



(<https://www.dfrobot.com/product-540.html>)

Introduction

The "Greenhouse Effect" is melting the icebergs every minute. By knowing the exact concentration of CO₂, we can do something to reduce the atmosphere's CO₂ level and to protect our earth. For that reason, DFRobot eningeer's designed a high quality CO₂ sensor. This is the first **CO₂ sensor** (<https://www.dfrobot.com/category-85.html>) on the opensource hardware market. The output voltage of the module falls as the concentration of the CO₂ increases. The potentiometer onboard is designed to set the threshold of voltage. Once the CO₂ concentration is high enough (voltage is lower than threshold), a digital signal (ON/OFF) will be released.

- It has MG-811 gas sensor onboard which is highly sensitive to CO₂ and less sensitive to alcohol and CO, low humidity & temperature dependency. All components have industrial quality which ensures stability and reproducibility.
- The onboard heating circuit brings the best temperature for sensor to function. 5V power input will be boosted to 6V for heating.
- This sensor has an onboard conditioning circuit for amplifying output signal.



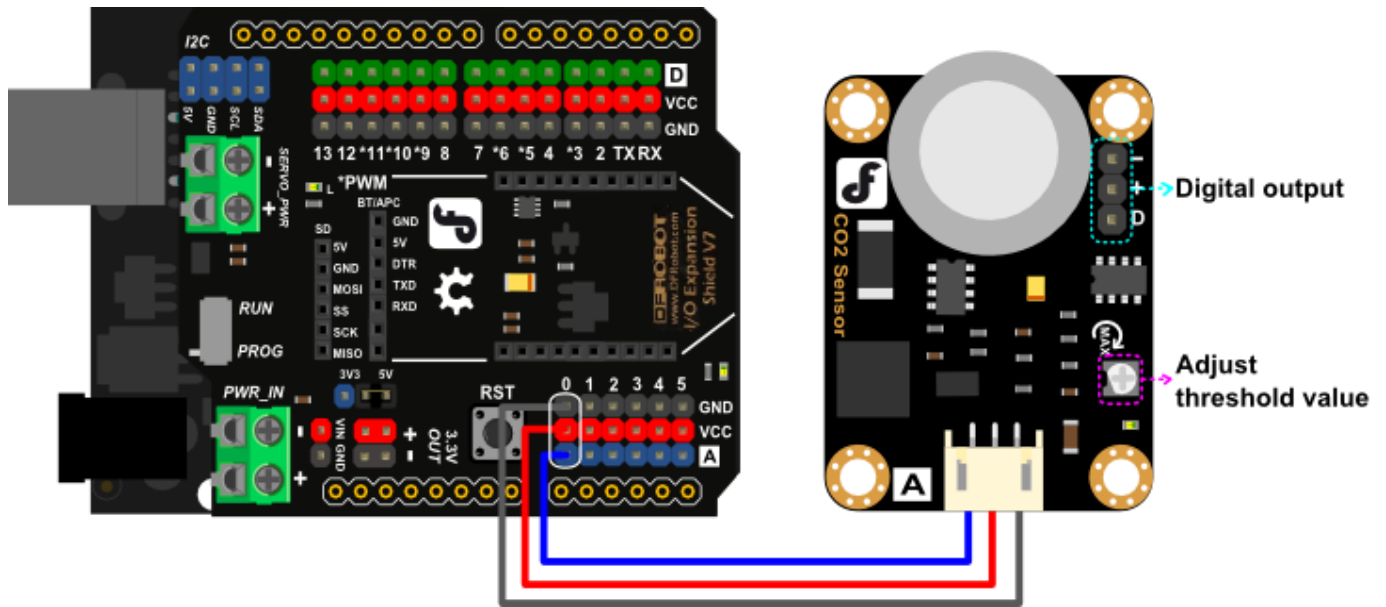
- External power supply (7~12V) is necessary to supply the microcontroller board when you using this CO₂ sensor module.
- This module is an electrochemical sensor, you need to calibrate it before actual measurement.

Specification

- Operating voltage:5V
- Interface: Analog (Gravity Compatible)

- One digital output
- High quality connector
- Immersion gold surface
- Onboard heating circuit
- Size:32x42mm (1.26x1.65")

Connecting Diagram



Tutorial

How to use this module? It is very easy. You need to set potentiometer onboard to the threshold value. Just make the red led turn off. With the CO2 concentration is enough high to make the sensor output voltage higher than threshold value, the led will be turned on. If you connect a buzzer to the module (right side), you will hear the alarm.

Calibration

This module is an electrochemistry sensor, you should calibrate it before actual measurement. You should provide stable power to this module, and the sensor will heat up while working. Please put this module into an area where the air is clean. After continuous working for about 48 hours, you can measure the output voltage of this module. Then modify the definition in the code with the voltage value (unit: V) divided by 8.5.

```
#define ZERO_POINT_VOLTAGE (voltage/8.5)
```

For example, the voltage you measured from the module is 2.4V, then $2.4/8.5=0.282$. So modify the definition as below:

```
#define ZERO_POINT_VOLTAGE (0.282)
```

After the modification, upload the sample code to your Arduino board.

Sample code

```

/*****Demo for MG-811 Gas Sensor Module V1.1*****/
Author: Tiequan Shao: tiequan.shao@sandboxelectronics.com
        Peng Wei:      peng.wei@sandboxelectronics.com

Lisence: Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0)

Note:   This piece of source code is supposed to be used as a demonstration ONLY. More
        sophisticated calibration is required for industrial field application.

                                           Sandbox Electronics    2012-05-31
*****/

/*****Hardware Related Macros*****/
#define      MG_PIN                (A0)      //define which analog input channel
#define      BOOL_PIN              (2)
#define      DC_GAIN                (8.5)    //define the DC gain of amplifier

/*****Software Related Macros*****/
#define      READ_SAMPLE_INTERVAL  (50)     //define how many samples you are going to read
#define      READ_SAMPLE_TIMES     (5)      //define the time interval(in milisecond)
                                           //normal operation

/*****Application Related Macros*****/
//These two values differ from sensor to sensor. user should determine this value.
#define      ZERO_POINT_VOLTAGE    (0.220) //define the output of the sensor in mV
#define      REACTION_VOLTAGE     (0.030) //define the voltage drop of the sensor

/*****Globals*****/
float      CO2Curve[3] = {2.602,ZERO_POINT_VOLTAGE,(REACTION_VOLTAGE/(2.602-3))};
                                           //two points are taken from the curve
                                           //with these two points, a line is formed
                                           //"approximately equivalent" to the curve
                                           //data format:{ x, y, slope}; point1:2.602,3
                                           //slope = ( reaction voltage ) / (log(2.602/3))

void setup()
{
    Serial.begin(9600);                //UART setup, baudrate = 9600bps
    pinMode(BOOL_PIN, INPUT);         //set pin to input
    digitalWrite(BOOL_PIN, HIGH);     //turn on pullup resistors

    Serial.print("MG-811 Demostration\n");
}

void loop()
{
    int percentage;
    float volts;

```

```

volts = MGRead(MG_PIN);
Serial.print( "SEN0159:" );
Serial.print(volts);
Serial.print( "V          " );

percentage = MGGetPercentage(volts,CO2Curve);
Serial.print("CO2:");
if (percentage == -1) {
    Serial.print( "<400" );
} else {
    Serial.print(percentage);
}

Serial.print( "ppm" );
Serial.print("\n");

if (digitalRead(BOOL_PIN) ){
    Serial.print( "=====BOOL is HIGH=====" );
} else {
    Serial.print( "=====BOOL is LOW=====" );
}

Serial.print("\n");

delay(500);
}

/***** MGRead *****/
Input:  mg_pin - analog channel
Output: output of SEN-000007
Remarks: This function reads the output of SEN-000007
*****/
float MGRead(int mg_pin)
{
    int i;
    float v=0;

    for (i=0;i<READ_SAMPLE_TIMES;i++) {
        v += analogRead(mg_pin);
        delay(READ_SAMPLE_INTERVAL);
    }
    v = (v/READ_SAMPLE_TIMES) *5/1024 ;
    return v;
}

/***** MQGetPercentage *****/
Input:  volts - SEN-000007 output measured in volts
        pcurve - pointer to the curve of the target gas
Output: ppm of the target gas
Remarks: By using the slope and a point of the line. The x(logarithmic value of ppm)
of the line could be derived if y(MG-811 output) is provided. As it is a
logarithmic coordinate, power of 10 is used to convert the result to non-logarithmic
value.
*****/

```

```
int MGGetPercentage(float volts, float *pcurve)
{
    if ((volts/DC_GAIN )>=ZERO_POINT_VOLTAGE) {
        return -1;
    } else {
        return pow(10, ((volts/DC_GAIN)-pcurve[1])/pcurve[2]+pcurve[0]);
    }
}
```

➔ Go Shopping CO2 Sensor (Arduino compatible) (SKU:SEN0159)

(<https://www.dfrobot.com/category-85.html>)

Category: DFRobot > [Sensors & Modules](https://www.dfrobot.com/category-156.html) > [Sensors](https://www.dfrobot.com/category-36.html) > [Air Sensors](https://www.dfrobot.com/category-85.html)

category: Product Manual (category__Product_Manual) category: SEN Series (category__SEN_Series) category: Module (category__Module) <!--Hidden Categories--!>

category: Diagram (category__Diagram) category: DFRobot (category__DFRobot)-->