

# SMART SENSING TOW



Dipal Patel  
Minal Shettigar  
Samantha Rodrigues  
Esha Jhamb

# **Index**

**Chapter 1 - Inspiration behind the project**

**Chapter 2 - Introduction**

**Chapter 3 - Setup and Execution**

**Appendix A - Sending Text Message in Node-RED through Twilio**

**Appendix B - MongoDB Connection**

**Appendix C - ARTIK Cloud**

**Appendix D - Web Interface**

**Appendix E - Demo**

## **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]

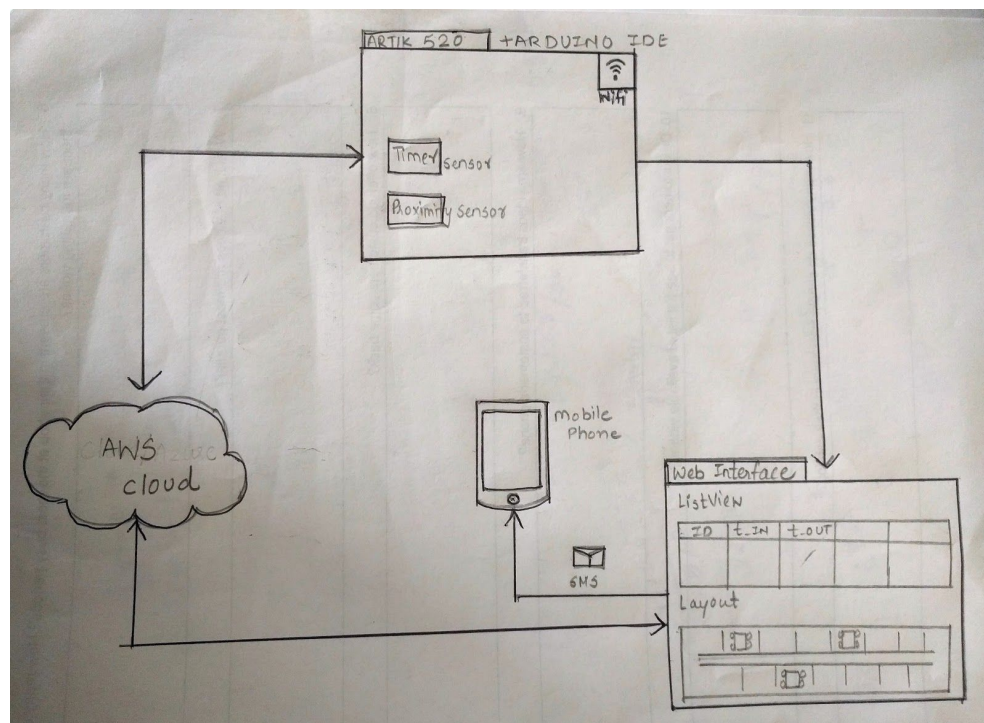
<http://nypost.com/2017/09/18/dead-man-sat-in-truck-at-airport-parking-lot-for-eight-months-and-no-one-noticed/>

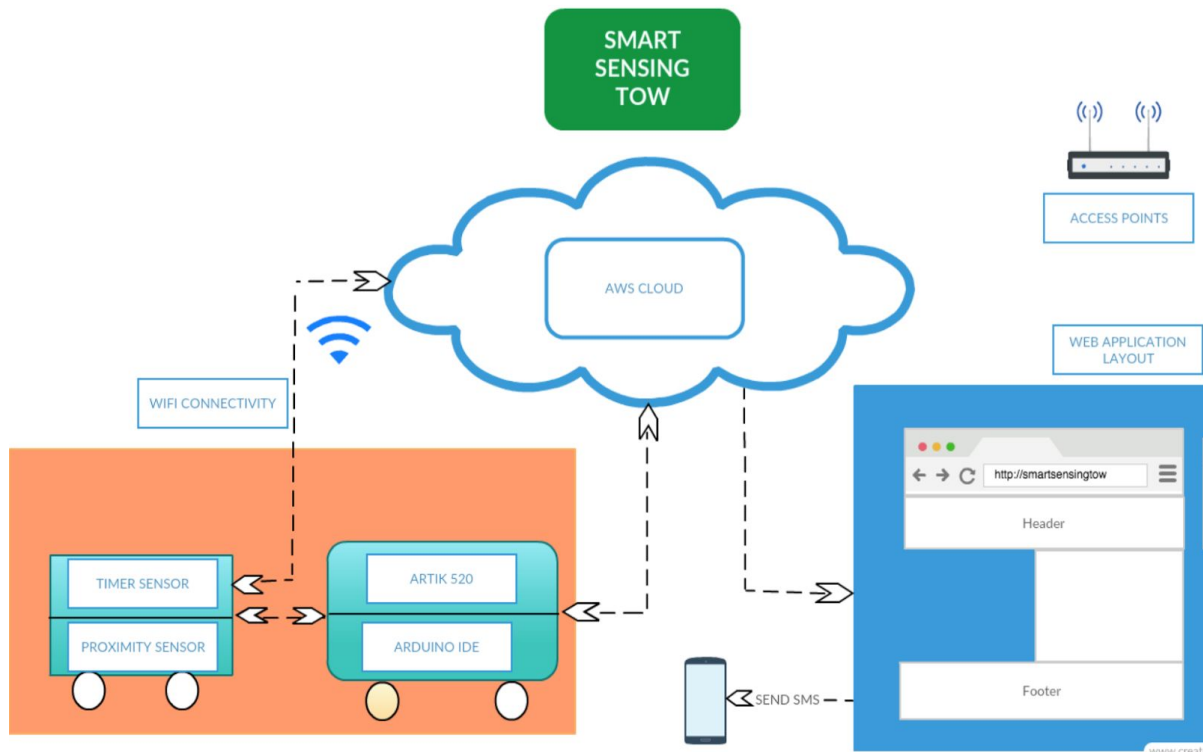
[2]

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

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.

### 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

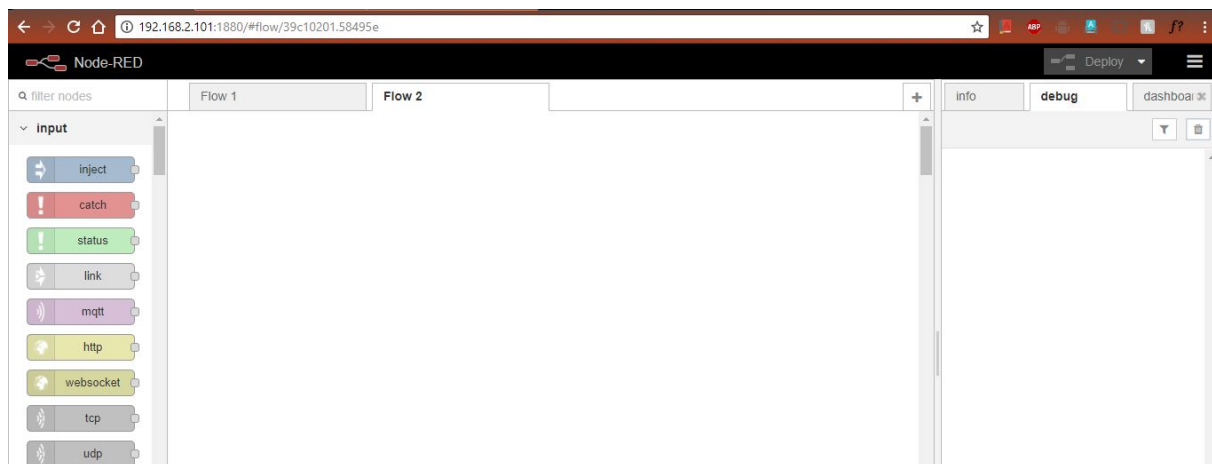
1. Connect board with USB to the computer. Open device manager and check the port of the connected board (ex:COM8). Open putty- Select serial, add port number (COM8) and baud rate =115200. Type username = root and password= root and you will be logged in local root console.
2. You have to make sure that 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



```
localhost login: root
Password:
Last login: Thu Dec  7 19:25:53 on ttySAC2
[root@localhost ~]#
[root@localhost ~]# wpa_passphrase IOT_CPS_COURSE scustudentiot243
network={
    ssid="IOT_CPS_COURSE"
    #psk="scustudentiot243"
    psk=ca3a99de045522338c2b8f8b1481e649d7fdee999060601b392008f989d0552a
}
[root@localhost ~]# systemctl restart wpa_supplicant
[root@localhost ~]# dhclient wlan0
[root@localhost ~]# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.2.101  netmask 255.255.255.0  broadcast 192.168.2.255
    inet6 fd46:76b4:c52c:0:eelf:72ff:fed6:fcce  prefixlen 64  scopeid 0x0<global>
    ether ec:1f:72:d6:fc:ce  txqueuelen 1000  (Ethernet)
    RX packets 32  bytes 2676 (2.6 KiB)
    RX errors 0  dropped 14  overruns 0  frame 0
    TX packets 82  bytes 11565 (11.2 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

[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”).

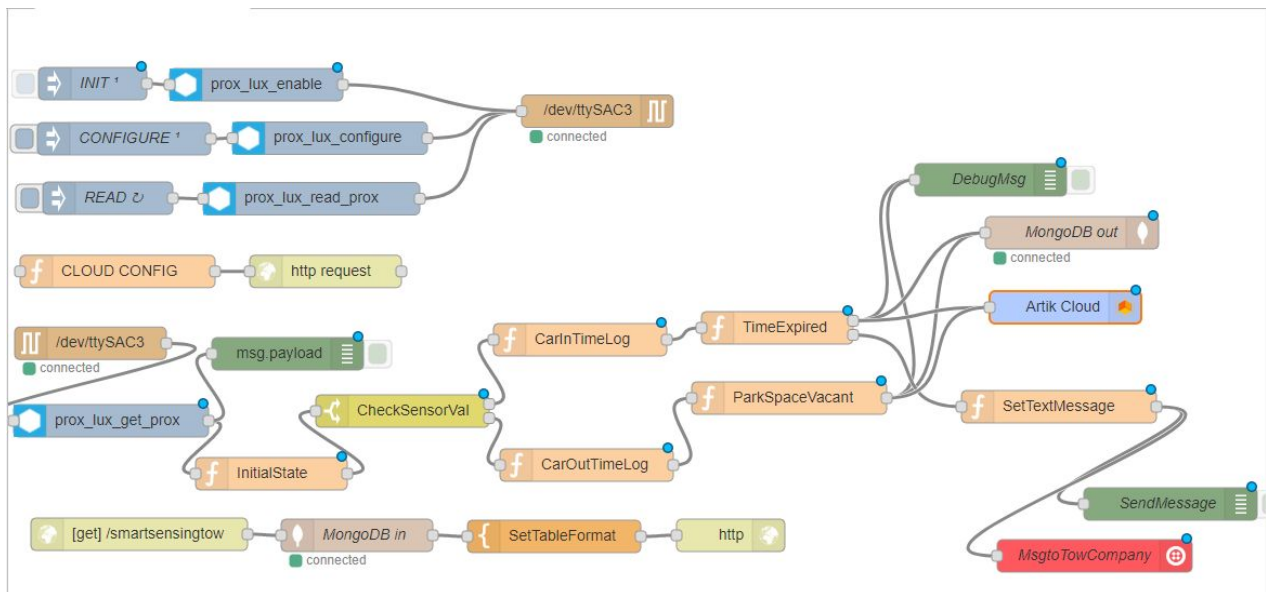
Drag an “artik adc” node to the right of the first node.

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

**Edit inject node**


Delete


Cancel

Done

 Payload


timestamp

 Topic

 Repeat

none

☒ Inject once at start?

 Name

INIT

**Note:** "interval between times" and "at a specific time" will use cron.  
See info box for details.


(2) Prox\_lux\_enable (Inject)

**Edit prox\_lux\_enable node**


Delete

Cancel

Done

 Name

Name

 Pin

1

☒ Enabled

Enabled


(3) CONFIGURE (Inject)

**Edit inject node**


Delete


Cancel

Done

 Payload


▼ timestamp

 Topic

 Repeat

none ▼

☒ Inject once at start?

 Name

**Note:** "interval between times" and "at a specific time" will use cron.  
See info box for details.


(4) Prox\_lux\_configure (kitra Input)

**Edit prox\_lux\_configure node**


Delete

Cancel


Done

 Name


Pin

 Disable lux

Yes ▼

 Disable prox

No ▼

 Extended range


(5) READ (Inject)

**Edit inject node**


Delete


Cancel

Done

 Payload

▼ timestamp

 Topic

 Repeat

interval ▼

every


10

▲▼

seconds ▼

☐

Inject once at start?

 Name

READ

**Note:** "interval between times" and "at a specific time" will use cron.  
See info box for details.


(6) Prox\_lux\_read\_prox (kitra Input)

**Edit prox\_lux\_read\_prox node**


Delete

Cancel

Done

 Name

Name

 Pin

1

(7) /dev/ttySAC3 (serial out)

**Edit serial out node**

Delete Cancel Done

Serial Port

Name

(8) CLOUD CONFIG (function)

**Edit function node**

Delete Cancel Done

Name

Function

```
1 msg.headers = {
2   "Content-Type": "application/json",
3   "Authorization": "Bearer <KEY>"
4 };
5 msg.payload = {"data":
6   {"prox1": msg.payload},
7   "sdid": "<ID>",
8   "type": "message"
9 };
10 return msg;
```

(9) Https request (function, https request)

**Edit http request node**

Delete Cancel Done

Method POST

URL https://api.artik.cloud/v1.1/messages

☐ Enable secure (SSL/TLS) connection

☐ Use basic authentication

Return a UTF-8 string

Name Name

(10) /dev/ttySAC3 (serial in)

**Edit serial in node**

Delete Cancel Done

Serial Port /dev/ttySAC3:115200-8N1

Name Name

(11) Prox\_lux\_get\_prox (kirta Input)

**Edit prox\_lux\_get\_prox node**

Delete Cancel Done

Name

(12) Msg.payload (debug)

**Edit debug node**

Delete

Cancel

Done

Output

▼ msg.payload

to

debug tab ▼

Name

Name

(13) Initial State (function)

**Edit function node**

Delete

Cancel

Done

Name

InitialState

Function

```
1 global.set('parkingSpace','Santa Clara University');
2 global.set('zoneName','a');
3 global.set('lotNumber','2');
4 global.set('id','dipal');
5
6 var str = "Lot ";
7 str = str.concat(global.get('lot'));
8 str = str.concat(" Space ");
9 str = str.concat(global.get('space'));
10 global.set('location', str);
11 return msg;
12
```

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)

**Edit switch node**

Delete

Cancel

Done

Name

CheckSensorVal

Property

▼ msg.payload

≡

>

▼

▼ a<sub>z</sub> 40

→ 1

✕

≡

<=

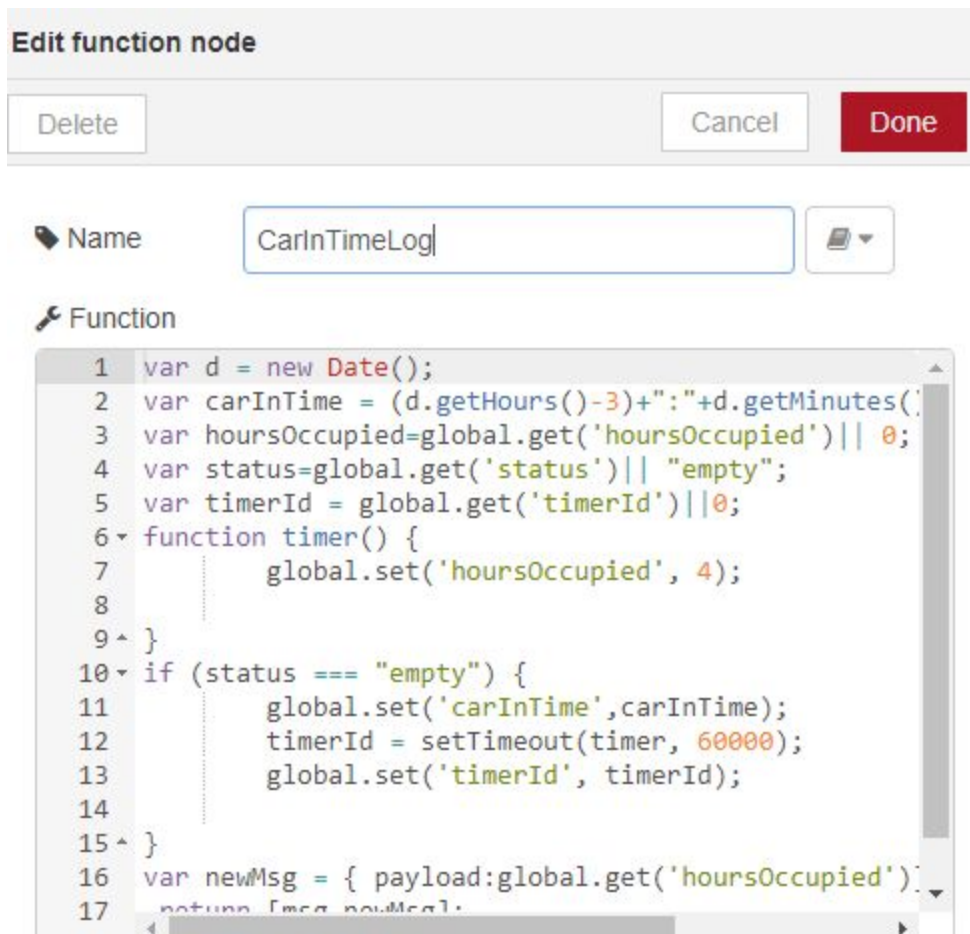
▼

▼ a<sub>z</sub> 40

→ 2

✕

(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)


**Edit function node**

Delete

Cancel

Done

 Name  

 Function

```
1 var hoursOccupied=global.get('hoursOccupied')||0;
2 var timerId = global.get('timerId')||0;
3 var status=global.get('status')||"empty";
4 global.set('counter',0);
5
6 clearTimeout(timerId);
7 global.set('hoursOccupied',0);
8 global.set('msgSent',false);
9 return msg;
10
```

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)


**Edit function node**

Delete


Cancel

Done

 Name

 Function

```
1 var hoursOccupied=global.get('hoursOccupied');
2 msg.payload={
3   "parkingSpace":global.get('parkingSpace'),
4   "zoneName":global.get('zoneName'),
5   "lotNumber":global.get('lotNumber'),
6   "carInTime":global.get('carInTime'),
7   "status":"full"
8 }
9
10 msg_id=global.get('id');
11 global.set('status',"full");
12
13 function counterSet() {
14   console.log("Hey");
15   var a = global.get('counter') + 1;
16   if(a<4){
17     global.set('counter', a);
18   }
19 }
```

 Outputs

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)

Edit function node

Delete
Cancel
Done

Name
ParkSpaceVacant

Function

```

1 msg.payload={
2     "parkingSpace":global.get('parkingSpace'),
3     "zoneName":global.get('zoneName'),
4     "lotNumber":global.get('lotNumber'),
5     "carInTime": 0,
6     "status":"empty",
7     "hoursOccupied":global.get('hoursOccupied')
8 }
9 msg_id=global.get('id');
10 global.set('status',"empty");
11
12 return msg;
13

```

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)

**Edit debug node**

Delete
Cancel
Done

Output
msg.payload

to
debug tab

Name
DebugMsg

(20) MongoDB out (mongodb out)

**Edit mongodb out node**

Delete
Cancel
Done

Server
tow\_MongoDB

Collection
Smartowing

Operation
save

☐ Only store msg.payload object

Name
MongoDB out

In the edit button right to server text field ->

mongodb out > **Edit mongodb node**

Delete

Cancel

Update

Host

ds115436.mlab.com|

Port

1543

Database

smarttowing

Username

AdminUser

Password

.....

Name

tow\_MongoDB

(21) Artik Cloud (artik cloud)

**Edit artik cloud node**

Delete

Cancel

Done

Name

Artik Cloud|

Device ID

b816aa99d7474a2e969071d3aad42a16

Device Token

.....


(22) SetTextMessage (function)


**Edit function node**

Delete

Cancel

Done

 Name

 Function

```
1 if (global.get('hoursOccupied') === 4 && global.get('m
2 {
3   msg.payload = "Please Tow the car from parking lot 2";
4   global.set('msgSent', true);
5 }
6
7
8 return msg;
```

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)


**Edit debug node**


Delete

Cancel

Done

 Output

 to

 Name




(24) MsgtoTowCompany (Twilio)

**Edit twilio out node**


Delete


Cancel

Done


 Twilio

+1 830-420-0541




 Output

SMS

 To

+16692548594

 Name

MsgtoTowCompany

In the edit button next to twilio text field ->

twilio out > **Edit twilio-api node**


Delete

Cancel


Update

Account SID


AC0b6d0ae02aa91bee9182acc33b8effbd

 From

+1 830-420-0541

 Token

.....

 Name

Name

(25) Smartsensingtow (http out)

**Edit http in node**

Delete

Cancel

Done

Method

GET

URL

/smartsensingtow

Name

Name

(26) Mongo DB in (mongodb in)

**Edit mongodb in node**

Delete

Cancel

Done

Server

tow\_MongoDB

Collection

Smartowing

Operation

aggregate

Name

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)

Edit http response node

Delete
Cancel
Done

Name
Name

The messages sent to this node **must** originate from an *http input* node

## Appendix A

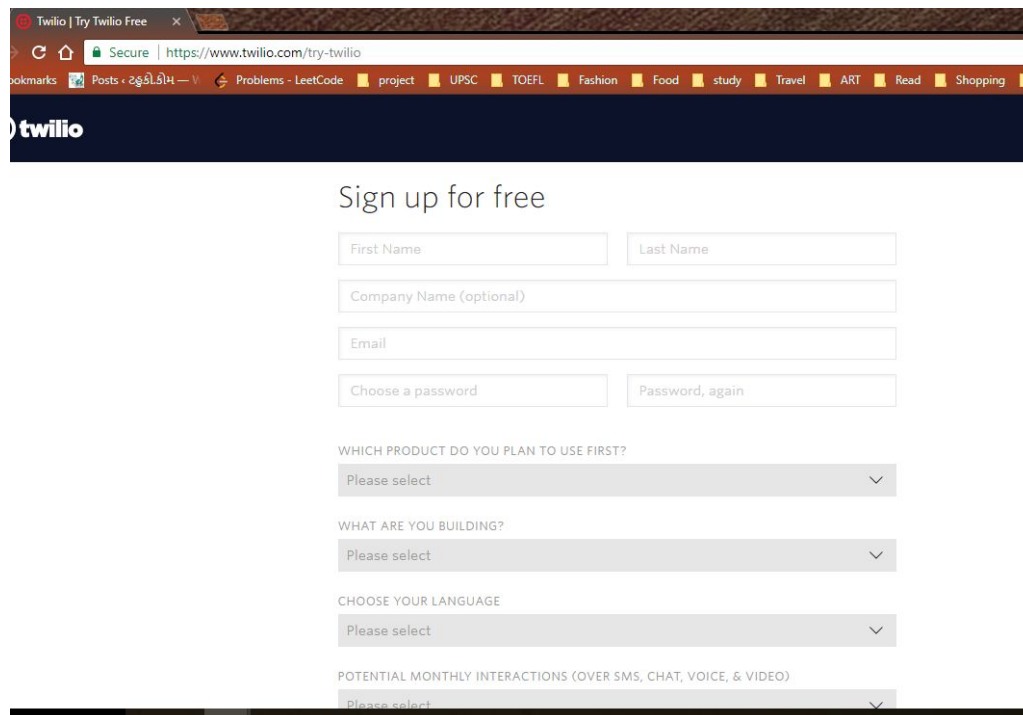
### Sending Text Message in Node-RED through Twilio

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

#### (A) Setup Twilio Account

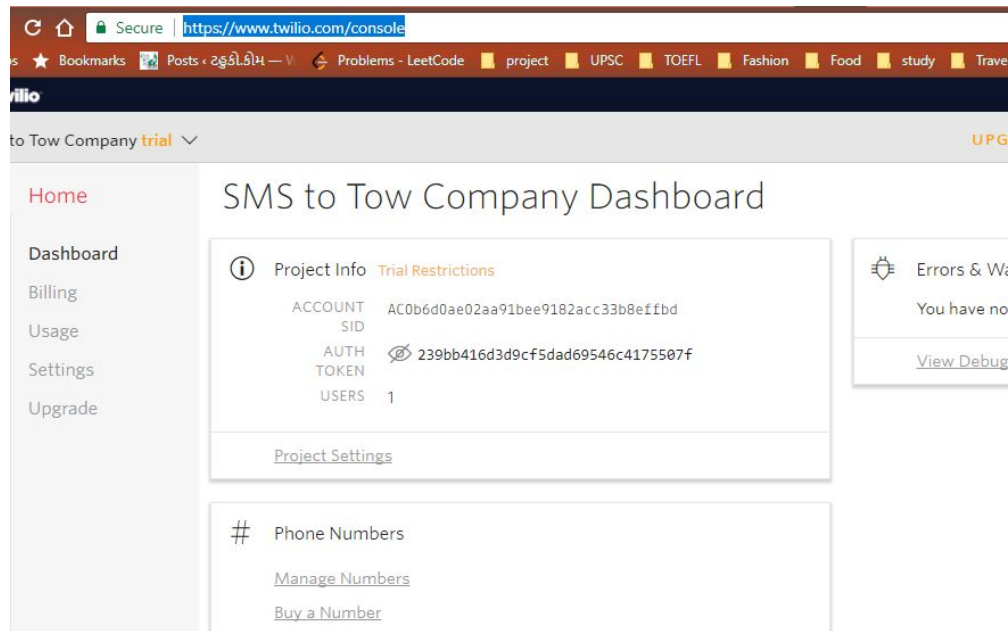
The first step is to create a twilio account and setup a number to send text message.

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

## Phone Numbers

Number

▼

Voice URL

▼

Filter

<div>+</div> NUMBER	FRIENDLY NAME	CAPABILITIES				CONFIGURATION
		VOICE	FAX	SMS	MMS	
<div>+1 830-420-0541</div> <div>Marion, TX</div>	(830) 420-0541	<div>📞</div>	<div>📠</div>	<div>💬</div>	<div>📧</div>	<div>Voice URL: <a href="https://demo.twilio.com/welcome/voice/">https://demo.twilio.com/welcome/voice/</a></div> <div>Messaging URL: <a href="https://demo.twilio.com/welcome/sms/reply/">https://demo.twilio.com/welcome/sms/reply/</a></div>

\* Can send/receive calls to domestic numbers only

† Can send/receive sms to domestic numbers only

‡ This number does NOT support SIP Trunking

(beta) This number is new to the Twilio Platform

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)

```
C:\Users\Dipal> node-red
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 are
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.

```
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.



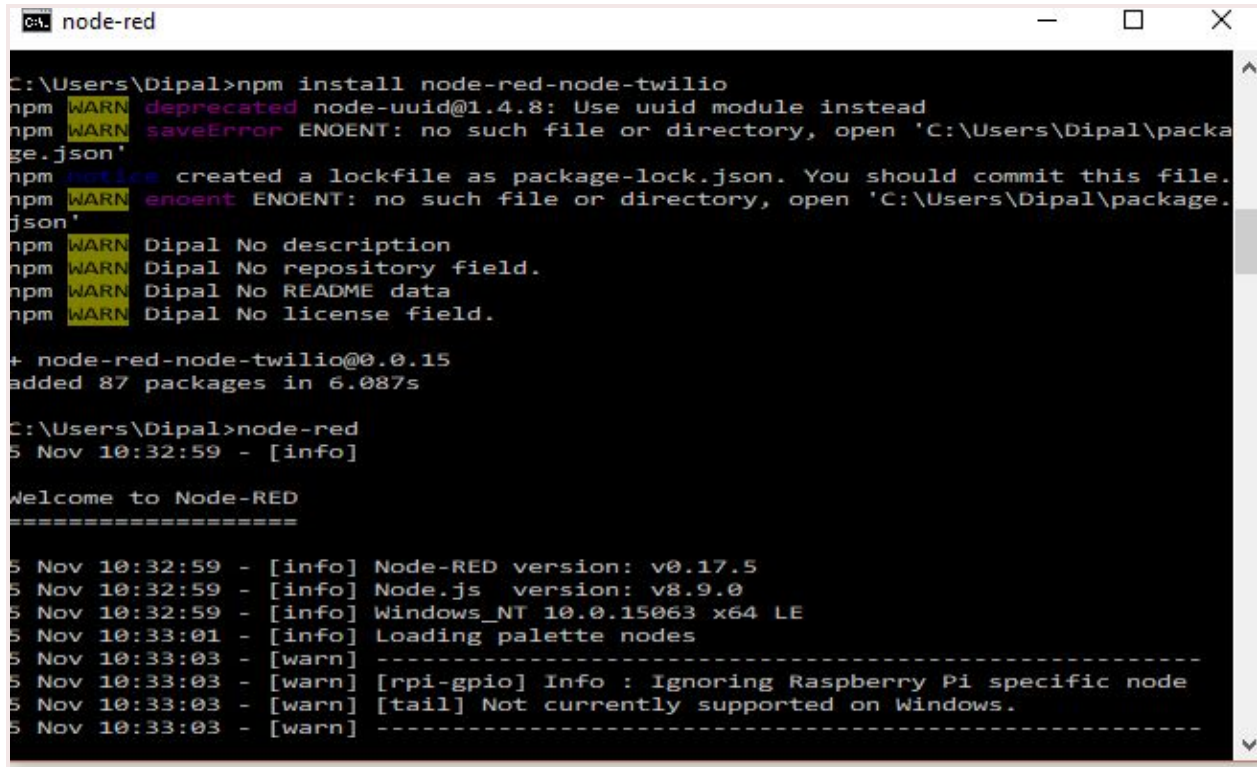
### (C) Adding Twilio node in Node-RED

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>]



```
node-red
C:\Users\Dipal>npm install node-red-node-twilio
npm WARN deprecated node-uuid@1.4.8: Use uuid module instead
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\Dipal\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\Dipal\package.json'
npm WARN Dipal No description
npm WARN Dipal No repository field.
npm WARN Dipal No README data
npm WARN Dipal No license field.

+ node-red-node-twilio@0.0.15
added 87 packages in 6.087s

C:\Users\Dipal>node-red
5 Nov 10:32:59 - [info]

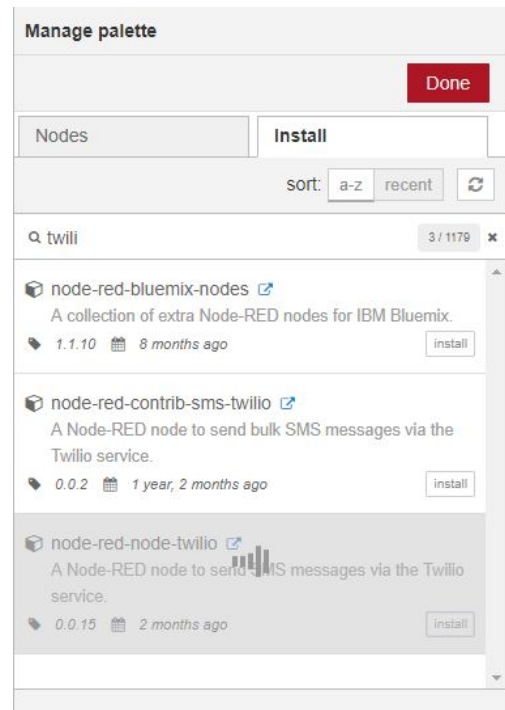
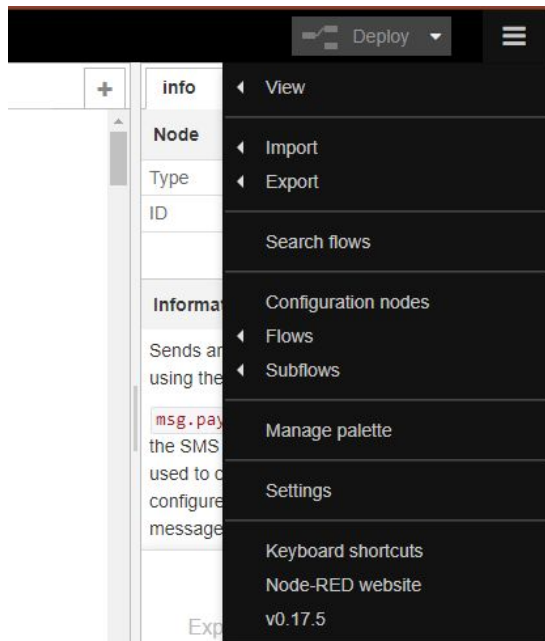
Welcome to Node-RED
=====

5 Nov 10:32:59 - [info] Node-RED version: v0.17.5
5 Nov 10:32:59 - [info] Node.js version: v8.9.0
5 Nov 10:32:59 - [info] Windows_NT 10.0.15063 x64 LE
5 Nov 10:33:01 - [info] Loading palette nodes
5 Nov 10:33:03 - [warn] -----
5 Nov 10:33:03 - [warn] [rpi-gpio] Info : Ignoring Raspberry Pi specific node
5 Nov 10:33:03 - [warn] [tail] Not currently supported on Windows.
5 Nov 10:33:03 - [warn] -----
```

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



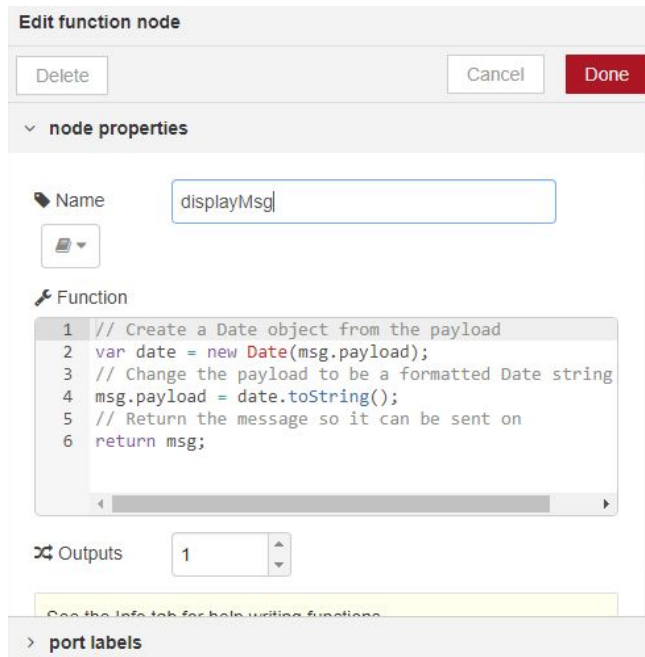
and search for twilio and install.



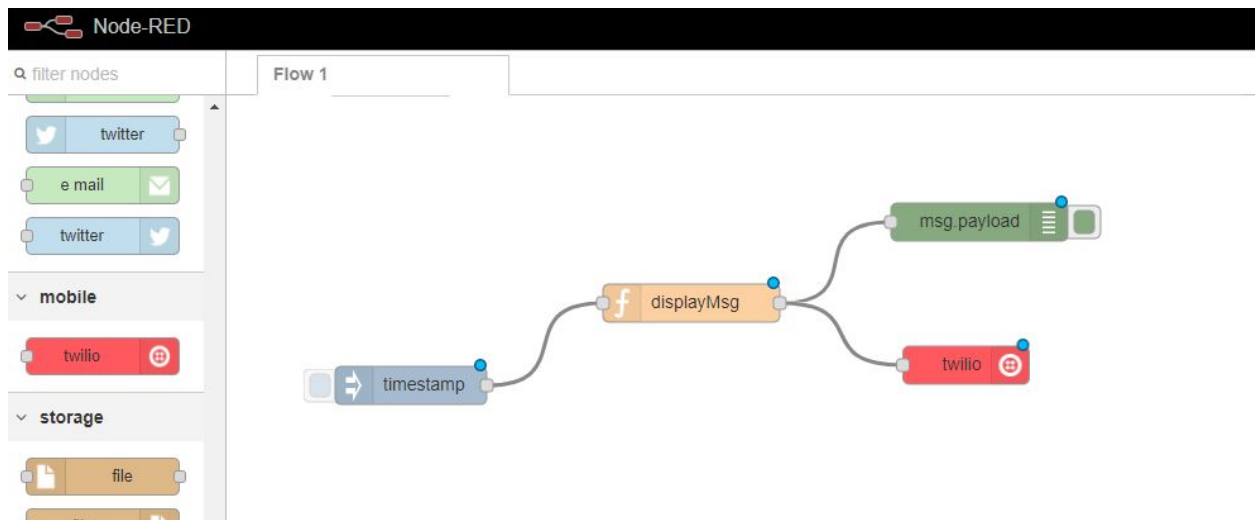
#### (D) Adding Twilio node and executing

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 through 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,

**Edit twilio out node**

Delete Cancel Done

node properties

Twilio +1 830-420-0541

Output SMS

To +16692548594

Name Name

> port labels

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.

twilio out > **Edit twilio-api node**

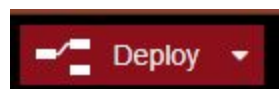
Delete Cancel Update

Account SID AC0b6d0ae02aa91bee9182acc33b8effbd

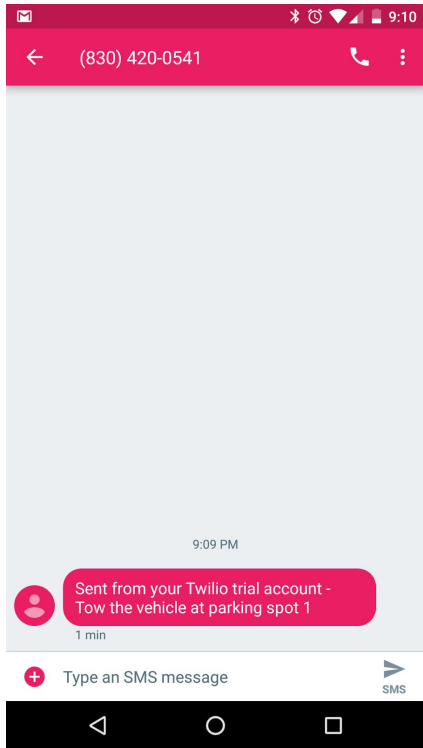
From +1 830-420-0541

Token .....

Name Name



Everything is set now, click on and inject the timestamp inject node. And text message will be sent to your given number.

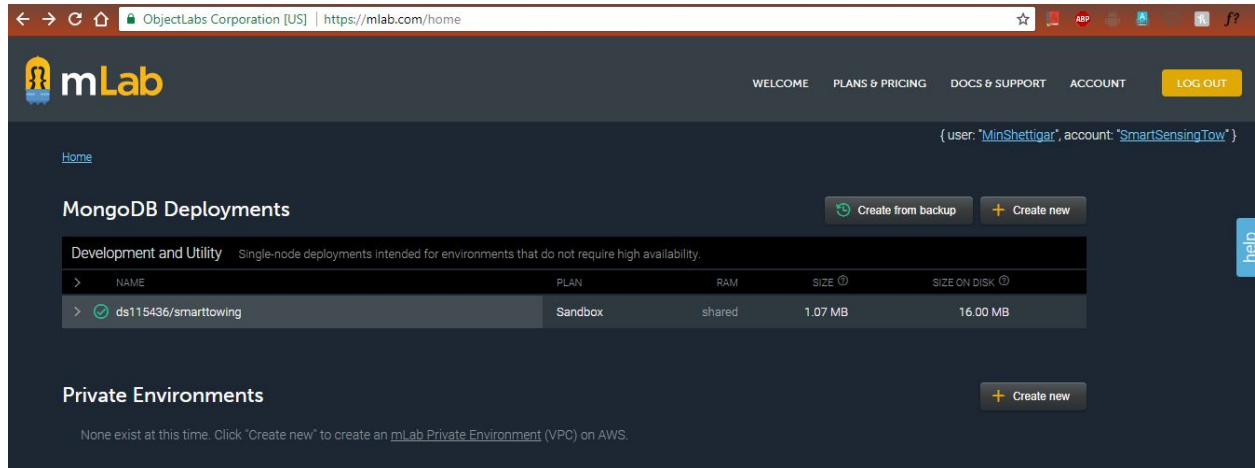


## Appendix B - MongoDB connection

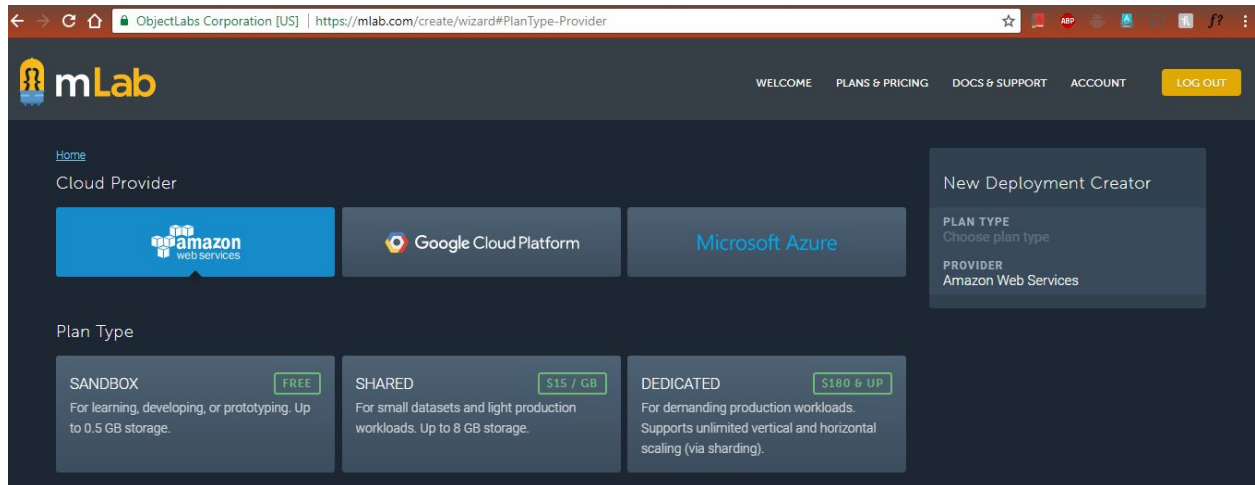
Run the following command in your Node-RED user directory to install MongoDB.

- `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

## Database: smarttowing

To connect using the mongo shell:

```
% mongo ds115436.mlab.com:15436/smarttowing -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard MongoDB URI ([what's this?](#)):

```
mongodb://<dbuser>:<dbpassword>@ds115436.mlab.com:15436/smarttowing
```

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

The screenshot shows the mLab web interface for a database named 'smarttowing'. The top navigation bar includes links for 'WELCOME', 'PLANS & PRICING', 'DOCS & SUPPORT', 'ACCOUNT', and a 'LOG OUT' button. A user profile is shown as '{ user: "MinShettigar", account: "SmartSensingTow" }'. The main content area has a 'Database: smarttowing' header with a 'Delete database' button. Below this is a code block with connection instructions for the mongo shell and a MongoDB URI. A warning message states: 'Sandbox databases do not have redundancy and therefore are not suitable for production. Read our documentation on how to upgrade.' A tabbed interface shows 'Collections' as the active tab, with other tabs for 'Users', 'Stats', 'Backups', and 'Tools'. The 'Collections' section includes a table with columns 'NAME', 'DOCUMENTS', 'CAPPED?', and 'SIZE'. The table lists two collections: 'parking' (9 documents, 10.09 KB) and 'Smarttowing' (3,747 documents, 1005.95 KB). Buttons for 'Delete all collections' and 'Add collection' are located above the table.

ObjectLabs Corporation [US] | <https://mlab.com/databases/smarttowing>

mLab

WELCOME PLANS & PRICING DOCS & SUPPORT ACCOUNT LOG OUT

{ user: "MinShettigar", account: "SmartSensingTow" }

Home

Database: smarttowing Delete database

To connect using the mongo shell:

```
% mongo ds115436.mlab.com:15436/smarttowing -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard MongoDB URI ([what's this?](#)):

```
mongodb://<dbuser>:<dbpassword>@ds115436.mlab.com:15436/smarttowing
```

mongod version: 3.4.9 (MMAPv1)

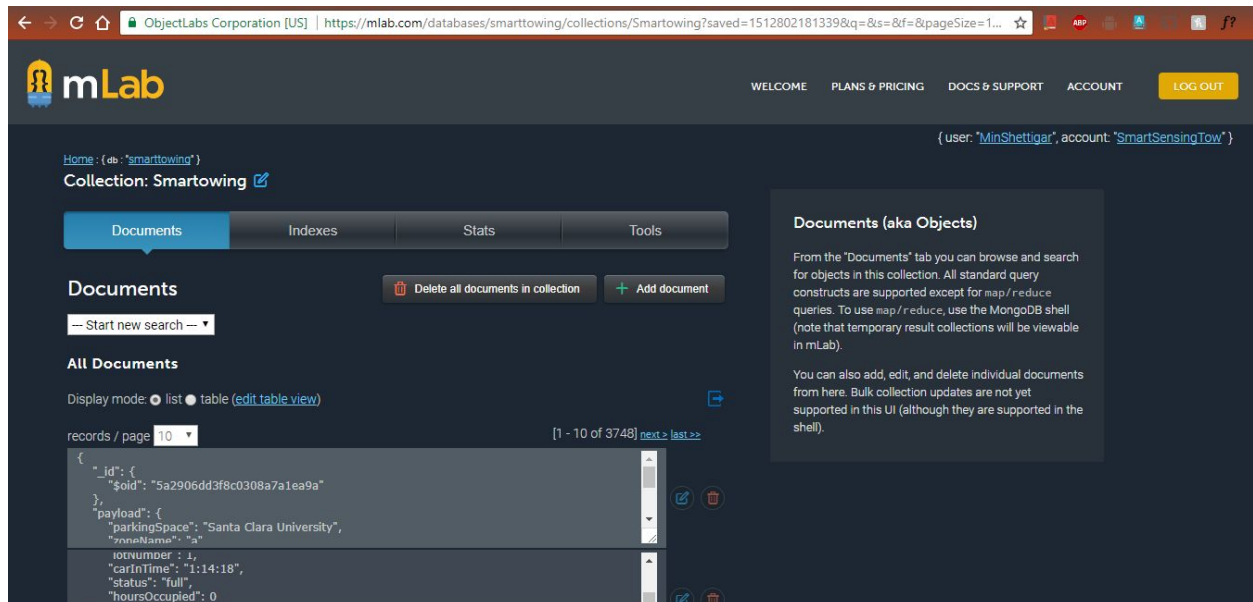
⚠ Sandbox databases do not have redundancy and therefore are not suitable for production. Read our documentation on [how to upgrade](#).

Collections Users Stats Backups Tools

Collections Delete all collections Add collection

NAME	DOCUMENTS	CAPPED?	SIZE
parking	9	false	10.09 KB
Smarttowing	3,747	false	1005.95 KB

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

mongodb out > **Edit mongodb node**

Delete Cancel Update

Host ds115436.mlab.com Port 1543

Database smarttowing

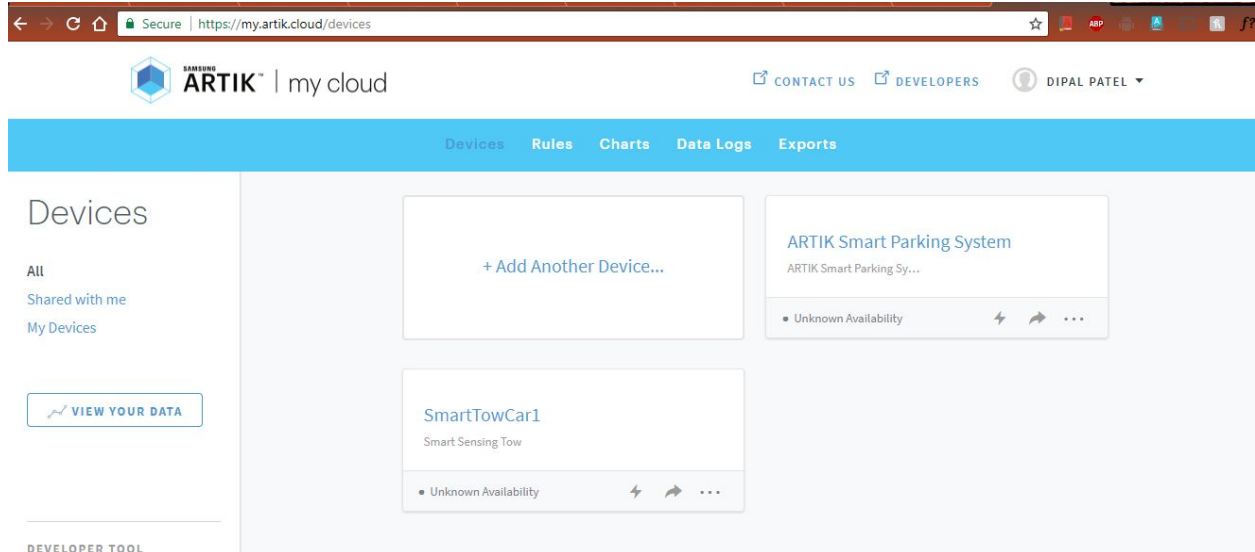
Username AdminUser

Password .....

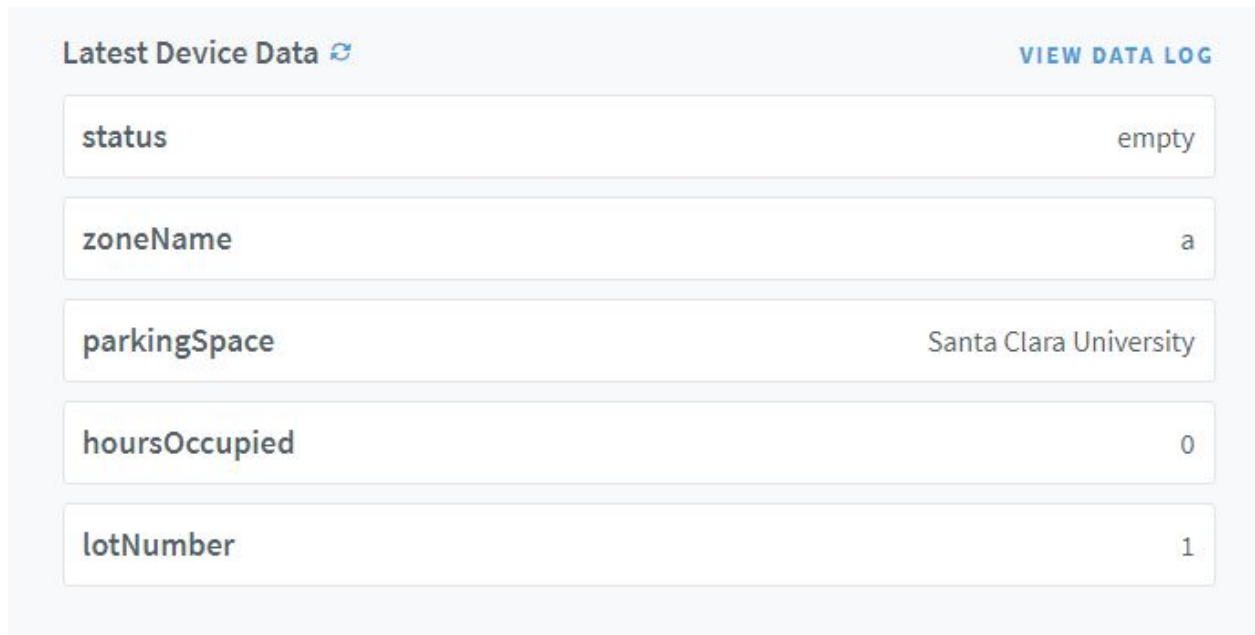
Name tow\_MongoDB

## Appendix C - ARTIK cloud

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

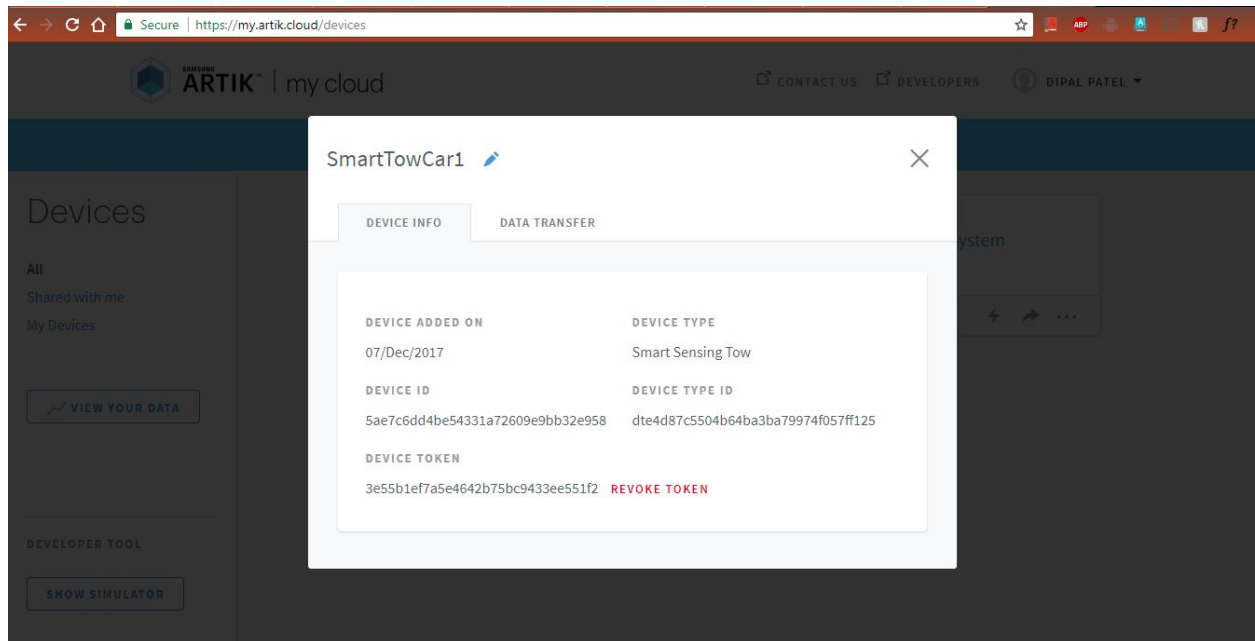


Search for “Smart Sensing tow” and select it. “Smart Sensing tow” is a public device type we created for our project, which includes parkingSpace: "Santa Clara University", zoneName: "a", lotNumber: "2", carInTime: 0 (time), status (“empty” or “full”), hoursOccupied (integer value) properties.





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.

The screenshot shows the 'Edit artik cloud node' dialog in Node-RED. The dialog has three input fields:

- Name:** Artik Cloud
- Device ID:** b816aa99d7474a2e969071d3aad42a16
- Device Token:** (masked with dots)

At the top of the dialog, there are three buttons: 'Delete', 'Cancel', and 'Done'.

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

[Devices](#)
[Rules](#)
[Charts](#)
[Data Logs](#)
[Exports](#)

+/- CHARTS

+

−

10s

30s

5m

1h

1d

1w

1m

6m

1y

Dec 8, 2017 20:54:39

TO

Dec 8, 2017 21:54:39

⏮

⏭

« OLDER

« HIMP TO DATA

» HIMP TO DATA

» NEWER »

6 CHARTS SELECTED FOR DISPLAY (50 MAXIMUM)

Filter by device

ARTIK Smart Parking System

☐ Lot

☐ Reserved

☐ Space

☐ State

SmartTowCar1

☒ CarInTime

☒ HoursOccupied

☒ LotNumber

☒ ParkingSpace

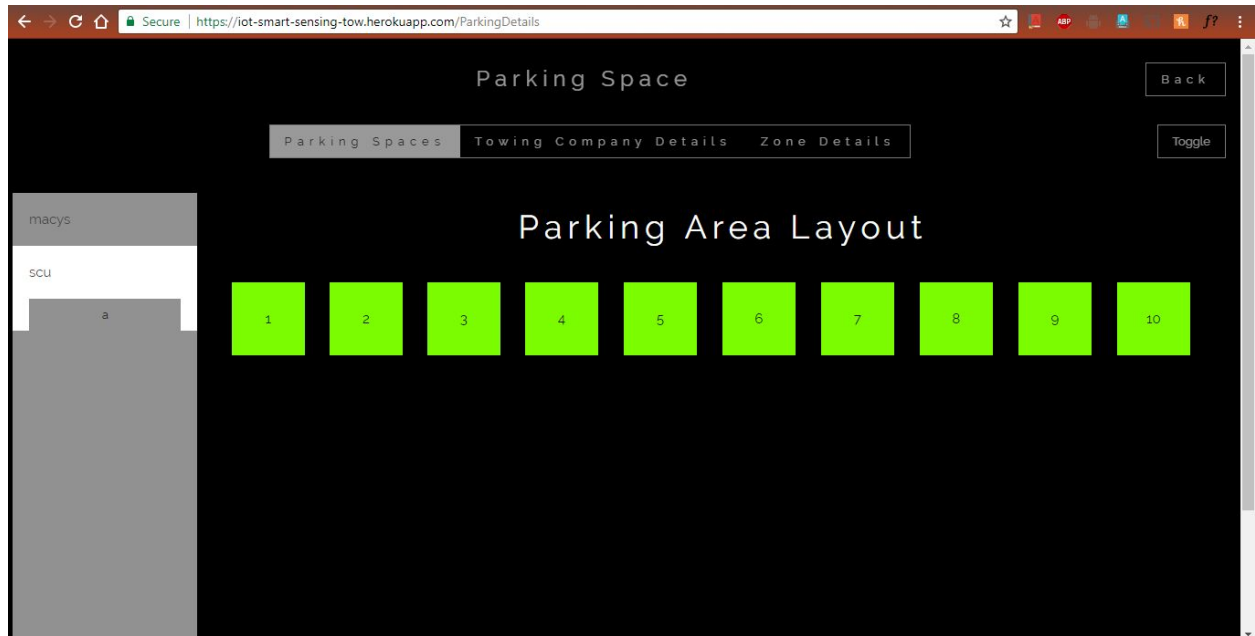
☒ Status

☒ ZoneName

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.

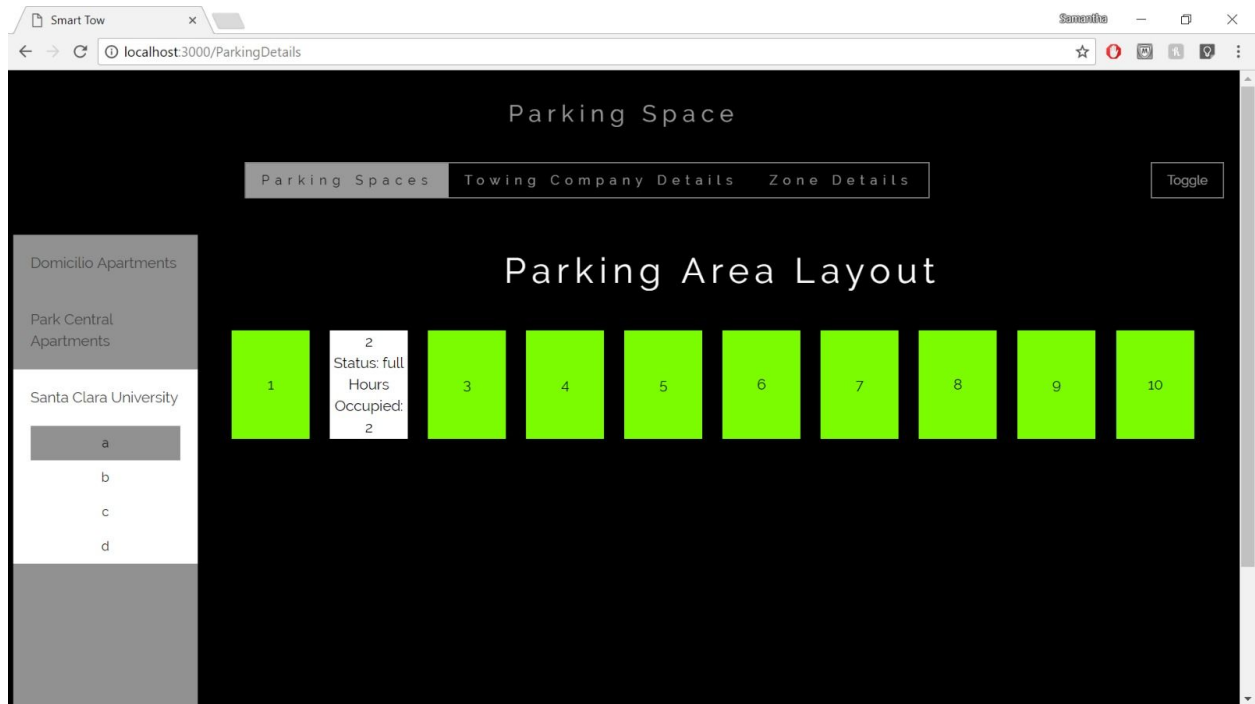
Used html and node js

Database - MongoDB

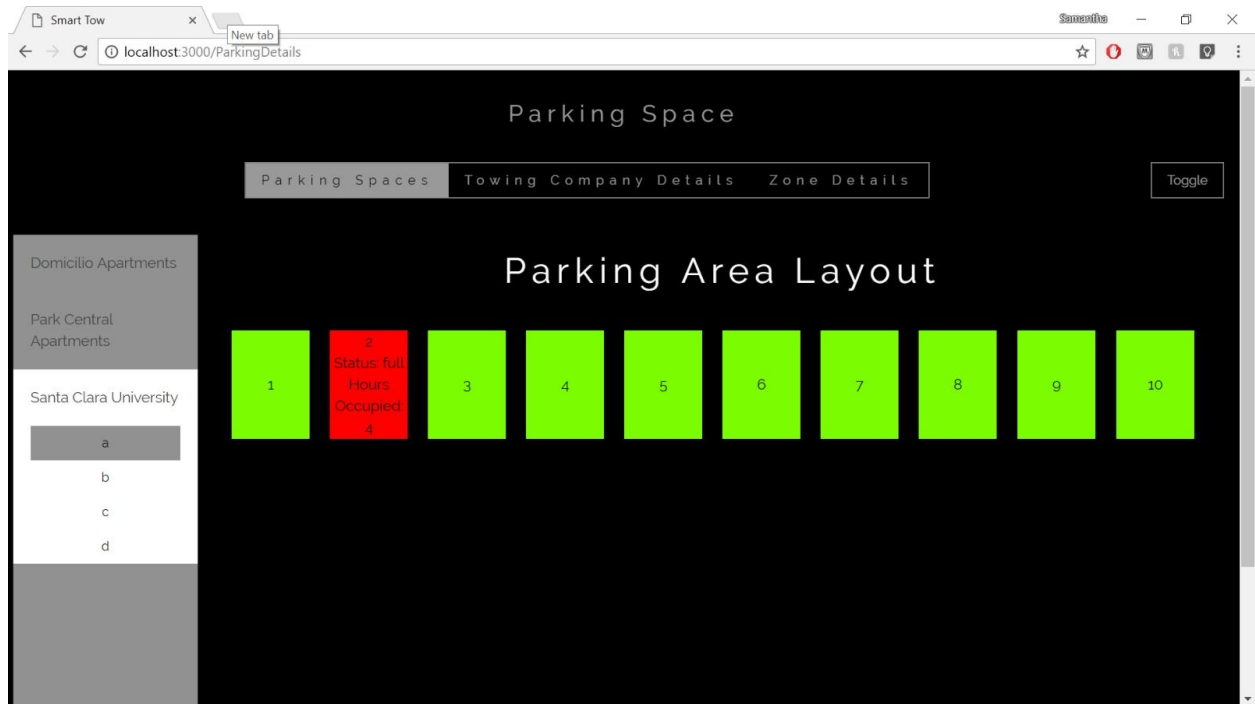
WorkFlow:

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.

The screenshot shows a web browser window with the URL <https://iot-smart-sensing-tow.herokuapp.com/addTow>. The page has a dark background with the title "Towing Company" at the top center. A "Back" button is in the top right corner. A white modal form titled "Add Towing Company Details" is centered on the screen. The form contains four input fields: "Towing Company Name:", "Location:", "Address:", and "Contact Details:". A "Submit" button is located at the bottom right of the form.

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

The screenshot shows a web browser window with the URL <https://iot-smart-sensing-tow.herokuapp.com/addZone>. The page has a dark background with the title "Towing Company" at the top center. A "Back" button is in the top right corner. Below the title, there are two tabs: "Parking Spaces" and "Zone Details". The "Zone Details" tab is active. A white modal form titled "Add Zone" is centered on the screen. The form contains one input field: "Zone Name:". A "Submit" button is located at the bottom right of the form. In the background, there is a list of items: "scu" and "macys".

New tab - for adding details of new zones into database

macys

scu

a

Parking Space

Back

Parking SpacesTowing Company DetailsZone Details

Grid

Parking Area Layout

Parking Number	Status	Hours Occupied
1	empty	0
2	empty	0
3	empty	0
4	empty	0
5	empty	0
6	empty	0
7	empty	0
8	empty	0
9	empty	0
10	empty	0

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>