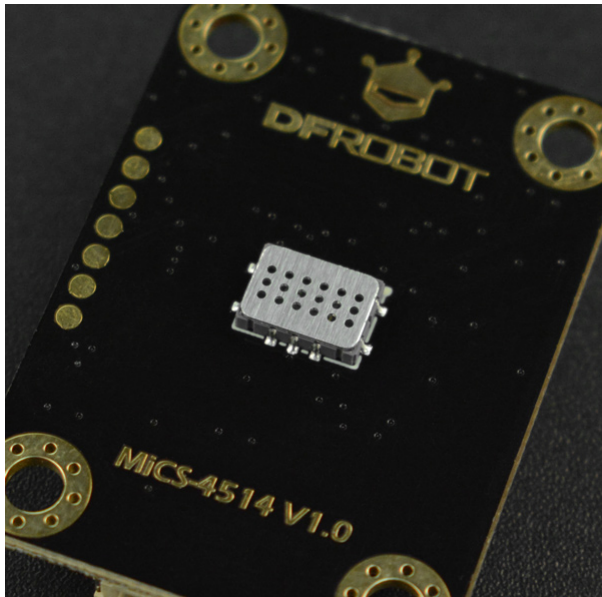


SKU:SEN0377 (<https://www.dfrobot.com/product-2417.html>)



(<https://www.dfrobot.com/product-2417.html>)

Introduction

This is a 3.3/5V compatible MEMS gas concentration sensor from DFRobot. This sensor supports the detection of various gas concentrations like CO, C₂H₅OH (Alcohol), H₂, NO₂, NH₃, and integrates various gas concentration conversion formulas in the code to facilitate the testing and use of sensors. With I²C output and 3.3~5.5V wide voltage input, it is compatible with Arduino, ESP32, Raspberry Pi and other mainstream controllers.

Features

- Support detection of a variety of harmful gas
- Integrate the calculation formulas of various gas concentration
- Low power consumption
- I²C digital output
- Compatible with the 3.3~5.5V master controller

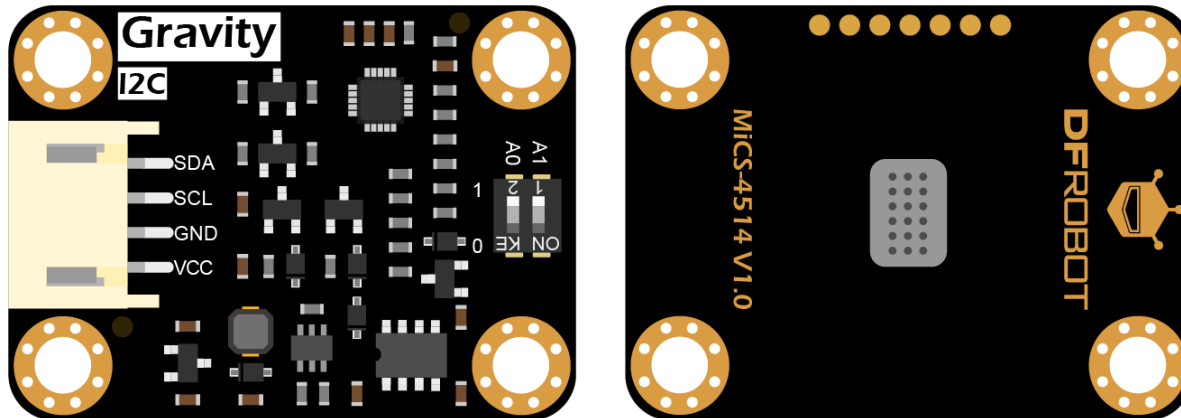
Specification

Specifications

- Detection of Physical Quantities: gas concentration of CO, C₂H₅OH(Alcohol), H₂, NO₂, NH₃, CH₄
- Operating Voltage: 3.3 ~ 5.5V DC
- Power Dissipation: 0.45W (5V)
- Output Signal: I²C (0~3V)
- Measuring Range:
 - 1 – 1000ppm(Carbon monoxide CO)
 - 0.05 – 10ppm(Nitrogen dioxide NO₂)
 - 10 – 500ppm(Ethanol C₂H₅OH)
 - 1 – 1000ppm(Hydrogen H₂)
 - 1 – 500ppm(Ammonia NH₃)
 - >1000ppm(Methane CH₄)
- Working Temperature: -30 ~ 85°C
- Working Humidity: 5 ~ 95%RH (No condensation)
- Storage Temperature: -40~85°C
- Lifespan: >2 years (in the air)

- Circuit Board Size: 27mm*37mm
- Mounting Hole Size: inner diameter 3.1mm/outer diameter 6mm
- Weight:

Board Overview



(<https://img.dfrobot.com.cn/wiki/5b973267c87e6f19943ab3ad/79715420937ccea986bff44a0275ae2.png>)

No.	Name	Functions
1	SDA	I2C data wire SDA

No.	Name	Functions.
2	SCL	I2C clock line SCL
3	GND	negative pole of power supply

4	VCC	positive pole of power supply
---	-----	-------------------------------

Tutorial for Arduino

Download the program to the UNO, open Serial Port Monitor to view the gas concentration, raw data, and other parameters.

Software and hardware preparation

- **Hardware**
 - DFRuino UNO R3 (<https://www.dfrobot.com/product-838.html>) x1
 - SEN0377 Gravity MEMS Gas Sensor (<https://www.dfrobot.com/product-2417.html>) x1
 - Dupont Wires

• Software

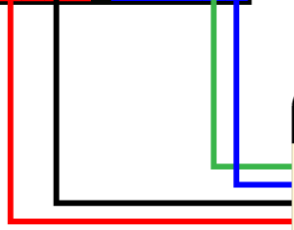
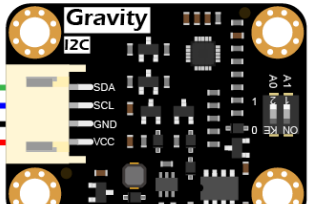
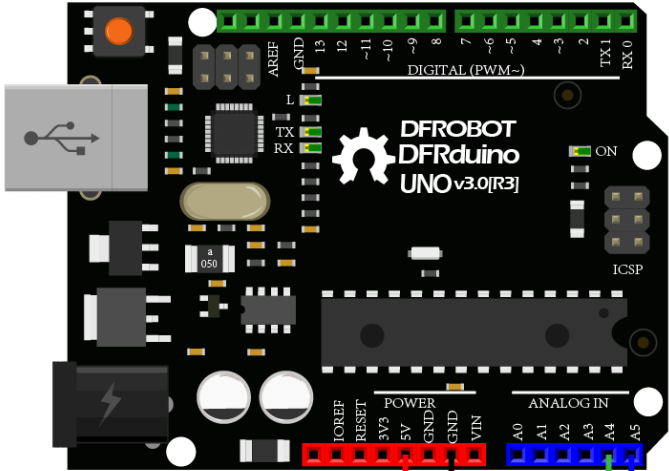
- **Software**

- Arduino IDE Arduino IDE (<https://www.arduino.cc/en/Main/Software>)
- Download and install DFRobot_MICS Library (https://github.com/DFRobot/DFRobot_MICS/archive/refs/heads/master.zip) (About how to install the library? (<https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0>))

Sample 1 - Read the gas concentration (PPM) data as calculated by the sensor

Download the sample program to the Arduino UNO, open Serial Port Monitor to view the gas concentration (PPM) data.

Connection Diagram





(<https://img.dfrobot.com.cn/wiki/5b973267c87e6f19943ab3ad/348bc7336eb02843bbe3c00d25509c25.png>)

Steps

- Connect the module to Arduino by the connection diagram above. You can certainly use with Gravity I/O () to complete the project prototype in a more convenient and faster way.
- Turn the selector switch to the I2C side.
- The I2C address defaults to 0x78, corresponding to the ADDRESS_3 in the code. If you need to modify the I2C address, you can first configure the hardware I2C address via the dial-up switch on the module, and modify the definition ADDRESS_X of the I2C address in the sample code. The correspondence between the dial switch and the I2C address parameter is as follows:
 - ADDRESS_0: 0x75, A0=0, A1=0 (Default Address)
 - ADDRESS_1: 0x76, A0=1, A1=0
 - ADDRESS_2: 0x77, A0=0, A1=1
 - ADDRESS_3: 0x78, A0=1, A1=1
- Download and install DFRobot_MICS Library

4. 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100.

(https://github.com/DFRobot/DFRobot_MICS/archive/refs/heads/master.zip) (About how to install the library? (<https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0>))

- Open Arduino IDE
 - Upload the following sample code to the Arduino UNO.
 - Or open the code **getGasPPM.ino** in the library file sample, and burn to Arduino UNO.
- Open the serial port monitor of the Arduino IDE, adjust the baud rate to 115200, and observe print results.

```

/*!
 * @file getGasPPM.ino
 * @brief Reading Gas concentration, A concentration of one part per million (PPM).
 * @n When using IIC device, select I2C address, set the dialing switch A0, A1 (Address_0)
 * @n When using the Breakout version, connect the adcPin and PowerPin
 * @copyright Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @licence The MIT License (MIT)
 * @author ZhixinLiu(zhixin.liu@dfrobot.com)
 * @version V1.1
 * @date 2021-04-19
 * @get from https://www.dfrobot.com
 * @url https://github.com/dfrobot/DFRobot\_MICS
 */
#include "DFRobot_MICS.h"

#define CALIBRATION_TIME 3 // Default calibration time is three minutes

// When using I2C communication, use the following program to construct an object by DFRobot
/**

```

```

/**
  * iic slave Address, The default is ADDRESS_3
  * ADDRESS_0          0x75          // i2c device address
  * ADDRESS_1          0x76
  * ADDRESS_2          0x77
  * ADDRESS_3          0x78
  */
#define Mics_I2C_ADDRESS ADDRESS_3
DFRobot_MICS_I2C mics(&Wire, Mics_I2C_ADDRESS);

// When using the Breakout version, use the following program to construct an object from
/**!
  * adcPin is A0~A5
  * powerPin is General IO
  */
#define ADC_PIN  A0
#define POWER_PIN 10
//DFRobot_MICS_ADC mics(/*adcPin*/ADC_PIN, /*powerPin*/POWER_PIN);

void setup()
{
  Serial.begin(115200);
  while(!Serial);
  while(!mics.begin()){
    Serial.println("MICS_I2C");
  }
}

```

```

    Serial.println("NO Devices !");
    delay(1000);
} Serial.println("Device connected successfully !");

/**!
    Gets the power mode of the sensor
    The sensor is in sleep mode when power is on,so it needs to wake up the sensor.
    The data obtained in sleep mode is wrong
*/
uint8_t mode = mics.getPowerState();
if(mode == SLEEP_MODE){
    mics.wakeUpMode();
    Serial.println("wake up sensor success!");
}else{
    Serial.println("The sensor is wake up mode");
}

/**!
    Do not touch the sensor probe when preheating the sensor.
    Place the sensor in clean air.
    The default calibration time is 3 minutes.
*/
while(!mics.warmUpTime(CALIBRATION_TIME)){
    Serial.println("Please wait until the warm-up time is over!");
    delay(1000);
}

```

```

    delay(1000);
}
}

void loop()
{
  /**!
  Gas type:
  MICS-4514 You can get all gas concentration
  MICS-5524 You can get the concentration of CH4, C2H5OH, H2, NH3, CO
  MICS-2714 You can get the concentration of NO2
  Methane      (CH4)    (1000 - 25000)PPM
  Ethanol      (C2H5OH) (10 - 500)PPM
  Hydrogen     (H2)     (1 - 1000)PPM
  Ammonia      (NH3)    (1 - 500)PPM
  Carbon Monoxide (CO)   (1 - 1000)PPM
  Nitrogen Dioxide (NO2) (0.1 - 10)PPM
  */
  float gasdata = mics.getGasData(C2H5OH);
  Serial.print(gasdata,1);
  Serial.println(" PPM");
  delay(1000);
  //mics.sleepMode();
}

```




Result

Open the serial port monitor and warm up for approximately 3 minutes to obtain the alcohol gas concentration data.

Notes:

- If you need to detect other gases, you should modify the detected gas settings in the sample code.
- The sensor takes 3 minutes to warm up.



```
158.4 PPM
158.4 PPM
158.4 PPM
158.4 PPM
158.4 PPM
156.9 PPM
156.9 PPM
156.9 PPM
156.9 PPM
156.9 PPM
155.2 PPM
155.2 PPM
155.2 PPM
```

0

```
155.2 PPM
155.2 PPM
155.2 PPM
```

Autoscroll Show timestamp

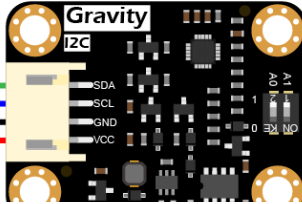
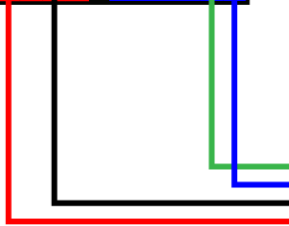
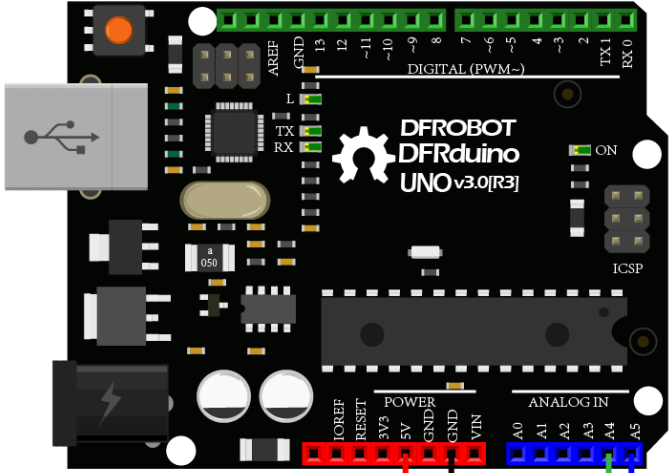
No line ending ▼ 115200 baud ▼ Clear output

Sample 2: Detection for the gas leakage

The MEMS chip MiCS-4514 reacts to a variety of gases, but the linearity of some gas concentration feedback value is poor. Therefore, the concentration value does not have a reference value, but can be used as the determination basis of the gas leakage.

If you need a leak detection of the relevant gas, you can download the sample program to the Arduino UNO, open Serial Port monitor to see if the set gas is leaking.

Connection Diagram





(<https://img.dfrobot.com.cn/wiki/5b973267c87e6f19943ab3ad/348bc7336eb02843bbe3c00d25509c25.png>)

Steps

- Connect the module to Arduino by the connection diagram above. You can certainly use with Gravity I/O ()to complete the project prototype in a more convenient and faster way.
- Turn the selector switch to the I2C side.
- The I2C address defaults to 0x78, corresponding to the ADDRESS_3 in the code. If you need to modify the I2C address, you can first configure the hardware I2C address via the dial-up switch on the module, and modify the definition ADDRESS_X of the I2C address in the sample code. The correspondence between the dial switch and the I2C address parameter is as follows:
 - ADDRESS_0: 0x75, A0=0, A1=0 (Default Address)
 - ADDRESS_1: 0x76, A0=1, A1=0
 - ADDRESS_2: 0x77, A0=0, A1=1
 - ADDRESS_3: 0x78, A0=1, A1=1
- Download and install DFRobot_MICS Library

4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100.

(https://github.com/DFRobot/DFRobot_MICS/archive/refs/heads/master.zip) (About how to install the library? (<https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0>))

- Open Arduino IDE
 - Upload the following sample code to the Arduino UNO.
 - Or open the code **getGasExist.ino** in the library file sample, and burn to Arduino UNO.
- Open the serial port monitor of the Arduino IDE, adjust the baud rate to 115200, and observe the print results.

```
/*!
 * @file getGasExist.ino
 * @brief Reading Gas concentration, A concentration of one part per million (PPM).
 * @n When using IIC device, select I2C address, set the dialing switch A0, A1 (Address_0)
 * @n When using the Breakout version, connect the adcPin and PowerPin
 * @copyright Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @licence The MIT License (MIT)
 * @author ZhixinLiu(zhixin.liu@dfrobot.com)
 * @version V1.1
 * @date 2021-04-19
 * @get from https://www.dfrobot.com
 * @url https://github.com/dfrobot/DFRobot\_MICS
 */
```

```
#include "DFRobot_MICS.h"
```

```
#define CALIBRATION_TIME 3 // Default calibration time is three minutes
```

```
// When using I2C communication, use the following program to construct an object by DFRobot
```

```

// when using I2C communication, use the following program to construct an object by DFROBOT
/**!
    iic slave Address, The default is ADDRESS_3
        ADDRESS_0            0x75            // i2c device address
        ADDRESS_1            0x76

        ADDRESS_2            0x77
        ADDRESS_3            0x78
*/
#define Mics_I2C_ADDRESS ADDRESS_3
DFRobot_MICS_I2C mics(&Wire, Mics_I2C_ADDRESS);

// When using the Breakout version, use the following program to construct an object from
/**!
    adcPin is A0~A5
    powerPin is General IO
*/
#define ADC_PIN    A0
#define POWER_PIN  10
//DFRobot_MICS_ADC mics(/*adcPin*/ADC_PIN, /*powerPin*/POWER_PIN);

void setup()
{
    Serial.begin(115200);
    while(!Serial);
    ...
}

```

```

while(!mics.begin()){
  Serial.println("NO Deivces !");
  delay(1000);
} Serial.println("Device connected successfully !");

/**!
  Gets the power mode of the sensor
  The sensor is in sleep mode when power is on,so it needs to wake up the sensor.
  The data obtained in sleep mode is wrong
  */
uint8_t mode = mics.getPowerState();
if(mode == SLEEP_MODE){
  mics.wakeUpMode();
  Serial.println("wake up sensor success!");
}else{
  Serial.println("The sensor is wake up mode");
}

/**!
  Do not touch the sensor probe when preheating the sensor.
  Place the sensor in clean air.
  The default calibration time is 3 minutes.
  */
while(!mics.warmUpTime(CALIBRATION_TIME)){
  Serial.println("Sensor is not ready for use");
}

```

```

Serial.println("Please wait until the warm-up time is over!");
delay(1000);
}
}

void loop()
{
  /**!
   Type of detection gas
   MICS-4514 You can get all gas state
   MICS-5524 You can get the state of CO, CH4, C2H5OH, C3H8, C4H10, H2, H2S, NH3
   MICS-2714 You can get the state of NO2, H2 ,NO
   CO      = 0x01 (Carbon Monoxide)
   CH4     = 0x02 (Methane)
   C2H5OH  = 0x03 (Ethanol)
   C3H8    = 0x04 (Propane)
   C4H10   = 0x05 (Iso Butane)
   H2      = 0x06 (Hydrogen)
   H2S     = 0x07 (Hydrothion)
   NH3     = 0x08 (Ammonia)
   NO      = 0x09 (Nitric Oxide)
   NO2     = 0x0A (Nitrogen Dioxide)
  */
  int8_t gasFlag = mics.getGasExist(CO);
  // ...
}

```

```
if(gasFlag == EXIST){
    Serial.println("The gas exists!");
}else{
    Serial.println("The gas does not exist!");
}

delay(1000);
//mics.sleepMode();
}
```

Result

Open the serial port monitor and warm up for about 3 minutes, you can detect carbon monoxide (CO)leakage in real time.

Notes:

- If you need to detect other gases, modify the detected gas settings in the sample code yourself.
- The sensor takes 3 minutes to warm up.

```
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!  
The gas does not exist!
```

Autoscroll Show timestamp

No line ending ▾ 115200 baud ▾ Clear output

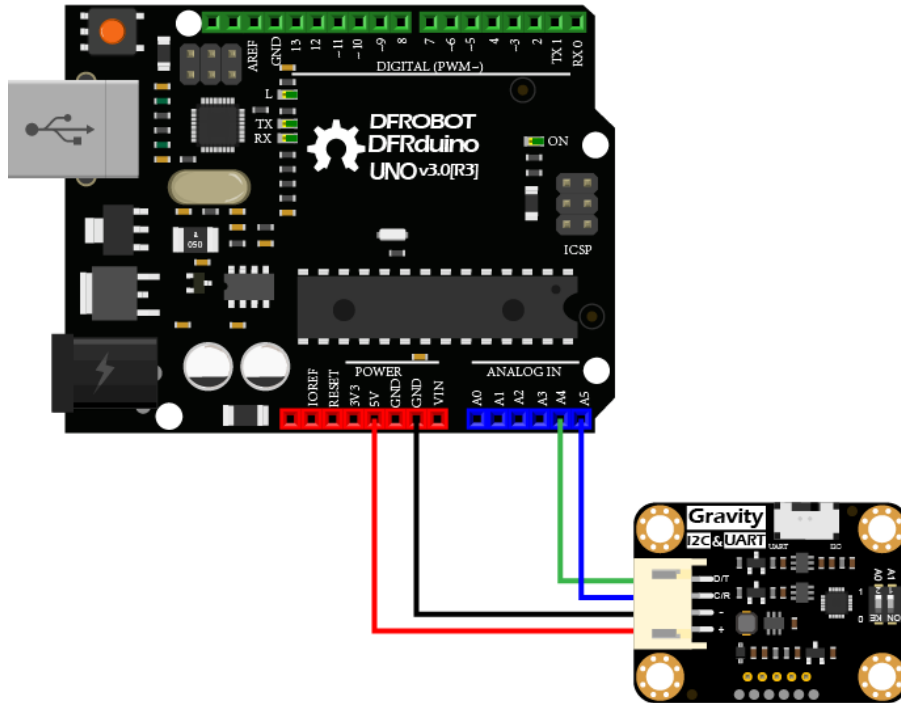
0

Sample Code 3 - Obtain Sensor Analog Value

If you want to obtain the original sensor data, calculate the gas concentration or add your own temperature compensation functions, you can download the sample program to the Arduino

UNO, open serial port monitor to view the original voltage output of the MEMS chip MiCS-4514.

Connection Diagram



(<https://img.dfrobot.com.cn/wiki/none/25070310f21fa2f0642100ba52844fe1.png>)

Steps

- Connect the module to Arduino by the connection diagram above. You can certainly use with Gravity I/O ()for a more convenient and faster completion of the project prototype.
- Turn the selector switch to the I2C side.
- The I2C address defaults to 0x78, corresponding to the ADDRESS_3 in the code. If you need to modify the I2C address, you can first configure the hardware I2C address via the dial-up switch on the module, and modify the definition ADDRESS_X of the I2C address in the sample code. The correspondence between the dial switch and the I2C address parameter is as follows:
 - ADDRESS_0: 0x75, A0=0, A1=0 (Default Address)
 - ADDRESS_1: 0x76, A0=1, A1=0
 - ADDRESS_2: 0x77, A0=0, A1=1
 - ADDRESS_3: 0x78, A0=1, A1=1
- Download and install DFRobot_MICS Library (https://github.com/DFRobot/DFRobot_MICS/archive/refs/heads/master.zip) (About how to install the library? (<https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0>))
- Open Arduino IDE
 - Upload the following sample code to the Arduino UNO.

- Or open the code **getGasExist.ino** in the library file sample, and burn to Arduino UNO.
- Open the serial port monitor of the Arduino IDE, adjust the baud rate to 115200, and observe the print results.

```
/*!
 * @file getADCDData.ino
 * @brief Reading MICS sensor ADC original value
 * @n When using IIC device, select I2C address, set the dialing switch A0, A1 (Address_0)
 * @n When using the Breakout version, connect the adcPin and PowerPin
 * @copyright Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @licence The MIT License (MIT)
 * @author ZhixinLiu(zhixin.liu@dfrobot.com)
 * @version V1.1
 * @date 2021-04-19
 * @get from https://www.dfrobot.com
 * @url https://github.com/dfrobot/DFRobot\_MICS
 */
```

```
#include "DFRobot_MICS.h"
```

```
// When using I2C communication, use the following program to construct an object by DFRobot_MICS
/**!
```

```
    MICS mics(Address_0); // The default is ADDRESS_0
```

```

    I2C Slave Address, The default is ADDRESS_3
    ADDRESS_0             0x75             // i2c device address
    ADDRESS_1             0x76
    ADDRESS_2             0x77
    ADDRESS_3             0x78

*/
#define Mics_I2C_ADDRESS ADDRESS_3
DFRobot_MICS_I2C mics(&Wire, Mics_I2C_ADDRESS);

// When using the Breakout version, use the following program to construct an object from
/**!
    adcPin is A0~A5
    powerPin is General IO
*/
#define ADC_PIN    A0
#define POWER_PIN  10
//DFRobot_MICS_ADC mics(/*adcPin*/ADC_PIN, /*powerPin*/POWER_PIN);

void setup()
{
    Serial.begin(115200);
    while(!Serial);
    while(!mics.begin()){
        Serial.println("NO Deivces !");
        delay(1000);
    }
}

```

```

    delay(1000);
} Serial.println("Device connected successfully !");

/**!
  Gets the power mode of the sensor

  The sensor is in sleep mode when power is on,so it needs to wake up the sensor.
  The data obtained in sleep mode is wrong
  */
uint8_t mode = mics.getPowerState();
if(mode == SLEEP_MODE){
  mics.wakeUpMode();
  Serial.println("wake up sensor success!");
}else{
  Serial.println("The sensor is wake up mode");
}
}

void loop()
{
  int16_t ox_data = 0;
  int16_t red_data = 0;
  /**!
    MICS-5524 Only OX_MODE ADC data can be obtained
    MICS-2714 Only RED_MODE ADC data can be obtained
  */

```

```
    MiCS-4514 Gravity can obtain AllMode ADC data
*/
ox_data = mics.getADCCData(OX_MODE);
//red_data = mics.getADCCData(RED_MODE);
Serial.print("ox  data = ");

Serial.println(ox_data);
//Serial.print("red data = ");
//Serial.println(red_data);
delay(1000);
}
```

Result

Open the serial port monitor and warm up for about 3 minutes, you can obtain the MEMS chip MiCS-4514 raw data.

Notes:

- Two detection units(ox and red) are integrated in MiCS-4514 to calculate the concentration data of different gas. For specific curves, please refer to the sensor data curve in the wiki.

- The sensor takes 3 minutes to warm up.

```
Device connected successfully !
The sensor is wake up mode
ox data = 761
ox data = 762
ox data = 762
ox data = 762
ox data = 762
ox data = 763
ox data = 763
ox data = 763
ox data = 763
ox data = 763
ox data = 764
ox data = 763
ox data = 764
ox data = 764
```

Autoscroll Show timestamp No line ending 115200 baud Clear output

Precautions for use

- Do not expose the sensor to high concentrations of organic solvents, silicone vapor or

cigarette smoke to avoid poisoning the sensitive layer.

- For stable performance, preheat the module for about 3 minutes before testing.
- Sensors shall be placed in a filtered enclosure to protect them from water and dust.

Sensor Data Curve

Two detection units are integrated in the MiCS-4514, into ox and red, to calculate the concentration data for different gases.

FAQ

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

More Documents

- MiCS-4514 Datasheet
(<https://img.dfrobot.com.cn/wiki/5b973267c87e6f19943ab3ad/b5b08fe2ea631f0becdfa0c15db88c4a.pdf>)
- Dimension and Component Layout

- Dimension and Component Layout

(<https://img.dfrobot.com.cn/wiki/5b973267c87e6f19943ab3ad/7ff4e494a339ab2c96220ad9e41582b1.pdf>)

 **Get Gravity MEMS Gas Sensor - I2C - MiCS-4514**

(<https://www.dfrobot.com/product-2417.html>) from DFRobot Store or **DFRobot Distributor**.

(<https://www.dfrobot.com/index.php?route=information/distributorslogo>)

Turn to the Top