HLD AND LLD DOCUMENT

IOT DATA PROCESSOR

Sprint – 1

Project Timeline: 7-12-2022 to 12-12-2022

INDEX

Contents

1. Introduction

Data loggers Processing is used to measure the environmental report of a particular city. The IoT data logger is used to measure and store environmental data such as temperature, humidity, illuminance, CO content, PM2.5, and PM10 contents. The climate IoT data logger also calculates the dew point, absolute humidity and the water vapor content in the air.

Data logger are electronic devices which automatically monitor and record environmental parameters over time, allowing conditions to be measured, documented, analysed and validated. Data Loggers are used in geotechnical monitoring and instrumentation on a large scale because they automatically collect real-time data from the sensors and display it over the devices. A data logger is an electronic device that records data (temperature, pressure, impact, humidity, etc.) at specific intervals.

CSV files are used for collecting different data of a particular city which consist of information about temperature, humidity, PM2.5, PM10, NO2. This information is for each day over a year and also it shows the averages of all the parameters of every month for a particular city. The overall strike of the project is it will also show the graphical representation report of a parameter change in a city over a year. It will also show the pollution level for a particular month and date. The data is processed through the CSV file and the data logger which work as a sensor for receiving the information through the csv file**.**

1.1. Intended Audience

This system is intended to read and process by any user to get the information about the climatic condition of a particular city. Our intended clients will be people who are deals with meteorology, government organisations, NGOs who work for the betterment of our environment, environmentalists, etc.

1.2. Acronyms/Abbreviations

IOT	Internet of Things
PM	Particulate Matter
NO2	Nitrogen Dioxide
CSV	Comma-Separated Values

1.3 Project Purpose

- The purpose of this software or project is to show the data files from different cities based on date.
- Data logging means to record events, observations, or measurements systematically. It consists of the following data like temperature, rainfall, humidity, PM2.5, PM10, N02 recordings date wise for a period of a year.
- This project will also show graphical report of a parameter changes in a city over a year.
- The user can also find the pollution level or temperature for a particular month. The Data Logger module allows you to save data in form of CSV file.
- You can create and configure file according to your needs and monitor them while they are being logged.

1.4 In Scope

This project aims to monitor environmental behavior using data loggers. Data loggers can record and transmit precise temperature, humidity, and pressure data so that an organization can keep a close eye on the environment of a given area. Data logger are implicitly stand-alone devices, this stand-alone aspect of data loggers implies onboard memory that is used to store acquired data. Sometimes this memory is very large to accommodate many days, or even months, of unattended recording. Some equipment log data 24 hours a day and 365 days of year while some logs data only during its predefined period. Once it is setup, it performs data logging automatically and does not require presence of human beings. It is very accurate as likelihood of human error is not there. In future, other climatic parameters such as wind, light intensity etc. can be included for analysis and prediction purpose and this system can therefore be used for analysis and prediction of such parameters in remote areas.

1.5 Out of scope

LIMITATIONS:

- Disparate Data and Data Storage Limitations
- If data logging equipment malfunctions, some data could be lost or will not be recorded.
- Certain data logging equipment can take readings only during interval configured at the start.
- Basic training is needed to use the equipment.

1.6 Functional Overview

menu():

The user is provided with various options and he/she can choose their desired option as per their requirement. It is the first interface that will be seen by client.

loadFromFile()

The process of obtaining raw data from a source and replicating that data somewhere else is called Data Extraction. Here we are using CSV files to store values that need to be processed. This function will help to load that file and perform tasks like fgets() to get lines from CSV file and perform string tokenization and create structure by using those tokens in the struct we created. We have to allocated memory for struct also using calloc() or malloc().

appendLinkedList()

This function is used to append the structure created in loadFromFile() function. We have to be sure to increment the location pointer of the structure.

avgTemperature():

The average function calculates the arithmetic mean of a series of temperature. That is, it adds all the values of the variable we are analysing and divides them by the number of values added. The user will select a city and get the averages of different parameters regarding environment. This average information consists of the averages of every month in a year. Also it will show the graphical comparison of the average temperatures of all the months over a year.

avgHumidity():

The average function calculates the arithmetic mean of a series of humidity. That is, it adds all the values of the variable we are analysing and divides them by the number of values added. The user will select a city and get the averages of different parameters regarding environment. This average information consists of the averages of every month in a year. Also it will show the graphical comparison of the average humidity of all the months over a year.

avgPM25():

The average function calculates the arithmetic mean of a series of Particulate Matter 2.5. That is, it adds all the values of the variable we are analysing and divides them by the number of values added. The user will select a city and get the averages of different parameters regarding environment. This average information consists of the averages of every month in a year. Also it will show the graphical comparison of the average PM 2.5 of all the months over a year.

avgPM10():

The average function calculates the arithmetic mean of a series of Particulate Matter 10. That is, it adds all the values of the variable we are analysing and divides them by the number of values added. The user will

select a city and get the averages of different parameters regarding environment. This average information consists of the averages of every month in a year. Also it will show the graphical comparison of the average PM 10 of all the months over a year.

avgNO2():

The average function calculates the arithmetic mean of a series of Nitrogen Dioxide. That is, it adds all the values of the variable we are analysing and divides them by the number of values added. The user will select a city and get the averages of different parameters regarding environment. This average information consists of the averages of every month in a year. Also it will show the graphical comparison of the average Nitrogen Dioxide of all the months over a year.

1.7 Assumptions & Dependencies

- The system should have any Distribution of Linux installed.
- The Proposed System is intended to work on Terminal CUI.
- The system should have either 4GB or more RAM.
- The service is used preferably on a desktop or laptop.

2. Design Overview

The IoT data logger processor is used for calculating temperature and humidity's in the air, in our project, we have taken temperature, humidity, PM2.5, PM10 and NO2 content. Of Delhi city through our project, we can calculate average temperature, humidity PM2.5, PM10 and NO2 content after the evaluation of all the contents Delhi Government can control the pollution, maintenance, transportation, etc., and the benefits of using such IOT data logger's processor are beyond doubt. Some of them can be integrated into systems, others can be installed in the area to be monitored or in the vehicle and after the monitoring time has expired, the data on the device can be called up and of course viewed on the computer - IOT data loggers, on the other hand, can be queried remotely at any time and installation is usually very easy, as it is wireless and fast.

The examples of IoT data logger's processor can be found in every area of life, not necessarily just in business, industry, or agriculture. The term "smart", which has always been applied to people, is increasingly being applied to things, especially the sensors that measure each of the desired parameters, monitor the state and can later "report" on them to the user, regardless of whether they are information about temperature and humidity, soil quality, moisture content, number of people, noise level and many, many other parameters.

2.1 Design Objectives

Instant getting information about the environment details of a particular city through the csv file. It is used to measure and store environment data such as temperature, humidity etc.

There are primary and secondary objectives behind this project.

Primary:

The user will get the averages of different parameters regarding environment of a particularity including the date and the city name. This average information consists of the averages of every month in a year. It is used to measure and store environment data.

The parameters include the following: -

- CITY
- AREA CODE
- DATE
- TEMPERATURE
- HUMIDITY
- PM2.5
- PM10
- NO2

Secondary:

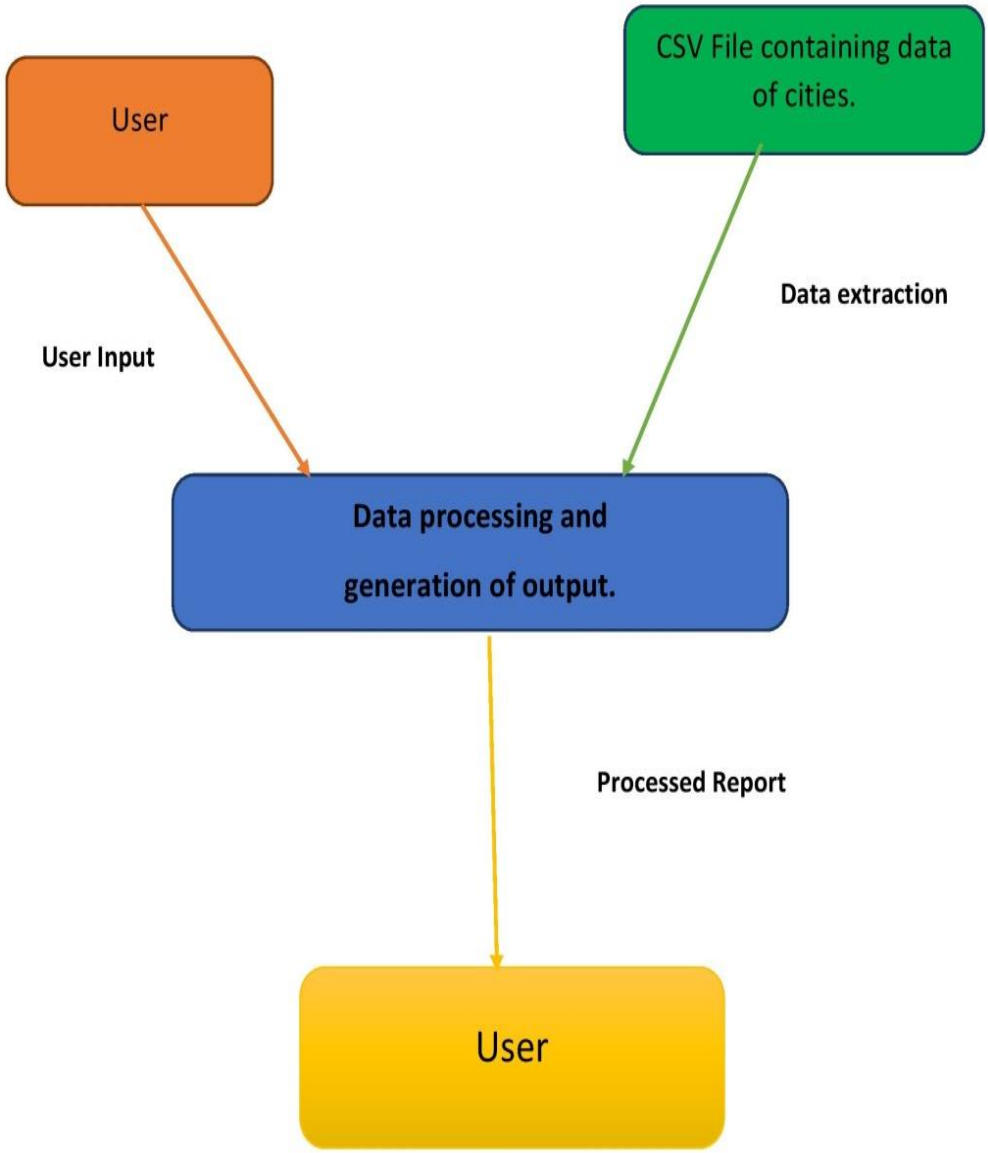
The user will get the average of the desired parameter which they want and also can see the whole date that is present in the CSV file.

2.2 Recommended Architecture (DATA FLOW DIAGRAM)

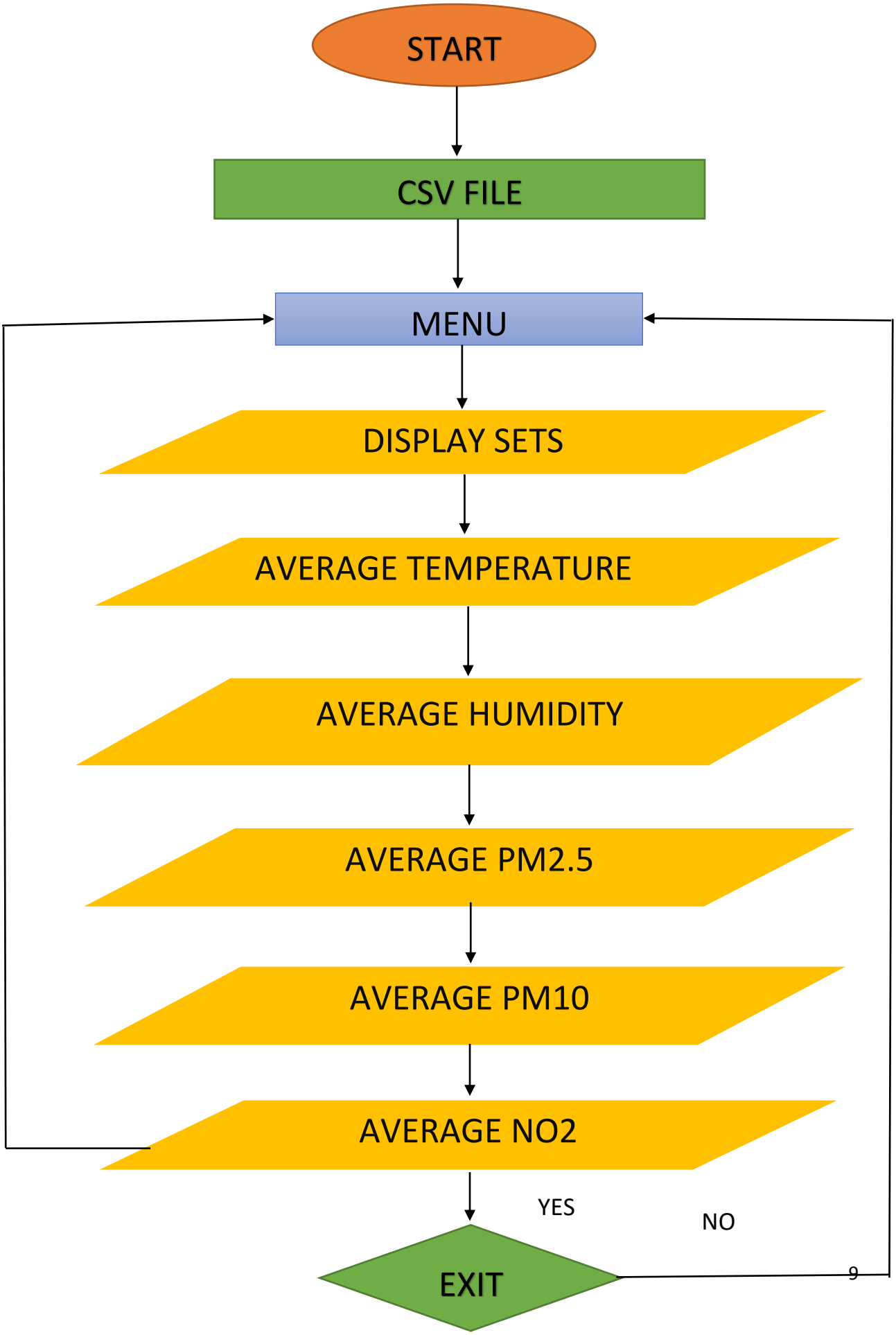
The DFD or Data Flow Diagram maps out the flow of information for the given software.The given DFD diagram depicted the flow of data and information of the proposed system.

1 DFD Level 0 –

On the next page is the DFD level 0 diagram of given software.


User
CSV File containing data of cities.
User Input
Data extraction
Data processing and generation of output.
Processed Report
User

2.3 PROCEDURAL FLOW DIAGRAM: -



3. SYSTEM FEATURE AND REQUIREMENTS

3.1 Functionality:

3.1.1 Load CSV FILE :- The csv file is first loaded in the system and is being used for the processing , The data to be processed is to be measured and compared by the CSV File

3.1.2 Display Parameters :- The parameters are then Displayed which the user wants to search . The user can choose any of the parameters like temperature , humidity , PM2.5,PM10 and O2. It will show the the choices to be entered by the user which they want to know.

3.1.3 Average Temperature:- This function shows the average temperature of any month in a year for a particular city. The average temperature is displayed as an output.

3.1.4 Average Humidity:- This function shows the average Humidity of any month in a year for a particular city. The average Humidity is displayed as an output.

3.1.5 Average PM2.5:- This function shows the average PM2.5 of any month in a year for a particular city. The average PM2.5 is displayed as an output.

3.1.6 Average PM10:- This function shows the average PM10 of any month in a year for a particular city. The average PM10 is displayed as an output.

3.1.7 Average NO2:- This function shows the average NO2 of any month in a year for a particular city. The average NO2 is displayed as an output.

3.1.8 Final Report:- At last, The final report will be given .The final report will include all the details searched by the user for separate parameters . The reports are generated in a graphical manner.

3.2 Performance:-

The system will work on the client's terminal. The performance will depend on the hardware component of the user's system.

3.3 Maintenance:-

Very little maintenance is required for this setup. Only the logging takes space. It is reliable and low cost and time efficient monitoring solution for any measuring opportunity .It is of high accuracy, easy to use and greater versatility in every applications.

3.4 System Requirements:

3.4.1 Data model

- Legacy system data model
- Proposed system data model
- Interface data model

3.4.2 Time Zone Support

- IST Kolkata (GMT + 5:30)

3.4.3 Language Support

- English

3.4.4 User Desktop Requirements

- 64-bit processor, 1 GHz or faster
- At least 2 GB free hard drive space
- At least 1 GB RAM

3.4.5 Database Server Disk Space

- No such disk space is required as the program is fully functional on online IDE(s) as well.
- The Local Operating System is required and CSV text files to store the records that need to be processed.

3.4.6 Integration Requirements

- Language: C
- Tools: Valgrind, Make
- Complier: gcc
- Operating System: Linux Environment, Ubuntu 22.04

3.5 Configuration:

3.5.1 Operating System

- Linux Environment, Ubuntu Version 22.04

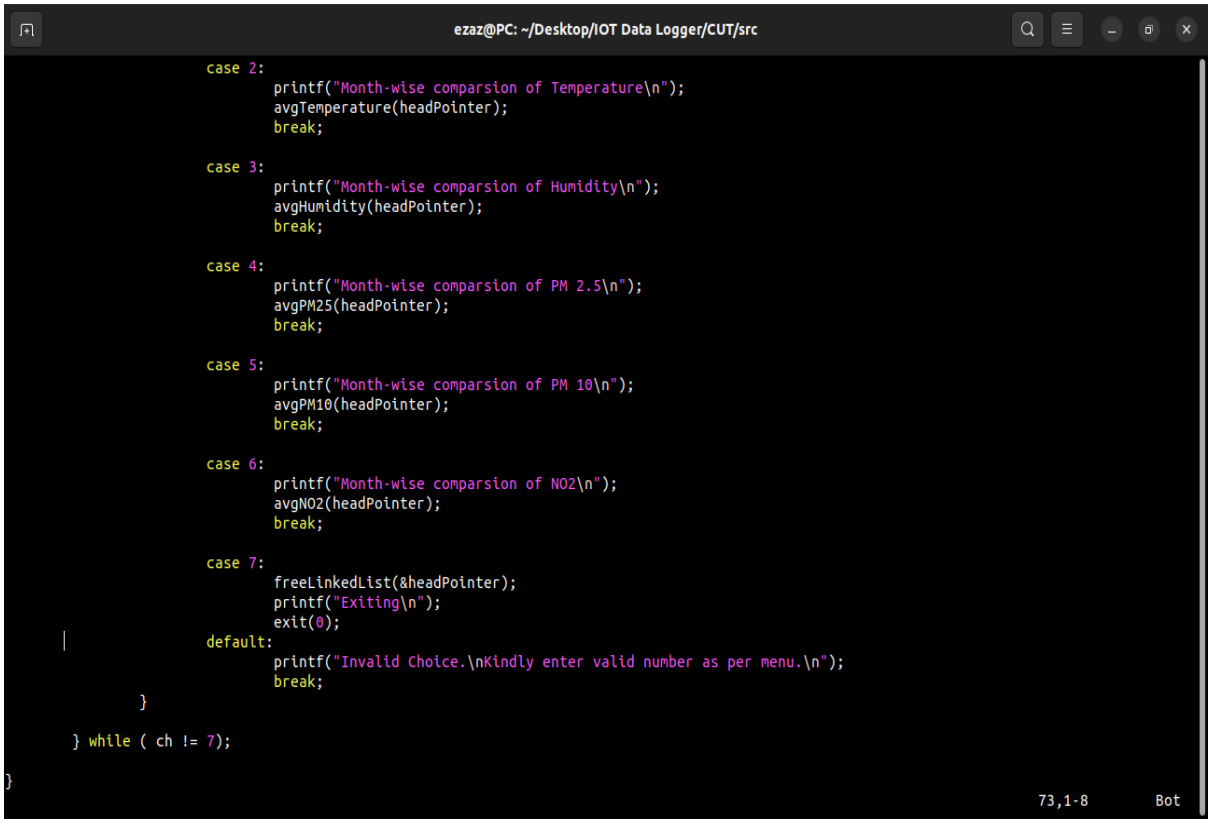
3.5.2 Database

- CSV FILE

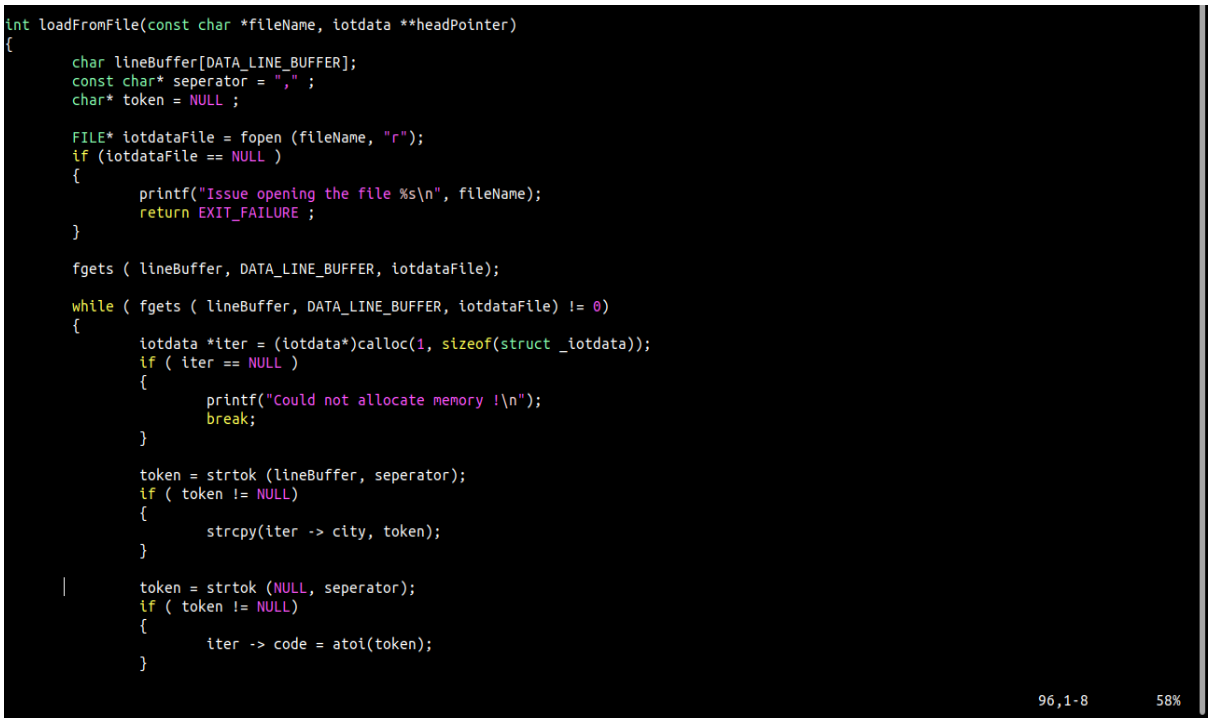
4. FUNCTIONS:-

4.1 menu() :

```
void menu()
{
        int ch ;
        int returnValue = 0;
        do
        {
                printf("\n\n*****************************************\n");
                printf("* Welcome to Data Logger Processing System\n");
                printf("*****************************************\n");
                printf("1. Displays the Dataset.\n");
                printf("2. Displays the monthly average Temperature.\n");
                printf("3. Displays the monthly average Humidity\n");
                printf("4. Displays the monthly average PM 2.5\n");
                printf("5. Displays the monthly average PM 10\n");
                printf("6. Displays the monthly average NO2\n");
                printf("7. Exit the system.\n");
                printf("*****************************************\n");

                printf("Enter a valid choice [1-7]: ");
                returnValue = scanf ("%d", &ch);
                printf("\n");
                if ( returnValue == 0 || returnValue ==  EOF)
                {
                        fflush(stdin);
                        printf("Invalid input. Please enter valid input\n");
                        break;
                }

                iotdata *headPointer = NULL;
                loadFromFile("../data/delhi.csv", &headPointer);

                switch(ch)
                {
                        case 1:
                                printLinkedList(headPointer);
                                break;
                                                                        37,1-8        11%
```



4.2 loadFromFile() :



```
        token = strtok (NULL, seperator);
        if ( token != NULL)
        {
                iter -> code = atoi(token);
        }

        token = strtok (NULL, seperator);
        if ( token != NULL)
        {
                strcpy(iter -> date, token);

        }

        token = strtok (NULL, seperator);
        if ( token != NULL)
        {
                iter -> temperature = atoi(token);
        }


        token = strtok (NULL, seperator);
        if ( token != NULL)
        {
                iter -> humidity = atoi(token);
        }

        token = strtok (NULL, seperator);
        if ( token != NULL)
        {
                iter -> pm25 = atof(token);
        }

        token = strtok (NULL, seperator);
        if ( token != NULL)
        {
                iter -> pm10 = atof(token);
                                                                    100,1-8       85%
```

```
        {
                iter -> temperature = atoi(token);
        }


        token = strtok (NULL, seperator);
        if ( token != NULL)
        {
                iter -> humidity = atoi(token);
        }

        token = strtok (NULL, seperator);
        if ( token != NULL)
        {
                iter -> pm25 = atof(token);
        }

        token = strtok (NULL, seperator);
        if ( token != NULL)
        {
                iter -> pm10 = atof(token);
        }

        token = strtok (NULL, seperator);
        if ( token != NULL)
        {
                iter -> no2 = atof(token);
        }

        iter -> next = NULL;
        appendLinkedList(headPointer, iter);
    }

    fclose(iotdataFile);
    return 0;
}
                                                                    147,0-1       Bot
```

4.3 printLinkedList() :

```
void printLinkedList(iotdata *ct)
{
    iotdata* iter = ct;
    printf("-----------------------------------\n");
    while ( iter != NULL )
    {
        display_iotdata(iter);
        iter = iter -> next ;
    }
    printf("-----------------------------------\n");
}
```

4.4 appendLinkedList() :

```
void appendLinkedList(iotdata **headPointer, iotdata* data)
{
    iotdata* iter = *headPointer ;

    if ( *headPointer == NULL )
    {
        *headPointer = data ;
    }
    else
    {
        while ( iter -> next != NULL)
            iter = iter -> next;
        iter -> next = data ;
    }
    data -> next = NULL ;
}
```

4.5 freeLinkedList() :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <main.h>

void freeLinkedList(iotdata** headPointer)
{
    iotdata* iter = *headPointer;
    //int recordsFreed;
    // Following code block frees the dynamically memory allocated
    while ( iter != NULL )
    {
        iotdata* temp = iter ;
        iter = iter -> next ;
        if ( temp != NULL)
                {
                        free (temp);
                        //recordsFreed++;
                }
    }
    //return recordsFreed;
}
```

4.6 avgTemperature() :

```c
void avgTemperature(iotdata *head)
{
    for(int i=1;i<13;i++)
    {
        int count = 0;
        float temp = 0;
        float avgTemperature;
        char mon[3];
        mon[2] = 0;
        iotdata *iter = head;

        while(iter!= NULL)
        {
                mon[0] = iter->date[3];
                mon[1] = iter->date[4];
            if(atoi(mon) == i)
            {
                temp += iter->temperature;
                count++;
            }
            iter = iter->next;
        }

        if(count!=0)
        {
                avgTemperature = temp/count;
        }

        printf("Month %2d -> %.2f\t", i, avgTemperature);
        for(int i=0; i<avgTemperature/2; i++)
        {
                printf("■");
        }
        printf("\n");
    }
}
```

19,1 3%

4.7 avgHumidity() :

```c
void avgHumidity(iotdata *head)
{
    for(int i=1;i<13;i++)
    {
        int count = 0;
        float temp = 0;
        float avgHumidity;
        char mon[3];
        mon[2] = 0;
        iotdata *iter = head;
        while(iter!= NULL)
        {
                mon[0] = iter->date[3];
                mon[1] = iter->date[4];
            if(atoi(mon) == i)
            {
                temp += iter->humidity;
                count++;
            }
            iter = iter->next;
        }

        if(count!=0)
        {
                avgHumidity = temp/count;
        }
        printf("Month %2d -> %.2f\t", i, avgHumidity);
        for(int i=0; i<avgHumidity/2; i++)
        {
                printf("■");
        }
        printf("\n");
    }
}
```

47,1 28%

avgPM25() :

```
void avgPM25(iotdata *head)
{
    for(int i=1;i<13;i++)
    {
        int count = 0;
        float temp = 0;
        float avgPM25;
        char mon[3];
        mon[2] = 0;
        iotdata *iter = head;
        while(iter!= NULL)
        {
                mon[0] = iter->date[3];
                mon[1] = iter->date[4];
            if(atoi(mon) == i)
            {
                temp += iter->pm25;
                count++;
            }
            iter = iter->next;
        }

        if(count!=0)
        {
                avgPM25 = temp/count;
        }

        printf("Month %2d -> %.2f\t", i, avgPM25);
        for(int i=0; i<avgPM25/4; i++)
        {
                printf("■");
        }
        printf("\n");
    }
}
                                                                    108,1         52%
```

4.9 avgPM10():

```
void avgPM10(iotdata *head)
{
    for(int i=1;i<13;i++)
    {
        int count = 0;
        float temp = 0;
        float avgPM10;
        char mon[3];
        mon[2] = 0;
        iotdata *iter = head;
        while(iter!= NULL)
        {
                mon[0] = iter->date[3];
                mon[1] = iter->date[4];
            if(atoi(mon) == i)
            {
                temp += iter->pm10;
                count++;
            }
            iter = iter->next;
        }

        if(count!=0)
        {
                avgPM10 = temp/count;
        }
        printf("Month %2d -> %.2f\t", i, avgPM10);
        for(int i=0; i<avgPM10/4; i++)
        {
                printf("■");
        }
        printf("\n");
    }
}
                                                                    118,1         76%
```

4.10 avgNO2():

```
void avgNO2(iotdata *head)
{
    for(int i=1;i<13;i++)
    {
        int count = 0;
        float temp = 0;
        float avgNO2;
        char mon[3];
        mon[2] = 0;
        iotdata *iter = head;
        while(iter!= NULL)
        {
                mon[0] = iter->date[3];
                mon[1] = iter->date[4];
            if(atoi(mon) == i)
            {
                temp += iter->no2;
                count++;
            }
            iter = iter->next;
        }

        if(count!=0)
        {
                avgNO2 = temp/count;
        }
        printf("Month %2d -> %.2f\t", i, avgNO2);
        for(int i=0; i<avgNO2/2; i++)
        {
                printf("");
        }
        printf("\n");
    }
}
                                                                    179,1         Bot
```

5. OUTPUTS :

5.1 Output of Menu Function:-

```
*********************************************
* Welcome to Data Logger Processing System
*********************************************
1. Displays the Dataset.
2. Displays the monthly average Temperature.
3. Displays the monthly average Humidity
4. Displays the monthly average PM 2.5
5. Displays the monthly average PM 10
6. Displays the monthly average NO2
7. Exit the system.
*********************************************
Enter a valid choice [1-7]:
```

5.2 Output of Display Function:-

```
**********************************************
* Welcome to Data Logger Processing System
**********************************************
1. Displays the Dataset.
2. Displays the monthly average Temperature.
3. Displays the monthly average Humidity
4. Displays the monthly average PM 2.5
5. Displays the monthly average PM 10
6. Displays the monthly average NO2
7. Exit the system.
**********************************************
Enter a valid choice [1-7]: 1

-----------------------------------
| Delhi | 11 | 01-01-2021 | 19 | 43 | 184.76 | 337.93 | 53.64 |
| Delhi | 11 | 02-01-2021 | 21 | 47 | 177.86 | 334.63 | 53.12 |
| Delhi | 11 | 03-01-2021 | 20 | 48 | 176.12 | 345.99 | 54.92 |
| Delhi | 11 | 04-01-2021 | 15 | 42 | 182.10 | 335.43 | 53.24 |
| Delhi | 11 | 05-01-2021 | 24 | 46 | 176.54 | 351.02 | 55.72 |
| Delhi | 11 | 06-01-2021 | 21 | 46 | 184.75 | 351.39 | 55.78 |
| Delhi | 11 | 07-01-2021 | 16 | 43 | 184.94 | 335.62 | 53.27 |
| Delhi | 11 | 08-01-2021 | 25 | 50 | 176.64 | 348.06 | 55.25 |
| Delhi | 11 | 09-01-2021 | 16 | 46 | 183.19 | 351.44 | 55.78 |
| Delhi | 11 | 10-01-2021 | 23 | 44 | 184.97 | 346.64 | 55.02 |
| Delhi | 11 | 11-01-2021 | 18 | 41 | 182.44 | 346.20 | 54.95 |
| Delhi | 11 | 12-01-2021 | 23 | 42 | 182.21 | 336.76 | 53.45 |
| Delhi | 11 | 13-01-2021 | 19 | 46 | 177.24 | 336.22 | 53.37 |
| Delhi | 11 | 14-01-2021 | 24 | 43 | 176.96 | 345.82 | 54.89 |
| Delhi | 11 | 15-01-2021 | 15 | 49 | 182.01 | 342.02 | 54.29 |
| Delhi | 11 | 16-01-2021 | 21 | 49 | 180.01 | 344.81 | 54.73 |
| Delhi | 11 | 17-01-2021 | 22 | 49 | 181.48 | 344.47 | 54.68 |
| Delhi | 11 | 18-01-2021 | 15 | 42 | 181.30 | 342.68 | 54.39 |
| Delhi | 11 | 19-01-2021 | 18 | 42 | 180.36 | 338.68 | 53.76 |
| Delhi | 11 | 20-01-2021 | 21 | 42 | 178.25 | 335.71 | 53.29 |
| Delhi | 11 | 21-01-2021 | 23 | 48 | 176.69 | 338.45 | 53.72 |
| Delhi | 11 | 22-01-2021 | 18 | 42 | 178.13 | 344.17 | 54.63 |
```

5.3 Output of avgtemperature:-

```
Month-wise comparsion of Temperature
Month  1 -> 20.06
Month  2 -> 25.71
Month  3 -> 28.55
Month  4 -> 33.73
Month  5 -> 37.16
Month  6 -> 37.87
Month  7 -> 32.39
Month  8 -> 28.10
Month  9 -> 27.07
Month 10 -> 21.65
Month 11 -> 18.17
Month 12 -> 8.74
```

5.4 Output of avghumidity:-

5.5 Output of avgPM25:-

5.6 Output of avgPM10:-

5.7 Output of avgNO2:-

5.8 Output of Exit :-


****
* Welcome to Data Logger Processing System
****
1. Displays the Dataset.
2. Displays the monthly average Temperature.
3. Displays the monthly average Humidity
4. Displays the monthly average PM 2.5
5. Displays the monthly average PM 10
6. Displays the monthly average NO2
7. Exit the system.
****
Enter a valid choice [1-7]: 7
Exiting