

หน่วยการเรียนรู้ที่ 5 การสร้างแอปพลิเคชันบน NETPIE2020



นายธงชัย ชาบุดศรี
แผนกวิชาเทคโนโลยีสารสนเทศ



วิทยาลัยเทคนิคชลบุรี

จุดประสงค์การเรียนรู้

1

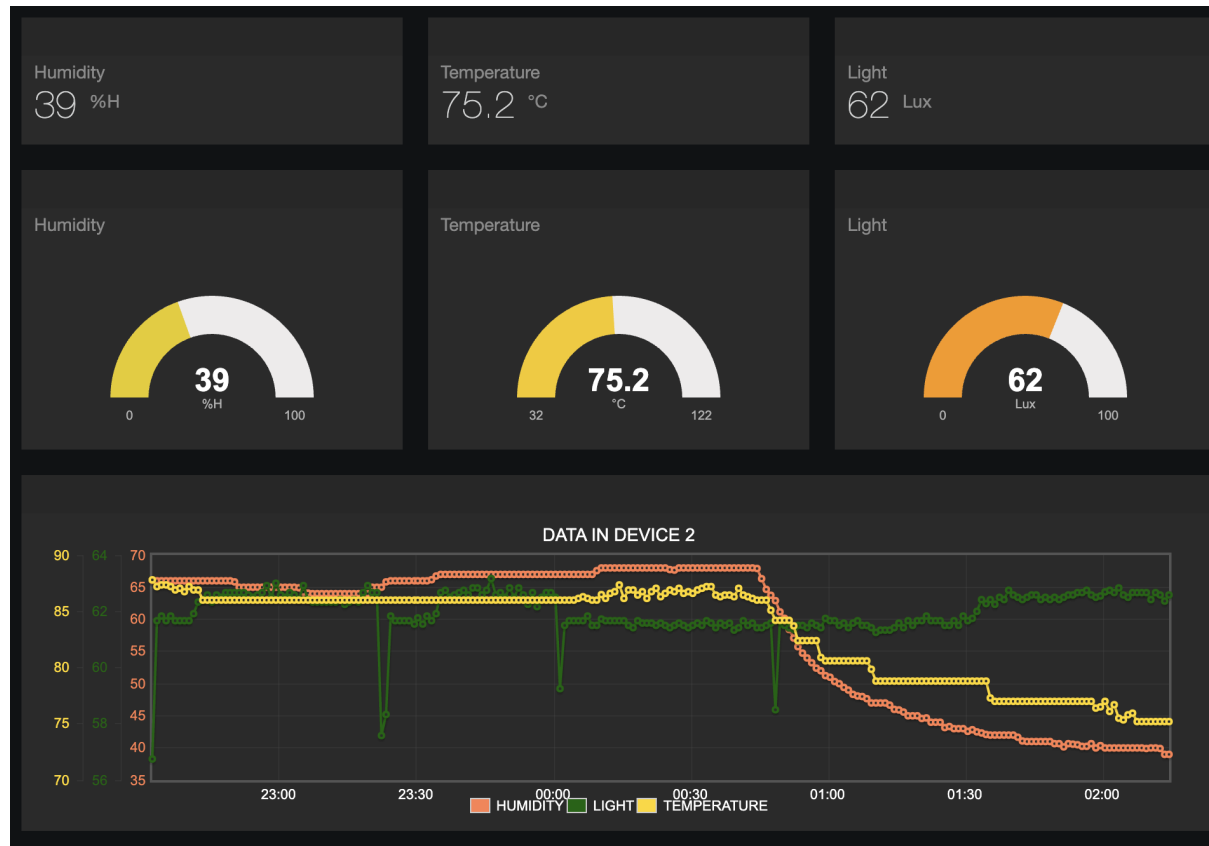
เรียนรู้การสร้างหน้าจอแสดงผลข้อมูลและควบคุม
Dashboard บน NETPIE2020

2

เรียนรู้การสร้างระบบการแจ้งเตือน Line Notify
บน NETPIE2020



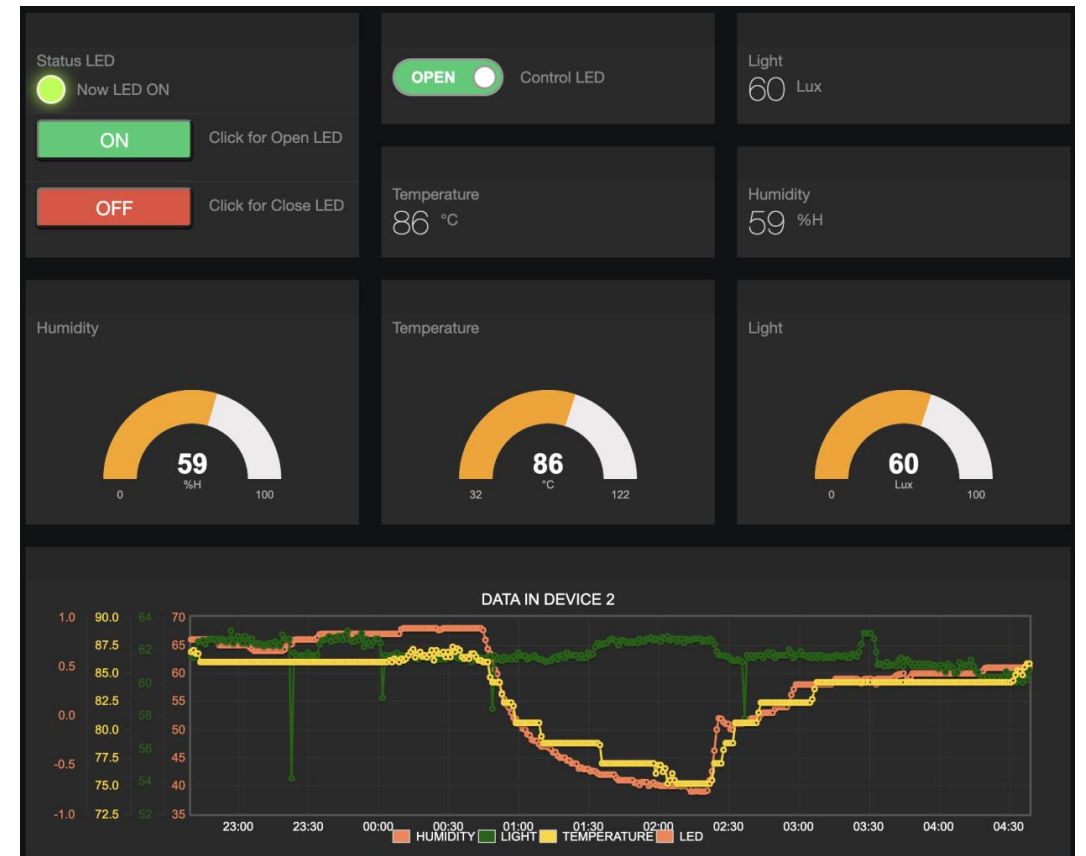
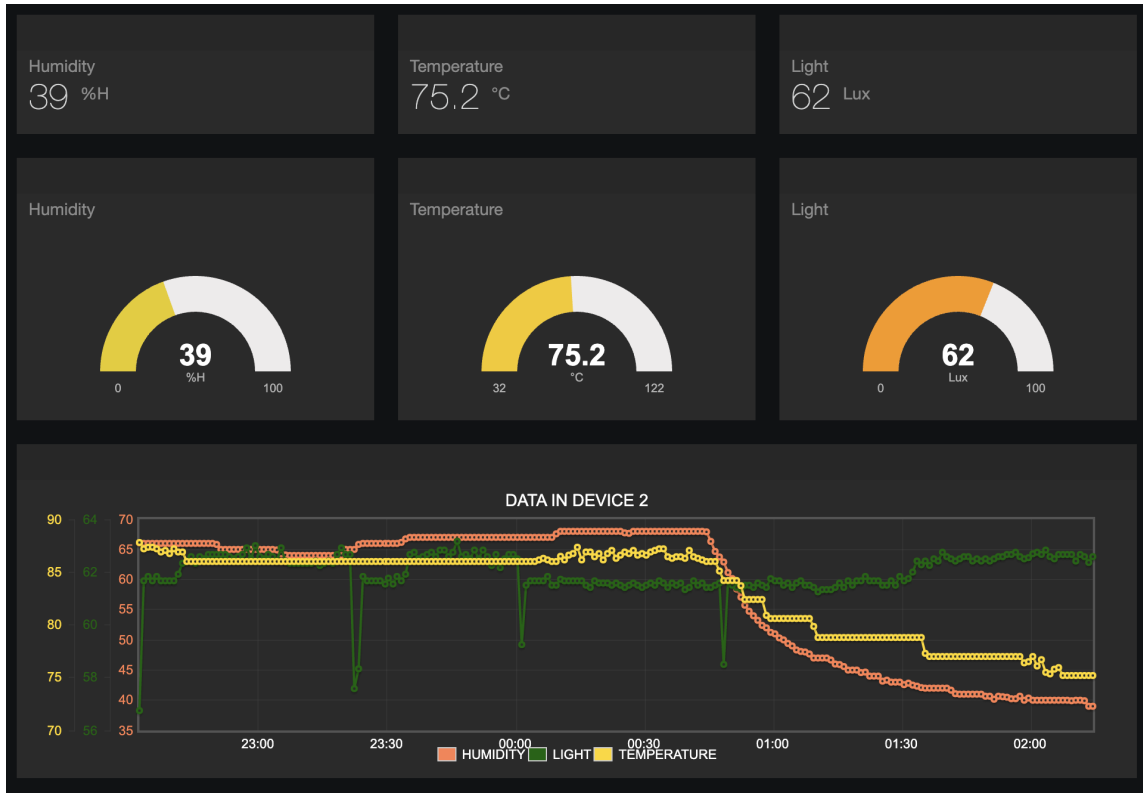
ใบงานที่ 5.1 การแสดงผลข้อมูลจากอุปกรณ์ด้วย Freeboard



ใบงานที่ 5.1 ทฤษฎีเบื้องต้น

Freeboard คืออะไร

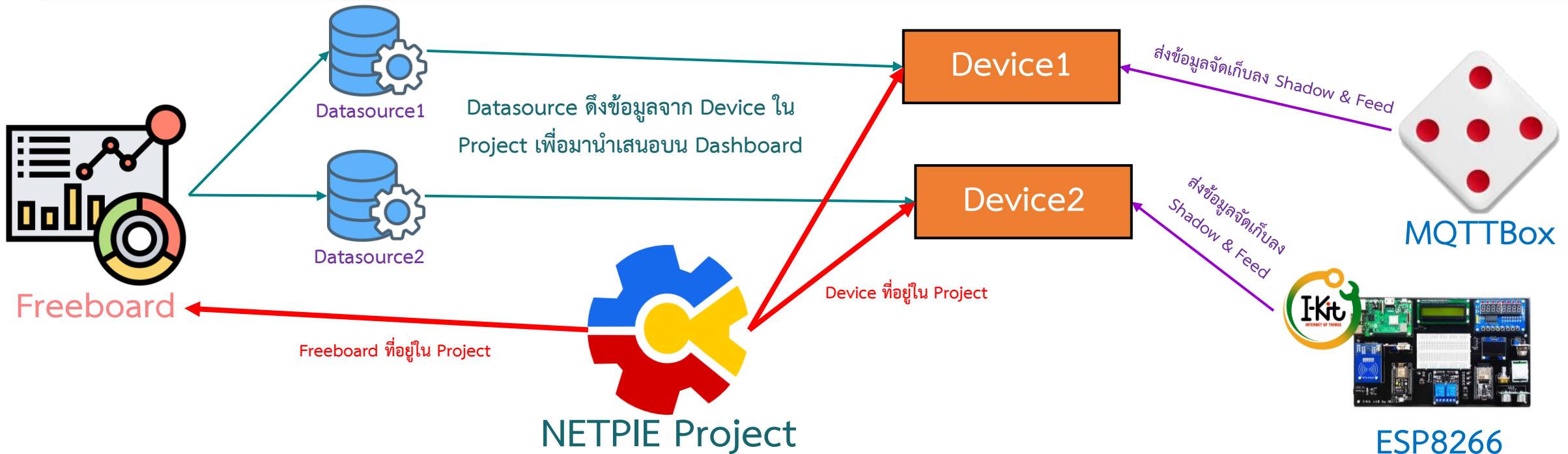
NETPIE Freeboard คือ Dashboard ที่ใช้สำหรับนำข้อมูลที่เก็บอยู่ใน Platform มาแสดงผลในรูปแบบต่างๆ เป็นเหมือนช่องทางให้ผู้ใช้สามารถติดตามหรือควบคุมการทำงานของ Device ของตัวเอง



ใบงานที่ 5.1 ทฤษฎีเบื้องต้น

Datasource บน Freeboard

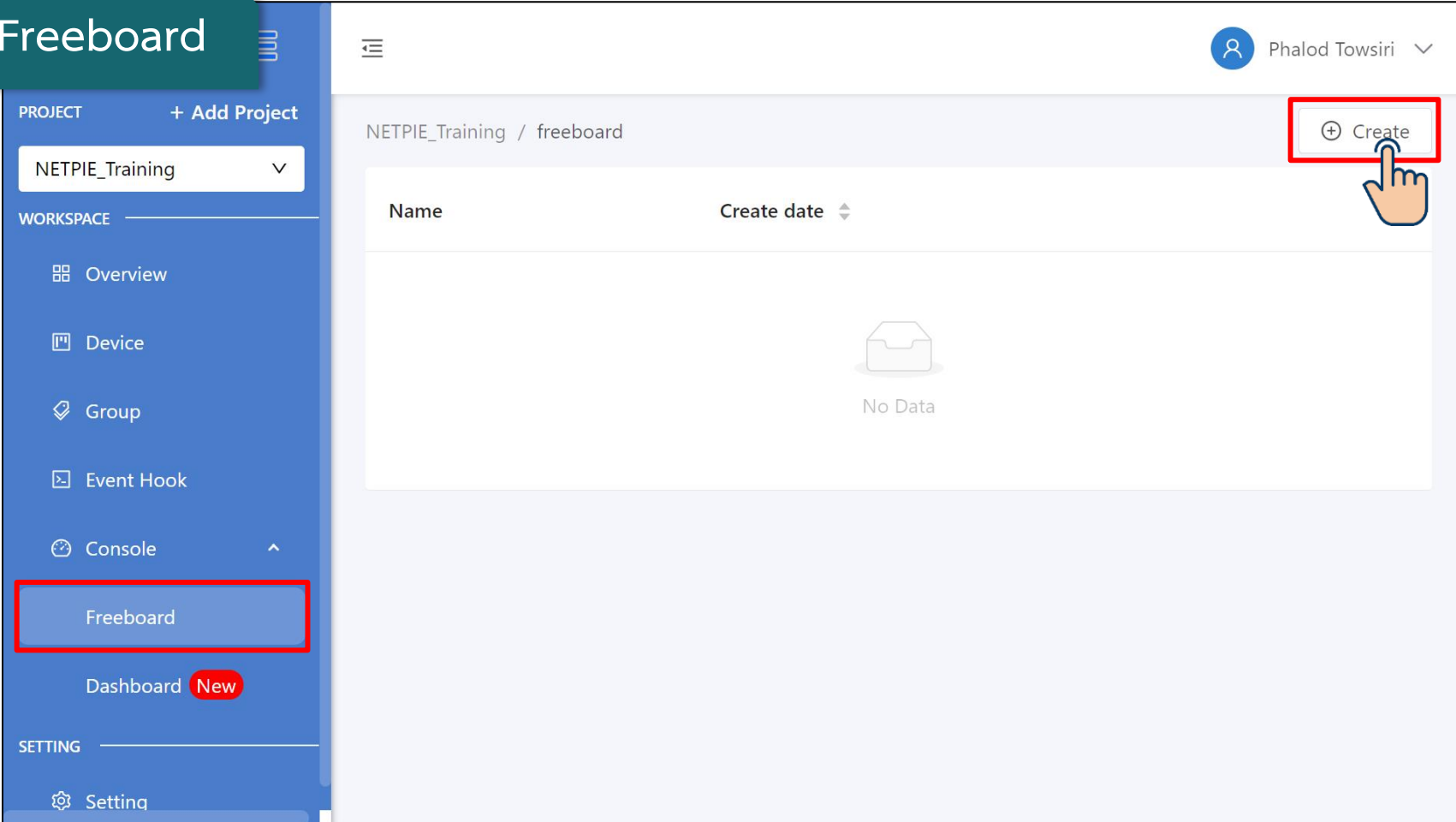
Datasource บน Freeboard คือ ฐานข้อมูลที่ตั้งมาจาก Device บน Project เดียวกันกับที่ Freeboard อาศัยอยู่ โดย Datasource นำข้อมูลต่างๆของ Device มารวมอยู่ด้วยกัน เช่น Device Shadow, Device Feed เป็นต้น อีกทั้งบน Freeboard สามารถสร้างจำนวน Datasource ได้มากกว่า 1 Datasource ดังรูปข้างล่าง



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource

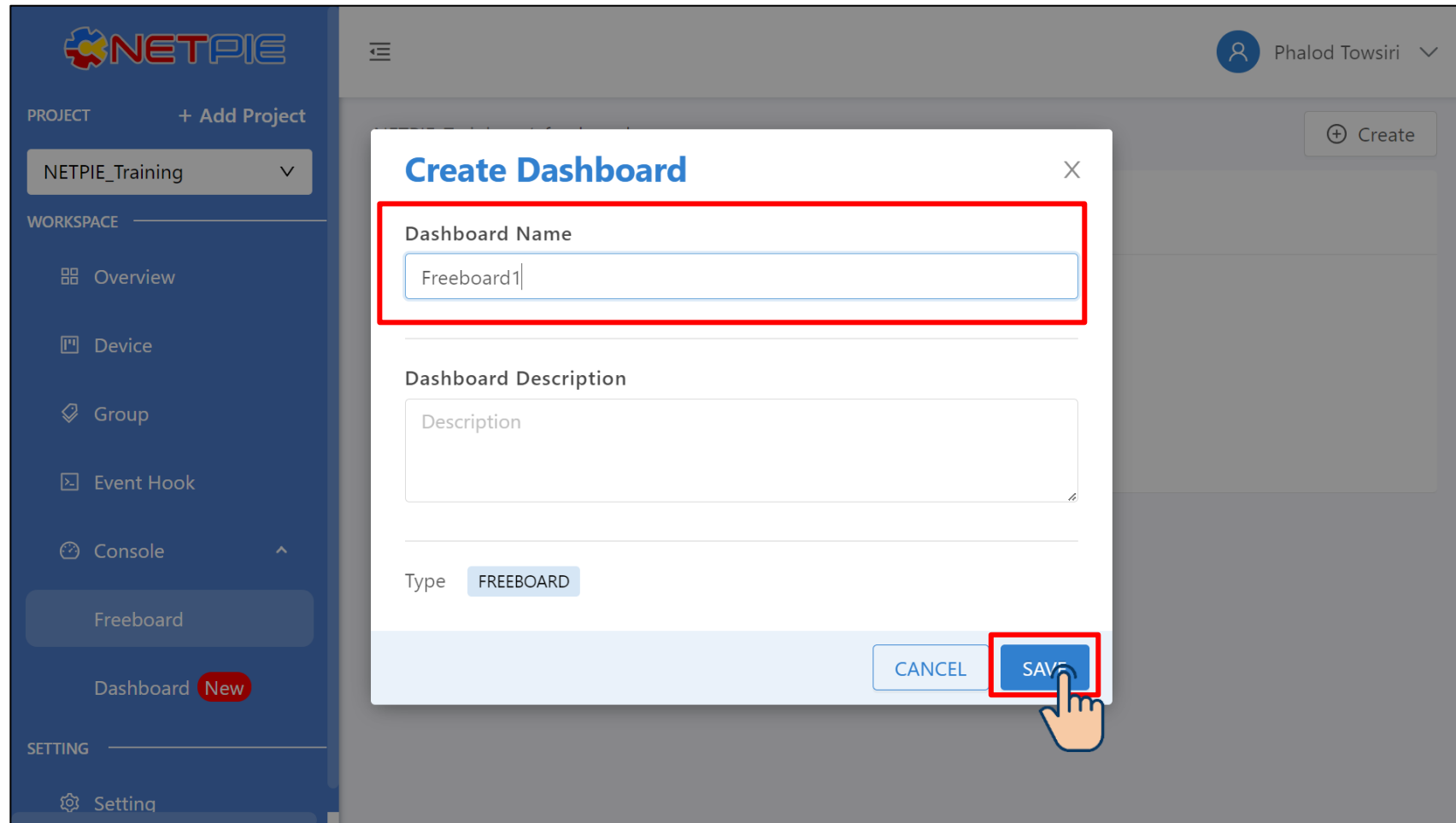
การสร้าง Freeboard



The screenshot displays the Freeboard application interface. On the left, a blue sidebar contains a navigation menu with the following items: PROJECT (+ Add Project), NETPIE_Training (selected), WORKSPACE (Overview, Device, Group, Event Hook, Console), Freeboard (highlighted with a red box), Dashboard (New), and SETTING (Setting). The main content area shows the path NETPIE_Training / freeboard and a '+ Create' button (highlighted with a red box and a hand cursor). Below the button is a table with columns 'Name' and 'Create date', and a 'No Data' message with a folder icon.

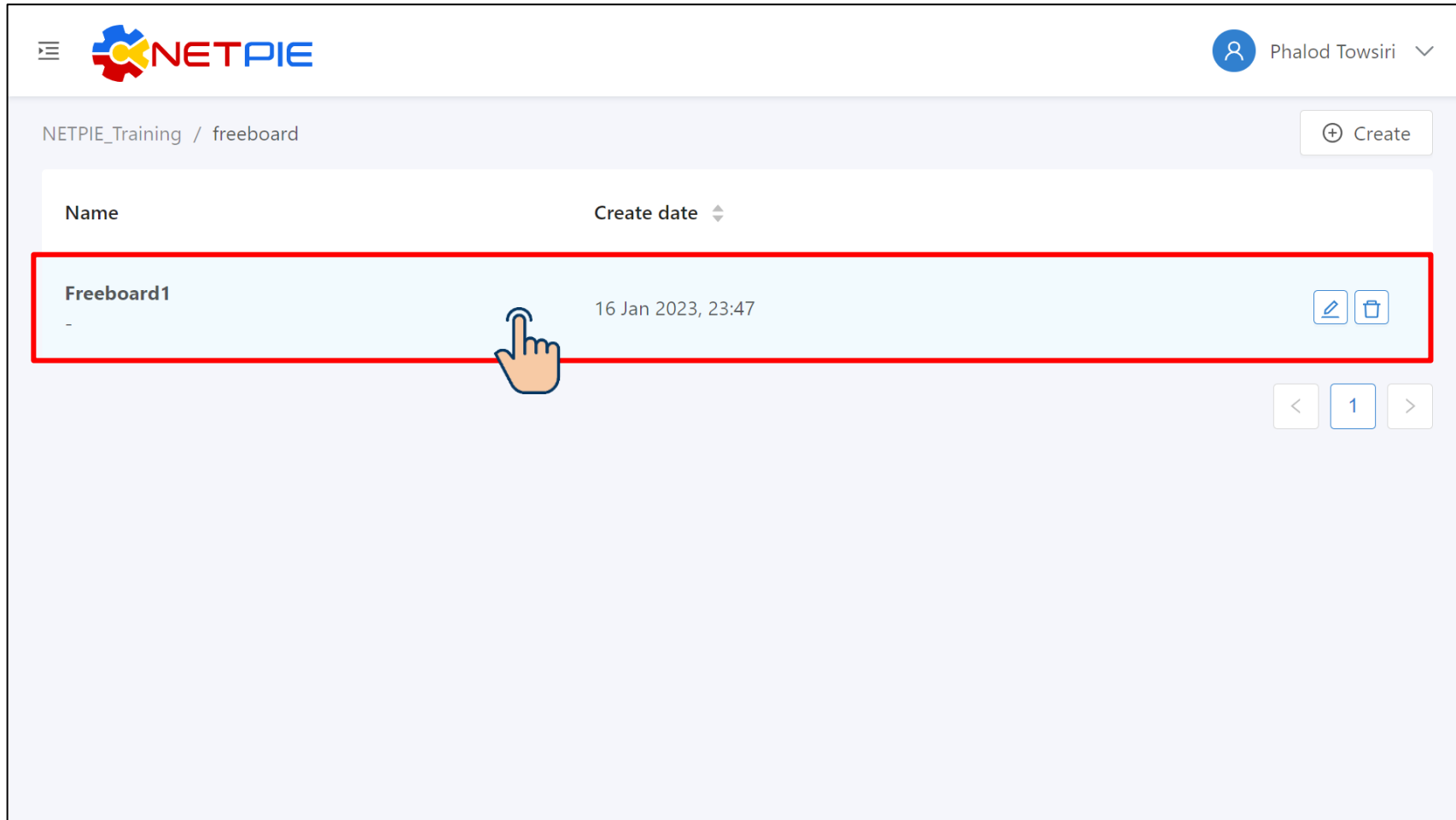
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource

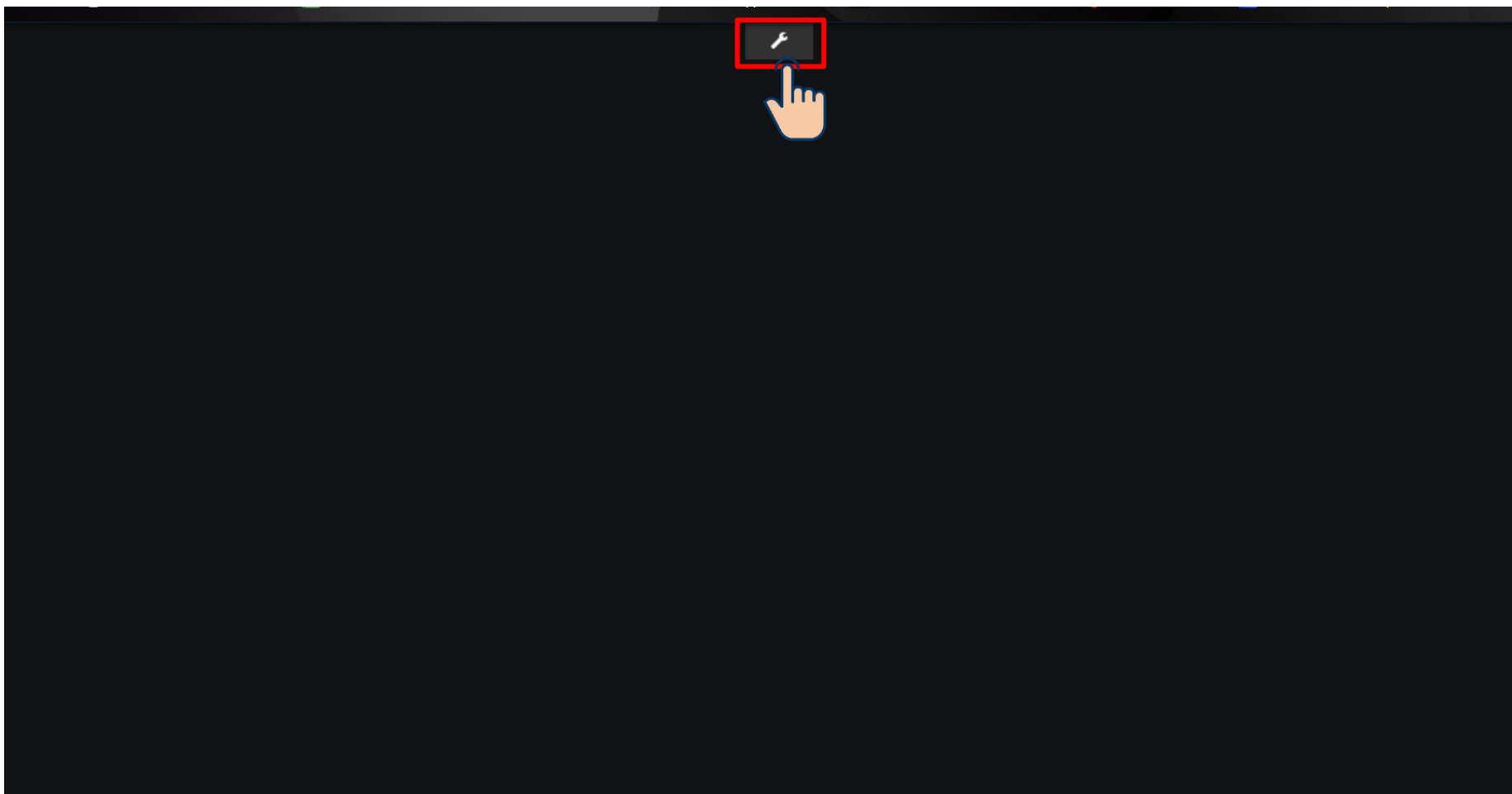


The screenshot displays the NETPIE web application interface. At the top left is the NETPIE logo, and at the top right is the user profile 'Phalod Towsiri'. The breadcrumb path is 'NETPIE_Training / freeboard'. A '+ Create' button is visible in the top right corner. Below the breadcrumb is a table with two columns: 'Name' and 'Create date'. The table contains one entry: 'Freeboard1' with a create date of '16 Jan 2023, 23:47'. This entry is highlighted with a red rectangular box, and a hand cursor is pointing to it. To the right of the entry are edit and delete icons. At the bottom right of the table area are navigation controls: '<', '1', and '>'.

Name	Create date
Freeboard1	16 Jan 2023, 23:47

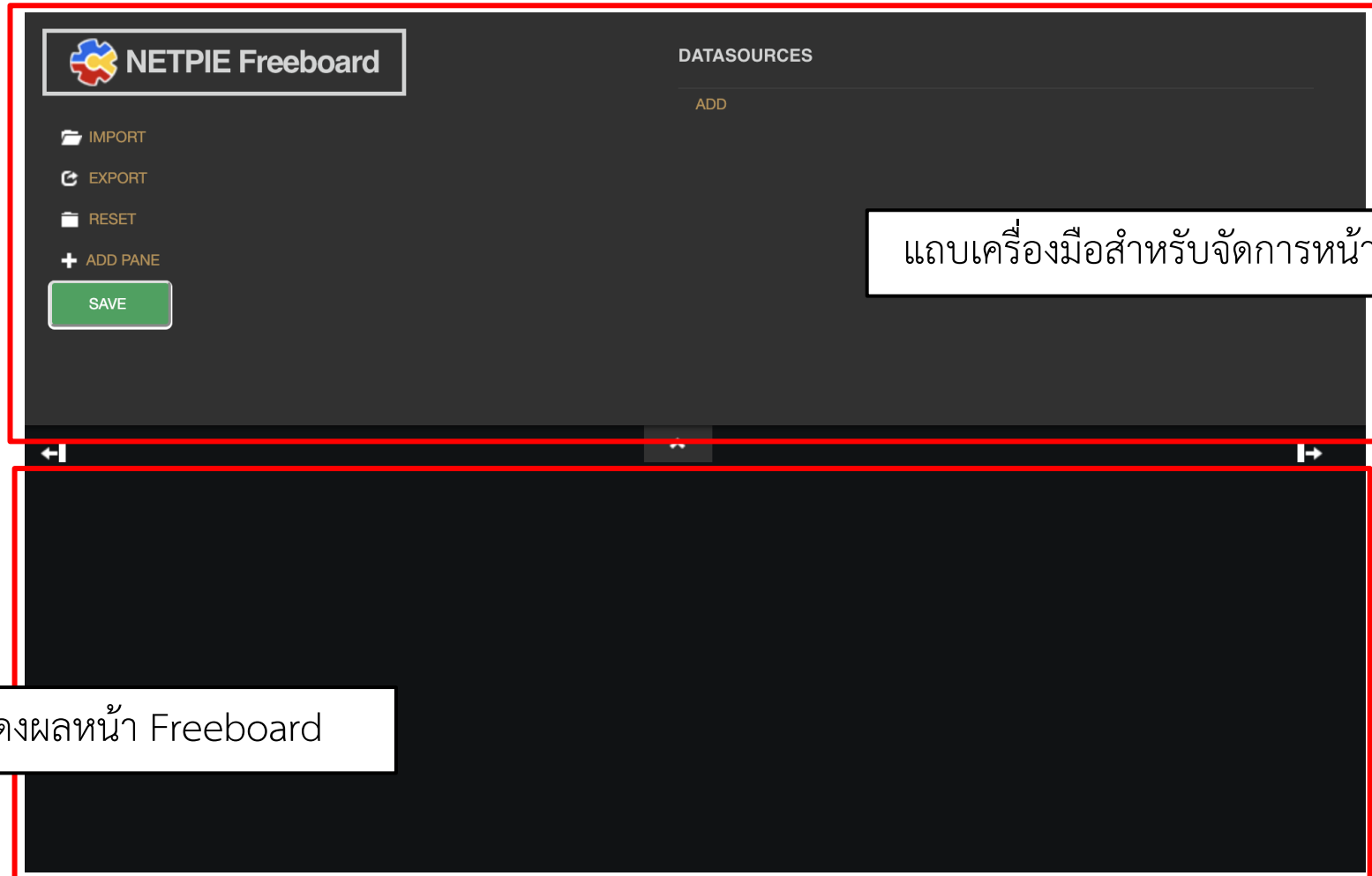
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource

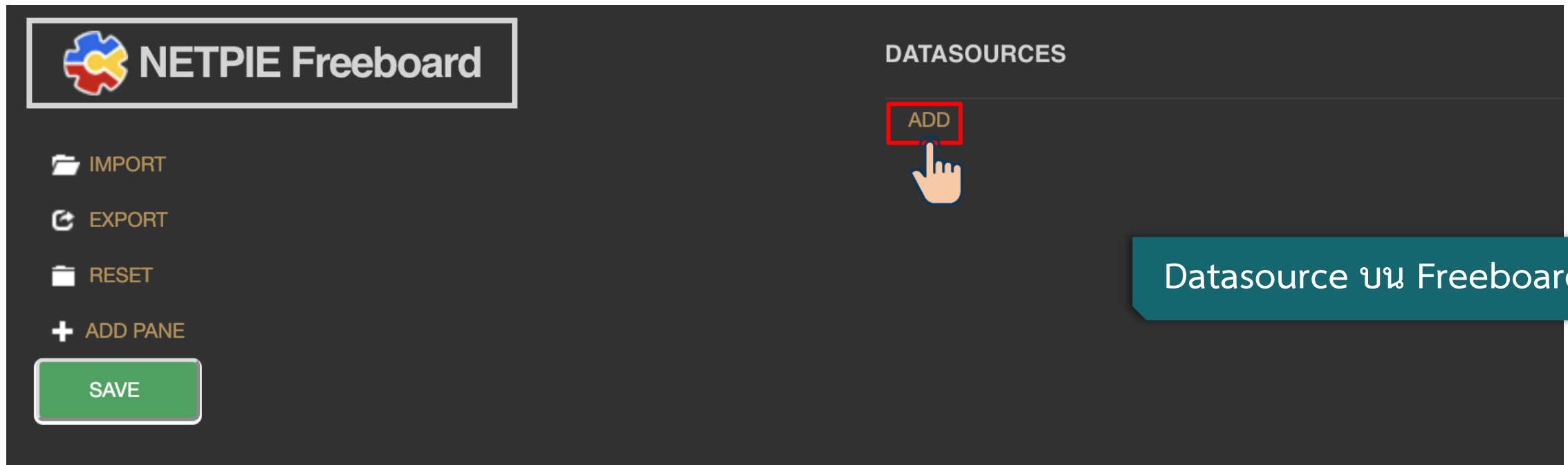


แถบเครื่องมือสำหรับจัดการหน้า Freeboard

แถบหน้าจอแสดงผลหน้า Freeboard

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource

ทดสอบการเพิ่ม Datasource ด้วยการดึงข้อมูลจาก Device2

ชื่อ Datasource
(เป็นชื่ออะไรก็ได้ แต่ไม่ควรเว้นวรรค ถ้าจำเป็นควรใช้ _ หรือ - แทน)

Client ID ของ Device

Token ของ Device

เปิดการใช้งาน FEED

การตั้งค่าส่วนของการแสดงผล Feed

NAME

DEVICE ID
Client ID ของ Device ที่ต้องการอ่านข้อมูล

DEVICE TOKEN
Token ของ Device ที่ต้องการอ่านข้อมูล

SUBSCRIBED TOPICS
Topic ที่ต้องการ Subscribe

FEED
NO

SINCE
6
Hour
Display data points since ... ago.

DOWN SAMPLING
1
Minute
Resolution of the data points.

SAVE CANCEL

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource

DATASOURCE

ทดสอบการเพิ่ม Datasource ด้วยการดึงข้อมูลจาก Device2

ตัวอย่างการตั้งค่าต่างๆบน Datasource

NAME: Device2

DEVICE ID: 4afbad5c-201c-4be5-9771-c7d499748624
Client ID ของ Device ที่ต้องการอ่านข้อมูล

DEVICE TOKEN: pu5ehH6TIF1Sqv7JZvnsKUUUKL53TTy7x
Token ของ Device ที่ต้องการอ่านข้อมูล

SUBSCRIBED TOPICS:
Topic ที่ต้องการ Subscribe

FEED: YES

SINCE: 6
Hour
Display data points since ... ago.

DOWN SAMPLING: 1
Minute
Resolution of the data points.

SAVE CANCEL

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Freeboard และทดสอบเชื่อม Datasource

The screenshot shows the NETPIE Freeboard interface. On the left, there is a sidebar with the following options: IMPORT, EXPORT, RESET, ADD PANE, and a green SAVE button. The main area is titled 'DATASOURCES' and contains a table with two columns: 'Name' and 'Last Updated'. The table has one entry: 'Device2' with a last updated time of '1:18:46 AM'. To the right of the table are refresh and delete icons. Below the table is an 'ADD' button. A red box highlights the 'Device2' row, and a red arrow points from this box to a text box containing the Thai text: 'เมื่อเชื่อมต่อสำเร็จจะขึ้นสถานะดังรูป' (When connected successfully, it will show the status like this).

Name	Last Updated
Device2	1:18:46 AM

ใบงานที่ 5.1 ทฤษฎีเบื้องต้น

การนำเสนอข้อมูล

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

Shadow Schema Trigger Feed

Tree Last Update : 11-01-23 16:31

Select a node...

- object {3}
- humidity : 54
- light : 15
- temperature : 77

ข้อมูลที่อยู่บน Device2



ใบงานที่ 5.1 ทฤษฎีเบื้องต้น

การใช้งาน Widget

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

WIDGET

TYPE

TITLE

VALUE + DATASOURCE .JS EDITOR

UNITS

SAVE CANCEL

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

ใบงานที่ 5.1 ทฤษฎีเบื้องต้น

การใช้งาน Widget

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

WIDGET

A simple button widget that can perform Javascript action.

TYPE:

BUTTON CAPTION:

LABEL TEXT:

BUTTON COLOR:

ONCLICK ACTION: + DATASOURCE .JS EDITOR

Add some Javascript here.

ONCREATED ACTION:

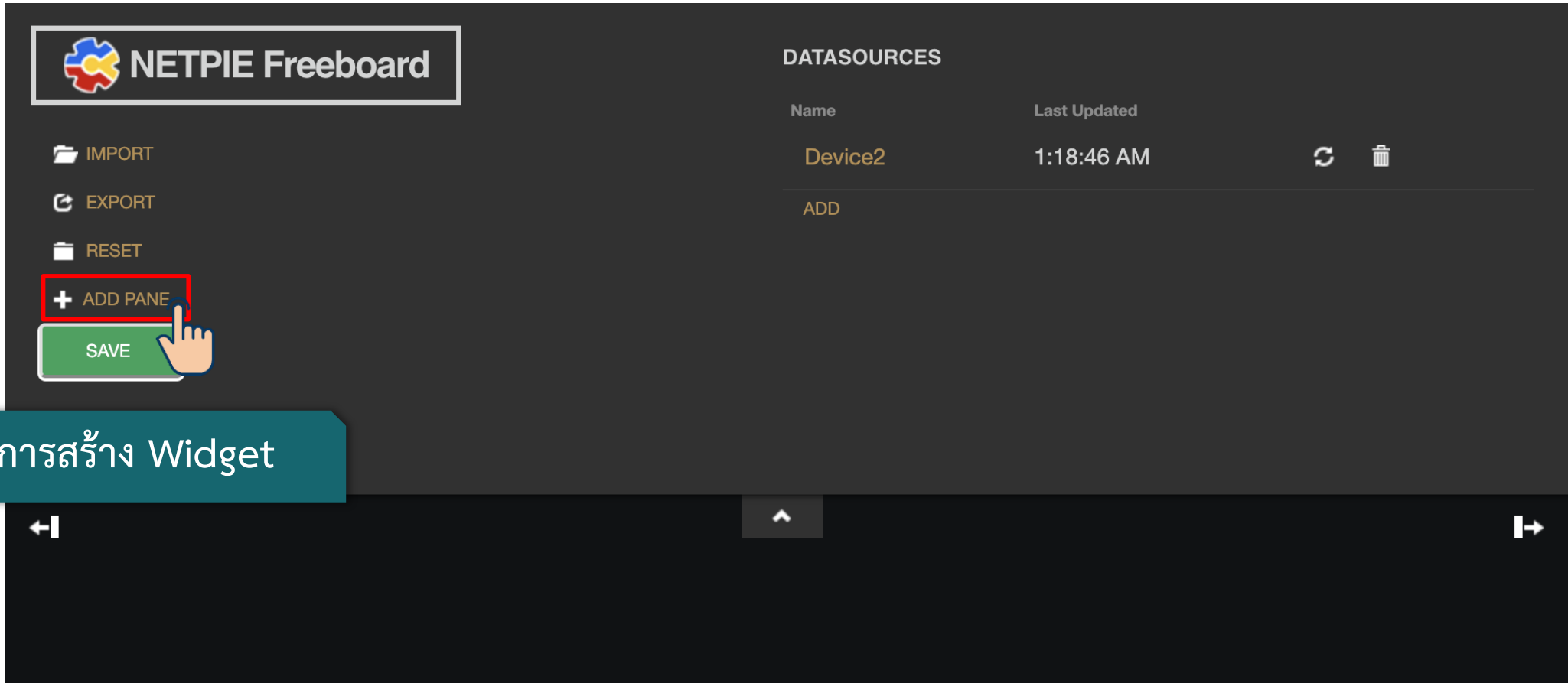
JS code to run after a button is created

SAVE CANCEL

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

ใบงานที่ 5.1 ขั้นตอนการทดลอง

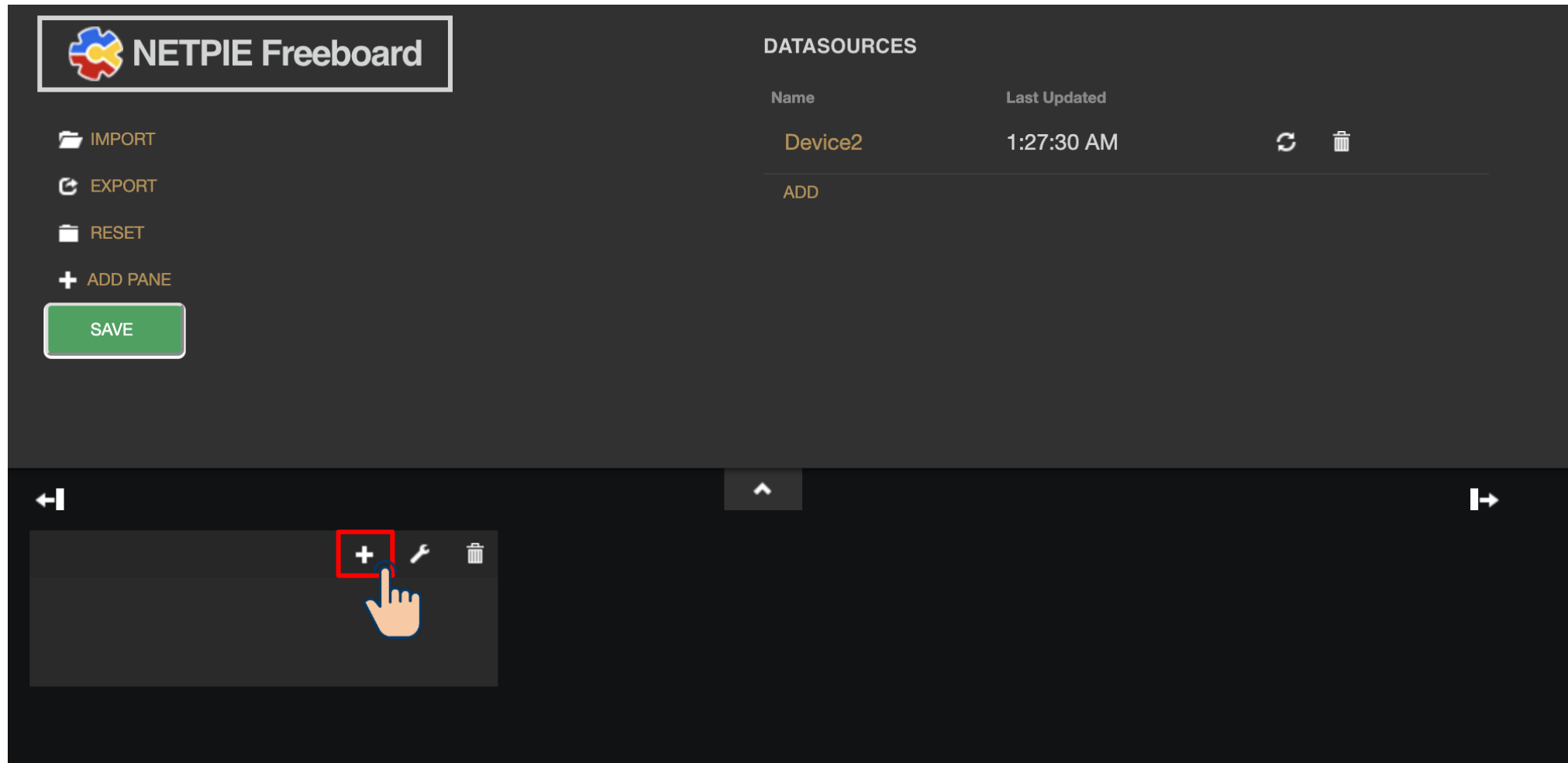
การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



การสร้าง Widget

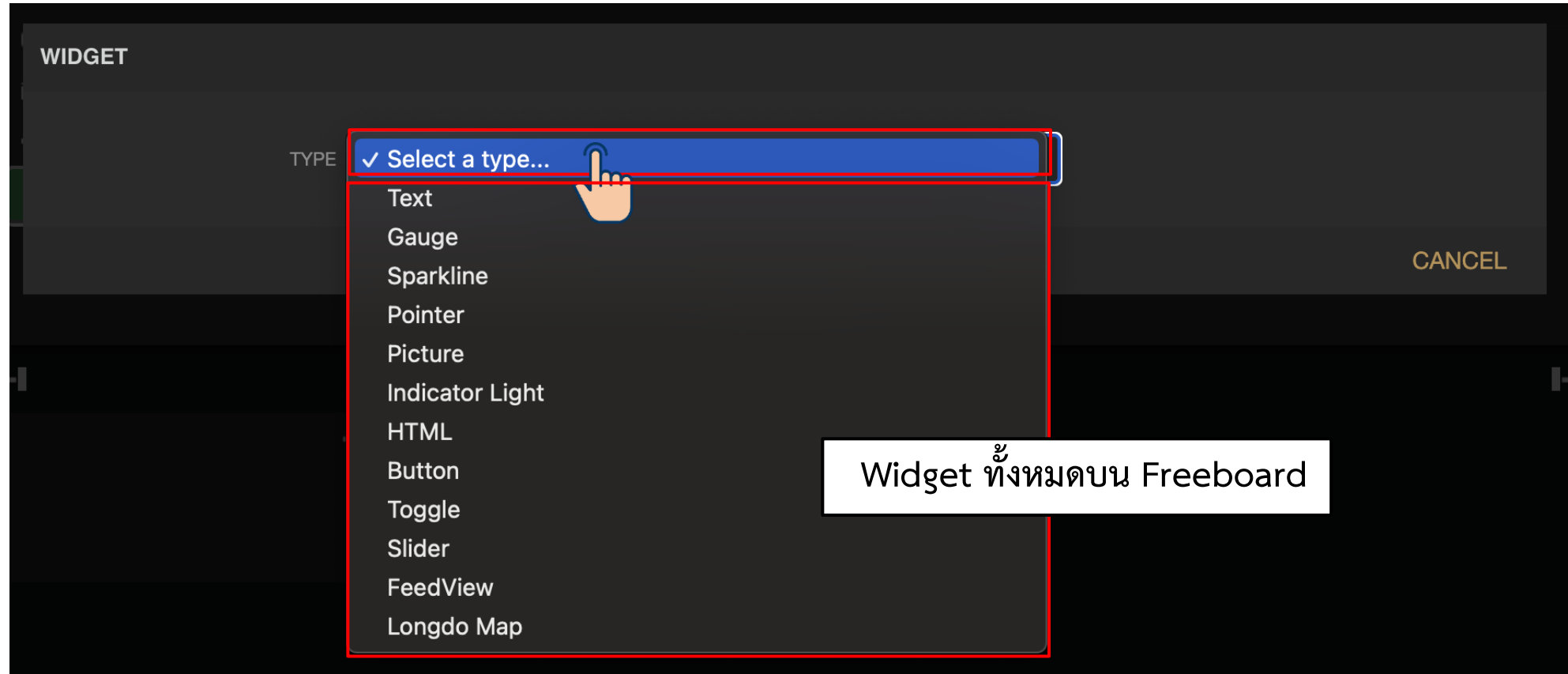
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



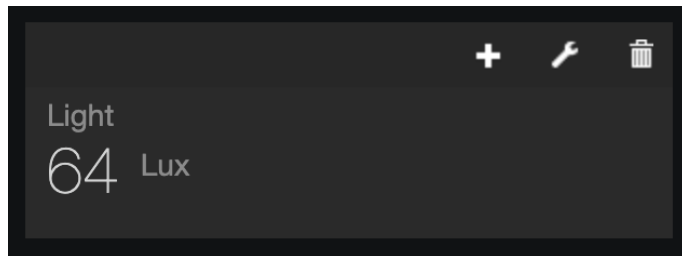
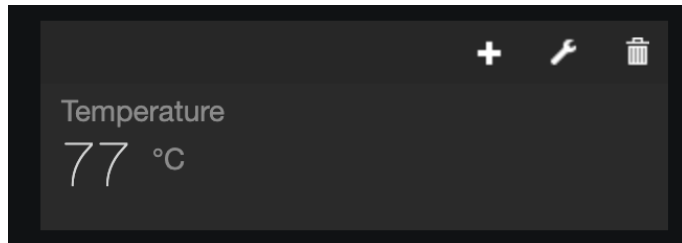
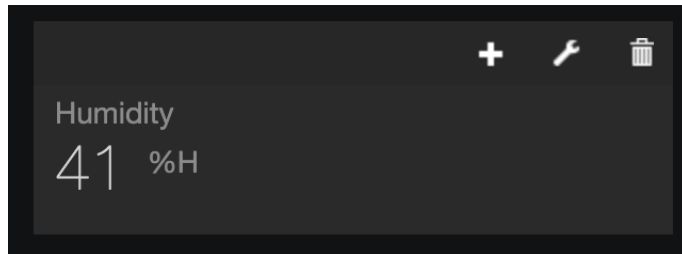
ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

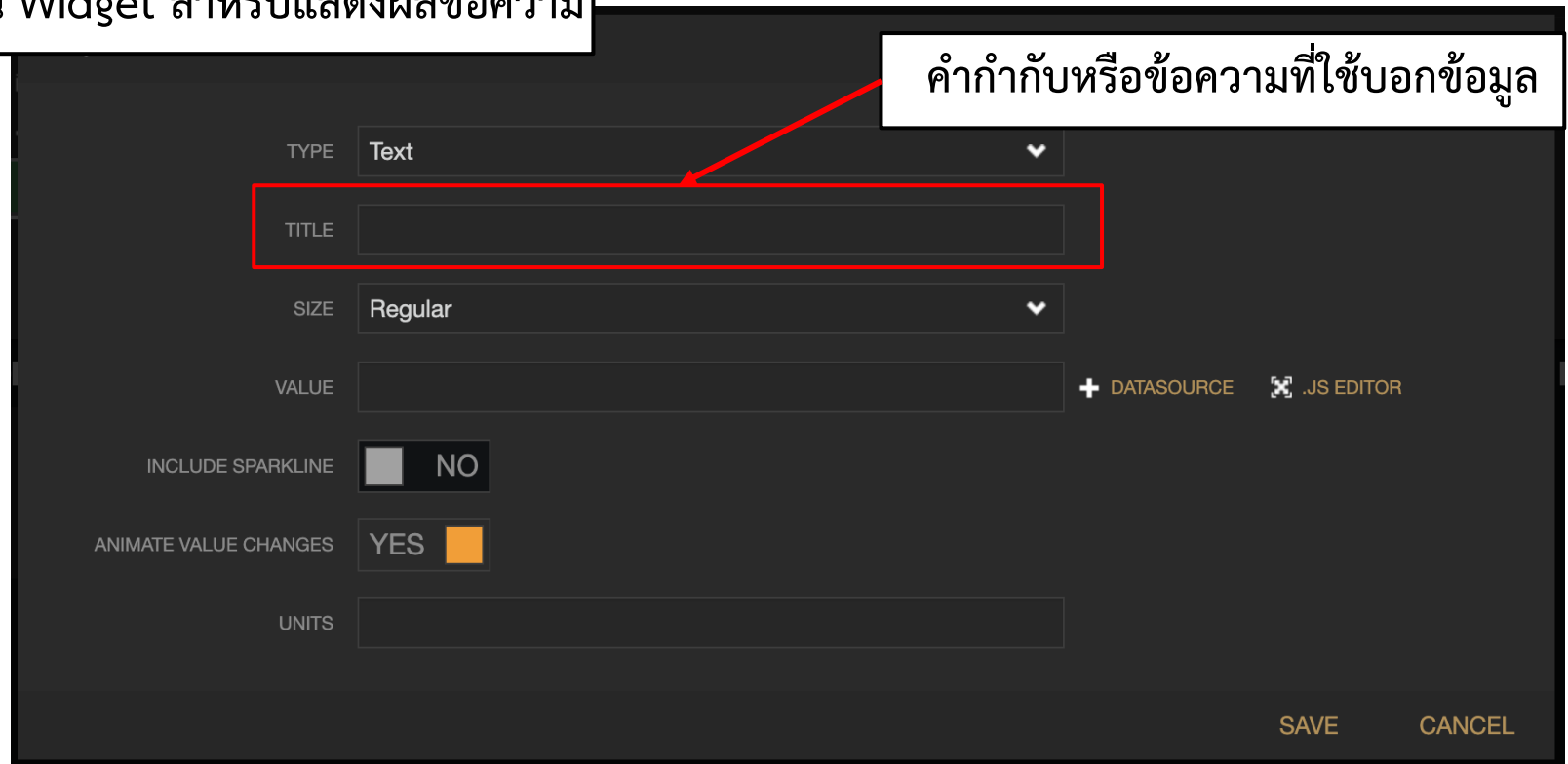
การสร้าง Widget : Text

ตัวอย่าง Widget : Text

เป็น Widget สำหรับแสดงผลข้อความ



คำกำกับหรือข้อความที่ใช้บอกข้อมูล



The configuration interface for a Text widget. It features a dark background with the following fields and options:

- TYPE:** Text (dropdown menu)
- TITLE:** An empty text input field, highlighted with a red box and pointed to by a red arrow from the label "คำกำกับหรือข้อความที่ใช้บอกข้อมูล".
- SIZE:** Regular (dropdown menu)
- VALUE:** An empty text input field, with a "+ DATASOURCE" button and a ".JS EDITOR" icon to its right.
- INCLUDE SPARKLINE:** A toggle switch set to "NO".
- ANIMATE VALUE CHANGES:** A toggle switch set to "YES".
- UNITS:** An empty text input field.
- SAVE** and **CANCEL** buttons are located at the bottom right.

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text

WIDGET

TYPE

TITLE

SIZE

VALUE + DATASOURCE .JS EDITOR

INCLUDE SPARKLINE NO

ANIMATE VALUE CHANGES YES

UNITS

SAVE CANCEL

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text

WIDGET

TYPE

TITLE

SIZE

VALUE + DATASOURCE .JS EDITOR

INCLUDE SPARKLINE NO

ANIMATE VALUE CHANGES YES

UNITS

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text

The screenshot displays the NETPIE Freeboard interface. On the left, there is a sidebar with navigation options: IMPORT, EXPORT, RESET, and ADD PANE, along with a green SAVE button. The main area is titled 'DATASOURCES' and contains a table with one entry: 'Device2' with a 'Last Updated' time of '1:41:17 AM'. Below the table is an 'ADD' button. A white text box with the Thai text 'ค่าของความชื้น' (Humidity value) is positioned above a widget. The widget itself is a dark grey rectangle with a red border, containing the text 'Humidity 42 %H'. A red arrow points from the text box to the widget. The bottom of the interface features navigation icons: a left arrow, an up arrow, and a right arrow.

Name	Last Updated		
Device2	1:41:17 AM	↻	🗑️
ADD			

ค่าของความชื้น

Humidity
42 %H

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text

The screenshot displays the NETPIE Freeboard interface. On the left, there is a sidebar with options: IMPORT, EXPORT, RESET, ADD PANE, and a green SAVE button. The main area is titled 'DATASOURCES' and contains a table with one entry: 'Device2' with a 'Last Updated' time of '1:41:17 AM'. Below the table is an 'ADD' button. At the bottom, a widget for 'Humidity' is shown with a value of '42 %H'. A red box highlights the edit icon (a wrench) next to the widget, with a red arrow pointing to it from a text box that says 'หากต้องการแก้ไขข้อมูล' (If you need to edit the information).

Name	Last Updated	
Device2	1:41:17 AM	🔄 🗑️
ADD		

หากต้องการแก้ไขข้อมูล

Humidity
42 %H

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text

The screenshot displays the NETPIE Freeboard interface. On the left, there is a sidebar with navigation options: IMPORT, EXPORT, RESET, ADD PANE, and a prominent green SAVE button. The top right section, titled 'DATASOURCES', contains a table with the following data:

Name	Last Updated		
Device2	1:43:32 AM	↻	🗑️
ADD			

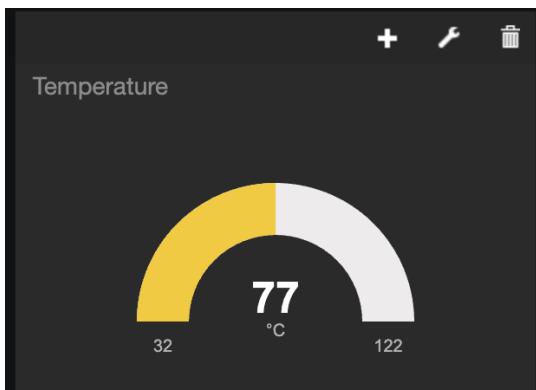
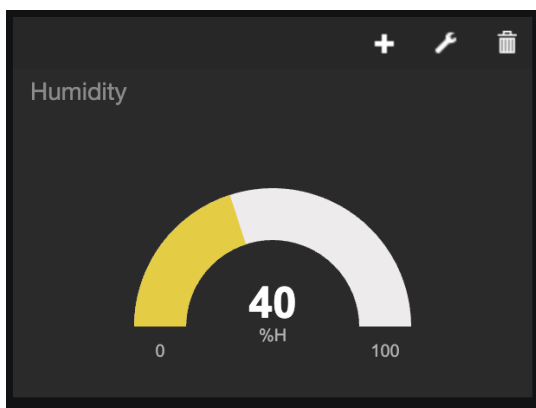
Below the table, a white callout box with a red arrow pointing to the Temperature widget contains the text: "ทดสอบสร้าง Text สำหรับแสดงผล Temperature และ Light ด้วยตัวเอง". At the bottom of the dashboard, three widget cards are visible: Humidity (41 %H), Temperature (77 °C), and Light (64 Lux). The Temperature and Light widgets are enclosed in a red rectangular box.

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge

ตัวอย่าง Widget : Gauge



เป็น Widget สำหรับแสดงผลข้อมูลแบบ Gauge

WIDGET

TYPE Gauge

TITLE

VALUE + DATA

UNITS

MINIMUM 0

MAXIMUM 100

SAVE CANCEL

คำกำกับหรือข้อความที่ใช้บอกข้อมูล

หน่วยของข้อมูลที่แสดงผล

กำหนดช่วงข้อมูลของ Gauge

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge

The screenshot displays the NETPIE Freeboard interface. On the left sidebar, the 'ADD PANE' button is highlighted with a red box and a hand icon. The main area shows a 'DATASOURCES' table with one entry 'Device2' and an 'ADD' button. Below the table, there are three widget panes: a blank pane with a red box around its '+' icon, a 'Temperature' gauge showing 77 °C, and a 'Light' gauge showing 62 Lux. A 'Humidity' gauge showing 41 %H is partially visible at the bottom left.

Name	Last Updated	
Device2	1:47:28 AM	🔄 🗑️
ADD		

Temperature
77 °C

Light
62 Lux

Humidity
41 %H

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge

WIDGET

TYPE	Gauge	
TITLE		
VALUE		+ DATASOURCE
UNITS		.JS EDITOR
MINIMUM	0	
MAXIMUM	100	

SAVE CANCEL

ใบงานที่ 5.1 ขั้นตอนการทดลอง


การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge

WIDGET

TYPE	Gauge	
TITLE	Humidity	
VALUE	<code>datasources["Device2"]["shadow"]["humidity"]</code>	+ DATASOURCE .JS EDITOR
UNITS	%H	
MINIMUM	0	
MAXIMUM	100	

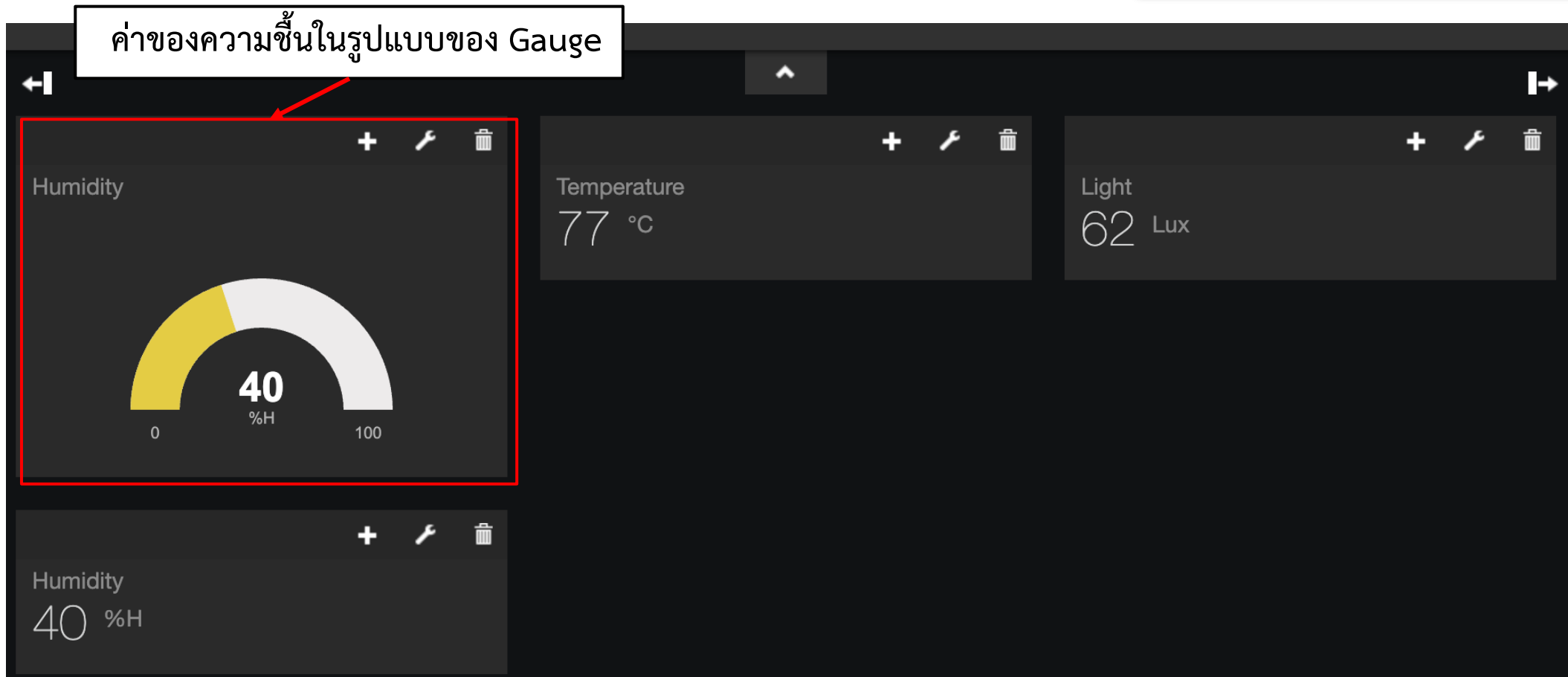
SAVE CANCEL



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge

The image displays a grid of six widget editors. The top row contains three Gauge widgets: Humidity (40 %H), Temperature (77 °C), and Light (62 Lux). The bottom row contains three corresponding Text widgets: Humidity (40 %H), Temperature (77 °C), and Light. A red box highlights the Temperature and Light Gauge widgets in the top row. A red arrow points from a callout box to the Temperature and Light Text widgets in the bottom row. The callout box contains the text: 'ทดสอบสร้าง Text สำหรับแสดงผล Temperature และ Light ด้วยตัวเอง'.

ใบงานที่ 5.1 ขั้นตอนการทดลอง

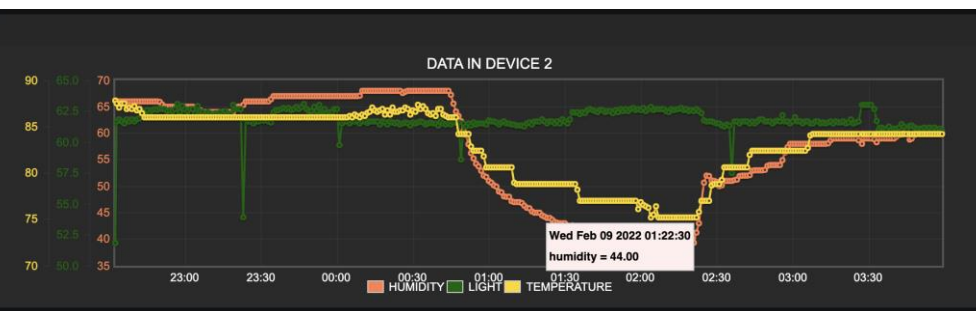
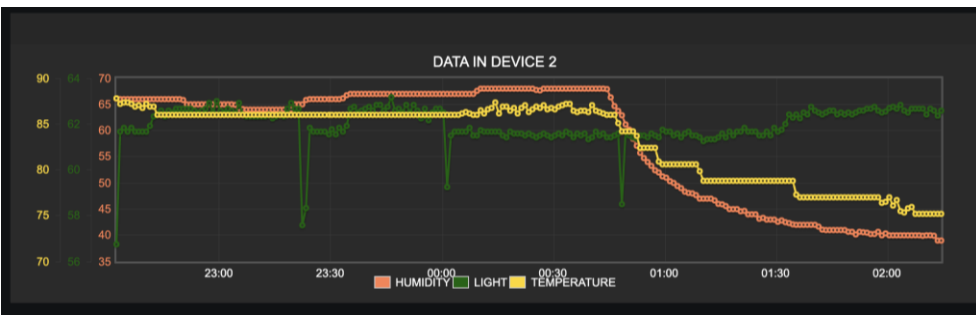
การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

ตัวอย่าง Widget : FeedView

เป็น Widget สำหรับแสดงผลข้อมูลแบบกราฟ

ข้อมูลที่ต้องการนำเสนอบนกราฟ
โดยใส่ชื่อข้อมูลที่ต้องการนำเสนอลงไป
หากเว้นว่างไว้จะนำเสนอทุกข้อมูล



TYPE: FeedView

TITLE: []

DATA SOURCE: [] + DATASOURCE [] JS EDITOR []

FILTER: []
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

การตกแต่งส่วนต่างๆของกราฟ

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

WIDGET

TYPE

TITLE

DATA SOURCE + DATASOURCE JS EDITOR

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

TYPE OF CHART

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

SAVE CANCEL

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

WIDGET

TYPE: FeedView

TITLE:

DATA SOURCE: + DATASOURCE .JS EDITOR

FILTER:

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:

กำหนด Datasource ให้ถูกต้อง

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

WIDGET

TYPE: FeedView

TITLE: Data in Device 2

DATA SOURCE: datasources["Device2"]["feed"] + DATASOURCE JS EDITOR

FILTER:
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: #FF7F50,#006400,#FFD700
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

กำหนดหัวข้อของกราฟ

สามารถกำหนดค่าต่างๆได้เพื่อเป็นการตกแต่งกราฟ

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

หลังจาก Save แล้วจะได้กราฟมา ซึ่งขนาดของกราฟอาจจะไม่เหมาะสม
ทำการปรับขนาดของกราฟด้วยการเลือกตั้งรูป

The screenshot displays a dashboard with the following components:

- Line Graph (DATA IN DEVICE 2):** Shows three data series: Humidity (orange), Light (green), and Temperature (yellow). The Y-axis ranges from 35 to 90. A red box highlights a wrench icon in the top right corner of the graph, indicating a resize handle.
- Temperature Gauge (Top Right):** Shows a value of 75.2 °C. The scale ranges from 32 to 122.
- Light Gauge (Top Right):** Shows a value of 64 Lux. The scale ranges from 0 to 100.
- Temperature Gauge (Bottom Middle):** Shows a value of 75.2 °C.
- Light Gauge (Bottom Right):** Shows a value of 64 Lux.
- Humidity Gauge (Bottom Left):** Shows a value of 30.

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

PANE

เปลี่ยนขนาดของ Columns จาก 1 ให้เป็น 3

TITLE

COLUMNS

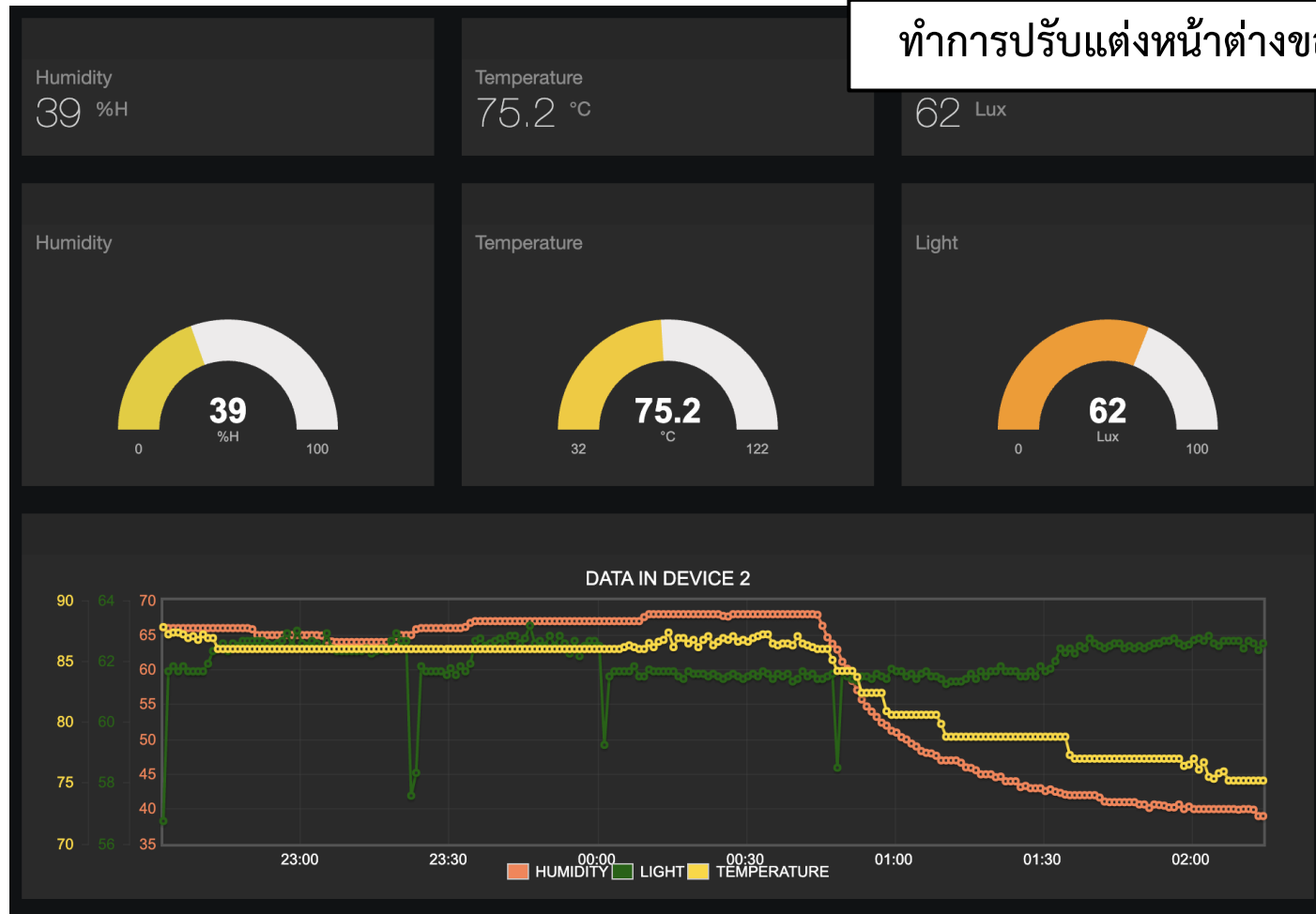
SAVE CANCEL

ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : FeedView

ทำการปรับแต่งหน้าตาต่างของ Freeboard



ใบงานที่ 5.1 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การ Save Freeboard

The screenshot shows the NETPIE Freeboard interface. At the top left, there is a logo and the text 'NETPIE Freeboard'. Below it, there are several menu items: 'IMPORT', 'EXPORT', 'RESET', and '+ ADD PANE'. A green 'SAVE' button is highlighted with a red box, and a hand cursor is pointing at it. To the right of the menu items, there is a 'DATASOURCES' section with a table containing one row: 'Device2' with 'Last Updated' as '2:17:46 AM'. Below the menu items, there are six data widgets arranged in a 2x3 grid. The top row shows 'Humidity 39 %H', 'Temperature 75.2 °C', and 'Light 63 Lux'. The bottom row shows 'Humidity' with a gauge, 'Temperature' with a gauge, and 'Light' with a gauge. The gauges show the current values: 39 for Humidity, 75.2 for Temperature, and 63 for Light.

ควรทำการ Save สำรองเสมอ เนื่องจาก Freeboard ไม่ได้ทำการ Save ให้อัตโนมัติเมื่อออกจากหน้าต่าง Freeboard โดยไม่ได้ Save จะทำให้ข้อมูลทั้งหมดหายไป

ใบงานที่ 5.1 คำถามท้ายหน่วยการเรียนรู้

คำถามท้ายหน่วยการเรียนรู้ที่ 1

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

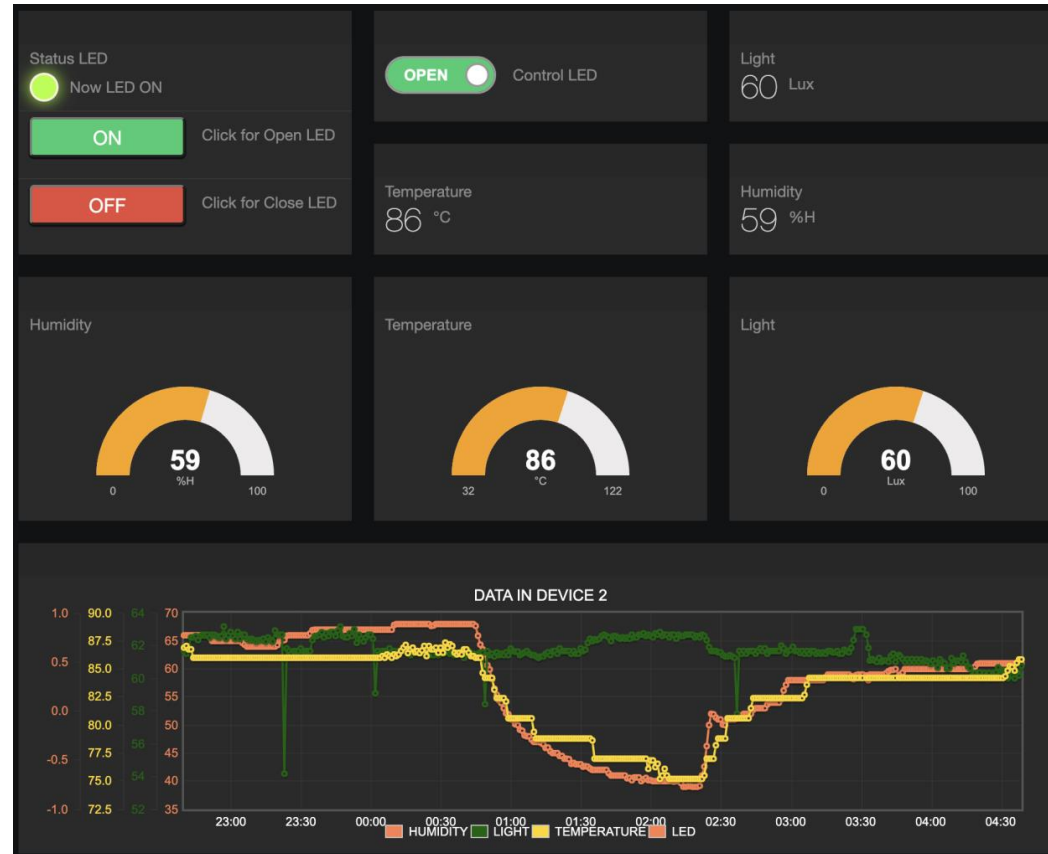
1. Widget : Text แสดงผลข้อมูล voltage, gas, led1, led2 และ led3
2. Widget : Gauge แสดงผลข้อมูล voltage และ gas
3. Widget : FeedView แสดงผลข้อมูล Voltage และ Gas

โดยทุก Widget ของแต่ละข้อมูลกำหนดให้เป็นอิสระจากกัน (นั่นคือ ต้องมี FeedView 2 อัน)

*** เสริมโจทย์ การแปลงข้อมูลบน Widget : Text ตามเงื่อนไขดังต่อไปนี้ได้

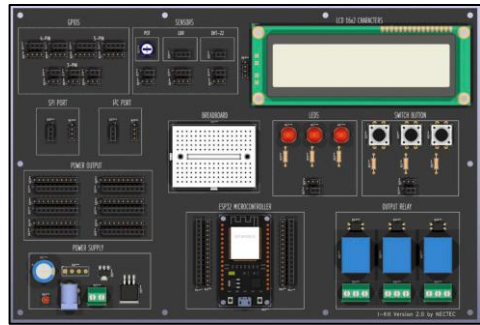
- led1, led2 และ led3 ปกติแล้วข้อมูลจะนำเสนอจาก 0 หรือ false ให้แสดงผลข้อมูลเป็น LED OFF
- led1, led2 และ led3 ปกติแล้วข้อมูลจะนำเสนอจาก 1 หรือ true ให้แสดงผลข้อมูลเป็น LED ON

ใบงานที่ 5.2 การควบคุมอุปกรณ์ด้วย Freeboard



ใบงานที่ 5.2 ทฤษฎีเบื้องต้น

หลักการสื่อสารระหว่างอุปกรณ์และ Freeboard



ESP32

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

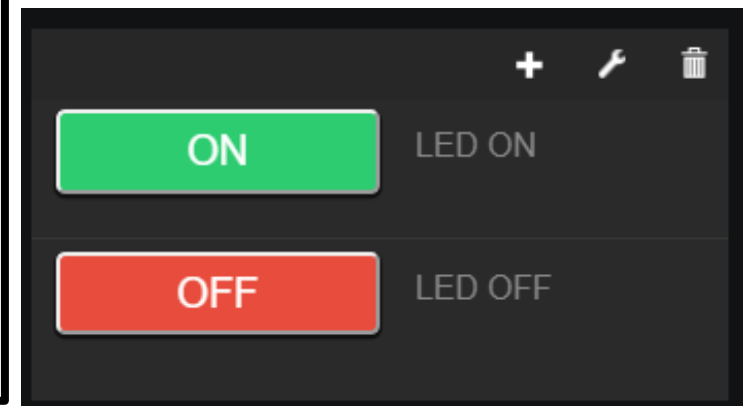


@msg/led : "ON/OFF"



NETPIE

Freeboard

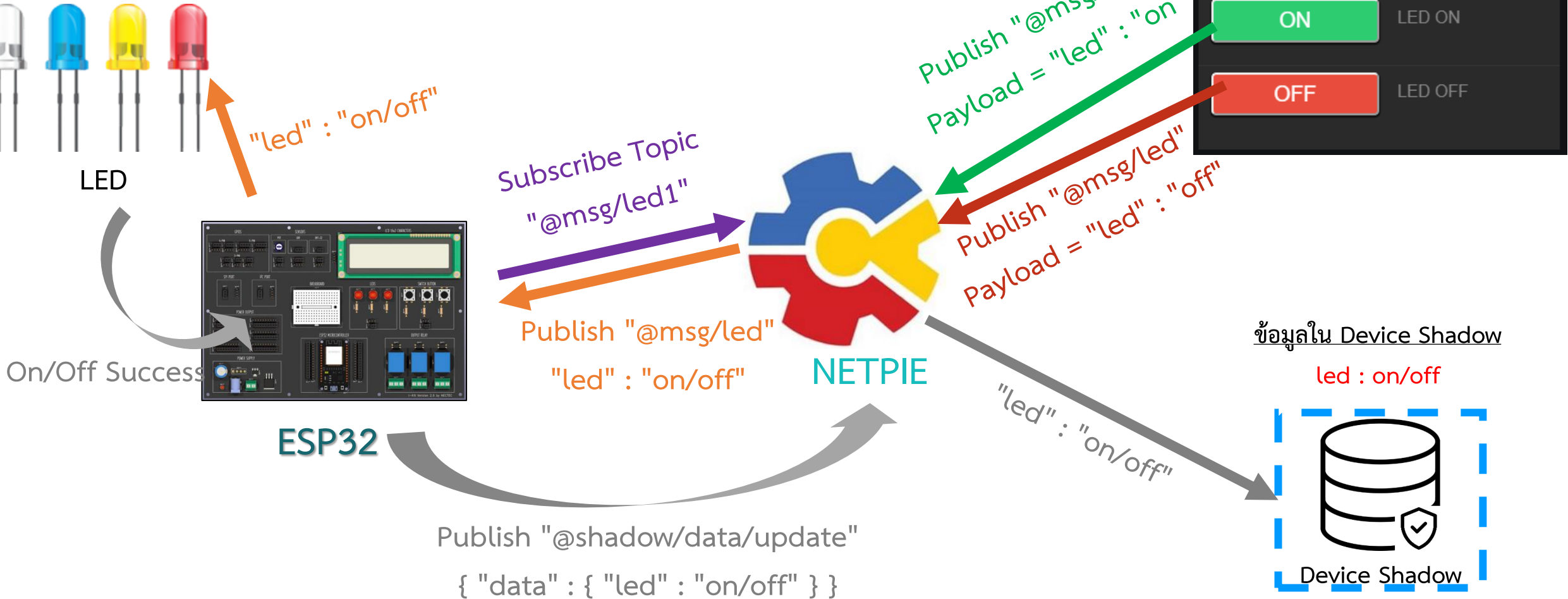


โดยปกติแล้วการส่งข้อมูลจากอุปกรณ์ไปยัง Device บน NETPIE เราต้องใช้ Shadow Topic เพื่อให้ข้อมูลถูกจัดเก็บลง Device Shadow แต่ในกรณี ของ Freeboard ไปยัง Device เราจะใช้วิธีการดังนี้

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

ใบงานที่ 5.2 ทฤษฎีเบื้องต้น

หลักการสื่อสารระหว่างอุปกรณ์และ Freeboard



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

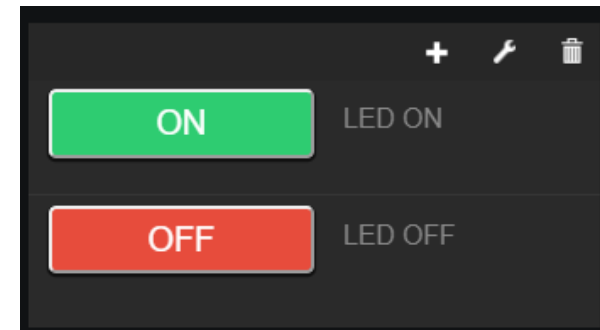
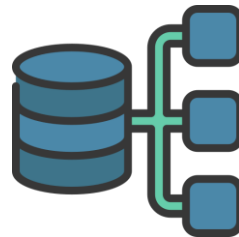
เช็คความพร้อมเพื่อทดสอบทั้งหมด 3 ส่วน

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



ESP32

Device Schema



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

1

ส่วนที่ 1 การเรียกใช้ Library และ ประกาศตัวแปร

ประกาศตัวแปรและขาใช้สำหรับ LED

ประกาศตัวแปรและขาใช้สำหรับการทำ millis เนื่องจากมีการรับส่งข้อมูลอยู่ตลอดการใช้ delay จะไม่เหมาะสม

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
const char* ssid = "Your_SSID";
const char* password = "Your_Password";
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "Client_ID";
const char* mqtt_username = "Token";
const char* mqtt_password = "Secret";
WiFiClient espClient;
PubSubClient client(espClient);
#define ldr 36
float ADC_value = 0.0048828125;
char msg[150];
#define DHTPIN 34
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

#define LED1 5

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

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

2 ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

```
void reconnect() {
```

ฟังก์ชันการเชื่อมต่อ MQTT Server

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

คำสั่ง Subscribe Topic
สำหรับรอรับคำสั่งจาก Freeboard

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

2

ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการรับข้อความ void callback

การเขียนเงื่อนไขเพื่อควบคุมการเปิด/ปิด LED

โดยเริ่มจากการเช็ค Topic ก่อนเสมอ

หลังจากนั้นทำการเช็คข้อความที่ได้รับ

เมื่อได้รับข้อความตามที่กำหนดจะทำการ Publish ข้อมูล

กลับไป NETPIE เพื่อทำการอัปเดตสถานะของ LED

```
void callback(char* topic, byte* payload, unsigned int length) {  
  Serial.print("Message arrived [");  
  Serial.print(topic);  
  Serial.print("] ");  
  String message;  
  for (int i = 0; i < length; i++) {  
    message = message + (char)payload[i];  
  }  
  Serial.println(message);  
  
  if (String(topic) == "@msg/led") {  
    if (message == "on") {  
      digitalWrite(LED1, 0);  
      client.publish("@shadow/data/update", "{\"data\" : {\"led\" : \"on\"}}");  
      Serial.println("LED ON");  
    }  
    else if (message == "off") {  
      digitalWrite(LED1, 1);  
      client.publish("@shadow/data/update", "{\"data\" : {\"led\" : \"off\"}}");  
      Serial.println("LED OFF");  
    }  
  }  
}
```

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

2

ส่วนที่ 2 ส่วนของฟังก์ชันต่างๆ

ฟังก์ชันการตั้งค่าต่างๆ

ประกาศการใช้งานฟังก์ชัน callback

กำหนดประเภทขาของ LED1
และสั่งให้ LED1 ดับ

```
void setup() {  
  Serial.begin(9600);  
  Serial.println();  
  Serial.print("Connecting to ");  
  Serial.println(ssid);  
  WiFi.begin(ssid, password);  
  while (WiFi.status() != WL_CONNECTED) {  
    delay(500);  
    Serial.print(".");  
  }  
  Serial.println("");  
  Serial.println("WiFi connected");  
  Serial.println("IP address: ");  
  Serial.println(WiFi.localIP());  
  client.setServer(mqtt_server, mqtt_port);  
  client.setCallback(callback);  
  pinMode(ldr, INPUT);  
  dht.begin();  
  pinMode(LED1, OUTPUT);  
  digitalWrite(LED1, 1);  
}
```

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

3

ส่วนที่ 3 ส่วนของฟังก์ชันหลัก

ฟังก์ชันการทำงานหลัก

```
void loop() {  
  if (!client.connected()) {  
    reconnect();  
  }  
  client.loop();  
  
  long now = millis();  
  if (now - lastMsg > 5000) {  
    float ldr_data = analogRead(ldr);  
    int lux = int((250.000000 / (ADC_value * ldr_data)) - 50.000000);  
  
    int humidity = dht.readHumidity();  
    int temperature = dht.readTemperature();  
  
    lastMsg = now;  
    String data = "{\"data\": {\"light\": " + String(lux) + ", \"temperature\": " + String(temperature) + ", \"humidity\":  
    \"\" + String(humidity) + "\"}}";  
    Serial.println(data);  
    data.toCharArray(msg, (data.length() + 1));  
    client.publish("@shadow/data/update", msg);  
  }  
  delay(1);  
}
```

ใน void loop นั้นส่วนของการส่งข้อมูลจากเซนเซอร์
ต่างๆได้นำ millis มาครอบการทำงานทั้งหมดเพื่อไม่ให้เกิด
การหน่วงเวลาการทำงานของโปรแกรม

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

The screenshot shows the Arduino IDE interface. The code editor displays the following code:

```
66 void setup() {  
67   Serial.begin(9600);  
68   Serial.println();  
69   Serial.print("Connecting to ");  
70   Serial.println(ssid);
```

The Serial Monitor window shows the following output:

```
09:46:24.308 -> {"data": {"light":141, "temperature":24, "humidity": 52}}  
09:46:29.322 -> {"data": {"light":136, "temperature":23, "humidity": 58}}  
09:46:34.325 -> {"data": {"light":165, "temperature":25, "humidity": 54}}  
09:46:39.300 -> {"data": {"light":122, "temperature":24, "humidity": 50}}  
09:46:44.313 -> {"data": {"light":165, "temperature":23, "humidity": 57}}  
09:46:49.327 -> {"data": {"light":121, "temperature":25, "humidity": 52}}  
09:46:54.329 -> {"data": {"light":154, "temperature":23, "humidity": 53}}  
09:46:59.309 -> {"data": {"light":122, "temperature":23, "humidity": 55}}  
09:47:04.319 -> {"data": {"light":155, "temperature":25, "humidity": 56}}  
09:47:09.327 -> {"data": {"light":128, "temperature":24, "humidity": 53}}  
09:47:14.327 -> {"data": {"light":128, "temperature":25, "humidity": 58}}  
09:47:19.298 -> {"data": {"light":151, "temperature":24, "humidity": 58}}  
09:47:24.312 -> {"data": {"light":159, "temperature":24, "humidity": 55}}  
09:47:29.326 -> {"data": {"light":155, "temperature":25, "humidity": 54}}
```

Annotations in the image:

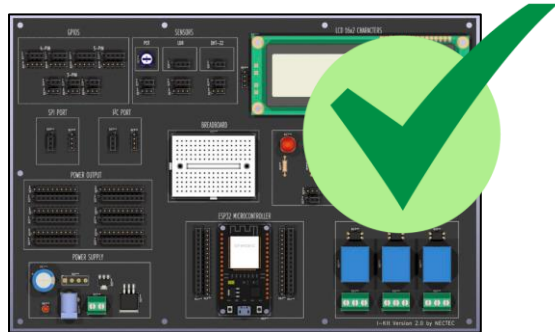
- A box labeled "ตรวจสอบที่ Serial Monitor" (Check at Serial Monitor) points to the Serial Monitor window.
- A box labeled "ข้อความที่ถูกส่งไป" (Message sent) points to the output data in the Serial Monitor.

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

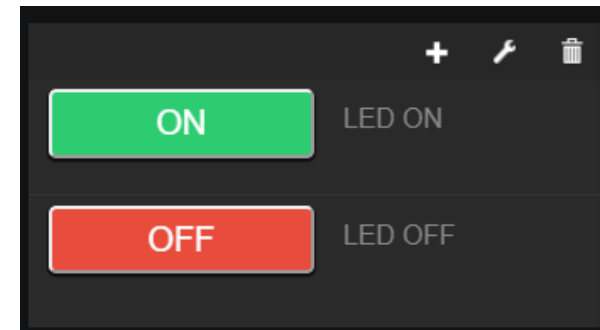
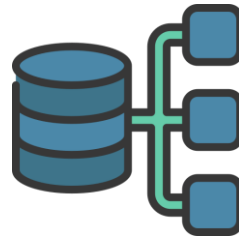
เช็คความพร้อมเพื่อทดสอบทั้งหมด 3 ส่วน

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



ESP32

Device Schema



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

มาที่ Device Schema

```
1 {
2   "additionalProperties": false,
3   "properties": {
4     "humidity": {
5       "operation": {
6         "store": {
7           "ttl": "7d"
8         }
9       },
10      "type": "number"
11    },
12    "light": {
13      "operation": {
14        "store": {
15          "ttl": "7d"
16        }
17      },
18      "type": "number"
19    },
20    "temperature": {
21      "operation": {
22        "store": {
23          "ttl": "30d"
24        }
25      },
26      "transform": {
27        "expression": "($.temperature * 1.8) + 32"
28      }
29    }
30  }
31 }
```

Ln: 1 Col: 1

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

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

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

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

The screenshot shows a JSON Schema editor with tabs for 'Shadow', 'Schema', 'Trigger', 'Feed', and 'i'. The 'Schema' tab is active. The tree view shows a root object with properties: 'additionalProperties' (false), 'humidity', 'light', 'temperature', and 'led'. The 'led' property is highlighted with a red box. A red arrow points from a text box to the 'led' property.

```
object {2}
  additionalProperties : false
  properties {4}
    humidity {2}
      operation {1}
        store {1}
          ttl : 7d
          type : number
    light {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
```

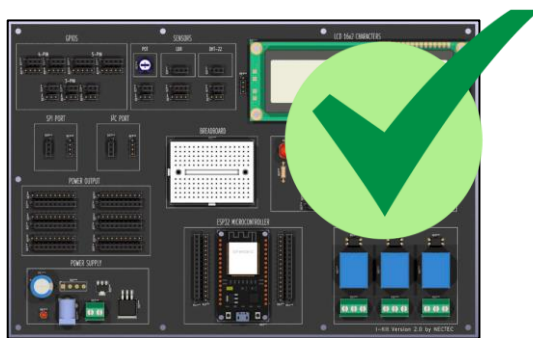
Schema ที่ถูกเพิ่มเข้ามา

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

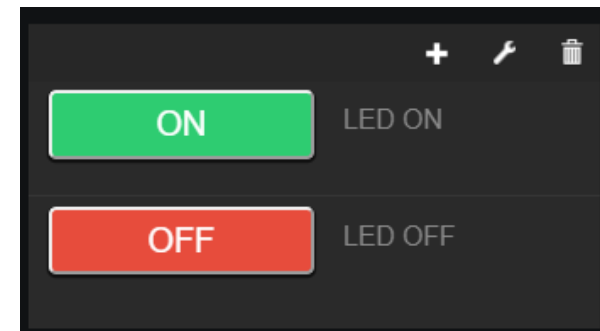
เช็คความพร้อมเพื่อทดสอบทั้งหมด 3 ส่วน

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



ESP32

Device Schema

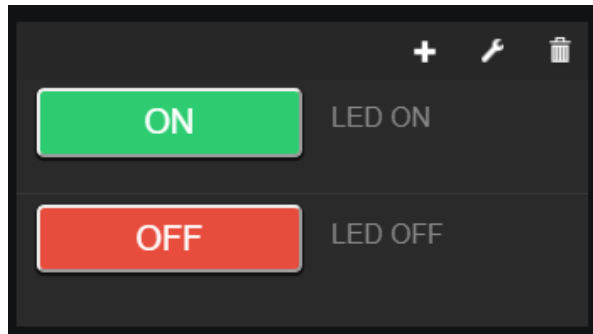


ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

การสร้าง Widget : Button

ตัวอย่าง Widget : Button



เป็น Widget สำหรับควบคุมแบบปุ่มกด

WIDGET

A simple button widget that can perform Javascript action.

คำหรือประโยคที่อยู่บนปุ่ม

TYPE Button

BUTTON CAPTION

LABEL TEXT

คำอธิบายของปุ่ม

BUTTON COLOR Red

สีของปุ่ม

ONCLICK ACTION + DATASOURCE .JS EDITOR

ONCREATED ACTION

SAVE CANCEL

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

WIDGET

A simple button widget that can perform Javascript action.

TYPE

BUTTON CAPTION

LABEL TEXT

BUTTON COLOR

ONCLICK ACTION + DATASOURCE .JS EDITOR

ONCREATED ACTION

SAVE CANCEL

กรอกข้อมูลต่างๆและกำหนดสีของปุ่ม

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ 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")` + DATASOURCE JS EDITOR

ONCREATED ACTION: JS code to run

SAVE CANCEL

ONCLICK ACTION คือ รูปแบบข้อความหรือข้อมูลที่ต้องการส่งออกเมื่อคลิกที่ปุ่ม โดยรูปแบบในตัวอย่างนี้คือ

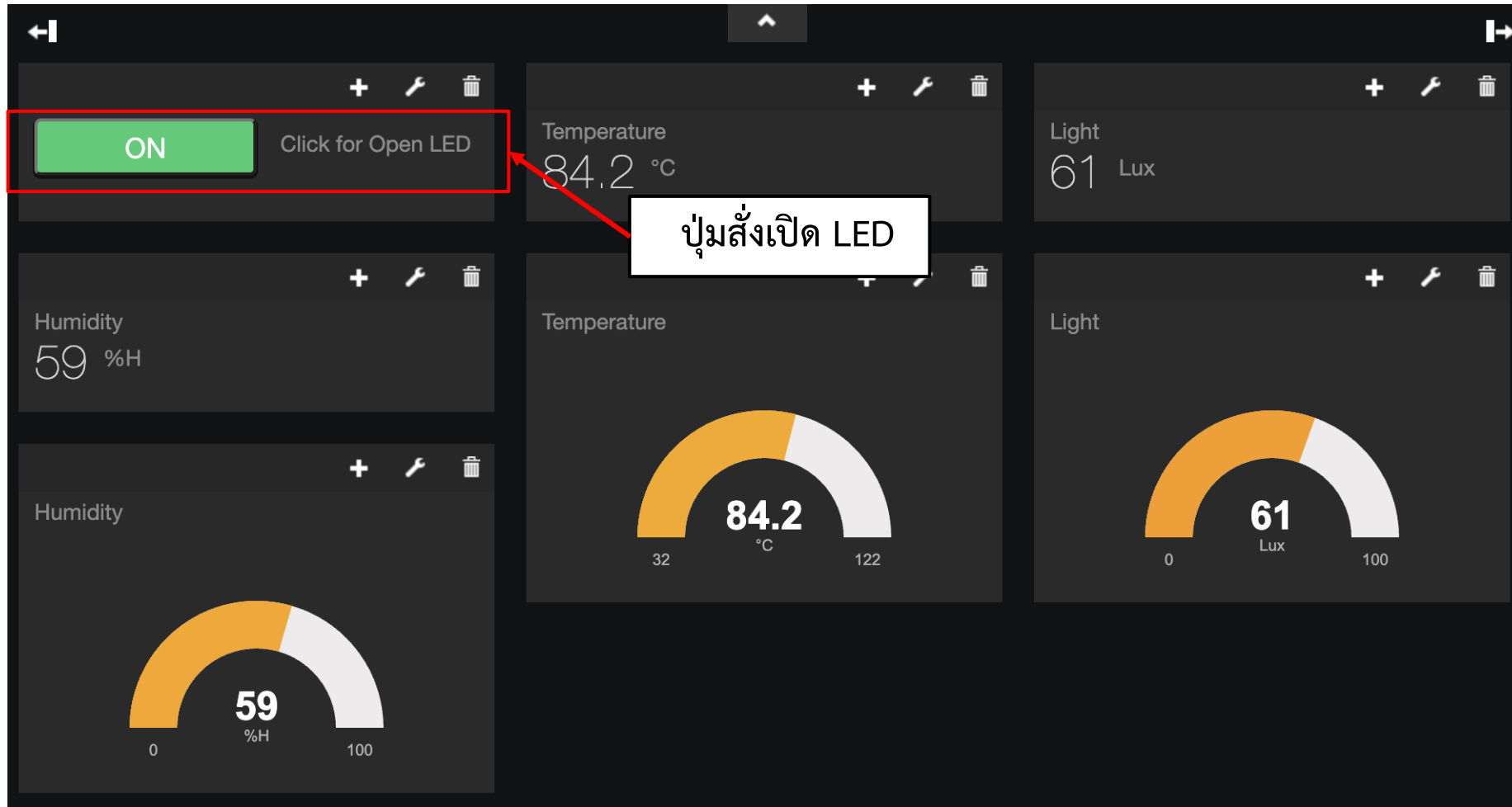
```
netpie["Device2"].publish("@msg/led","on")
```

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

หลังจากกด SAVE ปุ่มจะถูกทำงาน 1 ครั้ง นั้นหมายความว่า หากตั้งค่าถูกต้อง LED จะติด

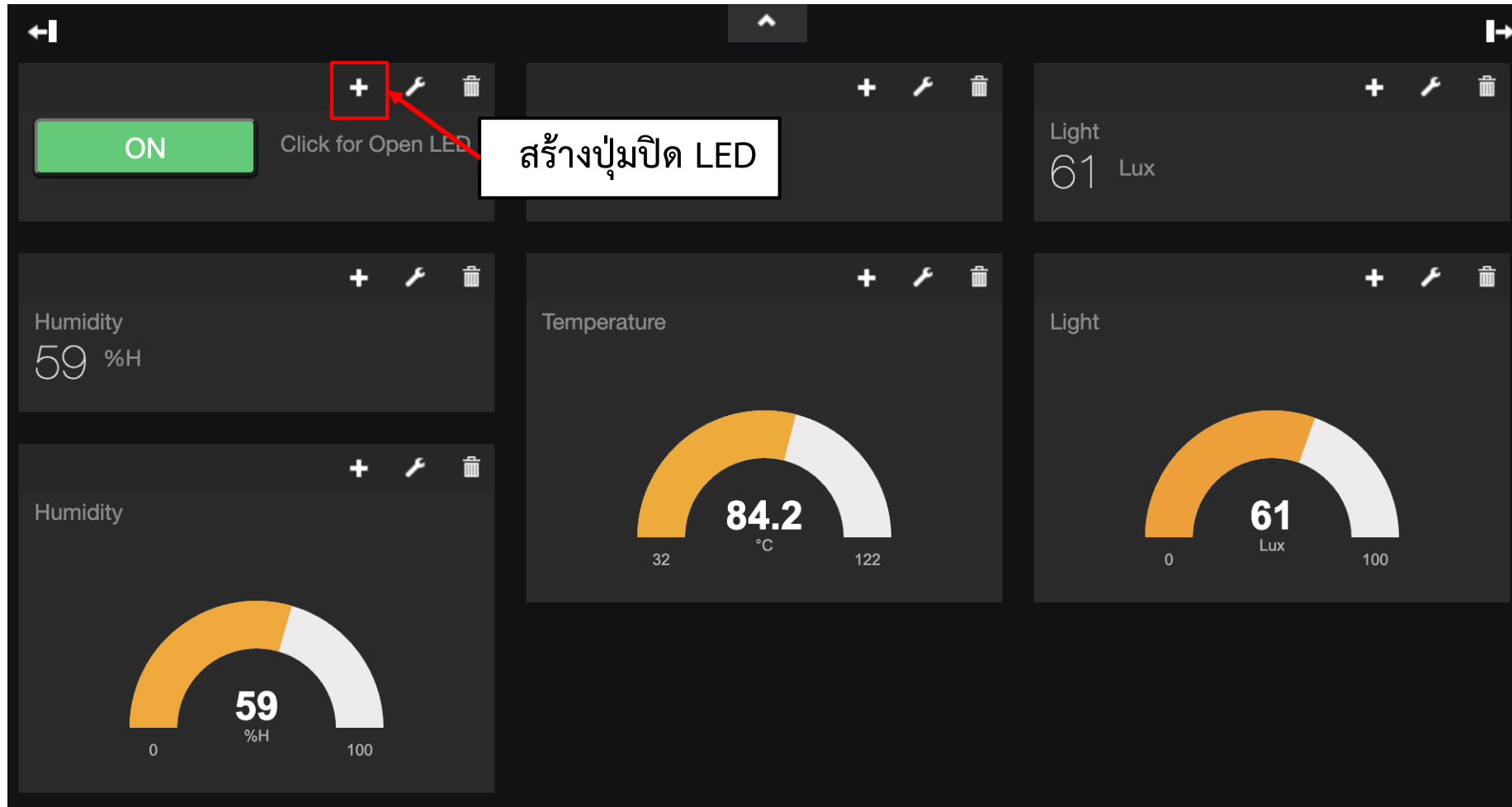
ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button



ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

WIDGET

A simple button widget that can perform Javascript action.

TYPE

BUTTON CAPTION

LABEL TEXT

BUTTON COLOR

ONCLICK ACTION + DATASOURCE .JS EDITOR

Add some Javascript here.

ONCREATED ACTION

JS code to run after a button is created

SAVE CANCEL

กรอกข้อมูลต่างๆและกำหนดสีของปุ่ม

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

WIDGET

A simple button widget that can perform Javascript

TYPE

BUTTON CAPTION

LABEL TEXT

BUTTON COLOR

ONCLICK ACTION + DATASOURCE JS EDITOR

ONCREATED ACTION

JS code to r

SAVE CANCEL

ONCLICK ACTION คือ รูปแบบข้อความหรือข้อมูลที่ต้องการส่งออกเมื่อคลิกที่ปุ่ม โดยรูปแบบในตัวอย่างนี้คือ

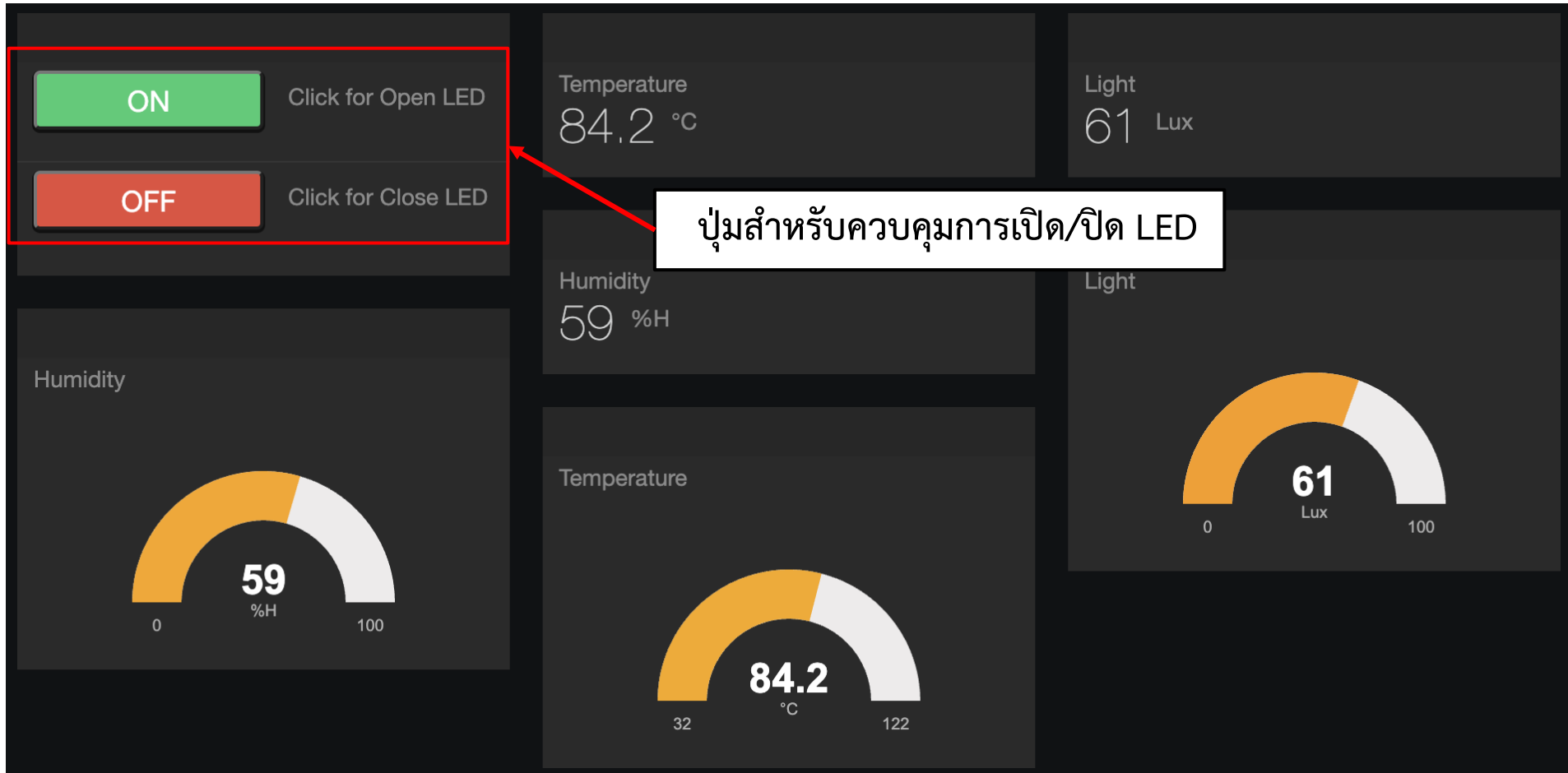
```
netpie["Device2"].publish("@msg/led","off")
```

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

หลังจากกด SAVE ปุ่มจะถูกทำงาน 1 ครั้ง นั้นหมายความว่า หากตั้งค่าถูกต้อง LED จะดับ

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 1 ทดสอบการควบคุม LED ที่ ESP32 โดยใช้ Widget Button

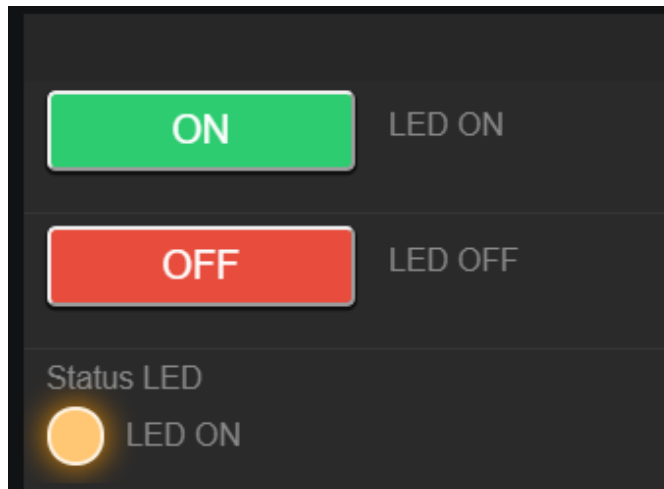


ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 2 สร้าง Widget แสดงสถานะของ LED ด้วย Widget Indicator

การสร้าง Widget : Indicator

ตัวอย่าง Widget : Indicator



เป็น Widget สำหรับแสดงผลสถานะการทำงาน

คำกำกับหรือข้อความที่ใช้บอกข้อมูล

ชื่อของ Indicator

คำอธิบายสถานะต่างๆ ของ Indicator

SAVE CANCEL

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 2 สร้าง Widget แสดงสถานะของ LED ด้วย Widget Indicator

WIDGET

TYPE

TITLE

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

VALUE + DATASOURCE

ON TEXT + DATASOURCE .JS EDITOR

OFF TEXT + DATASOURCE .JS EDITOR

SAVE CANCEL

ตั้งค่าส่วนต่างๆ

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 2 สร้าง Widget แสดงสถานะของ LED ด้วย Widget Indicator

WIDGET

กำหนด Value ดังนี้

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

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

TYPE In

TITLE St

DEFAULT COLOR #ADFF2F
enter the color e.g. #ff0000,red or leave blank for the default color set

VALUE `(datasources["Device2"]["shadow"]["led"]) == "on"` + DATASOURCE ✕ .JS EDITOR

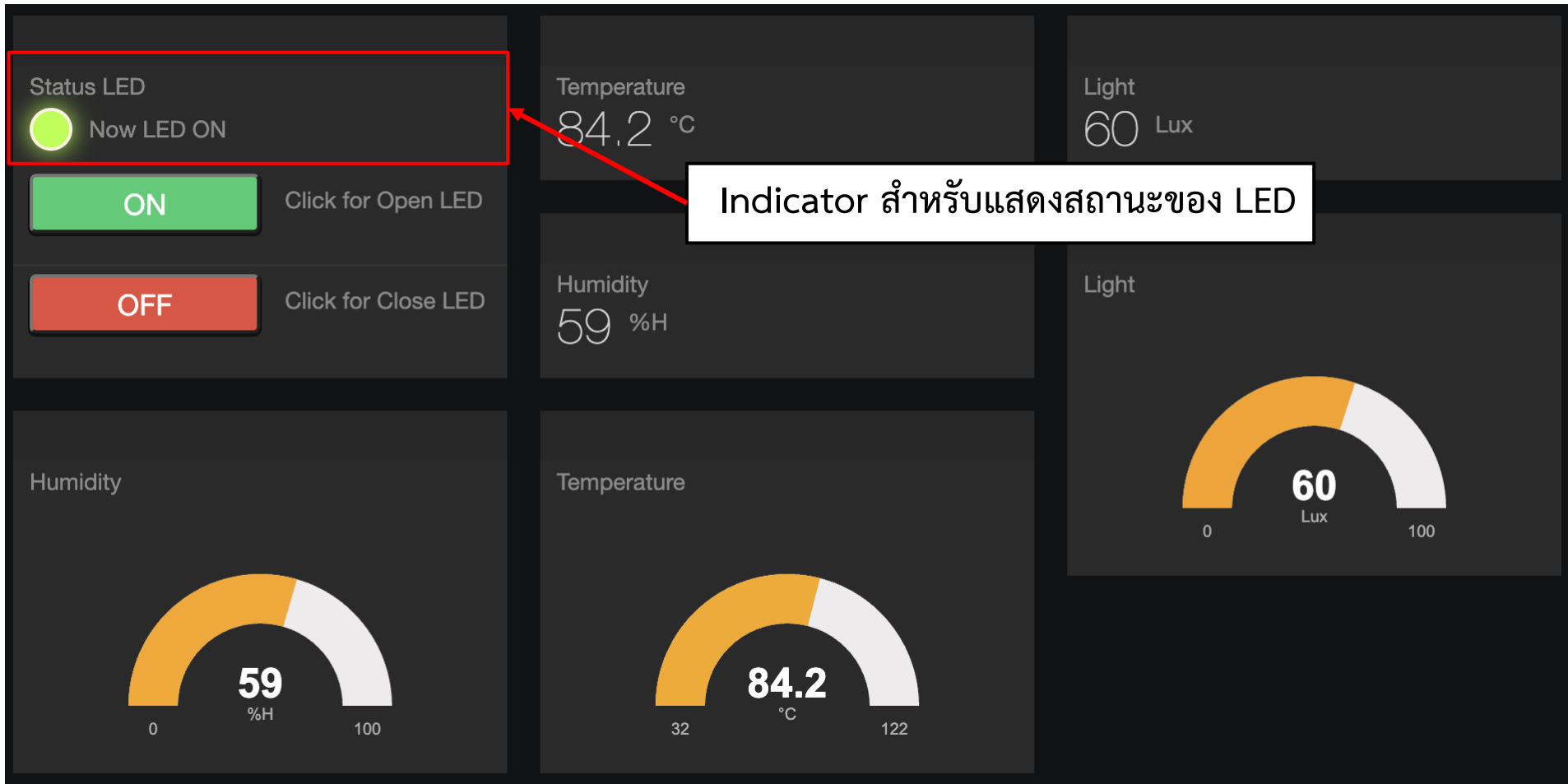
ON TEXT Now LED ON + DATASOURCE ✕ .JS EDITOR

OFF TEXT Now LED OFF + DATASOURCE ✕ .JS EDITOR

SAVE CANCEL

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 2 สร้าง Widget แสดงสถานะของ LED ด้วย Widget Indicator

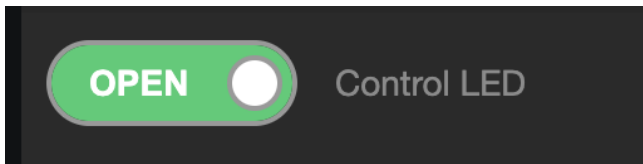


ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle

การสร้าง Widget : Toggle

ตัวอย่าง Widget : Toggle



เป็น Widget สำหรับควบคุมและแสดงผลสถานะการทำงาน

A simple toggle widget that can perform Javascript action.

คำอธิบายของปุ่ม

TYPE Toggle

TOGGLE CAPTION

TOGGLE STATE

+ DATASOURCE JS EDITOR

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

ON TEXT

ON

คำอธิบายบนปุ่มในแต่ละสถานะ

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

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle

WIDGET

A simple toggle widget that can perform Javascript action.

TYPE: Toggle

TOGGLE CAPTION: Control LED

TOGGLE STATE: + DATASOURCE JS EDITOR

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

ON TEXT: OPEN

OFF TEXT: CLOSE

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

ตั้งค่าส่วนต่างๆ

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle

กำหนด Toggle State ดังนี้

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

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

WIDGET

A simple toggle

TOGGLE CAPTION Control LED

TOGGLE STATE `(datasources["Device2"]["shadow"]["led"]) == "on"` + DATASOURCE .JS EDITOR

ON TEXT OPEN

OFF TEXT CLOSE

ONTOGGLEON ACTION

ONTOGGLEOFF ACTION

ONCREATED ACTION

SAVE CANCEL

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle

WIDGET

A simple toggle widget that can be used to control a device or display an LED.

TYPE

TOGGLE CAPTION

TOGGLE STATE

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

ON TEXT

OFF TEXT

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

ONTOGGLEON ACTION คือ รูปแบบข้อความหรือข้อมูลที่ต้องการส่งออกเมื่อคลิกที่ปุ่ม โดยรูปแบบในตัวอย่างนี้คือ

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

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle

WIDGET

A simple toggle widget that can perform Javascript action.

TYPE:

TOGGLE CAPTION:

TOGGLE STATE:

ON TEXT:

OFF TEXT:

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

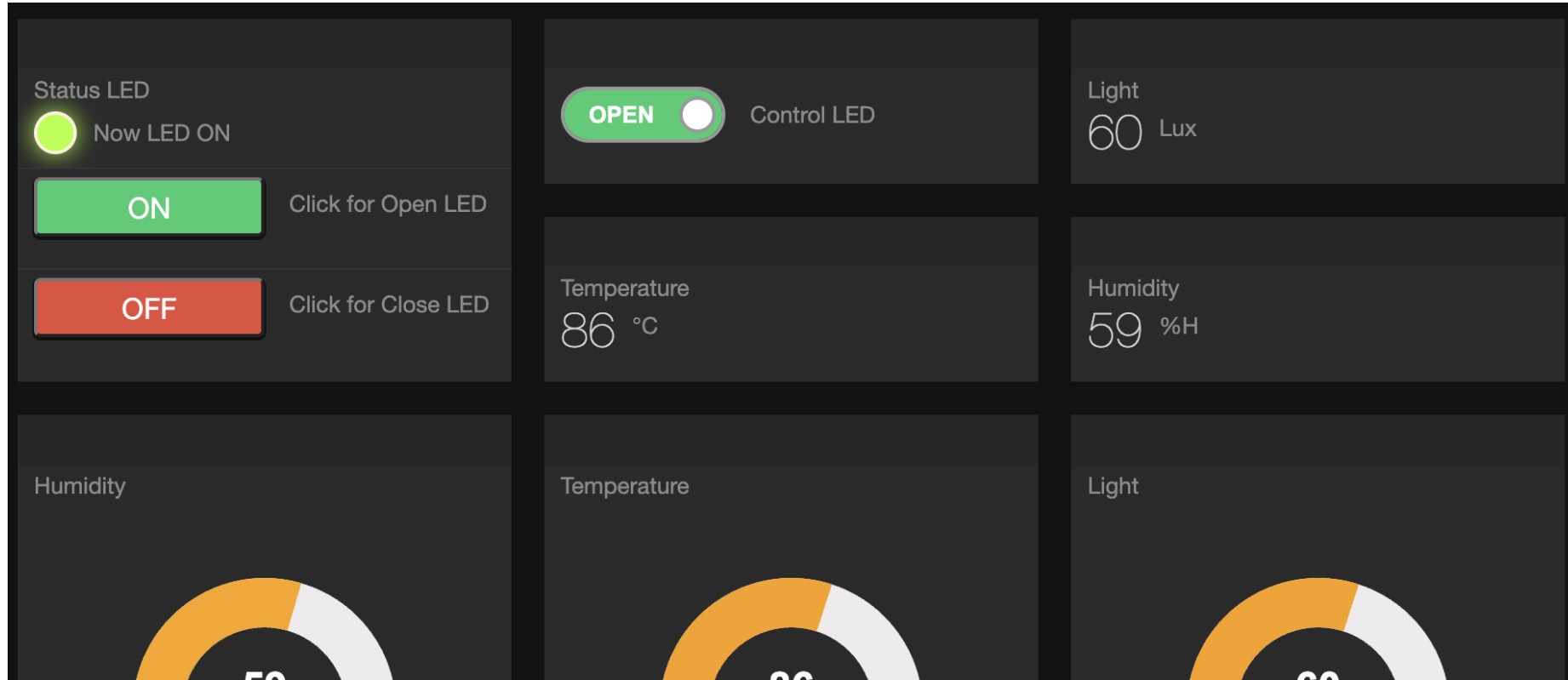
ONTOGGLEOFF ACTION คือ รูปแบบข้อความหรือข้อมูลที่ต้องการส่งออกเมื่อคลิกที่ปุ่ม โดยรูปแบบในตัวอย่างนี้คือ

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

โดยที่ Device2 คือ ชื่อของ Datasource ดังนั้นต้องใช้ชื่อ Datasource ให้ตรง

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle



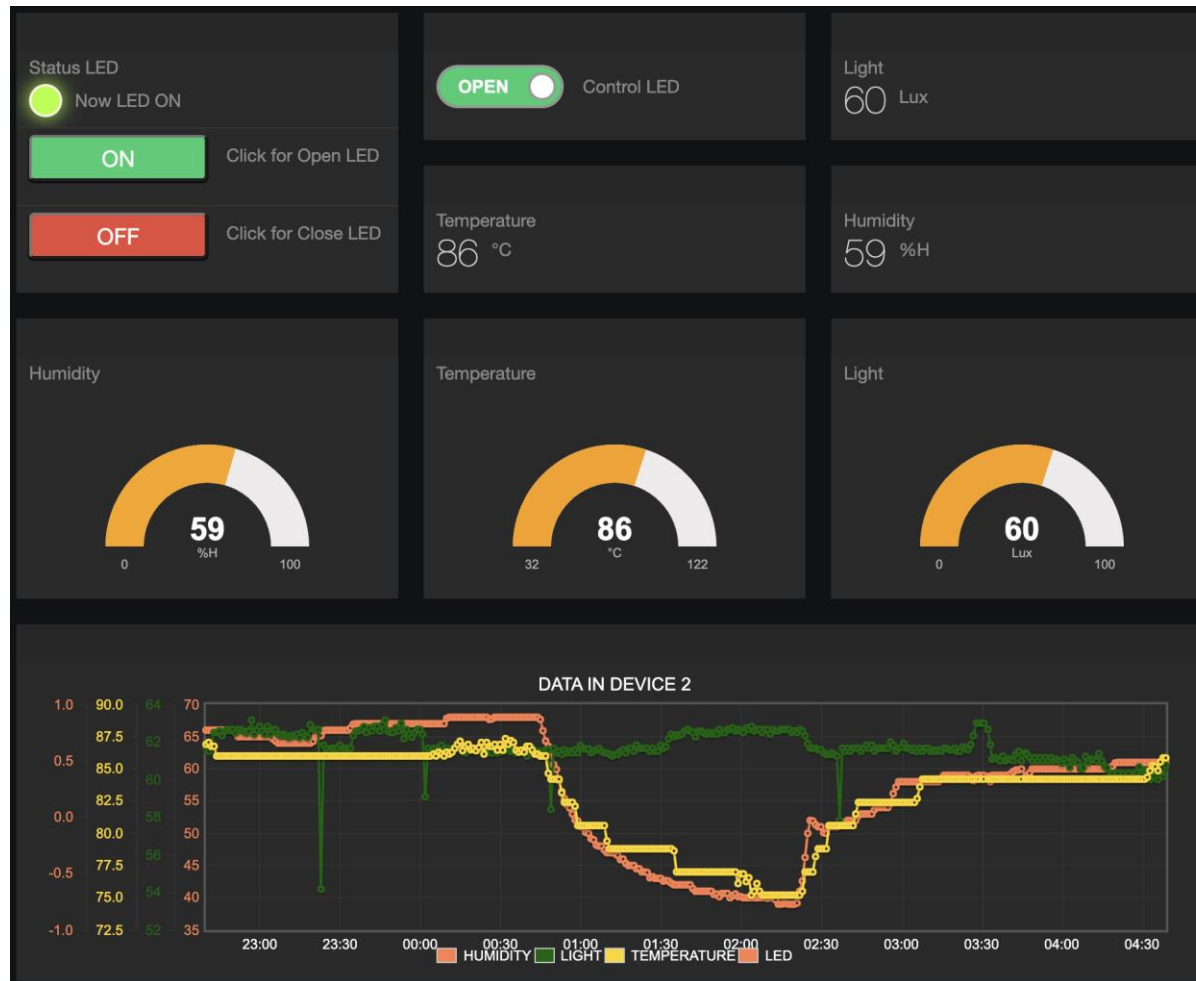
ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle

The screenshot shows the NETPIE Freeboard interface. In the top-left corner, there is a menu with options: IMPORT, EXPORT, RESET, + ADD PANE, and SAVE. The 'SAVE' button is highlighted with a red box and a hand cursor. A teal callout box with the text 'การ Save Freeboard' points to this button. The top-right corner shows a 'DATASOURCES' table with one entry: 'Device2' with a 'Last Updated' time of '4:39:13 AM'. The main dashboard area contains several widgets: 'Status LED' (Now LED ON), 'Control LED' (OPEN toggle), 'Light' (61 Lux), 'Temperature' (86 °C), and 'Humidity' (59 %H).

ใบงานที่ 5.2 ขั้นตอนการทดลอง

การทดลองที่ 3 ทดสอบการควบคุมและแสดงผล LED ด้วย Widget Toggle



ใบงานที่ 5.2 คำถามท้ายหน่วยการเรียนรู้

คำถามท้ายหน่วยการเรียนรู้ที่ 4

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

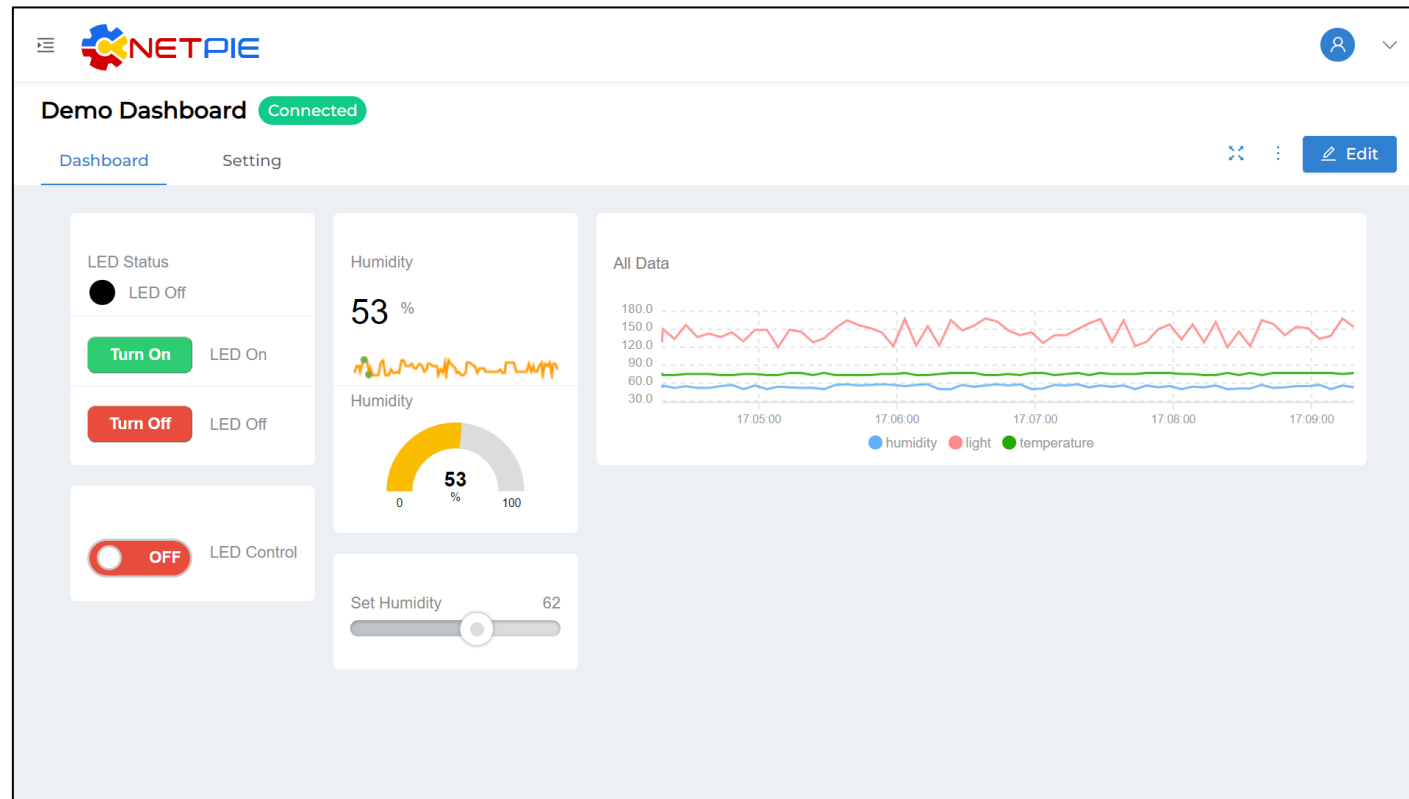
1. Widget : Text แสดงผลข้อมูล voltage และ gas
2. Widget : Gauge แสดงผลข้อมูล voltage และ gas
3. Widget : FeedView แสดงผลข้อมูล voltage และ gas

ข้อ 4 และ 5 เลือกทำเพียงข้อเดียว

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

โดยทุก Widget ของแต่ละข้อมูลกำหนดให้เป็นอิสระจากกัน (นั่นคือ ต้องมี FeedView 2 อัน)

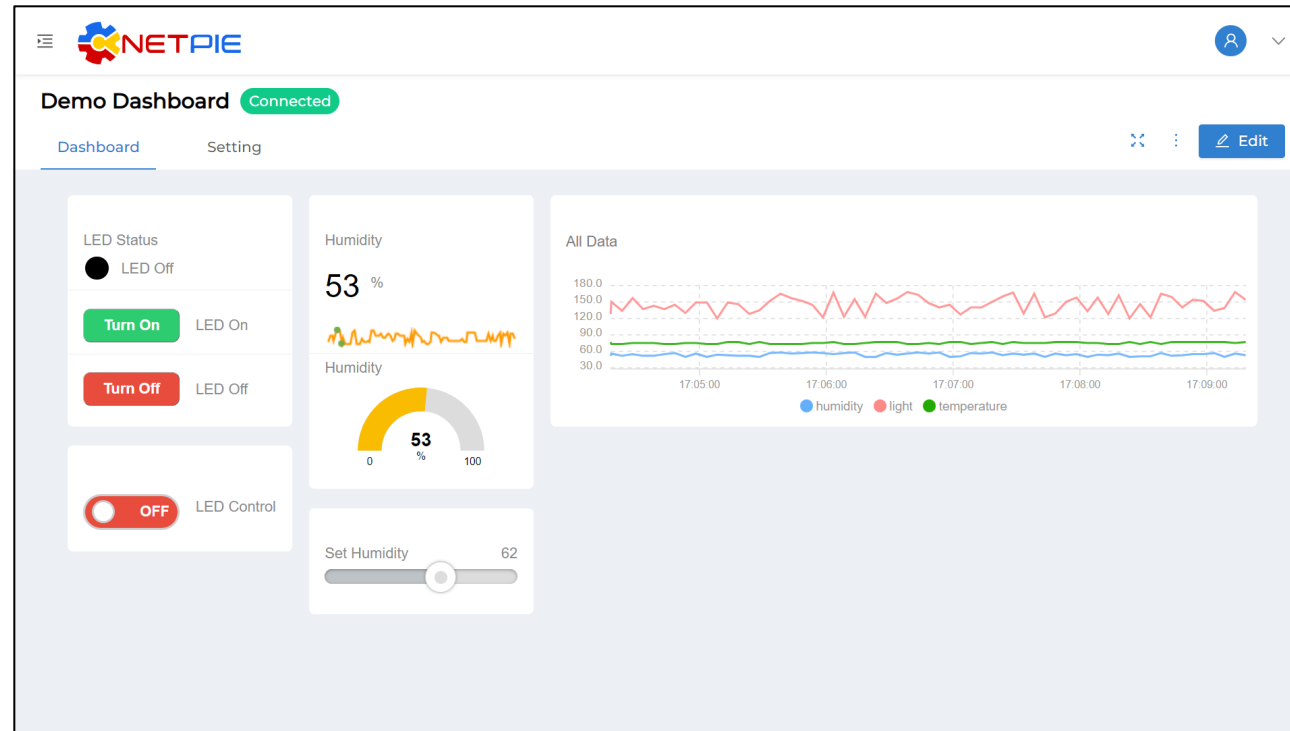
ใบงานที่ 5.3 การแสดงผลและควบคุมอุปกรณ์ด้วย Dashboard



ใบงานที่ 5.3 ทฤษฎีเบื้องต้น

Dashboard คืออะไร

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



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource

การสร้าง Dashboard

The screenshot displays a web application interface for creating a dashboard. On the left, a sidebar contains navigation options: Overview, Device, Group, Event Hook, Console, Freeboard, and Dashboard (highlighted with a red box and labeled '1'). The main content area shows the 'Dashboard' page for the 'NETPIE_training' project. It includes a '+ Create' button (highlighted with a red box and labeled '2'), a search input field, and a table with columns 'Name', 'Device', and 'Create Date'. The table currently displays 'No Data'.

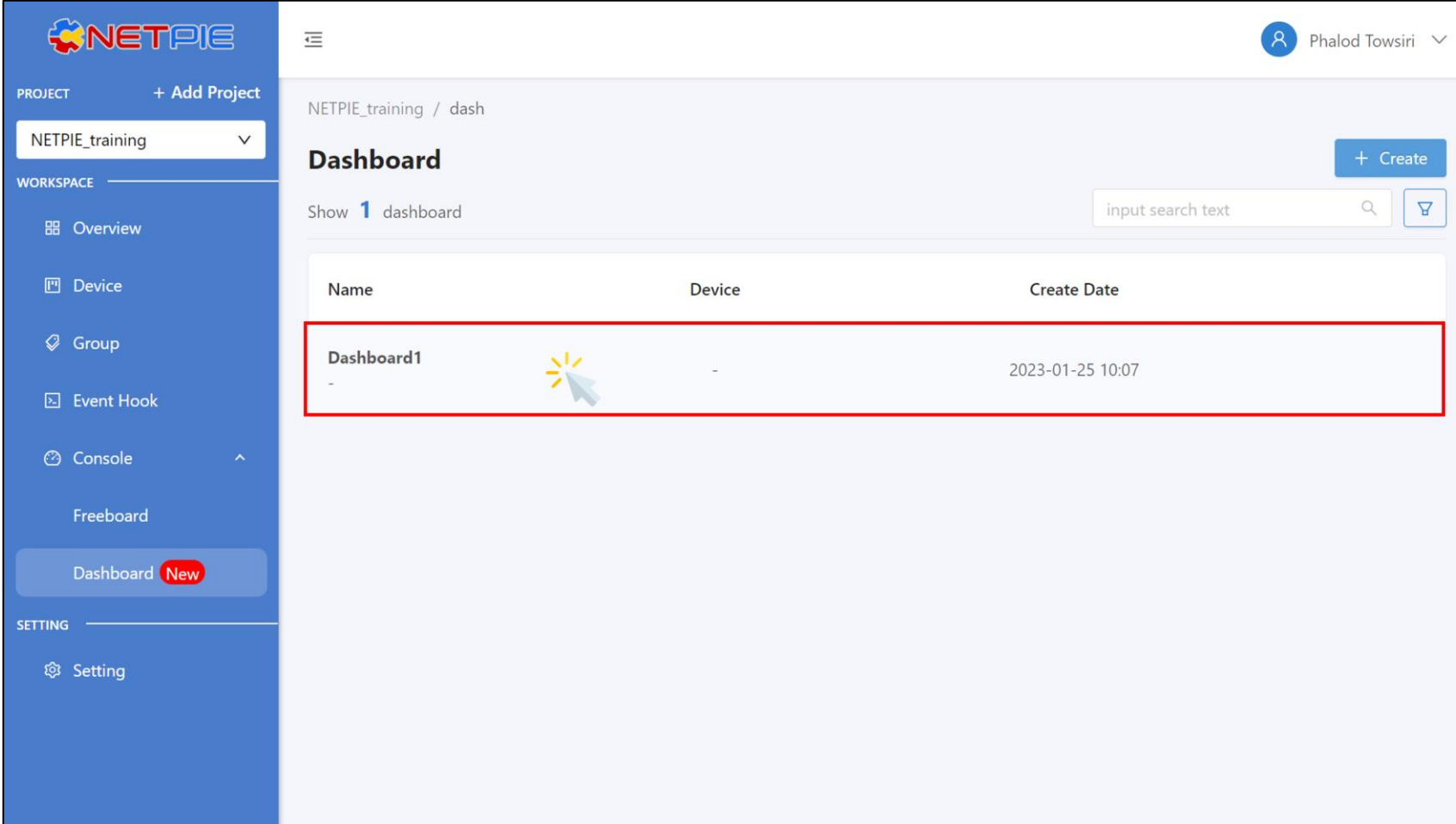
ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource

The screenshot shows the NETPIE web interface. On the left is a dark blue sidebar with navigation options: PROJECT (+ Add Project), WORKSPACE (Overview, Device, Group, Event Hook, Console, Freeboard, Dashboard **New**), and SETTING (Setting). The main area is titled 'Dashboard' and shows a 'Create Dashboard' modal. The modal has a title 'Create Dashboard' and a close button 'X'. It contains two text input fields: 'Dashboard Name' with the text 'Dashboard1' and 'Dashboard Description' with the text 'Description'. At the bottom of the modal are 'Cancel' and 'SAVE' buttons. A mouse cursor is pointing at the 'SAVE' button. A callout box with a black border and white background contains the Thai text 'ตั้งชื่อให้ Dashboard แล้วกด Save'.

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource

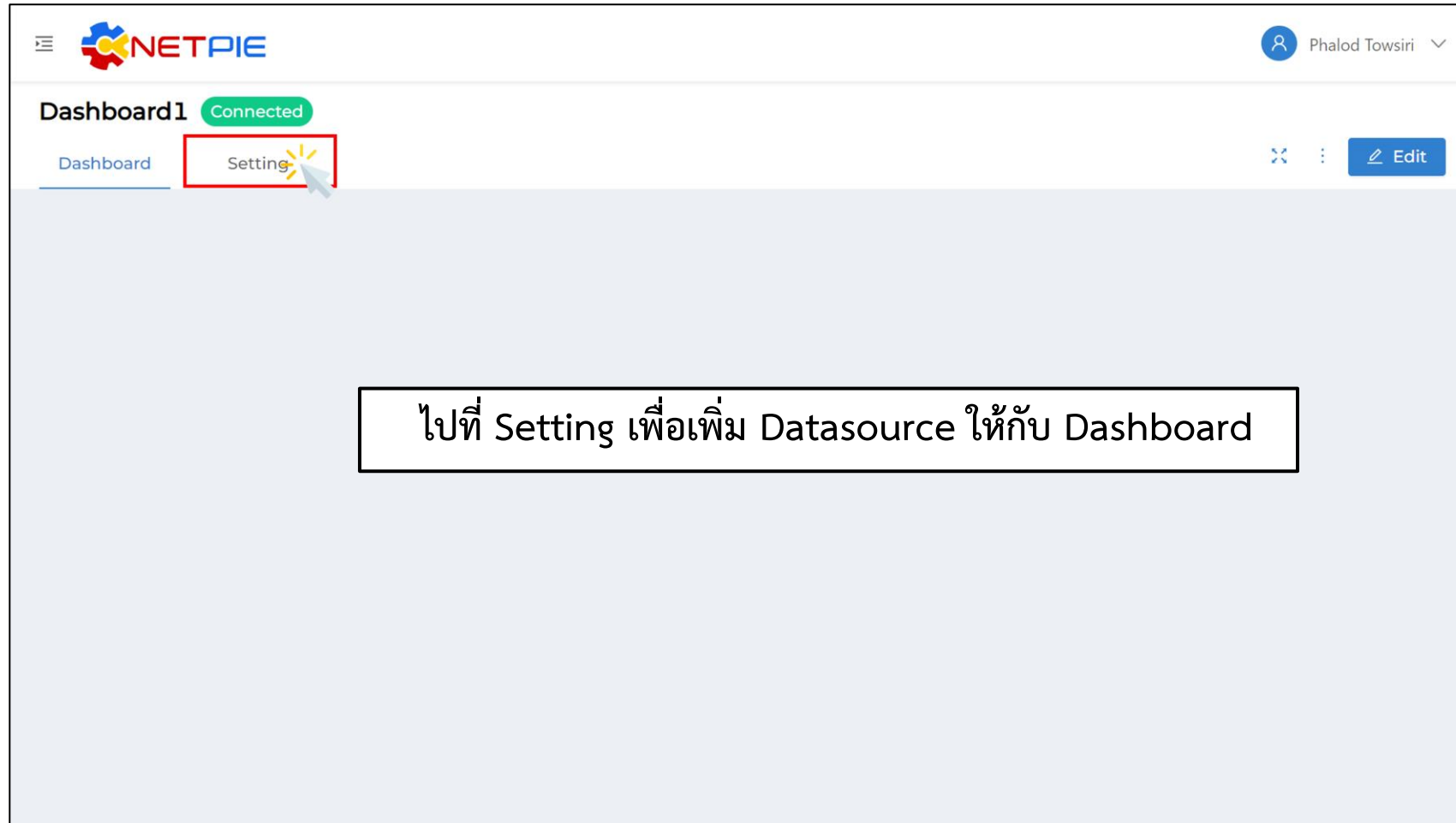


The screenshot displays the NETPIE web interface. On the left is a blue sidebar with navigation options: PROJECT (NETPIE_training), WORKSPACE (Overview, Device, Group, Event Hook, Console, Freeboard, Dashboard **New**), and SETTING (Setting). The main content area shows the 'Dashboard' section with a '+ Create' button and a search bar. Below is a table with the following data:

Name	Device	Create Date
Dashboard1	-	2023-01-25 10:07

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource

NETPIE

Dashboard1 Connected Cancel Save

Dashboard Setting

Dashboard setting

Name
Dashboard1

Description
Description

Show 0 Dashboard datasource + Add device

ID	Alias	Privileges
No device yet You haven't connected any devices to freeboard yet.		

กด Add Device

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource

The screenshot shows the NETPIE dashboard settings for 'Dashboard1'. An 'Add Device' modal is open, allowing the user to select a device and assign privileges. The 'Device' dropdown is set to 'Device2 (cafdc2e3-d856-4d02-b551-ac4d3d88065f)'. The 'Privileges' section includes 'Subscribe Message', 'Publish Message', 'Read Shadow', 'Write Shadow', 'Read Feed', and 'Write Feed'. The background shows the dashboard settings page with a 'No device yet' message at the bottom.

ตรง Device ให้เลือก Device2

Privileges ให้ลองเลือกทั้งหมด

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Dashboard และทดสอบเชื่อม Datasource

NETPIE

Phalod Towsiri

Dashboard1 Connected

Dashboard Setting

Dashboard setting

Name
Dashboard1

Description
Description

Show 1 Dashboard datasource

+ Add device

ID	Alias	Privileges	Create Date
cafdc2e3-d856-4d02-b551-ac4d3d88065f	Device2	R Message W Message R Shadow W Shadow R Feed W Feed	2023-01-25 10:13

จะได้ Datasource ตาม Device ที่เลือก เลยมี Privileges ตามที่ตั้งค่าไว้

ใบงานที่ 5.3 ทฤษฎีเบื้องต้น

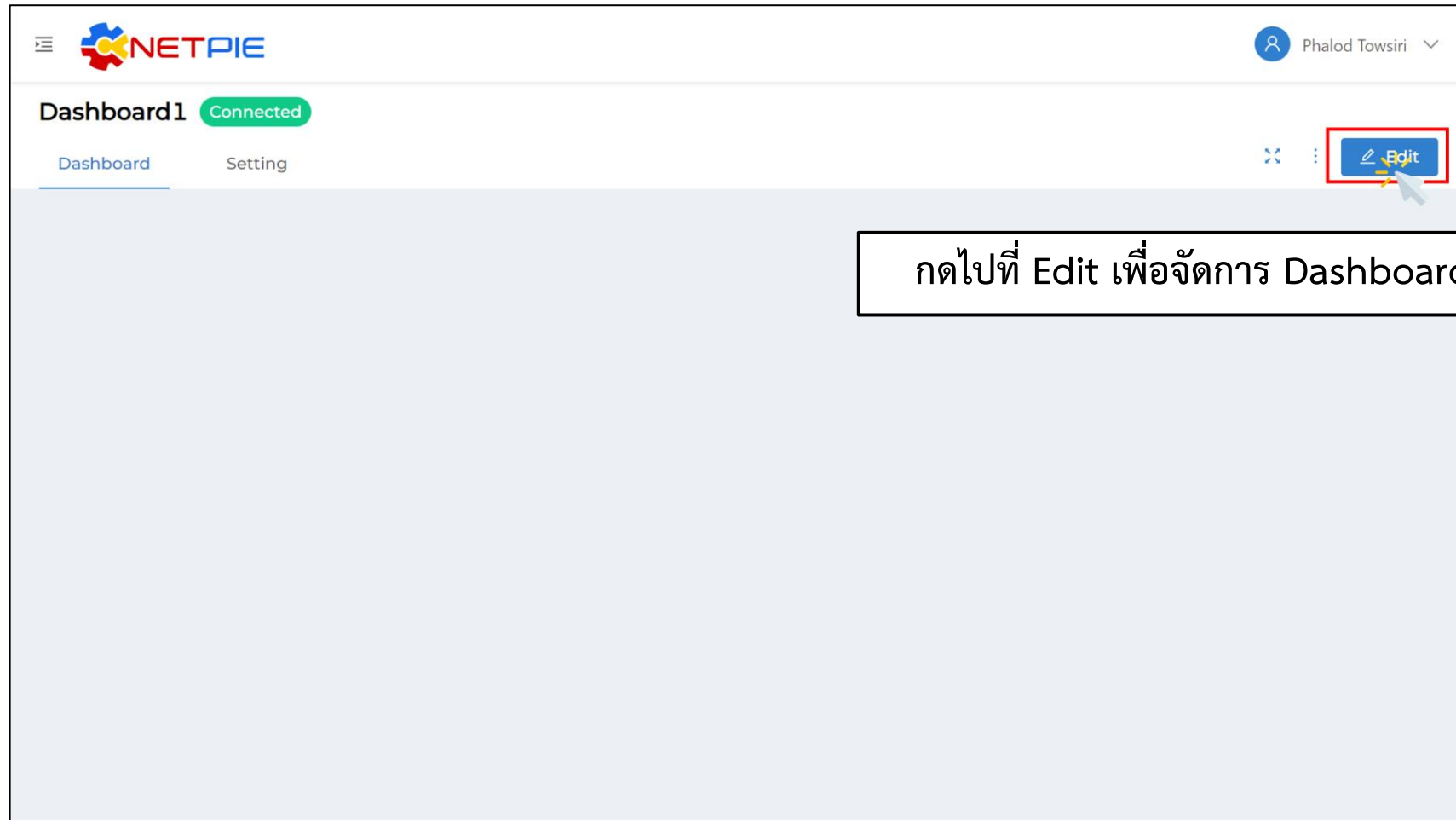
การนำเสนอข้อมูล

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



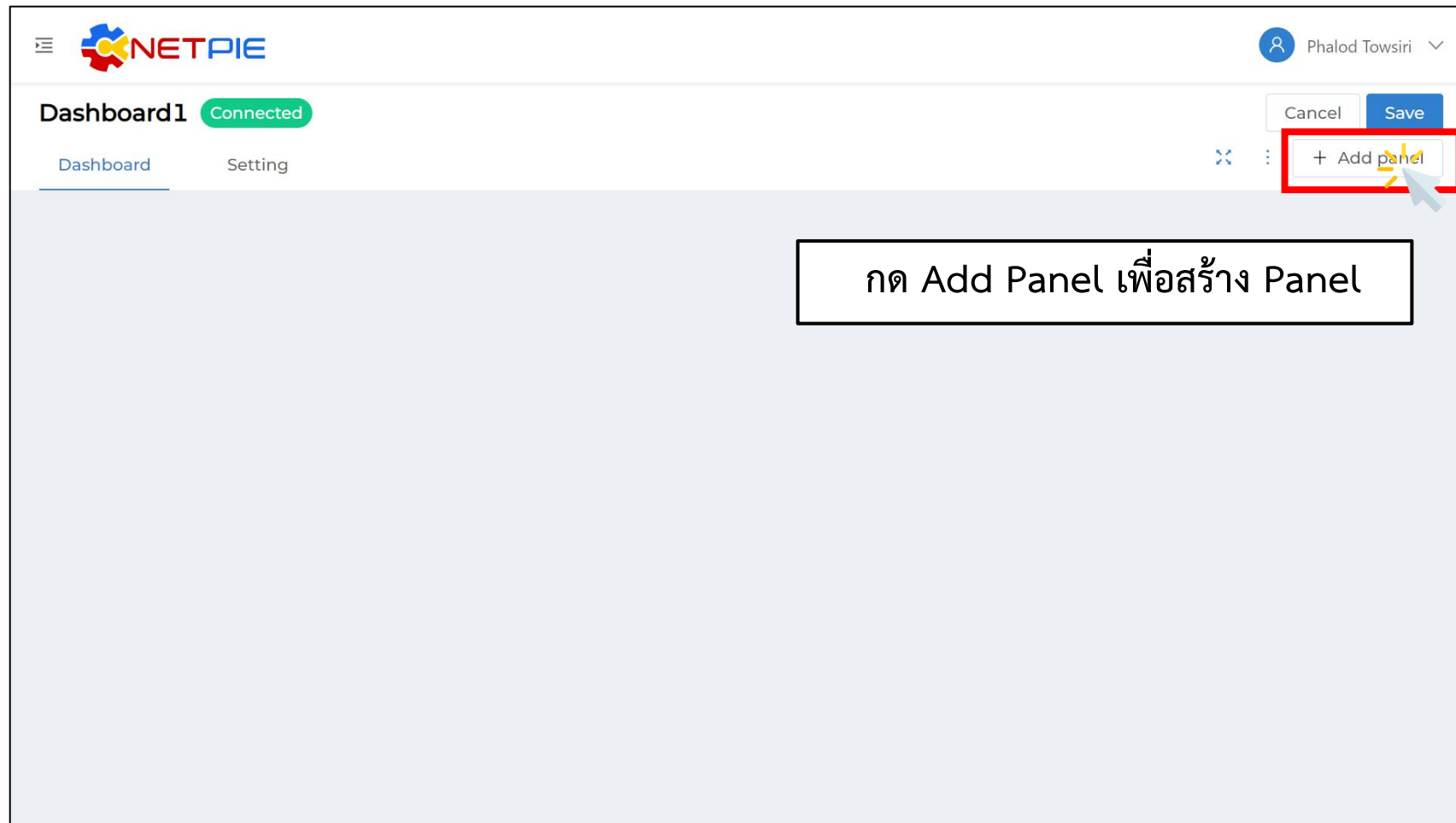
ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

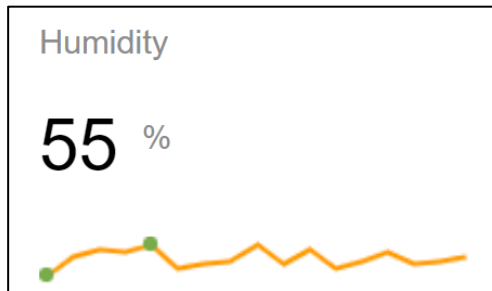


The screenshot shows the NETPIE dashboard interface. At the top left is the NETPIE logo. The main header displays "Dashboard1" with a "Connected" status indicator. Below the header are tabs for "Dashboard" and "Setting". On the right side of the header, there are "Cancel" and "Save" buttons, and a "+ Add panel" button. The main content area shows a panel with a white background. A red box highlights a "+" icon in the top-left corner of the panel, with a mouse cursor pointing at it. To the right of the panel, a text box contains the instruction: "ได้ Panel ขึ้นมา ให้กดปุ่ม + เพื่อเพิ่ม Widget ภายใน Panel".

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text



NETPIE

Dashboard1 Connected

Cancel Save

+ Add panel

Text เป็น Widget ที่นำข้อมูลมาแสดงผลแบบข้อความ

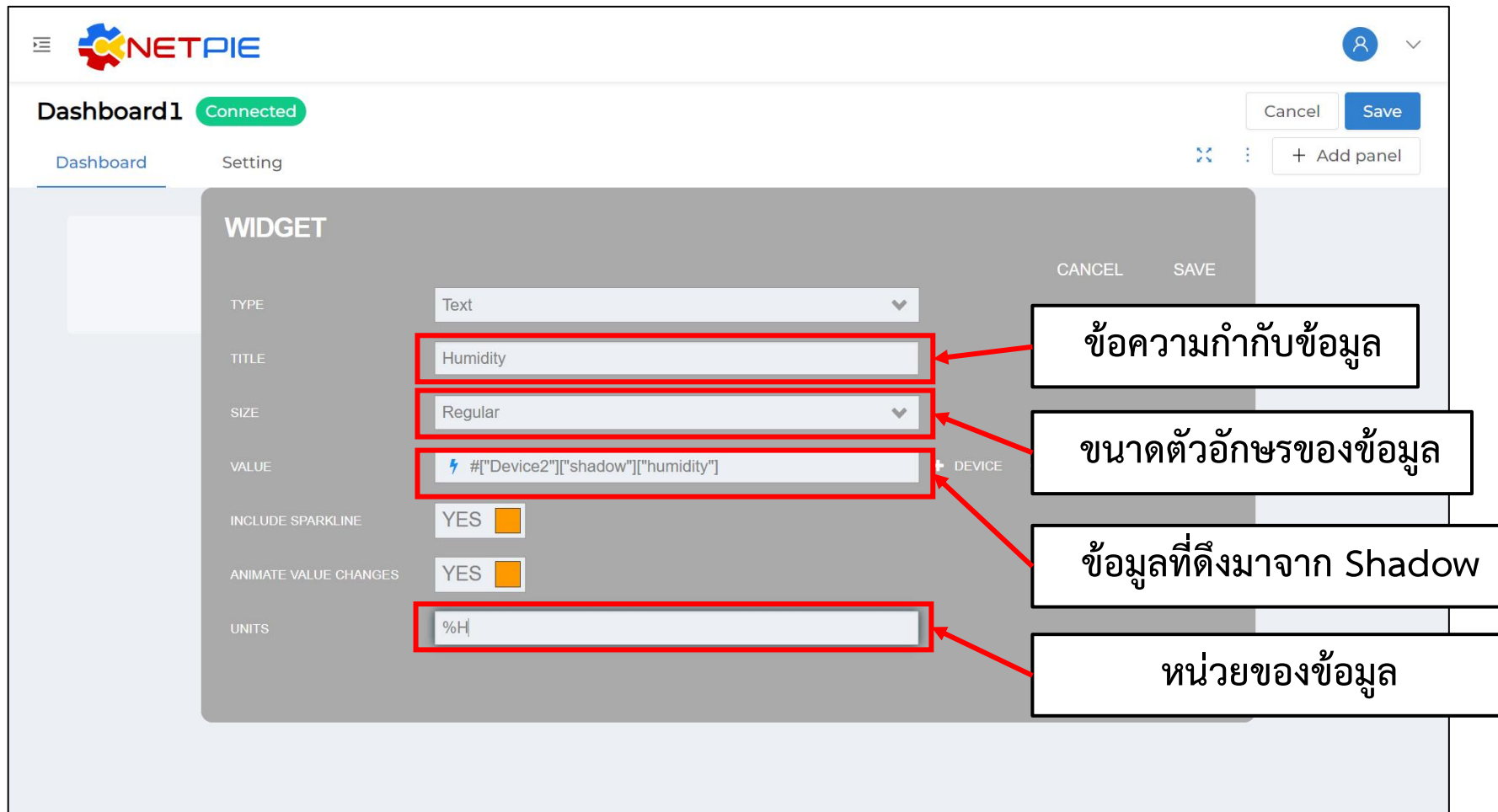
CANCEL SAVE

TYPE	Text	+	DEVICE	+	VARIABLE	✎	EDITOR
TITLE	Humidity						
SIZE	Regular						
VALUE	#["Device2"]["shadow"]["humidity"]						
INCLUDE SPARKLINE	YES	<input checked="" type="checkbox"/>					
ANIMATE VALUE CHANGES	YES	<input checked="" type="checkbox"/>					
UNITS	%H						

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text



The screenshot shows the NETPIE dashboard interface for configuring a Text widget. The widget is titled "Humidity" and is connected to a device named "Device2" at a location named "shadow". The configuration includes a size of "Regular", a value of "#["Device2"]["shadow"]["humidity"]", and units of "%H". The widget is currently displaying "Humidity".

Callouts in Thai explain the fields:

- ข้อความกำกับข้อมูล (Title)
- ขนาดตัวอักษรของข้อมูล (Size)
- ข้อมูลที่ดึงมาจาก Shadow (Value)
- หน่วยของข้อมูล (Units)

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Text



The screenshot shows the NETPIE dashboard interface. At the top left is the NETPIE logo. Below it, the dashboard is titled 'Dashboard1' with a 'Connected' status indicator. There are two tabs: 'Dashboard' (selected) and 'Setting'. In the top right corner, there are 'Cancel' and 'Save' buttons, and a '+ Add panel' button. A 'Humidity' widget is displayed, showing a value of '55 %H' and a small line graph. A text box with a black border is overlaid on the right side of the dashboard, containing the text: 'เมื่อกด Save จะได้ Widget Text ที่แสดงข้อมูล Humidity ของ Device2'.

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



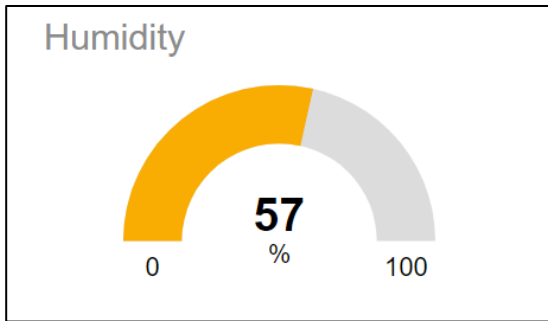
The screenshot shows the NETPIE dashboard interface. At the top left is the NETPIE logo. Below it, the dashboard is titled "Dashboard1" with a "Connected" status indicator. There are tabs for "Dashboard" and "Setting". On the right side, there are buttons for "Cancel", "Save", and "+ Add panel". A "Humidity" widget is displayed, showing a value of "55 %H" and a small line graph. A red box highlights the "+" icon in the top right corner of the widget, indicating where to click to add a new panel.

ต่อไป จะเป็นการเพิ่ม Widget Gauge เข้าไปที่ Panel เดิม โดยกดไปที่ปุ่ม +

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge



NETPIE

Dashboard1 Connected

Cancel Save

Dashboard + Add panel

Gauge เป็น Widget ที่นำข้อมูลมาแสดงผลแบบแถบพลัง

Humidity

50 %H

CANCEL SAVE

TYPE	Gauge
TITLE	Humidity
VALUE	<code>#["Device2"]["shadow"]["humidity"]</code> + DEVICE + VARIABLE EDITOR
UNITS	%H
MINIMUM	0
MAXIMUM	100

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge

The screenshot shows the NETPIE dashboard interface. On the left, a 'Humidity' gauge widget displays '50 %H'. The main area is the 'WIDGET' configuration panel for a 'Gauge' widget. The configuration fields are as follows:

Field	Value
TYPE	Gauge
TITLE	Humidity
VALUE	#[\"Device2\"]['shadow']['humidity']
UNITS	%H
MINIMUM	0
MAXIMUM	100

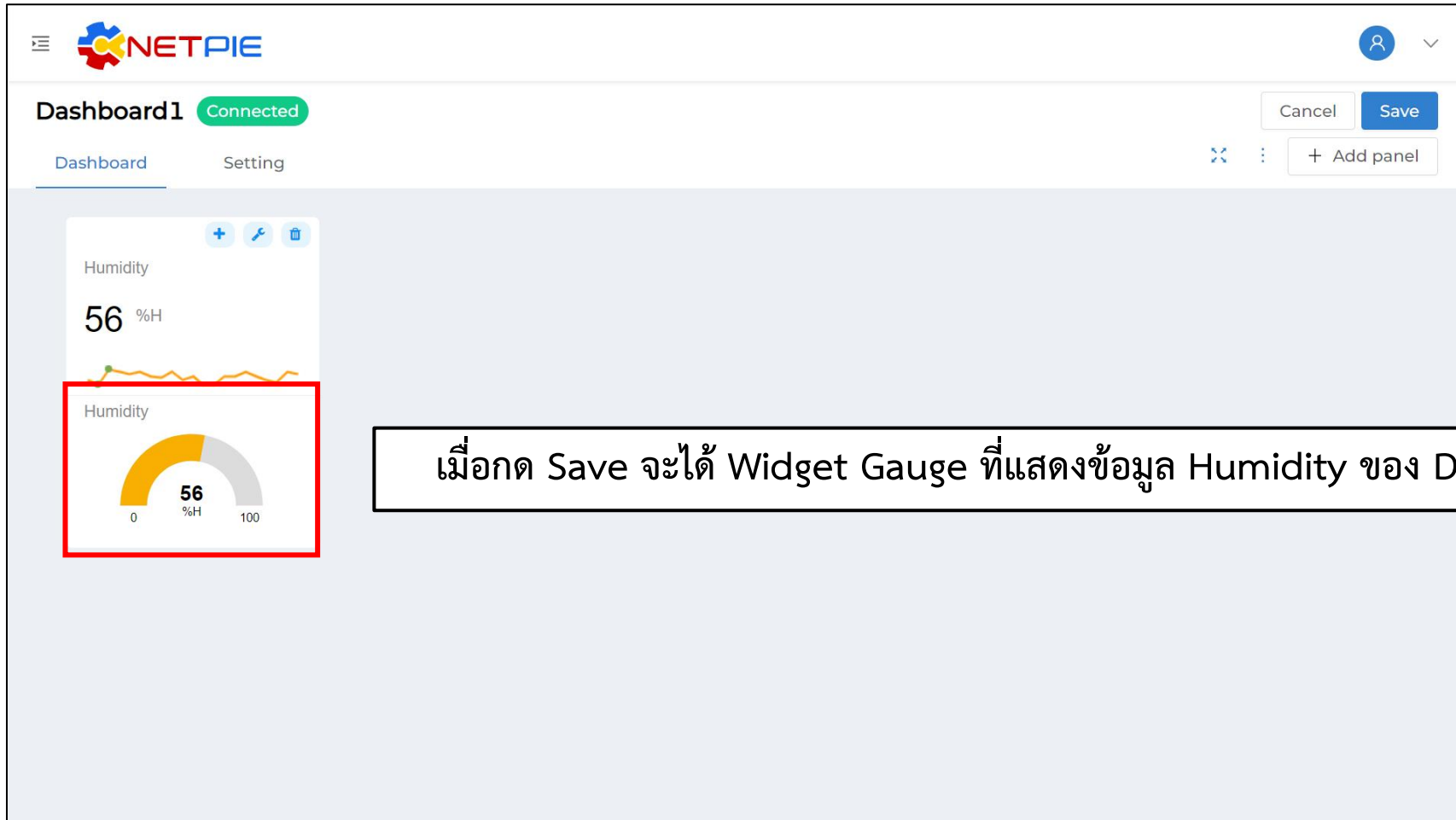
Red boxes highlight the TITLE, VALUE, UNITS, MINIMUM, and MAXIMUM fields. Arrows point from these fields to the following Thai labels:

- ข้อความกำกับข้อมูล (Label for the data)
- ข้อมูลที่ดึงมาจาก Shadow (Data retrieved from Shadow)
- หน่วยของข้อมูล (Data unit)
- ค่าต่ำสุดและสูงสุดของ Gauge (Minimum and maximum values of the Gauge)

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Gauge

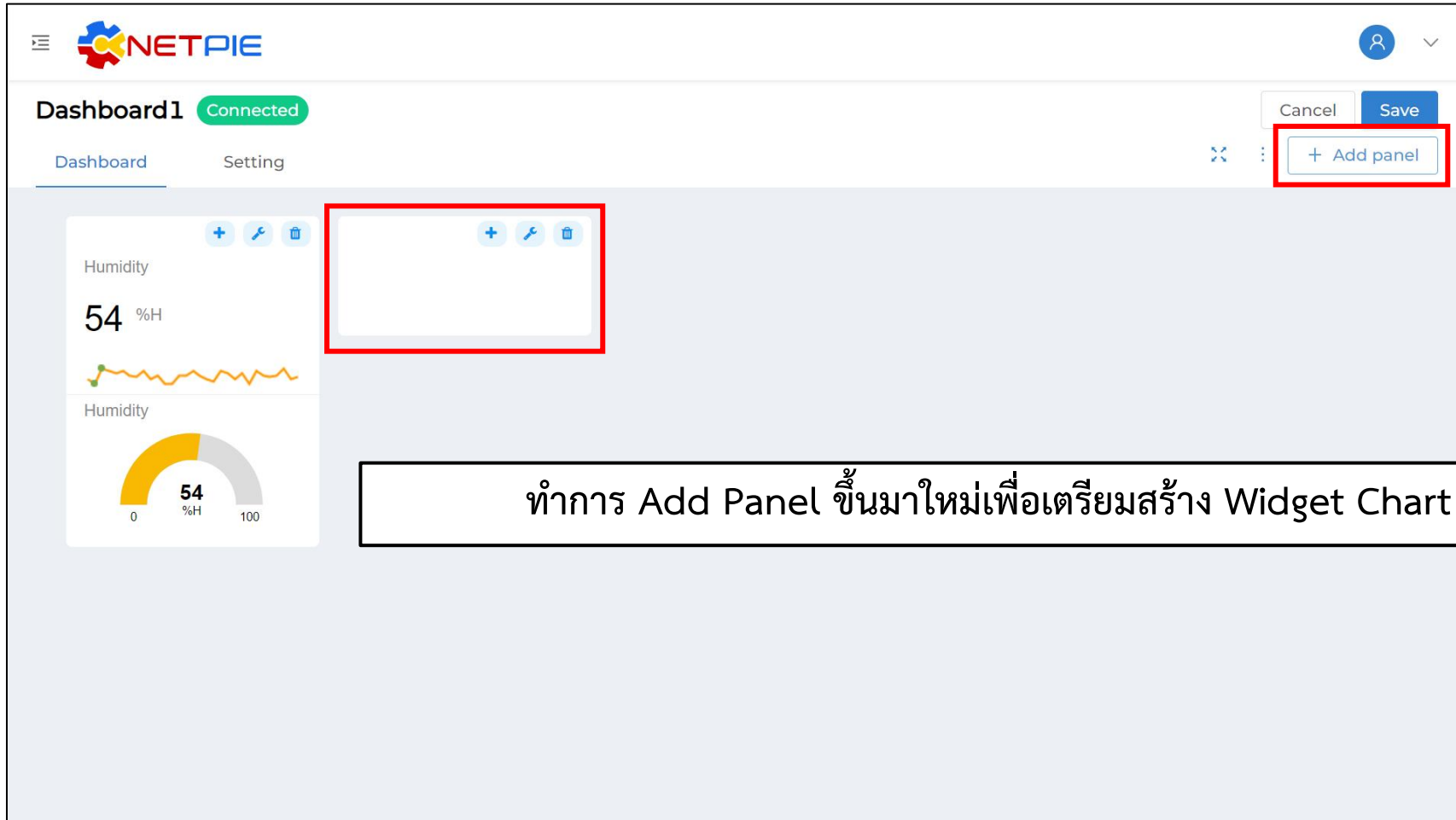


The screenshot shows the NETPIE dashboard interface. At the top left is the NETPIE logo. Below it, the dashboard is titled "Dashboard1" with a "Connected" status indicator. There are tabs for "Dashboard" and "Setting". In the top right corner, there are "Cancel" and "Save" buttons, and a "+ Add panel" button. The main area contains two widgets for "Humidity". The top widget shows a line graph with a value of 56 %H. The bottom widget, which is highlighted with a red box, is a Gauge chart showing a value of 56 %H on a scale from 0 to 100.

เมื่อกด Save จะได้ Widget Gauge ที่แสดงข้อมูล Humidity ของ Device2

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล



The screenshot shows the NETPIE dashboard interface. At the top left is the NETPIE logo. Below it, the dashboard is titled "Dashboard1" with a "Connected" status. There are two tabs: "Dashboard" and "Setting". In the top right corner, there are "Cancel" and "Save" buttons, and a "+ Add panel" button which is highlighted with a red box. The main area contains several widgets. On the left, there are two "Humidity" widgets: one showing a line graph with a value of 54 %H, and another showing a gauge with a value of 54 %H. To the right of these, there is an empty white box with a red border, also containing a "+ Add panel" button icon. A text box at the bottom of the screenshot contains the instruction: "ทำการ Add Panel ขึ้นมาใหม่เพื่อเตรียมสร้าง Widget Chart".

ทำการ Add Panel ขึ้นมาใหม่เพื่อเตรียมสร้าง Widget Chart

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Chart

The screenshot displays the NETPIE dashboard interface. On the left, a 'Humidity' widget shows a value of 52 %H with a gauge and a small line chart. The main area shows the configuration for a 'Chart' widget. A text box highlights: 'Chart เป็น Widget ที่นำข้อมูลมาแสดงผลแบบกราฟ Time-series เพื่อการดูข้อมูลเทียบกับเวลา'. The configuration panel includes fields for TITLE ('All Data'), DATA SOURCE ('#[\"Device2\"][\"feed\"]'), FILTER ('humidity x temperature x light x'), QUERY FROM THE LAST ('1'), PLOT MODE ('Normal'), and TYPE OF CHART ('Line'). A preview window on the right shows a time-series chart with three data series: humidity (blue), light (red), and temperature (green) over a time range from 15:40:00 to 16:30:00. The y-axis ranges from 30.0 to 180.0.

Chart เป็น Widget ที่นำข้อมูลมาแสดงผลแบบกราฟ Time-series เพื่อการดูข้อมูลเทียบกับเวลา

TITLE: All Data
DATA SOURCE: #[\"Device2\"][\"feed\"]
FILTER: humidity x temperature x light x
QUERY FROM THE LAST: 1
PLOT MODE: Normal
TYPE OF CHART: Line

All Data

180.0
150.0
120.0
90.0
60.0
30.0

15:40:00 15:50:00 16:00:00 16:10:00 16:20:00 16:30:00

● humidity ● light ● temperature

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Chart

NETPIE

Dashboard1 Connected

Dashboard Setting

Humidity 52 %H

Humidity 52 %H

TYPE: Chart

TITLE: All Data

DATA SOURCE: #["Device2"]["feed"]

FILTER: humidity × temperature × light ×

QUERY FROM THE LAST: 1

Hour

STRICT TIME RANGE: NO

PLOT MODE: Normal

TYPE OF CHART: Line

X AXIS TITLE

Y AXIS TITLE

BEGIN AT 0: NO

Cancel Save

+ Add panel

ข้อความกำกับข้อมูล

ข้อมูลที่ดึงมาจาก Feed

Field ของข้อมูล ที่ต้องการดึงมาแสดงกราฟ

ช่วงเวลาในการดึงข้อมูล

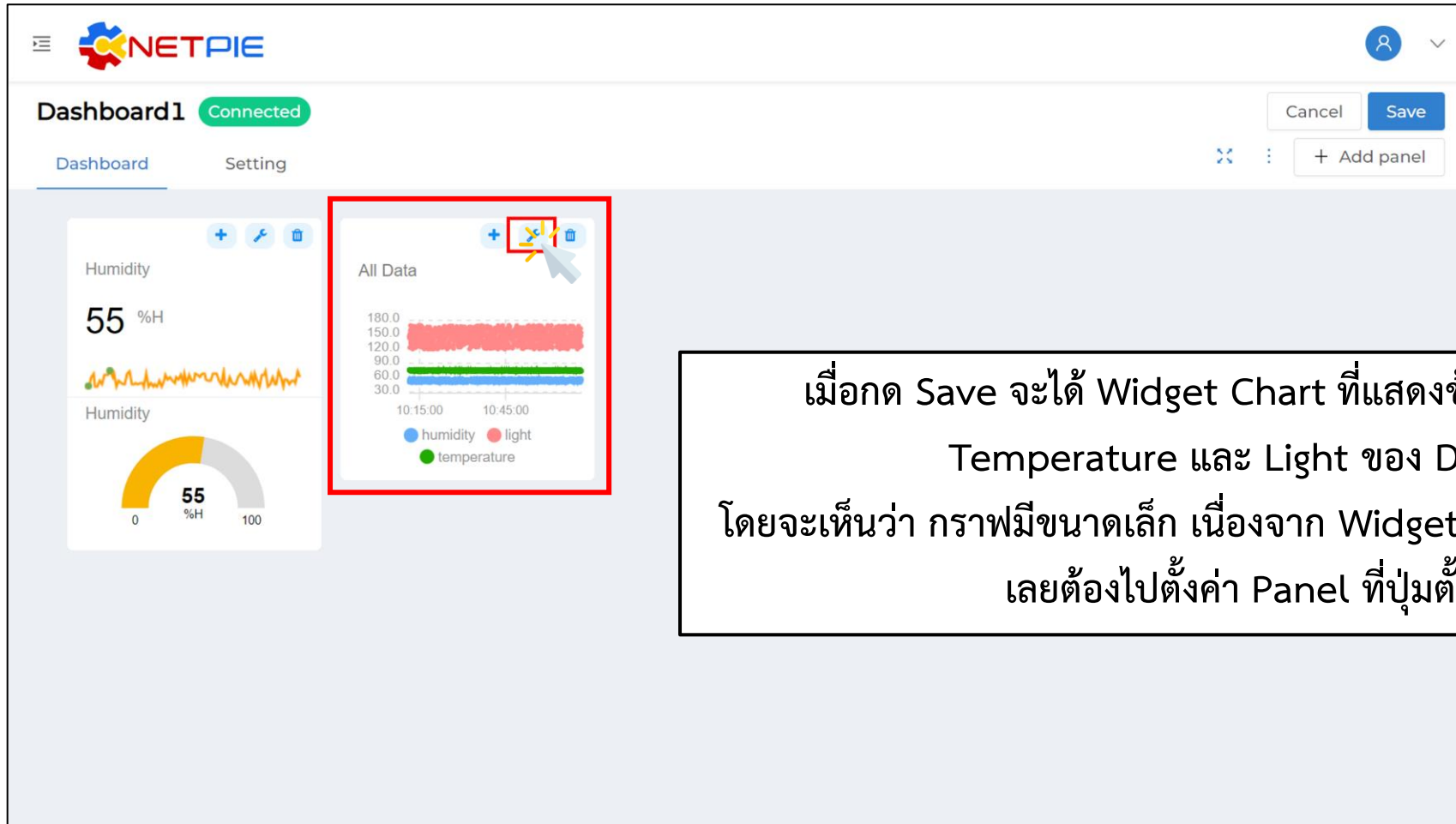
รูปแบบการ Plot ข้อมูล

ประเภทของกราฟ

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Chart

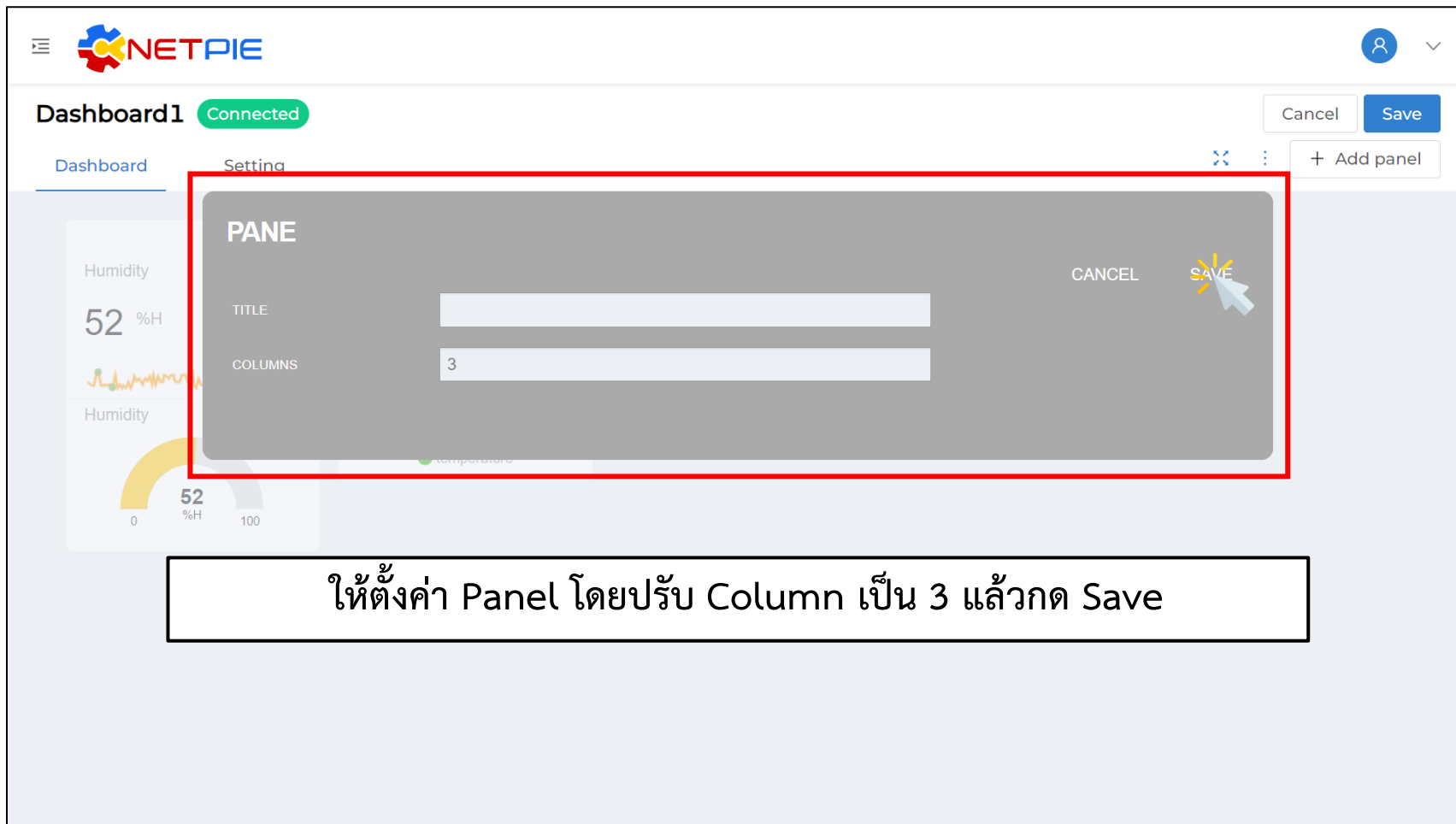


เมื่อกด Save จะได้ Widget Chart ที่แสดงข้อมูล Humidity, Temperature และ Light ของ Device2 โดยจะเห็นว่า กราฟมีขนาดเล็ก เนื่องจาก Widget ถูก Limit ตาม Panel เลยต้องไปตั้งค่า Panel ที่ปุ่มตั้งค่า

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

การสร้าง Widget : Chart



NETPIE

Dashboard1 Connected

Dashboard Setting

PANE

TITLE

COLUMNS 3

CANCEL SAVE

Humidity 52 %H

Humidity 52 %H

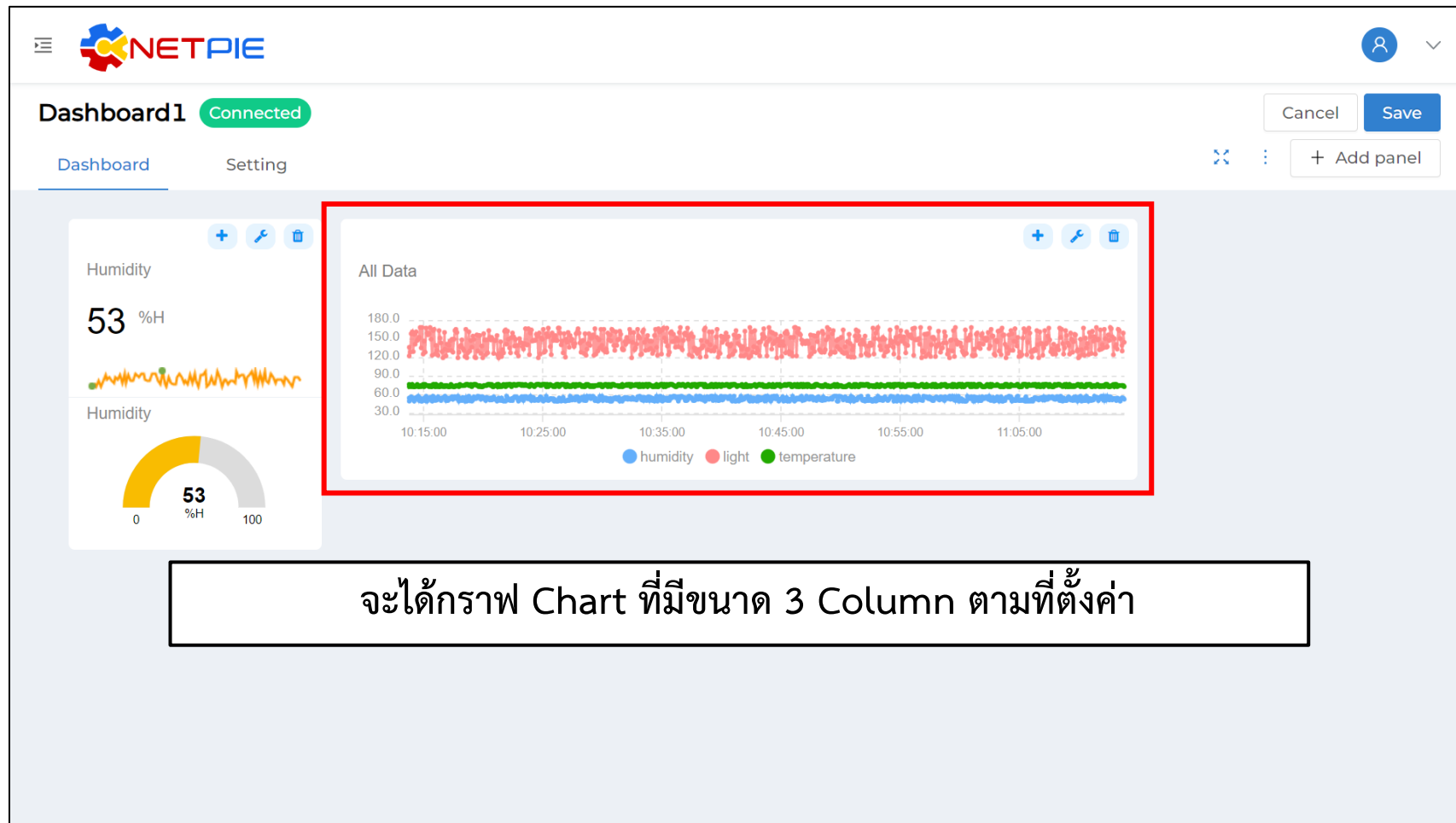
0 100

ให้ตั้งค่า Panel โดยปรับ Column เป็น 3 แล้วกด Save

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Widget ประเภทแสดงผล

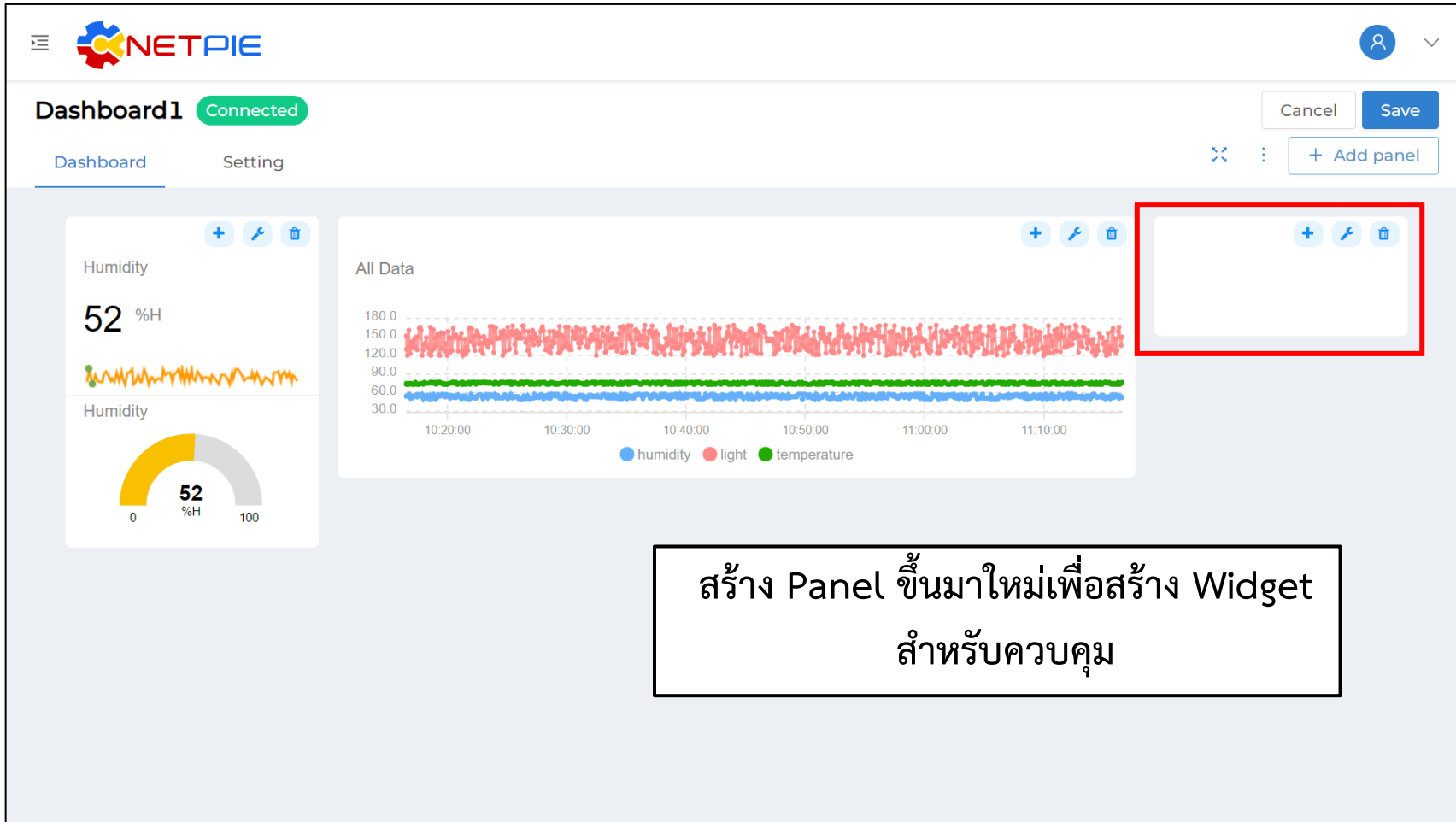
การสร้าง Widget : Chart



จะได้กราฟ Chart ที่มีขนาด 3 Column ตามที่ตั้งค่า

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม



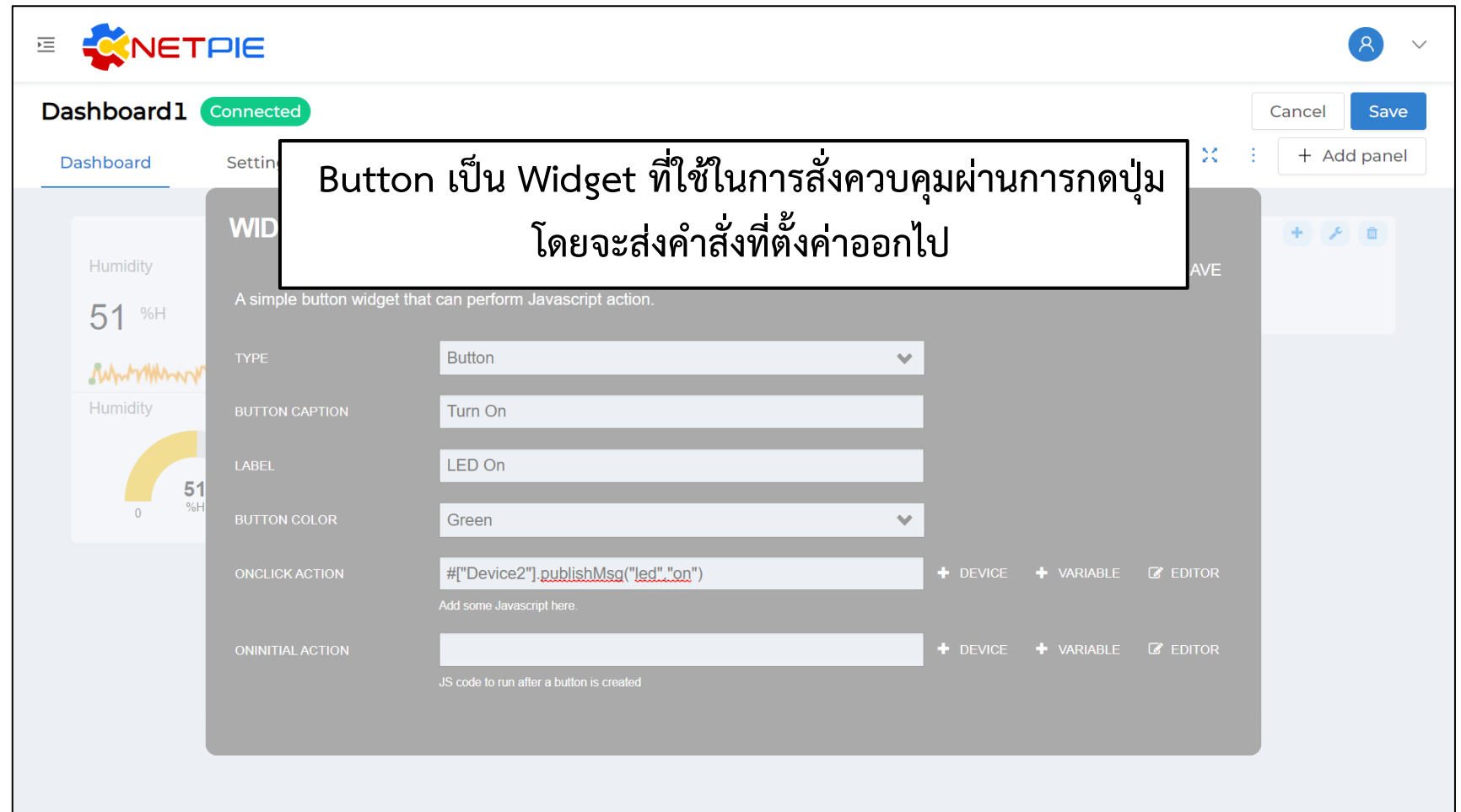
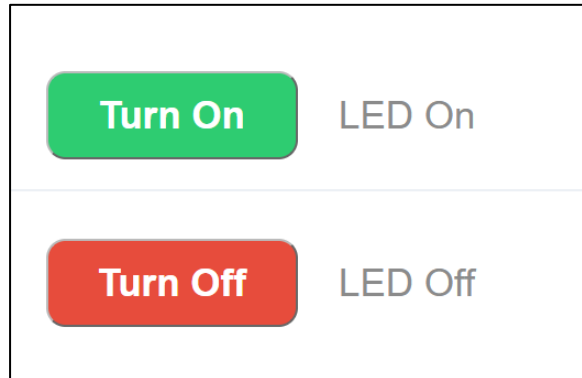
The screenshot displays the NETPIE dashboard interface. At the top, the logo 'NETPIE' is visible. Below it, the dashboard is titled 'Dashboard1' with a 'Connected' status. There are tabs for 'Dashboard' and 'Setting'. A 'Cancel' button and a 'Save' button are located in the top right corner. A '+ Add panel' button is also present. The main area contains two widgets: a 'Humidity' widget on the left showing a value of 52 %H and a gauge, and an 'All Data' chart on the right showing three data series: humidity (blue), light (red), and temperature (green). A red box highlights an empty panel area with a '+ Add panel' button, indicating where to create a new widget.

สร้าง Panel ขึ้นมาใหม่เพื่อสร้าง Widget สำหรับควบคุม

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Button



Button เป็น Widget ที่ใช้ในการสั่งควบคุมผ่านการกดปุ่ม โดยจะส่งคำสั่งที่ตั้งค่าออกไป

A simple button widget that can perform Javascript action.

TYPE	Button
BUTTON CAPTION	Turn On
LABEL	LED On
BUTTON COLOR	Green
ONCLICK ACTION	#["Device2"].publishMsg("led","on")
ONINITIAL ACTION	

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Button

The screenshot shows the NETPIE dashboard interface. On the left, there is a 'Humidity' widget displaying '51 %H'. The main area is the 'WIDGET' configuration panel for a 'Button' widget. The configuration fields are as follows:

- TYPE:** Button
- BUTTON CAPTION:** Turn On
- LABEL:** LED On
- BUTTON COLOR:** Green
- ONCLICK ACTION:** #["Device2"].publishMsg("led"."on")
- ONINITIAL ACTION:** (Empty)

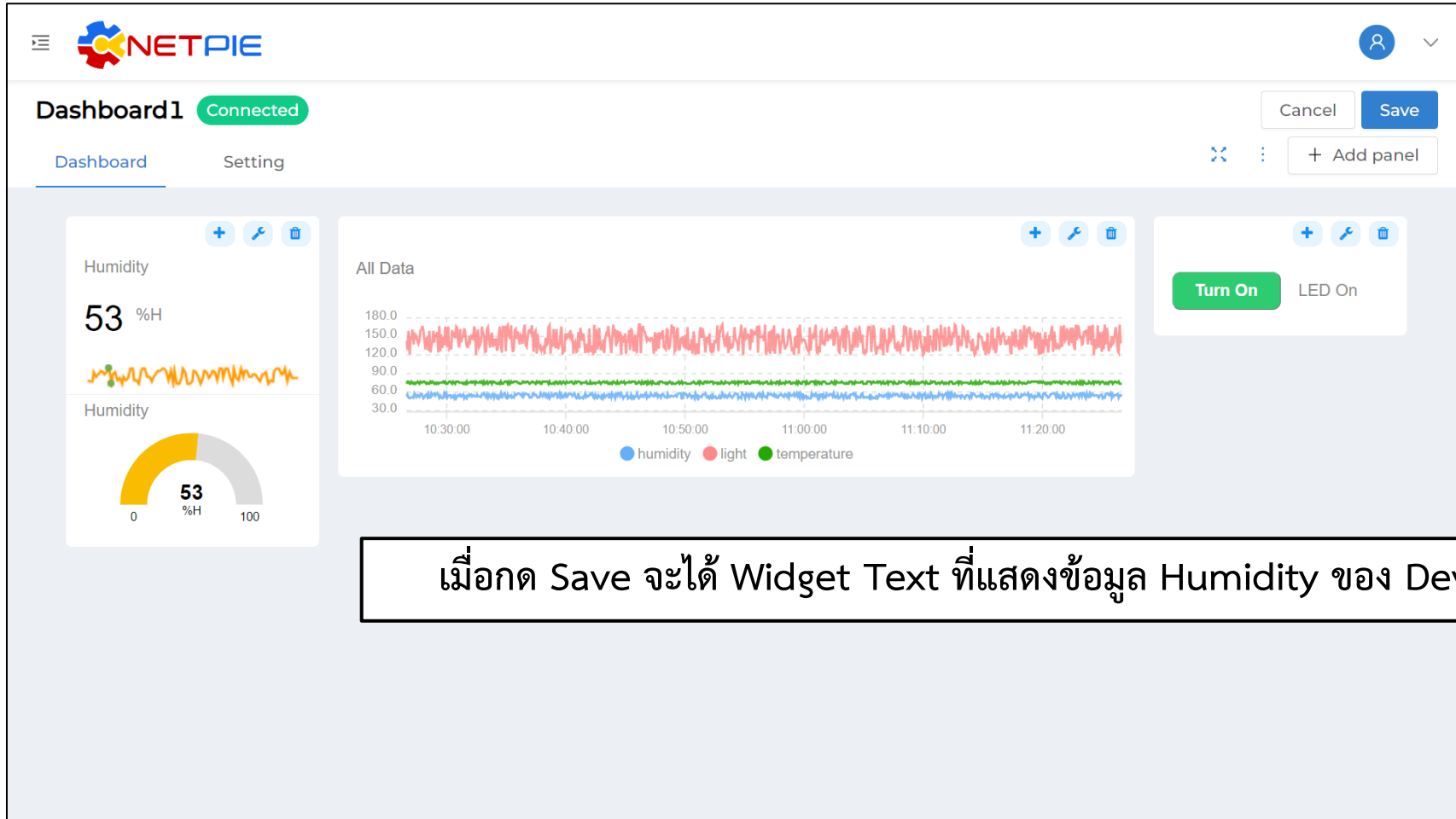
Red boxes highlight the following fields, with arrows pointing to Thai labels:

- TYPE:** ข้อความภายในปุ่ม
- BUTTON CAPTION:** ข้อความกำกับข้างปุ่ม
- LABEL:** สีของปุ่ม
- ONCLICK ACTION:** คำสั่งที่เกิดเมื่อกดปุ่ม Button

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Button



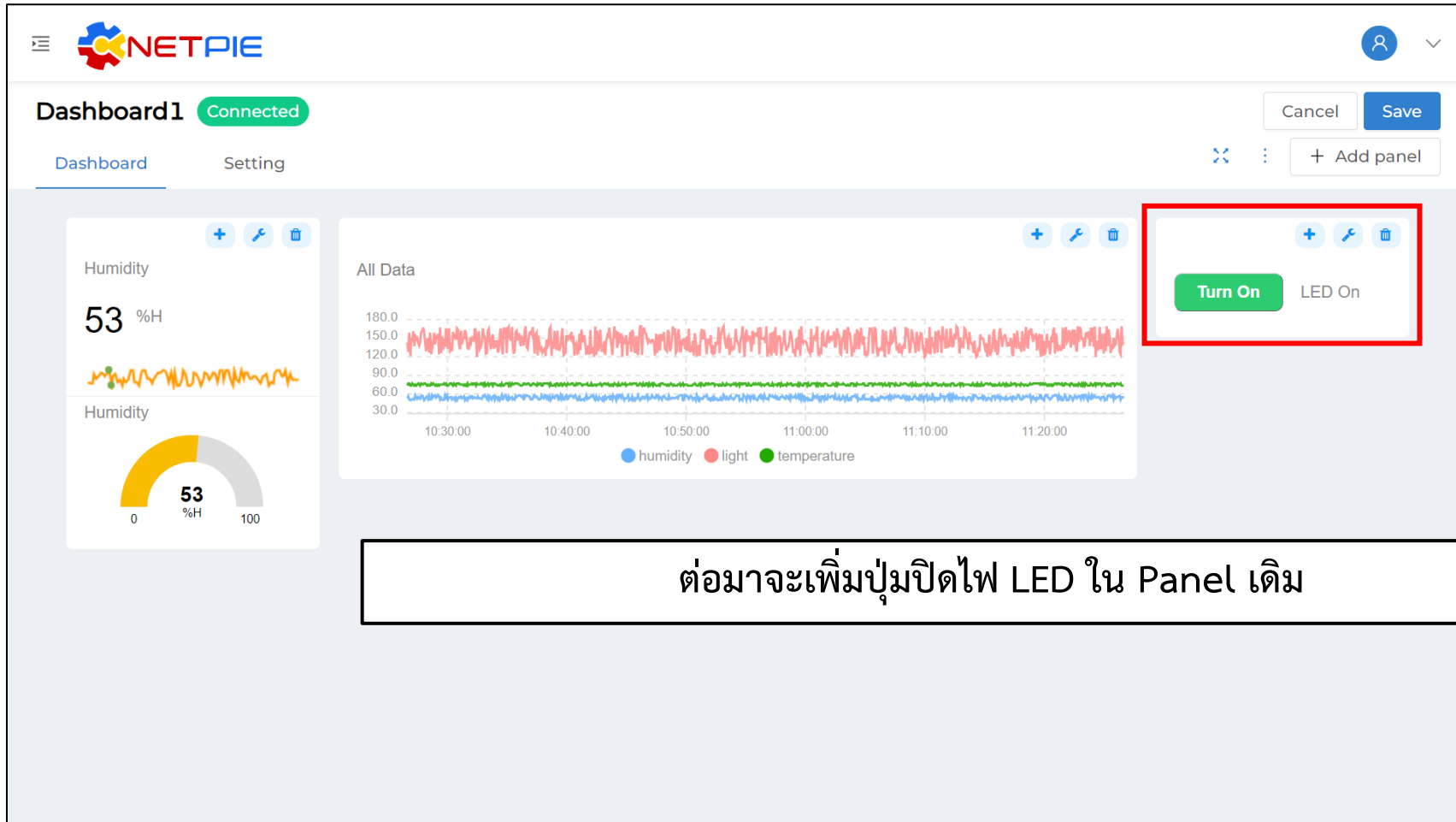
The screenshot shows the NETPIE dashboard interface. At the top left is the NETPIE logo. The dashboard title is "Dashboard1" with a "Connected" status indicator. There are tabs for "Dashboard" and "Setting". In the top right corner, there are "Cancel" and "Save" buttons, and a "+ Add panel" button. The main area contains three widgets: 1. A "Humidity" widget showing a value of 53 %H with a small line graph and a gauge below it. 2. An "All Data" chart showing three data series: humidity (blue), light (red), and temperature (green) over time from 10:30:00 to 11:20:00. 3. A "Turn On" button labeled "LED On".

เมื่อกด Save จะได้ Widget Text ที่แสดงข้อมูล Humidity ของ Device2

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Button



The screenshot shows the NETPIE dashboard interface. At the top left is the NETPIE logo. Below it, the dashboard is titled 'Dashboard1' with a 'Connected' status indicator. There are tabs for 'Dashboard' and 'Setting'. On the right side, there are 'Cancel' and 'Save' buttons, and a '+ Add panel' button. The main area contains several widgets: a 'Humidity' widget showing 53 %H with a line graph and a gauge; an 'All Data' widget showing a line graph for humidity, light, and temperature; and a 'Turn On' button widget for 'LED On', which is highlighted with a red box. The 'Turn On' button is green and has a white text label 'LED On' next to it.

ต่อมาจะเพิ่มปุ่มปิดไฟ LED ใน Panel เดิม

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Button

The screenshot shows the NETPIE dashboard interface. On the left, there is a 'Humidity' widget displaying '53 %H'. The main area is the 'WIDGET' configuration panel for a 'Button' widget. The configuration fields are as follows:

- TYPE:** Button
- BUTTON CAPTION:** Turn Off
- LABEL:** LED Off
- BUTTON COLOR:** Red
- ONCLICK ACTION:** #["Device2"].publishMsg("led"."off")
- ONINITIAL ACTION:** (Empty)

Red boxes highlight the 'BUTTON CAPTION', 'LABEL', 'BUTTON COLOR', and 'ONCLICK ACTION' fields. Arrows point from these boxes to Thai labels on the right:

- ข้อความภายในปุ่ม (Button text)
- ข้อความกำกับข้างปุ่ม (Text next to button)
- สีของปุ่ม (Button color)
- คำสั่งที่เกิดเมื่อกดปุ่ม Button (Command when button is pressed)

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Button

NETPIE

Dashboard 1 Connected

Dashboard Setting

Cancel Save

+ Add panel

Humidity

57 %H

Humidity

57 %H

All Data

180.0
150.0
120.0
90.0
60.0
30.0

10:35:00 10:45:00 10:55:00 11:05:00 11:15:00 11:25:00

● humidity ● light ● temperature

Turn On LED On

Turn Off LED Off

ได้ Button ที่ใช้สำหรับการเปิดและปิด LED

ใบงานที่ 5.3 ขั้นตอนการทดลอง

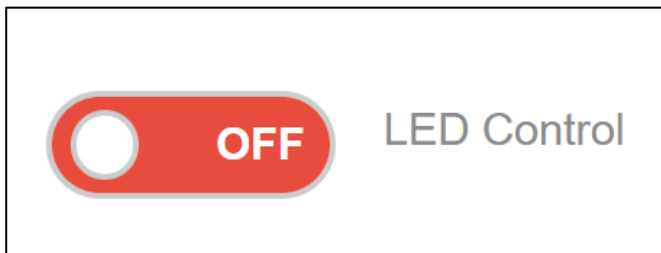
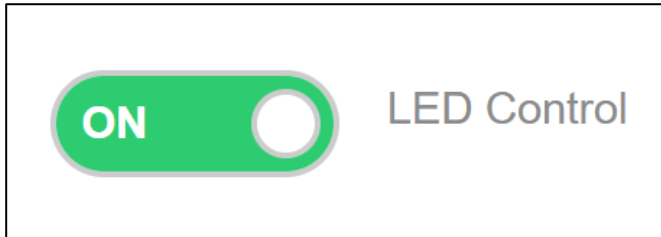
การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

The screenshot shows the NETPIE dashboard interface. At the top, there is a navigation bar with the NETPIE logo, a user profile icon, and a 'Save' button. Below the navigation bar, the dashboard is titled 'Dashboard 1' and shows a 'Connected' status. There are two tabs: 'Dashboard' and 'Setting'. The main area contains several widgets: a 'Humidity' widget showing 57 %H, a 'All Data' line chart showing humidity, light, and temperature over time, and a control widget for an LED. The control widget has two buttons: 'Turn On' (green) and 'Turn Off' (red). A red box highlights the control widget, and a text box below it says 'เพิ่ม Widget ถัดไป ใน Panel เดิม'.

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Toggle



Toggle เป็น Widget ที่ใช้ในการสั่งควบคุมโดยแตกต่างจาก Button ตรงที่เป็น ปุ่มแบบเปิดปิด ภายในปุ่มเดียว

Setting	Value	Options
TYPE	Toggle	
TOGGLE ON CAPTION	ON	
TOGGLE OFF CAPTION	OFF	
LABEL	LED Control	
TOGGLE STATE	<code>#["Device2"]["shadow"]["led"] == "on"</code>	+ DEVICE + VARIABLE EDITOR
ONTOGGLEON ACTION	<code>#["Device2"].publishMsg("led", "on")</code>	+ DEVICE + VARIABLE EDITOR
ONTOGGLEOFF ACTION	<code>#["Device2"].publishMsg("led", "off")</code>	+ DEVICE + VARIABLE EDITOR
ONINITIAL ACTION		+ DEVICE + VARIABLE EDITOR

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Toggle

The screenshot shows the NETPIE dashboard with a 'Dashboard1' widget selected. The 'WIDGET' configuration panel is open, showing the following fields:

- TYPE:** Toggle
- TOGGLE ON CAPTION:** ON
- TOGGLE OFF CAPTION:** OFF
- LABEL:** LED Control
- TOGGLE STATE:** `#["Device2"]["shadow"]["led"] == "on"`
- ONTOGGLEON ACTION:** `#["Device2"].publishMsg("led", "on")`
- ONTOGGLEOFF ACTION:** `#["Device2"].publishMsg("led", "off")`
- ONINITIAL ACTION:** (Empty)

Red boxes highlight the following fields, with arrows pointing to labels:

- TOGGLE ON CAPTION:** ข้อความภายในปุ่มตอนเปิด
- TOGGLE OFF CAPTION:** ข้อความภายในปุ่มตอนปิด
- LABEL:** ข้อความกำกับข้างปุ่ม
- TOGGLE STATE:** สถานะของ Toggle ตอนเปิด
- ONTOGGLEON ACTION:** คำสั่งที่ส่งออกเมื่อเปิด
- ONTOGGLEOFF ACTION:** คำสั่งที่ส่งออกเมื่อปิด

ใบงานที่ 5.3 ขั้นตอนการทดลอง

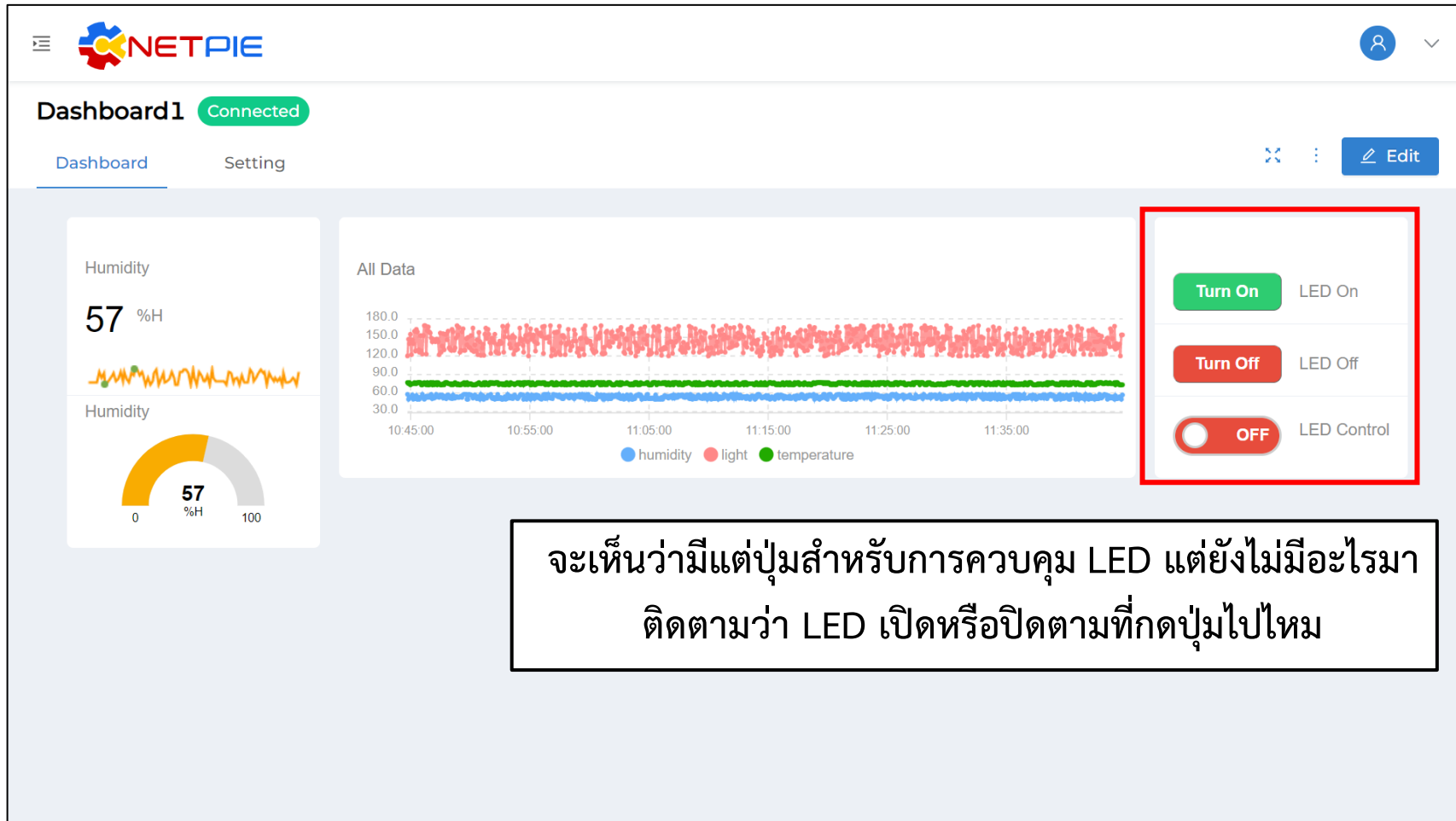
การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Toggle

The screenshot displays the NETPIE dashboard interface. At the top left, the NETPIE logo is visible. The dashboard title is "Dashboard1" with a "Connected" status indicator. Below the title, there are tabs for "Dashboard" and "Setting". The main content area is divided into three sections: a humidity widget on the left showing a value of 57 %H and a gauge; a central "All Data" graph showing three data series (humidity, light, and temperature) over time; and a control panel on the right. The control panel includes three buttons: "Turn On" (LED On), "Turn Off" (LED Off), and a toggle switch labeled "OFF" (LED Control). The toggle switch is highlighted with a red box. A text box at the bottom of the screenshot states: "ได้ Toggle ที่ใช้สำหรับการเปิดและปิด LED อีกปุ่มหนึ่ง".

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม



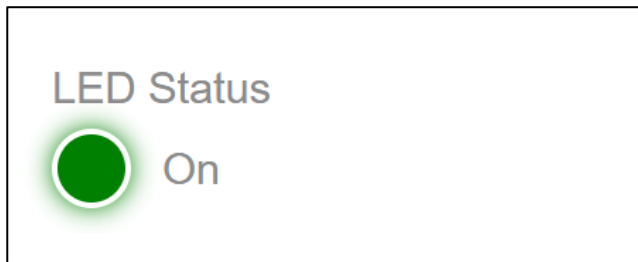
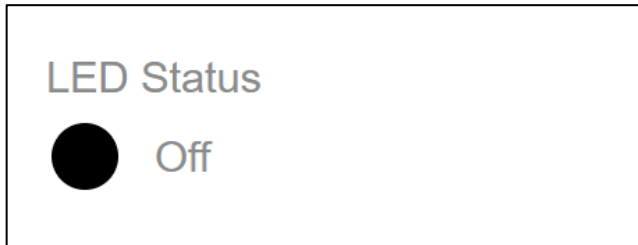
The screenshot displays the NETPIE dashboard interface. On the left, there is a 'Humidity' widget showing a value of 57 %H with a corresponding gauge. In the center, an 'All Data' chart shows three data series: humidity (blue), light (red), and temperature (green) over time. On the right, a control widget is highlighted with a red box, containing three buttons: 'Turn On' (green), 'Turn Off' (red), and 'LED Control' (red with a toggle switch). Below the control widget, a text box contains the following text:

จะเห็นว่า มีแต่ปุ่มสำหรับการควบคุม LED แต่ยังไม่มียังอะไรมาติดตามว่า LED เปิดหรือปิดตามที่กดปุ่มไปไหม

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Indicator Light



Indicator เป็น Widget ที่เป็นไฟสถานะที่สามารถใช้ได้ ในหลายสถานการณ์ เช่นการแสดงสถานะของ LED

FIELD	VALUE	ACTION
TYPE	Indicator Light	
TITLE	LED Status	
LIGHT COLOR	Green	+ DEVICE + VARIABLE EDITOR
VALUE	<code>#[["Device2"]]["shadow"]["led"] == "on"</code>	+ DEVICE + VARIABLE EDITOR
ON TEXT	LED On	+ DEVICE + VARIABLE EDITOR
OFF TEXT	LED Off	+ DEVICE + VARIABLE EDITOR

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

การสร้าง Widget : Indicator Light

The screenshot shows the NETPIE dashboard interface. The main content area is titled "WIDGET" and displays the configuration for an "Indicator Light" widget. The configuration fields are as follows:


- TYPE:** Indicator Light
- TITLE:** LED Status
- LIGHT COLOR:** Green
- VALUE:** `#["Device2"]["shadow"]["led"] == "on"`
- ON TEXT:** LED On
- OFF TEXT:** LED Off

Red boxes highlight the TITLE, LIGHT COLOR, VALUE, ON TEXT, and OFF TEXT fields. Red arrows point from these boxes to labels on the right side of the image:

- ข้อความกำกับบนปุ่ม (Label for the widget title)
- สีของไฟตอนเปิด (Label for the light color)
- สถานะของไฟตอนเปิด (Label for the value expression)
- ข้อความกำกับเมื่อเปิด (Label for the on text)
- ข้อความกำกับเมื่อปิด (Label for the off text)

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม



The screenshot displays the NETPIE dashboard interface. At the top, it shows 'Dashboard1' with a 'Connected' status. Below this, there are tabs for 'Dashboard' and 'Setting'. The main area contains three widgets: a 'Humidity' widget showing 52 %H with a gauge, an 'All Data' line chart showing humidity (blue), light (red), and temperature (green) over time, and a control widget. The control widget has three buttons: 'Turn On' (LED On), 'Turn Off' (LED Off), and 'LED Control' (OFF). Below these buttons is an 'LED Status' indicator showing a black circle next to 'LED Off'. A red box highlights the 'LED Status' section.

ได้ Indicator Light ที่แสดงสถานะของ LED มาเรียบร้อยแล้ว

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม



The screenshot displays the NETPIE dashboard interface. At the top left, the NETPIE logo is visible. The dashboard title is "Dashboard1" with a "Connected" status indicator. There are tabs for "Dashboard" and "Setting". On the right side, there are "Cancel" and "Save" buttons, and a "+ Add panel" button. The main area contains several widgets: a "Humidity" widget showing 54 %H with a line graph and a gauge; an "All Data" widget showing a line graph with three series: humidity (blue), light (red), and temperature (green); and a control panel on the right with buttons for "Turn On LED On", "Turn Off LED Off", and a toggle switch for "LED Control" (currently OFF). Below the control panel, there is an "LED Status" section with a "LED Off" indicator and a small icon of a person with a gear, which is highlighted with a red box and a mouse cursor. A text box at the bottom of the dashboard area contains the following text:

จัดเรียง Widget ที่อยู่ภายใน Panel เดียวกันได้
ด้วยการกดปุ่ม ขึ้นหรือลงบน Widget

ใบงานที่ 5.3 ขั้นตอนการทดลอง

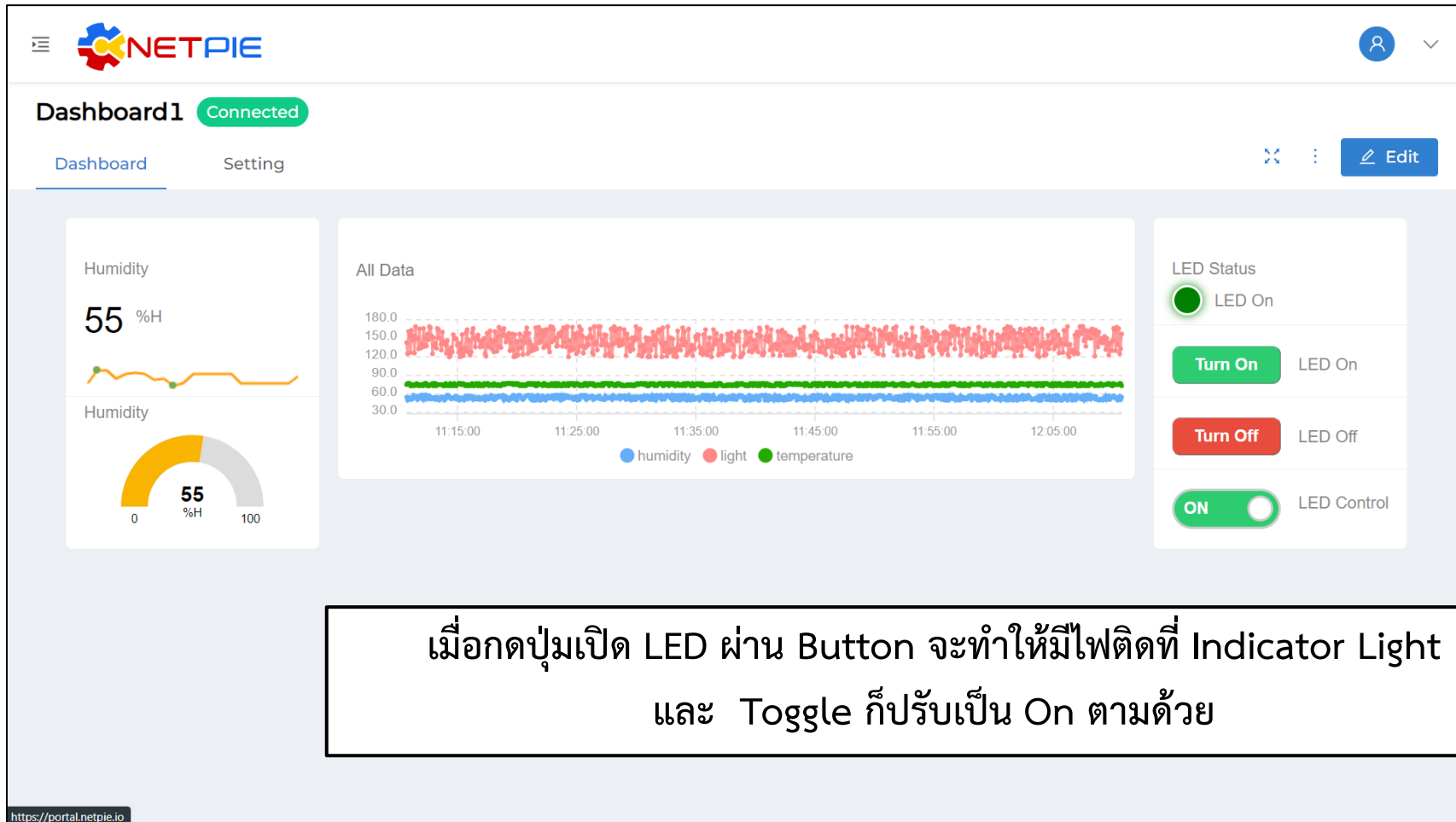
การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม

The screenshot shows the NETPIE dashboard interface. At the top, it says "Dashboard1" and "Connected". There are tabs for "Dashboard" and "Setting". The main area contains three widgets: 1. "Humidity" widget showing a value of 51 %H with a small line graph and a gauge below it. 2. "All Data" widget showing a line graph with three series: humidity (blue), light (red), and temperature (green) over time. 3. "LED Status" widget with a "Turn On" button (LED On), a "Turn Off" button (LED Off), and a toggle switch for "LED Control" (currently OFF).

ได้ Dashboard ที่แสดงข้อมูลและสามารถควบคุม LED
พร้อมกับดูสถานะ LED ได้เรียบร้อย

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 3 การสร้าง Widget ประเภทควบคุม



The screenshot displays the NETPIE dashboard interface. At the top left is the NETPIE logo. The main header shows 'Dashboard1' with a 'Connected' status indicator. Below the header are tabs for 'Dashboard' and 'Setting', and an 'Edit' button. The dashboard contains three main widgets:

- Humidity Widget:** Shows a current value of 55 %H, a small line graph, and a semi-circular gauge also showing 55 %H.
- All Data Chart:** A line graph showing three data series: humidity (blue), light (red), and temperature (green) over time from 11:15:00 to 12:05:00. The y-axis ranges from 30.0 to 180.0.
- LED Status Control Panel:** Features a green indicator light labeled 'LED On', a green 'Turn On' button, a red 'Turn Off' button, and a green toggle switch labeled 'ON' and 'LED Control'.

At the bottom left of the dashboard, the URL <https://portal.netpie.io> is visible.

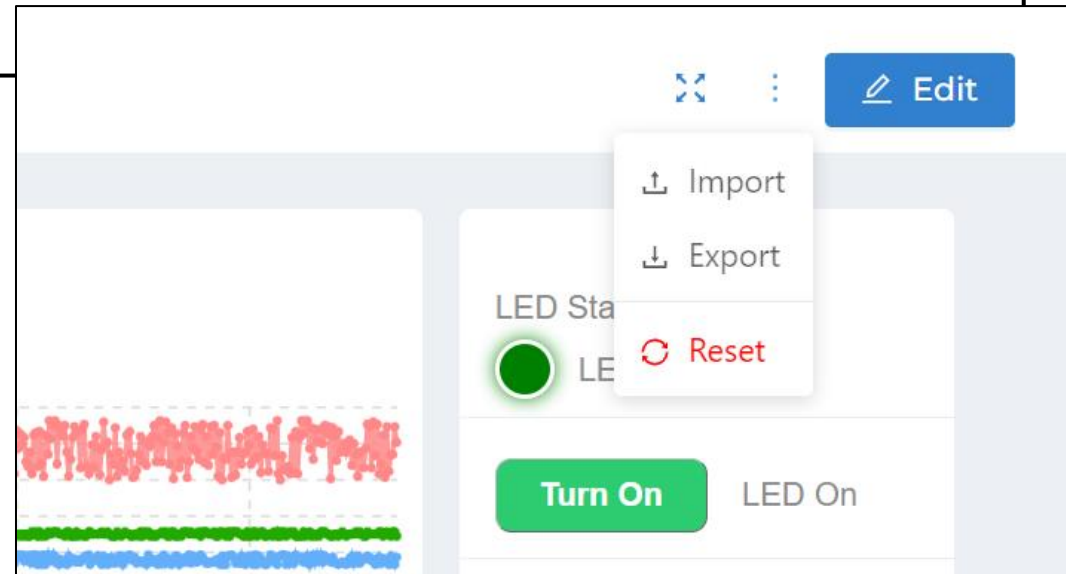
เมื่อกดปุ่มเปิด LED ผ่าน Button จะทำให้มีไฟติดที่ Indicator Light และ Toggle ก็ปรับเป็น On ตามด้วย

ใบงานที่ 5.3 ทฤษฎีเบื้องต้น

การจัดการ Dashboard

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

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



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

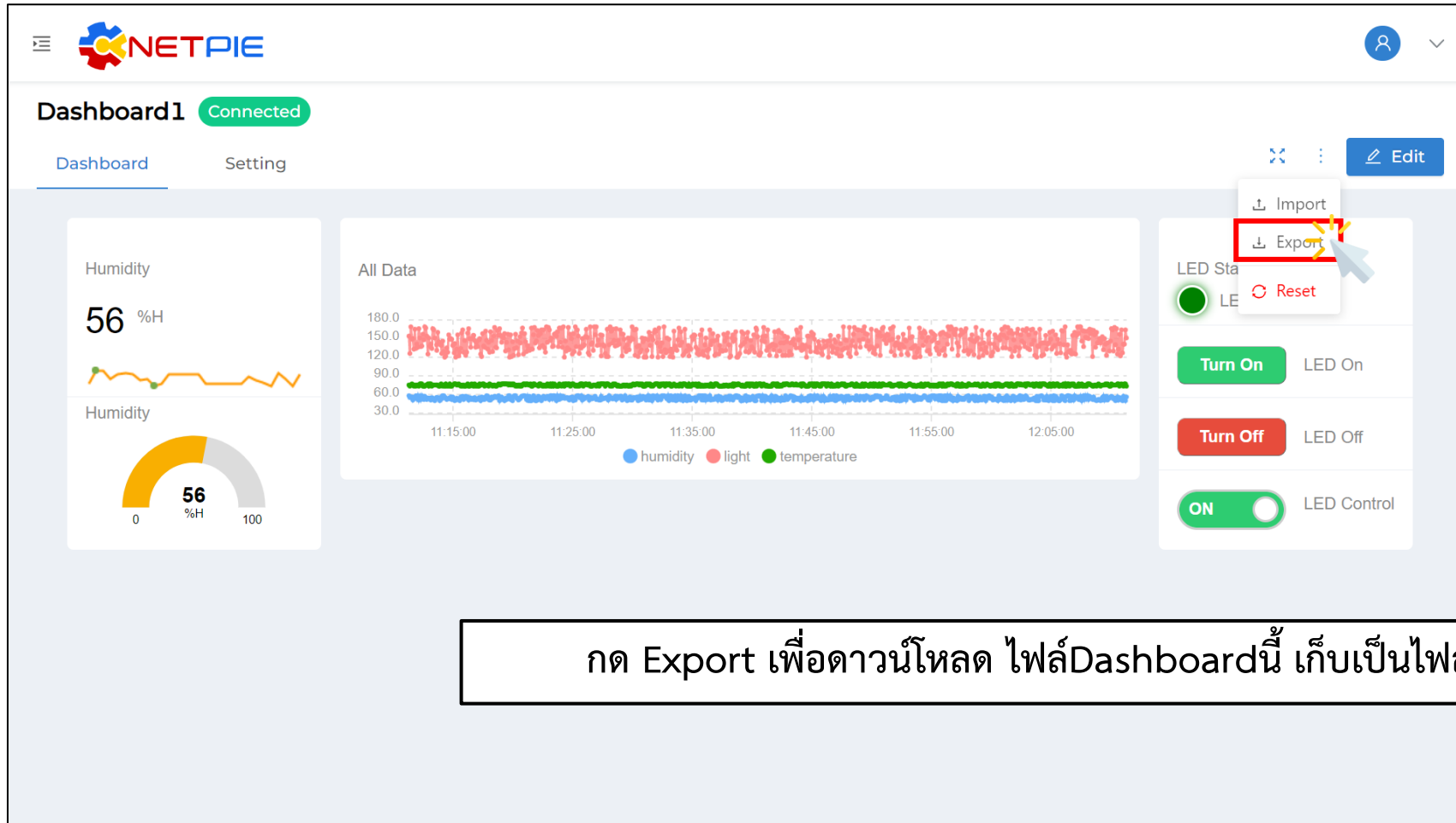
The screenshot displays the NETPIE dashboard for 'Dashboard1' which is 'Connected'. It features a navigation bar with 'Dashboard' and 'Setting' tabs. The main content area includes a 'Humidity' widget showing 56 %H with a gauge and line chart, an 'All Data' widget with a line graph for humidity (blue), light (red), and temperature (green) from 11:15:00 to 12:05:00, and an 'LED Status' control panel. The control panel has a 'Reset' button (circled in red in the original image), 'Turn On' and 'Turn Off' buttons, and an 'LED Control' toggle switch. A context menu is open over the 'Reset' button, showing 'Import' and 'Export' options. A red box highlights the 'Reset' button and the context menu.

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

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

Export



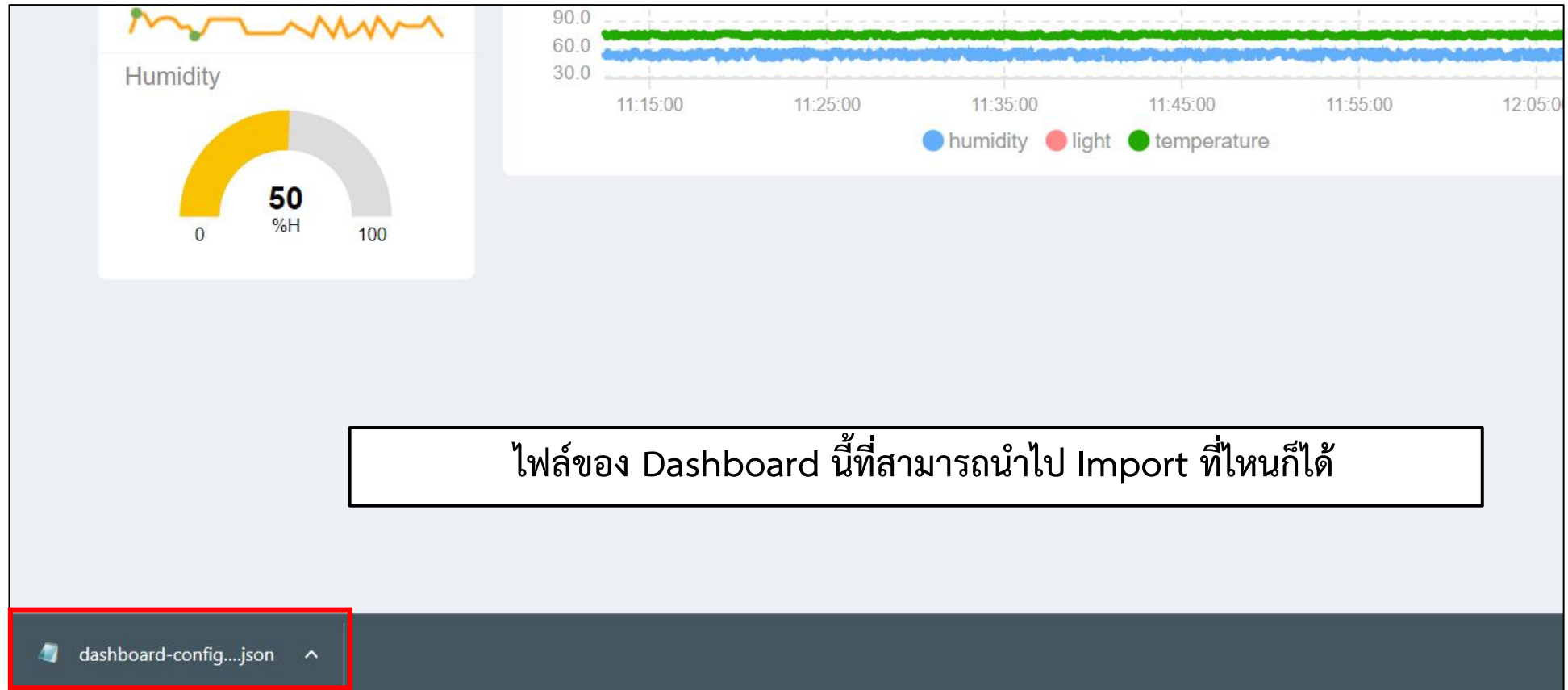
The screenshot shows the NETPIE dashboard for 'Dashboard1' which is 'Connected'. It features a 'Humidity' widget showing 56 %H, an 'All Data' line chart for humidity, light, and temperature, and an 'LED Control' panel with 'Turn On', 'Turn Off', and 'LED Control' buttons. A context menu is open over the LED Control panel, with the 'Export' option highlighted by a red box and a yellow arrow. The 'Import' option is also visible above it.

กด Export เพื่อดาวน์โหลดไฟล์ Dashboard นี้ เก็บเป็นไฟล์ json

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

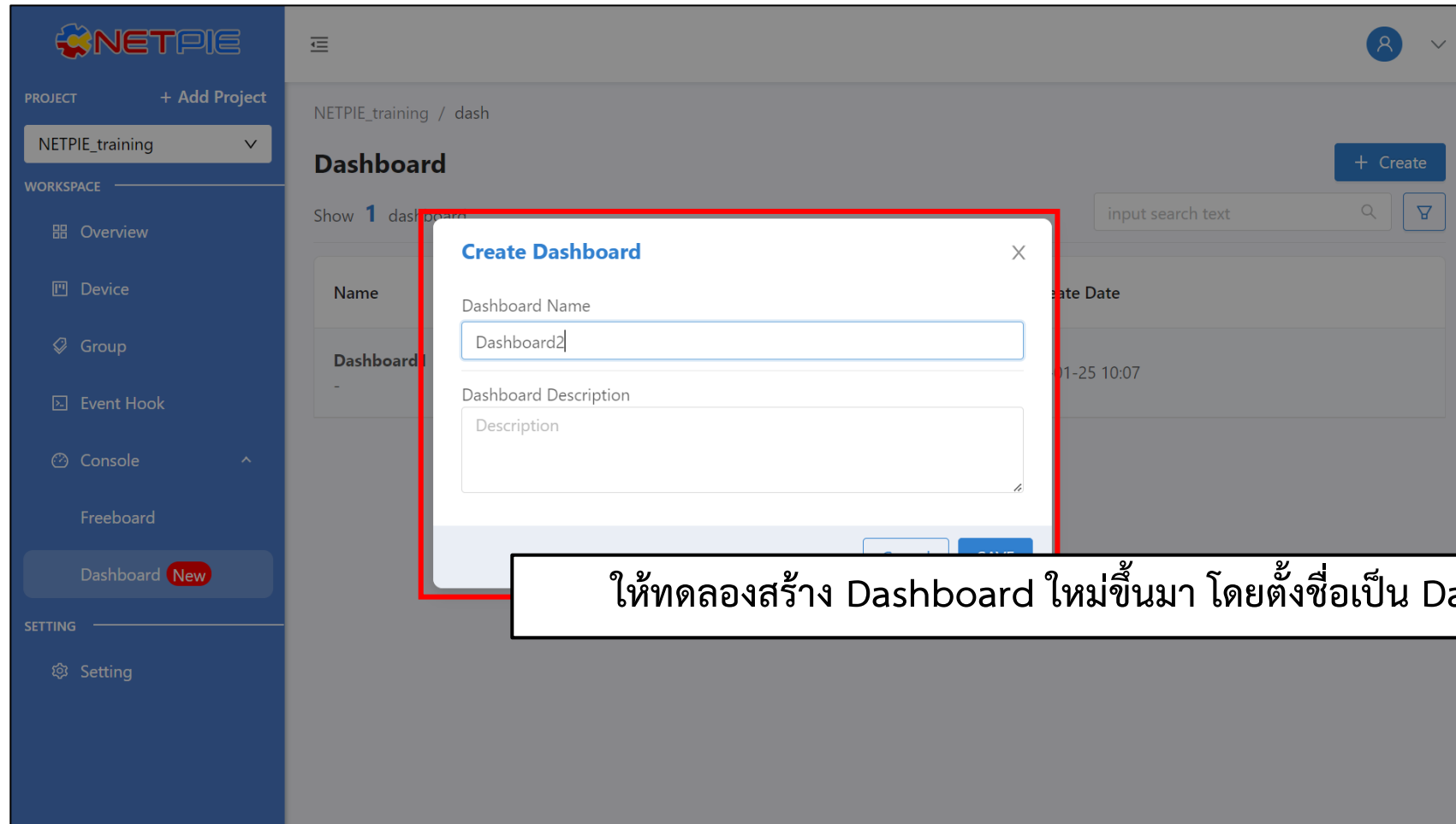
Export



ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

Import



The screenshot displays the NETPIE web interface. On the left is a dark blue sidebar with navigation options: PROJECT (+ Add Project), WORKSPACE (Overview, Device, Group, Event Hook, Console, Freeboard, Dashboard **New**), and SETTING (Setting). The main content area shows the 'Dashboard' management page for 'NETPIE_training / dash'. A modal window titled 'Create Dashboard' is open, containing a 'Dashboard Name' input field with the text 'Dashboard2' and a 'Dashboard Description' text area with the placeholder 'Description'. A red rectangular box highlights the modal form. Below the modal, a white text box contains the instruction: 'ให้ทดลองสร้าง Dashboard ใหม่ขึ้นมา โดยตั้งชื่อเป็น Dashboard2'.

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

Import

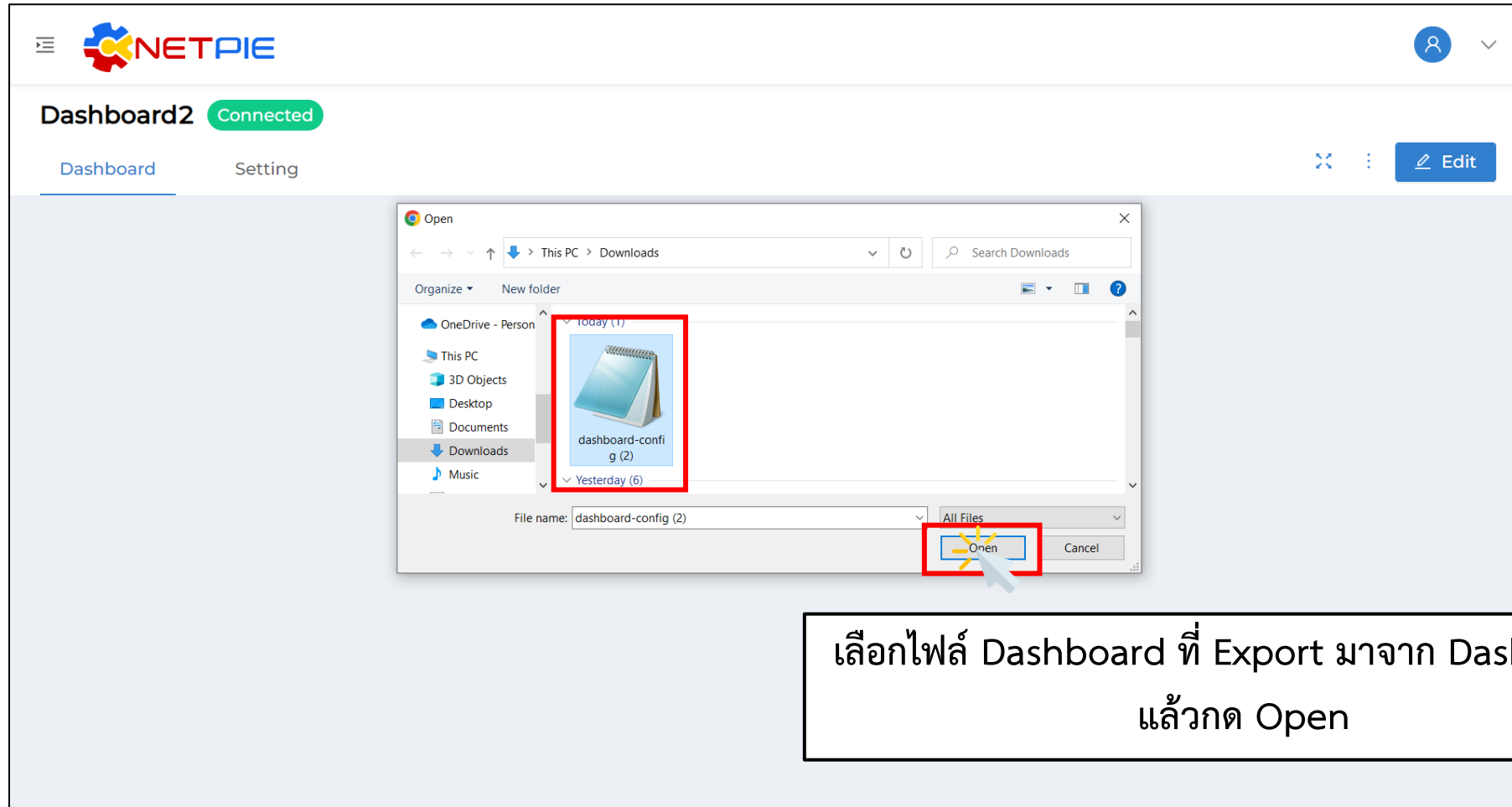


เข้ามาที่ Dashboard2 แล้วเลือกไปที่ Import

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

Import

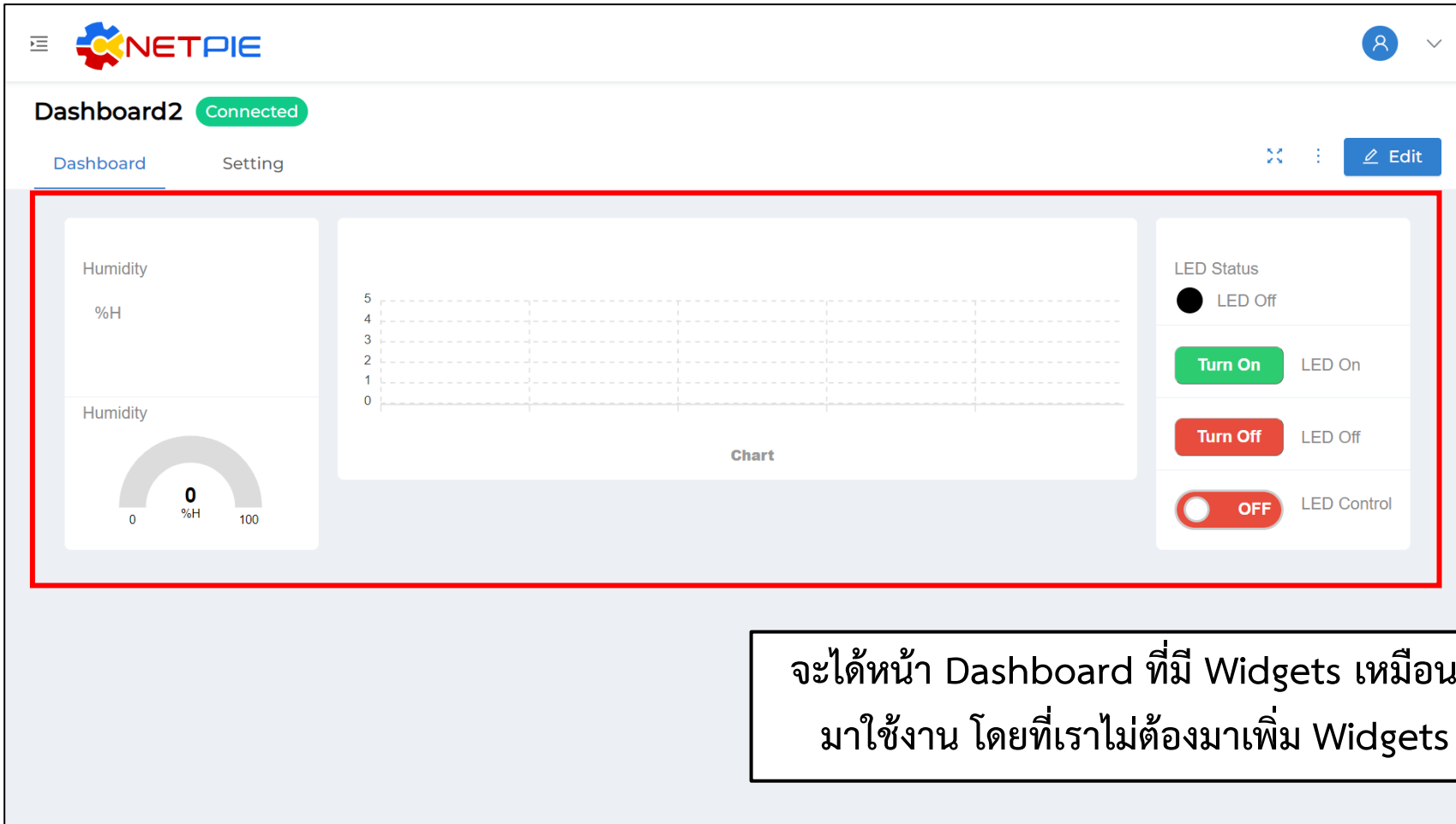


เลือกไฟล์ Dashboard ที่ Export มาจาก Dashboard1 แล้วกด Open

ใบงานที่ 5.3 ขั้นตอนการทดลอง

การทดลองที่ 4 การจัดการ Dashboard

Import



The screenshot displays the NETPIE dashboard interface. At the top left is the NETPIE logo. The main header shows 'Dashboard2' with a 'Connected' status indicator. Below the header are tabs for 'Dashboard' and 'Setting', and an 'Edit' button. The main content area is enclosed in a red border and contains three widgets: a 'Humidity' widget with a gauge showing 0% and a '%H' label; a 'Chart' widget with a grid and the label 'Chart'; and an 'LED Status' widget with three controls: 'LED Off' (radio button), 'Turn On' (green button), and 'LED Off' (red button). Below these is a red 'OFF' button labeled 'LED Control'. A text box at the bottom right of the dashboard area contains the following text:

จะได้หน้า Dashboard ที่มี Widgets เหมือนกับ Dashboard1 มาใช้งาน โดยที่เราไม่ต้องมาเพิ่ม Widgets ต่างๆ ขึ้นมาเอง

ใบงานที่ 5.4 การสร้างการแจ้งเตือนผ่าน Line Notify



LINE Notify
Connect Everything



ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

Device Trigger and Event Hook

Device Trigger

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

Event Hook

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

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

ส่วนต่างๆของ Device Trigger

Trigger ID

Automatically generate a trigger ID

Automatically generate a trigger ID

Status

Trigger Title

Title length of the 72-character limit.

When this happen ...

Event

SHADOW.UPDATED

DEVICE.STATUSCHANGED

Under conditions

This trigger fire when all these conditions are true.

Basic Custom

+ Add

Trigger ID (string) : รหัสของ Trigger ซึ่งระบบจะสร้างให้อัตโนมัติหรือผู้ใช้
ต้องการกำหนดเองก็ได้

Status : สถานะเปิด/ปิดการใช้งาน Trigger

Trigger Title (string) : ชื่อหรือคำอธิบายสั้น ๆ เกี่ยวกับ Trigger

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

ส่วนต่างๆของ Device Trigger

The screenshot shows the 'Add Trigger' configuration window. It has a title bar 'Add Trigger' with a close button. Below the title bar, there are two sections: 'Trigger ID' and 'Trigger Title'. The 'Trigger ID' section has a text input field with the placeholder 'Automatically generate a trigger ID' and a 'Status' toggle switch. The 'Trigger Title' section has a text input field with the placeholder 'Title length of the 72-character limit.'. Below these sections is the 'Event' section, which is highlighted with a red box. The 'Event' section has a 'When this happen ...' label and an 'Event' dropdown menu. The dropdown menu shows 'SHADOW.UPDATED' and 'DEVICE.STATUSCHANGED'. Below the dropdown menu is an 'Under conditions' section with a 'Basic' tab selected and a 'Custom' tab. The 'Under conditions' section has a text input field and a '+ Add' button.

Event : ประเภทการเปลี่ยนแปลงข้อมูลของ Device (Device Shadow)

มี 2 ประเภท คือ

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

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

1. ต้องการให้ Trigger ทุกครั้งที่สถานะการเชื่อมต่อ Platform เปลี่ยนไม่ว่าจะ Online เป็น Offline หรือ Offline เป็น Online ให้เช็คเงื่อนไข (Under conditions) ให้เป็นจริงเสมอ เช่น $true == true$ หรือ $1 == 1$ เป็นต้น

2. ต้องการให้ Trigger ในกรณีที่เปลี่ยนสถานะเป็น Online เท่านั้น ให้เช็ค ให้เช็คเงื่อนไข (Under conditions) เป็น $\$NEW.STATUS == 1$

3. ต้องการให้ Trigger ในกรณีที่เปลี่ยนสถานะเป็น Offline เท่านั้น ให้เช็ค ให้เช็คเงื่อนไข (Under conditions) เป็น $\$NEW.STATUS == 0$

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

ส่วนต่างๆของ Device Trigger

Do this ...

Action to

MyWebHook

With this context variables

+ Add context variables

Cancel OK

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

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

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

การอ้างอิงค่า Shadow ใน Trigger

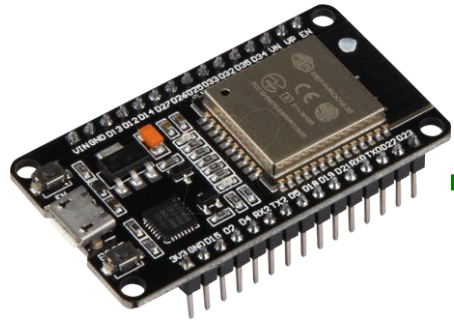
สำหรับการอ้างอิงค่าตัวแปร Shadow สามารถเรียกใช้ใน Condition หรือ Context Variable ของ Trigger มี 3 รูปแบบ ดังนี้

- 1 **\$CUR.พารของตัวแปร** คือ ค่าปัจจุบันล่าสุดที่ถูกอัปเดต (\$NEW merge \$PREV) โดยขึ้นต้นด้วย \$CUR ตามด้วย Path ตามโครงสร้างใน Shadow
- 2 **\$NEW.พารของตัวแปร** ค่าใหม่ที่ส่งมาอัปเดตลง Shadow โดยขึ้นต้นด้วย \$NEW ตามด้วย Path ตามโครงสร้างใน Shadow
- 3 **\$PREV.พารของตัวแปร** ค่าก่อนหน้าที่จะถูกอัปเดตลง Shadow โดยขึ้นต้นด้วย \$PREV ตามด้วย Path ตามโครงสร้างใน Shadow

ทั้งนี้ Device Trigger มีการอ้างอิงตัวแปรอื่นๆในระบบอีกมากมาย สามารถดูรายละเอียดได้ตามนี้
<https://docs.netpie.io/device-config.html#device-trigger-and-event-hook>

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

ตัวอย่างการทำงานของ Trigger and Event Hook



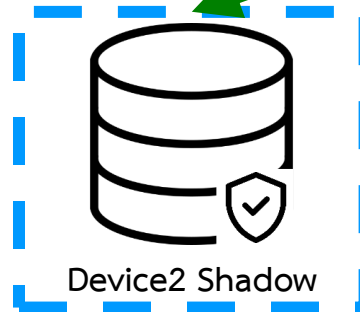
ESP8266/ESP32

Publish : @shadow/data/update

Event Hooks
LINE NOTIFY



NETPIE2020

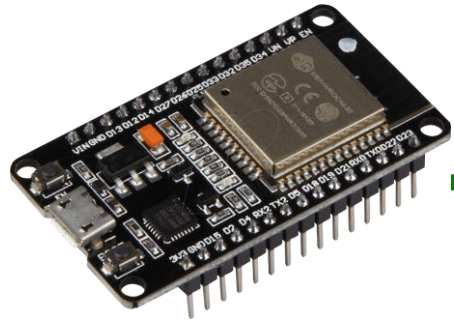


"temperature" : 74
"humidity" : 50
Status : Online

Trigger : DEVICE.STATUSCHANGE

ใบงานที่ 5.4 ทฤษฎีเบื้องต้น

ตัวอย่างการทำงานของ Trigger and Event Hook



ESP8266/ESP32

Publish : @shadow/data/update



Event Hooks
LINE NOTIFY



NETPIE2020



Device2 Shadow

"temperature" : 74

"humidity" : 50

Status : Offline

Trigger : DEVICE.STATUSCHANGE

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



LINE Notify
Connect Everything

เราจะทำการสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify ซึ่งเป็น 3rd Party ที่ได้รับความนิยมสำหรับการทำระบบแจ้งเตือน

NETPIE_Training / hook

Hook

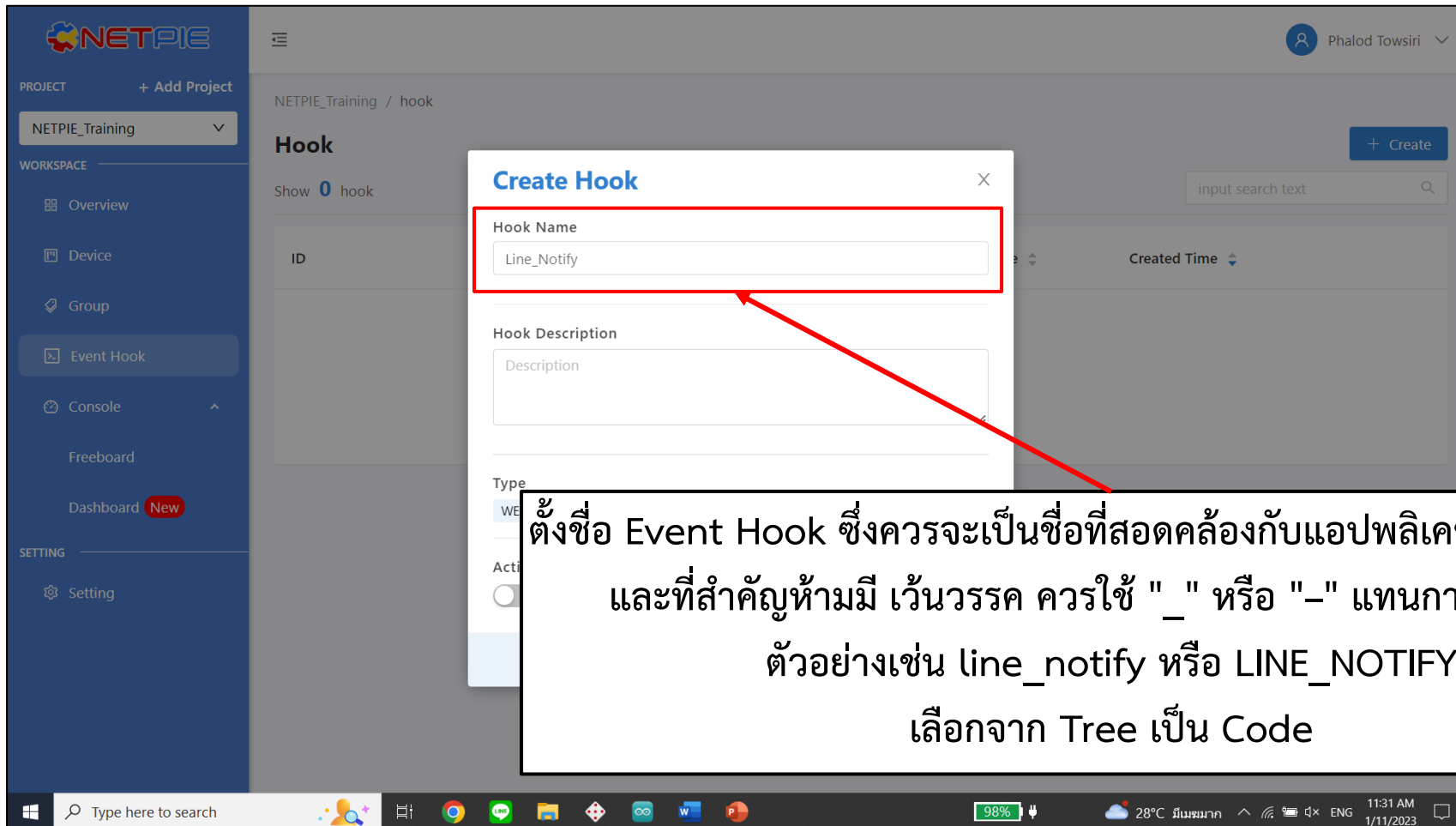
Show 0 hook

input search text

ID	Name	Active	Created Time
No Data			

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



ตั้งชื่อ Event Hook ซึ่งควรจะเป็นชื่อที่สอดคล้องกับแอปพลิเคชันที่เรียกใช้งาน และที่สำคัญห้ามมี เว้นวรรค ควรใช้ "_" หรือ "-" แทนการเว้นวรรค ตัวอย่างเช่น line_notify หรือ LINE_NOTIFY เลือกจาก Tree เป็น Code

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

The screenshot displays the NETPIE web application interface. On the left, a sidebar menu includes options like Overview, Device, Group, Event Hook, Console, Freeboard, and Dashboard. The main content area shows the 'Hook' management page for the 'NETPIE_Training' project. A 'Create Hook' modal dialog is open, featuring a 'Hook Name' input field containing 'Line_Notify', a 'Hook Description' text area, a 'Type' dropdown set to 'WEB HOOK', and an 'Active' toggle switch. The 'SAVE' button at the bottom right of the dialog is highlighted with a red box and a hand cursor. The background shows a table with columns for 'ID' and 'Created Time'.

ใบงานที่ 5.4 ขั้นตอนการทดลอง

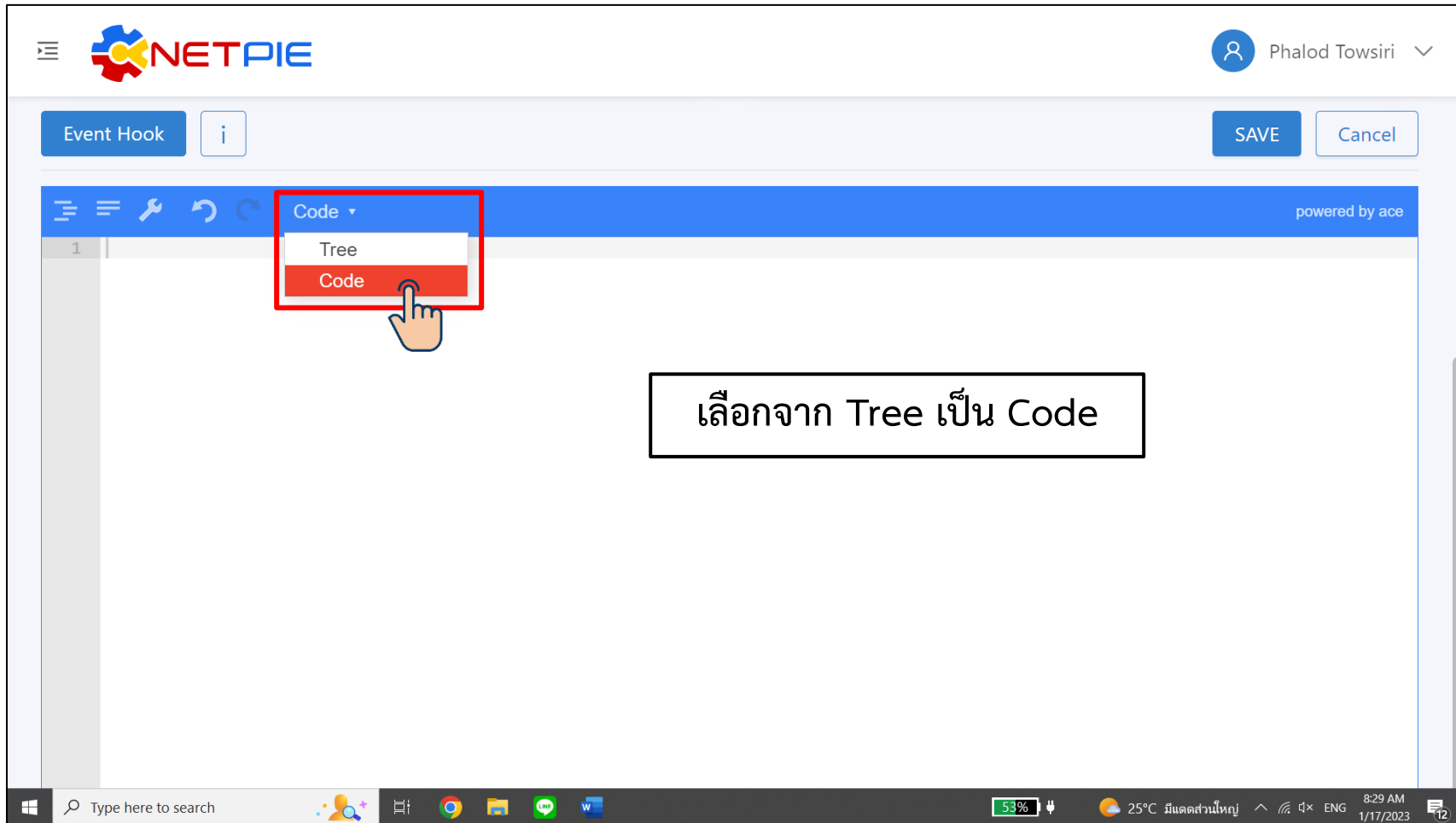
การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

The screenshot displays the NETPIE web application interface. On the left is a blue sidebar with the NETPIE logo and navigation menus for PROJECT, WORKSPACE, and SETTING. The main content area shows the 'Hook' management page for the 'NETPIE_Training' project. A table lists the hooks, with one entry highlighted by a red box and a hand cursor pointing to it. The table has columns for ID, Name, Active status, and Created Time. Below the table is a pagination control showing '1-1 of 1 items' and '10 / page'. The Windows taskbar is visible at the bottom of the screen.

ID	Name	Active	Created Time
H423285472342	Line_Notify	Disabled	2023-01-11 11:35

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify



The screenshot shows the NETPIE web interface. At the top left is the NETPIE logo. The user's name 'Phalod Towsiri' is displayed at the top right. Below the logo, there is a navigation bar with 'Event Hook' and an information icon. To the right of this bar are 'SAVE' and 'Cancel' buttons. The main content area has a blue header with a 'Code' dropdown menu. The dropdown menu is open, showing 'Tree' and 'Code' options. A red box highlights the 'Code' option, and a hand cursor is pointing at it. A text box in the center of the screen contains the Thai text 'เลือกจาก Tree เป็น Code'. The bottom of the screenshot shows a Windows taskbar with various icons and system information.

เลือกจาก Tree เป็น Code

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

Event Hook Example เป็นตัวอย่างการทำ Line Notify ซึ่งกำหนดค่าได้ 4 Attributes

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

คัดลอกทั้งหมดไปไว้ใน Event Hook บน NETPIE2020

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

NETPIE

PROJECT + Add Project

NETPIE_Training

WORKSPACE

- Overview
- Device
- Group
- Event Hook
- Console

SETTING

- Setting

NETPIE_Training / hook / Line_Notify

Line_Notify

created date: 2023-01-11

Edit

อย่าลืมกด Save เพื่อบันทึก

Event Hook

SAVE Cancel

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

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 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 ส่วนของข้อมูลในที่นี่คือ ข้อความที่จะส่งไปยังปลายทาง

header คือ ข้อมูลเพิ่มเติมที่ต้องการส่งไปยังปลายทาง เช่น Authorization, Content-Type เป็นต้น

method คือ ส่วนที่กำหนดว่าปลายทางต้องการให้ส่งไปในรูปแบบไหน GET, POST หรือ PUT

uri คือ Endpoint ปลายทางที่กำหนดว่าต้องการให้ส่งไปที่ใด

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

The screenshot shows the NETPIE web interface. On the left is a sidebar with navigation options: PROJECT (NETPIE_Training), WORKSPACE (Overview, Device, Group, Event Hook, Console), and SETTING (Setting). The main content area displays the configuration for an Event Hook named 'Line_Notify', created on 2023-01-11. The 'Configuration' section shows 'Type' as 'WEB HOOK' and 'Status' as 'Disabled'. A red box highlights the 'Edit' button and the 'Status' field. A text box with a black border contains the text: 'Event Hook ที่สร้างขึ้น จะยังไม่ถูกเปิดใช้งานสังเกตจาก Status เป็น Disabled จึงต้องเปิดใช้งานโดยกดไปที่ Edit'. Below the configuration, a code editor shows the following JSON configuration:

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

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 1 การสร้าง Event Hook สำหรับเรียกใช้งาน Line Notify

The screenshot displays the NETPIE web application interface. On the left is a dark blue sidebar with navigation options: PROJECT (+ Add Project), WORKSPACE (Overview, Device, Group, Event Hook, Console), and SETTING (Setting). The main content area shows the 'Event Hook' configuration for 'Line_Notify'. An 'Edit Hook' modal dialog is open, containing the following fields:

- Hook Name:** Line_Notify
- Hook Description:** Description
- Type:** WEB HOOK
- Active:** A toggle switch is turned on and highlighted with a red box. A hand icon is pointing at it.

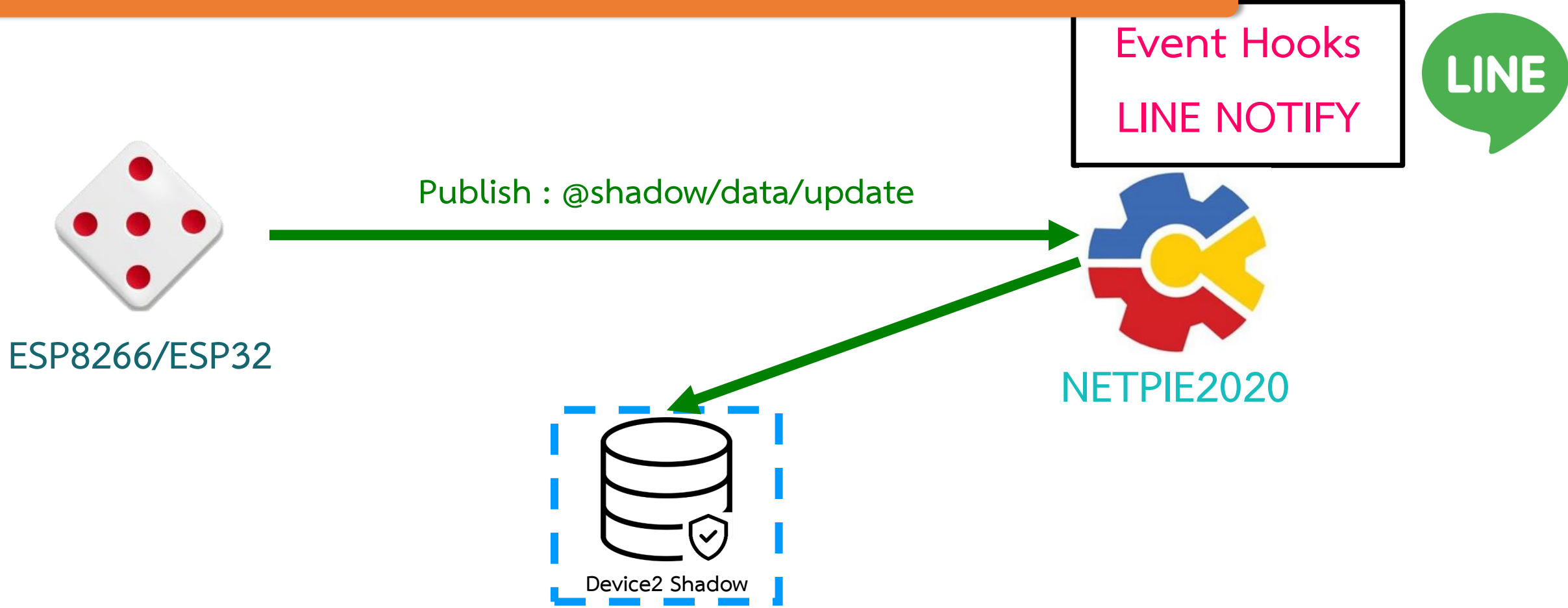
Buttons for 'CANCEL' and 'SAVE' are visible at the bottom of the dialog. In the background, a code editor shows a JSON snippet:

```
1 {  
2   "uri":  
3   "method":  
4   "headers":  
5   "authentication":  
6   "contentType":  
7 },  
8 "body":  
9 }
```

กดที่ Active เพื่อเปิดการใช้งาน Event Hook

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



Trigger : SHADOW UPDATE & DEVICE.STATUSCHANGE

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

NETPIE Training / device / Device1

Device1

created date: 2023-01-10

[Edit](#)

Detail

Key

Client ID	ec10973d-500a-4b56-b773-53d294135b28	Copy
Token	4jy5C1BjZWCWii66npdHfNVB9TieAHq6	Copy
Secret	rDBU3dE--F)DW40!PZ7t4e7btiVrW-pT	Copy
Status	Online	Refresh
Enable	<input checked="" type="checkbox"/>	

มาที่หน้า Device Trigger ของ Device1

เลือก Add Trigger

Shadow Schema **Trigger** Feed [i](#)

Trigger (0)

Status: [Enabled all](#) [Disabled all](#)

+ Add trigger

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

Add Trigger

Trigger ID

Automatically generate a trigger ID

Automatically generate a trigger ID

Trigger Title

Temperature Change

When this happen ...

Event

SHADOW.UPDATED

DEVICE.STATUSCHANGED

Under conditions

This trigger fire when all these conditions are true.

+ Add

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

ตั้งชื่อ Trigger

เลือก SHADOW.UPDATED

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

When this happen ...

Event

SHADOW.UPDATED

DEVICE.STATUSCHANGED

Under conditions

This trigger fire when all these conditions are true.

Basic Custom

\$PREV.temperature

!=

\$NEW.temperature

+ Add

เลือก \$PREV.temperature

เลือก \$NEW.temperature

เลือกเป็นเงื่อนไข != (ไม่เท่ากับ)

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

Do this ...

Action to

LINE_NOTIFY

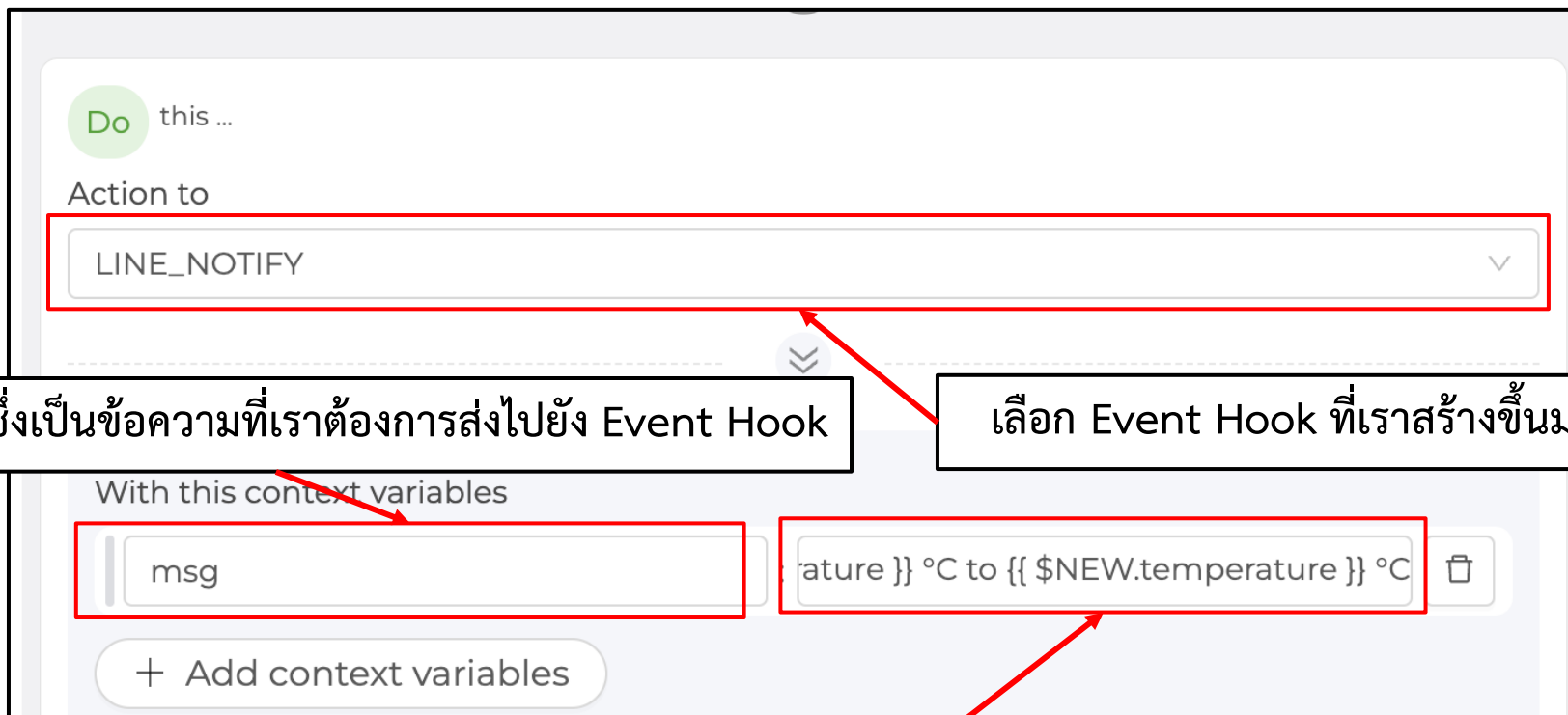
With this context variables

+ Add context variables

เลือก Event Hook ที่เราสร้างขึ้นมาในการทดลองที่ 1

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



เลือกเป็น msg ซึ่งเป็นข้อความที่เราต้องการส่งไปยัง Event Hook

เลือก Event Hook ที่เราสร้างขึ้นมาในการทดลองที่ 1

กรอกข้อความที่เราต้องการส่ง เช่น

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

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

Do this ...

Action to

LINE_NOTIFY

With this context variables

msg : ature }} °C to {{ \$NEW.temperature }} °C

+ Add context variables

เพิ่มตัวแปรที่จะส่งไปยัง Event Hook

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

Do this ...

Action to

LINE_NOTIFY

with this context variables

msg : ature }} °C to {{ \$NEW.temperature }} °C

linetoken :

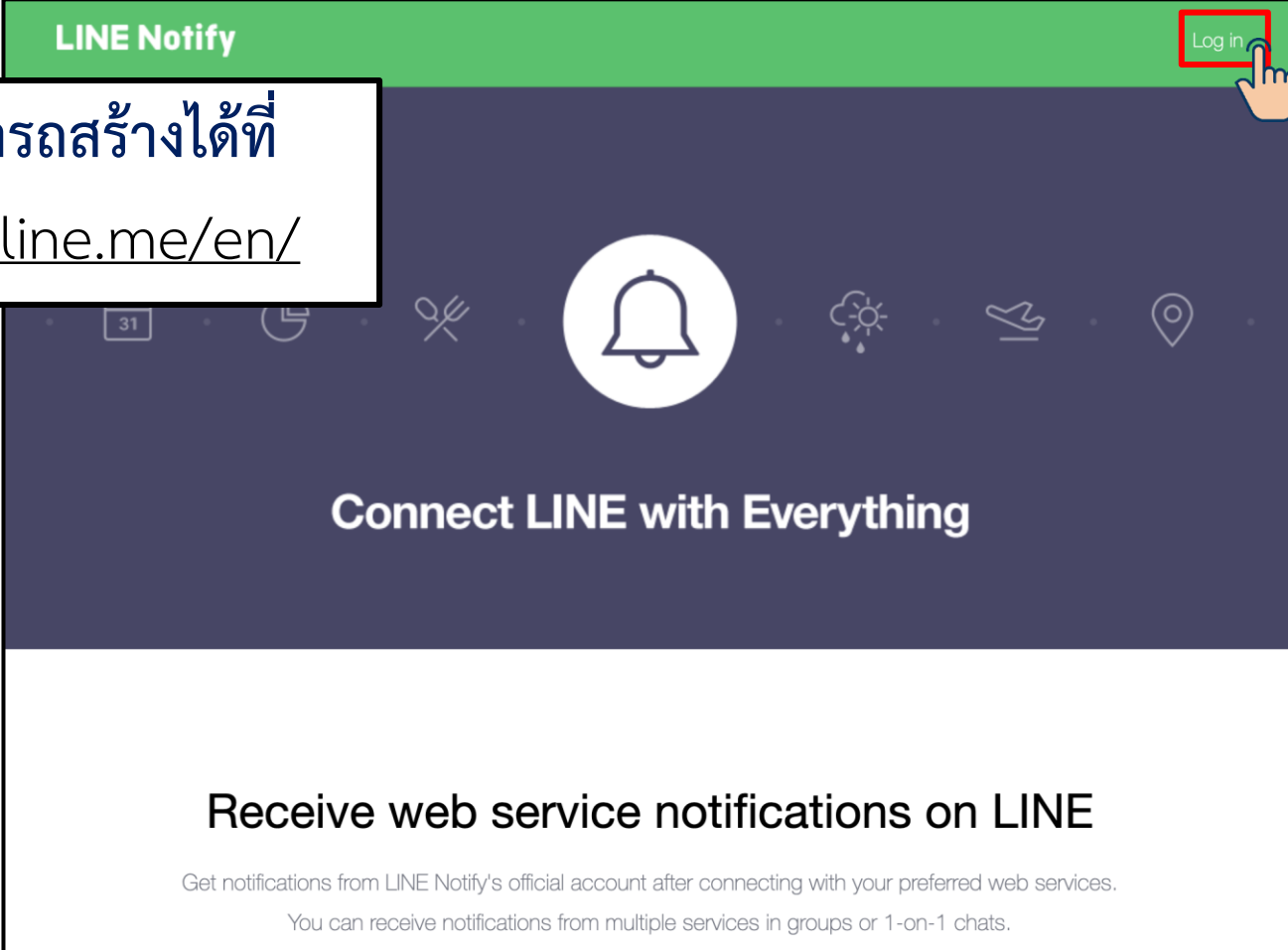
+ Add context variables

เลือกเป็น linetoken ซึ่งเป็น token ที่ใช้สำหรับระบุตัวตนบน Line

กรอก line token ซึ่งจะต้องนำมาจาก การ Generate บน Line Notify

ใบงานที่ 5.4 ขั้นตอนการทดลอง

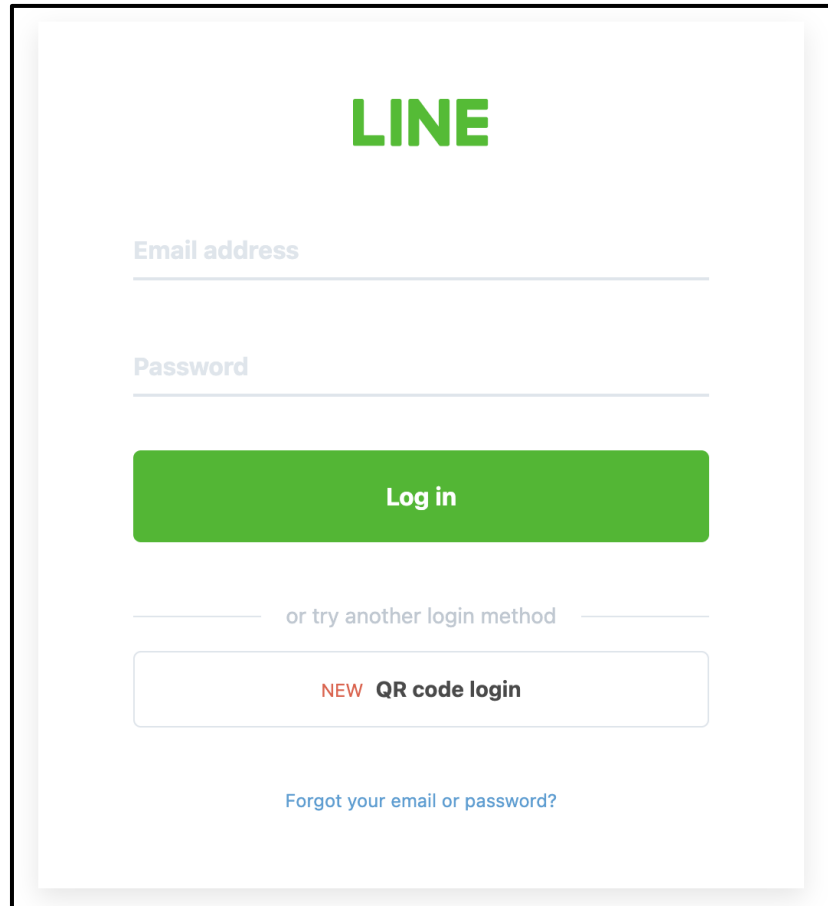
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



The screenshot shows the LINE Notify website. At the top, there is a green header with the text "LINE Notify" on the left and a "Log in" button on the right, which is highlighted with a red box and a hand cursor. Below the header is a dark blue section with a white bell icon in the center and the text "Connect LINE with Everything". Below this is a white section with the text "Receive web service notifications on LINE" and a smaller line of text: "Get notifications from LINE Notify's official account after connecting with your preferred web services. You can receive notifications from multiple services in groups or 1-on-1 chats." A white callout box on the left contains the text: "Token Line สามารถสร้างได้ที่ <https://notify-bot.line.me/en/>".

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

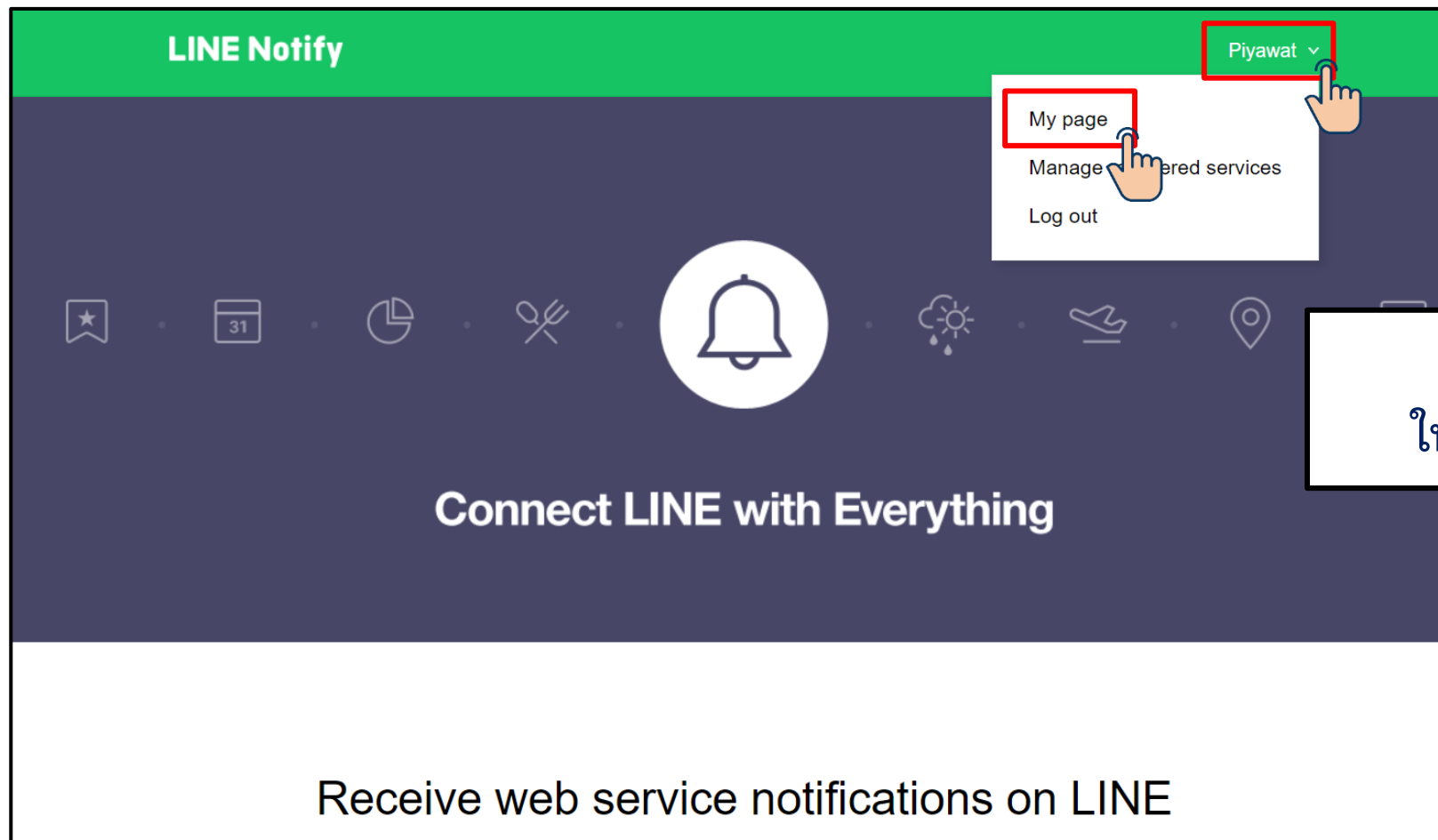


The image shows a screenshot of the LINE login interface. At the top, the word "LINE" is displayed in green. Below it, there are two input fields: "Email address" and "Password". A green "Log in" button is positioned below the password field. Underneath the button, there is a link that says "or try another login method". Below this link is a button labeled "NEW QR code login". At the bottom of the page, there is a link that says "Forgot your email or password?".

Login เข้าสู่ระบบโดยใช้
E-mail และ Password ของ LINE ที่ใช้งาน

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



เมื่อเข้าสู่ระบบได้แล้ว
ให้เลือกที่ชื่อ >> My page

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

เลื่อนมาข้างล่างแล้วเลือก

Generate Token

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

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

Generate token

Please enter a token name to be displayed before each notification.

NETPIE2020 Training

Select a chat to send notifications to.

Search by group name

1-on-1 chat with LINE Notify

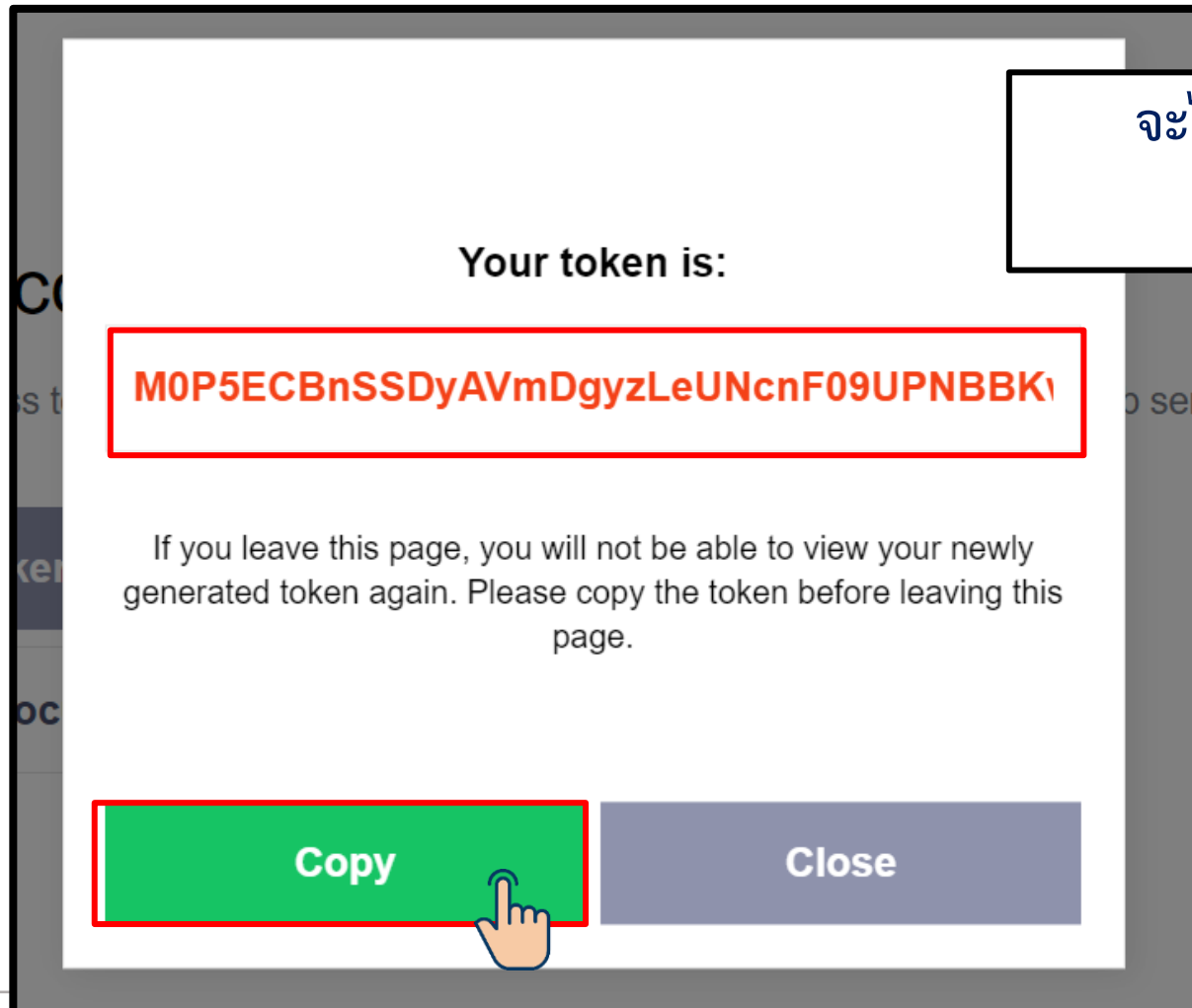
Note: Revealing your personal access token can allow a third party to obtain the names of your connected chats as well as your profile name.

Generate token

ตั้งชื่อหัวข้อ การสนทนา
แล้วเลือกแชทกับตัวเอง และคลิก Generate Token

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



จะได้รับ Token ให้ Copy แล้วไปวางใน
linetoken ใน Device Trigger

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

Do this ...

Action to

LINE_NOTIFY

With this context variables

msg : ature }} °C to {{ \$NEW.temperature }} °C

linetoken : z3VpYQNxWlaDrUlwi60nAmNsZCGja8

+ Add context variables

กรอก line token ที่สร้างมา

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

The screenshot shows the IFTTT configuration interface. At the top, there is a condition field with the text "!=" and a dropdown menu showing "\$NEW.temperature". Below this is a "+ Add" button. A downward arrow indicates the next step. The main section is titled "Do this ..." and "Action to" with a dropdown menu showing "LINE_NOTIFY". Below this is a section titled "With this context variables" containing two rows: "msg" with the value "ature }} °C to {{ \$NEW.temperature }} °C" and "linetoken" with the value "z3VpYONxWlaDrUlwj60nAmNsZCGiaB". There is a "+ Add context variables" button at the bottom of this section. At the bottom right of the interface, there are "Cancel" and "OK" buttons. A hand icon is pointing to the "OK" button, which is highlighted with a red box.

เมื่อสร้างทั้งหมดเสร็จแล้วทำการเลือก OK

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

The screenshot displays the NETPIE web interface. At the top, the user 'Phalod Towsiri' is logged in. The main area shows the configuration for a Device Trigger. The 'Token' is '4jy5C1BjZWCWii66npdHfNVB9TieAHq6' and the 'Secret' is 'rDBU3dE--F)DW40!PZ7t4e7btiVrW-pT'. The 'Status' is 'Online' and the 'Enable' toggle is turned on. Below this, there are tabs for 'Shadow', 'Schema', 'Trigger', and 'Feed'. The 'Trigger' tab is active, showing a list of triggers. The first trigger is 'Temperature Change' with ID 'TD091963199977'. Its status is 'Enabled all'. A red box highlights the 'Status' toggle, which is currently turned on. A hand icon is pointing to the toggle. Below the trigger list, there is a callout box with the text 'ทำการเลือก Status เป็น Enable'. The trigger configuration details are as follows:

- When:**
 - Event : SHADOW UPDATED
 - Under conditions : \$PREV.temperature != \$NEW.temperature
- Do:**
 - Action to : Line_Notify
 - with this context variables :
 - msg : Temperature is change from {{ \$PREV.temperature }} °C to {{ \$NEW.temperature }} °C linetoken : hj3fh9d28PWm3w5i631N20N1USb8A7sgwoq22

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

The screenshot shows the NETPIE web interface. At the top left is the NETPIE logo. At the top right, the user name 'Phalod Towsiri' is displayed. Below the header, there are fields for Client ID, Token, and Secret, each with a 'Copy' button. The Status is 'Online' and the Enable toggle is turned on. Below these fields are tabs for 'Shadow', 'Schema', 'Trigger', and 'Feed', along with 'SAVE' and 'Cancel' buttons. At the bottom, a 'Tree' view shows a data structure with the following values: humidity: 65, light: 120, place: Piyawat Home, and temperature: 20. A red box highlights the 'Tree' view, and a callout box points to it with the text 'ตรวจสอบข้อมูล temperature ก่อนส่งค่าเพื่อทดสอบ Line Notify'.

Field	Value	Action
Client ID	ec10973d-500a-4b56-b773-53d294135b28	Copy
Token	4jy5C1BjZWCWii66npdHfNVB9TieAHq6	Copy
Secret	rDBU3dE--F)DW40!PZ7t4e7btiVrW-pT	Copy
Status	Online	Refresh
Enable	Toggle (On)	

```
Tree | Last Update : 24-01-23 13:57
Select a node...
├── object {4}
│   ├── humidity : 65
│   ├── light : 120
│   ├── place : Piyawat Home
│   └── temperature : 20
```

ตรวจสอบข้อมูล temperature ก่อนส่งค่าเพื่อทดสอบ Line Notify

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

The screenshot shows the MQTT Box interface with the following configuration:

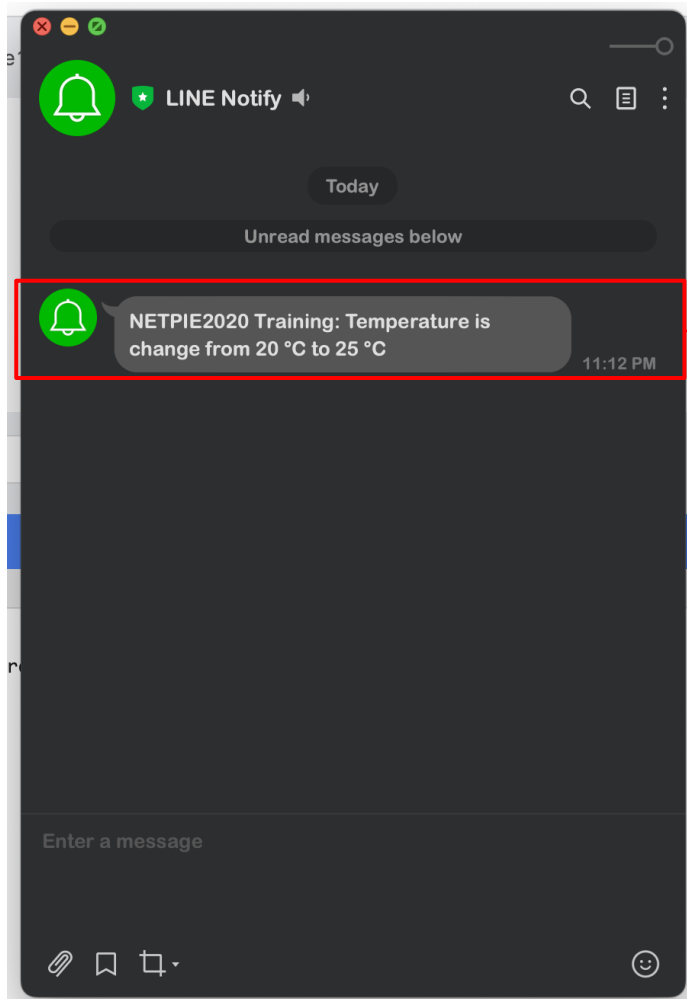
- Topic to publish:** @shadow/data/update
- QoS:** 0 - Almost Once
- Retain:**
- Payload Type:** Strings / JSON / XML / Characters
- Payload:** {"data": {"temperature": 25}}
- Buttons:** Publish (highlighted with a hand icon), Add publisher, Add subscriber, Subscribe

Annotations and instructions:

- ทำการทดสอบส่งข้อมูล โดยใช้ MQTT Box
- เลือก Topic ในการส่งคือ @shadow/data/update
- เลือก Payload เป็น {"data": {"temperature": 25}} โดยค่าของ temperature จะต้องไม่เท่ากับใน shadow เพื่อให้เกิดการแจ้งเตือน
- เมื่อตั้งค่าทั้งหมดเสร็จแล้วเลือก Publish

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ข้อความที่ถูกส่งเข้ามาใน Line Notify

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

The screenshot shows the NETPIE web interface. At the top, there is a navigation menu with 'Shadow', 'Schema', 'Trigger', and 'Feed' tabs. The 'Trigger' tab is selected. A text box in the center of the page reads: "ถัดไปจะสร้างการแจ้งเตือนประเภท Device Status Change นั่นคือ มีการแจ้งเตือนในกรณีเปลี่ยนสถานะการเชื่อมต่อ". Below the tabs, there is a '+ Add trigger' button highlighted with a red box and a hand cursor. A text box next to it says "เลือก Add Trigger". The main content area shows a configuration for a 'Temperature Change' trigger (ID: TD0919). The configuration includes a 'When' section with the event 'SHADOW UPDATED' and the condition '\$PREV.temperature != \$NEW.temperature'. The 'Do' section is set to 'Line_Notify' with the message: 'msg: Temperature is change from {{ \$PREV.temperature }} °C to {{ \$NEW.temperature }} °C linetoken : hj3fh9d28PWm3w5i631N20N1USb8A7sgwoq22'. The status is set to 'Enabled all'.

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

Add Trigger

Trigger ID

Automatically generate a trigger ID

Automatically generate a trigger ID

Trigger Title

Status Change

When this happen ...

Event

SHADOW.UPDATED

DEVICE.STATUSCHANGED

Under conditions

This trigger fire when all these conditions are true.

Basic Custom

+ Add

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

ตั้งชื่อ Trigger

เลือก DEVICE.STATUSCHANGED

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

The screenshot shows the configuration interface for a Device Trigger. At the top, it says "When this happen ...". Below that, the "Event" is set to "SHADOW.UPDATED" and "DEVICE.STATUSCHANGED". Under the "Under conditions" section, there is a note: "This trigger fire when all these conditions are true." There are two tabs: "Basic" and "Custom". A single condition is added, showing a text input field with "true", a dropdown menu with "==" selected, and another text input field with "true". Below the condition list is a "+ Add" button. Three red boxes with arrows point to specific parts of the interface:

- A box labeled "เลือกเป็น true" points to the first "true" value in the condition.
- A box labeled "เลือกเป็น true" points to the second "true" value in the condition.
- A box labeled "เลือกเป็นเงื่อนไข == (เท่ากับ)" points to the "==" operator in the condition.

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

เลือก Event Hook ที่เราสร้างขึ้นมาในการทดลองที่ 1

เลือกเป็น msg ซึ่งเป็นข้อความที่เราต้องการส่งไปยัง Event Hook

กรอกข้อความที่เราต้องการส่ง เช่น
Status change from {{ \$OLD.STATUSTEXT }} to {{ \$NEW.STATUSTEXT }}

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

Do this ...

Action to

LINE_NOTIFY

เลือกเป็น linetoken ซึ่งเป็น token ที่ใช้สำหรับระบุตัวตนบน Line

msg : Status change from {{ \$OLD.STATUSTE

linetoken : 2rvTOozErUFnh4J8fqilmKpTJ71TZHv6

+ Add context variables

กรอก line token ซึ่งจะต้องนำมาจาก การ Generate บน Line Notify

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

Under conditions

This trigger fire when all these conditions are true.

Basic Custom

true

== true

+ Add

Do this ...

Action to

LINE_NOTIFY

With this context variables

msg : Status change from {{{ \$OLD.STATUS}}

linetoken : 2rVTQozErUFnh4J8fqilmKpTJ71TZHv6

+ Add context variables

Cancel OK

เมื่อสร้างทั้งหมดเสร็จแล้วทำการเลือก OK

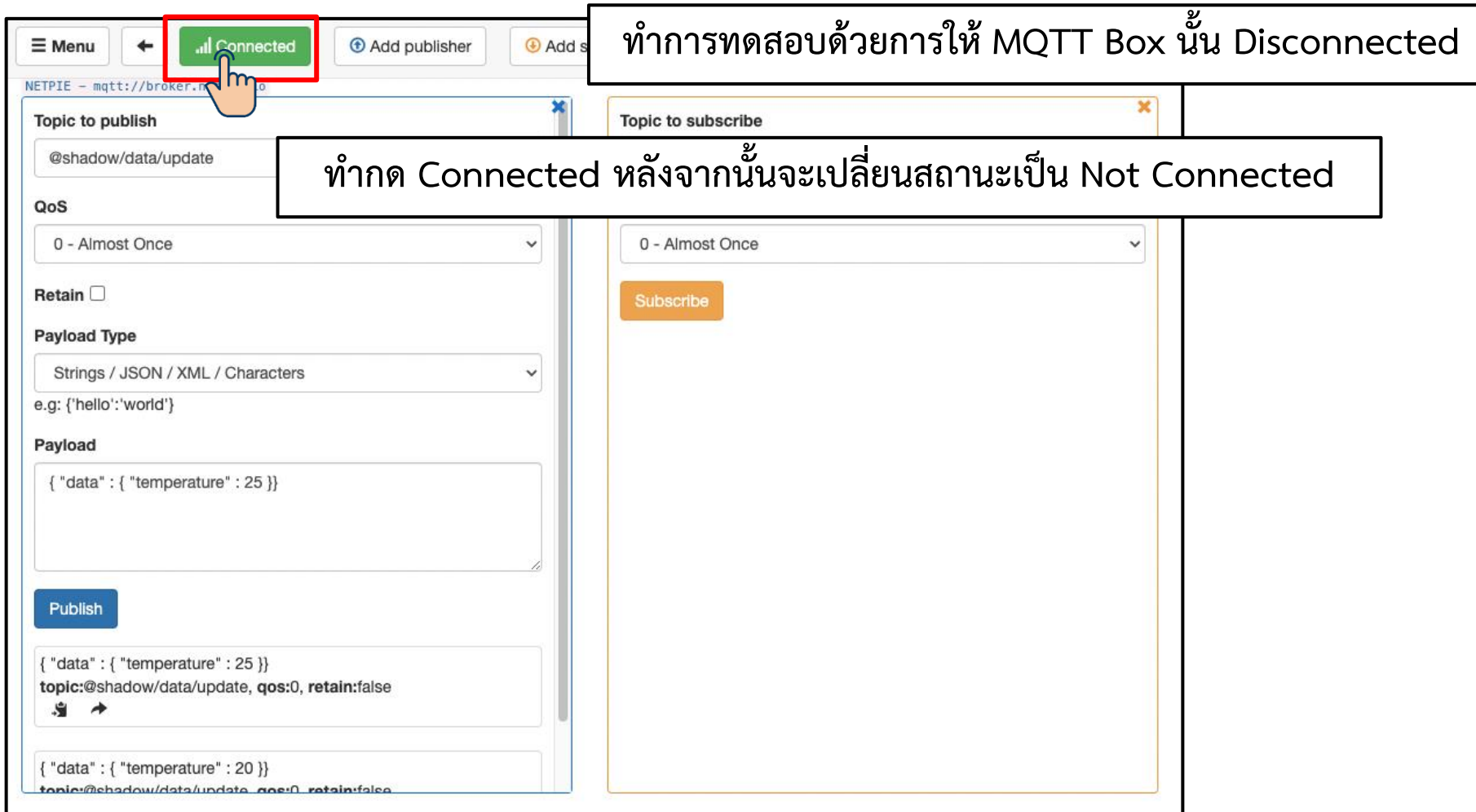
ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify

The screenshot displays the NETPIE web interface. At the top, there is a navigation menu, the NETPIE logo, and a user profile for 'Phalod Towsiri'. Below this, a 'Trigger (2)' section is visible, containing two triggers. The first trigger, 'Temperature Change' (ID: TD091963199977), is configured with the event 'SHADOW UPDATED' and the condition '\$PREV.temperature != \$NEW.temperature'. Its action is 'Line_Notify' with a message template: 'msg : Temperature is change from {{ \$PREV.temperature }} °C to {{ \$NEW.temperature }} °C'. The second trigger, 'Status Change' (ID: TD176284731760), is highlighted with a red box. It is configured with the event 'DEVICE STATUSCHANGED' and the condition 'true == true'. Its action is 'Line_Notify' with a message template: 'msg : Status change from {{ \$OLD.STATUSTEXT }} to {{ \$NEW.STATUSTEXT }}'. A callout box with a black border and white background points to the 'Status Change' trigger, containing the text 'Trigger ที่สร้างขึ้นใหม่'.

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



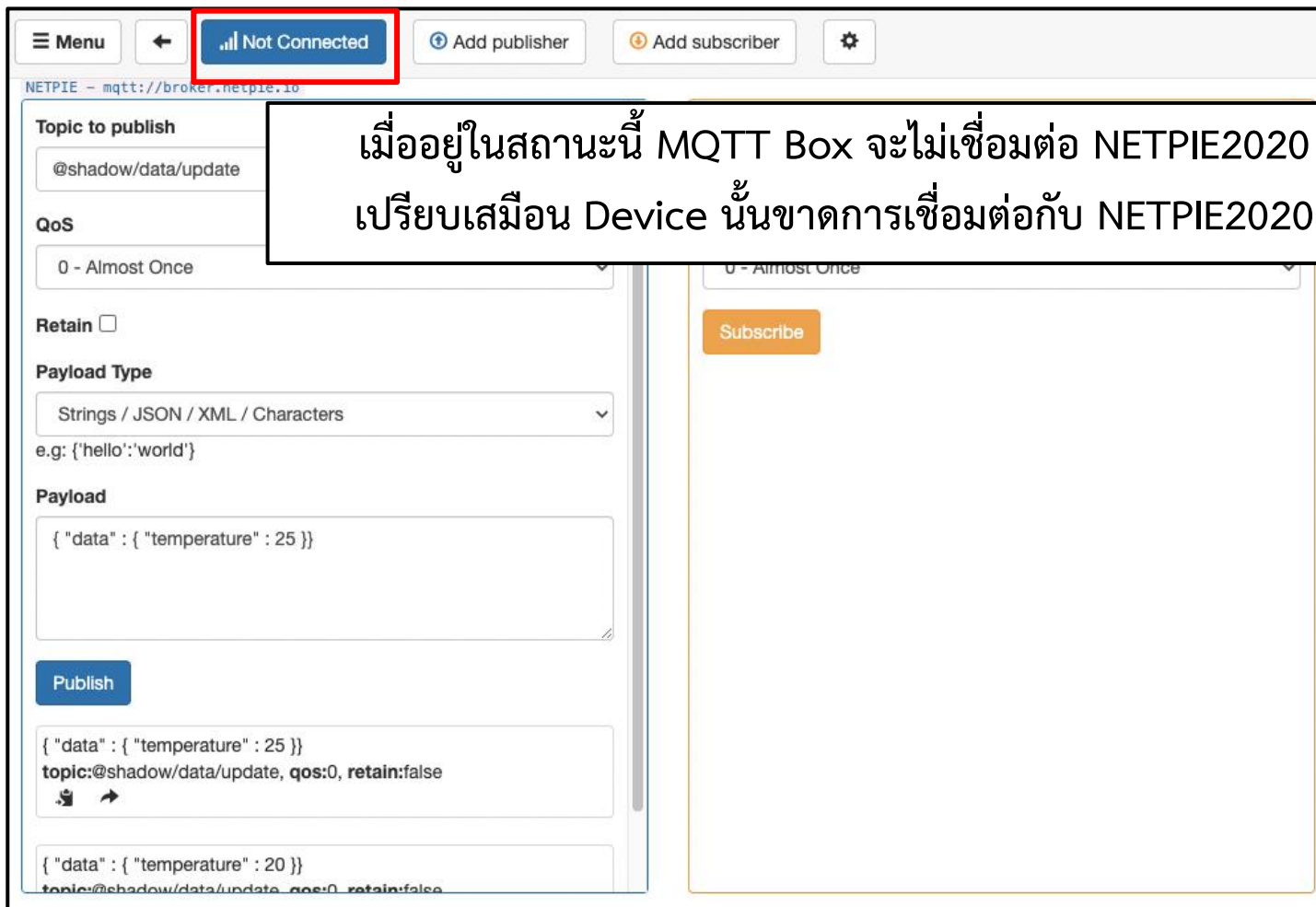
The screenshot shows the MQTT client interface. At the top left, there is a 'Menu' button and a 'Connected' status indicator with a green signal icon, which is highlighted by a red box and a hand cursor. To the right of the 'Connected' status, there are buttons for 'Add publisher' and 'Add s...'. Below the status bar, there are two main panels: 'Topic to publish' and 'Topic to subscribe'. The 'Topic to publish' panel has a text input field containing '@shadow/data/update', a 'QoS' dropdown menu set to '0 - Almost Once', a 'Retain' checkbox, a 'Payload Type' dropdown menu set to 'Strings / JSON / XML / Characters', and a 'Payload' text area containing the JSON object: `{ "data" : { "temperature" : 25 } }`. Below the payload area is a 'Publish' button. The 'Topic to subscribe' panel has a text input field, a 'QoS' dropdown menu set to '0 - Almost Once', and a 'Subscribe' button. The interface also shows a list of published messages at the bottom, including the JSON object from the payload area.

ทำการทดสอบด้วยการให้ MQTT Box นั้น Disconnected

ทำกด Connected หลังจากนั้นจะเปลี่ยนสถานะเป็น Not Connected

ใบงานที่ 5.4 ขั้นตอนการทดลอง

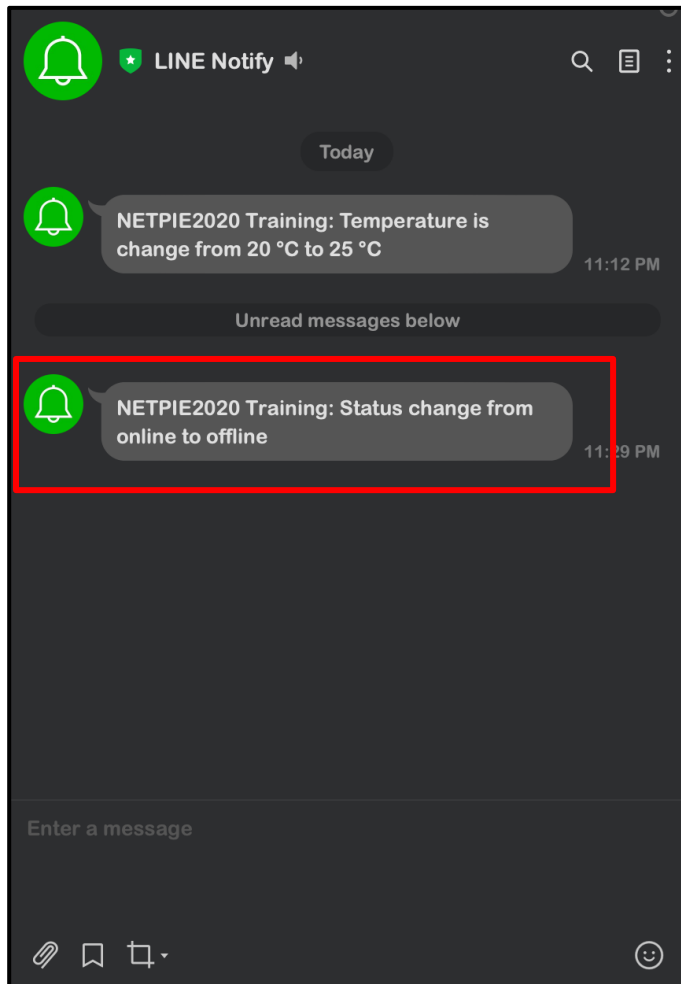
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



The screenshot shows the MQTT Box web interface. At the top, there is a status bar with a 'Not Connected' indicator highlighted by a red box. Below this, there are buttons for 'Add publisher', 'Add subscriber', and a settings gear. The main interface is split into two panels. The left panel is for publishing, with fields for 'Topic to publish' (set to '@shadow/data/update'), 'QoS' (set to '0 - Almost Once'), 'Retain' (unchecked), 'Payload Type' (set to 'Strings / JSON / XML / Characters'), and a 'Payload' field containing a JSON object: `{ "data" : { "temperature" : 25 } }`. A 'Publish' button is at the bottom of this panel. The right panel is for subscribing, with a 'Subscribe' button. A text box is overlaid on the interface, containing the text: 'เมื่ออยู่ในสถานะนี้ MQTT Box จะไม่เชื่อมต่อ NETPIE2020 เปรียบเสมือน Device นั้นขาดการเชื่อมต่อกับ NETPIE2020'.

ใบงานที่ 5.4 ขั้นตอนการทดลอง

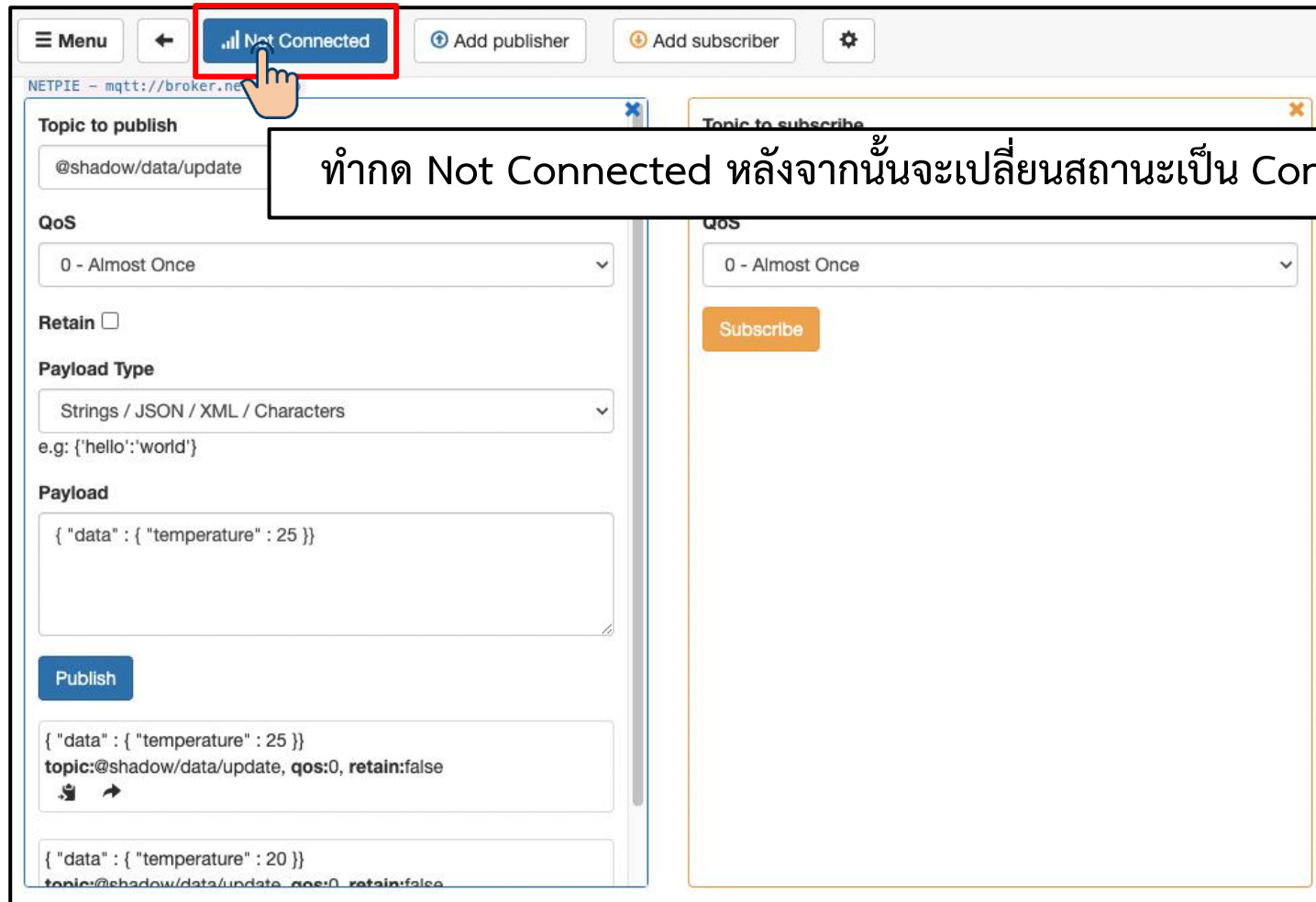
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ข้อความการแจ้งเตือนสถานะที่เปลี่ยนไปถูกส่งมายัง Line

ใบงานที่ 5.4 ขั้นตอนการทดลอง

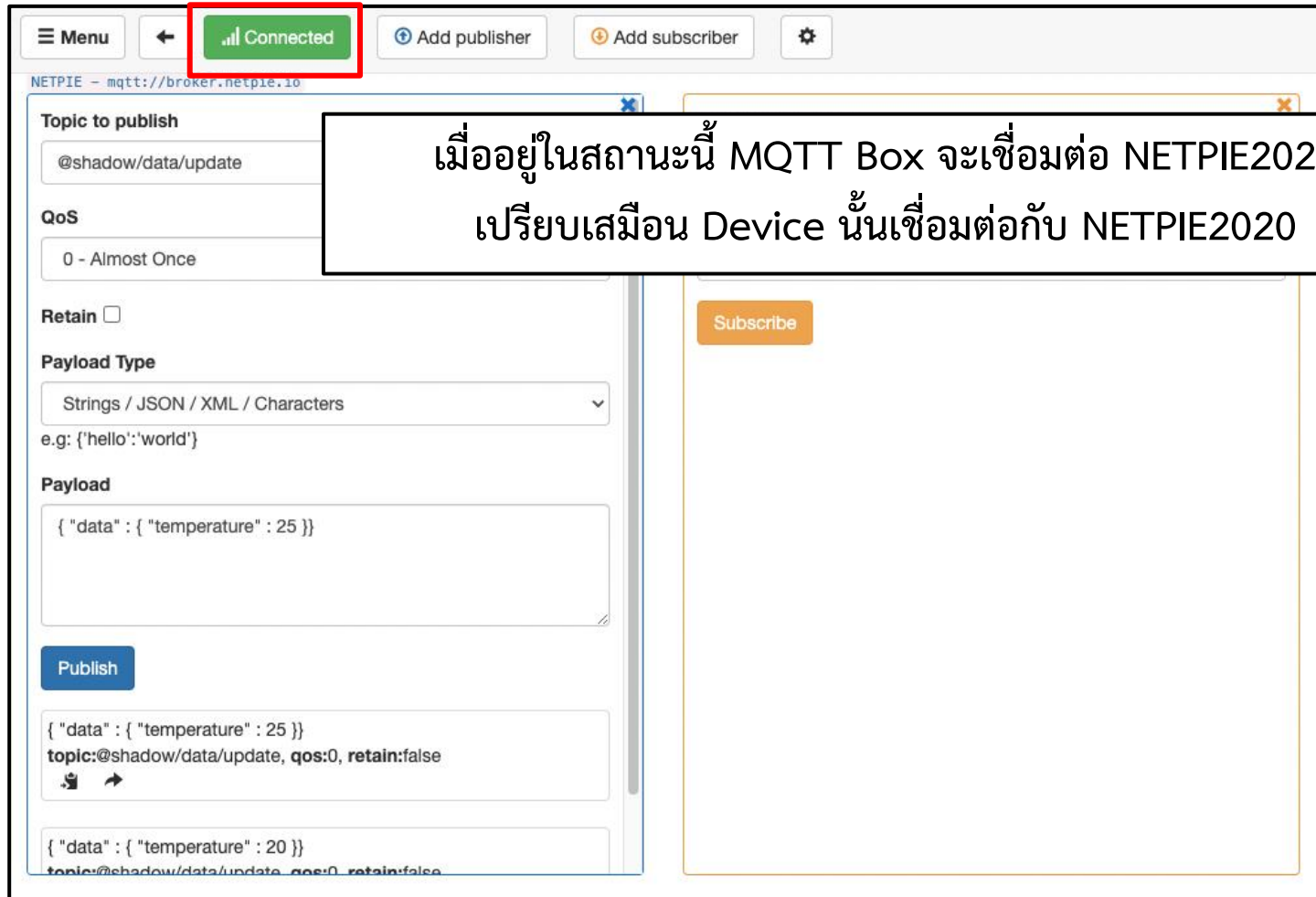
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



The screenshot shows the MQTT client interface. At the top, there is a status bar with a 'Not Connected' indicator, which is highlighted with a red box and a hand cursor. Below the status bar, there are two panels: 'Topic to publish' and 'Topic to subscribe'. The 'Topic to publish' panel has a text input field containing '@shadow/data/update', a 'QoS' dropdown set to '0 - Almost Once', a 'Retain' checkbox, a 'Payload Type' dropdown set to 'Strings / JSON / XML / Characters', and a 'Payload' text area containing '{ "data" : { "temperature" : 25 } }'. Below the payload area is a 'Publish' button. The 'Topic to subscribe' panel has a 'QoS' dropdown set to '0 - Almost Once' and a 'Subscribe' button. A text box is overlaid on the interface, containing the text: 'ทำกด Not Connected หลังจากนั้นจะเปลี่ยนสถานะเป็น Connected'.

ใบงานที่ 5.4 ขั้นตอนการทดลอง

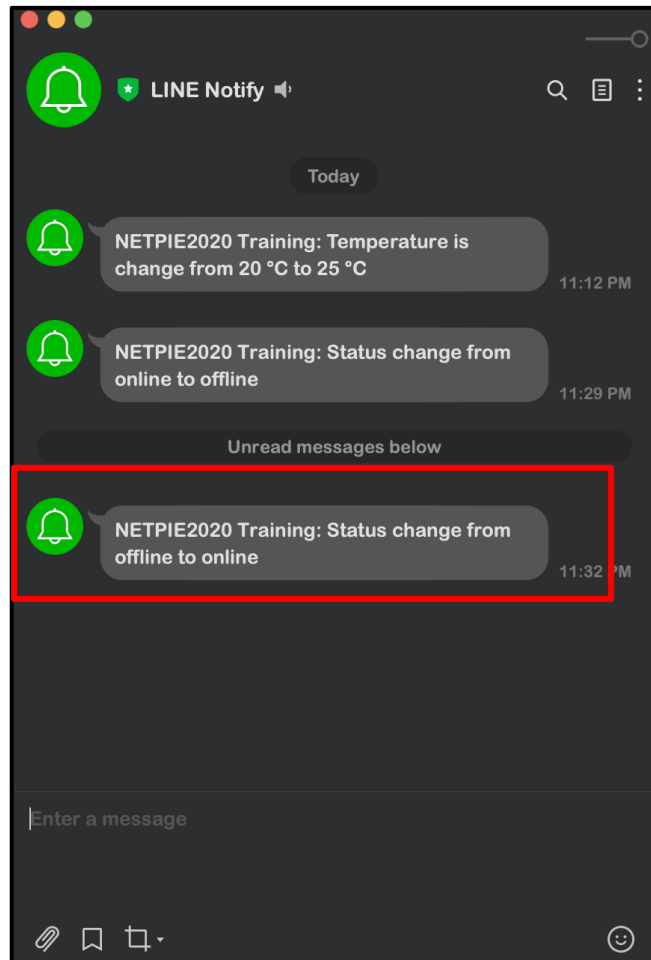
การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



The screenshot shows the MQTT Box web interface. At the top, there is a green 'Connected' status indicator with a signal strength icon, highlighted by a red box. Below it, there are buttons for 'Add publisher', 'Add subscriber', and a settings gear. The main area is divided into two panels. The left panel is for publishing, with fields for 'Topic to publish' (containing '@shadow/data/update'), 'QoS' (set to 0 - Almost Once), 'Retain' (unchecked), and 'Payload Type' (set to Strings / JSON / XML / Characters). The 'Payload' field contains a JSON object: `{ "data" : { "temperature" : 25 } }`. A 'Publish' button is at the bottom of this panel. The right panel is for subscribing, featuring a large orange 'Subscribe' button. A text box is overlaid on the interface, containing the text: 'เมื่ออยู่ในสถานะนี้ MQTT Box จะเชื่อมต่อ NETPIE2020' and 'เปรียบเสมือน Device นั้นเชื่อมต่อกับ NETPIE2020'.

ใบงานที่ 5.4 ขั้นตอนการทดลอง

การทดลองที่ 2 การสร้าง Device Trigger เพื่อสร้างเงื่อนไขสำหรับเรียกใช้งาน Line Notify



ข้อความการแจ้งเตือนสถานะที่เปลี่ยนไปถูกส่งมายัง Line