

Project Development Phase

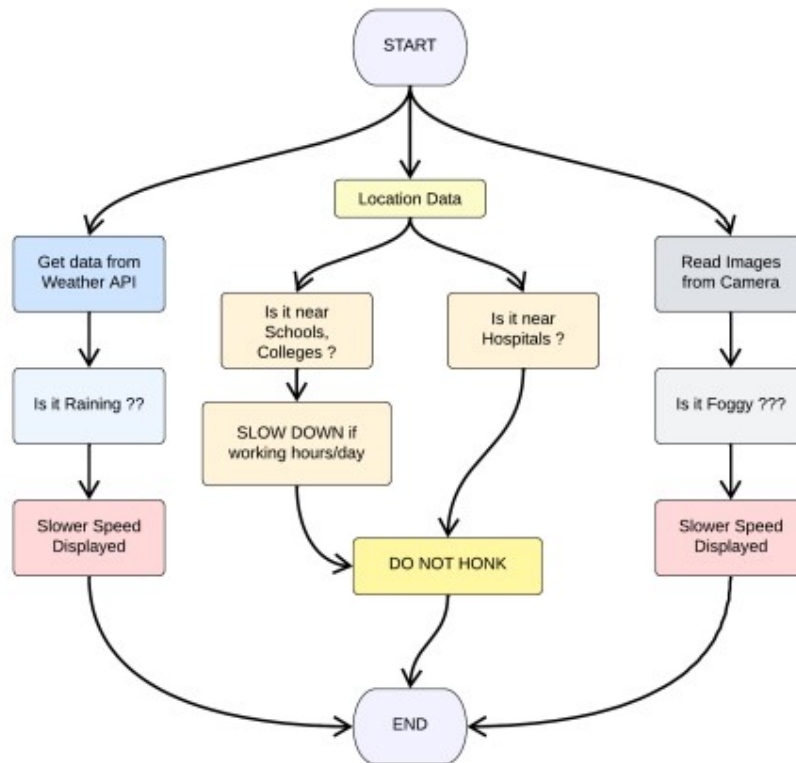
Date	November 2022
Team ID	PNT2022TMID41909
Project Name	Signs with Smart Connectivity for Better Road Safety

Project Development - Delivery of Sprint – 3

Sprint Goals :

1. Push data from local code to cloud

Code Flow :



Python 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)

```

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.

Python code

IMPORT SECTION STARTS

```

import weather
from datetime import datetime as dt

```

IMPORT SECTION ENDS

UTILITY LOGIC SECTION STARTS

```

def processConditions(myLocation,APIKEY,localityInfo):

```

```

    weatherData = weather.get(myLocation,APIKEY)

```

```

    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

publishData.py

This code pushes data to the cloud and logs data.

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" : "tx13k3",
        "typeId" : "device3",

```

```

        "deviceId" : "123123"

    },

    "auth" : {

        "token" : "123456123456"

    }

}

# 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

```

main.py

The code that runs in a forever loop in the micro-controller. This calls all the until 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 = "478d1352b25c4689912e8d6acbbc50b1"

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'''

Output:

A screenshot of a Windows desktop environment. The primary focus is a "Python 3.6.5 Shell" window. The title bar includes standard Windows window controls (minimize, maximize, close) and the text "*Python 3.6.5 Shell". Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Window, Help. The main area of the shell displays the following text:

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
RESTART: C:\Users\MONISH\AppData\Local\Programs\Python\New folder\s 3\main.py  
{'speed': 20.0, 'doNotHonk': True}  
{'speed': 20.0, 'doNotHonk': True}  
{'speed': 20.0, 'doNotHonk': True}  
{'speed': 20.0, 'doNotHonk': True}  
{'speed': 20.0, 'doNotHonk': True}  
{'speed': 20.0, 'doNotHonk': True}  
{'speed': 20.0, 'doNotHonk': True}
```


At the bottom of the screen is the Windows taskbar. It features a search bar on the left with the placeholder text "Type here to search". To the right of the search bar are several application icons: a house icon, a circular icon, a task view icon, the Microsoft Edge browser icon, the Google Chrome icon, and the Microsoft Word icon.

The image shows a Windows desktop with two application windows. The left window is titled 'main.py - C:\Users\MONISH\AppData\Local\Programs\Python\New folder\s 3\main.py (3.6.5)'. It contains Python code with comments in red and black. The code defines variables for location ('Chennai, IN'), API key, and a dictionary for school and hospital locations. It includes a loop for processing conditions. The right window is titled '"Python 3.6.5 Shell"'. It shows the command prompt output for running the script, displaying a list of speed and honk status pairs for various locations. The desktop taskbar at the bottom shows the Start button, search bar, and several pinned applications including File Explorer, Chrome, and Word. The system tray shows the date and time as 1:37 PM on 11/15/2022.

