**PROJECT REPORT**

**ON**

**CLOUD BASED AIR QUALITY MONITORING SYSTEM**

SUBMITTED TO :-

DR. SACHIN SHARMA

SUBMITTED BY :-

DEVANSH GAUR

**Problem statement:**

To know about which area contains more pollution and escape from that particular place easily.

**Introduction:**

Air pollution has become a serious issue in the present times. The CO2 level is increasing due to human activities. the effect of pollution is greater in closed spaces like living rooms, cars and offices. This project is developed to detect the level of air pollution and control the pollution level in a restricted area by activating an air purifier as the level exceeds a threshold limit. The system can be installed in houses, offices or cars. The project is developed in an adaptive manner. The device first takes multiple samples of air and determines a threshold level. If the level of CO2 increases the threshold level, the air purifier is activated through a relay circuit.

**Software Requirement:**

* ARDUINO 1.6.13 software

### Required Components:

* MQ135 Gas sensor
* Arduino Uno
* Wi-Fi module ESP8266
* 16X2 LCD
* Breadboard
* 10K potentiometer
* 220 ohm resistor

**Arduino Uno (R3) :**

The Uno is a great choice for first Arduino. It’s got everything we need to get started, and nothing we don’t. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a USB connection, a power jack, a reset button and more. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

**CIRCUIT DIAGRAM AND EXPLANATION :**

First of all we will connect the ESP8266 with the Arduino. ESP8266 runs on 3.3V and if you will give it 5V from the Arduino then it won’t work properly and it may get damage. Connect the VCC and the CH\_PD to the 3.3V pin of Arduino. The RX pin of ESP8266 works on 3.3V and it will not communicate with the Arduino when we will connect it directly to the Arduino. So, we will have to make a voltage divider for it which will convert the 5V into 3.3V. This can be done by connecting three resistors in series like we did in the circuit. Connect the TX pin of the ESP8266 to the pin 10 of the Arduino and the RX pin of the esp8266 to the pin 9 of Arduino through the resistors.The ESP8266 Wi-Fi Module is used to connect with any available internet hotspot and transfer sensor data to ThingSpeak Platform via Wi-Fi. The ESP8266 Wi-Fi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to a Wi-Fi network.The ESP8266 is capable of either hosting an application or off loading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware.  So, one can simply hook this up to an Arduino device. Here it uploads the monitoring data to the cloud It is a very cheap device and make your projects very powerful. It can communicate with any microcontroller and it is the most leading devices in the IOT platform.

Then we will connect the MQ135 sensor with the Arduino. Connect the VCC and the ground pin of the sensor to the 5V and ground of the Arduino and the Analog pin of sensor to the A0 of the Arduino. MQ-135 is a gas sensor which is used to measure the concentration of combustible gases. It has lower conductivity in clean air while its conductivity increases with the presence of the combustible gases in the air. The sensor is highly sensitive to gases like Ammonia, Sulphide and Benzene steam. The sensor can detect the concentration of combustible gases in range from 100 PPM to 1000 PPM.

In last, we will connect LCD with Arduino. The connections of the LCD are as follows

* Connect pin 1 (VEE) to the ground.
* Connect pin 2 (VDD or VCC) to the 5V.
* Connect pin 3 (V0) to the middle pin of the 10K potentiometer and connect the other two ends of the potentiometer to the VCC and the GND. The potentiometer is used to control the screen contrast of the LCD. Potentiometer of values other than 10K will work too.
* Connect pin 4 (RS) to the pin 12 of the Arduino.
* Connect pin 5 (Read/Write) to the ground of Arduino. This pin is not often used so we will connect it to the ground.
* Connect pin 6 (E) to the pin 11 of the Arduino. The RS and E pin are the control pins which are used to send data and characters.
* The following four pins are data pins which are used to communicate with the Arduino.

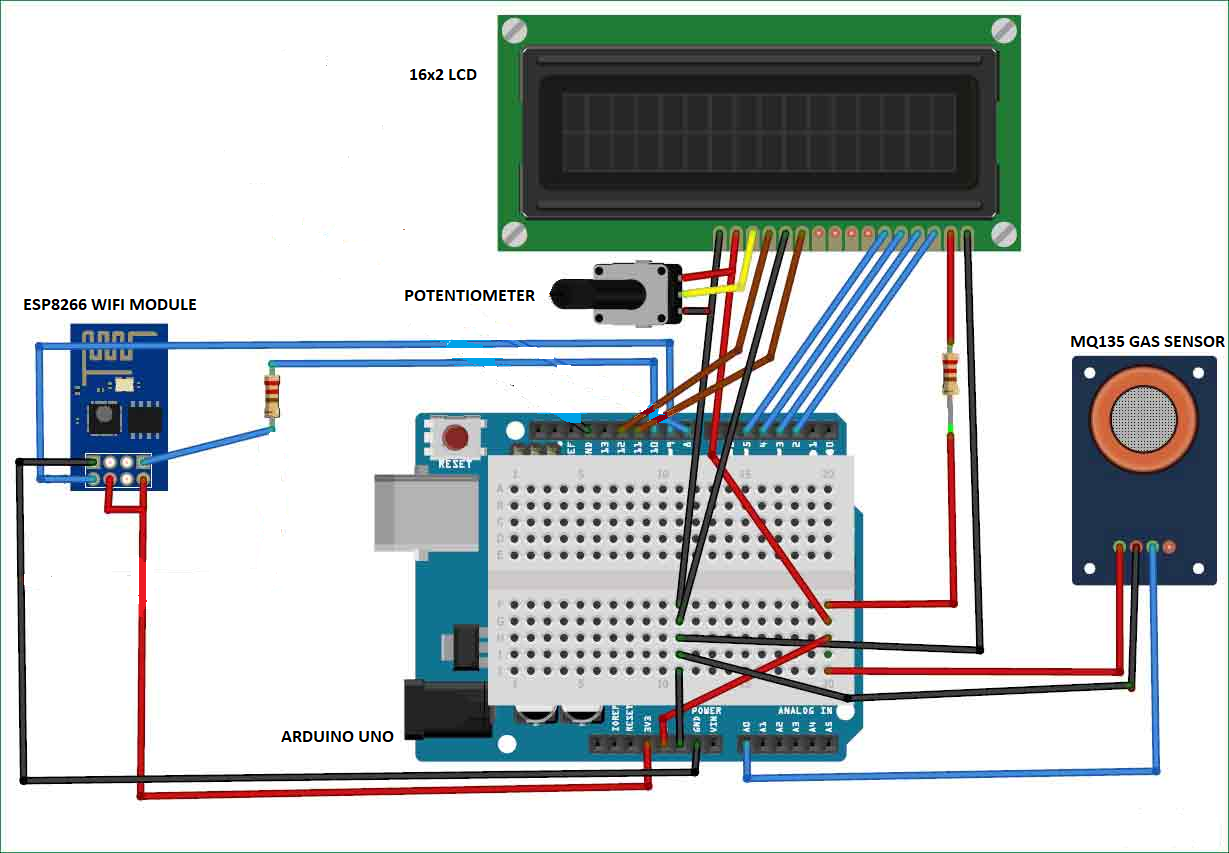
Connect pin 11 (D4) to pin 5 of Arduino.

Connect pin 12 (D5) to pin 4 of Arduino.

Connect pin 13 (D6) to pin 3 of Arduino.

Connect pin 14 (D7) to pin 2 of Arduino.

* Connect pin 15 to the VCC through the 220 ohm resistor. The resistor will be used to set the back light brightness. Larger values will make the back light much more darker.
* Connect pin 16 to the Ground

******

**HOW THE CIRCUIT WORKS :**

The device developed in this project can be installed near any Wi-Fi hotspot in a populated urban area. As the device is powered, the Arduino board loads the required libraries, flashes some initial messages on the LCD screen and start sensing data from the MQ-135 sensor. The sensitivity curve of the sensor for different combustible gases is already mentioned above. The sensor can be calibrated so that its analog output voltage is proportional to the concentration of polluting gases in PPM. The analog voltage sensed at the pin A0 of the Arduino is converted to a digital value by using the in-built ADC channel of the Arduino. The Arduino board has 10-bit ADC channels, so the digitized value ranges from 0 to 1023. The digitized value can be assumed proportional to the concentration of gases in PPM. The read value is first displayed on LCD screen and passed to the ESP8266 module wrapped in proper string through virtual serial function.  The Wi-Fi module is configured to connect with the ThingSpeak IOT platform. ThingSpeak is an IOT analytics platform service that allows to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by the IOT devices to ThingSpeak server. With the ability to execute MATLAB code in ThingSpeak one can perform online analysis and processing of the data as it comes in.

The Wi-Fi module can be connected with the ThingSpeak server by sending AT commands from the module. The module first test the AT startup by sending the following command -

AT

The command is passed by the controller to the Wi-Fi module using software serial function. In response to the command 'AT', the platform must respond with 'OK' if the  cloud service is running.

AT + CWMODE = 3

By setting the parameter in CWMODE to 3, the Wi-Fi connection is configured to SoftAP as well as station mode. This AT command can in fact take three parameters as follow -

1 - set Wi-Fi connection to station mode

2 - set Wi-Fi connection to SoftAP mode

3 - set Wi-Fi connection to SoftAP + station mode

In response to this command, the IOT platform must send back the string indication the Wi-Fi connection mode set. Now the AT command to reset the module is sent as follow -

AT + RST

In response to this command, the Wi-Fi module must restart and send back a response of 'OK'. After resetting the module, AT command to setup multiple connections is  enabled by sending the following command -

AT + CIPMUX=1

 This AT command can take two parameters - 0 for setting single connection and 1 for setting multiple connections. Next, the command to connect with the Access Point (AP) is passed which takes two parameters where first parameter is the SSID of the registered cloud service on ThingSpeak and the other parameter is the password to login the cloud service.

AT+CWJAP=\"EngineersGarage\",\"egP@$$w0rd?\

Now, the AT command to get local IP address is passed as follow -

The TCP IP connection is established by sending the following AT command -

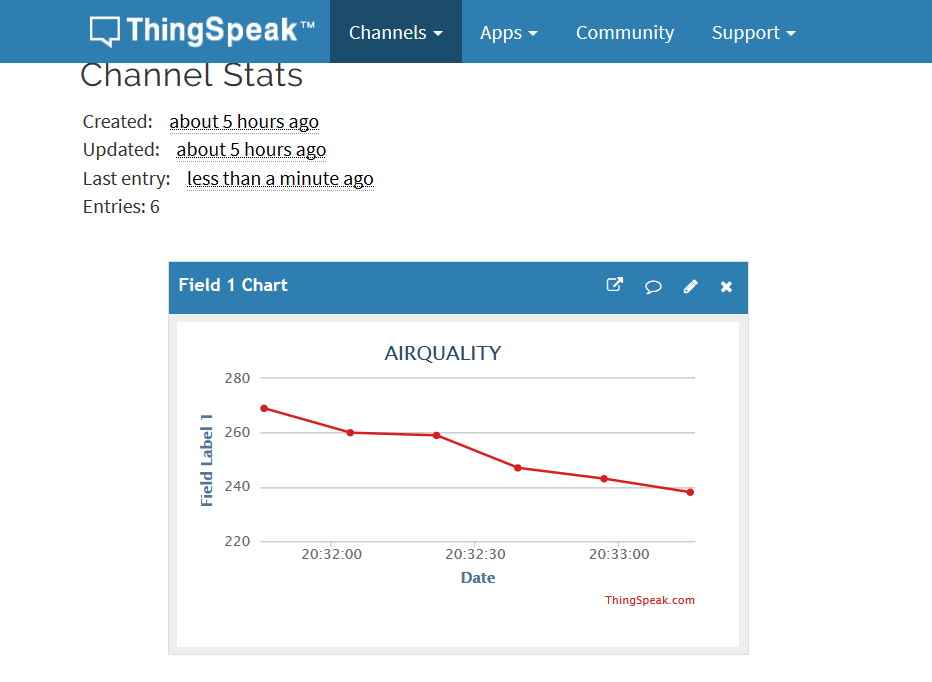
AT + CIPSTART = 4, "TCP", "184.106.153.149", 80

The AT + CIPSTART command can be used to establish a TCP connection, register an UDP port or establish an SSL connection. In the above command, it is used to establish a TCP IP connection. For establishing a TCP-IP connection, the command takes four parameters where first parameter is link ID which can be a number between 0 to 4, second parameter is connection type which can be TCP or UDP, third parameter is remote IP address or IP address of the cloud service to connect with and last parameter is detection time interval for checking if the connection is live. If the last parameter is set to 0, the TCP keep-alive feature is disabled otherwise a time interval in seconds range from 1 to 7200 can be passed as parameter. In response to this command, the server must respond with 'OK' if connection is successfully established otherwise it should respond with message 'ERROR'.

Now when the connection with the server is successfully established and the controller has read the sensor value, it can send the data to the cloud using the following command -

AT + CIPSEND = 4

 This command takes four parameters, where first parameter is the link ID which can be a number between 0 to 4, second parameter is data length which can be maximum 2048 bytes long, third parameter is remote IP in case of an UDP connection and remote port number in case of UDP connection. The third and fourth parameter are optional and used only in case of UDP connection with the server. Since, the TCP IP connection is established, these parameters are not used. The command is followed by a string containing the URL having the field names and values passed through the HTTP GET method. In this project, a string containing the URL having API Key and the sensor value as the field and value is passed. The passed field and its value are logged on the cloud server.  It is important to pass the API key in this URL as one of the field-value in order to connect with the registered cloud service. The Air quality measured by sensor can now be monitored and recorded through the thingspeak IOT plat form through the Wi-Fi module.  The recorded data is shown at the ThingSpeak platform as follow :



**ThingSpeak:**

### Setup ThingSpeak

ThingSpeak requires a user account and a channel. A channel is where you send data and where ThingSpeak stores data. Each channel has up to 8 data fields, location fields, and a status field. You can send data every 15 seconds to ThingSpeak, but most applications work well every minute.

* Sign up for new User Account – <https://thingspeak.com/users/sign_up>
* Create a new Channel by selecting Channels, My Channels, and then New Channel
* Note the Write API Key and Channel ID

Full REST Interface API information for ThingSpeak is available in the documentation .

### Install ThingSpeak Communication Library for Arduino

In the Arduino IDE, choose Sketch/Include Library/Manage Libraries. Click the ThingSpeak Library from the list, and click the Install button.

### Setup Arduino Sketch

We have provided a few Arduino sketch examples with the ThingSpeak library. They are designed to work right away with no changes. To make the examples work with your ThingSpeak channel, you will need to configure the myChannelNumber and myWriteAPIKey variables.

### Send an Analog Voltage to ThingSpeak

The Arduino sketch reads an analog voltage from pin 0, and writes it to a channel on ThingSpeak every 20 seconds. Load the example in the Arduino IDE. Make sure to select the correct Arduino board and COM port. Then, upload the code to your Arduino.

**REFERENCES :**

[**https://circuitdigest.com/microcontroller-projects/iot-air-pollution-monitoring-using-arduino**](https://circuitdigest.com/microcontroller-projects/iot-air-pollution-monitoring-using-arduino)

**https://www.engineersgarage.com/contribution/arduino-based-air-pollution-control**