ASNlab MQTT example using Contiki and Node-RED

Introduction.

This tutorial is a very brief compilation from multiple sources on a subject of implementing MQTT on Contiki based WSN with 6lbr edge router. Node-Red is used as front end.

Basic terms and definitions.

What is MQTT? – "MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging "machine-to-machine" (M2M) or "Internet of Things" world of connected devices, and for mobile applications where bandwidth and battery power are at a premium." – source http://mqtt.org/faq

Before you start.

You are required to read the online documents [1], [2], [3] and install and test the software as guided.

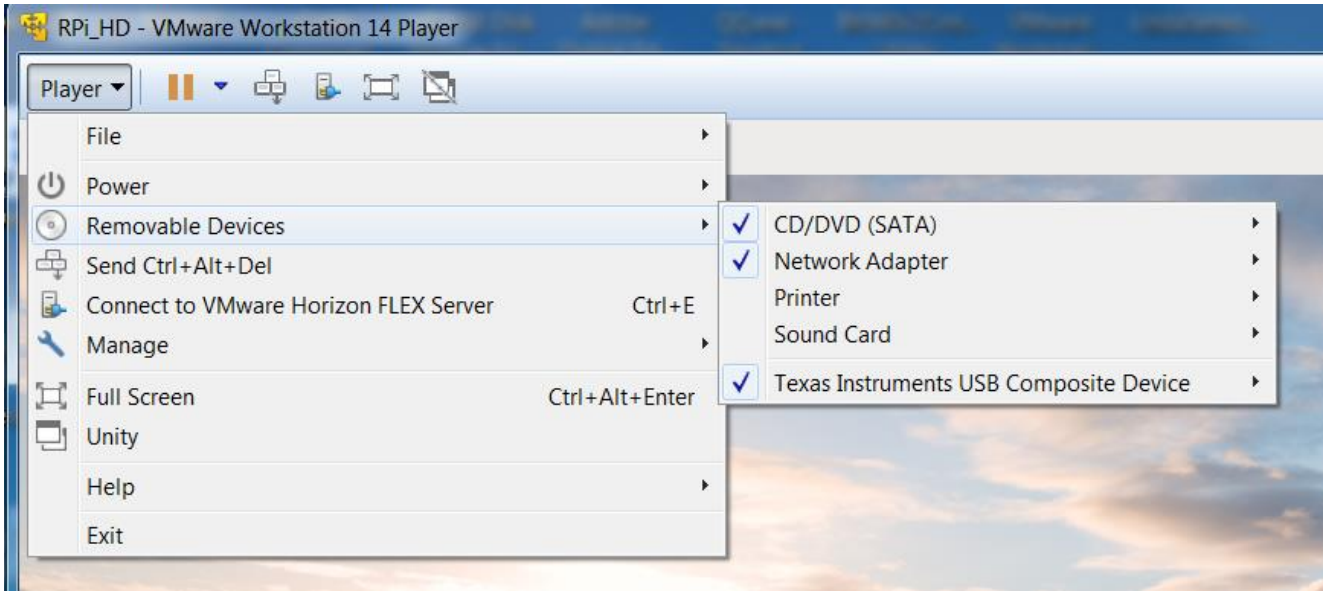
Setup and running of 6LoWPAN network.

Following previous tutorials you are required to build and flash the firmware into CC2650 Launchpad (SLIP radio) and one or more CC2650 Sensortag (cc26xx-web-demo). Use the Channel number as assigned to your group and the same on all of your motes. Also note, that the Launchpad firmware has to be done only once, while you will change and customize the Contiki firmware on Sensortags to suit your needs. Connect your SLIP radio into the USB port of Raspberry Pi (RPi), power on the system and connect either locally or through the Remote Desktop as described previously. If your RPi is setup correctly you should see the IP address similar to the one in a picture below.
If you use RPi Virtual Machine make sure that your Launchpad is connected to your guest OS (RPi VM)

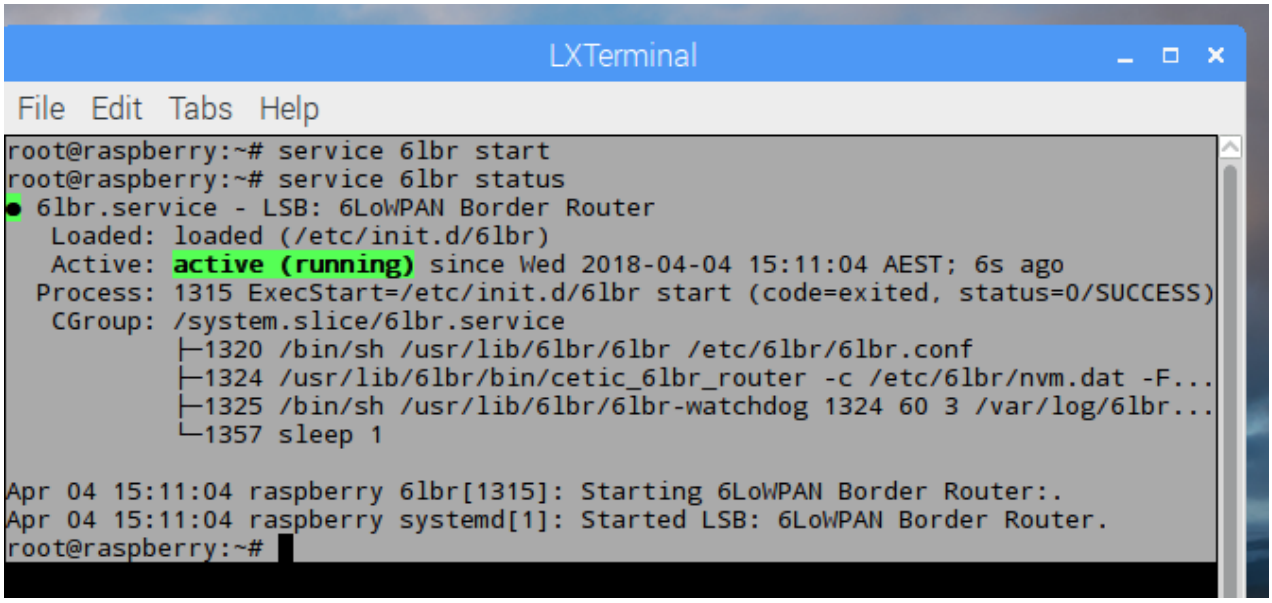
One time setup of 6LoWPAN network.

Following part is to be performed only once as the setup to the motes is stored in external memory and stays persistent during power cycling and flash/program memory changes. Additionally all other operations are now included in Node-Red flows and do not require any manual action.

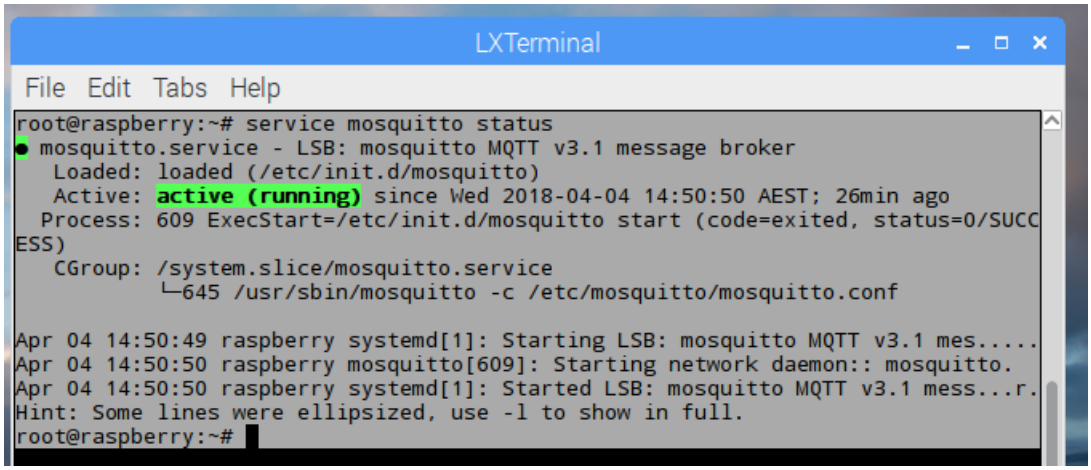
```
root@raspberrypi:~# ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:01:c9:7e
          inet6 addr: fe80::8943:5157:2336:12a7/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:144 errors:0 dropped:0 overruns:0 frame:0
          TX packets:144 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9648 (9.4 KiB)  TX bytes:9648 (9.4 KiB)

wlan0     Link encap:Ethernet  HWaddr b8:27:eb:54:9c:2b
          inet addr:169.254.162.190  Bcast:169.254.255.255  Mask:255.255.0.0
          inet6 addr: fe80::880d:43d2:b0ee:b09f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17 errors:0 dropped:0 overruns:0 frame:0
          TX packets:48 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4582 (4.4 KiB)  TX bytes:8711 (8.5 KiB)

root@raspberrypi:~#
```



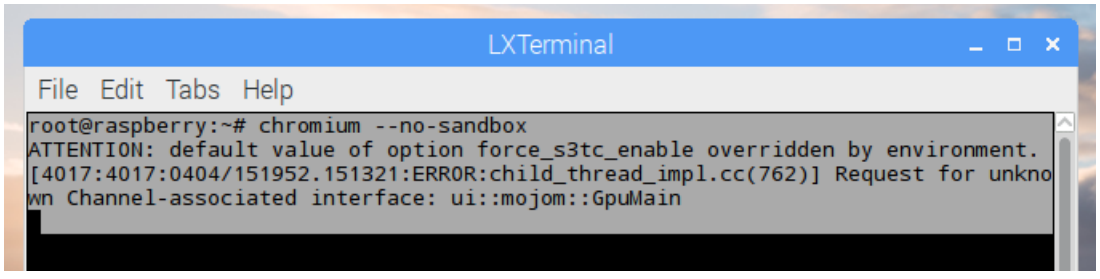
Start 6lbr service and check it



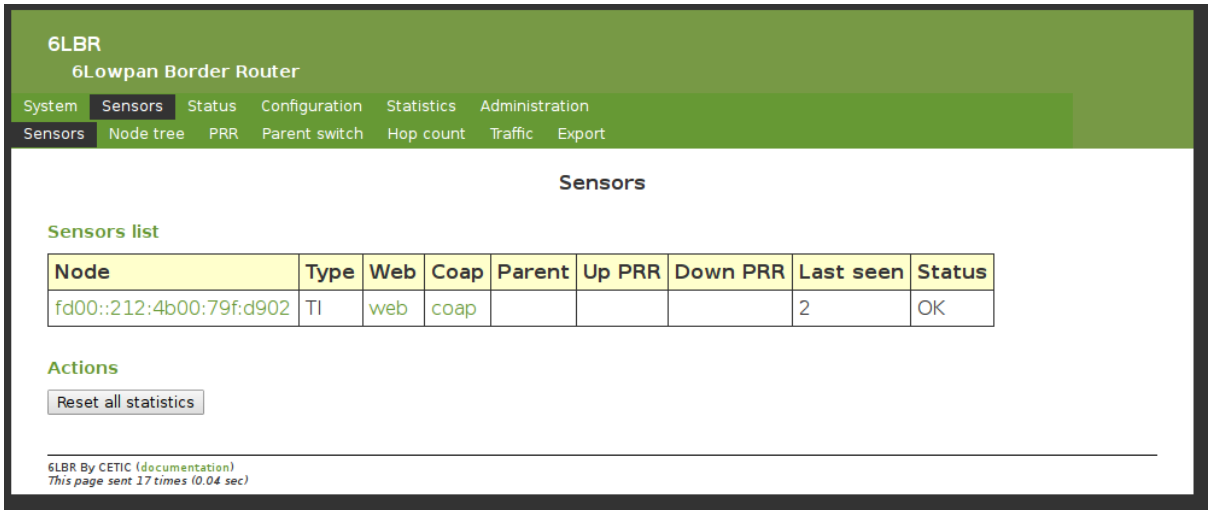
Check if your Mosquitto broker is running



Open the web browser



Open the web link



Open the MQTT/IBM Cloud Config and change the Broker IP to bbbb::101 and Interval (secs) to required values and then click submit button

We need to create some routes, the easiest way to do it is to start and stop the nodejs application:

```
root@raspberry:/www/CoAPwebserver_28_11_2016# cd /www
root@raspberry:/www# ls
CoAPwebserver_28_11_2016  webserver01
root@raspberry:/www# cd CoAPwebserver_28_11_2016
root@raspberry:/www/CoAPwebserver_28_11_2016# node app.js
PhDwebserver version 15/07/2016
This platform is linux
savedOS is: linux
serverOS is: linux
coap0 server created @: 15:54:58
```

```
Express server listening on:  192.168.1.7 : 3000
user_count: 1
socket is connected @: 15:56:49
coap0 server reopen @: 15:56:49
coap1 server reopen @: 15:56:49
coap2 server reopen @: 15:56:49
coap3 server reopen @: 15:56:49
coap4 server reopen @: 15:56:49
checking the 6LoWPAN border router...
software for new 6lbr version
Initial Neighbour Discovery
ND 15:56:50
6LoWPAN Border Router is running:
table length is 18
motes_num = 1
[ 'fd00::212:4b00:79f:d902' ]
no service discovery here, wait for websockets
^C
root@raspberry:/www/CoAPwebserver_28_11_2016#
```

Press CTRL-C to terminate

Check ifconfig

```
tap0      Link encap:Ethernet  HWaddr 02:0a:0b:0c:0d:0e
          inet addr:169.254.77.32  Bcast:169.254.255.255  Mask:255.255.0.0
          inet6 addr: fe80::b0fa:866e:73cd:21ca/64 Scope:Link
          inet6 addr: bbbb::bfc2:cd88:d32b:3fb9/64 Scope:Global
          inet6 addr: fe80::a:bff:fe0c:d0e/64 Scope:Link
          inet6 addr: bbbb::101/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4056 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4330 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:544591 (531.8 KiB)  TX bytes:363380 (354.8 KiB)

root@raspberry:/www/CoAPwebserver_28_11_2016#
```

The address bbbb::101 is the IPv6 address of our Mosquitto broker and is programmed into the motes

The end of one time setup of 6LoWPAN network.

Node-Red flow.

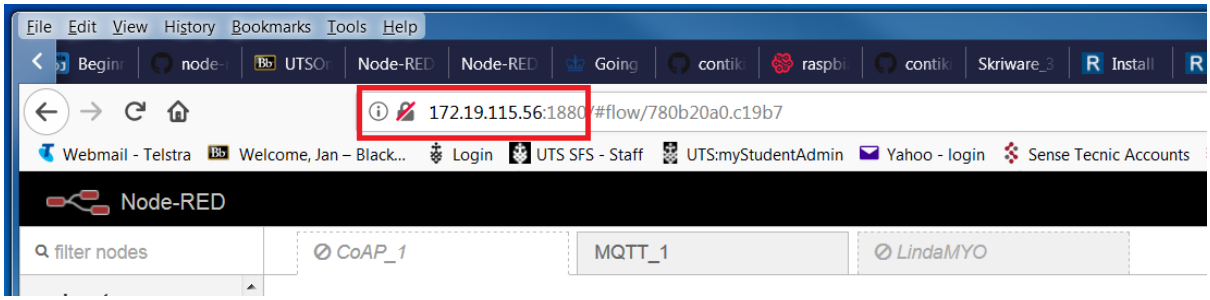
Now we will create a Node-Red flow.

It is assume that you are already familiar with Node-Red, if not please refer to previous tutorials and links given. You are given an example flow for your study, analysis and reference.

Run Node-Red on RPi with a command node-red-start

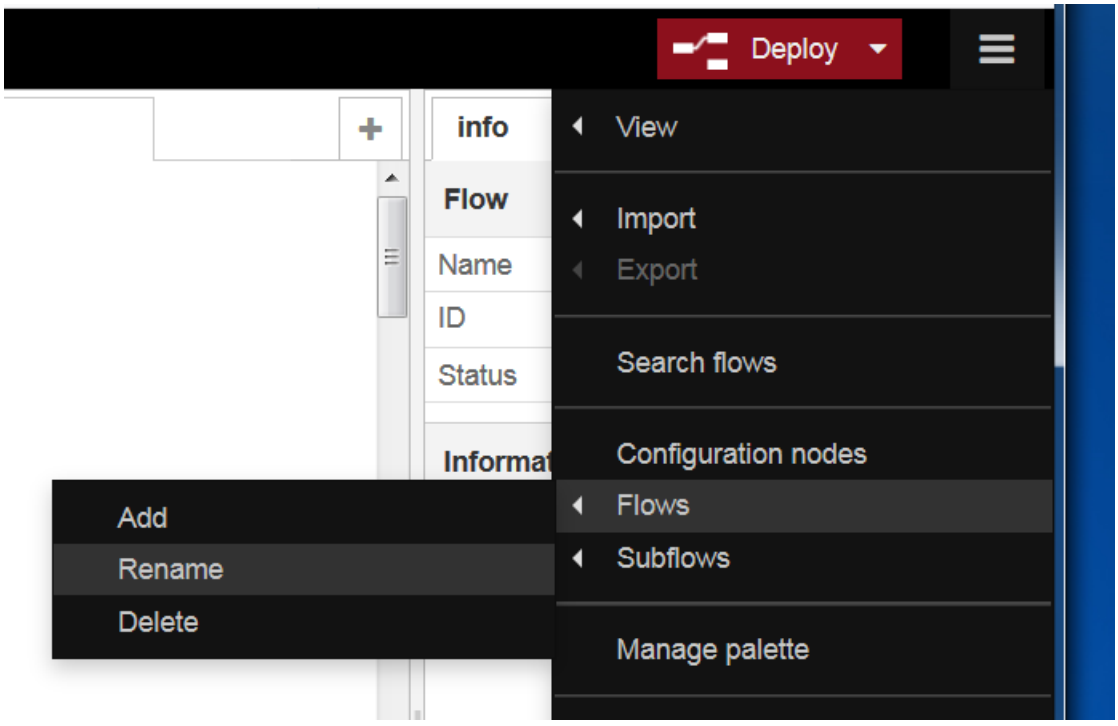
```
root@raspberrypi:~# node-red-start

Start Node-RED

Once Node-RED has started, point a browser at http://172.19.115.56:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use   sudo systemctl enable nodered.service  to autostart Node-RED at every boot
Use   sudo systemctl disable nodered.service to disable autostart on boot

To find more nodes and example flows - go to http://flows.nodered.org
5 Apr 10:33:47 - [info]

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

5 Apr 10:33:47 - [info] Node-RED version: v0.17.5
5 Apr 10:33:47 - [info] Node.js  version: v8.11.1
5 Apr 10:33:47 - [info] Linux 4.14.31-v7+ arm LE
5 Apr 10:33:48 - [info] Loading palette nodes
5 Apr 10:33:52 - [info] Dashboard version 2.6.2 started at /ui
5 Apr 10:33:54 - [info] Settings file  : /root/.node-red/settings.js
5 Apr 10:33:54 - [info] User directory : /root/.node-red
5 Apr 10:33:54 - [info] Flows file     : /root/.node-red/flows_raspberrypi.json
5 Apr 10:33:55 - [info] Server now running at http://127.0.0.1:1880/
5 Apr 10:33:55 - [info] Starting flows
5 Apr 10:33:55 - [info] Started flows
5 Apr 10:33:55 - [info] [mqtt-broker:55ed84f.974347c] Connected to broker: mqtt://localhost:1883
```

On your desktop PC open the browser and type the address given by Node-Red in this example it is 192.168.1.7:1880

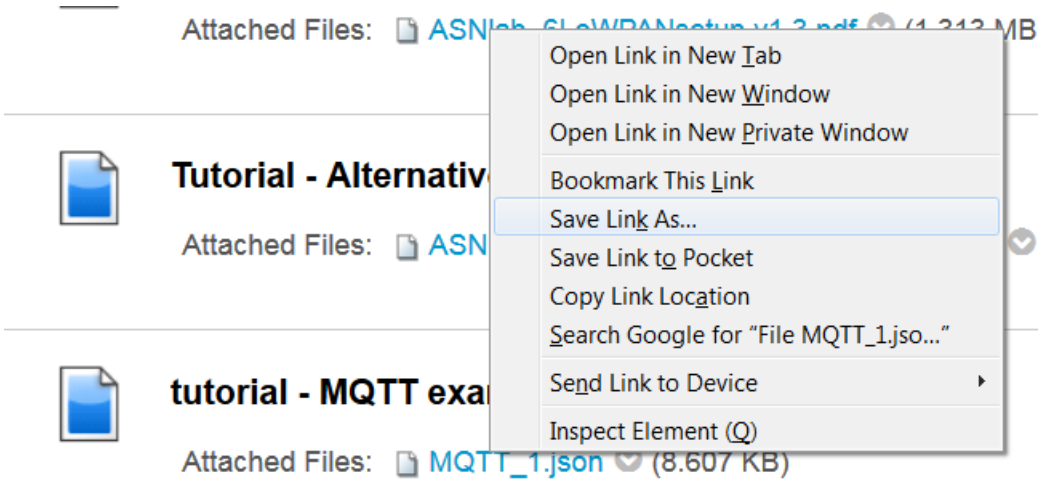


Create a new tab by clicking the + button

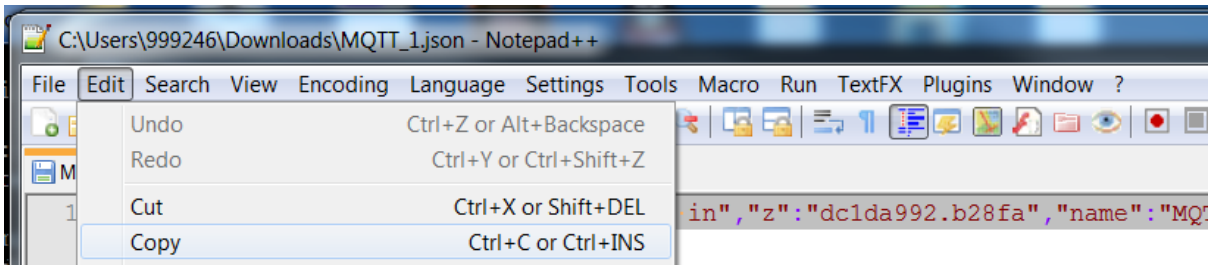
From the hamburger menu select Flows > Rename and give it a name MQTT_1



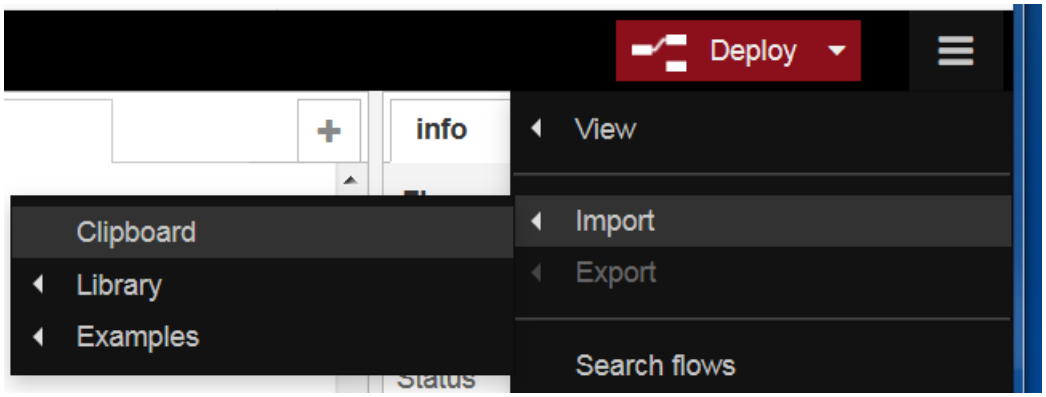
Download the file MQTT1.json from UTSonline and save it on your local PC



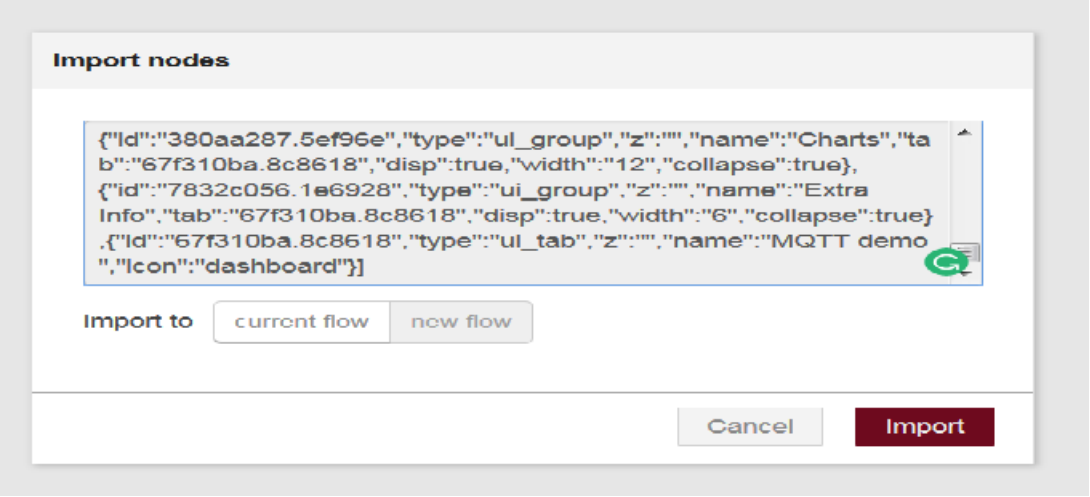
Open it from your favourite editor and copy into clipboard



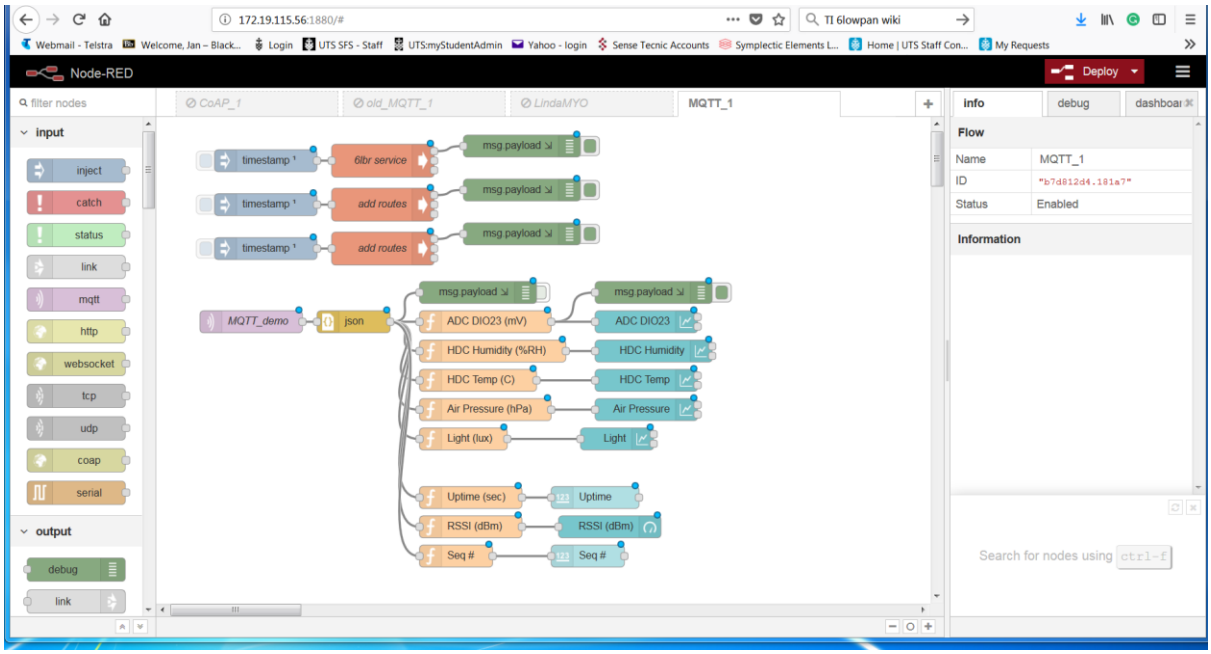
From your browser Import from Clipboard



Paste and Import

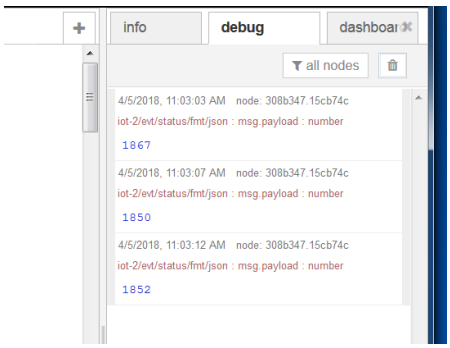


You should see the following picture

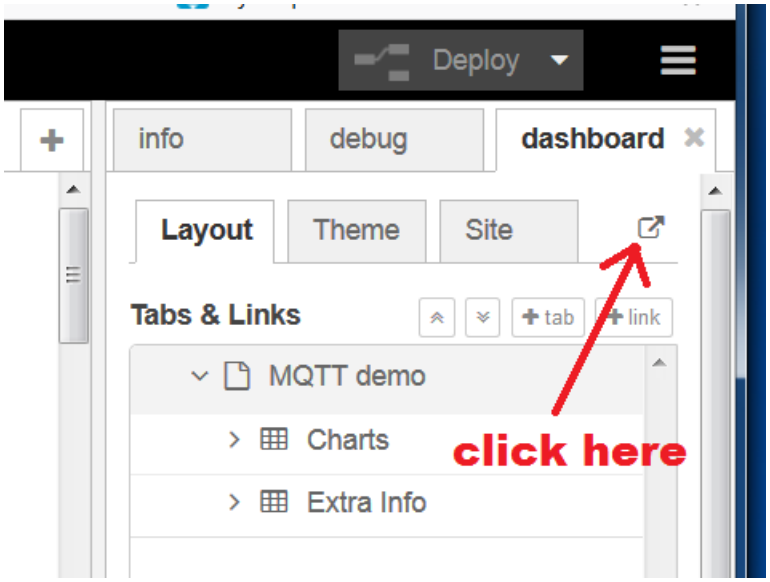


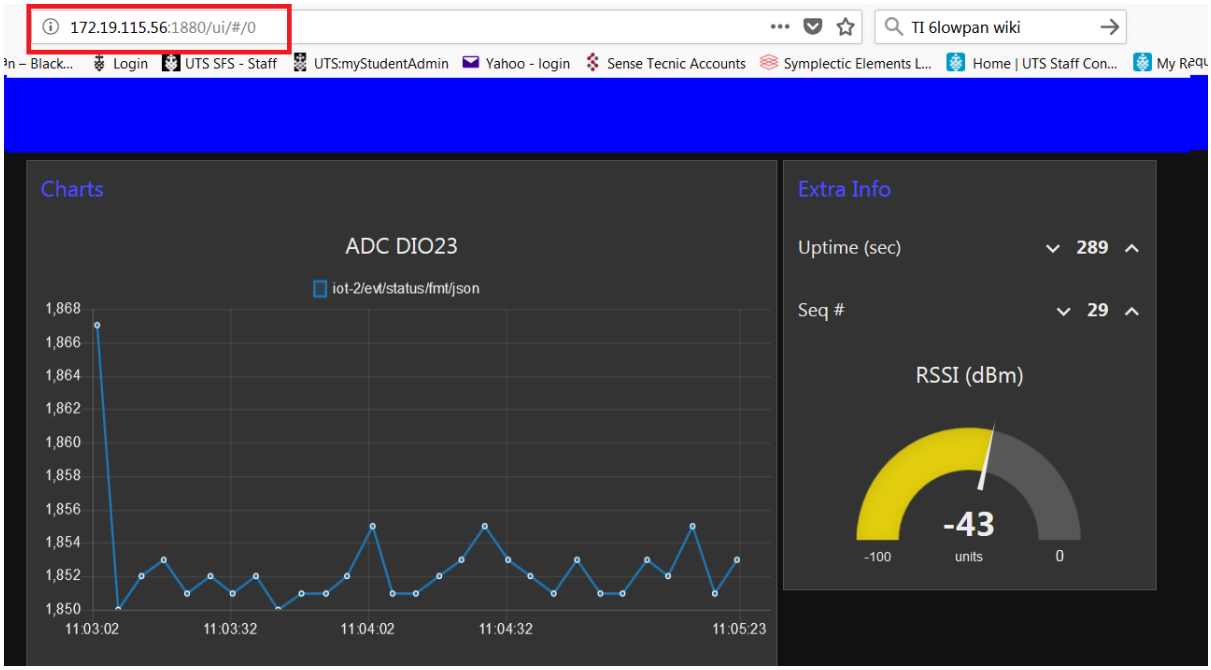
Click on the Deploy button

The example debug message is shown, you can disable it

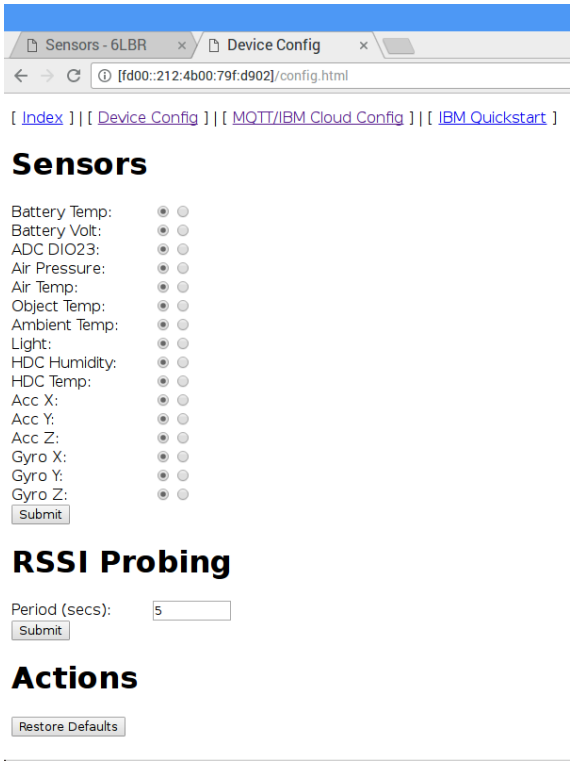


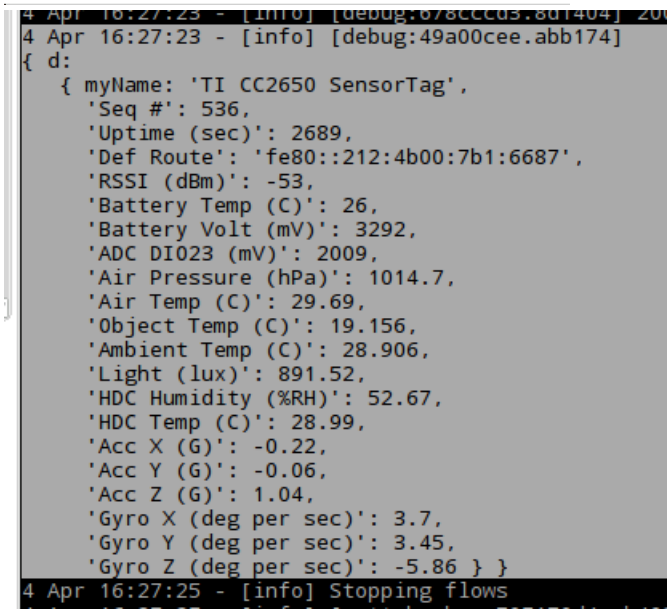
Switch into the UI tab on your browser





Note, that you can change the Device Config for a particular mote

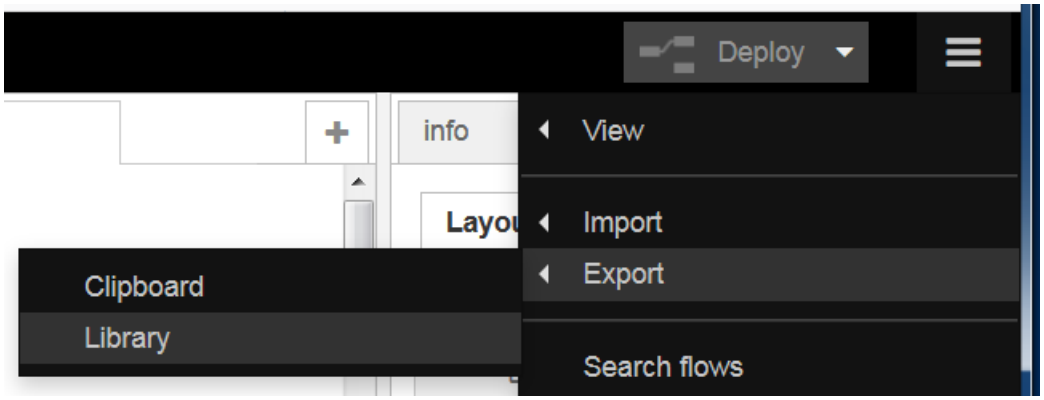


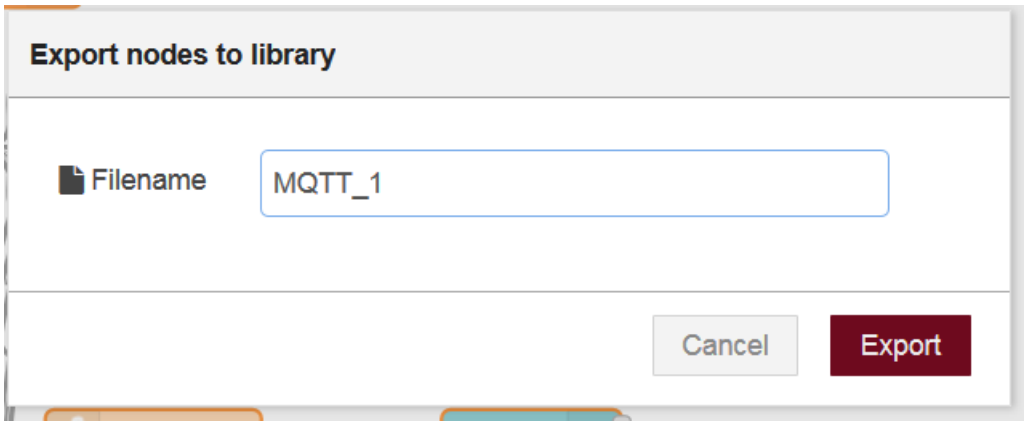


```
4 Apr 16:27:23 - [info] [debug:49a00cee.abb174]
{ d:
   { myName: 'TI CC2650 SensorTag',
     'Seq #': 536,
     'Uptime (sec)': 2689,
     'Def Route': 'fe80::212:4b00:7b1:6687',
     'RSSI (dBm)': -53,
     'Battery Temp (C)': 26,
     'Battery Volt (mV)': 3292,
     'ADC DIO23 (mV)': 2009,
     'Air Pressure (hPa)': 1014.7,
     'Air Temp (C)': 29.69,
     'Object Temp (C)': 19.156,
     'Ambient Temp (C)': 28.906,
     'Light (lux)': 891.52,
     'HDC Humidity (%RH)': 52.67,
     'HDC Temp (C)': 28.99,
     'Acc X (G)': -0.22,
     'Acc Y (G)': -0.06,
     'Acc Z (G)': 1.04,
     'Gyro X (deg per sec)': 3.7,
     'Gyro Y (deg per sec)': 3.45,
     'Gyro Z (deg per sec)': -5.86 } }
4 Apr 16:27:25 - [info] Stopping flows
```

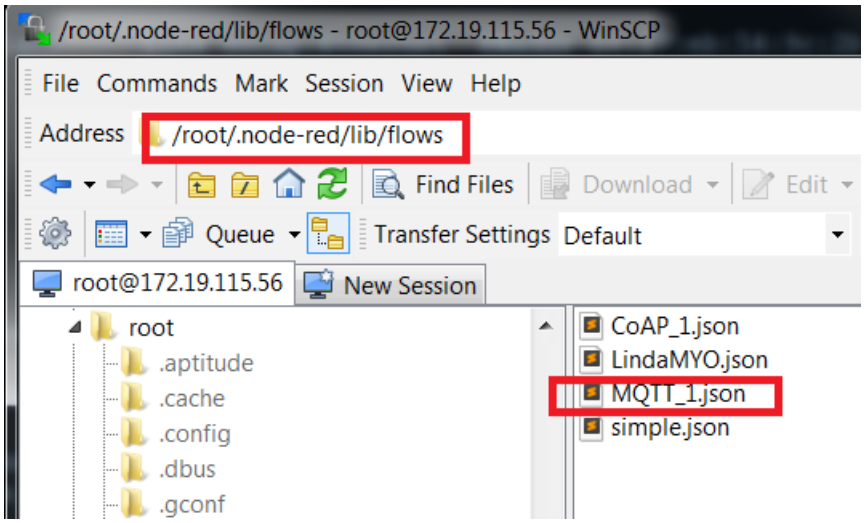
Currently all sensors readings are send:

Please note, that you can save the flow into library by exporting it





It is saved then into RPi file system



Congratulations, you have created successfully the MQTT example flow

This is the end of the MQTT example tutorial

References:

1. What is MQTT and How it Works https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/
2. How to Install Mosquitto Broker on Raspberry PI https://randomnerdtutorials.com/how-to-install-mosquitto-broker-on-raspberry-pi/
3. Testing Mosquitto Broker and Client on Raspberry Pi https://randomnerdtutorials.com/testing-mosquitto-broker-and-client-on-raspbbery-pi/
4. TI 6LoWPAN wiki http://processors.wiki.ti.com/index.php/Contiki-6LOWPAN