Node-RED

What is Node-RED?

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

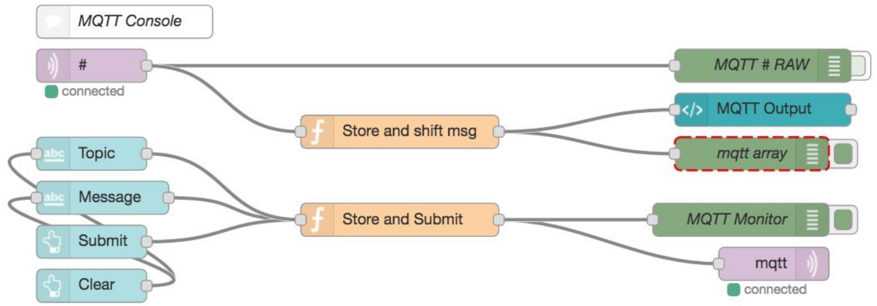
It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

Browser-based flow editing

Node-RED provides a browser-based flow editor that makes it easy to wire together flows using the wide range of nodes in the palette. Flows can be then deployed to the runtime in a single-click.

JavaScript and Python functions can be created within the editor using a rich text editor.

A built-in library allows you to save useful functions, templates or flows for re-use.



Built on Node.js

Node-RED is built on `Node.js`, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

With over 225,000 modules in Node's package repository, it is easy to extend the range of palette nodes to add new capabilities.

Requirements: Install Node

The first step is install `Node.js`.

- Windows
  - Download the latest 12.x LTS version of Node.js from the official Node.js download page.
- MacOS
  - Download the latest 12.x LTS version of Node.js from the official Node.js download page or if you have `brew` installed, just execute on the terminal the command `brew install node`
- Linux
  - Use `package manager` to install it

Requirements: Install Node-RED

Installing Node-RED as a global module adds the command node-red to your system path. Execute the following at the command prompt

Terminal

```
npm install -g –unsafe-perm node-red
```

Alternative: Docker

Instead of a local installation, if you want, you can work with Node-RED by running it in a simple Docker container. To do so, you need to run the following command:

Terminal

```
docker run -it -p 1880:1880 --name mynodered nodered/node-red
```

When working with Node-RED in Docker, you must be aware of how you are storing the flows and any information. You should make your data to persist no matter if your container is destroyed. For this you should store the data on an a “docker volume” outside the container. You can find more information in this guide

Alternative: Install Node-RED on cloud server

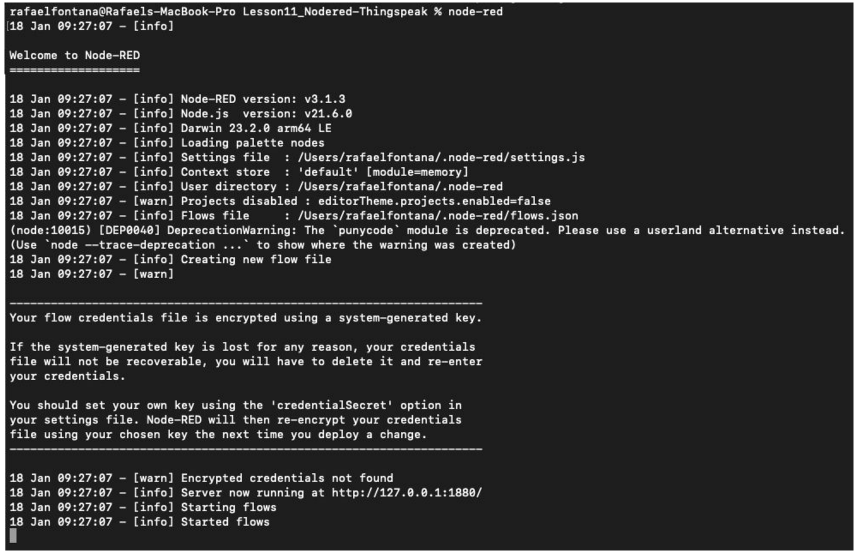
If you'd like to have your Node-RED application reachable from everywhere, the best choice is to install it on the cloud. If you wanna do it please refer to the following option:

- Microsoft Azure

Run Node-RED

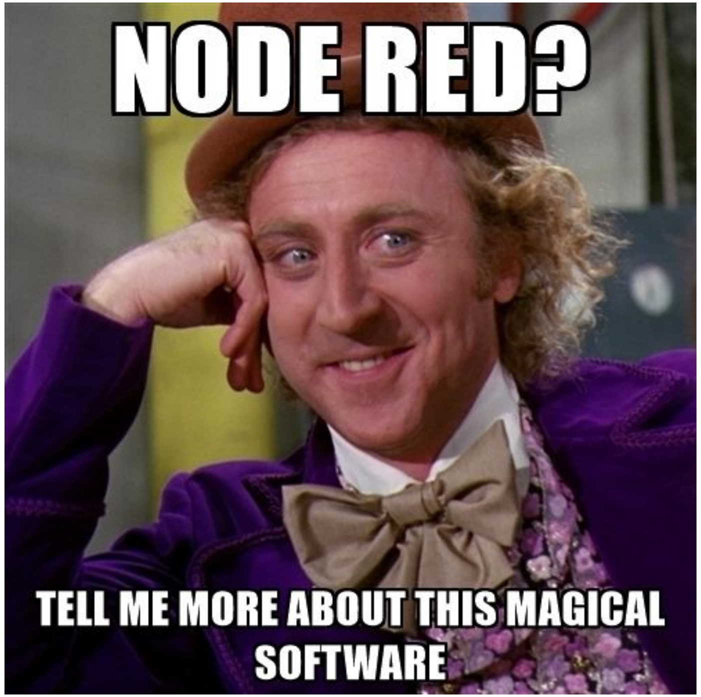
Once installed as a global module you can use the `node-red` command to start Node-RED in your terminal. You can use `Ctrl-C` or close the terminal window to stop Node-RED.

You can then access the Node-RED editor by pointing your browser at http://localhost:1880/



```
rafaelfontana@Rafaels-MacBook-Pro Lesson11_Nodered-Thingspeak % node-red
18 Jan 09:27:07 - [info]

Welcome to Node-RED
===================

18 Jan 09:27:07 - [info] Node-RED version: v3.1.3
18 Jan 09:27:07 - [info] Node.js  version: v21.6.0
18 Jan 09:27:07 - [info] Darwin 23.2.0 arm64 LE
18 Jan 09:27:07 - [info] Loading palette nodes
18 Jan 09:27:07 - [info] Settings file  : /Users/rafaelfontana/.node-red/settings.js
18 Jan 09:27:07 - [info] Context store  : 'default' [module=memory]
18 Jan 09:27:07 - [info] User directory : /Users/rafaelfontana/.node-red
18 Jan 09:27:07 - [warn] Projects disabled : editorTheme.projects.enabled=false
18 Jan 09:27:07 - [info] Flows file     : /Users/rafaelfontana/.node-red/flows.json
(node:10015) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
18 Jan 09:27:07 - [info] Creating new flow file
18 Jan 09:27:07 - [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.
---------------------------------------------------------------------

18 Jan 09:27:07 - [warn] Encrypted credentials not found
18 Jan 09:27:07 - [info] Server now running at http://127.0.0.1:1880/
18 Jan 09:27:07 - [info] Starting flows
18 Jan 09:27:07 - [info] Started flows
```

What actually Node-RED is?

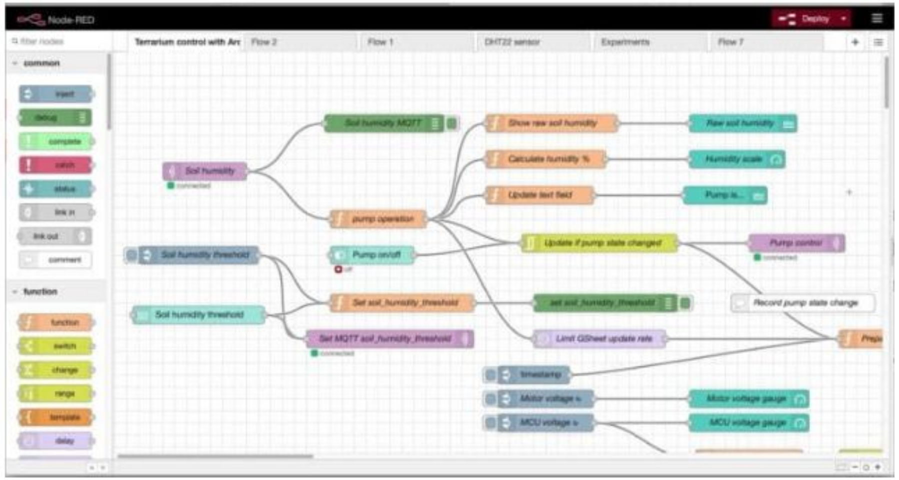


What actually Node-RED is?

Node-RED is a programming tool. Actually, a better term to describe it is as a "programming environment".

You use Node-RED to create graphical programs, which are called flows.

Flows are composed of nodes, which are rectangular objects that you see in the example on the right.



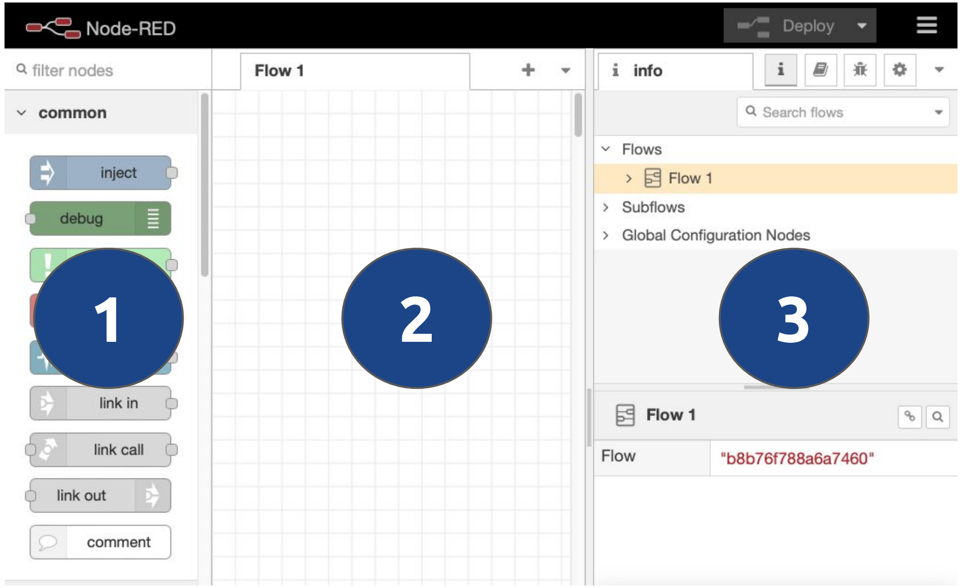
Node-RED PAGE

Node-RED is a programming tool. Actually, a better term to describe it is as a "programming environment".

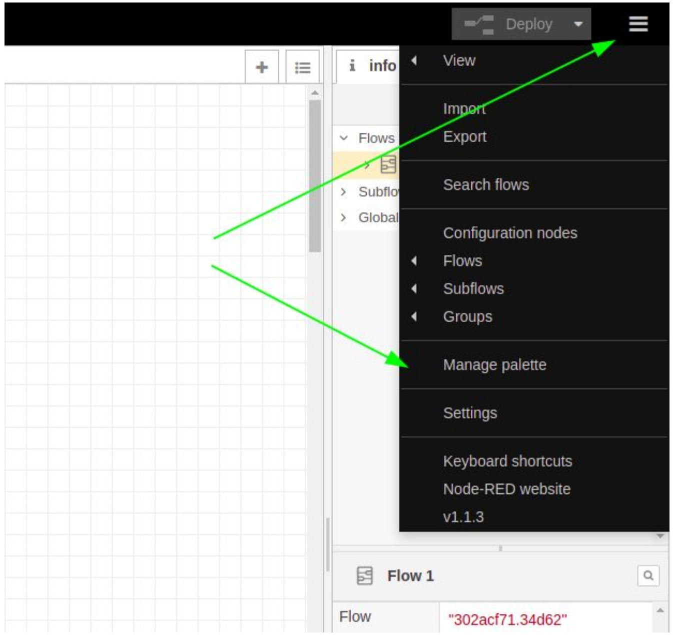
1. Node Panel: contains a list of nodes that you can use. Nodes are organized by categories, that include “input”, “output”, “function”. Node-RED comes with a selection of nodes you can use out of the box

2. Sheets Panel: main working space. You create Node-RED “flows” by placing nodes in sheets and wiring them together. To place a node on a Sheet you drag and drop that node from the Node Panel (1) onto the Sheet Panel (2)

3. Info and Debug Panel: contains two tabs: “info” and “debug”. The info tab provides information about a node when it’s clicked on the Sheet Panel (2); such information can help you understand how such node functions and gives you information about its type, properties, usage and required configuration. The debug tab provides information printed with the “debug” node, to print to the debug console you must have a debug node in the Sheet Panel (2)



Install additional libraries



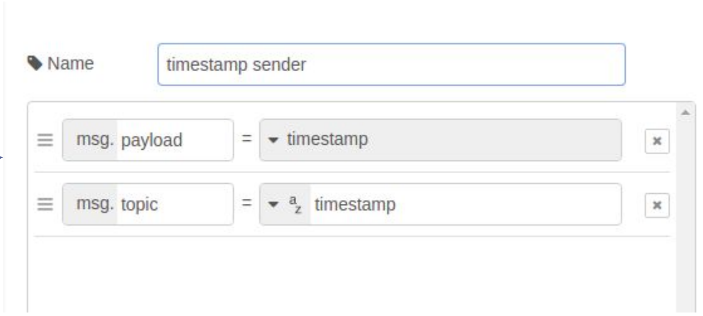
Using the manage palette from the toolbar on the top right on the screen is possible to install additional library that will add new nodes that can be used. For our objective we need to install the following libraries:

- `node-red-contrib-python3-function`
- `node-red-dashboard`

Our First Node-RED flow

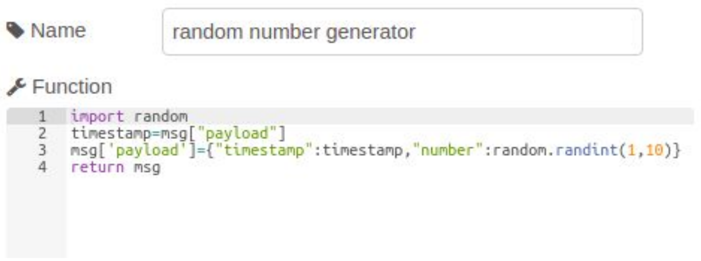
To create our first flow we need 3 nodes:

1. The `inject` node
2. The `python3function` node
3. The `debug` node


Name: timestamp sender

msg. payload = timestamp

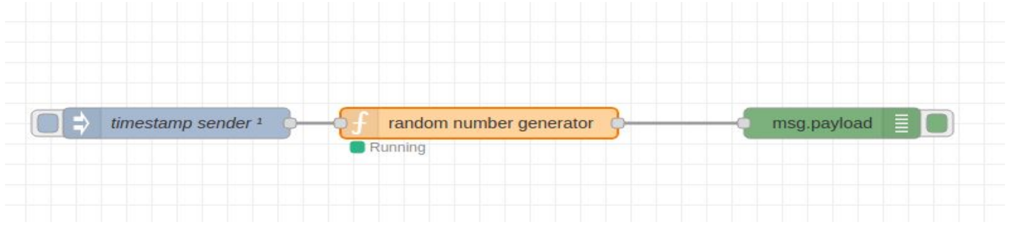
msg. topic = timestamp


Name: random number generator

Function

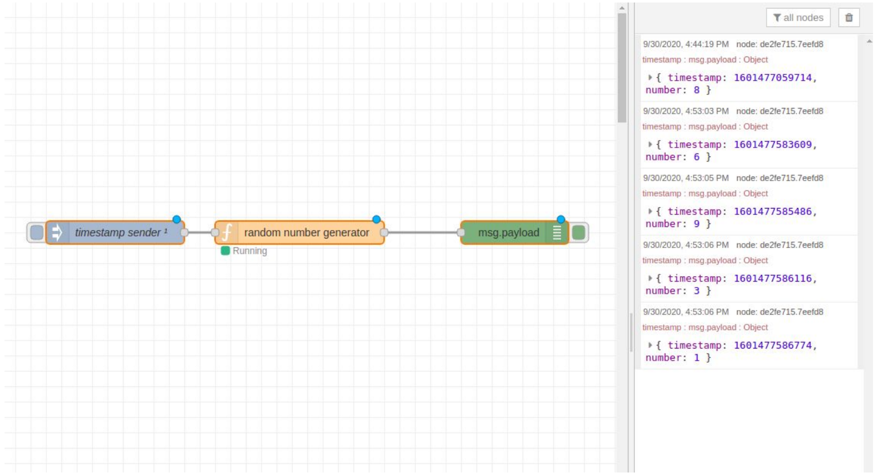
```
import random
timestamp=msg["payload"]
msg['payload']={"timestamp":timestamp,"number":random.randint(1,10)}
return msg
```

We will connect these 3 nodes as shown below



Our First Node-RED flow

To start the flow click on the button “Deploy” on the top right corner. Then if you click on the square to the left you will be able to see the result of the flow execution in the debug sidebar on the right, each time you will click it the ‘inject’ node will send a message that will be processed by the flow. You could also configure the ‘inject’ node to automatically send a message with a custom time interval



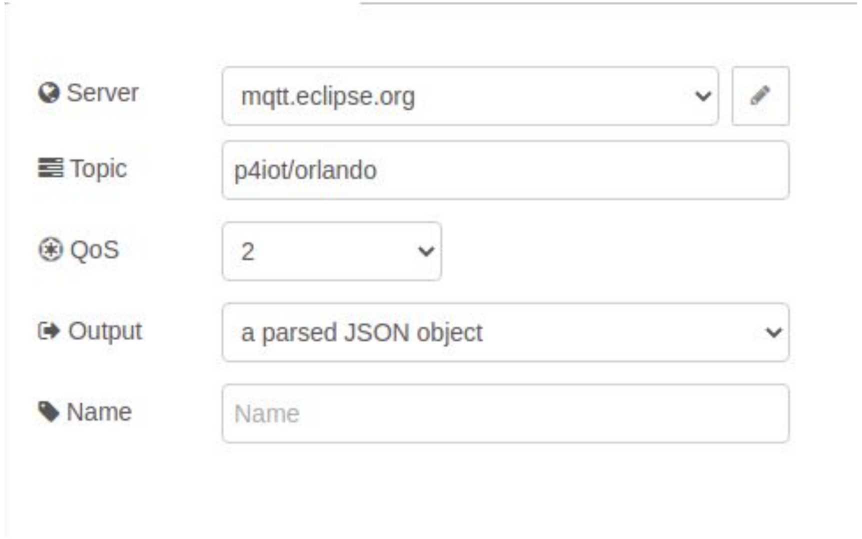
```
9/30/2020, 4:44:19 PM   node: de2fe715.7eefd8
timestamp : msg.payload : Object
 ▸{ timestamp: 1601477059714,
number: 8 }
9/30/2020, 4:53:03 PM   node: de2fe715.7eefd8
timestamp : msg.payload : Object
 ▸{ timestamp: 1601477583609,
number: 6 }
9/30/2020, 4:53:05 PM   node: de2fe715.7eefd8
timestamp : msg.payload : Object
 ▸{ timestamp: 1601477585486,
number: 9 }
9/30/2020, 4:53:06 PM   node: de2fe715.7eefd8
timestamp : msg.payload : Object
 ▸{ timestamp: 1601477586116,
number: 3 }
9/30/2020, 4:53:06 PM   node: de2fe715.7eefd8
timestamp : msg.payload : Object
 ▸{ timestamp: 1601477586774,
number: 1 }
```

Exercise - MQTT in Node-RED

If we want to integrate To add an MQTT subscriber to our flow we can do as follow

Add an `MQTT in` node to the flow and configure it (change the topic according to your needs)

As exercise ,to test your MQTT subscriber, try to send the data from one of the MQTT simulator you've created during the course



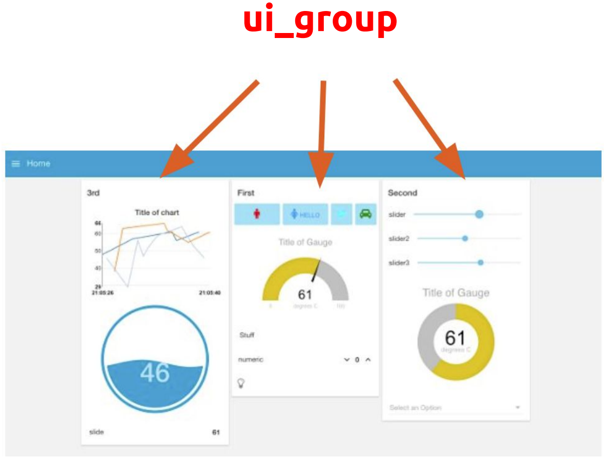
Server	mqtt.eclipse.org
Topic	p4iot/orlando
QoS	2
Output	a parsed JSON object
Name	Name

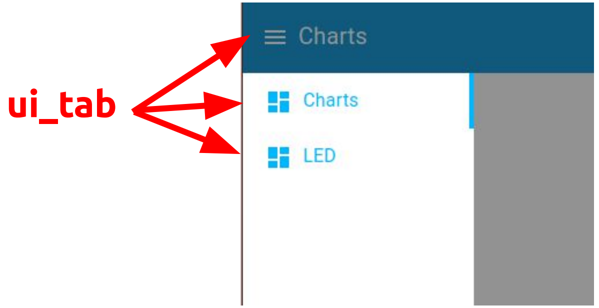
UI : node-RED-dashboard

The `node-red-dashboard` library we installed before will help us to create simple but effective dashboards. This dashboards can be exploited by the user to interact with the application or just to see the data collected by the system.

There are two main concepts to keep in mind:

- ui_tab : a standalone page containing different ui_group
- ui_group: a "column" containing the different graphic elements (plot, gauge etc..)





Example: Interactive LED status

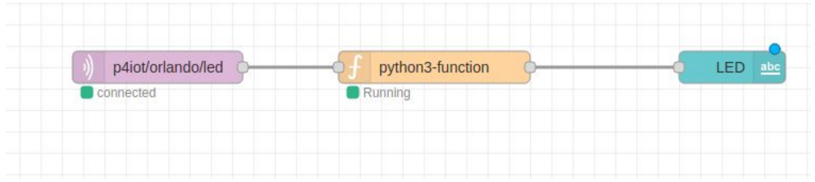
To simulate an actuation message for the led you use the result of one of the exercises done in class, write your own code on your pc, or create a new flow similar to the first one but with a different code (see right) fo the python node

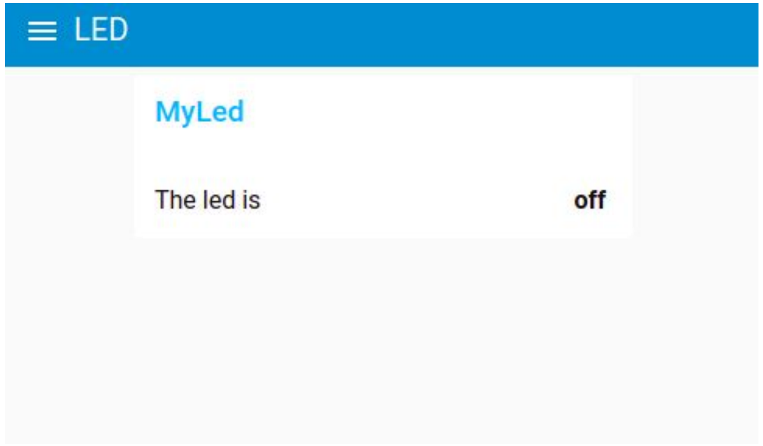
```
import random
msg["payload"]={
        'bn':'p4IoT/1/1',
        'e':[{
                'n':'status',
                'v':round(random.random()),
                't':str(msg["payload"]),
                'unit':'bool'
                }]
}
return msg
```

Example: Interactive LED status

We just need to add a `text-input` node that will receive the message from the `python` node and will display the text *"The led is on/off"*. The steps to follow in order to customize the `text-input` node are shown in the following slide.

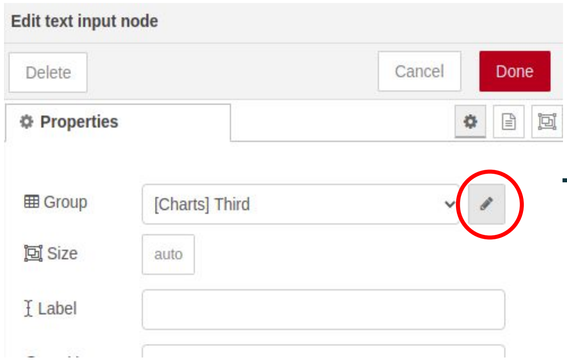
It is possible to use something like a circle button which changes the LED color (instead of text). For more details, you can follow this guide. For the moment, we will keep it simple.





The led is off

Example: Interactive LED status

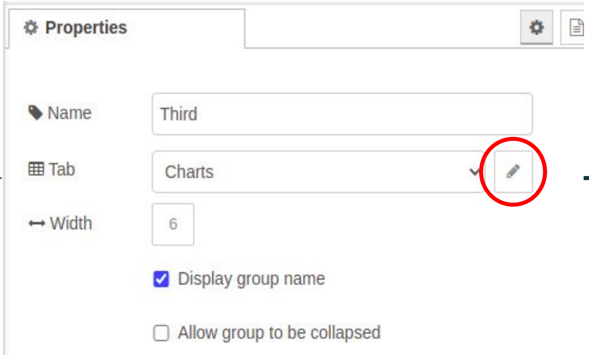


Edit text input node

Delete | Cancel | Done

Properties

Group	[Charts] Third
Size	auto
Label	

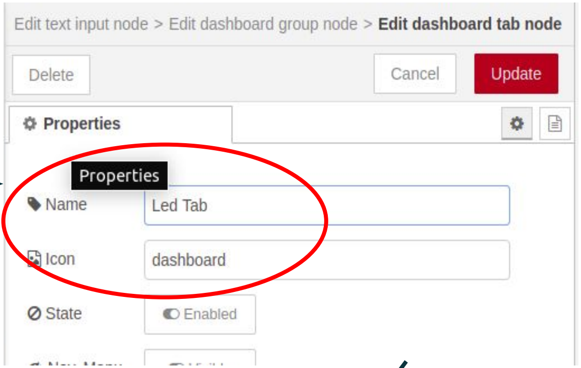


Properties

Name	Third
Tab	Charts
Width	6

☑ Display group name

☐ Allow group to be collapsed

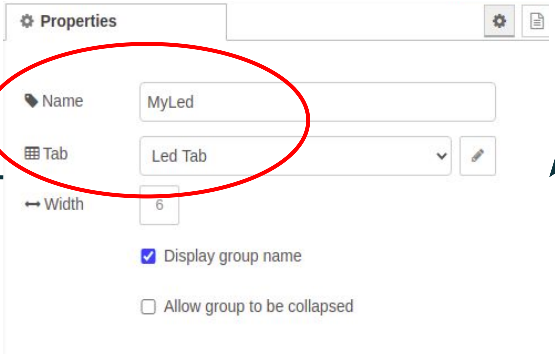


Edit text input node > Edit dashboard group node > **Edit dashboard tab node**

Delete | Cancel | Update

Properties

Name	Led Tab
Icon	dashboard
State	Enabled

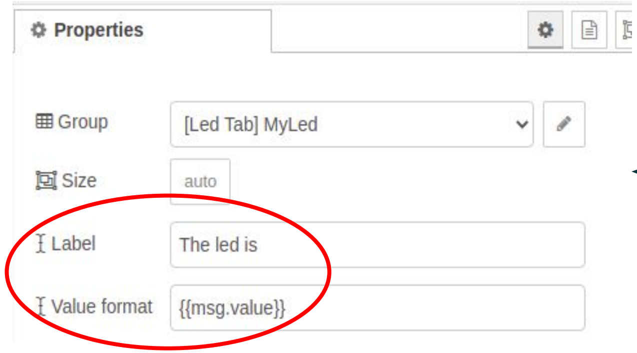


Properties

Name	MyLed
Tab	Led Tab
Width	6

☑ Display group name

☐ Allow group to be collapsed

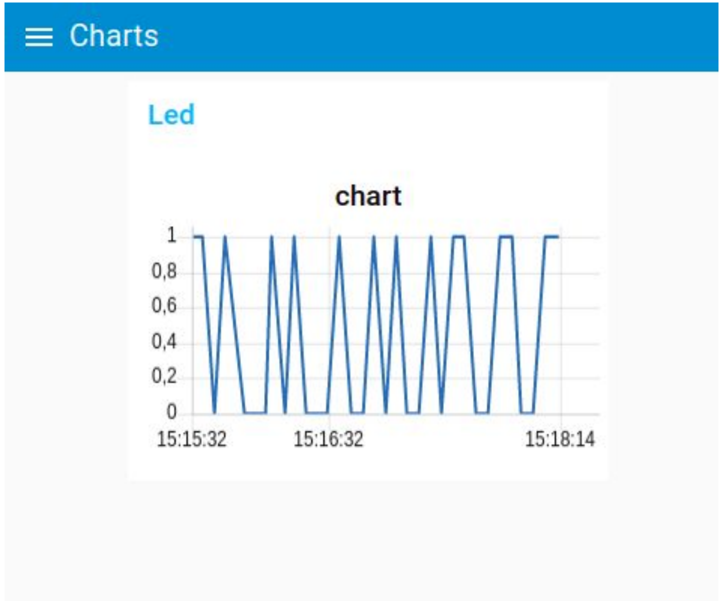


Properties

Group	[Led Tab] MyLed
Size	auto
Label	The led is
Value format	{{msg.value}}

Exercise - Plot data 1/2

We will now try to plot some data coming from MQTT messages. In this example, we will just plot the value for the status of the LED we used in the previous example. Nonetheless, the same rules apply for any other numerical data

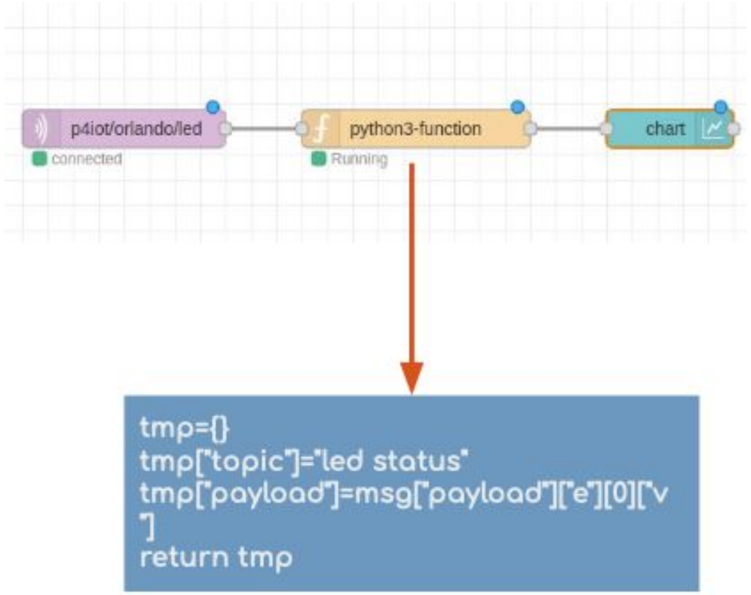


Exercise - Plot data 2/2

The main difference from before will be just the code of the python-node. The chart node indeed needs to receive a message with the following format:

```
{topic: “<name_of_series”>, payload: <value>}
```

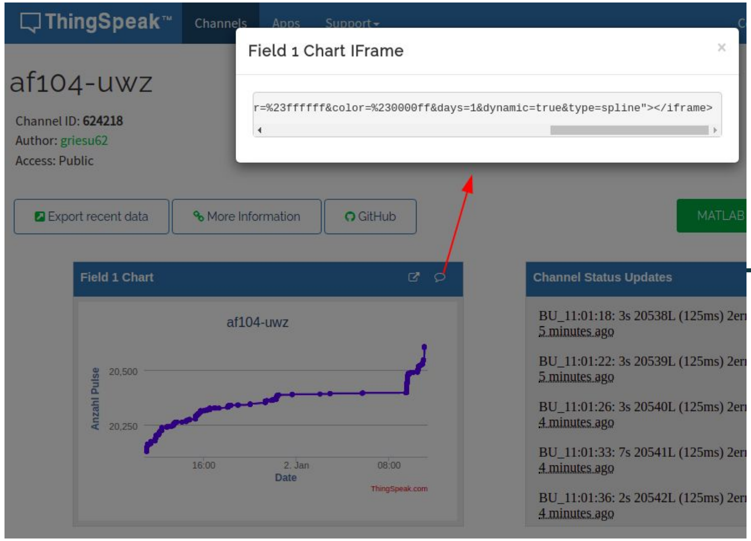
It will create a series for each topic and everytime a new message will arrive a new point will be added to the corresponding series.



```
tmp={}
tmp["topic"]="led status"
tmp["payload"]=msg["payload"]["e"][0]["v
"]
return tmp
```

Include Thingspeak Plots

Copy



Field 1 Chart IFrame

```
r=%23ffffff&color=%230000ff&days=1&dynamic=true&type=spline"></iframe>
```

af104-uwz

Channel ID: 624218
Author: griesu62
Access: Public

Channel Status Updates

BU_11:01:18: 3s 20538L (125ms) 2er
5 minutes ago

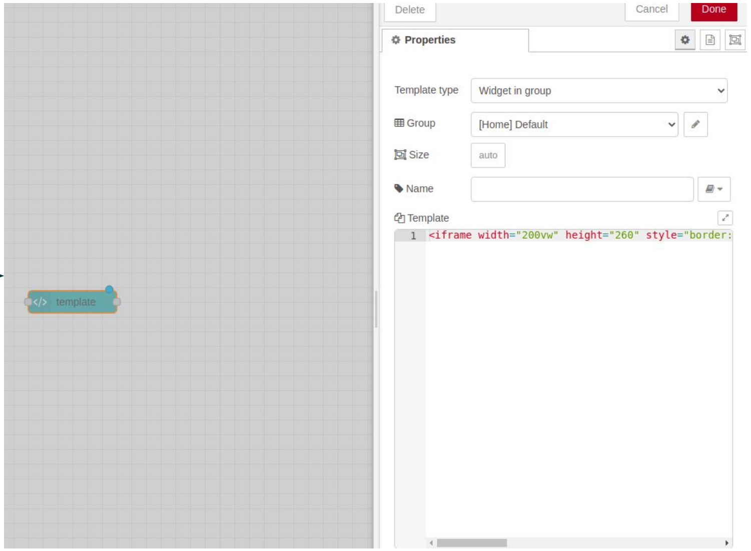
BU_11:01:22: 3s 20539L (125ms) 2er
5 minutes ago

BU_11:01:26: 3s 20540L (125ms) 2er
4 minutes ago

BU_11:01:33: 7s 20541L (125ms) 2er
4 minutes ago

BU_11:01:36: 2s 20542L (125ms) 2er
4 minutes ago

Insert "template" and Paste



Template type: Widget in group
Group: [Home] Default
Size: auto
Name:
Template

```
<iframe width="200vw" height="260" style="border:
```

Other UI options: StreamLit

Streamlit is an open-source Python library that is used for easy creating web applications. It is possible to create web applications using only Python code, without the need for HTML, CSS, or JavaScript. You can find more information here