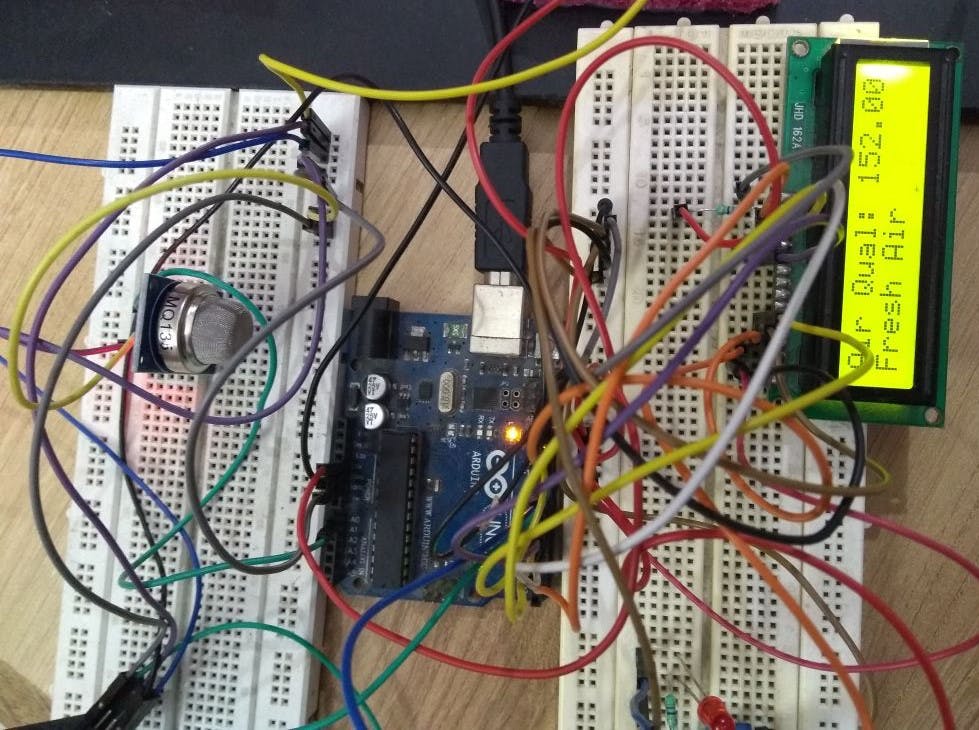
****

**Department of Computer Science and Engineering**

**A Report on**

**AIR QUALITY MONITORING SYSTEM**



**SUBMITTED BY: GUIDED BY:**

JEET TEWATIA DR. SACHIN SHARMA

CSE-CC Associate Professor

5TH SEM

2011716

OBJECTIVE:

The main objective of this project is to develop a IoT based Monitoring System which can actively monitor the quality of air by measuring the ppm of harmful gases present in the surroundings .

The data collected from sensors has to be uploaded to the cloud for analysis.

**Introduction**

Air pollution has become a common phenomenon everywhere. Specially in the urban areas, air pollution is a real-life problem. In the urban areas, the increased number of petrol and diesel vehicles and the presence of industrial areas at the outskirts of the major cities are the main causes of air pollution. The problem is seriously intensified in the metropolitan cities. The governments all around the world are taking every measure in their capacity. Many European countries have aimed to replace petrol and diesel vehicles with the electric vehicles by 2030. Even India has aimed to do so by 2025.

The main aim of this project is to develop a device which can monitor PPM in air in real time, tell the quality of air and log data to a remote server(ThingSpeak).

The air monitoring device developed in this project is based on Arduino Uno. The Arduino board connects with ThingSpeak platform using ESP8266 Wi-Fi module. The sensor used for monitoring the air pollution is MQ-135 gas sensor. The sensor data is also displayed on a character LCD.

**Component Required**

* Arduino Uno
* 16x2 LCD
* MQ135 gas sensor
* ESP8266 module
* Breadboard
* Connecting wires

**Hardware**

**1.Arduino Uno**

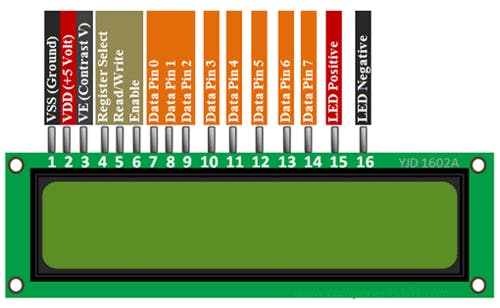
Arduino Uno is one of the most popular prototyping boards. It is small in size and packed with rich features. The board comes with built-in Arduino boot loader. It is an Atmega 328 based controller board which has 14 GPIO pins, 6 PWM pins, 6 Analog inputs and on board UART, SPI and TWI interfaces. In this IOT device, 9 pins of the board are utilized. There are six pins used to interface the character LCD. There are two pins utilized to interface the ESP8266 Wi-Fi Module and an analog input pin is used to connect the MQ-135 sensor.



Arduino Uno

**2. 16X2 Character LCD**

The 16X2 LCD display is used to monitor the sensor values read by the Arduino board from MQ-135. It is interfaced with the Arduino Uno by connecting its data pins D4 to D7 with pins 6 down to 3 of the controller respectively. The RS and E pins of the LCD are connected to pins 13 and 12 of the controller respectively. The RW pin of the LCD module is connected to the ground.



16x2 LCD

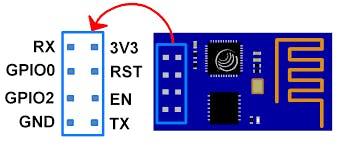
**3. ESP8266 Wi-Fi Module**

The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application networking functions from another application Each ESP8266 module comes pre-programmed with an AT command.

The ESP8266 supports APSD for VoIP applications and Bluetooth co-existence interfaces, it contains a self-calibrated RF allowing it to work under all operating conditions, and requires no external RF parts.

**Features**

* 802.11 b/g/n
* Wi-Fi Direct (P2P), soft-AP
* Integrated TCP/IP protocol stack
* Integrated TR switch, balun, LNA, power amplifier and matching network
* Integrated PLLs, regulators, DCXO and power management units
* +19.5dBm output power in 802.11b mode
* Power down leakage current of <10uA
* 1MB Flash Memory
* Integrated low power 32-bit CPU could be used as application processor
* SDIO 1.1 / 2.0, SPI, UART
* STBC, 1×1 MIMO, 2×1 MIMO
* A-MPDU & A-MSDU aggregation & 0.4ms guard interval
* Wake up and transmit packets in < 2ms
* Standby power consumption of < 1.0mW (DTIM3)



For connecting ESP8266 Module with Arduino Uno, you need 3.3 voltage regulator because Arduino is not capable of providing 3.3 v to ESP8266.

**4. MQ-135 Gas sensor**

The MQ-135 gas sensor senses the gases like ammonia nitrogen, oxygen, alcohols, aromatic compounds, sulfide and smoke. The operating voltage of this gas sensor is from 2.5V to 5.0V. MQ-135 gas sensor can be implementation to detect the smoke, benzene, steam and other harmful gases.



1 / 2 • MQ-135 Sensor

**Working Principle And Circuit Diagram**

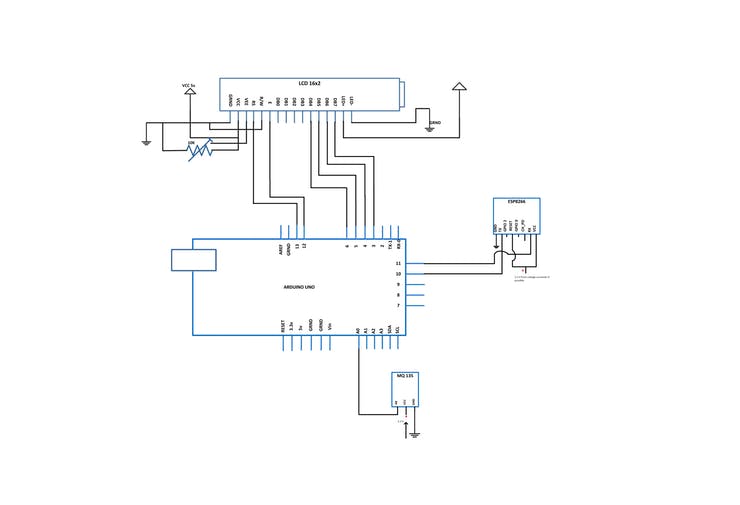
The MQ-135 alcohol sensor consists of a tin dioxide (SnO2), a perspective layer inside Aluminium Oxide micro tubes (measuring electrodes) and a heating element inside a tubular casing. The end face of the sensor is enclosed by a stainless steel net and the back side holds the connection terminals. Ethyl alcohol present in the breath is oxidized into acetic acid passing through the heat element. With the ethyl alcohol cascade on the tin dioxide sensing layer, the resistance decreases. By using the external load resistance the resistance variation is converted into a suitable voltage variation.

It has lower conductivity compare to clean air and due to air pollution the conductivity is increases. The air quality sensor detects ammonia, nitrogen oxide, smoke, CO2 and other harmful gases. The air quality sensor has a small potentiometer that permits the adjustment of the load resistance of the sensor circuit.

The air quality sensor is a signal output indicator instruction. It has two outputs: analog output and TTL output. The TTL output is low signal light which can be accessed through the IO ports on the microcontroller. The analog output is an concentration, i.e. increasing voltage is directly proportional to increasing concentration. The resistance of the sensor decreases as the concentration of the target gas is increased in PPM while for clean air its resistance remains constant.

The VCC and Ground terminals of the sensor are connected to the common VCC and Ground. The Analog Output pin of the sensor is connected to the A0 pin of the Arduino. The analog output voltage from the sensor can be assumed directly proportional to the concentration of CO2 gas in PPM under standard conditions. The analog voltage is sensed from the sensor and converted to a digital value in range from 0 to 1023 by the inbuilt ADC channel of the controller. The digitized value is hence equal to the gas concentration in PPM.

**Circuit Diagram**



Circuit diagram

**Connections**

**Arduino ==> LCD**

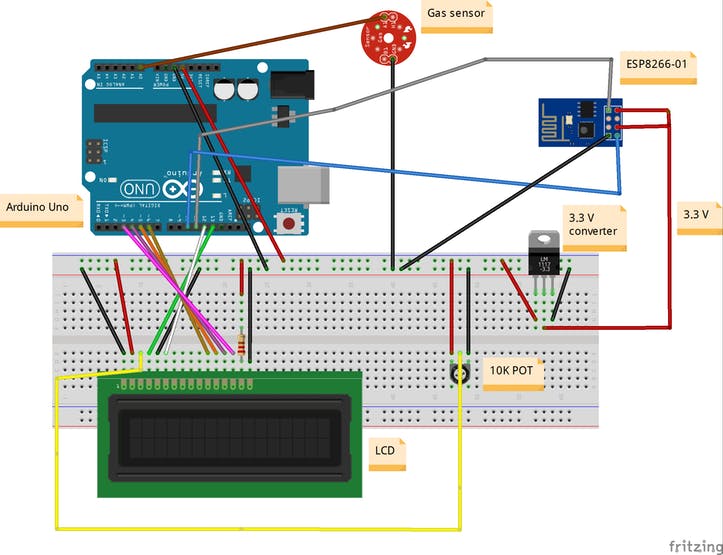
* GND ==> GND
* 5 V ==> Vcc
* D13 ==> RS
* GND ==> R/W
* D12 ==> Enable
* D6 ==> DB4
* D5 ==> DB5
* D4 ==> DB6
* D3 ==> DB7
* 5V ==> LED+
* GND ==> LED-

**Arduino ==> Gas sensor**

* GND ==> GND
* 3.3 V ==> Vcc
* Analog0 ==> A0

**Arduino ==> ESP8266**

* D10 ==> Tx
* D11 ==> Rx



## Code

#include <PubSubClient.h>

#include <WiFiEspClient.h>

#include <WiFiEspServer.h>

#include <WiFiEsp.h>

#include <WiFiEspUdp.h>

#include <ArduinoJson.h>

#include <SoftwareSerial.h>

#include <LiquidCrystal.h>

LiquidCrystal lcd(13, 12, 6, 5, 4, 3); // LCD connections

#define ORG "tryx3h"

#define DEVICE\_TYPE "Arduino"

#define DEVICE\_ID "air"

#define TOKEN "+lB96MbP8CkUq2Hauf"

#define WIFI\_AP "AK"

#define WIFI\_PASSWORD "ak123987"

WiFiEspClient espClient;

float t=0;

char data = 0;

SoftwareSerial soft(10, 11); // 10-tx,11-rx

int status = WL\_IDLE\_STATUS;

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

const char publishTopic[] = "iot-2/evt/status/fmt/json";

const char responseTopic[] = "iotdm-1/response";

const char manageTopic[] = "iotdevice-1/mgmt/manage";

const char updateTopic[] = "iotdm-1/device/update";

const char rebootTopic[] = "iotdm-1/mgmt/initiate/device/reboot";

void callback(char\* publishTopic, char\* payload, unsigned int payloadLength);

PubSubClient client(server, 1883, callback, espClient);

int publishInterval = 30000; // 30 seconds

long lastPublishMillis;

int period = 5000;

unsigned long time\_now = 0;

void setup() {

Serial.begin(115200);

InitWiFi();

Serial.print(WiFi.localIP());

if (!!!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {

Serial.print(".");

}

lcd.begin(16, 2); // to intialize LCD

lcd.setCursor(0,0);

lcd.print(" Welcome");

lcd.setCursor(0,1);

lcd.print(" To ");

delay(3000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print(" AIR");

lcd.setCursor(0,1);

lcd.print("QUALITY MONITOR");

delay(1000);

Serial.println();

}

}

void loop() {

t = analogRead(A0); // Read sensor value and stores in a variable t

Serial.print("Airquality = ");

Serial.println(t);

lcd.clear();

lcd.setCursor (0, 0);

lcd.print ("Air Qual: ");

lcd.print (t);

lcd.print (" PPM ");

lcd.setCursor (0,1);

if (t<=500)

{

lcd.print("Fresh Air");

Serial.print("Fresh Air ");

}

else if( t>=500 && t<=1000 )

{

lcd.print("Poor Air");

Serial.print("Poor Air");

}

else if (t>=1000 )

{

lcd.print("Very Poor");

Serial.print("Very Poor");

}

//lcd.scrollDisplayLeft();

delay(5000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print(" SENDING DATA");

lcd.setCursor(0,1);

lcd.print(" TO CLOUD");

String payload = "{\"d\":{ \"t\" :";

payload += t;

payload += "} }";

Serial.print("Sending payload: ");

Serial.println(payload);

while (!!!client.publish(publishTopic, (char \*)payload.c\_str())) {

Serial.println("Publish ok");

if (!!!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(1000);

}

Serial.println();

}

}

time\_now = millis();

Serial.println("5 sec Delay");

while(millis() < time\_now + period){

//wait approx. [period] ms

}

}

void InitWiFi()

{

// initialize serial for ESP module

soft.begin(112500);

// initialize ESP module

WiFi.init(&soft);

Serial.println("Connecting to AP …");

// attempt to connect to WiFi network

while ( status != WL\_CONNECTED) {

Serial.print("Attempting to connect to WPA SSID: ");

Serial.println(WIFI\_AP);

// Connect to WPA/WPA2 network

status = WiFi.begin(WIFI\_AP, WIFI\_PASSWORD);

delay(1000);

}

Serial.println("Connected to AP");

}

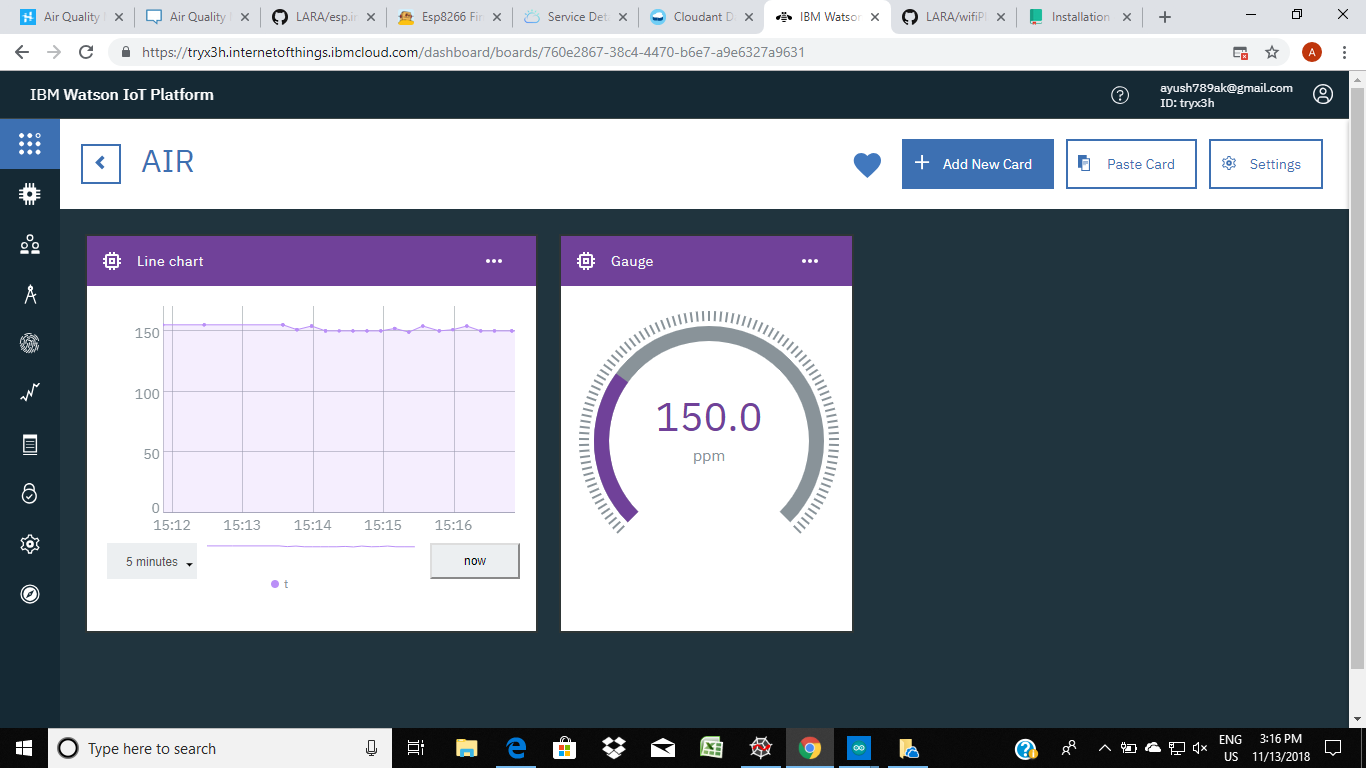
void callback(char\* publishTopic, char\* payload, unsigned int length) {

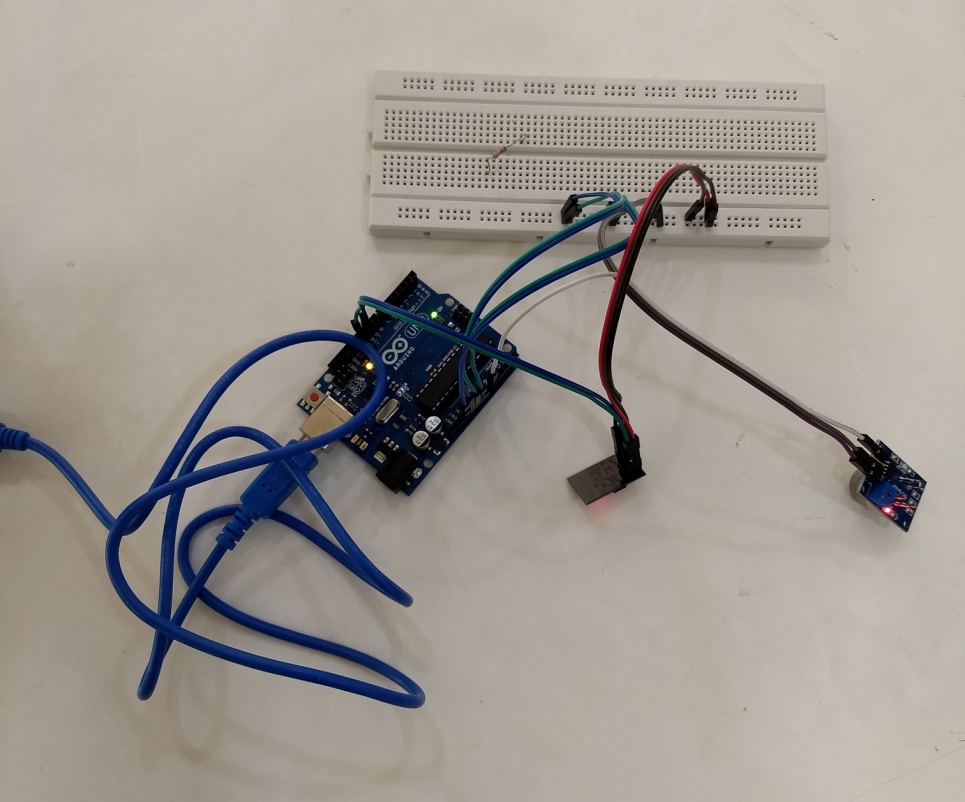
Serial.println("callback invoked");

}

**Project Real time Images:**







CONCLUSION

The model for air quality monitoring has been developed and the data has been sent to the cloud for further analysis i.e. the objective has been fulfilled.