

 Sensorian.



Programming the Sensorian Shield with Node-RED

All trademarks or registered trademarks referenced herein are the properties of their respective owners. The latest version of this booklet is available online at: www.sensorian.io

Copyrights

© Sensorian, 2015. The information contained herein is subject to change without notice. Sensorian assumes no responsibility for the use of any circuitry other than circuitry embodied in a Sensorian product. Nor does it convey or imply any license under patent or other rights. The Sensorian shield is not warranted nor intended to be used for medical, life support, lifesaving, critical control or safety applications, unless pursuant to an express written agreement with Sensorian. Sensorian reserves the right to make changes without further notice to the materials described herein. Sensorian does not assume any liability arising out of the application or use of any product or circuit described herein. Sensorian does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Sensorian product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Sensorian against all charges.

Regulatory Compliance

The Sensorian Kit is intended for use as a development platform for hardware or software in a laboratory environment. The board is an open system design, which does not include a shielded enclosure. Due to this reason, the board may cause interference to other electrical or electronic devices in close proximity. In a domestic environment, this product may cause radio interference. In such cases, the user may be required to take adequate preventive measures. Also, this board should not be used near any medical equipment or RF devices. Attaching additional wiring to this product or modifying the product operation from the factory default may affect its performance and cause interference with other apparatus in the immediate vicinity. If such interference is detected, suitable mitigating measures should be taken. The Sensorian Kit as shipped from the factory has been verified to meet with requirements of CE as a Class A product.



Handling Boards

Sensorian boards are sensitive to ESD. Hold the board only by its edges. After removing the board from its box, place it on a grounded, static free surface. Use a conductive foam pad if available. Do not slide board over any surface.

Table of Contents

Acknowledgment	4
Introduction.....	5
Project 1: Controlling the Sensorian LED.....	6
Project 2: Using the Real-Time Clock Alarm Trigger	9
Project 3: Using the Capacitive Touch Buttons	11
Project 4: Using the TFT-LCD with Basic Web Services	14
Project 5: Using Advanced Web APIs.....	17
Project 6: Environmental Data Server	25

Acknowledgment

Special thanks to Michael Lescisin for his contributions to the development of this booklet.

Introduction

Node-RED is a tool for creating “Internet of Things” applications in a web browser by wiring together various types of nodes. Each type of node in Node-RED performs a specific function. For example, there are nodes which will retrieve the contents of a web page. Everything in Node-RED is event driven. Node-RED nodes can receive events through their inputs and can create events on their outputs. There are nodes created for integrating Sensorian in a Node-RED application. Each sensor on the Sensorian board has a corresponding node in Node-RED. For example, there is a “touchpad” node in Node-RED which will output an event whenever a capacitive touch button is pressed on the Sensorian. This node’s output event will contain the number of the button which was pressed so that upstream nodes can make decisions based which button was pressed. There are also output nodes such as the “tft” node which will display the payload of an event on the Sensorian TFT-LCD display. Node-RED and the Sensorian nodes are included by default in our Raspbian image.

To use Sensorian with Node-RED open a terminal and run:

```
sudo ~/Sensorian/Handler_NodeRED/run_servers.sh
```

then open another terminal and run:

```
node-red
```

Open the browser and go to 127.0.0.1:1880 to use Node-RED on the Pi or <IP-Address-of-RPi>:1880 to access Node-RED from a desktop computer.

Sample workflows can be found under `~/Sensorian/Apps_NodeRED`

Project 1: Controlling the Sensorian LED

Project idea: The LED on the Sensorian board can be controlled in Node-RED by sending events to the “sensorian-led” node.

To get started:

- 1) Click and drag the “sensorian-led” node onto the worksheet.

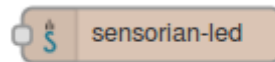


Figure 1: Sensorian LED has been added to the worksheet

- 2) Click and drag two “inject” nodes onto the worksheet.

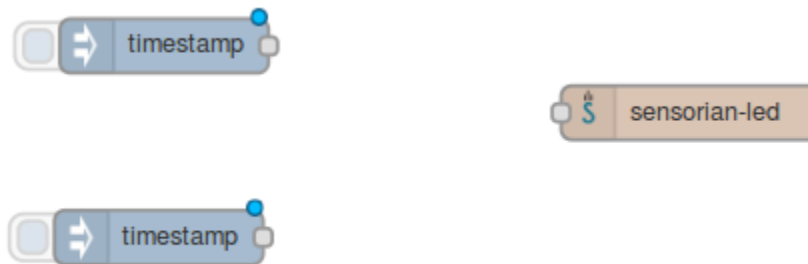


Figure 2: Two inject nodes have been added to the worksheet

- 3) Double-click on the first “inject” node and set its “Payload” to “string” – “1”. Set its “Name” to “ON”. Click “Ok”.

Edit inject node

Payload

string
▼

1

Topic

topic

Repeat

none
▼

Fire once at start ?

Name

ON

Note: "interval between times" and "at a specific time" will use cron. See info box for details.

Ok

Cancel

Figure 3: Configuring the ON inject node

- 4) Double-click on the second “inject” node and set its “Payload” to “string” – “0”. Set its “Name” to “OFF”. Click “Ok”.

Edit inject node

Payload

string
▼

0

Topic

topic

Repeat

none
▼

Fire once at start ?

Name

OFF

Note: "interval between times" and "at a specific time" will use cron. See info box for details.

Ok

Cancel

Figure 4: Configuring the OFF inject node

5) Wire both “inject” nodes into the “sensorian-led” node.



Figure 5: Wiring the control inject nodes to the Sensorian LED

6) Click “Deploy”. Once deployed, clicking on the button to the left of “ON” will turn the LED on and clicking on the button to the left of “OFF” will turn the LED off.

Project 2: Using the Real-Time Clock Alarm Trigger

Project idea: The alarm in the Sensorian Real-Time clock can be programmed to trigger events in Node-RED. This example will extend the previous example to illuminate the LED when an alarm is triggered.

To get started:

- 1) Ensure you have completed the previous example, “1) Controlling the Sensorian LED”.
- 2) Click and drag a “sensorian-rtcc.alarmtrigger” node into the workspace.

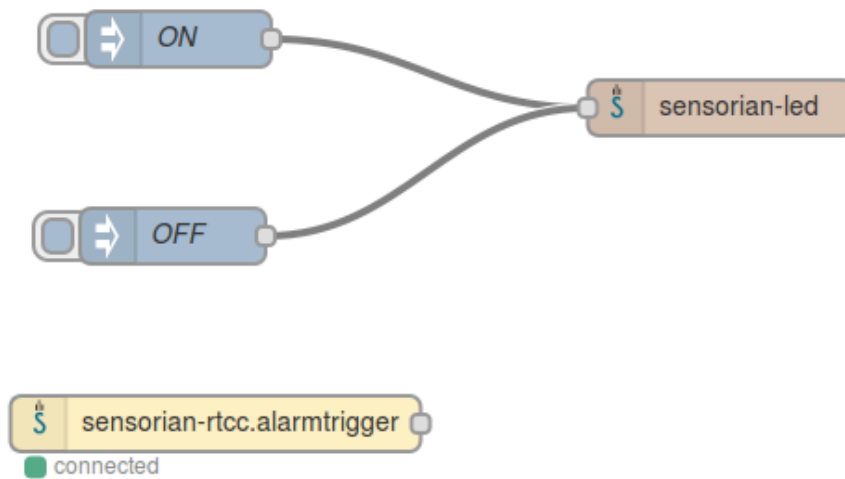


Figure 6: Add the alarmtrigger node to the worksheet of the previous project

- 3) Wire the “sensorian-rtcc.alarmtrigger” node into the “sensorian-led” node.

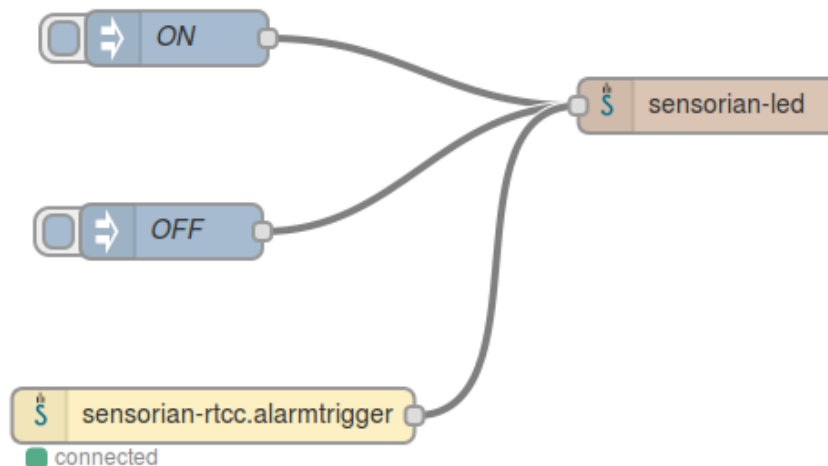
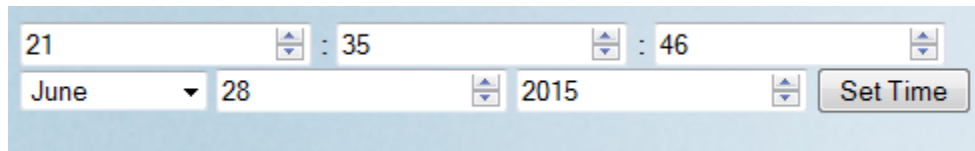


Figure 7: Wire the alarmtrigger node to the Sensorian LED

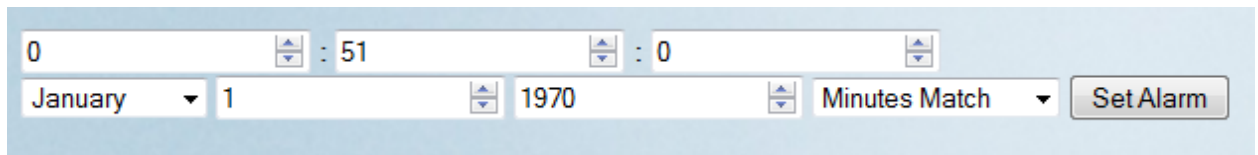
- 4) Open a new tab in your browser and go to <IP-ADDRESS-OF-RASPBERRY-PI>:5000. The time will automatically be set to your desktop's time. Click "Set Time" to set this time on the Sensorian Real-Time clock.



The screenshot shows a web interface for setting the time. It features three input fields for the hour, minute, and second, each with up and down arrows. The values are 21, 35, and 46 respectively. Below these is a dropdown menu for the month (set to June), followed by input fields for the day (28) and year (2015). A "Set Time" button is located to the right of the year field.

Figure 8: Setting the time on the Sensorian Real-Time clock through the web interface

- 5) Set the minute which you want the LED to illuminate at in the minute section of the alarm time and set the mode to "Minutes Match". Click "Set Alarm". At this minute, the LED will illuminate.



The screenshot shows a web interface for setting an alarm. It features three input fields for the hour, minute, and second, each with up and down arrows. The values are 0, 51, and 0 respectively. Below these is a dropdown menu for the month (set to January), followed by input fields for the day (1) and year (1970). A dropdown menu for the alarm mode is set to "Minutes Match". A "Set Alarm" button is located to the right of the mode dropdown.

Figure 9: Setting the alarm time on the Sensorian Real-Time clock through the web interface

Project 3: Using the Capacitive Touch Buttons

Project idea: The capacitive touch buttons can trigger events in Node-RED. This example will extend the previous example to turn off the LED whenever a touch button is pressed.

To get started:

- 1) Ensure you have completed the previous example “2) Using the Real-Time Clock Alarm trigger”.
- 2) Click and drag a “sensorian-touchpad” node into the workspace.

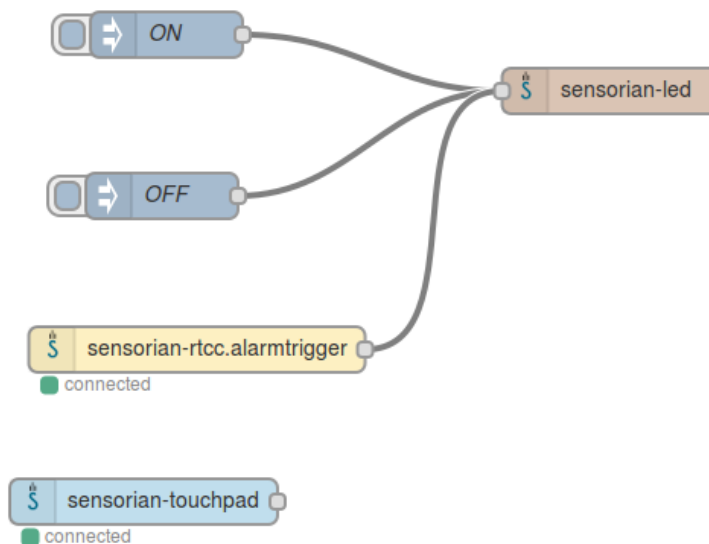


Figure 10: Add the touchpad node to the worksheet of the previous project

- 3) Click and drag a “change” node into the workspace and wire the left side to “sensorian-touchpad”

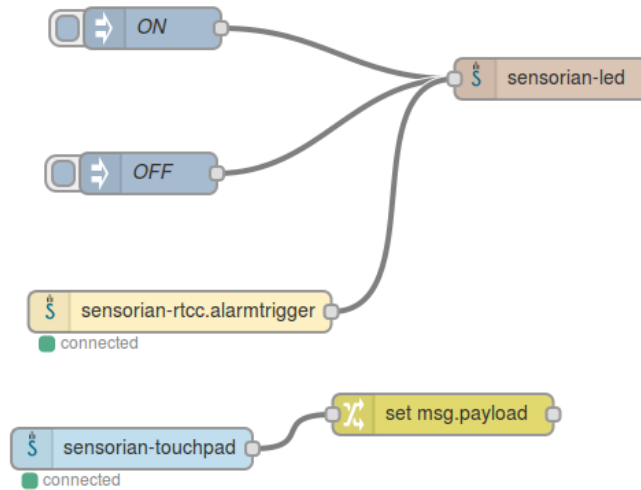


Figure 11: Add the set node to the worksheet

- 4) Double-click the “change” node and set its “msg.payload” to “0”. Click “Ok”.

Edit change node

Name

Rules

Set	▼	msg. payload	
		to	✕
		0	

Figure 12: Configure the change node

- 5) Wire the right side of the “change” node to the “sensorian-led” node.

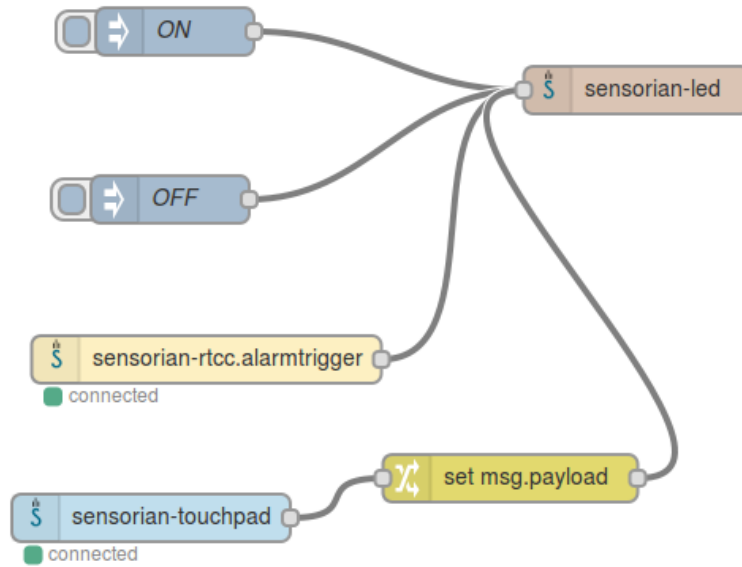


Figure 13: Connect the change node to the Sensorian LED

- 6) Now, whenever the alarm is triggered, pressing any capacitive touch button will turn off the LED.

Project 4: Using the TFT-LCD with Basic Web Services

Project idea: The TFT-LCD display can display whatever message is sent to it in a Node-RED event. Node-RED has nodes which implement a basic HTTP server. Together these can form a simple service which will print whatever is sent to the service on the LCD display.

To get started:

- 1) You may build this project on a separate worksheet.
- 2) Click and drag a “http” input node into the workspace. Double-click this node and set its “url” to “/lcd_write”. Leave its “Method” as “GET”.

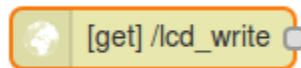


Figure 14: Add the HTTP input node

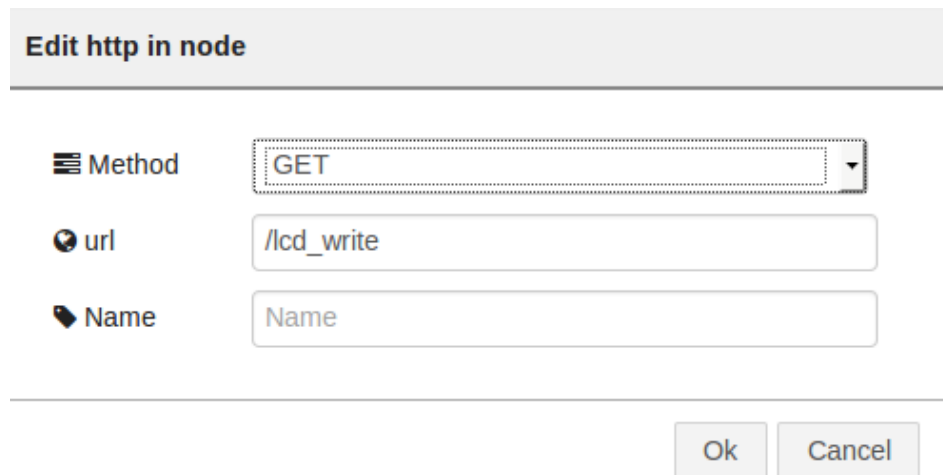
The screenshot shows the "Edit http in node" dialog box. It has a title bar "Edit http in node". Below the title bar, there are three fields: "Method" with a dropdown menu showing "GET", "url" with a text input field containing "/lcd_write", and "Name" with a text input field containing "Name". At the bottom right, there are "Ok" and "Cancel" buttons.

Figure 15: Configure the HTTP input node

- 3) Click and drag a “json” node into the workspace and wire its left side to the “http” node.



Figure 16: Add a JSON node

- 4) Click and drag a “function” node into the workspace and wire its left side to the right side of the “json” node.

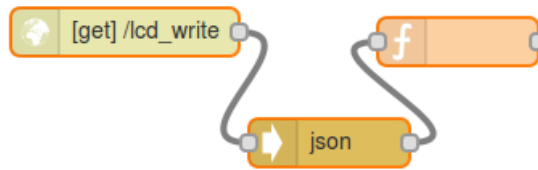


Figure 17: Add a function node

- 5) Double-click the “function” node and set its name to “GetMessage”. Set “Function” to the following. Then click “Ok”.

```
var lcd_message = JSON.parse(msg.payload);  
var message_text = lcd_message["msg"];  
var ret_message = {payload:message_text};  
return ret_message;
```

Edit function node

Name

Function

```
1 var lcd_message = JSON.parse(msg.payload);  
2 var message_text = lcd_message["msg"];  
3 var ret_message = {payload:message_text};  
4 return ret_message;
```

Outputs

See the Info tab for help writing functions.

Ok Cancel

Figure 18: Program the function node

- 6) Click and drag a “sensorian-tft” node into the workspace. Wire it to the right side of the “GetMessage” node.

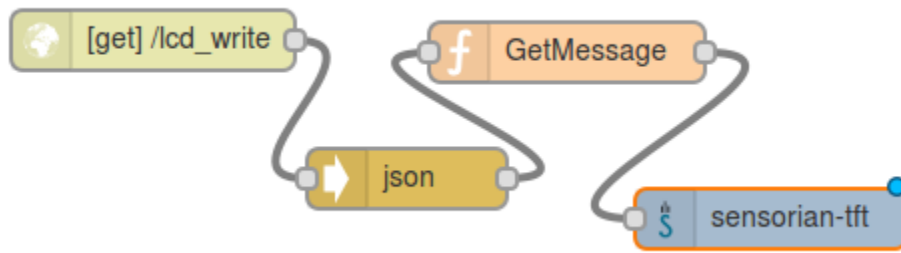


Figure 19: Wire the function node to sensorian-tft

- 7) Optionally, you can change the text orientation on the TFT-LCD. To do so, double-click the “sensorian-tft” node. Set the “Orientation” to “Landscape”. Click “Ok”.

Edit sensorian-tft node

Name

Orientation

Ok Cancel

Figure 20: Configure the orientation of the TFT-LCD

- 8) Deploy the program and then write something to the LCD. Open a new tab in your browser and go to “<IP-ADDRESS-OF-RASPBERRY-PI>:1880/lcd_write?msg=MYMESSAGE” replacing MYMESSAGE with your message. The message will be printed on the TFT-LCD.

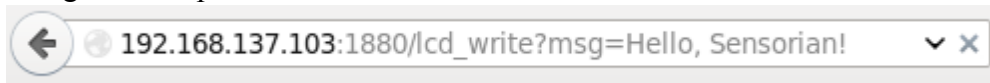


Figure 21: Writing to the TFT-LCD through the browser

Project 5: Using Advanced Web APIs

Project idea: Node-RED can integrate the Raspberry Pi and Sensorian with a variety of web-based services. This example will use the “http request” node to get weather data from Environment Canada and process it so that the current temperature is displayed on the Sensorian TFT-LCD.

To get started:

- 1) Click and drag an “inject” node onto the worksheet. Double-click the node and set its “Payload” to “blank”. Set “Repeat” to “interval”, every “5” “minutes”. Set the “Name” to “PollWeather”. This will update the current temperature information on the LCD every 5 minutes.



Figure 22: Create the PollWeather inject node

Edit inject node

✉ Payload

☰ Topic

🔄 Repeat

every minutes

Fire once at start ?

🔑 Name

Note: "interval between times" and "at a specific time" will use cron. See info box for details.

Figure 23: Configure the PollWeather inject node

- 2) We now need to find the URL for the city which we wish to get the weather for. Open a new tab in your browser and go to, http://dd.weatheroffice.ec.gc.ca/citypage_weather/xml/siteList.xml , choose a city, and make note of the “site code”. This example will get the current temperature in Toronto, ON so the “site code” is “s0000458”.

```

</site>
- <site code="s0000458">
  <nameEn>Toronto</nameEn>
  <nameFr>Toronto</nameFr>
  <provinceCode>ON</provinceCode>
</site>

```

Figure 24: Finding the site code for a Canadian city

- 3) Return to Node-RED and click and drag a “http request” node into the workspace. Double-click the node and set its “Method” to “GET”. Set its URL to http://dd.weatheroffice.ec.gc.ca/citypage_weather/xml/<PROVINCE>/<SITECODE>.e.xml replacing <PROVINCE> with the “province code” of your city and <SITECODE> with the “site code” from the previous step. Set “Return” to “a UTF-8 string”. Set the

“Name” to “InternetWeather” and click “Ok”. Wire the left side of “InternetWeather” to “PollWeather”.

The screenshot shows a dialog box titled "Edit http request node". It contains the following fields and options:

- Method:** A dropdown menu with "GET" selected.
- URL:** A text input field containing "http://dd.weatheroffice.gc.ca/citypage_weather/xr".
- Use basic authentication ?** An unchecked checkbox.
- Return:** A dropdown menu with "a UTF-8 string" selected.
- Name:** A text input field containing "InternetWeather".

At the bottom right of the dialog are "Ok" and "Cancel" buttons.

Figure 25: Configuring the http request node for your chosen city

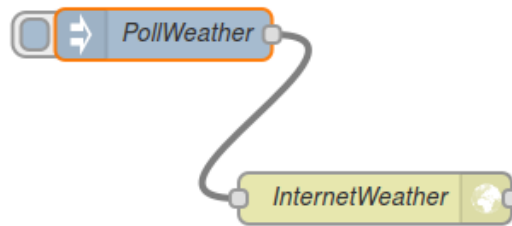


Figure 26: Wiring the left side of InternetWeather to PollWeather

- 4) Click and drag a “function” node into the workspace. Double-click the “function” node and set the “Name” to “GetCurrentTemperature”. Set the “Function” to:

```
var CC_TAG = "<currentConditions";
var CC_END_TAG = "</currentConditions";
var TEMP_TAG = "<temperature";
var TEMP_END_TAG = "</temperature";
var sample_xml = msg.payload
var c_begin = sample_xml.indexOf(CC_TAG)+1
sample_xml = sample_xml.slice(c_begin);
var c_end = sample_xml.indexOf(CC_END_TAG);
var t_begin = sample_xml.indexOf(TEMP_TAG);
if (t_begin > c_end)
{
    var new_msg = {payload:"NOT FOUND!"};
    return new_msg;
}
sample_xml = sample_xml.slice(t_begin);
var t_begin_close = sample_xml.indexOf(">");
sample_xml = sample_xml.slice(t_begin_close);
var t_end = sample_xml.indexOf(TEMP_END_TAG);
var new_msg = {payload:sample_xml.slice(1,t_end)};
```

```
return new_msg;
```

Then click “Ok”. Wire the left side of “GetCurrentTemperature” to the right side of “InternetWeather”.

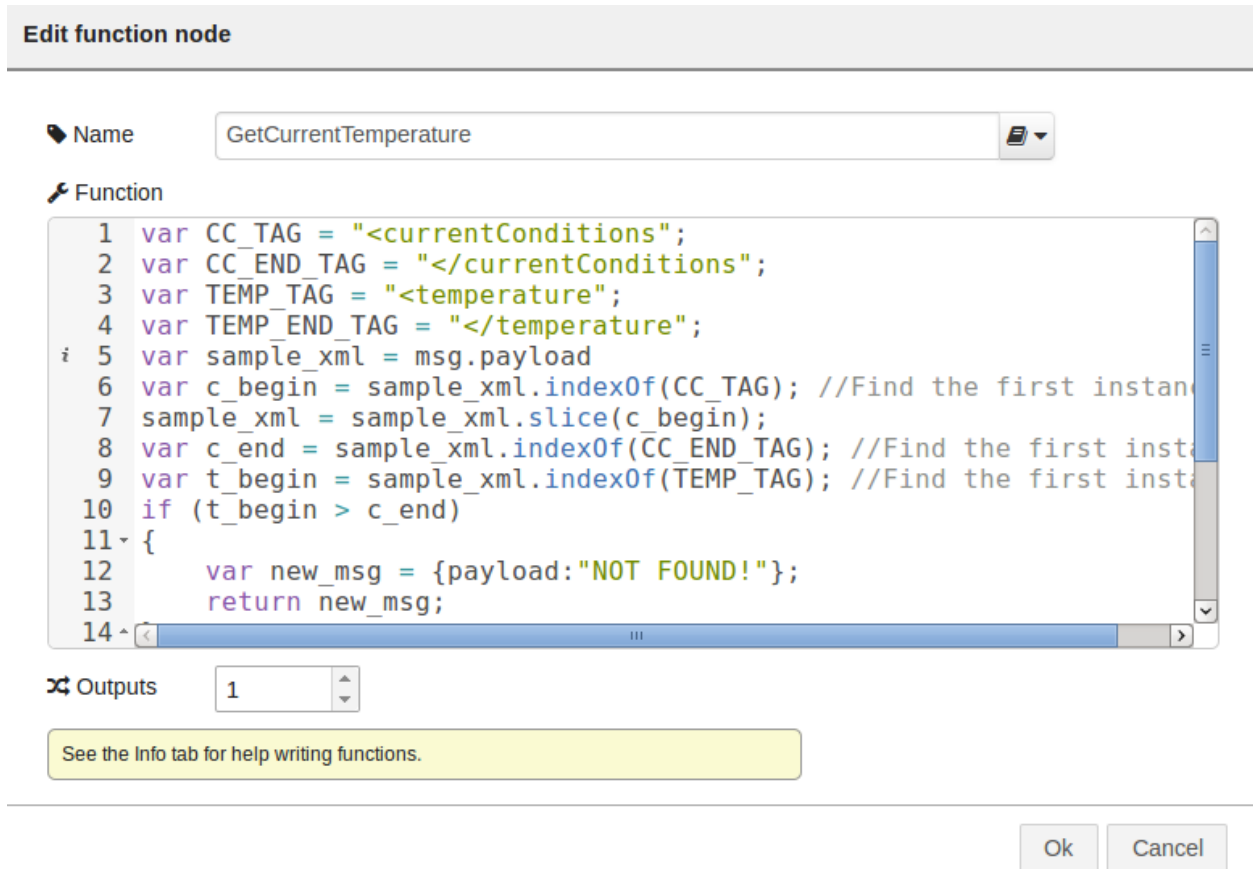


Figure 27: Programming the function node

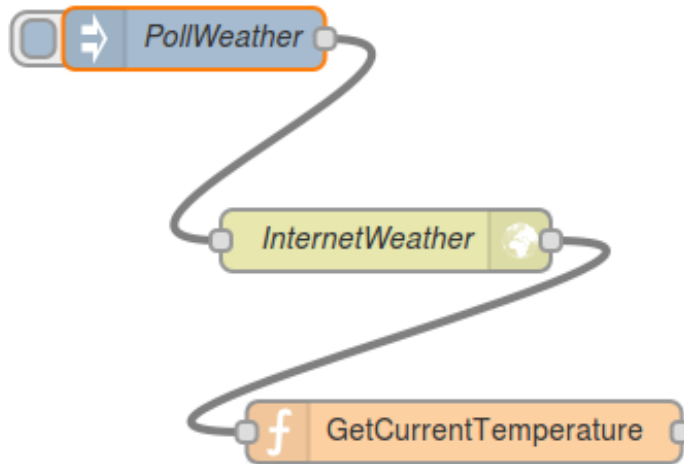


Figure 28: Wiring the left side of GetCurrentTemperature to the right side of InternetWeather

- 5) Click and drag a “template” node into the workspace. Double-click the node and set its “Name” to “TempLabel”. Set the template type to “mustache”. Set the “Template” code to:

<CITY>, <PROVINCE> - Current Temp: {{payload}}

replacing <CITY> with your chosen city and <PROVINCE> with its province. Set “Property” to “msg.payload” and click “Ok”. Wire the left side of “TempLabel” to the right side of “GetCurrentTemperature”.

Edit template node

Name

Template

```
1 Toronto, ON - Current Temp:{{payload}}
```

Property msg.

Ok Cancel

Figure 29: Configuring the template node

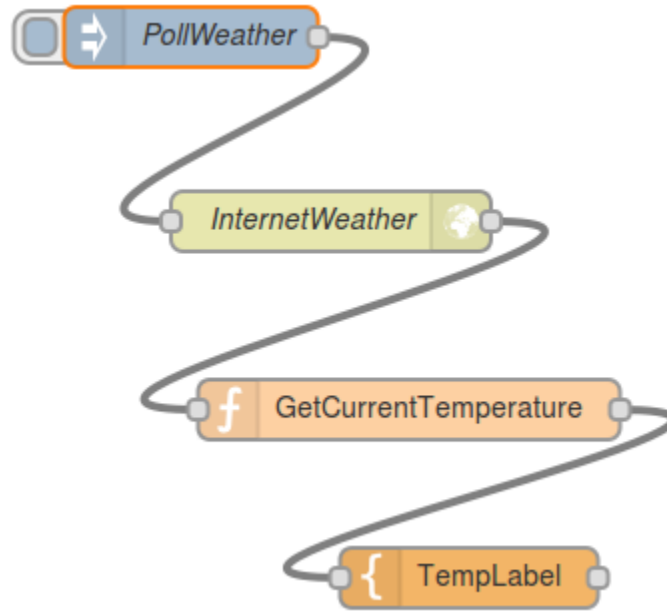


Figure 30: Wire the left side of TempLabel to the right side of GetCurrentTemperature

- 6) Click and drag a “sensorian-tft” node into the workspace. Double-click the node and set the “Orientation” to “Landscape”. Click “Ok” and wire it to the right side of “TempLabel”.

Edit sensorian-tft node

✎ Name

✎ Orientation

Ok Cancel

Figure 31: Set the orientation of the TFT-LCD to Landscape

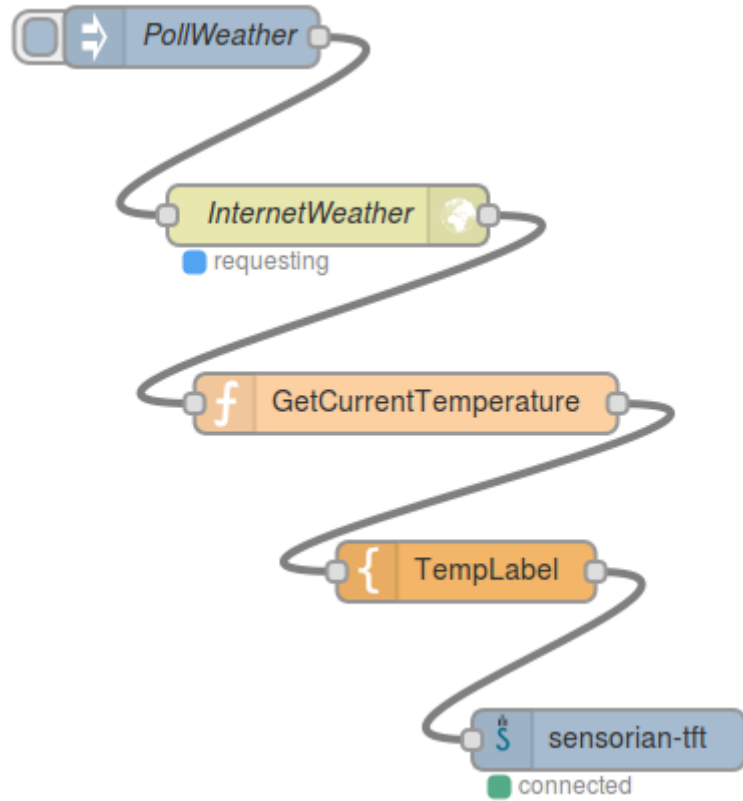


Figure 32: Wire the tft node to the right side of TempLabel

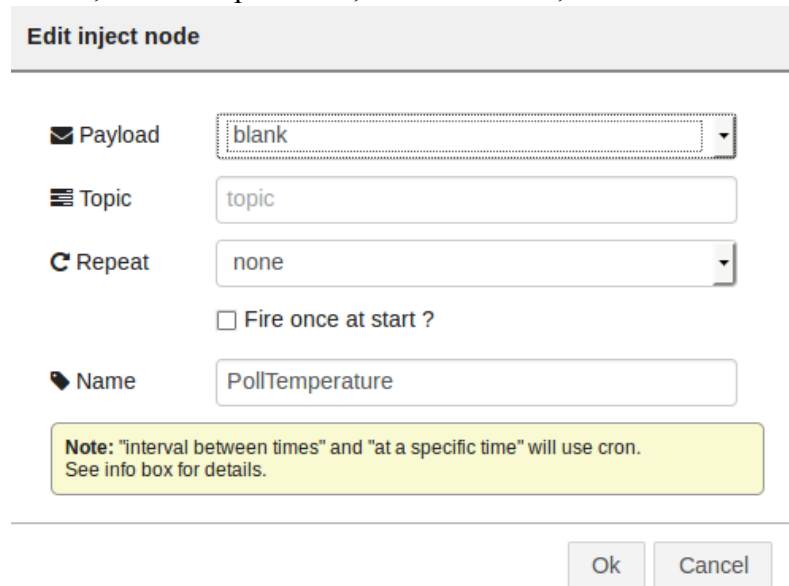
- 7) Deploy the program and trigger “PollWeather”. The current temperature should be updated every 5 minutes on the Sensorian TFT-LCD.

Project 6: Environmental Data Server

Project idea: Node-RED can receive HTTP requests and issue HTTP responses. This allows a developer to create a basic HTTP web server. This example will create a dynamic web page which will display the sensor values of “Temperature”, “Barometric Pressure”, “Altitude”, and “Ambient Light” based on the last time they were polled. This data can be read by anyone who points their browser to <IP-ADDRESS-OF-RASPBERRY-PI>:1880/get_sensors.

To get started:

- 1) Drag 4 “inject” nodes into the workspace. Double-click them and set the “Payload” to “blank”. Name them, “PollTemperature”, “PollPressure”, “PollAltitude” and “PollLight”.



Edit inject node

✉ Payload: blank

☰ Topic: topic

🔄 Repeat: none

Fire once at start ?

📌 Name: PollTemperature

Note: "interval between times" and "at a specific time" will use cron. See info box for details.

Ok Cancel

Figure 33: Name all four inject nodes and configure their names and payloads

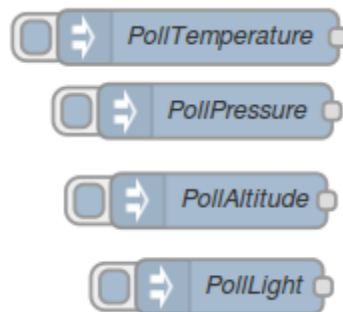


Figure 34: All four inject nodes used for polling have been created

- 2) Drag a “sensorian-temperature” node into the workspace and wire its left side to “PollTemperature”. Drag a “sensorian-pressure” node into the workspace and wire its left side to “PollPressure”. Drag a “sensorian-altitude” node into the workspace and wire its left side to “PollAltitude”. Drag a “sensorian-ambientlight” node into the workspace and wire its left side to “PollLight”.

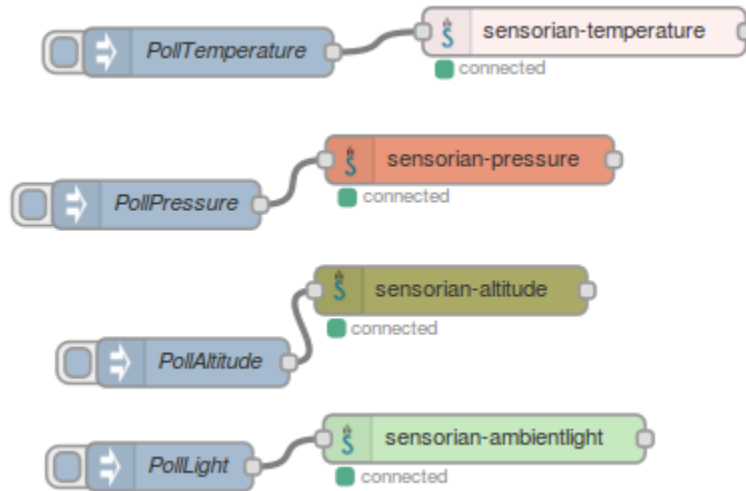


Figure 35: The Sensorian sensor nodes have been connected to their poll injectors

- 3) Drag 4 “function” nodes into the workspace. Connect the left side of one to the right side of “sensorian-temperature”. Double-click it and set its “Name” to “TemperatureTagger”. Set the “Function” to:

```
msg.topic = "temperature";  
return msg;
```

Repeat the same process connecting a “function” node to “sensorian-pressure”. Set its “Name” to “PressureTagger” and its “Function” to:

```
msg.topic = "pressure";  
return msg;
```

Repeat the same process connecting a “function” node to “sensorian-altitude”. Set its “Name” to “AltitudeTagger” and its “Function” to:

```
msg.topic = "altitude";  
return msg;
```

Repeat the same process connecting a “function” node to “sensorian-ambientlight”. Set its “Name” to “LightTagger” and its “Function” to:

```
msg.topic = "ambientlight";  
return msg;
```

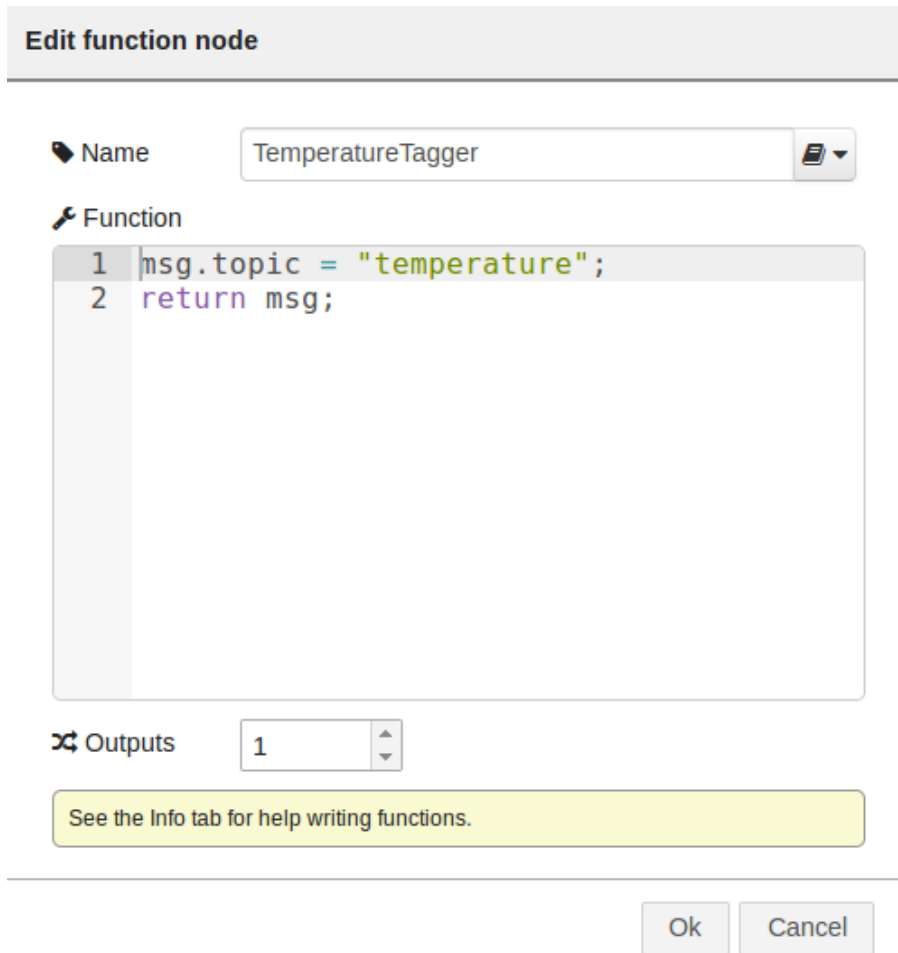


Figure 36: Programming TemperatureTagger. Procedure is similar for PressureTagger, AltitudeTagger, and LightTagger

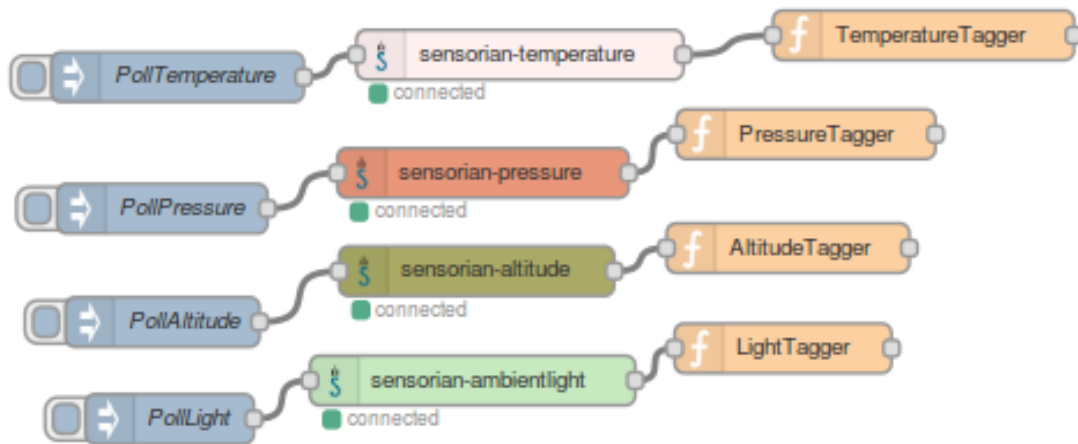


Figure 37: Taggers have been connected to their respective Sensorian sensor nodes

- 4) Drag a “function” node into the workspace and wire the right side of “TemperatureTagger”, “PressureTagger”, “AltitudeTagger”, and “LightTagger” into the

left side of this new “function” node. Double-click this node and set its “Name” to “DataBuffer”. Set its “Function” to:

```
context.data = context.data || new Object();
if ('topic' in msg)
{
    if (msg.topic == 'temperature')
    {
        context.data.temperature = msg.payload;
        msg = null;
    }
    else if (msg.topic == 'pressure')
    {
        context.data.pressure = msg.payload;
        msg = null;
    }
    else if (msg.topic == 'altitude')
    {
        context.data.altitude = msg.payload;
        msg = null;
    }
    else if (msg.topic == 'ambientlight')
    {
        context.data.ambientlight = msg.payload;
    }
    else
    {
        msg.data = context.data;
    }
}
else
{
    msg.data = context.data;
}
return msg;
```

Edit function node

Name:

Function

```

1 context.data = context.data || new Object();
2 if ('topic' in msg)
3 {
4     if (msg.topic == 'temperature')
5     {
6         context.data.temperature = msg.payload;
7         msg = null;
8     }
9     else if (msg.topic == 'pressure')
10    {
11        context.data.pressure = msg.payload;
12        msg = null;
13    }
14    else if (msg.topic == 'altitude')

```

Outputs:

See the Info tab for help writing functions.

Ok Cancel

Figure 38: Programming the DataBuffer function node

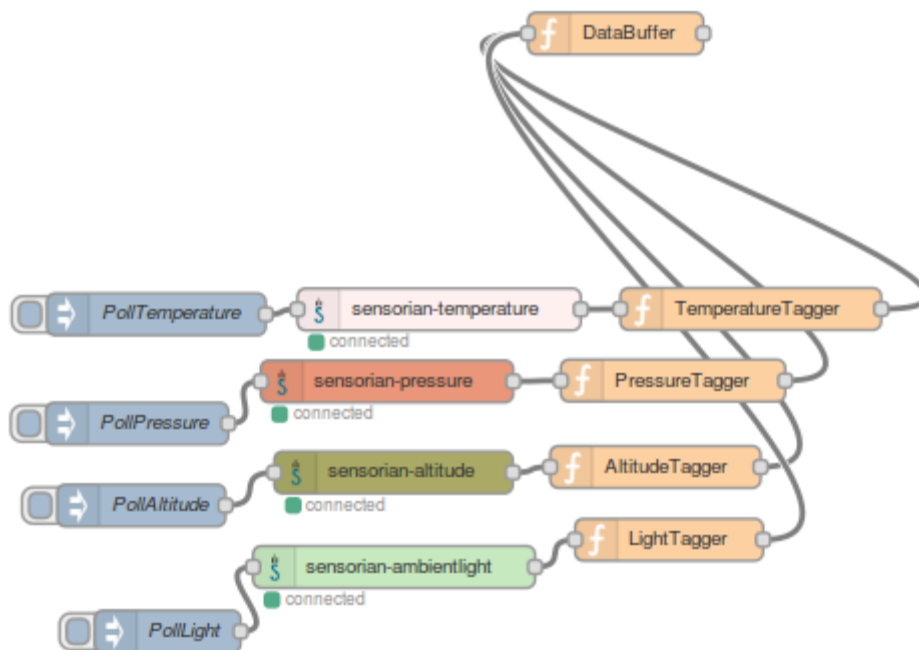


Figure 39: Wiring the DataBuffer function node to the Tagger function nodes

- 5) Drag a “http” node into the workspace and wire it to the left side of “DataBuffer”. Double-click it and set its “Method” to “GET” and its “url” to “/get_sensors”.

Edit http in node

Method	GET
url	/get_sensors
Name	Name

Ok Cancel

Figure 40: Configuring the http input node.

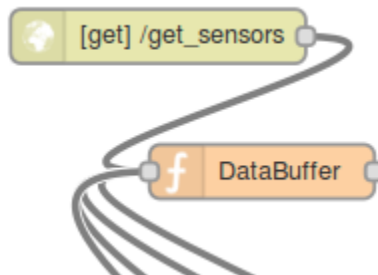


Figure 41: Wire the http node to the left side of DataBuffer

- 6) Drag a “template” node into the workspace and wire its left side to the right side of “DataBuffer”. Double-click it and set its “Name” to “PageTemplate”. Set its template type to “mustache”. Set its “Template” to:

```
Temperature: {{data.temperature}} </br>
Barometric Pressure: {{data.pressure}} </br>
Altitude: {{data.altitude}} </br>
Ambient Light: {{data.ambientlight}} </br>
```

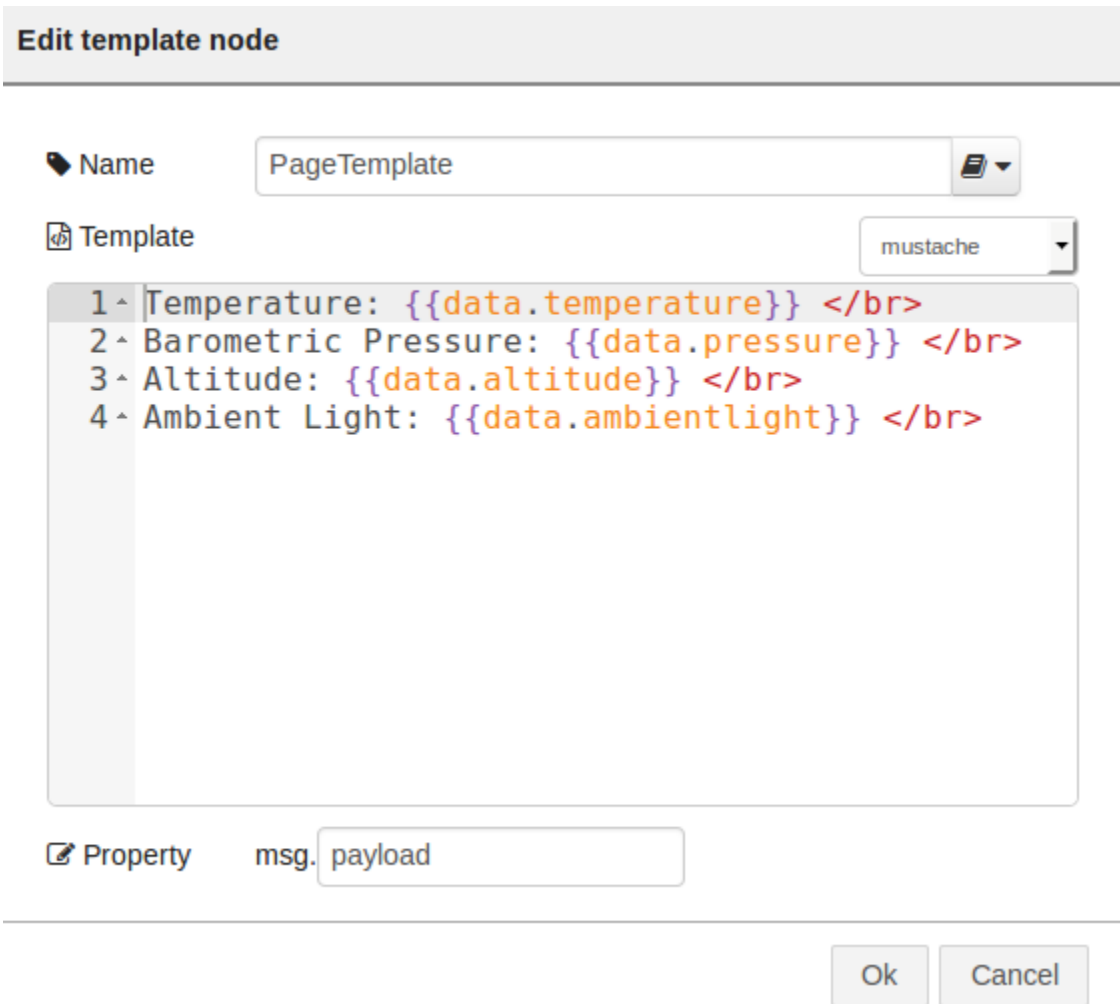


Figure 42: Configuring the PageTemplate template node

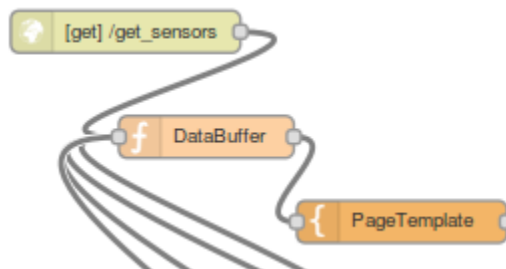


Figure 43: Wiring PageTemplate to DataBuffer

- 7) Drag a “http response” node into the workspace and wire its left side to the right side of “PageTemplate”.

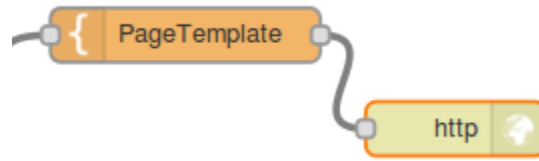


Figure 44: Adding a http response node and wiring it to PageTemplate

- 8) Deploy the program and trigger each “inject” node once. Open a new tab in the browser and go to <IP-ADDRESS-OF-RASPBERRY-PI>:1880/get_sensors. It will display the sensor readings at the time which they were polled.

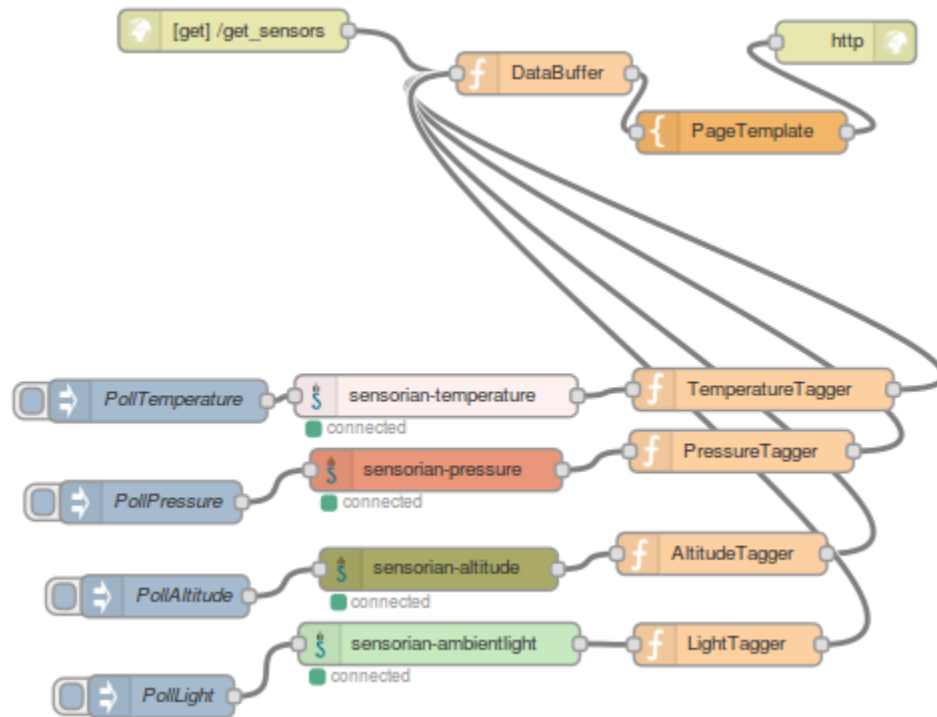


Figure 45: Worksheet for the final program

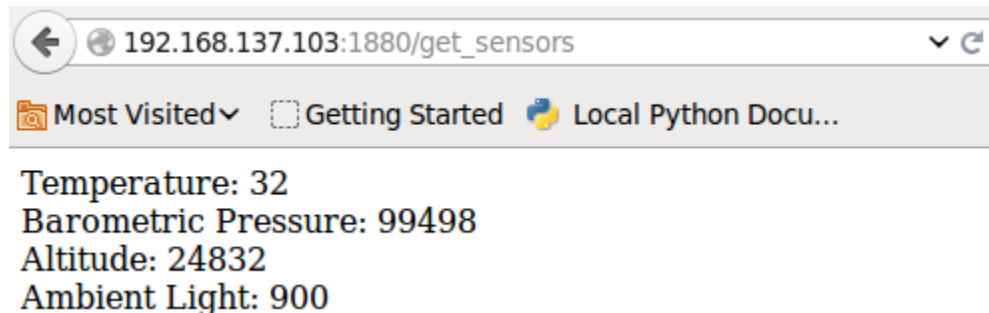


Figure 46: Connecting to Raspberry Pi + Sensorian sensors interface via browser