NOISE POLLUTION MONITORING

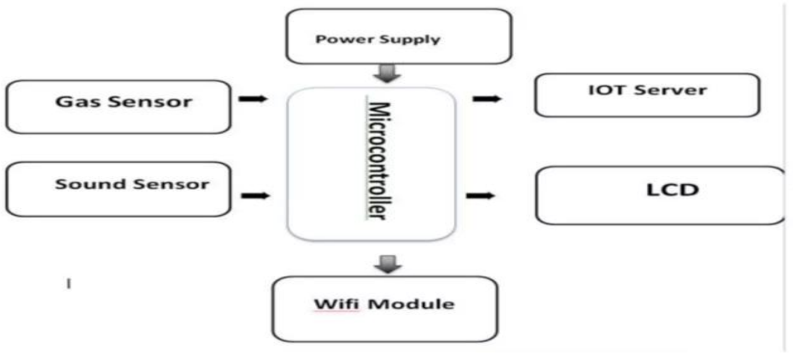
INTRODUCTION

Air, Sound, Temperature and humidity are main factors that most contribute to the global climate patterns. Their changes shall affect the environment worldwide where life cycles of plants and animals become different. This paper describes the implementation of a wireless air, sound, temperature and humidity for autonomous monitoring on a Raspberry Pi. Hence, the objective of this paper is two-fold, first is to emphasize the development of wireless environment for autonomous monitoring on an embedded Raspberry Pi. Secondly, we have explained the development of an alert system for monitoring the readings when it reached a certain condition. Finally perform the analysis on the result obtained from sensors readings in the experimental areas. It is also shall reduce the uses of manual monitoring such as analog thermometer that contribute to a lot of errors for the reading to be taken. It is good to be notified when a certain temperature condition is met for the next step to be taken. In this model, we are using a Raspberry Pi 3B microcontroller, which will have gas sensors, temperature and humidity sensors and noise sensors connected to it, to monitor the fluctuating environmental parameters. OS is coded with Python language for retrieving air, sound, temperature and humidity readings where the values are sensed through sensor and sent to the internet. The rest of this paper explains about literature review, methodology, result and its discussion for analysis part.

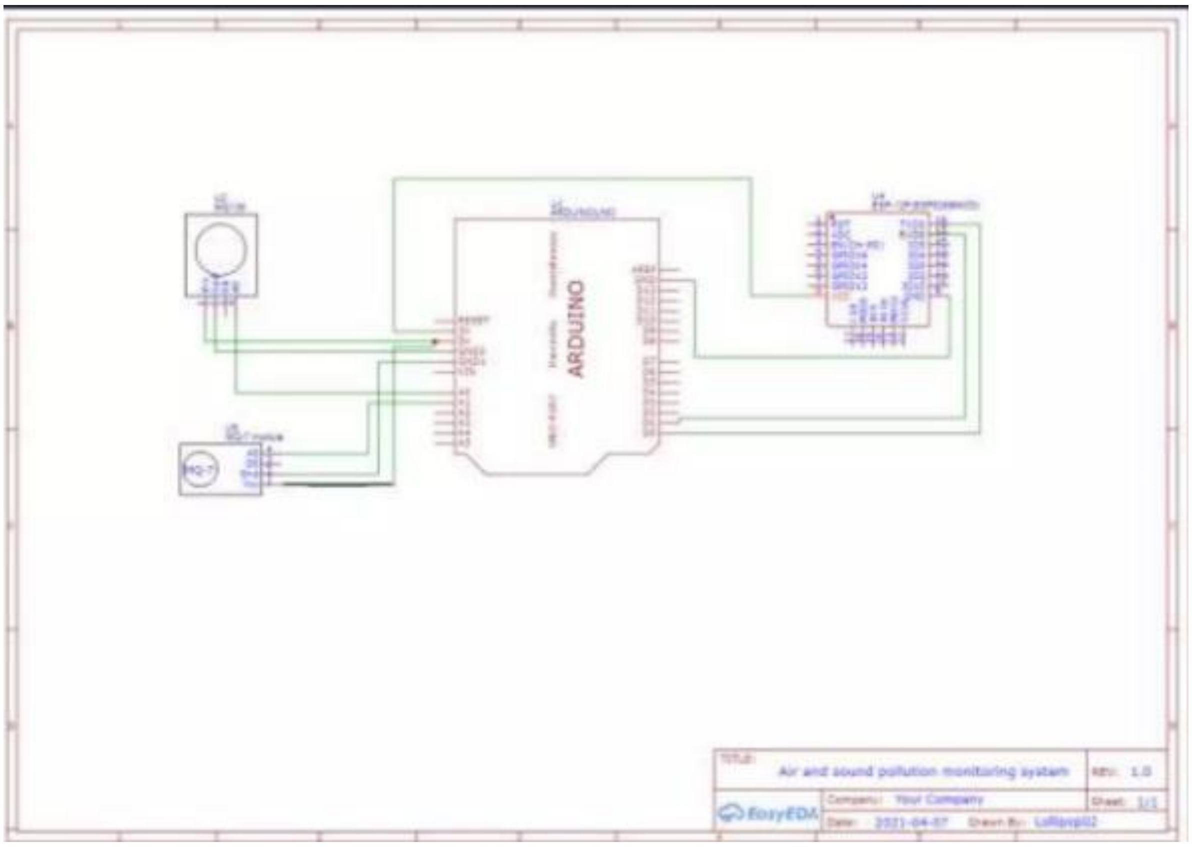
NEED OF PROJECT

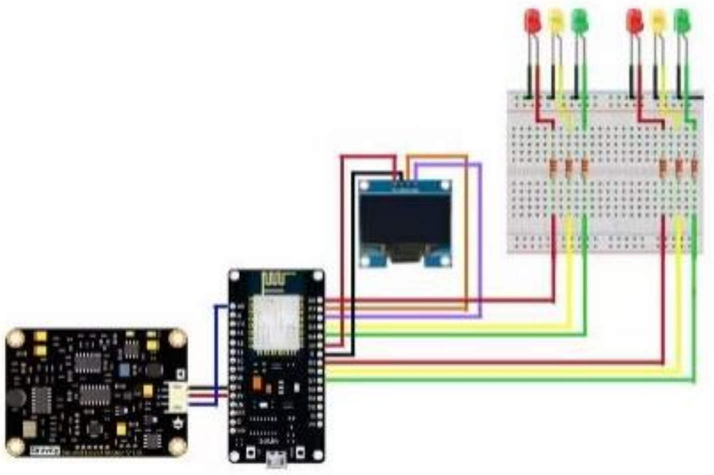
1) The air and sound pollution is increasing abruptly
2) To maintain its monitoring is essential
3) This system allows us to detect sound and air pollution levels over and over based on IOT

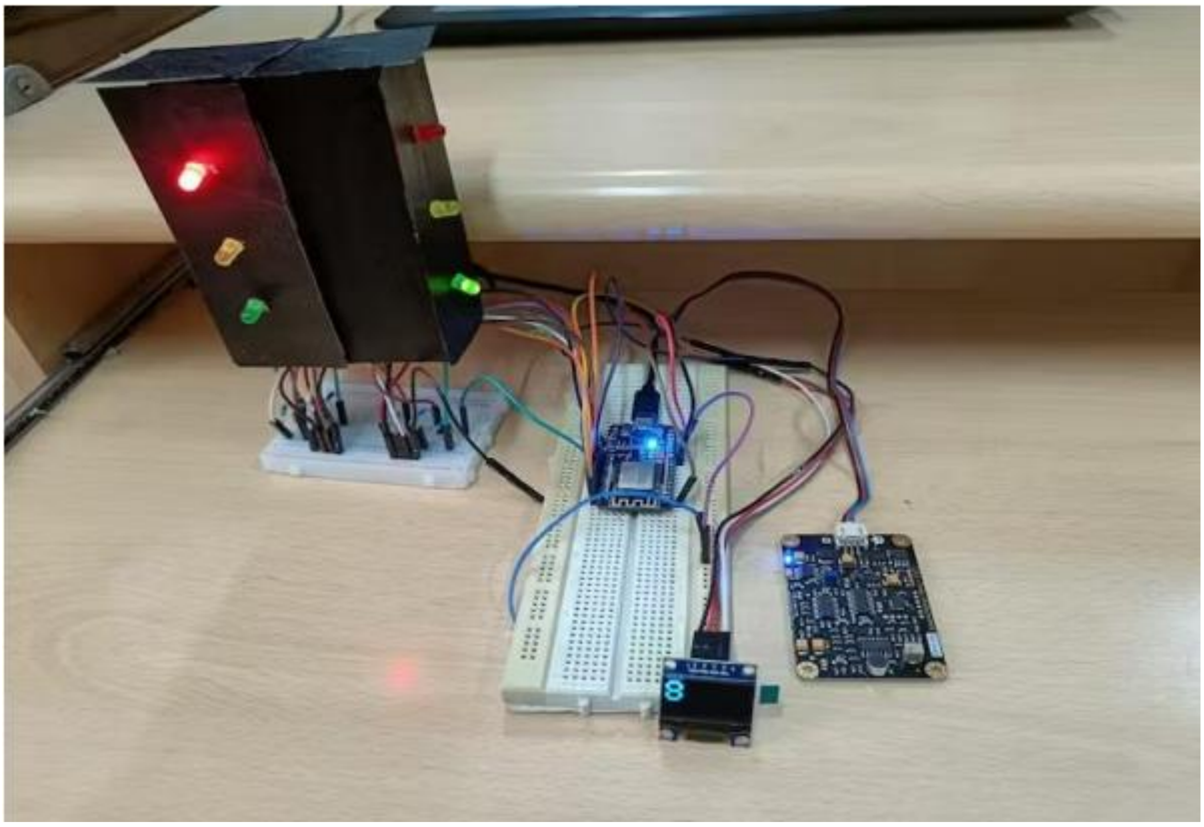
BLOCK DIAGRAM:

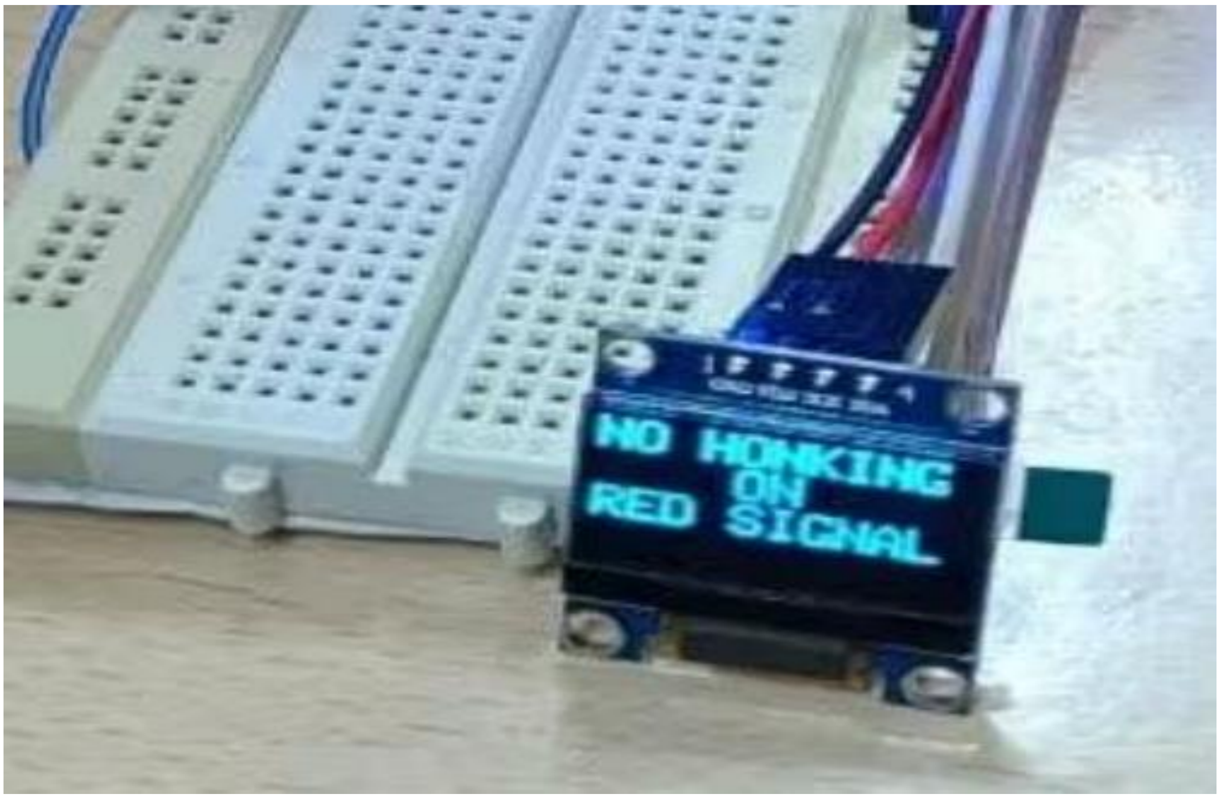


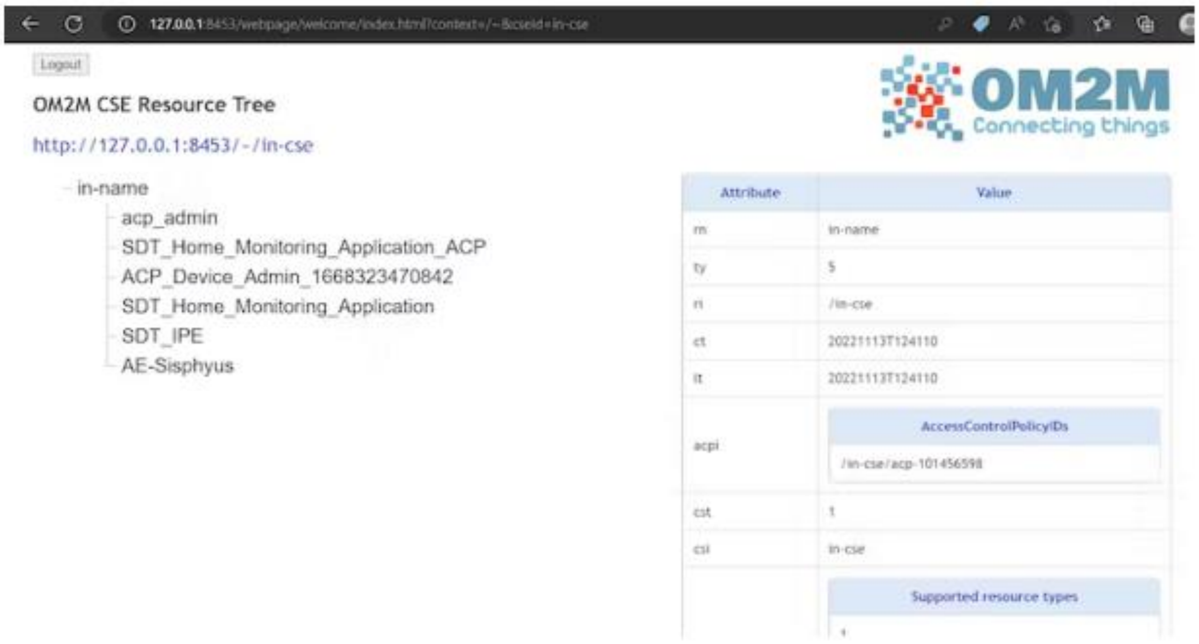
SCHEMATIC


ARDUINO
MQ-7
TITLE:
Air and sound pollution monitoring system
REV: 1.0
Company: Your Company
Sheet: 1/1
EasyEDA
Date: 2021-04-07
Drawn By: Lollipop02




8


NO HONKING
ON
RED SIGNAL


127.0.0.1:8453/webpage/welcome/index.html?context=/~&cseId=in-cse

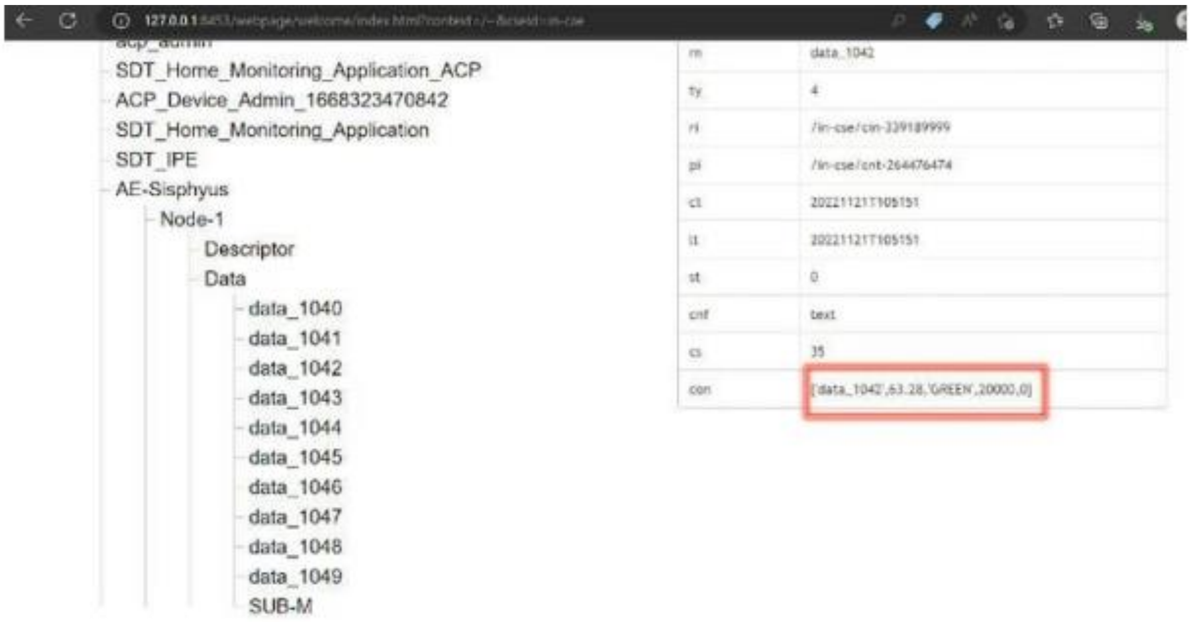
Logout

OM2M CSE Resource Tree

http://127.0.0.1:8453/~/in-cse

- in-name
  - acp_admin
  - SDT_Home_Monitoring_Application_ACP
  - ACP_Device_Admin_1668323470842
  - SDT_Home_Monitoring_Application
  - SDT_IPE
  - AE-Sisphyus

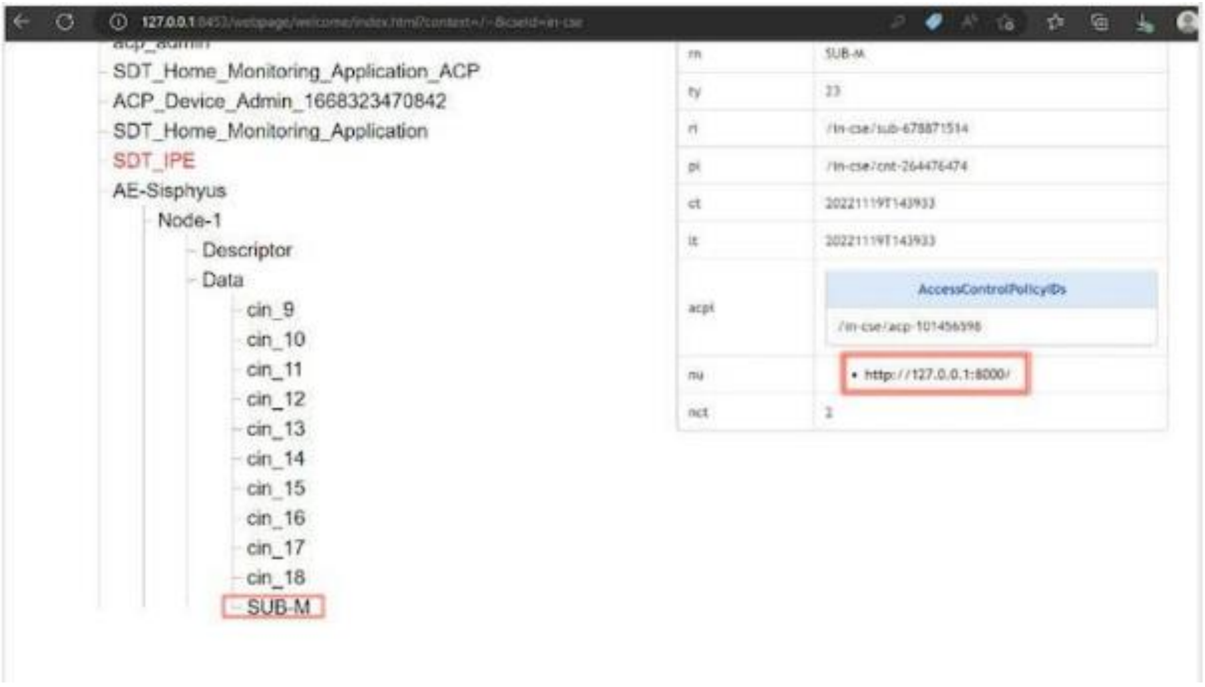
Attribute	Value
rn	in-name
ty	5
ri	/in-cse
ct	20221113T124110
lt	20221113T124110
acpi	AccessControlPolicyIDs: /in-cse/acp-101456598
cst	1
csi	in-cse
	Supported resource types: 1


127.0.0.1:8453/webpage/welcome/index.html?context=/~&cseId=in-cse

- SDT_Home_Monitoring_Application_ACP
- ACP_Device_Admin_1668323470842
- SDT_Home_Monitoring_Application
- SDT_IPE
- AE-Sisphyus
  - Node-1
    - Descriptor
    - Data
      - data_1040
      - data_1041
      - data_1042
      - data_1043
      - data_1044
      - data_1045
      - data_1046
      - data_1047
      - data_1048
      - data_1049
      - SUB-M

rn	data_1042
ty	4
ri	/in-cse/cin-339189999
pi	/in-cse/cnt-264476474
ct	20221121T105151
lt	20221121T105151
st	0
cnf	text
cs	35
con	['data_1042',63.28,'GREEN',20000,0]

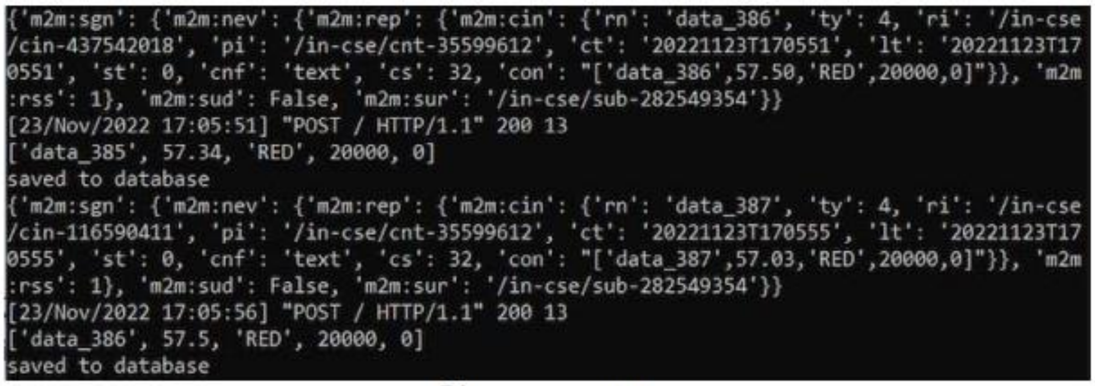
OM2M server and content instances



127.0.0.1:8453/webpage/welcome/index.html?context=/~&cseId=in-cse

```
acp_admin
SDT_Home_Monitoring_Application_ACP
ACP_Device_Admin_1668323470842
SDT_Home_Monitoring_Application
SDT_IPE
AE-Sisphyus
    Node-1
        Descriptor
        Data
            cin_9
            cin_10
            cin_11
            cin_12
            cin_13
            cin_14
            cin_15
            cin_16
            cin_17
            cin_18
            SUB-M
```

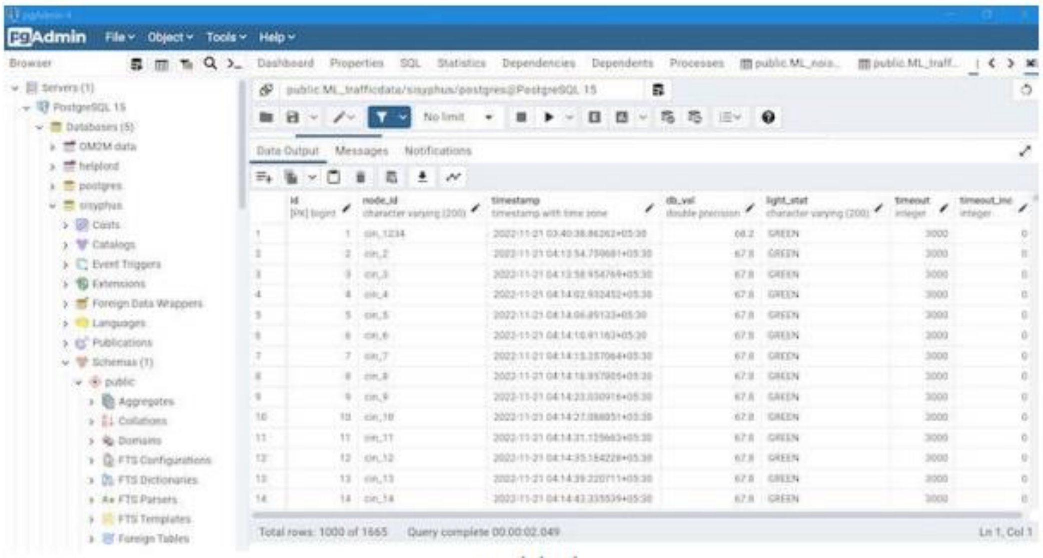
rn	SUB-M
ty	23
ri	/in-cse/sub-678871514
pi	/in-cse/cnt-264476474
ct	20221119T143933
lt	20221119T143933
acpi	AccessControlPolicyIDs /in-cse/acp-101456598
nu	• http://127.0.0.1:8000/
nct	2



```
{'m2m:sgn': {'m2m:nev': {'m2m:rep': {'m2m:cin': {'rn': 'data_386', 'ty': 4, 'ri': '/in-cse
/cin-437542018', 'pi': '/in-cse/cnt-35599612', 'ct': '20221123T170551', 'lt': '20221123T17
0551', 'st': 0, 'cnf': 'text', 'cs': 32, 'con': "['data_386',57.50,'RED',20000,0]"}}, 'm2m
:rss': 1}, 'm2m:sud': False, 'm2m:sur': '/in-cse/sub-282549354'}}
[23/Nov/2022 17:05:51] "POST / HTTP/1.1" 200 13
['data_385', 57.34, 'RED', 20000, 0]
saved to database
{'m2m:sgn': {'m2m:nev': {'m2m:rep': {'m2m:cin': {'rn': 'data_387', 'ty': 4, 'ri': '/in-cse
/cin-116590411', 'pi': '/in-cse/cnt-35599612', 'ct': '20221123T170555', 'lt': '20221123T17
0555', 'st': 0, 'cnf': 'text', 'cs': 32, 'con': "['data_387',57.03,'RED',20000,0]"}}, 'm2m
:rss': 1}, 'm2m:sud': False, 'm2m:sur': '/in-cse/sub-282549354'}}
[23/Nov/2022 17:05:56] "POST / HTTP/1.1" 200 13
['data_386', 57.5, 'RED', 20000, 0]
saved to database
```

Django server output

The following database is created in PostgreSQL.



public.ML_trafficdata/sisyphus/postgres@PostgreSQL 15

	id [PK] bigint	node_id character varying (200)	timestamp timestamp with time zone	db_val double precision	light_stat character varying (200)	timeout integer	timeout_inc integer
1	1	cin_1234	2022-11-21 03:40:38.86262+05:30	68.2	GREEN	3000	0
2	2	cin_2	2022-11-21 04:13:54.759681+05:30	67.8	GREEN	3000	0
3	3	cin_3	2022-11-21 04:13:58.954769+05:30	67.8	GREEN	3000	0
4	4	cin_4	2022-11-21 04:14:02.932452+05:30	67.8	GREEN	3000	0
5	5	cin_5	2022-11-21 04:14:06.89133+05:30	67.8	GREEN	3000	0
6	6	cin_6	2022-11-21 04:14:10.91163+05:30	67.8	GREEN	3000	0
7	7	cin_7	2022-11-21 04:14:15.257064+05:30	67.8	GREEN	3000	0
8	8	cin_8	2022-11-21 04:14:18.957905+05:30	67.8	GREEN	3000	0
9	9	cin_9	2022-11-21 04:14:23.030916+05:30	67.8	GREEN	3000	0
10	10	cin_10	2022-11-21 04:14:27.088051+05:30	67.8	GREEN	3000	0
11	11	cin_11	2022-11-21 04:14:31.125663+05:30	67.8	GREEN	3000	0
12	12	cin_12	2022-11-21 04:14:35.184228+05:30	67.8	GREEN	3000	0
13	13	cin_13	2022-11-21 04:14:39.220711+05:30	67.8	GREEN	3000	0
14	14	cin_14	2022-11-21 04:14:43.335539+05:30	67.8	GREEN	3000	0

Total rows: 1000 of 1665 Query complete 00:00:02.049 Ln 1, Col 1

postgresql database

Working

Imposing the RED signal is done when the noise levels exceed the threshold more than 3 times. This is done by initiating a counter and is based on the frequency of noise levels in dB(A) above the threshold. If it exceeds the threshold count again during the imposition of the RED signal, an additional 10 seconds are added to the wait time. The Traffic Signal is demonstrated by using 3 LEDs which glow corresponding to their interval. Considering an intersection with the main road and a side road crossing the main road the time interval taken for the signal is 20 seconds and the GREEN signal is 30 seconds for the main road and 30 seconds for RED and 20 seconds for GREEN for the side road. Yellow LED will glow on each transition between GREEN to RED for 3 seconds. The countdown timer for each state is shown on an OLED screen. When the noise level above a specified threshold is sensed more than 3 times when the traffic signal is in RED state then 10 seconds is added to the red signal countdown timer as a punishment for excessive and unnecessary honking. If the commuters still honk even after 10 second punishment has been added and the noise levels cross the threshold more than 3 times again another 10 seconds is added to count down the timer as a punishment the second time. We have limited the number of punishments to 3 times. After the maximum number of punishments, the traffic light will go back to green and it will work as the regular traffic light controller.