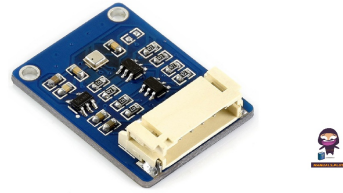


Chip Dip BME280 Environmental Sensor



# Chip Dip BME280 Environmental Sensor User Manual

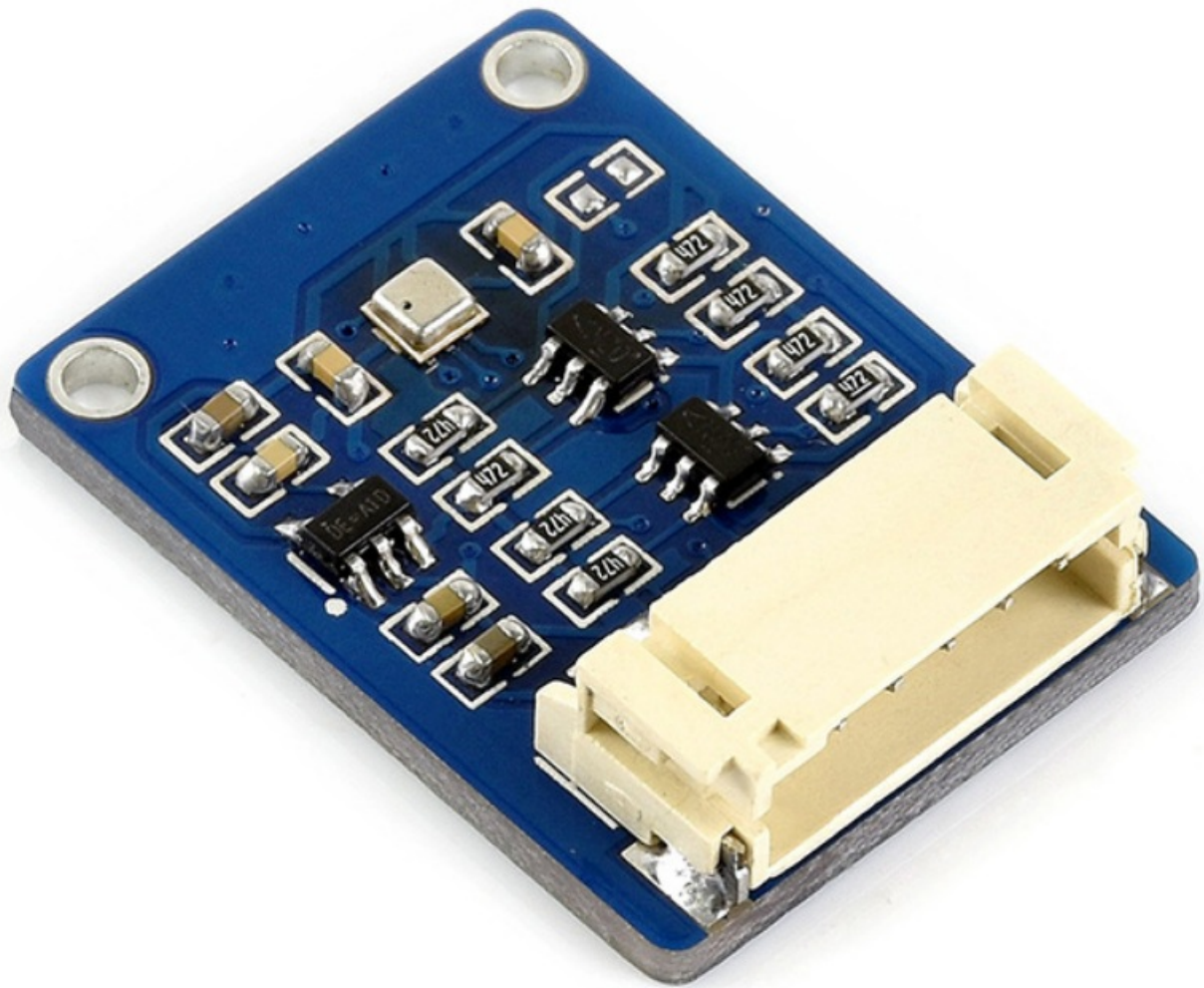
[Home](#) » [Chip Dip](#) » Chip Dip BME280 Environmental Sensor User Manual 

## Contents

- 1 Chip Dip BME280 Environmental Sensor
- 2 FAQs
- 3 Models
- 4 Introduction And Feature
- 5 Specifications
- 6 Interface Definition
- 7 Working with Raspberry Pi
- 8 Software Config
- 9 Working with Arduino
  - 9.1 Demo
- 10 Working with Raspberry Pi Pico
- 11 Working with ESP32
- 12 Resource
- 13 Support
- 14 Documents / Resources
  - 14.1 References
- 15 Related Posts

# Chip Dip

Chip Dip BME280 Environmental Sensor



## FAQs

- **Q:** What is the IAQ measuring range of the sensor?
  - **A:** The IAQ measuring range is from 0 to 500 IAQ. The sensor outputs changes in resistance due to VOC gas, and it requires the Bosch BSEC library for IAQ output.
- **Q:** How can I configure the I2C address of the sensor?
  - **A:** The I2C address is configurable by connecting the ADDR pin to either GND or leaving it non-connected. When connected to GND, the address is 0x76; otherwise, it is 0x77 by default.

## Models

## BME680 Environmental Sensor



## BME688 Environmental Sensor



I2C, SPI

### Introduction And Feature

#### Introduction

The BME68X Environmental Sensor is a four-in-one environmental sensor that can measure temperature, humidity, barometric pressure, and air quality. It is compact, low power, and suitable for smart homes, mobile application environment monitoring, wearable devices, etc.

#### Feature

- Onboard BME68X sensor to measure temperature, humidity, barometric pressure, and gas.
- Supports I2C communication, I2C address configurable, with I2C bus cascading support.
- Supports SPI communication, enabled via CS pin (I2C bus by default).
- Onboard voltage translator, compatible with 3.3V/5V level.
- Comes with online development resources and manual (examples for Raspberry Pi / Raspberry Pi Pico / Arduino / ESP32).

#### Specifications

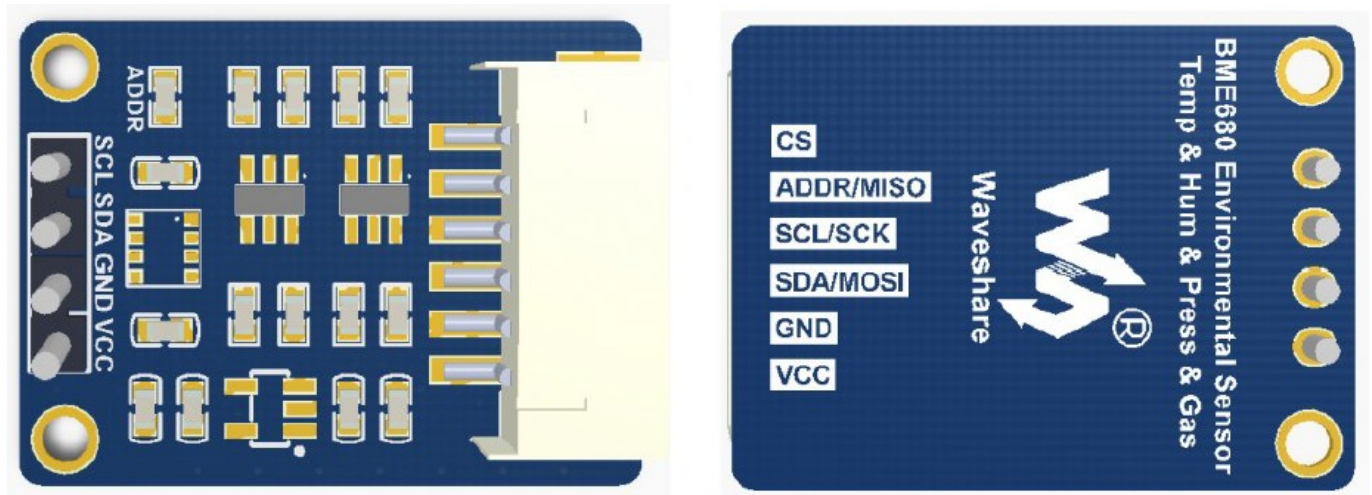
Model	BME280	BME680	BME688
Function	Barometric pressure, Environmental temperature, Relative humidity	Barometric pressure, Environmental temperature, Relative humidity, VOC gas change detection (supports IAQ calculation in combination with the software package)	Similar to BME680,  Suitable for detecting various additional gases (such as VSC, carbon monoxide, hydrogen, etc.) Multiple gas discrimination Artificial intelligence (requires secondary development by the user)
Communication Interface	I2C and SPI		
Temperature Measuring Range	-40~85°C		
Temperature Measuring Accuracy	±1.0°C (0~65°C)		±0.5°C (0~65°C)
Humidity Measuring Range	0~100% r.H.		
Humidity Measuring Accuracy	±3% r.H.		
Barometric Pressure Measurement Range	300~1100 hPa		

Barometric Pressure Measurement Accuracy	±1.0hPa (0~65°C)	±0.6hPa (0~65°C)
IAQ Measuring Range	Not support	0~500 IAQ  (The sensor outputs changes in resistance due to VOC gas, and the Bosch BSEC library is required to output IAQ.)
Dimensions	27mm × 20mm	

## Warning

The BME680 and BME688 sensors contain a mini MOX sensor. The heated metal oxide changes its resistance according to the concentration of volatile organic compounds (VOC) in the air, making it capable of detecting gases and alcohols such as ethanol, alcohol, and carbon monoxide, and measuring air quality. It provides a resistance value (Gas resistance in the figure), which represents the total VOC content, but cannot differentiate between different gases or alcohols. To convert this value to an IAQ air quality index, it is necessary to use the official BSEC software library (which is not open source). Bosch imposes certain restrictions and licensing requirements on the use of this software library, and users are advised to study the details of its use and integration according to their specific needs.

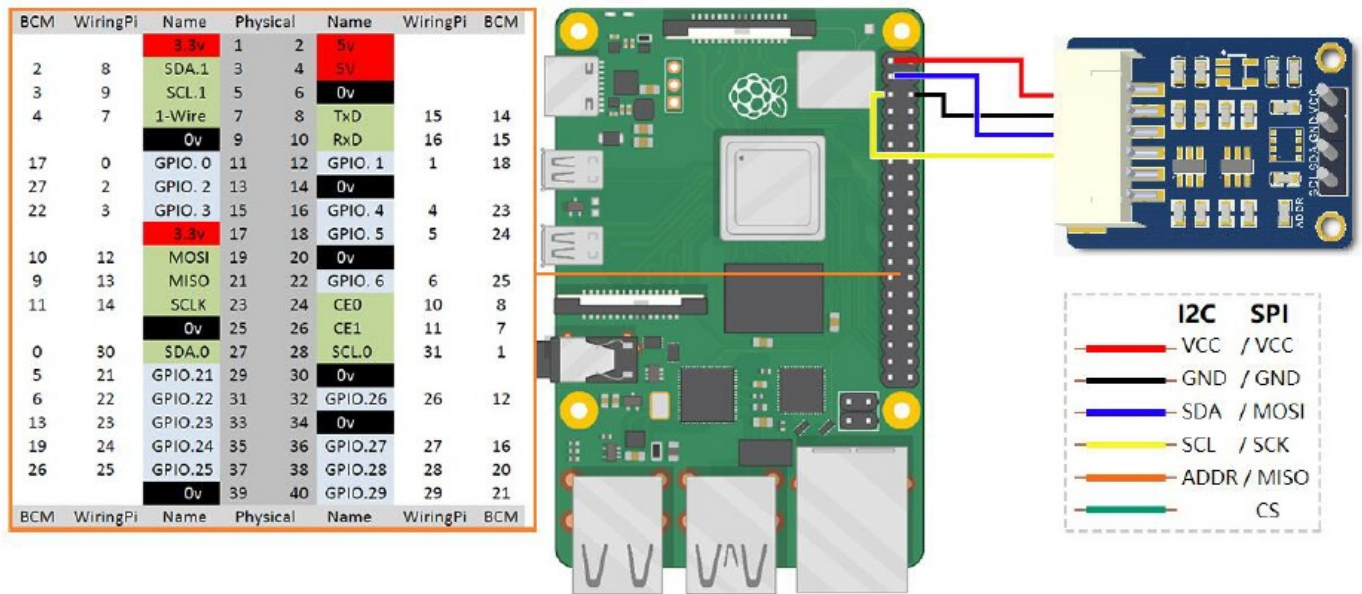
## Interface Definition



I2C		SPI	
Pins	Description	Pins	Description
VCC	Power Input	VCC	Power Input
GND	Ground	GND	Ground
SDA	Data Pin	MOSI	SPI Data Input
SCL	I2C Clock Pin	SCK	SPI Clock Input
ADDR	Address chip selection (high level by default): high level, the address is 0x77 low level, the address is 0x76	MISO	SPI data output
CS	NC	CS	SPI chip selection, low active

## Working with Raspberry Pi

### Hardware Connection



The above figure is connected to the I2C interface as an example as a demonstration, where the ADDR pin can be used to set the I2C address of the sensor, the default nonconnected I2C address is 0x77, if the ADDR is connected to GND, the I2C address is 0x76. If you want to connect Raspberry Pi through the SPI interface for communication, please refer to the following table for connection.

I2C		SPI	
Pins	Raspberry Pin	Pins	Raspberry Pin
VCC	3.3V /5V	VCC	3.3V /5V
GND	GND	GND	GND
SDA	SDA.1	MOSI	MOSI
SCL	SCL.1	SCK	SCLK
ADDR	NC/GND	MISO	MISO
CS	NC	CS	27(wiringPi)

## Software Config

### Enable I2C/SPI Interface

- Execute the following commands to configure the Raspberry Pi:
  - `sudo raspi-config`
- Choose Interfacing Options -> I2C -> yes to enable I2C kernel driver.
- Choose Interfacing Options -> SPI -> yes to enable SPI kernel driver.
- Save, exit, and then reboot the Raspberry Pi:
  - `sudo reboot`
- After rebooting, run the commands to view. Check whether the I2C and SPI modules are enabled.
  - `lsmod`
- The following print message will be available.



```
pi@raspberrypi: ~
文件(F) 编辑(E) 标签(T) 帮助(H)
pi@raspberrypi:~$ lsmod
Module                  Size      Used by
bnep                    12051     2
hci_uart                20020     1
btbcm                   7916      1 hci_uart
bluetooth               365780    22 hci_uart,bnep,btbcm
rtc_ds1307              13908     0
hwmon                   10552     1 rtc_ds1307
brcmfmac                289942     0
brcmutil                9863      1 brcmfmac
sg                       20781     0
spidev                  7373      0
cfg80211                543219     1 brcmfmac
rfkill                  20851     4 bluetooth,cfg80211
snd_bcm2835             24427     1
snd_pcm                 98501     1 snd_bcm2835
snd_timer               23968     1 snd_pcm
snd                      70032     5 snd_timer,snd_bcm2835,snd_pcm
i2c_bcm2835             7167      0
spi_bcm2835             7596      0
bcm2835_gpiomem         3940      0
w1_gpio                 4818      0
wire                     32619     1 w1_gpio
cn                       5889      1 wire
lirc_rpi                 9032      0
uio_pdrv_genirq         3923      0
lirc_dev                10583     1 lirc_rpi
uio                      10204     1 uio_pdrv_genirq
```

- If i2c\_bcm2835 and spi\_bcm2835 are displayed then the I2C, SPI module is booted.
- Connect the BME68x module to the Raspberry Pi as described in the previous I2C bus interface instructions.
- The default I2C device address of the BME68x module is 0x77, if ADDR is grounded, the device address will be changed to 0x76.
- Install the i2c-tools tool to confirm.
  - `sudo apt-get install i2c-tools`
- Query connected I2C devices
  - `i2cdetect -y 1`
- The following message will be printed.

```
pi@raspberrypi:~$ i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- 77
```


- If 77 is displayed then the BME68x module is successfully connected to the Raspberry Pi successfully.
- If the ADDR is connected to GND then 76 is printed.

```
pi@raspberrypi:~$ i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 76 -- -- -- -- -- -- -- --
```

**Note:** The above test ensures that there are no devices on the I2C bus that have the same address as the device. If the above test is successful, the I2C module is loaded successfully, and the BME68x module is successfully connected to the Raspberry Pi. In addition, the BME68x module supports the SPI driver, and you can refer to the SPI interface description section to connect the BME68x to the Raspberry Pi.

## Download Example Demo

- Download the [example demo](#) , decompress, and modify the file permissions.
  - `cd ~`
  - `wget https://files.waveshare.com/upload/4/49/BME68X\_Environmental\_Sensor\_code.zip`
  - `unzip BME68X_Environmental_Sensor_code.zip`
  - `sudo chmod -R 777 BME68X_Environmental_Sensor_code`

## C

### Demo

- After connecting the hardware as shown above and configuring the software properly.
- If I2C driver is used: first determine the I2C device address, BME68x module default I2C device address is 0x77, if the ADDR pin is grounded (or short the pad marked **ADDR** silkscreen on the PCB), then its I2C device address changes to 0x76.
- Enter BME68X\_Environmental\_Sensor\_code/RaspberryPi/C:
  - `cd BME68X_Environmental_Sensor_code/RaspberryPi/C`
- Open main.c file:
  - `nano main.c`

```
19 //Default write it to the register in one time
20 #define USESPISINGLEREADWRITE 0
21
22 //This definition you use I2C or SPI to drive the bme68x
23 //When it is 1 means use I2C interface, When it is 0, use SPI interface
24 #define USEIIC 1
25
26 #define BME68X_VALID_DATA UINT8_C(0xB0)
```

- Make sure the USEIIC macro in main.c is defined as 1 to adopt the I2C driver.



```

235  #if(USEIIC)
236  int main(int argc, char* argv[])
237  {
238      struct bme68x_dev dev;
239      static uint8_t dev_addr=BME68X_I2C_ADDR_HIGH;
240      int8_t rslt = BME68X_OK;
241
242      if((fd = open(IIC_Dev, O_RDWR)) < 0) {
243          printf("Failed to open the i2c bus %s", argv[1]);
244          exit(1);
245      }
246      if(ioctl(fd, I2C_SLAVE, dev_addr) < 0) {
247          printf("Failed to acquire bus access and/or talk to slave.\n");
248          exit(1);
249      }
250      //dev.dev_id = BME68X_I2C_ADDR_PRIM; //0x76
251      dev.intf_ptr = &dev_addr; //0x77
252      dev.intf = BME68X_I2C_INTF;
253      dev.read = user_i2c_read;
254      dev.write = user_i2c_write;
255      dev.delay_us = user_delay_us;

```

- Also check the I2C device address in main.c to make sure it is the same as the current BME68x module device address (default I2C device address is 0x77 (BME68X\_I2C\_ADDR\_HIGH). If ADDR is grounded then its device address is 0x76 (BME68X\_I2C\_ADDR\_HIGH)).

```

19  //Default write it to the register in one time
20  #define USESPISINGLEREADWRITE 0
21
22  //This definition you use I2C or SPI to drive the bme68x
23  //When it is 1 means use I2C interface, When it is 0, use SPI interface
24  #define USEIIC 0
25
26  #define BME68X_VALID_DATA UINT8_C(0xB0)

```

- If SPI driver is used: wire the BME68x module according to the SPI bus wiring in the interface description and change the USEIIC macro definition in the main.c file to 0.
- Save and exit the editor, then recompile.
  - sudo make clean
  - sudo make
- Run:
  - sudo ./bme68x
- The following data will be displayed.

```

pi@raspberrypi:~/Raspberry $ sudo ./bme68x

BME68X Init Result is:0
Temperature      Pressure      Humidity      Gas resistance
temperature:25.06°C  pressure:1022.93hPa  humidity:35.91%  Gas resistance:143085.09 ohmm

```

- From left to right, the temperature (°C), barometric pressure (hPa), relative humidity (%RH), and gas resistance (ohms) measured by the BME68x are displayed. If the data is not displayed successfully, or if the data is not displayed properly, please check the connection, communication method, and device address for errors.

## Python

- Python demo only has I2C mode.

## Install Function Library

- `sudo pip3 install bme680`

## Demo

- Enter the example demo file:
  - `cd BME68X_Environmental_Sensor_code/RaspberryPi/Python/examples`
- Run the demo:
  - `sudo python3 read-all.py`
- The demo will print a series of module information, from left to right, the temperature (°C), barometric pressure (hPa), relative humidity (%RH), and gas resistance (ohms) measured by the BME68x are displayed. If the data is not displayed successfully, or if the data is not displayed properly, please check the connection, the communication method, and the device address for errors.

```
pi@uuuuuu:~/bme680-python-master/bme680-python-master/examples $ sudo python read-all.py
read-all.py - Displays temperature, pressure, humidity, and gas.

Press Ctrl+C to exit!

Calibration data:
par_gh1: -45
par_gh2: -18303
par_gh3: 18
par_h1: 709
par_h2: 1023
par_h3: 0
par_h4: 45
par_h5: 20
par_h6: 120
par_h7: -100
par_p1: 36169
par_p10: 30
par_p2: -10398
par_p3: 88
par_p4: 7877
par_p5: -172
par_p6: 30
par_p7: 42
par_p8: -2607
par_p9: -2613
par_t1: 26058
par_t2: 26363
par_t3: 3
range_sw_err: 13
res_heat_range: 1
res_heat_val: 33
t_fine: 129387

Initial reading:
gas_index: 0
gas_resistance: 55858512 678457275
```

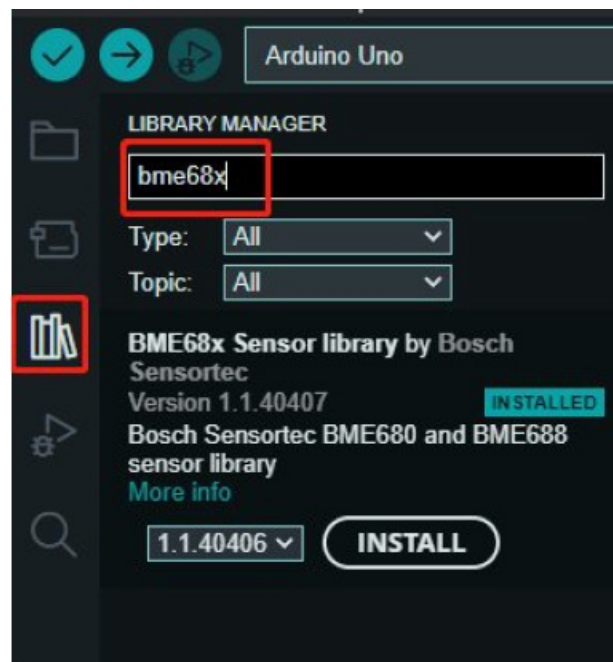
```
gas_resistance: 5585012.078457273
heat_stable: False
humidity: 64.188
meas_index: 0
pressure: 1010.66
status: 32
temperature: 25.27

Polling:
25.28 C,1010.66 hPa,64.19 %RH
25.30 C,1010.69 hPa,64.06 %RH,32837.35248845562 Ohms
25.34 C,1010.68 hPa,63.84 %RH,51990.25182778229 Ohms
25.39 C,1010.70 hPa,63.62 %RH,68577.55156710421 Ohms
25.42 C,1010.69 hPa,63.39 %RH,82500.80567193039 Ohms
25.46 C,1010.70 hPa,63.18 %RH,94955.48961424333 Ohms
25.49 C,1010.69 hPa,63.04 %RH,103790.79667545104 Ohms
25.51 C,1010.67 hPa,62.91 %RH,110870.50671286273 Ohms
25.53 C,1010.67 hPa,62.82 %RH,116948.37825491093 Ohms
25.55 C,1010.67 hPa,62.75 %RH,122224.87467175937 Ohms
25.56 C,1010.71 hPa,62.66 %RH,127141.79289793893 Ohms
```

## Working with Arduino

### Install Library

The library for the BME68x sensor can be downloaded from the library manager of the Arduino IDE:



- Open Arduino IDE 2.0.
- Open the "Library Manager" option in the left toolbar and search for BME68x.

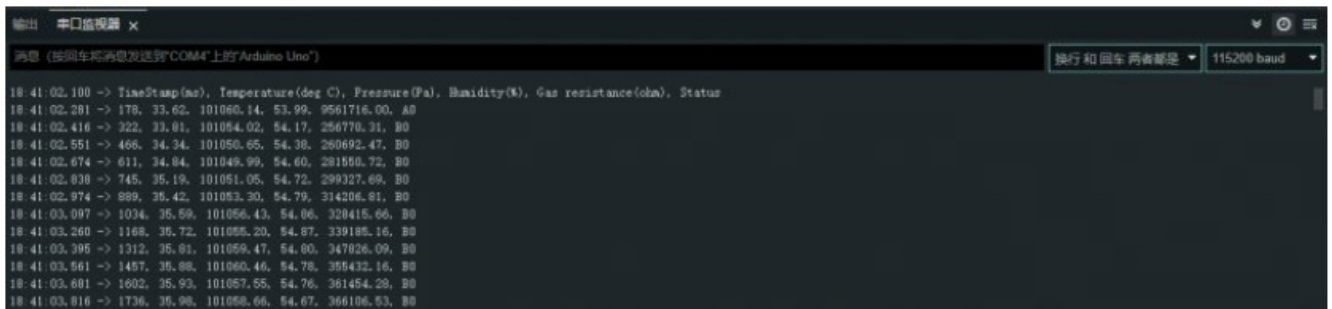
### Hardware Connection

I2C Interface		SPI Interface	
Pins	Arduino Pin	Pins	Arduino Pin
VCC	3.3V /5V	VCC	3.3V /5V
GND	GND	GND	GND
SDA	SDA	MOSI	D11
SCL	SCL	SCK	D13
ADDR	NC/GND	MISO	D12
CS	NC	CS	D10

## Demo

## SPI

- The default communication method of this demo is SPI, refer to the table above to connect the module to the development board (this demo uses Arduino Uno).
- Click File -> examples -> BME68x Sensor library -> forced\_mode to open the sample demo.
- Connect the development board to the computer (this demo uses Arduino uno), click
- Tools->Development Board, select the corresponding development board, click: Tools- >Port select the corresponding port.
- Click on the upload button to compile and upload the demo to see the development board and wait for a successful upload.



The screenshot shows the Arduino IDE Serial Monitor window. The title bar is '输出 串口监视器 x'. The message area says '消息 (按照时间和消息发送到"COM4"上的 Arduino Uno)'. The baud rate is set to '115200 baud'. The data being received is as follows:

```

18 41 02.100 -> TimeStamp(ms), Temperature(deg C), Pressure(hPa), Humidity(%RH), Gas resistance(ohms), Status
18 41 02.281 -> 178, 33.62, 101060.14, 53.99, 9561716.00, A0
18 41 02.416 -> 322, 33.81, 101054.02, 54.17, 256770.31, B0
18 41 02.551 -> 466, 34.34, 101050.65, 54.38, 260692.47, B0
18 41 02.674 -> 611, 34.84, 101049.99, 54.60, 281550.72, B0
18 41 02.838 -> 745, 35.19, 101051.05, 54.72, 299327.69, B0
18 41 02.974 -> 889, 35.42, 101053.30, 54.79, 314204.81, B0
18 41 03.097 -> 1034, 35.59, 101056.43, 54.86, 328415.66, B0
18 41 03.260 -> 1168, 35.72, 101055.20, 54.87, 339188.16, B0
18 41 03.395 -> 1312, 35.83, 101059.47, 54.80, 347826.09, B0
18 41 03.561 -> 1457, 35.86, 101060.46, 54.78, 359432.16, B0
18 41 03.681 -> 1602, 35.93, 101057.55, 54.76, 361454.28, B0
18 41 03.816 -> 1736, 35.98, 101058.66, 54.67, 366106.53, B0

```

- Click on Tools -> Serial Monitor, which shows from left to right the temperature (°C), barometric pressure (hPa), relative humidity (%RH), altitude (m), and gas resistance (ohms) measured by the BME68x sensor.
- If the data is not displayed successfully, or if the data is not displayed normally, please check the connection, communication method, and device address for errors.

## I2C

- If you want to change the communication way to I2C, you should modify the hardware connection according to the I2C.
- Modify the main demo according to the following figure.
- Compile and upload the demo, and open SSCOM. From left to right, the temperature (°C), barometric pressure (hPa), relative humidity (%RH), altitude (m), and gas resistance (ohms) measured by the BME68x sensor are shown.



The screenshot shows the Arduino IDE with the `bme68xLibrary.h` file open. The code defines the I2C address and initializes the sensor. The serial monitor shows the output of the `setup` function, which prints a timestamp and various sensor readings.

```
forced_mode.ino  bme68xLibrary.h
9  #include "bme68xLibrary.h"
10
11  #ifndef PIN_CS
12  #define PIN_CS SS
13  #endif
14
15  #ifndef ADD_I2C
16  #define ADD_I2C 0x77
17  #endif
18
19  Bme68x bme;
20
21  /**
22   * @brief Initializes the sensor and hardware settings
23   */
24  void setup(void)
25  {
26    Wire.begin(ADD_I2C);
27    //SPI.begin();
28    Serial.begin(115200);
29
30    while (!Serial)
31      delay(10);
32
33    /* initializes the sensor based on SPI library */
34    bme.begin(ADD_I2C, Wire);
35    //bme.begin(PIN_CS, SPI);
36    if(bme.checkStatus())
37    {
38      if (bme.checkStatus() == BME68X_ERROR)
39      {
40      }
```

输出 串口监视器 x

消息 (按回车将消息发送到"COM4"上的"Arduino Uno") 换行和回车两者都是 115200 baud

```
18:44:53.383 -> TimeStamp(ms), Temperature(deg C), Pressure(Pa), Humidity(%), Gas resistance(ohm), Status
18:44:53.518 -> 172, 30.78, 101047.77, 50.00, 8088867.00, A0
18:44:53.700 -> 335, 30.96, 101045.18, 50.14, 227656.73, B0
18:44:53.836 -> 483, 31.49, 101041.09, 50.35, 240319.17, B0
18:44:53.973 -> 619, 31.99, 101042.46, 50.52, 261892.58, B0
18:44:54.096 -> 766, 32.34, 101042.99, 50.67, 277883.31, B0
18:44:54.274 -> 904, 32.58, 101045.72, 50.76, 290909.09, B0
18:44:54.409 -> 1051, 32.75, 101046.45, 50.81, 301176.47, B0
18:44:54.547 -> 1187, 32.87, 101047.50, 50.86, 311625.06, B0
18:44:54.666 -> 1335, 32.96, 101047.67, 50.89, 321003.13, B0
18:44:54.847 -> 1471, 33.05, 101049.98, 50.87, 324667.09, B0
18:44:54.980 -> 1608, 33.11, 101050.91, 50.85, 330642.56, B0
18:44:55.116 -> 1756, 33.14, 101049.41, 50.86, 337174.84, B0
```

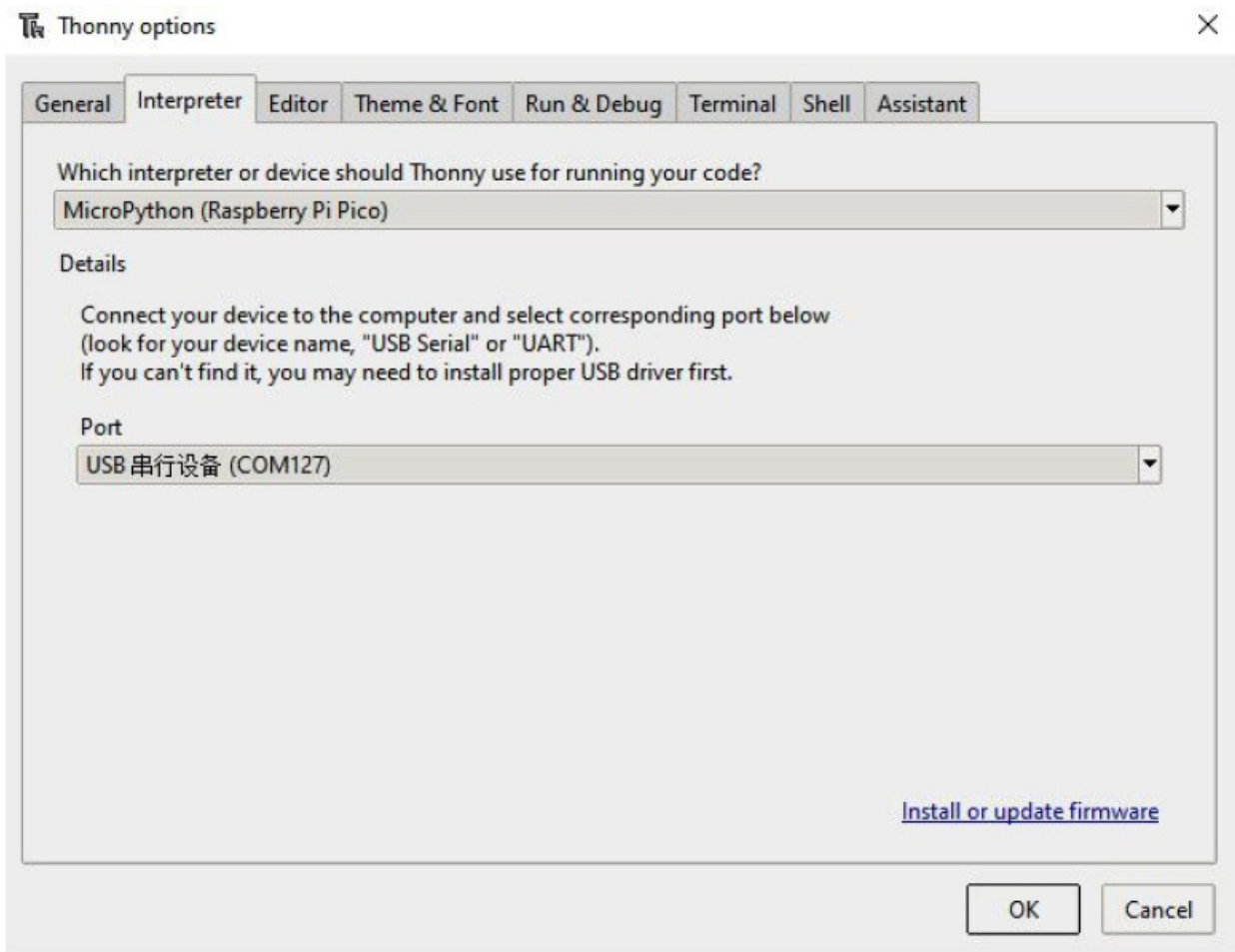
## Working with Raspberry Pi Pico

### Set up Environment

This tutorial uses Thonny for code testing, click to download the relevant IDE and install it, then open Thonny.

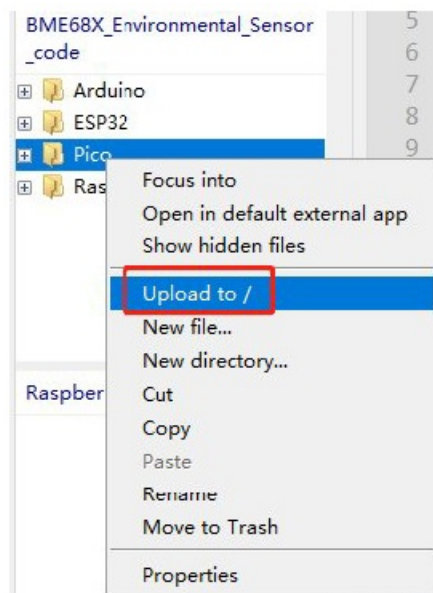
- Please refer to the official documentation to set up the python environment, in Thonny: Tools -> Options -> Interpret select the Raspberry Pi Pico device, as shown in the following figure:





## Download the Demo

1. Download the demo.
2. Unzip the sample demo.



3. Open Thonny, and check whether it is connected to the pico. Then, open the unpacked demo path in the upper left corner, right-click on the pico folder, and select Upload, as shown in the picture.

## Hardware Connection

I2C Interface		
Pins	Pico Pin	
VCC	3.3V /5V	
GND	GND	
SDA	GP6	
SCL	GP7	
ADDR	NC/GND	
CS	NC	

## Demo

1. Open Thonny IDE, choose the pico directory, and double-click to open the read-all.py file. The demo is shown below:

The screenshot shows the Thonny IDE interface. The left sidebar displays the file explorer with the 'Pico' directory selected, showing 'bme680' and 'read-all.py'. The main editor window displays the code for 'read-all.py'. The code imports the 'bme680' module and 'time' module, then prints a message and prompts the user to press Ctrl+C to exit. It then attempts to initialize the BME680 sensor. The bottom panel shows the output of the program, which displays a series of sensor readings (temperature, pressure, humidity, and gas) over time.

```

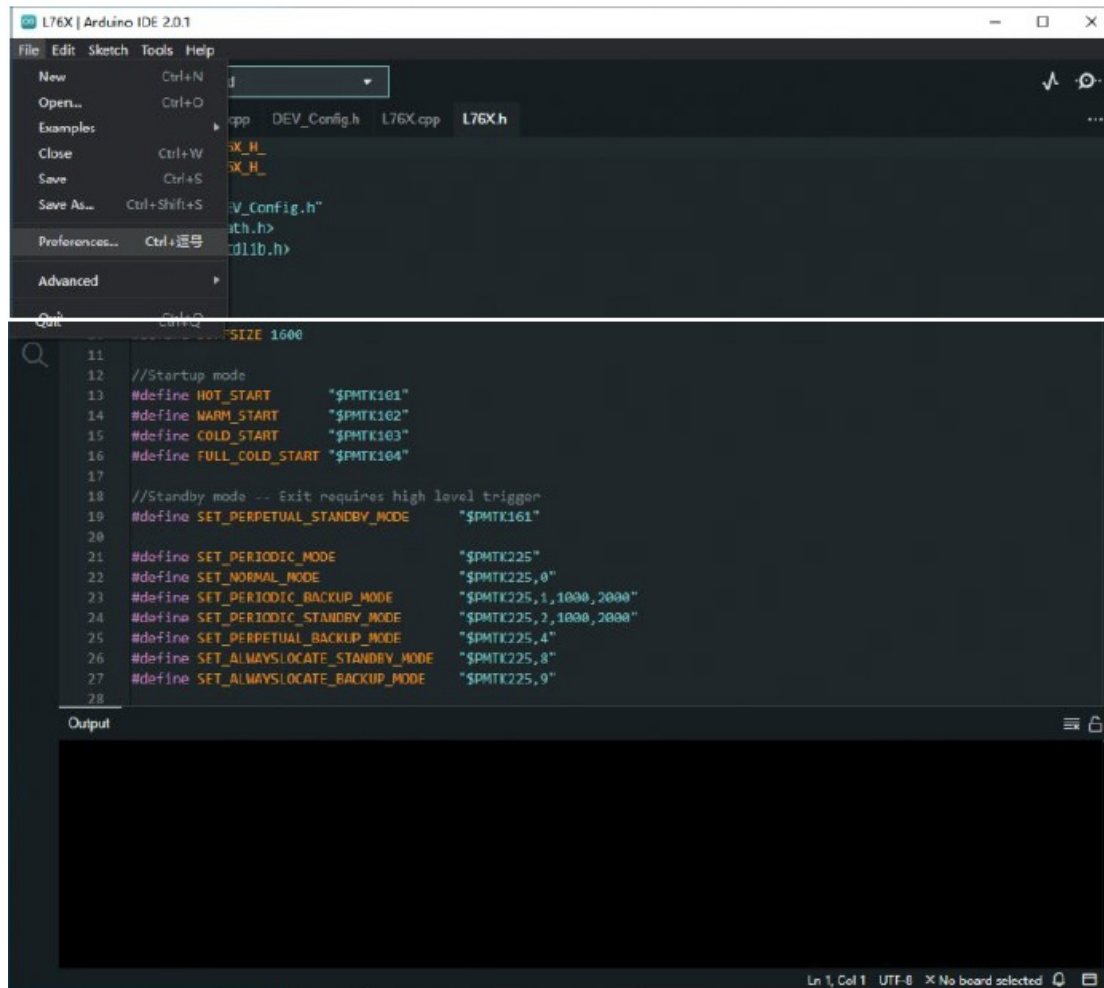
1 #!/usr/bin/env python
2
3 import bme680
4 import time
5
6 print("""read-all.py - Displays temperature, pressure, humidity, and gas.
7
8 Press Ctrl+C to exit!
9
10 """)
11
12 try:
13     sensor = bme680.BME680(bme680.I2C_ADDR_PRIMARY)
14 except (RuntimeError, OSError):
15     sensor = bme680.BME680(bme680.I2C_ADDR_SECONDARY)
16
17 # These calibration data can safely be commented
18 # out, if desired.
19
20 print('Calibration data:')
21
22 gas_index = 0
23 meas_index = 0
24
25 Polling:
26 27.18 C,1019.77 hPa,39.03 %RH
27 27.20 C,1019.77 hPa,39.00 %RH,95040.07 Ohms
28 27.25 C,1019.77 hPa,39.09 %RH,98434.34 Ohms
29 27.31 C,1019.76 hPa,39.05 %RH,101640.0 Ohms
30 27.36 C,1019.81 hPa,39.01 %RH,104450.9 Ohms
31 27.39 C,1019.79 hPa,38.95 %RH,107251.5 Ohms
32 27.42 C,1019.80 hPa,38.92 %RH,109559.6 Ohms

```

## Working with ESP32

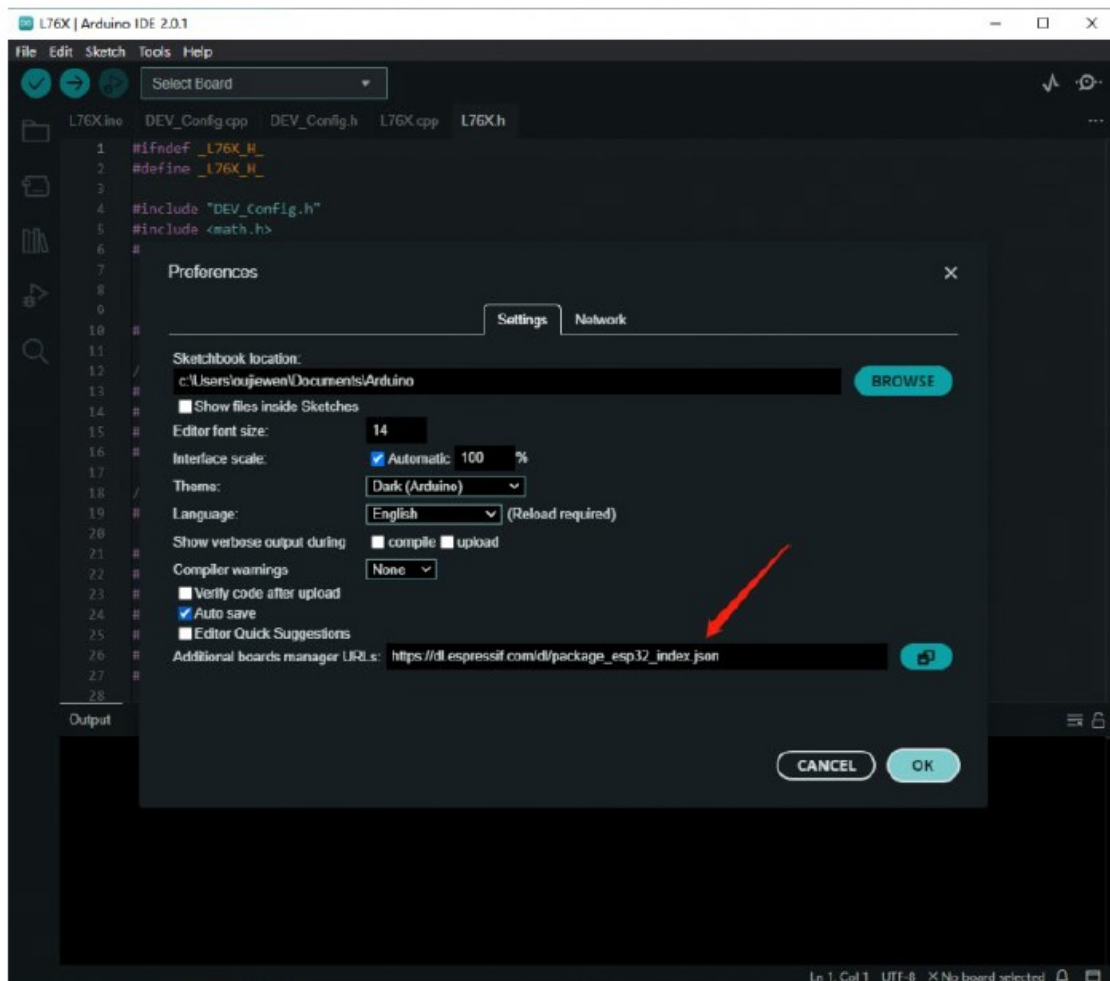
### Install ESP32 Plug-in in Arduino IDE

1. Open Arduino IDE, click “File” at the upper left corner, and choose “Preferences”.



2. Add the following link to the Additional Development Board Manager URL and click OK.

- [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)



- **Note:** If you already have the ESP8266 board URL, you can separate the URLs with commas like this:
  - [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json) [http://arduino.esp8266.com/stable/package\\_esp8266\\_index.json](http://arduino.esp8266.com/stable/package_esp8266_index.json)

3. Download the package and copy the packages file to the following path:

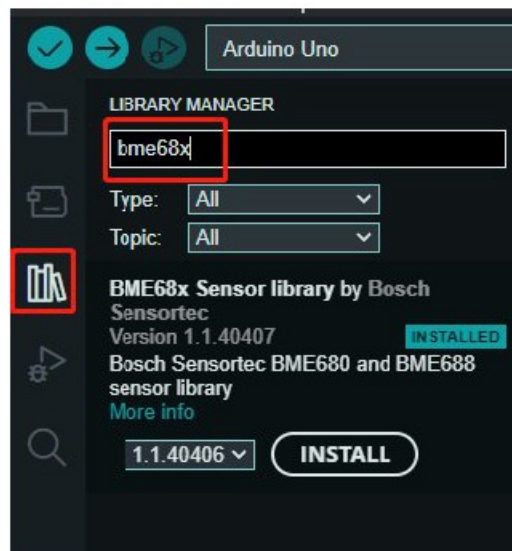
- C:\Users\xutong\AppData\Local\Arduino15



Note: Replace the username: **xutong** with your own username.

## Install Library

The library for the BME68x sensor can be downloaded from the library manager of the Arduino IDE:



- Open Arduino IDE 2.0.
- Open the “Library Manager” option in the left toolbar and search for BME68x.

## Hardware Connection

I2C Interface		SPI Interface	
Pins	ESP32 Pin	Pins	ESP32 Pin
VCC	3.3V /5V	VCC	3.3V /5V
GND	GND	GND	GND
SDA	P21	MOSI	P23
SCL	P22	SCK	P18
ADDR	NC/GND	MISO	P19
CS	NC	CS	P15

## Demo

### SPI

- The default communication method of this demo is SPI, refer to the table above to connect the module to the development board.
- Click on: File -> Examples -> BME68x Sensor library -> forced\_mode to open the sample demo.
- Connect the development board to the computer, click Tools->Development Board, select the corresponding development board, and click: Tools -> Port to select the corresponding port.
- Click the upload button to compile and upload the demo to the watch development board and wait for a successful upload.

```

forced_mode.ino
1  #include "bme68xlibrary.h"
2
3  #ifndef PIN_CS
4  #define PIN_CS 15
5  #endif
6
7  #ifndef ADDR_I2C
8  #define ADDR_I2C 0x77
9  #endif
10
11 BME68x bme;
12
13 /**
14  * @brief Initializes the sensor and hardware settings
15  */
16 void setup(void)
17 {
18     //Wire.begin();
19     SPI.begin();
20     Serial.begin(115200);
21
22     while (!Serial)
23         delay(10);
24
25     /* Initializes the sensor based on SPI library */
26     bme.begin(PIN_CS, SPI);
27     //bme.begin(ADDR_I2C, Wire);
28
29     if (bme.checkStatus())
30     {
31         if (bme.checkStatus() == BME68X_ERROR)
32         {
33             //Serial.println("BME68X Error: " + bme.getStatus());
34         }
35     }
36 }
37
38 void loop()
39 {
40     //Serial.println("Temperature: " + bme.getTemp());
41     //Serial.println("Humidity: " + bme.getHumidity());
42     //Serial.println("Pressure: " + bme.getPressure());
43     //Serial.println("Gas Resistance: " + bme.getGasResistance());
44     //Serial.println("Altitude: " + bme.getAltitude());
45 }

```

输出 串口监视器 x

勾选 (使用本消息发送到 COM11 上的 ESP32 Dev Module)

换行和 回车 两者都是 115200 baud

```

14:52:42.137 -> 3481444, 29.07, 101869.96, 32.95, 282911.76, B0
14:52:42.872 -> 3481591, 29.07, 101969.84, 32.98, 294494.44, B0
14:52:43.609 -> 3481725, 29.07, 101801.78, 32.97, 282911.76, B0
14:52:43.174 -> 3481876, 29.08, 101961.15, 33.00, 293431.84, B0
14:52:43.710 -> 3484011, 29.07, 101962.32, 32.99, 292654.47, B0
14:52:43.145 -> 3484145, 29.07, 101969.75, 32.97, 293964.40, B0
14:52:43.880 -> 3484294, 29.08, 101962.20, 32.98, 297131.37, B0
14:52:43.745 -> 3484438, 29.07, 101969.30, 32.97, 296331.31, B0
14:52:43.880 -> 3484584, 29.07, 101961.52, 32.98, 294741.00, B0
14:52:44.017 -> 3484718, 29.07, 101969.41, 32.97, 293174.59, B0
14:52:44.152 -> 3484863, 29.07, 101961.27, 32.98, 292174.50, B0
14:52:44.719 -> 3485008, 29.07, 101969.76, 32.97, 292911.76, B0
14:52:44.459 -> 3485142, 29.06, 101969.97, 32.97, 296634.10, B0

```

- Click on Tools -> Serial Monitor, which shows from left to right the temperature (°C), barometric pressure (hPa), relative humidity (%RH), altitude (m), and gas resistance (ohms) measured by the BME68x sensor.



- If the data is not displayed successfully, or if the data is not displayed properly, please check the connection, communication method, and device address for errors.

## I2C

- If you need to modify the communication mode to I2C, first modify the hardware connection according to the I2C mode.
- Refer to the following diagram, and modify the original main demo;

The screenshot shows the Arduino IDE with the 'forced\_mode.h' file open. The code defines I2C parameters and initializes the BME68x sensor. The serial monitor at the bottom displays a series of sensor readings.

```

forced_mode.h
9  #include "bme68xLibrary.h"
10
11  #ifndef PIN_CS
12  #define PIN_CS 15
13  #endif
14
15  #ifndef ADD_I2C
16  #define ADD_I2C 0x77
17  #endif
18
19  Bme68x bme;
20
21  /**
22   * @brief Initializes the sensor and hardware settings
23   */
24  void setup(void)
25  {
26    Wire.begin();
27    //SPI.begin();
28    Serial.begin(115200);
29
30    while (!Serial)
31      delay(10);
32
33    /* Initializes the sensor based on SPI library */
34    //bme.begin(PIN_CS, SPI);
35    bme.begin(ADD_I2C, Wire);
36
37    if (bme.checkStatus())
38    {
39      if (bme.checkStatus() == BME68X_ERROR)
40      {
41
42      }
43    }
44  }
  
```

串口监视器 (按回车键将内容发送到"COM11"上的"ESP32 Dev Module")

15:00:21.166 -> 274088, 29.54, 101944.12, 32.54, 284836.94, B1  
 15:00:21.303 -> 274231, 29.54, 101944.15, 32.54, 284924.00, B1  
 15:00:21.439 -> 274365, 29.55, 101943.16, 32.51, 284816.94, B1  
 15:00:21.567 -> 274509, 29.55, 101943.42, 32.54, 284991.26, B1  
 15:00:21.740 -> 274652, 29.54, 101941.70, 32.54, 284835.92, B1  
 15:00:21.876 -> 274796, 29.54, 101942.49, 32.47, 287816.12, B1  
 15:00:21.913 -> 274940, 29.55, 101943.42, 32.54, 284871.06, B1  
 15:00:21.148 -> 275074, 29.55, 101945.41, 32.48, 287816.12, B1  
 15:00:21.303 -> 275218, 29.55, 101944.41, 32.48, 284326.96, B1  
 15:00:21.450 -> 275362, 29.55, 101944.70, 32.48, 291365.72, B1  
 15:00:21.594 -> 275506, 29.55, 101943.47, 32.48, 284834.00, B1  
 15:00:21.720 -> 275649, 29.54, 101943.41, 32.47, 284326.10, B1  
 15:00:21.856 -> 275793, 29.54, 101942.45, 32.48, 287317.81, B1

- Compile and upload the demo, open the serial monitor, which from left to right shows the temperature (°C), barometric pressure (hPa), relative humidity (%RH), altitude (m), and gas resistance (ohms) measured by the BME68x sensor.

## Resource

### Document

- [Schematic](#)

### Demo

- [Example demo](#)

### Software

- [Arduino IDE](#)
- [SSCOM Serial Assistant](#)

## Related Resource

- [BME680 Datasheet](#)
- [BME688 Datasheet](#)


## Support

### Technical Support

- If you need technical support or have any feedback/review, please click the Submit Now button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.
- Working Time: 9 AM – 6 AM GMT+8 (Monday to Friday)

### [Submit Now](#)

## Documents / Resources

	<p><a href="#">Chip Dip BME280 Environmental Sensor</a> [pdf] User Manual BME280 Environmental Sensor, BME280, Environmental Sensor, Sensor</p>
--	---

## References

- [User Manual](#)

### [Manuals+](#), [Privacy Policy](#)

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.