# 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

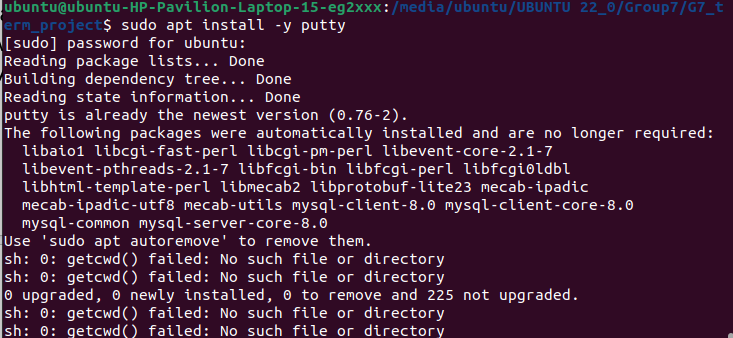
IIT2020089 - Devesh Parte

MWC2022007 - Adarsh Kushwaha

**1. Install and Configure Putty**

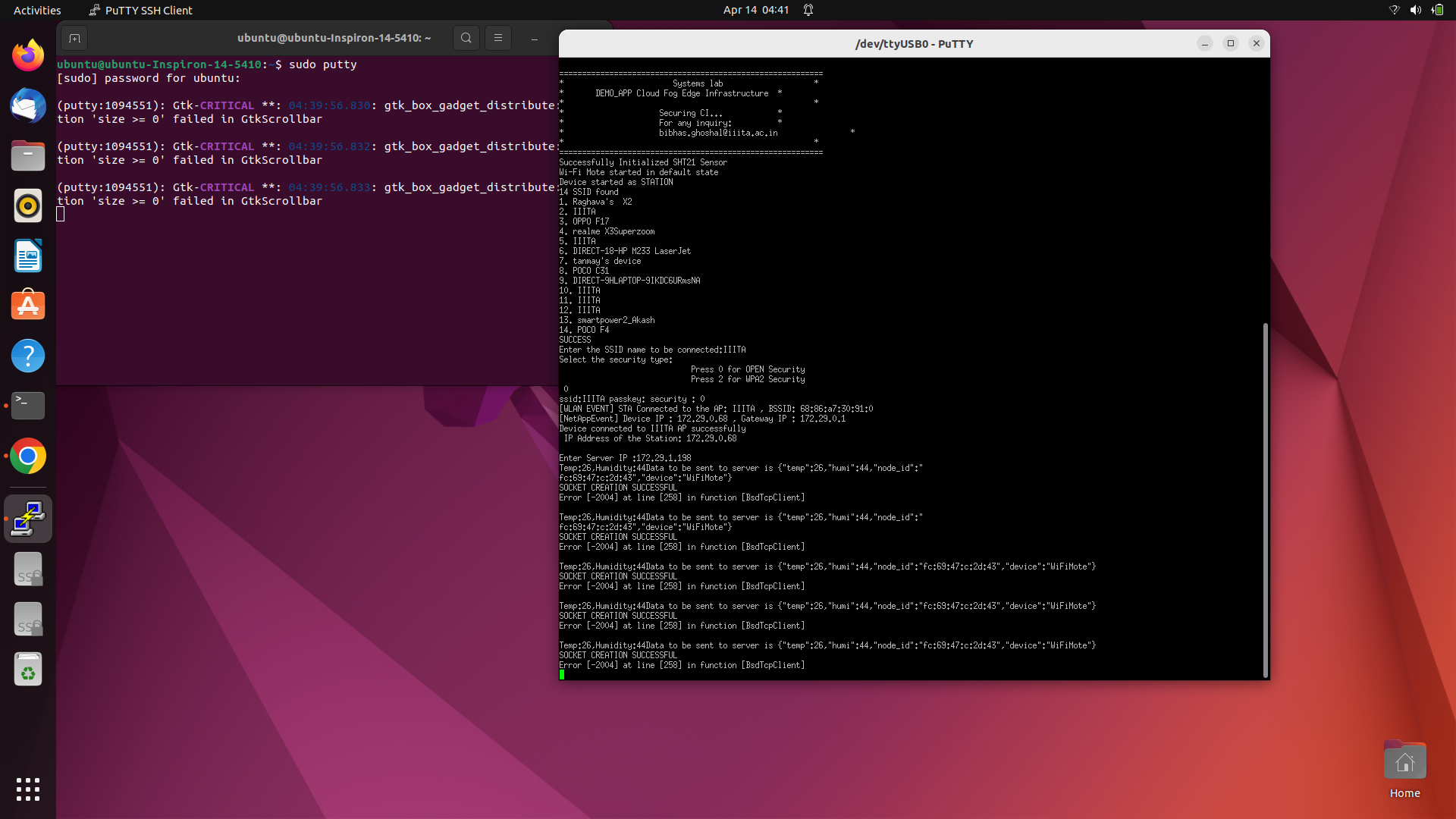
Putty is the application through which can visualize the data being recorded by sensor device. We can also initialize the ip address it will send the data to.  
To install Putty :

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



Then to run the Putty:

* First of all connect the WiFiMote 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 WifiMote.
* 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:

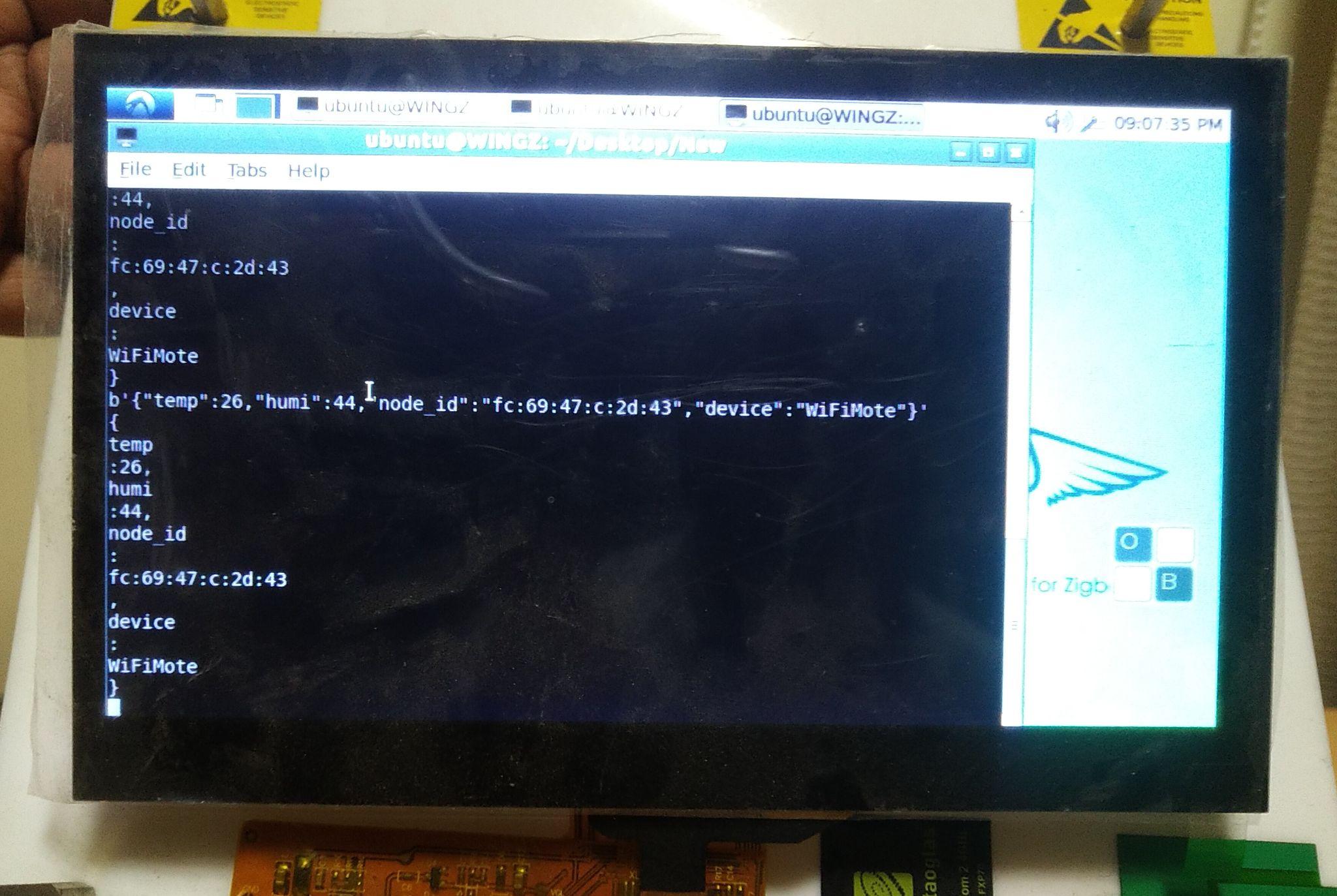
Run “python3 sensorToGateway\_G7.py” on the gateway.

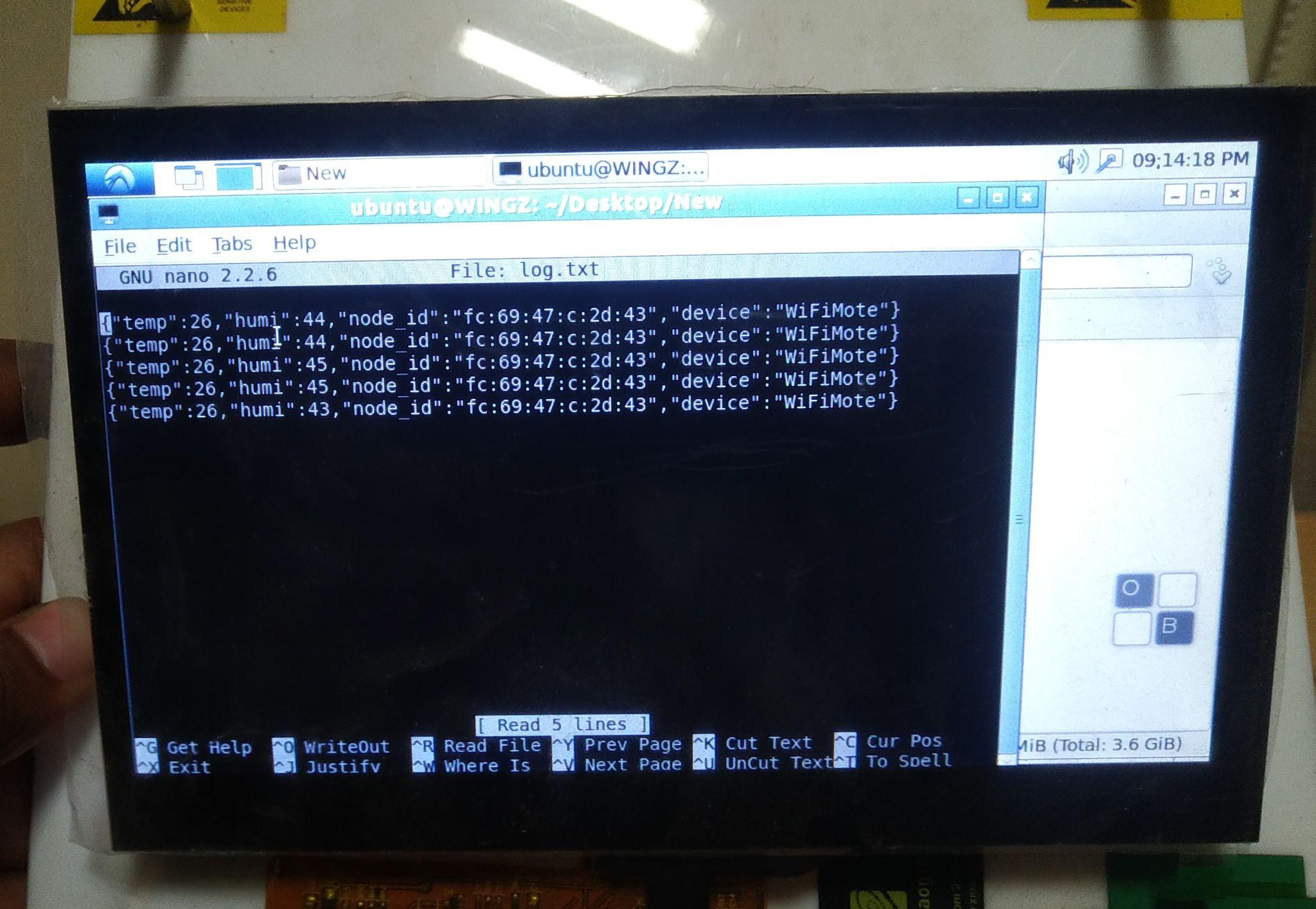
Make sure that you have a text file name recordData\_G7.txt.

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.



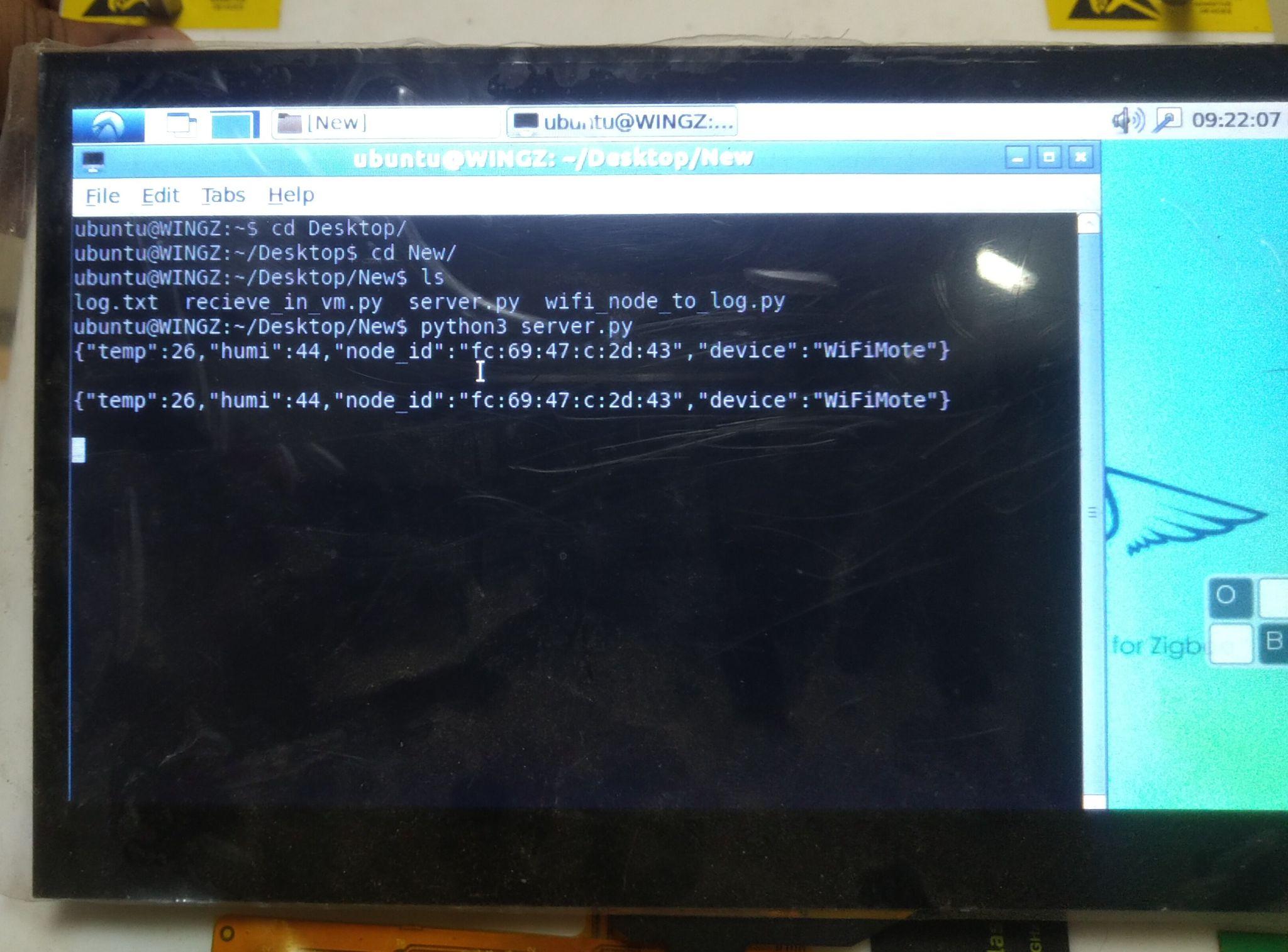


**3. Send the entries from Gateway to the cloud**

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:**

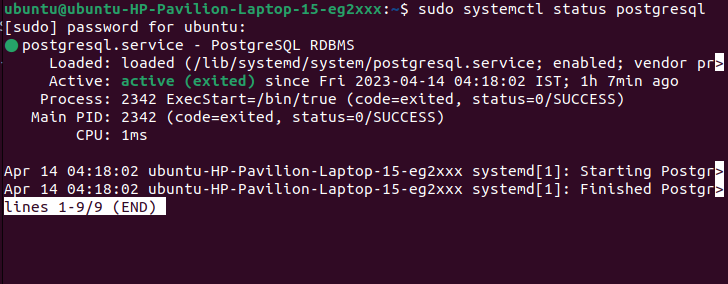
1. **Creation of the Database**

Install postgresql through command :

sudo apt-get install postgresql

To check the installation run :

sudo systemctl status postgresql



* 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 :

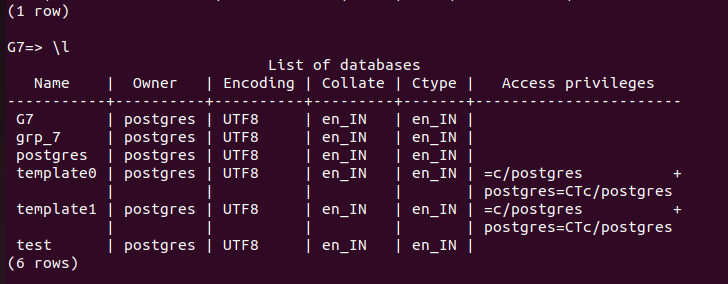
ALTER USER grp\_7 CREATEDB;

* Creation of database

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

createdb G7

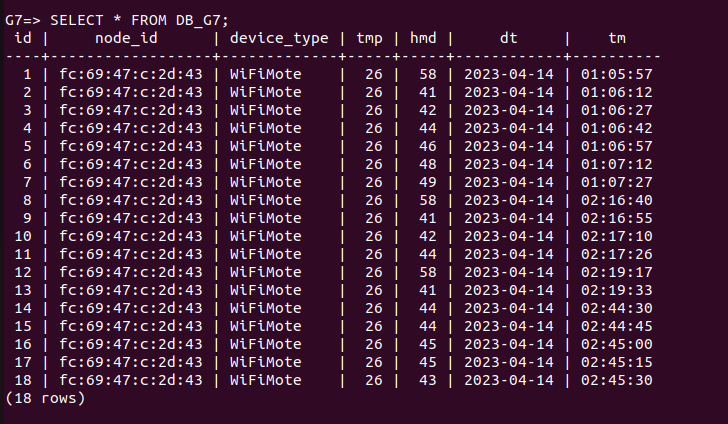
To check if the database is created run : \l



* Creating table with required entries in database

We used the createTable\_G7.py to create the following database

Run “python3 createTable\_G7.py”

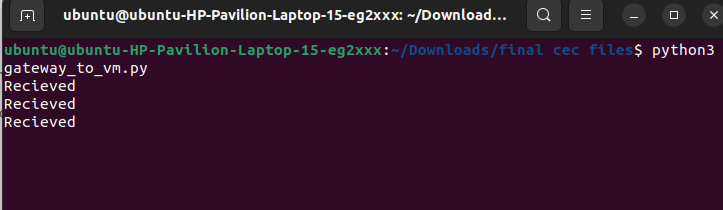


1. **Receive it in the cloud and send to database**

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.



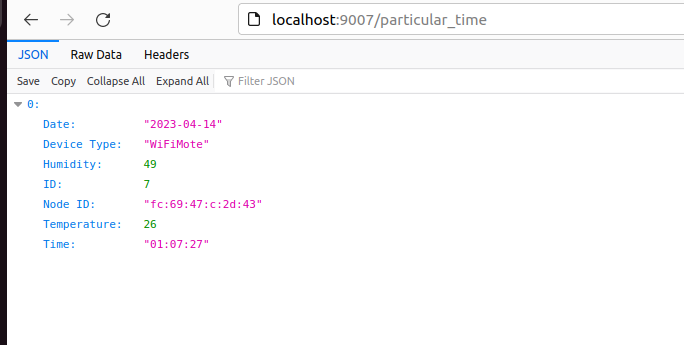
**5. Run the web services**

For this :

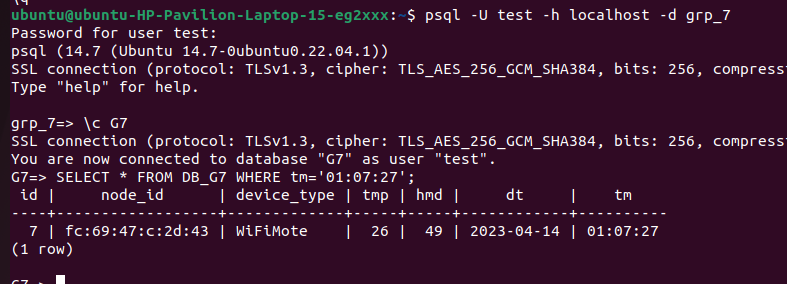
Run “python3 WebServices\_G7.py”

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 :

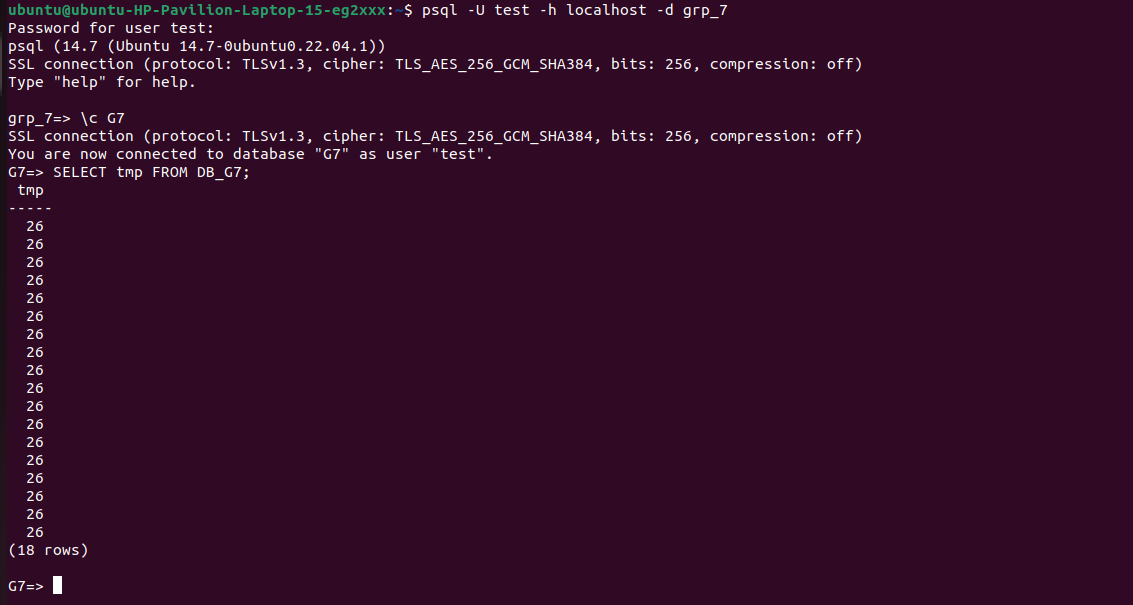


Of specific Column (here Temperature)

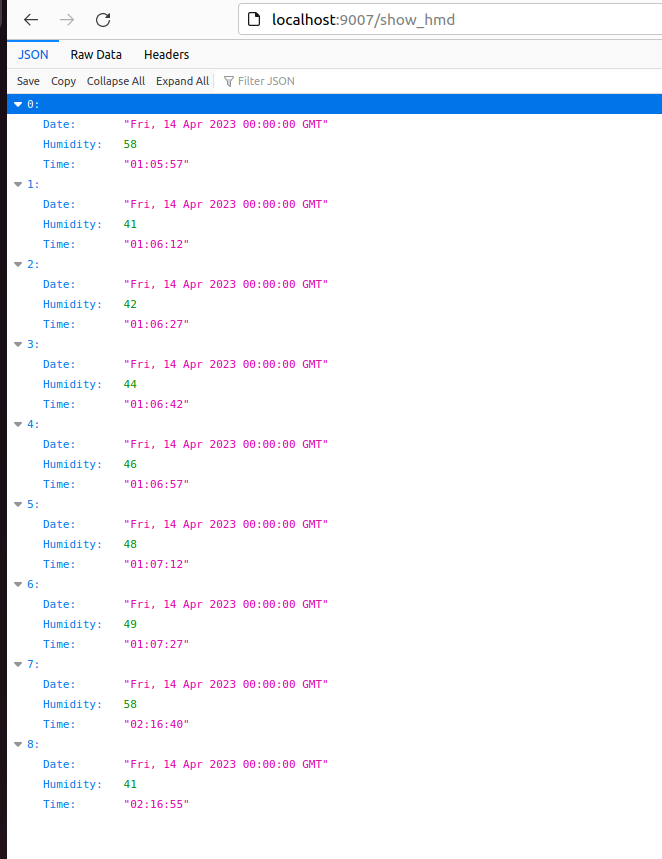


Through Terminal :

To display the temperature column through terminal run : SELECT tmp FROM DB\_G7;

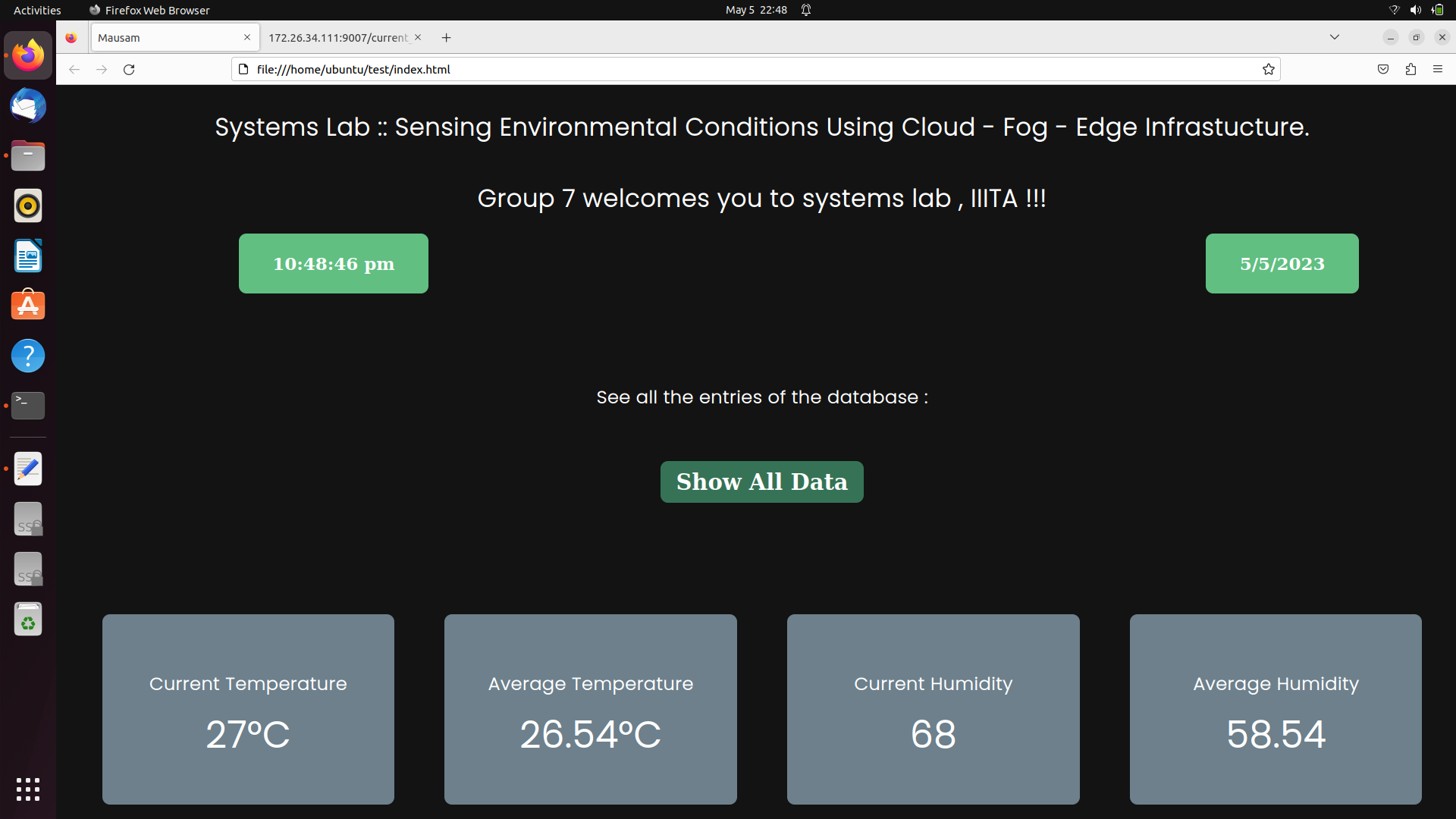
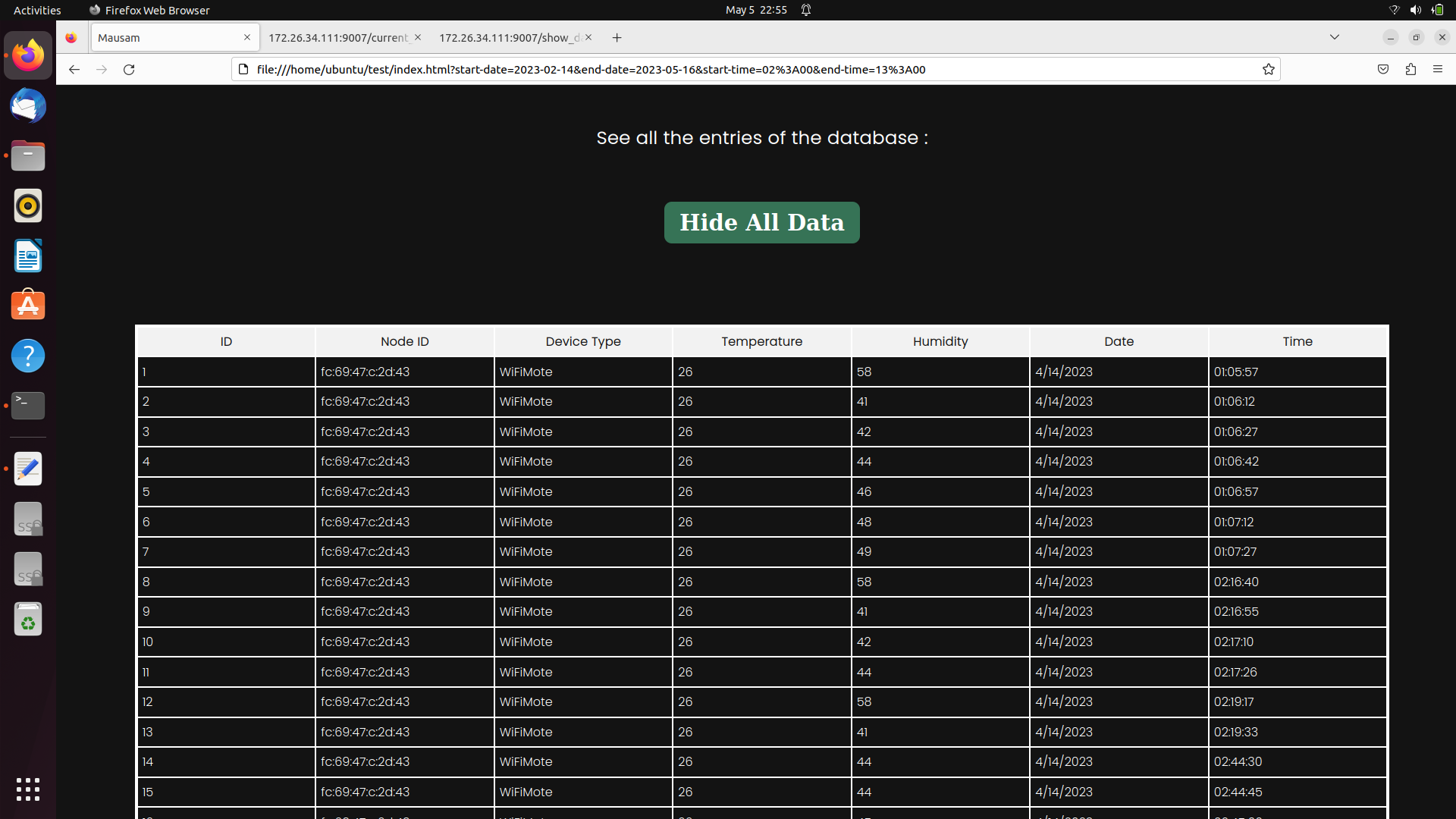


Larger than or less than a specific criterion



**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.  
  
  
  
  


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.   
  
