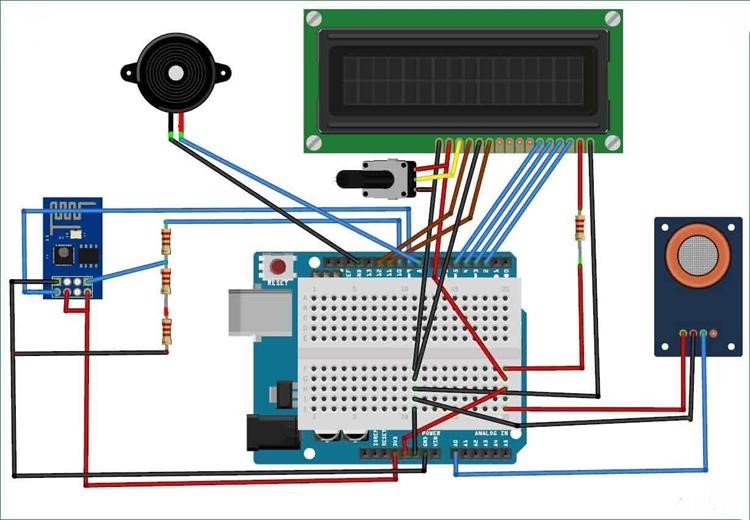
**Air quality monitoring**

The main goal of this project is to create a system that can measure various air quality parameters and display or transmit this information for real-time monitoring. This system is particularly useful for individuals, schools, communities, or environmental enthusiasts who want to keep track of air quality in their immediate surroundings.



# Air Pollution Monitoring System

**Components and Sensors:**

**The project involves the use of several components and sensors:**

* Arduino Board or raspberry pi
* Air Quality Sensor (MQ135 Gas sensor)
* Display (16X2 LCD)
* Wi-Fi Module (ESP8266)
* Power Supply
* Breadboard and Jumper Wires
* Buzzer (optional)
* 10K potentiometer
* 220 ohm resistor
* 1K ohm resistors

An Air Quality Monitoring (AQM) is a system that measures metrological parameters such as wind speed, wind direction, rainfall, radiation, temperature, barometric pressure and ambient parameters. The AQMS also integrates a series of ambient analyzers to monitor the concentration of air pollutants (such as SO2, NOx, CO, O3, THC, PM, etc.), continuously. HORIBA also provides mobile monitoring stations that can be used to monitor ambient conditions at multiple sites.

HORIBA has more than 50 years experience providing ambient monitoring solutions, recognized around the world. HORIBA has supplied over 15,000 units with the major share in many regions. The monitoring station is tailor-made according to the customer's request. HORIBA can provide several types of stations, calibration equipment and more to meet your challenging monitoring requirements.

The measured data can be remotely monitored and exported in various formats to the local central authorities. The data can be published via the Internet for easy public access to raise awareness on current air pollution levels. This way, the public can prevent outdoor activities and reduce health impacts during heavy polluted days.

**code for Arduino** **:**

#include

<SPI.h>

#include

<Wire.h>

#include

<Adafruit\_

GFX.h>

#include

<Adafruit\_

SSD1306.h

>

#include <Fonts/Fr eeSans9pt

7b.h>

#include

<Fonts/Fr

eeMonoOb lique9pt7

b.h>

#include <DHT.h>

#define

SCREEN\_

WIDTH

128

#define

SCREEN\_H EIGHT 64

#define

OLED\_RES

ET 4

Adafruit\_S SD1306 display(SC

REEN\_WI

DTH,

SCREEN\_H EIGHT, &Wire,

OLED\_RES

ET);

#define sensor A0

#define

DHTPIN 2 // Digital pin 2

#define

DHTTYPE

DHT11

// DHT 11

int gasLevel =

0; //int

variable

for gas

level String quality

="";

DHT dht(DHTP

IN,

DHTTYPE)

;

void sendSenso

r()

{

float h = dht.readH umidity(); float t = dht.readT emperatur

e();

if

(isnan(h)

|| isnan(t))

{

Serial.prin tln("Failed to read from DHT

sensor!"); return;

}

display.set TextColor( WHITE);

display.set TextSize(1

);

display.set Font();

display.set Cursor(0,

43);

display.pri ntln("Tem

p :");

display.set

Cursor(80,

43);

display.pri ntln(t);

display.set Cursor(11

4, 43);

display.pri ntln("C");

display.set Cursor(0,

56);

display.pri ntln("RH

:");

display.set

Cursor(80,

56);

display.pri ntln(h);

display.set Cursor(11

4, 56);

display.pri ntln("%");

}

void air\_sensor

()

{

gasLevel

=

analogRea d(sensor);

if(gasLeve l<181){

quality = " GOOD!";

}

else if (gasLevel

>181 &&

gasLevel< 225){

quality = " Poor!";

}

else if (gasLevel

>225 &&

gasLevel< 300){

quality = "Very bad!";

}

else if (gasLevel

>300 &&

gasLevel< 350){ quality = "ur dead!";

} else{ quality = " Toxic";

}

display.set TextColor( WHITE);

display.set TextSize(1

);

display.set

Cursor(1,5

); display.set Font();

display.pri ntln("Air

Quality:");

display.set TextSize(1

);

display.set Cursor(20,

23);

display.set Font(&Fre eMonoObl ique9pt7b

); display.pri ntln(qualit

y); }

void setup() {

Serial.begi n(9600);

pinMode(s ensor,INP UT);

dht.begin(

);

if(!display. begin(SSD

1306\_SWI TCHCAPV

CC, 0x3c))

{ //

Address

0x3D for

128x64

Serial.prin tln(F("SSD 1306

allocation failed")); }

display.cle arDisplay(

);

display.set TextColor( WHITE); display.set TextSize(2

);

display.set

Cursor(50,

0);

display.pri ntln("Air")

;

display.set TextSize(1

);

display.set

Cursor(23,

20);

display.pri ntln("Qula

ity

monitor");

display.dis play();

delay(120

0);

display.cle arDisplay(

);

display.set TextSize(2

);

display.set

Cursor(20,

20); display.pri ntln("BY Abid");

display.dis play();

delay(100

0);

display.cle arDisplay(

);

}

void loop() { display.cle arDisplay(

); air\_sensor

();

sendSenso r(); display.dis

play();

}

**Using this code, we can monitering the Air quality web page** :

<!DOCTYPE html>

<html>

<head>

<title>AIR QUALITY MONITERING</title>

</head>

<body>

<center>

<h1> Air Quality Monitering </h1>

<label class="switch">

<input type="checkbox">

<span class="slider"></span>

</label>

<h3> OFF/ON </h3>

<style type="text/css">

.switch { position: relative; display: inline-block; width: 60px; height: 34px;

}

/\* Hide the default checkbox \*/

.switch input { display: none;

}

/\* Style for the slider (the switch itself) \*/

.slider { position: absolute; cursor: pointer; top: 0; left: 0; right: 0;

bottom: 0;

background-color: #ccc; -webkit-transition: .4s; transition: .4s; border-radius: 34px;

}

/\* Style for the slider when it's in the "on" state \*/ input:checked + .slider { background-color: #2196F3;

}

/\* Rounded sliders \*/ .slider:after { content: ""; position: absolute; height: 26px; width: 26px; left: 4px; bottom: 4px; background-color: white; -webkit-transition: .4s; transition: .4s; border-radius: 50%;

}

/\* Style for the slider when it's in the "off" state \*/ input:checked + .slider:after {

-webkit-transform: translateX(26px); -ms-transform: translateX(26px); transform: translateX(26px);

}

</style>

<div class="display-window">

</div>

<style type="text/css">

/\* Style for the display window \*/

.display-window { width: 40%; height: 50%; margin: 20px auto; border: 2px solid #333; background-color: #f0f0f0;

overflow: auto; /\* Add scrollbars if content overflows \*/ padding: 10px; text-align: center;

}

</style>

</center>

</body>

</html>

This is the model interface for air quality monitering ,



Reference web page for the Air quality monitering , It is the addition of temperature and air moister readings also displayed.

**Python code:**

from tkinter import \* import requests

from bs4 import BeautifulSoup

# link for extract html data

defgetdata(url):

r = requests.get(url)

return r.text

defairinfo():

soup = BeautifulSoup(htmldata, 'html.parser')

res\_data = soup.find(class\_="DonutChart--innerValue--2rO41 AirQuality--extendedDialText--2AsJa").text

air\_data = soup.find\_all(class\_="DonutChart--innerValue--2rO41 AirQuality--pollutantDialText--3Y7DJ")

air\_data=[data.text for data in air\_data]

ar.set(res\_data)

o3.set(air\_data[0])

no2.set(air\_data[1])

so2.set(air\_data[2])

pm.set(air\_data[3])

pml.set(air\_data[4])

co.set(air\_data[5])

res = int(res\_data)

if res <= 50:

remark = "Good" impact = "Minimal impact" elif res <= 100 and res > 51:

remark = "Satisfactory"

impact = "Minor breathing discomfort to sensitive people"

elif res <= 200 and res >= 101:

remark = "Moderate"

impact = "Breathing discomfort to the people with lungs, asthma and heart diseases"

elif res <= 400 and res >= 201:

remark = "Very Poor"

impact = "Breathing discomfort to most people on prolonged exposure"

elif res <= 500 and res >= 401:

remark = "Severe"

impact = "Affects healthy people and seriously impacts those with existing diseases"

res\_remark.set(remark)

res\_imp.set(impact)

# object of tkinter

# and background set to grey

master = Tk()

master.configure(bg='light grey')

# Variable Classes in tkinter

air\_data = StringVar()

ar = StringVar()

o3 = StringVar() no2 = StringVar() so2 = StringVar()

pm = StringVar()

pml = StringVar()

co = StringVar()

res\_remark = StringVar()

res\_imp = StringVar()

# Creating label for each information

# name using widget Label

Label(master, text="Air Quality : ",

bg="light grey").grid(row=0, sticky=W)

Label(master, text="O3 (μg/m3) :", bg="light grey").grid(row=1, sticky=W)

Label(master, text="NO2 (μg/m3) :",

bg="light grey").grid(row=2, sticky=W)

Label(master, text="SO2 (μg/m3) :",

bg="light grey").grid(row=3, sticky=W)

Label(master, text="PM2.5 (μg/m3) :",

bg="light grey").grid(row=4, sticky=W)

Label(master, text="PM10 (μg/m3) :",

bg="light grey").grid(row=5, sticky=W)

Label(master, text="CO (μg/m3) :",

bg="light grey").grid(row=6, sticky=W) Label(master, text="Remark

:",

bg="light grey").grid(row=7, sticky=W)

Label(master, text="Possible Health Impacts :",

bg="light grey").grid(row=8, sticky=W)

# Creating label for class variable

# name using widget Entry

Label(master, text="", textvariable=ar,

bg="light grey").grid(

row=0, column=1, sticky=W)

Label(master, text="", textvariable=o3,

bg="light grey").grid(

row=1, column=1, sticky=W) Label(master, text="", textvariable=no2,

bg="light grey").grid(

row=2, column=1, sticky=W)

Label(master, text="", textvariable=so2,

bg="light grey").grid(

row=3, column=1, sticky=W)

Label(master, text="", textvariable=pm,

bg="light grey").grid(

row=4, column=1, sticky=W)

Label(master, text="", textvariable=pml,

bg="light grey").grid(

row=5, column=1, sticky=W) Label(master, text="", textvariable=co,

bg="light grey").grid(

row=6, column=1, sticky=W)

Label(master, text="", textvariable=res\_remark,

bg="light grey").grid(row=7, column=1, sticky=W)

Label(master, text="", textvariable=res\_imp,

bg="light grey").grid(row=8, column=1, sticky=W)

# creating a button using the widget

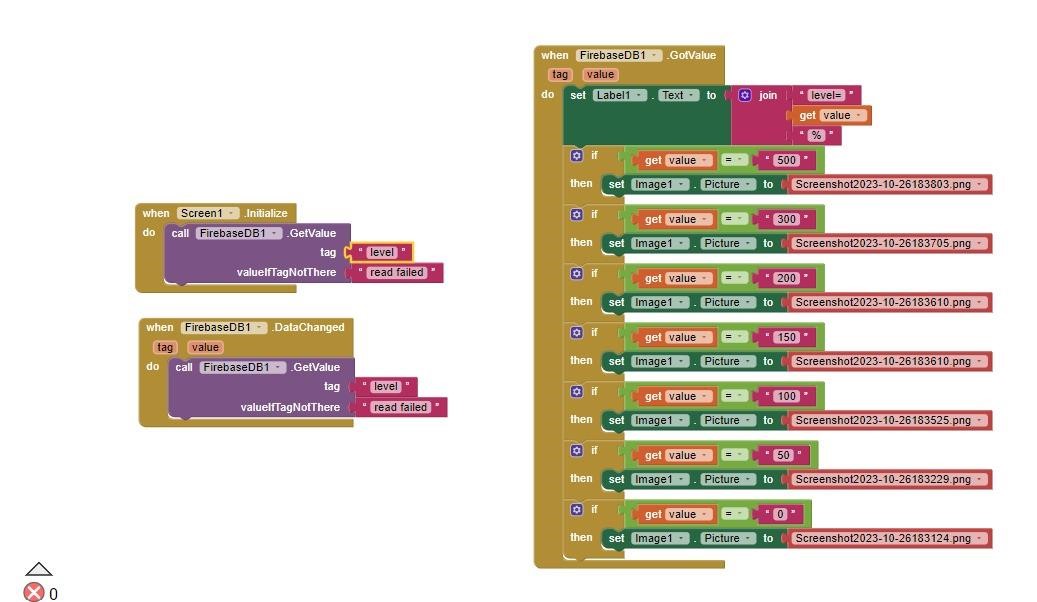
b = Button(master, text="Check",

command=airinfo, bg="Blue")

b.grid(row=0, column=2, columnspan=2,

rowspan=2, padx=5, pady=5,) mainloop()

**Block code for App interface:**



Creating an air quality monitoring app using MIT App Inventor involves several steps. Below is a high-level overview of how you can create such an app:

**Define App Objectives:**

Clearly define the objectives and features of your air quality monitoring app. Determine what air quality metrics you want to measure (e.g., PM2.5, PM10, CO2, etc.) and any additional features (location-based data, historical data, etc.).

Gather Data Sources:

Identify the data sources you will use to obtain air quality information. This could involve APIs from air quality monitoring services, sensors, or other data providers.

Create the User Interface (UI):

Use MIT App Inventor’s visual designer to create the user interface.

Add components like labels, buttons, and a display area to present air quality data.

Set Up Data Retrieval:

Use App Inventor’s features to connect to your chosen data sources. This may involve using the Web component to retrieve data from an API.

Implement logic to fetch air quality data periodically or on-demand.

**Conclusion:**

This project is a versatile and valuable tool for tracking air quality and can be customized to meet specific needs, whether for personal use or community monitoring efforts. It helps raise awareness about air quality issues and empowers individuals and communities to take action to improve environmental conditions.