UNIVERSITI TEKNOLOGI MARA

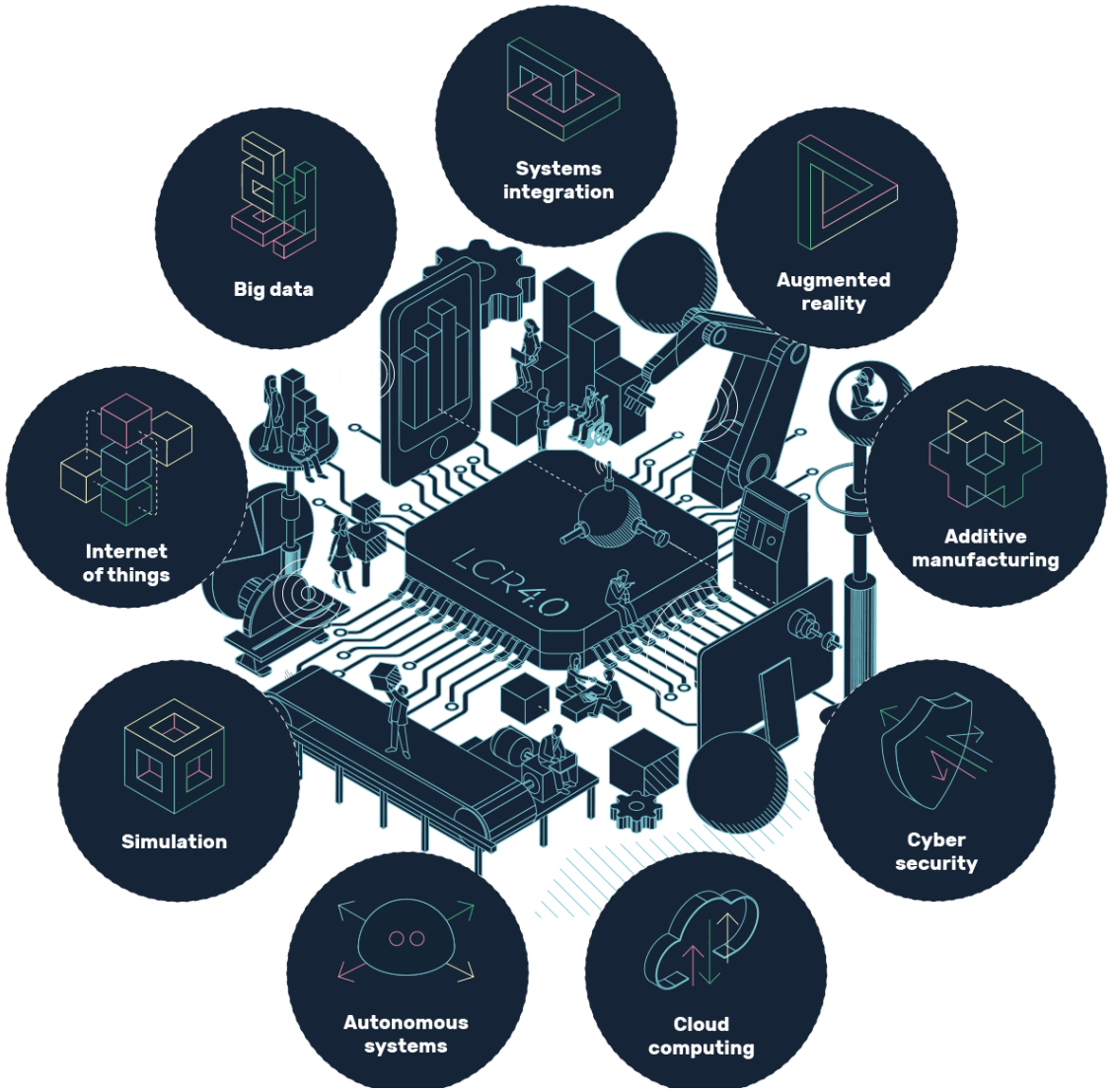
Node-Red: Flow Programming

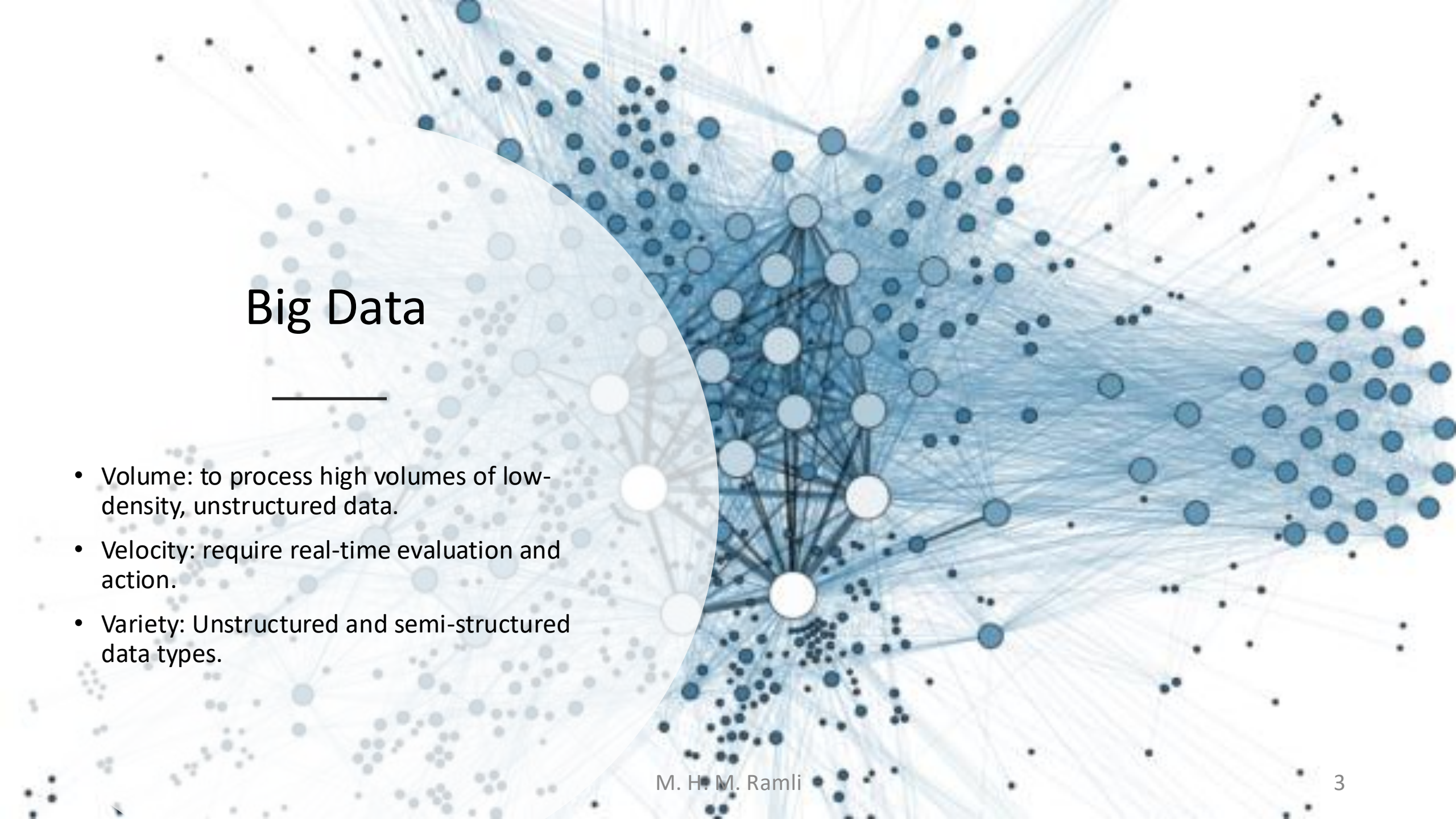
Simplify application/prototyping deployment

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.

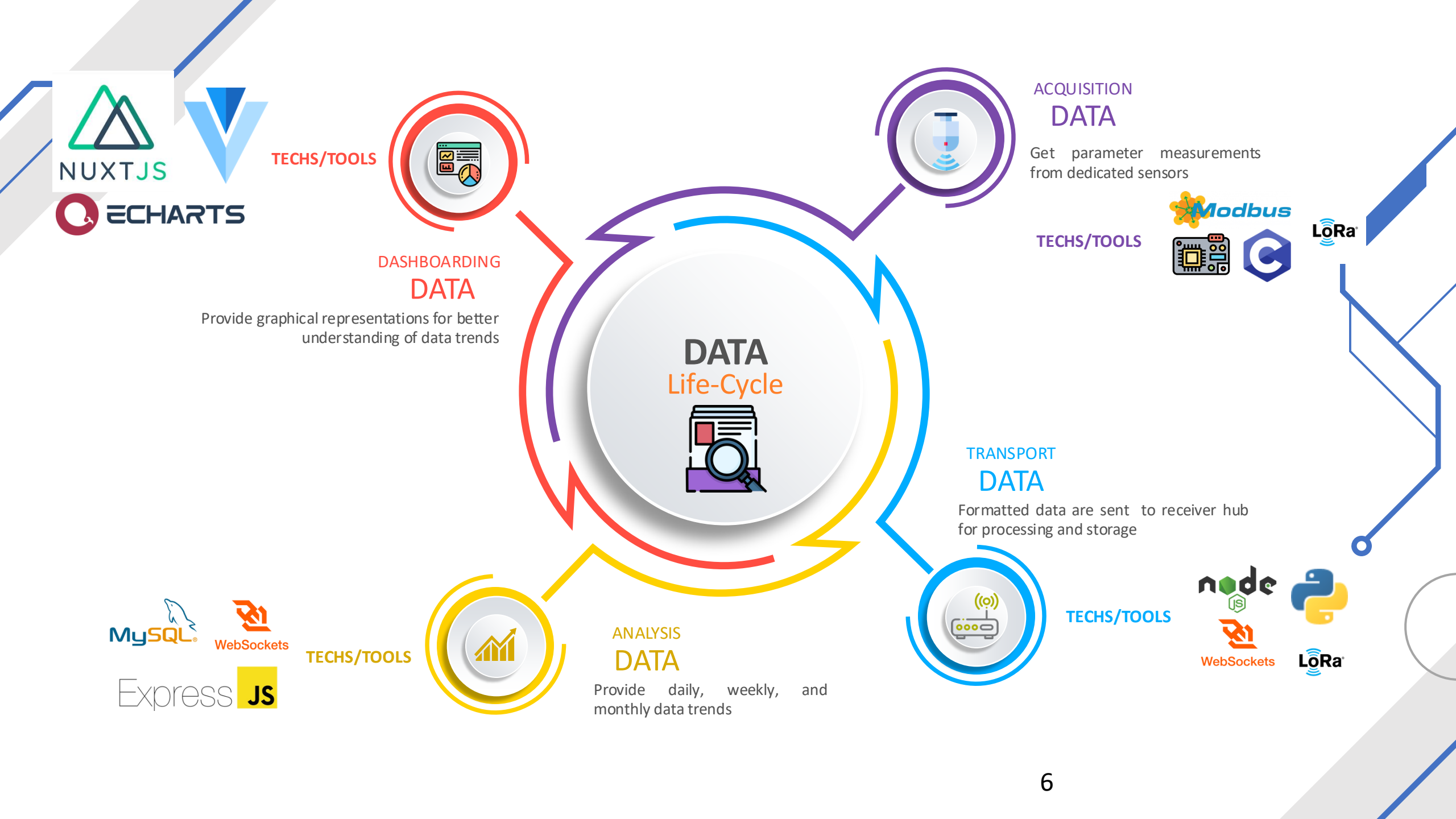
IoTs: BUILDING BLOCKS

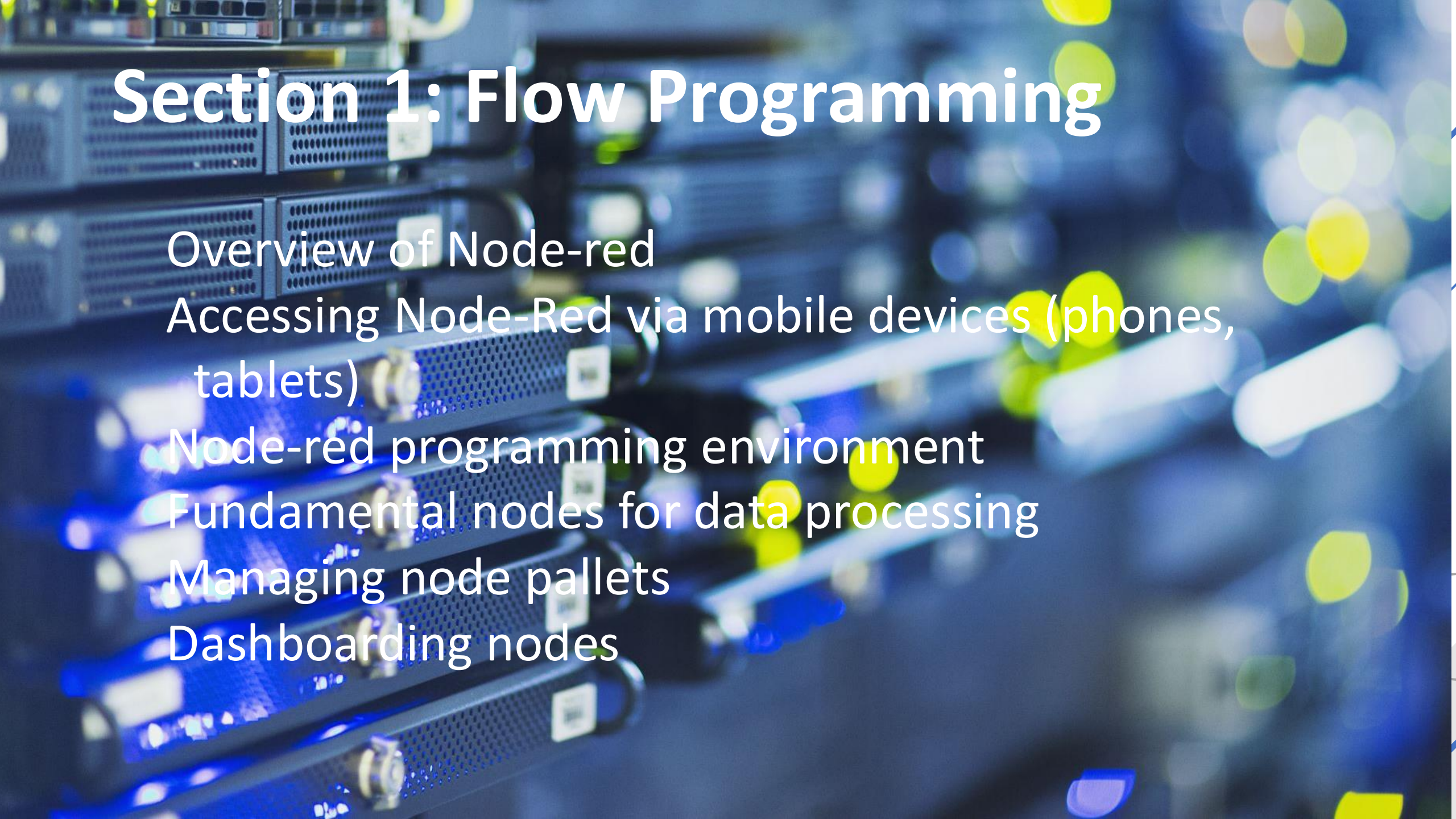
Identification and
Sensing: data acquisition.

Aggregator
(computations):
Algorithms, arithmetic.

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

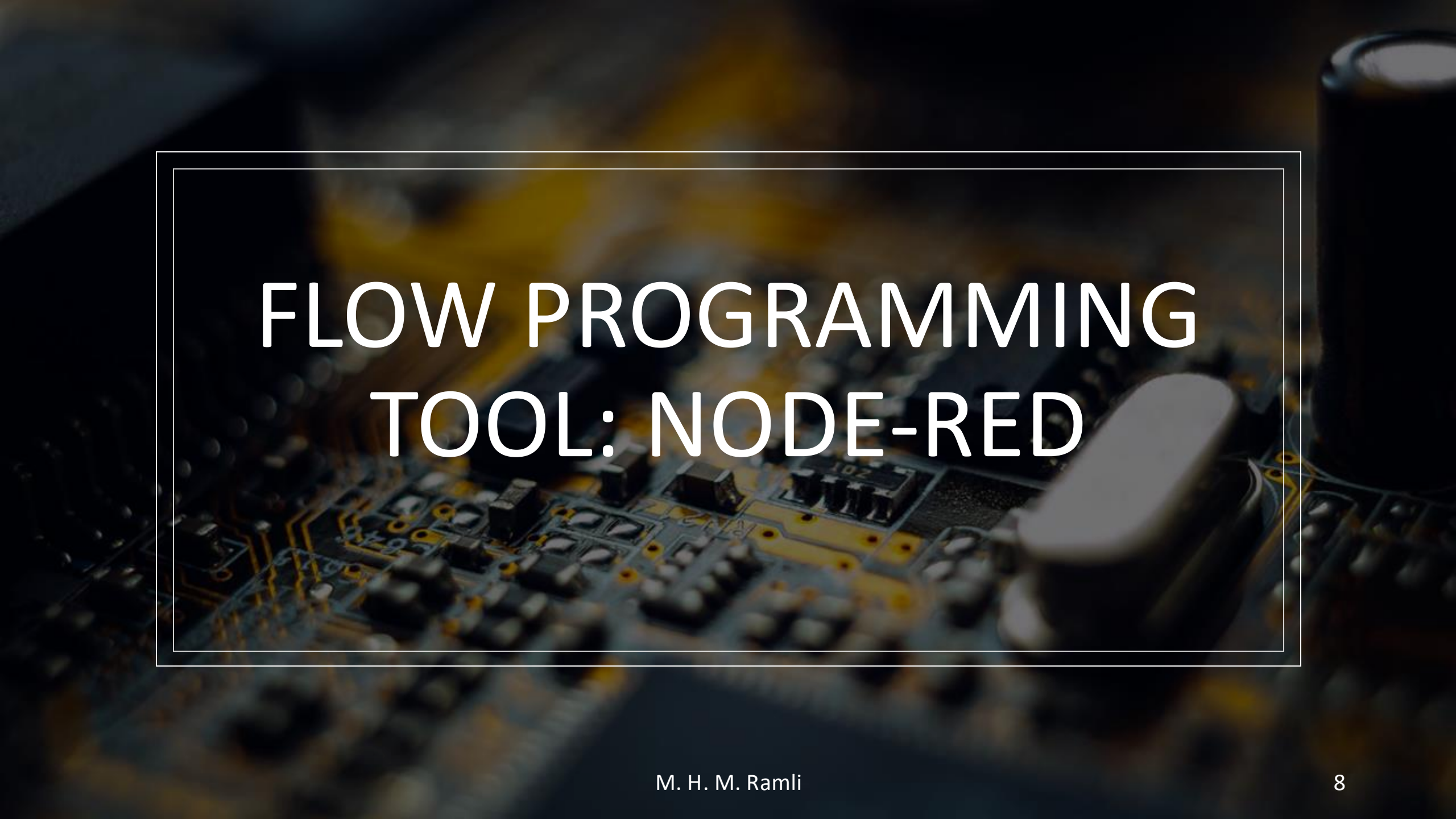
Services and Semantics
(logic): Monitoring,
Controls, Predictions.


DATA
Life-Cycle
ACQUISITION
DATA
Get parameter measurements from dedicated sensors
TECHS/TOOLS
Modbus
LoRa
TRANSPORT
DATA
Formatted data are sent to receiver hub for processing and storage
TECHS/TOOLS
WebSockets
LoRa
ANALYSIS
DATA
Provide daily, weekly, and monthly data trends
TECHS/TOOLS
MySQL
WebSockets
Express JS
DASHBOARDING
DATA
Provide graphical representations for better understanding of data trends
TECHS/TOOLS
NUXTJS
ECHARTS



Section 1: Flow Programming

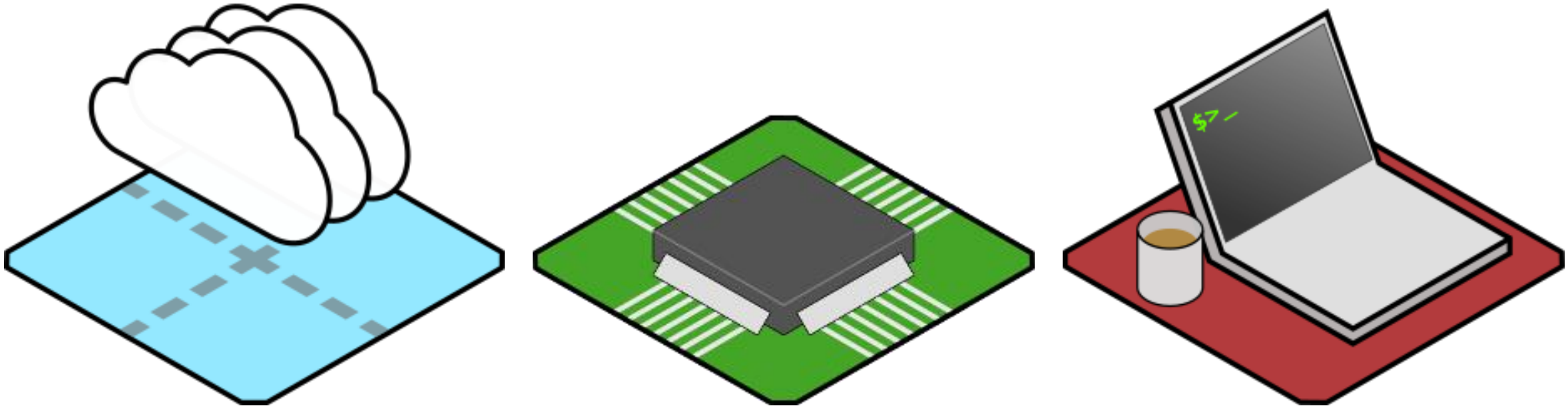
Overview of Node-red
Accessing Node-Red via mobile devices (phones, tablets)
Node-red programming environment
Fundamental nodes for data processing
Managing node pallets
Dashboarding nodes



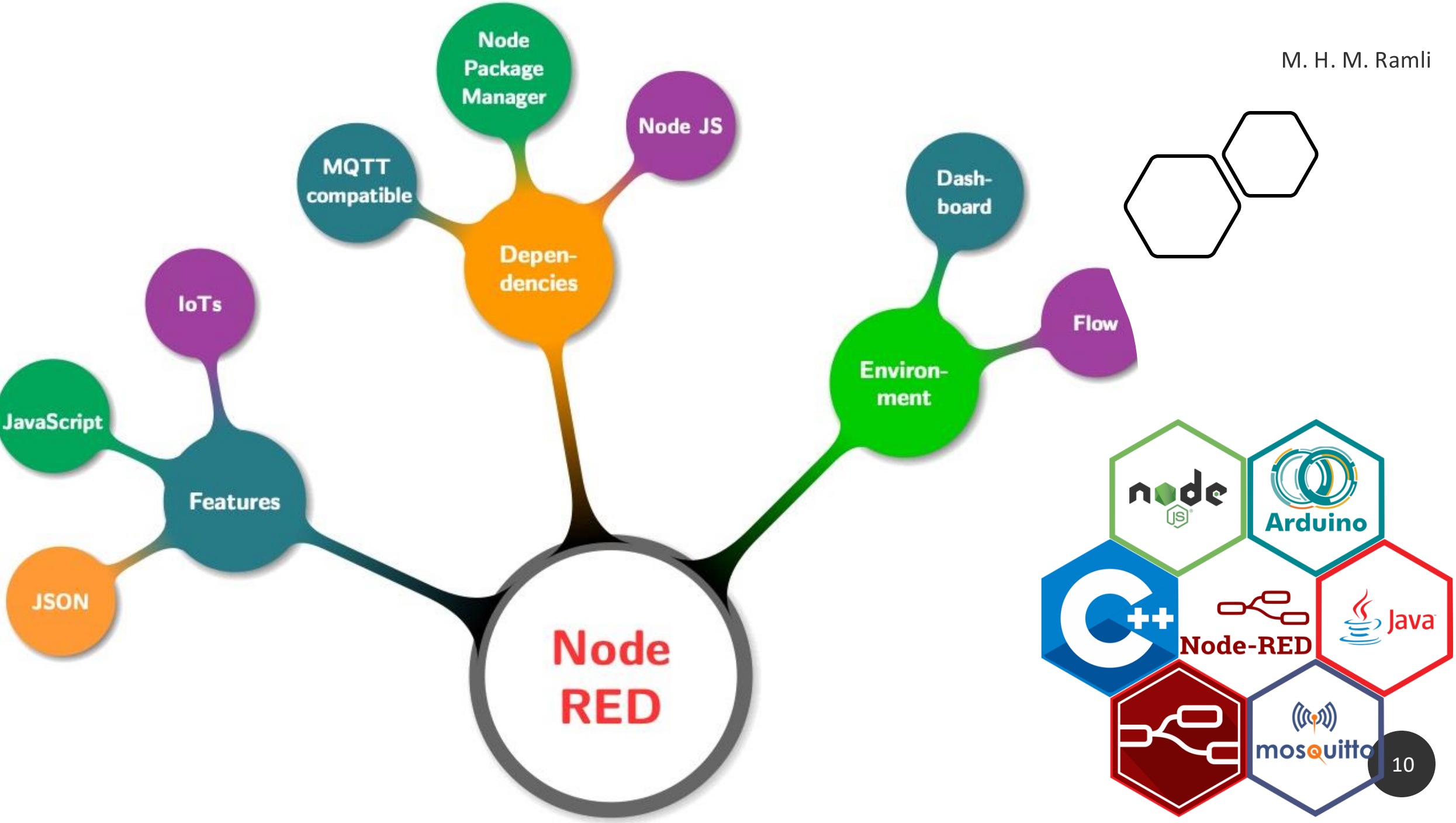
FLOW PROGRAMMING TOOL: NODE-RED

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


Node Package Manager
Node JS
MQTT compatible
Depen-dencies
Dash-board
Flow
Environ-ment
IoTs
JavaScript
Features
JSON
Node RED
node
Arduino
C++
Node-RED
Java
mosquitto

LASK 4.0 ENVIRONMENT

LOCAL ANAESTHESIA SIMULATOR KIT 4.0

GOAL

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

STRATEGY

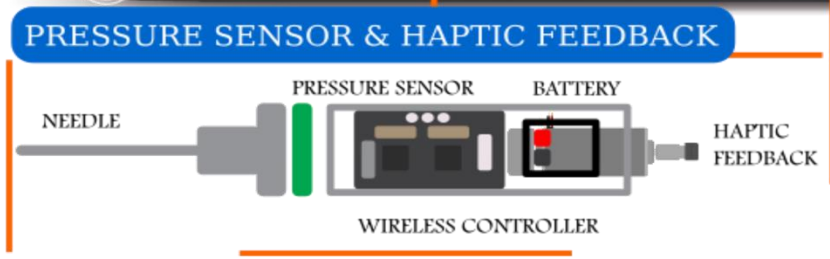
Integration of IoT, mechatronics, & IANB

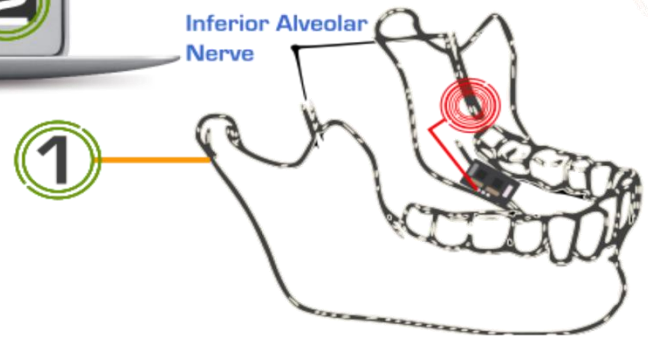
[1]: Provide evaluation of position accuracy as an injection is made.

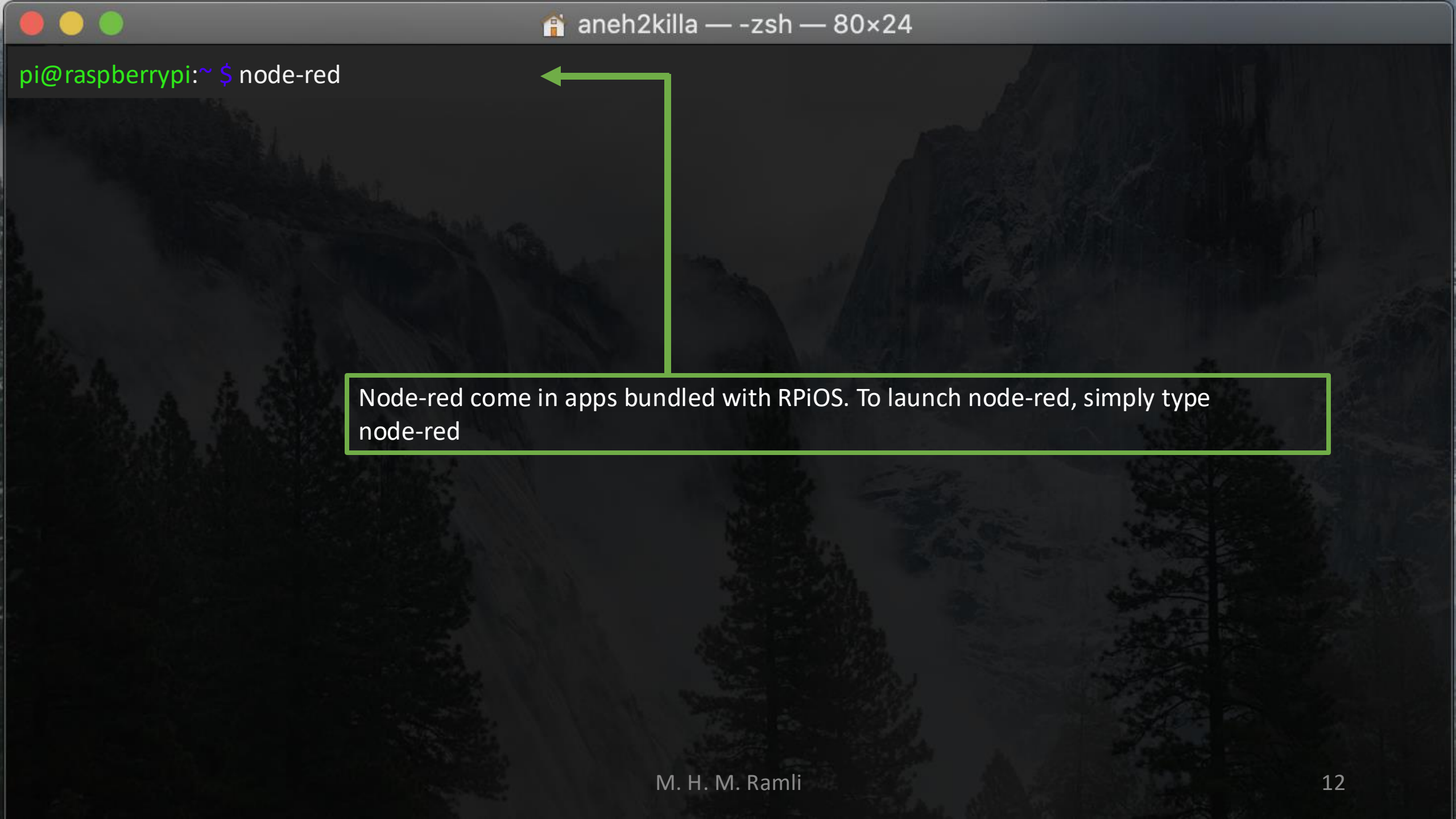
[2]: Monitor pressure applied during an injection

[3]: Response to the trainee via haptic feedback once inappropriate pressure is applied

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





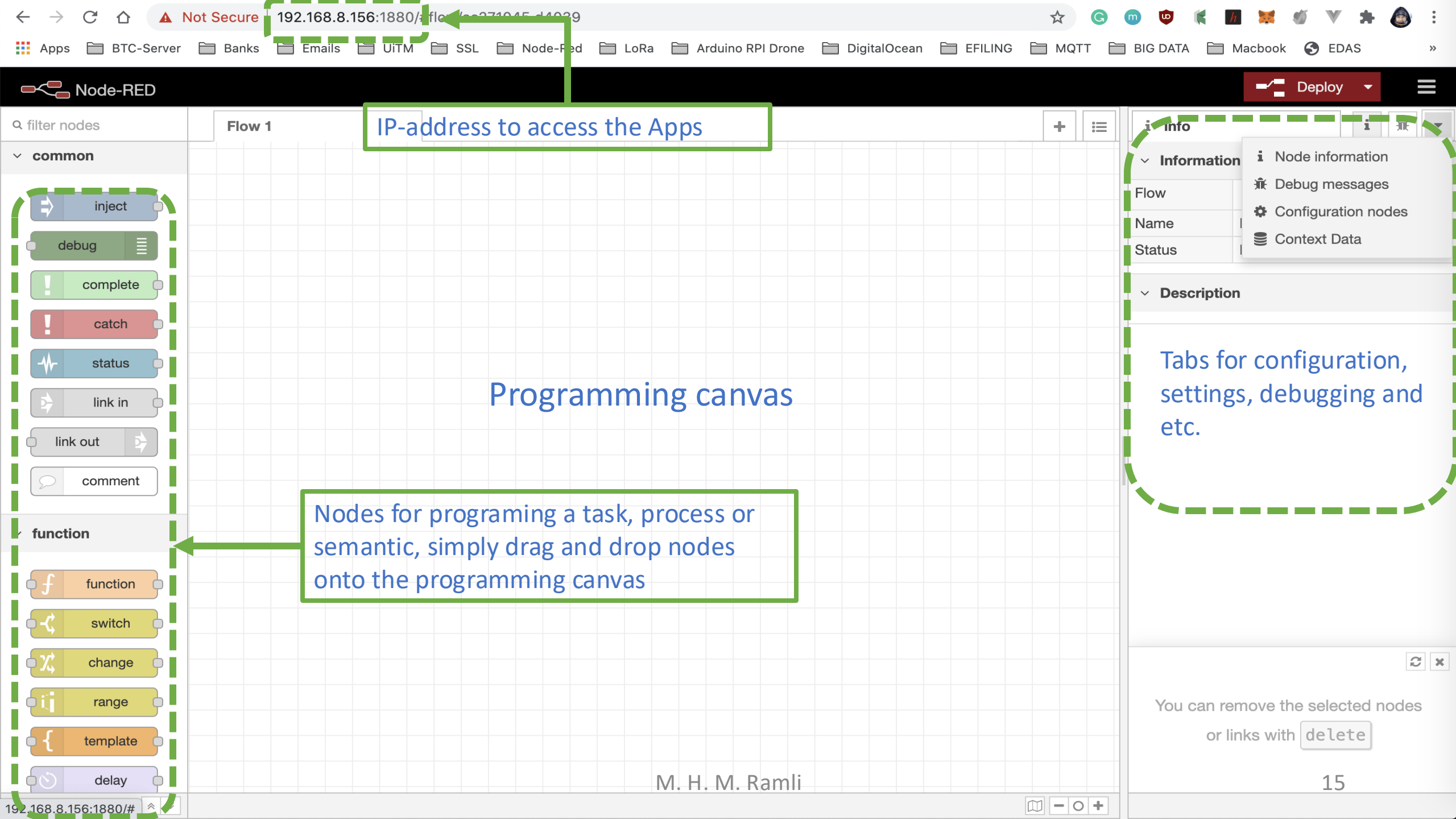

aneh2killa — -zsh — 80×24
pi@raspberrypi:~ $ node-red
Node-red come in apps bundled with RPiOS. To launch node-red, simply type node-red

aneh2killa — -zsh — 80×24

```
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)


Not Secure
192.168.8.156:1880/
Apps
BTC-Server
Banks
Emails
UiTM
SSL
Node-Red
LoRa
Arduino RPI Drone
DigitalOcean
EFILING
MQTT
BIG DATA
Macbook
EDAS
Node-RED
Deploy
IP-address to access the Apps
filter nodes
Flow 1
common
inject
debug
complete
catch
status
link in
link out
comment
function
function
switch
change
range
template
delay
Programming canvas
Nodes for programing a task, process or semantic, simply drag and drop nodes onto the programming canvas
info
Information
Node information
Debug messages
Configuration nodes
Context Data
Flow
Name
Status
Description
Tabs for configuration, settings, debugging and etc.
You can remove the selected nodes or links with delete
M. H. M. Ramli
15
192.168.8.156:1880/#

Node-Red: Basic Nodes

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

Debug: Versatile configurable output node: For debug purpose.

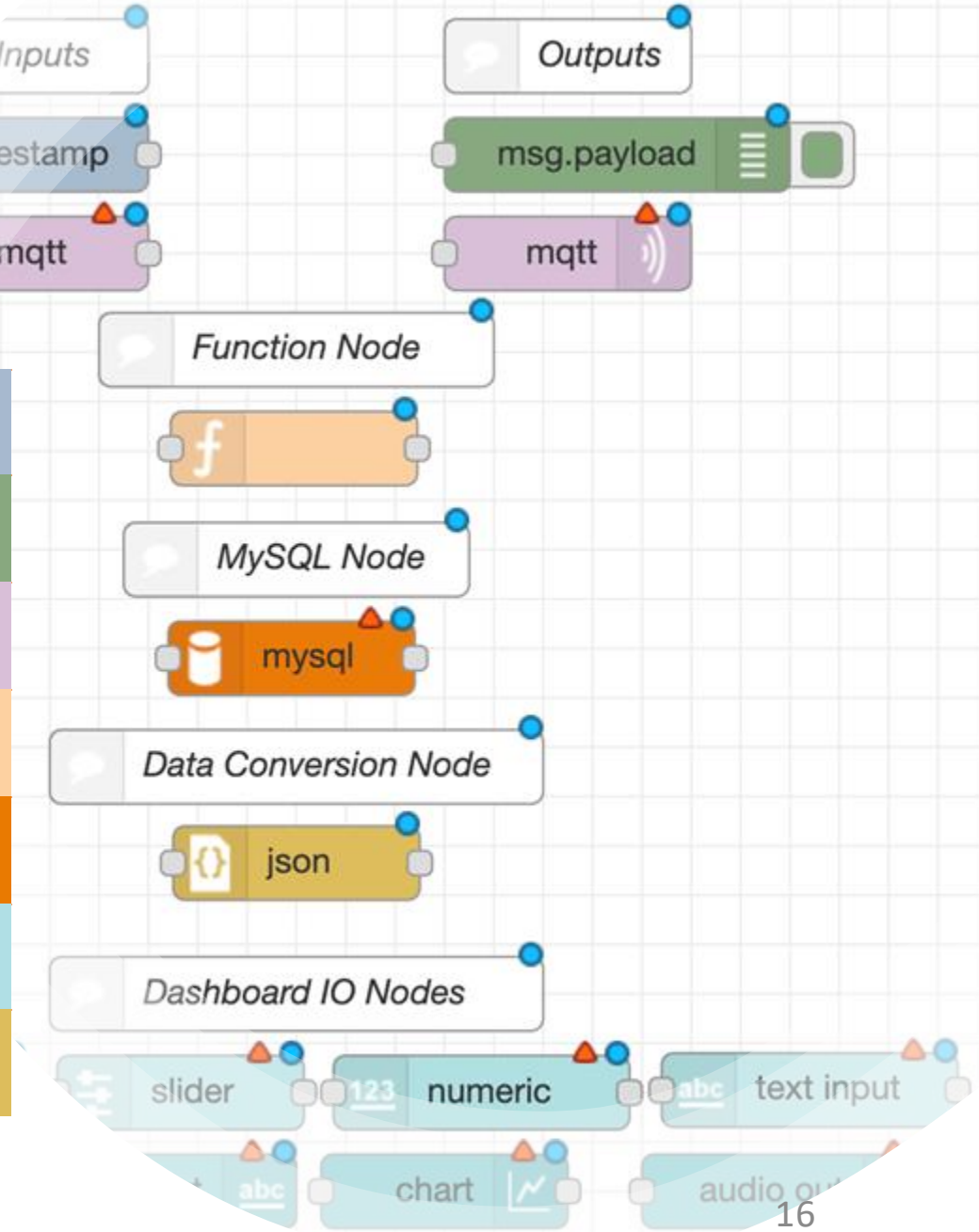
MQTT: Data Broker node: Subscribe/publish data from/to a database, hardware, website, and other API.

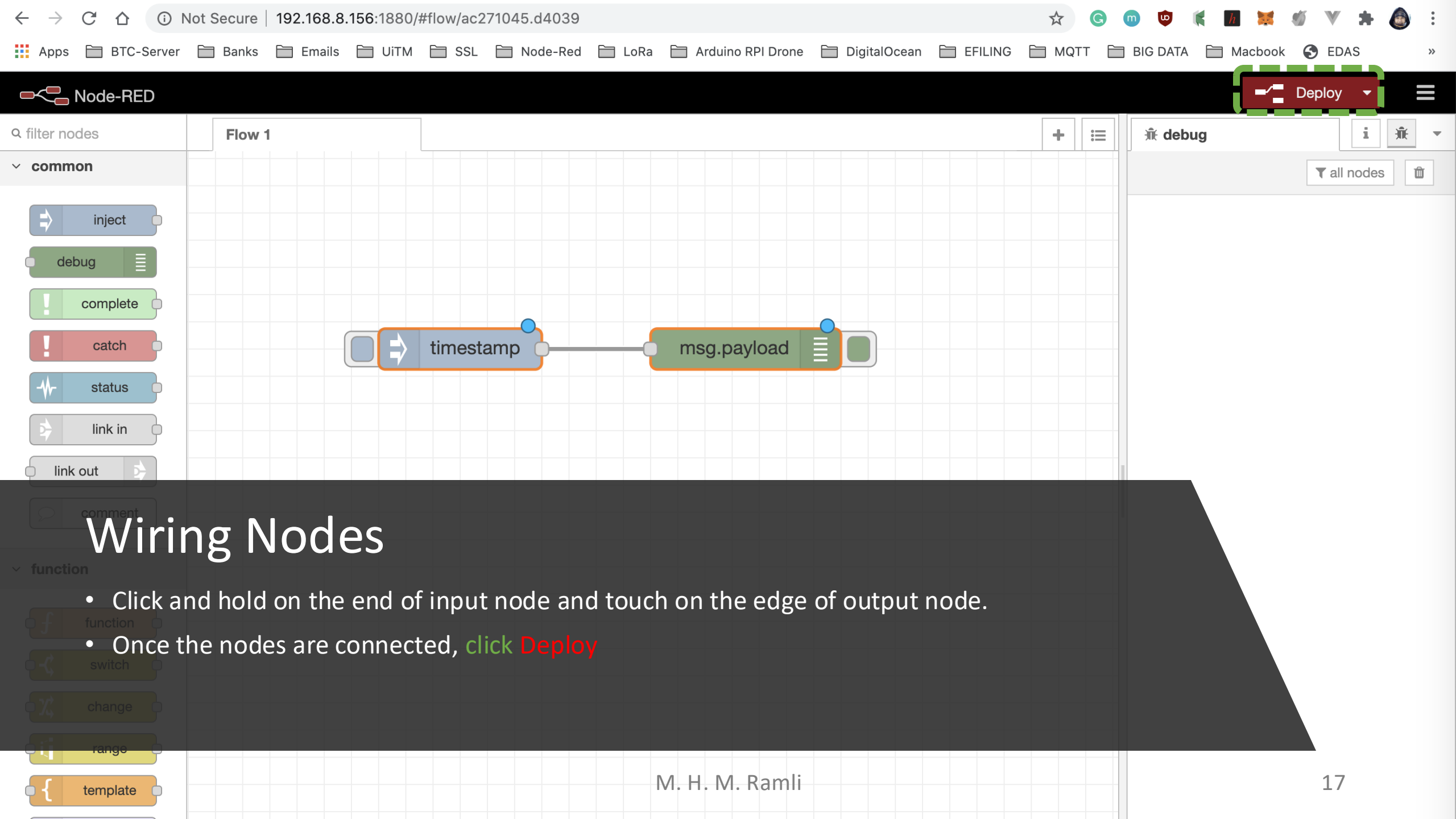
Function: Function node: Use JavaScript to process incoming data (sort, filter, count, statistical analysis, etc.).

MySQL: Manage databases based on Sequence Query Language.

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

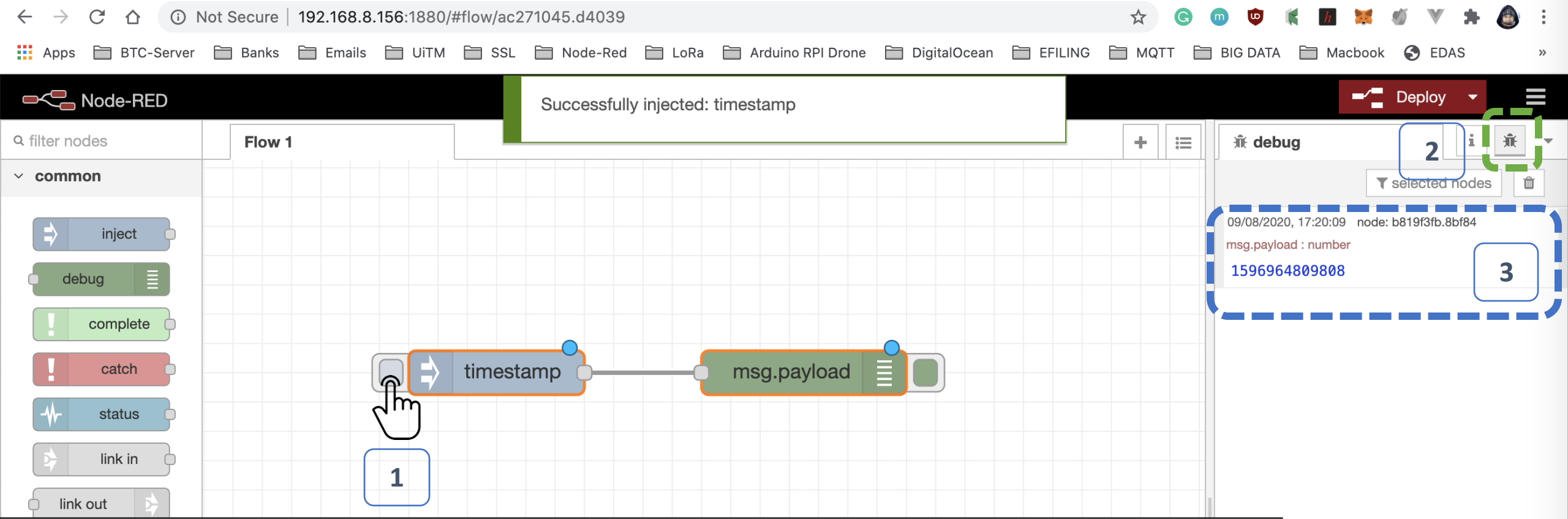
JSON: Converter node to suit flexible data format.





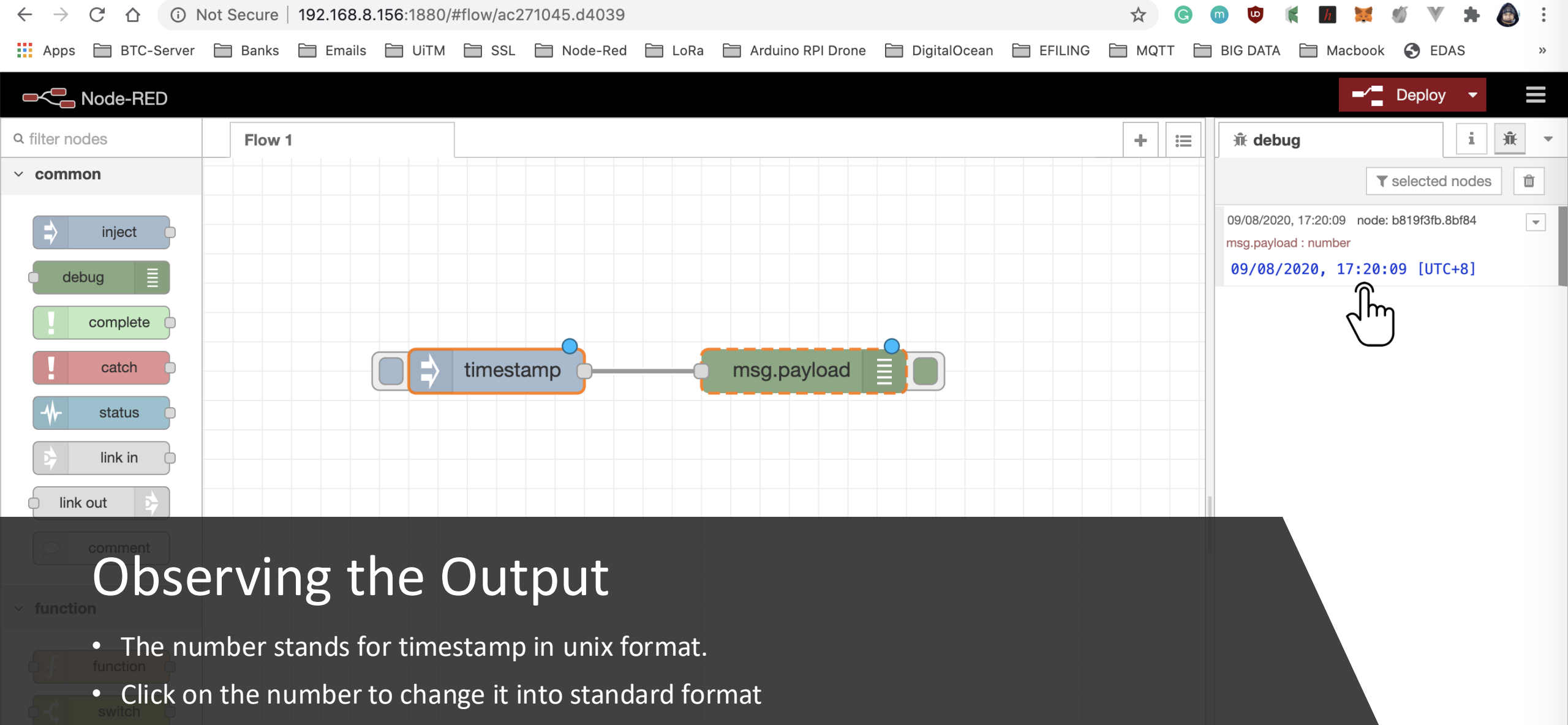
Wiring Nodes

- 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?

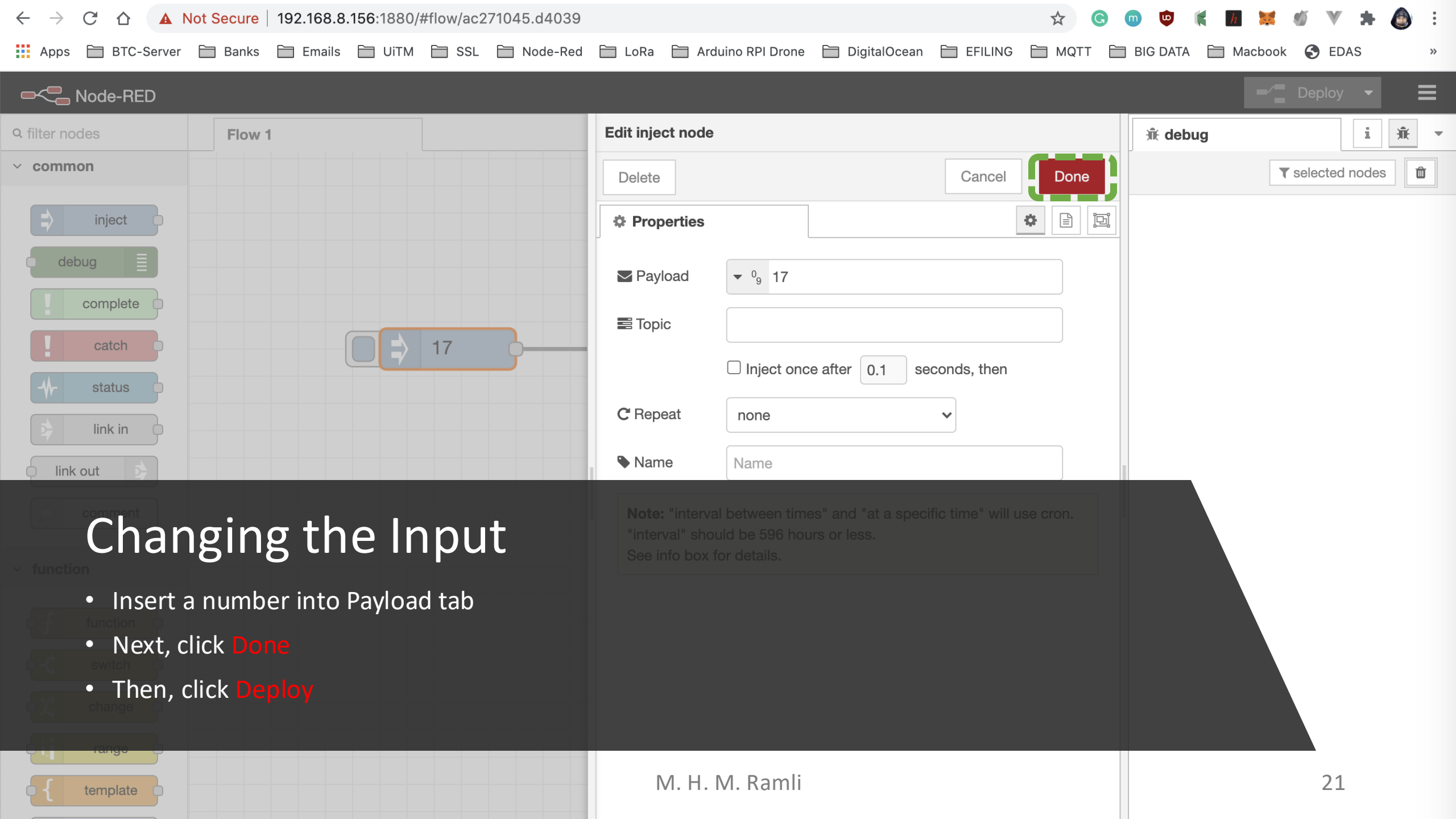


Observing the Output

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

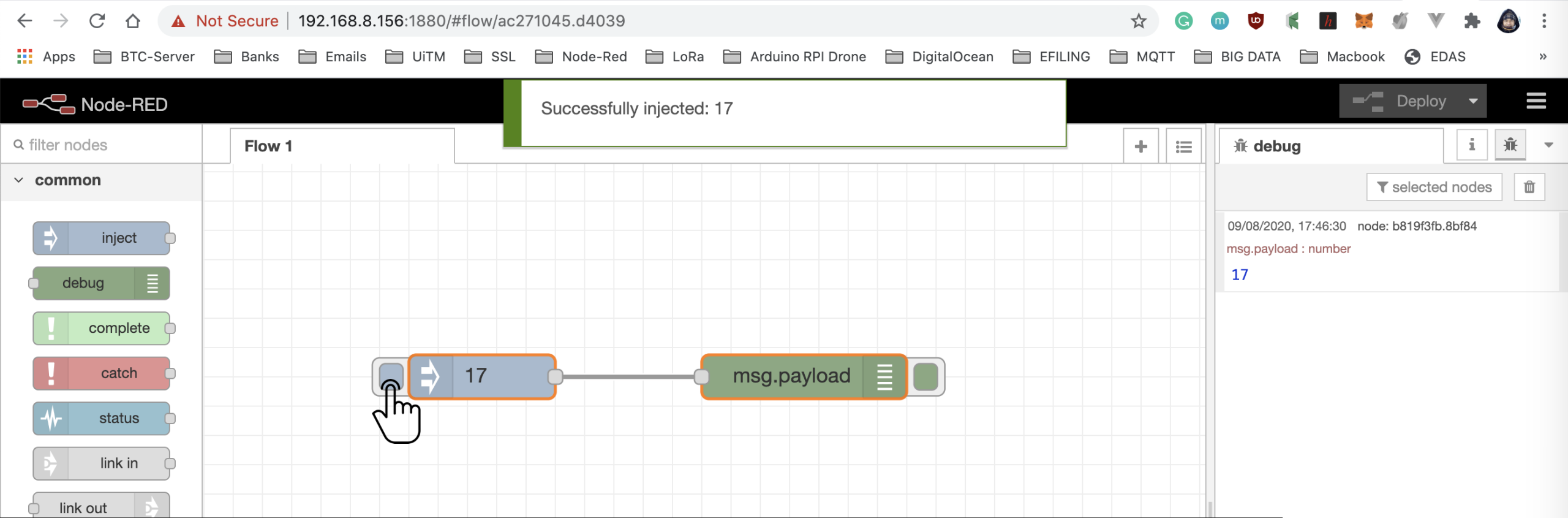
Changing the Input

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



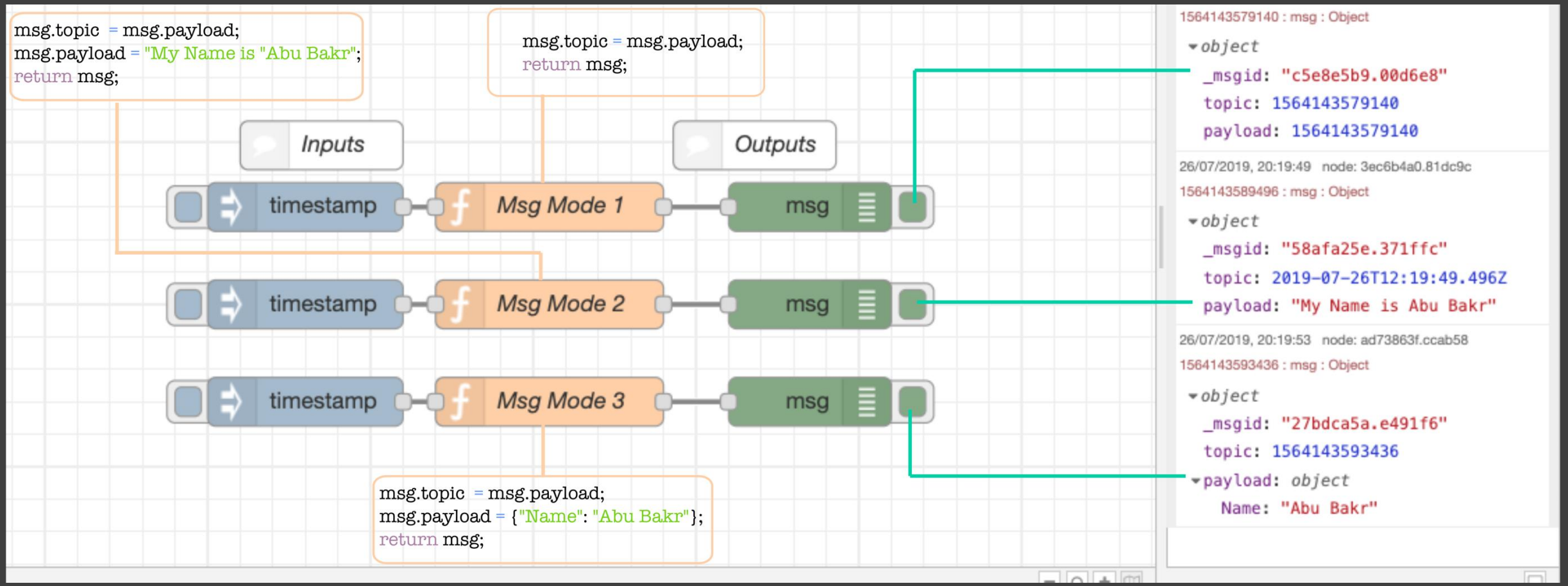
Changing the Input

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



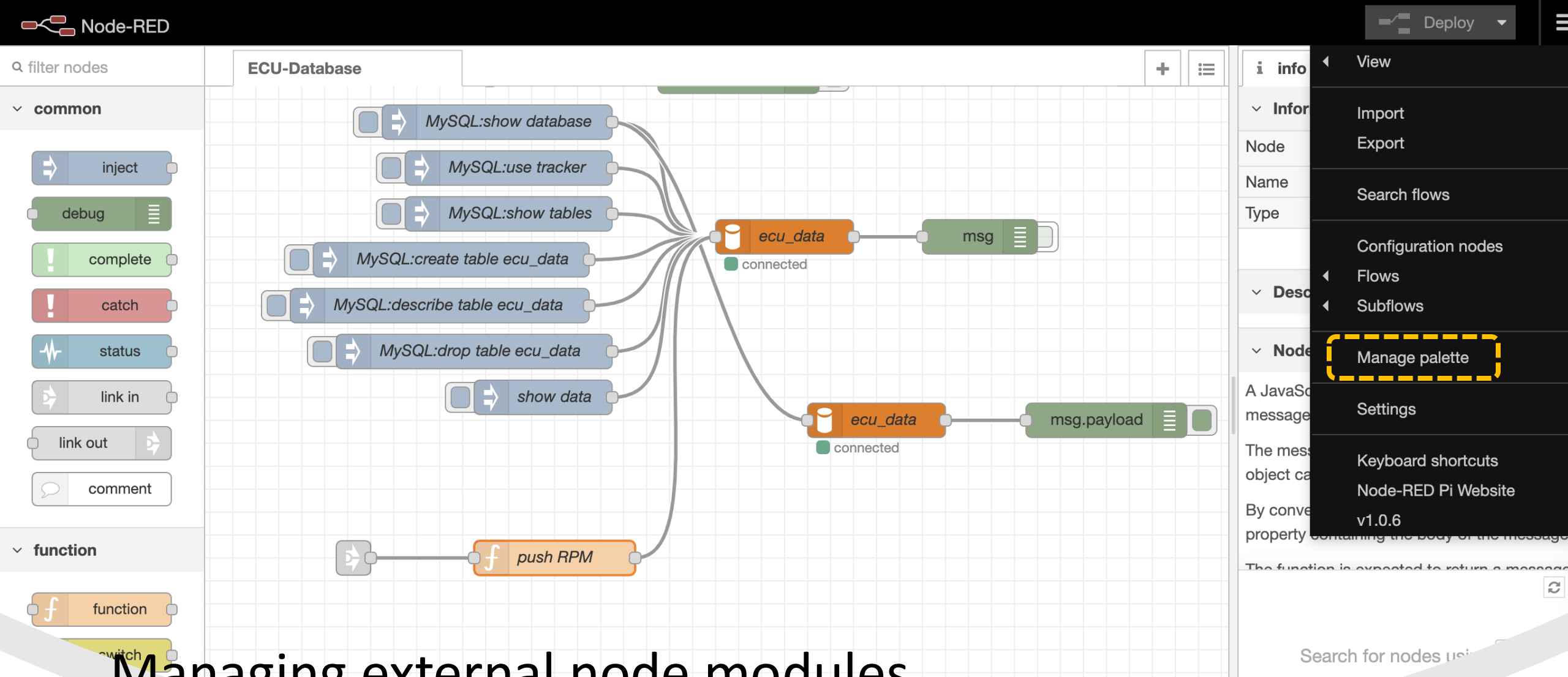
Observing the new Output

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



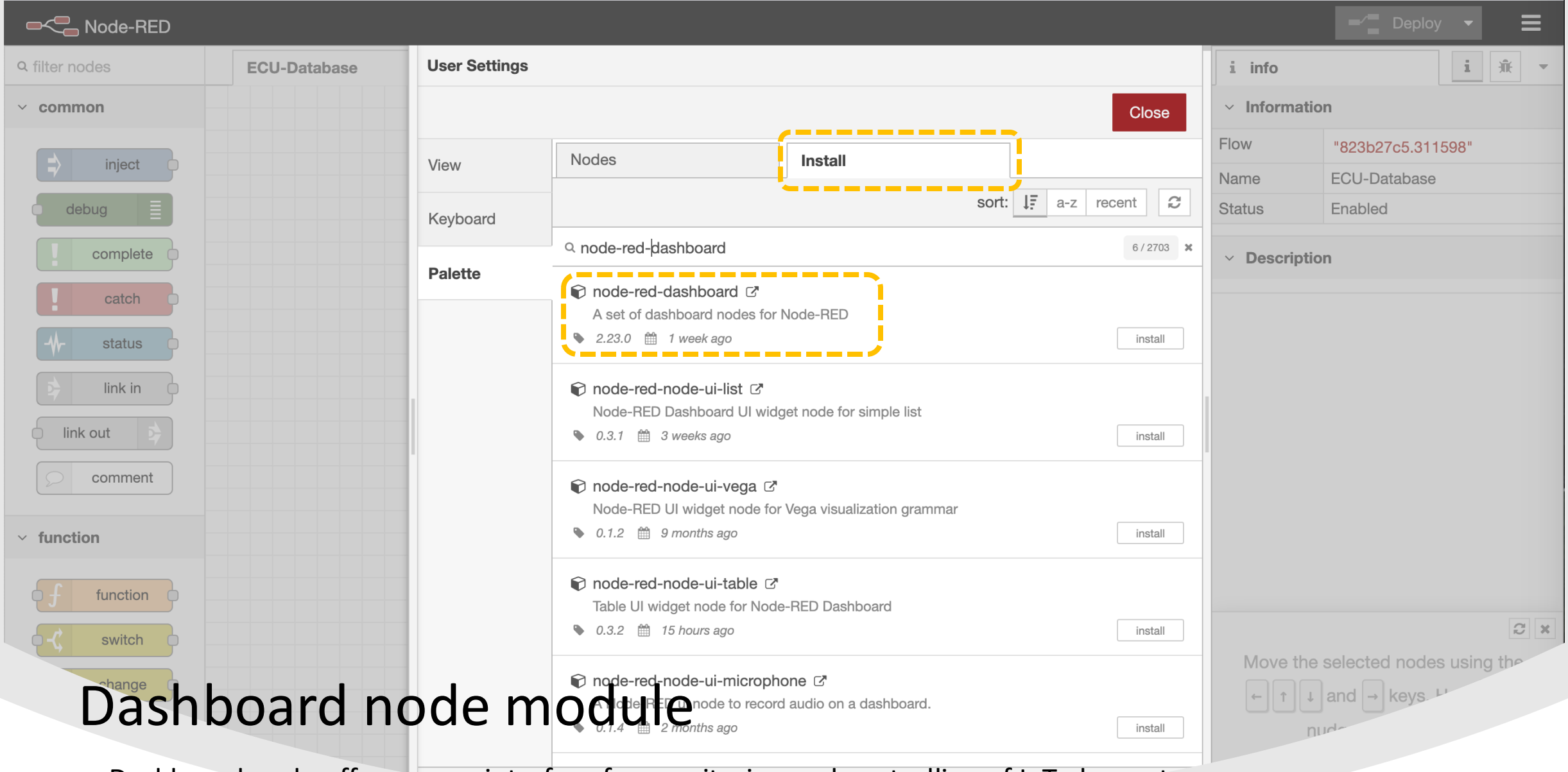
Using Function Node

- 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).



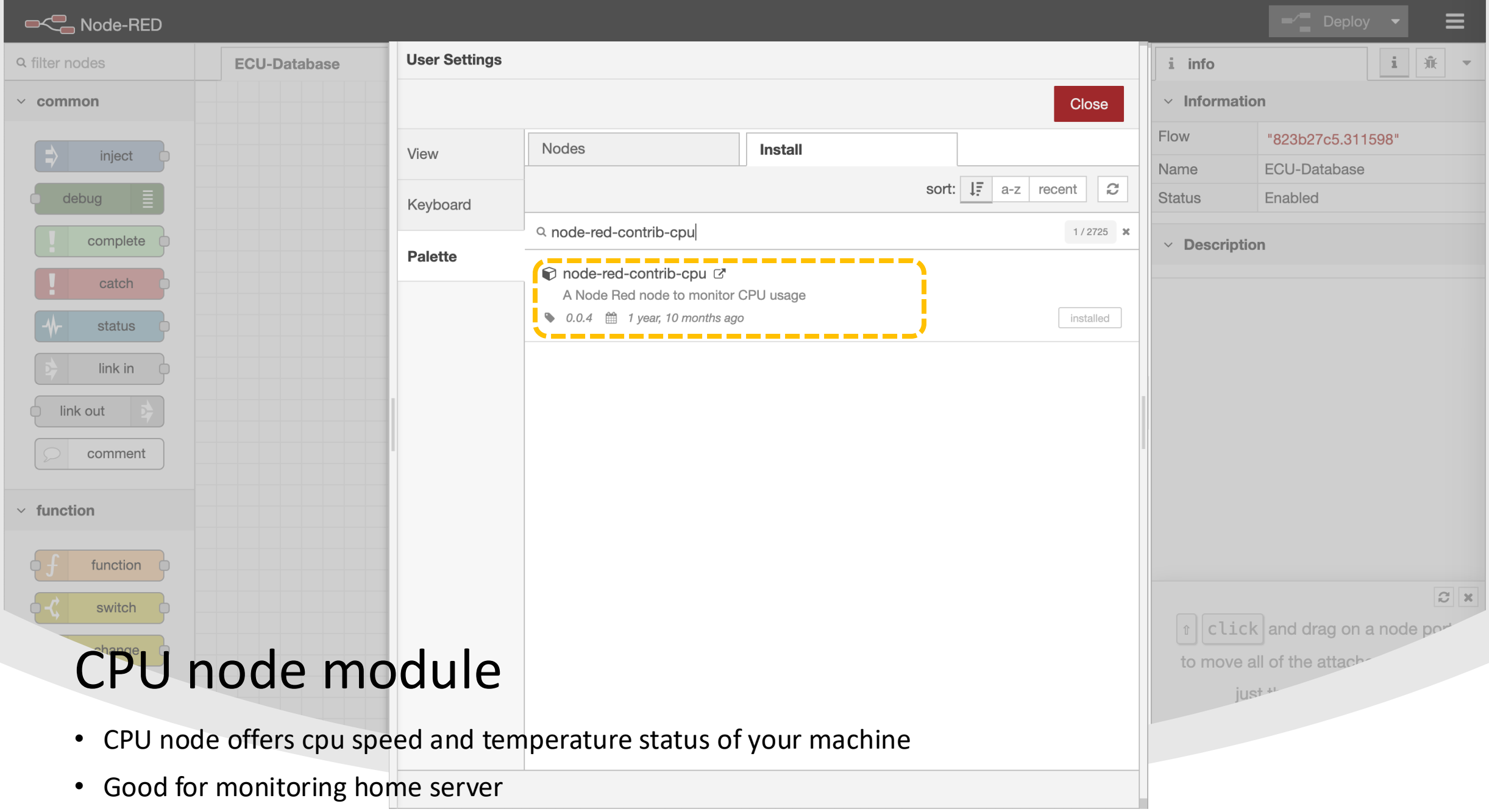
Managing external node modules

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



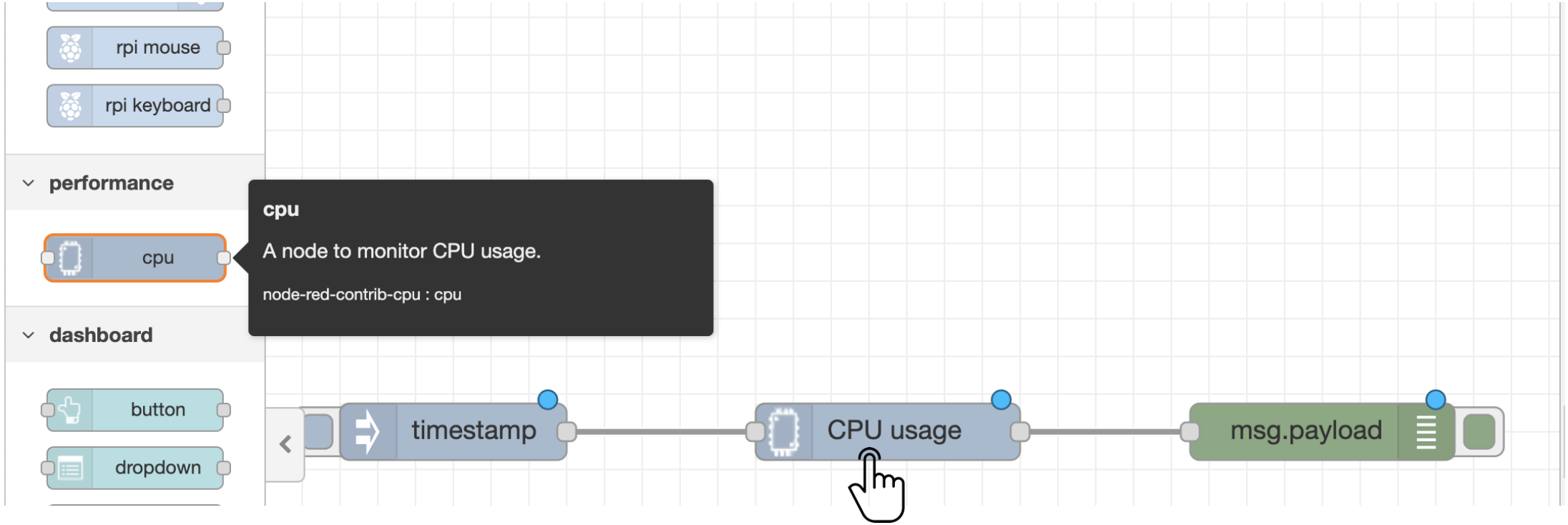
Dashboard node module

- 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



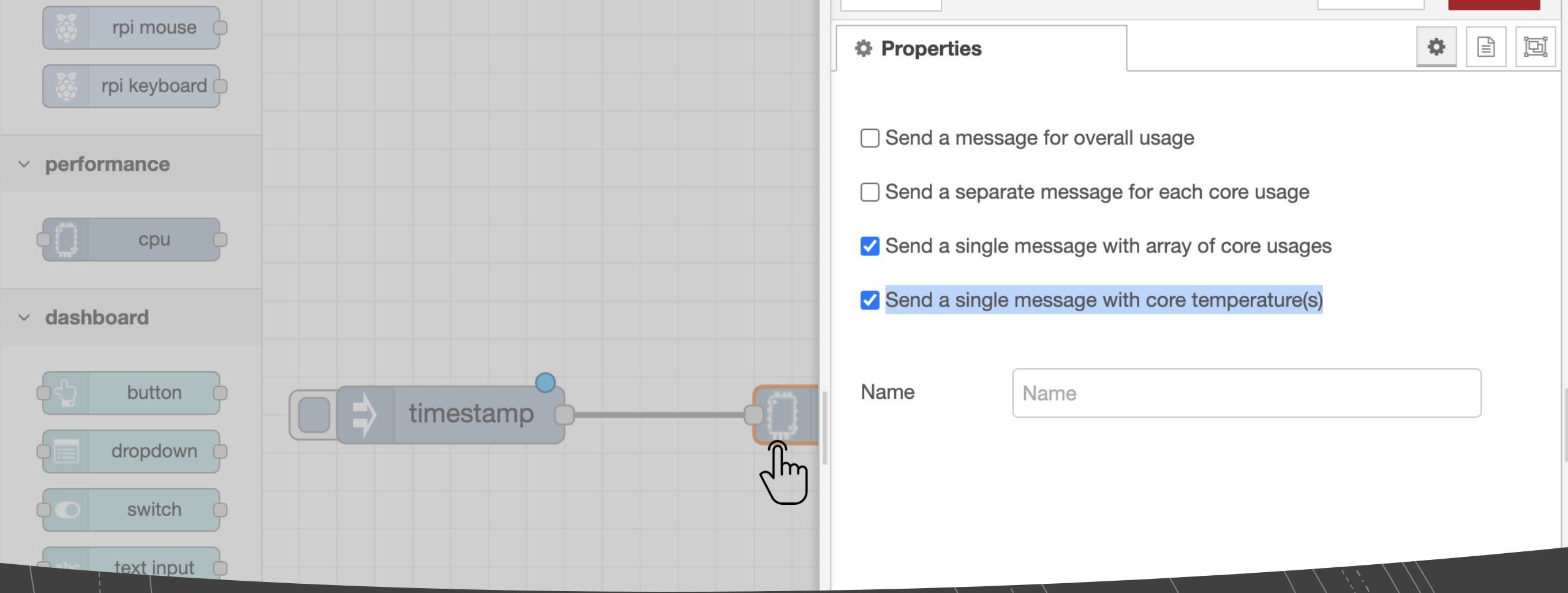
CPU node module

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



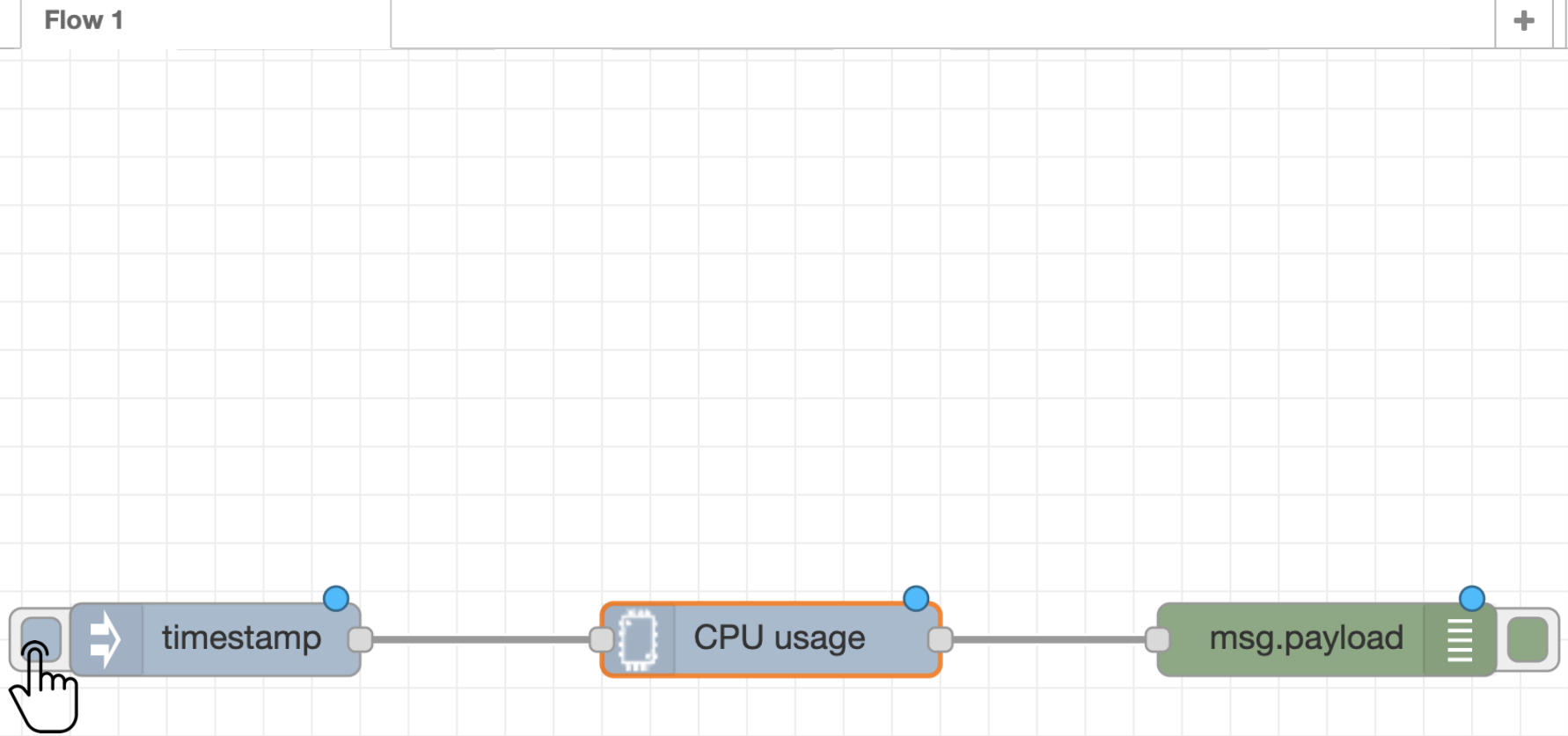
CPU Monitoring Apps

- 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)



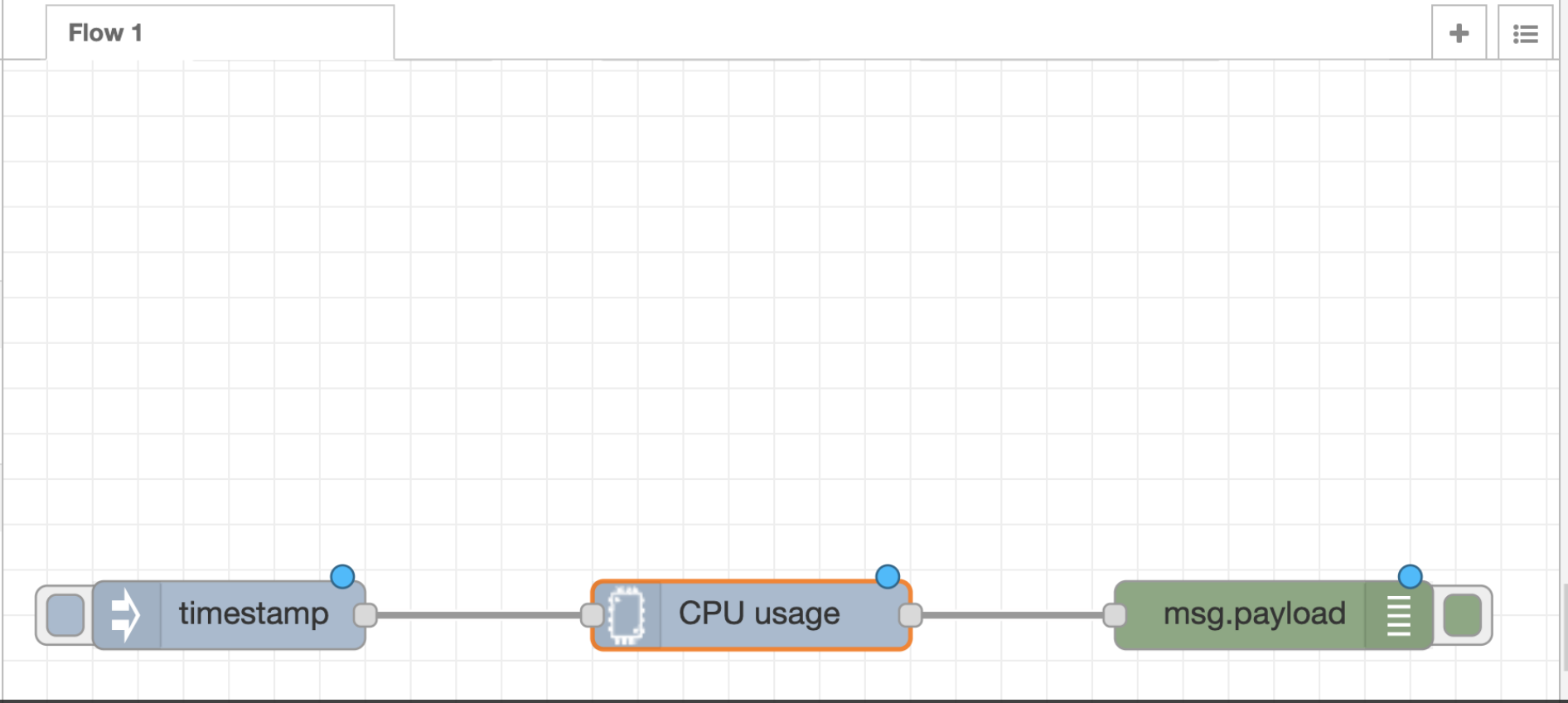
CPU Monitoring Apps

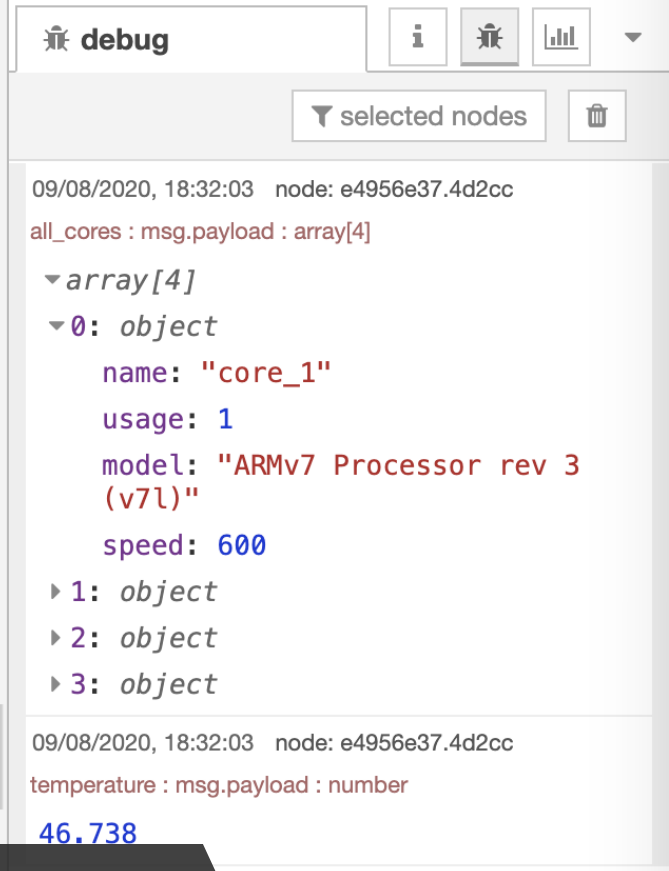
- Double click on cpu node and tick the following
  - Send a single message with array of core usages
  - Send a single message with core temperature(s)



CPU Monitoring Apps: Data Extraction

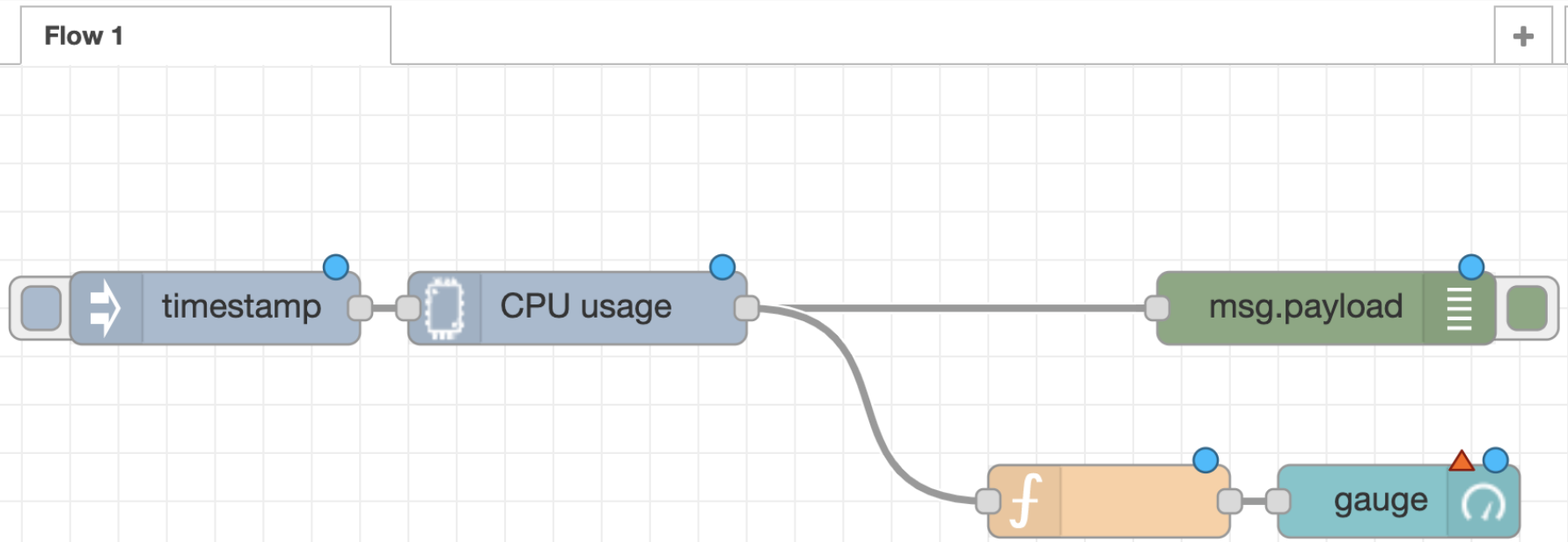
- Click the input node and observe the output on the debug tab, two types of data, one is object, the other is just a value
- The object provides total cpu cores available on RPi (click on object to see the details)
- The value tells us the current temperature of cpu die.





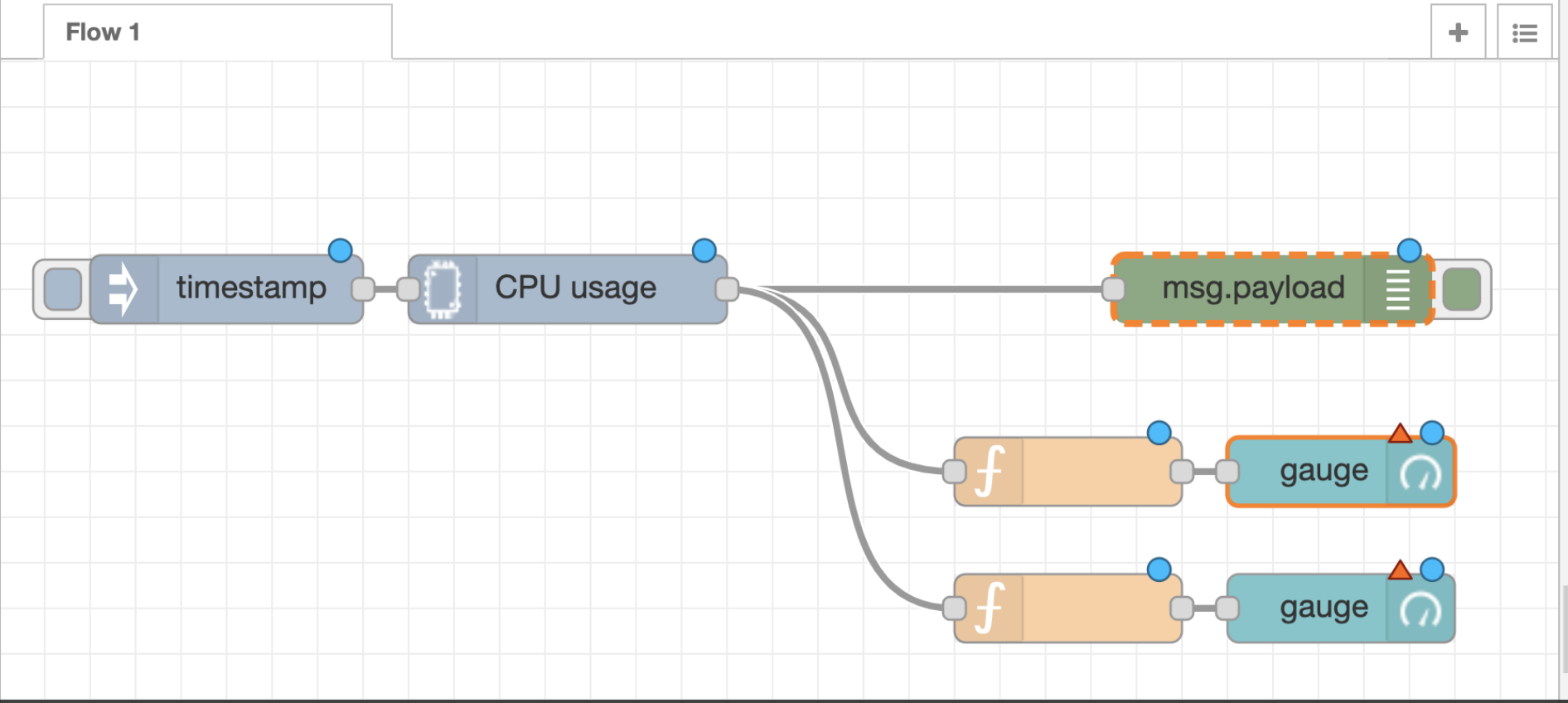
CPU Monitoring Apps: Data Extraction

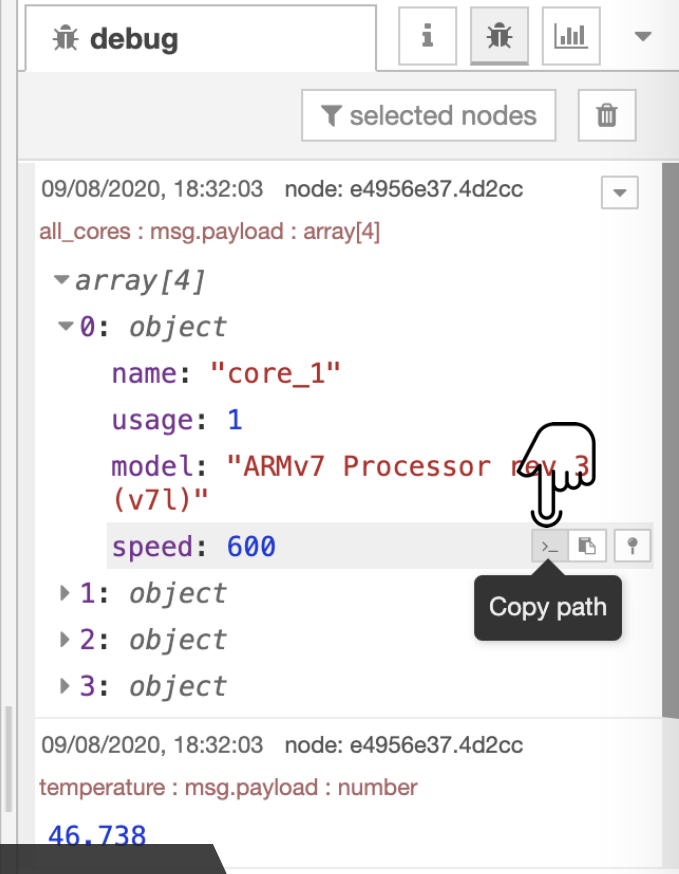
- In the debug tab, there are two types of data, one is object, the other is just a value
- There are 4 cpu cores in RPi, each current speed = 600MHz
- The current temperature = 46.738C.



CPU Monitoring Apps: Dashboarding

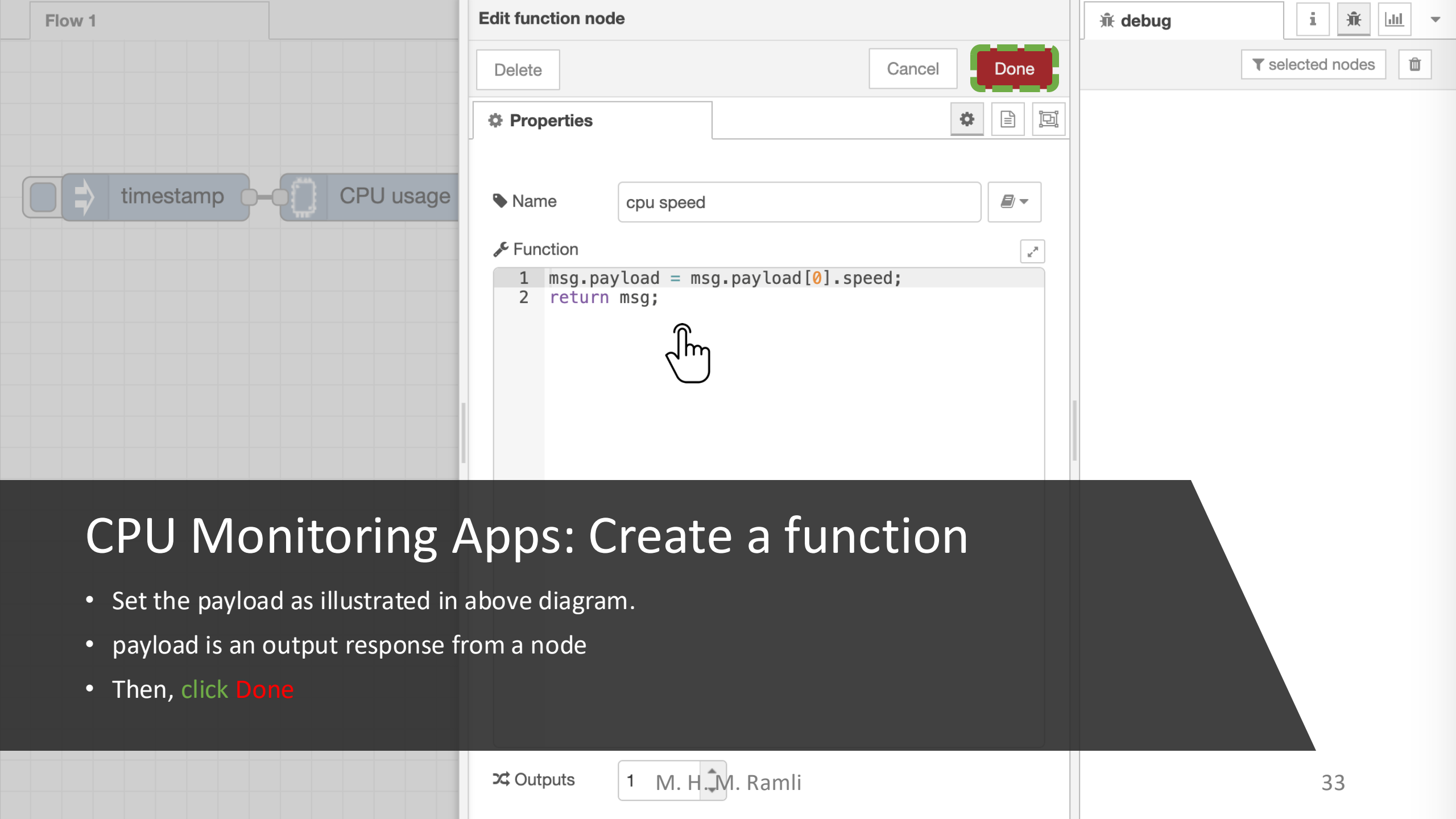
- Display the speed and temperature on dashboard
- Drag a function and gauge nodes, and wire them as per above diagram





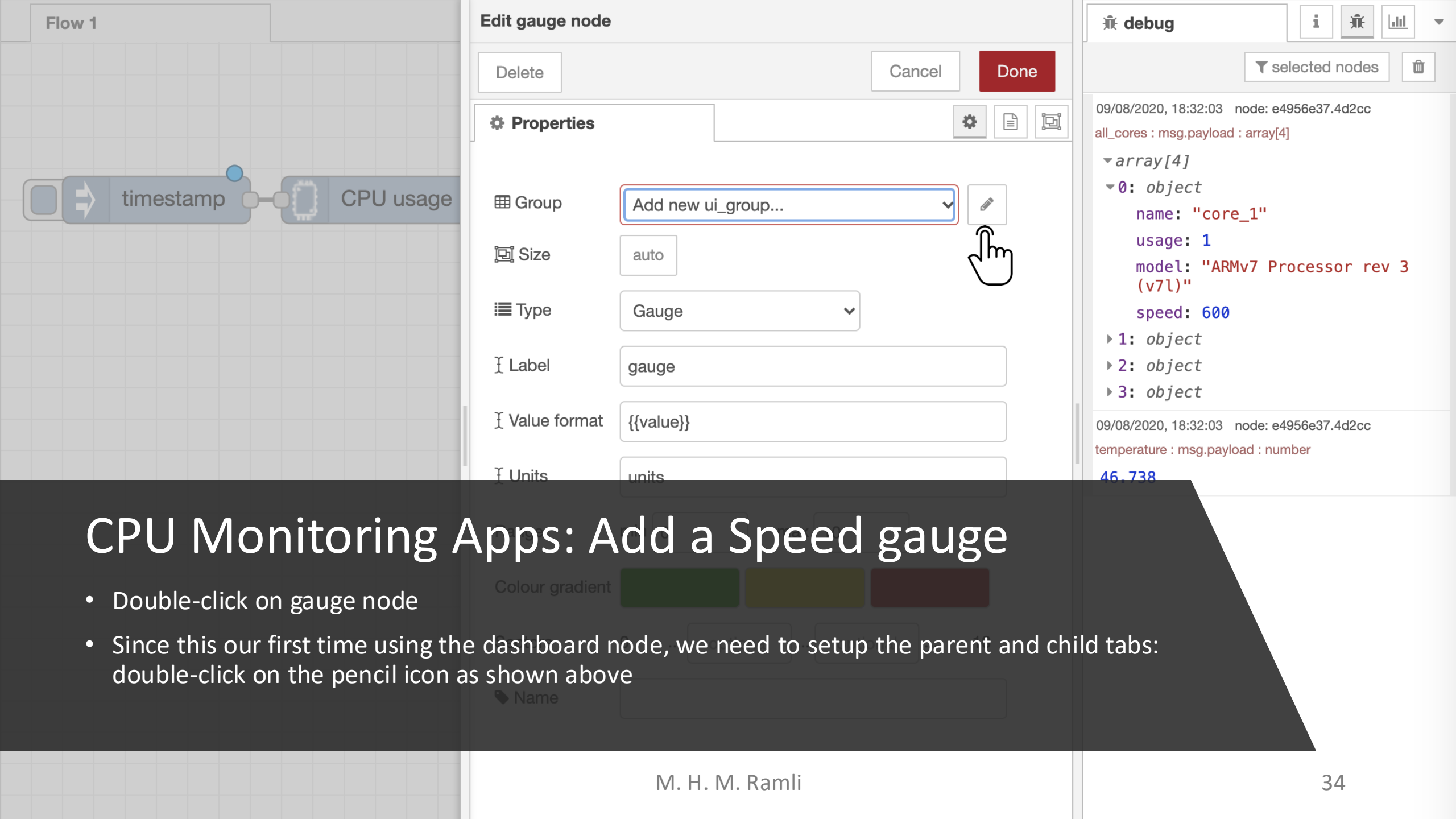
CPU Monitoring Apps: Create a function

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



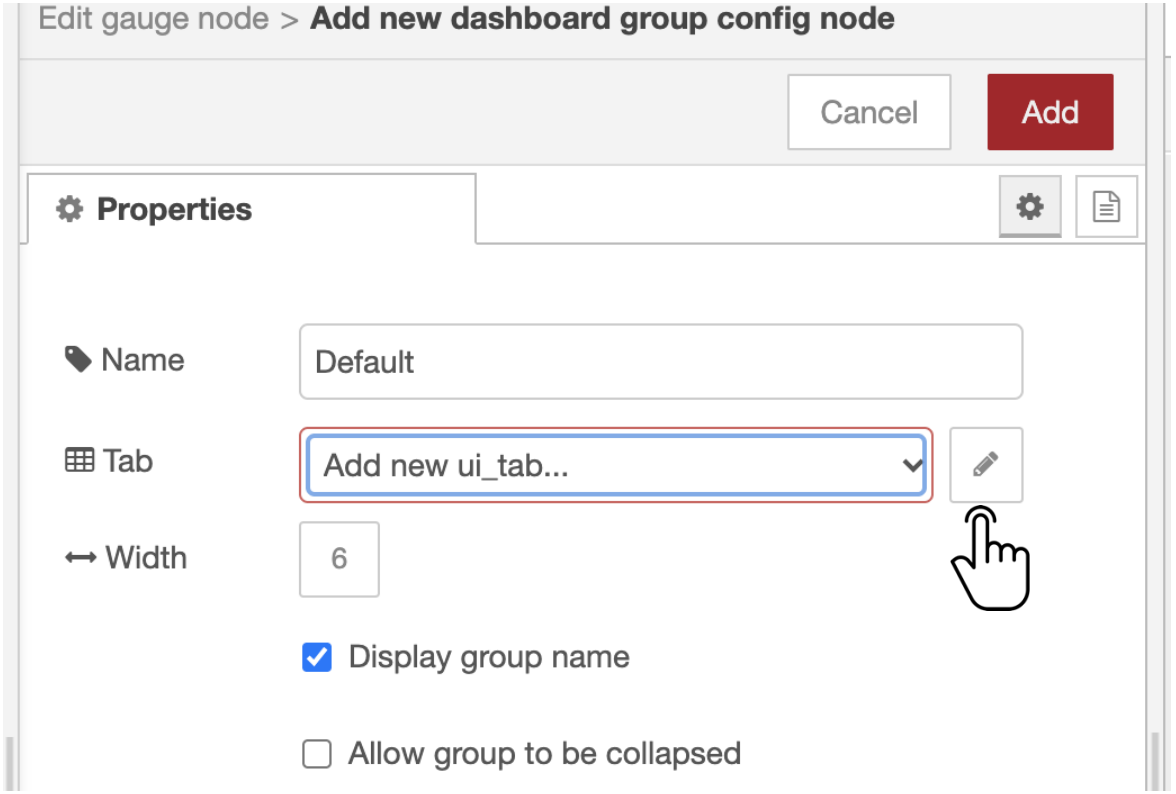
CPU Monitoring Apps: Create a function

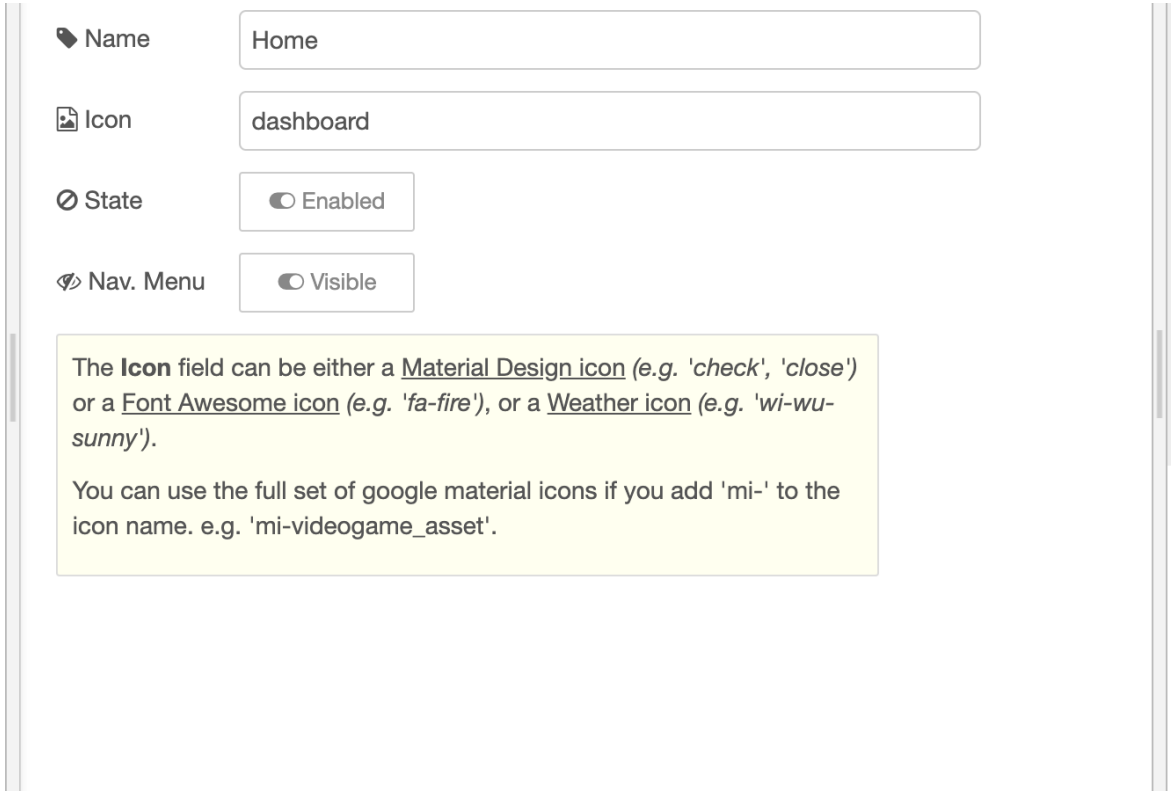
- Set the payload as illustrated in above diagram.
- payload is an output response from a node
- Then, click Done



CPU Monitoring Apps: Add a Speed gauge

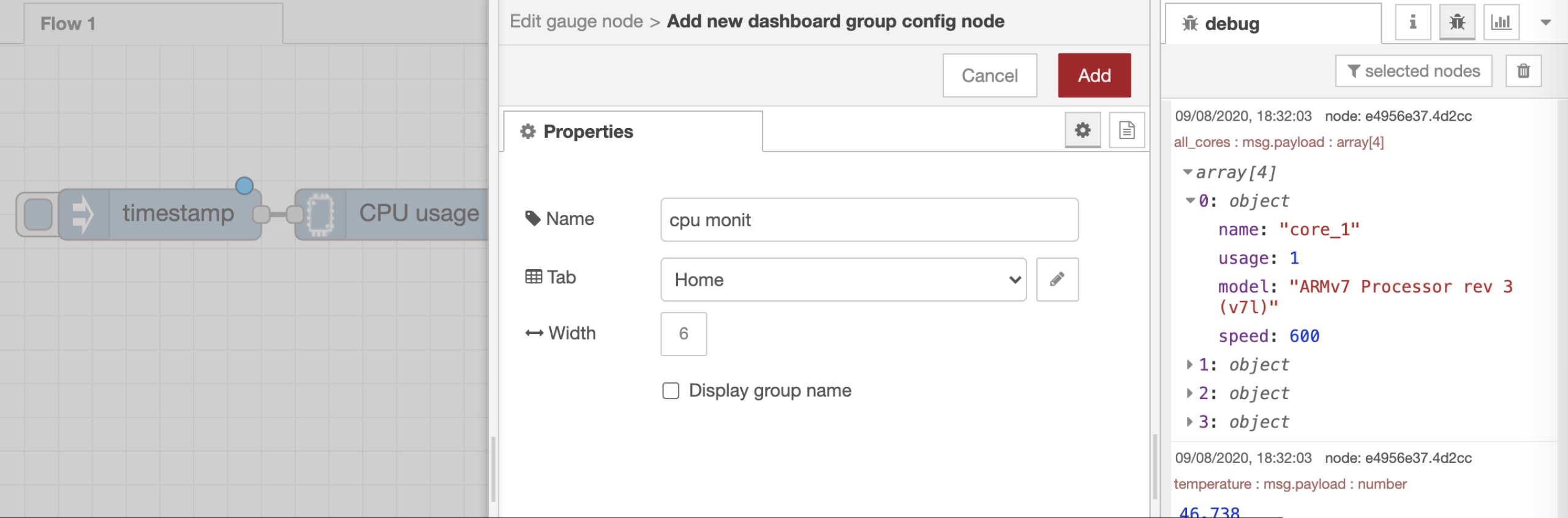
- Double-click on gauge node
- Since this our first time using the dashboard node, we need to setup the parent and child tabs: double-click on the pencil icon as shown above





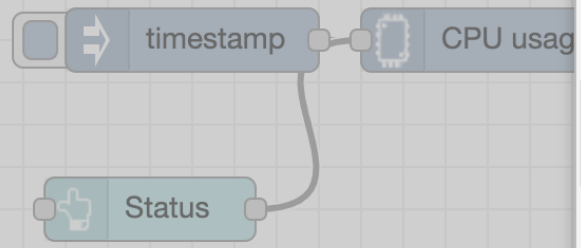
CPU Monitoring Apps: Add a Speed gauge

- 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



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.

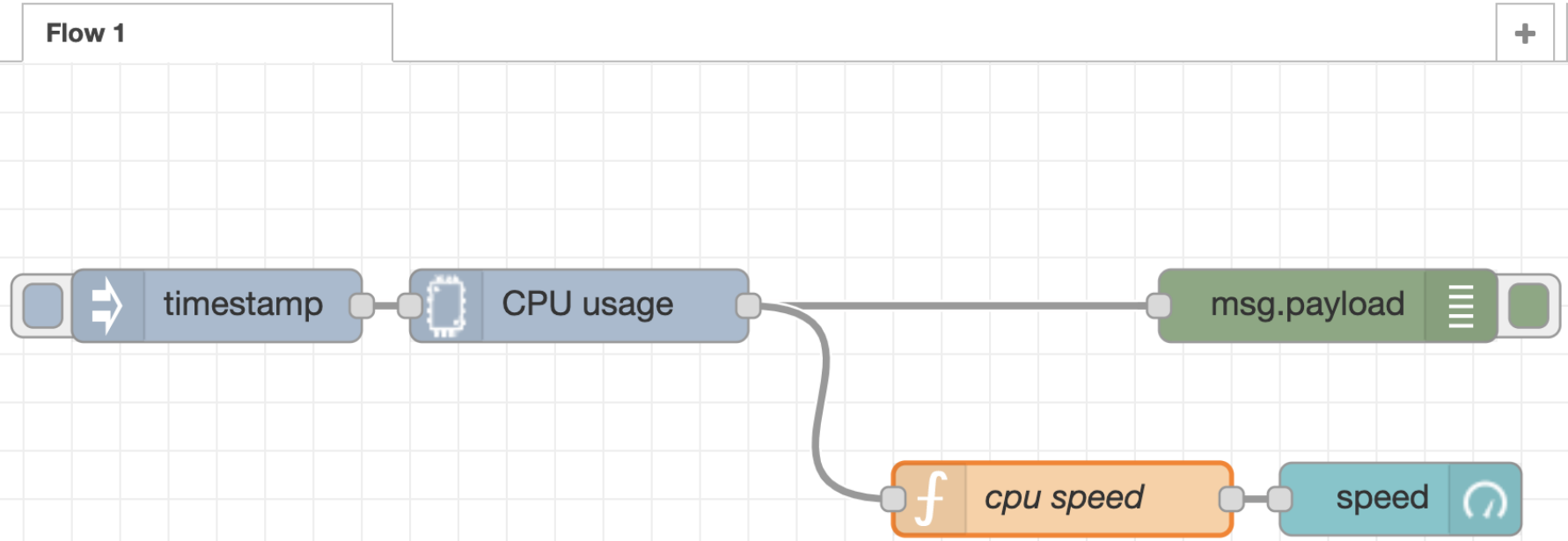
The `msg.payload` is searched for a numeric *value* and is formatted in accordance with the defined **Value Format**, which can then be formatted using Angular filters.

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.

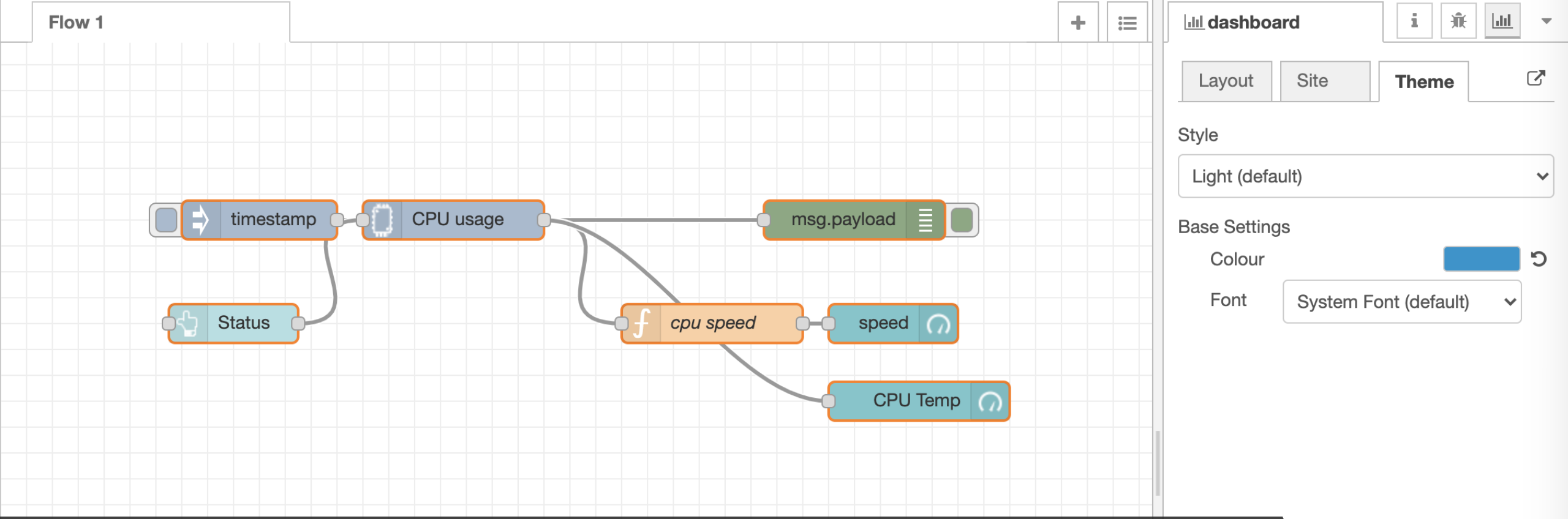
CPU Monitoring Apps: Add a Speed gauge

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



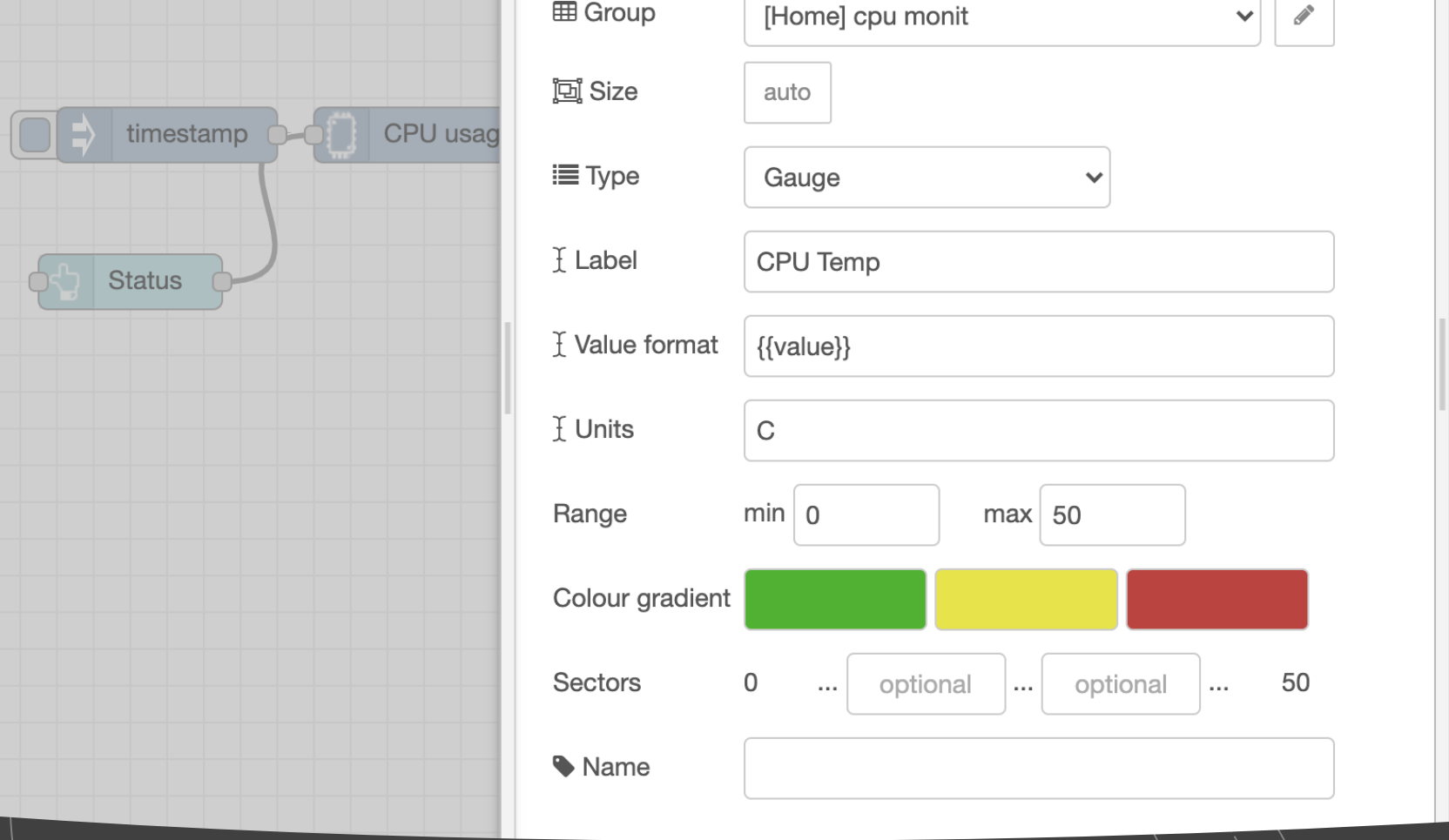
CPU Monitoring Apps: Add a Speed gauge

- 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



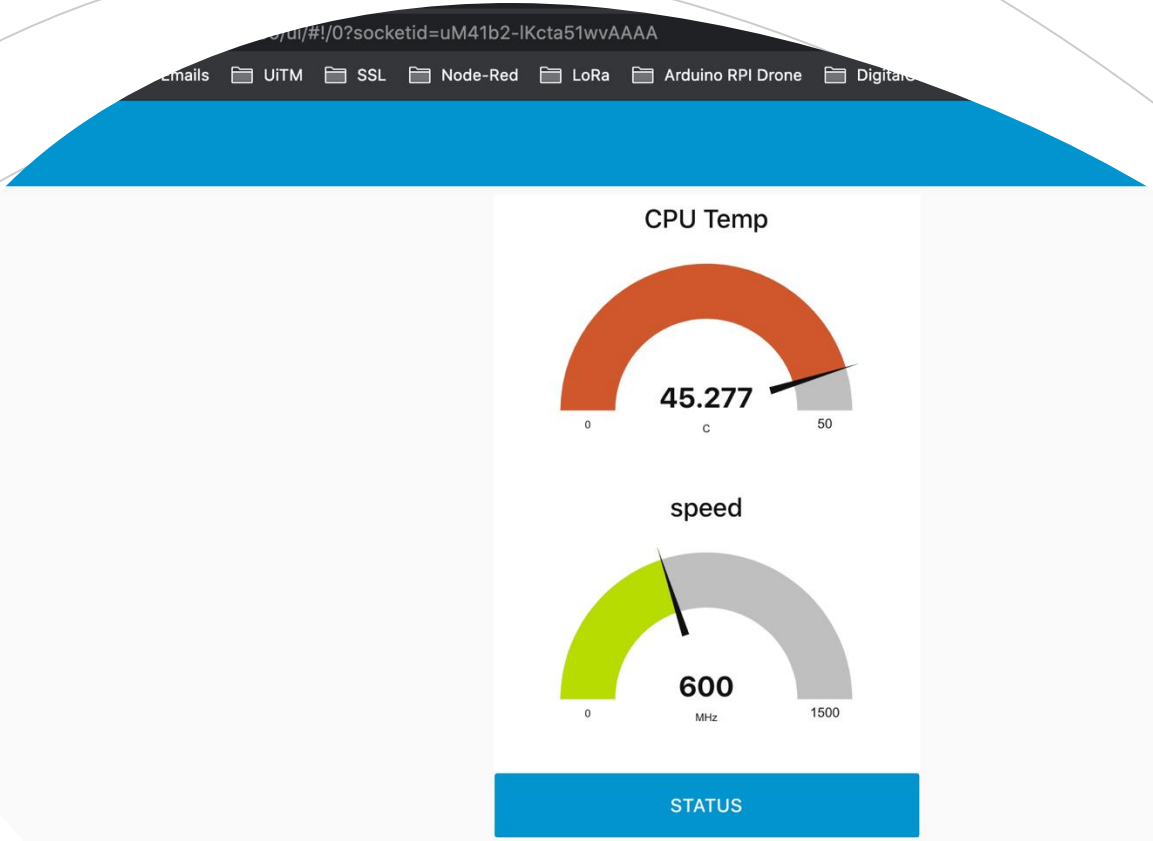
CPU Monitoring Apps: Add a Temp gauge

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

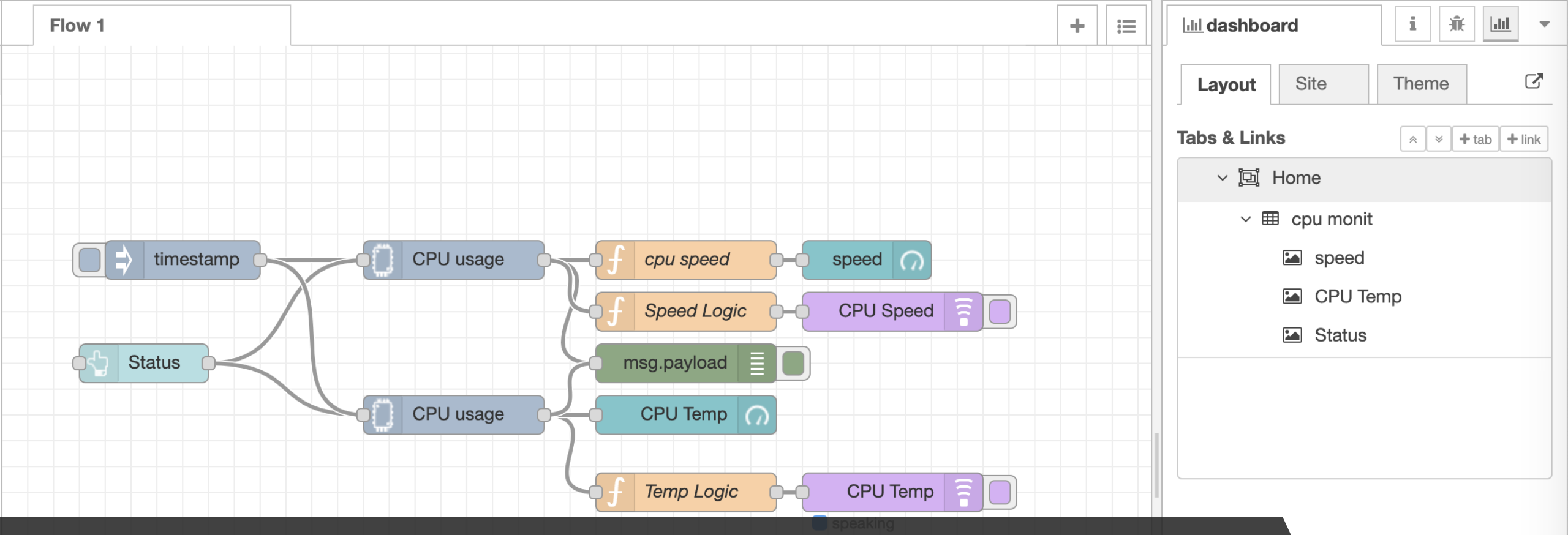


CPU Monitoring Apps: Adding Temp Gauge

- 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

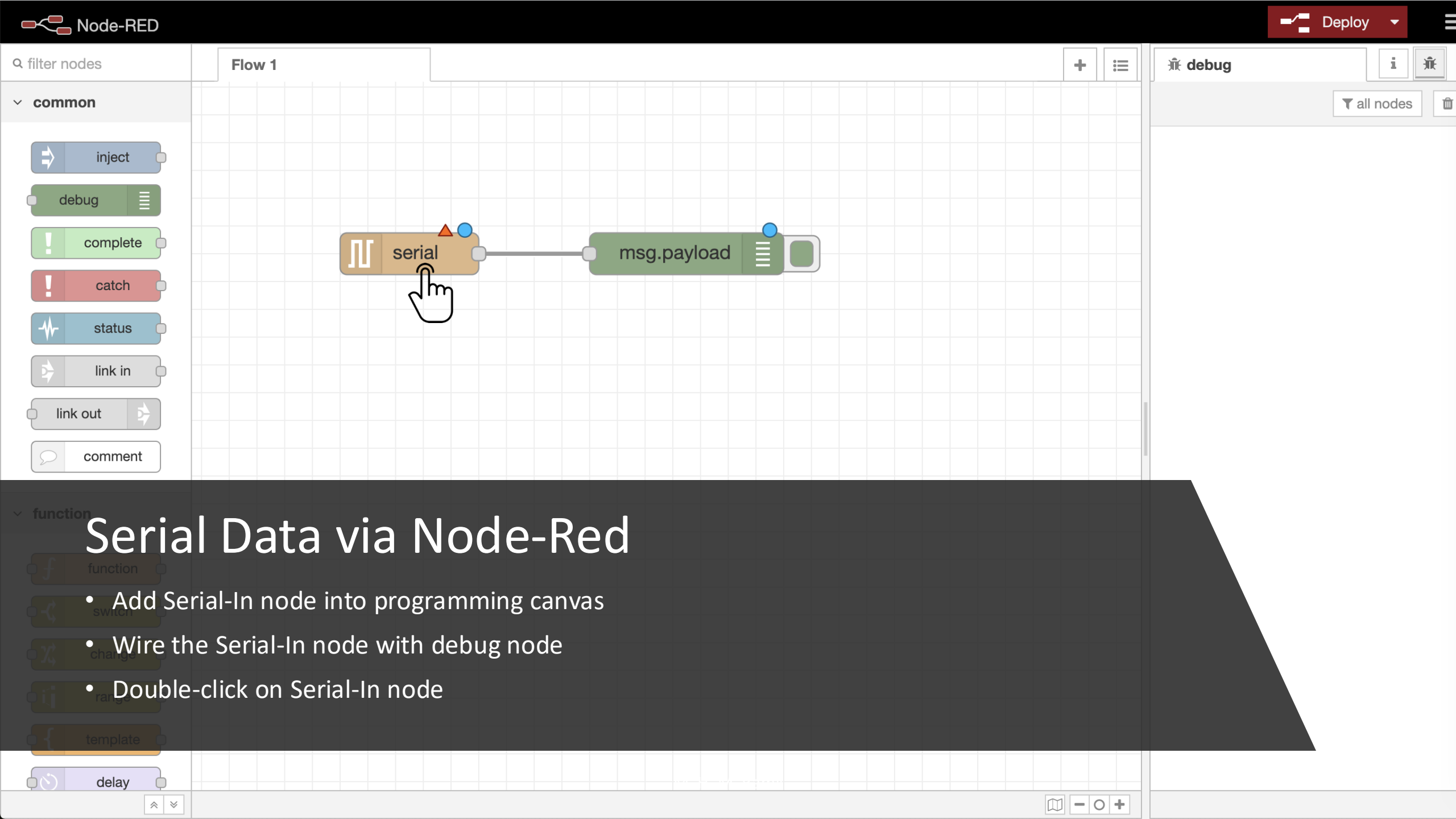


CPU Monitoring Apps: Dashboard UI



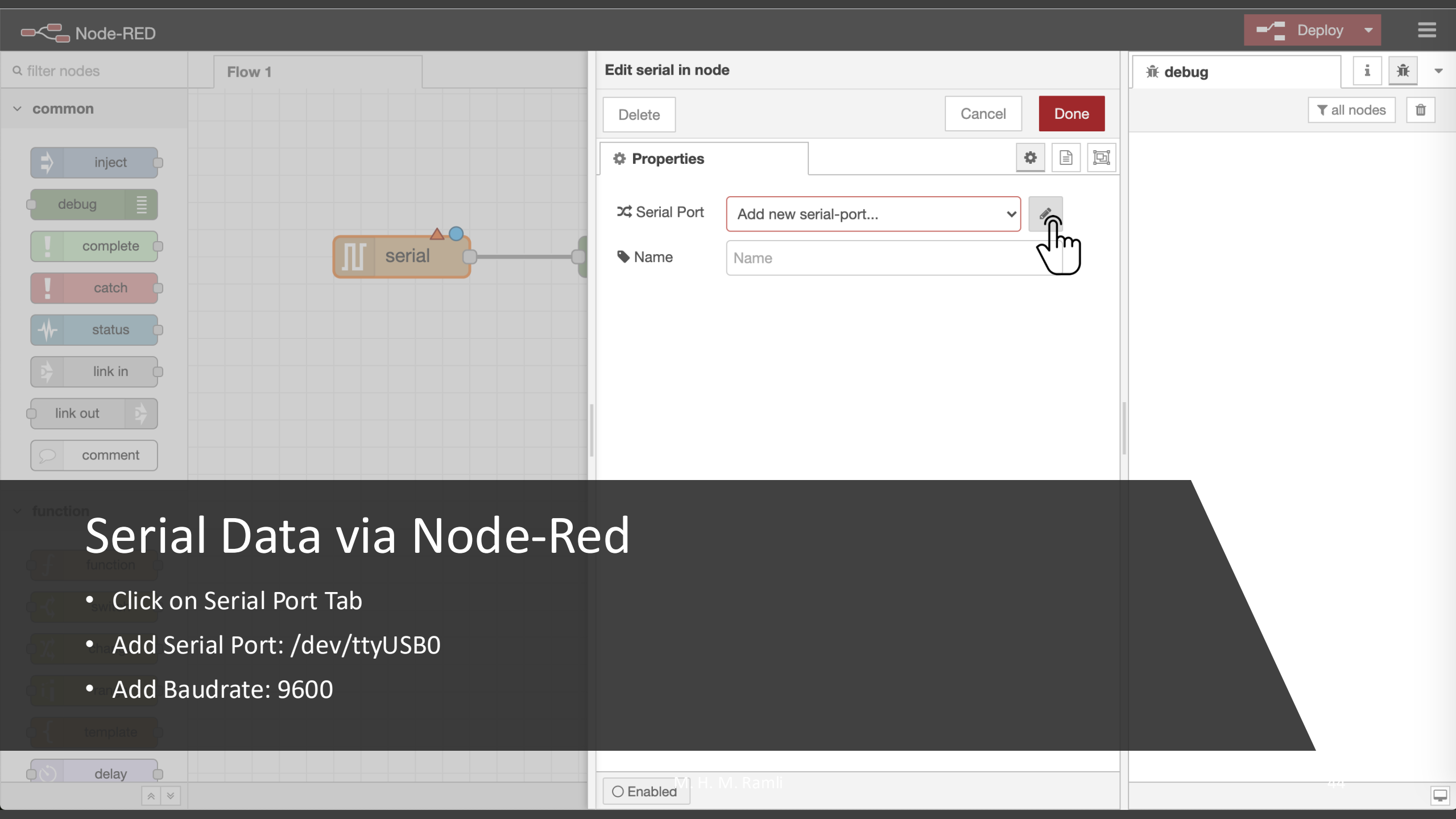
CPU Monitoring Apps: Add Sound Element

- 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



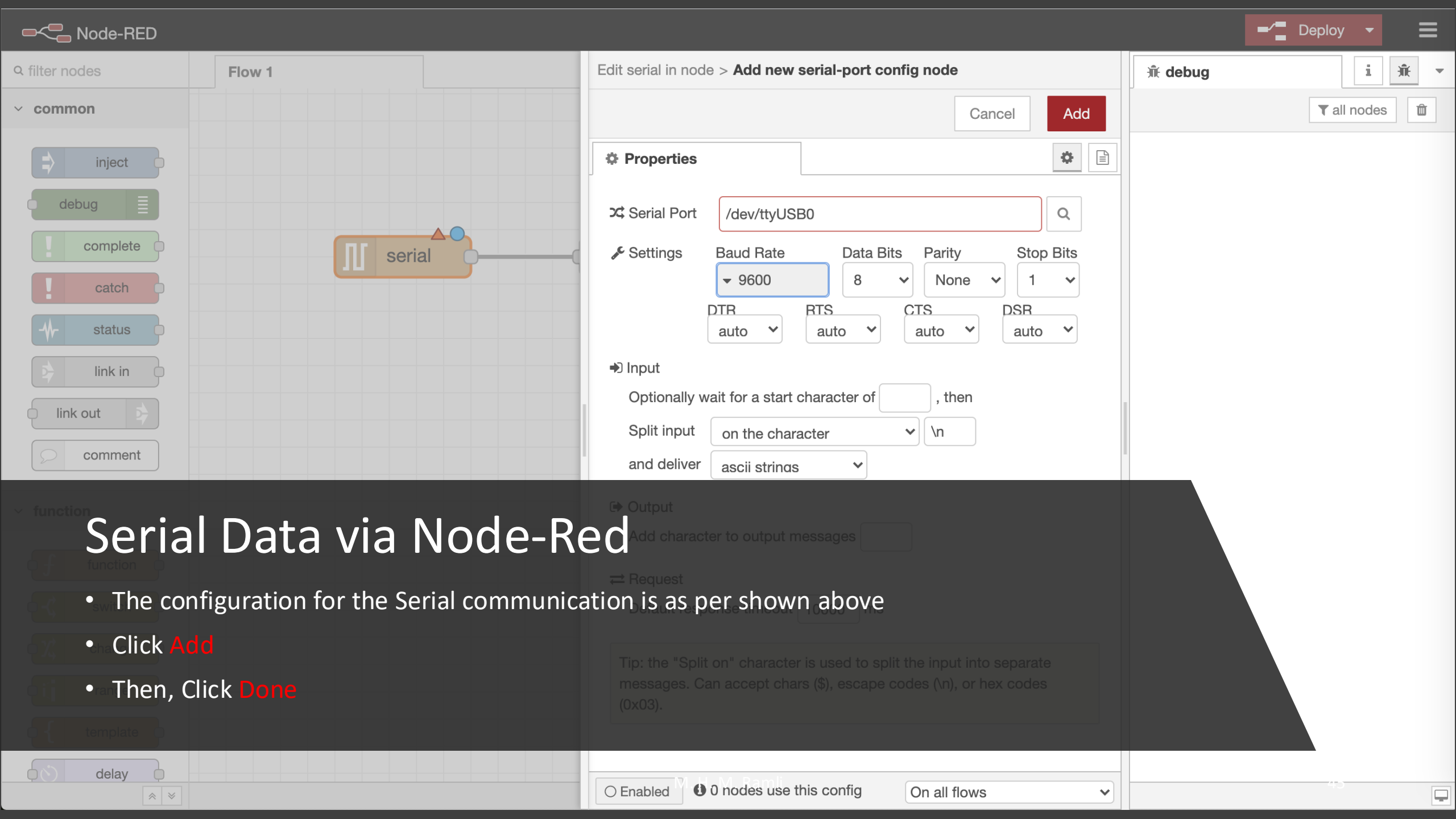
Serial Data via Node-Red

- Add Serial-In node into programming canvas
- Wire the Serial-In node with debug node
- Double-click on Serial-In node



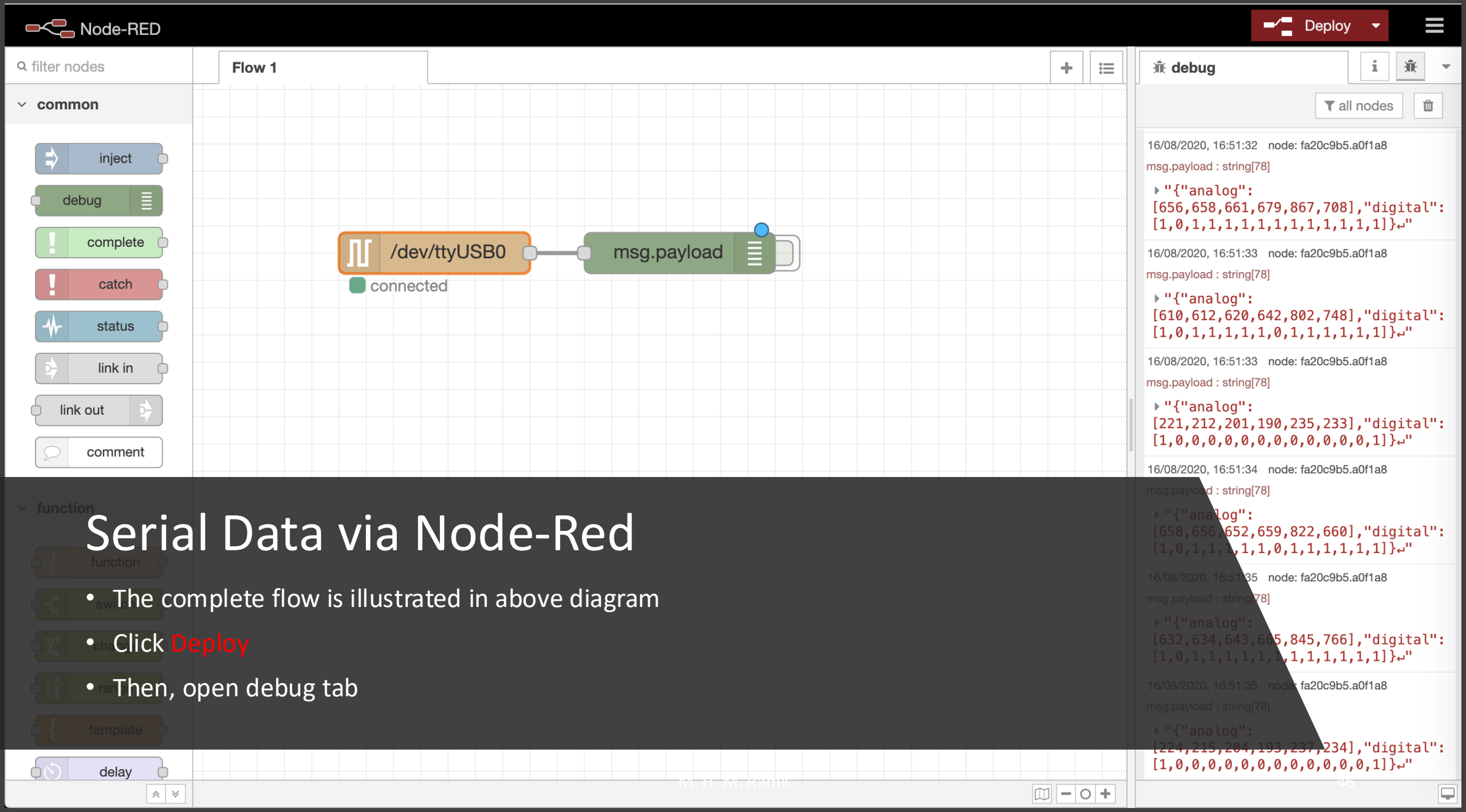
Serial Data via Node-Red

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



Serial Data via Node-Red

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

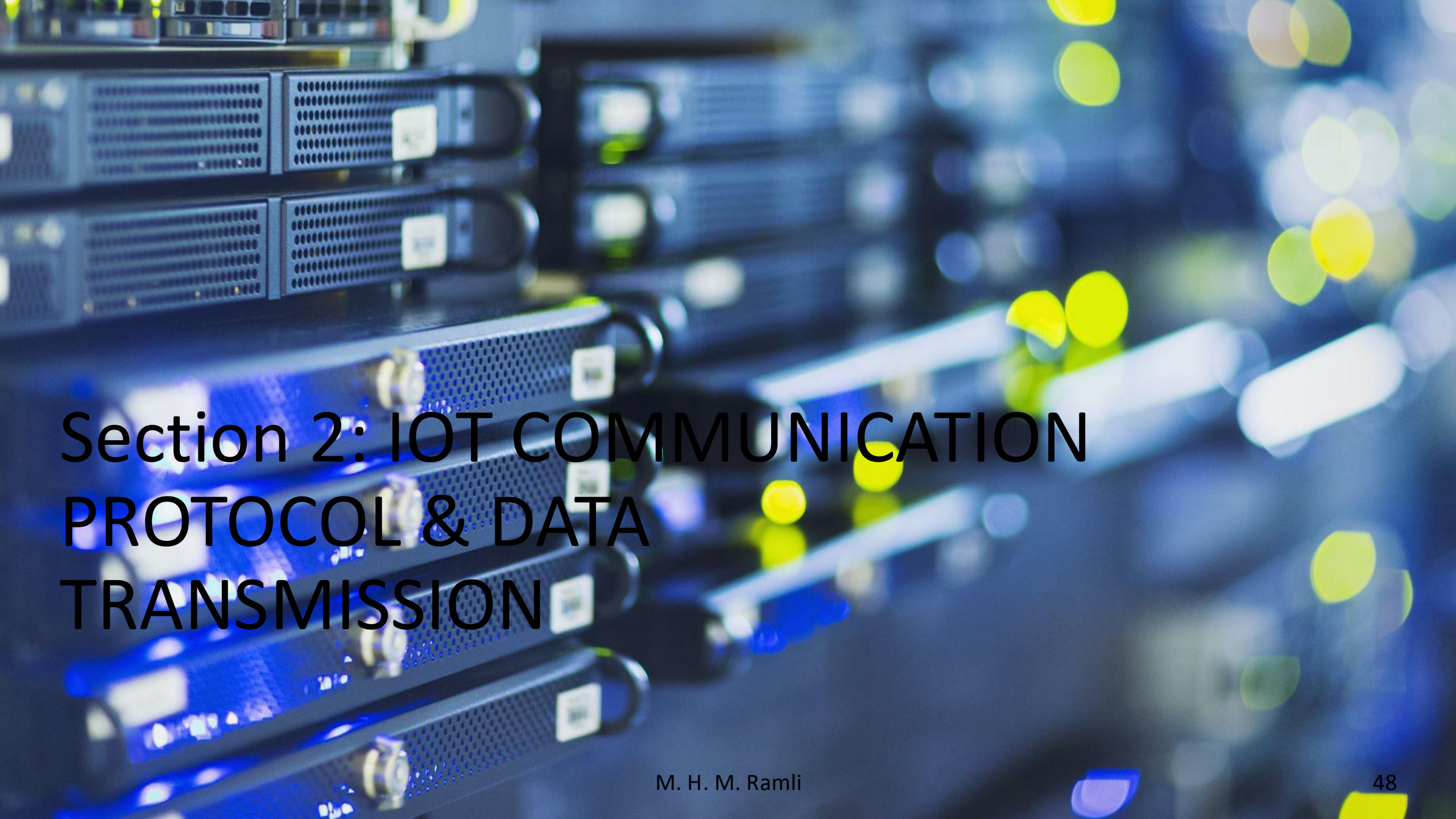


Serial Data via Node-Red

- The complete flow is illustrated in above diagram
- Click Deploy
- Then, open debug tab

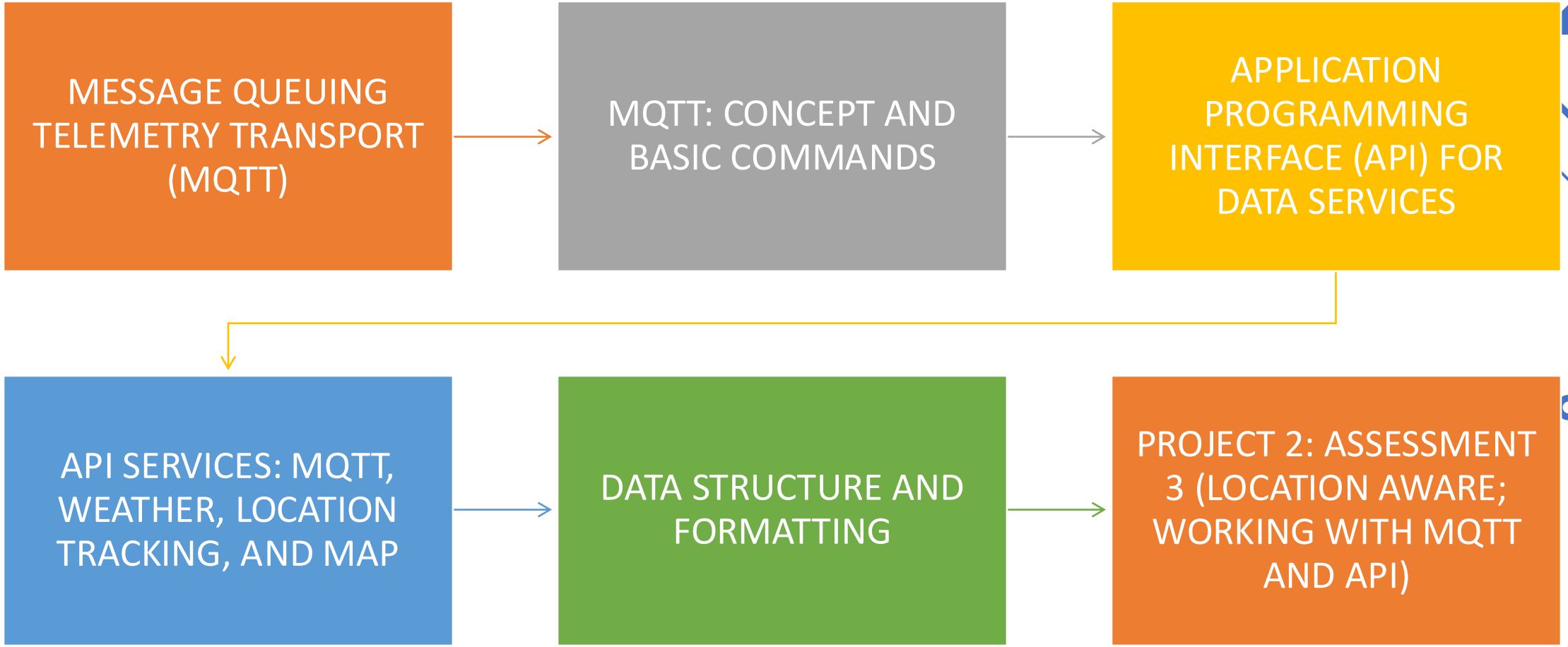
Project 1: Designing Dashboard for Weather Station

- 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

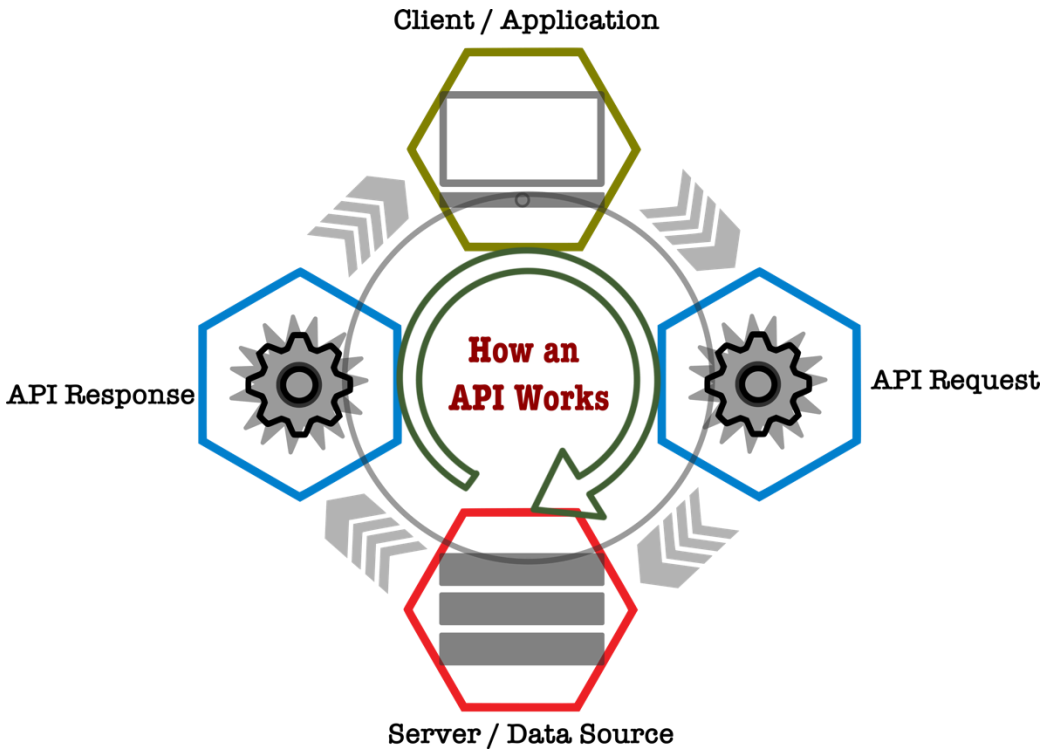
Using MQTT Node

- 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

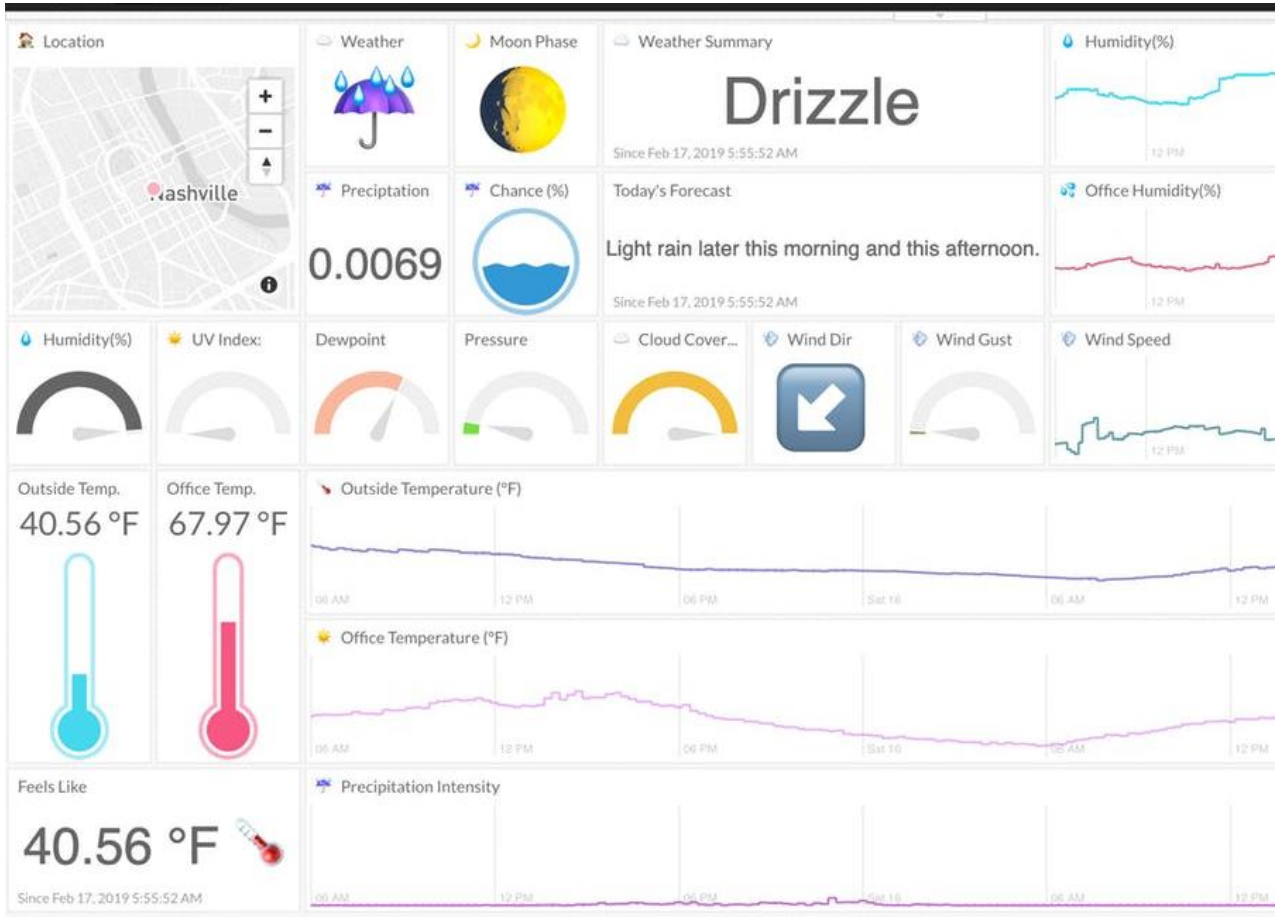
- topic: “test”, –subscribe/publish must be same topic.
- payload: “1564150020817”, –received payload (data) from the server (when subscribe to a topic).
- qos: 2, –Quality of Service: sent data
  - At most once (0)
  - At least once (1)
  - Exactly once (2)
- retain: true, –A retained message is a normal 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



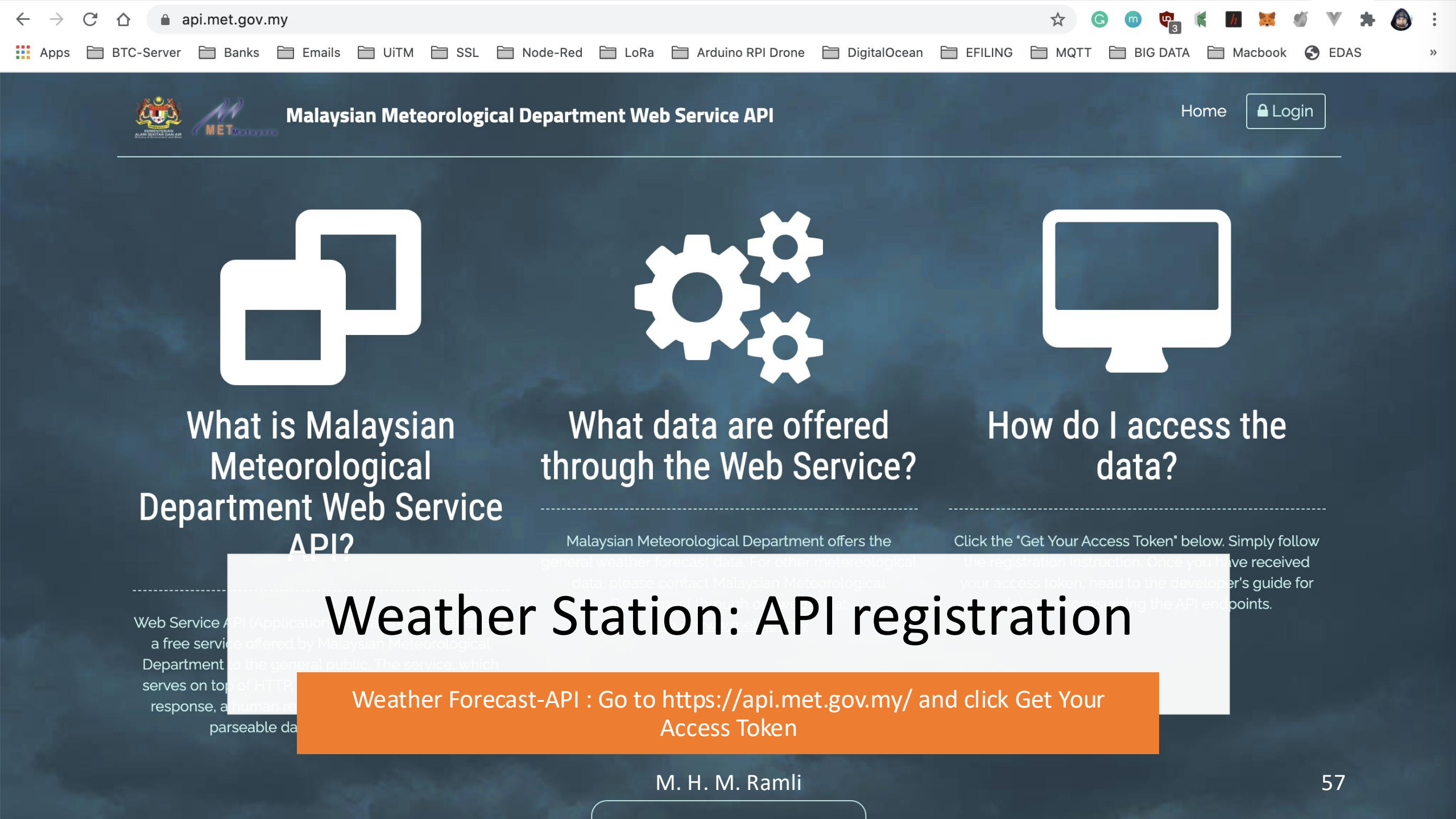
Concept: API

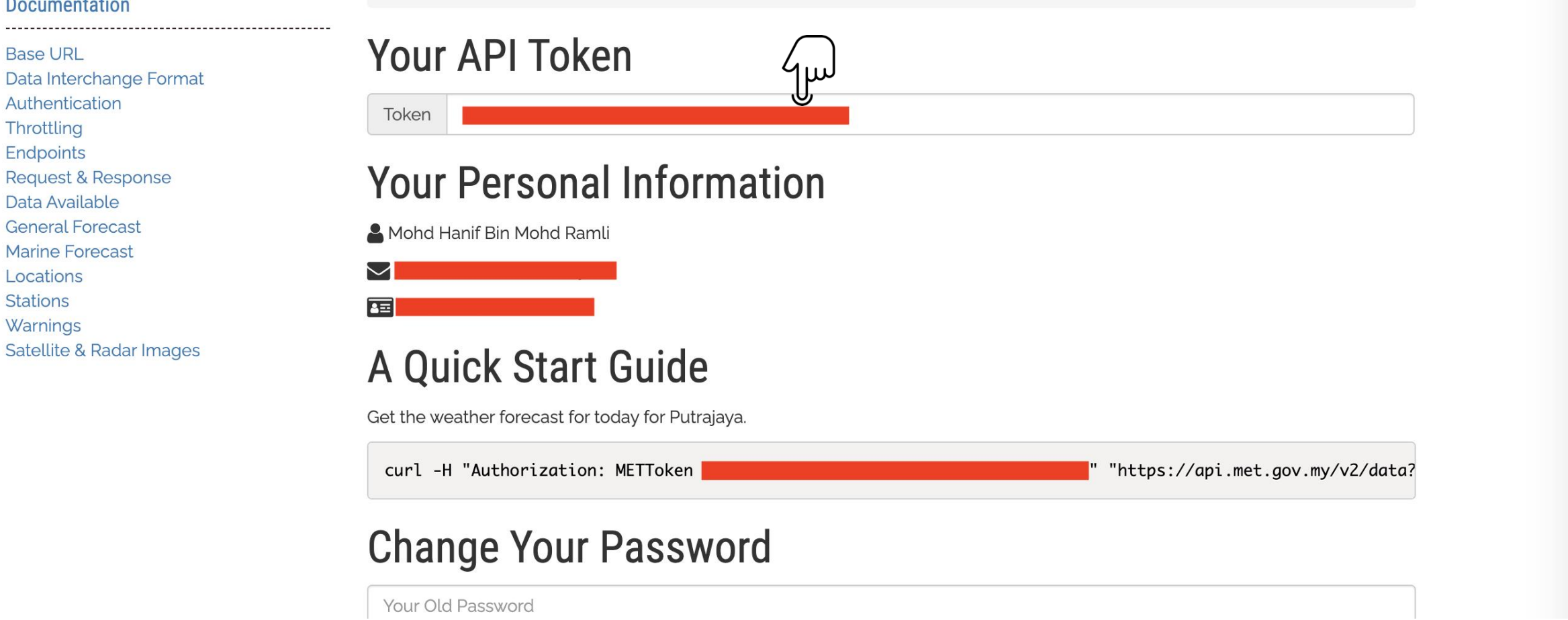
- **Programmatic access**: An application program interface that provides a developer with programmatic access to proprietary software application.
- **Communication**: A software intermediary that makes it possible for application programs to interact with each other and share data.
- **Goal**: get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


api.met.gov.my
Apps
BTC-Server
Banks
Emails
UiTM
SSL
Node-Red
LoRa
Arduino RPI Drone
DigitalOcean
EFILING
MQTT
BIG DATA
Macbook
EDAS
Malaysian Meteorological Department Web Service API
Home
Login
What is Malaysian Meteorological Department Web Service API?
What data are offered through the Web Service?
How do I access the data?
Malaysian Meteorological Department offers the
Click the "Get Your Access Token" below. Simply follow
Weather Station: API registration
Weather Forecast-API : Go to https://api.met.gov.my/ and click Get Your Access Token

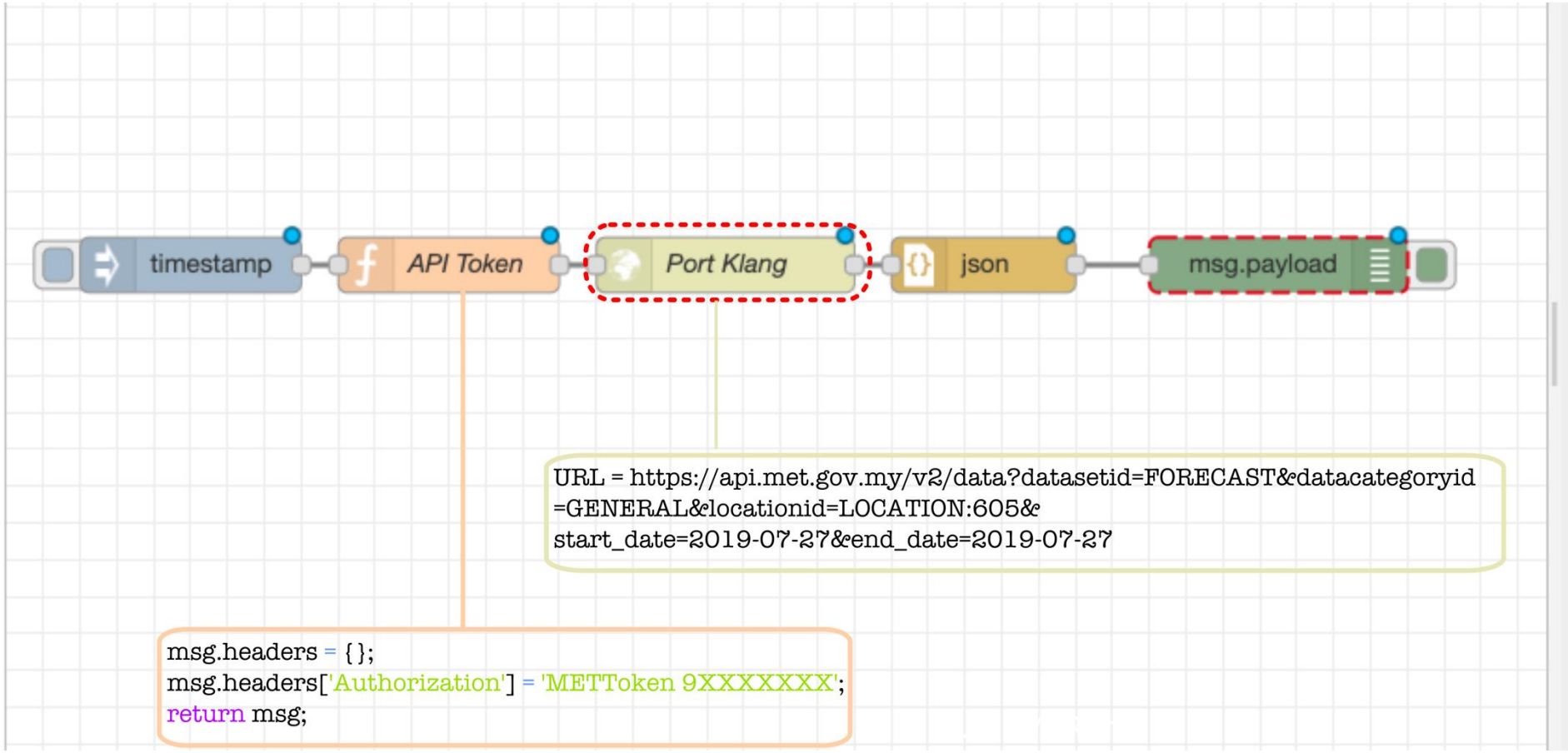


Weather Station: API registration

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

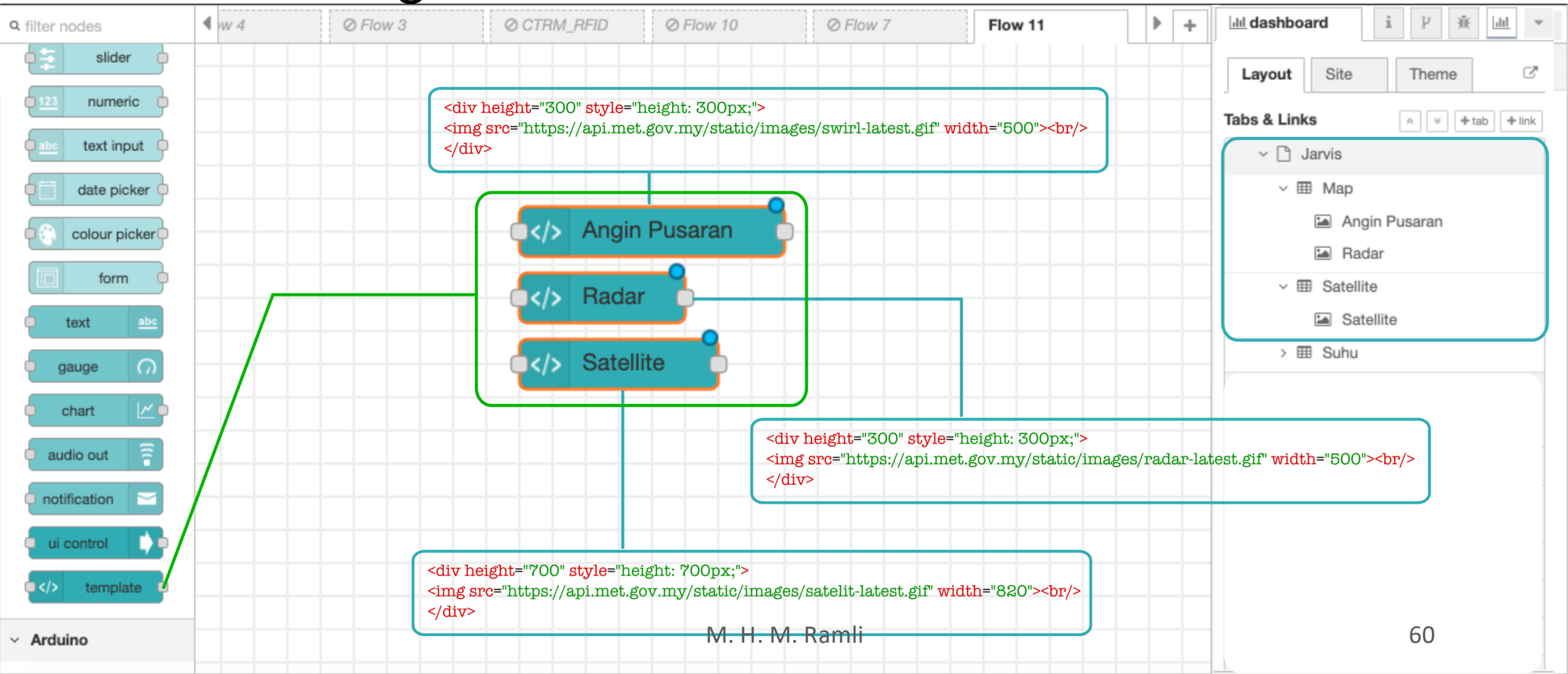
Weather Station

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

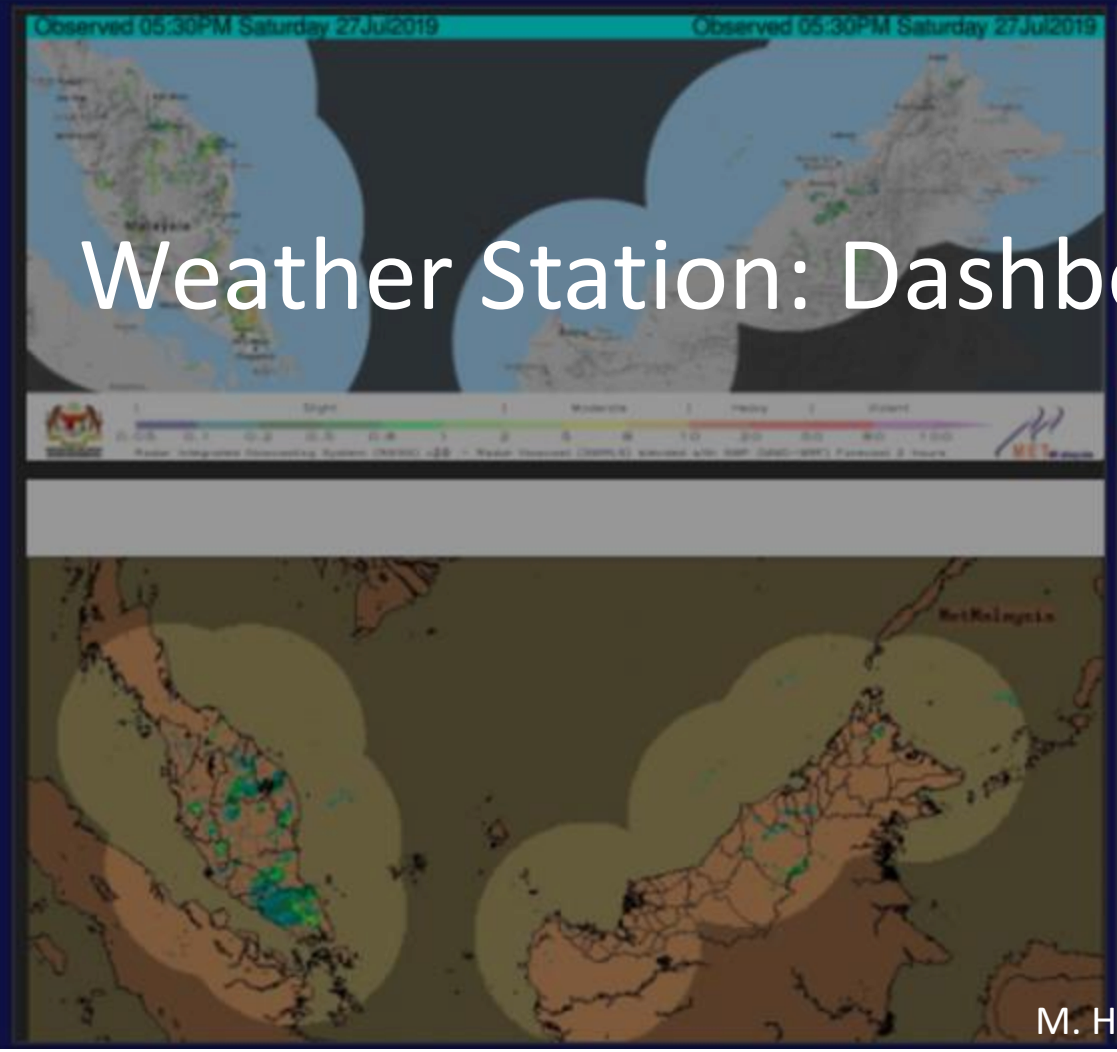


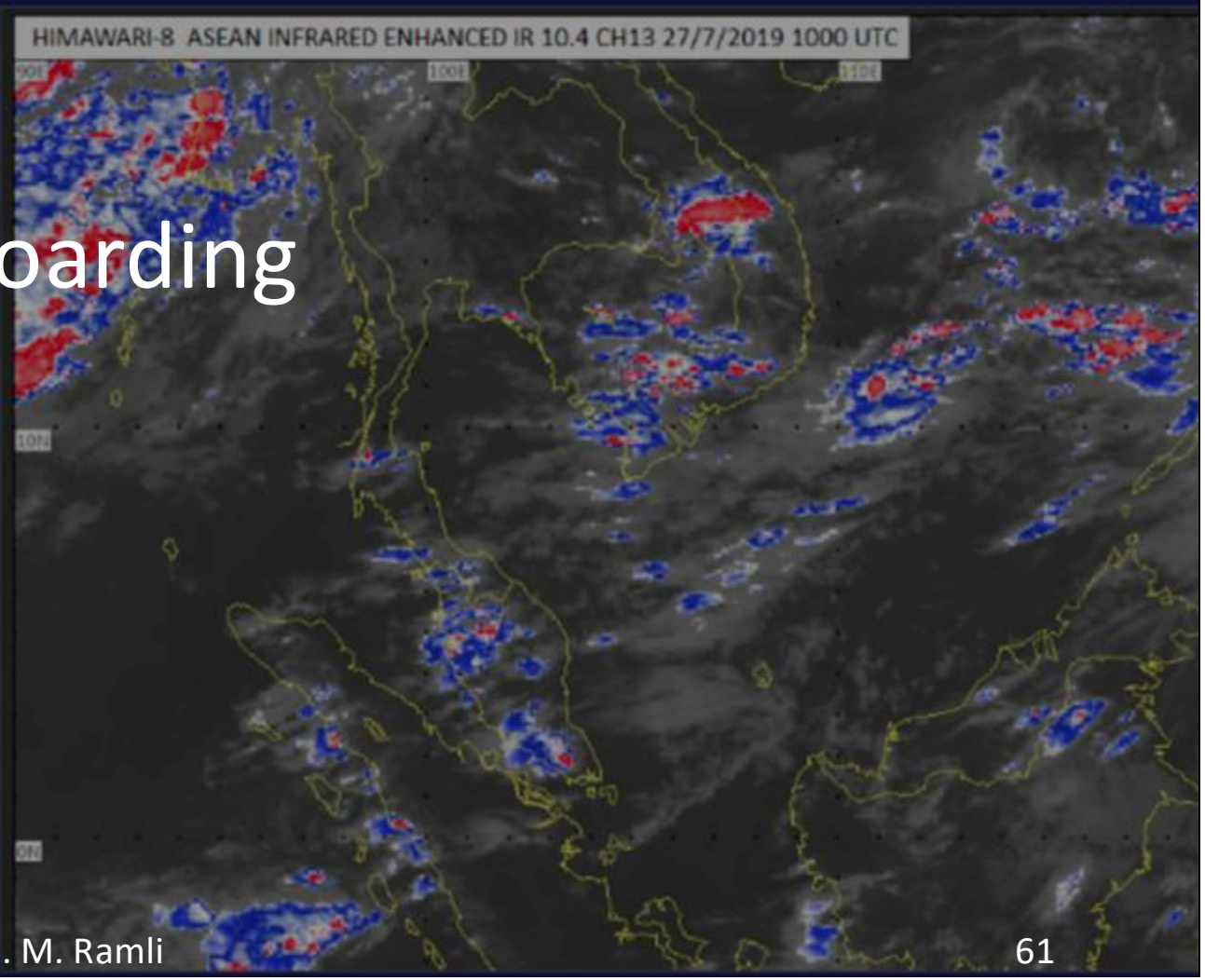
Weather Station: Dashboarding

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

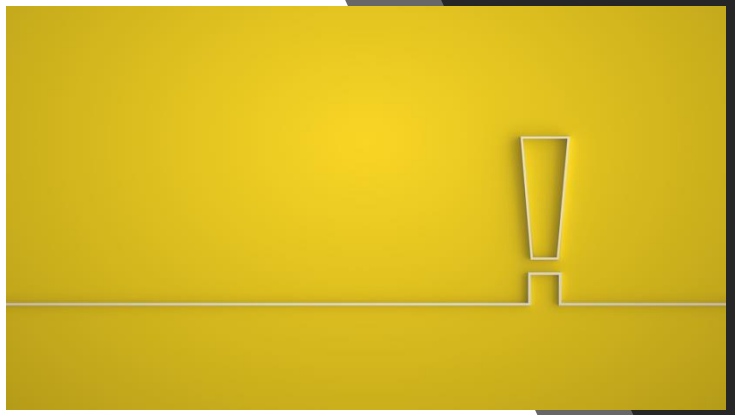


Weather Station: Dashboarding





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.

owntracks.org

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

OWNTRACKS

OWNTRACKS FOR IOS | OWNTRACKS FOR ANDROID

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.

8:53 4G

Search

OwnTracks
Christoph Krey

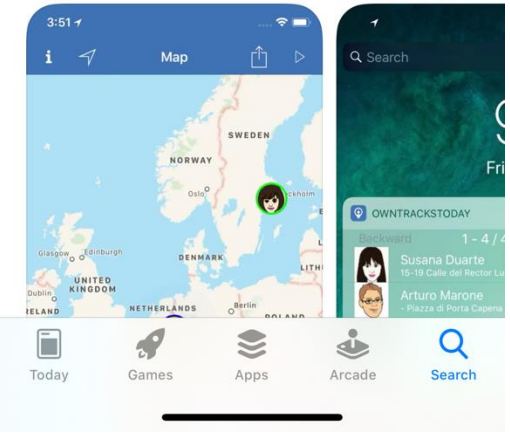
Not Enough Ratings | 4+ Age

What's New Version History

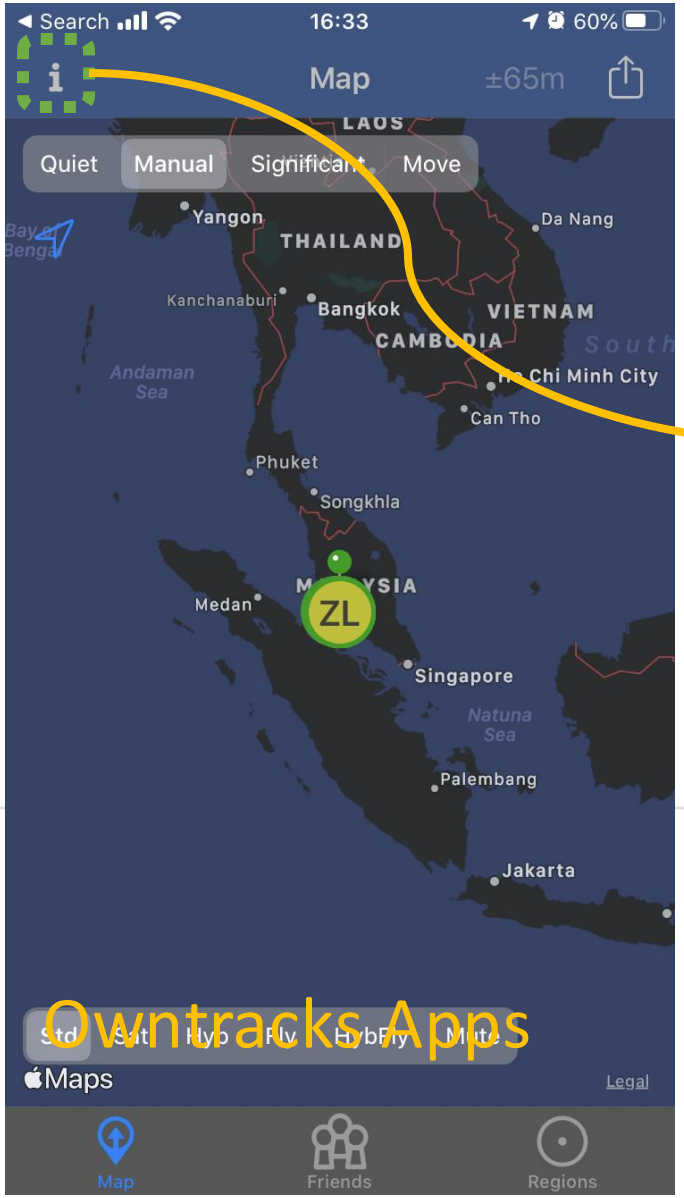
Version 13.1.7 7mo ago

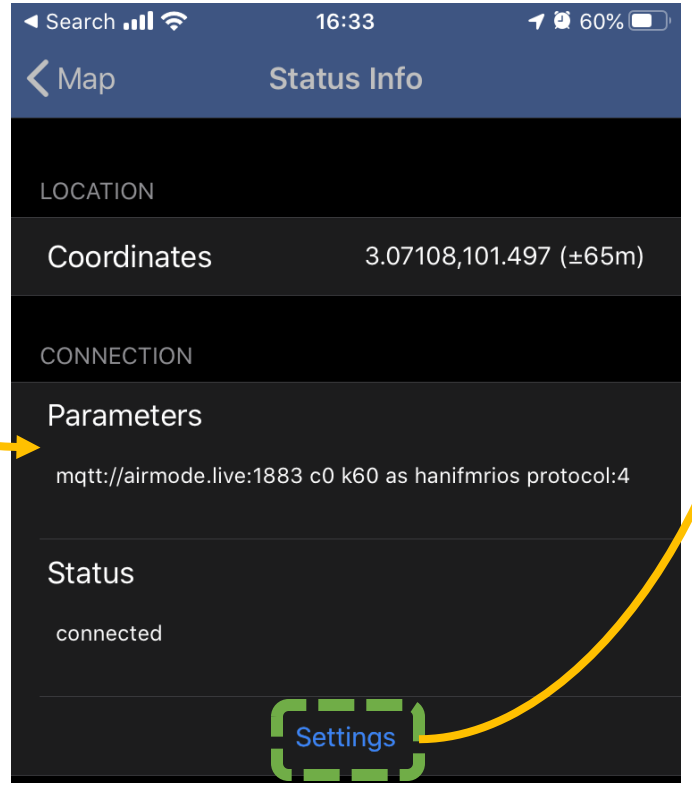
A few bug fixes, especially
- [FIX] Can't see friends #610 (https://github.com/owntracks/ios/issues/610)

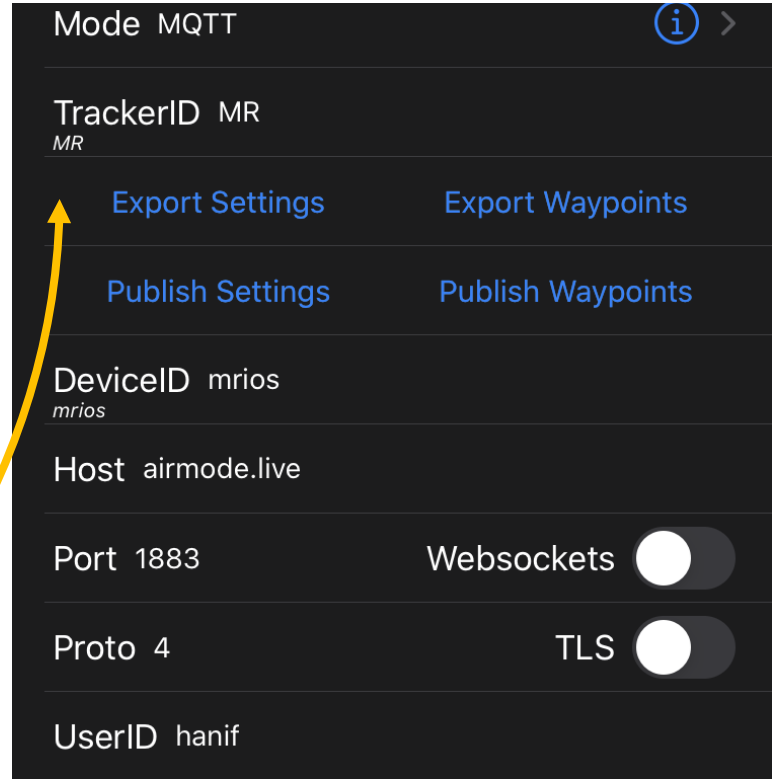
Preview



Location API

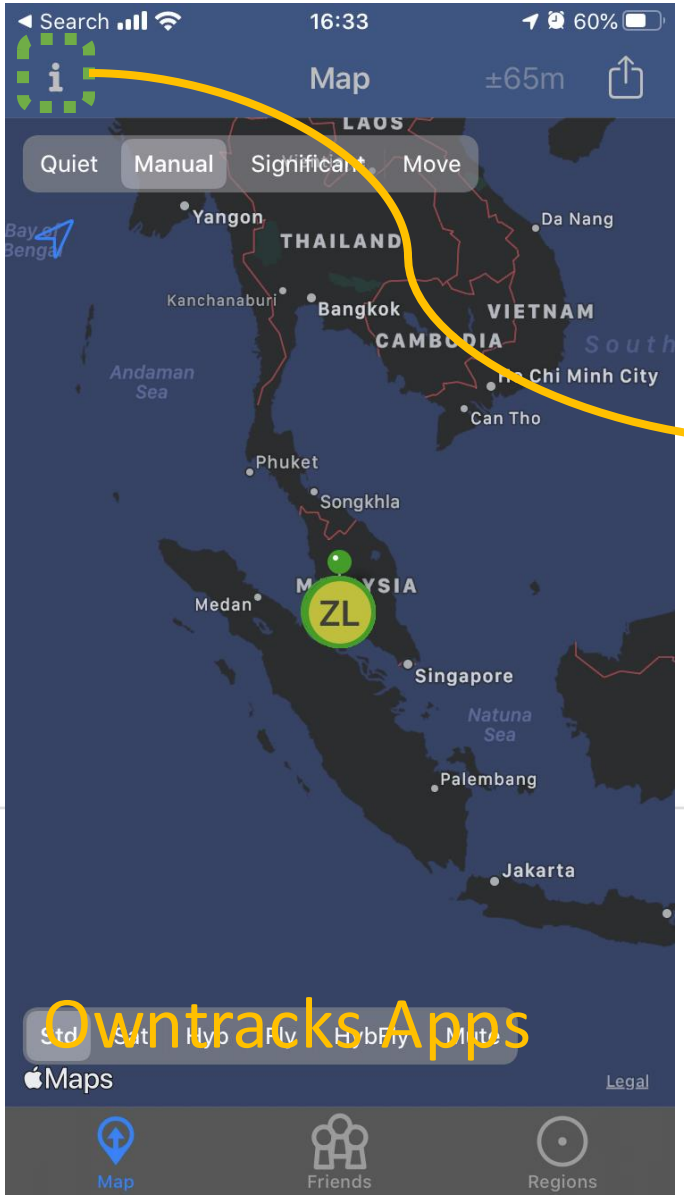


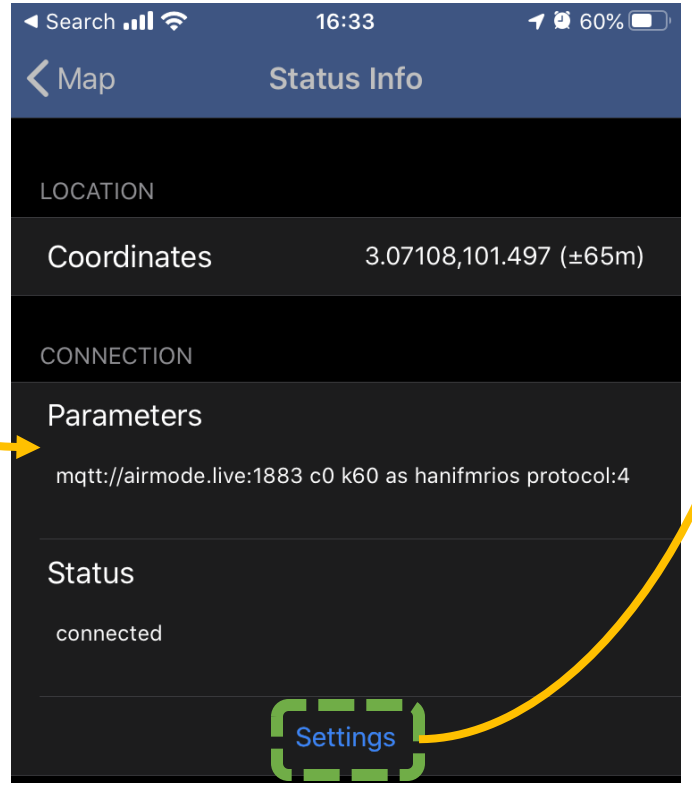


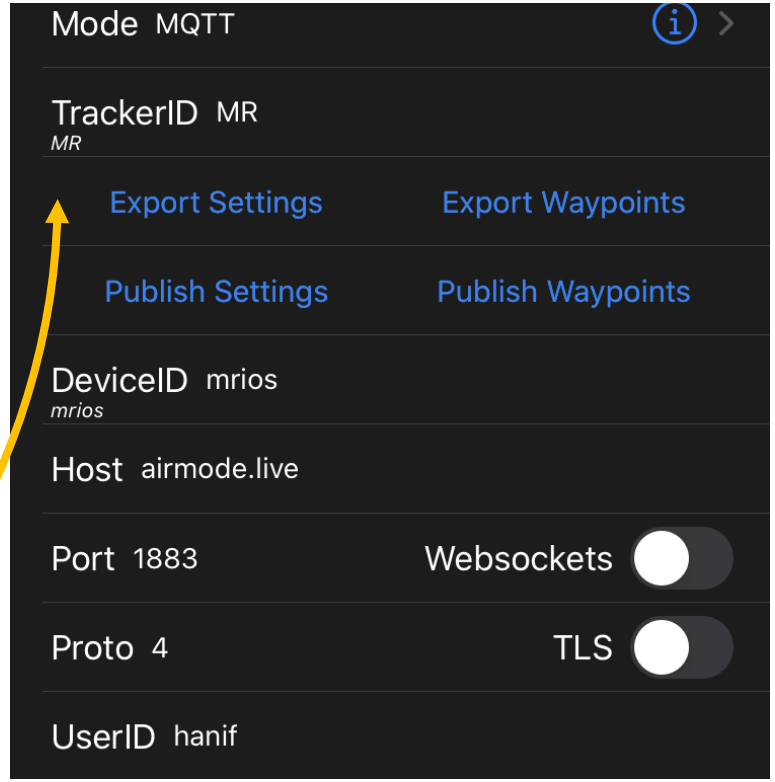


Owntracks Apps

- 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







Owntracks Apps

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

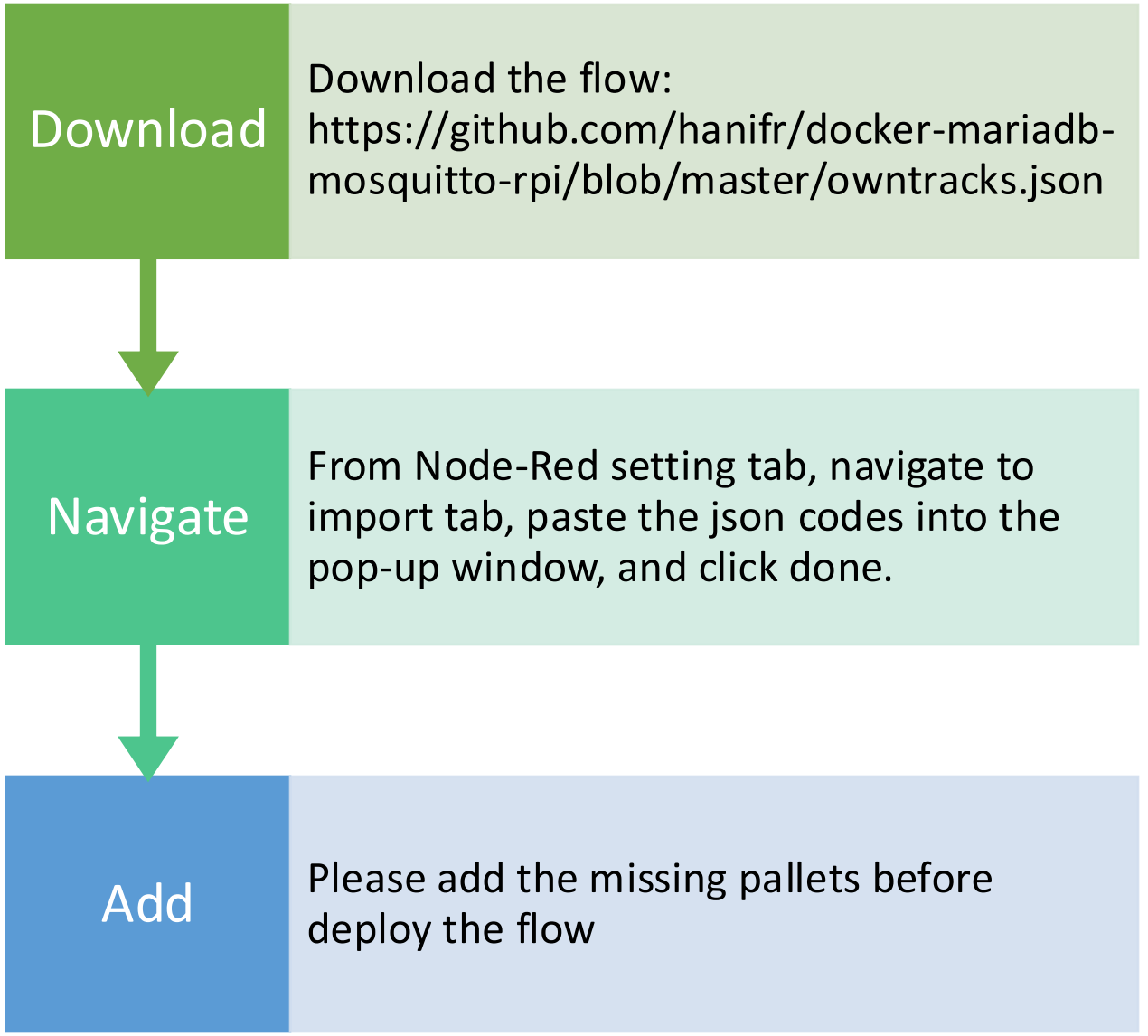
Owntracks API: JSON

- 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

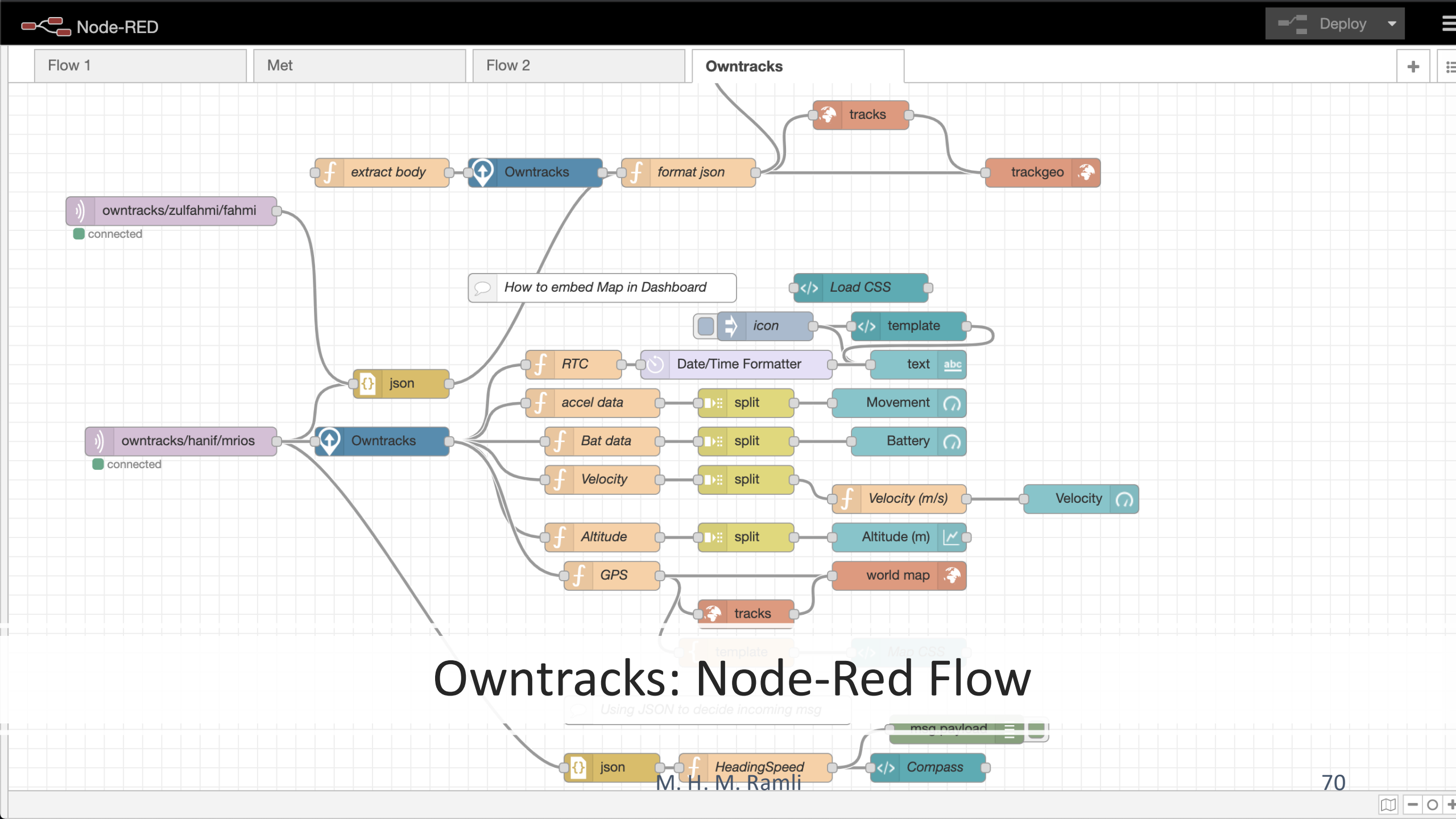
- tid: Tracker ID used to display the initials of a user *(iOS,Android/string/optional)* required for http mode
- tst: UNIX epoch timestamp in seconds of the location fix *(iOS,Android/integer/epoch/required)*
- vac: vertical accuracy of the alt element *(iOS/integer/meters/optional)*
- vel: velocity *(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



Location Aware: Owntracks

- Paste the downloaded codes into import node
- Click Import
- Next, click Deploy



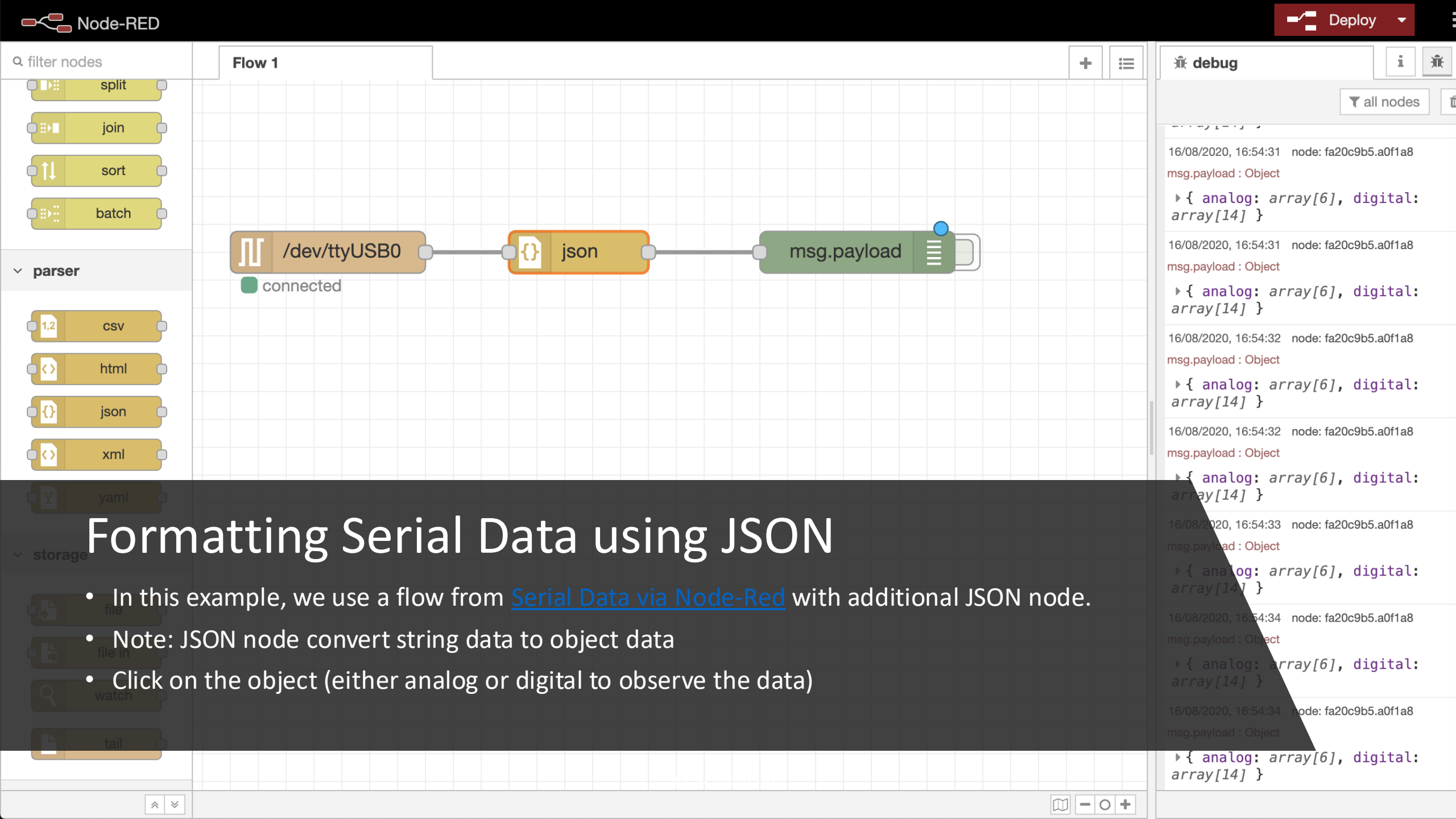
Owntracks: Node-Red Flow

M. H. M. Ramli

Data structure and formatting

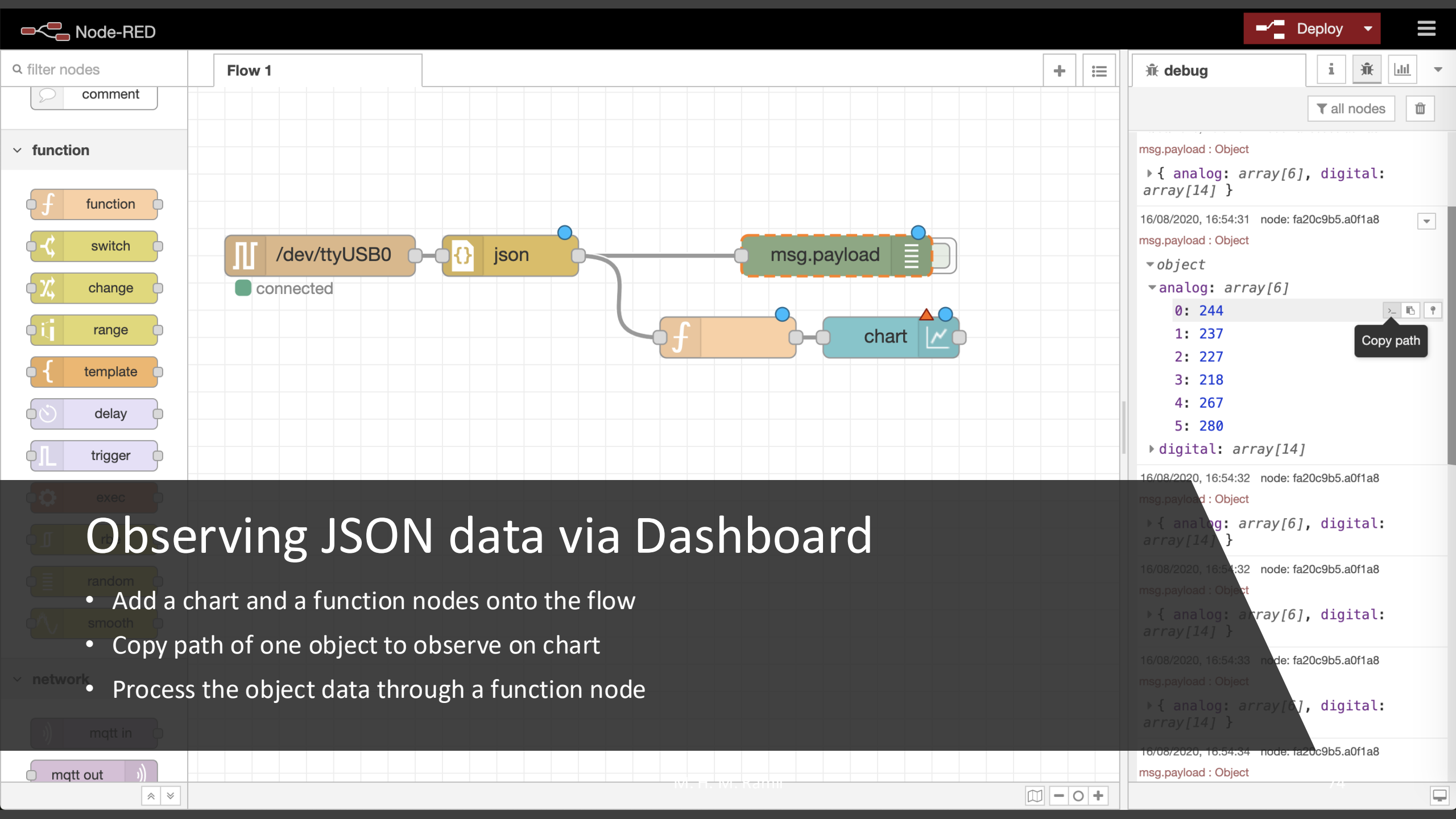
JavaScript Object Notation: JSON

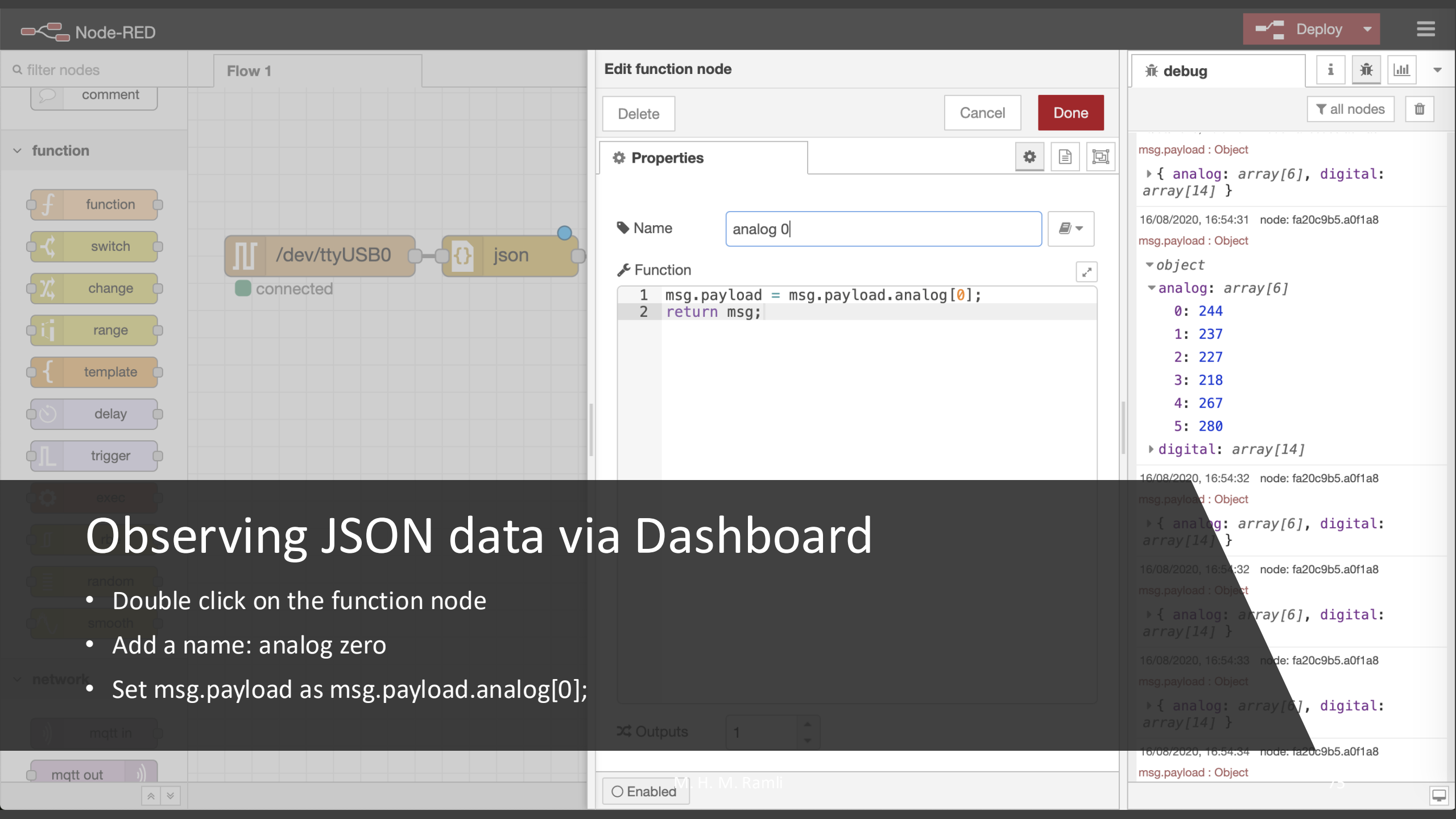
- JSON : is a language-independent data format. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data.
  - stands for JavaScript Object Notation
  - a lightweight format for storing and transporting data
  - often used when data is sent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

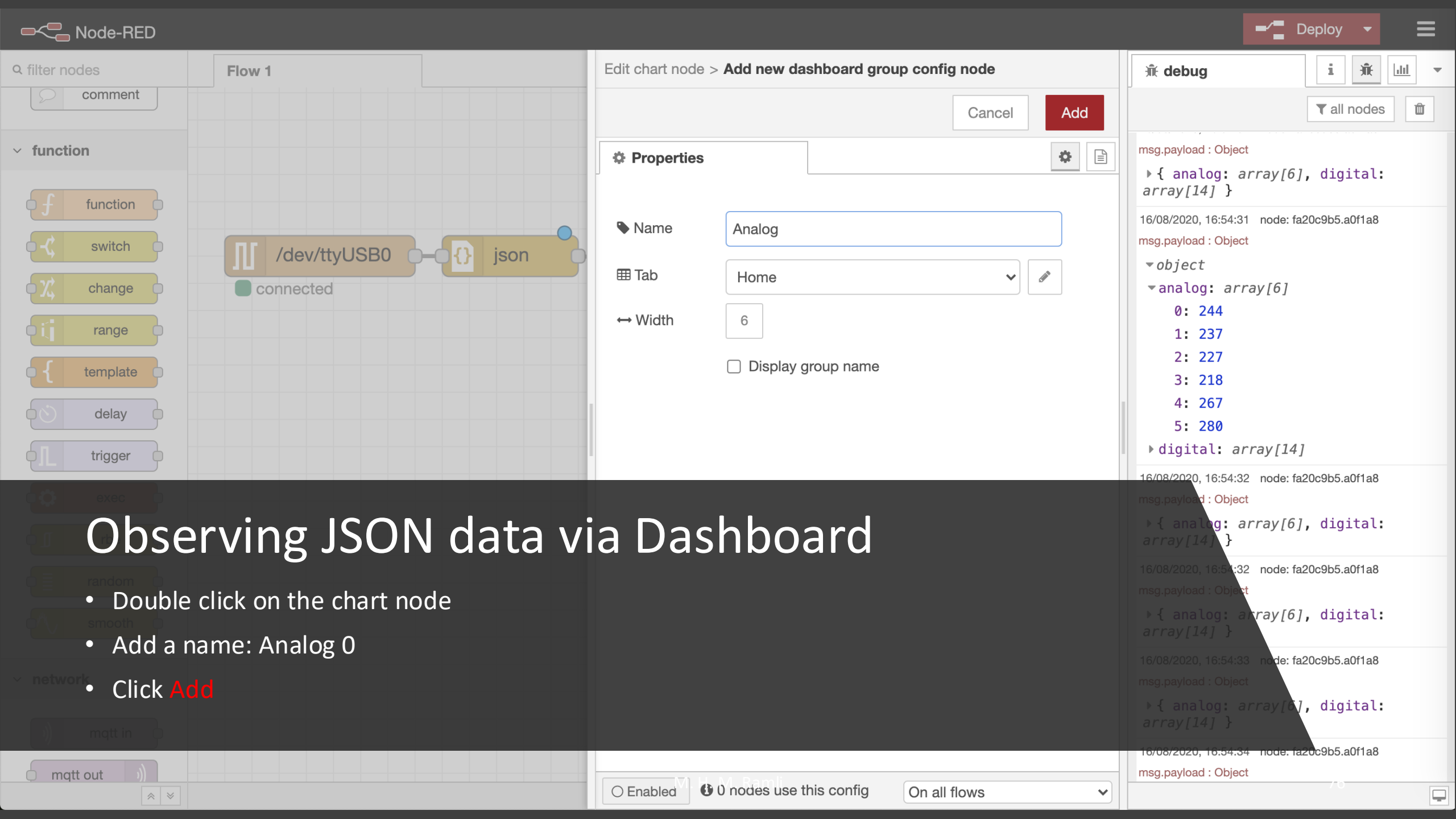


Formatting Serial Data using JSON

- 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)


Node-RED
Deploy
filter nodes
comment
function
function
switch
change
range
template
delay
trigger
exec
random
smooth
network
mqtt in
mqtt out
Flow 1
/dev/ttyUSB0
connected
json
msg.payload
chart
debug
all nodes
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]
Copy path
16/08/2020, 16:54:32 node: fa20c9b5.a0f1a8
16/08/2020, 16:54:33 node: fa20c9b5.a0f1a8
16/08/2020, 16:54:34 node: fa20c9b5.a0f1a8
Observing JSON data via Dashboard
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


Node-RED
Deploy
filter nodes
Flow 1
comment
function
function
switch
change
range
template
delay
trigger
exec
random
smooth
network
mqtt in
mqtt out
/dev/ttyUSB0
connected
json
Edit function node
Delete
Cancel
Done
Properties
Name
analog 0
Function
1 msg.payload = msg.payload.analog[0];
2 return msg;
Outputs
1
Enabled
debug
all nodes
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
16/08/2020, 16:54:33 node: fa20c9b5.a0f1a8
16/08/2020, 16:54:34 node: fa20c9b5.a0f1a8
Observing JSON data via Dashboard
Double click on the function node
Add a name: analog zero
Set msg.payload as msg.payload.analog[0];
M. H. M. Ramli

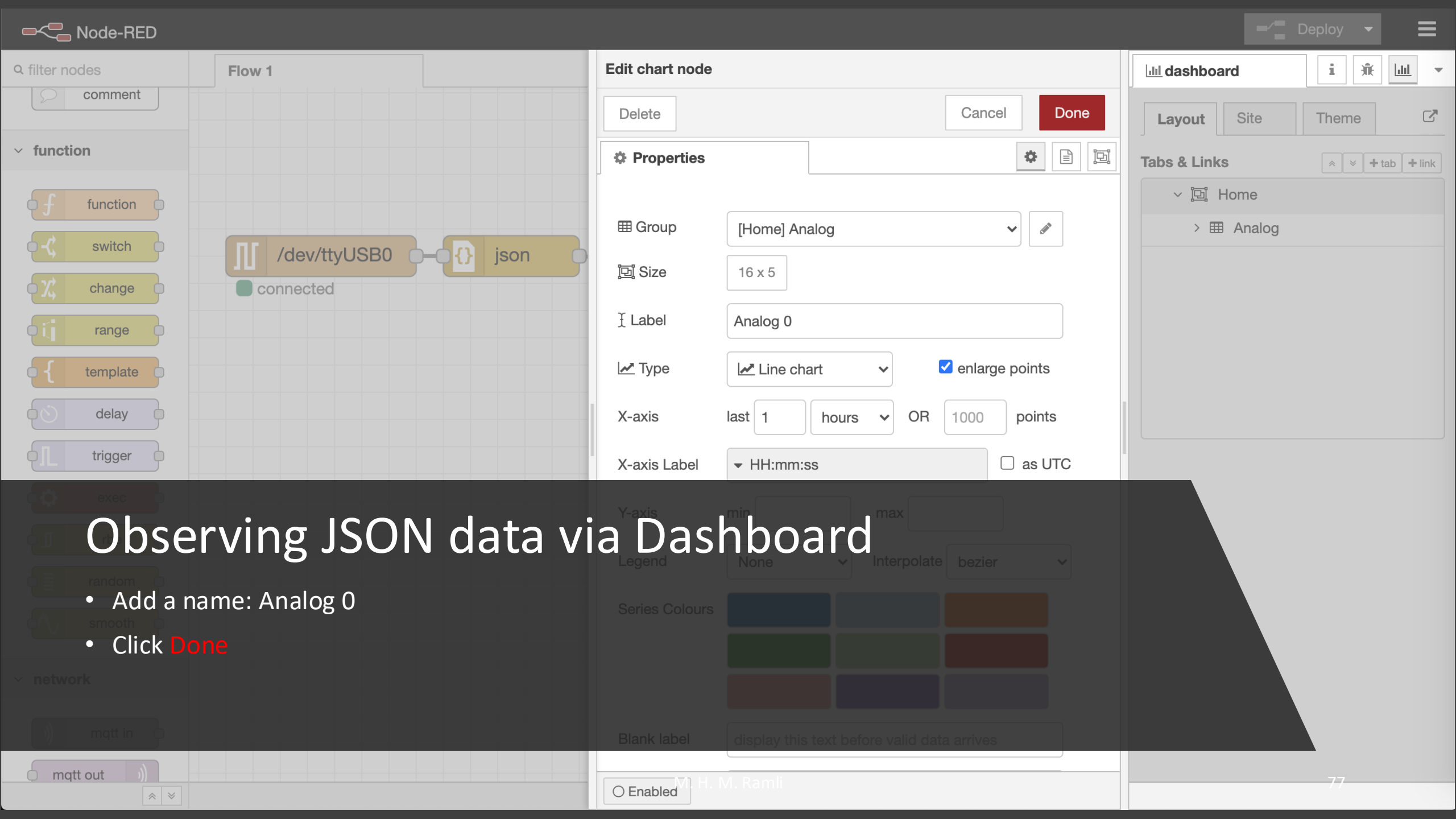


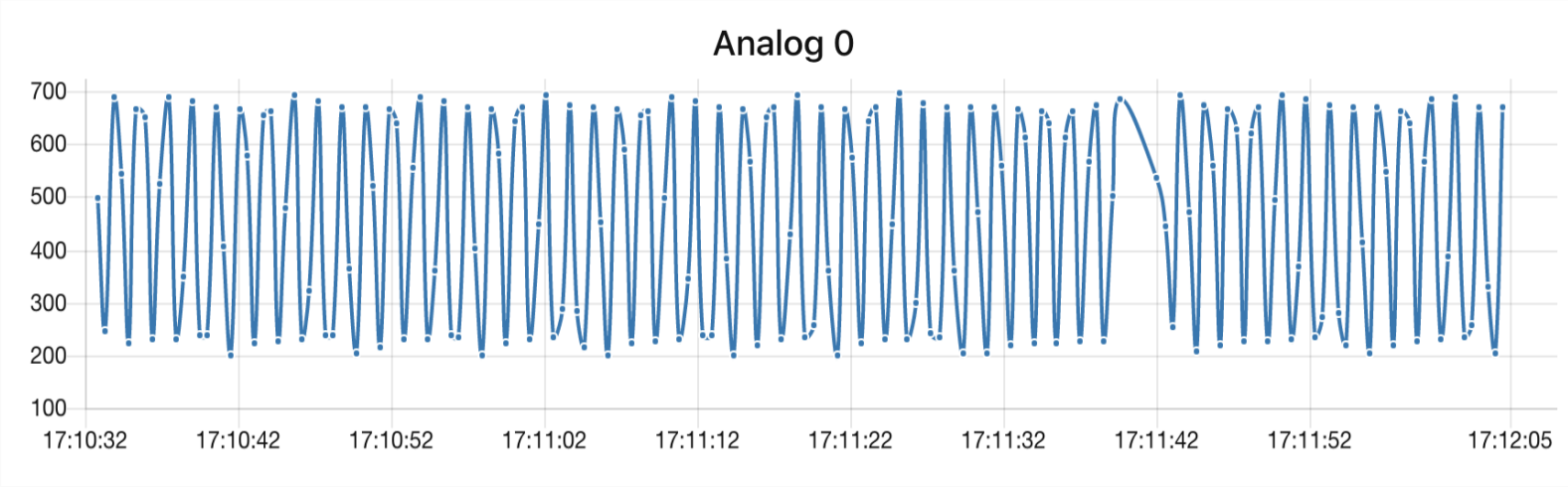
Observing JSON data via Dashboard

- Double click on the chart node
- Add a name: Analog 0
- Click Add

Observing JSON data via Dashboard

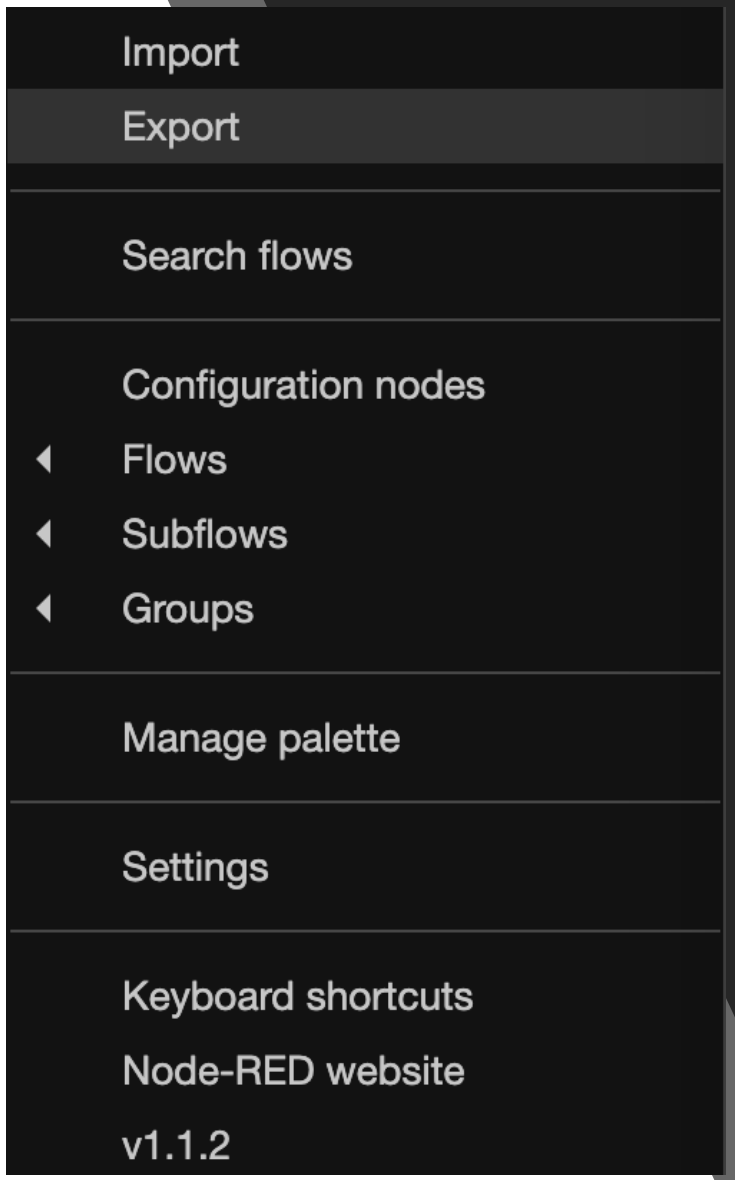
- Add a name: Analog 0
- Click Done





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



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