AIR QUALITY MONITORING

Team Members:
Mukesh M (810021106051)
Sriram S (810021106081)
Vengatesan M (810021106091)
Vengatesan T (810021106092)
Sushmitha K (810021106083)

AIR QUALITY MONITORING

IOT_Phase3 Project

Introduction:

In this project we will see about the design and source code for IOT based air quality monitoring using Arduino , MQ135, DHT22 and ESP8266 Wi-Fi module . The description and model of each components are included in the project.

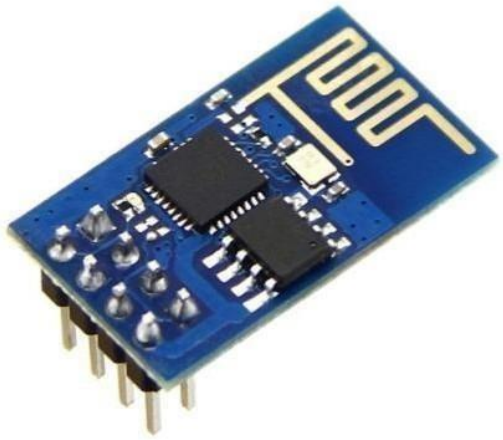
Details about components:

ARDUINO :

Arduino Uno

The Arduino Uno is the board we will be using for this project. Other Arduino boards may work for this project as well, but this board is all I needed. It's a great price and we can purchase it on many different websites.

ESP8266 Wi-Fi chip:



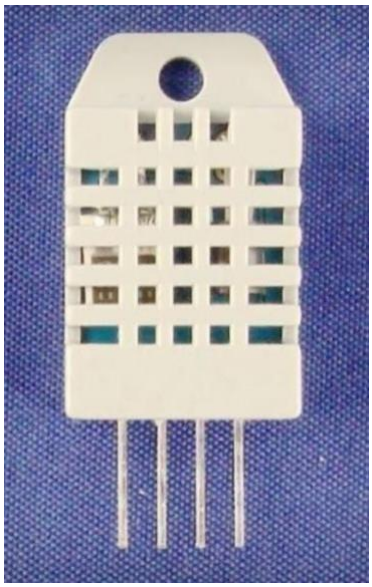
ESP8266 WiFi Chip

This is the chip that will allow our project to send data wirelessly. In order to program it, we must build a separate circuit and use a separate code then we will use for the main air quality monitor build.

MQ-135 Sensor:

The MQ-135 gas sensor is capable of detecting a bunch of different gases. We could build an air quality sensor for individual gases, such as carbon monoxide, but that would only give us a small snapshot of the overall air quality.

DHT-22 Sensor:



DHT22 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity

sensing technology, assuring its reliability and stability.Its sensing elements is connected with 8-bit single-chip.

ESP8266 Python Code (for forwarding data to ThingSpeak):

```
import serial

import requests

import time


ser = serial.Serial('COM4', 9600)  # Replace 'COM4' with our ESP8266's serial port


def send_to_thingspeak(api_key, field1, field2, field3):

    url = f"https://api.thingspeak.com/update?api_key={api_key}&field1={field1}&field2={field2}&field3={field3}"

    response = requests.get(url)

    if response.status_code == 200:

        print("Data sent to ThingSpeak successfully")

    else:

        print("Failed to send data to ThingSpeak")
```

```
while True:
    data = ser.readline().decode('utf-8').strip()
    print(data)

    if data.startswith("SENDING "):
        data = data[len("SENDING "):]
        values = data.split(",")
        if len(values) == 3:
            temperature, humidity, air_quality = values
            send_to_thingspeak("OUR_API_KEY", temperature, humidity, air_quality)

    time.sleep(10)  # Adjust the delay as needed
```

Make sure to replace `"OUR_API_KEY"` in the Python script with our actual ThingSpeak API key. Also,adjust the serial port name in the ESP8266 Python script to match our specific setup.

This code reads temperature, humidity, and air quality data from DHT22 and MQ135 sensors connected to the Arduino, sends it to the ESP8266 through serial communication, and then forwards it to ThingSpeak.

To connect an Arduino with a DHT22 temperature and humidity sensor and an MQ135 gas sensor to ThingSpeak through an ESP8266 WiFi module, we'll need to write code for both the Arduino and the ESP8266 module. Here's a basic example of the Arduino code and the Python script to collect data and send it to ThingSpeak:

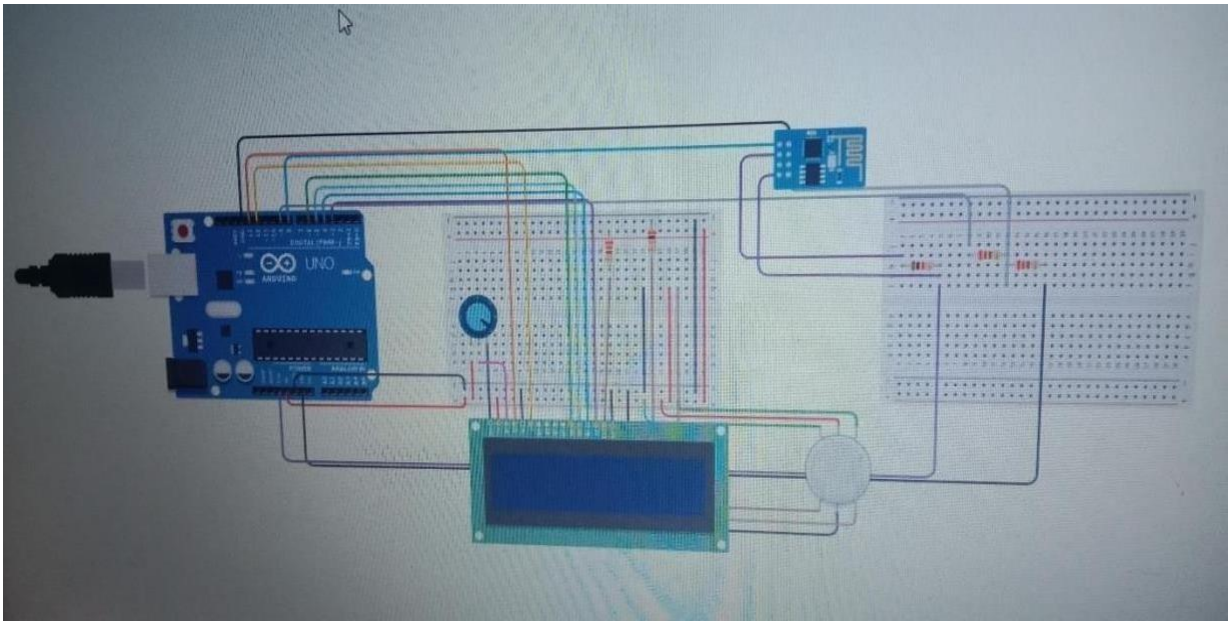
Arduino Code (for collecting sensor data):

```
#include "DHT.h"
#include "Adafruit_Sensor.h"
#include <MQ135.h>
```

```
#define DHTPIN 2  // DHT22 data pin
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

MQ135 gasSensor = MQ135(A0);

void setup() {
  Serial.begin(9600);
  ThingSpeak.begin(client);  // Initialize ThingSpeak
}

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  float airQuality = gasSensor.getPPM();

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print("°C, Humidity: ");
  Serial.print(humidity);
  Serial.print("%, Air Quality: ");
  Serial.print(airQuality);
  Serial.println(" ppm");

  // Send data to ESP8266 for forwarding to ThingSpeak
  Serial.print("SENDING ");
  Serial.print(temperature);
  Serial.print(",");
  Serial.print(humidity);
```

```
  Serial.print(",");
  Serial.print(airQuality);
  Serial.println(" TO ESP8266");

  // Send data to ESP8266 here using Serial communication
  Serial.print("AT+CIPSEND=");
  Serial.println(data.length());
  Serial.print("GET /update?api_key=OUR_API_KEY&field1=");
  Serial.print(temperature);
  Serial.print("&field2=");
  Serial.print(humidity);
  Serial.print("&field3=");
  Serial.print(airQuality);
  Serial.println(" HTTP/1.0");
  Serial.println("Host: api.thingspeak.com");
  Serial.println("Content-Type: application/x-www-form-urlencoded");
  Serial.println("User-Agent: ESP8266");
  Serial.println();
  delay(10000);  // Update every 10 seconds
}
```

Circuit Design:



•Connect the ESP8266 module to the Arduino as follows:

- ESP-01 VCC to 3.3V on the Arduino
- ESP-01 GND to GND on the Arduino
- ESP-01 TX to an Arduino RX pin (e.g., RX on the Arduino)
- ESP-01 RX to an Arduino TX pin (e.g., TX on the Arduino)
- GPIO0 on the ESP-01 should be connected to GND during programming.

•Connect the MQ-135 sensor to the Arduino as follows:

- MQ-135 VCC (or +) pin to Arduino 5V

- MQ-135 GND (or-) pin to Arduino GND

- MQ-135 AOUT pin to an analog input pin on the Arduino (e.g., A0)

- MQ-135 DOUT (or digital output) is optional and can be connected to a digital input pin if we want to use it, but it's not necessary for basic readings.

• - Connect the DHT22 sensor to the Arduino as follows:

- DHT22 VCC pin to Arduino 5V

- DHT22 GND pin to Arduino GND

- DHT22 DATA pin to a digital pin on the Arduino (e.g., D2)

- Place a 10kΩ resistor between the VCC and DATA pins of the DHT22 sensor.

Conclusion:

In conclusion, IoT-based air quality monitoring using Arduino presents a powerful and cost-effective solution for real-time environmental data collection. This system allows for continuous monitoring of air quality parameters, such as particulate matter, gases, and temperature, and enables data transmission to a central server for analysis. By leveraging Arduino's versatility and the connectivity of IoT, we can not only track air quality but also make informed decisions for public health, urban planning, and environmental protection. This technology has the potential to enhance our understanding of air quality and contribute to the creation of cleaner and healthier living environments.