

|  |  |
| --- | --- |
| **Date of Submission:** | 22 Aug 2017 |
| **Prepared for:** | Dora |
| **Class:** | DIT/FT/3A/41  DBIT/FT/3B/22 |
| **Submitted by:** |  |
| **Student ID** | **Name** |
| 1514531 | Gabriella Sussman |
| 1529731 | Tan Yong Wee |
| 1551693 | Weng Zhi Wei |



**Table of Contents**

[**Section 1  
Overview of project**](#_gjdgxs) **2**

[**Section 2  
Hardware requirements**](#_30j0zll) **12**

[**Section 3**](#_l2hwku90yce7) **12**

[**Setup**](#_2et92p0) **12**

[**Section 4  
How to Setup IBM Bluemix**](#_tyjcwt) **13**

[**Section 5  
How to Setup the Light Control**](#_3dy6vkm) **14**

[**Section 6  
How to Setup the LDR**](#_kgc6bnj7uyw0) **21**

[**Section 7  
How to Setup the NCF card reader**](#_jzahiona4ipo) **23**

[**Section 8  
How to Setup the Chat**](#_ivsnsvwgg2ze) **25**

# Section 1 Overview of project

* 1. **Where we have uploaded our tutorial**

https://github.com/Po-Ta-To/iotca2noname

* 1. **Why have we chosen to upload to this site**

As github is a large community with a lot of passionate developers wanting to learn more about IOT. Hence why we chose to upload it to github as we want to gain feedback on our IOT solution and to also help others learn in the process.

* 1. **What have we uploaded**

Project Files Uploaded:

How to set-up the web interface:

Step 1: Install Watson IOT into your RPi

Open a Terminal window and install the following Node‐RED nodes on your RPi using this code:

sudo npm i -g node-red-contrib-ibm-watson-iot

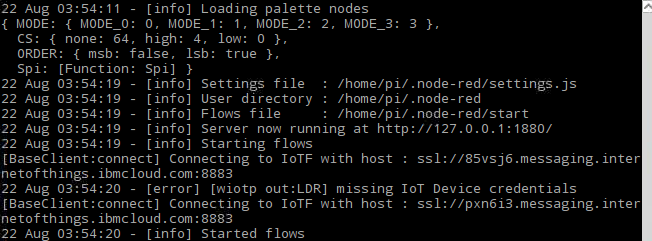
Once installation is successful, reboot your RPi using this code:

sudo reboot now

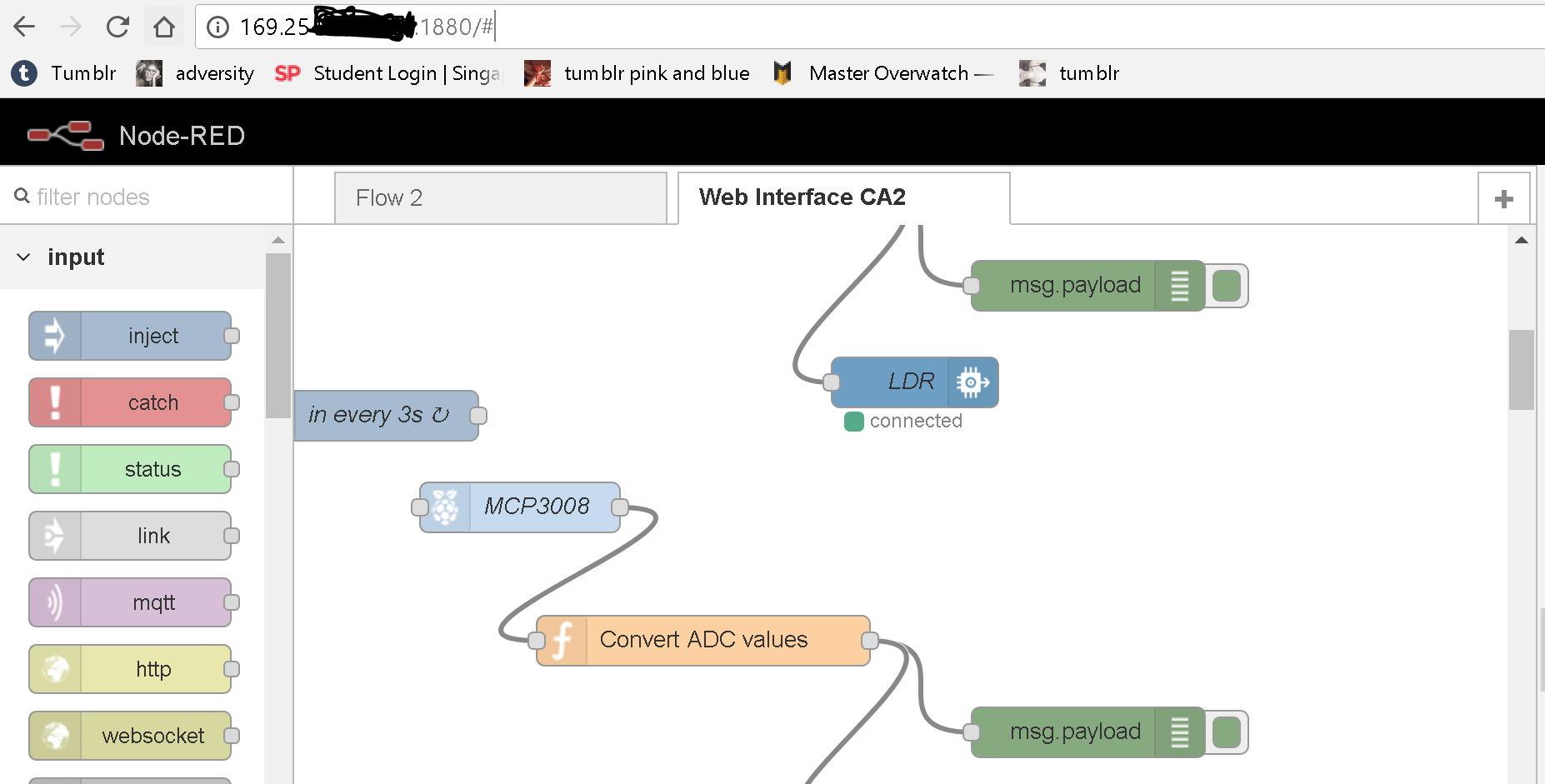
Open a Terminal window and start Node‐RED on your RPi

node‐red start

If it is successful, it will show a log similar to this in the terminal:

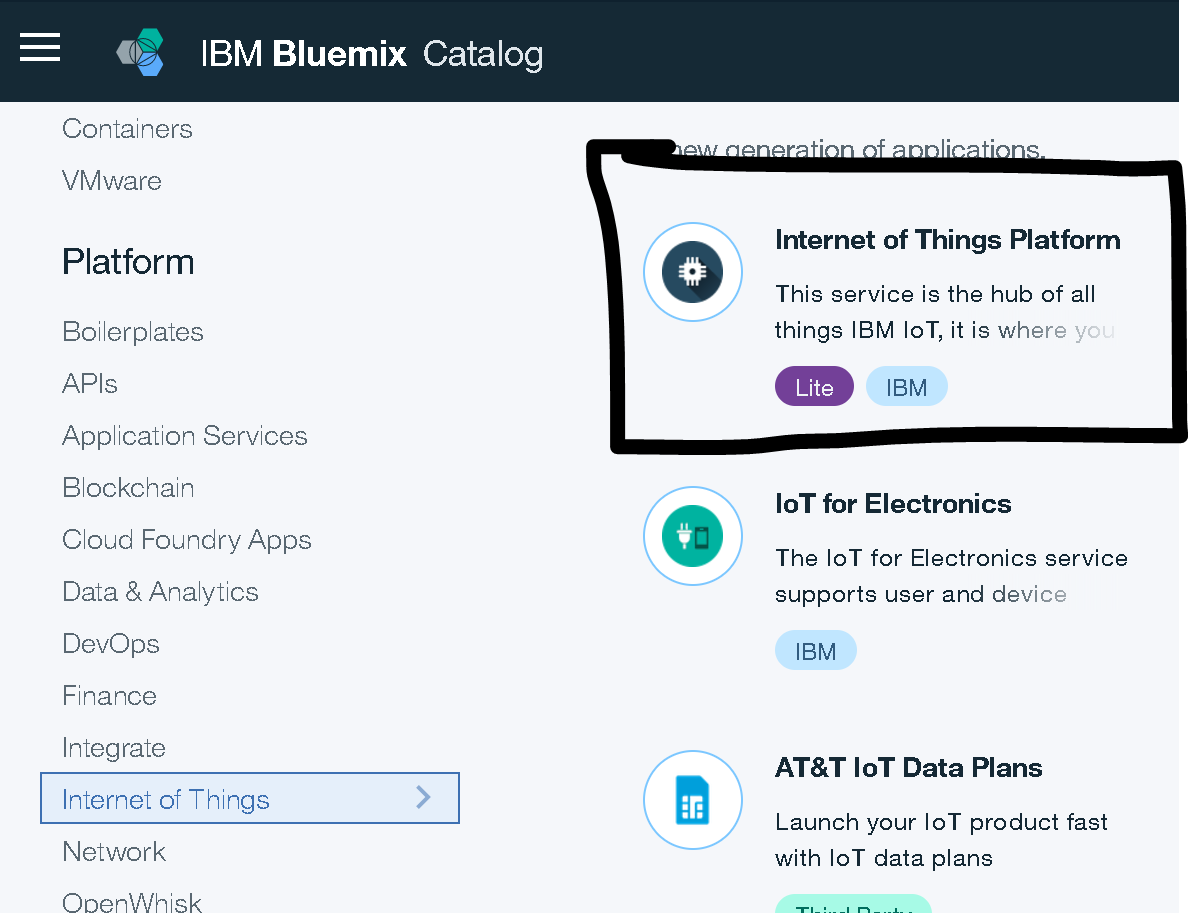


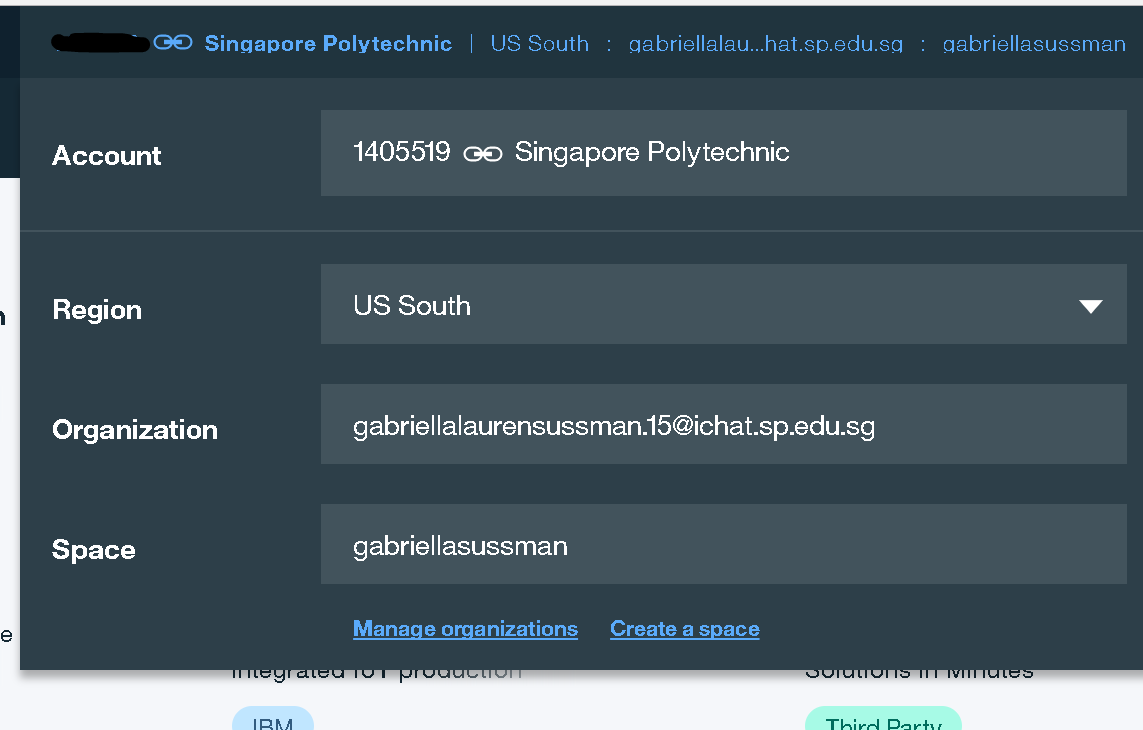
To access the node-red in your raspberry pi, enter <YOUR IP>:1880 and you will be shown this page:



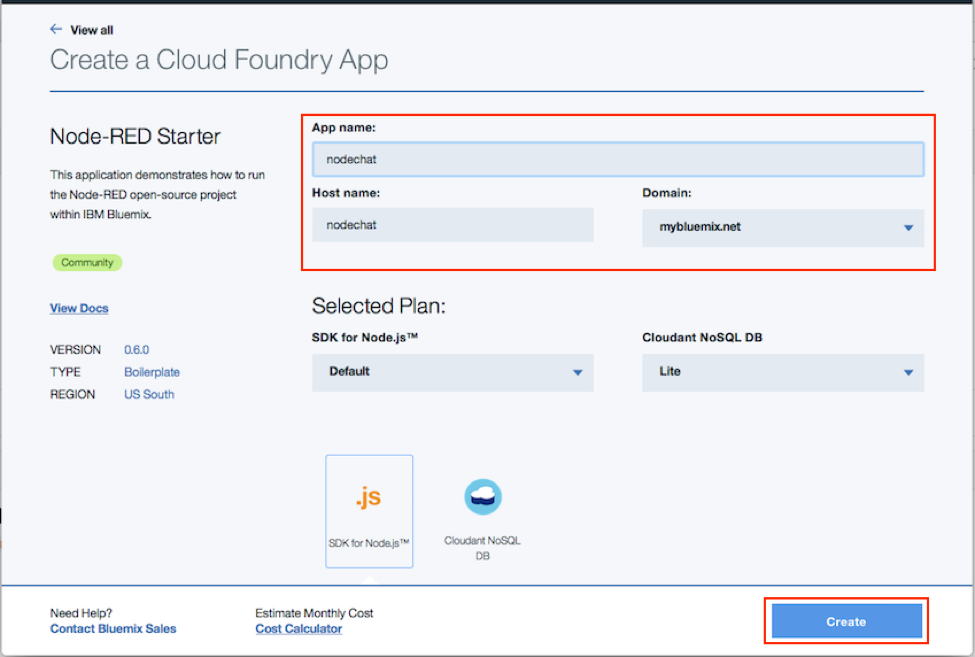
Step 2: Creating your IBM Watsons IOT Web App

Go to: <https://console.bluemix.net/catalog/> and navigate to the Internet of Things page and click on the Internet of Things Platform

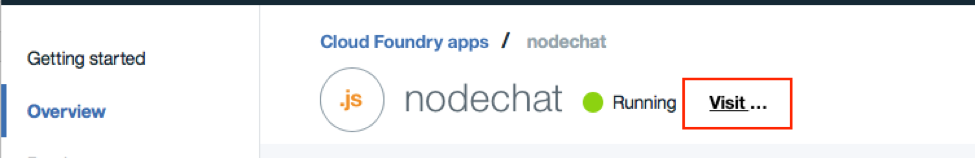


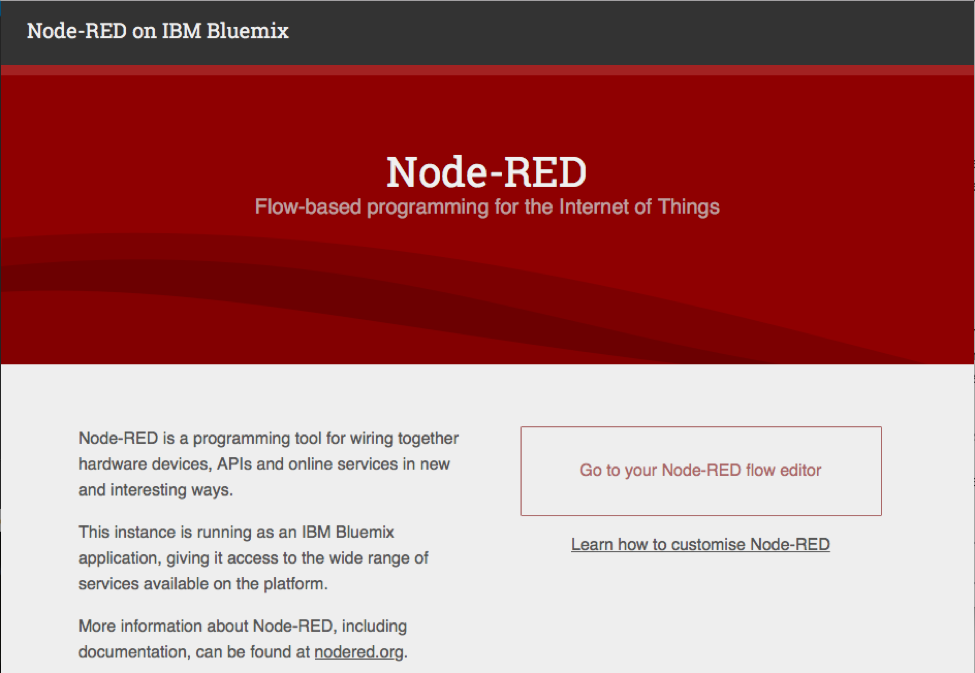


Create the app in a space you’ve created.

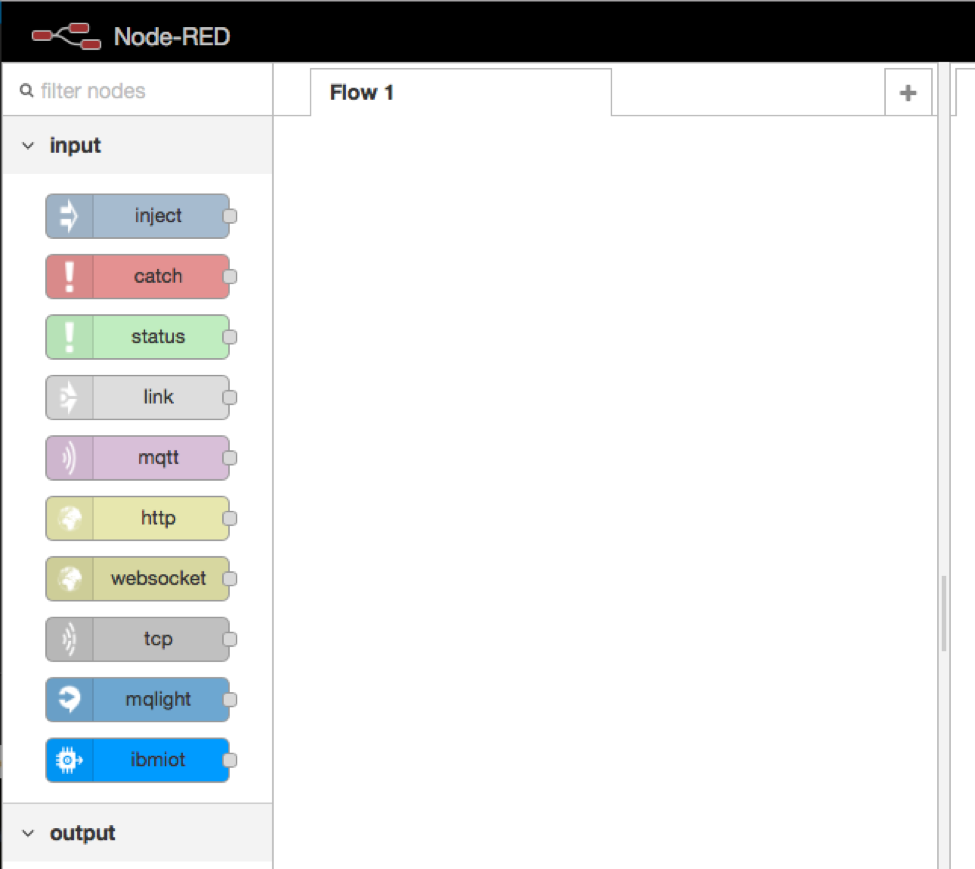


Create the foundary App, naming it whatever unique app name you want and click **Create.**

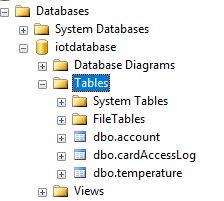
From the Bluemix dashboard, navigate to the Overview page for your application, then click the **Visit link** to launch **Node-RED's main page.** It can take a while for the application to start; the green circle and “**Running**” text will be visible when it’s ready to try.****

The first time you visit the site, you will be asked to do some basic configuration. Once that’s done, you can click Go to your Node-RED flow editor to open the flow editor.****

You should see a blank flow where you can start building your app. When using Node-RED we build our apps using this graphical editor interface to wire together the blocks we need. We can simply drag and drop the blocks from the left menu into the workspace in the center of the screen and connect them to create a new flow.



***DATABASE TABLES:***

There are 3 table you will need to create in your database:

Create table account(

cardId text not null,

userId text not null

)

Create table cardAccessLog(

id int not null identity(1,1),

cardId text not null,

classId text not null,

logDataTime datetime not null

)

Create table temperature(

id int not null identity(1,1),

datetime\_value datetime not null,

celsius text null,

fahrenheit text null

)

* 1. **What is the application about?**

Nowadays a lot of electricity is wasted due people leaving the light on when not necessary, especially in schools where the lights are constantly left on, even when students are not using the classrooms. In order to help reduce electricity wastage in singapore, we’ve decided to create a web interface where teachers can help reduce electricity wastage on lights by usings our features such as a on/off switch for the lights which can be accessed by them no matter where they are. The chat function can help teachers communicate with each other, for example, asking another teacher to turn off the aircon in a classroom if they might have forgotten to etc. Although our web interface doesn’t include a wide variety of features, we hope that our web interface will be upgraded with more features.

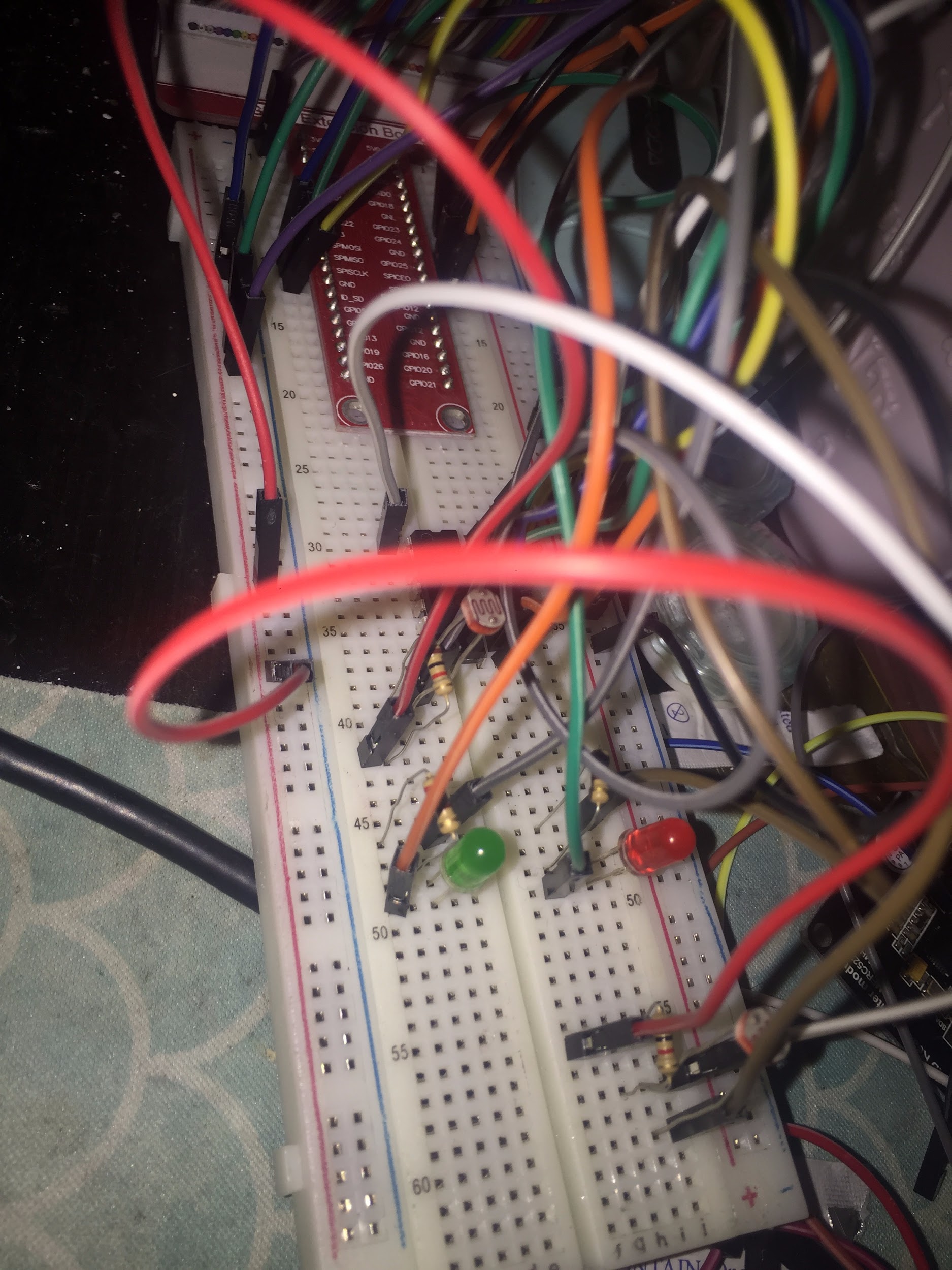
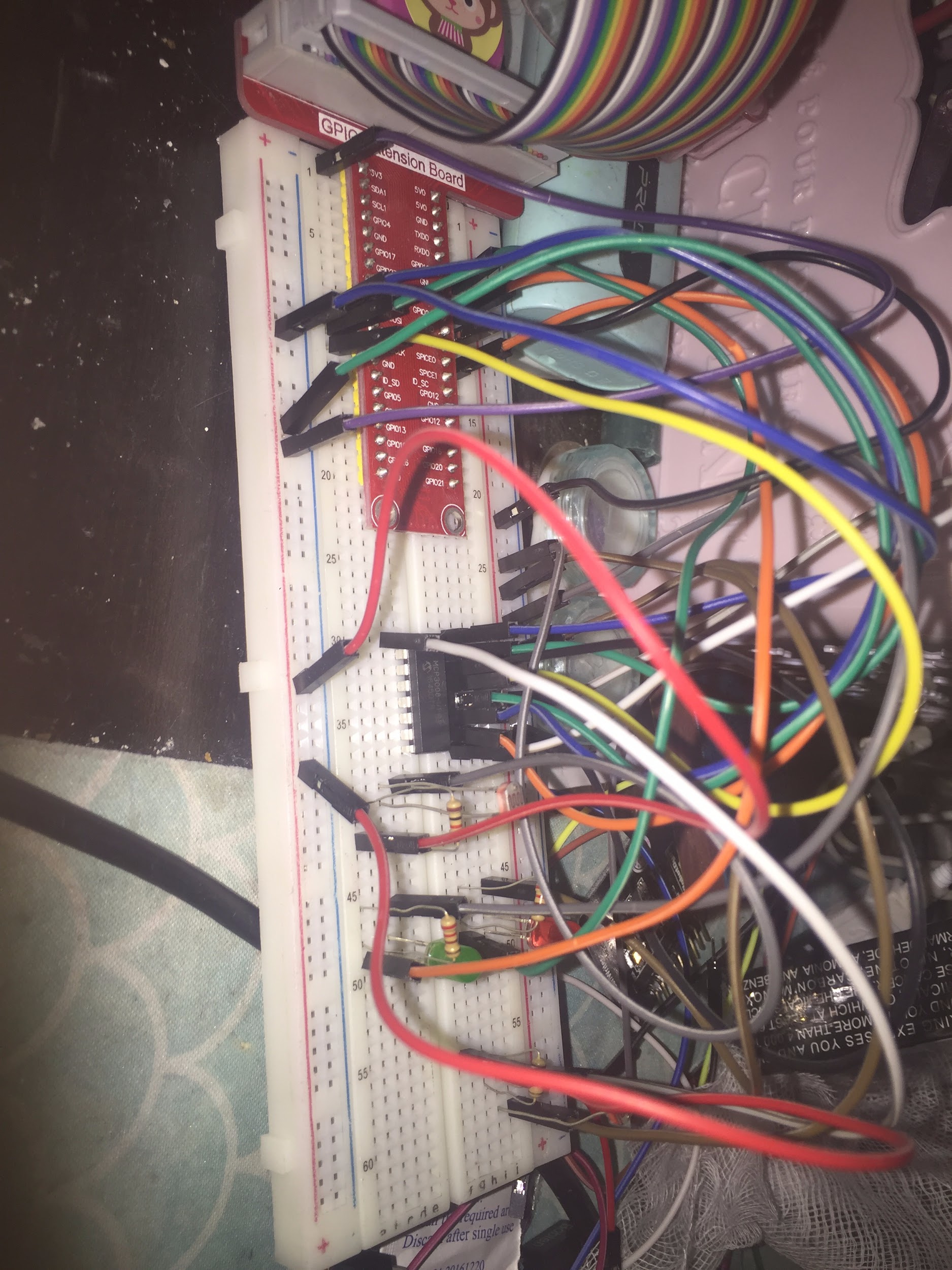
* 1. **Summary of the steps that will be described**

Provide a bullet list of the steps that will be covered in the other parts of this tutorial

|  |  |  |
| --- | --- | --- |
|  | Section | Description |
|  | Overview |  |
|  |  |  |
| Sections x to x provides the step-by-step instructions to set up the application | | |
|  | Hardware requirements | Provides overview of hardware required |
|  | Setup | Provides overview of how to setup the application |
|  | How to Setup IBM Bluemix | Provides overview of how to setup the application |
|  | How to Setup Light Control | Provides overview of how to setup the controlling of the lights |
| 6) | How to Setup LDR | Provides an overview of how to setup the LDR and collect the real time light levels |
| 7) | How to Setup the Database for NCF card reader | Provides an overview of how to setup the database and the NFC card reader |
| 8) | How to Set up Chat | Provides an overview of how to create the chat app |

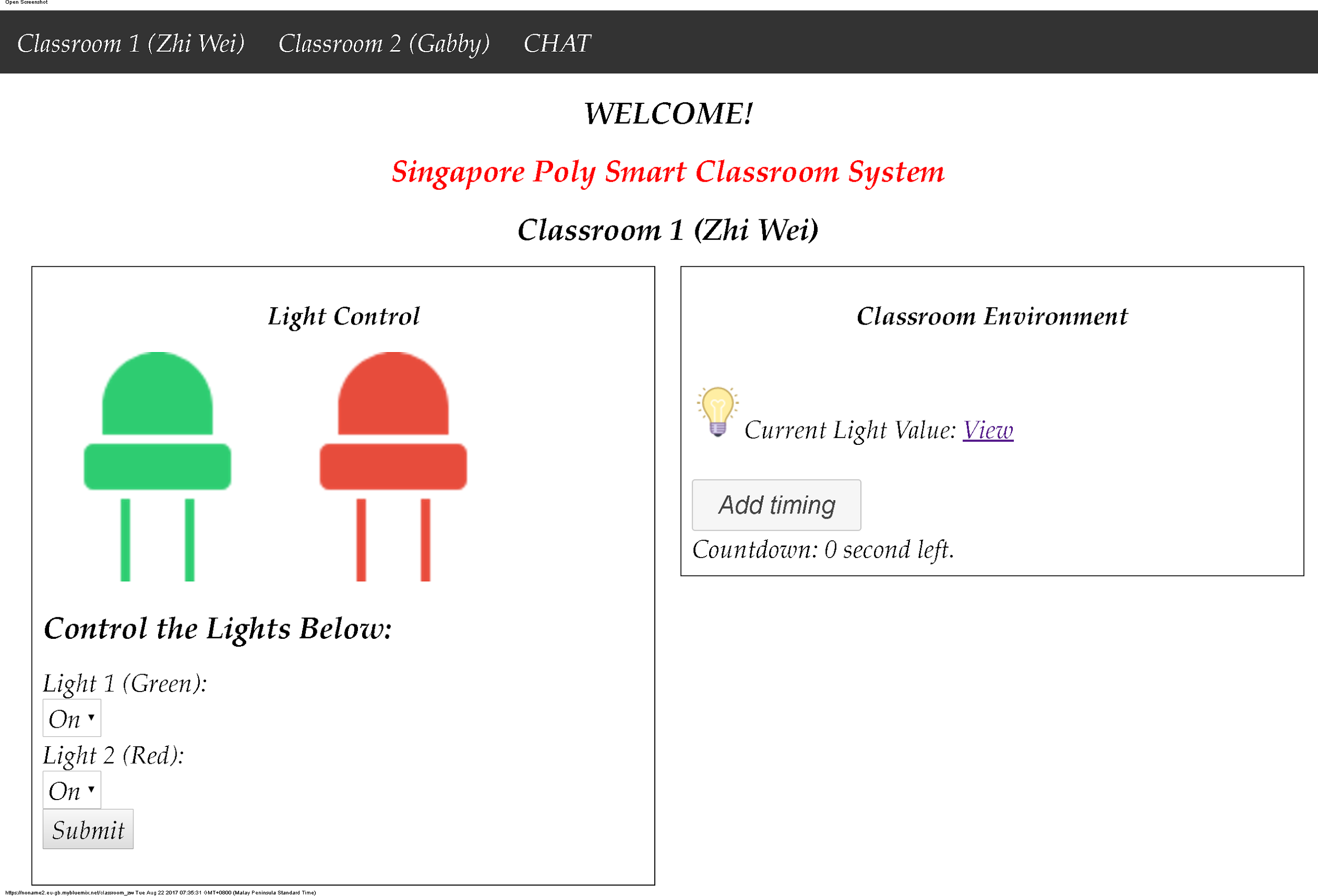
* 1. **How does the final RPI set-up looks like?**

Provide a photo of your final RPI set-up



(Gabby’s: Classroom 2)

* 1. **How does the web or mobile application look like?**

~~~~

Classroom webpages:

[http://noname2.eu-gb.mybluemix.net/classroom\_gabby](http://iotca2-nonamegroup.mybluemix.net/classroom_gabby)

<http://noname2.eu-gb.mybluemix.net/classroom_zw>

Chat Page:

[http://noname2.eu-gb.mybluemix.net/c](http://noname2.eu-gb.mybluemix.net/classroom_zw)hat

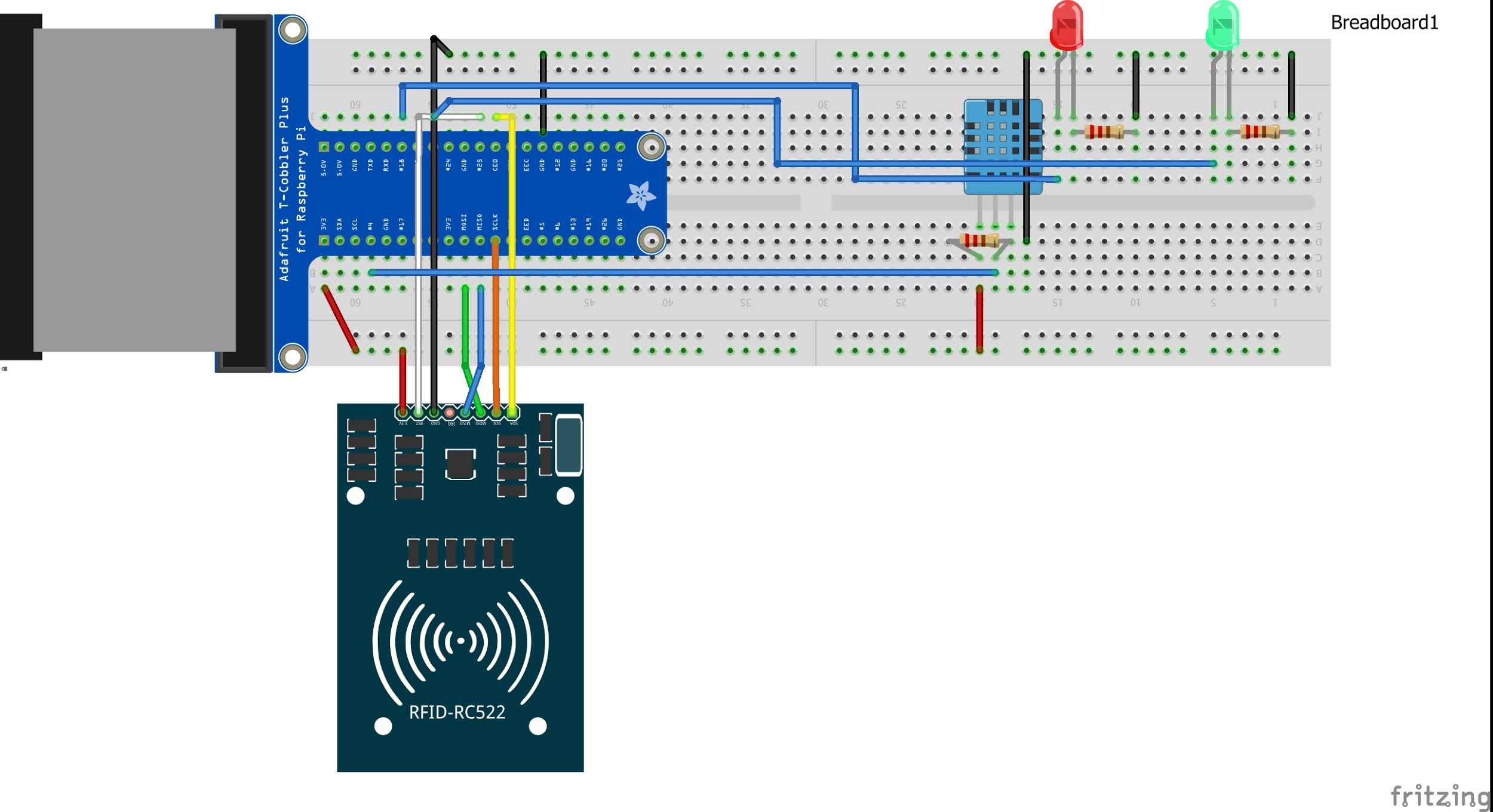
# Section 2 Hardware requirements

**Hardware checklist**

1. 2 x RaspBerryPi
2. 4 x LED
3. 1 x LDR
4. 2 x NCF cards
5. 1 x NCF card reader

# Section 3

# Setup



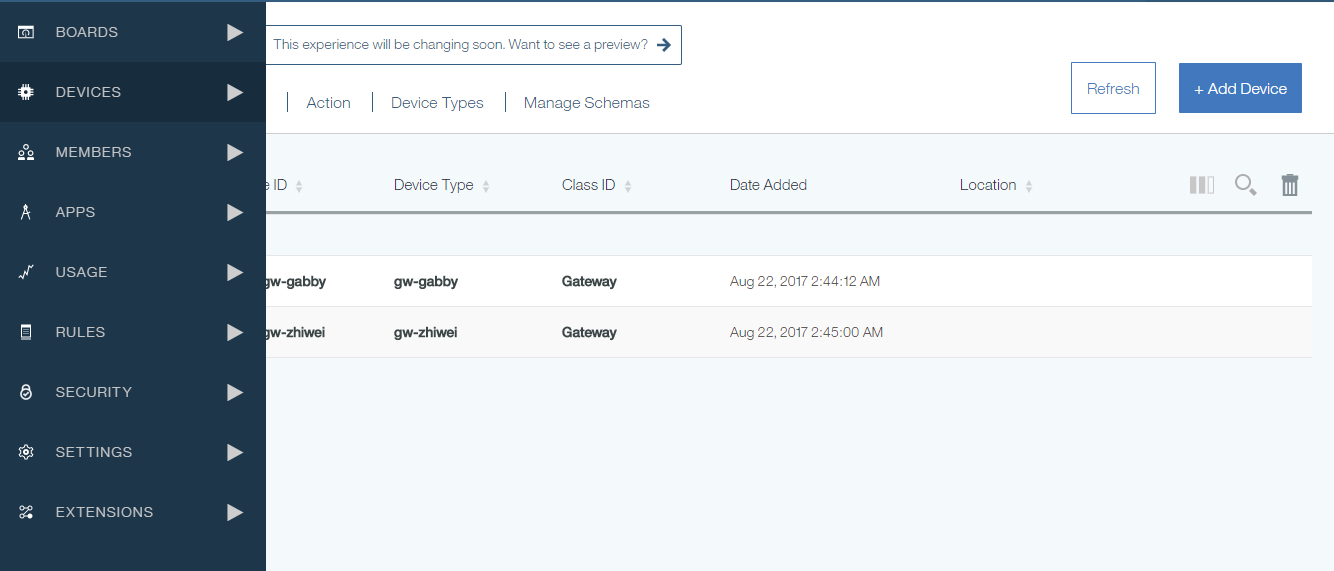
# Section 4 How to Setup IBM Bluemix

Org: pxn6i3

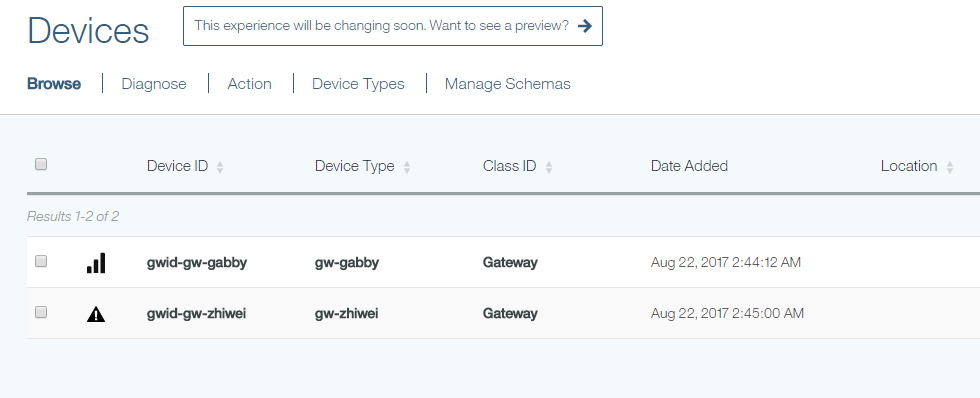
Url: <https://pxn6i3.internetofthings.ibmcloud.com>

Node-red flow for IBM Bluemix: <https://noname2.eu-gb.mybluemix.net/red/#>

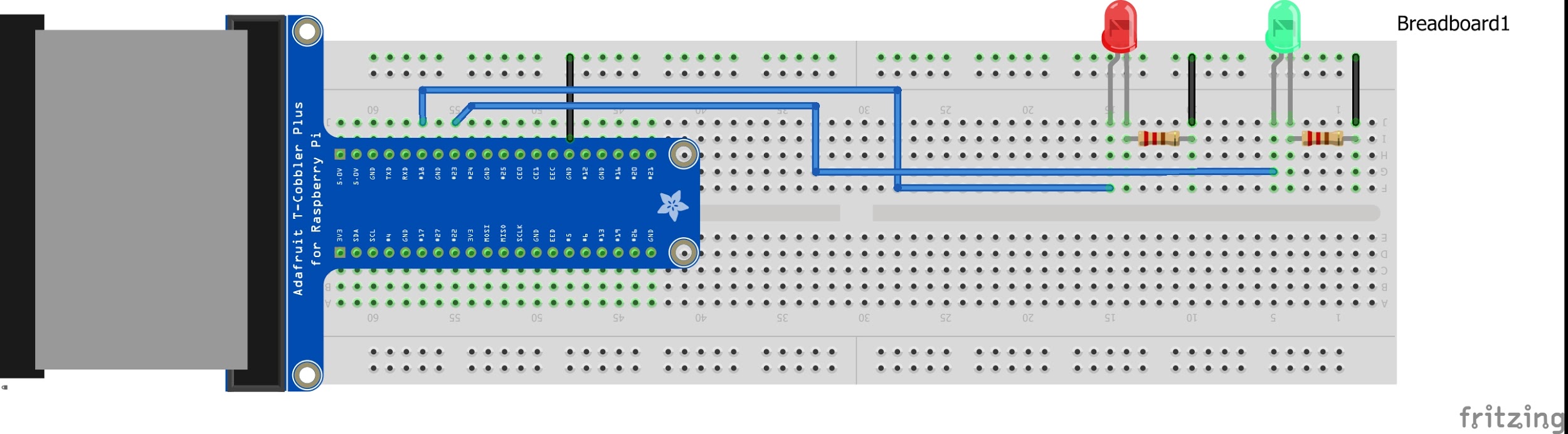
Step 1: In the bluemix IoT console, goto devices and click on add device. Then click on create device type and create a new device type.



Step 2: Then create devices according to the device ID as shown below, with auth token as: authtoken-gw-name

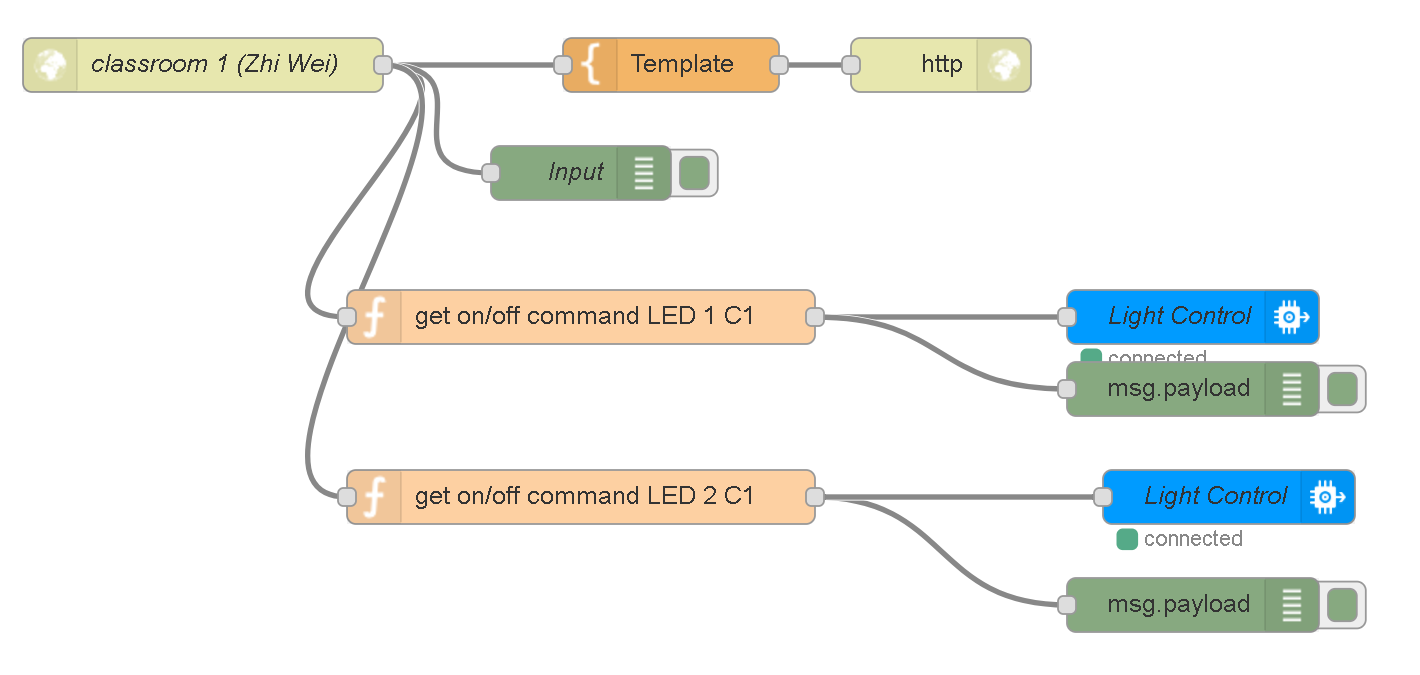


# Section 5 How to Setup the Light Control



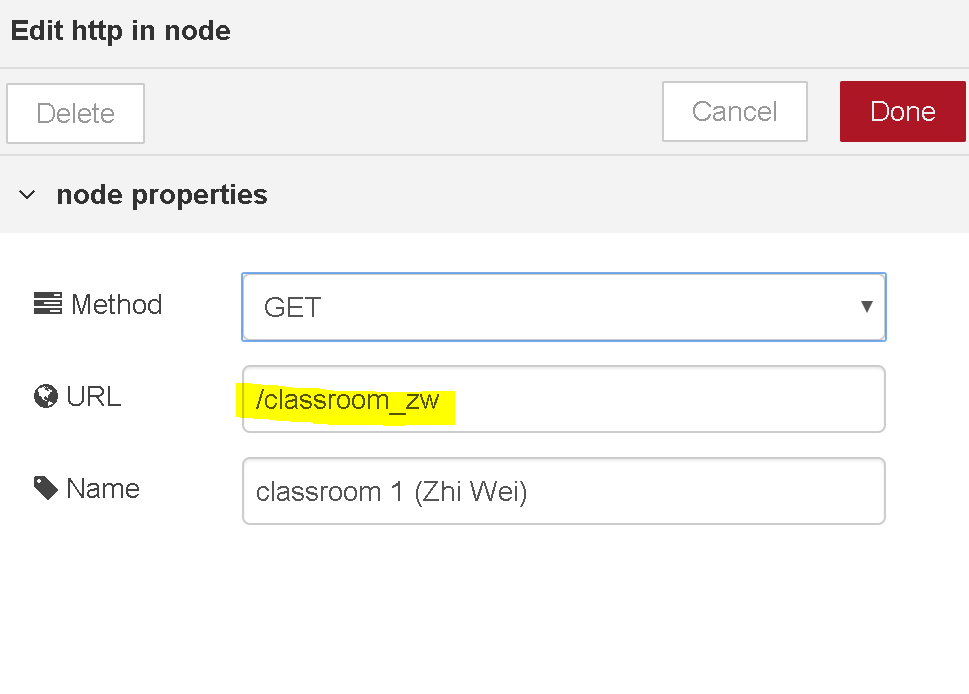
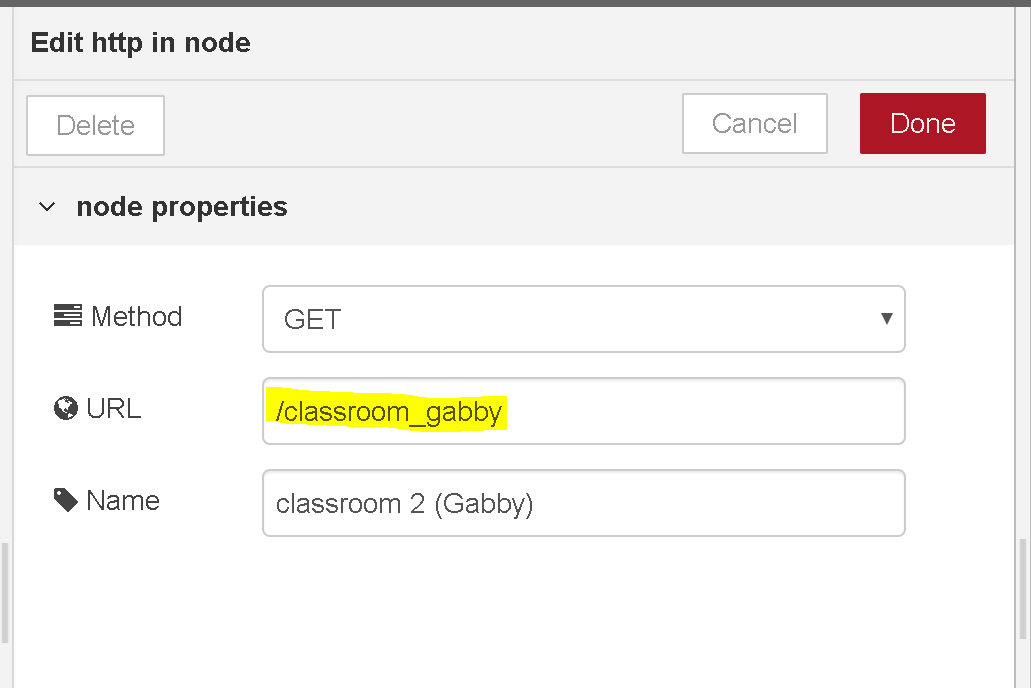
Step 1: Copy and Paste the flow from the file named “LightControl-IBM.text”:

An example of the result of the flow is shown below:

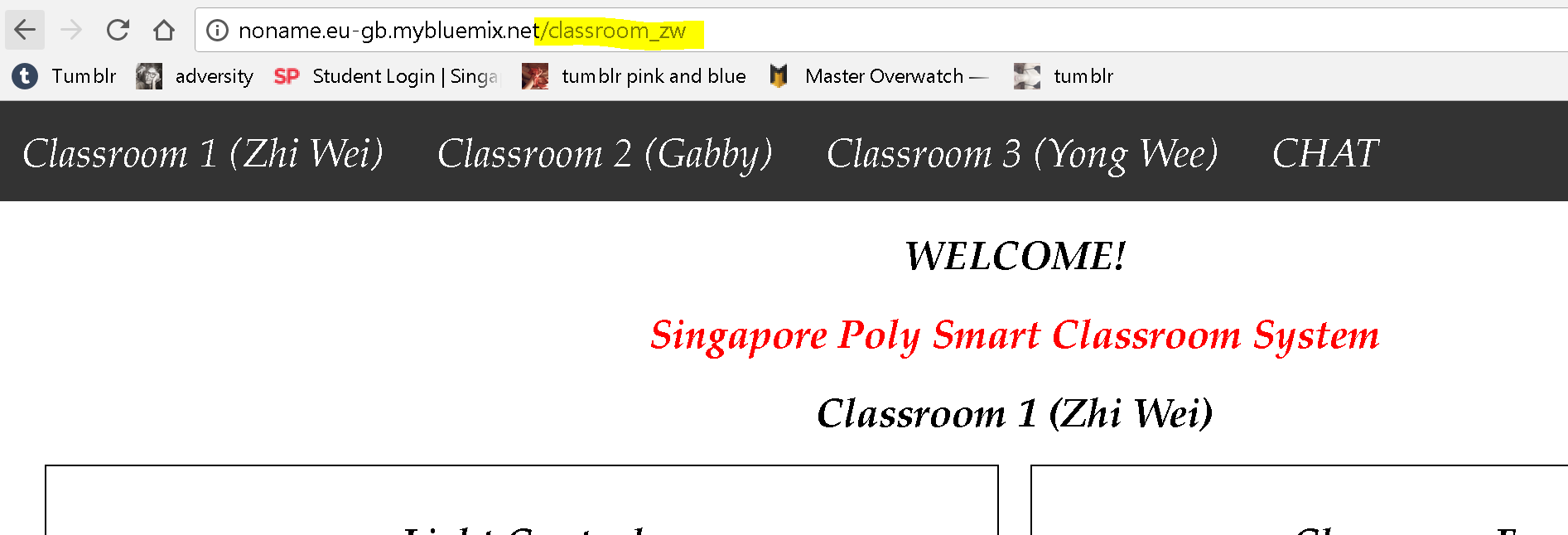


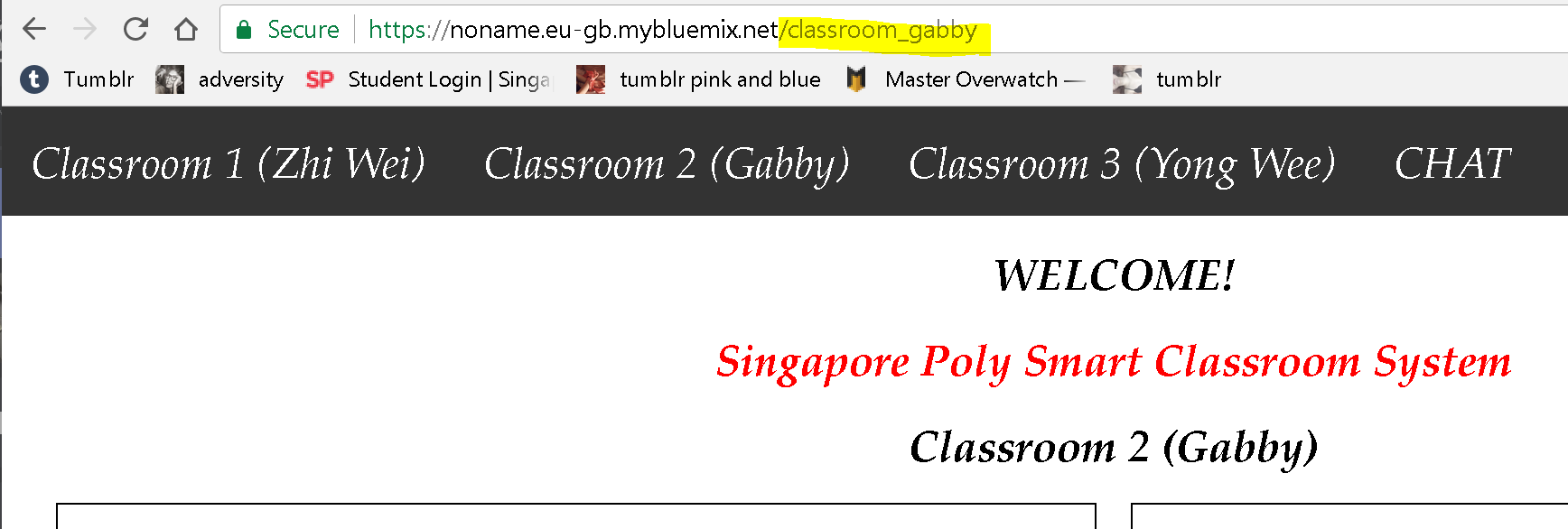
***DOCUMENTATION/EXPLANATION:***

*HTML Input Node (LOCALHOST Node-Red)*



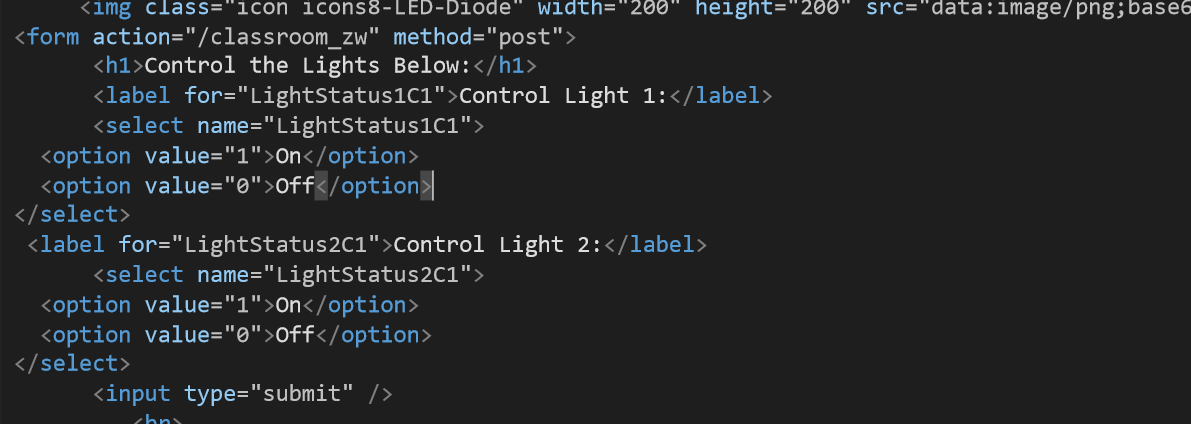
This will display each template in a different page as shown below.

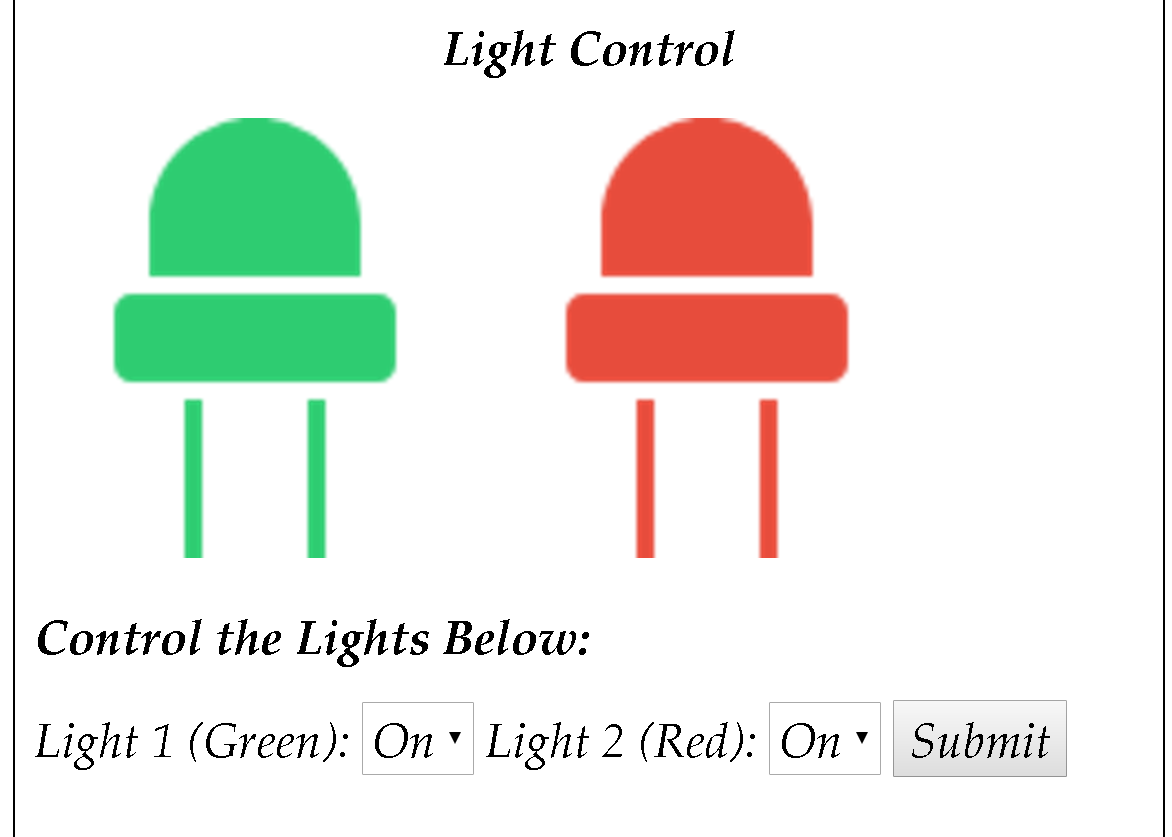


**

*Template Node (IBM Bluemix Node-Red)*

To control the status of the light, we will be using a form using the “POST” method AJAX request to inject the values into the URL. Below is what the form’s code looks like for classroom 1 (Zhi Wei)

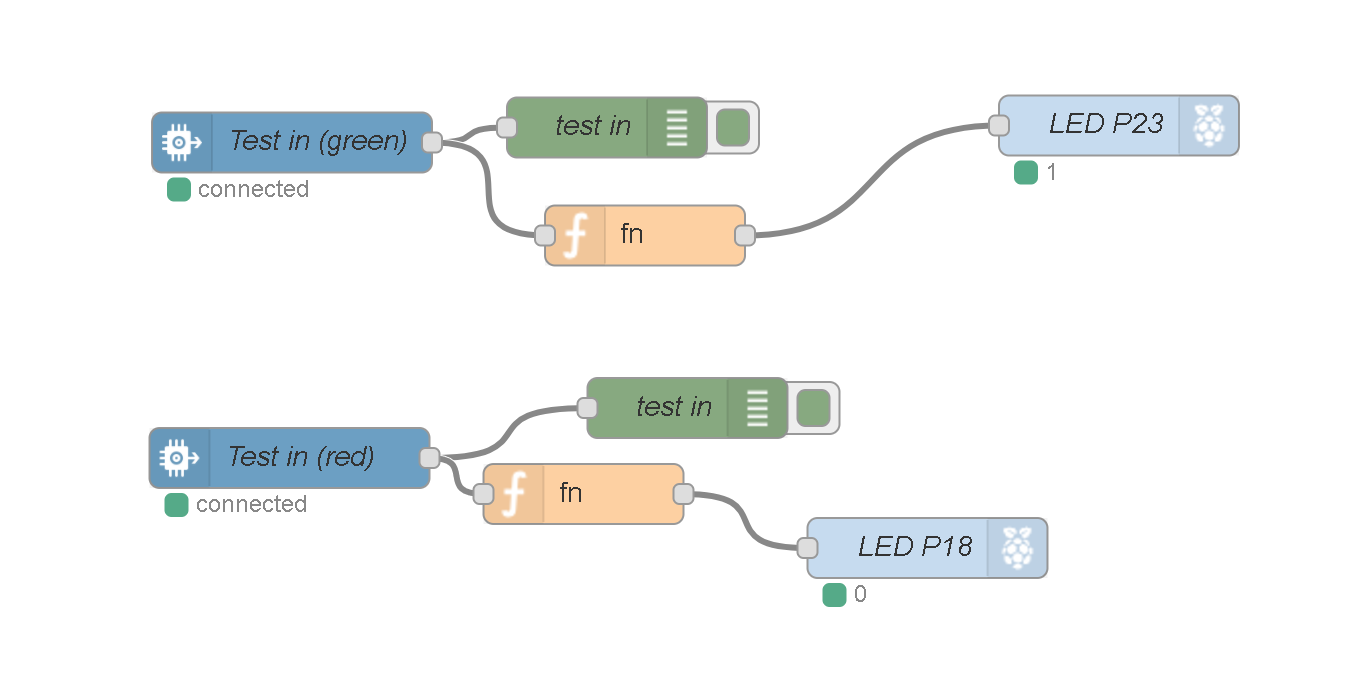




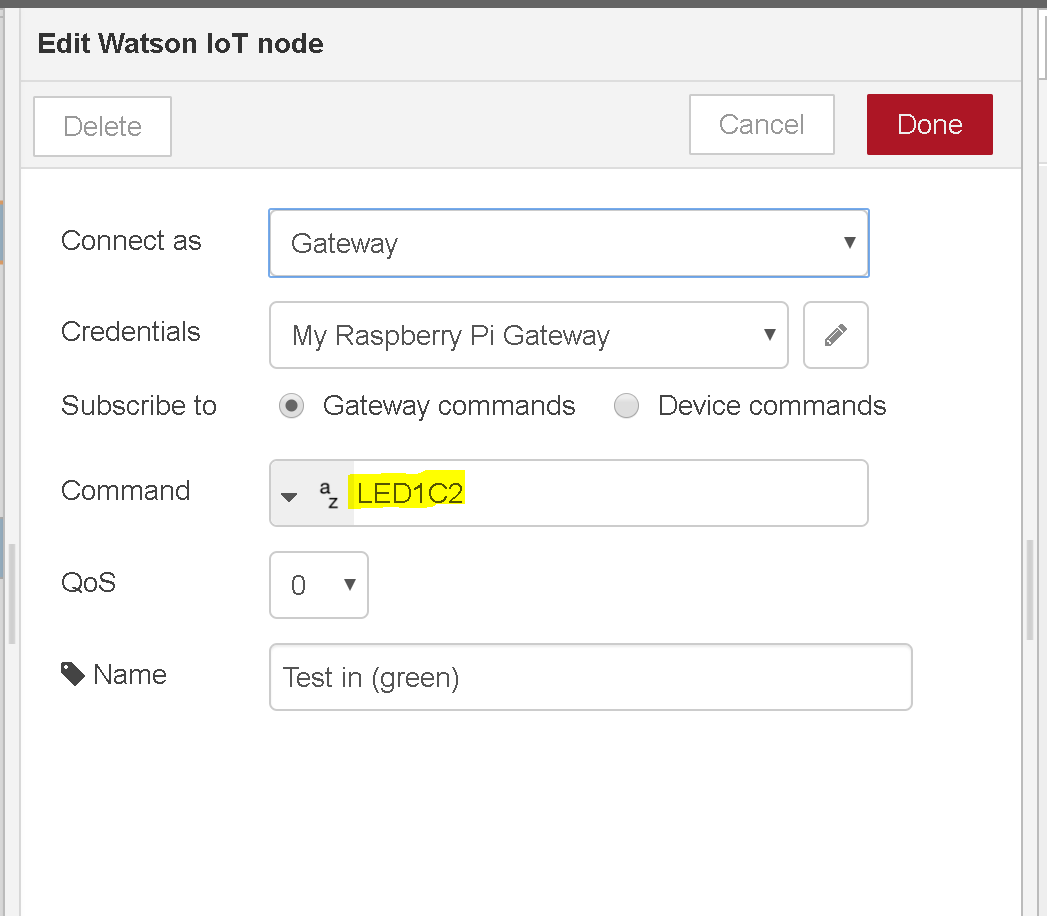
(The HTML output)

***WatsonIoT Input Node (LOCALHOST Node-Red)***

Copy and Paste the code from **LightControl-LH.txt** file into **Imports > Clipboard**

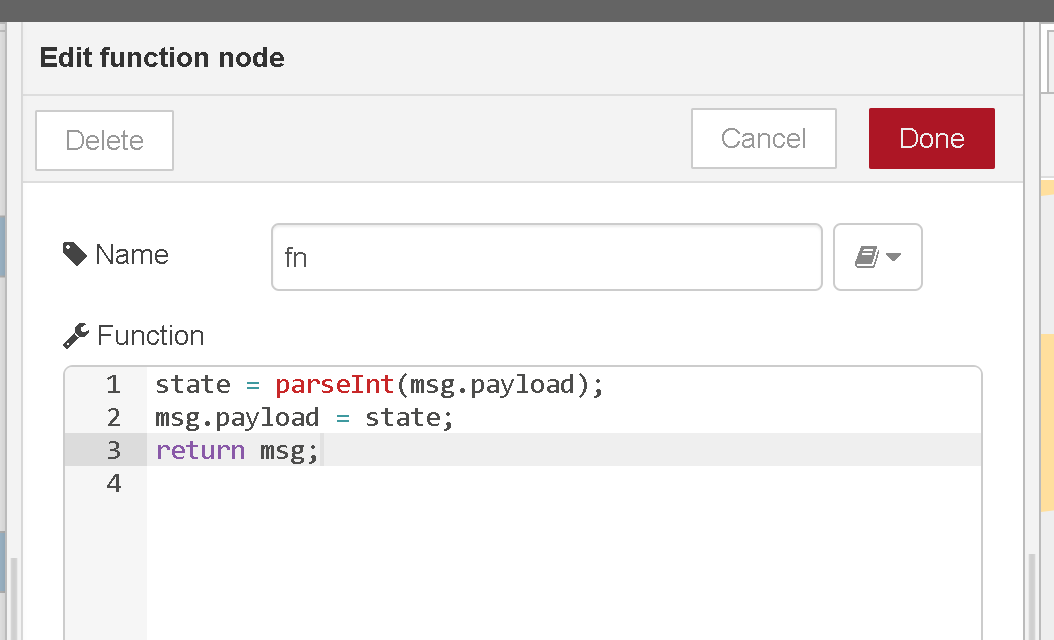


For each raspberry pi used, the input to be copied and pasted, which can be found here at this link: , should be copied and pasted to the localhost of each raspberry pi’s node-red.



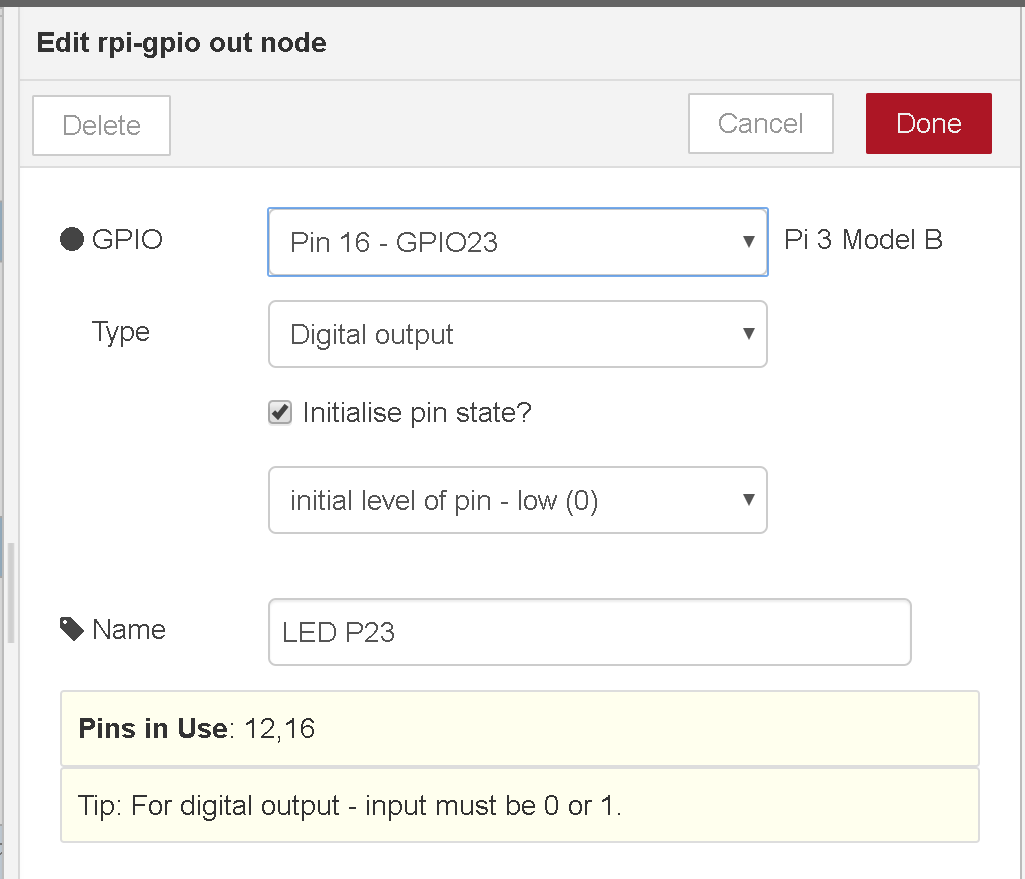
Set the settings to connect as a “Gateway” and Subscribe to “Gateway commands”. The command will be “LED1C2” (aka for LED #1 in classroom 2, Gabby).

Function Node (LOCALHOST Node-Red)



As the input will be sent in a string format (“1” or “0”) we use the parseInt() function to convert it into an integer such as 1 or 0, as the input needs to be an integer.

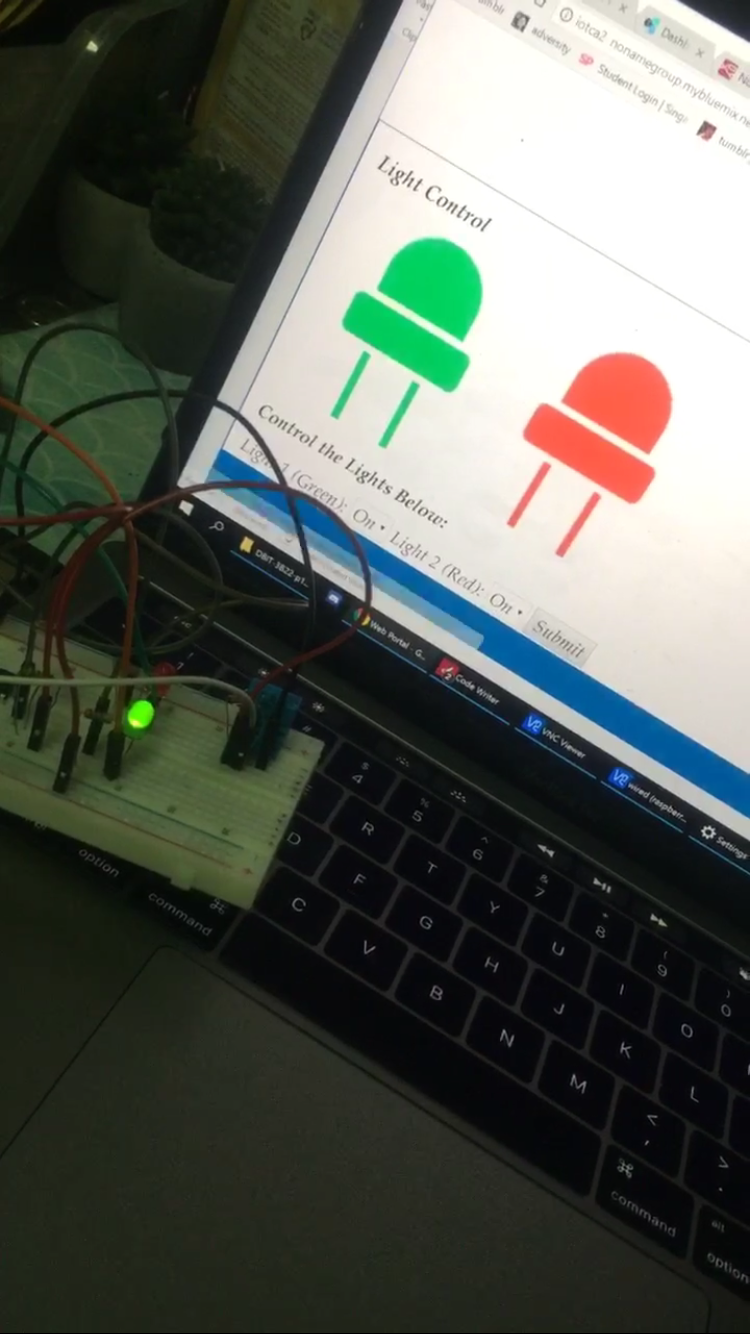
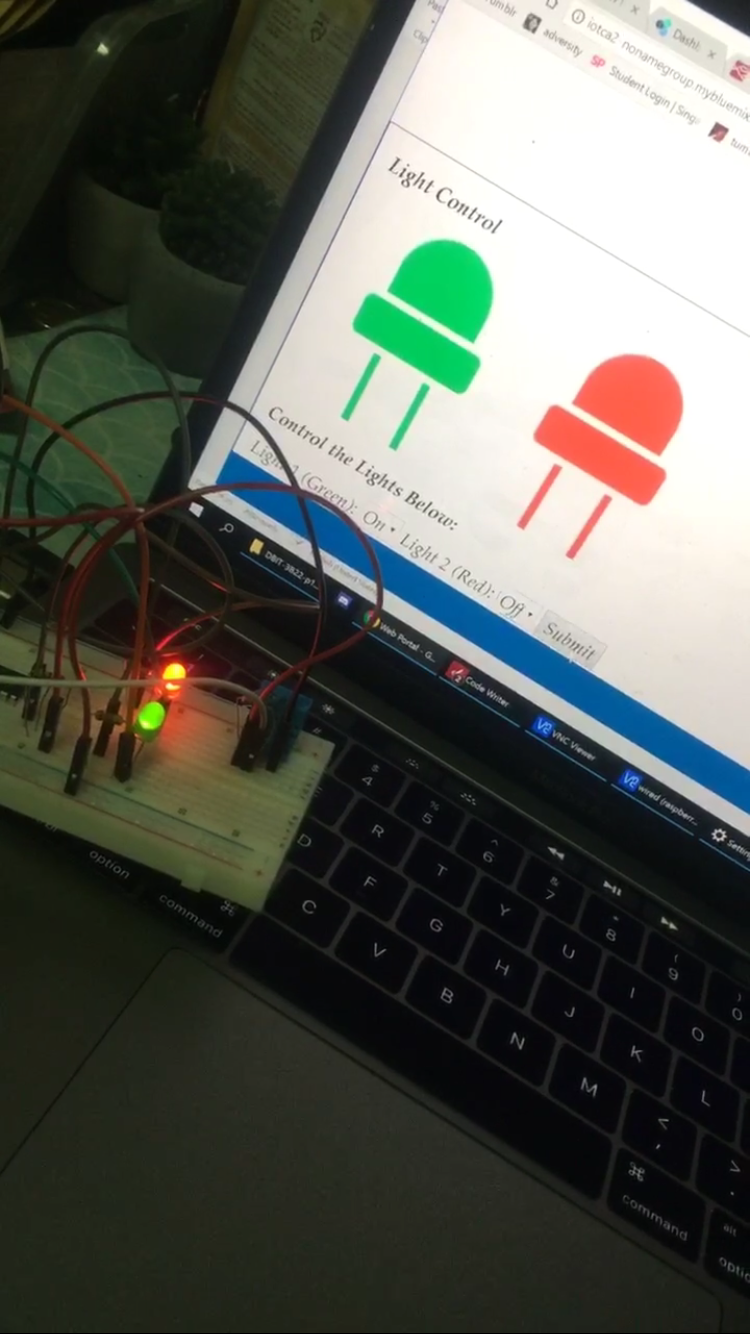
RPI-GPIO Out Node (LOCALHOST NodeRed)



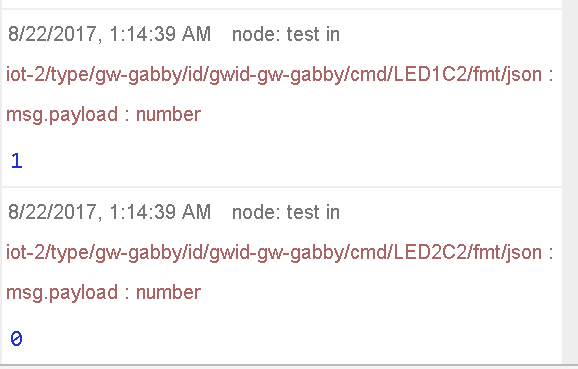
In the rpi-gpio out node, it connects the function and input value and sends it to the led light that is connected to the selected GPIO pin, in this case, the led light is connected to Pin 16, GPIO23. For each LED’s flow, the rpi-gpio node should be configured so that it is connected to the LED’s respective pin.

*Example*

For example, when “on” is selected for the LED 1 (green) and “off” is selected for the LED 2 (red), once the form is submitted, it will send an AJAX Post request to the URL where the parameters “LightStatus1C1” and “LightStatus1C2” will be extracted from the URL visa POST, and published through the watsonIoT Out Node in the IBM bluemix node-red. The Raspberry Pi Subscribes to this topic/message by using the input watsonIoT Input node in the Raspberry Pi’s Localhost Node-Red to retrieve the data published by the watsonIoT Output node.

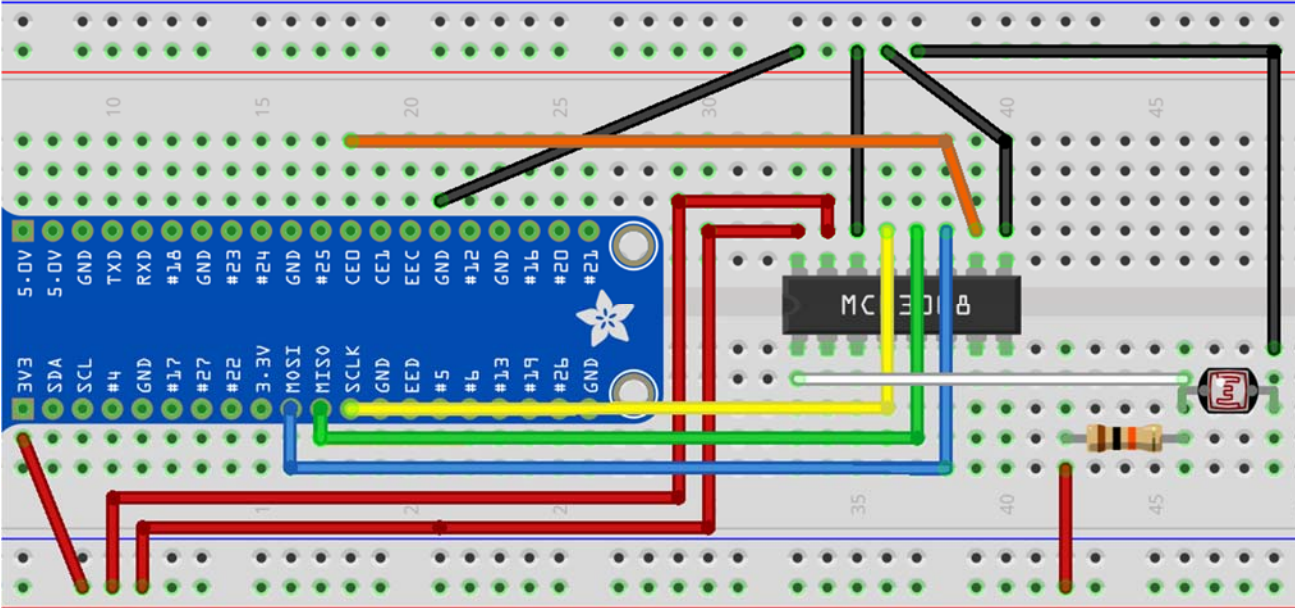


(Both lights are turned on, but when the form is submitted to turn on the green and turn off the red, the red light will be turned off once the form is submitted using POST Method and the page will refresh. Below is the debugger node’s logs, which shows the input values being sent from the watsonIoT node.



(The debugger log in localhost which shows the input values received from IBM Watson Node)

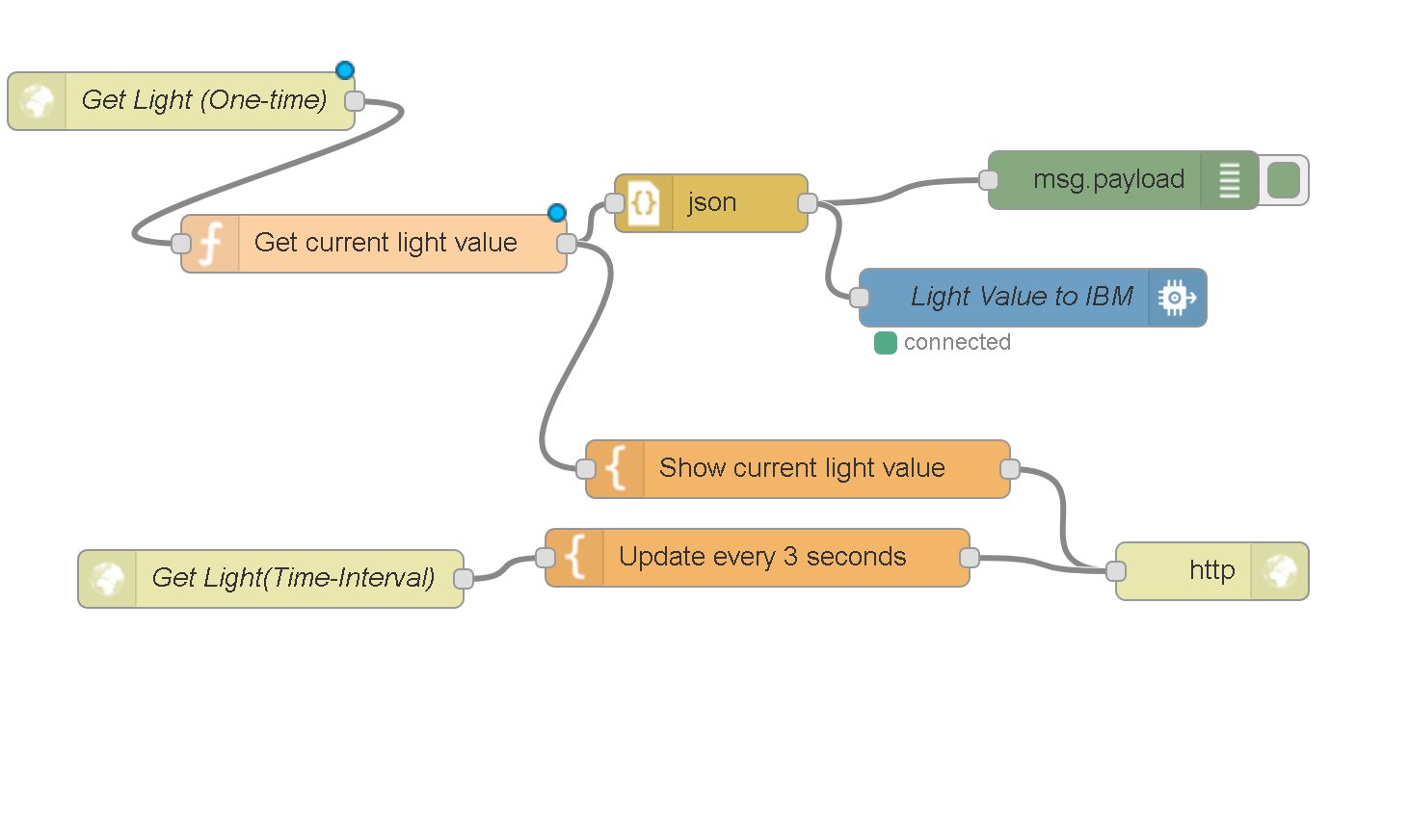
# Section 6 How to Setup the LDR



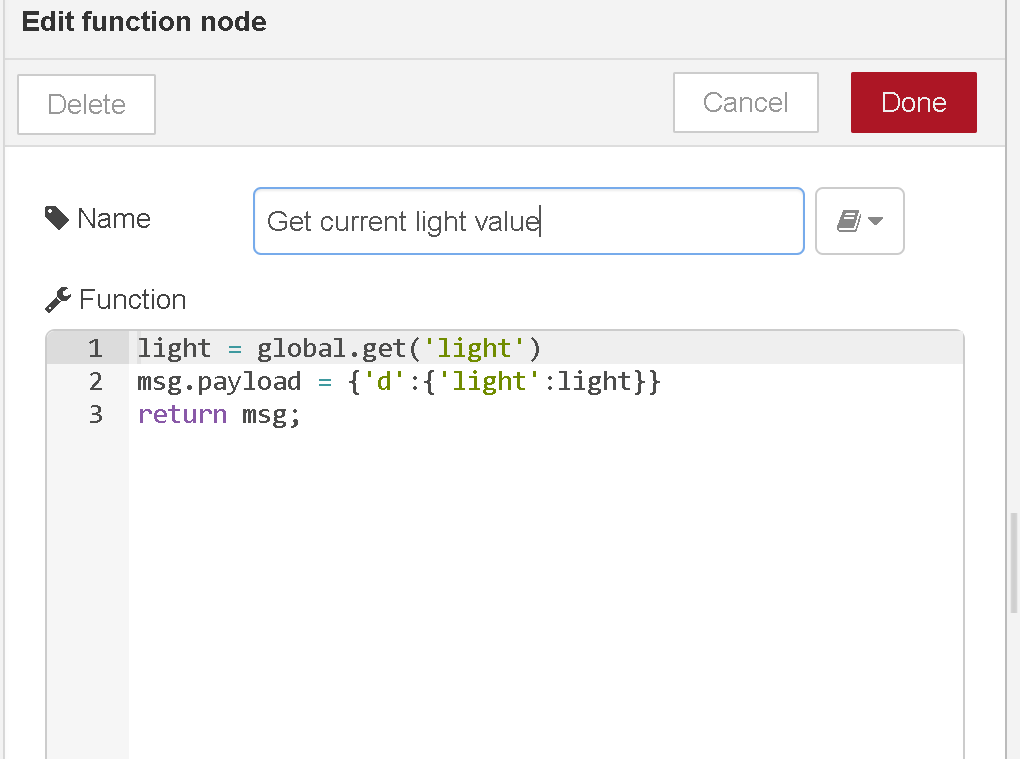
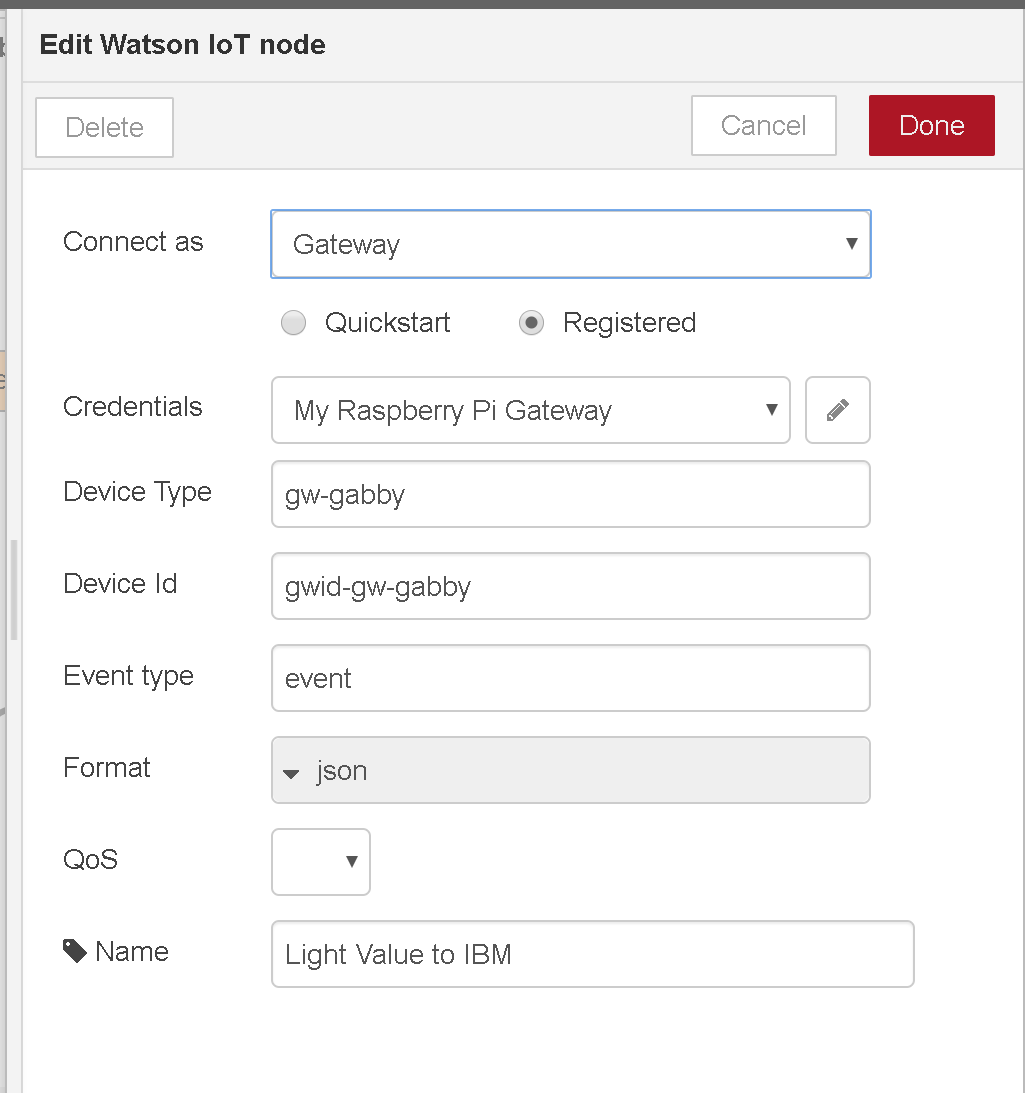
Setup the LDR as shown in the diagram above.

Input (LOCALHOST Node-Red):

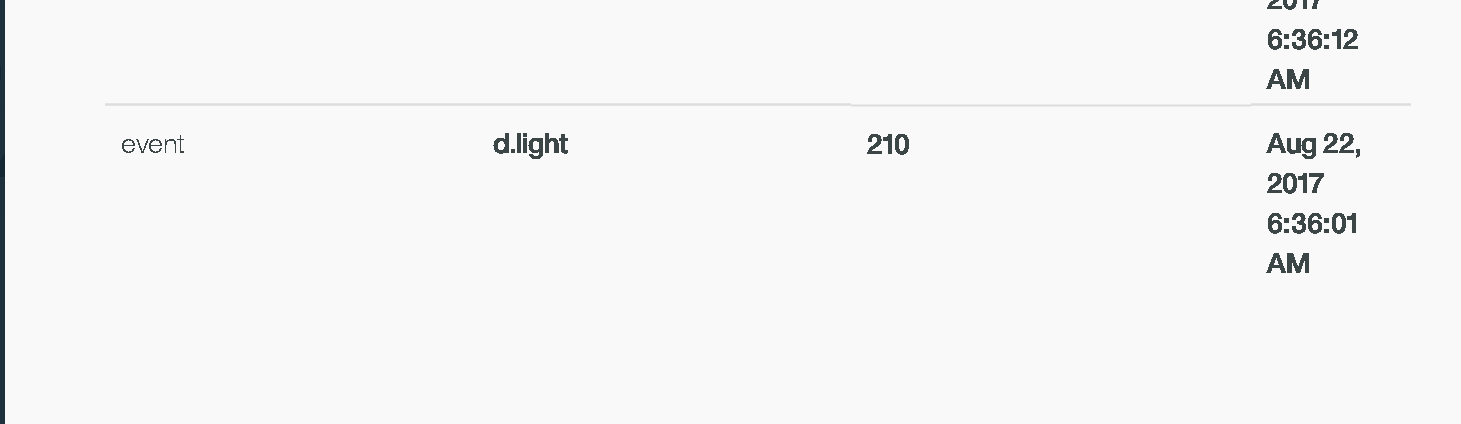
Copy and paste the contents of the file “LDR-LH.txt” and import the the clipboard of your RPi’s Node-Red. It will look like this:



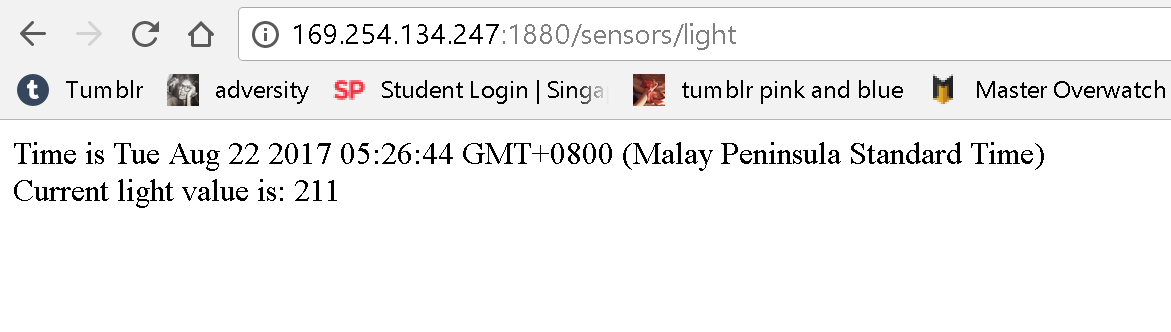
The values of the LDR are sent to the blue watson IOT output node as an event with the following settings and inputs:



The function node, which converts the input into a json friendly input, will have its input published in the events log in the device log in the watsons IoT Dashboard, as shown below in the screenshot, where 210 is the LDR light value.

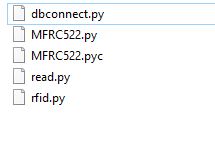


The light values are updated and shown in real time in the page /sensors/light on localhost:



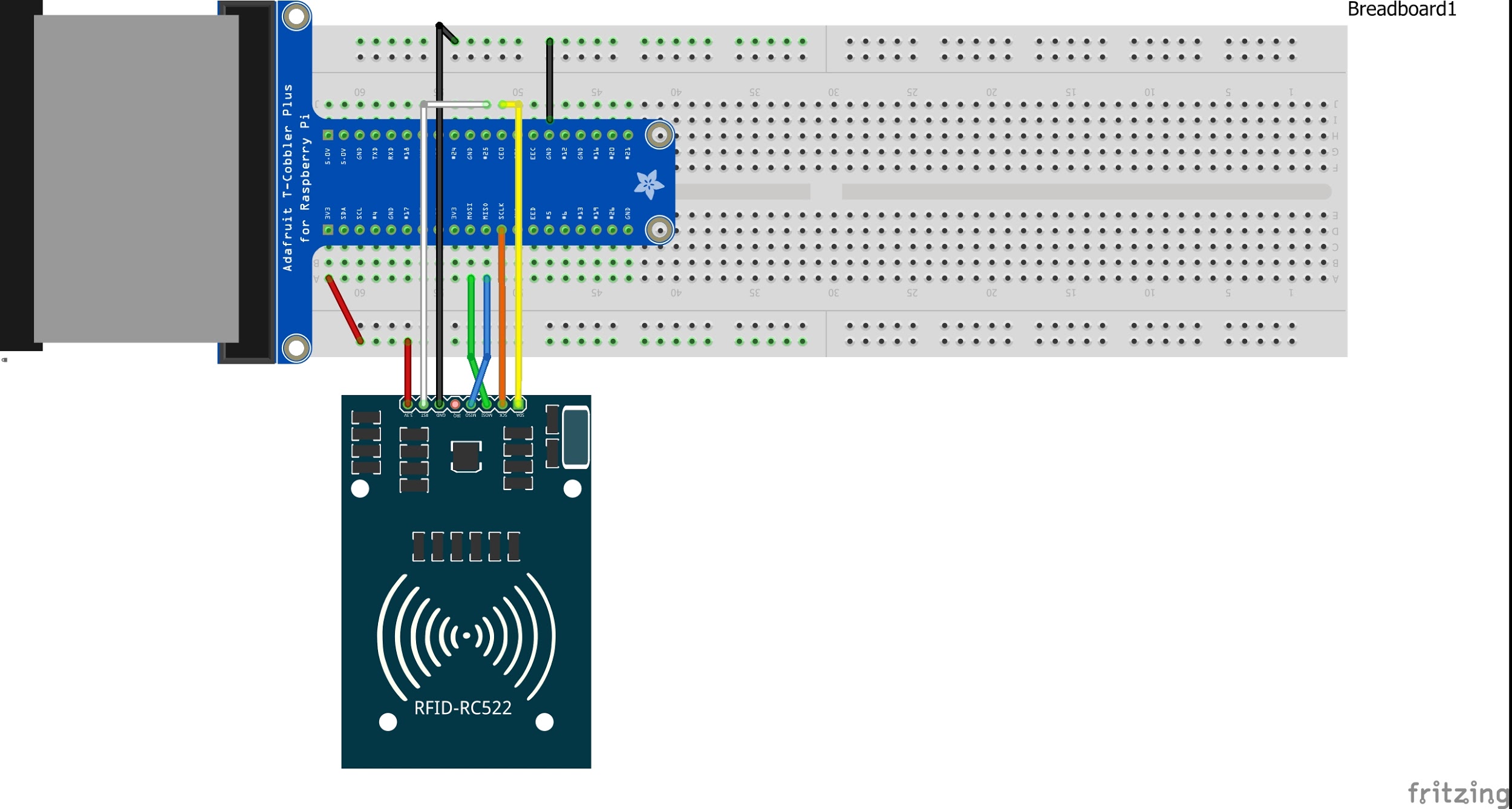
# Section 7 How to Setup the NCF card reader

Before we start, please download the source file from the github.



Step 1.

in your breadboard, follow this few diagram



Step 2.

for your dbconnect.py to work with your mssql, in your raspberry Pi,

install those python libraries :

sudo apt-get --assume-yes update

sudo apt-get --assume-yes install freetds-dev freetds-bin

sudo apt-get --assume-yes install python-dev python-pip

sudo pip install pymssql

source: https://docs.microsoft.com/en-us/sql/connect/python/pymssql/step-1-configure-development-environment-for-pymssql-python-development

Step 3.

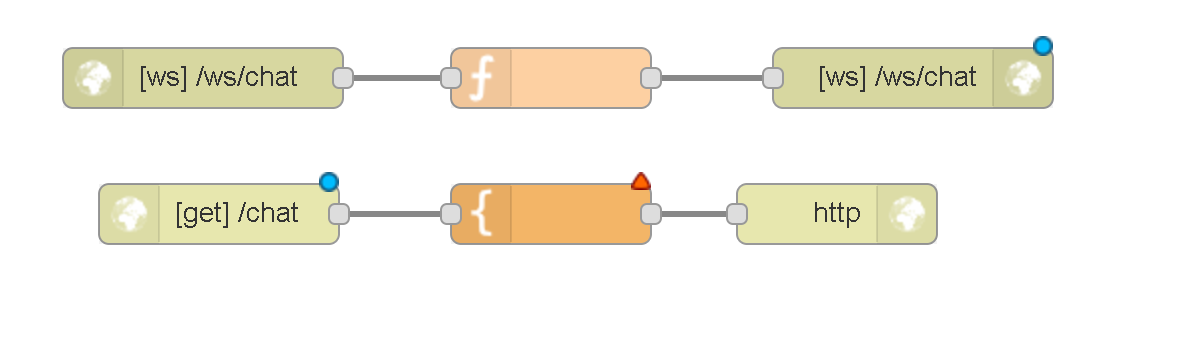
Open dbconnect.py and enter your database info

conn = pymssql.connect(host='yourhost',database='yourdatabase',user='yourusername',password='yourpassword')

Step 4.

run python rfid.py at your raspberry console.

# Section 8 How to Setup the Chat

****

copy and paste the file “**chat.txt**” by importing it into the IBM node-red flow.

In the first row, we have three nodes, WebSocket in, Function and WebSocket out

These are the blocks responsible for creating the communication channel and processing the messages in real-time using the WebSocket protocol. The Function block is very simple. It only removes the value for \_session from the msg object so that the message is broadcast to all clients connected to the WebSocket.

The HTTP in node ([get] /chat) creates an endpoint to receive GET requests and forwards those requests to the Template. The HTTP out node creates the proper response to be sent back to the user after the template has been rendered.

The template can be broken into three parts: page structure, message handling, and visual.

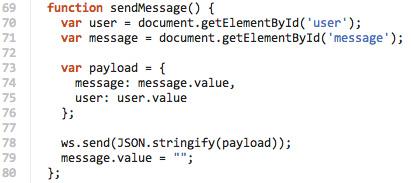
The page structure is a simple HTML page that contains a div that will receive all the chat messages, and a footer that holds the fields for sending messages.

Message handling is done using JavaScript. First, we open a connection to the WebSocket endpoint we created using Node-RED.

Openning a WebSocket connection

Next, we register event handlers for the ws object to handle events such as opening and closing connections with the server and receiving new messages.



When the user sends a message we create an object with the username and the message, which is sent to the server using the send method in our ws object.

(source: https://www.ibm.com/developerworks/cloud/library/cl-rtchat-app/index.html)

**-- End of CA2 Step-by-step tutorial --**