Develop IoT with Samsung ARTIK platform

July 25th, 2017

Samsung Strategic and Innovation Center

Table of Contents

Introduction

Welcome to the Samsung ARTIK workshop! This workshop is your introduction to developing IoT with the Samsung ARTIK platform. In this session, you will learn how to:

- Access your target board from the serial console
- Build program flows using Node-RED
- Collect sensor data
- Use MongoDB for storing and retrieving your information
- Enable MQTT so that your ARTIK boards can communicate with each other
- Stream data to ARTIK Cloud
- Use the ARTIK Cloud Rules Engine to trigger cross-device actions

Before you begin

1. Bring your own laptop. For Windows users, pre-install PuTTY.

   PuTTY (http://www.putty.org/) – SSH and Telnet client, for serial console access

2. Have a Gmail account: ARTIK will send emails to your Gmail account. In order to do this, we need to turn on "Access for less secure apps" setting in your Gmail.

   If you are using a corporate Gmail account or if you want to keep your project emails separate from your personal emails, please create a new account for the workshop.

3. Establish an ARTIK Cloud user portal account: We will stream data to ARTIK Cloud service. Create an account at ARTIK Cloud user portal (https://artik.cloud/).

Setup

1. Access Kitra from your serial console. Instructions can be found here:

   https://github.com/rushup/kitra520/wiki/Getting-started-with-Kitra520

   To power up Kitra, press the power button until you hear 2 beeps, then connect its serial USB to your host machine. Follow the instruction in the link below to access the board from your serial console.

   https://developer.artik.io/documentation/getting-started-beta/communicating-pc.html

2. Connect Kitra to the network.

   Your Kitra board should automatically obtain an IP address after it boots up. Do a 'ping' test from your console and take note of your board IP address. If you are not able to obtain an IP address, please go to step 3.

```
[root@localhost ~]# ping www.google.com
PING www.google.com (74.125.28.103) 56(84) bytes of data.
64 bytes from nuq04s18-in-f4.1e100.net (74.125.28.103): icmp_seq=1 ttl=52 time=8.83 ms
64 bytes from nuq04s18-in-f4.1e100.net (74.125.28.103): icmp_seq=2 ttl=52 time=59.9 ms
64 bytes from nuq04s18-in-f4.1e100.net (74.125.28.103): icmp_seq=3 ttl=52 time=17.2 ms
```

   (Ctrl-C to terminate)

```
[root@localhost ~]# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.0.0.2  netmask 255.255.255.0  broadcast 10.0.0.255
    inet6 2601:647:4e01:7b45:489d:21ff:fe85:6c27  prefixlen 64  scopeid 0x0<global>
    inet6 fe80::489d:21ff:fe85:6c27  prefixlen 64  scopeid 0x20<link>
    ether 4a:9d:21:85:6c:27  txqueuelen 1000  (Ethernet)
    RX packets 14769  bytes 20317291 (19.3 MiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 311790 (304.4 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3. In case you are not able to obtain an IP address, please restart dhclient by following the steps below, then repeat step 2.

```
[root@localhost ~]# dhclient wlan0 –r
Killed old client process
[root@localhost ~]# dhclient wlan0
```

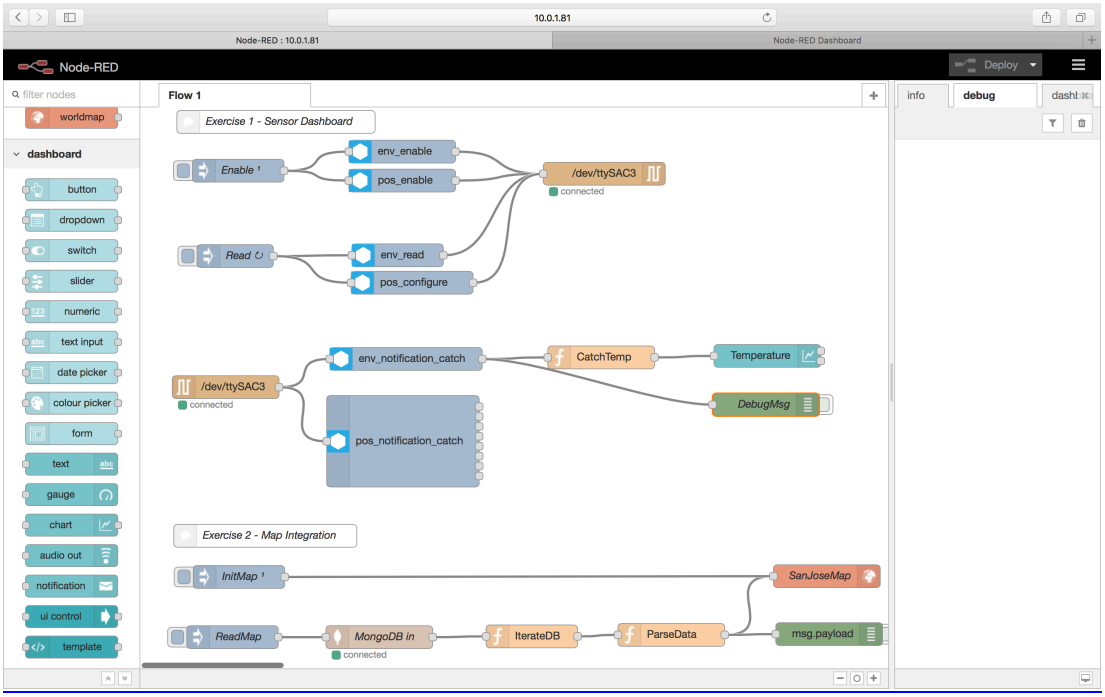
Exercise 1: Monitor sensor data and set up real time sensor dashboard (Node-RED)

In this exercise, we can utilize sensors on Kitra to monitor temperature of each pole, detect if a pole is falling and if there is an earthquake. At the end of the exercise, we will set up a real time sensor dashboard.

1. Start Node-RED in the background from your serial console.

```
[root@localhost ~]# node-red workshop.json &
…
Welcome to Node-RED
===================

29 Jan 13:53:43 - [info] Node-RED version: v0.16.2
29 Jan 13:53:43 - [info] Node.js  version: v4.7.3
…
29 Jan 13:53:48 - [info] Server now running at http://127.0.0.1:1880/
```

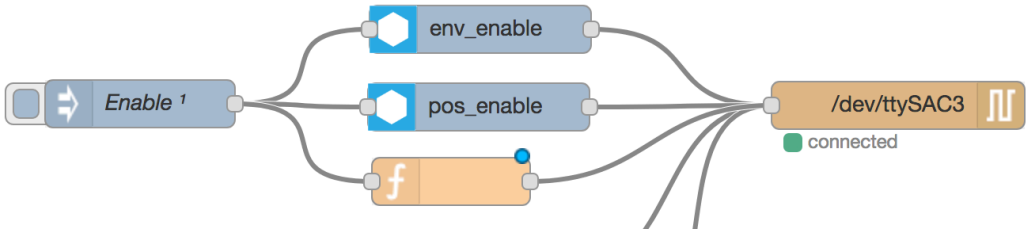
2. Open a browser on your laptop, and launch http://<your_board_ip_address>:1880/ . <your_board_ip_address> should be replaced by the IP address you obtained in Setup step earlier.



3. Design your first project flow.

3.1 Extend the ENABLE flow on the Node-RED canvas.

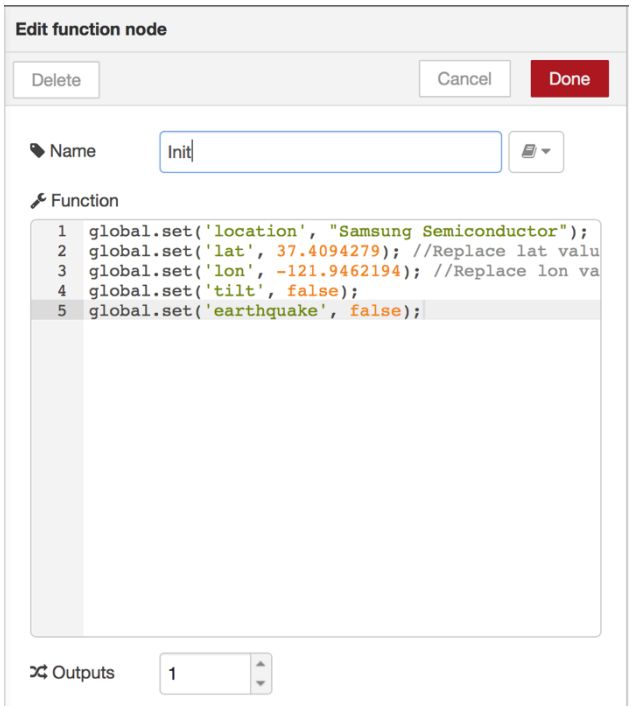
- Drag a "function" node from the node palette and place it between "Enable" node and "/dev/ttySAC3" serial node, and below "pos_enable" node.
- Connect these nodes like below by dragging "wires" between them.



- Double click the "function" node to configure it. In this function, we initialize global variables to define the

```
global.set('location', "Samsung Semiconductor"); //Replace "Samsung Semiconductor" with your own landmark location
global.set('lat', 37.4094279); //Replace lat value with your own latitude
global.set('lon', -121.94621940000002); //Replace lon value with your own longitude
global.set('tilt', false);
global.set('earthquake', false);
```

location of our utility pole, its GPS coordinates etc. Please replace the highlighted parameters with your own landmark location and GPS coordinates. Rename the node to "Init".



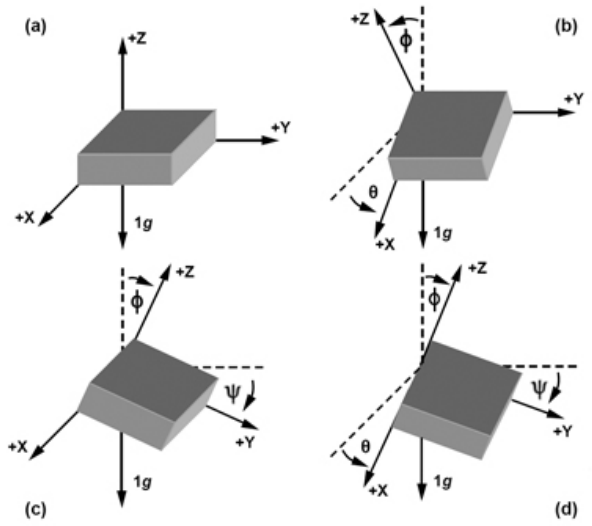
3.2 Click 

at the upper right corner to make your flow live.

4. Extend the CATCH flow to catch the sensor values and notifications.

4.1 Connect a function node to the 1st output of "pos_notification_catch" node.

In the function node, we calculate the angle between the surface of your Kitra board and the world XY plane (we call it inclination angle), and decide if the utility pole is tilted. If the inclination angle is greater than 155 degrees, we will consider the pole to be in a flat position, otherwise, the pole is in a non-flat position.

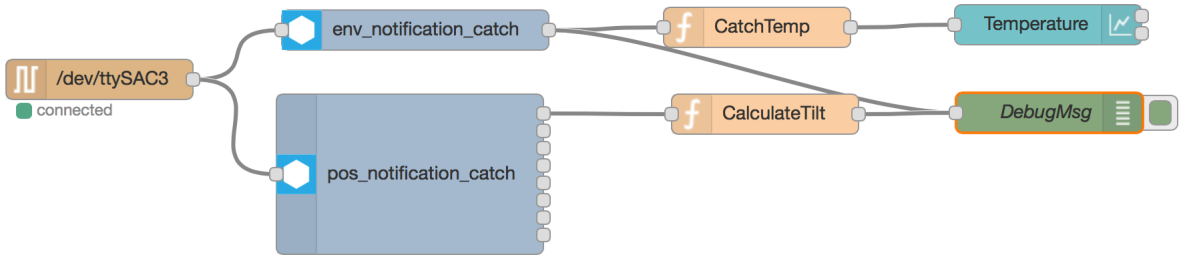


Rename the function node to “CalculateTilt”. And connect its output to the DebugMsg node.

```
//Calculate the angle of the inclination
var acc_x = msg.acc_x;
var acc_y = msg.acc_y;
var acc_z = msg.acc_z;

//Normalize the accelerometer vector
var normal = Math.sqrt(acc_x * acc_x + acc_y * acc_y + acc_z * acc_z);
acc_x = acc_x / normal;
acc_y = acc_y / normal;
acc_z = acc_z / normal;

//Calculate the inclination
var inclination = Math.round(Math.acos(acc_z) * (180 / Math.PI));
console.log("inclination", inclination);

if (inclination > 155) {
    global.set('tilt', false);
    return msg;
} else {
    global.set('tilt', true);
    return msg;
}
```

4.2 

the flow.

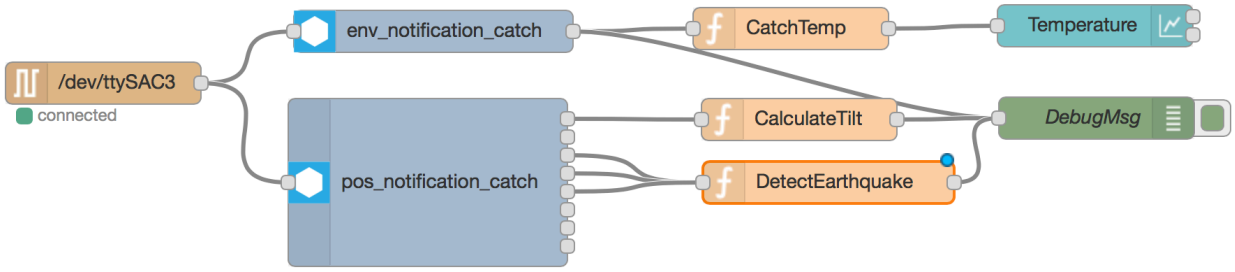


Now, in the debug panel, you will be able to see the raw environmental sensor data every 10 secs. Raw notification data(Accelerometer, gyroscope, magnet) is updated on the debug panel every second.

4.3 Add a function “DetectEarthquake” to capture “tap notification”, “double tap notification” and “free fall notification”.

We will use these notifications to simulate an earthquake.

```
if (msg.noti_mask == 2 || msg.noti_msk == 3 || msg.noti_mask == 4) {
  global.set('earthquake', true);
}
return msg;
```



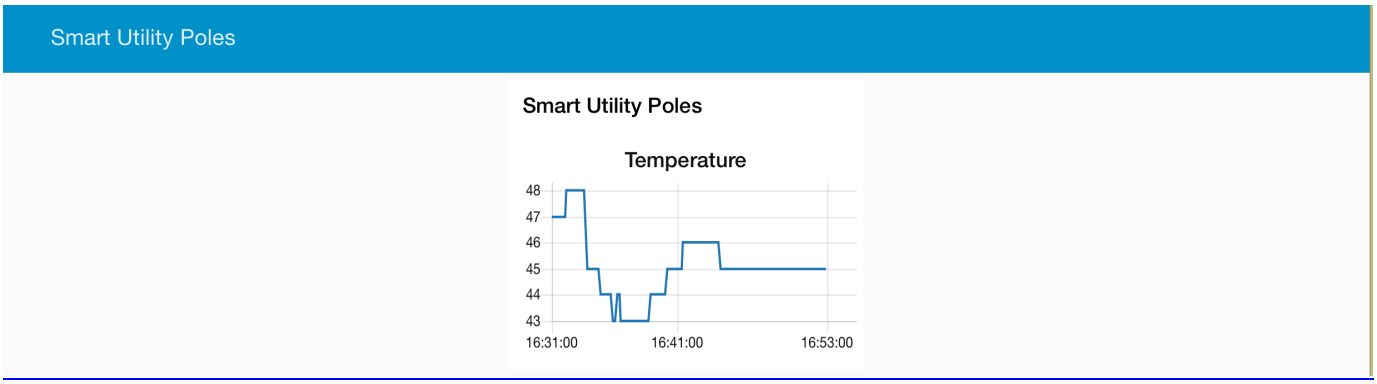
4.4 

the changes. When vibration or freefall is detected on Kitra board, earthquake notification shows up on the debug panel.

5. Set up a real time sensor dashboard. We will use the widgets under the "dashboard" category from the Node-RED palette to build the sensor dashboard.

5.1

In our existing flow, we have a flow for streaming temperature sensor data to the dashboard. In your browser, open http://<your_board_ip_address>:1880/ui. You can see the streamed temperature data.



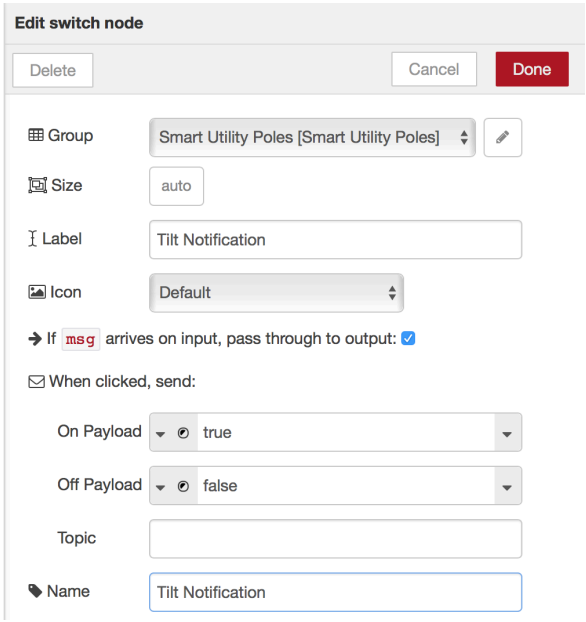
5.2 Display Tilt Notification on dashboard.

- Extend "CalculateTilt" function to support dashboard. Define *msg.payload* to true or false as a tilt indicator. The lines in blue are what have been added to the function.

```
//Calculate the angle of the inclination
var acc_x = msg.acc_x;
var acc_y = msg.acc_y;
var acc_z = msg.acc_z;

//Normalize the accelerometer vector
var normal = Math.sqrt(acc_x * acc_x + acc_y * acc_y + acc_z * acc_z);
acc_x = acc_x / normal;
acc_y = acc_y / normal;
acc_z = acc_z / normal;

//Calculate the inclination
var inclination = Math.round(Math.acos(acc_z) * (180 / Math.PI));
console.log("inclination", inclination);

if (inclination > 155) {
    msg.payload = false;
    global.set('tilt', false);
    return msg;
} else {
    msg.payload = true;
    global.set('tilt', true);
    return msg;
}
```

- Drag a "switch" UI node 

as the output node of "CalculateTilt" function. Change "Label" and "Name" to "Tilt Notification".



6.3 Display Earthquake Notification on dashboard

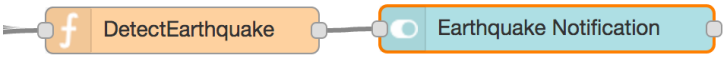
- Extend “DetectEarthquake” function to support dashboard. Define *msg.payload* to true as an earthquake

```
if (msg.noti_mask == 2 || msg.noti_mask == 3 || msg.noti_mask == 4) {
  global.set('earthquake',true);
  msg.payload = true;
}
return msg;
```

indicator. The lines in blue are what have been added to the function.

- Append a “switch” UI node 

as the output of “DetectEarthquake” function node. Rename it to “Earthquake Notification”.

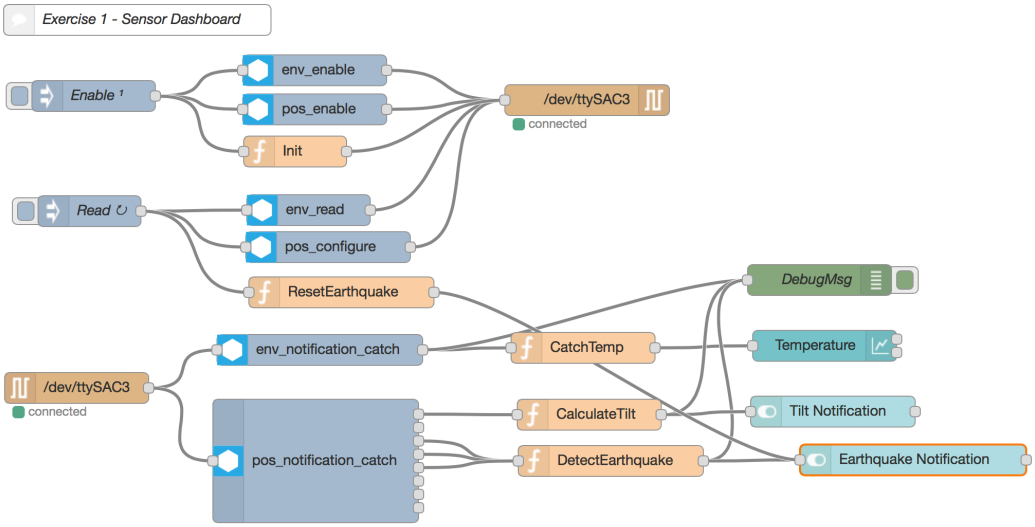


- Correspondingly, we need a function to reset earthquake alert. Drag a function node to the READ flow, and place it under “pos_configure” node. Rename the node to “ResetEarthquake”. The following code resets the earthquake status every time then a sensor reading is triggered.

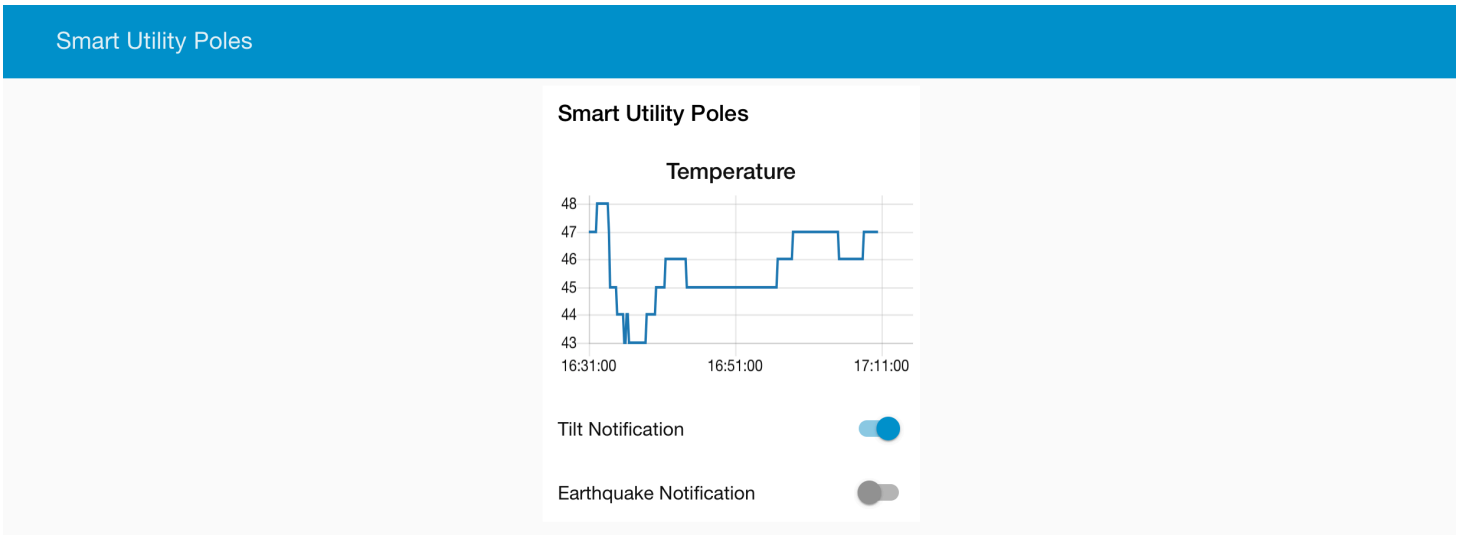
  Wire the output of “Read” node to “ResetEarthquake” node, and output of “ResetEarthquake” node to “Earthquake Notification”, so the UI widget can be reset every time we read the sensor data.

```
global.set('earthquake', false);
msg.payload = false;
return msg;
```

Here is our final flow:

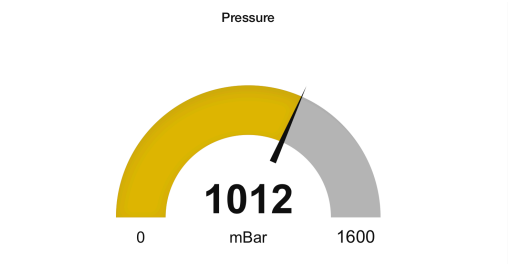


Our real-time sensor dashboard looks like below:



Bonus Question 1:

Extend existing flow in your exercise 1, add gauge UI widget to show Pressure data on the dashboard.

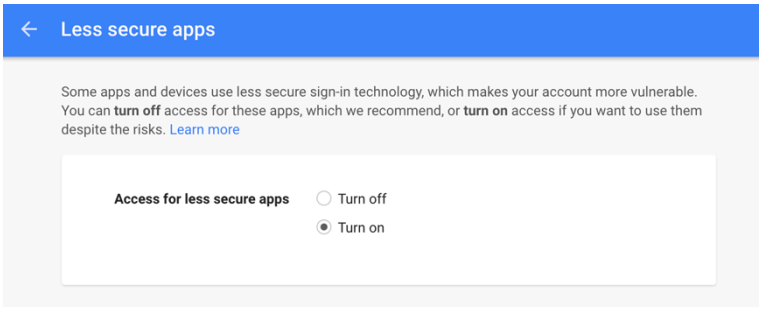


Exercise 2: Get email notification and show aggregate data on a map (Node-RED, MongoDB)

In exercise 2, we will trigger an email alert when the utility pole is tilted. We will also stream our sensor data to MongoDB, and show aggregated real-time data on a New York map.

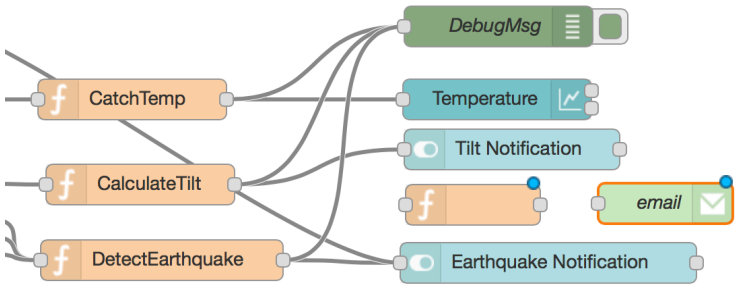
1. We will be using Gmail's SMTP service. For this workshop, turn on the "Access for less secure apps" setting within Gmail.

   Log into your Gmail account, and go to https://www.google.com/settings/security/lesssecureapps to enable the setting.



2. Trigger an email notification when the utility pole is tilted.

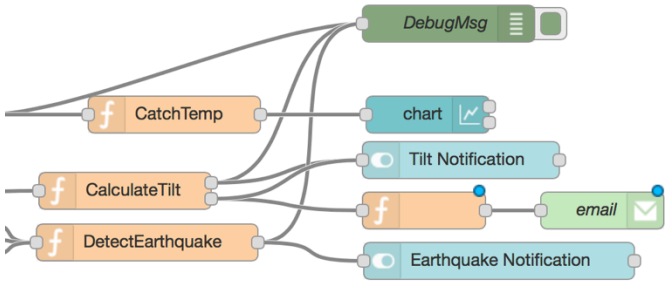
   2.1 Drag a "function" node and an "email" **out** node, and place them between "Tilt notification" node and "Earthquake notification" node.



   2.2 In the "CalculateTilt" function node, change the return values from one parameter to two parameters. Also, update the number of outputs from 1 to 2. The lines in blue are what we have changed in the function.


```
//Calculate the angle of the inclination
var acc_x = msg.acc_x;
var acc_y = msg.acc_y;
var acc_z = msg.acc_z;

//Normalize the accelerometer vector
var normal = Math.sqrt(acc_x * acc_x + acc_y * acc_y + acc_z * acc_z);
acc_x = acc_x / normal;
acc_y = acc_y / normal;
acc_z = acc_z / normal;
global.set('tiltchange', false);

//Calculate the inclination
var inclination = Math.round(Math.acos(acc_z) * (180 / Math.PI));
console.log("inclination", inclination);
if (inclination > 155) {
    msg.payload = false;
    global.set('tilt', false);
    return [msg, null];
} else {
    msg.payload = true;
    global.set('tilt', true);
    return [null, msg];
}
```

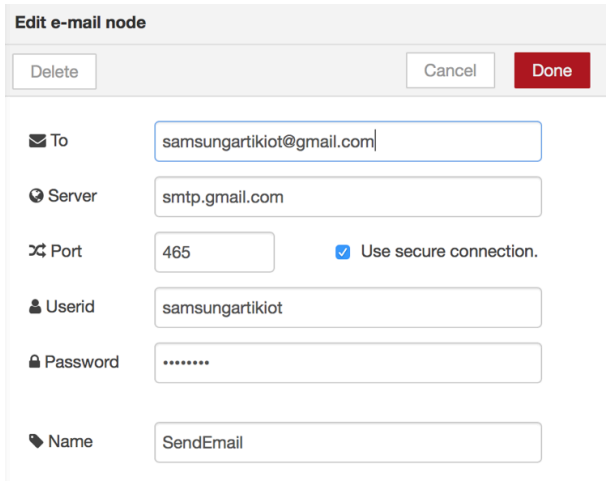
- Connect the 2nd output of “CalculateTilt” function to “Tilt Notification” node for UI update, also connect it to the newly added function node. Connect the output of the function node to the input of email node.



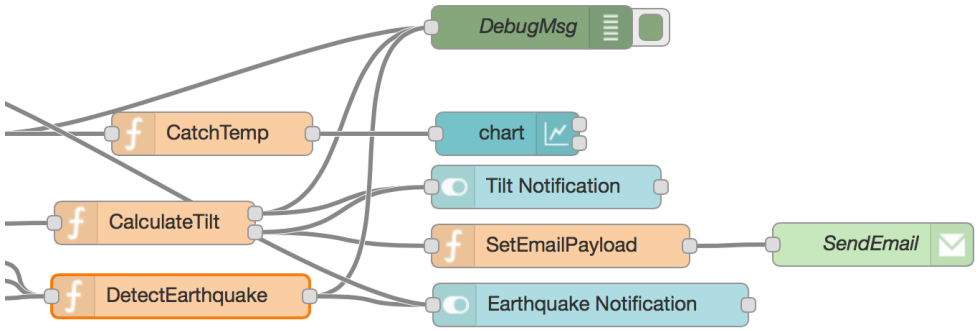
- In the "function" node, we define the email topic and email message in *msg.topic* and *msg.payload*, and Rename the node to "SetEmailPayload".

```
msg.payload = 'Utility pole at  ' + global.get('location') + '  is tilted';
msg.topic = 'Utility Pole Tilt Notification';
return msg;
```

- In the "email" node: Enter your email address in the "To" field, email address user id in the "UserId" field, and email password in the "Password" field. Rename the node to "SendEmail".



Here is what the final flow looks like:

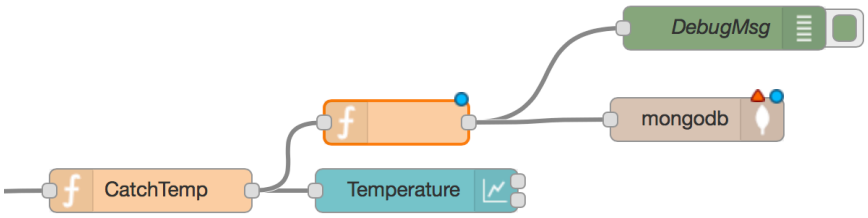


2.3 

your change, and you should be able to get an email alert when your utility pole is tilted.

3. Stream sensor data to MongoDB

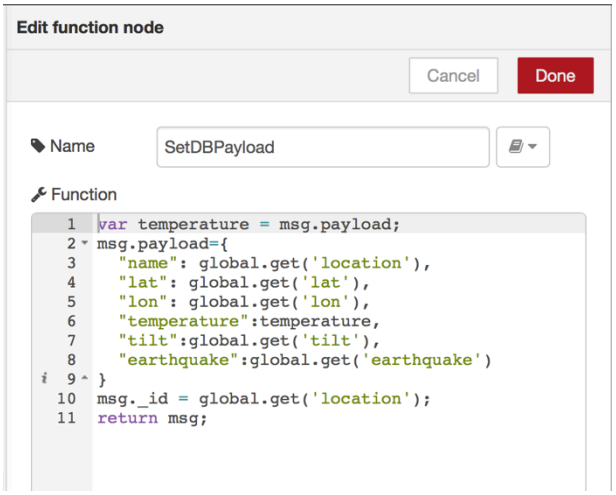
3.1 Place a “function” node and a “mongodb” **out** node above the “temperature” chart node. Connect the nodes as below.



3.2

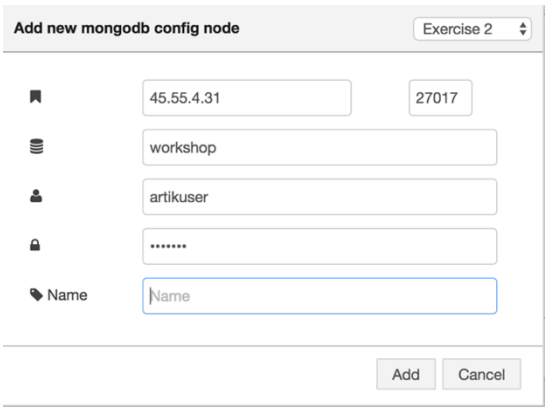
- In the function between “CatchTemp” and “mongodb” nodes, we define a payload that will be sent to MongoDB. Rename to “SetDBPayload”. Here is the code:

```
var temperature = msg.payload;

msg.payload={
    "name": global.get('location'),
    "lat": global.get('lat'),
    "lon": global.get('lon'),
    "temperature":temperature,
    "tilt":global.get('tilt'),
    "earthquake":global.get('earthquake')
}
msg._id = global.get('location');

return msg;
```

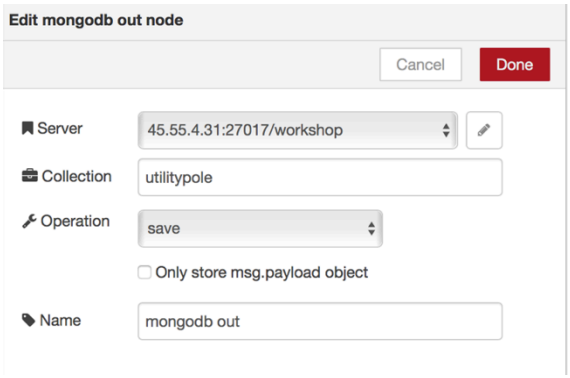


- “MongoDB” output node: Here, we need to configure our remote MongoDB host.

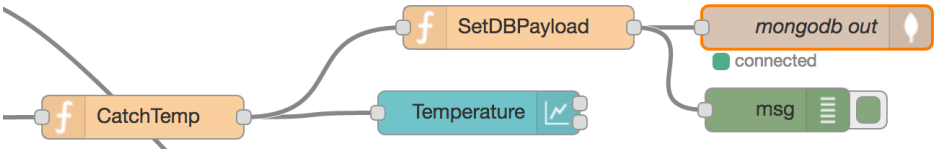
Click  in the “Edit mongodb out node” dialog, “Add new mongodb config node” opens up. Enter “45.55.4.31” as the host IP address, “27017” as the port number, and “workshop” as the database name. Use “artikuser” and “iot2017” as your username and password.



Click “Add” (or “Update” if changing existing parameters) to return to the “Edit mongodb out node” dialog. Enter “utilitypole” as the collection name. Rename to “mongodb out”.

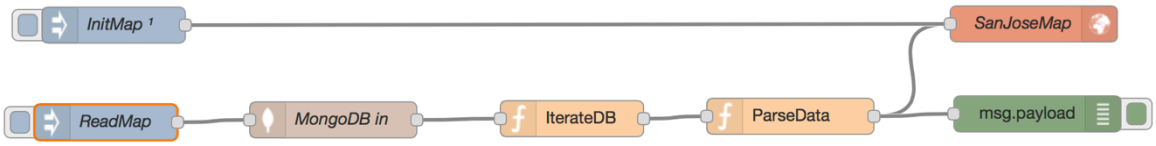


This is what the flow looks like after deployment. Now, your real-time sensor data is being streamed to MongoDB.

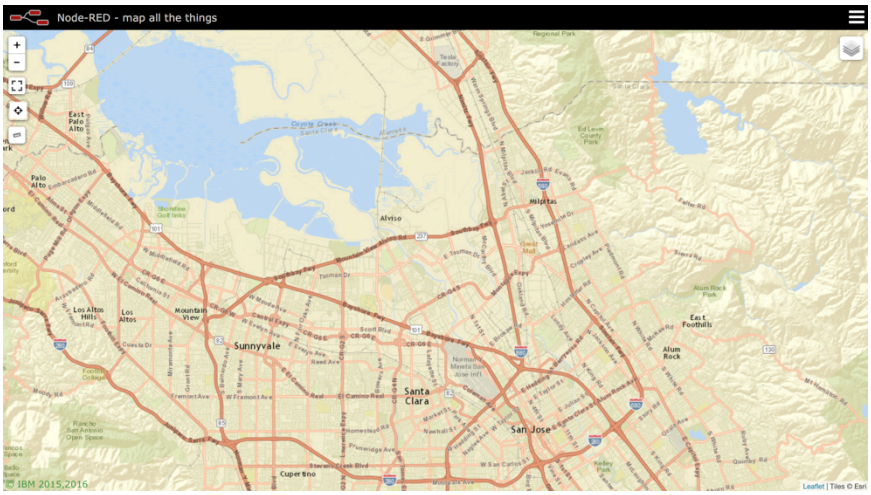


4. Display real-time aggregate data on a map.

4.1 At the bottom of your existing flow, you should find a map integration flow like below.

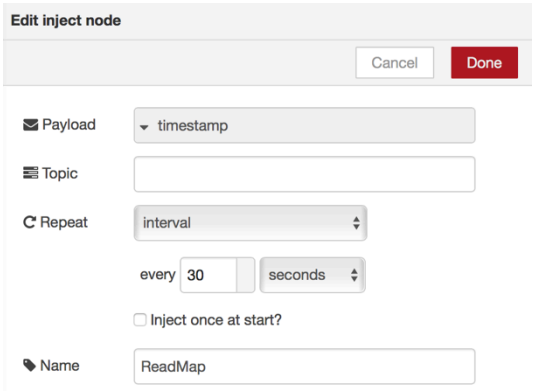


Launch http://<your_board_ip_address>:1880/worldmap, you will see a map of San Jose on your browser.

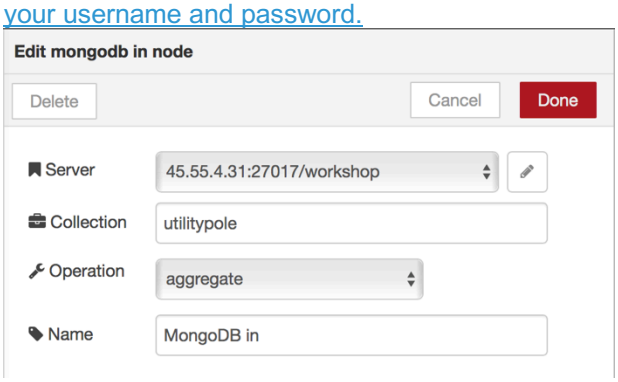


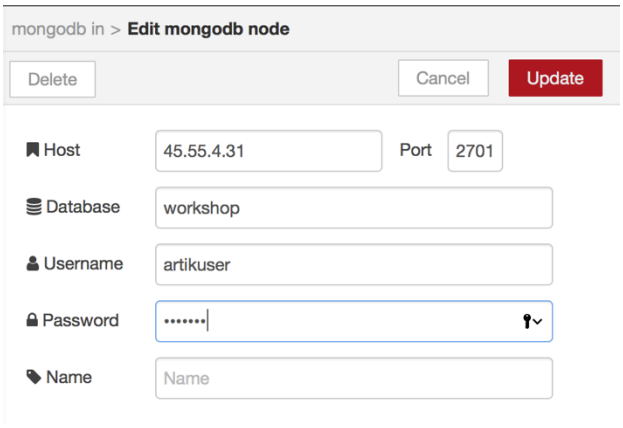
4.2 Configure "ReadMap" node and "MongoDB In" node by double clicking.

- In "ReadMap" node. Set the interval to 15 seconds.

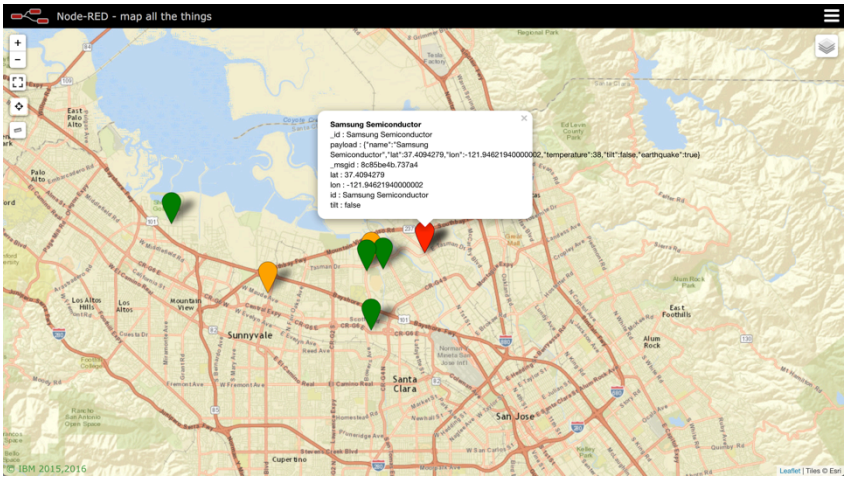


In the "Mongodb In" node, click next to the Server field, "Edit new mongodb config node" opens up. Host server IP address, port number and database name should have been filled in. Enter "artikuser" and "iot2017" as your username and password.

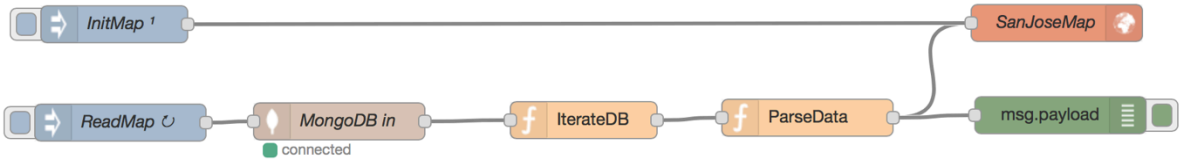




Now, Deploy your changes. Launch <your_ARTIK_IP_address>:1880/worldmap and you will be able to see the real-time sensor data from all of our utility poles. If a pole is tilted, its marker turns orange; if earthquake is detected at a certain utility pole, its marker turns red.



Here is what our final flow looks like:



Exercise 3: Build a utility pole network (MQTT & Node-RED)

In this exercise, we are going to use MQTT to monitor the heartbeats of utility poles. Each MQTT client sends heartbeat message (its location) to the MQTT broker at a fixed interval and subscribes to the heartbeat messages from other MQTT clients.

Select one device as the MQTT broker on each table. Launch MQTT broker 'mosquitto' on the board, and let your group members know your board's IP address.

#mosquitto –d

Both brokers and clients, please go through the steps below.

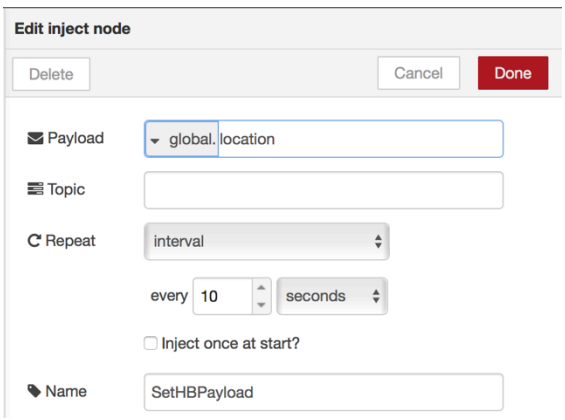
1.

1.1 Publish MQTT heartbeat messages to MQTT brokers.

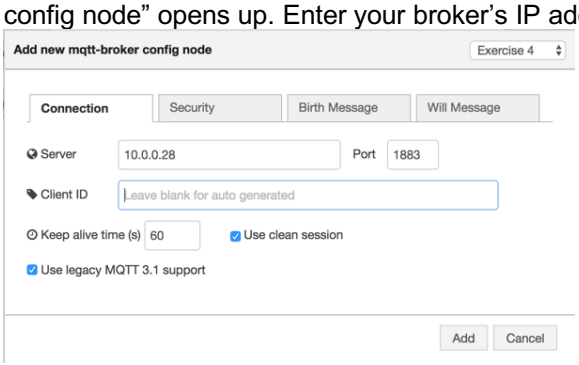
- Drag an "inject" node(shows as "timestamp" node) and an mqtt **out** node 

to the canvas. Wire them up.

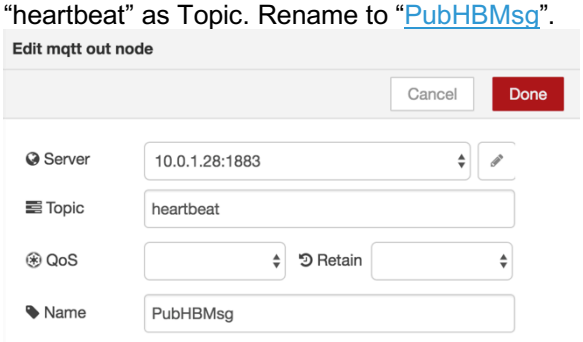
- In the "timestamp" node, change the Payload to be global.location. Configure Repeat Interval to 10 seconds so that your MQTT client will send heartbeat messages to the MQTT broker every 10 seconds. Rename to "SetHBPayload".



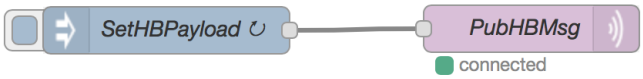
- In the “mqtt” out node, click on the  button in the “Edit mqtt out node” dialog, “Add new mqtt-broker config node” opens up. Enter your broker’s IP address and use default Port number 1883.



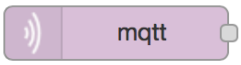
Click “Add” (or “Update” if changing existing parameters) to return to the “Edit mqtt out node” dialog. Enter your “heartbeat” as Topic. Rename to “PubHBMsg”.



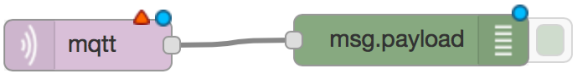
This is how MQTT’s client looks like right now:



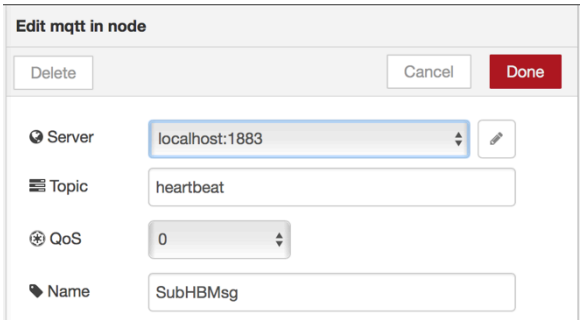
3.2 Subscribe to MQTT hearbeat message:

- Drag an mqtt **in** node 

and a debug node to the canvas. Wire them up.



- In the“mqtt” node, enter “heartbeat” as topic name and change the QoS level to 0. Rename to “SubHBMsg”.



This is how MQTT's client looks like right now:

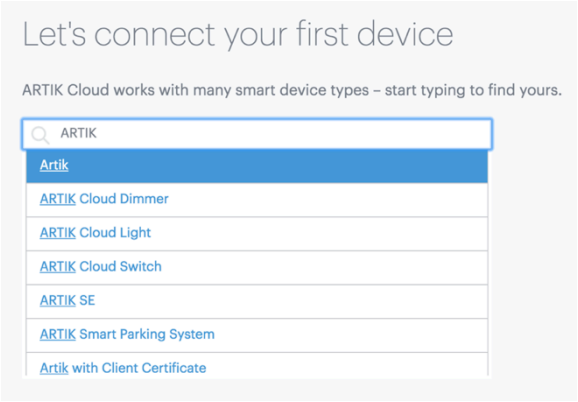
You can then extend your flow to listen to heartbeat messages from utility poles in other groups.

Exercise 4: Connect to ARTIK Cloud and trigger cross-device action by using Rules Engine (ARTIK Cloud, Node-RED)

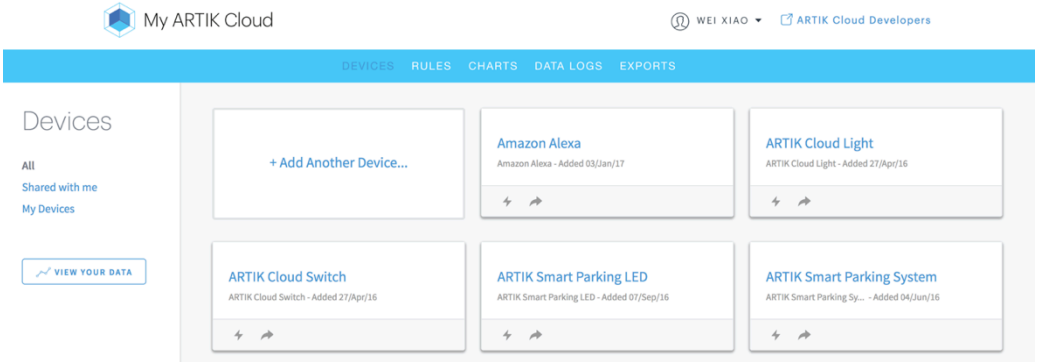
In this exercise, we are going to stream utility pole sensor data to ARTIK Cloud.

1. Log onto the ARTIK Cloud user portal https://artik.cloud.

1.1 If this is your first device, you will be re-directed to the screen below:

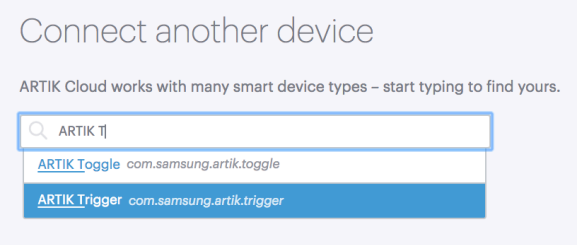


Otherwise, under "MY ARTIK CLOUD"/"DEVICES", you will see all of your connected devices.

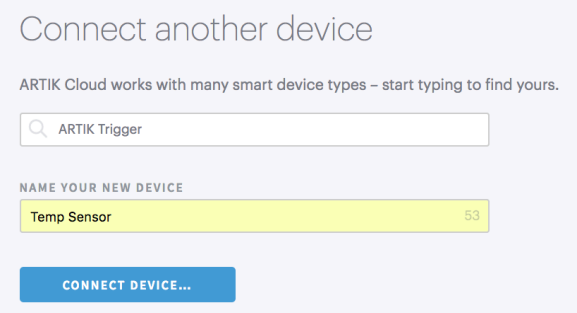


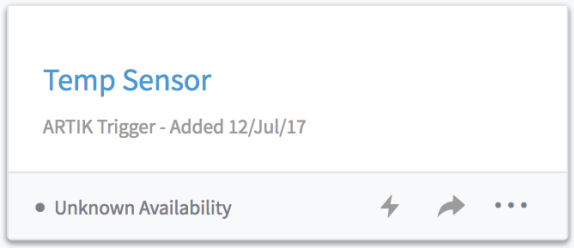
Click on the "Add Another Device…" button.

1.2 Search for "ARTIK Trigger" and select it. "ARTIK Trigger" is a public device type, which includes "temperature" attribute.

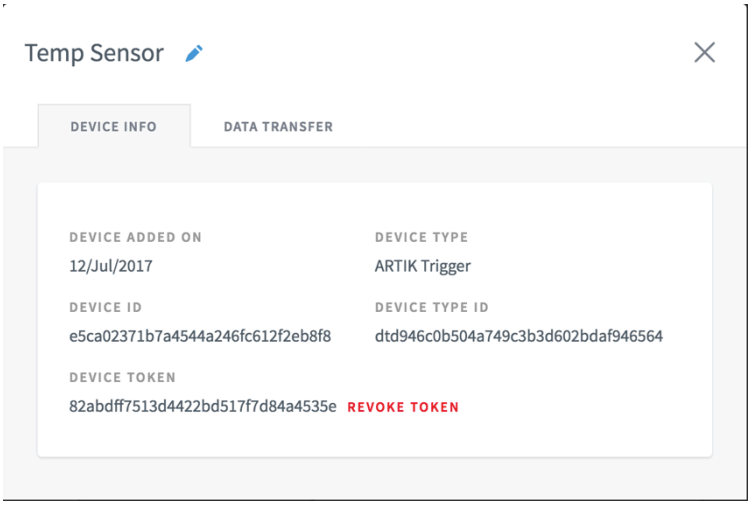


1.3 Change the device name to "Temp Sensor", and click the "CONNECT DEVICE…" button. A new device will be created.

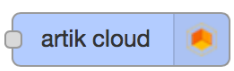




1.4 Click the "Temp Sensor" device name. A Device Info modal will popup which contains your Device Type, Device ID, Device name, and additional details. Click the "GENERATE DEVICE TOKEN…" link to generate a device token.



We need to use the **Device ID** (not "**DEVICE TYPE ID**") and **Device Token** to associate your temperature sensor with ARTIK Cloud. Make a copy or leave the window open.

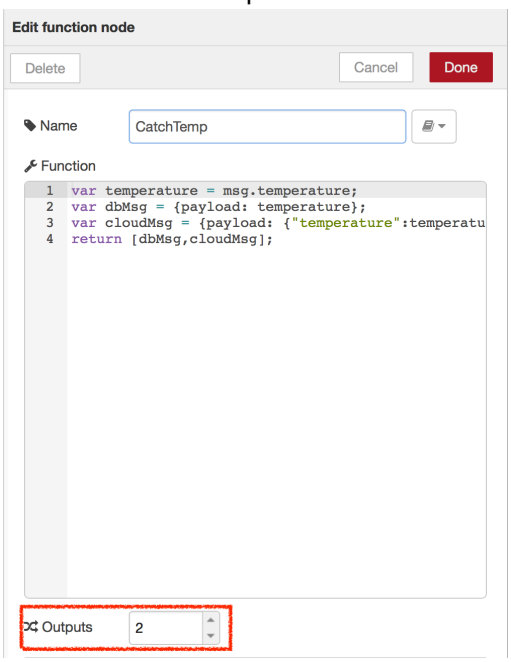
2. Drag an ARTIK Cloud **out** node 

to the canvas, and place it below the "Temperature" nodes.

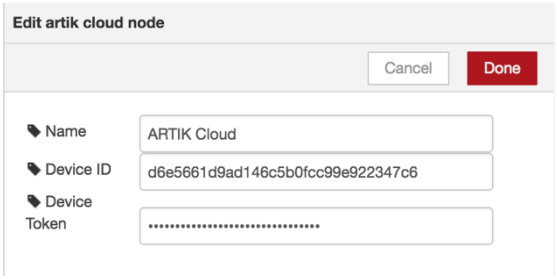
2.1 Update the “CatchTemp” function to define a message payload for ARTIK Cloud.

```
var temperature = msg.temperature;
var dbMsg = {payload: temperature};
var cloudMsg = {payload: {"temperature":temperature}};
return [dbMsg, cloudMsg];
```

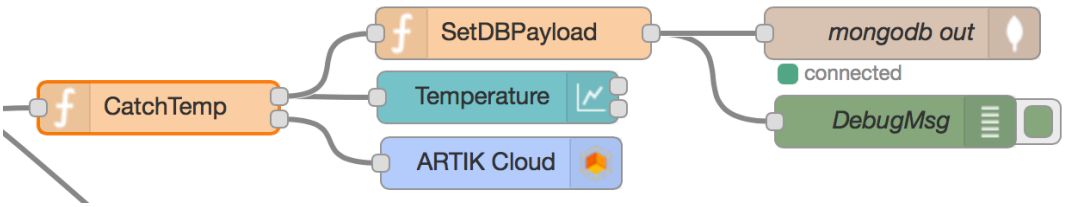
Increase “CatchTemp” function’s number of outputs to 2.



2.2 Double click the ARTIK Cloud **out** node and enter the Device ID and Token in the Edit Window, Rename to “ARTIK Cloud”.

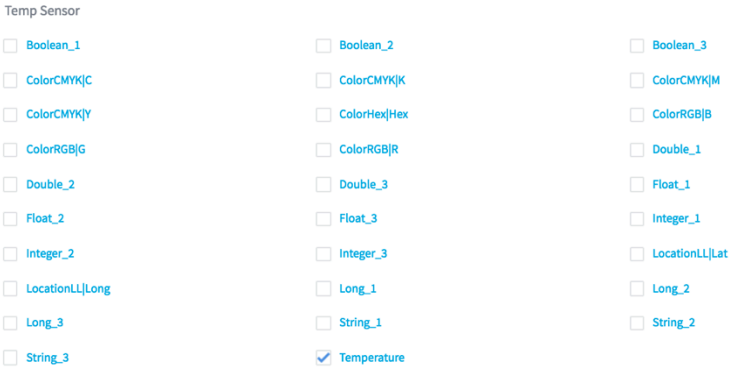


2.3 Connect the 2nd output of ARTIK Cloud **out** node to ARTIK Cloud node.
This is what your final flow should look like:

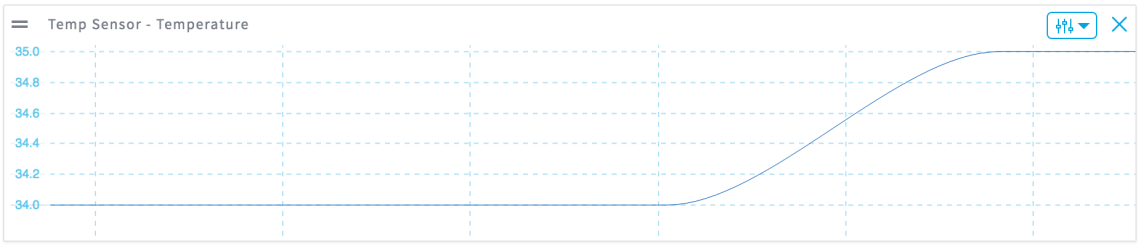


3. Go to the ARTIK Cloud user portal, click 

button, then click the "+/- CHARTS" button on the upper left side. Click on the "Temperature" checkbox to view the utility pole's temperature.



You will see the streamed temperature in your portal:

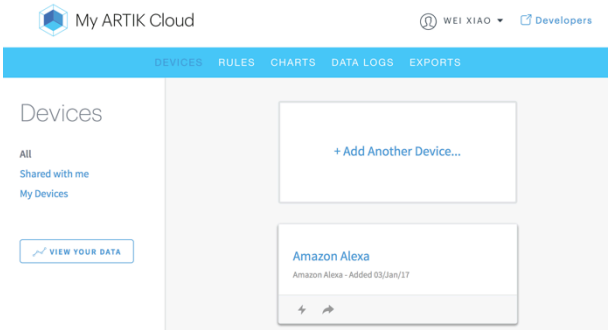


Rules Engine

Now, let's explore an advanced feature within ARTIK Cloud: the Rules Engine. We will define a rule in ARTIK Cloud that monitors your temperature sensor data. If your temperature is higher than a threshold value, the rules engine will turn on the LED on your Kitra board.

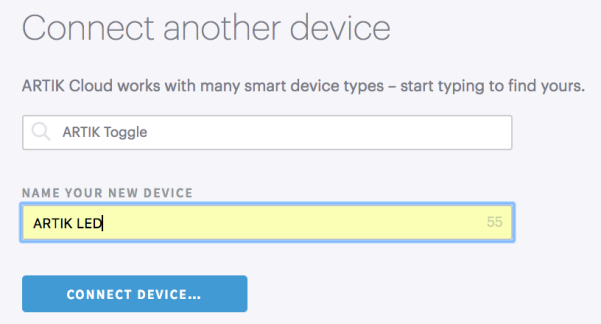
1. Create an LED device in the ARTIK Cloud user portal.

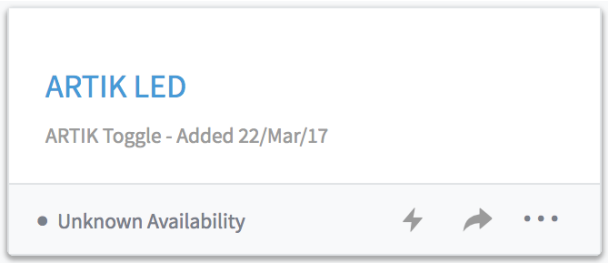
   1.1 From ARTIK Cloud user portal https://artik.cloud, go to "MY ARTIK CLOUD"/"DEVICES", click "Add Another Device…" link.



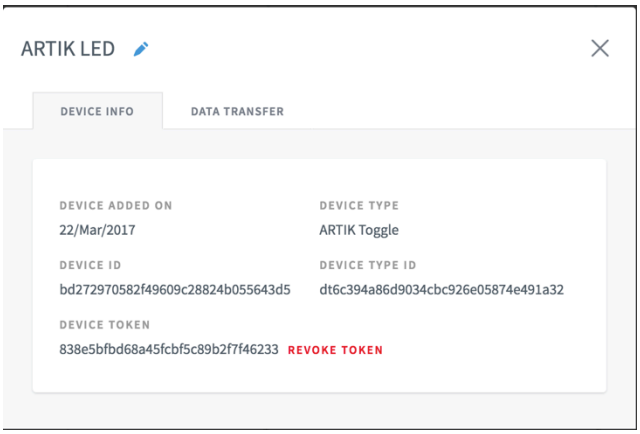
   1.2 Search for "ARTIK Toggle" and select it. This is the public device type that we created for this workshop. The device type includes one attribute "toggle", which is a Boolean type. Its valid value is either true or false, to represent if the LED is ON or OFF.

1.3 Change the device name to "ARTIK LED", then click the "CONNECT DEVICE…" button. A new device will be created.





1.4 Click on the device name, you will see the Device Info popup, which shows your Device Type, Device ID, Device name, and additional details. Click the "GENERATE DEVICE TOKEN…" link to generate a device token.

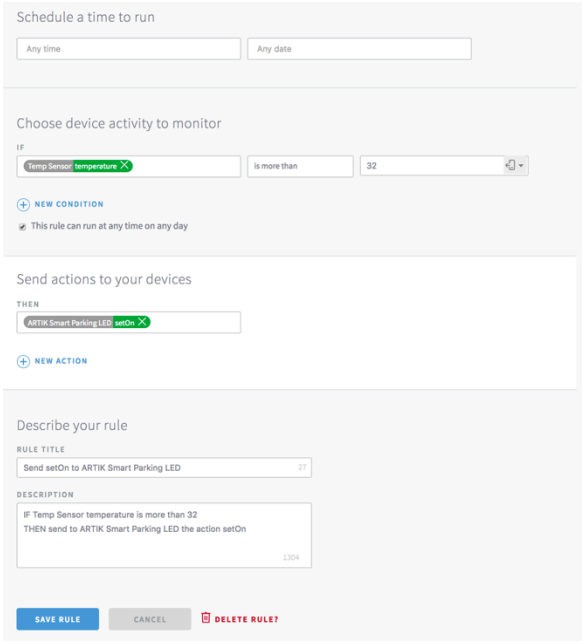


2. Add Rules to our Rules Engine. By doing this, we will be able to connect our "Temp Sensor" device and "ARTIK LED" device, and trigger cross-device actions.

2.1 In the ARTIK user portal, go to "RULES", and click the "CREATE A NEW RULE" button. The first rule we will add is to turn on the LED when temperature is higher than 32 degrees. You can create the rule by using plain English.

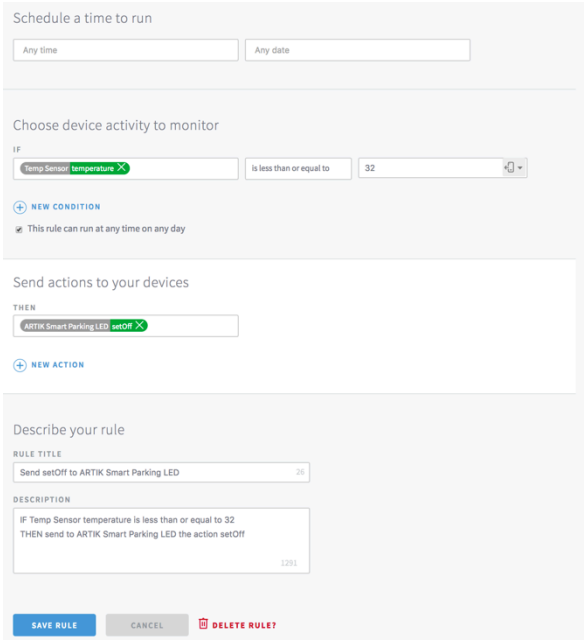
IF
"Temp Sensor" temperature is more than **32**
THEN
"ARTIK LED" setOn

Click the "SAVE RULE" button.



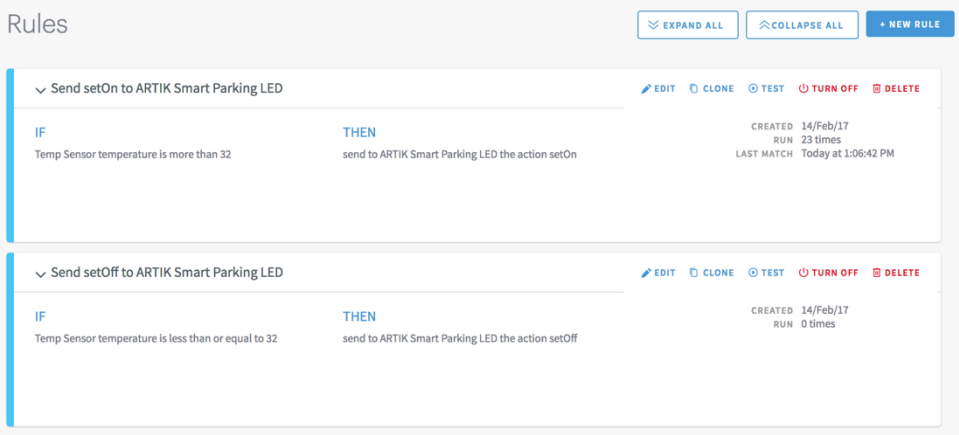
2.2 Repeat the above step, and create a second rule to turn off the LED when the reservation is cancelled or expired.

IF
"Temp Sensor" temperature is less than or equal to **32**
THEN
"ARTIK LED" setOff

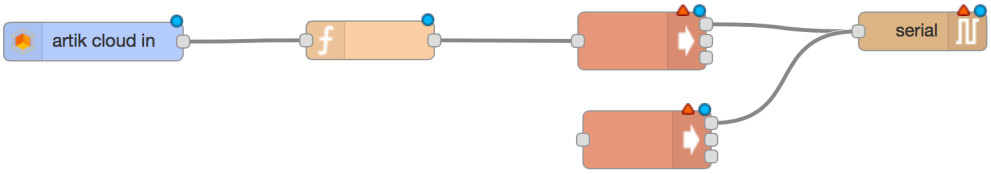


Click the "SAVE RULE" button.

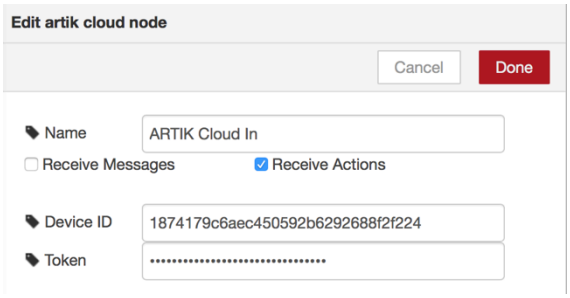
2.3 Now, we have 2 rules defined:



3. Listen for the triggered action.
    - Drag an “artik cloud” **in** node, a “function” node, two “exec” nodes and a serial **out** node to the canvas. Wire them up as below.



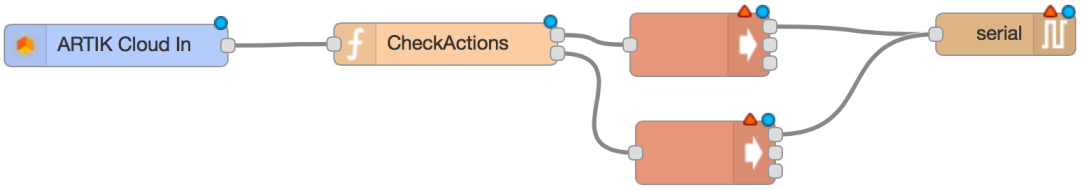
    - In “artik cloud” in node, select “Enable Actions”, enter your ARTIK LED device’s device ID/Token. Rename the node as “ARTIK Cloud In”.



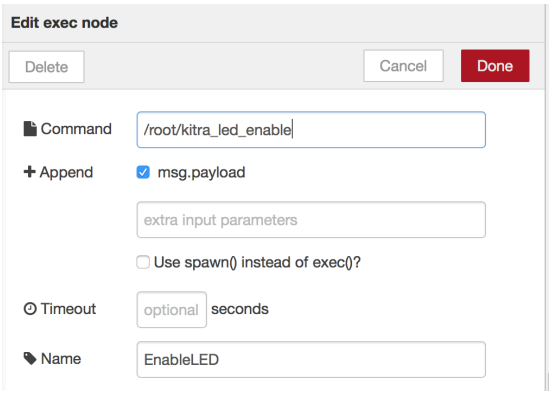
    - In the “function” node, increase the number of outputs to 2. We listen for incoming actions: When the incoming action is “setOn”, we go to the 1st output to turn on an LED. If the action is “setOff”, we go to the 2nd output to disable the LED. Rename the node to “CheckActions”.

```
var actions = msg.actions;
var action = actions[0].name;

if (action === 'setOn') {
    return [msg, null];
} else if (action === 'setOff') {
    return [null, msg];
}
```

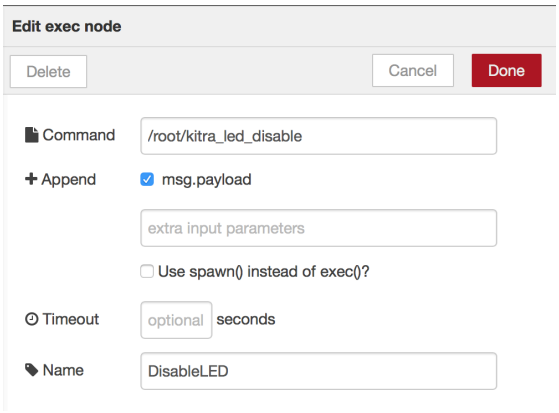
Connect the 2nd output of “CheckActions” node to the 2nd “Exec” node(the “Exec” node at the bottom).



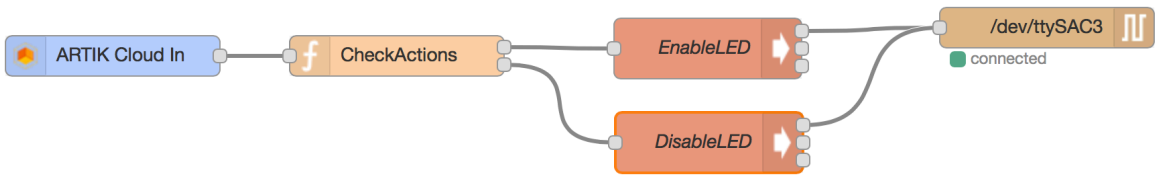
- In the top “Exec” node(the one connects to the 1st output of CheckActions node), configure Command as “/root/kitra_led_enable”, and change Name to “EnableLED”.



- In the bottom “Exec” node(the one connects to the 2nd output of CheckActions node), configure Command as “/root/kitra_led_disable”, and change Name to “DisableLED”.



Here is what the final flow looks like:



Now, when your temperature rises above 32 degrees, the LED on your board will be turned on by rules engine. When the temperature is less than or equal to 32 degrees, the LED will turn off.

Bonus Question 2: Publish message to ARTIK Cloud by using MQTT

In Exercise 3, we introduced how to use MQTT for device to device communication. MQTT can also be used for device to Cloud communication. Now, Can you change the flow in Exercise 4, step 2 to use MQTT to stream temperature sensor data to ARTIK Cloud?

Reference: ARTIK Cloud MQTT endpoint information can be found at https://developer.artik.cloud/documentation/data-management/mqtt.html.

Tips: *msg.topic* is the Publish Path, should be defined as `/v1.1/messages/<deviceID>`.