

Sending data from sensor node to Cloud via Wingz Gateway

-GROUP 7

Members :

IIT2020160 - Anushka Kalwale

IIT2020179 - Karus Manisha

IIT2020193 - Kontham Pavani

IIT2020195 - Inder Sonu

IIT2020217 - Velagana Nagendra

IIT2020089 - Devesh Parte

MWC2022007 - Adarsh Kushwaha

1. Install and Configure Putty

Putty is the application through which can visualize the data being recorded by sensor device. We can also initialize the ip address it will send the data to.

To install Putty :

- Open the terminal
- Run “sudo apt install -y putty”
- Then enter your password if provoked to start the installation process.

```
ubuntu@ubuntu-HP-Pavilion-Laptop-15-eg2xxx:/media/ubuntu/UBUNTU 22_0/Group7/G7_t
erm_project$ sudo apt install -y putty
[sudo] password for ubuntu:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
putty is already the newest version (0.76-2).
The following packages were automatically installed and are no longer required:
 libaio1 libcgi-fast-perl libcgi-pm-perl libevent-core-2.1-7
 libevent-pthreads-2.1-7 libfcgi-bin libfcgi-perl libfcgi0ldbl
 libhtml-template-perl libmecab2 libprotobuf-lite23 mecab-ipadic
 mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0
 mysql-common mysql-server-core-8.0
Use 'sudo apt autoremove' to remove them.
sh: 0: getcwd() failed: No such file or directory
sh: 0: getcwd() failed: No such file or directory
0 upgraded, 0 newly installed, 0 to remove and 225 not upgraded.
sh: 0: getcwd() failed: No such file or directory
sh: 0: getcwd() failed: No such file or directory
```

Then to run the Putty:

- First of all connect the WiFi Mote via USB to your laptop.
- Run “sudo putty” in the terminal.
- A window will open where, set the connection as “Serial”, speed as “115200” and Serial line as “/dev/ttyUSB0”. (The letters after tty can be got through running “ls /dev/tty” and press tab twice, you will see the list of wifi connection and other connection available and then write the connection in which we have connected the USB cord).
- Hit Open and we can see that a black window opens.
- Reset the WiFi Mote.
- Enter the name of wifi you want to connect. In our case it is IIITA.
- Enter the WPA Security associated with the wifi connection. 0 if not password protected, 1 if password protected. (If password protected enter the password).
- Enter the IP address of the device you want to send the data to. (For testing you can enter your own IP address and check whether you are receiving data or not)

2. Store the data in the file

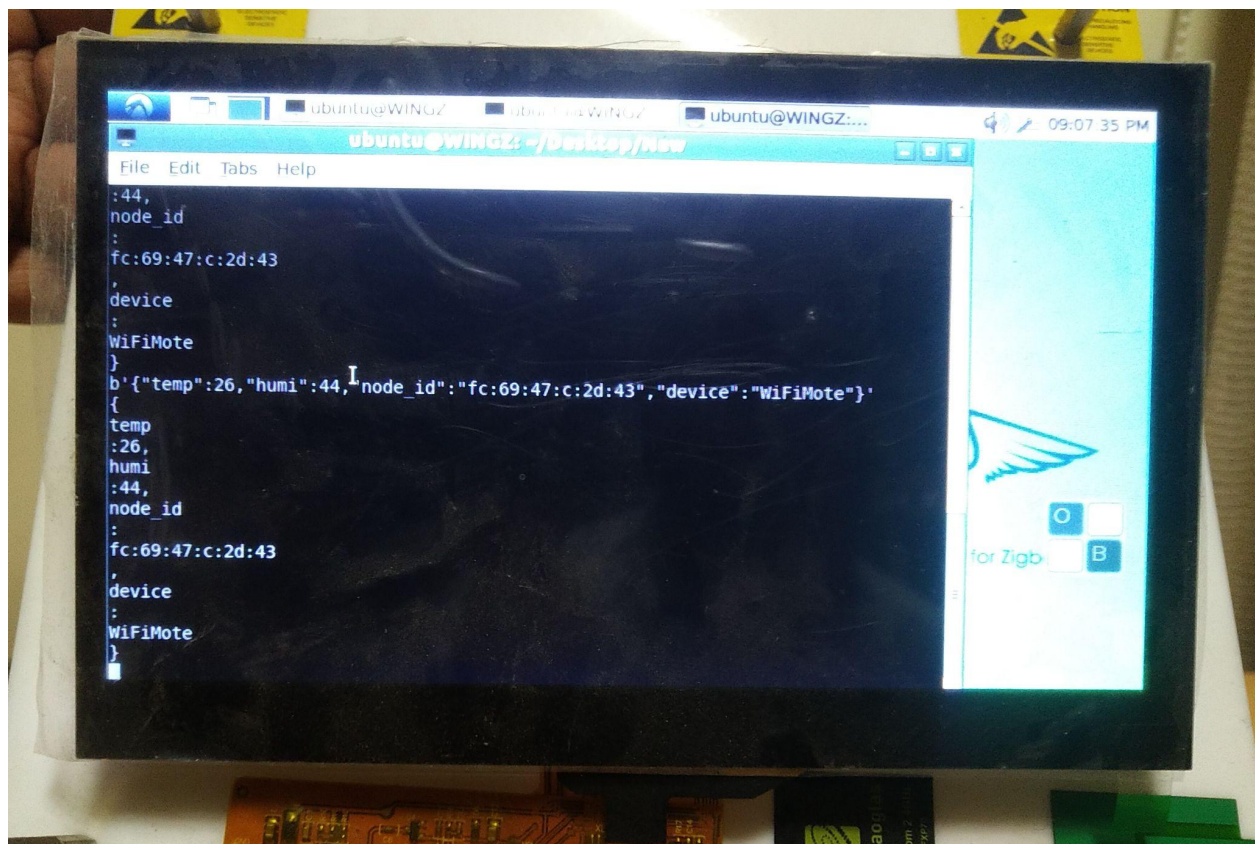
Once putty starts sending data to the Gateway (server). We need to store the data in a file.
For this:

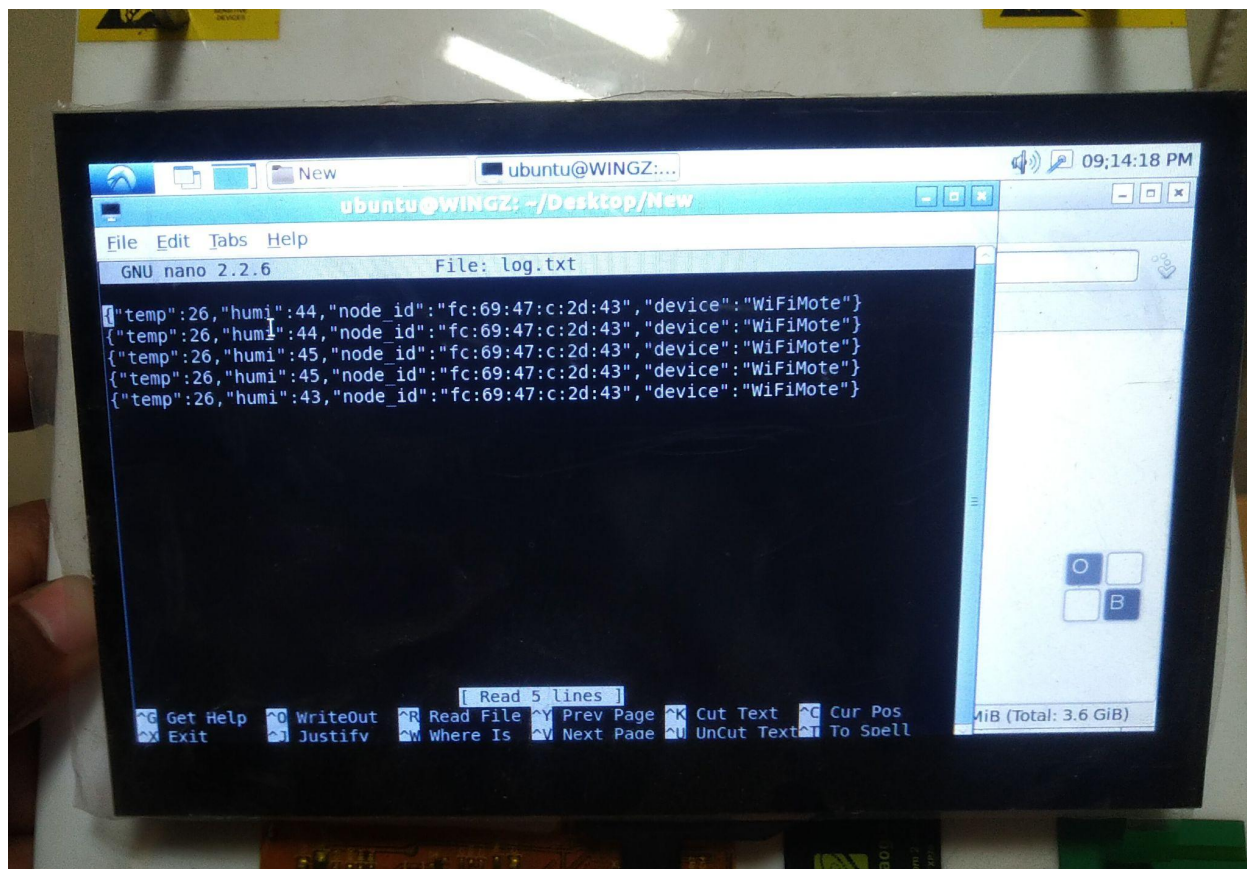
Run “python3 sensorToGateway_G7.py” on the gateway.

Make sure that you have a text file name recordData_G7.txt.

Once you have run the python programming you can see it connects to port 5556 and you can see the data which is being sent to the gateway via WiFiMote and it gets stored in the text file. Moreover we only take 10 entries per minute so added a sleep of 6 sec.

Process of storing the data : Well there will be many WiFiMote sending data to the gateway. So we check in the code if the mac_id through which the data is being sent matches the mac_id of our WiFiMote of our device then only we store it in the txt file.





ubuntu@WINGZ: ~/Desktop/New

File Edit Tabs Help

GNU nano 2.2.6 File: log.txt

```
{ "temp":26,"humi":44,"node_id":"fc:69:47:c:2d:43","device":"WiFiMote"}
{ "temp":26,"humi":44,"node_id":"fc:69:47:c:2d:43","device":"WiFiMote"}
{ "temp":26,"humi":45,"node_id":"fc:69:47:c:2d:43","device":"WiFiMote"}
{ "temp":26,"humi":45,"node_id":"fc:69:47:c:2d:43","device":"WiFiMote"}
{ "temp":26,"humi":43,"node_id":"fc:69:47:c:2d:43","device":"WiFiMote"}
```

[Read 5 lines]

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justifv ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

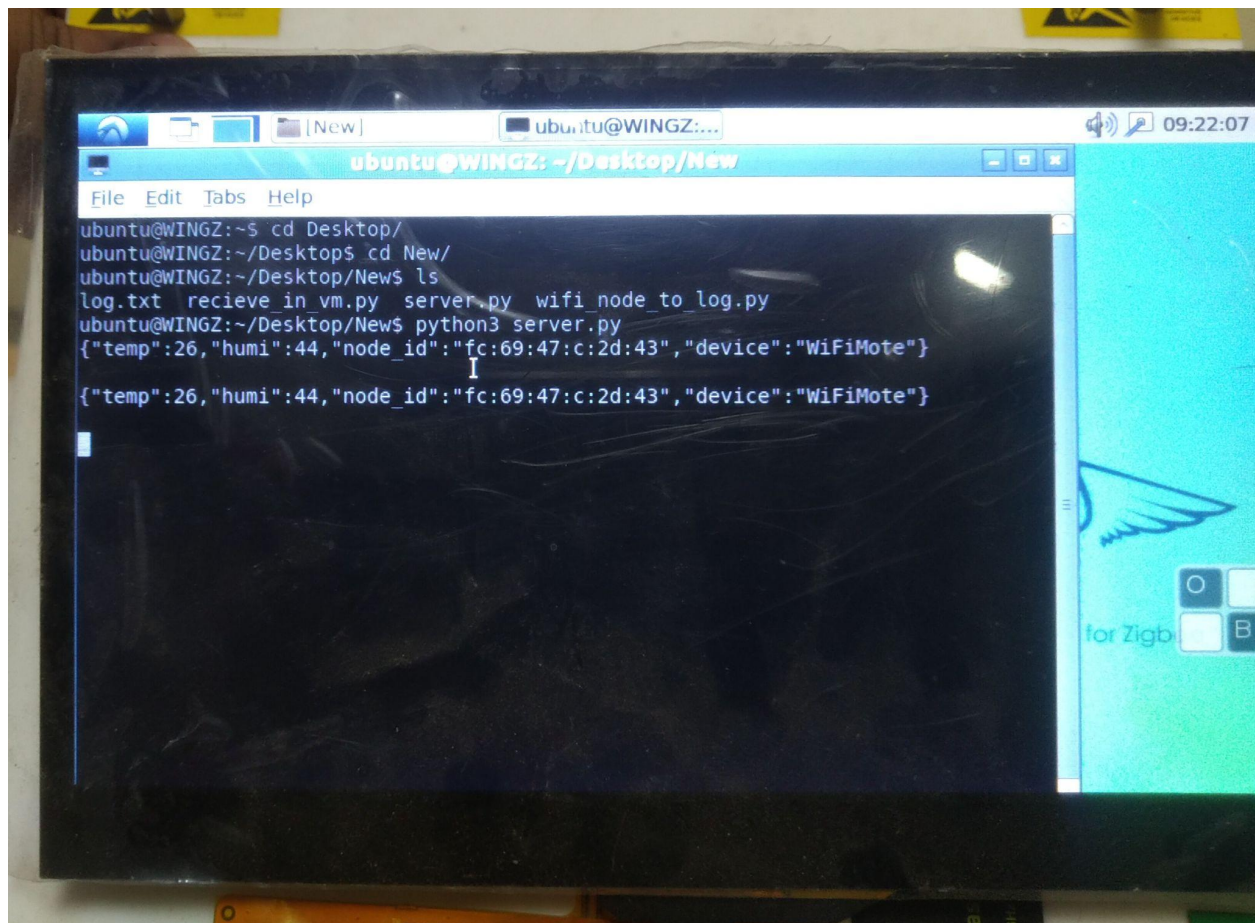
41B (Total: 3.6 GiB)

3. Send the entries from Gateway to the cloud

For this :

Run "python3 Gateway2Cloud_G7.py"

In this file we are accepting connections to the Gateway via different virtual machine. Once we get the connection from the virtual machine, the program starts reading line by line entries from the text file. It reads a line and closes the file instantly to prevent concurrent access. The read line is then sent to the virtual machine immediately. We do this 5 times and then delete the top 5 entries as they have been sent to the virtual machine already. The file is read and closed immediately and then we do the write operation and close the file immediately.



4. Receive the data on the VM and send it to the DataBase:

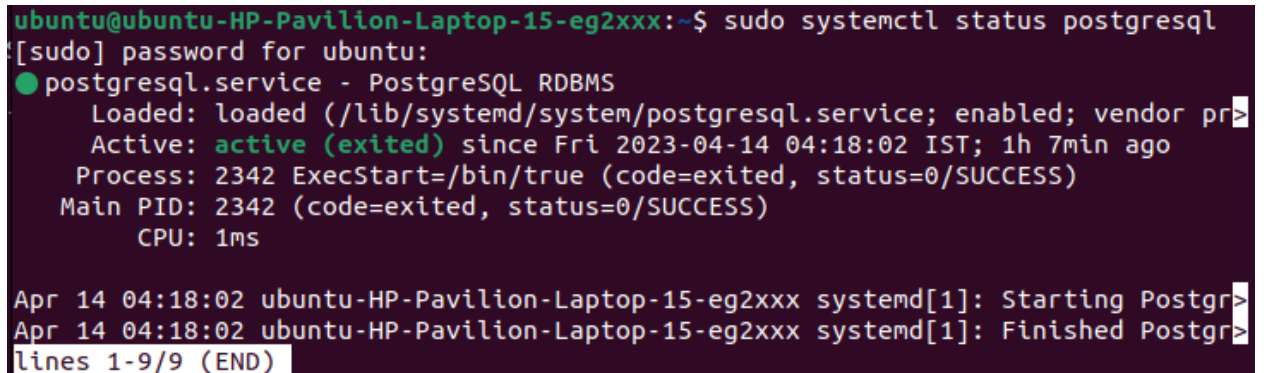
A. Creation of the Database

Install postgresql through command :

`sudo apt-get install postgresql`

To check the installation run :

`sudo systemctl status postgresql`



```
ubuntu@ubuntu-HP-Pavilion-Laptop-15-eg2xxx:~$ sudo systemctl status postgresql
[sudo] password for ubuntu:
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
   Active: active (exited) since Fri 2023-04-14 04:18:02 IST; 1h 7min ago
     Process: 2342 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
    Main PID: 2342 (code=exited, status=0/SUCCESS)
       CPU: 1ms

Apr 14 04:18:02 ubuntu-HP-Pavilion-Laptop-15-eg2xxx systemd[1]: Starting PostgreSQL Database Service: PostgreSQL Database Service (PostgreSQL 15 on Ubuntu 22.04 LTS) using systemd.
Apr 14 04:18:02 ubuntu-HP-Pavilion-Laptop-15-eg2xxx systemd[1]: Finished PostgreSQL Database Service: PostgreSQL Database Service (PostgreSQL 15 on Ubuntu 22.04 LTS) using systemd.
lines 1-9/9 (END)
```

- Creation of user

To create a user run :

`CREATE USER grp_7 WITH PASSWORD 'root';`

With grp_7 being the user name and root being the password

Now to give the permissions to the user inorder to create the databases run :

`ALTER USER grp_7 CREATEDB;`

- Creation of database

So to create a database, run the following command. This database will be used by us further :

`createdb G7`

To check if the database is created run : `\l`

```
(1 row)
G7=> \l
```

List of databases					
Name	Owner	Encoding	Collate	Ctype	Access privileges
G7	postgres	UTF8	en_IN	en_IN	
grp_7	postgres	UTF8	en_IN	en_IN	
postgres	postgres	UTF8	en_IN	en_IN	
template0	postgres	UTF8	en_IN	en_IN	=c/postgres + postgres=CTc/postgres
template1	postgres	UTF8	en_IN	en_IN	=c/postgres + postgres=CTc/postgres
test	postgres	UTF8	en_IN	en_IN	

```
(6 rows)
```

- Creating table with required entries in database
We used the createTable_G7.py to create the following database
Run “python3 createTable_G7.py”

```
G7=> SELECT * FROM DB_G7;
```

id	node_id	device_type	tmp	hmd	dt	tm
1	fc:69:47:c:2d:43	WiFiMote	26	58	2023-04-14	01:05:57
2	fc:69:47:c:2d:43	WiFiMote	26	41	2023-04-14	01:06:12
3	fc:69:47:c:2d:43	WiFiMote	26	42	2023-04-14	01:06:27
4	fc:69:47:c:2d:43	WiFiMote	26	44	2023-04-14	01:06:42
5	fc:69:47:c:2d:43	WiFiMote	26	46	2023-04-14	01:06:57
6	fc:69:47:c:2d:43	WiFiMote	26	48	2023-04-14	01:07:12
7	fc:69:47:c:2d:43	WiFiMote	26	49	2023-04-14	01:07:27
8	fc:69:47:c:2d:43	WiFiMote	26	58	2023-04-14	02:16:40
9	fc:69:47:c:2d:43	WiFiMote	26	41	2023-04-14	02:16:55
10	fc:69:47:c:2d:43	WiFiMote	26	42	2023-04-14	02:17:10
11	fc:69:47:c:2d:43	WiFiMote	26	44	2023-04-14	02:17:26
12	fc:69:47:c:2d:43	WiFiMote	26	58	2023-04-14	02:19:17
13	fc:69:47:c:2d:43	WiFiMote	26	41	2023-04-14	02:19:33
14	fc:69:47:c:2d:43	WiFiMote	26	44	2023-04-14	02:44:30
15	fc:69:47:c:2d:43	WiFiMote	26	44	2023-04-14	02:44:45
16	fc:69:47:c:2d:43	WiFiMote	26	45	2023-04-14	02:45:00
17	fc:69:47:c:2d:43	WiFiMote	26	45	2023-04-14	02:45:15
18	fc:69:47:c:2d:43	WiFiMote	26	43	2023-04-14	02:45:30

```
(18 rows)
```

B. Receive it in the cloud and send to database

For this run : “python3 Cloud2DB_G7.py”

This file first makes a connection with the gateway and then starts receiving the entries.

Once received we take out the value of temperature and humidity and along with the date and time and send it to the database as an entry. The connection to the database is made at the start in the program and that is used to send the data to the database we just created. Also to acknowledge that we are getting the data we print Received once entry is received.

A terminal window with a dark background. The title bar shows 'ubuntu@ubuntu-HP-Pavilion-Laptop-15-eg2xxx: ~/Download...'. The prompt is 'ubuntu@ubuntu-HP-Pavilion-Laptop-15-eg2xxx:~/Downloads/final cec files\$'. The command 'python3 gateway_to_vm.py' has been executed, and the output consists of three lines, each saying 'Recieved' (with a typo).

```
ubuntu@ubuntu-HP-Pavilion-Laptop-15-eg2xxx: ~/Download...
ubuntu@ubuntu-HP-Pavilion-Laptop-15-eg2xxx:~/Downloads/final cec files$ python3
gateway_to_vm.py
Recieved
Recieved
Recieved
```

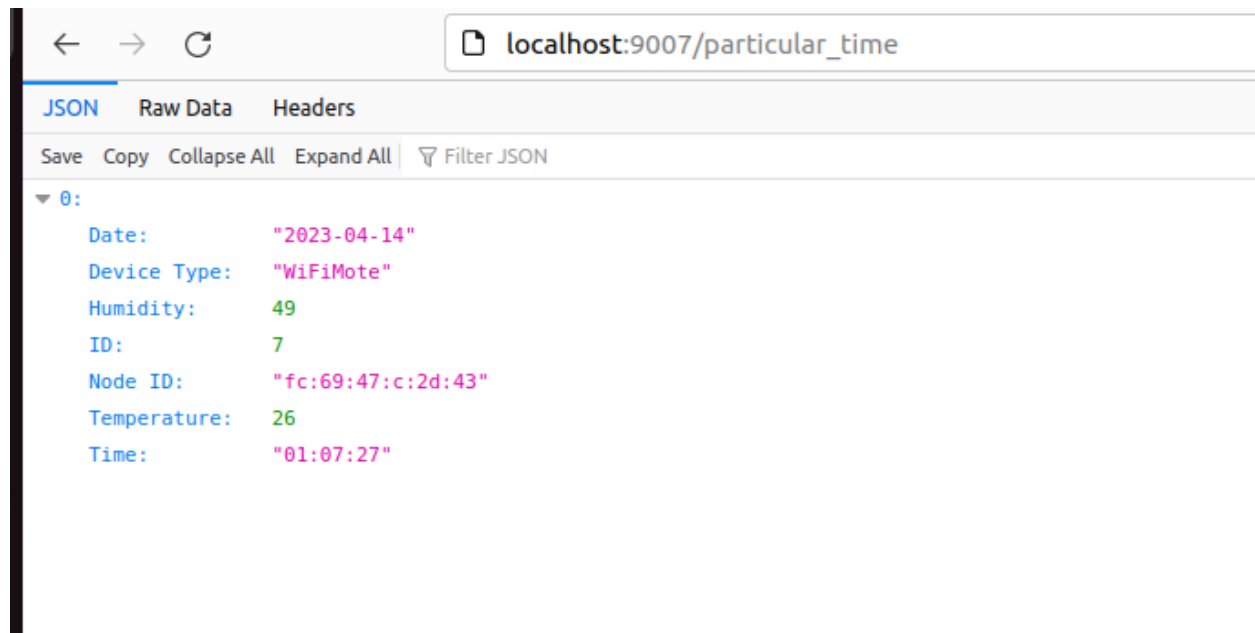
5. Run the web services

For this :

Run “python3 WebServices_G7.py”

This command runs the whole application in the background and now our web services can be accessed by anyone who is connected to our network, in our case all the people connected to IIITA network can access to the web services, just the person need to the link and then can access the data according to the link he/she is visiting.

Of a specific time stamp.



Terminal :

```
ubuntu@ubuntu-HP-Pavilion-Laptop-15-eg2xxx:~$ psql -U test -h localhost -d grp_7
Password for user test:
psql (14.7 (Ubuntu 14.7-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compress
Type "help" for help.

grp_7=> \c G7
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compress
You are now connected to database "G7" as user "test".
G7=> SELECT * FROM DB_G7 WHERE tm='01:07:27';
 id |      node_id      | device_type | tmp | hmd |      dt      |      tm
-----+-----+-----+-----+-----+-----+-----
  7 | fc:69:47:c:2d:43 | WiFiMote   |  26 |  49 | 2023-04-14 | 01:07:27
(1 row)
```

Of specific Column (here Temperature)

			localhost:9007/display_tmp		
JSON			Raw Data		
Save			Copy		
Collapse All			Expand All		
			Filter JSON		
▼ 0:			Temperature: 26		
▼ 1:			Temperature: 26		
▼ 2:			Temperature: 26		
▼ 3:			Temperature: 26		
▼ 4:			Temperature: 26		
▼ 5:			Temperature: 26		
▼ 6:			Temperature: 26		
▼ 7:			Temperature: 26		
▼ 8:			Temperature: 26		
▼ 9:			Temperature: 26		
▼ 10:			Temperature: 26		
▼ 11:			Temperature: 26		
▼ 12:			Temperature: 26		
▼ 13:			Temperature: 26		
▼ 14:			Temperature: 26		
▼ 15:			Temperature: 26		
▼ 16:			Temperature: 26		
▼ 17:			Temperature: 26		

Through Terminal :

To display the temperature column through terminal run : `SELECT tmp FROM DB_G7;`

```
ubuntu@ubuntu-HP-Pavilion-Laptop-15-eg2xxx:~$ psql -U test -h localhost -d grp_7
Password for user test:
psql (14.7 (Ubuntu 14.7-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

grp_7=> \c G7
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
You are now connected to database "G7" as user "test".
G7=> SELECT tmp FROM DB_G7;
 tmp
-----
 26
 26
 26
 26
 26
 26
 26
 26
 26
 26
 26
 26
 26
 26
 26
 26
 26
 26
(18 rows)

G7=> █
```

Larger than or less than a specific criterion

←

→

↺

localhost:9007/show_hmd

JSONRaw DataHeaders

SaveCopyCollapse AllExpand AllFilter JSON

▼ 0:

Date:

"Fri, 14 Apr 2023 00:00:00 GMT"

Humidity:

58

Time:

"01:05:57"

▼ 1:

Date:

"Fri, 14 Apr 2023 00:00:00 GMT"

Humidity:

41

Time:

"01:06:12"

▼ 2:

Date:

"Fri, 14 Apr 2023 00:00:00 GMT"

Humidity:

42

Time:

"01:06:27"

▼ 3:

Date:

"Fri, 14 Apr 2023 00:00:00 GMT"

Humidity:

44

Time:

"01:06:42"

▼ 4:

Date:

"Fri, 14 Apr 2023 00:00:00 GMT"

Humidity:

46

Time:

"01:06:57"

▼ 5:

Date:

"Fri, 14 Apr 2023 00:00:00 GMT"

Humidity:

48

Time:

"01:07:12"

▼ 6:

Date:

"Fri, 14 Apr 2023 00:00:00 GMT"

Humidity:

49

Time:

"01:07:27"

▼ 7:

Date:

"Fri, 14 Apr 2023 00:00:00 GMT"

Humidity:

58

Time:

"02:16:40"

▼ 8:

Date:

"Fri, 14 Apr 2023 00:00:00 GMT"

Humidity:

41

Time:

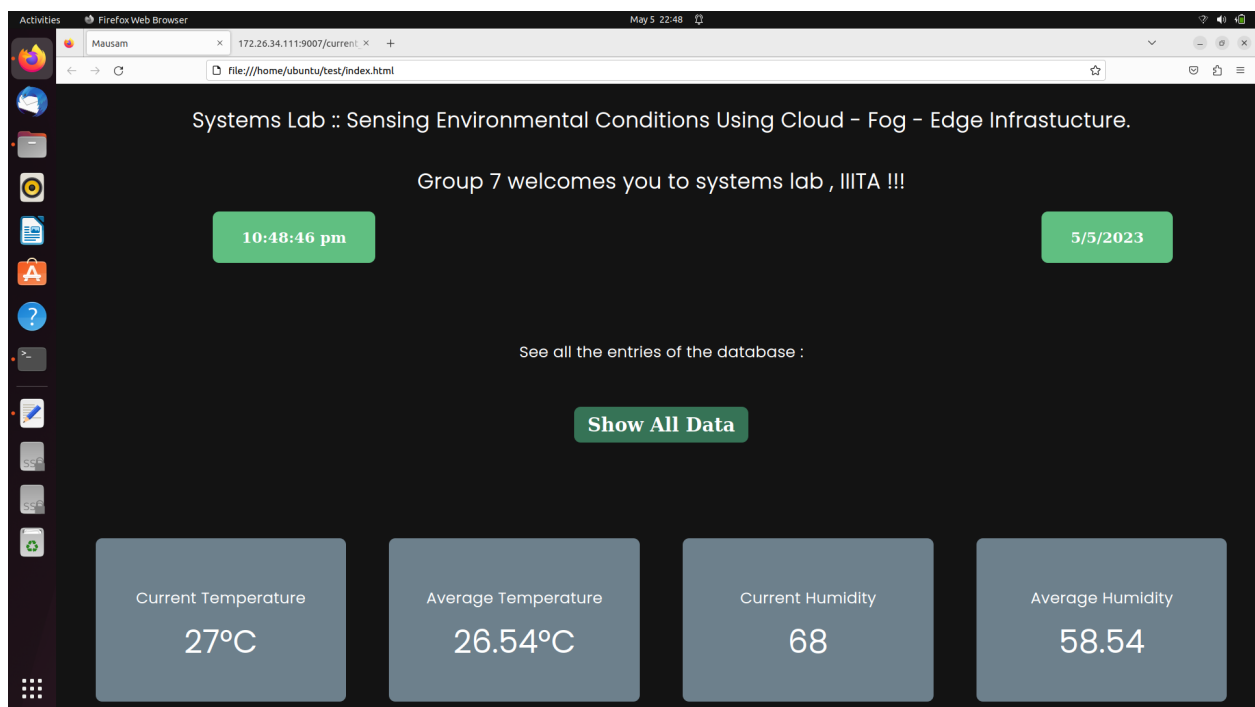
"02:16:55"

6. Running the frontend

To run the frontend, download the html, css and java script files and make sure your web service is running in the background and you are connected to the right api.

Now open the html file through the browser, by doing that you can see your website up and running in the browser.

This is our landing page, where you can see attributes like current and average temperature , current and average humidity alongside the clock.



There is also a button “show all data” present in the landing page of the website, which shows all the data that is collected by our device till date in a tabular format.

See all the entries of the database :

[Hide All Data](#)

ID	Node ID	Device Type	Temperature	Humidity	Date	Time
1	fc:69:47:c2:d4:3	WiFiMote	26	58	4/14/2023	01:05:57
2	fc:69:47:c2:d4:3	WiFiMote	26	41	4/14/2023	01:06:12
3	fc:69:47:c2:d4:3	WiFiMote	26	42	4/14/2023	01:06:27
4	fc:69:47:c2:d4:3	WiFiMote	26	44	4/14/2023	01:06:42
5	fc:69:47:c2:d4:3	WiFiMote	26	46	4/14/2023	01:06:57
6	fc:69:47:c2:d4:3	WiFiMote	26	48	4/14/2023	01:07:12
7	fc:69:47:c2:d4:3	WiFiMote	26	49	4/14/2023	01:07:27
8	fc:69:47:c2:d4:3	WiFiMote	26	58	4/14/2023	02:18:40
9	fc:69:47:c2:d4:3	WiFiMote	26	41	4/14/2023	02:16:55
10	fc:69:47:c2:d4:3	WiFiMote	26	42	4/14/2023	02:17:10
11	fc:69:47:c2:d4:3	WiFiMote	26	44	4/14/2023	02:17:26
12	fc:69:47:c2:d4:3	WiFiMote	26	58	4/14/2023	02:19:17
13	fc:69:47:c2:d4:3	WiFiMote	26	41	4/14/2023	02:19:33
14	fc:69:47:c2:d4:3	WiFiMote	26	44	4/14/2023	02:44:30
15	fc:69:47:c2:d4:3	WiFiMote	26	44	4/14/2023	02:44:45

When you scroll down you will see another section where you will have to enter the start and end date along with start and end time to get the entries in the database that are between those values.

Start Date:

End Date:

Start Time:

End Time:

[Show Data](#)

ID	Temperature	Humidity	Date	Time
1	26	58	4/14/2023	01:05:57
2	26	41	4/14/2023	01:06:12
3	26	42	4/14/2023	01:06:27
4	26	44	4/14/2023	01:06:42
5	26	46	4/14/2023	01:06:57
6	26	48	4/14/2023	01:07:12
7	26	49	4/14/2023	01:07:27
8	26	58	4/14/2023	02:18:40
9	26	41	4/14/2023	02:16:55
10	26	42	4/14/2023	02:17:10
11	26	44	4/14/2023	02:17:26