



Introducing NETPIE2020

& Workshop

Mr. Piyawat Jomsathan (Tae)

Cyber-Physical Systems (CPS)

National Electronics and Computer Technology Center (NECTEC), Thailand



Note : This powerpoint is modified significantly by Varodom Toochnida to keep up with the latest version of NETPIE 2020.

Some slides may still show user interface from previous version when changes are deemed unnecessary.



Personal Contact

NETPIE2020 Trainer

Cyber Physical Systems (CPS)

NECTEC Building Room 418



Mr.Piyawat Jomsathan

Nickname : Tae

Age : 23

E-mail : piyawat.jom@nectec.or.th

Tel. 02-564-6900 ext. 2469



Mr.Natapon Tansangworn

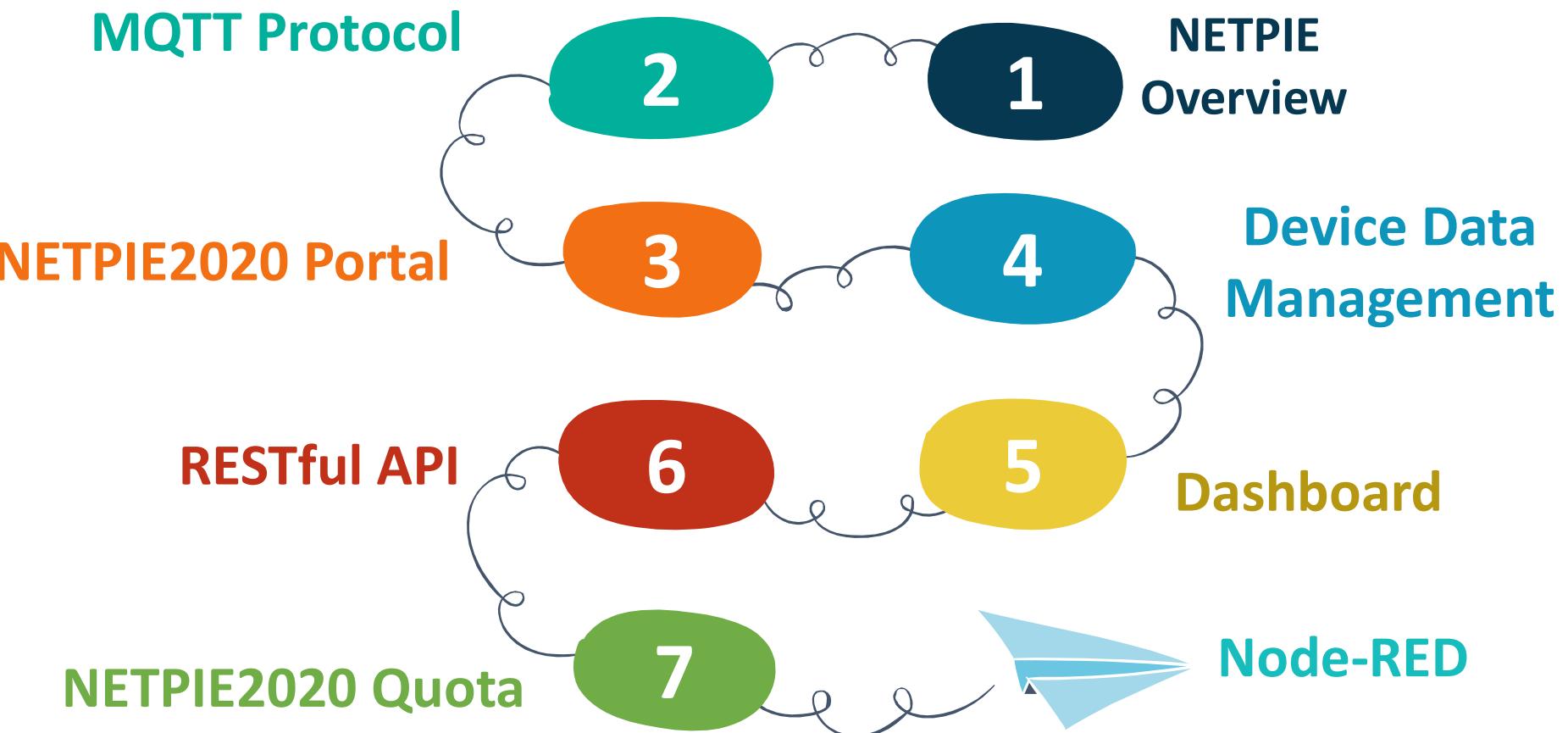
Nickname : Prig

Age : 28

E-mail : natapon.tan@nectec.or.th

Tel. 02-564-6900 ext. 2463

Schedule





NETPIE Overview



29 July 2024



Page 5

Module 1 : IoT & NETPIE

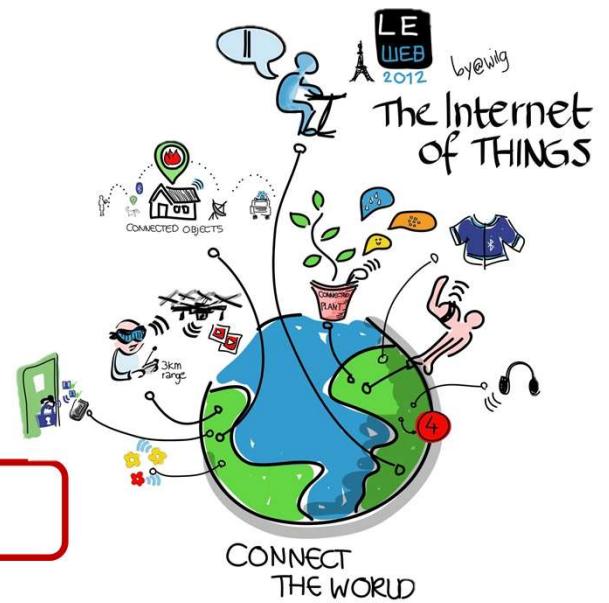


What is NETPIE?

NETPIE is an IoT cloud platform that is open to the public. The platform will help devices able to communicate with each other, receive - transmit data between devices in real-time, allowing users to know the information of the device at that time, no matter where the user is.

Main features of NETPIE

1. Monitoring : to display device or sensor data in real time
2. Controlling : to control the operation of devices via the cloud platform
3. Data Storage : to collect data from sensors or devices
4. Notification : to alert the user when sensor malfunction occurs



Module 1 : IoT & NETPIE

NETPIE 2015

At present, NETPIE is developed in 2 versions
NETPIE 2015 and NETPIE 2020

The screenshot shows the NETPIE 2015 web interface. At the top, there are two large blue boxes: one labeled '3 APPLICATIONS' and another labeled '26 THINGS'. Below these are smaller boxes for individual projects: 'm5StackProject' (0 Thing(s)), 'myrasberry' (20 Thing(s)), and 'netpiedemotest' (6 Thing(s)). The URL <https://2015.netpie.io/login> is highlighted with a red border.

NETPIE 2020

The screenshot shows the NETPIE 2020 web interface. It displays two project cards: 'Project_Docume...' (1 Device | 0 Groups) and 'Smart_Factory_I...' (6 Devices | 1 Groups). The URL <https://auth.netpie.io/login> is highlighted with a purple border. The bottom right corner features a blue '+' button. The footer contains the text 'Copyright © 2018-2020 Created by NETPIE'.

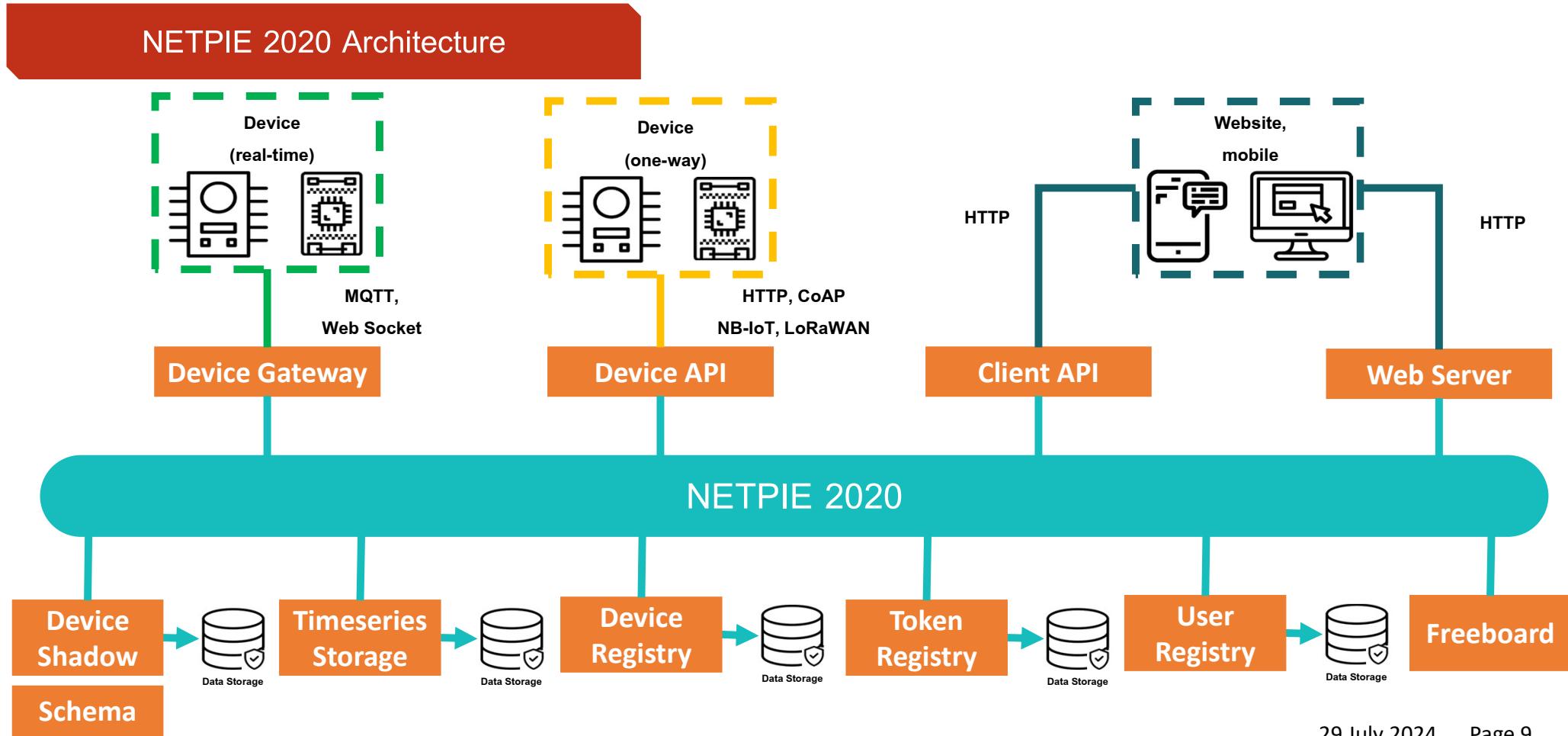
Module 1 : IoT & NETPIE

NETPIE 2020 and NETPIE 2015 comparison

	NETPIE 2020	NETPIE 2015
Design Philosophy	Platform - Centric	Device – Centric
Commercial Ability	Commercial – Ready	Need to re-program Hardware migrate to commercial platform
Suitable Usage	Mass production, Project-based	Project-based
Target User	IoT consumer product makers, Hobbyists, Students	Makers, hobbyists, students
Communication Protocol	MQTT, HTTP	Microgear
Programming Language	Any languages with MQTT library support	Limited to Microgear library
Hardware Support	Unlimited as long as it supports MQTT	Limited to those with Microgear support
Device Identity and Group	No APPID Device identity and group can be adjusted after product is sold/installed	Use APPID Device identity and group must be programmed into firmware
Rate Limit	Allow burst	Everyone is subject to the same rate limit
Trigger	Can set trigger action in cloud platform	Set trigger action inside IoT devices

NETPIE Overview

NETPIE 2020 Architecture





MQTT Protocol



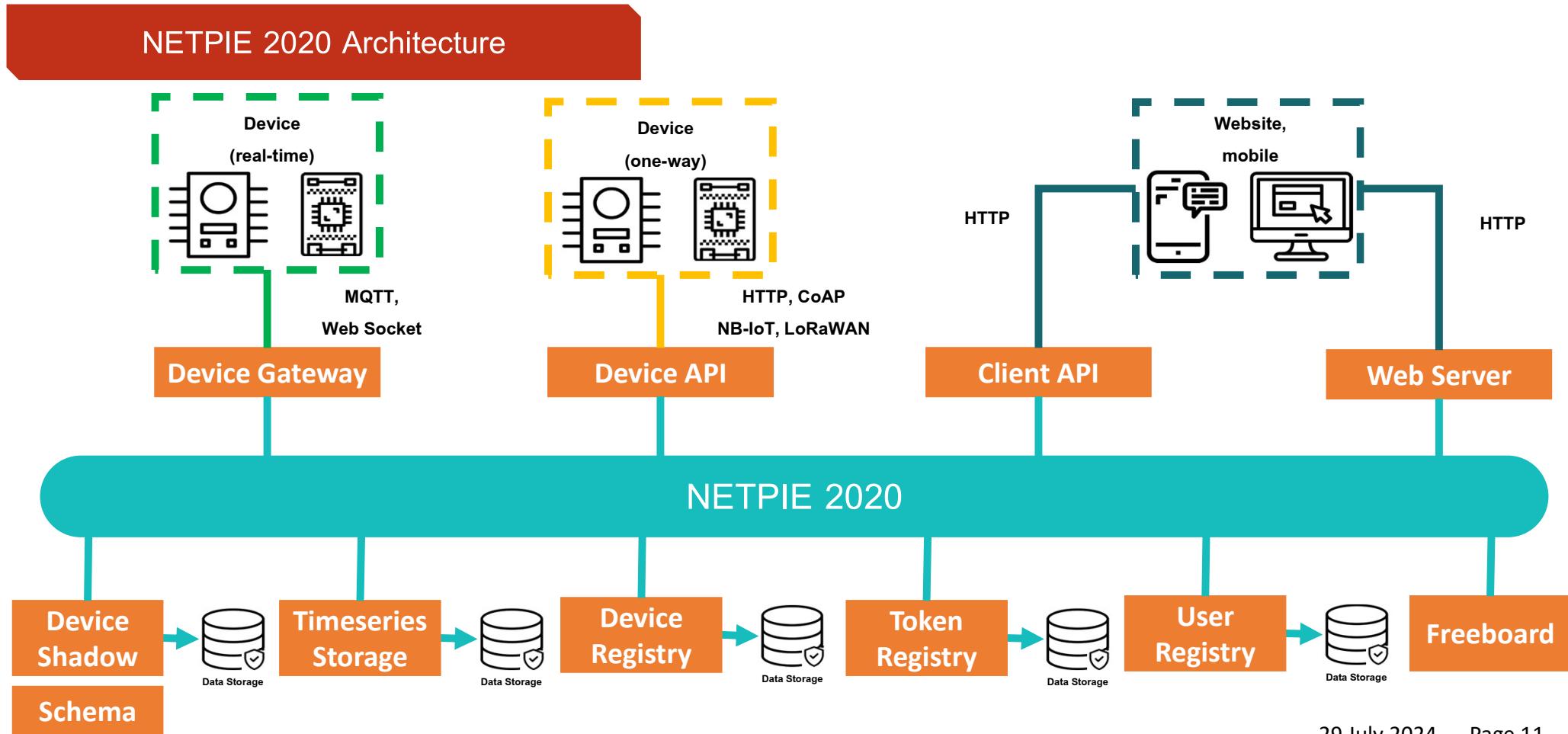
29 July 2024



Page 10

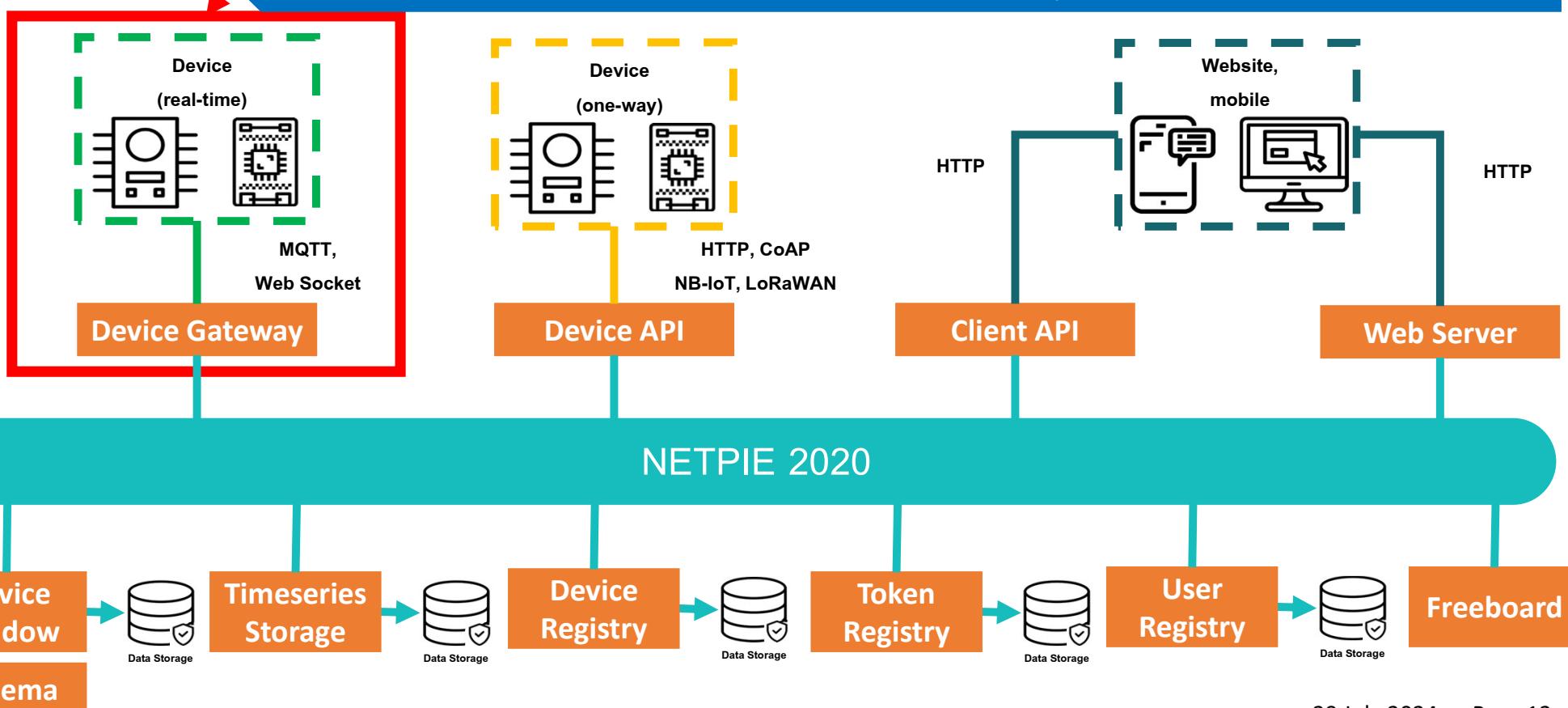
NETPIE Overview

NETPIE 2020 Architecture



MQTT Protocol

If real-time monitoring is required on NETPIE2020, a device gateway is required to communicate with the MQTT protocol.

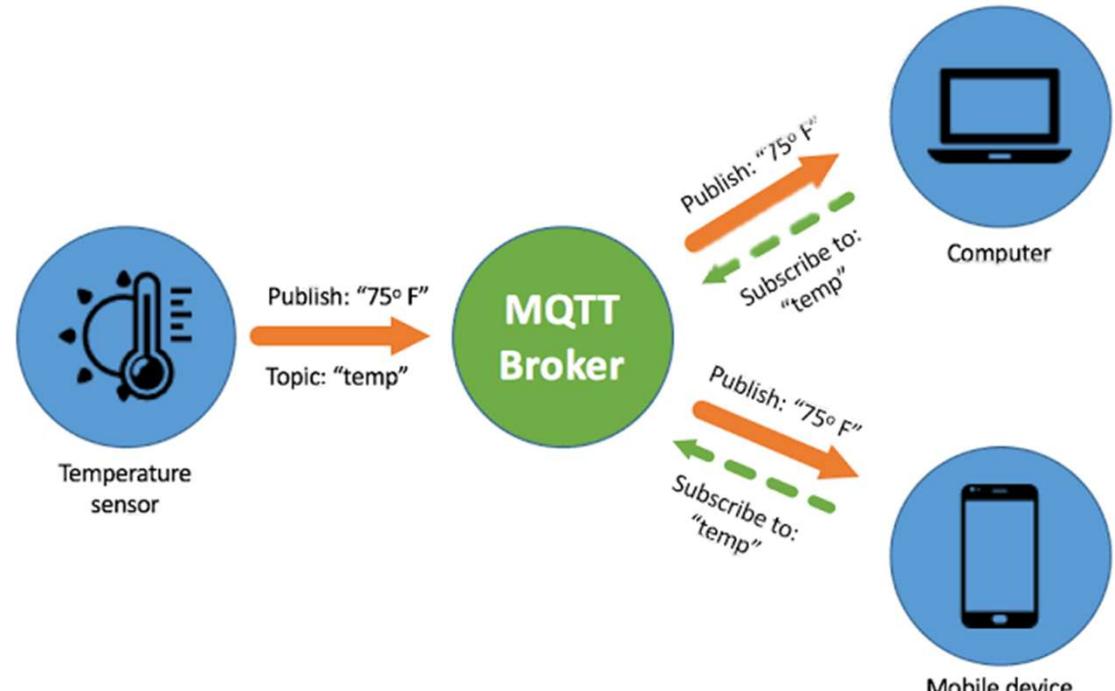


MQTT Protocol

NETPIE2020 communication [MQTT]

What is MQTT ??

- MQTT is a protocol designed to connect M2M or device to device, which is supported by IoT
- Use the principle of data transmission, Publish / Subscribe, similar to the principles used in the Web Service that requires a Web Server as an intermediary between users' computers.
- But MQTT uses an intermediary called Broker that manages the transmission order between devices and Publish / Subscribe mechanisms.



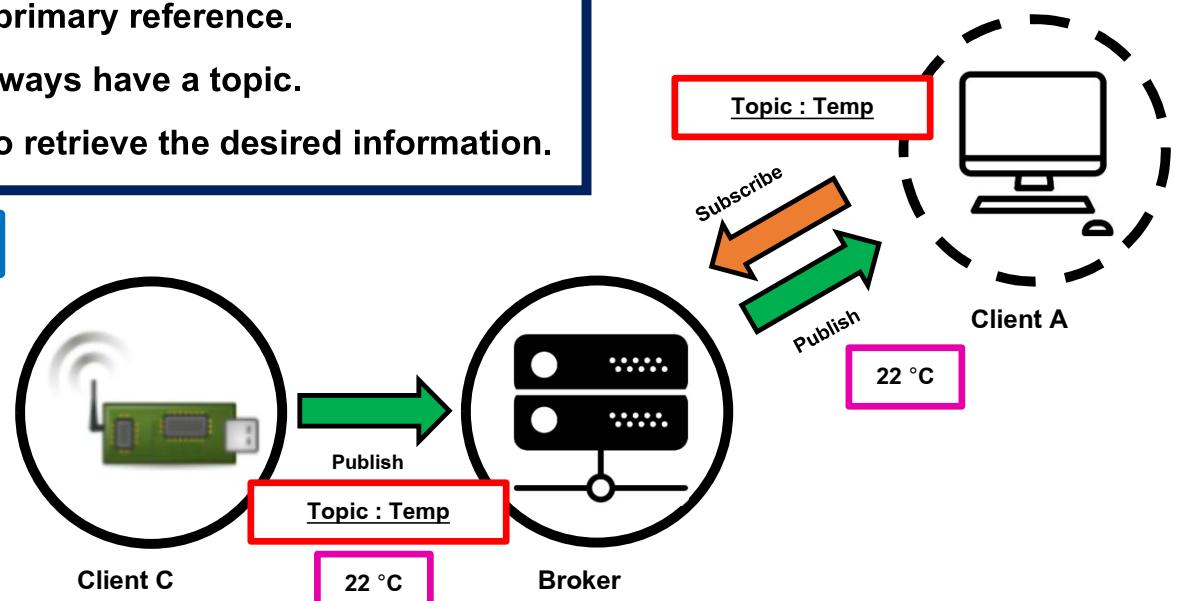
MQTT Protocol

transmissions on the MQTT have a topic as their primary reference.

- The data to be published to the Broker must always have a topic.
- On the subscribe end, it must refer to a topic to retrieve the desired information.

MQTT communication example

- Client C publishes information on a topic named **Temp**, while Client A subscribes to topic **Temp**.
- The data from Client C is a temperature with values equal to 22°C , so Client A receives the information.



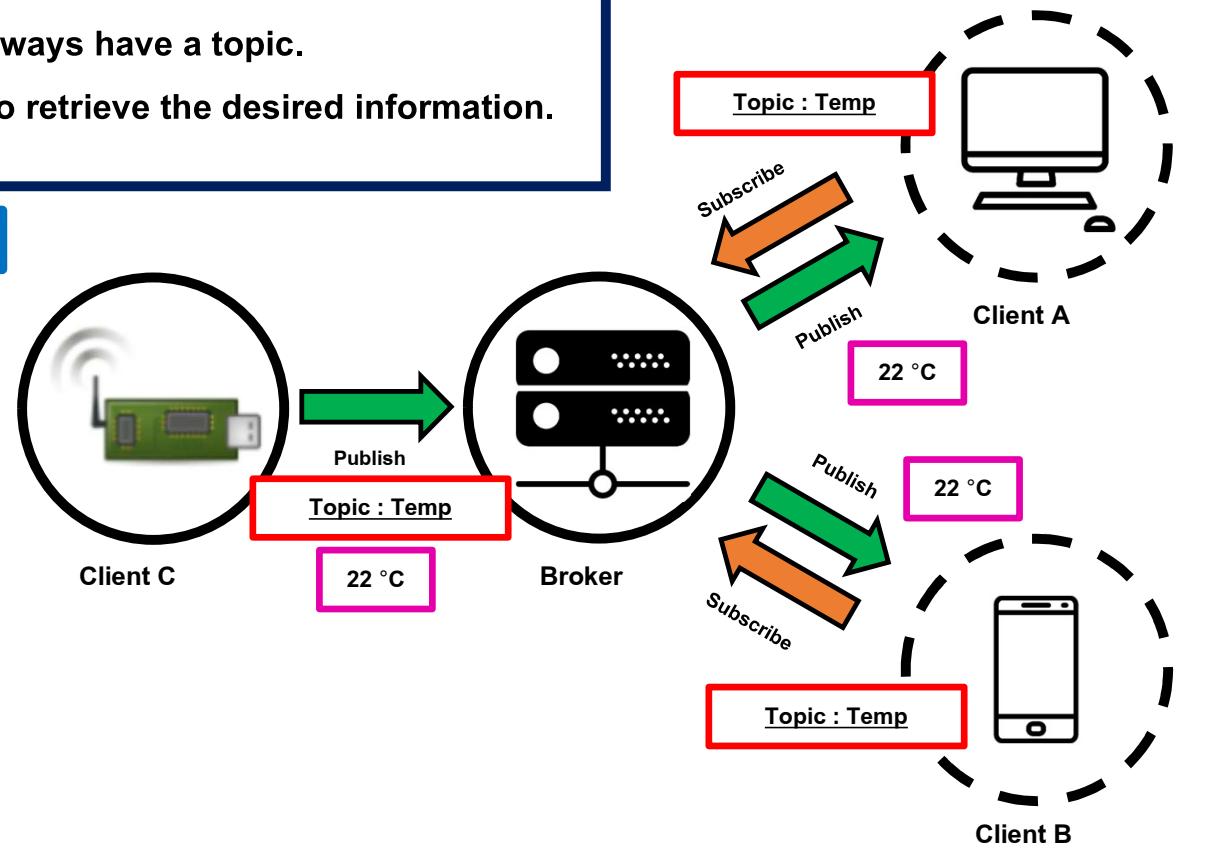
MQTT Protocol

transmissions on the MQTT have a topic as their primary reference.

- The data to be published to the Broker must always have a topic.
- On the subscribe end, it must refer to a topic to retrieve the desired information.

MQTT communication example

- Client C publishes information on a topic named **Temp**, while Client A subscribes to topic **Temp**.
- The data from Client C is a temperature with values equal to 22°C , so Client A receives the information.
- Later, Client B subscribes to topic **Temp** as well. The stored temperature is immediately sent to Client B after subscribing.



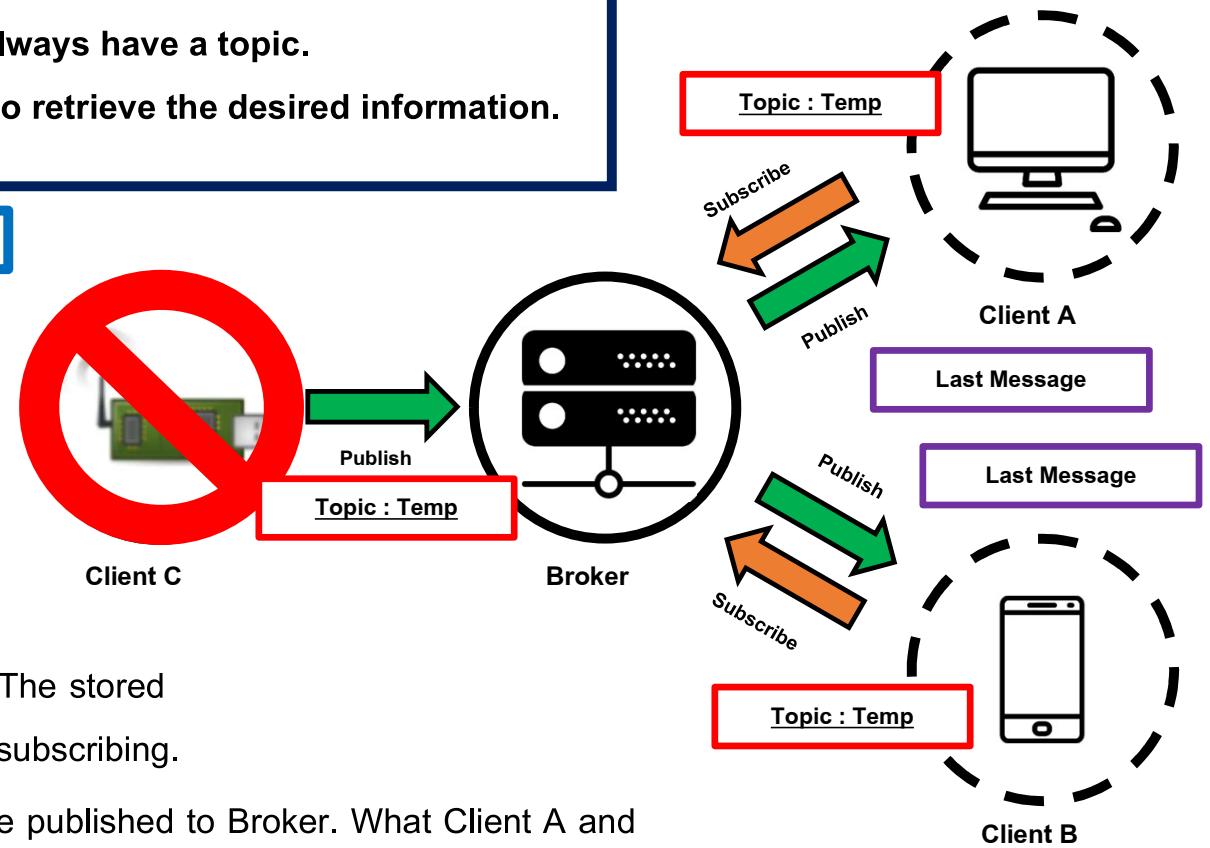
MQTT Protocol

transmissions on the MQTT have a topic as their primary reference.

- The data to be published to the Broker must always have a topic.
- On the subscribe end, it must refer to a topic to retrieve the desired information.

MQTT communication example

- Client C publishes information on a topic named **Temp**, while Client A subscribes to topic **Temp**.
- The data from Client C is a temperature with values equal to 22°C , so Client A receives the information.
- Later, Client B subscribes to topic **Temp** as well. The stored temperature is immediately sent to Client B after subscribing.
- But when Client C is disconnected, no data will be published to Broker. What Client A and B display is the last message sent by Client C



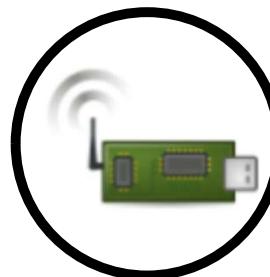
MQTT Protocol



There are two types of messages sent to NETPIE2020 via MQTT.

1

Message



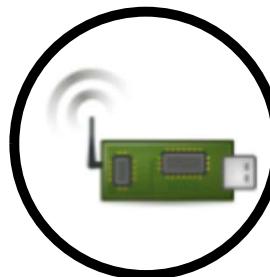
"Hello NETPIE2020"

MQTT Protocol



2

Data



MQTT Protocol
Temp = 25

MQTT Protocol

MQTT on NETPIE2020 has two types of transmission

1

Message

Send a message via MQTT by referring to a topic. The format is **@msg/topic**

2

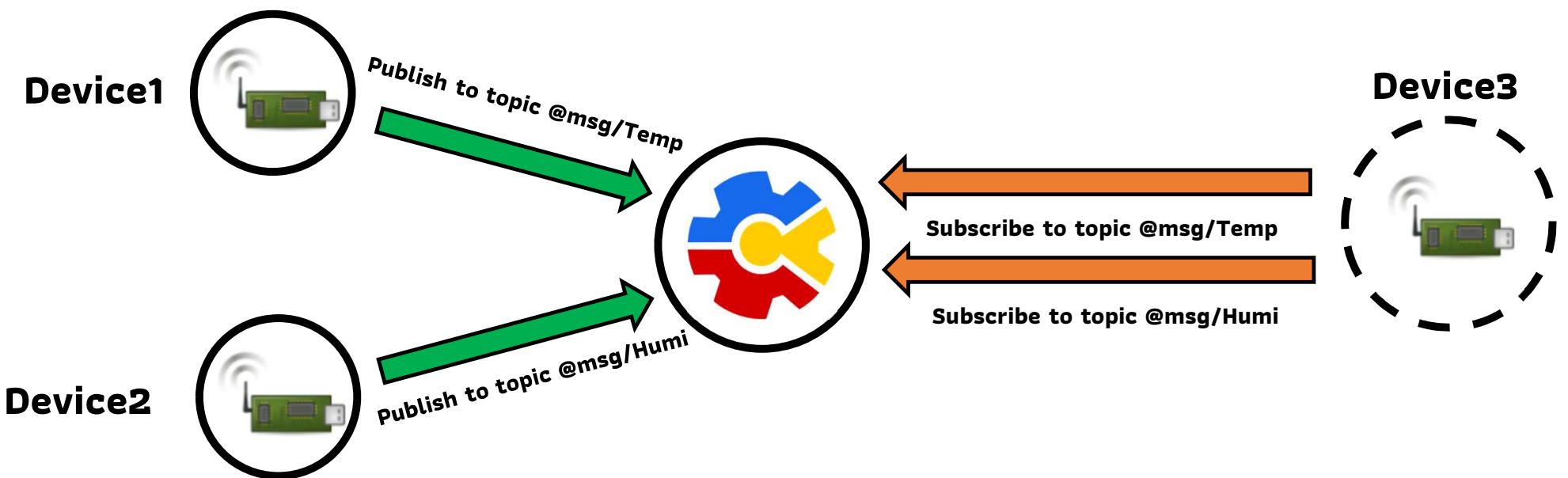
Shadow [Data]

Send message via MQTT to Device Shadow, to record latest message. The topic is **@shadow/data/update** with message in JSON format. For example,
{data:{temp:24}}

MQTT Protocol

Wildcard Topic

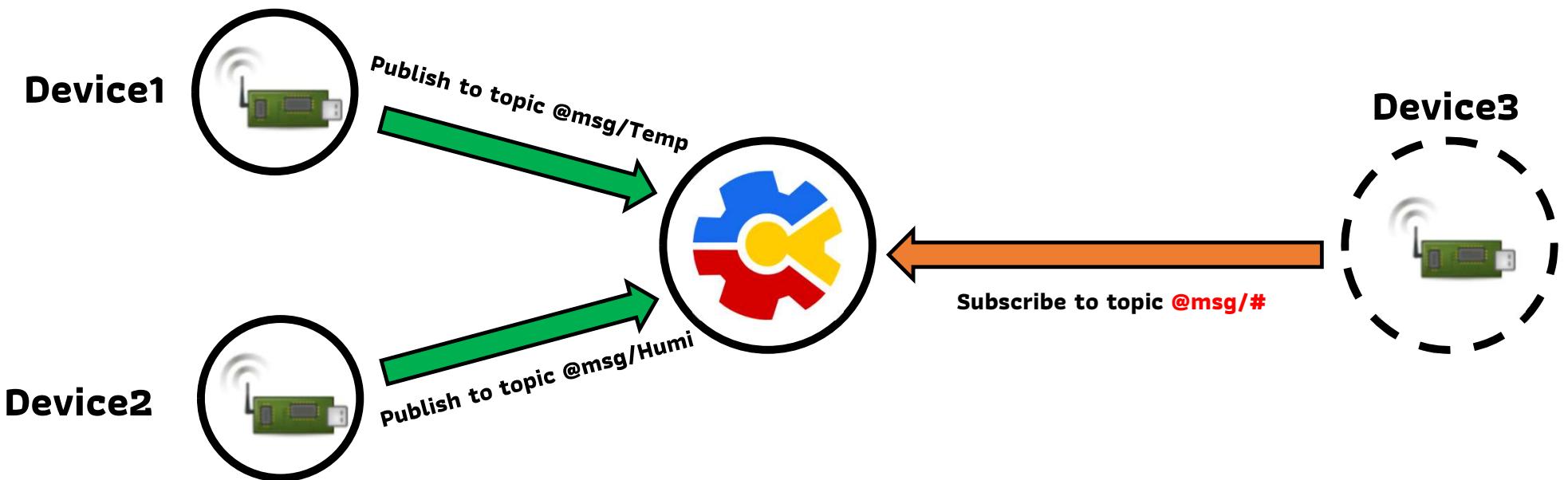
If you want Device 3 to receive messages from Device1 and Device2,
you need to subscribe to 2 topics



MQTT Protocol

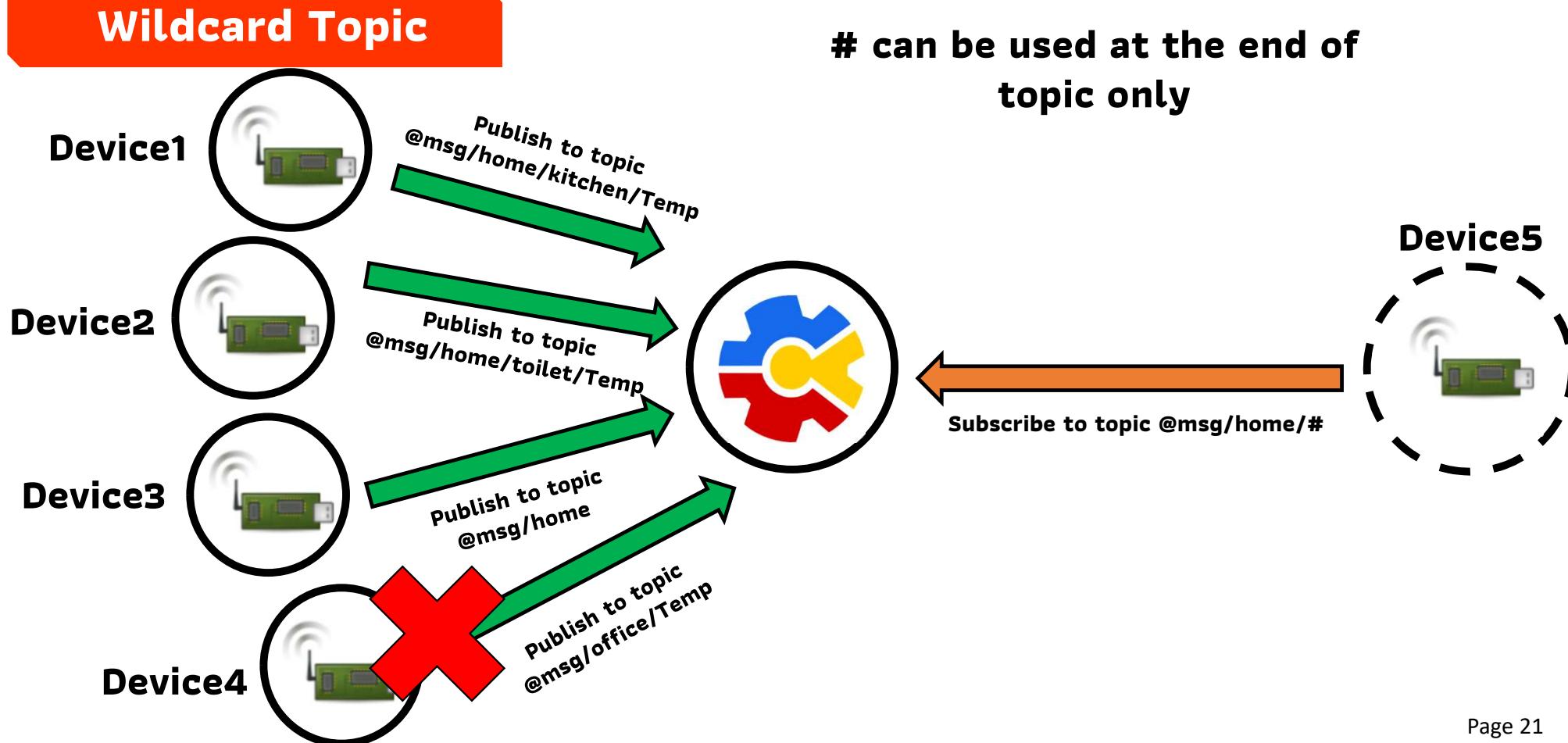
Wildcard Topic

The symbol # in part of a topic represents every word and all section of words within / after that position. If Subscribe topic / # is equivalent to subscribe all.



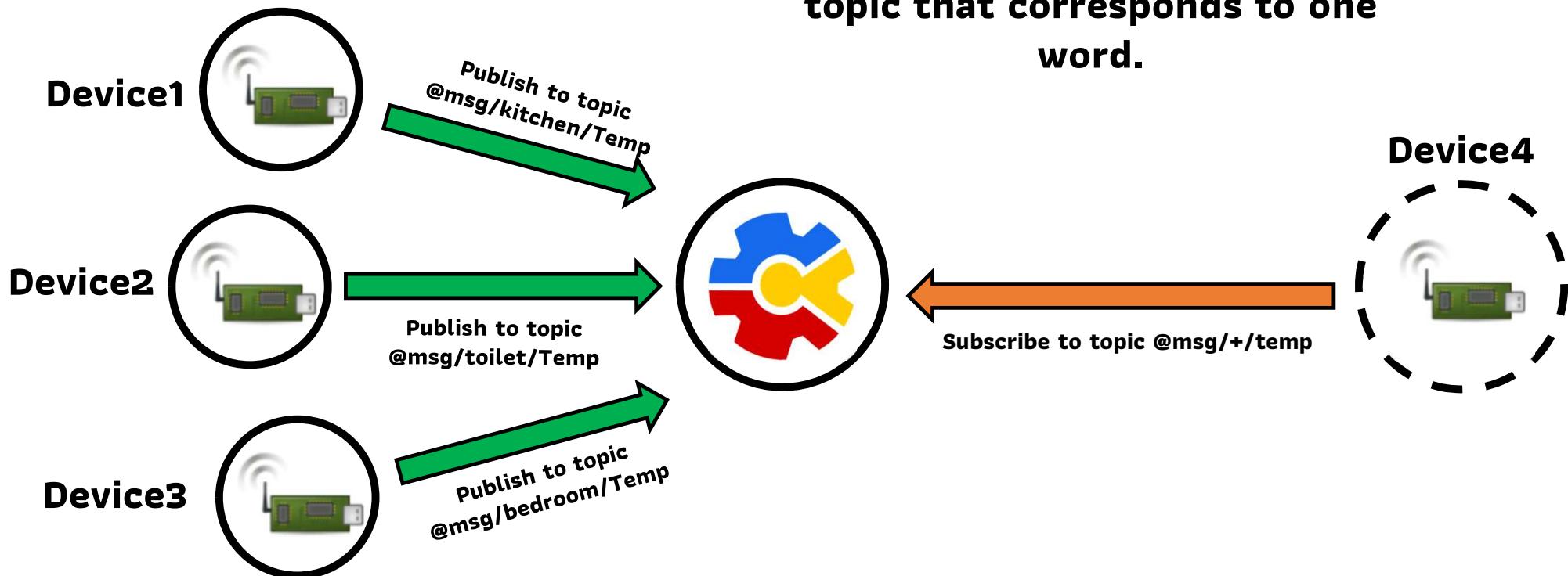
MQTT Protocol

Wildcard Topic



MQTT Protocol

Wildcard Topic



MQTT Protocol

MQTT Ports

Connection	Port
mqtt/tcp [broker.netpie.io]	1883
mqtts/tls [broker.netpie.io]	8883
ws [mqtt.netpie.io/mqtt]	80
wss [mqtt.netpie.io/mqtt]	443



NETPIE 2020 Portal



29 July 2024



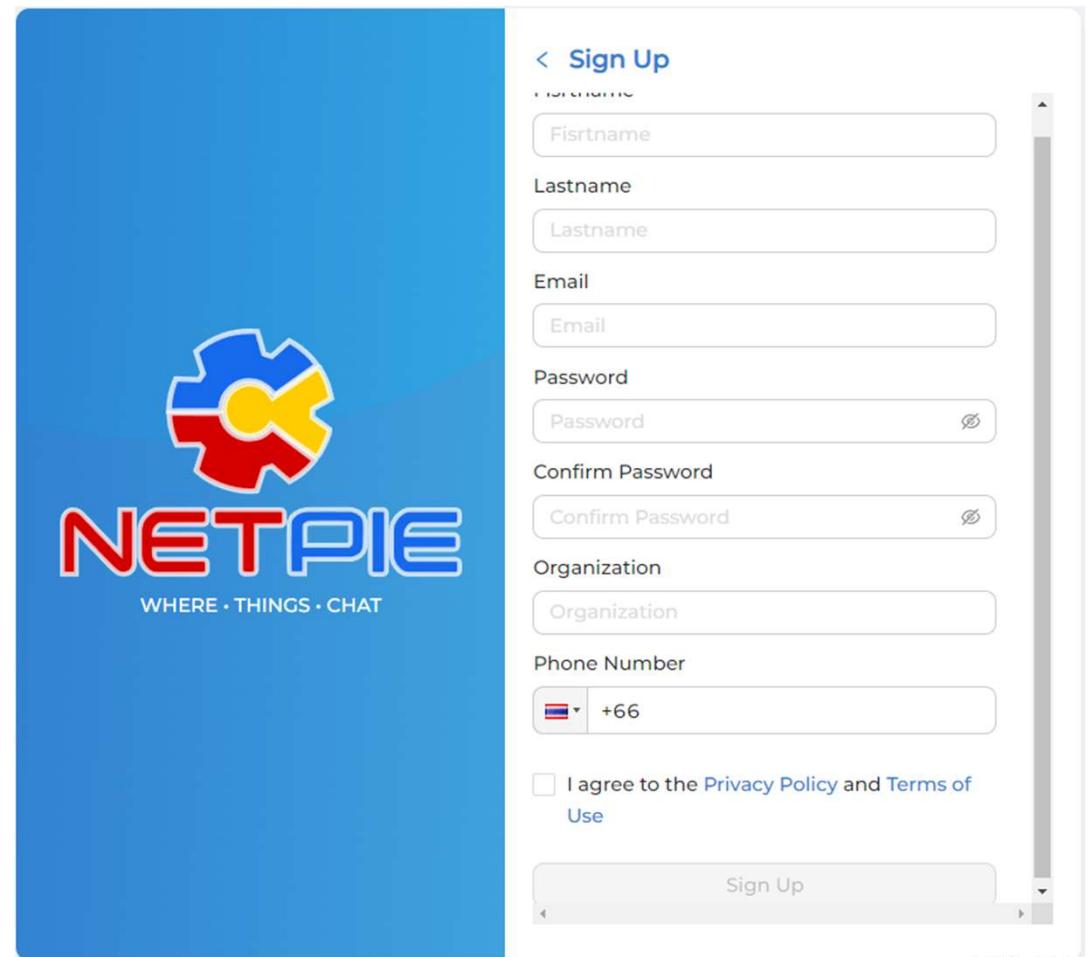
Page 24

NETPIE 2020 Portal

Getting started with
NETPIE2020

Create NETPIE2020 user
account at

<https://authx.netpie.io/signup>



The screenshot shows the "Sign Up" page for the NETPIE portal. At the top left is a back arrow labeled "< Sign Up". Below it are input fields for Firstname, Lastname, Email, and Password. There are also fields for Confirm Password, Organization, and Phone Number. A dropdown menu for the phone number country code shows "+66". At the bottom left is a checkbox for agreeing to the Privacy Policy and Terms of Use, followed by a large "Sign Up" button at the bottom right.

< Sign Up

Firstname

Lastname

Lastname

Email

Email

Password

Confirm Password

Organization

Organization

Phone Number

+66

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

Sign Up

NETPIE 2020 Portal

Getting started with
NETPIE2020

After sign up
NETPIE2020, login
With Username [email]
and Password

Or login at this link
<https://authx.netpie.io/login>



Welcome to NETPIE

Username

Username

Password

Password

[Forgot your password?](#)

Log In

[Don't have an account yet?](#)

[Register with NETPIE](#)

NETPIE 2020 Portal



Getting started with NETPIE2020



Varodom ▾

Ready to get started?

You haven't added anything yet.
Click the button below to add your first project.

+ Add Project

After login, you will see the window as shown

NETPIE 2020 Portal



NECTEC
a member of NSTDA

Create a Project

To create a project, click at [+] Add Project as shown in the picture



Varodom ▾

Ready to get started?

You haven't added anything yet.
Click the button below to add your first project.

+ Add Project



Create a Project

**Fill out the information for creating the project.
The Project Name field is required.**

Add Project X

Project Name

Project Description

Hashtag
[+ Add hashtag](#)

Project Tags
[+ Add tag](#)

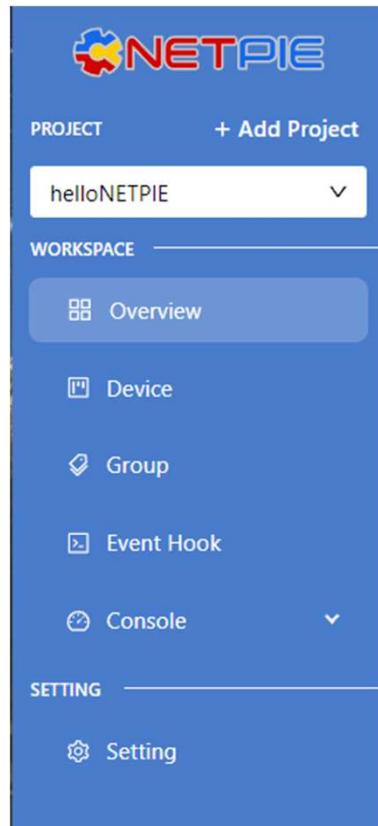
[CANCEL](#) [SAVE](#)

NETPIE 2020 Portal



Create a Project

After creating a project, you'll see the first page .



A screenshot of the NETPIE Project Overview page for the "helloNETPIE" project. The page shows the project ID (P273062830624) and a "Copy" button. It also displays statistics for Devices (0 devices, 0 online, 0 offline) and Groups (0 groups). The user "Varodom" is logged in.

NETPIE 2020 Portal

Create a Project

The screenshot shows the NETPIE 2020 Portal interface. On the left, a sidebar menu includes 'PROJECT' (selected), '+ Add Project', 'helloNETPIE' (selected), 'WORKSPACE' (Overview, Device, Group, Event Hook, Console), and 'SETTING' (Setting). The main area displays the 'helloNETPIE / overview' page. It shows the project name 'helloNETPIE' and creation date '2024-07-28'. A 'Detail' section contains the 'Project ID' P273062830624 with a 'Copy' button. To the right are three cards: 'Devices' (0 devices, 0 online, 0 offline), 'Groups' (0 groups), and a summary card with '0' for both devices and groups.

The section of the Overview page that displays all the information for the Project are:

- 1. Project details (name, ID, description)**
- 2. Devices (Device Online/Offline)**
- 3. Groups**

NETPIE 2020 Portal

Device Creation

The screenshot shows the NETPIE 2020 Portal interface. On the left, there is a sidebar with the following options:

- PROJECT: helloNETPIE (selected)
- + Add Project
- WORKSPACE:
 - Overview (marked with a red box and circled 1)
 - Device (marked with a red box and circled 1)
 - Group
 - Event Hook
 - Console
- SETTING:
 - Setting

The main content area is titled "Device" and shows a table with columns: ID, Name, Group, Status, and Created Time. There are currently 0 devices listed. A search bar at the top right contains "Varodom". A large blue button labeled "+ Create" is highlighted with a red box and circled 2. A hand cursor icon is pointing at this button.

To create a device, you will need to enter the Device List page, a window will appear like in the picture. After that, click [Create] to create a device.

NETPIE 2020 Portal

Device Creation

The screenshot shows the NETPIE 2020 Portal interface. On the left, there is a sidebar with sections for PROJECT, WORKSPACE, and SETTING. The WORKSPACE section has a sub-menu with 'Overview', 'Device' (which is selected), 'Group', 'Event Hook', and 'Console'. The SETTING section has a sub-menu with 'Setting'. In the center, there is a 'Device' list with a single item 'helloNETPIE / device'. A modal window titled 'Create Device' is open, prompting the user to fill out information. The 'Device Name' field is highlighted with a red border, and a hand cursor icon is pointing at the 'Description' text area. The 'Create Device' form includes fields for 'Device Description', 'Group', 'Hashtag', and 'Device Tags', each with an associated 'Add' button. At the bottom of the modal are 'CANCEL' and 'SAVE' buttons.

Fill out the information for creating the device. The “Device Name” field is required.

NETPIE 2020 Portal



Device Creation

A screenshot of the NETPIE 2020 Portal. The left sidebar shows 'PROJECT' with 'helloNETPIE' selected, 'WORKSPACE' with 'Overview' and 'Device' selected, 'GROUP' with 'Group', 'EVENT HOOK', 'CONSOLE', and 'SETTING' with 'Setting'. The main area shows 'helloNETPIE / device' with 'Device' selected. It displays 1 device: 'Device1' (My first device), ID: 331be25f-75d1-4041-8cac-c..., Status: Offline, Created Time: 2024-07-28 15:20. A red box highlights this row, and a hand cursor icon is positioned over it. Navigation buttons at the bottom include '<', '1', '>', and '10 / page'.

After creating the device, click the Device tab to view the device information.

NETPIE 2020 Portal

Device Creation

device / Device1

Device1
created date: 2024-07-28

Edit

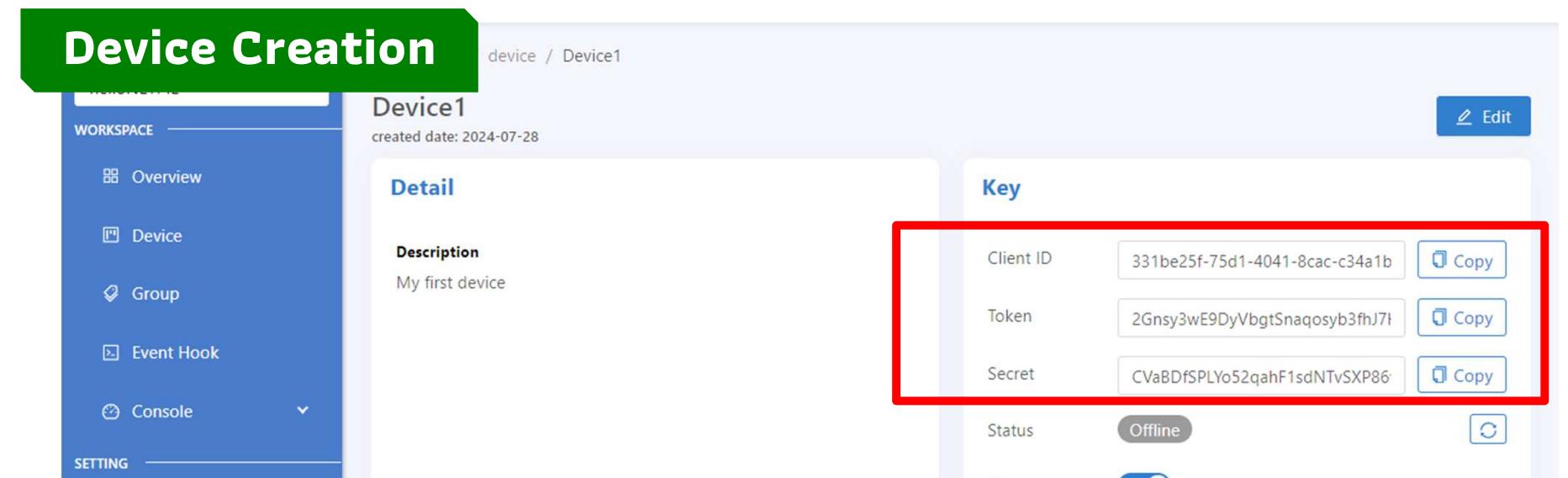
Detail

Description
My first device

Key

Client ID	331be25f-75d1-4041-8cac-c34a1b	Copy
Token	2Gnsy3wE9DyVbgtSnaqosyb3fhJ7t	Copy
Secret	CVaBDFSPLYo52qahF1sdNTvSXP86	Copy

Status Offline

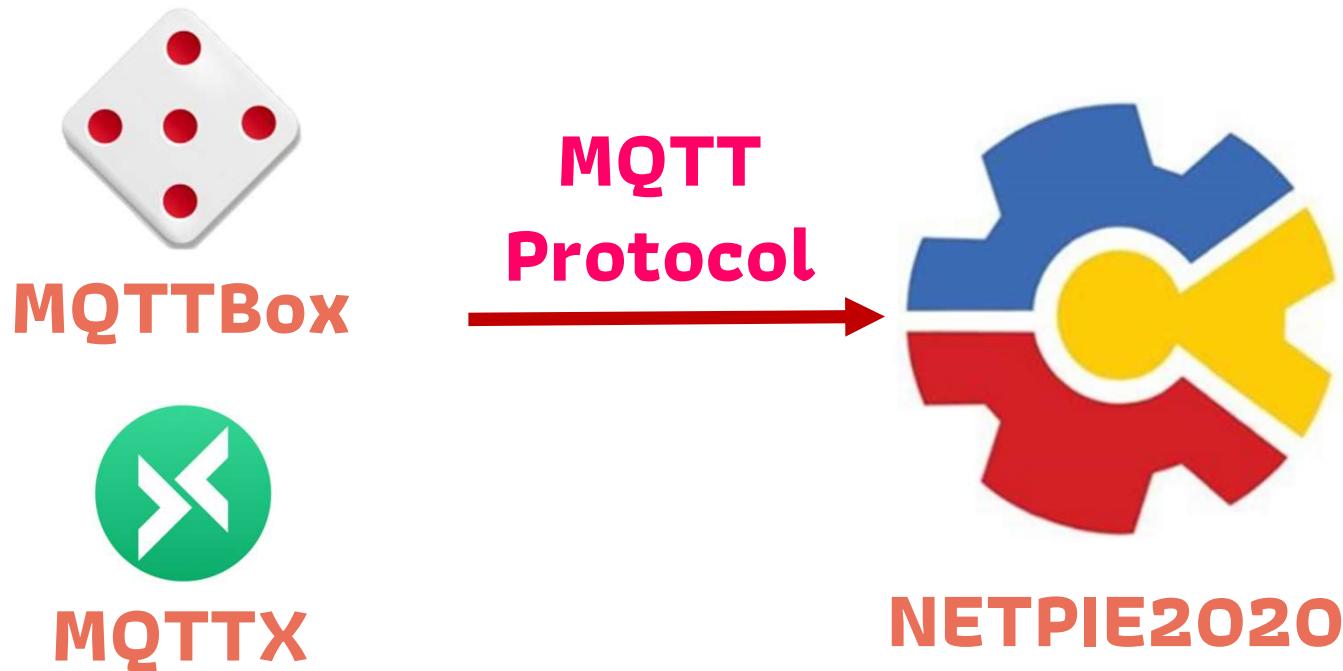


After entering the Device, you will find the details of the Device, that is

1. Client ID
2. Token
3. Secret

These 3 parameters are important to connect the device to NETPIE2020 using various protocols.

Workshop 1 : connecting NETPIE2020 with MQTT clients



Note : MQTTBox is no longer supported by Chrome. MQTTX, an MQTT client alternative, will be used in class. MQTTBox slides are kept for reference only.

Workshop 1 : connecting NETPIE2020 with MQTTBox

NETPIE2020 connection via MQTT, a small and popular protocol for M2M communication, can use any MQTT Library compatible with the active device.

MQTT connection requires 4 Parameters

1. Host : broker.netpie.io
2. Client ID : [Client ID of Device](#) created in NETPIE2020
3. Username : [Token of Device](#) created in NETPIE2020
4. Password : [Secret of Device](#) created in NETPIE2020 [used for more identity verification]

The screenshot shows a table with three rows of generated keys:

Key
Client ID
Token
Secret

Each row contains a value and a 'Copy' button. Red arrows point from the labels 'Client ID', 'Username', and 'Password' to the 'Client ID', 'Token', and 'Secret' columns respectively.

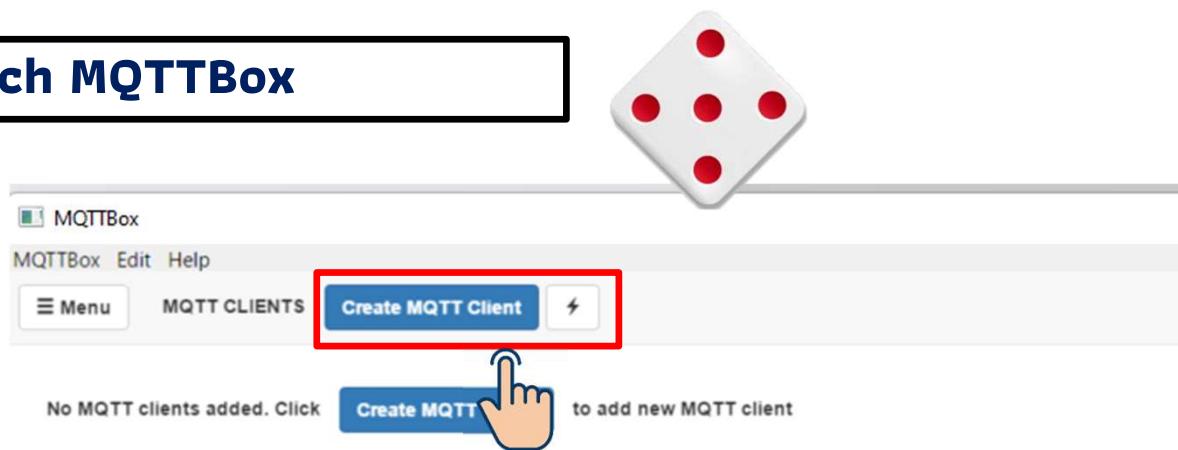
Key
Client ID
Token
Secret

Values:

- Client ID: 331be25f-75d1-4041-8cac-c34a1b
- Token: 2Gnsy3wE9DyVbgtSnaqosyb3fhJ7I
- Secret: CVaBDfSPLYo52qahF1sdNTvSXP86

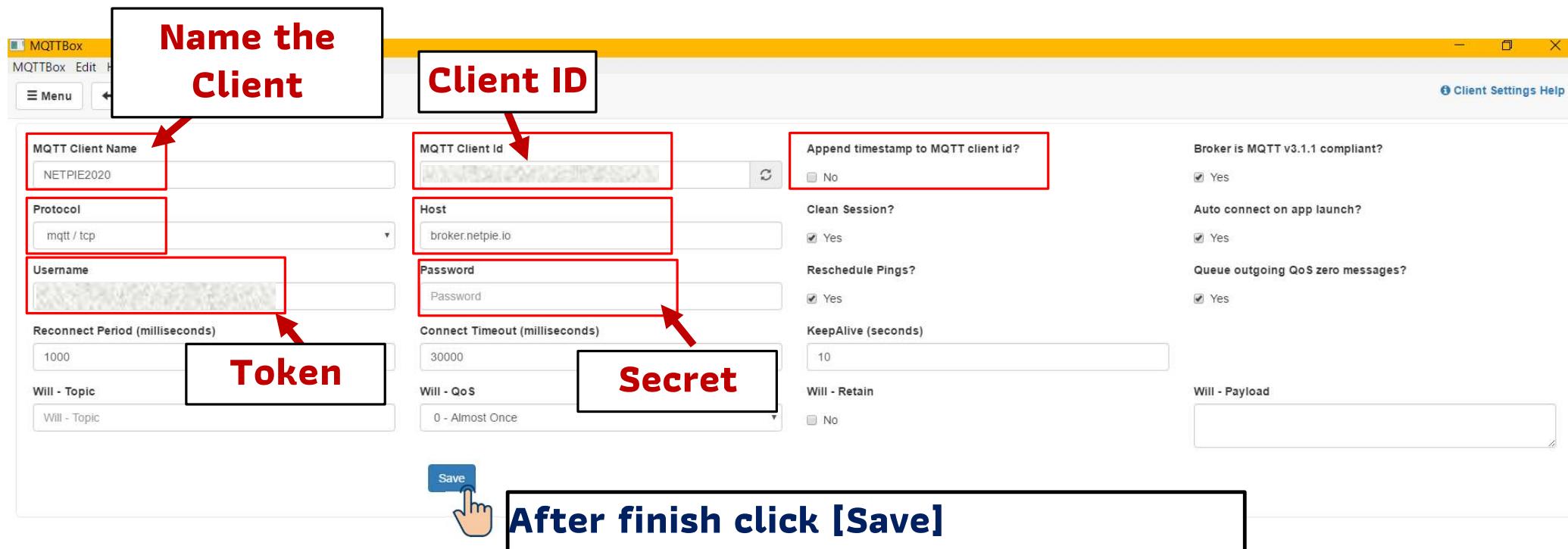
Workshop 1 : connecting NETPIE2020 with MQTTBox

1. Launch MQTTBox



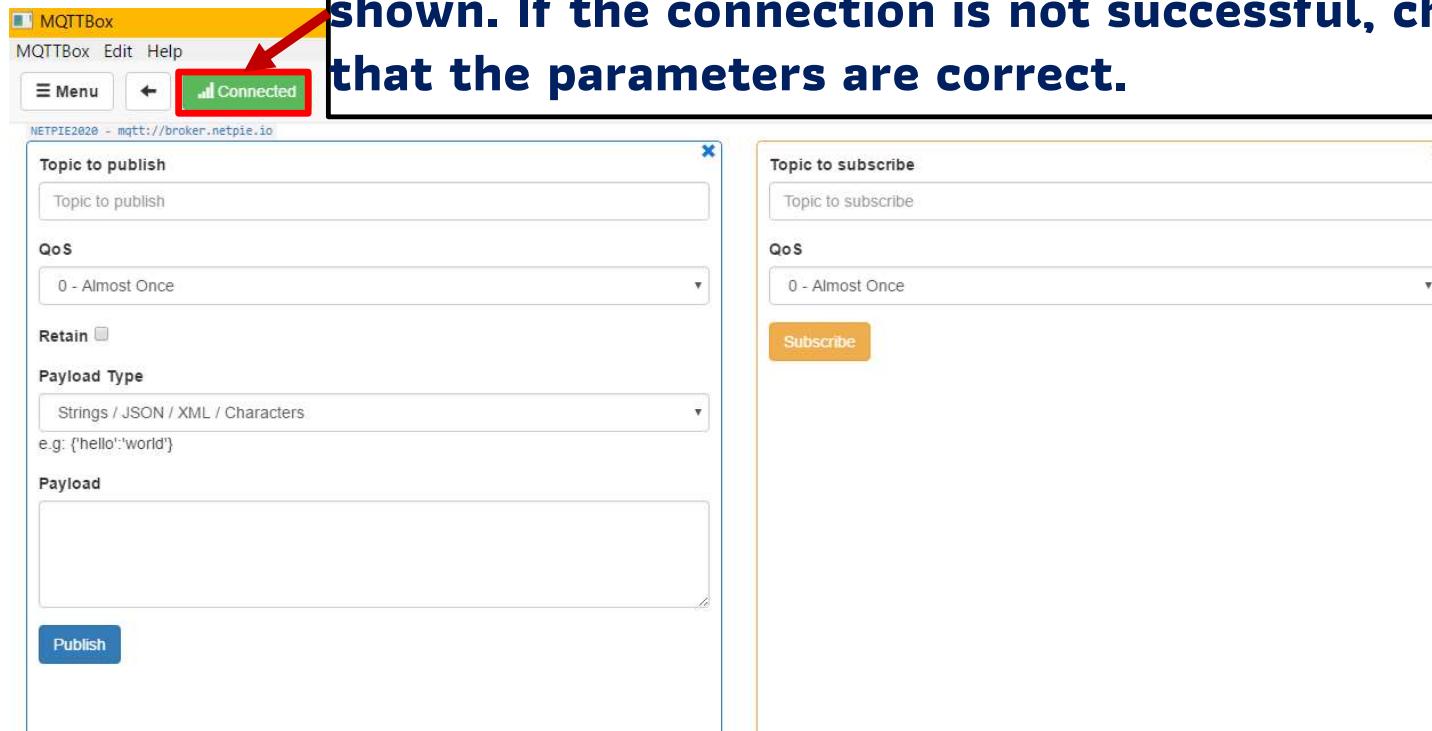
2. Click on [Create MQTT Client]

Workshop 1 : connecting NETPIE2020 with MQTTBox



Workshop 1 : connecting NETPIE2020 with MQTTBox

When connected successfully, it will display the results as shown. If the connection is not successful, check to make sure that the parameters are correct.



Workshop 1 : connecting NETPIE2020 with MQTTBox

The screenshot shows the NETPIE Portal interface. On the left, the sidebar has 'PROJECT' set to 'helloNETPIE'. The 'WORKSPACE' section includes 'Overview', 'Device' (selected), 'Group', 'Event Hook', and 'Console'. The main area displays 'Device1' created on 2024-07-28. The 'Detail' tab shows a 'Description' of 'My first device'. To the right, a 'Key' section lists 'Client ID', 'Token', 'Secret', and 'Status' (set to 'Online'). A red box highlights the 'Status' button, and a red arrow points from it to a callout box.

Can also check the device status on NETPIE Portal

29 July 2024 Page 41

Workshop 1 : connecting NETPIE2020 with MQTTBox

After connection is established, test by sending message to
NETPIE2020

MQTT message is referenced by Topic : **@msg/topic**

The screenshot shows the MQTTBox application interface. At the top, there's a yellow header bar with the title 'MQTTBox'. Below it, the main window has two main sections: 'Topic to publish' on the left and 'Topic to subscribe' on the right. In the 'Topic to publish' section, there's an input field where the topic '@msg/test' is entered. A red arrow points to this input field. Below the input field are sections for 'QoS' (set to '0 - Almost Once'), 'Retain' (checkbox), 'Payload Type' (set to 'Strings / JSON / XML / Characters'), and a payload example 'e.g. {"hello": "world"}'. At the bottom of this section is a blue 'Publish' button. To the right, the 'Topic to subscribe' section has an input field for the topic, a 'QoS' dropdown set to '0 - Almost Once', and a large orange 'Subscribe' button.

Workshop 1 : connecting NETPIE2020 with MQTTBox

The screenshot shows the MQTTBox application interface. On the left, the 'Publish' section is visible, containing fields for 'Topic to publish' (@msg/test), 'QoS' (0 - Almost Once), 'Retain' (unchecked), 'Payload Type' (Strings / JSON / XML / Characters), and a 'Payload' text area. A 'Publish' button is at the bottom. On the right, the 'Subscribe' section is shown, featuring a 'Topic to subscribe' field (@msg/test) with a red box around it, a 'QoS' dropdown (0 - Almost Once), and a 'Subscribe' button with a hand cursor icon. A large callout box points to the 'Topic to subscribe' field with the text: 'Before publishing a message, it will subscribe to the topic that will be sent first.' Another callout box points to the 'Subscribe' button with the text: 'Specify the Topic and click [Subscribe]'.

Before publishing a message, it will subscribe to the topic that will be sent first.

Specify the Topic and click [Subscribe]

Workshop 1 : connecting NETPIE2020 with MQTTBox

The screenshot shows the MQTTBox interface for publishing messages to a broker. On the left, the configuration panel includes:

- Topic to publish:** @msg/test
- QoS:** 0 - Almost Once
- Retain:** (checkbox)
- Payload Type:** Strings / JSON / XML / Characters
- Payload:** Hello NETPIE2020

A red box highlights the **Payload** field, and a red arrow points from it to a callout box containing the instruction: **Type the message and click [Publish]**. A hand icon is shown clicking the **Publish** button at the bottom-left of the payload area.

On the right, the message is displayed in the message list:

- * @msg/test

Workshop 1 : connecting NETPIE2020 with MQTTBox

The screenshot shows the MQTTBox interface with two main panels. On the left, the 'Publisher' panel is active, displaying fields for Topic (@msg/test), QoS (0 - Almost Once), Retain, Payload Type (Strings / JSON / XML / Characters), and Payload (Hello NETPIE2020). A 'Publish' button is at the bottom. The right panel shows the 'Subscribers' list with one entry: '@msg/test'. The message details for this subscriber show the topic, qos, retain, cmd, dup, topic, messageId, length, and raw payload. A red box highlights the message content, and a red arrow points from a callout box below it to the highlighted area. The callout box contains the text: 'Message appears at receiving end'.

Topic to publish: @msg/test

QoS: 0 - Almost Once

Retain

Payload Type: Strings / JSON / XML / Characters

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

Payload: Hello NETPIE2020

Publish

Hello NETPIE2020
topic:@msg/test, qos:0, retain:false

Subscribers: @msg/test

Message details: @msg/test
Hello NETPIE2020
qos:0, retain:false, cmd: publish, dup: false, topic: @msg/test, messageId: 1, length: 27.
Raw payload: 721011081081113278698480736950485048

Message appears at receiving end

Workshop 1 : connecting NETPIE2020 with MQTTX

Name the Client

New

Client ID

Connect 

* Name device1

* Client ID 331be25f-75d1-4041-8cac-c34a1b26b084

* Host mqtt:// broker.netpie.io

* Port 1883

Username 2Gnsy3wE9DyVbgtSnaqosyb3fhJ7HkW

Password *****

SSL/TLS

Token

Secret

Advanced ▲

MQTT Version 3.1.1 **Changed to 3.1.1**

Connect Timeout 10 (s)

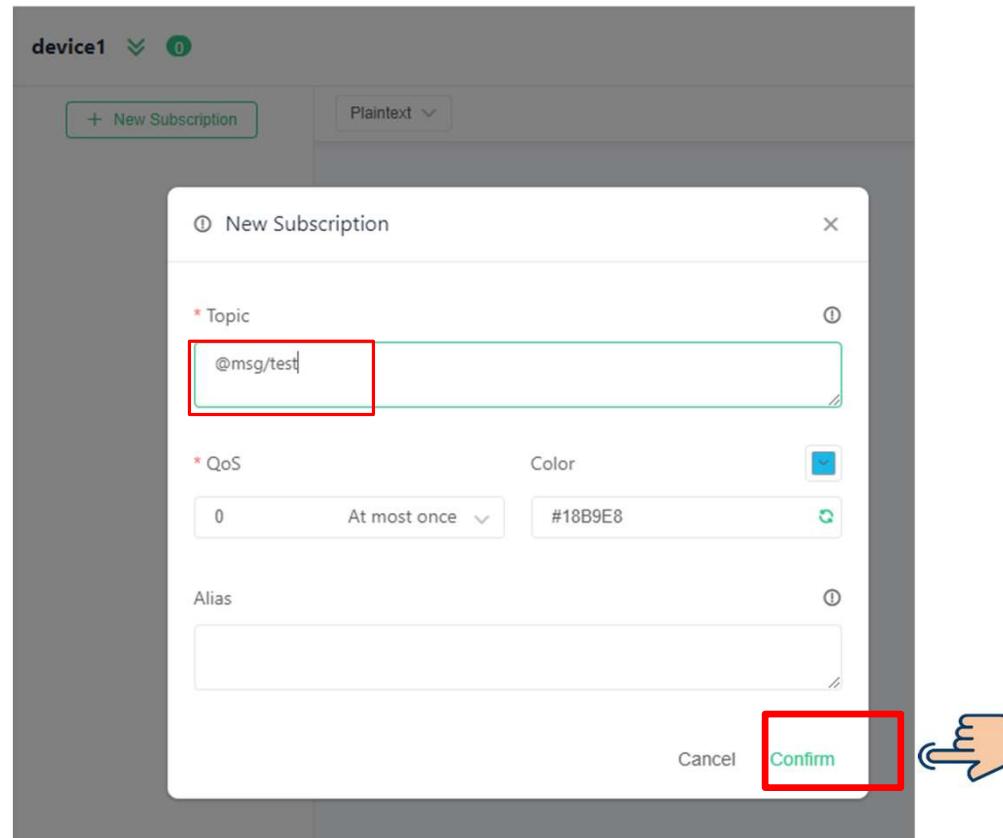
Keep Alive 60 (s)

Auto Reconnect

Reconnect Period 4000 (ms)

Clean Session

Workshop 1 : connecting NETPIE2020 with MQTTX



Workshop 1 : connecting NETPIE2020 with MQTTX

The screenshot shows the NETPIE Portal interface for managing MQTT connections. On the left, a sidebar lists a connection named "device1@broker.netpie...". The main area displays two messages sent to the topic "@msg/test" with QoS 0:

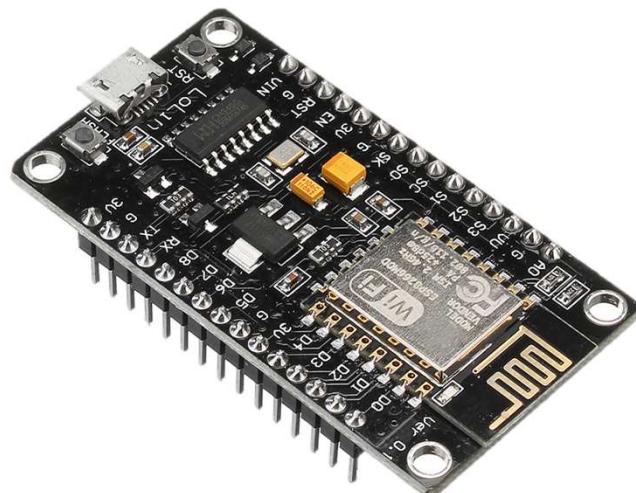
- Message 1:** Topic: @msg/test QoS: 0
Hello NETPIE 2020
2024-07-29 08:23:32:639
- Message 2:** Topic: @msg/test QoS: 0
Hello NETPIE 2020
2024-07-29 08:23:44:638

Below the messages, there is a configuration section with the following settings:

- Plaintext dropdown menu (selected)
- QoS 0 dropdown menu
- Retain checkbox (unchecked)
- Meta checkbox (unchecked)
- Up and down arrows for message order

At the bottom of the configuration section, the message "Hello NETPIE 2020" is entered into a text input field, which is highlighted with a red border. To the right of the input field are three small green circular buttons with icons: a left arrow, a minus sign, and a right arrow. At the very bottom right is a green circular button with a white checkmark icon.

Workshop 2 : Connect NETPIE2020 with NodeMCU



NodeMCU

MQTT
Protocol



NETPIE2020

Workshop 2 : Connect NETPIE2020 with NodeMCU

PROJECT + Add Project

helloNETPIE / device

WORKSPACE Overview Device Group Event Hook Console

SETTING Setting

Device

Show 2 devices

+ Create input search text

Manage Device

ID	Name	Group	Status	Created Time
adb59c76-3be2-4d83-93b4...	esp8266	-	Offline	2024-07-28 15:50
331be25f-75d1-4041-8cac-c...	Device1 My first device	-	Online	2024-07-28 15:20

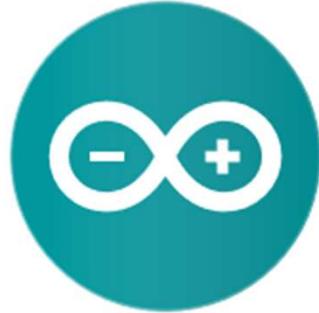
Create new device for NodeMCU

Module 2 : NETPIE 2020 Portal

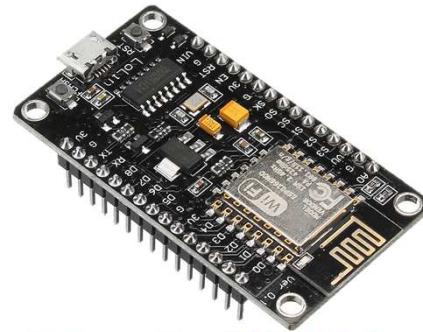


Workshop 2 : Connect NETPIE2020 with NodeMCU

In order for NodeMCU to connect to NETPIE2020, it needs to be programmed on the Arduino IDE to connect NodeMCU to internet and then to NETPIE2020.



Arduino IDE



NodeMCU

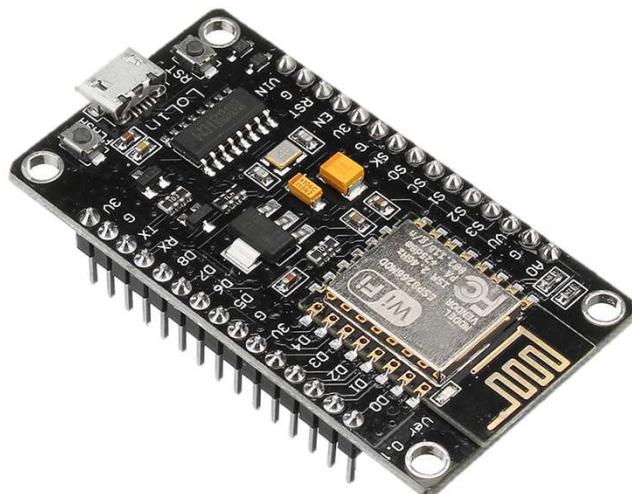
Internet
connection
through
WiFi



NETPIE2020

Workshop 2 : Connect NETPIE2020 with NodeMCU

In order for NodeMCU to connect to NETPIE2020, the following libraries are needed



NodeMCU

1. **ESP8266WiFi**

Used to allow NodeMCU to connect to the Internet via WiFi network.

2. **PubSubClient**

Used for NodeMCU to connect and communicate on NETPIE 2020 platform. (MQTT Protocol)

Workshop 2 : Connect NETPIE2020 with NodeMCU

Key command and functions in ESP8266WiFi

WiFi.begin()

WiFi.begin : function for setup Wi-Fi Library and network, with status as results Usage example:

WiFi.begin(); ssid is the Wi-Fi network to connected to

WiFi.begin(ssid); pass is the wifi password WPA encrypted

WiFi.begin(ssid ,pass);

WiFi.status

WiFi.status is a function to check connection with the wifi network. Usage example :

WiFi.status() != WL_CONNECTED

Results from WiFi.begin()

WL_CONNECTED : successful connection

WL_IDLE_STATUS : ready but not connected

Results from WiFi.status()

WL_CONNECTED : successful connection

WL_NO_SHIELD : Wi-Fi Shield not found

WL_DILE_STATUS : ready but not connected. This is a temporary status while connection is attempted.

WL_NO_SSID_AVAIL : no SSID to connect

WL_SCAN_COMPLETED : network scan finished

WL_CONNECT_FAILED : connection not successful

WL_CONNECTION_LOST : lost the connection

WL_DISCONNECTED : disconnected

Workshop 2 : Connect NETPIE2020 with NodeMCU

Important commands & functions in PubSubClient

void callback

callback is a function for receiving payloads or messages sent by the subscribed Topic.

void reconnect

reconnect is a function to connect to the MQTT Server that has been set up. Or when there is a disconnection event with the MQTT Server, the reconnect function will be invoked to reconnect.

client.connect()

client.connect() : command to connect to MQTT Broker.
Format is

client.connect[client.username, password]

client.setServer()

client.setServer() : command to setup MQTT Server.
Format is

client.setServer[mqtt_server, mqtt_port]
client.publish()

client.publish() : to publish with specified topic
client.publish["topic", "Message"]

client.subscribe()

client.subscribe to subscribe specified Topic.
Often placed in reconnect function. Format is
client.subscribe["topic"]

Workshop 2 : Connect NETPIE2020 with NodeMCU

Open Workshop2.ino

Coding in Workshop2.ino consists of 3 sections

1

Section 1 : Library and variable declaration

Call to various libraries.

Variable declaration for connecting to WiFi and for NETPIE2020 connection

Create instances for NETPIE2020 connection

Global variables

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

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

WiFiClient espClient;
PubSubClient client[espClient];

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

Workshop 2 : Connect NETPIE2020 with NodeMCU

2

Section 2 : functions

Functions to connect to MQTT

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

Enter NETPIE2020 MQTT connection.

- If connected successfully, it will display "connected".
- If the connection is not successful, it will display "failed..." and automatically reconnect

Workshop 2 : Connect NETPIE2020 with NodeMCU

2

Section 2 : functions

```
void setup() {  
    Serial.begin(115200);  
    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);  
}
```

Performing initialization

In this function, it will connect to WiFi and NETPIE2020 according to the settings

Workshop 2 : Connect NETPIE2020 with NodeMCU

3

Section 3 : loop() function

```
void loop() {
```

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

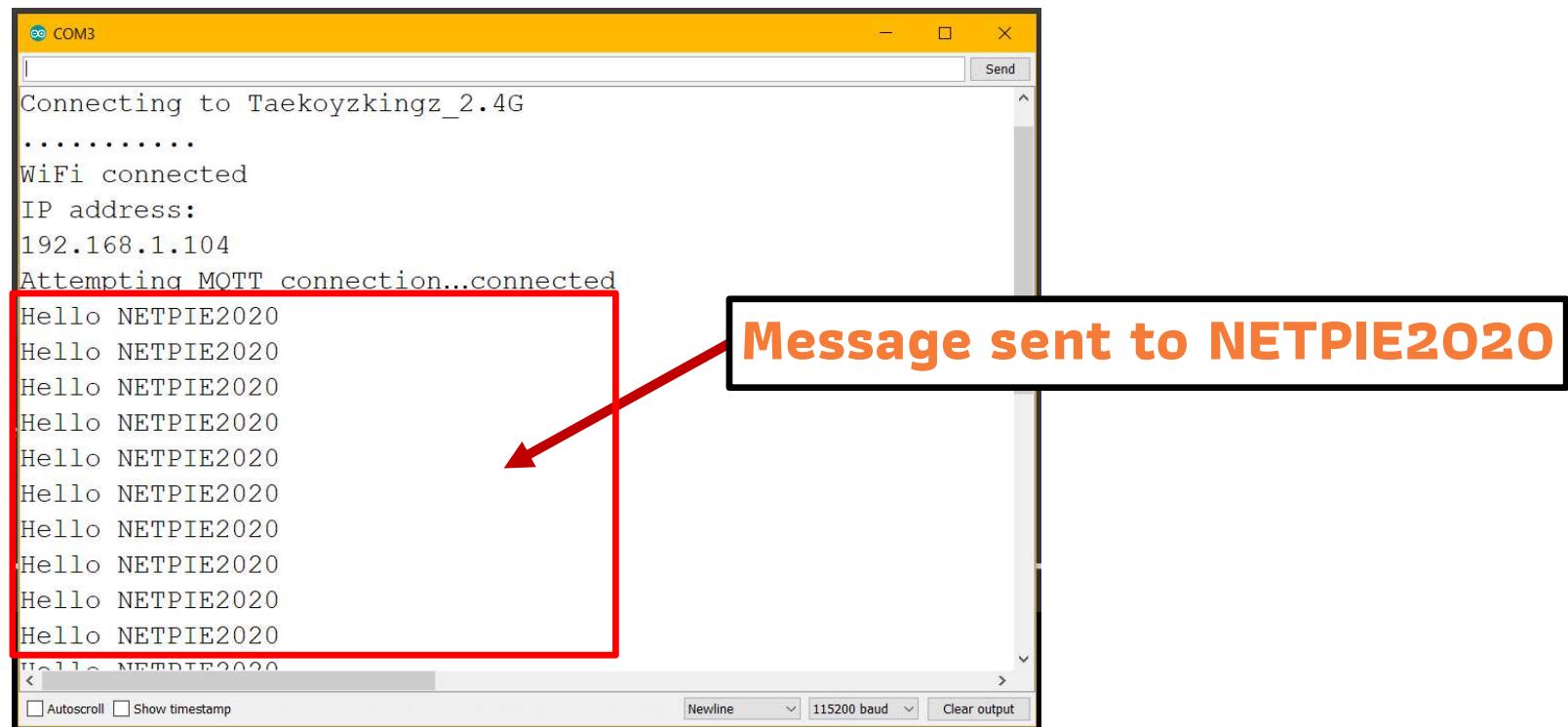
```
    long now = millis();  
    if (now - lastMsg > 2000) {  
        lastMsg = now;  
        ++value;  
        client.publish("@msg/test", "Hello NETPIE2020");  
        Serial.println("Hello NETPIE2020");  
    }  
    delay(1);  
}
```

Main function that runs iteratively

Command set that maintains the connection status and functions of NETPIE2020.

Commands to send message “Hello NETPIE2020” to Topic : @msg/test every 2 seconds

Workshop 2 : Connect NETPIE2020 with NodeMCU



Workshop 2 : Connect NETPIE2020 with NodeMCU

The screenshot shows the NETPIE Portal interface. On the left, there's a sidebar with sections for PROJECT (+ Add Project), WORKSPACE (helloNETPIE selected), and SETTINGS (Console, Setting). The main area displays a device named "esp8266" created on 2024-07-28. The "Detail" section is visible, and the "Key" section shows Client ID, Token, Secret, and Status. The Status button is highlighted with a red box and a red arrow points from the "Check NodeMCU status on NETPIE Portal" callout to it.

Key	Value	Action
Client ID	adb59c76-3be2-4d83-93b4-f75da...	<button>Copy</button>
Token	A5pGYWRFXjkkqorVLM95Qn4BUb	<button>Copy</button>
Secret	5Uz6sUqUPiHFPH7gWPP1uFWS6b	<button>Copy</button>
Status	Online	<button>Online</button>

Check NodeMCU status on NETPIE Portal

Displaying message NodeMCU sent to NETPIE2020

To display message from NodeMCU can be done by
bringing another Device/App to connect vis MQTT Protocol
And subscribing topic published by NodeMCU
Using same Client-ID, Token and Secret



NodeMCU

Authentication

Client-ID : 13456789

Token : abcdefghijk

Secret : lmnopqrstuvwxyz

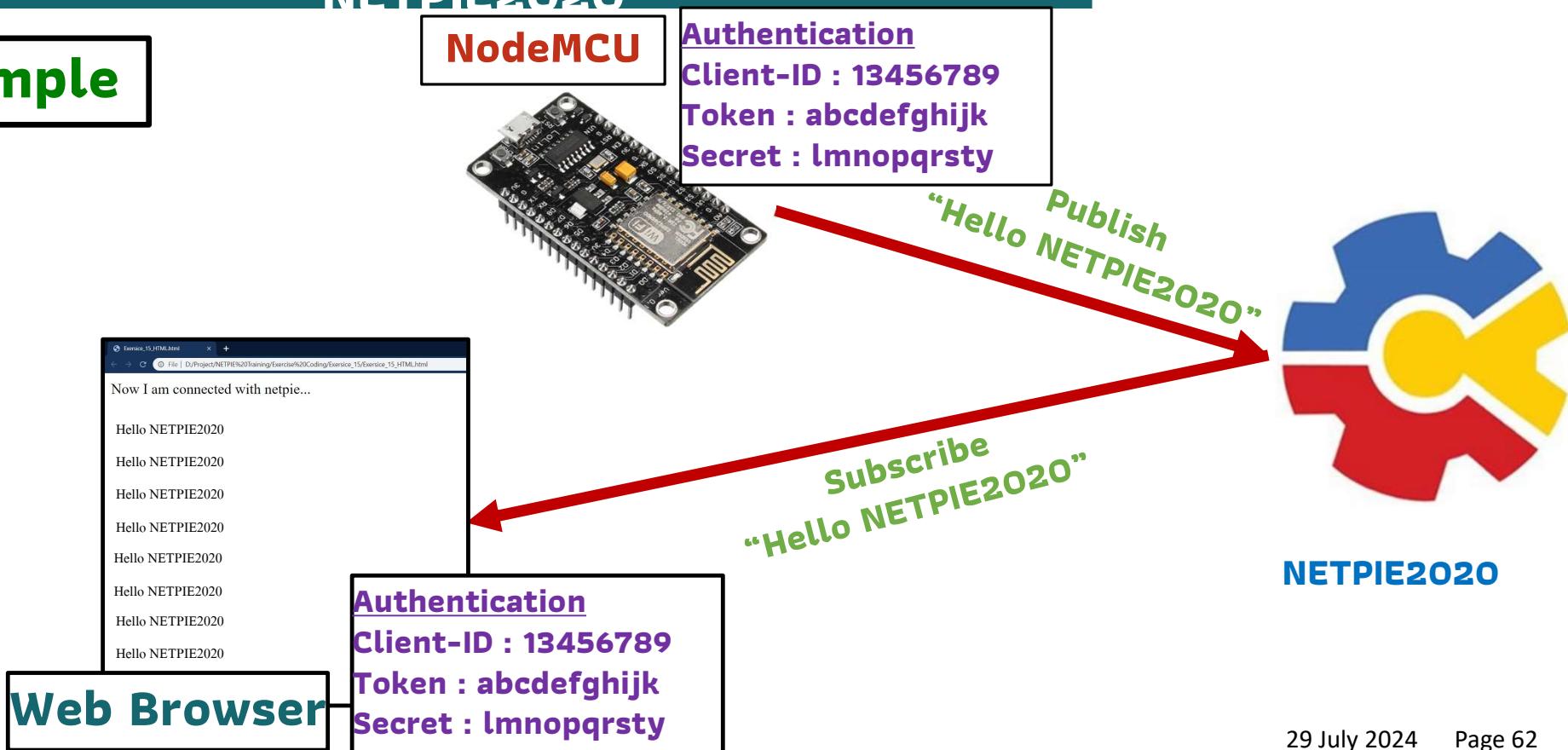
→ Publish
“Hello NETPIE2020”



NETPIE2020

Displaying message NodeMCU sent to NETPIE2020

Example



NETPIE Portal



Displaying message NodeMCU sent to NETPIE2020

An example to connect Web Browser with NETPIE2020 via MQTT Protocol using same Client-ID, Token, Secret as NodeMCU This in effect prevents the Web Browser from connecting with NETPIE2020 since the Client-ID, Token, Secret are already used

NodeMCU

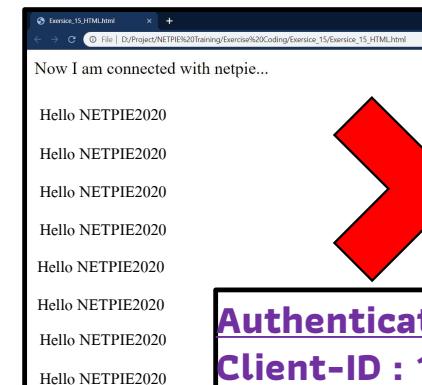


Authentication

Client-ID : 13456789

Token : abcdefghijk

Secret : lmnopqrstuvwxyz



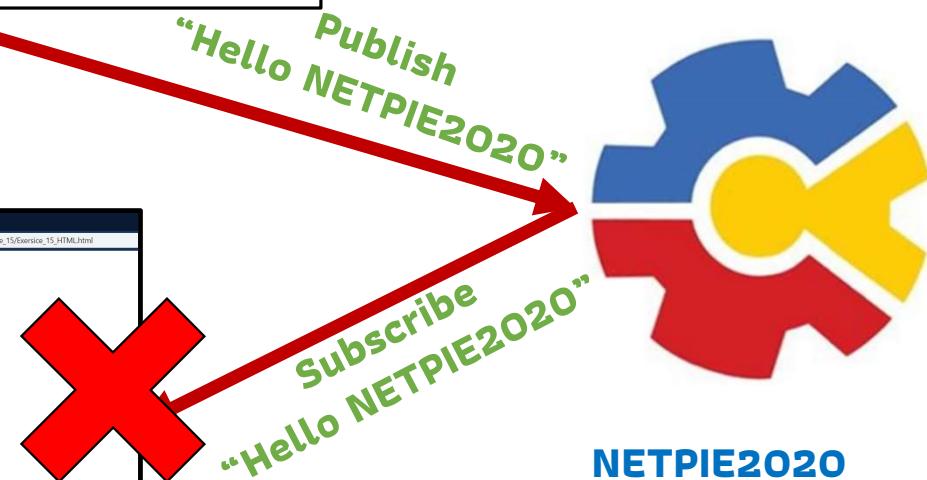
Web Browser

Authentication

Client-ID : 13456789

Token : abcdefghijk

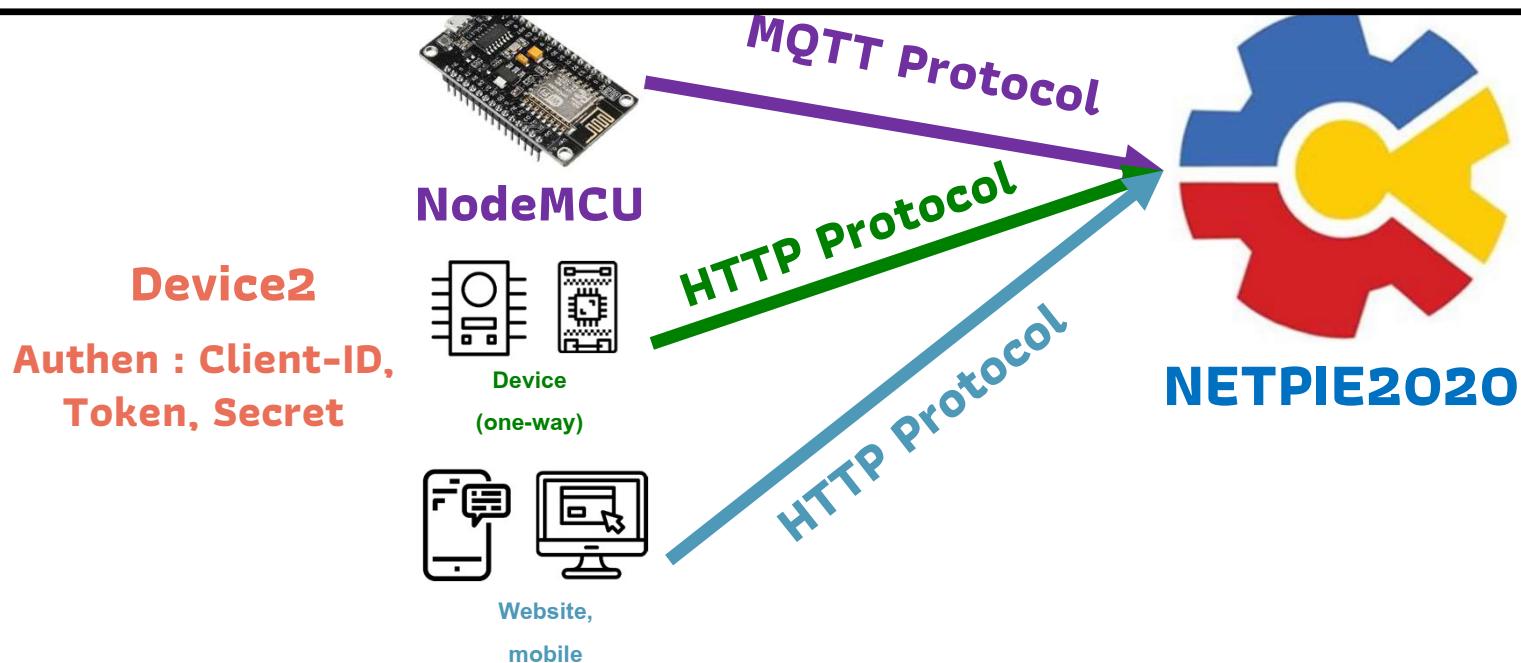
Secret : lmnopqrstuvwxyz



NETPIE2020

Device connection with MQTT Protocol

If you want to use the same device to connect, you must select another protocol.



Device connection with MQTT Protocol

If you want to see messages that are sent using the MQTT Protocol, you must create additional devices and place them in the same group.

NodeMCU

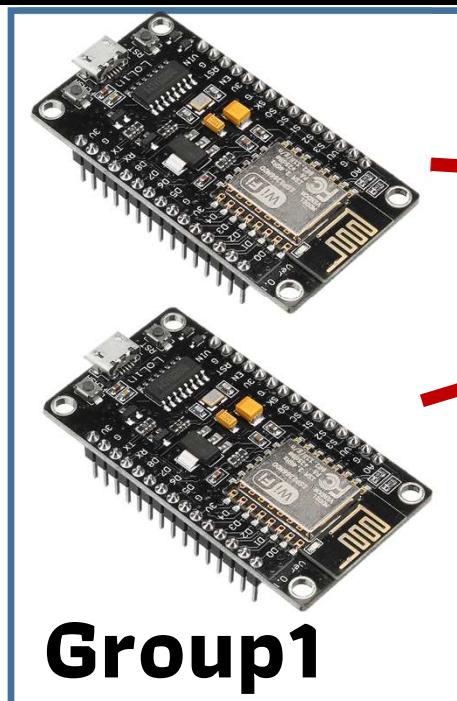
Device2

Authen : Client-ID,
Token, Secret

NodeMCU

Device1

Authen : Client-ID,
Token, Secret



MQTT
Protocol

MQTT
Protocol



NETPIE2020

Group creation



MQTT Protocol

Message “Hello NETPIE2020”
subscribe @msg/test

@msg/test
Message “Hello NETPIE2020”

NETPIE2020

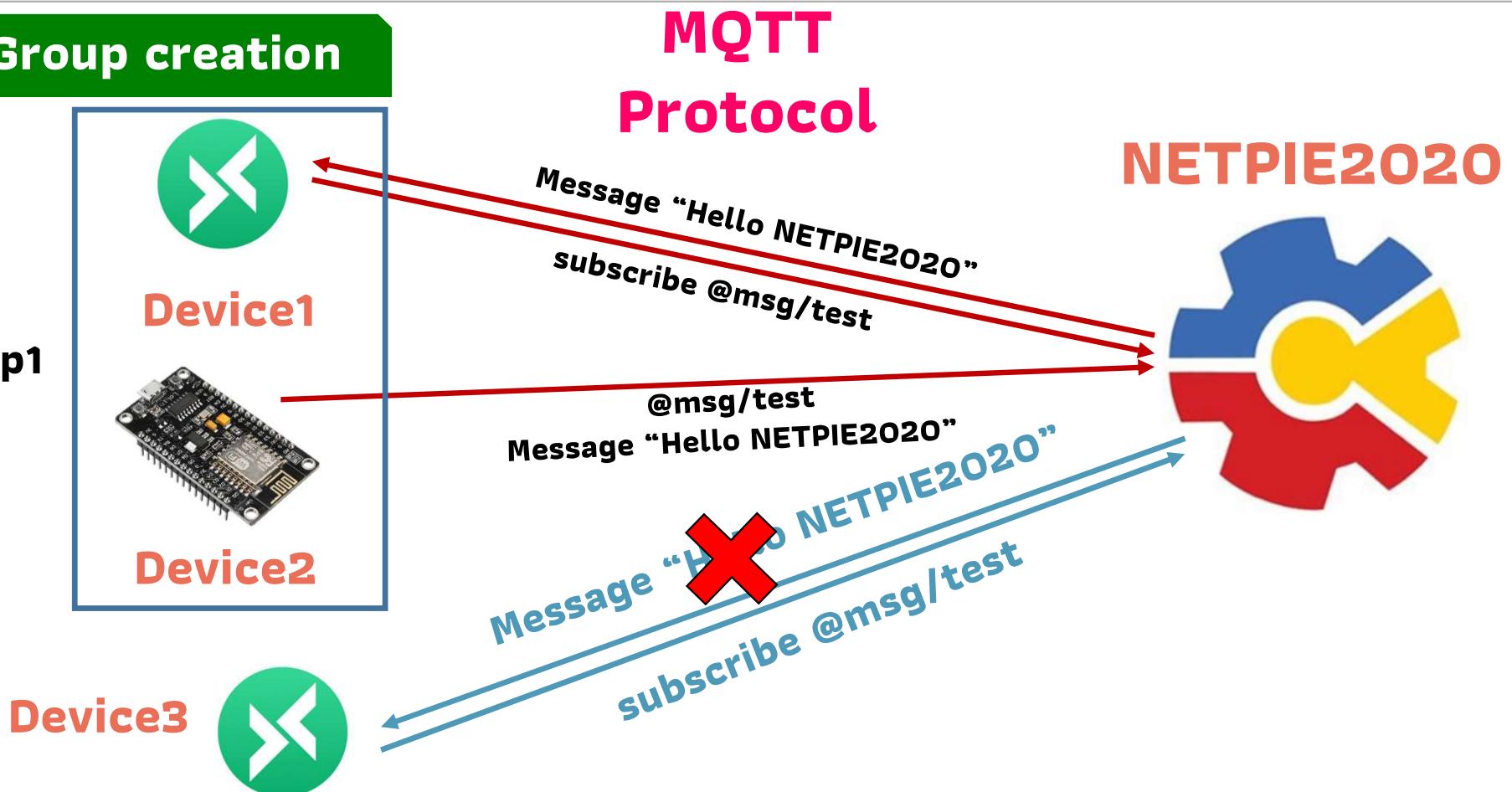


Group creation



Group1

MQTT Protocol



Group creation

The screenshot shows the NETPIE Portal interface. On the left, a sidebar menu is open with several options: PROJECT (+ Add Project), WORKSPACE (Overview, Device, Group, Event Hook, Console), and SETTING (Setting). The 'Group' option is highlighted with a red box and a circled '1'. A hand cursor is shown clicking on the 'Event Hook' option. On the right, the main content area is titled 'Group' and shows a table with columns: ID, Name, Device, and Created Time. The table displays 'No Data'. At the top right of the main area, there is a search bar labeled 'input search text' and a 'Create' button with a circled '2' and a hand cursor pointing at it.

PROJECT + Add Project

helloNETPIE / group

WORKSPACE

Overview 1

Device

Group

Event Hook

Console

SETTING

Setting

Group

Show 0 group

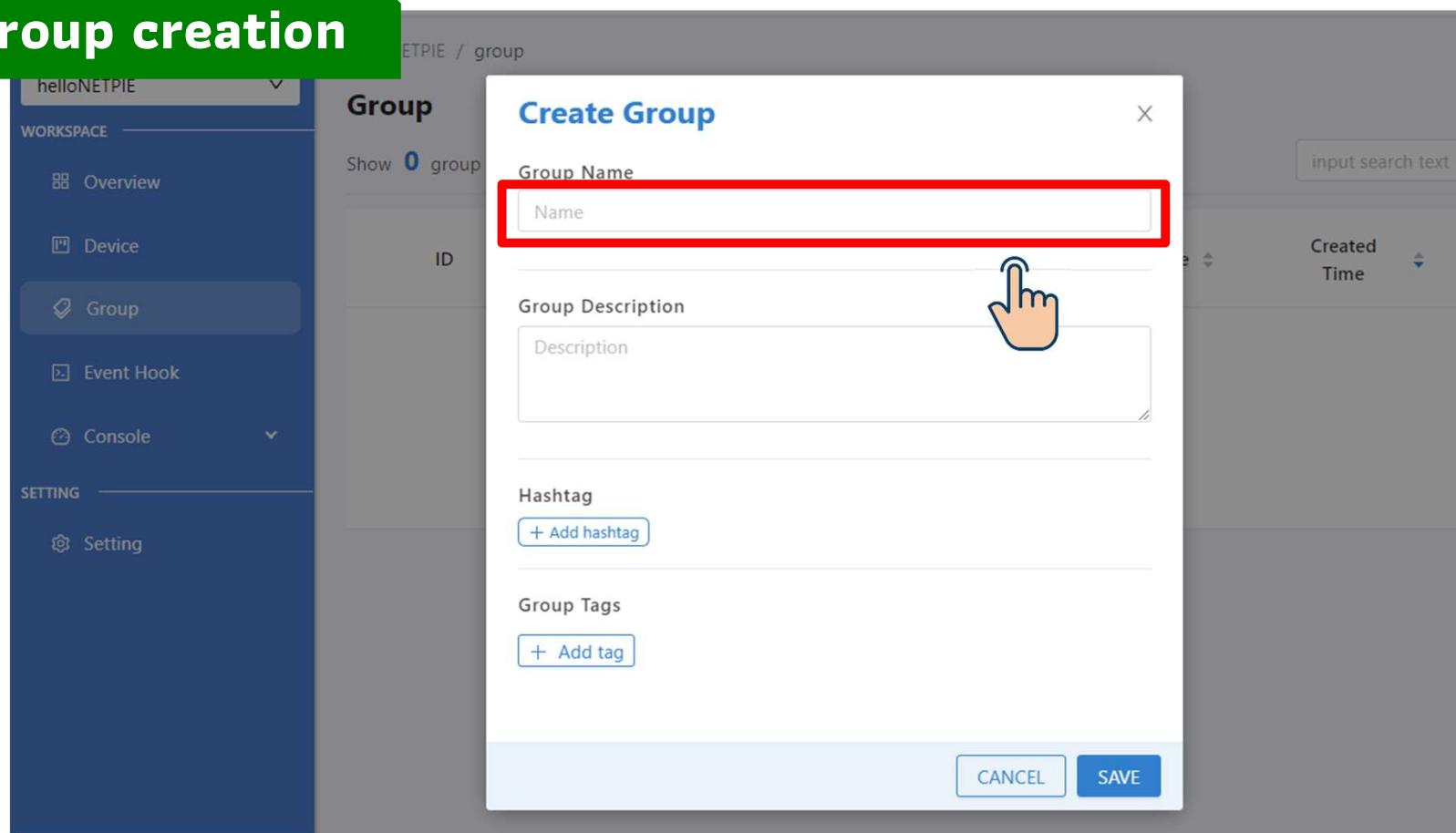
ID Name Device Created Time

No Data

+ Create

input search text

Group creation



Group creation

The screenshot shows the NETPIE Portal interface for creating a new group. On the left, there's a sidebar with 'PROJECT' and 'helloNETPIE' dropdowns, 'WORKSPACE' with 'Overview', 'Device', 'Group' (which is selected and highlighted in blue), 'Event Hook', and 'Console'. On the right, under 'SETTING', there's a 'Setting' option. The main area shows a 'Group' section with a table. The table has columns: ID, Name, Device, and Created Time. One row is listed: ID G944460371864, Name group1, Device 0, and Created Time 2024-07-28 16:12. A red box highlights this row, and a red arrow points from it to a callout box at the bottom. The callout box contains the text: 'Group is created with no device' and 'An existing device must be brought into the group'.

ID	Name	Device	Created Time
G944460371864	group1	0	2024-07-28 16:12

Group is created with no device
An existing device must be brought into the group

Group creation

PROJECT + Add Project

helloNETPIE

WORKSPACE

Overview

Device (highlighted with a red box)

Group

Event Hook

Console

SETTING

Setting

helloNETPIE / device

Device

Show 2 devices

+ Create

input search text

Manage Device

<input type="checkbox"/>	ID	Name	Group	Status	Created Time
<input type="checkbox"/>	adb59c76-3be2-4d83-93b4...	esp8266	-	Online	2024-07-28 15:50
<input type="checkbox"/>	331be25f-75d1-4041-8cac-c...	Device1	My first device	Online	2024-07-28 15:20

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

Group creation

helloNETPIE

WORKSPACE

- Overview
- Device**
- Group
- Event Hook
- Console

SETTING

- Setting

Device

Show 2 devices

+ Create

input search text

Manage Device

ID	Name	Group	Status	Created Time
adb59c76-3be2-4d83-93b4...	esp8266	-	Online	2024-07-28 15:50
331be25f-75d1-4041-8cac-c...	Device1	My first device	Online	2024-07-28 15:20

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

Group creation

PROJECT + Add Project

helloNETPIE

WORKSPACE

Overview Device Group Event Hook Console

SETTING Setting

Device

Show 2 devices

+ Create input search text

Check all Selected 2 devices

Manage Device

ID Name Group Status Created Time

ID	Name	Group	Status	Created Time
adb59c76-3be2-4d83-93b4...	esp8266	-	Online	2024-07-28 15:50
331be25f-75d1-4041-8cac-c...	Device1	My first device	Online	2024-07-28 15:20

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

ID	Name	Group	Status	Created Time
adb59c76-3be2-4d83-93b4...	esp8266	-	Online	2024-07-28 15:50
331be25f-75d1-4041-8cac-c...	Device1	My first device	Online	2024-07-28 15:20

Group creation

The screenshot shows the NETPIE Portal interface for creating a device group. On the left, there's a sidebar with 'PROJECT' and 'helloNETPIE' selected. The main area is titled 'Device' and shows '2 devices'. A modal window titled 'Group Device' is open, displaying a table with one row:

Group ID	Name	Device
G944460371864	group1	0

A red box highlights the first row of the table. A hand cursor icon is positioned over the 'MOVE' button at the bottom right of the modal. The background shows a list of devices with their status and creation time.

Group creation

PROJECT + Add Project

helloNETPIE

WORKSPACE Overview

Device

Show 2 devices

+ Create

input search text

Manage Device

Check all

ID	Name	Group	Status	Created Time
adb59c76-3be2-4d83-93b4...	esp8266	group1	Online	2024-07-28 15:50
331be25f-75d1-4041-8cac-c...	Device1 My first device	group1	Online	2024-07-28 15:20

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

Group indication



Group creation

More details can be observed in the Device Groups

PROJECT + Add Project

helloNETPIE / group

WORKSPACE

- Overview
- Device
- Group**
- Event Hook
- Console

SETTING

Setting

Group

Show 1 group

+ Create

input search text

ID	Name	Device	Created Time
G944460371864	group1	2	2024-07-28 16:12

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

Group creation

PROJECT + Add Project

helloNETPIE

WORKSPACE

- Overview
- Device
- Group
- Event Hook
- Console

SETTING

Setting

helloNETPIE / group / group1

group1

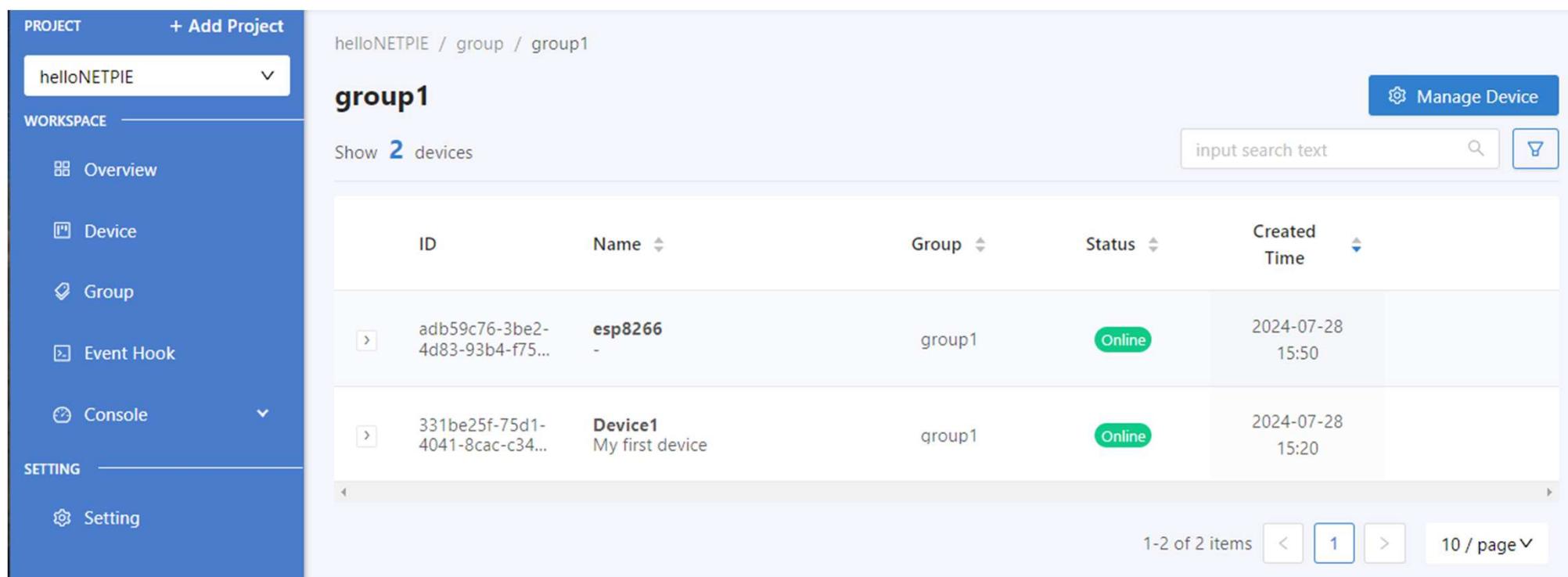
Show 2 devices

input search text

Manage Device

ID	Name	Group	Status	Created Time
adb59c76-3be2-4d83-93b4-f75...	esp8266 -	group1	Online	2024-07-28 15:50
331be25f-75d1-4041-8cac-c34...	Device1 My first device	group1	Online	2024-07-28 15:20

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



Messages from NodeMCU to MQTTX

The screenshot shows the NETPIE Portal interface. On the left is a sidebar with icons for device management, adding new devices, code editor, settings, and help. The main area displays a list of messages from a device named "device1@broker.netpi...". A message is selected, showing its details: topic "@msg/test" and QoS 0. The message content is "Hello NETPIE2020" and it was sent on 2024-07-29 09:20:37:380. Below this, four more messages are listed, all with the same content and timestamp, indicating they were sent sequentially.

Topic	Message Content	Timestamp
@msg/test	Hello NETPIE2020	2024-07-29 09:20:37:380
@msg/test	Hello NETPIE2020	2024-07-29 09:20:39:351
@msg/test	Hello NETPIE2020	2024-07-29 09:20:41:357
@msg/test	Hello NETPIE2020	2024-07-29 09:20:43:366

Workshop 3 : Communication within a Group and outside

Group1



Device 1 and 2 were
created in Workshop 1-2

NETPIE2020

subscribe @msg/test
@msg/test
Message "Hello NETPIE2020"



subscribe @msg/test

Workshop 3 : Communication within a Group and outside

PROJECT + Add Project

helloNETPIE / device

Device

Show 3 devices

+ Create

input search text

Check all

Manage Device

ID Name Group Status Created Time

ID	Name	Group	Status	Created Time
f174d3dd-53d9-40a7-b2c3-46e...	Device3 MQTTbox outside group		Offline	2024-07-29 07:26
adb59c76-3be2-4d83-93b4-f75...	esp8266	group1	Online	2024-07-28 15:50
331be25f-75d1-4041-8cac-c34...	Device1 My first device	group1	Offline	2024-07-28 15:20

Create new Device3

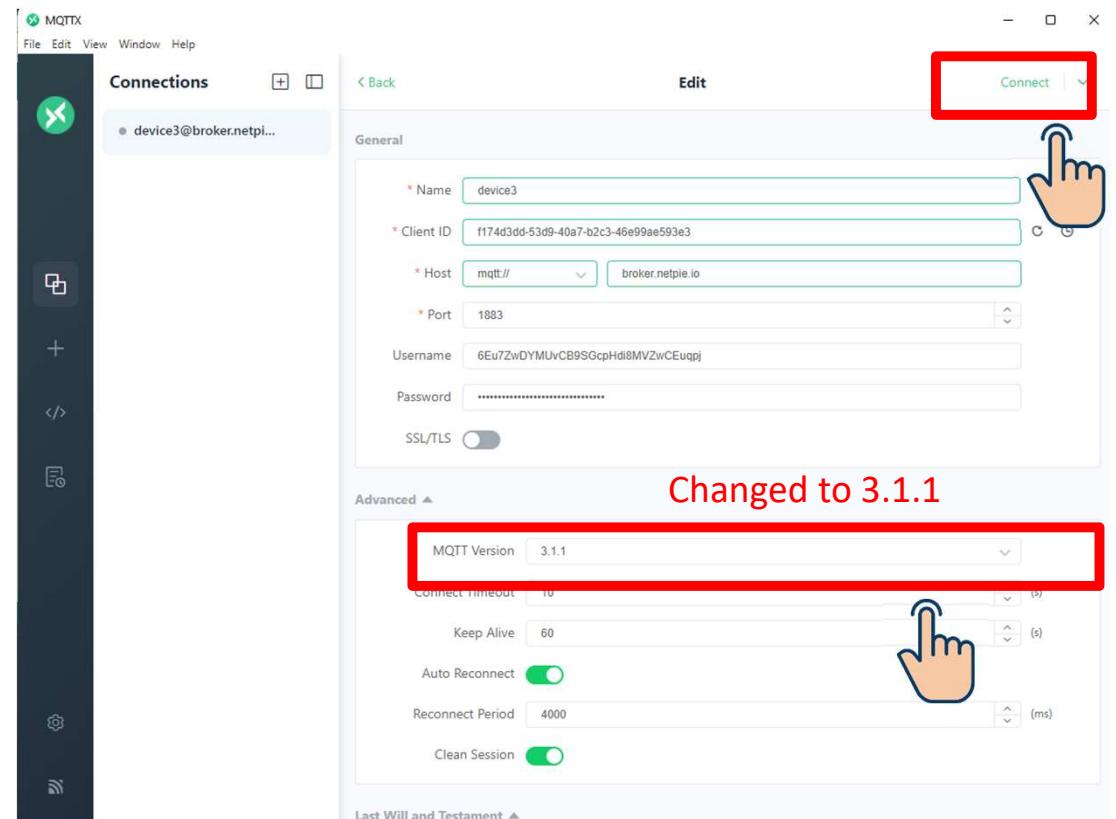
1-3 of 3 items < 1 > 10 / page

ID	Name	Group	Status	Created Time
f174d3dd-53d9-40a7-b2c3-46e...	Device3 MQTTbox outside group		Offline	2024-07-29 07:26
adb59c76-3be2-4d83-93b4-f75...	esp8266	group1	Online	2024-07-28 15:50
331be25f-75d1-4041-8cac-c34...	Device1 My first device	group1	Offline	2024-07-28 15:20

Workshop 3 : Communication within a Group and outside



Open new MQTTX window to connect with Device3



Workshop 3 : Communication within a Group and outside

The screenshot shows the NETPIE Portal interface for managing MQTT connections. On the left, a sidebar menu includes 'Connections', 'Group', 'Event Hook', 'Console' (selected), and 'Setting'. The main area displays a connection for 'device3' with a red box highlighting the connection entry 'device3@broker.netpie...'. A red arrow points from this box to a callout box containing the text 'Check Device3 connection status'. Another red arrow points from this callout to the 'Status' field in a 'Key' dialog box, which shows 'Online' in a green button. The 'Key' dialog also lists 'Token' and 'Secret' fields with their respective values.

MQTTX

File Edit View Window Help

Connections

device3 0

New Subscription

All Received Published

MQTTbox outside group

Group

Event Hook

Console

Setting

Setting

Shadow Schema Trigger Feed i

Key

Check Device3 connection status

Token: 3-46e99ae59

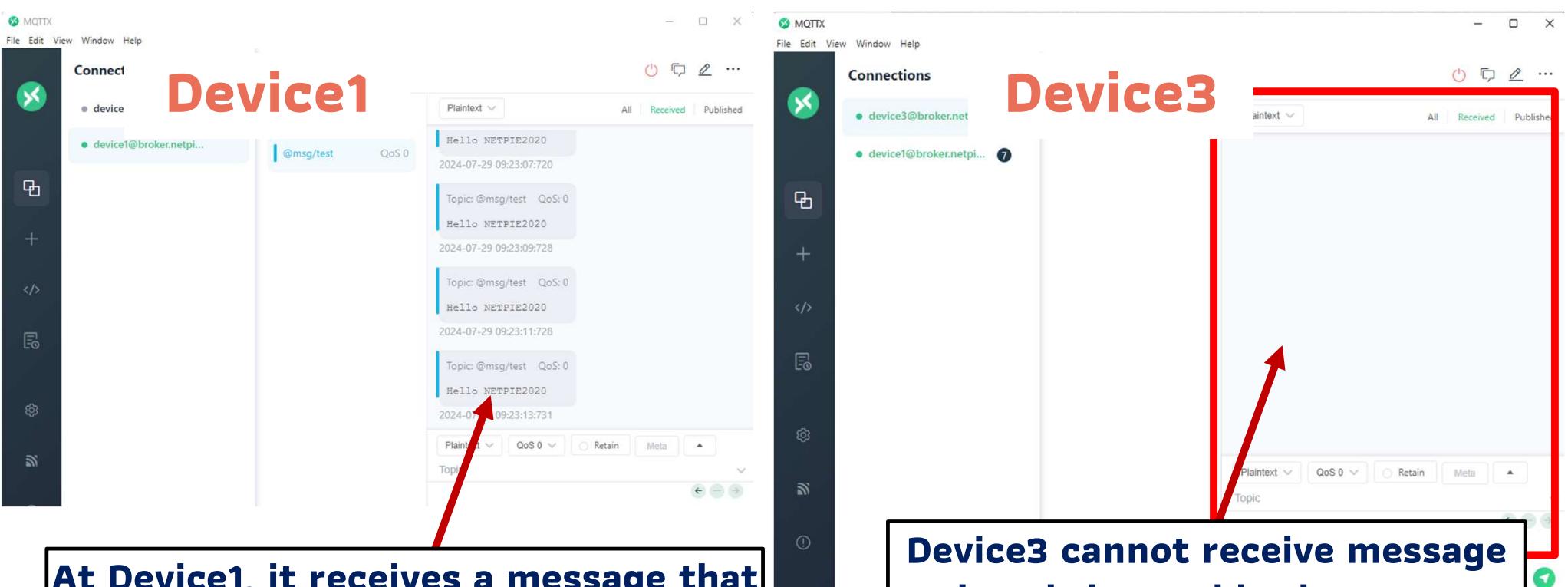
Secret: 6Eu7ZwDYMUvCB9SGcpHdi8MVZwCEt

Status: Online

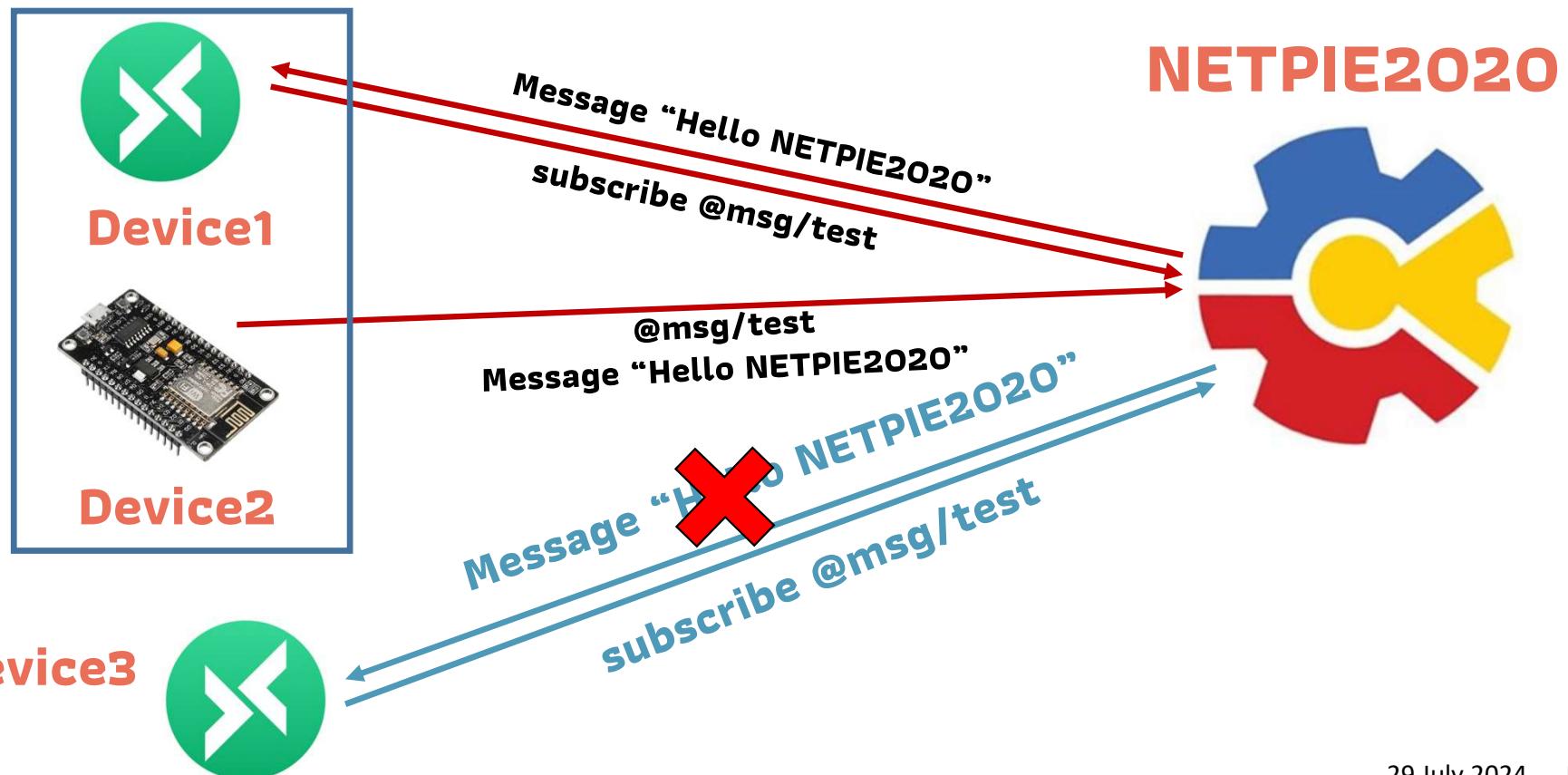
Enable

Cancel SAVE

Workshop 3 : Communication within a Group and outside



Workshop 3 : Communication within a Group and outside





Device Data Management



29 July 2024



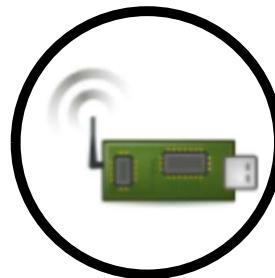
Page 85

Device Data Management

There are two types of messages sent to NETPIE2020 via MQTT.

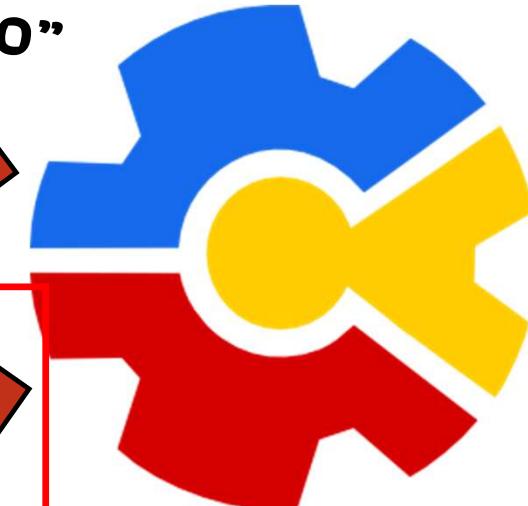
1

Message



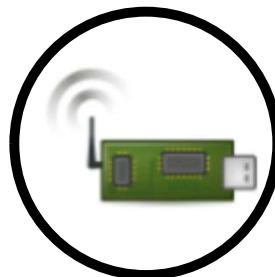
“Hello NETPIE2020”

MQTT Protocol



2

Data



MQTT Protocol
Temp = 25

Device Data Management

NETPIE2020 manages device information in 4 main sections.

- 1 Device Shadow : Latest device database**
- 2 Device Schema : Device data structure**
- 3 Device Trigger : Conditional device data**
- 4 Event Hooks : Device transmission formats**

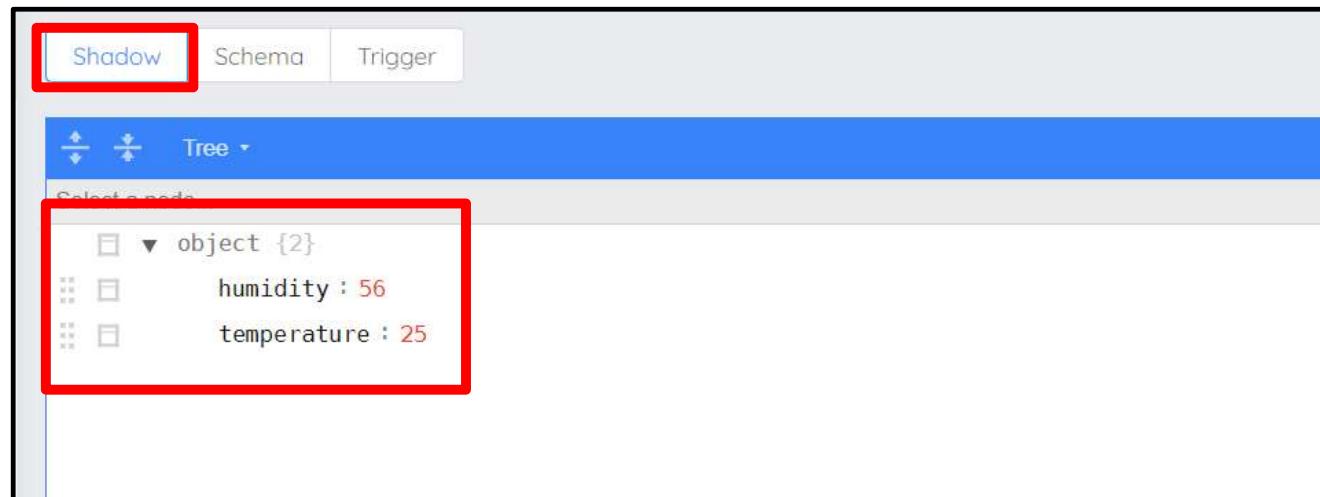


NETPIE2020

Device Shadow

Device Shadow is the virtual database of the device that is allocated for every device for two types of data storage

- 1. Device Shadow Data : sensor data, for example**
- 2. Device Shadow State ឧទ្ទេ : device online/offline status, for example**



Note: previous
Version appearance

Device Shadow

The MQTT topic that is involved with managing Device Shadow

Shadow Topic

Used to manage own device shadow for publishing to edit Shadow information, and subscribing to receive Shadow information.

Publish Topic	Description	Subscribe Topic
@shadow/data/update	To update Shadow Data value by sending a payload in JSON format	@shadow/data/updated

Device Data Management



Device Shadow

Example of sending data in JSON format



NodeMCU

@shadow/data/update
Publish : { "data": { "Temp" : 24, "Humi" : 58 }}



NETPIE2020

"Temp" : 24
"Humi" : 58



Device Shadow

But in order to store data in Device Shadow, you need to create a Device Schema first.

Device Data Management

Device Schema

Device Schema is a data structure defined to manage Device Shadow. For devices that need data management, a Device Schema should be created. This Device Schema allows the server to

- Check data types before storing
- Converting data before storage, such as changing data units.
- Data collection in Timeseries Database (Feed)

The screenshot shows a user interface for managing Device Schema. At the top, there are three tabs: 'Shadow' (disabled), 'Schema' (selected and highlighted with a red box), and 'Trigger'. Below the tabs is a toolbar with icons for 'Tree' (selected) and other operations. The main area displays a hierarchical tree structure of schema definitions:

```
Object > properties > humidity > type
  object {2}
    additionalProperties : false
    properties {2}
      humidity {2}
        operation {1}
          store {1}
            ttl : 7d
        type : number
      temperature {2}
        operation {2}
          store {1}
            ttl : 7d
          transform {1}
            expression : $.temperature + 32
        type : number
```

The 'humidity' node has two properties: 'operation' (with one child 'store') and 'type' (set to 'number'). The 'temperature' node also has two properties: 'operation' (with one child 'store') and 'transform' (with one child 'expression' containing the value '\$.temperature + 32'). The 'type' field for both properties is highlighted with a yellow box.

Device Data Management



NECTEC
a member of NSTDA

Device Schema

Declaring the Device Schema in JSON format.

```
{  
  "additionalProperties": false,  
  "properties": {  
    "light": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "type": "number"  
    },  
    "temperature": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "transform": {  
        "expression": "[$.temperature]*1.8] + 32"  
      }  
    },  
    "type": "number"  
  }  
}
```

Device Schema consists of 2 main parts

additionalProperties

Properties

Device Data Management



Device Schema

Declaring the Device Schema in JSON format

```
{  
  "additionalProperties": false,  
  "properties": {  
    "light": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "type": "number"  
    },  
    "temperature": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "transform": {  
        "expression": "[\$temperature]*1.8] + 32"  
      },  
      "type": "number"  
    }  
  }  
}
```

additionalProperties

This is the permission to save data to Shadow or Timeseries Database [in case the data is not declared in the properties section](#).

additionalProperties has 2 status values

true : authorize data write to Shadow or Timeseries Database

false : prohibit data write for data not defined in Properties

In the example, two items in properties are humidity and temperature

If the data received are temp, humid, light

additionalProperties = true : will record temperature, humidity and light

additionalProperties = false : will record only temperature, light

Device Data Management

Device Schema

Declaring the Device Schema in JSON format

```
{  
  "additionalProperties": false,  
  "properties": {  
    "light": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "type": "number"  
    },  
    "temperature": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "transform": {  
        "expression": "[$.temperature]*1.8 + 32"  
      },  
      "type": "number"  
    }  
  }  
}
```

First, define a field name [for example, 'light' and 'Temperature'] and define properties for each field, which are divided into two parts:

Properties

Operation For setting up data handling in that field, including **store** for keeping data inTimeseries Database **ttl** duration of data inTimeseries Database. Data that exceeds expiration date will be automatically deleted. To store system data, this value needs to be set in units of ms [milliseconds], s [seconds], m [minutes], h [hours], d [days], y [years]

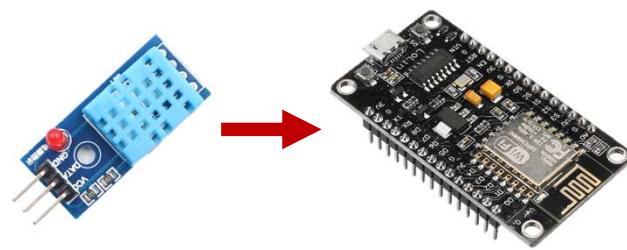
Transform data transformation before keeping **expression** formula for transformation

For example, a formula to convert Celsius to Fahrenheit= $[\$.temperature * 1.8] + 32$

Type data type in a field, such as number, string, array, object

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.



DHT11 NodeMCU

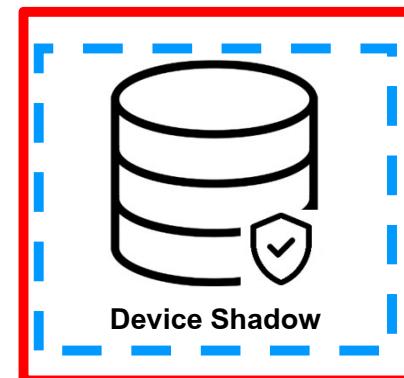
Note : if IGR board is used, light intensity is also written to shadow.



@shadow/data/update
Publish : { "data": { "Temp" : 24, "Humi" : 58, "place" : "NECTEC" }}



NETPIE2020



"Temp" : 75
"Humi" : 58
"place" : "NECTEC"

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

Provide a Device Schema before sending data from NodeMCU

The screenshot shows the Device Data Management interface. On the left, there's a sidebar with 'Device' (selected), 'Group', 'Event Hook', 'Console' (with a dropdown arrow), 'SETTING', and 'Setting'. The main area has tabs for 'Shadow', 'Schema' (which is highlighted with a red box and has a hand cursor pointing at it), 'Trigger', 'Feed', and an info icon. Below the tabs is a tree view with a search bar and a 'Select a node...' placeholder. It shows two nodes: 'object {0}' and '(empty object)'. To the right, there are configuration fields for Client ID, Token, Secret, Status (set to 'Online'), and Enable (a toggle switch). There are also 'Copy' buttons for each token field and a refresh button for status. At the bottom right are 'Cancel' and 'SAVE' buttons.

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

Provide a Device Schema before sending data from NodeMCU

The screenshot shows the Device Data Management interface. On the left, there's a sidebar with 'WORKSPACE' and 'SETTING' sections. Under 'WORKSPACE', there are links for Overview, Device, Group, Event Hook, and Console. Under 'SETTING', there is a link for Setting. The main area has a 'Detail' section with a creation date of 2024-07-28. To the right is a 'Key' section with Client ID, Token, Secret, Status (Online), and Enable toggle. Below these are tabs for Shadow, Schema (which is selected), Trigger, Feed, and an info icon. A red box highlights the 'Tree' dropdown button. A callout box with the text 'Change from Tree to Code' has a hand cursor pointing at the 'Tree' button. The bottom navigation bar includes icons for back, forward, search, cancel, and save.

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

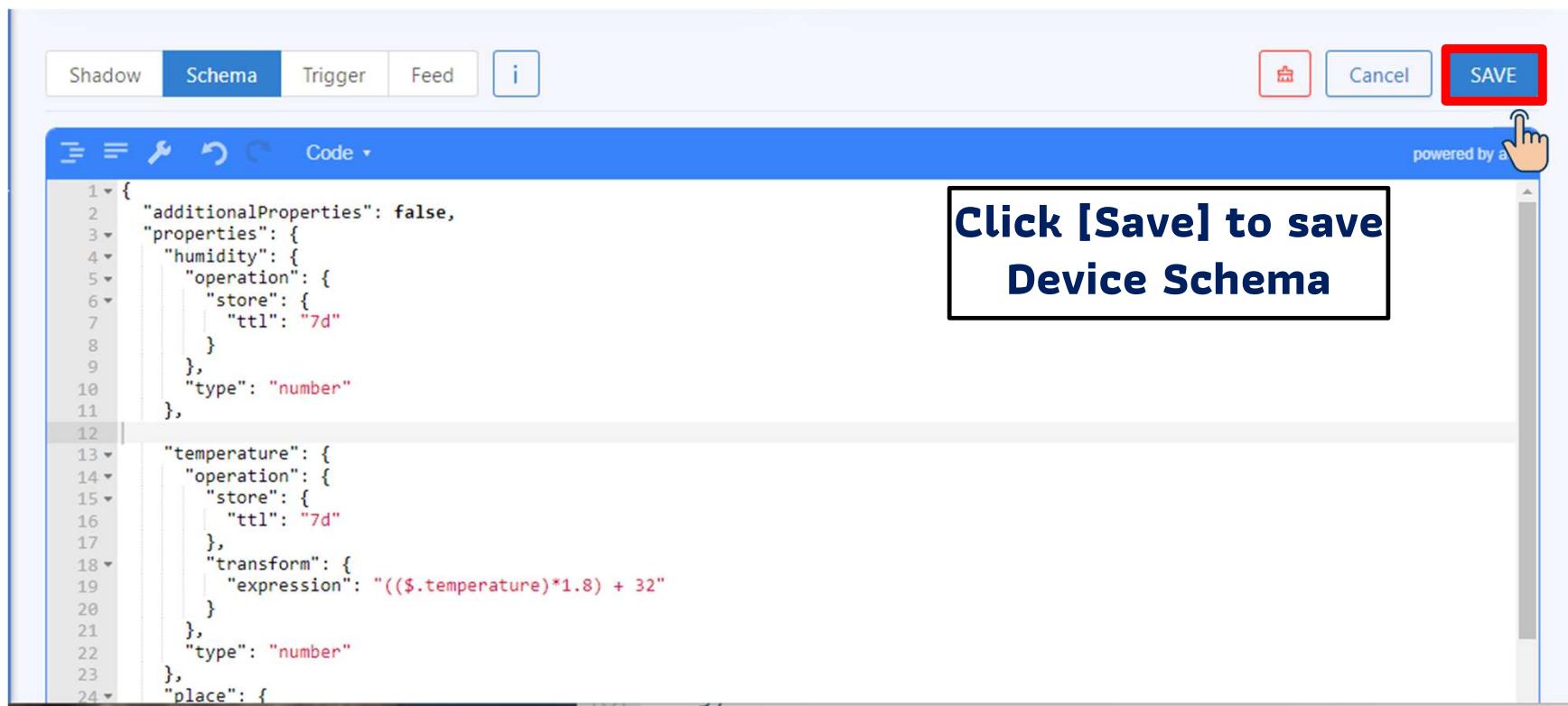
Device Schema in JSON format

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

Copy to Device
Schema

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.



The screenshot shows a device schema editor interface. At the top, there are tabs: Shadow, Schema (which is selected), Trigger, Feed, and i. Below the tabs is a toolbar with icons for copy, paste, delete, and a dropdown labeled 'Code'. On the right side of the toolbar are buttons for 'Cancel' and 'SAVE', with 'SAVE' being highlighted by a red box and a hand cursor pointing at it. A message box in the center says 'Click [Save] to save Device Schema'. The main area contains a JSON configuration:

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

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

Device Schema in JSON format

Set **additionalProperties = false**

In order not to save values other than listed in properties to Shadow.

The first variable is **light**

with properties

- Store for 7 days
- Data type is number

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

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

Device Schema in JSON

format

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

```
{"temperature": {  
  "operation": {  
    "store": {  
      "ttl": "7d"  
    }  
  },  
  "transform": {  
    "expression": "[($.temperature)*1.8] + 32"  
  },  
  "type": "number"  
},  
"place": {  
  "operation": {  
    "store": {  
      "ttl": "7d"  
    }  
  },  
  "type": "string"  
},  
"humidity": {  
  "operation": {  
    "store": {  
      "ttl": "7d"  
    }  
  },  
  "type": "number"  
},  
"type": "object"
```

Variable to store is **temperature** with properties

- Store for 7 days
- Convert from Celsius to Fahrenheit
- Variable type is number

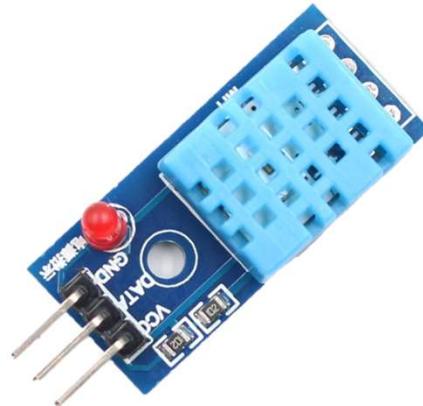
Variable to store is **place** with properties

- Store for 7 days
- Variable type is string

Device Data Management

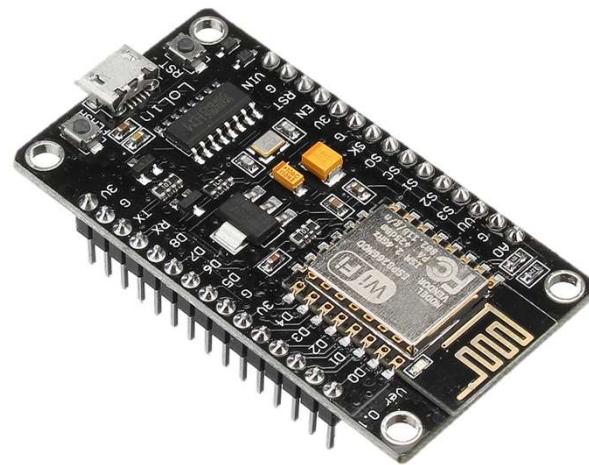
Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

Connect DHT11 to NodeMCU



DHT11

Data -> D4
Vcc -> 3v3
GND -> GND



NodeMCU

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

Open Workshop4.ino

Code in Workshop4.ino consists of 3 parts

1 Part 1 Library and variable declaration

Include required libraries

Declare pins for DHT11

Variable declaration and assignment to connect to WiFi and MQTT

Variable declarations

MQTT commands

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
```

```
#define DHTPIN D4
#define DHTTYPE DHT11
```

```
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);
DHT dht(DHTPIN, DHTTYPE);
```

```
char msg[100]
```

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

2 Part 2 : various functions

MQTT connection function

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

MQTT connection

- “connected” is shown when succeed
- if not, “failed ...” is shown and reconnect is attempted automatically

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

2 Part 2 : various functions

```
void setup() {  
    Serial.begin(115200);  
    Serial.println();  
    Serial.print("Connecting to ");  
    Serial.println(ssid);  
  
    WiFi.begin(ssid, password);  
    while (WiFi.status() != WL_CONNECTED) {  
        delay(500);  
        Serial.print(".");  
    }  
    Serial.println("");  
    Serial.println("WiFi connected");  
    Serial.println("IP address: ");  
    Serial.println(WiFi.localIP());  
    client.setServer(mqtt_server, mqtt_port);  
    dht.begin();  
}
```

Setup function

Connect WiFi and MQTT with specified settings

Start the DHT object instance

Device Data Management

1



Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

3

Part 3 : loop() function

```
void loop() {
    int humidity = dht.readHumidity();
    int temperature = dht.readTemperature();
    String place = "NECTEC Thailand";
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    String data = "{\"data\": {\"humidity\": " + String(humidity) + ", \"temperature\": " +
String(temperature) + ", \"place\": " + String(place) + "}}";
    Serial.println(data);
    data.toCharArray(msg, [data.length() + 1]);
    client.publish("@shadow/data/update", msg);
    delay(2000);
}
```

Read humidity and temperature data from sensor, declare the place before sending to NETPIE2020

To maintain the connection state of MQTT

command to send message toTopic : @shadow/data/update every 2 seconds

Device Data Management

Workshop 4: Send Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

Sending message in JSON format on
Arduino IDE

Message to be sent is

```
{ "data" : { "humidity" : 52, "temperature" : 25, "place" : "NECTEC Thailand" }}
```

But the compiler on Arduino IDE sees " as String declaration. To fix this, put \ in front of " so that the program interprets " as part of the message

```
String data = "{\"data\": {\"humidity\":\"" + String(humidity) + ",  
\"temperature\":\"" + String(temperature) + ",  
\"place\":\"" + String(place) + "}}";
```

Note : Light intensity is added for IGR board



Device Data Management

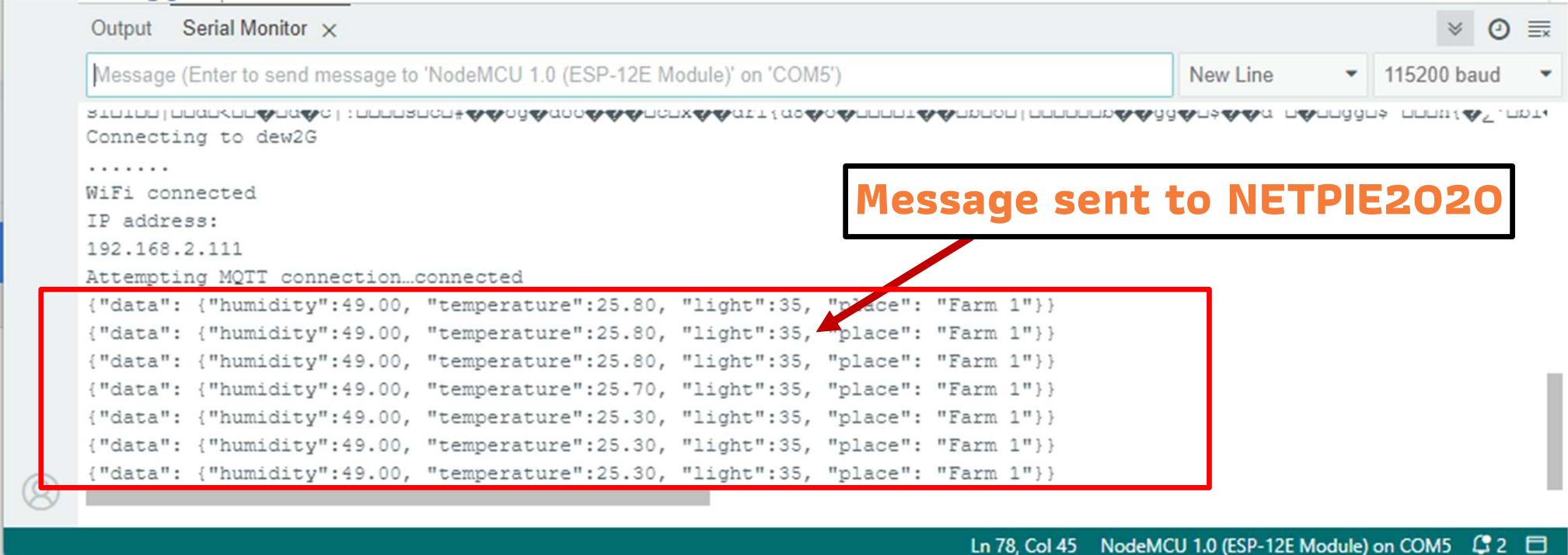
Workshop 4: Send Humidity and Temperature values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

```
COM3
| Send
.....
WiFi connected
IP address:
172.18.48.44
Attempting MQTT connection...connected
{"data": {"humidity":50, "temperature":24, "place": "NECTEC Thailand"} }

Message sent to NETPIE2020
```

Device Data Management

Workshop 4: Send Humidity and Temperature and Light values from NodeMCU to NETPIE2020 and save them in Timeseries DB [IGR board]



Output Serial Monitor X

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM5')

New Line 115200 baud

```
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM5')
New Line 115200 baud
Connecting to dew2G
.....
WiFi connected
IP address:
192.168.2.111
Attempting MQTT connection...connected
{"data": {"humidity":49.00, "temperature":25.80, "light":35, "place": "Farm 1"}}
{"data": {"humidity":49.00, "temperature":25.80, "light":35, "place": "Farm 1"}}
{"data": {"humidity":49.00, "temperature":25.80, "light":35, "place": "Farm 1"}}
{"data": {"humidity":49.00, "temperature":25.70, "light":35, "place": "Farm 1"}}
{"data": {"humidity":49.00, "temperature":25.30, "light":35, "place": "Farm 1"}}
{"data": {"humidity":49.00, "temperature":25.30, "light":35, "place": "Farm 1"}}
{"data": {"humidity":49.00, "temperature":25.30, "light":35, "place": "Farm 1"}}

Ln 78, Col 45 NodeMCU 1.0 (ESP-12E Module) on COM5 2 2
```

Message sent to NETPIE2020

Device Data Management

Workshop 4: Send sensor values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

The screenshot shows the Device Shadow interface of a cloud platform. At the top, there are tabs for 'Shadow' (which is selected), 'Schema', 'Trigger', and 'Feed'. On the right, there are buttons for 'Cancel' and 'SAVE'. Below the tabs, there's a toolbar with icons for creating a new node, deleting, and search. The main area is titled 'Tree' and shows a single node named 'object {3}'. Underneath it, three properties are listed: 'humidity : 48', 'temperature : 77.53999999999999', and 'place : Farm 1'. A red box highlights the entire list of properties, and a red arrow points from this box to a text overlay. The text 'Data recorded on Device Shadow' is displayed in a large orange font within a black-bordered box. Another text overlay at the bottom left says 'Before adding schema properties for light'.

Data recorded on Device Shadow

Before adding schema properties for light

Property	Value
humidity	48
temperature	77.53999999999999
place	Farm 1

Device Data Management

Workshop 4: Send sensor values from NodeMCU to NETPIE2020 and save them in Timeseries DB.

The screenshot shows a software interface for managing device shadows. At the top, there are tabs: Shadow (which is selected), Schema, Trigger, Feed, and an information icon. To the right of the tabs are buttons for Cancel and Save. A large orange box highlights the title "Data recorded on Device Shadow". Below the title, a red arrow points to a red box around a list of sensor data. A blue box contains the text "After adding schema properties for light".

Data recorded on Device Shadow

Shadow Schema Trigger Feed i Cancel SAVE

Tree Last Update : 29-07-24 11:08

Select a node...

object {4}

- humidity : 53
- temperature : 25.8
- light : 32
- place : Farm 1

After adding schema properties for light

Device Data Management

Workshop 4: Wokwi simulation

WOKWI Lect6_ws4 Docs

sketch.ino diagram.json libraries.txt Library Manager

Simulation

Editing DHT22

Temperature: 28.0°C

Humidity: 56.5%

<https://wokwi.com/projects/40>

```
1 // Lect6_ws4_wokwi.ino
2 // dew.ninja July 2024
3 // read humidity, temperature, light data
4 // and publish to NETPIE 2020 together with a
5 // string that indicates place
6
7 #include <WiFi.h>
8 #include <PubSubClient.h>
9 #include "DHT.h"
10 #include <Wire.h>
11 //##include <BH1750FVI.h> // remove for Wokwi
12
13
14 #define DHTPIN 4 // change to 4 for Wokwi sim on ESP32
15 #define DHTTYPE DHT22 // Wokwi library uses DHT22
16
17 const char* ssid = "Wokwi-GUEST";
18 const char* password = "";
19 const char* mqtt_server = "broker.netpie.io";
20 const int mqtt_port = 1883;
21 const char* mqtt_Client = "adb59c76-3be2-4d83-93b4-f75da4abba99";
22 const char* mqtt_username = "ASPgYnRFXkkqorVLM95Qn4BUbqjvLvf";
23 const char* mqtt_password = "5Uz6sUquP1HFPH7gWP1uFWS6bDyKZiv";
24
25
26 WiFiClient espClient;
27 PubSubClient client(espClient);
28 DHT dht(DHTPIN, DHTTYPE);
29
30 // --- remove these 4 lines for Wokwi sim-----
31 // uint8_t ADDRESSPIN = 15; // D8, not used
32 // BH1750FVI::eDeviceAddress_t DEVICEADDRESS = BH1750FVI::k_DevAddress_L;
33 // BH1750FVI::eDeviceMode_t DEVICEMODE = BH1750FVI::k_DevModeContHighRes;
34
35 // Create the Lightsensor instance
36 //BH1750FVI LightSensor(ADDRESSPIN, DEVICEADDRESS, DEVICEMODE);
37 // -----
38
39
40 char msg[100];
41
42 void reconnect() {
43     while (!client.connected()) {
```

Note for **Device Trigger and Event Hook** slides

Since this topic is not covered in my course. The original slides are left intact.

Device Trigger and Event Hook

It is a system that binds to changing Device Shadow information to external actions (Event Hook), such as setting alerts according to different status. According to the condition of the device set by the Trigger will be declared in JSON format:

```
{  
    "enabled": true,  
    "trigger": [  
        {  
            "action": "EVENT_HOOK_NAME",  
            "event": "SHADOW.UPDATED or DEVICE.STATUSCHANGED",  
            "condition": "Operation List ==, !=, >, >=, <, <=, in",  
            "msg": "text",  
            "option": {}  
        }  
    ]  
}
```

Device Trigger and Event Hook

Trigger Format consists of 2 parts

```
{  
  "enabled": true,  
  "trigger": [  
    {  
      "action": "EVENT_HOOK_NAME",  
      "event": "SHADOW.UPDATED or DEVICE.STATUSCHANGED",  
      "condition": "Operation List ==, !=, >, >=, <, <=, in",  
      "msg": "text",  
      "option": {}  
    }  
  ]  
}
```

1. enable is used to turn the Trigger on/off

Device Data Management

1



Device Trigger and Event Hook

Trigger Format consist of 2 parts

2. trigger is for Trigger settings

- action : what Trigger has to do when an event occurs. Specify Event Hook name.

```
{  
  "enabled": true,  
  "trigger": [  
    {  
      "action": "EVENT_HOOK_NAME",  
      "event": "SHADOW.UPDATED or DEVICE.STATUSCHANGED",  
      "condition": "Operation List ==, !=, >, >=, <, <=, in",  
      "msg": "text",  
      "option": {}  
    }  
  ]  
}
```

Device Shadow

SHADOW.UPDATED : happens when Device Shadow Data changes according to certain condition [in this case we have to specify some condition, otherwise the Trigger would not fire]

DEVICE.STATUSCHANGED : happens when the Device changes its platform connection status from On

Device Trigger and Event Hook

Trigger Format consist of 2 parts

```
{  
  "enabled": true,  
  "trigger": [  
    {  
      "action": "EVENT_HOOK_NAME",  
      "event": "SHADOW.UPDATED or DEVICE.STATUSCHANGED",  
      "condition": "Operation List ==, !=, >, >=, <, <=, in",  
      "msg": "text",  
      "option": {}  
    }  
  ]  
}
```

Condition : conditional change in Device Shadow Data . Use in cas of SHADOW.UPDATED

msg : a message to notify the user when trigger fires

option : use for specify some other parameters

For reference, variables in Trigger can be found
[in https://docs.nexpie.io/device-config.html#device-trigger-and-event-hook](https://docs.nexpie.io/device-config.html#device-trigger-and-event-hook)

Device Data Management



Device Trigger and Event Hook

Trigger and Event Hook example

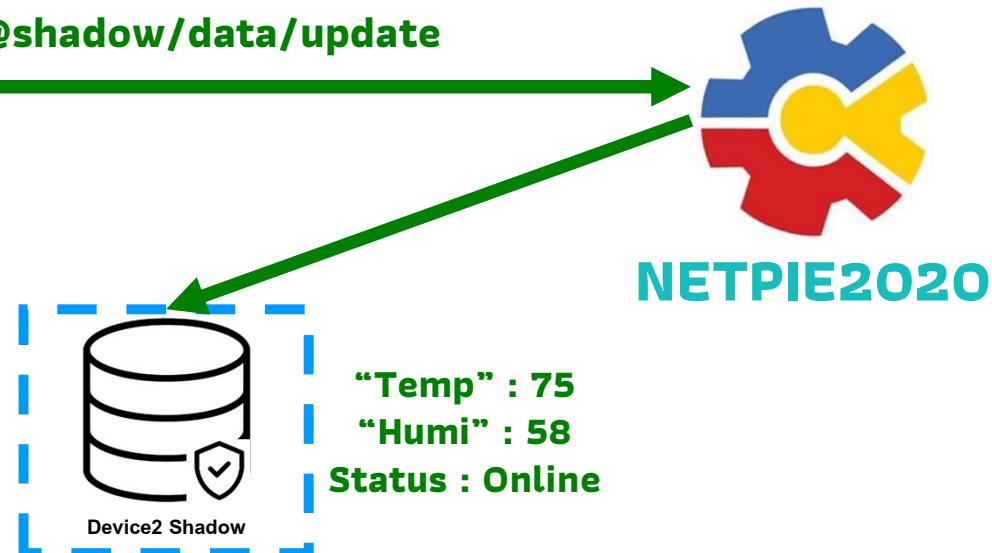
Event Hooks
LINE NOTIFY

LINE



Device2

Publish : @shadow/data/update



Trigger : DEVICE.STATUSCHANGE

Device Data Management



Device Trigger and Event Hook

Trigger and Event Hook example



Device2

Publish : @shadow/data/update



Event Hooks
LINE NOTIFY



NETPIE2020



Trigger : DEVICE.STATUSCHANGE

Device Data Management

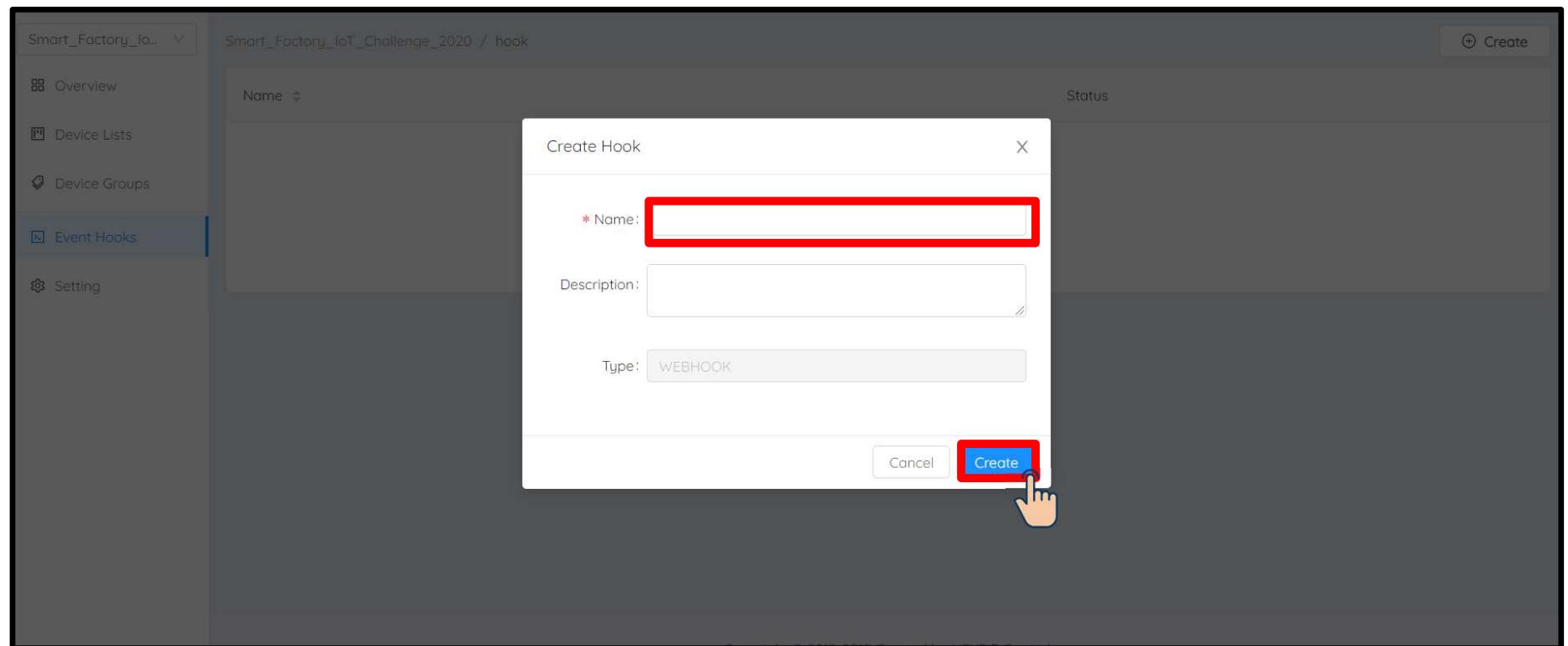


Create Event Hook

The screenshot shows a web-based interface for managing device data. On the left, a sidebar menu lists several options: Overview, Device Lists, Device Groups, Event Hooks, and Setting. The 'Event Hooks' option is highlighted with a red box and has a hand cursor icon pointing at it. The main content area is titled 'Smart_Factory_IoT_Challenge_2020 / hook'. It contains a table with columns for 'Name' and 'Status'. There is one entry in the table: a folder icon under 'Name' and 'No Data' under 'Status'. In the top right corner of the main area, there is a '+ Create' button.

Device Data Management

Create Event Hook



Device Data Management

Create Event Hook

The screenshot shows a user interface for managing event hooks in a smart factory IoT challenge. The left sidebar includes links for Overview, Device Lists, Device Groups, Event Hooks (which is selected and highlighted in blue), and Setting. The main content area displays a table titled 'Smart_Factory_IoT_Challenge_2020 / hook'. The table has columns for Name and Status. One row, named 'LINE_NOTIFY', is highlighted with a red box and has a hand cursor pointing to its status column.

Name	Status
LINE_NOTIFY	(checkbox icon)

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

Device Data Management



Create Event Hook

The screenshot shows the 'Create Event Hook' interface. On the left, a sidebar lists navigation options: Overview, Device Lists, Device Groups, Event Hooks (which is selected), and Setting. The main area has a title bar 'Smart_Factory_IoT_Challenge_2020 / hook / LINE_NOTIFY'. Below the title, there are two panels: 'Description' (empty) and 'Configuration'. In the 'Configuration' panel, the 'Enable' toggle switch is highlighted with a red box and a hand cursor. The 'Type' dropdown is set to 'WEBHOOK'. At the bottom, a toolbar has several icons, with the 'Code' icon highlighted by a red box and a hand cursor. A large callout box with a black border and white text 'Change from Tree to Code' is positioned over the toolbar area. The bottom right corner of the interface says 'powered by ace'.

Create Event Hook

Event Hook Example of a Line Notify that specifies 4 Attributes

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

Copy and paste to Event Hook on
NETPIE2020

Device Data Management

Create Event Hook

Smart_Factory_IoT_Challenge_2020 / hook / LINE_NOTIFY

Description

Configuration

Enable : WEB

Type : WEB

Slide to Enable

Tree

Select a node...

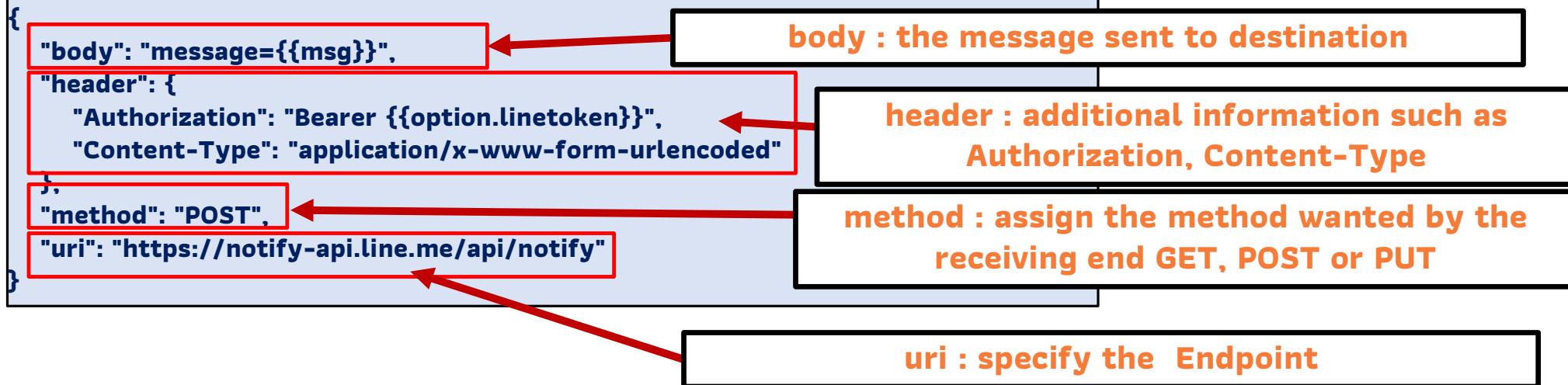
- object {4}
 - uri : <https://notify-api.line.me/api/notify>
 - method : POST
 - header {2}
 - Authorization : Bearer {{option.linetoken}}
 - Content-Type : application/x-www-form-urlencoded
 - body : message={{msg}}

Save Cancel

Device Data Management

Create Event Hook

Event Hook Example of a Line Notify that specifies 4 Attributes



*** In Event Hook, variables sent from Trigger can be referenced by using {{...}} symbols around them. For example, if we want to reference msg from Trigger, use {{msg}} Or in the option of linetoken, use {{option.linetoken}}

Device Data Management

Create Event Hook

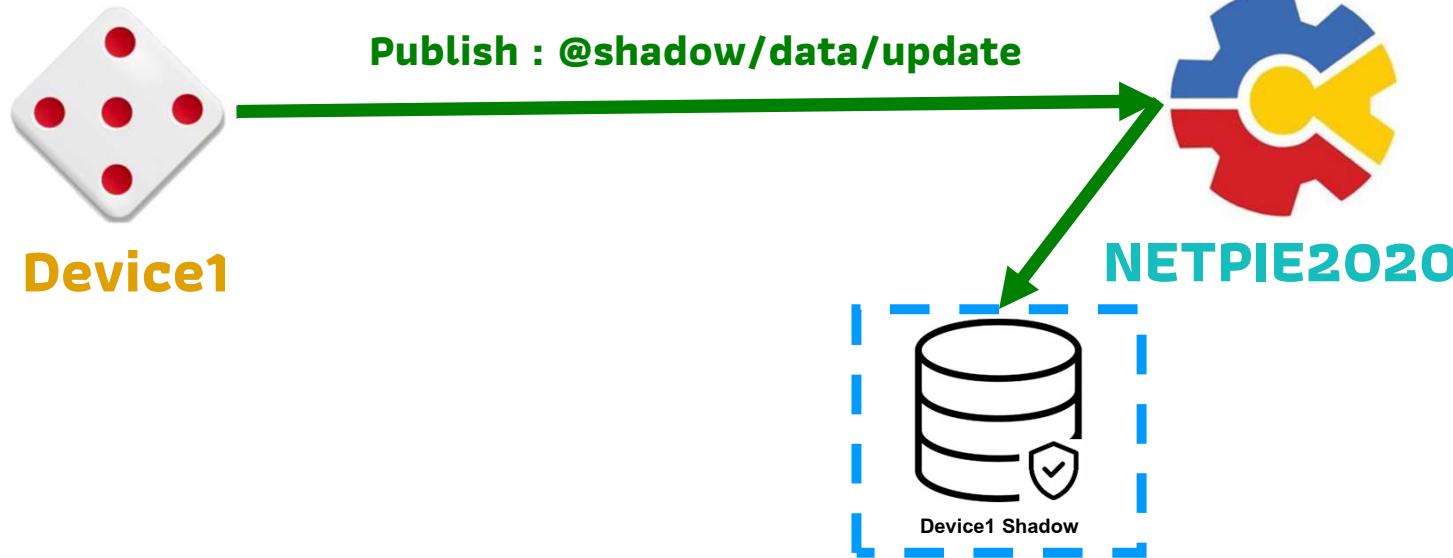
The screenshot shows a software interface for managing event hooks. The top navigation bar includes a dropdown for 'Smart_Factory_Io...', a breadcrumb path 'Smart_Factory_IoT_Challenge_2020 / hook / LINE_NOTIFY', and a 'Edit' button. On the left, a sidebar lists 'Overview', 'Device Lists', 'Device Groups', 'Event Hooks' (which is selected), and 'Setting'. The main area is titled 'Smart_Factory_IoT_Challenge_2020 / hook / LINE_NOTIFY'. It contains two sections: 'Description' (empty) and 'Configuration' (with 'Enable' set to 'On' and 'Type' set to 'WEBHOOK'). Below these is a 'Tree' view with a blue header, showing a hierarchical structure of configuration parameters:

- object {4}
 - uri : <https://notify-api.line.me/api/notify>
 - method : POST
 - header {2}
 - Authorization : Bearer {{option.linetoken}}
 - Content-Type : application/x-www-form-urlencoded
 - body : message={{msg}}

Device Data Management



Workshop 5 : Temperature and device status change notification



Trigger : SHADOW UPDATE & DEVICE.STATUSCHANGE

Device Data Management

Workshop 5 : Temperature and device status change notification

Select Device1 to create Trigger

The screenshot shows a device management interface for a MQTTBox named 'Device1'. The left sidebar includes links for Overview, Device Lists, Device Groups, Event Hooks, and Setting. The main panel displays the device's description as 'MQTTBox' and its status as 'Online'. A 'Key' section shows placeholder fields for Client ID, Token, and Secret, each with a 'Generate' button. Below these are tabs for Shadow, Schema, and Trigger, with the Trigger tab highlighted and a red box and cursor icon pointing to it. A blue navigation bar at the bottom features icons for back, forward, tree view, and search, followed by a 'Select a node...' dropdown containing 'object {0}' and '(empty object)'.

Device Data Management

Workshop 5 : Temperature and device status change notification

The screenshot shows a device configuration interface for a device named 'Device1'. The left sidebar includes options like Overview, Device Lists, Device Groups, Event Hooks, and Setting. The main area displays the device's description ('MQTTBox') and key information (Client ID, Token, Secret, Status: Online). A toolbar at the bottom has tabs for Shadow, Schema, Trigger, and Code. A callout box highlights the 'Code' tab, which is currently selected. The interface is powered by ace.

Device Data Management



NECTEC
a member of NSTDA

Workshop 5 : Temperature and device status change notification

```
{  
  "enabled": true,  
  "trigger": [  
    {  
      "action": "LINE_NOTIFY",  
      "event": "SHADOW.UPDATED",  
      "condition": "$.temperature!= $$.temperature",  
      "msg": "My temperature 2 was change from {{$$.temperature}} to {{$$.temperature}}.",  
      "option": {  
        "linetoken": "Your_Token_LINE"  
      },  
      {  
        "action": "LINE_NOTIFY",  
        "event": "DEVICE.STATUSCHANGED",  
        "msg": "My Device {{$$.statustext}}, statuscode: {{$$.status}}.",  
        "option": {  
          "linetoken": "Your_Token_LINE"  
        }  
      }  
    }  
  ]  
}
```

Turn on Trigger

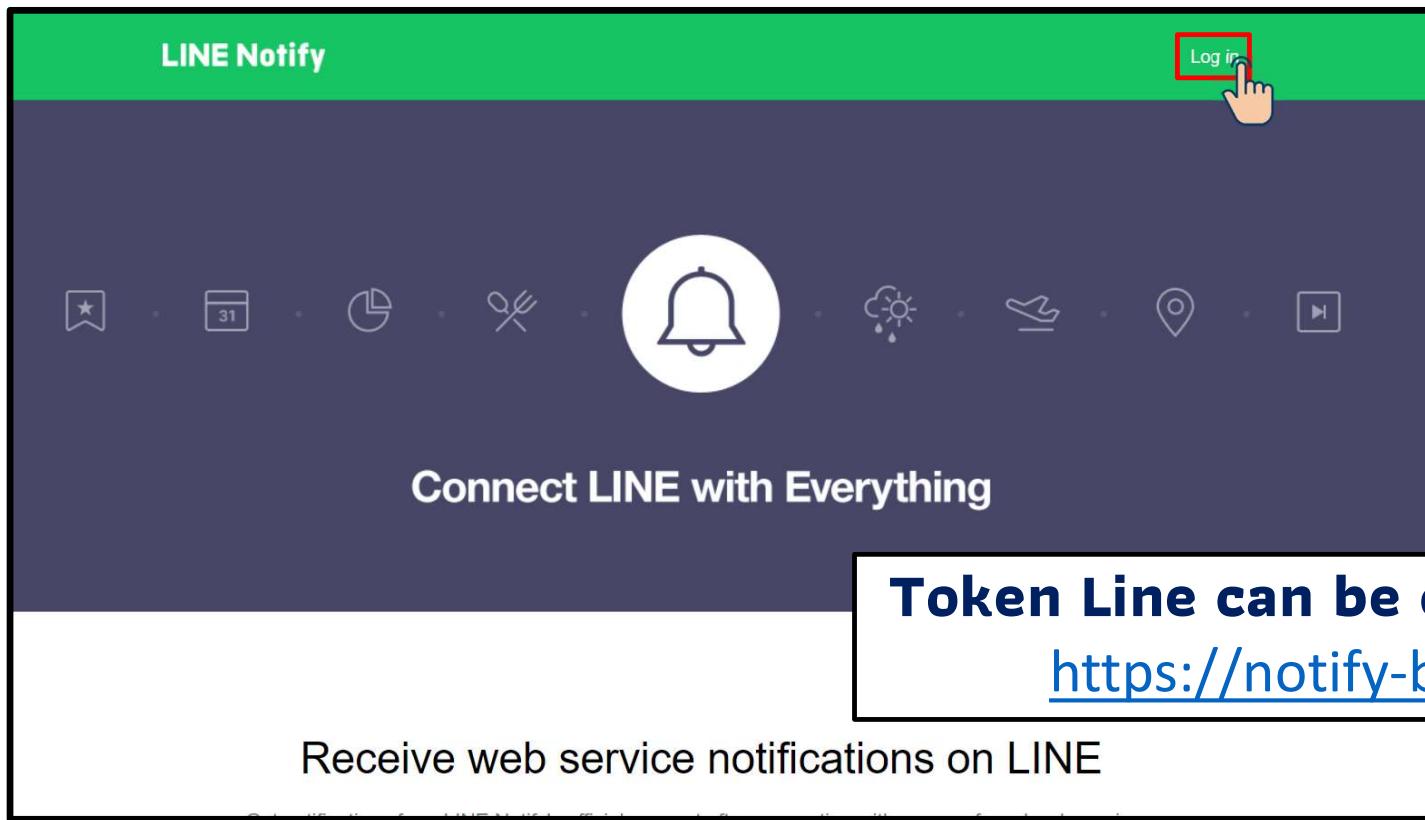
Copy this trigger code and paste to Device Trigger

Trigger Shadow Update : the condition is to take action whenever the temperature changes, with action : “LINE_NOTIFY” and the message is “My temperature 2 was changed from xx to xx.”

Trigger Status Change : The condition is to take action whenever device status changes, with action: “LINE_NOTIFY” and the message is “My device xx statuccode: xx”

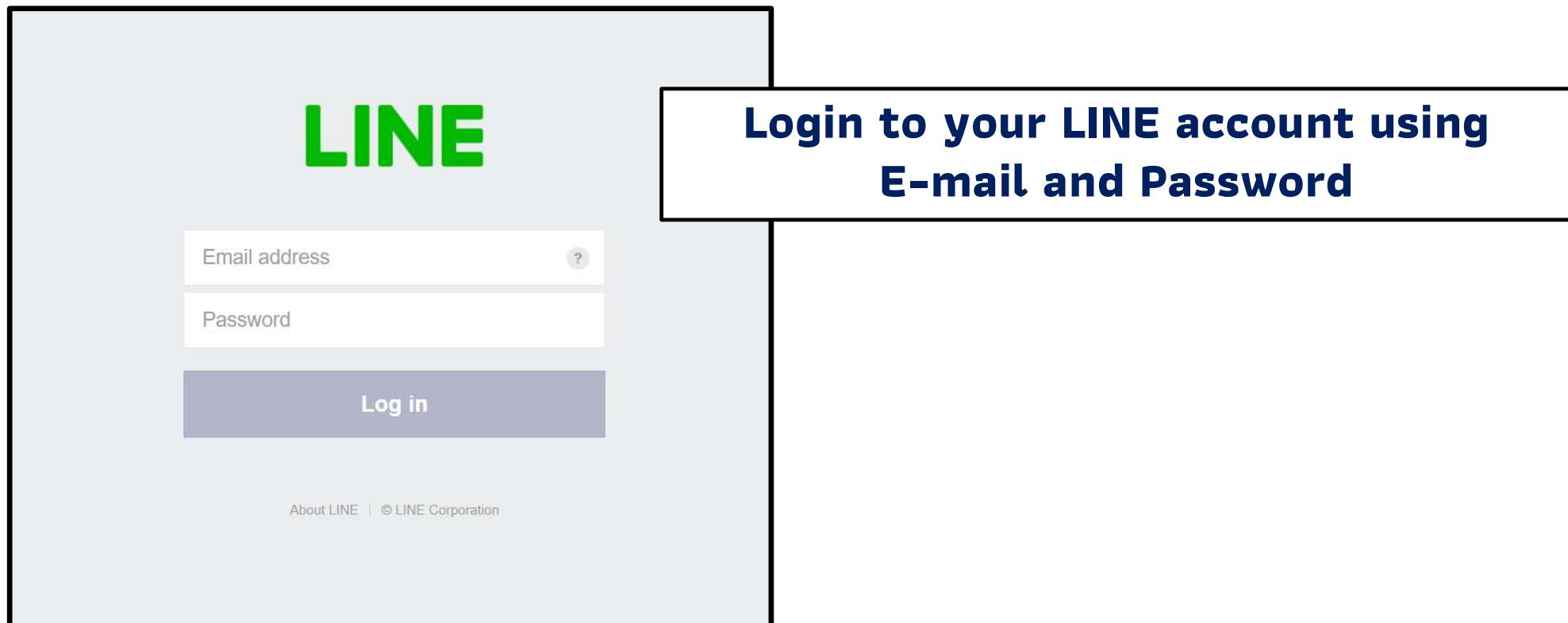
Device Data Management

Workshop 5 : Temperature and device status change notification



Device Data Management

Workshop 5 : Temperature and device status change notification



Device Data Management



NECTEC
a member of NSTDA

Workshop 5 : Temperature and device status change notification

The screenshot shows the LINE Notify web interface. At the top left is the text "LINE Notify". On the right is a user profile dropdown menu with the name "Piyawat" and a dropdown arrow. A red box highlights the "My page" option in the menu. Below the profile is a large white button with a hand icon pointing to it, labeled "My page". The main interface has a dark blue header with various icons: a star, a calendar (31), a pie chart, a scissor-like icon, a bell (highlighted with a red box), a sun with rain, a hand, a location pin, and a video camera. Below the header, the text "Connect LINE with Everything" is visible. At the bottom, a white box contains the text "Receive web service notifications on LINE".

**After login, select >> My page
as shown in the figure**

Workshop 5 : Temperature and device status change notification

**Scroll down and click on
[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

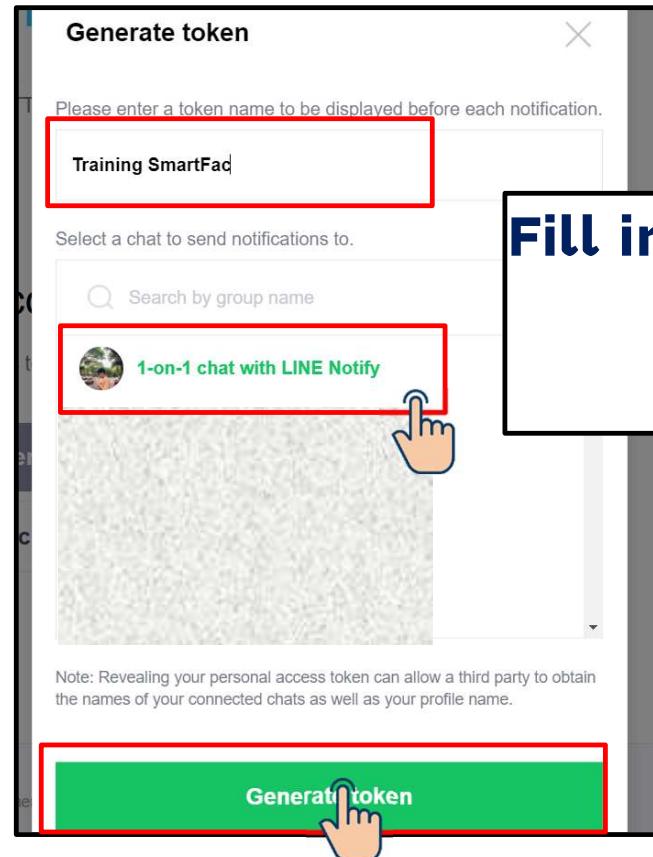
LINE Notify API Document



Device Data Management



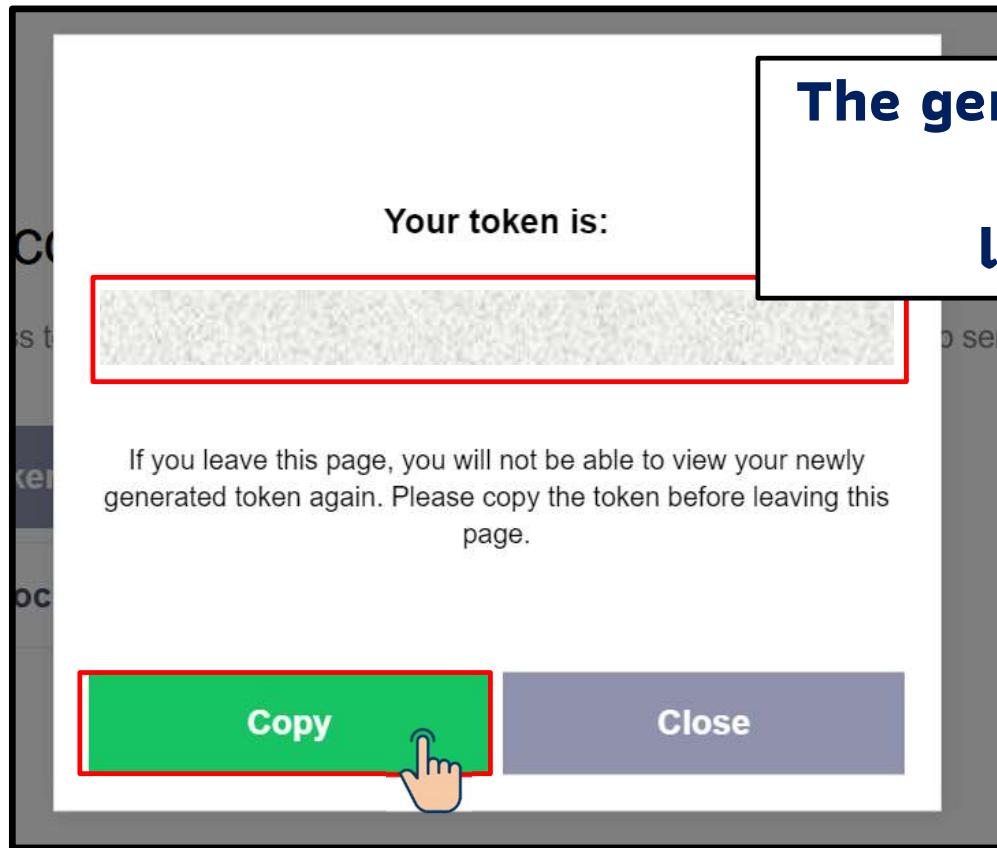
Workshop 5 : Temperature and device status change notification



**Fill in token name, and select 1-on-1 chat with LINE notify.
Click [Generate Token]**

Device Data Management

Workshop 5 : Temperature and device status change notification



The generated token is displayed. Copy and paste to linetoken in Device Trigger

Device Data Management



NECTEC
a member of NSTDA

Workshop 5 : Temperature and device status change notification

```
Shadow Schema Trigger
Code
1  [
2    "enabled": true,
3    "trigger": [
4      {
5        "action": "LINE_NOTIFY",
6        "event": "SHADOW.UPDATED",
7        "condition": "$.temperature!= $$.temperature",
8        "msg": "My temperature 2 was change from {$$.temperature} to {{$temperature}}",
9        "option": [
10          {
11            "linetoken": "Your_Token_LINE"
12          }
13        ],
14        {
15          "action": "LINE_NOTIFY",
16          "event": "DEVICE.STATUSCHANGED",
17          "msg": "My Device {$statustext}, statuscode: {{$status}}",
18          "option": [
19            {
20              "linetoken": "Your_Token_LINE"
21            }
22          ]
23        }
24      ]
25    ]
26  ]
Ln:22 Col:2
```

Device Data Management



NECTEC
a member of NSTDA

Workshop 5 : Temperature and device status change notification

The screenshot shows the MQTT publish interface and the MQTT shadow tree. The publish interface has the following steps highlighted:

- Topic to publish**: @shadow/data/update (Step 1)
- Payload**: { "data" : { "humidity" : 52, "temperature" : 28, "place" : "NECTEC Thailand" }} (Step 2)
- Publish** button (Step 3)

The payload message is set to:

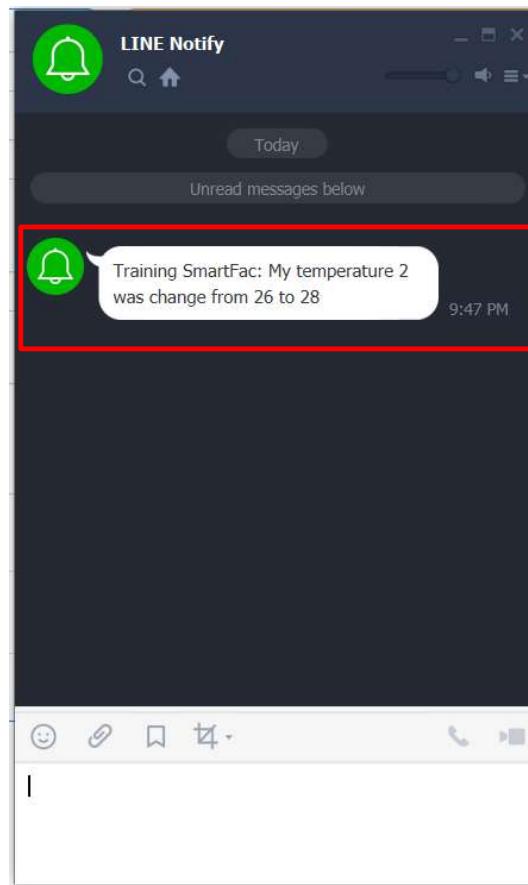
```
"data" : {"temperature" : 30}
```

The MQTT shadow tree shows the following state:

- object {3}
 - humidity : 52
 - place : NECTEC Thailand
 - temperature : 26

Device Data Management

Workshop 5 : Temperature and device status change notification



Temperature change notification from LINE

Device Data Management

Workshop 5 : Temperature and device status change notification

The screenshot shows the MQTTBox application interface. A red arrow points from the text "Device status change test" to the "Not Connected" button in the top navigation bar. Another red arrow points from the text "Device status change notification LINE" to a LINE Notify message window.

MQTTBox
MQTTBox Edit Help

Menu Not Connected Add publisher

Topic to publish @shadow/data/update

QoS 0 - Almost Once

Retain

Payload Type Strings / JSON / XML / Characters
e.g: {"hello": "world"}

Payload

```
{ "data": { "humidity": 52, "temperature": 28, "place": "NECTEC Thailand" }}
```

Publish

```
{"data": { "humidity": 52, "temperature": 28, "place": "NECTEC Thailand" }}  
topic:@shadow/data/update, qos:0, retain:false
```

Device status change test

Device status change notification LINE



NETPIE Dashboard



29 July 2024



Page 142

Dashboard

What is Dashboard?

NETPIE2020 Dashboard is a software panel for control and display data from a device. A developer creates and customizes Widget Plugins to suit user requirements, such as gauges, sliders, control buttons, and put Javascript commands for various actions.



Dashboard

Create a Dashboard

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

- PROJECT: helloNETPIE
- WORKSPACE: Overview, Device, Group, Event Hook, Console, **Dashboard** (highlighted with a red box), and Setting.

The main content area is titled "Dashboard" and shows the message "No Data". There is a search bar with the placeholder "input search text" and a "Create" button. A hand cursor icon is pointing at the "Create" button, which is highlighted with a red box.

Dashboard



NECTEC
a member of NSTDA

PROJECT + Add Project

Create a Dashboard

WORKSPACE

- Overview
- Device
- Group
- Event Hook
- Console
- Dashboard

SETTING

- Setting

Show 0 dashboard

input search text

Name	Device	Create Date

Create Dashboard

Dashboard Name

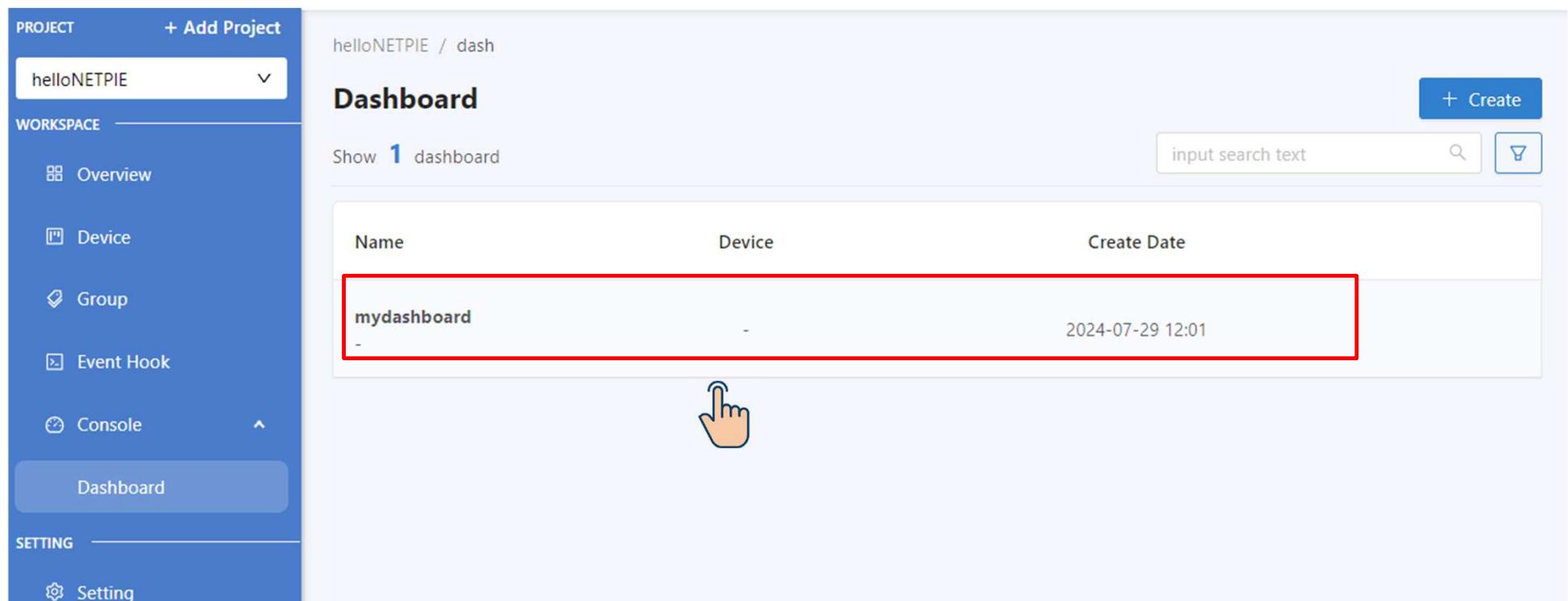
Dashboard Description

Description

Cancel SAVE

Dashboard

Create a Dashboard



The screenshot shows the helloNETPIE dashboard creation interface. On the left, a sidebar menu includes 'PROJECT' (helloNETPIE selected), 'WORKSPACE' (Overview, Device, Group, Event Hook), 'Console' (selected), and 'SETTING' (Setting). The main area displays the 'Dashboard' page for the 'helloNETPIE / dash' workspace. It shows 1 dashboard listed in a table with columns: Name, Device, and Create Date. The first row, 'mydashboard', is highlighted with a red border. A hand cursor icon is positioned over this row. The table also contains a '-' under the Device column and the date '2024-07-29 12:01' under Create Date.

Name	Device	Create Date
mydashboard	-	2024-07-29 12:01

Dashboard



Workshop 6 : Construct a Dashboard on NETPIE2020



DHT11



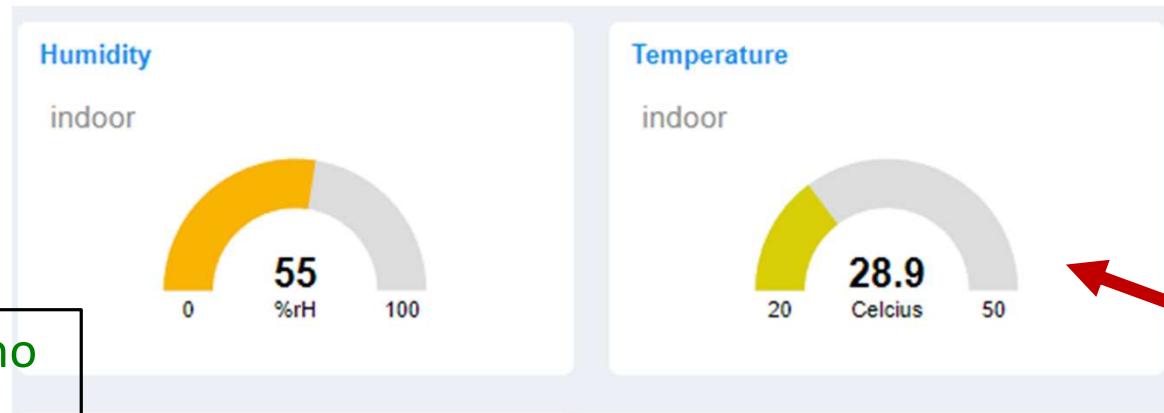
NodeMCU

@shadow/data/update

Publish : { "data": { "Temp" : 24, "Humi" : 58 }}



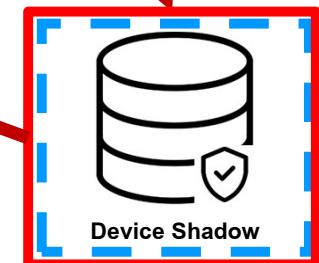
NETPIE2020



Use the Arduino
Program from
Workshop 4

Dashboard

"Temp" : 75
"Humi" : 58



Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 dashboard interface. On the left, there is a sidebar with the following sections:

- PROJECT**: Shows "helloNETPIE" as the selected project.
- + Add Project**: A button to add a new project.
- WORKSPACE**: A dropdown menu with options: Overview, Device, Group, Event Hook, Console, and Dashboard (which is currently selected).

The main area displays the "mydashboard" project, which is connected. It has two tabs: "Dashboard" and "Settings". The "Settings" tab is highlighted with a red box and has a hand cursor icon pointing at it. Above the tabs, there is a green button labeled "Connected". On the far right of the top bar, there are icons for Edit, Info, and More.

Dashboard



NECTEC
a member of NSTDA

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 dashboard configuration interface. On the left, there is a sidebar with two sections: WORKSPACE and SETTING. The WORKSPACE section contains links for Overview, Device, Group, Event Hook, Console, and Dashboard (which is currently selected). The SETTING section contains a Setting link. The main area is titled "Dashboard setting" and includes fields for "Name" (set to "mydashboard") and "Description". Below this, it says "Show 0 Dashboard datasource" and features a " + Add device" button, which is highlighted with a red box and has a hand cursor icon pointing at it. A table below lists "ID", "Alias", "Privileges", "Create Date", and a "Delete" column. At the bottom, it says "No device yet" and "You haven't connected any devices to freeboard yet.", with a small folder icon.

Dashboard



Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 dashboard interface. On the left, there is a sidebar with the following menu items:

- Overview
- Device
- Group
- Event Hook
- Console
- Dashboard
- Setting

The "Dashboard" item is currently selected. In the main area, there is a "Dashboard setting" panel. A modal window titled "Add Device" is open, prompting the user to "Select a device to add to the dashboard." The modal contains a dropdown menu labeled "Device" with the placeholder "Selects a device." Below the dropdown, a list of devices is shown, with "esp8266 (adb59c76-3be2-4d83-93b4-f75da4abba99)" highlighted with a red box and a hand cursor icon pointing at it. Other listed devices include "Device3 (f174d3dd-53d9-40a7-b2c3-46e99ae593e3)" and "Device1 (331be25f-75d1-4041-8cac-c34a1b26b084)". At the bottom of the modal are "Cancel" and "SAVE" buttons.

Dashboard



Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 dashboard interface. On the left, there is a sidebar with the following menu items:

- Overview
- Device
- Group
- Event Hook
- Console
- Dashboard
- Setting

The "Dashboard" item is currently selected. In the main area, there is a "Dashboard setting" panel. A modal window titled "Add Device" is open, prompting the user to "Select a device to add to the dashboard." The modal contains a dropdown menu labeled "Device" with the placeholder "Selects a device." Below the dropdown, a list of devices is shown, with "esp8266 (adb59c76-3be2-4d83-93b4-f75da4abba99)" highlighted with a red box and a hand cursor icon pointing at it. Other listed devices include "Device3 (f174d3dd-53d9-40a7-b2c3-46e99ae593e3)" and "Device1 (331be25f-75d1-4041-8cac-c34a1b26b084)". At the bottom of the modal are "Cancel" and "SAVE" buttons.

Dashboard



NECTEC
a member of NSTDA

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 dashboard interface. On the left, there is a sidebar with the following menu items: Overview, Device, Group, Event Hook, Console, Dashboard, Setting, and a collapsed section under 'TING'. The 'Setting' item is currently selected. In the main area, there is a 'Dashboard setting' card with fields for Name, Description, and Device ID. A modal window titled 'Add Device' is open, prompting the user to 'Select a device to add to the dashboard.' It contains fields for 'Device' (set to 'esp8266 (adb59c76-3be2-4d83-93b4-f75da4abba99)'), 'Alias' (set to 'esp8266'), and 'Privileges'. The 'Privileges' section includes options for 'Selects a data type.', 'Subscribe Message', 'Publish Message', 'Read Shadow' (which is checked and highlighted with a red box), 'Write Shadow' (which is being clicked by a cursor icon), 'Read Feed', and 'Write Feed'. A button labeled '+ Add device' is visible in the background.

Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows a user interface for managing dashboard data sources. On the left, a sidebar has 'Console' and 'Dashboard' buttons, with 'Setting' selected. A central panel displays a table titled 'Show 1 Dashboard datasource'. The table has columns for 'ID', 'Alias', 'Privileges', and 'Create Date'. One row is shown, with the ID 'adb59c76-3be2-4d83-93b4-f75da4abba99', alias 'esp8266', privileges 'R Shadow' and 'W Shadow', and a creation date of '2024-07-29 12:55'. The entire row is highlighted with a red box.

ID	Alias	Privileges	Create Date
adb59c76-3be2-4d83-93b4-f75da4abba99	esp8266	R Shadow W Shadow	2024-07-29 12:55

Dashboard



NECTEC
a member of NSTDA

Workshop 6 : Construct a Dashboard on NETPIE2020

Data recorded in Device Shadow

The screenshot shows the NETPIE2020 Device Shadow interface. On the left, there's a sidebar with tabs for "Shadow" (which is selected), "Schema", "Trigger", and "Feed". Below the tabs is a toolbar with icons for tree navigation, last update, and refresh. A dropdown menu shows "Tree" and "Last Update: 29-". The main area has a search bar with "Select a node..." and a tree view under "object {4}". Four nodes are listed: "humidity : 55", "temperature : 28.9", "light : 10", and "place : Farm 1". A red box highlights these four nodes. Red arrows point from each highlighted node to their corresponding gauge meters on the right. The right side features three gauges: "Humidity indoor" (55 %rH), "Temperature indoor" (28.9 Celcius), and "Light indoor" (10 lux). Each gauge has a scale from 0 to 100.

```
graph LR; SH[Shadow] --- S[Schema]; SH --- T[Trigger]; SH --- F[Feed]; SH --- Tree[Tree]; SH --- Refresh[Refresh]; SH --- LastUpdate[Last Update: 29-]; SH --- Search[Select a node...]; SH --- Object[object {4}]; SH --- Node1["humidity : 55"]; SH --- Node2["temperature : 28.9"]; SH --- Node3["light : 10"]; SH --- Node4["place : Farm 1"]; Node1 --> H[Humidity indoor]; Node2 --> T[Temperature indoor]; Node3 --> L[Light indoor]; Node4 --> Place[Place: Farm 1]
```

humidity : 55

temperature : 28.9

light : 10

place : Farm 1

Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 dashboard interface. On the left, there is a sidebar with the following sections:

- PROJECT**: A dropdown menu showing "helloNETPIE".
- WORKSPACE**: A list of workspace items:
 - Overview
 - Device
 - Group
 - Event Hook
 - Console
- Dashboard**: This item is highlighted with a blue background.

The main area is titled "mydashboard" with a "Connected" status indicator. It contains two tabs: "Dashboard" (which is selected) and "Settings". In the top right corner of the main area, there is a red-bordered button labeled "Edit" with a pencil icon. To the right of the "Edit" button, there is a small hand cursor icon pointing towards it.

Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 interface. On the left, a sidebar menu includes 'PROJECT' (with 'helloNETPIE' selected), 'WORKSPACE' (with 'Overview', 'Device', 'Group', 'Event Hook', 'Console', and 'Dashboard' listed), and a 'Dashboard' button at the bottom. The main workspace is titled 'mydashboard' and is 'Connected'. It contains a single panel labeled 'Dashboard'. In the top right corner of the workspace, there is a button labeled '+ Add Panel' with a red box around it, and a hand cursor icon pointing towards it.

Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020

PROJECT + Add Project

helloNETPIE

WORKSPACE

- Overview
- Device
- Group
- Event Hook
- Console

Dashboard

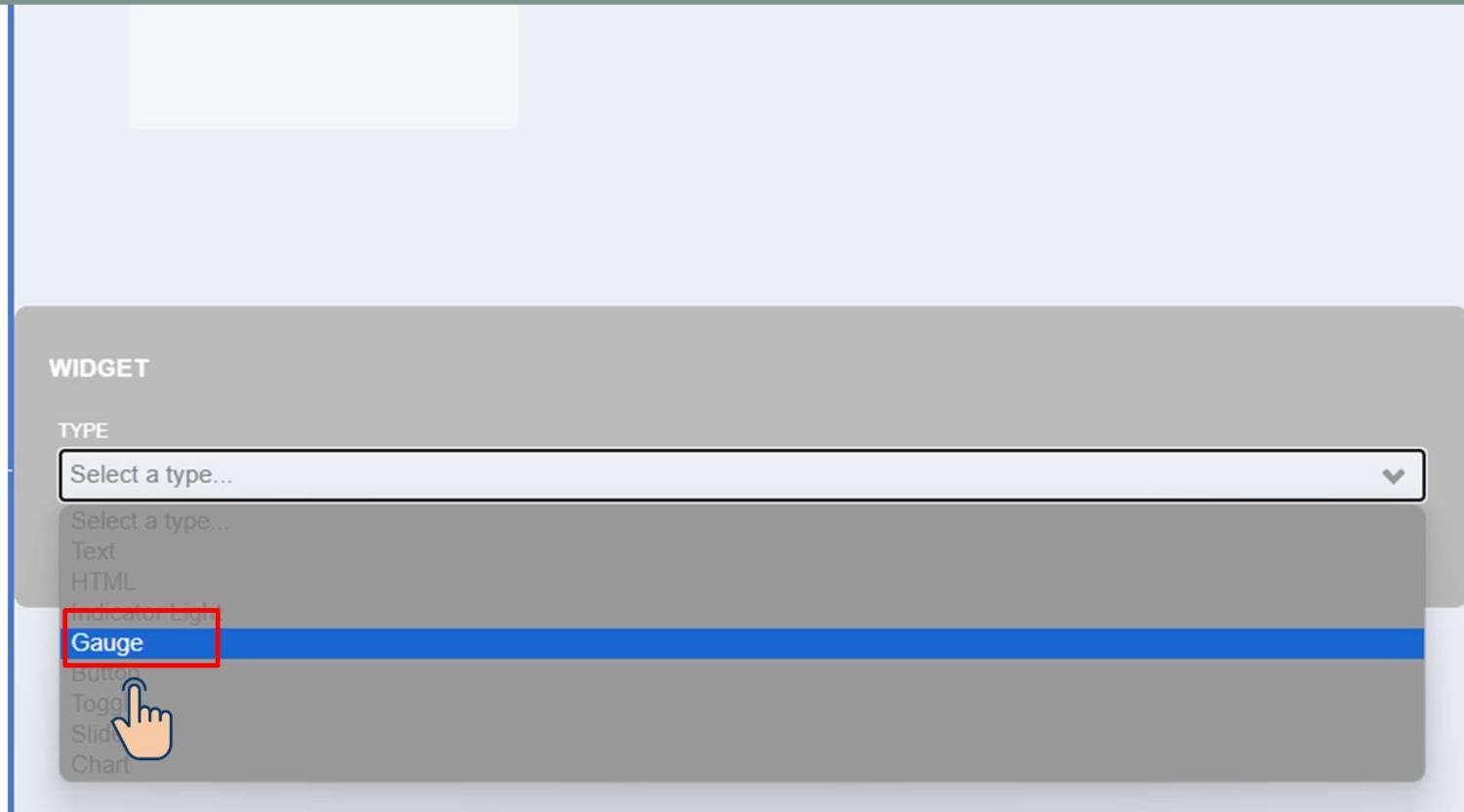
mydashboard Connected

+ Add Panel

X Cancel Save

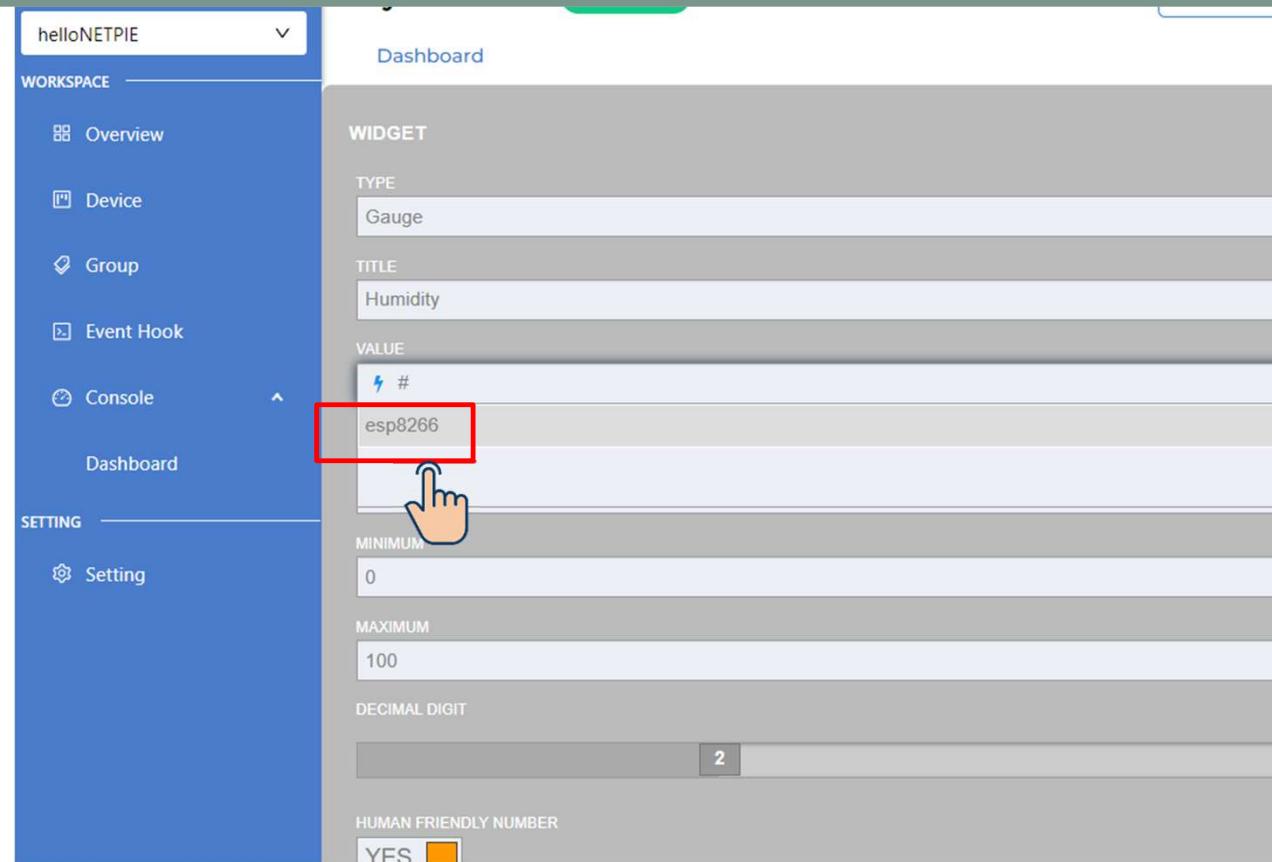
Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020



Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020



Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 dashboard configuration interface. On the left, there's a sidebar with 'WORKSPACE' and 'SETTING' sections. Under 'WORKSPACE', options like Overview, Device, Group, Event Hook, Console, and Dashboard are listed. Under 'SETTING', there's a 'Setting' option. The main area is titled 'WIDGET'. It has fields for 'TYPE' (set to 'Gauge'), 'TITLE' (set to 'Humidity'), and 'VALUE'. The 'VALUE' field contains a code snippet: `⚡ ##["esp8266"]["shadow"]`. Below this, a dropdown menu is open, showing 'humidity' and 'temperature'. The 'humidity' option is highlighted with a red box and a cursor icon. A large orange callout box with the text 'Select from pulldown menu' is overlaid on the 'humidity' option. The 'MINIMUM' and 'MAXIMUM' fields are set to '0'.

Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 web interface. On the left, a sidebar titled 'PROJECT' has '+ Add Project' and a dropdown menu set to 'helloNETPIE'. Below it, the 'WORKSPACE' section includes 'Overview', 'Device', 'Group', 'Event Hook', 'Console', and 'Dashboard'. The main area is titled 'mydashboard' with a 'Connected' status. It displays a 'Dashboard' section with a 'Humidity' gauge. The gauge has a yellow arc from 0 to 55, a grey arc from 55 to 100, and a central value of '55 %rH'. There are four small blue icons above the gauge: a plus sign, a square, a wrench, and a trash can.

Dashboard



Workshop 6 : Construct a Dashboard on NETPIE2020

The rest of dashboard can be constructed the same way...

Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 dashboard interface. On the left, there is a sidebar with sections for PROJECT, WORKSPACE, and SETTING. Under PROJECT, 'helloNETPIE' is selected. Under WORKSPACE, 'mydashboard' is shown as 'Connected'. Under SETTING, 'Dashboard' is selected. The main area displays three analog gauges: Humidity (55 %rH), Temperature (28.9 Celcius), and Light (10 lux). Below the gauges, the text 'Farm 1' is displayed.

PROJECT + Add Project

helloNETPIE

WORKSPACE

Overview

Device

Group

Event Hook

Console

Dashboard

mydashboard Connected

Dashboard Settings

Humidity indoor 55 %rH

Temperature indoor 28.9 Celcius

Light indoor 10 lux

Place

Farm 1

Dashboard



NECTEC
a member of NSTDA

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 dashboard configuration interface. On the left, a sidebar has 'Console' at the top, followed by a 'Dashboard' section, then a 'SETTING' section with 'Setting' selected. The main area is titled 'Show 1 Dashboard datasource'. It contains a table with one row. The row is highlighted with a red border. The columns are labeled 'ID', 'Alias', 'Privileges', and 'Create Date'. The data in the row is: ID - adb59c76-3be2-4d83-93b4-f75da4abba99, Alias - esp8266, Privileges - R Shadow, W Shadow, R Feed, and Create Date - 2024-07-29 12:55.

ID	Alias	Privileges	Create Date
adb59c76-3be2-4d83-93b4-f75da4abba99	esp8266	R Shadow, W Shadow, R Feed	2024-07-29 12:55

Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020

WIDGET

TYPE

Select a type...

Select a type...

Text

HTML

Indicator Light

Gauge

Button

Toggle

Slider

Chart



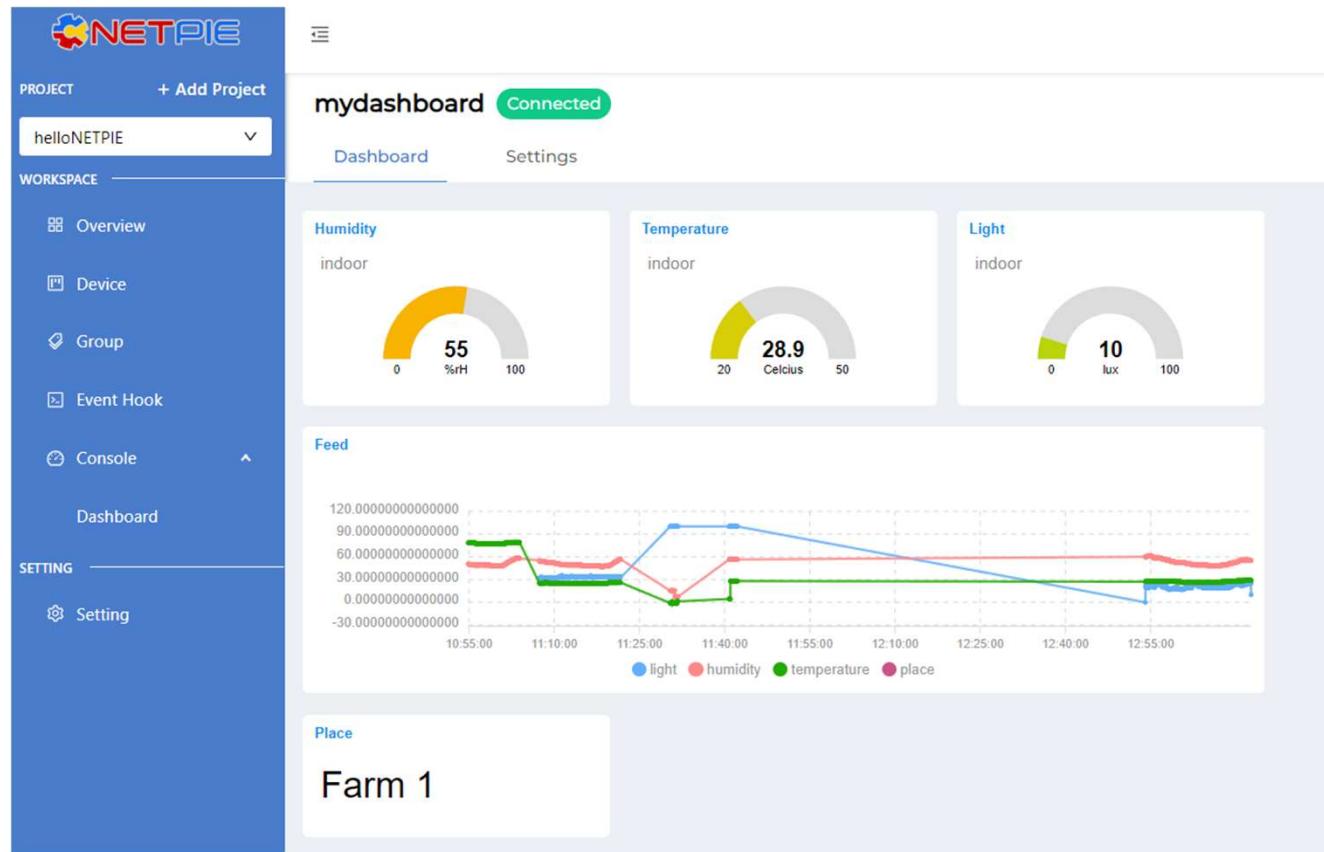
Dashboard

Workshop 6 : Construct a Dashboard on NETPIE2020

The screenshot shows the NETPIE2020 dashboard configuration interface. On the left, there's a sidebar with 'WORKSPACE' and 'SETTING' sections. Under 'WORKSPACE', options include Overview, Device, Group, Event Hook, Console, and Dashboard (which is selected). Under 'SETTING', there's a Setting option. The main area is titled 'WIDGET' and contains fields for 'TYPE' (set to 'Chart'), 'TITLE' (empty), 'DATA SOURCE' (containing the value '#["esp8266"]["feed"]' which is highlighted with a red box), 'DEVICE' (dropdown), 'VARIABLE' (dropdown), 'EDITOR' (button), 'FILTER' (empty), and 'QUERY FROM THE LAST' (set to '6 Hour'). A note below says 'Display data points on the last ... ago'. At the bottom, there's a 'STRICT TIME RANGE' section with a 'NO' button.

Dashboard

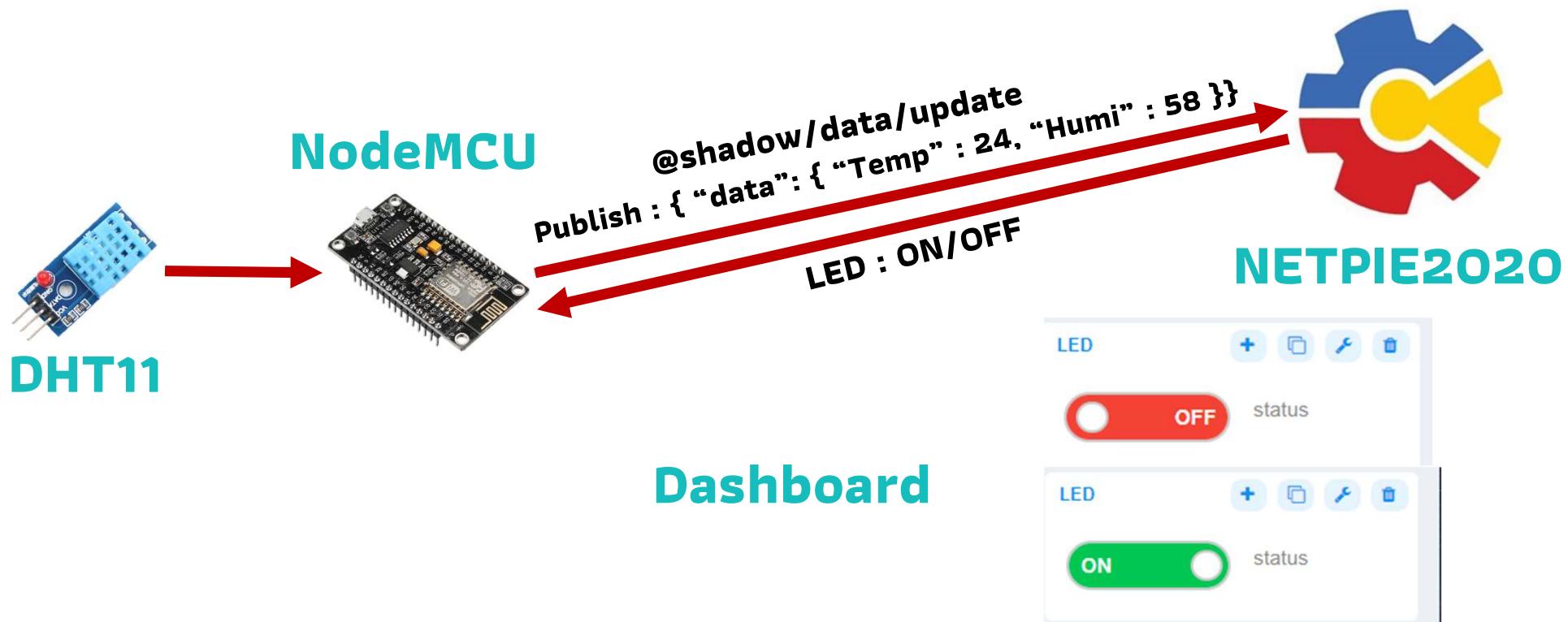
Workshop 6 : Construct a Dashboard on NETPIE2020



Dashboard



Workshop 7 : Control LED on NodeMCU via Dashboard



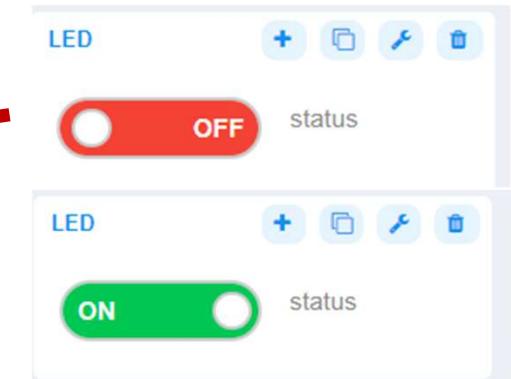
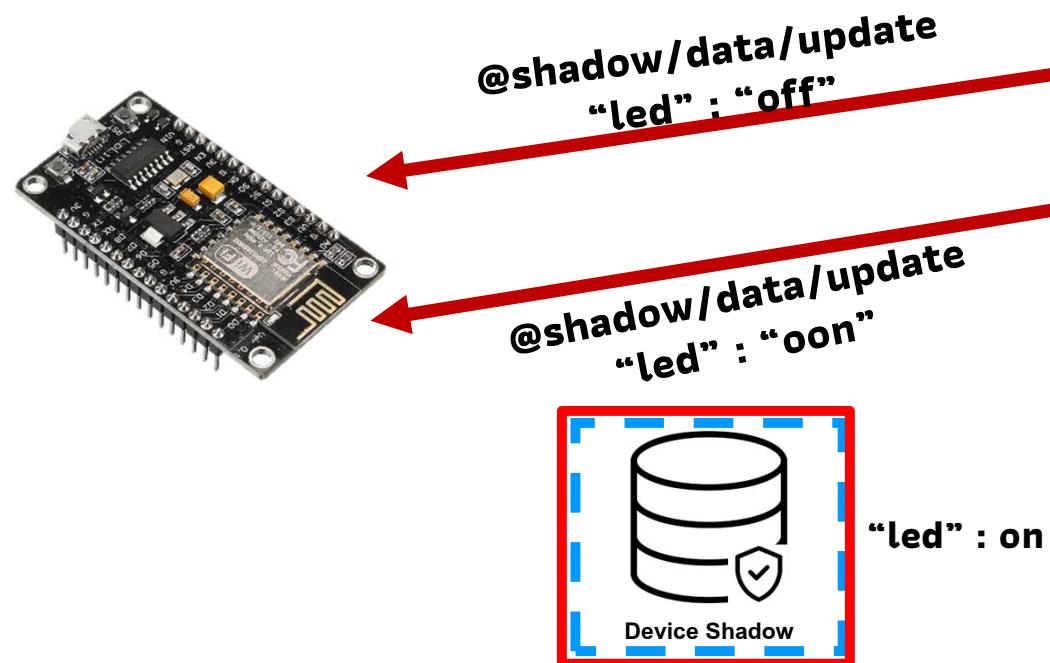
Dashboard



Workshop 7 : Control LEDs on NodeMCU via Dashboard

LED on NodeMCU can be controlled via Dashboard by 2 methods

1. Command via Shadow



Dashboard



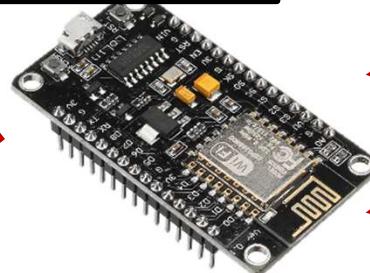
Workshop 7 : Control LEDs on NodeMCU via Dashboard

LED on NodeMCU can be controlled via Dashboard by 2 methods

1. Command via Shadow

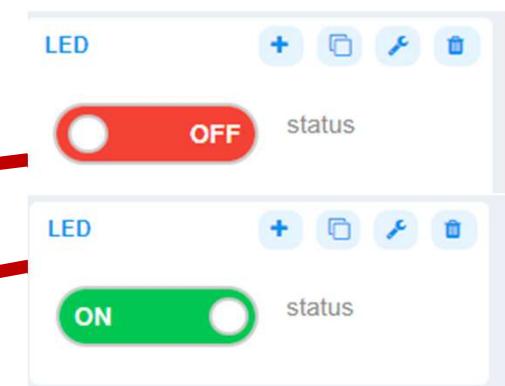
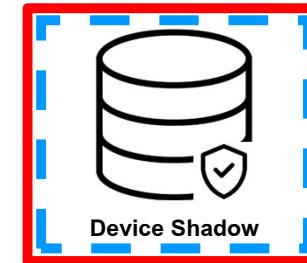
Message sent via shadow has the following format

```
b'{"deviceid":"085676f3-6c9d-44c3-a96f-c94f90412728","data":{"led":"on"},"rev":3,"modified":1579864929867}'
```



@shadow/data/update
"led" : "off"

@shadow/data/update
"led" : "oon"



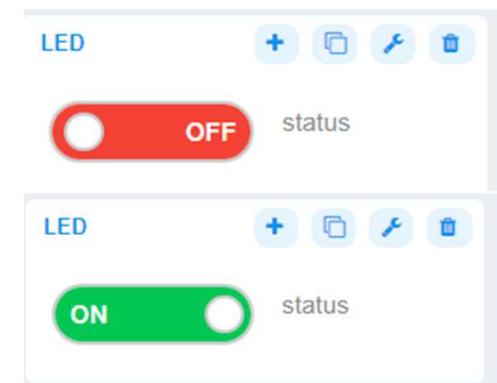
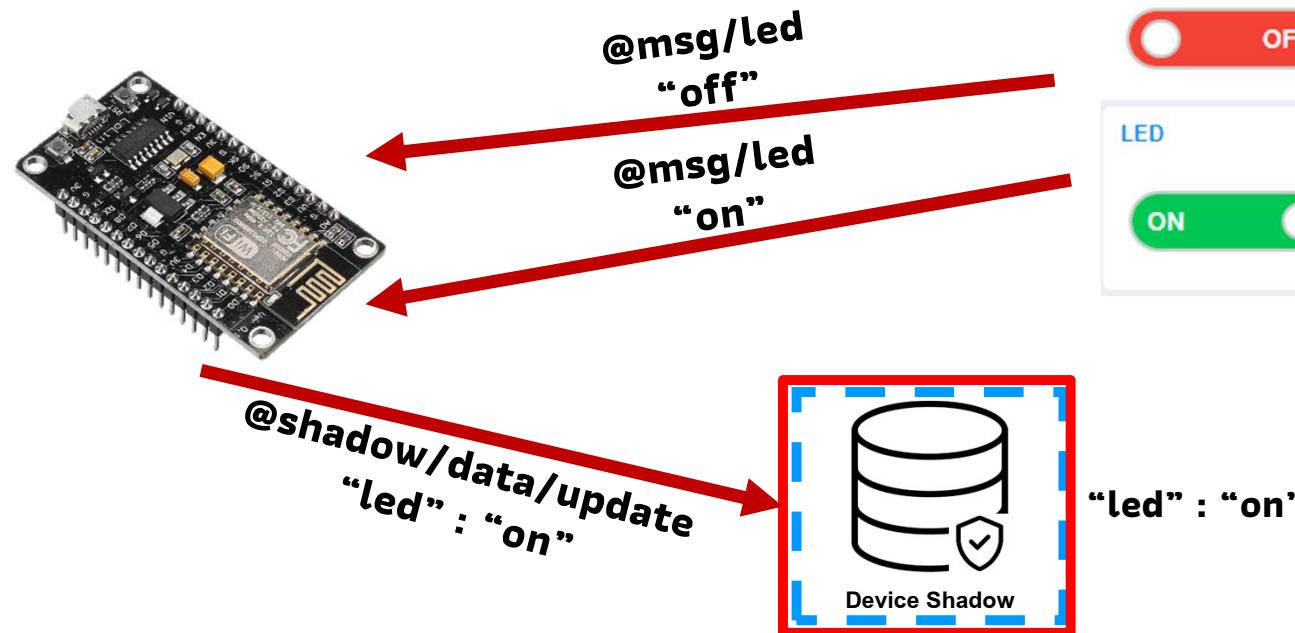
Dashboard



Workshop 7 : Control LEDs on NodeMCU via Dashboard

LED on NodeMCU can be controlled via Freeboard by 2 methods

2. Command by Message



Dashboard



Workshop 7 : Control LEDs on NodeMCU via Dashboard

My preferred method

Formulate command structure

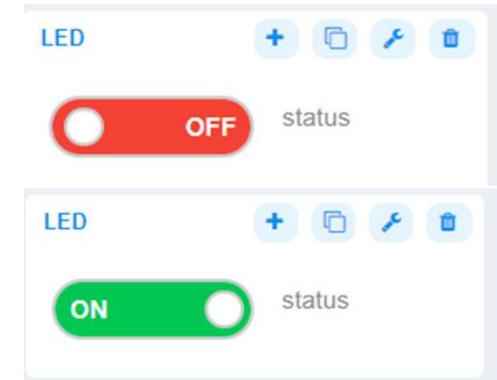


@msg/cmd
“led=off”

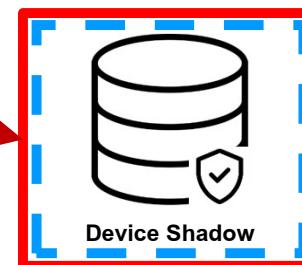
@msg/cmd
“led=on”

@shadow/data/update
“led” : “on”

Command = parameter



“led” : “on”



Dashboard

Workshop 7 : Control LEDs on NodeMCU via Dashboard

WOKWI SAVE SHARE Lect6_ws7_cmdparm sketch.ino

```
117 void loop() {
118     newDHTRead = millis();
119     if (newDHTRead<lastDHTRead) lastDHTRead = newDHTRead; // prevent overflow
120     if(newDHTRead - lastDHTRead > 2000){
121         lastDHTRead = newDHTRead;
122         float humidity = dht.readHumidity();
123         float temperature = dht.readTemperature();
124         uint16_t light = 100; //lightSensor.GetLightIntensity();
125         String place = "Farm 1";
126         if (!client.connected()){
127             reconnect();
128         }
129         client.loop();
130         String data = "{\"data\": {\"humidity\": " + String(humidity) + ", \"t
131         + ", \"light\": " + String(light)+ ", \"place\": \" + String(place)
132         Serial.println(data);
133         data.toCharArray(msg, (data.length() + 1));
134         client.publish("@shadow/data/update", msg);
135     }
136     while (Serial.available() > 0) { // detect new input
137         rcvdstring = Serial.readString();
138         newcmd = 1;
139     }
140     if (newcmd) { // execute this part only when new command received
141         cmdInt(); // invoke the interpreter
142         newcmd = 0;
143     }
144     delay(100);
145 }
```

Simulation

<https://wokwi.com/projects/404734188988206081>

Device Data Management

Workshop 7 : Control LEDs on NodeMCU via Dashboard

Open ws7_cmdparm.ino

**Coding in ws7_cmdparm.ino
consists of 3 parts**

Part 1 Library and variable declaration

1

**Variable and LED output pin
declaration**

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
#include <Wire.h>
#include <BH1750FVI.h>

#define LAMP D5
#define VALVE D6
#define FERTILIZER D7

#define DHTPIN D4
#define DHTTYPE DHT11

const char* ssid = "dew2G";
const char* password = "dewninja666";
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "adb59c76-3be2-4d83-93b4-f75da4abba99";
const char* mqtt_username = "A5pGYWRFXjkkqorVLM95Qn4BUbqWbLvf";
const char* mqtt_password = "5Uz6sUqUPiHFPH7gWPP1uFWS6bDyKZiv";

WiFiClient espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);
uint8_t ADDRESSPIN = 15; // D8, not used
BH1750FVI::eDeviceAddress_t DEVICEADDRESS = BH1750FVI::k_DevAddress_L;
BH1750FVI::eDeviceMode_t DEVICEMODE = BH1750FVI::k_DevModeContHighRes;
```

Device Data Management

Workshop 7 : Control LEDs on NodeMCU via Dashboard

2

Part 2 various functions

MQTT connection function

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

Subscribe Topic sent via Dashboard

Device Data Management

Workshop 7 : Control LEDs on NodeMCU via Dashboard

2 Part 2 various functions

Setup function

```
void setup() {  
  
    pinMode(LAMP, OUTPUT);      // Grow lamp intensity 0 - 100%  
    pinMode(VALVE, OUTPUT);     // water valve open 0 - 100%  
    pinMode(FERTILIZER, OUTPUT); // turn off/on fertilizer unit  
  
    Serial.begin(115200);  
    Serial.println();  
    Serial.print("Connecting to ");  
    Serial.println(ssid);  
    WiFi.begin(ssid, password);  
    while (WiFi.status() != WL_CONNECTED) {  
        delay(500);  
        Serial.print(".");  
    }  
}
```

Assign callback function to MQTT

```
Serial.println("");  
Serial.println("WiFi connected");  
Serial.println("IP address: ");  
Serial.println(WiFi.localIP());  
client.setServer(mqtt_server, mqtt_port);  
client.setCallback(callback);  
  
dht.begin();  
LightSensor.begin();  
}  
}
```

Workshop 7 : Control LEDs on NodeMCU via Dashboard

2 Part 2 callback functions

```
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/cmd") {  
        rcvdstring=message;  
        cmdInt(); // call command interpreter  
    }  
}
```

MQTT message receive function

Commands to display MQTT topic and message

If topic is @msg/cmd,
call command intepreter

Workshop 7 : Control LEDs on NodeMCU via Dashboard

3

```
void cmdInt(void)
{
    rcvdstring.trim(); // remove leading&trailing whitespace, if any
    // find index of separator '='
    sepIndex = rcvdstring.indexOf('=');
    if (sepIndex== -1) { // no parameter in command
        cmdstring = rcvdstring;
        noparm = 1;
    }
    else {
        // extract command and parameter
        cmdstring = rcvdstring.substring(0, sepIndex);
        cmdstring.trim();
        parmstring = rcvdstring.substring(sepIndex+1);
        parmstring.trim();
        noparm = 0;
    }
}
```

Part 3 command interpreter

Workshop 7 : Control LEDs on NodeMCU via Dashboard

```
if (cmdstring.equalsIgnoreCase("lamp"))    {  
    if (noparm==1)    {  
        Serial.println("Lamp intensity = "+(String)map(lampvalue,0,1023,0,100) + " %");  
    }  
    else    {  
        parmvalint = parmstring.toInt();  
        if (parmvalint > 100) parmvalint = 100; // limit to 100%  
        else if (parmvalint<0) parmvalint = 0;  
        lamp = parmvalint;  
        lampvalue = map(lamp,0,100,0,1023);  
        analogWrite(LAMP,lampvalue);  
        datastr = "{\"data\":{\"lamp\":"+ String(lamp)+"}}";  
        datastr.toCharArray(msg, (datastr.length() + 1));  
        client.publish("@shadow/data/update", msg);  
        Serial.print("Lamp set to ");  
        Serial.print(lamp);  
        Serial.println(" %");  
        Serial.println(datastr);  
    }  
}
```

3

Part 3 command interpreter

Device Data Management

1



Workshop 7 : Control LEDs on NodeMCU via Dashboard

```
else if (cmdstring.equalsIgnoreCase("valve"))    {  
    if (noparm==1)    {  
        Serial.println("Valve opening = "+(String)map(valvevalue,0,1023,0,100) + " %");  
    }  
    else    {  
        parmvalint = parmstring.toInt();  
        if (parmvalint > 100) parmvalint = 100; // limit to 100%  
        else if (parmvalint<0) parmvalint = 0;  
        valve = parmvalint;  
        valvevalue = map(valve,0,100,0,1023);  
        analogWrite(VALVE,valvevalue);  
        datastr = "{\"data\":{\"valve\":" + String(valve)+"}}";  
        datastr.toCharArray(msg, (datastr.length() + 1));  
        client.publish("@shadow/data/update", msg);  
        Serial.print("Valve set to ");  
        Serial.print(valve);  
        Serial.println(" %");  
        Serial.println(datastr);  
    }  
}
```

3

Part 3 command interpreter

Device Data Management

Workshop 7 : Control LEDs on NodeMCU via Dashboard

```
else if (cmdstring.equalsIgnoreCase("fertilizer")) { // fertilizer feed
    if (noparm==1) {
        Serial.print("Current fertilizer state = ");
        Serial.println(fertilizer_state);
    }
    else {
        if (parmstring.equalsIgnoreCase("on")){
            digitalWrite(FERTILIZER,1);
            client.publish("@shadow/data/update", "{\"data\" : {\"fertilizer\" : \"on\"}}");
            Serial.println("FERTILIZER ON");
            fertilizer_state = parmstring;
        }
        else if (parmstring.equalsIgnoreCase("off")) {
            digitalWrite(FERTILIZER,0);
            client.publish("@shadow/data/update", "{\"data\" : {\"fertilizer\" : \"off\"}}");
            Serial.println("FERTILIZER OFF");
            fertilizer_state = parmstring;
        }
        else Serial.println("Invalid fertilizer state");
    }
}
```

3

Part 3 command interpreter

Device Data Management

Workshop 7 : Control LED on NodeMCU via Freeboard

3

Part 3 loop function

Use millis[] for timing instead of delay[] to avoid blocking the execution while waiting Freeboard message

```
void loop() {
    newDHTRead = millis();
    if (newDHTRead < lastDHTRead) lastDHTRead = newDHTRead; // prevent overflow
    if(newDHTRead - lastDHTRead > 2000){
        lastDHTRead = newDHTRead;
        float humidity = dht.readHumidity();
        float temperature = dht.readTemperature();
        uint16_t light = LightSensor.GetLightIntensity();
        String place = "Farm 1";
        if (!client.connected()) {
            reconnect();
        }
        client.loop();
        String data = "{\"data\": {\"humidity\": " + String(humidity) + ", \"temperature\": " + String(temperature)
        + ", \"light\": " + String(light)+ ", \"place\": \" " + String(place) + "\"}}";
        Serial.println(data);
        data.toCharArray(msg, (data.length() + 1));
        client.publish("@shadow/data/update", msg);
    }
}
```



Dashboard

Workshop 7 : Control LED on NodeMCU via Dashboard

Device Schema in JSON format

```
{  
  "additionalProperties": false,  
  "properties": {  
    "humidity": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      }  
    },  
    "temperature": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      }  
    },  
    "transform": {  
      "expression": "[[$.temperature]*1.8] + 32"  
    }  
  },  
  "type": "number"  
},  
  "place": {  
    "operation": {  
      "store": {  
        "ttl": "7d"  
      }  
    },  
    "type": "string"  
  },  
  "lamp": {  
    "operation": {  
      "store": {  
        "ttl": "7d"  
      }  
    },  
    "type": "string"  
  }
```

Maybe easier to set
additionalProperties
to true

Add code on Schema for lamp, valve and fertilizer

Record **lamp** data with property
- Store for 7 days
- Variable type is number

Dashboard

Workshop 7 : Control LEDs on NodeMCU Dashboard

**Issue serial port commands to write lamp,
Valve, fertilizer on shadow. Ex. lamp=20**

189

Output Serial Monitor X

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM5')

```
FERTILIZER ON
{"data": {"humidity":55.00, "temperature":27.60, "light":8, "place": "Farm 1"}}
{"data": {"humidity":55.00, "temperature":27.60, "light":8, "place": "Farm 1"}}
{"data": {"humidity":55.00, "temperature":27.60, "light":9, "place": "Farm 1"}}
Lamp set to 20 %
{"data": {"lamp":20}}
{"data": {"humidity":55.00, "temperature":27.60, "light":9, "place": "Farm 1"}}
{"data": {"humidity":55.00, "temperature":27.60, "light":9, "place": "Farm 1"}}
Valve set to 30 %
{"data": {"valve":30}}
{"data": {"humidity":55.00, "temperature":27.60, "light":9, "place": "Farm 1"}}
```

Shadow Schema Trigger Feed i

Tree ▾ Last Update : 29-07-24 14:55

Select a node...

- object {7}
 - humidity : 59
 - temperature : 28
 - light : 8
 - place : Farm 1
 - lamp : 20
 - valve : 30
 - fertilizer : on

Dashboard

Workshop 7 : Control LEDs on NodeMCU Dashboard

The screenshot shows the mydashboard interface. On the left, there's a sidebar with 'PROJECT' (helloNETPIE), 'WORKSPACE' (Overview, Device, Group, Event Hook, Console, Dashboard), and 'SETTING' (Setting). The main area shows a device named 'mydashboard Connected'. A modal window titled 'Edit Device' is open, prompting 'Select a device to add to the dashboard.' It contains fields for 'Device' (set to 'adb59c76-3be2-4d83-93b4-f75da4abba99') and 'Alias' ('esp8266'). Under 'Privileges', three buttons are shown: 'Read Shadow X', 'Write Shadow X', and 'Read Feed X'. Below these are two buttons: 'Subscribe Message' and 'Publish Message', which are highlighted with a red border. At the bottom of the modal, there's a 'Write Feed' button. The footer of the modal shows the device ID 'adb59c76-3be2-4d83-93b4-f75da4abba99', the alias 'esp8266', and three status indicators: 'R Shadow' (orange), 'W Shadow' (orange), and 'R Feed' (blue).

Dashboard

Workshop 7 : Control LEDs on NodeMCU Dashboard

The screenshot shows the NetPie dashboard interface. On the left, there's a sidebar with 'PROJECT' and 'WORKSPACE' sections. Under 'WORKSPACE', 'Overview', 'Device', 'Group', 'Event Hook', and 'Console' are listed. Under 'SETTING', 'Setting' is selected. The main area is titled 'mydashboard' (Connected) and has tabs for 'Dashboard' and 'Settings'. The 'Settings' tab is active, showing 'Dashboard setting' with fields for 'Name' (mydashboard) and 'Description' (Description). Below this, it says 'Show 1 Dashboard datasource'. A table lists one datasource: ID (adb59c76-3be2-4d83-93b4-f75da4abba99), Alias (esp8266), and Privileges (R Message, W Message, R Shadow, W Shadow, R Feed). The 'Privileges' row is highlighted with a red box.

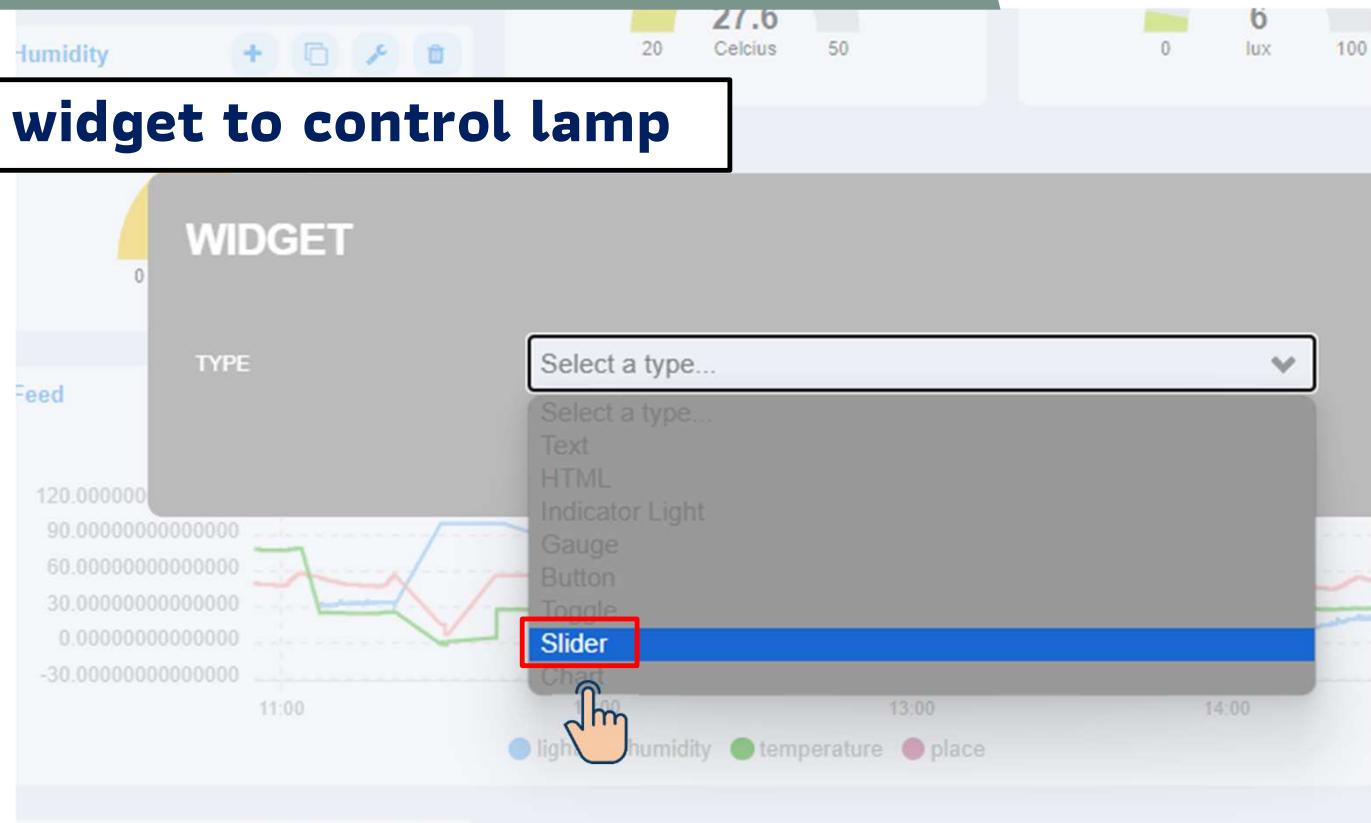
ID	Alias	Privileges
adb59c76-3be2-4d83-93b4-f75da4abba99	esp8266	R Message W Message R Shadow W Shadow R Feed

Dashboard



Workshop 7 : Control LEDs on NodeMCU Dashboard

Create a slider widget to control lamp



Dashboard



Workshop 7 : Control LEDs on NodeMCU Dashboard

Create a slider widget to control lamp

TYPE: Slider

FILLED COLOR: Grey

MAX VALUE: 100

STEP: 1

SENSITIVITY (MS): 200

INITIAL VALUE: 0

The default value set only the first time the widget is loaded.

AUTO UPDATED VALUE: `#["esp8266"]["shadow"]["lamp"]`

ONSTART ACTION:

ONSLIDE ACTION:

ONSTOP ACTION: `#["esp8266"].publishMsg("cmd","lamp="+value)`

+ DEVICE + VARIABLE EDITOR

+ DEVICE + VARIABLE EDITOR

+ DEVICE + VARIABLE EDITOR

Add some Javascript here. You can access to a slider attribute using variables 'value' and 'percent'.

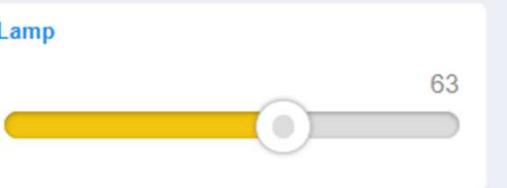
Add some Javascript here. You can access to a slider attribute using variables 'value' and 'percent'.

Add some Javascript here. You can access to a slider attribute using variables 'value' and 'percent'.

Dashboard

Workshop 7 : Control LEDs on NodeMCU Dashboard

Create a slider widget to control lamp



```
145 | ;((array
144 } )
```

Output Serial Monitor ×

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM5')

```
{"data": {"humidity":52.00, "temperature":28.90, "light":10, "place": "farm 1"}}
Message arrived [@msg/cmd] lamp=36
Lamp set to 36 %
{"data": {"lamp":36}}
{"data": {"humidity":52.00, "temperature":28.90, "light":10, "place": "Farm 1"}}
{"data": {"humidity":52.00, "temperature":28.90, "light":10, "place": "Farm 1"}}
 {"data": {"humidity":52.00, "temperature":28.90, "light":10, "place": "Farm 1"}}
Message arrived [@msg/cmd] lamp=63
Lamp set to 63 %
 {"data": {"lamp":63}}
 {"data": {"humidity":52.00, "temperature":28.90, "light":10, "place": "Farm 1"}}
```

Ln 142, Col 8 No

29 July 2024 Page 188

29 July 2024 Page 189

Dashboard



NECTEC
a member of NSTDA

Workshop 7 : Control LEDs on NodeMCU Dashboard

Group



Create a toggle widget to control fertilizer

The screenshot shows the NodeMCU Dashboard configuration interface. On the left, there's a sidebar with 'Console', 'Dashboard' (selected), and 'Setting'. Under 'Setting', there's a 'Setting' section. The main area is titled 'Create a toggle widget to control fertilizer'. It shows a configuration for a toggle switch:

- TOGGLE ON CAPTION:** ON
- TOGGLE OFF CAPTION:** OFF
- TOGGLE STATE:** `#[`esp8266`][`shadow`][`fertilizer`] == "on"` (highlighted with a red box)
- ONTOGGLEON ACTION:** `#["esp8266"].publishMsg("cmd", "fertilizer=on")` (highlighted with a red box)
- ONTOGGLEOFF ACTION:** `#["esp8266"].publishMsg("cmd", "fertilizer=off")` (highlighted with a red box)

Below the configuration fields, there are buttons for DEVICE, VARIABLE, EDITOR, and a note: 'Add a condition to switch a toggle state here. Otherwise it just toggle by click.'

Dashboard



Workshop 7 : Control LEDs on NodeMCU Dashboard

**It is left as an exercise to create another slider for valve
(represented by a small green LED).**

Dashboard

Workshop 7 : Control LEDs on NodeMCU Dashboard

The dashboard displays the following data:

- Humidity:** indoor, 47 %RH
- Temperature:** indoor, 29.3 Celcius
- Light:** indoor, 13 lux
- Fertilizer:** Status: ON
- Lamp:** %, 63
- Valve:** %, 61
- Place:** Farm 1

At the bottom right, there is a photograph of an Arduino Uno connected to a breadboard. The breadboard has various components and sensors attached, including a small screen displaying the sensor data.

```
134 }
135 while (Serial.available() > 0) {
136   rcvdstring = Serial.readString();
137   newcmd = 1;
138 }
139 if (newcmd) { // execute this par-
140   cmdInt(); // invoke the interpre-
141   newcmd = 0;
142 }
143 delay(100);
144 }
```

```
Output Serial Monitor ×

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM5')

Message arrived [@msg/cma] fertilizer=off
FERTILIZER OFF
("data": {"humidity":48.00, "temperature":29.30, "light":12, "place":
Message arrived [@msg/cmd] fertilizer=on
FERTILIZER ON
("data": {"humidity":47.00, "temperature":29.30, "light":13, "place":
("data": {"humidity":47.00, "temperature":29.30, "light":13, "place":
Message arrived [@msg/cmd] lamp=63
Lamp set to 63 %
("data": {"lamp":63})
("data": {"humidity":47.00, "temperature":29.30, "light":13, "place":
```

Note for the rest of this Powerpoint file



Since the rest of this presentation is not covered in my course.
The original slides are left intact.



RESTful API



29 July 2024

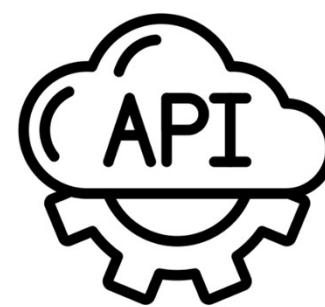
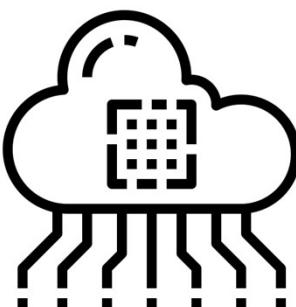


Page 194

What is RESTful API?

It is a channel for the device to call the Platform service via the RESTful API which uses HTTP protocol, suitable for integrating various systems, either existing or new development, without limitation by programming language or hardware. There are two groups of APIs:

1. Device API
2. Data Store API



Module 5 : Restful API

Device API

It is an API related to device management or calling.
The EndPoint is <https://api.netpie.io/v2/device>

Publish message to a topic

```
curl -X PUT "https://api.netpie.io/v2/device/message?topic=%40msg%2Ftest" -H  
"Authorization: Basic ClientID:Token" -H "Content-Type: text/plain" -d "message"
```

Read Shadow Data of Device

```
curl -X GET "https://api.netpie.io/v2/device/shadow/data" -H "Authorization:  
Basic ClientID:Token"
```

Write Shadow Data using Merge method

```
curl -X PUT "https://api.netpie.io/v2/device/shadow/data" -d "{data:{humid:63.7,  
temp:25.2}}" -H "Authorization: Device ClientID:Token"
```

Module 5 : Restful API



Device API

Write Shadow Data using Overwrite method

```
curl -X POST "https://api.netpie.io/v2/shadow/data" -d "{data:{temp:31.7}}" -H  
"Authorization: Device ClientID:Token"
```

Write to Shadow with specified time in the past

```
curl -X PUT " https://api.netpie.io/v2/shadow/data" -d "{data:{humid:61.9,  
temp:28.6,timestamp:1566863843262}}" -H "Authorization: Device ClientID:Token"
```

Module 5 : Restful API

Data Store API

It is an API that deals with retrieving data stored in Time series Database, where the Domain name of API is

<https://api.netpie.io/v2/feed> . The database is stored in KairosDB format, so querying the parameters can be done using the same format as KairosDB

Read data from Timeseries Database of Device

```
curl -X POST "https://api.netpie.io/v2/feed/api/v1/datapoints/query" -H "Content-Type: application/json" -H "Authorization: Bearer userToken" -d ' {  
  "start_relative": { "value": 1, "unit": "days" },  
  "metrics": [ { "name": "ClientID", "tags": [ { "attr": "temperature" } ] },  
  "limit": 50,  
  "group_by": [ { "name": "tag", "tags": [ { "attr": " " } ] } ],  
  "aggregators": [ { "name": "avg", "sampling": { "value": 1, "unit": "minutes" } } ] } }
```

Webpage to Generate Code for API calls

<https://trial-api.netpie.io/>



NETPIE2020 Quota



29 July 2024



Page 199

NETPIE2020 Quota



NETPIE2020 Quota

Connecting limit	10 Devices
Project	3 Projects
Real time message	9,000,000 Messages/Month
Data Storage	1,000,000 Points-Month
Shadow read/write	500,000 Operations/Month
API Call	800,000 Operations/Month
Trigger and Action	5,000 Operations/Month
Dashboard	3 Freeboards/Project
Freeboard Connection	3 Concurrent Views



Node-RED



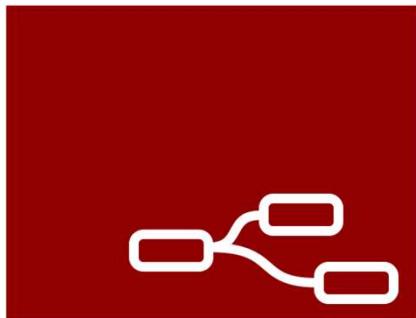
29 July 2024



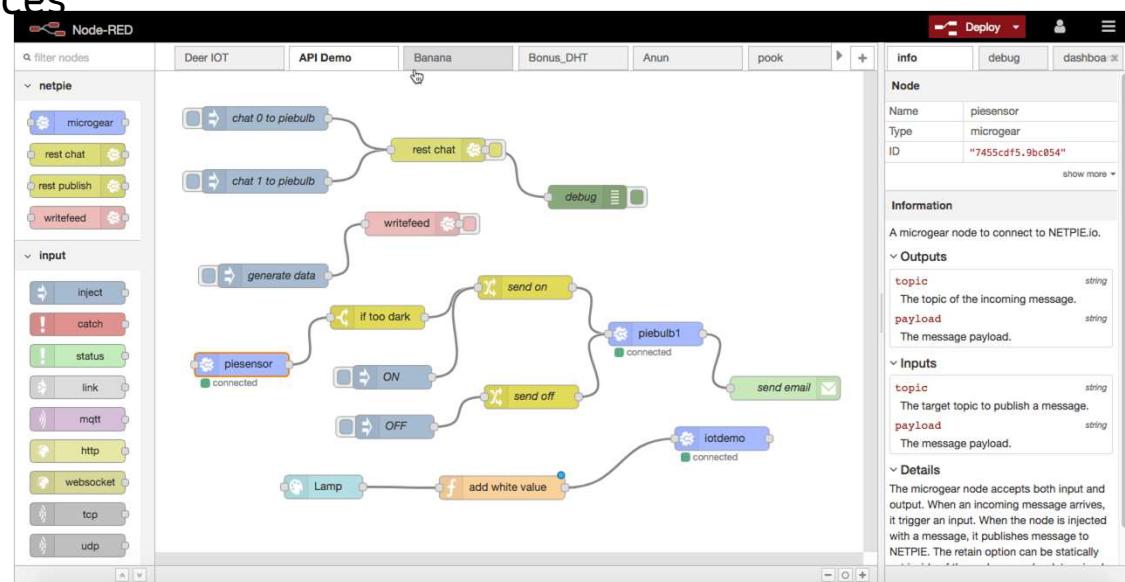
Page 201

What is Node-RED ?

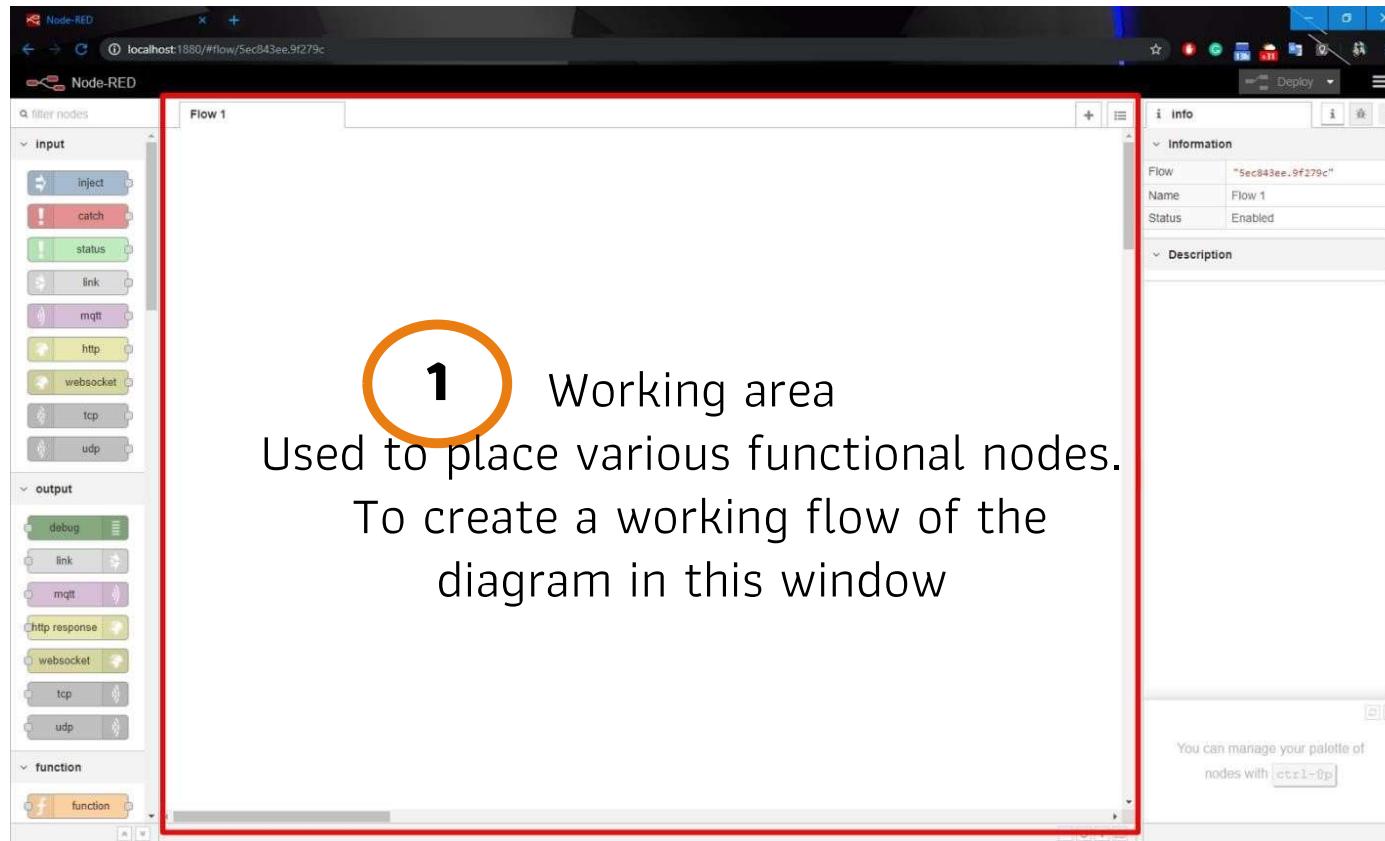
Node-RED is a tool for programmers to connect their hardware devices to APIs [Application Programming Interface] and online services in new ways. They're available for free as well as for paid portions. Node-RED can design APIs. To receive, calculate, convert, store, or connect to other services.



Node-RED



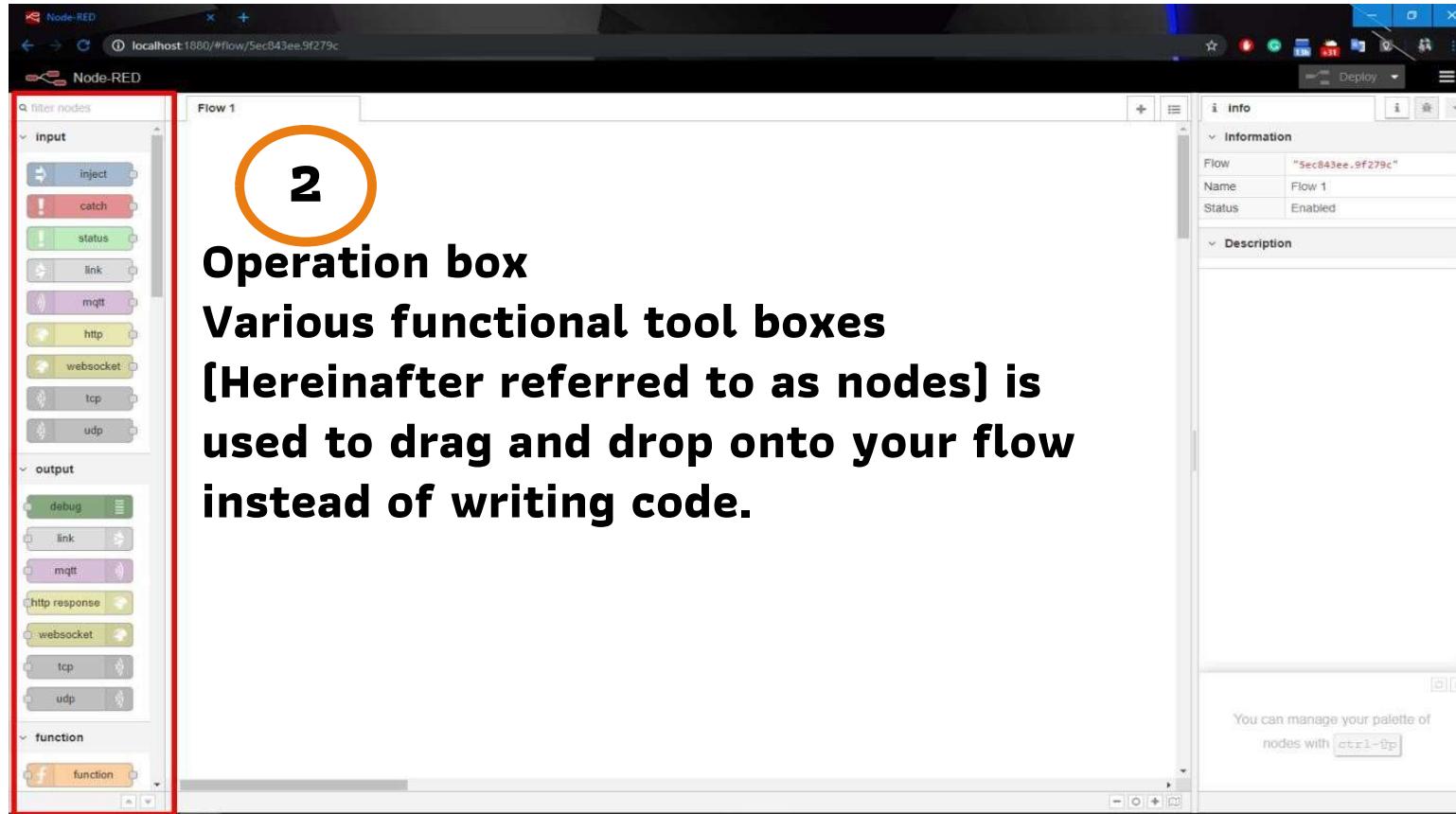
Component



1

Working area
Used to place various functional nodes.
To create a working flow of the
diagram in this window

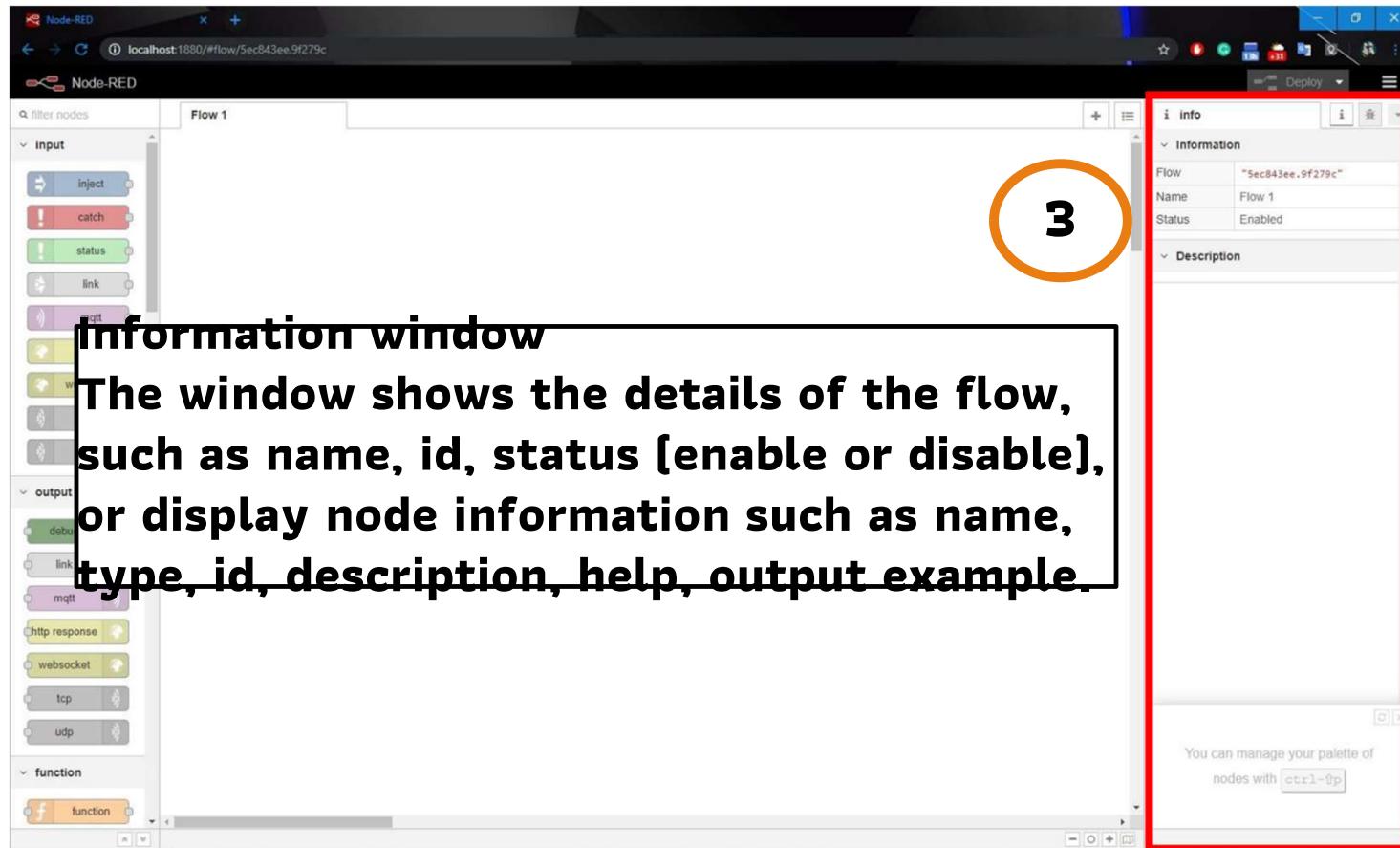
Component



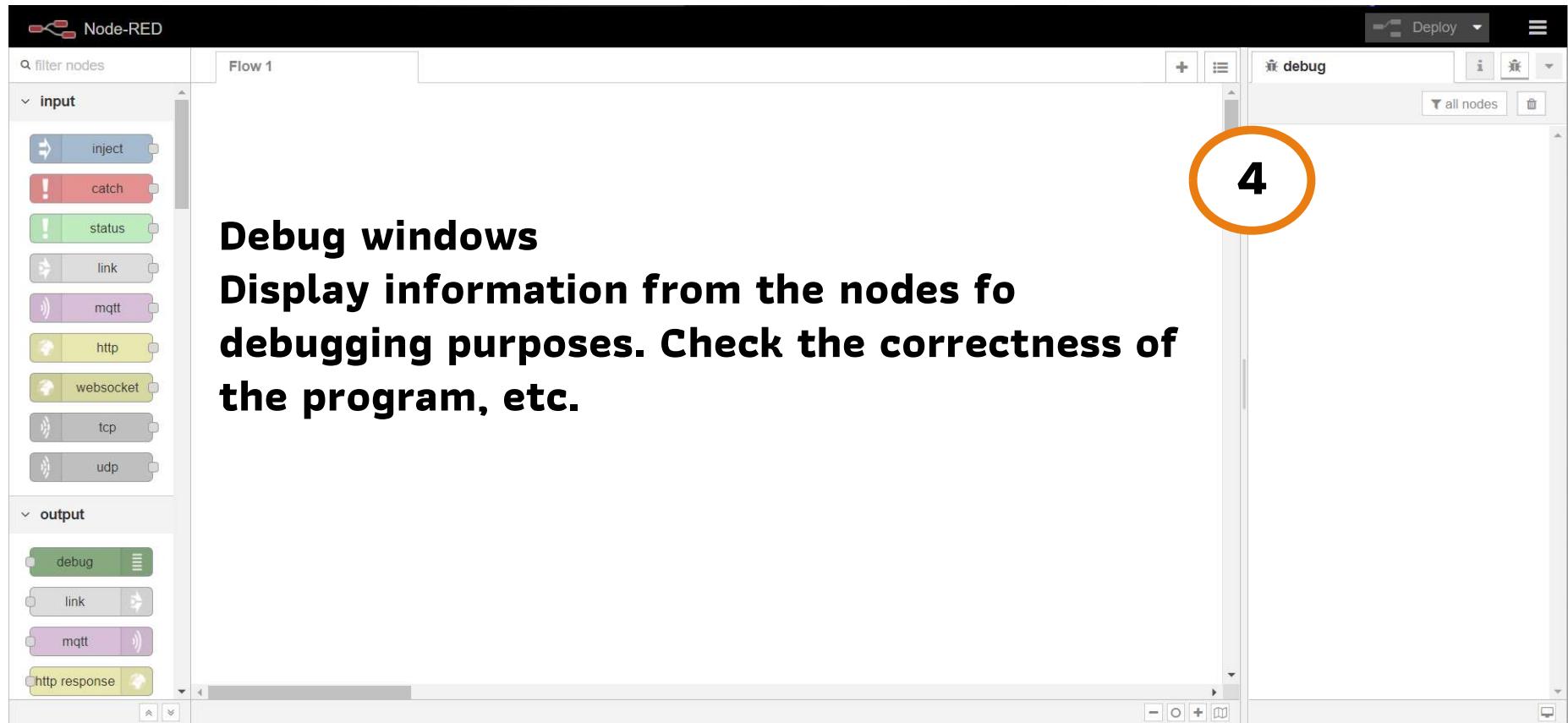
2
Operation box

**Various functional tool boxes
(Hereinafter referred to as nodes) is
used to drag and drop onto your flow
instead of writing code.**

Component



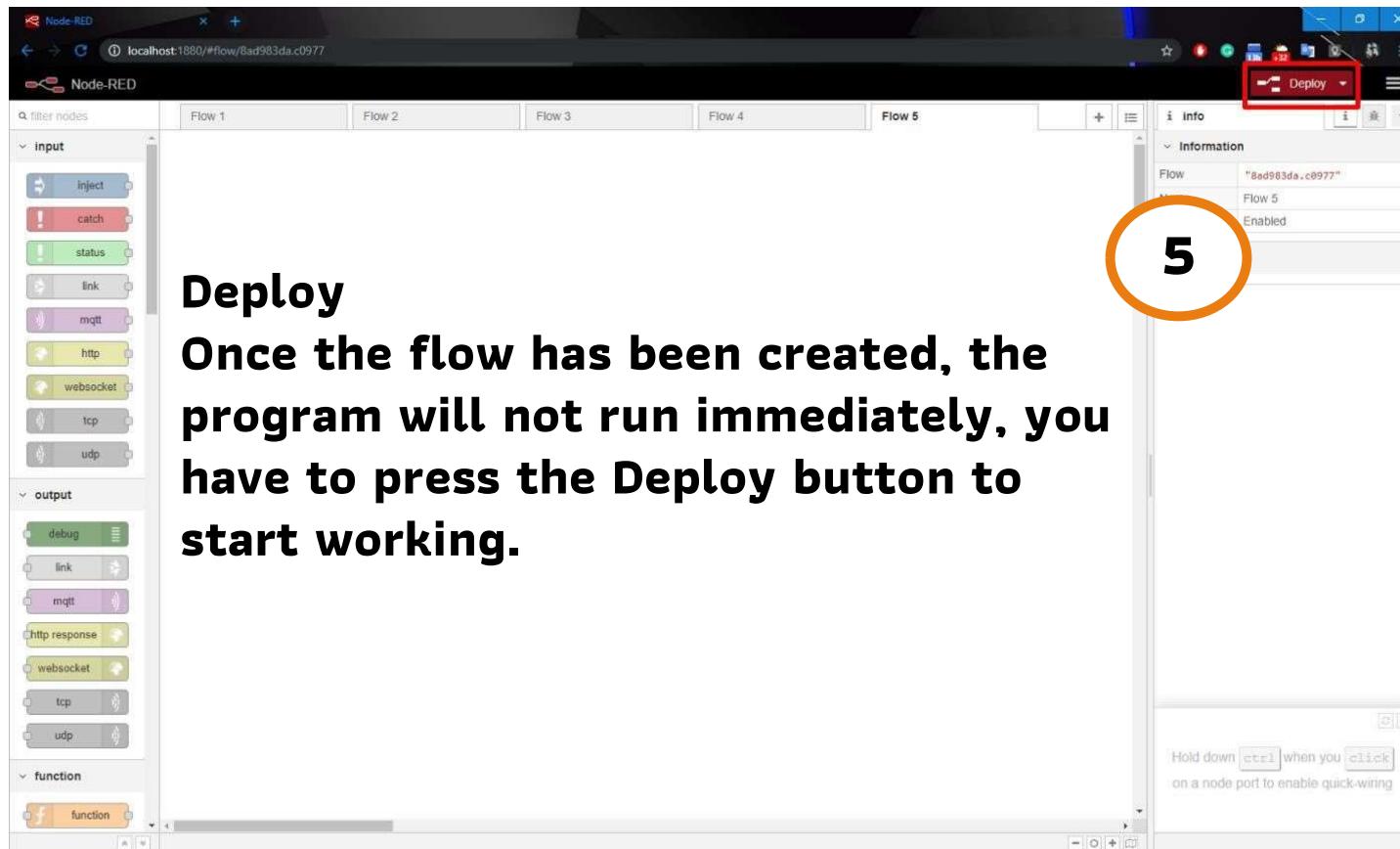
Component



Debug windows

Display information from the nodes for debugging purposes. Check the correctness of the program, etc.

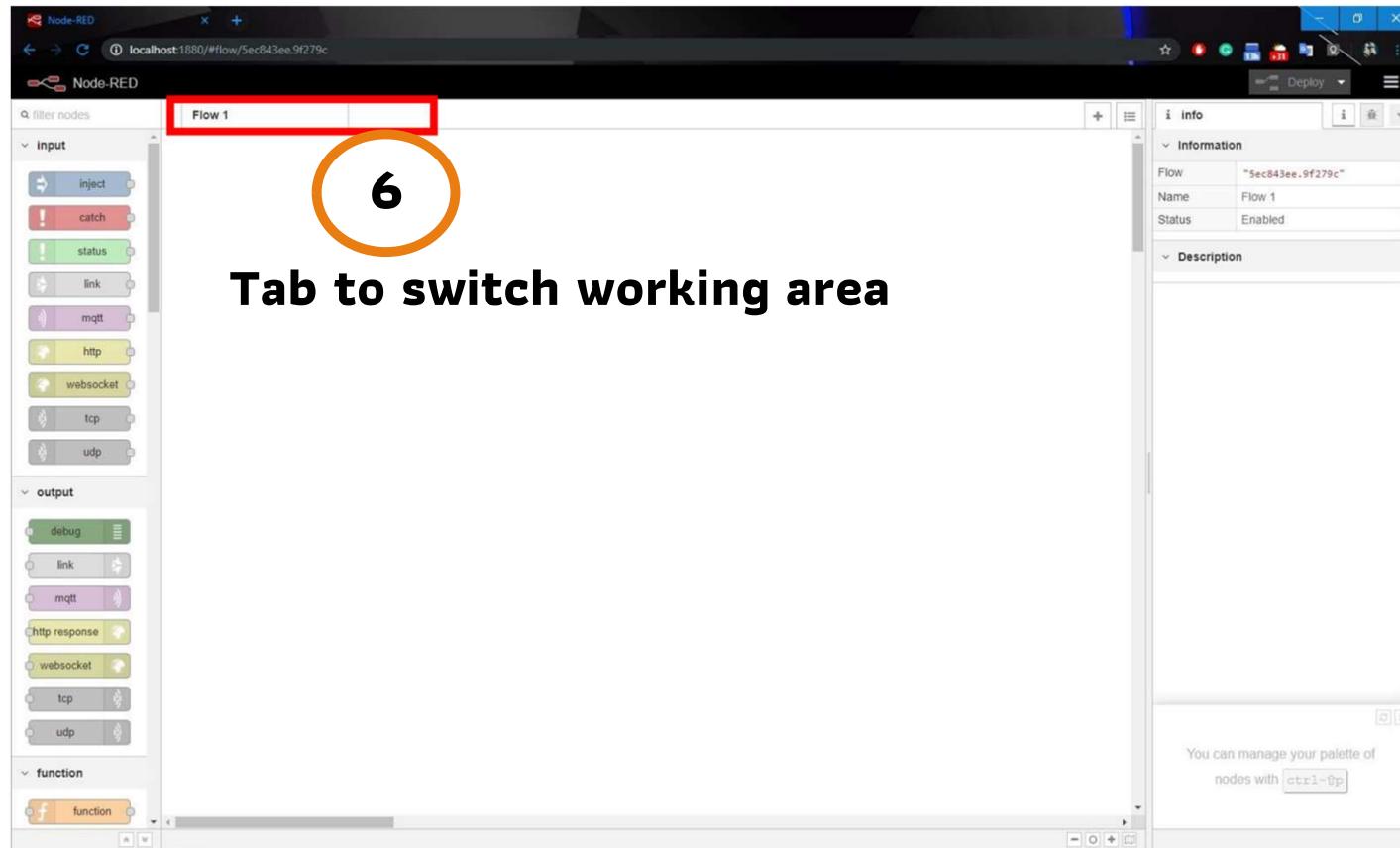
Component



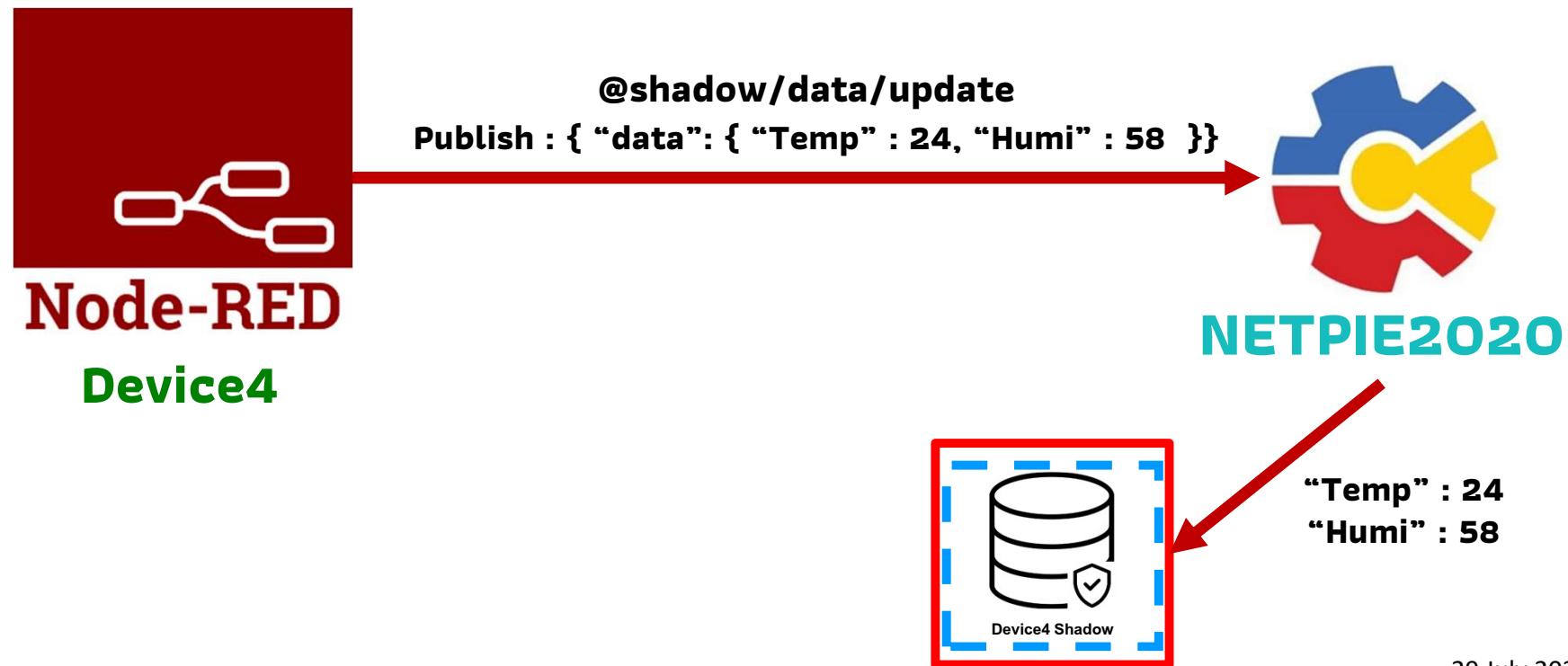
Deploy

Once the flow has been created, the program will not run immediately, you have to press the Deploy button to start working.

Component



Workshop 9 : Connect NETPIE2020 with Node-RED



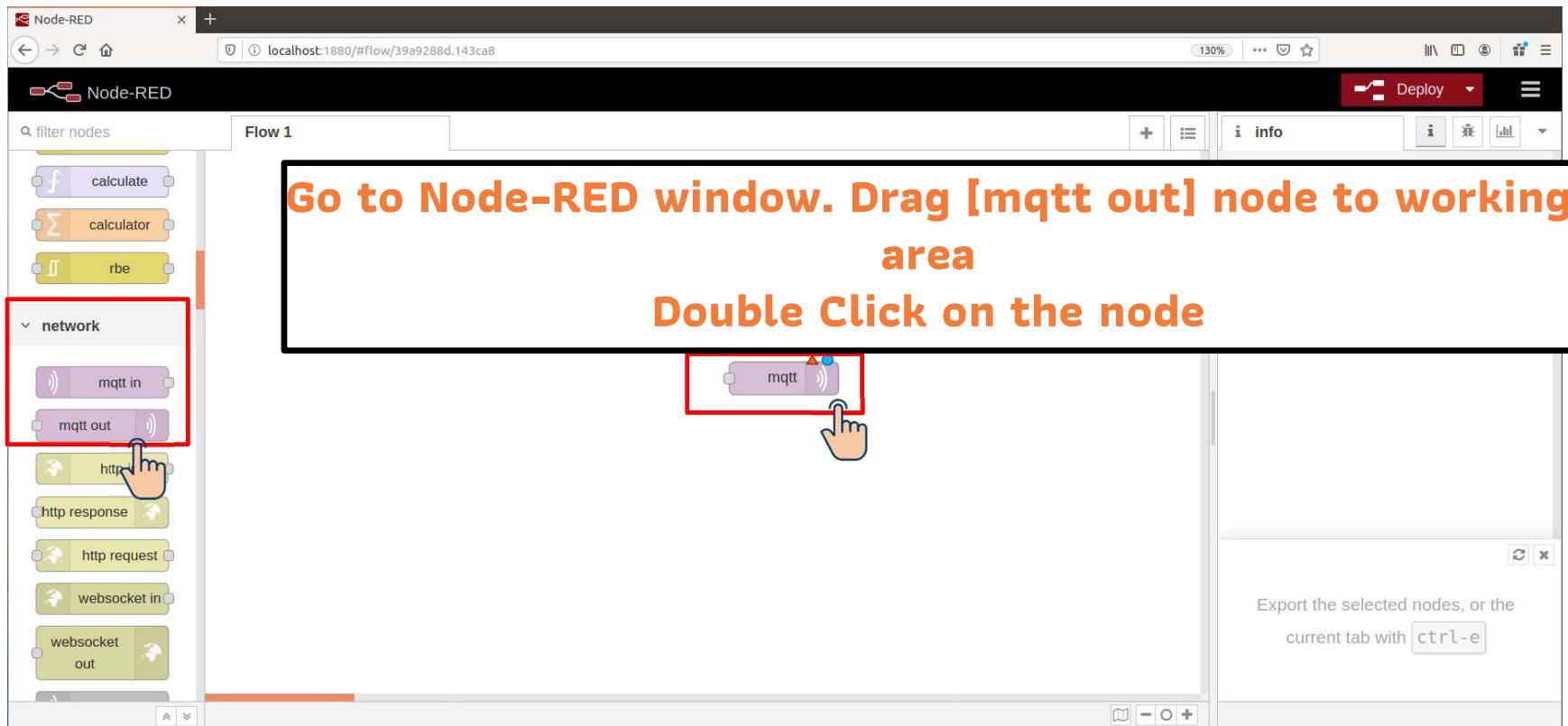
Workshop 9 : Connect NETPIE2020 with Node-RED

The screenshot shows a web-based interface for managing IoT devices. On the left, there's a sidebar with options: Overview, Device Lists (which is selected and highlighted in blue), Device Groups, Event Hooks, and Setting. The main area is titled "Smart_Factory_IoT_Challenge_2020 / device". It displays a list of four devices:

Name	Tags	Group	Create Date
Device4 Node-RED			2020-01-14 19:21
Device3 MQTTBox2, not join Group			
Device2 NodeMCU		Group1	2020-01-12 22:48
Device1 MQTTBox			

A red box highlights the first device, "Device4 (Node-RED)". A blue hand cursor is positioned over this row. A callout box with a black border and orange text overlaid on the list says "Create a new Device for Node-RED Click into that device".

Workshop 9 : Connect NETPIE2020 with Node-RED



Workshop 9 : Connect NETPIE2020 with Node-RED

Edit mqtt out node

Properties

Server: Add new mqtt-broker... (highlighted with a red box)

Topic: Topic

QoS: QoS

Name: Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

Enabled:



Edit mqtt out node > Add new mqtt-broker config node

Properties

Name: Name

Connection: Server: e.g. localhost, Port: 1883

Enable secure (SSL/TLS) connection:

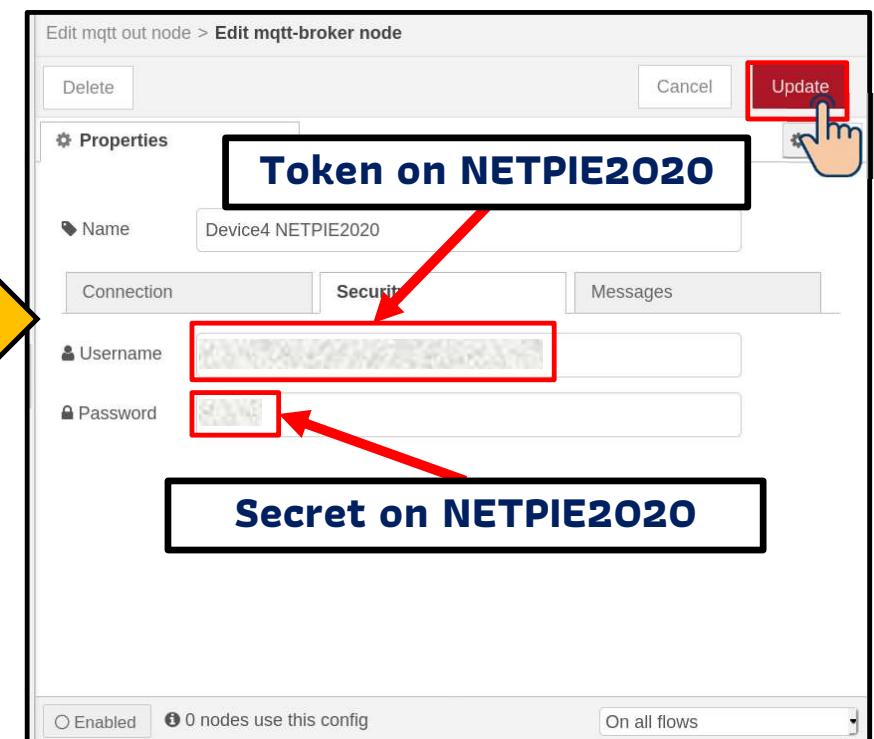
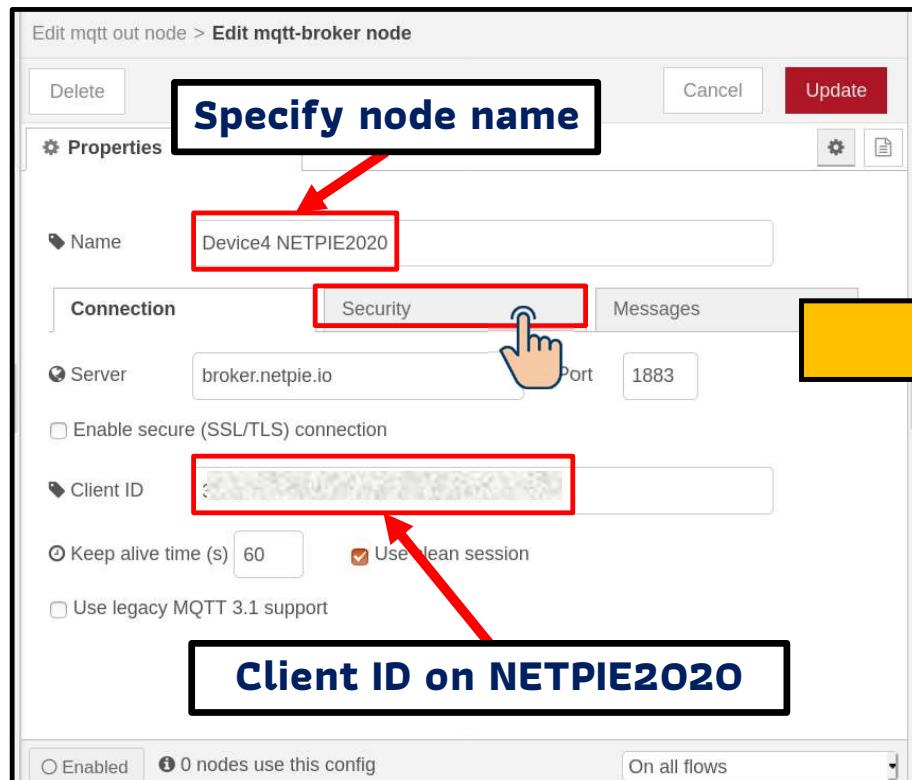
Client ID: Leave blank for auto generated

Keep alive time (s): 60, Use clean session:

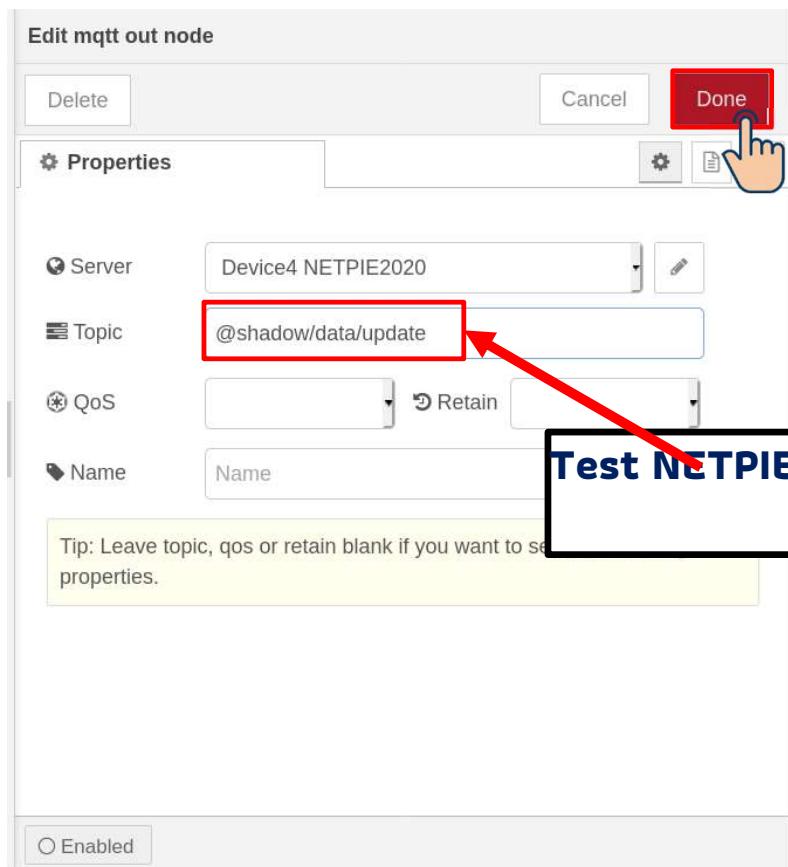
Use legacy MQTT 3.1 support:

Enabled: 0 nodes use this config, On all flows:

Workshop 9 : Connect NETPIE2020 with Node-RED

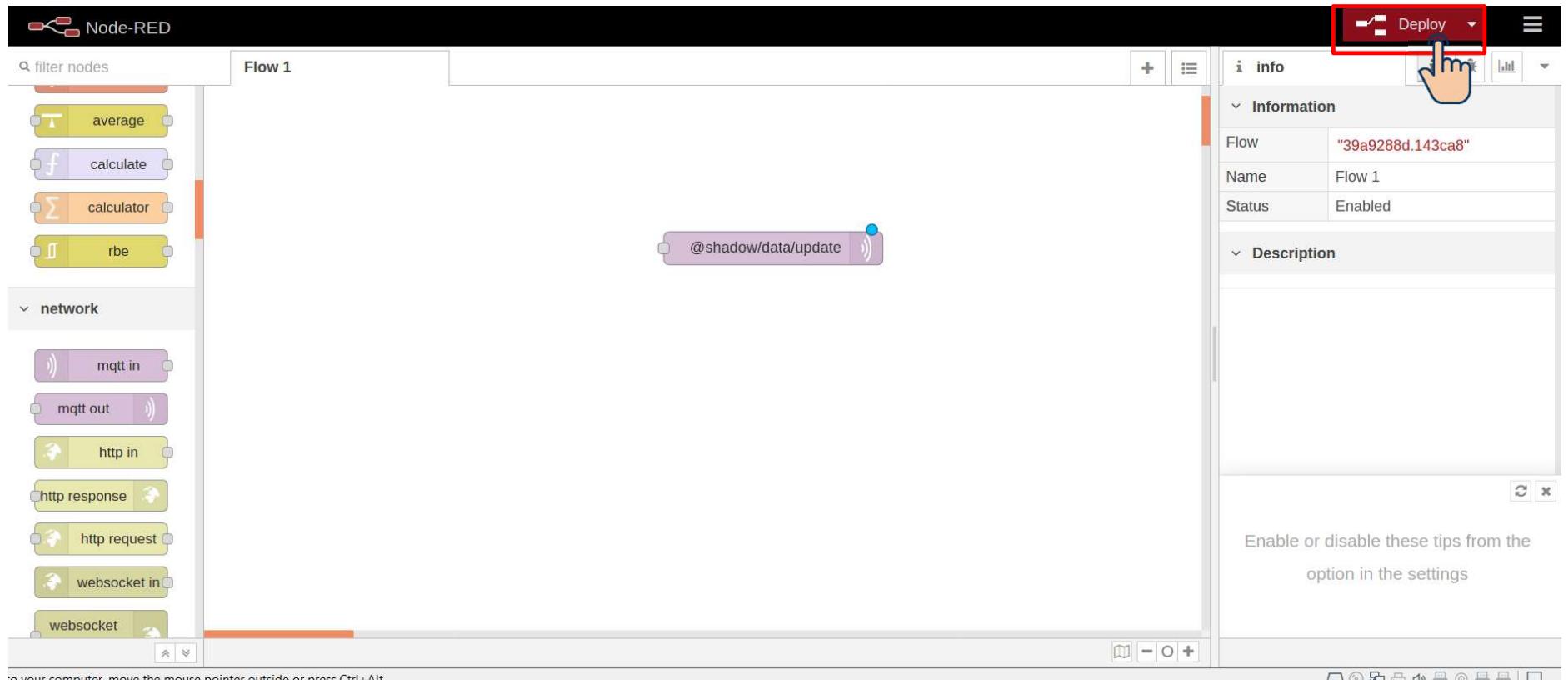


Workshop 9 : Connect NETPIE2020 with Node-RED



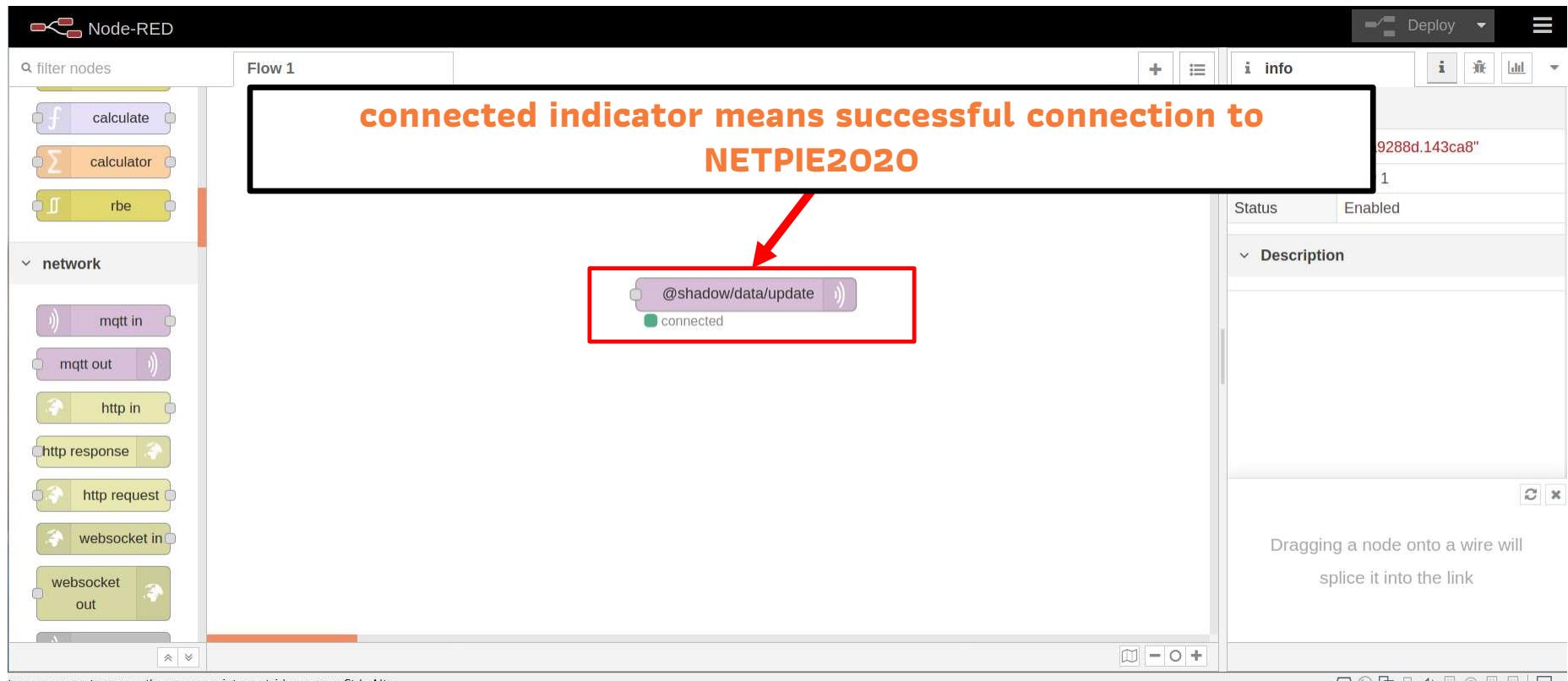
**Test NETPIE2020 on Node-RED by updating data to Device Shadow.
Specify topic : @shadow/data/update**

Workshop 9 : Connect NETPIE2020 with Node-RED



To your computer. move the mouse pointer outside or press Ctrl+Alt.

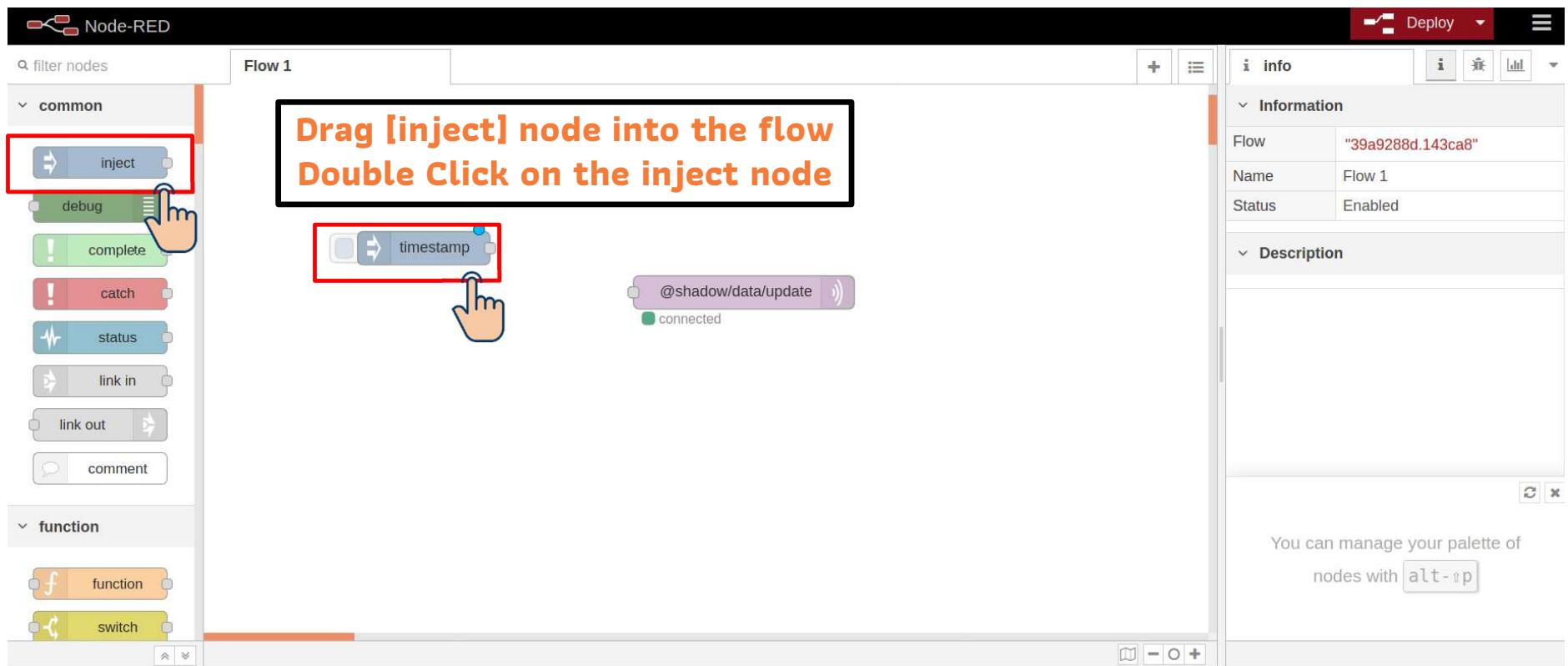
Workshop 9 : Connect NETPIE2020 with Node-RED



Workshop 9 : Connect NETPIE2020 with Node-RED

The screenshot shows the NETPIE portal interface. On the left, a sidebar lists navigation options: Overview, Device List, Device Groups, Freeboard, Event Hooks, and Setting. The main area displays a device configuration page for 'Device4' under 'Smart_Factory_IoT_Challenge_2020 / device'. The 'Description' field contains 'Node-RED'. In the 'Key' section, fields for Client ID, Token, and Secret are present, each with a redacted value. The 'Status' field shows 'Online' with a green circular icon, which is highlighted with a red rectangular box and a red arrow pointing towards it from below. A large black callout box with orange text overlaid on the status field contains the instruction: 'Check connection status on NETPIE2020 portal'. At the bottom right of the page are 'Save' and 'Cancel' buttons.

Workshop 9 : Connect NETPIE2020 with Node-RED



Workshop 9 : Connect NETPIE2020 with Node-RED

Edit inject node

Properties

Payload **timestamp** 

Topic 

Inject once after seconds, then

Repeat

Name

Note: "interval between times" and "at a specific time" will use cron.
 "interval" should be 596 hours or less.
 See info box for details.

Enabled

Click timestamp and choose JSON



Edit inject node

Properties

Payload **{}** 

Topic 

Inject once after seconds, then

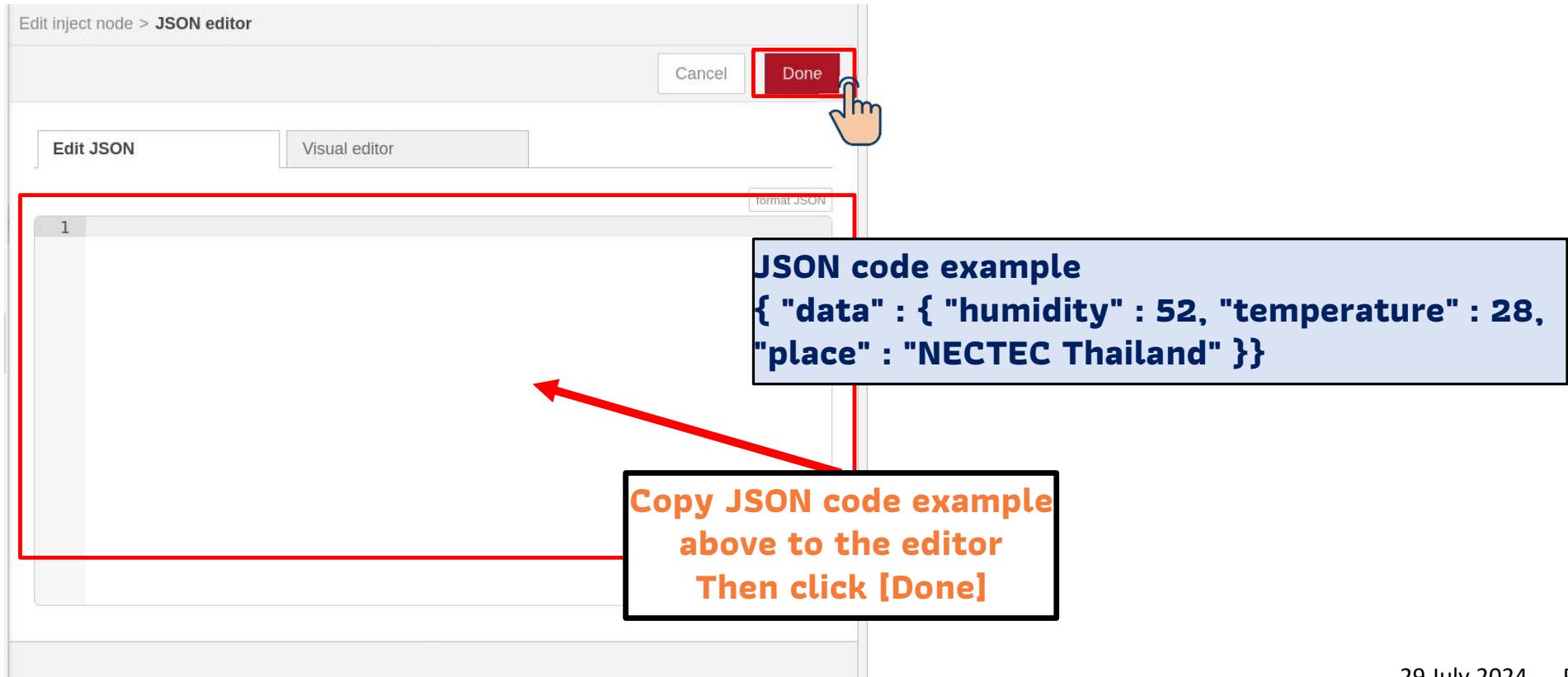
Repeat

Name

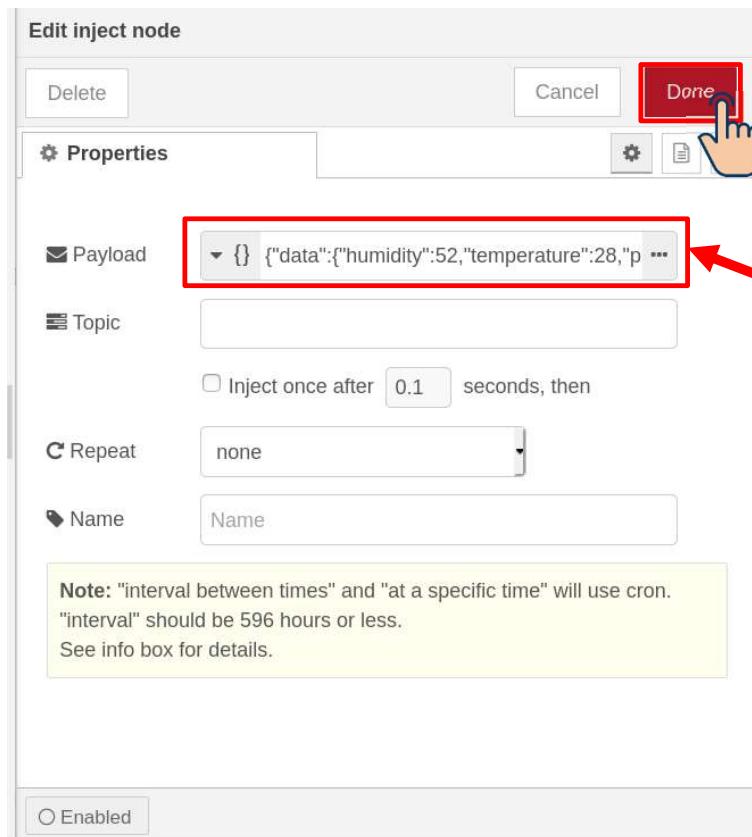
Note: "interval between times" and "at a specific time" will use cron.
 "interval" should be 596 hours or less.
 See info box for details.

Enabled

Workshop 9 : Connect NETPIE2020 with Node-RED

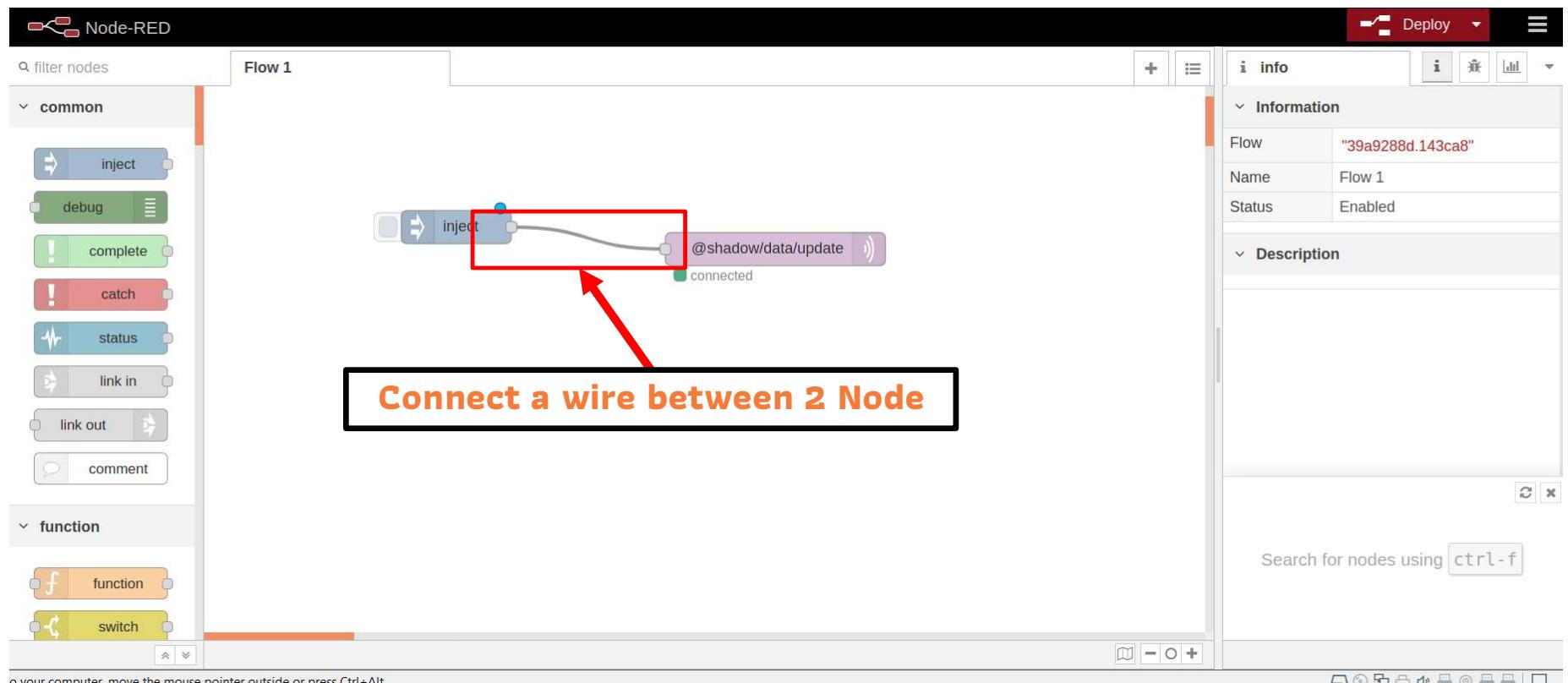


Workshop 9 : Connect NETPIE2020 with Node-RED

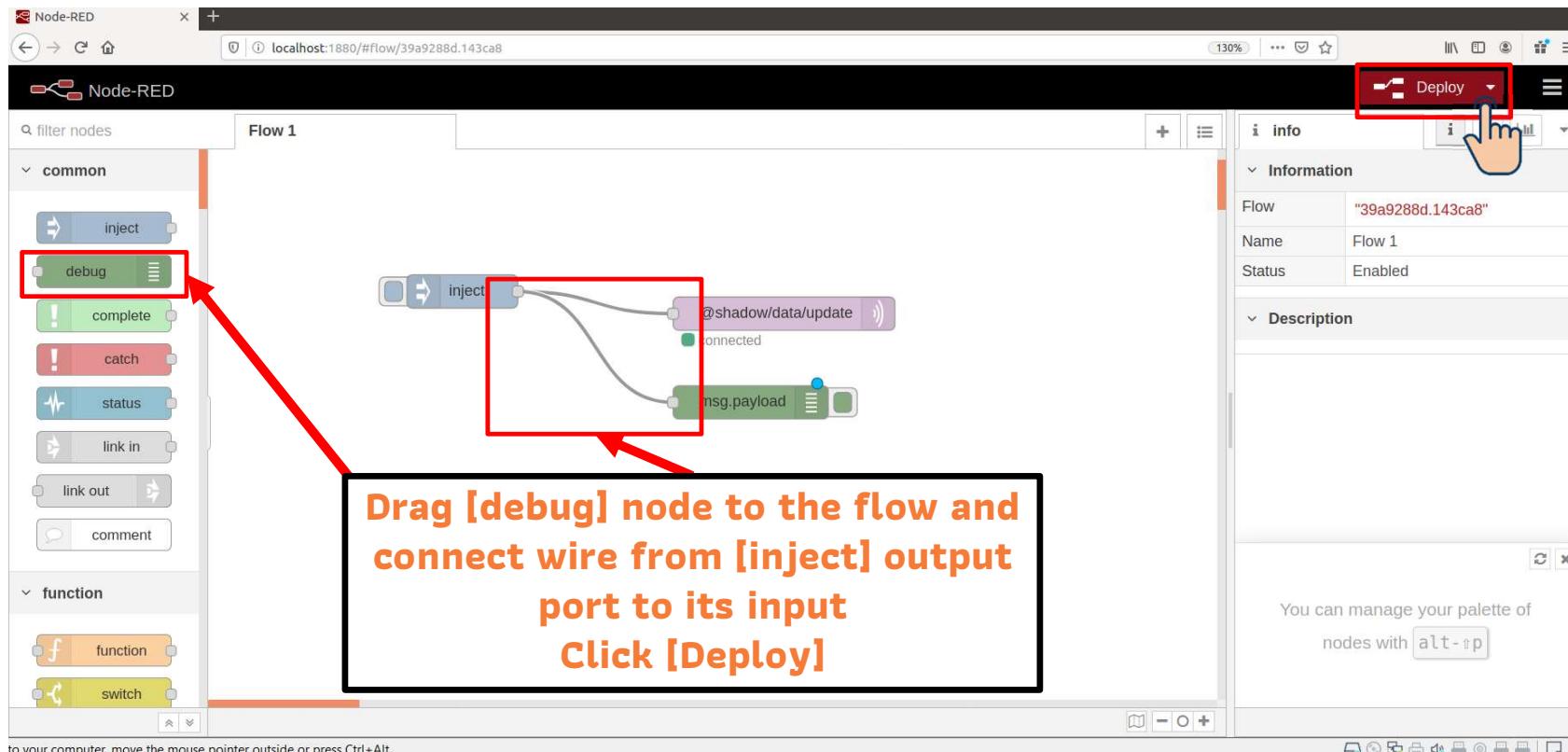


the message to be sent

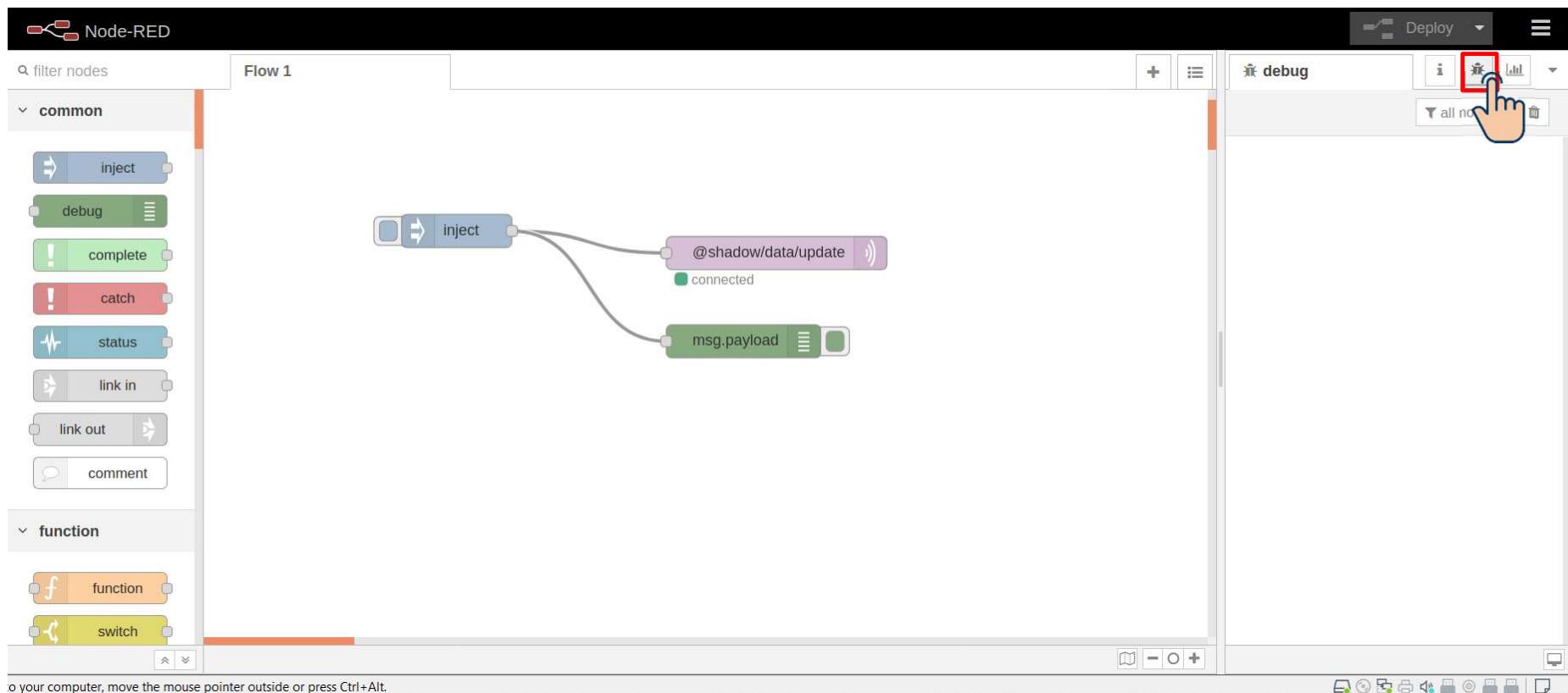
Workshop 9 : Connect NETPIE2020 with Node-RED



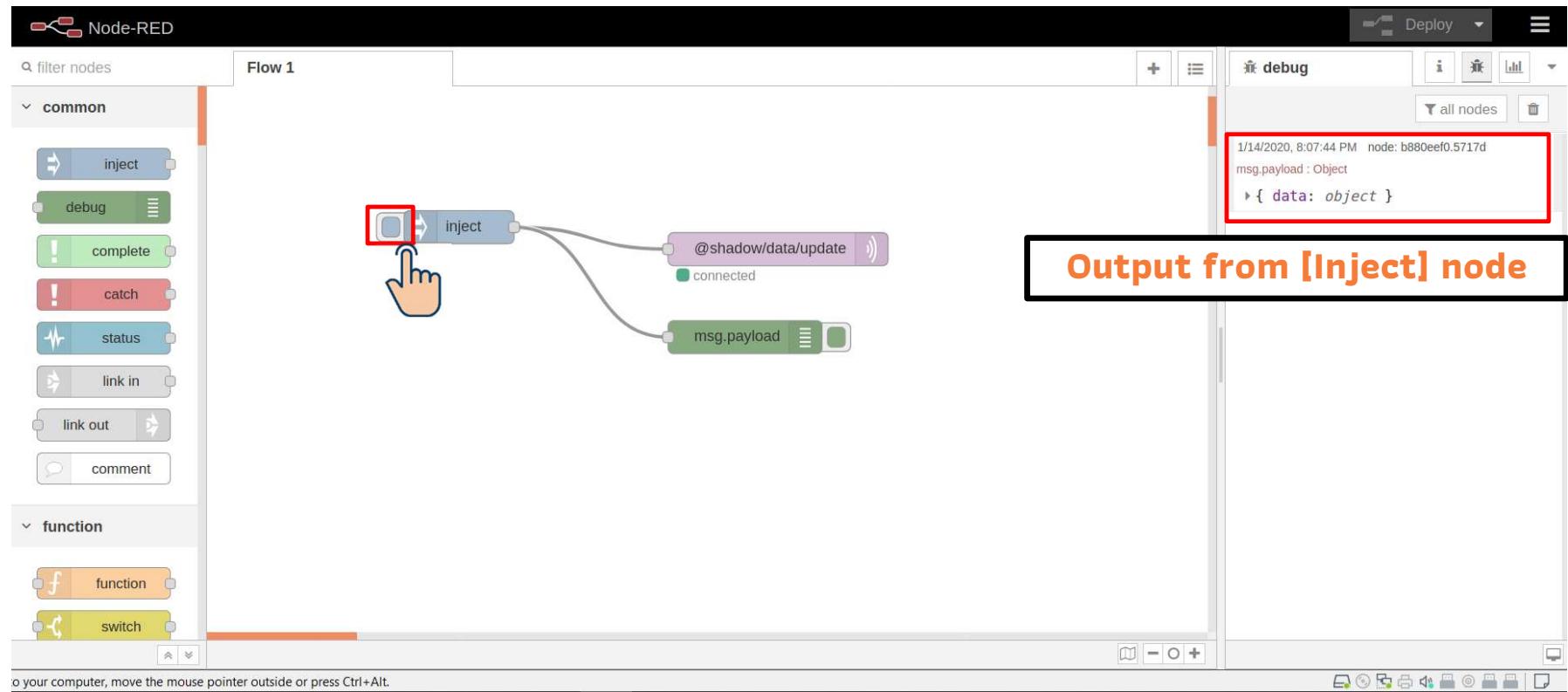
Workshop 9 : Connect NETPIE2020 with Node-RED



Workshop 9 : Connect NETPIE2020 with Node-RED



Workshop 9 : Connect NETPIE2020 with Node-RED



Workshop 9 : Connect NETPIE2020 with Node-RED

The screenshot shows the NETPIE Device Management interface. On the left, there's a sidebar with options: Overview, Device List, Device Groups, Freeboard, Event Hooks, and Setting. The main area shows a device configuration for 'Device4' under 'Smart_Factory_IoT_Challenge_2020 / device'. The 'Description' field contains 'Node-RED'. On the right, there's a 'Key' section with fields for Client ID, Token, and Secret, all of which are redacted. The 'Status' is shown as 'Online' with a green circle. Below this is a 'Shadow' tab, which is selected, showing a tree view of device data. A red box highlights the 'humidity : 52', 'temperature : 28', and 'place : NECTEC Thailand' entries under the 'object {3}' node. A callout box with a black border and orange text says 'Data on [Shadow] is updated'. At the bottom right are 'Save' and 'Cancel' buttons.

Smart_Factory_IoT_...

Smart_Factory_IoT_Challenge_2020 / device / Device4

Node-RED

Description

Key

Client ID :

Token :

Secret :

Status : Online

Shadow Schema Trigger

Select a node...

object {3}

humidity : 52

temperature : 28

place : NECTEC Thailand

Data on [Shadow] is updated

Save Cancel



Can follow
Tutorials and Example Projects from

<https://netpie.io/tutorials>

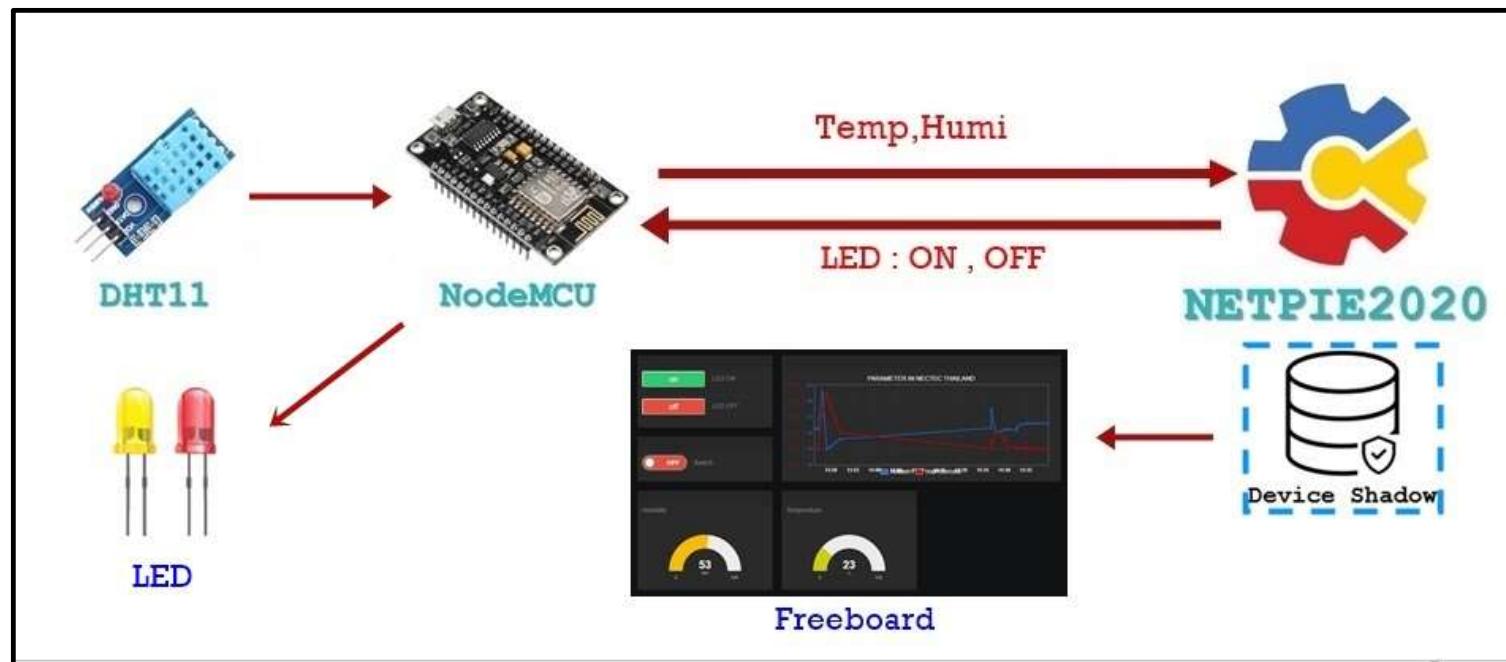
<https://netpie.io/exampleprojects>



Coming Soon...

Example of NETPIE 2020 connection with various hardware platforms

Hardware : NodeMCU[ESP8266]



Coming Soon...

Example of NETPIE 2020 connection with various hardware platforms

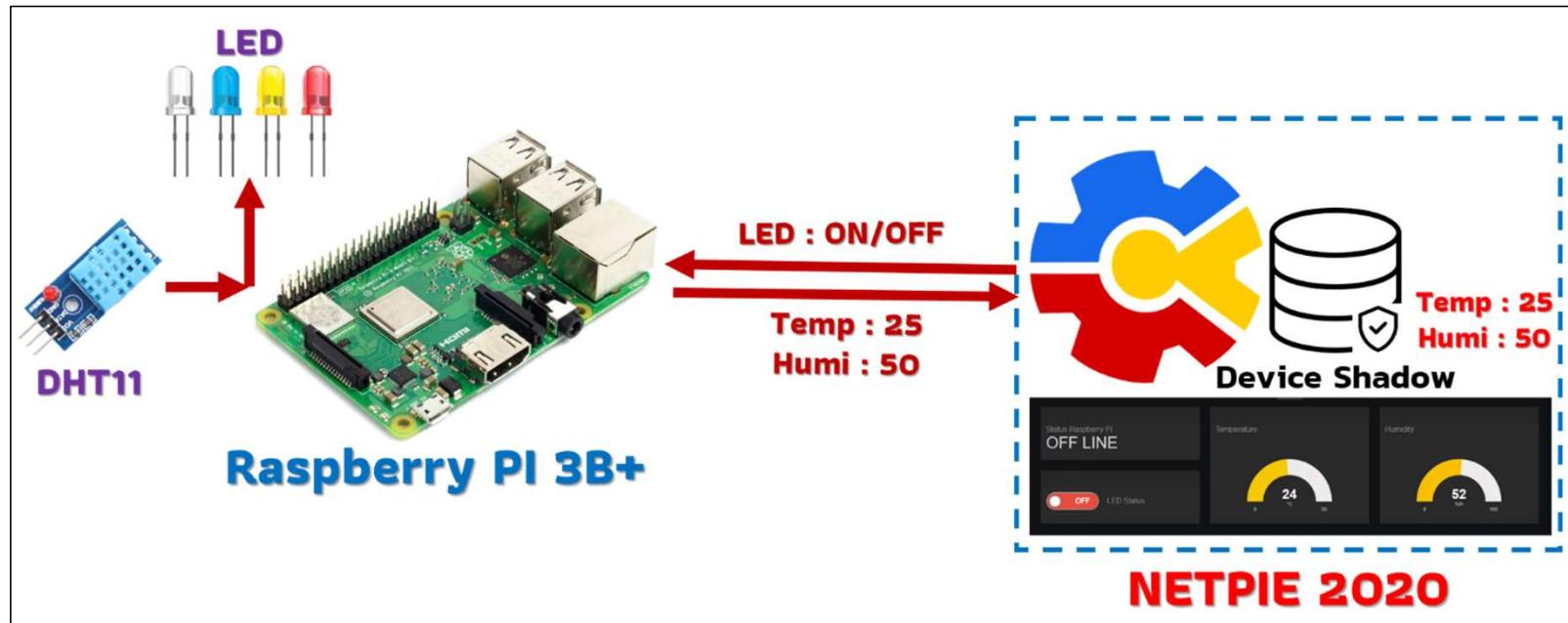
Hardware : M5Stack[ESP32]



Coming Soon...

Example of NETPIE 2020 connection with various hardware platforms

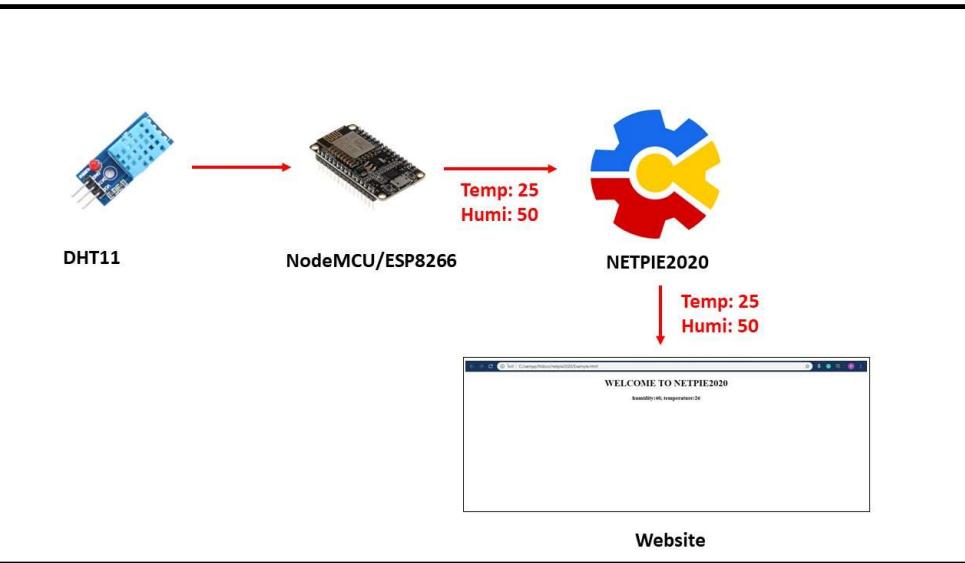
Hardware : Raspberry PI



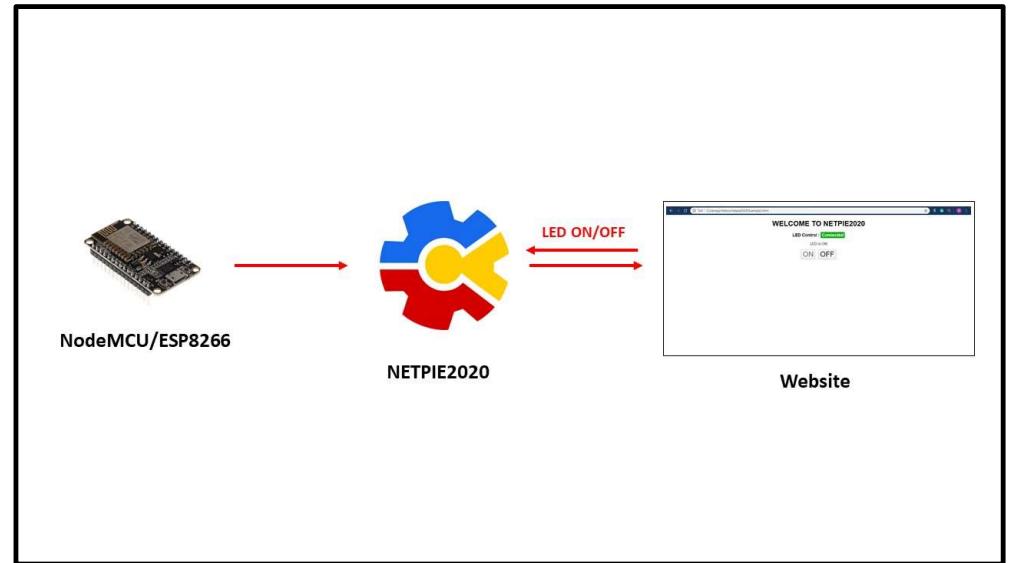
Coming Soon...

Example of NETPIE 2020 connection with various hardware platforms

Monitor sensor values on Web Browser



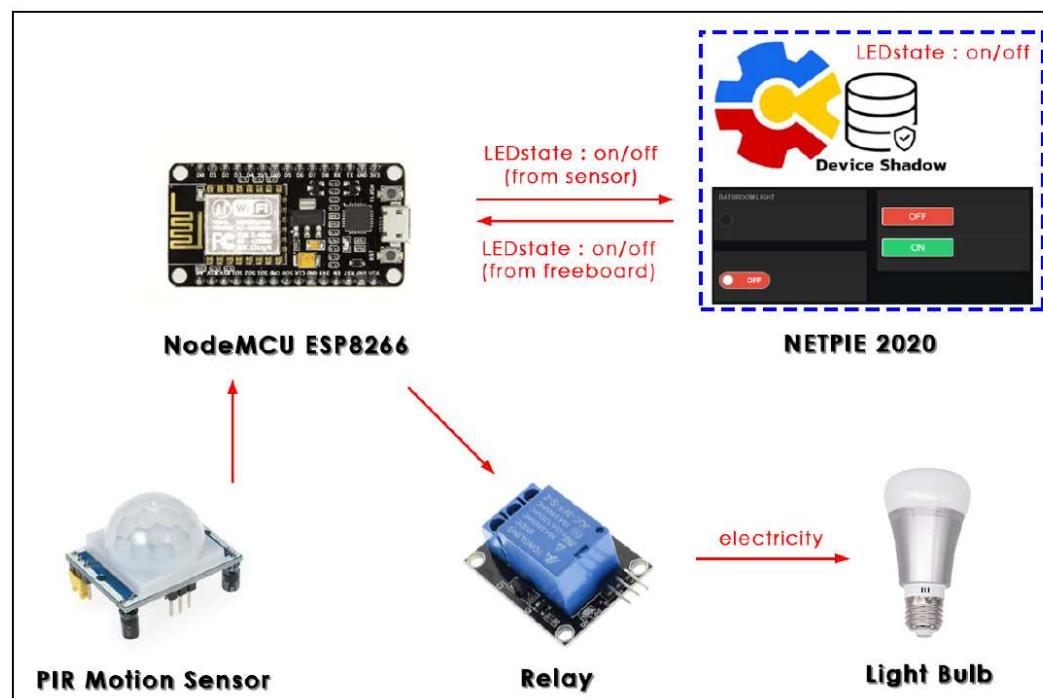
Control devices via Web Browser



Coming Soon...

Example of NETPIE 2020 connection with various hardware platforms

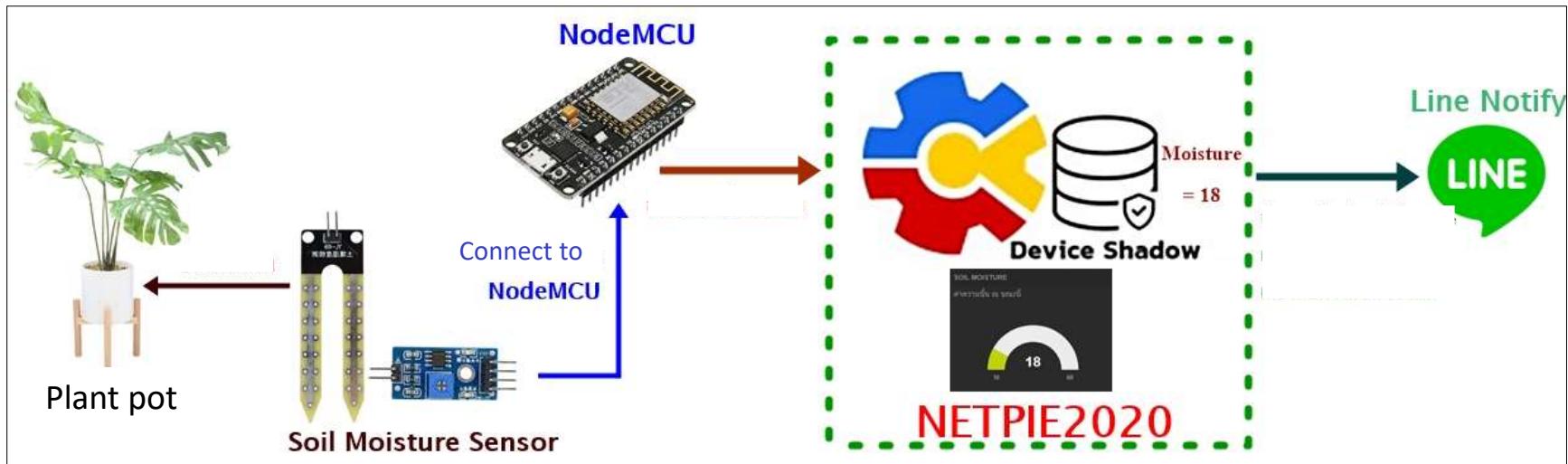
Smart Home



Coming Soon...

Project examples that use NETPIE2020

Smart Farm



Coming Soon...



NETPIE 2020

From **Makers Nation**
Toward **Smart Nation**

New Version

Grand opening: 20.02.2020



เอกสารฉบับนี้เป็นลิขสิทธิ์ของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ

เอกสารฉบับนี้เผยแพร่ภายใต้สัญญาอนุญาตแบบ

Creative Commons Attribution 4.0 International License



อนุญาตให้นำผลงานไปใช้ได้ แก้ไขได้ ดัดแปลงเนื้อหาได้ ใช้ในเชิงพาณิชย์ได้ โดยอ้างอิงที่มา

ดูรายละเอียดสัญญาอนุญาตที่ <https://creativecommons.org/licenses/by/4.0/>



Translation to English by Dr.Varodom Toochinda
Dept. of Mechanical Engineering,
Kasetsart University

Thank You



29 July 2024



Page 236