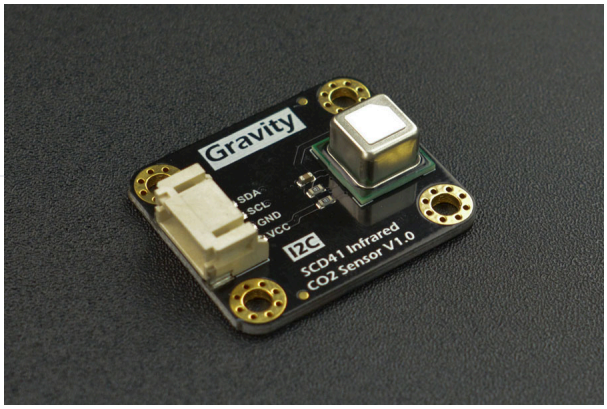
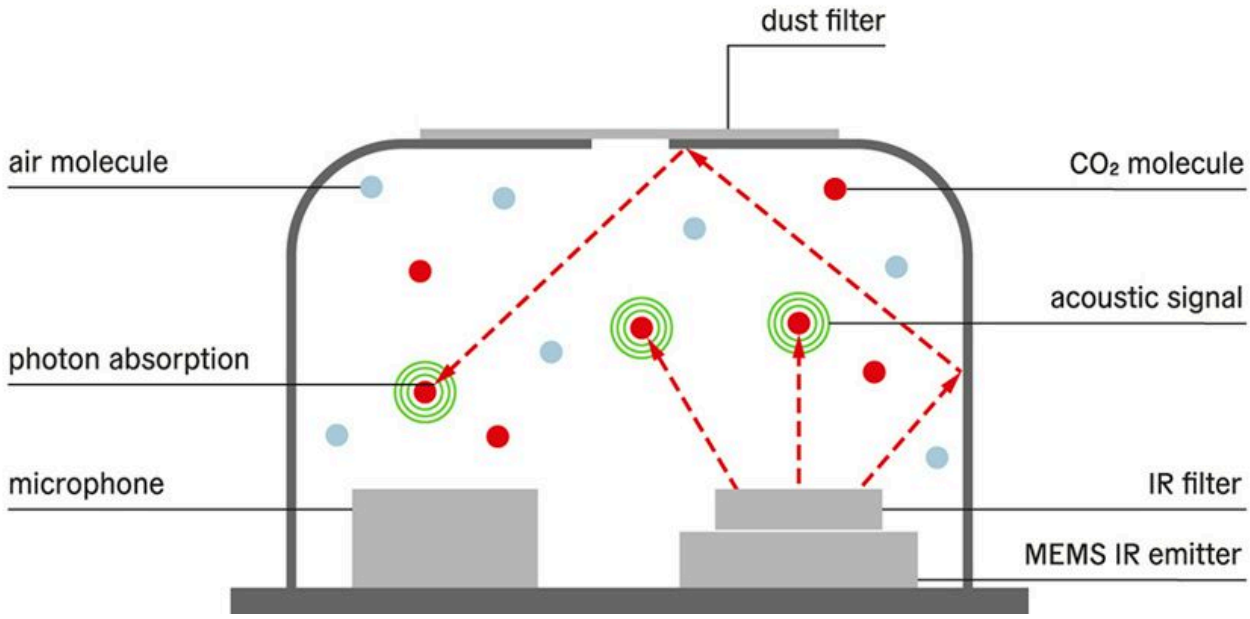
SKU:SEN0536 (https://www.dfrobot.com/product-2646.html)

(https://www.dfrobot.com/product-2646.html)



Introduction

CO2 is a key factor for indoor air quality as high levels compromise human's cognitive performance and well-being.
This Gravity: SCD41 CO2 sensor is based on the SCD41 miniature CO2 sensor from Sensirion.
SCD41 builds on the photoacoustic NDIR sensing principle and Sensirion's patented PAsens® and CMOSens® technology to offer high accuracy at an unmatched price and smallest form factor. On-chip signal compensation is realized with the built-in temperature and humidity sensor, while temperature and humidity data outputs are also available.
SCD41 sensor detects the amount of energy that is absorbed by CO2 molecules. When pulsing the infra-red emitter, CO2 molecules absorb infrared light periodically. This causes additional molecular vibration resulting in a pressure wave inside the measurement chamber. The higher the CO2 concentration, the more light is absorbed, and thus the greater the amplitude of this acoustic wave becomes. A microphone inside the gas chamber measures this, from which the CO2 concentration can then be calculated.



Click to learn more about PAsens® technology (https://sensirion.com/products/technology/).
Click to learn more about NDIR sensor (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6539445/).

Click to learn more about What types of NDIR sensors exist and how do they work? (https://dfimg.dfrobot.com/nobody/wiki/107b314ca1b944c895d061c3ec4de28c.pdf)

Click to learn more about Design-in guide (https://dfimg.dfrobot.com/nobody/wiki/3b565b4e51f83945db7012624ec8682d.pdf)

Features

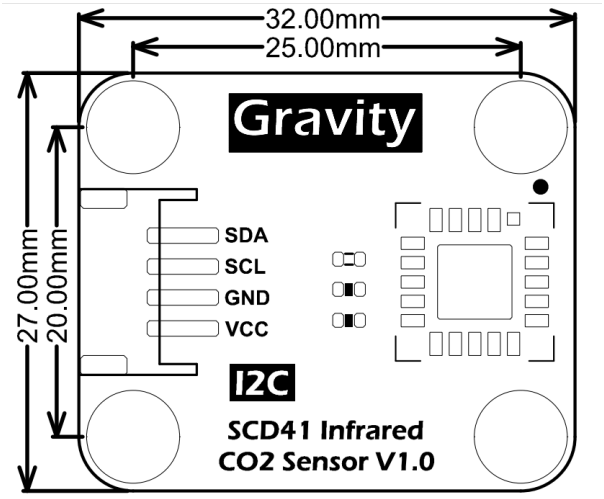
- CO2, temperature and humidity, three in one
- A small size of 32*27*8mm
- Low power, average current<4mA

Application

- Indoor CO2 concentration monitoring
- Ambient monitoring in greenhouse
- Smart ventilation systems

Specification

- Power Supply: 3.3V to 5V
- Average Operating Current: <4mA
- I2C Address: 0x62
- Product Size: 32×27×8mm/1.26×1.06×0.31"



CO2

- Accuracy: ±(40 ppm + 5% MV)
- Measuring Range: 400 - 5000 ppm
- Response Time: 60s

Humidity

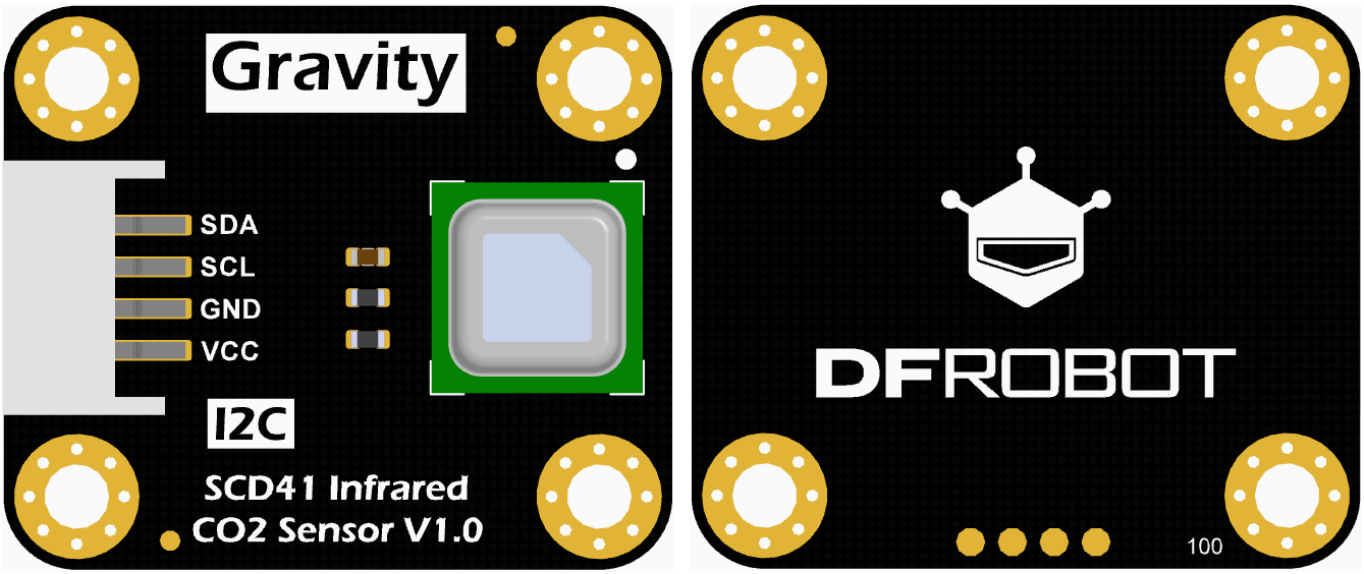
Humidity

- Typical Relative Humidity Accuracy: 6%RH
- Relative Humidity Measuring Range: 0 to 95%RH
- Response Time: 120s

Temperature

- Typical Temperature Accuracy: 0.8℃
- Temperature Measuring Range: -10 to 60℃
- Response Time: 120s

Board Overview



Num	Label	Description
1	VCC	+
2	GND	-
3	SCL	I2C Clock Line
4	SDA	I2C Data Line

Arduino Tutorial

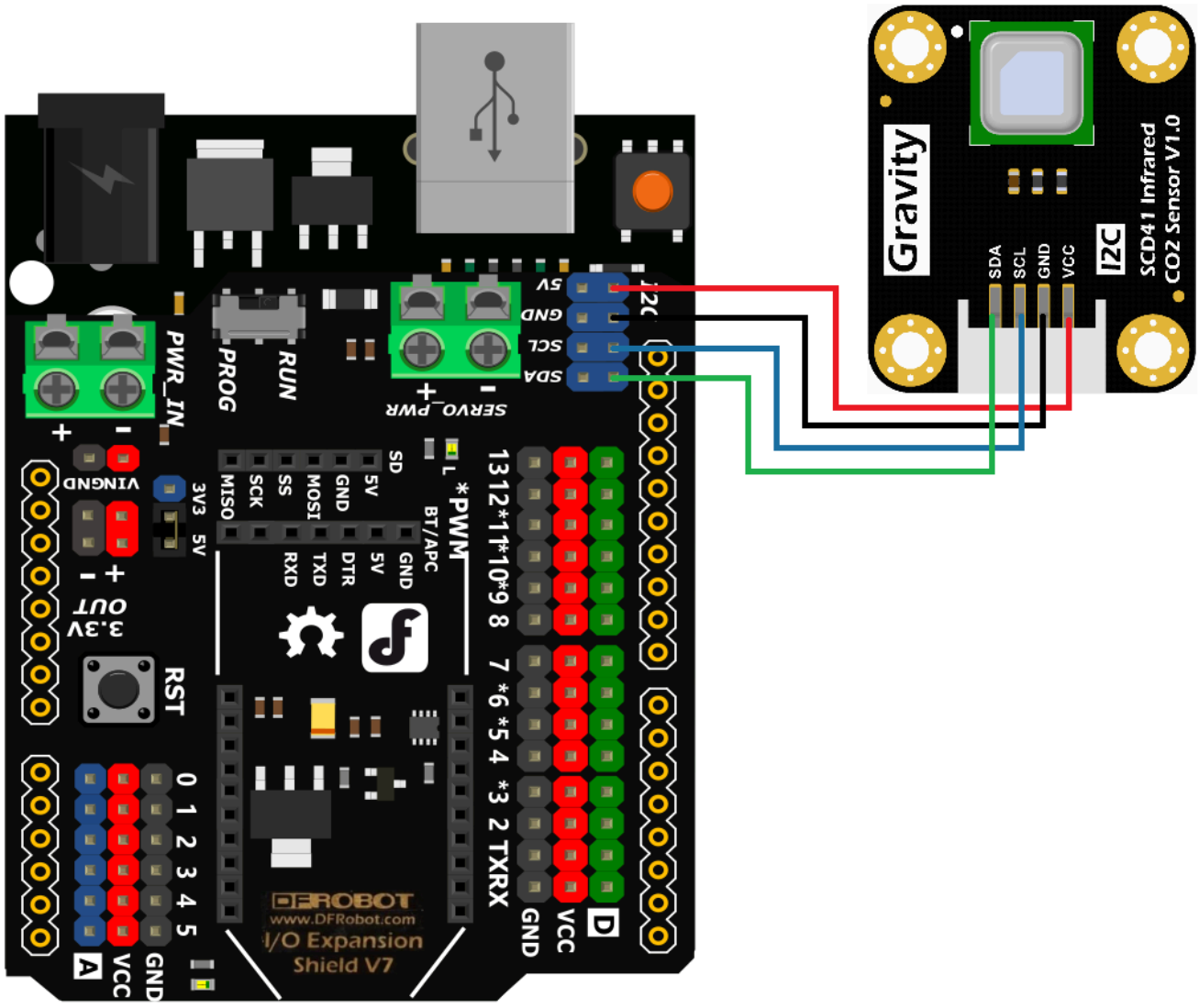
Requirements

- **Hardware**

- DFRduino UNO R3 (https://www.dfrobot.com/product-838.html) (or similar) × 1
- Gravity: SCD41 Infrared CO2 Sensor - I2C × 1
- M-M/F-M/F-F Jumper wires

- **Software**
  - Arduino IDE (https://www.arduino.cc/en/Main/Software)
  - Download and install the **SCD4X Library** (https://github.com/DFRobot/DFRobot_SCD4X) (About how to install the library? (https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0))

Connection Diagram



Sample Code 1 - Period Measurement

Read data from SCD41 periodly.

```
/*!
 * @file  periodMeasure.ino
 * @brief  This sample shows how to configure period measurement mode, compensati
 * @note  The actual compensation and calibration parameter should be changed acc
 * @copyright  Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @license  The MIT License (MIT)
 * @author  [qsjhyy](yihuan.huang@dfrobot.com)
 * @version  V1.0
 * @date  2022-05-11
 * @url  https://github.com/DFRobot/DFRobot_SCD4X
 */
#include <DFRobot_SCD4X.h>

/**
 * @brief Constructor
 * @param pWire - Wire object is defined in Wire.h, so just use &Wire and the metl
 * @param i2cAddr - SCD4X I2C address.
 */
DFRobot_SCD4X SCD4X(&Wire, /*i2cAddr = */SCD4X_I2C_ADDR);

void setup(void)
{
  Serial.begin(115200);

  // Init the sensor
  while( !SCD4X.begin() ){
    Serial.println("Communication with device failed, please check connection");
    delay(3000);
  }
  Serial.println("Begin ok!");

  /**
   * @brief set periodic measurement mode
   * @param mode - periodic measurement mode:
   * @n       SCD4X_START_PERIODIC_MEASURE : start periodic measurement, signal up
   * @n       SCD4X_STOP_PERIODIC_MEASURE : stop periodic measurement command
   * @n       SCD4X_START_LOW_POWER_MEASURE :  start low power periodic measuremer
   * @return None
   * @note The measurement mode must be disabled when configuring the sensor; afte
   */
  SCD4X.enablePeriodMeasure(SCD4X_STOP_PERIODIC_MEASURE);

  /**
   * @brief set temperature offset
```

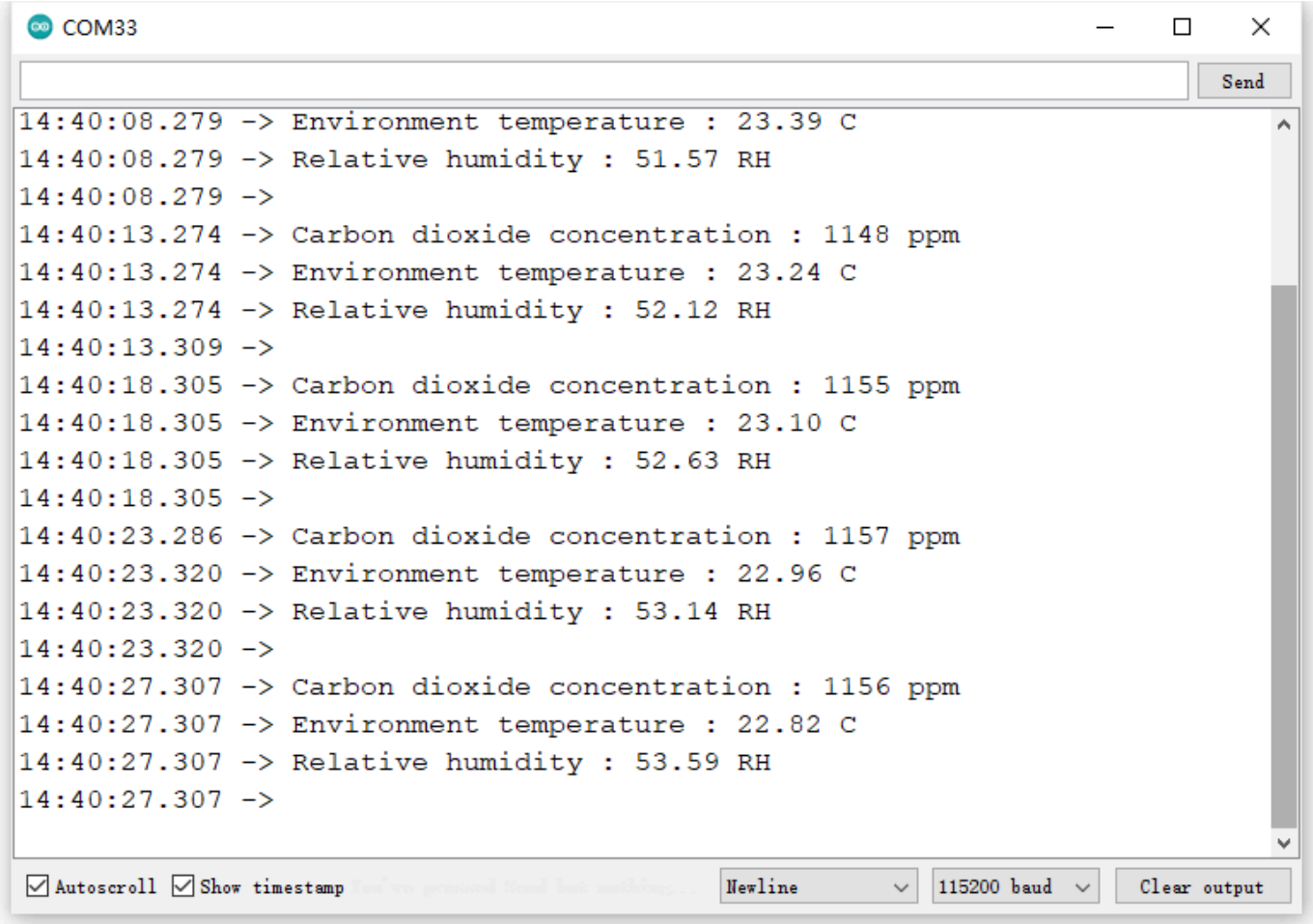
```
 * @details T(offset_actual) = T(SCD4X) - T(reference) + T(offset_previous)
 * @n T(offset_actual): the calculated actual temperature offset that is requir
 * @n T(SCD4X): the temperature measured by the sensor (wait for a period of tir
 * @n T(reference): the standard reference value of the current ambient tempera

 * @n T(offset_previous): the previously set temperature offset
 * @n For example : 32(T(SCD4X)) - 30(T(reference)) + 2(T(offset_previous)) = 4
 * @param tempComp - temperature offset value, unit ℃
 * @return None
 * @note When executing the command, the sensor can't be in period measurement r
 */
SCD4X.setTempComp(4.0);

/**
 * @brief get temperature offset
 * @return The current set temp compensation value, unit ℃
 * @note When executing the command, the sensor can't be in period measurement r
 */
float temp = 0;
temp = SCD4X.getTempComp();
Serial.print("The current temperature compensation value : ");
Serial.print(temp);
Serial.println(" C");

/**
 * @brief set sensor altitude
 * @param altitude - the current ambient altitude, unit m
 * @return None
 * @note When executing the command, the sensor can't be in period measurement r
 */
SCD4X.setSensorAltitude(540);

/**
 * @brief get sensor altitude
 * @return The current set ambient altitude, unit m
 * @note When executing the command, the sensor can't be in period measurement r
 */
uint16_t altitude = 0;
altitude = SCD4X.getSensorAltitude();
Serial.print("Set the current environment altitude : ");
Serial.print(altitude);
Serial.println(" m");

/**
 * @brief set automatic self calibration enabled
 * @param mode - automatic self-calibration mode:
 * @n       true : enable automatic self-calibration
 * @n       false : disable automatic self-calibration
 * @return None
 * @note When executing the command, the sensor can't be in period measurement r
```

```
   @note When executing the command, the sensor can't be in period measurement
 */
// SCD4X.setAutoCalibMode(true);

/**

 * @brief get automatic self calibration enabled
 * @return Automatic self-calibration mode:
 * @n        true : enable automatic self-calibration
 * @n        false : disable automatic self-calibration
 * @note When executing the command, the sensor can't be in period measurement
 */
// if(SCD4X.getAutoCalibMode()) {
//   Serial.println("Automatic calibration on!");
// } else {
//   Serial.println("Automatic calibration off!");
// }

/**
 * @brief persist settings
 * @details Configuration settings such as the temperature offset, sensor altitu
 * @n  parameter are by default stored in the volatile memory (RAM) only and wi
 * @return None
 * @note To avoid unnecessary wear of the EEPROM, the persist_settings command
 * @n  when persistence is required and if actual changes to the configuration
 * @n  The EEPROM is guaranteed to endure at least 2000 write cycles before fai
 * @note Command execution time : 800 ms
 * @n When executing the command, the sensor can't be in period measurement mod
 */
// SCD4X.persistSettings();

/**
 * @brief reinit reinit
 * @details  The reinit command reinitializes the sensor by reloading user sett
 * @return None
 * @note Before sending the reinit command, the stop measurement command must b
 * @n  If the reinit command does not trigger the desired re-initialization,
 * @n  a power-cycle should be applied to the SCD4x.
 * @n  Command execution time : 20 ms
 * @n When executing the command, the sensor can't be in period measurement mod
 */
//SCD4X.moduleReinit();

/**
 * @brief set periodic measurement mode
 * @param mode - periodic measurement mode:
 * @n       SCD4X_START_PERIODIC_MEASURE : start periodic measurement, signal u
 * @n       SCD4X_STOP_PERIODIC_MEASURE : stop periodic measurement command
 * @n       SCD4X_START_LOW_POWER_MEASURE :  start low power periodic measureme
 * @return None
 * @note The measurement mode must be disabled when changing the sensor setting
```

```
    @note The measurement mode must be disabled when changing the sensor setting.
   */
  SCD4X.enablePeriodMeasure(SCD4X_START_PERIODIC_MEASURE);

  Serial.println();

}

void loop()
{
  /**
   * @brief get data ready status
   * @return data ready status:
   * @n        true : data ready
   * @n        false : data not ready
   */
  if(SCD4X.getDataReadyStatus()) {
    /**
     * @brief set ambient pressure
     * @param ambientPressure - the current ambient pressure, unit Pa
     * @return None
     */
    // SCD4X.setAmbientPressure(96000);

    /**
     * @brief Read the measured data
     * @param data - sSensorMeasurement_t, the values measured by the sensor, inc
     * @n  typedef struct {
     * @n    uint16_t   CO2ppm;
     * @n    float   temp;
     * @n    float   humidity;
     * @n  } sSensorMeasurement_t;
     * @return None
     * @note CO2 measurement range: 0~40000 ppm; temperature measurement range: -
     */
    DFRobot_SCD4X::sSensorMeasurement_t data;
    SCD4X.readMeasurement(&data);

    Serial.print("Carbon dioxide concentration : ");
    Serial.print(data.CO2ppm);
    Serial.println(" ppm");

    Serial.print("Environment temperature : ");
    Serial.print(data.temp);
    Serial.println(" C");

    Serial.print("Relative humidity : ");
    Serial.print(data.humidity);
    Serial.println(" RH");

    Serial.println();
```

```
    }
    delay(1000);
}
```

Expected Result 1



Sample Code 2 - Single Measurement

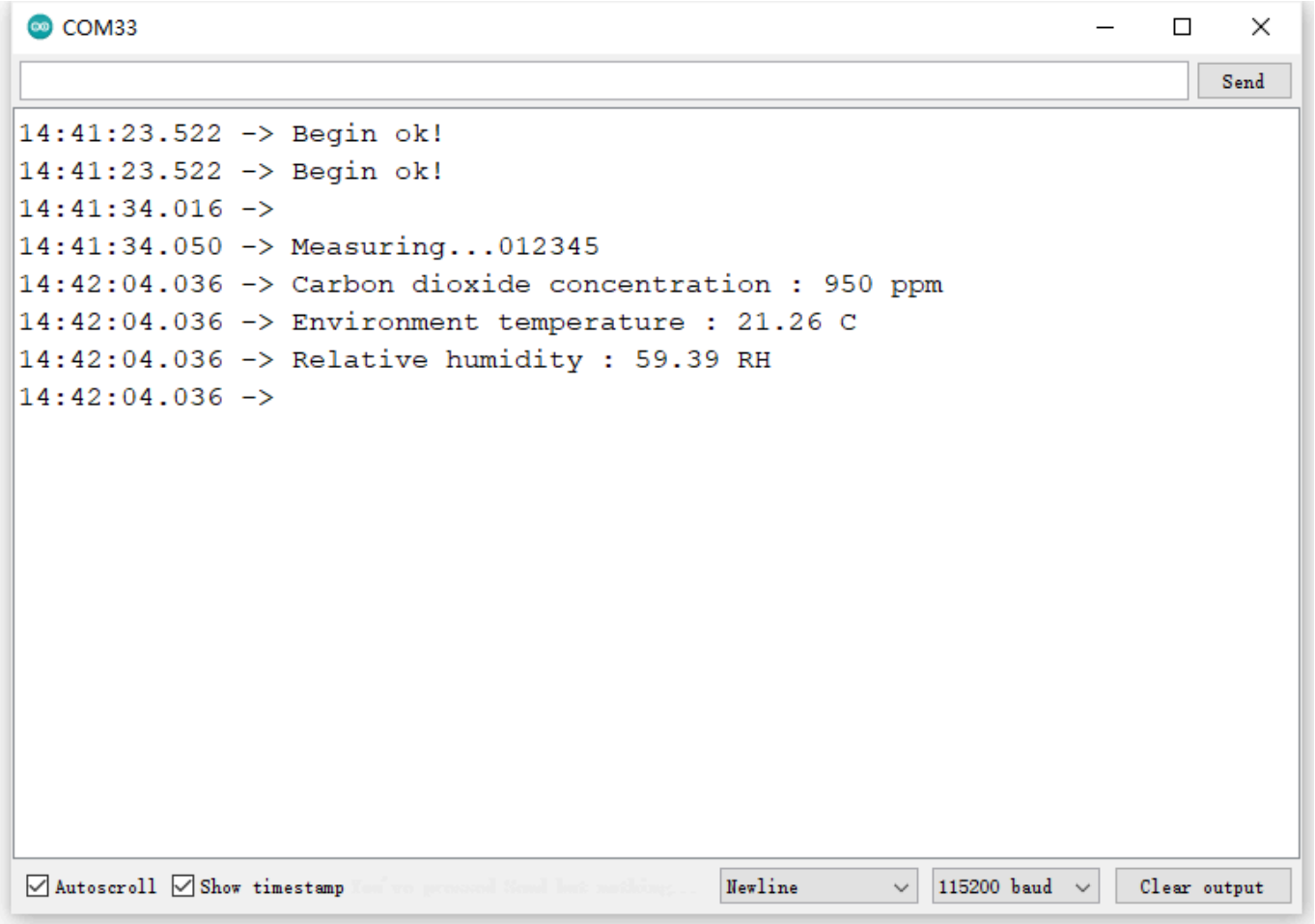
Read data after waking up the sensor, then let it enter sleep mode again.

```
/*!
 * @file  singleShotMeasure.ino
 * @brief  This sample shows how to set single measurement mode, perform reset op
 * @details Get 6 data from single measurement, take the average value, print it,
 * @copyright  Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @license  The MIT License (MIT)
 * @author  [qsjhyy](yihuan.huang@dfrobot.com)
 * @version  V1.0
 * @date  2022-05-11
 * @url  https://github.com/DFRobot/DFRobot_SCD4X
 */
#include <DFRobot_SCD4X.h>

/**
 * @brief Constructor
 * @param pWire - Wire object is defined in Wire.h, so just use &Wire and the metl
 * @param i2cAddr - SCD4X I2C address.
 */
DFRobot_SCD4X SCD4X(&Wire, /*i2cAddr = */SCD4X_I2C_ADDR);

void setup(void)
{
  Serial.begin(115200);

  // Init the sensor
  while( !SCD4X.begin() ){
    Serial.println("Communication with device failed, please check connection");
    delay(3000);
  }
  Serial.println("Begin ok!");

  /**
   * @brief set periodic measurement mode
   * @param mode - periodic measurement mode:
   * @n       SCD4X_START_PERIODIC_MEASURE : start periodic measurement, signal up
   * @n       SCD4X_STOP_PERIODIC_MEASURE : stop periodic measurement command
   * @n       SCD4X_START_LOW_POWER_MEASURE :  start low power periodic measuremer
   * @return None
   * @note The measurement mode must be disabled when configuring the sensor; afte
   */
  SCD4X.enablePeriodMeasure(SCD4X_STOP_PERIODIC_MEASURE);

  /**
   * @brief perform self test
```

```
   * @details The perform_self_test feature can be used as an end-of-line test to
   * @n  functionality and the customer power supply to the sensor.
   * @return module status:
   * @n        0 : no malfunction detected

   * @n        other : malfunction detected
   * @note Command execution time : 10000 ms
   * @n When executing the command, the sensor can't be in period measurement mod
   */
  if(0 != SCD4X.performSelfTest()) {
    Serial.println("Malfunction detected!");
  }

  Serial.println();
}

void loop()
{
  /**
   * @brief Set the sensor as sleep or wake-up mode (SCD41 only)
   * @param mode - Sleep and wake-up mode:
   * @n       SCD4X_POWER_DOWN : Put the sensor from idle to sleep to reduce curr
   * @n       SCD4X_WAKE_UP : Wake up the sensor from sleep mode into idle mode.
   * @return None
   * @note Note that the SCD4x does not acknowledge the wake_up command. Command
   * @n When executing the command, the sensor can't be in period measurement mod
   */
  Serial.print("Waking sensor...");
  SCD4X.setSleepMode(SCD4X_WAKE_UP);

  DFRobot_SCD4X::sSensorMeasurement_t data[6];
  memset(data, 0, sizeof(data));
  uint32_t averageCO2ppm=0;
  float averageTemp=0.0, averageHumidity=0.0;
  Serial.print("Measuring...");
  for(uint8_t i=0; i<6; i++) {
    /**
     * @brief measure single shot (SCD41 only)
     * @details  On-demand measurement of CO2 concentration, relative humidity an
     * @n  Get the measured data through readMeasurement(sSensorMeasurement_t dat
     * @param mode - Single-measurement mode:
     * @n       SCD4X_MEASURE_SINGLE_SHOT : On-demand measurement of CO2 concentr
     * @n                                   Max command duration 5000 [ms].
     * @n       SCD4X_MEASURE_SINGLE_SHOT_RHT_ONLY : On-demand measurement of rel
     * @n                                            Max command duration 50 [ms]
     * @note In SCD4X_MEASURE_SINGLE_SHOT_RHT_ONLY mode, CO2 output is returned a
     * @return None
     * @note When executing the command, the sensor can't be in period measuremen
     */
    SCD4X.measureSingleShot(SCD4X_MEASURE_SINGLE_SHOT);
```

```
    /**
     * @brief get data ready status
     * @return data ready status:
     * @n        true : data ready
     * @n        false : data not ready
     */
    while(!SCD4X.getDataReadyStatus()) {
      delay(100);
    }

    /**
     * @brief Read the measured data
     * @param data - sSensorMeasurement_t, the values measured by the sensor, inc
     * @n  typedef struct {
     * @n    uint16_t   CO2ppm;
     * @n    float   temp;
     * @n    float   humidity;
     * @n  } sSensorMeasurement_t;
     * @return None
     * @note CO2 measurement range: 0~40000 ppm; temperature measurement range: -
     */
    SCD4X.readMeasurement(&data[i]);
    if(0 != i) {   // Discard the first set of data, because the chip datasheet i
      averageCO2ppm += data[i].CO2ppm;
      averageTemp += data[i].temp;
      averageHumidity += data[i].humidity;
    }
    Serial.print(i);
  }
  Serial.print("\nCarbon dioxide concentration : ");
  Serial.print(averageCO2ppm / 5);
  Serial.println(" ppm");

  Serial.print("Environment temperature : ");
  Serial.print(averageTemp / 5);
  Serial.println(" C");

  Serial.print("Relative humidity : ");
  Serial.print(averageHumidity / 5);
  Serial.println(" RH\n");

  // Put the sensor from idle to sleep to reduce current consumption.
  Serial.print("Sleeping sensor...");
  SCD4X.setSleepMode(SCD4X_POWER_DOWN);
  delay(300000);   // Wake up the sensor after 5 minutes, and repeat the above mea
}
```

Expected Result 2



Main API Functions

```
/**
   * @fn enablePeriodMeasure
   * @brief set periodic measurement mode
   * @param mode - periodic measurement mode:
   * @n       SCD4X_START_PERIODIC_MEASURE : start periodic measurement, signal up
   * @n       SCD4X_STOP_PERIODIC_MEASURE : stop periodic measurement command
   * @n       SCD4X_START_LOW_POWER_MEASURE :  start low power periodic measuremen
   * @return None
   * @note The measurement mode must be disabled when configuring the sensor; afte
   */
  void enablePeriodMeasure(uint16_t mode);

  /**
   * @fn readMeasurement
   * @brief Read the measured data
   * @param data - sSensorMeasurement_t, the values measured by the sensor, includ
   * @return None
   * @note CO2 measurement range: 0~40000 ppm; temperature measurement range: -10~
   */
  void readMeasurement(sSensorMeasurement_t * data);

  /**
   * @fn getDataReadyStatus
   * @brief get data ready status
   * @return data ready status:
   * @n        true : data ready
   * @n        false : data not ready
   */
  bool getDataReadyStatus(void);
```

Raspberry Pi Tutorial

Requirements

- **Hardware**
  - Raspberry Pi 4 Model B (https://www.dfrobot.com/product-2028.html) (or similar) x 1
  - Gravity: SCD41 Infrared CO2 Sensor - I2C x 1

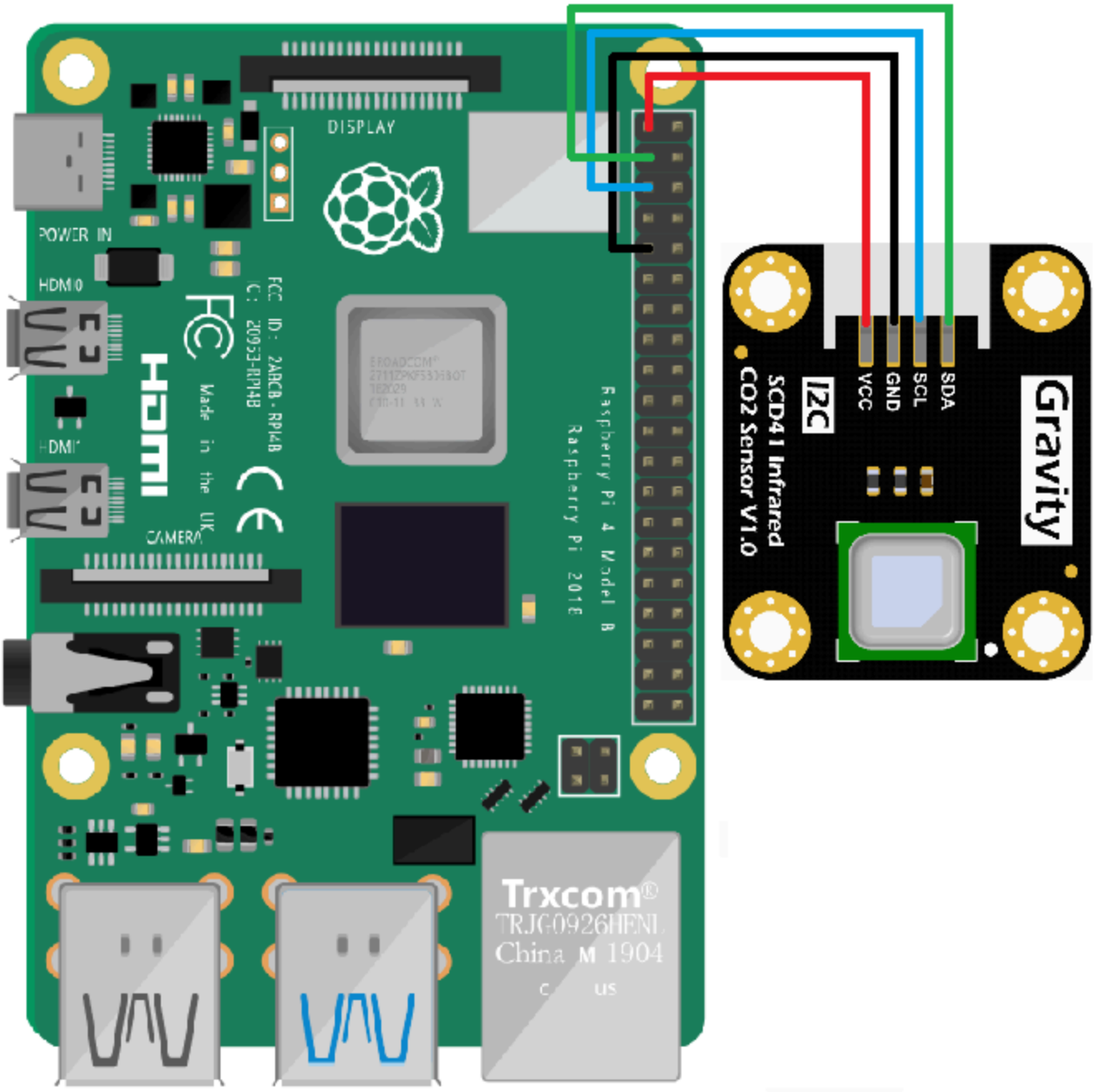
- M-M/F-M/F-F Jumper wires x 1

- **Software**

  - Download and install the SCD41 CO2 Sensor python library (https://github.com/dfrobot/DFRobot_SCD4X).
  - RASPBIAN (https://www.raspberrypi.org/downloads/raspbian)

Connection Diagram

- Connect the module to Raspberry Pi according to the connection diagram. The I2C address is 0x62.



Installation

1. Enable the I2C interface of Raspberry Pi. If it's already enabled, skip this step. Open Terminal, type the following command, and press Enter:
   ```
   pi@raspberrypi:~ $ sudo raspi-config
   ```
   Then use the up and down keys to select "5 Interfacing Options" -> "P5 I2C" and press Enter to confirm “YES”. Reboot the Raspberry Pi.

2. Installing Python dependency libraries and git (networking required). If it is already installed, skip this step. In the Terminal, type the following commands, and press Enter:
   ```
   pi@raspberrypi:~ $ sudo apt-get update
   pi@raspberrypi:~ $ sudo apt-get install build-essential python-dev python-smbus git
   ```

3. Download DFRobot_SCD4X driver library. In Terminal, type the following commands, and press Enter:
   ```
   pi@raspberrypi:~ $ cd Desktop/
   pi@raspberrypi:~/Desktop $ git clone https://github.com/dfrobot/DFRobot_SCD4X
   ```

Sample Code

- Sample Code 1 - Period Measurement (period_measure.py)
- Sample Code 2 - Single-shot Measurement (single_shot_measure.py)

Sample Code 1 - Period Measurement (period_measure.py)

- In Terminal, type the following commands and press Enter to run sample codes:
  ```
  pi@raspberrypi:~/Desktop $ cd DFRobot_SCD4X/python/raspberrypi/examples/
  pi@raspberrypi:~/Desktop/DFRobot_SCD4X/python/raspberrypi/examples/ $ python period_measure.py
  ```

Sample Code 2 - Single-shot Measurement (single_shot_measure.py)

- In Terminal, type the following commands and press Enter to run sample codes:
  ```
  pi@raspberrypi:~/Desktop $ cd DFRobot_SCD4X/python/raspberrypi/examples/
  pi@raspberrypi:~/Desktop/DFRobot_SCD4X/python/raspberrypi/examples/ $ python single_shot_measure.py
  ```

FAQ

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

Q：Encountering I2C address conflicts? A: For a comprehensive guide on identifying and resolving I2C address conflicts in embedded systems, check out this detailed article: How to Resolve I2C Address Conflicts (https://wiki.dfrobot.com/How_to_Resolve_I2C_Address_Conflicts_in_Embedded_Systems). It covers practical hardware and software solutions to ensure smooth communication in your IoT devices.

More Documents

- Schematics (https://dfimg.dfrobot.com/nobody/wiki/4482c60dae7dfada6884e717e83894da.pdf)
- Dimensions (https://dfimg.dfrobot.com/nobody/wiki/40a0a11823fde3e944bf4fa5890b4b12.pdf)
- Dimensions (https://dfimg.dfrobot.com/nobody/wiki/d89501eb233f7f72908980d9969dd40d.pdf)
- Design-in guide (https://dfimg.dfrobot.com/nobody/wiki/3b565b4e51f83945db7012624ec8682d.pdf)