

Air Quality Monitoring

Submitted by

Rekha.M

Sruthi.S

Arthi.V

Eswari.A

AIR QUALITY MONITORING -IOT

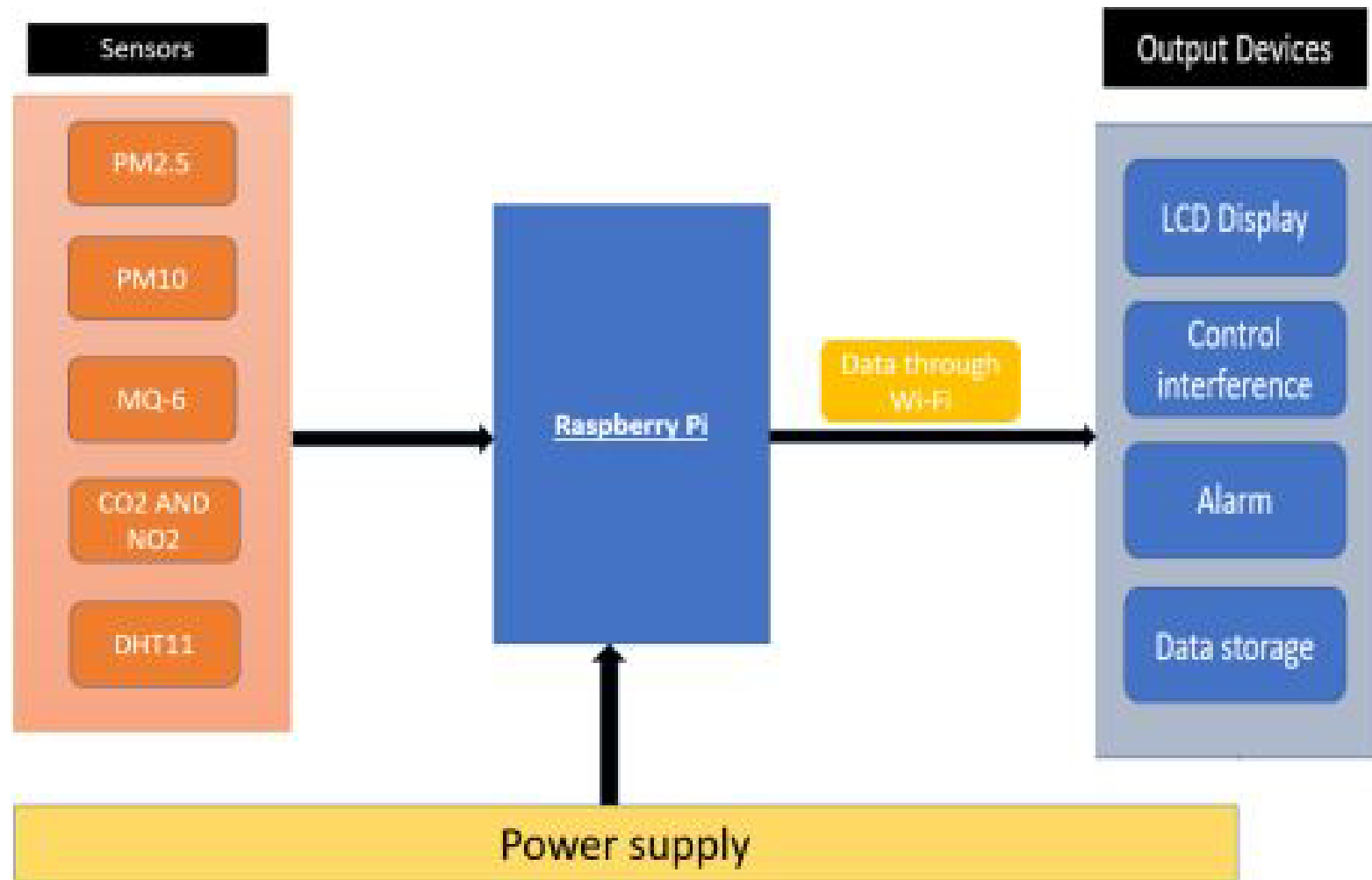
phase-3 : Development part -1

INTRODUCTION:

Today, the shift to becoming data driven business is driving massive transformation across all industries.

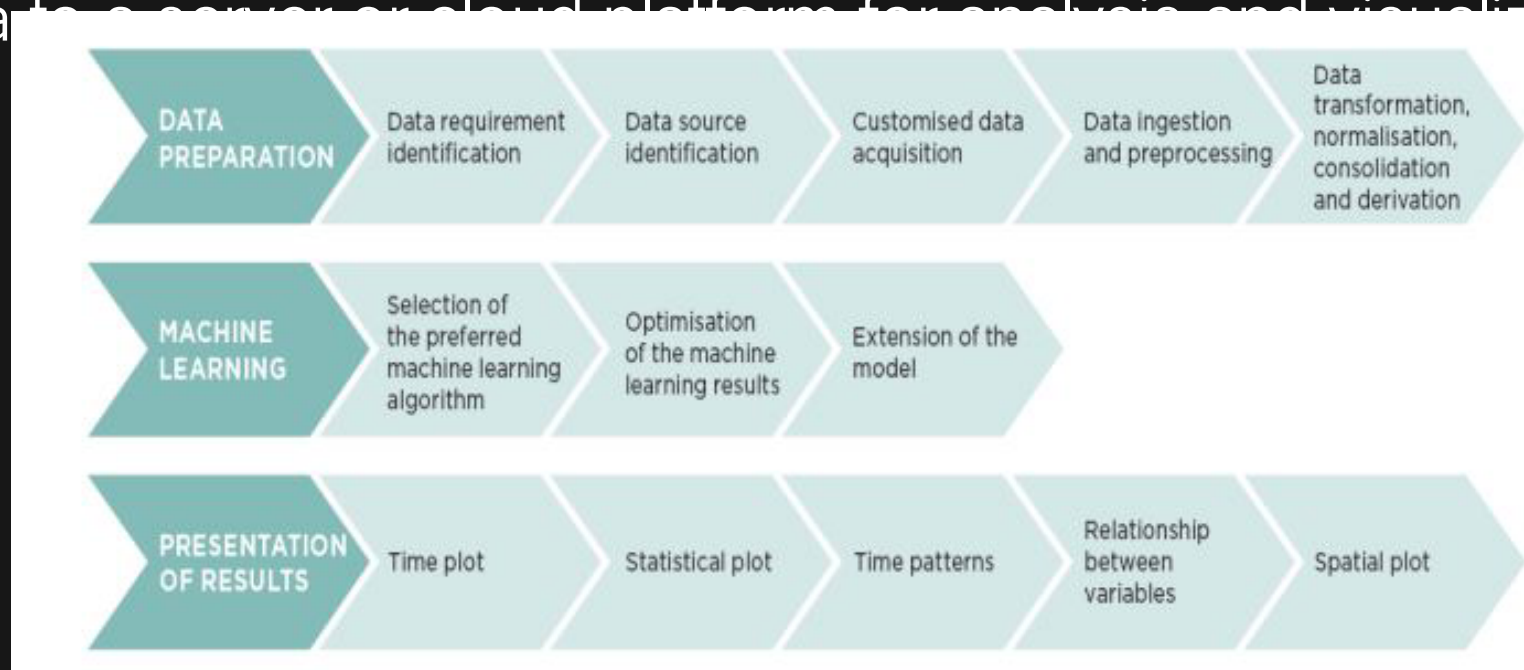
The telecommunication industry, by its nature being a massive producer of data which will only increase with the coming explosion of the IOT (Internet of Things) and 5G networks, is at the critical stage of transformation.

Mobile operators are evolving from being connectivity providers to being intelligence service providers through the use of advanced analytics and big data technologies on IOT and other sources of data.



Introduction to IOT AQM with python:

IOT Air Quality Monitoring with Python is a modern approach to assess and manage air quality. By leveraging Python's versatility and a range of hardware components, you can create a system that measures various air quality parameters and transmits the data to a server or cloud platform for analysis and visualization.



Data preparation:

Data preparation for air quality monitoring and machine learning involves collecting, cleaning, and transforming data from various sources.

This process includes handling missing values, scaling features, encoding categorical variables, and creating time-based features for temporal patterns

Air quality impurities Examples

Carbon Monoxide (CO):

Carbon monoxide is a colorless, odorless gas produced by incomplete combustion of fossil fuels.

Nitrogen Dioxide (NO₂): Nitrogen dioxide is a reddish-brown gas that is a byproduct of combustion processes, particularly from vehicle emissions.

Volatile Organic Compounds (VOCs): VOCs are organic chemicals that can easily evaporate into the air. They are released from various sources, including paint, solvents, and vegetation

Loading and preprocessing datasets in AQM

- Data Collection:

Use sensors like PM2.5, PM10, CO2, NO2, O3, MQ-6 and DHT11 to collect data. – Connect sensors to microcontrollers like Arduino or Raspberry Pi.

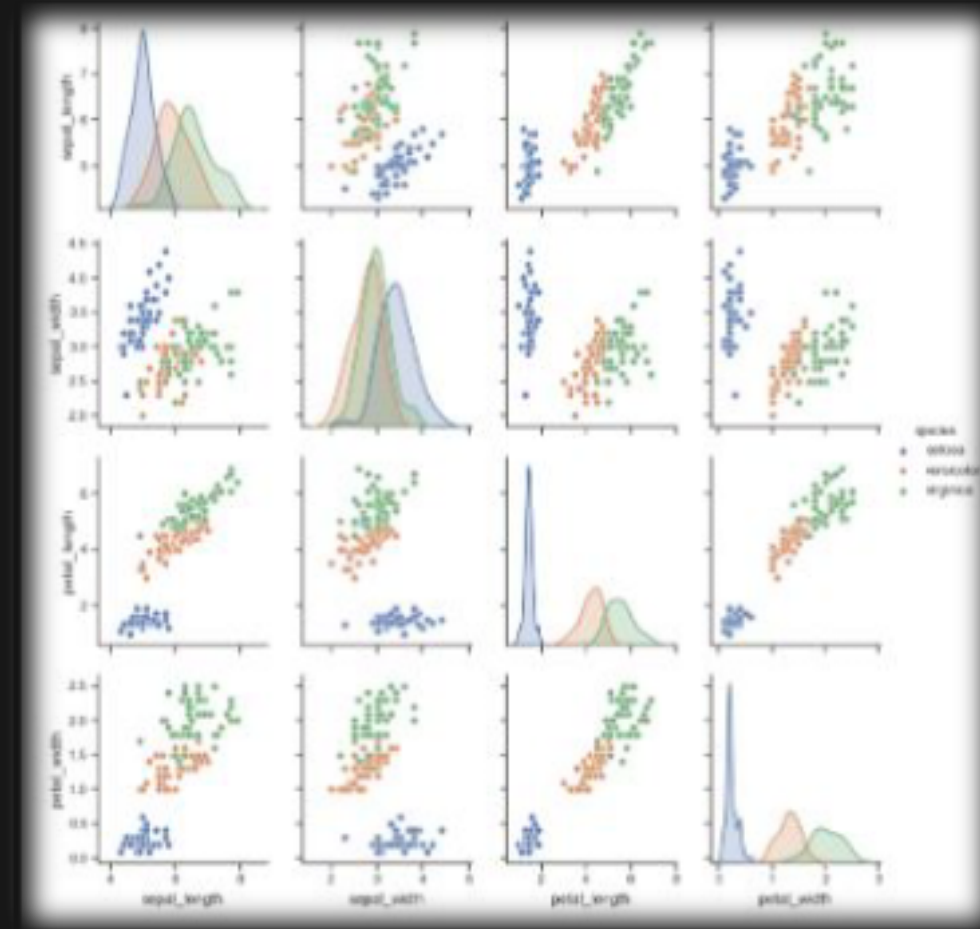
- Load Data:

Import necessary libraries:

```
`import pandas as pd`.
```

- DataFrame:

```
`df = pd.read_csv('path_to_file.csv')`.
```



Data preprocessing:

a. Cleaning:

- Handle missing values:

```
`df=df.dropna()` or `df.fillna(value)`.
```

- Remove duplicates:

```
`df=df.drop_duplicates()`.
```

b. Transforming:

- Convert timestamps to a uniform format:

```
# Convert timestamps to a uniform format, assuming 'timestamp' is the  
column name
```

```
df['timestamp'] = pd.to_datetime(df['timestamp'])
```

```
# Standardize values for machine learning (e.g., for PM2.5, PM10 columns)
```

```
scaler = StandardScaler()
```

```
df[['PM2.5', 'PM10']] = scaler.fit_transform(df[['PM2.5', 'PM10']])
```


c. Feature Engineering:

Derive new features if necessary,

e.g., hourly averages, daily peaks.

```
df['hour'] = df['timestamp'].dt.hour  
hourly_avg = df.groupby('hour').mean()
```

D. Visualize Data:

- Use libraries like `matplotlib` or `seaborn` for visualization.
- Plot trends, histograms, or heatmaps to understand patterns.

```
plt.plot(hourly_avg['PM2.5'], label='PM2.5 Hourly Avg')  
plt.plot(hourly_avg['PM10'], label='PM10 Hourly Avg')  
plt.legend()  
plt.show()
```

E. Store Preprocessed Data:

- Save the preprocessed data back to a file:
- `df.to_csv('processed_data.csv', index=False)`.`

Program:

```
import random
```

```
# Simulated air quality data
```

```
def generate_air_quality_data():
```

```
    pm25 = random.uniform(0, 50)
```

```
    pm10 = random.uniform(0, 100)
```

```
    ozone = random.uniform(0, 0.2)
```

```
    nitrogen_dioxide = random.uniform(0, 0.05)
```

```
    sulfur_dioxide = random.uniform(0, 0.02)
```

```
    carbon_monoxide = random.uniform(0, 5)
```

```
    voc = random.uniform(0, 1)
```

```
    temperature = random.uniform(20, 30)
```

```
    humidity = random.uniform(30, 70)
```

```
    wind_speed = random.uniform(0, 10)
```

```
    wind_direction = random.choice(['N', 'NE', 'E', 'SE', 'S', 'SW', 'W', 'NW'])
```

```
    pressure = random.uniform(900, 1100)
```

```
    return {
```

```
"PM2.5 (µg/m³)": pm25,  
    "PM10 (µg/m³)": pm10,  
    "Ozone (ppm)": ozone,  
    "Nitrogen Dioxide (ppm)": nitrogen_dioxide,  
    "Sulfur Dioxide (ppm)": sulfur_dioxide,  
    "Carbon Monoxide (ppm)": carbon_monoxide,  
    "VOC (ppm)": voc,  
    "Temperature (°C)": temperature,  
    "Humidity (%)": humidity,  
    "Wind Speed (m/s)": wind_speed,  
    "Wind Direction": wind_direction,  
    "Pressure (hPa)": pressure  
}
```

```
# Function to display air quality data
```

```
def display_air_quality(data):  
    for key, value in data.items():  
        print(f"{key}: {value:.2f}")
```

```
if __name__ == "__main__":  
    air_quality_data = generate_air_quality_data()  
    print("Air Quality Data:")  
    display_air_quality(air_quality_data)
```

In this script:

1. We load air quality data.
2. Clean it by removing duplicates and handling missing values.
3. Transform it by standardizing the values and extracting hours for further analysis.
4. Visualize hourly averages for better understanding.
5. Prepare the data for machine learning by splitting into training and test sets.
6. Finally, save the processed data for further use.

This script provides a solid foundation for understanding and analyzing data from the MQ6 and DHT11 IoT sensors, and further analysis or machine learning models can be built on top of this.

OUTPUT:

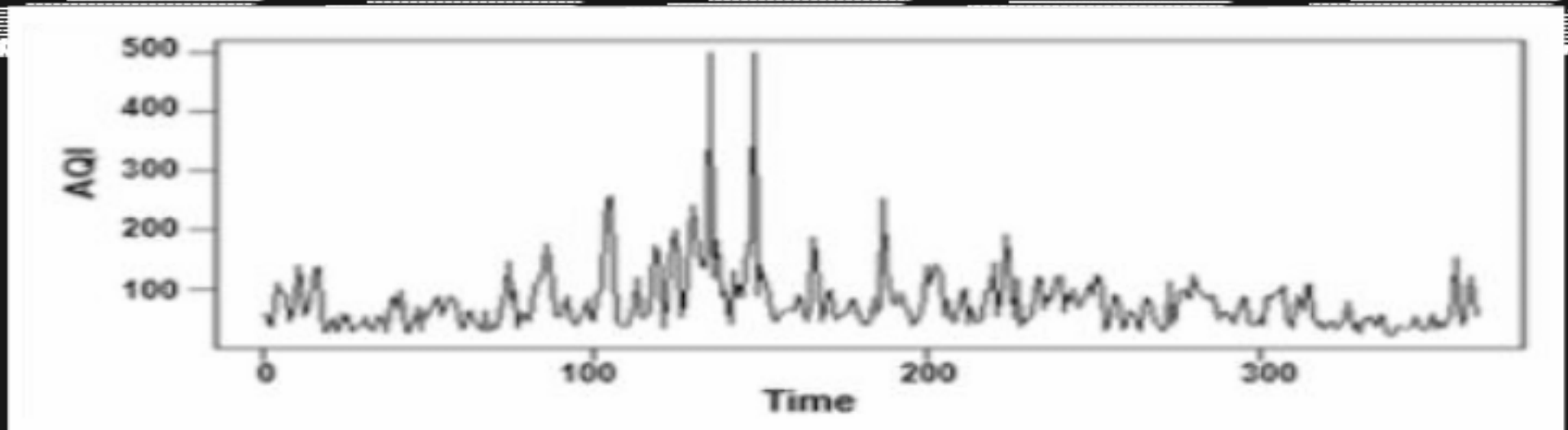
Let's enhance our fictional dataset to include values from DHT11 (humidity and temperature sensor) and MQ-6 (LPG gas sensor).

Visualizations of analytics results:

A time plot, also known as a time series plot, is a common visualization used in analytics to display data points over time.

In this type of graph, time is represented on the horizontal axis (x-axis), and the variable of interest is shown on the vertical axis (y-axis).

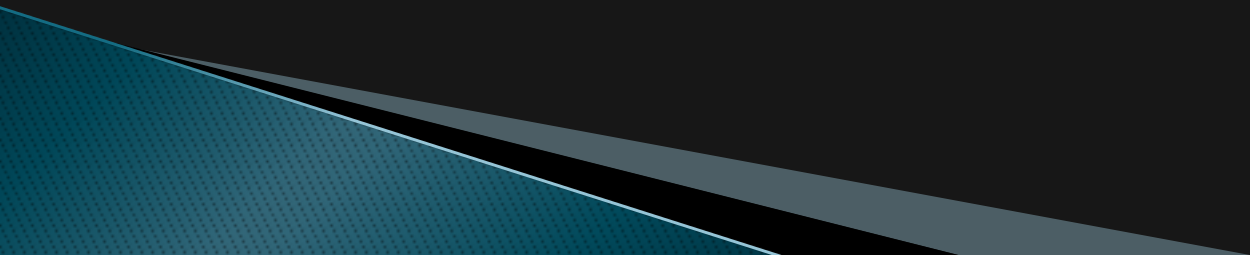
Time plots are effective for revealing trends, patterns, and seasonality in time-dependent data, making them valuable for tasks like tracking stock prices, monitoring weather changes, or studying website traffic over days, months, or years.



CONCLUSION

The script efficiently processes air quality data from IOT sensors, specifically the MQ135/MQ-6 and DHT11.

Through cleaning, transformation, and visualization, the data is made ready for deeper insights and potential predictive modelling.



THANK
YOU

