Develop IoT with Samsung ARTIK platform

Oct 20th, 2016, IoT Tech Expo, Santa Clara

Samsung Strategic and Innovation Center

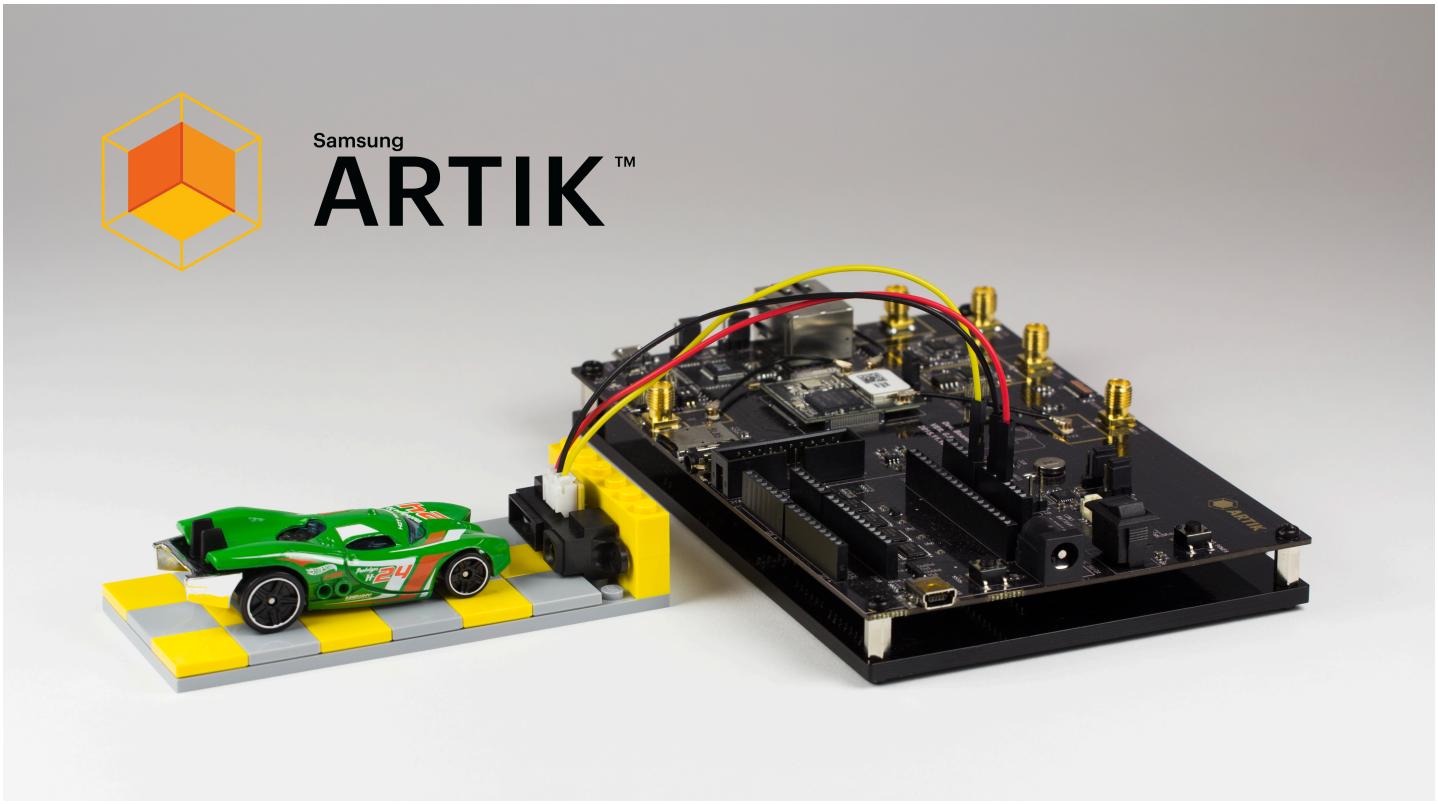


Table of Contents

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 ARTIK board from the serial console
- Collect sensor data from ARTIK GPIO pins
- 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
- Stream data to ARTIK Cloud
- Use ARTIK Cloud Rules Engine to trigger cross-device actions

Before you begin

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

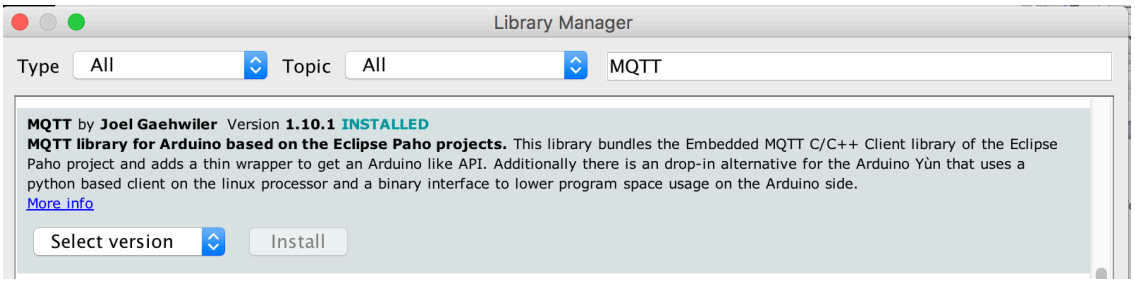
   PuTTY (http://www.putty.org/) – SSH and Telnet client, for serial console access
   Filezilla (https://filezilla-project.org/) – scp client for Windows

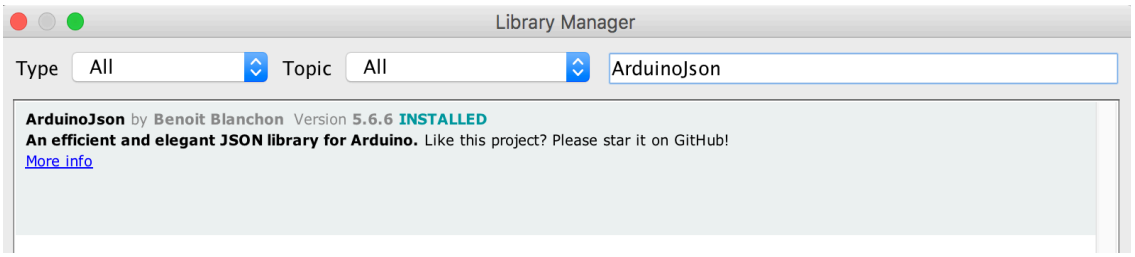
2. Have a Gmail account: ARTIK will send emails to your Gmail account. In order to do this, we need to turn on "Access for less secure apps" setting in your Gmail. If you want to keep your project emails separate from your personal emails, we suggest you create a Gmail account for this workshop.

3. Establish 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/).

4. Install Arduino IDE: Please follow our installation guide at https://developer.artik.io/documentation/developer-guide/ide/arduino.html to set up Arduino IDE on your laptop.

   After installation, launch Arduino IDE, go to Sketch->Include Library ->Manage Libraries. Install "MQTT" library(MQTT library for Arduino based on the Eclipse Paho projects) and "ArduinoJson" library.





Setup

1. Access ARTIK from your serial console. Instructions is at https://developer.artik.io/documentation/getting-started-beta/communicating-pc.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 10.0.0.2  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
```

   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 the state of your parking space (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 (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 car 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/

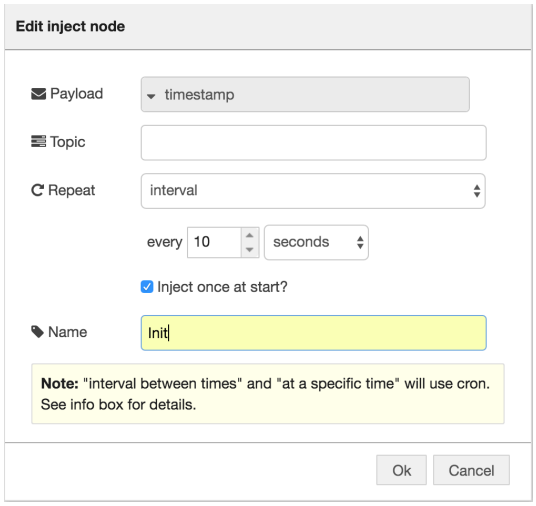
5. Design your first project flow.

5.1 Create a flow on the Node-RED canvas.

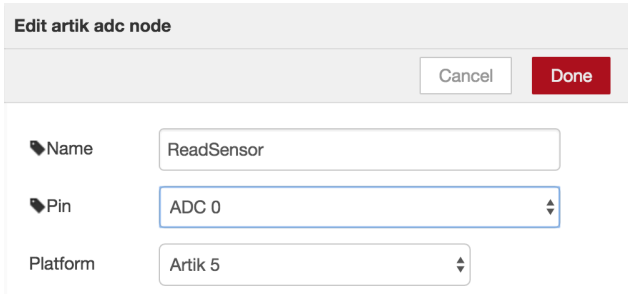
- Drag an “inject” input node from the node palette to the canvas (it initially shows “timestamp”).
- Drag an “artik adc” node to the right of the first node.
- Drag a “debug” output node to the right of the second node.
- Connect these 3 nodes by dragging a “wire” from the right side of the “inject”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

5.2 Configure each node by double-clicking.

- “inject” node(shows as “timestamp”): 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 adc” node: Select “ARTIK 5” as the target platform and set Pin to “ADC 0”. Rename the node as “ReadSensor”.

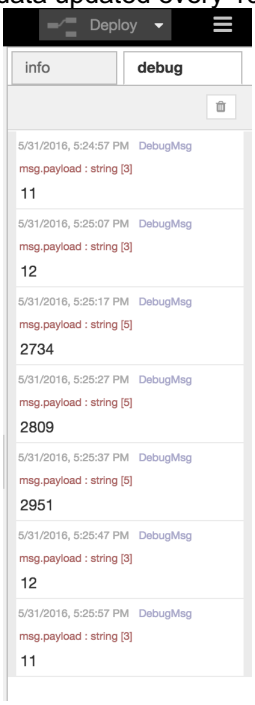


- “debug” node: Name as “DebugMsg”. Here is our final flow.

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: Track Parking Lots Occupancy and get email notification for parking violation (Node-RED & MongoDB)

In exercise 2, we will monitor if our parking space is occupied. When a lot is occupied, we will start a timer to keep track of your parking time to avoid violation. When the timer interval elapses and timer fires, ARTIK will send you an email alert. In the meantime, ARTIK streams parking space states to backend MongoDB so we can monitor real-time occupancy of our parking lots.

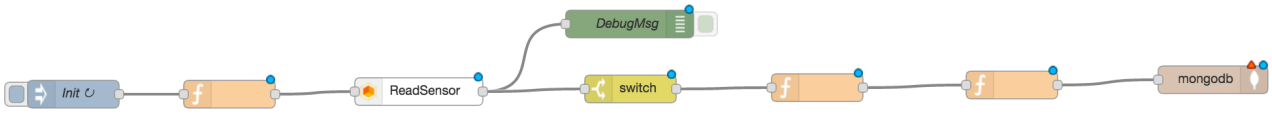
1. Use MongoDB to track the parking space occupancies.

We will mark our parking lot number, space number, space state and if there is a parking violation in our backend MongoDB. In order to do this, we introduce a global variable *global.lastState* to keep track of our latest parking space state, and a global variable *global.expired* to check if there is a parking violation in our space.

1.1

- Drag a “function” node, and place it between ‘Init’ node and ‘ReadSensor’ node.
- Drag a “switch” function node from the node palette; drop it under the debug node on the canvas.
- Drag two “function” nodes to the right of the switch node.
- Drag a “mongodb” **output** node 

to the rightmost.
- Connect the nodes as shown.

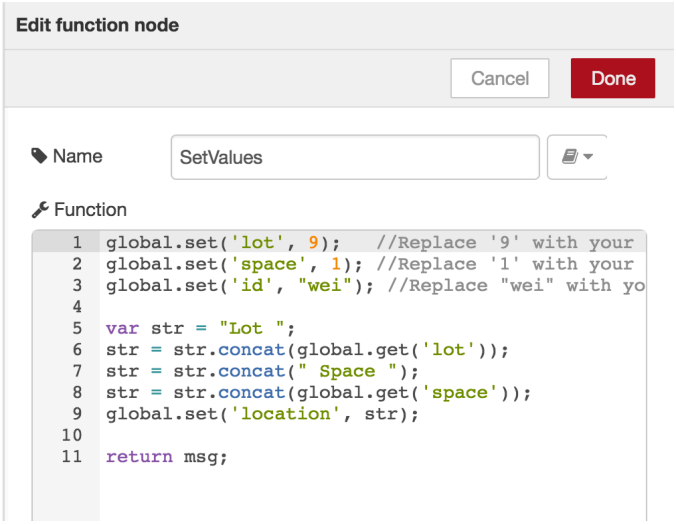


1.2 Configure each node by double clicking.

- In the “function” node in the middle of “Init” and “ReadSensor” nodes: Set our lot# and space#.

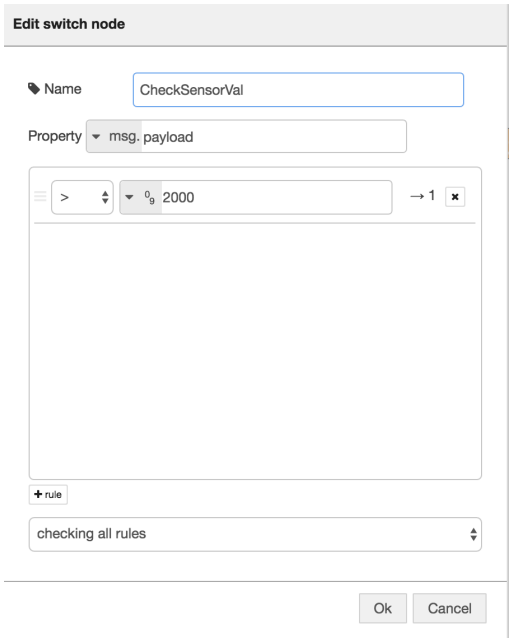
  Define our function as below, and rename the node to “SetValues”. Please replace the lot#, space# and id with your own. Make sure your id is unique. We will use the id as the unique identifier for MongoDB.

```
global.set('lot', 9);   //Replace '9' with your own lot#
global.set('space', 1); //Replace '1' with your own space#
global.set('id', "wei"); //Replace "wei" with your own UNIQUE id

var str = "Lot ";
str = str.concat(global.get('lot'));
str = str.concat(" Space ");
str = str.concat(global.get('space'));
global.set('location', str);

return msg;
```



- "Switch" node: Add our first rule.

  IF "msg.payload > 2000 (**a number**)" THEN "follow Rule 1".
  Rename to "CheckSensorVal". This node decides whether the parking space is occupied or still empty.



- In the first function between "CheckSensorVal" and "mongodb" nodes, we initialize our global variables. Here, we check if global.expired and global.lastState have been defined. If not, we define them and initialize them to "false" and "empty" respectively.
  Rename to "SetParkingTimer".

```
var expired = global.get('expired')||false;
var lastState = global.get('lastState')||"empty";
return msg;
```

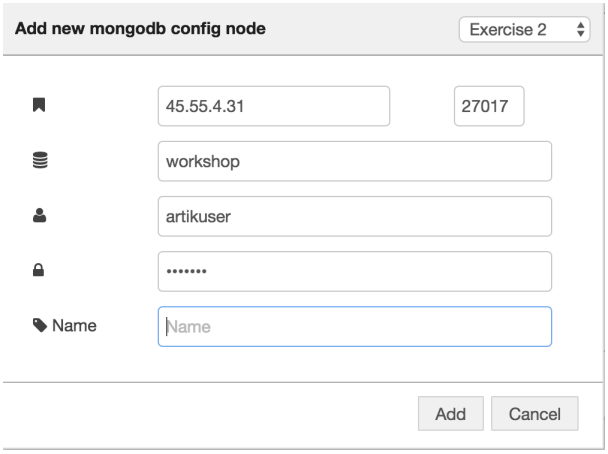
- In the second function between “CheckSensorVal” and “mongodb” nodes, we format the message payload for MongoDB. There are 4 fields in our database collection: Lot#, Space#, Space occupancy state(“full”) and if there is a parking violation in this space(“yes” or “no”). Here is the code:

```
var expired = global.get('expired');

msg.payload={
    "lot":global.get(‘lot’),
    "space": global.get(‘space’);
    "state": "full"
}

msg._id= global.get(‘id’);
global.set('lastState', "full");

if (expired === true) {
    msg.payload.expired = "yes";
    return msg;
} else {
    msg.payload.expired = "no";
    return msg;
}
```

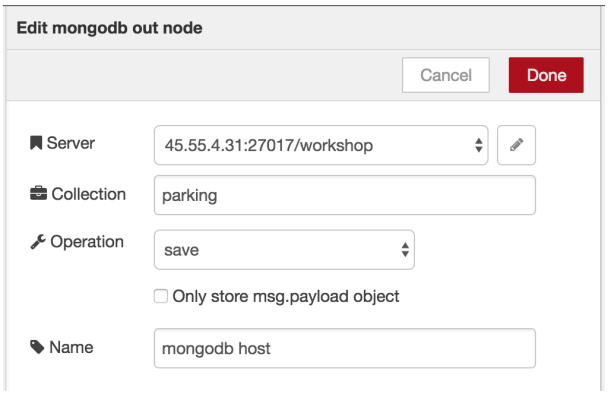
Rename to “SetPayloadFull”.

- “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 host ip address, “27017” as port number, and “workshop” as database name. Use “artikuser” and “iot2016” as your username and password.



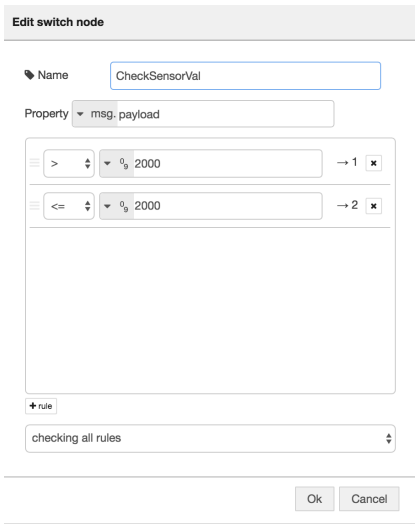
Click “Add” (or “Update” if changing existing parameters) to return to the “Edit mongodb out node” dialog. Enter “parking” as the collection name. Rename to “mongodb host”.



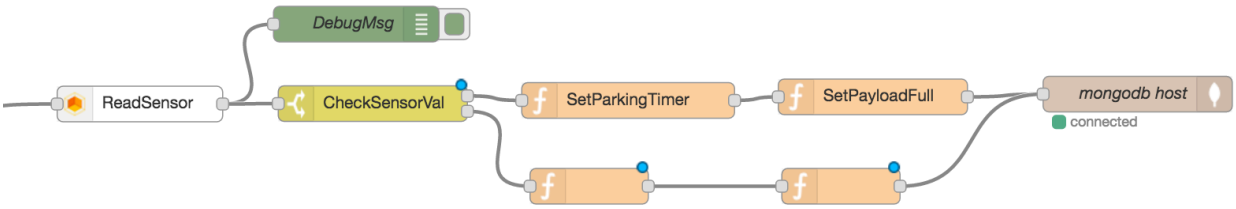
This is what the flow looks like now.

2. Add flow to track parking space state when it is empty.

2.1 Add a 2nd rule to “CheckSensorVal” node. If “msg.payload <= 2000 (**a number**)” THEN “follow Rule 2”.



2. 2 Define the 2nd rule to stream parking space “empty” state to MongoDB.
Drag another two “function” nodes to our canvas, put them below “SetParkingTimer” and “SetPayloadFull” functions. Wire them up.
The output of “CheckSensorVal” node 2nd rule connects to the 1st function.
The output of the 2nd function connects to “mongodb host” node.



Configure both “function” nodes by double clicking.

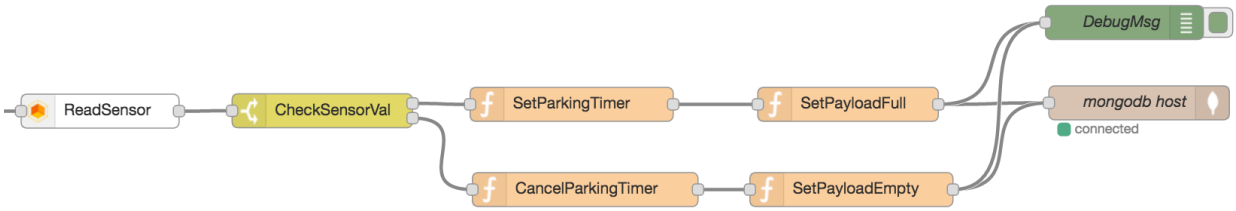
- In the first function node, we initialize our global variables. Here, as in “SetParkingTimer”, we check if *global.expired* and *global.lastState* have been defined. If not, we define them and initialize them to “false” and “empty” respectively.
  We also re-set *global.expired* to be false.
  Rename to “CancelParkingTimer”.

```
var expired = global.get('expired')||false;
var lastState = global.get('lastState')||"empty";

global.set('expired', false);

return msg;
```

- In the second function node, we format the message payload for MongoDB. There are 4 fields in our database collection: Lot#, Space#, Space occupancy(“empty”)and if there is a parking violation in this space(“no”). Here is what the code looks like:
  Rename to “SetPayloadEmpty”.

```
msg.payload={
   "lot": global.get('lot'),
   "space":global.get('space'),
   "state": "empty",
   "expired": "no"
}

msg._id= global.get('id');
global.set('lastState', "empty");
return msg;
```

- Change the input of our "DebugMsg" node to be the outputs of "SetPayloadFull" and "SetPayloadEmpty", so we can monitor our payloads.



Once you deploy your changes, you should be able to see messages similar to the ones below on your Debug Tab.

3. Display real-time parking occupancy on a web page.

3.1 Drag a “http” **input** node 

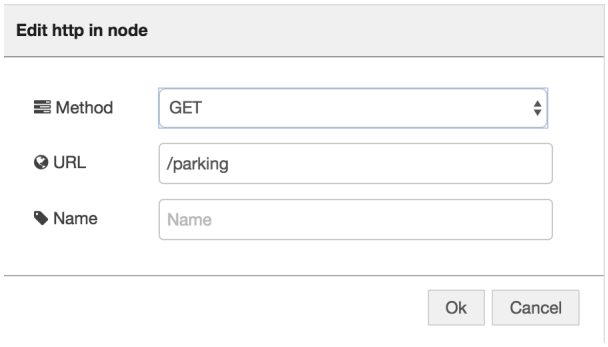
, a mongodb **in** node 

, a “template” node and a “**http response**” node 

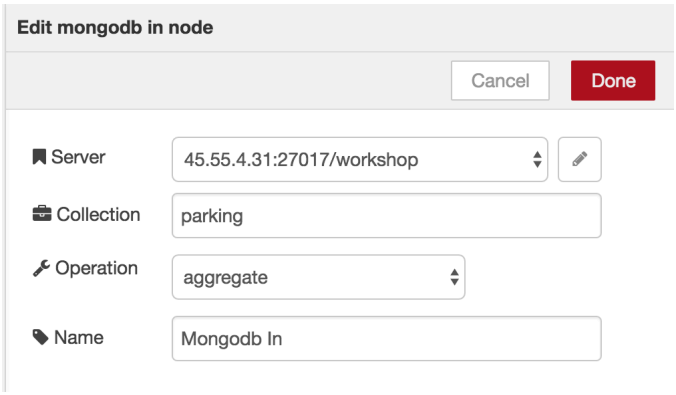
to the canvas. Wire them up

Configure each node by double clicking:

- In “http” input node, we define our parking status URL to be “/parking”.

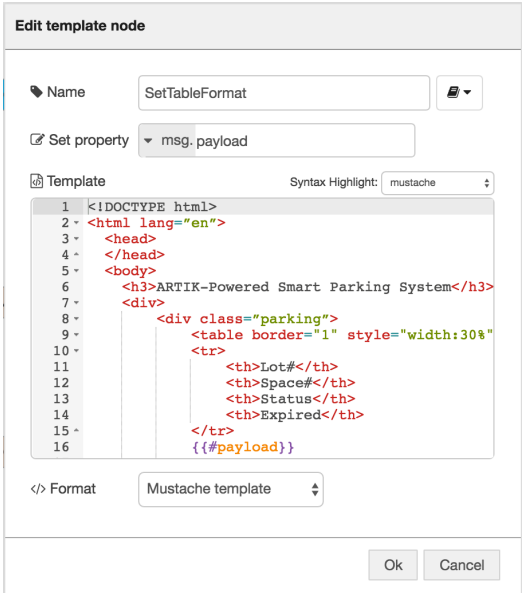


- In “Mongodb” in node, it should have picked up our settings from the MongoDB output node configuration above. Add “parking” as collection name. Rename the node as “Mongodb In”.



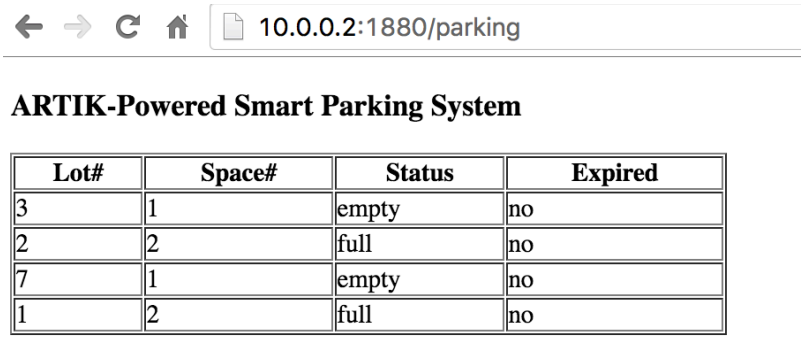
- In “template” node, here is the code you need to copy over. Rename to “SetTableFormat”.

```
<!DOCTYPE html>
<html lang=”en”>
 <head>
 </head>
 <body>
  <h3>ARTIK-Powered Smart Parking System</h3>
  <div>
    <div class=”parking”>
      <table border="1" style="width:30%">
      <tr>
        <th>Lot#</th>
        <th>Space#</th>
        <th>Status</th>
        <th>Expired</th>
      </tr>
      {{#payload}}
      <tr>
        <td>{{payload.lot}}</td>
        <td>{{payload.space}}</td>
        <td>{{payload.state}}</td>
        <td>{{payload.expired}}</td>
      </tr>
      {{/payload}}
      </table>
    </div>
  </div>
 </body>
</html>
```



Here is what our final flow looks like:

Now, deploy your changes. open a browser and launch <your_ARTIK_IP_address>:1880/parking, you will be able to see the real-time occupancy of our parking lots.



4. Monitor parking violation

4.1 In “SetParkingTimer” function, change the code to the block below. Here we introduce one more global variable *global.timerId*, and use it to keep track of our timer. When the parking space state changes from “empty” to “full”, set up our timer. Timer will be fired after 20 seconds, then, timer() function will be invoked.

The highlighted parts are the parts to be added.

```
var expired = global.get('expired')||false;
var lastState = global.get('lastState')||"empty";
var timerId = global.get('timerId')||0;

function timer() {
  global.set('expired', true);
}

if (lastState === "empty") {
  timerId = setTimeout(timer, 20000);
  global.set('timerId', timerId);
}

return msg;
```

4.2 In “CancelParkingTimer” node, cancel the timer. The highlighted lines are what we added.

```
var expired = global.get('expired')||false;
var timerId = global.get('timerId')||0;
var lastState = global.get('lastState')||"empty";

clearTimeout(timerId);
global.set('expired', false);

return msg;
```

4.3 Deploy our changes, and leave your car in the space for more than 20 seconds, then check if your parking space violation status has been updated on the webpage.

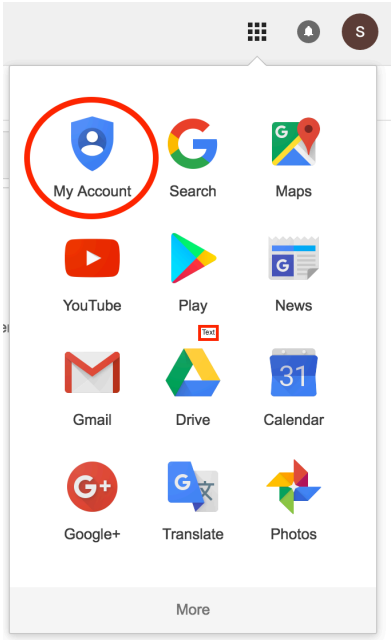
ARTIK-Powered Smart Parking System

Lot#	Space#	Status	Expired
3	1	empty	no
2	2	full	no
7	1	empty	no
1	2	full	yes

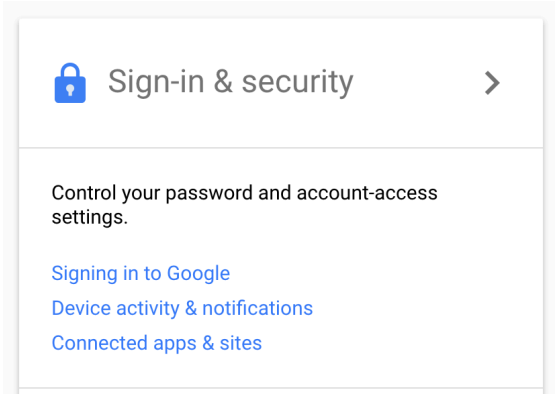
5. Get an email alert when there is a parking violation

5.1 We are using SMTP for Gmail service. For this workshop, we need to turn on “Access for less secure apps” option in our Gmail setting.

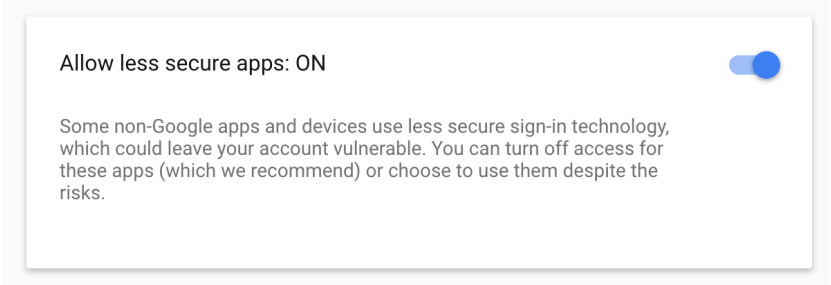
Go to My Account settings page,



Click the “Sign-in & security” link,



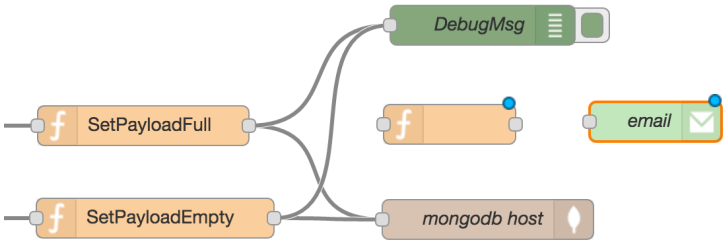
And turn on “Allow less secure apps”.



5.2 Extend the existing Node-RED flow to include “send email” capability.

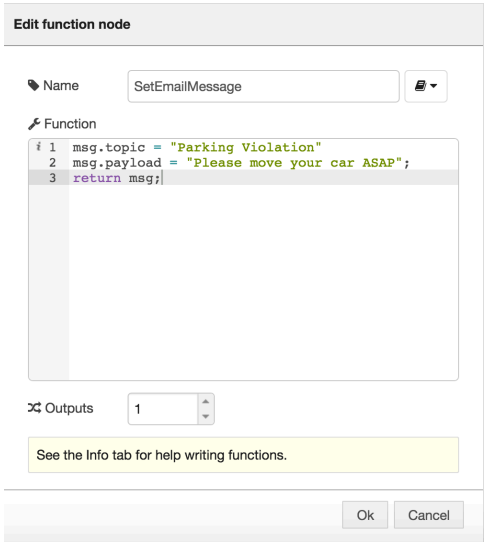
- Drag a “function” node and an “email” **output** node 

, drop them under the “DebugMsg” node and above “MongoDB host” node.

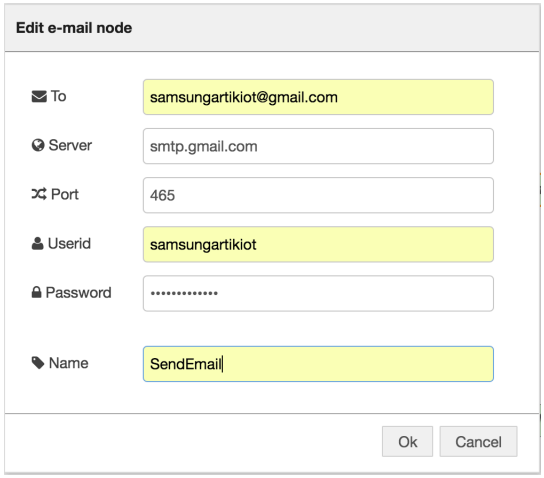


- Configure “function” node by double-clicking. In this “function” node, we will define the email subject (msg.topic) and email message body (msg.payload). Rename the node to “SetEmailMessage”.

```
msg.topic = "Parking Violation";
msg.payload = "Please move your car ASAP";
return msg;
```



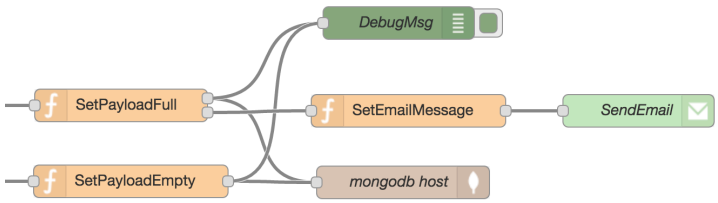
- Configure “email” output node by double-clicking. Enter your email address in “To”, “UserId” and “Password”. Rename to “SendEmail”.



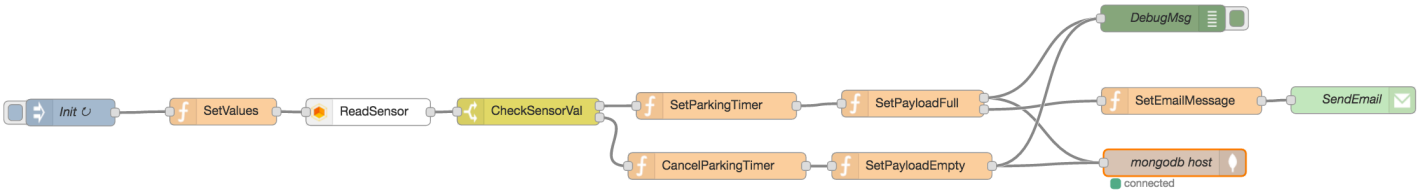
- Now update “SetPayloadFull” function. Change the number of Outputs to 2, and update our function to include return values for the 2nd output.

```
var expired = global.get('expired');

msg.payload={
    "lot": global.get('lot'),
    "space": global.get('space'),
    "state": "full"
}

msg._id= global.get('id');
global.set('lastState', "full");

if (expired === true) {
  msg.payload.expired = "yes";
  return [msg, msg];
} else {
  msg.payload.expired = "no";
  return [msg, null];
}
```

- Wire up the 2nd output of “SetPayloadFull” to the “SendEmail” path.



5.3 Deploy your flow. You should see that you get an email when there is a parking violation in your parking space.



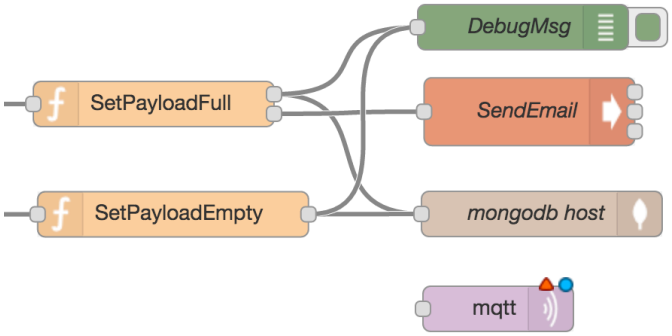
Exercise 3: Subscribe to parking space state change notification (M2M & Node-RED)

1. Select one board as the MQTT broker in the group, and let your group members know your board's IP address.

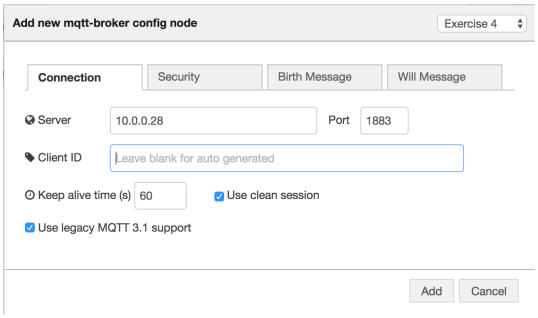
2. Send MQTT Messages when your parking space state changes.

2.1 Drag an mqtt **output** node 

below "mongodb host" node, double click it to configure.

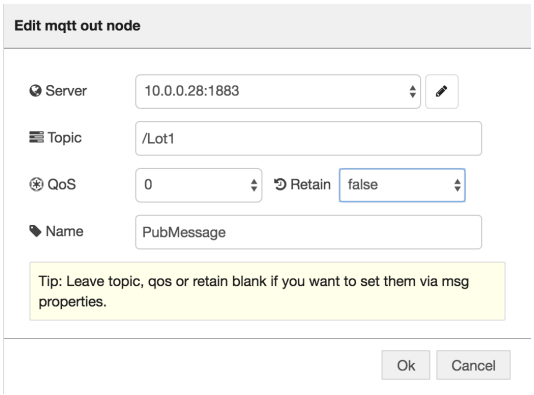


2.2 Click  in the "Edit mqtt out node"dialog, "Add new mqtt-broker config node" opens up. Enter your MQTT broker's IP address and Port number defaults to 1883.



2.3 Click "Add" (or "Update" if changing existing parameters) to return to the "Edit mqtt out node" dialog. Enter your Lot number as the Topic name(e.g, "/Lot1", Replace the topic by using your own Lot#).

Rename to "PubMessage".



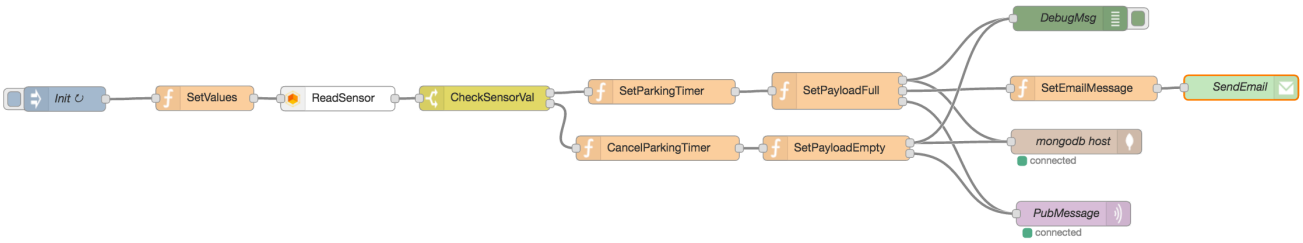
2.4 Update “SetPayloadFull” function. Change the number of Outputs to 3, and update our function to include MQTT pub message output.

```
var lastState = global.get('lastState');
var expired = global.get('expired');

msg.payload={
    "lot":global.get('lot'),
    "space":global.get('space'),
    "state": "full"
}

msg._id= global.get('id');
global.set('lastState', "full");

if (expired === true) {
  msg.payload.expired = "yes";
  return [msg, msg, null];
} else if (lastState === "full") {
  msg.payload.expired = "no";
  return [msg, null, null];
} else if (lastState === "empty") {
  msg.payload.expired = "no";
  var str = global.get('location');
  str = str.concat(" is full");
  var mqttMsg = {payload: str};
  return [msg, null, mqttMsg];
}
```

2.5 Update “SetPayloadEmpty” function. Change the number of Outputs to 2, and update our function to include MQTT pub message output.

2.6 Wire up the 3rd output of “SetPayloadFull” node and the 2nd output of “SetPayloadEmpty” node to “PubMessage” node.

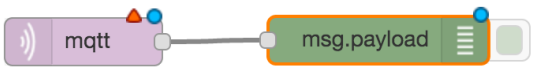
```
var lastState = global.get('lastState');

msg.payload={
    "lot":global.get('lot'),
    "space":global.get('space'),
    "state": "empty",
    "expired": "no"
}
 var str = global.get('location');
  str = str.concat(" is empty");
  console.log(str);
msg._id= global.get('id');
global.set('lastState', "empty");

if (lastState === "full") {
  var str = global.get('location');
  str = str.concat(" is empty");
  var mqttMsg = { payload: str };
  return [msg, mqttMsg];
} else {
  return [msg, null];
}
```



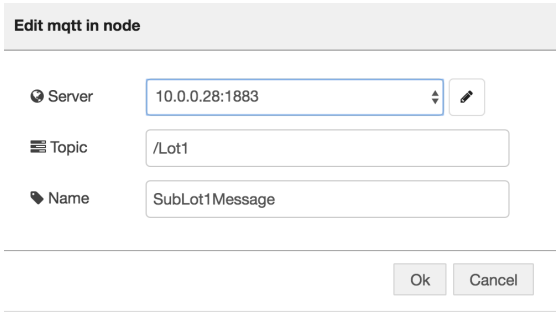
3. Subscribe to MQTT Parking space messages. Now we are creating another flow to listen for messages from a specific lot.

3.1 Add “mqtt” **input** node 

and “debug” nodes, and wire them up.



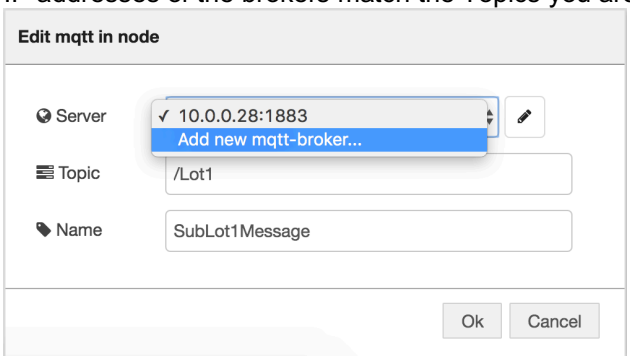
- "mqtt" input node: Set the Server address to the IP address of the broker of Lot1. Change Topic to "/Lot1"(Please replace the broker IP address and topic of the Lot# you are interested in). Rename to "SubLot1Message"(Or "SubLot2Message", "SubLot3Message").



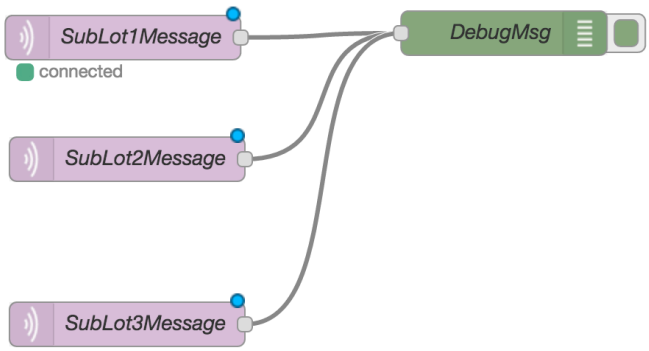
Here is what the 3rd flow looks like:

You can then extend your flow to listen to parkig space states from other lots too. Drag another "mqtt" **input** node to our canvas, and drop it under "SubLot1Message" node.

Double click the node to configure. In order to add another mqtt broker to listen to, click the dropdown list from "Server" and select "Add new mqtt-broker…" Configure the node as in the step above. Please make sure the IP addresses of the brokers match the Topics you are subscribing to.



You can repeat this and add more mqtt sub messages.



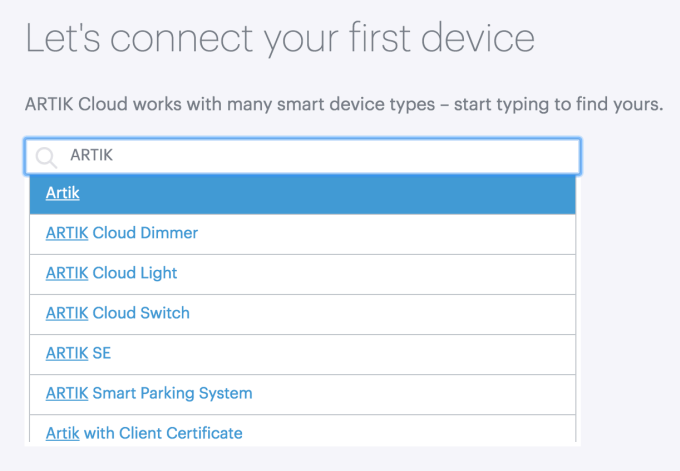
3.2 Deploy your flows, and get a notification when the parking lot you subscribed to has a parking space state change.

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

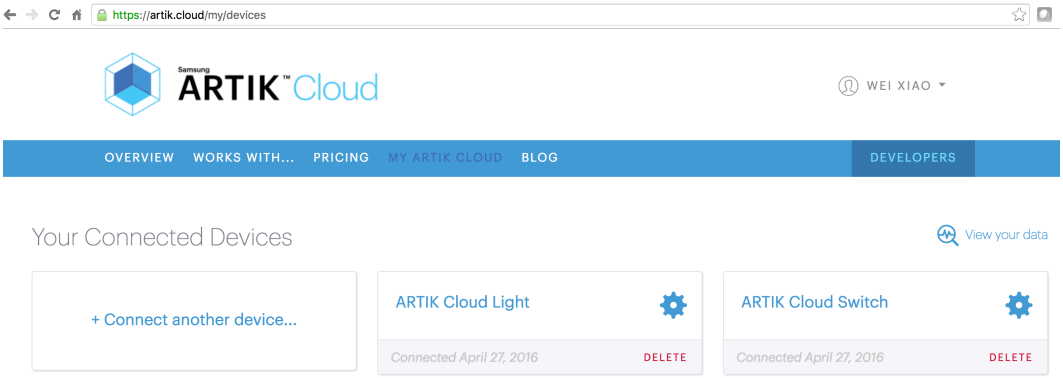
In this exercise, we are going to stream parking lot states to ARTIK Cloud by using MQTT. As an extension to Exercise 3, we will use ARTIK Cloud as the MQTT broker.

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

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



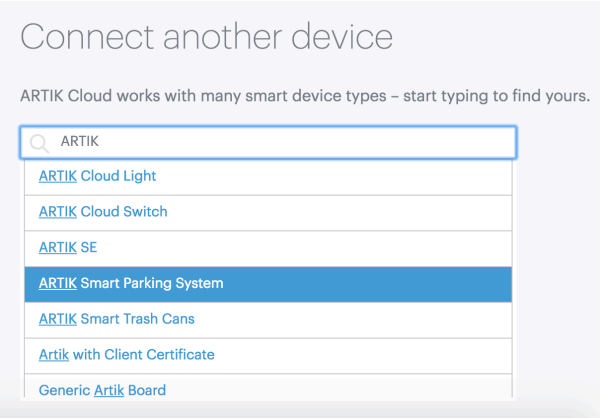
Otherwise, Under “MY ARTIK CLOUD”/”DEVICES”, you will see all you connected devices.



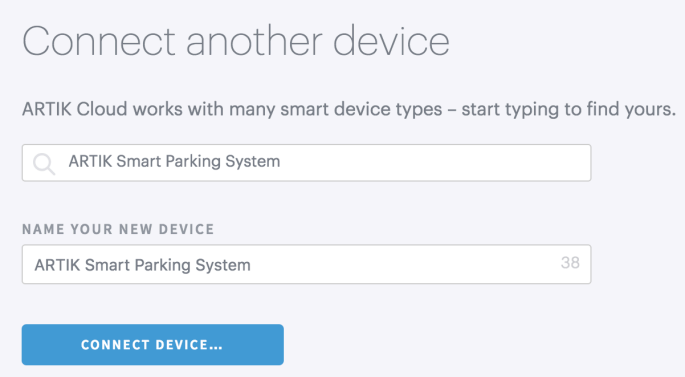
Click “Connect another device…” link.

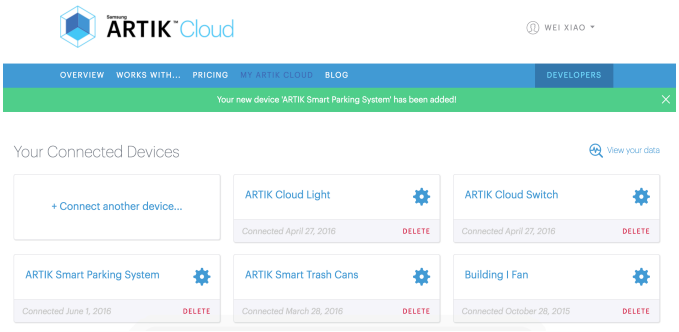
1.2 Search for “ARTIK Smart Parking System” and select it.

“ARTIK Smart Parking System” is a public device type we created for the workshop, which includes “lot”(a number), “space”(a number), “state”(“empty” or “full”) and “reserved”(true or false) properties.

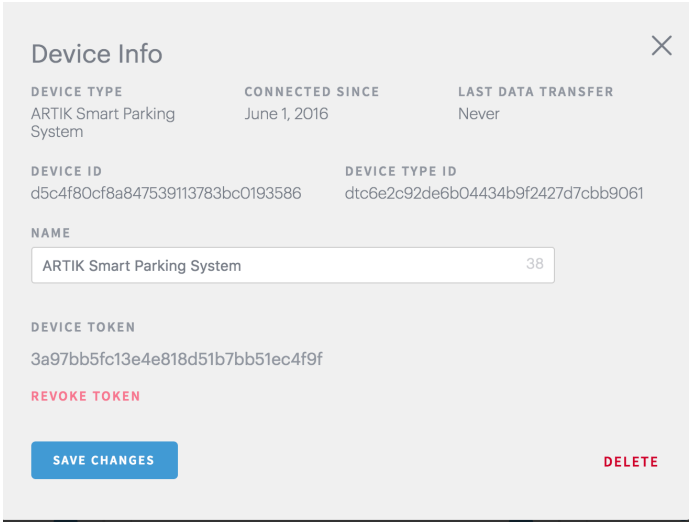


1.3 Use the default device name “ARTIK Smart Parking System”, then click the “CONNECT DEVICE…” button. A new device will be created.



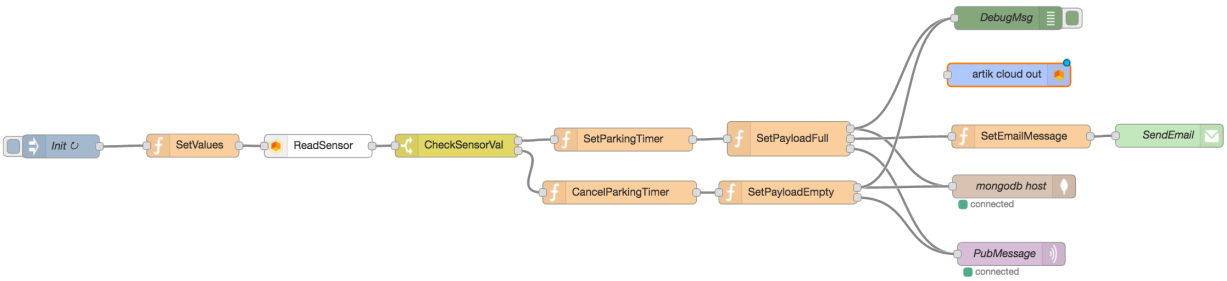


1.4 Click the Gear icon next to your newly created device, you will see the Device Info popup, which shows your Device Type, Device ID, Device name etc. details. Click the “GENERATE DEVICE TOKEN…” link to generate a device token.

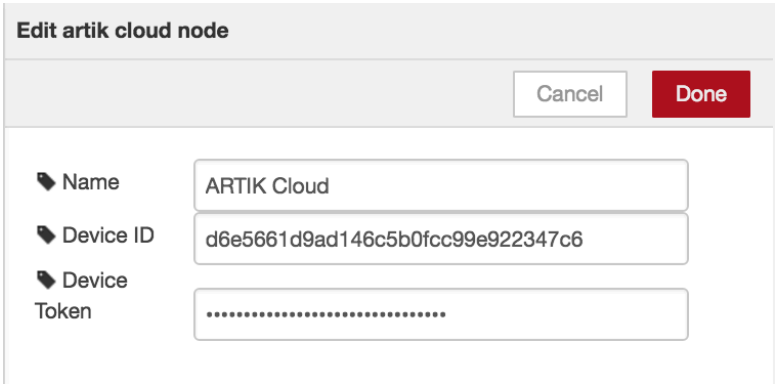


We need to use the **Device ID** (not “DEVICE TYPE ID”) and **Device Token** to associate our parking space with ARTIK Cloud. Make a copy or leave the dialog open.

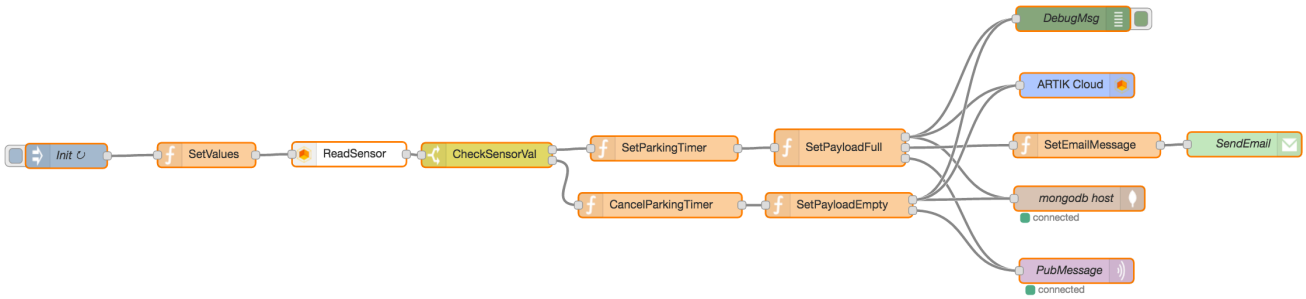
2. Drag an ARTIK Cloud output node to the canvas, and place it under the DebugMsg node.



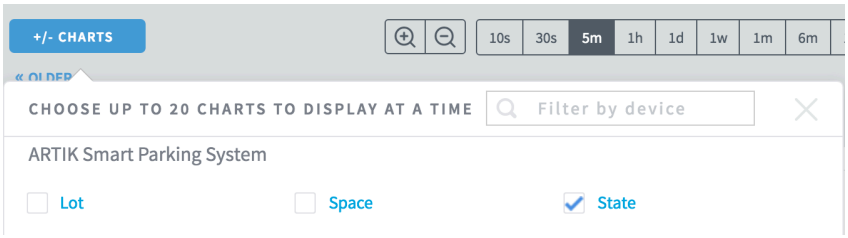
Wire the first output of SetPayloadFull and SetPayloadEmpty to the ARTIK Cloud output node. Double click the node and enter the Device ID and Token in the Edit Window, Rename to “ARTIK Cloud”.



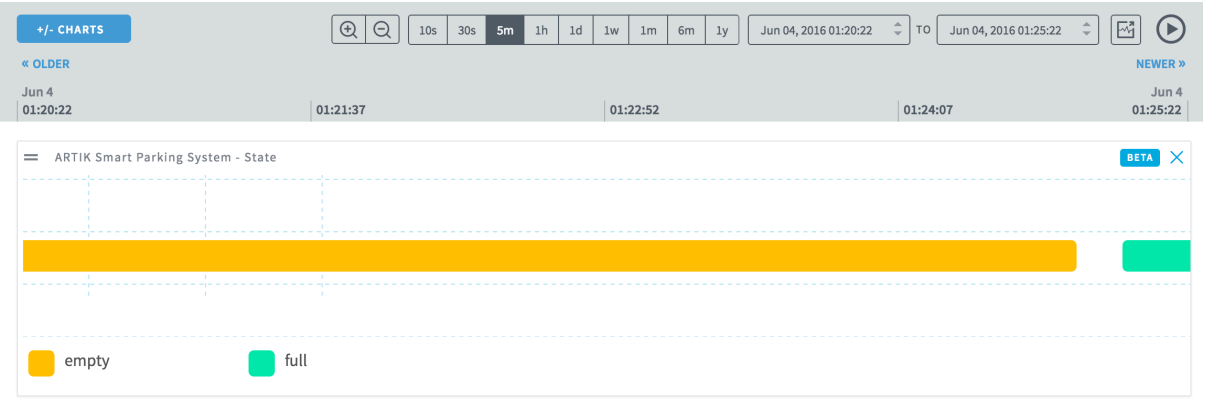
This is what your final flow should look like:



Go to ARTIK Cloud user portal, click the “+/- CHARTS” button, and enable to view the “State” of your parking space.



You will see the streamed parking space states in your portal:

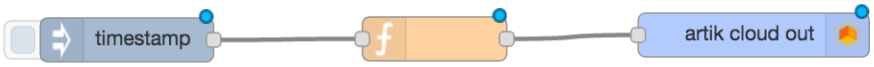

+/- CHARTS
10s
30s
5m
1h
1d
1w
1m
6m
1y
Jun 04, 2016 01:20:22
TO
Jun 04, 2016 01:25:22
« OLDER
NEWER »
Jun 4
01:20:22
01:21:37
01:22:52
01:24:07
Jun 4
01:25:22
ARTIK Smart Parking System - State
BETA
empty
full

Exercise 5 (Bonus Exercise): ARTIK Cloud Rules Engine (ARTIK Cloud, Arduino, Node-RED)

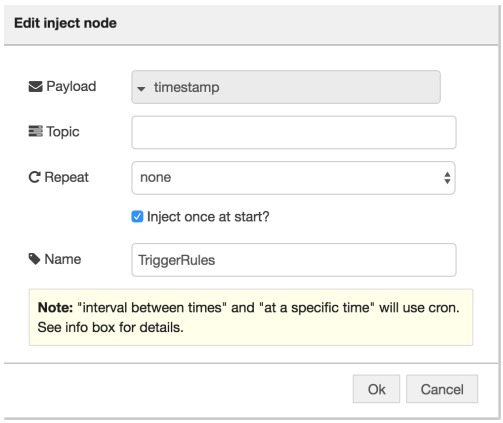
In this exercise, we are going to explore a more advanced feature of ARTIK Cloud – the Rules Engine.

In the “ARTIK Smart Parking System” device type we used in Exercise 4, it also includes a property “reserved”. Here, we will use this property to simulate parking space reservation and trigger a cross-device action, which toggles a LED to indicate reservation status.

1. Use Node-RED to simulate parking space reservation.
   1.1 Drag an “inject” node, a “function” node and an “ARTIK Cloud” **output** node to the canvas and wire them up.

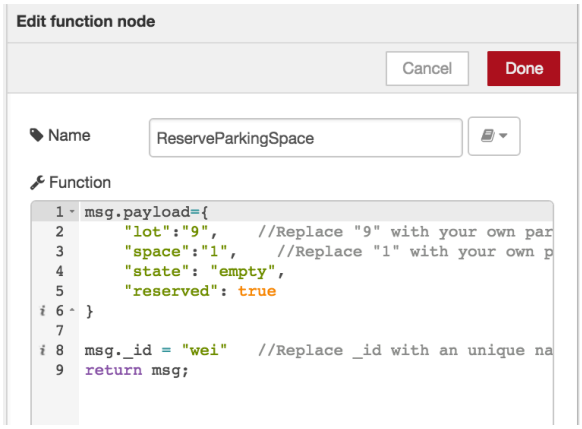


   1.2 Double click each node to configure.
   - “inject” node(shows as “timestamp”): enable “Inject once at start?” option. Rename it as “TriggerRules”.

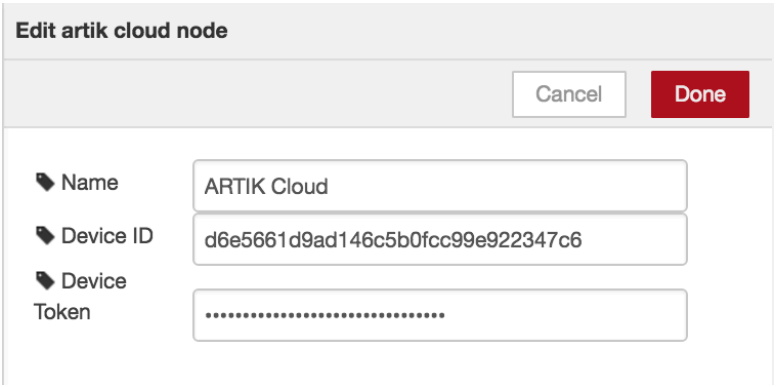


   - “function” node: Here, We set the message payload “reserved” property to be true. Rename the node as “ReserveParkingSpace”.

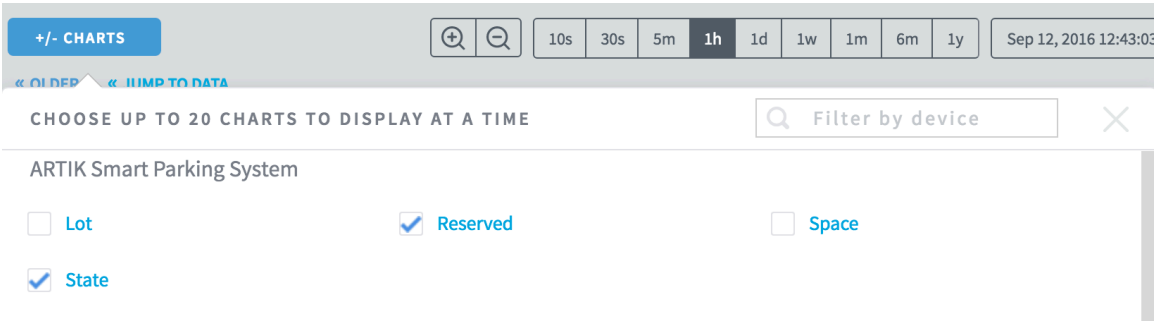
```
msg.payload={
    "lot":"9",    //Replace "9" with your own parking lot#
    "space":"1",    //Replace "1" with your own parking space#
    "state": "empty",
    "reserved": true
}

msg._id = "wei"   //Replace _id with an unique name
return msg;
```



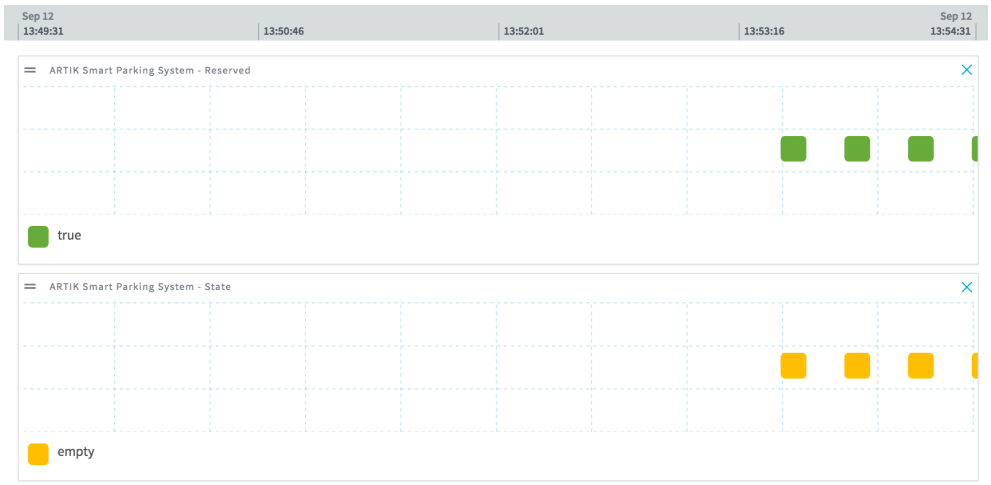
- "ARTIK Cloud" output node should have automatically picked up our previous configuration.



1.3 Go to ARTIK Cloud user portal, click "+/- CHARTS" button, and enable to view the "Reserved" property of your parking space.



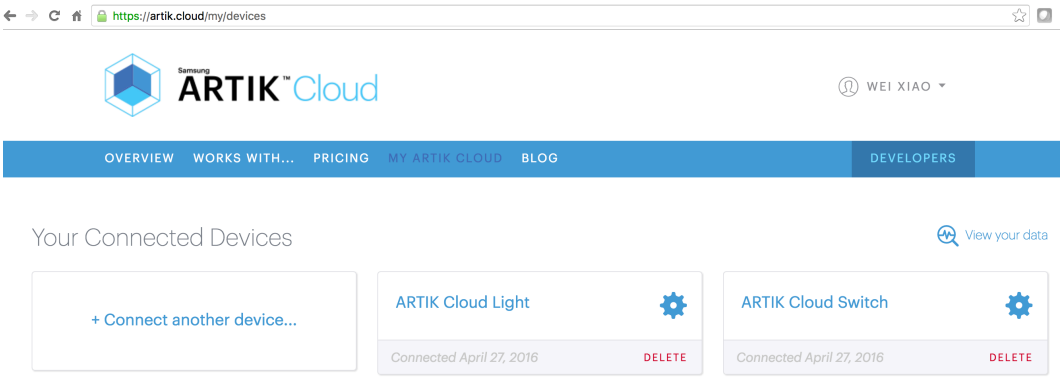
1.4 Click the little button on the left side of your "TriggerRules" node to inject a message. You should be able to see streamed "reserved" data in your ARTIK Cloud portal.



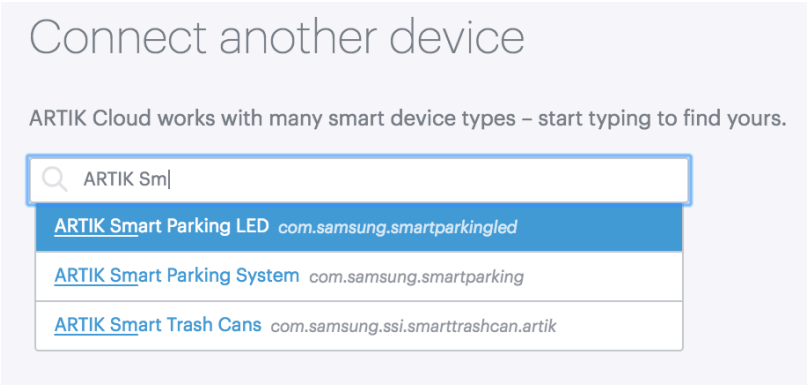
2. Connect a LED/resistor combo to the ARTIK board, and create an LED device in ARTIK Cloud user portal accordingly.

   2.1 Connect the longer leg of your LED/resistor combo to header J27 pin 13 and shorter leg to header J27 GND.

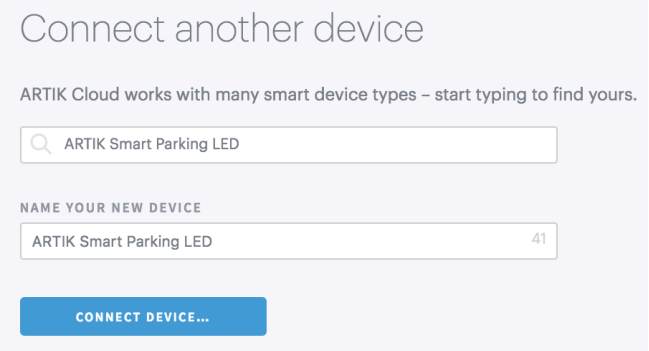
   2.2 From ARTIK Cloud user portal https://artik.cloud, go to “MY ARTIK CLOUD”/”DEVICES”, click “Connect another device…” link.

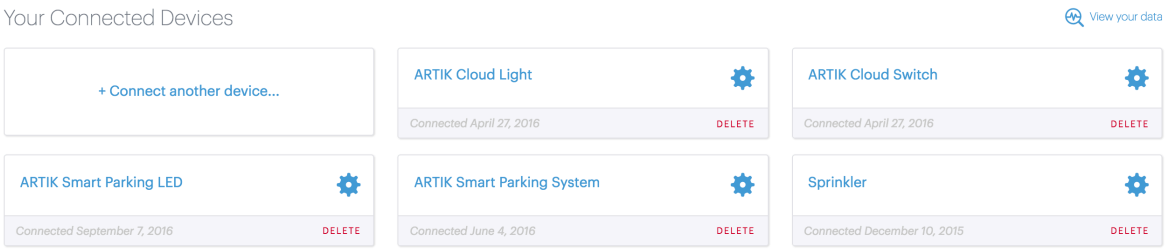


   2.2 Search for “ARTIK Smart Parking LED” and select it. This is the LED public device type we created for this workshop. The device type includes only one property “LED”, which is a Boolean type. Its valid value is either true or false, to represent if the LED is ON or OFF.

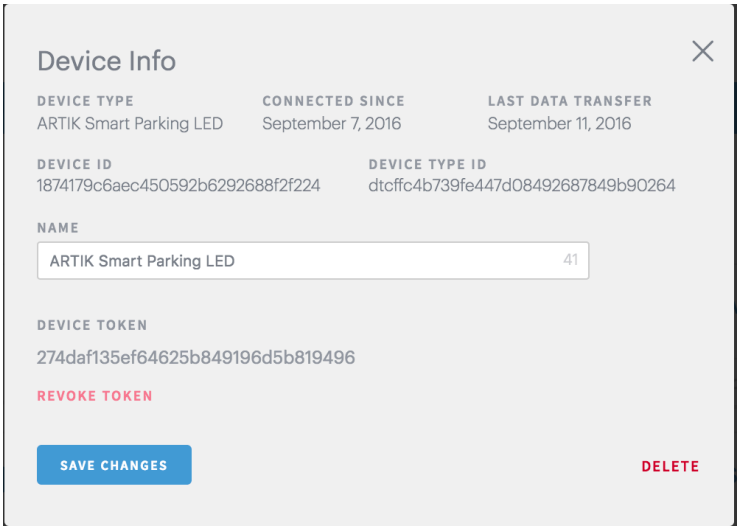


2.3 Use the default device name “ARTIK Smart Parking LED”, then click the “CONNECT DEVICE…” button. A new device will be created.





2.4 Click the Gear icon next to your newly created device, 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.



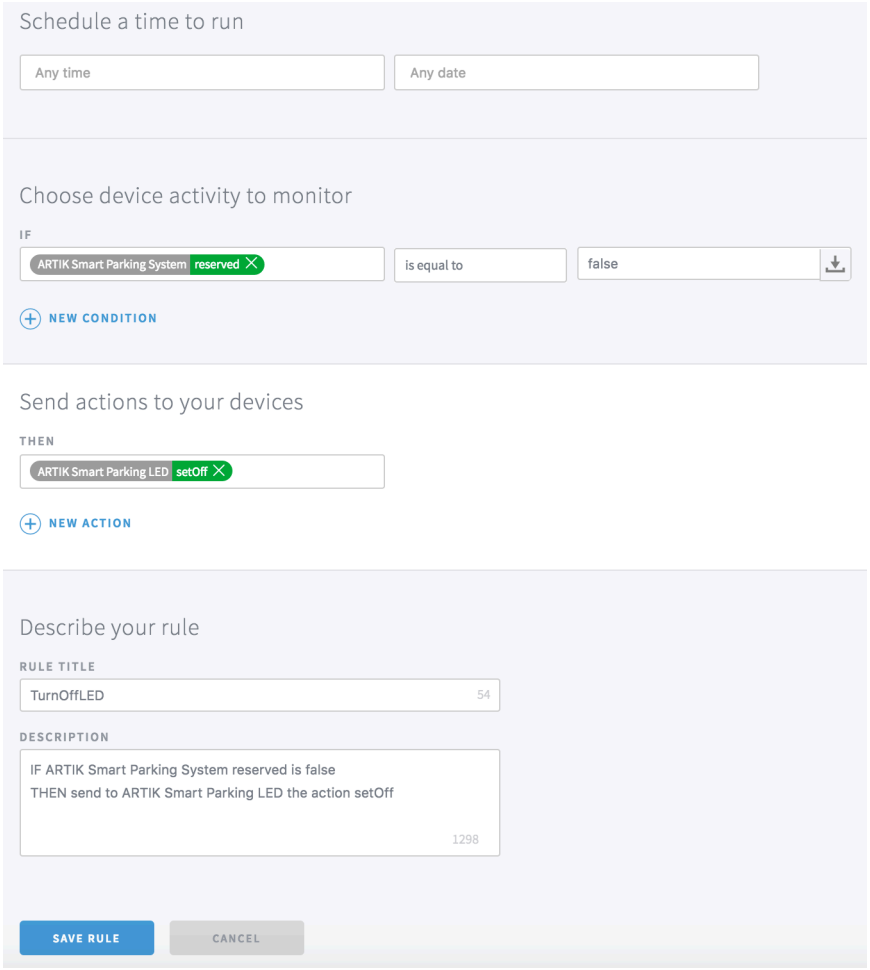
3. Add Rules to our Rules Engine. By doing this, we will be able to connect our “ARTIK Smart Parking System” device and “ARTIK Smart Parking LED” device, and trigger cross-device actions.

3.1 In ARTIK user portal, go to “MY ARTIK CLOUD/RULES”, and click the “+NEW RULE” button. The first rule we will add is to turn on LED when the parking space is reserved. You can create the rule by using plain English language.

IF
“ARTIK Smart Parking System” reserved is equal to **true**
THEN
“ARTIK Smart Parking LED” setOn

Click “SAVE RULE” button.

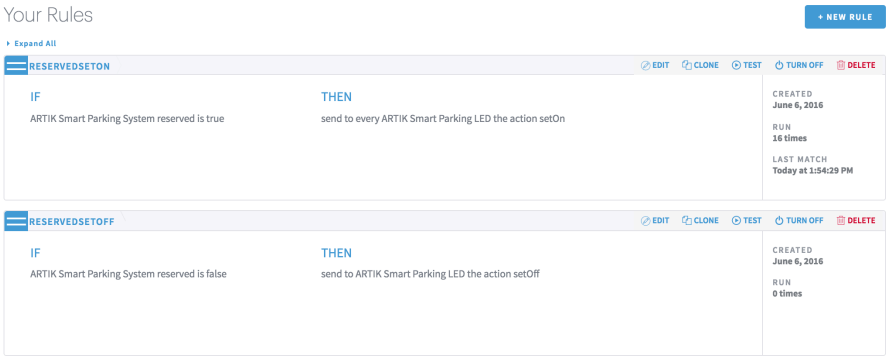
3.2 Repeat the step above, and create a second rule to turn off the LED when the reservation is canceled or expired.



IF
“ARTIK Smart Parking System” reserved is equal to **false**
THEN
“ARTIK Smart Parking LED” setOff

Click “SAVE RULE” button.

3.3 Now, we have 2 rules defined:



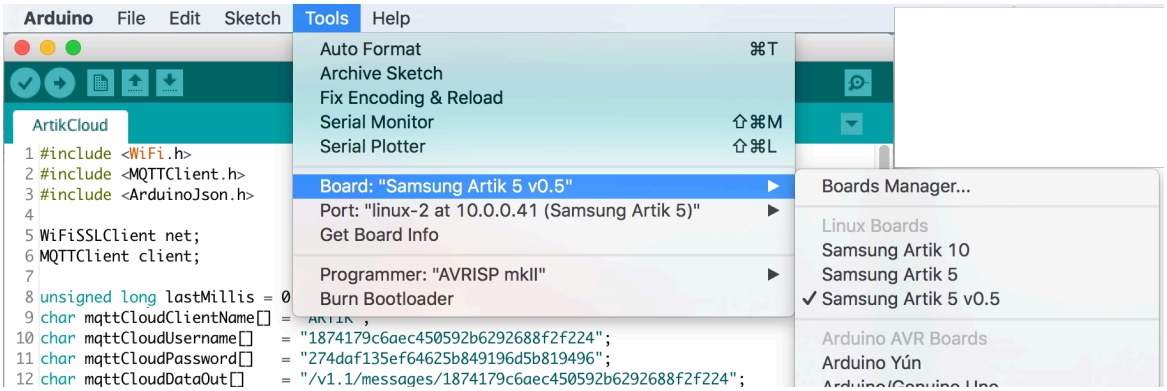
4. Use Arduino to listen to the triggered action.

4.1 Launch Arduino IDE, and click File->New. Copy the sketch below to your Arduino editor:

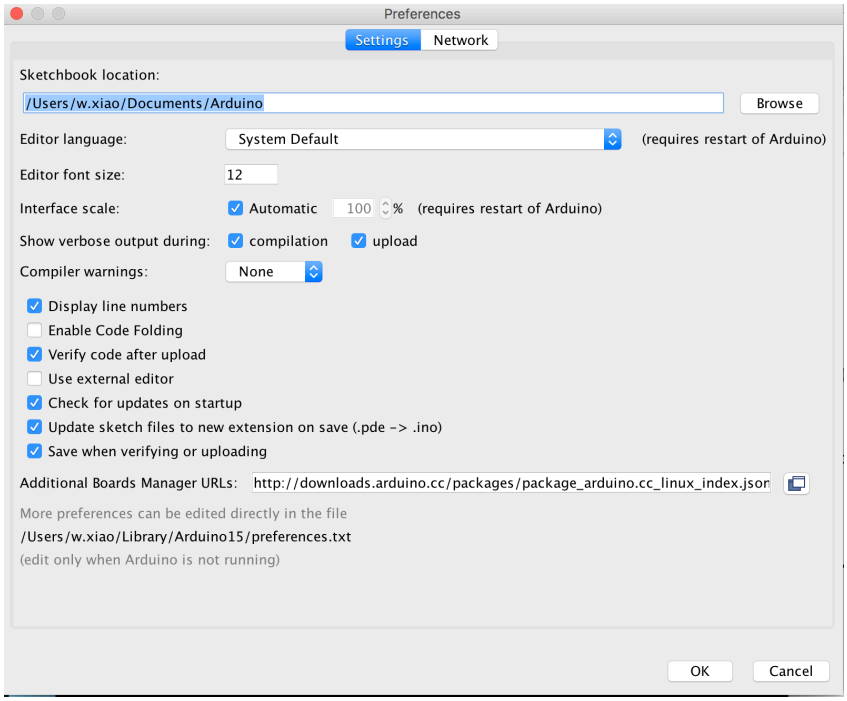
```
#include <WiFi.h>
#include <MQTTClient.h>
#include <ArduinoJson.h>

#define Serial DebugSerial
WiFiSSLClient net;
MQTTClient client;

char mqttCloudClientName[] = "ARTIK";
char mqttCloudUsername[]   = "your_led_DEVICE_ID_goes_here";
char mqttCloudPassword[]   = "your_led_DEVICE_TOKEN_goes_here";
char mqttCloudActionsIn[]  = "/v1.1/actions/your_led_DEVICE_ID_goes_here";

void setup() {
  Serial.begin(115200);
  client.begin("api.artik.cloud", 8883, net);
  pinMode(13, OUTPUT);
  connect();
}

void connect() {
  Serial.print("\nconnecting...");
  while (!client.connect(mqttCloudClientName, mqttCloudUsername, mqttCloudPassword)) {
    Serial.print(".");
    delay(1000);
  }
  Serial.println("\nconnected!");
  client.subscribe(mqttCloudActionsIn);
}

void loop() {
  client.loop();
}

void messageReceived(String topic, String payload, char * bytes, unsigned int length) {
  parseBuffer(payload);
}

void parseBuffer(String payload) {
  StaticJsonBuffer<200> jsonBuffer;
  String json = payload;
  JsonObject& root = jsonBuffer.parseObject(json);
  const char* actionname = root["actions"][0]["name"];

  if (strcmp(actionname, "setOff") == 0) {
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  }

  if (strcmp(actionname, "setOn") == 0) {
    digitalWrite(13, HIGH);   // turn the LED on by making the voltage HIGH
  }
}
```

4.2 In Arduino IDE,

- Go to Tools menu and select “Samsung ARTIK 5 v0.5” as your target board.



- Go to “Arduino -> Preferences”, enable show verbose output during both compilation and upload.



- Click “Verify” button to compile your sketch and upload the compiled sketch to your board by using serial upload or network upload introduced in our online tutorial https://developer.artik.io/documentation/developer-guide/ide/arduino.html

  Alternatively, please follow **Copying Executable Binary File to ARTIK** section in the link above to transfer the executable to your ARTIK board, and launch it from the command line.

4.3 Now, when you set “reserved” parking space to be true or false from your Node-RED. Your LED should be turned on or off immediately.