Sprint 2

Aim:

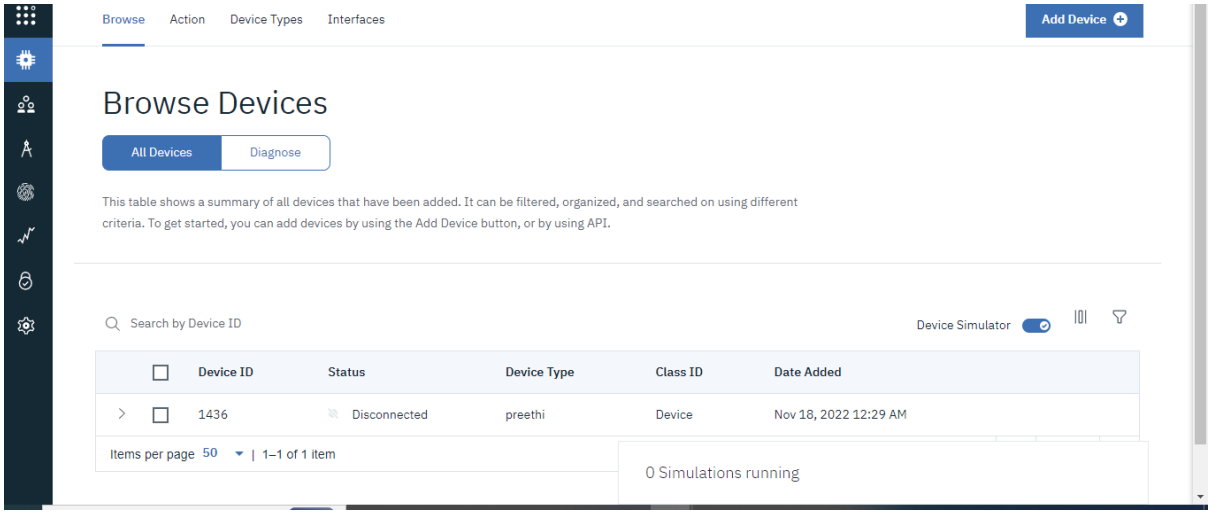
To create python script and establish Node-RED

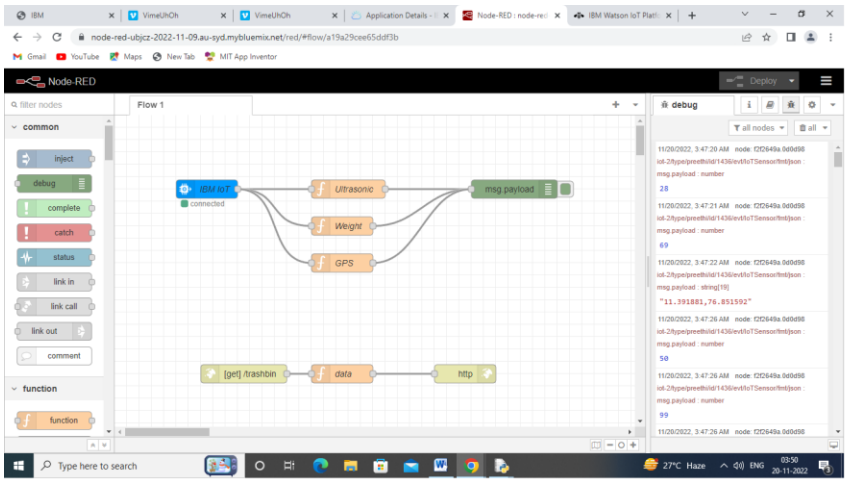
Requirements:

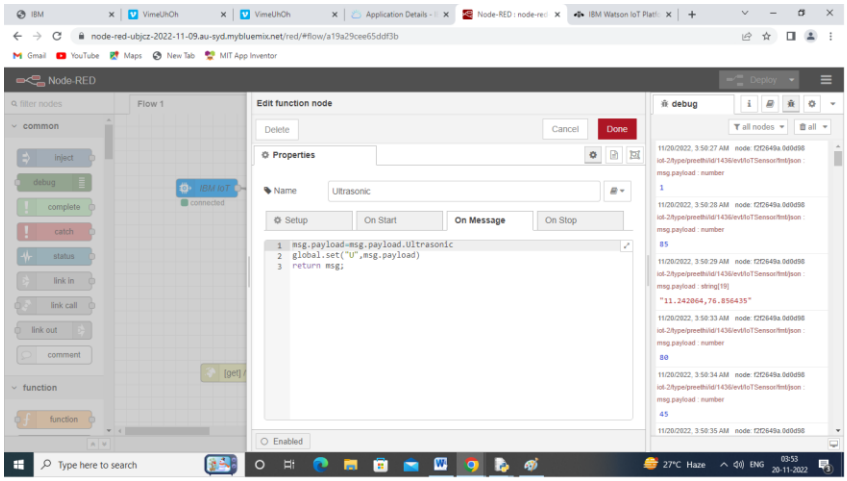
- Python 3.7
- IBM Cloud Account (IoT platform and Node RED App)

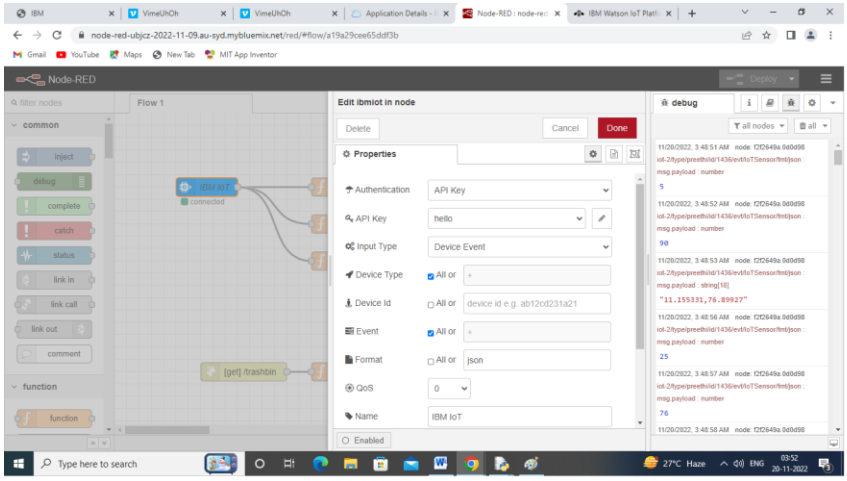
Activities:

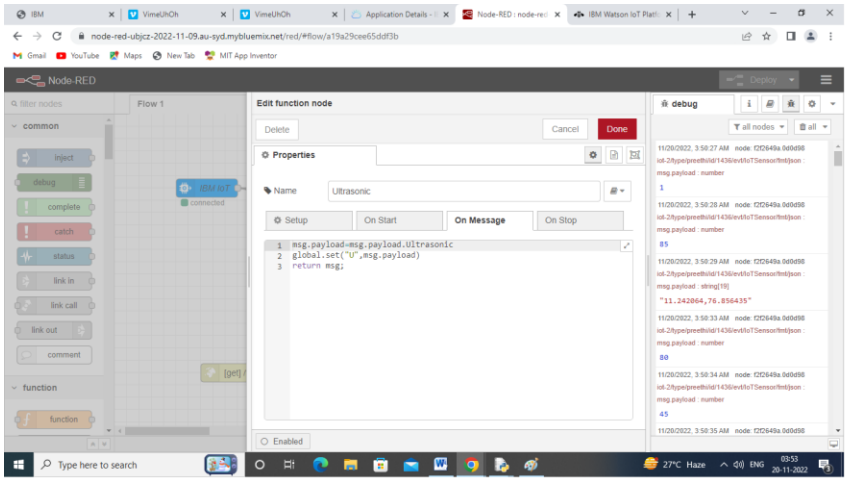
- Make sure IBM Watson IoT Platform and Node RED App is created in cloud account
- In IBM Watson IoT platform create device and cards
- Develop python script for generating trash bin parameters like weight of the bin and location and GPS
- In Node RED App ,install necessary node
- To retrieve data from IoT Platform use IBMIoT node from node column on left and once double click select the necessary properties
- Place function nodes to get the values ,then type code and connect them to IBMIot node
- Place debug node and connect them to the function nodes to view the data in debug

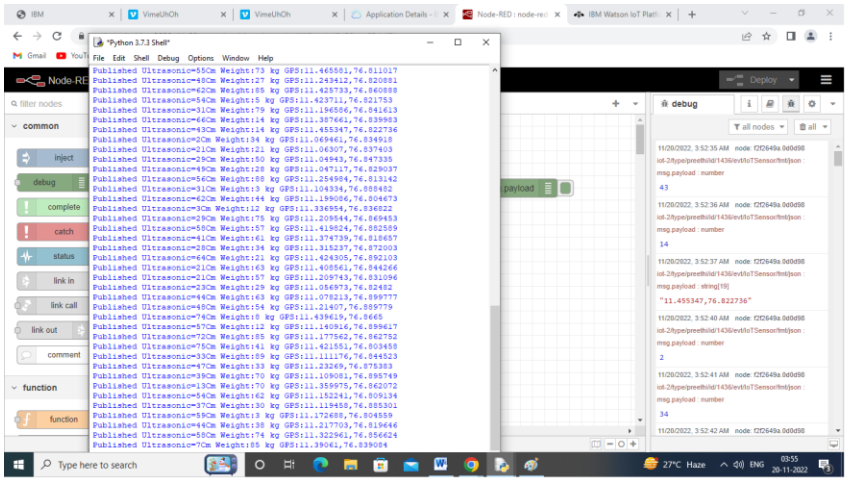

Browse
Action
Device Types
Interfaces
Add Device
Browse Devices
All Devices
Diagnose
This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.
Search by Device ID
Device Simulator
Device ID
Status
Device Type
Class ID
Date Added
1436
Disconnected
preethi
Device
Nov 18, 2022 12:29 AM
Items per page 50
1–1 of 1 item
0 Simulations running


Node-RED
Flow 1
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
IBM IoT
connected
Ultrasonic
Weight
GPS
msg.payload
[get] /trashbin
data
http
debug


Node-RED
Edit function node
Delete
Cancel
Done
Properties
Name
Ultrasonic
Setup
On Start
On Message
On Stop
msg.payload=msg.payload.Ultrasonic
global.set("U",msg.payload)
return msg;
Enabled
debug
Flow 1
common
function
Type here to search


Node-RED
Edit ibmiot in node
Delete
Cancel
Done
Properties
Authentication
API Key
API Key
hello
Input Type
Device Event
Device Type
Device Id
Event
Format
json
QoS
Name
IBM IoT
Enabled
debug
Flow 1
common
function
Type here to search





Python script

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
```

```
organization="7lnn7p"
devicetype="preethi"
deviceid="1436"
authMethod="token"
authToken="09876543211"

def myCommandCallback(cmd):
    print("Command received:%s"%cmd.data['command'])
    status=cmd.data['command']
    if status =="lighton":
        print("led in on")
    else:
            print("led is off")
try:

deviceOptions={"org":organization,"type":devicetype,"id":deviceid,"
auth-method":authMethod,"auth-token":authToken}
    deviceCli=ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Caught exception connecting device:%s"%str(e))
    sys.exit()
deviceCli.connect()
```

```
while True:
    time.sleep(5)
    Ultrasonic=random.randint(0,80)
    Weight=random.randint(0,100)
    lat=round(random.uniform(11.03,11.50),6)
    long=round(random.uniform(76.80,76.90),6)
    GPS=str(lat)+str(',')+str(long)
    myData={'Ultrasonic':Ultrasonic,'Weight':Weight,'GPS':GPS}
    def myOnpublishCallback():
        print("Published Ultrasonic=%sCm"%Ultrasonic,"Weight:%s kg"%Weight,"GPS:%s"%GPS)


success=deviceCli.publishEvent("IoTSensor","json",data=myData,qos=0,on_publish=myOnpublishCallback)
    if not success:
        print("Not connected to IoTF")
    time.sleep(1)
    deviceCli.commandCallback=myCommandCallback
deviceCli.disconnect()
```