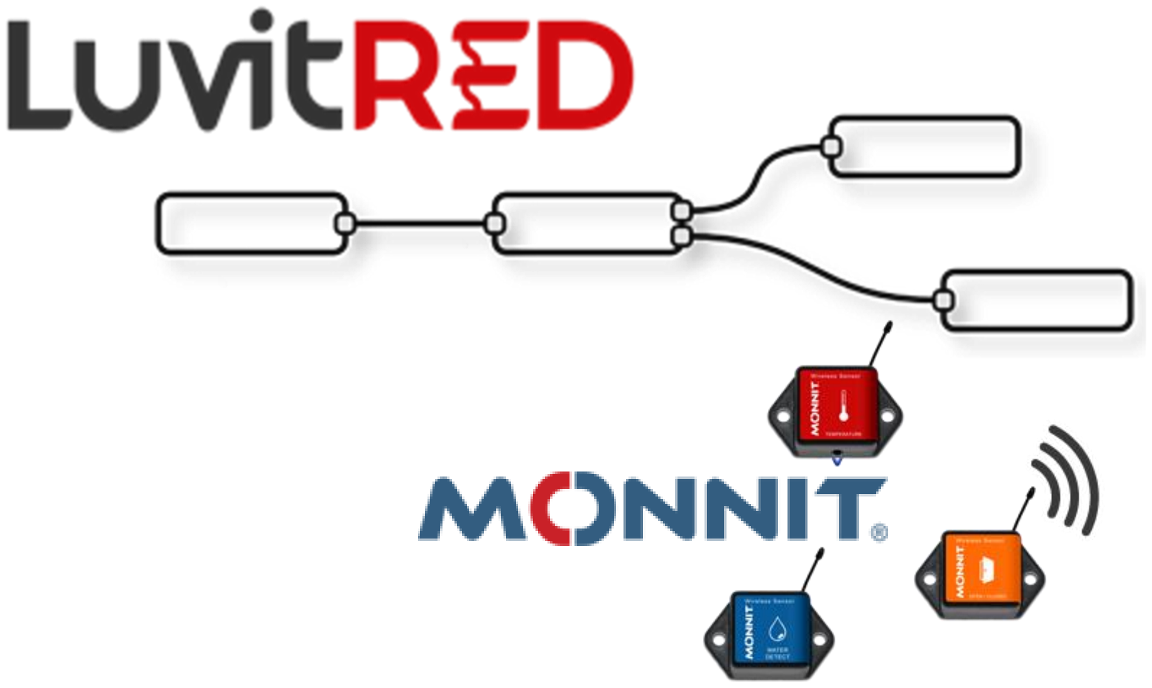


# Monnit sensors using LuvitRED

Franco Arboleda

27-Jun-16

## Table of Contents

# 1 Introduction

This document explains how to connect Monnit sensors to the CloudGate and use them on LuvitRED.

This document was written using CloudGate's firmware version 2.62.4 and LuvitRED version 2.8.0. Although older versions of firmware and LuvitRED might work in the same way, we strongly recommend to upgrade to the above mentioned versions or newer in order to ensure the same results.

The following are the different variants of Monnit back slot expansion cards from Option:

| Option Part # | Frequency Band | WiFi Support? |
|---|---|---|
| CG2103 | 868 MHz | Yes |
| CG2104 | 900 MHz | Yes |
| CG2105 | 433 MHz | Yes |
| CG2111 | 868 MHz | No |
| CG2112 | 900 MHz | No |
| CG2113 | 433 MHz | No |

Table 1: Monnit expansion card variants.

The Option card and Monnit's sensor operate on the ISM 900 MHz (902-928 MHz) band as well as 868 MHz and 433 MHz bands.

The Monnit expansion is shown on Figure 1 below:

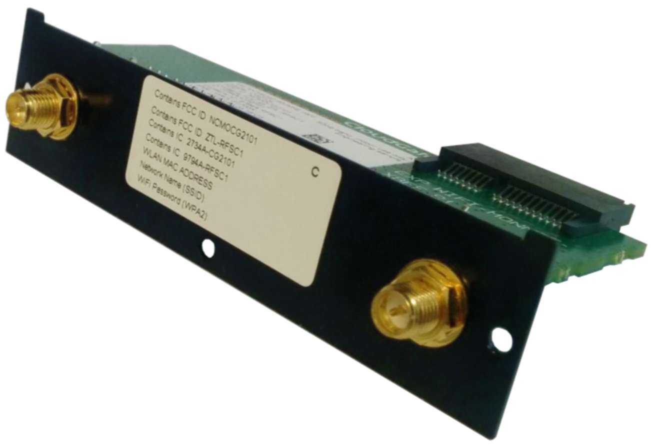


Figure 1: Monnit expansion card with Wi-Fi (CG2104).

For these configurations we are going to use the "Advanced Editor" of LuvitRED.

# 2 Hardware configuration

Insert the Monnit card on the back slot of the CloudGate (See Figure 2):

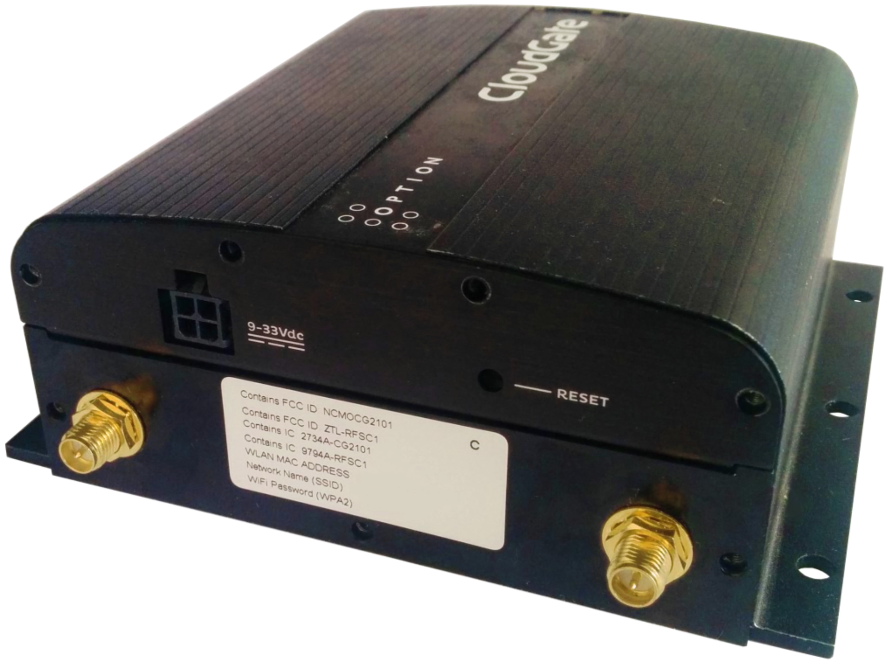


Figure 2: Monnit expansion card with Wi-Fi (CG2104) installed on CloudGate.

Install the correct antenna on the SMA-Male connectors (See Figure 3):

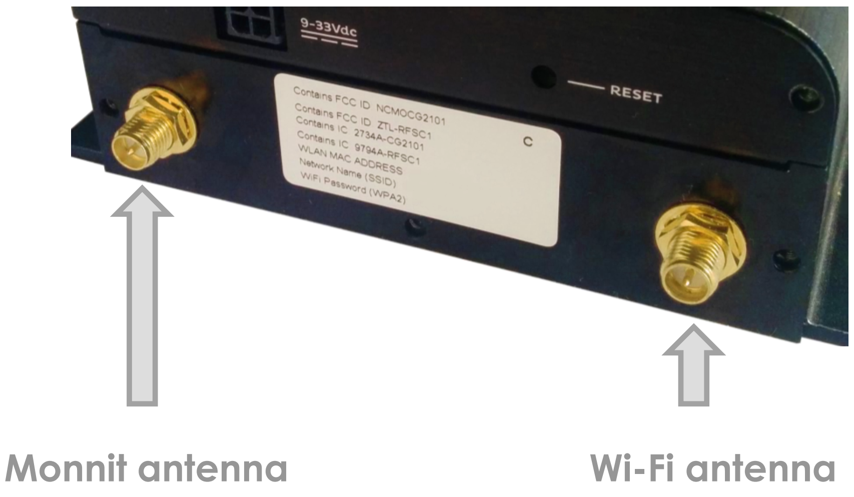


Figure 3: Correct antenna position.

**Note:** Make sure your sensors do not have the battery inserted in them yet.

# 3 LuvitRED configuration.

Go to the web interface of the CloudGate and then to the "Plugin" tab and sub-tab called "LuvitRED"

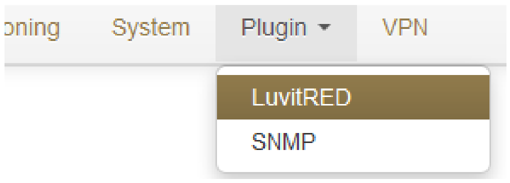


Figure 4: Plugin tab, LuvitRED.

**<u>Note</u>**: Do not focus on the SNMP (Simple Network Management Protocol) sub-tab. This tab is not going to be used on this document.

Under the "Advanced Editor" of LuvitRED, there are two nodes that are related to Monnit (See Figure 5):

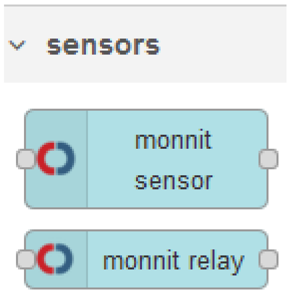


Figure 5: Monnit nodes.

In this document we only focused on the "monnit sensor" node. The "monnit relay" node is used to control the Monnit relay/control sensors, which are not going to be covered in this document.

## 3.1 Receiving data from the Monnit sensors.

1. Drag and drop a "monnit sensor" node:

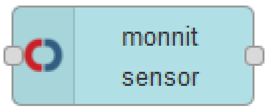
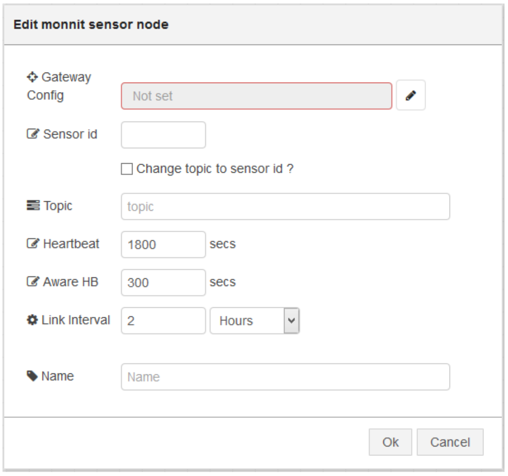


Figure 6: monnit sensor node.

2. Configure the monnit sensor node in the following way:

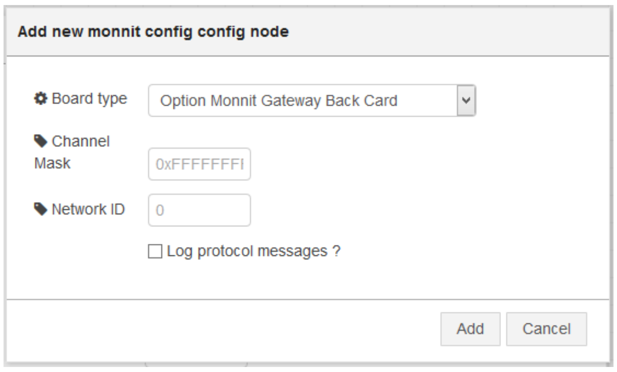


Edit monnit sensor node

| | |
|---|---|
| Gateway Config | Not set |
| Sensor id | |
| | ☐ Change topic to sensor id ? |
| Topic | topic |
| Heartbeat | 1800 secs |
| Aware HB | 300 secs |
| Link Interval | 2 Hours |
| Name | Name |

Figure 7: Default monnit sensor node.

a. Add a **new** Gateway configuration by clicking on the pencil icon and configure the "Board type" for the **Option Monnit Gateway Back Card** and click on "Add":



Add new monnit config config node

| | |
|---|---|
| Board type | Option Monnit Gateway Back Card |
| Channel Mask | 0xFFFFFFFF |
| Network ID | 0 |
| | ☐ Log protocol messages ? |

Figure 8: Selecting the back slot card.

i. The "Channel Mask" is by default 0xFFFFFFFF, this means that the CloudGate will allow the 32 available channels in order to find sensors.
ii. The "Network ID" is by default 0, this means that the ID is randomly generated. Change this value manually to make sure there are no

conflics when using more than one Gateway. The maximum value for a network ID is **255**.

b. On the node configuration type the "Sensor id" of the monnit sensor we are going to monitor (See Figure 9 for an example on where to find the sensor information). We are going to use a temperature sensor for this example, because of this, we will type **temperature** as the "Topic". We are going to change the "Heartbeat" to 10 seconds and the "Aware HB" to 5 seconds (these heartbeats are just for our testing). Finally, we are going to change the "Name" of the node to **Temp_Sensor**. Click on "OK" when finished (See Figure 10):

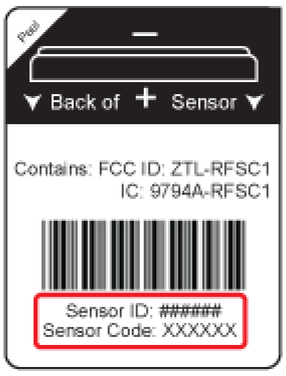
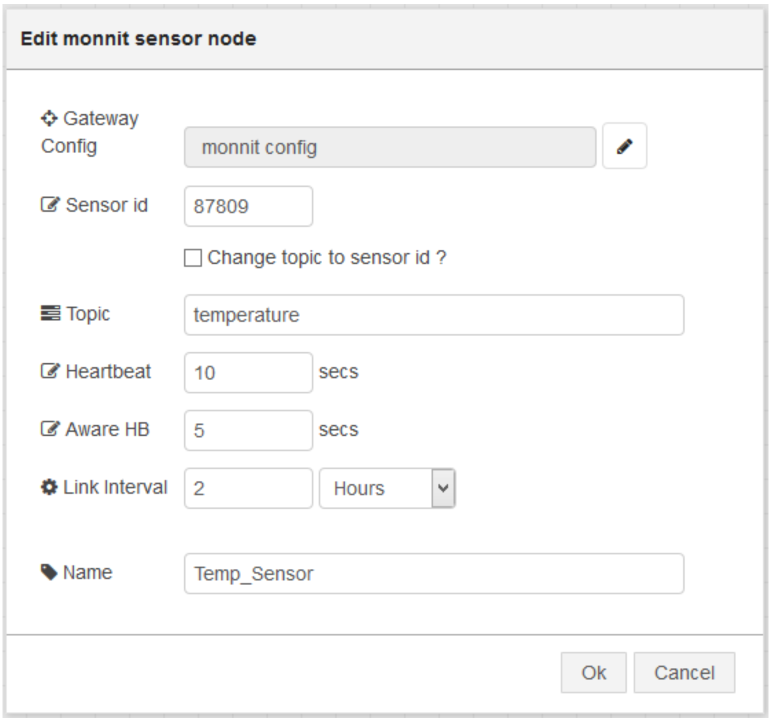


Figure 9: Monnit sensor information example (sensor bottom label).



Edit monnit sensor node

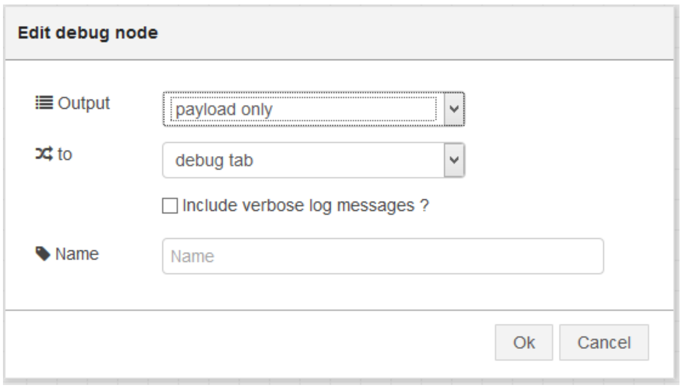
Gateway Config: monnit config
Sensor id: 87809
☐ Change topic to sensor id ?
Topic: temperature
Heartbeat: 10 secs
Aware HB: 5 secs
Link Interval: 2 Hours
Name: Temp_Sensor

Figure 10: Final node configuration.

i. The "Hearbeat" is the normal reporting interval.
ii. The "Aware HB" is a special reporting interval when the sensor enters a certain range. At this moment the range is the one programmed at factory and it could be changed using the iMonnit platform (not via LuvitRED)
iii. "Link interval" is the interval to maintain the link between the sensor and the Gateway.

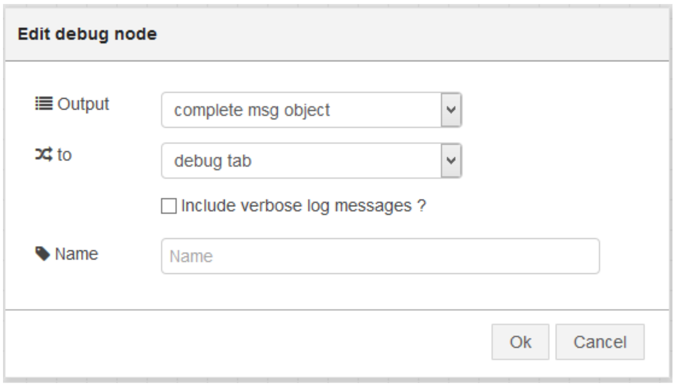
This node is now configured to send out the information from that specific temperature sensor. Now we need to display the data:

1. Drag and drop two debug nodes and configure one to output a "complete msg object" (See Figure 12 - do not change the other debug node):


Edit debug node

Output: payload only
to: debug tab
☐ Include verbose log messages ?
Name: Name

Figure 11: Standard debug node configuration.


Edit debug node

Output: complete msg object
to: debug tab
☐ Include verbose log messages ?
Name: Name

Figure 12: complete msg object debug node configuration.

**<u>Note</u>**: We actually only need one debug node to show the values coming from the sensor, but for this example we are using two debug nodes to compare the information shown by the standard one and by the "complete msg object" one (the information is always there, but the standard debug configuration does not show it all).

2. Let's connect the nodes together in the following way:

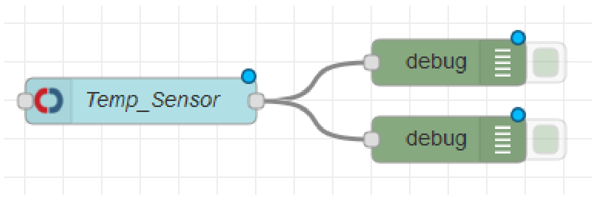


Figure 13: Final Configuration.

3. Click on "Deploy" and wait for the "Temp_Sensor" node status to say "waiting for sensor":

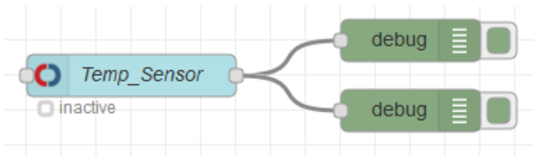


Figure 14: Temp_Sensor node status "inactive".

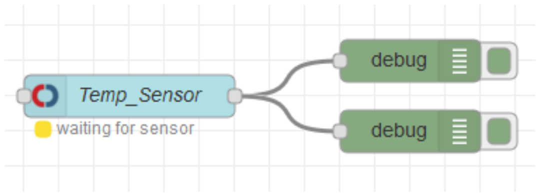


Figure 15: Temp_Sensor node status "waiting for sensor".

Now, it is time to insert the battery into the sensor and wait for it to register on our configuration (node status becomes "active"):

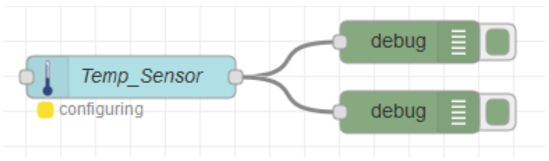


Figure 16: Temp_Sensor node status "configuring".

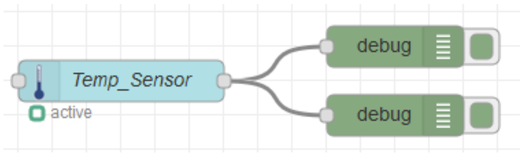
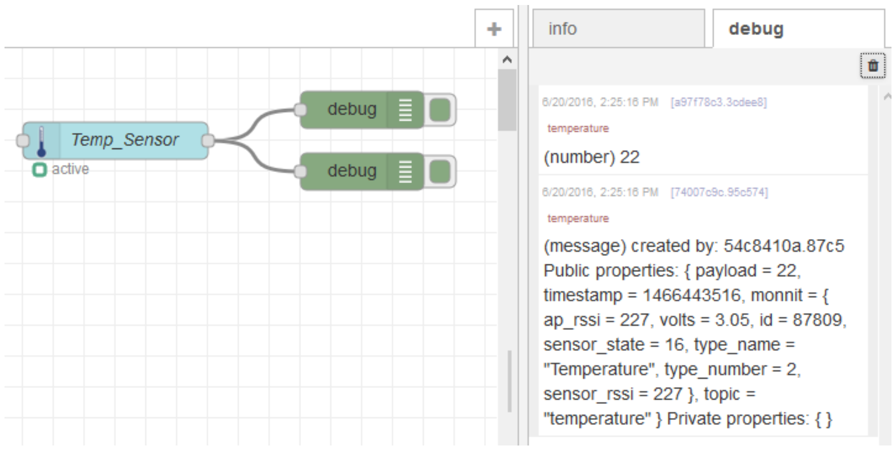


Figure 17: Temp_Sensor node status "active".

We should now be seeing the values being reported by the sensor under the debug tab:



```
6/20/2016, 2:25:16 PM   [a97f78c3.3cdee8]
temperature
(number) 22

6/20/2016, 2:25:16 PM   [74007c9c.95c574]
temperature
(message) created by: 54c8410a.87c5
Public properties: { payload = 22,
timestamp = 1466443516, monnit = {
ap_rssi = 227, volts = 3.05, id = 87809,
sensor_state = 16, type_name =
"Temperature", type_number = 2,
sensor_rssi = 227 }, topic =
"temperature" } Private properties: { }
```

**Figure 18: Values coming from the temperature sensor.**

The first line is coming from the default debug node as it only shows the payload which in the temperature sensor is just the actual temperature (in Degrees Celsius).

The second line is coming from the debug node configured to output the "complete msg object", on that line we can see the payload (the value as on the first line), but we can also see in deep information about the sensor inside the monnit section of the message:

(message)created by: 54c8410a.87c5 Public properties: { **payload = 22**, timestamp = 1466443516, **monnit = { ap_rssi = 227, volts = 3.05, id = 87809, sensor_state = 16, type_name = "Temperature", type_number = 2, sensor_rssi = 227 }**, topic = "temperature" } Private properties: { }

The information included is:

- ap_rssi: AP RSSI
- volts: battery voltage
- id: sensor ID
- sensor_state: sensor state (numeric)
- type_name: type of sensor (text)
- type_number: type of sensor (numeric)
- sensor_rssi: sensor RSSI

More sensors can be included into the configuration using the same steps explained above.

## 3.2 Remote access to the Monnit data using a local TCP server

For this example we are going to start a local TCP server running on the CloudGate that will print a formatted version of the same information we saw over the debug tab, but on a TCP connection.

This TCP connection can be started remotely by a computer that wants to obtain the readings from the Monnit sensor.

Of course, this is not the only possibility with LuvitRED, we could also send this information to a remote server or take an action (send an SMS or an email) if a value is higher or lower than a threshold, etc.

Drag and drop a TCP out node:

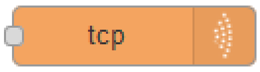
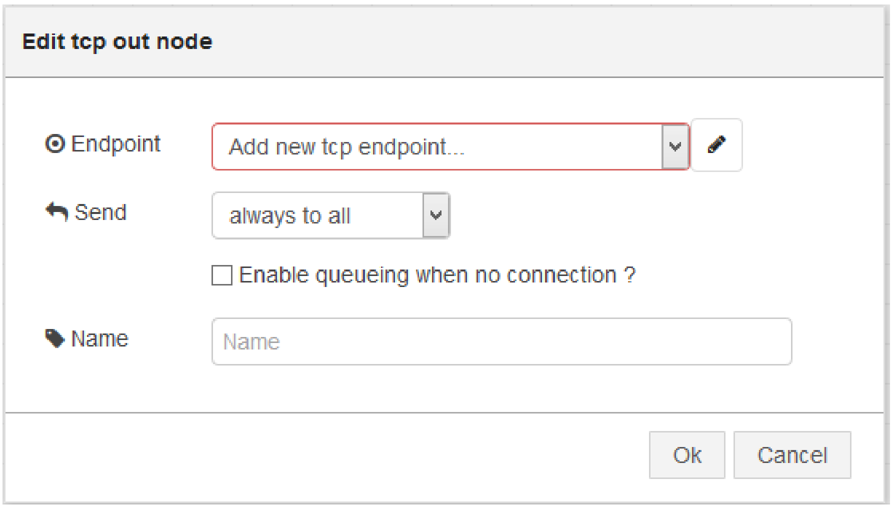


Figure 19: TCP out node.

In order to start a local TCP server on the CloudGate, we need to modify the following parameters on the TCP output node:



Edit tcp out node

Endpoint: Add new tcp endpoint...

Send: always to all

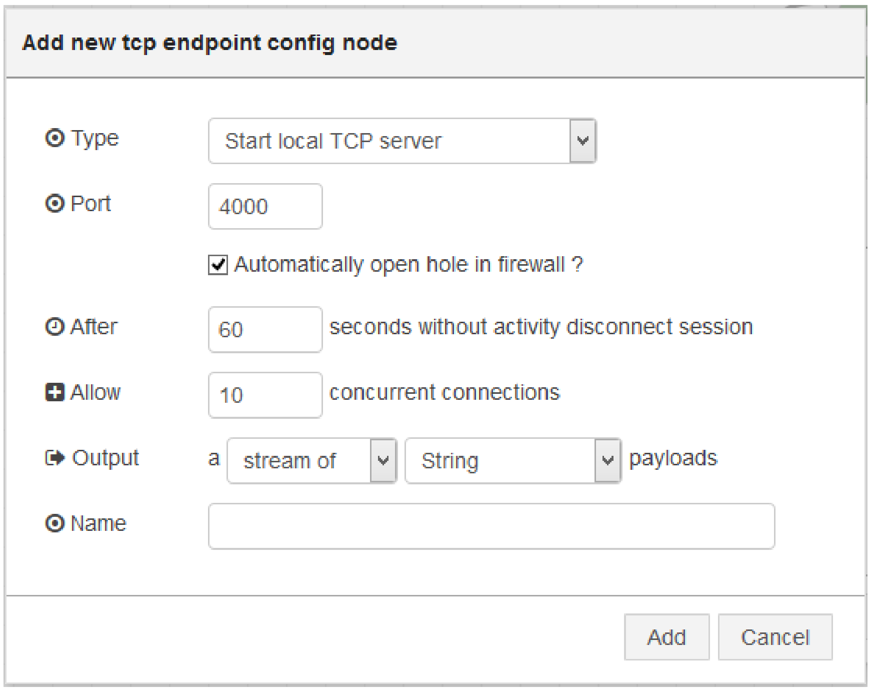
☐ Enable queueing when no connection ?

Name: Name

Figure 20: Default TCP out node.

Add a new "Endpoint" by clicking over the pencil icon. The new endpoint should have the following configuration:

1. Change the "Type" to "Start local TCP server"
2. Change the "Port" to any port you would like to use for this test. On this example we are going to use TCP port 4000
3. Check the "Automatically open hole in firewall?" setting to allow incoming WAN connections to the port.
4. Keep the rest of the settings as default.



**Add new tcp endpoint config node**

Type: Start local TCP server

Port: 4000

☑ Automatically open hole in firewall ?

After: 60 seconds without activity disconnect session

Allow: 10 concurrent connections

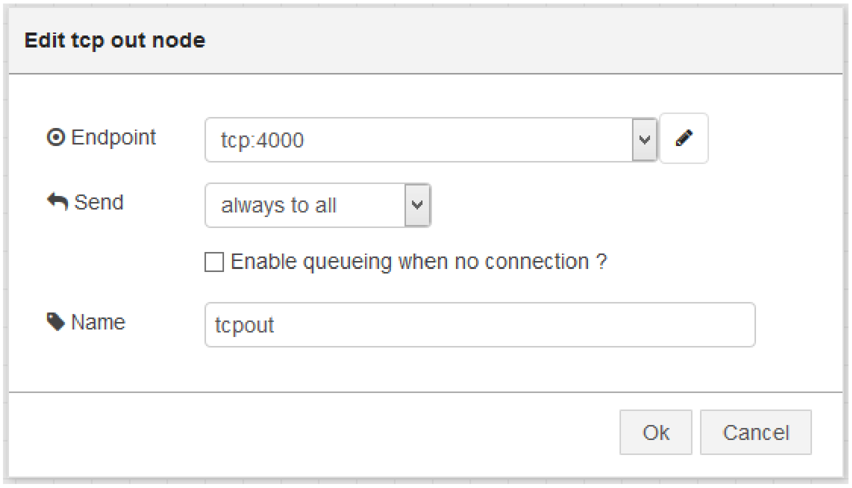
Output: a stream of String payloads

Name:

**Figure 21: Configuration of the TCP endpoint.**

Click on "Add" to close the "Endpoint" configuration.

Once back on the node configuration, change the name of the node to something descriptive like ***tcpout:***


Edit tcp out node

Endpoint: tcp:4000

Send: always to all

☐ Enable queueing when no connection ?

Name: tcpout

Figure 22: TCP out node configuration finished.

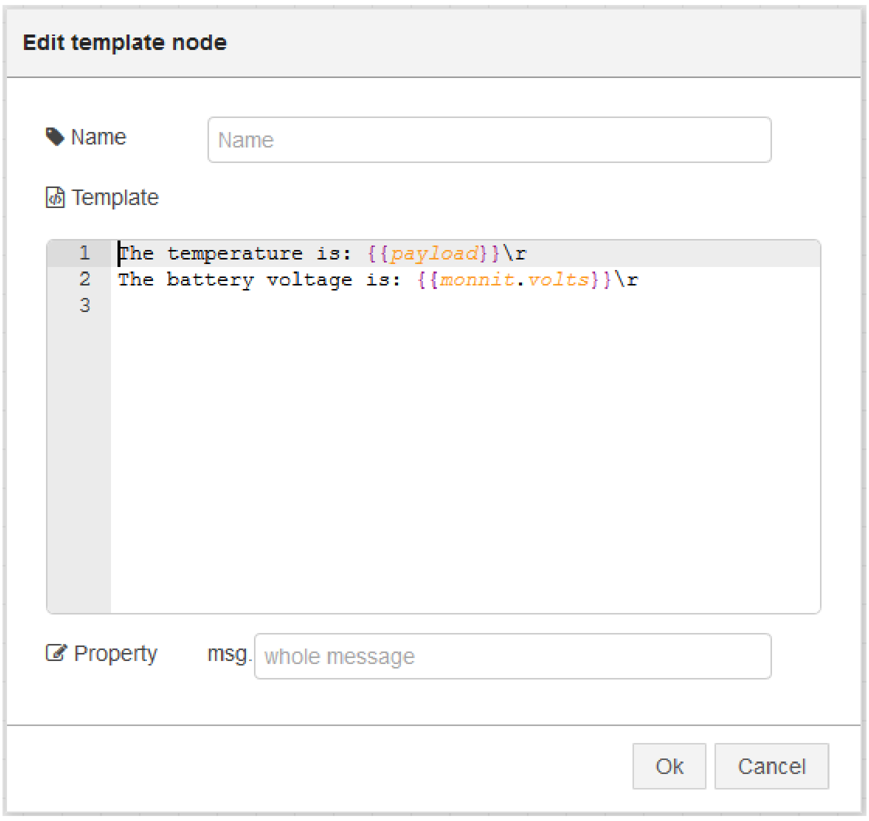
Drag and drop a template node:

Figure 23: Template node.

Change the content of the template node to (See Figure 24):

**The temperature is: {{payload}}\r**

**The battery voltage is: {{monnit.volts}}\r**



**Edit template node**

Name: Name

Template

```
The temperature is: {{payload}}\r
The battery voltage is: {{monnit.volts}}\r

```

Property msg. whole message

Figure 24: Template node configured.

**Note**: Mind the extra line (line 3). This help with the visualization of the data once we are connected to the TCP server.

Let's connect the nodes together (including the Monnit configuration we did in section 3.1) in the following way:

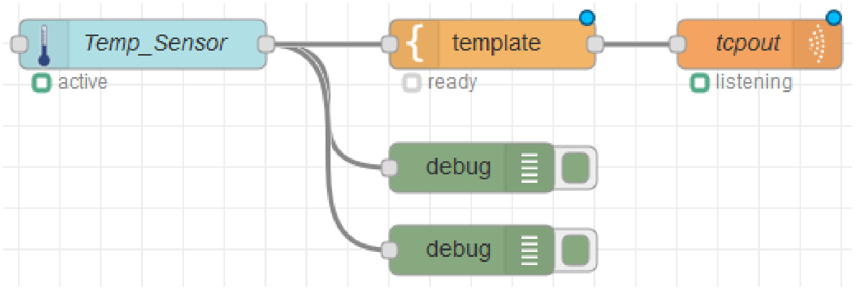


Figure 25: Final Configuration.

Click on "Deploy" and wait for the temperature sensor to connect again to the "monnit sensor" node until the configuration looks like this:

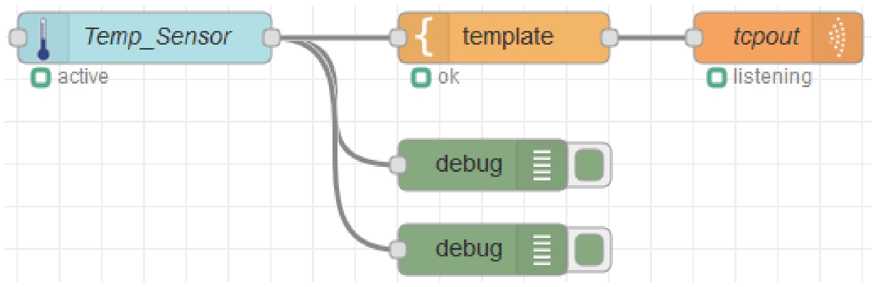
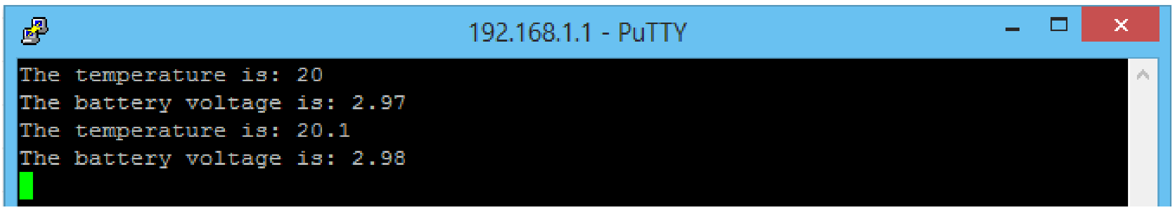


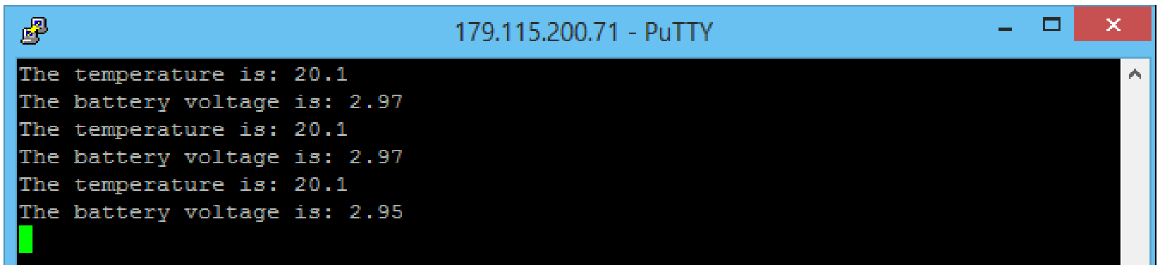
Figure 26: Configuration working.

If we now try to open a TCP connection to the CloudGate on port 4000, we should be able to get the following output:



```
The temperature is: 20
The battery voltage is: 2.97
The temperature is: 20.1
The battery voltage is: 2.98
```

Figure 27: Monnit output on a local TCP connection (CloudGate local IP).



```
The temperature is: 20.1
The battery voltage is: 2.97
The temperature is: 20.1
The battery voltage is: 2.97
The temperature is: 20.1
The battery voltage is: 2.95
```

Figure 28: Monnit output on a remote TCP connection (CloudGate WAN IP).

OPTION