SKU:SEN0465toSEN0476 (https://www.dfrobot.com/product-2510.html)

(https://www.dfrobot.com/product-2510.html)

Introduction

The gas sensor series is designed to detect various flammable, combustible, and toxic gases in the environment. With pre-calibrated sensors, it can measure the concentration of specific gases quickly and accurately. Supporting multiple output modes, including Analog, I2C, UART, and digital alarm signals, it offers flexibility in different applications. The probes in this series use electrochemical principles, providing strong stability and sensitivity with a lifespan of up to two years.

Using the simple and user-friendly Gravity interface, you can easily build various gas concentration detectors. This sensor series has widespread applications in safety production, industrial manufacturing, and environmental protection, making it the ideal choice for settings such as coal mines, chemical industries, chemical laboratories, and environmental management.

If you use the PCB for V1.0 version and work for a long time there will be data abnormalities, please contact technical support by email: techsupport@dfrobot.com (mailto:techsupport@dfrobot.com)

Precautions for use

- The white waterproof and breathable membrane of the sensor on the module is strictly forbidden to open, otherwise it is regarded as artificial damage.
- It is forbidden to plug or unplug the probe with power on.
- It is forbidden to directly solder the pins of the module, but the sockets of the pins can be soldered.

- The module should avoid contact with organic solvents (including silica gel and other adhesives), paints, pharmaceuticals, oils and high-concentration gases.
- The module must not be subjected to excessive shock or vibration.

- The module needs to be warmed up for more than 5 minutes when powered on for the first time. It is recommended to warm up for more than 24 hours if it has not been used for a long time.
- Do not apply this module to systems involving personal safety.
- Do not install the module in environment with strong air convection.
- Do not leave the module in high-concentration organic gas for a long time.
- The data returned by the serial port of the module is the real-time concentration value in the current environment. If there is no standard gas, please do not try the calibration command. This command will clear the calibrated data, and the data returned by the serial port will be inaccurate.
- To judge whether the module communication is normal, it is recommended to use a USB to TTL tool (communication level 3V) to observe and judge according to the communication protocol through the serial debugging assistant software.

Features

- Factory calibrated, accurate measurement
- High sensitivity, low power consumption
- Excellent stability and anti-interference
- Three output modes: I2C, UART and analog
- Long service life(2 years)
- Compatible with 3.3~5.5V main controllers
- 32 modifiable I2C addresses
- Reverse connection protection
- Temperature compensation
- Threshold alarm

Specification

- Detection Gas: CO, O2, NH3, H2S, NO2, HCL, H2, PH3, SO2, O3, CL2, HF(Need to change different probe)
- Working Voltage: 3.3 ~ 5.5V DC
- Working Current: <5mA
- Output Signal: I2C, UART output (0~3V), analog voltage (see the characteristic parameters of specific probe)
- Detection error: ±10% of output value or ±5% of full scale (whichever is greater)

- Working Temperature: -20 ~ 50℃
- Working Humidity: 15 ~ 90%RH (non-condensing)
- Storage Temperature: -20 ~ 50℃
- Storage Humidity: 15 ~ 90%RH (non-condensing)
- Lifespan: >2 years (in the air)
- Adapter Plate Size: 37×32mm

Characteristic Parameters

SKU	SEN0465	SEN0466	SEN0467	SEN0468	SEN0469
Type	O2	CO	H2S	Cl2	NH3
Detection Range	(0-25)%Vol	(0-1000)ppm	(0-100)ppm	(0-20)ppm	(0-100)pp
Resolution	0.1%Vol	1ppm	1ppm	0.1ppm	1ppm
V0 Voltage output range	(1.5-0)V	(0.6-3)V	(0.6-3)V	(2-0)V	(0.6-3)V
Vout1	1.0V@10%vol	0.9V@200ppm	1.5V@50ppm	1.3V@10ppm	1.4V@50
Response Time (T90)	≤15S	≤30S	≤30S	≤60S	≤150S

SKU	SEN0471	SEN0472	SEN0473	SEN0474	SEN0475
Type	NO2	O3	H2	HCL	HF
Detection Range	(0-20)ppm	(0-10)ppm	(0-1000)ppm	(0-10)ppm	(0-10)ppm
Resolution	0.1ppm	0.1ppm	1ppm	0.1ppm	0.1ppm
V0 Voltage output range	(2-0)V	(2-0.7)V	(0.6-3)V	(2-0.7)V	(2-0.5)V
Vout1	1.2V@10ppm	1.3V@5ppm	1.3V@500ppm	1.4V@5ppm	1.3V@5ppm
Response					

Time (T90)	≤30S	≤120S	≤120S	≤60S	≤60S

Explanation of VO use:

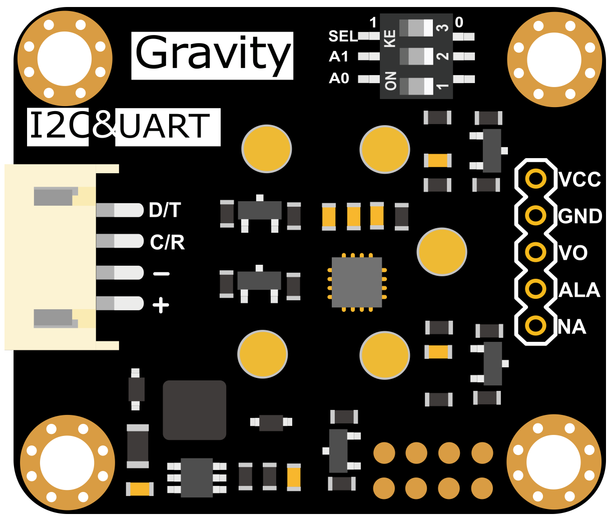
VO: It means original voltage (linear) after amplifying circuit, rather than concentration value of current environment.

Calculation method: concentration in the current environment N= 200/(Vout1-Vout0)*(Voutx-Vout0)

Where Vout1 corresponds to Vout1 in the table and Vout0 corresponds to the voltage value of the gas at 0 ppm in the table. Take CO as an example: zero point voltage Vout0 = 0.6V, Vout1 = 0.9V, the current voltage of VO Voutx = 1.2V, then the current concentration in the environment N = 400ppm

Note: The analog output is the original uncalibrated voltage of the probe, the UART/I2C data is factory calibrated, if there is no special requirement, it is recommended to use the calibrated UART/I2C data.

Board Overview



(https://dfimg.dfrobot.com/nobody/wiki/617e7b52992ac13109305c38bd4fbd7c.png)
Smart Gas Sensor Terminal

Label	Name	Function description
1	D/T	I2C data line SDA / UART data transmitting-TX
2	C/R	I2C clock line SCL / UART data receiving-RX
3	-	GND -
4	+	Power supply + (3.3-5V compatible)

Label	Name	Function description
1	VCC	Positive power supply (3.3-5V compatible)

Label	Name	Function description
2	GND	GND negative power supply
3	V0	The raw voltage output of the gas probe. You can develop your own conversion algorithm based on the original output.
4	ALA	Threshold alarm function, the threshold can be set through API, when exceeding this value, the pin will output high level.
5	NA	Reserve custom pins, you can contact us for custom functions.

Tutorial for Arduino

Download the program to UNO and open the serial monitor to check the gas concentration.

Note:

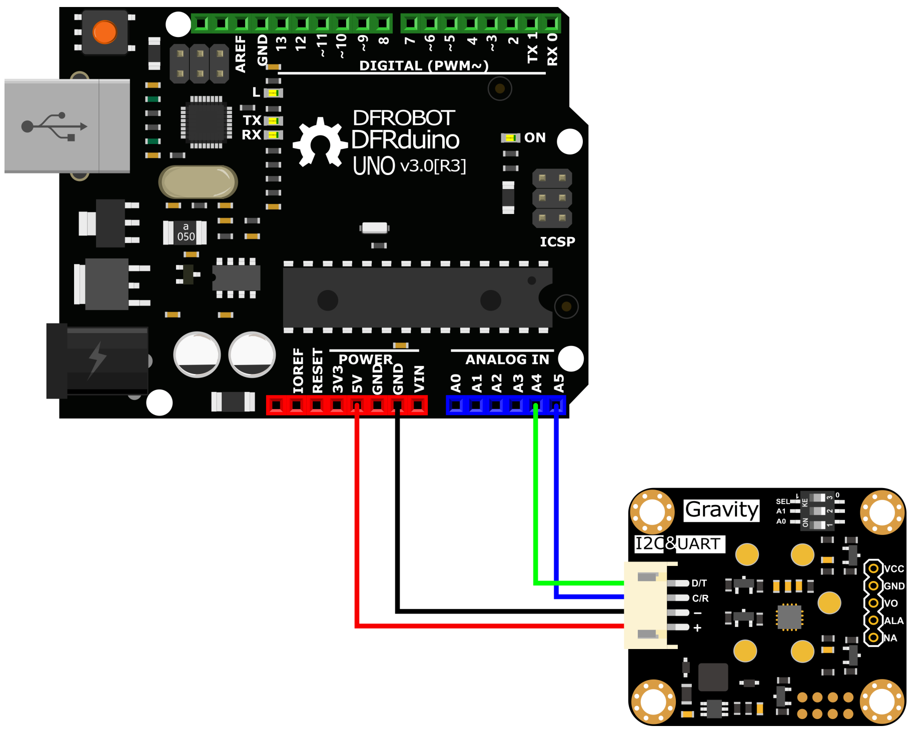
- **The initial power-on requires more than 5 minutes of preheating. It is recommended to preheat more than 24 hours if it has not been used for a long time.**
- **After switching the communication mode or changing the I2C address, the system needs to be powered off and on again.**

Requirements

- **Hardware**
  - DFRuino UNO R3 (https://www.dfrobot.com/product-838.html) x1
  - DFR0784 Smart Gas Sensor Terminal x1
  - Gas probe x1
  - Jumper wires
- **Software**
  - Arduino IDE (https://www.arduino.cc/en/Main/Software)
  - Download and install the **DFRobot_GasSensor Library** (https://github.com/DFRobot/DFRobot_MultiGasSensor) (About how to install the library? (https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0))

Acquire data in passive mode

Connection



(https://dfimg.dfrobot.com/nobody/wiki/5b8919ea31cafb8d2ddbc0d0ee1627d6.png)

Sample code

- Connect the module to the Arduino according to the connection diagram above. Of course, you can also use it with Gravity I/O Expansion Board () to build the project prototype more conveniently and quickly.
- Set the DIP switch SEL on the sensor to 0, and use I2C communication by default.
- The default I2C address is 0x74. If you need to modify the I2C address,You can configure the hardware I2C address through the DIP switch on the module, or run the code to modify the address group to modify the address. The corresponding relationship between the DIP switch and the I2C address parameter is as follows:
  - ADDRESS_0: 0x74, A0=0, A1=0
  - ADDRESS_1: 0x75, A0=1, A1=0
  - ADDRESS_2: 0x76, A0=0, A1=1
  - ADDRESS_3: 0x77, A0=1, A1=1
- Download and install the **DFRobot_GasSensor Library** (https://github.com/DFRobot/DFRobot_MultiGasSensor) (About how to install the library? (https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0))
- Open Arduino IDE and upload the following code to Arduino UNO.
- Open the serial port monitor of Arduino IDE, adjust the baud rate to 115200, and observe the serial port print result.

Statement

- In this routine, the controller needs to request data from the sensor every time, and then the sensor returns the data.

- Default use I2C communication, mask `#define I2C_COMMUNICATION` in the code, and set the dip switch SEL to 1, the sensor is connected to the corresponding port defined by the controller, if use UNO, the blue line is connected to D3 and the green line is connected to D2, if use ESP32, the blue line is connected to IO17 and the green line is connected to IO16. After re-uploading the code, the whole system will be re-powered and will switch to UART communication.
- Turn off temperature compensation by default, modify the code `gas.setTempCompensation(gas.ON);` , turn on temperature compensation after re-uploading the code

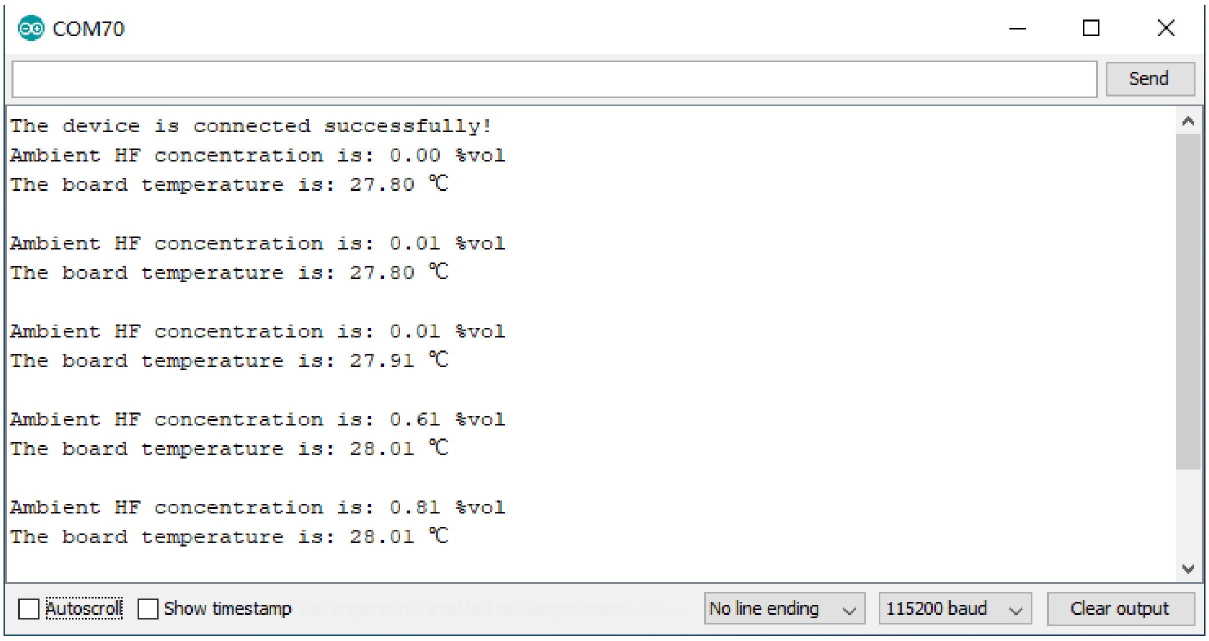
```
/*!
  * @file  initiativereport.ino
  * @brief The sensor actively reports all data
  * @n Experimental method: Connect the sensor communication pin to the main control, ther
  * @n Communication mode selection, dial switch SEL:0: IIC, 1: UART
@n I2C address selection, the default I2C address is 0x74, A1 and A0 are combined into 4 t
                | A1 | A0 |
                | 0  | 0  |    0x74
                | 0  | 1  |    0x75
                | 1  | 0  |    0x76
                | 1  | 1  |    0x77   default i2c address
  * @n Experimental phenomenon: Print all data via serial port
*/
#include "DFRobot_MultiGasSensor.h"

//Enabled by default, use IIC communication at this time. Use UART communication when disa
#define I2C_COMMUNICATION

#ifdef  I2C_COMMUNICATION
#define I2C_ADDRESS    0x74
  DFRobot_GAS_I2C gas(&Wire ,I2C_ADDRESS);
#else
#if (!defined ARDUINO_ESP32_DEV) && (!defined __SAMD21G18A__)
/**
  UNO:pin_2-----RX
      pin_3-----TX
*/
  SoftwareSerial mySerial(2,3);
  DFRobot_GAS_SoftWareUart gas(&mySerial);
#else
/**
  ESP32:IO16-----RX
        IO17-----TX
*/
  DFRobot_GAS_HardWareUart gas(&Serial2); //ESP32HardwareSerial
#endif
#endif

void setup() {

  Serial.begin(115200);

  while(!gas.begin())
  {
    Serial.println("NO Deivces !");
    delay(1000);
  }
```

```
    Serial.println("The device is connected successfully!");

    gas.changeAcquireMode(gas.PASSIVITY);
    delay(1000);

    gas.setTempCompensation(gas.OFF);
  }

  void loop() {

    Serial.print("Ambient ");
    Serial.print(gas.queryGasType());
    Serial.print(" concentration is: ");
    Serial.print(gas.readGasConcentrationPPM());
    Serial.println(" %vol");
    //The measurement unit will only be %vol when the sensor is SEN0465
    //Otherwise the unit will be PPM
    Serial.print("The board temperature is: ");
    Serial.print(gas.readTempC());
    Serial.println(" ℃");
    Serial.println();
    delay(1000);
  }
```

Result

Open the serial monitor to get the gas type, concentration and temperature.

- **The initial power-on requires more than 5 minutes of preheating. It is recommended to preheat more than 24 hours if it has not been used for a long time.**
- **After switching the communication mode and changing the I2C address, the system needs to be powered off and on again.**

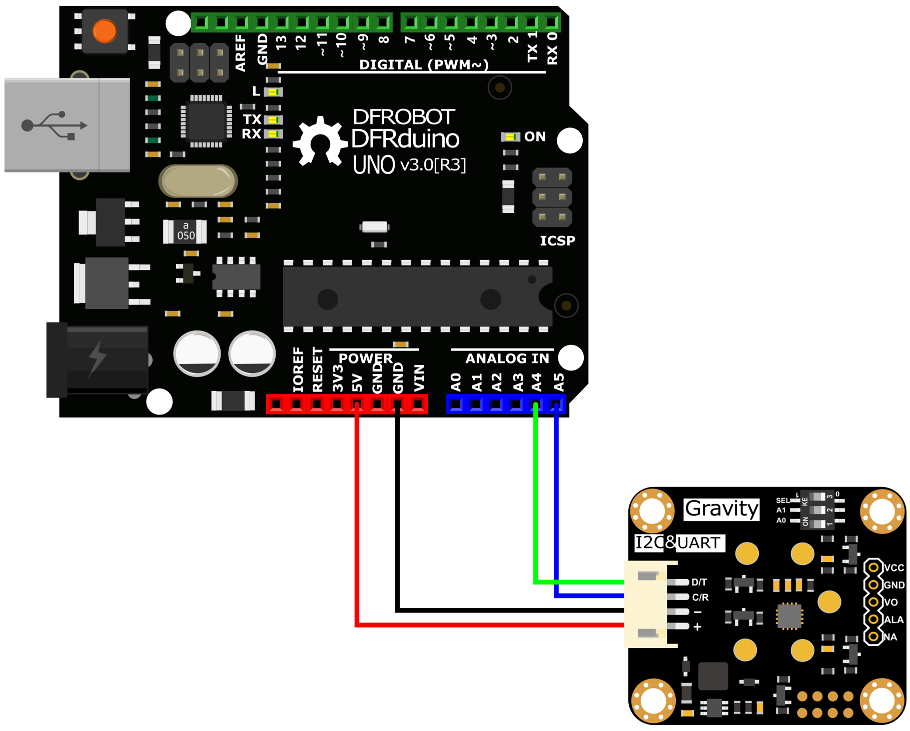


```
The device is connected successfully!
Ambient HF concentration is: 0.00 %vol
The board temperature is: 27.80 ℃

Ambient HF concentration is: 0.01 %vol
The board temperature is: 27.80 ℃

Ambient HF concentration is: 0.01 %vol
The board temperature is: 27.91 ℃

Ambient HF concentration is: 0.61 %vol
The board temperature is: 28.01 ℃

Ambient HF concentration is: 0.81 %vol
The board temperature is: 28.01 ℃
```

()

Acquire data in initiative mode

Connection



(https://dfimg.dfrobot.com/nobody/wiki/f51a4c58a71a062118ca7bdfeeae63ae.png)

- **Sample code**
- Connect the module to the Arduino according to the connection diagram above. Of course, you can also use it with Gravity I/O Expansion Board () to build the project prototype more conveniently and quickly.
- Set the DIP switch SEL on the sensor to 0, and use I2C communication by default.
- The default I2C address is 0x74. If you need to modify the I2C address,You can configure the hardware I2C address through the DIP switch on the module, or run the code to modify the address group to modify the address. The corresponding relationship between the DIP switch and the I2C address parameter is as follows:
  - ADDRESS_0: 0x77, A0=0, A1=0
  - ADDRESS_1: 0x76, A0=1, A1=0
  - ADDRESS_2: 0x75, A0=0, A1=1
  - ADDRESS_3: 0x74, A0=1, A1=1
- Download and install the **DFRobot_GasSensor Library** (https://github.com/DFRobot/DFRobot_MultiGasSensor) (About how to install the library? (https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0))
- Open Arduino IDE and upload the following code to Arduino UNO.

- Open the serial port monitor of Arduino IDE, adjust the baud rate to 115200, and observe the serial port print result.

  Statement

  - In this routine, the sensor will actively return data once a second, and the controller will receive and parse the data.

- Default use I2C communication, mask `#define I2C_COMMUNICATION in the code, and set the dip switch SEL to 1, the sensor is connected to the corresponding port defined by the controller, if use UNO, the blue line is connected to D3 and the green line is connected to D2, if use ESP32, the blue line is connected to IO17 and the green line is connected to IO16. After re-uploading the code, the whole system will be re-powered and will switch to UART communication.

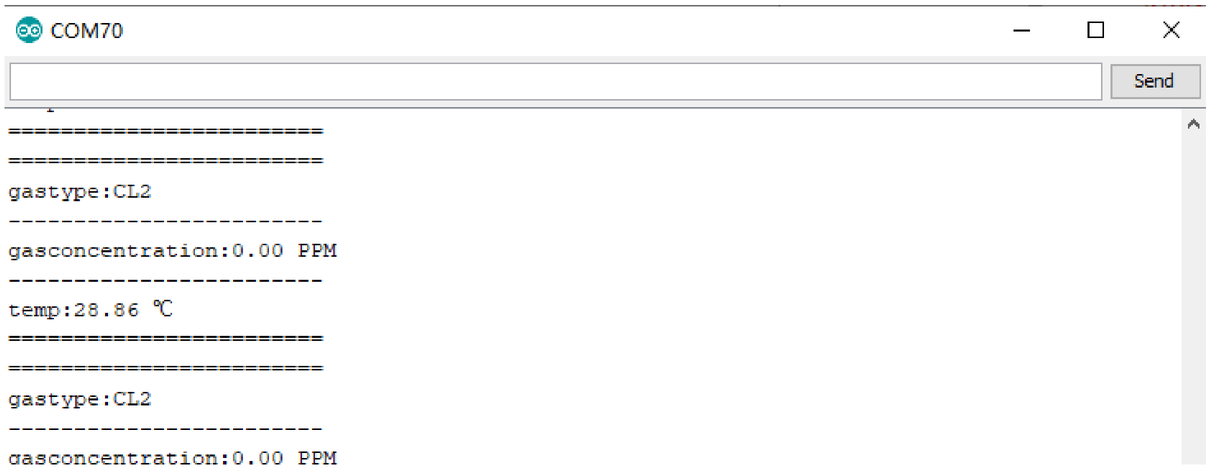
- Turn off temperature compensation by default, modify the code `gas.setTempCompensation(gas.ON);` , turn on temperature compensation after re-uploading the code

```
/*!
  * @file  readGasConcentration.ino
  * @brief Obtain the corresponding gas concentration in the current environment and outpu
  * @n Experiment method: Connect the sensor communication pin to the main control and bur
  * @n Communication mode selection, dial switch SEL:0: IIC, 1: UART
    @n i2c address selection, the default i2c address is 0x74, A1 and A0 are combined into
                | A1 | A0 |
                | 0  | 0  |    0x74
                | 0  | 1  |    0x75
                | 1  | 0  |    0x76
                | 1  | 1  |    0x77   default i2c address
  * @n Experimental phenomenon: You can see the corresponding gas concentration value of t
  */
#include "DFRobot_MultiGasSensor.h"

//Enabled by default, use IIC communication at this time. Use UART communication when disa
#define I2C_COMMUNICATION

#ifdef I2C_COMMUNICATION
#define I2C_ADDRESS 0x74
DFRobot_GAS_I2C gas(&Wire, I2C_ADDRESS);
#else
#if (!defined ARDUINO_ESP32_DEV) && (!defined __SAMD21G18A__)
/**
  UNO:pin_2-----RX
      pin_3-----TX
*/
SoftwareSerial mySerial(2, 3);
DFRobot_GAS_SoftWareUart gas(&mySerial);
#else
/**
  ESP32:IO16-----RX
        IO17-----TX
*/
DFRobot_GAS_HardWareUart gas(&Serial2); //ESP32HardwareSerial
#endif
#endif

void setup() {

  Serial.begin(115200);

  while(!gas.begin())
  {
    Serial.println("NO Deivces !");
    delay(1000);
  }
  gas.setTempCompensation(gas.OFF);

  gas.changeAcquireMode(gas.INITIATIVE);
  delay(1000);
}

void loop() {
  if(true==gas.dataIsAvailable())
  {
    Serial.println("========================");
    Serial.print("gastype:");
    Serial.println(AllDataAnalysis.gastype);
    Serial.println("------------------------");
    Serial.print("gasconcentration:");
    Serial.print(AllDataAnalysis.gasconcentration);
    if (AllDataAnalysis.gastype.equals("O2"))
      Serial.println(" %VOL");
    else
      Serial.println(" PPM");
    Serial.println("------------------------");
    Serial.print("temp:");
    Serial.print(AllDataAnalysis.temp);
    Serial.println(" ℃");
    Serial.println("========================");
  }
  delay(1000);
}
```

Result

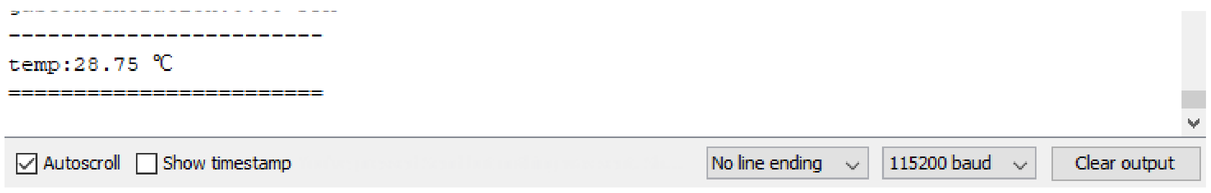
Open the serial monitor, then you can get the corresponding gas concentration.

- **The initial power-on requires more than 5 minutes of preheating. It is recommended to preheat more than 24 hours if it has not been used for a long time.**
- **After switching the communication mode and changing the I2C address, the system needs to be powered off and on again.**



```
========================
========================
gastype:CL2
------------------------
gasconcentration:0.00 PPM
------------------------
temp:28.86 ℃
========================
========================
gastype:CL2
------------------------
gasconcentration:0.00 PPM
```

()

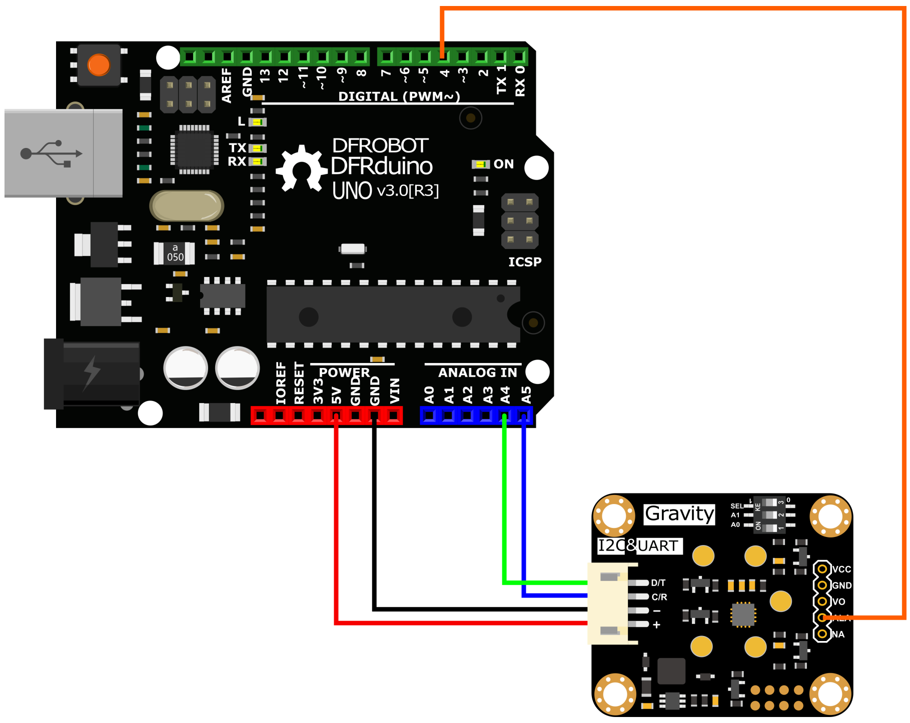


```
------------------------
temp:28.75 ℃
========================
```

[✓] Autoscroll [ ] Show timestamp No line ending 115200 baud Clear output

Threshold alarm function

Connection



(https://dfimg.dfrobot.com/nobody/wiki/28bbfa6d627f27af8ec05e30afbef3c8.png)

- **Sample code**
- Connect the module to the Arduino according to the connection diagram above. Of course, you can also use it with Gravity I/O Expansion Board () to build the project prototype more conveniently and quickly.
- Set the DIP switch SEL on the sensor to 0, and use I2C communication by default.
- The default I2C address is 0x74. If you need to modify the I2C address,You can configure the hardware I2C address through the DIP switch on the module, or run the code to modify the address group to modify the address. The corresponding relationship between the DIP switch and the I2C address parameter is as follows:
  - ADDRESS_0: 0x74, A0=0, A1=0
  - ADDRESS_1: 0x75, A0=1, A1=0
  - ADDRESS_2: 0x76, A0=0, A1=1
  - ADDRESS_3: 0x77, A0=1, A1=1
- Download and install the **DFRobot_GasSensor Library** (https://github.com/DFRobot/DFRobot_MultiGasSensor) (About how to install the library?

(https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0))

- Open Arduino IDE and upload the following code to Arduino UNO.

- Open the serial port monitor of Arduino IDE, adjust the baud rate to 115200, and observe the serial port print result.

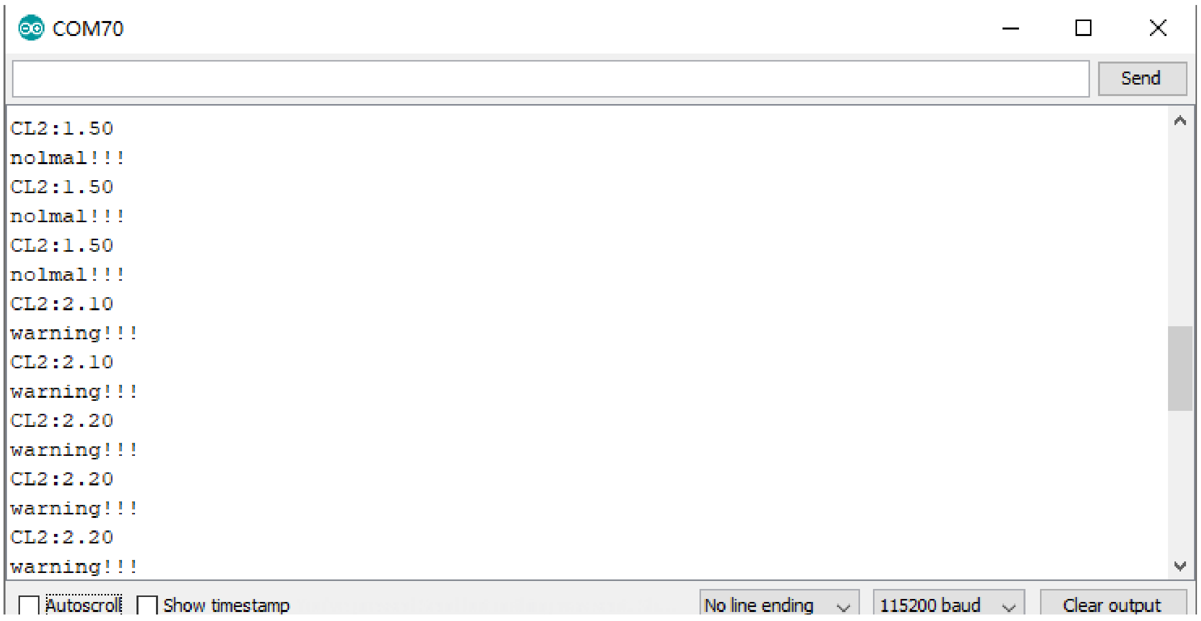
```
/*!
  * @file  setThresholdAlarm.ino
  * @brief Set the threshold alarm of the sensor
  * @n Experiment method: Connect the sensor communication pin to the main control and bur
  * @n Communication mode selection, dial switch SEL:0: IIC, 1: UART
*/
#include "DFRobot_MultiGasSensor.h"

//Enabled by default, use IIC communication at this time. Use UART communication when disa
#define I2C_COMMUNICATION

#ifdef  I2C_COMMUNICATION
#define I2C_ADDRESS    0x77
  DFRobot_GAS_I2C gas(&Wire ,I2C_ADDRESS);
#else
#if (!defined ARDUINO_ESP32_DEV) && (!defined __SAMD21G18A__)
/**
  UNO:pin_2-----RX
      pin_3-----TX
*/
  SoftwareSerial mySerial(2, 3);
  DFRobot_GAS_SoftWareUart gas(&mySerial);
#else
/**
  ESP32:IO16-----RX
        IO17-----TX
*/
  DFRobot_GAS_HardWareUart gas(&Serial2); //ESP32HardwareSerial
#endif
#endif

#define ALA_pin 4

void setup() {

  Serial.begin(115200);

  while(!gas.begin())
  {
    Serial.println("NO Deivces !");
    delay(1000);
  }

  while (!gas.changeAcquireMode(gas.PASSIVITY))
  {
    delay(1000);
  }
```

```
    Serial.println("change acquire mode success!");

    while (!gas.setThresholdAlarm(gas.ON, 2, gas.HIGH_THRESHOLD_ALA ,gas.queryGasType()))
    {
      Serial.println("Failed to open alarm!");
      delay(1000);
    }
    pinMode(ALA_pin,INPUT);
  }

  void loop() {

    Serial.print(gas.queryGasType());
    Serial.print(":");
    Serial.println(gas.readGasConcentrationPPM());
    if (digitalRead(ALA_pin) == 1)
    {
      Serial.println("warning!!!");
    }
    else
    {
      Serial.println("nolmal!!!");
    }
    delay(200);
  }
```

Result

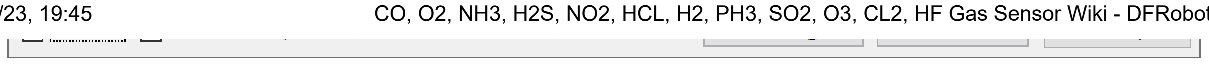
-**After uploading the code successfully, open the serial monitor and you can observe the alarm message.* *

-ALA outputs low level by default when no alarm is triggered. Modify the `HIGH_THRESHOLD_ALA` parameter in the `gas.setThresholdAlarm` function to `LOW_THRESHOLD_ALA` , then ALA outputs high level when no alarm is triggered



```
CL2:1.50
nolmal!!!
CL2:1.50
nolmal!!!
CL2:1.50
nolmal!!!
CL2:2.10
warning!!!
CL2:2.10
warning!!!
CL2:2.20
warning!!!
CL2:2.20
warning!!!
CL2:2.20
warning!!!
```

()



API description

DFR0784 Gravity: Electrochemical Smart Gas Sensor Terminal () There are two data reading modes: active upload and passive response. The factory default is active upload mode, and users can adjust them in the code according to their needs.

Mode selection function "changeAcquireMode()"

Modify the parameters in brackets of the "changeAcquireMode()" function to adjust the data sending mode.

"INITIATIVE" is the active upload mode. In the active upload mode, the sensor will automatically upload parameters every 1 second;

"PASSIVITY" is the passive response mode. In the passive response mode, the sensor will feedback the parameters only every time the data reading function is called.

```
gas.changeAcquireMode(gas.INITIATIVE)
/*
      gas.INITIATIVE            // Active upload mode
      gas.PASSIVITY             // Passive response mode
*/
```

Set the probe type function "setGasType()"

Set the probe type by the "setGasType()" function.

```
gas.setGasType(/*Gas type*/gas.O2);
```

Read the probe type function "queryGasType()"

Through the "queryGasType()" function, You can get the type of current gas probe.

```
gas.queryGasType();
```

For probe compatible types and corresponding parameters, please refer to the table below.

Gas type	CO	O2	NH3	H2S	NO2	HCL
Detection	(0	(0	(0	(0	(0	(0

Detection range	(0-1000)ppm	(0-25)%VOL	(0-100)ppm	(0-100)ppm	(0-20)ppm	(0-10)ppm
Resolution	1ppm	0.1%VOL	1ppm	1ppm	0.1ppm	0.1ppm

Gas type	CO	O2	NH3	H2S	NO2	HCL
V0 voltage output range	(0.6-3)V	(1.5-0)V	(0.6-3)V	(0.6-3)V	(2-0)V	(2-0)V
Response time (T90)	≤30S	≤15S	≤150S	≤30S	≤30S	≤60S

Gas type	H2	PH3	SO2	O3	CL2	HF
Detection range	(0-1000)ppm	(0-1000)ppm	(0-20)ppm	(0-10)ppm	(0-20)ppm	(0-10)ppm
Resolution	1ppm	0.1ppm	0.1ppm	0.1ppm	0.1ppm	0.1ppm
V0 voltage output range	(0.6-3)V	(0.6-3)V	(0.6-3)V	(2-0)V	(2-0)V	(2-0)V
Response time (T90)	≤120S	≤30S	≤30S	≤120S	≤60S	≤60S

Gas concentration reading function "readGasConcentrationPPM()"

The feedback gas concentration value of the gas sensor can be read through the "readGasConcentrationPPM()" function.

```
gas.readGasConcentrationPPM();
```

Temperature reading function "readTempC()"

The onboard temperature sensor data can be read through the "readTempC()" function.

```
gas.readTempC();
```

Voltage reading function "getSensorVoltage()"

The original voltage output V0 of the gas probe can be read through the "getSensorVoltage()" function.

```
gas.getSensorVoltage();
gas.getSensorVoltage();
```

Configure temperature compensation function "setTempCompensation()"

You can enable/disable the temperature compensation function through the "setTempCompensation()" function.

```
gas.setTempCompensation();
/*
      gas.ON      Turn on
      gas.OFF        Turn off
*/
```

Threshold alarm function "setThresholdAlarm()"

You can configure the threshold alarm information through the "setThresholdAlarm()" function

```
gas.setThresholdAlarm(gas.ON, 200, gas.LOW_THRESHOLD_ALA ,gas.queryGasType());
/*
    gas.ON      Turn on
      gas.OFF        Turn off
      200            Set threshold
      gas.LOW_THRESHOLD_ALA Jump to low level when alarming
      gas.HIGH_THRESHOLD_ALA Jump to high level when alarming
      gas.queryGasType() Set alarm gas type
*/
```

I2C address group configuration function "changeI2cAddrGroup()"

You can configure the I2C address group code and switch between different address groups through the "changeI2cAddrGroup()" function.

In order to prevent address conflicts when using multiple sensors, we have prepared 8 groups with a total of 23 addresses. If necessary, You can use "change_sensor_iic_addr.ino" in the library file "example",to switch by modifying the group serial number configuration of "changeI2cAddrGroup()". After the serial port information displays "IIC addr change success!", power on again.