**Air quality Monitoring**

**Design of user interface with python:**

*import pandas as pd*

*import time*

*# 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 function to calculate the AQI*

*def calculate\_aqi(pollutant\_data\_readings, pollutant\_standards):*

*aqi\_values = {}*

*for pollutant in pollutant\_data\_readings:*

*aqi\_values[pollutant] = (pollutant\_data\_readings[pollutant] / pollutant\_standards[pollutant]) \* 100*

*return aqi\_values*

*# Get the latest data from the sensor*

*def get\_sensor\_readings():*

*# This function will return a dictionary of pollutant data readings from the sensor*

*return {*

*"PM2.5": 20,*

*"PM10": 40,*

*"Carbon monoxide": 8,*

*"Nitrogen dioxide": 0.07,*

*"Ozone": 0.06,*

*"sulphur dioxide": 0.07,*

*"ammonia": 8,*

*"lead": 0.07*

*}*

*# Calculate the AQI*

*def update\_aqi():*

*pollutant\_data\_readings = get\_sensor\_readings()*

*aqi\_values = calculate\_aqi(pollutant\_data\_readings, pollutant\_standards)*

*return aqi\_values*

*# Store the AQI values in a DataFrame*

*aqi\_df = pd.DataFrame(update\_aqi(), index=["AQI"])*

*# Display the AQI DataFrame*

*print(aqi\_df)*

*# Get the user's input*

*user\_input = input("Press 'p' to display the stored AQI values, or any other key to continue: ")*

*# Display the stored AQI values if the user enters 'p'*

*def display\_stored\_aqi\_values():*

*print("Stored AQI values:")*

*for pollutant in aqi\_df.index:*

*print(f"{pollutant}: {aqi\_df.loc[pollutant, 'AQI']}")*

*if user\_input == "p":*

*display\_stored\_aqi\_values()*

*# Wait for 1 second*

*time.sleep(1)*

*# Update the AQI values every second*

*while True:*

*aqi\_values = update\_aqi()*

*aqi\_df.update(aqi\_values)*

*time.sleep(1)*

**Circuit For Air Quality Monitoring**

* Electrostatic sensors for particulate matter
* Electrochemical CO Sensors for Carbon monoxide
* Ozone sensors
* Gas sensitive Semiconductor Sensors for nitrogen di oxide and Sulphur di oxide
* Metal oxide sensors for ammonia
* Atomic absorption sensors for lead
* The sensors are connected to the ESP8266 microcontroller using analog and digital pins. The specific pin connections will vary depending on the specific sensors that you are using. The other components of the circuit are
* ESP8266 microcontroller
* Resistors (1Ω, 2Ω)
* Capacitors (various values)
* Voltage regulators (various voltages)
* Sensors (listed above)
* Jumper wires
* Breadboard

The following is the code to be uploaded to ESP8266

**Code :**

**import time**

**import board**

**import adafruit\_esp8266**

**# Create an ESP8266 object**

**esp8266 = adafruit\_esp8266.ESP8266(board.D4, board.D3)**

**# Connect to a Wi-Fi network**

**esp8266.connect('YOUR\_WIFI\_NETWORK', 'YOUR\_WIFI\_PASSWORD')**

**# Define the analog pins that the sensors are connected to**

**PM25\_SENSOR\_PIN = board.A0**

**CO\_SENSOR\_PIN = board.A1**

**OZONE\_SENSOR\_PIN = board.A2**

**NO2\_SENSOR\_PIN = board.A3**

**SO2\_SENSOR\_PIN = board.A4**

**NH3\_SENSOR\_PIN = board.A5**

**PB\_SENSOR\_PIN = board.A6**

**# Read the sensor values**

**pm25Reading = esp8266.analog\_read(PM25\_SENSOR\_PIN)**

**coReading = esp8266.analog\_read(CO\_SENSOR\_PIN)**

**ozoneReading = esp8266.analog\_read(OZONE\_SENSOR\_PIN)**

**no2Reading = esp8266.analog\_read(NO2\_SENSOR\_PIN)**

**so2Reading = esp8266.analog\_read(SO2\_SENSOR\_PIN)**

**nh3Reading = esp8266.analog\_read(NH3\_SENSOR\_PIN)**

**pbReading = esp8266.analog\_read(PB\_SENSOR\_PIN)**

**# Calculate the AQI index for each sensor**

**pm25Aqi = (pm25Reading / 100) \* 100**

**coAqi = (coReading / 100) \* 100**

**ozoneAqi = (ozoneReading / 100) \* 100**

**no2Aqi = (no2Reading / 100) \* 100**

**so2Aqi = (so2Reading / 100) \* 100**

**nh3Aqi = (nh3Reading / 100) \* 100**

**pbAqi = (pbReading / 100) \* 100**

**# Send the AQI values to a server**

**# ...**

**# Delay for 1 second**

**time.sleep(1)**