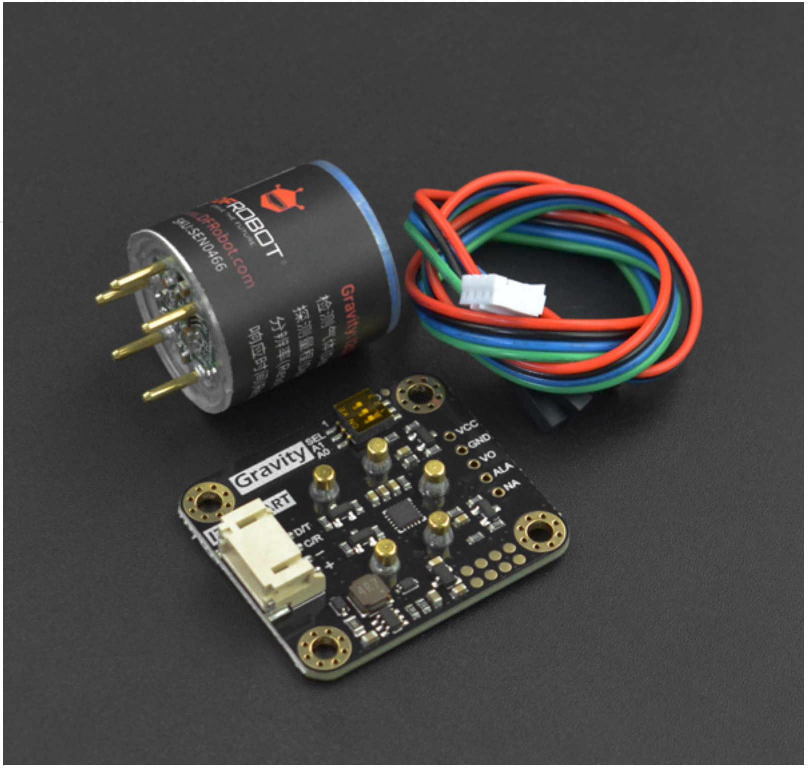
SKU:SEN0465toSEN0476 (https://www.dfrobot.com/product-2510.html)

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

Introduction

This is a sensor that detects oxygen concentration and supports three output modes: analog, I2C, and UART. The probe has been calibrated at the factory, which can quickly and accurately measure the concentration of oxygen in the environment. Can be widely applied to fields like portable devices, air quality monitoring devices, industries, mines, warehouses, and other spaces where the air is not easy to circulate.

The probe adopts the electrochemical principle, has the characteristics of strong anti-interference ability, high stability, high sensitivity, etc., and the service life is as long as two years. The sensor has 32 modifiable I2C addresses, an integrated temperature compensation algorithm, and a threshold alarm function, It has good compatibility with mainstream main control devices such as Arduino, ESP32, and Raspberry Pi. The easy-to-use Gravity interface, coupled with our sample code, can quickly build an oxygen concentration detector.



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, O_2, NH_3, H_2S, NO_2, HCL, H_2, PH_3, SO_2, O_3, CL_2, 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)
- 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**
Gas type	**O2**	**CO**	**H2S**	**Cl2**	**NH3**
Detection range	(0-25)%Vol	(0-1000)ppm	(0-100)ppm	(0-20)ppm	(0-100)p
Resolution	0.1%Vol	1ppm	1ppm	0.1ppm	1pp
V0 voltage output range	(1.5-0)V	(0.6-3)V	(0.6-3)V	(2-0)V	(0.6-3)
Vout1	1.0V@10%vol	0.9V@200ppm	1.5V@50ppm	1.2V@10ppm	1.4V@50

SKU	SEN0465	SEN0466	SEN0467	SEN0468	SEN0469
Response time (T90)	≤15S	≤30S	≤30S	≤60S	≤150

SKU	SEN0471	SEN0472	SEN0473	SEN0474	SEN0475
Gas 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)V	(0.6-3)V	(2-0)V	(2-0)V
Vout1	1.3V@10ppm	0.7V@5ppm	1.3V@500ppm	0.7V@5ppm	0.7V@5ppm
Response time (T90)	≤30S	≤120S	≤120S	≤60S	≤60S

Explanation of VO use:

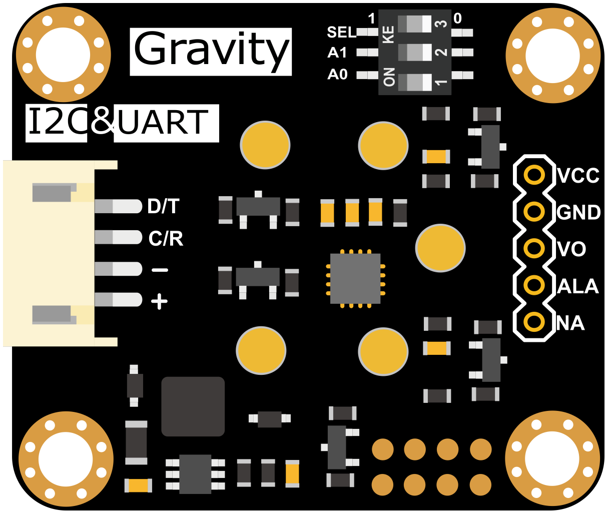
VO: It means original voltage (linear) after amplifying circuit, rather than concentration value of current environment.

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

Label	Name	Function description
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)
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

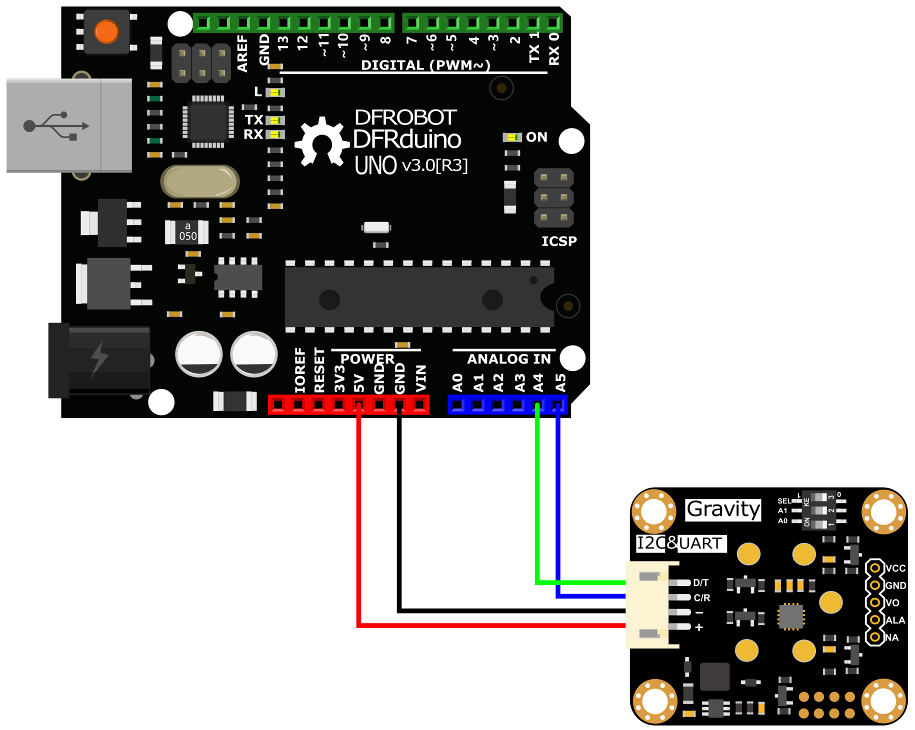
- **Hardware**
    - DFRuino UNO R3 (https://www.dfrobot.com/product-838.html) x1
    - DFR0784 Smart Gas Sensor Terminal x1
    - Gas probe x1
    - Jumper wires
- **Software**

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: 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 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  |    0x77
                | 0  | 1  |    0x76
                | 1  | 0  |    0x75
                | 1  | 1  |    0x74   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
```

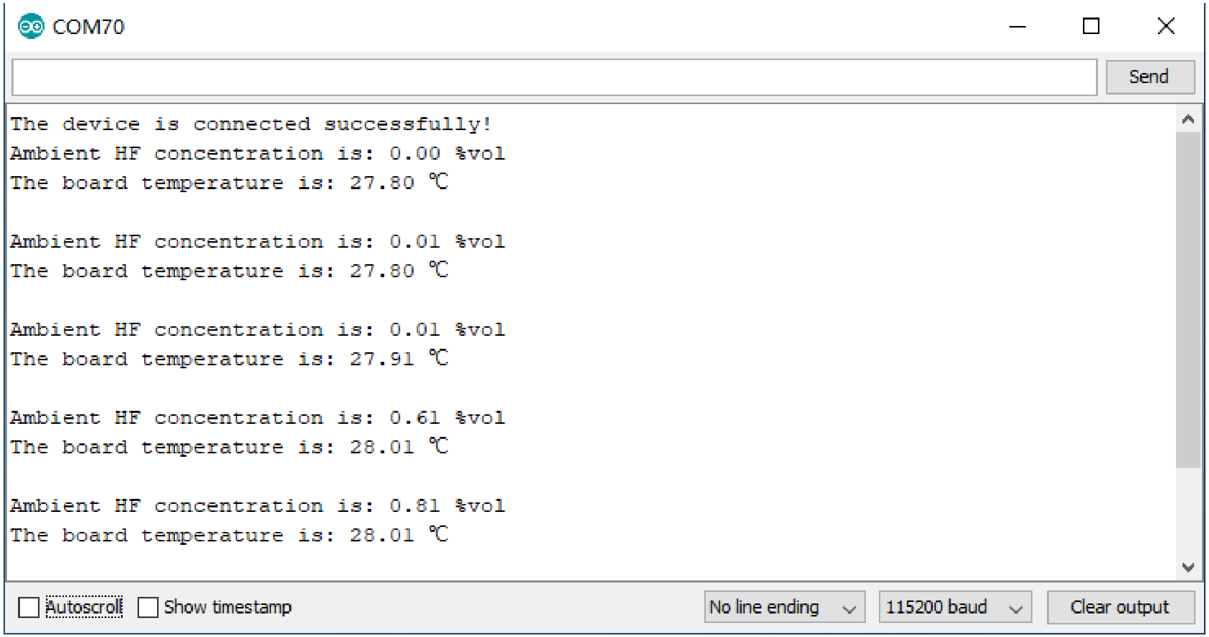
```
#ifdef  I2C_COMMUNICATION
#define I2C_ADDRESS    0x74
  DFRobot_GAS_I2C gas(&Wire ,I2C_ADDRESS);
#else
#if (!defined ARDUINO_ESP32_DEV) && (!defined __SAMD21G18A__)

/**
  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");
  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.**

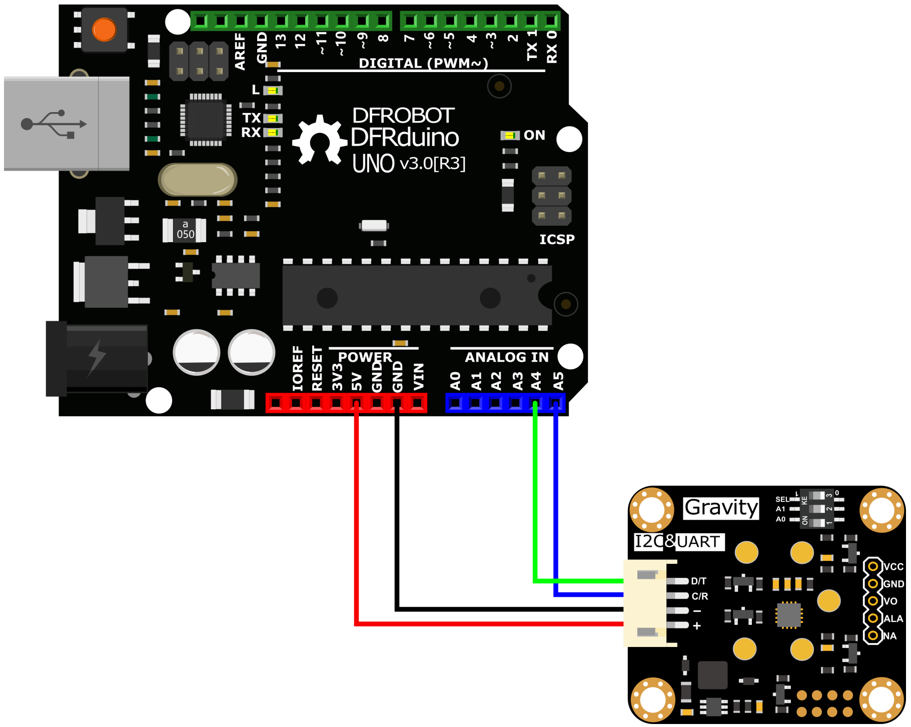


```
The device is connected successfully!
Ambient HF concentration is: 0.00 %vol
The board temperature is: 27.80 ℃

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  |    0x77
                | 0  | 1  |    0x76
                | 1  | 0  |    0x75
                | 1  | 1  |    0x74   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
```

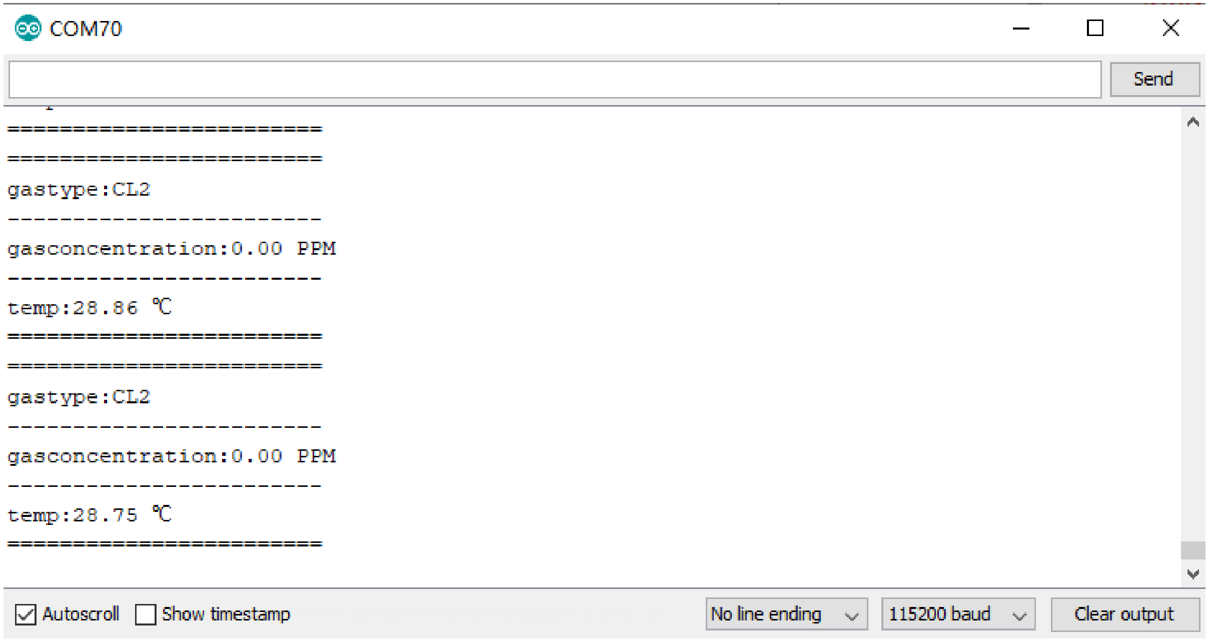
```
#ifdef I2C_COMMUNICATION
#define I2C_ADDRESS 0x74
DFRobot_GAS_I2C gas(&Wire, I2C_ADDRESS);
#else
#if (!defined ARDUINO_ESP32_DEV) && (!defined __SAMD21G18A__)

/**
  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.**

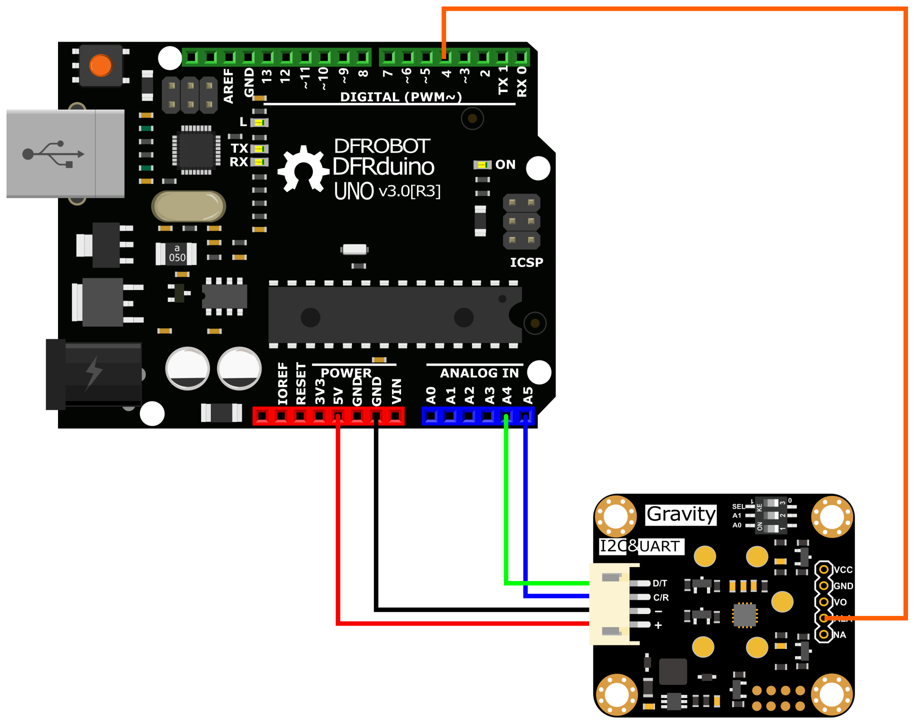


```
========================
========================
gastype:CL2
------------------------
gasconcentration:0.00 PPM
------------------------
temp:28.86 ℃
========================
========================
gastype:CL2
------------------------
gasconcentration:0.00 PPM
------------------------
temp:28.75 ℃
========================
```

()

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: 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**

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
```

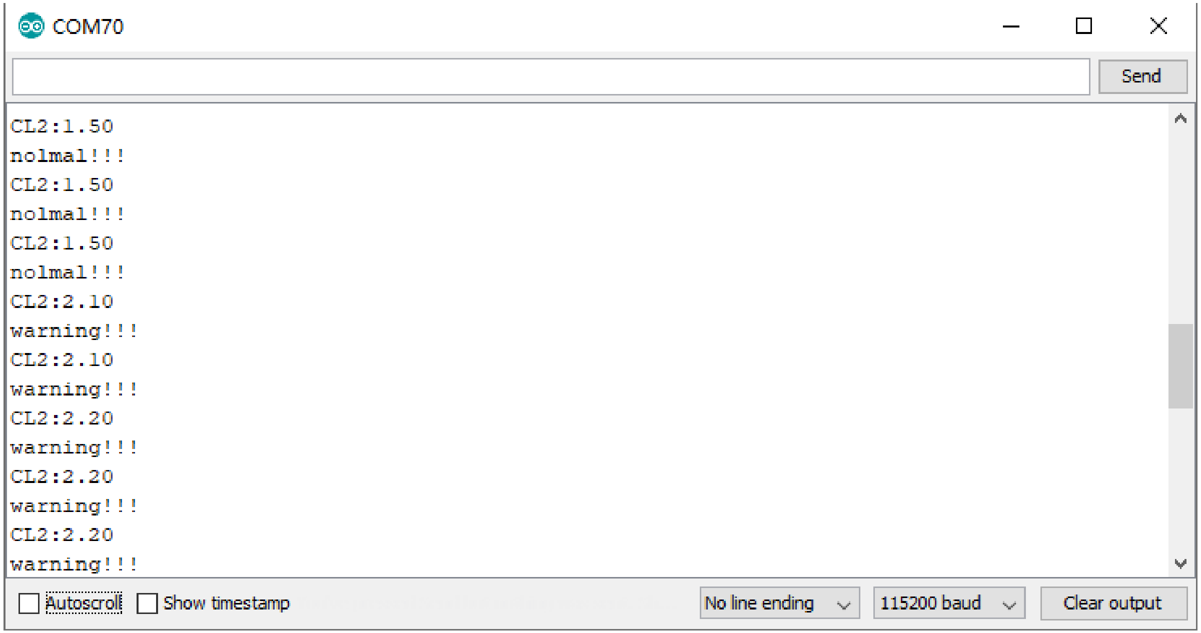
```
      pin_3-----TX
*/
  SoftwareSerial mySerial(2, 3);
  DFRobot_GAS_SoftWareUart gas(&mySerial);
#else

/**
  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 range	(0-1000)ppm	(0-25)%VOL	(0-100)ppm	(0-100)ppm	(0-20)ppm	(0-10)ppm

Gas type	CO	O2	NH3	H2S	NO2	HCL
Resolution	1ppm	0.1%VOL	1ppm	1ppm	0.1ppm	0.1ppm
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();
```

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.

```
gas.changeI2cAddrGroup(i);
 /*
     i            Group number

   //Group serial number and DIP switch configuration table
   A0 A1Dial level    00    01    10    11
   Group number          Group address
     1            0x60  0x61  0x62  0x63
     2            0x64  0x65  0x66  0x67
     3            0x68  0x69  0x6A  0x6B
     4            0x6C  0x6D  0x6E  0x6F
     5            0x70  0x71  0x72  0x73
     6（Default address group） 0x74  0x75  0x76  0x77
     7            0x78  0x79  0x7A  0x7B
     8            0x7C  0x7D  0x7E
 */
```

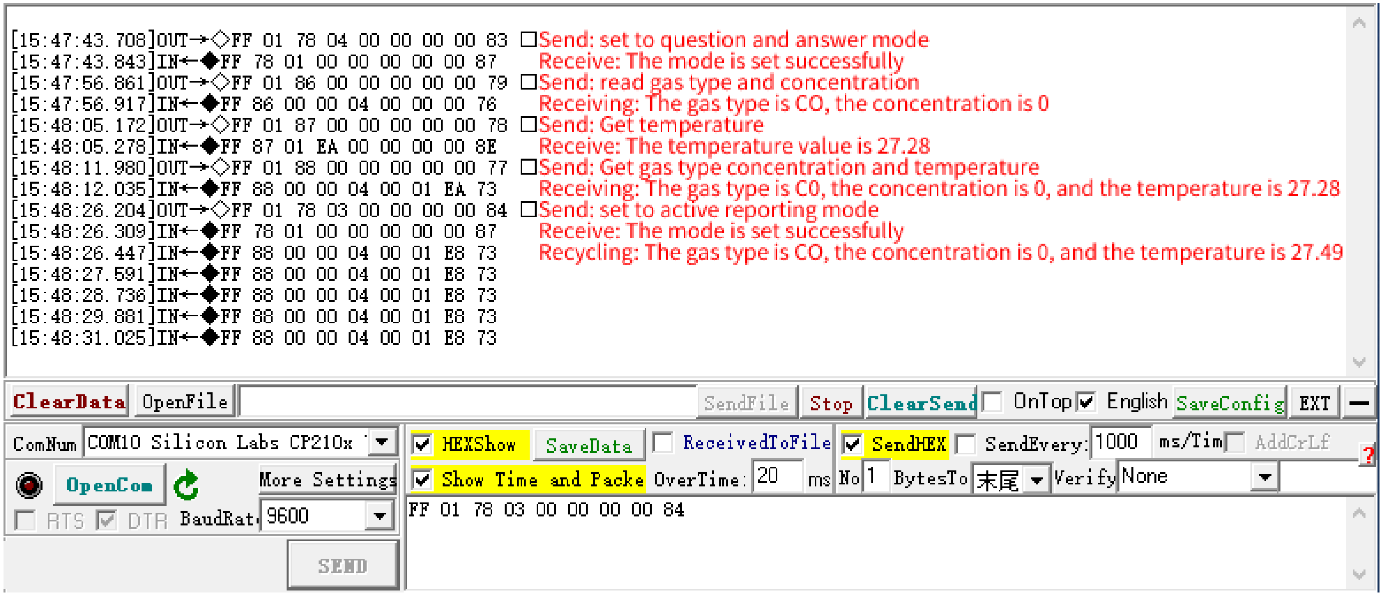
Serial port protocol usage tutorial

Through the UART serial communication protocol, you can connect DFR0784 Gravity: Electrochemical Smart Gas Sensor Terminal () to any controller with UART for data reading and sensor configuration. Note: At this time, the SEL end of the DIP switch on the sensor must be placed in the "1" position

Serial port parameter setting

Baud rate	9600
Data bit	8 bit
Check bit	1 bit

Communication protocol description



```
[15:47:43.708]OUT→◇FF 01 78 04 00 00 00 00 83 □Send: set to question and answer mode
[15:47:43.843]IN←◆FF 78 01 00 00 00 00 00 87   Receive: The mode is set successfully
[15:47:56.861]OUT→◇FF 01 86 00 00 00 00 00 79 □Send: read gas type and concentration
[15:47:56.917]IN←◆FF 86 00 00 04 00 00 00 76   Receiving: The gas type is CO, the concentration is 0
[15:48:05.172]OUT→◇FF 01 87 00 00 00 00 00 78 □Send: Get temperature
[15:48:05.278]IN←◆FF 87 01 EA 00 00 00 00 8E   Receive: The temperature value is 27.28
[15:48:11.980]OUT→◇FF 01 88 00 00 00 00 00 77 □Send: Get gas type concentration and temperature
[15:48:12.035]IN←◆FF 88 00 00 04 00 01 EA 73   Receiving: The gas type is C0, the concentration is 0, and the temperature is 27.28
[15:48:26.204]OUT→◇FF 01 78 03 00 00 00 00 84 □Send: set to active reporting mode
[15:48:26.309]IN←◆FF 78 01 00 00 00 00 00 87   Receive: The mode is set successfully
[15:48:26.447]IN←◆FF 88 00 00 04 00 01 E8 73   Recycling: The gas type is CO, the concentration is 0, and the temperature is 27.49
[15:48:27.591]IN←◆FF 88 00 00 04 00 01 E8 73
[15:48:28.736]IN←◆FF 88 00 00 04 00 01 E8 73
[15:48:29.881]IN←◆FF 88 00 00 04 00 01 E8 73
[15:48:31.025]IN←◆FF 88 00 00 04 00 01 E8 73
```

ComNum COM10 Silicon Labs CP210x
BaudRat 9600
[✓ HEXShow] [✓ SendHEX] SendEvery: 1000 ms/Tim
[✓ Show Time and Packe] OverTime: 20 ms No 1 BytesTo 末尾 Verify None
[✓ English]

```
FF 01 78 03 00 00 00 00 84
```

① 0x78——Modify terminal communication mode

The terminal has two communication modes, active uploading and question and answer. The factory default is active uploading mode, and data is sent every 1s.

Send

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byt
Start bit	addr	Command	Communication mode	--	--	--	--	Che val
0xFF	0x01	0x78	Active upload mode: 0x03 Question and answer mode: 0x04	0x00	0x00	0x00	0x00	0x8 0x8

EXP.FF 01 78 03 00 00 00 00 84 (switch to initiative mode)

EXP.FF 01 78 04 00 00 00 00 83 (switch to passive mode)

Return

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
Start bit	Command	Back to calibration	--	--	--	--	--	Check value
0xFF	0x78	Success: 0x01 Failure: 0x00	0x00	0x00	0x00	0x00	0x00	0x87 0x88

EXP.FF 78 01 00 00 00 00 00 87

② Initiative mode，Data Format

In the active upload mode, the terminal will return data every 1s. The data format is as follows.

Return

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
Start bit	Command	Gas concentration high bit	Gas concentration low bit	Gas type	Decimal places	Temperature value High
0xFF	0x88	0x00	0x00	0x00	0x00	0x00

Note:

- Gas concentration value = (high gas concentration x 256 + low gas concentration) x resolution
- The decimal place is 0, the resolution is 1; the decimal place is 1, the resolution is 0.1; the decimal place is 2, the resolution is 0.01
- For the calculation method of temperature value, please refer to the sample code below: "Calculation of temperature value"

Gas Type Table

Gas Type	Command	Gas Type	Command
NH3	0x02	SO2	0x2B
H2S	0x03	NO2	0x2C
CO	0x04	HCL	0x2E
O2	0x05	CL2	0X31
H2	0x06	HF	0x33
O3	0x2A	PH3	0x45

③ 0x86——Passive mode,Read gas concentration data

In the question and answer mode, you need to send commands to read various parameters of the terminal. The method of reading the gas concentration is as follows.

Send

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
Start bit	addr	Command	--	--	--	--	--	Check value
0xFF	0x01	0x86	0x00	0x00	0x00	0x00	0x00	0x79

EXP.FF 01 86 00 00 00 00 00 79

Return

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte
Start bit	Command	Gas concentration high bit	Gas concentration low bit	Gas type	Decimal places	--	--
0xFF	0x86	0x00	0x00	0x00	0x00	0x00	0x0

EXP.FF 86 00 00 00 00 00 00 7A

Note:

- Gas concentration value = (high gas concentration x 256 + low gas concentration) x resolution
- The decimal place is 0, the resolution is 1; the decimal place is 1, the resolution is 0.1; the decimal place is 2, the resolution is 0.01

④ 0x87——Passive mode,Read temperature data

In the question and answer mode, you need to read various parameters of the terminal by sending commands. The terminal integrates the thermistor, which can obtain the real-time temperature of the terminal. The way to read the terminal temperature is as follows.

Send

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
Start bit	addr	Command	--	--	--	--	--	Check value
0xFF	0x01	0x87	0x00	0x00	0x00	0x00	0x00	0x78

EXP.FF 01 87 00 00 00 00 00 78

Return

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Start bit	Command	Temperature data high bit	Temperature data low bit	--	--	--	--
0xFF	0x87	0x00	0x00	0x00	0x00	0x00	0x00

EXP.FF 87 00 00 00 00 00 00 79

Note:

For the calculation method of temperature value, please refer to the sample code below:

"Calculation of temperature value"

⑤ 0x88——Passive mode,Read temperature and gas concentration data

In the question and answer mode, you need to read various parameters of the terminal by sending commands, and the way to read the temperature and gas concentration data of the terminal is as follows.

Send

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
Start bit	addr	Command	--	--	--	--	--	Check value
0xFF	0x01	0x88	0x00	0x00	0x00	0x00	0x00	0x77

EXP.FF 01 88 00 00 00 00 00 77

Return

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
Start bit	Command	Gas concentration high bit	Gas concentration low bit	Gas type	Decimal places	Temperature value High
0xFF	0x88	0x00	0x00	0x00	0x00	0x00

EXP. FF 88 00 00 00 00 00 00 78

Note:

- Gas concentration value = (high gas concentration x 256 + low gas concentration) x resolution
- The decimal place is 0, the resolution is 1; the decimal place is 1, the resolution is 0.1; the decimal place is 2, the resolution is 0.01
- For the calculation method of temperature value, please refer to the sample code below: "Calculation of temperature value"

⑥ 0x89——Configure threshold alarm function

The terminal has a threshold alarm function, the alarm threshold and judgment logic can be configured. The configuration method is as follows,After the configuration is successful, the entire system needs to be powered on again to take effect.

Note: When no external controller is connected and only the sensor is used to achieve this function, the sensor must be set to active upload mode after the parameters related to the threshold alarm function are configured.

Send

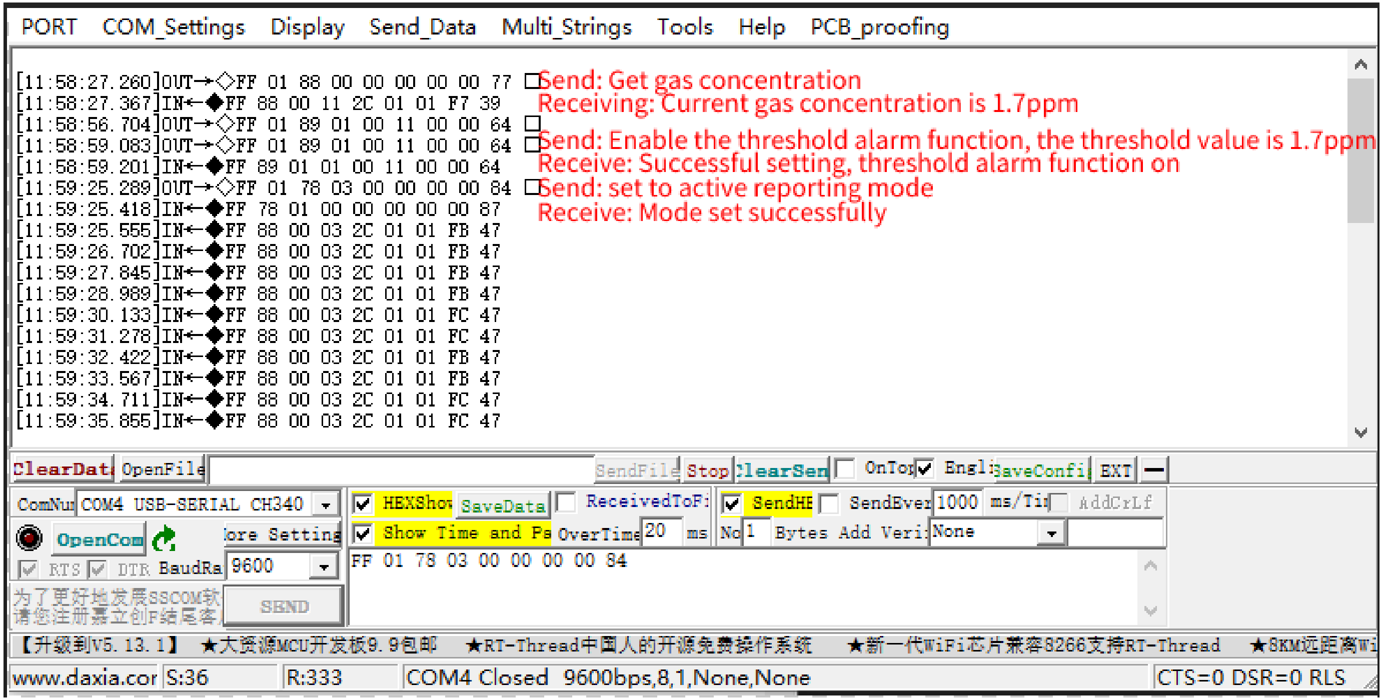
Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byt
Start bit	Empty	Command	Function switch setting	Alarm concentration threshold high bit	Alarm concentration threshold low bit	--	--
0xFF	0x01	0x89	On: 0x01 Off: 0x00	0x00	0x00	0x00	0x0

EXP. FF 01 89 00 00 05 00 00 71 (turn off the alarm function)

EXP. FF 01 89 01 00 05 00 00 70 (open the alarm function)
Please refer to ⑤ for how to calculate the concentration.

Return

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
Start bit	Command	Return configuration result	Function switch status	Alarm concentration threshold high bit	Alarm concentration threshold low bit	--
0xFF	0x89	Success: 0x01 Failure: 0x00	On: 0x01 Off: 0x00	0x00	0x00	0x00



```
[11:58:27.260]OUT→◇FF 01 88 00 00 00 00 00 77 □
[11:58:27.367]IN←◆FF 88 00 11 2C 01 01 F7 39
[11:58:56.704]OUT→◇FF 01 89 01 00 11 00 00 64 □
[11:58:59.083]OUT→◇FF 01 89 01 00 11 00 00 64 □
[11:58:59.201]IN←◆FF 89 01 01 00 11 00 00 64
[11:59:25.289]OUT→◇FF 01 78 03 00 00 00 00 84 □
[11:59:25.418]IN←◆FF 78 01 00 00 00 00 00 87
[11:59:25.555]IN←◆FF 88 00 03 2C 01 01 FB 47
[11:59:26.702]IN←◆FF 88 00 03 2C 01 01 FB 47
[11:59:27.845]IN←◆FF 88 00 03 2C 01 01 FB 47
[11:59:28.989]IN←◆FF 88 00 03 2C 01 01 FB 47
[11:59:30.133]IN←◆FF 88 00 03 2C 01 01 FC 47
[11:59:31.278]IN←◆FF 88 00 03 2C 01 01 FC 47
[11:59:32.422]IN←◆FF 88 00 03 2C 01 01 FB 47
[11:59:33.567]IN←◆FF 88 00 03 2C 01 01 FB 47
[11:59:34.711]IN←◆FF 88 00 03 2C 01 01 FC 47
[11:59:35.855]IN←◆FF 88 00 03 2C 01 01 FC 47
```

Send: Get gas concentration
Receiving: Current gas concentration is 1.7ppm

Send: Enable the threshold alarm function, the threshold value is 1.7ppm
Receive: Successful setting, threshold alarm function on
Send: set to active reporting mode
Receive: Mode set successfully

ComNu COM4 USB-SERIAL CH340

BaudRa 9600

SendEver 1000 ms/Ti

OverTime 20 ms

No 1 Bytes Add Veri: None

```
FF 01 78 03 00 00 00 00 84
```

www.daxia.cor S:36 R:333 COM4 Closed 9600bps,8,1,None,None CTS=0 DSR=0 RLS

To configure the threshold alarm by code using a controller such as Raspberry Pi, you can use

this python code: GAS_ALA.zip (https://img.dfrobot.com.cn/wiki/5cabf4771804207b131ae8cb/4511aa3ce95c0c0ac3224b59f90dbf75.zip)

Checksum calculation

Check value = (inverted (byte 1 + byte 2 + ⋯ ⋯ + byte 7) + 1

The reference routine is as follows:

```
/***************************************************************
* Function name: unsigned char FucCheckSum(uchar *i,ucharln)
* Function description: Sum check (reverse the sum of 1\2\3\4\5\6\7 of the sending and rec
* Function description: reverse the sum of the array element 1 to the penultimate element
***************************************************************/
char data[] = {0xFF,0x01,0x89,0x00,0x00,0x05,0x00,0x00};

unsigned char FucCheckSum(unsigned char *i,unsigned char ln)
{
    unsigned char j,tempq=0;
    i+=1;
    for(j=0;j<(ln-2);j++)
    {
        tempq+=*i;
        i++;
    }
    tempq=(~tempq)+1;
    return(tempq);
}
}

void setup() {
  Serial.begin(115200);
  Serial.println(FucCheckSum(data,8),HEX);

}

void loop() {


}
```

Calculation of temperature value

```
byte Temp_H = 0x01;//Temperature data high bit
byte Temp_L = 0xD9;//Temperature data low bit

void setup() {
  Serial.begin(115200);
  uint16_t temp_ADC = (Temp_H << 8) + Temp_L;
  float Vpd3 = 3 * (float)temp_ADC / 1024;
  float Rth = Vpd3 * 10000 / (3 - Vpd3);
  float Temp = 1 / (1 / (273.15 + 25) + 1 / 3380.13 * log(Rth / 10000)) - 273.15;
  Serial.println(Temp);
}

void loop() {

}
```

Precautions for use

- 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.

FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum** (https://www.dfrobot.com/forum/).

More Documents

- DFRobot-Electrochemical Module .pdf (https://dfimg.dfrobot.com/nobody/wiki/109c27f066f92d0a9d117e7b15663f97.pdf)
- Dimension.pdf (https://dfimg.dfrobot.com/nobody/wiki/c1c65716cf68166ccd23e2b2809a204c.pdf)

DFshopping_car1.png Get **Smart Gas Sensor Terminal** (https://www.dfrobot.com/product-2510.html) from DFRobot Store or **DFRobot Distributor**. (https://www.dfrobot.com/index.php?route=information/distributorslogo)

Turn to the Top