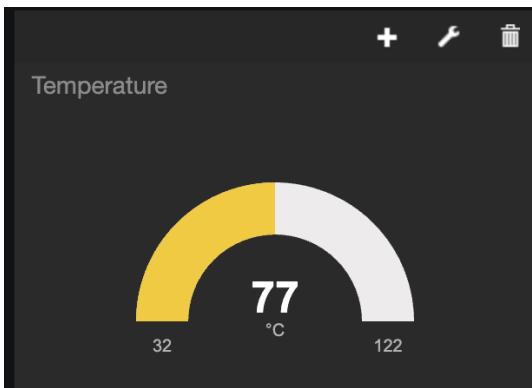
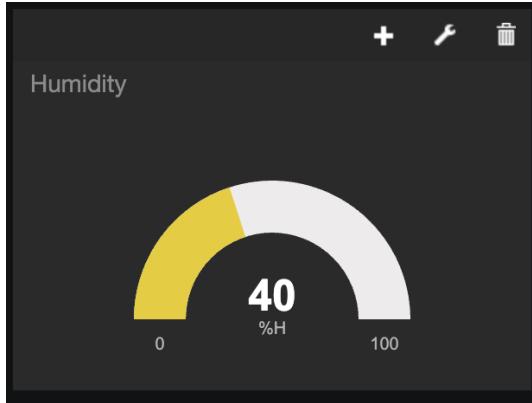
ทดสอบสร้าง Text สำหรับแสดงผล
Temperature และ Light ด้วยตัวเอง

Temperature
77 °C

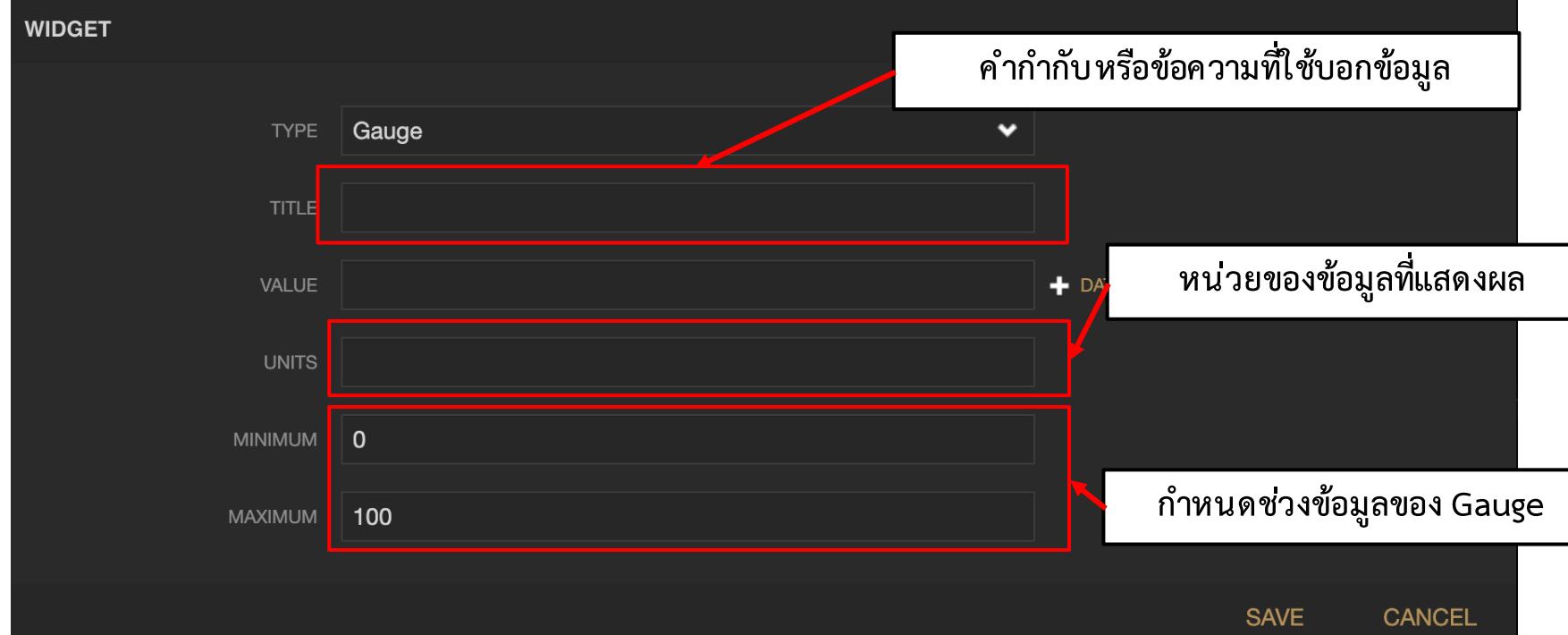
Light
64 Lux

การสร้าง Widget : Gauge

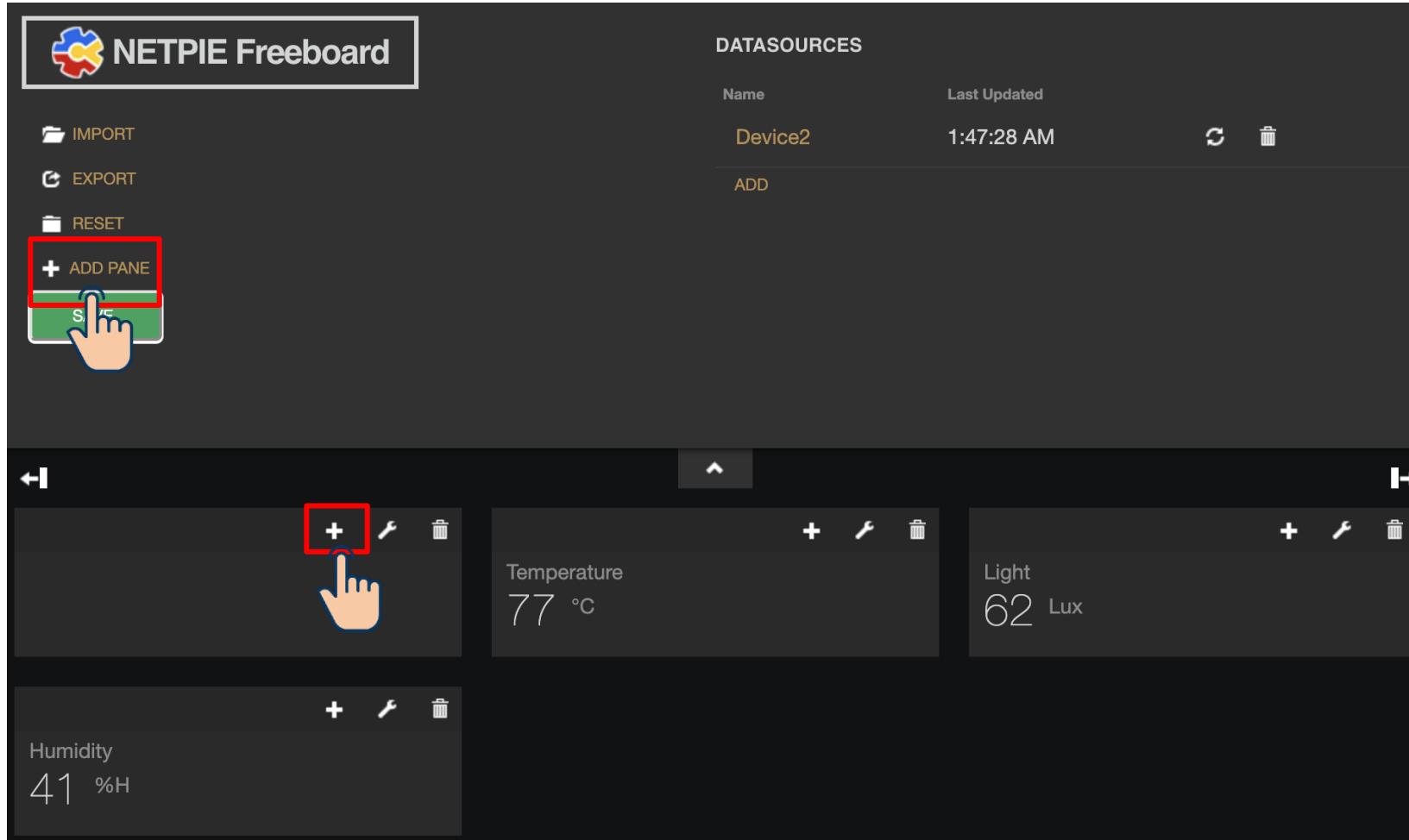
ตัวอย่าง Widget : Gauge



เป็น Widget สำหรับแสดงผลข้อมูลแบบ Gauge



การสร้าง Widget : Gauge



การสร้าง Widget : Gauge

WIDGET

TYPE

TITLE

VALUE + DATASOURCE .JS EDITOR

UNITS

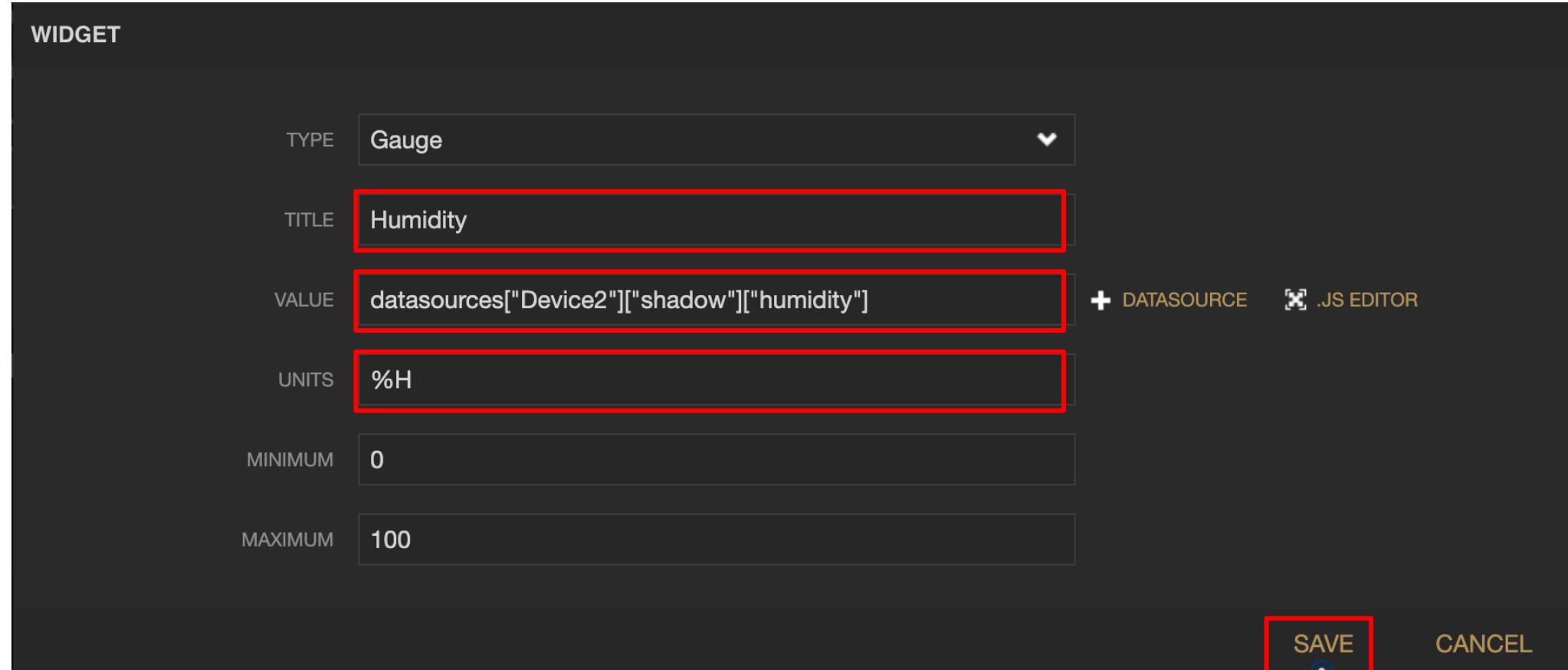
MINIMUM

MAXIMUM

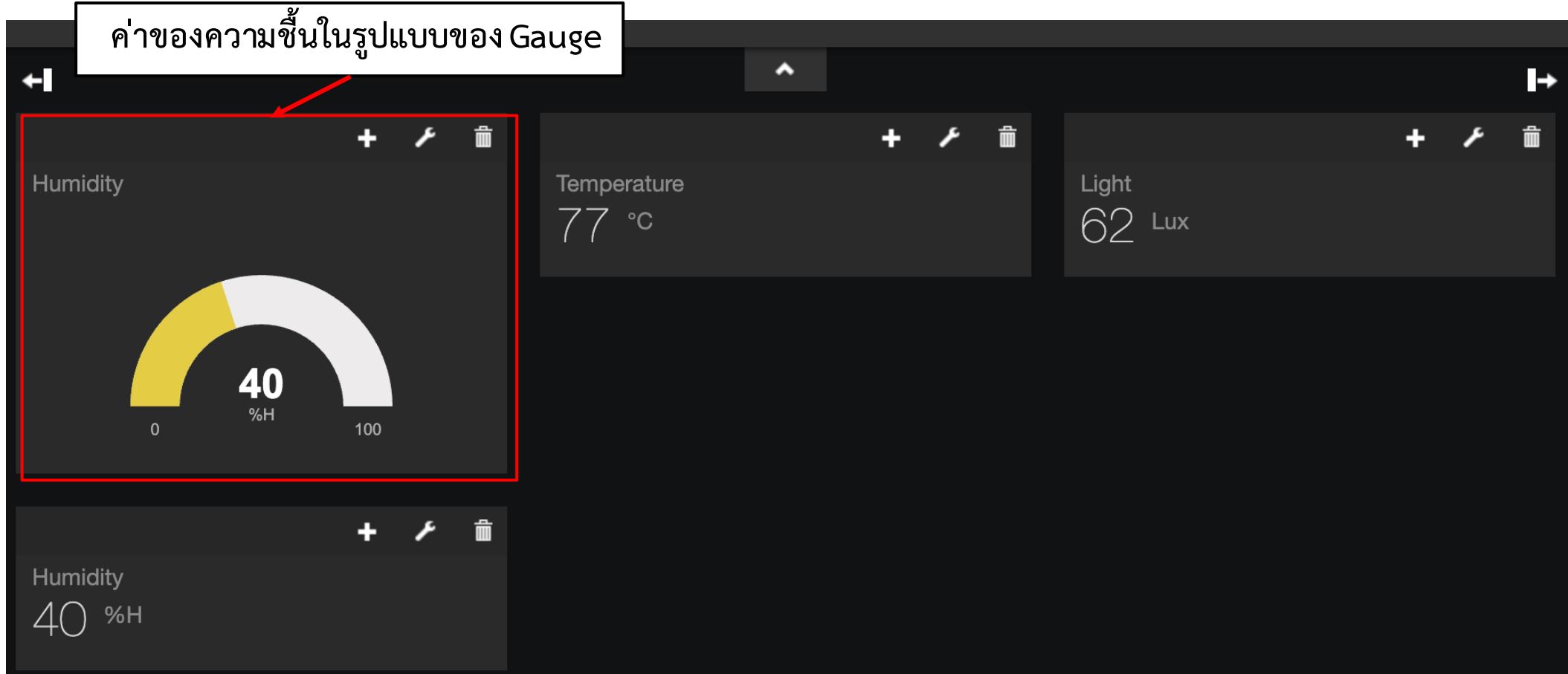
SAVE CANCEL

The screenshot shows the 'Widget' configuration screen for a 'Gauge' type. It includes fields for Title, Value (with a highlighted 'DATASOURCE' button), Units, Minimum (0), and Maximum (100). The 'SAVE' and 'CANCEL' buttons are at the bottom.

การสร้าง Widget : Gauge

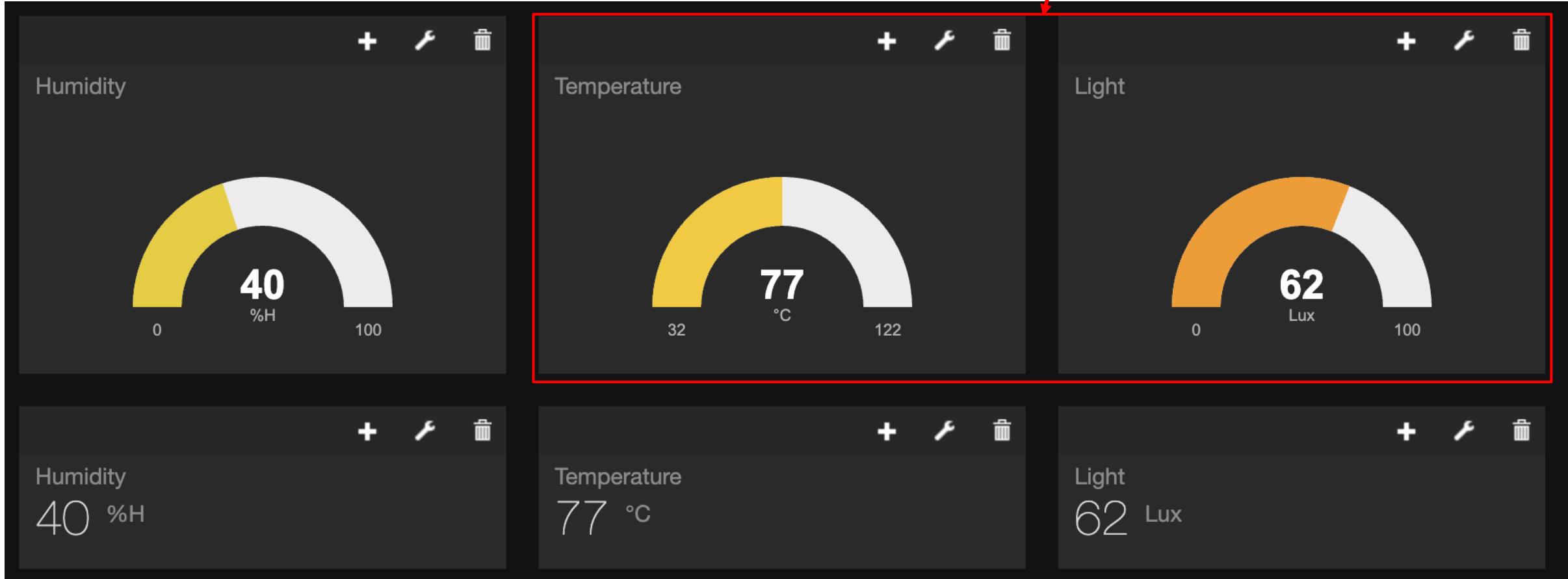


การสร้าง Widget : Gauge



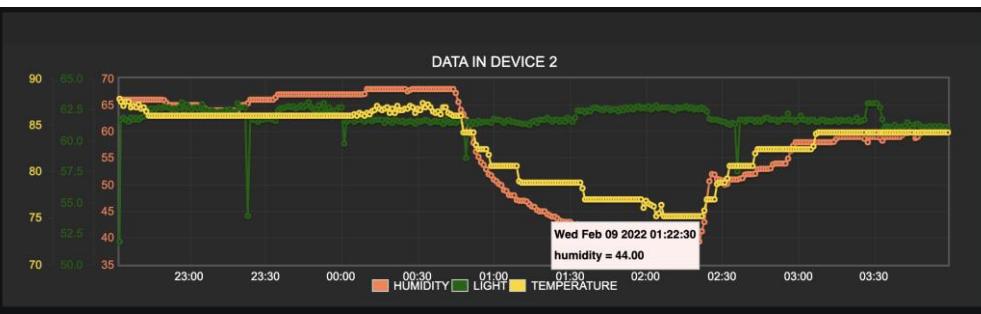
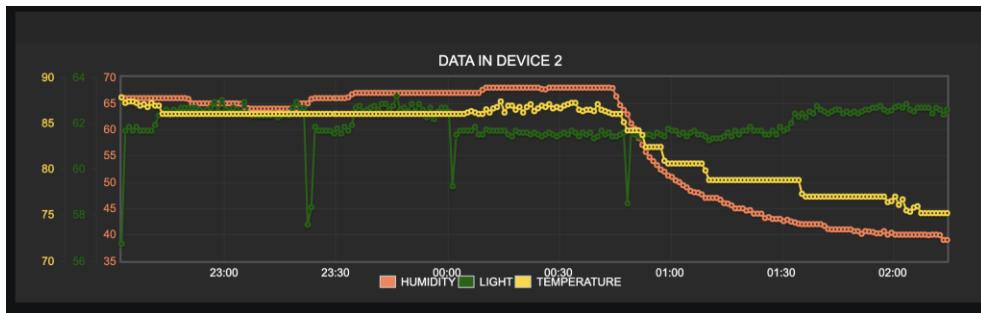
การสร้าง Widget : Gauge

ทดสอบสร้าง Text สำหรับแสดงผล
Temperature และ Light ด้วยตัวเอง



การสร้าง Widget : FeedView

ตัวอย่าง Widget : FeedView



เป็น Widget สำหรับแสดงผลข้อมูลแบบกราฟ

ข้อมูลที่ต้องการนำเสนอในกราฟ โดยใส่ชื่อข้อมูลที่ต้องการนำเสนอลงไป หากเว้นว่างไว้จะนำเสนอทุกข้อมูล

WIDGET

TYPE: FeedView

TITLE: [empty]

DATA SOURCE: [empty] + Datasource JS EDITOR

FILTER: [empty]
Data fields separated with comma e.g. temp,humid,light. Blank means display all fields.

TYPE OF CHART: Line

X AXIS TITLE: [empty]

Y AXIS TITLE: [empty]

BEGIN AT 0: NO

LINE COLORS: [empty]
enter the color set separated by comma e.g. #ff0000,#00ff00,#0000ff or leave blank for the default color set

MAKER: YES

MULTIPLE AXIS: YES

HEIGHT BLOCKS: 4

SAVE CANCEL

การตั้งค่าต่างๆของกราฟ

การสร้าง Widget : FeedView

WIDGET

TYPE: FeedView

TITLE:

DATA SOURCE:

FILTER:

+ DATASOURCE  [X] JS EDITOR

Data fields separated with comma e.g. temp,humid,light. Blank means display all fields.

TYPE OF CHART: Line

X AXIS TITLE:

Y AXIS TITLE:

BEGIN AT 0: NO

LINE COLORS:

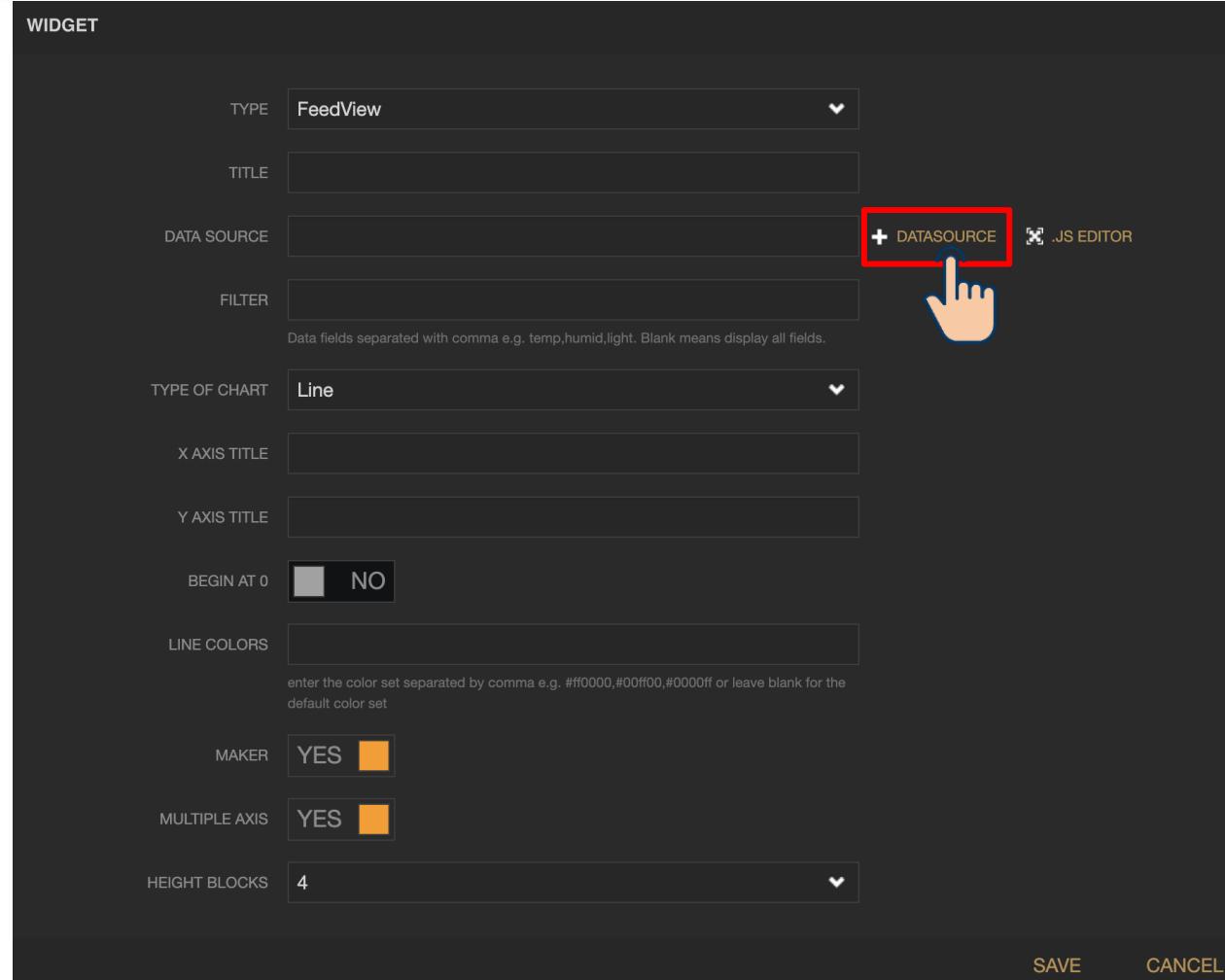
enter the color set separated by comma e.g. #ff0000,#00ff00,#0000ff or leave blank for the default color set

MAKER: YES 

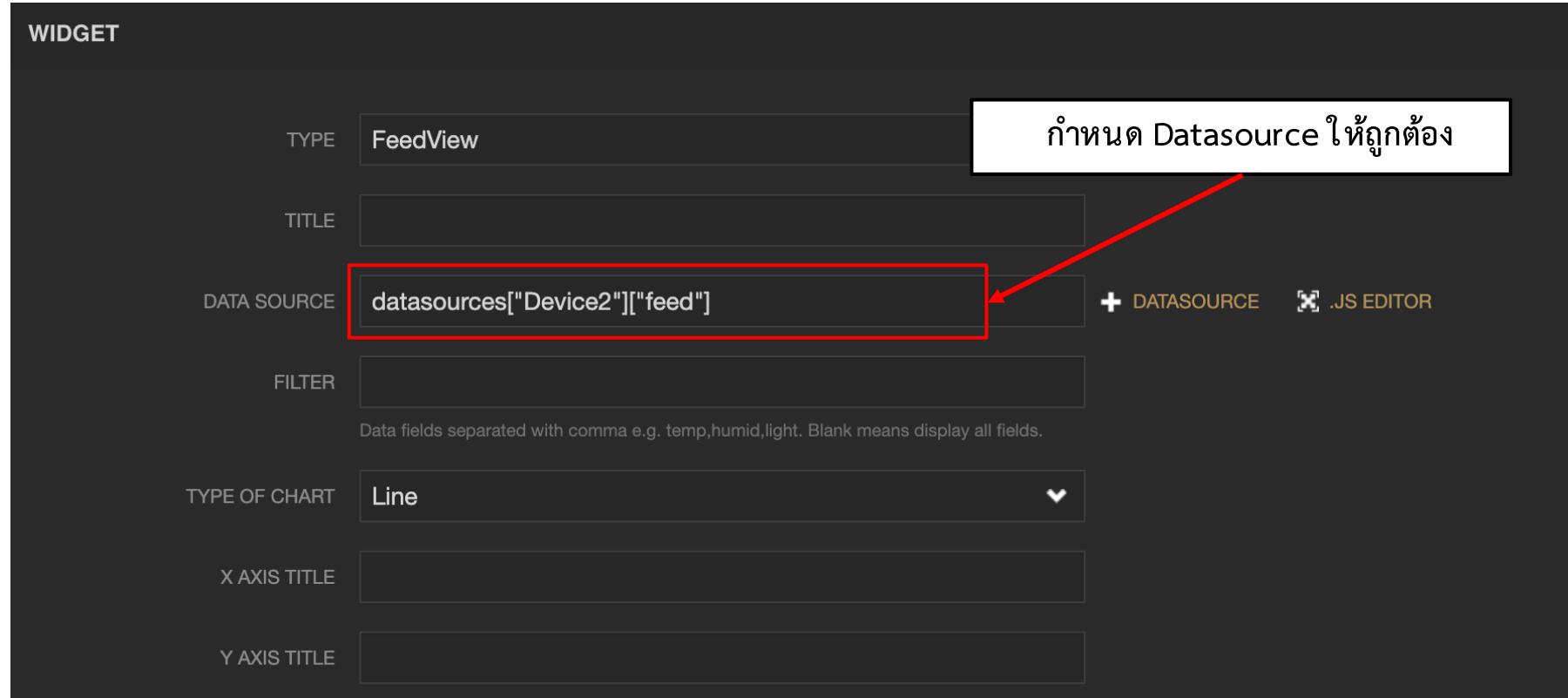
MULTIPLE AXIS: YES 

HEIGHT BLOCKS: 4

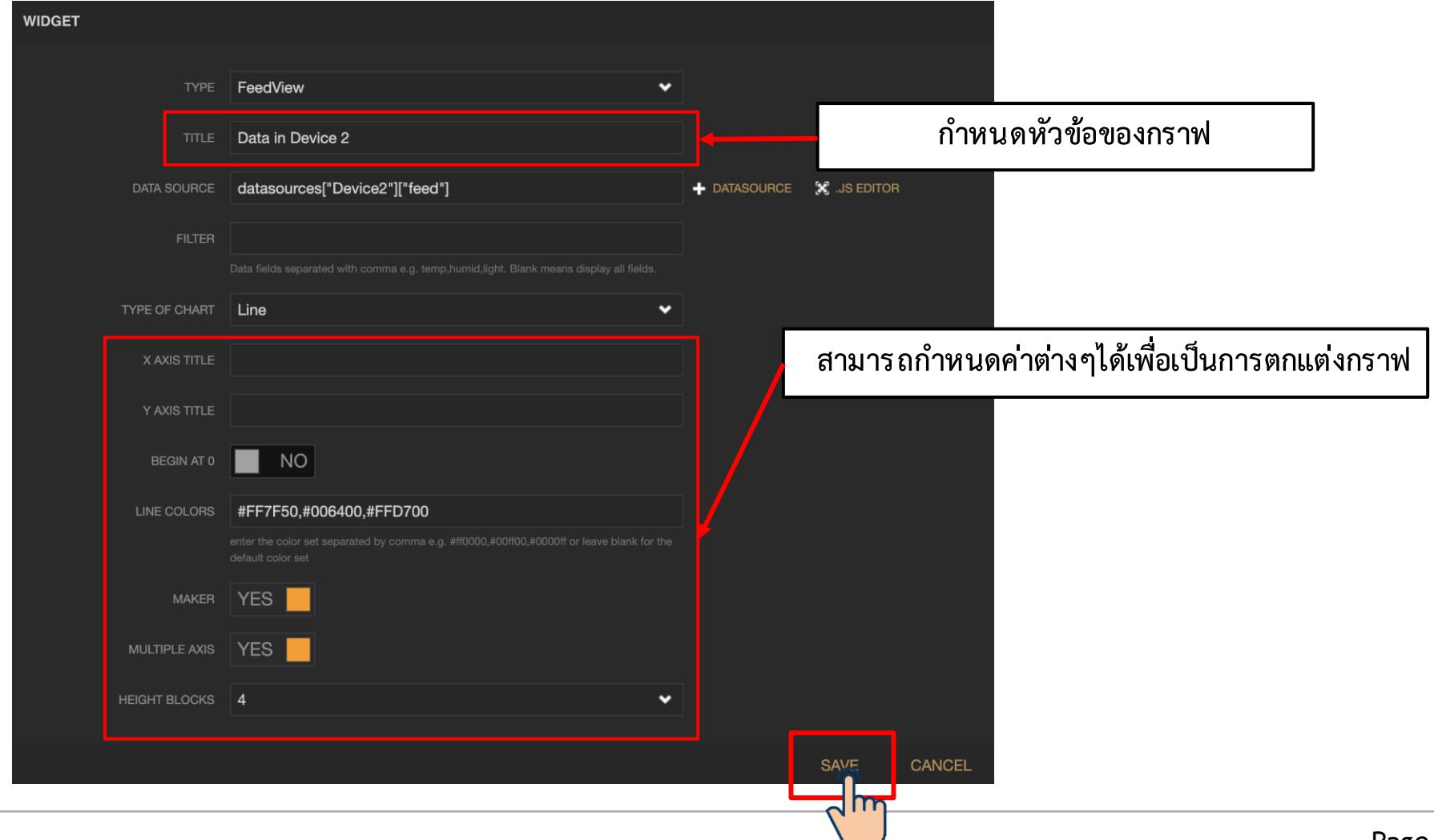
SAVE CANCEL



การสร้าง Widget : FeedView



การสร้าง Widget : FeedView

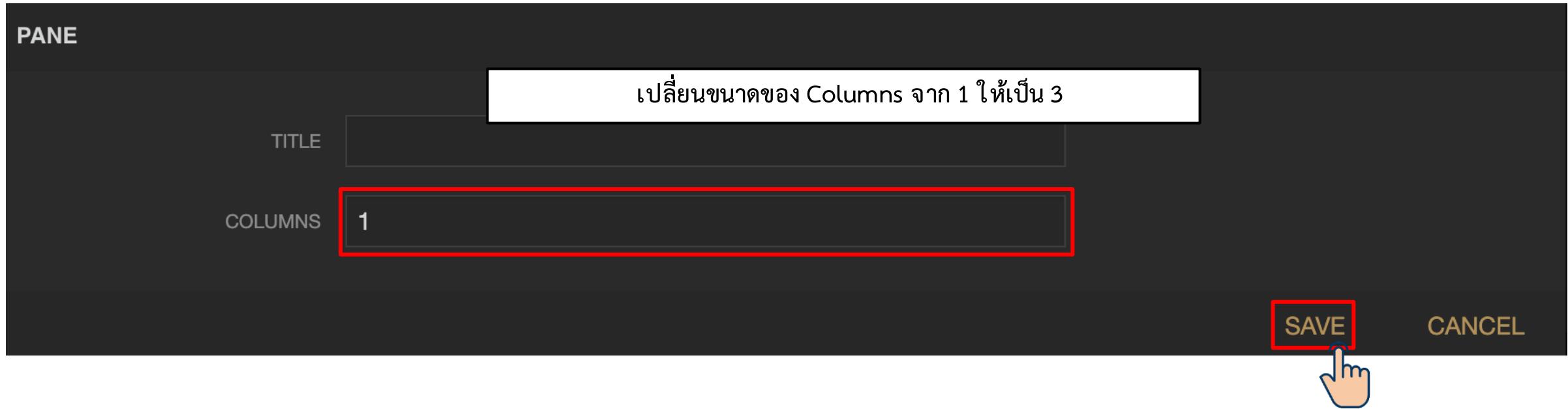


การสร้าง Widget : FeedView

หลังจาก Save แล้วจะได้กราฟมา ซึ่งขนาดของกราฟอาจจะไม่เหมาะสม
ทำการปรับขนาดของกราฟด้วยการเลือกดังรูป

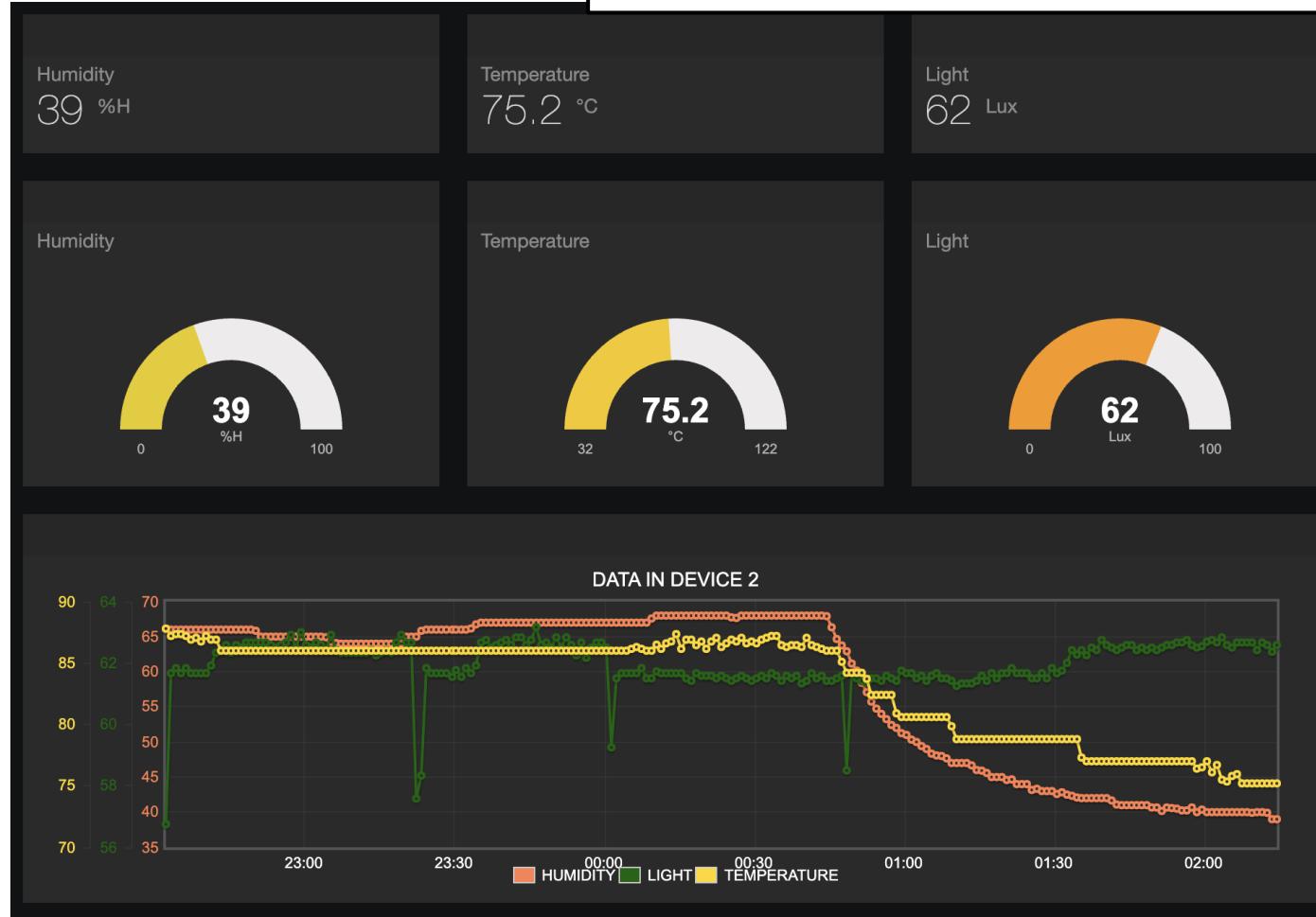


การสร้าง Widget : FeedView



การสร้าง Widget : FeedView

ทำการปรับแต่งหน้าต่างของ Freeboard



การ Save Freeboard



การใช้งาน NETPIE Freeboard รับส่งข้อมูลระหว่าง DEVKIT



หลักการสื่อสารระหว่างอุปกรณ์และ Freeboard



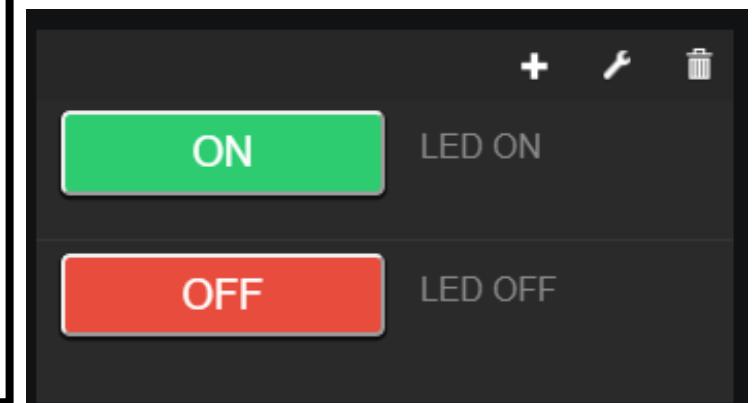
DEVKIT

Topic = @shadow/data/update
Payload = { "data" : { "temperature" : 25, "humidity" : 65, "light" : 120} }
@msg/led : "ON/OFF"



NETPIE

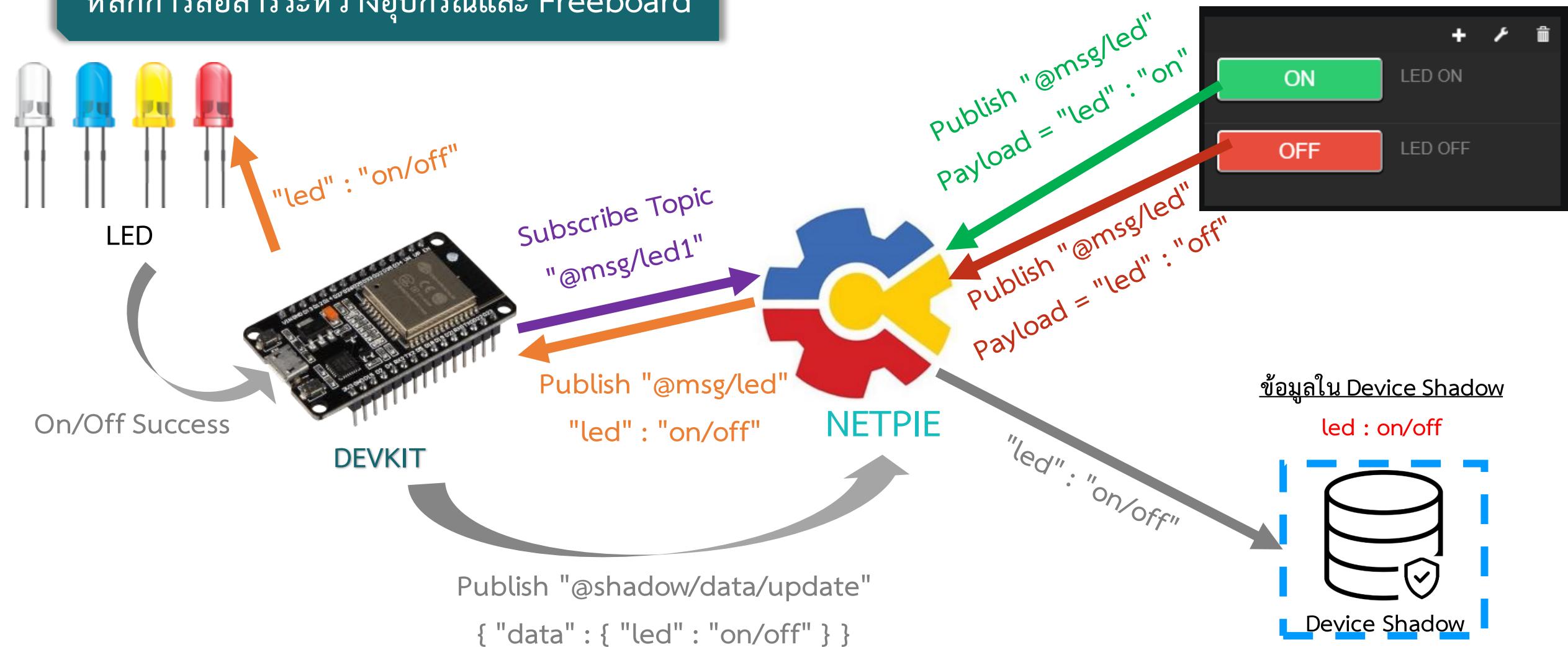
Freeboard



โดยปกติแล้วการส่งข้อมูลจากอุปกรณ์ไปยัง Device บน NETPIE เราต้องใช้ Shadow Topic เพื่อให้ข้อมูลถูกจัดเก็บลง Device Shadow แต่ในกรณีของ Freeboard ไปยัง Device เราจะใช้วิธีการดังนี้

1. Freeboard จะส่งข้อความผ่าน Message Topic ไปยัง DEVKIT เพื่อควบคุมอุปกรณ์
2. เมื่ออุปกรณ์ทำงานตามคำสั่งที่ได้รับเสร็จสิ้น อุปกรณ์จะส่งข้อมูลกลับไปผ่าน Shadow Topic เพื่ออัพเดทสถานะการทำงานของอุปกรณ์ที่ควบคุม ดังรูปหน้าตัดไป

หลักการสื่อสารระหว่างอุปกรณ์และ Freeboard



Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

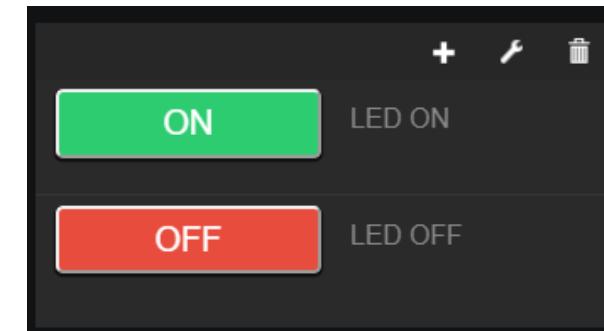
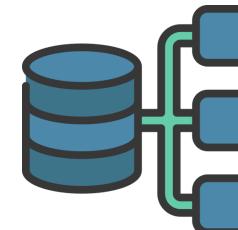
เช็คความพร้อมเพื่อทดสอบทั้งหมด 3 ส่วน

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



DEVKIT

Device Schema



Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

Coding ใน Example22.ino แบ่งออกเป็น 3 ส่วน

1

ส่วนที่ 1 การเรียกใช้ Library และ ประกาศตัวแปร

ประกาศตัวแปรและขาใช้สำหรับ LED

ประกาศตัวแปรและขาใช้สำหรับการทำ millis เนื่องจากมีการรับส่ง
ข้อมูลอยู่ตลอดการใช้ 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);
```

```
char msg[150];
#define DHTPIN 0
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

```
#define LED1 2
```

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

Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

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

ฟังก์ชันการเชื่อมต่อ MQTT Server

คำสั่ง Subscribe Topic
สำหรับรอรับคำสั่งจาก Freeboard

Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

2

ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

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

การเขียนเงื่อนไขเพื่อควบคุมการเปิด/ปิด LED

โดยเริ่มจากการเช็ค Topic ก่อนเสมอ

หลังจากนั้นทำการเช็คข้อมูลที่ได้รับ

เมื่อได้รับข้อมูลตามที่กำหนดจะทำการ Publish ข้อมูลกลับไปที่ NETPIE เพื่อทำการอัพเดทสถานะของ LED

Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

2

ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการตั้งค่าต่างๆ

ประการการใช้งานฟังก์ชัน callback

กำหนดประเภทข้าของ 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(matt_server, matt_port);
    client.setCallback(callback);
    dht.begin();
    pinMode(LED1, OUTPUT);
    digitalWrite(LED1, 1);
}
```

Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

3

ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

ฟังก์ชันการทำงานหลัก

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

    long now = millis();
    if (now - lastMsg > 5000) {

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

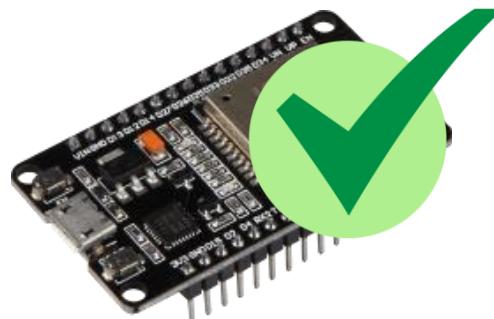
        lastMsg = now;
        String data = "{\"data\":{\"temperature\":" + String(temperature) + ", \"humidity\":" + String(humidity) + "\"}}";
        Serial.println(data);
        data.toCharArray(msg, (data.length() + 1));
        client.publish("@shadow/data/update", msg);
    }
    delay(1);
}
```

ใน void loop นั้นส่วนของการส่งข้อมูลจากเซนเซอร์ต่างๆได้นำ millis มาครอบการทำงานทั้งหมดเพื่อไม่ให้เกิดการหน่วงเวลาการทำงานของโปรแกรม

Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

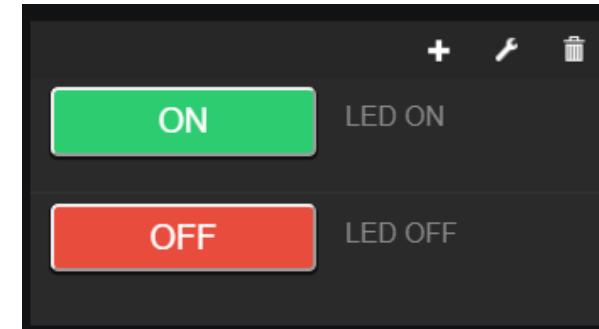
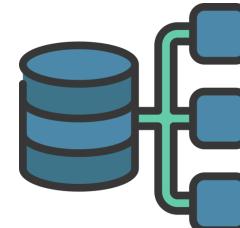
เช็คความพร้อมเพื่อทดสอบทั้งหมด 3 ส่วน

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



DEVKIT

Device Schema



Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

The screenshot shows the Freeboard Schema editor interface. At the top, there are tabs: Shadow, Schema (which is selected and highlighted in blue), Trigger, Feed, and an information icon. On the far right are 'SAVE' and 'Cancel' buttons. Below the tabs is a toolbar with icons for creating nodes like 'object', 'array', 'number', 'string', 'boolean', and 'null'. A search bar is also present. The main area is a tree view labeled 'Tree ▾' with the title 'Select a node...'. The tree structure is as follows:

- object {2}
 - additionalProperties : false
 - properties {2}
 - humidity {2}
 - operation {1}
 - store {1}
 - ttl : 7d
 - type : number
 - temperature {2}
 - operation {2}
 - store {1}
 - ttl : 30d
 - transform {1}
 - expression : (\$.temperature * 1.8) + 32
 - type : number

มาที่ Device Schema

Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

```
{  
    "additionalProperties": false,  
    "properties": {  
        "humidity": {  
            "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

Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

The screenshot shows a tree view of a JSON schema. The schema structure is as follows:

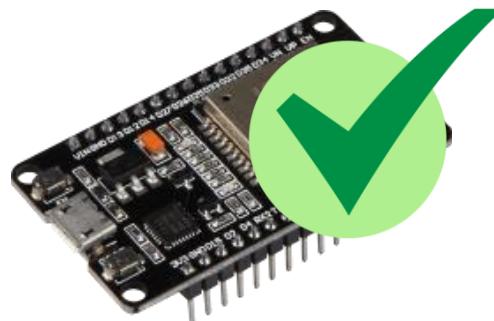
```
object {2}
  additionalProperties : false
  properties {3}
    humidity {2}
      operation {1}
      store {1}
        ttl : 7d
    type : number
    temperature {2}
      operation {2}
      store {1}
        ttl : 30d
      transform {1}
        expression : ($.temperature * 1.8) + 32
    type : number
    led {2}
      operation {1}
      store {1}
        ttl : 7d
    type : string
```

A red box highlights the "led" node under "object". A red arrow points from this box to a callout box containing the text "Schema ที่ถูกเพิ่มเข้ามา" (Schema that was added).

Example 22 : ทดสอบการควบคุม LED ที่ DEVKIT ผ่าน Freeboard

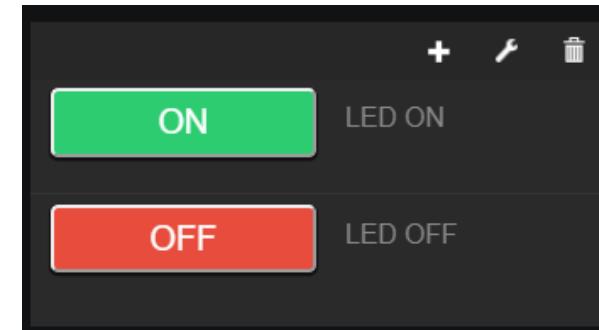
เช็คความพร้อมเพื่อทดสอบทั้งหมด 3 ส่วน

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



DEVKIT

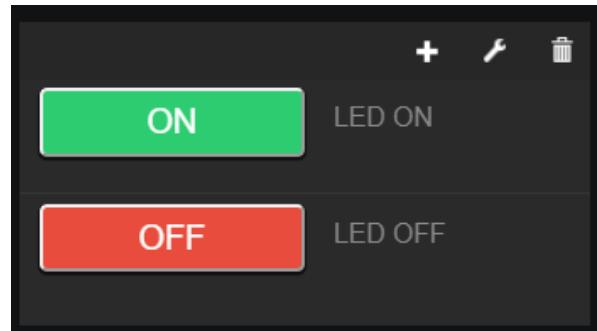
Device Schema



การสร้าง Widget : Button

เป็น Widget สำหรับควบคุมแบบปุ่มกด

ตัวอย่าง Widget : Button



WIDGET

A simple button widget that can perform Javascript action.

TYPE: **Button**

BUTTON CAPTION: **คำหรือประโยคที่อยู่บนปุ่ม**

LABEL TEXT: **คำอธิบายของปุ่ม**

BUTTON COLOR: **สีของปุ่ม**

ONCLICK ACTION: **Add some Javascript here.**

ONCREATED ACTION: **JS code to run after a button is created**

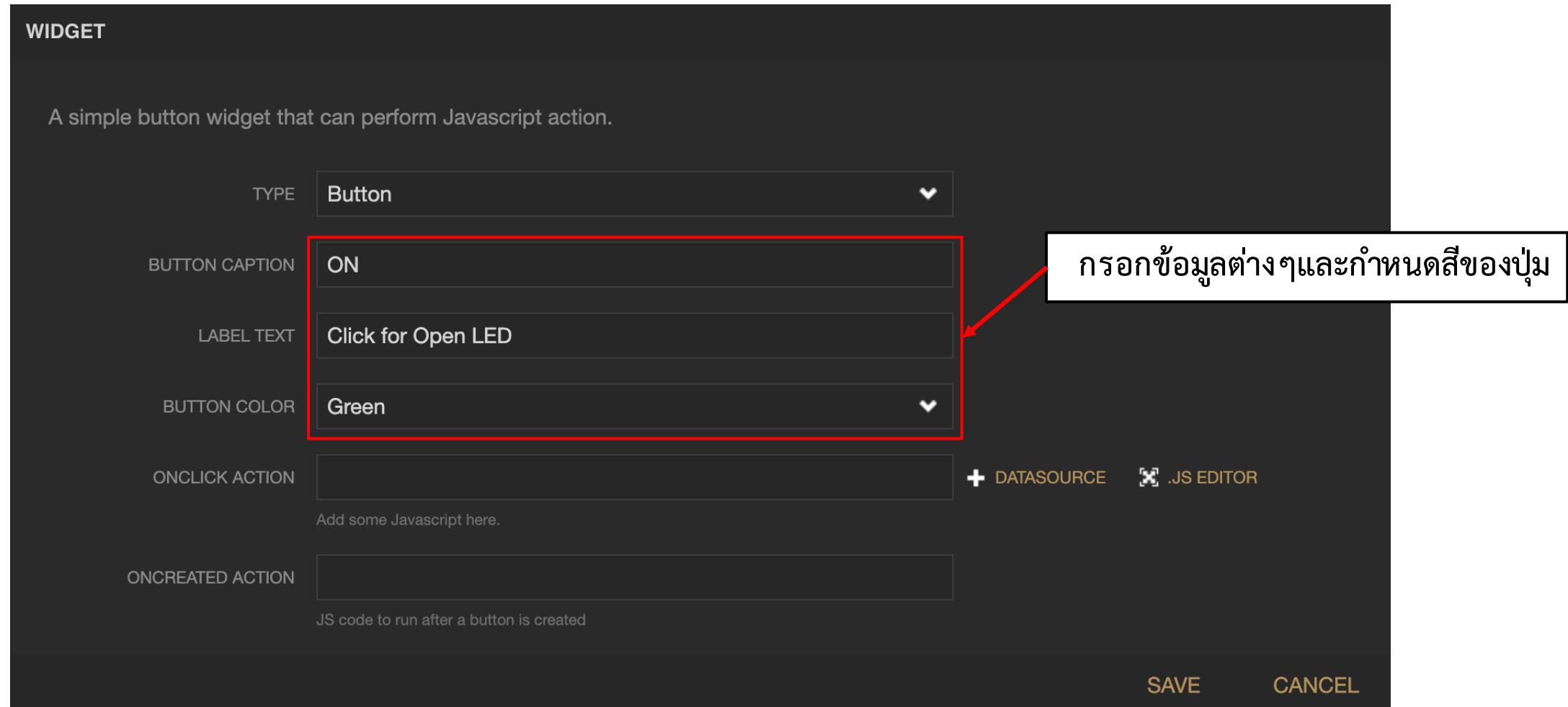
SAVE CANCEL

The screenshot shows the configuration of a 'Button' widget. It highlights several fields with red boxes and arrows pointing to their corresponding labels in Thai:

- BUTTON CAPTION:** คำหรือประโยคที่อยู่บนปุ่ม
- LABEL TEXT:** คำอธิบายของปุ่ม
- BUTTON COLOR:** สีของปุ่ม

The 'ONCLICK ACTION' field contains the placeholder text "Add some Javascript here." and the 'ONCREATED ACTION' field contains the placeholder text "JS code to run after a button is created".

การสร้าง Widget : Button



การสร้าง Widget : Button

WIDGET

A simple button widget that can perform Javascript.

TYPE: Button

BUTTON CAPTION: ON

LABEL TEXT: Click for Open LED

BUTTON COLOR: Green

ONCLICK ACTION: `netpie["Device2"].publish("@msg/led","on")`

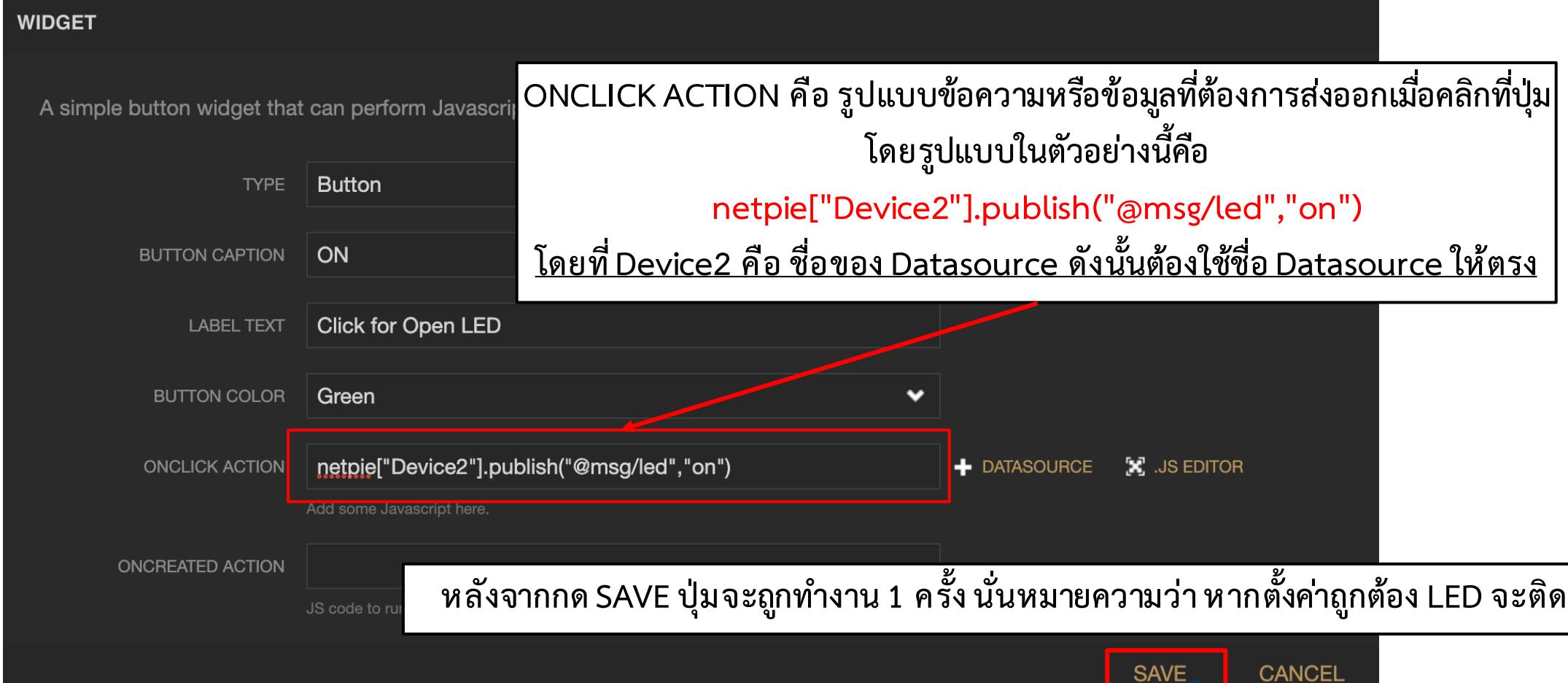
ONCREATED ACTION: `JS code to run`

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

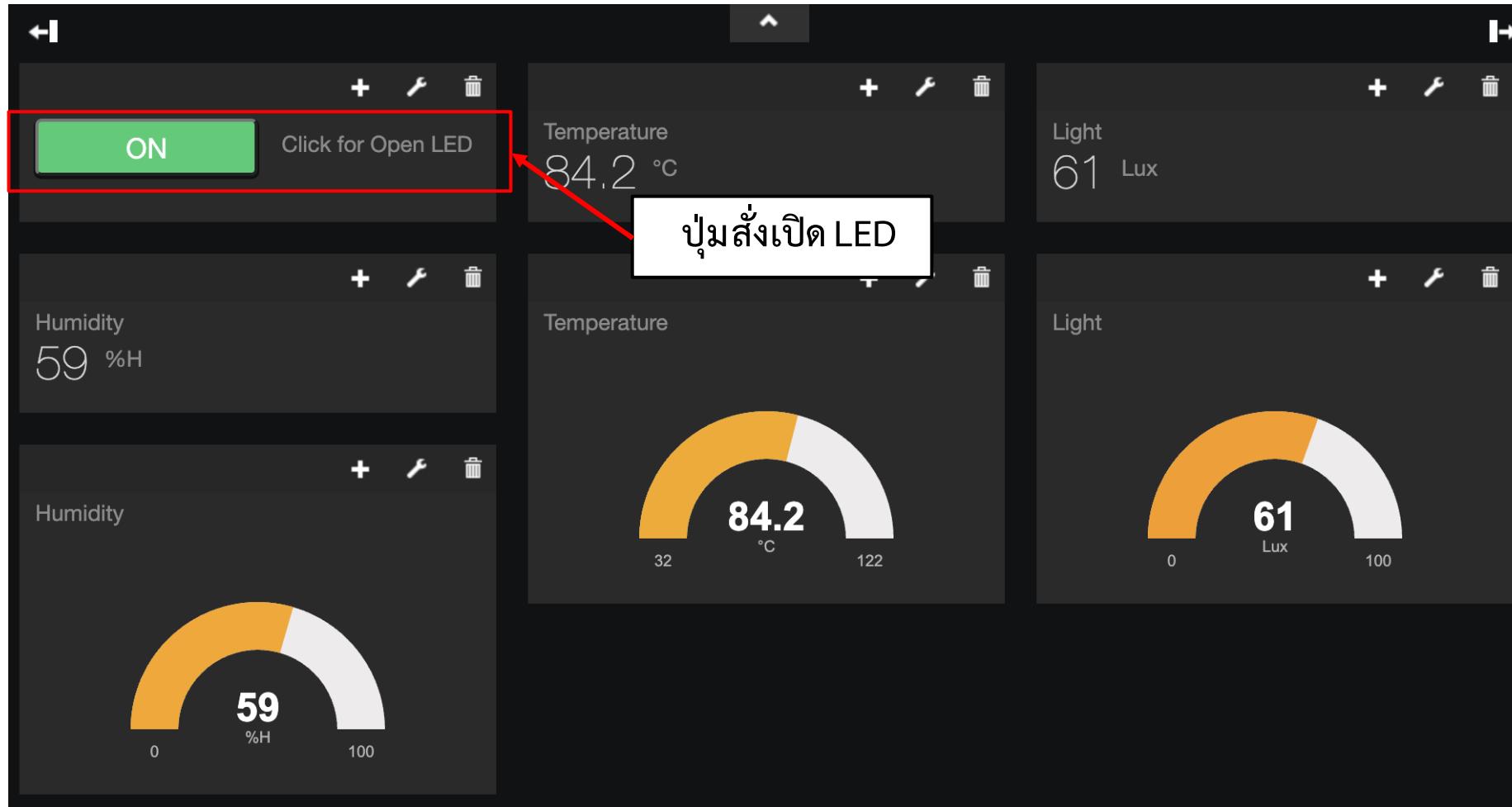
โดยรูปแบบในตัวอย่างนี้คือ
`netpie["Device2"].publish("@msg/led","on")`

หลังจากกด SAVE ปุ่มจะถูกทำงาน 1 ครั้ง นั่นหมายความว่า หากตั้งค่าถูกต้อง LED จะติด

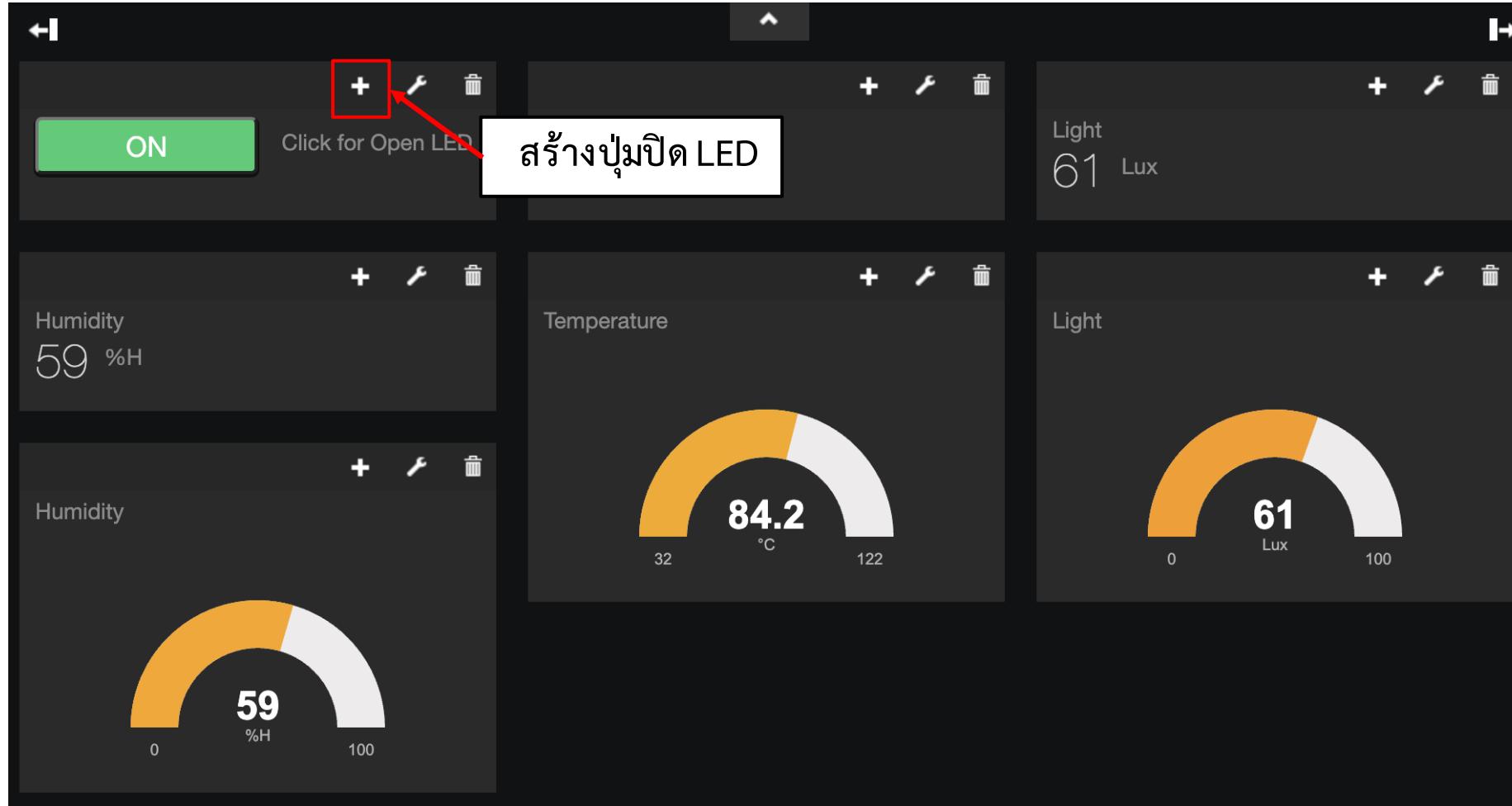
SAVE **CANCEL**



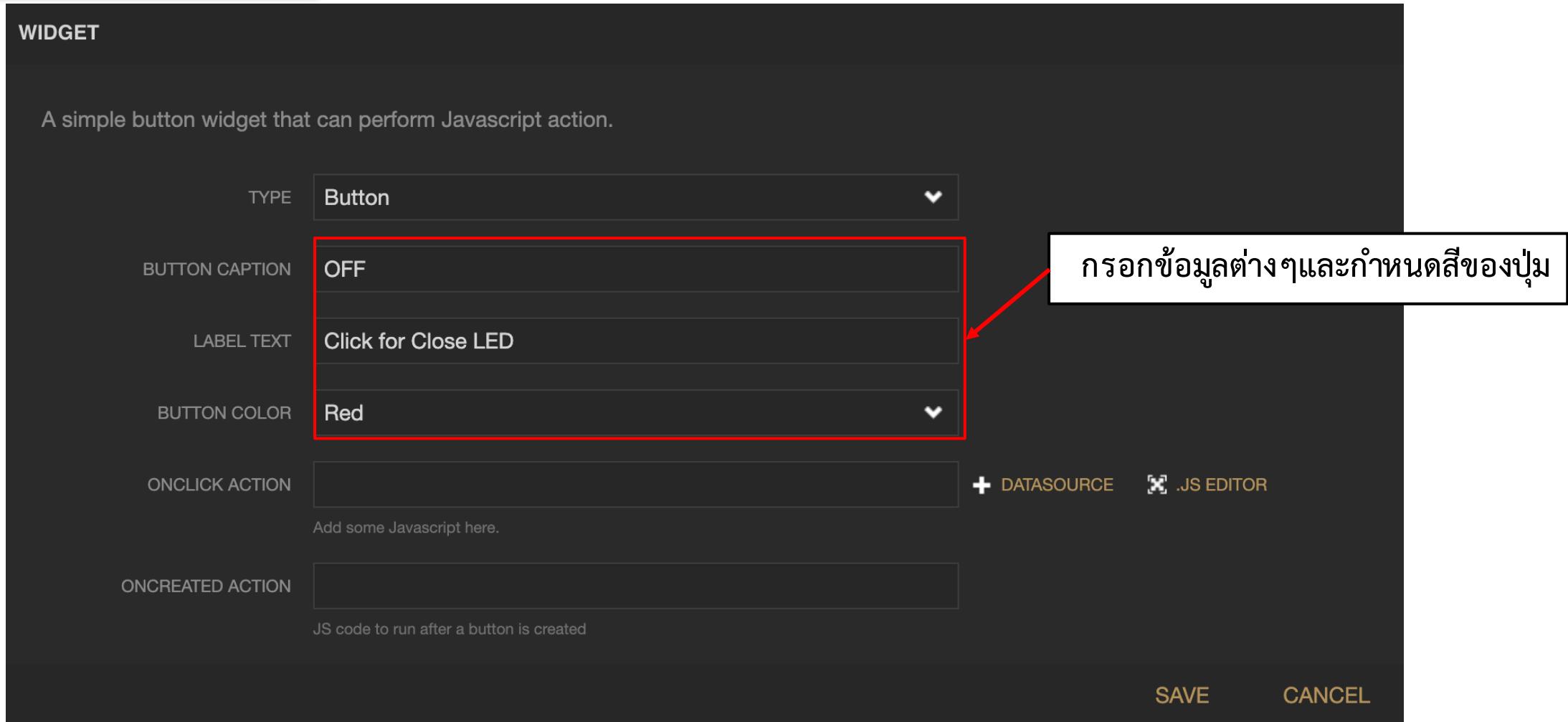
การสร้าง Widget : Button



การสร้าง Widget : Button



การสร้าง Widget : Button



การสร้าง Widget : Button

WIDGET

A simple button widget that can perform Javascript.

TYPE: Button

BUTTON CAPTION: OFF

LABEL TEXT: Click for Close LED

BUTTON COLOR: Red

ONCLICK ACTION: `netpie["Device2"].publish("@msg/led","off")`

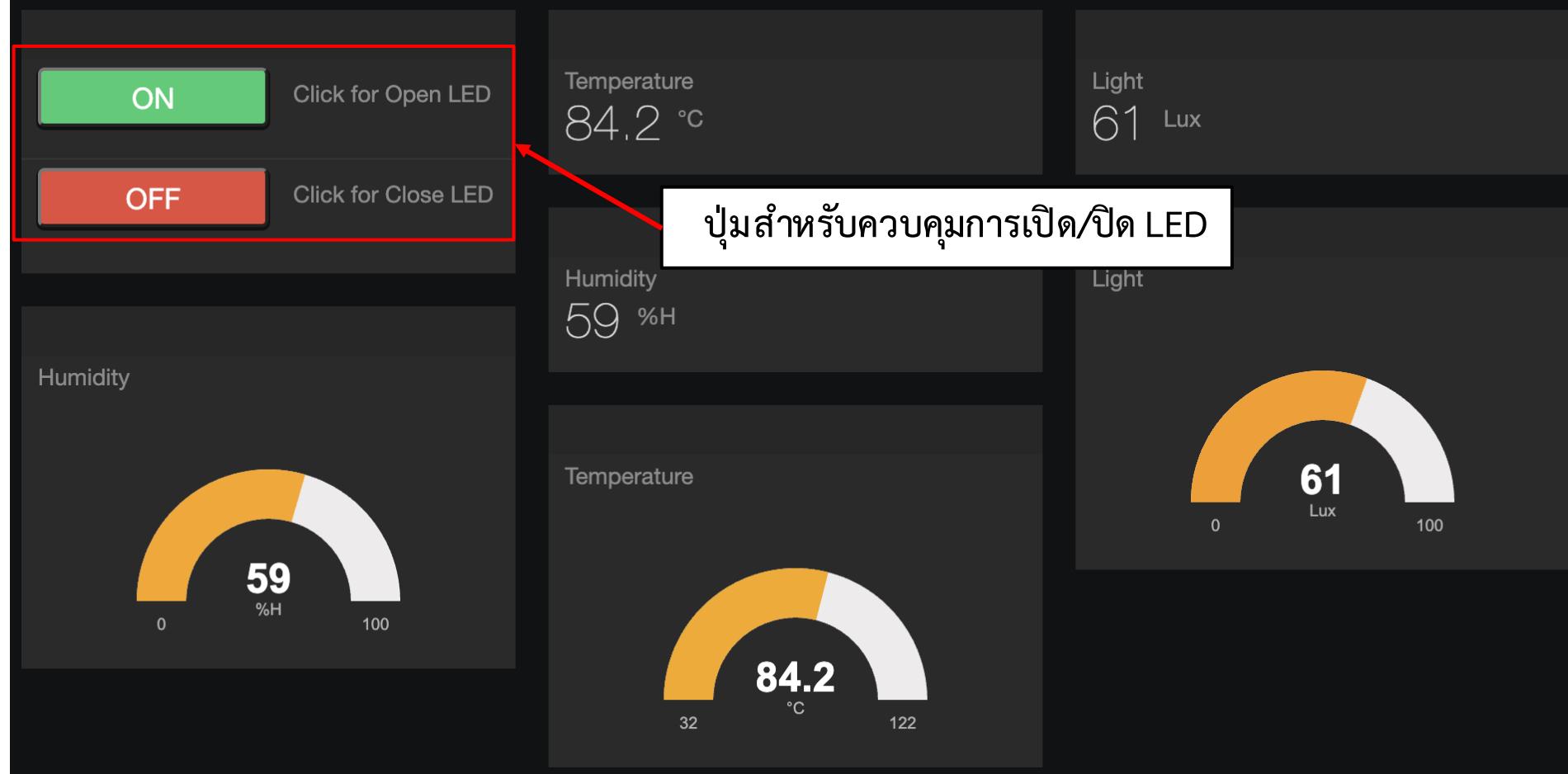
ONCREATED ACTION: `JS code to run when the button is created.`

DATA SOURCE: `.JS EDITOR`

หลังจากกด SAVE ปุ่มจะถูกทำงาน 1 ครั้ง นั่นหมายความว่า หากตั้งค่าถูกต้อง LED จะดับ

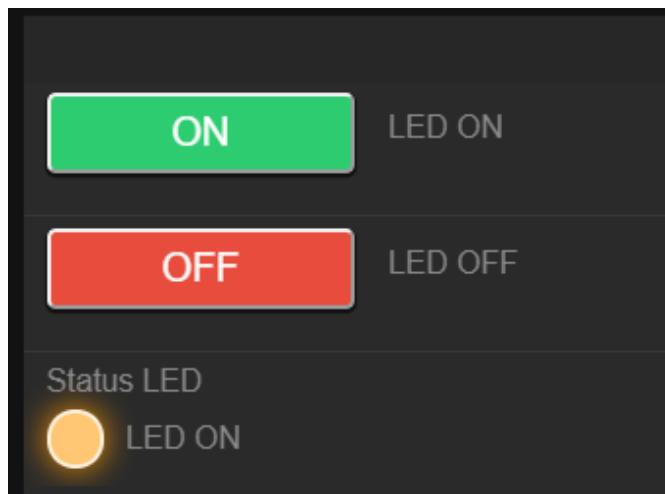
SAVE CANCEL

การสร้าง Widget : Button



การสร้าง Widget : Indicator

ตัวอย่าง Widget : Indicator



เป็น Widget สำหรับแสดงผลสถานะการทำงาน

WIDGET

TYPE: Indicator Light

TITLE:

DEFAULT COLOR: enter the color e.g. #ff0000,red or leave blank for the default color set

VALUE:

ON TEXT:

OFF TEXT:

+ DATASOURCE JS EDITOR

+ DATASOURCE IO EDITOR

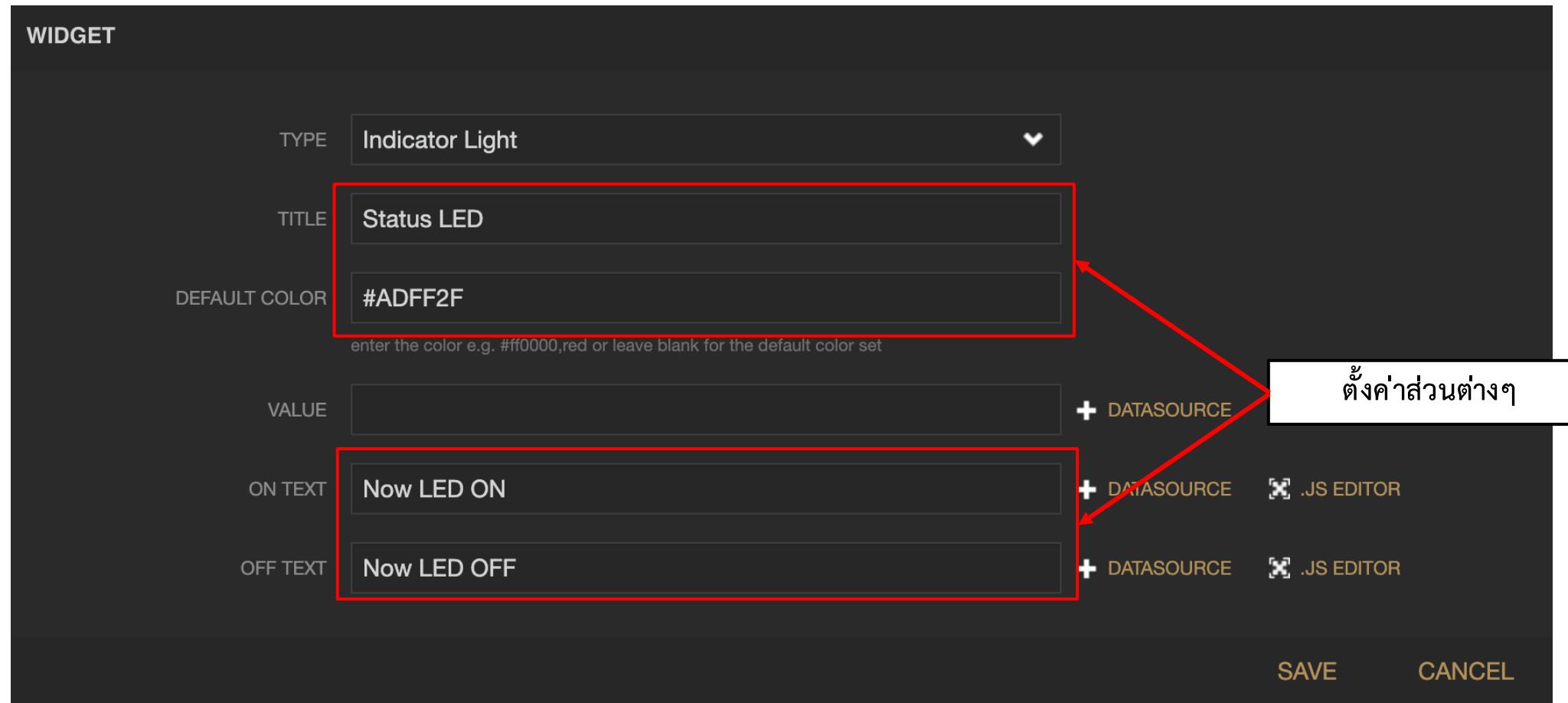
+ DA

SAVE CANCEL

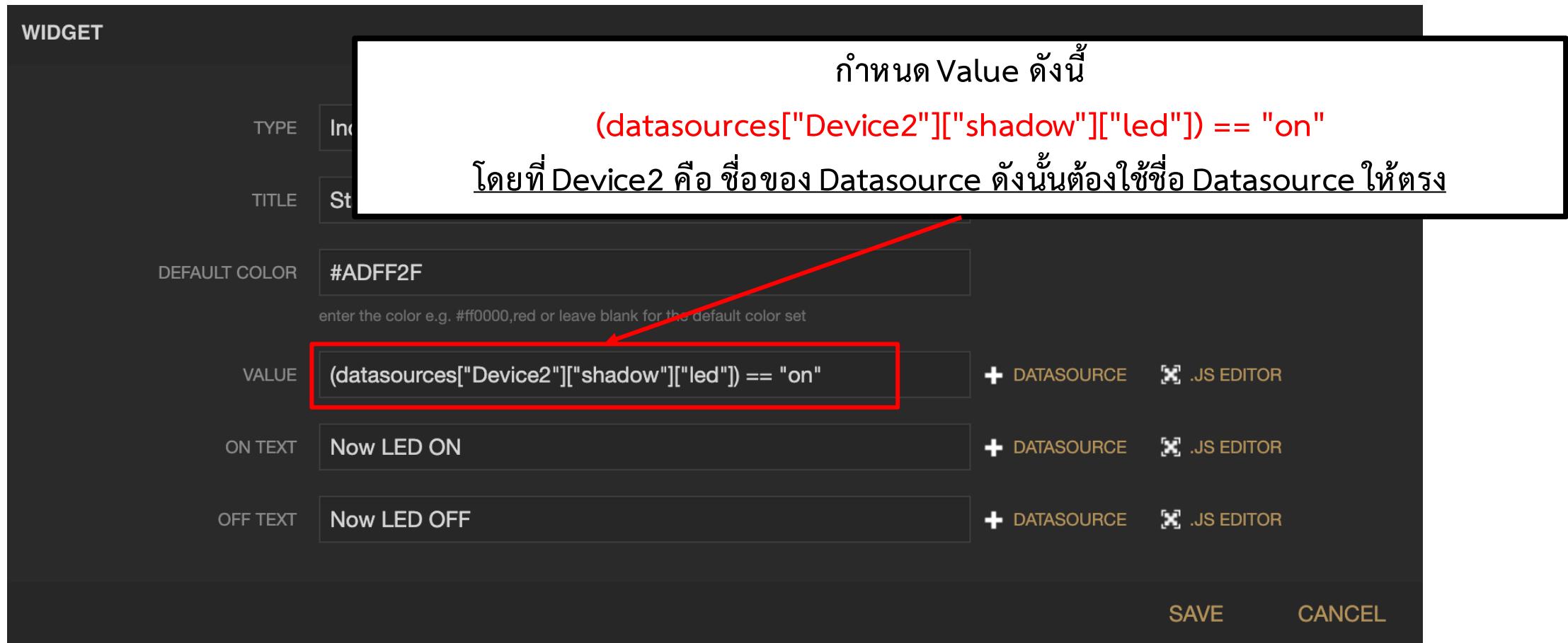
คำจำกัดหรือข้อความที่ใช้บอกข้อมูล
สีของ Indicator
คำอธิบายสถานะต่างๆ
ของ Indicator

The configuration interface for the 'Indicator Light' widget. It includes fields for 'TITLE', 'DEFAULT COLOR', 'ON TEXT', and 'OFF TEXT'. Buttons for 'DATASOURCE' and 'EDITOR' are shown next to each field. Callout boxes explain the purpose of the highlighted fields: 'Title/Message' for the title and color, and 'Description of different states of Indicator' for the text fields.

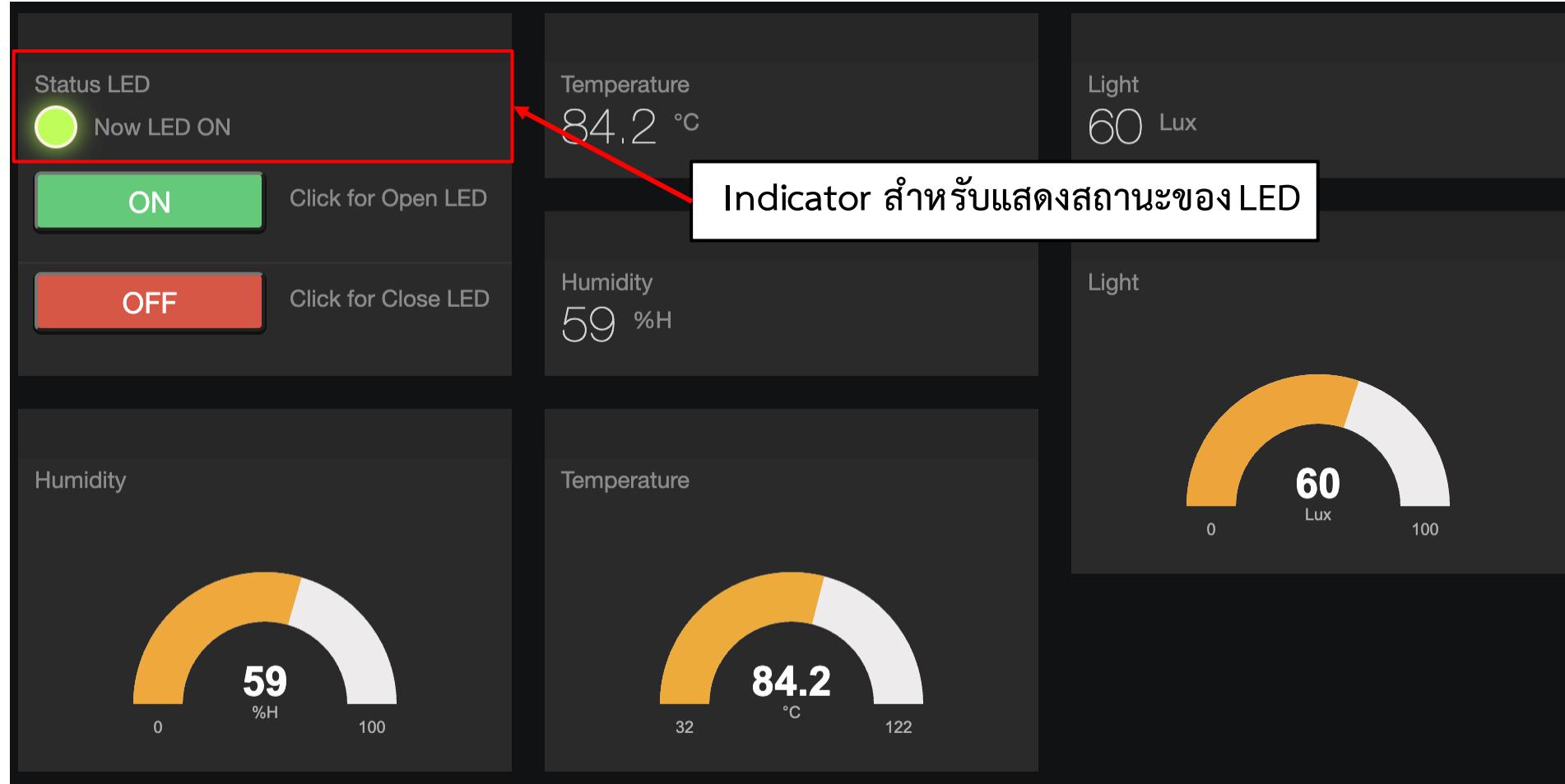
การสร้าง Widget : Indicator



การสร้าง Widget : Indicator



การสร้าง Widget : Indicator



การสร้าง Widget : Toggle

ตัวอย่าง Widget : Toggle

เป็น Widget สำหรับควบคุมและแสดงผลสถานะการทำงาน

WIDGET

A simple toggle widget that can perform Javascript action.

TYPE: Toggle

TOGGLE CAPTION:

TOGGLE STATE: Add a condition to switch a toggle state here. Otherwise it just toggle by click.

ON TEXT: ON

OFF TEXT: OFF

ONTOGGLEON ACTION: JS code to run when a toggle is switched to ON

ONTOGGLEOFF ACTION: JS code to run when a toggle is switched to OFF

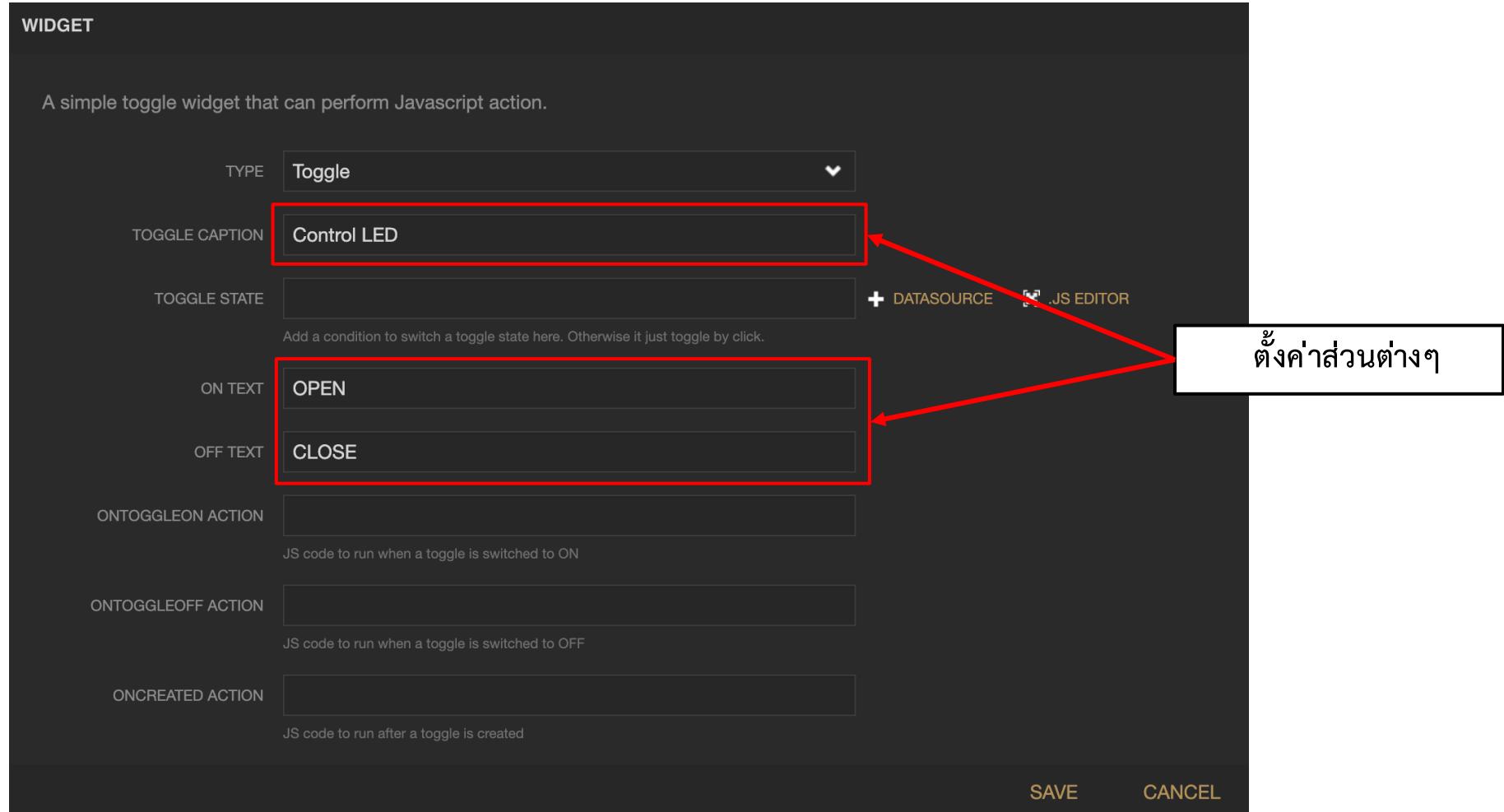
ONCREATED ACTION: JS code to run after a toggle is created

SAVE CANCEL

คำอธิบายของปุ่ม

คำอธิบายบนปุ่มในแต่ละสถานะ

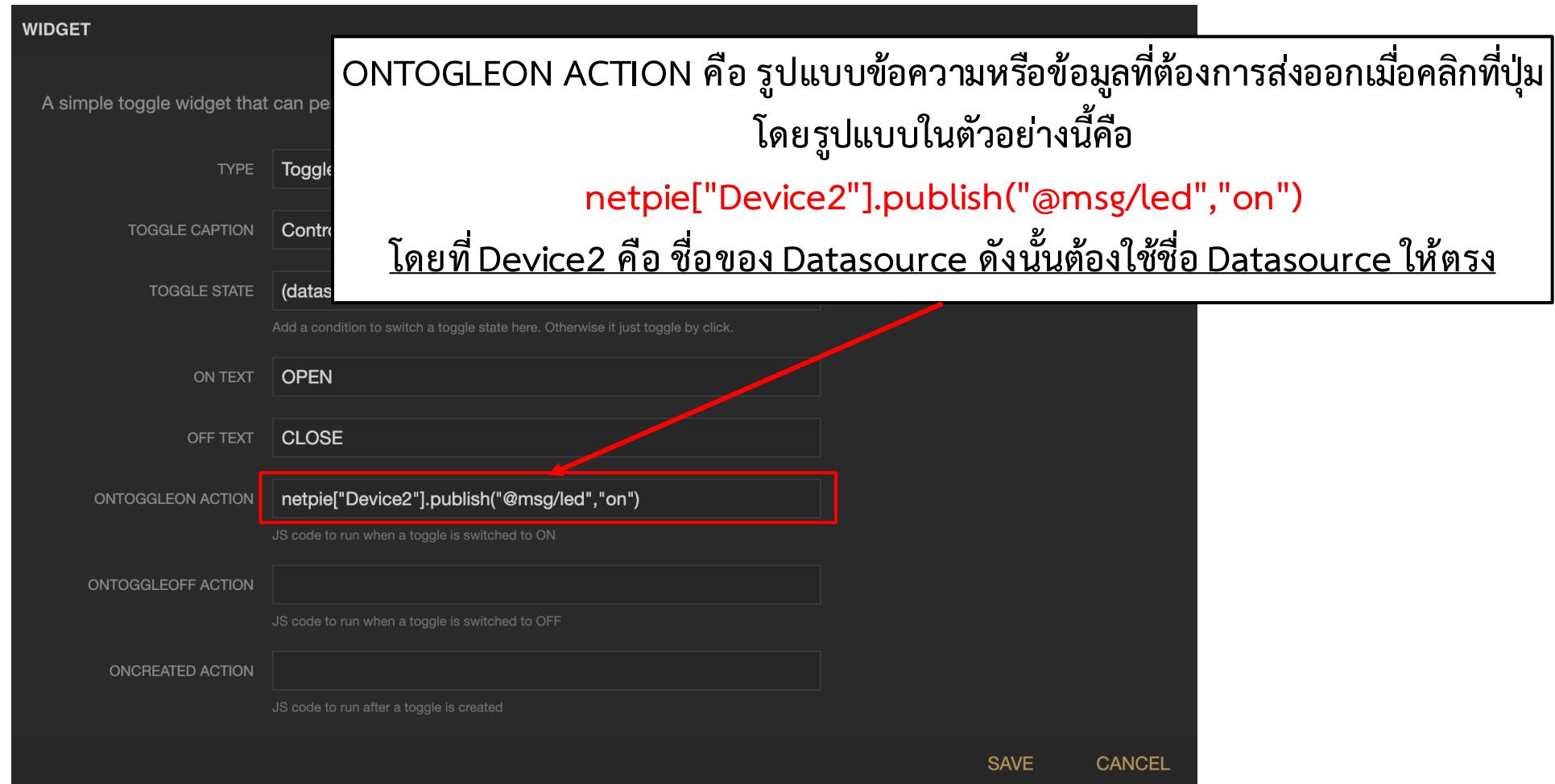
การสร้าง Widget : Toggle



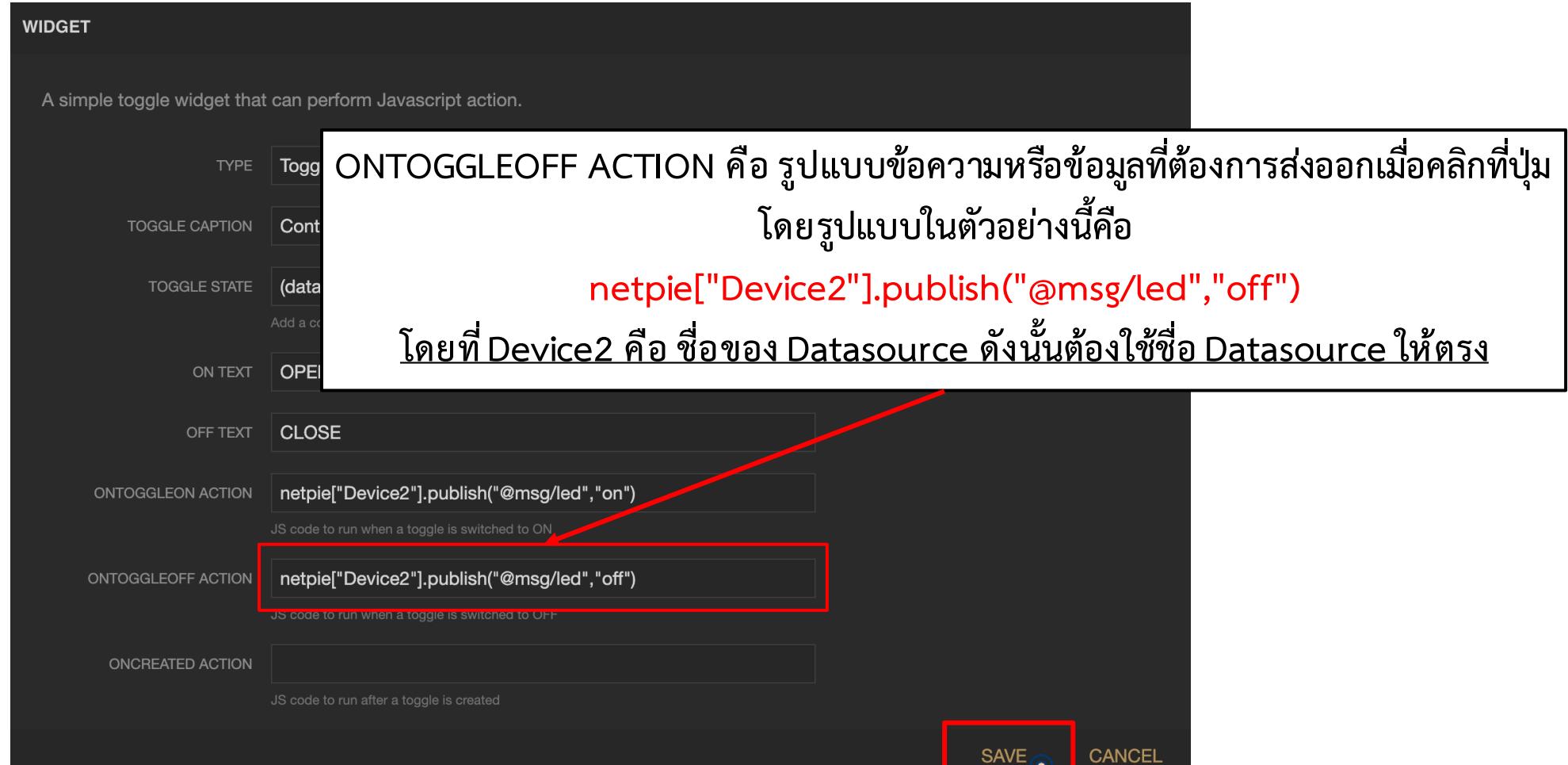
การสร้าง Widget : Toggle



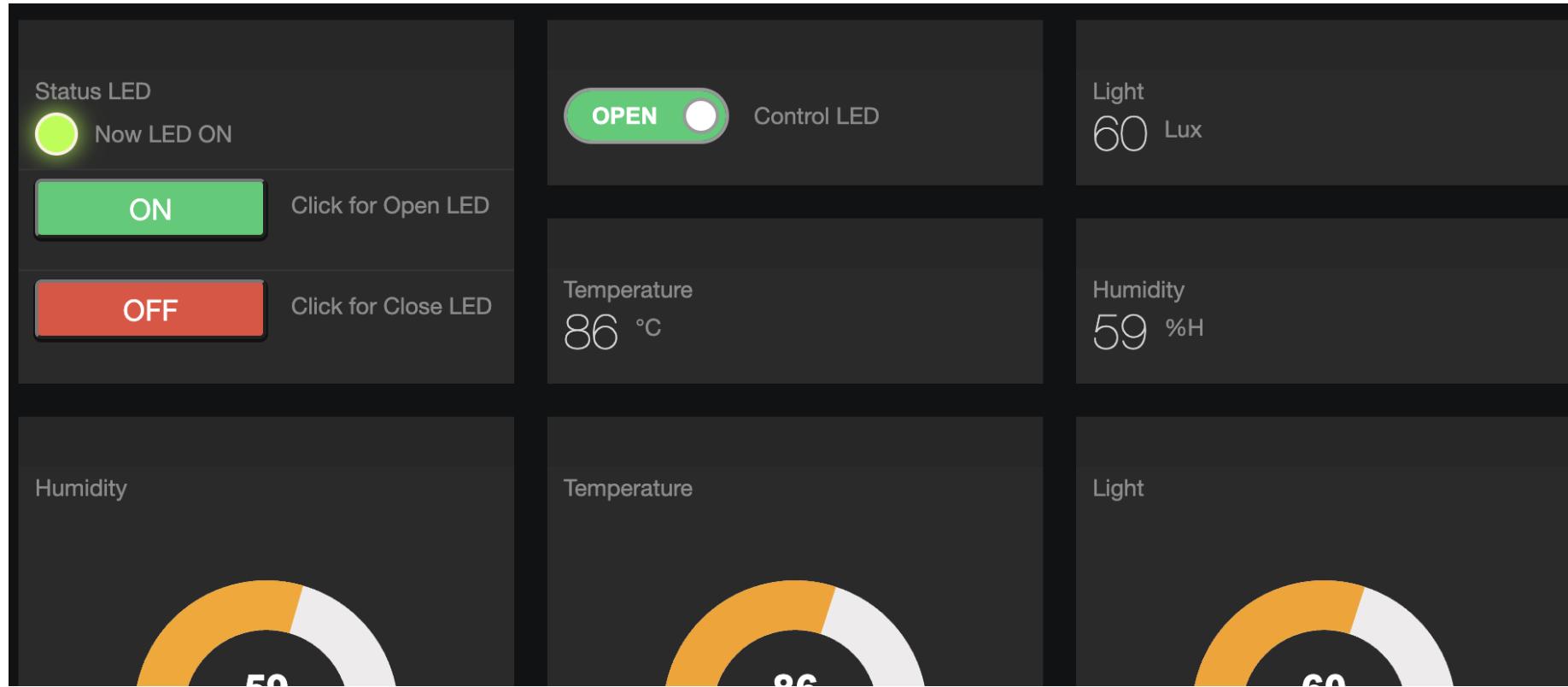
การสร้าง Widget : Toggle



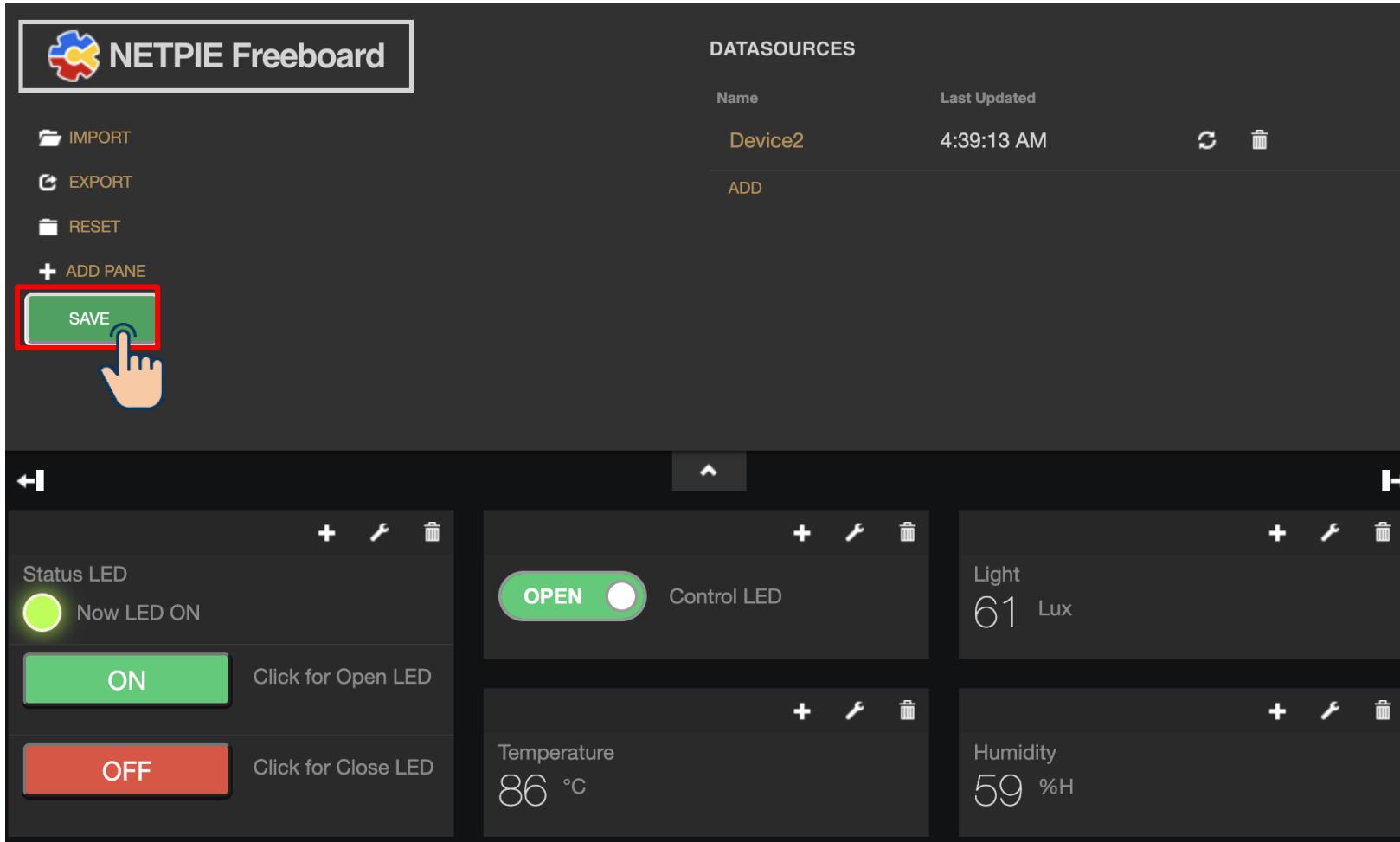
การสร้าง Widget : Toggle



การสร้าง Widget : Toggle



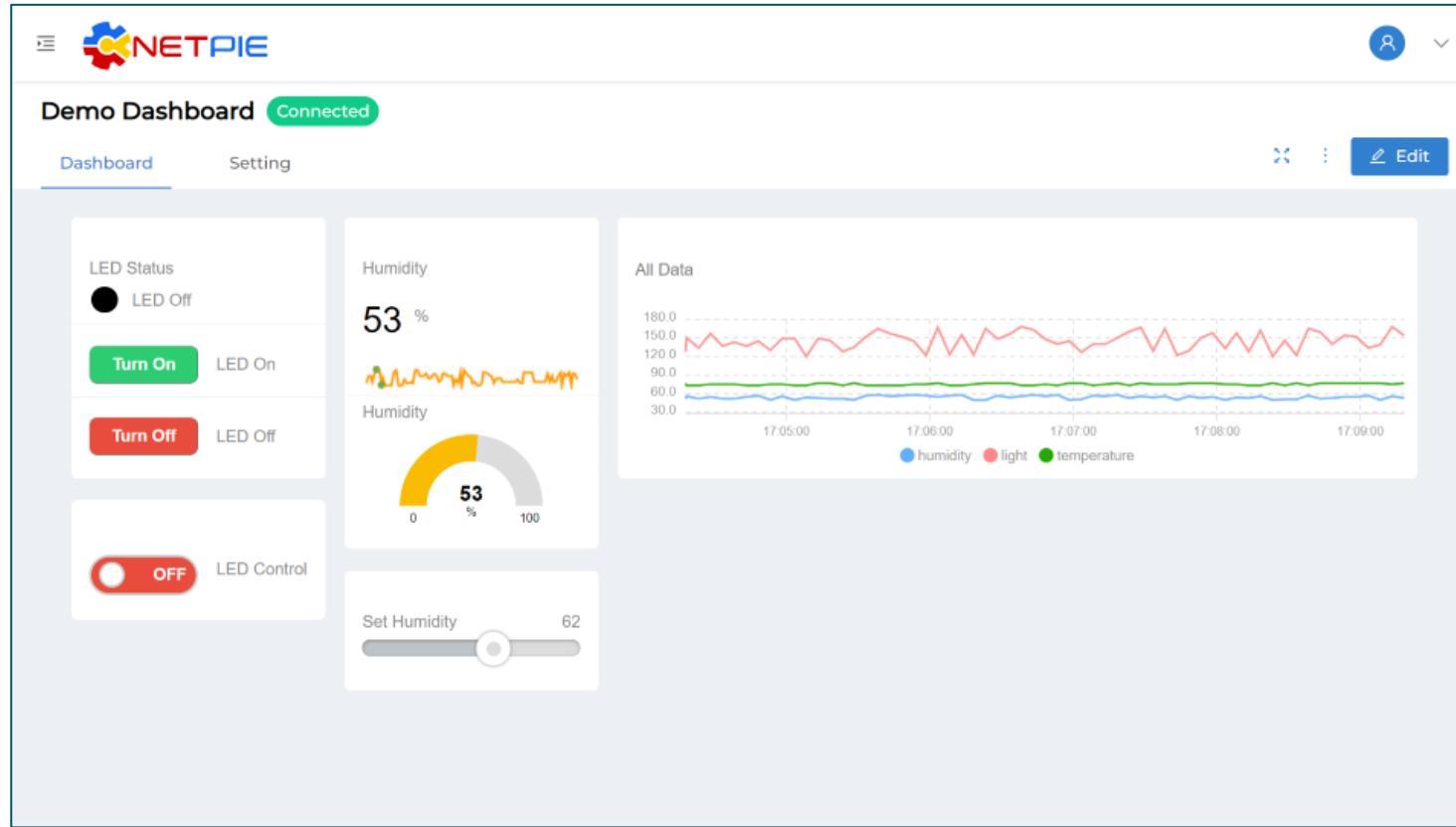
ການ Save Freeboard



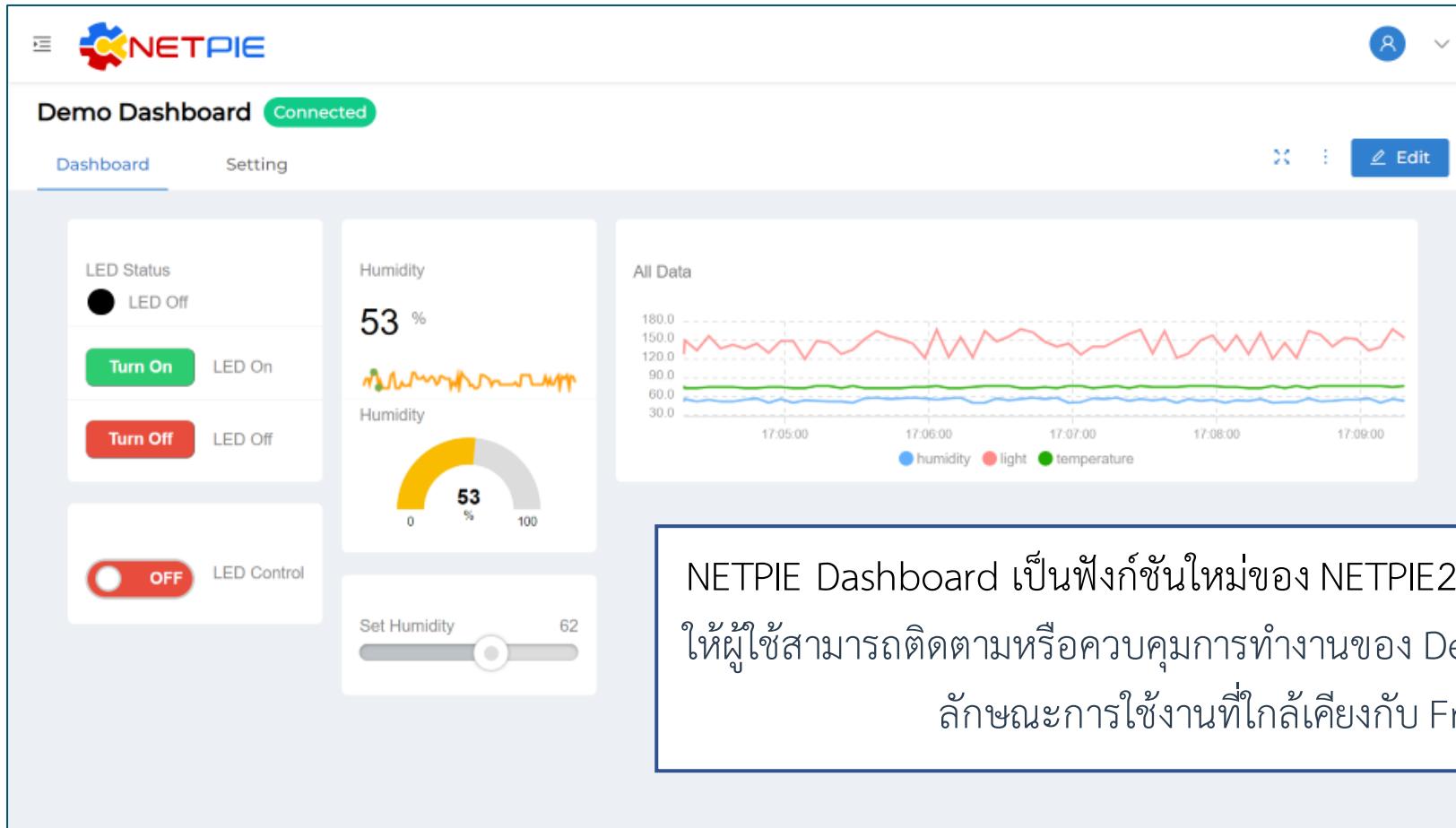
การใช้สร้างและใช้งาน Freeboard ในการอบรมครั้งนี้



การใช้งาน NETPIE Dashboard



NETPIE Dashboard คืออะไร



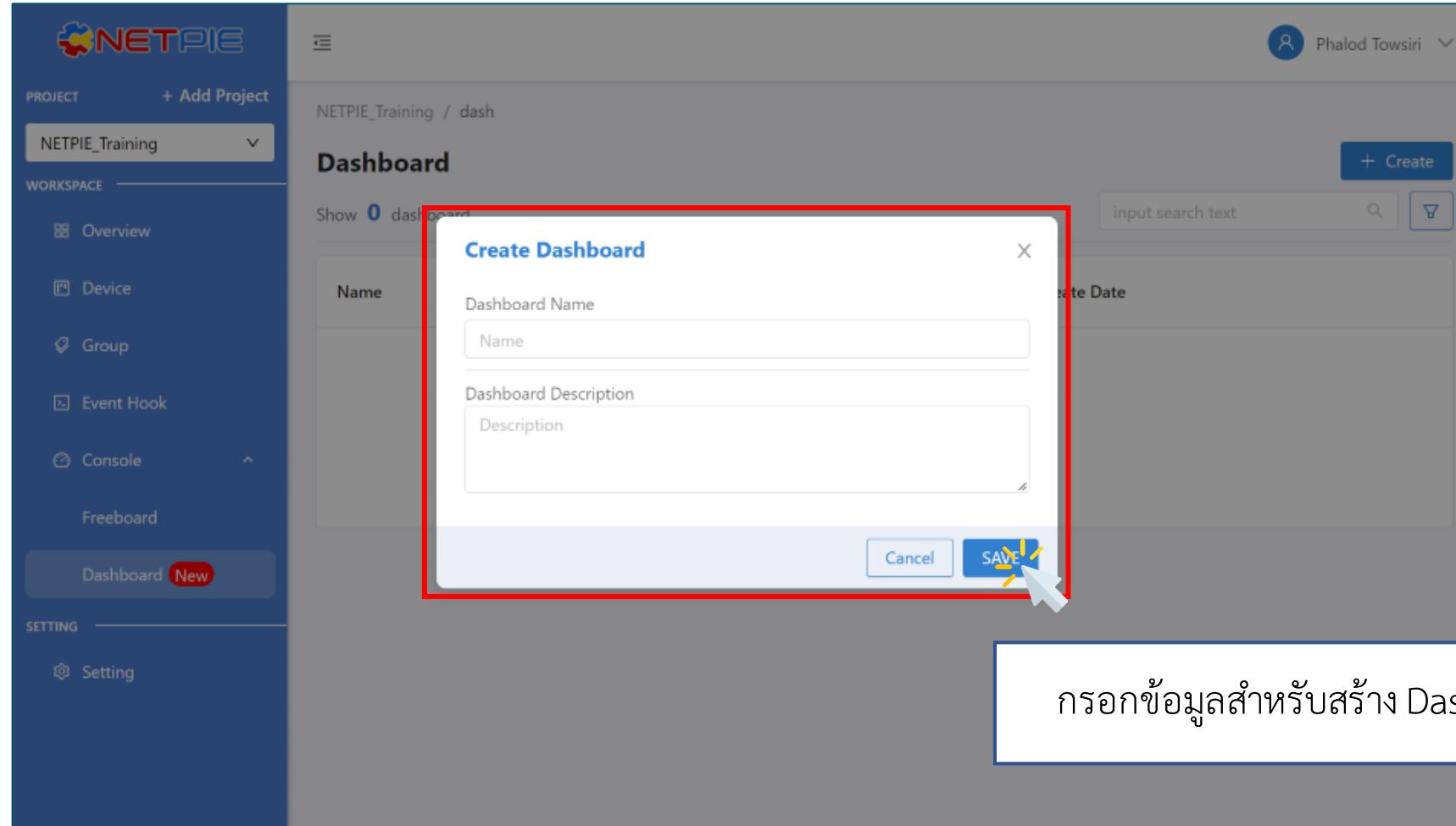
NETPIE Dashboard เป็นฟังก์ชันใหม่ของ NETPIE2020 ที่ เป็นเหมือนช่องทางให้ผู้ใช้สามารถติดตามหรือควบคุมการทำงานของ Device ที่ตัวเองจัดการ โดยมีลักษณะการใช้งานที่ใกล้เคียงกับ Freeboard

การสร้าง NETPIE Dashboard

The screenshot shows the NETPIE web interface. On the left, there's a sidebar with 'PROJECT' and 'WORKSPACE' sections. Under 'WORKSPACE', 'Dashboard' is highlighted with a red box and a cursor icon. In the main area, it says 'NETPIE_Training / dash'. Below that is a 'Dashboard' section with a table header: 'Name', 'Device', and 'Create Date'. A message 'No Data' is displayed. At the top right of the main area, there's a blue button labeled '+ Create' with a yellow cursor icon pointing at it, also enclosed in a red box.

เริ่มการใช้งาน NETPIE Dashboard โดยกดไปที่แท็บ Dashboard ทางด้านซ้ายมือ จะแสดงหน้าต่างดังรูปแล้วกด Create เพื่อสร้าง Dashboard

การสร้าง NETPIE Dashboard



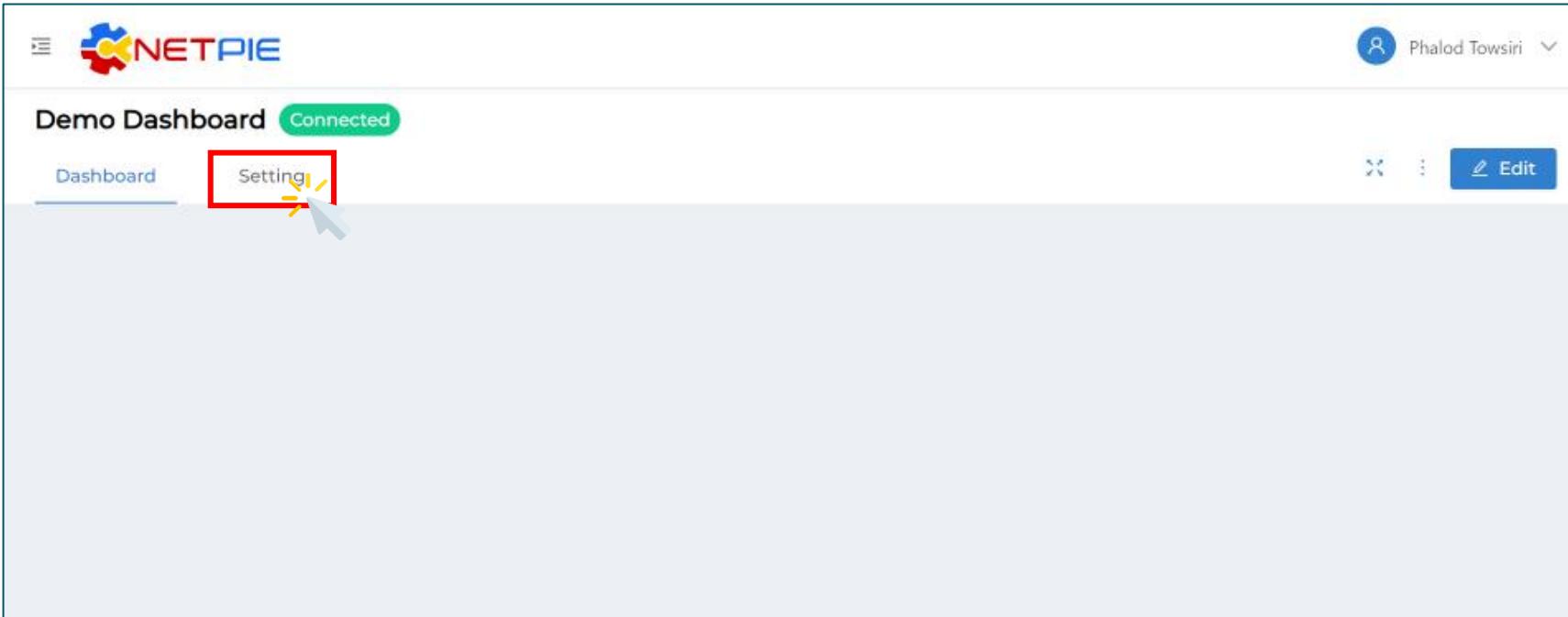
การสร้าง NETPIE Dashboard

The screenshot shows the NETPIE Dashboard interface. On the left, there's a sidebar with 'PROJECT' and 'WORKSPACE' sections. Under 'WORKSPACE', 'Overview' is selected. Below it are 'Device', 'Group', 'Event Hook', 'Console', and 'Freeboard'. Under 'Freeboard', 'Dashboard' is highlighted with a red 'New' badge. The main area is titled 'Dashboard' and shows 'NETPIE_Training / dash'. It displays a table with one row: 'Name' (Demo Dashboard), 'Device' (empty), and 'Create Date' (2023-01-13 13:23). A cursor is hovering over the 'Demo Dashboard' row. A callout box with a blue border and yellow arrow points to the 'Demo Dashboard' row, containing the text: 'เมื่อสร้างสำเร็จ ให้กดเข้าไป เพื่อจัดการ Dashboard'.

Name	Device	Create Date
Demo Dashboard	-	2023-01-13 13:23

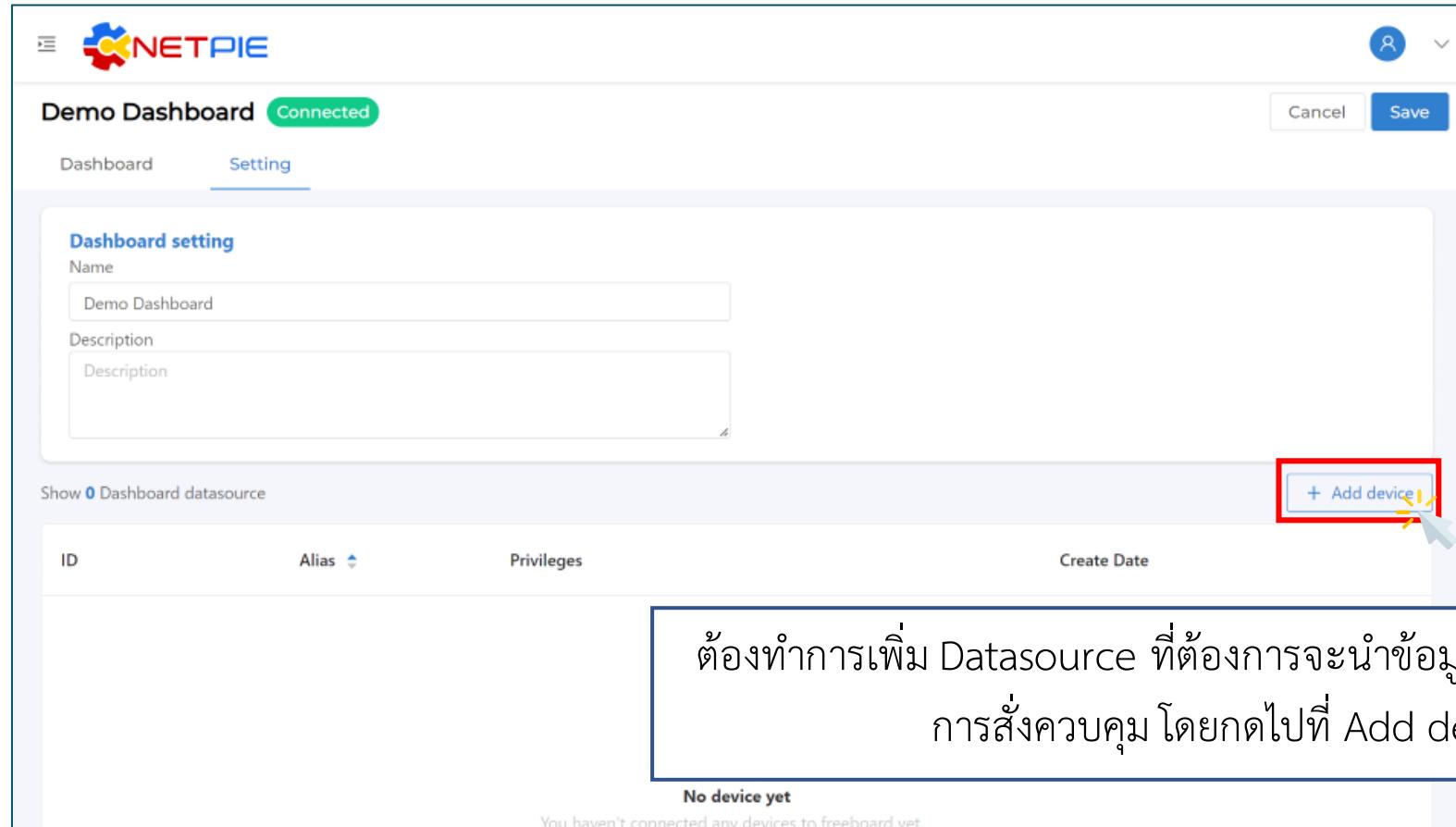
เมื่อสร้างสำเร็จ ให้กดเข้าไป เพื่อจัดการ Dashboard

การสร้าง Datasource

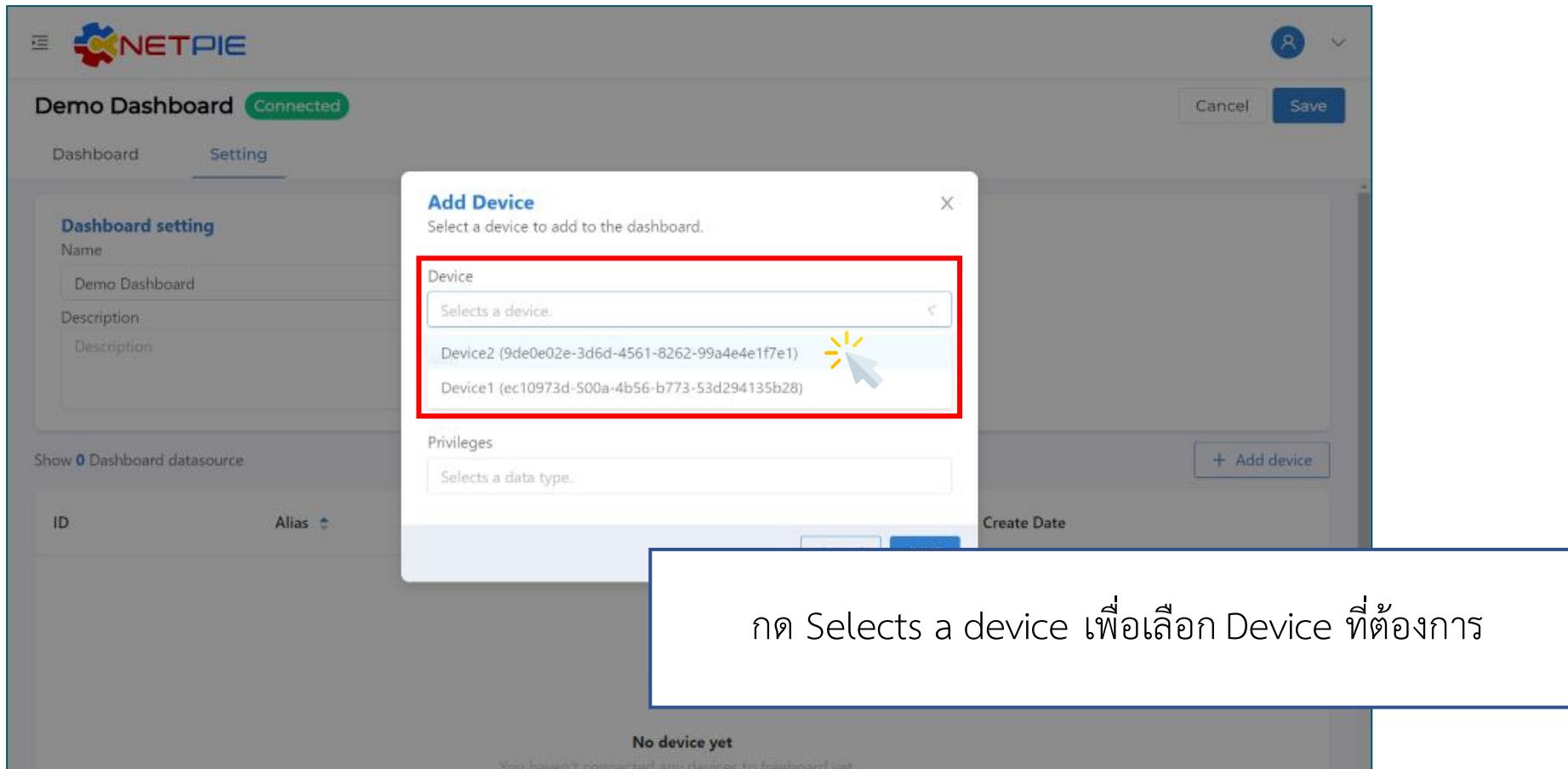


เมื่อเข้ามาจะเป็นหน้าต่างที่ยังว่างเปล่า ให้กดไปที่ Setting เพื่อตั้งค่า Dashboard

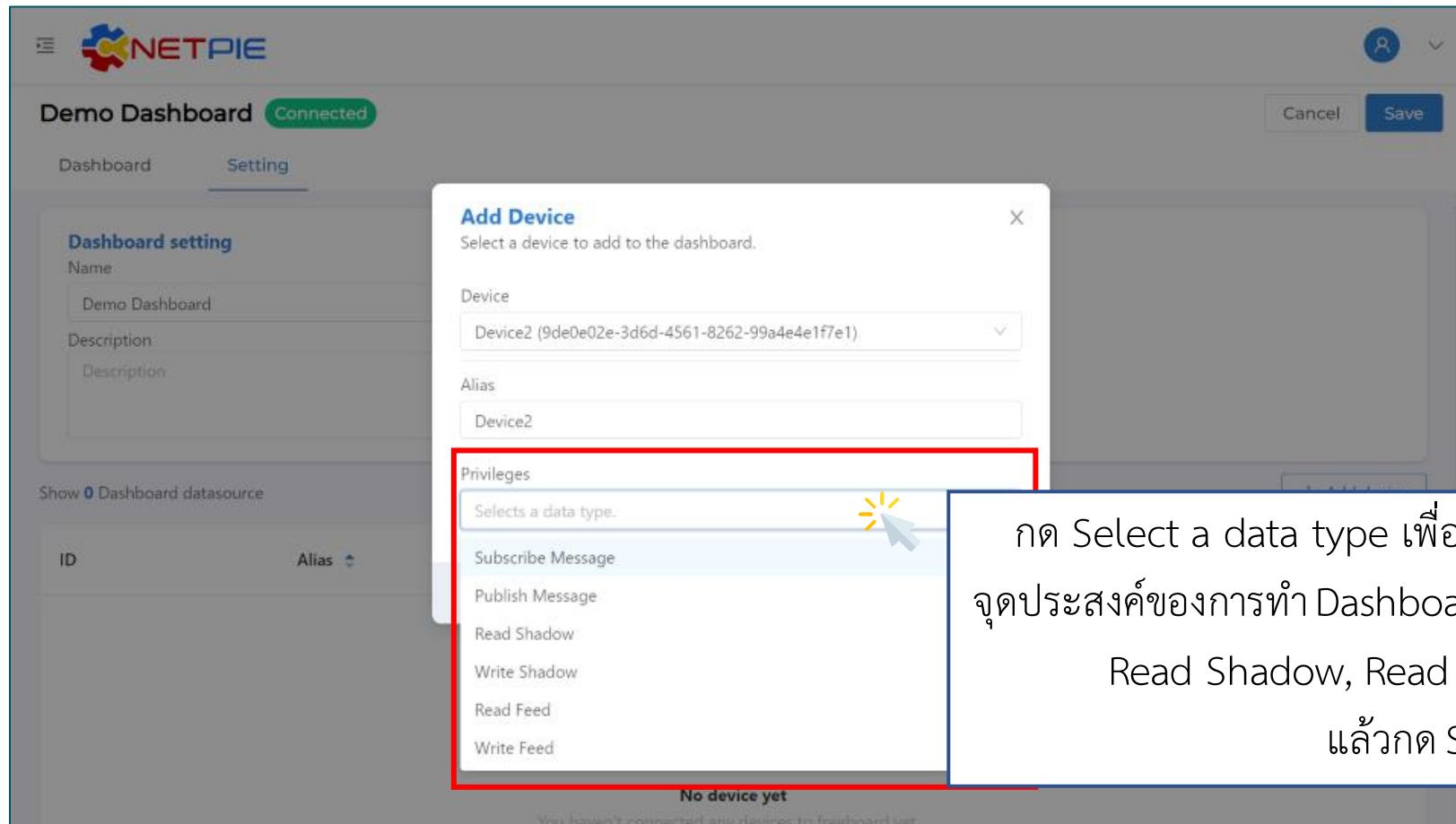
การสร้าง Datasource



การสร้าง Datasource



การสร้าง Datasource



กด Select a data type เพื่อเลือกประเภทของ Privileges ให้ตาม
จุดประสงค์ของการทำ Dashboard โดยมีหลาย Actions ให้เลือกใช้ เช่น
Read Shadow, Read Feed หรือ Publish Message
แล้วกด Save เพื่อบันทึก

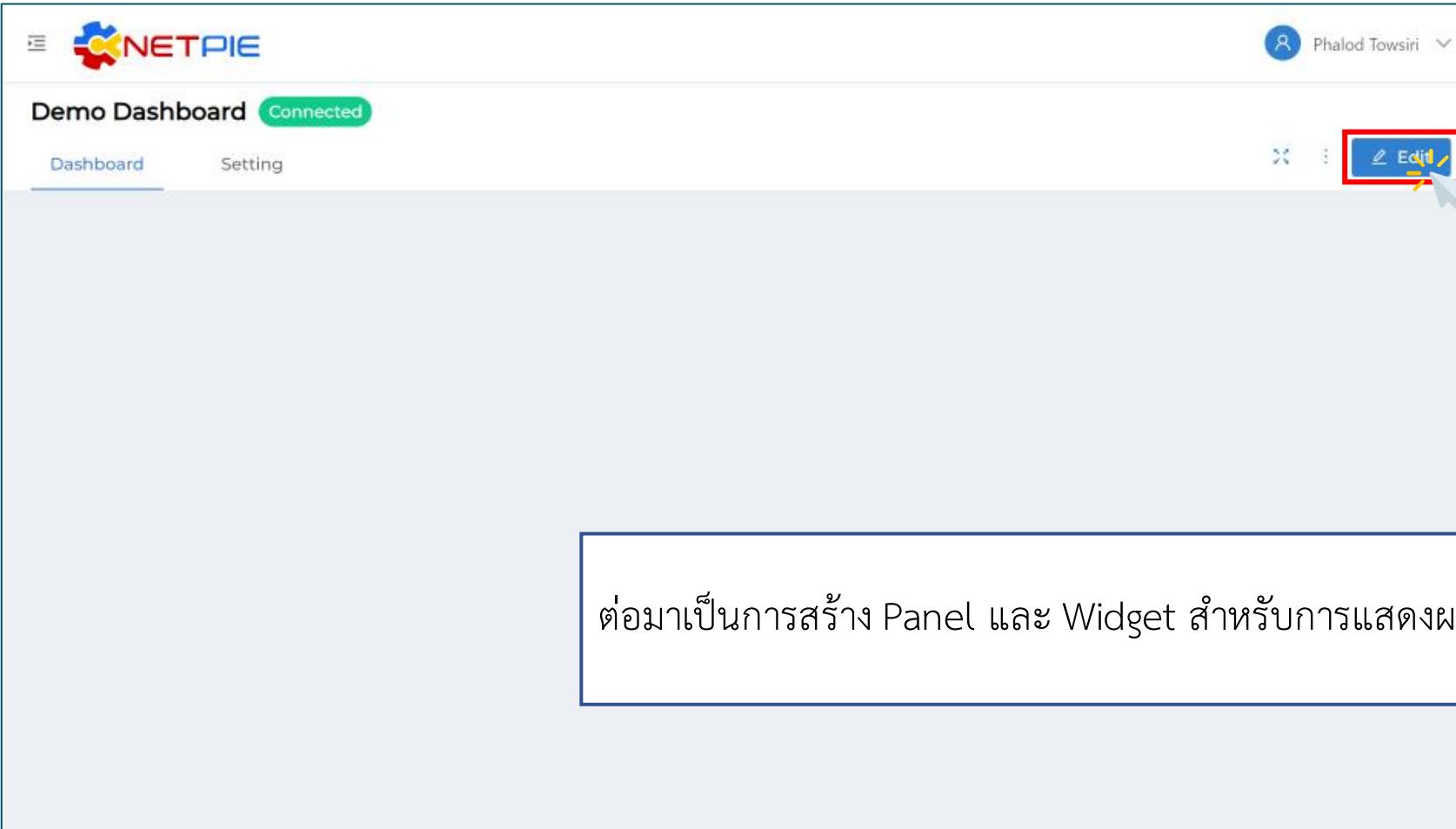
4 – NETPIE2020

NETPIE Dashboard คืออะไร

The screenshot shows the NETPIE Dashboard settings interface. At the top, it says "Demo Dashboard Connected". Below that, there are tabs for "Dashboard" and "Setting", with "Setting" being the active tab. Under "Setting", there's a "Dashboard setting" section with fields for "Name" (set to "Demo Dashboard") and "Description". A note on the right says: "จะได้ Datasource จาก Device และ Privileges ตามที่ได้ระบุไว้ พร้อมในการทำ Dashboard ในขั้นต่อไป". Below this, there's a table titled "Show 1 Dashboard datasource" with one entry. The table has columns: ID, Alias, Privileges, and Create Date. The entry shows an ID of "9de0e02e-3d6d-4561-8262-99a4e4e1f7e1", an alias of "Device2", and a row of privilege icons: R Message (red), W Message (pink), R Shadow (orange), W Shadow (yellow), R Feed (blue), and W Feed (light blue). The "Create Date" is "2023-01-13 14:15". A red box highlights the first row of the table.

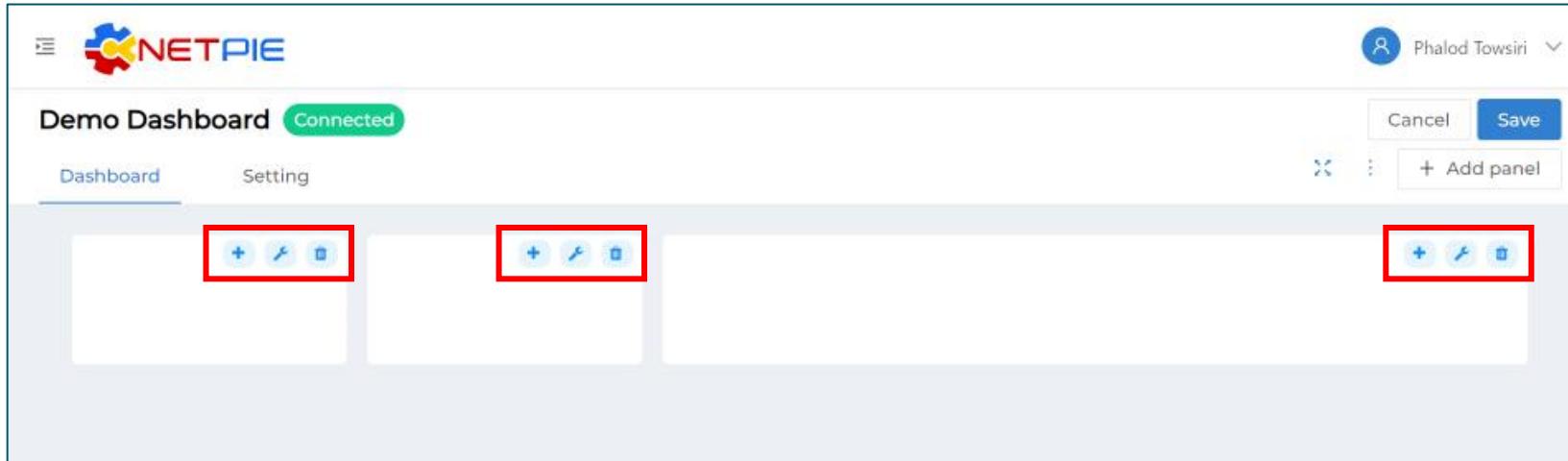
ID	Alias	Privileges	Create Date
9de0e02e-3d6d-4561-8262-99a4e4e1f7e1	Device2	R Message (red) W Message (pink) R Shadow (orange) W Shadow (yellow) R Feed (blue) W Feed (light blue)	2023-01-13 14:15

การใช้งาน Dashboard



ต้องมาเป็นการสร้าง Panel และ Widget สำหรับการแสดงผลข้อมูล โดยกดไปที่ Edit

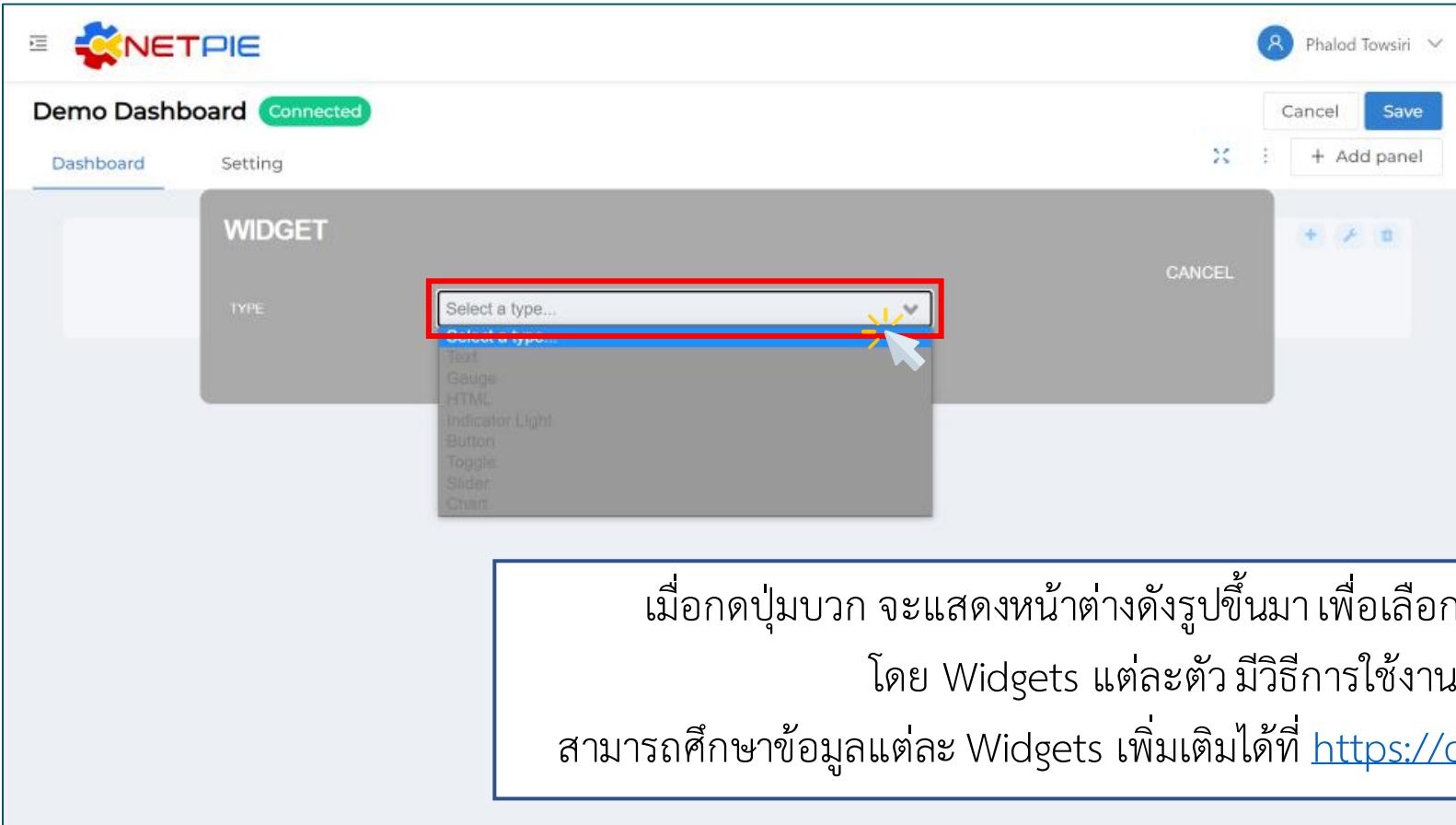
การใช้งาน Dashboard



สามารถกด Add Panel เพิ่ม Panel ได้ตามที่ต้องการ โดยบน Panel จะประกอบไปด้วยปุ่ม 3 แบบ คือ

1. ปุ่มบวก ใช้เพื่อเพิ่ม Widget เข้าไปใน Panel
2. ปุ่มตั้งค่า ใช้เพื่อตั้ง Title ให้ Panel และปรับขนาดของ Panel
3. ปุ่มถังขยะ ใช้เพื่อลบ Panel ที่ไม่ต้องการ

การใช้งาน Dashboard



The screenshot shows the NETPIE Dashboard interface titled "Demo Dashboard Connected". On the left, there's a "WIDGET" section with a "TYPE" dropdown menu open. A red box highlights the dropdown menu, and a cursor points to the dropdown arrow. Below the dropdown is a list of widget types: Text, Gauge, HTML, Indicator Light, Button, Toggle, Slider, and Chart. In the top right corner of the dashboard, there are "Cancel" and "Save" buttons, and a "+ Add panel" button.

เมื่อกดปุ่มบวก จะแสดงหน้าต่างดังรูปขึ้นมา เพื่อเลือก Widget ที่ต้องการจะแสดง โดย Widgets แต่ละตัว มีวิธีการใช้งานที่แตกต่างกัน สามารถศึกษาข้อมูลแต่ละ Widgets เพิ่มเติมได้ที่ <https://docs.netpie.io/freeboard.html>

การใช้งาน Dashboard

Demo Dashboard Connected

Dashboard Setting

WIDGET

Humidity 59

TYPE Text

TITLE Humidity

SIZE Regular

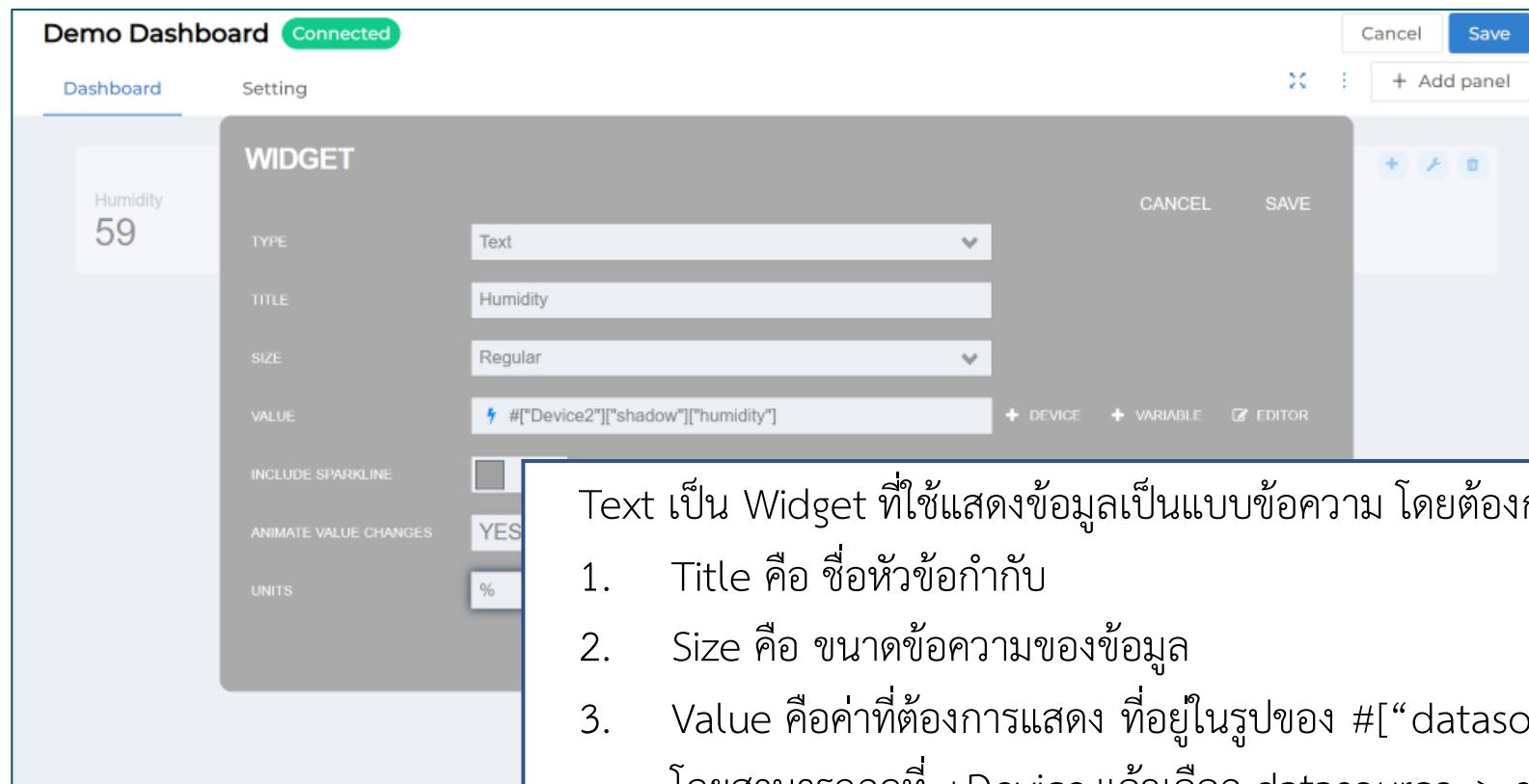
VALUE #["Device2"]["shadow"]["humidity"]

INCLUDE SPARKLINE YES

ANIMATE VALUE CHANGES

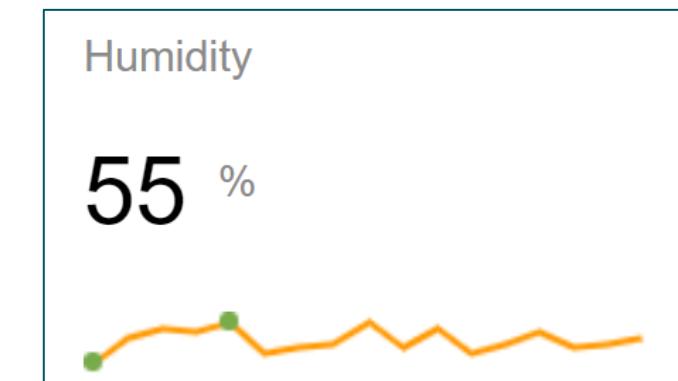
UNITS %

CANCEL SAVE + Add panel



Humidity

55 %



Text เป็น Widget ที่ใช้แสดงข้อมูลเป็นแบบข้อความ โดยต้องกรอกข้อมูลต่างๆ ดังนี้

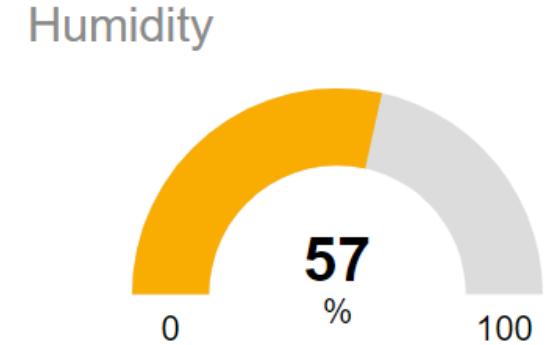
1. Title คือ ชื่อหัวข้อกำกับ
2. Size คือ ขนาดข้อความของข้อมูล
3. Value คือค่าที่ต้องการแสดง ที่อยู่ในรูปของ #[“datasource”][“shadow”][“field”] โดยสามารถกดที่ +Device และเลือก datasource -> shadow -> field ได้ตามลำดับ
4. Sparkline คือ กราฟขนาดเล็ก
5. Animate value change คือ อนิเมชั่นเมื่อค่าเปลี่ยน
6. Units คือ หน่วยของข้อมูล

By Piyawat Jomsathan

Page 220

การใช้งาน Dashboard

The screenshot shows the NETPIE Dashboard configuration interface. On the left, there's a preview panel showing a digital gauge reading "Humidity 59". The main area is a configuration dialog for a "Widget". The "TYPE" is set to "Gauge". The "TITLE" is "Humidity". The "VALUE" field contains the expression "#["Device2"]["shadow"]["humidity"]". The "UNITS" field is "%". The "MINIMUM" is "0" and the "MAXIMUM" is "100". There are buttons for "+ DEVICE", "+ VARIABLE", and "EDITOR". At the bottom right of the dialog are "CANCEL" and "SAVE" buttons. Above the dialog, there are "Cancel" and "Save" buttons for the dashboard itself. A "+ Add panel" button is also visible.

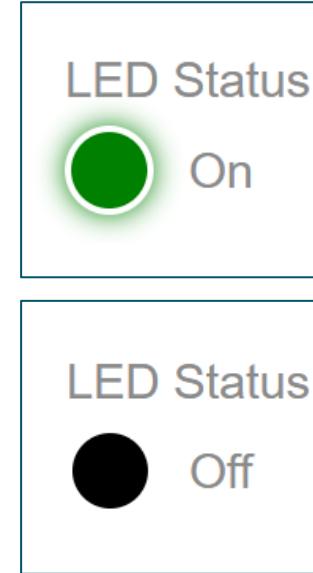


Gauge เป็น Widget ที่ใช้แสดงข้อมูลเป็นแบบแอบแบบแสดงข้อมูล โดยต้องกรอกข้อมูลต่างๆ ดังนี้

1. Title คือ ชื่อหัวข้อกำกับ
2. Value คือค่าที่ต้องการแสดง ที่อยู่ในรูปของ #["datasource"]["shadow"]["field"] โดยสามารถกดที่ +Device และเลือก datasource -> shadow -> field ได้ตามลำดับ
3. Units คือ หน่วยของข้อมูล
4. Minimum คือ ค่าต่ำสุดของ Gauge นี้
5. Maximum คือ ค่าสูงสุดของ Gauge นี้

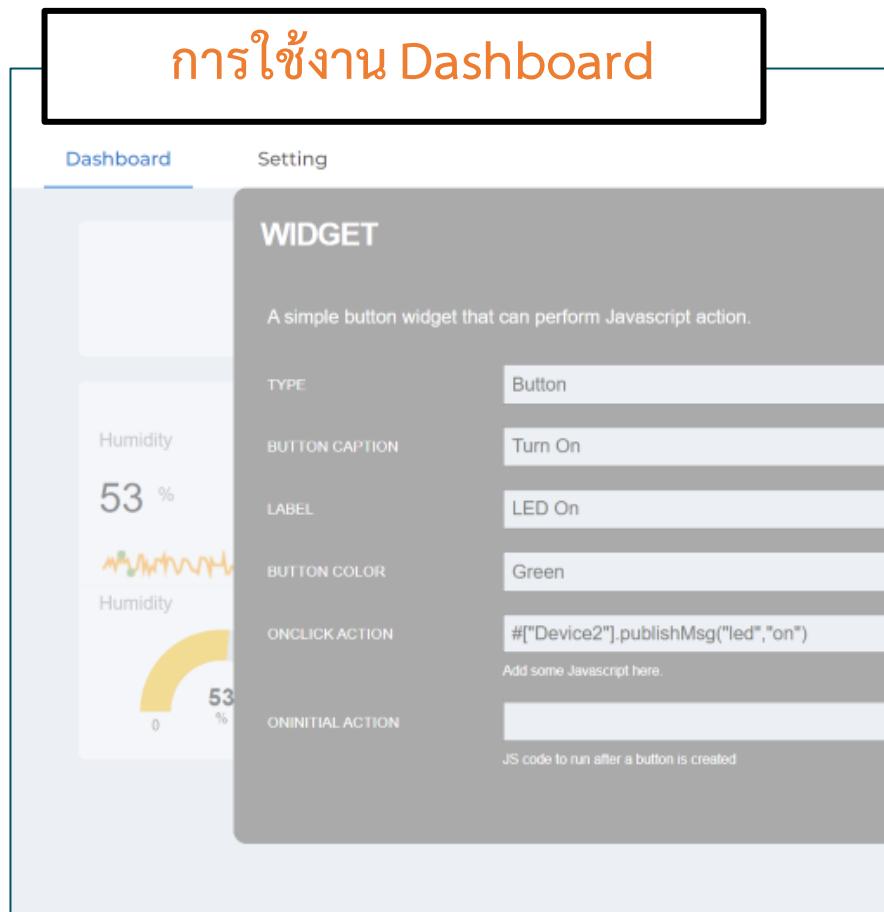
การใช้งาน Dashboard

The screenshot shows the NETPIE Dashboard configuration interface. On the left, there's a preview of the dashboard with a humidity sensor reading of 54%. In the center, a modal window titled "WIDGET" is open for creating a new indicator light. The "TYPE" is set to "Indicator Light", the "TITLE" is "LED Status", the "LIGHT COLOR" is "green", and the "VALUE" is a Boolean expression: "#["Device2"]["shadow"]["led"] == "on"". Below this, the "ON TEXT" is "On" and the "OFF TEXT" is "Off". At the top right of the modal are "Cancel" and "Save" buttons, and at the bottom right are "CANCEL" and "SAVE" buttons. A "Dashboard" tab is selected on the left, and a "Setting" tab is also present.



Indicator Light เป็น Widget ที่ใช้แสดงข้อมูลเป็นไฟสถานะ โดยต้องกรอกข้อมูลต่างๆ ดังนี้

1. Title คือ ชื่อหัวข้อกำกับ
2. Light Color คือ สีของไฟในสถานะ On
3. Value คือค่าที่ต้องการแสดง ที่อยู่ในรูปของ Boolean โดย 1, True คือ On เช่น #["datasource"]["shadow"]["field"] == "on"
4. On text คือ ข้อความเมื่อเปิด
5. Off text คือ ข้อความเมื่อปิด



Turn On LED On

Turn Off LED Off

Button เป็น Widget ที่ใช้ส่งข้อมูลเมื่อกด โดยต้องกรอกข้อมูลต่างๆ ดังนี้

1. Button Caption คือ ข้อความในปุ่ม
2. Label คือ ข้อความกำกับข้างปุ่ม
3. Button Color คือ สีของปุ่ม
4. Onclick Action คือ การส่งข้อมูลเมื่อกดปุ่มออกไป โดยอยู่ในรูปแบบของ
#[“datasource”].publishMsg(“topic”, “msg”)

การใช้งาน Dashboard

Setting

A simple toggle widget that can perform Javascript action.

TYPE: Toggle

TOGGLE ON CAPTION: ON

TOGGLE OFF CAPTION: OFF

LABEL: LED Control

TOGGLE STATE: Add a condition to switch a toggle state here. Otherwise it just toggle by click.

ONTOGGLEON ACTION: `#[Device2].publishMsg("led", "on")`
JS code to run when a toggle is switched to ON

ONTOGGLEOFF ACTION: `#[Device2].publishMsg("led", "off")`
JS code to run when a toggle is switched to OFF

ONINITIAL ACTION: `JS code to run after a toggle is created`

Cancel Save + Add panel

LED Control

LED Control

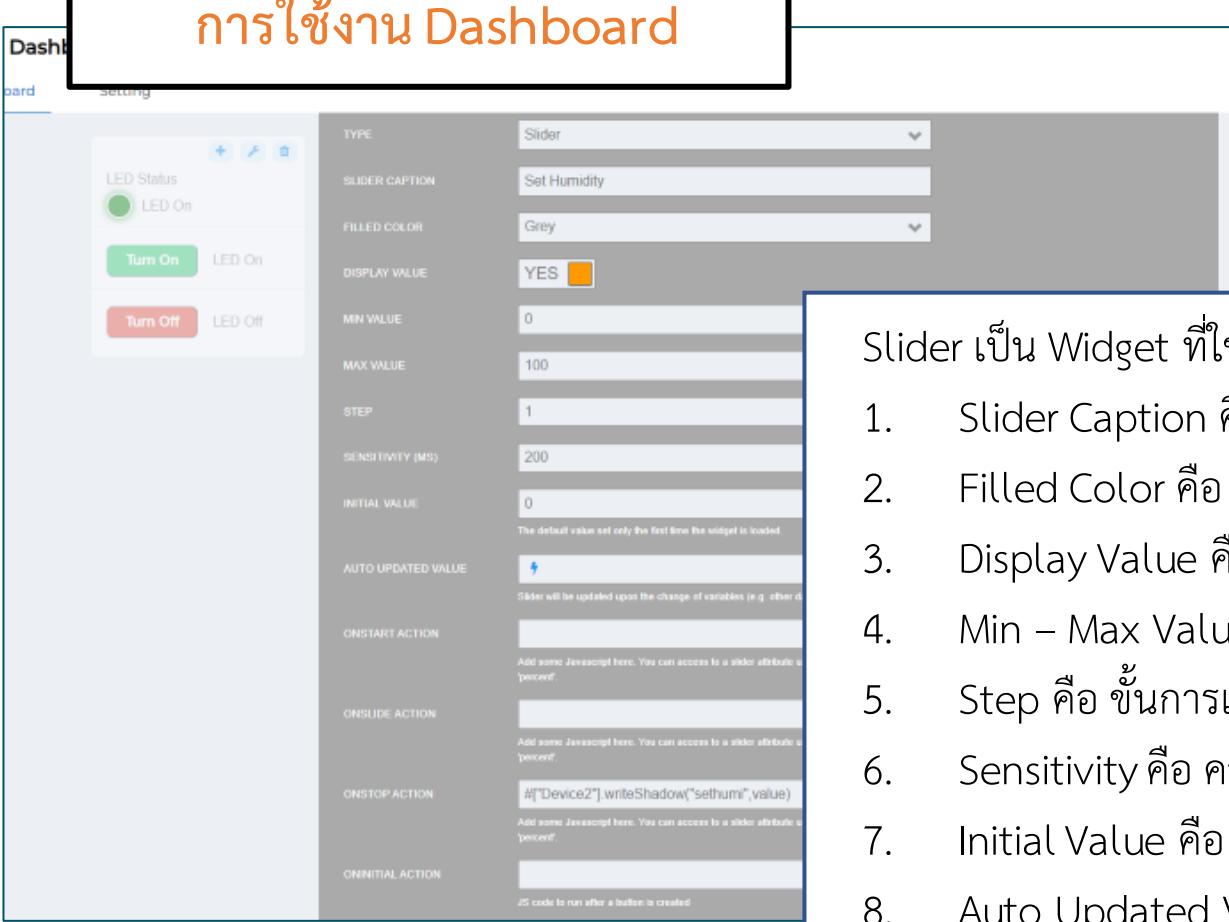
Toggle เป็น Widget ที่ใช้ส่งข้อความเมื่อสถานะของปุ่มเปลี่ยน โดยต้องกรอกข้อมูลต่างๆ ดังนี้

1. Toggle On Caption คือ ข้อความในปุ่มเมื่อเปิด
2. Toggle Off Caption คือ ข้อความในปุ่มเมื่อปิด
3. Label คือ ข้อความกำกับข้างปุ่ม
4. Toggle State เว้นไว้เพื่อเปิดปิดเมื่อกดปุ่ม หรือจะตั้งค่าเพื่อให้เปิดปิดตามสถานการณ์
5. On Toggle Action คือ การส่งข้อมูลเมื่อ Toggle เปลี่ยนเป็น On โดยอยู่ในรูปแบบของ `#[“datasource”].publishMsg(“topic”, “msg”)`
6. Off Toggle Action คือ การส่งข้อมูลเมื่อ Toggle เปลี่ยนเป็น Off โดยอยู่ในรูปแบบของ `#[“datasource”].publishMsg(“topic”, “msg”)`

By Piyawat Jomsathan

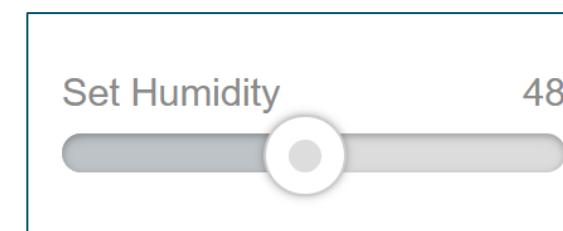
Page 22

การใช้งาน Dashboard



The configuration interface shows a Slider widget being set up. The configuration fields include:

- TYPE:** Slider
- SLIDER CAPTION:** Set Humidity
- FILLED COLOR:** Grey
- DISPLAY VALUE:** YES [checkbox]
- MIN VALUE:** 0
- MAX VALUE:** 100
- STEP:** 1
- SENSITIVITY (MS):** 200
- INITIAL VALUE:** 0
- AUTO UPDATED VALUE:** A dropdown menu with options like "Slider will be updated upon the change of variables (e.g. other device's value)" and "JS code to run after a button is created".
- ONSTART ACTION:** A text area with placeholder "Add some Javascript here. You can access to a slider attribute via 'percent'." and a sample code: `#[["Device2"],writeShadow("sethum!",value)]`.
- ONSIDE ACTION:** A text area with placeholder "Add some Javascript here. You can access to a slider attribute via 'percent'." and a sample code: `#[["Device2"],writeShadow("sethum!",value)]`.
- ONSTOP ACTION:** A text area with placeholder "Add some Javascript here. You can access to a slider attribute via 'percent'." and a sample code: `#[["Device2"],writeShadow("sethum!",value)]`.
- ONINITIAL ACTION:** A text area with placeholder "JS code to run after a button is created".



Set Humidity 48

Slider เป็น Widget ที่ใช้ส่งข้อความตัวเลขแบบแบบ滑尺式 โดยต้องกรอกข้อมูลต่างๆ ดังนี้

- Slider Caption คือ ข้อความกำกับบน Slider
- Filled Color คือ สีของແບ Slides
- Display Value คือ การเปิดหรือปิดแสดงค่า Slider
- Min – Max Value คือ ค่าต่ำสุดและสูงสุดของ Slider
- Step คือ ขั้นการเพิ่มขึ้นหรือลดลงของ Slider
- Sensitivity คือ ความไวในการเลื่อน Slider
- Initial Value คือ ค่าเริ่มต้นของ Slider
- Auto Updated Value คือ การผูกกับข้อมูลใน Datasource เพื่อให้ข้อมูลอัพเดตตามกัน
- On Start Action คือ Action เมื่อเริ่มกดที่ปุ่ม Slider
- On Slide Action คือ Action ระหว่างที่เลื่อน Slider
- On Stop Action คือ Action เมื่อปล่อยปุ่ม Slider

By Piyawat Jomsathan

การใช้งาน Dashboard

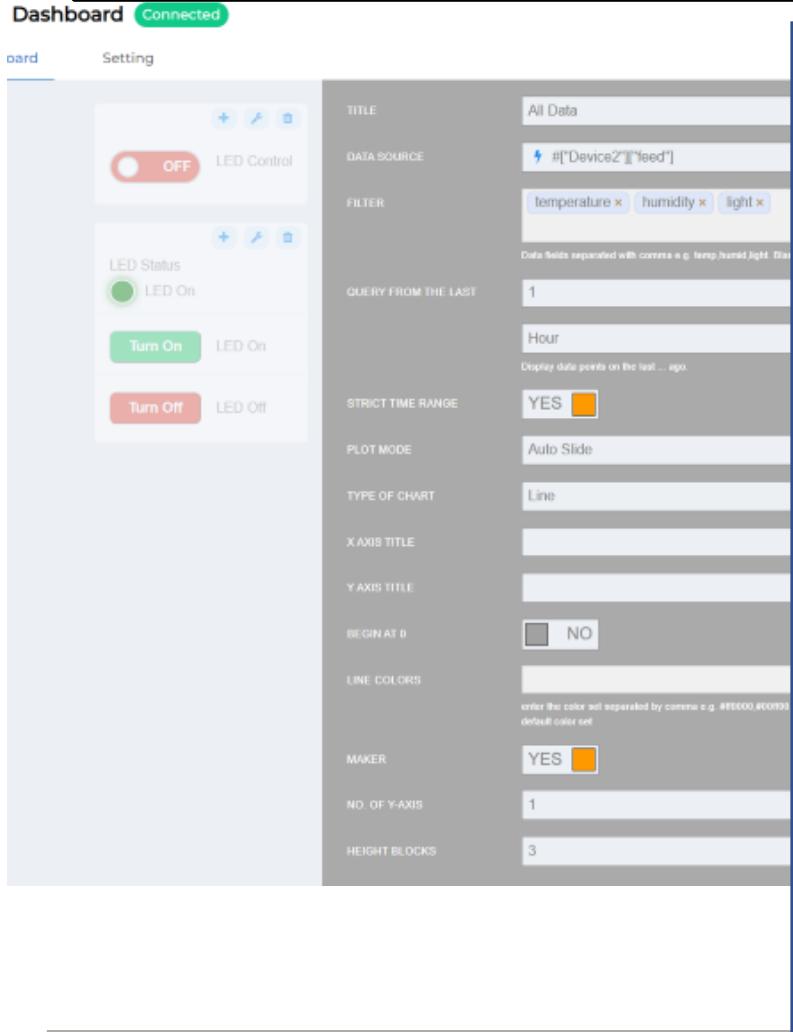
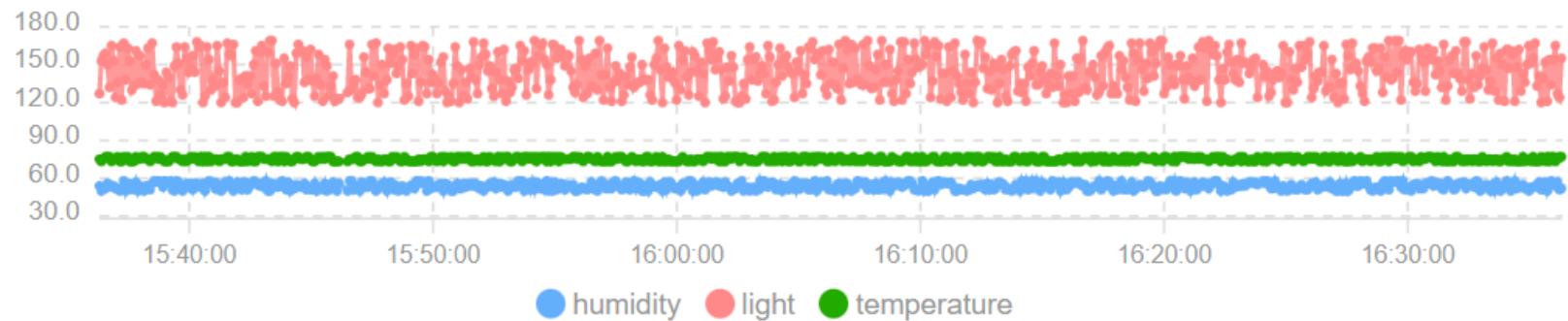


Chart เป็น Widget ที่ใช้แสดงข้อมูลกราฟในรูปแบบ Time-Series โดยต้องกรอกข้อมูลต่างๆ ดังนี้

1. Title คือ ชื่อกำกับ Chart
2. Datasource คือ Datasource ของข้อมูลที่จะแสดง ในรูปของ #[“datasource”][“feed”]
3. Filter คือ field ที่จะแสดงข้อมูล สามารถเว้นว่างเพื่อแสดงข้อมูลทั้งหมดได้
4. Query From The Last คือ เวลาในการดึงข้อมูลย้อนหลัง
5. Strict Time Range คือ การให้กราฟตรงช่วงเวลาตามที่ตั้งเอาไว้
6. Plot Mode คือ รูปแบบการplotข้อมูล
7. Type of Chart คือ รูปแบบของกราฟ
8. X-axis, Y-axis Title คือ ชื่อกำกับของแกน X, Y
9. Begin at 0 คือ ตั้งค่าจุดเริ่มของกราฟเป็น 0
10. Line color คือ สีของกราฟ
11. Marker คือ เปิดหรือปิดจุดของแต่ละข้อมูล
12. No of Y-axis คือ จำนวนแกน Y ที่แสดง
13. Height Blocks คือ ความสูงของกราฟ

การใช้งาน Dashboard

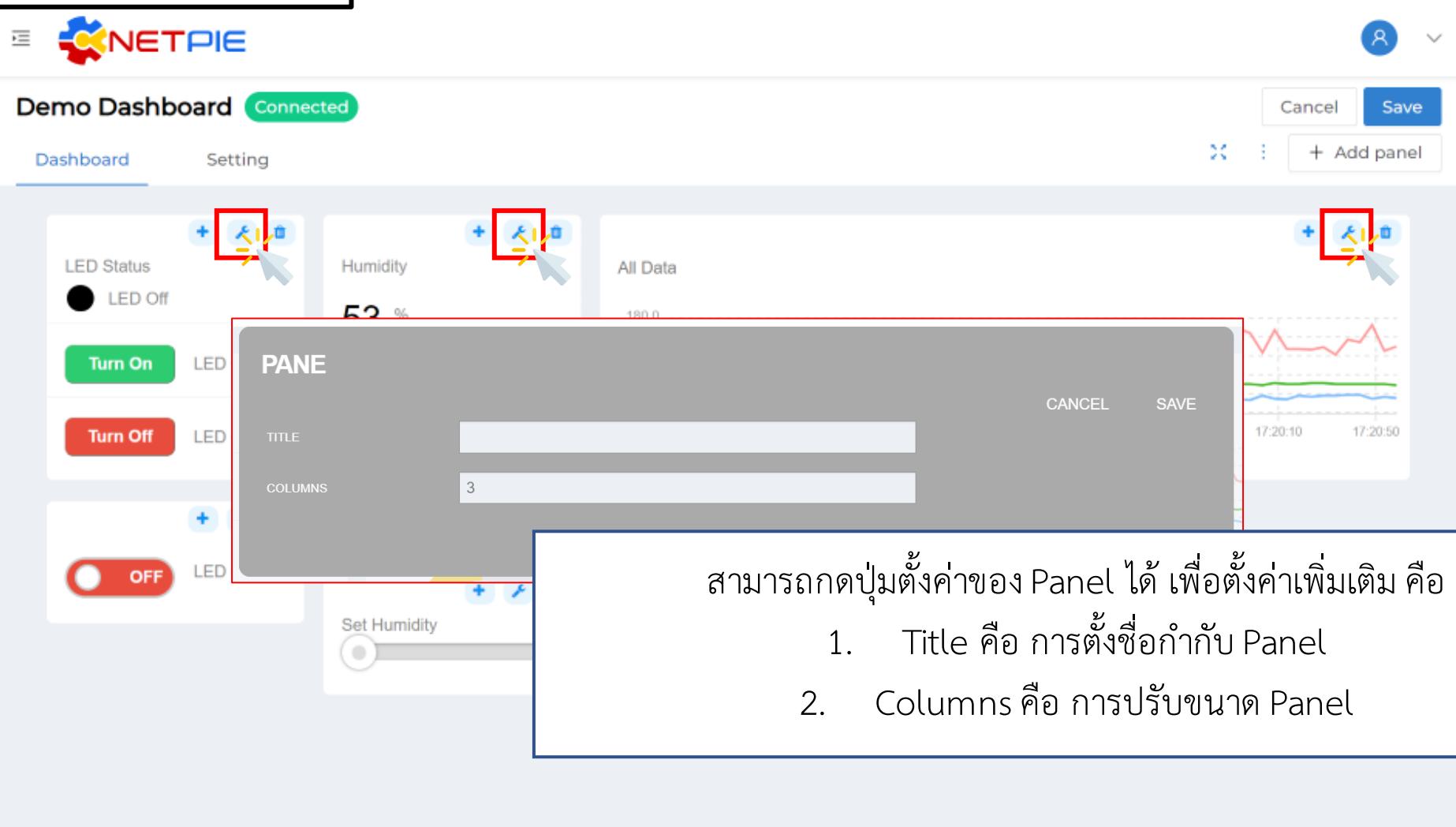
All Data



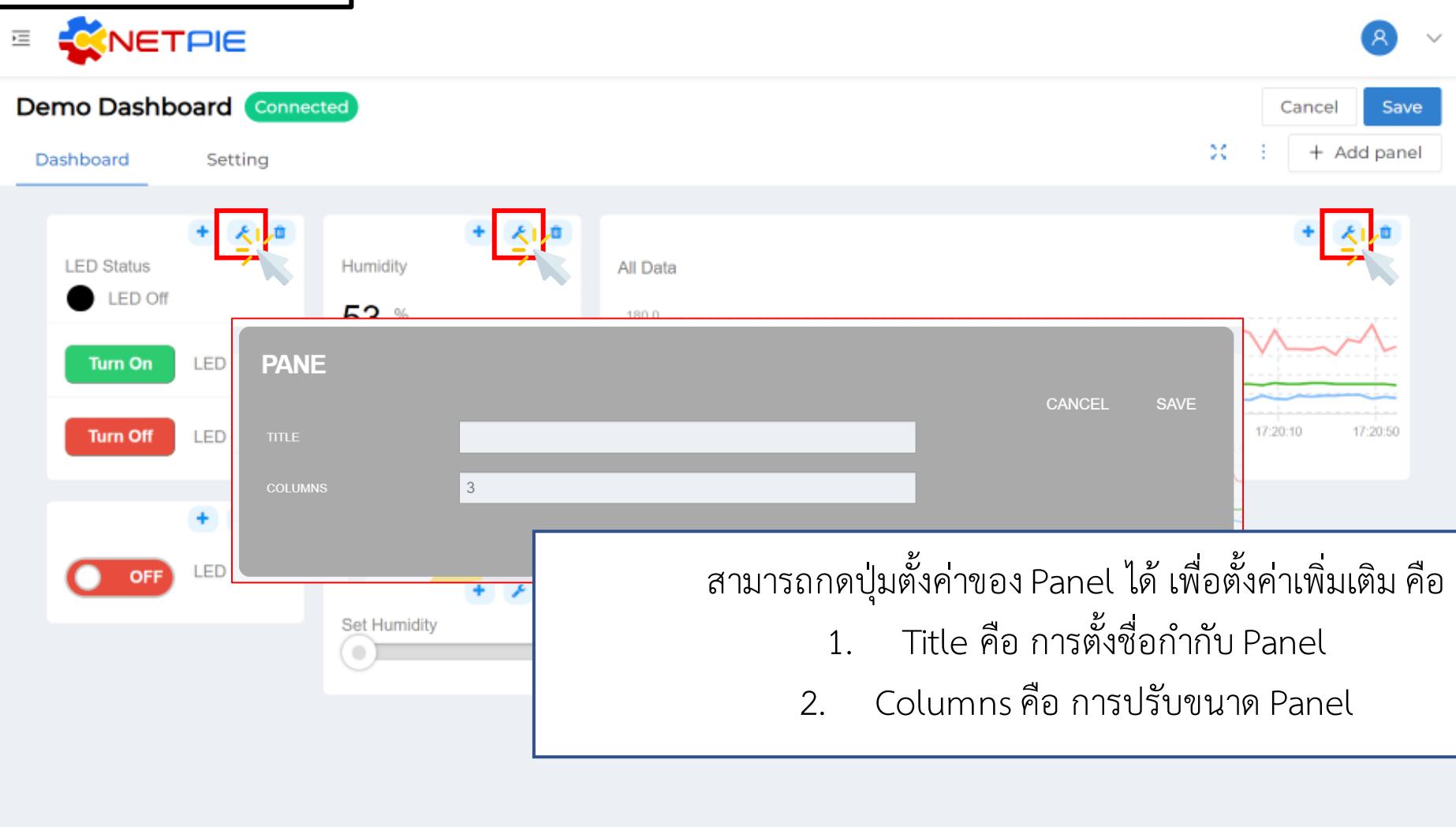
Humidity



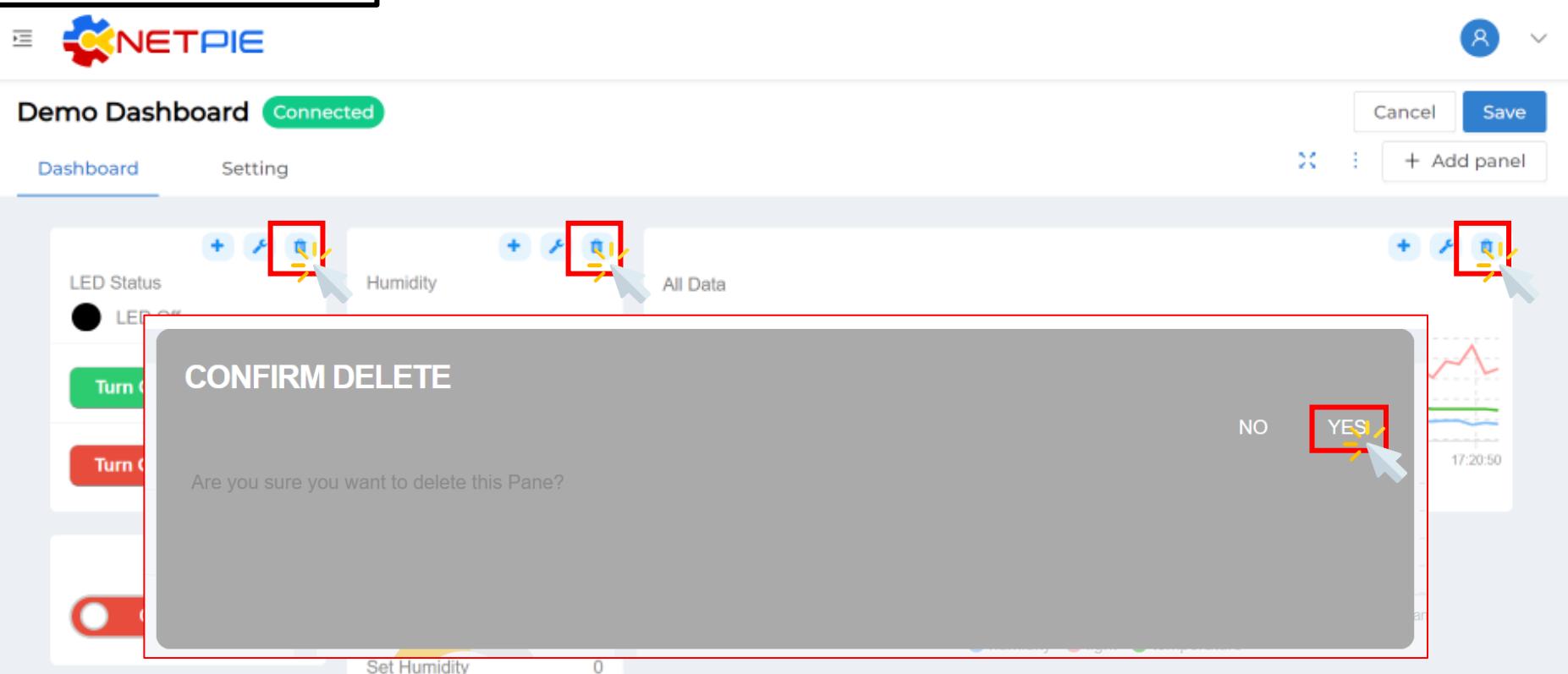
การใช้งาน Dashboard



การใช้งาน Dashboard



การใช้งาน Dashboard



สามารถกดปุ่มถังขยะของ Panel เพื่อลบ Panel ที่ไม่ต้องการ

การใช้งาน Dashboard

The screenshot shows the NETPIE Dashboard interface titled "Demo Dashboard" with a "Connected" status. The interface includes several panels:

- LED Status:** Shows a button to "Turn On" (green) and "Turn Off" (red). Below it is a "LED Control" section with a switch labeled "OFF".
- Humidity:** Displays a digital value of 50 %, a line graph showing fluctuating humidity levels, and a gauge meter also showing 50 %.
- All Data:** A line chart titled "All Data" showing three data series over time: humidity (blue), light (red), and temperature (green). The chart shows a general downward trend from approximately 150 to 100 over the hour shown.

In the top right corner, there is a modal dialog box with "Save" highlighted by a red box and a yellow arrow. The dialog also contains "Cancel" and "+ Add panel" buttons.

Text overlay: "ทำการกด Save เพื่อบันทึกการเปลี่ยนแปลงทุกครั้ง"

การใช้งาน Dashboard



ชี้ไปที่ปุ่มจุด ดังภาพ เพื่อเลือก

1. Import Dashboard ใหม่ที่ต้องการ โดยต้องเป็นไฟล์สกุล JSON
2. Export Dashboard นี้ เก็บเป็นไฟล์ JSON
3. Reset Dashboard เพื่อล้าง Panel ทั้งหมด

สร้างการแจ้งเตือนผ่าน Line Notification



LINE Notify
Connect Everything



Device Trigger and Event Hook

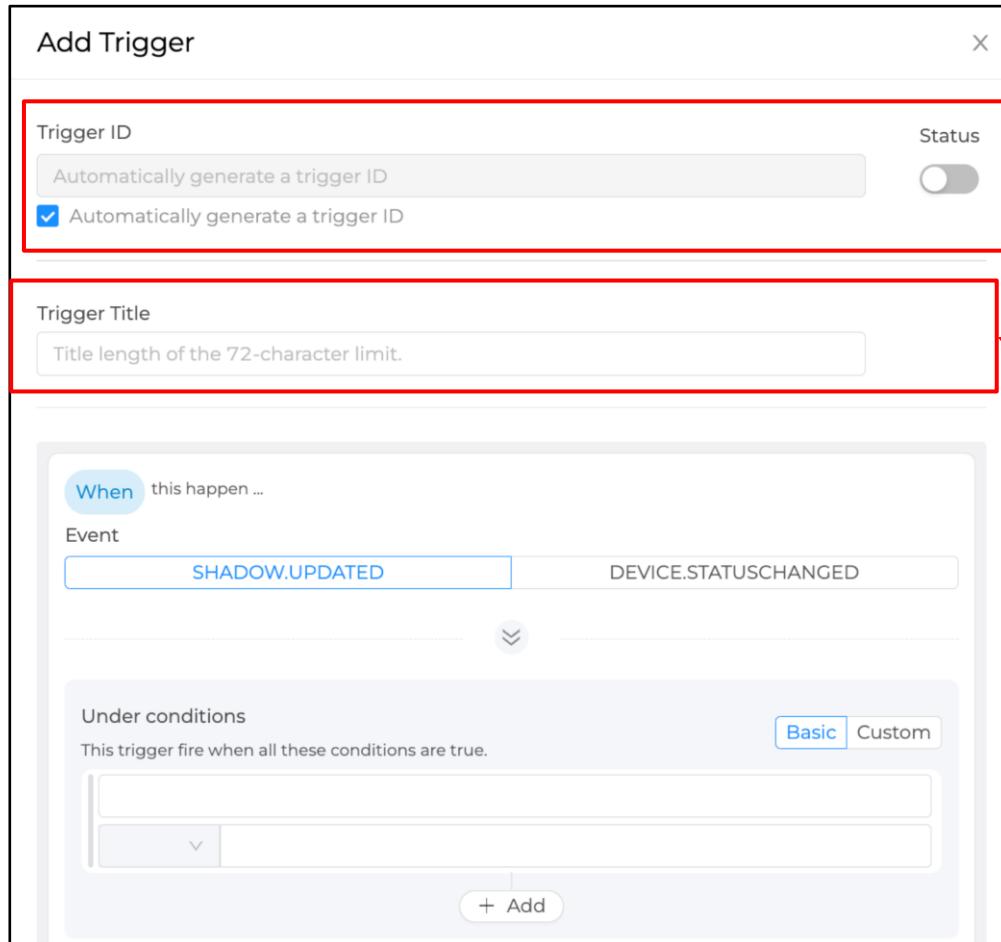
Device Trigger

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

Event Hook

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

ส่วนต่างๆของ Device Trigger

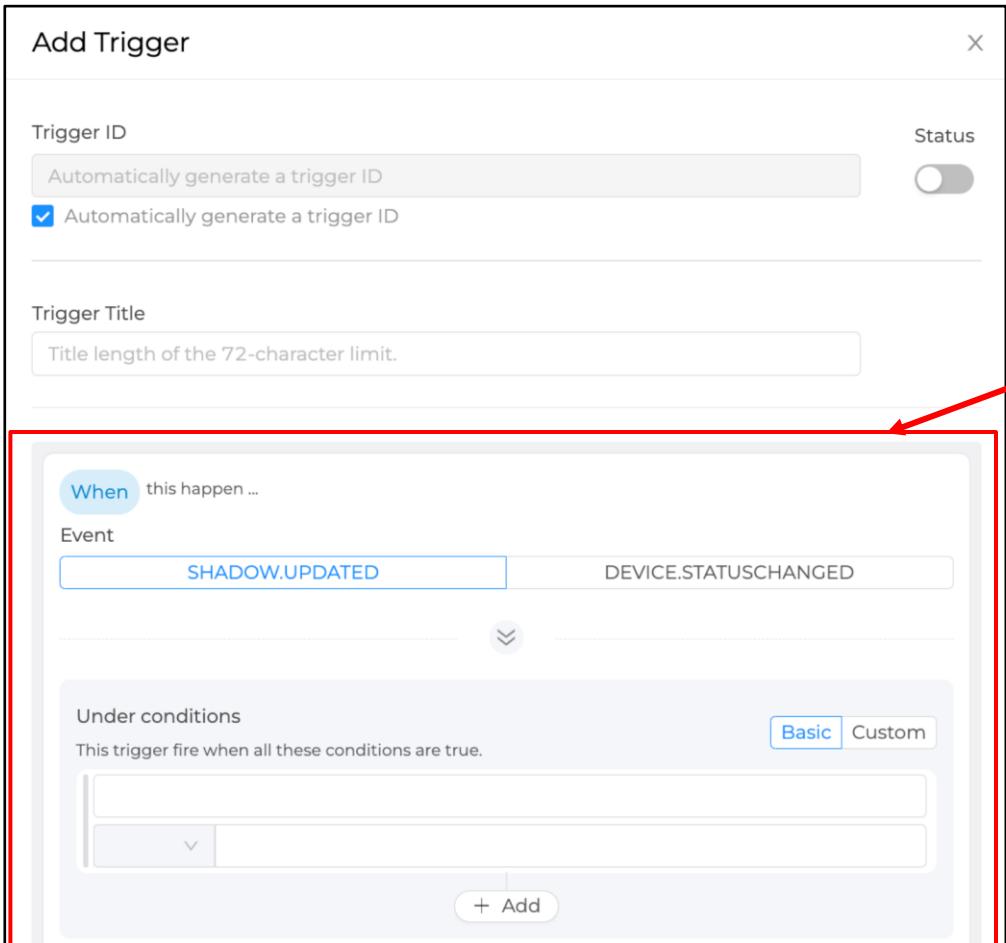


Trigger ID (string) : รหัสของ Trigger ซึ่งระบบจะสร้างให้อัตโนมัติหรือผู้ใช้ต้องการกำหนดเองก็ได้

Status : สถานะเปิด/ปิดการใช้งาน Trigger

Trigger Title (string) : ชื่อหรือคำอธิบายสั้น ๆ เกี่ยวกับ Trigger

ส่วนต่างๆของ Device Trigger

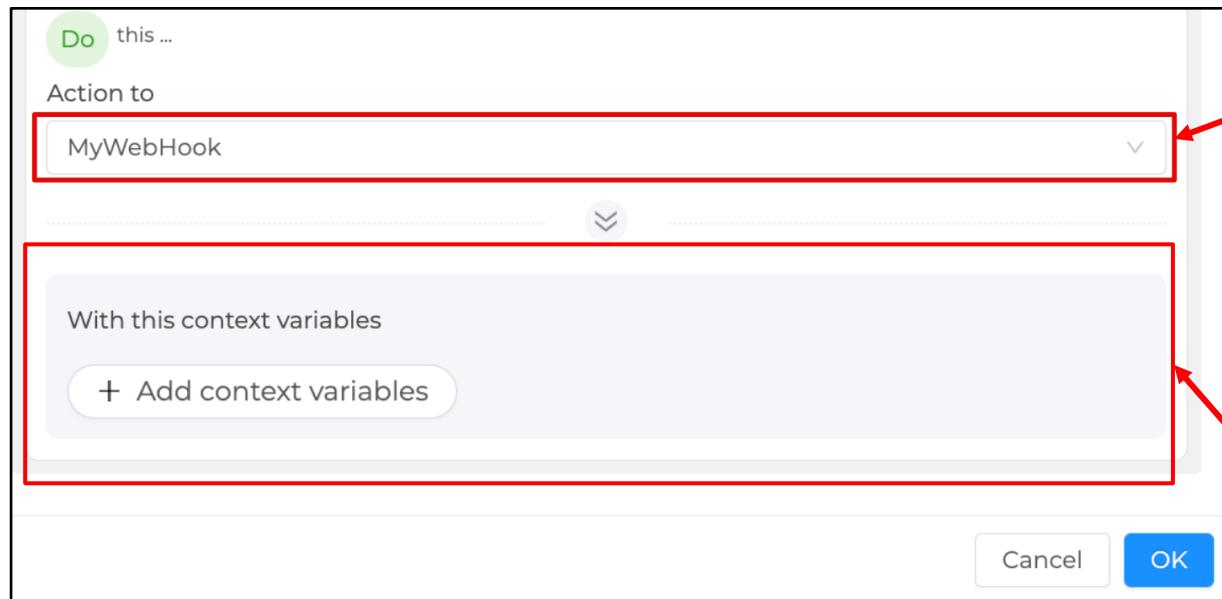


Event : ประมวลผลการเปลี่ยนแปลงข้อมูลของ Device (Device Shadow)

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 เปลี่ยนไปจาก Online เป็น Offline หรือ Offline เป็น Online ให้เซ็ตเงื่อนไข (Under conditions) ให้เป็นจริงเสมอ เช่น `true == true` หรือ `1 == 1` เป็นต้น
 2. ต้องการให้ Trigger ในกรณีที่เปลี่ยนสถานะเป็น Online เท่านั้น ให้เซ็ตให้เซ็ตเงื่อนไข (Under conditions) เป็น `$NEW.STATUS == 1`
 3. ต้องการให้ Trigger ในกรณีที่เปลี่ยนสถานะเป็น Offline เท่านั้น ให้เซ็ตให้เซ็ตเงื่อนไข (Under conditions) เป็น `$NEW.STATUS == 0`

ส่วนต่างๆของ Device Trigger



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

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

การอ้างอิงค่า Shadow ใน Trigger

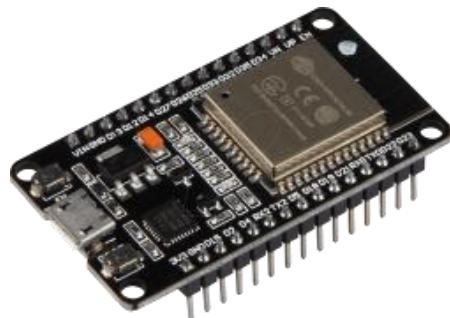
สำหรับการอ้างอิงค่าตัวแปร Shadow สามารถเรียกใช้ใน Condition หรือ Context Variable ของ Trigger มี 3 รูปแบบ ดังนี้

- 1 \$CUR.พารของตัวแปร คือ ค่าปัจจุบันล่าสุดที่ถูกอัพเดท (\$NEW merge \$PREV) โดยขึ้นต้นด้วย \$CUR ตามด้วย Path ตามโครงสร้างใน Shadow
- 2 \$NEW.พารของตัวแปร ค่าใหม่ที่ส่งมาอัพเดตลง Shadow โดยขึ้นต้นด้วย \$NEW ตามด้วย Path ตามโครงสร้างใน Shadow
- 3 \$PREV.พารของตัวแปร ค่าก่อนหน้าที่จะถูกอัพเดตลง Shadow โดยขึ้นต้นด้วย \$PREV ตามด้วย Path ตามโครงสร้างใน Shadow

ทั้งนี้ Device Trigger มีการอ้างอิงตัวแปรอื่นๆ ในระบบอีกมากมาย สามารถดูรายละเอียดได้ตามนี้

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

ตัวอย่างการทำงานของ Trigger and Event Hook



ESP8266/ESP32

Publish : @shadow/data/update



Trigger : DEVICE.STATUSCHANGE

Event Hooks
LINE NOTIFY



NETPIE2020

LINE

ตัวอย่างการทำงานของ Trigger and Event Hook



ESP8266/ESP32

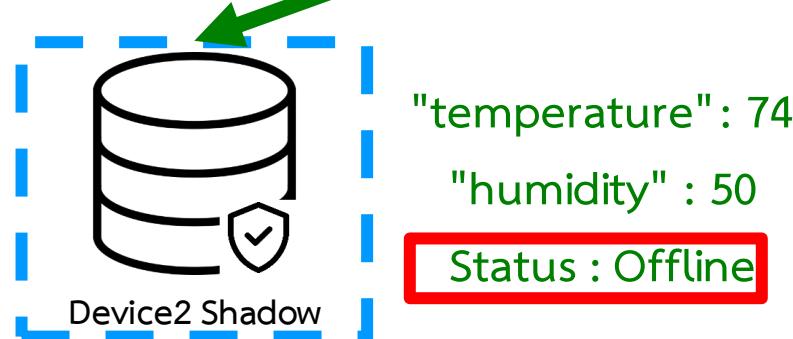
Publish : @shadow/data/update



Event Hooks
LINE NOTIFY



NETPIE2020



Trigger : DEVICE.STATUSCHANGE



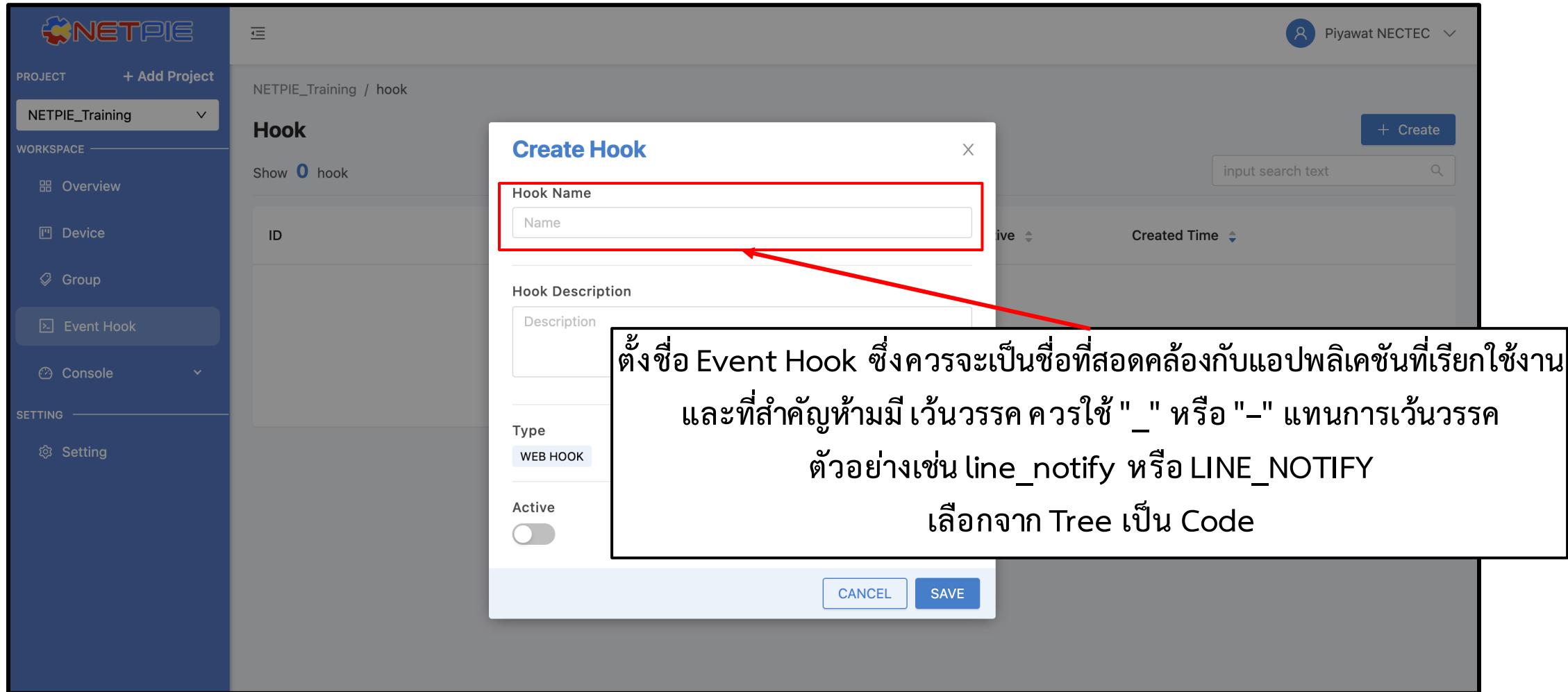
Example การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



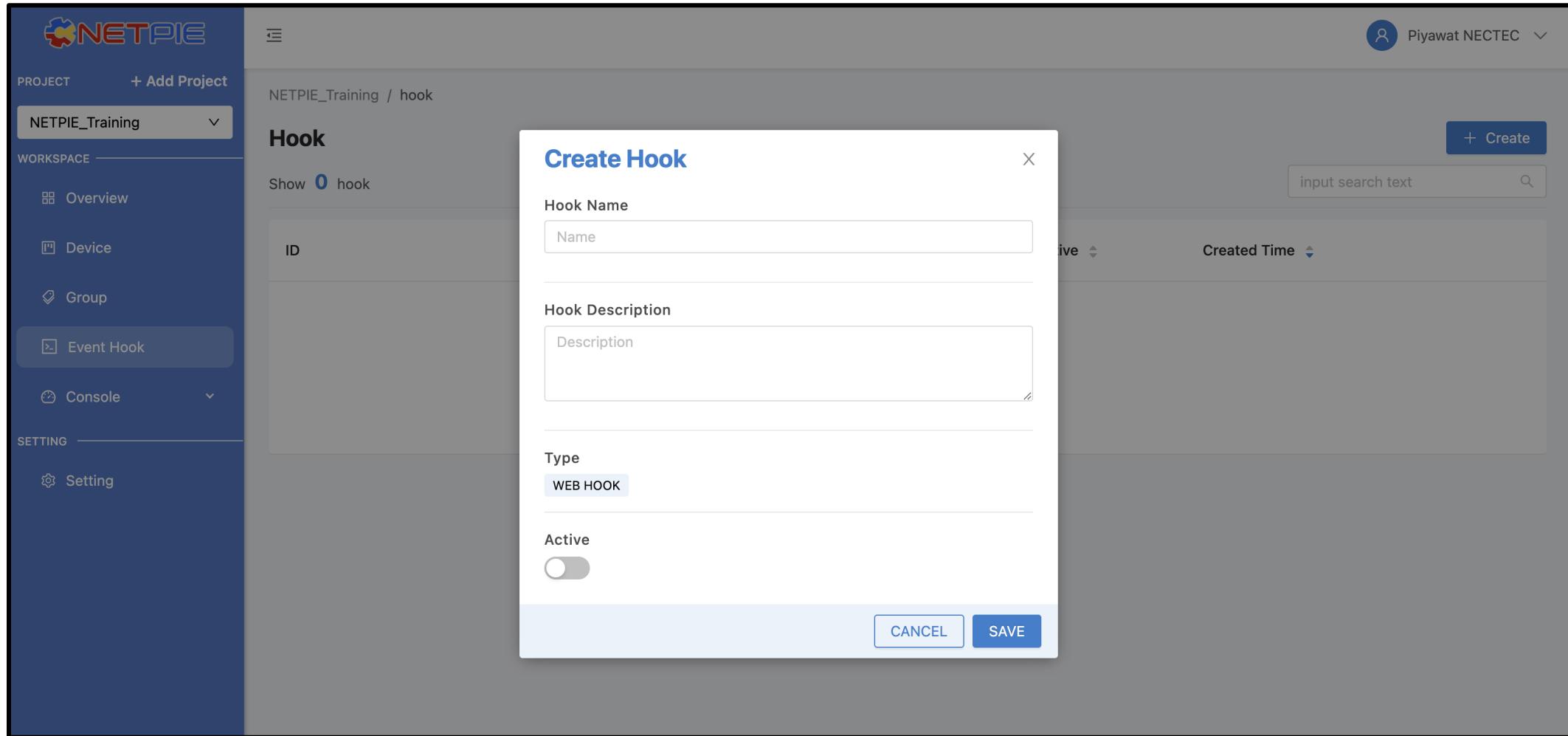
เราจะทำการสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify ซึ่งเป็น 3rd Party ที่ได้รับความนิยมสำหรับการทำระบบแจ้งเตือน

The screenshot shows the NETPIE web interface. On the left sidebar, under the 'PROJECT' section, the 'Event Hook' option is highlighted with a red box. At the top right, there is a user profile for 'Piyawat NECTEC'. In the main content area, the title 'NETPIE_Training / hook' is displayed above a table titled 'Hook'. The table has columns for 'ID', 'Name', 'Active', and 'Created Time'. A blue button labeled '+ Create' is located at the top right of the table area. A hand cursor is shown clicking on this button. Below the table, it says 'No Data'.

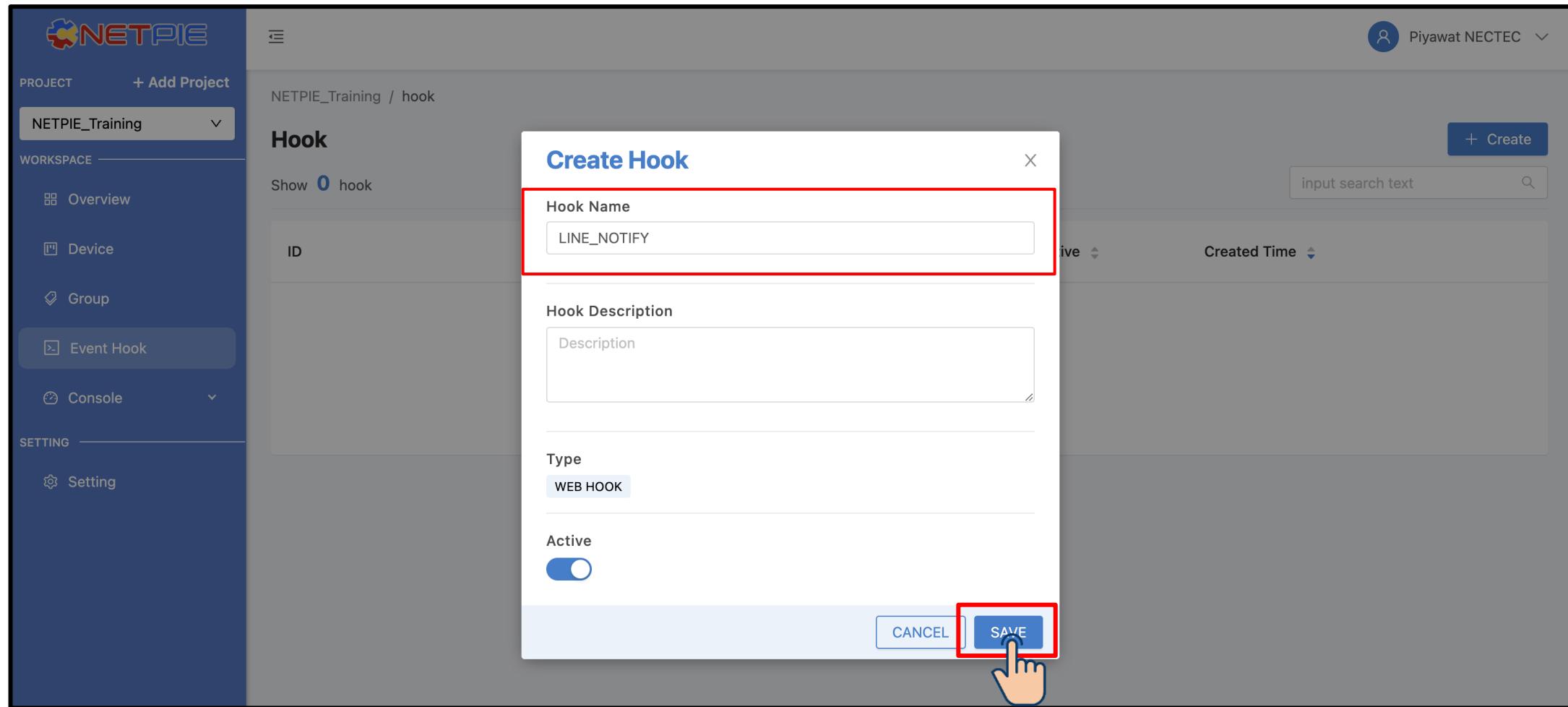
Example การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



Example การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



Example การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



Example การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

The screenshot shows the NETPIE web interface with the following details:

- Project:** NETPIE_Training
- Workspace:** Overview, Device, Group, **Event Hook** (selected), Console, Setting.
- Hook:** A table showing one hook entry:

ID	Name	Active	Created Time
H792573644699	LINE_NOTIFY	Enabled	2023-01-19 16:06
- Buttons:** + Create, input search text, 10 / page ▾.

Example การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

The screenshot shows the NETPIE Platform interface. On the left, there's a sidebar with 'PROJECT' (NETPIE_Training selected), 'WORKSPACE' (Overview, Device, Group, Event Hook selected, Console), and 'SETTING' (Setting). The main area shows a project structure: NETPIE_Training / hook / LINE_NOTIFY. The LINE_NOTIFY page has a 'Detail' section and a 'Configuration' section. In the Configuration section, 'Type' is set to 'WEB HOOK' and 'Status' is 'Enabled'. Below these, there are 'Event Hook' and 'i' buttons, and 'SAVE' and 'Cancel' buttons. A large input field contains a tree-based code editor. A red box highlights the 'Tree' button in the toolbar above the tree view. A callout box with the text 'เลือกจาก Tree เป็น Code' (Select from Tree as Code) points to the tree view. The tree view shows a single node: 'object {0}' under '(empty object)'.

Example 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

Example การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

The screenshot shows the NETPIE web interface. On the left sidebar, under the WORKSPACE section, the 'Event Hook' option is selected. In the main area, a project named 'LINE_NOTIFY' is displayed, created on 2023-01-19. The configuration shows it is a 'WEB HOOK' of type 'Enabled'. A red box highlights the code editor where the following JSON code is written:

```
1 {  
2   "body": "message={{context.msg}}",  
3   "header": {  
4     "Authorization": "Bearer {{context.linetoken}}",  
5     "Content-Type": "application/x-www-form-urlencoded"  
6   },  
7   "method": "POST",  
8   "uri": "https://notify-api.line.me/api/notify"  
9 }  
10 |
```

A hand cursor is pointing at the 'SAVE' button, which is highlighted with a red box. The 'Code' tab is active in the editor.

Example การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

คำอธิบายการใช้งาน Line Notify

```
{  
  "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"  
}
```

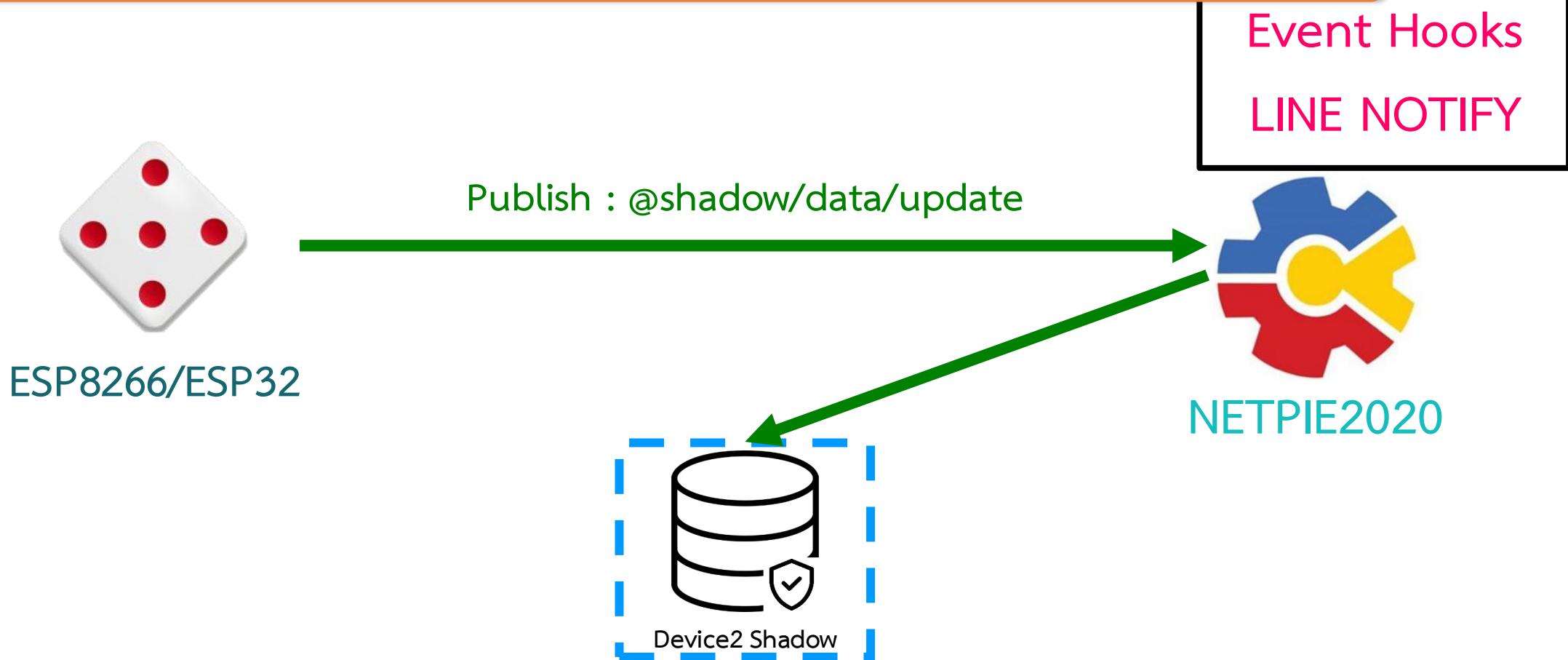
body คือ ส่วนของข้อมูลในที่นี่คือ ข้อความที่จะส่งไปยังปลายทาง

header คือ ข้อมูลเพิ่มเติมที่ต้องการส่งไปยังปลายทาง เช่น Authorization, Content-Type เป็นต้น

method คือ ส่วนที่กำหนดว่าปลายทางต้องการให้ส่งไปในแบบไหน GET, POST หรือ PUT

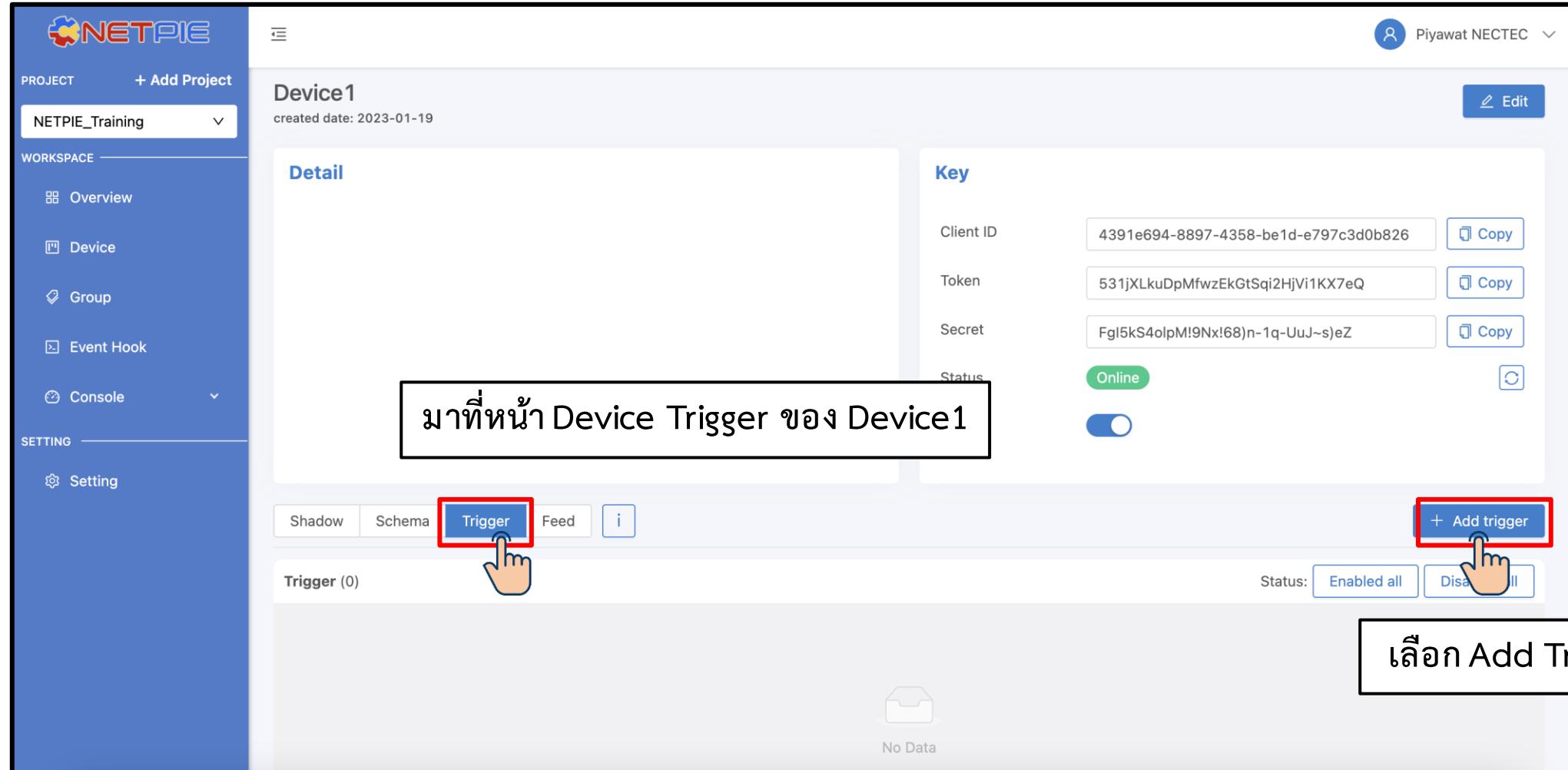
uri คือ Endpoint ปลายทางที่กำหนดว่าต้องการให้ส่งไปที่ใด

Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

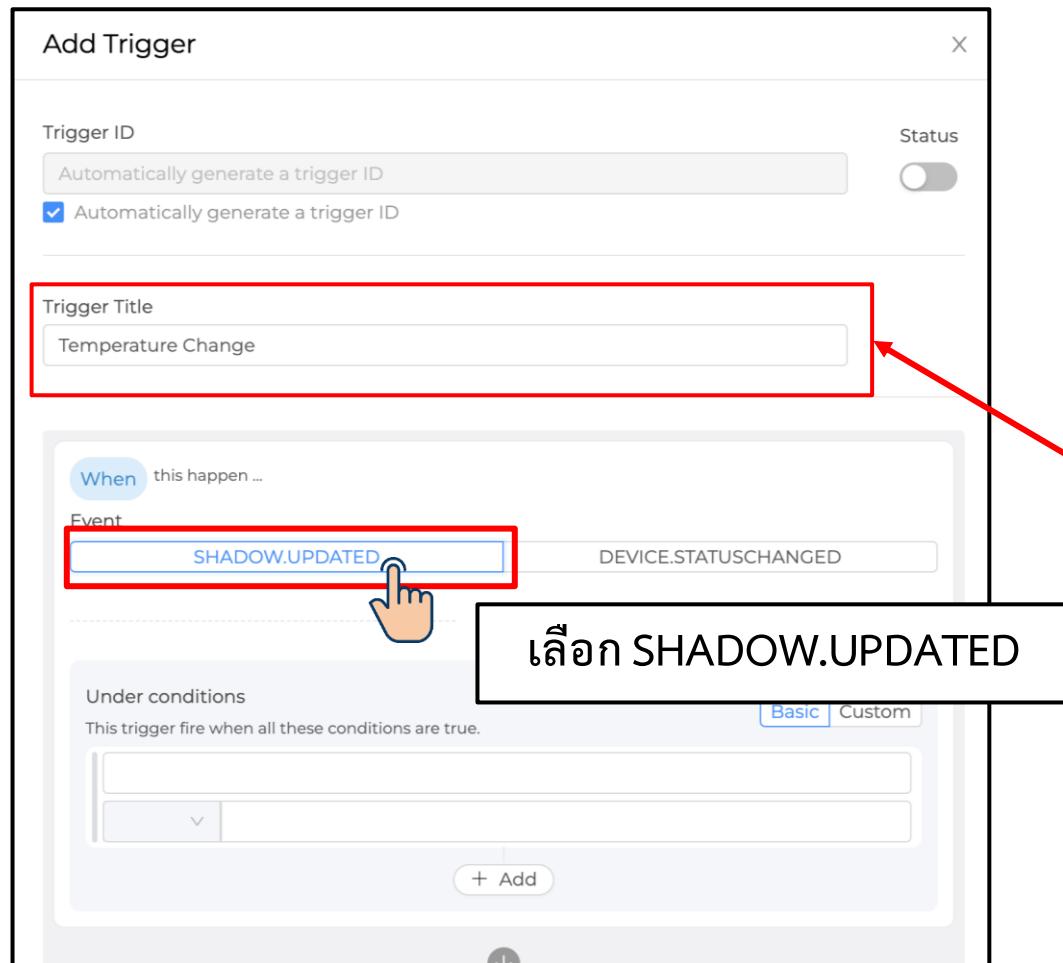


Trigger : SHADOW UPDATE & DEVICE.STATUSCHANGE

Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



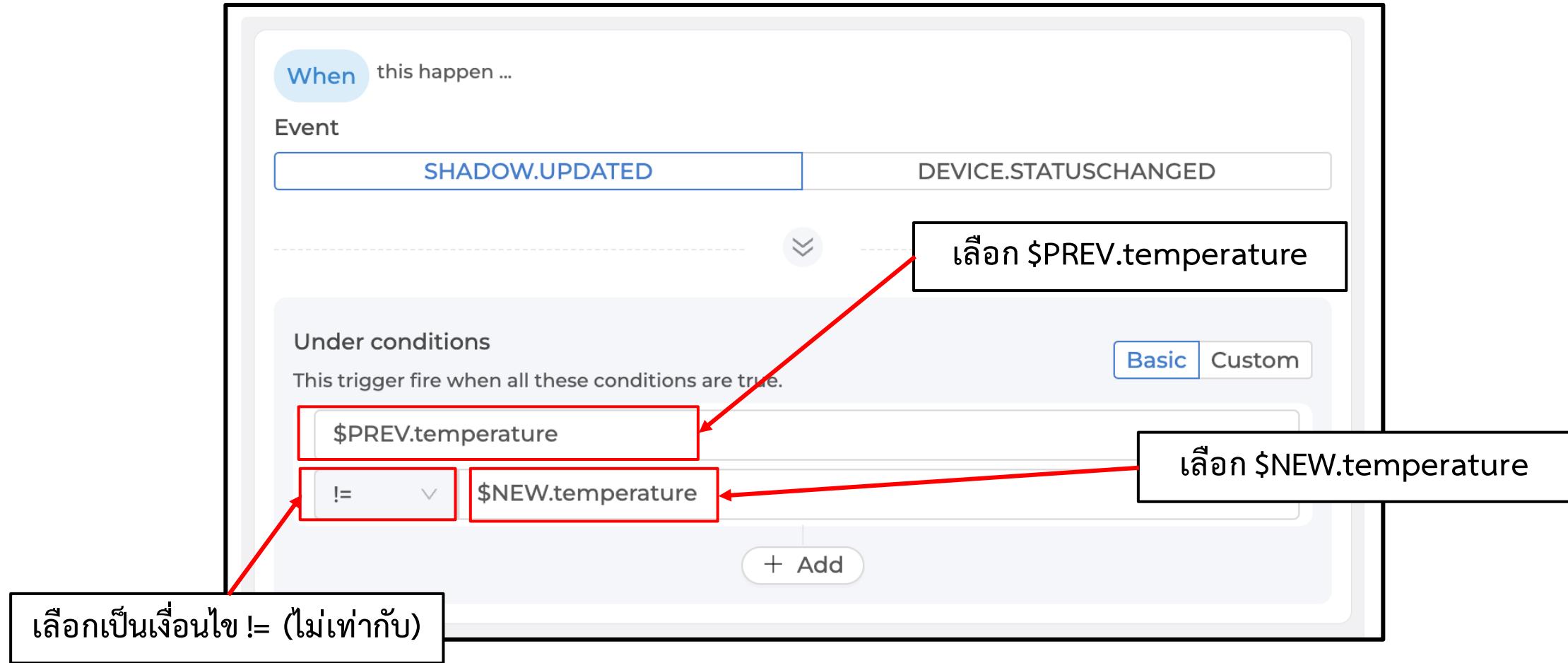
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



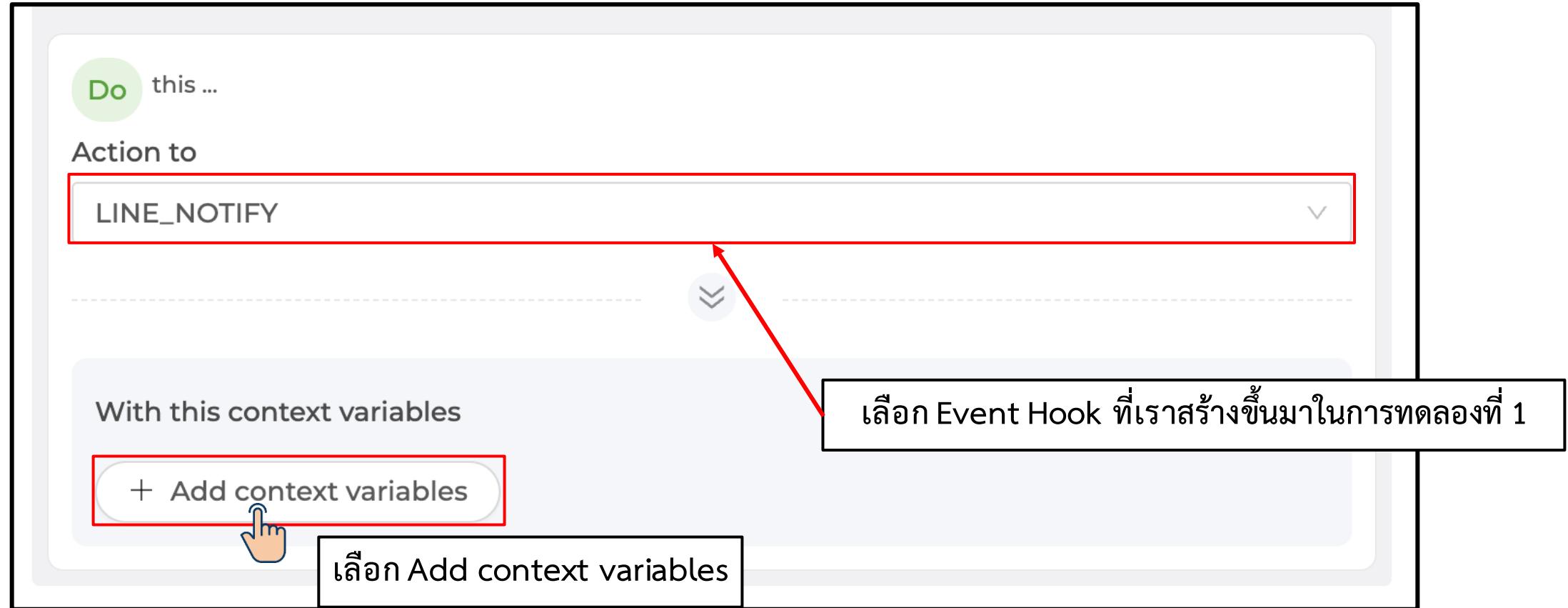
เราจะสร้างเงื่อนไขการแจ้งเตือน คือ เมื่ออุณหภูมิมีการเปลี่ยนแปลง
ให้ทำการแจ้งเตือนเข้า Line Notify โดยมีข้อความคือ
"Temperature is change from PREV.Temperature °C to
NEW.Temperature °C"

ตั้งชื่อ Trigger

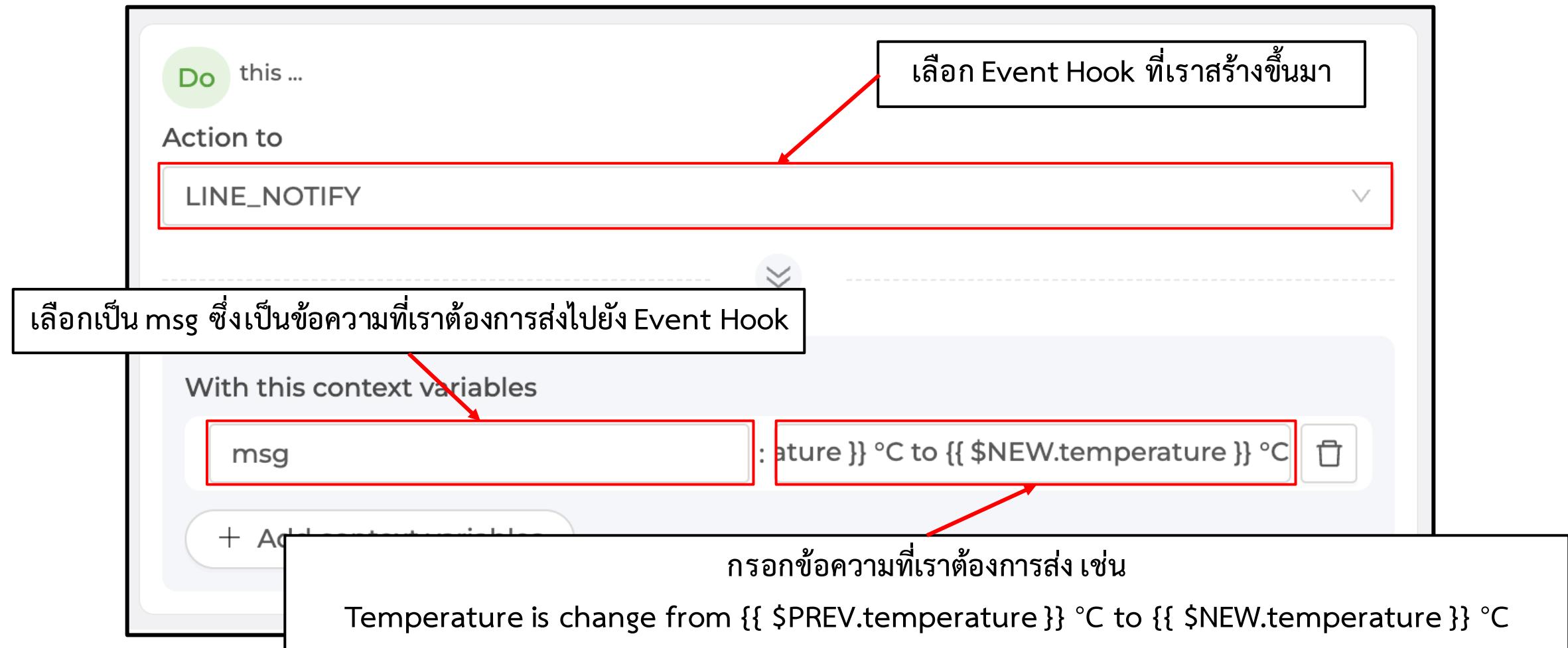
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



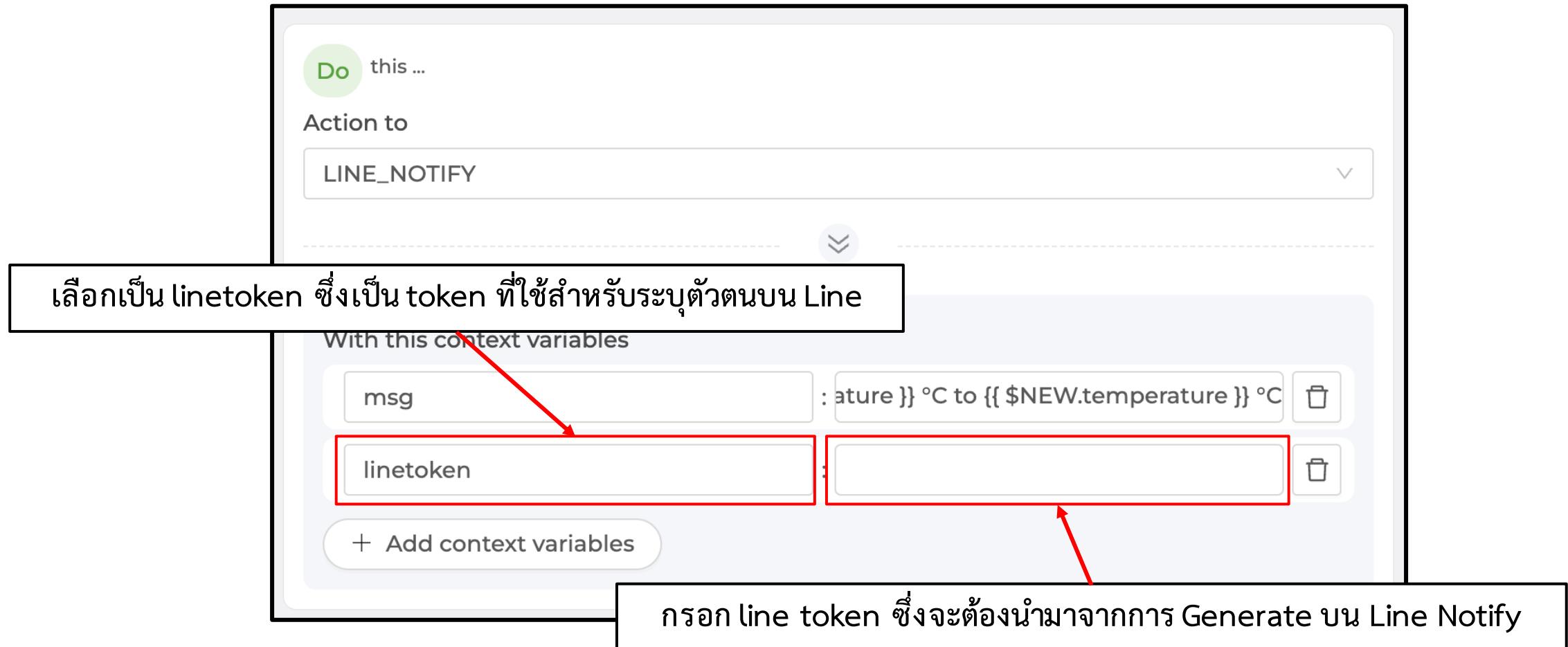
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



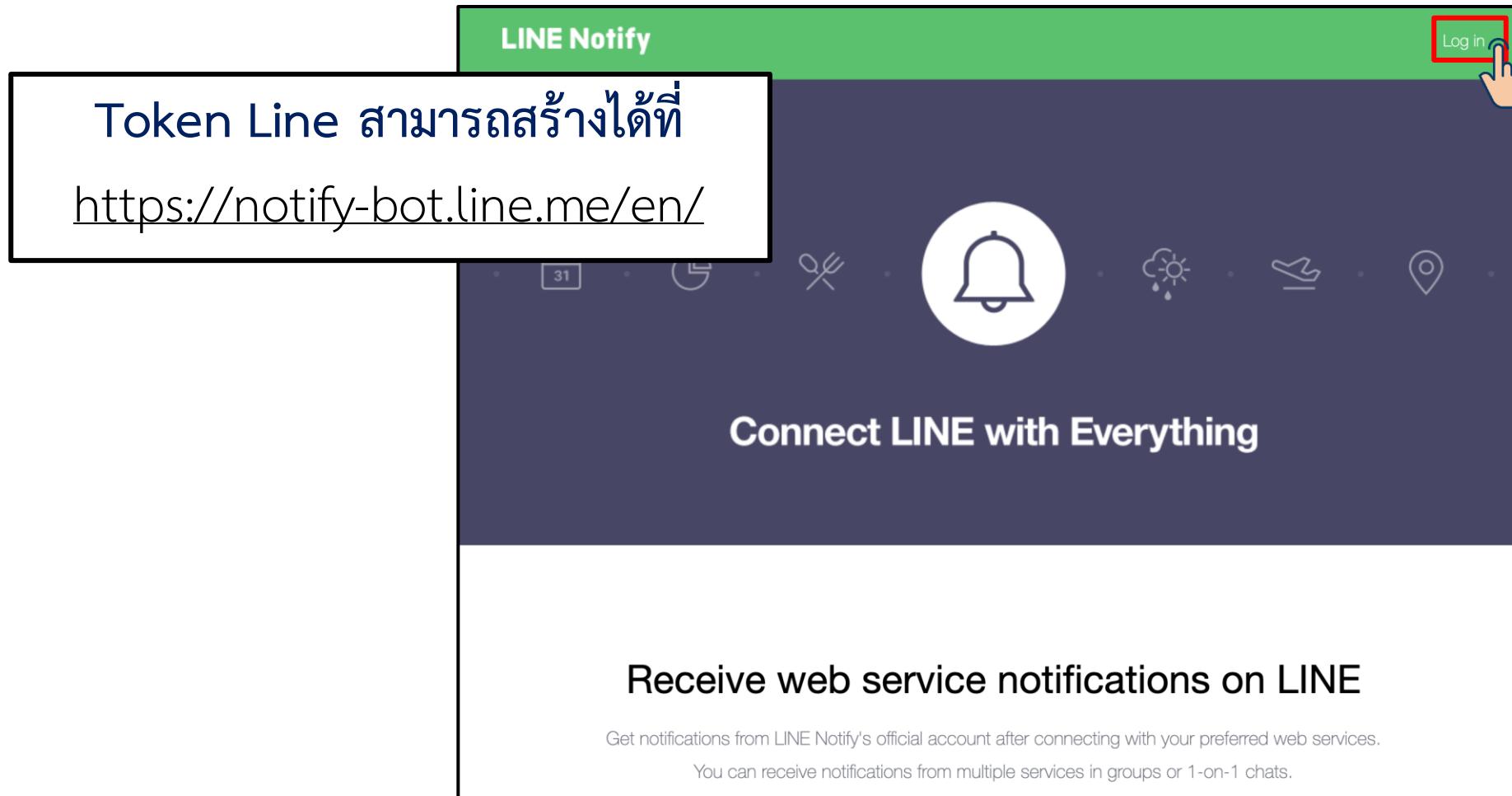
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



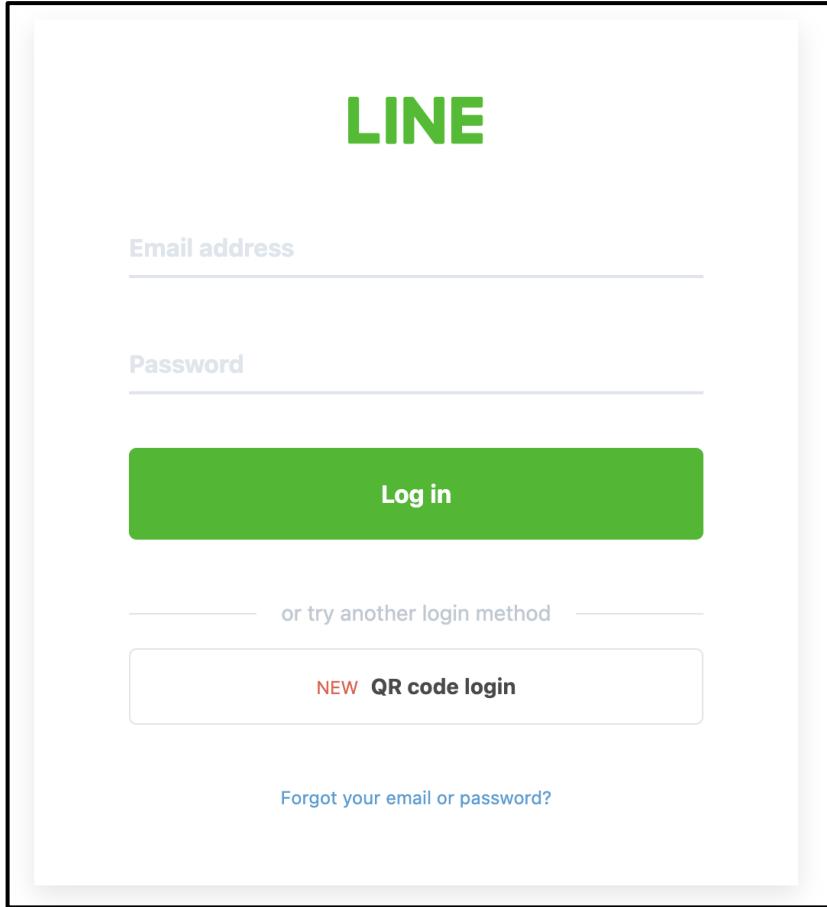
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

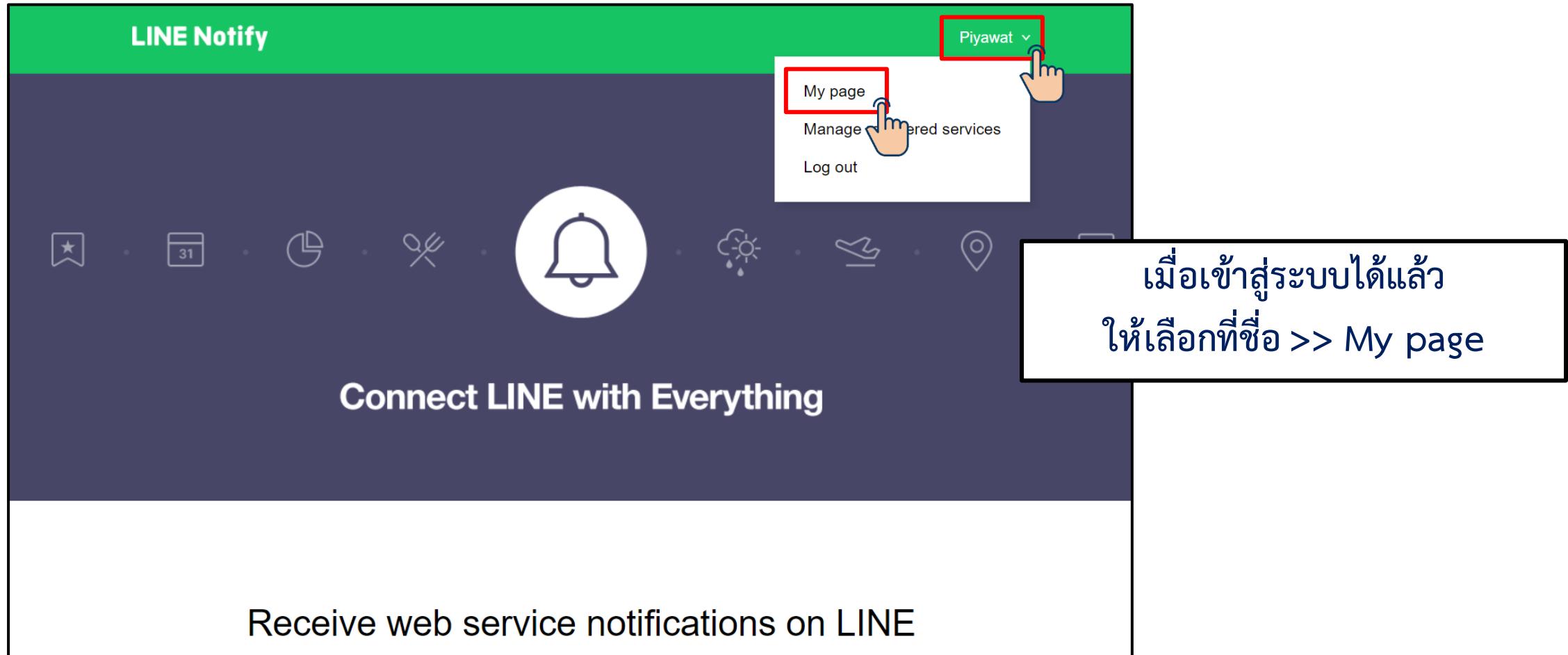


Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Login เข้าสู่ระบบโดยใช้
E-mail และ Password ของ LINE ที่ใช้งาน

Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

Generate access token (For developers)

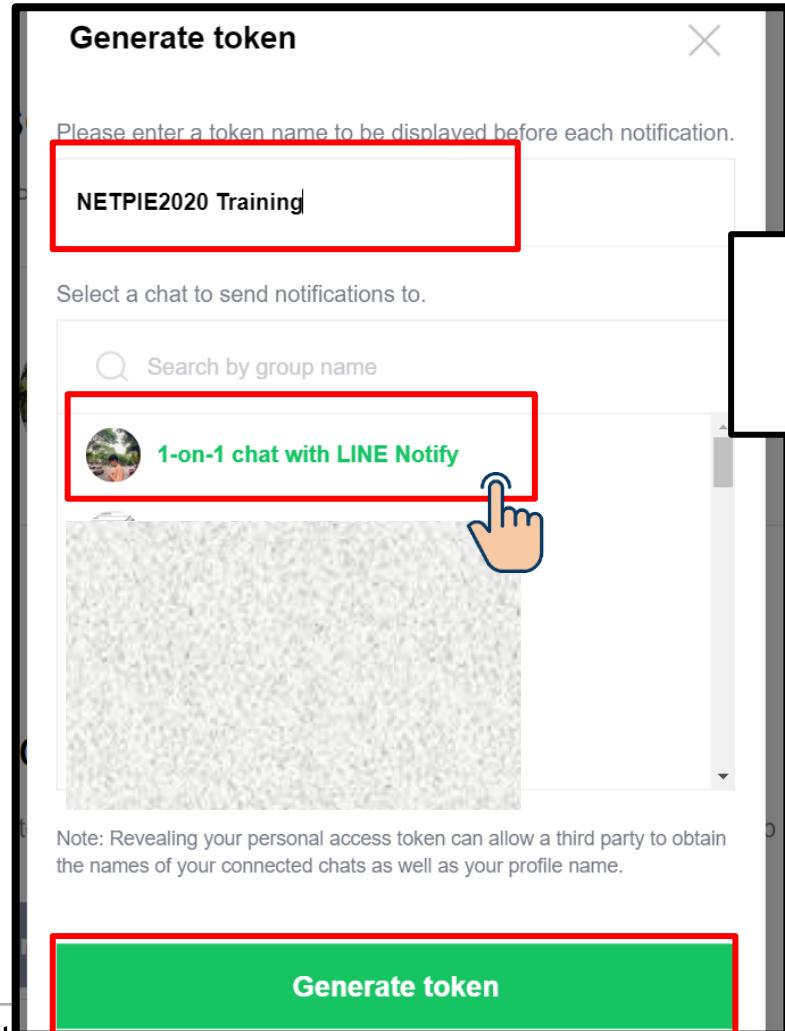
By using personal access tokens, you can configure notifications without having to add a web service.

Generate token

LINE Notify API Document

เลื่อนลงมาข้างล่างแล้วเลือก
Generate Token

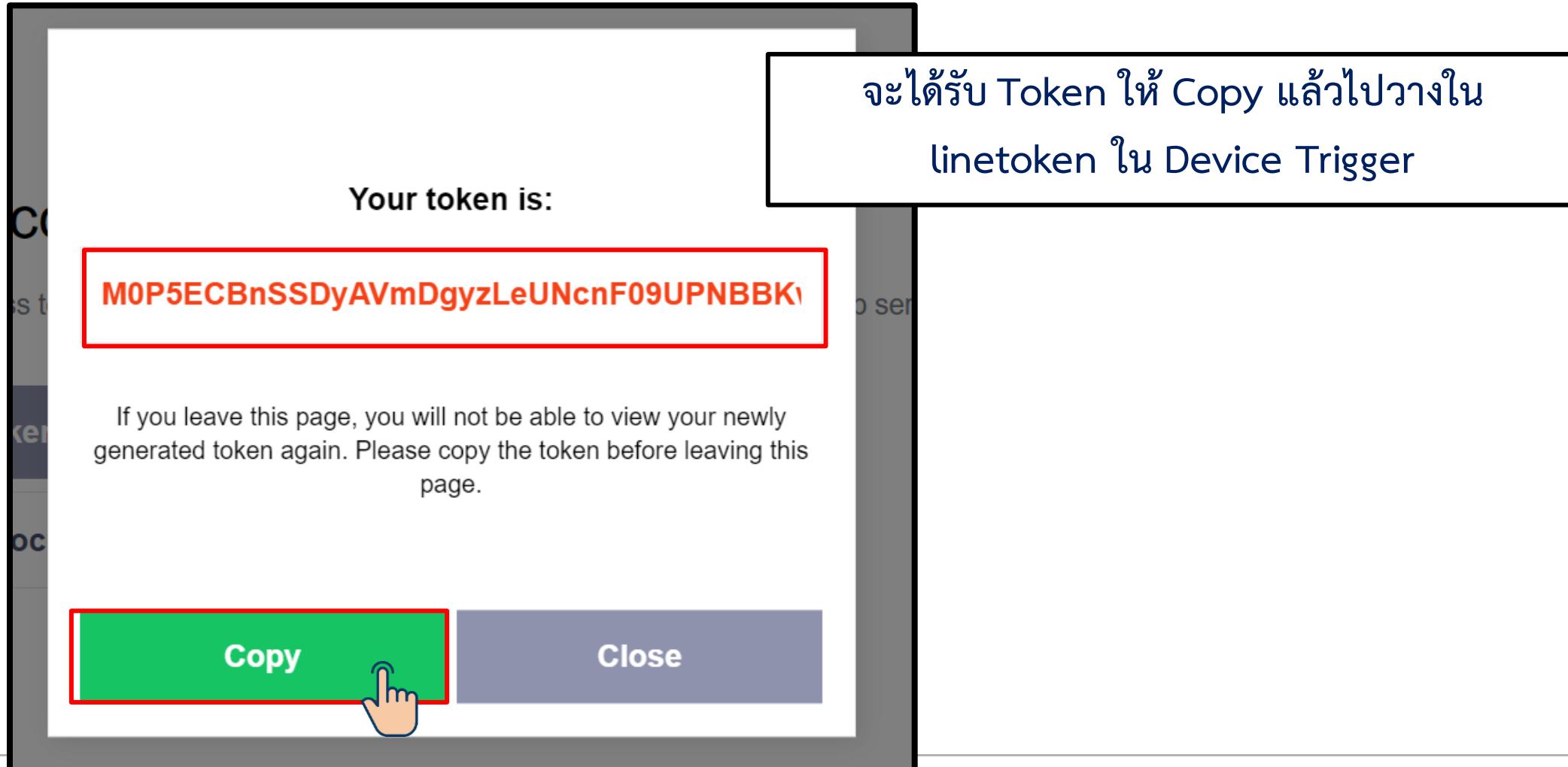
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ตั้งชื่อหัวข้อ การสนทนา

แล้วเลือกแชทกับตัวเอง และคลิก Generate Token

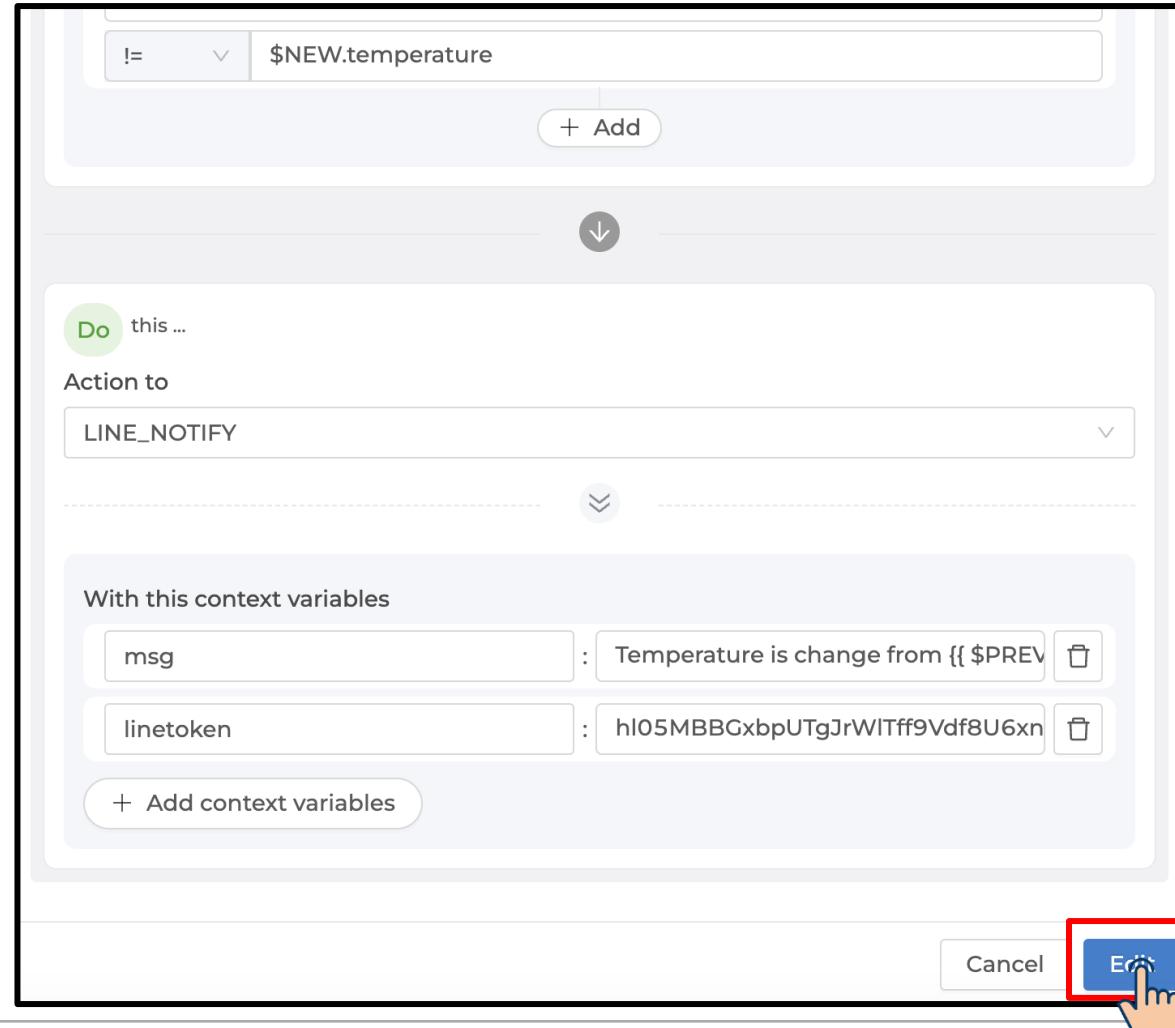
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



เมื่อสร้างทั้งหมดเสร็จแล้วทำการเลือก OK

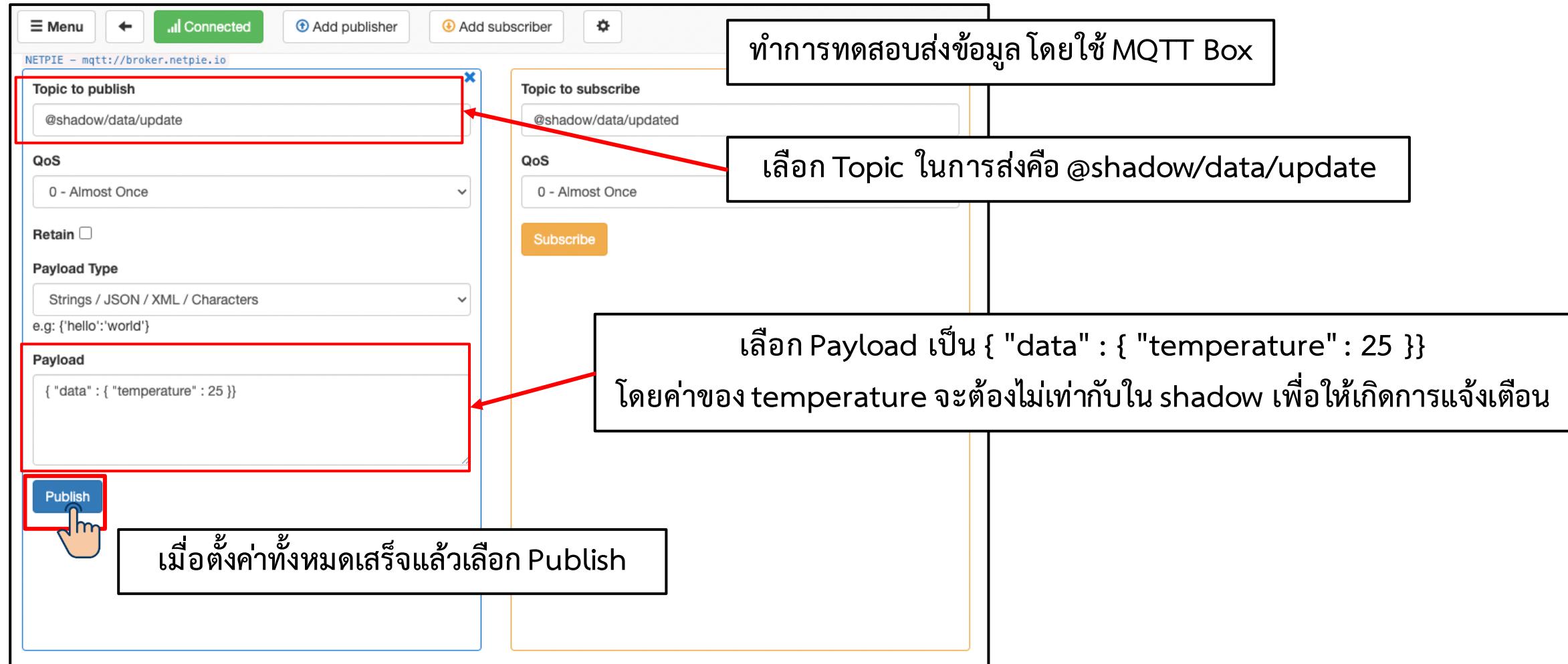
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

The screenshot shows the NETPIE Platform interface. On the left, there's a sidebar with 'PROJECT' (NETPIE_Training selected), 'WORKSPACE' (Overview, Device, Group, Event Hook, Console, Freeboard, Dashboard (New)), and 'SETTING' (Setting). The main area has tabs for 'Shadow', 'Schema', 'Trigger' (selected), and 'Feed'. A button '+ Add trigger' is visible. Below the tabs, it says 'Trigger (1)' and 'Status: Enabled all'. A specific trigger named 'Temperature Change' is shown with an ID of TD949588993653. The trigger condition is 'Event : SHADOW UPDATED Under conditions : \$PREV.temperature != \$NEW.temperature'. The action is set to 'LINE_NOTIFY' with the message: 'msg : Temperature is change from {{ \$PREV.temperature }} °C to {{ \$NEW.temperature }} °C'. The token is listed as 'linetoken : hI05MBBGxbpUTgJrWITff9Vdf8U6xnadLgPY2nytwVs'. A red box highlights the 'Status' toggle switch for this trigger, which is currently off. A hand cursor icon is pointing at this switch. A callout box with the text 'ทำการเลือก Status เป็น Enable' (Select Status to be Enabled) is positioned over the trigger card.

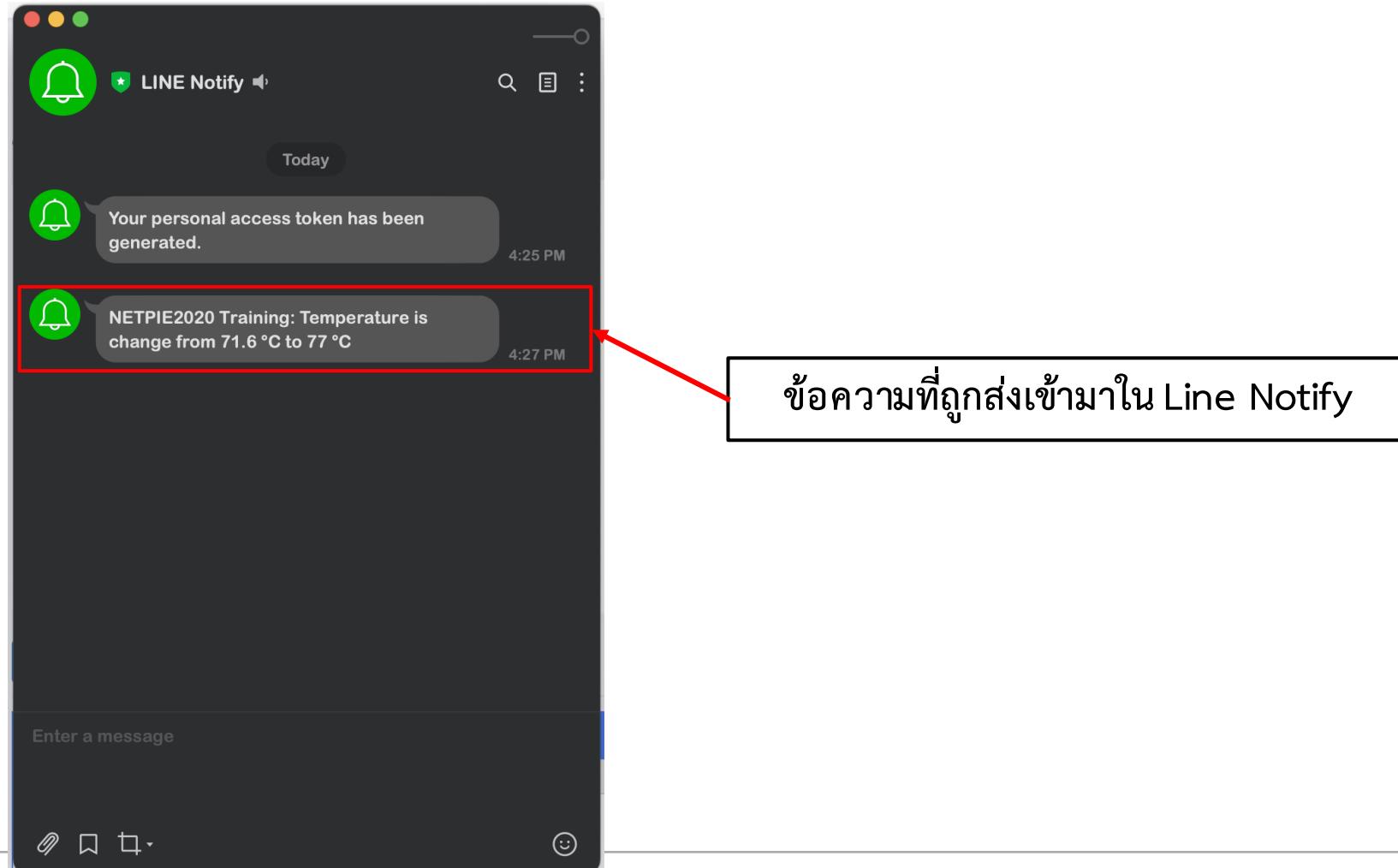
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

The screenshot shows the NETPIE Platform interface. On the left, the sidebar includes 'PROJECT' (NETPIE_Training), 'WORKSPACE' (Overview, Device, Group, Event Hook, Console, Freeboard, Dashboard), and 'SETTING' (Setting). The main area displays 'Device1' (created date: 2023-01-19) with a 'Detail' tab and a 'Key' section containing Client ID, Token, Secret, Status (Online), and Enable (switched on). Below this is a configuration panel with tabs for Shadow, Schema, Trigger, Feed, and an info icon. A red box highlights the 'Trigger' tab, which contains the text: 'ตรวจสอบข้อมูล temperature ก่อนส่งค่าเพื่อทดสอบ Line Notify'. The 'Feed' tab shows a list of sensor values: humidity: 65, light: 120, place: Piyawat Home, and temperature: 68.

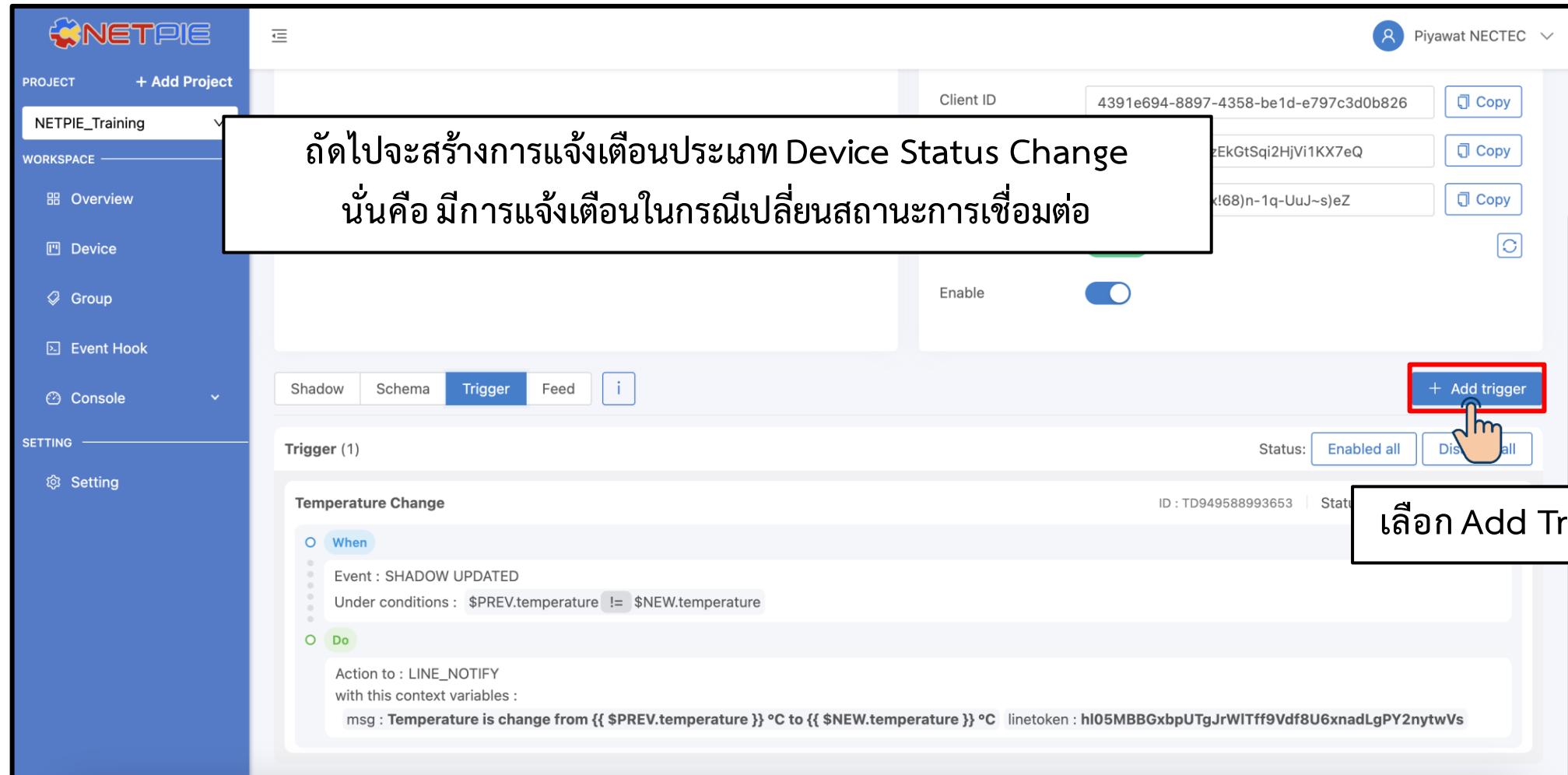
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



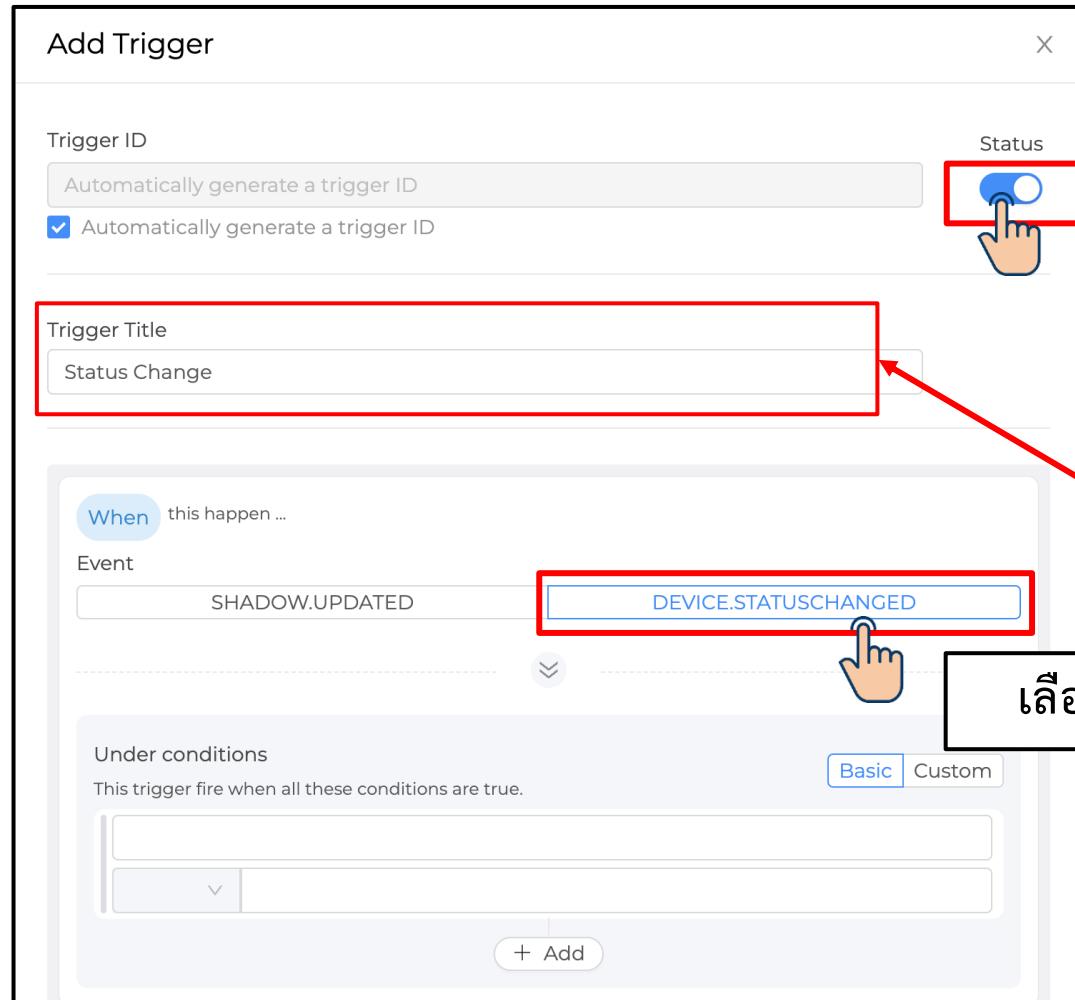
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

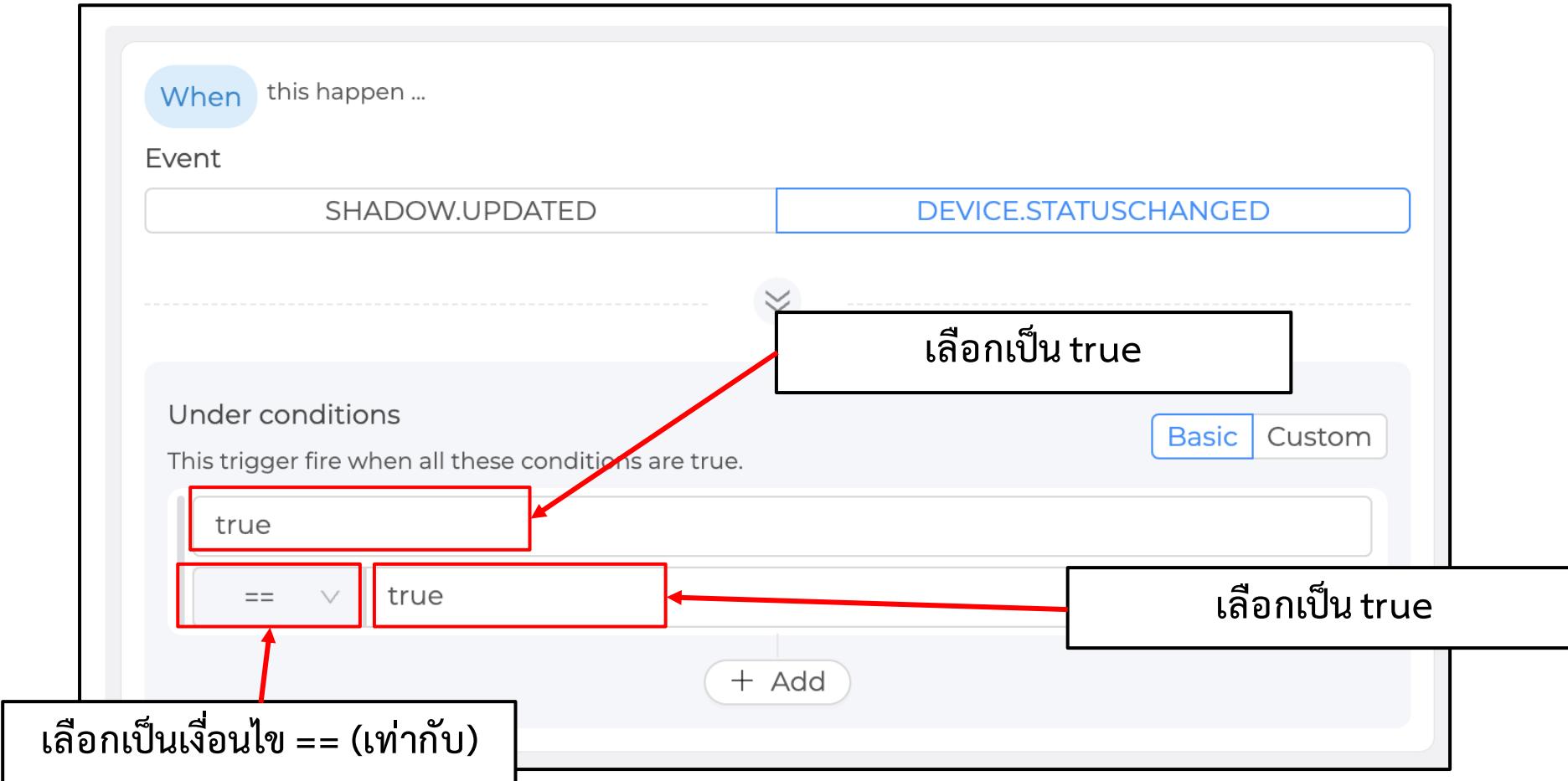


เราจะสร้างเงื่อนไขการแจ้งเตือน คือ เมื่อมีการเปลี่ยนสถานะไม่ประจำเป็นรูปแบบใหม่ก็ตาม ให้แจ้งเตือนไปยัง Line Notify โดยมีข้อความคือ "Status change from OLD.Status to NEW.Status"

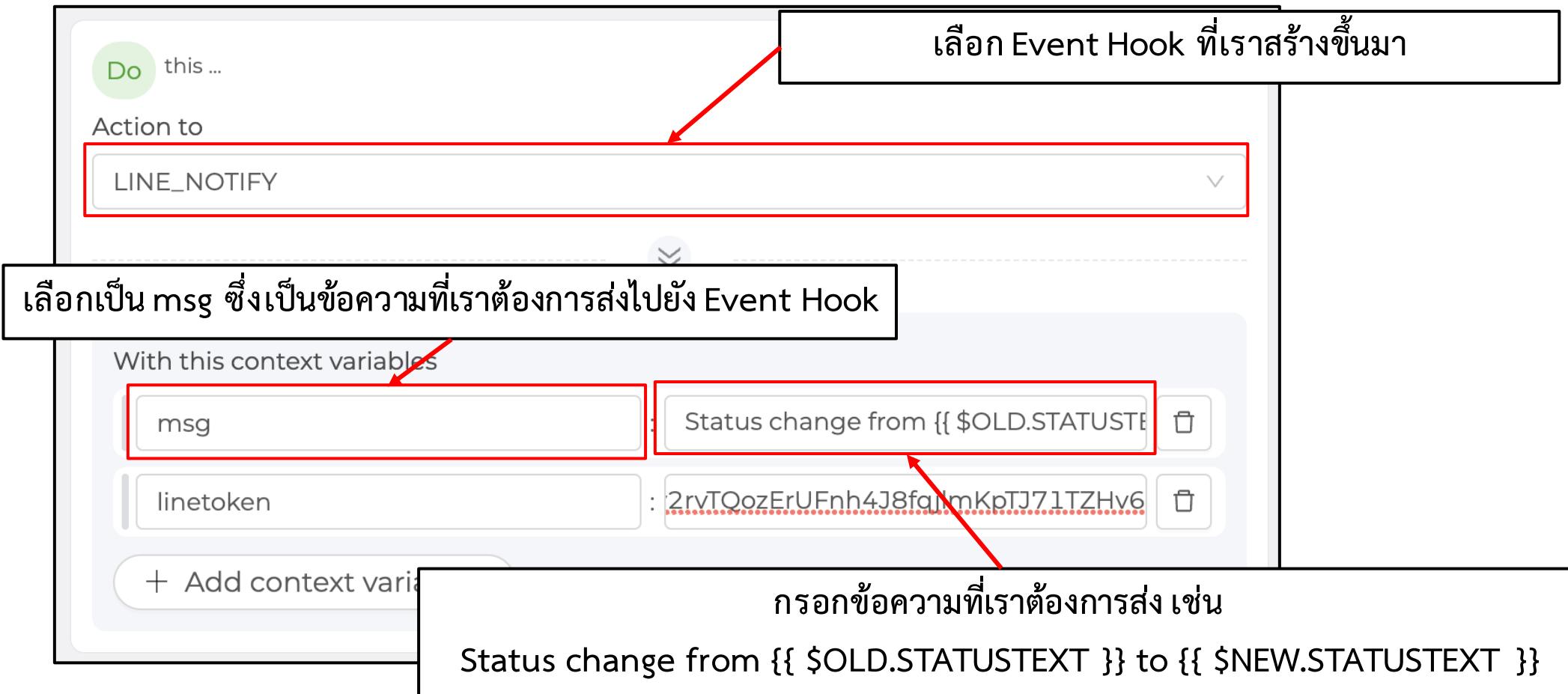
ตั้งชื่อ Trigger

เลือก DEVICE.STATUSCHANGED

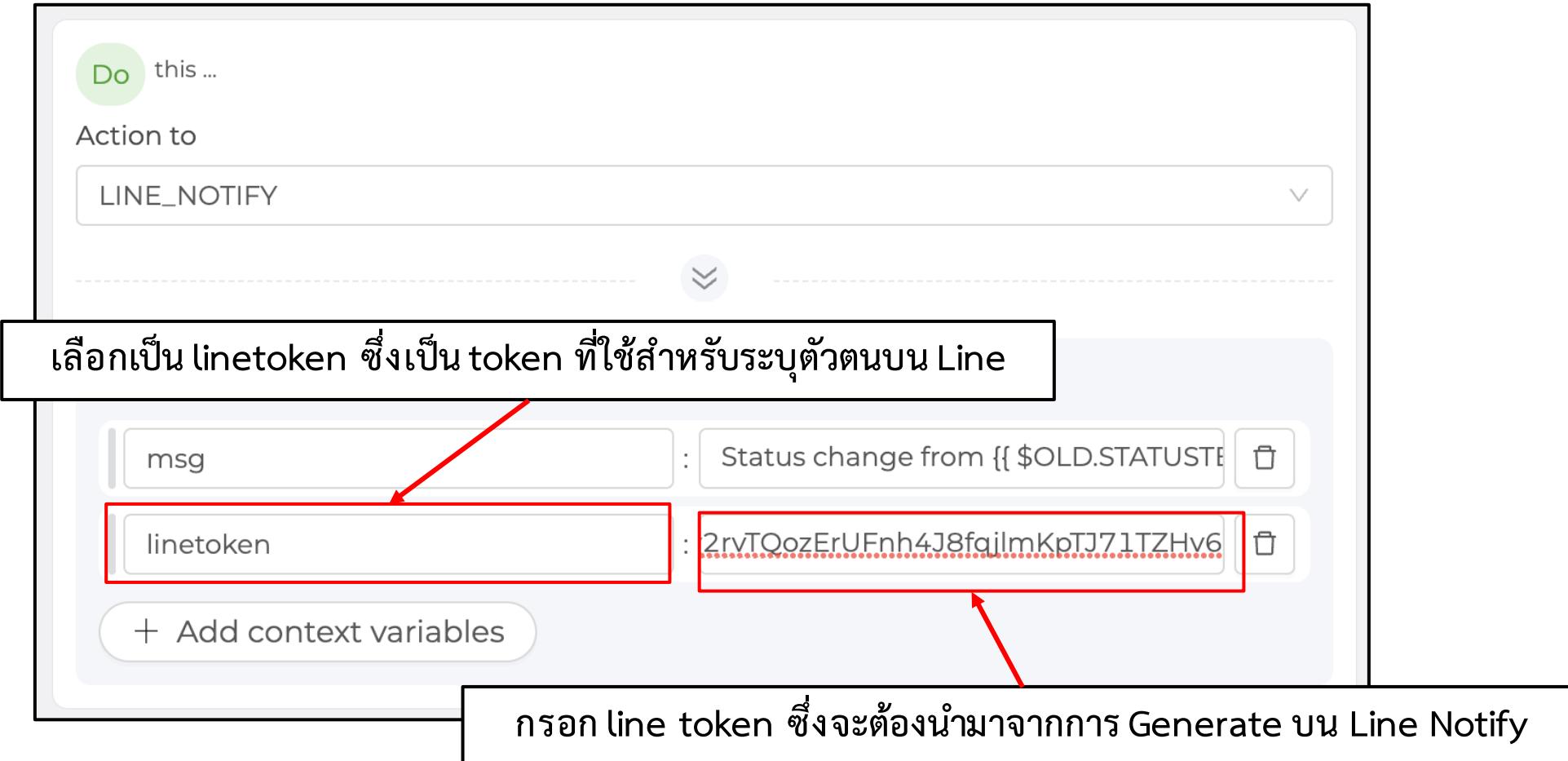
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



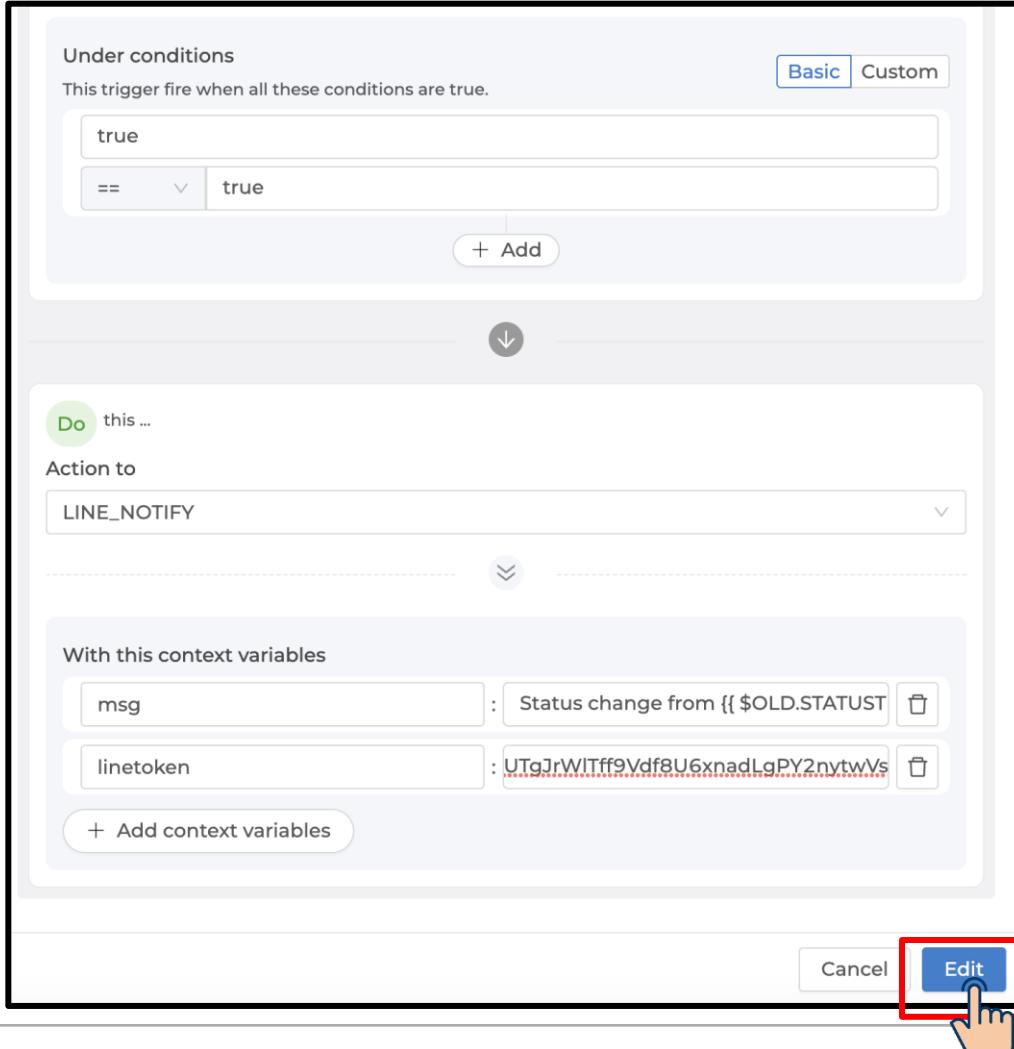
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



เมื่อสร้างทั้งหมดเสร็จแล้วทำการเลือก OK

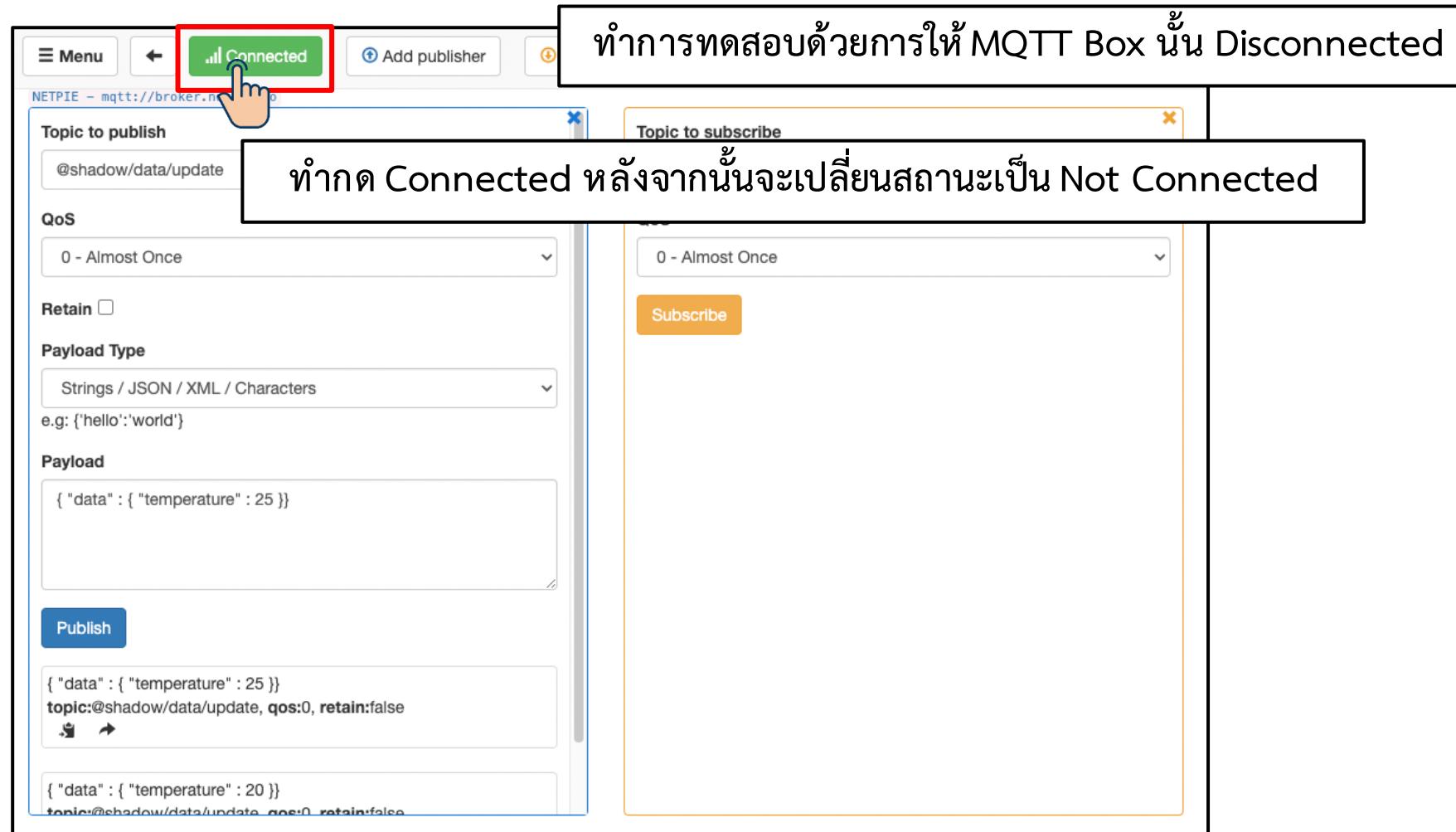
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

The screenshot shows the NETPIE Platform interface with the 'Trigger' tab selected. There are two triggers listed:

- Temperature Change**:
 - When**: Event: SHADOW UPDATED, Under conditions: \$PREV.temperature != \$NEW.temperature
 - Do**: Action to: LINE_NOTIFY with context variables: msg : Temperature is change from {{ \$PREV.temperature }} °C to {{ \$NEW.temperature }} °C linetoken : hI05MBBGxbpUTgJrWITff9Vdf8U6xnadLgPY2nytwVs
- Status Change**:
 - When**: Event: DEVICE STATUSCHANGED, Under conditions: true == true
 - Do**: Action to: LINE_NOTIFY with context variables: msg : Status change from {{ \$OLD.STATUSTEXT }} to {{ \$NEW.STATUSTEXT }} linetoken : hI05MBBGxbpUTgJrWITff9Vdf8U6xnadLgPY2nytwVs

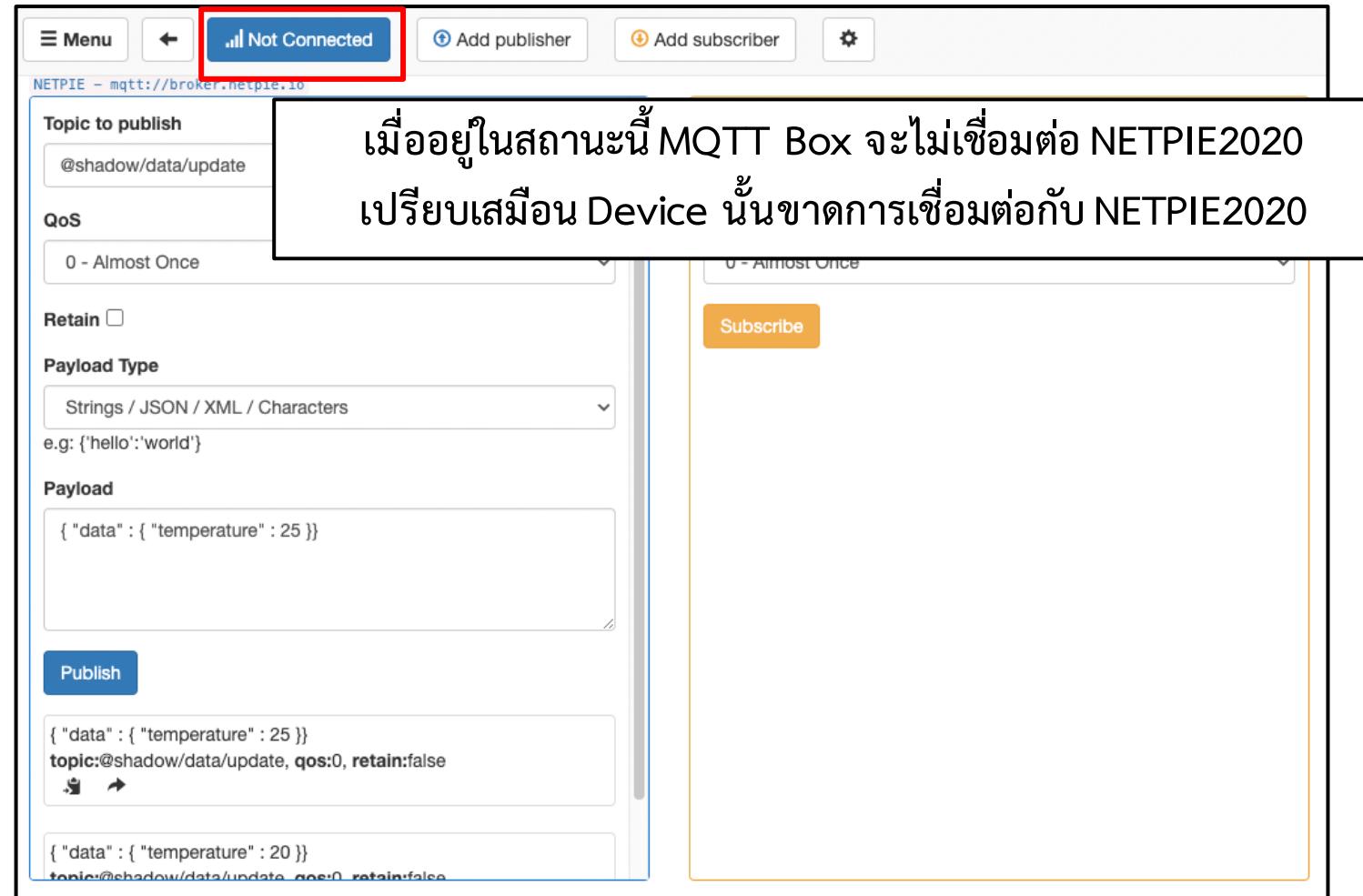
A red box highlights the second trigger, and a callout box points to it with the text "Trigger ที่สร้างขึ้นมาใหม่".

Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

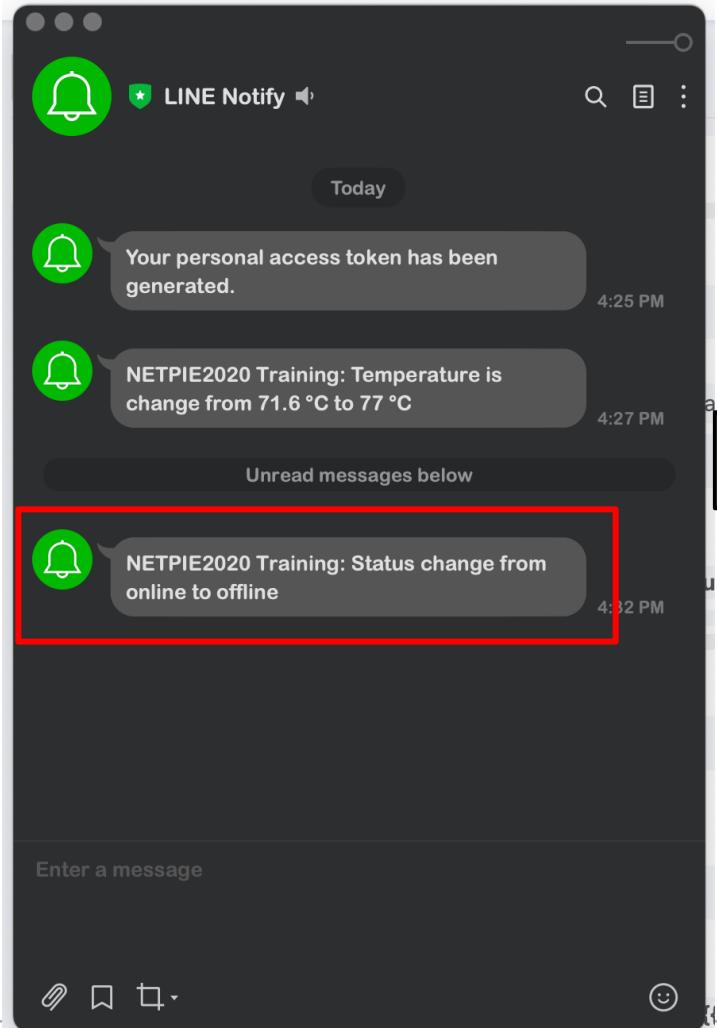


4 – NETPIE2020

Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

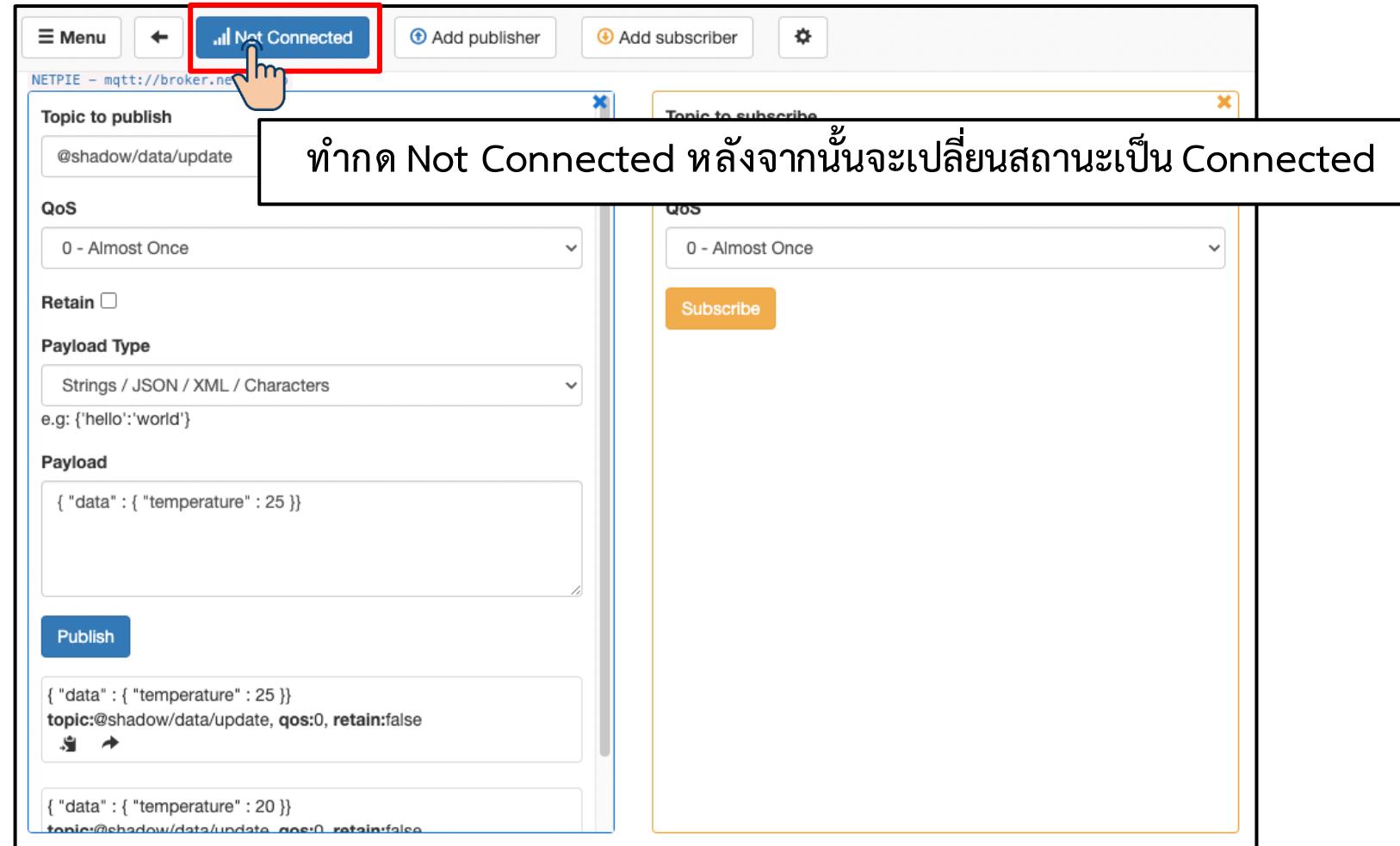


Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

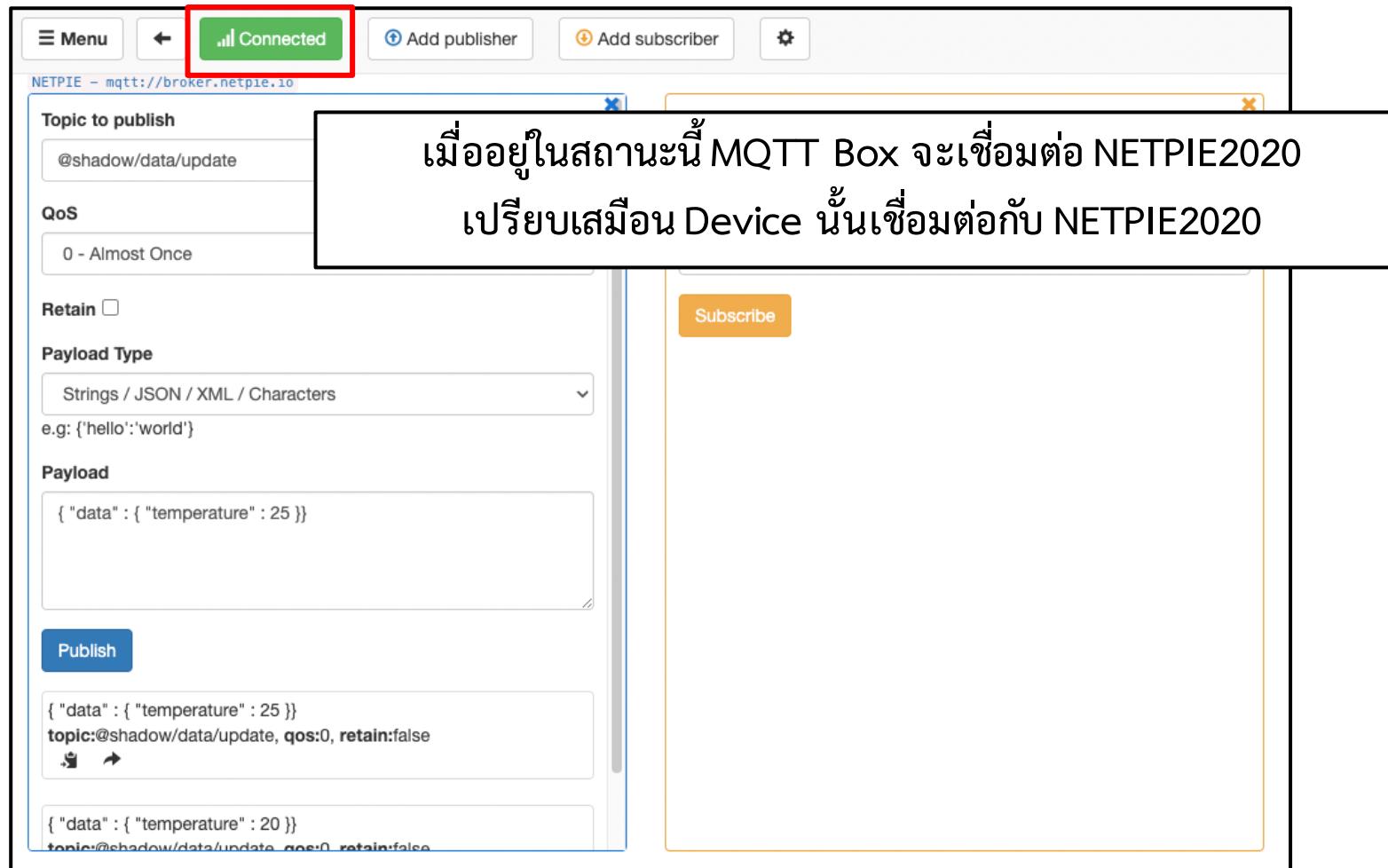


ข้อความการแจ้งเตือนสถานะที่เปลี่ยนไปถูกส่งมายัง Line

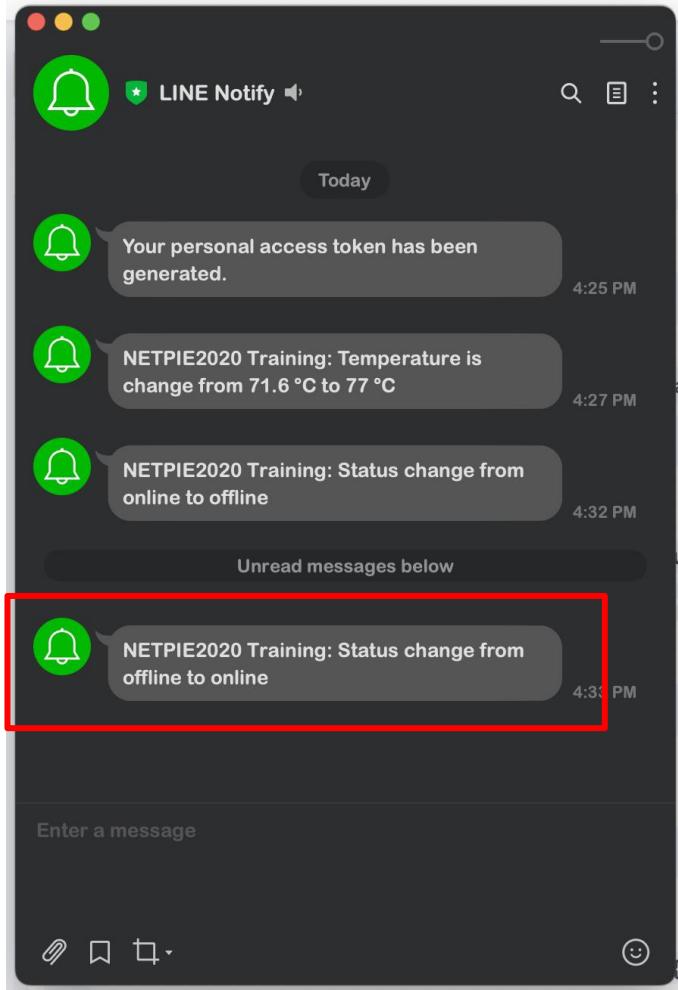
Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

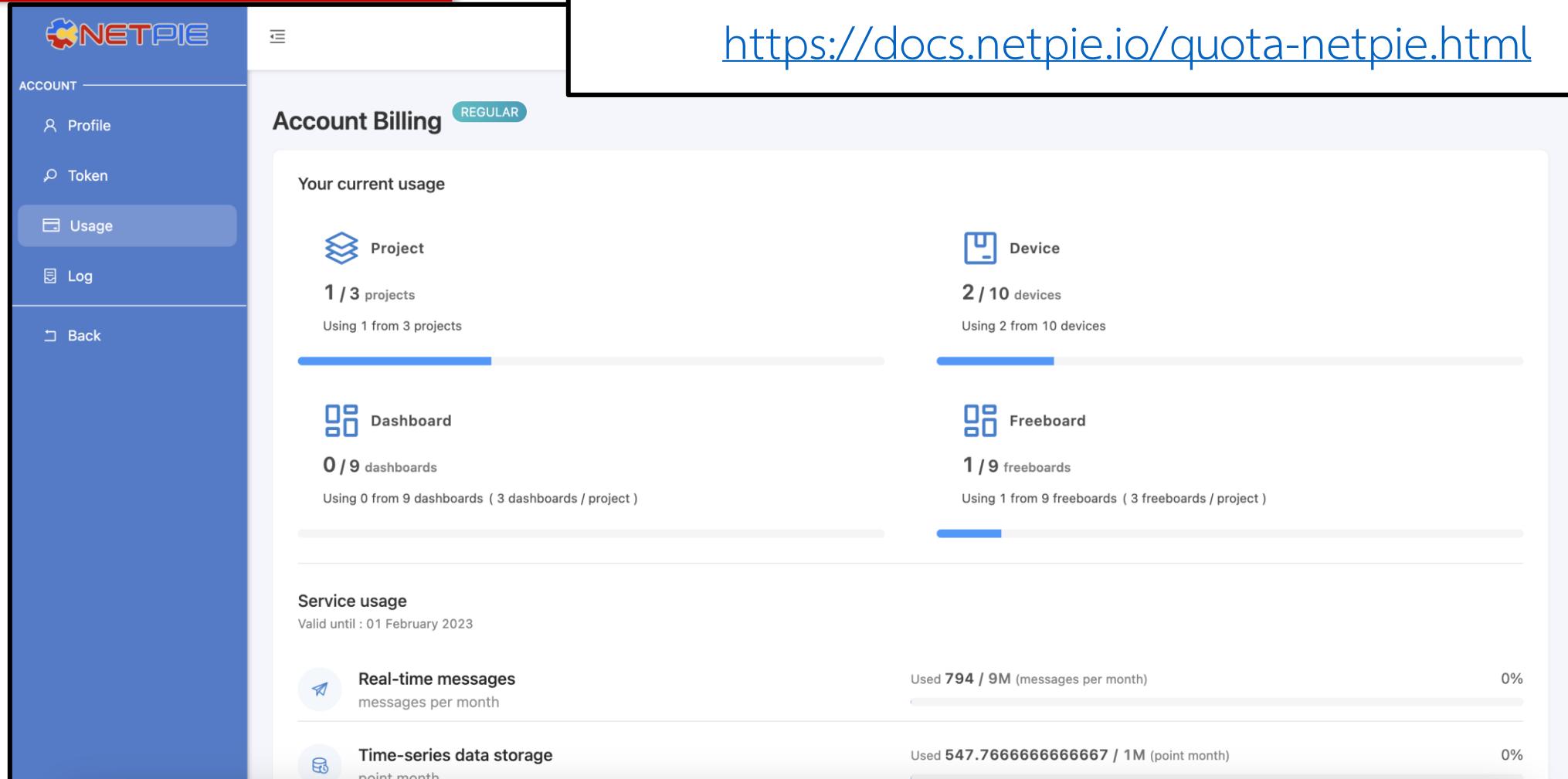


Example การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ข้อความการแจ้งเตือนสถานะที่เปลี่ยนไปถูกส่งมายัง Line

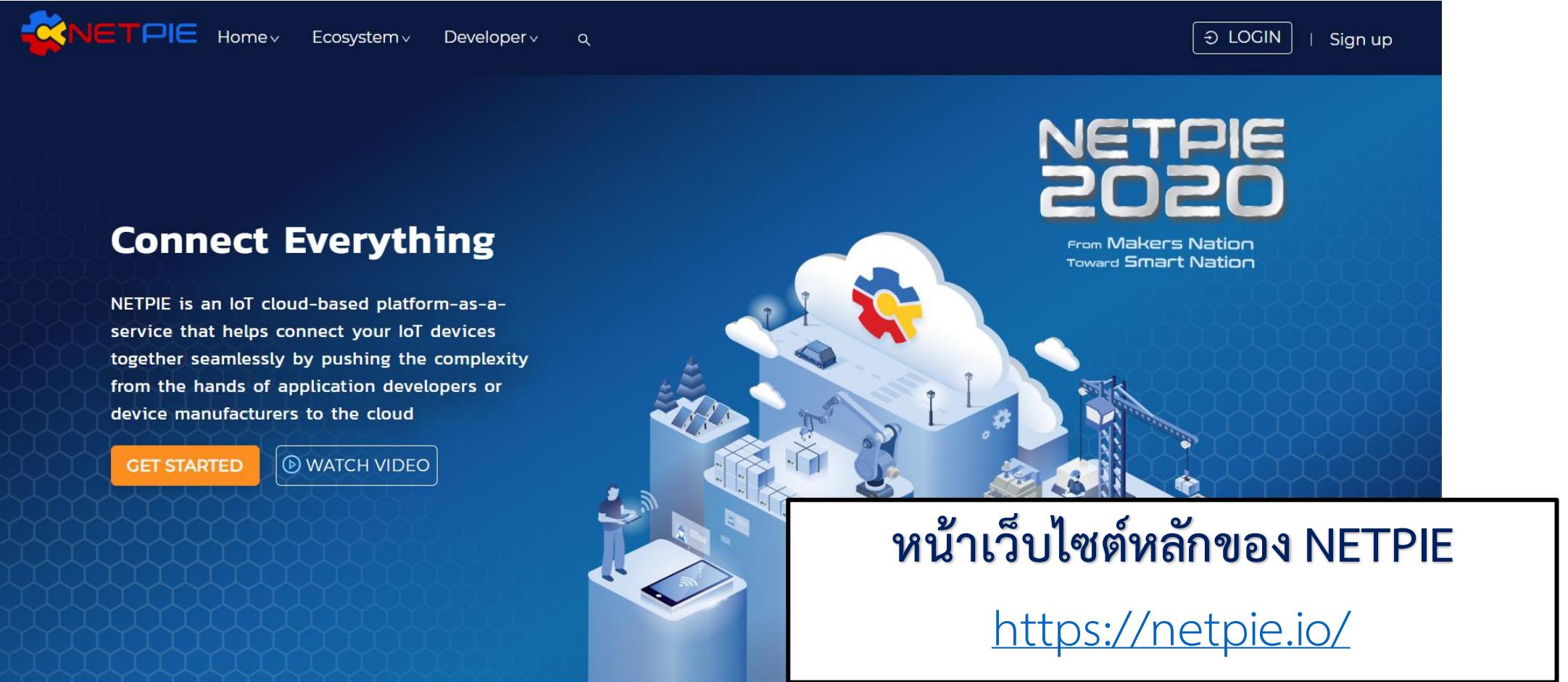
NETPIE2020 QUOTA



ศึกษาการคำนวณการใช้งานต่างๆได้

<https://docs.netpie.io/quota-netpie.html>

NETPIE Community



The image shows the NETPIE 2020 website homepage. At the top, there is a dark blue header with the NETPIE logo, navigation links for Home, Ecosystem, Developer, and a search bar, along with LOGIN and Sign up buttons. The main background is a blue hexagonal pattern. On the left, the text "Connect Everything" is displayed above a paragraph describing NETPIE as an IoT cloud-based platform-as-a-service. Below this are two buttons: "GET STARTED" and "WATCH VIDEO". On the right, the text "NETPIE 2020" is prominently displayed in large white letters, with the subtitle "From Makers Nation Toward Smart Nation" underneath. A central graphic features a white cloud containing a red and yellow gear icon, with various IoT devices like a car, a crane, and a robot connected to it via lines.

หน้าเว็บไซต์หลักของ NETPIE

<https://netpie.io/>

4 – NETPIE2020

NETPIE Community

The screenshot shows the NETPIE Community website. The top navigation bar includes a Facebook icon, the text 'NETPIE', and a search bar. The main header features the text 'NETPIE 2020' and 'From Makers Nation Toward Smart Nation'. On the left, a sidebar lists categories such as 'เกี่ยวกับ' (About), 'พูดคุย' (Talk), 'การฝึกสอน' (Training), 'หน่วยการเรียนรู้' (Learning Unit), 'ประกาศ' (Announcement), 'สมาชิก' (Members), 'งานกิจกรรม' (Activities), 'วิดีโอ' (Videos), 'รูปภาพ' (Images), 'ไฟล์' (Files), 'ข้อมูลเชิงลึกเกี่ยวกับกลุ่ม' (Detailed information about the group), 'ปาร์ตี้รับชม' (Viewing party), 'ความคุณกลุ่ม' (Group's merit), and 'คุณภาพกลุ่ม' (Group's quality). Below the sidebar is a search bar labeled 'ค้นหาในกลุ่มนี้'. The main content area displays a large image of a factory or industrial setting with various machinery and a person holding a tablet. Below the image are buttons for 'เข้าร่วมแล้ว' (Joined), 'การแจ้งเตือน' (Notifications), and 'เพิ่มเติม' (More). A user profile for 'Panita Pongpaibool' is shown, featuring a photo, the text 'ได้เพิ่มรูปภาพ 11 ภาพลงในอัลบั้ม: NETPIE 2020**', and the date 'ผู้ดูแล · 24 กุมภาพันธ์'. At the bottom, there are sections for 'ประกาศ' (Announcements) and 'UserQuestions ...'.

Facebook Group ของ NETPIE

<https://www.facebook.com/groups/netpie/>

NETPIE Community

NEXPIE Home Developer S

NEXPIE IoT Platform

Internet of Things Experts in Thailand

GET STARTED



หากสิทธิ์ในการใช้ NETPIE ไม่เพียงพอ แล้วต้องการอัพเกรด
สามารถทำได้โดยการเปลี่ยนจาก NETPIE เป็น NEXPIE

<https://nexpie.io/>

Question ?



การใช้สร้างและใช้งาน Freeboard ในการอบรมครั้งนี้



การใช้งาน NETPIE Dashboard

