

IOT BASED VECHICLE POLLUTION
MONITORING SYSTEM

An Internship project

Submitted by

J.V.Srikanth
(18481A0478)

I.Manikanta
(18481A0475)

G.Jayadeep sai
(18481A0474)

J.Hynar
(18481A0477)

J.Ganesh
(18481A0479)

INTRODUCTION

Overview:

The overview of our project is to design a monitoring system for vehicles so that the percentage of pollution caused due to each individual vehicle can be monitored and recorded. We will also be able to observe the individual pollutants and their percentages.

Purpose:

The main source of pollution in cities is due to vehicles. The increase use of vehicles in cities results in vital increase in the emission load of various toxins into air. As a result, increase in environmental problems which will affect the human health in urban places. So, in order to regulate the level of pollution, it is necessary to check the amount of pollution, being made by individual vehicles. For this reason, we've made a pollution monitoring system for vehicles using Internet of Things.

LITERATURE SURVEY

Existing problem:

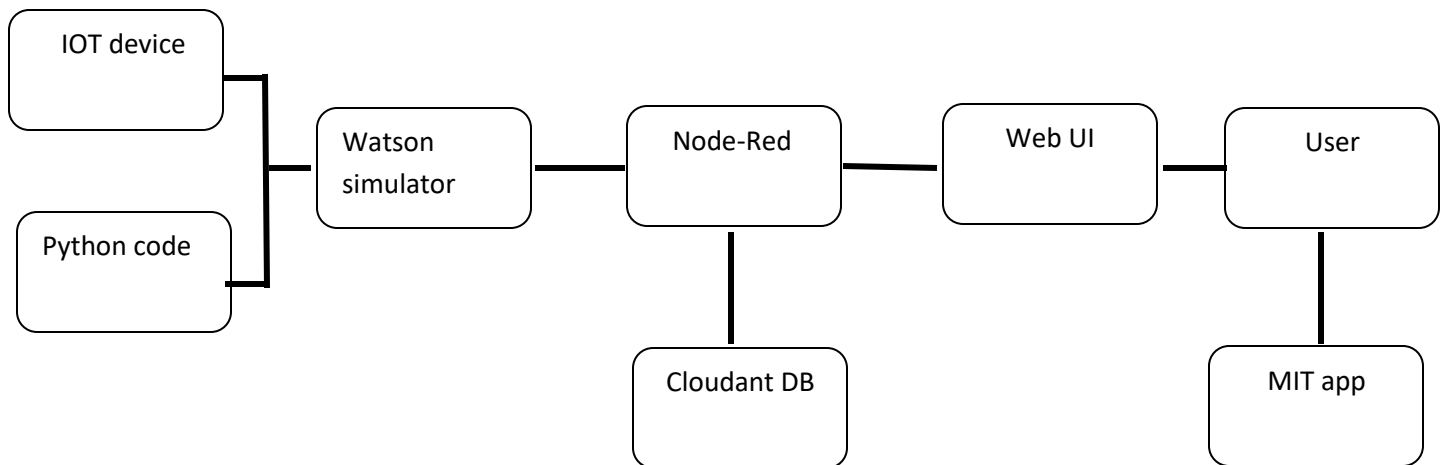
Every vehicle in a city emits a certain amount of pollutants into the atmosphere. This in turn, on a massive vehicle count, leads to major air pollution problem in the city. People are not aware of the percentage of pollution they are causing with unnecessary use of motor vehicles. This problem exists until some solution is made.

Proposed solution:

In this project, we find a smart solution to the above mentioned problem, making use of Internet of Things. The solution is such that, the pollutants intensity will be recorded by the sensors, and the data will be sent to mobile app installed in the vehicle owner's mobile. With this continuous evaluation, one would limit his/her usage of vehicles. We also provide the locations covered by the vehicle on a particular day or the current location of the vehicle using Google Maps.

THEORITICAL ANALYSIS

Block diagram:



Software design:

In our project, we develop a software design for achieving the mentioned objective. Initially, we write a python code with the Watson credentials in order to link the data to the simulator. In Node-red we connect the required nodes in order to complete the process flow. The pollutant values recorded i.e., through the python code are displayed in the UI page. We also include location access in this software.

Now, we need a mobile app in order update the information to the owner. So, we make use of MIT app inventor and make necessary format of the mobile app and with this we are done.

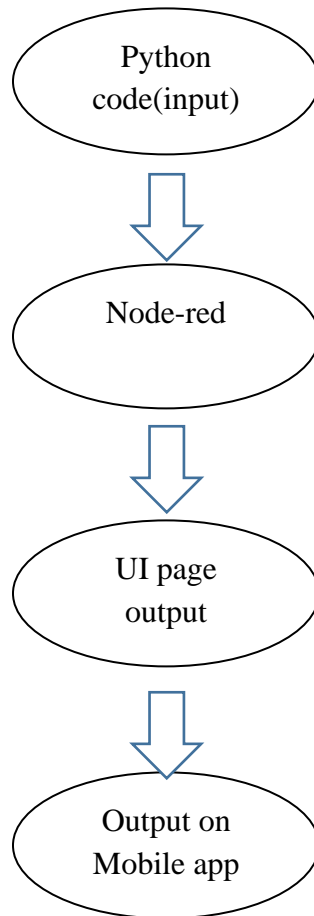
EXPERIMENTAL INVESTIGATIONS

Initially, the data is assumed to be recorded in a hardware model and the values are given to the python program. Using, the credentials, we link the python program to the cloudant DB for syncing the data to the cloud dynamically. Now we move on to node-red, where a defined process flow is constructed using the nodes provided in the dashboard. This defines a meaningful path for our data to be processed to further execution.

Using node-red, we observe the output on the UI page, python program running in the background. The output is also displayed in the debug window of node-red. For now, our data is received successfully from the cloud, this is to be updated and displayed on the mobile app. For, this we build the front end of our mobile app, and we download the apk of the app and our app is ready to use.

For example, let us consider two pollutants, nitrogen and carbondioxide. Their intensity levels will be updated to the python program also the co-ordinates of the location of the vehicle. Now these all data will be displayed in the UI page as well as in the debug window in node-red. The final output will be the values being updated to the mobile app.

FLOWCHART



RESULT

The proposed software design provides a solution for the problem statement mentioned above and we have implemented and proved the solution. The final result can be checked on the mobile app i.e., the pollutant levels and location of the vehicle.

ADVANTAGES & DISADVANTAGES

There are definitely many advantages of designing this software. This model aids one to check the pollutant levels every day. Through the statistical data made, say for a week, one could plan to limit the usage of vehicle so that for the next week they could reach their minimized target.

Our proposed idea helps in reducing air pollution definitely to an extent in a city/town if brought into implementation. This in turn helps a lot in reducing the global warming making our model eco-friendly. Whereas, coming on to the disadvantages side, one should be checking regularly if its working, there isn't any self-configurable design embedded in it.

APPLICATIONS

This monitoring system is very effective in metropolitan cities like Delhi, Mumbai, Hyderabad etc. This system can be installed in any motor vehicle and can know the details of the pollution along with the location of the vehicle.

CONCLUSION

- In this project, we have designed and implemented a proposed model of IOT based vehicle pollution monitoring system, using Node-Red, Python code and MIT App inventor.
- This proposed idea helps us in finding a solution for the problems faced due to large extents of air pollution.

FUTURE SCOPE

Our project can be further extended by adding message pop-up if the pollutant levels above a particular threshold value. Some modifications can be made so that the monitoring if is about to expire, would identify its health status and send a notification to the owner to check its status. Further setups like suggesting number of kilometers travelled in a day would also be improvising its performance.

BIBLIOGRAPHY

Famesh D. Thakre , Bidyut K. Talukdar, Gaurav S. Gosavi , Prashant R. Tayade, “Minimization of CO & CO₂ from Exhaust of Two Wheeler Motorcycle”, Vol. 4, Special Issue 3, January 2017.

Y. J. Jung, Y. K. Lee, D. G. Lee, K. H. Ryu, and S. Nittel, “Air pollution monitoring system based on geosensor network”, in Proc. IEEE Int. Geoscience Remote Sensing Symp., 2008, vol. 3, pp. 1370- 1373.

APPENDIX

Source code:

```
import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

import json

#Provide your IBM Watson Device Credentials

organization = "gaxcrt"

deviceType = "iotdevice"

deviceId = "1001"

authMethod = "token"

authToken = "Chinna@9966"

# Initialize the device client.

co2=0

n2=0

la=0

lg=0

loc=[]

def myCommandCallback(cmd):

    print("Command received: %s" % cmd.data['command'])

    if cmd.command == "setInterval":

        if 'interval' not in cmd.data:
```

```

        print("Error - command is missing required information: 'interval'")

    else:

        interval = cmd.data['interval']

    elif cmd.command == "print":

        if 'message' not in cmd.data:

            print("Error - command is missing required information: 'message'")

        else:

            print(cmd.data['message'])

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()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type
"greeting" 10 times

deviceCli.connect()

while True:

    co2=89

    n2=45

    la=16.0948

    lg=80.1656

    loc="chilakaluripet"

    #Send waterlevel & light intensity to IBM Watson

```

```
data = {"d":{ 'CO2' : co2, 'N2': n2,'LA': la,'LG': lg,"LOC": loc, }}

#print data

def myOnPublishCallback():

    print ("Published CO2 = %s units" % co2, "N2 = %s %" % n2, "LA = %s degrees" % la,
"LG = %s degrees %" % lg,"to IBM Watson")

    success      =      deviceCli.publishEvent("Data",      "json",      data,      qos=0,
on_publish=myOnPublishCallback)

    if not success:

        print("Not connected to IoTF")

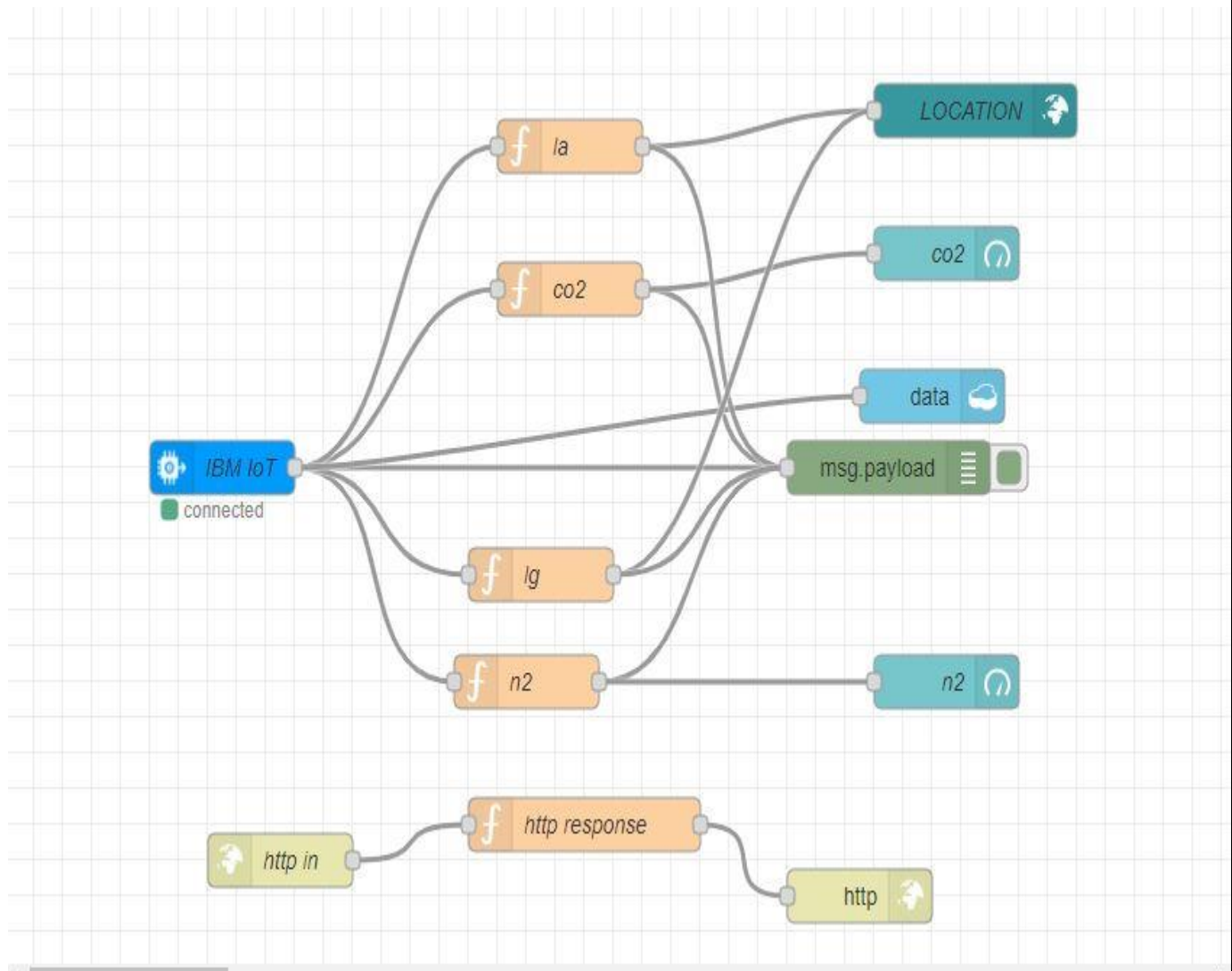
    time.sleep(1)

    deviceCli.commandCallback = myCommandCallback

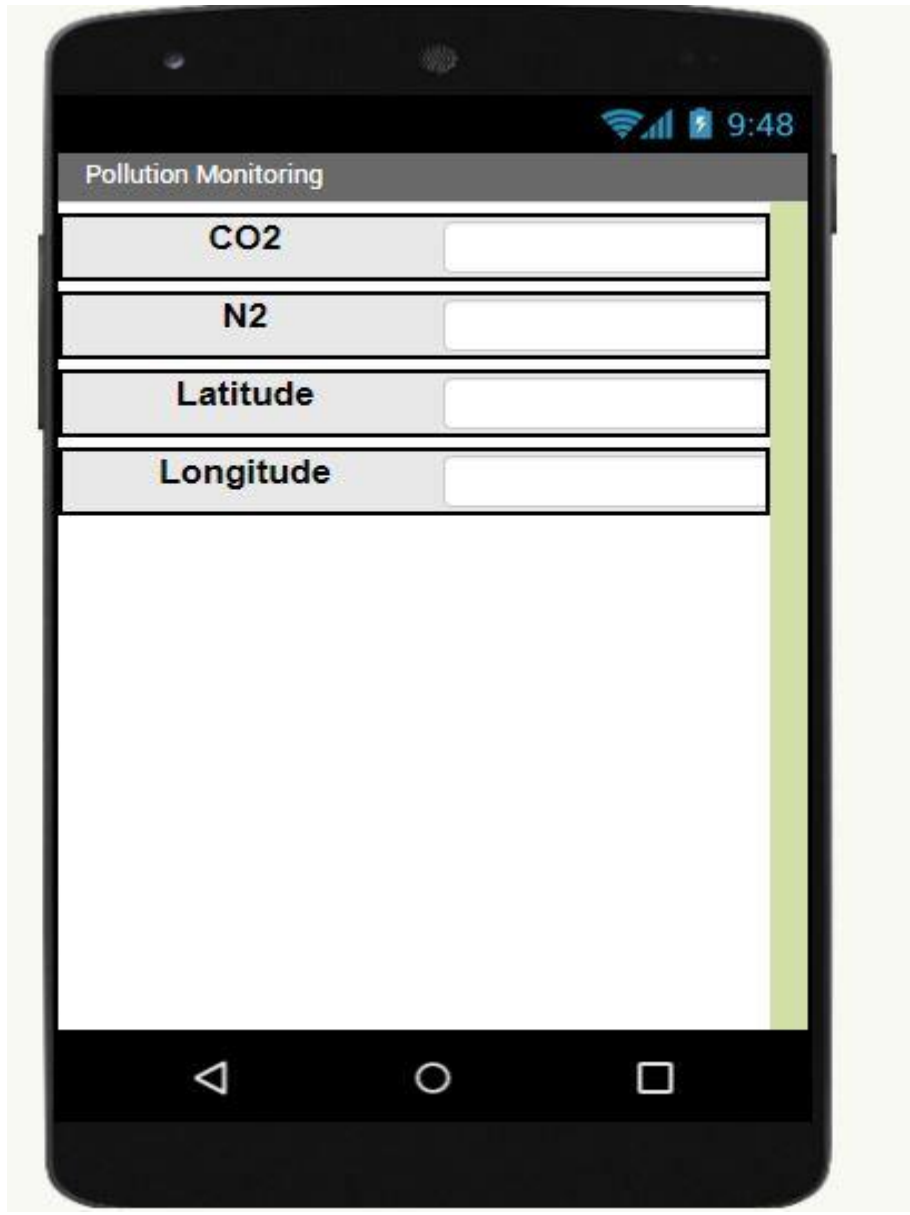
# Disconnect the device and application from the cloud

deviceCli.disconnect()
```

Node-red screenshot:



MIT App screenshot:



A screenshot of a smartphone displaying a "Pollution Monitoring" app. The app's interface is a form with four input fields, each preceded by a label in a grey box. The labels are "CO2", "N2", "Latitude", and "Longitude". Each label is followed by a white rectangular input field. The form is set against a light green background. The smartphone's status bar at the top shows a Wi-Fi icon, a signal strength icon, a battery icon, and the time "9:48". The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.

Pollution Monitoring	
CO2	<input type="text"/>
N2	<input type="text"/>
Latitude	<input type="text"/>
Longitude	<input type="text"/>