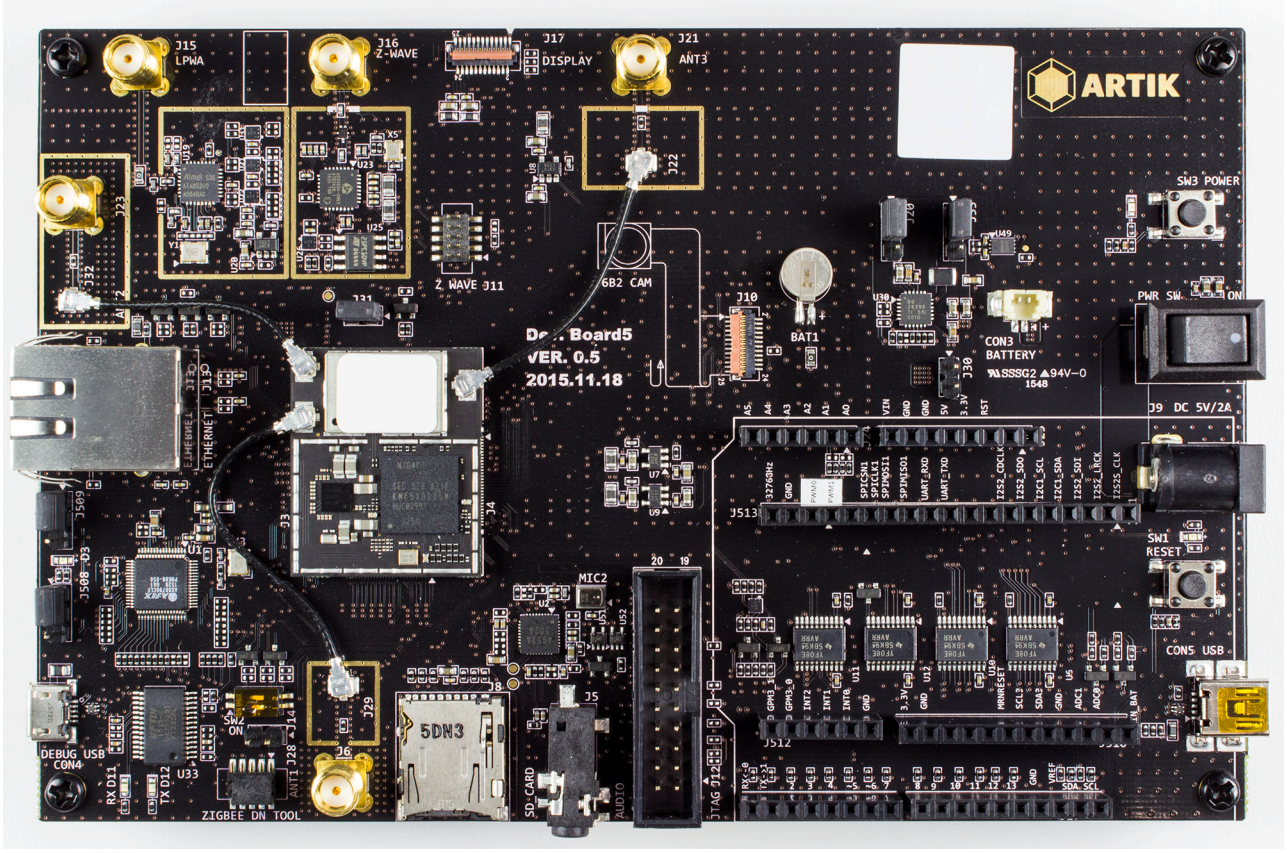
Develop IoT with Samsung ARTIK platform

April 26, 2016
Moscone West, San Francisco

Samsung Strategic and Innovation Center



SAMSUNG DEVELOPER CONFERENCE™

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 ARTIK board from the serial console
- Collect sensor data from ARTIK GPIO pins
- Use Temboo for Web service integration
- Use MongoDB for storing and retrieving your information
- Build program flow using Node-RED
- Enable MQTT so that your ARTIK boards can communicate with each other
- Connect to the Cloud

Before you begin

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

   PuTTY (http://www.putty.org/) – serial console
   Filezilla (https://filezilla-project.org/) – scp client for Windows

2. Have a Gmail account: ARTIK will send emails to your Gmail account. In order to do this, we need to enable 2-step verification in your Gmail account. If you want to keep your project emails separate from your personal emails, we suggest you create a Gmail account for this workshop.

3. Establish a Temboo account: We will use Temboo for Web services integration. Create an account at https://www.temboo.com/.

4. Establish a Samsung SAMIIO user portal account: We will stream data to Samsung Cloud service. Create an account at Samsung SAMIIO user portal (https://portal.samsungsami.io/).

5. Bring your cell phone. When we enable 2-step verification in your Gmail account, Google will text message you an activation code.

Setup

1. Access ARTIK from your serial console.

   Windows users: https://developer.artik.io/documentation/getting-started-beta/communicating-pc.html
   Mac users: https://developer.artik.io/documentation/getting-started-beta/communicating-mac.html
   Linux users: https://developer.artik.io/documentation/getting-started-beta/communicating-linux.html

2. Connect ARTIK to the network.

2.1 Ethernet. Plug an Ethernet cable into your ARTIK Ethernet port, do a 'ping' test from your console and take note of your board IP address.

```
[root@localhost ~]# ping www.google.com
PING www.google.com (172.217.3.36) 56(84) bytes of data.
64 bytes from nuq04s18-in-f4.1e100.net (172.217.3.36): icmp_seq=1 ttl=52 time=8.83 ms
64 bytes from nuq04s18-in-f4.1e100.net (172.217.3.36): icmp_seq=2 ttl=52 time=59.9 ms
64 bytes from nuq04s18-in-f4.1e100.net (172.217.3.36): icmp_seq=3 ttl=52 time=17.2 ms
```

(Ctrl-C to terminate)

```
[root@localhost ~]# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.0.0.23  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
```

2.2 WiFi.(Optional) In place of Ethernet, set up WiFi on your ARTIK, and do the 'ping' test as noted above.

https://developer.artik.io/documentation/developer-guide/configuring-wifi-on-artik-10.html - configuring-wifi-on-artik-5-and-10

Exercise 1: Monitor your trash level (Node-RED)

1. Wire up your distance sensor to your ARTIK board. ARTIK 5 has two analog pins exposed: J24 A0 and J24 A1. There are 3 wires on your distance sensor:
   - 5V (red wire) connects to header J25 5V
   - GND (black wire) connects to header J25 GND
   - Signal (blue or yellow wire) connects to header J24 A0

2. Read the analog GPIO pin using *sysfs*, which allows sensor data to be read from a system file. To get sensor data on J24 A0, copy the highlighted text below to your terminal and hit [Enter].

```
[root@localhost ~] # cat  /sys/devices/126c0000.adc/iio:device0/in_voltage0_raw
1413
```

   Move your hand towards or away from the distance sensor, and repeat the command above by typing [Up] and then [Enter]. You should observe that the output value changes.

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

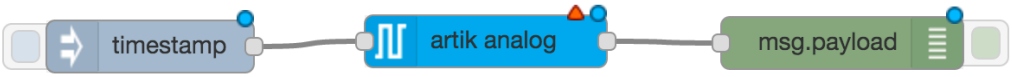
```
[root@localhost ~]# node-red &
…
Welcome to Node-RED
===================

12 Feb 13:53:43 - [info] Node-RED version: v0.13.1
12 Feb 13:53:43 - [info] Node.js  version: v0.10.36
…
12 Feb 13:53:48 - [info] Server now running at http://127.0.0.1:1880/
```

4. Open a browser on your laptop, and launch http://<your_board_ip_address>:1880/

5. Design your first project flow.

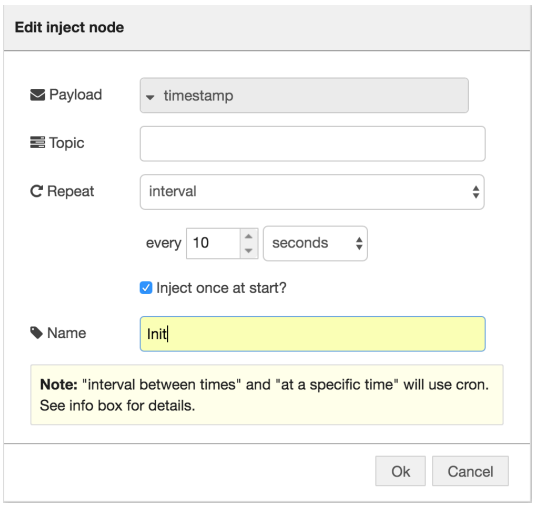
5.1 Create a flow on the Node-RED canvas.

- Drag an “inject” input node from the node palette to the canvas (it initially shows “timestamp”).
- Drag an “artik analog” node to the right of the first node.
- Drag a “debug” output node to the right of the second node.
- Connect these 3 nodes by dragging a “wire” from the right side of the “inject” node to the left side of the “artik analog” node, then from the right side of “artik analog” node to the left side of the “debug” node.

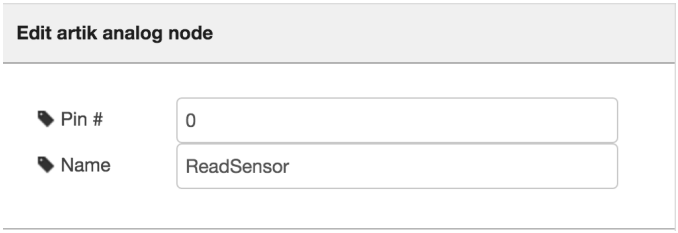


5.2 Configure each node by double-clicking.

- “inject” node: Set Repeat to a 10s interval, enable ‘Inject once at start’ and rename the node to “Init”. Your flow will read the sensor data when it is first launched, and continue to read it every 10 seconds.



- “artik analog” node: Set Pin to 0. Name as “ReadSensor”.



5.3 Run the flow.

Click 

at the upper right corner to make your application live. You should be able to see your sensor data updated every 10 seconds in the Debug tab.

Exercise 2: Make your ARTIK talk to you (Temboo, Node-RED & MongoDB)

1. Let ARTIK send an email when the trash can is full (use your hoodie to fill up the trash can).

1.1 Follow https://www.temboo.com/hardware/samsung/send-an-email to generate code for ARTIK platform. Since you already have a Gmail account created, we will start from step 6.

When you reach Temboo's tutorial step 16, Windows users can use Filezilla to transfer *sendemail.zip* from the host machine to ARTIK. You need to use port 22 for the file transfer.

From step 17, follow our tutorial below.

1.2 From the ARTIK command line, run the following commands to tweak the code and compile it. By default, auto-generated Temboo code runs in a loop, and executes the main logic 10 times (defined by MAX_RUNS variable). Since we only want to send one email every time we fill up the trash can, we need to change the value of the MAX_RUNS variable to 1. This is done by running the 'sed –i' command below.

```
[root@localhost ~]# cd /root
[root@localhost ~]# unzip sendemail.zip
[root@localhost ~]# cd sendemail
[root@localhost ~]# sed -i 's/MAX_RUNS = 10/MAX_RUNS = 1/g' sendemail.c
[root@localhost ~]# gcc -L/root/temboo_artik_library/lib -ltemboo -I/root/temboo_artik_library/include
sendemail.c -o sendemail
```

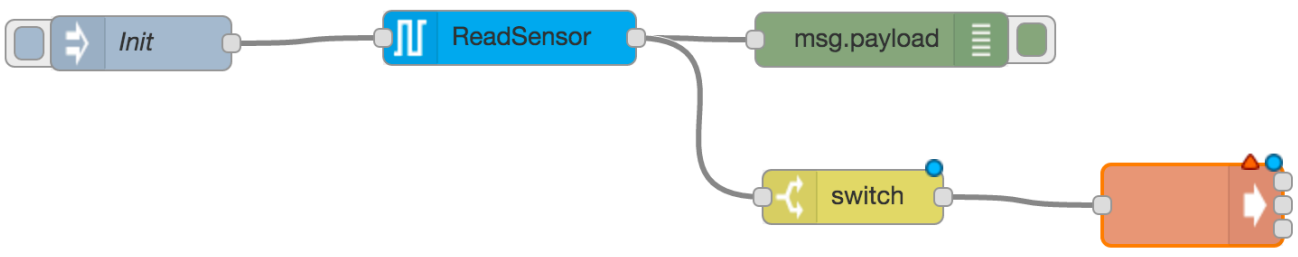
1.3 Launch 'sendemail' from the command line to see if you can get an email.

```
[root@localhost ~]# ./sendemail
```

2. Update the Node-RED flow to add Temboo Web service code generated above.

2.1 Extend the existing Node-RED flow to include Temboo email services.

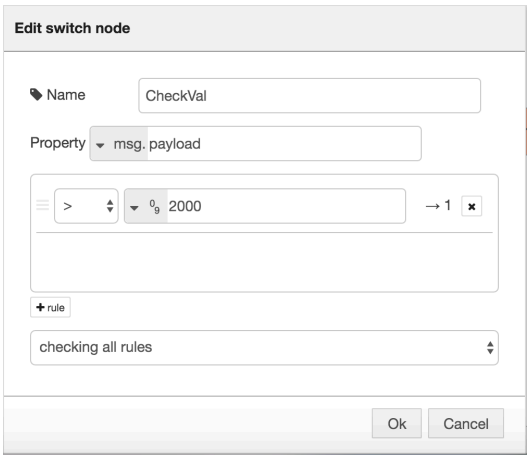
- Drag a "switch" function node from the node palette; drop it under the debug node on the canvas.
- Drag an "exec" node to the right of the switch node.
- Connect the nodes as shown.



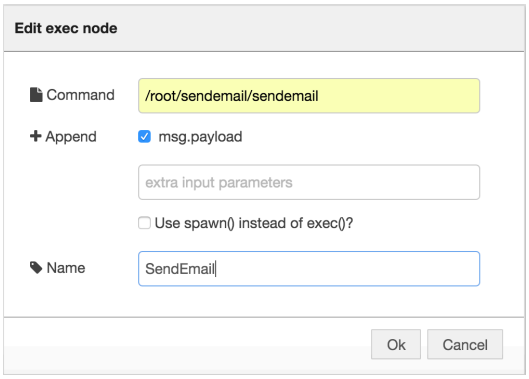
2.2 Configure each node by double-clicking.

- "Switch" node: Add our first rule.

  IF "msg.payload > 2000 (**a number**)" THEN "follow Rule 1".
  Rename to "CheckVal". This node decides whether the trash can is filled up or still empty.



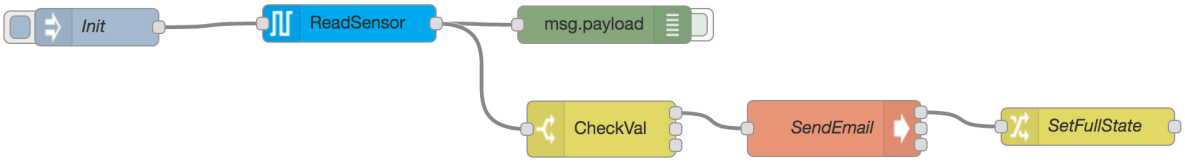
- "Exec" node: Enter "/root/sendemail/sendemail" as the Command we are going to execute. Name as SendEmail.



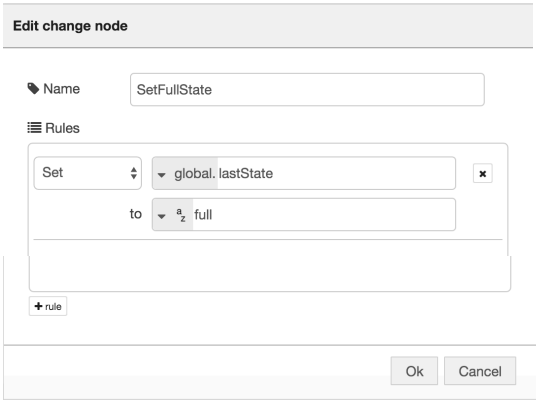
2.3 Deploy your flow. You should see that you get an email when you fill up the trash can.

3. Set up MongoDB to track when the trash can state goes from Full to Empty. We will mark the date/time the trash can is emptied in our backend MongoDB, introducing a global variable to keep track of our trash can state.

3.1 Drag a “Change” node to the right of the “SendEmail” node, and wire it up.

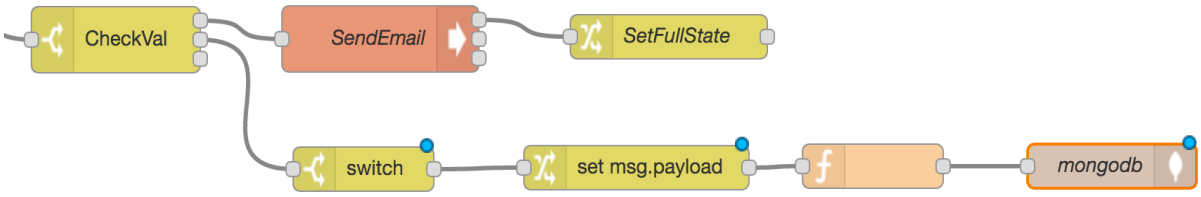


- Double click the node and define global.lastState = “full” (a string value). Rename to “SetFullState”.



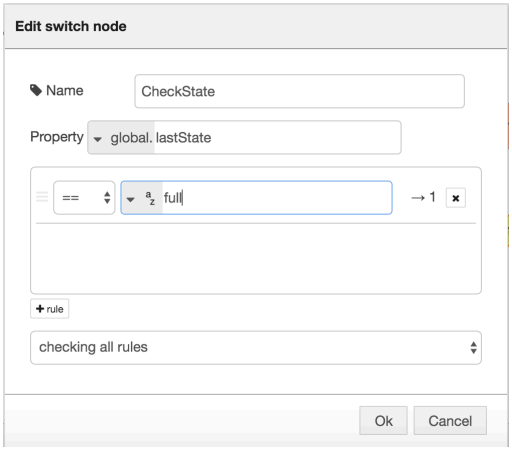
- Again double-click the “CheckVal” switch node, to add a 2nd rule.
  IF “msg.payload <= 2000 (**a number**)” THEN “follow Rule 2”.

3.2 Append “switch”, “change”, “function”, and “mongodb output” nodes to Rule 2, and wire them up.

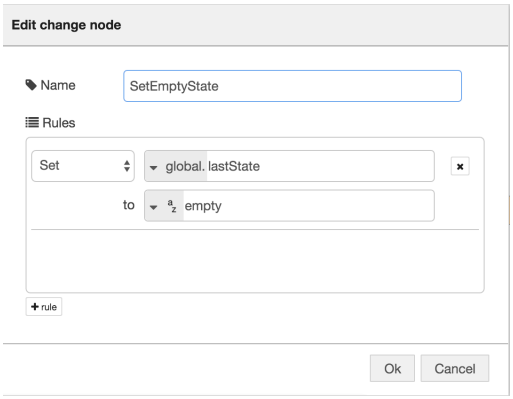


When Rule 2 is satisfied, our trash can is empty. So we will check if the last state of the can was “full”; if so, we update it to “empty”. In the meantime, we mark the trash can name, trash can location, and date/time the trash can is being emptied in our backend MongoDB.

- “switch” node: Check IF global.lastState == “full”. Rename to “CheckState”.



- “change” node: Change global.lastState to “empty”. Rename to “SetEmptyState”.

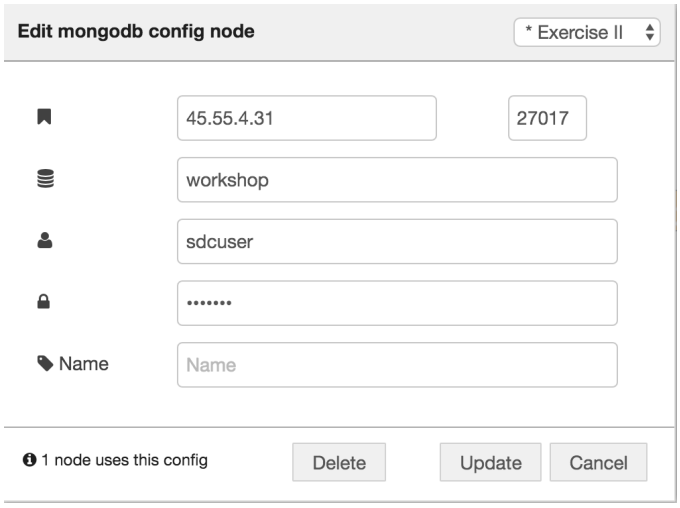


- "function" node: Here, we format our msg.payload. There are 3 fields in our database collection: name, address and date/time. Replace name and address with your own when you add your function. Here is what I have in mine:

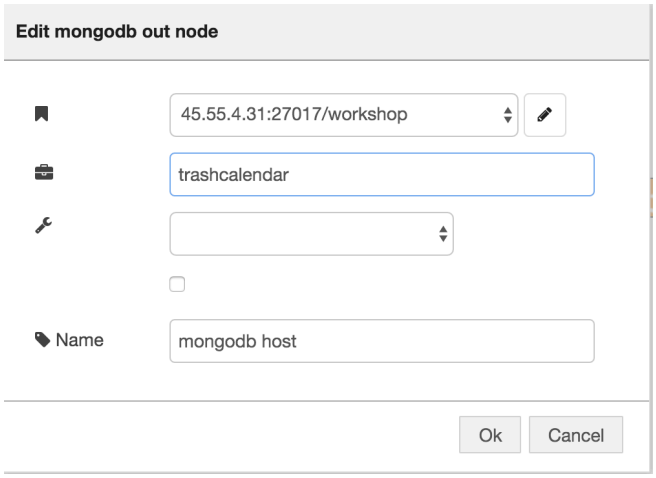
```
msg.payload = {
  "name":"wei",
  "address":"3665 N 1st, San Jose, CA",
  "date": Date()
}
return msg.payload;
```

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

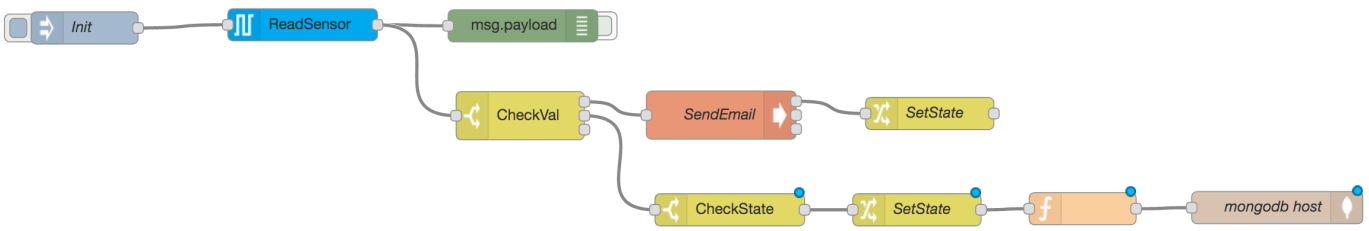
  - Click  in the "Edit mongodb out node"dialog, "Edit mongodb config node" opens up. Enter "45.55.4.31" as host ip address, "27017" as port number, and "workshop" as database name. Use "sdcuser" and "sdc2016" as your username and password.



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



This is what the flow looks like now.



4. Query MongoDB database.

   We have the MongoDB shell pre-installed on your board, so we can use shell commands to query MongoDB database. Commands in bold are what you need to enter in the following examples.

4.1 Connect to the database

```
[root@localhost ~]# mongo --host 45.55.4.31 -u "sdcuser" -p "sdc2016" --authenticationDatabase "workshop"
MongoDB shell version: 2.6.11
connecting to: 45.55.4.31:27017/test
> use workshop
switched to db workshop
```

4.2 Query the entries in our trashcalendar document:

```
> db.trashcalendar.find()
{ "_id" : ObjectId("570c8ed040d640f70d77fbae"), "name" : "wei", "address" : "3665 N 1st, San Jose, CA", "date" : "Tue Apr 12 2016 01:59:44 GMT-0400 (EDT)", "_msgid" : "abc79cdc.54386" }
{ "_id" : ObjectId("570c8fa940d640f70d77fbaf"), "name" : "dan", "address" : "665 Clyde Ave, Mountain View, CA", "date" : "Tue Apr 12 2016 02:03:21 GMT-0400 (EDT)", "_msgid" : "bbd59bc4.442a68" }
{ "_id" : ObjectId("570c902040d640f70d77fbb0"), "name" : "andrew", "address" : "735 Battery St, San Franciso, CA", "date" : "Tue Apr 12 2016 02:05:20 GMT-0400 (EDT)", "_msgid" : "2882af79.d77d5" }
{ "_id" : ObjectId("570c90c440d640f70d77fbb1"), "name" : "shan", "address" : "2665 N. 1st street, San Jose, CA", "date" : "Tue Apr 12 2016 02:08:04 GMT-0400 (EDT)", "_msgid" : "5ae7fa17.a51804" }
```

4.3 Query the trash can status at a specific address:

```
> db.trashcalendar.find({"address":"3665 N 1st, San Jose, CA"})
{ "_id" : ObjectId("5706cc28b69a84c312a0daf6"), "name" : "samsung", "address" : "3665 N 1st, San Jose, CA", "date" : "Thu Apr 07 2016 17:07:52 GMT-0400 (EDT)", "_msgid" : "8030752b.7fcf88" }
{ "_id" : ObjectId("570c8ed040d640f70d77fbae"), "name" : "wei", "address" : "3665 N 1st, San Jose, CA", "date" : "Tue Apr 12 2016 01:59:44 GMT-0400 (EDT)", "_msgid" : "abc79cdc.54386" }
```

4.4 Use a logical OR for a list of query conditions by using the $or query operator.

```
> db.trashcalendar.find({$or:[{"name":"wei"},{"name":"shan"}]})
{ "_id" : ObjectId("570c8ed040d640f70d77fbae"), "name" : "wei", "address" : "3665 N 1st, San Jose, CA", "date" : "Tue Apr 12 2016 01:59:44 GMT-0400 (EDT)", "_msgid" : "abc79cdc.54386" }
{ "_id" : ObjectId("570c90c440d640f70d77fbb1"), "name" : "shan", "address" : "2665 N. 1st street, San Jose, CA", "date" : "Tue Apr 12 2016 02:08:04 GMT-0400 (EDT)", "_msgid" : "5ae7fa17.a51804" }
```

Bonus Question:

Twilio APIs can help us further extend this application to include voice and SMS functionality. We have installed a Twilio output node for you on your ARTIK. From your Node-RED flow, can you send a text message to your phone when the trash can is full?

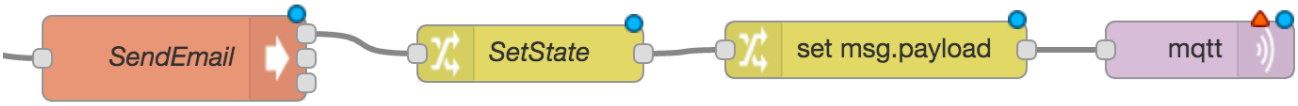
Exercise 3: Add your trash can to a trash can network (Node-RED)

1. Launch MQTT broker 'mosquitto' on one of the boards in your group, and let your group members know your board's IP address.

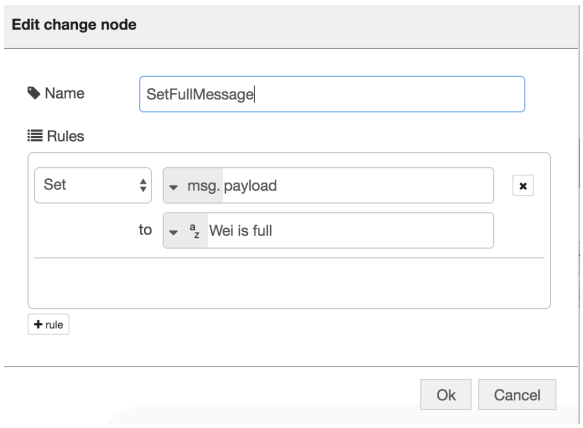
   #mosquitto -d

2. Send MQTT Messages when your trash can is full.

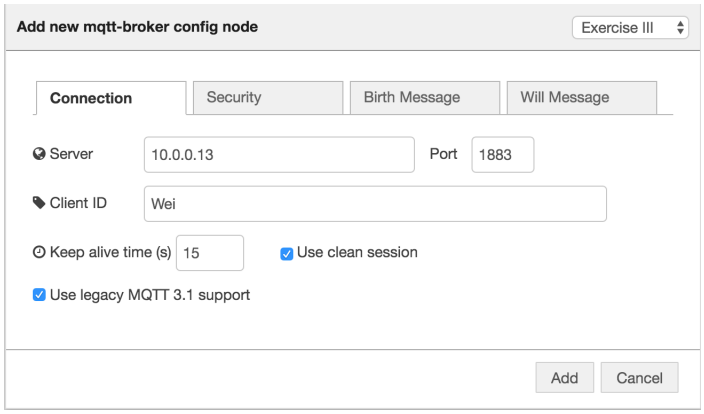
2.1 To the "SendEmail" path, add "change" and "mqtt output" nodes, and wire them up.



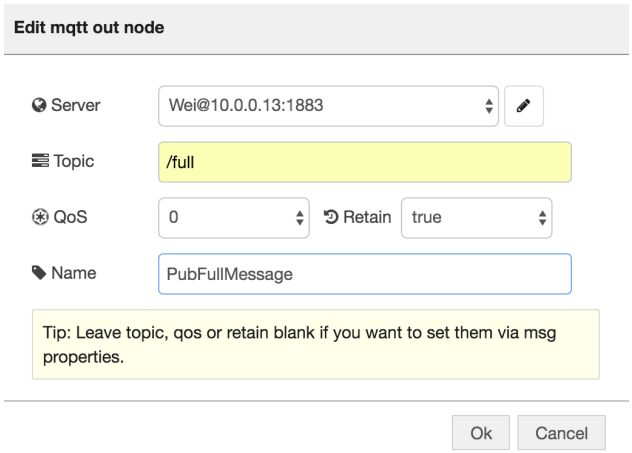
- "change" node: Set the msg.payload to be "*<your name>* is full". Rename to "SetFullMessage".

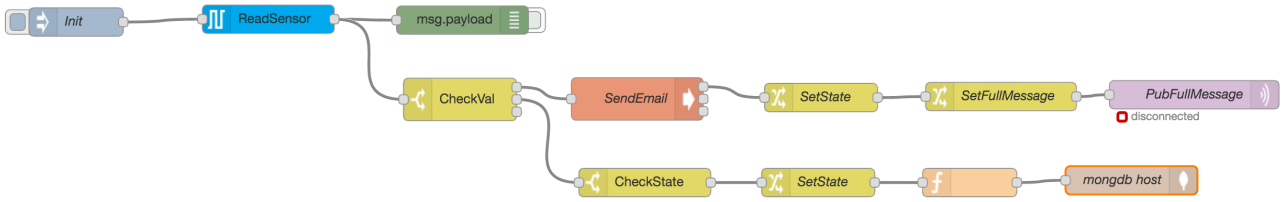


- "mqtt" output node: Edit MQTT broker IP address



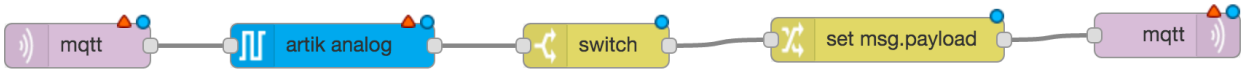
- "mqtt" output node (cont'd): Define the Topic as "/full". Rename to "PubFullMessage".



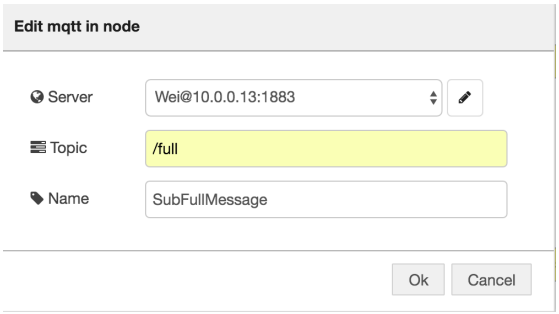


3. Subscribe to MQTT "/full" messages. We will create a 2nd flow for this. When your ARTIK receives a "/full" message from other trash cans in the group, it will check its own trash level, then if it is still empty, publish the "/empty" message to the broker.

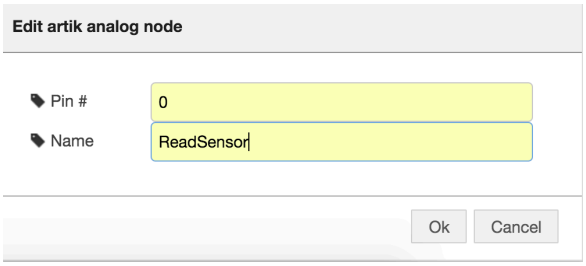
3.1 Add and wire up "mqtt input", "artik analog", "switch", "change" and "mqtt output" nodes.



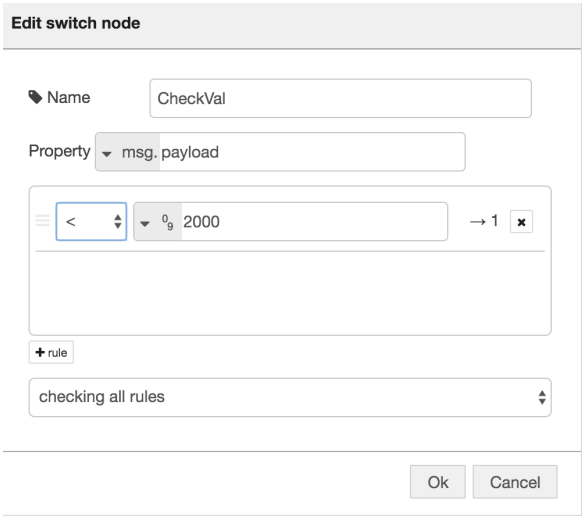
- "mqtt" input node: Set the Server address to the IP address of your broker. Change Topic to "/full", in order to listen to "/full" messages in the network.



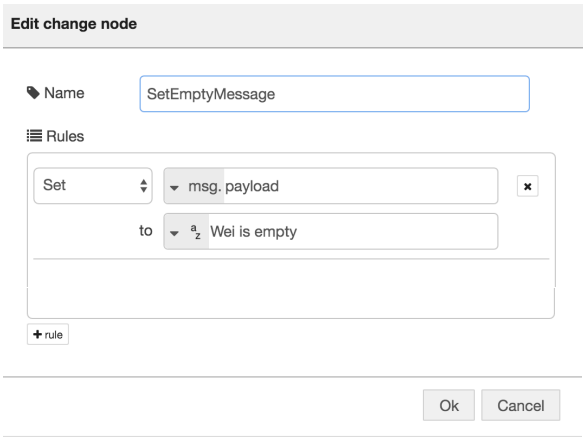
- "artik analog" node: Listen to Pin 0. Rename to "ReadSensor" (same as in the first flow).



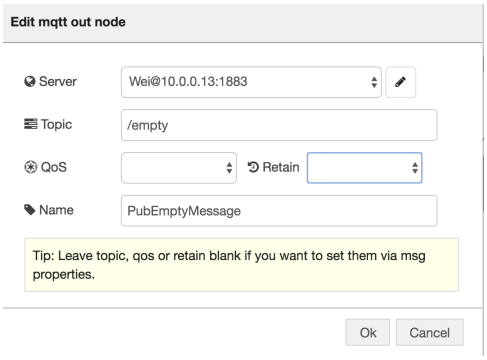
- "switch" node: Check IF "msg.payload < 2000(number)" THEN "follow Rule 1". Rename to "CheckVal".



- "change" node: Set the msg.payload to be "*<your name>* is empty".



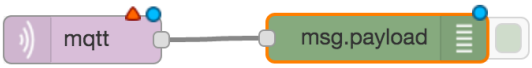
- "mqtt" output node: Edit MQTT broker IP address, and define the Topic as "/empty".



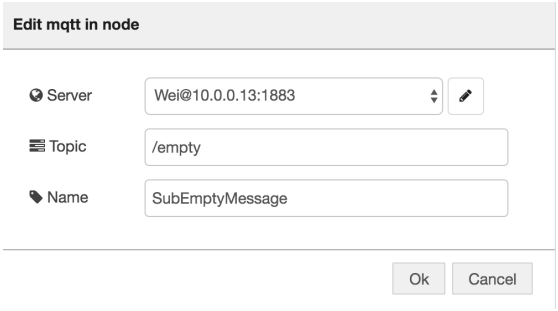
Here is the final look of our 2nd flow:

4. Subscribe to MQTT "/empty" message: Now we are creating a 3rd flow to listen for "/empty" message.

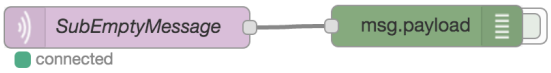
4.1 Add "mqtt" input and "debug" nodes, and wire them up.



- "mqtt" input node: Set the Server address to the IP address of your broker. Change Topic to "/empty". Rename to "SubEmptyMessage".



Here is what the 3rd flow looks like:

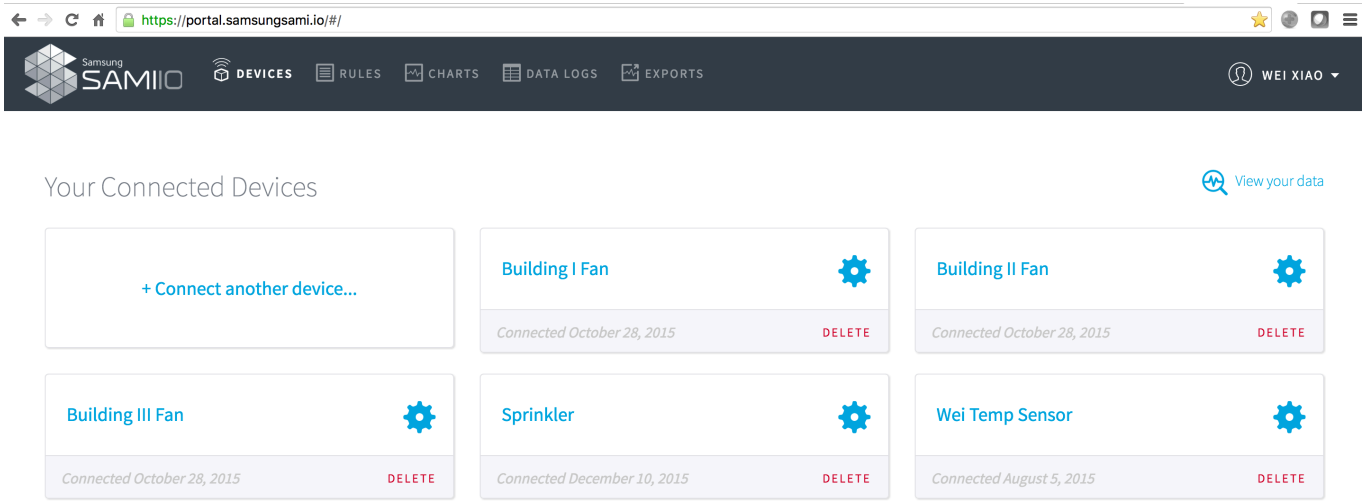


5. Deploy your flows, and enjoy your Smart Trash Can Network!

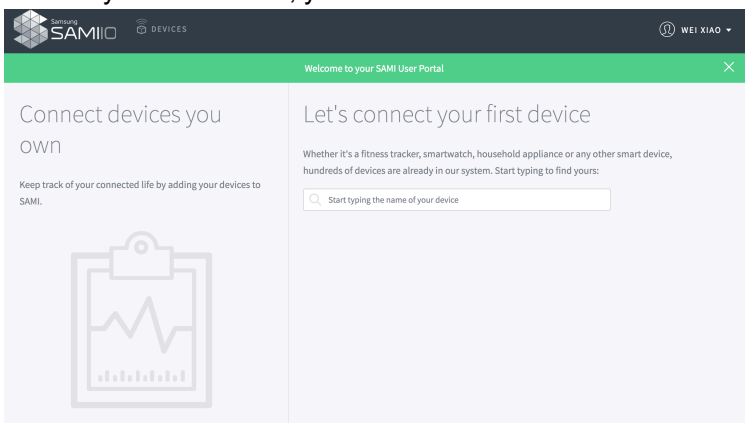
Exercise 4: Connect to Samsung Cloud (Node-RED, Samsung Cloud)

In this exercise, we are going to use Samsung Cloud as our MQTT broker.

1. Log into SAMIIO user portal.

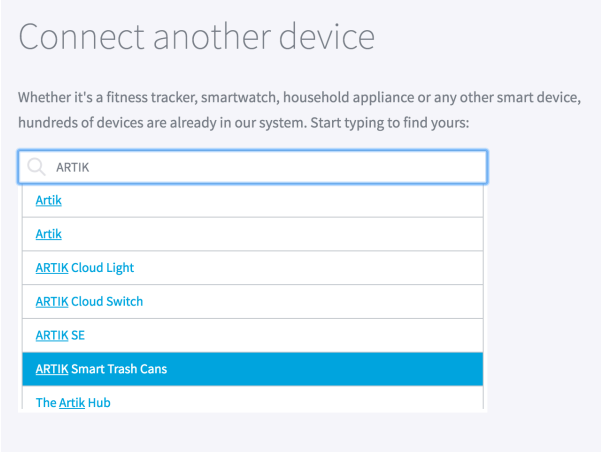


2. If this is your first device, you will be re-directed to the screen below:

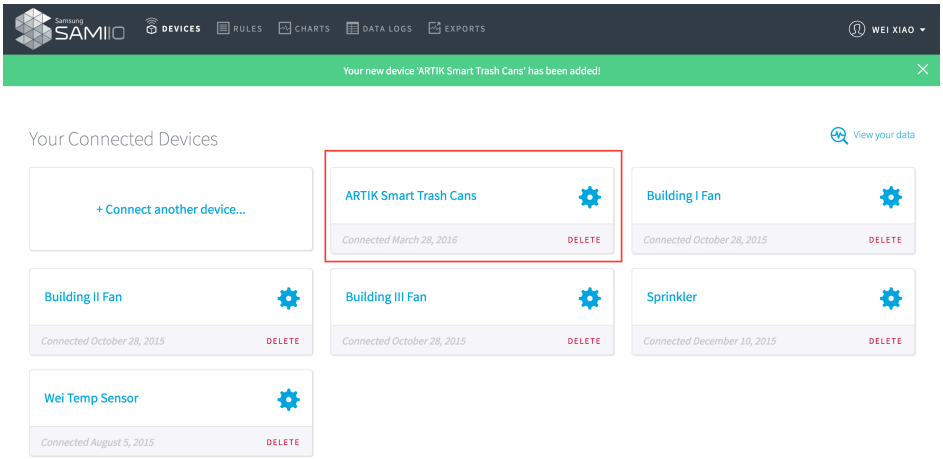


Otherwise, Click “Connect another device…” link.

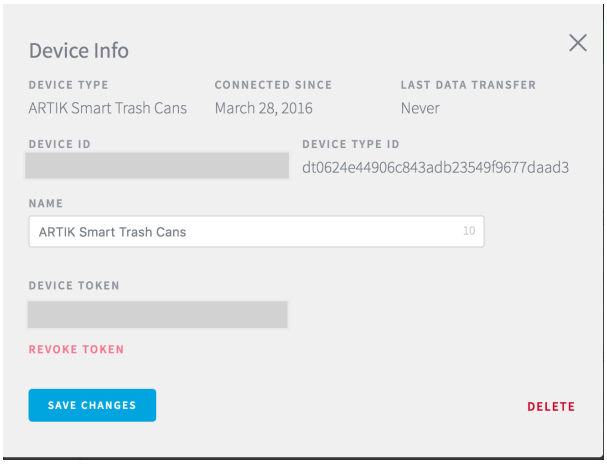
Search for “ARTIK Smart Trash Cans” and select it. This is a public device we created for the workshop, which includes “state” (“full” or “empty”) and “address” of the trash can.



3. Use the default device name “ARTIK Smart Trash Cans”, then click the “CONNECT DEVICE…” button. A new device will be created.



4. Click the Gear icon next to your device name, you will see the Device Info popup, which shows your Device Type, Device ID, Device name etc. details. Click the “GENERATE DEVICE TOKEN…” link to generate a device token.



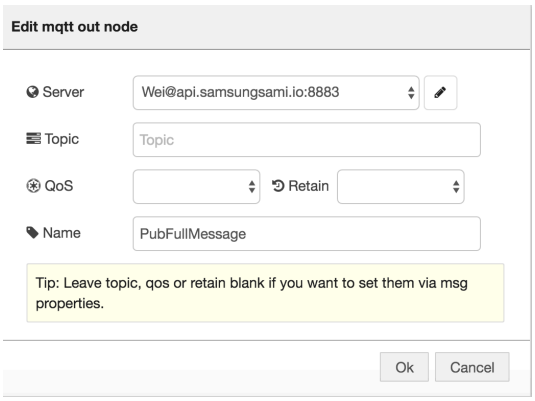
We need to use the Device ID and Device Token to associate our trash can with Samsung Cloud. Leave the window open.

5. In our first flow “SendEmail” path, we will insert a function node between “SetFullMessage” node and “PubFullMessage” node. In the function, we are converting our message payload format for “ARTIK Smart Trash Cans” device type. Replace the highlighted part of msg.topic below with your own Device ID (from step above). Click “OK” to close the dialog.

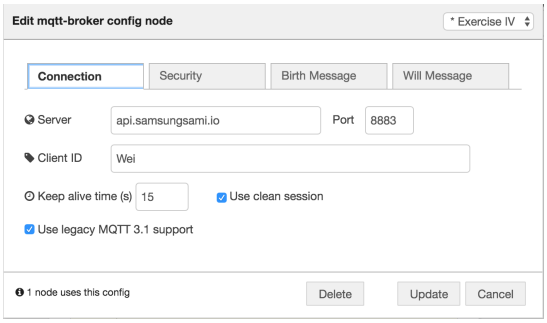
```
msg.topic = "/v1.1/messages/2e44708377cb410985321064d7dda52e"
msg.payload = {
   "state":"full",
   "address":"3665 N 1st, San Jose, CA"
}
return msg;
```

6. Configure the “PubFullMessage” node to use Samsung Cloud’s MQTT server.

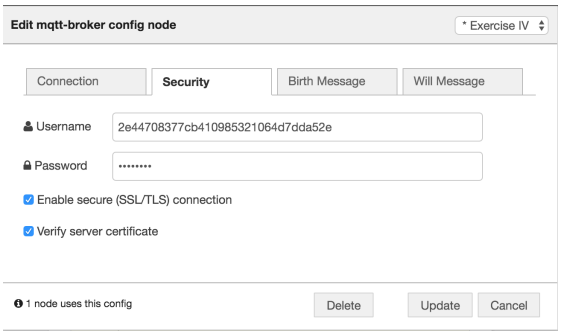
- Click  in the “Edit mqtt out node” dialog.



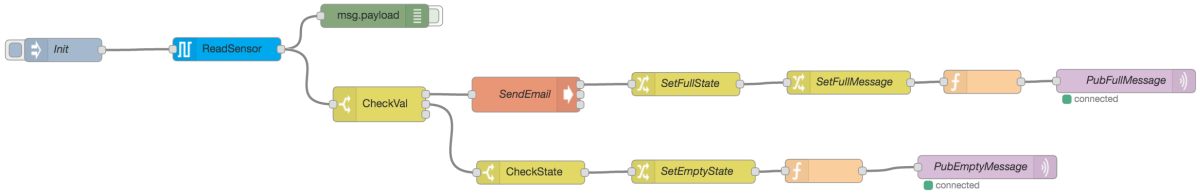
- “Edit mqtt-broker” config node: Use “api.samsungsami.io” as the Server address, 8883 as the port number.



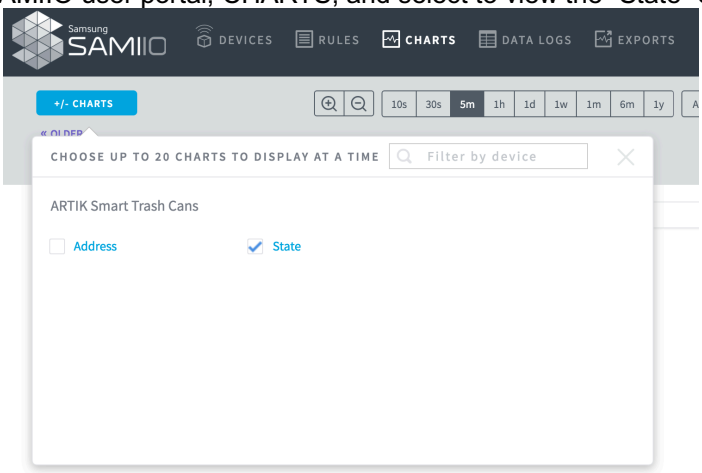
- “Security” Tab: From the Device Info window, copy and paste your Device ID as Username, and Device Token as Password. Make sure “Enable secure (SSL/TLS) connection” is checked.



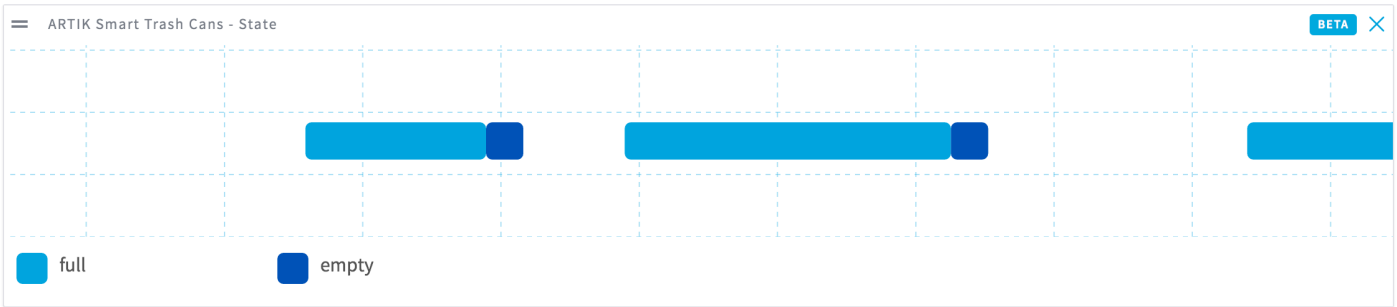
This is what your final flow should look like:



Go to SAMIIO user portal, CHARTS, and select to view the “State” of your ARTIK Smart Trash Cans.



You will see the streamed trash can states in your SAMI portal:



Bonus Question:

How can you monitor the state of other smart trash cans in the workshop?