Honors Thesis

Measuring CO_2 Concentration using Arduino and MQ-135 Sensor

Spring 2021

Mona Samsam

Table of Contents

Introduction

The Unit Operations Lab would like to develop a new experiment for students to understand reactors and utilize chemical reactions. The experiment apparatus would be a packed bed reactor with chemicals producing carbon dioxide. This project looks into designing and testing a CO_2 sensor using an Arduino UNO. An MQ-135 sensor is tested under different conditions in order to obtain detection limits and measure trends. The literature shows that there is around 400 PPM of carbon dioxide in the atmosphere, the average indoor values range from 35-450 PPM and the maximum acceptable level is around 1000 PPM. The MQ-135 detected 426 PPM inside of a room and up to 1200 PPM of CO_2 as a match burned under it before it was extinguished. The sensor was able to detect trends of increasing and decreasing concentrations of CO_2 around it under different conditions. This report will further investigate the MQ-135 sensor, the Arduino setup, and the experiments performed.

Objective

- Set up and create a design to measure CO_2 for a potential packed bed reactor experiment.
- Test the CO_2 sensor detection range under different conditions
    - High: burning match
    - Medium: human breath
    - Low: ambient air

Design

MQ-135 Sensor

The MQ-135 sensor is one of many MQ sensors that is used to detect dangerous gases. MQ gas sensors are analog devices that are specifically designed to be compatible with Arduino. This specific sensor is designed to detect NH_3, NO_x, alcohol, benzene, smoke, CO_2 and other dangerous gases in the environment. The device contains a sensitive layer that detects gases based on conductivity. The sensor's conductivity gets higher as the gas concentration increases. The gas sensitivity depends on both the temperature and the humidity. The sensor contains a built-in heater that operates at 5 V to provide an appropriate environment for the sensor.

The sensor contains four pins: A_O, D_O, GND, and VCC. VCC is the pin connecting to the Arduino power source of 5 V and GND connects to the Arduino ground pin. A_O is the analog out pin that outputs 0-5V analog voltage based on the intensity of the gas. This pin is what makes the air quality information readable. D_O is the digital output pin that operates based on predefined levels of detection.

When the sensor is powered on with 5 V, a red power LED is going to remain on. The output green LED is linked to the digital output pin, so when gas is introduced both the LED and the digital pin will turn on and signal a high voltage. If gas is introduced and the output is not high, the sensor has a built-in potentiometer that will need to be adjusted. A potentiometer is a mechanical device that provides variable resistance by turning a shaft. Adjusting the potentiometer changes the voltage that is sent to the analog, which changes the detection threshold of the output. The potentiometer also serves as a variable resistor for the sensor. The resistor is necessary to adjust the sensors sensitivity and accuracy. The resistance can range from 10 KΩ to 47 KΩ. The higher the resistance, the more sensitive the sensor is, which allows it to detect low concentrations of gas. The MQ-135 sensor has a wide detecting range of concentration varying from 10-1000 ppm. It is extremely sensitive and has a quick response time for detection.

When using the sensor for the first time, it needs to be calibrated or in a preheating period for a minimum of 24 hours. The burn-in period is performed by leaving the sensor working in a room with clean air for a day or two. After that, the sensor needs to be preheated for 20 seconds before measuring concentration.

OLED Screen

An organic light-emitting diode (OLED) display with 128x64 pixels is used in this design. The screen emits very strong light, not needing any back light, and it consumes less power than other types of displays. The Monochrome SSD1306 model has 7 pins, and it communicates with the Arduino using SPI communication protocol. The pins on the OLED display are listed below.

Pin Number	Pin Name	Usage
1	GND/ Ground	Ground pin
2	VCC	Power pin (3.3-5V)
3	D0/ SCK	Clock pin
4	D1/ SDA	Data pin
5	RES/ Reset	Resets the module
6	DC/ A_O	Data command pin
7	CS	Chip Select- used when there is more than one module

Table 1- OLED Display Pins

Circuit Diagram

Figure 1 shows the circuit schematic to connect the MQ-135 sensor to the Arduino board.

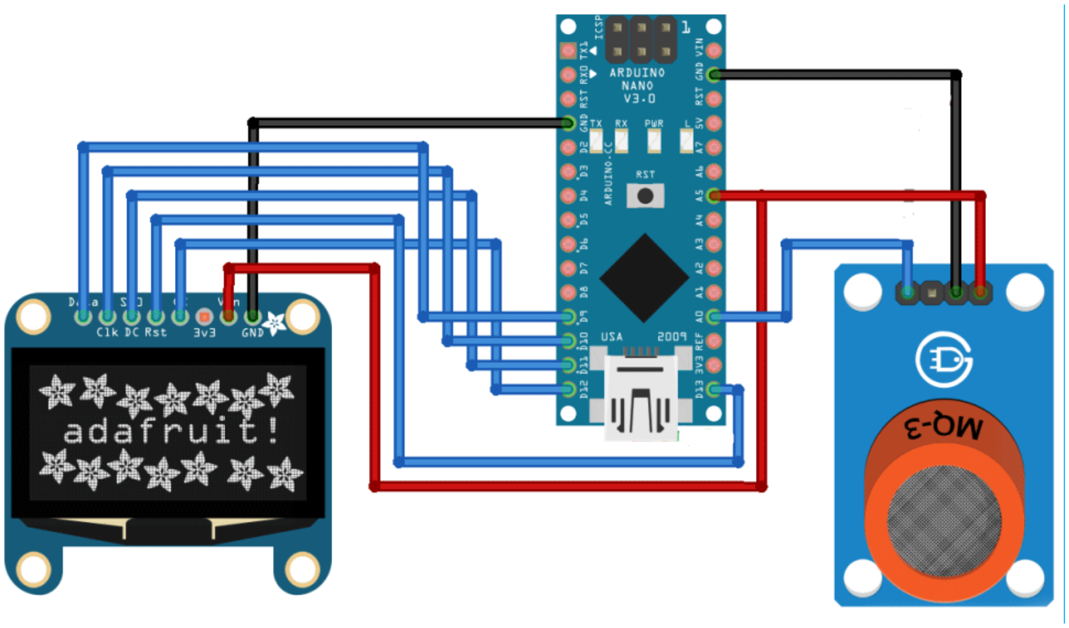

Figure 1- Circuit Schematic [1]

The circuit is very simple as this design only utilizes two units in order to achieve its purpose. Both units are connected to a 5V pin and GND. The MQ-135 sensor analog out pin is connected to A_O pin of the Arduino. The OLED Display is communicating with the Arduino using SPI communication as mentioned previously. The connections on the board are as follow:

OLED Pins	Arduino Pin
GND	Ground
VCC	5V
D0	10
D1	9
RES	13
DC	11
CS	12

Table 2- OLED Display Pins Connection to Arduino Pins

The setup should look something like this image:

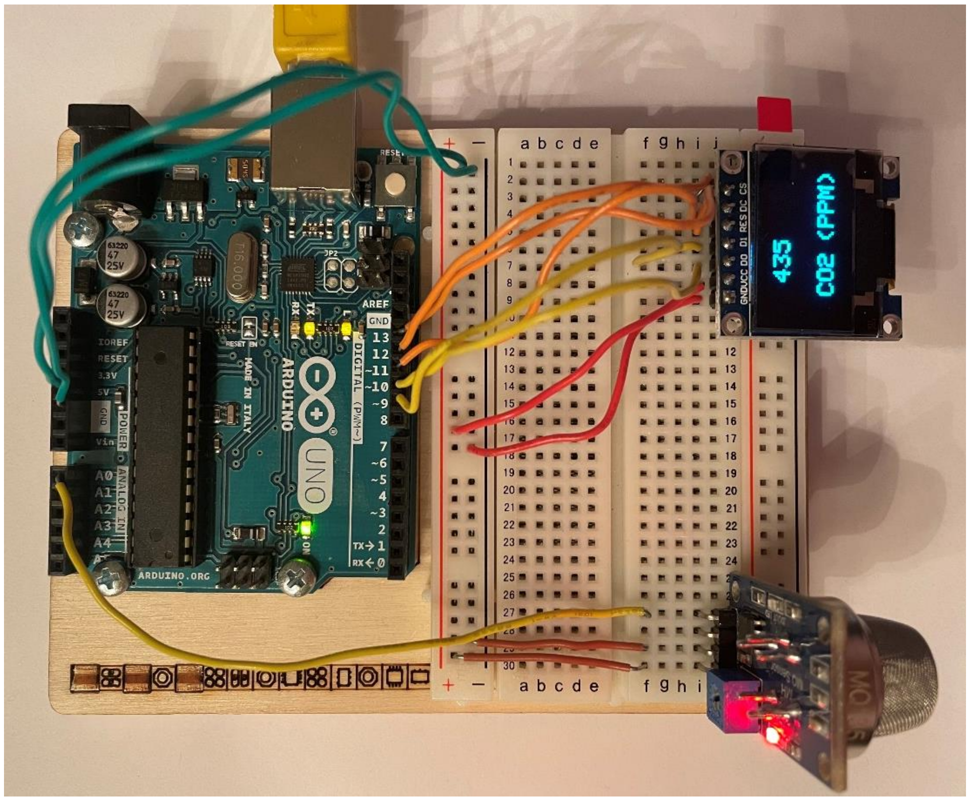

Figure 2- Arduino Board Setup

Code to Measure CO_2 Using Arduino MQ-135 Sensor

The final code to execute the experiment is in Appendix A. This code uses SPI.h, Adafruit_GFX.h, and Adafruit_SSD1306.h libraries. These libraries can be downloaded from the Library Manager in Arduino IDE or from online sources. They are necessary in the code to use the OLED Display to ensure SPI communication and size specifications are met. Both OLED and MQ-135 pins have to be defined in the code, so the Arduino knows where everything is connected. Next inside the setup () function, the serial monitor needs to be initialized at 9600 for debugging purposes. Also, the OLED display needs to be initialized using the begin () function. Note that it is important to specify the pin of the sensor as an input in the setup () function. In the loop () function, the analogRead () function is called to tell the analog pin of the Arduino to read the signal values. Co2lvl is the actual PPM being measured by the sensor. It is subtracted by 64 in this code because that is the minimum CO_2 value obtained from calibration. The map () function is used to specify bounds of measurement of the sensor. The next lines utilize the println () functions, which tell the Arduino to collect data and print in its Serial Monitor and Serial Plotter. Both the Serial Monitor and Serial Plotter are used to see and export data over time for further analyzing. Lastly, the following lines communicate with the display by specifying text size, location, and what to print. Everything under the loop () function is then cleared and looped every 0.2 seconds as specified in the delay function.

Methods

Sensor Calibration

As stated in a previous section, the MQ-135 sensor requires calibration. The preheat period is required to "burn-in" the sensor. The sensor needs the preheating period to set the load resistance. The load resistor on the sensor is a 1 KΩ resistor, but the resistance values change depending on the gas concentration. To obtain the correct resistance after the burn-in period, the MQ-135 manual recommends operating at 20 °C and 35% humidity and adding a 20 KΩ resistor.

For this project, calibration was completed in a cool room inside of a house where humidity tends to be around 40%. Calibration code in Appendix B can be run for 30 minutes and the most common value is the load resistance. This number can be specified in the sensor library as RZERO so the sensor is calibrated.

In addition to the resistance value, another option to calibrate requires finding the minimum analog CO2 value of the sensor. Once that value is found by running code in Appendix C, it can be used in the final code by subtracting the minimum value by the value the pin is actually reading, providing us an accurate measurement. It needs to be run for about an hour.

Procedure

After the sensor was calibrated and the code was running, the Arduino was put in an enclosed box to start measuring CO_2. The box is made out of Styrofoam and it has an opening from the top that acts as a window to allow air in. Initially, the CO_2 level in the room was 426 PPM. To obtain concentration readings, a balloon inflated with breath was slowly released inside of the box. The values were collected under Arduino's serial monitor and were plotted in Arduino's serial plotter. As more breath filled the box, the concentration of CO_2 increased. Since this was performed in an enclosed space, the carbon dioxide was trapped inside, and it took about 5 minutes to return back to the concentration of the room. The sensor was able to detect the values and show the increasing and decreasing concentrations as time went on as shown in Figures 3 and 4.

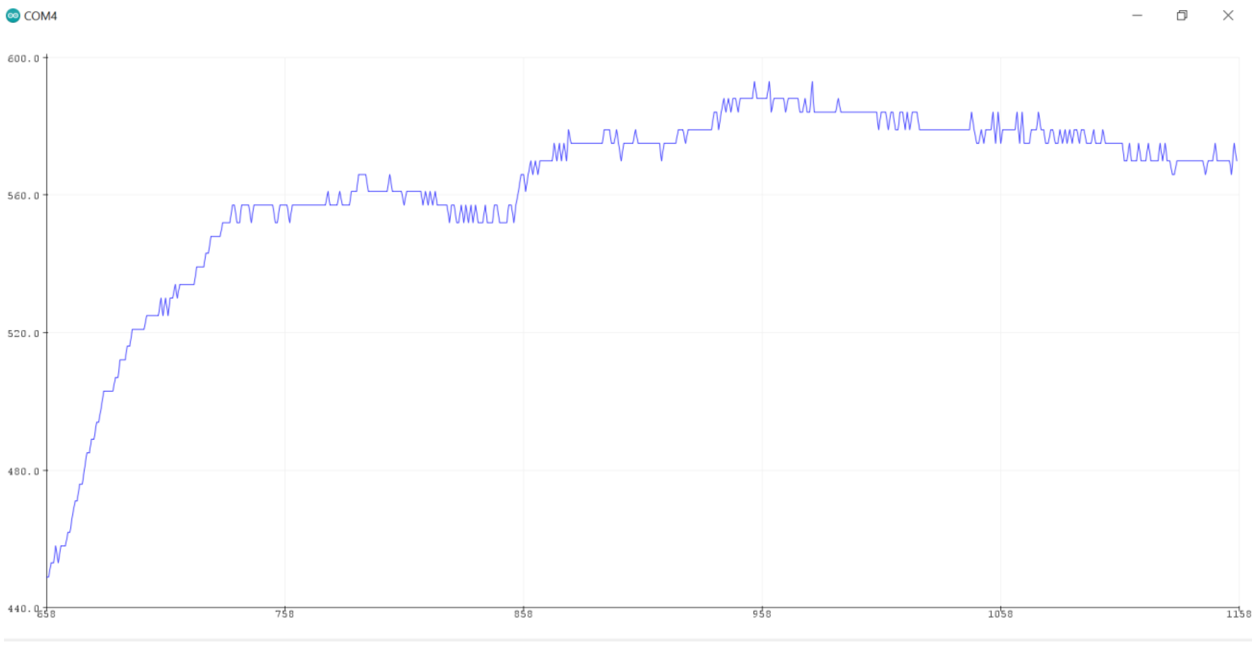

Figure 3- Concentration of CO_2 in Box when Balloon was Initially Released

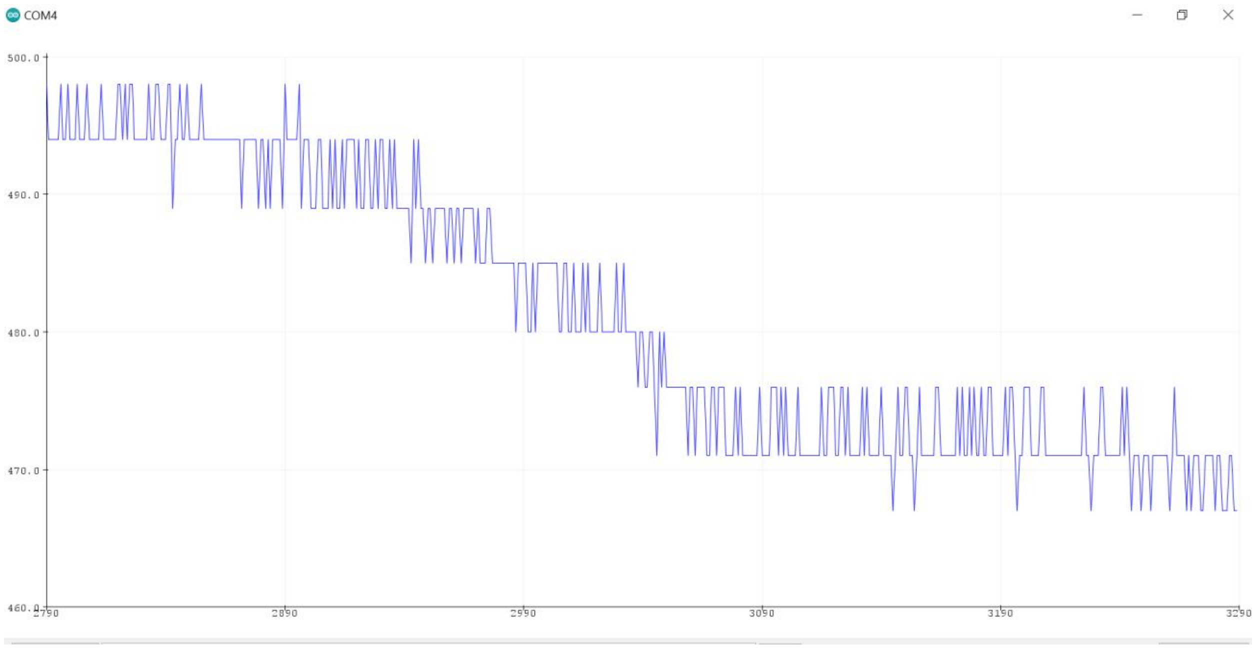

Figure 4- Concentration of CO_2 in Box After 3 Minutes

Lighting up a match was another method that was used to collect CO_2 concentration using the MQ-135 sensor. Lighting a match utilizes a combustion reaction which produces carbon dioxide and water. For safety reasons, this was conducted outside of the box and in an open area. Even with the sensor being exposed to outside sources of CO_2 such that in air, it was still able to capture rising CO_2 concentration from the reaction. Figure 5 shows the measured concentration of CO_2 vs time when the match was lit. The initial concentration in the room was around 430 PPM and when the match was placed close to the sensor, the concentration reached up to 1180 PPM before it went out. The amount of CO_2 around the sensor went back to the concentration of the room as soon as the match was extinguished due to it being in an open space, as compared to the longevity of CO_2 being in the box in the previous figures.

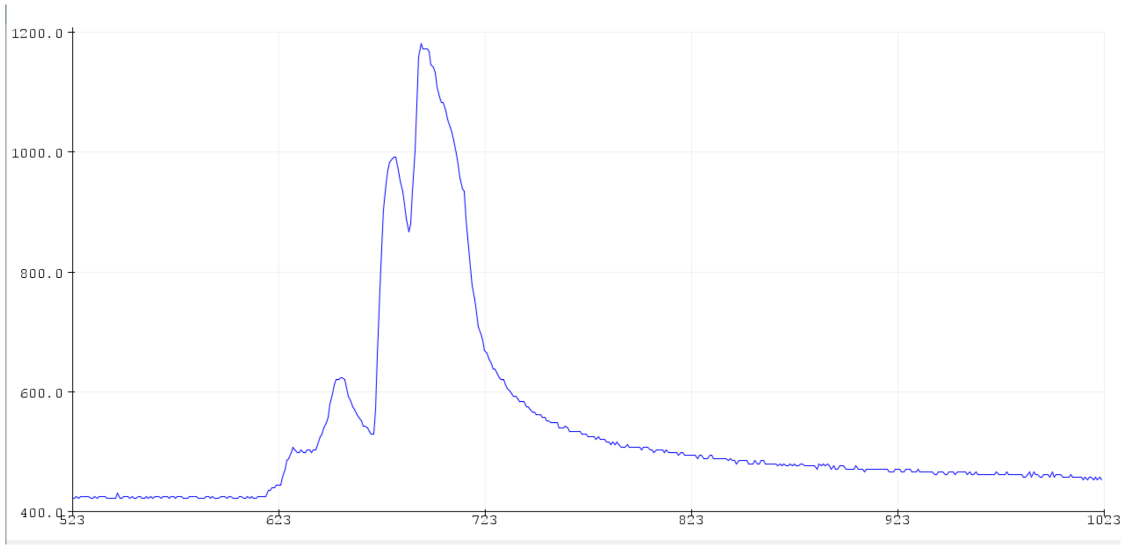

Figure 5- Concentration of CO_2 with Match

Troubleshooting

One of the main challenges of this experiment is calibrating the sensor. The sensor is extremely sensitive, and it needs to be done under the exact conditions that the MQ-135 sensor manual recommends. There are multiple ways to calibrate the sensor, and I found that finding the minimum CO_2 and subtracting it from the pin reading is easier and provides more accurate readings.

Another challenge came from the abundance of code and methods online to complete this experiment. Many codes called on a library for the sensor, but the library had to be edited based on the resistor, the calibration, and the outside CO_2 concentration, which introduced too many variables. Running a code with the library caused many issues due to many sources of variability. Specifying a code and writing it with clear calculations and values is way easier to debug and understand for users.

Reactor Proposal

Now that the CO_2 sensor is functioning, it can be utilized to design a Packed Bed Reactor with gas phase reaction to produce CO2. Designing a PBR based unit depends on many factors including the catalyst, shape, dimensions, and use. A possible chemical reaction, known as the Sabatier reaction, could be using methane and water to produce “Syngas”.

$$CH_4 + 2\ H_2O \rightarrow CO_2 + 4\ H_2 \qquad \Delta H_{298} = +165.47\ \text{kJ/mol}$$

For safety reasons, this reactor would be designed to be very small and contain very small concentration of gases. Both methane and carbon dioxide are major contributors to environmental issues and could also cause harm to people if exposed.

The specifics of the design could be other semesters’ worth of work, but as for the basics, this reaction needs elevated temperatures of around 300-400 °C and pressures around 30 bar with a nickel catalyst. There are many different catalysts such as aluminum oxide, silica, and others that could be more efficient.

The MQ-135 sensor can be installed inside the reactor or in an enclosed container by the exit valve of the reactor. This sensor would be compatible with the proposed reaction since it doesn’t have any sensitivity to the other chemicals present. The sensor will be able to pick up on trace amounts of CO_2 and users can see a trend and calculate the concentration per time.

Conclusion

The MQ-135 sensor is an effective device to measure CO_2 concentration in PPM. It is cost efficient, easy to debug, and user friendly for first time users. The Arduino interface makes it easy to install add-ons and explore more function with the build. The sensor measured accurate readings of carbon dioxide in air, human breath and a burning match compared to the literature values. The sensor was able to prove that as more breath was blown inside of a box, the higher the concentration it was able to detect. The sensor can be utilized in future projects to add a packed bed reactor producing carbon dioxide in the Unit Operations Lab.

References

1- https://circuitdigest.com/microcontroller-projects/interfacing-mq135-gas-sensor-with-arduino-to-measure-co2-levels-in-ppm
2- https://www.olimex.com/Products/Components/Sensors/Gas/SNS-MQ135/resources/SNS-MQ135.pdf
3- https://blog.anavi.technology/?p=56
4- https://components101.com/sensors/mq135-gas-sensor-for-air-quality
5- https://www.codrey.com/electronic-circuits/how-to-use-mq-135-gas-sensor/
6- https://www.youtube.com/watch?v=E9PSSxRO6Ik&t=231s
7- https://bmcchemeng.biomedcentral.com/articles/10.1186/s42480-019-0007-7
8- https://en.wikipedia.org/wiki/Sabatier_reaction

Appendix

A-Final Code

```
#include <SPI.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>


#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 64

#define sensor A0

#define OLED_MOSI   9

#define OLED_CLK   10

#define OLED_DC    11

#define OLED_CS    12

#define OLED_RESET 13


Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, OLED_MOSI,
OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);


int gas, co2lvl;


void setup()

{

  Serial.begin(9600);

  pinMode(sensor, INPUT);

  display.begin(SSD1306_SWITCHCAPVCC);

  display.clearDisplay();

  display.display();

}
```

```
void loop()
{
  gas = analogRead(sensor);
  co2lvl = gas - 64;
  co2lvl = map(co2lvl,0,1024,400,5000);
  Serial.print(co2lvl);
  Serial.print(  "ppm");
  Serial.println();
  Serial.println(co2lvl);

  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(18,43);
  display.println("CO2");
  display.setCursor(63,43);
  display.println("(PPM)");
  display.setTextSize(2);
  display.setCursor(28,5);
  display.println(co2lvl);
  display.display();
  display.clearDisplay();

  delay(200);
}
```

B- Resistor Calibration

```
#include "MQ135.h"
void setup () {
Serial.begin (9600);
}
void loop() {
MQ135 gasSensor = MQ135(A0);
float rzero = gasSensor.getRZero();
Serial.println (rzero);
delay(1000);
}
```

C- Minimum CO_2 Calibration

```
#define sensor = A0
Int gas, minval= 1000;
void setup()
{
  pinMode(sensor, INPUT);
  Serial.begin(9600);
}

void loop()
{
  gas = analogRead(sensor);
if (gas <= minval)
{
        minval =gas;
}
Serial.print(gas);
Serial.print("   ");
Serial.print (minval);
Serial.println();
delay(200);
```