AN PRESENTATION

ON

AIR QUALITY MONITORING

SUBMITTED BY

k.Badri Prasath

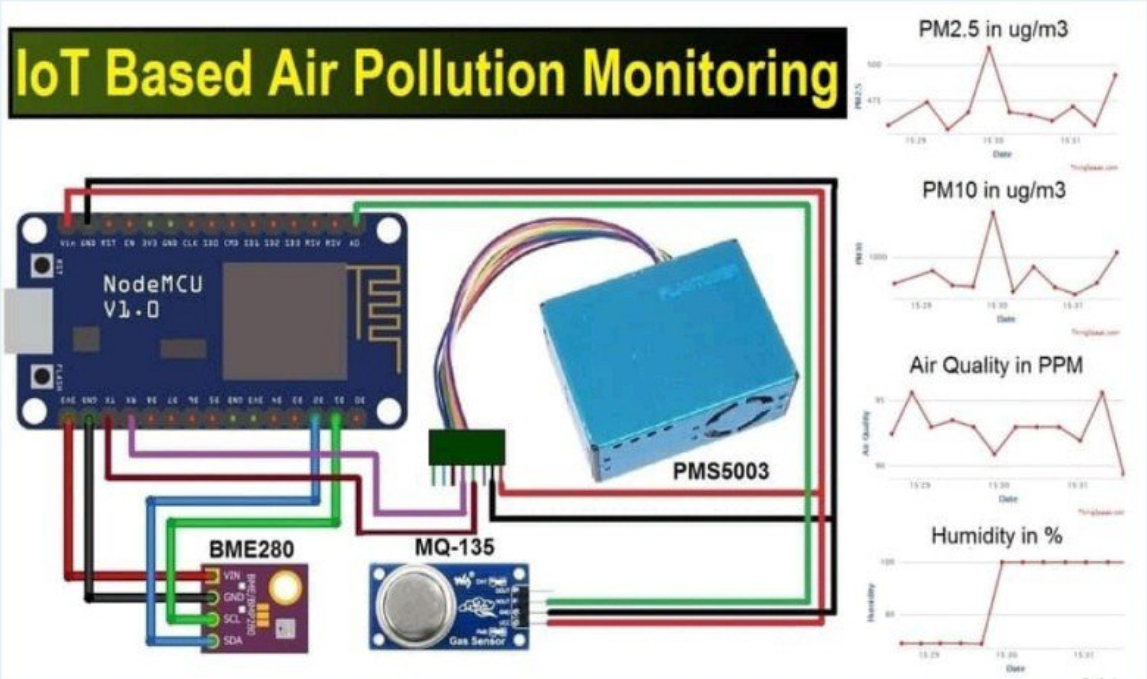
P.Ganapathi

K.Anjali

P.Indhumathi

K.Karthika

IoT Based Air Quality Monitoring with Node Red Services with ESP8266



In this project we are going to make an IoT Based Air Pollution/Quality Monitoring System with ESP8266, PM2.5 Particulate Matter Sensor, MQ-135 Air Quality Sensor & BME280 Barometric Pressure Sensor.

Abstract

This paper present an implementation of MQTT based air quality monitoring system. The air quality measurement device is a hardware using ESP8266 NodeMCU that connects to sensors for measuring temperature, humidity, concentration of carbon monoxide (CO), ozone gas (O3), and PM2.5.

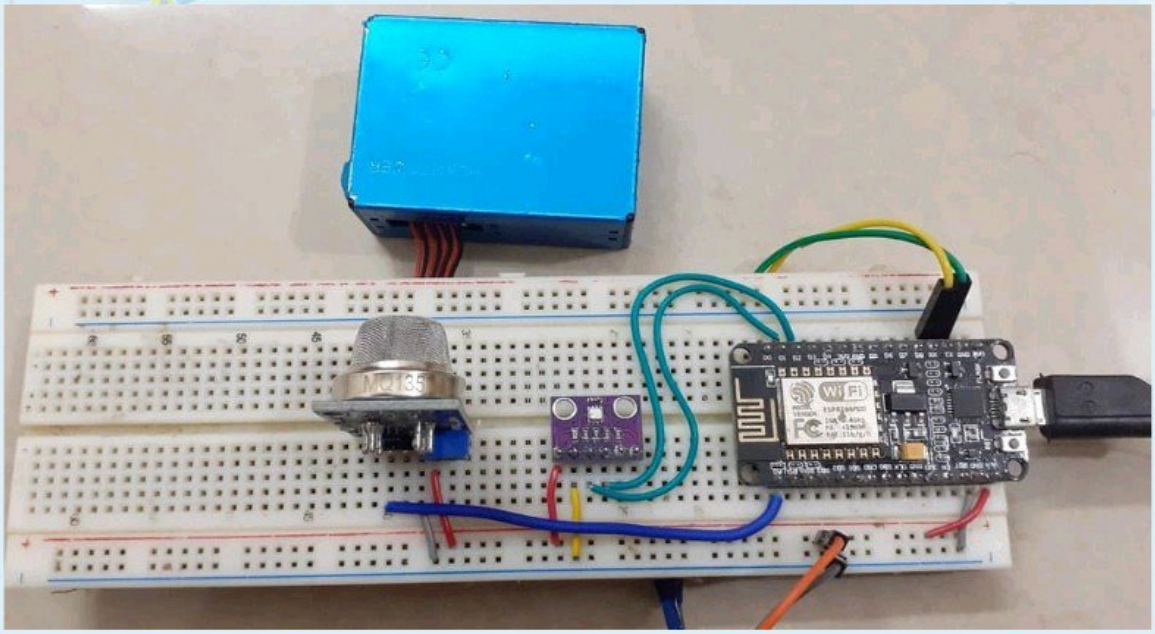
The firmware of device makes the device act as a publisher that reads the sensor data and sends them to MQTT Broker. Node-RED is used to be a subscriber that subscribes to receive data from MQTT Broker. With Node-RED, we can easily make a flow to manage and handle received data.

Then, Node-RED will send data to the air quality monitoring dashboard which is a responsive web application to display data in gauge, text and chart user interface.

In addition, when the value of some data exceed the configured range, Node-RED will send an alarm message via LINE Notify to notify users.

Overview

This is a simple prototype for an Environmental IoT Air Pollution/Quality Monitoring System for monitoring the concentrations of major air pollutant gases. The system uses 3 sensors like PMS5003 PM2.5 Particulate Matter Sensor, MQ-135 Air Quality Sensor, BME280 Barometric Pressure Sensor. In this IoT project, you can monitor the pollution level from anywhere using your computer or mobile.



PMS5003 PM2.5 Particulate matter sensor from Plantpower measure particle concentration in PM1.0, PM2.5 & PM10. This MQ-135 Air Quality Sensor measures concentrations of gases such as CO, CO2, SO2, and NO2 and gives the result in PPM (Part per Million). Similarly, BME280 Measures environmental Temperature, Pressure & Humidity.

Components Required

Following are the components required for making IoT-based Air Pollution/Quality Monitoring System. All the components can be easily purchased from Amazon. The components Purchased links are given as well.

Components Name

1 NodeMCU ESP8266 Board -1

2 PMS5003 PM2.5/PM10 Sensor -1

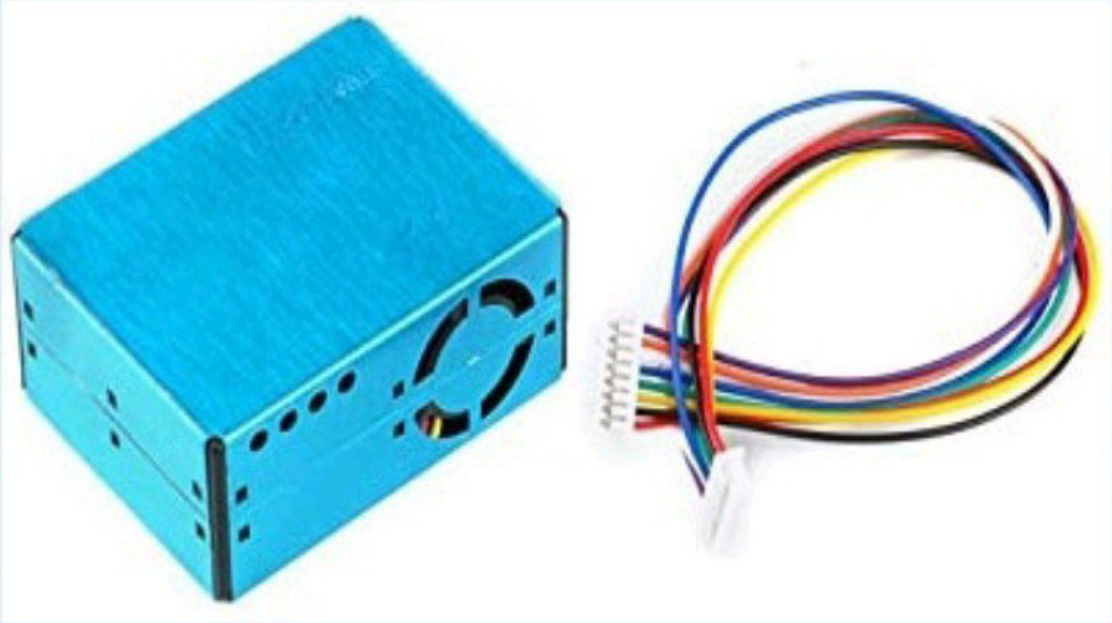
3 MQ-135 Air Quality Sensor -1

4 BME280 Sensor -1

5 Connecting Wires -10

6 Breadboard -1

PMS5003 PM2.5 Particulate Matter Sensor



The Plantower PMS5003 is a low-cost laser particle counter, one of a range of sensors by Plantower that also include the PMS1003, PMS3003, and PMS7003. PMS5003 is a kind of digital and universal particle concentration sensor, which can

be used to obtain the number of suspended particles in the air, i.e. the concentration of particles, and output them in the form of a digital interface. This sensor can be inserted into variable instruments related to the concentration of suspended particles in the air or other environmental improvement equipment to provide correct concentration data in time.

MQ-135 Air Quality Sensor

The MQ-135 gas sensor senses the gases like ammonia nitrogen, oxygen, alcohols, aromatic compounds, sulfide and smoke.

The MQ-3 gas sensor has a lower conductivity to clean the air as a gas sensing material. In the atmosphere, we can find polluting gases, but the conductivity of the gas sensor increases as the concentration of polluting gas increases. MQ-135 gas sensor can be implemented to detect the smoke, benzene, steam, and other harmful gases.

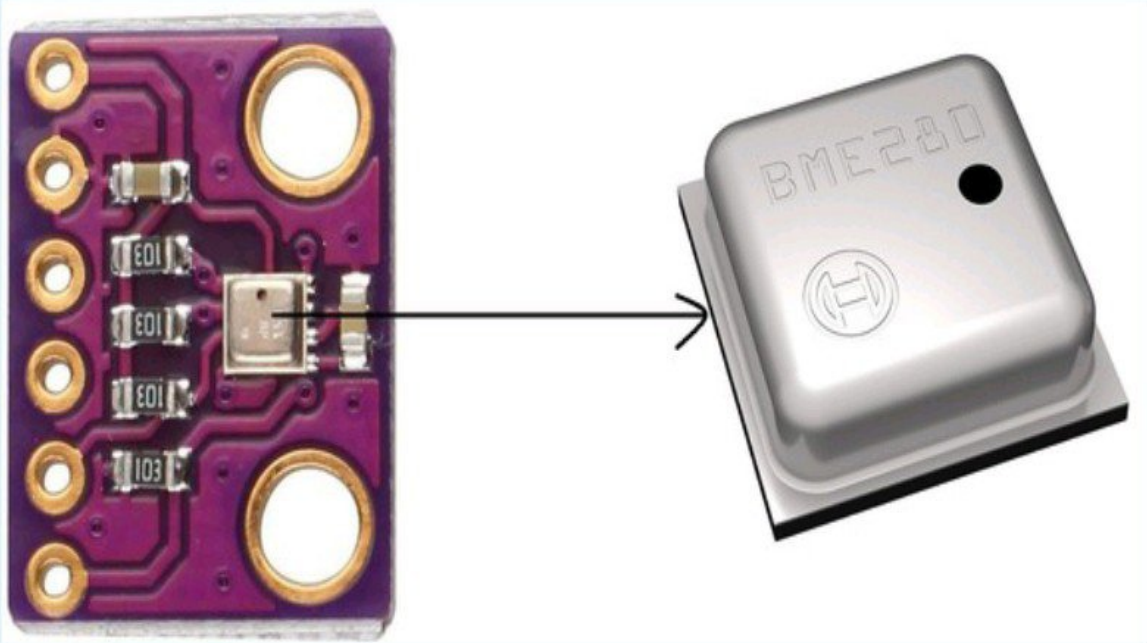
It has the potential to detect different harmful gases. It is at a low cost and particularly suitable for

Air quality monitoring applications.

The MQ135 sensor is a signal output indicator instruction. It has two outputs: analog output and TTL output. The TTL output is low signal light that can be accessed through the IO ports on the Microcontroller. The analog output is an concentration, i.e. increasing voltage is directly proportional to increasing concentration. This sensor has a long life and reliable stability as well.

BME280 Barometric Pressure Sensor

Bosch BME280 Humidity, Temperature & Pressure Sensor is an integrated environmental sensor which is very small-sized with low power consumption.This BME280 Atmospheric Sensor Breakout is the easy way to measure barometric pressure, humidity, and temperature readings all without taking up too much space. Basically, anything you need to know about atmospheric conditions you can find out from this tiny breakout.



BME280 Chip

This module uses an environmental sensor manufactured by Bosch with temperature, barometric pressure sensor that is the next generation upgrade to the popular

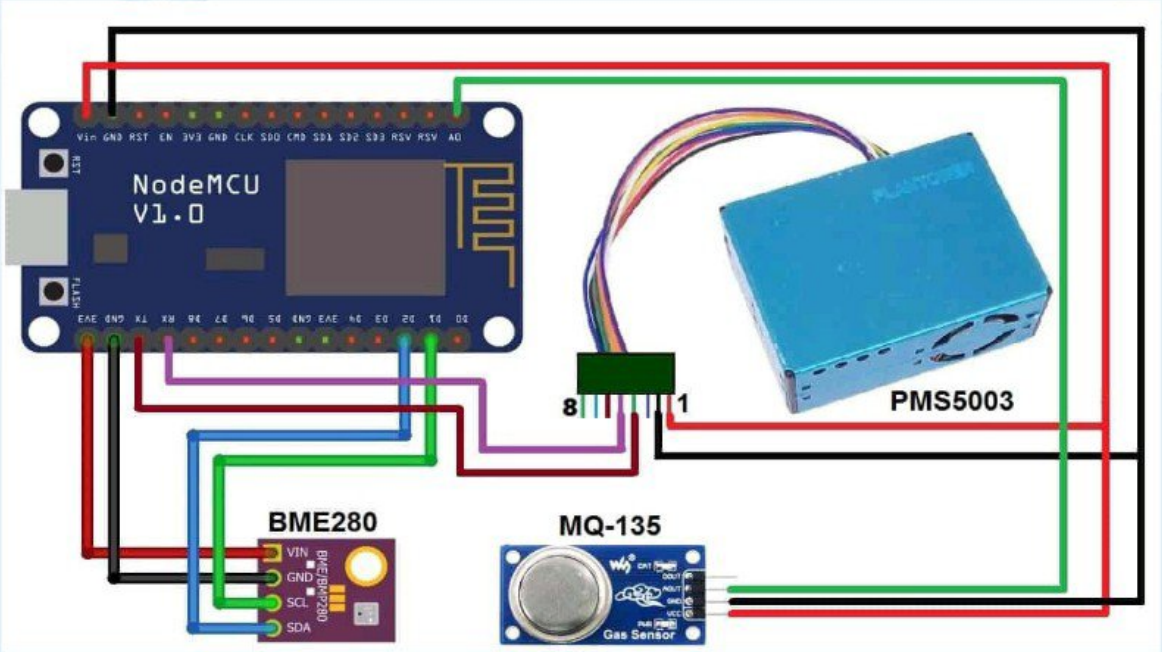
BMP085/BMP180/BMP183 Sensor.

This sensor is great for all sorts of weather sensing and can even be used in both I2C and SPI! This precision sensor from Bosch is the best low-cost, precision sensing solution for measuring barometric pressure with ±1 hPa absolute accuracy, and temperature with ±1.0°C accuracy.

Because pressure changes with altitude and the pressure measurements are so good, you can also use it as an altimeter with ±1 meter accuracy.

IoT Based Air Pollution/Quality Monitoring with ESP8266

Below is the circuit diagram or connection for IoT Based Air Pollution/Quality Monitoring System. There are 3 sensors which is connected to wifi chip NodeMCU ESP8266 12E.



PMS5003 Sensor works on UART Communication. It has 8 pins and counting is done from the right. Connect Pin1 VCC to Vin pin of NodeMCU & Pin2 GND to GND.

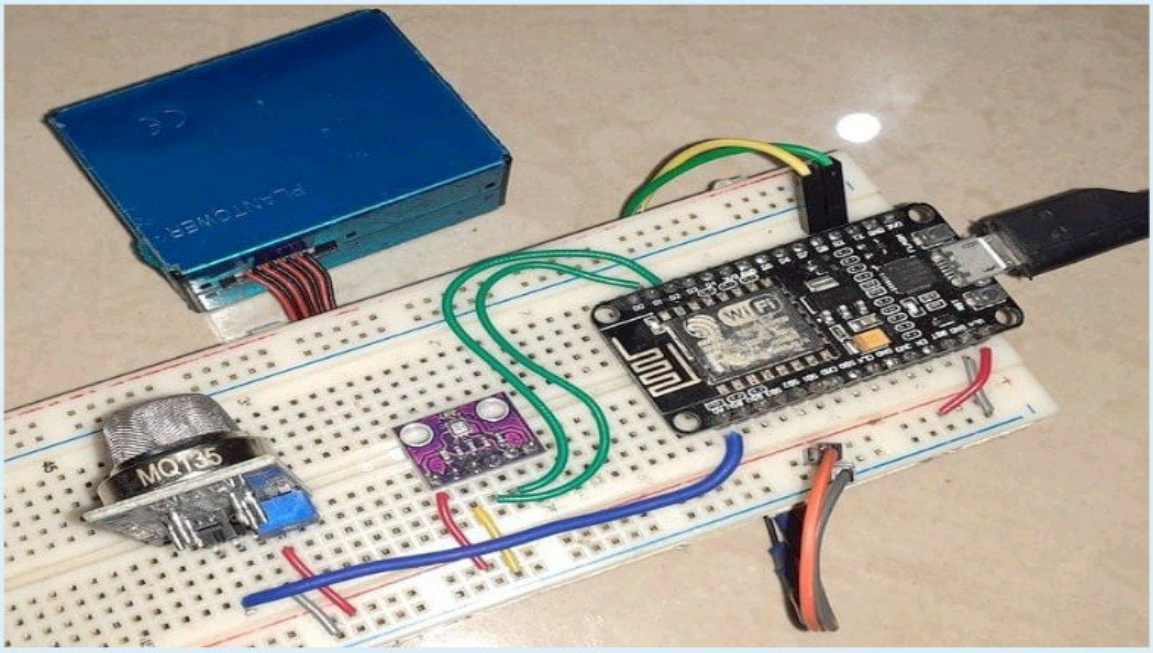
The 4th pin is the Rx pin which is connected to the Tx pin of NodeMCU. Similarly, the 5th pin the Tx pin which is connected to the Rx pin of NodeMCU.

MQ-135 is an analog sensor. So connect its A0 pin to A0 pin of NodeMCU. Connect the VCC pin to Vin 5V & GND to GND.

And then we have BME280 Sensor which works on I2C Communication. So connect it SCL & SDA pin to

I2C pin of NodeMCU, i.e D1, D2.

So here you can see how all the sensors are interfaced with NodeMCU ESP8266-12E.



Source Code/Program IoT based Air Quality Monitoring

The source code/Program is wriiten for NodeMCU ESP8266 12E Board.

CODE

```cpp
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
const char* mqttServer = "your_MQTT_broker_IP";
const int mqttPort = 1883;
const char* topic = "air_quality";

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
```

```
    client.setServer(mqttServer, mqttPort);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }

    int gasValue = readGasSensor(); // Replace with
your actual sensor reading code
    client.publish(topic, String(gasValue).c_str());
    delay(10000); // Adjust as per your requirements
}

void reconnect() {
    while (!client.connected()) {
        Serial.println("Connecting to MQTT...");
        if (client.connect("NodeMCU_Client")) {
            Serial.println("Connected to MQTT");
        } else {
            delay(5000);
        }
```

```
  }
}

int readGasSensor() {
  // Code to read gas data from the sensor
  // Replace this with your actual sensor reading code
}
```

Creating Node-RED Flow

- In Node-RED, create a flow that subscribes to the MQTT topic where the NodeMCU is publishing data. Process the data as needed (e.g., store it in a database, visualize it, etc.).

Data Visualization

- Use Node-RED dashboard nodes or other visualization nodes to display the air quality data.

Output:

- When you run the NodeMCU code, it will

connect to the WiFi network, read gas data from the MQ-135 sensor, and publish it to the MQTT broker. Node-RED will receive this data and process it according to your flow, displaying it in the chosen visualization.

Remember, this is a simplified example. Depending on your specific sensor and requirements, you'll need to adjust the code accordingly.

For a complete project with all the details, including source code and output, I recommend looking for detailed tutorials or guides online, or considering a more in-depth course or resource on IoT projects with Node-RED and the MQ-135 gas sensor.

MOBILE APPLICATION DEVELOPMENT USING MIT APP INVENTOR

Abstract

In present times, it is apparent that weather andclimatic circumstances have grown variable and unpredictable,not just in our country but also in other parts of the world,potentially wreaking havoc on agricultural productivity.

There are many environmental factors like air, precipitation,temperature, humidity, and many other parameters andconditions that keep on changing alarmingly and unpredictably.

It is critical of the hour to have a real-time, local weatherstation that can keep farmers informed about current weather conditions.

As a consequence, they are able to make the bestjudgments at the correct moment and safeguard their crops.

Progress has been made in the Internet of Things (IoT) sectorto track the condition of weather and control air quality, whichhas allowed the number of

harmful substances in air particles.

The goal is to build a device that can monitor the condition ofthe weather in real-time. A dedicated server is available for anandroid application to monitor the weather on the internet.

The system can deploy anywhere. Some external devices can enablewhen pollution crosses the safe zone, like informing all thepeople there via an android application.

INTRODUCTION

Climate changes are so bad for people to live. Everyone issuffering from the curse of climate change from top to bottom.Farmers are discovering that agriculture is no longer a feasibleor sustainable alternative.

Despite their efforts, current agricultural techniques are not producing the anticipated outcomes infavor of farmers. Farmers are under immense pressure to fulfillthe ever-increasing demand for agricultural commodities, yetresources such as machinery and water, as well as various tools such as short-term

irrigation, are in limited supply(high cost). Agriculture has also been damaged by currentclimate circumstances such as moisture levels and humidity,and wind speed has a major influence on crop growth [1]. Ithas brought climate change, such as global warming, globaldim, rainfall, drought, acid rain, foggy weather, etc. Living things on earth and underwater suffers from many problems,such as a change in life due to a lack of adequate living facilities.

To quantify noise present in theenvironment, a noise sensor is also added. These statistics were used for a server and Android program, as almost everyonenow has an android computer and Internet access.

Thesesensors collect data from fields on various gases (e.g., methane(CH4), humidity, nitrogen dioxide (NO2), carbon monoxide(CO) in real-time from different affected locations.

It will alsogather data on the sound level in the area. The proposed systemmainly provides for the monitoring, through an applicationdeveloped via

Graphical User Interface (GUI), of weather pollution and purity or quality conditions of a smart city ona computer (Desktop or laptop) send a signal when pollution. exceeds appropriate standards.

SYSTEM OVERVIEW

This section describes the block diagram, circuit diagram,hardware components and Android application overview.

Block Diagram of the System

Diagram 1 depicts the block diagram of this project. Hardwarecomprises of an Arduino microcontroller. The microcontrollertakes data from sensors such as humidity, temperature and COconcentration. ThingSpeak, an open IoT platform, receives the sensor data and displays it on a LCD.

Circuit Diagram of the System

The ESP8266 from Systems is a low-cost WIFI modulethat includes a complete TCP/IP stack and a

microcontroller[14]. It features 17 GPIO pins and a working voltage of 3to 3.6 volts. On the other hand air quality sensor (MQ 135),temperature and humidity sensor (DHT 22), CO gas sensor(MQ 7) is connected to microcontroller (Arduino Uno) [15]
as input and LCD display is connected to see the output of theproject.

The complete circuit diagram of this model is shownin Diagram 2, This diagram is drawn using Proteus software.

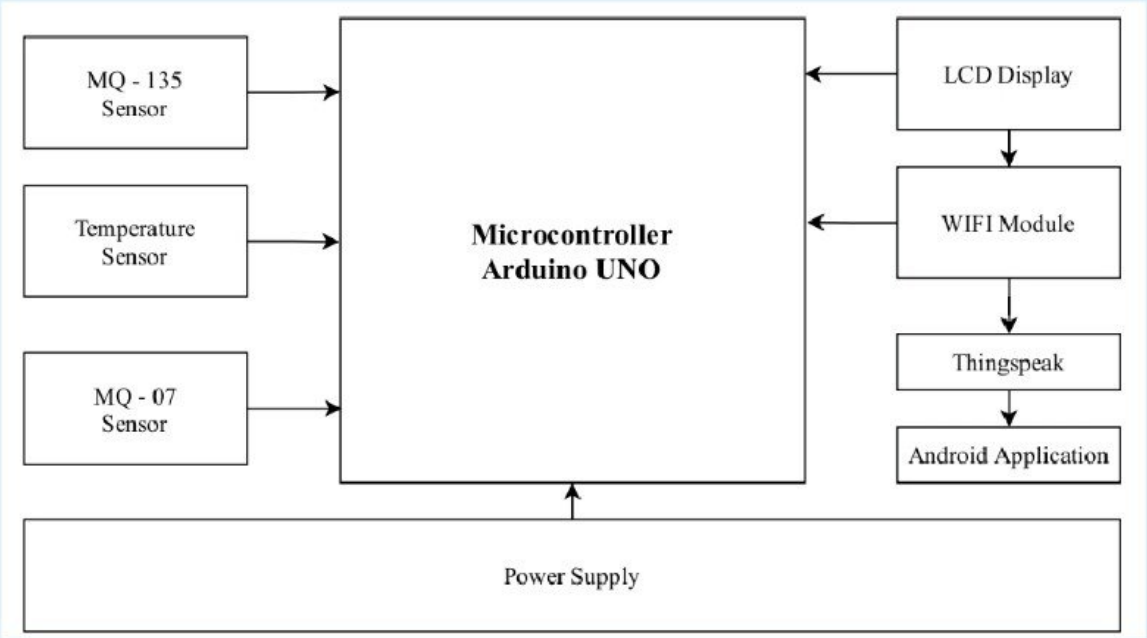
Software Overview

In this proposed model, MIT App Inventor service [16] isused to remotely monitor sensor data and operate the hardware.The MIT App Inventor has an easy-to-use interface.

It mayshow sensor data in the form of widgets. The main componentsof MIT App Inventor are the application building service,libraries and servers. A user can operate the device through theInternet using these components.

It is also cable of connectingby Ethernet, WiFi,

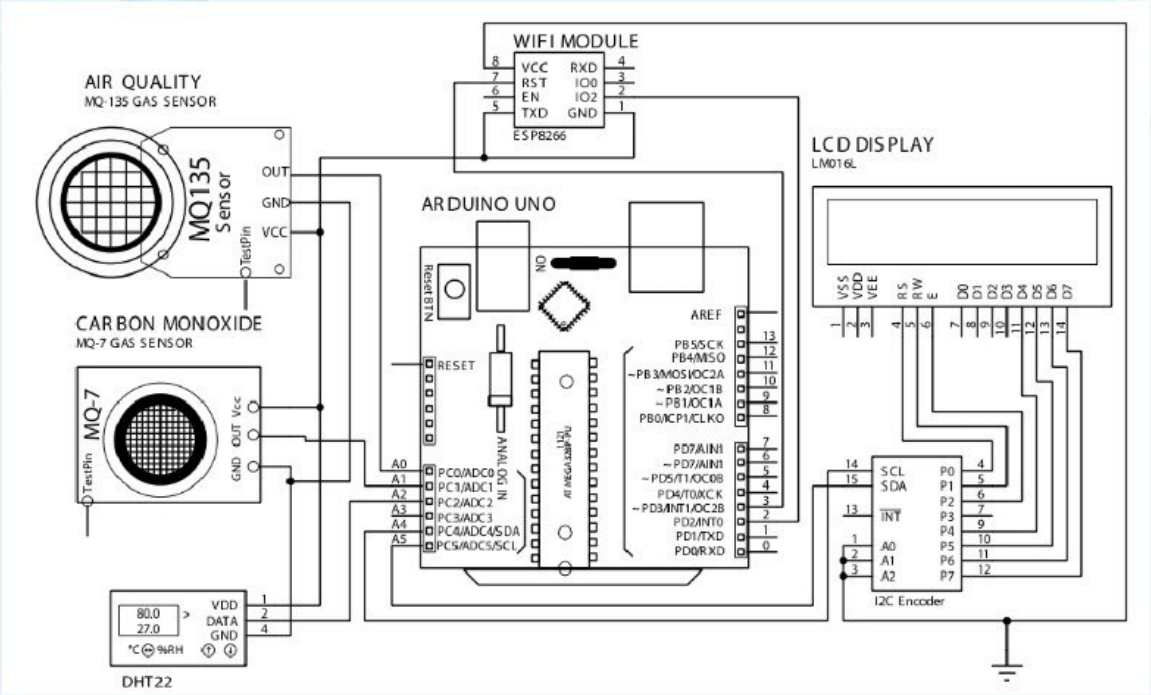
Bluetooth, and serial USB. Widgets like asButton, Slider, Timer, and Joystick are accessible in this app creation service to monitor and manage the system's hardware.



Overview of the whole system

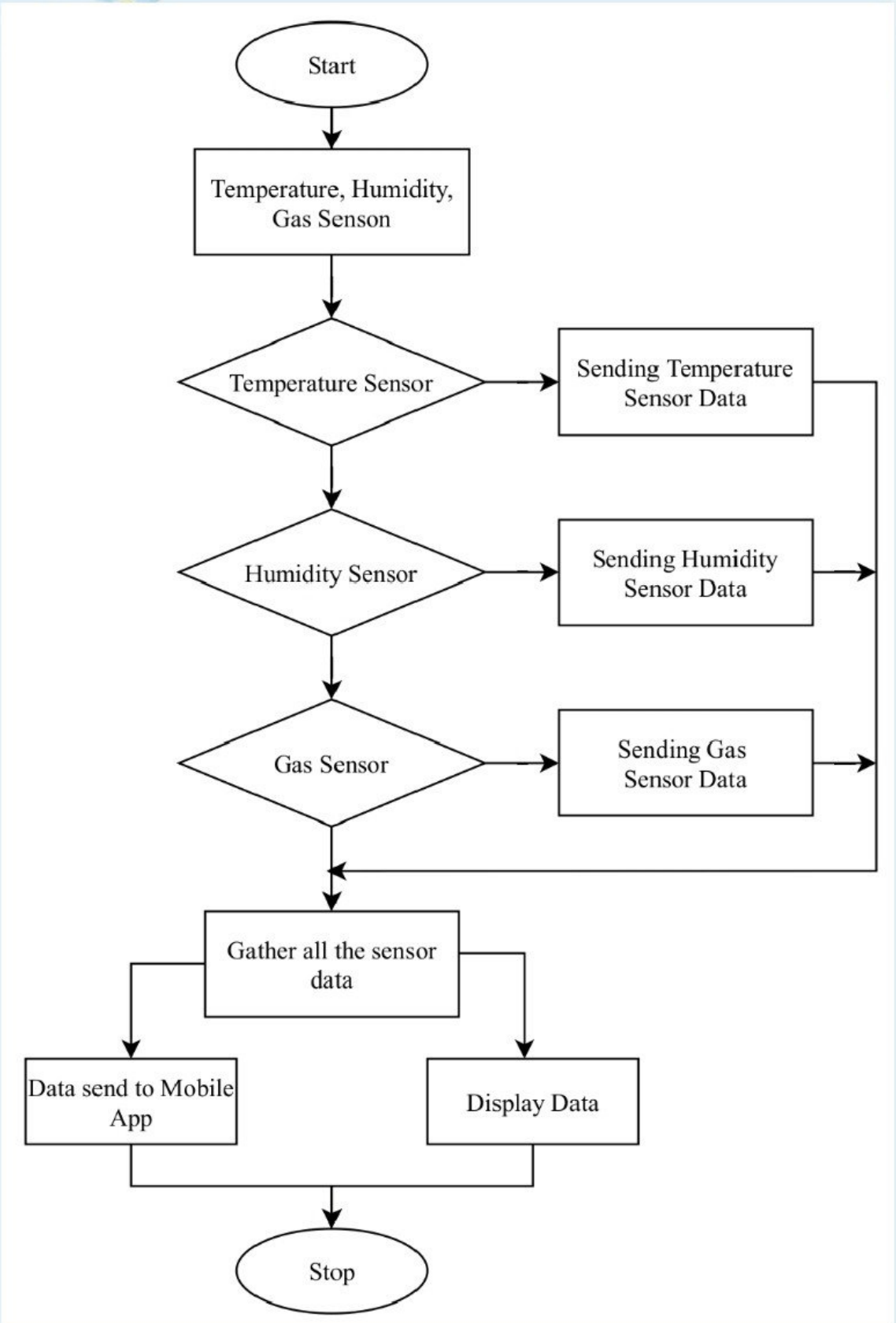
SYSTEM DESIGN

CIRCUIT DIAGRAM



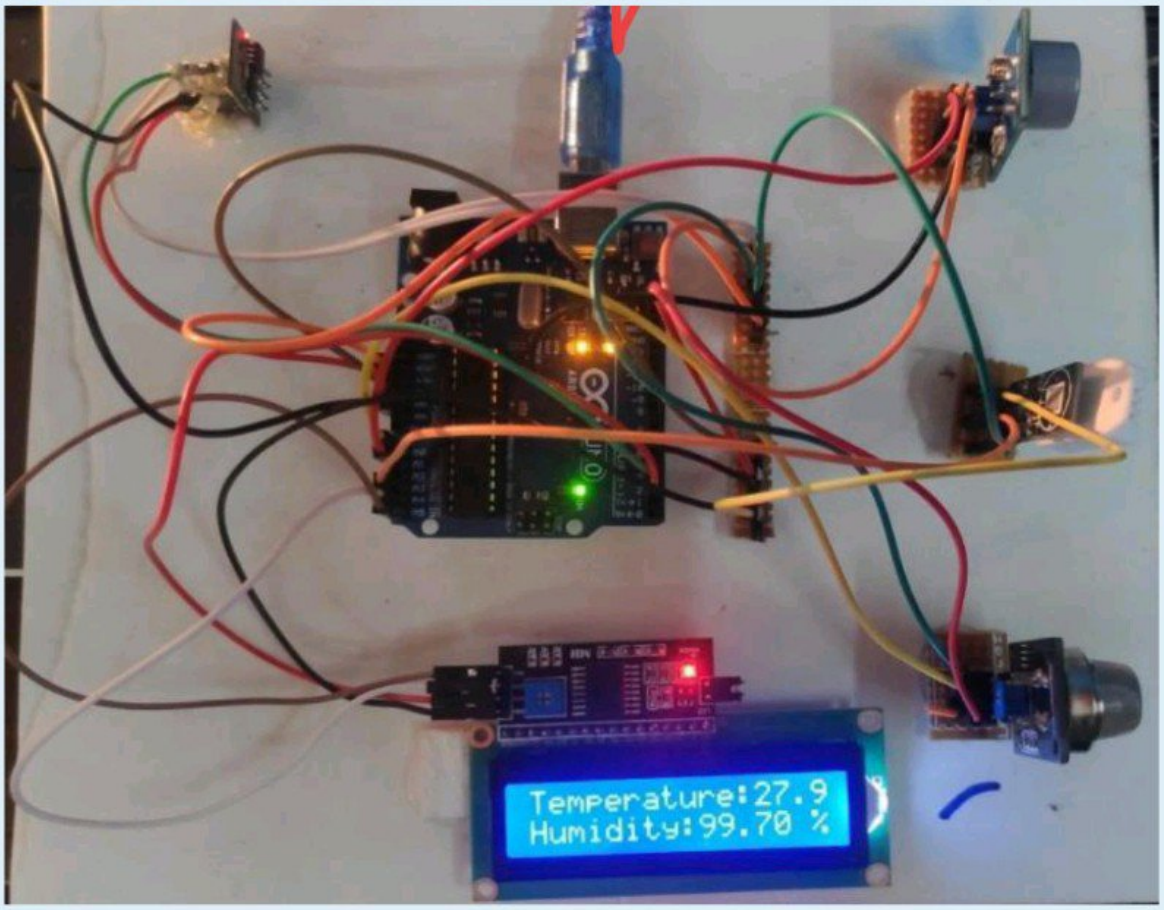
Circuit diagram for proposal model of the system

The flow chart of this proposed model is shown in Diagram 3 which was created using the Flowchart Maker



Flowchart of the whole system

The Diagram 4 shows the implemented proposed model in real life.



Proposed model in real life

Setting up the IoT Device

For this example, we'll assume you're using an ESP8266 board and a PM2.5 sensor for air quality monitoring.

You'll need the Arduino IDE for programming the ESP8266.Connect the PM2.5 sensor to the ESP8266 board.Write the Arduino code to read data

from the sensor and send it to a server (e.g., ThingSpeak).

Here's a simplified Arduino code snippet:

CODE

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
const String apiKey = "YOUR_API_KEY";
const String server = "api.thingspeak.com";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
}
```

```
void loop() {
  // Read sensor data (PM2.5, CO2, etc.)
  int pm25Value = getPM25Value(); // Function to get PM2.5 value

  // Send data to ThingSpeak
  HTTPClient http;
  http.begin("http://" + server + "/update?api_key=" + apiKey + "&field1=" + String(pm25Value));
  int httpCode = http.GET();
  http.end();

  delay(30000); // Send data every 30 seconds (adjust as needed)
}
```

Setting up ThingSpeak

Create an account on ThingSpeak (if you haven't already).Create a new channel and set up a field to store PM2.5 data.

Creating the MIT App Inventor Project

Open MIT App Inventor and start a new project.Design the user interface with components like labels, buttons, and a Web component.Drag and drop a Web component from the palette.

Adding Blocks in MIT App Inventor

Use blocks to handle the communication between your app and the ThingSpeak server.

In this example, we're using a Web1 component to make a GET request to ThingSpeak. When the screen initializes, it fetches the PM2.5 value from ThingSpeak and updates a label with that value.

Testing and Deployment

Connect your phone to MIT App Inventor for testing. You can do this via the MIT AI2 Companion app.Once you're satisfied with the app's functionality, you can package it and deploy it on your Android device.Please note that this example is simplified and you may need to adapt it to your

specific sensor, IoT board, and server setup. Additionally, consider adding error handling, notifications, and other features based on your project's requirements.

RESULT ANALYSIS

Diagram 5-7 shows the output of air quality, CO, temperatureand humidity in LCD Display respectively.

Five datasets were chosen and sent to the cloud platform.Every 10 seconds, the server receives one Arduino input. Thedata is stored in five variables and sent to five separate fieldsof the cloud channel. Figure 8 depicts the system's MobileApplication view.

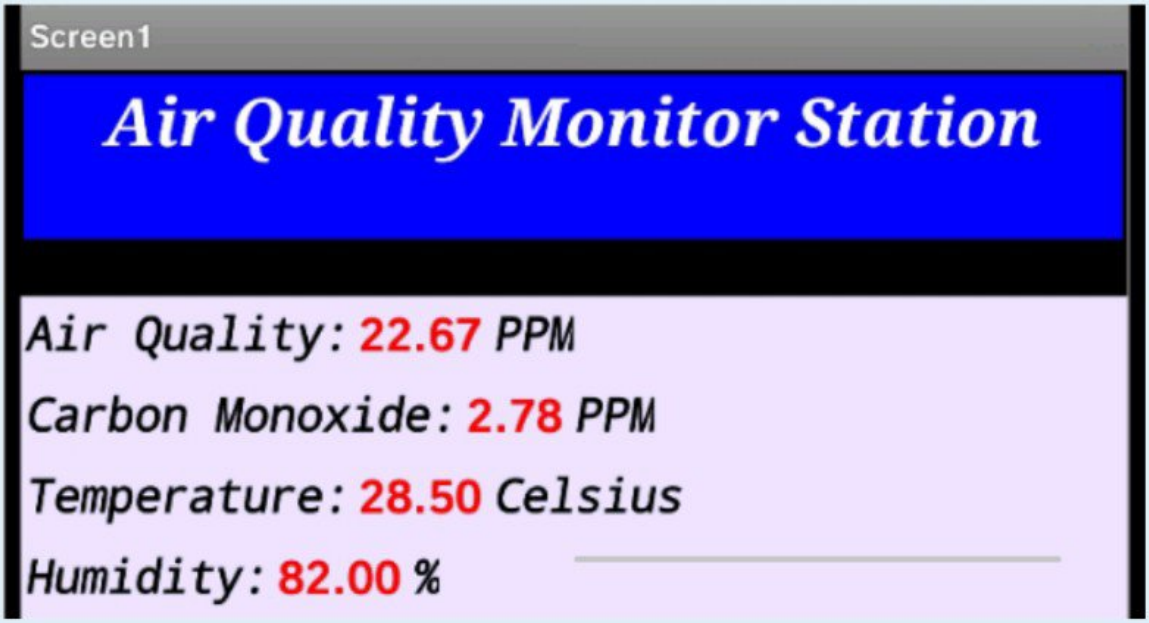
Because the skin relies on the air to expel moisture, humans are extremely sensitive to humidity. If theambient temperature is 24 degrees Celsius and the relative

LCD display view (air quality)

LCD display view (CO)

LCD display view (Temperature & Humidity)

humidity is 100%. It feels like 27°C outside. The humidityrate we receive at room temperature of 28.4°C is 90%, whichis somewhat higher than the dew point. CO levels in the airshould be between 0 and 5 parts per million. 2.78 ppm wasdiscovered.



Android application view

Dust sensors or warning systems to warn of danger are not employed in this project concept, according to [17].

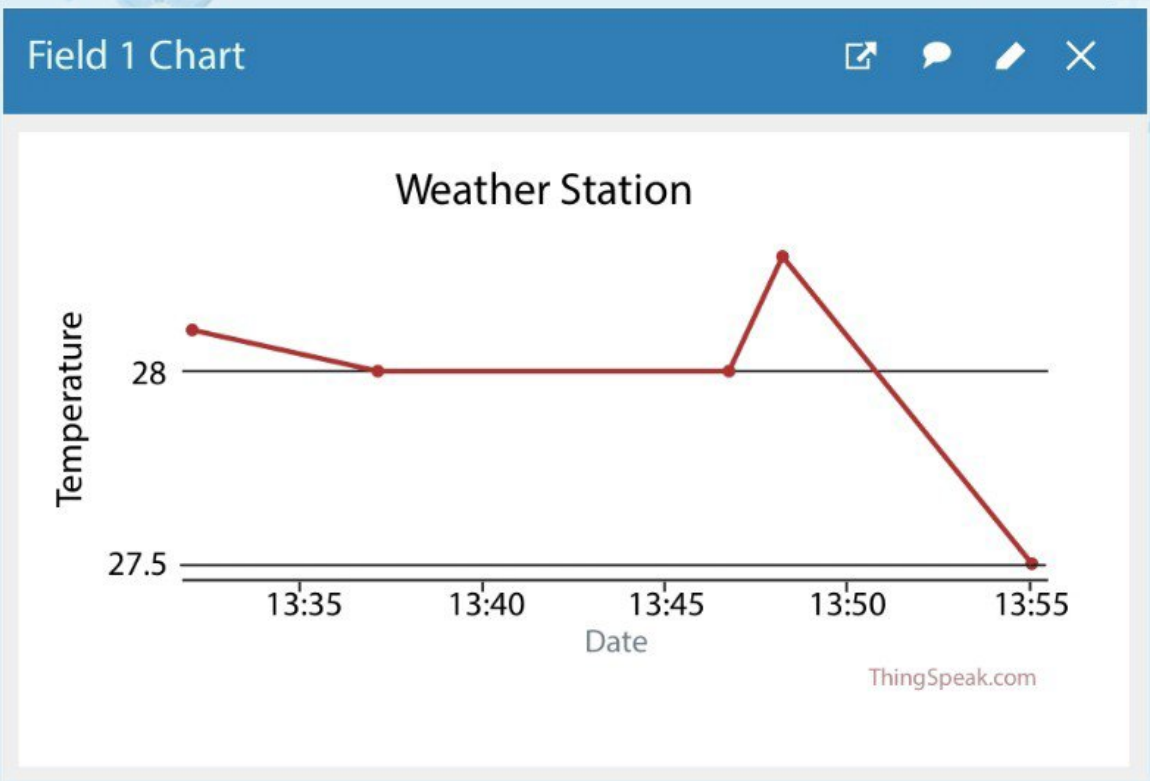
Itwill be done at some point in the future. This type is notdesigned for larger homes, such as workplaces or schools. Insuch circumstance, a wireless sensor network was required.

However, due to a shortage of funds, such gadgets cannotbe built. There is a goal to combine radiation detection with machine learning methods in the future to improve air qualitydetection.

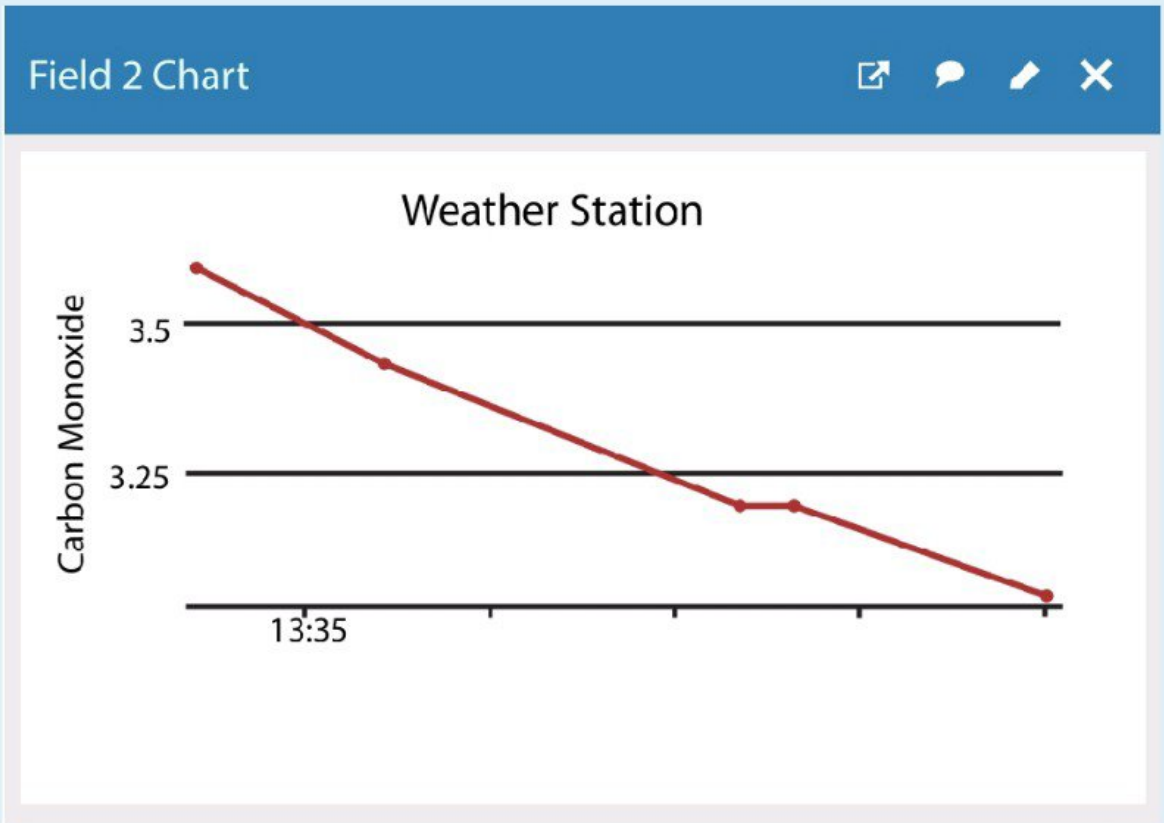
ThingSpeak Visualizations

The most important tabs on the ThingSpeak dashboard arethe channel id and API credentials (read and write). You havethe option of making the channel private or public (visible to others).

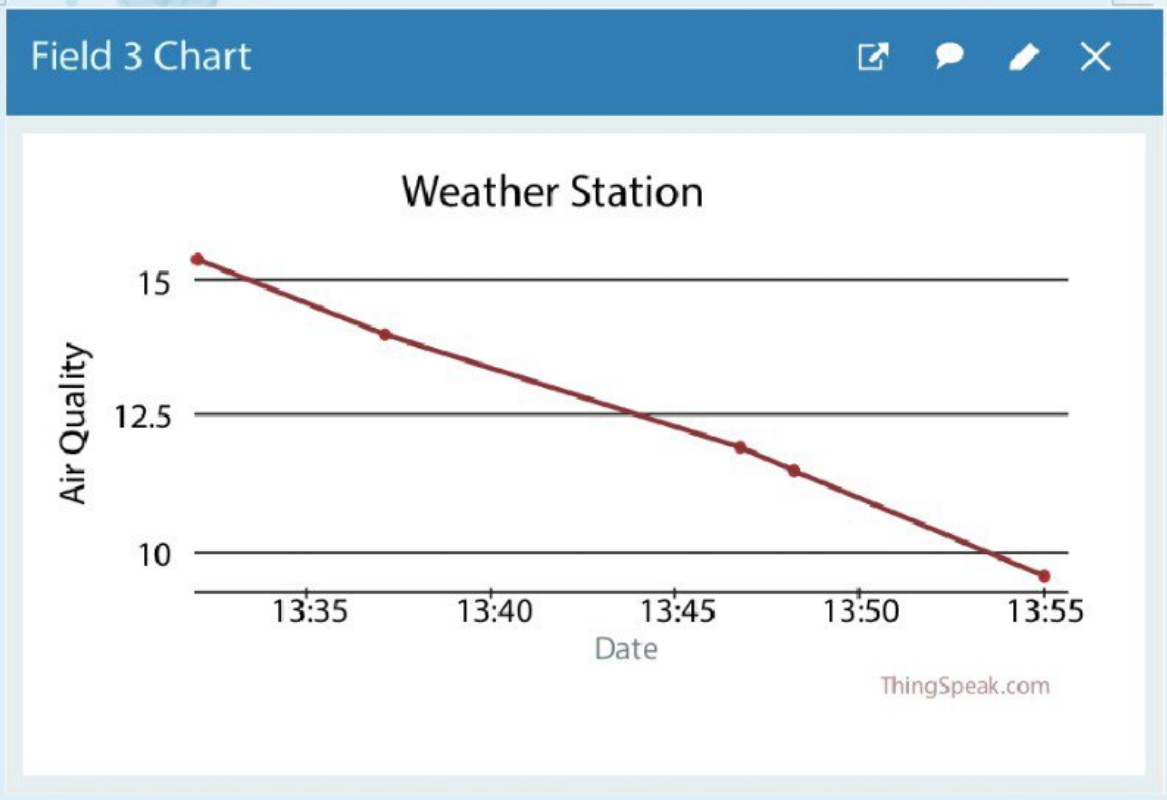
In this prototype, the channel is presently set toprivate. It also contains the MATLAB Analysis and Visualization tab, as well as the other visualization tabs .When thesuggested program is implemented, the air quality, temperature and humidity data are posted to the website in every 10seconds from the Ardunio, and a graphical representation of the detected data is displayed to the Thingspeak which is shown in diagram 9-12



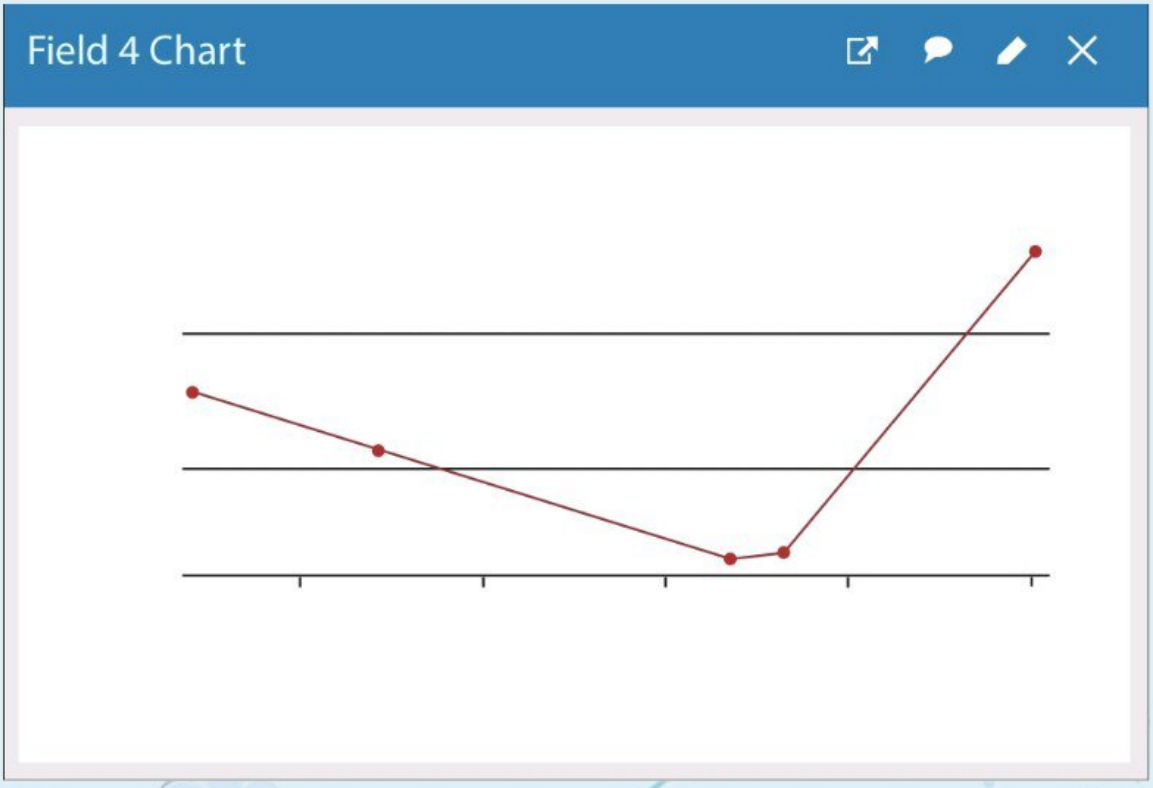
Real-time variation in temperature (Field 1 chart)



Real-time variation time in CO (Field 2 chart)



Real-time variation in air quality (Field 3 chart)



Real -time variation in Humidity (Field 4 chart)

CONCLUSION

This per finds an intelligent way of monitoring the atmosphere and weather in a low-cost but effective and integratedsystem.

Different sensor functions and working procedureswere discussed in the proposed architecture.

Here we alsoaddressed the manner in which they can operate, their features, their best use and processes for data collection, and comparisons to standard basic data.

The built prototype's low cost(open-source hardware/software, low-cost sensors) characteristics will assist farmers in carrying out agricultural tasks at the proper and advantageous moment.

This is a modest steptoward incorporating technology into the agricultural sector, making agricultural techniques more profitable and sustainable for farmers.

The sensor parameters were also forwarded to the data server. Our project system has shown that it is powerful, cheap, and can be reliable to anyone with a few highly functioning sensors.

It's data is crucial to making some of the steps possible for a better society.

A few additional sensors will be added in the future to sense wind speed and direction, and the prototype will be deployed onto an agricultural field to monitor weather parameters, transforming it into a full fledged Precision Weather Station that may act as an important element of the system.

As a result, we'll be able to see the problem early on and take steps to protect the next generation.

THANK YOU