



# Introduction

---

Gravity: CCS811 Air Quality Sensor can measure the eCO<sub>2</sub> (equivalent CO<sub>2</sub>) and TVOC (Total Volatile Organic Compounds) density. It can be widely used in many applications, such as air quality detection, air purifiers, ventilation system and so on. This CCS811 air quality sensor uses AMS's unique micro-hot plate technology. Compared to conventional gas sensors, this sensor is lower power consumption, heating faster, and smaller. Internally integrated ADCs and MCUs allow data to be collected, calculated, and returned via IIC. The CCS811 supports a concentration alarm, which is triggered when the concentration exceeds the user-set threshold.

The CCS811 air quality sensor supports multiple modes, such as detect in every second, in every 10s, in every one minute, in every 250ms and sleep mode. These modes are optimized for low power consumption during the sensor measurement, so the CCS811 is also suitable for portable applications.



**NOTE:** The chip has stretched the clock in IIC. So, it is not compatible with some controllers, such as Raspberry Pi.

The following table shows the effect of each mode on the detection of TVOC and the power consumption.

The following table shows the effects of carbon dioxide and TVOC on the human body.

Carbon Dioxide (PPM)	Effect on Human		TVOC Concentration (PPB)	Effect on Human
<500	Normal		<50	Normal
500-1000	A little uncomfortable		50-750	Anxious,uncomfortable
1000-2500	Tired		750-6000	depressive, headache
2500-5000	Unhealthy		>6000	headache and other nerve problems

## Features

---

- Short warm-up time

- Low power consumption
- High integrated MCUs and ADCs
- IAQ threshold alert

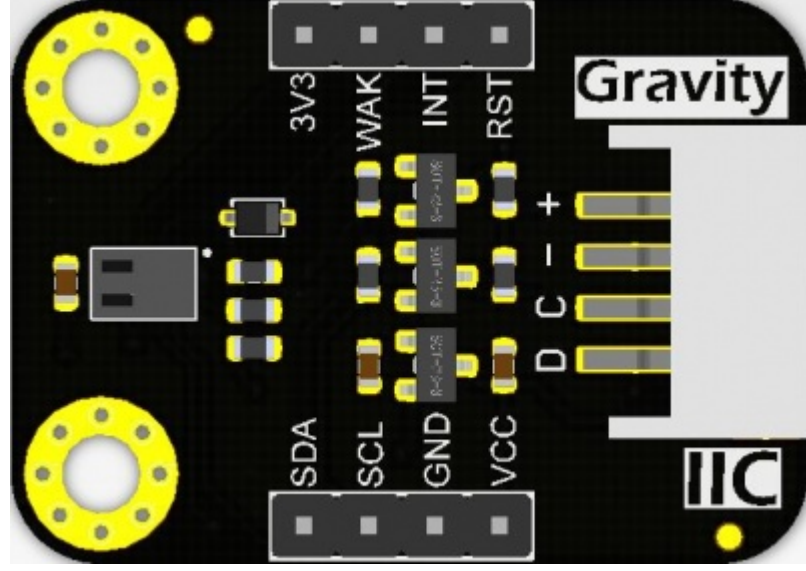
## Specification

---

- Operating Voltage: 3.3V~5.5V
- Warm-up Time: <15s
- IIC Address: 0x5A (in default) / 0x5B
- Operating Temperature: -40°C~85°C
- Operating Humidity: 0ppb~1100ppb
- Dimension: 22x31mm / 0.87x1.22 inches

## Board Overview

---





**DFROBOT**

CCS811 Ultra-Low Power Air Quality  
Sensor-eTVOC and eCO2(V1.0)



RST



INT



WAKE



3V3



ADDR\_SEL



VCC



GND

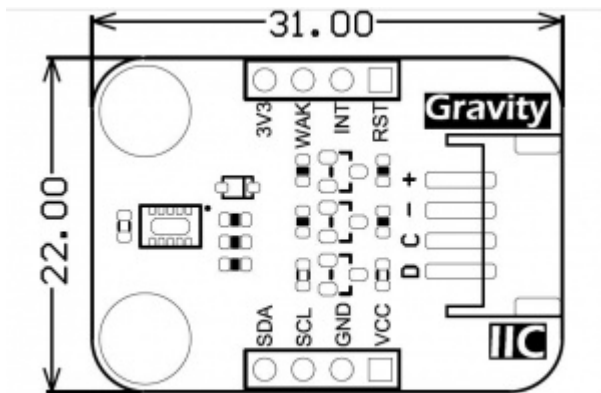


SCL



SDA





Num	Label	Description
1	+/VCC	+
2	-/GND	-
3	C/SCL	IIC clock line

Num	I2C data-line Label	Description
5	3V3	+ 3.3V

6	WAKE	Switch pin: awake in low level / sleep in high level
7	INT	Interrupt pin: interrupt in low level
8	RST	Reset pin: reset in low level
9	ADDR_SEL	Select IIC address: 0x45 in low(in default) / 0x5B in high

## Tutorial

---

The product uses the Gravity standard IIC interface, which is relatively simple to use. Connect the sensor to UNO (or other motherboard) as shown in the wiring diagram.

The product warm-up time is short, accurate readings can be made quickly once powered up. The read time can be shortened by setting the environmental baseline (the baseline acquisition and



setup method is explained below; for more details about baseline, go to the end of the wikipage to find the related document).



**NOTE:** Please run the sensor for 48hours when using it for the first time.

## Requirements

- **Hardware**
  - DFRduino UNO R3 (<https://www.dfrobot.com/product-838.html>) (or similar) x 1
  - Gravity: CCS811 Air Quality Sensor x 1
  - Jumper wires
- **Software**
  - Arduino IDE (<https://www.arduino.cc/en/Main/Software>)
  - Download and install the **CCS811 Library and examples** ([https://github.com/DFRobot/DFRobot\\_CCS811](https://github.com/DFRobot/DFRobot_CCS811)) (About how to install the library? (<https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0>))
- **About API Function List**

```

/**
 * @brief Judge if the data can be read
 * @return true when the reading is successful, false means it fails to read.
 */
bool checkDataReady();

/**
 * @brief Set environment parameter
 * @param temperature Input temperature value, unit: centigrade, range (-40~85°C)
 * @param humidity Input humidity value, unit: RH, range (0~100)
 */
void setInTemHum(float temperature, float humidity);

/**
 * @brief Measurement parameter configuration
 * @param mode:in typedef enum{
 *             eClosed, //Idle (Measurements are disabled in this mode)
 *             eCycle_1s, //Constant power mode, IAQ measurement every second
 *             eCycle_10s //Pulse heating mode, IAQ measurement every 10 seconds

```

```

    *         eCycle_10s,    //Pulse heating mode IAQ measurement every 10 seconds
    *         eCycle_60s,    //Low power pulse heating mode IAQ measurement every 60
    *         eCycle_250ms    //Constant power mode, sensor measurement every 250ms 1
    *         }eCycle_t;
    * @param thresh:0 for Interrupt mode operates normally; 1 for interrupt mode only as
    * @param interrupt:0 for Interrupt generation is disabled; 1 for the nINT signal is
    */
setMeasurementMode(eCycle_t mode, uint8_t thresh = 0, uint8_t interrupt = 0),

/**
 * @brief Get the current carbon dioxide concentration
 * @return current carbon dioxide concentration, unit:ppm
 */
uint16_t getCO2PPM();

/**
 * @brief Get current TVOC concentration
 * @return Return current TVOC concentration, unit: ppb
 */
uint16_t getTVOCPPB();

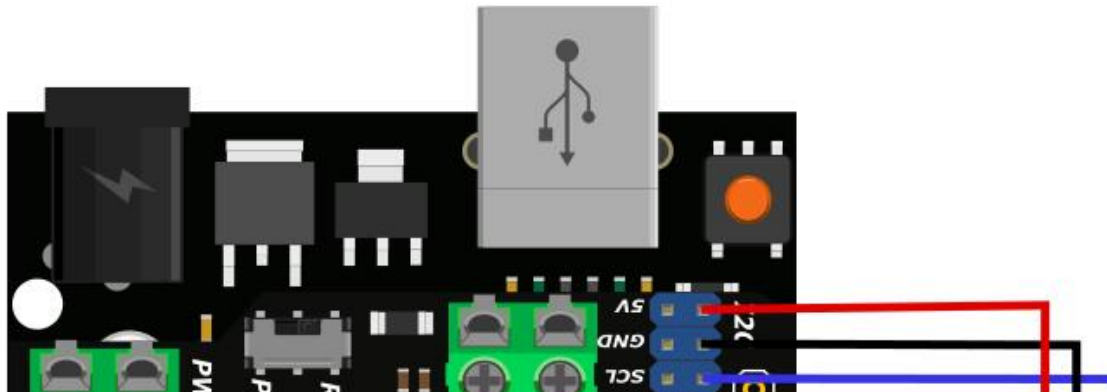
/**
 * @brief get the current baseline number
 * @return a Hexadecimal number of the current baseline number
 */

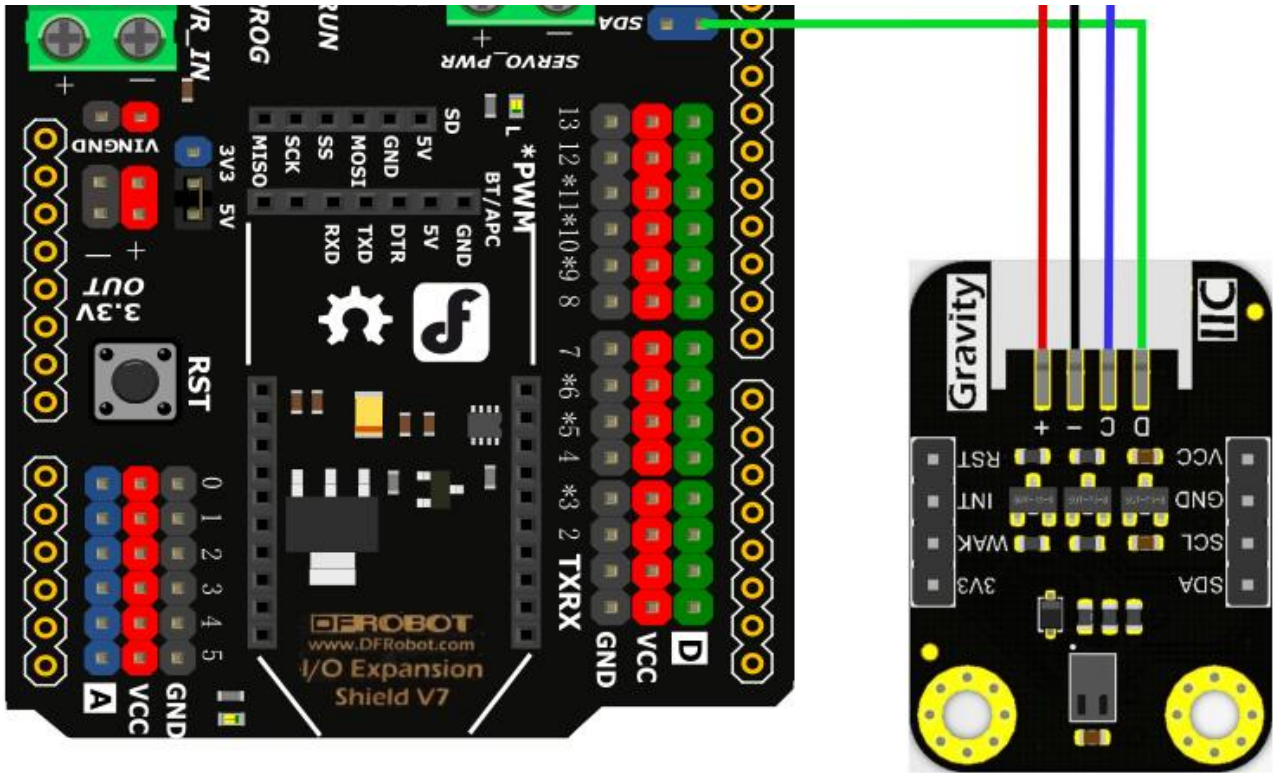
```

```
*/
uint16_t readBaseLine();

/**
 *@brief write a baseline number into register
 *@param a Hexadecimal number get from getBaseLine.ino
 */
void writeBaseLine(uint16_t baseLine);
```

## Connection Diagram





## Sample Code

1 #include <Wire.h>

## 1. Get Baseline

Why should we get the baseline? Because once we get it, we can show the air quality quickly after the sensor warm-up by inputting the baseline. Otherwise, it will cost a long time to read correctly when startup in polluted air.

During the first week of running the sensor, it is recommended to save a new baseline every 24 hours. After 1 week of operation, it can be saved every 1-28 days



### NOTE:

- Please place it in a fresh air environment (20 minutes or more) to obtain the baseline.
- Different sensors, different measurement cycles have different baselines.

```
/*!
 * @file getBaseLine.ino
 * @brief Put the module in clear air and work a few minutes, wait for baseline doing not
 * @n Experiment phenomenon: get
 *
 * @copyright Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @licence The MIT License (MIT)
 * @author [LuoYufeng](yufeng.luo@dfrobot.com)
 * @version V0.1
 * @date 2019-07-19
 * @get from https://www.dfrobot.com
 * @url https://github.com/DFRobot/DFRobot\_CCS811
 */
#include "DFRobot_CCS811.h"

/*!
 * IIC address default 0x5A, the address becomes 0x5B if the ADDR_SEL is soldered.
 */
```

```

//DFRobot_CCS811 CCS811(&Wire, /*IIC_ADDRESS=*/0x5A);
DFRobot_CCS811 CCS811;

void setup(void)

{
    Serial.begin(115200);
    /*Wait for the chip to be initialized completely, and then exit*/
    while(CCS811.begin() != 0){
        Serial.println("failed to init chip, please check if the chip connection is fine")
        delay(1000);
    }
}

void loop() {
    if(CCS811.checkDataReady() == true){
        /*!
        * @brief Set baseline
        * @return baseline in clear air
        */
        Serial.println(CCS811.readBaseLine(), HEX);

    } else {
        Serial.println("Data is not ready!");
    }
}

```

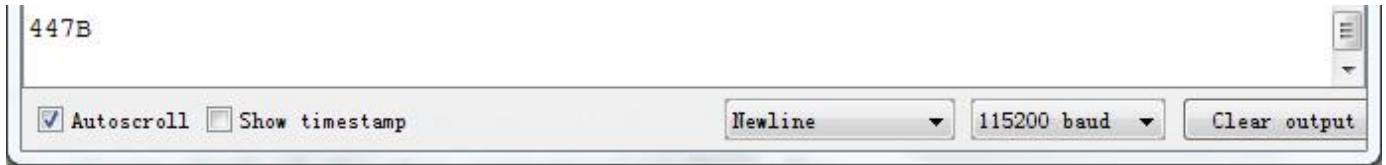


```
//delay cannot be less than measurement cycle  
delay(1000);  
}
```

## Expected Results

After a while, the baseline reaches to a stable value.





## 2. Get Data

Input the baseline value to the function `sensor.writeBaseLine()`; If you don't want to set the baseline, please disable this function in the sample program. The sensor will automatically calibrate the baseline, but it would be a pretty long process. after the function uploaded to UNO, open the serial port monitor to check the carbide dioxide concentration and TVOC concentration.

```
/*!
 * @file readData.ino
 * @brief Read the concentration of carbon dioxide and TVOC
 * @n Experiment phenomenon: read data every 0.5s, and print it out on serial port.
 *
 * @copyright Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @licence The MIT License (MIT)
 * @author [LuoYufeng](yufeng.luo@dfrobot.com)
 * @version V0.1
 * @date 2019-07-19
 * @get from https://www.dfrobot.com
 * @url https://github.com/DFRobot/DFRobot\_CCS811
 */
#include "DFRobot_CCS811.h"

/*!
 * IIC address default 0x5A, the address becomes 0x5B if the ADDR_SEL is soldered.
 */
//DFRobot_CCS811_CCS811(0x5A) //IIC ADDRESS=*(0x5A).
```





```
CO2: 872ppm, TVOC: 71ppb
CO2: 855ppm, TVOC: 69ppb
CO2: 855ppm, TVOC: 69ppb
```

Autoscroll  
 Show timestamp  
Newline  
115200 baud  
Clear output

### 3. Concentration Alarm

Input the baseline value to the function `sensor.writeBaseLine()`; Upload it to UNO, when the CO2 concentration moves from the current range (low, medium, high) to another range (more than 50 ppm), an interruption is generated and the current CO2 value will be printed.

**NOTE:** This example requires the INT pin of the sensor to be connected to the corresponding interrupt pin on the main board (in the sample, D2 of UNO is selected).

#### AVR Series Interrupt Pin and Interrupt Number

	328 Mainboards: Uno, Nano, Mini...	Mega2560	32u4 Mainboards: Leonardo...
Interrupt Pin	D2, D3	D2, D3, D21, D20, D19, D18	D3, D2, D0, D1, D7
Interrupt	0 1	0 1 2 3 4 5	0 1 2 3 4

Number	0, 1 328 Mainboards: Uno,	0, 1, 2, 3, 4, 5 Mega2560	0, 1, 2, 3, 4 32u4 Mainboards:
--------	------------------------------	------------------------------	-----------------------------------

```

/!*
 * @file setInterrupt.ino
 * @brief Set interrupt parameter, when CO2 concentration range changes, get an interrupt
 * @n Experiment phenomenon: read data every 1s, and print it out on serial port.
 *
 * @copyright Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @licence The MIT License (MIT)
 * @author [LuoYufeng](yufeng.luo@dfrobot.com)
 * @version V1.0
 * @date 2019-07-13
 * @get from https://www.dfrobot.com
 * @url https://github.com/DFRobot/DFRobot\_Sensor
 */
#include "DFRobot_CCS811.h"

volatile int8_t GPIO1TRIG = 0;

/!*
 * IIC address default 0x5A, the address becomes 0x5B if the ADDR_SEL is soldered

```

```

    * IIC address default 0x5A, the address becomes 0x5B if the ADDR_SEL is soldered.
    */
//DFRobot_CCS811 CCS811(&Wire, /*IIC_ADDRESS=*/0x5A);
DFRobot_CCS811 CCS811;

void setup(void)
{
    Serial.begin(115200);
    /*wait for the chip to be initialized completely, and then exit*/
    while(CCS811.begin() != 0){
        Serial.println("failed to init chip, please check if the chip connection is fine")
        delay(1000);
    }
    attachInterrupt(0, interrupt, RISING);
    /**
    * @brief Measurement parameter configuration
    * @param mode:in typedef enum{
    *         eClosed,      //Idle (Measurements are disabled in this mode)
    *         eCycle_1s,    //Constant power mode, IAQ measurement every second
    *         eCycle_10s,   //Pulse heating mode IAQ measurement every 10 seconds
    *         eCycle_60s,   //Low power pulse heating mode IAQ measurement every 60
    *         eCycle_250ms //Constant power mode, sensor measurement every 250ms }
    *         }eCycle_t;
    * @param thresh:0 for Interrupt mode operates normally; 1 for interrupt mode only as
    * @param thresh:0 for Interrupt mode operates normally; 1 for interrupt mode only as

```



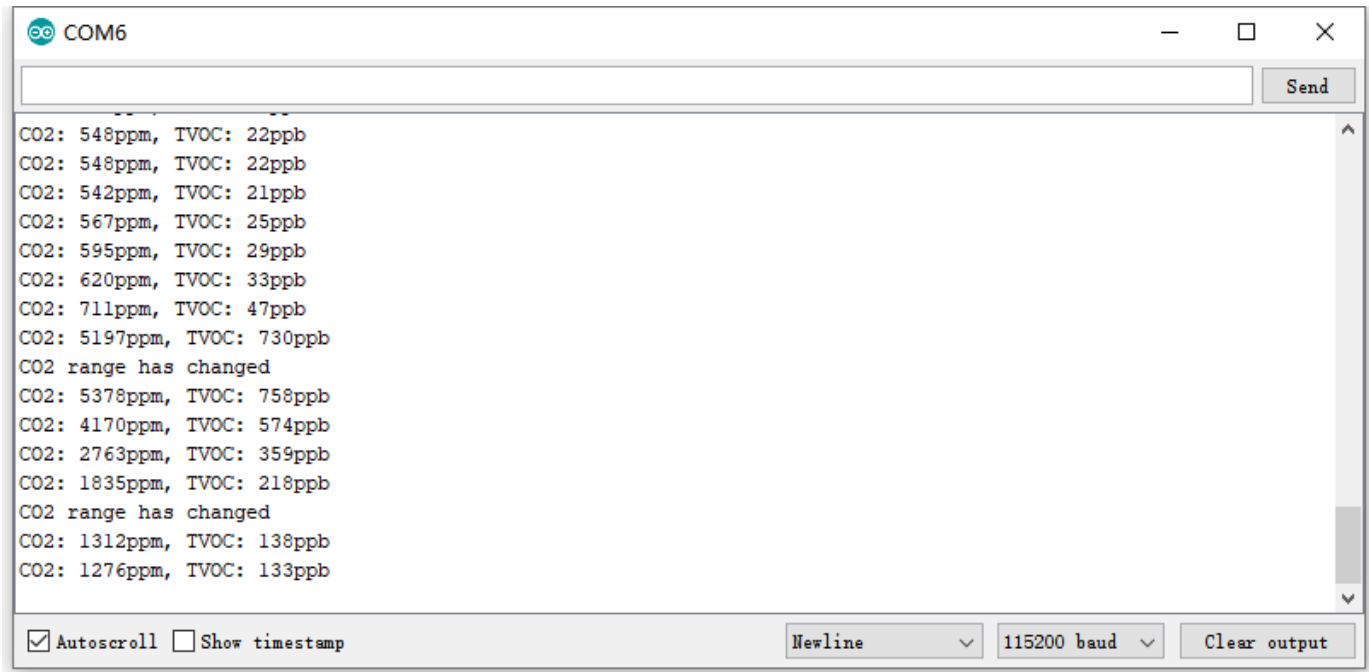
```

    ^ @param interrupt:0 for interrupt generation is disabled; 1 for the nLNI signal is a
    */
    CCS811.setMeasurementMode(CCS811.eCycle_250ms, 1, 1);
    /**
    * @brief Set interrupt thresholds
    * @param lowToMed: interrupt triggered value in range low to middle
    * @param medToHigh: interrupt triggered value in range middle to high
    */
    CCS811.setThresholds(1500,2500);
}
void loop() {
    if(GPIO1TRIG == 1){
        Serial.println("CO2 range has changed");
        Serial.print("CO2: ");
        Serial.print(CCS811.getCO2PPM());
        Serial.print("ppm, TVOC: ");
        Serial.print(CCS811.getTVOCPPB());
        Serial.println("ppb");
        delay(1000);
    }
    GPIO1TRIG = 0;
    Serial.print("CO2: ");
    Serial.print(CCS811.getCO2PPM());
    Serial.print("ppm, TVOC: ");
    Serial.print(CCS811.getTVOCPPB());
}

```

```
Serial.print(CCS811.getIvocPPB());  
Serial.println("ppb");  
CCS811.writeBaseLine(0x447B);  
delay(1000);  
}  
  
void interrupt(){  
  GPIO1TRIG = 1;  
}
```

## Expected Results




More Documents

## MORE DOCUMENTS

---

- Schematics  
(<https://dfimg.dfrobot.com/nobody/wiki/83f29407c31b407e4a220724652dd2fb.pdf>)
- Dimension  
(<https://dfimg.dfrobot.com/nobody/wiki/551fdcbfd429f0451ed59bf55b752034.pdf>)
- Datasheet  
(<https://dfimg.dfrobot.com/nobody/wiki/7334c560756596ba0cf3f1d2102d19dd.pdf>)
- CCS811 Baseline  
(<https://dfimg.dfrobot.com/nobody/wiki/ab83d61ed52c66c2dd4067eed25b0c35.pdf>)

 Get **Gravity: CCS811 Air Quality Sensor** (<https://www.dfrobot.com/product-1981.html>) from DFRobot Store or **DFRobot Distributor**. (<https://www.dfrobot.com/index.php?route=information/distributorslogo>)

**Turn to the Top**