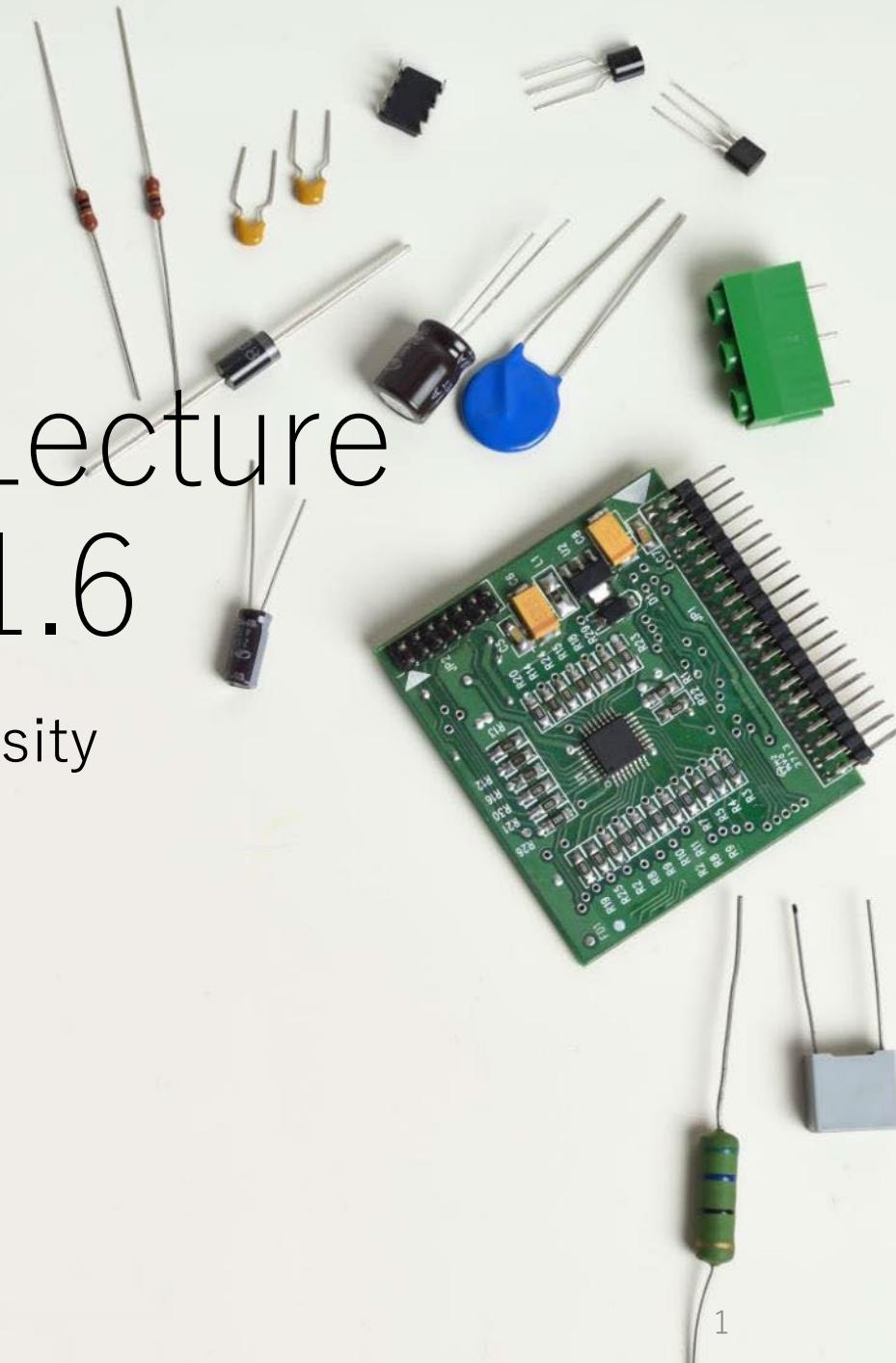


# 1451 Hands-on Lecture

## IEEE P1451.1.6

Hiroaki Nishi, Keio University



# Directions

- The program will be facilitated in a hands-on style.
- In two lectures, we will design your own NCAP application and connect with sensors and actuators over the Internet.
- We will use Low-Code style design (Node-RED).
  - It is needless to be familiar with programming language.
  - It is needless to know about the Internet protocol.
- However, we will develop:
  - Installing OS and set-up real server
  - Set-up all design environments and applications by yourself
  - Design your own 1451.1.6 network application
- We have a deadline
  - Please do not hesitate asking if you have a trouble.
  - Due to the progress of the program, delays may result in joining a breakout session with different content.

# Preparation in advance

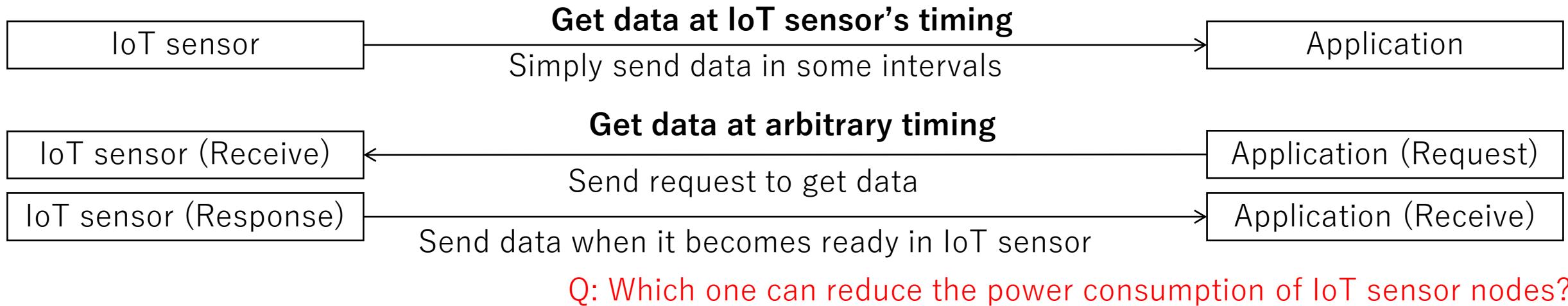
- Prepare one Raspberry-Pi per person. You will also need a USB keyboard, mouse, HDMI monitor, and HDMI cable. If you are using your own PC, use that.
  - <https://www.raspberrypi.com/documentation/computers/getting-started.html>
  - It does not matter if you are at the university or at home, you should participate with a Raspberry-Pi with internet access available.
- If you do not use a Raspberry-Pi, you may use free Node-RED cloud service to design application. It is free, but it has 30-day limitation for the use.
- Either wired or Wi-Fi Internet connection is acceptable.
  - Please follow the installation guides in this text. It is recommended to connect to the Internet. If you cannot connect to the Internet, it is required to set-up your own Raspberry-Pi for NCAP server. All applications in a domain can share one NCAP.
- Minimum requirements
  - For designing NCAP applications
    - Window/Mac/Raspberry-Pi machine and install Node-RED or access to Node-RED cloud service
  - For designing NCAP
    - Additionally, Raspberry-Pi, DHT111, Micro Servo (SG-90), wire, LED, breadboard, etc.

# What will we do?

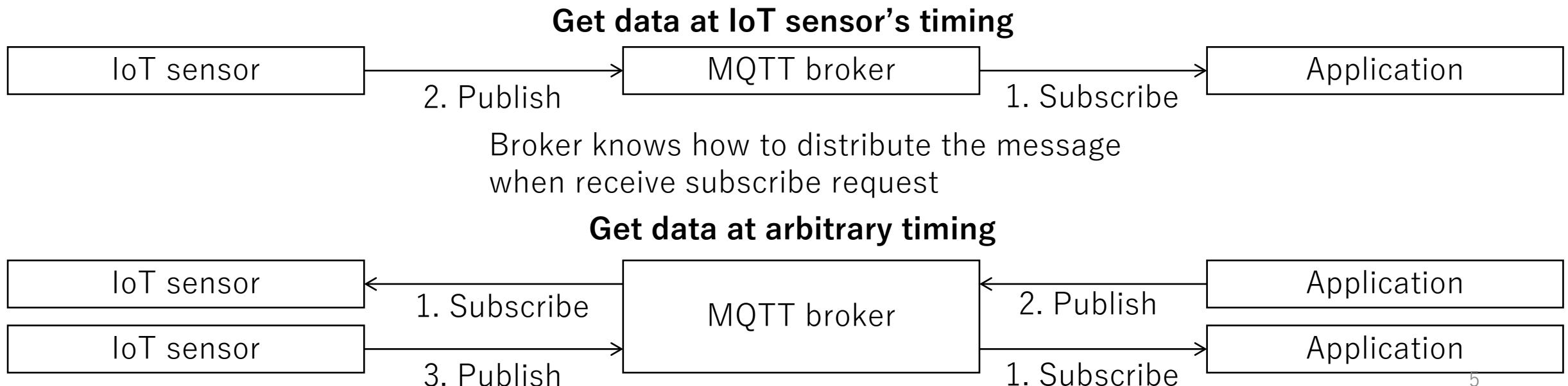
- Confirm knowledge about 1451.0
- Learn about 1451.1.6
- Learn about MQTT
- Installation
  - Install Node-RED to Raspberry-Pi
- Learn about Node-RED
  - Install required libraries to Node-RED
- Access to sensor (using direct method) and design visualization application
- Access to sensor
- Access to actuator

# What is MQTT? (General communication model)

- General communication model (request-response communication model)

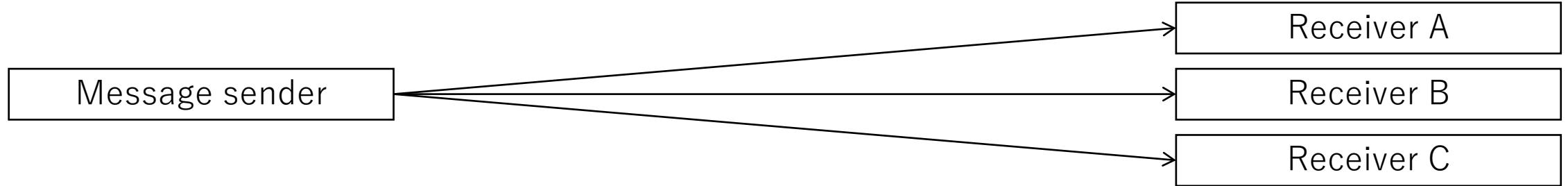


- Publish-subscribe communication model



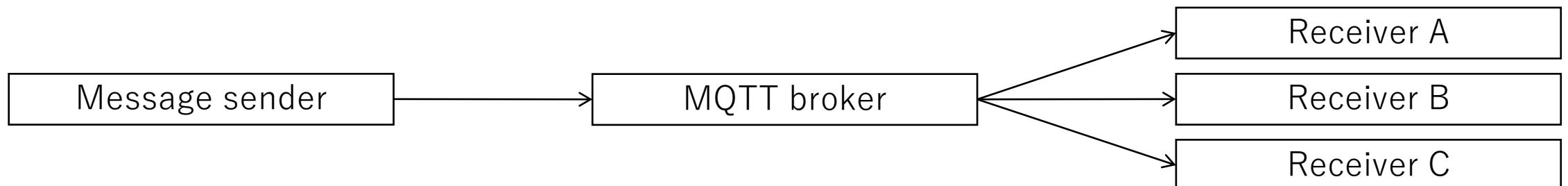
# What is MQTT? (Multicast / Broadcast)

- General communication model



Single casts must be repeated to convey information to multiple receiving nodes

- Publish-subscribe communication model



Only one message is enough to send all subscribers

Q: Which one can reduce the power consumption of IoT sensor nodes?

# What is the merit of MQTT? (general perspectives)

- Most IoT systems support MQTT because…
  - Simple protocol design: MQTT header size is as small as 2 bytes minimum.
  - MQTT uses a broker server to distribute the message from an MQTT client.
    - When multicasting or broadcasting messages, it is required to manage a protocol to know all IP addresses for the network nodes and generally to send the message one by one for all the network nodes.
  - Low-power design
    - In general communication models, the power cost for processing and network communication to distribute messages increases linearly according to the number of network nodes.
    - MQTT broker only receives one publish message, and the broker distributes the message to all subscribers.
    - In this delivery, the topic name plays an important role. MQTT message has the topic name, and the topic name is the keyword that indicates a group of nodes receiving the message. Therefore, the message distribution is completely controlled only by using the topic name.
    - The naming rules of this topic name are the essential contributions as IEEE P21451.1.6.
- IEEE P1451.1.6 standard defines the topic name rules for emulating IEEE 1451.0 message transactions to achieve IEEE 1451 functions and services.

# What is the merit of MQTT? (1451 perspectives)

- 1451.0 requires broadcast message to announce and discover NCAPs.
  - However, the broadcast messages are not available on the Internet because it causes DDOS attacks. All broadcast address was closed.
  - Therefore, some 1451.0 messages are not available on the Internet.
  - It is required to predefine the NCAP address as firmware or to find NCAPs by using another protocol, such as DNS.
- MQTT provides broadcast/multicast access methods in the Internet.
  - MQTT provides full functions of 1451 over the Internet.
- In this lecture we do not design announce and discover messages.
  - Refer 1451.0 and 1451.1.6 document to implement these messages.

# MQTT v5 is now available!

- MQTTv5 is the advanced new standard of MQTT
- Message properties
  - The most important improvement of MQTT v5 for IEEE P1451.1.6.
- Properties provide a new field for the message body
  - The field is independently existing with the original message body.
  - The properties give flexibilities for MQTT message chains and total protocol design.
  - It natively supports request-response style MQTT message by using special properties.
- IEEE P1451.1.6 is targeting both uses of MQTTv3 and MQTTv5.
  - MQTTv3 is the traditional MQTT and is used as the IoT de facto standard messaging protocol.
- It also advanced in security, authorization, and usability.
- MQTT broker mosquitto (> version 1.6) supports MQTT v5, and…

# Status of Open Source MQTTv5 support

- Broker server
  - EMQ X (MQTT-SN, Erlang)
  - flespi MQTT broker (Erlang)
  - HiveMQ Community Edition (Java)
  - JoramMQ (Java, Free eva. ready)
  - KMQTT (Kotlin)
  - mosquitto (Java)
  - **Mosquitto (Thread safe, C)**
  - mqtttools (Python)
  - VerneMQ (Erlang, OTP)
- Client
  - HiveMQ MQTT Client (Java)
  - M2Mqtt (C#)
  - Machine Head (? , Clojure)
  - Mosquitto (C, >1.6)
  - MQTT-C (C)
  - mqtttools (Python)
  - net-mqtt (Haskell)
  - **paho MQTT (C++, Java(scr.), Python, Go)**
  - gMQTT (Python)
  - wolfMQTT (C)
  - OpenHAB MQTT binding (Java)

There are many commercial MQTT v5 services, including AWS and other cloud services.

# P1451.1.6 for 1451.0

- What is 1451?
  - Refer Kang and William's lecture on 1451.
- What are defined in P1451.1.6?
  - Topic naming rules to exchange IEEE1451 messages
  - Message data formats to use with MQTT
  - Design of states and processes for maintaining concerning protocols
  - IEEE1451 network service messages over MQTT
  - Simple and general style of MQTT message for the use of 1451 devices
  - Time synchronization
- Status of IEEE P1451.1.6
  - Documentation has already finished, but not revised according to the new 1451.0D4 document.

# Implementation using Open Sources

- Implementations for Interoperability and simple operation check
- Need Raspberry-Pi and sensors (DHT11)
- All message formats of D-OP, C-OP, and D0-OP are supported.
- Only reading sample data, writing data to control servo motor and reading TEDS are supported according to the request.
- Data formats are extended to use with MQTT and from the perspective of natural definition.
- The system environment of this lecture uses NCAP with TIM and attached sensors on Raspberry-Pi. NCAP application is implemented as Node-RED.
- GitHub (World biggest software development platform)
  - Including sandboxes of testing codes and data
  - <https://github.com/westlab/IEEEP1451.1.6-2023>
- IEEE-SA OPEN (Will be updated)
  - <https://opensource.ieee.org/west/ieee-p1451.1.6>

# Message data formats in .1.6

- D0-OP (Dot Zero Operation)
  - IEEE P1451.0-based (Dot Zero-based) Operation (D0-OP)
  - D0-Message uses new 1451.0 standards message formats without any modification.
    - Strictly observe new 1451.0 standards.
  - The suffix of the topic name shall start with '\_1451.1.6/D0/' as SPFX and following TOM.
  - LNS follows the string to be a completed topic name. (LNS: Location Name String)  
ex) '\_1451.1.6/D0/BLD1/FL2/ROOM3'
- C-OP (CSV Operation)
  - Comma Separated Values (CVS) format can be used as an option. TOM is changed to 'C'.  
ex) '\_1451.1.6/C/BLD1/FL2/ROOM3'
  - Perfectly compatible with 1451.0 messages, but is change to text-based format.
  - Extend the field size limitation in D0-OP considering usability and solve debuggability
- D-OP (Direct Operation)
  - Direct format for enabling a simpler and more suitable format to use with MQTT.
  - Significant messages, such as simple read/write sensor or TEDS operations are only defined.
  - Strictly observes OASIS MQTT guidelines and general MQTT use cases.



IEEE 1451 similarity



MQTT OASIS similarity

# Message topics

- Request-Response message is used
  - Request:
    - Data Request Topic: \_1451.1.6/[D0 or C]/LNS of server/DATA/[TIM UUID or TIM NAME]
    - Data Request Message: The unique topic name to publish the sensor data, shortly [LNS of client]
    - NCAP server subscribes to this topic. SPFX, TOM, and concerning descriptors are not included.
  - Response:
    - Data Response Topic: [LNS of client]
    - Data Response Topic: \_1451.1.6/[D0 or C]/[LNS of client]/RES/[TIM UUID or TIM NAME]
    - Data Response Message: Sensor Data
    - NCAP client subscribes to this topic.
- Publish-Subscribe communication
  - General communication style for MQTT
  - Dedicated topic: \_1451.1.6/D/LNS/[TIM UUID or TIM NAME]
  - NCAP server publishes message to the topic.

# Network Services of 1451.0

## Discovery services

- NCAPAnnouncement
- NCAPDeparture
- NCAPAbandonment
- NCAPTIMAnnouncement
- NCAPTIMDeparture
- NCAPTIMTransducerAnnouncement
- NCAPTIMTransducerDeparture
- NCAPDiscovery
- NCAPTIMDiscovery
- NCAPTIMTransducerDiscovery

## TEDS Services

- Query TEDS
- Read TEDS
- Write TEDS
- Update TEDS

## Event notification services

- SubscribeTransducerEventFromOneChannelOfOneTIM
- NotifyTransducerEventFromOneChannelOfOneTIM
- UnsubscribeTransducerEventFromOneChannelOfOneTIM
- SubscribeTransducerEventFromMultipleChannelsOfOneTIM
- NotifyTransducerEventFromMultipleChannelsOfOneTIM
- UnsubscribeTransducerEventFromMultipleChannelsOfOneTIM
- SubscribeTransducerEventFromMultipleChannelsOfMultipleTIMs
- NotifyTransducerEventFromMultipleChannelsOfMultipleTIMs
- UnsubscribeTransducerEventFromMultipleChannelsOfMultipleTIMs
- SubscribeNCAPHeartbeat
- NotifyNCAPHeartbeat
- UnsubscribeNCAPHeartbeat

## Transducer Access Service

**SyncReadTransducerSampleDataFromAChannelOfATIM**  
SyncReadTransducerBlockDataFromAChannelOfATIM  
SyncReadTransducerSampleDataFromMultipleChannelsOfATIM  
SyncReadTransducerBlockDataFromMultipleChannelsOfATIM  
SyncReadTransducerSampleDataFromMultipleChannelsOfMultipleTIMs  
SyncReadTransducerBlockDataFromMultipleChannelsOfMultipleTIMs

**SyncWriteTransducerSampleDataFromAChannelOfATIM**  
SyncWriteTransducerBlockDataFromAChannelOfATIM  
SyncWriteTransducerSampleDataFromMultipleChannelsOfATIM  
SyncWriteTransducerBlockDataFromMultipleChannelsOfATIM  
SyncWriteTransducerSampleDataFromMultipleChannelsOfMultipleTIMs  
SyncWriteTransducerBlockDataFromMultipleChannelsOfMultipleTIMs

AsyncReadTransducerBlockDataFromAChannelOfATIM  
CallbackAsyncReadTransducerBlockDataFromAChannelOfATIM  
AsyncReadTransducerStreamDataFromAChannelOfATIM  
CallbackAsyncReadTransducerStreamDataFromAChannelOfATIM  
AsyncReadTransducerBlockDataFromMultipleChannelsOfATIM  
Callback AsyncReadTransducerBlockDataFromMultipleChannelsOfATIM  
AsyncReadTransducerBlockDataFromMultipleChannelOfMultipleTIMs  
CallbackAsyncReadTransducerBlockDataFromMultipleChannelOfMultipleTIMs

# Message example

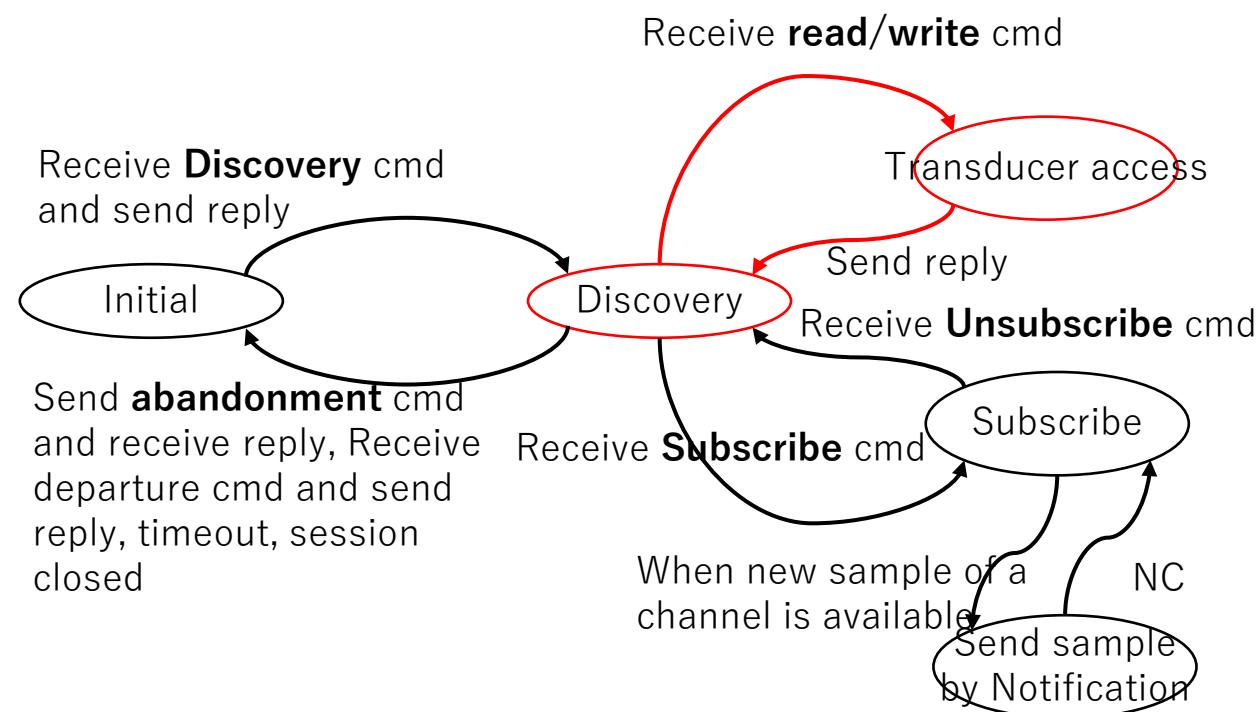
- NCAP announcement message is explained as an example
- D0-OP (binary)
  - IDL:

```
Args:: UInt16 netSvc0101 {
    in Args::NetSvcType  netSvcType = 1,
    in Args::NetSvclId   netSvclId = 1,
    in Args::MsgType     msgType = 3,
    in Args::UInt16      msgLength (2 bytes),
    in Args::UUID        ncapId,
    in Args::_String    ncapName (length = 16),
    in Args::AddressType addressType (1:IPv4, 2:IPv6),
    in Args::UInt8Array  ncapAddress (length 4/16 bytes)
};
```
- C-OP (Type dependent but size independent)
  - 1,1,3,ncapId,ncapName,addressType,ncapAddress

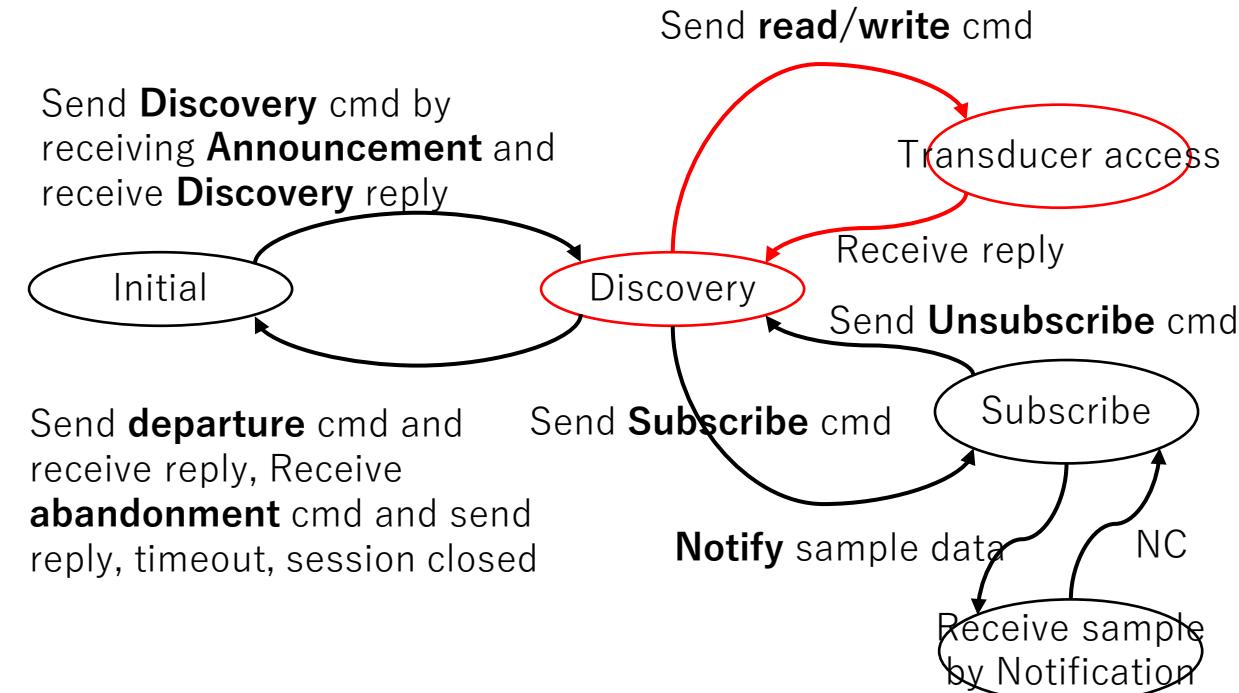
# Network Service State Transition

- The true NCAP and NCAP application must follow the following state machines.

NCAP state machine



NCAP application state machine

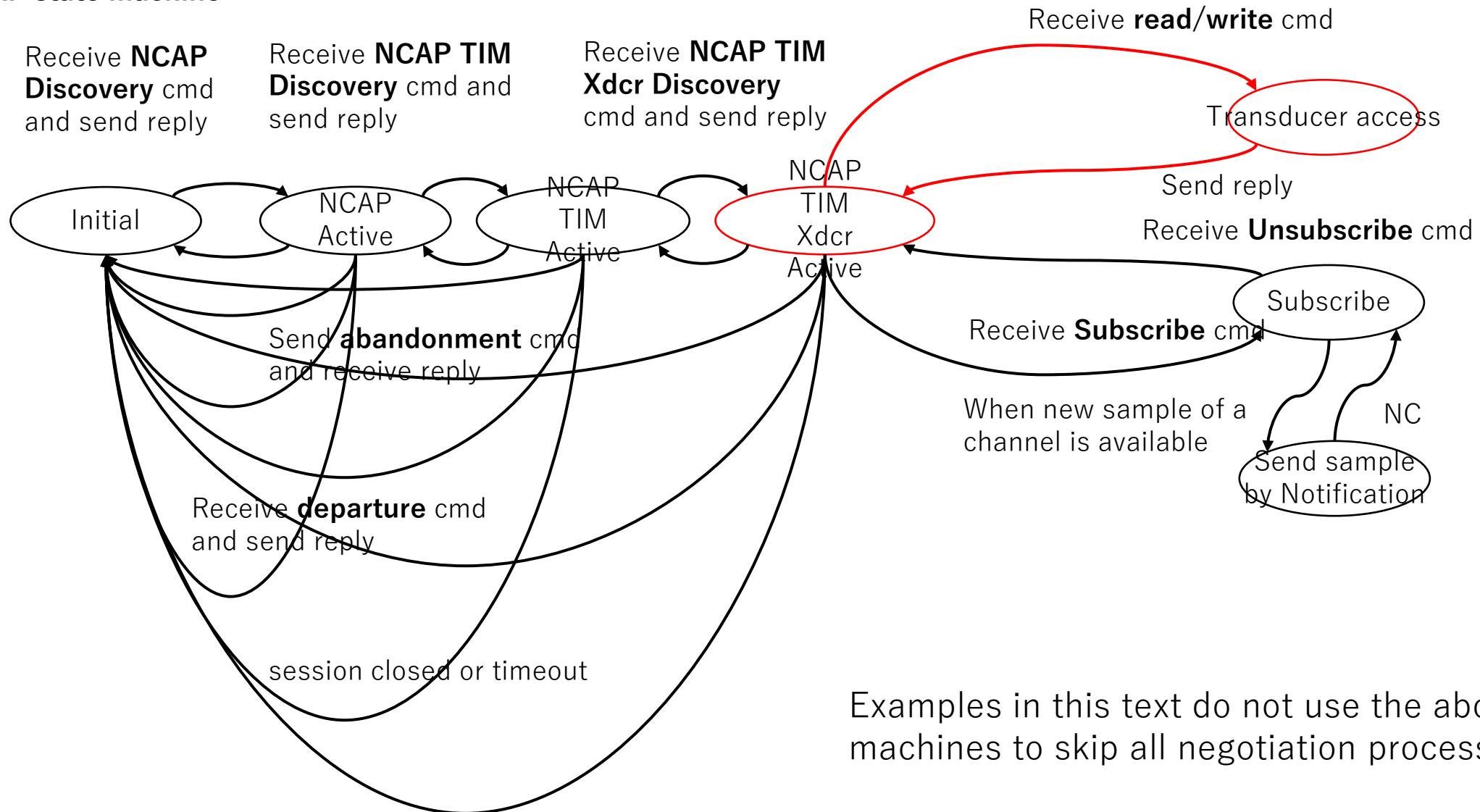


Examples in this text do not use the above state machines to skip all negotiation processes.

# Network Service State Transition

- Considering all above things, the state machine should be as follows:

NCAP state machine

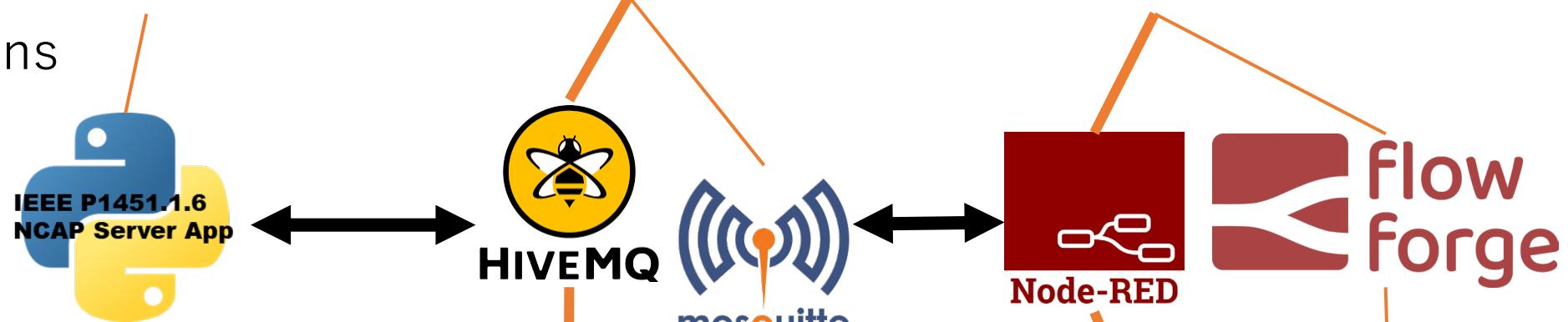


# System configurations in this lecture

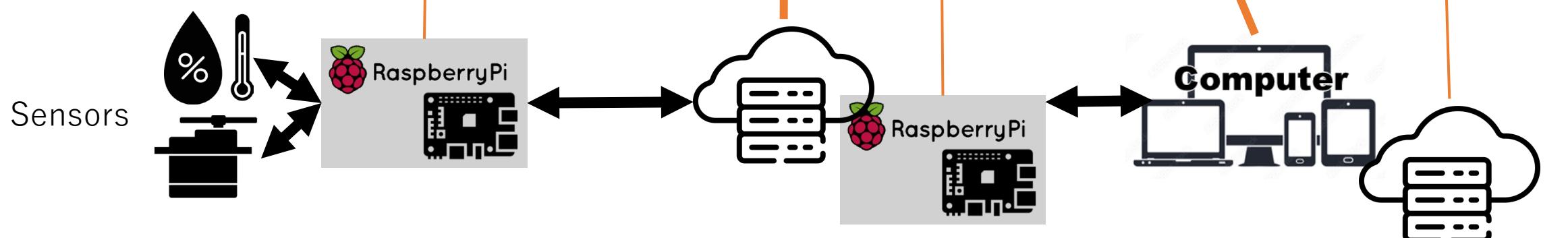
- Logical Connections



- Software Connections

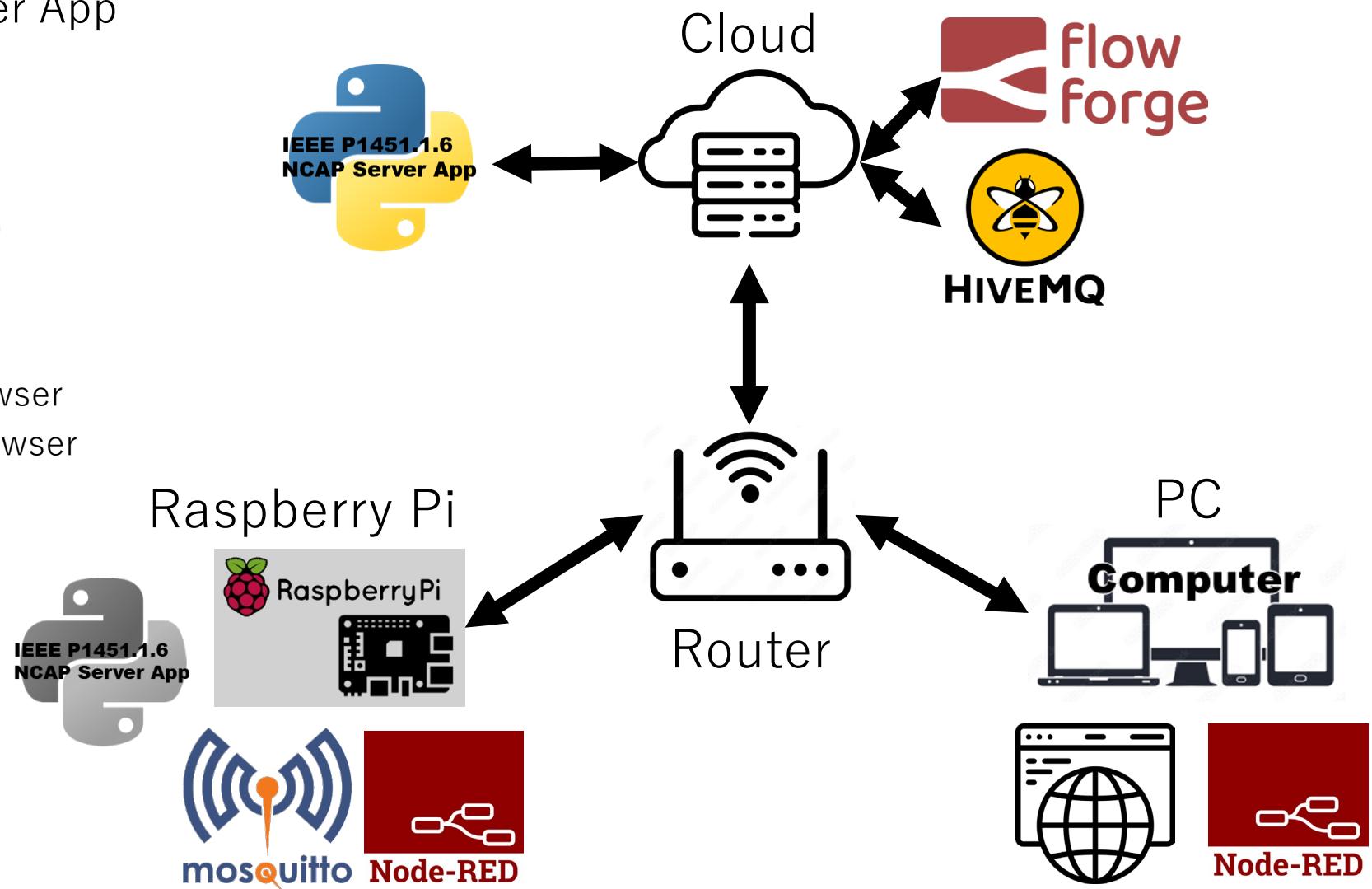


- Physical Connections



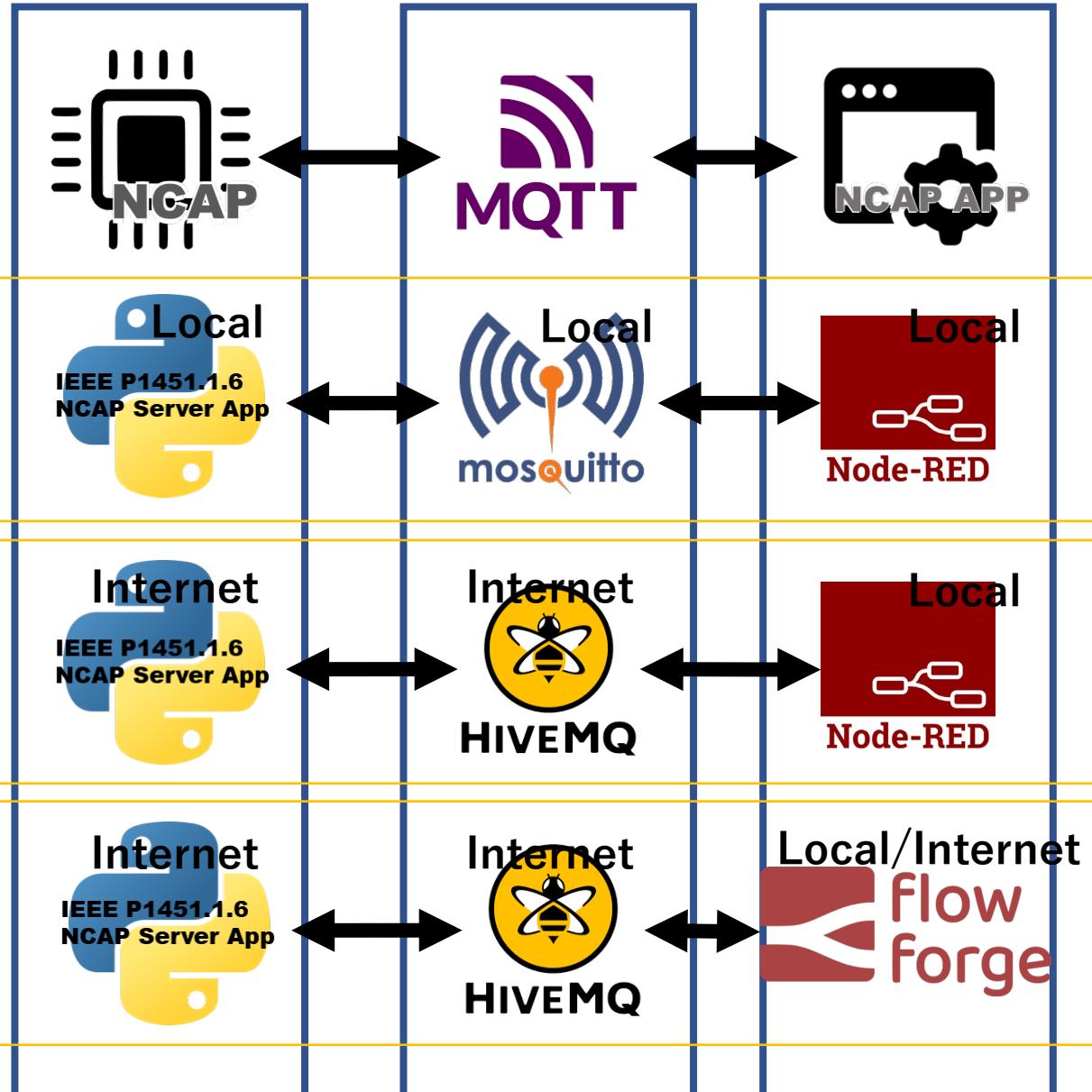
# Typical network configurations

- Select one from NCAP, broker and App
  - IEEE P1451.1.6 NCAP Server App
  - MQTT broker
    - HIVE MQ (Cloud)
    - mosquitto (Raspberry Pi)
    - Keio@COE broker (option)
  - NCAP Application
    - Node-RED
      - Raspberry Pi and Pi browser
      - Raspberry Pi and PC browser
      - PC and PC browser
    - Flowforge
      - Cloud

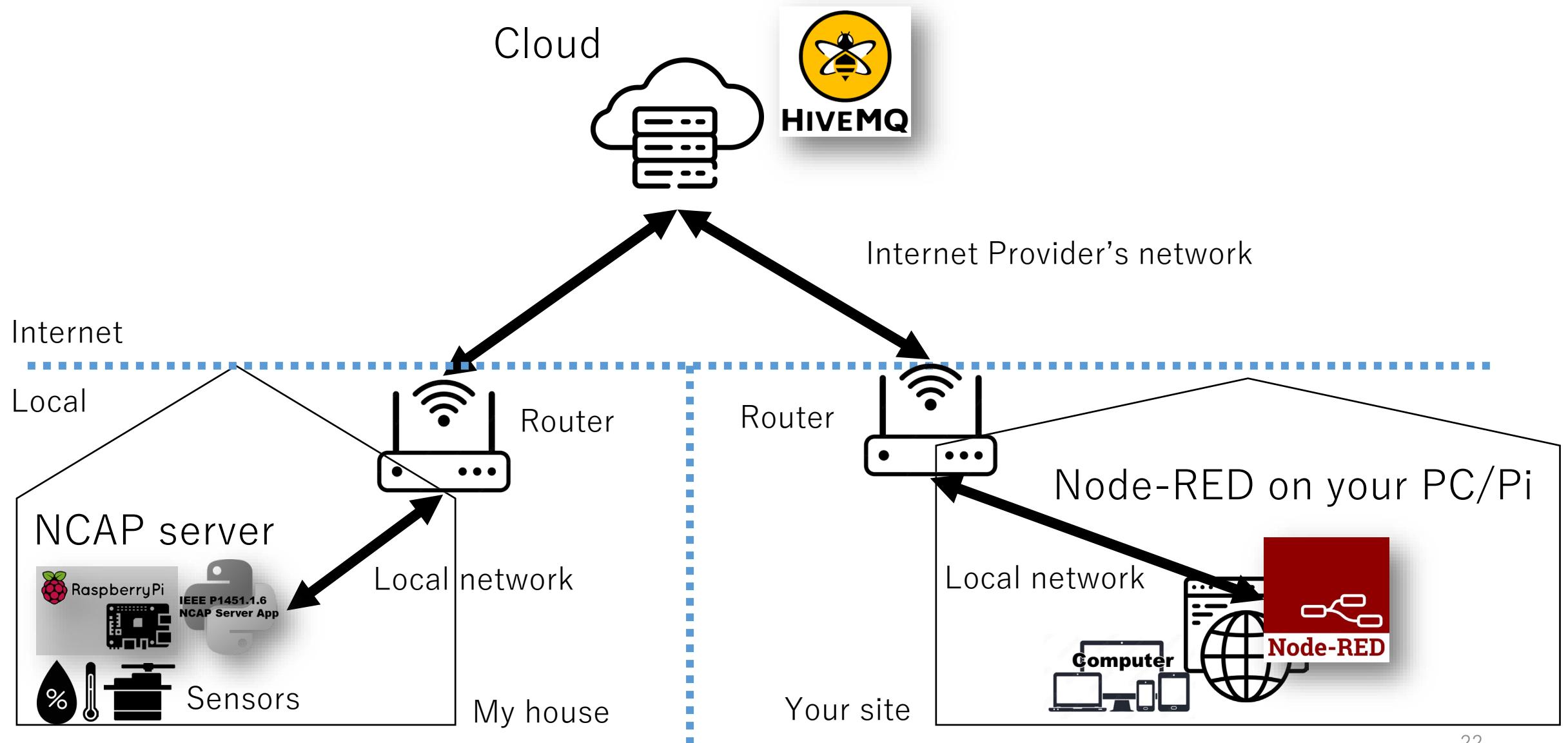


# Find your best combinations

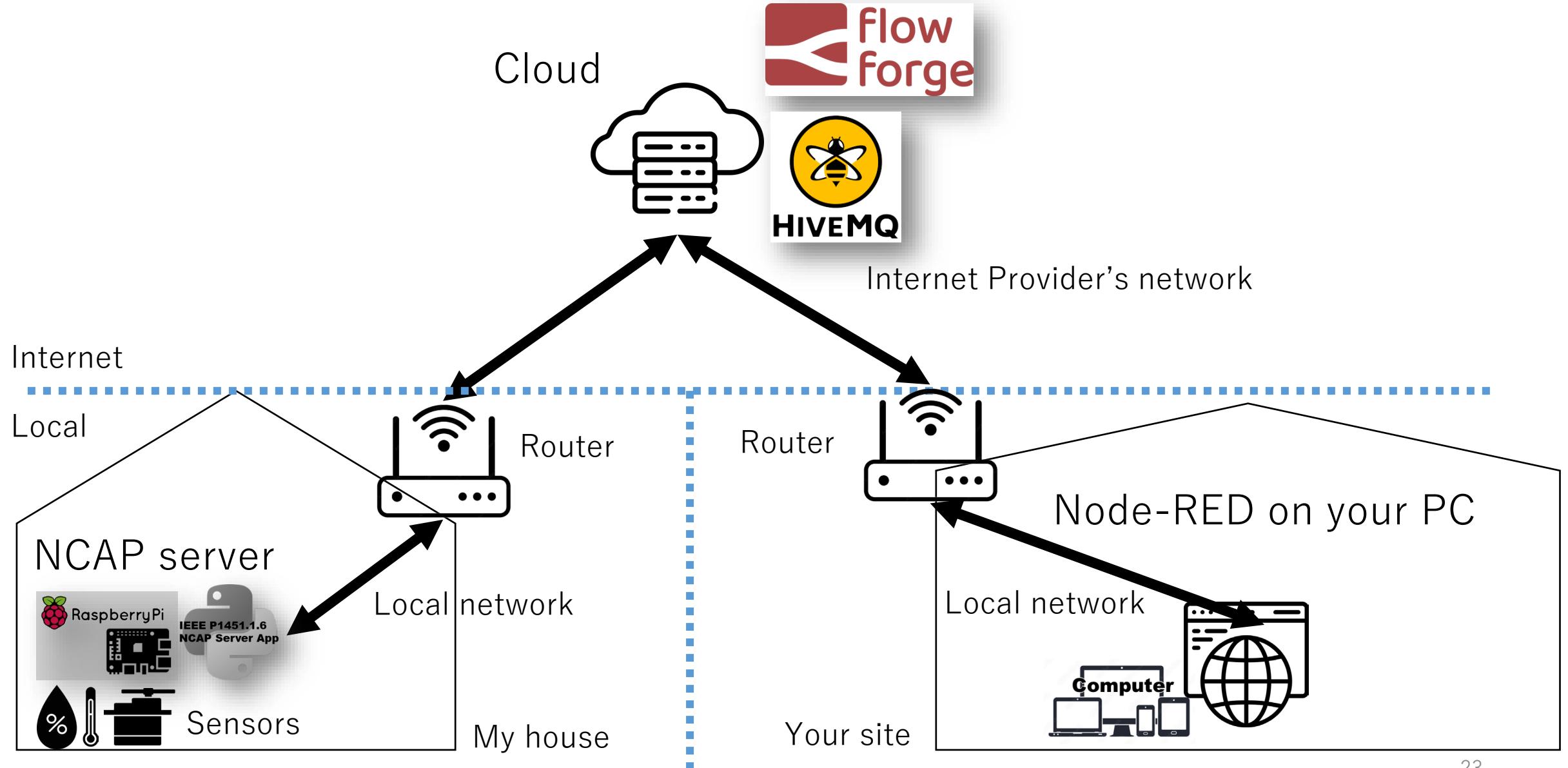
- If you want to learn all configurations or if you only have Raspberry Pi
  - If you cannot access the Internet (you must configure sensors)
- Recommended configuration of Local Node-RED on Pi
- If you do not have Raspberry Pi
- If you do not want to install anything



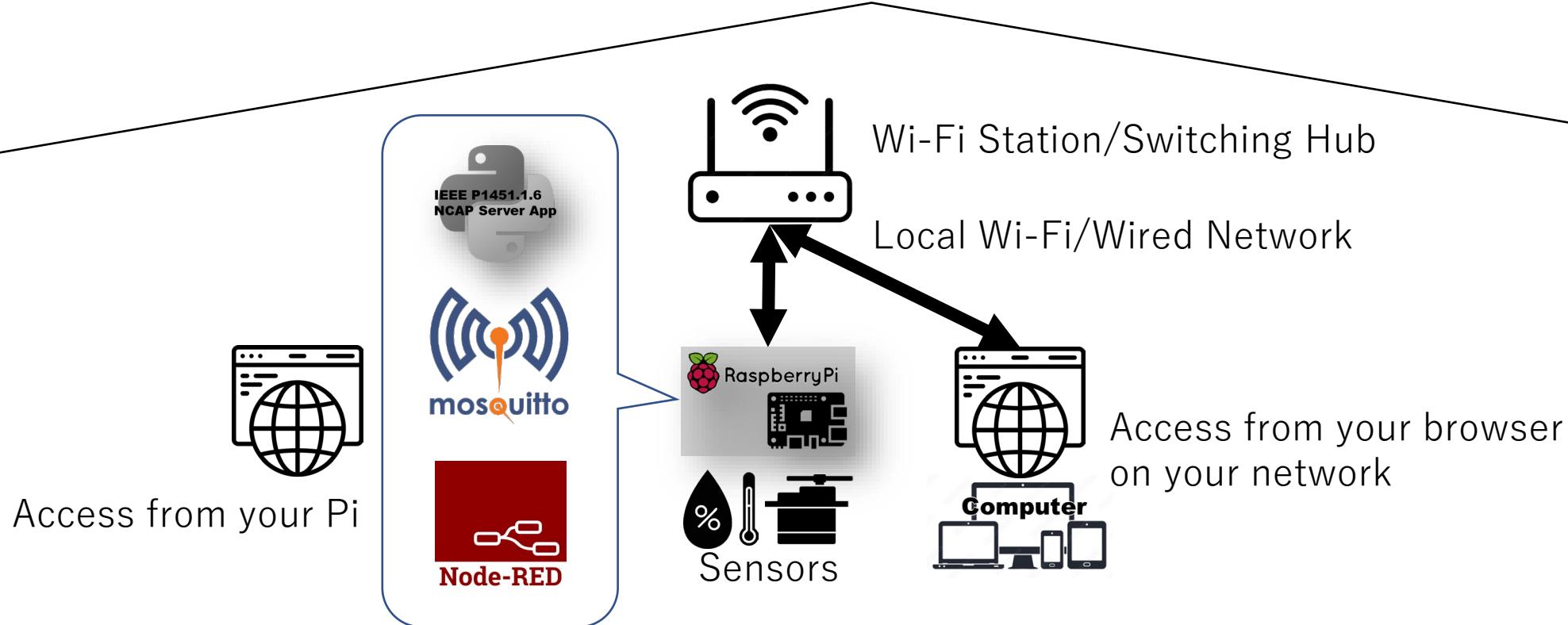
# System configurations (recommended)



# System configurations (only browser)



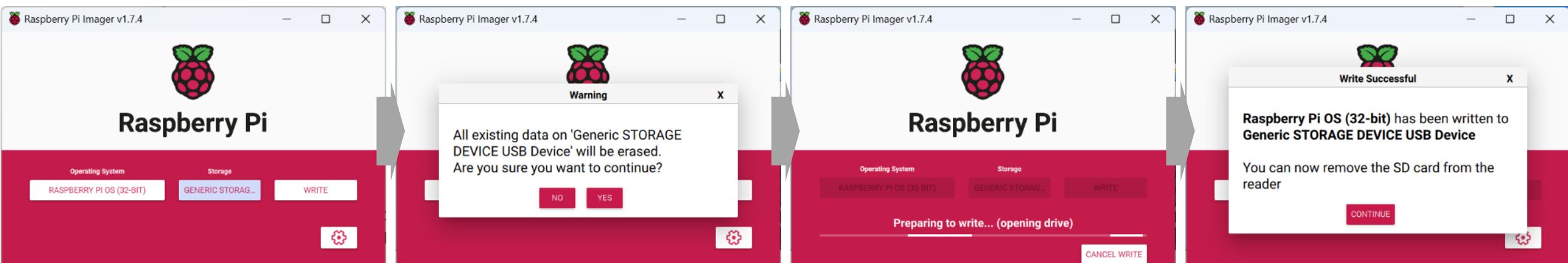
# System configurations (all in Local)



Install and setup your own NCAP server, Mosquitto, and Node-RED server  
(You can separately install them to different machines)

# Pi OS Installation (need to be completed in advance)

- <https://www.raspberrypi.com/software/>
- Download Pi Imager
- Select operatin system RASPBERRY PI OS(32-BIT) recommended
- Select Storage (Depends on your environments)
- Push Setting  to modify install options
  - You can set your hostname, default username/password, Wi-Fi, Local, and enable SSH
- Push WRITE to burn OS image to your micro-SD card memory
- Check configuration menu to set your default login, password, and other settings



- Select Raspberry Pi OS 32-BIT
- Select Generic Storage Device USB

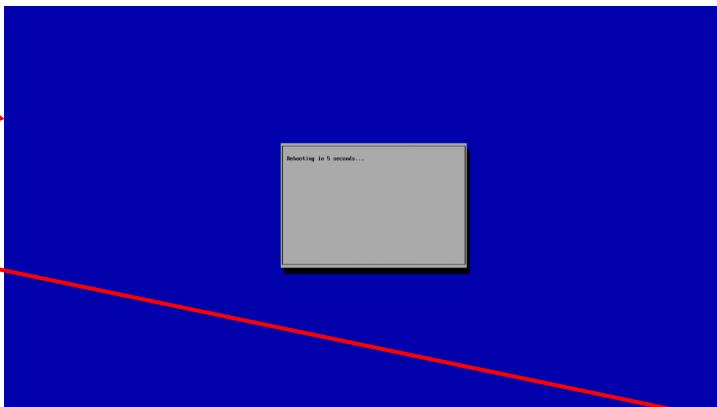
- Confirm your storage device is erased

- Wait OS installation

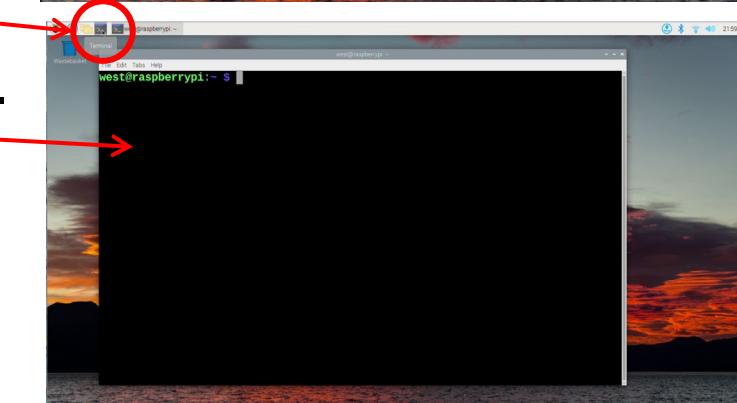
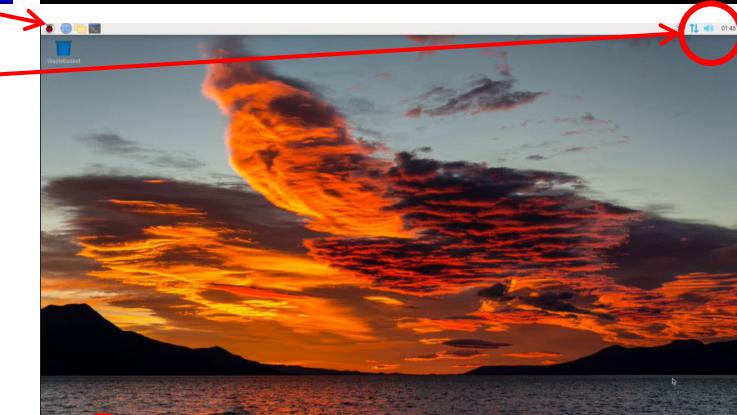
- Completed

# Start up your Raspberry Pi with micro-SD card

- First boot for set-up
- opening screen



- Verify the configured network connection is established
  - Connect via Wi-Fi or wired LAN
- Open Terminal
- Shell starts with the username and machine name you set.



# Install Node-RED

- Input the following command as one-liner command
  - bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
- Press **y ↵** twice
- Installation in progress
- Press **n ↵** for finalize settings

```
west@raspberrypi:~$ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
ed
to setup security.

### ADDITIONAL RECOMMENDATIONS #####
- Remove the /etc/sudoers.d/010_pi-nopasswd file to require entering your password when performing any sudo/root commands:
  sudo rm -f /etc/sudoers.d/010_pi-nopasswd

- You can customise the initial settings by running:
  node-red admin init

- After running Node-RED for the first time, change the ownership of the settings file to 'root' to prevent unauthorised changes:
  sudo chown root:root ~/node-red/settings.js

Would you like to customise the settings now (y/N) ? y
```

```
west@raspberrypi:~$ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
Running Node-RED install for user west at /home/west on raspbian
This can take 20-30 minutes on the slower Pi versions - please wait.

Stop Node-RED
Remove old version of Node-RED
Remove old version of Node.js
Install Node.js 16 LTS
Clean npm cache
Install Node-RED core
Move global nodes to local
Npm rebuild existing nodes
Install extra Pi nodes
Add shortcut commands
Update systemd script

Any errors will be logged to /var/log/nodered-install.log
```

```
west@raspberrypi:~$ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

This script checks the version of node.js installed is 14 or greater. It will try to install node 16 if none is found. It can optionally install node 12, 14, 16 or 18 LTS for you.

If necessary it will then remove the old core of Node-RED, before then installing the latest version. You can also optionally specify the version required.

It also tries to run 'npm rebuild' to refresh any extra nodes you have installed that may have a native binary component. While this normally works ok, you need to check that it succeeds for your combination of installed nodes.

To do all this it runs commands as root - please satisfy yourself that this will not damage your Pi, or otherwise compromise your configuration.
If in doubt please backup your SD card first.

See the optional parameters by re-running this command with --help

Are you really sure you want to do this? [y/N] ? y
Would you like to install the Pi-specific nodes? [y/N] ? y

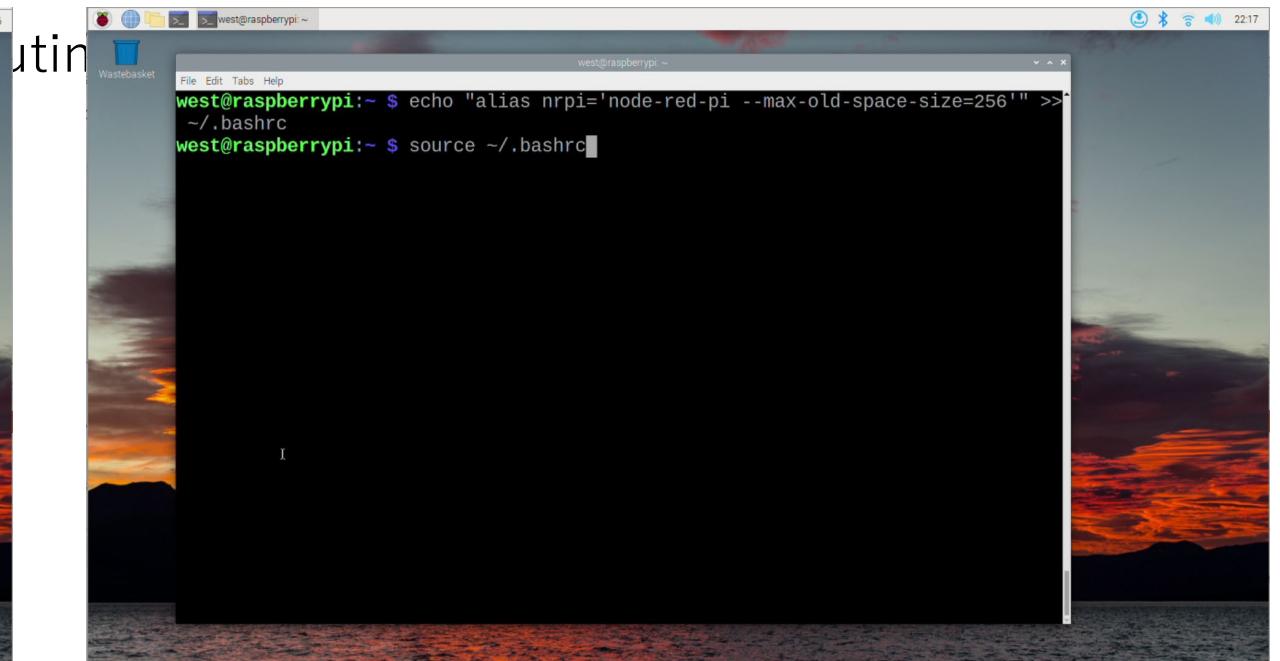
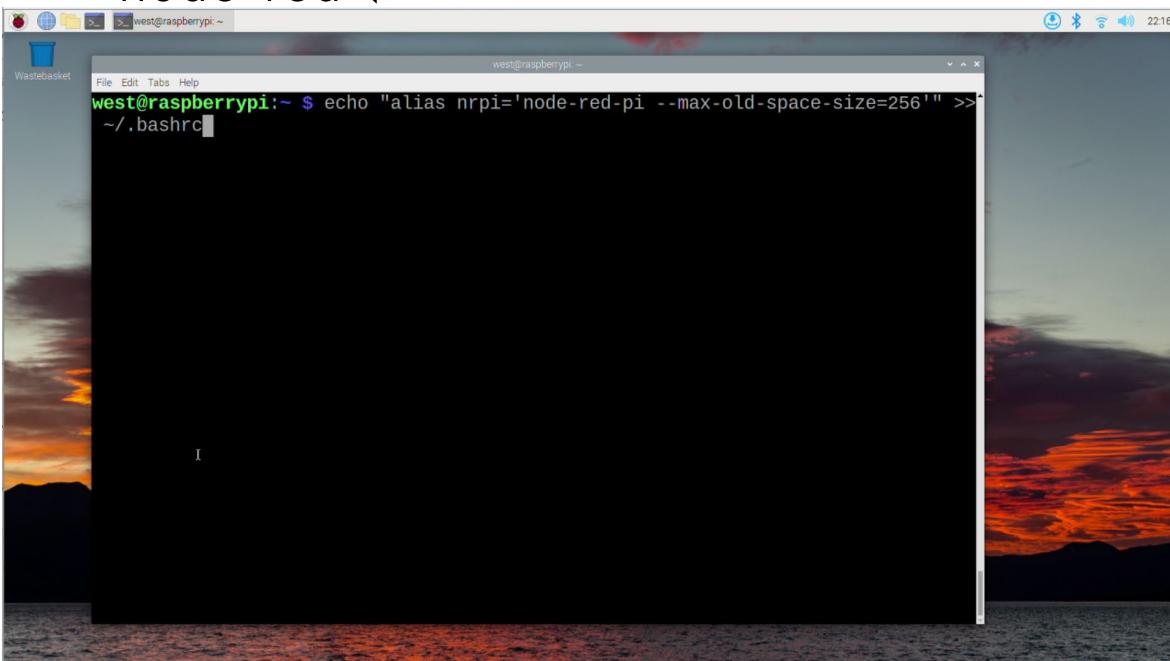
Running Node-RED install for user west at /home/west on raspbian
This can take 20-30 minutes on the slower Pi versions - please wait.

Stop Node-RED
Remove old version of Node-RED
Remove old version of Node.js
Install Node.js 16 LTS
Clean npm cache
Install Node-RED core
Move global nodes to local
Npm rebuild existing nodes
Install extra Pi nodes
Add shortcut commands
Update systemd script

Any errors will be logged to /var/log/nodered-install.log
```

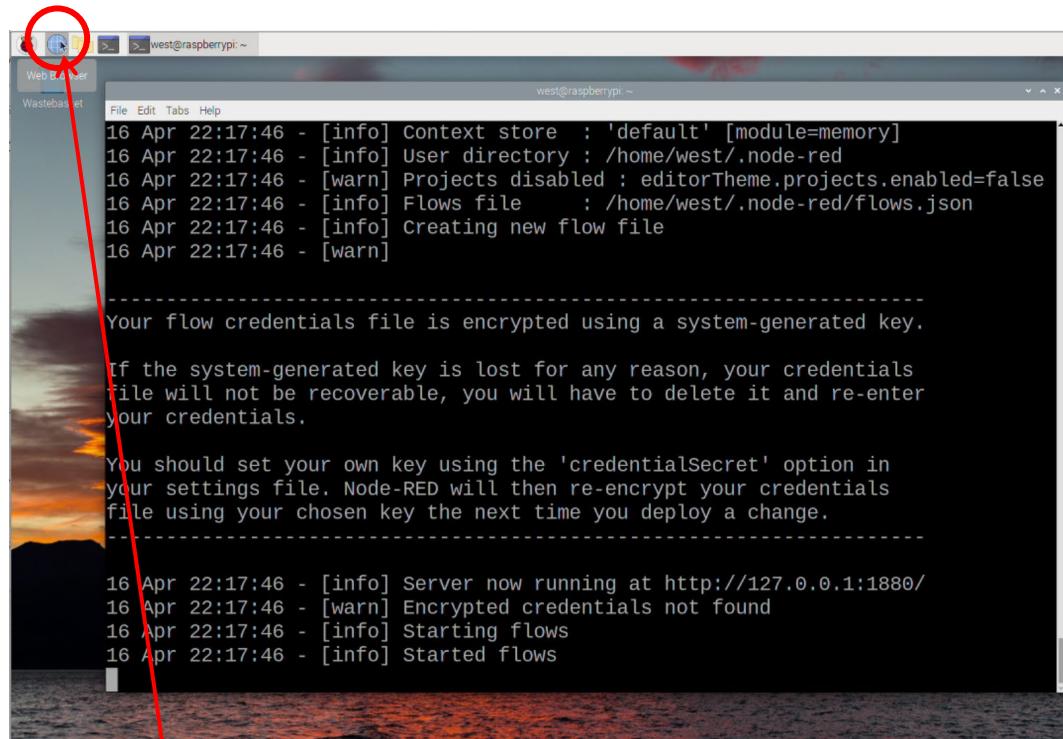
# Avoid memory shortage problem (option)

- If you do not have Raspberry-Pi 4B, the following set-up is required to avoid the memory shortage problems.
- Type and enter the following commands (only first time)
  - echo "alias nrpi='node-red-pi --max-old-space-size=256'" >> ~/.bashrc ↵
  - source ~/.bashrc ↵
- Run Node-RED
  - node-red ↵



# Run your browser to access your Node-RED

- Node-RED is a web server for IoT design center
- Access to <http://localhost:1880> or <http://127.0.0.1:1880>



```

west@raspberrypi: ~
Web browser
Wastebasket
File Edit Tabs Help
16 Apr 22:17:46 - [info] Context store : 'default' [module=memory]
16 Apr 22:17:46 - [info] User directory : /home/west/.node-red
16 Apr 22:17:46 - [warn] Projects disabled : editorTheme.projects.enabled=false
16 Apr 22:17:46 - [info] Flows file      : /home/west/.node-red/flows.json
16 Apr 22:17:46 - [info] Creating new flow file
16 Apr 22:17:46 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

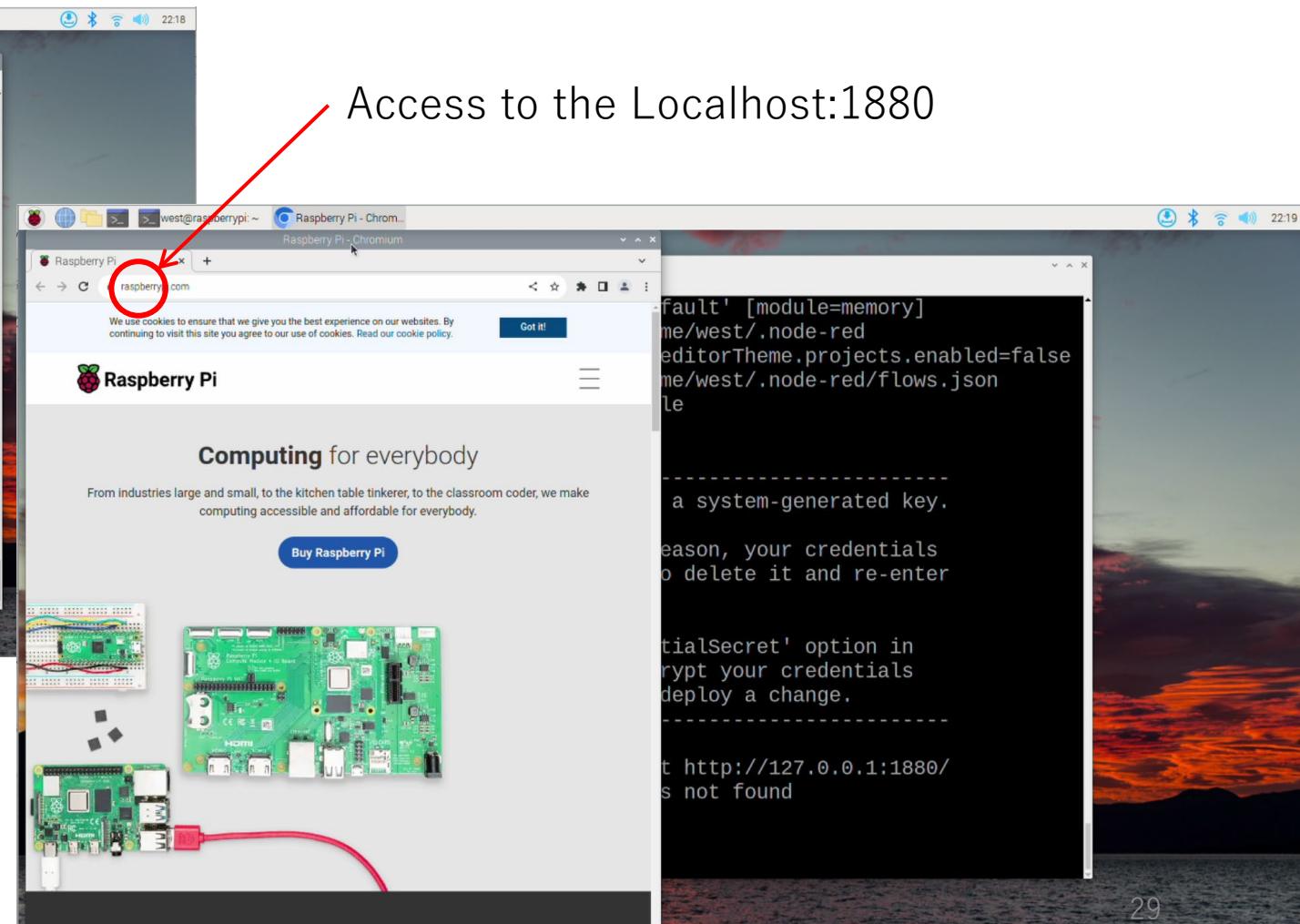
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.

16 Apr 22:17:46 - [info] Server now running at http://127.0.0.1:1880/
16 Apr 22:17:46 - [warn] Encrypted credentials not found
16 Apr 22:17:46 - [info] Starting flows
16 Apr 22:17:46 - [info] Started flows

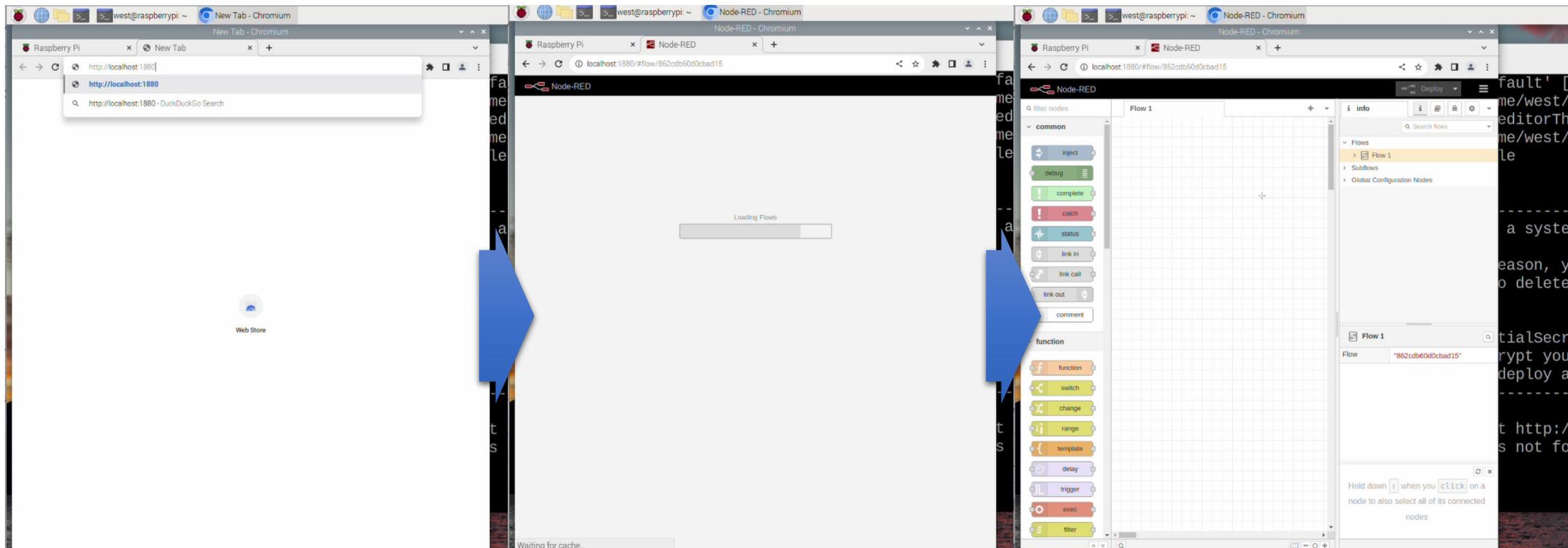
```

Open browser



# Access to Node-RED

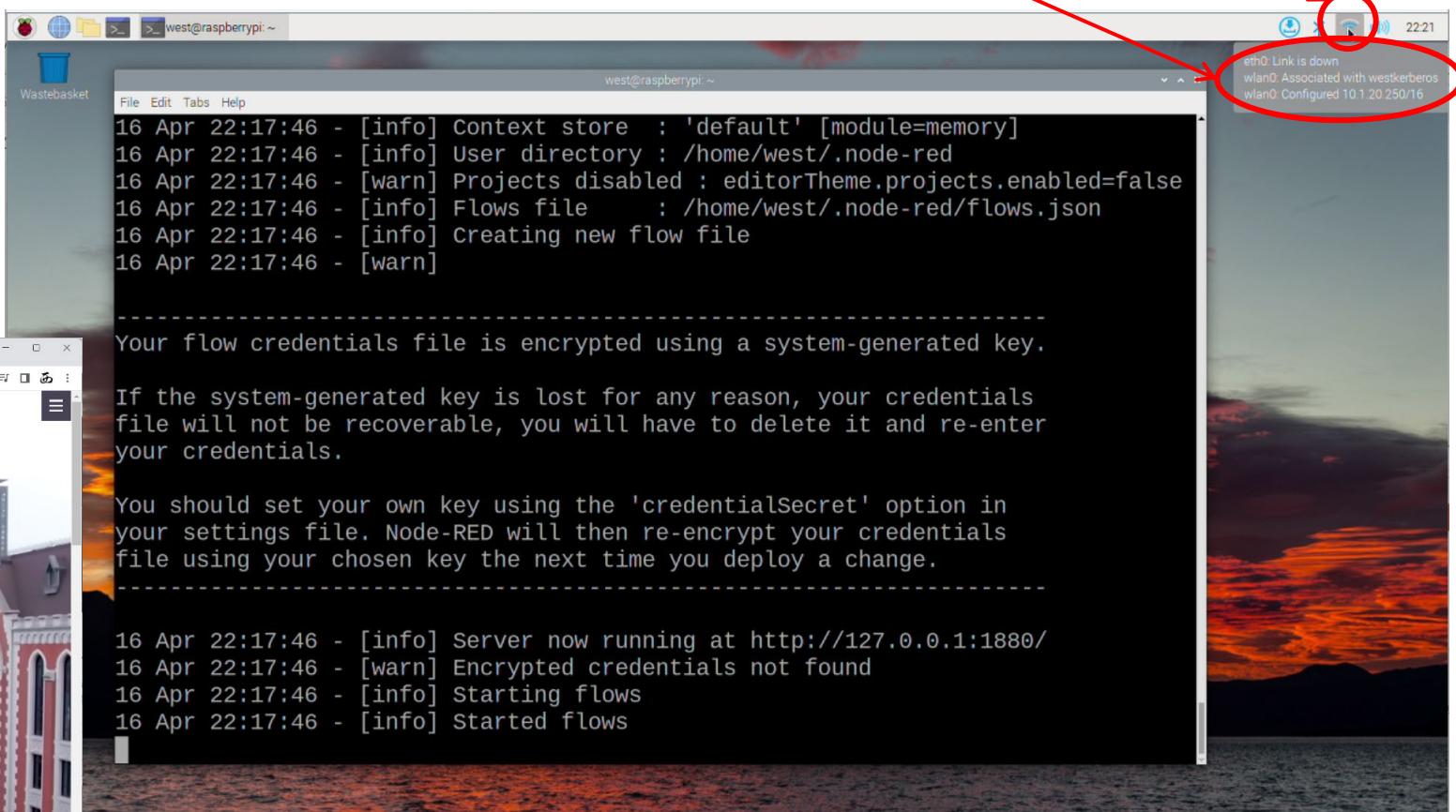
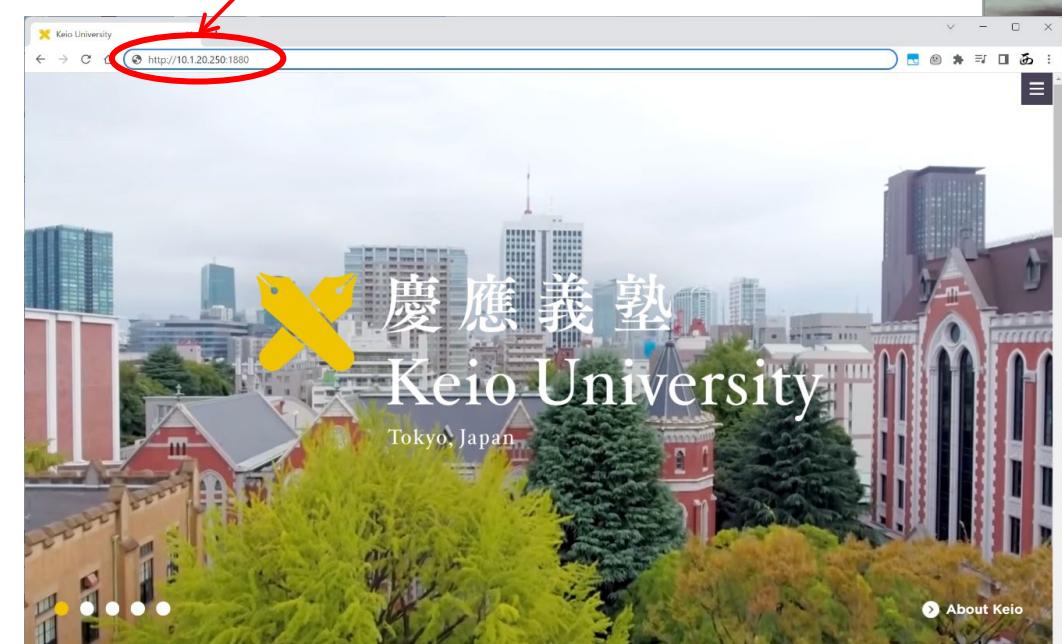
- Input <http://localhost:1880>
- Press NEXT to show top menu
- You can also install Node-RED to your PC (such as Windows / Ubuntu ).
  - <https://nodered.org/docs/getting-started/>





# Another Access method (from your PC)

- Check IP address by putting your mouse cursor on the taskbar's network icon
- **In this case**, the IP address of raspberry Pi is 10.1.20.250
- Access the **YOUR Pi's IP:1880** from another PC (Win, Mac, etc.)



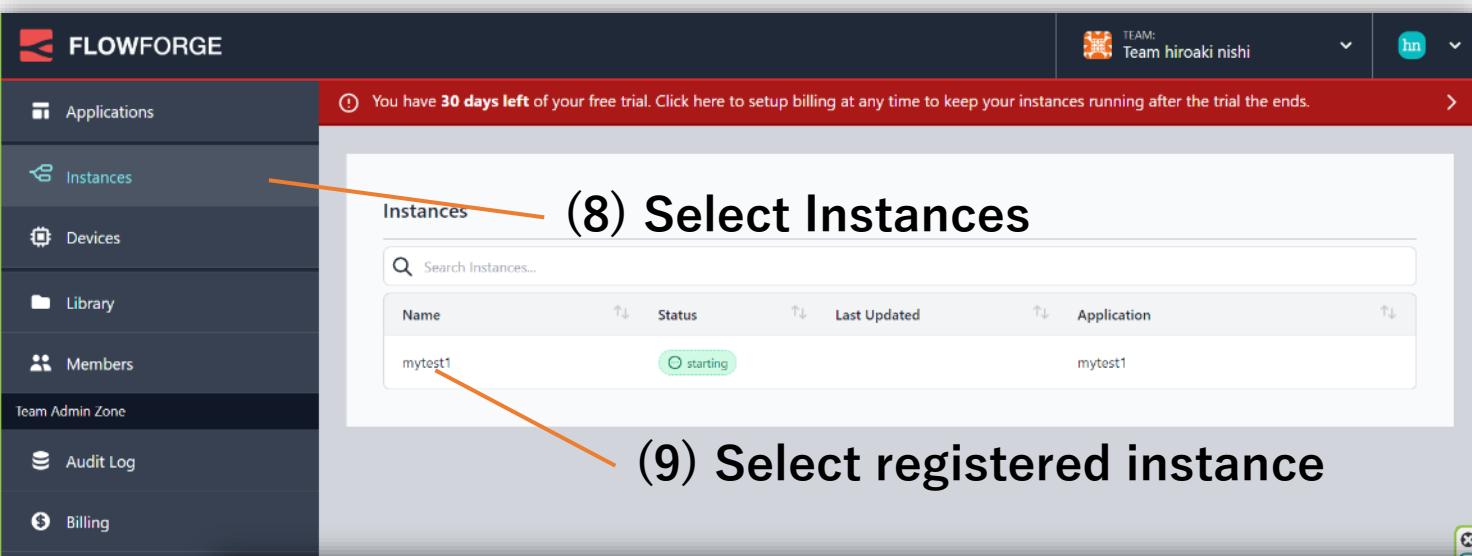
# Use of Flowforge (option / if you do not have Pi)



The screenshot shows the FlowForge website with the following steps highlighted:

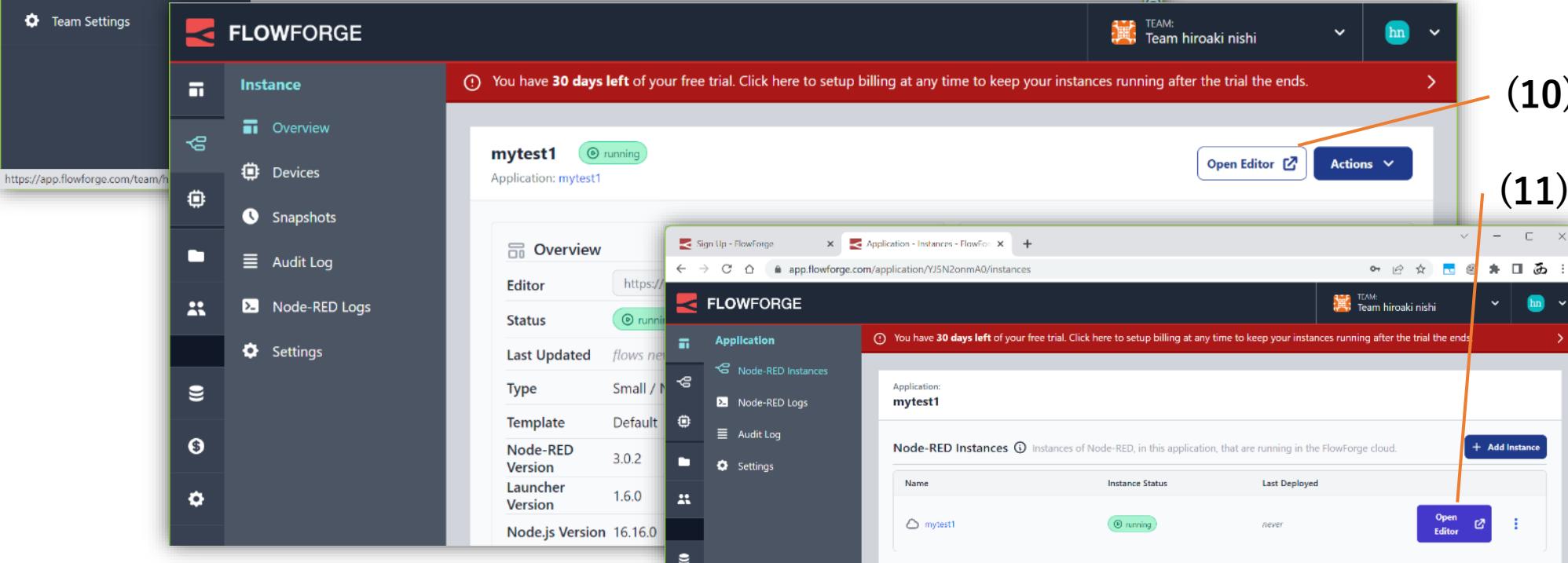
- (1) Access to <https://flowforge.com>
- (2) Select FREE TRIAL
- (3) Input required forms  
USERNAME: (your username)  
FULL NAME: (your name)  
E-MAIL ADDRESS: (your e-mail address)  
PASSWORD: パスワード (your new password for this site)
- (4) Respond to the email sent to your registered address
- (5) Input Application Name and Instance Name
- (6) Select Small Free Trial
- (7) Select Create Application

# Set-up Flowforge



The screenshot shows the Flowforge Instances page. On the left sidebar, the 'Instances' option is selected. The main area displays a table of registered instances. One instance, 'mytest1', is highlighted with an orange arrow pointing to it. The table columns include Name, Status, Last Updated, and Application.

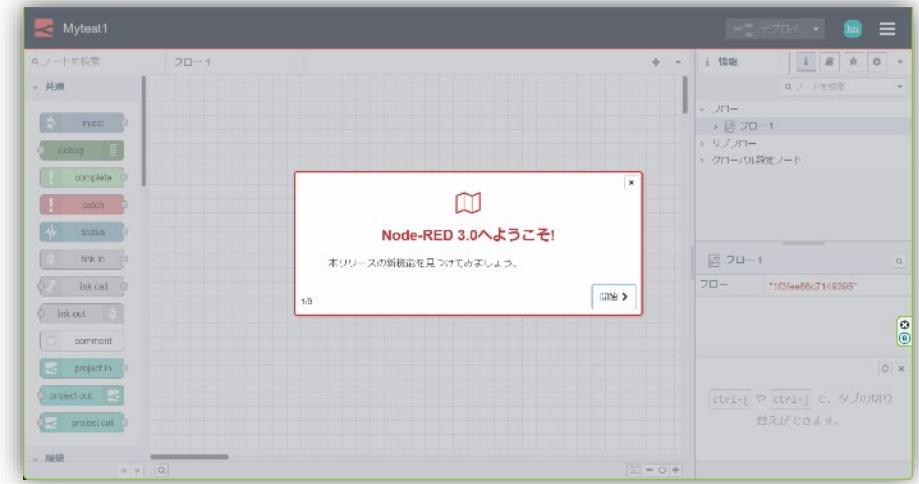
**(8) Select Instances**



The screenshot shows the Flowforge Instance details page for 'mytest1'. The left sidebar lists various instance-related options: Overview, Devices, Snapshots, Audit Log, Node-RED Logs, and Settings. The main area shows the instance status as 'running' and provides an 'Open Editor' button. A secondary window titled 'Sign Up - FlowForge' is also visible in the background.

**(9) Select registered instance**

**(12) Node-RED will open**

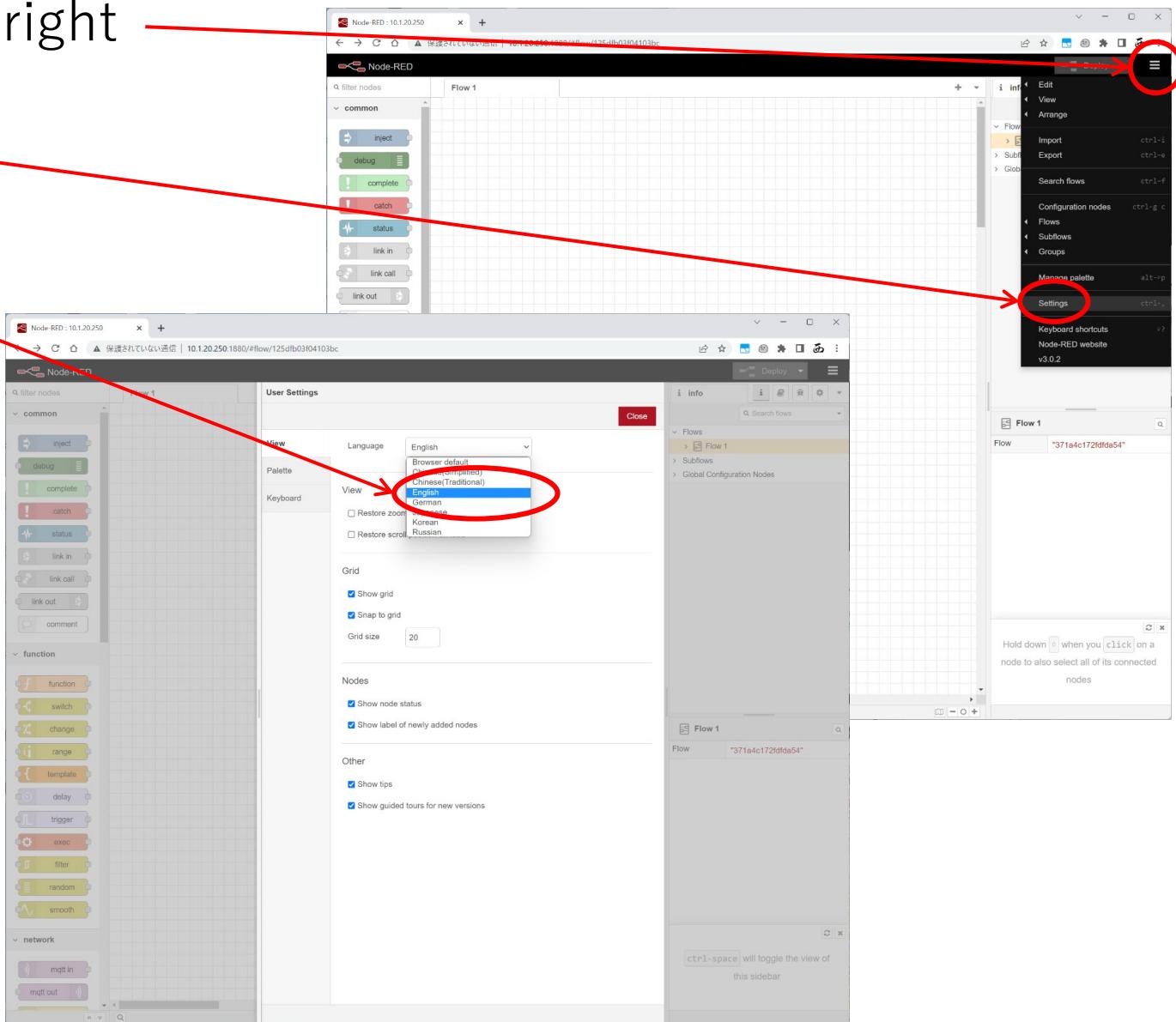
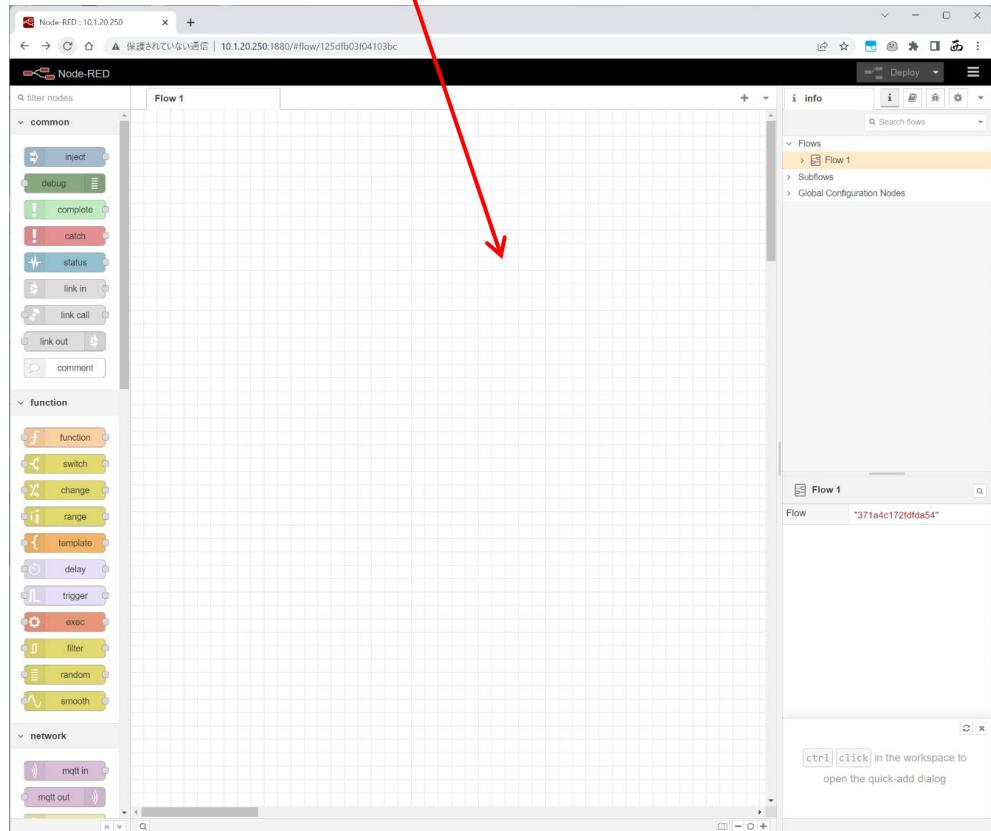


**(10) Select Open Editor**

**(11) Select Open Editor**

# Change Node-RED language

- Press three-bar icon on the top right
- Press Settings
- Select language
- Working window

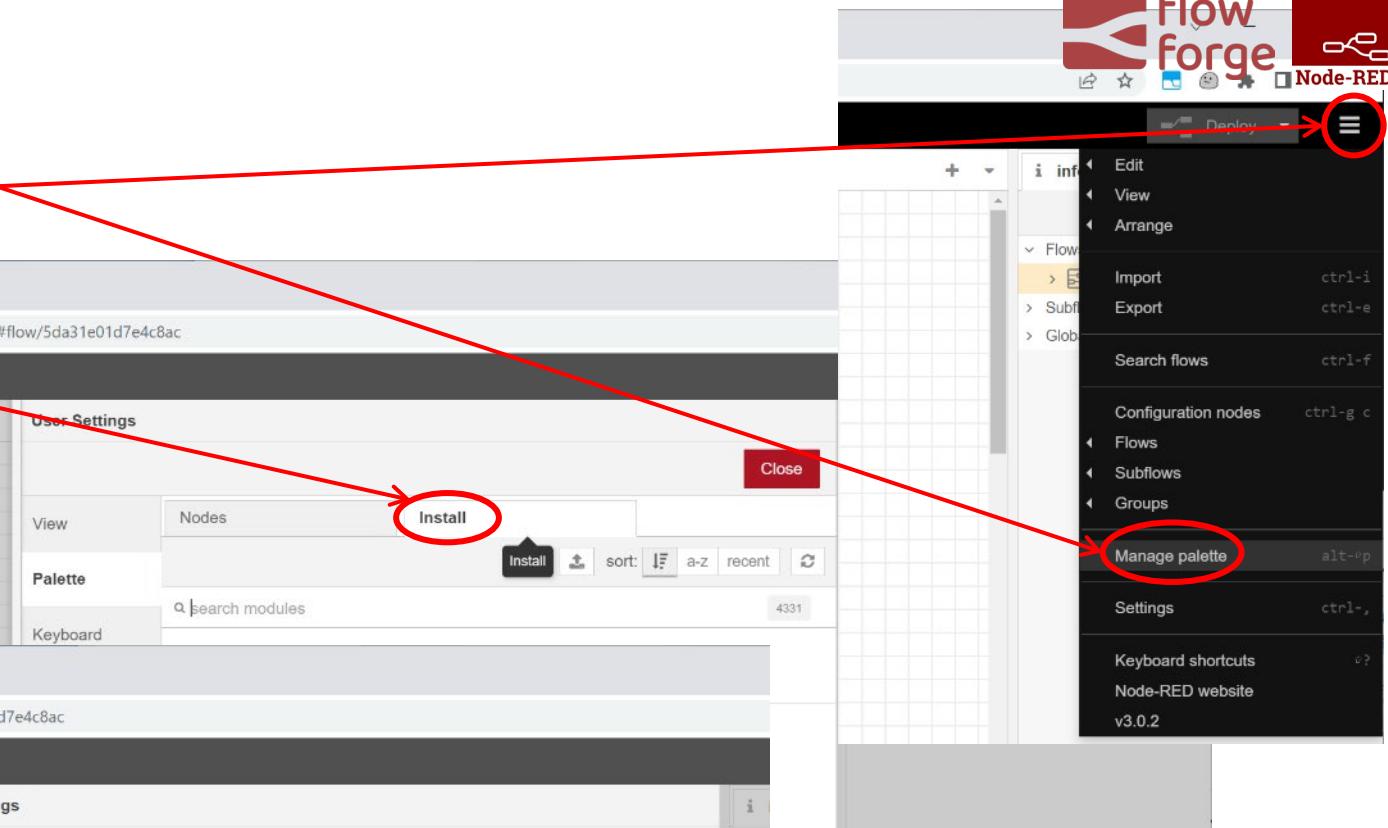
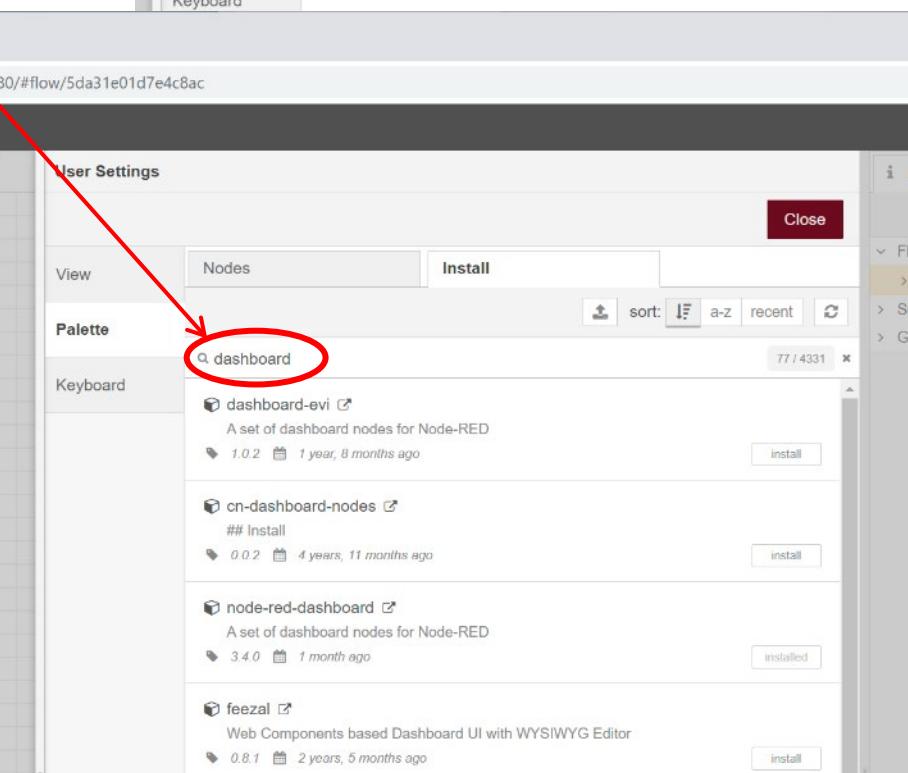
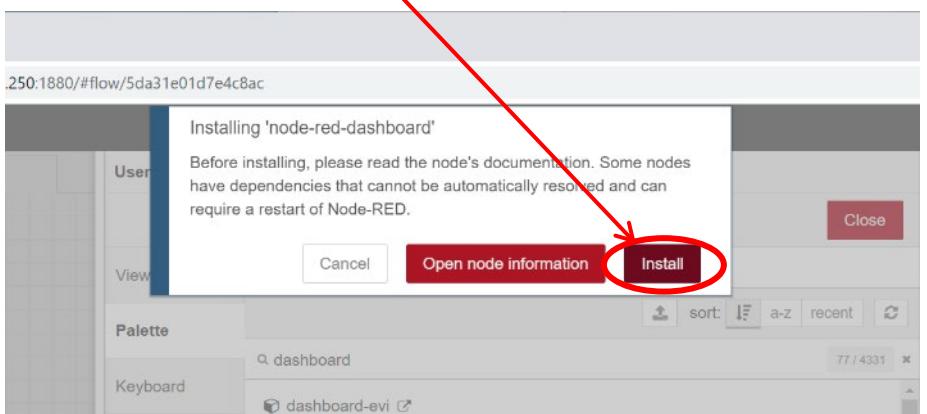


# Design application on Node-RED

- Why Node-RED
  - Low-Code design
    - Do not type codes, but put nodes and connect them to design data flows
  - Many IoT interfaces
    - Dashboard
    - Cloud connections
    - Raspberry Pi connections
    - MQTT/HTTP/REST etc. etc.
    - Speech, Notification, email, SNS, etc. etc.
    - More than 4,000 add-ons are available
  - Quickly design IoT services; it's a magic.
- Copy and import json file from Github repository to quickly start NCAP application
  - <https://github.com/westlab/IEEEP1451.1.6-2023>

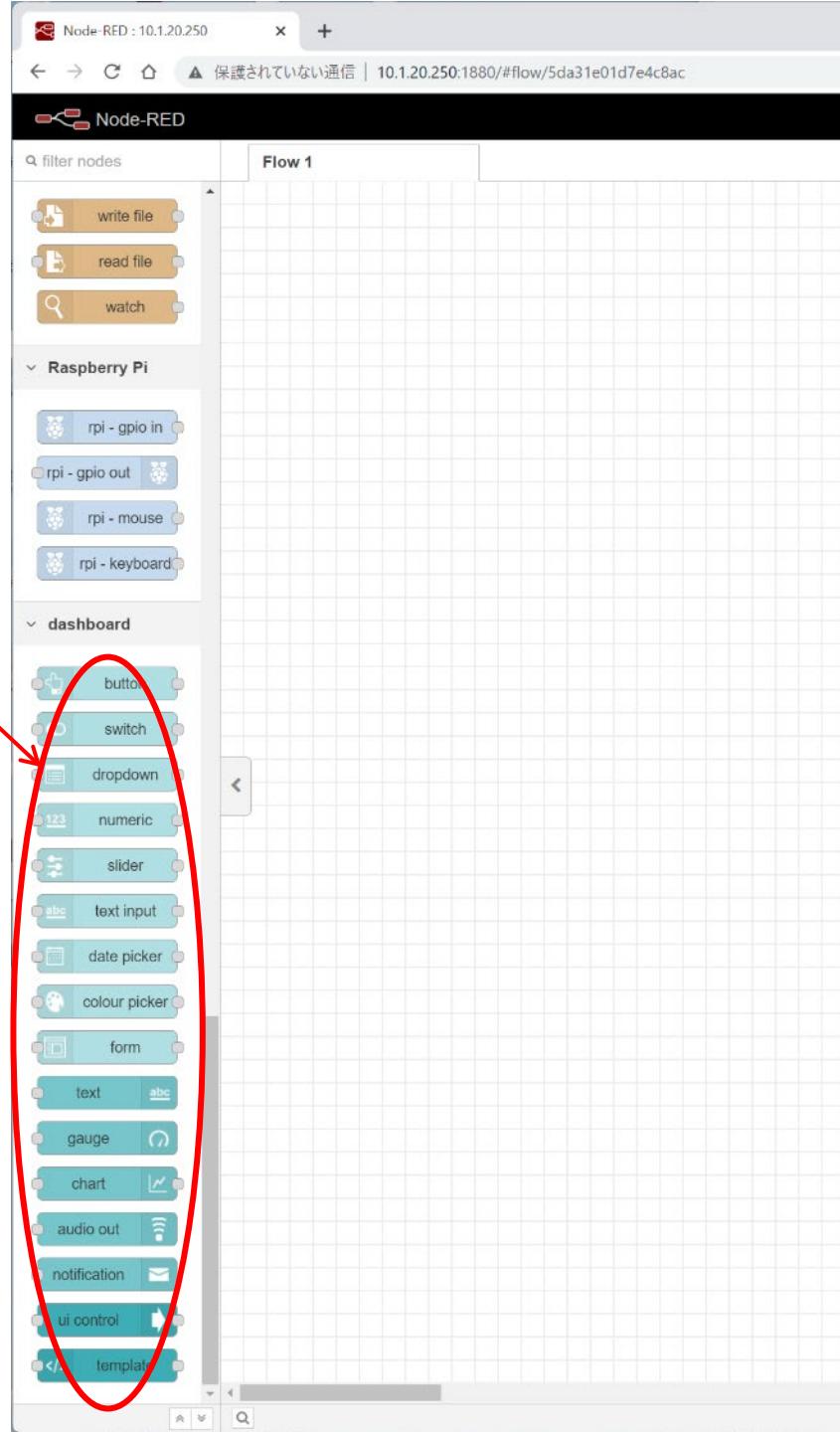
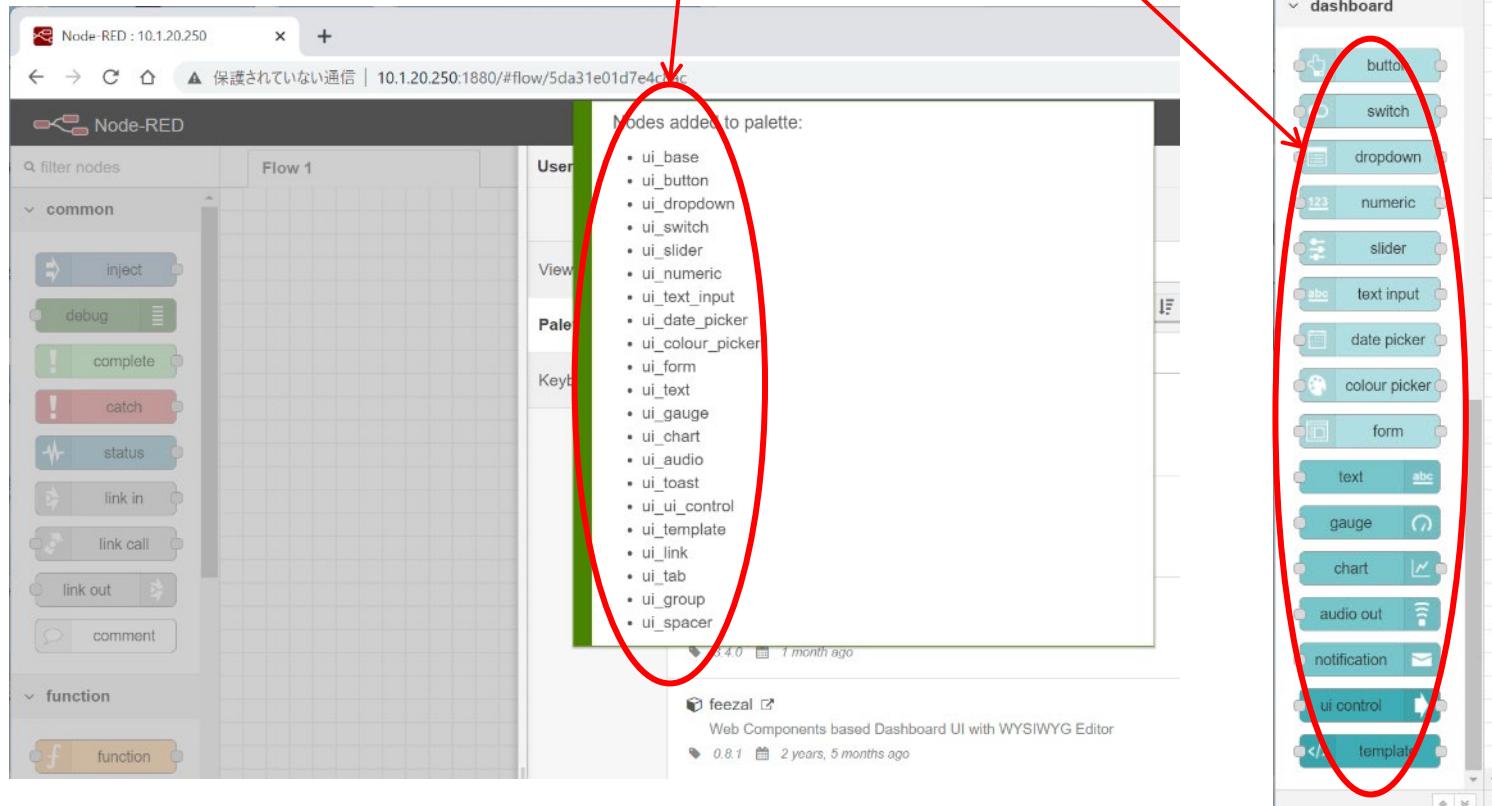
# Dashboard installation

- Press Menu icon and Manage pallet
- Select Install tab
- Input dashboard in search window
- Press install



# Dashboard nodes

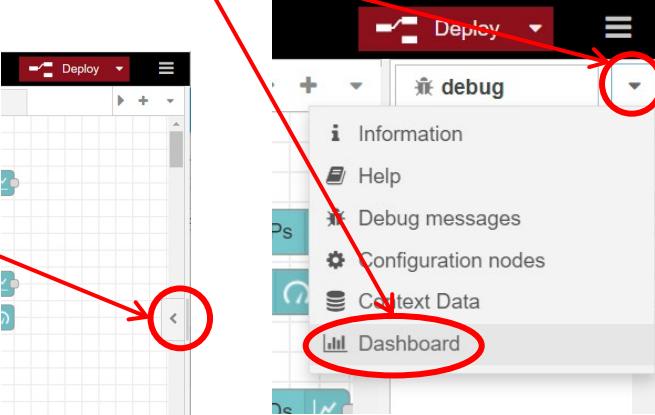
- Installed nodes for dashboard



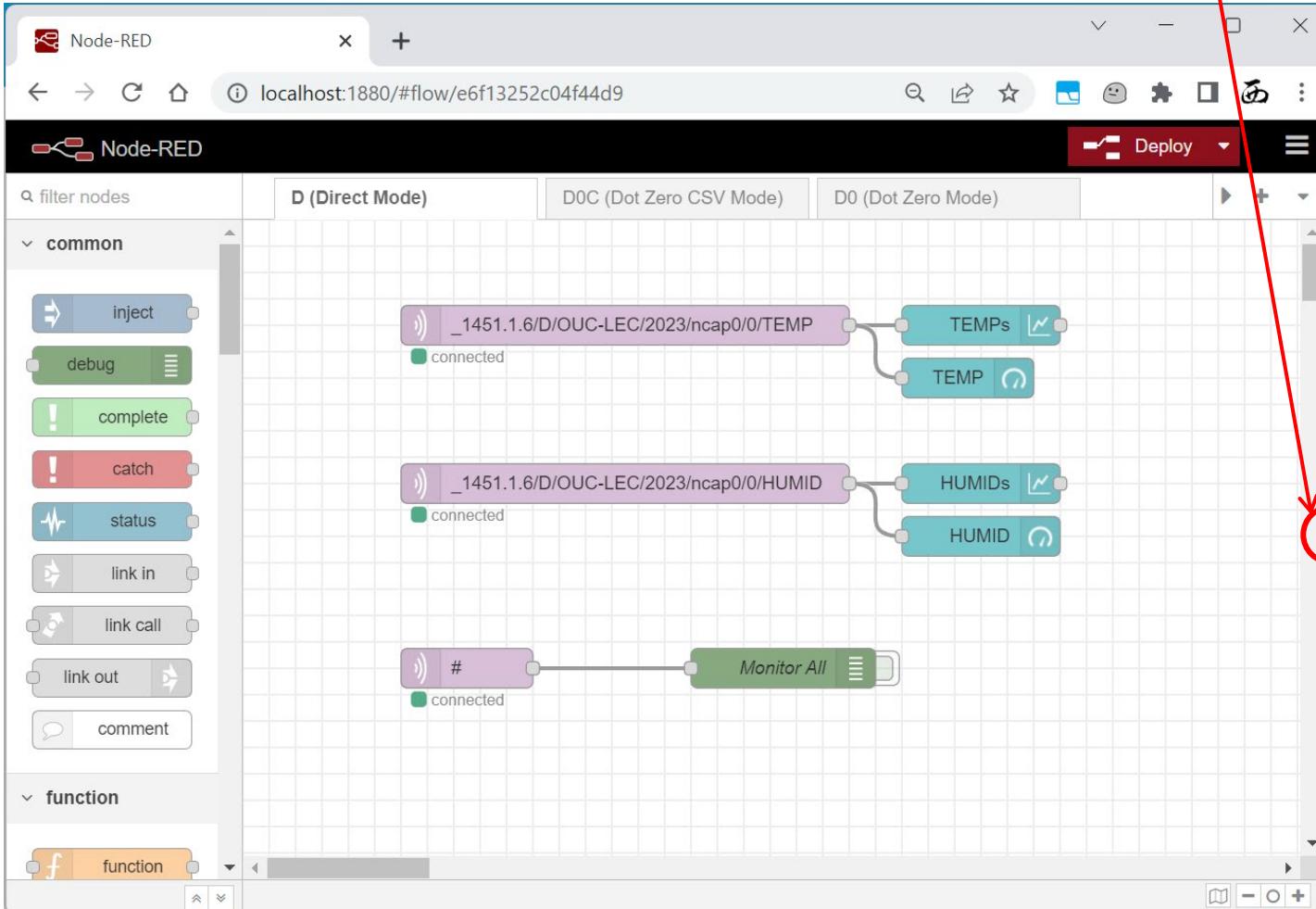
# Direct Mode

- NCAP publish sensor data and subscribe them.
- Visualize the data

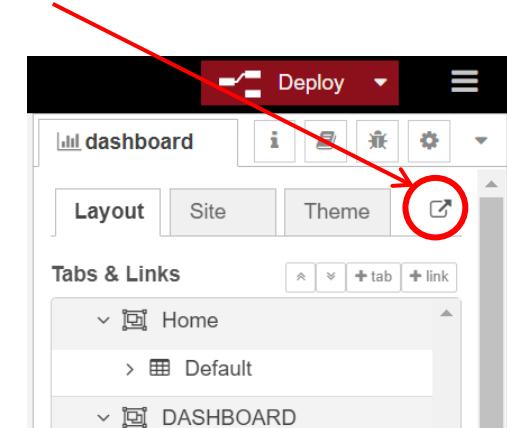
(2) Push open menu and Dashboard



(1) Push sidebar open switch

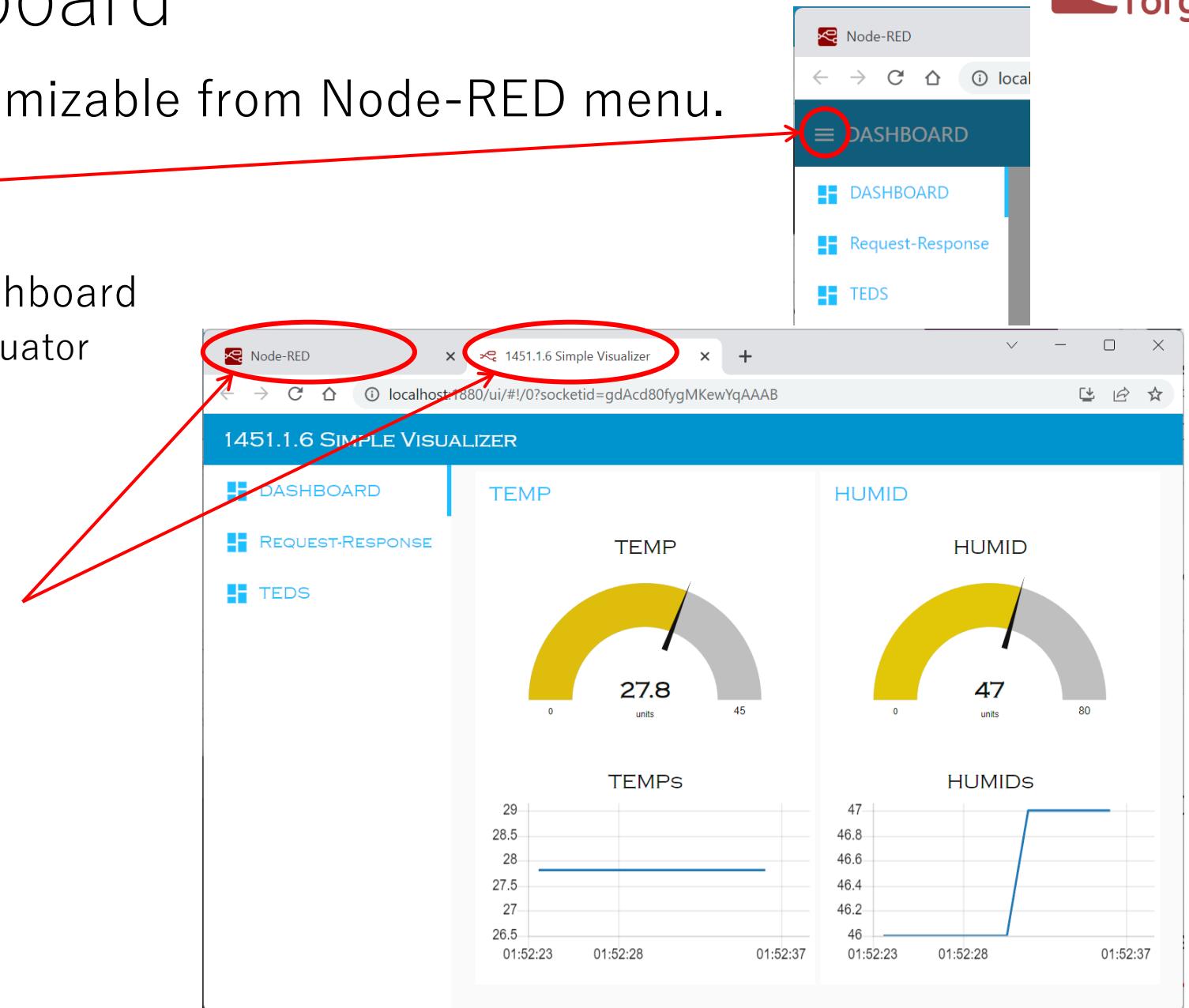


(3) Push open Dashboard link



# Direct Mode Dashboard

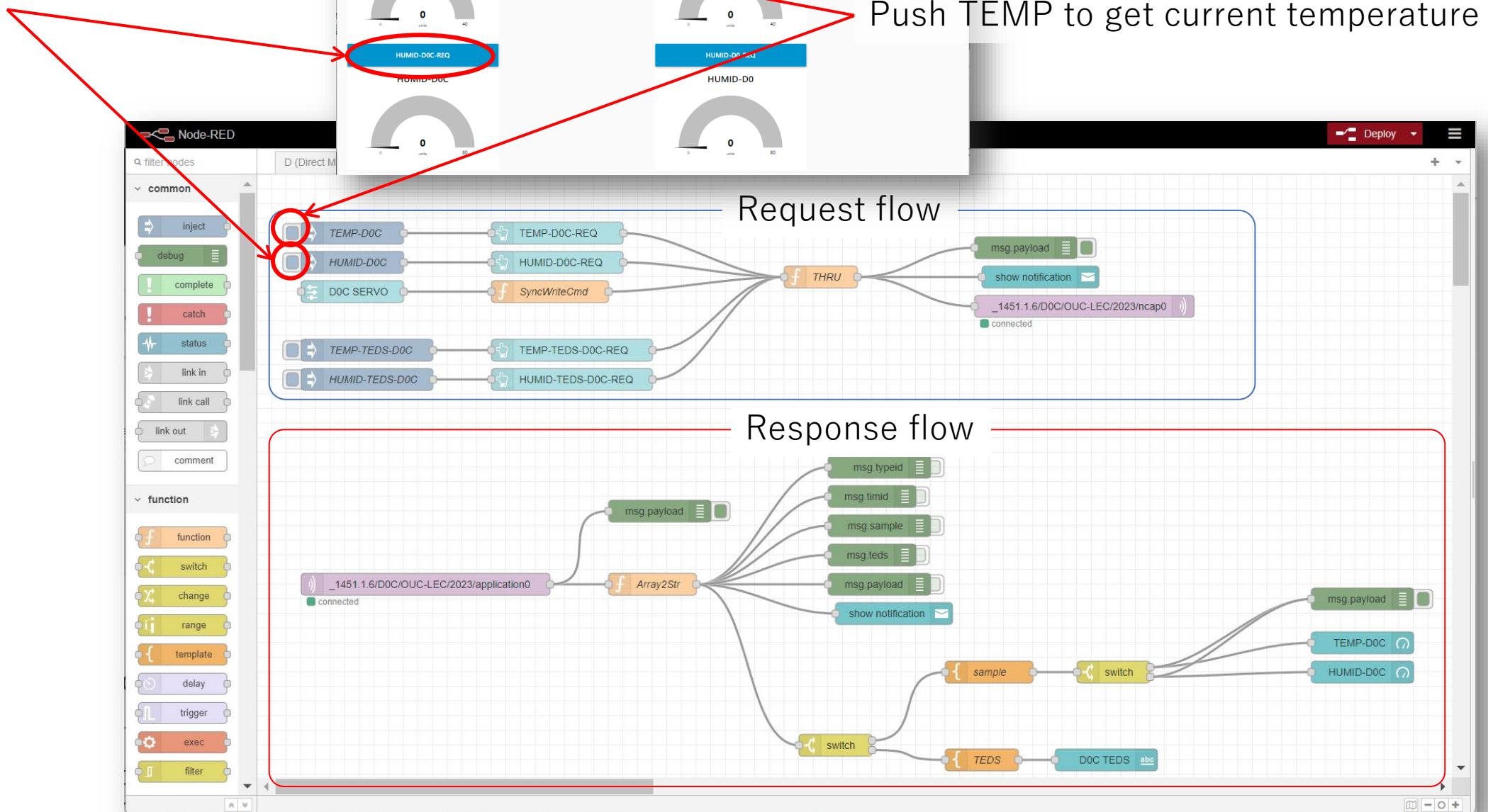
- The dashboard is fully customizable from Node-RED menu.
- Push menu to select a tab.
  - DASHBOARD: D0 dashboard
  - Request-Response: C, D0 dashboard
    - Including sensors and an actuator
  - TEDS: C, D0 TEDS dashboard



Push to select design screen  
or dashboard screen

# C Operation

Push HUMID to get current humidity



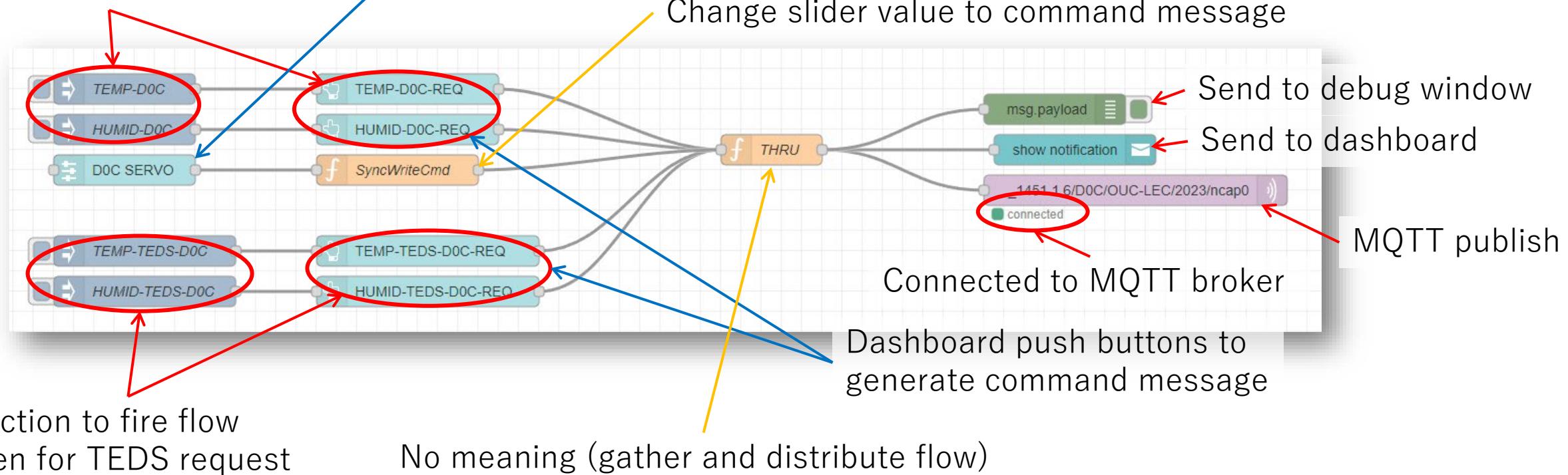
Slide SERVO to change servo angle

Push TEMP to get current temperature

# Request flow for C

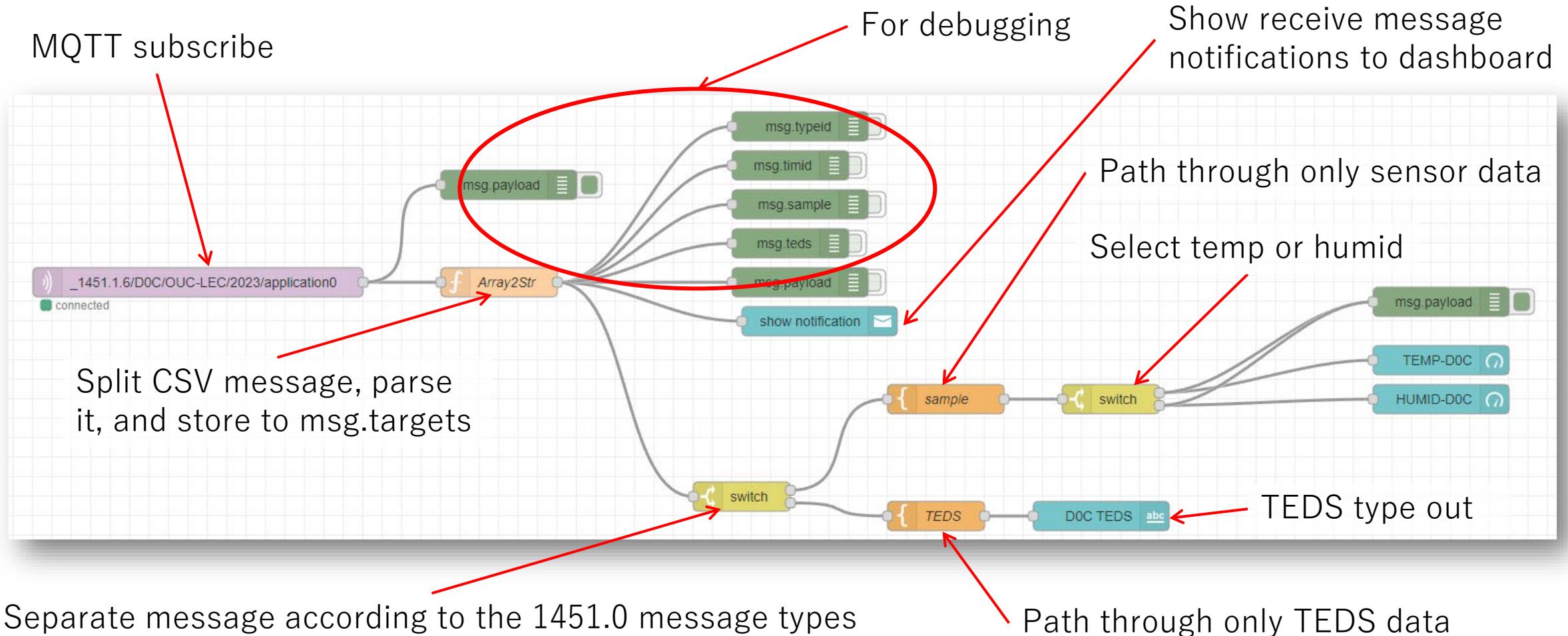
- Click the nodes to show or change the properties of the node.

Injection to fire flow token for sensor request



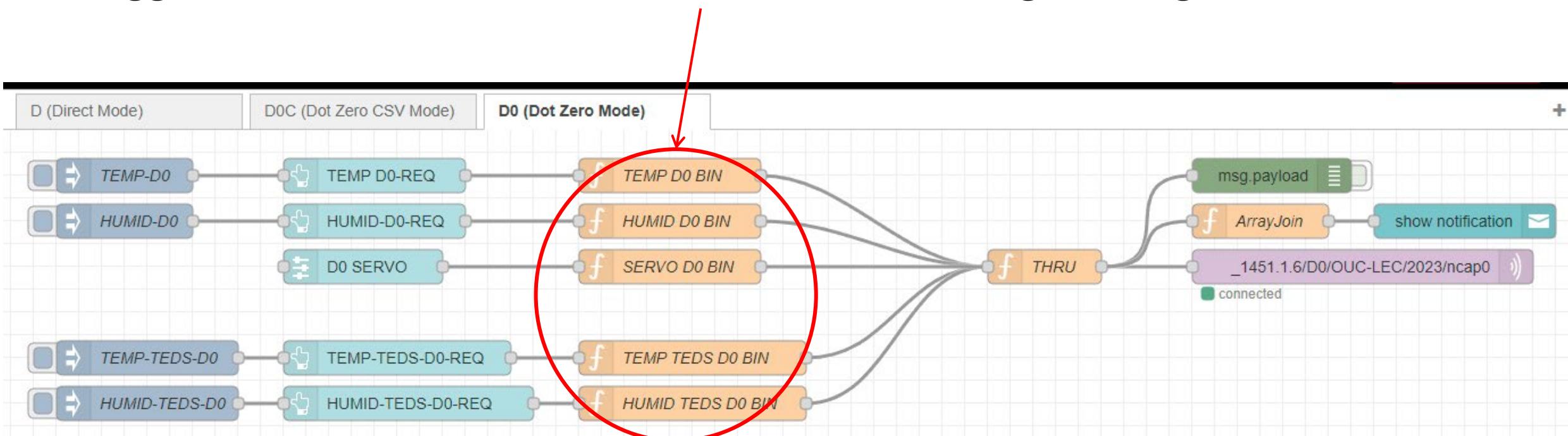
Injection to fire flow token for TEDS request

# Request flow for C

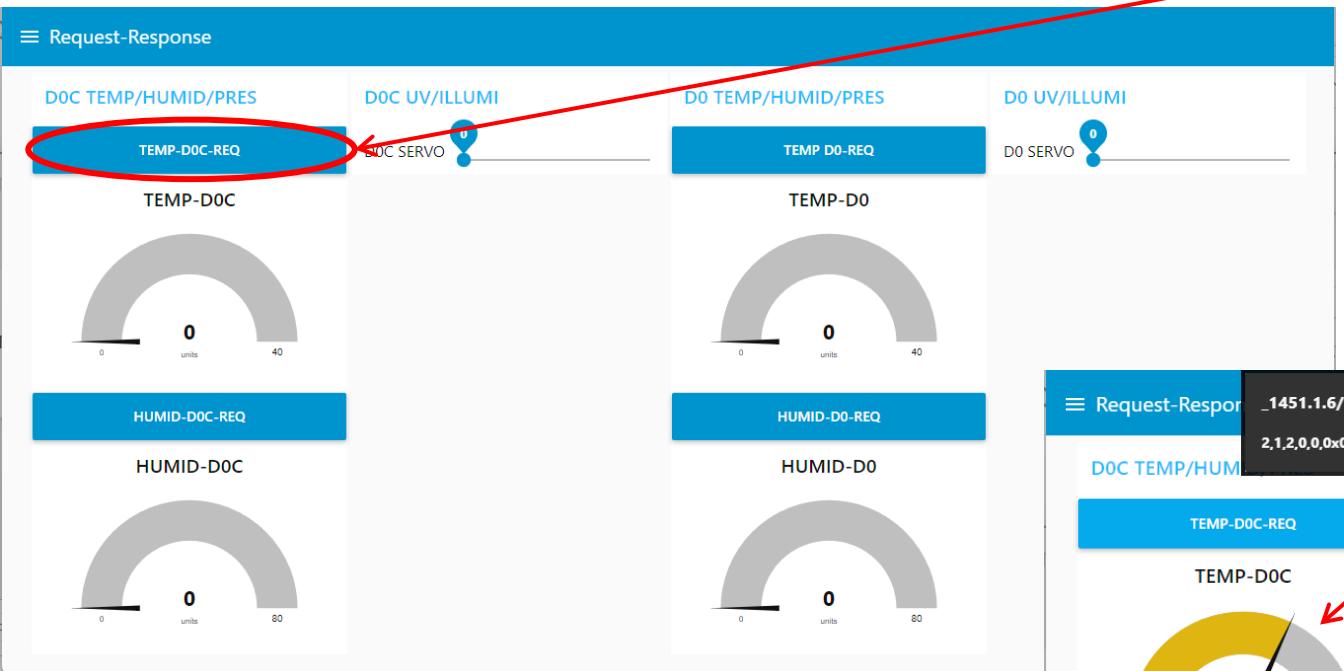


# D0 operation

- It is quite similar to C operations.
- The generated data is 1451.0 native binary message, and it is configured as a binary number array.
- The biggest difference to the C is functions for creating messages.



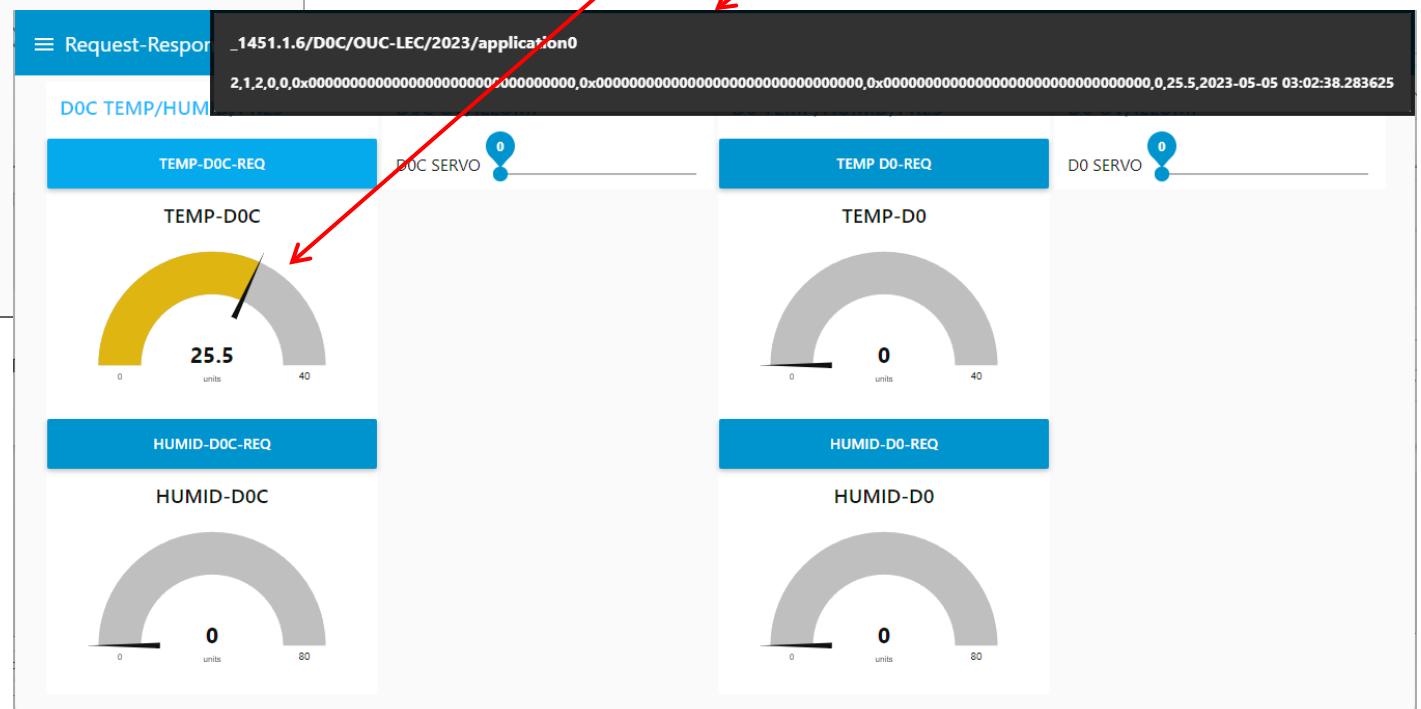
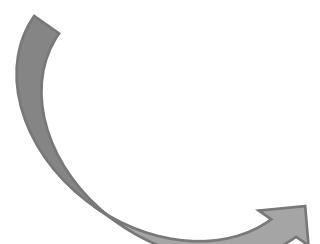
# Request-Response Dashboard



Push temperature request to send read request command message

Receive response message and show the data in the gauge

Show response message in the notification window



# Q: Extend your system

- Q: Extend your NCAP application
  - Notify if the temperature exceeds some thresholds
    - Show message bubble
    - Send e-mail
    - Speech
    - Use SNS / Cloud services
  - Control servo motor when temperature exceeds some thresholds
  - Connect multiple NCAP servers
    - Show all NCAP TIM sensors into one dashboard
    - Calculate average temperature
- Q: Extend your NCAP
  - Show additional sensors or actuators on your dashboard
    - Show values of variable register as gauge and chart
    - Control LED from your dashboard
  - Notify or control additional sensors
  - Implement other messages of 1451.0
    - Announce/Discovery/Abandonment/Departure etc.
    - TEDS operations

# Summary

- Read and write sensor and actuator by using 1451.0 messages
- Accessed NCAP with TIM and Transducers
  - Temperature Sensor
  - Humidity Sensor
  - Servo Motor
- Designed NCAP Application as a visualizer of sensor data
- Accessed TEDS of Transducers
- All was achieved by using Raspberry Pi and Node-RED
  
- NCAP is designed by using Python and is available on GitHub
- 1451.0 and 1451.1.6 provides more complex operations of 1451

# Appendix

- A1: MQTT Broker
- A2 NCAP server
- A3: Node-RED nodes
- A4: Install other nodes
- A5: Queueing Calculations Using Node-RED
- A6: Context (global variables)
- A7: Request-Response examples
- A8: MQTT Version confirmation
- A9: MQTT on FPGA

# A1: MQTT Broker

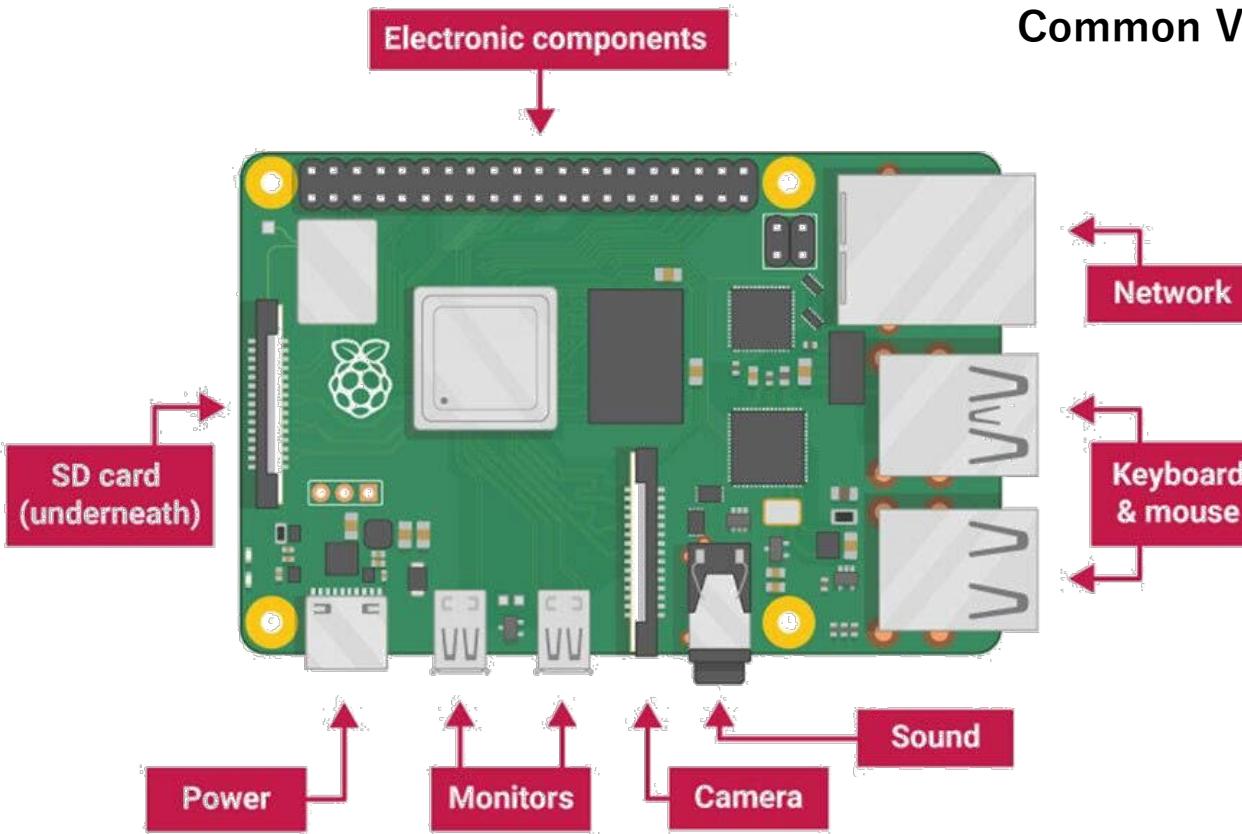
- Install Mosquitto
  - sudo apt install mosquitto
- Open mosquitto when booting
  - systemctl enable mosquitto
- Allow access from other machines
  - sudo echo "listener 1883" >> /etc/mosquitto/mosquitto.conf
  - sudo echo "allow\_anonymous true" >> /etc/mosquitto/mosquitto.conf
  - sudo systemctl restart mosquitto
- Change MQTT broker specification of Node-RED to Localhost:1883
  - NCAP must send the data to the MQTT broker; Namely, it is required to install your own NCAP for publishing sensor data to the broker.

# A2: NCAP server (software installation)

- Download NCAP model
  - git clone <https://github.com/westlab/IEEEP1451.1.6-2023>
- NCAP.py is Located at IEEEP1451.1.6-2023/NCAP.py
- Install required libraries
  - pip3 install paho-mqtt PyYAML temporenc
- Edit config.yml
  - Change the server address and other settings
- Run NCAP server
  - python3 NCAP.py or ./NCAP.py

# A2: NCAP server (Raspberry Pi installation)

- Raspberry-Pi 4B (3 is also Ok)
- DHT11 Temperature and Humidity Sensor
- Servo Motor (Micro Servo Motor SG-90 or compatible servo motors)



Common V+

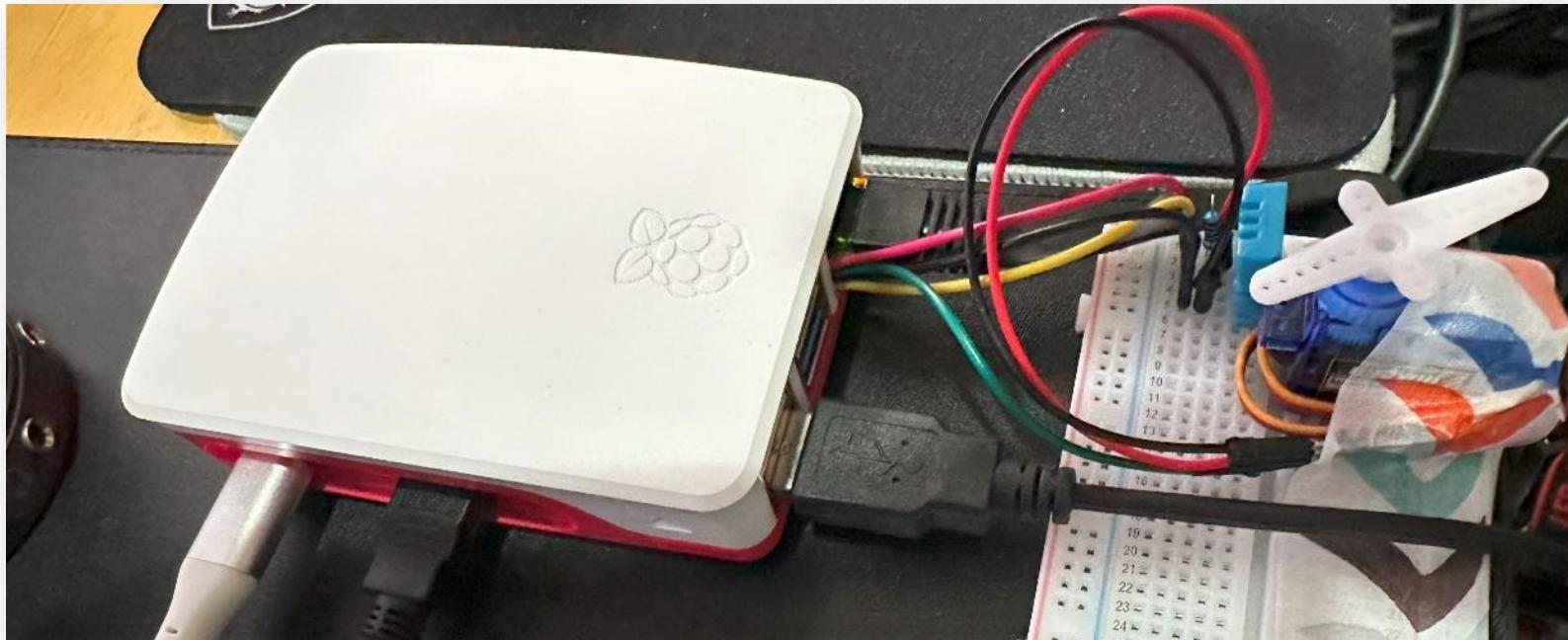
PIN	NAME	NAME	PIN
01	3.3V DC Power	5V DC Power	02
03	GPIO02 (SDA1,I <sup>2</sup> C)	5V DC Power	04
05	GPIO03 (SDL1,I <sup>2</sup> C)	Ground	06
07	GPIO04 (GPCLK0)	GPIO14 (TXD0, UART)	08
09	Ground	GPIO15 (RXD0, UART)	10
11	GPIO17	GPIO18(PWM0)	12
13	GPIO27	Ground	14
15	GPIO22	GPIO23	16
17	3.3V DC Power	GPIO24	18
19	GPIO10 (SP10_MOSI)	Ground	20
21	GPIO09 (SP10_MISO)	GPIO25	22
23	GPIO11 (SP10_CLK)	GPIO08 (SPI0_CE0_N)	24
25	Ground	GPIO07 (SPI0_CE1_N)	26
27	GPIO00 (SDA0, I <sup>2</sup> C)	GPIO07 (SCL0, I <sup>2</sup> C)	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12 (PWM0)	32
33	GPIO13 (PWM1)	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Common GND

Servo

DHT11(1K Pull Up)

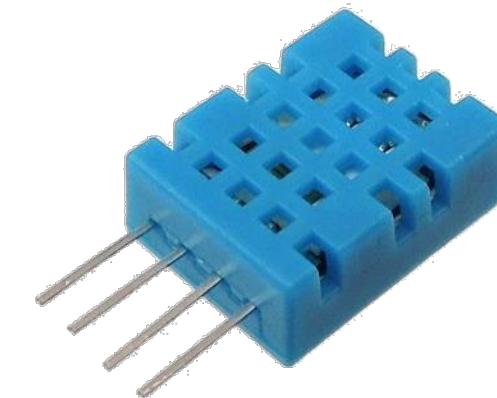
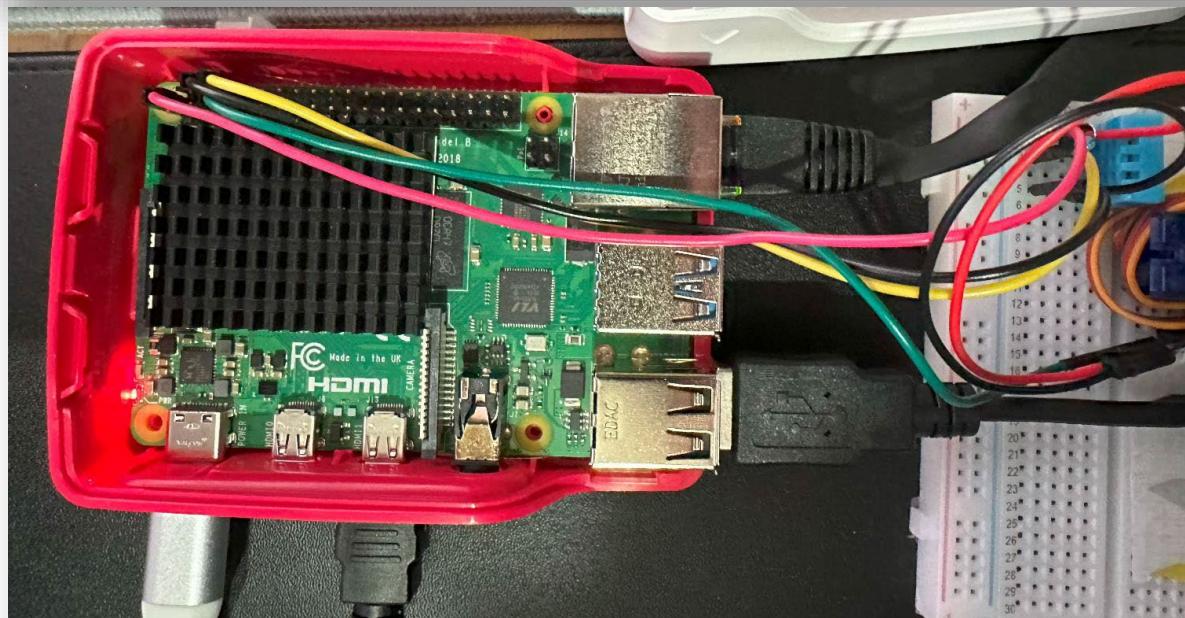
# A2: NCAP/TIM implementation on my desktop



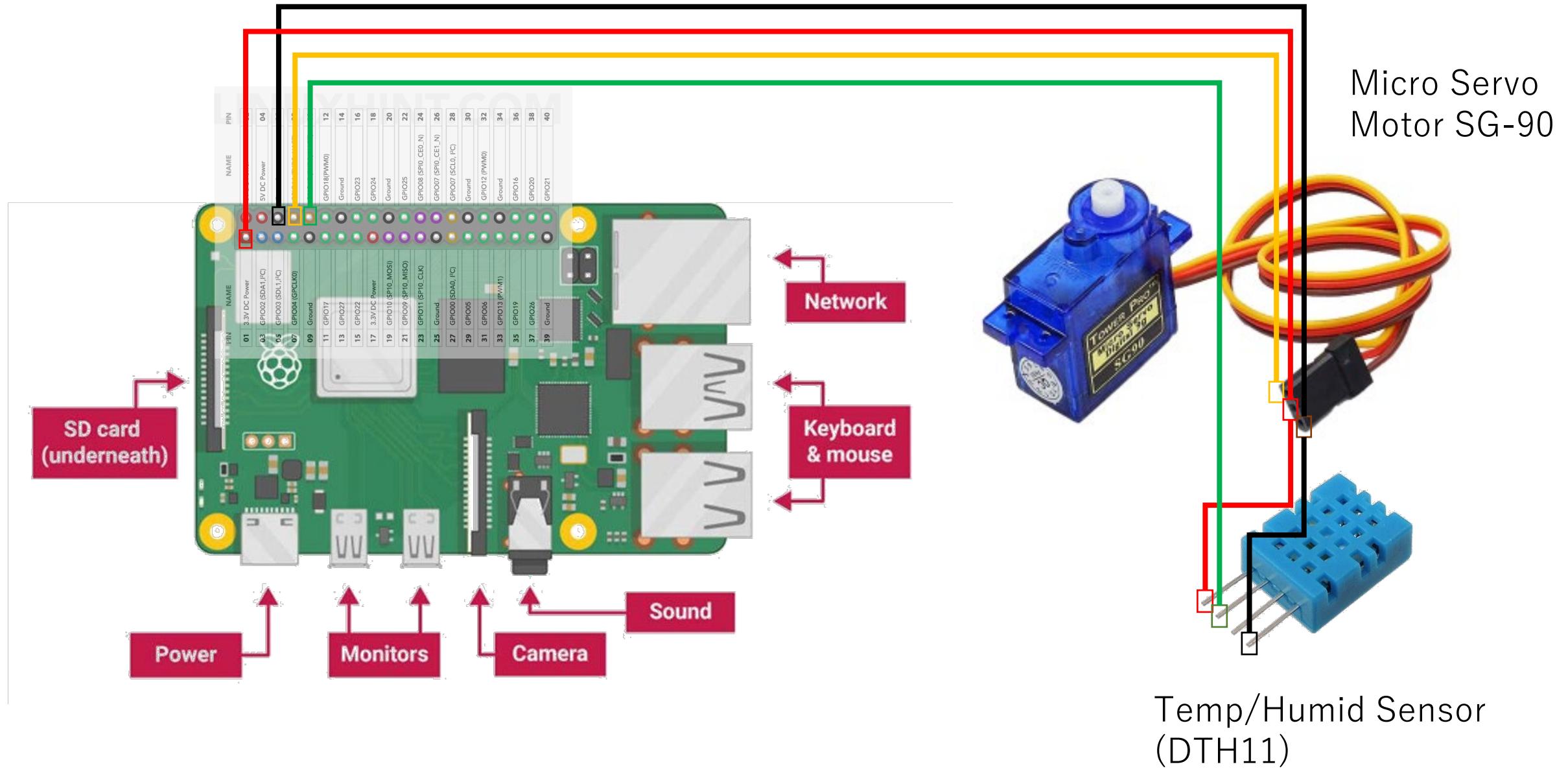
Micro Servo Motor SG-90



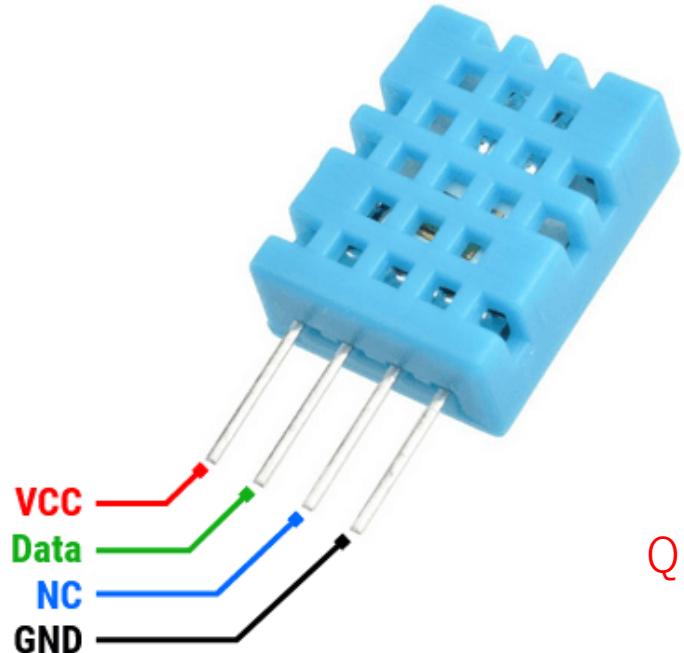
Temp/Humid Sensor (DHT11)



# Wirings



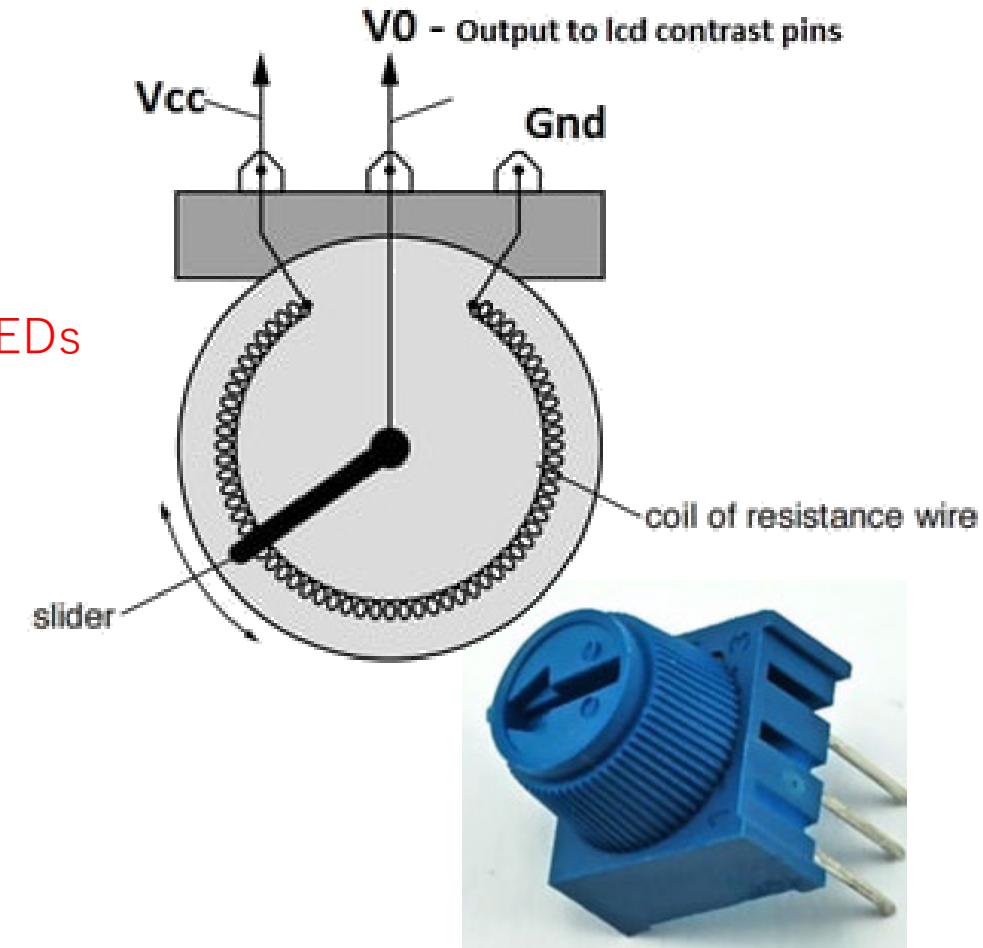
# Sensors / Actuators wirings



Q: Extend your system to read your variable resistor



Q: Extend your system to control LEDs



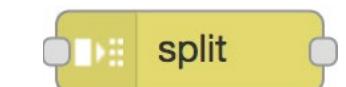
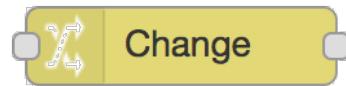
# A3: Node-RED nodes (1)

- Inject nodes can be clicked in the editor to manually initiate a flow, or they can be clicked in the editor to automatically initiate a flow at regular intervals.
  - Outgoing messages can have payload and topic properties
    - The payload can be set to any type:
      - Flow or global context property values
      - String type, numeric type, Boolean type, buffer type, object type
      - Transmission interval (epoch milliseconds)
- Debug node displays the message in the Debug sidebar
  - Contains the time the message was received and information about the Debug node that sent the message
  - Click on the source node ID to see which node it is in the workspace
  - Enable/disable output can be controlled by buttons on the node
  - All messages can be output to the runtime log or a short (32 characters) status text can be displayed under the Debug node
- Function node executes JavaScript code
  - For example, if you want to return the length, var newMsg = {payload: msg.payload.length}; return newMsg;



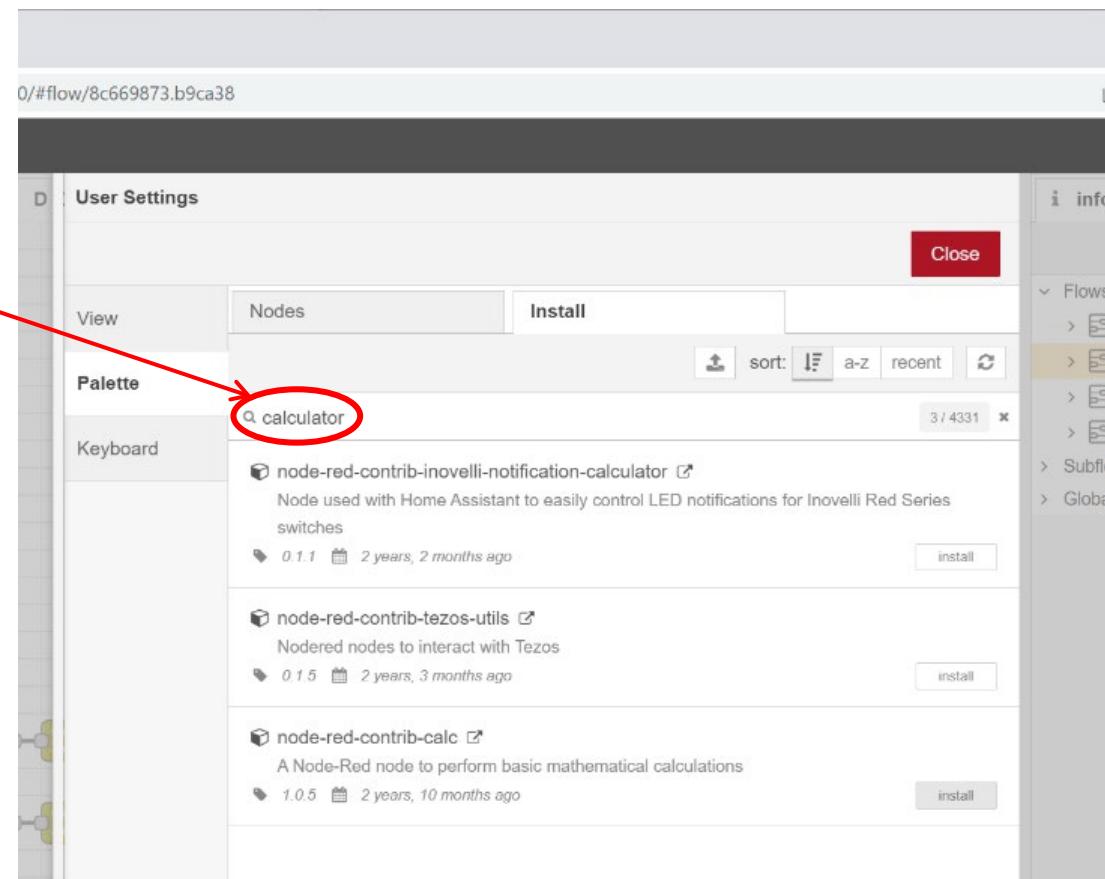
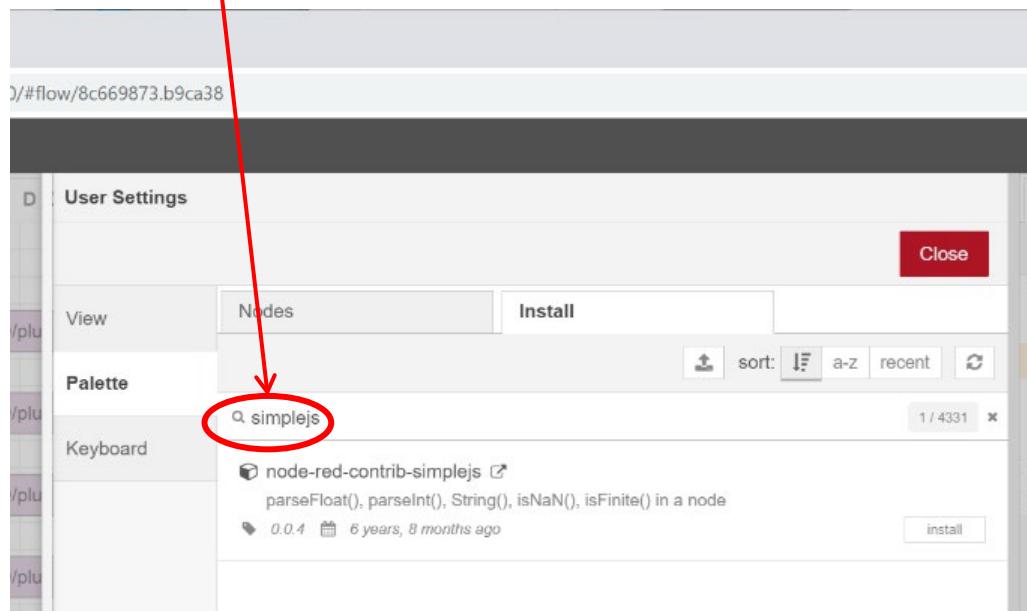
# A3: Nodes in Node-RED (2)

- Change node can change message properties and set context properties
  - Each node can have multiple operations applied to it in turn
    - Value Assignment - Assign properties, various types are available, existing message and context properties are also allowed
    - Value Substitution - find and replace parts of message properties
    - Move or delete values - move, rename, or delete properties; JSONata format can be set for property settings
- Switch node evaluates each message and routes messages to different flows
  - Set properties (message properties or context properties) to be evaluated
  - There are four types of rules
    - Value rule evaluates the set properties
    - Sequence rule can use message sequences such as those generated by Split nodes.
    - JSONata expressions evaluate the entire message and consider a match if it returns a value of true.
    - Others match if none of the aforementioned rules match.
  - It is possible to specify whether to send the message to all matching rules or to stop evaluating when a matching rule is found.
- Template node generates text from the message properties (Mustache notation).
  - For example, This is the payload: {{payload}} !
- Split node splits the flow



# A4: Install other nodes

- There are many kinds of nodes for almost all of IoT and Cloud services, such as ChatGTP.
- Useful examples
  - Install calculator for calculations
  - Install simplejs for `parseFloat()`, `parseInt()`, `String()`, `isNaN()`, `isFinite()` in a node



# A5: Queueing Calculations Using Node-RED



- Node-RED is a bit cumbersome to "calculate between multiple streams".
  - Of course, since they are different streams, the values do not come at the same time
- Node-RED uses a clever way to do this
  - Name the message by attaching a topic to the message
  - Using a join node, configure as follows
  - Specify the number of streams to mix in "after receiving the specified number of message parts".
  - Finally, add in function
  - The name becomes msg.payload.topic

The screenshot shows the Node-RED interface with a function node configuration window open. The node is named 'Name' and has an 'On Message' setup. The code in the script tab is:

```
1 msg.payload = msg.payload.data1+msg.payload.data2+msg.payload.data3;
2 return msg;
```

To the right, a list of messages is shown:

- 2022/5/31 16:23:13 node: 2385016a.82c82e  
d3 : msg.payload : Object  
↳ { d1: 1, d2: 10, d3: 100 }
- 2022/5/31 16:23:13 node: 84530c72.a5645  
d3 : msg.payload : number  
111
- 2022/5/31 16:23:17 node: 2385016a.82c82e  
d3 : msg.payload : Object  
↳ { d1: 2, d2: 20, d3: 200 }
- 2022/5/31 16:23:17 node: 84530c72.a5645  
d3 : msg.payload : number  
222

The screenshot shows the Node-RED interface with a join node configuration window open. The node is set to 'manual' mode, combining each message payload into a key/value object using the message topic as the key. It is configured to send the message after 3 seconds.

**Edit join node**

Properties

Mode: manual

Combine each: msg. payload

to create: a key/value Object

using the value of: msg. topic as the key

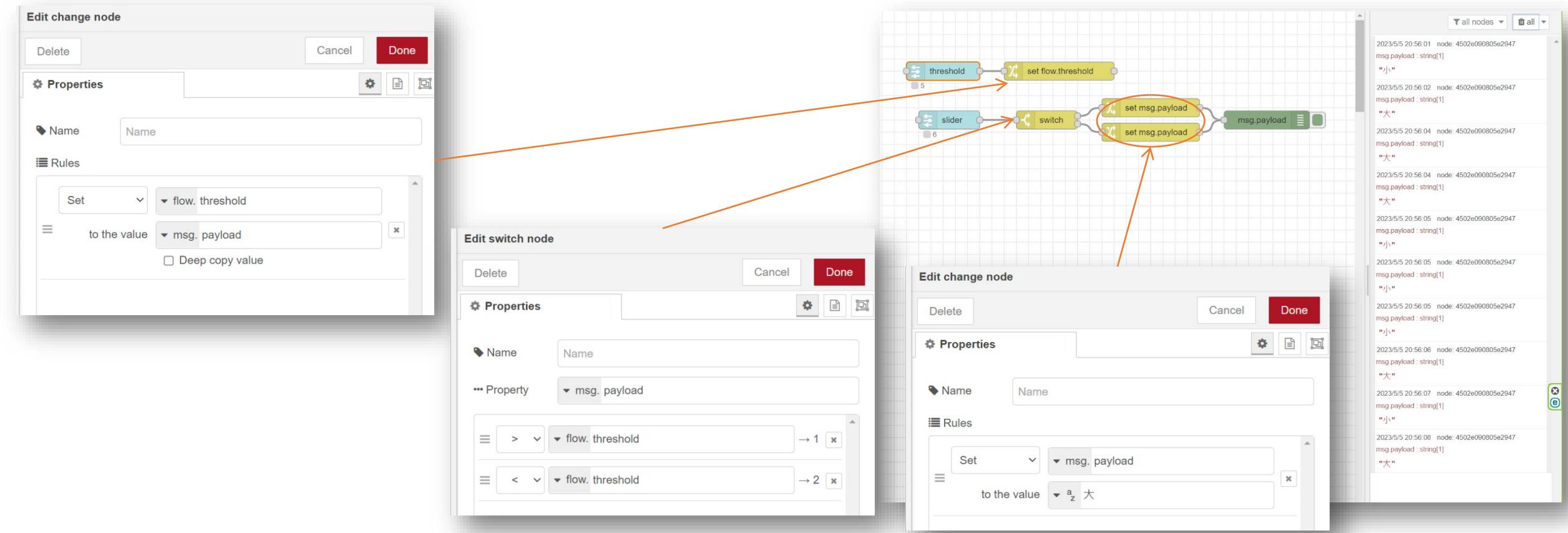
Send the message:

- After a number of message parts: 3
- and every subsequent message.
- After a timeout following the first message: seconds
- After a message with the msg.complete property set

Name: Name

# A6: Context (global variables)

- Node-RED has contexts that are equivalent to global variables.
  - You can assign or reference values to contexts.
- Example: Design a system with two sliders of threshold and input. Output “BIG” if the input value is greater than the threshold slider and “SMALL“, vice versa.



# A7: Request-Response sample codes (Github)

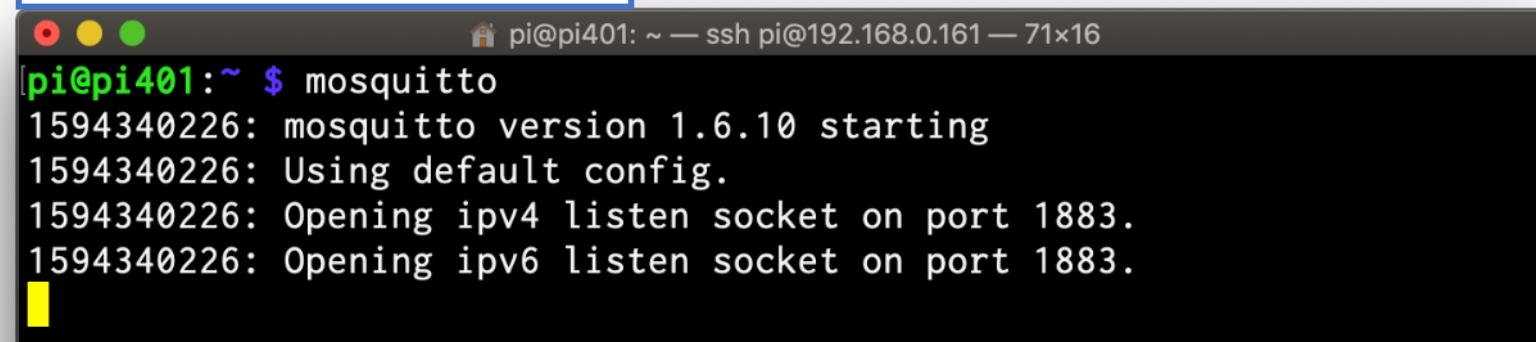
- Sample code 1: ncap.py (NCAP (Response Node))
  - Provide TEDS as a response of request message from applications
- Sample code 2: app.py (Application (Request Node))
  - Receive TEDS from NCAP by publishing a message with response\_topic property
  - Time-out when Application does not receive TEDS soon
    - Application will notice unavailability of the NCAP

## Application (app.py)

```
pi@pi401: ~/MQTT/demo — ssh pi@192.168.0.161 — 56x8
[mqtt] pi@pi401:~/MQTT/demo $ python ./request-response/
requester.py
Connected with flags: 0
Subscribe response topic...
Publish request message with response topic...
Subscribed.
Timeout.
(mqtt) pi@pi401:~/MQTT/demo $
```

# A7: Step1: Run mosquitto (MQTT broker)

MQTT broker (mosquitto)



A terminal window titled "MQTT broker (mosquitto)" showing the output of the "mosquitto" command. The window title bar has three colored circles (red, yellow, green) and the path "pi@pi401: ~ — ssh pi@192.168.0.161 — 71x16". The terminal content shows the broker starting up, using default config, and opening listen sockets on ports 1883 for both IPv4 and IPv6.

```
pi@pi401: ~ $ mosquitto
1594340226: mosquitto version 1.6.10 starting
1594340226: Using default config.
1594340226: Opening ipv4 listen socket on port 1883.
1594340226: Opening ipv6 listen socket on port 1883.
```

MQTT Broker

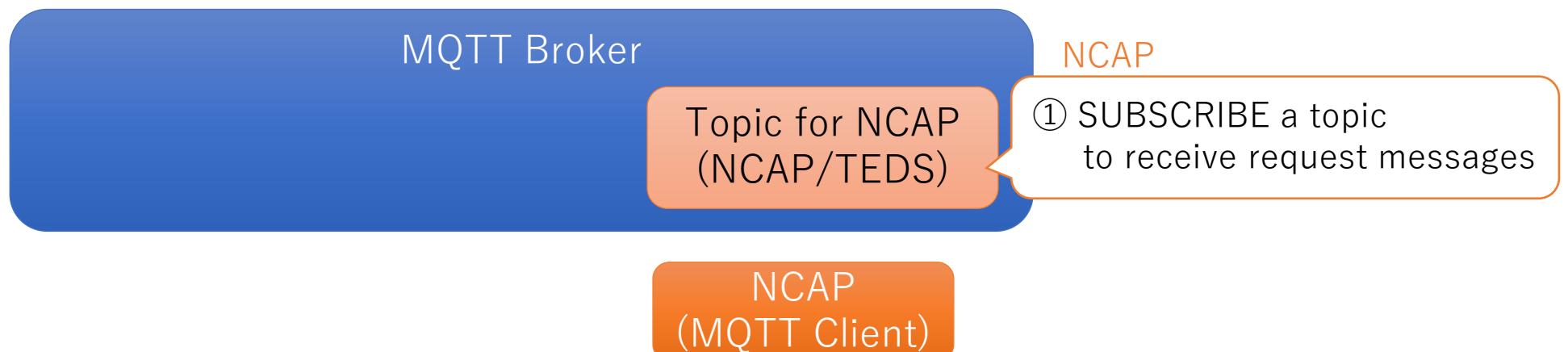
# A7: Step2: Run ncap.py

NCAP (ncap.py) QTT/demo — ssh pi@192.168.0.161 — 46x13

```
[mqtt) pi@pi401:~/MQTT/demo $ python ./request  
-response/ncap.py  
① Connected with flags: 0  
Subscribe topic to receive request messages...  
Subscribed.
```

MQTT broker

```
1594340226: Opening ipv4 listen socket on port 1883.  
1594340226: Opening ipv6 listen socket on port 1883.  
1594340243: New connection from 127.0.0.1 on port 1883.  
1594340243: New client connected from 127.0.0.1 as NCAP (p5, c1, k60).
```



# A7: Step3: Run app.py

## Application (app.py)

```
mo — ssh pi@192.168.0.161 — 56x11
(mqtt) pi@pi401:~/MQTT/demo $ python ./request-response/
app.py
Connected with flags: 0
② Subscribe response topic...
③ Publish request message with response topic...
Subscribed.
⑤ Message received.
Received message: TEDS information from NCAP
Properties in received message: {'dup': 0, 'retain': 0}
Disconnected.
(mqtt) pi@pi401:~/MQTT/demo $
```

## NCAP (ncap.py)

```
TT/demo — ssh pi@192.168.0.161 — 46x13
(mqtt) pi@pi401:~/MQTT/demo $ python ./request-
-response/ncap.py
Connected with flags: 0
Subscribe topic to receive request messages...
Subscribed.
④ Message received.
Properties in received message: {'response_top
ic': ['client/res'], 'dup': 0, 'retain': 0}
Response topic in received message: client/res
Publish response message to response topic...
Published.
```

## Application

② SUBSCRIBE a topic  
to receive TEDS

Topic for Application  
(client/res)

⑤ Receive TEDS

Application  
(MQTT Client)

③ PUBLISH a message  
to NCAP/TEDS with **response\_topic="client/res"**

## MQTT Broker

Topic for NCAP  
(NCAP/TEDS)

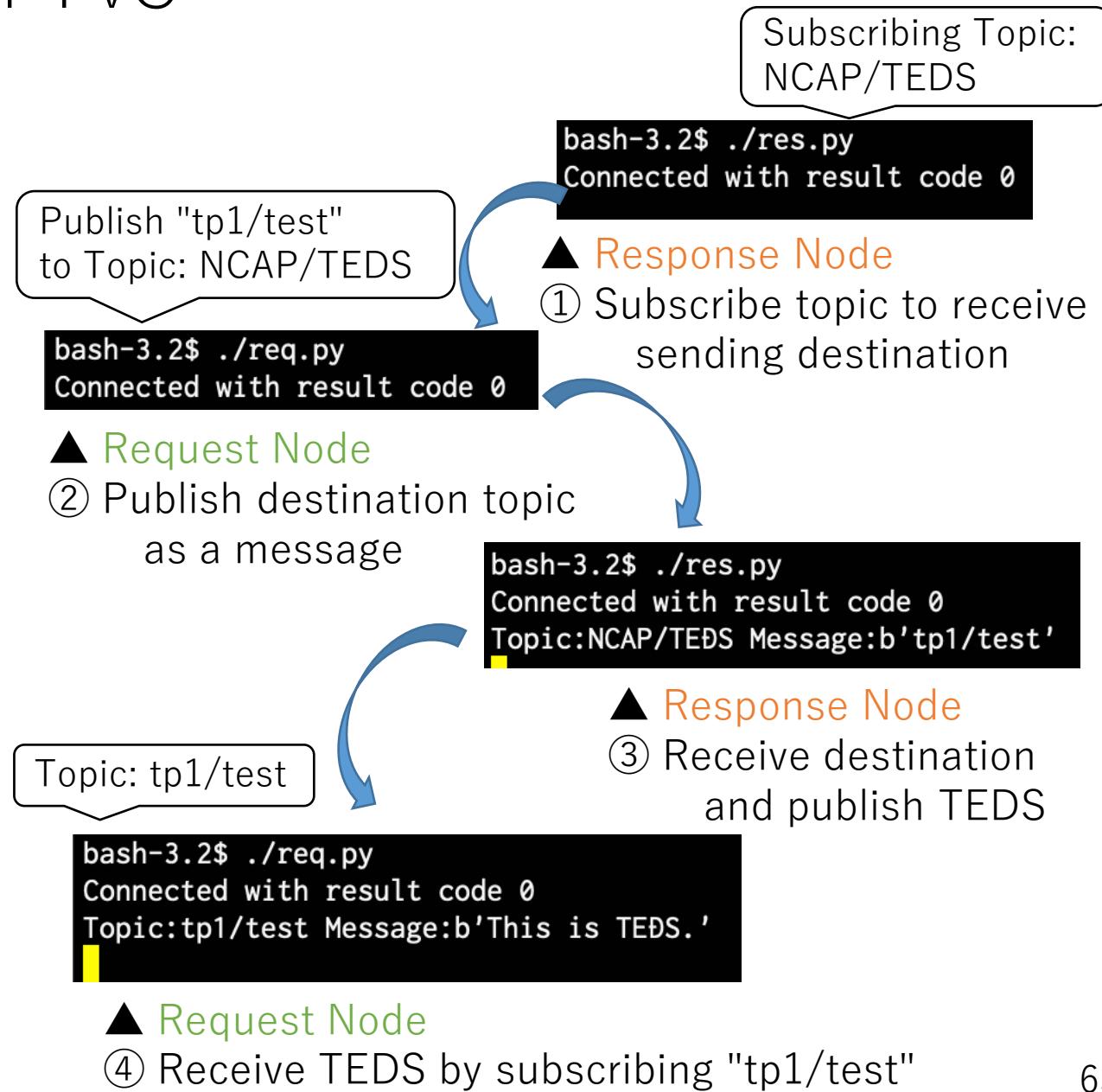
## NCAP

- ④
1. Extract response\_topic property
  2. Set it to publishing destination
  3. PUBLISH TEDS to client/res

# A7: Sample codes for MQTTv3

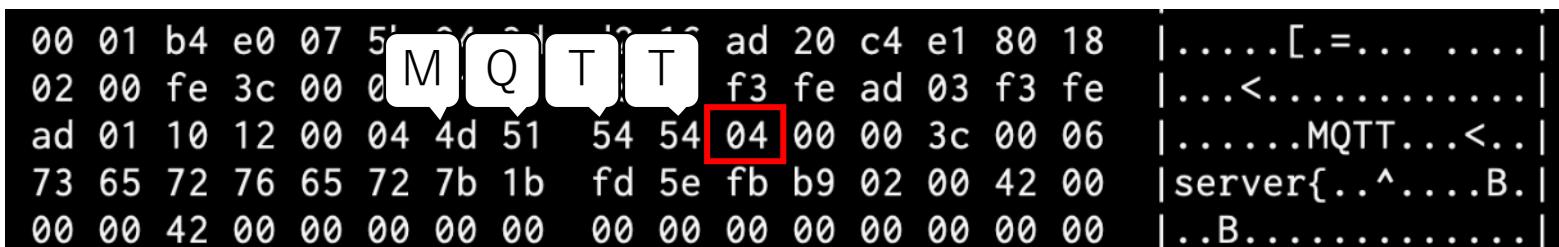
- Software Environment
  - MQTT Broker:
    - mosquitto 1.6.10
  - MQTT Clients:
    - Python3 with paho-mqtt 1.5.0
- Brief description
  - "Request Node" is the Application in the prior code
  - "Response Node" is the NCAP in the prior code
- Difference between MQTTv3 and MQTTv5
  - Request Node sends response\_topic information as a normal message in MQTTv3

Message without property fields



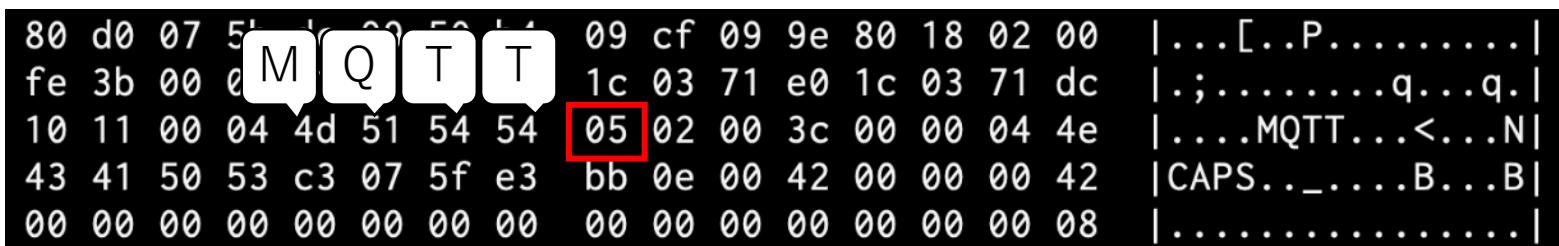
# A8: How to confirm MQTT version?

- Mosquitto and gmqtt automatically downgrade to MQTTv3 when it receives MQTTv3 message
- If MQTTv3
  - "04" appears after "MQTT" in network packet



00	01	b4	e0	07	5f	01	0d	01	0c	ad	20	c4	e1	80	18	.....[.=.... . . . .
02	00	fe	3c	00	00	M	Q	T	T	f3	fe	ad	03	f3	fe	...<..... . . . .
ad	01	10	12	00	04	4d	51	54	54	04	00	00	3c	00	06	.....MQTT...<.. .
73	65	72	76	65	72	7b	1b	fd	5e	fb	b9	02	00	42	00	server{..^....B.
00	00	42	00	00	00	00	00	00	00	00	00	00	00	00	00	..B..... . . . .
00	00	42	00	00	00	00	00	00	00	00	00	00	00	00	00	..B..... . . . .

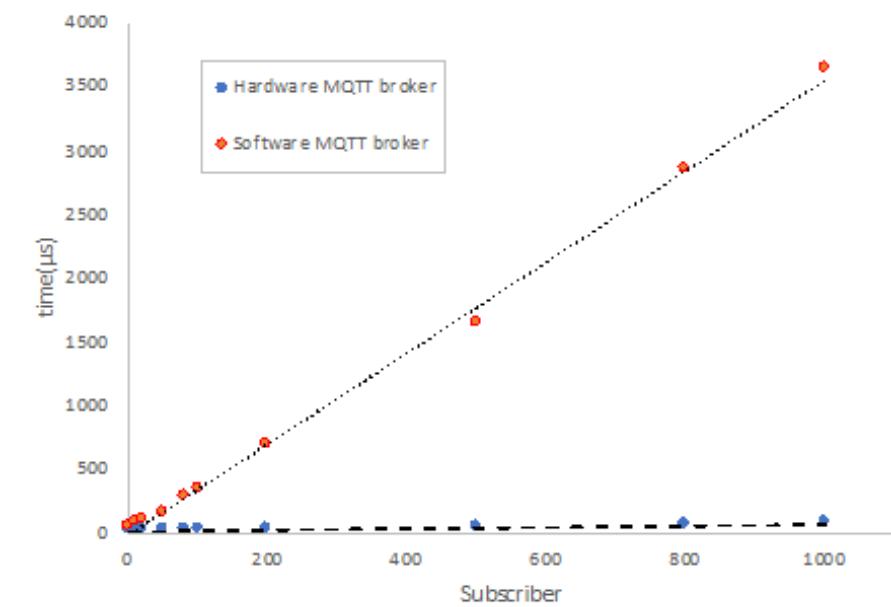
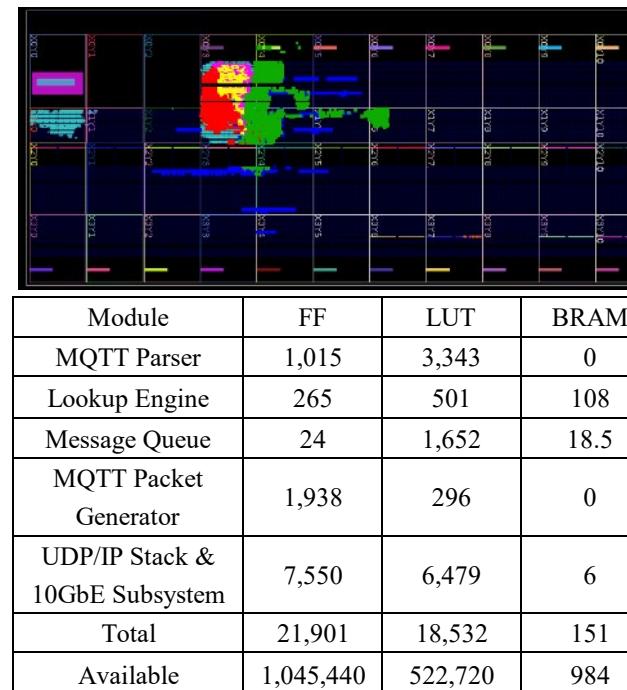
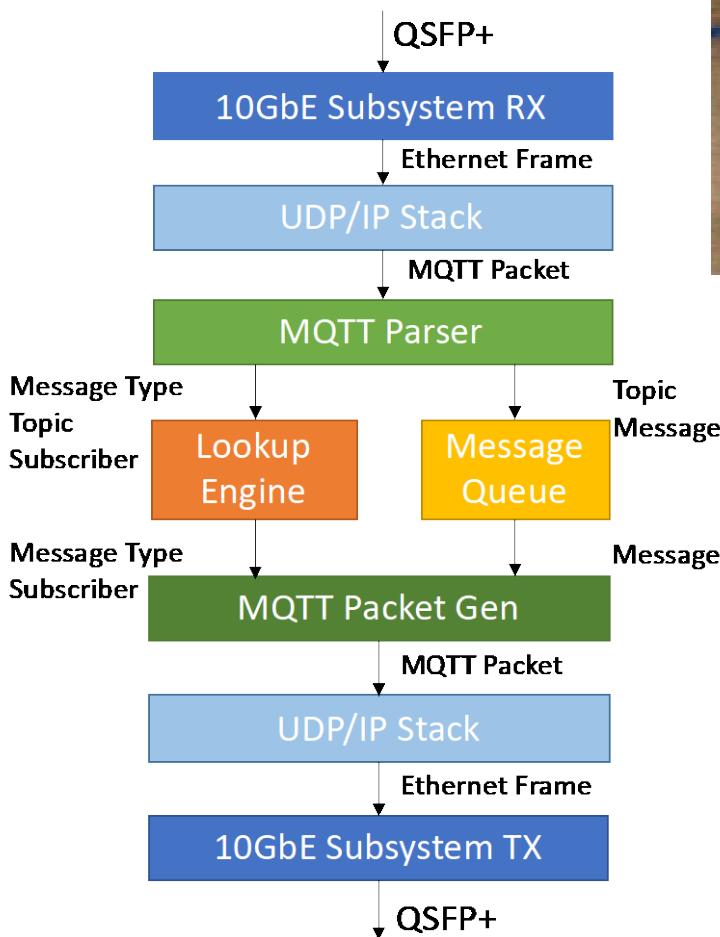
- If MQTTv5
  - "05" appears after "MQTT" in network packet



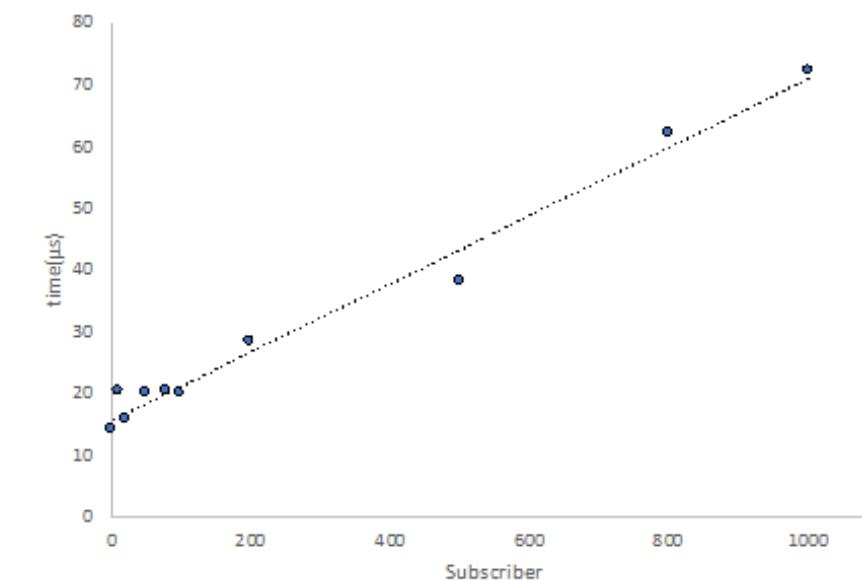
80	d0	07	5f	01	0d	02	5f	01	09	cf	09	9e	80	18	02	00	...[..P..... . . . .
fe	3b	00	00	00	M	Q	T	T	1c	03	71	e0	1c	03	71	dc	.;.....q...q.
10	11	00	04	4d	51	54	54	05	02	00	3c	00	00	04	4e	....MQTT...<...N	
43	41	50	53	c3	07	5f	e3	bb	0e	00	42	00	00	00	42	CAPS._....B...B	
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	08	..... . . . .	

# A9: MQTT on FPGA

- MQTT accelerator as a hardware device implemented on FPGA
- Accelerate x50!



(a) software-implemented broker



(b) hardware-implemented broker