Design and Implementation of an IoT-Based Air Pollution Monitoring System using Environmental Sensors

[1]Rahul Raj Singh, [1]Aaditya Amitabh, [1]Anushka Singh, [1]Mukta Tehri, [2] Dr.Gurwinder Singh

[1,2] Department of AIT-CSE, Chandigarh University, Mohali, Punjab, India

Abstract- The design and execution of an Internet of Things-based air pollution monitoring system using environmental sensors. The system incorporates the MQ135 air quality sensor, MQ6 LPG gas sensor LM35 temperature sensor, and, along with a Node MCU and ESP8266 Wi-Fi module. The objective is to enable real-time monitoring and analysis of air quality parameters. The sensors detect gases like CO2, ammonia, CO, and LPG, while the temperature sensor measures temperature variations. The IoT infrastructure facilitates data collection and transmission to a cloud-based platform for processing and visualization. The system provides valuable insights into air quality, enablesearly pollution detection, and supports data-driven decision- making. The modular design and use of widely available sensors make it adaptable for different environments. This research contributes to addressing the global challenge of air pollution and its impact on human health and the environment.

Words: Air Pollution, MQ135 Sensor, IOT, Arduino Uno.

I. INTRODUCTION

Around the globe, the health and overall well-being of humans are negatively impacted by air pollution. It is possible for dangerous airpollutants to have an adverse effect on the environment and public health as a result of traffic, industrialization, and other human activities. Systems for measuring air quality and locating pollution sources have been developed to solve this problem. Adulterants including carbon monoxide, nitrogen dioxide, and particulate matter can be detected and measured by these systems using cutting-edge sensors. A color-coded map or graph is one example of an accessible presentation of the data analysis and collection.

The goal of air pollution monitoring systems is to give the general public and decision-makers access to real-time information on the state of the air so they can make informed choices that will safeguard public health and the environment. These devices can assist in lowering exposure to pollutants and improving air quality for everyone by giving accurate and current information on air quality.

In order to review the most recent advancements in air pollution monitoring systems and their applications, this research paper will do so. The several kinds of monitoring systems are covered, including passive and active monitoring systems, remote sensing, and mobile monitoring devices. It

also examines the difficulties and restrictions that air pollution monitoring systems now encounter and suggests possible solutions. We can better inform the public and policymakers to enhance air quality and safeguard public health by knowing the capabilities and limitations of these systems.

II. LITERATURE SURVEY

Air pollution is an important environmental issue that influences both human and environmental health. Systems for monitoring air pollution have grown to be crucial tools for assessing air quality and minimizing the harmful effects of air pollution. Several research projects on the creation and application of air pollution monitoring systems have been carried out recently.

A study on the creation and application of an IoT-based air pollution monitoring system was carried out in 2020 by Kumar et al. The project's goal was to create a dependable, affordable system for real-time air quality monitoring. The sensors in the system were built to measure many aspects of air quality, including particulate matter, carbon monoxide, and nitrogen dioxide.

Another study (Snyder et al., 2013) concentrated on the use of inexpensive sensors to measure air pollution. To determine the amount of particulate matter and other pollutants in the air, the researchers employed a variety of sensors. The findings demonstrated that low-cost sensors might be used to detect changes in air quality and track air pollution in real-time.

In a study done in China, researchers created a monitoring system for air pollution utilizing ground-level observations and satellite data (Tian et al., 2017). In order to build a complete picture of air quality, the system merged ground-level observations with satellite data to determine the concentration of contaminants in the air. The outcomes demonstrated the system's efficiency in air quality monitoring. The findings indicated that the technology might be used to lessen exposure to dangerous chemicals and was efficient at tracking air pollution levels.

Other studies (Zheng et al., 2017) have concentrated on the creation of mobile air pollution monitoring devices. To assess air quality in real-time, these systems combine sensors and mobile technology. The findings demonstrated the potential of mobile air pollution monitoring devices to deliver precise and timely information about air quality, assisting people in making decisions that will affect their health and well-being.

Overall, these studies show how crucial air pollution monitoring systems are to safeguarding both the

environment and public health. The creation of new technology and techniques for air quality monitoring can aid in lowering exposure to dangerous pollutants and enhancing the general standard of living for people who live in polluted areas.

III. PROPOSED SYSTEM

The use of a MQ135 Air Quality Sensor, LM35 Temperature Sensor, and MQ6 LPG Gas Sensor coupled to a node MCU in an Internet of Things (IOT)- based air pollution monitoring system is presented. The system is connected to an ESP8266 Wi-Fi module for the purpose of analyzing sensor data.

IV. SYSTEM DESIGN

The MQ135 air quality sensor and Node MCU are utilized in the hardware portion of the system. This sensor can detect smoke, CO2, CO, ammonia, and other gases. Because Arduino lacks Wi-Fi functionality, we must utilize the Node MCU, also known as the ESP8266 Wi-Fi chip, and connect it to our mobile hotspot in order to communicate this data to the cloud. One of the most often used power sources in use today is the 5V power source. An LCD display that is frequently used for interacting with embedded systems is the H44780 Character LCD. In this project, we're using the 16X2 Configuration's 4-bit write mode.

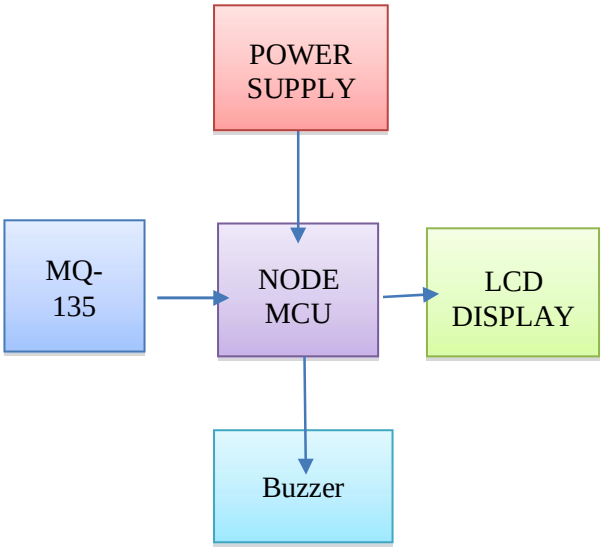

Figure: Block Diagram Air Quality Monitoring System

Fig. 1

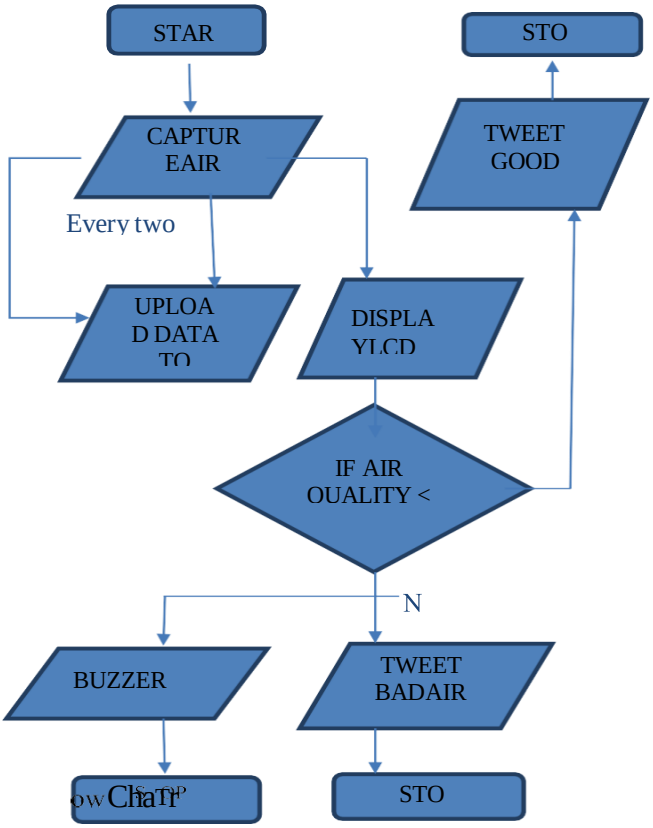

Figure: Flow Chart of Air Quality Monitoring System

Fig. 2

V. RESULT AND DISCUSSION

Access to Wi-Fi or the internet is made possible by the ESP8266 Wi-Fi module for the project. It is a cost-effective solution that gives your projects a lot of power. It is the most advanced gadget on the Internet of Things platform and can interface with any microcontroller. More information on this. Then, an Arduino and MQ135 sensor will be coupled. The VCC and ground connections of the sensor should be connected to the Arduino's 5V and ground, and the analogue pin should be connected to the Arduino's A0. To make a buzzer start beeping when the condition is satisfied, connect it to Arduino pin 8. The MQ135 sensor is the ideal choice because it can identify NH3, NOx, alcohol, benzene, smoke, CO2, and a few more compounds.

350 PPM is the highest permissible level of air quality, and it shouldn't go over 1000 PPM. Headaches, weariness, and stuffy, stagnant air start to occur when it exceeds the limit of 1000 PPM. If it rises above 2000 PPM, it may potentially worsen your health and increase heart rate. The LCD and website will display "Good Quality of Air" when the reading is less than 1000 PPM. A buzzer will start to sound whenever the figure goes over 1000 PPM, and the LCD and website will display "Bad quality of air."

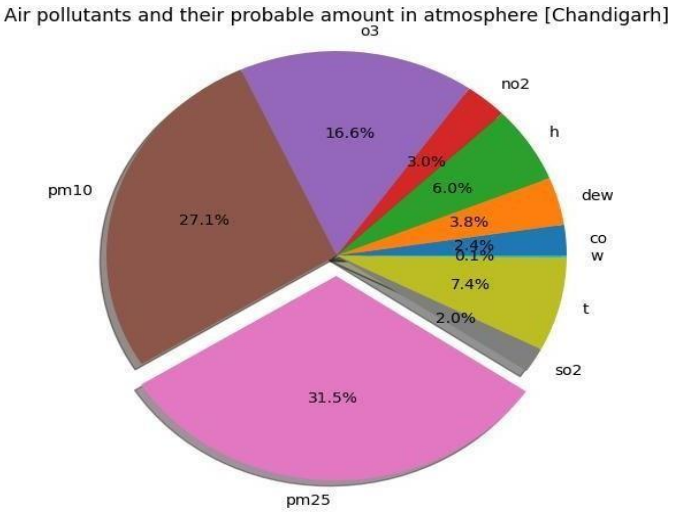

Fig. 3: Pie Chart Showing Pollutants Percentage in Air

```python
import matplotlib.pyplot as plt

pollutants = [i for i in iaqi]
values = [i['v'] for i in iaqi.values()]


# Exploding the first slice
explode = [0 for i in pollutants]
mx = values.index(max(values))  # explode 1st slice
explode[mx] = 0.1

# Plot a pie chart
plt.figure(figsize=(8,6))
plt.pie(values, labels=pollutants,explode=explode,autopct='%1.1f%%', shadow=True)

plt.title('Air pollutants and their probable amount in atmosphere [Chandigarh]')

plt.axis('equal')
plt.show()
```

Fig.4)Python Code to plot Pie Chart

Using the Python Matplotlib package, the provided code generates a pie chart showing the most likely concentration of air contaminants in Chandigarh's atmosphere. Let us step-by-step dissect the code:

1. Using in the required libraries:

```python
import matplotlib.pyplot as plt
```

Using the Python Matplotlib package, the provided code generates a pie chart showing the most likely concentration of air contaminants in Chandigarh's atmosphere. Let's step-by-step dissect the code

2. Using in the required libraries:

```python
pollutants = [i for i in iaqi]
values = [i['v'] for i in iaqi.values()]
```

In this case, the iaqi variable—which is assumed to be a dictionary—is transformed into a list comprehension named pollutants, and those names are extracted from it. With the use of the iaqi dictionary, values is a distinct list comprehension that extracts the values associated with each pollutant.

3. Exploding the first slice:

```python
explode = [0 for i in pollutants]
mx = values.index(max(values))
explode[mx] = 0.1
```

This step generates an explode list that is the same length as the pollutants list. The initial value of each element in the explode list is 0. The element in the explode list that corresponds to that position is then set to 0.1 using the index() method to locate the greatest value in the values list. This is done to visually distinguish the slice of the pie chart that represents the pollutant with the greatest value.

4. Plotting the pie chart:

```python
plt.figure(figsize=(8, 6))
plt.pie(values, labels=pollutants, explode=explode, autopct='%1.1f%%', shadow=True)
plt.title('Air pollutants and their probable amount in atmosphere [Chandigarh]')
plt.axis('equal')
plt.show()
```

The pie chart's actual creation and presentation fall under the purview of this section.

Using the command plt.figure(figsize=(8, 6)), a new figure with the specified 8 by 6 dimensions is created.

The function plt.pie(values, labels=pollutants, explode=explode) creates the pie chart.

autopct='%1.1f%%', shadow=True). Each pollutant's name is listed in the labels list, which also contains the values for each slice. The explode list decides whether anyslice should be visually separated, and autopct='%1.1f%%' formats the percentage labels to have one decimal place. Finally, shadow=True adds a shadow effect to the chart.

The title of the chart is specified using plt.title("Air pollutants and their probable amount in atmosphere [Chandigarh]").
The pie chart is displayed as a circle rather than an ellipse thanks to plt.axis('equal').
The chart is shown on the screen by plt.show().

Overall, the code creates a pie chart showing slices with labels that correspond to the likely concentrations of air contaminants in Chandigarh's environment. For better visualisation, the slice of the pie chart that represents the pollutant with the greatest value is slightly offset from the other slices.

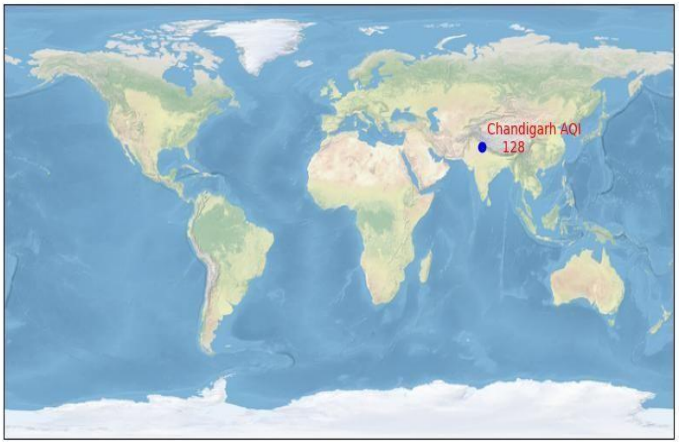

Fig. 4)Showing Map of a city with it's AQI Level

```
1   import cartopy.crs as ccrs
2
3   geo = data['city']['geo']
4
5   fig = plt.figure(figsize=(10,8))
6   ax = plt.axes(projection=ccrs.PlateCarree())
7   ax.stock_img()
8
9   plt.scatter(geo[1],geo[0],color='blue')
10  plt.text(geo[1] + 3,geo[0]-2,f'{city} AQI \n    {aqi}',color='red')
11
12  plt.show()
13
```
Fig.5) Python Code to show map of the city with AQI Level.

The provided code plots a scatter point reflecting a city's geographic position on a map and displays the AQI (Air Quality Index) value related to that city using the Cartopy library in Python. Let's go over the code line by line:

1. The required libraries are imported:

```
import cartopy.crs as ccrs
```

This line imports the cartopy.crs module from the Cartopy library, which supports a number of different coordinate reference systems and map projections.

2. Getting the geographic coordinates:

```
geo = data['city']['geo']
```

This phrase assumes that there is a dictionary called data that has information about the city in it. The city's geographic coordinates, latitude and longitude, are extracted and kept in the geo variable.

3. The figure's axes are made:

```
fig = plt.figure(figsize=(10, 8))
ax = plt.axes(projection=ccrs.PlateCarree())
```

Using plt.figure(figsize=(10, 8)), a figure with the desired dimensions of 10 by 8 inches is created in this section. The map projection is then changed to Plate Carrée (equidistant cylindrical projection) using plt.axes(projection=ccrs.PlateCarree()), which also produces an axes object called axe.

4. Including a stock photo to the story:

```
ax.stock_img()
```

With the addition of this line, the plot gains a stock image that gives a background map with coastlines, land, and ocean regions.

5. The scatter point and text label are plotted:

```
plt.scatter(geo[1], geo[0], color='blue')
plt.text(geo[1] + 3, geo[0] - 2, f'{city} AQI \n    {aqi}', color='red')
```

Using the longitude (geo[1]) and latitude (geo[0]) values, the function plt.scatter(geo[1], geo[0],color='blue') plots a scatter point on the map. The point is coloured blue.

With the help of the function plt.text(geo[1] + 3, geo[0] - 2, f'city AQI n aqi', color='red'), the text "city AQI" is placed adjacent to the scatter point. The text uses the words "city" and "aqi" interchangeably. The parameters geo[1] + 3 and geo[0] - 2 specify where the text label willbe in reference to the scatter point. The hue of the text label is crimson.

6. Displaying the plot:

```
plt.show()
```

On the screen, this line presents the plot.

In general, the code produces a map plot using Cartopy, where a scatter point denotes a city's precise location. Near the scatter point, a text label with the AQI value is visible. The background of the map is a stock photo.

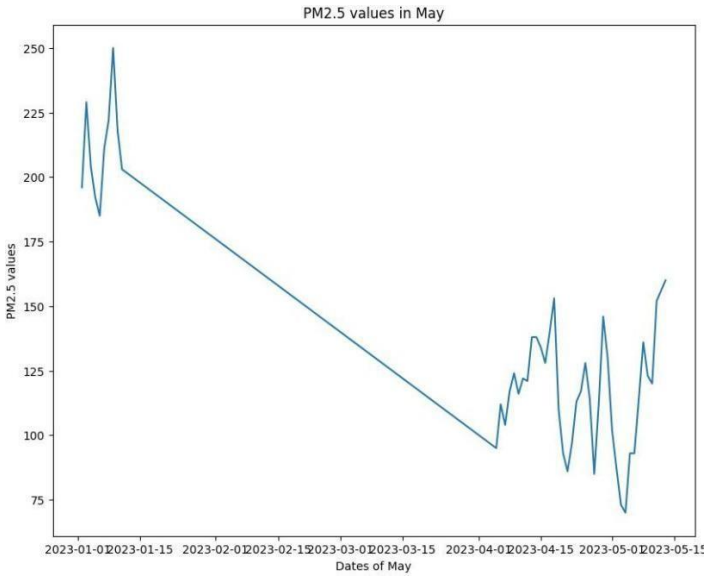

Fig. 5) Graph Showing PM 2.5 values

```
1   import matplotlib.pyplot as plt
2
3   dates = df21['date']
4   pm25 = df21['pm25']
5   pm25 = [int(i) for i in pm25]
6
7   plt.figure(figsize=(10,8))
8
9   length = len(dates)
10
11  plt.plot(dates,pm25)
12  plt.title('PM2.5 values in May')
13  plt.xlabel('Dates of May')
14  plt.ylabel('PM2.5 values')
15  plt.show()
```

Python Code to show graph

The provided code generates a line plot of the PM2.5 values over time, specifically for the month of May, using the Python Matplotlib package. Let's step-by-step dissect the code:

1. The required libraries are imported:

```
import matplotlib.pyplot as plt
```

This line imports the pyplot module from the matplotlib library, which gives users a straightforward interface for making different kinds of plots and charts.

2. Setting the parameters:

```
dates = df21['date']
pm25 = df21['pm25']
pm25 = [int(i) for i in pm25]
```

Dates and PM25 are given values from the "date" and "pm25" columns of a DataFrame in this example. The 'pm25' values are also transformed into integers using a list comprehension.

3. Creating the figure and setting its size:

```
plt.figure(figsize=(10, 8))
```

With the supplied dimensions of 10 inches by 8 inches, this line generates a new figure.

4. Laying down the line graph:

```
length = len(dates)
plt.plot(dates, pm25)
```

Creating a line plot requires the usage of the plt.plot() function. The 'dates' and 'pm25' values are plotted on the x- and y-axes, respectively. To demonstrate a trend or pattern across time, a line is drawn connecting the data points.

5. Incorporating a title, x-axis label, and y-axis label

```
plt.title('PM2.5 values in May')
plt.xlabel('Dates of May')
plt.ylabel('PM2.5 values')
```

These lines provide the y-axis label as "PM2.5 values,"the x-axis label as "Dates of May," and the plot title as "PM2.5 values."

6. Presenting the story:

```
plt.show()
```

On the screen, this line presents the plot.

Overall, the code creates a line plot that shows the PM2.5 values over time in the month of May. The y- axis displays the relevant PM2.5 readings, and the x- axis displays the dates. There is a screen showing the plot.

CONCLUSION

The MQ135 Gas Sensor is used by this system to send gas, such as benzene, alcohol, smoke, etc. By using an Arduino microcontroller to monitor the air quality of the area, IOT technology is said to improve air quality. The use of Io technology enhances the process of monitoring various environmental parameters, such as the air quality monitoring issue raised in this study. This board's Wi-Fi module functions as both an internet connection and an informational portal for the air quality. To measure the air quality in real-time, this uses a MQ135 Gas Sensor and Node MCU. Node MCU will send information to the Things Peak platform, which is linked to Twitter, when the air quality drops below a set threshold so that it may send the Twitter notification and warn the neighborhood. Here, the MQ135 gas sensor is used to detect a range of hazardous substances, while Arduino, the project's brain, regulates every step of the procedure. The visual output is provided by an LCD, and the system as a whole is linked to the internet by a Wi-Fi module. It successfully supports modern technologies while still advocating for a healthy lifestyle. The capabilities of this system and an app for smartphones can be used by people to monitor the pollution level.

VI. REFERENCES

[1] S. Kumar and A. Jasuja. Air quality monitoring system based on internet of things using raspberry pi. Pages 1341-1346, May 2017.

[2] S.R. Enigella and H. Shahnasser. Real-time air quality monitoring. Pages 182-185, Jan 2018.

[3] D. Wang, C. Jiang, and Y. Dan. Design of air quality monitoring system based on the internet of things. Pages 418-423, Dec 2016.

[4] L. Peng, F. Danni, J. Shengqian, and W. Mingjie. A movableindoor air quality monitoring system. Pages 126-129, July 2017.

[5] Sumanth Reddy Enigella, Hamid Shahnasser. "Real-Time Air Quality Monitoring", 2018 10th International Conference on Knowledge and Smart Technology (KST), 2018.

[6] www.irjet.net.

[7] Poonam Paul, Ritik Gupta, Sanjana Tiwari, Ashutosh Sharma, “IoT based Air Pollution Monitoring System with Arduino”, UART, May 2005.

[8] Zishan Khan, Abbas Ali, Moin Moghal,” IoT based Air Pollution using NodeMCU and Thingspeak”, IRANS, pp. 11-16, March 2014.

[9] Mohan Joshi, “Research Paper on IoT based Air and Sound Pollution monitoring system”, IETS Journal, pp. 11-17, September 2015.

[10] "Malaya Ranjan, Raj Kumar, "Understanding Partsper million in real-time air quality index", Journal of Mathematics and advanced sciences, pp. 23-29, September 2009

[11] H. Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications. Boston, MA: Springer US, 2011, ch. Internet of Things, pp. 307-323.

[12] Fouzi Harrow, Mohamed Nounou, Hazem Nounou “Detecting Abnormal Ozone Levels Using Pea Based Gir Hypothesis Testing”2013 Ieee Symposium On Computational Intelligence AndData Mining.

[13] Srinivas Devarakonda, Parveen Sevusu, Hong Hang Liu, Ruilin Liu, Liviu Iftode, Badri Nath Urbcomp “Real-Time Air Quality Monitoring Through Mobile Sensing In Metropolitan Areas”13, August 2013 Acm.