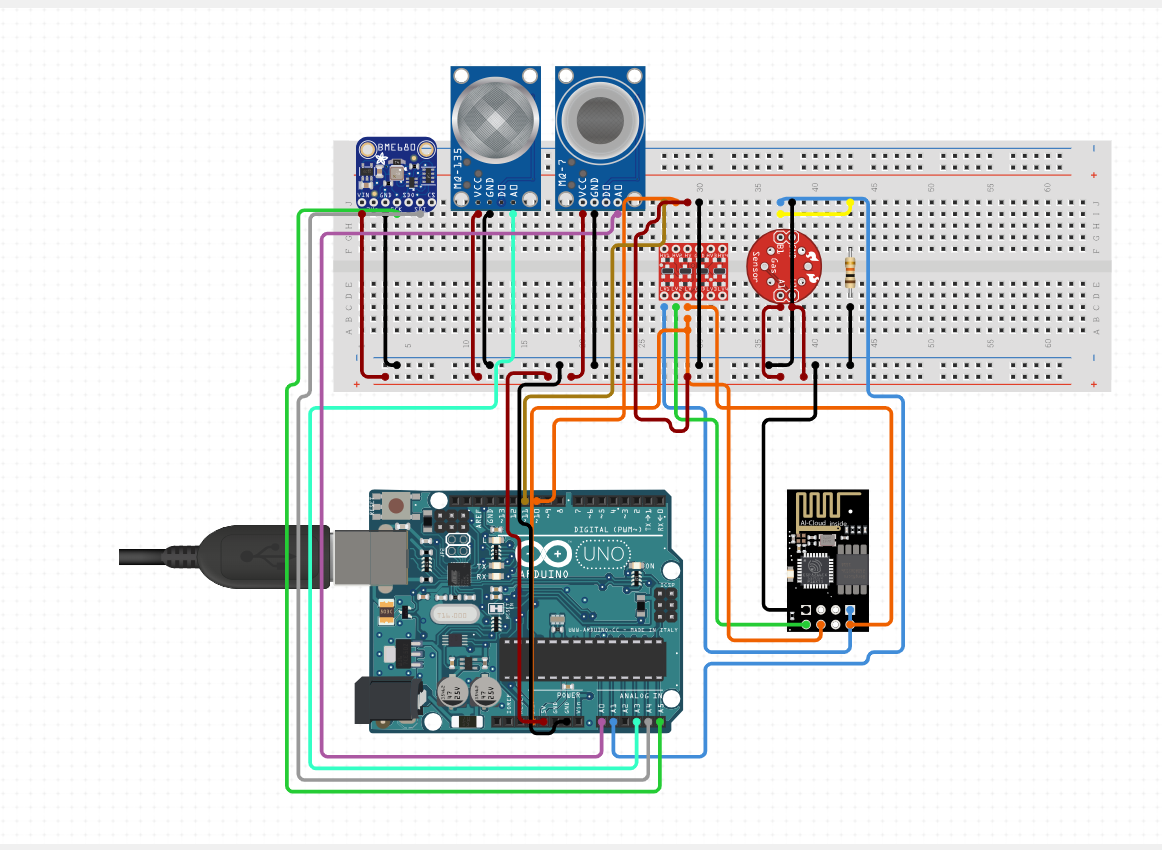
**Phase 5 (Documentation)**

**Air Quality Monitoring (with Sensi-Air Quality Monitor)**

**Objective:**

The primary aim and objective of the project is to constantly monitor the air quality of the developing and developed cities and regularly updating the people of such cities about the quality of the air they inhale, The project also has an objective of triggering the people to realize the importance of the preserving the quality of air, Technically this particular “Sensi- air-quality monitor” performs the following function. It monitors the presence of the following components in the air around it such as particulate matter (PM 10 and PM 2.5), carbon monoxide (CO), ozone (O3), nitrogen dioxide (NO2), sulphur dioxide (SO2), ammonia (NH3), and lead (Pb) and calculate the air quality index (AQI) the calculated AQI is shared to the user’s mobile phone via SMS.

**IOT device Setup:**



The above picture describes the interior view of the “Sensi-Air-Quality Monitor“ , the IoT device consists of the following components

* Arduino UNO (instead of ESP-32 board)
* MQ135 Sensor
* MQ7 Sensor
* MQ 9 Sensor
* BME 680 sensor
* WI-FI
* Resistor of 1 KΩ
* Bread Board

**Arduino code: (in python)**

import BME680

import MQ135

import MQ9

import MQ7

import time

import requests

# Define the BME680 sensor

bme680 = BME680.BME680()

# Define the MQ135, MQ9, and MQ7 sensors

mq135 = MQ135()

mq9 = MQ9()

mq7 = MQ7()

# Define the WiFi module

wifi = WiFi()

# Connect to the WiFi network

wifi.connect("YOUR\_WIFI\_SSID", "YOUR\_WIFI\_PASSWORD")

# Define the URL of the software where the data should be shared

data\_sharing\_url = "https://YOUR\_SOFTWARE\_URL/api/data"

# Define a function to collect the data from the sensors

def collect\_data():

# Get the temperature, humidity, pressure, air quality, and TVOC readings from the BME680 sensor

temperature = bme680.temperature

humidity = bme680.humidity

pressure = bme680.pressure

air\_quality = bme680.air\_quality

tvoc = bme680.tvoc

# Get the MQ135, MQ9, and MQ7 readings

mq135\_reading = mq135.read()

mq9\_reading = mq9.read()

mq7\_reading = mq7.read()

# Return the collected data

return temperature, humidity, pressure, air\_quality, tvoc, mq135\_reading, mq9\_reading, mq7\_reading

# Define a function to share the data with the software

def share\_data(data):

# POST the data to the data sharing URL

requests.post(data\_sharing\_url, json=data)

# Start a loop to continuously collect and share the data

while True:

# Collect the data from the sensors

data = collect\_data()

# Share the data with the software

share\_data(data)

# Wait for 1 second

time.sleep(1)

**Explanation of the code:**

* Import the required libraries.
* Define the sensors.
* Define the Wi-Fi module.
* Connect to the Wi-Fi network.
* Define the URL of the software where the data should be shared.
* Define a function to collect the data from the sensors.
* Define a function to share the data with the software.
* Start a loop to continuously collect and share the data.

**Working Of IoT device in Detail:**

Functioning of the sensors can be explained from the table as follows:

|  |  |
| --- | --- |
| Sensor Type | Equivalent Sensor |
| Electrostatic sensors for particulate matter | MQ135, MQ7, MQ9 |
| Electrochemical CO Sensors for Carbon monoxide | BME 680 |
| Ozone Sensors | MQ135 |
| Gas Sensitive Semiconductor Sensors for Nitrogen di Oxide and Sulphur di Oxide | MQ135, MQ7 |
| Metal Oxide Sensors for ammonia | MQ135 |

* Electrostatic sensors for particulate matter: They use an electrostatic field to attract particles. The number and size of particles collected on the sensor's surface are used to estimate PM2.5 and PM10 concentrations.
* Electrochemical CO Sensors for Carbon monoxide: These sensors use a chemical reaction that occurs at an electrode when CO is present. The resulting current is proportional to the CO concentration.
* Ozone Sensors: Ozone sensors typically use electrochemical or ultraviolet (UV) absorption principles to measure ozone levels.
* Gas Sensitive Semiconductor Sensors for Nitrogen di Oxide and Sulphur di Oxide: These sensors rely on the change in electrical conductivity of a semiconductor material when exposed to NO2 andSO2.
* Metal Oxide Sensors for ammonia: Some metal oxide sensors can detect ammonia by measuring changes in electrical resistance.

**Block Diagram for working of the AQI monitor:**



Data Processing

Data Collection

Data Storage

User Interface and data processing

Data shared via sms

comparison of previous data

**Data Collection:**

**Data collection** is done by the sensors such as MQ135, MQ7, MQ9, BME680,

**Data Transmission:**

The WI-FI module in the product helps to share the collected data to a software platform.

**User Interface and data processing:**

Python code is used to build a software which collects the data from the sensor, process the data such that the obtained data is used to calculate the air quality index.

**Sharing of data via SMS:**

The software is designed in the manner that the air quality index calculated is shared to the users via a mobile SMS.

**Data Storage:**

The air quality index is calculated for every 10 seconds and the data is being in stored in the same software

**Data Comparison:**

The air quality index of every day is compared with that of the previous day and the data and result is published in the interface.

**Python Code for the software:**

import pandas as pd

import time

from serial import Serial

import requests

# Define the pollutant\_data\_readings dictionary

pollutant\_data\_readings = {}

# Define the pollutant\_standards dictionary

pollutant\_standards = {

"PM2.5": 25,

"PM10": 50,

"Carbon monoxide": 9.0,

"Nitrogen dioxide": 0.08,

"Ozone": 0.065,

"sulphur dioxide": 0.08,

"ammonia": 9.0,

"lead": 0.08

}

# Define a list to store the mobile numbers

mobile\_numbers = []

# Define a list to store the previous AQI data

previous\_aqi\_data = []

# Define a function to send a text message to a mobile number

def send\_sms(mobile\_number, message):

# Use a third-party SMS API to send the text message

requests.post("https://api.twilio.com/2010-04-01/Accounts/<YOUR\_TWILIO\_ACCOUNT\_SID>/Messages.json",

auth=("YOUR\_TWILIO\_ACCOUNT\_SID", "YOUR\_TWILIO\_AUTH\_TOKEN"),

data={"To": mobile\_number, "From": "+15555555555", "Body": message})

# Define a function to update the AQI values and share them with the mobile numbers

def update\_and\_share\_aqi():

# Update the AQI values

update\_aqi()

# Store the previous AQI data

previous\_aqi\_data.append(aqi\_df.loc["AQI"])

# Classify the AQI

aqi = aqi\_df.loc["AQI"]

aqi\_classification = ""

if aqi <= 50:

aqi\_classification = "Good"

elif aqi <= 100:

aqi\_classification = "Satisfactory"

elif aqi <= 200:

aqi\_classification = "Moderate"

elif aqi <= 300:

aqi\_classification = "Poor"

elif aqi <= 400:

aqi\_classification = "Very poor"

else:

aqi\_classification = "Severe"

# Share the AQI values with the mobile numbers

for mobile\_number in mobile\_numbers:

message = f"AQI for {mobile\_number}: {aqi} ({aqi\_classification})"

send\_sms(mobile\_number, message)

# Define a function to compare the AQI data

def compare\_aqi\_data():

# Get the current AQI

current\_aqi = aqi\_df.loc["AQI"]

# Get the previous AQI

previous\_aqi = previous\_aqi\_data[-1]

# Calculate the difference between the current AQI and the previous AQI

aqi\_difference = current\_aqi - previous\_aqi

# Compare the AQI data

if aqi\_difference > 0:

message = "AQI has increased."

elif aqi\_difference < 0:

message = "AQI has decreased."

else:

message = "AQI has remained the same."

# Print the comparison message

print(message)

# Define a function to add a mobile number to the list

def add\_mobile\_number():

mobile\_number = input("Enter mobile number: ")

mobile\_numbers.append(mobile\_number)

# Define a function to remove a mobile number from the list

def remove\_mobile\_number():

mobile\_number = input("Enter mobile number to remove: ")

mobile\_numbers.remove

**Steps to connect the data collected by the sensors to this software:**

1. Install the necessary Python libraries. You will need to install the following Python libraries:
   * serial: To communicate with the Arduino Uno microcontroller over the serial port.
   * requests: To send HTTP requests to the Python user interface software.
   * pandas: To manipulate and analyze the sensor data in Python.
2. Start the Python user interface software. This will start a server that will listen for incoming HTTP requests.
3. Modify the Arduino code to send the sensor data to the Python user interface software. Add the following code to the end of the Arduino setup() function:

Serial.begin(9600); // Set the serial baud rate to 9600 bps.

Then, add the following code to the Arduino loop() function:

// Collect the sensor data.

float temperature = bme680.temperature;

float humidity = bme680.humidity;

float pressure = bme680.pressure;

float air\_quality = bme680.air\_quality;

float tvoc = bme680.tvoc;

float mq135\_reading = mq135.read();

float mq9\_reading = mq9.read();

float mq7\_reading = mq7.read();

// Create a JSON object containing the sensor data.

DynamicJsonDocument jsonDoc;

jsonDoc["temperature"] = temperature;

jsonDoc["humidity"] = humidity;

jsonDoc["pressure"] = pressure;

jsonDoc["air\_quality"] = air\_quality;

jsonDoc["tvoc"] = tvoc;

jsonDoc["mq135\_reading"] = mq135\_reading;

jsonDoc["mq9\_reading"] = mq9\_reading;

jsonDoc["mq7\_reading"] = mq7\_reading;

// Serialize the JSON object to a string.

char jsonBuffer[512];

serializeJson(jsonDoc, jsonBuffer, sizeof(jsonBuffer));

// Send the JSON string to the Python user interface software.

Serial.println(jsonBuffer);

1. Modify the Python user interface software to receive and process the sensor data. Add the following code to the Python user interface software:

Python

import serial

import json

# Open the serial port to communicate with the Arduino Uno microcontroller.

ser = serial.Serial("/dev/ttyUSB0", 9600)

# Start a loop to continuously receive and process the sensor data.

while True:

# Read a line of data from the serial port.

data = ser.readline().decode("utf-8")

# Deserialize the JSON string into a Python dictionary.

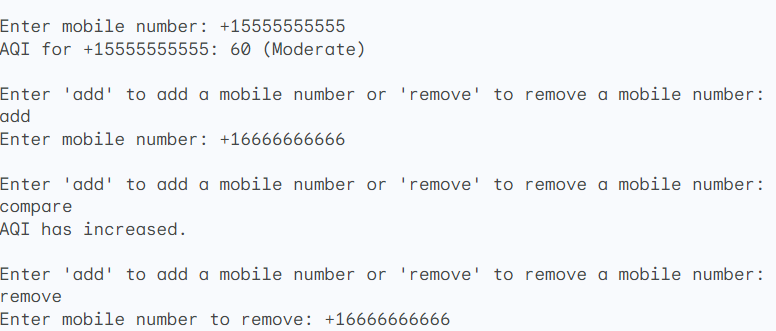
sensor\_data = json.loads(data)

# Process the sensor data.

# ...

You can then use the sensor data in the Python user interface software to display it to the user, store it in a database, or send it to a cloud.

**Sample output:**

****

**Social Impact of This Air Quality Monitor:**

Imagine a world where everyone knows how clean or polluted the air is around them, in real time. Where people can make informed decisions about their health and well-being based on accurate air quality data. This is the world that real-time air quality monitoring systems can help to create.

**Air pollution is a silent killer:**

It can cause a range of health problems, including respiratory infections, heart disease, and cancer. But many people are unaware of the risks, or they don't have access to real-time information about the air quality in their area.

**Real-time air quality monitoring systems can help to raise public awareness about air pollution and its health impacts in several ways:**

* They provide real-time data on the levels of air pollutants in the air. This information can be disseminated to the public through a variety of channels, such as websites, apps, and social media.
* They can be used to create air quality maps and forecasts. This can help people to understand the air quality in their area and to plan their activities accordingly.
* They can be used to develop public health alerts and advisories. This can help people to protect their health from the harmful effects of air pollution.

For example, a real-time air quality monitoring system could be used to send an alert to people in an area where the air quality is poor, advising them to stay indoors or to wear a mask when they go outside. This could help to prevent people from breathing in dangerous pollutants.

Real-time air quality monitoring systems can also be used to raise awareness about the sources of air pollution and the steps that can be taken to reduce it. For example, a system could be used to track the emissions from factories and power plants. This information could then be used to develop policies and regulations to reduce air pollution from these sources.

Here is an interesting analogy to illustrate how real-time air quality monitoring systems can raise public awareness:

Imagine that you are walking down the street on a hot day. You are thirsty, but you don't know where the nearest water fountain is. You see a sign that says "Water fountain 100 feet ahead." This sign helps you to find the water fountain and to quench your thirst.

In a similar way, real-time air quality monitoring systems can help people to find clean air. They provide people with the information they need to make informed decisions about their health and well-being.

**Conclusion:**

Real-time air quality monitoring systems are an essential tool for raising public awareness about air pollution and its health impacts. By empowering people with information, these systems can help to create a healthier and more sustainable world.