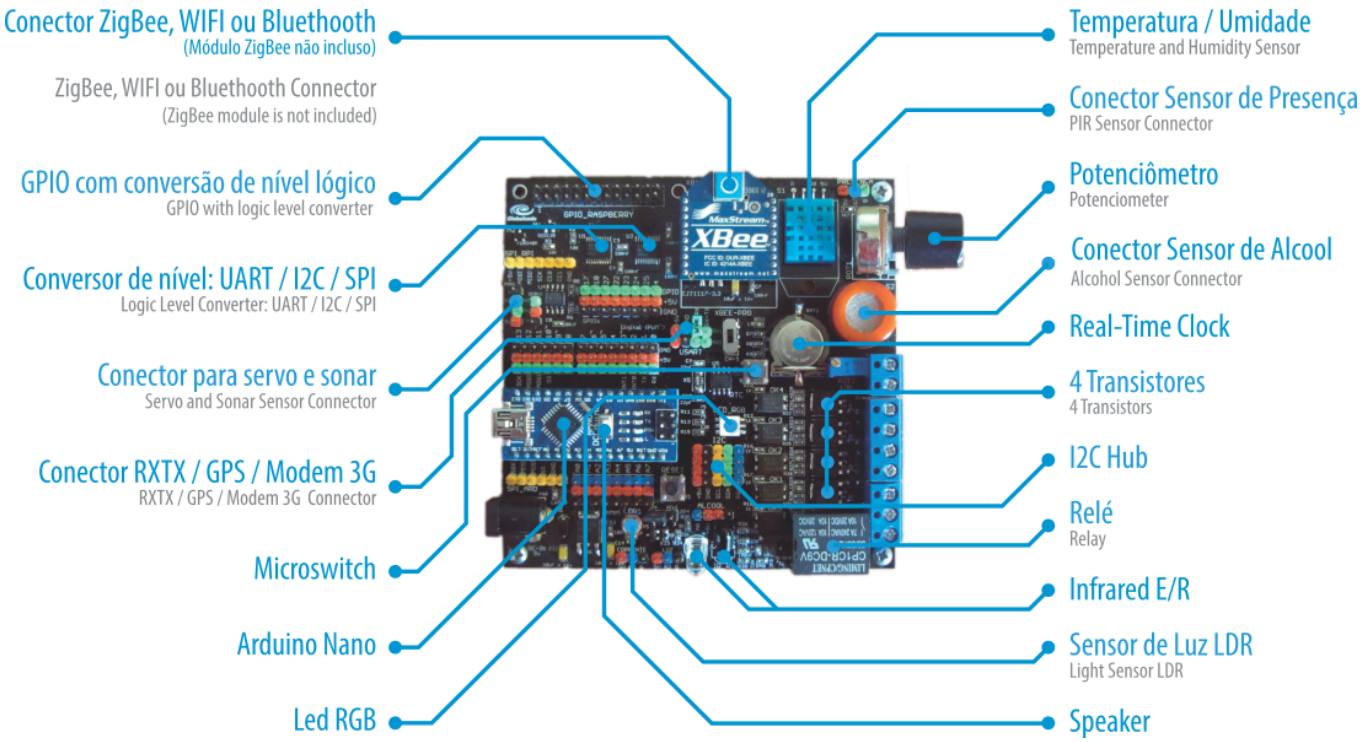
Starting with MQTT using the IoT Surfboard

Vinicius Senger
http://twitter.com/vsenger
vinicius@globalcode.com.br

Yara Senger
http://twitter.com/yarasenger
yara@globalcode.com.br

More information about the IoT Surfboard
www.globalcode.com.br/surfboard/en



Lab 1: Surfing IoT Surfboard with MQTT

Use MQTT.fx or myMQTT Android to control some IoT Surfboard

REST is nice but for IoT can be heavy and you must have direct access to the REST Server. MQTT is a lightweight pub / sub message protocol that works very nice with IoT scenarios and is very, very easy to use. At this time anyone, in any part of the world, will be able to control any IoT Surfboard remotely!

We are using as MQTT "Server", mostly named MQTT Broker, the MQTT Sandbox from IoT Eclipse Foundation. It's a MQTT Broker FREE and OPEN, all you need to do is configure the server and port number and start using!

Step 1: install MQTT client

Install MQTT.fx on your Desktop
http://mqttfx.jfx4ee.org/index.php/download

or MyMQTT for Android phones

Step 2: broker setup

Open MQTT.fx or MyMQTT and go to the connection configuration screen:

MQTT Broker Address : `iot.eclipse.org`
MQTT Port Number : `1883`

Click the connect button and you should get a green light showing you are connected to the broker.

Reading Sensors

To read the sensor from some Surfboard device, you must subscribe to the queue:
`Subscribe Queue Name:` `globalcode/things/<surfboard ID>`

The following standard sensors are available:

Sensor	Value	Sensor
temp	Celsius	temperature
humidity	%	humidity
light	0 - 1023	light
pot	0 - 1023	potentiometer
clock	dd/mm/yyyy	date and time
alcohol	0 - 1023	alcohol sensor

For this workshop we are using the following devices and queues for reading sensors:

Device Description	MQTT Queue
Surfboard Raspberry Pi	globalcode/things/surfboard177
Surfboard USB Cable	globalcode/things/surfboard0
Surfboard 3G Gemalto Concept Board	globalcode/things/surfboardXXX
Surfboard Zigbee	globalcode/things/surfboardXXX

Step 3: Read sensor data for a specific Surfboard

Click the subscribe button and subscribe to any of the queues listed on the the previous table.
globalcode/things/surfboard177

You must receive a JSON including all the sensors and the IoT Surfboard description, ex:

```
{"name" : "surfboard177","firmware" : "3","serial" : "416670068","key" :
"177068","components" : [{"name" : "alcohol","value" : "0"},{"name" : "pot","value" :
"81"},{"name" : "light","value" : "48"},{"name" : "disance","value" : "0"},{"name" :
"clock","value" : "4/10/2016 17:13:21"},{"name" : "temp","value" : "28.00"},{"name" :
"humidity","value" : "45.00"}]}
```

Tip: How to setup the font size: If the JSON text is too small go to Extras -> Settings -> Font size and select the option Use dynamically scaled font size option.

Controlling Actuators

Now you can try to control your IoT Surfboard using the following queue and message:

Publishing Queue Name : globalcode/things
Message : <surfboard ID>/command?value

Examples:
surfboard0/relay?1
surfboard0/relay?0
surfboard0/green?255

Or you can control ALL the surfboard using the * as identification:

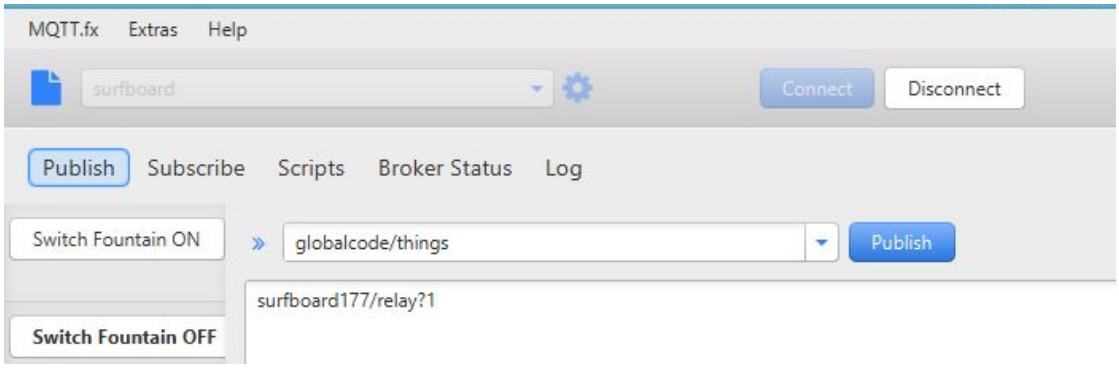
***/relay?1**

The following actuators are available

Command	Value	Actuator
red?<value>	From 0 to 255	red LED
green?<value>	From 0 to 255	green LED
blue?<value>	From 0 to 255	blue LED
speaker?<value>	0 or 1	buzzer
transistor?<value>	0 or 1	General proposal transistor
relay?<value>	0 or 1	relay

Step 4: Play with actuators

Click the Publish button, set the Queue name and message and see the Surfboard in action!



Lab 2: Java + MQTT + IoT Surfboard

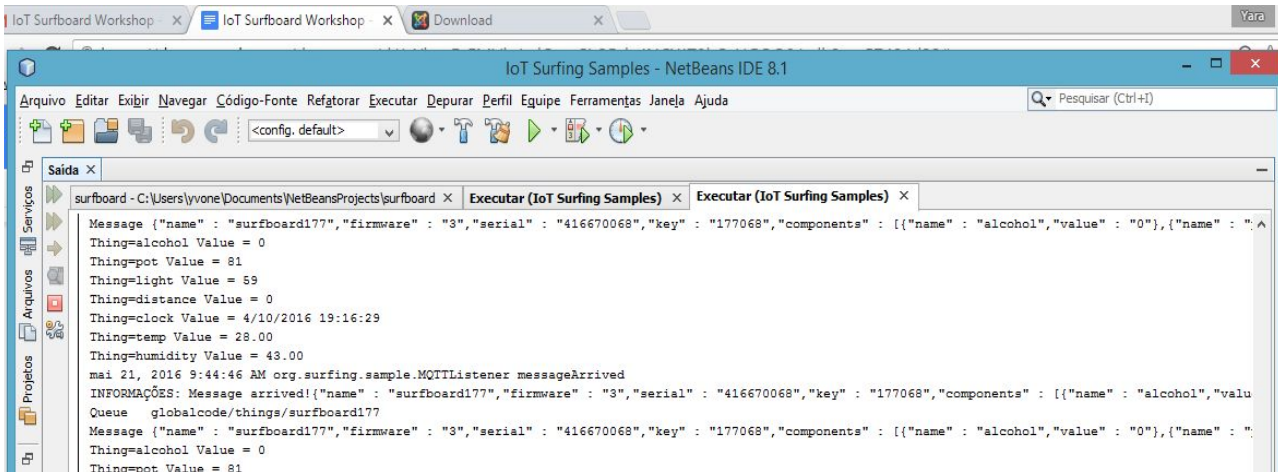
MQTT has nice API's for different programming language called PAHO.
To simplify this workshop we created a sample in our github:

Step 1: clone our git repository https://github.com/Surfinglot/surfboard

Step 2: open the **IoT Surfing Sames** project with your preffered IDE that can be found on the following directory **Surfboard/java/IoTSurfboardSamples**

Step 3: run the sample Main class

When you run this class you will subscribe to the queue of surfboard177, turn on the relay and start getting its sensors information.



It´s a simple maven project and you can edit the class **org.surfing.sample.HelloMQTT and** change the QUEUE to subscribe to other Surdboards by changing the surfboard id and run it again.

```
public static void main(String[] args) {
   try {
      MQTTBroker.getInstance().publish( "globalcode/things", "*/relay?1");
      MQTTBroker.getInstance().subscribe( "globalcode/things/surfboard177");
```

You can also send new messages to control a specific surfboard (as shown on the subscribe method or **all** of them, using the * as you can see on the following publish method.

```
MQTTBroker.getInstance().publish("globalcode/things", "*/relay?1");
```

The code used to get information from the sensors is inside the processMessage method will be executed for ever.

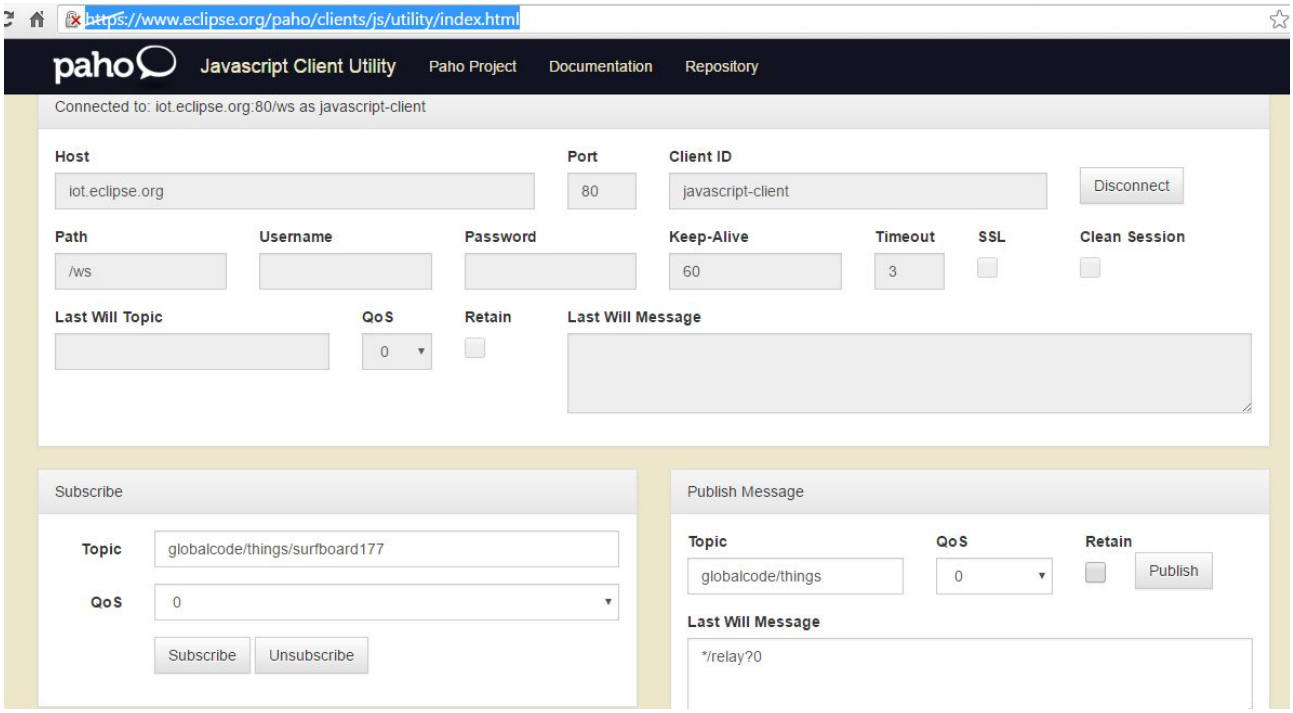
```
public void processMessage(String queue, String message)
```

Lab 3: JavaScript + MQTT + Paho

Another nice usage of MQTT is combined with HTML page and JavaScript.

To start playing with HTML + JS + Paho you can check this nice Web UI at:

https://www.eclipse.org/paho/clients/js/utility/index.html



Step 1: Configure the following fields for publishing messages and play with the actuators or subscribe to get sensors information:

- **Host**: iot.eclipse.org
- **Client ID**: must be unique, so change it!
- **Topic**: globalcode/things
- **Message**: <surfboard ID or * to control all>/<command>?<value>

Command	Value	Actuator
red?<value>	From 0 to 255	red LED
green?<value>	From 0 to 255	green LED
blue?<value>	From 0 to 255	blue LED
speaker?<value>	0 or 1	buzzer
transistor?<value>	0 or 1	General proposal transistor
relay?<value>	0 or 1	relay

Step 2: Open the MQTTSample project on your prefered IDE from the directory:

Surfboard/js/MQTT Sample

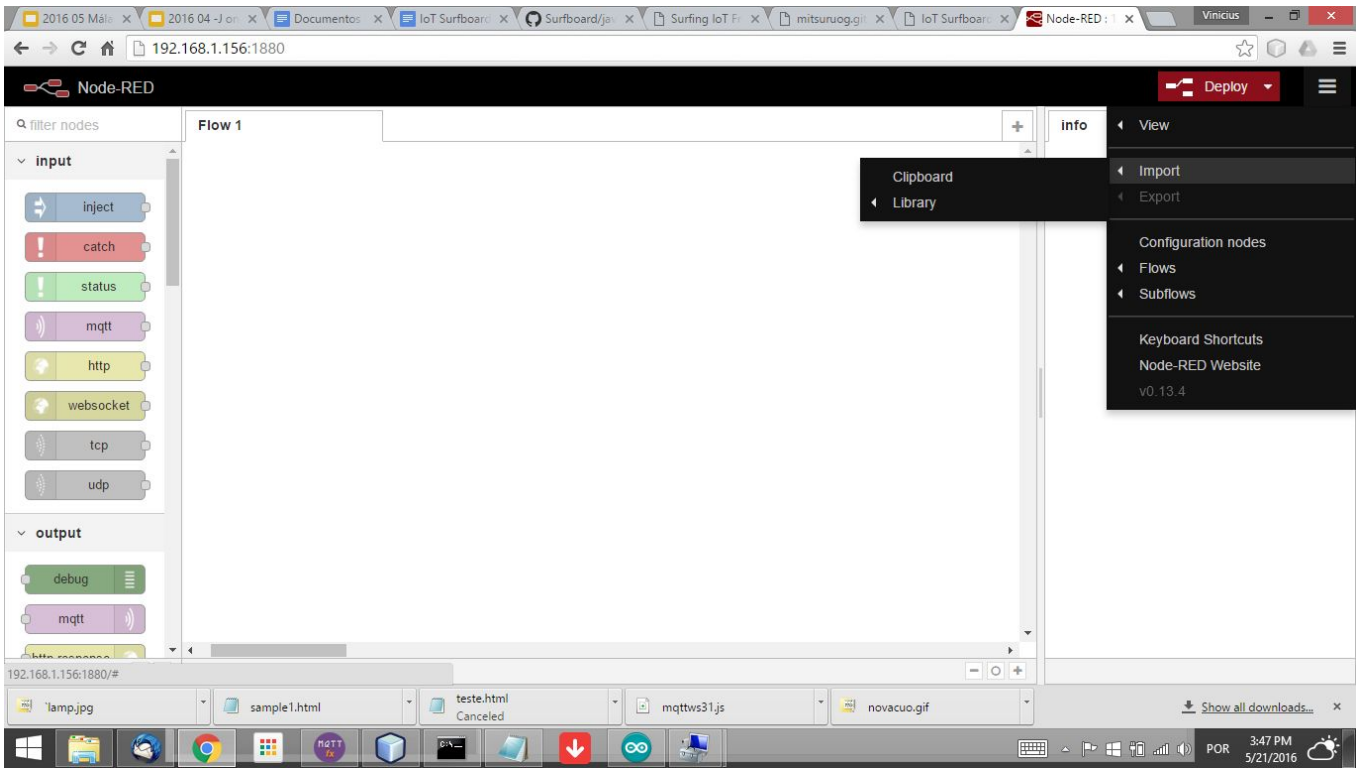
The HTML + JS code is very simple to start, open the index.html file and play with sensors and actuators.

```
<!DOCTYPE html >
<html lang="pt-br">
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1,
user-scalable=no" />
    <meta name="author" content="@Globalcode" />
    <title>
      IoT Surfboard | by Globalcode
    </title>
    <script type="text/javascript" src="js/jquery-1.11.3.min.js"></script>
    <script type="text/javascript" src="js/mqttws31.js"></script>
    <script type="text/javascript">
      // Create a client instance
      client = new Paho.MQTT.Client("ws://iot.eclipse.org/ws", "myClientId"
+ new Date().getTime());
      //"iot.eclipse.org", 80, "aasAaaa");

      // set callback handlers
      client.onConnectionLost = onConnectionLost;
      client.onMessageArrived = onMessageArrived;

      // connect the client
      client.connect({onSuccess: onConnect});

      // called when the client connects
      function onConnect() {
        // Once a connection has been made, subscribes and send a message.
        console.log("onConnect");
        client.subscribe("globalcode/things/surfboard177");
      }
      // called when the client loses its connection
      function onConnectionLost(responseObject) {
        if (responseObject.errorCode !== 0) {
          console.log("onConnectionLost:" + responseObject.errorMessage);
          //alert("conexao perdida" + responseObject.errorMessage);
          client = new Paho.MQTT.Client("ws://iot.eclipse.org/ws",
"myClientId" + new Date().getTime());
        }
```

```
      }
      // called when a message arrives
      function onMessageArrived(message) {
        console.log("onMessageArrived:" + message.payloadString);
        var obj = jQuery.parseJSON(message.payloadString);
        var alcohol = obj.components[0].value;
        var potentiometer = obj.components[1].value;
        var light = obj.components[2].value;
        var temperature = obj.components[5].value;
        var humidity = obj.components[6].value;
        $("#tAlcohol").val(alcohol);
        $("#tPot").val(potentiometer);
        $("#tLight").val(light);
        $("#tTemp").val(temperature);
        $("#tHumidity").val(humidity);
        if (obj.alcohol > 600) {
          alert("Someone is drinking alcohol!!!!!!");
        }
        if (light < 20) {
          alert("you hand is near the light sensor!");
        }
      }
      var status = 0;
      function relay() {
        message = new Paho.MQTT.Message("*/relay?" + status);
        status = status == 0 ? 1 : 0;
        message.destinationName = "globalcode/things";
        client.send(message);
      }
    </script>
  </head>
  <body>
    <header>
    </header>
      Alcohol <input type="text" id="tAlcohol"/><br/>
      Potentiometer <input type="text" id="tPot"/><br/>
      Light  <input type="text" id="tLight"/><br/>
      Temperature <input type="text" id="tTemp"/><br/>
      Humidity<input type="text" id="tHumidity"/><br/>
      <img src="image/lamp.jpg" onclick="relay()"/>
  </body>
</html>
```

Lab 4: Node Red

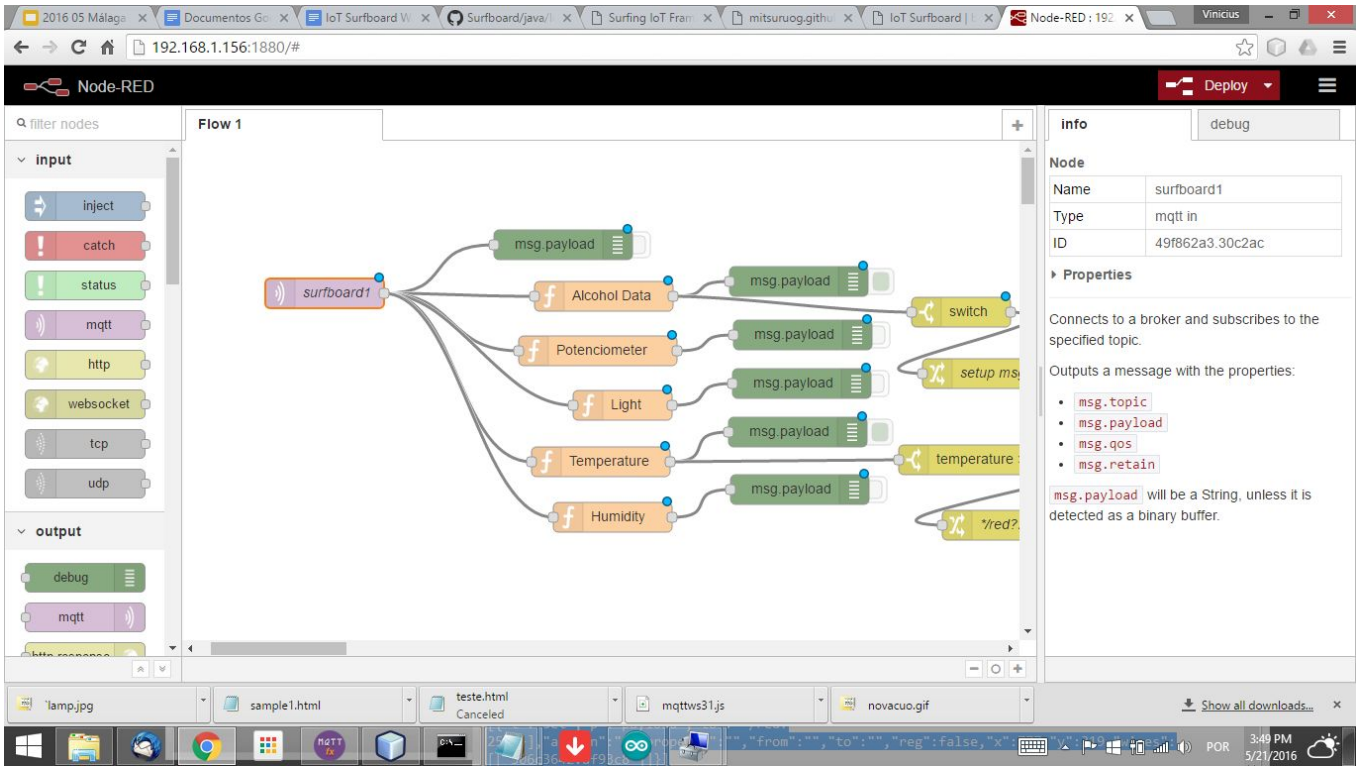
Node Red is an amazing tool to program flows almost codeless. The easiest way to start using is creating an IBM Bluemix trial account and create a Node Red instance. Open your Node Red instance and click in the top right menu, choose Import -> Clipboard.



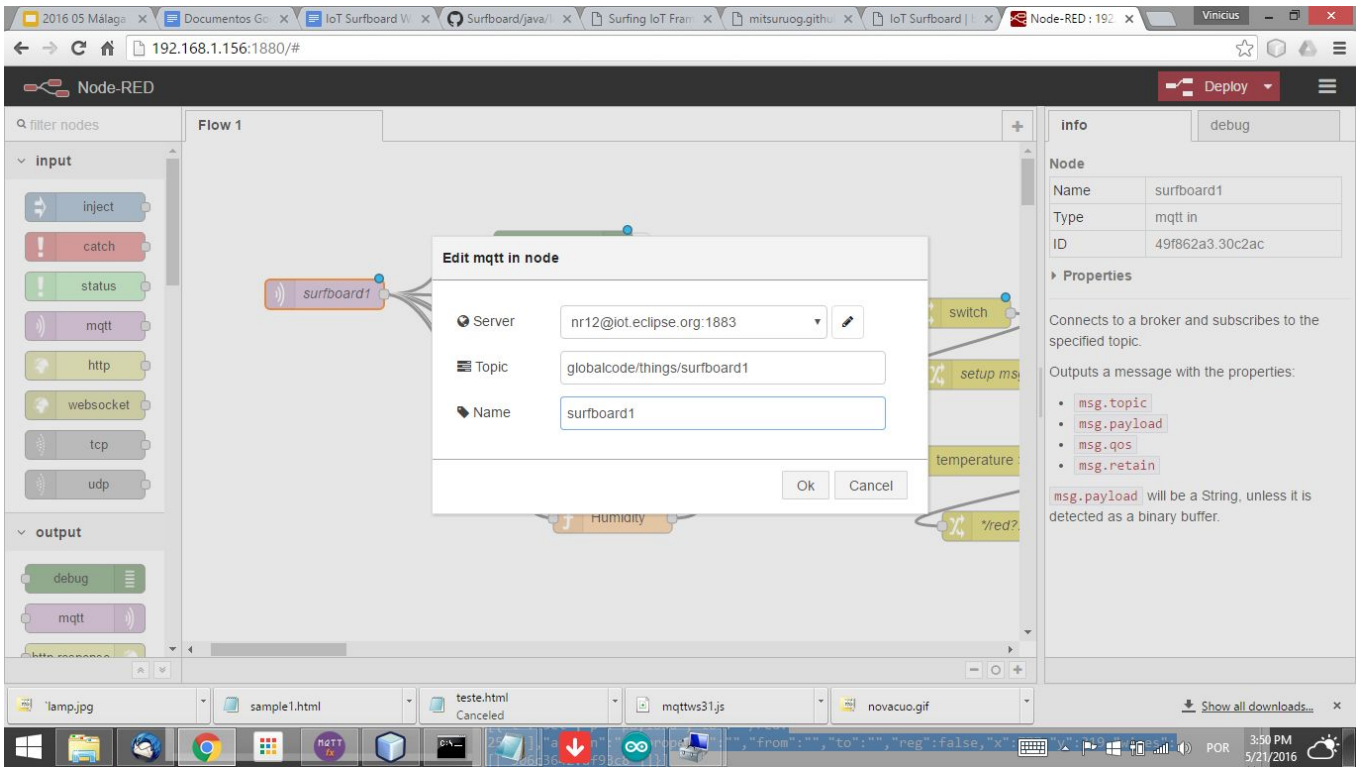
Node Red will ask you a text that represents your Flow, you will find the sample flow for IoT Surfboard at this directory:

Surfboard/nodered

Open the sample-flow and copy the content and then paste it! You just need to drag the flow in any part of the screen:



Now open the surfboard1 node and configure the surfboard name that you want subscribe to read the sensors:



Lab 5: Voice Recognition + Android + Tasker + MQTT

This is is a great combination for Android phone and you can use voice recognition to control your connected stuff.

To use it you must download:

Android Tasker
Tasker Auto-voice plugin
Tasker MQTT publisher plugin

And then ask for instructor support to make the configuration in your phone!

Xamarin Forms + MQTT

If you are Xamarin developer you can check our friend's repository that did a port of MQTT Paho C# to Xamarin Forms:

https://github.com/Studyxnet/MQTT_Android/tree/master/XamForms