

## Sprint 02

# Signs with Smart Connectivity for Better Road Safety

## Team ID - PNT2022TMID27042

Sprint Goals:

Push data from local code to cloud

Program Code:

> weather.py

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" : "epmoec",
        "typeId" : "testDevice",
        "deviceId" : "device0"
    },
    "auth" : {
        "token" : "?-KDXUPMvDo_TK2&b1"
    }
}
```

```
# 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
```

```
> brain.py
```

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.

```
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
```

```
> main.py
```

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 = "bf4a8d480ee05c00952bf65b78ae826b"

```
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

2022-11-06 21:38:33,452 wiotp.sdk.device.client.DeviceClient INFO

Connected successfully: d:epmoec:testDevice:device0

2022-11-06 21:38:33,452 wiotp.sdk.device.client.DeviceClient INFO

Disconnected from the IBM Watson IoT Platform

2022-11-06 21:38:33,452 wiotp.sdk.device.client.DeviceClient INFO Closed  
connection to the IBM Watson IoT Platform

{'speed': 40, 'doNotHonk': False}

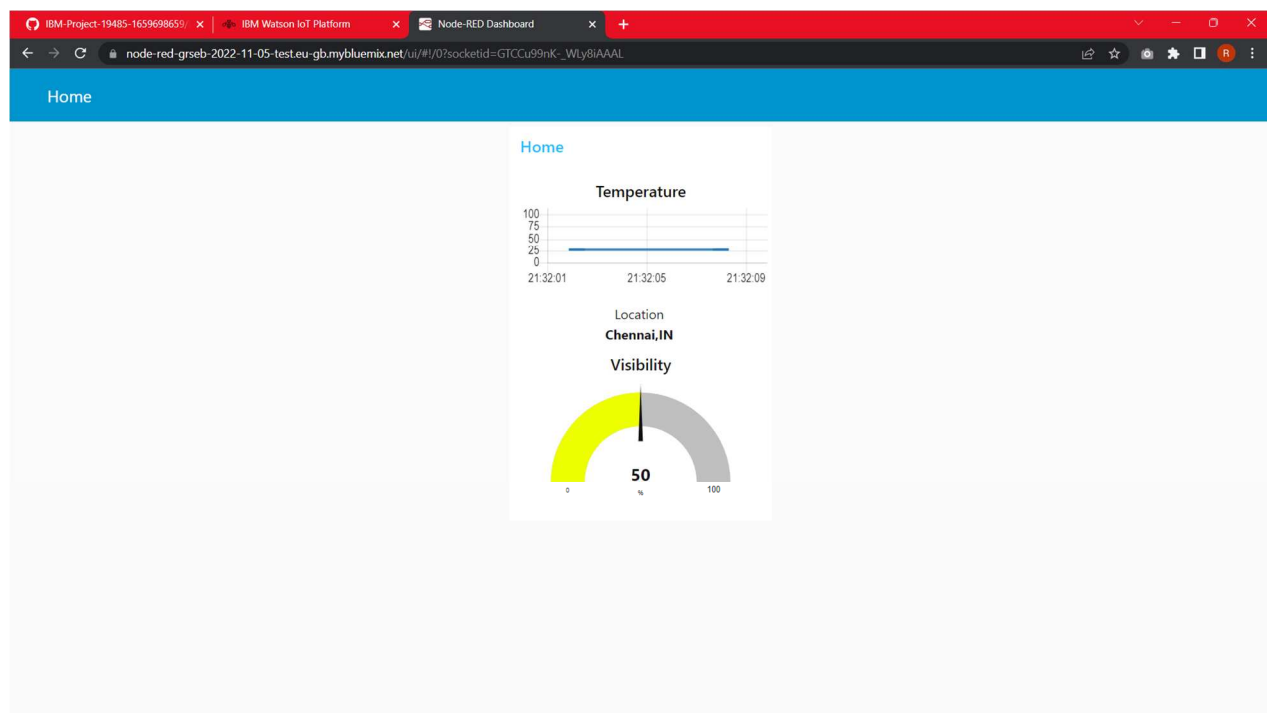
```
2022-11-06 21:38:35,631 wiotp.sdk.device.client.DeviceClient INFO
Connected successfully: d:epmoec:testDevice:device0
2022-11-06 21:38:35,631 wiotp.sdk.device.client.DeviceClient INFO
Disconnected from the IBM Watson IoT Platform
2022-11-06 21:38:35,631 wiotp.sdk.device.client.DeviceClient INFO Closed
connection to the IBM Watson IoT Platform
{'speed': 40, 'doNotHonk': False}
```

.  
.  
.

... repeats every 1 sec

Images :

OutputImage2



OutputImage

The screenshot shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left displays a file tree for a project named 'IBM'. The file tree includes folders for 'Project Development Phase' and 'Sprint 2', with 'main.py' selected. The main editor area shows the content of 'main.py', which is a Python script. The script includes configuration for 'hospitalsNearby' and 'usualSpeedLimit', a section for user input, and a while loop that prints the output of 'brain.processConditions'. The terminal at the bottom shows the execution of 'python main.py' and the resulting log output, which includes connection status messages to the IBM Watson IoT Platform.

```
19     "hospitalsNearby" : False,
20     "usualSpeedLimit" : 40 # in km/hr
21 }
22
23 # USER INPUT SECTION ENDS
24 # -----
25 # MICRO-CONTROLLER CODE STARTS
26 while True :
27     print(brain.processConditions(myLocation,APIKEY,localityInfo))
28
29     '''
30 MICRO CONTROLLER CODE WILL BE ADDED IN SPRINT 2 AS PER OUR PLANNED
31     '''
32 '''
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
P:\temp\IBM\Project Development Phase\Sprint 2>python main.py
2022-11-06 21:32:02,167 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:epmoec:testDevice:device0
2022-11-06 21:32:02,182 wiotp.sdk.device.client.DeviceClient INFO Disconnected from the IBM Watson IoT Platform
2022-11-06 21:32:02,182 wiotp.sdk.device.client.DeviceClient INFO Closed connection to the IBM Watson IoT Platform
({'speed': 40, 'doNotHonk': False})
2022-11-06 21:32:04,330 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:epmoec:testDevice:device0
2022-11-06 21:32:04,330 wiotp.sdk.device.client.DeviceClient INFO Disconnected from the IBM Watson IoT Platform
2022-11-06 21:32:04,330 wiotp.sdk.device.client.DeviceClient INFO Closed connection to the IBM Watson IoT Platform
({'speed': 40, 'doNotHonk': False})
```

Ln 29, Col 4 Spaces: 4 UTF-8 CRLF Python 3.8.5 64-bit Go Live Prettier