**AWS IOT**

* Register your AWS thing in AWS (Aws)
* create policy(Aws)
* create certificate(aws)
* attach certificate with policy(aws)
* simulate device
* create action and rule on aws(Aws)
* insert message into dynamo db (Reopubluish messge to dyanamo db)
* Get data from dynamo db and create api gate way
* Take api end point fetch datae and visuklaize it



* Get that json data and visualize using html like above one

Follow aws iot pdf launch the isnatnce using template given by aws

**Files to read…**



Device.py

# This class provides class and methods for simple IoT device simulation

import random

import math

import datetime

import time

import json

import settings

# set some constants

MAX\_BATTERY = 100.0

GPS\_RADIUS = 1000

# method to move device

def gps\_circle(center\_lat, center\_lon, radius):

"""

Creates a circular path of GPS points for simulating thing movement around a point in meters

"""

N = 360 # number of discrete sample points to be generated along the circle

circle\_points = []

for k in range(N):

# compute

angle = math.pi \* 2 \* k / N

dx = radius \* math.cos(angle)

dy = radius \* math.sin(angle)

point = {}

point['lat'] = center\_lat + (180 / math.pi) \* (dy / 6378137)

point['lon'] = center\_lon + (180 / math.pi) \* (dx / 6378137) / math.cos(center\_lat \* math.pi / 180)

# add to list

circle\_points.append(point)

return circle\_points

# define the Device class

class Device:

def \_\_init\_\_(self, name, lat, lon): # initialize device with name and GPS coordinates

self.name = name

self.location = {}

self.path = gps\_circle(lat, lon, GPS\_RADIUS) # List of 360 degree GPS points

self.location = self.path[0] # set the initial position

self.batteryCharge = random.uniform(1, 100)

self.batteryDischargeRate = random.uniform(settings.BATTERY\_DISCHARGE\_RANGE[0], settings.BATTERY\_DISCHARGE\_RANGE[1])

self.sensorRange = random.randint(5, 20)# defines the maximum value the sensor can detect - this is just to provide some randomization of the max val

# self.sensorReading = random.randint(settings.RANDOM\_INTEGER\_RANGE[0],settings.RANDOM\_INTEGER\_RANGE[1])

self.sensorReading = random.randint(1, self.sensorRange)

self.timeStampEpoch = int(time.time() \* 1000)

self.timeStampIso = datetime.datetime.isoformat(datetime.datetime.now())

def get\_payload(self):

return json.dumps({"batteryDischargeRate": self.batteryDischargeRate, "sensorReading": self.sensorReading, "deviceId": self.name, "timeStampEpoch": self.timeStampEpoch, "timeStampIso": self.timeStampIso, "batteryCharge": self.batteryCharge, "location": {"lat": self.location['lat'], "lon": self.location['lon']}})

def update(self):

self.batteryDischargeRate = random.uniform(settings.BATTERY\_DISCHARGE\_RANGE[0], settings.BATTERY\_DISCHARGE\_RANGE[1])

self.batteryCharge = self.batteryCharge - self.batteryDischargeRate

# check battery charge, and if it's zero, sensorReading "sensor" doesn't work

if self.batteryCharge <= 0.0:

self.batteryCharge = 0.0

self.sensorReading = 0

else:

# self.sensorReading = random.randint(settings.RANDOM\_INTEGER\_RANGE[0],settings.RANDOM\_INTEGER\_RANGE[1])

self.sensorReading = random.randint(1, self.sensorRange)

self.timeStampEpoch = int(time.time() \* 1000) # update timestamp

self.timeStampIso = datetime.datetime.isoformat(datetime.datetime.now()) # update timestamp

self.move() # move the device

def recharge\_device\_battery(self):

self.batteryCharge = 100.0

def move(self):

"""Set GPS location to the next spot on the path"""

self.location = self.path[0] # Set current location to the next item in the path.

# print(json.dumps(self.location))

# print(json.dumps(self.path[0]))

# print(len(self.path))

# Move location to the end of the path

self.path.remove(self.location)

self.path.append(self.location)

settings.py

"""

Modify these values to match your configuration

"""

# AWS IoT endpoint settings

HOST\_NAME = "a2wbvtm3kolnb3.iot.ap-southeast-1.amazonaws.com" # replace with your AWS IoT endpoint for your region

HOST\_PORT = 8883 # leave this as-is

# thing certs & keys

PRIVATE\_KEY = "certs/private.pem.key" # replace with your private key name

DEVICE\_CERT = "certs/certificate.pem.crt" # replace with your certificate name

ROOT\_CERT = "certs/root-ca.pem"

# device & message settings

BATTERY\_DISCHARGE\_RANGE = (1, 3) # tuple that stores the possible discharge rates of the battery

# RANDOM\_INTEGER\_RANGE = (1,10) # tuple that stores the possible range of your sensor reading

QOS\_LEVEL = 0 # AWS IoT supports QoS levels 0 & 1 for MQTT sessions

~

App.py

#### Leverages the AWS IoT SDK for Python

# Import SDK packages

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

import settings # importing configuration file

import device # import device class

import time

import random

import string

# set up the gps coords variables

def random\_GPS():

"""Returns a random Latitude/Longitude pair roughly located in the Continental US"""

lat = random.uniform(30, 50)

lon = random.uniform(-77, -122)

coor = (lat,lon)

return coor

# create devices

dev\_names = ['turing', 'hopper', 'knuth']

devs = [] # set up a list to hold your devices

for i in dev\_names:

coor = random\_GPS()

devs.append(device.Device(i, coor[0], coor[1]))

RECHARGE\_ALERT\_TOPIC = "device/+/rechargeAlert" # subscribe to wildcard device\_id

# Create a random 8-character string for connection id

CLIENT\_ID = ''.join(random.SystemRandom().choice(string.ascii\_uppercase + string.digits) for \_ in range(8))

def on\_message(client, userdata, msg):

# CURRENT\_COLOR = new\_color()

# print(CURRENT\_COLOR+"ALERT: Received message on topic "+msg.topic+" with payload: "+str(msg.payload))

# print("ALERT: Received message on topic "+msg.topic+" with payload: "+str(msg.payload))

name = msg.topic.split('/')[1]

print("ALERT: Received message on topic "+msg.topic+" to recharge device: " + name)

# loop through device list to recharge the correct device

for x in devs:

if x.name == msg.topic.split('/')[1]:

x.recharge\_device\_battery()

# create AWS IoT MQTT client

client = AWSIoTMQTTClient(CLIENT\_ID)

# configure client endpoint / port information & then set up certs

client.configureEndpoint(settings.HOST\_NAME, settings.HOST\_PORT)

client.configureCredentials(settings.ROOT\_CERT, settings.PRIVATE\_KEY, settings.DEVICE\_CERT)

# configure client connection behavior

client.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing

client.configureDrainingFrequency(2) # Draining: 2 Hz

client.configureConnectDisconnectTimeout(10) # 10 sec

client.configureMQTTOperationTimeout(5) # 5 sec

print("Connecting to endpoint " + settings.HOST\_NAME)

client.connect()

print("Subscribing to " + RECHARGE\_ALERT\_TOPIC)

client.subscribe(RECHARGE\_ALERT\_TOPIC, 1, on\_message)

# start loop to begin publishing to topic

while True:

for dev in devs:

client.publish("device/" + dev.name + "/devicePayload", dev.get\_payload(), settings.QOS\_LEVEL)

print("Published message on topic " + "device/" + dev.name + "/devicePayload" + " with payload: " + dev.get\_payload())

dev.update() # update the device with new data

time.sleep(5) # just wait a sec before publishing next message

DASHBOARD FOLDER

INDEX.HTML

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.5/angular.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.5/angular-route.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.js"></script>

<script src="https://cdn.rawgit.com/esvit/ng-table/1.0.0/dist/ng-table.js"></script>

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

<script src="http://openlayers.org/en/v3.18.2/build/ol.js" type="text/javascript"></script>

<!-- Bootstrap -->

<link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

<!-- Font-awesome -->

<link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-awesome.min.css">

<link rel="stylesheet" href="css/main.css" type="text/css">

<script src="app.js"></script>

<!-- This file covers 3 devices, a real program should be designed to handle any number of devices -->

<div ng-app="ShowMap" ng-controller="mapController" ng-init="func()">

<div class="container">

<!-- Static navbar -->

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">AWS IoT Getting Started</a>

</div>

<div id="navbar" class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li class="active"><a href="#">Home</a></li>

<li><a href="https://aws.amazon.com/iot/" target="\_blank">AWS IoT</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

</ul>

</div>

<!--/.nav-collapse -->

</div>

<!--/.container-fluid -->

</nav>

<div class="panel panel-default">

<div class="panel-heading">

<div class="panel-title">

<rd-widget-header title=" Devices Geolocation" icon="fa-cog fa-spin"></rd-widget-header>

</div>

</div>

<div ng-show="showSpinkit">

<div class="spinner">

<div class="bounce1"></div>

<div class="bounce2"></div>

<div class="bounce3"></div>

</div>

</div>

<div class="panel-body" ng-show="!showSpinkit">

<center>

<div id="mapdiv" style="width: 650px; height: 360px;" align="center"></div>

</center>

</div>

</div>

<!-- -->

<div class="row">

<div class="col-md-4 portfolio-item">

<h3>

<a href="#">turing</a>

</h3>

<p></p>

<!-- turing Battery Charge -->

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Battery Charge</h3>

</div>

<div class="panel-body">

<div ng-show="showSpinkit\_tables">

<div class="spinner">

<div class="bounce1"></div>

<div class="bounce2"></div>

<div class="bounce3"></div>

</div>

</div>

<div id="chart\_div\_battery\_charge\_turing" ng-show="!showSpinkit\_tables"></div>

</div>

</div>

</div>

<div class="col-md-4 portfolio-item">

<h3>

<a href="#">hopper</a>

</h3>

<p></p>

<!-- hopper Battery Charge -->

<div class="panel panel-default">

<rd-loading ng-show="showSpinkit"> </rd-loading>

<div class="panel-heading">

<h3 class="panel-title">Battery Charge</h3>

</div>

<div class="panel-body">

<div ng-show="showSpinkit\_tables">

<div class="spinner">

<div class="bounce1"></div>

<div class="bounce2"></div>

<div class="bounce3"></div>

</div>

</div>

<div id="chart\_div\_battery\_charge\_hopper" ng-show="!showSpinkit\_tables"></div>

</div>

</div>

</div>

<div class="col-md-4 portfolio-item">

<h3>

<a href="#">knuth</a>

</h3>

<p></p>

<!-- knuth Battery Charge -->

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Battery Charge</h3>

</div>

<div class="panel-body">

<div ng-show="showSpinkit\_tables">

<div class="spinner">

<div class="bounce1"></div>

<div class="bounce2"></div>

<div class="bounce3"></div>

</div>

</div>

<div id="chart\_div\_battery\_charge\_knuth" ng-show="!showSpinkit\_tables"></div>

</div>

</div>

</div>

</div>

<!-- row Discharge-->

<div class="row">

<!-- Discharge turing -->

<div class="col-md-4 portfolio-item">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Battery Discharge</h3>

</div>

<div class="panel-body">

<div ng-show="showSpinkit\_tables">

<div class="spinner">

<div class="bounce1"></div>

<div class="bounce2"></div>

<div class="bounce3"></div>

</div>

</div>

<div id="chart\_div\_battery\_discharge\_turing" ng-show="!showSpinkit\_tables"></div>

</div>

</div>

</div>

<!-- Discharge hopper -->

<div class="col-md-4 portfolio-item">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Battery Discharge</h3>

</div>

<div class="panel-body">

<div ng-show="showSpinkit\_tables">

<div class="spinner">

<div class="bounce1"></div>

<div class="bounce2"></div>

<div class="bounce3"></div>

</div>

</div>

<div id="chart\_div\_battery\_discharge\_hopper" ng-show="!showSpinkit\_tables"></div>

</div>

</div>

</div>

<!-- Discharge knuth" -->

<div class="col-md-4 portfolio-item">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Battery Discharge</h3>

</div>

<div class="panel-body">

<div ng-show="showSpinkit\_tables">

<div class="spinner">

<div class="bounce1"></div>

<div class="bounce2"></div>

<div class="bounce3"></div>

</div>

</div>

<div id="chart\_div\_battery\_discharge\_knuth" ng-show="!showSpinkit\_tables"></div>

</div>

</div>

</div>

</div>

<!-- row -->

<!-- row Sensor Data (numval) -->

<div class="row">

<!-- Discharge turing -->

<div class="col-md-4 portfolio-item">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Sensor Data</h3>

</div>

<div class="panel-body">

<div ng-show="showSpinkit\_tables">

<div class="spinner">

<div class="bounce1"></div>

<div class="bounce2"></div>

<div class="bounce3"></div>

</div>

</div>

<div id="chart\_div\_sensor\_data\_turing" ng-show="!showSpinkit\_tables"></div>

</div>

</div>

</div>

<!-- Discharge hopper -->

<div class="col-md-4 portfolio-item">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Sensor Data</h3>

</div>

<div class="panel-body">

<div ng-show="showSpinkit\_tables">

<div class="spinner">

<div class="bounce1"></div>

<div class="bounce2"></div>

<div class="bounce3"></div>

</div>

</div>

<div id="chart\_div\_sensor\_data\_hopper" ng-show="!showSpinkit\_tables"></div>

</div>

</div>

</div>

<!-- Discharge knuth" -->

<div class="col-md-4 portfolio-item">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Sensor Data</h3>

</div>

<div class="panel-body">

<div ng-show="showSpinkit\_tables">

<div class="spinner">

<div class="bounce1"></div>

<div class="bounce2"></div>

<div class="bounce3"></div>

</div>

</div>

<div id="chart\_div\_sensor\_data\_knuth" ng-show="!showSpinkit\_tables"></div>

</div>

</div>

</div>

</div>

<!-- row -->

</div>

<div class="row">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title"> <i class="fa fa-cog fa-spin" style="font-size:24px"></i> Devices Data</h3>

</div>

<div class="panel-body">

<div class="table-responsive" icon="fa-cog fa-spin" title="Devices Information">

<table ng-table="tableParams" class="table table-striped" show-filter="false">

<tr>

<th>Device Id</th>

<th>Battery Charge</th>

<th>Battery Discharge Rate</th>

<th>Timestamp</th>

</tr>

<tr ng-repeat="item in $data">

<td> {{item.deviceId}}</td>

<td> {{item.batteryCharge}}</td>

<td> {{item.batteryDischargeRate}}</td>

<td> {{item.timeStamp}}</td>

</tr>

</table>

</div>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-body">

<center>

<h3> AWS IoT Getting Started <img alt="AWS" height="40" src="http://awsmedia.s3.amazonaws.com/AWS\_Logo\_PoweredBy\_127px.png"> </h3></center>

</div>

</div>

<!-- /container -->

App.js

/\* Enter Device Status Endpoit URL here \*/

var devices\_endpoint\_url = 'https://5g758y71ye.execute-api.ap-southeast-1.amazonaws.com/prod';

// This file covers 3 devices, a real program should be designed to handle any number of devices

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

\* Angular Part of the code

\*/

var appVar = angular.module('ShowMap', ['ngTable']);

var icon = "M21.25,8.375V28h6.5V8.375H21.25zM12.25,28h6.5V4.125h-6.5V28zM3.25,28h6.5V12.625h-6.5V28z";

var icon2 = "M3.5,13.277C3.5,6.22,9.22,0.5,16.276,0.5C23.333,0.5,29.053,6.22,29.053,13.277C29.053,14.54,28.867,15.759,28.526,16.914C26.707,24.271,16.219,32.5,16.219,32.5C16.219,32.5,4.37,23.209,3.673,15.542C3.673,15.542,3.704,15.536,3.704,15.536C3.572,14.804,3.5,14.049,3.5,13.277C3.5,13.277,3.5,13.277,3.5,13.277M16.102,16.123C18.989,16.123,21.329,13.782,21.329,10.895C21.329,8.008,18.989,5.668,16.102,5.668C13.216,5.668,10.876,8.008,10.876,10.895C10.876,13.782,13.216,16.123,16.102,16.123C16.102,16.123,16.102,16.123,16.102,16.123";

var chart\_count = 0; // counts number of chart items

var chart\_count\_max = 80; // This is the threshold when charts start removing items from the left (movig x axis)

appVar.controller('mapController', function ($scope, $http, $timeout, NgTableParams) {

//anammr func was here

$scope.func = function () {

$scope.dataVar = [];

$http({

method: "GET",

url: devices\_endpoint\_url + '/turing',

}).success(function (data) {

var maxlat;

var maxlon;

var maxDeviceId;

var maxtimeStamp = 0;

var hum = JSON.stringify(data);

for (i = 0; i < data.Items.length; i++) {

var lat = JSON.stringify(data.Items[i].payload.location.lat);

var lon = JSON.stringify(data.Items[i].payload.location.lon);

var timeStamp = JSON.stringify(data.Items[i].payload.timeStampEpoch);

var deviceId = JSON.stringify(data.Items[i].payload.deviceId);

if (timeStamp > maxtimeStamp) {

maxlat = lat;

maxlon = lon;

maxDeviceId = deviceId

maxtimeStamp = timeStamp

}

}

console.log("maxlat " + maxlat + " maxtimeStamp " + maxtimeStamp);

$scope.dataVar.push({

"latitude": maxlat,

"longitude": maxlon,

"svgPath": icon,

"color": "#CC0000",

"scale": 0.5,

"label": maxDeviceId,

"labelShiftY": 2,

//"zoomLevel": 5,

"title": "Latitude: " + maxlat + " Longitude: " + maxlon,

"description": maxDeviceId

});

// Code for Get call start

$http({

method: "GET",

url: devices\_endpoint\_url + '/hopper',

}).success(function (data) {

var hum = JSON.stringify(data);

maxtimeStamp = 0;

for (i = 0; i < data.Items.length; i++) {

var lat = JSON.stringify(data.Items[i].payload.location.lat);

var lon = JSON.stringify(data.Items[i].payload.location.lon);

var timeStamp = JSON.stringify(data.Items[i].payload.timeStampEpoch);

var deviceId = JSON.stringify(data.Items[i].payload.deviceId);

if (timeStamp > maxtimeStamp) {

maxlat = lat;

maxlon = lon;

maxDeviceId = deviceId

maxtimeStamp = timeStamp

}

}

console.log("maxlat " + maxlat + " maxtimeStamp " + maxtimeStamp);

$scope.dataVar.push({

"latitude": maxlat,

"longitude": maxlon,

"svgPath": icon,

"color": "#CC0000",

"scale": 0.5,

"label": maxDeviceId,

"labelShiftY": 2,

//"zoomLevel": 5,

"title": "Latitude: " + maxlat + " Longitude: " + maxlon,

"description": maxDeviceId

});

// Code for Get call start

$http({

method: "GET",

url: devices\_endpoint\_url + '/knuth',

}).success(function (data) {

//console.log("data"+JSON.stringify(data));

var hum = JSON.stringify(data);

maxtimeStamp = 0;

for (i = 0; i < data.Items.length; i++) {

var lat = JSON.stringify(data.Items[i].payload.location.lat);

var lon = JSON.stringify(data.Items[i].payload.location.lon);

var deviceId = JSON.stringify(data.Items[i].payload.deviceId);

if (timeStamp > maxtimeStamp) {

maxlat = lat;

maxlon = lon;

maxDeviceId = deviceId

maxtimeStamp = timeStamp

}

}

console.log("maxlat " + maxlat + " maxtimeStamp " + maxtimeStamp);

$scope.dataVar.push({

"latitude": maxlat,

"longitude": maxlon,

"svgPath": icon2,

"color": "#0000ff",

"scale": 0.5,

"label": maxDeviceId,

"labelShiftY": 2,

//"zoomLevel": 5,

"title": "Latitude: " + maxlat + " Longitude: " + maxlon,

"description": maxDeviceId

});

//getMap($scope.dataVar);

}).error(function (error) {

});

//code for get call end

}).error(function (error) {

});

//code for get call end

}).error(function (error) {

});

}

$scope.all\_devices = [];

$scope.hopper\_data = [];

$scope.knuth\_data = [];

$scope.turing\_data = [];

setInterval(function () {

$http.get(devices\_endpoint\_url)

.success(function (data, status, headers, config) {

$scope.table\_data = build\_table\_data(data);

console.log('normalize data')

console.log($scope.table\_data);

var dataset = {};

$scope.tableParams = new NgTableParams({

page: 1,

count: 3

}, {

counts: [],

total: 1,

dataset: $scope.table\_data

});

});

}, 5000);

// Get data ready for chart and map display

function build\_table\_data(response) {

print\_obj = [];

response["Items"].forEach(function (value) {

print\_obj.push(parsePilotLightItem(value));

// add new devices' data to 'all devices' array

var normalize\_data = parsePilotLightItem(value);

// Build array for each device

if (normalize\_data.deviceId == 'hopper') {

$scope.hopper\_data.unshift(normalize\_data);

console.log('--------------> hopper')

console.log($scope.hopper\_data);

if ($scope.hopper\_data.length > 20) {

$scope.hopper\_data.pop();

}

} else if (normalize\_data.deviceId == 'knuth') {

$scope.knuth\_data.unshift(normalize\_data);

console.log('--------------> knuth')

console.log($scope.knuth\_data);

if ($scope.knuth\_data.length > 20) {

$scope.knuth\_data.pop();

}

} else if (normalize\_data.deviceId == 'turing') {

$scope.turing\_data.unshift(normalize\_data);

console.log('--------------> turing')

console.log($scope.turing\_data);

if ($scope.turing\_data.length > 20) {

$scope.turing\_data.pop();

}

} else {

console.log('Error no such device' + normalize\_data.deviceId)

};

$scope.all\_devices.unshift(normalize\_data);

if (($scope.all\_devices.length) >= 20) {

$scope.all\_devices.pop();

$scope.all\_devices.pop();

$scope.all\_devices.pop(); //was shift

}

});

return print\_obj;

};

// parse data - if json format changes this function should change.

function parsePilotLightItem(json\_obj) {

// get the payload json

var payload = json\_obj.payload;

var json\_for\_table = {};

json\_for\_table.deviceId = payload.deviceId;

json\_for\_table.batteryCharge = payload.batteryCharge.toFixed(2);

//json\_for\_table.batteryDischargeRate = payload.batteryDischargeRate.toFixed(2);

json\_for\_table.batteryDischargeRate = Math.round(payload.batteryDischargeRate \* 100) / 100;

json\_for\_table.lon = payload.location.lon;

json\_for\_table.lat = payload.location.lat;

json\_for\_table.timeStampEpoch = payload.timeStampEpoch;

json\_for\_table.sensorReading = payload.sensorReading;

var iso\_time = payload.timeStampIso;

// This is for time display

iso\_time = iso\_time.slice(11, 19);

json\_for\_table.timeStamp = iso\_time;

return json\_for\_table;

};

// Chart section starts here

google.charts.load('current', {

packages: ['corechart', 'line']

},'upcoming', {'packages': ['geochart']});

google.charts.setOnLoadCallback(drawMapAndCharts);

// google chart addition

function drawMapAndCharts() {

var data\_chart\_turing\_battery\_charge = new google.visualization.DataTable();

var data\_chart\_turing\_battery\_Discharge = new google.visualization.DataTable();

var data\_chart\_turing\_sensor\_data = new google.visualization.DataTable();

var data\_chart\_hopper\_battery\_charge = new google.visualization.DataTable();

var data\_chart\_hopper\_battery\_Discharge = new google.visualization.DataTable();

var data\_chart\_hopper\_sensor\_data = new google.visualization.DataTable();

var data\_chart\_knuth\_battery\_charge = new google.visualization.DataTable();

var data\_chart\_knuth\_battery\_Discharge = new google.visualization.DataTable();

var data\_chart\_knuth\_sensor\_data = new google.visualization.DataTable();

// set up GeoChart

var geo\_chart\_data = new google.visualization.DataTable();

geo\_chart\_data.addColumn('number', 'Latitude');

geo\_chart\_data.addColumn('number', 'Longitude');

geo\_chart\_data.addColumn('string', 'Device ID');

geo\_chart\_data.addColumn('number', 'Sensor Reading');

// create dummy points in the "center" of country

geo\_chart\_data.addRows([

[39.14, -98.1, 'hopper', data\_chart\_hopper\_sensor\_data],

[39.14, -98.1, 'knuth', data\_chart\_knuth\_sensor\_data],

[39.14, -98.1, 'turing', data\_chart\_turing\_sensor\_data]

]);

// geo\_chart\_data.removeRows(0,3);

var geo\_chart\_options = {

region: 'US',

displayMode: 'markers',

colorAxis: {colors: ['blue', 'red']}

};

// anamer - changed number to date

data\_chart\_turing\_battery\_charge.addColumn('date', 'X');

data\_chart\_turing\_battery\_charge.addColumn('number', 'battery\_charge');

data\_chart\_turing\_battery\_Discharge.addColumn('date', 'X');

data\_chart\_turing\_battery\_Discharge.addColumn('number', 'battery\_Discharge');

data\_chart\_turing\_sensor\_data.addColumn('date', 'X');

data\_chart\_turing\_sensor\_data.addColumn('number', 'sensor\_data');

// hopper

data\_chart\_hopper\_battery\_charge.addColumn('date', 'X');

data\_chart\_hopper\_battery\_charge.addColumn('number', 'battery\_charge');

data\_chart\_hopper\_battery\_Discharge.addColumn('date', 'X');

data\_chart\_hopper\_battery\_Discharge.addColumn('number', 'battery\_Discharge');

data\_chart\_hopper\_sensor\_data.addColumn('date', 'X');

data\_chart\_hopper\_sensor\_data.addColumn('number', 'sensor\_data');

//knuth

data\_chart\_knuth\_battery\_charge.addColumn('date', 'X');

data\_chart\_knuth\_battery\_charge.addColumn('number', 'battery\_charge');

data\_chart\_knuth\_battery\_Discharge.addColumn('date', 'X');

data\_chart\_knuth\_battery\_Discharge.addColumn('number', 'battery\_Discharge');

data\_chart\_knuth\_sensor\_data.addColumn('date', 'X');

data\_chart\_knuth\_sensor\_data.addColumn('number', 'sensor\_data');

var options\_charge = {

hAxis: {

title: 'Time'

},

vAxis: {},

legend: {

position: 'none'

},

backgroundColor: '#f1f8e9'

};

var options\_Discharge = {

hAxis: {

title: 'Time'

},

vAxis: {},

legend: {

position: 'none'

},

backgroundColor: '#f1f8e9'

};

var options\_SensorData = {

hAxis: {

title: 'Time'

},

vAxis: {},

legend: {

position: 'none'

},

backgroundColor: '#f1f8e9'

};

var chart\_turing\_battery\_charge = new google.visualization.LineChart(document.getElementById('chart\_div\_battery\_charge\_turing'));

chart\_turing\_battery\_charge.draw(data\_chart\_turing\_battery\_charge, options\_charge);

var chart\_div\_battery\_discharge\_turing =

new google.visualization.LineChart(document.getElementById('chart\_div\_battery\_discharge\_turing'));

chart\_div\_battery\_discharge\_turing.draw(data\_chart\_turing\_battery\_Discharge, options\_Discharge);

var chart\_div\_sensor\_data\_turing =

new google.visualization.LineChart(document.getElementById('chart\_div\_sensor\_data\_turing'));

chart\_div\_sensor\_data\_turing.draw(data\_chart\_turing\_sensor\_data, options\_SensorData);

// hopper

var chart\_hopper\_battery\_charge = new google.visualization.LineChart(document.getElementById('chart\_div\_battery\_charge\_hopper'));

chart\_hopper\_battery\_charge.draw(data\_chart\_hopper\_battery\_charge, options\_charge);

var chart\_div\_battery\_discharge\_hopper =

new google.visualization.LineChart(document.getElementById('chart\_div\_battery\_discharge\_hopper'));

chart\_div\_battery\_discharge\_hopper.draw(data\_chart\_hopper\_battery\_Discharge, options\_Discharge);

var chart\_div\_sensor\_data\_hopper =

new google.visualization.LineChart(document.getElementById('chart\_div\_sensor\_data\_hopper'));

chart\_div\_sensor\_data\_hopper.draw(data\_chart\_hopper\_sensor\_data, options\_SensorData);

//knuth

var chart\_knuth\_battery\_charge = new google.visualization.LineChart(document.getElementById('chart\_div\_battery\_charge\_knuth'));

chart\_knuth\_battery\_charge.draw(data\_chart\_knuth\_battery\_charge, options\_charge);

var chart\_div\_battery\_discharge\_knuth =

new google.visualization.LineChart(document.getElementById('chart\_div\_battery\_discharge\_knuth'));

chart\_div\_battery\_discharge\_knuth.draw(data\_chart\_knuth\_battery\_Discharge, options\_Discharge);

var chart\_div\_sensor\_data\_knuth =

new google.visualization.LineChart(document.getElementById('chart\_div\_sensor\_data\_knuth'));

chart\_div\_sensor\_data\_knuth.draw(data\_chart\_knuth\_sensor\_data, options\_SensorData);

// write GeoChart

var geo\_chart = new google.visualization.GeoChart(document.getElementById('mapdiv'));

geo\_chart.draw(geo\_chart\_data, geo\_chart\_options);

// Update dashboard data with new values.

setInterval(function () {

console.log(geo\_chart\_data.getNumberOfRows());

// #### turing #####

var y = Math.round($scope.turing\_data[0].batteryCharge);

var x = new Date($scope.turing\_data[0].timeStampEpoch);

data\_chart\_turing\_battery\_charge.addRows([[x, y]]);

chart\_turing\_battery\_charge.draw(data\_chart\_turing\_battery\_charge, options\_charge);

if (chart\_count > chart\_count\_max) {

data\_chart\_turing\_battery\_charge.removeRow(0); //slide

}

// -- Sensor data

var y = Math.round($scope.turing\_data[0].sensorReading);

var x = new Date($scope.turing\_data[0].timeStampEpoch);

data\_chart\_turing\_sensor\_data.addRows([[x, y]]);

chart\_div\_sensor\_data\_turing.draw(data\_chart\_turing\_sensor\_data, options\_SensorData);

if (chart\_count > chart\_count\_max) {

data\_chart\_turing\_sensor\_data.removeRow(0); //slide

}

var y = $scope.turing\_data[0].batteryDischargeRate;

var x1 = $scope.turing\_data[0].timeStamp;

data\_chart\_turing\_battery\_Discharge.addRows([[x, y]]);

chart\_div\_battery\_discharge\_turing.draw(data\_chart\_turing\_battery\_Discharge, options\_Discharge);

// super hack to update values in place

geo\_chart\_data.setValue(2, 0, $scope.turing\_data[0].lat);

geo\_chart\_data.setValue(2, 1, $scope.turing\_data[0].lon);

geo\_chart\_data.setValue(2, 3, $scope.turing\_data[0].sensorReading);

// geo\_chart\_data.removeRow(0);

if (chart\_count > chart\_count\_max) {

data\_chart\_turing\_battery\_Discharge.removeRow(0); //slide

}

////////// hopper ///////////////////

var y = Math.round($scope.hopper\_data[0].batteryCharge);

var x1 = $scope.hopper\_data[0].timeStamp;

data\_chart\_hopper\_battery\_charge.addRows([[x, y]]);

chart\_hopper\_battery\_charge.draw(data\_chart\_hopper\_battery\_charge, options\_charge);

chart\_count = chart\_count + 1;

if (chart\_count > chart\_count\_max) {

data\_chart\_hopper\_battery\_charge.removeRow(0); //slide

}

// -- Sensor data

var y = Math.round($scope.hopper\_data[0].sensorReading);

var x = new Date($scope.hopper\_data[0].timeStampEpoch);

data\_chart\_hopper\_sensor\_data.addRows([[x, y]]);

chart\_div\_sensor\_data\_hopper.draw(data\_chart\_hopper\_sensor\_data, options\_SensorData);

if (chart\_count > chart\_count\_max) {

data\_chart\_hopper\_sensor\_data.removeRow(0);

}

var y = $scope.hopper\_data[0].batteryDischargeRate;

var x1 = $scope.hopper\_data[0].timeStamp;

data\_chart\_hopper\_battery\_Discharge.addRows([[x, y]]);

// super hack to update values in place

geo\_chart\_data.setValue(0, 0, $scope.hopper\_data[0].lat);

geo\_chart\_data.setValue(0, 1, $scope.hopper\_data[0].lon);

geo\_chart\_data.setValue(0, 3, $scope.hopper\_data[0].sensorReading);

chart\_div\_battery\_discharge\_hopper.draw(data\_chart\_hopper\_battery\_Discharge, options\_Discharge);

if (chart\_count > chart\_count\_max) {

data\_chart\_hopper\_battery\_Discharge.removeRow(0); //slide

}

//

///////////// knuth ////////////////

var y = Math.round($scope.knuth\_data[0].batteryCharge);

var x1 = $scope.knuth\_data[0].timeStamp;

data\_chart\_knuth\_battery\_charge.addRows([[x, y]]);

chart\_knuth\_battery\_charge.draw(data\_chart\_knuth\_battery\_charge, options\_charge);

if (chart\_count > chart\_count\_max) {

data\_chart\_knuth\_battery\_charge.removeRow(0); //slide

}

// -- Sensor data

var y = Math.round($scope.knuth\_data[0].sensorReading);

var x = new Date($scope.knuth\_data[0].timeStampEpoch);

data\_chart\_knuth\_sensor\_data.addRows([[x, y]]);

chart\_div\_sensor\_data\_knuth.draw(data\_chart\_knuth\_sensor\_data, options\_SensorData);

if (chart\_count > chart\_count\_max) {

data\_chart\_knuth\_sensor\_data.removeRow(0); //slide

}

var y = $scope.knuth\_data[0].batteryDischargeRate;

var x1 = $scope.knuth\_data[0].timeStamp;

data\_chart\_knuth\_battery\_Discharge.addRows([[x, y]]);

chart\_div\_battery\_discharge\_knuth.draw(data\_chart\_knuth\_battery\_Discharge, options\_Discharge);

// super hack to update values in place

geo\_chart\_data.setValue(1, 0, $scope.knuth\_data[0].lat);

geo\_chart\_data.setValue(1, 1, $scope.knuth\_data[0].lon);

geo\_chart\_data.setValue(1, 3, $scope.knuth\_data[0].sensorReading);

if (chart\_count > chart\_count\_max) {

data\_chart\_knuth\_battery\_Discharge.removeRow(0); //slide

}

// redraw map

geo\_chart.draw(geo\_chart\_data, geo\_chart\_options);

},

3000);

}

// Allow time for map to read Geolocation

$scope.showSpinkit = true;

$timeout(function () {

$scope.showSpinkit = false;

}, 2000);

// Allow charts to collect data before printing

$scope.showSpinkit\_tables = true;

$timeout(function () {

$scope.showSpinkit\_tables = false;

}, 5000);

// End of controller

});

// geocharts

// Animation directives

angular

.module('ShowMap')

.directive('rdWidgetHeader', rdWidgetTitle);

function rdWidgetTitle() {

var directive = {

requires: '^rdWidget',

scope: {

title: '@',

icon: '@'

},

transclude: true,

template: '<div class="widget-header"><div class="row"><div class="pull-left"><i class="fa" ng-class="icon"></i> {{title}} </div><div class="pull-right col-xs-6 col-sm-4" ng-transclude></div></div></div>',

restrict: 'E'

};

return directive;

};

angular

.module('ShowMap')

.directive('rdWidgetFooter', rdWidgetFooter);

function rdWidgetFooter() {

var directive = {

requires: '^rdWidget',

transclude: true,

template: '<div class="widget-footer" ng-transclude></div>',

restrict: 'E'

};

return directive;

};

/\*\*

\* Widget Body Directive

\*/

angular

.module('ShowMap')

.directive('rdWidgetBody', rdWidgetBody);

function rdWidgetBody() {

var directive = {

requires: '^rdWidget',

scope: {

loading: '@?',

classes: '@?'

},

transclude: true,

template: '<div class="widget-body" ng-class="classes"><rd-loading ng-show="loading"></rd-loading><div ng-hide="loading" class="widget-content" ng-transclude></div></div>',

restrict: 'E'

};

return directive;

};

/\*\*

\* Widget Directive

\*/

angular

.module('ShowMap')

.directive('rdWidget', rdWidget);

function rdWidget() {

var directive = {

transclude: true,

template: '<div class="widget" ng-transclude></div>',

restrict: 'EA'

};

return directive;

function link(scope, element, attrs) {

/\* \*/

}

};

/\*\*

\* Loading Directive

\* @see http://tobiasahlin.com/spinkit/

\*/

angular

.module('ShowMap')

.directive('rdLoading', rdLoading);

function rdLoading() {

var directive = {

restrict: 'AE',

template: '<div class="loading"><div class="double-bounce1"></div><div class="double-bounce2"></div></div>'

};

return directive;

};