Laboratory 11: Android-based IoT interactions with Node-Red/Thingsboard and AWS IoT

Instructor: Young H. Cho T.A: Naveen Vijayan

Due Nov 7 at 11:59 PM (Report), Nov 9 at 11:59 PM (Video)

You will repurpose Android phones with sensors such as GPS, accelerometer, gyro, barometer, light sensor, camera, and microphone as an Internet of Things (IoT). Then you will send the sensor data to a gateway to aggregate, filter, format, and retransmit them using an industry standard protocol to an IoT Hub. The collected data can be further processed and displayed via AWS.

1. Install Termux & Termux API

Download these two required apps from the play store, following are the links.
https://play.google.com/store/apps/details?id=com.termux
https://play.google.com/store/apps/details?id=com.termux.api

Please review https://wiki.termux.com/wiki/Remote_Access for steps to enable ssh access to your phone from your PC after setting a password using passwd. (would be easier to execute commands from your PC)

2. See All the Sensors

Open a new session in Termux; swipe left drawer and click new session to open a new session. Switch between the session from the same left drawer menu. You can discontinue a termux-process running in the current session by pressing phone's **Volume Down + c** in the keyboard.

To get a list of available sensor in the phone, type in Termux:

```
termux-sensor -l
```

This list will vary based on handset model.
Termux will print the value of all the sensors continuously if you type

```
termux-sensor -a
```

This consumes a lot power. To print it once, type in

```
termux-sensor -a -n 1
```

To get all the list of options available, just type in:

```
termux-sensor
```

3. Filter Light Sensor

Light sensor is typically placed beside front camera (selfie camera) along with proximity sensor, and pretty much available in all handset.
Take the light sensor data to showcase how to extract and pipe this data. To filter out the light sensor in termux-sensor, find the name from the list of printed sensor, in my case I picked **"TMD3702_Light Ambient Light Sensor Non-wakeup"** in my oneplus. To get a filtered data with this sensor, type in the following to see its data.

```
termux-sensor -s "TMD3702_Light Ambient Light Sensor Non-wakeup" <sensor name>
```

4. **AWS IoT Setup** (use free tier account)

In the AWS IoT Core console, create a new policy in the “Policies” tab under “Secure”.

In the “Action field, enter: **"iot:*"** for all IoT actions, and under the “Resource ARN” field, enter **"*"** for all devices and topics. Check “Allow” for the “Effect” field.

Back in the AWS IoT Core console, create a new device in the “Things” tab under “Manage”. Name your device, create a new type, and a new group for your IoT device.(create a single thing)

Select “One-click certificate creation” and download your certificate, public, and private keys for your device. Make sure to click on “Activate” for your root certificate and click on the download link. This will not actually download anything and will redirect you to a download page with several root certificates. Download either Amazon Root CA 1 or CA 3. Lastly, attach the policy you created in previous step to your device.

5. **Node-red setup in PC** (acting as aggregator)

Install node-red on your PC by following steps from:
https://nodered.org/docs/getting-started/windows (if you have windows)
If any other OS search in google for corresponding step. Installing OS on VM should also work if you set n/w adapter in bridged mode)

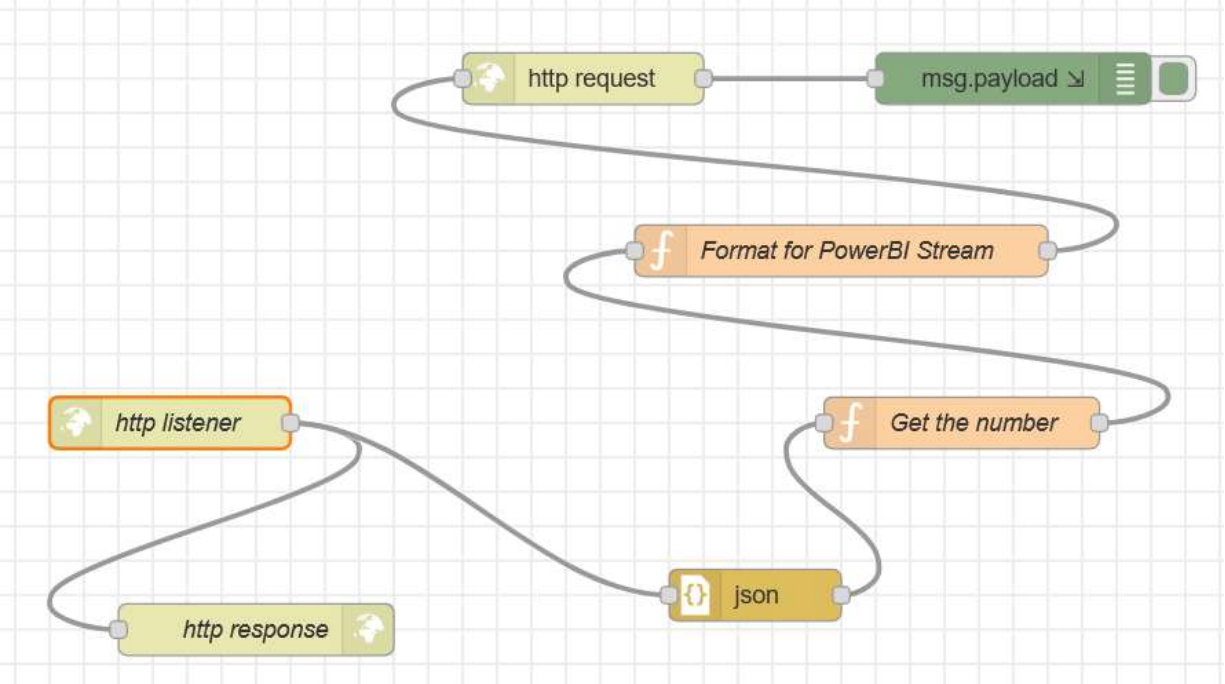
After this open 127.0.0.1:1880 on browser.

To import any external flow into Node-RED, click on the Hamburger Menu on the Top Right, then click **Import > Clipboard,** you'll get an import dialogue box, then copy paste the below JSON content into the box and hit **Import**.


```
[{"id":"c12e9e18.651ae","type":"tab","label":"Flow 3","disabled":false,"info":""},{"id":"dda5983.8467f68","type":"json","z":"c12e9e18.651ae","name":"","property":"payload","action":"","pretty":false,"x":490,"y":360,"wires":[["3fed63c8.f80ccc"]]},{"id":"3fed63c8.f80ccc","type":"function","z":"c12e9e18.651ae","name":"Get the number","func":"msg.payload.light_sensor = msg.payload[₩"TMD3702 Light Ambient Light Sensor Non-wakeup₩"][₩"values₩"][0];₩nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":610,"y":260,"wires":[["a8cd33f3.1c049"]]},{"id":"6d1fabef.8f4bf4","type":"http request","z":"c12e9e18.651ae","name":"","method":"POST","ret":"txt","paytoqs":"ignore","url":"a3s67bg7sesm5q-ats.iot.us-west-2.amazonaws.com:8443/topics/lab11?qos=1","tls":"eea8ed50.ce04e","persist":false,"proxy":"","authType":"","x":390,"y":60,"wires":[["e7c0faa5.356a38"]]},{"id":"a8cd33f3.1c049","type":"function","z":"c12e9e18.651ae","name":"Format for PowerBI Stream","func":"msg.payload = JSON.stringify([{₩"light_sensor₩":msg.payload.light_sensor}]);₩nreturn msg;","outputs":1,"noerr":0,"x":540,"y":160,"wires":[["6d1fabef.8f4bf4"]]},{"id":"e7c0faa5.356a38","type":"debug","z":"c12e9e18.651ae","name":"","active":true,"tosidebar":true,"console":true,"tostatus":false,"complete":"payload","x":640,"y":60,"wires":[]},{"id":"585e7e79.cd3a88","type":"http in","z":"c12e9e18.651ae","name":"http listener","url":"/iot","method":"post","upload":false,"swaggerDoc":"","x":150,"y":260,"wires":[["eb40a42e.13afd8","dda5983.8467f68"]]},{"id":"eb40a42e.13afd8","type":"http response","z":"c12e9e18.651ae","name":"http response","statusCode":"202","headers":{},"x":200,"y":380,"wires":[]},{"id":"eea8ed50.ce04e","type":"tls-config","name":"","cert":"","key":"","ca":"","certname":"9fd0757b11-certificate.pem.crt.txt","keyname":"9fd0757b11-private.pem.key","caname":"root CA.pem","servername":"","verifyservercert":false}]
```


Node-RED Flow Import Process

Since the sensor command output’s, a value with a single number, the rest of the flow nodes are designed to handle that single value; if you call for any sensor that return multiple values, the later nodes will break. You will also have to modify the **"Get the number"** node to the name of your light sensor.



Here a “http listener” node is listening at port 1880 (node-red port) for post events which would come from mobile on /iot and an “http response” node is present to sent a response back to mobile http request as needed by the http protocol.

Go back into AWS IoT Core and copy the endpoint URL found in the “Settings” tab.

In the **HTTP request** node, set method to "Post" and paste the endpoint URL that we got from AWS IoT. Append the following text in yellow to your URL so it looks like the one shown below.

XXXX-ats.iot.us-east-1.amazonaws.com:8443/topics/myTopic?qos=1

You can change “myTopic” in the URL to any topic name of your choosing.

Check the box for “Enable secure SSL/TLS connection” and leave it as a TLS connection. Click on the pencil to open the “Properties” menu.

Upload the root certificate, private key, and certificate files for your device in the fields listed (got form previous iot step). Save your changes to the node and exit the configuration menu. (certificate.pem.crt.txt -> for as certificate, private.pem.key -> for private key, downloaded Root CA1 or 3 certificates for CA certificate). Also uncheck “verify server certificate”.

Click on the **Deploy** button on the upper right corner of Node-RED flows window.

6. Post Data from the Phone to PC

There are many different ways to send data from Termux to the gateway. Please go through two methods described below.

a. Install Node-RED & Termux API Module on Termux

Now open the Termux App and run the following commands to install Node-RED

```
apt update
apt upgrade
apt install termux-api
apt install python
apt install coreutils nodejs
npm i -g --unsafe-perm node-red
```

If you are running the latest version of Android you might face an error on the **“npm I -g --unsafe-perm node-red”** line saying **“cannot read property ‘length’ of undefined”**. Run the following commands if you face them.

```
apt-get install yarn
yarn global add npm
```

Now launch Node-RED

```
node-red
```

Node-RED is now running on localhost on the Android phone; it will be showing the IP address and port it is running on in Termux.

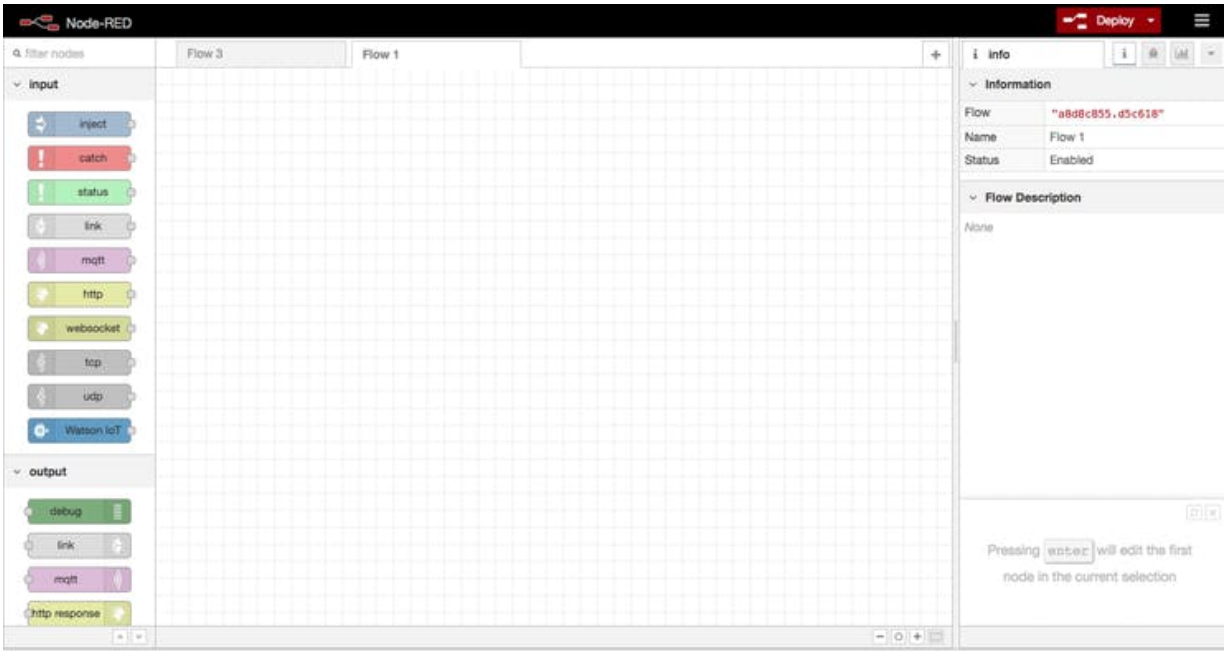
On your phone you can point a browser to **localhost:1880** and it will open up Node-RED flows editor. But for practical purposes you'll have to open it up on a bigger screen such as a laptop.
To access the flows editor on a browser from a laptop running on the same network as the phone, you can do it in two ways.

One, your laptop/desktop has to be connected to the same WiFi router as the phone is connected to (preferred way as you need internet access). Two, you can run a WiFi hotspot from your phone and connect to the hotspot from your laptop.
Now type in ifconfig in Termux and get your phone's IP address. (you might need to open new session by swiping from left to right on screen as node-red should always run in other terminal)

```
$ ifconfig
Warning: cannot open /proc/net/dev (Permission denied). Limited output.
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1  (U
NSPEC)

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.100  netmask 255.255.255.0  broadcast 192.168.0.255
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 3000
 (UNSPEC)
```

Add a **colon** and port number of “**1880**” to the ip address and enter it into a browser from the laptop/desktop. In my case it was 192.168.0.100:1880



Node-RED Blank Flow

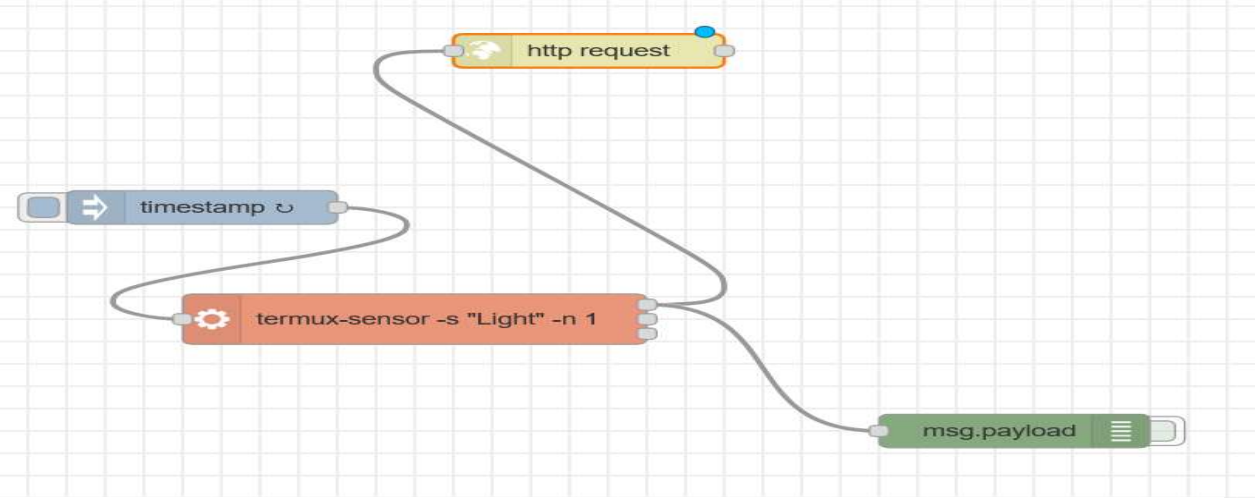
Use Node-RED's "exec" node to run a terminal command and get the sensor data into Node-RED's flow editor and post it to PC.

To import any external flow into Node-RED, click on the Hamburger Menu on the Top Right, then click **Import > Clipboard,** you'll get an import dialogue box, then copy paste the following JSON content into the box and hit **Import**.

```
[{"id":"c12e9e18.651ae","type":"tab","label":"Flow 3","disabled":false,"info":""},{"id":"bec0b05.54ba45","type":"inject","z":"c12e9e18.651ae","name":"","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"5","crontab":"","once":true,"onceDelay":"","topic":"","payload":"","payloadType":"date","x":170,"y":200,"wires":[["a9c3b25c.f6cc1"]]},{"id":"a9c3b25c.f6cc1","type":"exec","z":"c12e9e18.651ae","command":"termux-sensor -s ₩"Light₩" -n 1","addpay":true,"append":"","useSpawn":"false","timer":"","oldrc":false,"name":"","x":280,"y":300,"wires":[["6d1fabef.8f4bf4","a7281c9a.b89b6"],[],[]]},{"id":"6d1fabef.8f4bf4","type":"http request","z":"c12e9e18.651ae","name":"","method":"POST","ret":"txt","paytoqs":"ignore","url":"http://192.168.0.102:1880/iot","tls":"","persist":false,"proxy":"","authType":"","x":370,"y":60,"wires":[[]]},{"id":"a7281c9a.b89b6","type":"debug","z":"c12e9e18.651ae","name":"","active":false,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","statusVal":"","statusType":"auto","x":590,"y":400,"wires":[]}]
```

Node-RED Flow Import Process

Now, click on **exec** node and put in **termux-sensor -s "Light Sensor Name"-n 1,** make sure to put in your light sensor name. You would also need to change “http request” node to your corresponding IP of the PC and the path on which you are listening (port would be same as that of node-red port). Click **Deploy** on mobile node-red. Here “timestamp” node is responsible for the periodic execution of the commands present under the exec node.



Here mobile is sending an http post request to the http listener of the PC. The path where I have http listener running is http://192.168.0.102:1880/iot since I have configured this as path of listener at PC. The IP "192.168.0.102" is the PC IP in private IP subnet to which mobile also belongs.

On the PC node-red you should begin to see messages in the following format in the debug window.

```
"{"message":"OK","traceId":"XXXX"}"
```

If you see "OK" as shown above, your packet was successfully sent to AWS IoT Core.

In AWS IoT Core, go into the "Test" tab. Subscribe to your topic using your chosen "topic name" and verify that IoT Core is reading your packets correctly.

b. CURL HTTP POST Method

```
#!/bin/bash
while [ 1 ]
do
termux-sensor -s "TMD3702_Light Ambient Light Sensor Non-wakeup" -n 1 > oo
curl -X POST --data-binary @oo 192.168.0.102:1880/iot -H "Content-Type: text/plain"
sleep 5
done
```

Here first the sensor is read and redirect its output to a file and then curl read data from this file and send this data to PC node-red instance.

You might also want to disable sleep in your phone as termux will be in suspended state when mobile goes for sleep to conserve power. So, sensor data would not be sent during this period.

Explore more functionality supported in node-red from the left panel. The different apis supported etc. Also learn how to code in node-js which you can write within a function block. Also explore output/input relations of a block and how different blocks can be inter-connected using edges which have multiple o/p and i/p.

7. Visualize Data in AWS IoT

Install ThingsBoard CE on an EC2 instance following the instructions in the link below. Use ubuntu server 16.04 x86 OS. (use t2.micro instance type). In security group enable port 8080 access. Use in-memory queue service (or if you want you can explore kafka). Use tenant@thingsboard.org for logging in.

https://thingsboard.io/docs/user-guide/install/ubuntu/

Send your data stream to your instance and create a dashboard in ThingsBoard. This dashboard must display a real-time graph of your light sensor value on the y-axis and time on the x-axis. Tutorials on using ThingsBoard are in the following link.

https://thingsboard.io/docs/guides/

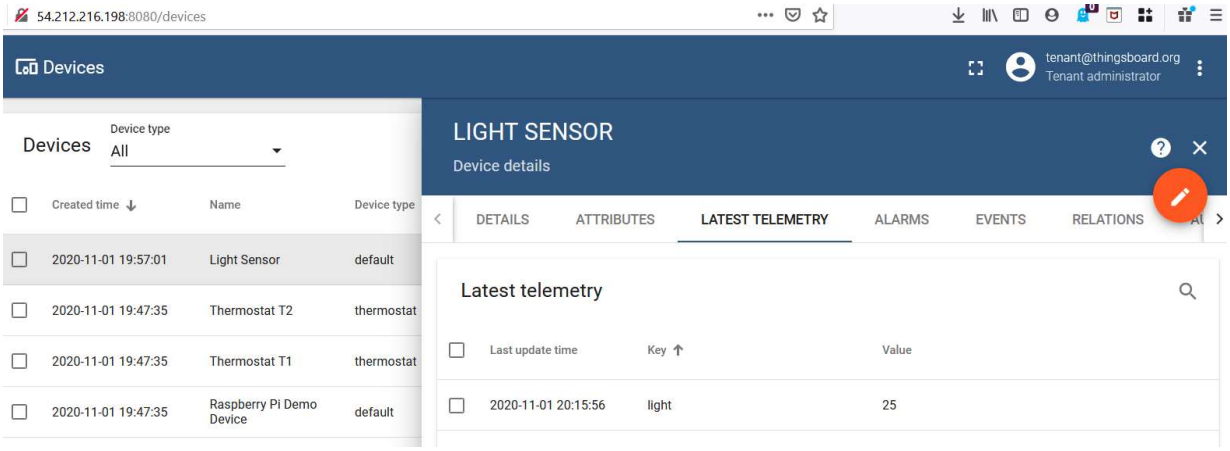
Note that there are multiple ways you may route your stream to your instance. Some possible methods include modifying your Node-RED flow or republishing your stream from AWS IoT. You are free to achieve this in anyway you like, with any combination of AWS tools available.

Here I will concentrate on using http method for posting data to thingsboard.
https://thingsboard.io/docs/reference/http-api/

- Login with tenant@thingsboard.org and password as tenant.
- Go to devices tab and rename TEST Device A1 as Temperature. This can be done by clicking it and selecting orange button. Further explore all the tabs present in thingsboard and its relations. You can google for further information. You can create new device also if you wish here.
- Go to devices tab and create new device with name as “light sensor”. This can be done by clicking on “+” at right most corner. Further explore all the tabs present in thingsboard and its relations. You can google for further information.
- In the device tab now open newly created “light sensor”. Go to details and copy the access token. This would be needed at PC node-red to publish data to thingsboard.
- Follow steps in https://thingsboard.io/docs/getting-started-guides/helloworld/ to push data to thingboard by using curl. (below command execute from cmd on windows)

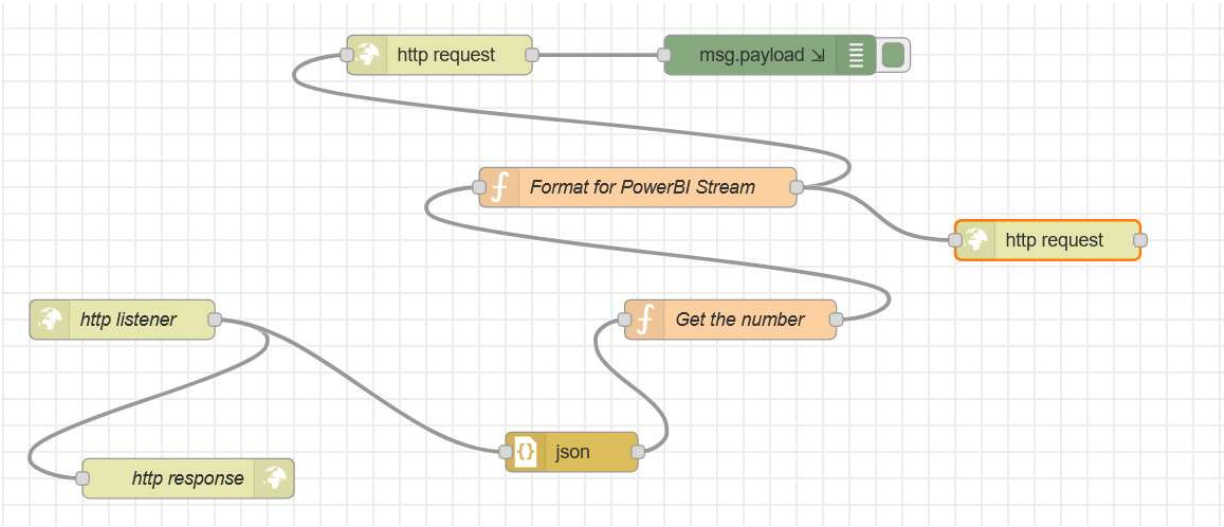
  curl -v -X POST -d "{₩"light₩": 25 }” http://<ec2 public ip>:8080/api/v1/$ACCESS_TOKEN/telemetry --header "Content-Type:application/json"

- Observe the data getting posted from curl is visible in “**Latest telemetry tab”** under **specific device**. Based on your requirement you can create new devices and repeat the same procedure.



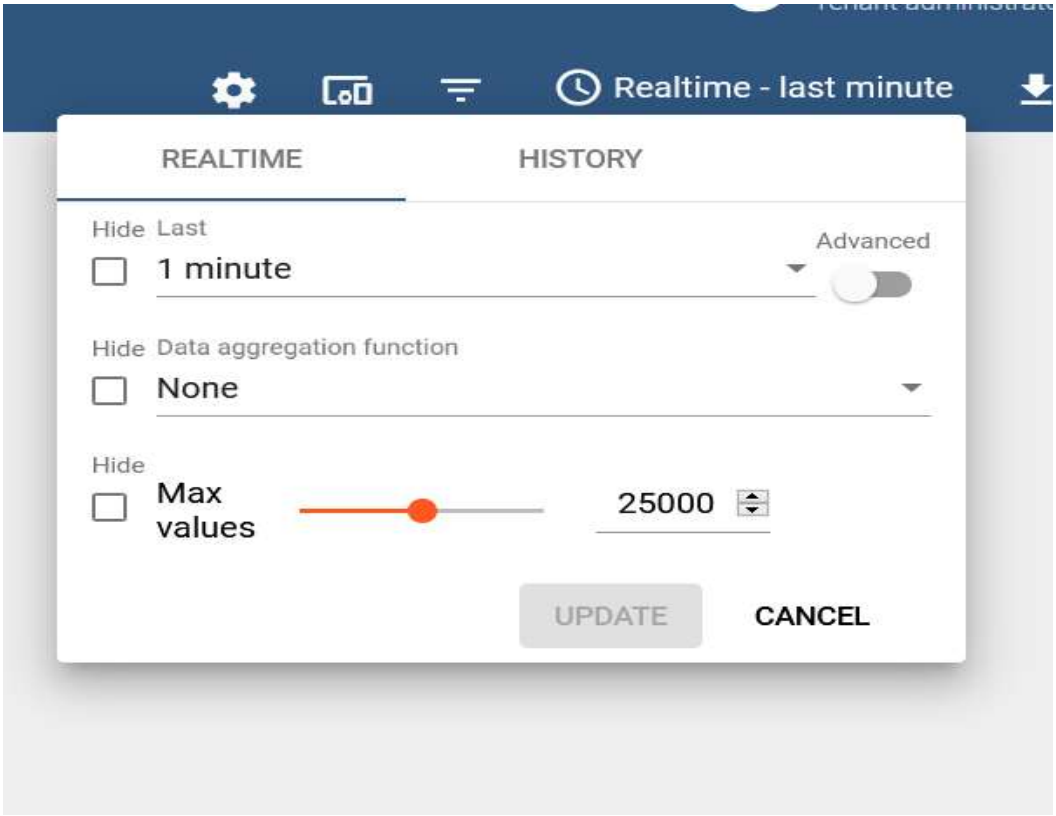
- Now add one more http request node with post method to send data into thingsboard. You should give http link which you have given for the curl command. Don't enable SSL.

  http://<ec2 public ip>:8080/api/v1/<access token>/telemetry



- Go to latest telemetry tab and select your sensor key. Click on show on widget and select **charts**. You can select any widget for your display based on your requirements.
- Select Time series float chart and press add to dashboard.
- Choose create new dashboard and give it a name.
- Go to dashboard window on left column and you should see your newly created dashboard. Now under the dashboard you have created (by clicking it) you can create multiple widget you want and then map to telemetry data you want to for your devices. (explore all the options present under dashboard).

- When PC node-red is successfully sending data, you can observe it getting plotted in your widget. The plot time period can be configured. You can select display time window in data tab and select your time period in time window option.



8. Redirecting stream data to S3 bucket

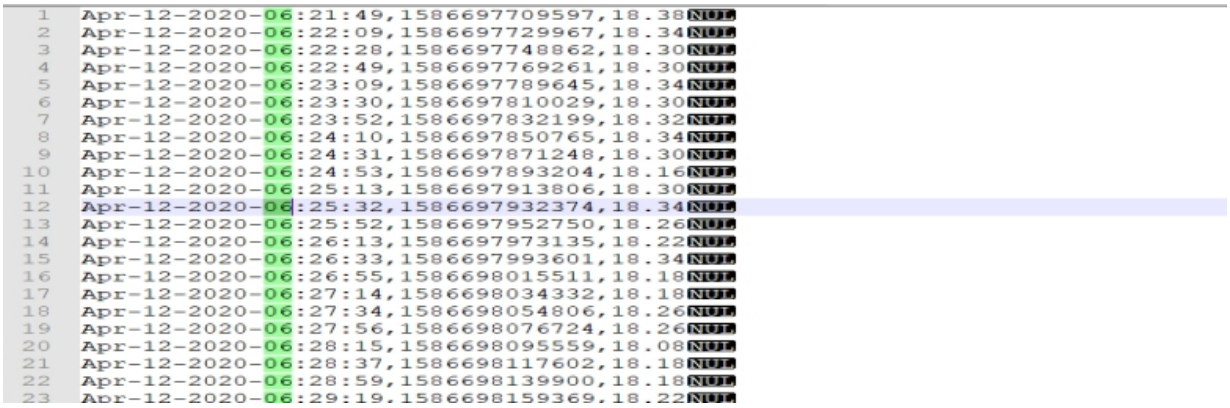
- Create a new S3 bucket. Give bucket name and grant appropriate access permission (don't give public permission) so that **kinesis firehose stream** agent can write files to this bucket. Click "create bucket" on rightmost corner down
- select kinesis (under analytics) from aws dashboard. Create a "*Kinesis Firehose delivery streams*" **by** Selecting **create delivery stream**. Enter a stream name. Select *source* as Direct PUT or other sources. Click **next** in bottom right corner. (explore what does kinesis data stream option does).

- Click next to skip step2 (of process records, you can transform your input stream here by using lambda function). Select Amazon S3 as your destination in step 3. Select your newly created S3 bucket under the S3 destination. If you want, you can add a S3 prefix but is optional. You can even redirect data to other sources here as per your use-case.
- Click next and give buffer size as 5 mb and buffer interval as 120 seconds. This can be given as per your application requirements. **Select IAM role** under permissions (your role should have permission to write to your corresponding s3 bucket). Keep all other configurations as default. Choose "create or update IAM role".
- Select next and click create delivery stream on rightmost bottom corner. Now your iot streaming data will be stored at S3 bucket in a period of 120 seconds. Write to S3 will happen from kinesis at 120 second interval or 5 MB buffer is exhausted.
- Now again go to "**Iot Core"** and select "**Act"** and then "**Rules"** tab and create a new rule. Click create button .
- Under Rule query statement you should write a query to fetch the data from iot.Write below query

  *SELECT * FROM 'iot endpoint where you post from PC'*
- Click "Add action" and select "send a message to Amazon Kinesis Firehose stream". Click **configure action** on the bottom most right corner. (explore all other supported services apart from firehose stream)
- Select your newly created firehose stream under stream name which was created in previous step and give separator as "₩n".
- In "**Choose or create a role to grant AWS IoT access to perform this action**" choose an existing policy if you have or create a new policy from here. Click "Add action".
- Now select **create rule** in new page under bottom most right corner.
- Now your data from PC should get stored at s3 bucket. For 120 seconds it gets buffered in kinesis and then at 120th second it writes to S3 bucket.
- You would also have to enable the stream under "rules" window.
- To verify that your iot device is posting data, go to **IOT core** window, then to **Test** tab and give **subscription topic** as same as your post id from PC. Make note of your QOS field value also with which you are posting. This should enable you to see the data from PC is being received at aws or not.

9. Further Tasks Which You Must Implement:

1. Add a timestamp in unix format and date in human readable format to your sensor value as shown below i.e date + time, unix timestamp, sensor value. In S3 your data should be written in these formats. The timestamp and date should be inserted from PC node-red.



2. You should take data from at least 5 different sensors of your phone and should also take reading from more than one phone which is sending data to your node-red running on the PC. Display these multiple sensor readings under the dashboard in thingsboard by plotting graph. Moreover, your thingsboard should use the **timestamp** pushed from the PC node-red and should not use thingsboard current system timestamp when data is received.
3. Also write a lambda function which writes your sensor data to a database (like DynamoDB) with all the columns and data from multiple different sensors (date, timestamp, sensor 1 reading, sensor 2 reading etc.).
4. Also write a lambda function which at every 240 seconds (i.e at 2 firehose stream push) should write a single file to S3 bucket which contains all data upto this point (in normal case 2 files would be present at S3). You should keep on appending data to this file so that all the data could be retrieved from this single file.
5. Moreover, At every 240 seconds you should find the median, mean and standard deviation of your readings and group you data under median+1 standard deviation, median-1 standard deviation, median+2 standard deviation, median-2 standard deviation, median+3 standard deviation, median-3 standard deviation. You should be using all the sensor values up to that point for calculation. For example, for first time it would be all values up to 240 second, then all value upto 480 seconds from start etc. You should be writing this to a single separate file and you would keep on appending new data with grouping along with median, mean, standard deviation to this single file with timestamp marked as separation like 240, 480 etc. (format is given below). Write an efficient algorithm.

```
data till 240:
median : 10
mean: 15
standard deviation: 5
median +- 1 sd
<sensor data> (same format as would be present in normal S3 file)
median +- 2 sd
<sensor data> (same format as would be present in normal S3 file)
median +- 3 sd
<sensor data> (same format as would be present in normal S3 file)
================================================================
data till 480:
median : 20
mean: 25
standard deviation: 4
median +- 1 sd
<sensor data> (same format as would be present in normal S3 file)
median +- 2 sd
<sensor data> (same format as would be present in normal S3 file)
median +- 3 sd
<sensor data> (same format as would be present in normal S3 file)
================================================================
................
```

QUESTION

How would you integrate an ML model into this lab which trains on collected sensor data. How would you deploy the model and make prediction?. How will you implement such a functionality in AWS?. Explain in detail how would the **architecture look like** and what all **components** you would be needing from AWS with **message sequence** interconnecting each stage. In other words, how would you pass the sensor input to ML model to train it and make prediction on continuous streaming data (as you would have to train and predict on data continuously) and how would the ML predicted output flow to your mobile to pop up some message or trigger an event?. Explain in detail. (you can draw a design of such system and explain it).

REFERENCES

https://nodered.org/docs/platforms/android

https://nodered.org/docs/node-red-admin

https://github.com/termux/termux-packages/issues/1855

https://www.hackster.io/mainul/build-an-iot-viz-with-your-android-phone-s-sensors-cc241e

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.Lambda.Tutorial.html