AIR QUALITY MONITORING

Phase-5

FINAL DEVELOPMENT

INTRODUCTION:

Air quality is a critical aspect of our daily lives, as the air we breathe directly impacts our health and well-being. Monitoring and maintaining good air quality is essential to ensure a safe and healthy environment. In this project, we have developed an Air Quality Monitoring System using Arduino and various sensors, including the MQ-135 gas sensor, DHT11 or DHT22 for temperature and humidity, a flame sensor, and an MQ-2 gas sensor. These sensors play a crucial role in assessing the air quality by detecting various gases and environmental parameters.

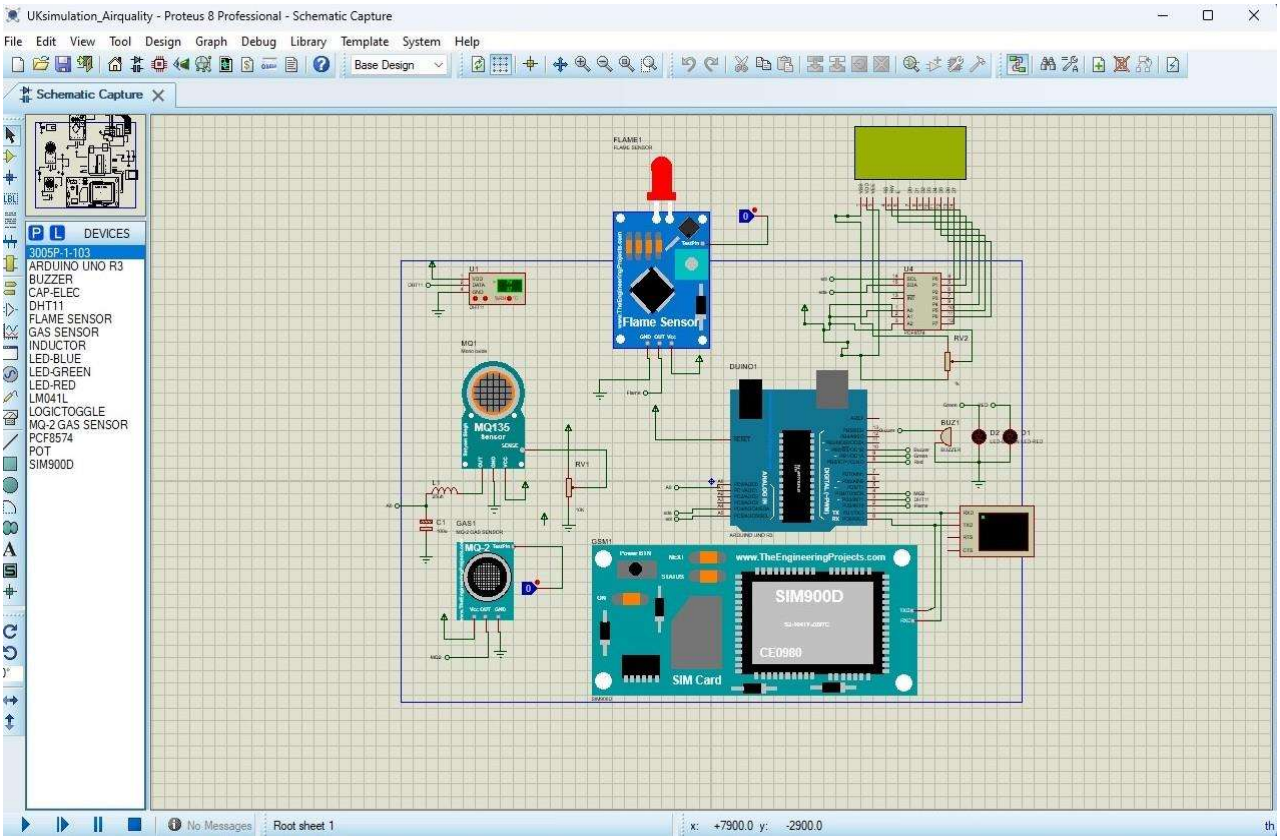
The MQ-135 sensor is a versatile gas sensor capable of detecting a range of gases, including NH3, NOx, alcohol, benzene, smoke, CO2, and more. It provides output in the form of voltage levels, which we can convert to parts per million (PPM) to gauge the pollution level. When there is no gas present, the sensor's baseline reading is around 90 PPM. For reference, safe air quality levels are typically considered to be below 350 PPM, with an upper limit of 1000 PPM. Exceeding this limit can lead to discomfort, such as headaches and sleepiness. Beyond 2000 PPM, the air quality can become hazardous, potentially causing increased heart rate and various health issues.

Our system is designed to provide real-time monitoring of air quality. When the sensor readings are below 1000 PPM, the LCD display and webpage will show "Fresh Air," indicating safe air quality. As the PPM level increases beyond 1000, an integrated buzzer will start beeping to alert users, and the LCD and webpage will display "Poor Air, Open Windows" to encourage ventilation. If the PPM level surpasses 2000, the buzzer will continuously beep, and the LCD and webpage will show "Danger! Move to fresh Air," signaling a hazardous environment.

In addition to the MQ-135 sensor, we have incorporated other sensors like the DHT11 or DHT22 for measuring temperature and humidity, a flame sensor to detect potential fire hazards, and an MQ-2 gas sensor for detecting additional gases. These sensors, when combined, provide a comprehensive understanding of the air quality and environmental conditions in the monitored area.

With this Air Quality Monitoring System, you can take proactive measures to ensure a healthy living or working environment and respond promptly to deteriorating air quality, thereby promoting well-being and safety.

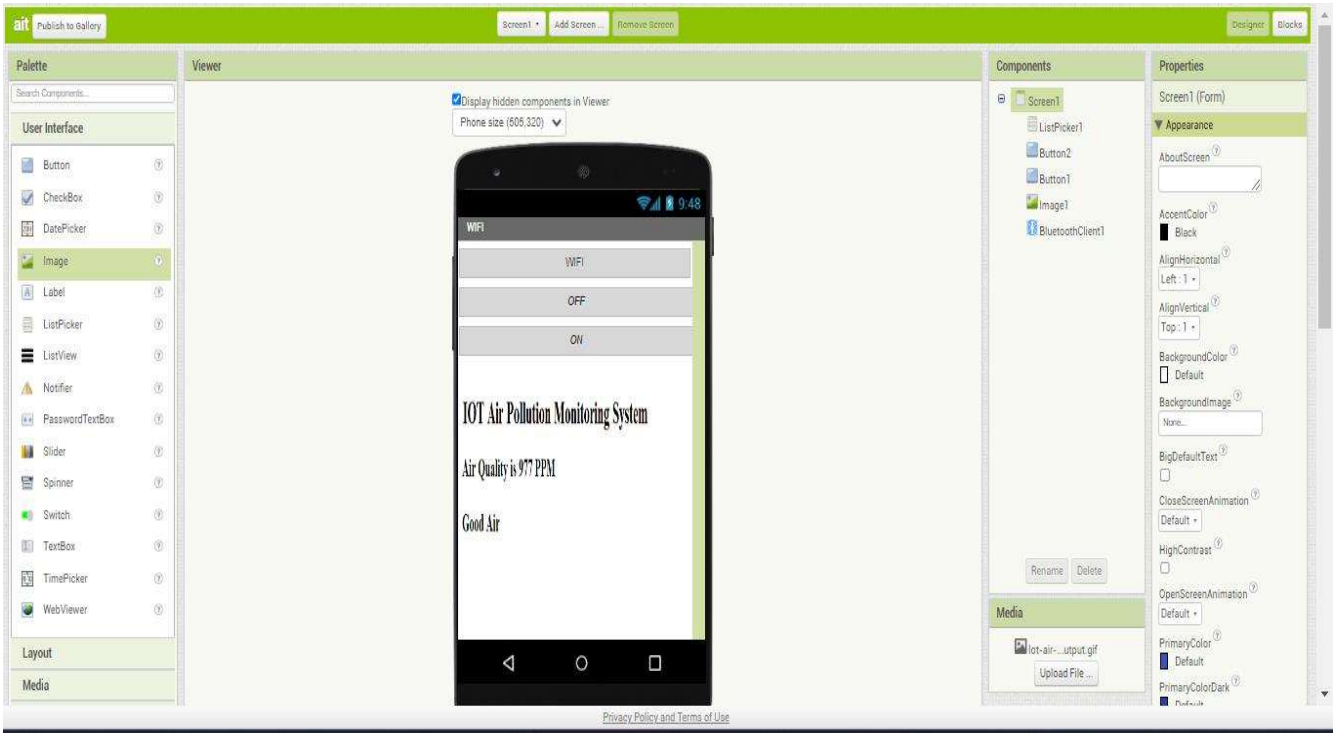
CIRCUIT DIAGRAM:

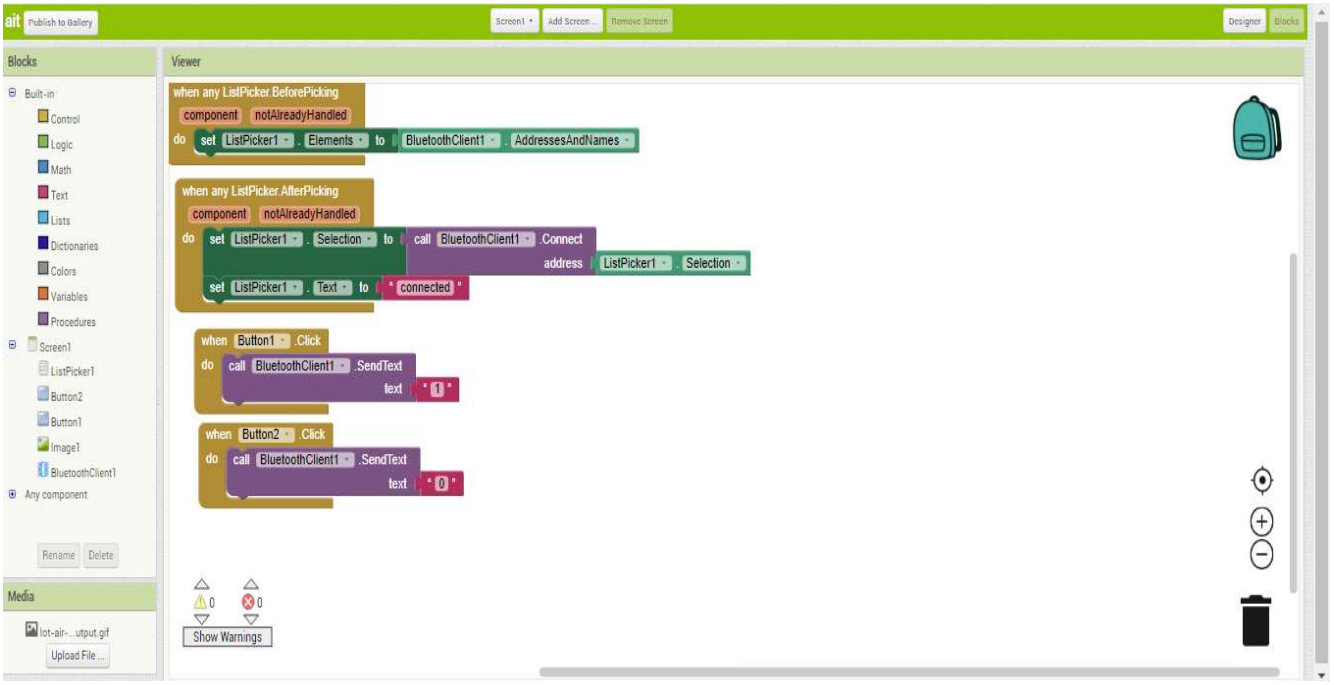


COMPONENTS USED

- ***3005P-1-103***
- ***ARDUINO UNO R3***
- ***BUZZER***
- ***LEDGREEN***
- ***FLAME SENSOR***
- ***GAS SENSOR***
- ***LED-BLUE***
- ***SIM900D***
- ***LEDRED***
- ***INDUCTOR***
- ***PCF8574***
- ***MO-2 GAS SENSOR***
- ***LOGICTOGGLE***
- ***DHTI 1***

CREATED APP:


Publish to Gallery
Screen1
Add Screen ...
Remove Screen
Designer
Blocks
Palette
Search Components...
User Interface
Button
CheckBox
DatePicker
Image
Label
ListPicker
ListView
Notifier
PasswordTextBox
Slider
Spinner
Switch
TextBox
TimePicker
WebViewer
Layout
Media
Viewer
Display hidden components in Viewer
Phone size (505,320)
9:48
WIFI
WIFI
OFF
ON
IOT Air Pollution Monitoring System
Air Quality is 977 PPM
Good Air
Components
Screen1
ListPicker1
Button2
Button1
Image1
BluetoothClient1
Rename
Delete
Media
Iot-air-...utput.gif
Upload File ...
Properties
Screen1 (Form)
Appearance
AboutScreen
AccentColor
Black
AlignHorizontal
Left : 1
AlignVertical
Top : 1
BackgroundColor
Default
BackgroundImage
None...
BigDefaultText
CloseScreenAnimation
Default
HighContrast
OpenScreenAnimation
Default
PrimaryColor
Default
PrimaryColorDark
Privacy Policy and Terms of Use


Publish to Gallery
Screen1
Add Screen ...
Remove Screen
Designer
Blocks
Blocks
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Screen1
ListPicker1
Button2
Button1
Image1
BluetoothClient1
Any component
Rename
Delete
Media
Iot-air-...utput.gif
Upload File ...
Viewer
when any ListPicker.BeforePicking
component notAlreadyHandled
do set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
when any ListPicker.AfterPicking
component notAlreadyHandled
do set ListPicker1 . Selection to call BluetoothClient1 .Connect
address ListPicker1 . Selection
set ListPicker1 . Text to " connected "
when Button1 .Click
do call BluetoothClient1 .SendText
text " 1 "
when Button2 .Click
do call BluetoothClient1 .SendText
text " 0 "
0
0
Show Warnings

CODE:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
#include <DHT.h>
#include <SoftwareSerial.h>
#define DHTPIN 3
#define DHTTYPE DHT22
#define MQ135 A0
#define Flame_Sen 2
#define MQ2 4
#define Red 8
#define Green 9
#define Buzzer 10
SoftwareSerial esp8266(6, 7); // Change the pins according to your ESP8266 connections
int MQ2_output;
int F_output;
int Gas_Sensor;
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  esp8266.begin(9600);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Air Quality");
  lcd.setCursor(0, 1);
  lcd.print("System");
  delay(1000);

  dht.begin();
```

```
  pinMode(MQ135, OUTPUT);
  pinMode(Flame_Sen, OUTPUT);
  pinMode(MQ2, OUTPUT);
  pinMode(Green, OUTPUT);
  pinMode(Red, OUTPUT);
  pinMode(Buzzer, OUTPUT);
  lcd.clear();
  // Connect to your Wi-Fi network (replace with your network credentials)
  esp8266.println("AT+CWJAP=\"Your_SSID\",\"Your_Password\"");
  delay(3000);
}
void loop() {
  // Gas Sensor
  Gas_Sensor = analogRead(MQ135);
  Gas_Sensor = map(Gas_Sensor, 0, 1023, 0, 100);
  Serial.print("Carbon_monoxide  Present in air=");
  Serial.print(Gas_Sensor);
  Serial.println("%");
  delay(200);
  // Flame Sensor and MQ2
  F_output = digitalRead(Flame_Sen);
  MQ2_output = digitalRead(MQ2);
  if (F_output == HIGH) {
    Serial.println("Fire  Detected Please take action");
  }
  if (MQ2_output == HIGH) {
    Serial.println("Smoke  Detected Please take action as soon as possible");
  }


  // DHT22 Sensor
  float h = dht.readHumidity();
```

```
float t = dht.readTemperature();
if (isnan(h) || isnan(t)) {
  Serial.println(F("Failed  to read from DHT sensor!"));
  return;
}
Serial.print(F("Humidity: "));
Serial.print(h);
Serial.print("%  Temperature: ");
Serial.print(t);
Serial.print("°C");
lcd.setCursor(0, 0);
lcd.print("Air Temp=");
lcd.setCursor(9, 0);
lcd.println(t);
lcd.setCursor(0, 1);
lcd.print("Humidity=");
lcd.setCursor(10, 1);
lcd.println(h);

// Wi-Fi functionality
if (esp8266.available())  {
  if (esp8266.find("Ready")) {
    esp8266.println("AT+CIPSTART=\"TCP\",\"your_server_address\",your_server_port");
    if (esp8266.find("OK")) {
      esp8266.println("AT+CIPSEND");
      delay(1000);

      String data = "<h1>Environmental Data</h1>";
      data += "<p>Temperature: ";
      data += t;
      data += "°C</p>";
```

```
    data += "<p>Humidity: ";
    data += h;
    data += "%</p>";
    data += "<p>Gas Sensor: ";
    data += Gas_Sensor;
    data += "%</p>";
    esp8266.print("AT+CIPSEND=");
    esp8266.println(data.length());
    delay(1000);
    esp8266.println(data);
    delay(1000);
    esp8266.println("AT+CIPCLOSE");
   }
  }
}


// Control outputs based on sensor readings
if (((t > 25) && (t < 35)) || (F_output == LOW) || (MQ2_output == LOW)) {
  digitalWrite(Red, LOW);
  digitalWrite(Green, HIGH);
  digitalWrite(Buzzer, LOW);
}

if ((t > 36) || (F_output == HIGH) || (MQ2_output == HIGH)) {
  digitalWrite(Red, HIGH);
  digitalWrite(Green, LOW);
  digitalWrite(Buzzer, HIGH);
}

if (t > 36) {
  Serial.println("Temperature  Increase Please take action ASAP");
```

```
  }
}
```

STEPS INVOLVED

Step 1: Import Libraries

- Import necessary libraries to enable communication with sensors and modules. These libraries include Wire, LiquidCrystal_I2C, DHT, and SoftwareSerial.

Step 2: Define Pin Constants

Define pin constants for the various components used in the system. These include the DHT22 sensor, MQ135 sensor, flame sensor, MQ2 sensor, and pins for LEDs and a buzzer.

Step 3: Initialize SoftwareSerial and LCD

- Initialize SoftwareSerial to communicate with the ESP8266 module.
- Initialize the LCD display for showing air quality data.

Step 4: Setup Function

- In the `setup` function:
- Initialize serial communication with a baud rate of 9600 for debugging.
- Begin communication with the ESP8266 module using the SoftwareSerial object.
- Initialize the LCD display and print "Air Quality System" on the first line and "System" on the second line.
- Initialize the DHT22 sensor for temperature and humidity readings.
- Set pin modes for the MQ135 sensor, flame sensor, MQ2 sensor, and output pins for LEDs and the buzzer.
- Clear the LCD screen to prepare for displaying data.
- Establish a Wi-Fi connection by sending appropriate AT commands to the ESP8266.

Step 5: Loop Function

In the `loop` function, the following tasks are performed:

Gas Sensor Reading:

- Read the analog value from the MQ135 sensor.
- Map the analog value to a percentage to represent carbon monoxide presence in the air.
- Print the gas sensor reading to the serial monitor.
- Introduce a delay of 200 milliseconds.
- Flame Sensor and MQ2 Reading:
- Read digital values from the flame sensor and MQ2 sensor.
- If the flame sensor detects a flame (HIGH), print "Fire Detected Please take action."
- If the MQ2 sensor detects smoke (HIGH), print "Smoke Detected Please take action as soon as possible."

DHT22 Sensor Reading:

- Read temperature and humidity values from the DHT22 sensor.
- Check for errors in the sensor reading to ensure valid data.
- Print humidity and temperature values to the serial monitor.
- Display the temperature and humidity on the LCD screen.

Wi-Fi Functionality:

- Check if data is available from the ESP8266 module.
- If the module is ready, establish a TCP connection to a server with the appropriate address and port.
- Send environmental data (temperature, humidity, gas sensor reading) to the server.
- Close the TCP connection.

Control Outputs:

- Based on the sensor readings and temperature conditions:
- If the temperature is between 25°C and 35°C, or if the flame and MQ2 sensors are LOW, turn on the green LED and turn off the red LED and the buzzer.
- If the temperature is above 36°C, or if the flame and MQ2 sensors are HIGH, turn on the red LED and the buzzer and turn off the green LED.

- If the temperature is above 36°C, print "Temperature Increase Please take action ASAP."

This code continually monitors environmental parameters, displays them on the LCD, and provides alerts and actions based on sensor readings. It also sends data to a remote server for remote monitoring.

CONCLUSION:

In this project, we've built a system to check the air quality around us. We used a bunch of sensors like MQ-135 to measure different gases, DHT22 for temperature and humidity, a flame sensor, and MQ-2 to detect more gases. These sensors help us know if the air is clean and safe to breathe.

The MQ-135 sensor tells us about the gases in the air. When there's no bad stuff in the air, it starts at 90 parts per million (PPM). Good air is usually under 350 PPM, and it's risky if it goes above 1000 PPM. Beyond 1000 PPM, it can make you feel unwell. If it's over 2000 PPM, it's really dangerous and can make you sick.

Our system keeps an eye on the air quality all the time. When the air is good, it shows "Fresh Air" on the screen. If the air gets a bit bad (above 1000 PPM), a buzzer beeps, and it tells you to "Open Windows." If it gets really bad (over 2000 PPM), the buzzer keeps beeping, and it warns you to "Move to fresh Air."

We also added other sensors. DHT22 tells us the temperature and humidity, and we can see that on the screen too. The flame sensor can alert us if there's a fire, and MQ-2 can spot more gases.

With this system, we can make sure the air we breathe is safe and healthy. It keeps us informed and warns us when things aren't right. It's all about looking out for our well-being and keeping us safe