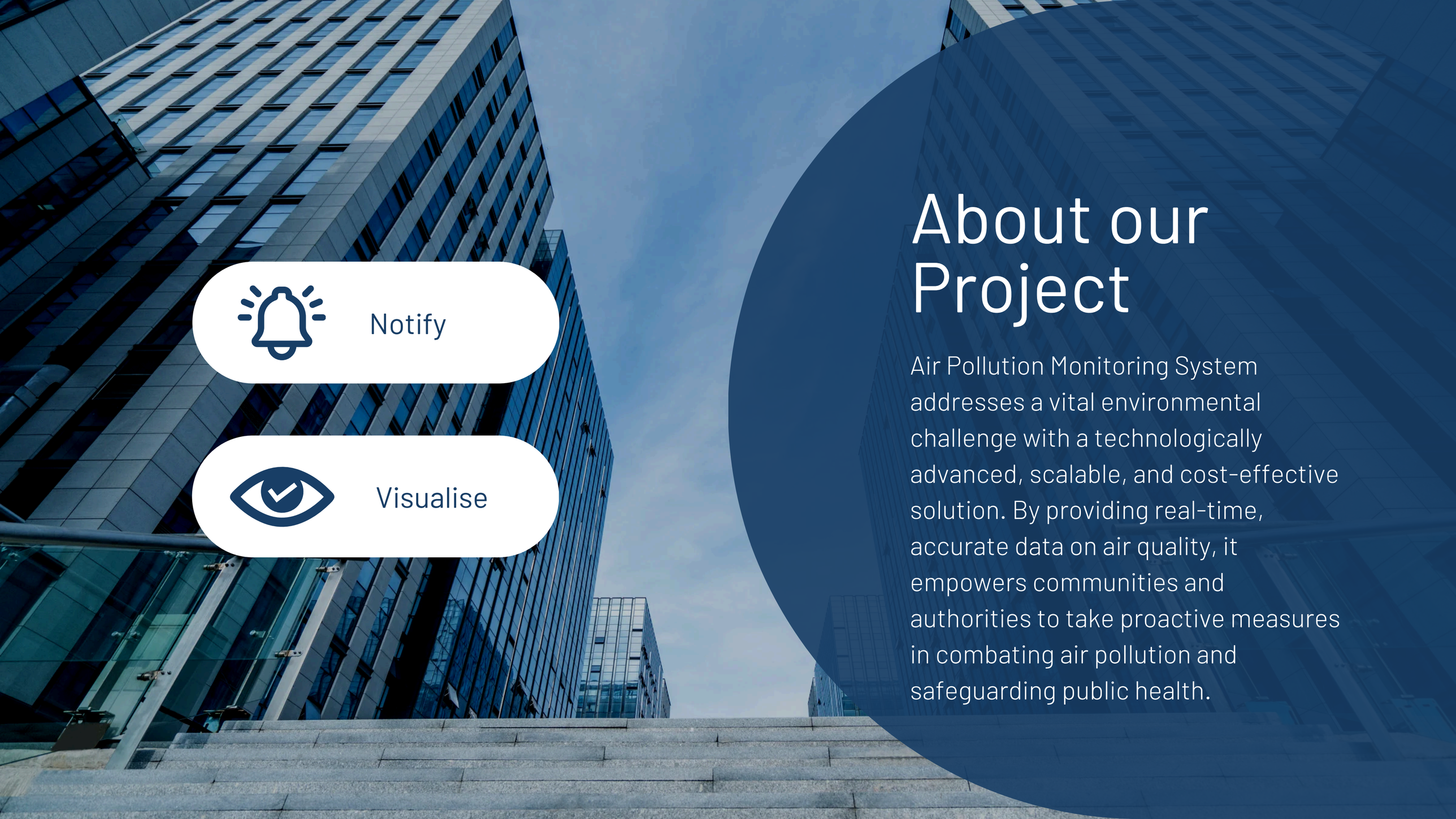
VISIONAIR

IOT - Air Qality Monitoring System



About our Project

Air Pollution Monitoring System addresses a vital environmental challenge with a technologically advanced, scalable, and cost-effective solution. By providing real-time, accurate data on air quality, it empowers communities and authorities to take proactive measures in combating air pollution and safeguarding public health.

Relevant **Statistics**

Motor Vehicles

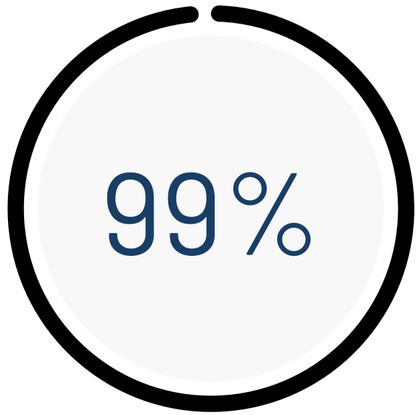
Motor vehicles contribute 80% of air pollution in the Philippines.

IQAir | First in Air Quality

Stationary Sources

Comes from stationary sources such as factories and the open burning of organic matter.

IQAir | First in Air Quality



Global Population

Global population in areas where air quality levels do not meet WHO guidelines.

World Health Organization

Premature Deaths

Low- and middle-income countries, with the highest burden in the WHO South-East Asia and Western Pacific Regions.

World Health Organization

Our **Goals**

Provide Accurate Readings

Awareness

Promote Air Quality

Engagement

Solution to Problem

Visualisation

Readings

The results are visualised in the form of graphs making it easy for authorities and the public to interpret the data and make informed decisions.

Alert System

Notify

Integrated into the monitoring platform to notify users when levels exceed safe thresholds, enabling timely actions to mitigate pollution exposure.

IoT Connectivity

Accessibillity

The real-time data is accessible via a web-based dashboard or a mobile application, allowing users to monitor air quality from anywhere at any time.


Traffic Sources Overview
Map Overlay
Direct Traffic
Visitors
2,958

Planning

Design and Planning

- Gather and analyze user requirements and specifications for the software.
- Create a detailed requirements document outlining functional and non-functional requirements, and acceptance criteria.

Hardware Setup

- Hardware simulation, finalisation, and setup.

Software Development

- Develop the software components/modules based on the system design and requirements specifications.
- Follow coding standards, best practices, and coding guidelines established during the system design phase.
- Use appropriate programming languages, frameworks, and tools as per the project requirements.

Integration, Testing, and Prototyping

- Verify that the integrated system operates seamlessly and meets the requirements outlined in the design phase.
- Conduct integration testing to validate interactions between modules, data flow, and overall system functionality.
- Address any integration issues or conflicts that arise during testing.
- Perform a range of testing activities to validate the software's functionality, performance, and reliability.
- Iterate on the prototype based on feedback and refine the software design and requirements as necessary.

Deployment and Maintenance

- Prepare the software for deployment to the production environment, following deployment procedures and guidelines.
- Conduct deployment testing to verify that the software functions correctly in the production environment and is compatible with the target infrastructure.
- Monitor system performance, and compliance with industry standards and regulations.
- Implement preventive maintenance measures to proactively address potential issues and ensure system reliability.

Materials & Methods

Our **Materials**

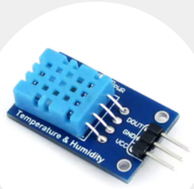
Includes Breadboard, Jumper Wires, and Enclosure*

ESP32

Serves as the central processing unit, executing the code to control the air pollution monitoring system.

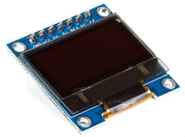
MQ-Series (135, 4, 7)

Detect gases like ammonia, NO2, alcohol, benzene, smoke, and CO2 in the air.



DHT11

Measures the ambient temperature and humidity levels.



LCD

Shows real-time data from the sensors, allowing users to visually monitor air quality metrics.

Sensors

These are the sensors used:

DHT11

Temperature and Humidity

Measures the ambient temperature and humidity levels

MQ-135

Ammonia, NO, Alcohol, Benzene

Detect various harmful gases like ammonia, NO, alcohol, benzene, smoke, and CO2 in the air.

MQ-4

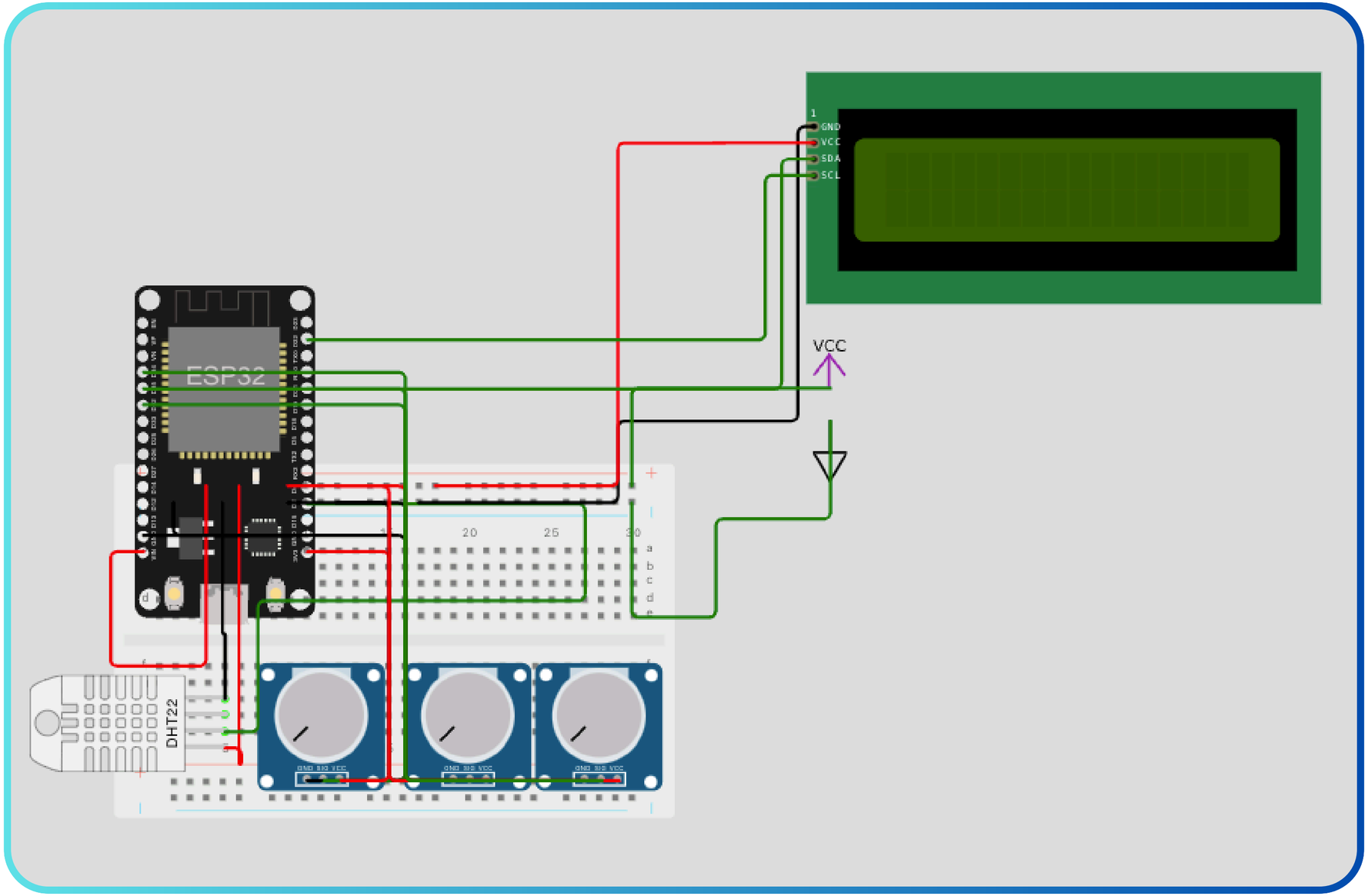
Methane and LPG

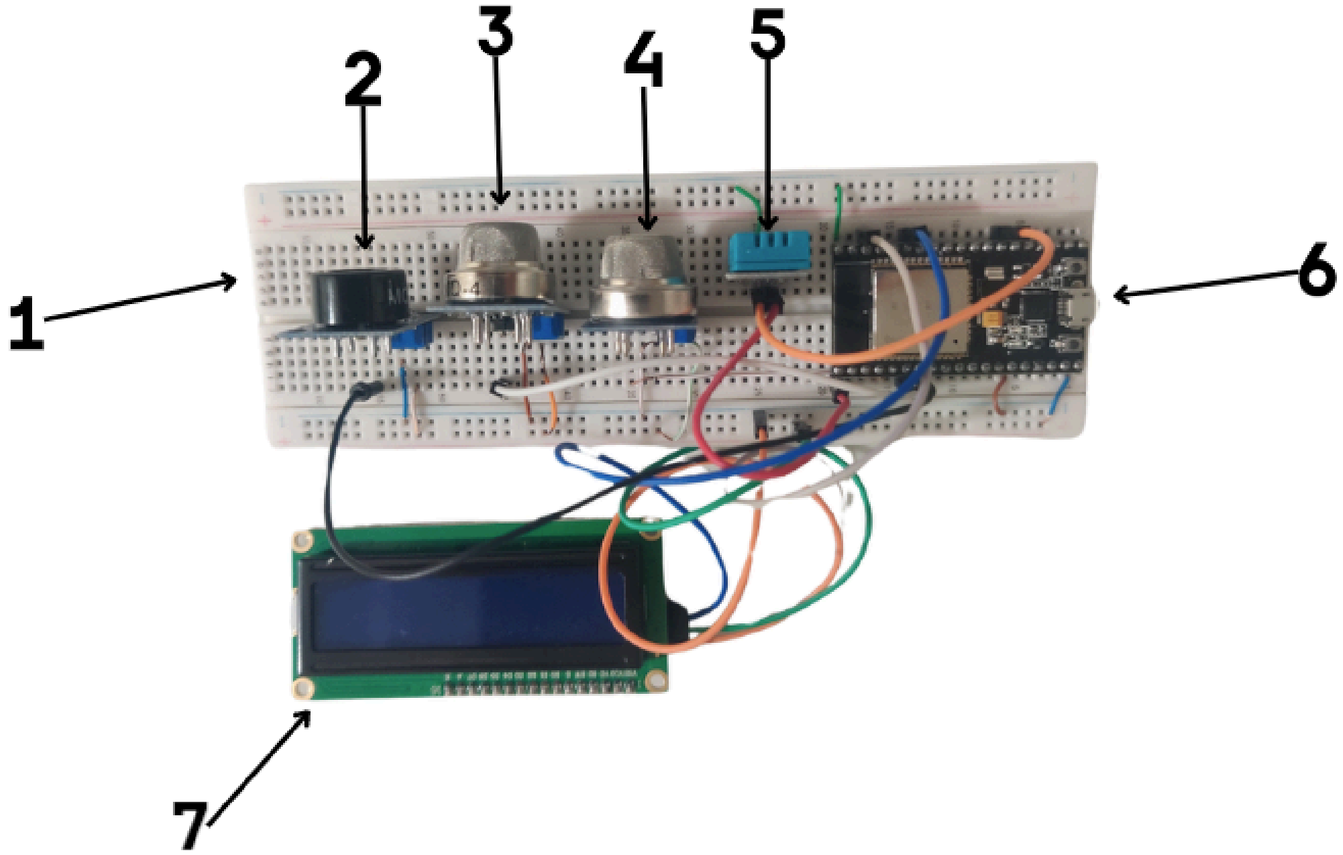
Detects Methane, and can also sense other gases such as LPG and natural gas.

MQ-7

CO and Hydrogen Gas

Detects Carbon Monoxide (CO) and can also sense hydrogen gas.


1
GND
VCC
SDA
SCL
VCC
ESP32
DHT22
20
25
30
a
b
c
d
e

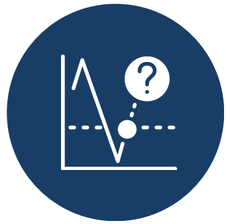


a breadboard (1)
an MQ-7 gas sensor (2)
an MQ-4 gas sensor (3)
an MQ-135 gas sensor (4)
a DHT11 sensor (5)
an ESP32 module (6)
an LCD display (7)

System & Algorithm

Algorithm

Threshold



Triggers actions or alerts when sensor readings exceed predefined limits, ensuring timely responses to significant changes in air quality.

SMA

Simple Moving Average

Smooths sensor data by averaging a set number of recent readings, filtering out short-term fluctuations to reveal clearer trends.

Hysteresis

Introduces buffer zones around thresholds to prevent frequent toggling of actions due to minor fluctuations, ensuring more stable system responses.

Threshold-Based Algorithm

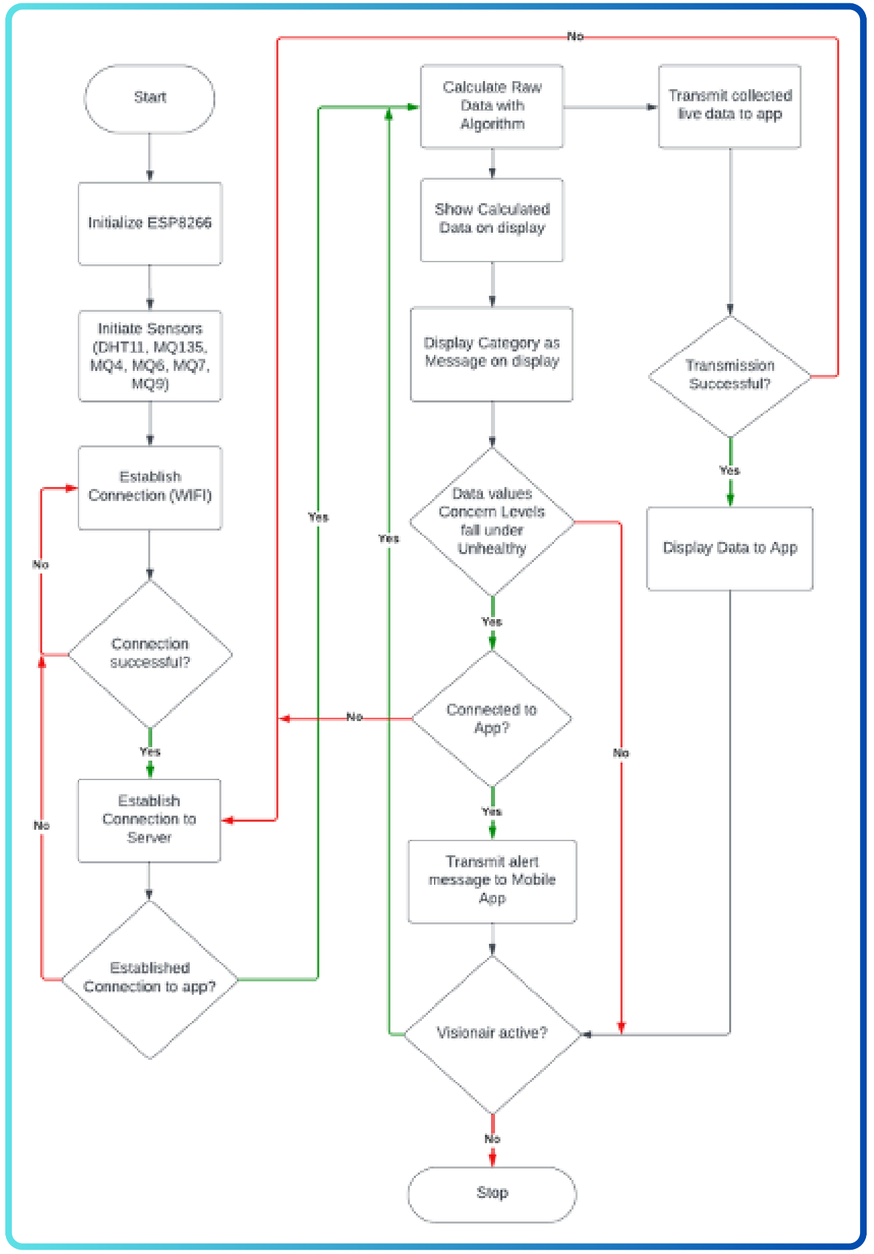
```
initialise variable threshold_value to predefined limit
initialise array alert_messages

for each data_point in sensor_data:
    if data_point > threshold_value:
        append "High pollution alert!" to alert_messages
    else:
        append "Pollution levels normal" to alert_messages

return alert_messages
```

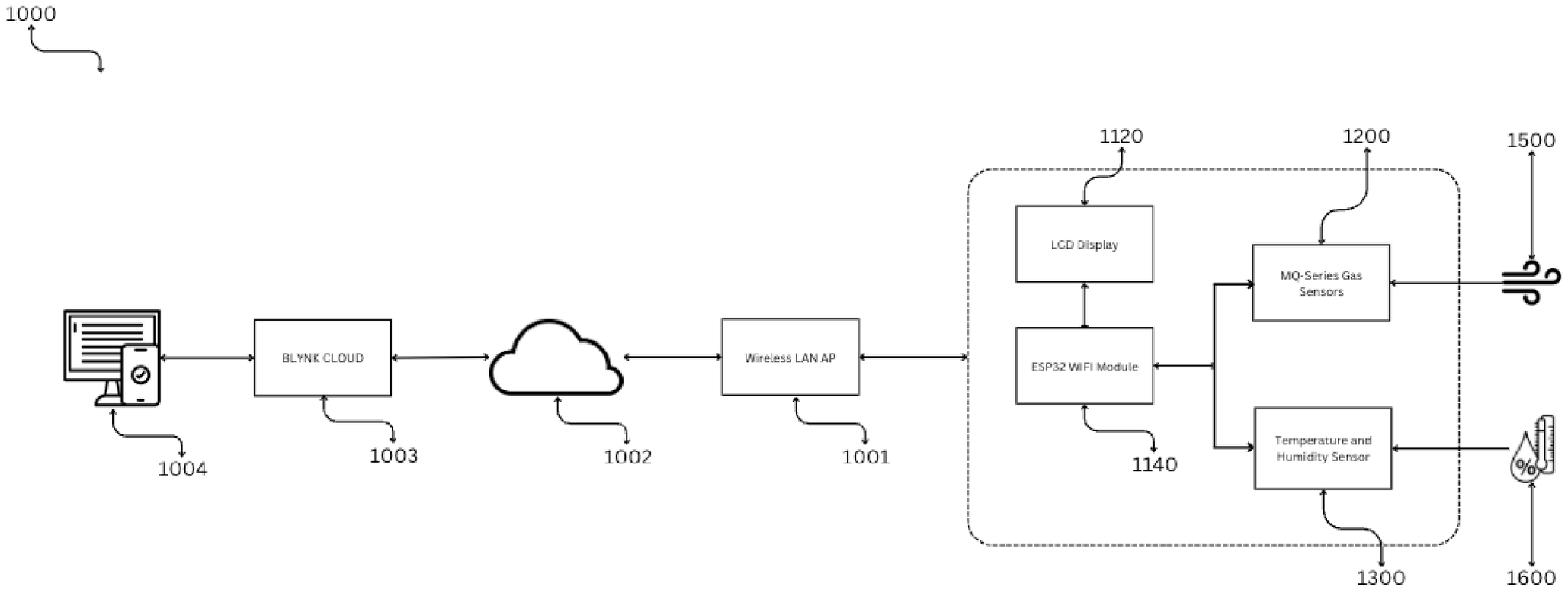
SMA Algorithm

```
initialise array data_window
initialise variable window_size to desired window length
initialise variable sum to 0
initialise array sma_values

for each new data_point in sensor_data:
    if data_window has less than window_size elements:
        append data_point to data_window
        add data_point to sum
        sma = sum / length of data_window
    else:
        remove the oldest data point from data_window
        subtract the oldest data point from sum
        append data_point to data_window
        add data_point to sum
        sma = sum / window_size

    append sma to sma_values

return sma_values
```

Hysteresis Algorithm

```
initialize variable upper_threshold to high limit
initialize variable lower_threshold to low limit
initialize variable alert_status to False
initialize array hysteresis_alerts

for each data_point in sensor_data:
    if data_point > upper_threshold:
        if alert_status is False:
            append "High pollution alert!" to hysteresis_alerts
            set alert_status to True
    elif data_point < lower_threshold:
        if alert_status is True:
            append "Pollution levels normal" to hysteresis_alerts
            set alert_status to False

return hysteresis_alerts
```


Start
Initialize ESP8266
Initiate Sensors (DHT11, MQ135, MQ4, MQ6, MQ7, MQ9)
Establish Connection (WIFI)
Connection successful?
No
Yes
Establish Connection to Server
Established Connection to app?
No
Yes
Calculate Raw Data with Algorithm
Show Calculated Data on display
Display Category as Message on display
Data values Concern Levels fall under Unhealthy
Yes
Connected to App?
No
Yes
Transmit alert message to Mobile App
Visionair active?
No
Stop
Transmit collected live data to app
Transmission Successful?
No
Yes
Display Data to App


1000
1120
1200
1500
LCD Display
MQ-Series Gas Sensors
BLYNK CLOUD
Wireless LAN AP
ESP32 WIFI Module
Temperature and Humidity Sensor
1004
1003
1002
1001
1140
1300
1600

Blynk Cloud


My organization - 4734KT
AirPollutionMonitoring
Online
Daniel
My organization - 4734KT
Edit
Live
1h
6h
1d
1w
Air Quality
96
0
1,000
Temperature
30
0
100
Humidity
44
0
100
Methane
0
0
100
Chart
Air Quality
Chart
Temperature


1:55
AirPollutionMonitoring
79
0
600
30
0
100
44
0
100
0
0
100
407
0
1000
Stream
100.00
92.50

POLLUTION SHOULD NEVER BE THE PRICE OF PROSPERITY

AL GORE

Fin

Canarias | Sanchez | Mandap