Sprint 02

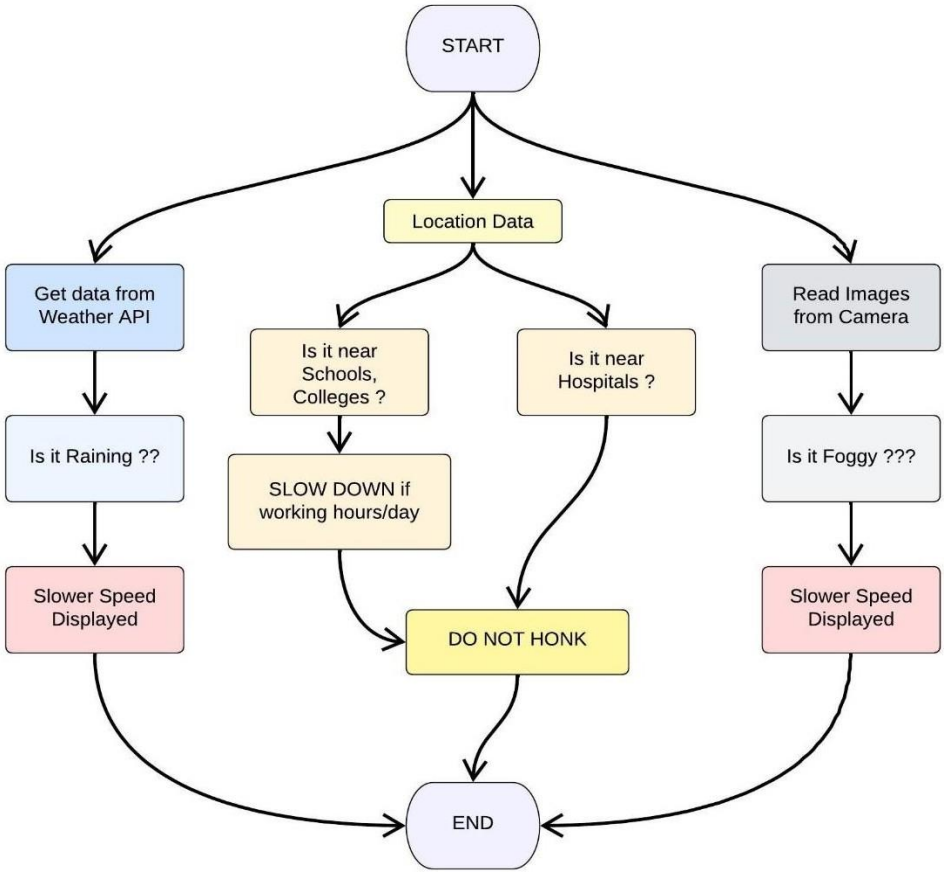
Signs with Smart Connectivity for Better Road Safety

Team ID - PNT2022TMID47020

Sprint Goals :

1. Push data from local code to cloud

Code Flow :



Program Code :

This file is a utility function that fetches the weather from OpenWeatherAPI. It returns only certain required parameters of the API response.

Python code

```
import requests as reqs


def get(myLocation,APIKEY):
    apiURL =
f"https://api.openweathermap.org/data/2.5/weather?q={myLocation}&appid={APIKEY}"
    responseJSON = (reqs.get(apiURL)).json()
    returnObject = {
        "temperature" : responseJSON['main']['temp'] - 273.15,
        "weather" : [responseJSON['weather'][_]['main'].lower() for _ in
range(len(responseJSON['weather']))],
        "visibility"  : responseJSON['visibility']/100, # visibility  in percentage where 10km is 100% and
0km is 0%
    }
    if("rain" in responseJSON):
        returnObject["rain"] = [responseJSON["rain"][key] for key in responseJSON["rain"]]
    return(returnObject)
```

> publishData.py

This code pushes data to the cloud and logs data. IBM Cloud is configured such that the data is displayed in the following website: CLICK TO OPEN NODE RED DASHBOARD

```
# Python code


# IMPORT SECTION STARTS


import wiotp.sdk.device # python -m pip install wiotp
import time


# IMPORT SECTION ENDS
# ---------------------------------------------
# API CONFIG SECTION STARTS


myConfig = {
    "identity"  : {
        "orgId" : "f59trs",
```

```
        "typeId" : "testdevice",
        "deviceId" : "device1"
    },
    "auth" : {
        "token" : "Jrwa7c8Os2Zpq)WW18"
    }
}

# API CONFIG SECTION ENDS
# ----------------------------------------------
# FUNCTIONS SECTION STARTS

def myCommandCallback(cmd):
    print("recieved cmd : ",cmd)



def logData2Cloud(location,temperature,visibility):
    client = wiotp.sdk.device.DeviceClient(config=myConfig,logHandlers=None)
    client.connect()
    client.publishEvent(eventId="status",msgFormat="json",data={
        "temperature" : temperature,
        "visibility"  : visibility,
        "location"  : location
    },qos=0,onPublish=None)
    client.commandCallback = myCommandCallback
    client.disconnect()
    time.sleep(1)
```

 # FUNCTIONS SECTION ENDS

This file is a utility function that returns only essential information to be displayed at the hardware side and abstracts all the unnecessary details. This is where the code flow logic is implemented.

Python code

```
# IMPORT SECTION STARTS

import weather
from datetime import datetime as dt
from publishData import logData2Cloud as log2cloud

# IMPORT SECTION ENDS
# ---------------------------------------------
# UTILITY LOGIC SECTION STARTS
def processConditions(myLocation,APIKEY,localityInfo):
    weatherData = weather.get(myLocation,APIKEY)

    log2cloud(myLocation,weatherData["temperature"],weatherData["visibility"])

    finalSpeed = localityInfo["usualSpeedLimit"] if "rain" not in weatherData else
localityInfo["usualSpeedLimit"]/2
    finalSpeed = finalSpeed if weatherData["visibility"]>35 else finalSpeed/2

    if(localityInfo["hospitalsNearby"]):
        # hospital zone
        doNotHonk = True
    else:
        if(localityInfo["schools"]["schoolZone"]==False):
            # neither school nor hospital zone
            doNotHonk = False
        else:
            # school zone
            now = [dt.now().hour,dt.now().minute]
            activeTime = [list(map(int,_.split(":"))) for _ in localityInfo["schools"]["activeTime"]]
            doNotHonk = activeTime[0][0]<=now[0]<=activeTime[1][0] and
activeTime[0][1]<=now[1]<=activeTime[1][1]
```

```
    return({
        "speed" : finalSpeed,
        "doNotHonk" : doNotHonk
    })
```

UTILITY LOGIC SECTION ENDS

The code that runs in a forever loop in the micro-controller. This calls all the util functions from other python files and based on the return value transduces changes in the output hardware display.

```
# Python code

# IMPORT SECTION STARTS

import brain

# IMPORT SECTION ENDS
# ---------------------------------------------
# USER INPUT SECTION STARTS

myLocation = "Chennai,IN"
APIKEY = "9cd610e5fd400c74212074c7ace0d62c"

localityInfo = {
    "schools" : {
        "schoolZone" : True,
        "activeTime" : ["7:00","17:30"]  # schools active from 7 AM till 5:30 PM
        },
    "hospitalsNearby" : False,
    "usualSpeedLimit" : 40 # in km/hr
}

# USER INPUT SECTION ENDS
#---------------------------------------------
```

MICRO-CONTROLLER CODE STARTS

```
while True :
    print(brain.processConditions(myLocation,APIKEY,localityInfo))


'''
MICRO CONTROLLER CODE WILL BE ADDED IN SPRINT 3 AS PER OUR PLANNED SPRINT SCHEDULE
'''
```

MICRO-CONTROLLER CODE ENDS

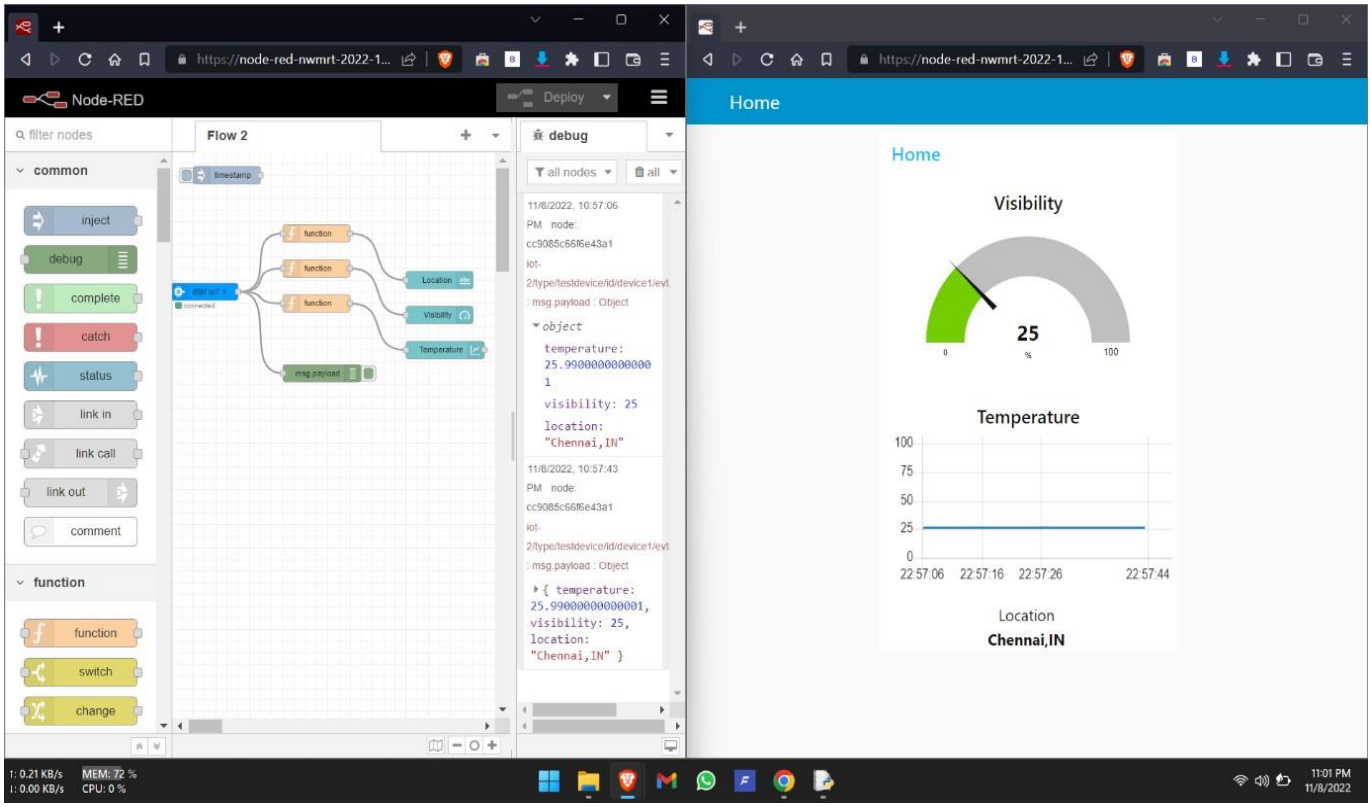
Output :

LINK TO NODE RED DASHBOARD

Code Output

```
22022-11-08 22:57:43,506 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:f59trs:testdevice:device1

2022-11-08 22:57:43,574 wiotp.sdk.device.client.DeviceClient INFO Disconnected from the IBM Watson IoT Platform

2022-11-08 22:57:43,580 wiotp.sdk.device.client.DeviceClient INFO Closed connection to the IBM Watson IoT Platform

{'speed': 20.0, 'doNotHonk': False}

.

.
```

.


Node-RED
Deploy
Flow 2
debug
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
function
switch
change
timestamp
function
function
function
Location
Visibility
Temperature
msg.payload
all nodes
all
11/8/2022, 10:57:06 PM node: cc9085c66f6e43a1
iot-2/type/testdevice/id/device1/evt : msg.payload : Object
object
temperature: 25.990000000000001
visibility: 25
location: "Chennai,IN"
11/8/2022, 10:57:43 PM node: cc9085c66f6e43a1
iot-2/type/testdevice/id/device1/evt : msg.payload : Object
{ temperature: 25.99000000000001, visibility: 25, location: "Chennai,IN" }
Home
Home
Visibility
25
%
0
100
Temperature
100
75
50
25
0
22:57:06
22:57:16
22:57:26
22:57:44
Location
Chennai,IN
MEM: 72 %
CPU: 0 %
11:01 PM
11/8/2022