IOT PROJECT

<u>Smart Connected Signs for Improved Road Safety</u>

By:

Karthik P AjithKumar(19BEE1216)

Aaron Richard Thomas(19BEE1132)

Mishel Thomas Mohan(19BEE1145)

Noel Saji (19BEE1173)

I.<u>INTRODUCTION</u>

- **OVERVIEW**
  In this project we have built '*Smart Connected signs for Improved Road Safety'* using the IBM IOT platforms. Depending upon the weather condition received from the OpenWeatherApp we have displayed the speed to be maintained while driving, the traffic Density etc. through different areas like Schools and Hospitals. We also have got the data in a mobile application using MITInventorApp.

- **PURPOSE**
  The road safety in our country has remained idle and constant for a brief span of time. In present Systems the road signs and the speed limits are Static. But the road signs can be changed in some cases. We can consider

some cases when there are some road diversions due to heavy traffic or due to accidents then we can change the road signs accordingly if they are digitalized. This project proposes a system which has digital sign boards on which the signs can be changed dynamically. If there is rainfall then the roads will be slippery and the speed limit would be decreased. There is a web app through which you can enter the data of the road diversions, accident prone areas and the information sign boards can be entered through web app. This data is retrieved and displayed on the sign boards accordingly.

II.LITERATURE SURVEY

- **EXISTING PROBLEM**

In our country there are numerous problems regarding the road safety for the driver and also the people surrounding him. Some of these existing problems are:

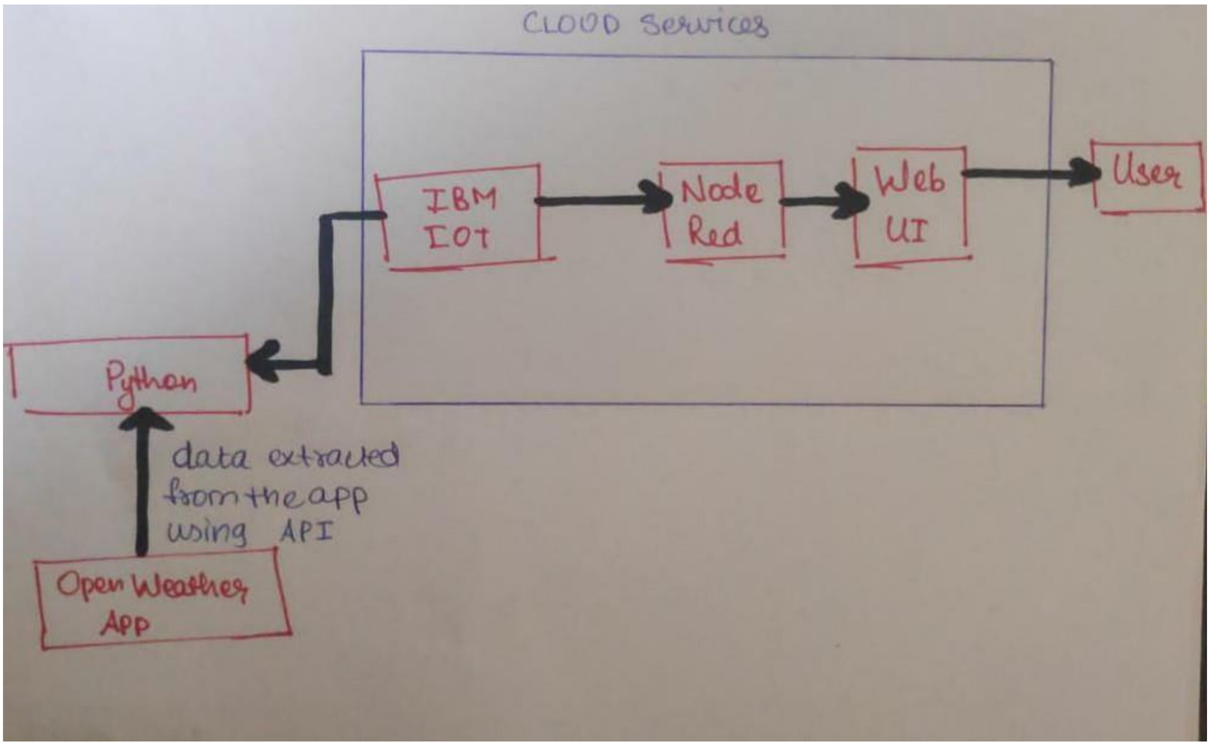
1. Alcohol Impaired Driving
2. Autonomous Vehicle
3. Drowsy Driving
4. Drug Impaired Driving
5. Speeding and Aggressive Driving

- **PROPOSED SOLUTION**

This project proposes a system which has digital sign boards on which the signs can be changed dynamically. If there is rainfall then the roads will be slippery and the speed limit would be decreased. There is a web app through which traffic density is displayed and also instructs to take diversion if needed. Sign boards can be changed from webapp and android app .This data is retrieved and displayed on the sign boards accordingly.

III. THEORITITCAL ANALYSIS

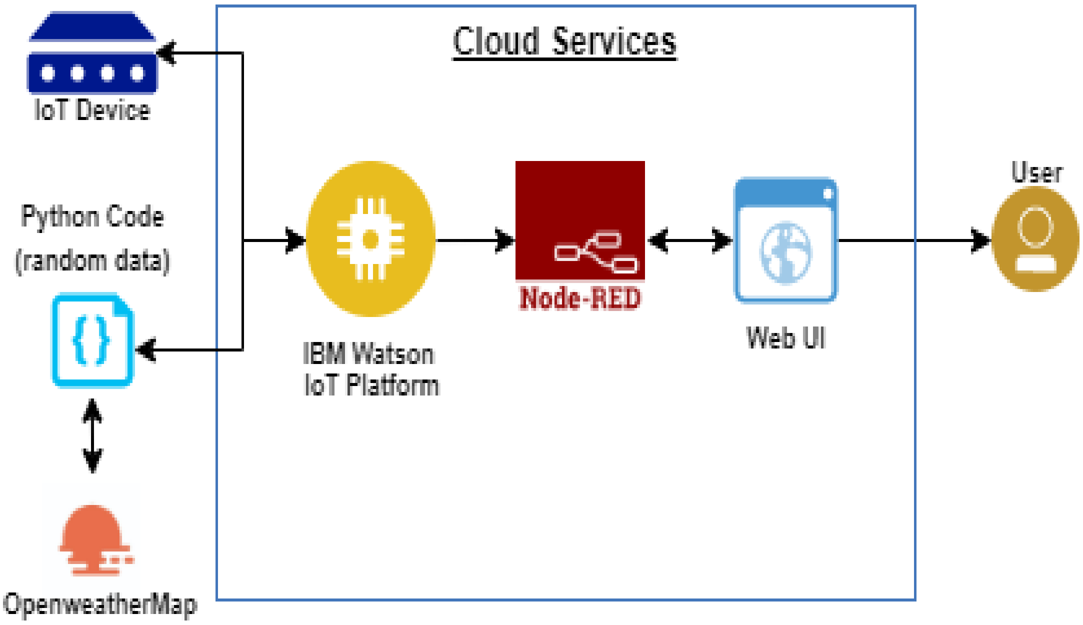
- **BLOCK DIAGRAM**



- **HARDWARE AND SOFTWARE DESIGNING**

  In this project that we have completed, my team have not used any hardware device for obtaining data. Every aspect of the project was designed from the software point of view including obtaining Weather Data. Usually, the weather data is obtained using hardware devices since it was not possible for us, we used *OpenWeatherApp* to obtain the weather data using the API keys. As far the other processes are concerned, we used *Python* to extract the weather data and we used the *IBM Cloud* platform to store the data and the signs for the zones (School, Hospital, etc.). Then we created an app using *MIT Inventor App* for displaying the details.

IV.FLOWCHARTS



V.RESULTS

Weather data is obtained with the help of openweathermap and all those data are stored in cloudant database. The speed limit is decided based on the visibility and displayed in web ui and Android app. Signs to be displayed can be selected using both webapp and Android app.

Mobile app :

V.ADVANTAGES AND DISADVANTAGES

- **ADVANTAGES**
  When it comes to the advantages, Smart improved road signs are highly beneficial for us in our day to day life because the current road safety system in our country does not account for weather conditions which is one of a major factor for road safety, but in our work we have considered that factor and also the traffic density which gives us a clear information on speed to maintained while driving through different zones(Schools, Hospital etc.).

- **DISADVANTAGES**
  One of the disadvantages is that we do not have the accurate value for the weather description.

VI.APPLICATIONS AND FUTURE SCOPE

- In our near future we can improve the road safetify with the following benefits:

Less-congested streets. For an average US citizen, congestion costs 99 hours of their time and USD 1,377 each year.[2] Smart road technology can track vehicles and adjust traffic lights when there are fewer or no cars approaching, helping prevent bumper-to-bumper traffic. This could help drivers and passengers save 9.4 hours each year.[1]

Improved traffic and pedestrian safety. Traffic-monitoring solutions powered by computer vision can detect vehicles, pedestrians, and bicyclists to help enable safety practices. In the event of a crash or crime on the street, smart devices can immediately alert first responders.

Enhanced parking and e-tolling. E-tolling reduces congestion by using license plate recognition and vehicle tracking to automatically charge highway and bridge tolling fees—all without making vehicles stop or slow down.

By sending select data to the cloud to be analyzed over time, cities can make continual improvements in traffic management, road maintenance, and environmental quality, for example:

Identifying problem areas. Analytics can help detect intersections or other sites that have a high rate of collisions or near misses between vehicles or pedestrians. This helps cities determine if the site would benefit from a yield or stop sign, crosswalk, or traffic light.

Improving pavement conditions. Over time, streets erode or weather away. With road condition monitoring, cities can assess pavement conditions and act accordingly.

Reducing pollution. Smart infrastructure can help reduce carbon emissions from daily transportation by optimizing traffic flow to avoid idling engines. Cities can also help reduce pollution by understanding where to best place electric vehicle charging stations

VII.BIBLIOGRAPHY

www.youtube.com

www.smartinternz.com

www.smartbridge.com

VIII.APPENDIX

- **SOURCE CODE**
  Python Code for obtaining the weather data and publishing it to Nodered

```
# Python program to find current
# weather details of any city
# using openweathermap api

# import required modules
import requests, json
import wiotp.sdk.device
import time
import random


'''
# Enter your API key here
api_key = "560c4b69bb0a2c87ee3b704e9efed8e8"

# base_url variable to store url
base_url = "http://api.openweathermap.org/data/2.5/weather?"

# Give city name
city_name = input("Enter city name : ")

# complete_url variable to store
# complete url address
#complete_url = base_url  + "appid=" + api_key + "&q=" + city_name '''


complete_url =
"https://api.openweathermap.org/data/2.5/weather?units=metric&appid=5
60c4b69bb0a2c87ee3b704e9efed8e8&q=Delhi"
print(complete_url)

# get method of requests module
# return response object
response = requests.get(complete_url)
```

```
# json method of response object
# convert json format data into
# python format data
x = response.json()

# Now x contains list of nested dictionaries
# Check the value of "cod" key is equal to
# "404", means city is found otherwise,
# city is not found
if x["cod"] != "404":

	# store the value of "main"
	# key in variable y
	y = x["main"]

	# store the value corresponding
	# to the "temp" key of y
	current_temperature = y["temp"]

	# store the value corresponding
	# to the "pressure" key of y
	current_pressure = y["pressure"]

	# store the value corresponding
	# to the "humidity" key of y
	current_humidity = y["humidity"]



	# store the value of "weather"
	# key in variable z
	z = x["weather"]

	# store the value corresponding
	# to the "description" key at
	# the 0th index of z
	weather_description = z[0]["description"]

	current_visibility = x["visibility"]

	# print following values
	print(" Temperature (in Celcius unit) = " +
					str(current_temperature) +
```

```
                    "\n atmospheric pressure (in hPa unit) = " +
                                        str(current_pressure) +
                    "\n humidity (in percentage) = " +
                                        str(current_humidity) +
                    "\n description = " +
                                        str(weather_description)+
                "\n Visibilty = " +
                                str(current_visibility))
else:
        print(" City Not Found ")

if current_visibility < 200:
      speed = 10
elif current_visibility <= 3000 and current_visibility >=200:
      speed = 30
elif current_visibility <= 6000 and current_visibility >= 3000:
      speed = 40
else:
      speed = 60
print("Maitain Speed to: ",speed)
myConfig = {
   "identity": {
      "orgId": "jt2v00",
      "typeId": "VITDevice",
      "deviceId":"12345"
   },
   "auth": {
      "token": "12345678"
   }
}

def myCommandCallback(cmd):
   print("Message received from IBM IoT Platform: %s" %
cmd.data['command'])
   #m=cmd.data['command']
   #if m == "lighton":
   #    print("Light is on")
   #elif m == "lightoff" :
   #    print("Light is off")
   print()

client = wiotp.sdk.device.DeviceClient(config=myConfig,
logHandlers=None)
```

```
client.connect()

tf_density = random.randint(0,100)
if tf_density > 80:
        tf_density = 'Pls take left diversion , Traffic ahead'
        divr = True
else:
        tf_density = 'No Traffic Ahead...Safe Journey'
        divr = False
print(tf_density)

myData={'temperature':current_temperature,
'humidity':current_humidity,
'des':weather_description,'visible':current_visibility,'Speed':speed,'density'
:tf_density,'div':divr}
client.publishEvent(eventId="status", msgFormat="json", data=myData,
qos=0, onPublish=None)
print("Published data Successfully: %s", myData)
while True:
    #temp=random.randint(-20,125)
    #myData={'density':tf_density}
    client.commandCallback = myCommandCallback
    time.sleep(2)
client.disconnect()
```

Python Code for uploading the signs to cloudant database

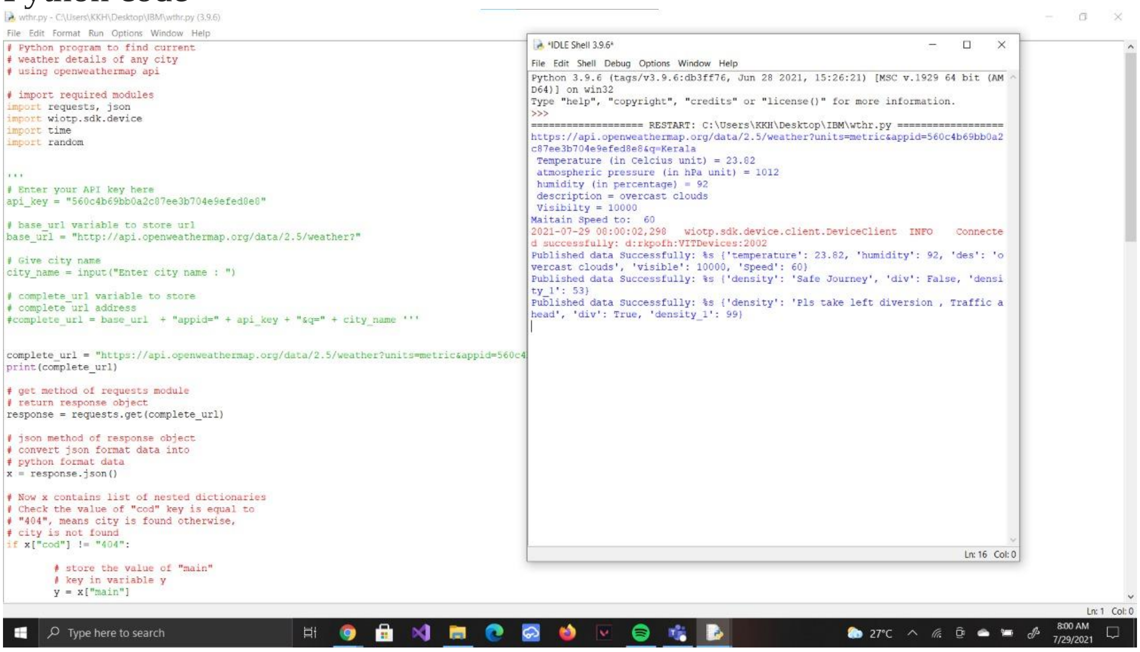
```
import datetime
import ibm_boto3
from ibm_botocore.client import Config, ClientError
from ibmcloudant.cloudant_v1 import CloudantV1
from ibmcloudant import CouchDbSessionAuthenticator
from ibm_cloud_sdk_core.authenticators import BasicAuthenticator

# Constants for IBM COS values
COS_ENDPOINT = "https://s3.jp-tok.cloud-object-
storage.appdomain.cloud" # Current list avaiable at https://control.cloud-
object-storage.cloud.ibm.com/v2/endpoints
#COS_API_KEY_ID =
"eJuMGEJg913QufpYpcw8H4yIlhWMfTA8IKbKwB2syTbQ" # eg
"W00YixxxxxxxxxxMB-odB-2ySfTrFBIQQWanc--P3byk"
#COS_INSTANCE_CRN = "crn:v1:bluemix:public:cloud-object-
storage:global:a/68a32c0a4a824d6399a39e40e6a6ca31:faa157de-e615-
```

```
452c-9015-98f3efbc9173::" # eg "crn:v1:bluemix:public:cloud-object-storage:global:a/3bf0d9003xxxxxxxxxx1c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"
COS_API_KEY_ID =
"USvlJcgdNYOed5l2dpRYOLZEIdyizovP27GlnbEdelzP" # eg
"W00YixxxxxxxxxxMB-odB-2ySfTrFBIQQWanc--P3byk"
COS_INSTANCE_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/b4407533a10a4e5da667852ff663e7dd:3f1789f6-8d9b-4e04-ae79-8058fd18c1f3::" # eg "crn:v1:bluemix:public:cloud-object-storage:global:a/3bf0d9003xxxxxxxxxx1c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"

# Create resource
cos = ibm_boto3.resource("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_INSTANCE_CRN,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)

authenticator = BasicAuthenticator('apikey-v2-2ms0p6wxyzft3spsof0u2dslfgntpzzupu7pcwp9rwl','72aaf50add88e1e54ad53b3f53418cea')
service = CloudantV1(authenticator=authenticator)
service.set_service_url('https://apikey-v2-2ms0p6wxyzft3spsof0u2dslfgntpzzupu7pcwp9rwl:72aaf50add88e1e54ad53b3f53418cea@2b65707c-b34f-474b-99bc-9be10d93ff37-bluemix.cloudantnosqldb.appdomain.cloud')

bucket = "karthikvit"
def multi_part_upload(bucket_name, item_name, file_path):
    try:
        print("Starting file transfer for {0} to bucket:
{1}\n".format(item_name, bucket_name))
        # set 5 MB chunks
        part_size = 1024 * 1024 * 5

        # set threadhold to 15 MB
        file_threshold = 1024 * 1024 * 15

        # set the transfer threshold and chunk size
        transfer_config = ibm_boto3.s3.transfer.TransferConfig(
            multipart_threshold=file_threshold,
```

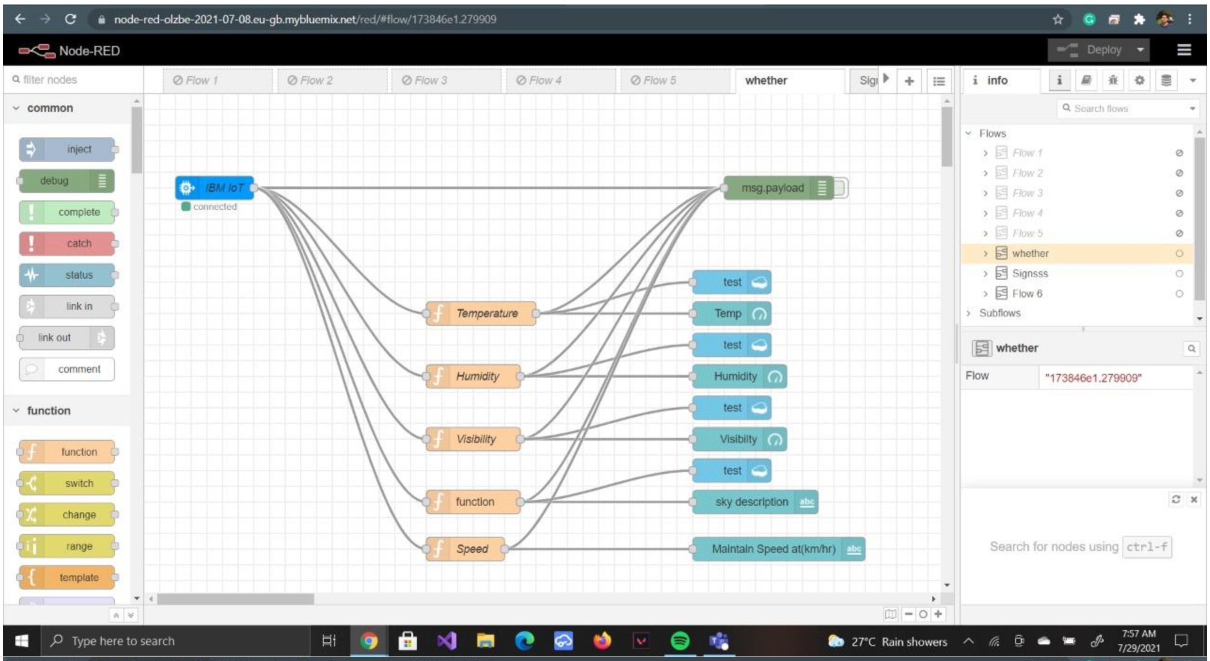
```
        multipart_chunksize=part_size
    )

    # the upload_fileobj method will automatically execute a multi-part upload
    # in 5 MB chunks for all files over 15 MB
    with open(file_path, "rb") as file_data:
        cos.Object(bucket_name, item_name).upload_fileobj(
            Fileobj=file_data,
            Config=transfer_config
        )

    print("Transfer for {0} Complete!\n".format(item_name))
  except ClientError as be:
    print("CLIENT ERROR: {0}\n".format(be))
  except Exception as e:
    print("Unable to complete multi-part upload: {0}".format(e))
picname='download (4)' #datetime.datetime.now().strftime("%y-%m-%d-%H-%M")
multi_part_upload('karthikvit', picname+'.jpg', picname+'.jpg')
json_document={"link":COS_ENDPOINT+'/'+bucket+'/'+picname+'.jpg'}
response = service.post_document(db='ped', document=json_document).get_result()
```

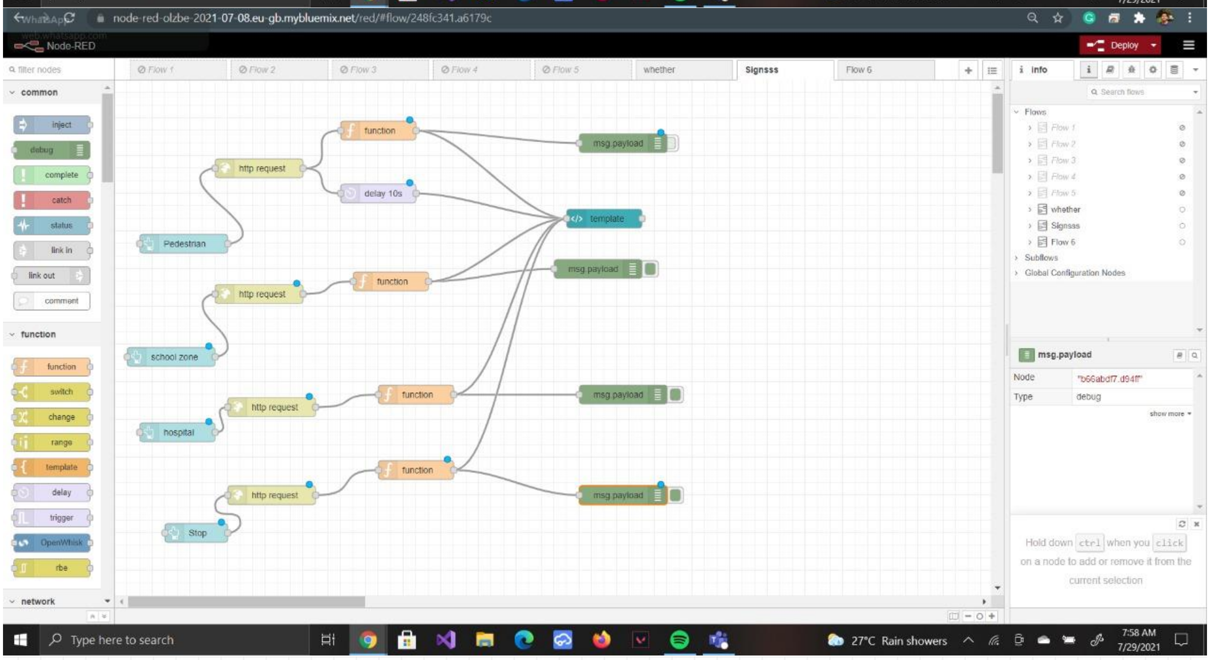
- **SCREENSHOTS**

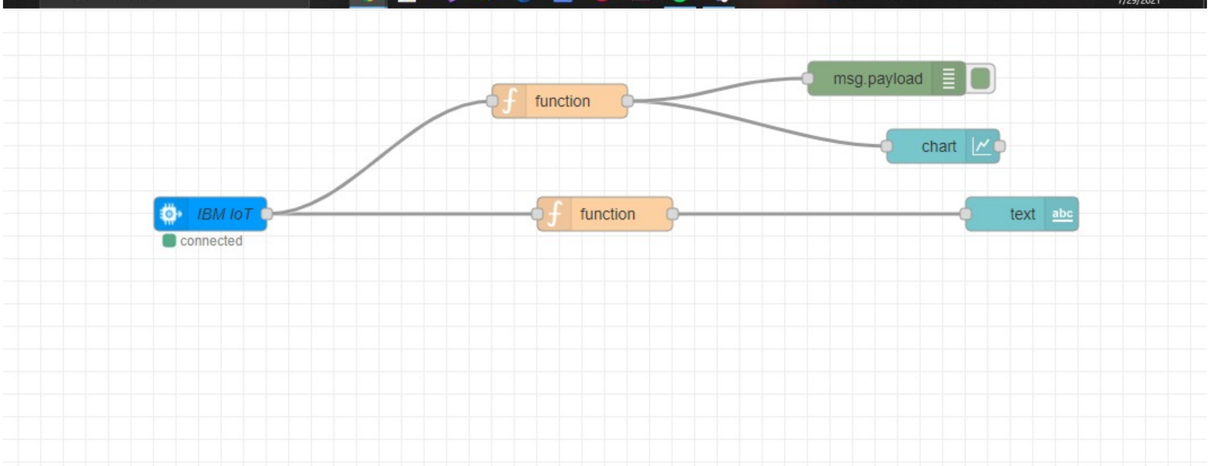
Python code



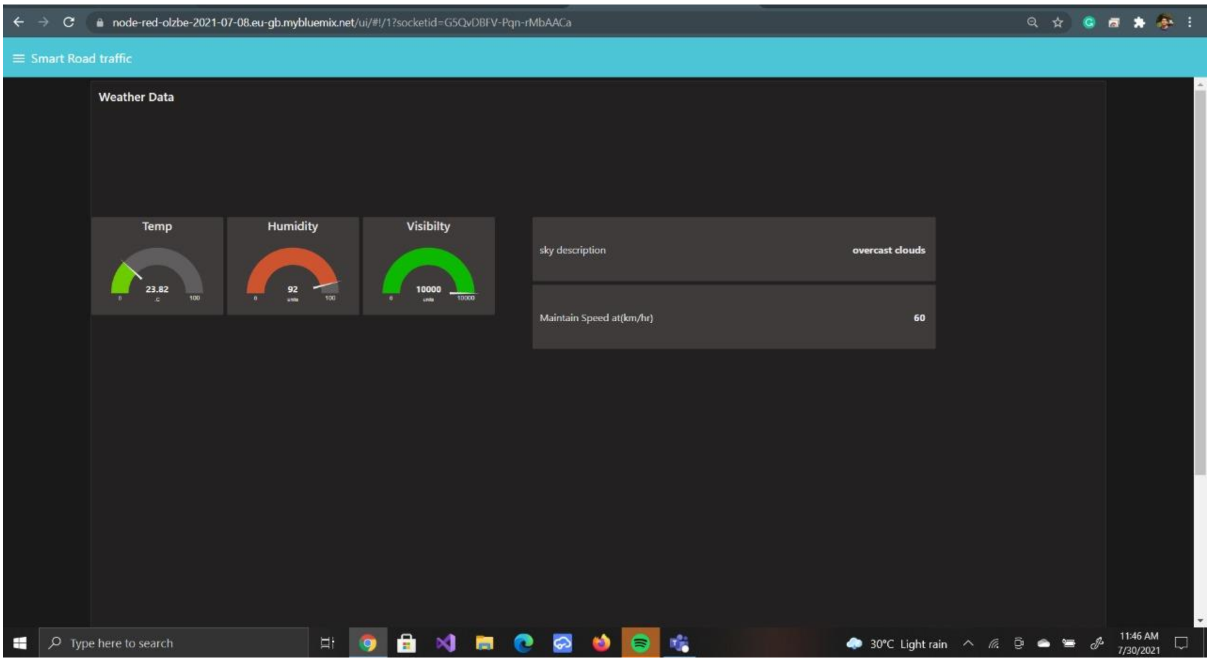
Nodered Flows:



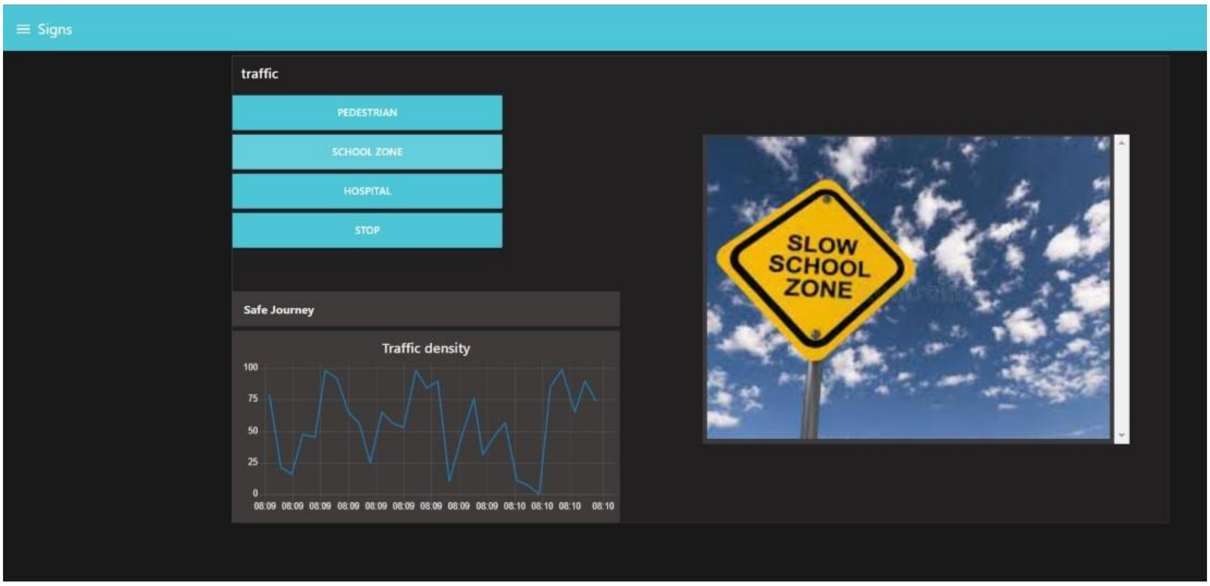




Web UI Outputs :




Signs
traffic
PEDESTRIAN
SCHOOL ZONE
HOSPITAL
STOP
Safe Journey
Traffic density
HOSPITAL
shutterstock.com · 102481208


Signs
traffic
PEDESTRIAN
SCHOOL ZONE
HOSPITAL
STOP
Safe Journey
Traffic density
SLOW
SCHOOL
ZONE


Signs
traffic
PEDESTRIAN
SCHOOL ZONE
HOSPITAL
STOP
Pls take left diversion , Traffic ahead
Traffic density

Data stored in IBM Cloudant database