

NETPIE 2020

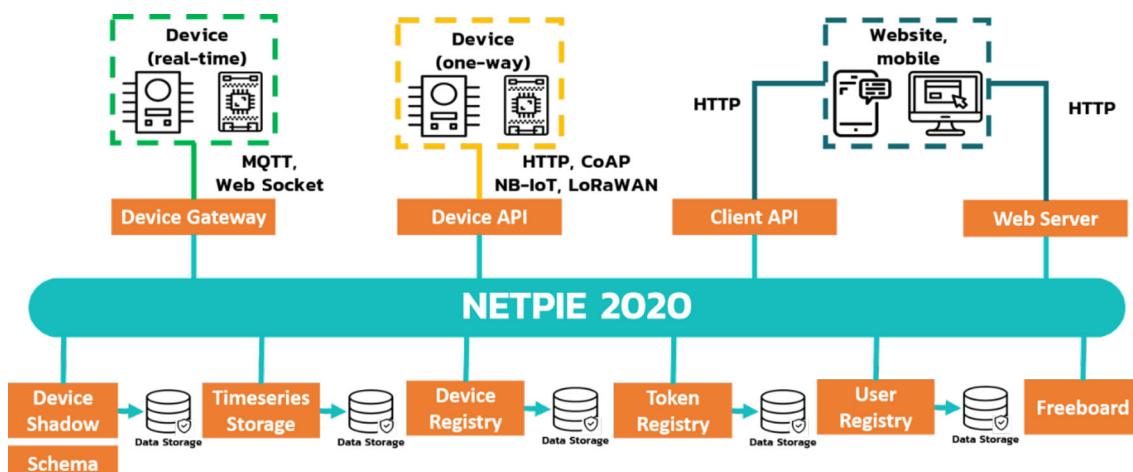


NETPIE 2020

ที่มา: <https://netpie.io/compare>

คุณสมบัติหลักๆของ NETPIE 2020 Platfrom ประกอบไปด้วย

1. การแสดงค่าข้อมูลจากเซ็นเซอร์หรืออุปกรณ์แบบ Real-time (Monitoring)
 2. การควบคุมการทำงานของอุปกรณ์ต่างๆ ผ่าน Cloud Platform (Controlling)
 3. การเก็บค่าข้อมูลที่ได้จากเซ็นเซอร์หรืออุปกรณ์ (Data Storage)
 4. การแจ้งเตือนความผิดปกติของเซ็นเซอร์หรืออุปกรณ์จากที่ได้กำหนดไว้ (Notification)
 5. การแสดงผลและควบคุมการทำงานของอุปกรณ์ผ่าน Dashboard (Dashboard for monitor & control)



แผนภาพระบบทั้งหมดของ NETPIE 2020

ที่มา: <https://docs.netpie.io/overview-netpie.html>

เริ่มต้นใช้งาน NETPIE 2020

เริ่มต้นลงทะเบียนใช้งาน และ เข้าสู่ระบบ NETPIE 2020 ผ่าน <https://netpie.io/>

NETPIE 2020
From Makers Nation
Toward Smart Nation

Connect Everything

NETPIE is an IoT cloud-based platform-as-a-service that helps connect your IoT devices together seamlessly by pushing the complexity from the hands of application developers or device manufacturers to the cloud

GET STARTED **WATCH VIDEO**

Introducing NETPIE 2020

NETPIE 2020 is the latest version of NETPIE. It aims to fulfill the needs of commercial users, especially those in the industrial sector. The platform's new version can shorten and ease the process of IoT product development considerably, from the stage of designing, prototyping, implementing right down to administering and maintaining. NETPIE 2020 comes with the newly designed architecture and many exciting new features.

ภาพหน้าหลัก NETPIE

ให้กดปุ่ม Sign up ด้านขวาเมื่อบน จะปรากฏหน้านี้

NETPIE 2020

EMAIL

NAME

ORGANIZATION

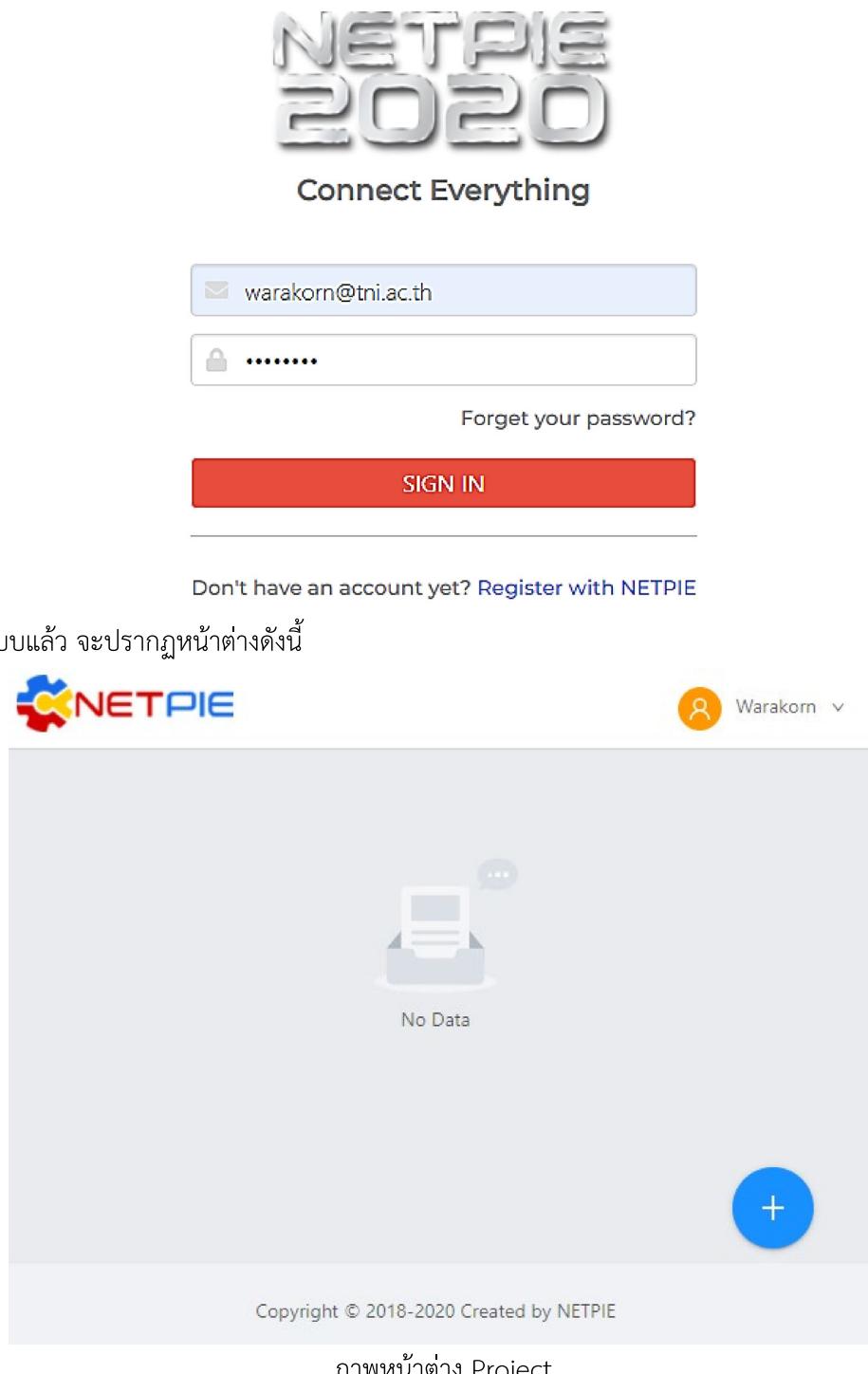
COUNTRY CODE

MOBILE PHONE NUMBER* (NO COUNTRY CODE)

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

SIGN UP

เมื่อสมัครเสร็จ ให้ SIGN IN



การสร้าง Project

สามารถสร้าง Project ได้โดยการกดที่ปุ่มบวกบริเวณมุมล่างซ้ายของหน้าต่าง



เมื่อกดแล้ว จะมีหน้าต่างให้กรอกข้อมูล Project โดยที่ * คือข้อมูลที่จำเป็นต้องกรอก ให้กรอกชื่อ Project “Test”

Create Project X

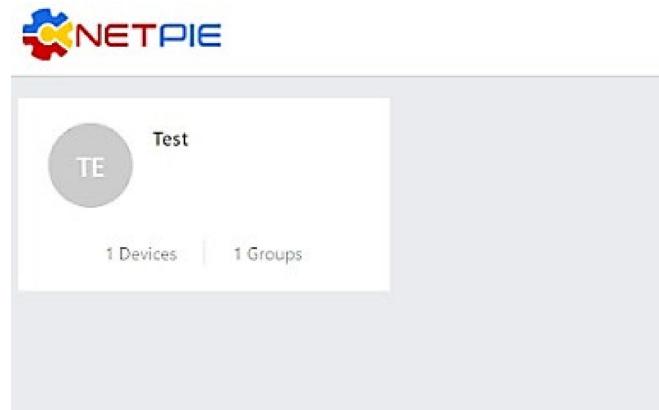
* Name:

Description:

Tag:

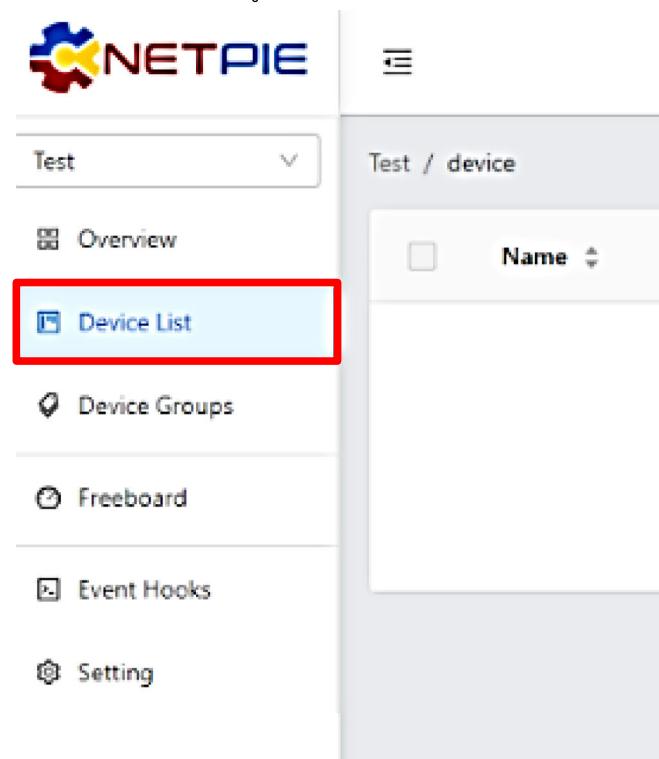
Cancel Create

เมื่อกรอกข้อมูลเสร็จแล้ว กดปุ่ม “Create” ผลลัพธ์ที่ได้จะเป็นดังนี้ สามารถกด Project เพื่อดำเนินการต่อไป



การสร้าง Device

การสร้าง Device สามารถทำได้โดยการเลือกเมนู “Device Lists” ทางด้านขวาเมื่อ



จากนั้นให้คลิกที่ปุ่ม “Create” เพื่อสร้าง Device ใหม่ โดยที่ * คือข้อมูลที่จำเป็นต้องกรอก จากนั้นกดปุ่ม “Create” เพื่อทำการสร้าง Device จากนั้นให้ระบุชื่อ Device “ESP8266” ดังภาพ

The screenshot shows the NETPIE web interface. On the left, there's a sidebar with options: Test (selected), Overview, Device List (highlighted in blue), Device Groups, Freeboard, Event Hooks, and Setting. The main content area is titled "Test / device" and shows a table with one row. The row has a checkbox, the header "Name" (sorted by "Tags"), and a "No Data" message with an envelope icon. In the top right of the main area, there's a "Create" button with a plus sign, which is also highlighted with a red box. The top right corner of the screen shows a user profile for "Warakorn". At the bottom of the main content area, it says "Copyright © 2018-2020 Created by NETPIE".

The screenshot shows a "Create" dialog box. At the top, it says "Create" and has an "X" button. Below that, there's a field labeled "* Name:" with the value "ESP8266". There's also a "Description:" field which is currently empty. Under "Tag:", there's a button labeled "+ New Tag". At the bottom of the dialog, there are two buttons: "Cancel" and "Create".

The screenshot shows the NETPIE platform's Device List interface. On the left, a sidebar menu includes options like Overview, Device List (which is selected and highlighted in blue), Device Groups, Freeboard, Event Hooks, and Setting. The main content area displays a table titled 'Test / device' with columns for Name, Tags, Group, and Create Date. A single row is present for the device named 'ESP8266'. The table includes standard filtering and sorting tools at the top, and a pagination control at the bottom indicating 1-1 of 1 items.

กดเข้าไปที่ Device รายละเอียดต่างๆจะปรากฏขึ้น รวมถึง Key, Token และ Secret ที่จะนำไปใช้เพื่อให้ Device สามารถเชื่อมต่อเข้ามายัง Platform ได้

The screenshot shows the NETPIE platform's Device Detail interface for the 'ESP8266' device. The left sidebar is identical to the previous screenshot. The main area is titled 'Test / device / ESP8266' and contains two tabs: 'Description' and 'Key'. The 'Key' tab is active, displaying three fields: Client ID (d5706e49-c340-4b4f-8670-5242b5f59a47), Token (ZFJYWWwEA8mGTmT14Wvhw14ynqPfitRr), and Secret (qQt1Shr0EeaNM(Vy5n7Fvlv4(CAQjglI)). Below these fields are 'Status' (Offline) and 'Enable' (on). At the bottom, there are 'Save' and 'Cancel' buttons, and a 'Tree' selection dropdown with the option '(empty object)'.

The screenshot shows the NETPIE Platform interface for managing devices. On the left, a sidebar menu is visible with the following items:

- Test (selected)
- Overview
- Device List
- Device Groups
- Freeboard
- Event Hooks
- Setting

The main content area displays the following information:

Description

MQTT Client ID
MQTT Username
MQTT Password

สถานะการเชื่อมต่อ Platform
 สถานะการเปิดไฟ้งานอุปกรณ์

Key

Client ID	: d5706e49-c340-4b4f-8670-5242b5f59a47	
Token	: ZFjYWwEAY8mGTmT14Wvhv14ynqPfitRr	
Secret	: qQt1Shr0EeaNM(Vy5n7Fvlv4(CAQjgl1	

Status : Offline

Enable :

ปุ่ม Resync Status

Below the status section are four tabs: Shadow, Schema, Trigger, and Feed. To the right are Save and Cancel buttons.

A large blue-highlighted section titled "Tree" contains the text "Select a node...". Below it is a tree view with one node listed:

- object {0}
- (empty object)

การเชื่อมต่อ Device กับ Platform

ในการเชื่อมต่ออุปกรณ์กับ Platform จะใช้ Key สำหรับการเชื่อมต่อของ Device mayoría Platform กรณีเชื่อมต่อผ่าน MQTT Protocol ให้เลือกใช้งาน MQTT Client Library ที่เหมาะสมหรือรองรับกับ Device ที่จะใช้ในการเชื่อมต่อ โดยการเชื่อมต่อของ MQTT จะต้องใช้ 4 Parameters คือ Host, Client id, Username และ Password โดยดูข้อมูลที่จะนำมาใช้ สามารถระบุค่าได้ดังนี้

Host	mqtt.netpie.io
Port	1883 (mqtt), 1884 (mqqtts)
Client ID	Client ID ของ Device ที่สร้างขึ้นใน NETPIE
Username	Token ของ Device ที่สร้างขึ้นใน NETPIE
Password	ยังไม่ต้องระบุ (ใช้สำหรับกรณีที่ต้องการตรวจสอบที่เพิ่มมากขึ้น)

ทดลองเชื่อมต่อ Platform ด้วย MQTT Box ซึ่งเป็นแอปพลิเคชันที่สามารถดาวน์โหลดได้จาก Microsoft Store

"MQTTBox" download

All Videos Images Maps News More Tools

About 8,860 results (0.33 seconds)

<https://chrome.google.com/webstore/detail/mqttbox> ::

MQTTBox
May 9, 2560 BE — Please check download link
<http://workswithweb.com/html/mqttbox/downloads.html> **MQTTBox** features: 1. MQTT Clients...

<https://www.microsoft.com/en-us/mqttbox> ::

Get MQTTBox - Microsoft Store

Nov 3, 2559 BE — Download this app from Microsoft Store for Windows 10. See screenshots, read the latest customer reviews, and compare ratings for **MQTTBox**.

microsoft.com/en-us/p/mqttbox/9nblggh55jzg?activetab=pivot:overviewtab

Microsoft | Home Devices Software More All Microsoft Search Cart

MQTTBox

workswithweb • [Developer tools > Utilities](#)

Developers helper program to create, develop and test MQTT connectivity protocol. MQTTBox enables you to create MQTT clients to publish or subscribe to topics, create MQTT virtual device, load test MQTT devices or brokers and much more.

[More](#)

Free

EVERYONE
Digital Purchases

Get

+ Offers in-app purchases
⚠ See System Requirements

เมื่อดาวน์โหลดเรียบร้อยให้เปิดโปรแกรมขึ้นมาดังรูป

MQTTBox

MQTTBox Edit Help

☰ Menu MQTT CLIENTS **Create MQTT Client** ⚡

No MQTT clients added. Click **Create MQTT Client** to add new MQTT client

หลังจากนั้นกดที่ปุ่ม “Create MQTT Client” เพื่อสร้างการเชื่อมต่อไปยัง MQTT Server (ในที่นี้คือ NETPIE Platform) :

MQTT CLIENT SETTINGS

MQTT Client Name	MQTT Client Id	Append timestamp to MQTT client id?	Broker is MQTT v3.1.1 compliant?
test device	a25caa14-23fd-40f0-94c1-50dfd16cb50b	<input type="checkbox"/> No	<input checked="" type="checkbox"/> Yes
Protocol	Host	Clean Session?	Auto connect on app launch?
mqtt / tcp	mqtt.netpie.io	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Yes
Username	Password	Reschedule Pings?	Queue outgoing QoS zero messages?
z2hwv6TG2nNbUXg91ejbvm8C5SApo57a	Password	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Yes
Reconnect Period (milliseconds)	Connect Timeout (milliseconds)	KeepAlive (seconds)	
1000	30000	10	
Will - Topic	Will - QoS	Will - Retain	Will - Payload
Will - Topic	0 - Almost Once	<input type="checkbox"/> No	

Save **Delete**

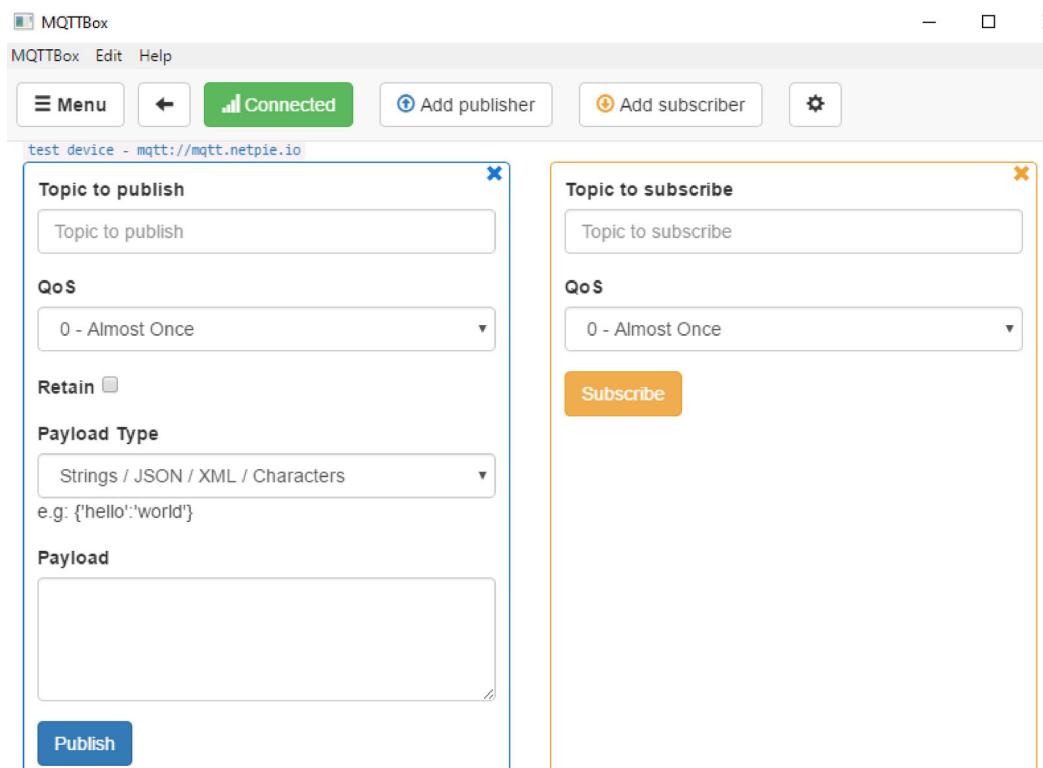
MQTT CLIENT SETTINGS

MQTT Client Name	MQTT Client Id
test device	a25caa14-23fd-40f0-94c1-50dfd16cb50b
Protocol	Host
mqtt / tcp	mqtt.netpie.io
Username	Password
z2hwv6TG2nNbUXg91ejbvm8C5SApo57a	Password
Reconnect Period (milliseconds)	Connect Timeout (milliseconds)
1000	30000
Will - Topic	Will - QoS
Will - Topic	0 - Almost Once

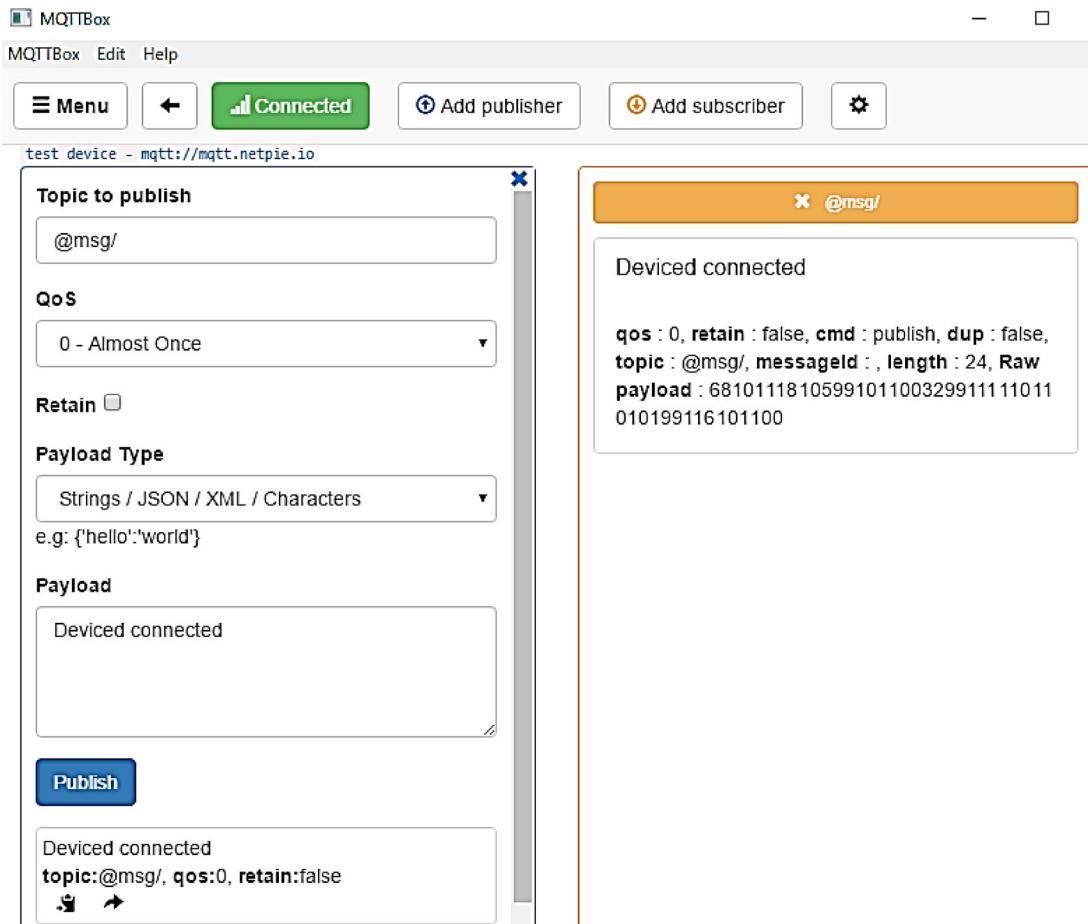
Save

Append timestamp to MQTT client id?	Broker is MQTT v3.1.1 compliant?
<input type="checkbox"/> No	<input checked="" type="checkbox"/> Yes
Clean Session?	Auto connect on app launch?
<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Yes
Reschedule Pings?	Queue outgoing QoS zero messages?
<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Yes
KeepAlive (seconds)	
<input type="text" value="10"/>	
Will - Retain	Will - Payload
<input type="checkbox"/> No	<input type="text"/>
Delete	

ระบบค่าการเชื่อมต่อให้ถูกต้อง จากนั้นกดปุ่ม “Save” และ MQTT Box จะทำการเชื่อมต่อ Platform ให้อัตโนมัติ ถ้าเชื่อมต่อได้สำเร็จจะเป็นดังนี้



ทดสอบว่าสามารถเชื่อมต่อ Platform ได้จริงโดย Publish เข้าหาตัวเอง การเช็คค่า Topic ที่จะ Publish/Subcribe ให้ขึ้น Topic ด้วย @msg/ (คลิกปุ่ม “Subscribe” ก่อนที่จะคลิกปุ่ม “Publish”)



ข้อควรระวัง

การตั้งค่าเชื่อมต่อผ่าน MQTT Box ต้องเลือก Append timestamp to MQTT client id ให้เป็น No เพื่อไม่ให้ MQTT Box นำ Timestamp มาต่อท้าย MQTT Client Id ซึ่งจะทำให้ Key ที่ใช้เชื่อมต่อไม่ถูกต้อง และไม่สามารถเชื่อมต่อ Platform ได้

การสื่อสารระหว่าง Device

Device ที่จะสามารถสื่อสารกันได้ต้องอยู่ภายใต้ Device Group เดียวกัน โดยเริ่มจากไปที่เมนู “Device Groups”

The screenshot shows the NETPIE web interface. On the left, a sidebar menu has 'Device Groups' selected. The main area displays a table titled 'Test / group' with columns 'Name', 'Tags', and 'Devices'. A single row is present, showing a folder icon and the text 'No Data'. At the top right, there is a user profile icon for 'Warakorn'.

ให้ทำการสร้าง Group โดยคลิกที่ปุ่ม “Create” และกรอกข้อมูล

Create

* Name:

Description:

Tag:

The screenshot shows the NETPIE web interface under the 'Test' tab. On the left sidebar, 'Device Groups' is selected. The main area displays a table titled 'Test / group' with one item: 'Group_Test_01'. The table has columns for Name, Tags, and Devices. It shows 0 Online and 0 Offline devices. Navigation buttons at the bottom indicate 1-1 of 1 items, page 1, and 10 / page.

Name	Tags	Devices
Group_Test_01	-	0 Online 0 Offline

กลับไปที่ “Device Lists” และทำการสร้าง Device ใหม่อีก 1 ตัวเพื่อใช้ในการสื่อสารระหว่าง Device กับ Device

The screenshot shows the NETPIE web interface under the 'Test' tab. On the left sidebar, 'Device List' is selected. The main area displays a table titled 'Test / device' with two items: 'NodeMCU8266' and 'ESP8266'. The table has columns for Name, Tags, and Group. The 'NodeMCU8266' row has a grey circular icon next to it, while the 'ESP8266' row has a green circular icon. Navigation buttons at the bottom indicate 1-2 of 2 items, page 1, and 10 / page.

<input type="checkbox"/>	Name	Tags	Group
<input type="checkbox"/>	NodeMCU8266	-	-
<input type="checkbox"/>	ESP8266	-	-

จัด Device ทั้ง 2 ตัวเข้า Group ที่สร้างไว้

The screenshot shows the NETPIE web interface under the 'Test' project. On the left sidebar, 'Device List' is selected. In the main area, two devices are listed: 'NodeMCU8266' and 'ESP8266'. Both devices have a blue checkmark icon next to them, indicating they are selected. A red box highlights the 'Move' button in the top right corner of the list area. The 'Group' column header is also highlighted with a red box.

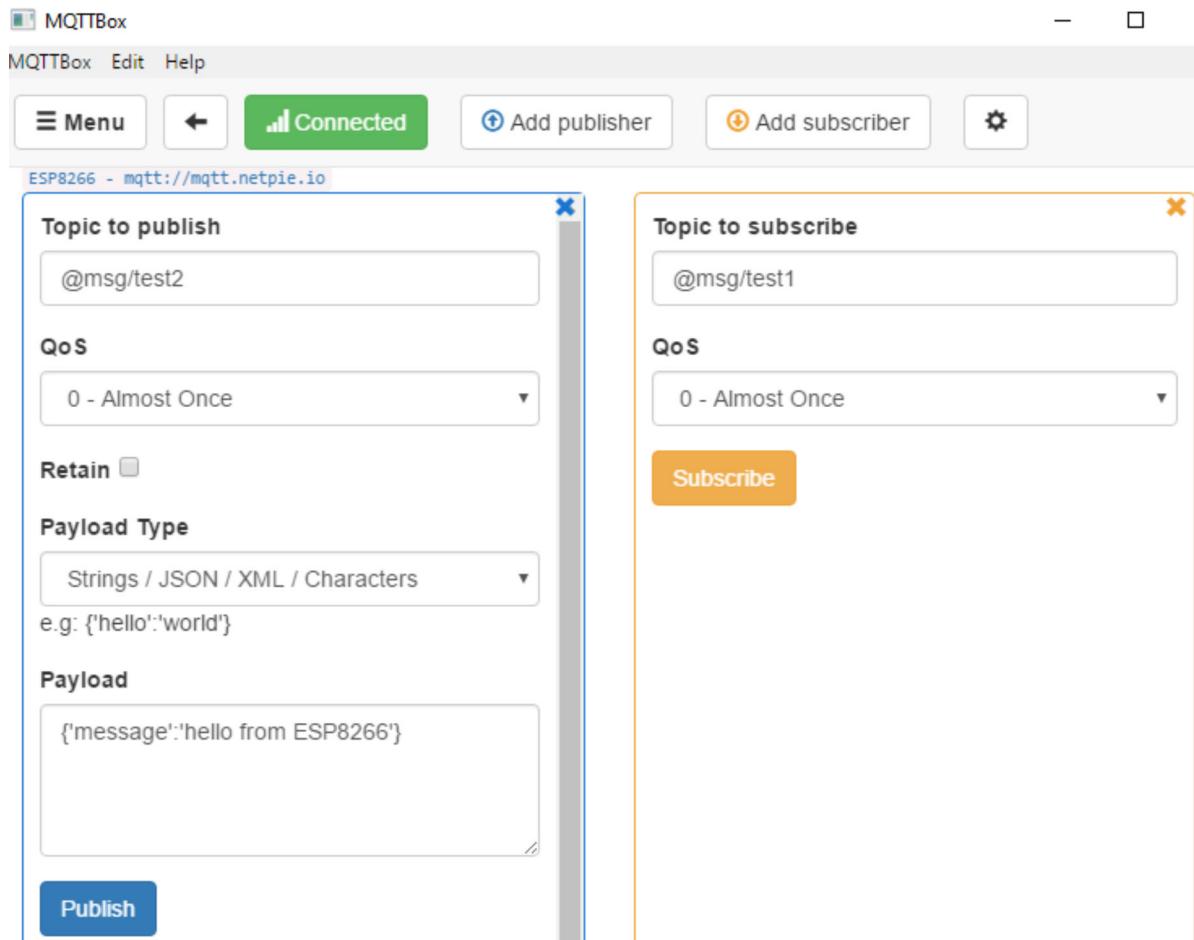
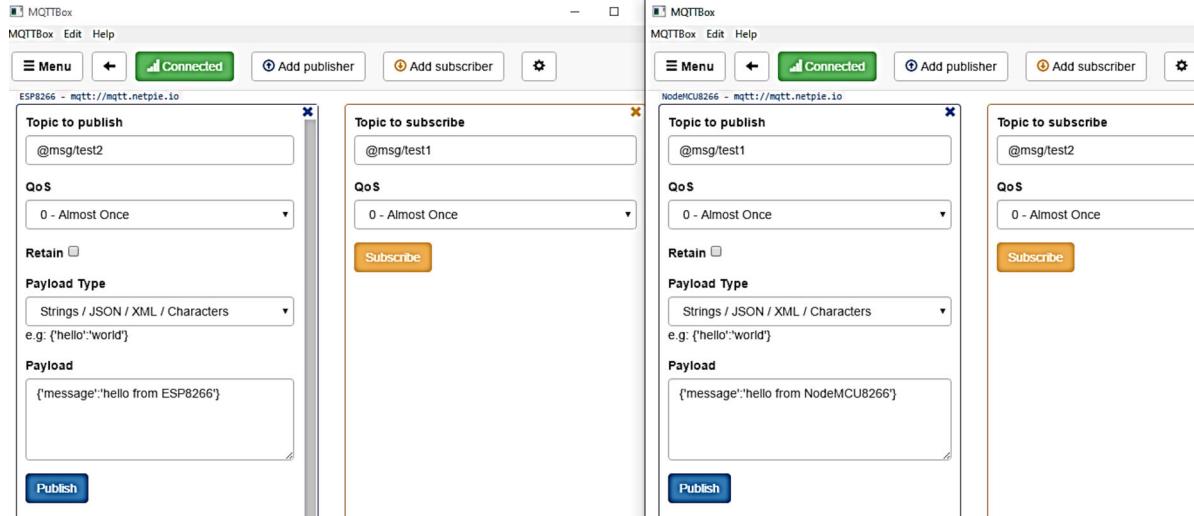
	Name	Tags	Group
<input checked="" type="checkbox"/>	NodeMCU8266	-	-
<input checked="" type="checkbox"/>	ESP8266	-	-

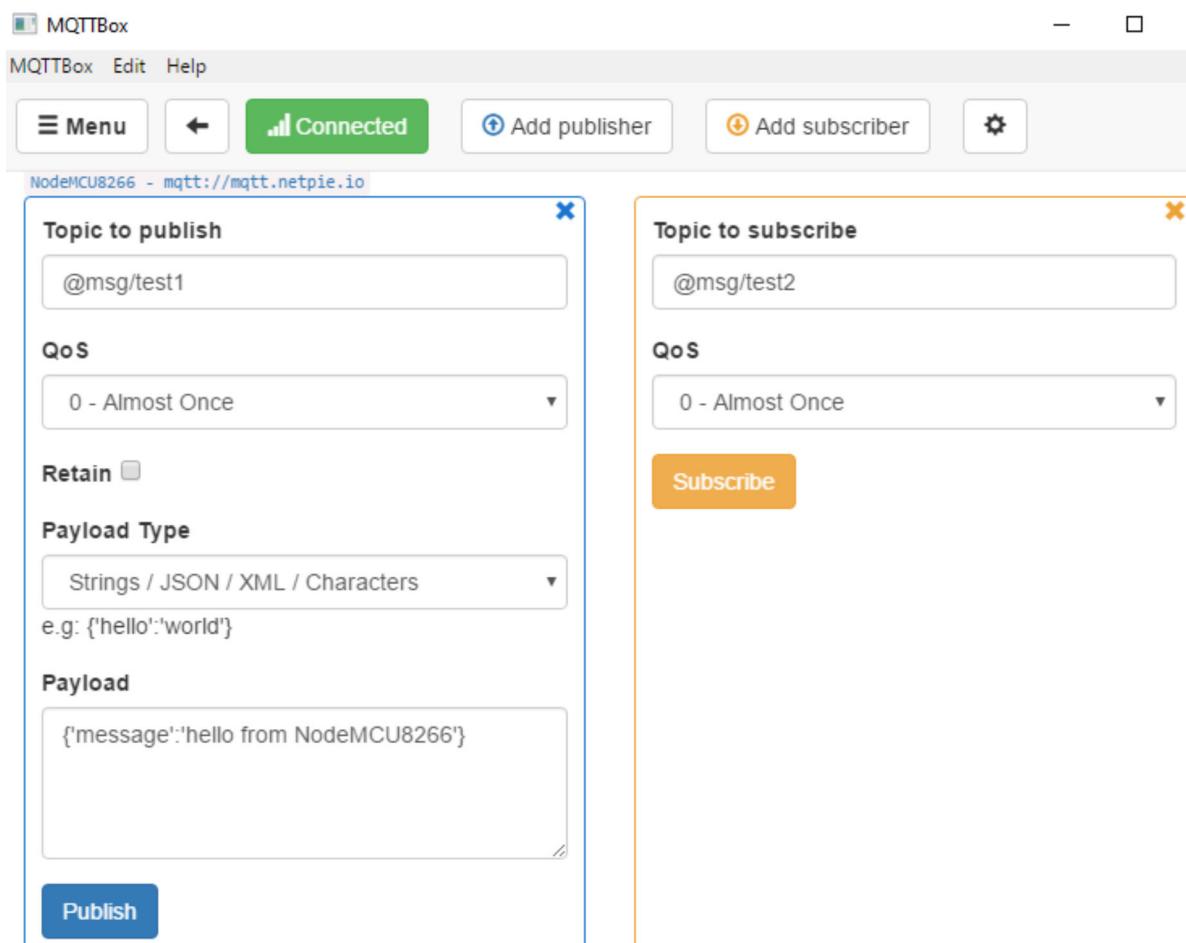
เมื่อนำ Device เข้า Group แล้ว ในแต่ละ Device จะมี Group ปรากฏขึ้น

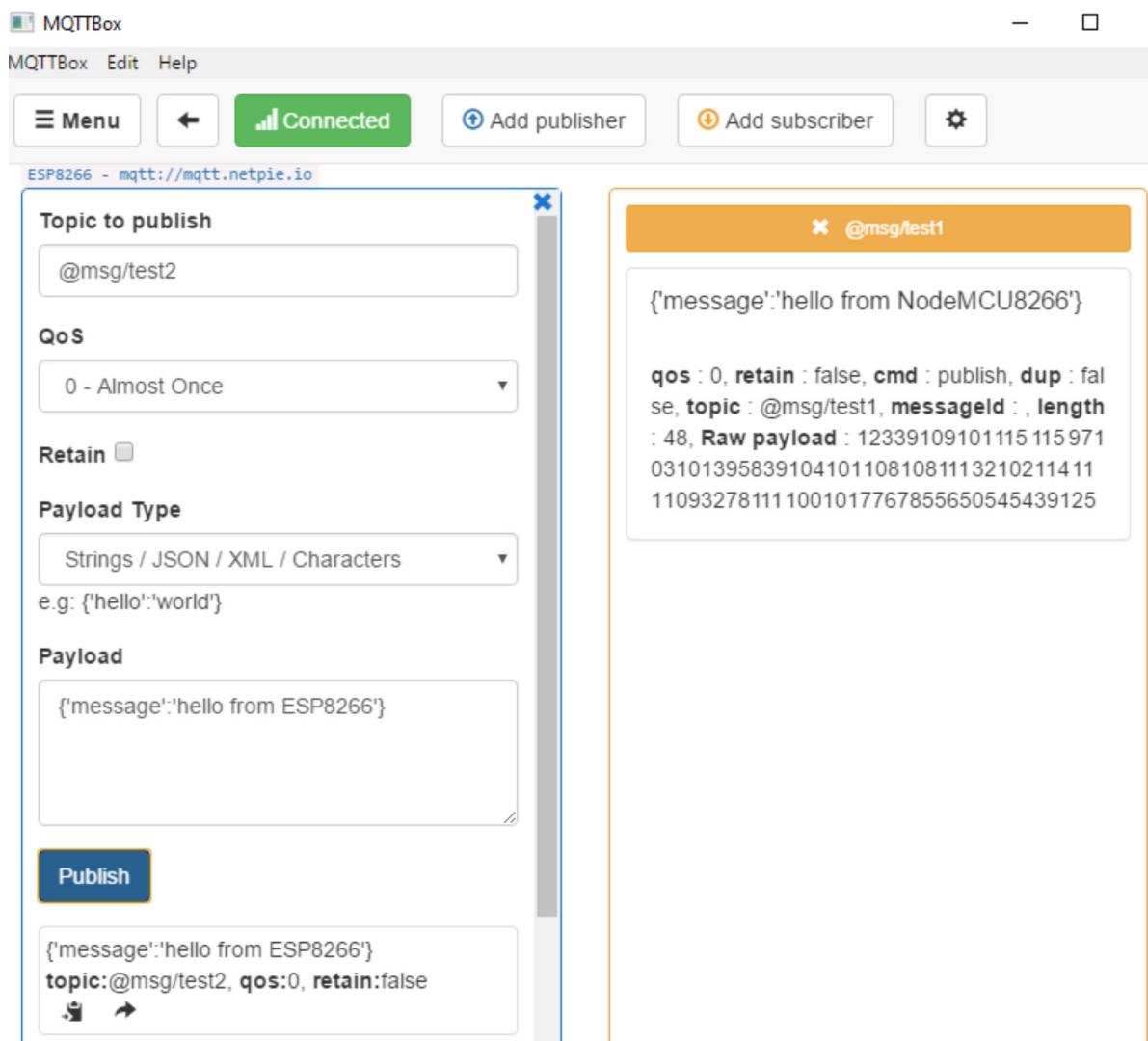
The screenshot shows the same NETPIE interface after the devices have been moved into a group. The 'Group' column now displays the name 'Group_Test_01' for both devices. A red box highlights the 'Group' column header. The 'Move' button is no longer highlighted.

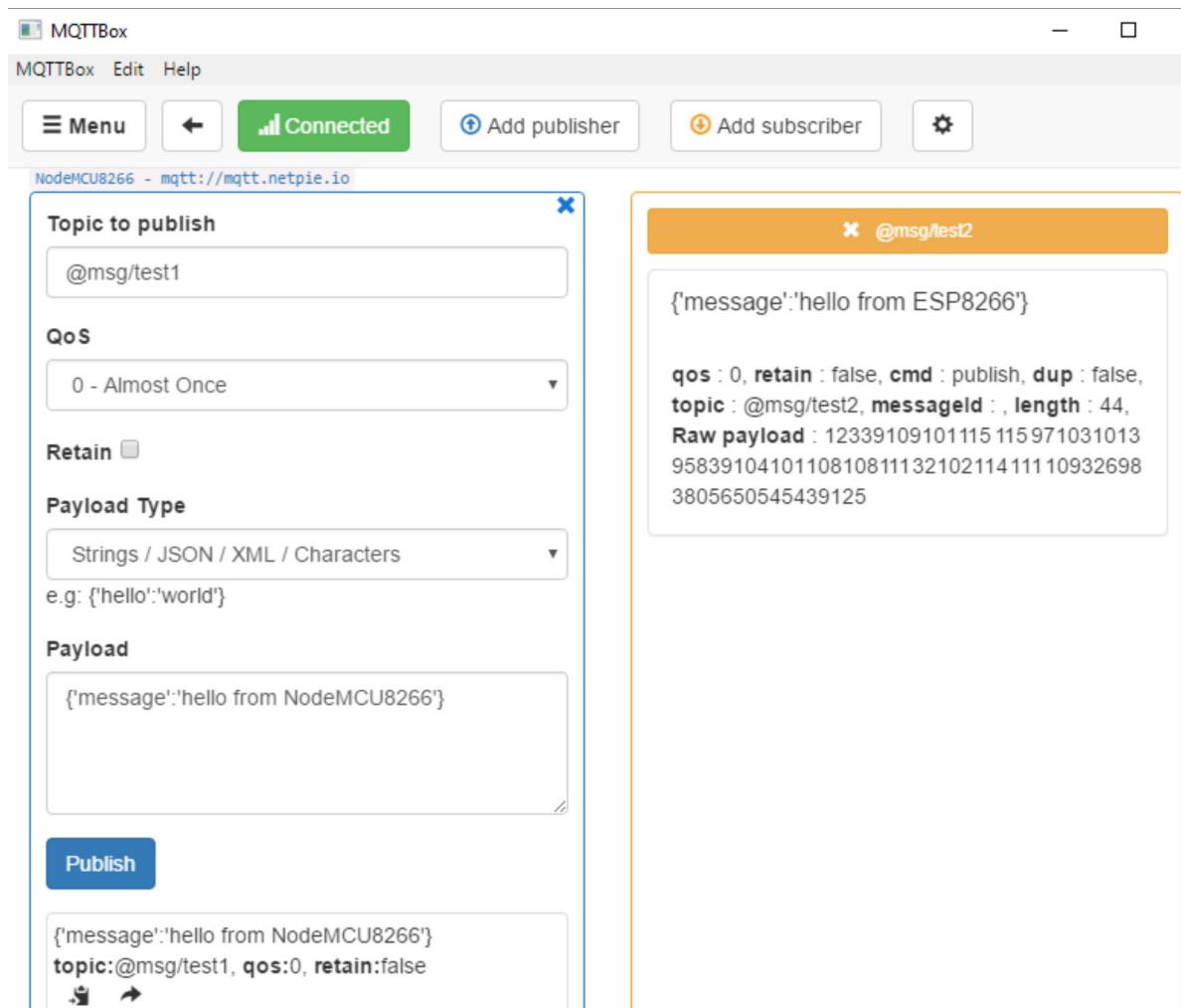
	Name	Tags	Group
<input type="checkbox"/>	NodeMCU8266	-	Group_Test_01
<input type="checkbox"/>	ESP8266	-	Group_Test_01

ทดลองการสื่อสารโดยการเปิดหน้าต่าง MQTT Box ใหม่ขึ้นมา โดยที่อีกหน้าต่างหนึ่งตั้งค่าการเชื่อมต่อตาม Device ใหม่ที่สร้างขึ้น จากนั้นจึงรีเมสื่อสารกันโดยการกรอก topic และ payload ดังรูป









การตั้งค่า Device

Device Shadow

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

Device Schema

นิยามของ Device Schema ในที่นี่ คือ แผงผังข้อมูลที่กำหนดไว้เพื่อใช้กับ Device Shadow สำหรับ Device ที่ต้องมีการจัดการข้อมูล แนะนำให้สร้าง Device Schema ของข้อมูลเตรียมไว้ Device Schema เสมือนเป็น Device Template ทำให้ Server สามารถ

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

โดย Device Schema จะประมวลผลในรูปแบบ JSON มีลักษณะดังนี้

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

Device Schema จะประกอบด้วย

additionalProperties (boolean)

- สถานะการอนุญาตให้บันทึกข้อมูลลง Shadow หรือ Timeseries Database ในกรณีที่ข้อมูลไม่ตรงตามที่กำหนดใน Properties
- additionalProperties : true
 - อนุญาตให้บันทึกลง Shadow หรือ Timeseries Database
- additionalProperties : false
 - ไม่อนุญาตให้บันทึกเฉพาะส่วนที่ไม่ตรงตาม Properties
- ตัวอย่าง
 - Schema มีการกำหนด Properties เป็น temp, humid ข้อมูลที่ส่งมาเป็น temp, humid และ color ถ้าหาก additionalProperties เป็น true ข้อมูลของ color จะถูกบันทึกลงไปใน shadow หรือ feed แต่หากเป็น false จะมีเพียง humid และ temp เท่านั้นที่จะถูกบันทึกลง shadow หรือ Timeseries Database

properties(json)

เริ่มจากกำหนดชื่อฟิลด์ (จากตัวอย่าง คือ “temp” และ “place”) และกำหนดคุณสมบัติของแต่ละฟิลด์ซึ่งจะอยู่ในรูปแบบ JSON โดยจะแยก 2 ส่วน คือ

- operation สำหรับตั้งค่าการจัดการข้อมูลในฟิลด์นั้น ๆ ประกอบด้วย
 - store สำหรับตั้งค่าการเก็บข้อมูลลง Timeseries Database
 - ttl คือ ระยะเวลาของการเก็บข้อมูลใน Timeseries Database แต่ละจุดข้อมูลที่มีอายุการเก็บครบตามที่กำหนดจะถูกลบทิ้งอัตโนมัติ จำเป็นต้องกำหนดค่านี้ ระบบถึงจะเก็บข้อมูลลง Timeseries Database การกำหนดค่าจะระบุตัวเลขจำนวนเต็มตามด้วยหน่วยเวลา ดังนี้ ms(มิลลิวินาที), s(วินาที), m(นาที) h(ชั่วโมง), d(วัน), y(ปี) ถ้าไม่ระบุหน่วยค่า default จะเป็น ms(มิลลิวินาที) เช่น 30d หมายถึง เก็บข้อมูลนาน 30 วัน, 1y หมายถึง เก็บข้อมูลนาน 1 ปี, 3000 หมายถึง เก็บข้อมูลนาน 3 วินาที
 - transform การแปลงข้อมูล (Data Transformation) ก่อนการจัดเก็บ
 - expression คือ สูตรหรือวิธีการแปลงข้อมูล (Data Transformation) ก่อนการจัดเก็บ เช่น กำหนด expression เท่ากับ (\$.temp * 1.8) + 32 เป็นการแปลงหน่วยอุณหภูมิค่าที่เซนเซอร์วัดได้จากหน่วยเซลเซียสเป็นฟาเรนไฮต์ โดยนำมารูด้วย 1.8 และบวกด้วย 32 จะได้ค่าอุณหภูมิเป็นหน่วยฟาเรนไฮต์ ก่อนบันทึกลงใน Device Shadow หรือ Timeseries Database

- **type** คือ ชนิดของข้อมูลในพิล์ดนั้น ๆ ได้แก่ number, string, boolean, array, object

สามารถอ่านข้อมูลเพิ่มเติมเกี่ยวกับ Device Schema ได้ที่
<https://docs.netpie.io/device-config.html#device-schema>

MQTT

การเชื่อมต่อ Platform ผ่าน MQTT (Message Queuing Telemetry Transport) ซึ่งเป็น Protocol ที่มีขนาดเล็กและได้รับความนิยมสำหรับการสื่อสารแบบ M2M (Machine to Machine) โดยสามารถใช้ MQTT Library ตัวใดก็ได้ที่รองรับกับ Device ที่ใช้งานอยู่ การเชื่อมต่อของ MQTT จะต้องใช้ 4 Parameters คือ Host, Client ID, Username และ Password โดยให้ระบุแต่ละค่าดังนี้

Host	mqtt.netpie.io
Port	1883 (mqtt), 1884 (mqtts)
Client ID	Client ID ของ Device ที่สร้างขึ้นใน NETPIE
Username	Token ของ Device ที่สร้างขึ้นใน NETPIE
Password	ยังไม่ต้องระบุ (ใช้สำหรับกรณีที่ต้องการตรวจสอบที่เพิ่มมากขึ้น)

MQTT API จะมีลักษณะการใช้งานเป็นแบบ Publish / Subscribe โดย Publish จะเป็นการส่งข้อมูลไปยัง Topic ที่ต้องการ ส่วน Subscribe จะเป็นการรับข้อมูลใน Topic ที่ต้องการ การสั่ง Subscribe Topic ได้ก็ตามทำเพียงครั้งเดียว ก็จะได้รับข้อมูลใน Topic นั้นไปตลอดจนกว่าจะสั่ง Unsubscribe Topic นั้น หรือการเชื่อมต่อกับ Platform หยุดลง

องค์ประกอบสำคัญที่จำเป็นต้องทราบสำหรับการใช้งาน MQTT API คือ Topic เพราะ Publish / Subscribe จำเป็นต้องระบุ Topic ที่ต้องการ Topic จะหน้าที่เหมือน Endpoint บน MQTT Broker เพื่อให้ MQTT Client มาเชื่อมต่อและสื่อสารกัน โดยจะรองรับคุณสมบัติดังต่อไปนี้

- **QoS (Quality of Service) 3 ระดับ** คือ ข้อตกลงระหว่างผู้ส่ง (Publisher) และผู้รับ (MQTT Broker) ในการรับประกันการส่งข้อมูล
- **QoS Level 0 :** การส่งข้อมูลที่มีการรับประกันผลในระดับต่ำสุด หรือเป็นข้อมูลที่ต้องการความรวดเร็วในการส่ง คือ ไม่ต้องการการตอบกลับว่าข้อมูลถึง MQTT Broker เล้า

- **QoS Level 1 :** การส่งข้อมูลที่มีการรับประกันผลในระดับที่ทุกครั้งของการส่งข้อมูล ต้องมีการตอบกลับเสมอเพื่อยืนยันว่าข้อมูลได้ส่งไปถึง MQTT Broker แล้ว ถ้าการตอบกลับสูญหาย ผู้ส่งจะทำการส่งข้อมูลไปใหม่จนกว่าจะได้รับการตอบกลับ นั่นหมายความว่า MQTT Broker จะได้รับข้อมูลอย่างน้อย 1 ครั้ง แน่นอน แต่ข้อมูลเดียวกันอาจจะได้รับมากกว่า 1 ครั้งด้วยเช่นกัน
- **QoS Level 2 :** การส่งข้อมูลที่มีการรับประกันผลระดับสูงสุด ข้อมูลมีความสำคัญมาก คือ ทุกครั้งของการส่งข้อมูลจะมีการยืนยันว่าถูกส่งไปถึง MQTT Broker อย่างแน่นอนและได้รับข้อมูลครั้งเดียว
- **Shared Subscription** คือ การส่งข้อมูลกระจายไปได้ทุกหนึ่งที่ Subscribe Topic เดียวกัน
- **Transparent Virtual Host** คือ การ Publish / Subscribe ไปที่ Topic เดียวกัน ถ้า Device อยู่ต่างกลุ่ม ก็จะเหมือนเป็นคนละ Topic กัน

โครงสร้าง Topic ใน NETPIE Platform สามารถแยกได้เป็น 3 ประเภท ดังนี้

Message API Topic

เป็นการกำหนด Topic สำหรับ Publish / Subscribe ข้อมูลเพื่อสื่อสารระหว่าง Devices ที่อยู่ภายใต้ Group เดียวกัน รูปแบบการใช้งานให้เขียนต้น Topic ด้วย @msg ตามด้วยโครงสร้าง Topic ที่ต้องการ ดังนี้

publish	publish @msg/{any}/{topic}
subscribe	subscribe @msg/{any}/{topic}
ตัวอย่าง Topic	@msg/myhome/bedroom/lamp @msg/sensor/temp @msg/john

Shadow API Topic

เป็น Topic ที่เกี่ยวข้องกับการจัดการ Device Shadow สามารถแยกได้เป็น Publish และ Subscribe โดย Publish จะใช้กรณีที่ต้องการขอข้อมูลหรืออัพเดทข้อมูลใน Device Shadow ส่วน Subscribe จะเป็นการรับข้อมูลในกรณีที่มีการ Publish ไปขอข้อมูล หรือในกรณีที่มีการเปลี่ยนข้อมูล Device Shadow และได้ทำการ Subscribe ไว้ ซึ่งการใช้งานจะมีรายละเอียด ดังนี้

- **Private Channel Topic** คือ ช่องทางพิเศษสำหรับการตอบกลับ (Response) หรือรับข้อมูลทุกอย่างที่เกิดขึ้นกับตัวเอง เช่น Device Shadow ตัวเองมีการเปลี่ยนแปลง เป็นต้น โดยรูปแบบการใช้งานให้เขียนต้น Topic ด้วย @private มีลักษณะตามนี้ @private/{response topic} จะมีเพียงการ Subscribe เท่านั้น Response Topic สำหรับ Subscribe @private มีดังนี้

Subscribe Topic	คำอธิบาย
@private/#	การรับทุกข้อมูลที่ Publish มาอย่าง Topic ที่ขึ้นต้นด้วย @private/ รวมถึงข่าสารต่างๆ ที่ Platform ต้องการ แจ้งให้ทราบก็จะถูก Publish มาอย่าง Topic นี้
@private/shadow/data/get/response	การรับ Device Shadow Data เมื่อมีการร้องขอ ข้อมูลไป
@private/shadow/batch/update/response	การรับข้อมูลความต้องการ กรณีอัปเดท Shadow แบบ เป็นชุด (Shadow Batch Update)

- Shadow Topic คือ Topic ที่ใช้สำหรับจัดการ Device Shadow ของตัวเอง Topic ที่เกี่ยวข้องมีดังนี้

Subscribe Topic	คำอธิบาย
@shadow/data/get	เป็นการร้องขอข้อมูล Shadow Data ของตัวเองแบบ ทั้งหมด โดยการรับข้อมูลให้ Subscribe Topic @private/# หรือ @private/shadow/data/get/response เพื่อรับ ข้อมูล (ใช้ในกรณีที่เป็น Shadow ตัวเองท่านนั้น)
@shadow/data/update	เป็นการอัปเดตค่าใน Shadow Data โดยส่ง Payload ดังนี้ { "data":{ "field name 1": value 1, "field name 2": value 2, ..., "field name n": value n }} ถ้าต้องการได้รับข้อมูลเมื่อค่าต่าง ๆ ใน Shadow Data ถูกอัปเดตให้ Subscribe Topic @shadow/data/updated รอไว้
@shadow/batch/update	เป็นการอัปเดตค่าใน Shadow แบบเป็นชุดข้อมูล (Shadow Batch Update)

สามารถอ่านข้อมูลเพิ่มเติมเกี่ยวกับ MQTT ของ NETPIE ได้ที่ <https://docs.netpie.io/mqtt-api.html>

Dashboard

ใช้สำหรับนำข้อมูลที่เก็บอยู่ใน Platform มาแสดงผลในรูปแบบต่างๆ เป็นเหมือนช่องทางให้ผู้ใช้สามารถติดตามหรือควบคุมการทำงานของ Device ของตัวเอง โดย Dashboard ที่มีให้ใช้งาน ณ ปัจจุบันมีเพียงประเภทเดียว คือ Freeboard ซึ่งสามารถเข้าใช้งานได้ด้วยการคลิกที่เมนู “Freeboard” ในแท็บข้างมือ ก็จะปรากฏหน้าจอแสดงรายละเอียด Freeboard ทั้งหมดที่เคยสร้างไว้ภายใน Project นั้นๆ (ถ้ามี) ดังรูป

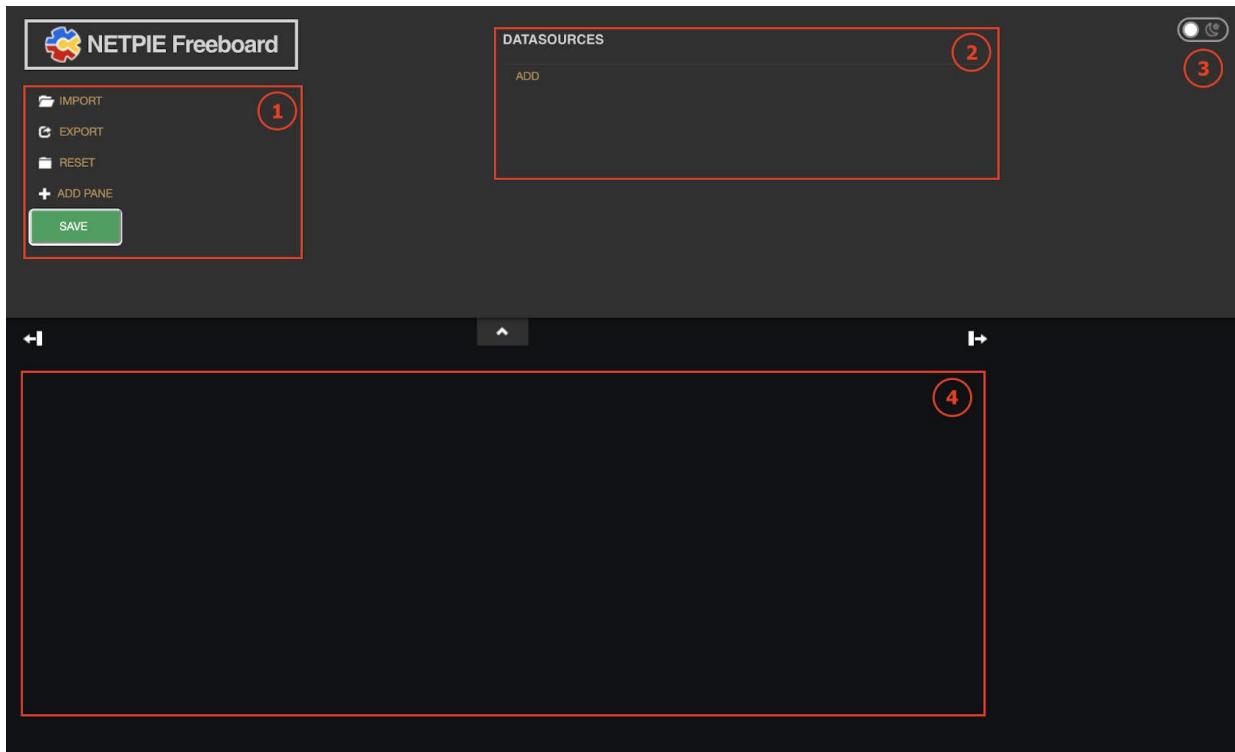
จากรูปด้านบน หลังรายชื่อ Freeboard แต่ละรายการ ถ้านำเมาส์ไปวางไว้เหนือรายการใด จะปรากฏปุ่ม “Edit” และ “Delete” สำหรับทำการแก้ไขข้อมูลที่ว่าไปและลบ Freeboard นั้นๆ ตามลำดับ ถ้าต้องการสร้าง Freeboard ใหม่ให้คลิกที่ปุ่ม “Create” มุ่งหมายว่ามีของหน้าจอ ก็จะปรากฏหน้าจอสำหรับให้กรอกข้อมูลที่ว่าไปของ Freeboard ดังรูป

Create Dashboard

* Name:

Description:

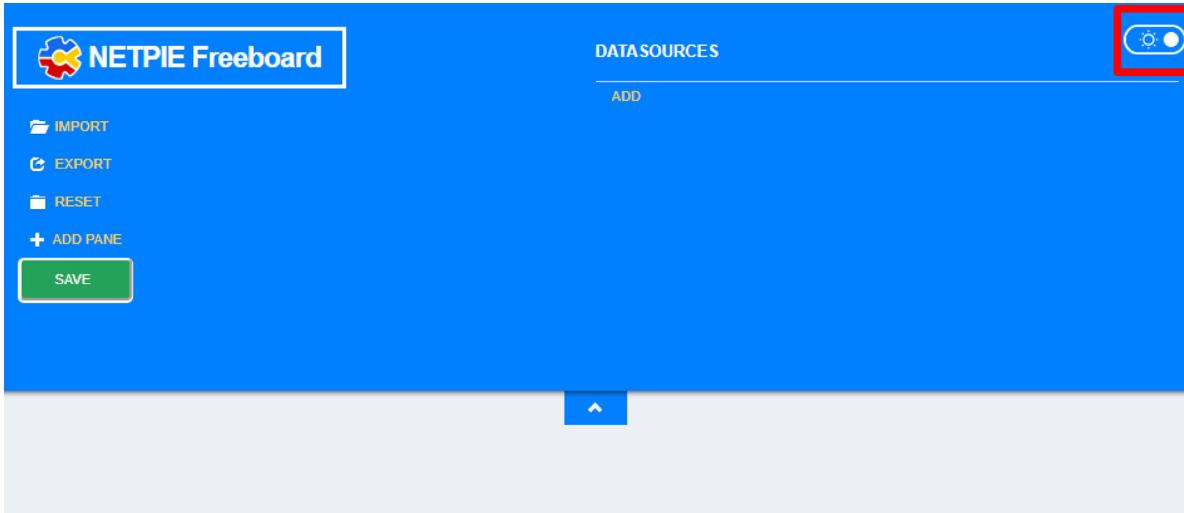
สำหรับการเข้าไปเช็ต Configuration ของ Freeboard ให้คลิกที่รายการที่ต้องการเข้าไปดำเนินการ โดยหน้าจอสำหรับจัดการ Configuration ของ Freeboard มีรายละเอียดดังนี้



จากรูปด้านบน หน้าจอสำหรับจัดการ Freeboard แบ่งเป็น 4 ส่วนใหญ่ๆ ดังนี้

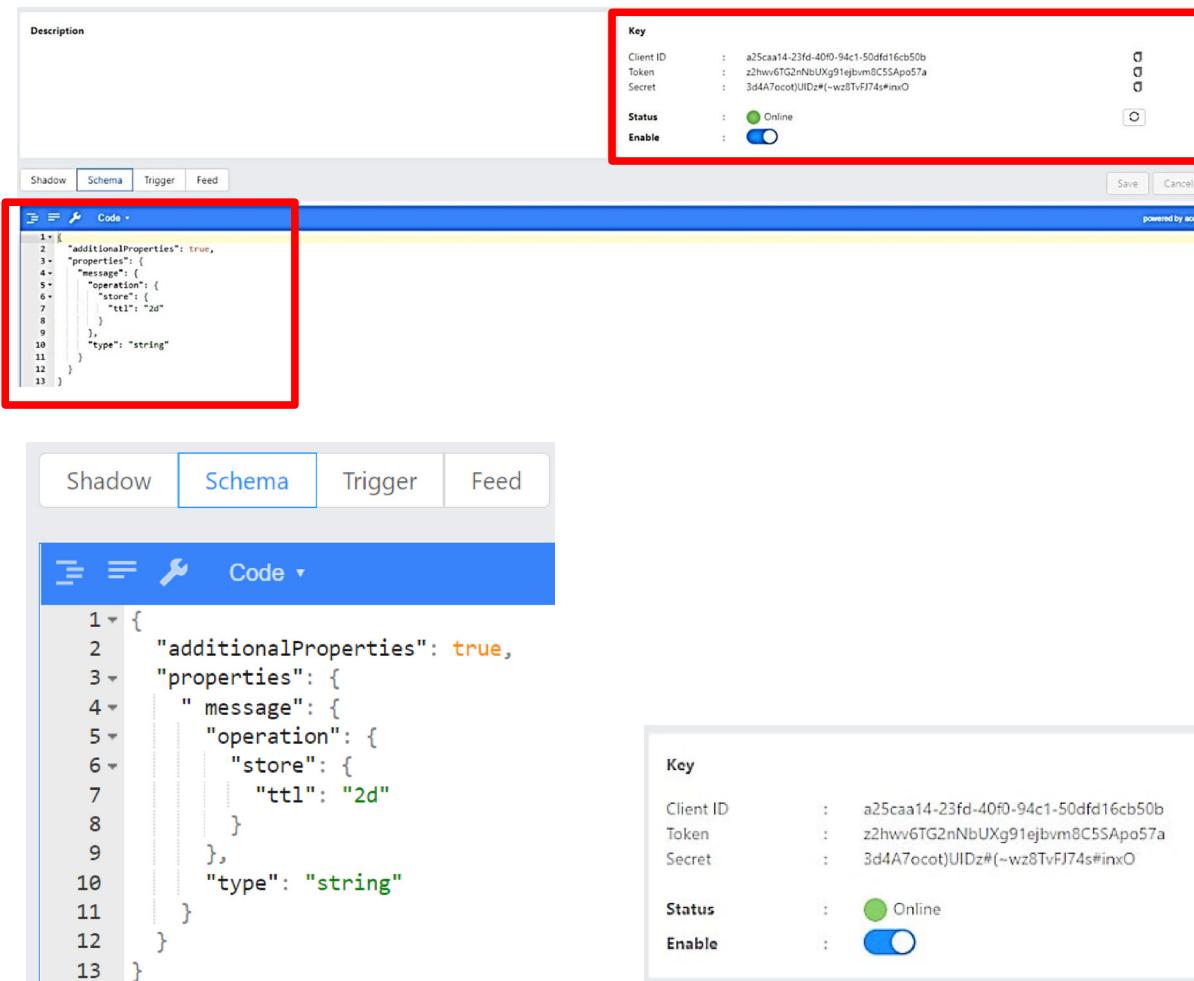
1. เมนูสำหรับจัดการส่วนต่างๆ ของ Freeboard ได้แก่
 - a. เมนู “IMPORT” ใช้สำหรับนำเข้า Configuration Code ซึ่งจะอยู่ในรูปแบบ JSON File ที่ส่งออกไปจาก Freeboard ที่เคยเช็ต Configuration ไว้แล้ว เมื่อคลิกแล้วจะปรากฏ browse file ให้เลือกไฟล์ที่ต้องการนำเข้า

- b. เมนู “EXPORT” ใช้สำหรับส่งออก Configuration Code ที่เคยเซ็ตไว้แล้ว ซึ่งจะอยู่ในรูปแบบ JSON File เช่นกัน เพื่อนำเข้า (เมนู “IMPORT”) กลับไป Freeboard อีก หรือเพื่อการสำรองข้อมูล (Backup)
 - c. เมนู “RESET” ใช้สำหรับล้างค่า Configuration Code ที่เคยเซ็ตไว้แล้วทั้งหมด
 - d. เมนู “ADD PANE” ใช้สำหรับสร้าง Block หรือ Panel ที่ใช้จัดกลุ่มการแสดงผลแต่ละ Widget ที่จะสร้างใน Freeboard
 - e. ปุ่ม “SAVE (สีเขียว)” ใช้สำหรับบันทึกการเปลี่ยนแปลงทุกอย่างที่มีการดำเนินการไป คลิกปุ่มนี้ทุกครั้งก่อนออกจากหรือปิดหน้าจอเพื่อบันทึก Configuration ต่างๆ ที่ได้เปลี่ยนแปลงไป
2. ส่วนจัดการ “DATASOURCES” ใช้สำหรับสร้างการเชื่อมต่อเพื่อดึงข้อมูลจาก Device ต่างๆ ใน Platform มาแสดงที่ Freeboard รายละเอียดจะอธิบายเพิ่มเติมในหัวข้อถัดไป
3. Theme ใช้สำหรับเปลี่ยนรูปสีของ Freeboard มีให้เลือก 2 โทนสี คือ Dark Color (ค่า Default) และ Light Color
4. ส่วนจัดการและแสดงผล Freeboard ใช้สำหรับจัดการ Widget ต่างๆ ที่จะนำค่ามาแสดง และยังเป็นส่วนที่ใช้ดู Freeboard (View) ที่เซ็ตค่าไว้แล้ว
- เมื่อกดเปลี่ยน Theme จะได้รูปสีของ Freeboard ที่แตกต่างไป



ทดลองใช้ Freeboard กับ Device Shadow

ในส่วนนี้ จะทำการทดลองการใช้ Freeboard โดยการส่งข้อมูลข้อความขึ้น Device Shadow จากนั้นจึงแสดงผลข้อมูลข้อความนั้นใน Freeboard โดยเริ่มจากการสร้าง Device Schema ดังนี้



The screenshot shows the MQTTBox Device Management interface. At the top, there's a 'Description' section with tabs for 'Shadow', 'Schema', 'Trigger', and 'Feed'. The 'Schema' tab is selected. Below it is a code editor window titled 'Code' containing the following JSON schema:

```

1< {
2   "additionalProperties": true,
3   "properties": {
4     "message": {
5       "operation": {
6         "store": {
7           "ttl": "2d"
8         }
9       },
10      "type": "string"
11    }
12  }
13 }

```

To the right of the code editor is a 'Key' configuration panel with the following settings:

Key	
Client ID	: a25caa14-23fd-40f0-94c1-50dfd16cb50b
Token	: z2hwv6TG2nNbUXg91ejbvm8C55Apo57a
Secret	: 3d4A7ecot)UIDz#(-wz0TvFJ74s#inxO
Status	: Online
Enable	: <input checked="" type="checkbox"/>

Below the 'Schema' tab, there's another code editor window with the same schema, and the 'Key' configuration panel is also present.

เมื่อตั้งค่า Device Schema เสร็จแล้ว ต่อไปให้ตั้งค่า Device ใน MQTTBox เพื่อให้พร้อมสำหรับส่งข้อมูลข้อความขึ้น Device Shadow โดย Topic และ Payload ที่กรอกสามารถรอกได้ดังรูป

ESP8266 - mqtt://mqqt.netpie.io

Topic to publish

@shadow/data/update

QoS

0 - Almost Once

Retain

Payload Type

Strings / JSON / XML / Characters

e.g: {'hello': 'world'}

Payload

```
{"data": {"message": "Hello from ESP8266"}}
```

Publish

Connected

Add publisher

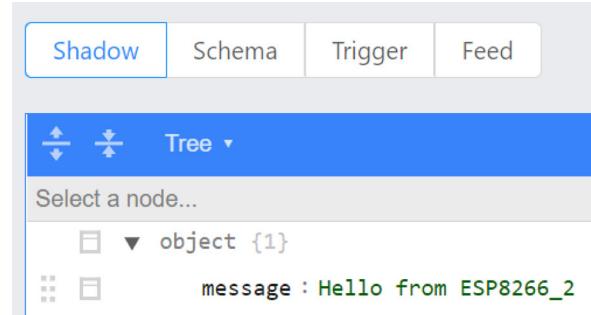
Add subscriber

Menu

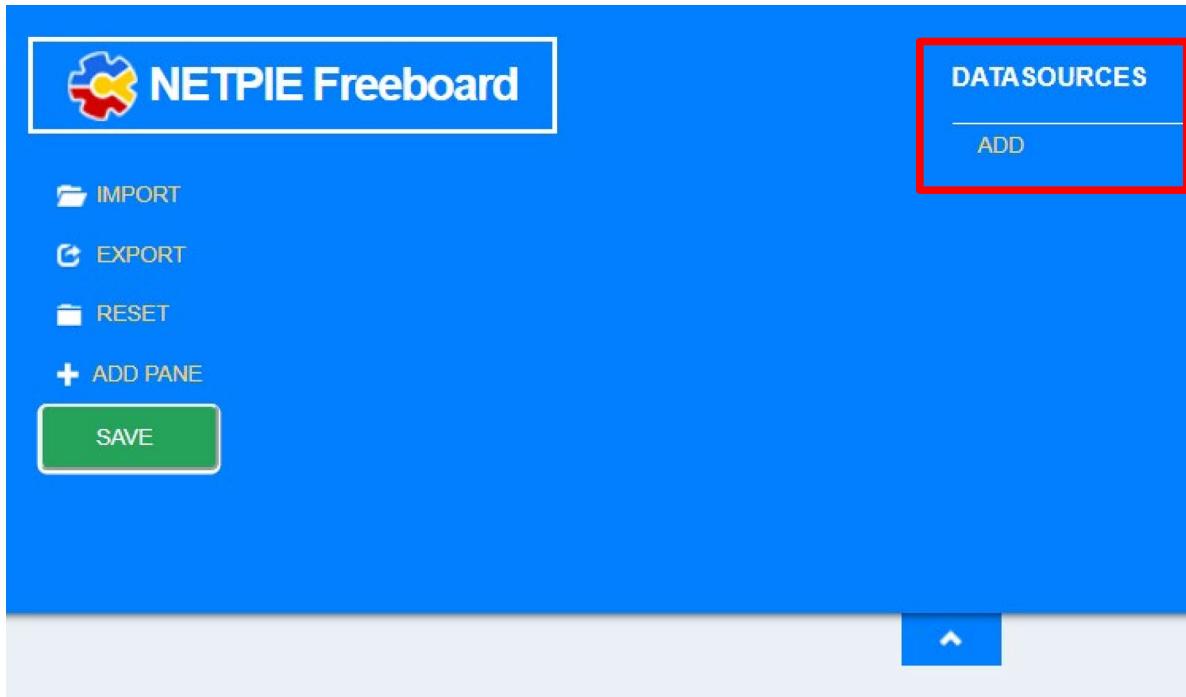
←

⚙

กด Publish ใน MQTTBox จากนั้นให้ไปดูผลลัพธ์ใน Device Shadow ว่าผลลัพธ์เป็นไปดังรูปต่อไปนี้หรือไม่ หากไม่เป็นไปตามรูป ให้ลองกด Refresh หน้าต่างใหม่



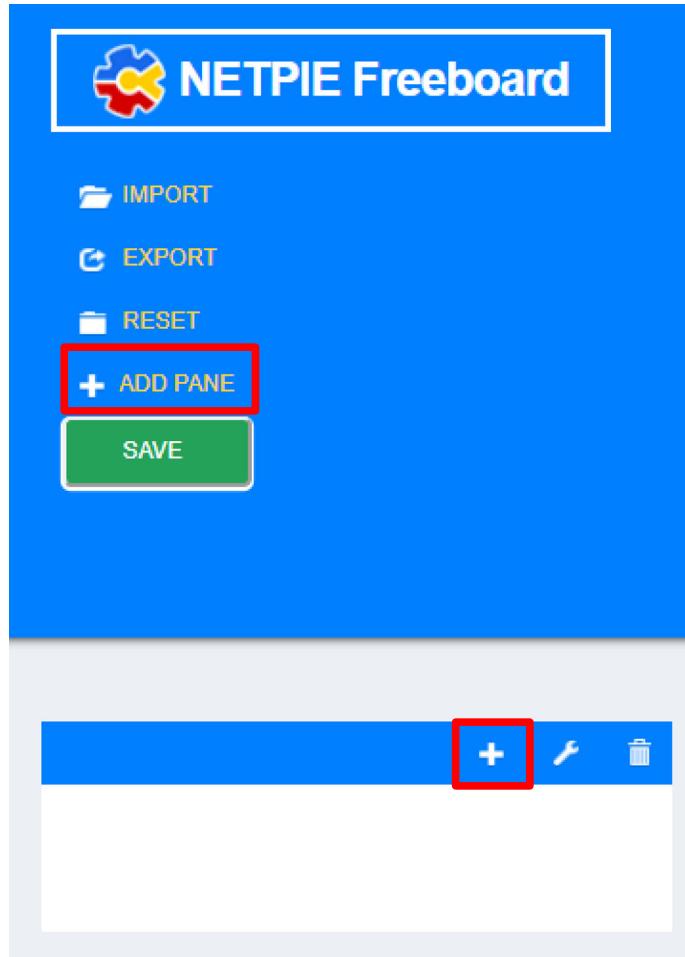
เมื่อข้อความถูกส่งขึ้น Device Shadow และ ให้ทำการตั้งค่าในส่วนของ Freeboard เริ่มต้นที่การเพิ่ม Datasource โดยป้อนข้อมูลที่จำเป็นให้หมด



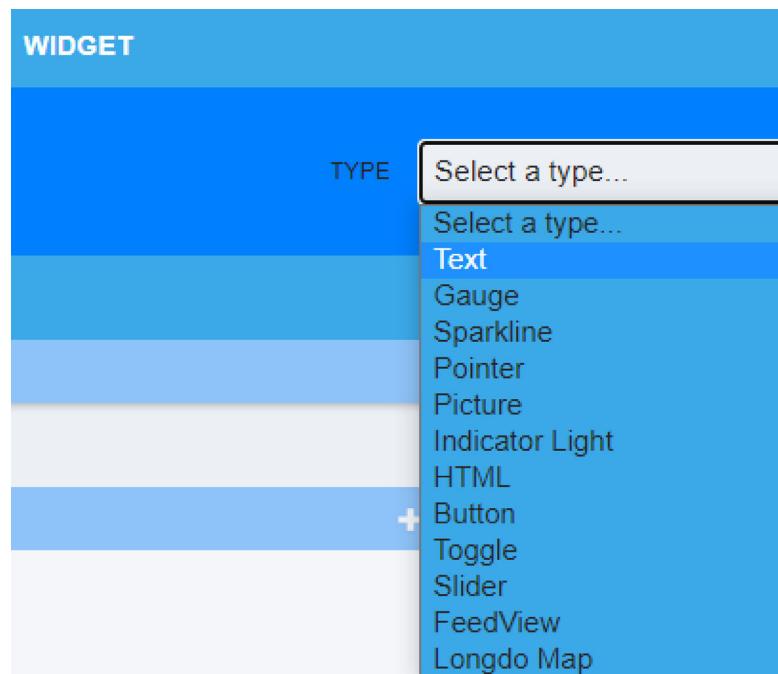
DATASOURCE

NAME	Data_ESP8266_2
DEVICE ID	03ca2f5f-b8cf-4cf4-9130-5e409219f232
Client ID ของ Device ที่ต้องการอ่านข้อมูล	
DEVICE TOKEN	5Ji3q42GWYBTcRxU4wP8Xh8XStsnWf3c
Token ของ Device ที่ต้องการอ่านข้อมูล	
SUBSCRIBED TOPICS	@shadow/data/update
Topic ที่ต้องการ Subscribe	
FEED	<input checked="" type="checkbox"/> NO
SINCE	6
Hour ▼	
Display data points since ... ago.	

จากนั้นกดปุ่ม Add Pane เพื่อเพิ่มช่องสำหรับใส่ Widget ที่เราต้องการ เมื่อสร้าง Pane เสร็จแล้ว ให้กด สัญลักษณ์ + ตรง Pane ที่สร้าง เพื่อเพิ่ม Widget ที่เราต้องการใช้งาน



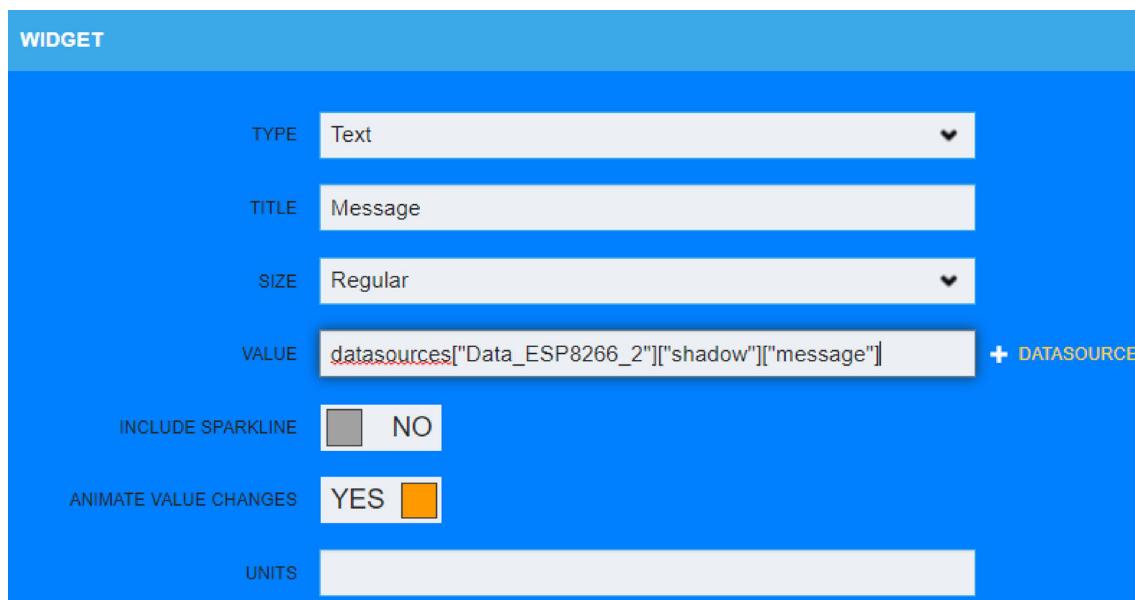
หน้าต่างการเลือก Widget จะปรากฏให้มาให้เห็น



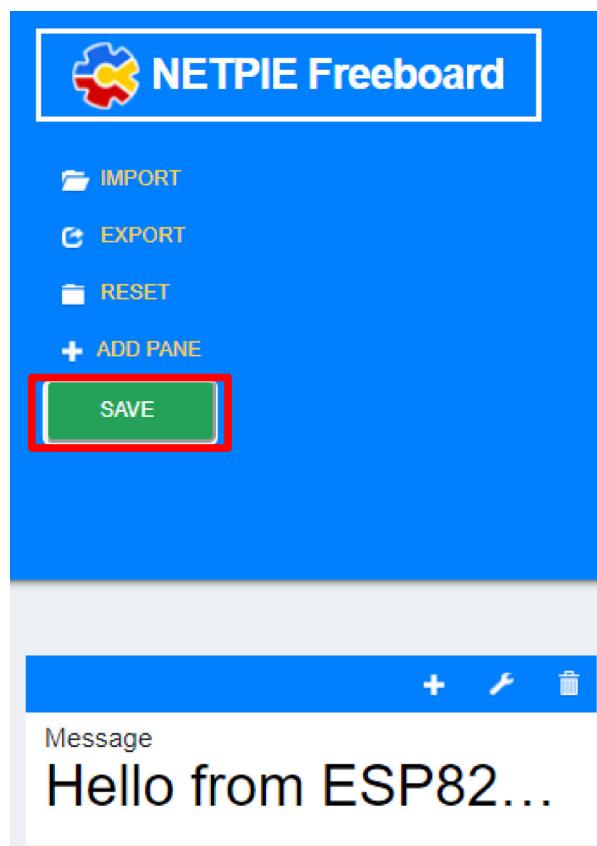
เลือก Widget ที่ต้องการใช้งาน ซึ่งในที่นี่เราจะใช้ Text เพื่อแสดงผลข้อความที่ส่งขึ้น Device Shadow เมื่อเลือกเสร็จแล้ว ให้กรอกข้อมูลในช่องต่างๆดังรูป

TYPE	Text
TITLE	Message
SIZE	Regular
INCLUDE SPARKLINE	Data_ESP8266_2
ANIMATE VALUE CHANGES	<input checked="" type="checkbox"/> TECO <input checked="" type="checkbox"/> orange
UNITS	

+ DATA SOURCE X JS EDITOR



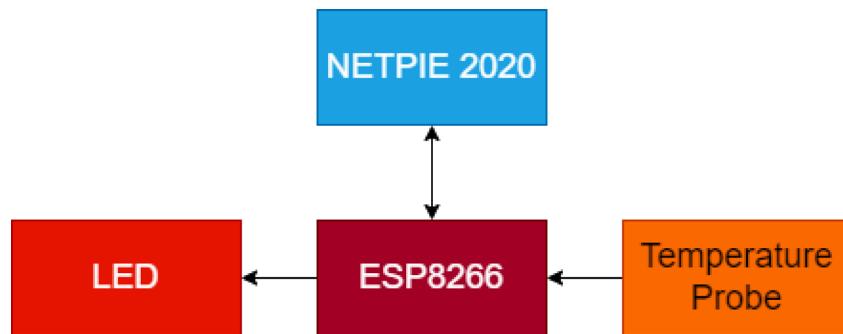
จากนั้นจึงกด Save ผลลัพธ์ที่ได้จะต้องเป็นไปตามรูป



สามารถหาข้อมูล และ ตัวอย่างการใช้งานควบคู่กับบอร์ดได้ที่ <https://netpie.io/guide>

Workshop 17: ระบบตรวจสอบอุณหภูมิโดยใช้ NETPIE 2020

ในการทดลองนี้ จะเป็นการทดลองใช้ Temperature Probe เพื่อวัดอุณหภูมิส่งไปยัง NETPIE และสามารถควบคุมหลอด LED ผ่าน NETPIE ได้



ขั้นตอนแรก จะเริ่มทำในส่วนของ NETPIE ก่อน โดยสิ่งที่เราต้องทำคือการสร้าง Project ที่ชื่อว่า “ESP8266_TEST” ดังรูป

Create Project

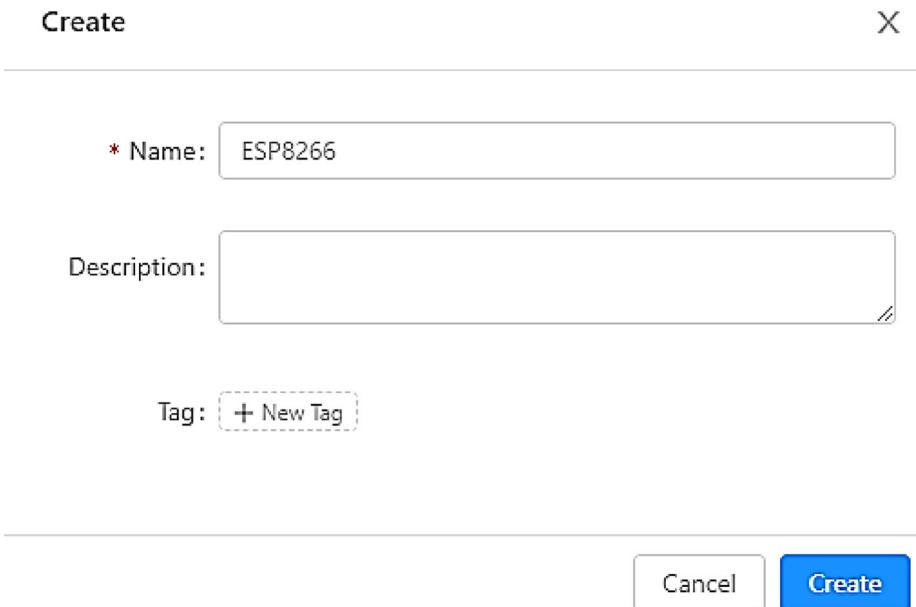
X

* Name:

Description:

Tag:

เมื่อสร้าง Project เสร็จแล้ว ต่อมาให้สร้าง Device ใน Project เพื่อรับการเชื่อมต่อที่มายจาก ESP8266 ของเรา โดยในที่นี่จะตั้งชื่อว่า “ESP8266”



หลังจากสร้าง Device ของ ESP8266 แล้ว ให้ตั้งค่าในส่วนของ Device Schema เพื่อรับข้อมูลที่ ESP8266 จะส่งมาดังนี้

```

1  [
2      "additionalProperties": true,
3      "properties": {
4          "temperature": {
5              "operation": {
6                  "store": {
7                      "ttl": "2d"
8                  }
9              },
10             "type": "number"
11         },
12         "ledState": {
13             "operation": {
14                 "store": {
15                     "ttl": "2d"
16                 }
17             },
18             "type": "string"
19         }
20     }
21 }

```

The code editor displays a JSON schema. It starts with an array bracket [on line 1, followed by a colon and a brace { on line 2. Inside the brace, there is a key 'additionalProperties' with a value of 'true' (line 2). Then there is a key 'properties' with a value of a brace {} (line 3). Inside this brace, there are two keys: 'temperature' and 'ledState' (lines 4-5). Each key has an associated 'operation' object (lines 6-7) which contains a 'store' object (line 7) with a 'ttl' value of '2d' (line 8). Finally, there is a 'type' key with a value of either 'number' or 'string' (lines 9-18). The brace for the properties ends on line 20, and the brace for the entire schema ends on line 21.

หลังจากตั้งค่า Device Schema แล้ว จะนั่นจึงเริ่มเขียนโปรแกรมให้กับ ESP8266 เพื่อรับส่งข้อมูลกับ NETPIE โดยโปรแกรมสามารถเขียนได้ดังนี้

```

1 #include <ESP8266WiFi.h>
2 #include <PubSubClient.h>
3 #include <OneWire.h>
4 #include <DallasTemperature.h>
5
6 #define ONE_WIRE_BUS 4
7 #define LED 13
8
9 const char* ssid = ""; ;                                //****change
10 const char* password = ""; ;                            //****change
11 const char* mqtt_server = "broker.netpie.io";
12 const int mqtt_port = 1883;
13 const char* mqtt_Client = ""; ;                         //****change
14 const char* mqtt_Token = ""; ;                          //****change
15 const char* mqtt_Secret = ""; ;                         //****change
16
17 OneWire oneWire(ONE_WIRE_BUS);
18 DallasTemperature tempProbe(&oneWire);
19 WiFiClient espClient;
20 PubSubClient client(espClient);

```

บรรทัด ที่	คำอธิบาย
1	เพิ่ม Library ESP8266WiFi เพื่อใช้งาน WiFi บนบอร์ด ESP8266
2	เพิ่ม Library PubSubClient เพื่อใช้งานกับการสื่อสารแบบ MQTT
3	เพิ่ม Library OneWire เพื่อใช้งานกับการสื่อสารแบบ One-Wire Communication
4	เพิ่ม Library DallasTemperature เพื่อใช้งานกับ Temperature Probe Sensor
6	กำหนดค่าคงที่ชื่อ ONE_WIRE_BUS เป็น 4 เพื่อระบุว่า ขารับส่งข้อมูลของ Temperature Probe Sensor อยู่ที่ GPIO4 (D2)
7	กำหนดค่าคงที่ LED เป็น 13 เพื่อระบุว่า ขาควบคุม LED อยู่ที่ GPIO13 (D7)

บรรทัด ที่	คำอธิบาย
9	กำหนดค่าคงที่ชื่อ ssid เพื่อรับชื่อของสัญญาณอินเทอร์เน็ตที่จะเชื่อมต่อ
10	กำหนดค่าคงที่ชื่อ password เพื่อรับรหัสผ่านของสัญญาณอินเทอร์เน็ตที่จะเชื่อมต่อ
11	กำหนดค่าคงที่ชื่อ mqtt_server เพื่อรับที่อยู่ของเซิฟเวอร์ MQTT ที่จะเชื่อมต่อ
12	กำหนดค่าคงที่ชื่อ mqtt_port เพื่อรับพอร์ตของเซิฟเวอร์ MQTT ที่จะเชื่อมต่อ
13	กำหนดค่าคงที่ชื่อ mqtt_Client เพื่อรับ Client ID ของอุปกรณ์ที่จะเชื่อมต่อ
14	กำหนดค่าคงที่ชื่อ mqtt_Token เพื่อรับ Token ของอุปกรณ์ที่จะเชื่อมต่อ
15	กำหนดค่าคงที่ชื่อ mqtt_Secret เพื่อรับ Secret ของอุปกรณ์ที่จะเชื่อมต่อ
17	สร้าง Object ของ OneWire ชื่อ onewire เพื่อสร้างการเชื่อมต่อแบบ One-Wire Communication โดยป้อนค่าคงที่ ONE_WIRE_BUS เข้าไปเพื่อรับขาพินที่จะใช้ในการสื่อสาร
18	สร้าง Object ของ DallasTemperature ชื่อ tempProbe เพื่อสร้างชุดคำสั่งสำหรับในการอ่านค่า อุณหภูมิจาก Temperature Probe Sensor โดยป้อน onewire เข้าไปเพื่อให้ tempProbe อ่านค่า อุณหภูมิผ่านทาง onewire
19	สร้าง Object ของ WiFiClient ชื่อ espClient เพื่อสร้างชุดคำสั่งสำหรับสำหรับใช้งาน WiFi บน ESP8266
20	สร้าง Object ของ PubSubClient ชื่อ client เพื่อสร้างชุดคำสั่งสำหรับสำหรับการสื่อสารผ่าน MQTT Protocol โดยป้อน espClient เพื่อให้ client ใช้งาน WiFi บน ESP8266 ผ่านทาง espClient

```

22 char msg[100];
23 float temp = 0.0;
24 bool isReady = false;
25
26 void reconnect();
27 void onoff(int ); //sent LED status to netpie
28 void callback(char* , byte* , unsigned int ); //get data from netpie
29 float readTemp();
30 void displayTemp();
31 void displayLED();

```

บรรทัด ที่	คำอธิบาย
22	สร้าง array of char ขึ้นมาชื่อ msg มีขนาด 100 elements เพื่อรับข้อมูลที่เซิฟเวอร์ MQTT จะส่งมา
23	สร้างตัวแปร float ขึ้นมาชื่อ temp เพื่อรับข้อมูลที่ได้จาก Temperature Probe Sensor โดยกำหนดค่าเริ่มต้นไว้ที่ 0.0
24	สร้างตัวแปร boolean ขึ้นมาชื่อ isReady เพื่อบอกสถานะของโปรแกรมว่าทำงานจบในส่วนของพังก์ชัน void setup() หรือยัง โดยกำหนดให้ค่าเริ่มต้นมีค่าเป็น false
26	สร้างฟังก์ชัน void ชื่อ reconnect เพื่อเชื่อมต่อ กับเซิฟเวอร์ MQTT ใหม่ในกรณีที่ขาดการเชื่อมต่อ
27	สร้างฟังก์ชัน void ชื่อ onoff เพื่อควบคุม LED และ ส่งข้อมูลกลับไปให้เซิฟเวอร์ MQTT
28	สร้างฟังก์ชัน void ชื่อ callback เพื่อรับข้อมูลที่มาจากการเซิฟเวอร์ MQTT
29	สร้างฟังก์ชัน float ชื่อ readTemp เพื่อควบคุมการอ่านค่าอุณหภูมิจาก Temperature Probe Sensor
30	สร้างฟังก์ชัน void ชื่อ displayTemp เพื่อแสดงผลข้อมูลอุณหภูมิที่ได้รับออกมาที่ Serial Monitor
31	สร้างฟังก์ชัน void ชื่อ displayLED เพื่อแสดงผลสถานะของ LED ออกมาที่ Serial Monitor

```

33 void setup() {
34   Serial.begin(115200);
35   Serial.println("Dallas Temperature IC Control Library");
36   tempProbe.begin();
37   pinMode(LED, OUTPUT);
38
39   Serial.println("Connecting to ");
40   Serial.println(ssid);
41   WiFi.mode(WIFI_STA);
42   WiFi.begin(ssid, password);           // access wifi
43   while (WiFi.status() != WL_CONNECTED) //check disconnect
44   {
45     delay(500);
46     Serial.print(".");
47   }
48   Serial.println(" WiFi connected");
49   Serial.println("IP address: ");
50   Serial.println(WiFi.localIP());
51   client.setServer(mqtt_server, mqtt_port);
52   client.setCallback(callback);
53
54   isReady = true;
55 }
```

บรรทัด ที่	คำอธิบาย
33	เข้าสู่ฟังก์ชัน void setup()
34	ตั้ง BaudRate ของ Serial Monitor ไว้ที่ 115200
35	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
36	เปิดการใช้งาน Temperature Probe Sensor
37	ตั้งรูปแบบการทำงานของขา LED เป็น OUTPUT
39	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่

บรรทัด ที่	คำอธิบาย
40	แสดงชื่อของอินเทอร์เน็ตที่เชื่อมต่อออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
41	ตั้งรูปแบบการทำงานของ WiFi บน ESP8266 ให้ทำงานแบบ station
42	เปิดการใช้งาน WiFi บน ESP8266 โดยให้เชื่อมต่อไปยังชื่ออินเทอร์เน็ตที่ระบุ และ ป้อนรหัสตามที่ระบุไว้
43	สร้าง while loop โดยกำหนดเงื่อนไขให้ตรวจสอบสถานะการเชื่อมต่อว่าเชื่อมต่ออินเทอร์เน็ตสำเร็จ หรือไม่
44	เข้าสู่ while loop
45	หน่วงเวลาไว้ 500 มิลลิวินาที
46	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
47	จบการทำงาน while loop
48	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
49	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
50	แสดง local IP Address ออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
51	ตั้งค่าเซิฟเวอร์ MQTT และพอร์ตปลายทางที่จะเชื่อมต่อ
52	ระบบฟังก์ชันที่จะทำงานเมื่อมีข้อมูลส่งมาจากเซิฟเวอร์ MQTT
54	กำหนดค่าให้ isReady เป็น true เพื่อระบบให้โปรแกรมรู้ว่า void setup() ทำงานเสร็จสิ้นแล้ว
55	จบการทำงานของ void setup()

```

57 void loop() {
58   if (!client.connected())
59     reconnect();
60
61   client.loop();
62
63   if (millis() % 1000 == 0 && isReady) {
64     temp = readTemp();
65     String data = "{\"data\": {\"temperature\": " + String(temp) + "}}";
66     data.toCharArray(msg, data.length() + 1);
67     client.publish("@shadow/data/update", msg); // sent data to shadow netpie
68     displayTemp();
69     displayLED();
70   }
71 }

```

บรรทัด ที่	คำอธิบาย
57	เริ่มต้นการทำงานของ void loop()
58	สร้างเงื่อนไขดักไว้ ในกรณีที่การเชื่อมต่อ กับเซิฟเวอร์ MQTT มีปัญหา ให้ทำงานคำสั่งภายใน
59	เรียกใช้ฟังก์ชัน reconnect()
61	เริ่มต้นการสื่อสารกับเซิฟเวอร์ MQTT
63	สร้างเงื่อนไขการแสดงผล และ ส่งข้อมูลไปที่เซิฟเวอร์ MQTT โดยให้ทำงานทุกๆ 1 วินาที และ void setup() ต้องทำงานเสร็จสมบูรณ์แล้วเท่านั้น
64	อ่านค่าอุณหภูมิโดยใช้ฟังก์ชัน readTemp() และมาเก็บไว้ที่ตัวแปร temp
65	สร้างตัวแปร String ชื่อ data เพื่อประกอบข้อมูลให้พร้อมส่ง โดยกำหนดค่าเป็นโครงสร้างข้อมูลแบบ JSON ในรูปแบบ String
66	นำตัวแปร data ไปแปลงเป็น array of char โดยเก็บไว้ที่ msg
67	ส่งข้อมูลแบบ MQTT ออกไปที่เซิฟเวอร์ MQTT โดยข้อมูลที่ส่งไปจะถูกบันทึกที่ shadow ของ NETPIE

บรรทัด ที่	คำอธิบาย
68	แสดงผลข้อมูลอุณหภูมิทาง Serial Monitor
69	แสดงผลสถานะ LED ทาง Serial Monitor
70	จบการทำงานของเงื่อนไขการแสดงผล และ ส่งข้อมูล
71	จบการทำงานของ void loop()

```

73 void reconnect()
74 {
75   while (!client.connected()) {
76     Serial.print("Attempting MQTT connection...");
77     if (client.connect(mqtt_Client, mqtt_Token, mqtt_Secret)) {
78       Serial.println("connected");
79       client.subscribe("@msg/ledState");
80     }
81     else
82     {
83       Serial.print("failed, rc=");
84       Serial.print(client.state());
85       Serial.print("Try again in 5 sec");
86       delay(5000);
87     }
88   }
89 }
```

บรรทัด ที่	คำอธิบาย
73	ประกาศการทำงานของฟังก์ชัน void reconnect()
74	เริ่มการประกาศการทำงาน
75	สร้าง while loop เพื่อตรวจสอบการเชื่อมต่อระหว่างอุปกรณ์กับเซิฟเวอร์ MQTT จนกว่าการเชื่อมต่อจะเป็นปกติ

บรรทัด ที่	คำอธิบาย
76	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
77	เขียนต่อเข้าสู่เซิฟเวอร์ MQTT ปลายทาง พร้อมทั้งตรวจสอบผลการเขียนต่อ
78	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
79	ให้อุปกรณ์ subscribe ข้อมูลที่ส่งมาใน topic ที่ระบุ
80	จบการทำงานของ if
81	หากการเขียนต่อในบรรทัดที่ 77 ไม่สำเร็จ ให้ทำงานตามที่ระบุใน else
82	เริ่มต้นระบุการทำงานของ else
83	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
84	แสดงผลลัพธ์การเขียนต่อออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
85	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
86	หน่วงเวลาไว้ 5 วินาที
87	จบการทำงานของ else
88	จบการทำงานของ while loop
89	จบการประกาศการทำงานของฟังก์ชัน void reconnect()

```

91 void callback(char* topic, byte* payload, unsigned int length) //get data from netpie
92 {
93     Serial.print("Message arrived [");
94     Serial.print(topic);
95     Serial.print(": ");
96     String msg;
97     for (int i = 0; i < length; i++) {
98         msg = msg + (char)payload[i];
99     }
100    Serial.println(msg);
101    if (String(topic) == "@msg/ledState")
102        if (msg == "on") {
103            onoff(true);
104            client.publish("@shadow/data/update", "{\"data\" : {\"ledState\" : \"on\"}}");
105        }
106        else if (msg == "off") {
107            onoff(false);
108            client.publish("@shadow/data/update", "{\"data\" : {\"ledState\" : \"off\"}}");
109        }
110 }

```

บรรทัด ที่	คำอธิบาย
91	ประกาศการทำงานของฟังก์ชัน void callback(char* topic, byte* payload, unsigned int length)
92	เริ่มต้นการประกาศการทำงาน
93	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
94	แสดง topic ที่เชิฟเวอร์ MQTT ส่งมาให้ออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
95	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
96	สร้างตัวแปร String ชื่อ msg (คำลั่ตัวกับที่ใช้ใน void loop())
97	เริ่มต้น for loop เพื่อรับข้อมูลที่ส่งมาเก็บไว้ที่ msg โดยวนรอบตามความยาวของข้อมูลที่ส่งมา
98	รับข้อมูล payload แต่ละตัว มาเก็บไว้ใน msg โดยเรียงลำดับกัน
99	จบการทำงานของ for loop
100	แสดงข้อมูลที่ msg รับมาออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่

บรรทัด ที่	คำอธิบาย
101	สร้างเงื่อนไข if เพื่อตรวจสอบว่า topic ที่รับมาเป็น topic ที่ใช้สั่งการ LED หรือไม่
102	สร้างเงื่อนไข if เพื่อตรวจสอบว่าข้อมูลที่ส่งมาให้ คือการสั่งเปิด LED หรือไม่
103	เรียกใช้ฟังก์ชัน onoff() เพื่อควบคุม LED โดยป้อน true เข้าไป
104	ส่งข้อมูลแบบ MQTT ออกไปที่เซิฟเวอร์ MQTT โดยข้อมูลที่ส่งไปจะถูกบันทึกที่ shadow ของ NETPIE
105	จบการทำงานของ if ในบรรทัดที่ 102
106	สร้างเงื่อนไข else if เพื่อตรวจสอบว่าข้อมูลที่ส่งมาให้ คือการสั่งปิด LED หรือไม่
107	เรียกใช้ฟังก์ชัน onoff() เพื่อควบคุม LED โดยป้อน false เข้าไป
108	ส่งข้อมูลแบบ MQTT ออกไปที่เซิฟเวอร์ MQTT โดยข้อมูลที่ส่งไปจะถูกบันทึกที่ shadow ของ NETPIE
109	จบการทำงานของ else if ในบรรทัดที่ 106
110	จบการประกาศการทำงานของฟังก์ชัน void void callback(char* topic, byte* payload, unsigned int length)

```

112 void onoff(bool led) //sent LED status to netpie
113 {
114     if (led)
115     {
116         Serial.println("Turn on LED");
117         digitalWrite(LED, HIGH);
118     }
119     else if (!led)
120     {
121         Serial.println("Turn off LED");
122         digitalWrite(LED, LOW);
123     }
124 }
```

บรรทัดที่	คำอธิบาย
112	ประกาศการทำงานของฟังก์ชัน void onoff(bool)
113	เริ่มต้นการประกาศการทำงาน
114	สร้างเงื่อนไข if เพื่อตรวจสอบว่า led เป็น true หรือไม่
115	เริ่มประกาศการทำงานของเงื่อนไข if
116	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
117	สั่งเปิด LED
118	จบการทำงานของเงื่อนไข if
119	สร้างเงื่อนไข else if เพื่อตรวจสอบว่า led เป็น false หรือไม่
120	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
121	สั่งปิด LED

บรรทัดที่	คำอธิบาย
122	จบการทำงานของเงื่อนไข else if
123	จบการประกาศการทำงานของฟังก์ชัน void onoff(bool)

```

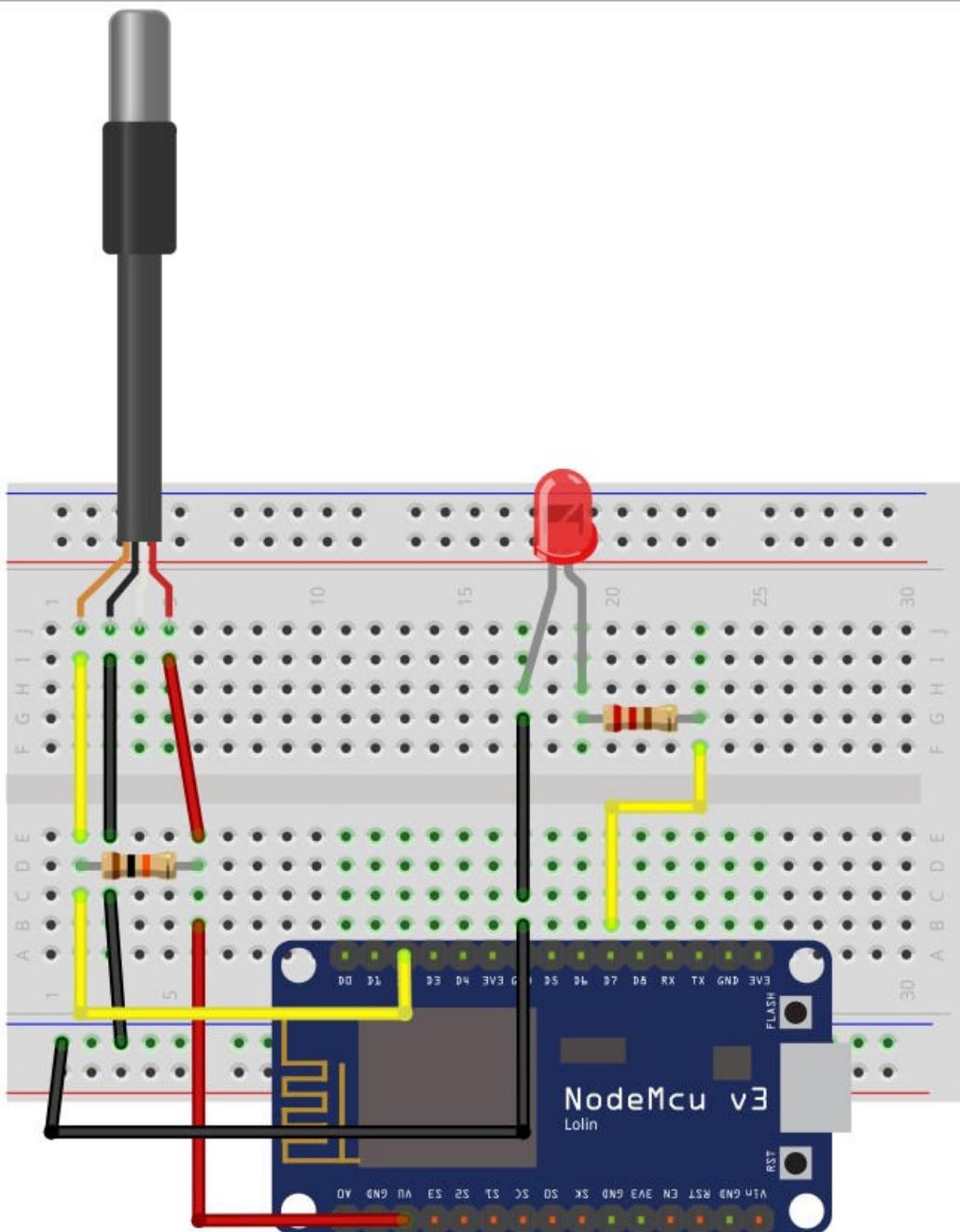
126 float readTemp() {
127     tempProbe.requestTemperatures();
128     return tempProbe.getTempCByIndex(0);
129 }
130
131 void displayTemp() {
132     Serial.print("temp = ");
133     Serial.print(temp);
134     Serial.println(" °C");
135 }
136
137 void displayLED() {
138     Serial.print("LED Status : ");
139     Serial.println(digitalRead(LED));
140     Serial.println();
141 }
```

บรรทัดที่	คำอธิบาย
126	ประกาศการทำงานของฟังก์ชัน float readTemp()
127	ส่งคำสั่งเพื่อขอข้อมูลอุณหภูมิจาก Temperature Probe Sensor
128	รับ และ คืนค่าอุณหภูมิที่ได้จาก Temperature Probe Sensor
129	จบการประกาศการทำงานของฟังก์ชัน float readTemp()
131	ประกาศการทำงานของฟังก์ชัน void displayTemp()

บรรทัดที่	คำอธิบาย
132	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
133	แสดงอุณหภูมิที่อ่านได้ออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
134	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
135	จบการประกาศการทำงานของฟังก์ชัน void displayTemp()
137	ประกาศการทำงานของฟังก์ชัน void displayLED()
138	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
139	แสดงสถานะของ LED ออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
140	เว้นบรรทัดใน Serial Monitor
141	จบการประกาศการทำงานของฟังก์ชัน void displayLED()

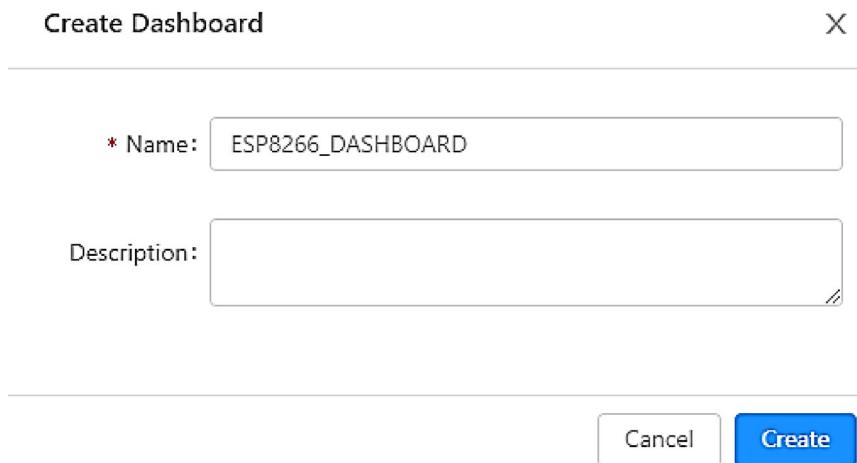
การต่อวงจร

PIN	อุปกรณ์
D1	DS18B20
D7	ตัวต้านทานขนาด 4.7k โอห์ม



รูปจำลองการต่อวงจร

เมื่อเขียนโปรแกรม, อัพโหลดโปรแกรมลง ESP8266 และ ต่อวงจรเสร็จแล้ว ทำการสร้าง Dashboard สำหรับการแสดงผลข้อมูลโดยไปที่ Freeboard โดยตั้งชื่อว่า “ESP8266_DASHBOARD”



จากนั้น เข้าสู่ Freeboard และทำการเพิ่ม Data Source โดยใส่ข้อมูลที่จำเป็นให้ครบ

DATA SOURCE	
NAME	ESP8266_DATA
DEVICE ID	a43fa447-1a4c-433c-83ca-05253a926dbd
Client ID ของ Device ที่ต้องการอ่านข้อมูล	
DEVICE TOKEN	67DTu49sdTeaHrzuKGDH17a6oZ6x4nvJ
Token ของ Device ที่ต้องการอ่านข้อมูล	
SUBSCRIBED TOPICS	
Topic ที่ต้องการ Subscribe	
FEED	<input checked="" type="checkbox"/> NO
SINCE	6
Hour	
Display data points since ... ago.	

หลังจากเพิ่ม Data Source เสร็จแล้ว ทำการเพิ่ม Pane 3 ตัวเพื่อรับ Widget จะเพิ่มเข้าไปใน Dashboard โดย Widget ที่เพิ่มมี 3 ตัวได้แก่ Gauge, Toggle และ Indicator Light โดยสามารถตั้งค่า Widget ทั้ง 3 ตัวได้ดังภาพข้างล่างต่อไปนี้

The image shows two screenshots of the Grafana interface. The top screenshot is a configuration panel for a 'Gauge' widget. It includes fields for 'TITLE' (Temperature), 'VALUE' (datasources["ESP8266_DATA"]["shadow"]["temperature"] - highlighted in red), 'UNITS' (C), 'MINIMUM' (0), and 'MAXIMUM' (100). Below this is a 'JS EDITOR' button. The bottom screenshot shows the live preview of the dashboard. It displays a gauge chart titled 'Temperature' with a value of -127°C, set against a scale from 0 to 100.

WIDGET

TYPE
Gauge

TITLE
Temperature

VALUE
datasources["ESP8266_DATA"]["shadow"]["temperature"]

UNITS
C

MINIMUM
0

MAXIMUM
100

JS EDITOR

DATA SOURCES

Name	Last Updated
ESP8266_DATA	1:54:12 AM

ADD

Temperature

-127 °C

WIDGET

TYPE
Indicator Light

TITLE
LED Status

DEFAULT COLOR

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

VALUE
`datasources["ESP8266_DATA"]["shadow"]["ledState"]=="on"`

+ DATASOURCE  JS EDITOR

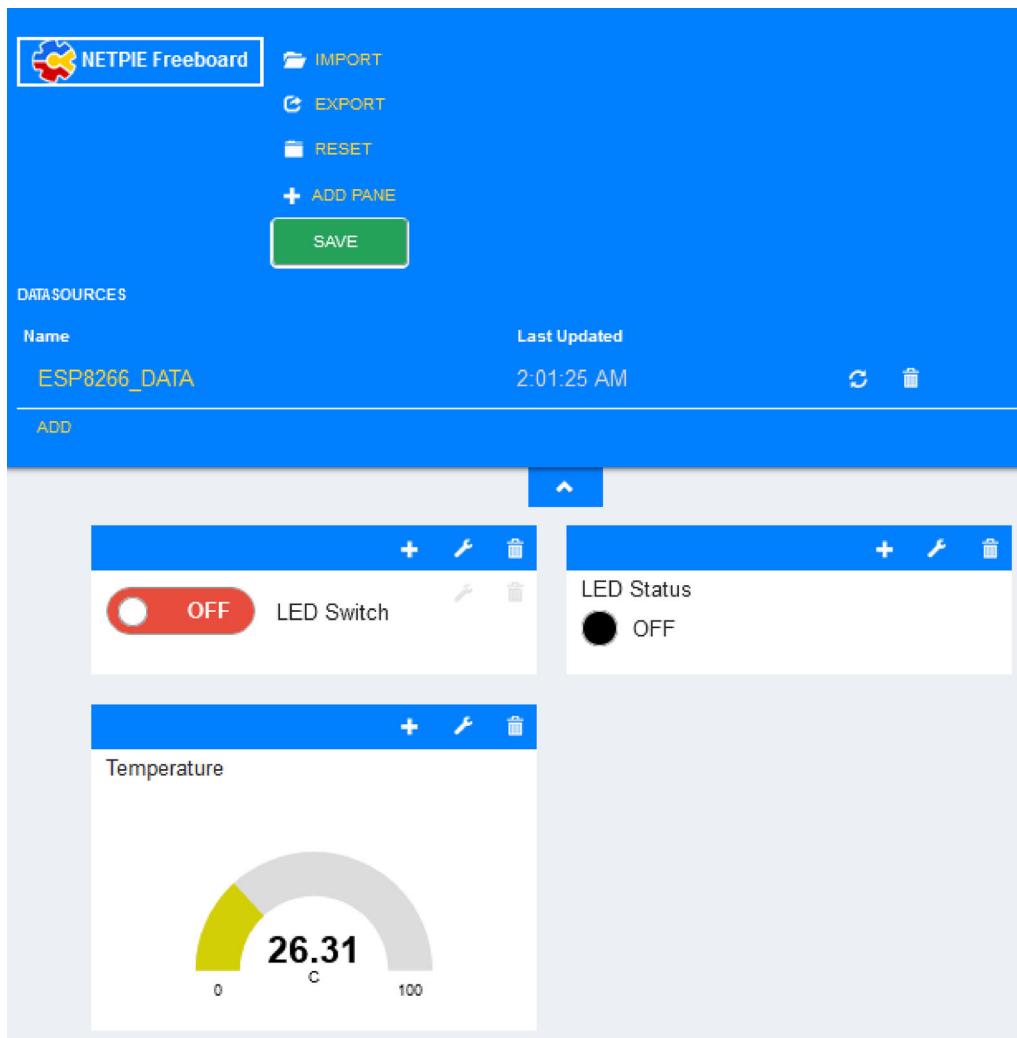
ON TEXT
ON

+ DATASOURCE  JS EDITOR

OFF TEXT
OFF

+ DATASOURCE  JS EDITOR

SAVE



ตัวอย่างผลลัพธ์การทำงานบน Freeboard

The screenshot shows the NETPIE Device Management interface. On the left, a sidebar menu includes: Overview, Device List, Device Groups, Freeboard, Event Hooks, and Setting. The main area displays the device configuration for 'ESP8266_TEST / device / ESP8266'. The top right shows a user profile for 'Warakorn'. The central panel has tabs for Description, Key, and Status. The Status tab shows 'Status : Online' and an 'Enable' toggle switch. Below these are buttons for Save and Cancel. A 'Tree' section titled 'Select a node...' shows a hierarchical structure with 'object {2}' expanded, showing 'ledState : off' and 'temperature : 26.25'. The 'Shadow' tab is currently selected.

ตัวอย่างผลลัพธ์การทำงานบน Device Shadow

The screenshot shows a Windows-style application window titled "Serial Monitor". The title bar includes standard window controls (minimize, maximize, close) and a "Send" button. The main window displays a list of text entries, each consisting of a timestamp followed by a message. The messages indicate periodic updates of temperature and LED status. The bottom of the window contains several configuration buttons: "Autoscroll" (checked), "Show timestamp" (checked), "Newline" (dropdown menu), "115200 baud" (dropdown menu), and "Clear output".

```
18:54:06.838 -> temp = 25.38 °C
18:54:06.838 -> LED Status : 1
18:54:06.838 ->
18:54:07.866 -> temp = 25.38 °C
18:54:07.866 -> LED Status : 1
18:54:07.866 ->
18:54:08.877 -> temp = 25.31 °C
18:54:08.877 -> LED Status : 1
18:54:08.877 ->
18:54:09.854 -> temp = 25.31 °C
18:54:09.854 -> LED Status : 1
18:54:09.854 ->
18:54:10.879 -> temp = 25.31 °C
18:54:10.879 -> LED Status : 1
18:54:10.879 ->
```

ตัวอย่างผลลัพธ์บน Serial Monitor