Node-Red

Raspberry pi-IoT

Dr. Sarwan Singh
Deputy Director(S)
NIELIT Chandigarh

agenda

- About Node-Red
- History
- Architecture
- Building your first flows
- Installing Node-Red-Dashboard
- ESP8266 and Node-RED with MQTT

About ..

- Node-RED is a flow-based development tool developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things.
- Node-RED provides a browser-based flow editor, which can be used to create **JavaScript** functions.
- Elements of applications can be saved or shared for re-use.
- The runtime is built on **Node.js**.
- The flows created in Node-RED are stored using **JSON**.
- In 2016, IBM contributed Node-RED as an **open source** *JS Foundation* project.

Source:wikipedia

History

- Flow-based Programming
  - Invented by J. Paul Morrison in the 1970s, flow-based programming is a way of describing an application's behaviour as a network of black-boxes, or "nodes" as they are called in Node-RED.
  - Each node has a well-defined purpose; it is given some data, it does something with that data and then it passes that data on.
  - The network is responsible for the flow of data between the nodes.

Source: nodered.org

History

- Node-RED started life in early 2013 as a side-project by Nick O'Leary and Dave Conway-Jones of IBM's Emerging Technology Services group.
- What began as a proof-of-concept for visualising and manipulating mappings between MQTT topics, quickly became a much more general tool that could be easily extended in any direction.
- It was open-sourced in September 2013 and has been developed in the open ever since, culminating in it being one of the founding projects of the JS Foundation in October 2016.

Source: nodered.org

Node-RED is

- An application composition tool experience
- A lightweight proof of concept runtime
- Easy to use for simple tasks
- Simple to extend to add new capabilities and types of integration
- Capable of creating the back-end glue between social applications
- A great way to try…
  - "can I just get this data from here to there?"
  - "and maybe change it just slightly along the way…"

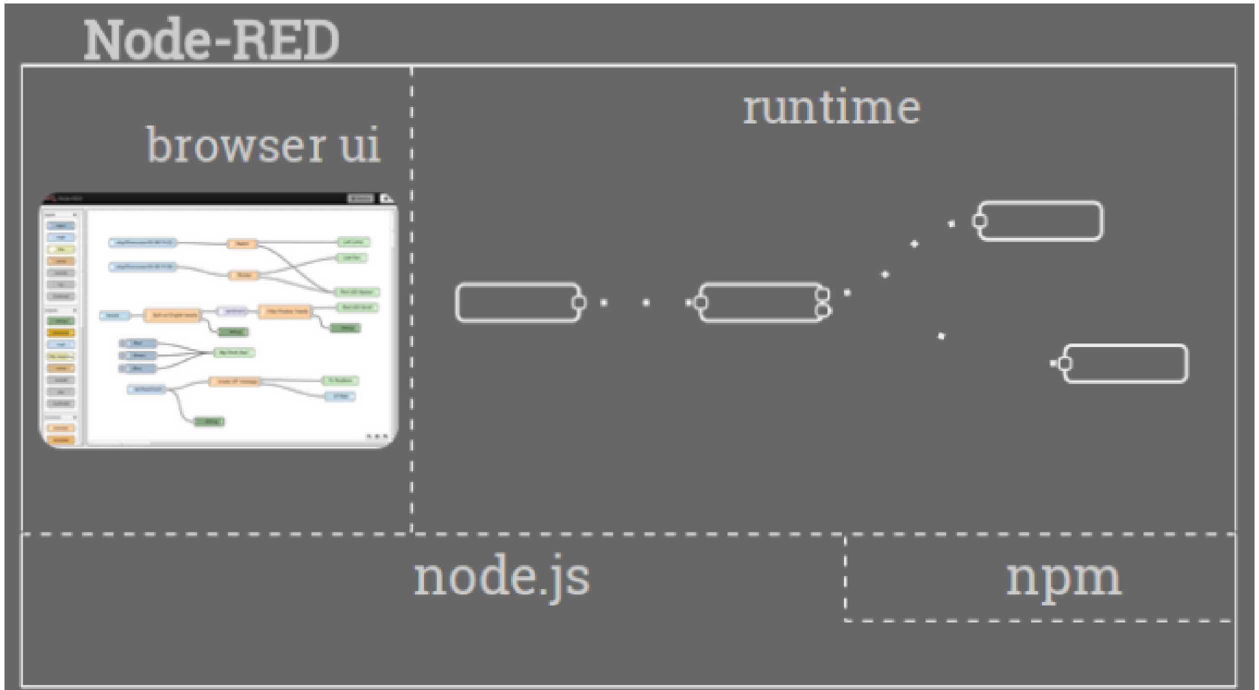
Node-RED is not

- A fully-scalable, high-performance, enterprise-capable application runtime
- A dashboard with widgets
- A mobile application builder

  - The answer to life, the universe, and everything…

Architecture of Node-RED

- Node.js v8-engine driven; so it's fast
- Event-driven, asynchronous io; it's all about the events
- Single-threaded event queue; built for fairness
- Javascript front and back; only one language runtime to deal with
- Built using express, d3, jquery and ws

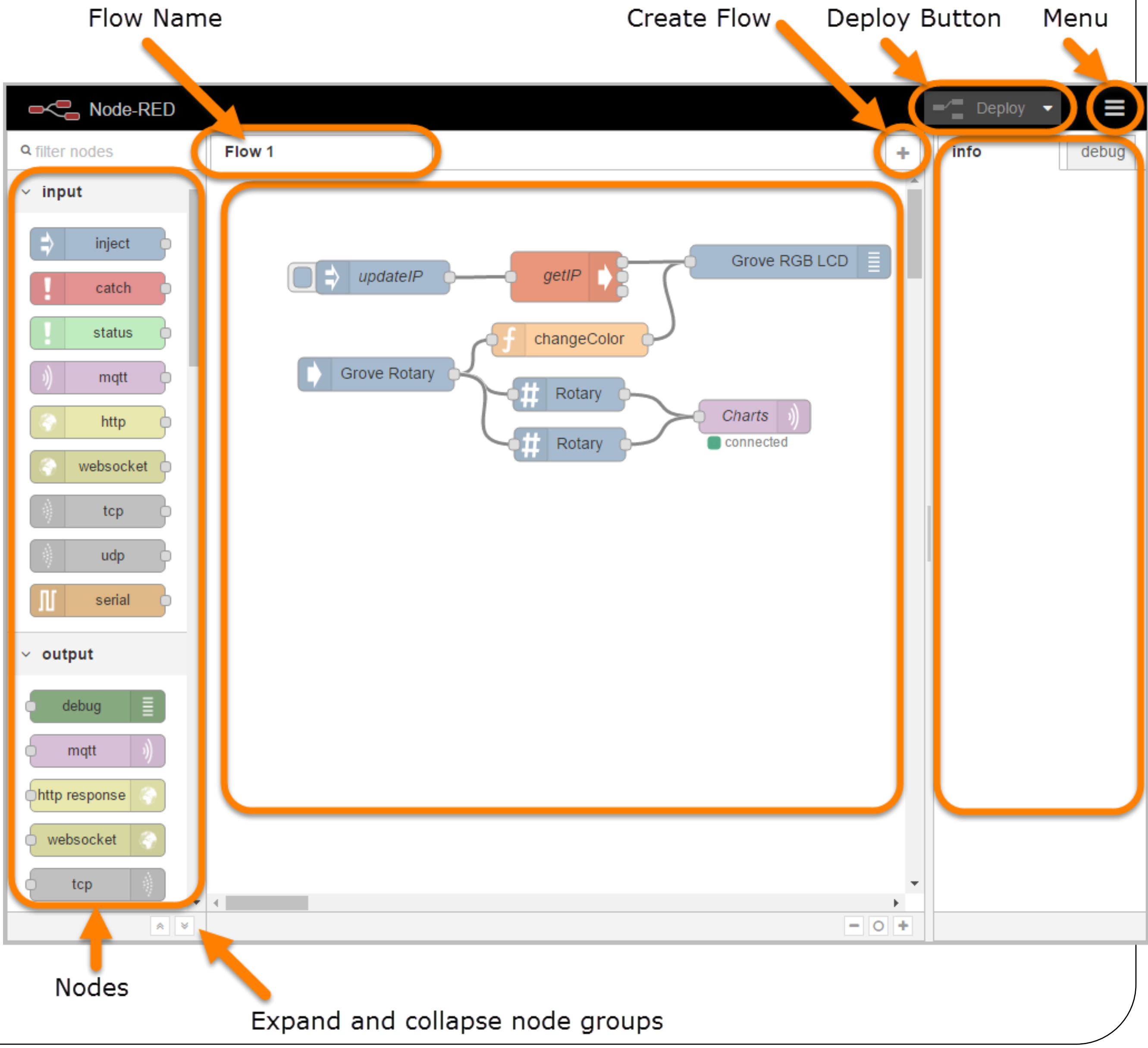


Building your first flows

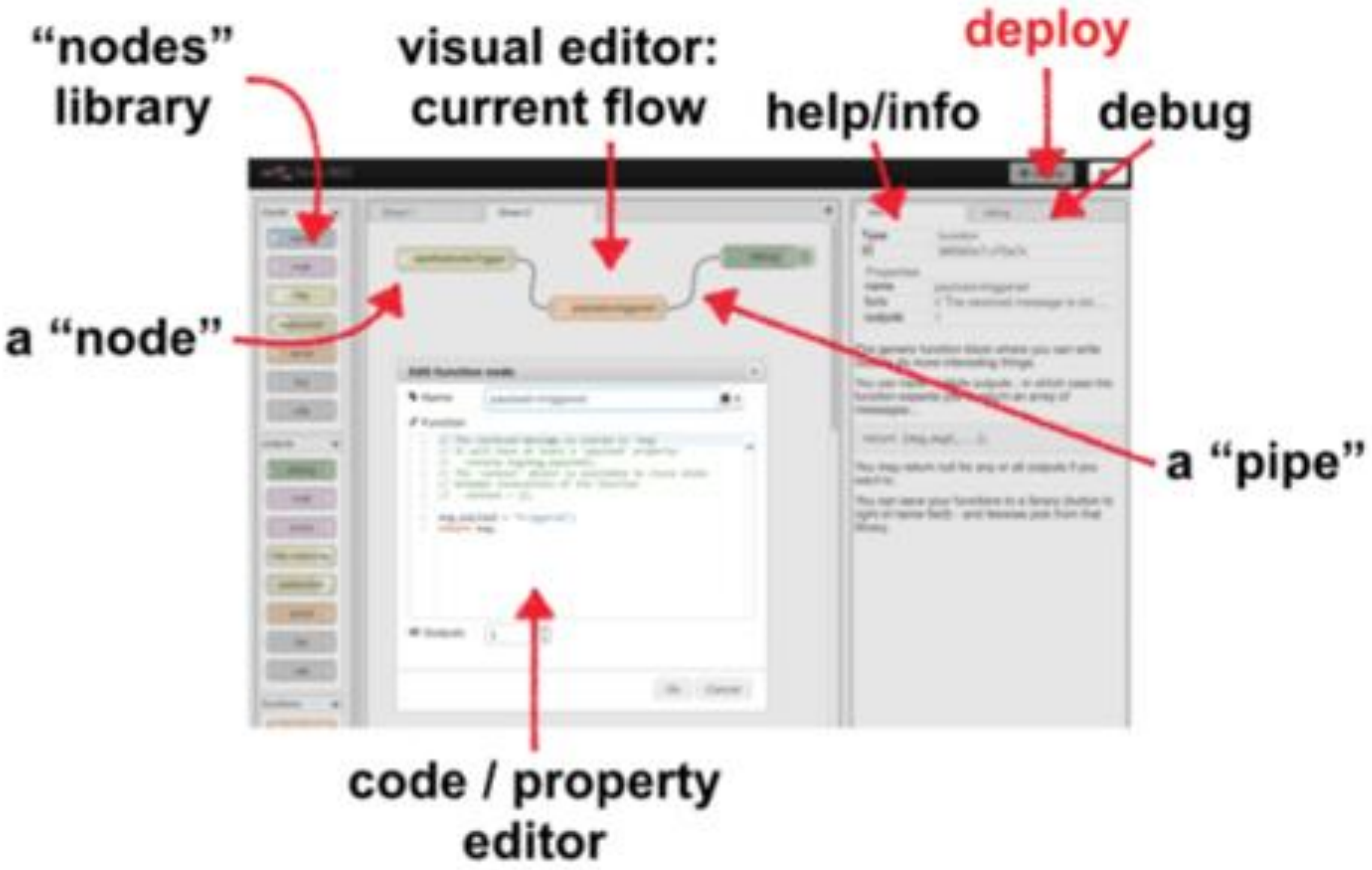
FRED –
Frontend for
Node-RED-
cloud base
utility

or

Raspberry Pi



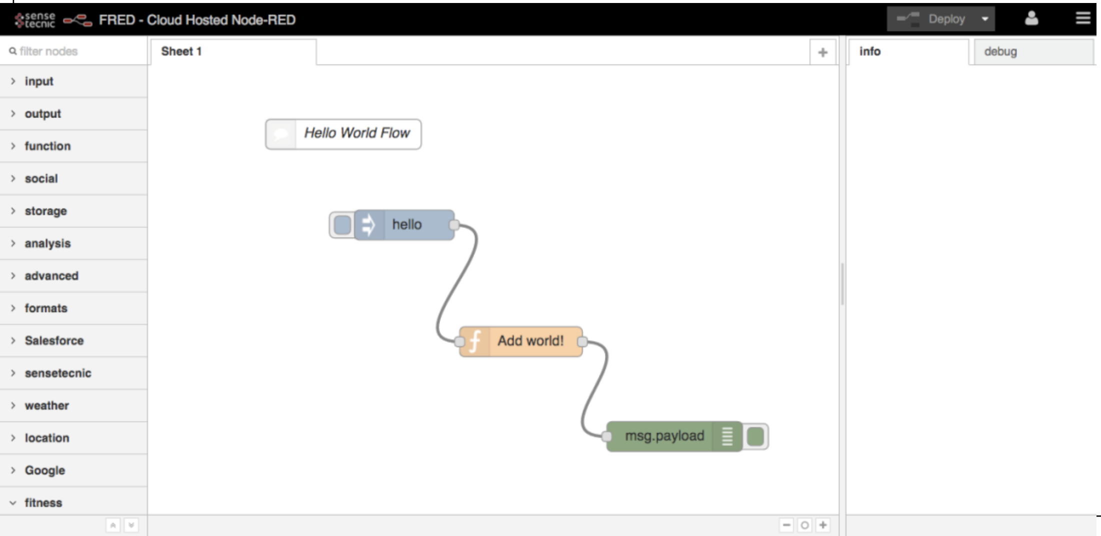
Building your first flows

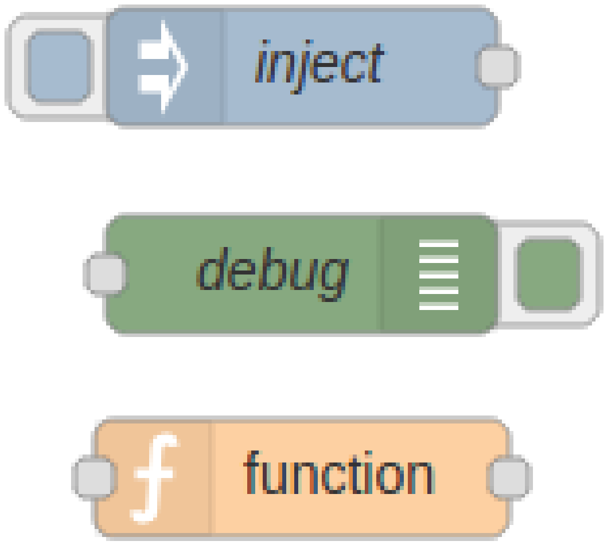


Node-RED nodes and messages

There are three main types of nodes:

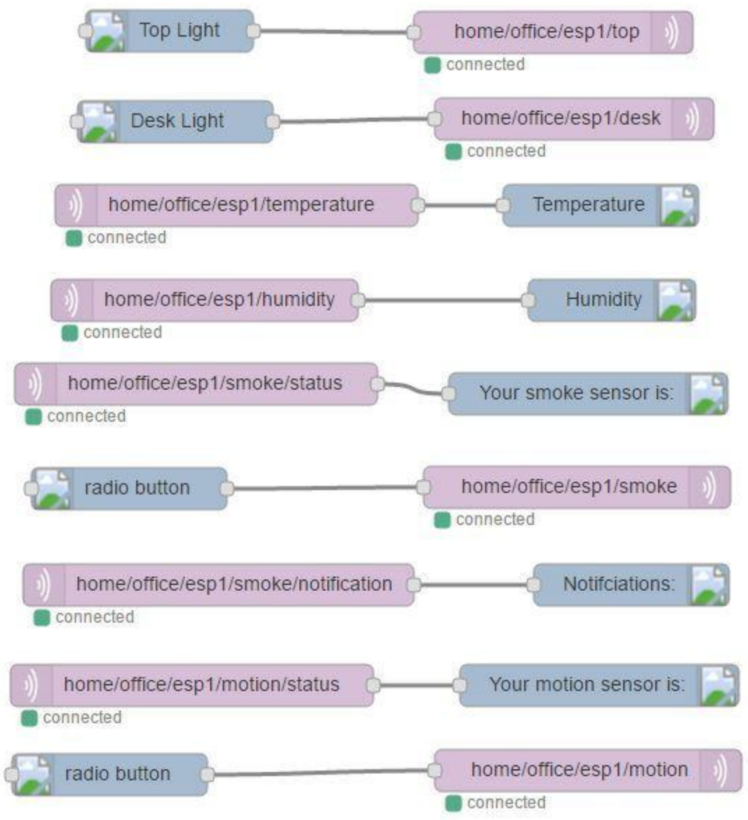
- Input Nodes (e.g. inject)
- Output Nodes (e.g. debug)
- Processing Nodes (e.g. function)

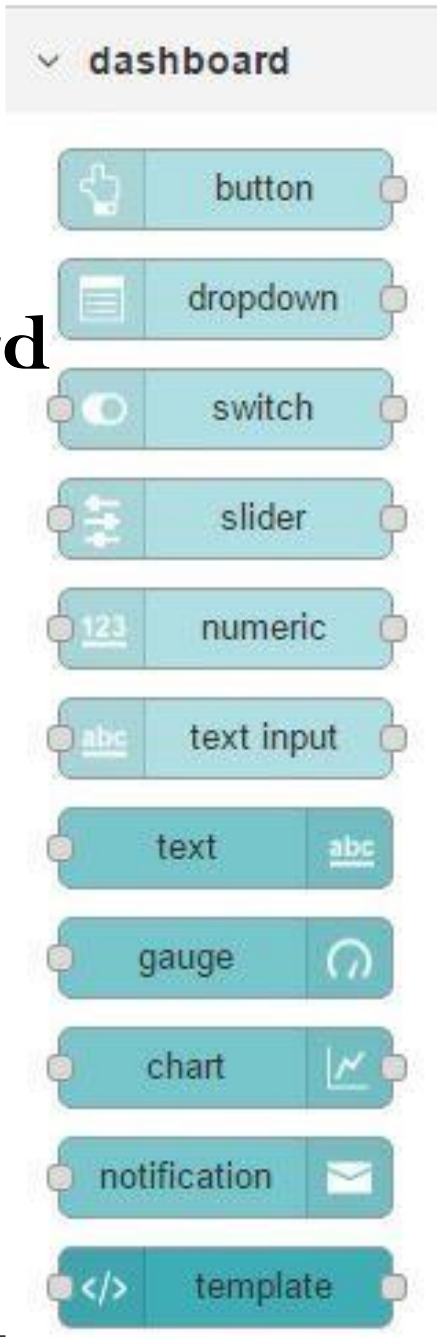




Installing Node-Red-Dashboard

```
$ sudo apt-get install npm
$ sudo npm install node-red-dashboard
$ sudo reboot
```

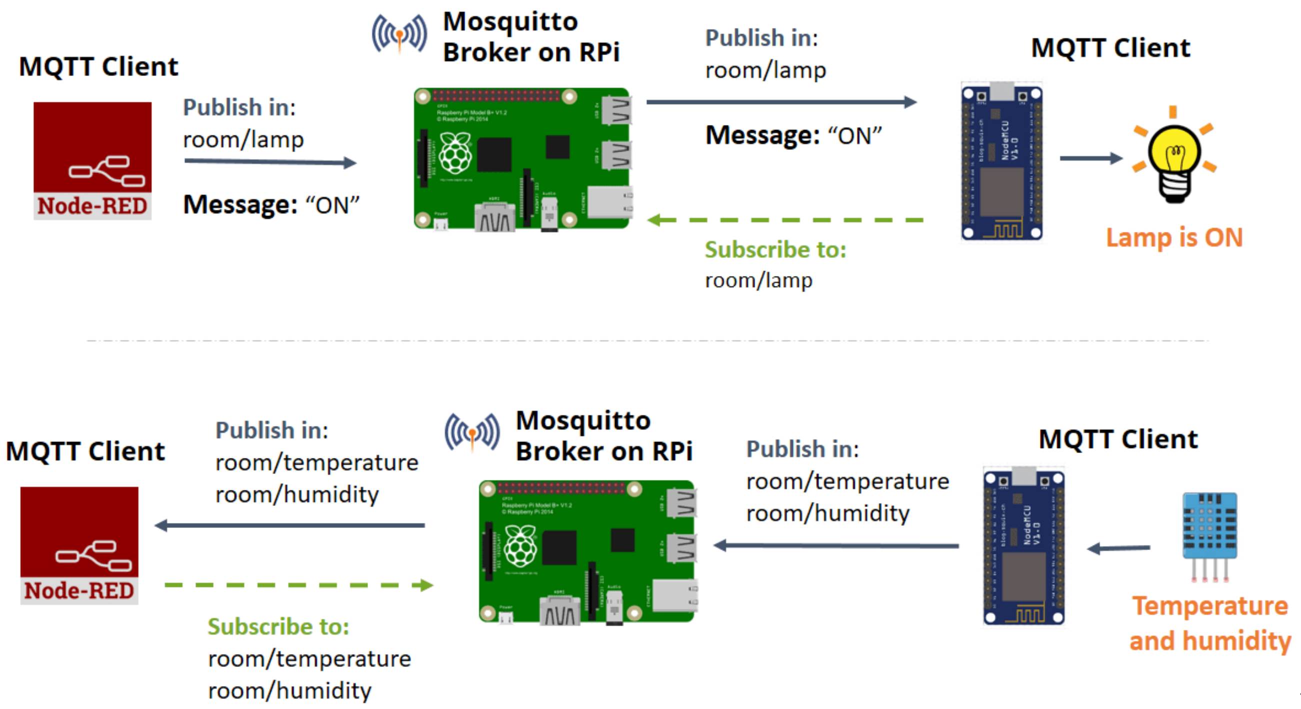




ESP8266 and Node-RED with MQTT

(Publish and Subscribe)

- Installing Mosquitto Broker
- Establishing an MQTT communication with Node-RED
- Preparing your Arduino IDE



Installing Mosquitto Broker

- In MQTT, the broker is primarily responsible for **receiving** all messages, **filtering** the messages, **decide** who is interested in it and then **publishing** the message to all subscribed clients.
- **Mosquitto Broker** to be installed on Raspberry Pi.

```
$ sudo apt install mosquitto
$ sudo systemctl enable mosquitto.service
```

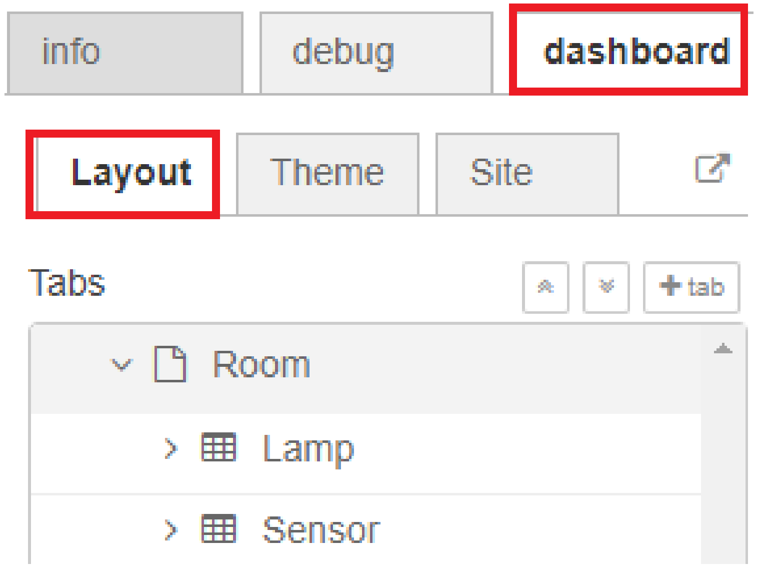
To see if Mosquitto broker was successfully installed

```
$ mosquitto -v
```

Node-Red flow

- Dashboard Layout

On the top right corner of the Node-RED window, select the **Layout** tab under the **dashboard** tab. Create a tab called **Room** and inside the Room tab, create two groups: **Lamp** and **Sensor**

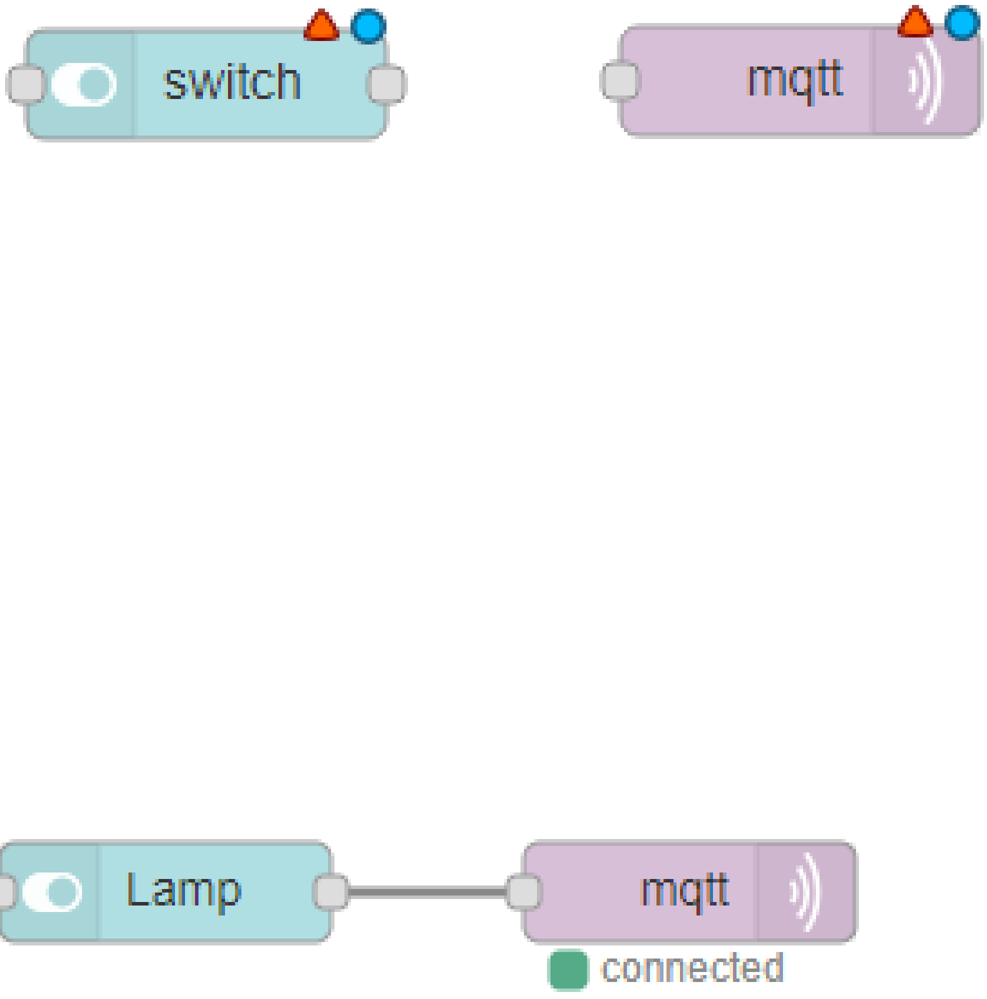


Node-Red flow

- Creating Flow

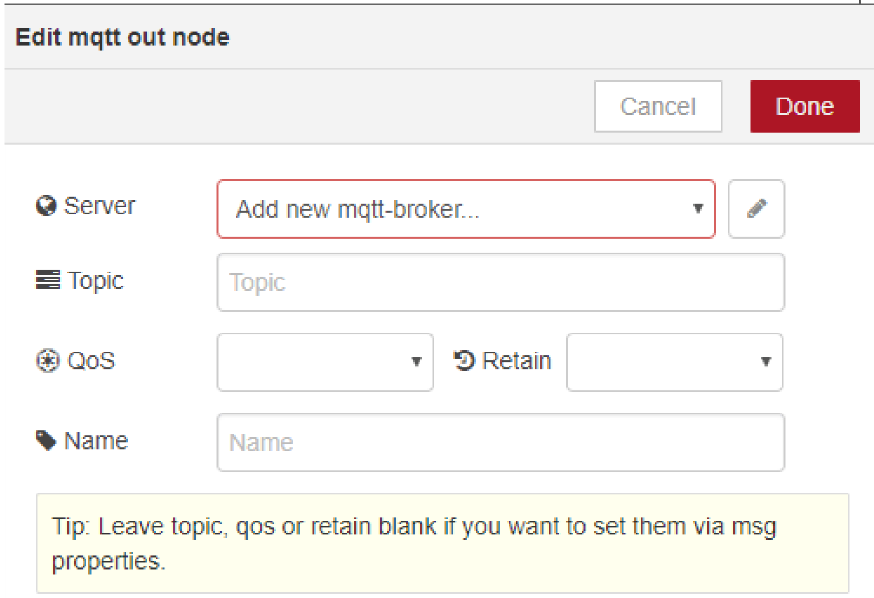
switch – this will control the ESP8266 output

mqtt output node – this will publish a message to the ESP8266 accordingly to the switch state



Node-Red flow

- Click the **Add new mqtt-broker** option.
- Type **localhost** in the server field
- All the other settings are configured properly by default.
- Press **Add** and the MQTT output node automatically connects to your broker.



Edit mqtt out node

Cancel Done

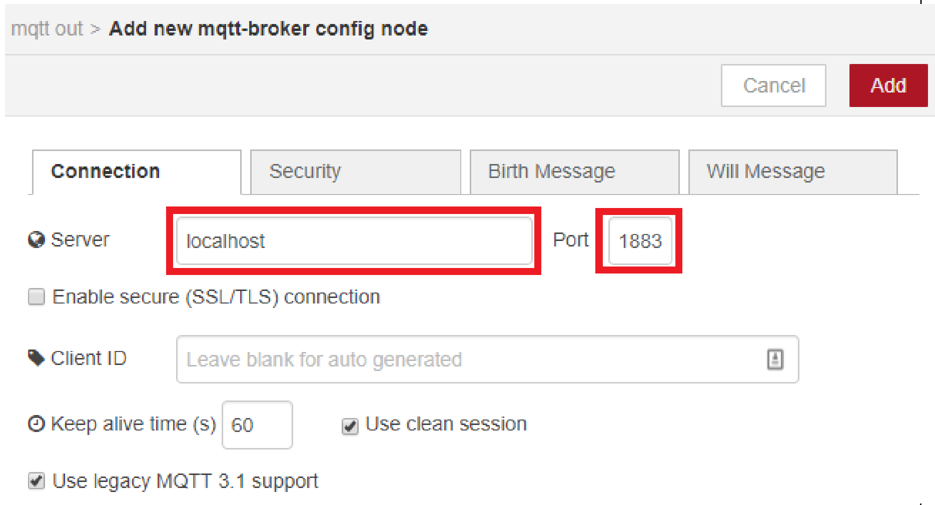
Server: Add new mqtt-broker...

Topic: Topic

QoS: Retain:

Name: Name

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



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

Cancel Add

Connection | Security | Birth Message | Will Message

Server: localhost Port: 1883

☐ Enable secure (SSL/TLS) connection

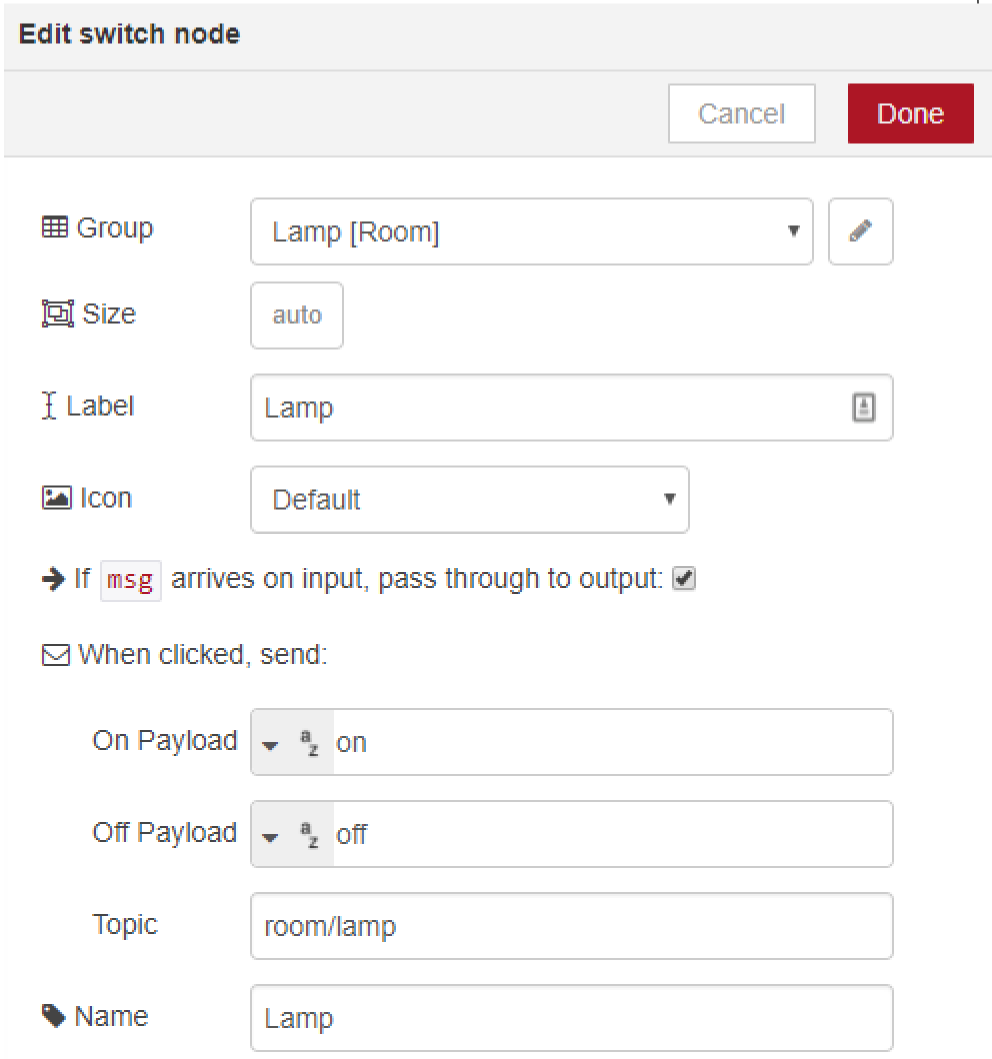
Client ID: Leave blank for auto generated

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

☑ Use legacy MQTT 3.1 support

Node-Red flow

- **switch –** the switch sends an **on** string message when it's on; and sends an **off** string message when it's off. This node will publish on the **room/lamp** topic. ESP will subscribe to this topic, to receive its messages.



Edit switch node

Group	Lamp [Room]
Size	auto
Label	Lamp
Icon	Default

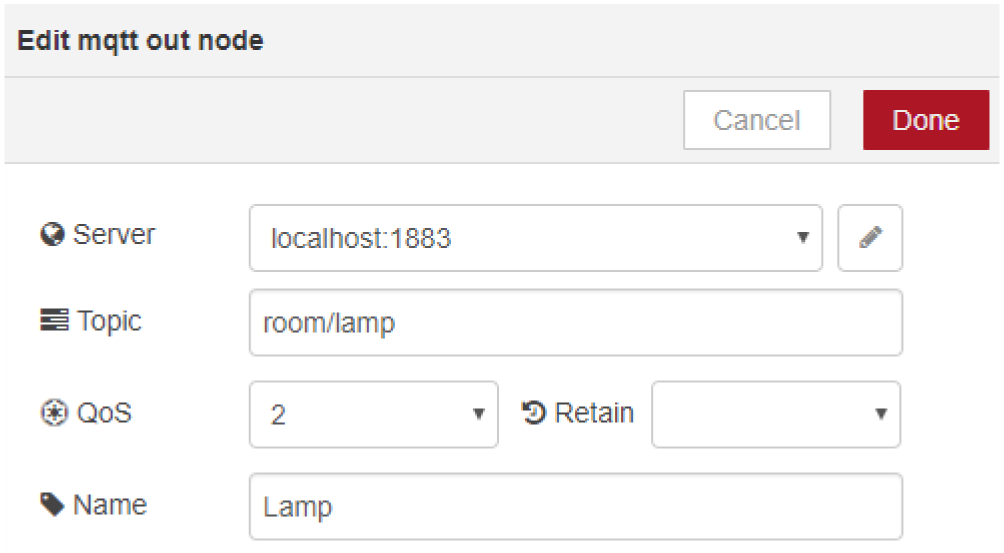
If msg arrives on input, pass through to output: ✓

When clicked, send:

On Payload	on
Off Payload	off
Topic	room/lamp
Name	Lamp

Node-Red flow

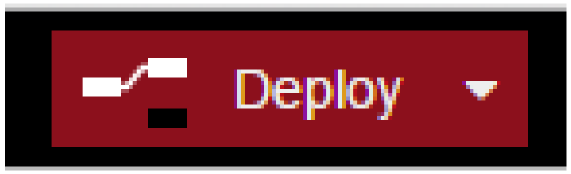
- **mqtt output node**. This node is connected to the mosquitto broker and it will publish in the **room/lamp** topic.


Edit mqtt out node

Server	localhost:1883
Topic	room/lamp
QoS	2
Retain	
Name	Lamp

Deploy

- Your Node-RED application is ready. Click the **Deploy** button on the top right corner.
- The Node-RED application is ready. To see how your dashboard looks go to ***http://your-pi-ip-address/ui***.



Arduino ESP8266

```
void callback(String topic, byte* message, unsigned int length) {

    for (int i = 0; i < length; i++) {
      Serial.print((char)message[i]);
      messageTemp += (char)message[i];
    }
    Serial.println();

    // Feel free to add more if statements to control more GPIOs with MQTT

    // If a message is received on the topic room/lamp, you check if the message
  is either on or off. Turns the lamp GPIO according to the message
    if(topic=="room/lamp"){
        Serial.print("Changing Room lamp to ");
        if(messageTemp == "on"){
          digitalWrite(lamp, HIGH);
          Serial.print("On");
        }
        else if(messageTemp == "off"){
          digitalWrite(lamp, LOW);
          Serial.print("Off");
        }
    }
    Serial.println();
  }
```