IOT BASED AIR QUALITY CHECKING SYSTEM USING MQ135

Pratik Raipure (24305)
Nachiket Kathoke(24611)

Abstract

This paper deals with measuring the Air Quality using MQ135 sensor. Measuring Air Quality is an important element for bringing awareness to take care of the future generations and for a healthier life. Based on this, Government of India has already taken certain measures to ban Single Stroke and Two Stroke Engine based motorcycles which are emitting high pollution. We are trying to implement a system using IoT platforms like Thingspeak in order to bring awareness to every individual about the harm we are doing to our environment. Already, New Delhi is remarked as the most pollution city in the world recording Air Quality above 1000 PPM. We have used easiest platform like Thingspeak and set the dashboard to public such that everyone can come to know the Air Quality at the location where the system is installed.

Introduction

Air pollution sensors are devices that can detect the presence of air pollution in the surroundings, as the name suggests. The detection is in terms of a ppm value of the pollutants the sensor can detect, and then those ppm values can be used to monitor the environment, based on standard safety values. There are different types of sensors, depending on the usage. The major ones are specialized for one or more of the following components: Ozone, PM, Carbon Monoxide, Sulphur Dioxide and Nitrous Oxide. Our project is a simple demonstration to show that it can be easily done. We have used an MQ135 Gas Sensor, which is sensitive to gases like Ammonia, Benzene and Carbon Dioxide.

Air Pollution is increasing heavily these days due to the many important factors like Vehicle Emissions, Deforestation, Industrialization. Old vehicles could produce more smoke and hence those vehicles could be banned from using. But we need a proper metric that tells us whats the intensity level of air pollution, whether it is low, medium or high. So, we can take help of various sensors that could detect the air quality and from the corresponding values, we do mathematical calculations and from that results, we could fix the threshold value from which we can say whether the desired air quality is below the threshold value or not.

MQ135

The MQ-135 Gas sensors are used in air quality control equipments and are suitable for detecting or measuring of NH3, NOx, Alcohol, Benzene, Smoke, CO2. The MQ-135 sensor module comes with a Digital Pin which makes this sensor to operate even without a microcontroller and that comes in handy when you are only trying to detect one particular

gas. If you need to measure the gases in PPM the analog pin need to be used. The analog pin is TTL driven and works on 5V and so can be used with most common microcontrollers.
If you are looking for a sensor to detect or measure common air quality gases such as CO2, Smoke, NH3, NOx, Alcohol, Benzene then this sensor might be the right choice for you. MQ-135 gas sensor applies SnO2 which has a higher resistance in the clear air as a gas-sensing material. When there is an increase in polluting gases, the resistance of the gas sensor decreases along with that.

For our Project we have considered following air quality index:

Air Quality Index

Range(PPM)	**Status**
0-1000	Good
1001-2000	Moderate
2001-3000	Unhealthy for sensitive groups
3000<	Hazardous

Required Components:

- MQ135 Gas Sensor

- Arduino UNO

- ESP8266-01

.

- Breadboard
- Jumper Wires

Keywords: Thingspeak, Android Studio

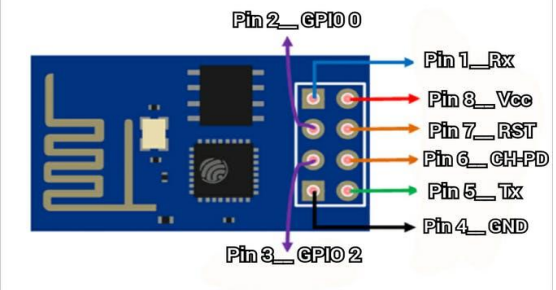
Circuit and Explanation

First we needed to calibrate the circuit. For that we setup the circuit (minus the Wi-Fi Module), by doing appropriate connections,

- For the MQ135 sensor, the analog pin of the sensor was connected to the analog pin (A0) of the arduino. Vcc and ground pins were appropriately connected.

After this we checked the connections thoroughly. We wanted to test if the primary circuit was working, so we wrote a minimal code and uploaded it to see if it works. As it was expected, output values were arbitrary (no calibration done), but our sensor was sensitive to air from a person's mouth and also, to the hydrocarbons present in LPG (obtained from general lighter). Whenever we bring a lighter (not burning but emitting LPG) close, the ppm value rose exponentially. We also tested the buzzer by lowering the ppm threshold for polluted air, and it was working, a further testimony to the fact that our circuit was correct.

ESP8266-01 : PINOUT



- pin 1 _____ Rx -- connect it to Rx of Arduino
- pin 2_____GPIO 0--connect it to ground while uploading the code to arduino IDE
- pin 3_____GPIO 2,
- pin 4_____ GND--connect it to ground
- pin 5_____Tx--connect it to Tx of Arduino
- pin 6_____CH_PD(EN) -- connect it to 3.3v
- pin 7_____RST(reset)--(not necessary) connect it to 3.3v for normal operation and 0v(GND) for reset
- pin 8____Vcc--supply 3.3v from Arduino or from an external source

After doing this connection upload a code by Arduino IDE and then remove GPIO 0 and interchange the RX and TX connection respectively i.e RX of wifi module to TX of Arduino and vice versa.

For running the Wi-Fi Module, we used a Mobile Hotspot, and connected it to our PC and the ESP. The data by ESP was being sent across through the Wi-Fi, which could be read by the terminal using appropriate code.

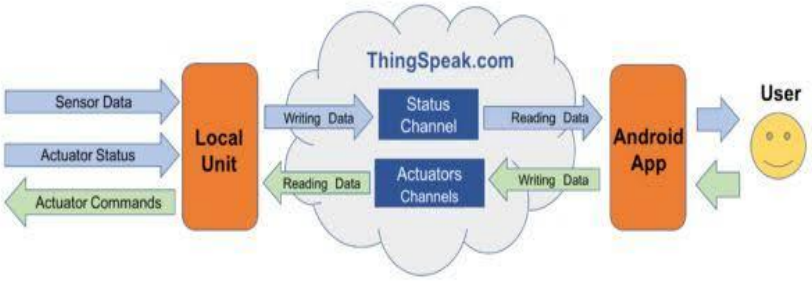
Note:- Connect the Arduino board to the computer and open the serial monitor of Arduino and check for connectivity using the following command.

- AT //Use to check ESP8266-01 connection
- Note that I have set the baud rate to 115200 and also selected “Both NL & CR” option in the Serial Monitor.
-

- After you get a response as “OK”, you can proceed with connecting your ESP Module to your WiFi Network using the following command.
-
-
- AT+CWJAP=”SSID”,”PASSWORD”
- Replace SSID with the name of your WiFi Network and enter the password in place of PASSWORD.
- You will now get a confirmation response regarding wifi connection.

Implementation

Our goal will be to basically collect information from a local unit and send it to the internet. A user anywhere on the planet looking at this information will make decisions by sending remote commands to the actuators, which will also be in this local unit. Any sensor or actuator could be used. After connecting the ESP-01 successfully to the hotspot, it gets established with Thingspeak wesbite and the account API Key is written in Arduino Code which helps to save the data only to our account bearing the given API key. Thingspeak needs 15 seconds of refresh interval to push to the data.



Connection of android studio to ThingsSpeak :

- Use asynTask to perform activity in background and posting it after done
- For getting data from TingsSpeak use url object
- HttpURLConnection httpurlconnection = (HttpURLConnection) url.openConnection() ;
- InputStream inputStream = httpurlconnection.getInputStream();
- BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
- String line = "" ;
- Using loop i read the json file from bufferreader
- After that i convert json file to The the json file to json array.
- From json array i get json object from that i get a string of required output.
- After this I set the values in UI.

Conclusion:

As mentioned we can calculate the air quality in PPM. The problem with MQ135 sensor is that specifically it cannot tell the Carbon

Monoxide or Carbon Dioxide level in the atmosphere, but the pros of MQ135 is that it is able to detect smoke. This project can be used both for indoor as well as outdoor. For indoor, we can make this kit as a compact device such that if every home started using the device, we can monitor the indoor air quality of a particular targeted area. Due to increasing air pollution, there is necessity to keep an eye on Indoor air quality too. But for outdoor purpose, certainly one sensor is not sufficient because one sensor has a sensitivity range of around 1 meter, so a network of sensors has to be deployed to monitor the outdoor air quality. Enough care is taken while calibrating the sensors. In this project, I have shown you how to connect ESP8266 to ThingSpeak API using both the direct AT Commands as well as through Arduino.

Code:

Arduino Code:

```
#include <MQ135.h>

float voltage;
String A = "GET /update?api_key=UHQ8UGU5ESR3YZUT&field1=0";
String Z = " HTTP/1.1 \nHOST: api.thingspeak.com \r\n\r\n";   //web link
MQ135 gasSensor = MQ135(A0) ;
void setup()
{
    Serial.begin(115200);
    Serial.println("AT\r\n");
    delay(2000);
}
void loop()
{
/* VOLTAGE */
  float voltage = analogRead(A0);
  char voltage_buff[16];
  String voltageX = dtostrf(voltage, 4, 1, voltage_buff);
  String postStr = A + voltageX + Z;
/*********   Uploading to ThinkSpeak Cloud    **********/
/*********  Sending AT Commands to ESP8266    **********/
  Serial.print("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");
  delay(3000);
  String ciplength = "AT+CIPSEND=" + String(postStr.length()) + "\r\n";
  Serial.print(ciplength);
  delay(3000);
  Serial.print(postStr);
  delay(3000);
}

```

Android Studio App Code:

- **mainActivity.java**

```
        refresh = new Runnable() {
            public void run() {
                start();
                handler.postDelayed(refresh, 15000);
            }
        };
        handler.post(refresh);
    }
    private void start() {
        click.setVisibility(View.GONE);
        FatchData process = new FatchData();
        process.execute();
    }
    private void init(){
        txt_end = findViewById(R.id.txt_end);
        click  = findViewById(R.id.button);
        gauge = (CustomGauge) findViewById(R.id.gauge1);
        value_ppm = findViewById(R.id.textView);
        condi = findViewById(R.id.txt_condi);
        gauge.setStartValue(0);
        gauge.setEndValue(2000);
    }
}

```

```
package com.example.airquality;

import androidx.appcompat.app.AppCompatActivity;
import pl.pawelkleczkowski.customgauge.CustomGauge;

import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    Runnable refresh ;
    Handler handler = new Handler();
    public static Button click ;
    public static CustomGauge gauge ;
    public static TextView value_ppm ,condi , txt_end;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        init();
        start();
        click.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                start();
                FatchData process = new FatchData();
                process.execute();
            }
```

- **Activity.xml**

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:text="Refresh"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/gauge1"
        app:layout_constraintVertical_bias="0.498"
        tools:ignore="MissingConstraints" />

    <pl.pawelkleczkowski.customgauge.CustomGauge
        android:id="@+id/gauge1"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_below="@+id/button"
        android:layout_centerHorizontal="true"
        android:layout_marginStart="50dp"
        android:layout_marginTop="192dp"
        android:layout_marginBottom="531dp"
        android:layout_marginBottom="531dp"
        android:paddingLeft="20dp"
        android:paddingTop="20dp"
        android:paddingRight="20dp"
        android:paddingBottom="20dp"
        app:gaugeEndValue="1000"
        app:gaugePointEndColor="@color/colorAccent"
        app:gaugePointSize="6"
        app:gaugePointStartColor="@color/colorAccent"
        app:gaugeStartAngle="135"
        app:gaugeStartValue="0"
        app:gaugeStrokeCap="ROUND"
        app:gaugeStrokeColor="@color/colorPrimaryDark"
        app:gaugeStrokeWidth="10dp"
        app:gaugeSweepAngle="270"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.341"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        tools:ignore="MissingConstraints" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:textSize="36sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/gauge1"
        app:layout_constraintVertical_bias="0.12" />

    <TextView
        android:id="@+id/txt_condi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:textSize="36sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.067" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="37dp"
        android:layout_marginEnd="9dp"
        android:text="0"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/gauge1"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.443" />

    <TextView
        android:id="@+id/txt_end"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="15dp"
        android:layout_marginEnd="8dp"
        android:text="2000"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toEndOf="@+id/gauge1"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.432" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

- **FatchData Class**

```java
package com.example.airquality;

import android.graphics.Color;
import android.os.AsyncTask;
import android.view.View;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

public class FatchData extends AsyncTask<Void,Void,Void> {
    String data = "";
    public static  String value;
    @Override
    protected Void doInBackground(Void... voids) {
        try {
            URL url = new URL("https://api.thingspeak.com/channels/1124845/fields/1.json?api_key=54C6QJNZ7EI9I3J2&results=2");
            HttpURLConnection httpurlconnection = (HttpURLConnection) url.openConnection() ;
            InputStream inputStream = httpurlconnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader( new InputStreamReader(inputStream));
            String line = "" ;
            while(line != null){
                line = bufferedReader.readLine();
                data = data + line ;
            }

        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
    @Override
    protected void onPostExecute(Void aVoid) {
        super.onPostExecute(aVoid);
        try {
            JSONObject jo = new JSONObject(data);
            JSONArray sys  = jo.getJSONArray("feeds");

            this.value= sys.getJSONObject(1).getString("field1");

        } catch (JSONException e) {
            e.printStackTrace();
        }
        float val = Float.parseFloat(value);
        MainActivity.gauge.setValue((int)val);
        val = val*10 ;
        MainActivity.value_ppm.setText(Float.toString(val));
        MainActivity.click.setVisibility(View.VISIBLE);

        set_values(this.value);

    }
```

```
61      public void  set_values(String Value){
62          float val = Float.parseFloat(Value) ;
63          if (val < 1000){
64              MainActivity.gauge.setEndValue(3000);
65              MainActivity.txt_end.setText("3000");
66              MainActivity.condi.setText("Fresh Air");
67              MainActivity.condi.setTextColor(Color.GREEN);
68          }
69          else if (val < 2000){
70              MainActivity.gauge.setEndValue(3000);
71              MainActivity.txt_end.setText("3000");
72              MainActivity.condi.setText("Modarte Air");
73              MainActivity.condi.setTextColor(Color.YELLOW);
74          }
75          else {
76              if (val  >= 2000){
77                  MainActivity.gauge.setEndValue(7000);
78                  MainActivity.txt_end.setText("7000");
79              }
80              else {
81                  MainActivity.gauge.setEndValue(2000);
82                  MainActivity.txt_end.setText("2000");
83              }
84              MainActivity.condi.setText("Harmful Air");
85              MainActivity.condi.setTextColor(Color.RED);
86          }
87      }
88  }
89
```