

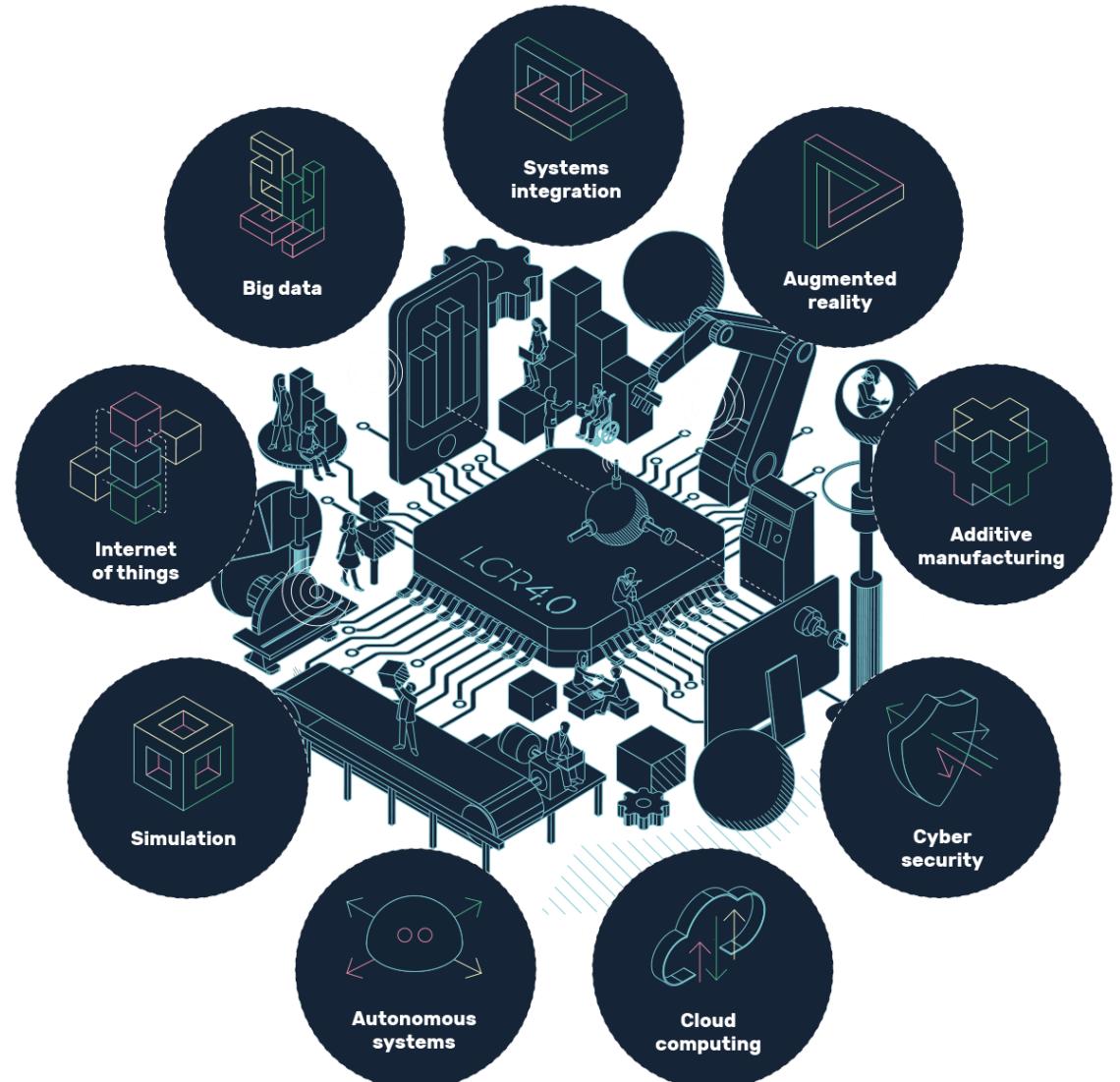
Node-Red: Flow Programming

Simplify application/prototyping deployment

Dr. M. H. M. Ramli

4th INDUSTRIAL REVOLUTION

- Big Data
- IoTs
- Autonomous Systems



Big Data

- Volume: to process high volumes of low-density, unstructured data.
- Velocity: require real-time evaluation and action.
- Variety: Unstructured and semi-structured data types.

IoTs: CONCEPT



Architecture: an ecosystem of objects equipped with sensors that generate data and interconnected via the internet.



Applications: wide range; from Industrial plants, track and classify information so as to reduce human interventions.

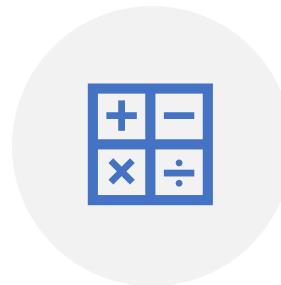


Goal: Monitor, collect, exchange and analyse information efficiently. It aims on connecting physical devices over a network so that they can communicate with each other and take decisions intelligently.

IoTs: BUILDING BLOCKS



Identification and
Sensing: data acquisition.



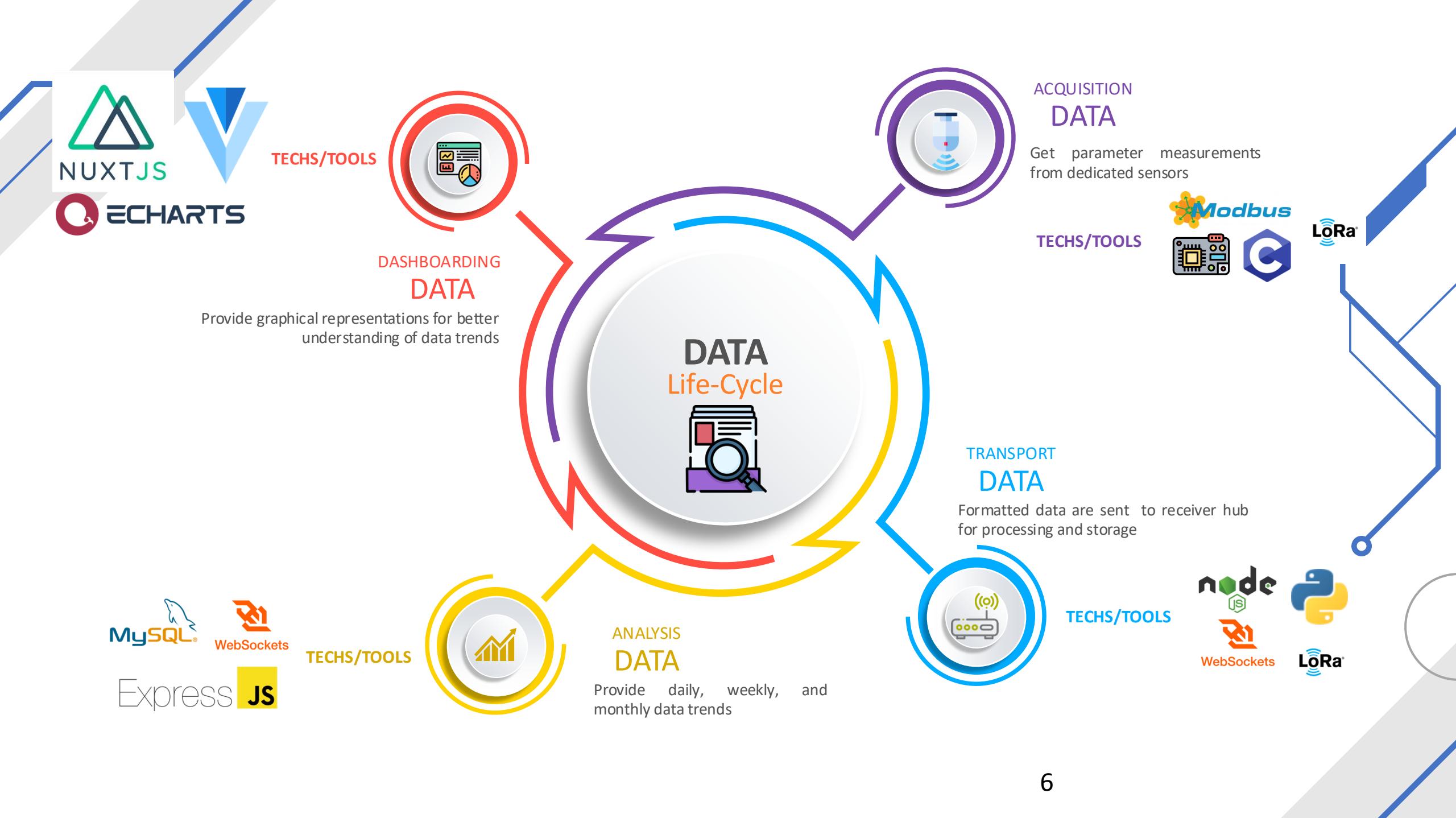
Aggregator
(computations):
Algorithms, arithmetic.



Communication
Channels: TCP, UDP,
Websocket, MQTT, & etc.



Services and Semantics
(logic): Monitoring,
Controls, Predictions.



Section 1: Flow Programming

Overview of Node-red

Accessing Node-Red via mobile devices (phones, tablets)

Node-red programming environment

Fundamental nodes for data processing

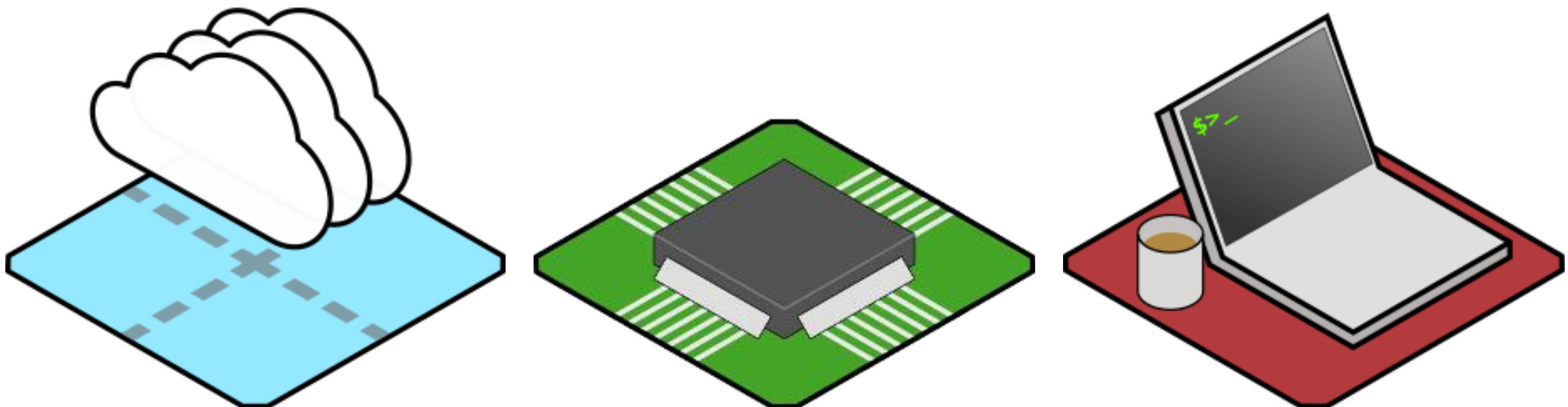
Managing node pallets

Dashboarding nodes

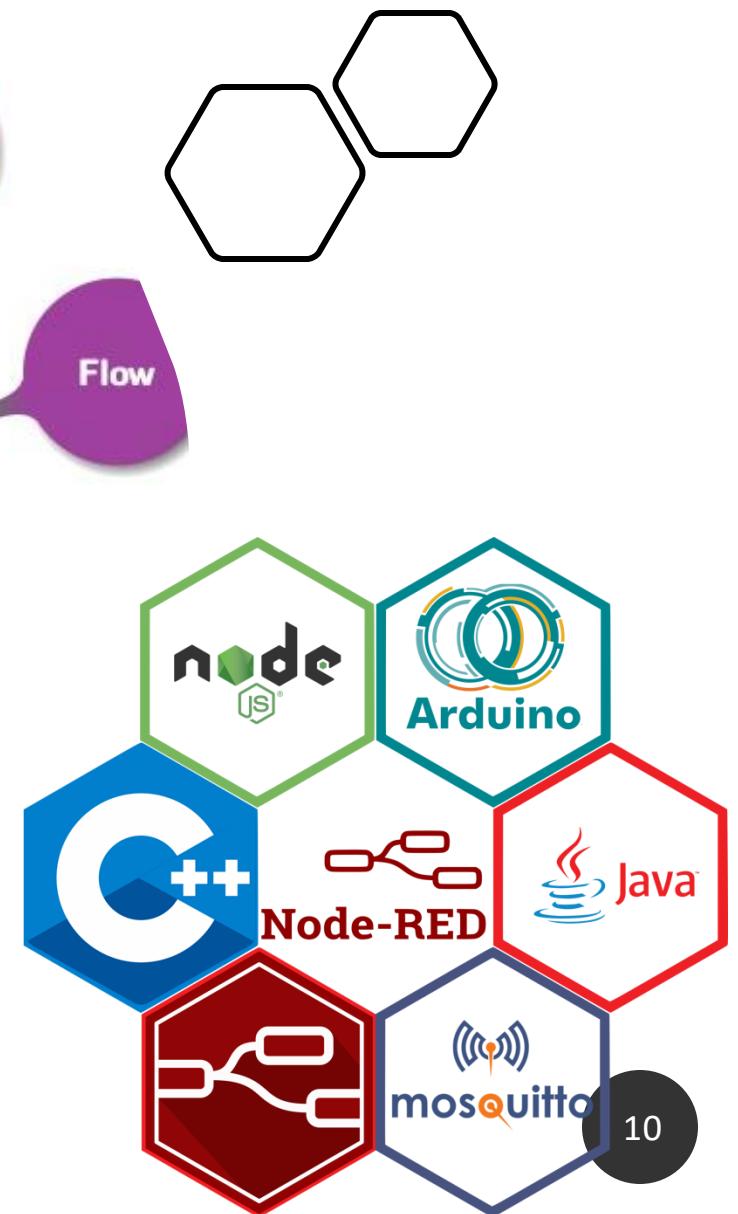
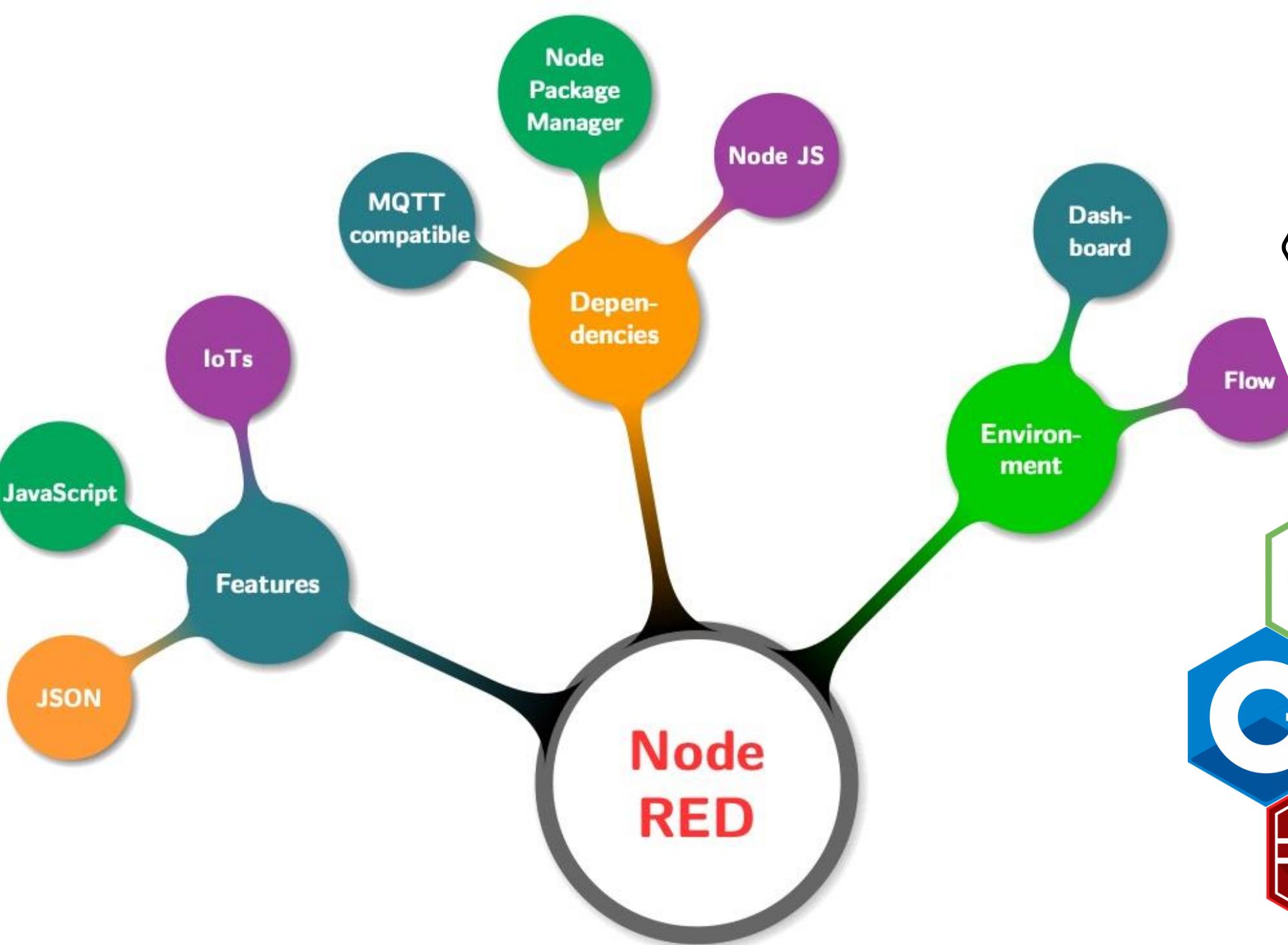
FLOW PROGRAMMING TOOL: NODE-RED

IoT Gateway: Node-Red

- Flow-based programming for the Internet of Things
 - i. connect devices, APIs, and online services.
 - ii. use JavaScript functions.
 - iii. various built-in libraries.



Runs on various platforms: local, dedicated devices, and cloud



Finder Sun 1:44 PM

LASK ENVIRONMENT 4.0 LOCAL ANAESTHESIA SIMULATOR KIT 4.0

GOAL

Get student mastered the anaesthetic administration skill via augmented physical-simulation environment

STRATEGY

Integration of IoT, mechatronics, & IANB

- [1]: Provide evaluation of position accuracy as an injection is made.
- [2]: Monitor pressure applied during an injection
- [3]: Response to the trainee via haptic feedback once inappropriate pressure is applied

Real-time data acquisition is done wirelessly & immediately upload to server for analysis

PRESSURE SENSOR & HAPTIC FEEDBACK

NEEDLE — PRESSURE SENSOR — BATTERY — WIRELESS CONTROLLER — HAPTIC FEEDBACK

Inferior Alveolar Nerve

pi@raspberrypi:~ \$ node-red



Node-red come in apps bundled with RPiOS. To launch node-red, simply type
node-red

```
pi@raspberrypi:~ $ node-red
```

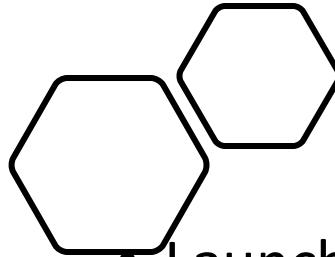
```
18 Jul 09:21:38 - [info] Settings file : /home/pi/.node-red/settings.js
18 Jul 09:21:38 - [info] Context store : 'default' [module=memory]
18 Jul 09:21:38 - [info] User directory : /home/pi/.node-red
18 Jul 09:21:38 - [warn] Projects disabled : editorTheme.projects.enabled=false
18 Jul 09:21:38 - [info] Flows file   : /home/pi/.node-red/flows_raspberrypi.json
18 Jul 09:21:38 - [warn]
```

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

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.

```
18 Jul 09:21:38 - [info] Starting flows
18 Jul 09:21:38 - [info] Started flows
18 Jul 09:21:38 - [info] Server now running at http://127.0.0.1:1880/
18 Jul 09:21:38 - [info] [mqtt-broker:airmode] Connected to broker: mqtt://airmode.live:1883
```

Accessing Node-Red on Browser



- Launch your favorite internet browser,
- Type the IP-address of RPi that connected to your Wi-Fi router on the internet browser (ip-add:1880; normally start with 192.168.1.xxx:1880)

The screenshot shows the Node-RED interface with the following components and annotations:

- Header:** Shows the URL `192.168.8.156:1880/flows/271045-d4039` and a warning icon indicating "Not Secure".
- Left Sidebar:** Contains a search bar for "filter nodes" and two sections: "common" and "function". The "common" section includes nodes like inject, debug, complete, catch, status, link in, link out, and comment. The "function" section includes nodes like function, switch, change, range, template, and delay.
- Center Canvas:** Labeled "Flow 1" and "IP-address to access the Apps". It contains a single node labeled "inject".
- Bottom Status Bar:** Shows the IP address `192.168.8.156:1880/#` and navigation icons.
- Right Sidebar:** Includes a "Deploy" button, a "Nodes" tab (highlighted), and a "Logs" tab. A callout box points to the "Information" tab under "Nodes", which lists Flow, Name, and Status. Another callout box points to the "Description" tab under "Nodes".
- Bottom Right Content:** Text stating "Tabs for configuration, settings, debugging and etc." and instructions for removing selected nodes or links.
- Bottom Center:** Author information: M. H. M. Ramli.
- Bottom Right:** Page number 15.

Node-Red: Basic Nodes

Inject: Versatile configurable input node: inject timestamp, strings, or values.

Debug: Versatile configurable output node: For debug purpose.

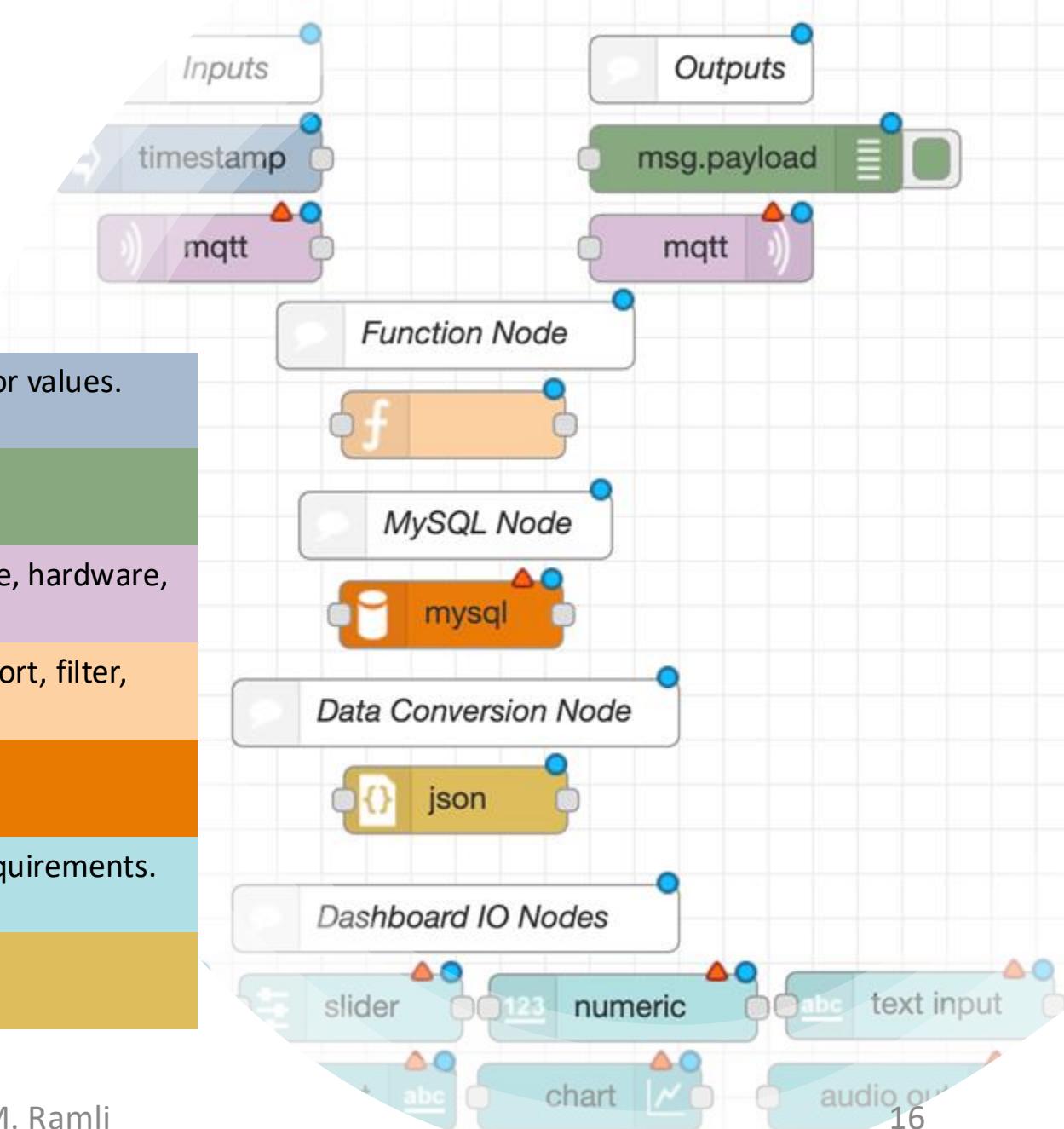
MQTT: Data Broker node: Subscribe/publish data from/to a database, hardware, website, and other API.

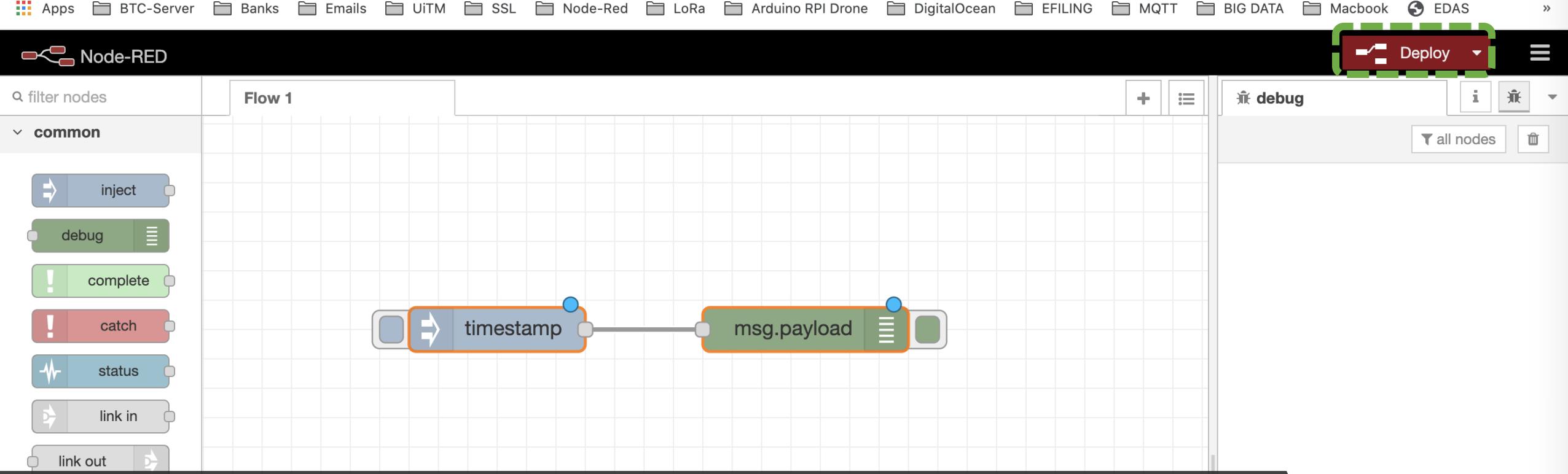
Function: Function node: Use JavaScript to process incoming data (sort, filter, count, statistical analysis, etc.).

MySQL: Manage databases based on Sequence Query Language.

Dash IO: Versatile configurable nodes to suit flexible data display requirements.

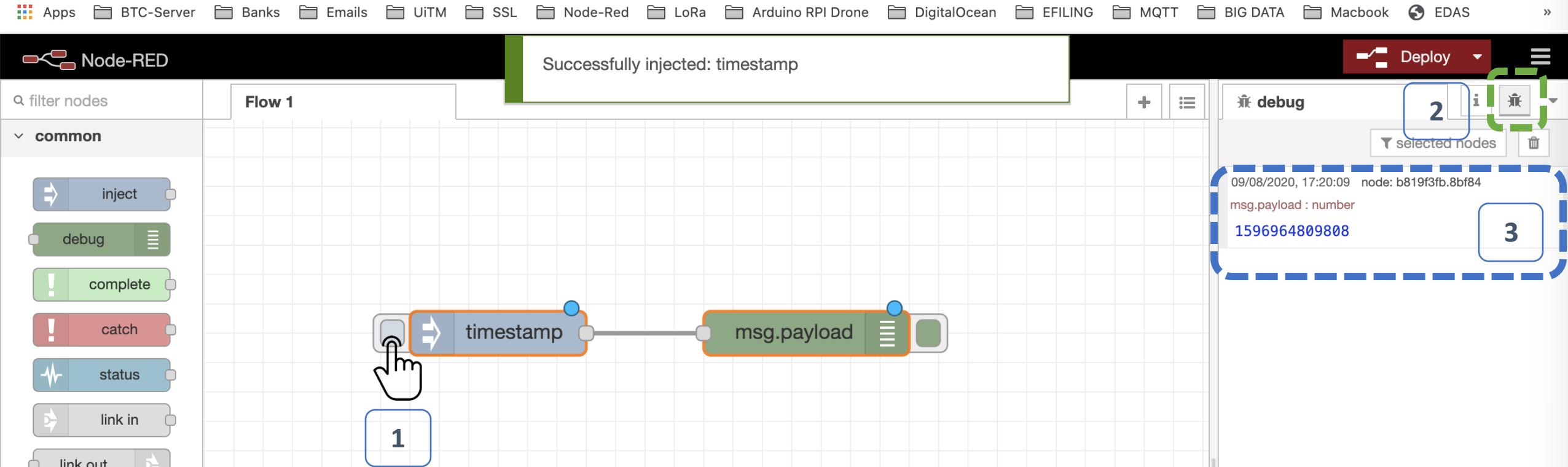
JSON: Converter node to suit flexible data format.





Wiring Nodes

- Click and hold on the end of input node and touch on the edge of output node.
- Once the nodes are connected, click Deploy



Observing the Output

- 1-Click on above-shown input node
- 2-Then, click on debug icon
- 3-The output is shown in the highlighted tab, but what this number stands for?

Node-RED

Deploy

Flow 1

inject

debug

complete

catch

status

link in

link out

comment

function

timestamp

msg.payload

selected nodes

09/08/2020, 17:20:09 node: b819f3fb.8bf84
msg.payload : number
09/08/2020, 17:20:09 [UTC+8]

Observing the Output

- The number stands for timestamp in unix format.
- Click on the number to change it into standard format

M. H. M. Ramli

19

Apps BTC-Server Banks Emails UiTM SSL Node-Red LoRa Arduino RPI Drone DigitalOcean EFILING MQTT BIG DATA Macbook EDAS

Node-RED

Flow 1

Edit inject node

Properties

Payload: timestamp

Topic: flow.global.a_z_string^0_9_number^0_10_buffer^\$ env variable

Repeat: 0.1 seconds, then

Name:

Note: "interval" should be used for "at a specific time". See info box for more.

Deploy

debug

selected nodes

09/08/2020, 17:20:09 node: b819f3fb.8bf84
msg.payload : number
09/08/2020, 17:20:09 [UTC+8]

Changing the Input

- Double click on input node
- A number of input formats is available to choose from
- Select: number

M. H. M. Ramli

20

Node-RED

filter nodes

common

- inject
- debug
- complete
- catch
- status
- link in
- link out

function

- function
- switch
- change
- range
- template

Flow 1

Edit inject node

Properties

Payload: 17

Topic:

Inject once after: 0.1 seconds, then

Repeat: none

Name: 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.

Deploy

selected nodes

Done

Changing the Input

- Insert a number into Payload tab
- Next, click Done
- Then, click Deploy

M. H. M. Ramli

21

Not Secure | 192.168.8.156:1880/#flow/ac271045.d4039

Apps BTC-Server Banks Emails UiTM SSL Node-Red LoRa Arduino RPI Drone DigitalOcean EFILING MQTT BIG DATA Macbook EDAS

Node-RED

Successfully injected: 17

Deploy

filter nodes

common

- inject
- debug
- complete
- catch
- status
- link in
- link out

comment

function

- function
- switch
- change
- range
- template

debug

selected nodes

09/08/2020, 17:46:30 node: b819f3fb.8bf84
msg.payload : number
17

```
graph LR; inject[inject] --> fn1(function{fn}); fn1 --> msgPayload[msg.payload];
```

Observing the new Output

- Click on the input node
- Observe the output in the debug tab
- So what is next?

```
msg.topic = msg.payload;
msg.payload = "My Name is Abu Bakr";
return msg;
```

```
msg.topic = msg.payload;
return msg;
```

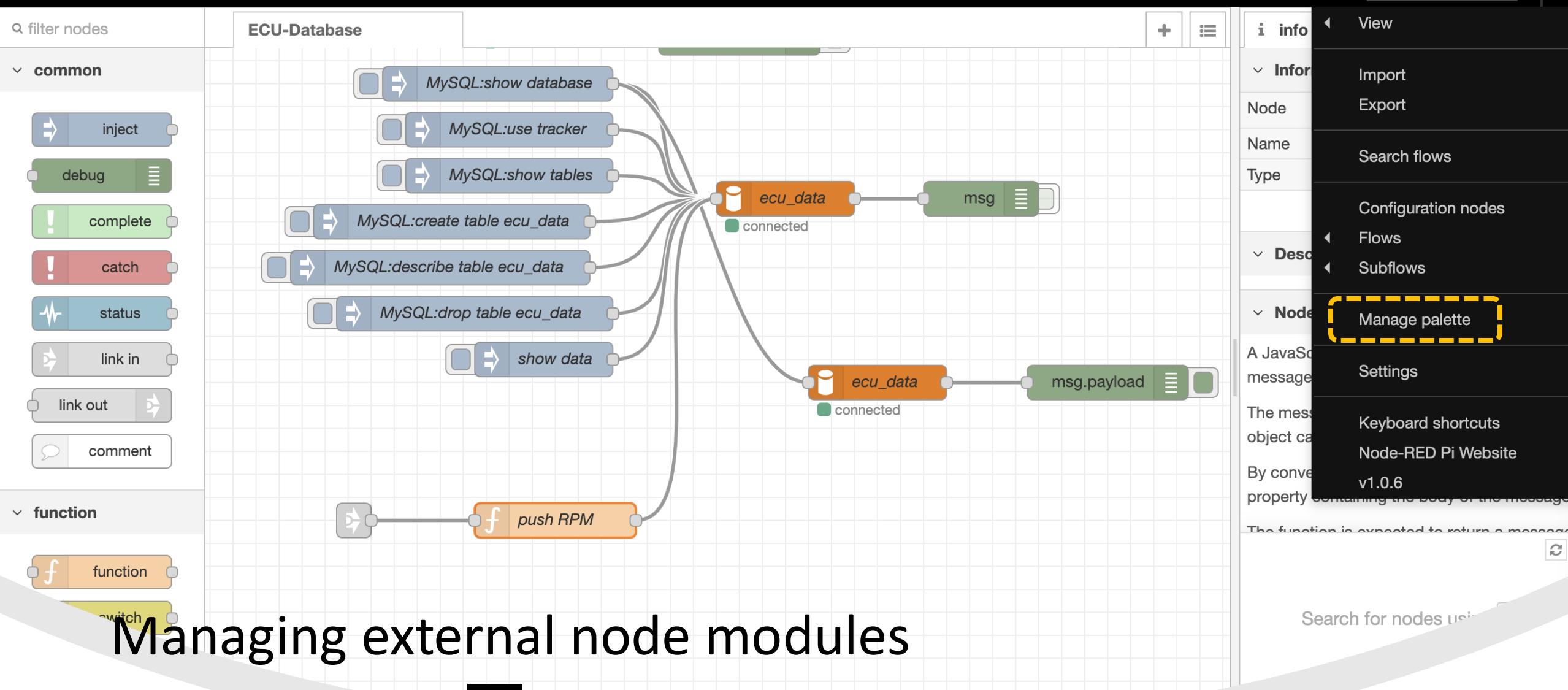


```
msg.topic = msg.payload;
msg.payload = {"Name": "Abu Bakr"};
return msg;
```

```
1564143579140 : msg : Object
  ↳ object
    _msgid: "c5e8e5b9.00d6e8"
    topic: 1564143579140
    payload: 1564143579140
26/07/2019, 20:19:49 node: 3ec6b4a0.81dc9c
1564143589496 : msg : Object
  ↳ object
    _msgid: "58afa25e.371ffc"
    topic: 2019-07-26T12:19:49.496Z
    payload: "My Name is Abu Bakr"
26/07/2019, 20:19:53 node: ad73863f.ccab58
1564143593436 : msg : Object
  ↳ object
    _msgid: "27bdca5a.e491f6"
    topic: 1564143593436
  ↳ payload: object
    Name: "Abu Bakr"
```

Using Function Node

- Notice the difference between Msg Modes, 1: let topic as payload, 2: set payload as string, 3: set payload as object data (refer [Appendix JSON](#) for further details).



Managing external node modules

- Click on the setting tab
- Navigate to Manage palette
- Search for the following palletes

The screenshot shows the Node-RED interface with the following components:

- Left Sidebar (Nodes Palette):** Contains categories like "common", "function", and "ECU-Database".
- User Settings Panel:** Shows the "View" and "Keyboard" sections.
- Central Area:** Displays the "User Settings" palette. A modal window titled "Install" is open, showing the search results for "node-red-dashboard". The first result, "node-red-dashboard", is highlighted with a yellow dashed box. It includes details: version 2.23.0, published 1 week ago, and an "install" button.
- Right Sidebar (Flow Details):** Shows information for the flow "823b27c5.311598" named "ECU-Database". It indicates the flow is Enabled.

Dashboard node module

- Dashboard node offers a user interface for monitoring and controlling of IoT elements
- Data from various sensors can be displayed on the dashboard
- Language: Javascript, HTML

ECU-Database

User Settings

Common

inject

debug

complete

catch

status

link in

link out

comment

function

switch

change

Nodes

Install

View

Keyboard

Palette

node-red-contrib-cpu

A Node Red node to monitor CPU usage

0.0.4 1 year, 10 months ago

installed

info

Information

Flow "823b27c5.311598"

Name ECU-Database

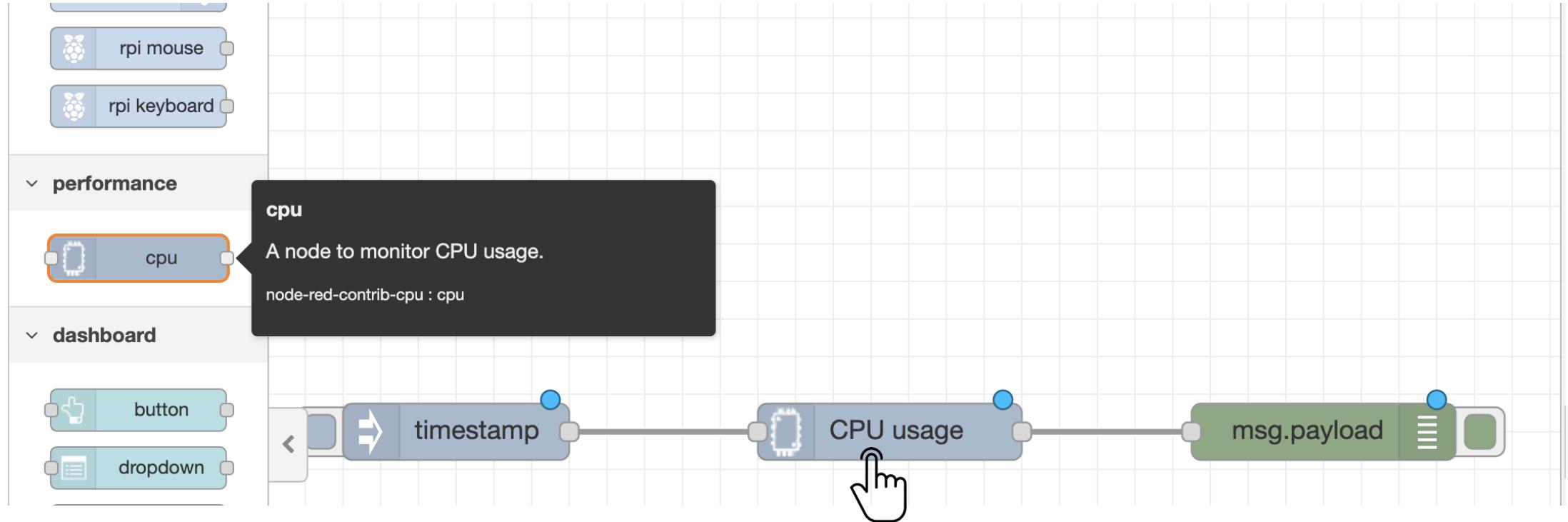
Status Enabled

Description

click and drag on a node port to move all of the attached just

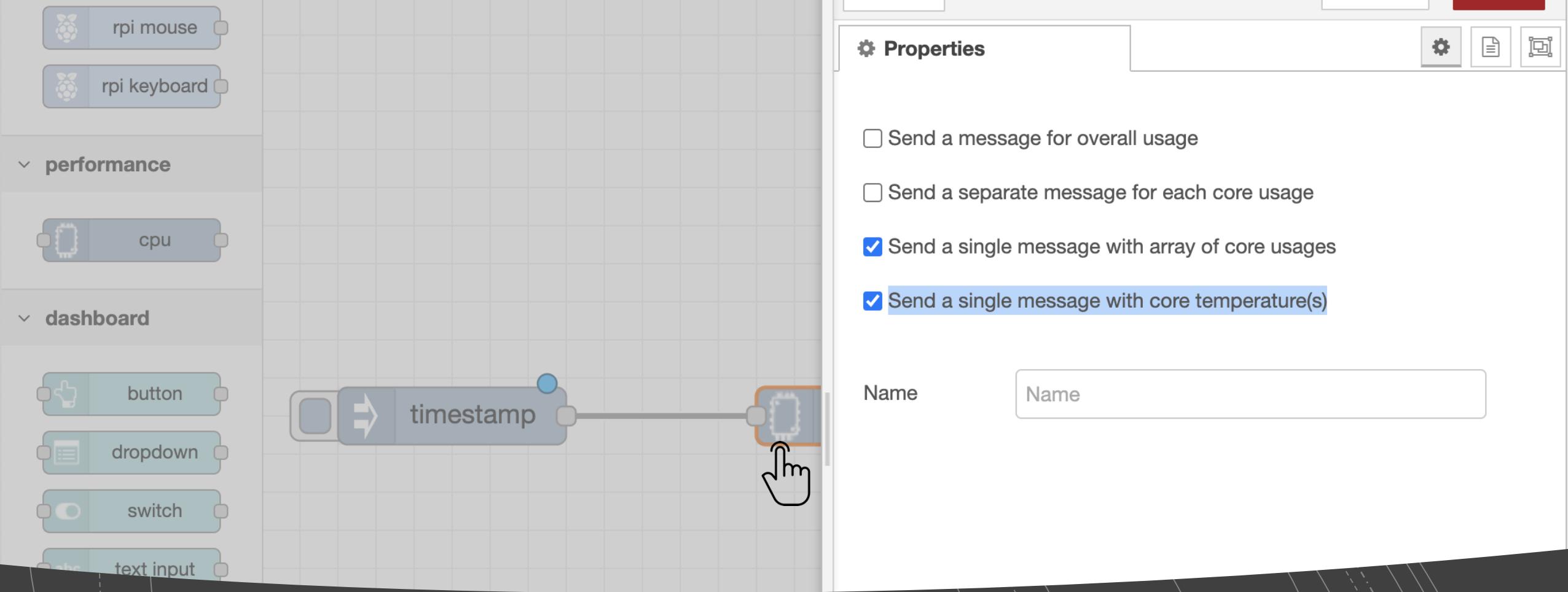
CPU node module

- CPU node offers cpu speed and temperature status of your machine
- Good for monitoring home server
- Easy to use



CPU Monitoring Apps

- Drag cpu, input, and output nodes and wire them.
- Double click on cpu node and tick the following
 - Send a single message with array of core usages
 - Send a single message with core temperature(s)



CPU Monitoring Apps

- Double click on cpu node and tick the following
 - Send a single message with array of core usages
 - Send a single message with core temperature(s)

09/08/2020, 18:32:03 node: e4956e37.4d2cc

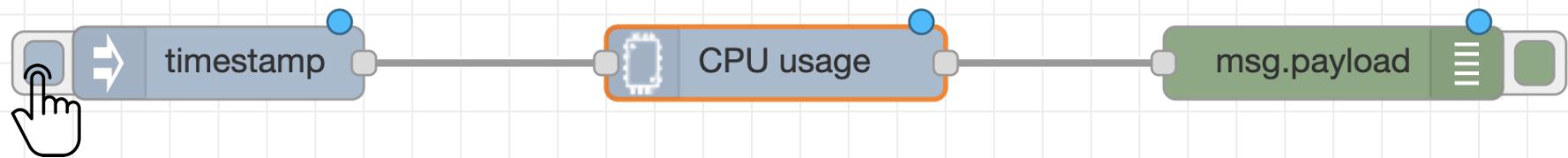
all_cores : msg.payload : array[4]

▶ [object, object, object, object]

09/08/2020, 18:32:03 node: e4956e37.4d2cc

temperature : msg.payload : number

46.738

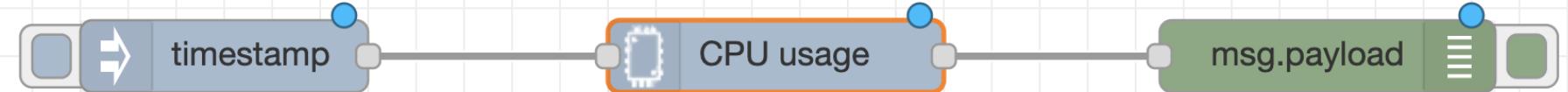


CPU Monitoring Apps: Data Extraction

- Click the input node and observe the output on the debug tab, two types of data, one is object, the other is just a value
- The object provides total cpu cores available on RPi (click on object to see the details)
- The value tells us the current temperature of cpu die.



selected nodes



09/08/2020, 18:32:03 node: e4956e37.4d2cc
all_cores : msg.payload : array[4]

array[4]

0: object

name: "core_1"
usage: 1
model: "ARMv7 Processor rev 3 (v7l)"
speed: 600

1: object

2: object

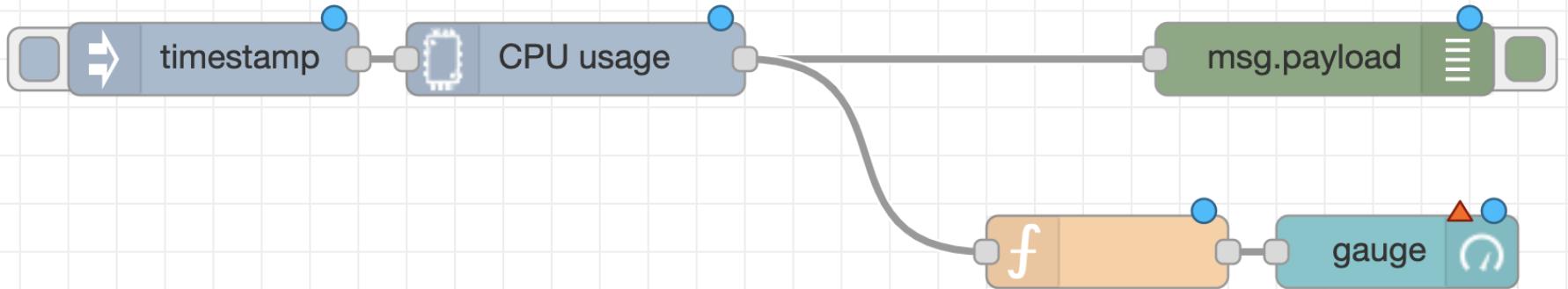
3: object

09/08/2020, 18:32:03 node: e4956e37.4d2cc
temperature : msg.payload : number

46.738

CPU Monitoring Apps: Data Extraction

- In the debug tab, there are two types of data, one is object, the other is just a value
- There are 4 cpu cores in RPi, each current speed = 600MHz
- The current temperature = 46.738C.



selected nodes

09/08/2020, 18:32:03 node: e4956e37.4d2cc

all_cores : msg.payload : array[4]

array[4]

0: object

name: "core_1"

usage: 1

model: "ARMv7 Processor rev 3
(v7l)"

speed: 600

1: object

2: object

3: object

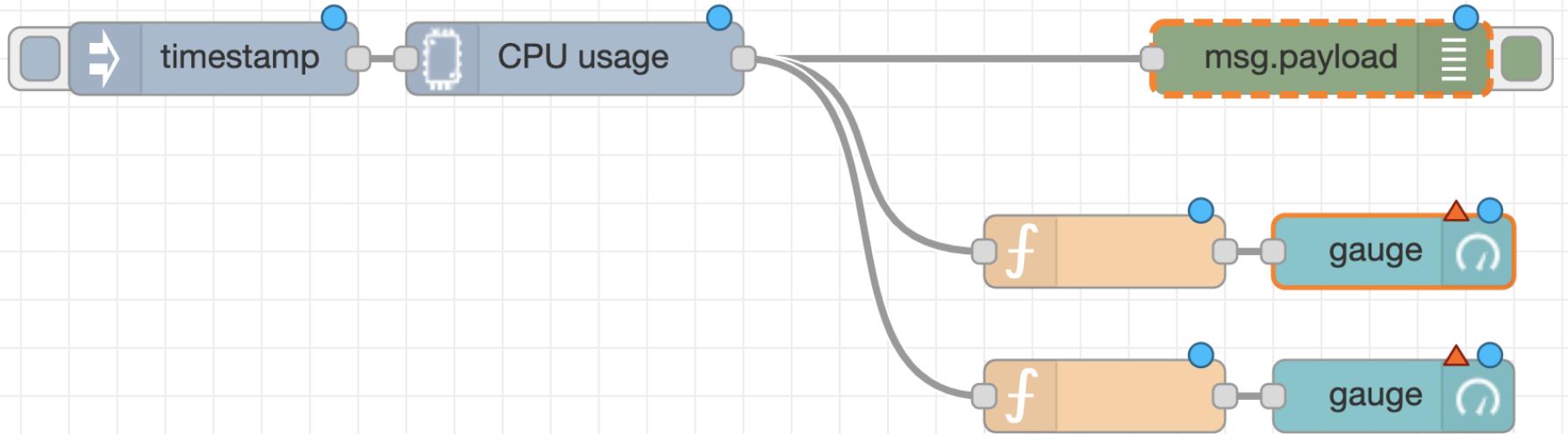
09/08/2020, 18:32:03 node: e4956e37.4d2cc

temperature : msg.payload : number

46.738

CPU Monitoring Apps: Dashboarding

- Display the speed and temperature on dashboard
- Drag a function and gauge nodes, and wire them as per above diagram



09/08/2020, 18:32:03 node: e4956e37.4d2cc

all_cores : msg.payload : array[4]

array[4]

0: object

name: "core_1"

usage: 1

model: "ARMv7 Processor rev 3
(v7l)"

speed: 600

1: object

2: object

3: object



Copy path

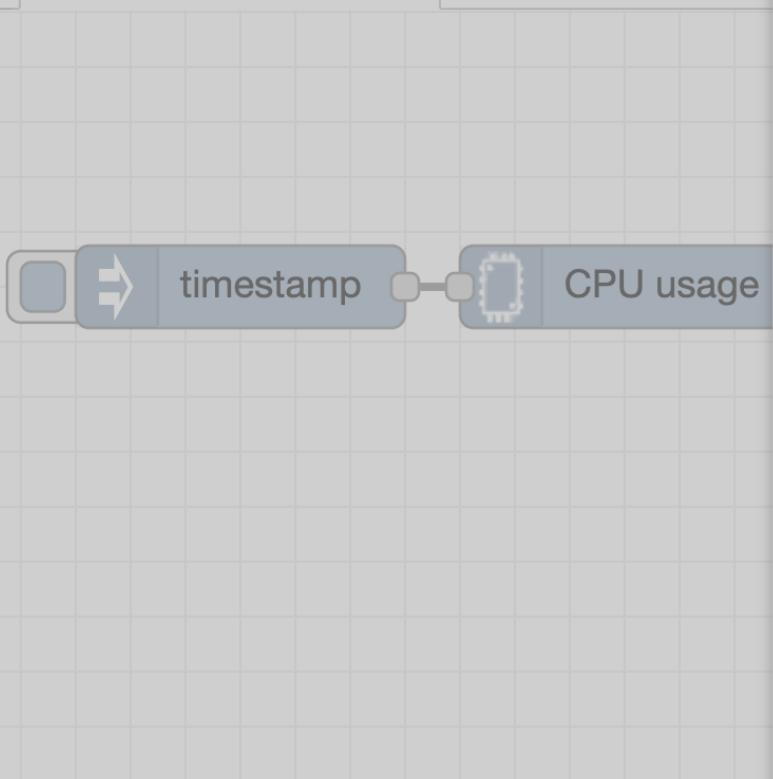
09/08/2020, 18:32:03 node: e4956e37.4d2cc

temperature : msg.payload : number

46.738

CPU Monitoring Apps: Create a function

- Copy the path of object 0: speed
- Next, double-click on the function node



Edit function node

Delete

Cancel

Done

Properties



Name

cpu speed



Function

```
1 msg.payload = msg.payload[0].speed;  
2 return msg;
```



CPU Monitoring Apps: Create a function

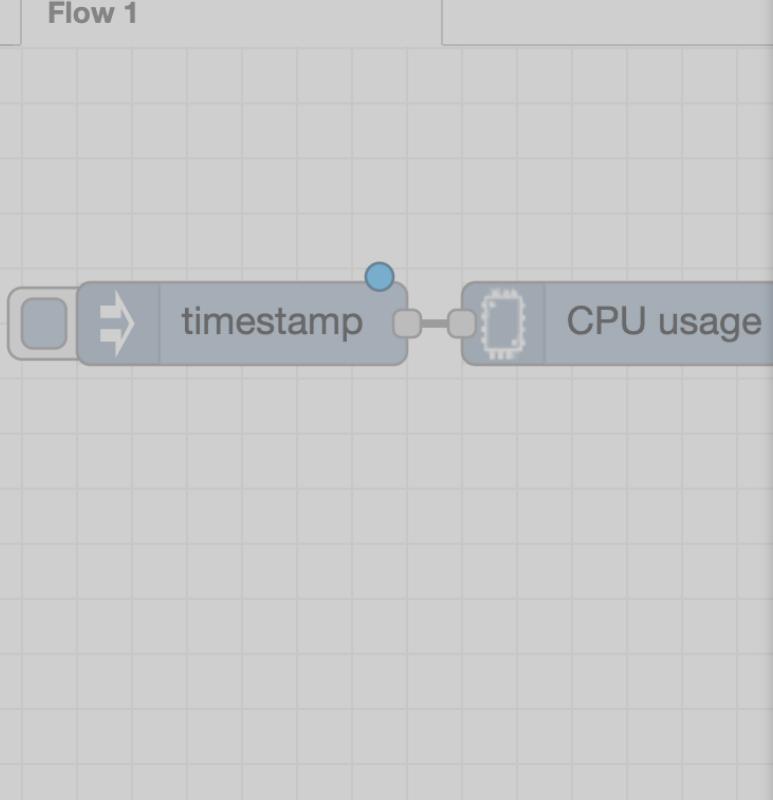
- Set the payload as illustrated in above diagram.
- payload is an output response from a node
- Then, click Done

debug



selected nodes





Edit gauge node

Delete Cancel Done

Properties

Group: Add new ui_group... 

Size: auto

Type: Gauge

Label: gauge

Value format: {{value}}

Units: units

CPU Monitoring Apps: Add a Speed gauge

- Double-click on gauge node
- Since this our first time using the dashboard node, we need to setup the parent and child tabs: double-click on the pencil icon as shown above

Colour gradient

Name

debug

selected nodes

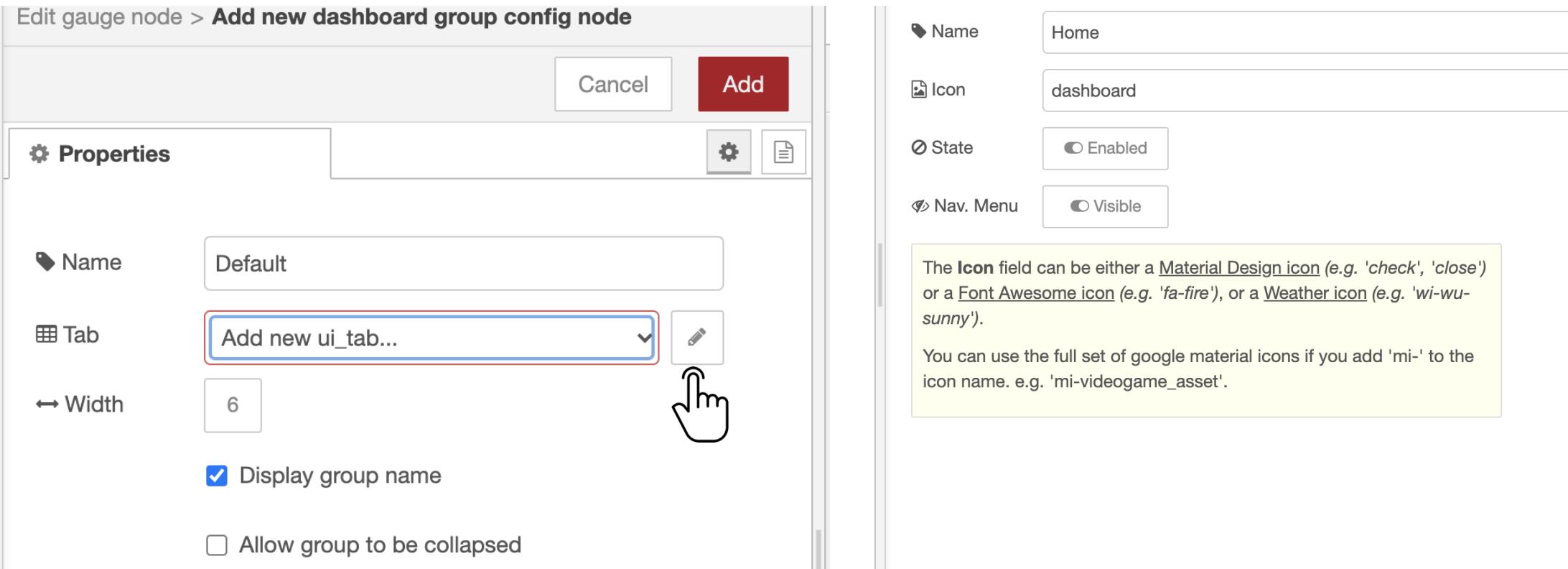
```

09/08/2020, 18:32:03 node: e4956e37.4d2cc
all_cores : msg.payload : array[4]

array[4]
  0: object
    name: "core_1"
    usage: 1
    model: "ARMv7 Processor rev 3
(v7l)"
    speed: 600
  1: object
  2: object
  3: object

09/08/2020, 18:32:03 node: e4956e37.4d2cc
temperature : msg.payload : number
46.738

```



CPU Monitoring Apps: Add a Speed gauge

- Click on pencil icon on the Tab, and a display on the right will pop-up
- For this in session, just use the **Home** name,
- And click **Add**

Cancel

Add

Properties**Name**

cpu monit

Tab

Home

**Width**

6

 Display group name

09/08/2020, 18:32:03 node: e4956e37.4d2cc

all_cores : msg.payload : array[4]

array[4]

0: object

name: "core_1"

usage: 1

model: "ARMv7 Processor rev 3
(v7l)"

speed: 600

1: object

2: object

3: object

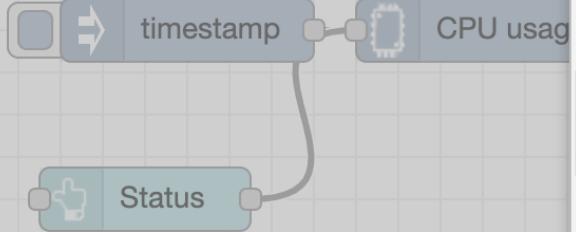
09/08/2020, 18:32:03 node: e4956e37.4d2cc

temperature : msg.payload : number

46.738

CPU Monitoring Apps: Add a Speed gauge

- Add a name for the gauge: cpu monit
- Untick : Display group name
- Click **Add**



Group: [Home] cpu monit ▼

Size: auto

Type: Gauge ▼

Label: CPU speed

Value format: {{value}}

Units: MHz

Range: min 0 max 1500

Colour gradient:

Sectors: 0 ... optional ... optional ... 1500

Name:

▼ Description

▼ Node Help

Adds a gauge type widget to the user interface.

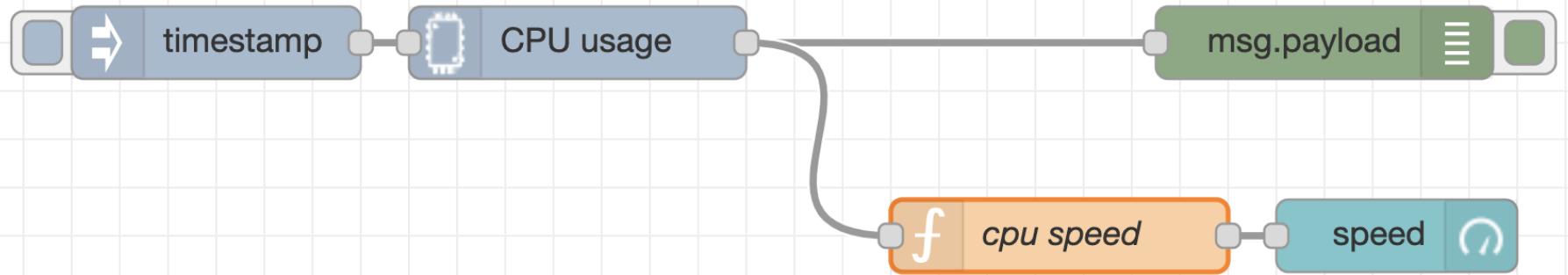
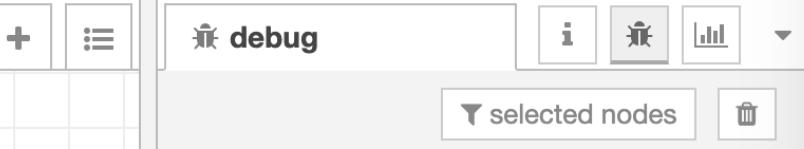
The **msg.payload** is searched for a numeric value and is formatted in accordance with the defined **Value Format**, which can then be formatted using [Angular filters](#).

For example : `{{value | number:1}}%` will round the value to one decimal place and append a % sign.

The colours of each of 3 sectors can be specified and the gauge will blend between them. The colours should be specified in hex (#rrggbb) format.

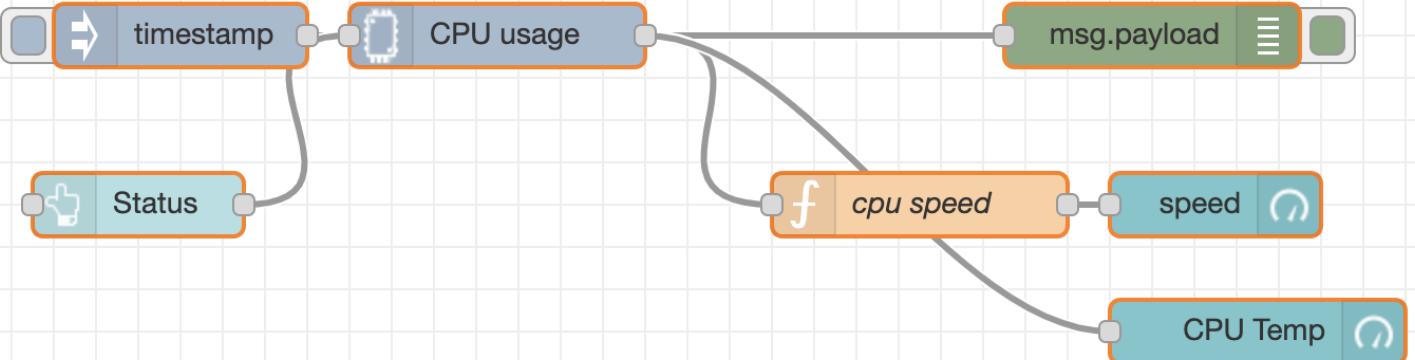
CPU Monitoring Apps: Add a Speed gauge

- Add a label name: CPU speed
- Add Units: MHz
- Set Range 0 to 1500
- Click **Done**



CPU Monitoring Apps: Add a Speed gauge

- The completed flow is illustrated in above diagram
- Next, Click **Deploy**
- Open a new internet browser: insert the RPi IP-add:1880 followed by /ui
- 192.168.1.xxx:1880/ui



dashboard



Layout

Site

Theme



Style

Light (default)

Base Settings

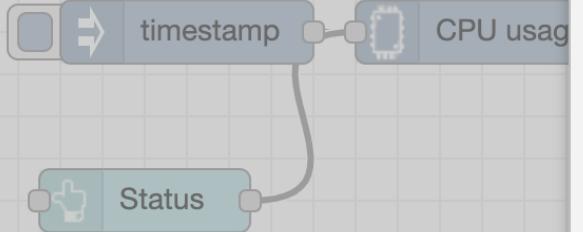
Colour

Font

System Font (default)

CPU Monitoring Apps: Add a Temp gauge

- Add another gauge node and a button node (named as status)
- Double click on second gauge node
- Set name, label and range



Group	[Home] cpu monit	<input type="button" value="▼"/>	<input type="button" value="✎"/>	
Size	auto			
Type	Gauge	<input type="button" value="▼"/>		
Label	CPU Temp			
Value format	<code>{{value}}</code>			
Units	C			
Range	min <input type="text" value="0"/>	max <input type="text" value="50"/>		
Colour gradient	<div style="display: flex; justify-content: space-around;"> </div>			
Sectors	0	... optional	... optional	... 50
Name	<input type="text"/>			

Description

Node Help

Adds a gauge type widget to the user interface. The `msg.payload` is searched for a numeric value and is formatted in accordance with the defined **Value Format**, which can then be formatted using [Angular filters](#).

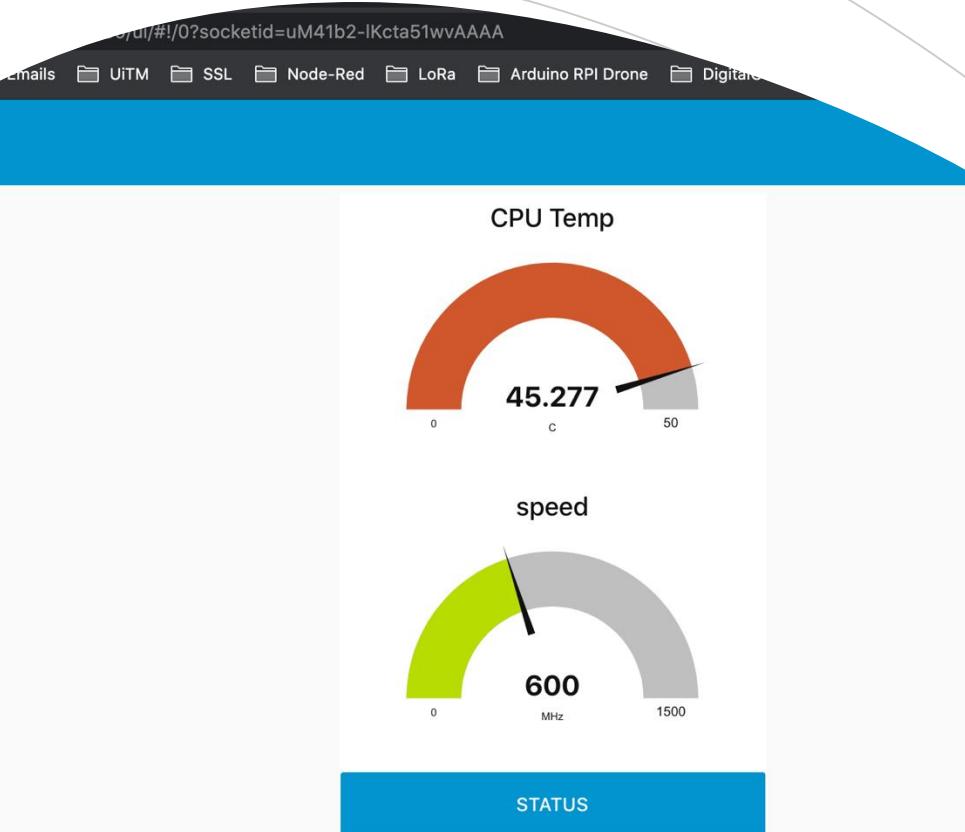
For example : `{{value | number:1}}%` will round the value to one decimal place and append a % sign.

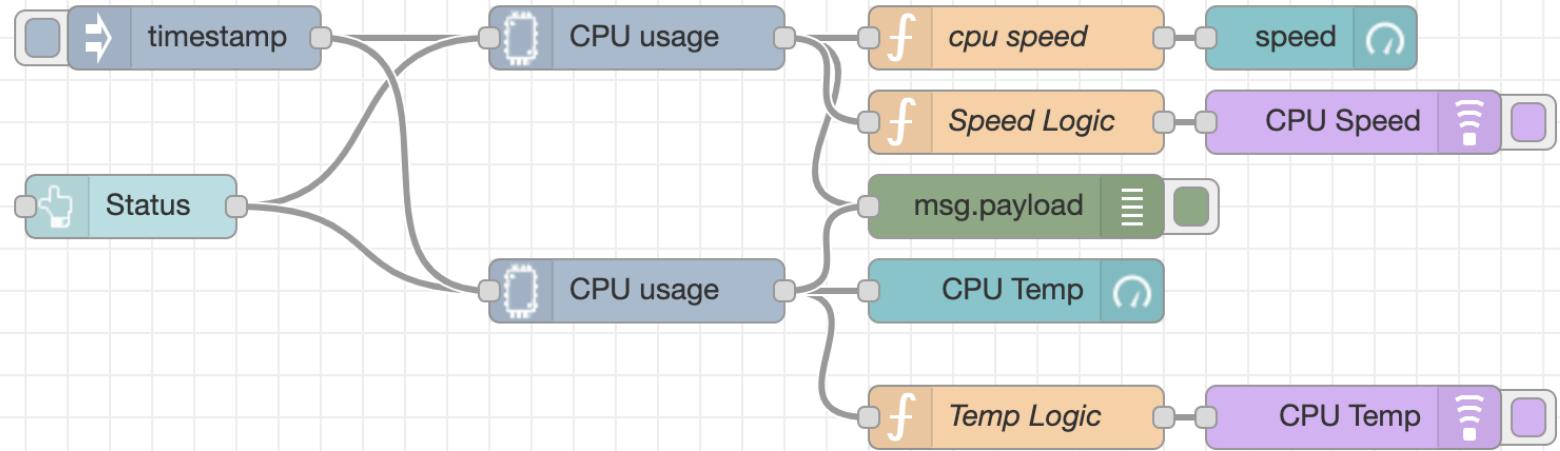
The colours of each of 3 sectors can be specified and the gauge will blend between them. The colours should be specified in hex (#rrggbb) format.

CPU Monitoring Apps: Adding Temp Gauge

- Set name, label and range
- Click **Done**
- Next, Click **Deploy**
- Open a new internet browser: insert the RPi IP-add:1880 followed by the /ui, or simply
- `192.168.1.xxx:1880/ui`

CPU Monitoring Apps: Dashboard UI



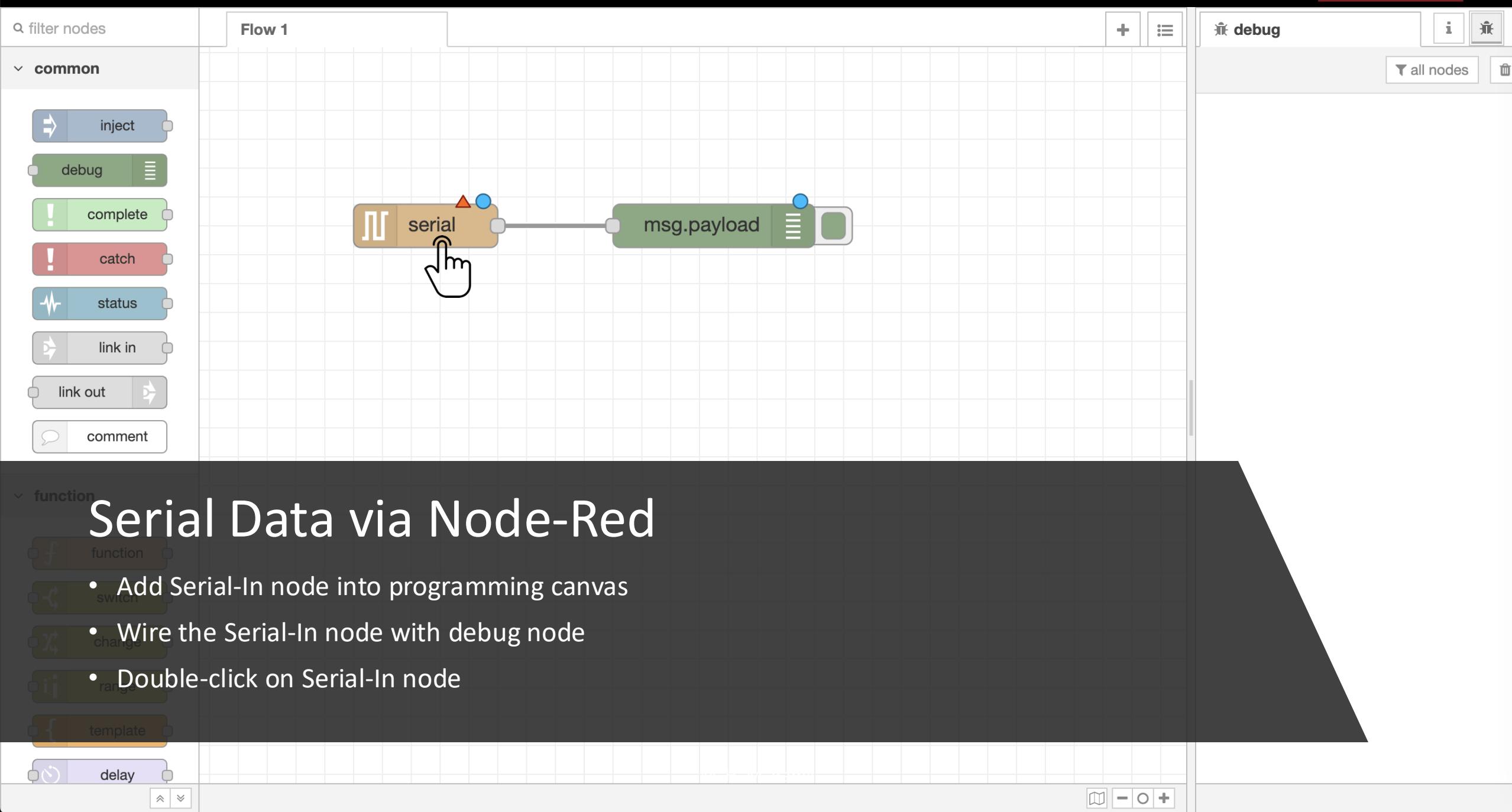


The dashboard interface shows a 'dashboard' tab selected. Below it, there are tabs for 'Layout', 'Site', and 'Theme'. On the right, there are icons for 'info', 'user', and 'chart'. The 'Tabs & Links' section contains a tree view of monitoring components:

- Home (selected)
- cpu monit
 - speed
 - CPU Temp
 - Status

CPU Monitoring Apps: Add Sound Element

- The complete node-red flow is available on github: <https://github.com/hanifr/docker-mariadb-mosquitto-rpi/blob/master/cpumonit.json>
- Please download and try to understand the logics and semantics



filter nodes

Flow 1

common

- inject
- debug
- complete
- catch
- status
- link in
- link out
- comment

function

- function
- switch
- change
- random
- template

delay

Edit serial in node

Delete Cancel Done

Properties

Serial Port Add new serial-port... 

Name 

Serial Data via Node-Red

- Click on Serial Port Tab
- Add Serial Port: /dev/ttyUSB0
- Add Baudrate: 9600

M H. M. Ramli

Enabled

all nodes

debug

i

warn

?

!

trash

filter nodes

Flow 1

common

- inject
- debug
- complete
- catch
- status
- link in
- link out
- comment

function

- function
- switch
- change
- random
- template

delay

Edit serial in node > **Add new serial-port config node**

Cancel Add

Properties

Serial Port: /dev/ttyUSB0

Settings

Baud Rate: 9600	Data Bits: 8	Parity: None	Stop Bits: 1
DTR: auto	RTS: auto	CTS: auto	DSR: auto

Input

Optionally wait for a start character of , then

Split input on the character \n

and deliver ascii strings

Output

Add character to output messages

Request

Default response timeout: 10000 ms

Tip: the "Split on" character is used to split the input into separate messages. Can accept chars (\$), escape codes (\n), or hex codes (0x03).

Enabled 0 nodes use this config On all flows

debug

all nodes

Nodes

Serial Data via Node-Red

- The configuration for the Serial communication is as per shown above
- Click **Add**
- Then, Click **Done**

Flow 1



filter nodes

common



inject



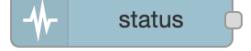
debug



complete



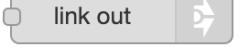
catch



status



link in



link out



comment

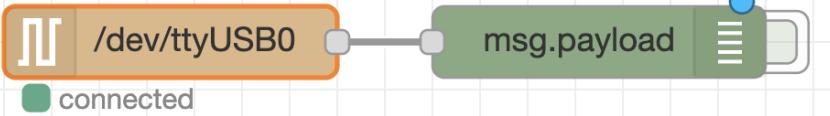
function

Serial Data via Node-Red

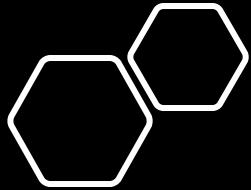
- The complete flow is illustrated in above diagram

- Click Deploy

- Then, open debug tab



16/08/2020, 16:51:32 node: fa20c9b5.a0f1a8
msg.payload : string[78]
▶ {"analog": [656, 658, 661, 679, 867, 708], "digital": [1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}◀
16/08/2020, 16:51:33 node: fa20c9b5.a0f1a8
msg.payload : string[78]
▶ {"analog": [610, 612, 620, 642, 802, 748], "digital": [1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1]}◀
16/08/2020, 16:51:33 node: fa20c9b5.a0f1a8
msg.payload : string[78]
▶ {"analog": [221, 212, 201, 190, 235, 233], "digital": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]}◀
16/08/2020, 16:51:34 node: fa20c9b5.a0f1a8
msg.payload : string[78]
▶ {"analog": [658, 656, 652, 659, 822, 660], "digital": [1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1]}◀
16/08/2020, 16:51:35 node: fa20c9b5.a0f1a8
msg.payload : string[78]
▶ {"analog": [632, 634, 643, 665, 845, 766], "digital": [1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}◀
16/08/2020, 16:51:35 node: fa20c9b5.a0f1a8
msg.payload : string[78]
▶ {"analog": [224, 215, 204, 193, 237, 234], "digital": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]}◀



Project 1: Designing Dashboard for Weather Station

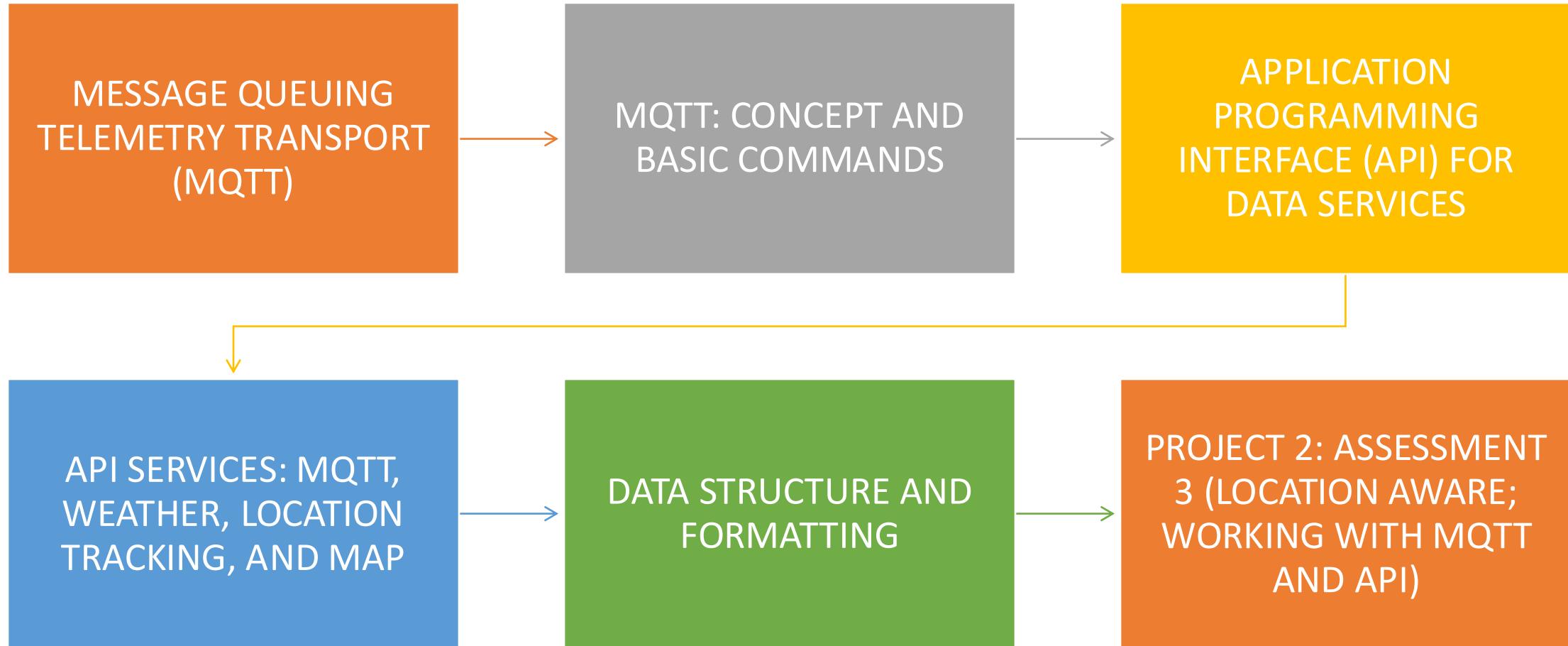
M. H. M. Ramli

- Use the following concepts
 - Data Acquisition via Arduino (LM35 and LDR sensors)
 - serial communication
 - Node-Red input and output nodes
 - Some creativity



Section 2: IOT COMMUNICATION PROTOCOL & DATA TRANSMISSION

Agenda: Section 2



Message Queuing Telemetry Transport (MQTT) : Concept



Architecture:

a Client Server publish/subscribe messaging transport protocol.



Applications:

Wide range; Machine to Machine (M2M) & IoT contexts with small code footprint and network bandwidth.



Goal:

light weight, open, simple, and designed to be easy to implement.



For this Training, we will be using Mosquitto MQTT.



Others including HIVEMQ, CLOUDMQTT, Heroku, IBM Bluemix, ThingStudio, ThingMQ, Microsoft Azure IoT, and MQTT.io (under development).

MQTT Syntax

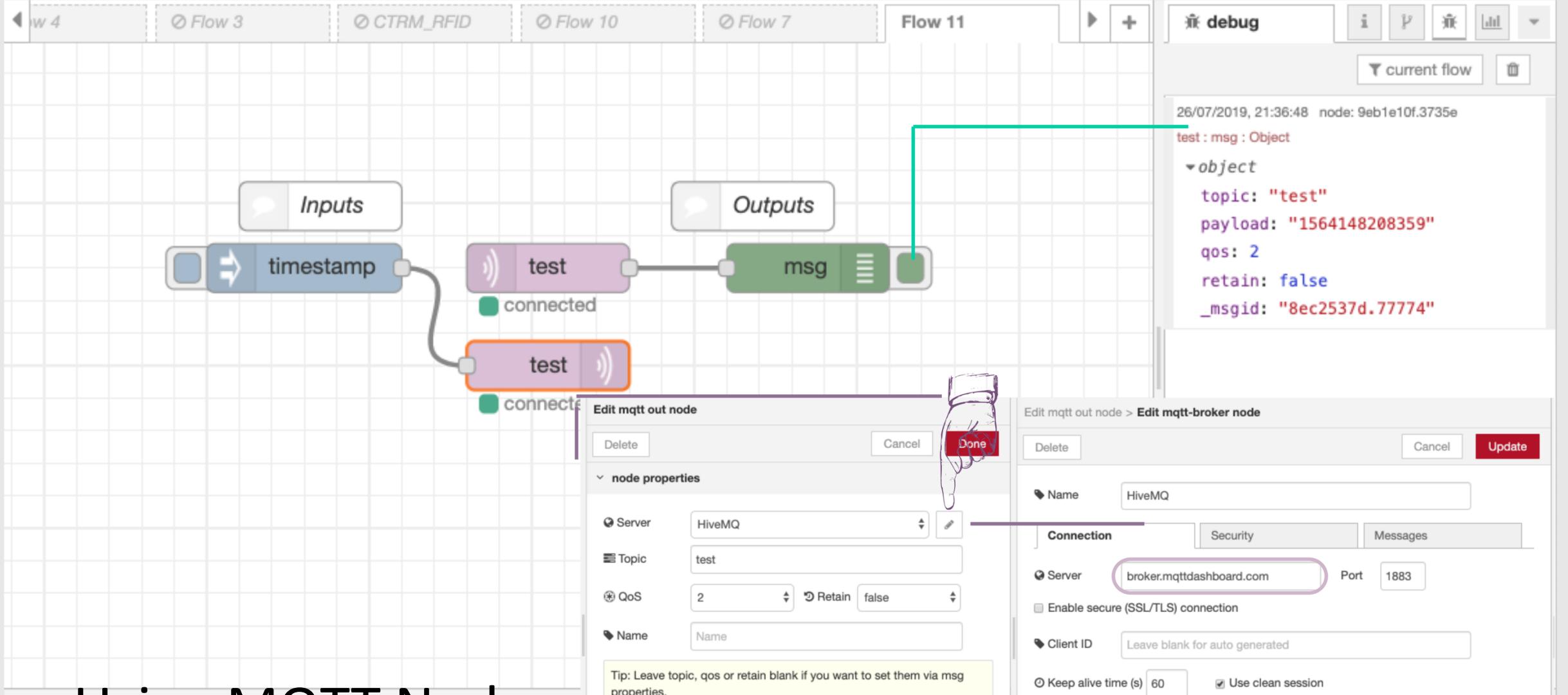
to subscribe to a topic, use:

```
mosquitto_sub -d  
-h server_address  
-t topic
```

to publish to a topic, use:

```
mosquitto_pub -d  
-h server_address  
-t topic -m "msg"
```

-d: stands for debug, it displays the status of connection



Using MQTT Node

- This example uses a public host to serve a data transmission. Set MQTT node as per illustrated, once finished click deploy. (Details: [MQTT Syntax](#))

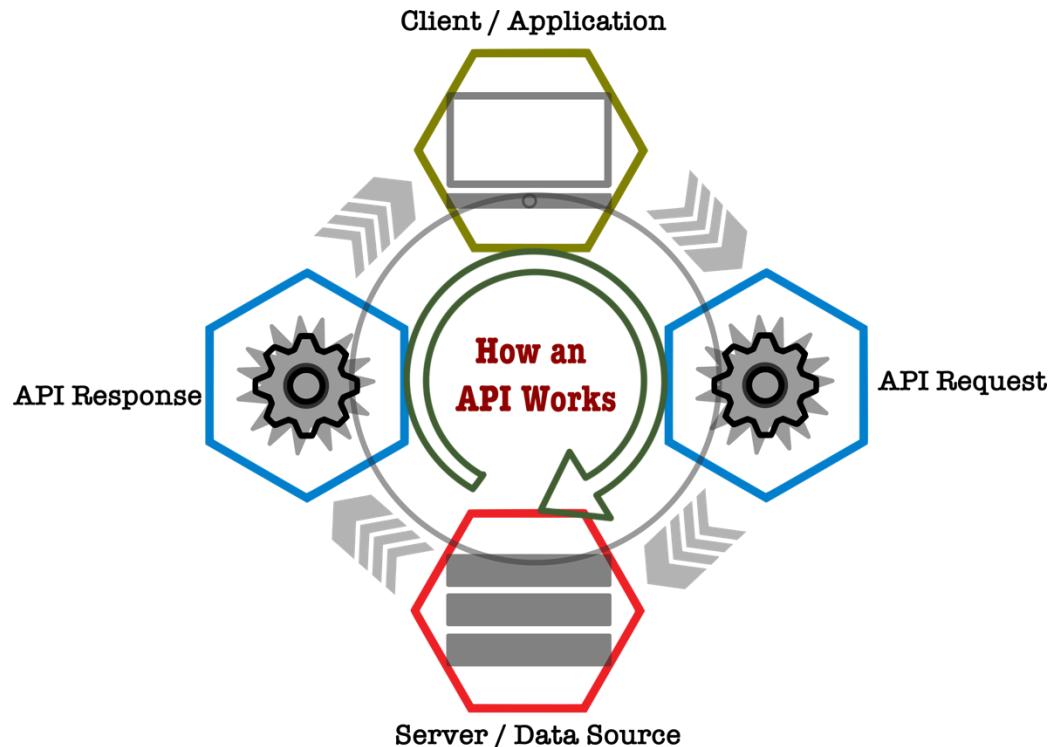
MQTT Syntax

- **topic:** “**test**”, –subscribe/publish must be same topic.
- **payload:** “**1564150020817**”, –received payload (data) from the server (when subscribe to a topic).
- **qos:** **2**, –Quality of Service: sent data
 - At most once (0)
 - At least once (1)
 - Exactly once (2)
- **retain:** **true**, –A retained message is a normal MQTT message with the retained flag set to true. The broker stores the last retained message and the corresponding QoS for that topic.
- **_msgid:** “**d8f2a40.3de586**” –id for the output

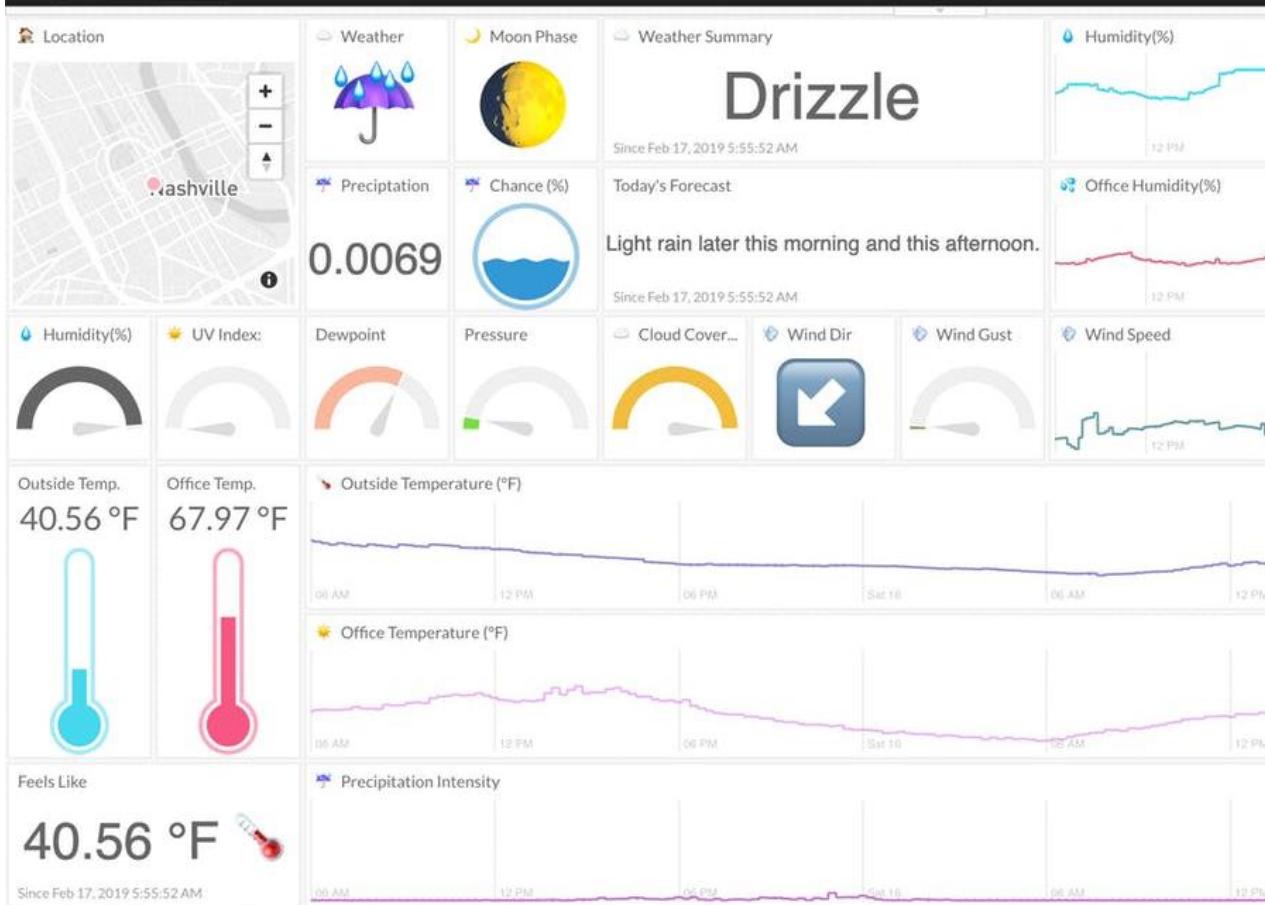


APPLICATION PROGRAMMING INTERFACE: API

Concept: API



- **Programmatic access:** An application program interface that provides a developer with programmatic access to proprietary software application.
- **Communication:** A software intermediary that makes it possible for application programs to interact with each other and share data.
- **Goal:** get information/data and response.



- This example illustrates how to use Malaysian Meteorological Department Web Service API and get the weather data to UI dashboard.

Weather Station



What is Malaysian Meteorological Department Web Service API?

Web Service API (Application Programming Interface) is a free service offered by Malaysian Meteorological Department to the general public. The service, which serves on top of HTTP, response, a human readable parseable data

What data are offered through the Web Service?

Malaysian Meteorological Department offers the general weather forecast data. For other meteorological data, please contact Malaysian Meteorological

How do I access the data?

Click the "Get Your Access Token" below. Simply follow the registration instruction. Once you have received your access token, head to the developer's guide for detailed information regarding the API endpoints.

Weather Station: API registration

Weather Forecast-API : Go to <https://api.met.gov.my/> and click Get Your Access Token

Base URL
Data Interchange Format
Authentication
Throttling
Endpoints
Request & Response
Data Available
General Forecast
Marine Forecast
Locations
Stations
Warnings
Satellite & Radar Images

Your API Token



Token

Your Personal Information

Mohd Hanif Bin Mohd Ramli

REDACTED

REDACTED

A Quick Start Guide

Get the weather forecast for today for Putrajaya.

```
curl -H "Authorization: METToken REDACTED" "https://api.met.gov.my/v2/data?
```

Change Your Password

Your Old Password

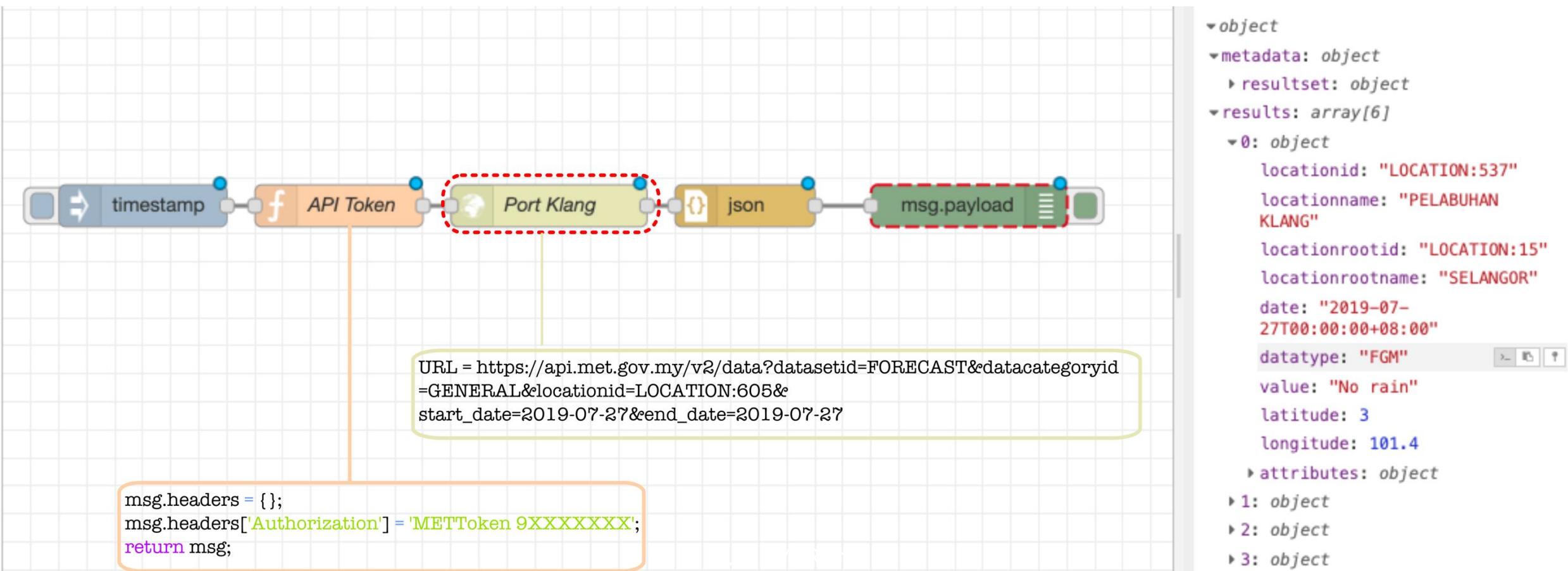
Weather Station: API registration

Weather Forecast-API : Go to <https://api.met.gov.my/dashboard> and copy Your API Token

M. H. M. Ramli

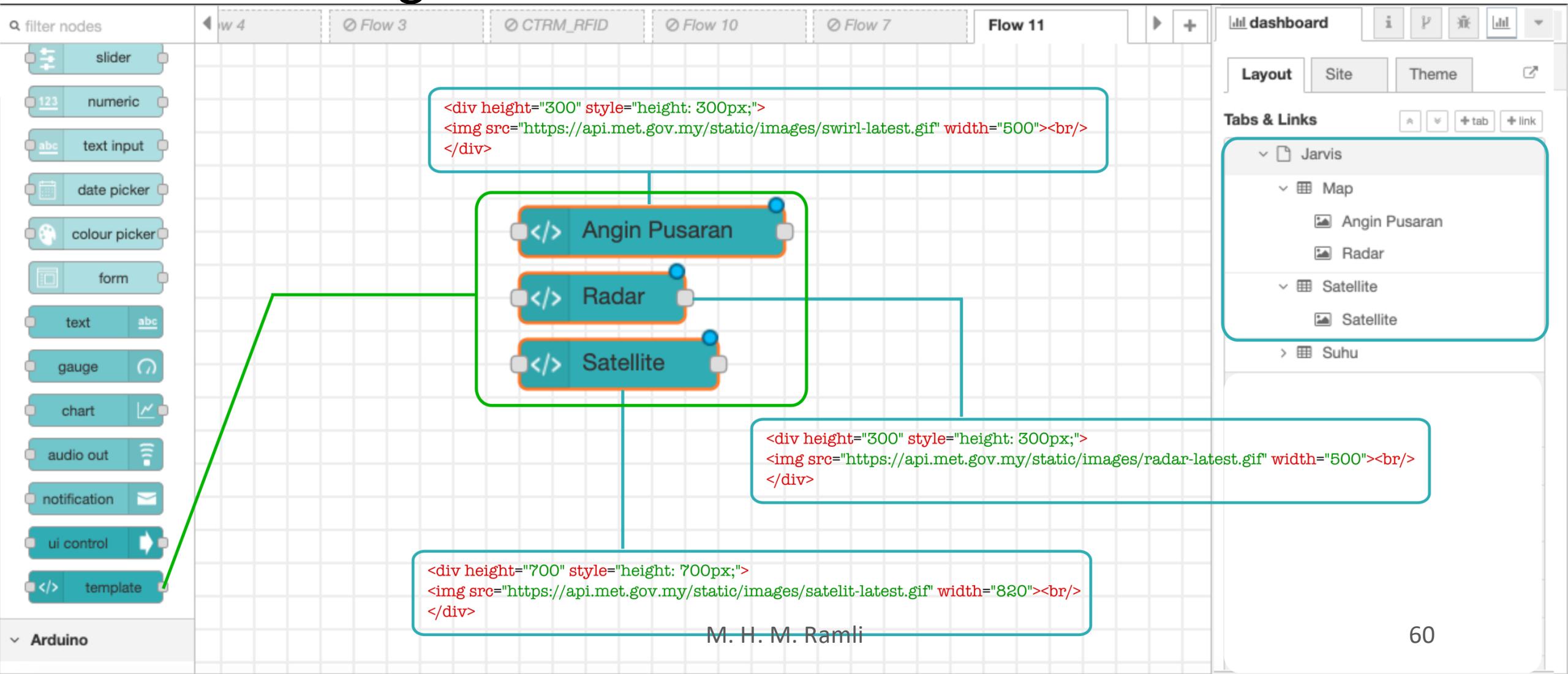
Weather Station

- This flow access a weather data from location ID:605 (Port Klang) for specific date. Please change the date accordingly.



Weather Station: Dashboarding

- Dashboard Layout : The settings of dashboard is as per illustrated. Size: 8X4 (Angin Pusaran), 8X5 (Radar) & 13X9 (Satellite).





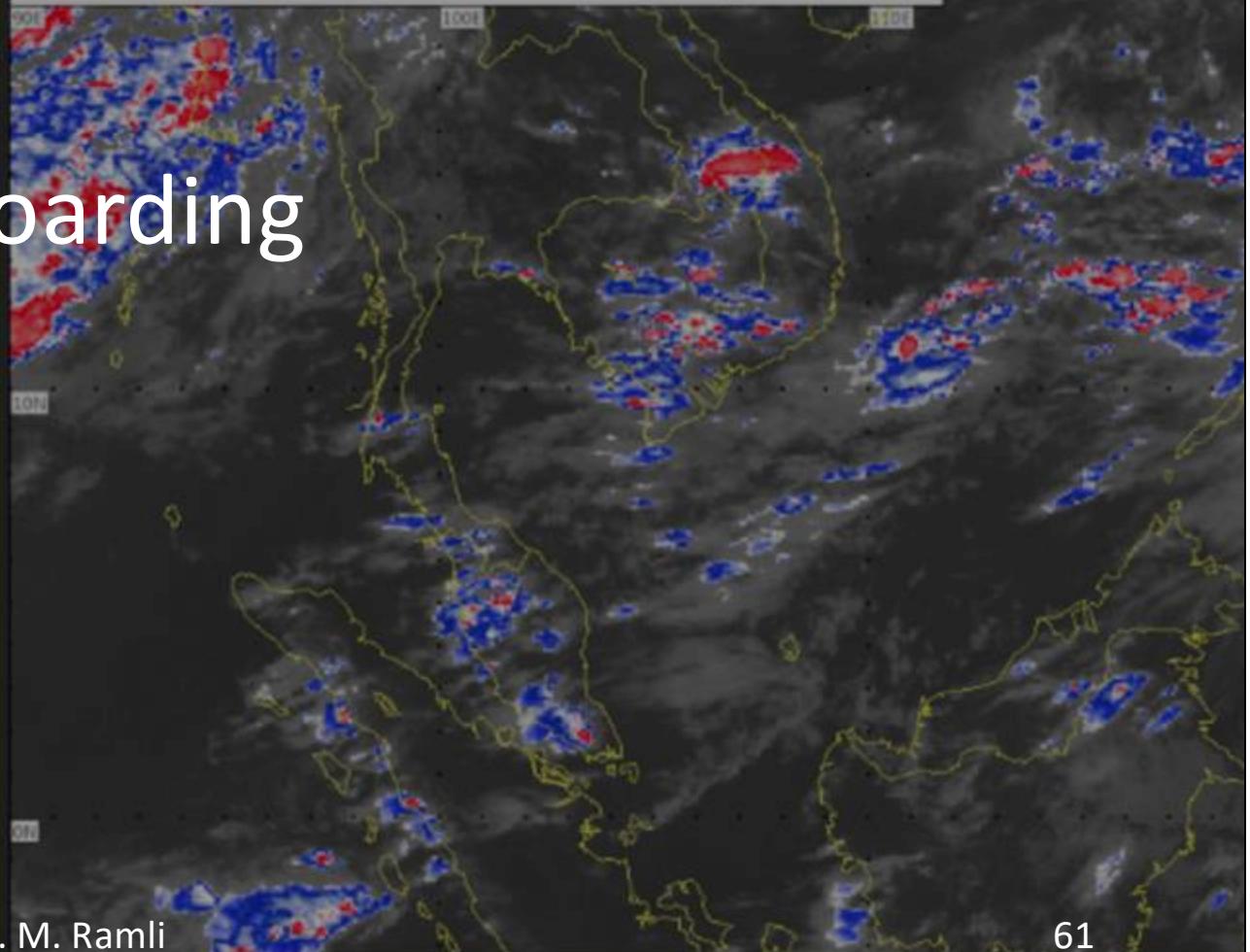
Observed 05:30PM Saturday 27Jul2019

Observed 05:30PM Saturday 27Jul2019



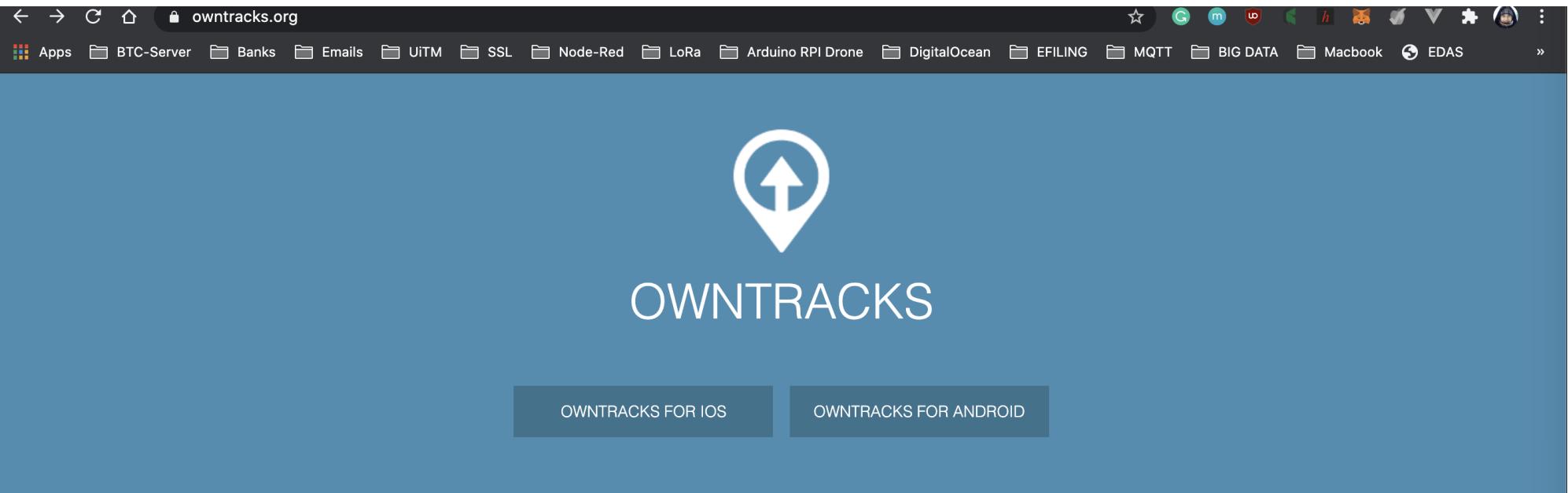
Weather Station: Dashboarding

HIMAWARI-8 ASEAN INFRARED ENHANCED IR 10.4 CH13 27/7/2019 1000 UTC



Other Weather-API

- AccuWeather: 50 Calls per day
- OpenWeatherMap: 60 Calls per minute
- Weatherbit: 500 Calls per day
- Dark Sky: 1000 calls per day (acquired by Apple in late 2019)
- Weather2020
 - a) In general, all above-mentioned APIs are paid platforms.
 - b) However, it is still free if you make request less than maximum calls.
 - c) In the previous example, the provided data is not as rich as above APIs. Furthermore, it is quite challenging to extract specific accordingly.
 - d) In view of above limitations, it is best to consider Dark Sky or OpenWeatherMap Weather API as it allows request of data based on GPS points.

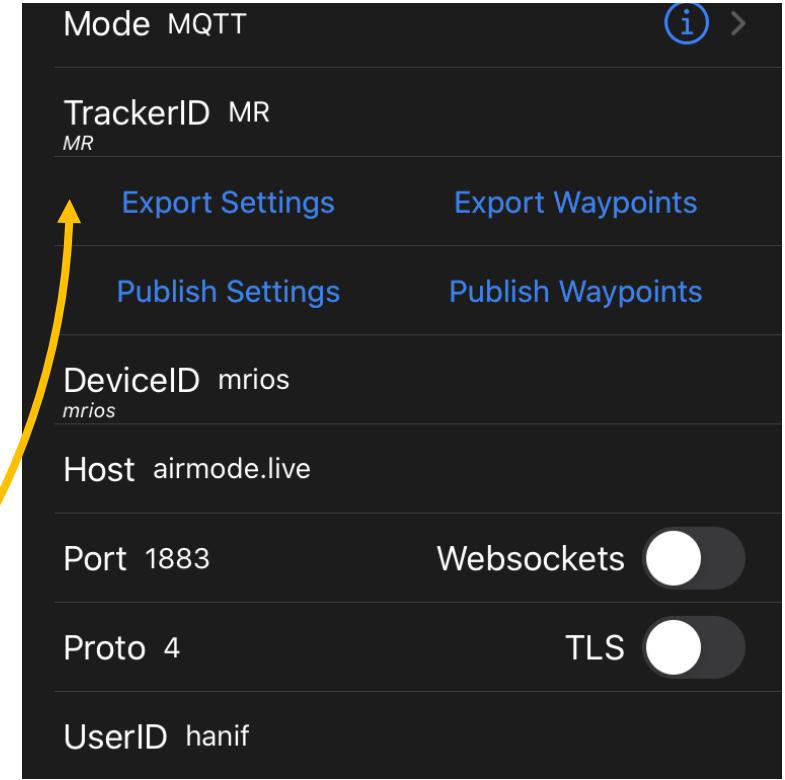
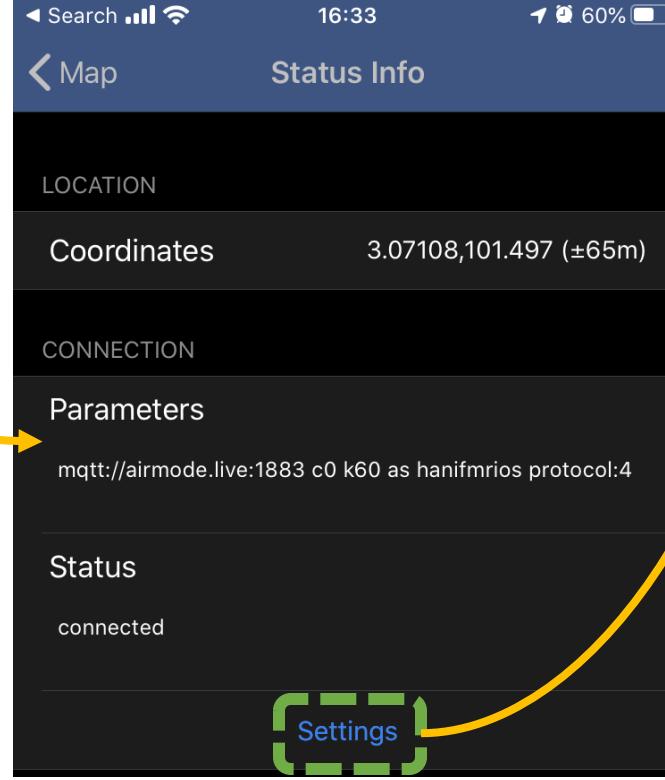
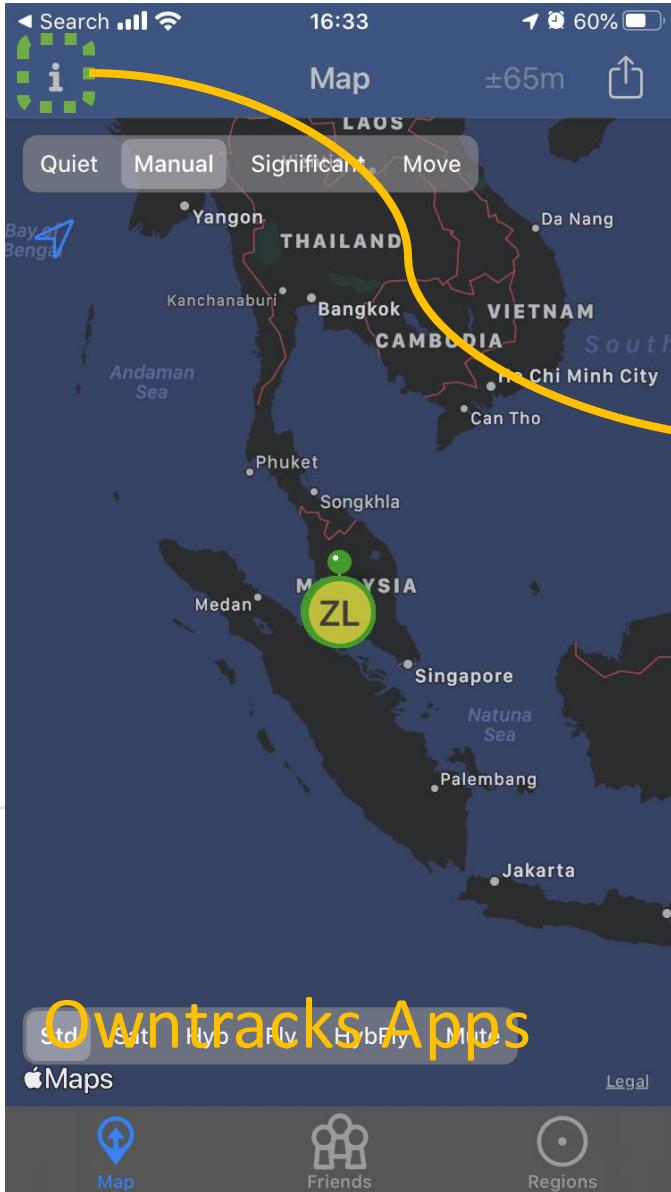


The screenshot shows the homepage of the OwnTracks website. At the top, there's a navigation bar with various links like Apps, BTC-Server, Banks, Emails, UI-ITM, SSL, Node-Red, LoRa, Arduino RPI Drone, DigitalOcean, EFILING, MQTT, BIG DATA, Macbook, EDAS, and a search icon. Below the navigation is a large blue header with the OwnTracks logo (a white location pin with an upward arrow) and the word "OWNTRACKS". Underneath the logo are two buttons: "OWNTRACKS FOR IOS" and "OWNTRACKS FOR ANDROID". To the right of the header, there's a "What's New" section showing version 13.1.7 released 7 months ago, which includes a few bug fixes, notably a fix for not seeing friends. There's also a "Version History" link. On the far right, there's a "Preview" section showing screenshots of the app interface on iOS and Android devices.

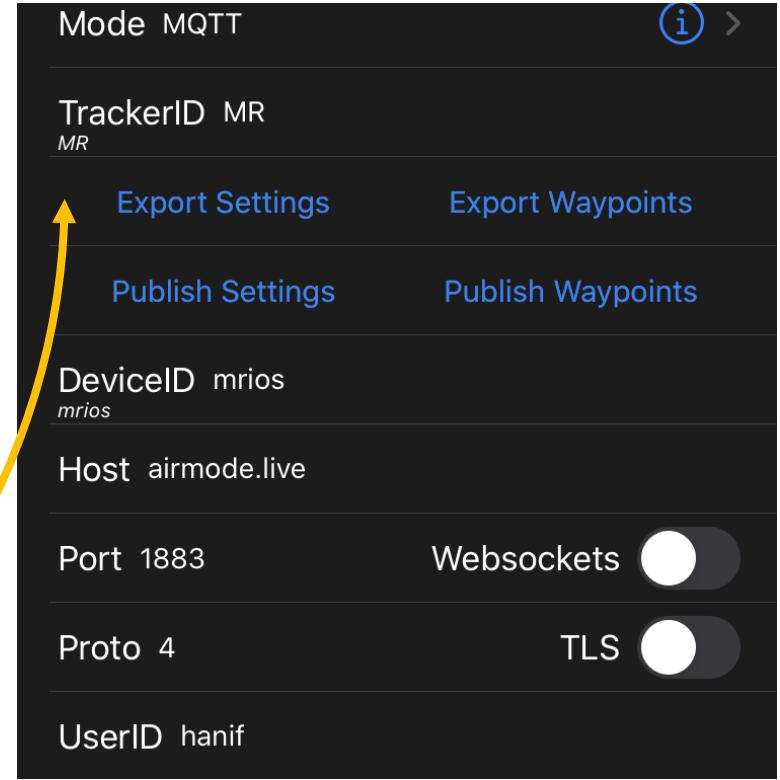
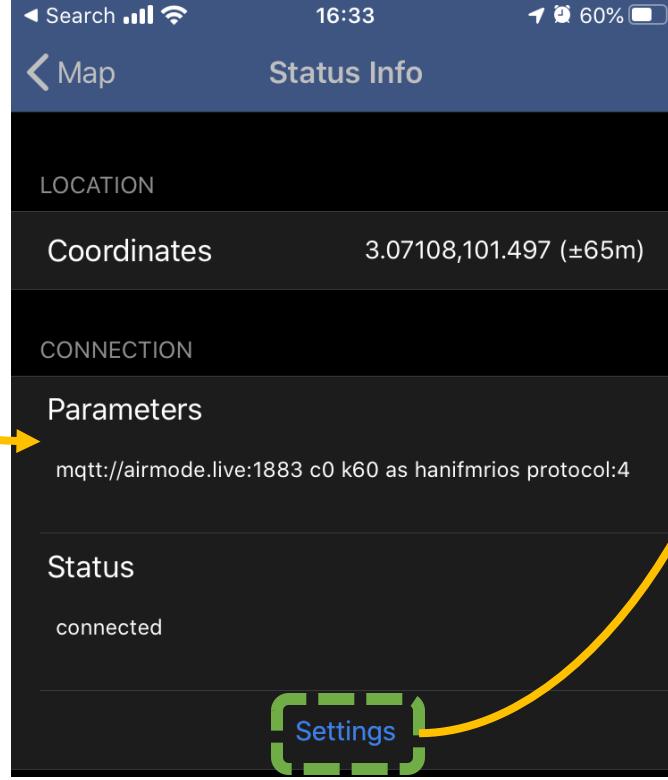
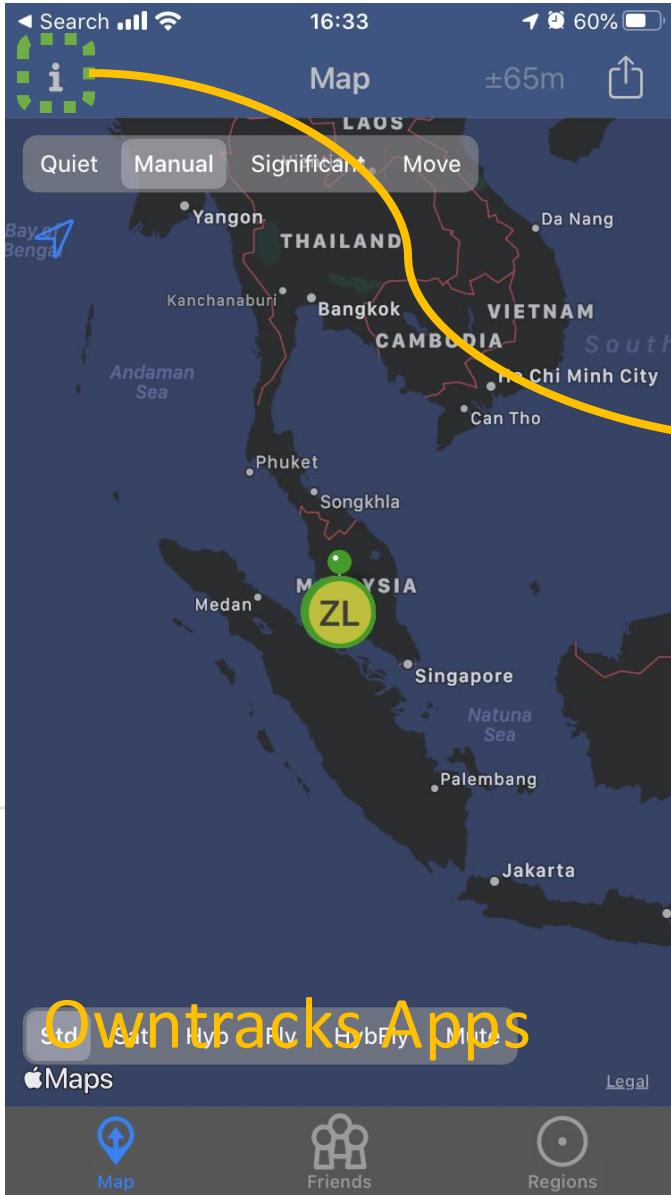
Your location companion

OwnTracks allows you to keep track of your own location. You can build your private location diary or share it with your family and friends. OwnTracks is open-source and uses open protocols for communication so you can be sure your data stays secure and private.

Location API



- Download Owntracks apps on google store for android, or apps store for iOS.
- Tool for tracking a person. Using MQTT comm. protocol.
- Launch the apps and set TrackerID, DeviceID, Host and Port



- Set TrackerID: XX
- set DeviceID: yourname
- set Host: airmode.live
- Set Port: 1883

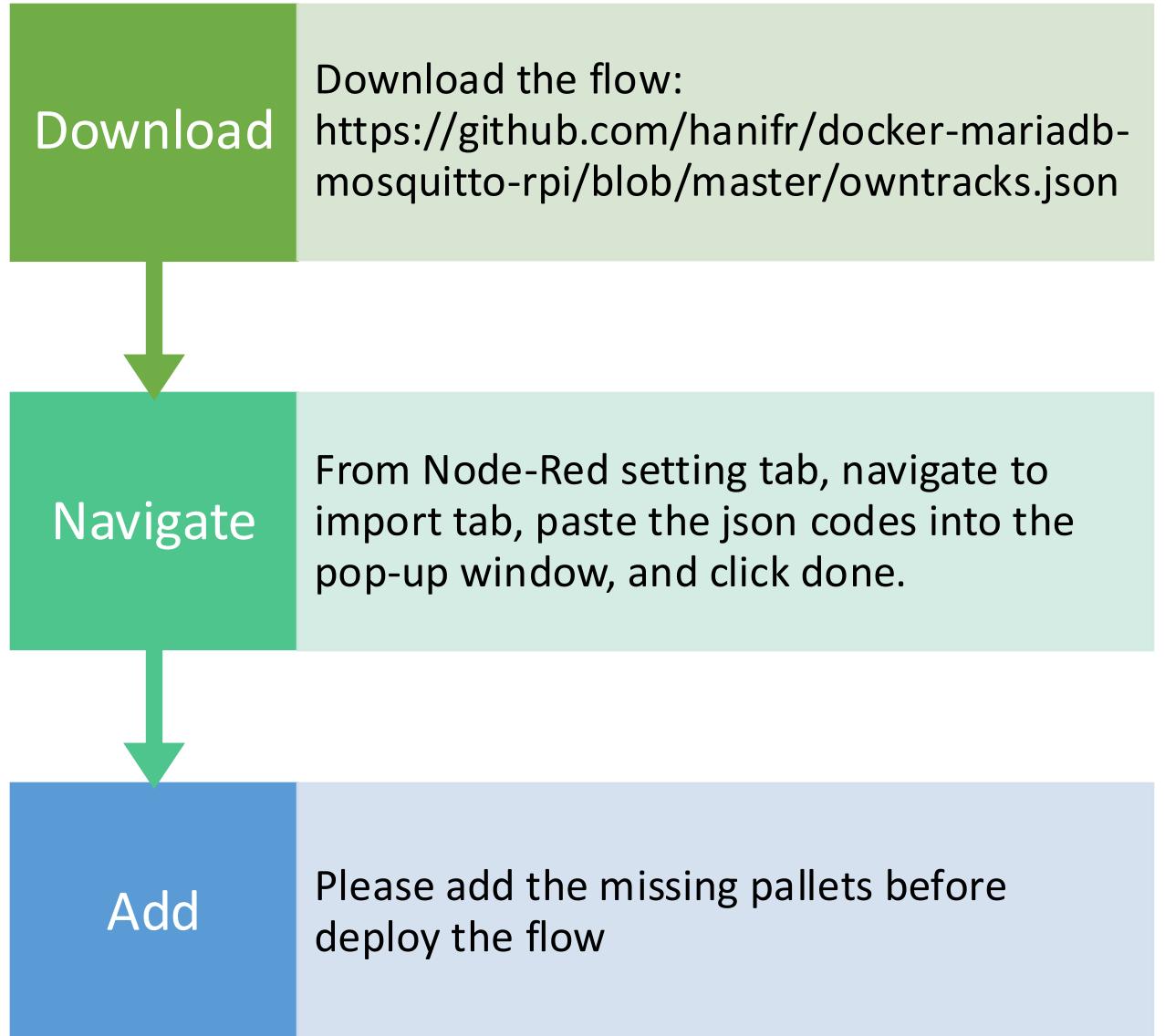
Owntracks API: JSON

- acc: Accuracy of the reported location in meters without unit (*iOS,Android/integer/meters/optional*)
- alt: Altitude measured above sea level (*iOS,Android/integer/meters/optional*)
- batt: Device battery level (*iOS,Android/integer/percent/optional*)
- bs: Battery Status 0=unknown, 1=unplugged, 2=charging, 3=full (*iOS*)
- cog: Course over ground (*iOS/integer/degree/optional*)
- lat: latitude (*iOS,Android/float/degree/required*)
- lon: longitude (*iOS,Android/float/degree/required*)
- rad: radius around the region when entering/leaving (*iOS/integer/meters/optional*)

Owntracks API: JSON

- tid: Tracker ID used to display the initials of a user (*iOS,Android/string/optional*) required for http mode
- tst: UNIX [epoch timestamp](#) in seconds of the location fix (*iOS,Android/integer/epoch/required*)
- vac: vertical accuracy of the alt element (*iOS/integer/meters/optional*)
- vel: velocity (*iOS,Android/integer/kmh/optional*)
- p: barometric pressure (*iOS/float/kPa/optional/extended data*)
- conn: Internet connectivity status (route to host) when the message is created (*iOS,Android/string/optional/extended data*)
 - w: phone is connected to a WiFi connection (*iOS,Android*)
 - o: phone is offline (*iOS,Android*)
 - m: mobile data (*iOS,Android*)

Practical Session with Owntracks API



Node-RED

Bomba_Jinjang sonar actual data + ⚡

common

inject

debug

complete

catch

status

link in

link out

comment

function

switch

change

range

template

delay

Import nodes

Clipboard

Paste flow json or select a file to import

Library Examples

```
es": [{"id": "6684c1a8.f9931", "type": "inject", "z": "5c19add1.ab3d64", "name": "MySQL:drop table owntracks_data", "topic": "drop table owntracks_data", "payload": "", "payloadType": "date", "repeat": "", "crontab": "", "once": false, "onceDelay": 0.1, "x": 240, "y": 320, "wires": [{"id": "354bc158.637b0e"}]}, {"id": "f8032ceb.10023", "type": "inject", "z": "5c19add1.ab3d64", "name": "show data", "topic": "select * from owntracks_data", "payload": "", "payloadType": "date", "repeat": "", "crontab": "", "once": false, "onceDelay": 0.1, "x": 320, "y": 360, "wires": [{"id": "354bc158.637b0e"}]}, {"id": "e5d5787.1d5bc88", "type": "debug", "z": "5c19add1.ab3d64", "name": "", "active": true, "toSidebar": true, "console": false, "tosatus": false, "complete": true, "targetType": "full", "x": 690, "y": 220, "wires": []}, {"id": "354bc158.637b0e", "type": "mysql", "z": "5c19add1.ab3d64", "mydb": "3eb2880c.095418", "name": "owntracks_data", "x": 540, "y": 220, "wires": [{"id": "5d5787.1d5bc88"}]}, {"id": "5d5787.1d5bc88", "type": "MySQLdatabase", "z": "5c19add1.a53dc87", "name": "owntracks_data", "host": "192.168.1.4", "port": "3306", "db": "tracker", "tz": "UTC + 8.00"}]
```

Import to current flow new flow Cancel Import

bombaJinjang/distance/Ducati Ducati_actual

M. H. M. Ramli Sonar ⚡

info

Search flows

Flows

- Bomba_Jinjang
- sonar actual data

Subflows

Global Configuration Nodes

Selection 9 nodes

Switch flow tabs with **⌘↑j** and **⌘↑k**

69

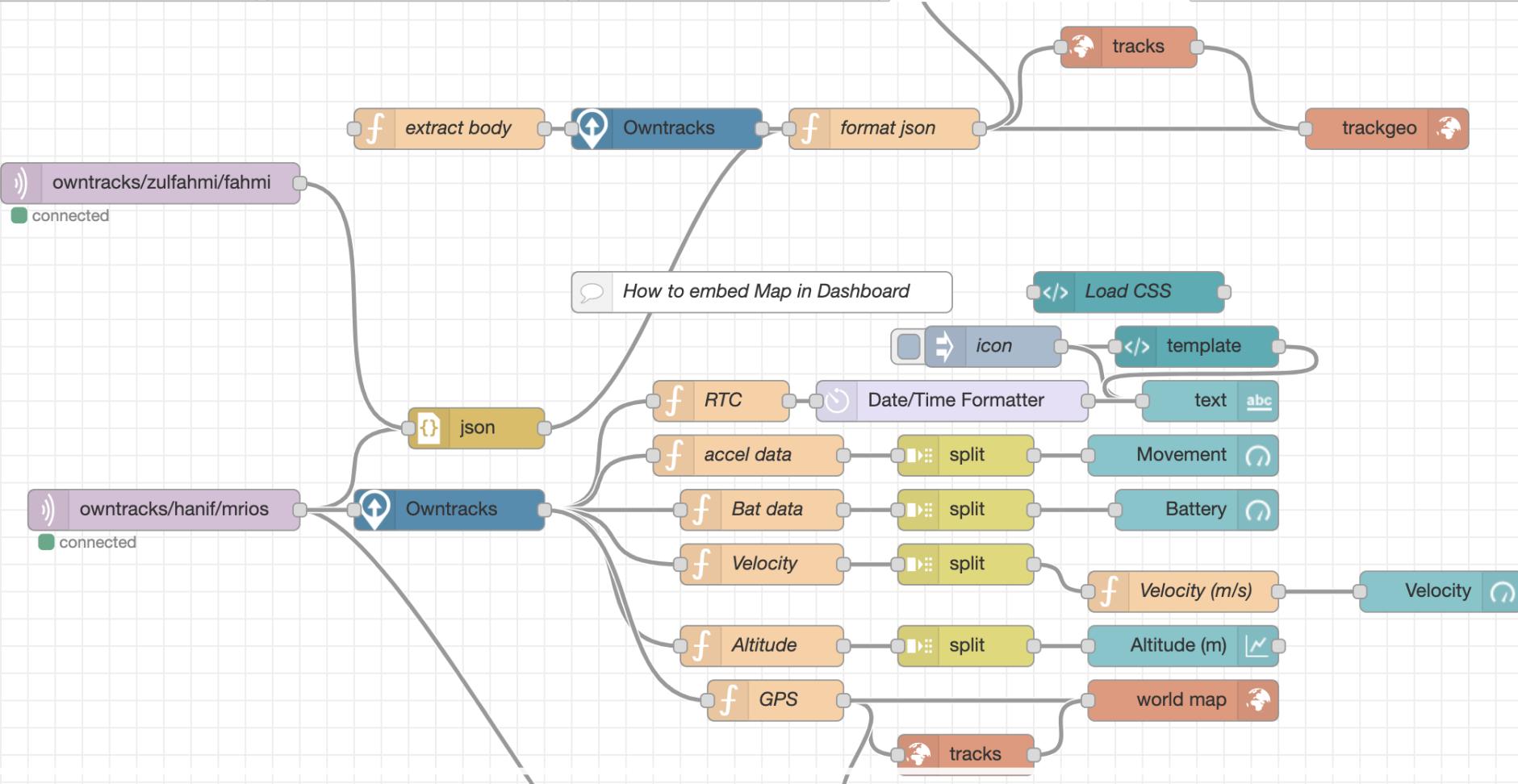
Flow 1

Met

Flow 2

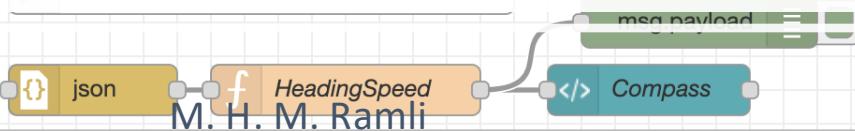
Owntracks

+



Owntracks: Node-Red Flow

Using JSON to decide incoming msg



Data structure and formatting



JavaScript Object Notation: JSON

- JSON : is a language-independent data format. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data.
 - stands for JavaScript Object Notation
 - a lightweight format for storing and transporting data
 - often used when data is sent from a server to a web page
 - “self-describing” and easy to understand

filter nodes

split
join
sort
batch

parser

/dev/ttyUSB0
connected

json

msg.payload

storage

yaml

file

file in

watch

tail

Flow 1

debug

all nodes

16/08/2020, 16:54:31 node: fa20c9b5.a0f1a8
msg.payload : Object
▶ { analog: array[6], digital: array[14] }

16/08/2020, 16:54:31 node: fa20c9b5.a0f1a8
msg.payload : Object
▶ { analog: array[6], digital: array[14] }

16/08/2020, 16:54:32 node: fa20c9b5.a0f1a8
msg.payload : Object
▶ { analog: array[6], digital: array[14] }

16/08/2020, 16:54:32 node: fa20c9b5.a0f1a8
msg.payload : Object
▶ { analog: array[6], digital: array[14] }

16/08/2020, 16:54:33 node: fa20c9b5.a0f1a8
msg.payload : Object
▶ { analog: array[6], digital: array[14] }

16/08/2020, 16:54:34 node: fa20c9b5.a0f1a8
msg.payload : Object
▶ { analog: array[6], digital: array[14] }

16/08/2020, 16:54:34 node: fa20c9b5.a0f1a8
msg.payload : Object
▶ { analog: array[6], digital: array[14] }

Formatting Serial Data using JSON

- In this example, we use a flow from [Serial Data via Node-Red](#) with additional JSON node.
- Note: JSON node convert string data to object data
- Click on the object (either analog or digital to observe the data)

filter nodes

comment

function

- function
- switch
- change
- range
- template
- delay
- trigger

exec

random

- Add a chart and a function nodes onto the flow
- Copy path of one object to observe on chart
- Process the object data through a function node

network

mqtt in

mqtt out

Flow 1

msg.payload : Object

► { analog: array[6], digital: array[14] }

16/08/2020, 16:54:31 node: fa20c9b5.a0f1a8

msg.payload : Object

► object

► analog: array[6]

- 0: 244
- 1: 237
- 2: 227
- 3: 218
- 4: 267
- 5: 280

► digital: array[14]

16/08/2020, 16:54:32 node: fa20c9b5.a0f1a8

msg.payload : Object

► { analog: array[6], digital: array[14] }

16/08/2020, 16:54:32 node: fa20c9b5.a0f1a8

msg.payload : Object

► { analog: array[6], digital: array[14] }

16/08/2020, 16:54:33 node: fa20c9b5.a0f1a8

msg.payload : Object

► { analog: array[6], digital: array[14] }

16/08/2020, 16:54:34 node: fa20c9b5.a0f1a8

msg.payload : Object

Copy path

W. H. W. Kamm

Navigation icons: back, forward, search, etc.

filter nodes comment

function

- function
- switch
- change
- range
- template
- delay
- trigger

exec

random

smooth

network

- mqtt in
- mqtt out

Flow 1

Edit function node

Delete Cancel Done

Properties

Name: analog 0

Function:

```
1 msg.payload = msg.payload.analog[0];
2 return msg;
```

Outputs 1

M H. M. Ramli Enabled

debug

msg.payload : Object

▶ { analog: array[6], digital: array[14] }

16/08/2020, 16:54:31 node: fa20c9b5.a0f1a8

msg.payload : Object

object

analog: array[6]

0: 244
1: 237
2: 227
3: 218
4: 267
5: 280

digital: array[14]

16/08/2020, 16:54:32 node: fa20c9b5.a0f1a8

msg.payload : Object

▶ { analog: array[6], digital: array[14] }

16/08/2020, 16:54:32 node: fa20c9b5.a0f1a8

msg.payload : Object

▶ { analog: array[6], digital: array[14] }

16/08/2020, 16:54:33 node: fa20c9b5.a0f1a8

msg.payload : Object

▶ { analog: array[6], digital: array[14] }

16/08/2020, 16:54:34 node: fa20c9b5.a0f1a8

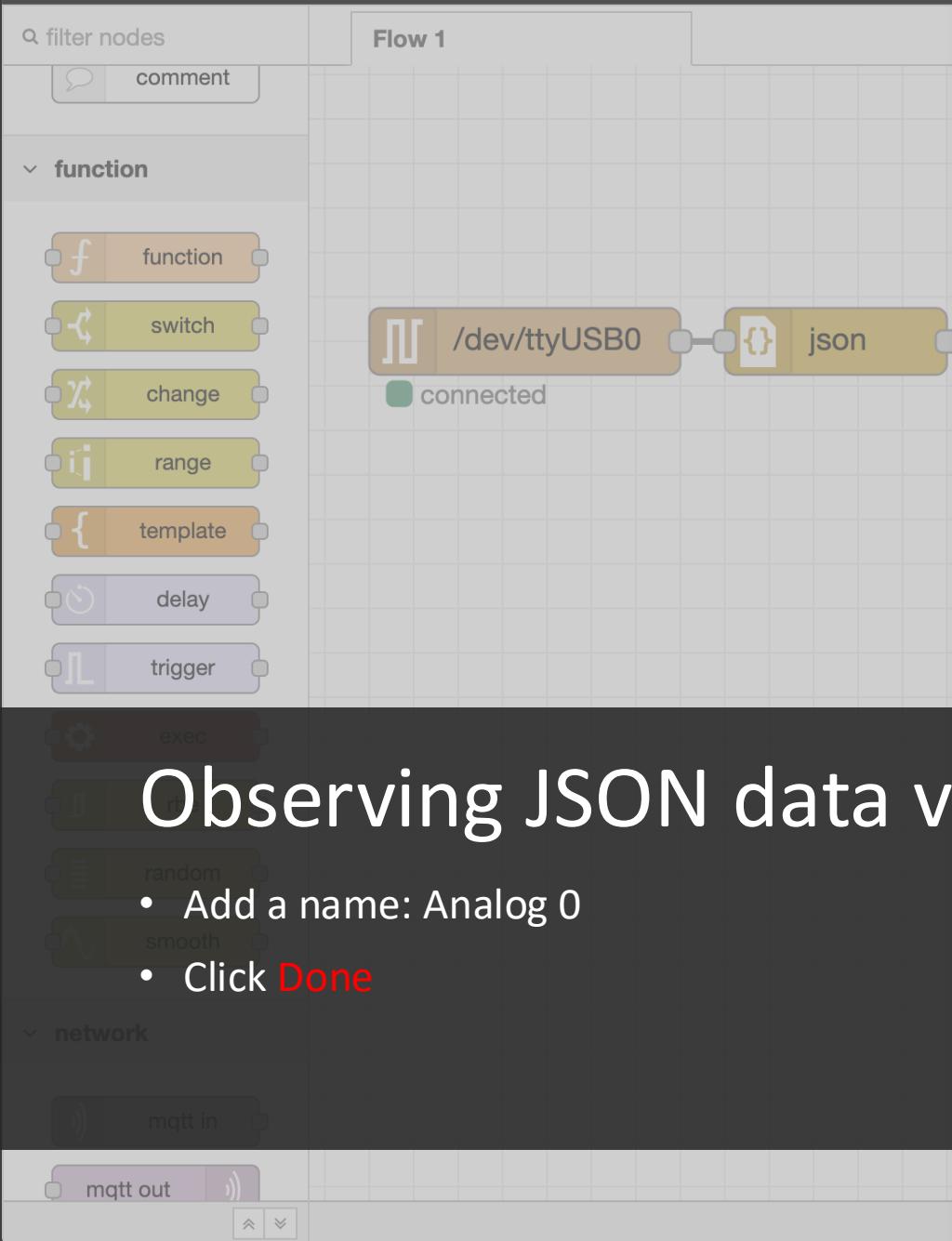
msg.payload : Object

all nodes

⋮

Observing JSON data via Dashboard

- Double click on the function node
- Add a name: analog zero
- Set msg.payload as msg.payload.analog[0];



Edit chart node

Delete Cancel Done

Properties

Group: [Home] Analog

Size: 16 x 5

Label: Analog 0

Type: Line chart enlarge points

X-axis: last 1 hours OR 1000 points

X-axis Label: HH:mm:ss as UTC

Y-axis: min max

Legend: None

Interpolate: bezier

Series Colours:

- Dark Blue
- Grey
- Brown
- Green
- Dark Green
- Dark Brown
- Red
- Dark Red
- Dark Purple

Blank label: display this text before valid data arrives

Enabled:

M. H. M. Ramli

This is a screenshot of the "Edit chart node" dialog. It contains fields for Group (set to "[Home] Analog"), Size (16 x 5), Label (Analog 0), Type (Line chart with "enlarge points" checked), X-axis settings (last 1 hours OR 1000 points), and X-axis Label (HH:mm:ss with "as UTC" unchecked). The Y-axis, Legend, and Interpolate options are also visible. Below the dialog is a preview area showing a chart with multiple colored series and a legend entry for "None". A note at the bottom says "display this text before valid data arrives". A radio button for "Enabled" is shown, and the name "M. H. M. Ramli" is listed at the bottom.

dashboard

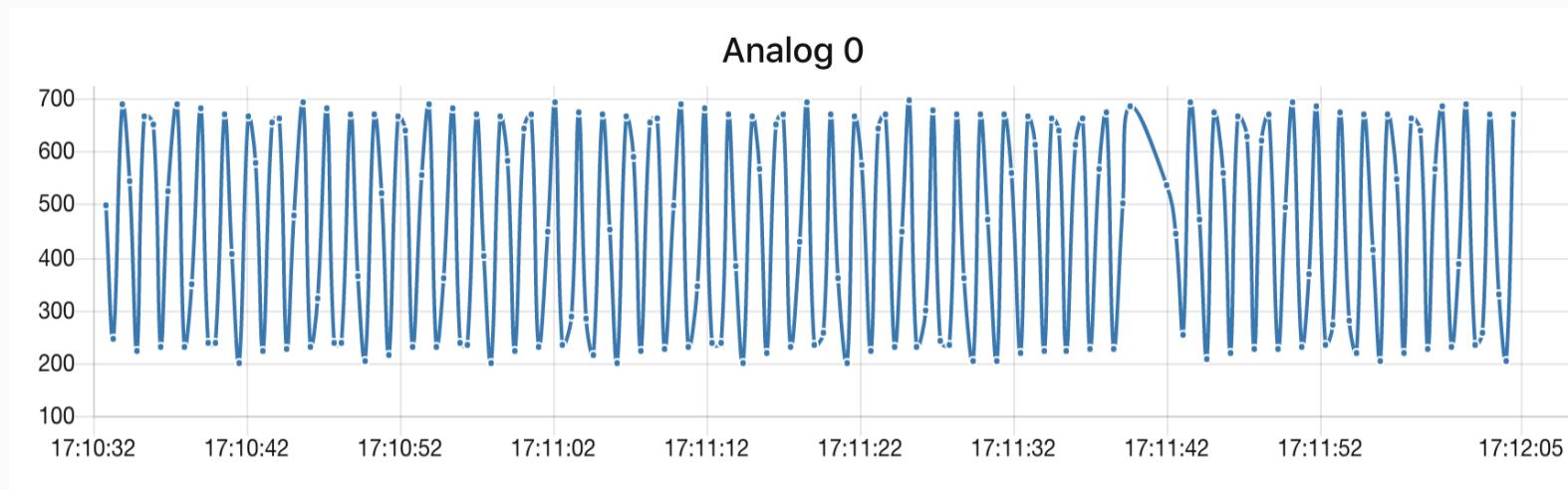
Layout Site Theme

Tabs & Links

Home

Analog

This screenshot shows the Node-RED dashboard. It includes tabs for Layout, Site, and Theme, and a "Tabs & Links" section with entries for Home and Analog. The main content area displays the flow from the previous screenshot.



Observing JSON data via Dashboard

- Open a new internet browser and insert the RPi-add onto the browser address tab
- 192.168.1.XXX:1880/ui

Import

Export

Search flows

Configuration nodes

◀ Flows

◀ Subflows

◀ Groups

Manage palette

Settings

Keyboard shortcuts

Node-RED website

v1.1.2

Database API via Node-Red

- Download the Node-Red flow of database API from
<https://github.com/hanifr/docker-mariadb-mosquitto-rpi/blob/master/owntracks-table-api.json>
- Open Node-Red apps, add a new flow
- Navigate to setting tab, and select import

Project 2: Designing Dashboard for Location Aware

- Use the following concepts
 - Node-Red input and output nodes
 - Weather and/or Location APIs
 - Managing SQL database
 - Some creativity



Node-Red: Adding Credential

- This setup will modify the default configuration file in `/home/pi/.node-red/settings.js`
- Ensure node.js is later version (at least version 12)
- To verify; type
- `nodejs -v`
- To update to newer version (version 12), type
- `curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -`
- To install, type
- `sudo apt-get install -y nodejs`



Node-Red: Adding Credential

- Once dependency nodejs is installed, generate hash code using the following command
- `node-red admin hash-pw`