PHASE 5:PROJECT DOCUMENTATION

PROJECT:AIR QUALITY MONITORING

TEAM LEADER NAME :	PAVITHRA R
TEAM MEMBER NAME1:	PRIYDHARSHINI K
TEAM MEMBER NAME2:	RAMYA R
TEAM MEMBER NAME3:	SHANMUGA PRIYA G

PROJECT TITLE : AIR QUALITY MONITORING

DOMAIN : IOT

PROBLEM STATEMENT:

- Air pollution is one of the biggest threats to the present-day environment.
- Everyone is being affected by air pollution day by day including humans, animals, crops, cities, forests and aquatic ecosystems.
- Besides that, it should be controlled at a certain level to prevent the increasing rate of global warming.

Objective:

- To measure and display temperature and humidity level of the environment.
- To combine advanced detection technologies to produce an air quality sensing system
- with advanced capabilities to provide low cost comprehensive monitoring.
- To display the sensed data in user friendly format in LCD display panel.

Proposed concept:

- This project aims to design an IOT-based air pollution monitoring system using the internet from anywhere using a computer or mobile to monitor the air quality of the surroundings and environment.

Innovation:

This project proposes an idea to install monitoring applications on smartphones. It is innovative because it provides easy access to the public to monitor real time air quality in their area. It uses low cost and readily available devices such as a dust sensor, carbon monoxide gas sensor, carbon dioxide gas sensor, and nitrogen dioxide gas sensor. For controlling these sensors, microcontrollers are used and the microcontrollers also act as transmitter to transmit the data to the cloud database. The information on air quality can be accessed through a smartphone app in real time.

IoT Devices Designs:

Air pollution is one of the biggest threats to the present-day environment. Everyone is being affected by air pollution day by day including humans, animals, crops, cities, forests and aquatic ecosystems. Besides that, it should be controlled at a certain level to prevent the increasing rate of global warming. This project aims to design an IOT-based air pollution monitoring system using the internet from anywhere using a computer or mobile to monitor the air quality of the surroundings and environment. There are various methods and instruments available for the measurement and monitoring quality of air. The IoT-based air pollution monitoring system would not only help us to monitor the air quality but also be able to send alert signals whenever the air quality deteriorates and goes down beyond a certain level. In this system, NodeMCU plays the main controlling role. It has been programmed in a manner, such that, it senses the sensory signals from the sensors and shows the quality level via led indicators. Besides the harmful gases (such as CO2, CO, smoke, etc) temperature and humidity can be monitored through the temperature and humidity sensor by this system. Sensor responses are fed to the NodeMCU which displays the monitored data in the ThingSpeak cloud which can be utilized for analyzing the air quality of that area.The following simple flow diagram indicates the working mechanism of the IoT-based Air Pollution Monitoring System.

Components Used

Hardware Components

1. NodeMCU V3
2. DHT11 Sensor Module
3. MQ-135 Gas Sensor Module
4. Veroboard(KS100)
5. Breadboard
6. Connecting Wires
7. AC-DC Adapters
8. LEDs emitting green, yellow and red colours
9. Resistors

SOFTWARE COMPONENTS

1. ThinkSpeak Cloud
2. Arduino IDE

NodeMCU V3

- NodeMCU V3 is an open-source ESP8266 development kit, armed with the CH340G USBTTL Serial chip. It has firmware that runs on ESP8266 Wi-Fi SoC from Espressif Systems.

- Whilst cheaper, CH340 is super reliable even in industrial applications. It is tested to be stable on all supported platforms as well. It can be simply coded in Arduino IDE. It has a very low current consumption between 15 µA to 400 mA

DHT11 Sensor Module

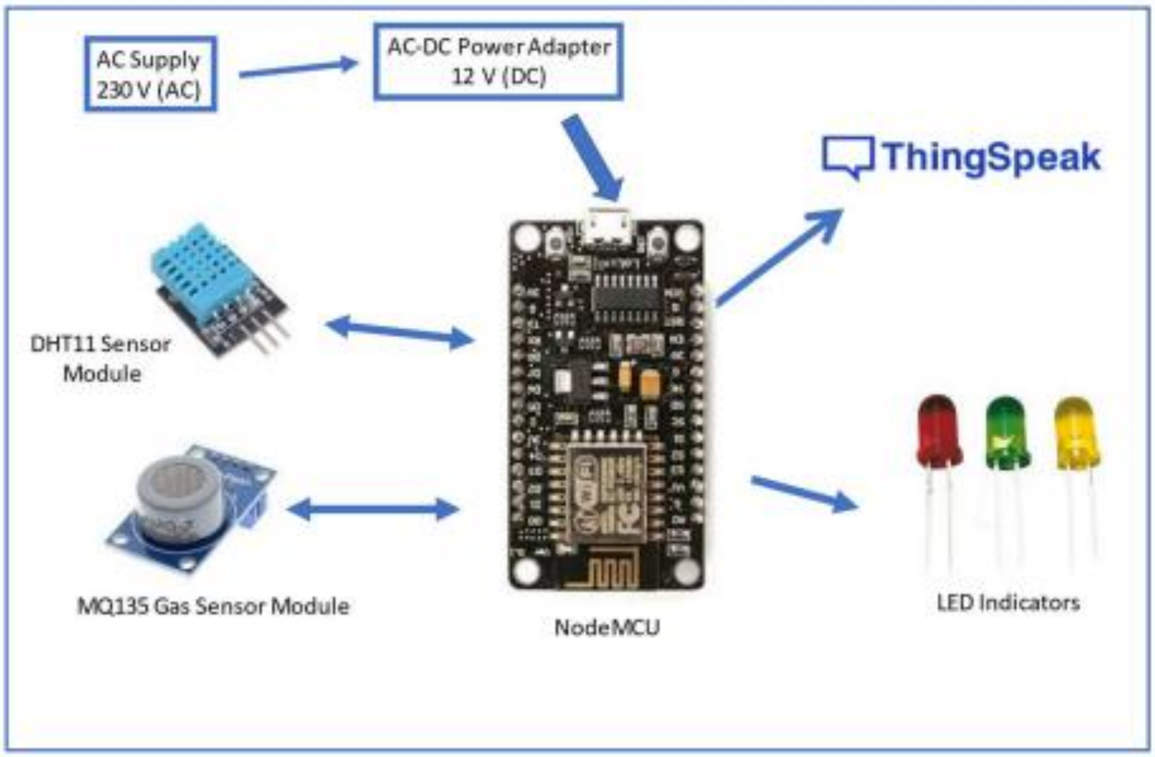
- The DHT11 is a temperature and humidity sensor that gives digital output in terms of voltage. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air.
- we need to supply a voltage of 5V (DC) to the Vcc pin and ground it to the GND pin. The sensor output can be easily read from the Data pin in terms of voltage (in digital mode).

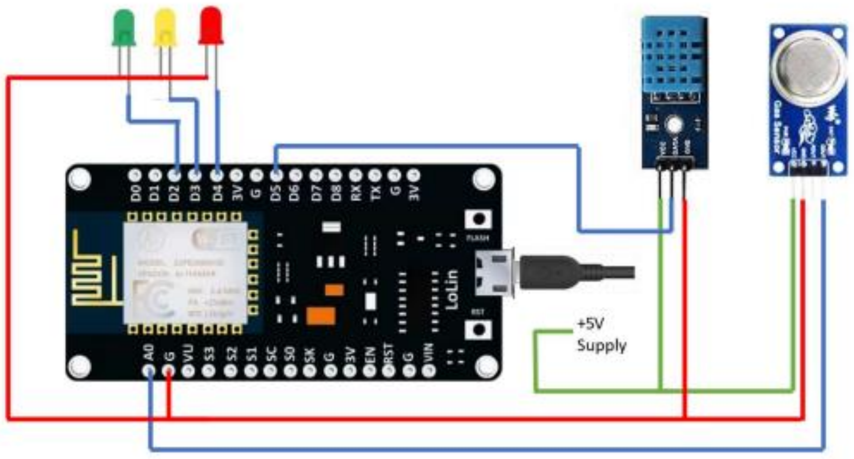
Humidity Measurement:

- The humidity sensing capacitor has two electrodes with a moisture-holding substrate as a dielectric between them .
- The IC measure, process these changed resistance values and then converts them into digital form.
- Change in the capacitance value occurs with the change in humidity levels.

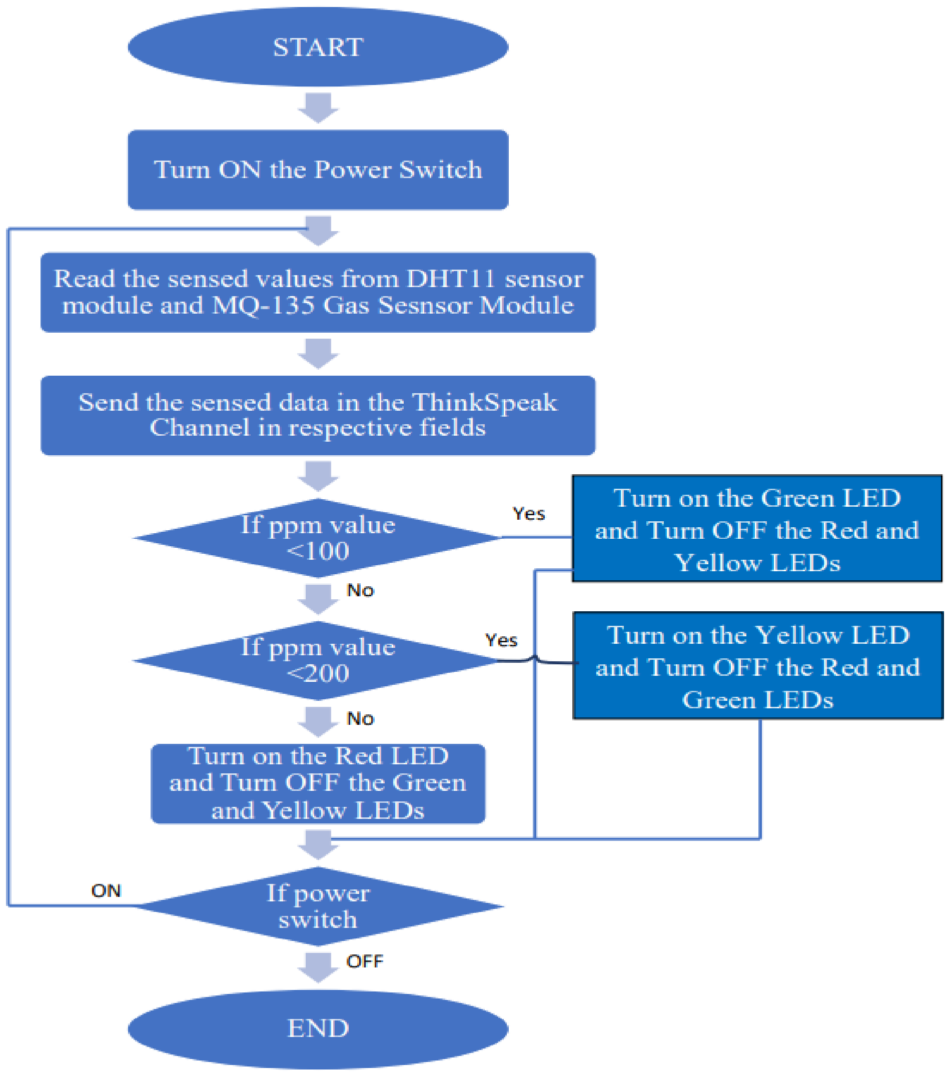
Temperature Measurement:

- For measuring the temperature, the DHT11 sensor uses a negative temperature coefficient thermistor, which causes a decrease in its resistance value with an increase in temperature.
- To get a wide range of resistance values, the sensor is made up of semiconductor ceramics or polymer.




D0 D1 D2 D3 D4 3V G D5 D6 D7 D8 RX TX G 3V
A0 G VU S3 S2 S1 SC S0 SK G 3V EN RST G VIN
FLASH
RST
LoLin
+5V Supply

Flow diagram


START
Turn ON the Power Switch
Read the sensed values from DHT11 sensor module and MQ-135 Gas Sesnsor Module
Send the sensed data in the ThinkSpeak Channel in respective fields
If ppm value <100
Yes
Turn on the Green LED and Turn OFF the Red and Yellow LEDs
No
If ppm value <200
Yes
Turn on the Yellow LED and Turn OFF the Red and Green LEDs
No
Turn on the Red LED and Turn OFF the Green and Yellow LEDs
ON
If power switch
OFF
END

Deploying IoT devices

NodeMCU plays the main controlling role in this project. It has been programmed in a manner, such that, it senses the sensory signals from the sensors and shows the quality level via led indicators. The DHT11 sensor module is used to measure the temperature and the humidity of the surroundings.With the help of the MQ-135 gas sensor module, air quality is measured in ppm. These data are fed to the ThinkSpeak cloud over the internet. We have also provided LED indicators to indicate safety levels.

STEP 1. Firstly, the calibration ofthe MQ-135 gas sensor module is done. The sensor is set to preheat for 24 minutes. Then the software code is uploaded to the NodeMCU followed by the hardware circuit to calibrate the sensor has been performed.

STEP 2. Then, the DHT11 sensor is set to preheat for 10 minutes.

STEP 3. The result of calibration found in STEP 1 is used to configure the final working code.

STEP 4. The final working code is then uploaded to the NodeMCU.

STEP 5. Finally, the complete hardware circuit is implemented.

HARDWARE MODEL

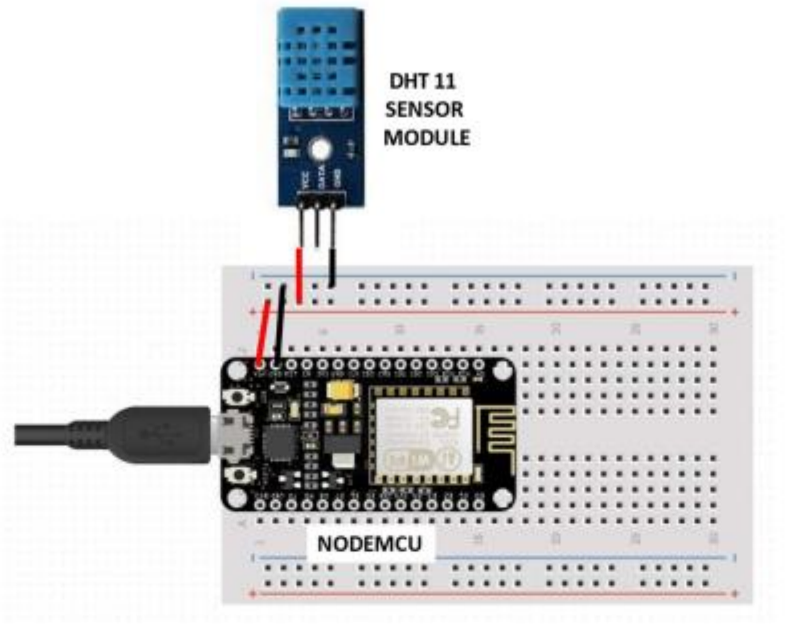
Hardware Model to Preheat DHT11 Sensor Module .

The following steps were performed to preheat the DHT11 sensor module:

STEP 1 : The Vcc pin of the DHT11 sensor module was connected with the VU pin of NodeMCU.

STEP 2 : The Gnd pin of the DHT11 sensor module was connected with the Gnd pin of NodeMCU.

STEP 3 : The NodeMCU is powered with a 12V DC via AC-DC adapter for 20 minutes. This setup has been disconnected.



(Circuit Diagram to Preheat the DHT11 sensor module)

Hardware Model to Preheat and Calibrate MQ-135 Gas Sensor Module

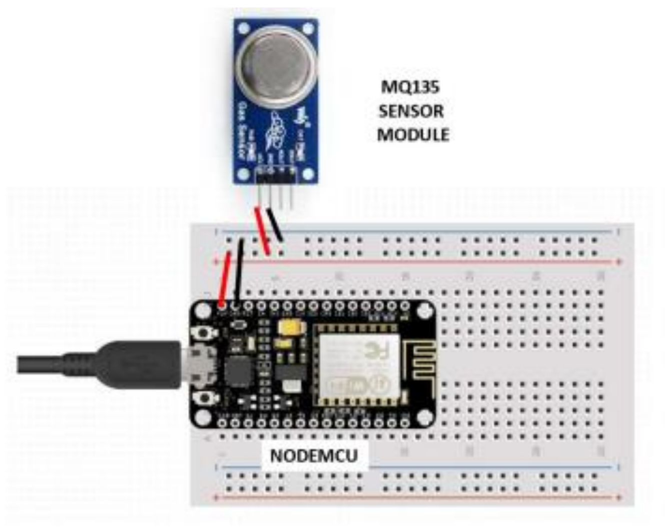
The following steps were performed to preheat the MQ-135 gas sensor module

STEP 1 : The Vcc pin of the MQ-135 gas sensor module was connected with the VU pin of NodeMCU.

STEP 2 : The Gnd pin of the MQ-135 gas sensor module was connected with the Gnd pin of NodeMCU.

STEP 3 : The NodeMCU is powered with a 12V DC via AC-DC adapter for a day.

STEP 4 : The setup was then disconnected



(Circuit Diagram to Preheat the MQ-135 Gas sensor module)

Final Hardware Model The following steps were performed to execute the project:

STEP 1 : The Vcc pin of the MQ-135 gas sensor module and DHT11 sensor module was connected via Veroboard with an adapter delivering around 5V.

STEP 2 : The Gnd pin of the MQ-135 gas sensor module, DHT11 sensor module and the cathode of the LED indicators was connected via Veroboard with the Gnd pin of the NodeMCU.

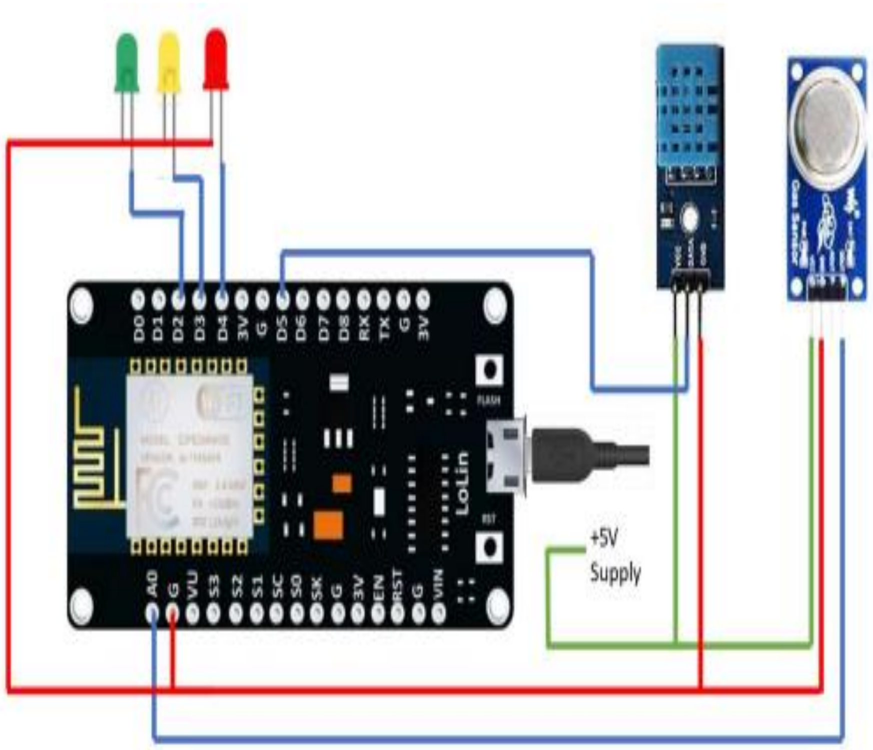
STEP 3 : The analog DATA pin of the MQ-135 gas sensor module was connected with the A0 Pin of the NodeMCU.

STEP 4 : The DATA pin of the DHT11 sensor module was connected with the D0 pin of the NodeMCU.

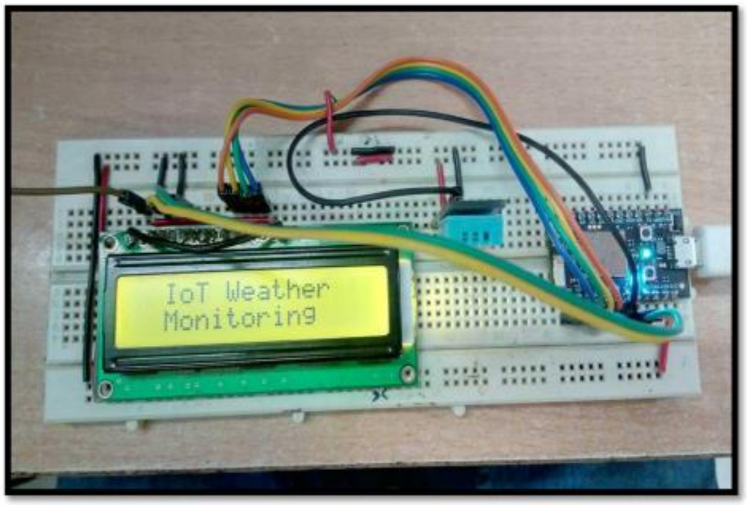
STEP 5 : The anode of the three LED indicators (green, yellow, and red) were connected to the D2, D3, and D4 pins of the NodeMCU respectively.

STEP 6 : The software code to execute the project was then uploaded to the NodeMCU.

STEP 7 : The setup was then powered with 9V DC via AC-DC adapter.



Circuit Diagram of the setup



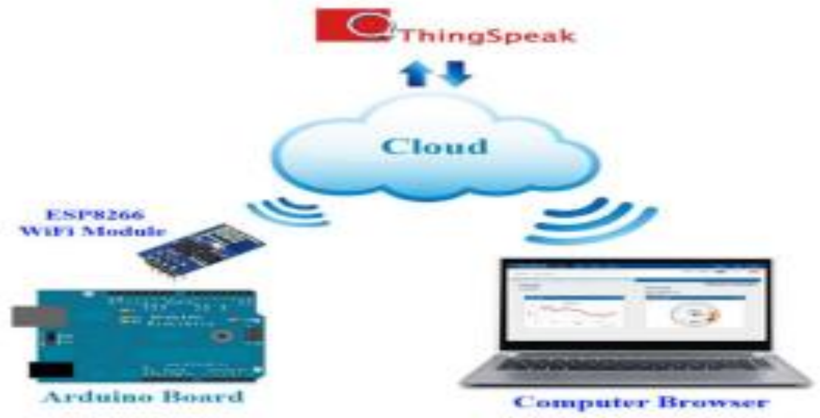
DEVELOPING A PYTHON SCRIPT:

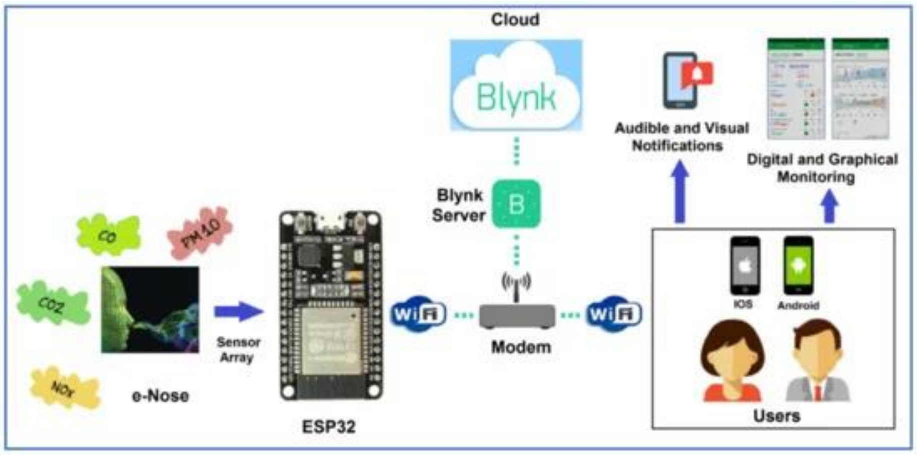
```
import streams

# import the wifi interface
from wireless import wifi

# import wifi support
from espressif.esp8266wifi import esp8266wifi as wifi_driver

streams.serial()

# init the wifi driver!
# The driver automatically registers itself to the wifi interface
# with the correct configuration for the selected board
wifi_driver.auto_init()

# use the wifi interface to link to the Access Point
# change network name, security and password as needed
print("Establishing Link...")
try:
   # FOR THIS EXAMPLE TO WORK, "Network-Name" AND "Wifi-Password"
MUST BE SET
```

```
    # TO MATCH YOUR ACTUAL NETWORK CONFIGURATION
    wifi.link("Network-name",wifi.WIFI_WPA2,"password")
except Exception as e:
    print("ooops, something wrong while linking :(", e)
    while True:
        sleep(1000)
```

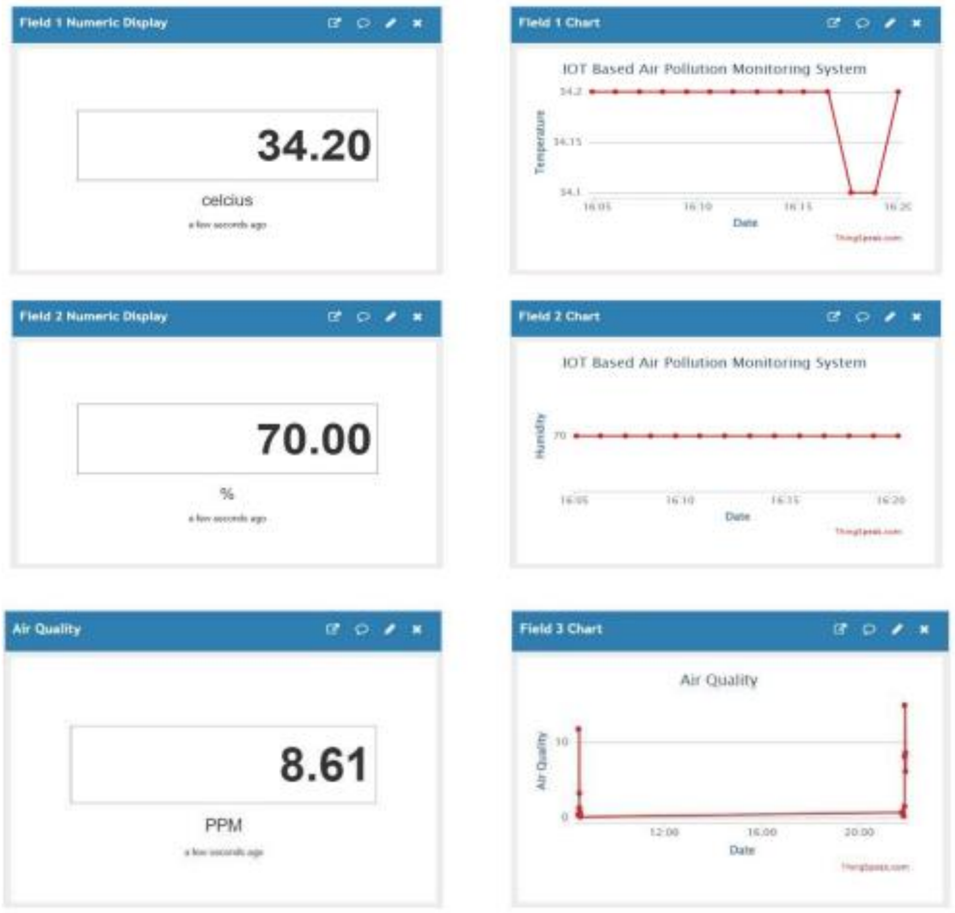
CONNECTIVITY:

- ThingSpeak is open-source software written in Ruby which allows users to communicate with internet-enabled devices.
- It facilitates data access, retrieval and logging of data by providing an API to both the devices and social network websites.
- ThingSpeak was originally launched by ioBridge in 2010 as a service in support of IoT applications.
- ThingSpeak has integrated support from the numerical computing software MATLAB from MathWorks, allowing ThingSpeak users to analyse and visualize uploaded data using MATLAB without requiring the purchase of a MATLAB license from MathWorks.
-

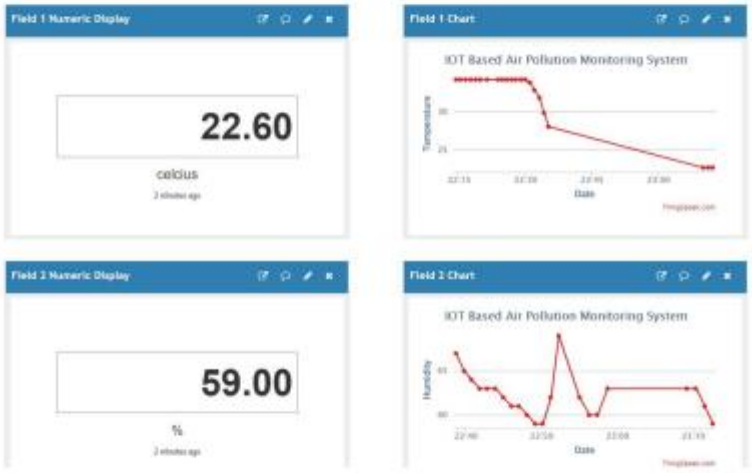


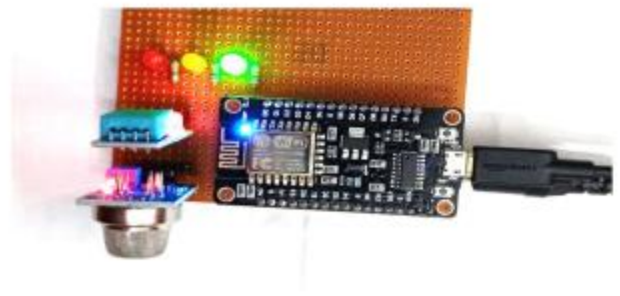


DATA SHARING PLATFORM:



INTEGRATION APPROACH:





To implement air quality monitoring system into a mobile app, we need to integrate various technologies and APIs.

Code snippet using the Google Maps API and Firebase Realtime Database to get air quality information and display it in a mobile app:

1.Setup the project:

```
// Initialize Firebase

FirebaseApp.initializeApp(this);


// Initialize Google Maps

MapView mapView = findViewById(R.id.mapView);

mapView.onCreate(savedInstanceState);

mapView.getMapAsync(new OnMapReadyCallback() {

   @Override

   public void onMapReady(GoogleMap googleMap) {

      // Handle map initialization here

   }
```

```
});
DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference("air_quality");

databaseReference.addValueEventListener(new
ValueEventListener() {
   @Override
   public void onDataChange(DataSnapshot dataSnapshot) {
      for (DataSnapshot data : dataSnapshot.getChildren()) {
         // Parse and process air quality data
         AirQuality airQuality = data.getValue(AirQuality.class);

         // Use the data to display markers on the map
         LatLng location = new LatLng(airQuality.getLatitude(),
airQuality.getLongitude());
         MarkerOptions markerOptions = new MarkerOptions()
               .position(location)
               .title("AQI: " + airQuality.getAqi());

         googleMap.addMarker(markerOptions);
      }
```

```
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Handle database error
    }
});
public class AirQualityData {
    private String aqi;
    private String pm25;
    private String pm10;
    private String co2;

    public AirQualityData() {
        // Default constructor required for Firebase
    }

    public AirQualityData(String aqi, String pm25, String pm10,
String co2) {
        this.aqi = aqi;
        this.pm25 = pm25;
        this.pm10 = pm10;

        this.co2 = co2;

    }


    public String getAqi() {

        return aqi;

    }


    public String getPm25() {

        return pm25;

    }


    public String getPm10() {

        return pm10;

    }


    public String getCo2() {

        return co2;

    }

}
```

2.Prepare the Firebase database structure:

```
// Initialize Firebase Realtime Database reference
DatabaseReference airQualityRef =
FirebaseDatabase.getInstance().getReference("air_quality");

// Sample data
double latitude = 37.7749; // Replace with actual latitude
double longitude = -122.4194; // Replace with actual longitude
int aqi = 50; // Replace with actual AQI value
long timestamp = System.currentTimeMillis(); // Timestamp of
data

// Create an AirQuality object
AirQuality airQuality = new AirQuality(latitude, longitude, aqi,
timestamp);

// Generate a unique key for the new data entry
String entryKey = airQualityRef.push().getKey();

// Save the data to Firebase
airQualityRef.child(entryKey).setValue(airQuality);
```

This code sets up a simple mobile app with a Google Map embedded, connecting to a Firebase Realtime Database to fetch air quality information.

Cost Estimation of the Project:

For making the project we have used the following components . As per the pricing on the online websites for electronic components, we have formulated a cost estimation.

Components	**Price(in Rs)**
NodeMcu V3	288
DHT11 Sensor Module	120
MQ135 Gas Sensor Module	135
Connecting Wires	60
LEDs(Red,Green & Yellow)	9
AC-DC Power Adapter	120
Female PCB Berg Terminal and cable	80
Veroboard	100
Breadboard	70
TOTAL	**982**

Conclusion:

In this project IoT based on measurement and display of Air Quality Index (AQI), Humidity and Temperature of the atmosphere have been performed. From the information obtained from the project, it is possible to calculate Air Quality in PPM. The disadvantage of the MQ135 sensor is that specifically it can’t tell the Carbon Monoxide or Carbon Dioxide level in the atmosphere, but the advantage of MQ135 is that it is able to detect smoke, CO, CO2, NH4,etc harmful gases.

After performing several experiments, it can be easily concluded that the setup is able to measure the air quality in ppm, the

temperature in Celsius and humidity in percentage with considerable accuracy. The results obtained from the experiments are verified through Google data. Moreover, the led indicators help us to detect the air quality level around the
setup. However, the project experiences a drawback that is it cannot measure the ppm values of the pollutant components separately. This could have been improved by adding gas sensors for different pollutants. But eventually, it would increase the cost of the setup and not be a necessary provision to monitor the air quality. Since it's an IOT-based project, it will require
a stable internet connection for uploading the data to the ThinkSpeak cloud. Therefore, it is possible to conclude that the designed prototype can be utilized for air quality, humidity and temperature of the surrounding atmosphere successfully.