Develop IoT with Samsung ARTIK platform

June 13th, 2016
IoT Tech Expo, Berlin

Samsung Strategic and Innovation Center

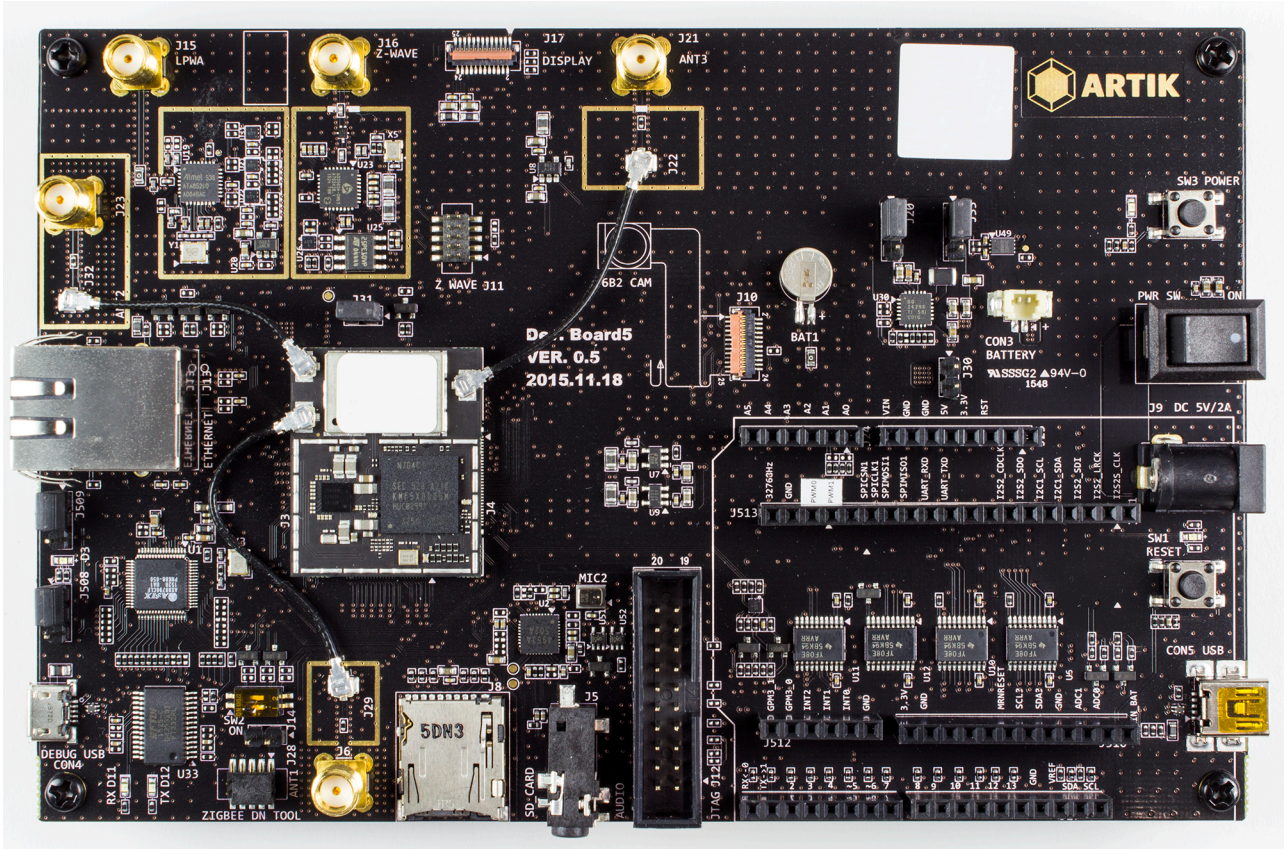


Table of Contents

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
- Use Temboo for Web service integration
- Enable MQTT so that your ARTIK boards can communicate with each other
- Connect to ARTIK 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 ARTIK Cloud user portal account: We will stream data to ARTIK Cloud service. Create an account at ARTIK Cloud user portal (https://artik.cloud/).

5. Bring your cell phone with SMS capability. 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 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 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.

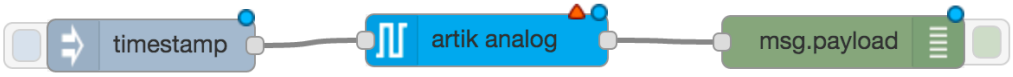
```
[root@localhost ~]# node-red &
…
Welcome to Node-RED
===================

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.

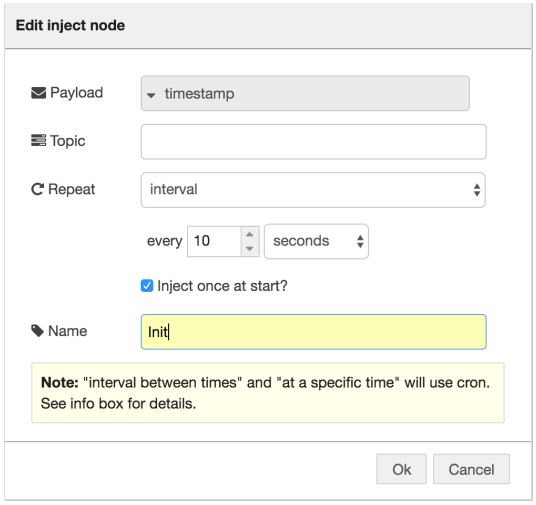
5.1 Create a flow on the Node-RED canvas.

- 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.

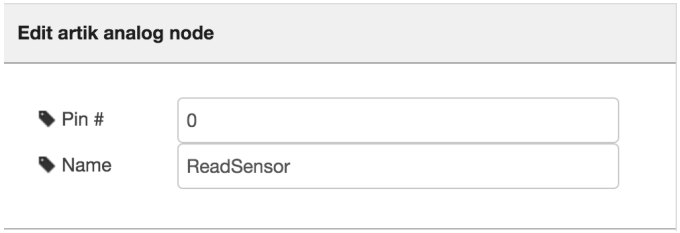


5.2 Configure each node by double-clicking.

- “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”.

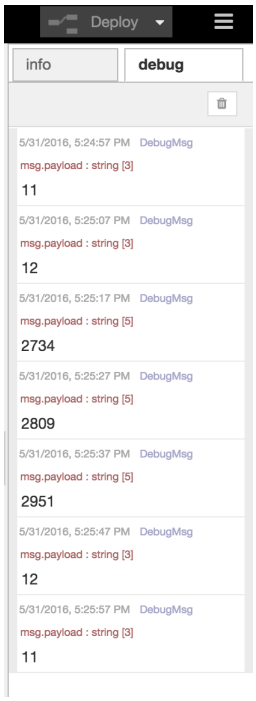


- “debug” node: Name as “DebugMsg”. Here is our final flow.

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.


Deploy
info
debug
5/31/2016, 5:24:57 PM DebugMsg
msg.payload : string [3]
11
5/31/2016, 5:25:07 PM DebugMsg
msg.payload : string [3]
12
5/31/2016, 5:25:17 PM DebugMsg
msg.payload : string [5]
2734
5/31/2016, 5:25:27 PM DebugMsg
msg.payload : string [5]
2809
5/31/2016, 5:25:37 PM DebugMsg
msg.payload : string [5]
2951
5/31/2016, 5:25:47 PM DebugMsg
msg.payload : string [3]
12
5/31/2016, 5:25:57 PM DebugMsg
msg.payload : string [3]
11

Exercise 2: Track Parking Lots Occupancy (Node-RED & MongoDB)

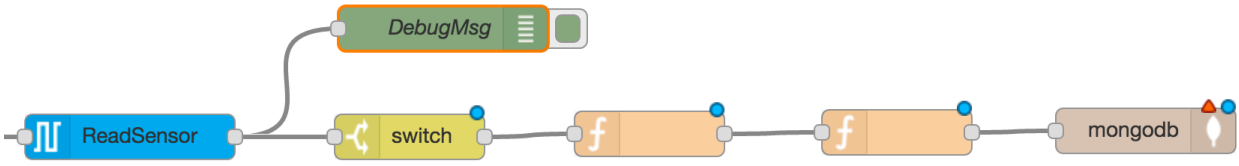
In exercise 2 and exercise 3, we will monitor if our parking space is occupied. If it is getting occupied, we will start a timer. When the timer interval elapses and timer fires, ARTIK will send you an email to alert you that you have a parking violation. In the meantime, ARTIK streams parking space 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

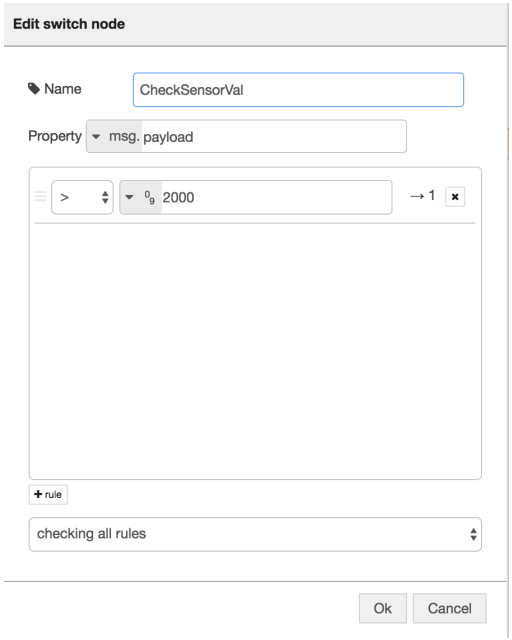
- Drag a “switch” function node from the node palette; drop it under the debug node on the canvas.
- Drag two “function” nodes to the right of the switch node.
- Drag a “mongodb” **output** node to the rightmost.
- Connect the nodes as shown.



1.2 Configure each node by double clicking.

- “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 node, 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;
```

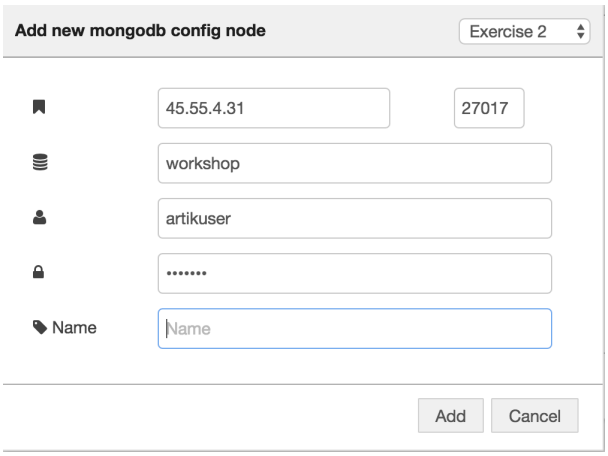
- In the second function node, we format the message payload for MongoDB. There are 4 fields in our database collection: Lot#, Space#, Space occupancy state(“full”) and if there is a parking violation in this space(“yes” or “no”). Replace Lot#, Space#, msg._id with your own when you create your function. Here is what I have as an example:

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

msg.payload={
   "lot":"7",
   "space":"1",
   "state": "full"
}

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

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

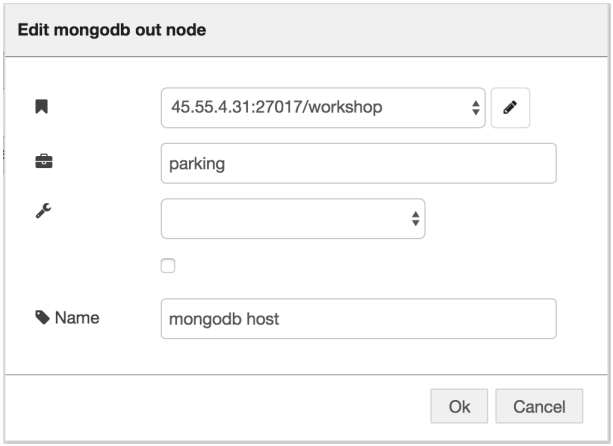
  Rename to “SetPayloadFull”.

- “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.



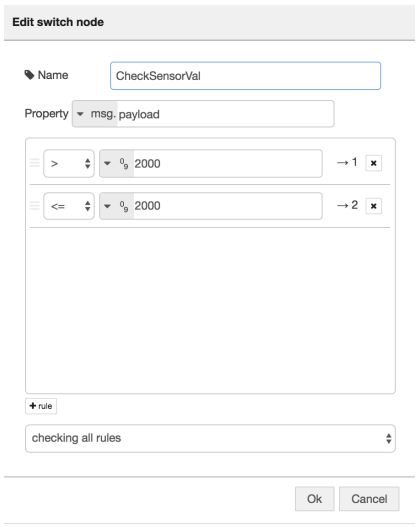
Click “Add” (or “Update” if changing existing parameters) to return to the “Edit mongodb out node” dialog. Enter “parking” as the collection name. Rename to “mongodb host”.



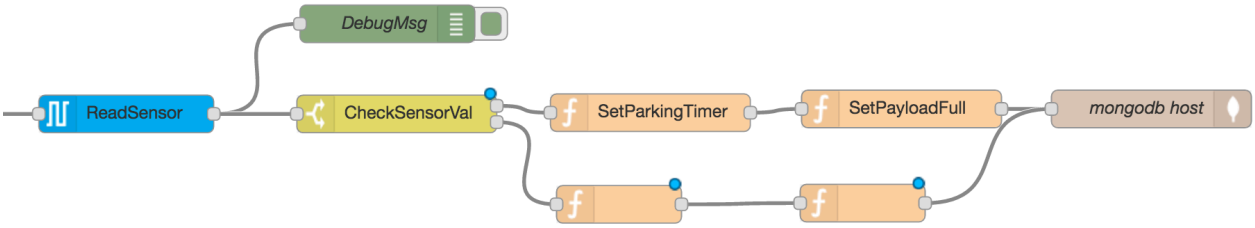
This is what the flow looks like now.

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”.

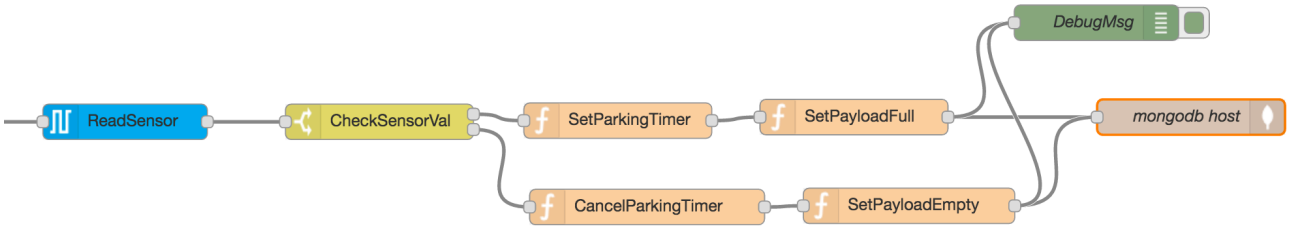
```
var expired = global.get('expired')||false;
var lastState = global.get('lastState')||"empty";

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”). Replace Lot#, Space#, msg._id with your own when you create your function. Here is what I have as an example:

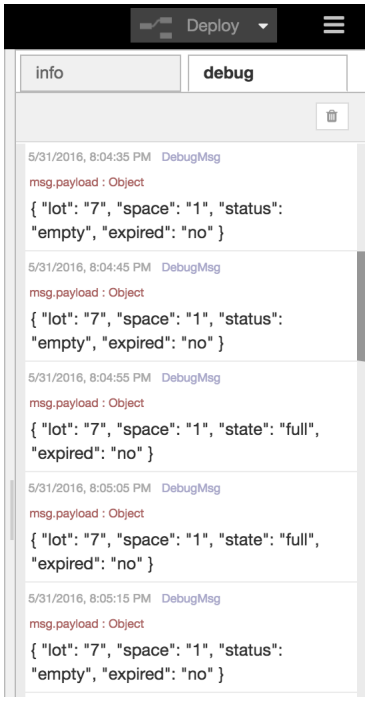
  Rename to “SetPayloadEmpty”.

```
msg.payload={
   "lot":"7",
   "space":"1",
   "state": "empty",
   "expired": "no"
}

msg._id="wei";
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.

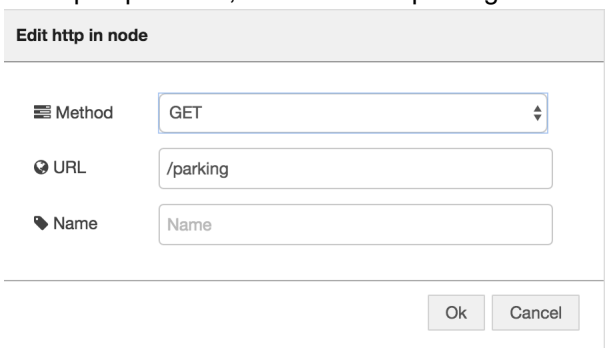


3. Display real-time parking occupancy on a web page.

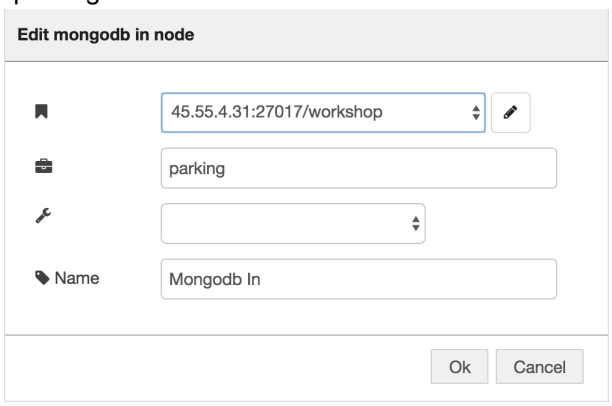
3.1 Drag a “http” input node, a “mongodb in” node, a “template” node and a “http response” node to the canvas. Wire them up

Configure each node by double clicking:

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

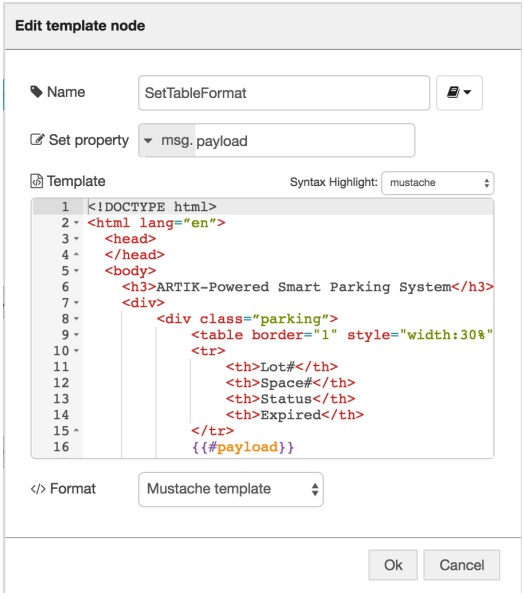


- In “Mongodb” in node, it should have picked up our settings from the MongoDB In node configuration above. Add “parking” as collection name. Rename the node as “MongoDB”.



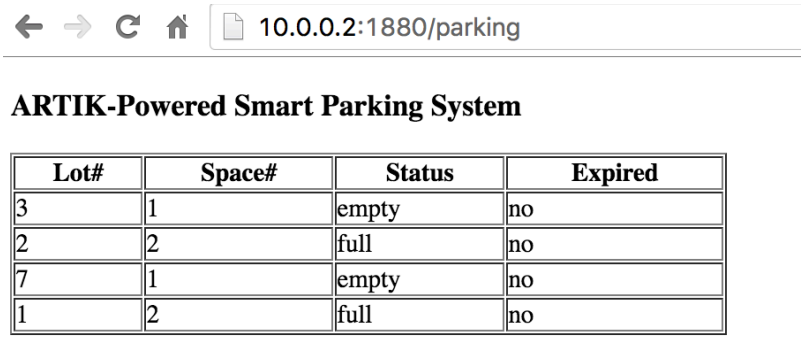
- In “template” node, here is the code you need to copy over. Rename to “SetTableFormat”.

```
<!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. Time to monitor parking violation

4.1 In “SetParkingTimer” function, change the code to the block below. Here we introduced 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, and after 20 seconds, timer will be fired, timer() function is 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

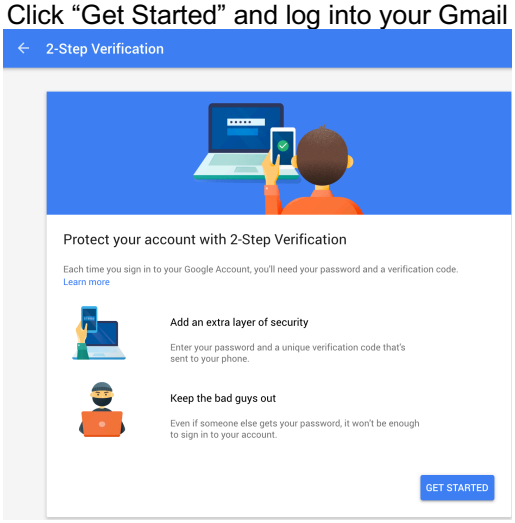
Exercise 3: Alert you of a parking violation (Temboo & Node-RED)

In this exercise, ARTIK will send you an email if it detects a parking violation in the parking space. We will use Temboo for email integration.

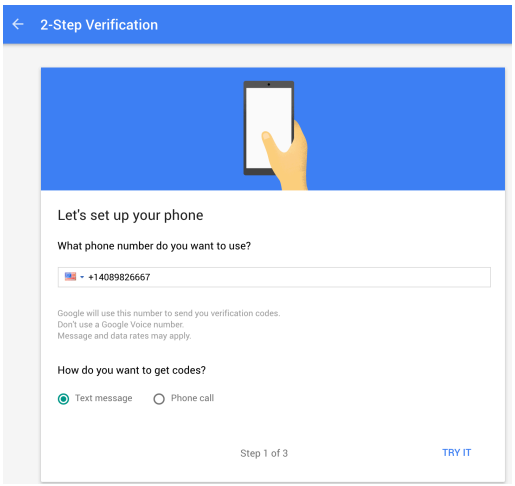
1. Enable 2-step verification in your Gmail account

1.1 Go to the 2-step verification page by following the link below:

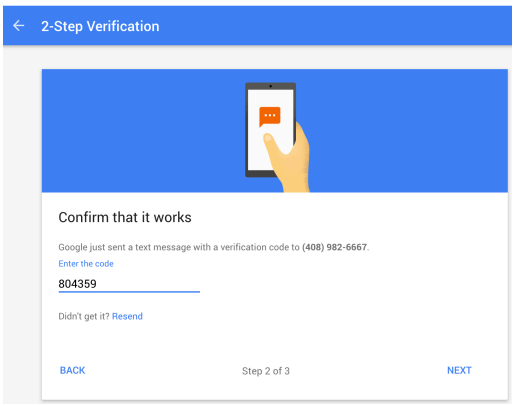
https://accounts.google.com/ServiceLogin?passive=1209600&continue=https%3A%2F%2Faccounts.google.com%2FSmsAuthConfig&followup=https%3A%2F%2Faccounts.google.com%2FSmsAuthConfig
Click “Get Started” and log into your Gmail account.



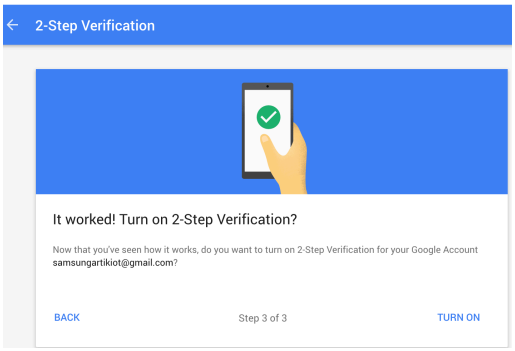
- Enter your phone number, and click “TRY IT” button.



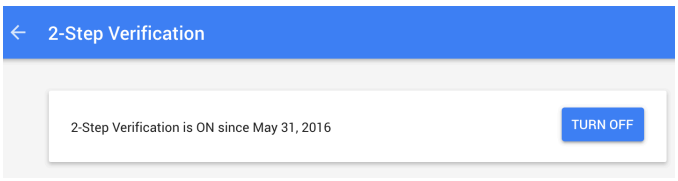
- You should be able to get a verification code via SMS. Enter the code, and click “NEXT”.



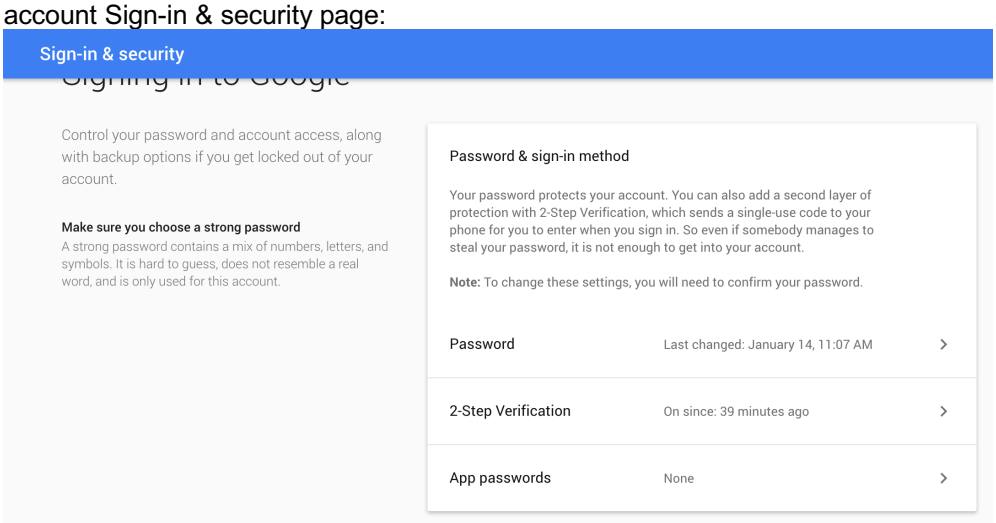
- Turn on 2-step verification.

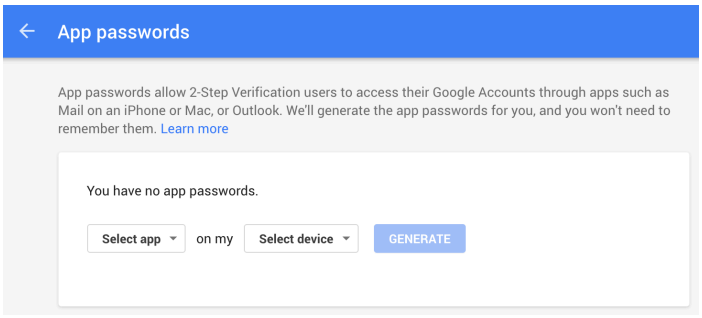


You will see the confirmation.

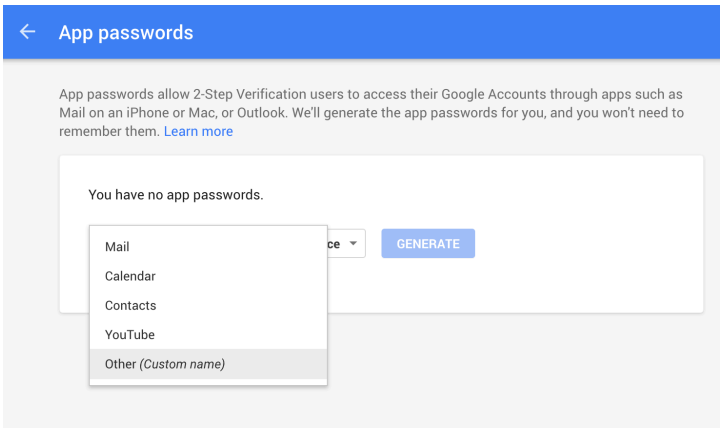


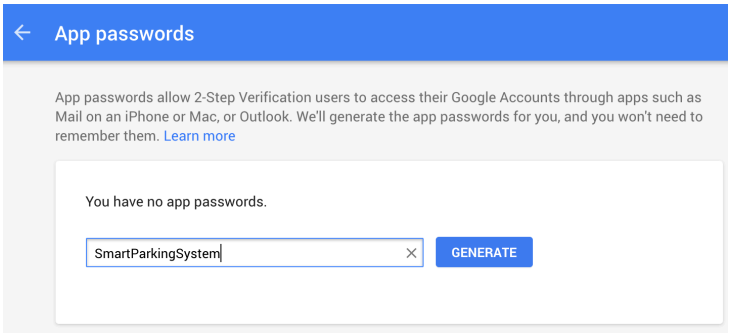
1.2 After you have enabled 2-step verification, you will be promoted to create an App Password. If you didn't get the prompt, click the "<- 2-Step Verification" link in the screenshot above, you can find the App passwords link from your account Sign-in & security page:



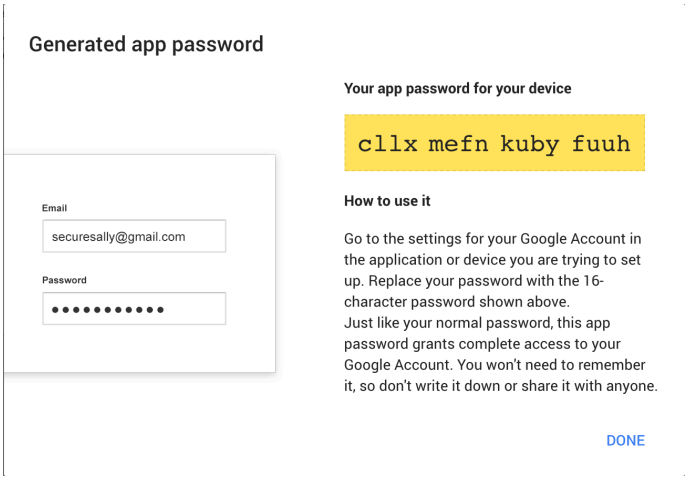


- In the “Select app” dropdown list, select “Other(Custom Name)” and enter “SmartParkingSystem” as our application name.





- Click “GENERATE” and a 16-letter passcode will be generated.



2. Auto-generate SendEmail code by using Temboo

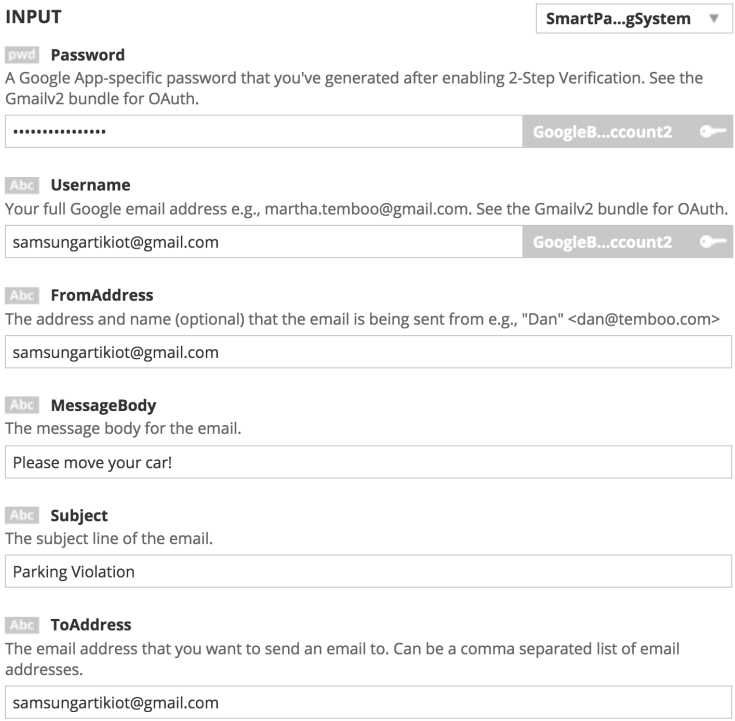
2.1 Log into your Temboo account and go to https://temboo.com/library/Library/Google/Gmail/SendEmail/ Choreo.

2.2 Turn on IoT mode 

, and select “Samsung ARTIK 10” as your target board, we will also use ARTIK’s built-in WiFi or Ethernet support to send data.

2.3 Fill out all the INPUT parameters in your SendEmail Choreo. **The password should be the 16-digit passcode generated in the step above.**

Test out the Choreo from Temboo’s website and confirm that you can receive an email.



2.4 When you've confirmed that the Choreo runs successfully, scroll down to the CODE section then download the auto-generated C code.

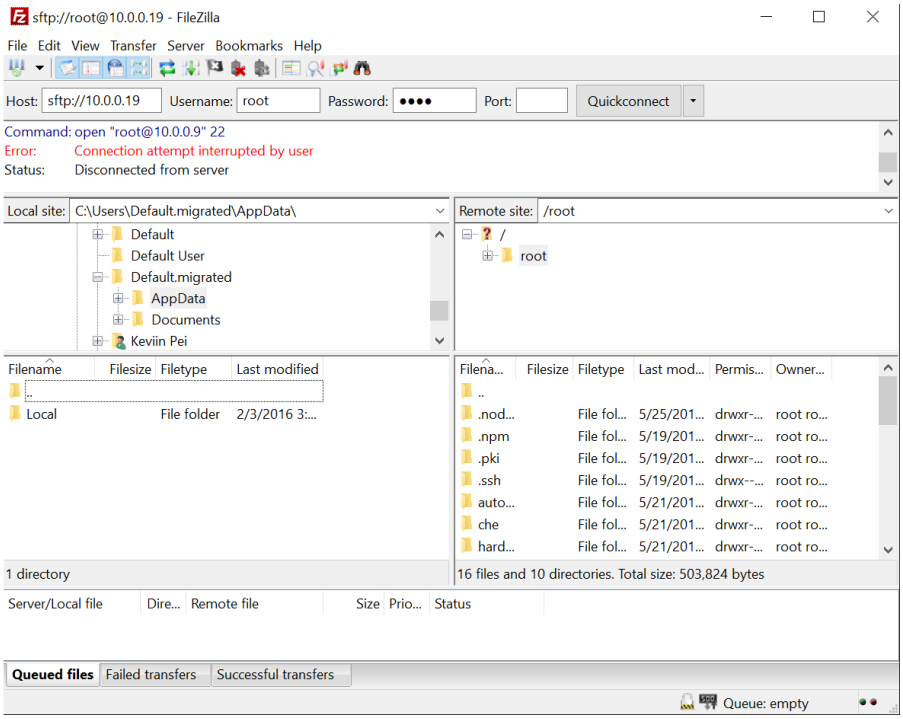
2.5 The auto-generated code references the TembooAccount.h header file, which contains your Temboo account information and network interface details. You’ll find the code for this file beneath your generated sketch. Make sure that you keep this file alongside your main C code on your ARTIK device.

2.6 Copy the Download zip file to your ARTIK board.

- On Linux or Mac host machine, use ‘scp’ command to send the zip file from your host machine to your ARTIK board.

```
# scp sendemail.zip  root@<IP_address_of_your_ARTIK>:/root
```

- On Windows machine, you can use Filezilla. Enter your ARTIK board IP address, Use “root” / “root” for username and Password. Port number should be 22. Then click “Quickconnect”. Files on ARTIK will show up on the right side in “Remote site” part. You can then drag and drop sendemail.zip from your Windows host to your ARTIK board.



2.7 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 detect a parking violation, 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

Then use one of the approaches below to tweak your sendemail.c and get it compiled
Approach 1:
[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
(Notes:
1. In the ‘sed’ command line above, when you search for “MAX_RUNS = 10”, please make sure to
include a space on each side of the equal sign.
2. The “l” in “-ltemboo” above is lower case L. the “I” in “-I/root/temboo_artik_library/include” above is
upper case “i”. )

OR Approach 2:
[root@localhost ~]# curl –O –k https://s3-us-west-2.amazonaws.com/artik/build.sh
[root@localhost ~]# bash build.sh
```

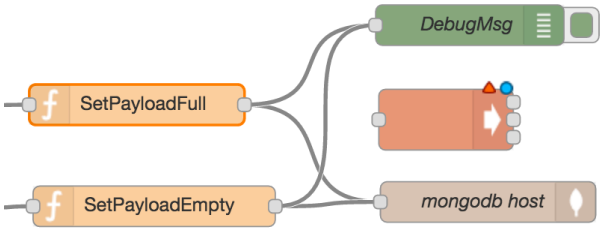
2.8 Launch ‘sendemail’ from the command line to see if you can get an email.

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

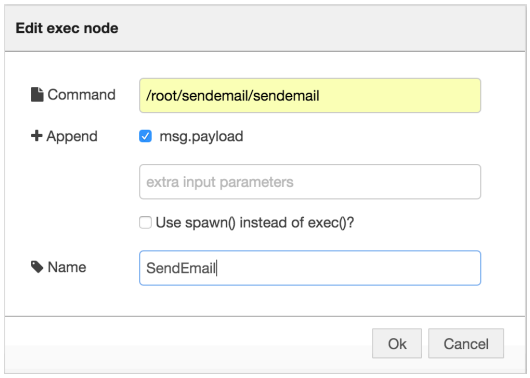
3. Update the Node-RED flow to add Temboo Web service code generated above.

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

- Drag an “exec” node, drop it under the “DebugMsg” node and above “MongoDB host” node.



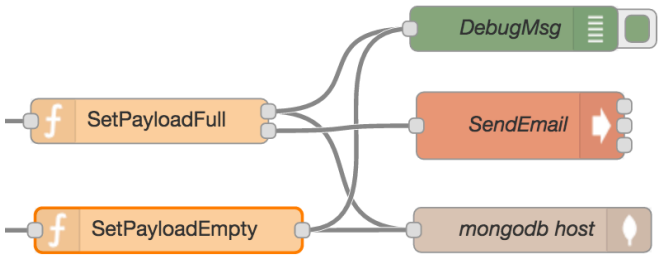
- Configure “exec” node by double-clicking. Enter “/root/sendemail/sendemail” as the Command we are going to execute. Name as SendEmail.



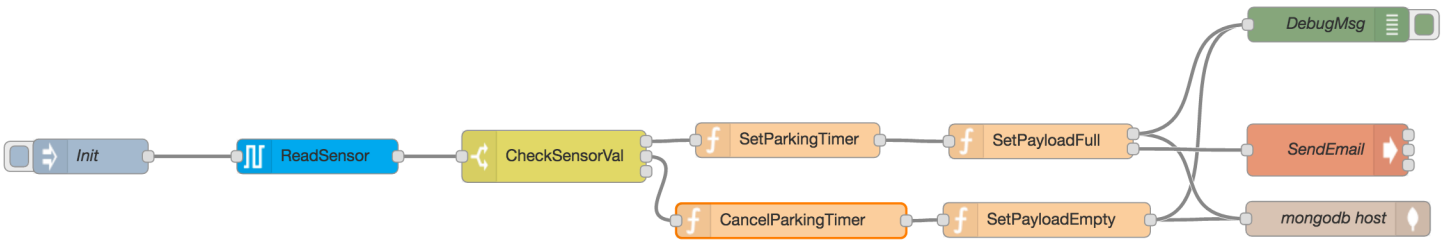
- Update “SetPayloadFull” function. Change the number of Outputs to 2, and update our function to include return values for the 2nd output.

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

msg.payload={
    "lot":"7",
    "space":"1",
    "state": "full"
}

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

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

3.2 Wire up the 2nd output of “SetPayloadFull” to “SendEmail” node.



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



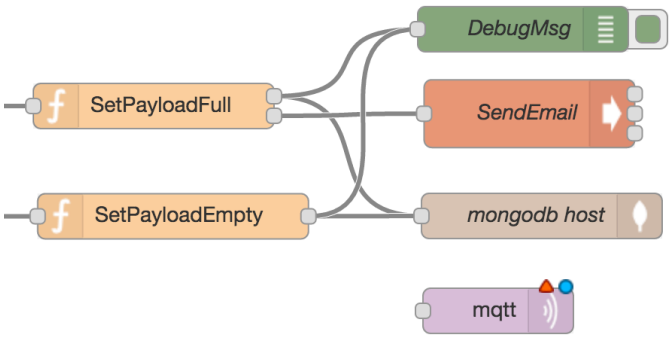
Exercise 4: Subscribe to parking space state change notification (M2M & Node-RED)

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

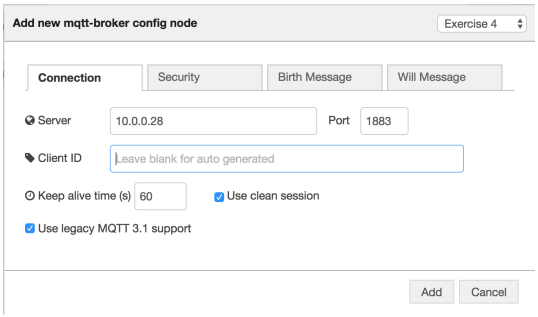
   #mosquitto –d

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

   2.1 Drag an “mqtt output” node below “mongodb host” node.

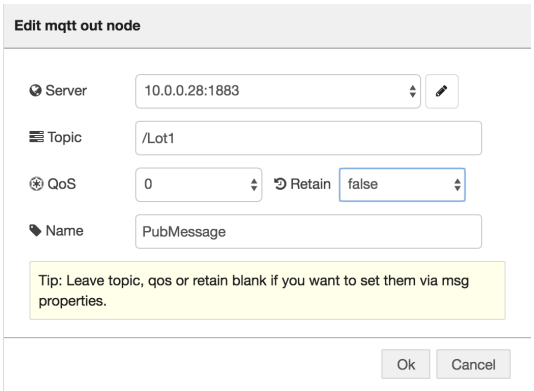


2.2 Click in the “Edit mqtt out node”dialog, “Add new mqtt-broker config node” opens up. Enter your MQTT broker’s IP address and Port number defaults to 1883.



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”.



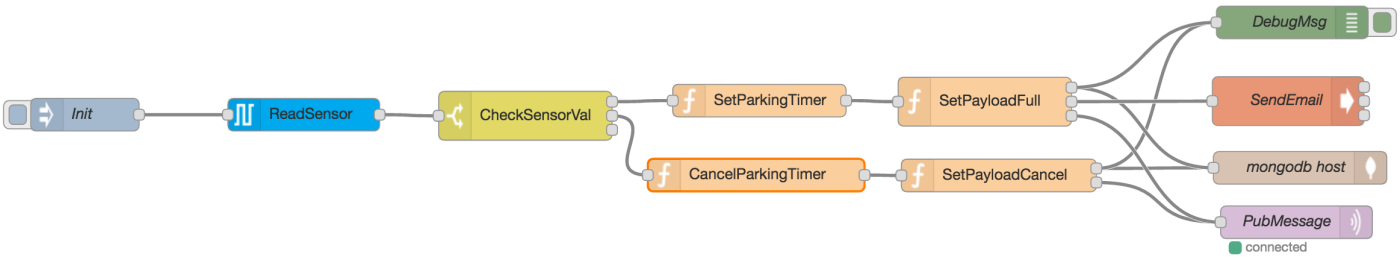
2.4 Update “SetPayloadFull” function. Change the number of Outputs to 3, and update our function to include MQTT pub message output.

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

msg.payload={
   "lot":"7",
   "space":"1",
   "state": "full"
}

msg._id="wei"
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 mqttMsg = {payload: "Lot 7 Space 1 is full"};
  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.

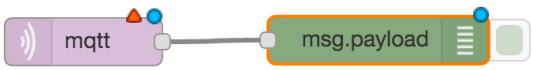
2.6 Wire up the 3rd output of “SetPayloadFull” node and the 2nd output of “SetPayloadEmpty” node to “PubMessage” node.

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

msg.payload={
    "lot":"7",
    "space":"1",
    "state": "empty",
    "expired": "no"
}

msg._id="wei";
global.set('lastState', "empty");

if (lastState === "full") {
  var mqttMsg = { payload: "Lot 7 Space 1 is empty" };
  return [msg, mqttMsg];
} else {
  return [msg, null];
}
```

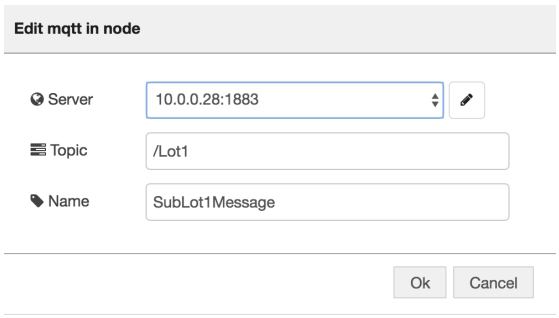


3. Subscribe to MQTT Parking space messages. Now we are creating another flow to listen for messages from a specific lot.

3.1 Add “mqtt” input and “debug” nodes, and wire them up.

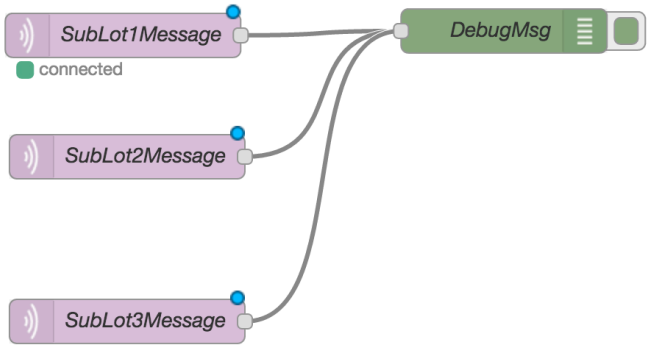


- “mqtt” input node: Set the Server address to the IP address of the broker of Lot1. Change Topic to “/Lot1”(Please replace the broker IP address and topic of the Lot# you are interested in). Rename to “SubLot1Message”(Or “SubLot2Message”, “SubLot3Message”).



Here is what the 3rd flow looks like:

You can then extend your flow to listen to space states from other lots too. Please make sure the IP addresses of the brokers match the Topics you are subscribing to. Drag another “mqtt” input node to our canvas, and drop it under “SubLot1Message” node.



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

Bonus Question:

Twilio APIs can help us extend this application to include voice and SMS functionality. From your Node-RED flow, can you send a text message when the parking space state changes? So if you are driving around the parking lots for an open space, you can get a real-time alert.

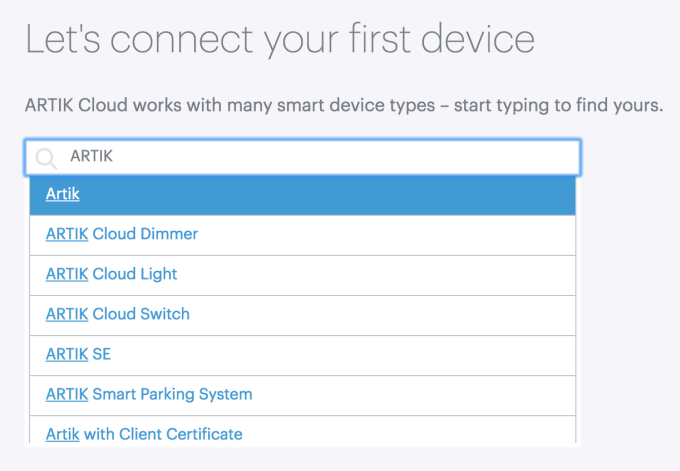
(Hints: You can use either Temboo or Node-RED for Twillio integration)

Exercise 5: Connect to ARTIK Cloud (Node-RED, ARTIK Cloud)

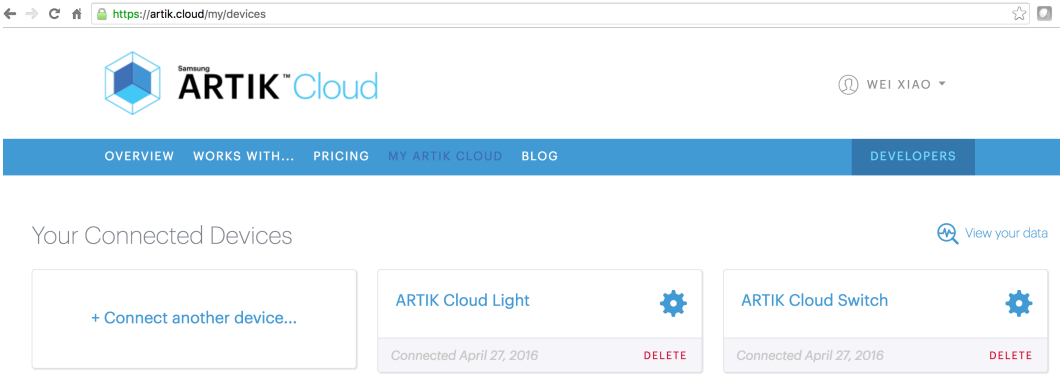
In this exercise, we are going to use ARTIK Cloud as our MQTT broker.

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:

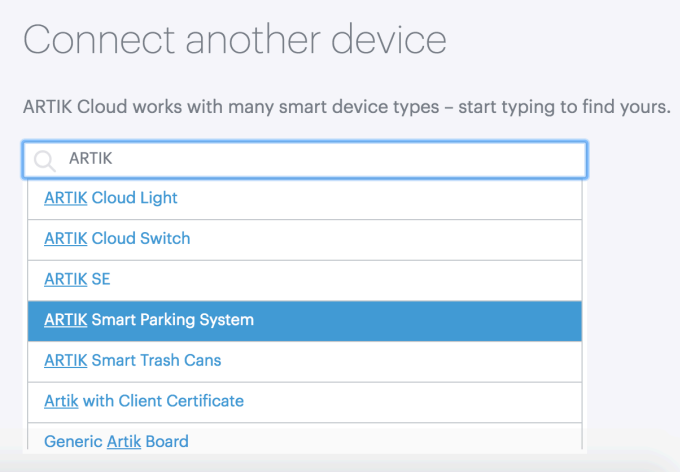


Otherwise, Under “MY ARTIK CLOUD”/”DEVICES”, you will see all you connected devices.

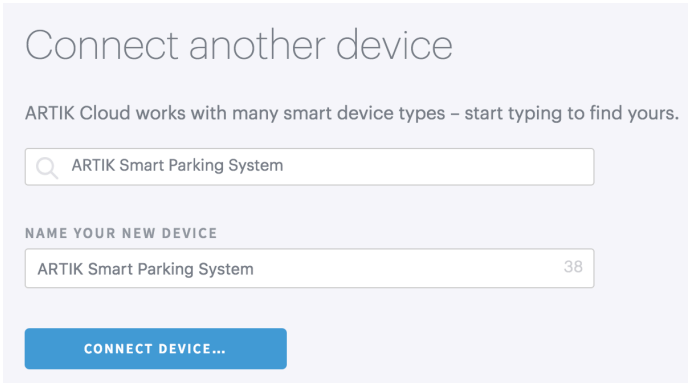


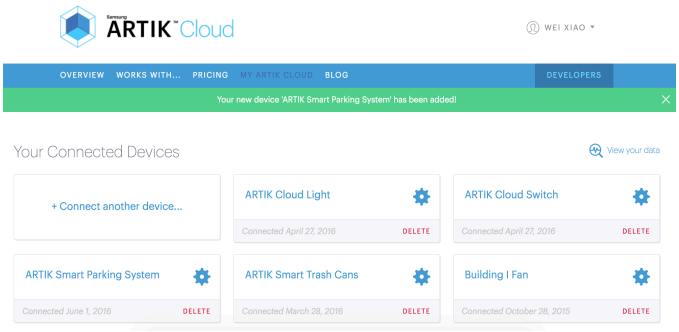
Click “Connect another device…” link.

1.2 Search for “ARTIK Smart Parking System” and select it. This is a public device we created for the workshop, which includes “lot” and “address” of the parking space.

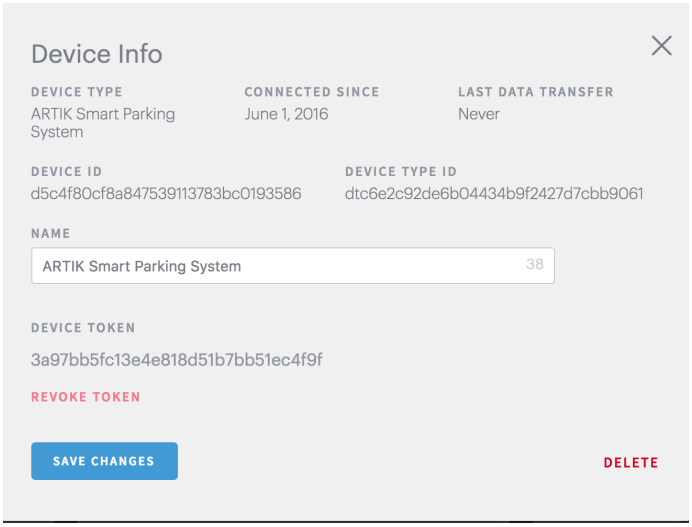


1.3 Use the default device name “ARTIK Smart Parking System”, then click the “CONNECT DEVICE…” button. A new device will be created.





1.4 Click the Gear icon next to your 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** (not “DEVICE TYPE ID”) and **Device Token** to associate our parking space with ARTIK Cloud. Make a copy or leave the dialog open.

2. Now, Let’s go back to our Node-RED flow.

2. 1 Update “SetPayloadFull” function to convert our message payload format for “ARTIK Smart Parking System” device type. Replace the bolded part of msg.topic below with your own Device ID (from step above). Click “OK” to close the dialog.

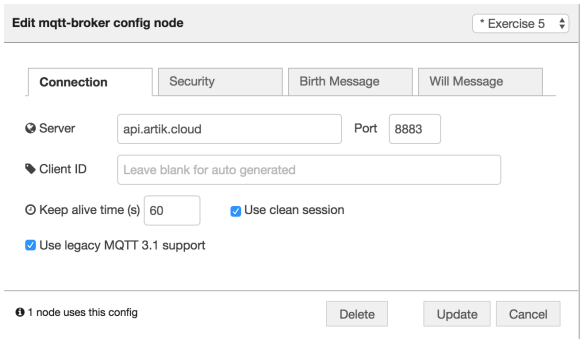
```
msg.topic = "/v1.1/messages/d6e5661d9ad146c5b0fcc99e922347c6";
var lastState = global.get('lastState');
var expired = global.get('expired');
msg.payload={
    "lot":"7",
    "space":"1",
    "state": "full"
}

msg._id="wei"
global.set('lastState', "full");
if (expired === true) {
    msg.payload.expired = "yes";
    return [msg, msg, msg];
} else if (lastState === "full") {
    msg.payload.expired = "no";
    return [msg, null, msg];
} else if (lastState === "empty") {
    msg.payload.expired = "no";
    var mqttMsg = {payload: "Lot 7 Space 1 is full"};
    return [msg, null, msg];
}
```

2.2 Update “SetPayloadEmpty” function as well to convert our message payload for “empty” state. Replace the bolded part of msg.topic below with your own Device ID

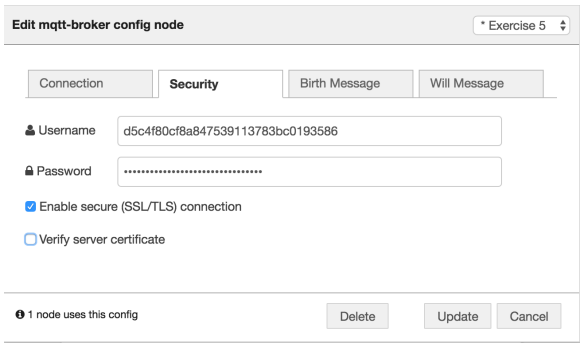
```
var lastState = global.get('lastState');
msg.topic = "/v1.1/messages/d6e5661d9ad146c5b0fcc99e922347c6";
msg.payload={
    "lot":"7",
    "space":"1",
    "state": "empty",
    "expired": "no"
}

msg._id="wei";
global.set('lastState', "empty");

if (lastState === "full") {
  var mqttMsg = { payload: "Lot 7 Space 1 is empty" };
  return [msg, msg];
} else {
  return [msg, msg];
}
```

2.3 Configure the “PubMessage” node to use Samsung Cloud’s MQTT server.

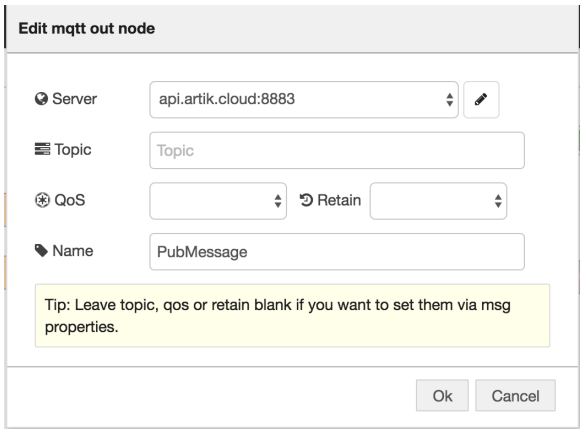
- Click  in the “Edit mqtt out node” dialog. In “Edit mqtt-broker config node”: Use “api.artik.cloud” as the Server address, 8883 as the port number.



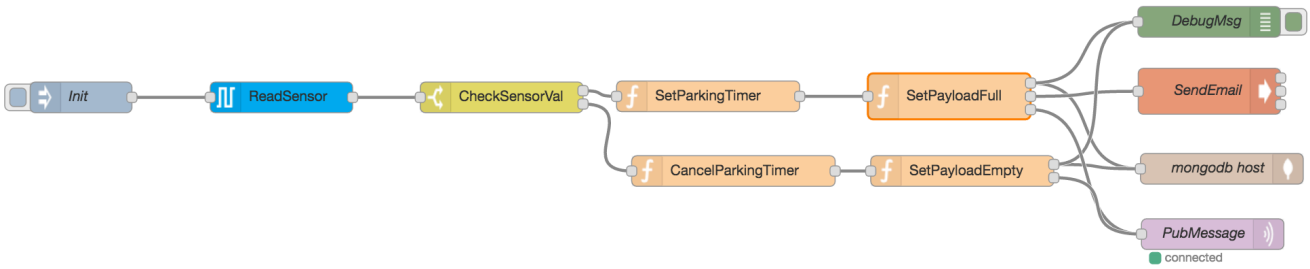
- “Security” Tab: From the Device Info window, copy and paste your Device ID as Username, and Device Token as Password. Make sure “Enable secure (SSL/TLS) connection” is checked. Click the “Update” button.



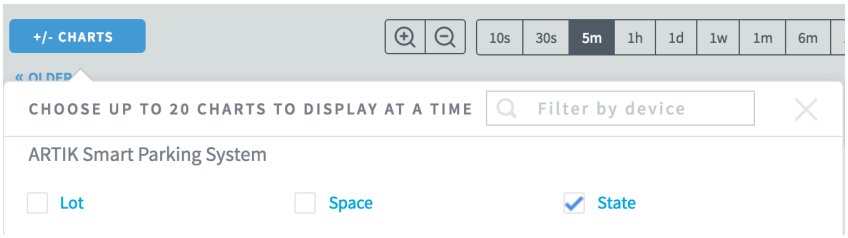
- In “Edit mqtt out node”, leave the Topic blank.



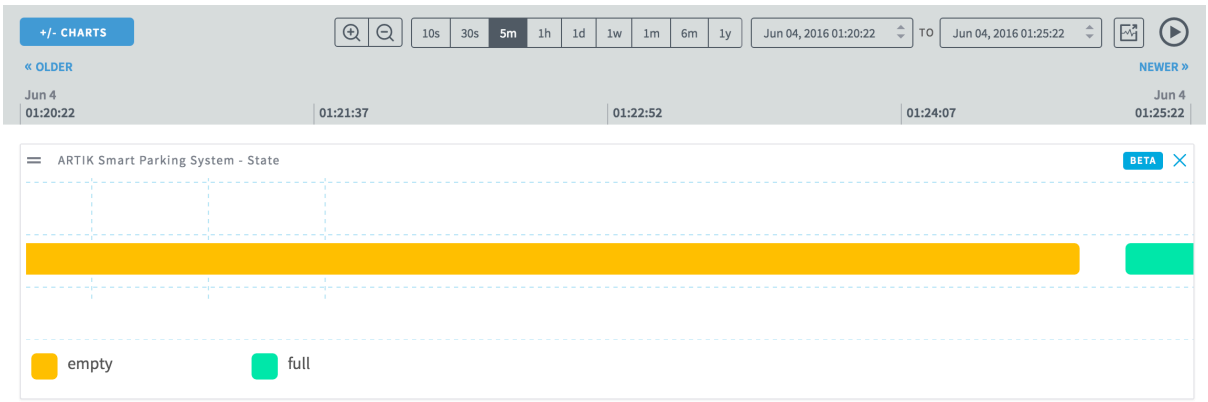
This is what your final flow should look like:



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



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



Bonus Question:

ARTIK Cloud offers Rules Engine capabilities, where you can use a rule to trigger a device action.

- Connect a RED LED to your ARTIK Board
- Add a rule that “If the parking space is reserved, RED LED should be turned on”.

Could you design your hardware and software to simulate? Once the rule is enforced, the RED LED can be turned on?