SMART SENSING TOW

Dipal Patel
Minal Shettigar
Samantha Rodrigues
Esha Jhamb

Index

Chapter 1 - Inspiration behind the project

Problems with marking of tires methodology

To track how long vehicles have been parked in time-restricted zones in the parking lot, officers use a chalk stick to mark the tires of the parked vehicles. They then return after the time restriction has passed, and if the vehicle is still parked there in violation of the time zone restriction, a ticket is issued. This vehicle must travel a minimum of one mile before it can legally park in another time-restricted zone.

Marking tires for parking enforcement has to be targeted to a smaller populated area for the best results. If a cop spreads out their work too much, they could get confused about who is really parked for too long.

Using chalk may have its limits and can easily be wiped off. Catching an illegal parker is easier for a 30 minute stall compared to a 24 hour one. The short time frame makes the work more routine, whereas a 24 hour lot may cause the cop to have to remember who was where. What if the person, who parked after 24 hours, looked exactly in the same position when they returned? The time length allowed to park is based on frequency of use and population.

Problems without Parking Meter

Parking meter is an essential tool to manage on-street parking and generate revenues in the crowded public places. But in some places like parking spot of an airport/store/university/offices where there is no parking meter, it is difficult to monitor and manage which violates the right to park a vehicle in a particular place for a limited amount of time.

Accidental Deaths in car

You might have read in the newspaper about the accidental deaths in the car in the parking spots and people noticed it after death occurs. Here are two cases which blew up our mind while exploring the ideas about projects. (1) Dead man sat in truck at airport parking lot for eight months and no one noticed[1]. (2) California woman found dead in Walmart parking lot, may have been hidden in her car for months[2]. Sometimes people die in the car at parking lot and people did not notice it and their dead bodies are found after months. In both the above cases, the car was parking in airport parking lot and walmart parking lot.

Benefits from Smart towing system

Smart towing system is a mechanism which enables the user to park their car for a limited amount of time as per the rule. We will use motion sensor to measure the movements of car parked at a parking space which has bounded time restriction. If the car is parked for more than X time then a message will be sent to the towing company to tow the car from that parking spot.

Therefore, it will mainly focus on reducing the time in finding the parking slots and also it avoids accidental death in parked car. This parking solution can greatly benefit the user, towing agency and the lot owner. Here are some of the top benefits:

1. Optimized parking – Users find the best spot available, saving time, resources and effort. The parking lot fills up efficiently and space can be utilized properly by commercial and corporate entities.

2. Reduced traffic – Traffic flow decreases as fewer cars are required to drive around in search of an open parking space.

3. Reduced pollution – Searching for parking burns around one million barrels of oil a day. An optimal parking space availability will significantly decrease driving time, thus lowering the amount of daily vehicle emissions and ultimately reducing the global environmental footprint.

4. Decreased Management Costs – More automation and less manual activity saves on labor cost and resource exhaustion.

5. Towing company benefits - Automated SMS will save time, energy and efforts to check parking spot regularly.

6. Lifesaver - Smart sensing towing might help to avoid accidental death or at least might help to inform family.

Future Enhancement

The future enhancement to the project will be smart parking system. Smart Parking system involves the use of low cost sensors, real-time data and applications that allow users to monitor available and unavailable parking spots. The goal would be to automate and decrease time spent manually searching for the optimal parking floor, spot and even lot. Some solutions will encompass a complete suite of services such as online payments, parking time notifications and even car searching functionalities for very large lots**.** This enhancement would give more benefits to the users.

References:

[1]
http://nypost.com/2017/09/18/dead-man-sat-in-truck-at-airport-parking-lot-for-eight-months-and-no-one-noticed/

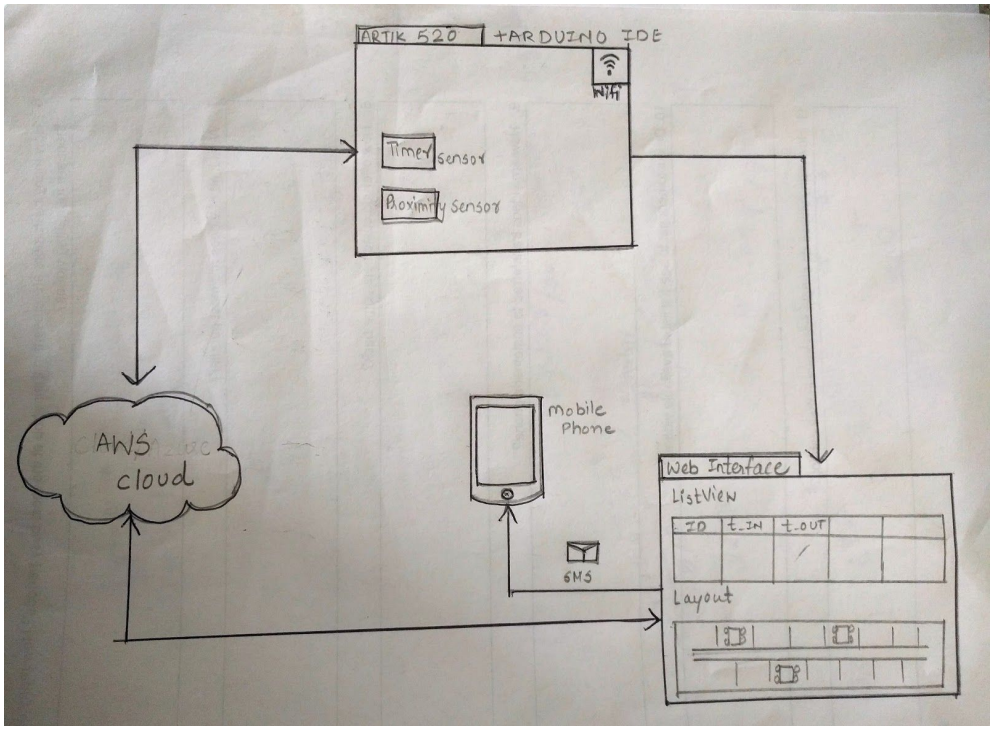
[2]
http://www.nydailynews.com/news/national/california-woman-found-dead-hidden-car-parking-lot-article-1.2527585

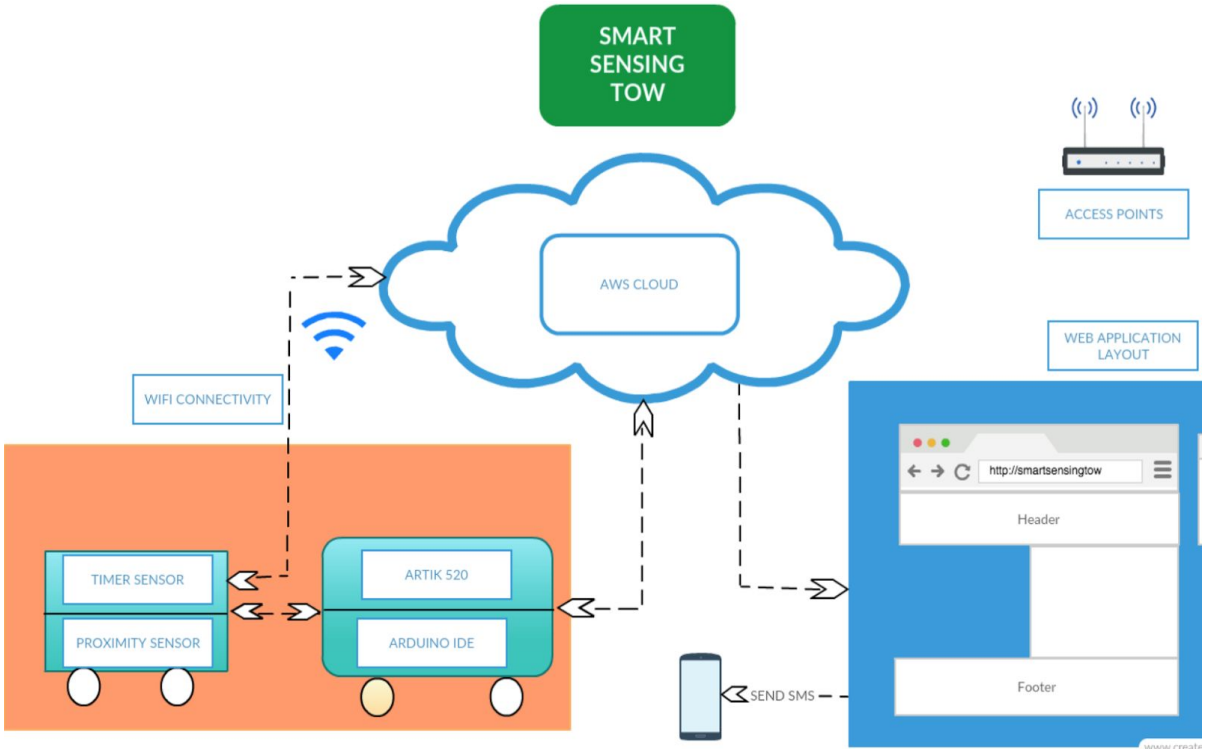
Chapter 2 - Introduction

Internet of Things plays a vital role in connecting the surrounding environmental things to the network and its makes easy to access those un-internet things from any remote location. Generally people face a lot of problems while parking vehicles either on roads or parking spaces as these spaces are already occupied by other vehicles which are not bounded by a particular time. In this project, we are planning to implement a mechanism which enables the user to park their car for a limited amount of time as per the rule. We will use motion sensor to measure the movements of car parked at a parking space which has bounded time restriction. If the car is parked for more than X time then a message will be sent to the towing company to tow the car from that parking spot. Therefore, it will mainly focus on reducing the time in finding the parking slots and also it avoids accidental death in parked car.

For example: At Santa Clara university's parking vicinity area, there is a parking time limit of 4 hours but no one actually monitors it. We can intelligently make use of motion sensors and IOT in that parking spot and measure the duration of the parking. If the time limit exceeds then our application will send message to the towing company to tow the car. In this way it will avoid unnecessary congestion of cars parked inside the parking areas.

Block diagram





Description of the application

The scope of this project is covering the private parking space. The board and all other attached components would be mounted on the wall/floor or on the curbside of the parking space. The proximity sensor/IR sensor would detect the presence of the car in the allotted slot. Once a car is detected by the sensor, it will trigger the timer circuit within the board, thus running a loop to keep track of the duration the car is being parked. If this duration exceeds a defined time, the parking slot number along with the duration will be passed onto the cloud(AWS). This data from the cloud will then be fetched by the web interface. If the duration of parking exceeds, a message would be sent to the mobile number of towing company using Twilio application. Only after the car from said parking slot is shifted will the timer circuit in the board be reset to initial state, thus ready to track data for another parked car.

Bill of Material

Sr. No	Quantity	Components	Manufacturer
1	1	KITRA 520, ARTIK 520	Samsung
2	1	Proximity Sensor	Samsung
3	1	USB cables	Allde
4	1	Motors and wheels (Toy car)	Amazon
5	1	WiFi - Router	Comcast

Chapter 3 - Setup and Execution

Setup steps

- For Windows users, pre-install PuTTY and Filezilla.
  PuTTY (http://www.putty.org/) – SSH and Telnet client, for serial console access

- Create Twilio account and generate number. (Appendix A)

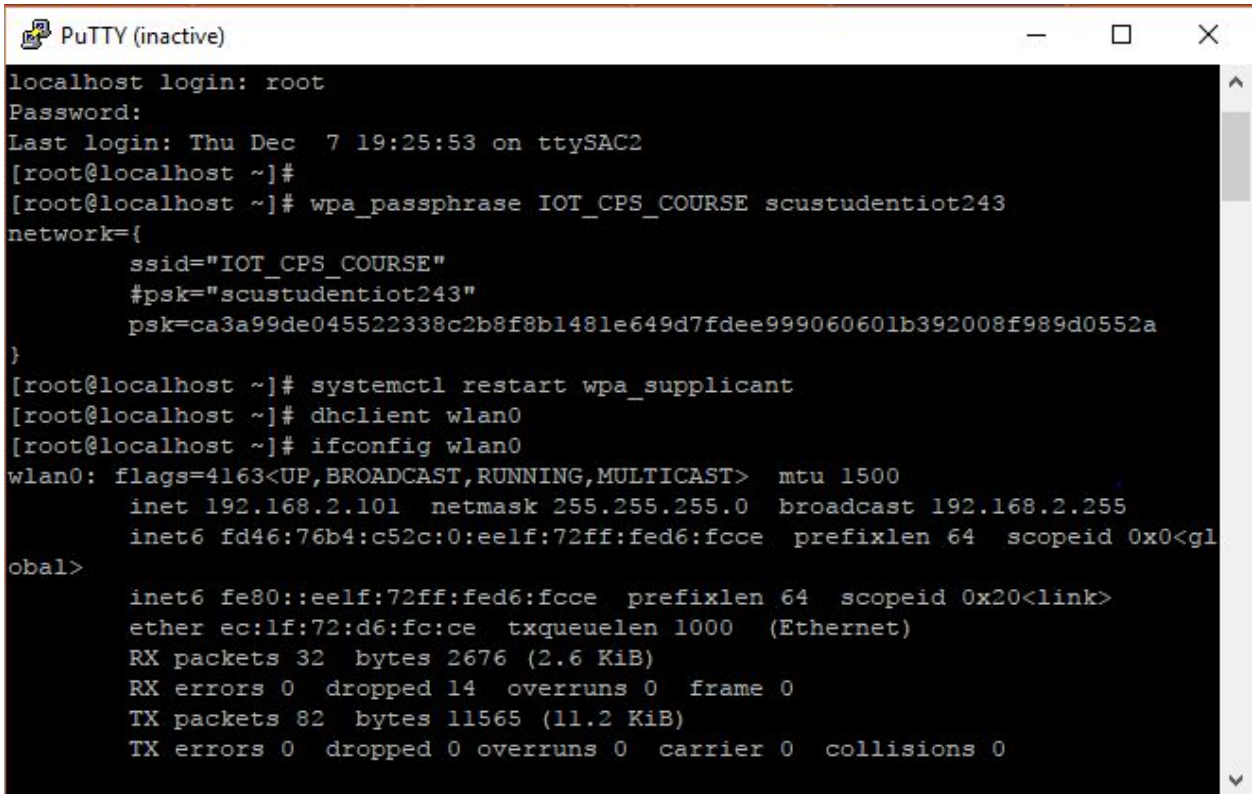
- Establish an ARTIK Cloud user portal account: We will stream data to ARTIK Cloud service. Create an account at ARTIK Cloud user portal (https://artik.cloud/). (Appendix B)

- Create MongoDB account (Appendix C)

- Create Web Interface. (Appendix D)

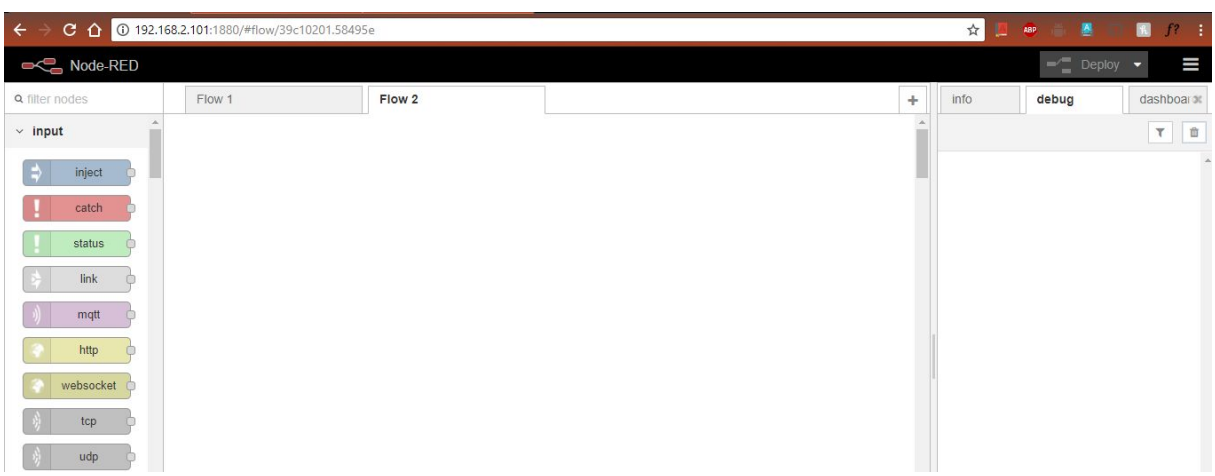
Execution

1. Connect board with USB to the computer. Open device manager and check the port of the connected board (ex:COM8). Open putty- Select serial, add port number (COM8) and baud rate =115200. Type username = root and password= root and you will be logged in local root console.
2. You have to make sure that your computer and board are in connected in the same wifi network. To connect the board to the wifi. Perform the following commands:
   [root@localhost ~]# wpa_passphrase IOT_CPS_COURSE scustudentiot243 //Wifi SSID and password
   [root@localhost ~]# systemctl restart wpa_supplicant
   [root@localhost ~]# dhclient wlan0 // to get IP address of board
   [root@localhost ~]# ifconfig wlan0 // to see the IP address



[root@localhost ~]# node-red & // to start the flow in node-red

3. Open web browser and type IP address followed by :1880 to see the node-red flows
   -> 192.168.2.101:1880

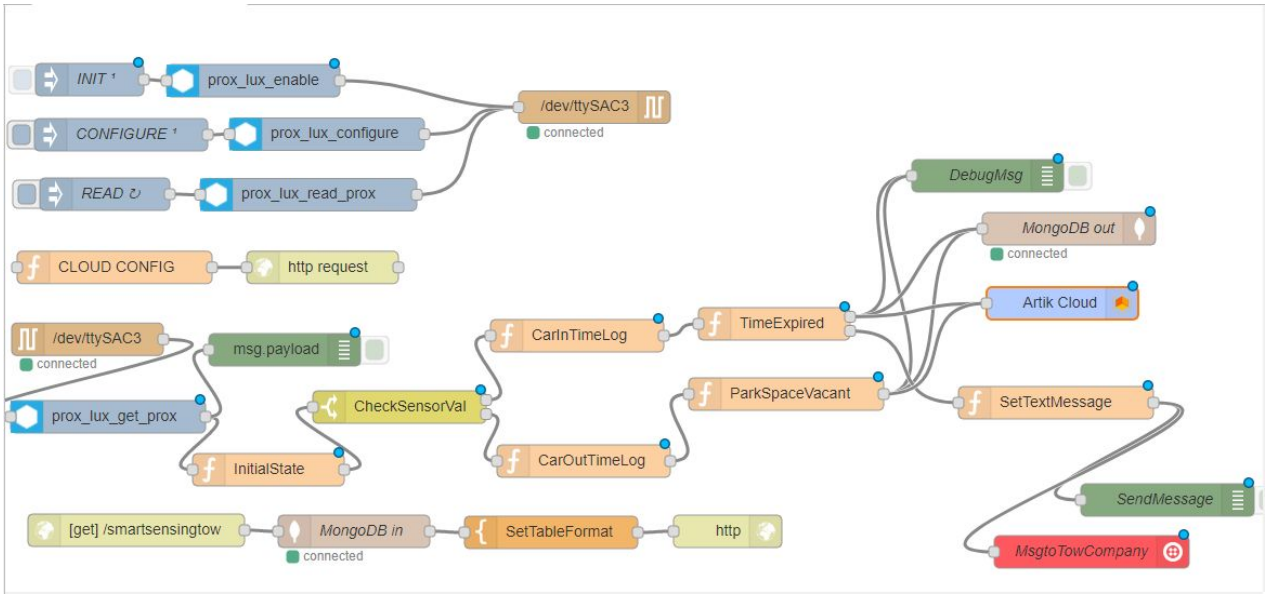


4. Once the node-red is open. Start creating flows using nodes.

Creating Node-red Flows

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.

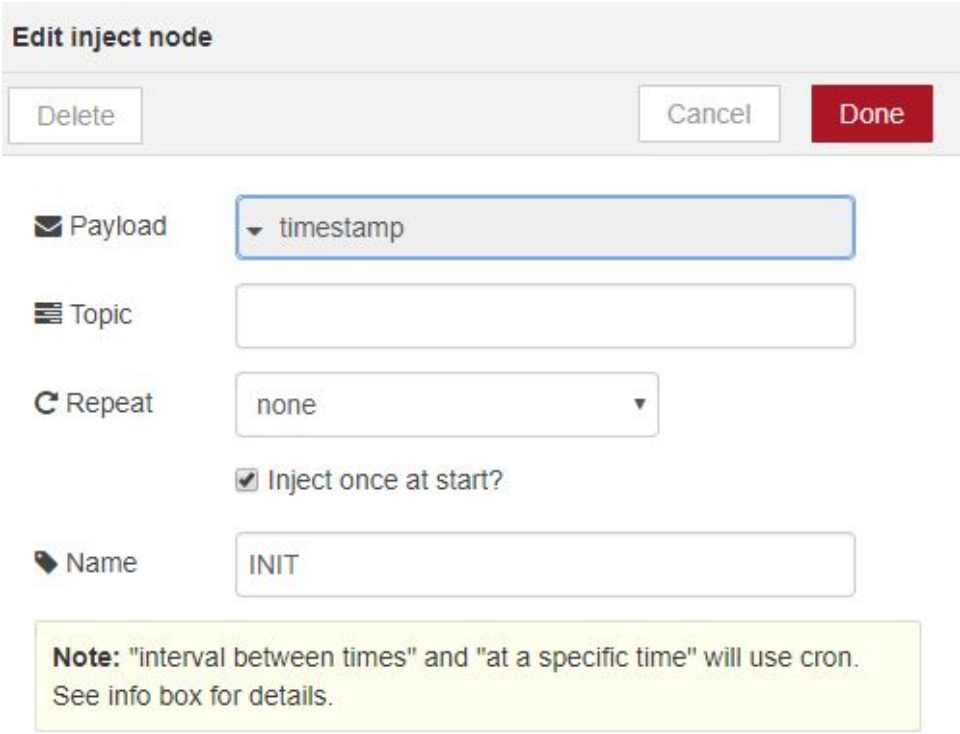
In the same way - Create the flow as shown below:



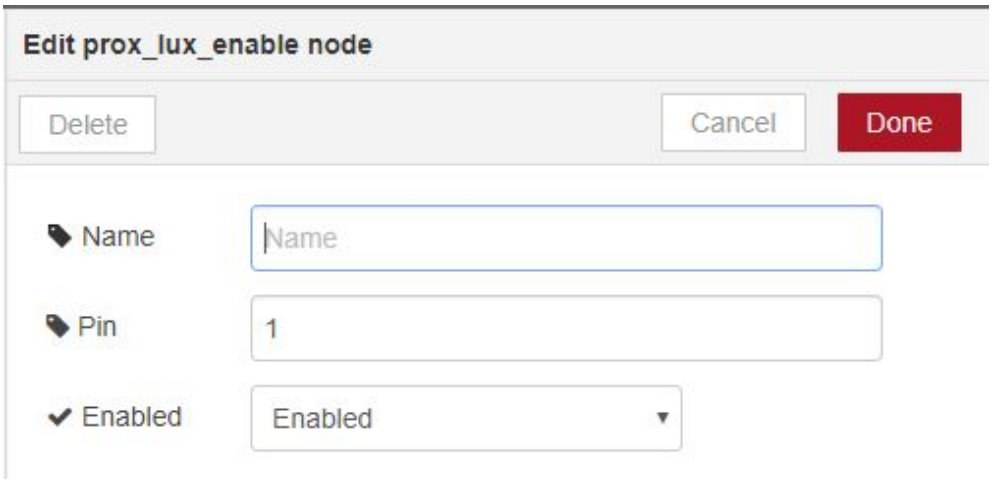
Configure each node by double clicking it and changing code when required.

Configurations snapshots and code:

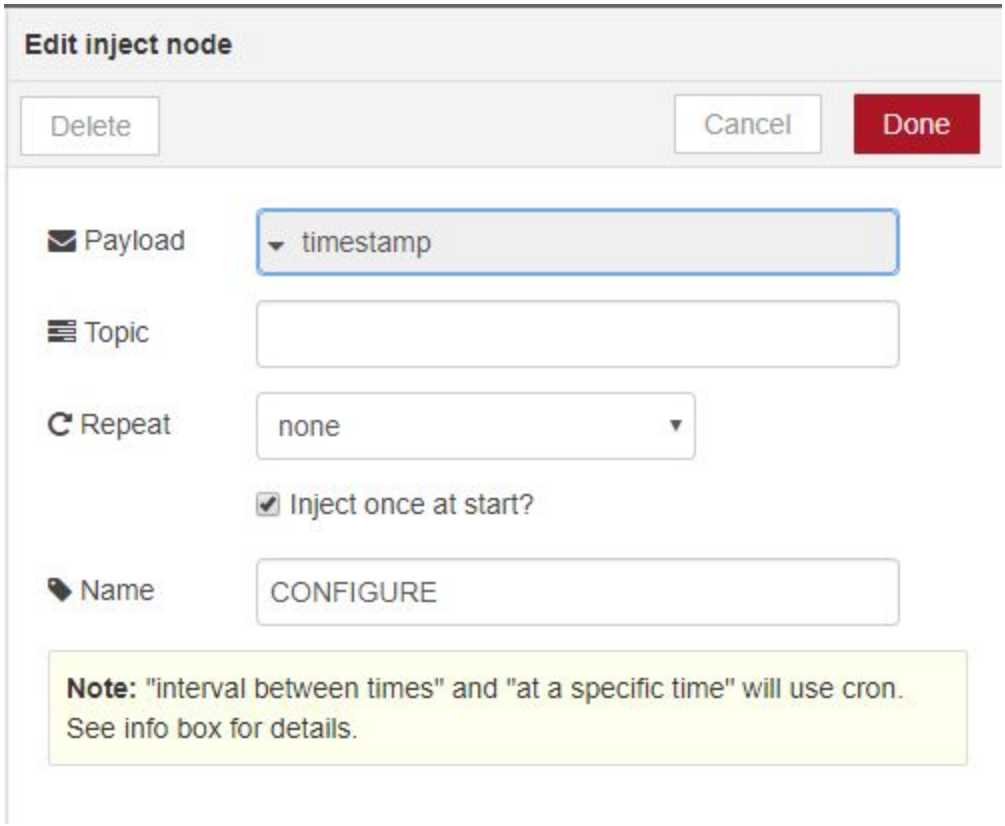
(1) INIT



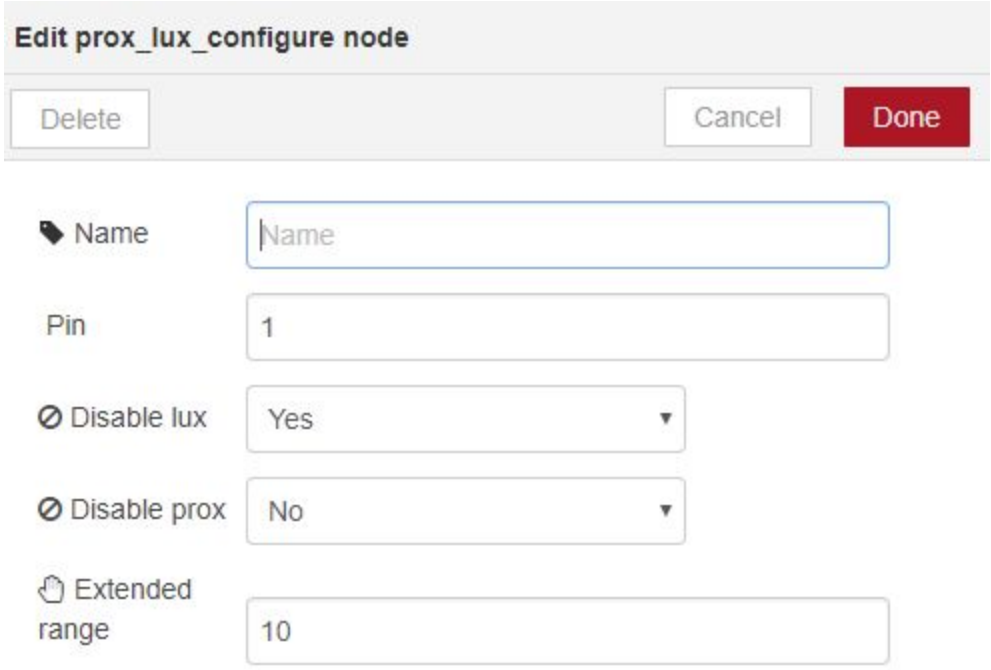
(2) Prox_lux_enable (Inject)



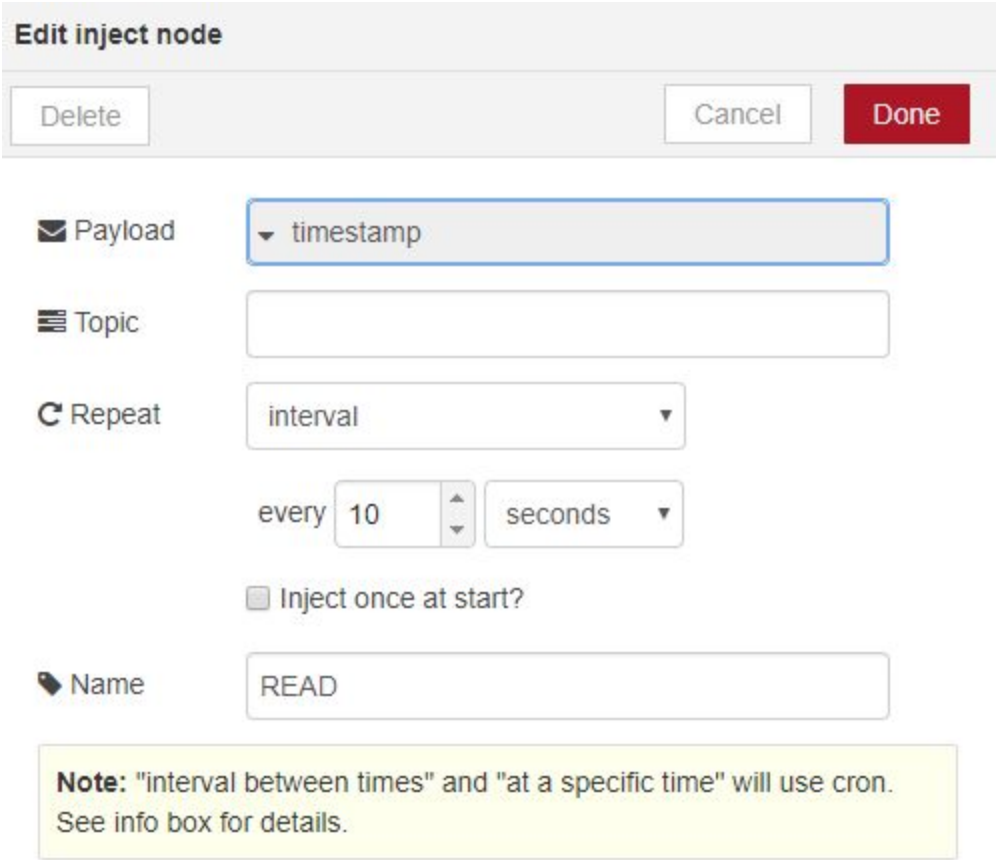
(3) CONFIGURE (Inject)



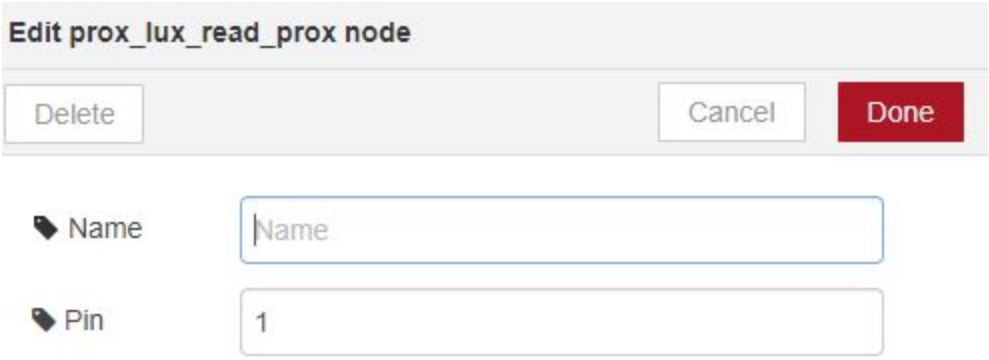
(4) Prox_lux_configure (kitra Input)



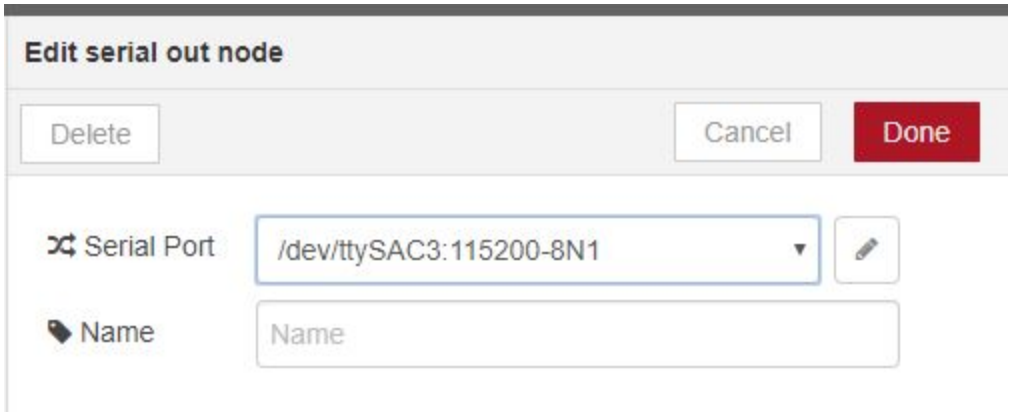
(5) READ (Inject)



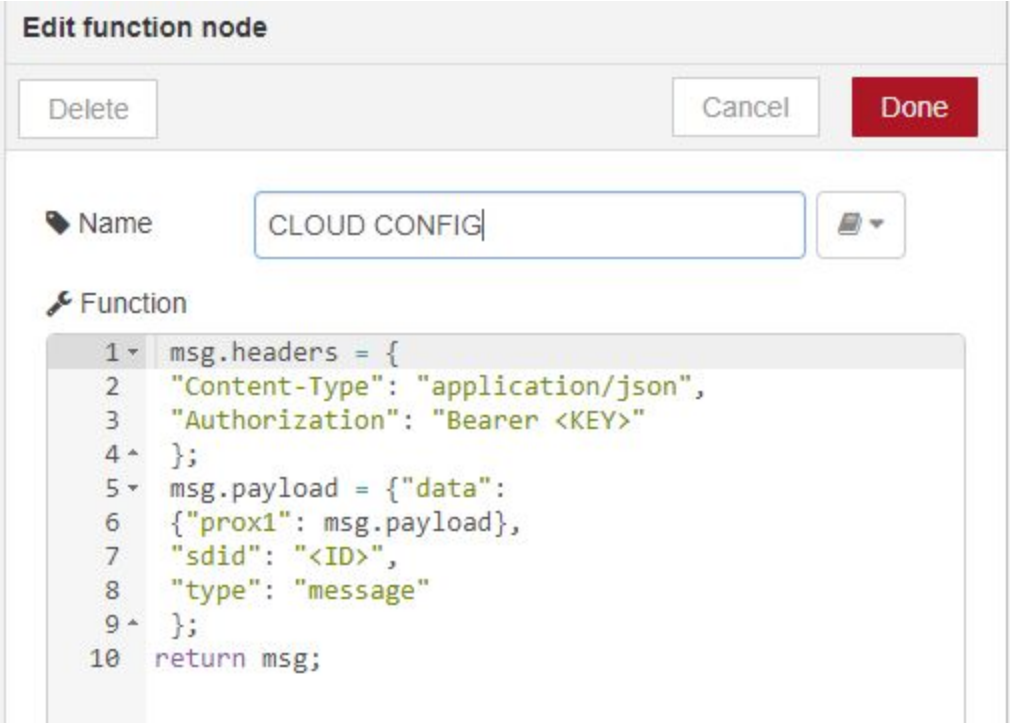
(6) Prox_lux_read_prox (kitra Input)



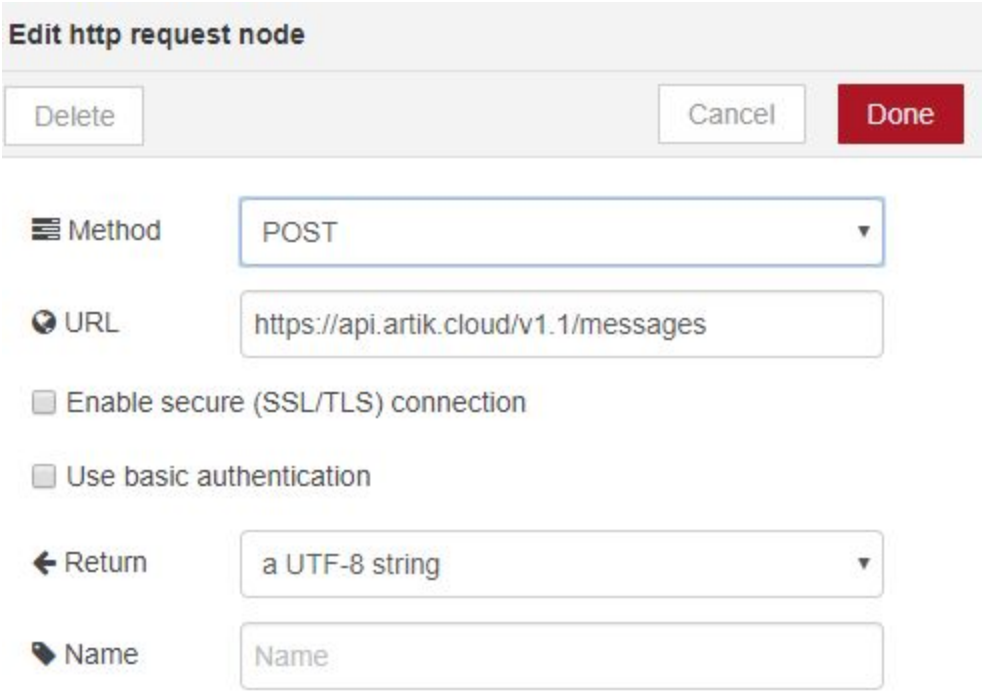
(7) /dev/ttySAC3 (serial out)



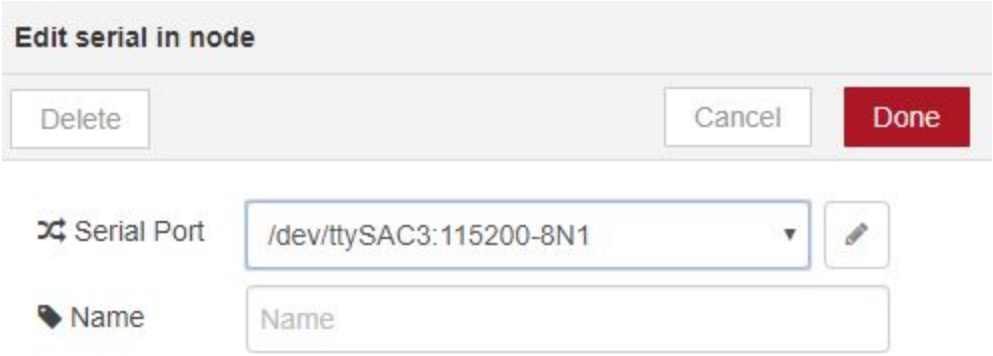
(8) CLOUD CONFIG (function)



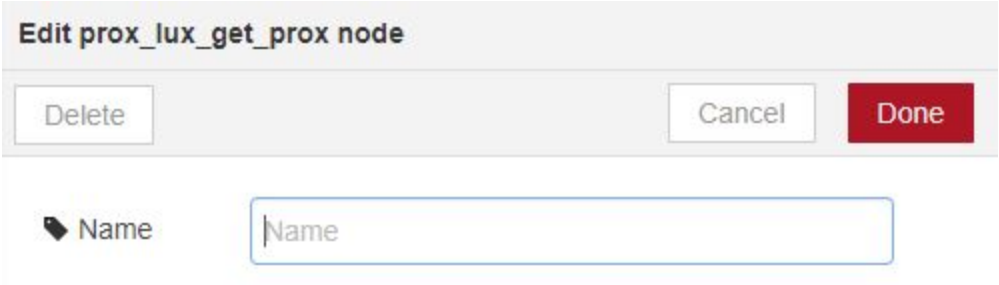
(9) Https request (function, https request)



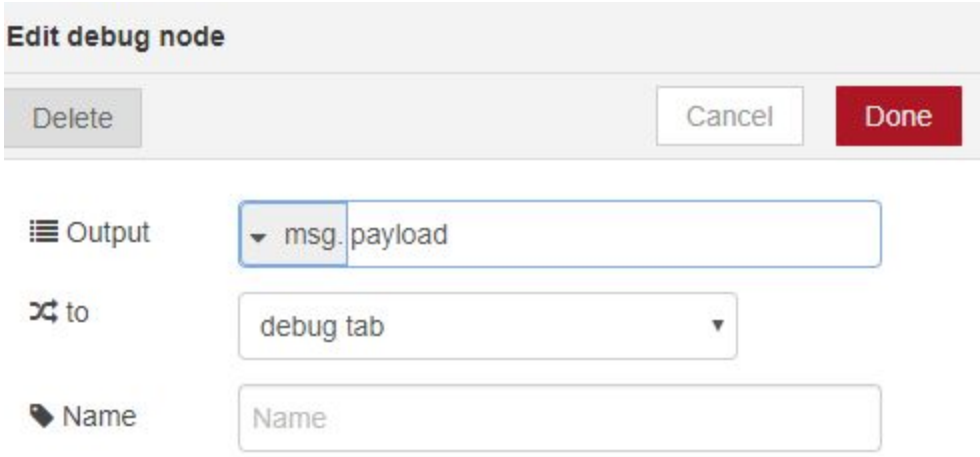
(10) /dev/ttySAC3 (serial in)



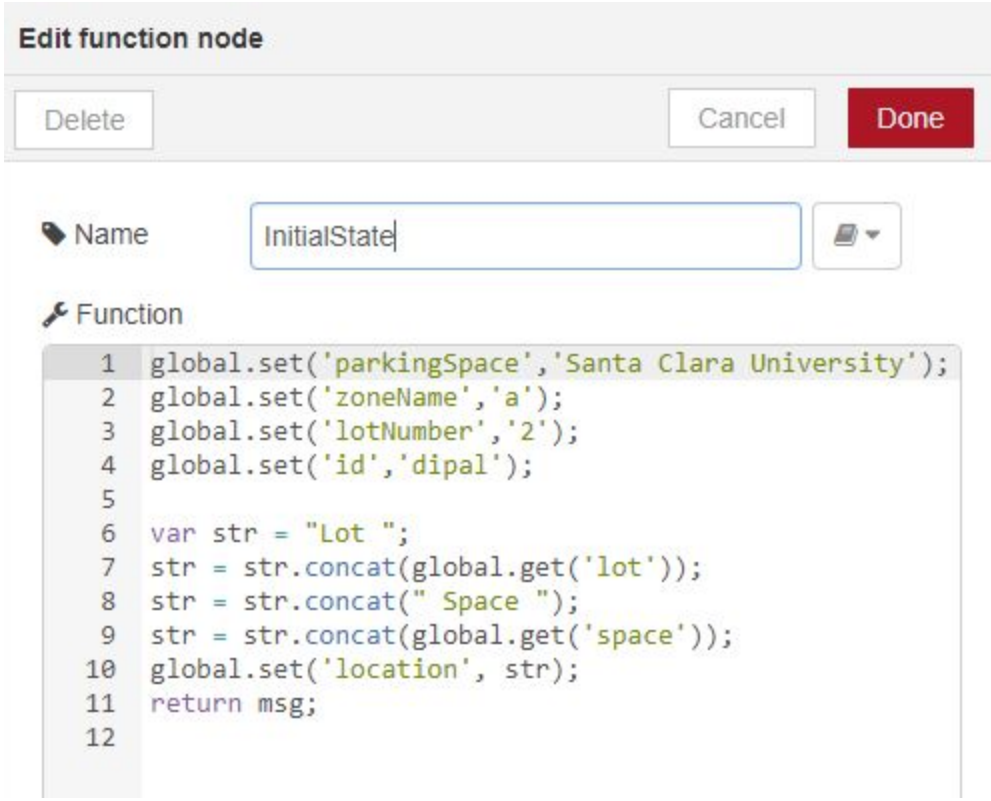
(11) Prox_lux_get_prox (kirta Input)



(12) Msg.payload (debug)



(13) Initial State (function)

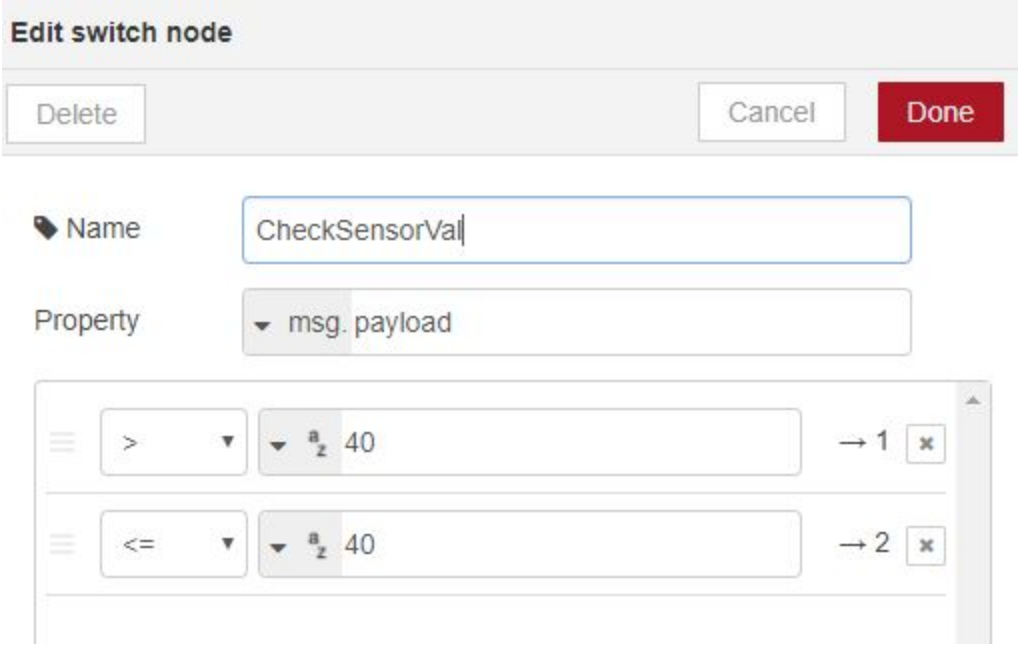


Code:

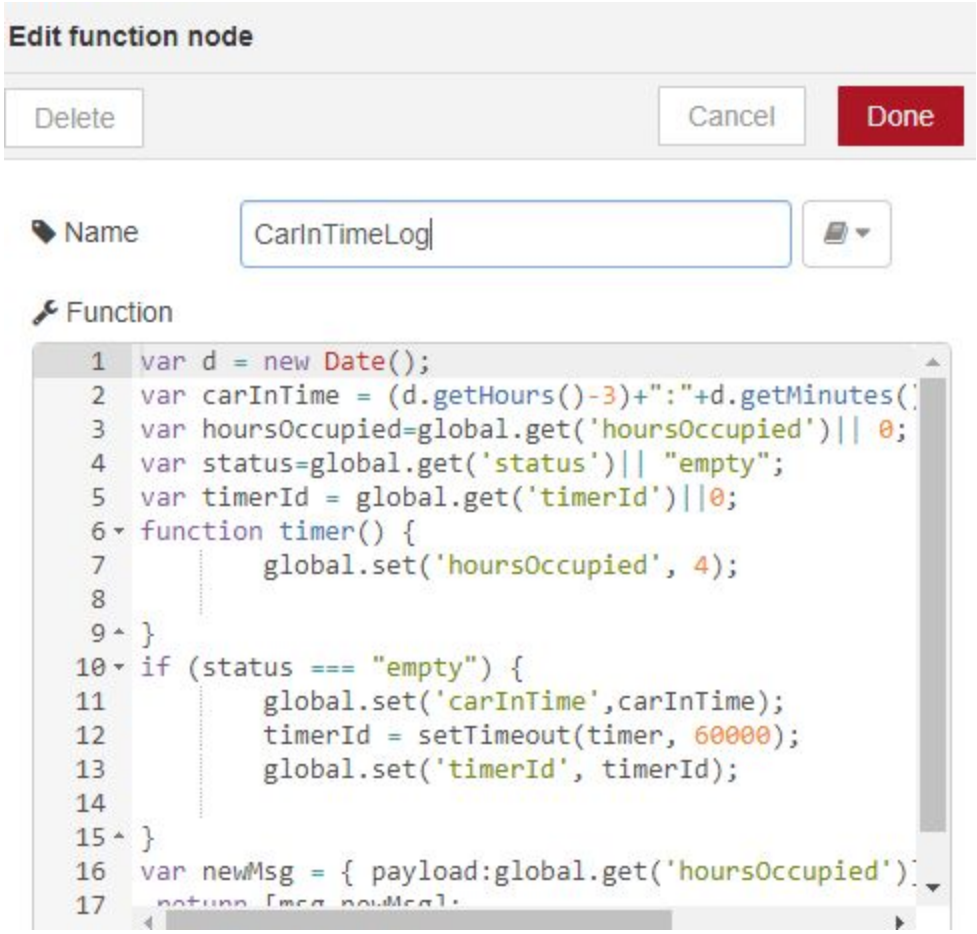
```
global.set('parkingSpace','Santa Clara University');
global.set('zoneName','a');
global.set('lotNumber','2');
global.set('id','dipal');
var str = "Lot ";
str = str.concat(global.get('lot'));
```

```
str = str.concat(" Space ");
str = str.concat(global.get('space'));
global.set('location', str);
return msg;
```

(14) CheckSensorVal (change node)



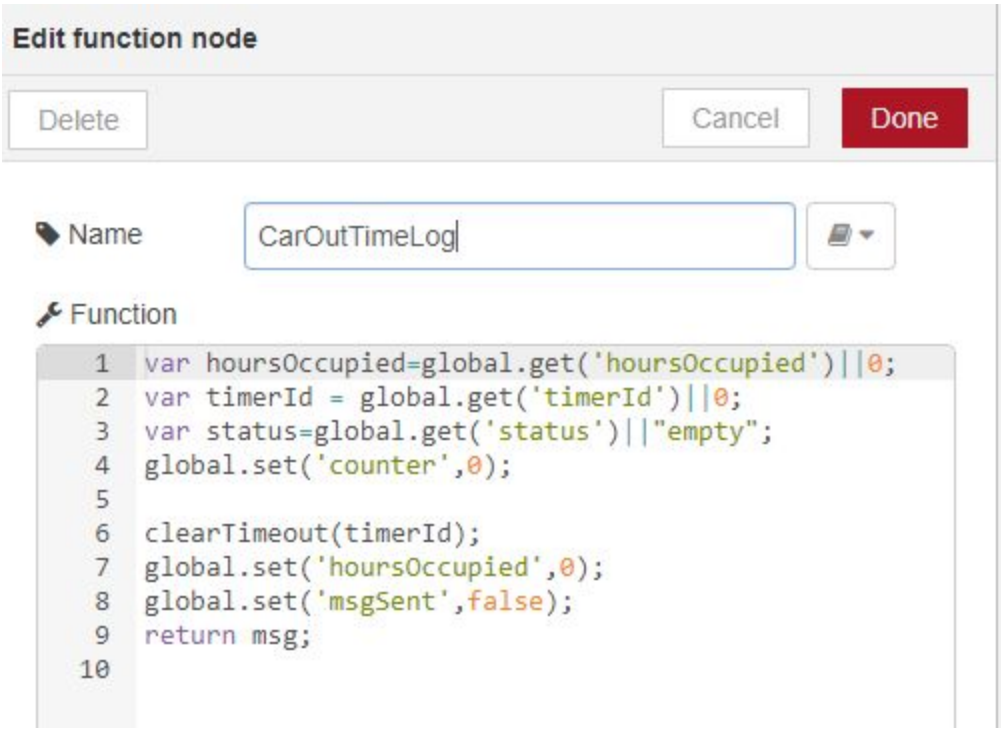
(15) CarInTimeLog (function)



Code: var d = new Date();

```
var carInTime = (d.getHours()-3)+":"+d.getMinutes()+":"+d.getSeconds();
var hoursOccupied=global.get('hoursOccupied')|| 0;
var status=global.get('status')|| "empty";
var timerId = global.get('timerId')||0;
function timer() {
    global.set('hoursOccupied', 4);   }
if (status === "empty") {
    global.set('carInTime',carInTime);
    timerId = setTimeout(timer, 60000);
            global.set('timerId', timerId);  }
var newMsg = { payload:global.get('hoursOccupied')};
 return [msg,newMsg];
```

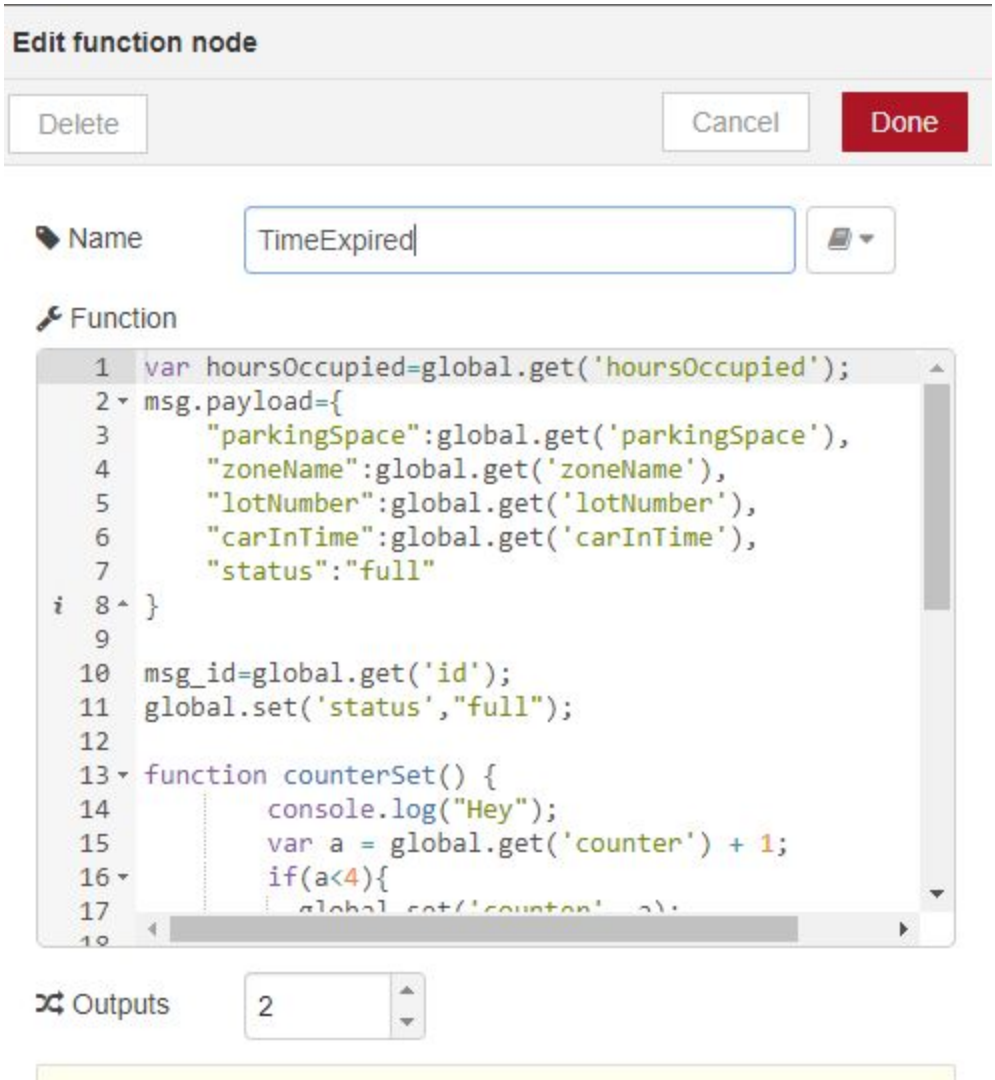
(16) CarOutTimeLog (function)



Code:
```
var hoursOccupied=global.get('hoursOccupied')||0;
var timerId = global.get('timerId')||0;
var status=global.get('status')||"empty";
global.set('counter',0);

clearTimeout(timerId);
global.set('hoursOccupied',0);
global.set('msgSent',false);
return msg;
```

(17) TimeExpired (function)

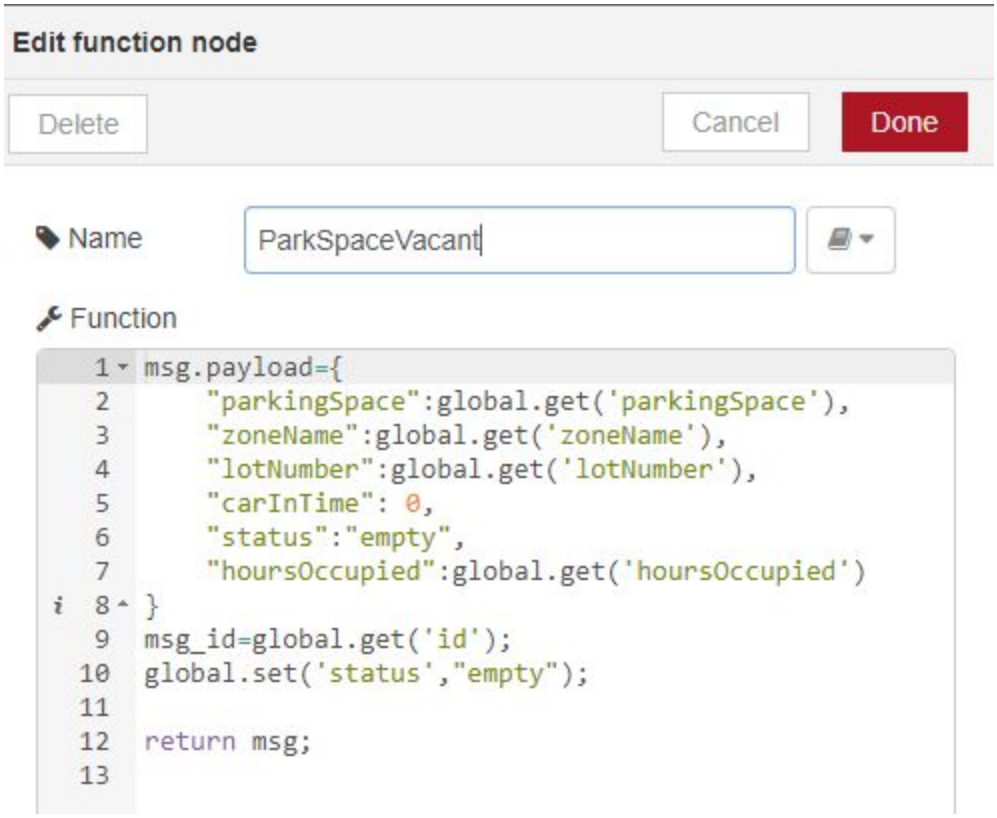


Code:
```
var hoursOccupied=global.get('hoursOccupied');
msg.payload={
   "parkingSpace":global.get('parkingSpace'),
   "zoneName":global.get('zoneName'),
   "lotNumber":global.get('lotNumber'),
   "carInTime":global.get('carInTime'),
   "Status":"full" }
msg_id=global.get('id');
global.set('status',"full");
function counterSet() {
      console.log("Hey");
      var a = global.get('counter') + 1;
      if(a<4){
```

```
            global.set('counter', a);
            console.log(a);     }
          clearTimeout(myVar); }

        if(hoursOccupied >= 4 && global.get('msgSent') === false)
        {
           global.set('counter', 0);
           msg.payload.hoursOccupied=4;
            return [msg, msg];
        }
        else if(hoursOccupied < 4)
        {
           console.log("I am here now");
           myVar = setTimeout(counterSet, 15000);
           msg.payload.hoursOccupied=global.get('counter');
           return [msg, null];
        }
```

(18) ParkSpaceVacant (function)

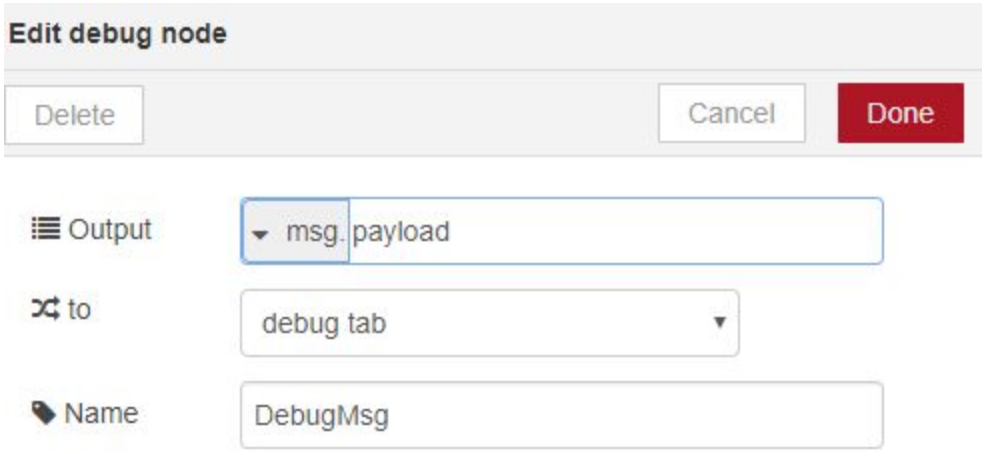


Code: msg.payload={
"parkingSpace":global.get('parkingSpace'),

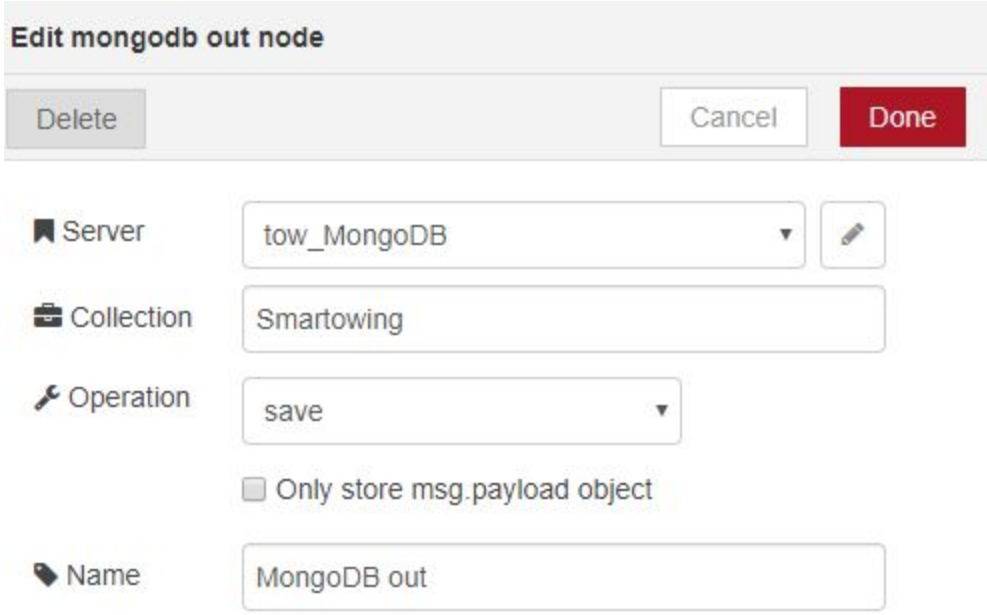
```
    "zoneName":global.get('zoneName'),
    "lotNumber":global.get('lotNumber'),
    "carInTime": 0,
    "status":"empty",
    "hoursOccupied":global.get('hoursOccupied')
  }
  msg_id=global.get('id');
  global.set('status',"empty");

  return msg;
```

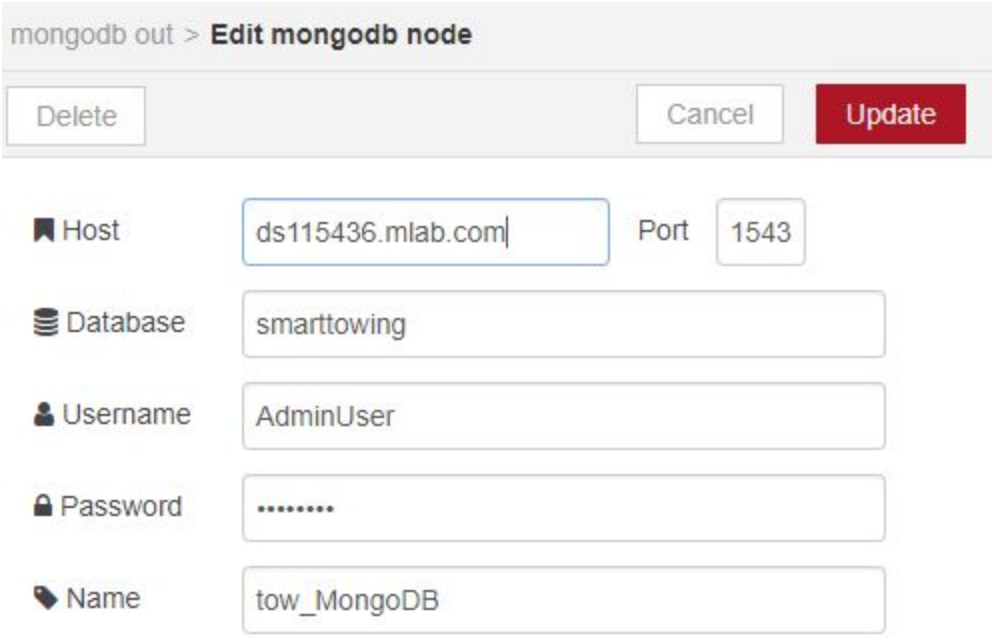
(19) DebugMsg (debug)



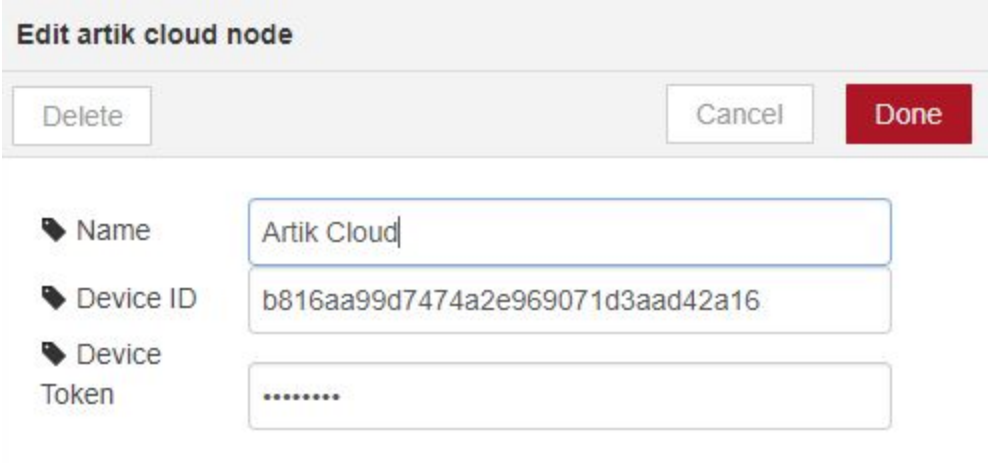
(20) MongoDB out (mongodb out)



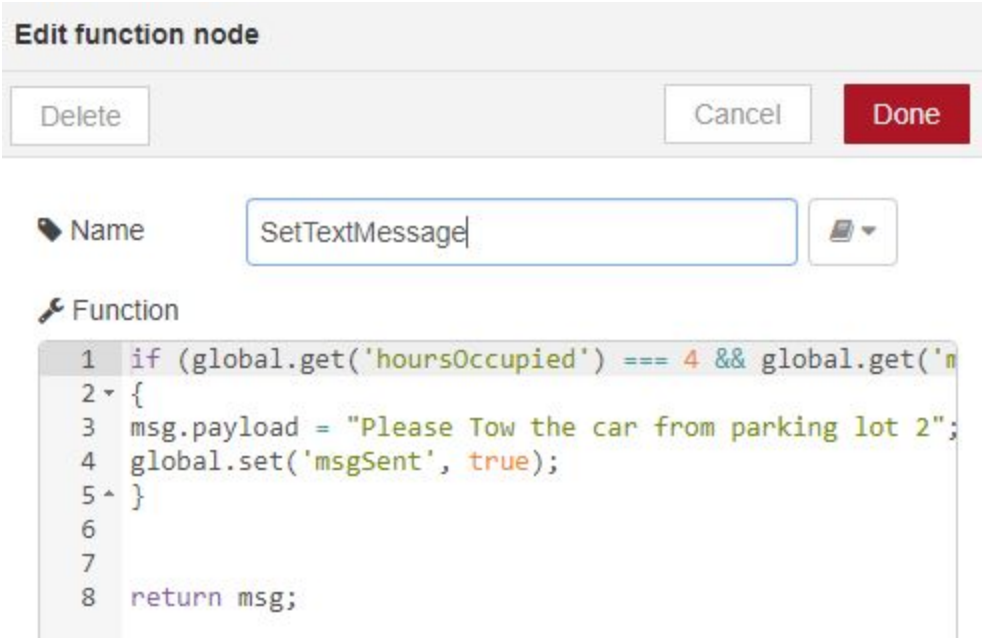
In the edit button right to server text field ->



(21) Artik Cloud (artik cloud)

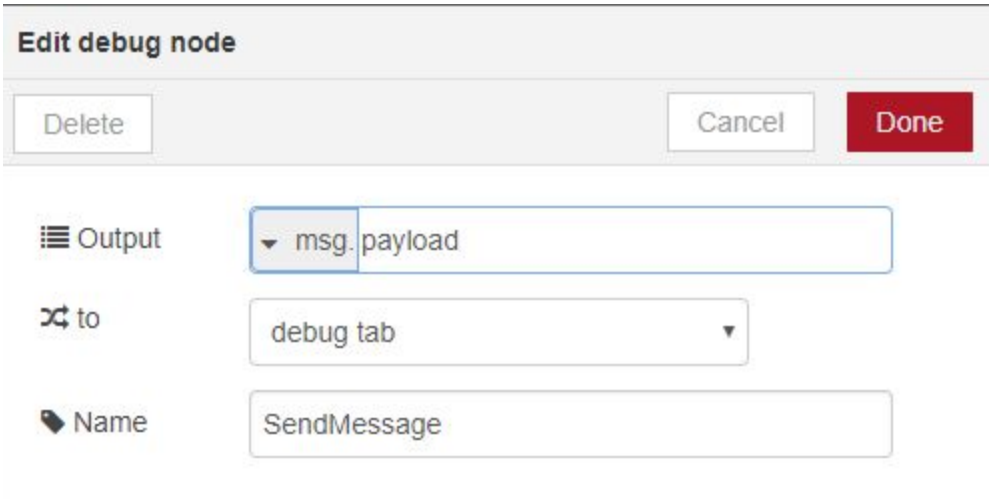


(22) SetTextMessage (function)

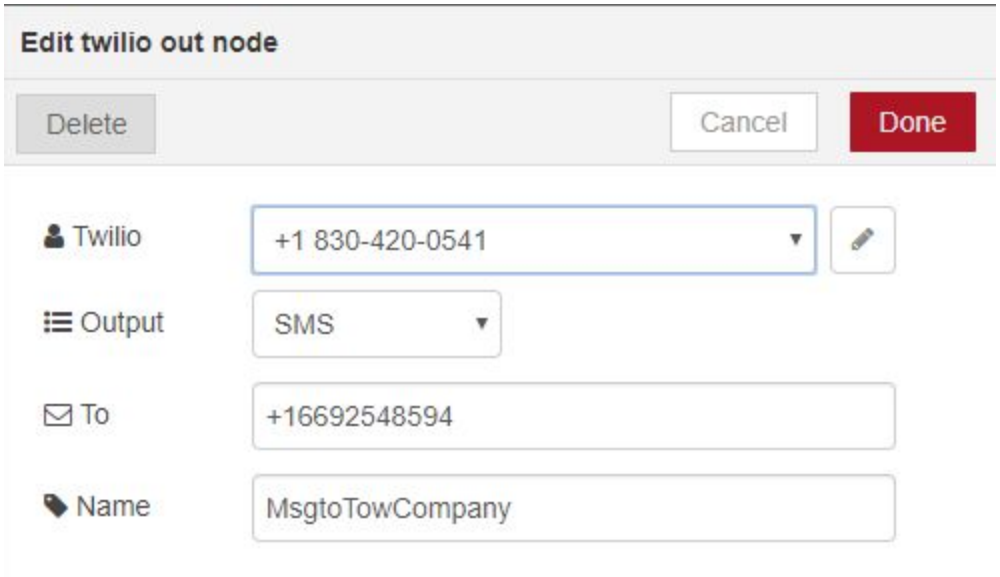


Code: if (global.get('hoursOccupied') === 4 && global.get('msgSent') === false)
{ msg.payload = "Please Tow the car from parking lot 2";
global.set('msgSent', true); }
return msg;

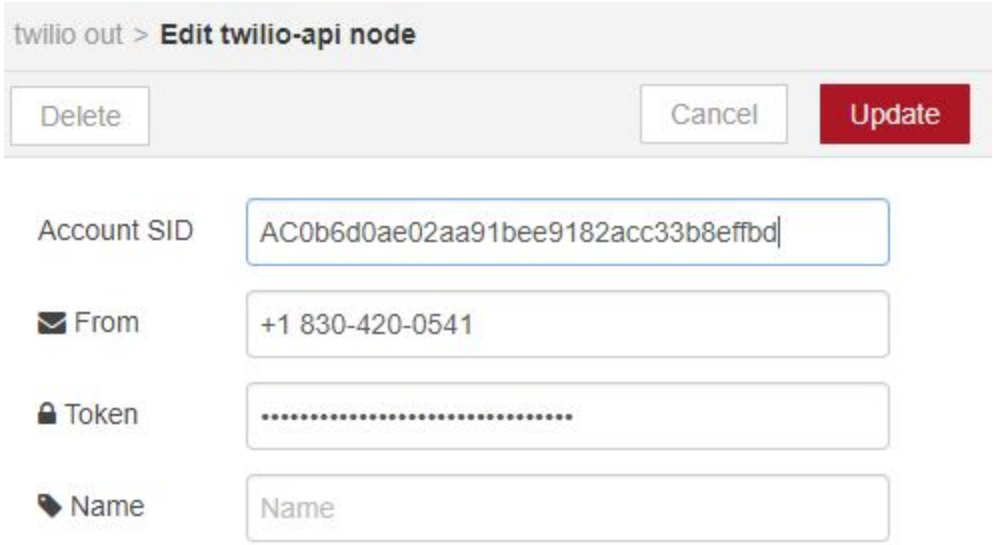
(23) SendMessage (debug)



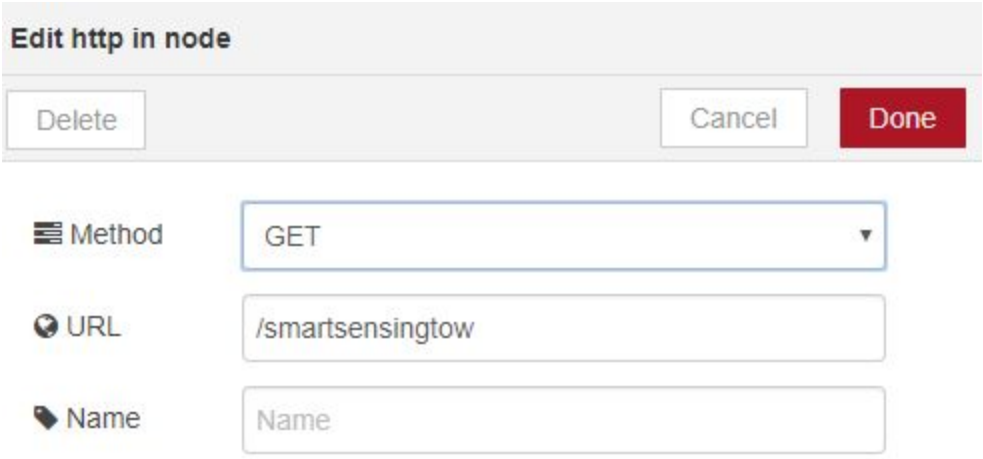
(24) MsgtoTowCompany (Twilio)



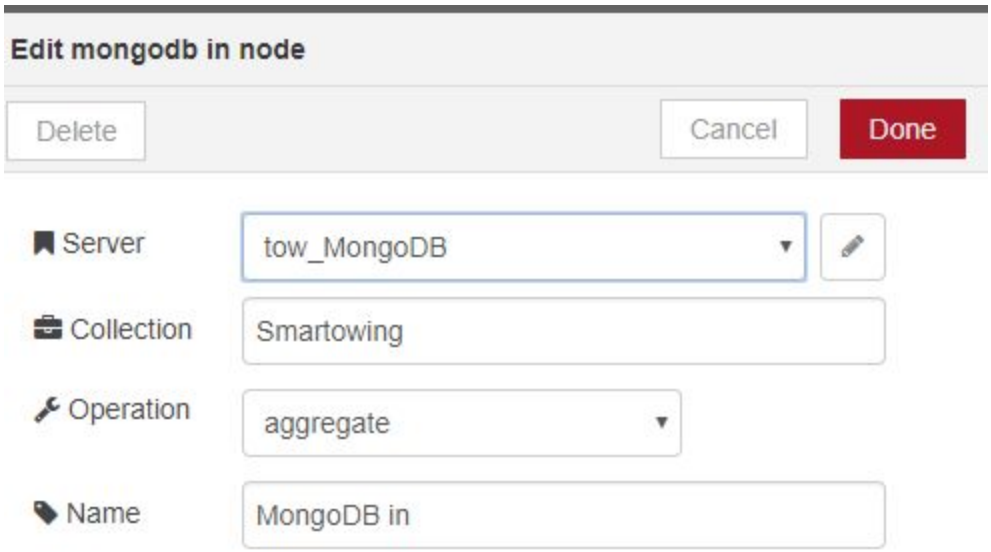
In the edit button next to twilio text field ->



(25) Smartsensingtow (http out)



(26) Mongo DB in (mongodb in)



In the edit button next to server text field ->

mongodb in > **Edit mongodb node**

Delete | Cancel | Update

Host: ds115436.mlab.com Port: 1543

Database: smarttowing

Username: AdminUser

Password: ••••••••

Name: tow_MongoDB

(27) SetTableFormat (template)

Edit template node

Delete | Cancel | Done

Name: SetTableFormat

Set property: msg. payload

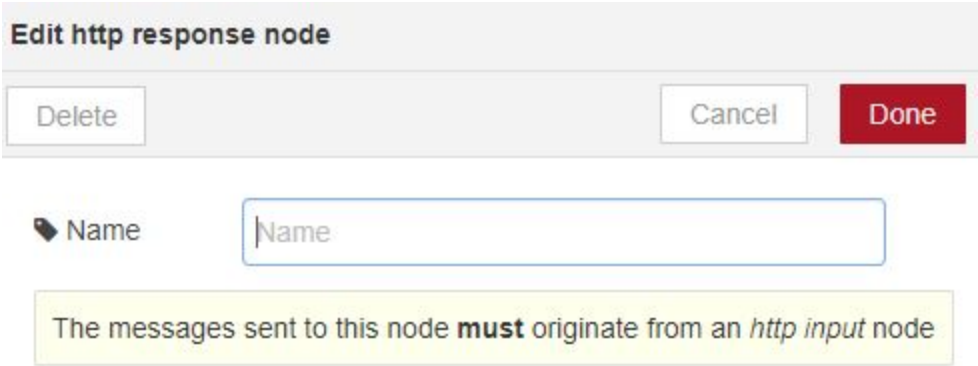
Template Syntax Highlight: mustache

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  </head>
5  <body>
6  <h3>ARTIK-Powered Smart Parking System</h3>
7  <div>
8  <div class="parking">
```

Code: <!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>
```

(28) Http (http in)



Appendix A

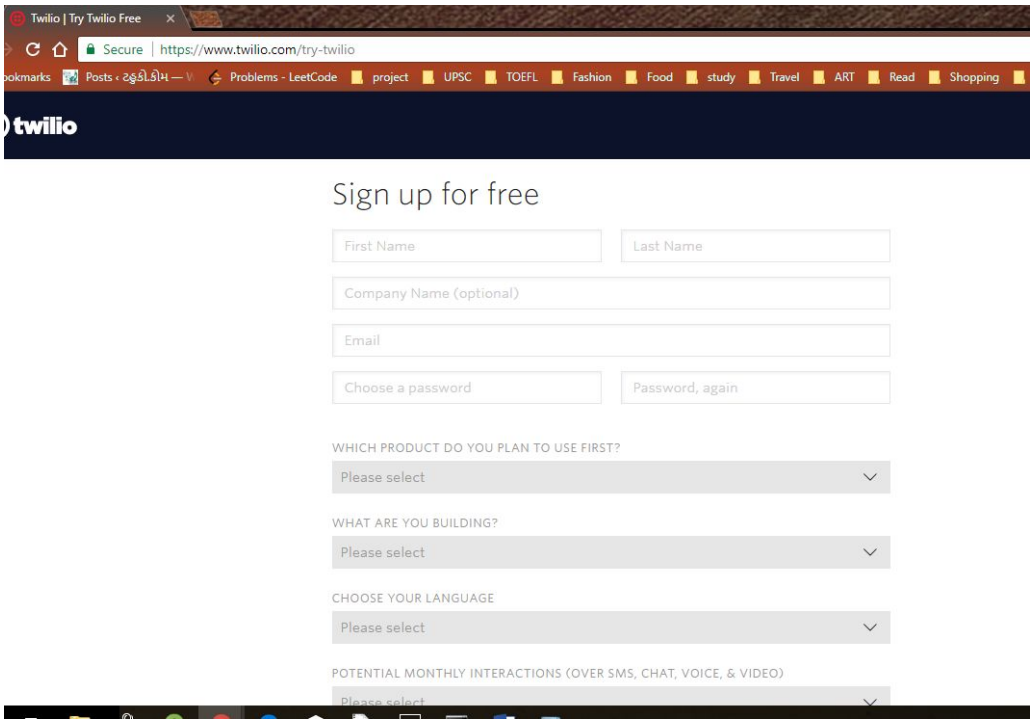
Sending Text Message in Node-RED through Twilio

This document explains the steps of sending text message in Node-red using Twilio.

(A) Setup Twilio Account

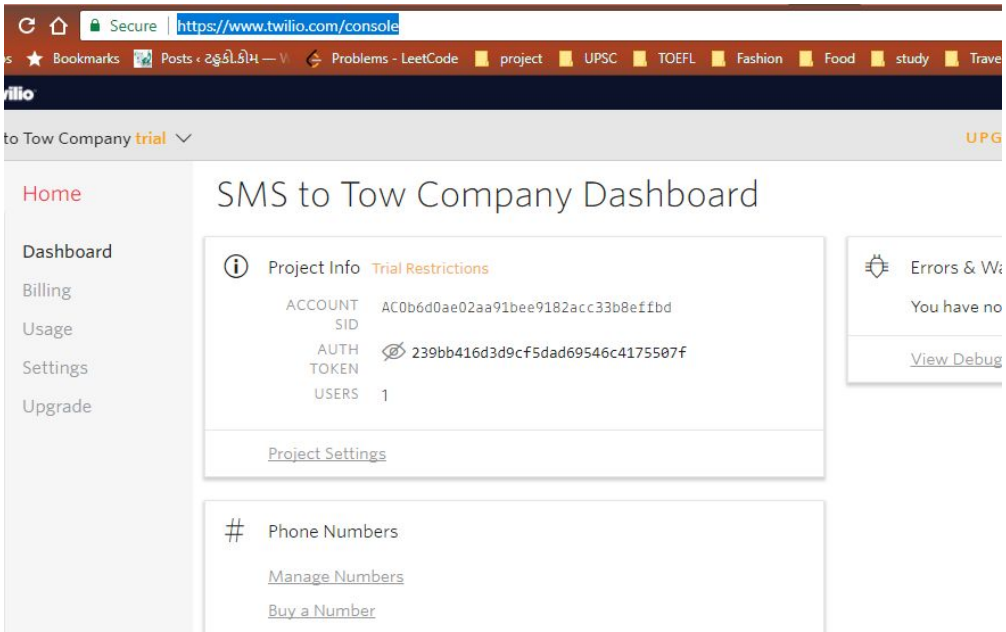
The first step is to create a twilio account and setup a number to send text message.

(1) Create a Twilio Account - Go to www.twilio.com/try-twilio



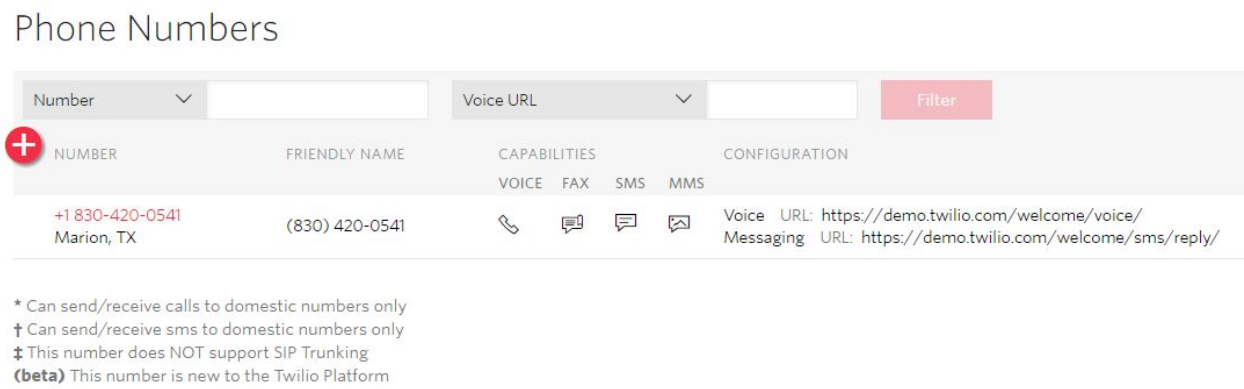
Create new project and name it. Project name: SMS to Tow company.

(2) Go to Console tab in twilio - https://www.twilio.com/console



In this console tab, your Account SID and Auth Token will be generated which will be used in Node-red for configuration.

(3) Got to Manage numbers from console page to create new number. Select a number from the list which has SMS. You can buy the number for 2$ for a month but as I used a number which comes with trial period of twilio. Now this number will be used to send text messages.



After setting up account in twilio, next step is installing Node-red in your local computer.

(B) Installing Node-red

Open command prompt/text editor of your computer and type:

`sudo npm install -g --unsafe-perm node-red` (for MAC/Linux)

`npm install -g --unsafe-perm node-red` (for windows)

```
node-red
C:\Users\Dipal>
C:\Users\Dipal>npm install -g --unsafe-perm node-red
npm WARN deprecated i18next-client@1.10.3: you can use npm install i18next from
version 2.0.0
npm WARN deprecated nodemailer@1.11.0: All versions below 4.0.1 of Nodemailer ar
e deprecated. See https://nodemailer.com/status/
npm WARN deprecated node-uuid@1.4.8: Use uuid module instead
C:\Users\Dipal\AppData\Roaming\npm\node-red -> C:\Users\Dipal\AppData\Roaming\np
m\node_modules\node-red\red.js
C:\Users\Dipal\AppData\Roaming\npm\node-red-pi -> C:\Users\Dipal\AppData\Roaming
\npm\node_modules\node-red\bin\node-red-pi

> bcrypt@1.0.3 install C:\Users\Dipal\AppData\Roaming\npm\node_modules\node-red\
node_modules\bcrypt
> node-pre-gyp install --fallback-to-build

[bcrypt] Success: "C:\Users\Dipal\AppData\Roaming\npm\node_modules\node-red\node
_modules\bcrypt\lib\binding\bcrypt_lib.node" is installed via remote
+ node-red@0.17.5
added 367 packages in 26.659s

C:\Users\Dipal>node-red
5 Nov 10:21:14 - [info]
```

Once you installed Node-RED as a global npm package, you can use the node-red command to start running Node-RED.

Type node-red in command prompt.

```
node-red
C:\Users\Dipal>node-red
5 Nov 10:21:14 - [info]

Welcome to Node-RED
===================

5 Nov 10:21:14 - [info] Node-RED version: v0.17.5
5 Nov 10:21:14 - [info] Node.js  version: v8.9.0
5 Nov 10:21:14 - [info] Windows_NT 10.0.15063 x64 LE
5 Nov 10:21:16 - [info] Loading palette nodes
5 Nov 10:21:17 - [warn] ---------------------------------------------------------
5 Nov 10:21:17 - [warn] [rpi-gpio] Info : Ignoring Raspberry Pi specific node
5 Nov 10:21:17 - [warn] [tail] Not currently supported on Windows.
5 Nov 10:21:17 - [warn] ---------------------------------------------------------
5 Nov 10:21:17 - [info] Settings file  : C:\Users\Dipal\.node-red\settings.js
5 Nov 10:21:17 - [info] User directory : C:\Users\Dipal\.node-red
5 Nov 10:21:17 - [info] Flows file     : C:\Users\Dipal\.node-red\flows_dipal-la
ptop.json
5 Nov 10:21:17 - [info] Creating new flow file
5 Nov 10:21:17 - [info] Server now running at http://127.0.0.1:1880/
5 Nov 10:21:17 - [info] Starting flows
5 Nov 10:21:17 - [info] Started flows
5 Nov 10:27:11 - [info] Stopping flows
5 Nov 10:27:11 - [info] Stopped flows
5 Nov 10:27:11 - [info] Starting flows
5 Nov 10:27:11 - [info] Started flows
5 Nov 10:28:55 - [info] Stopping flows
5 Nov 10:28:55 - [info] Stopped flows
5 Nov 10:28:55 - [info] Starting flows
```

Once Node-RED is running, point a local browser at http://localhost:1880. You can always use a browser from another machine if you know the ip address or name of the Node-RED instance - http://{Node-RED-machine-ip-address}:1880

[For more information about installation : https://nodered.org/docs/getting-started/installation]

Now, next step is to add twilio node in Node-RED for sending text messages.

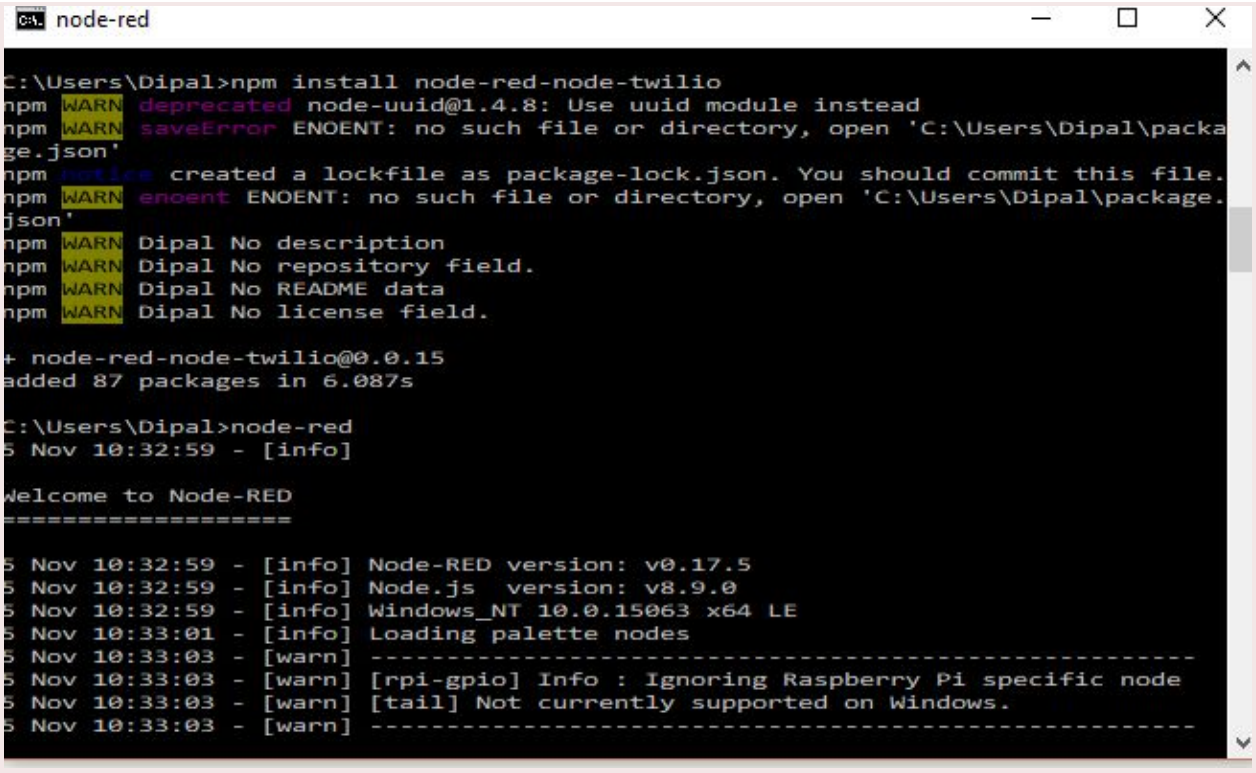
(C) Adding Twilio node in Node-RED

Run the following command in your Node-RED user directory.

npm install node-red-node-twilio

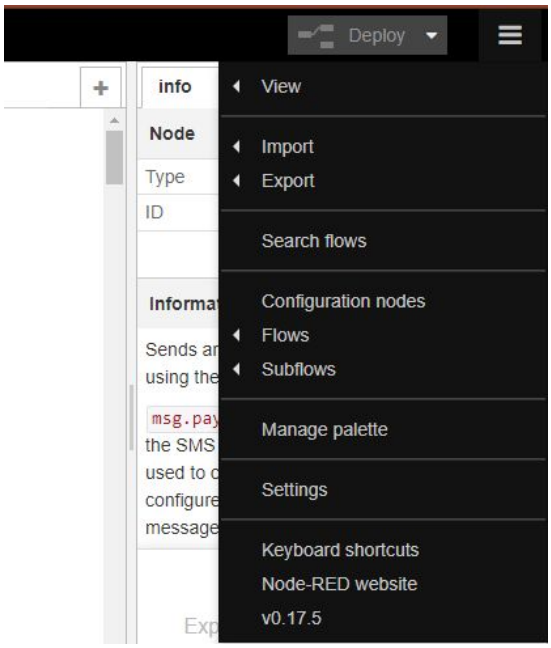
The Twilio out node is configured to send SMS or make call, depending on the option selected you enter the phone number. msg.payload is used as the body of the message. The node can be configured with the number to send the message to.

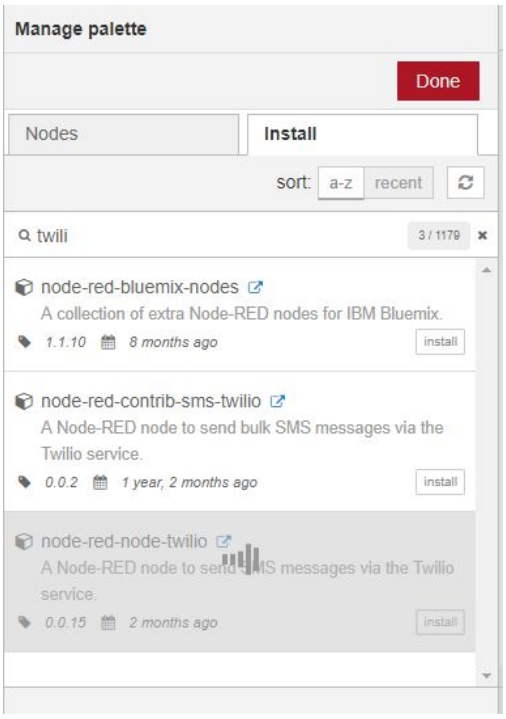
[More information about installation: https://flows.nodered.org/node/node-red-node-twilio]



Or you can install twilio node from Node-red console. Go to http://localhost:1880 and Click on the option menu on top right and select manage palette option. Then go to install tab

and search for twilio and install.





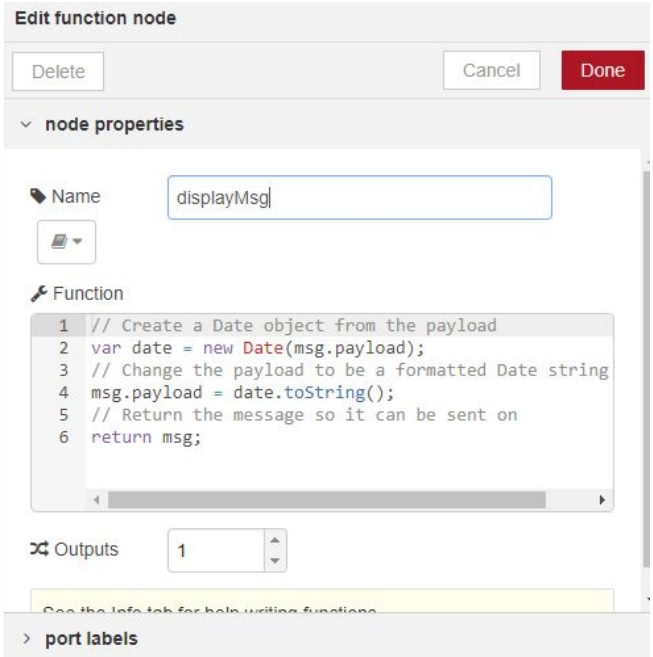
(D) Adding Twilio node and executing

Go to http://localhost:1880. Add following nodes:

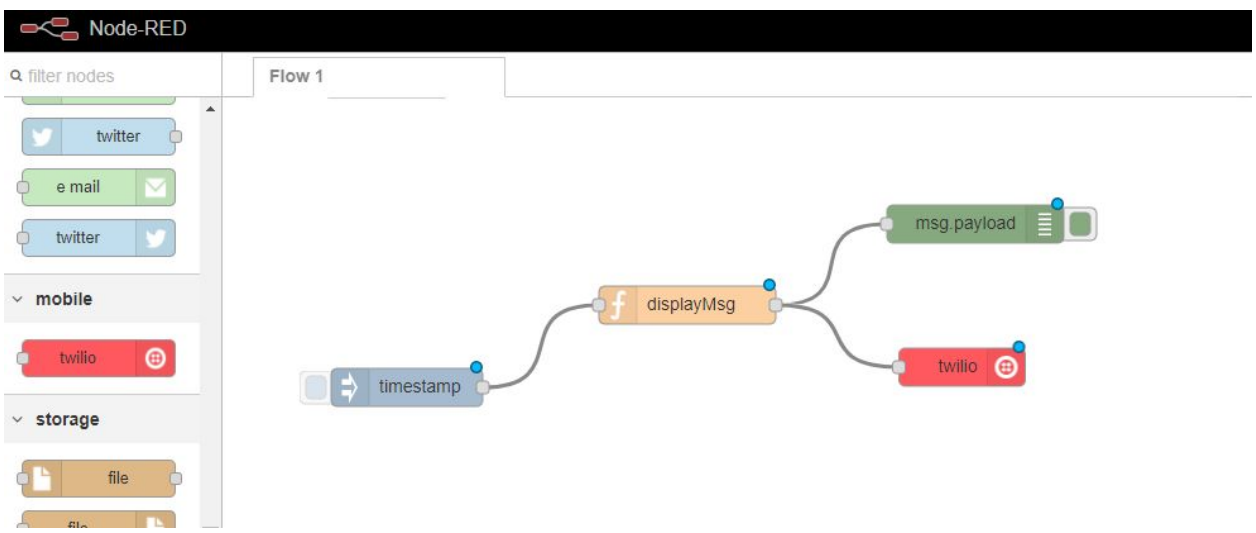
(1) **Inject Node:** The Inject node allows you to inject messages into a flow, either by clicking the button on the node, or setting a time interval between injects. Drag one onto the workspace from the palette.

(2) **Debug node:** The Debug node causes any message to be displayed in the Debug sidebar.

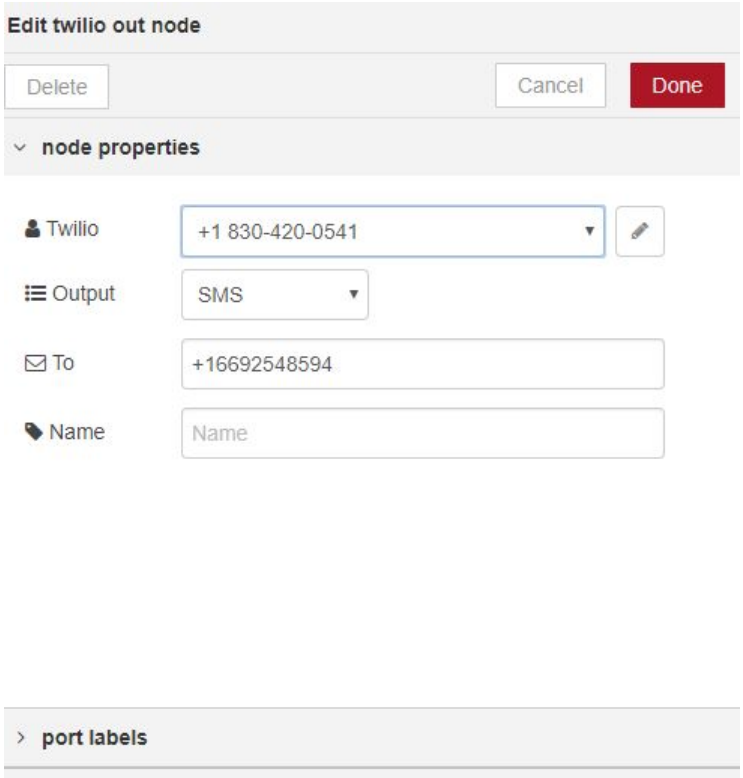
(3) **Function node**: The Function node allows you to pass each message though a JavaScript function. Wire the Function node in between the Inject and Debug nodes. Edit function node,



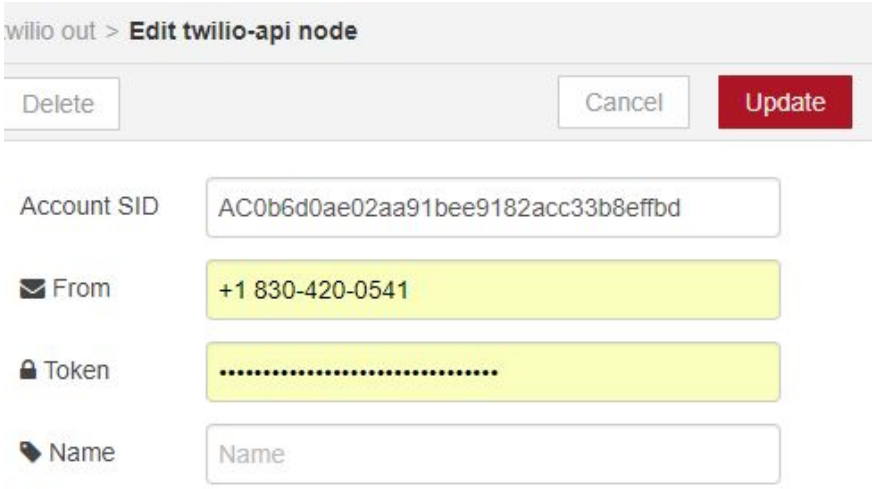
(4) **Twilio node:** The twilio node allows you to send message in function node to a given number. Wire the Function node and twilio node.

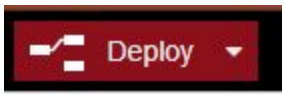


Edit twilio out node,

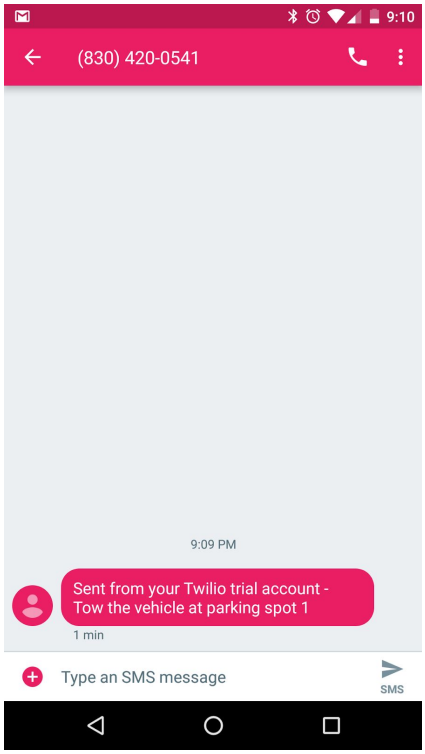


Enter the twilio number in Twilio field and click on  icon and edit account SID and Token from your twilio account shown in Step A.



Everything is set now, click on 

and inject the timestamp inject node. And text message will be sent to your given number.

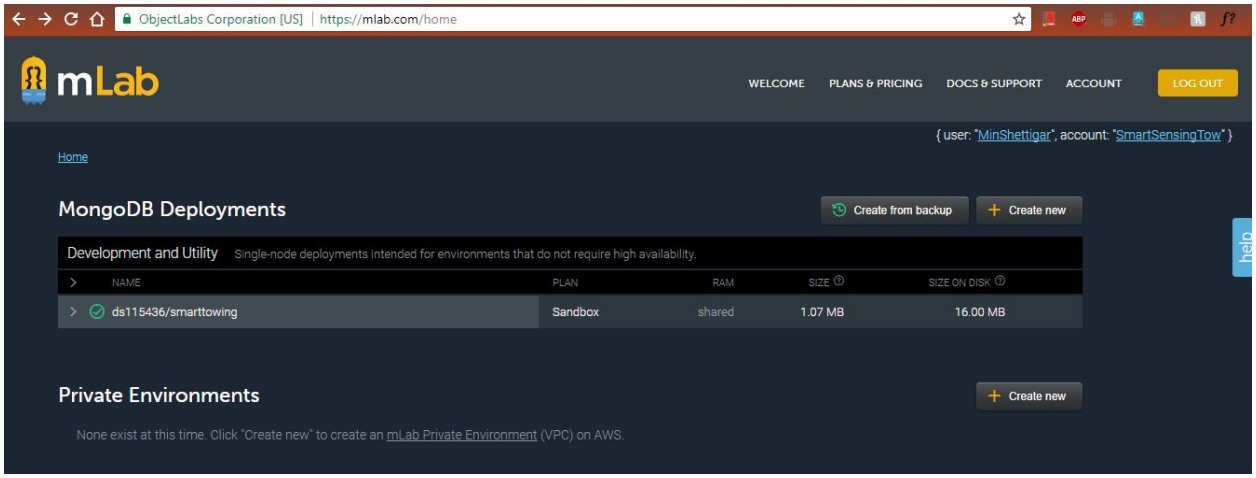

9:10
(830) 420-0541
9:09 PM
Sent from your Twilio trial account -
Tow the vehicle at parking spot 1
1 min
Type an SMS message
SMS

Appendix B - MongoDB connection

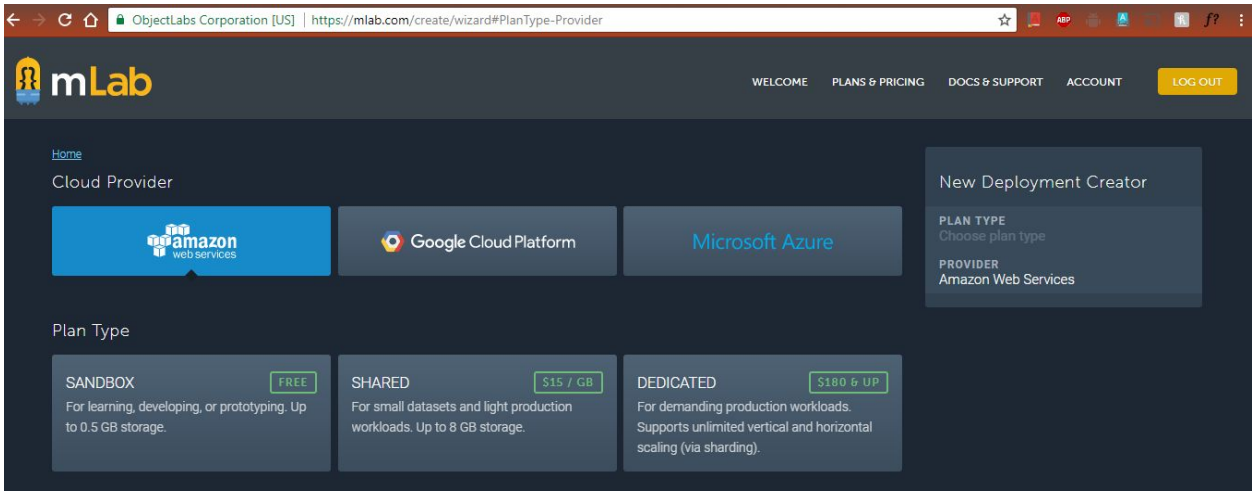
Run the following command in your Node-RED user directory to install MongoDB.

- npm install node-red-node-mongodb

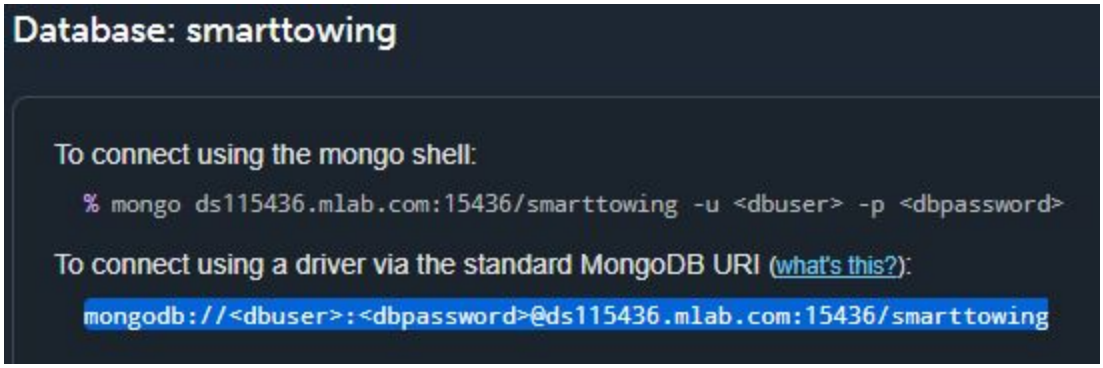
Then the first step is - create an account in https://mlab.com/home. Once you are logged in click on 'Create New' command.



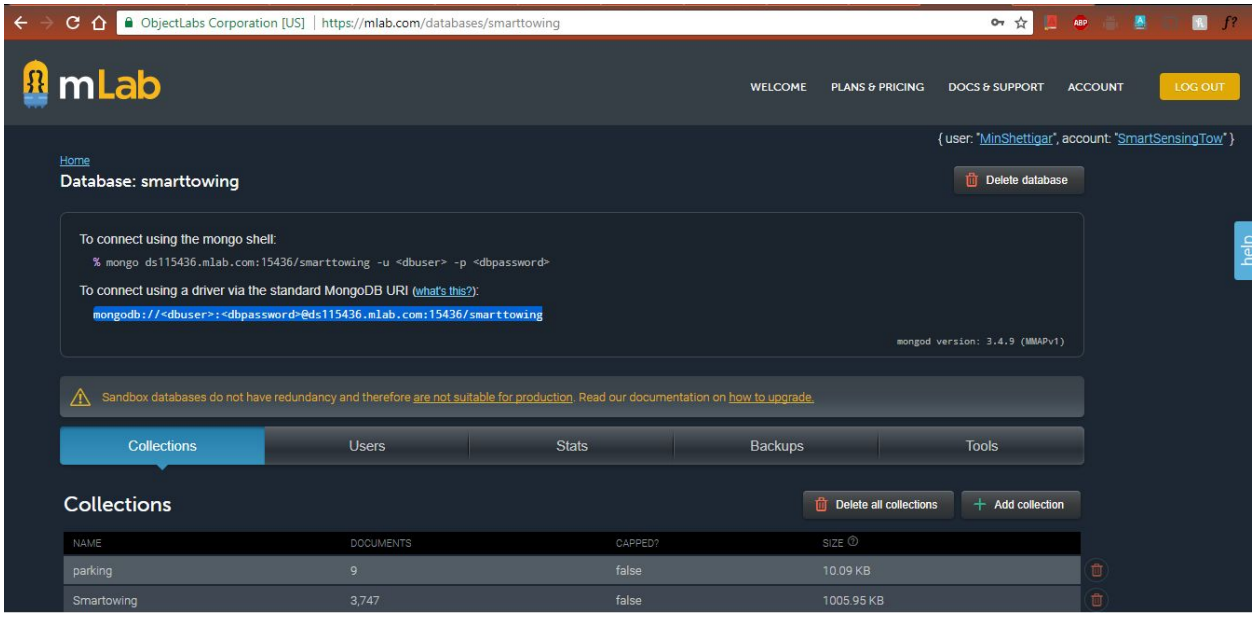
From this page select basic sandbox plan type which provides 16MB of free data storage.



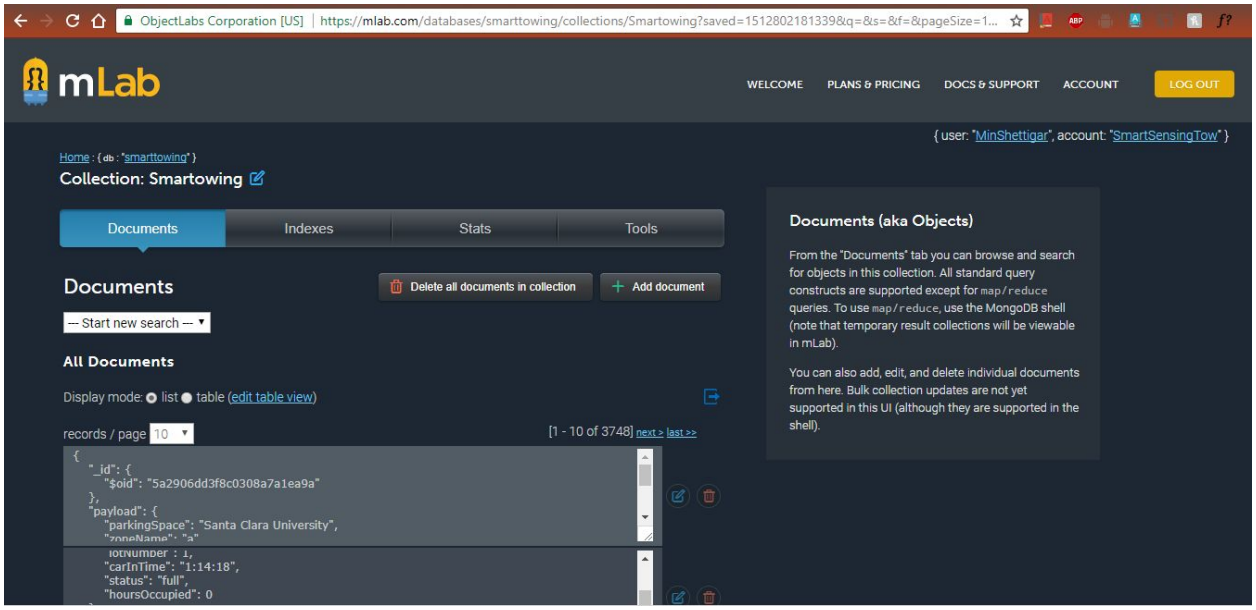
You will get mongoDB connection string



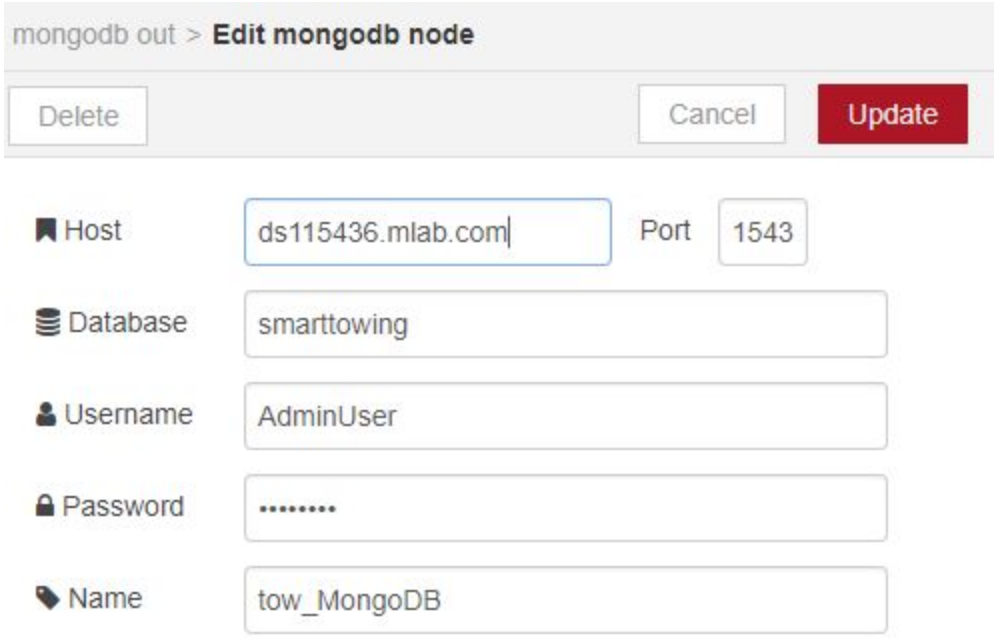
Create a database named 'smarttowing' and add new collection into database by clicking 'Add Collection'.



Now, create new document by clicking ‘Add document’. And write rules.

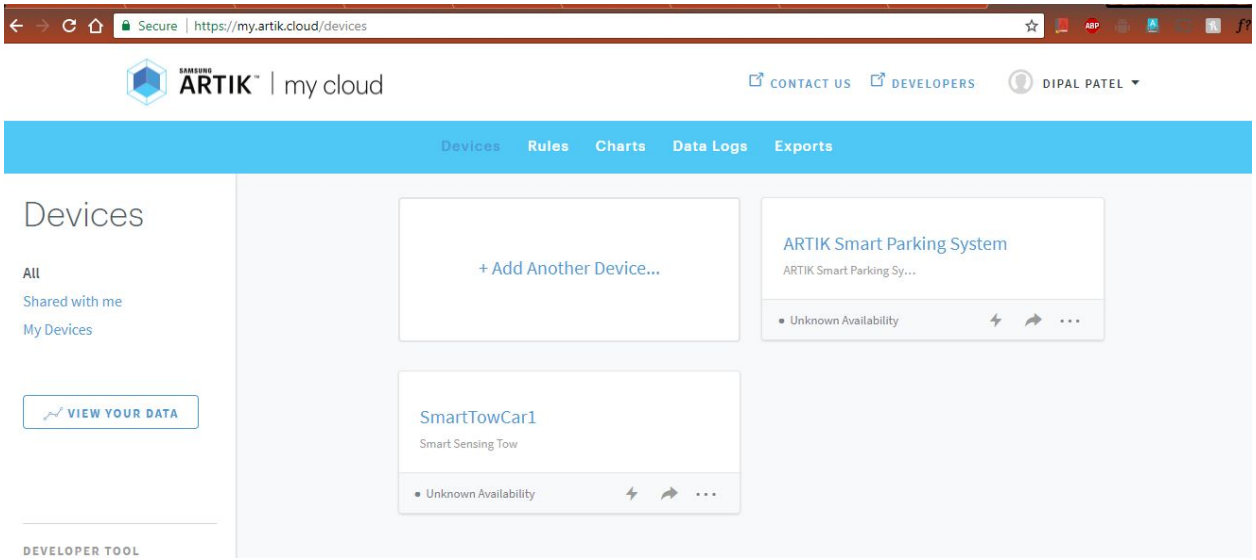


Now open MongoDB node in Node-red and edit it
host: ds115436.mlab.com
port:15436
database: smarttowing
userName: AdminUser
password: smarttow123 and
name: MongoLab_SmartSensingTow

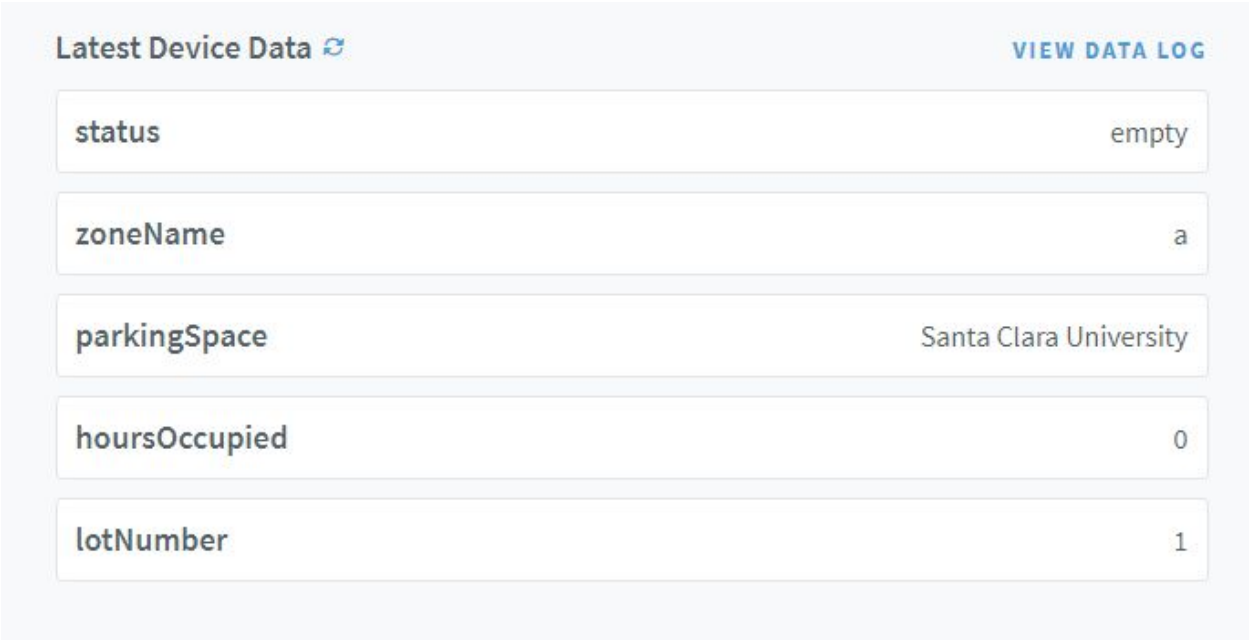


Appendix C - ARTIK cloud

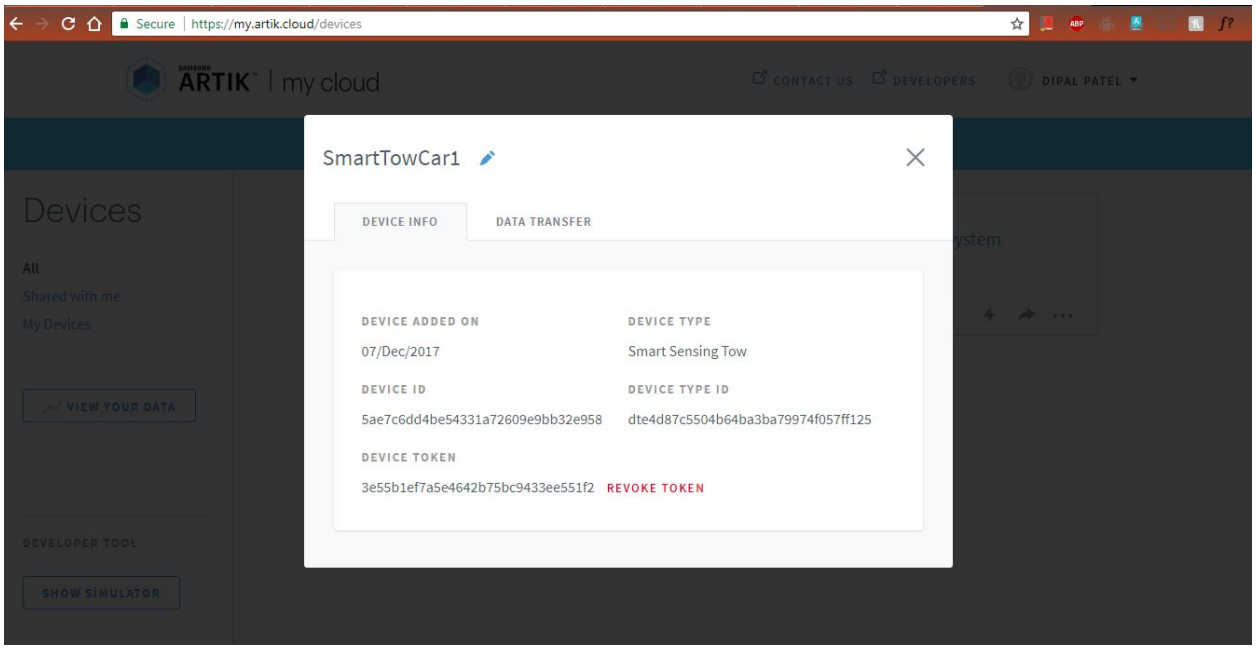
Once you create an account in ARTiK cloud. You will be redirected to this page for creating new devices.



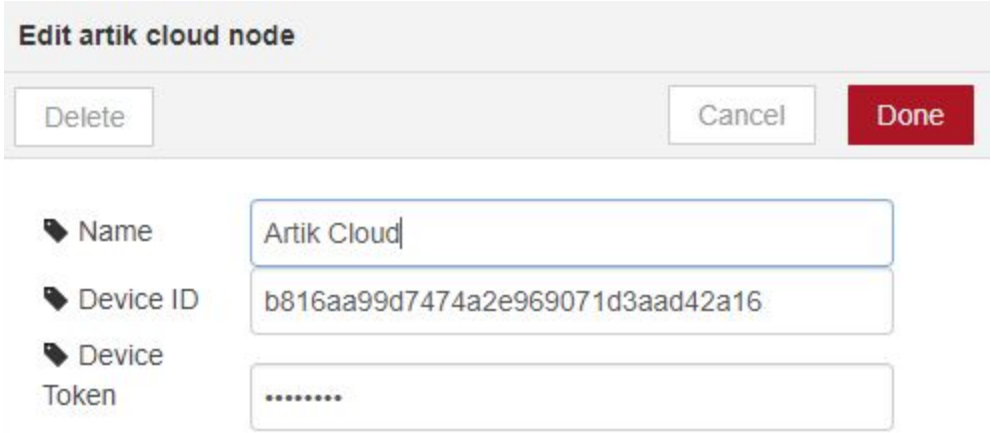
Search for “Smart Sensing tow” and select it. “Smart Sensing tow” is a public device type we created for our project, which includes parkingSpace: "Santa Clara University", zoneName: "a", lotNumber: "2", carInTime: 0 (time), status (“empty” or “full”), hoursOccupied (integer value) properties.



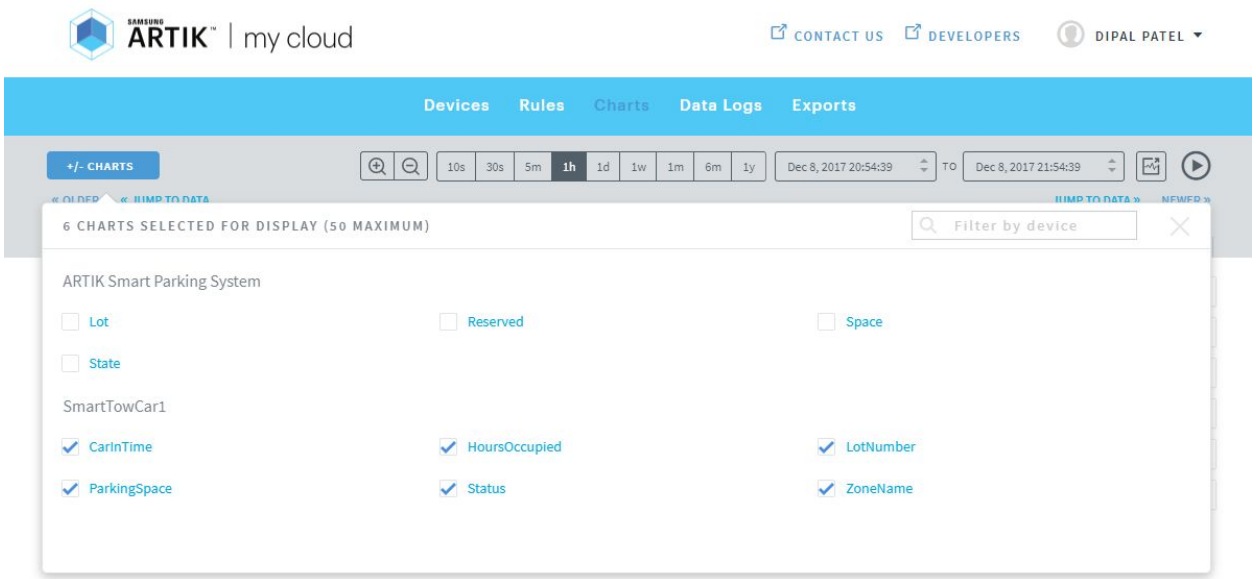
Double click on the device you created and you will be able to see the window as shown below:



You need to take Device ID and Device Token from here and paste it into ARTIK cloud node in node-red.



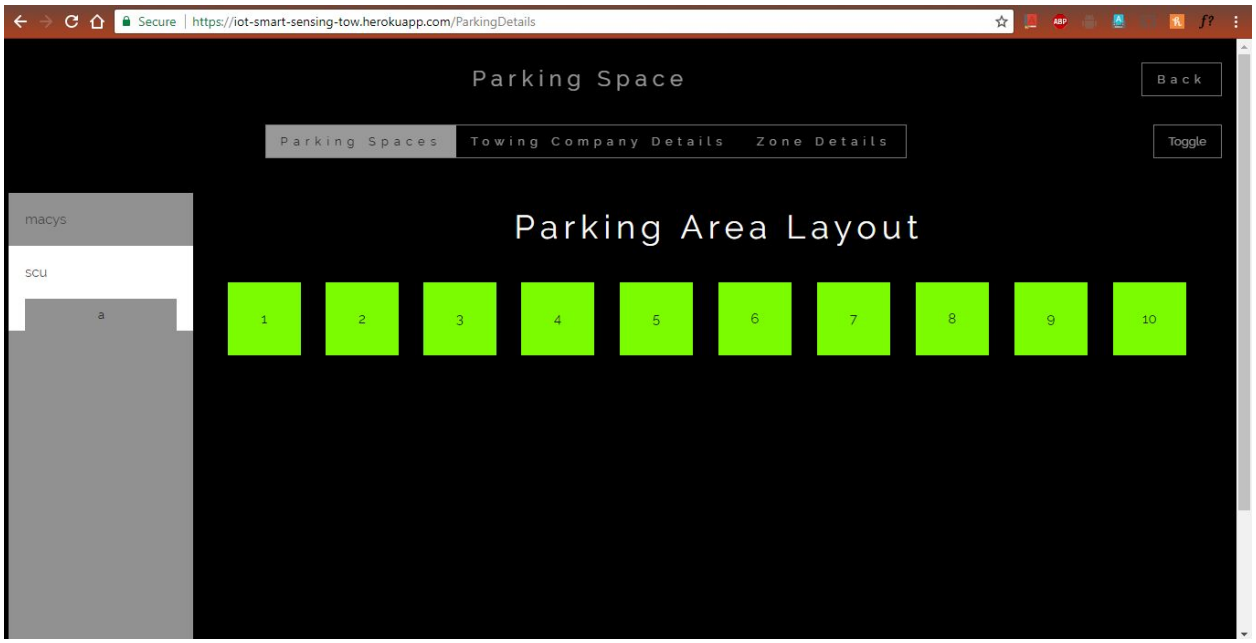
Go to ARTIK Cloud user portal, click the “+/- CHARTS” button, and enable to view the “State” of your parking space.



If you want to add Rules then, In ARTIK user portal, go to “MY ARTIK CLOUD/RULES”, and click the “+NEW RULE” button.

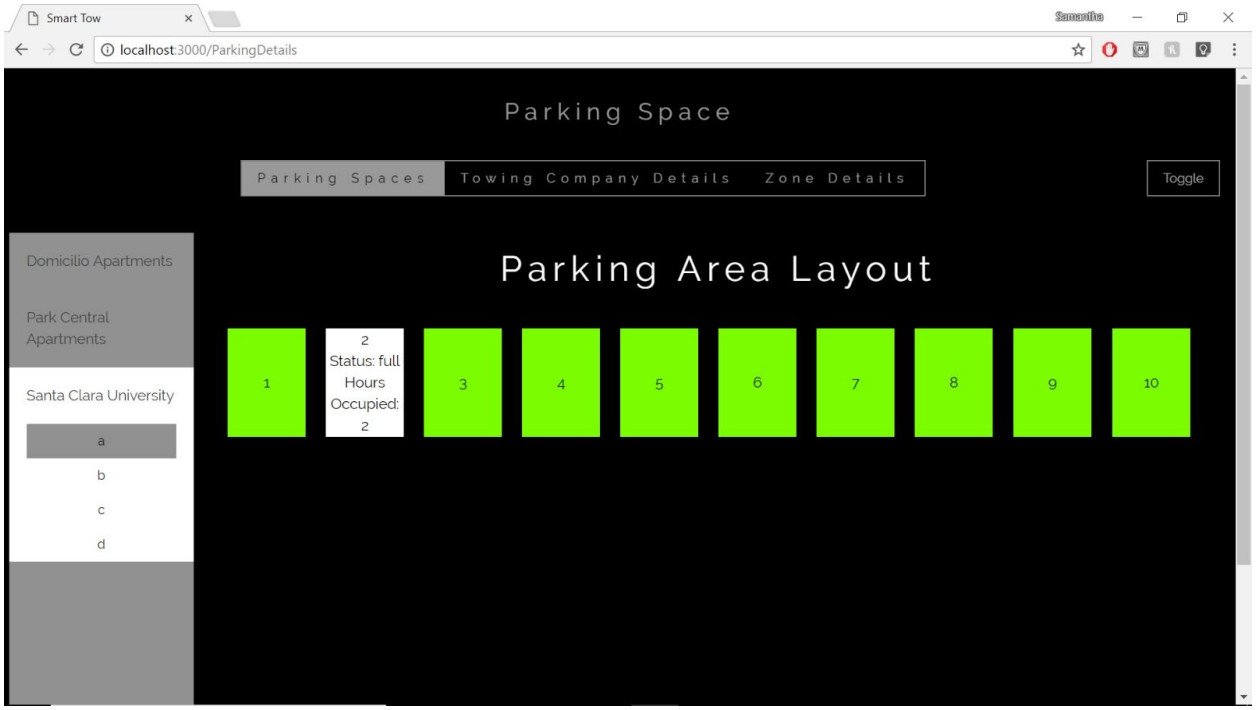
Appendix D - Web Interface

URL : https://iot-smart-sensing-tow.herokuapp.com/

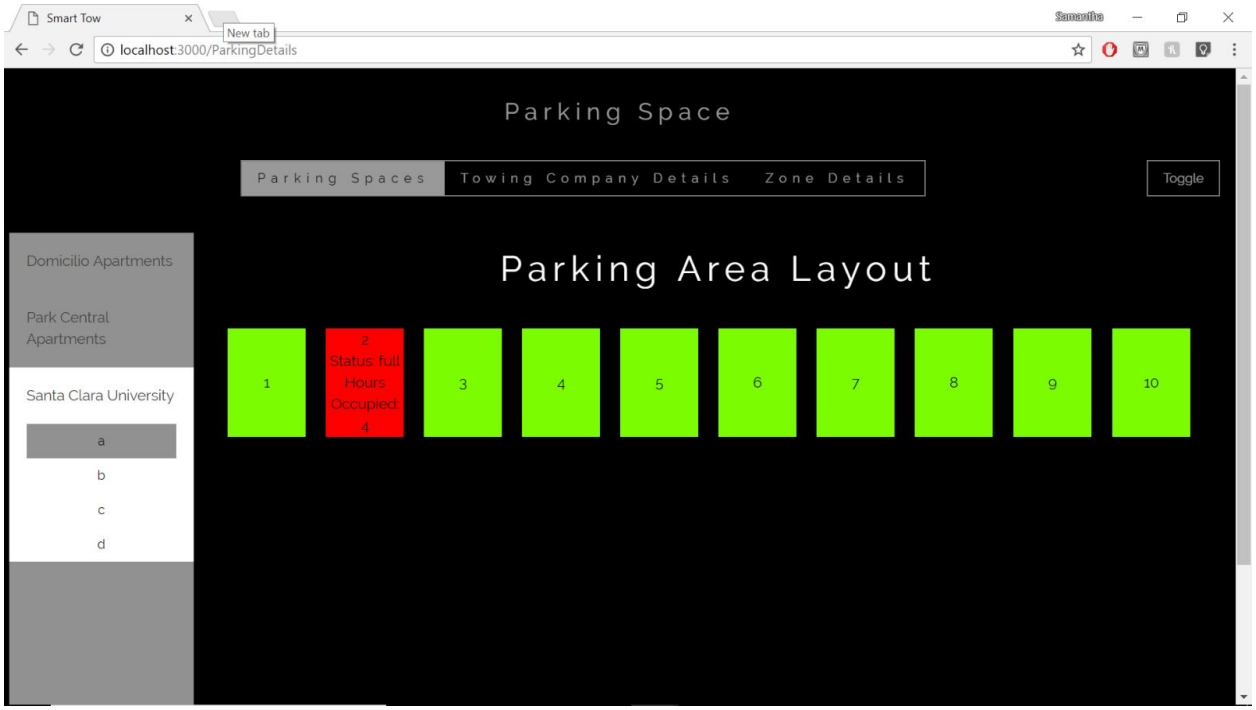


Technology used: Sensor data is collected from the cloud and after every change its updated on screen via api call.
Used html and node js
Database - MongoDB

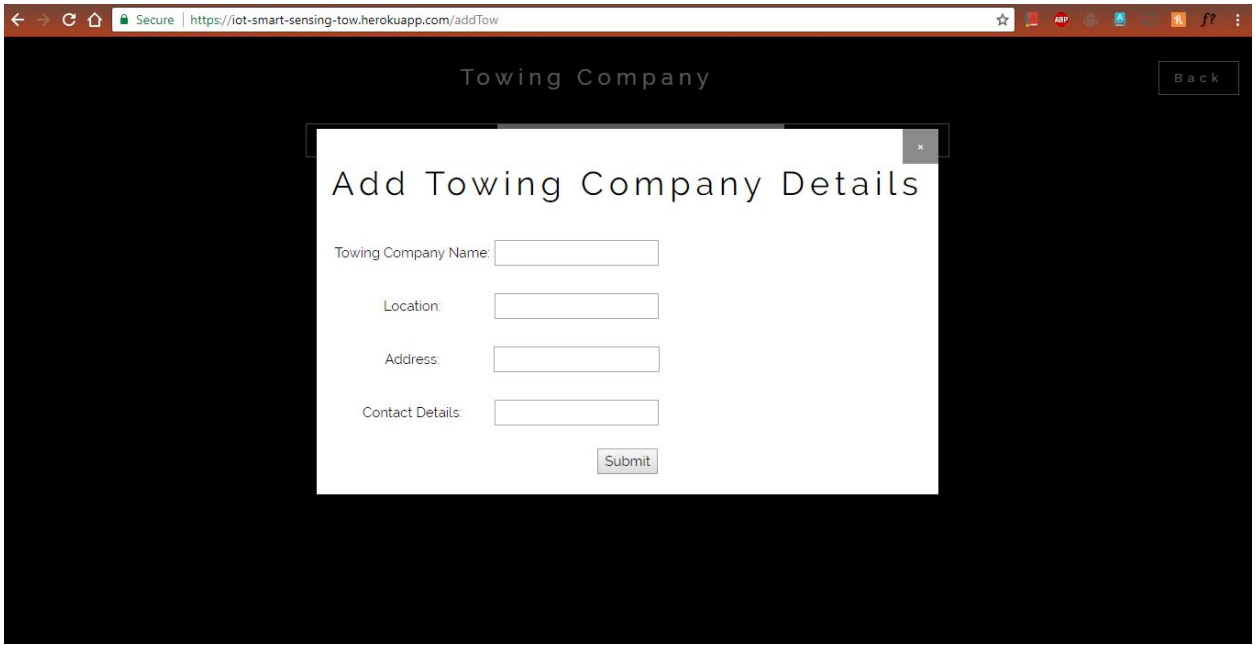
WorkFlow:
Initial available parking spot will appear in green color.
Once the parking spot is occupied by a car it will appear in white color .



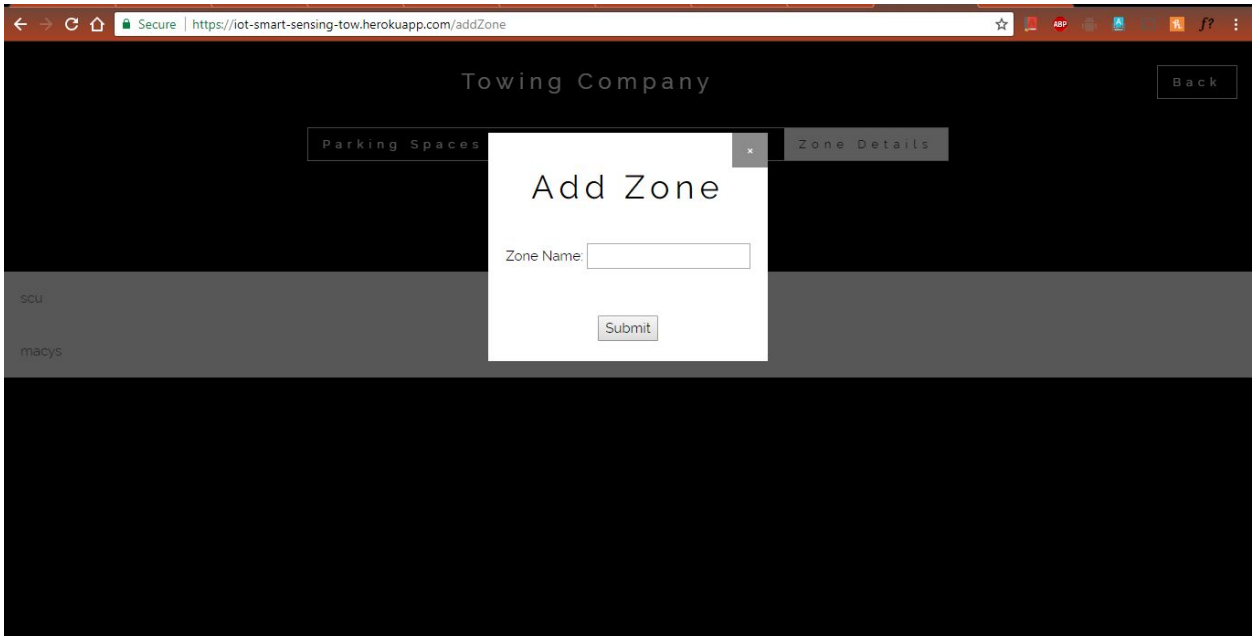
and if the time limit is exceeded then it will appear in red color.



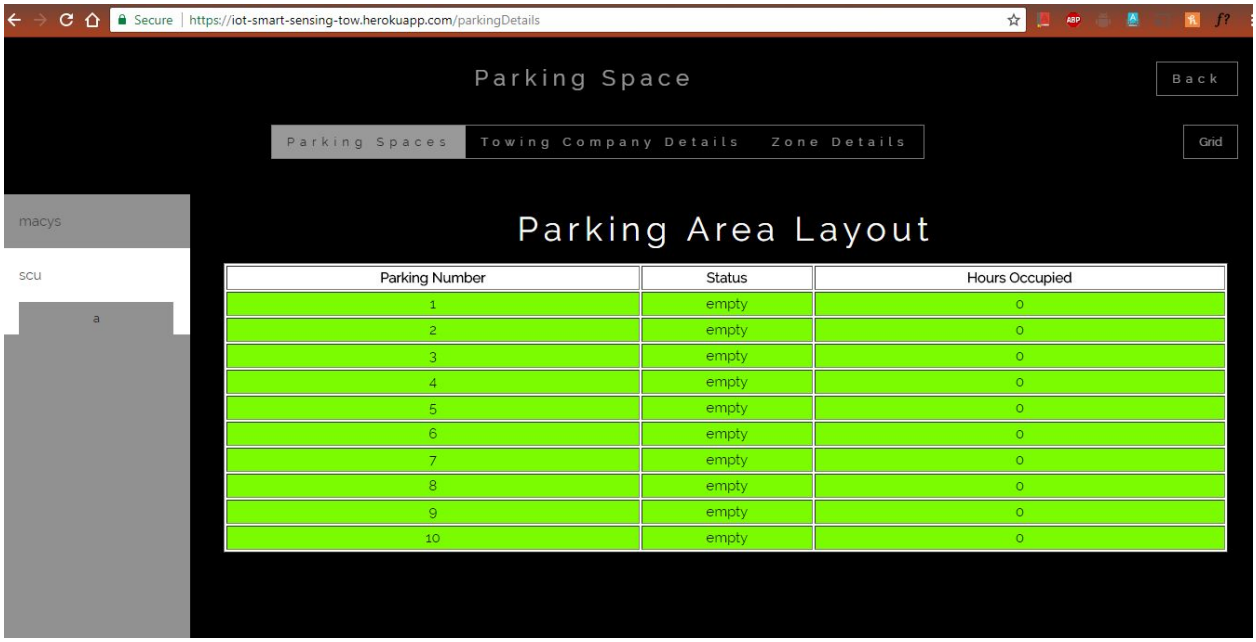
That way towing company will be able to see the status through web interface.



New tab - for adding details of new towing company into database



New tab - for adding details of new zones into database



Toggle/Grid button on right side will change the tableview and gridview.

Appendix E - Demo

Video: https://www.youtube.com/watch?v=6xzf1go26Vw&feature=youtu.be