Connect Arrow SmartEverything Fox board to the Watson IoT Platform

Requirements

- Arrow SmartEverything FOX device registered and connected to the Sigfox backend.
- A Bluemix application based on the "Internet of Things Platform Starter" boilerplate.
- Access to the Sigfox backend to setup the integration towards the Watson IoT Platform application in Bluemix.

Steps

1. Introduction

This guide is building on the content from another guide:
Connecting Sigfox backend to Watson IoT Platform – with the following changes and additions:

- We will add a new Device type and callback script in the Sigfox backend to reflect another device type in step 5
- We will make changes to the Watson IoT Platform gateway in step 6 to change the device type.
- We will make changes to the Watson IoT Platform application in step 7 to extract the SmartEverything data and visualize them in a node-red dashboard.

You'll need to run through the other guide before continuing with the next steps in this guide, hence have access to the Sigfox backend and your Bluemix application with the Watson IoT Platform service working properly.

2. SmartEverything: Prepare the device to be a weather station

Download the User Guide for the Arrow SmartEverything here:
http://docs-europe.electrocomponents.com/webdocs/144b/0900766b8144b09d.pdf

Follow the steps in Chapter 5 of this User Guide in order to setup your development environment for the Arrow SmartEverything device.

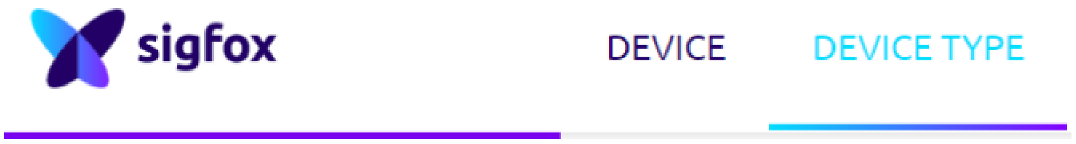
Then go to https://github.com/nicolsc/sigfox-weather-station to get the code and instructions for the Weather Station.

You should now be up and running with the weather station on your Arrow SmartEverything utilizing the Sigfox network to submit the following data in the 12 bytes available every 10 minutes:

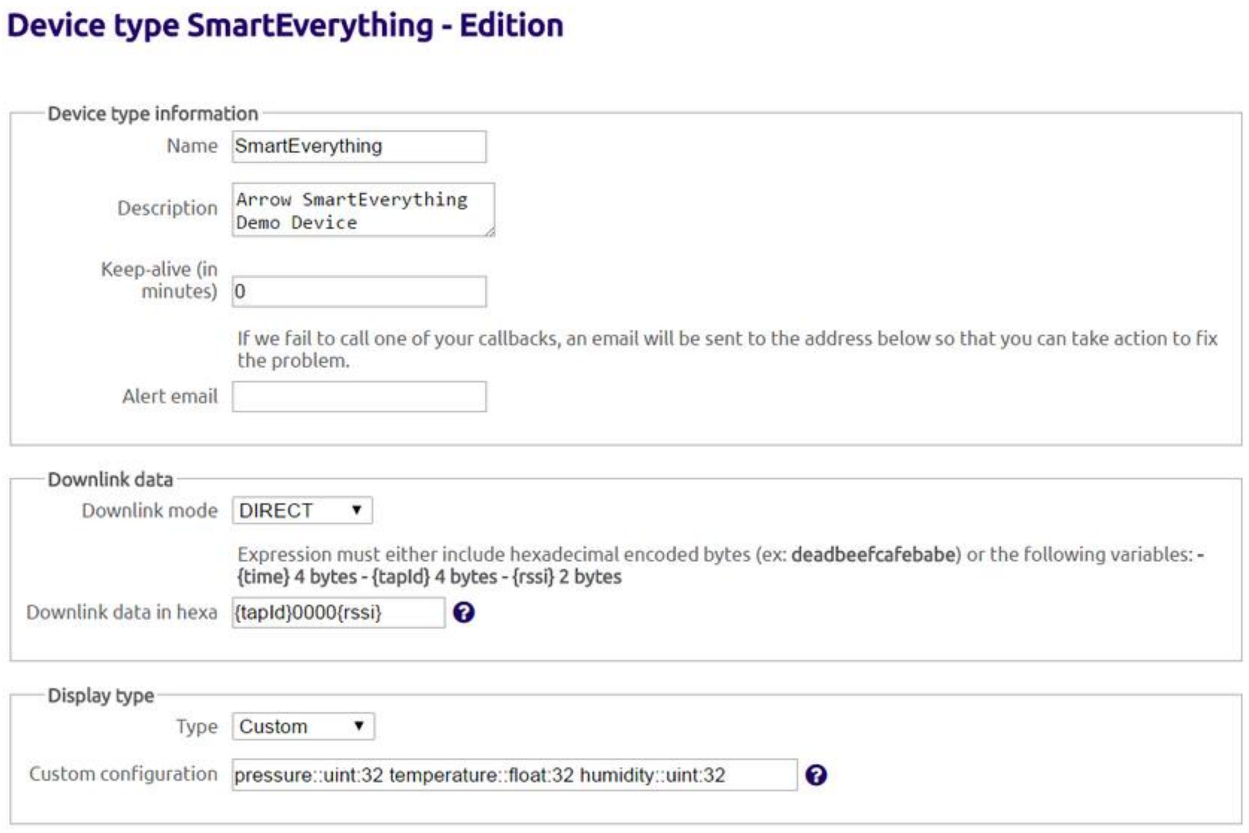
- Pressure in mbar
- Temperature in °C
- Humidity in %

3. Sigfox: Create new Device type and callback script

3.1. Go to the Sigfox Backend, where you login with your credentials and go to the Device Type tab:



3.2. Click on New in the upper right corner in case you haven't already created a device type for your SmartEverything, and fill in the details for this new device type, e.g. like this:



Device type SmartEverything - Edition

Device type information

Name	SmartEverything
Description	Arrow SmartEverything Demo Device
Keep-alive (in minutes)	0
	If we fail to call one of your callbacks, an email will be sent to the address below so that you can take action to fix the problem.
Alert email	

Downlink data

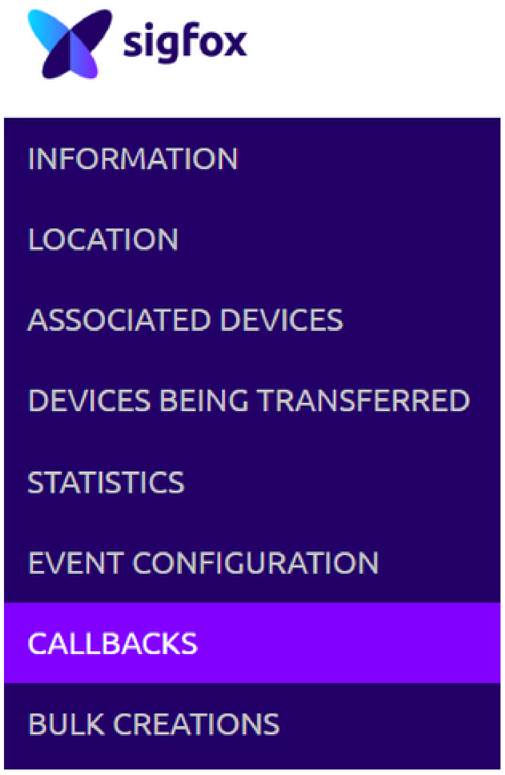
Downlink mode	DIRECT
	Expression must either include hexadecimal encoded bytes (ex: **deadbeefcafebabe**) or the following variables: - {time} 4 bytes - {tapId} 4 bytes - {rssi} 2 bytes
Downlink data in hexa	{tapId}0000{rssi}

Display type

Type	Custom
Custom configuration	pressure::uint:32 temperature::float:32 humidity::uint:32

Click on OK.

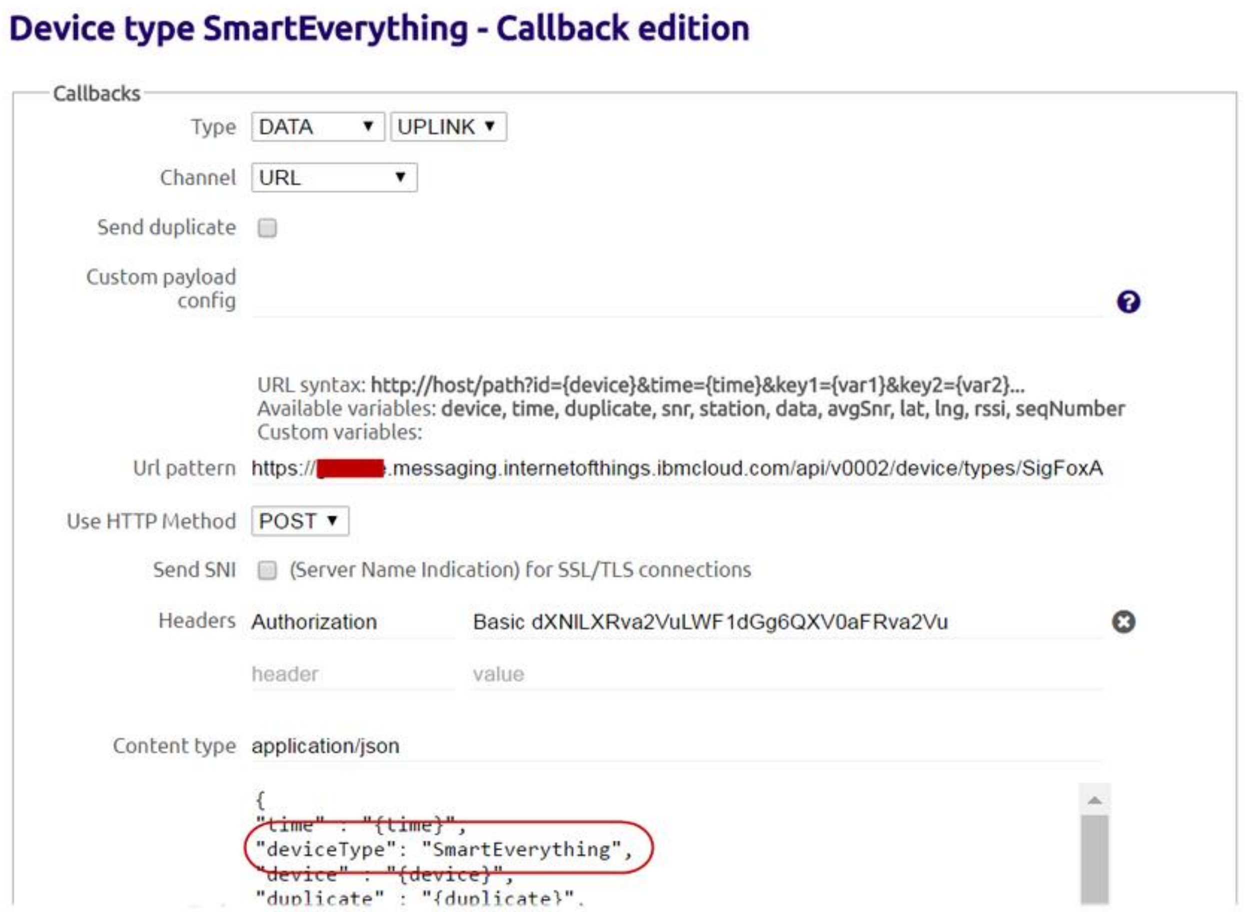
3.3. Go to the Callbacks section:



3.4. Follow step 5.4-5.7 in the other guide:
[Connecting Sigfox backend to Watson IoT Platform – by building an IoT Gateway in Bluemix.]()
Then add the content to the Body in order to send over the parameters available
+ the device type, so you can use that in the Gateway:

```
{
"time" : "{time}",
"deviceType": "SmartEverything",
"device" : "{device}",
"duplicate" : "{duplicate}",
"snr" : "{snr}",
"rssi" : "{rssi}",
"avgSnr" : "{avgSnr}",
"station" : "{station}",
"lat" : "{lat}",
"lng" : "{lng}",
"seqNumber" : "{seqNumber}",
"data" : "{data}"
}
```

– and we are done creating the callback:



Device type SmartEverything - Callback edition

Callbacks

Type	DATA ▾ UPLINK ▾
Channel	URL ▾
Send duplicate	☐
Custom payload config	
	URL syntax: http://host/path?id={device}&time={time}&key1={var1}&key2={var2}... Available variables: device, time, duplicate, snr, station, data, avgSnr, lat, lng, rssi, seqNumber Custom variables:
Url pattern	https://[redacted].messaging.internetofthings.ibmcloud.com/api/v0002/device/types/SigFoxA
Use HTTP Method	POST ▾
Send SNI	☐ (Server Name Indication) for SSL/TLS connections
Headers	Authorization — Basic dXNlLXRva2VuLWF1dGg6QXV0aFRva2Vu
	header — value
Content type	application/json

```
{
"time" : "{time}",
"deviceType": "SmartEverything",
"device" : "{device}",
"duplicate" : "{duplicate}",
```

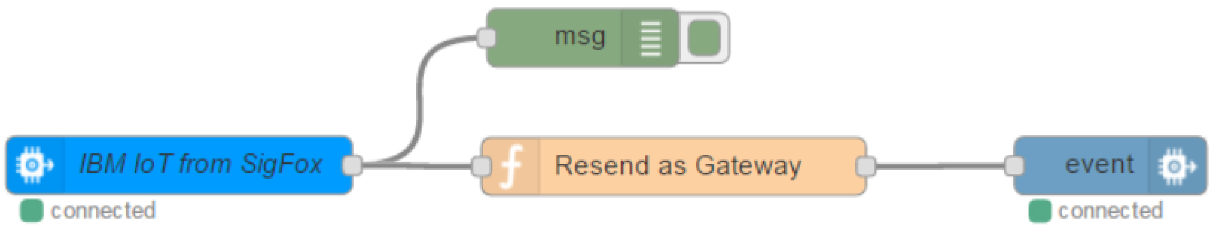
3.5. Click on OK

3.6. Add your device as this Device type using the Device ID and PAC provided with the SmartEverything device.

The device is now properly configured in the Sigfox backend to send data over to the Watson IoT Platform.

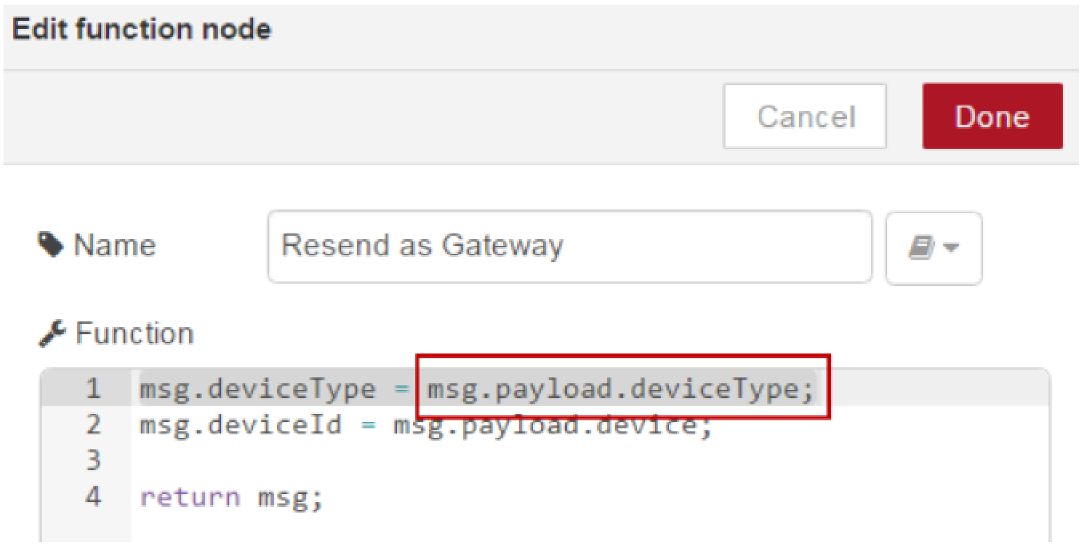
4. WIoTP: Edit the IoT Gateway in Node-Red to extract device type from message

At the end of step 6 in the other guide, the flow for the Gateway looked like this:



We will now edit the Resend as Gateway function node to reflect the SmartEverything device type.

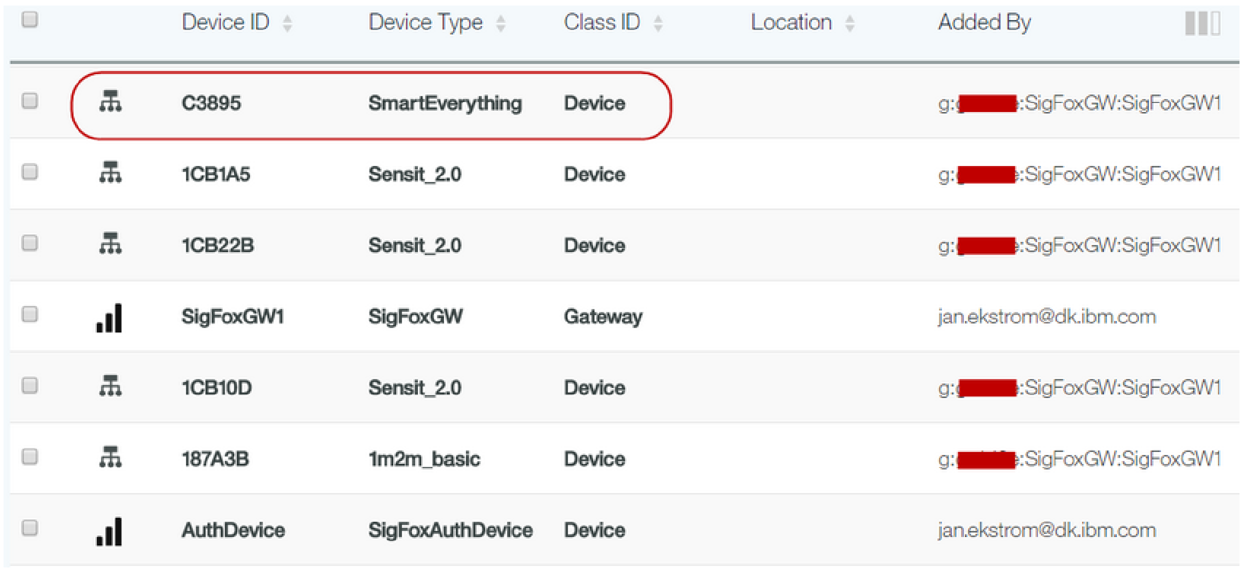
4.1. Double click the function node and change the line that define the deviceType, so it looks like this:



Name: Resend as Gateway

```
msg.deviceType = msg.payload.deviceType;
msg.deviceId = msg.payload.device;

return msg;
```

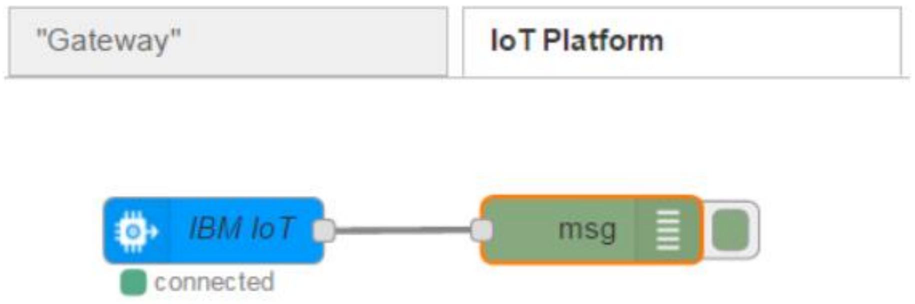
4.2. Deploy the application. A new deviceType will be created at the Watson IoT Platform called SmartEverything and new data flowing in from that specific deviceType in the Sigfox backend will be accessible.
Go to the Device tab at your Watson IoT Platform of your application and you will see that your device have been auto registered with the new deviceType (once it have send some data):



Device ID	Device Type	Class ID	Location	Added By
C3895	SmartEverything	Device		g:[redacted]:SigFoxGW:SigFoxGW1
1CB1A5	Sensit_2.0	Device		g:[redacted]:SigFoxGW:SigFoxGW1
1CB22B	Sensit_2.0	Device		g:[redacted]:SigFoxGW:SigFoxGW1
SigFoxGW1	SigFoxGW	Gateway		jan.ekstrom@dk.ibm.com
1CB10D	Sensit_2.0	Device		g:[redacted]:SigFoxGW:SigFoxGW1
187A3B	1m2m_basic	Device		g:[redacted]:SigFoxGW:SigFoxGW1
AuthDevice	SigFoxAuthDevice	Device		jan.ekstrom@dk.ibm.com

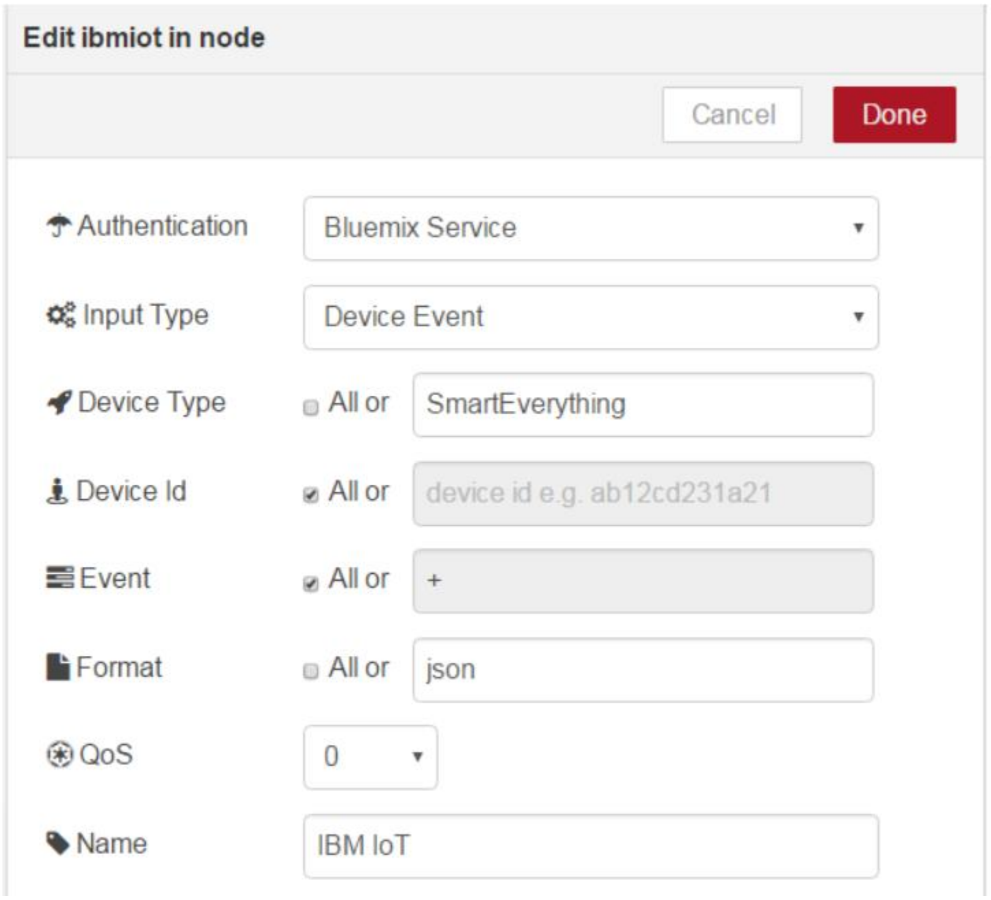
5. WIoTP: Edit the "IoT Platform" Node-Red flow for receiving the SmartEverything data

At the end of step 7 in the other guide, the flow for the IoT Platform looked like this:



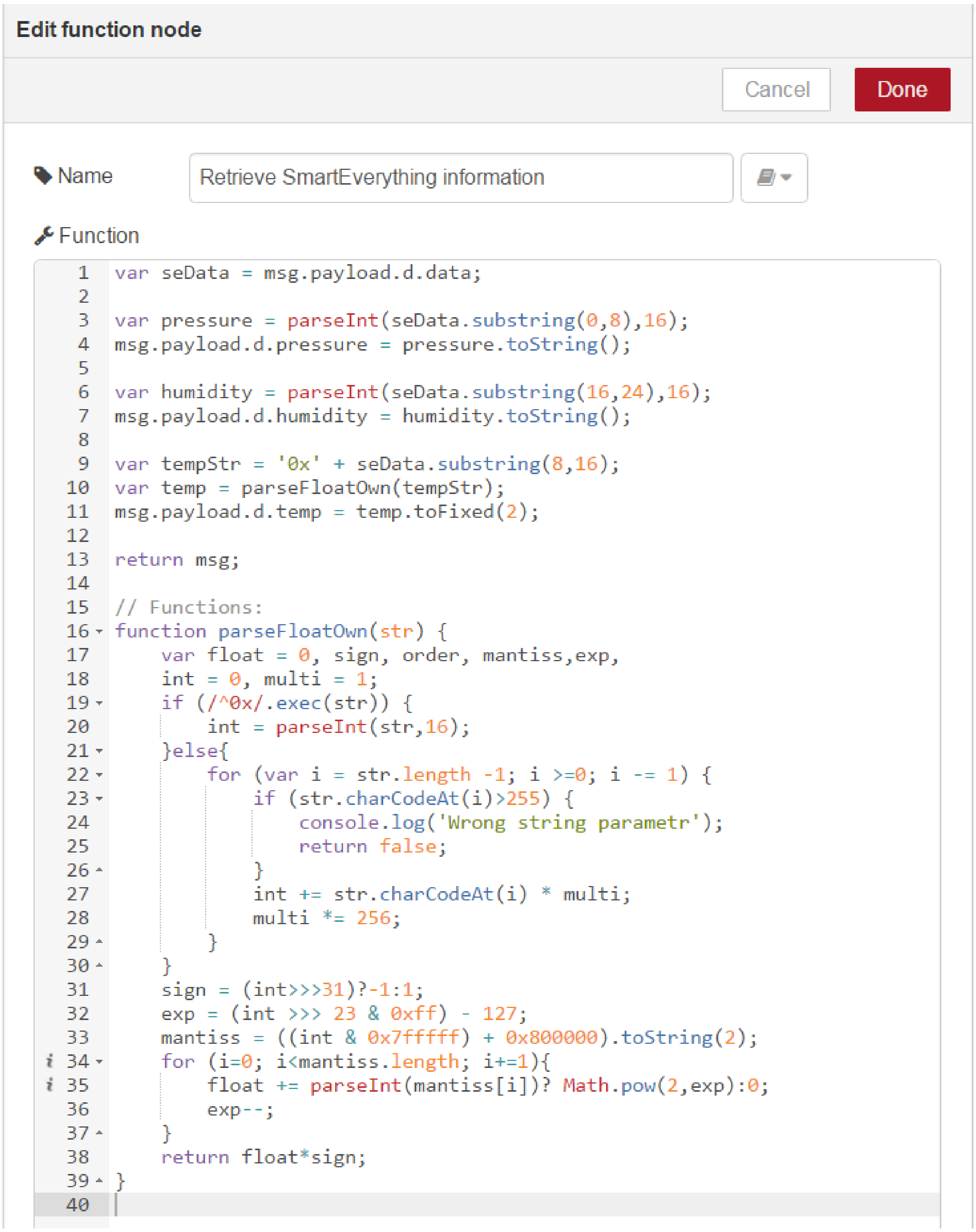
We will now change this flow to receive MQTT messages for the devices of the SmartEverything device type and extract the data for temperature, humidity and barometric pressure.

5.1. Double click the IBM IoT node and change Device Type to SmartEverything and click Done:



Authentication		Bluemix Service
Input Type		Device Event
Device Type	☐ All or	SmartEverything
Device Id	✓ All or	device id e.g. ab12cd231a21
Event	✓ All or	+
Format	☐ All or	json
QoS		0
Name		IBM IoT

5.2. Add a function node and add the following code to decode the SmartEverything data (including temperature, humidity and barometric pressure):



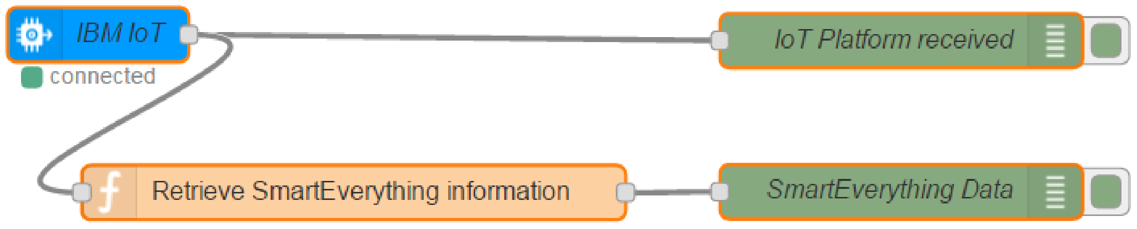
Retrieve SmartEverything information

```
var seData = msg.payload.d.data;

var pressure = parseInt(seData.substring(0,8),16);
msg.payload.d.pressure = pressure.toString();

var humidity = parseInt(seData.substring(16,24),16);
msg.payload.d.humidity = humidity.toString();

var tempStr = '0x' + seData.substring(8,16);
var temp = parseFloatOwn(tempStr);
msg.payload.d.temp = temp.toFixed(2);

return msg;

// Functions:
function parseFloatOwn(str) {
    var float = 0, sign, order, mantiss,exp,
    int = 0, multi = 1;
    if (/^0x/.exec(str)) {
        int = parseInt(str,16);
    }else{
        for (var i = str.length -1; i >=0; i -= 1) {
            if (str.charCodeAt(i)>255) {
                console.log('Wrong string parametr');
                return false;
            }
            int += str.charCodeAt(i) * multi;
            multi *= 256;
        }
    }
    sign = (int>>>31)?-1:1;
    exp = (int >>> 23 & 0xff) - 127;
    mantiss = ((int & 0x7fffff) + 0x800000).toString(2);
    for (i=0; i<mantiss.length; i+=1){
        float += parseInt(mantiss[i])? Math.pow(2,exp):0;
        exp--;
    }
    return float*sign;
}

```

Below is the code in textual form, to make it easier for you to create your “Retrieve SmartEverything information” function node:

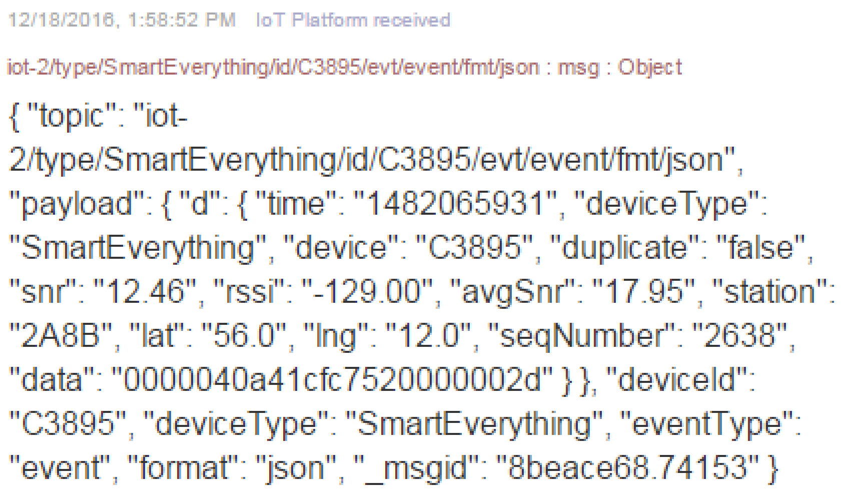
```
var seData = msg.payload.d.data;

var pressure = parseInt(seData.substring(0,8),16);
msg.payload.d.pressure = pressure.toString();

var humidity = parseInt(seData.substring(16,24),16);
msg.payload.d.humidity = humidity.toString();

var tempStr = '0x' + seData.substring(8,16);
var temp = parseFloatOwn(tempStr);
msg.payload.d.temp = temp.toFixed(2);

return msg;

// Functions:
function parseFloatOwn(str) {
  var float = 0, sign, order, mantiss,exp,
  int = 0, multi = 1;
  if (/^0x/.exec(str)) {
    int = parseInt(str,16);
  }else{
    for (var i = str.length -1; i >=0; i -= 1) {
      if (str.charCodeAt(i)>255) {
        console.log('Wrong string parametr');
        return false;
      }
      int += str.charCodeAt(i) * multi;
      multi *= 256;
    }
  }
  sign = (int>>>31)?-1:1;
  exp = (int >>> 23 & 0xff) - 127;
  mantiss = ((int & 0x7fffff) + 0x800000).toString(2);
  for (i=0; i<mantiss.length; i+=1){
    float += parseInt(mantiss[i])? Math.pow(2,exp):0;
    exp--;
  }
  return float*sign;
}
```

5.3. Drag in another debug node and change it to Output: complete msg object. Connect the additional nodes to the existing flow and optionally rename the two debug nodes:



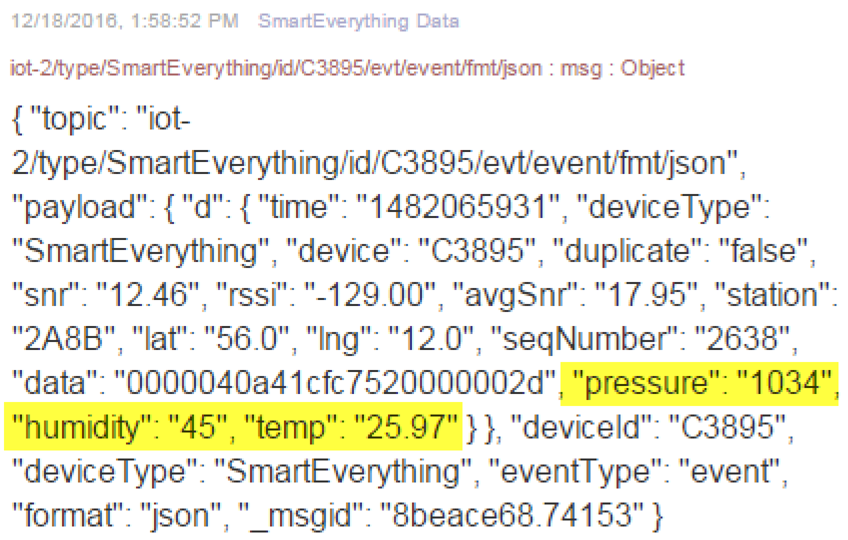
5.4. Deploy the application and wait until a new message is received and then compare the two debug messages.

The “IoT Platform received” debug node output shows the raw data (“0000040a41cfc7520000002d”) and the new deviceType (“SmartEverything”):



```
12/18/2016, 1:58:52 PM   IoT Platform received
iot-2/type/SmartEverything/id/C3895/evt/event/fmt/json : msg : Object
{ "topic": "iot-2/type/SmartEverything/id/C3895/evt/event/fmt/json", "payload": { "d": { "time": "1482065931", "deviceType": "SmartEverything", "device": "C3895", "duplicate": "false", "snr": "12.46", "rssi": "-129.00", "avgSnr": "17.95", "station": "2A8B", "lat": "56.0", "lng": "12.0", "seqNumber": "2638", "data": "0000040a41cfc7520000002d" } }, "deviceId": "C3895", "deviceType": "SmartEverything", "eventType": "event", "format": "json", "_msgid": "8beace68.74153" }
```

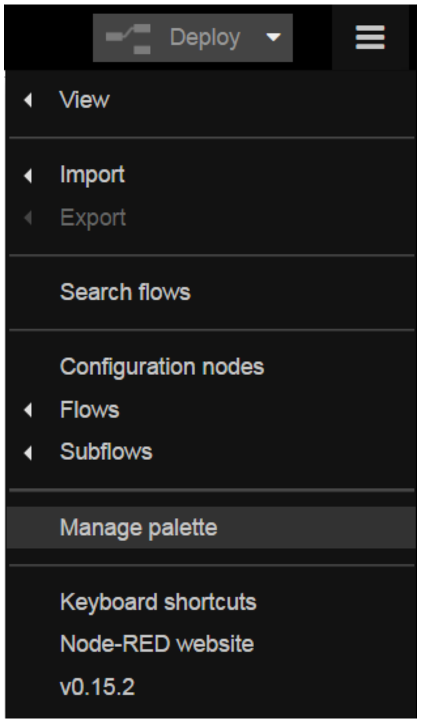
The “SmartEverything Data” debug node output shows the added extracted data added by the function node:


```
12/18/2016, 1:58:52 PM   SmartEverything Data
iot-2/type/SmartEverything/id/C3895/evt/event/fmt/json : msg : Object
{ "topic": "iot-
2/type/SmartEverything/id/C3895/evt/event/fmt/json",
"payload": { "d": { "time": "1482065931", "deviceType":
"SmartEverything", "device": "C3895", "duplicate": "false",
"snr": "12.46", "rssi": "-129.00", "avgSnr": "17.95", "station":
"2A8B", "lat": "56.0", "lng": "12.0", "seqNumber": "2638",
"data": "0000040a41cfc7520000002d", "pressure": "1034",
"humidity": "45", "temp": "25.97" } }, "deviceId": "C3895",
"deviceType": "SmartEverything", "eventType": "event",
"format": "json", "_msgid": "8beace68.74153" }
```

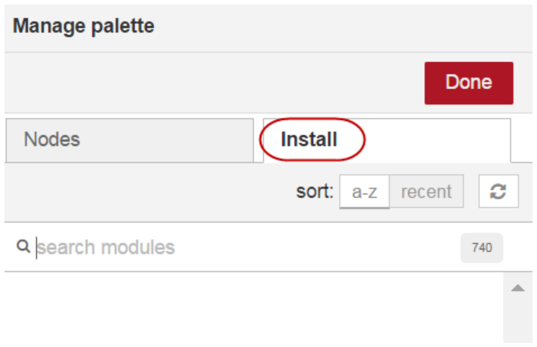
6. WIoTP: Edit the "IoT Platform" Node-Red flow to visualize the SmartEverything data

We'll use the node-red-dashboard capabilities to visualize the extracted data from the SmartEverything device on a dashboard. First we have to add that library to your application:

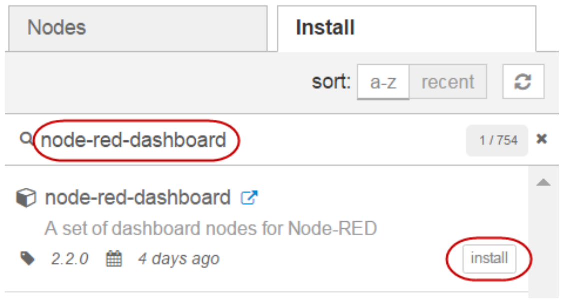
6.1. Go to the Node-red editor of your application, e.g. http://sigfoxgw.eu-gb.mybluemix.net/red. Select Manage palette in the right menu:



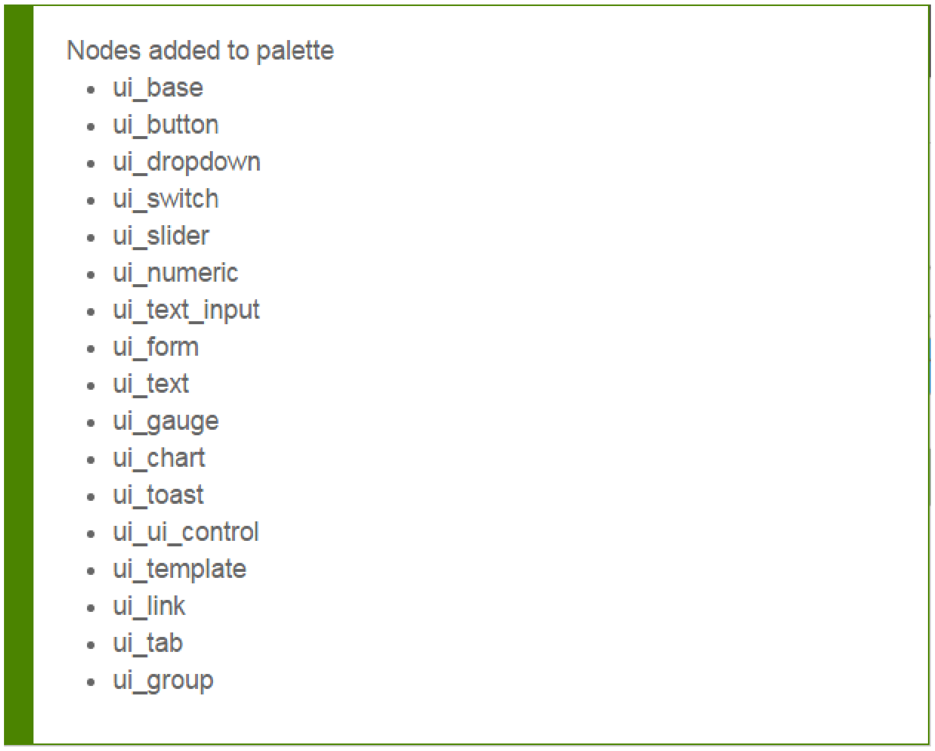
6.2. Press the Install tab:



6.3. Write “node-red-dashboard” in the search field and then press “install” once the node-red-dashboard library is found:



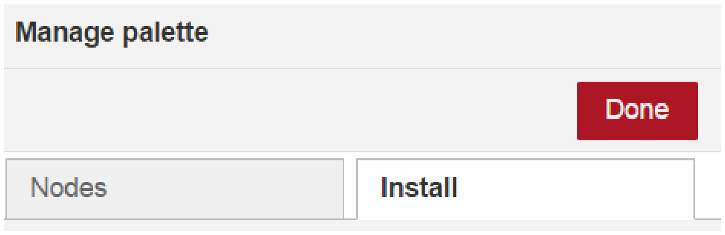
6.4. Wait until the nodes have been added:



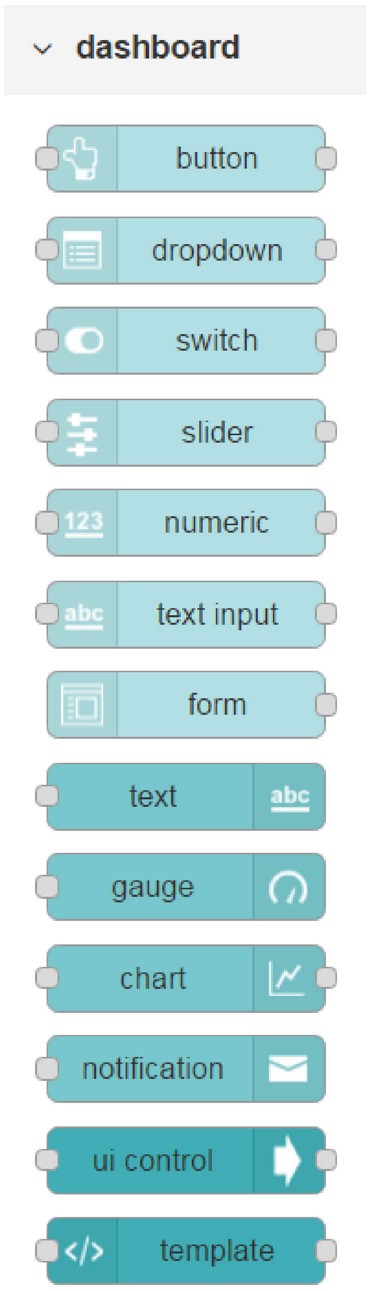
Nodes added to palette

- ui_base
- ui_button
- ui_dropdown
- ui_switch
- ui_slider
- ui_numeric
- ui_text_input
- ui_form
- ui_text
- ui_gauge
- ui_chart
- ui_toast
- ui_ui_control
- ui_template
- ui_link
- ui_tab
- ui_group

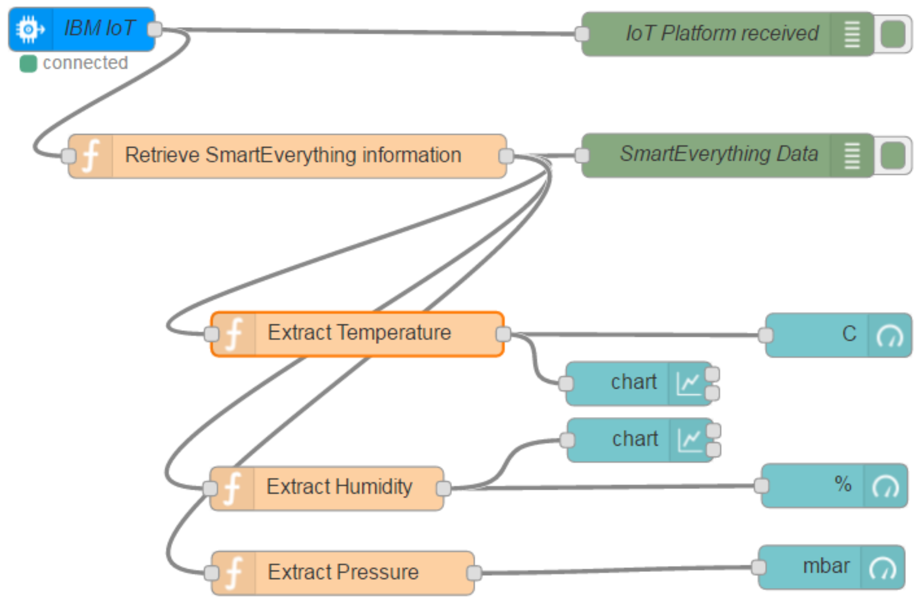
6.5. Press Done:



6.6. Reload the web page and scroll down to the bottom of the nodes and you will now see this:

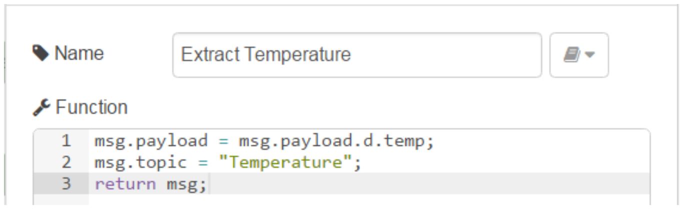


Then we are ready to use the newly added Dashboard capabilities in our application. The following steps will guide you to change the “IoT Platform” flow to look like this:



6.6.1.First add the three function nodes and connect them to the “Retrieve SmartEverything information” function node:

Extract Temperature – the msg.topic value will be shown in the chart node replacing “data”:


Name: Extract Temperature

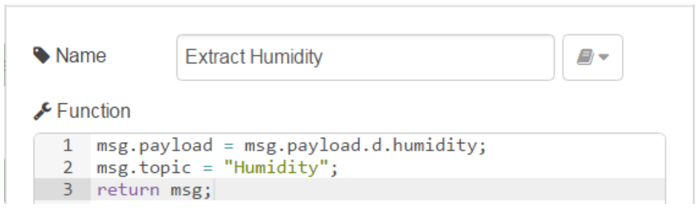
Function

```
msg.payload = msg.payload.d.temp;
msg.topic = "Temperature";
return msg;
```

– in text:

```
msg.payload = msg.payload.d.temp;
msg.topic = "Temperature";
return msg;
```

Extract Humidity:



Name: Extract Humidity

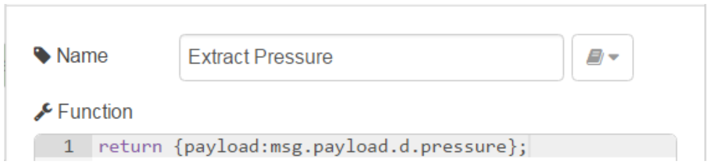
Function

```
msg.payload = msg.payload.d.humidity;
msg.topic = "Humidity";
return msg;
```

– in text:

```
msg.payload = msg.payload.d.humidity;
msg.topic = "Humidity";
return msg;
```

Extract Pressure:



Name: Extract Pressure

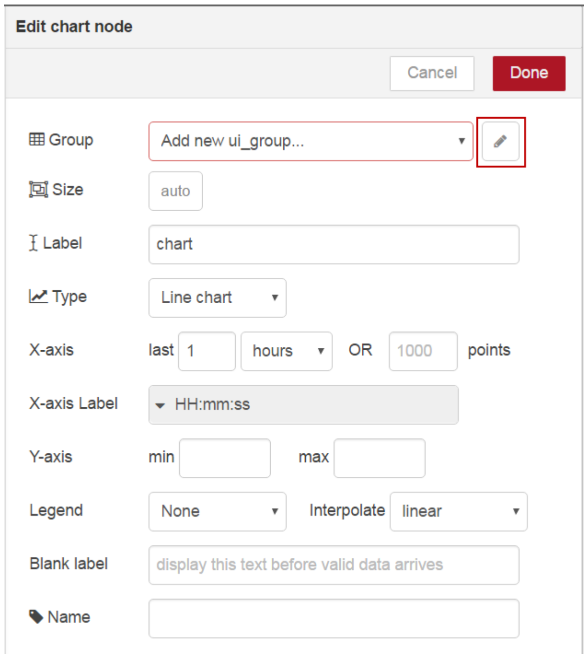
Function

```
return {payload:msg.payload.d.pressure};
```

– in text:

```
return {payload:msg.payload.d.pressure};
```

6.6.2. Deploy the application. Drag in a dashboard chart node and start configuring it by adding a new Group to the dashboard – click at the pencil:



Group: Add new ui_group...

Size: auto

Label: chart

Type: Line chart

X-axis: last 1 hours OR 1000 points

X-axis Label: HH:mm:ss

Y-axis: min max

Legend: None Interpolate: linear

Blank label: display this text before valid data arrives

Name:

6.6.3.We need to create the first Tab by clicking the pencil:

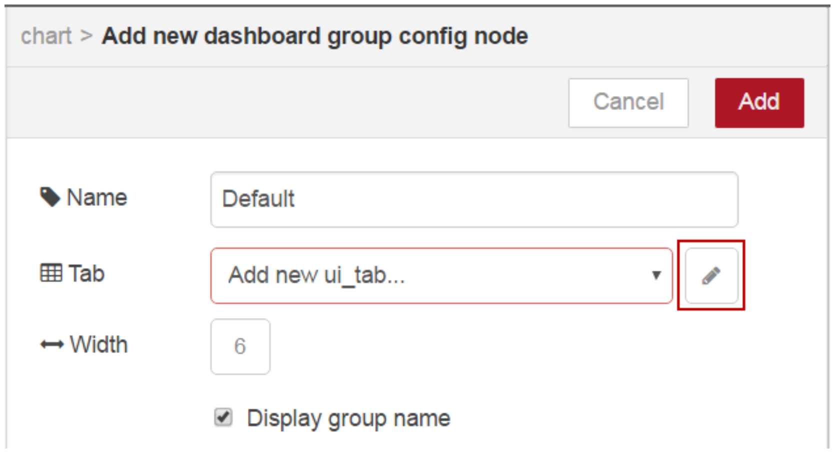


chart > **Add new dashboard group config node**

Name	Default
Tab	Add new ui_tab...
Width	6

☑ Display group name

6.6.4.Give it a name like SmartEverything:

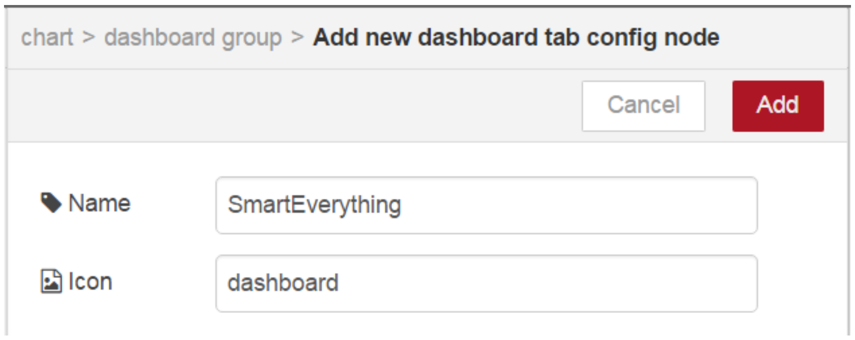


chart > dashboard group > **Add new dashboard tab config node**

Name	SmartEverything
Icon	dashboard

6.6.5.Press Add. Change the Width to 10 and give it a Name, e.g. Indoor 1:

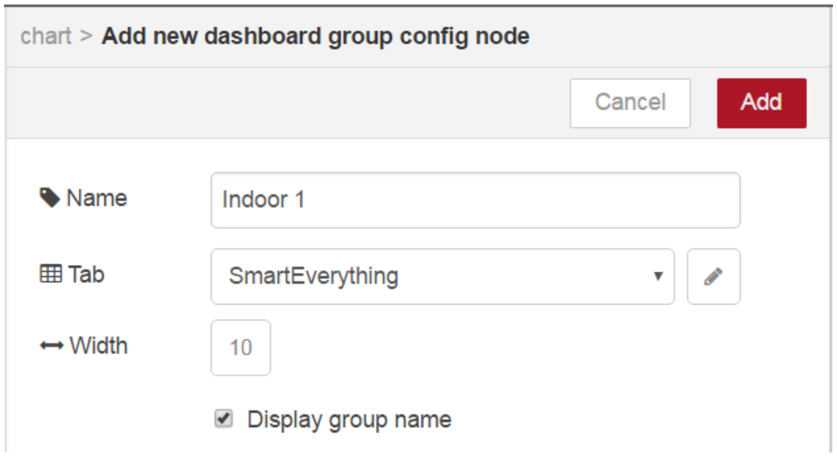
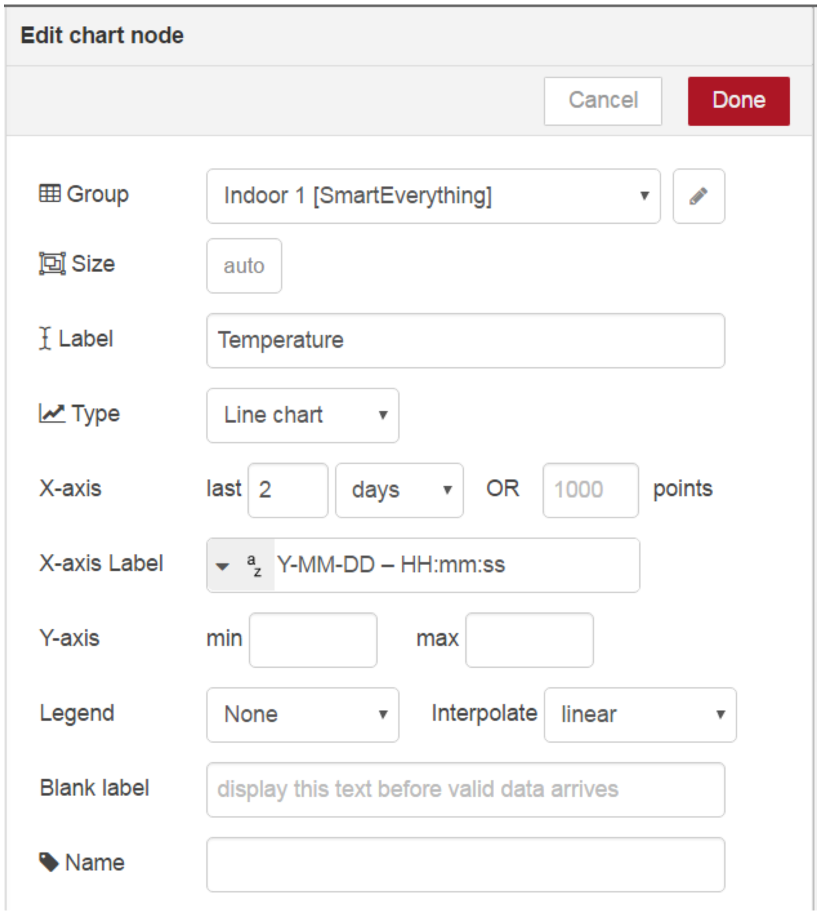


chart > **Add new dashboard group config node**

Name	Indoor 1
Tab	SmartEverything
Width	10

☑ Display group name

6.6.6.Press Add. Label it Temperature, change the X-axis to last 2 days, and define a custom X-axis Label as “Y-MM-DD – HH:mm:ss”:


Edit chart node

Group Indoor 1 [SmartEverything]

Size auto

Label Temperature

Type Line chart

X-axis last 2 days OR 1000 points

X-axis Label Y-MM-DD – HH:mm:ss

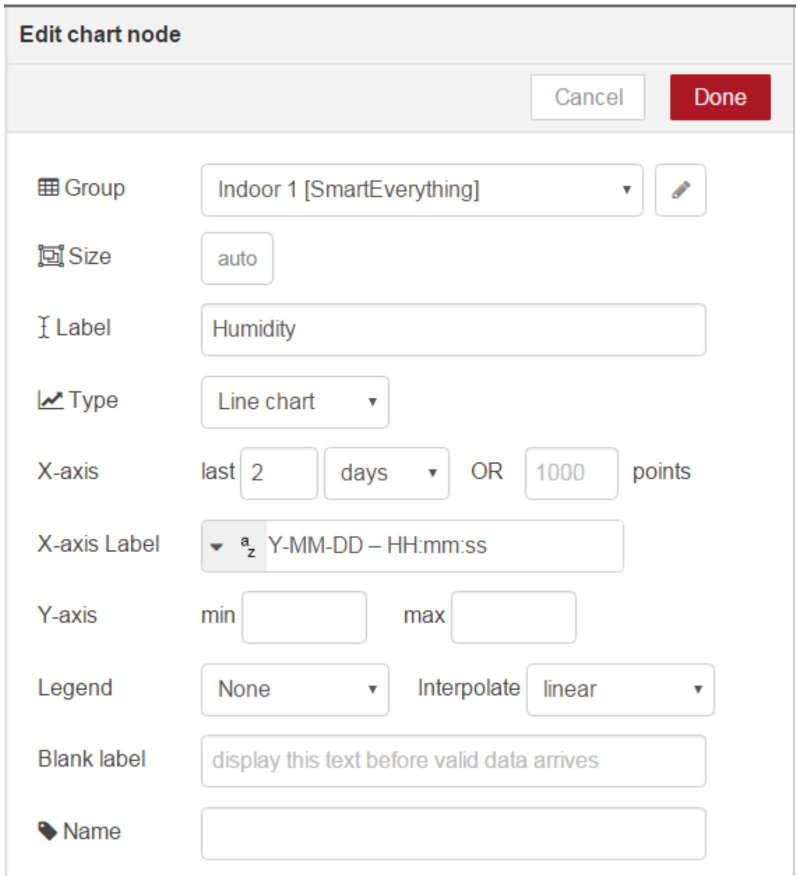
Y-axis min max

Legend None Interpolate linear

Blank label display this text before valid data arrives

Name

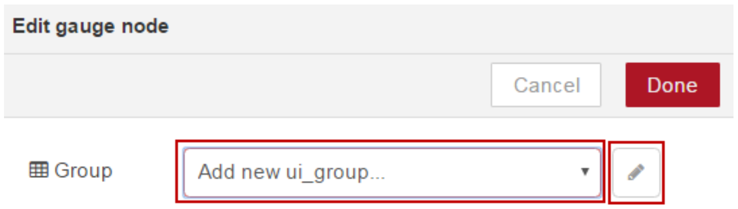
6.6.7. Press Done and connect the chart node to the “Extract Temperature” function node.
Add another Chart node. Label it Humidity, change the X-axis to last 2 days,
and define a custom X-axis Label as “Y-MM-DD – HH:mm:ss”:



Edit chart node

Group: Indoor 1 [SmartEverything]
Size: auto
Label: Humidity
Type: Line chart
X-axis last 2 days OR 1000 points
X-axis Label Y-MM-DD – HH:mm:ss
Y-axis min max
Legend: None Interpolate: linear
Blank label: display this text before valid data arrives
Name:

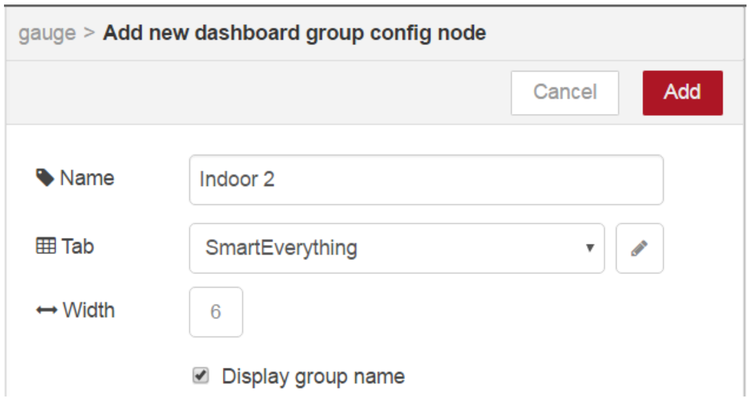
6.6.8. Press Done and connect the chart node to the “Extract Humidity” function node.
Drag in a dashboard gauge node and connect it to the “Extract Temperature” function node.
Configure the gauge node: change the Group to “Add new ui_group” and press the pencil:



Edit gauge node

Group: Add new ui_group...

6.6.9.Give it another name, e.g. Indoor 2:


gauge > Add new dashboard group config node

Cancel | Add

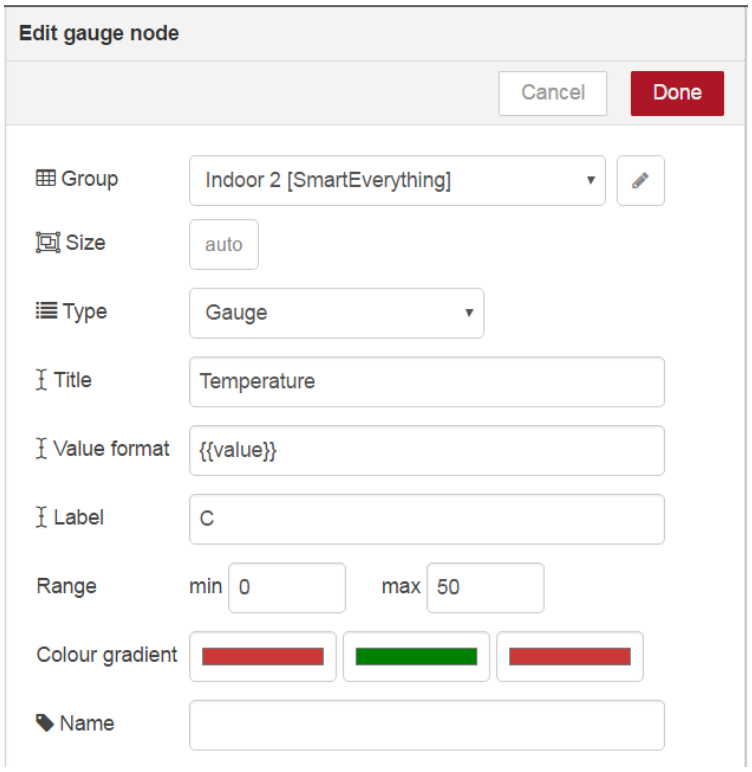
Name: Indoor 2

Tab: SmartEverything

Width: 6

☑ Display group name

6.6.10. Press Add. Change Title to Temperature. Change Label to C. Change Range to 0-50. Optionally change the Color gradients:


Edit gauge node

Cancel | Done

Group: Indoor 2 [SmartEverything]

Size: auto

Type: Gauge

Title: Temperature

Value format: {{value}}

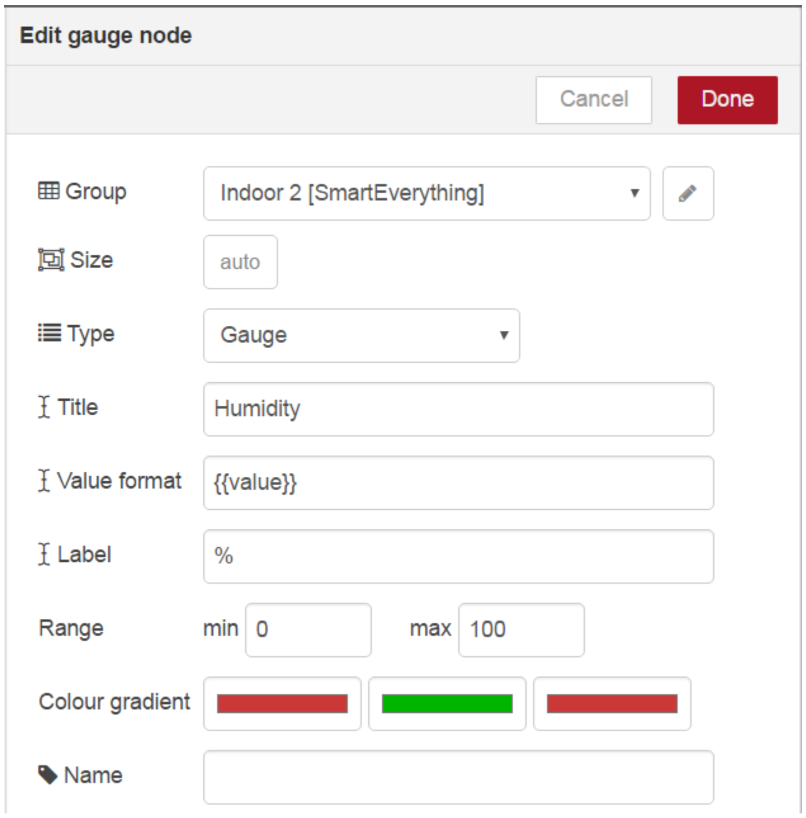
Label: C

Range: min 0 max 50

Colour gradient: 🟥 🟩 🟥

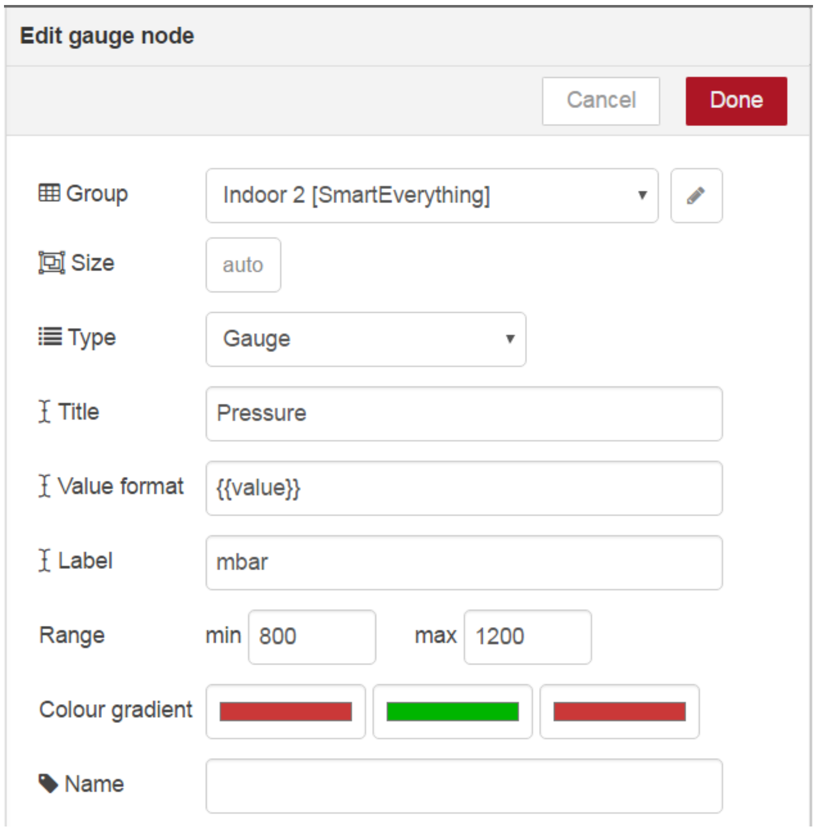
Name:

6.6.11. Press Done. Drag in a dashboard gauge node and connect it to the “Extract Humidity” function node.
Change Group to Indoor 2. Change Title to Humidity. Change Label to %. Change Range to 0-100.
Optionally change the Color gradients:


Edit gauge node

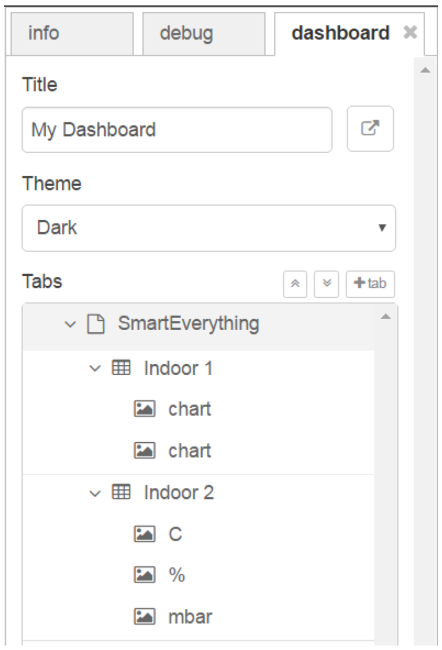
Group	Indoor 2 [SmartEverything]
Size	auto
Type	Gauge
Title	Humidity
Value format	{{value}}
Label	%
Range	min 0 max 100
Colour gradient	🟥 🟩 🟥
Name	

6.6.12. Press Done. Drag in a dashboard gauge node and connect it to the “Extract Pressure” function node.
Change Group to Indoor 2. Change Title to Pressure. Change Label to mbar. Change Range to 800-1200.
Optionally change the Color gradients:



Group	Indoor 2 [SmartEverything]
Size	auto
Type	Gauge
Title	Pressure
Value format	{{value}}
Label	mbar
Range	min 800 max 1200
Colour gradient	🟥 🟩 🟥
Name	

6.6.13. Press Done. Select the dashboard tab in the upper right corner. Select the Dark Theme. Optionally change the Title.
Expand the two Groups and this is what you should see:



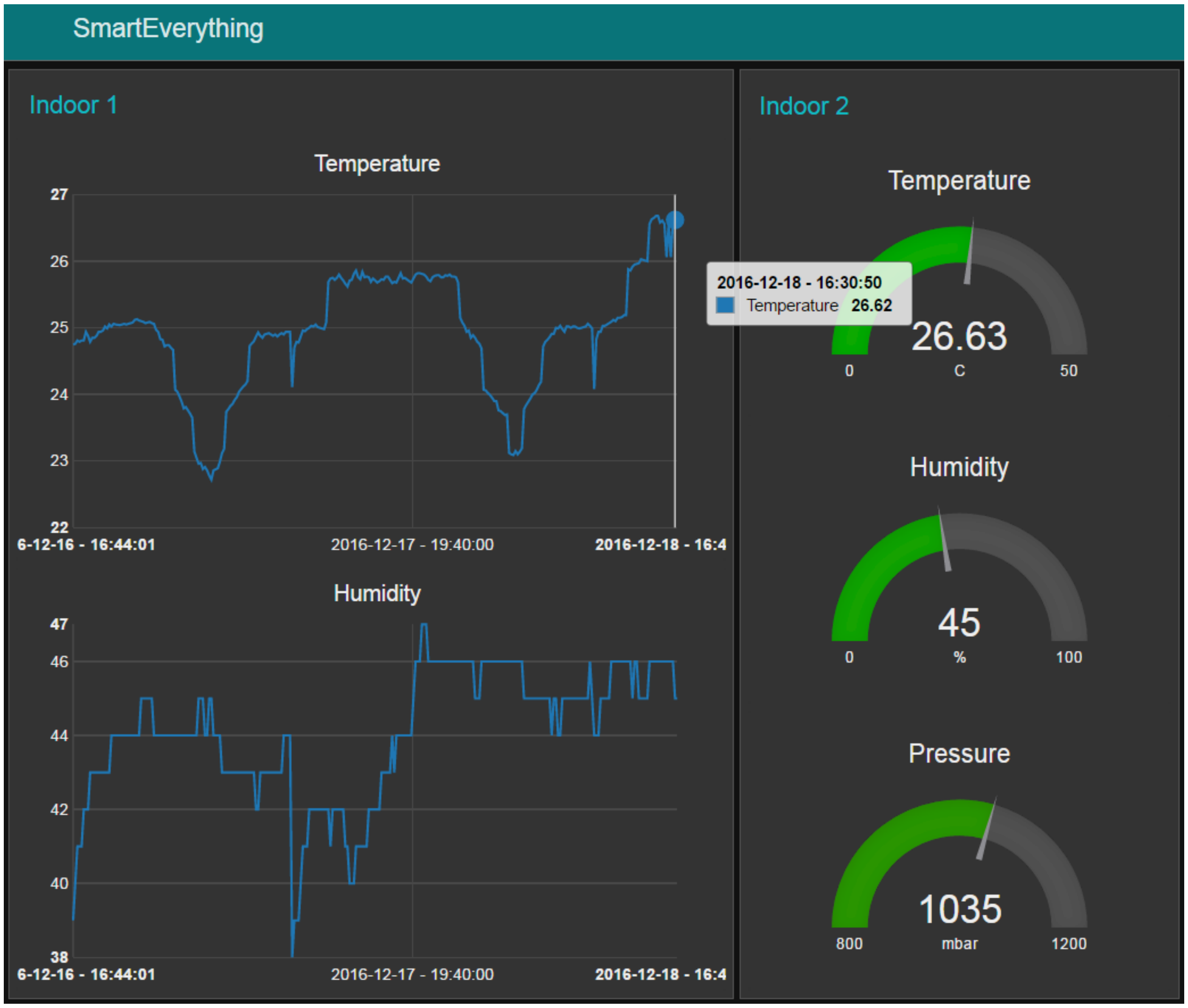
Title: My Dashboard

Theme: Dark

Tabs:
- SmartEverything
  - Indoor 1
    - chart
    - chart
  - Indoor 2
    - C
    - %
    - mbar

6.6.14. Deploy the application.

6.6.15. You access the dashboard by changing the url in your browser to point at /ui in your Bluemix application,
e.g. http://sigfoxgw.eu-gb.mybluemix.net/ui.
Enough data has flown in after two days and the dashboard could look like this:



Note: Moving the cursor into the Temperature chart shows the measured temperature at a given time. The format is defined by the msg.topic =”Temperature”; in step 1 and the customized X-axis Label in step 6.