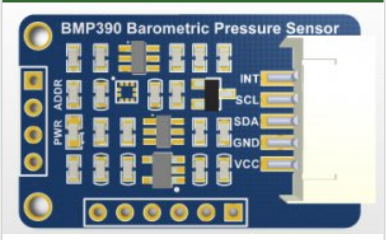
Overview

BMP390 Barometric Pressure Sensor



I2C, SPI

Introduction

The BMP390 is a 24-bit absolute barometric pressure sensor, with the features of ultra-small form factor, low power consumption, and low noise. This digital high-performance sensor is ideal for a variety of altitude-tracking applications. Supports both I2C / SPI interfaces, compatible with 3.3V/5V voltage levels. It can be easily integrated into projects such as GPS modules, wearables devices, audible devices, and drones, suitable for applications such as accurate altimeters, environment monitoring, and IoT projects.

Feature

- Support I2C/SPI interface communication, I2C interface by default.
- Onboard voltage translator and compatible with 3.3V/5V operating voltage.
- Provide online resources and manuals (Raspberry/Arduino/Pico deno example, user manual, etc.)

Specification

- Operating voltage: 5V/3.3V
- Communication interface: I2C/SPI
- Average operating current: 3.2μA (1Hz)
- Barometric detection range: 300~1250hPa
- Barometric pressure absolute accuracy: ±0.50hPa (P=300 ...1100 hPa T=0 ... 65 °C)
- Barometric pressure relative accuracy: ±0.03hPa (P=700...1100 hPa T=25...40°C)
- Temperature coefficient offset: ±0.6Pa/K (25°...40°C at 900 hPa)
- Temperature absolute accuracy: ±1.5℃ (0...65℃)
- Resolution support: 0.016Pa (In high precision mode)
- Possible sampling rates: 200Hz

- Operating temperature: -40~85℃
- Dimensions: 32mm × 20mm
- Via-hole diameter: 2.0mm

Interface Description

- Pin function:

Pin	Description
VCC	Power Input
GND	GND
SDA	I2C data
SCL	I2C clock
INT	Interrupt output, can be connected to I/O
SDO	SPI data, can be connected to the host MISO
SDI	SPI data, can be connected to the host MOSI
CS	Chip selection, can be connected to I/O

The I2C address can be configured through the ADDR pad, the default is not soldered 0Ω resistor, and the I2C address is 0x77, after soldering is 0x76. (Please do not solder this 0Ω resistor when using SPI mode).

Working with Raspberry Pi

Function Library Installation

- Install WiringPi:

```
sudo apt-get install wiringpi
#For the Raspberry Pi 4B it may be necessary to upgrade:
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
# Run gpio -v and version 2.52 will appear, if it does not appear, it means there is
an installation error.
```

Preparation

Execute the following commands to debug the Raspberry Pi:

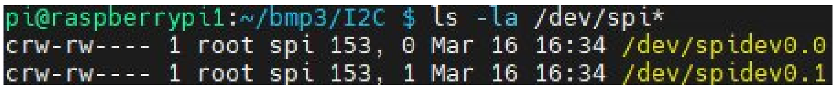
```
sudo raspi-config
```

Choose Interfacing Options -> I2C -> Yes to reboot the I2C kernel driver.
Choose Interfacing Options -> SPI -> Yes to reboot the SPI kernel driver.
Save, exit and then reboot the Raspberry Pi.

```
sudo reboot
```

Check whether SPI is normally booted after rebooting:

```
ls /dev/spi*
```



```
pi@raspberrypi1:~/bmp3/I2C $ ls -la /dev/spi*
crw-rw---- 1 root spi 153, 0 Mar 16 16:34 /dev/spidev0.0
crw-rw---- 1 root spi 153, 1 Mar 16 16:34 /dev/spidev0.1
```

WiringPi Demo

Download and unzip:

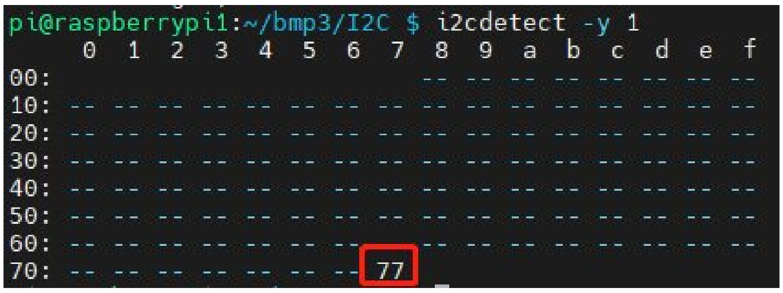
```
sudo apt-get update
sudo apt-get unzip
wget https://files.waveshare.com/upload/b/bf/BMP390_Barometric_Pressure_Sensor_code.
zip
unzip BMP390_Barometric_Pressure_Sensor_code.zip
```

I2C

Module	Raspberry Pi
VCC	3.3V/5V
GND	GND
SDA	SDA
SCL	SCL
INT	NC

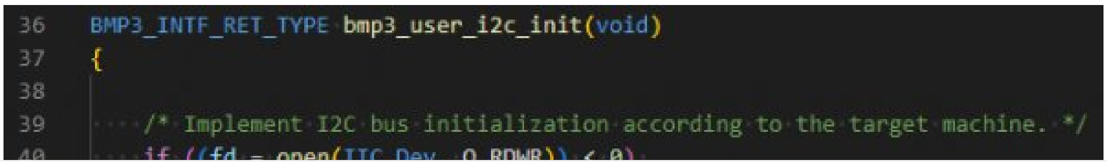
- Check the connection:

```
i2cdetect -y 1
```

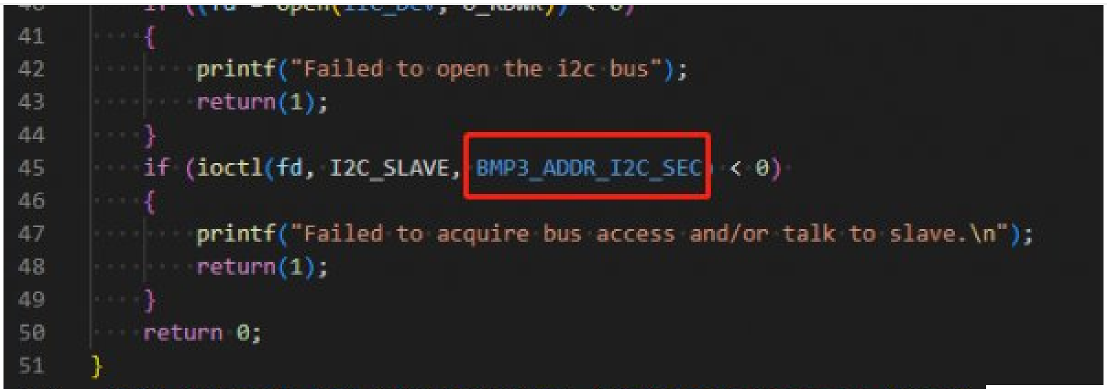


```
pi@raspberrypi1:~/bmp3/I2C $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
```

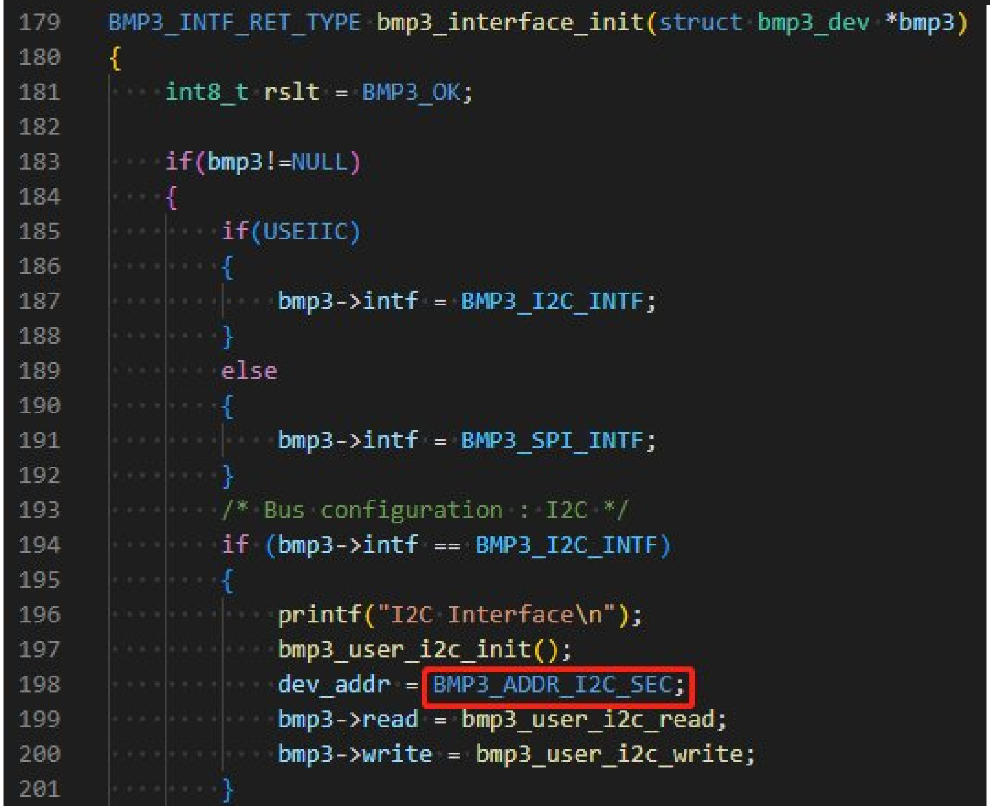
The I2C address of the module is 0X77. The address can be changed by modifying the pad, and the I2C address in the program should be modified after modifying the hardware address.



```
BMP3_INTF_RET_TYPE bmp3_user_i2c_init(void)
{

    /* Implement I2C bus initialization according to the target machine. */
    if ((fd = open(IIC_Dev, O_RDWR)) < 0)
```



```c
    {
        printf("Failed to open the i2c bus");
        return(1);
    }
    if (ioctl(fd, I2C_SLAVE, BMP3_ADDR_I2C_SEC) < 0)
    {
        printf("Failed to acquire bus access and/or talk to slave.\n");
        return(1);
    }
    return 0;
}
```



```c
BMP3_INTF_RET_TYPE bmp3_interface_init(struct bmp3_dev *bmp3)
{
    int8_t rslt = BMP3_OK;

    if(bmp3!=NULL)
    {
        if(USEIIC)
        {
            bmp3->intf = BMP3_I2C_INTF;
        }
        else
        {
            bmp3->intf = BMP3_SPI_INTF;
        }
        /* Bus configuration : I2C */
        if (bmp3->intf == BMP3_I2C_INTF)
        {
            printf("I2C Interface\n");
            bmp3_user_i2c_init();
            dev_addr = BMP3_ADDR_I2C_SEC;
            bmp3->read = bmp3_user_i2c_read;
            bmp3->write = bmp3_user_i2c_write;
        }
```

```
BMP3_ADDR_I2C_PRIM=0x76
BMP3_ADDR_I2C_SEC=0x77
```

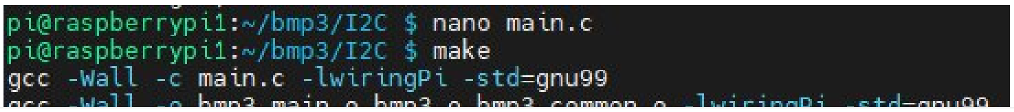
- Enter the I2C directory to compile the program:

```
cd
cd BMP390_Barometric_Pressure_Sensor_code/RaspberryPi/I2C
make clean
make
```

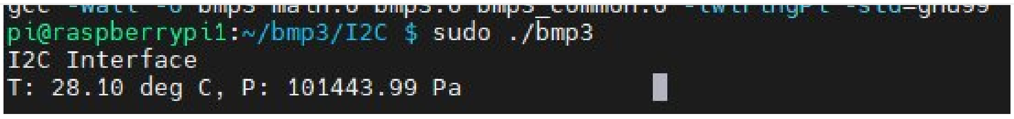
- Run the demo:

```
sudo ./bmp3
```

- Data is displayed:



```
pi@raspberrypi1:~/bmp3/I2C $ nano main.c
pi@raspberrypi1:~/bmp3/I2C $ make
gcc -Wall -c main.c -lwiringPi -std=gnu99
gcc -Wall -o bmp3 main.o bmp3.o bmp3_common.o -lwiringPi -std=gnu99
```



```
gcc -Wall -o bmp3 main.o bmp3.o bmp3_common.o -lwiringPi -std=gnu99
pi@raspberrypi1:~/bmp3/I2C $ sudo ./bmp3
I2C Interface
T: 28.10 deg C, P: 101443.99 Pa
```

Wait a while to get stable data. From left to right, the temperature (in Celsius), and atmospheric pressure (in Pascals) measured by the BMP390 are shown.

SPI

Module	Raspberry Pi
VCC	3.3V
GND	GND
SDO	MISO
SDI	MOSI
SCK	SCLK
CS	27 (wiringPi code)

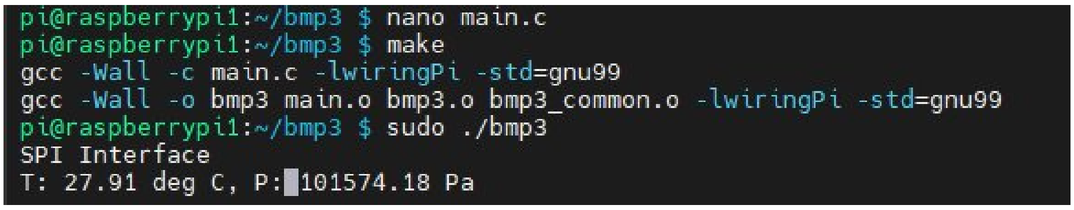
- Enter the SPI directory and compile the demo:

```
cd
cd BMP390_Barometric_Pressure_Sensor_code/RaspberryPi/SPI
make clean
make
```

- Run the demo:

```
sudo ./bmp3
```

- Data is displayed:



```
pi@raspberrypi1:~/bmp3 $ nano main.c
pi@raspberrypi1:~/bmp3 $ make
gcc -Wall -c main.c -lwiringPi -std=gnu99
gcc -Wall -o bmp3 main.o bmp3.o bmp3_common.o -lwiringPi -std=gnu99
pi@raspberrypi1:~/bmp3 $ sudo ./bmp3
SPI Interface
T: 27.91 deg C, P: 101574.18 Pa
```

Wait a while to get stable data. From left to right, the temperature (in Celsius) and atmospheric pressure (in Pascals) measured by the BMP390 are shown.

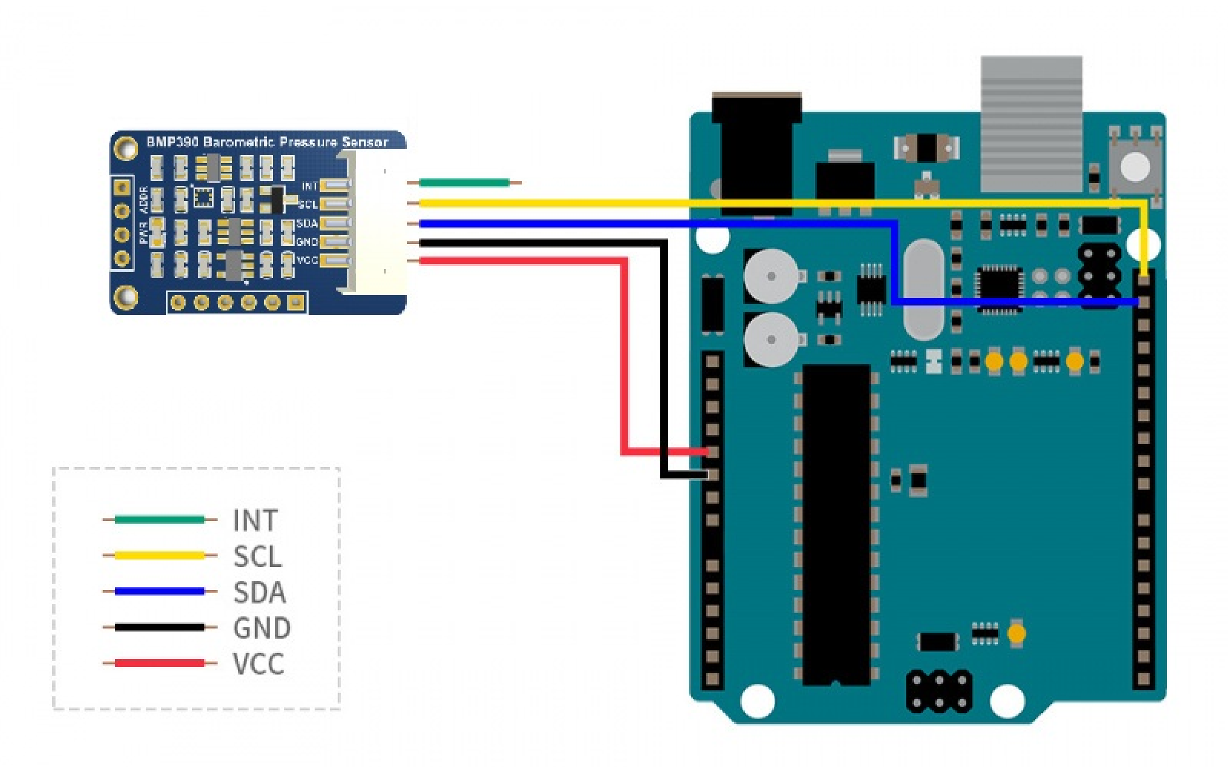
Working with Arduino

I2C

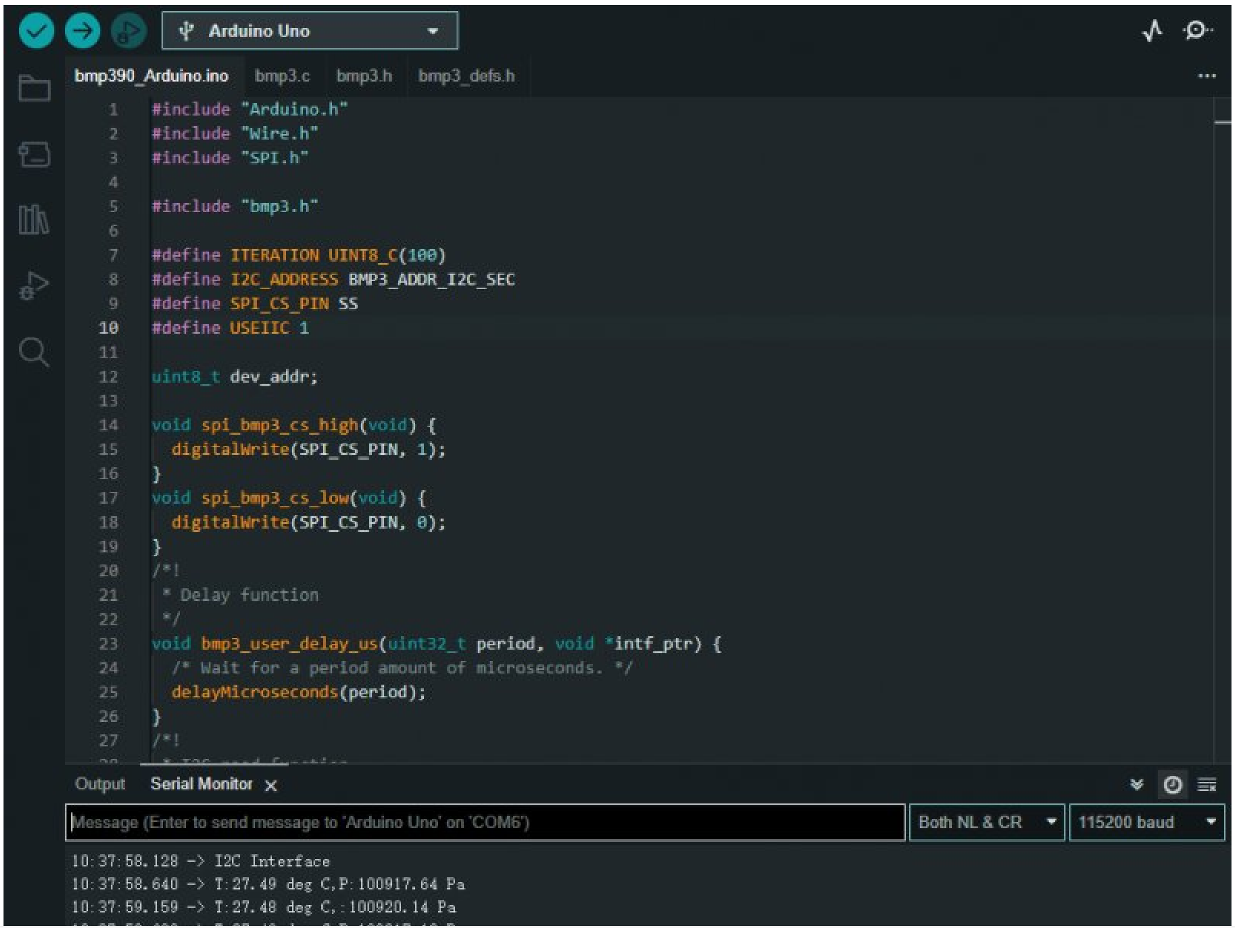
- Connect the module to the Arduino development board.

Module	Arduino
VCC	5V
GND	GND
SDA	SDA

SCL	SCL
INT	NC



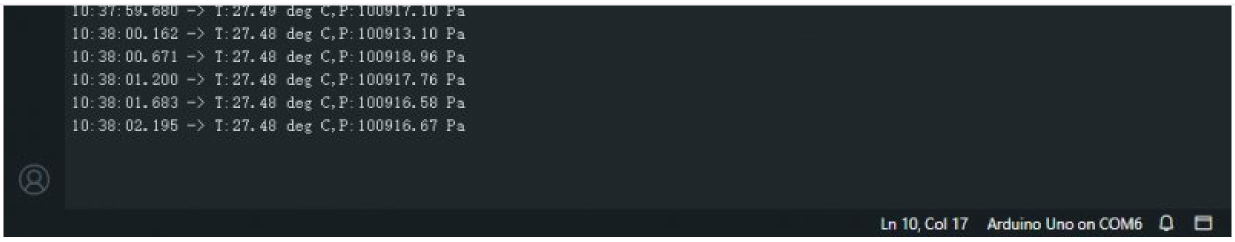
After wiring the device correctly and determining the communication method and device address, compile and download it to Arduino.
Open: Tools -> Serial Monitor, and select baud rate 115200, you can get the following information.



```
#include "Arduino.h"
#include "Wire.h"
#include "SPI.h"

#include "bmp3.h"

#define ITERATION UINT8_C(100)
#define I2C_ADDRESS BMP3_ADDR_I2C_SEC
#define SPI_CS_PIN SS
#define USEIIC 1

uint8_t dev_addr;

void spi_bmp3_cs_high(void) {
  digitalWrite(SPI_CS_PIN, 1);
}
void spi_bmp3_cs_low(void) {
  digitalWrite(SPI_CS_PIN, 0);
}
/*!
 * Delay function
 */
void bmp3_user_delay_us(uint32_t period, void *intf_ptr) {
  /* Wait for a period amount of microseconds. */
  delayMicroseconds(period);
}
/*!
```

Message (Enter to send message to 'Arduino Uno' on 'COM6')
Both NL & CR
115200 baud

```
10:37:58.128 -> I2C Interface
10:37:58.640 -> T:27.49 deg C,P:100917.64 Pa
10:37:59.159 -> T:27.48 deg C,:100920.14 Pa
```



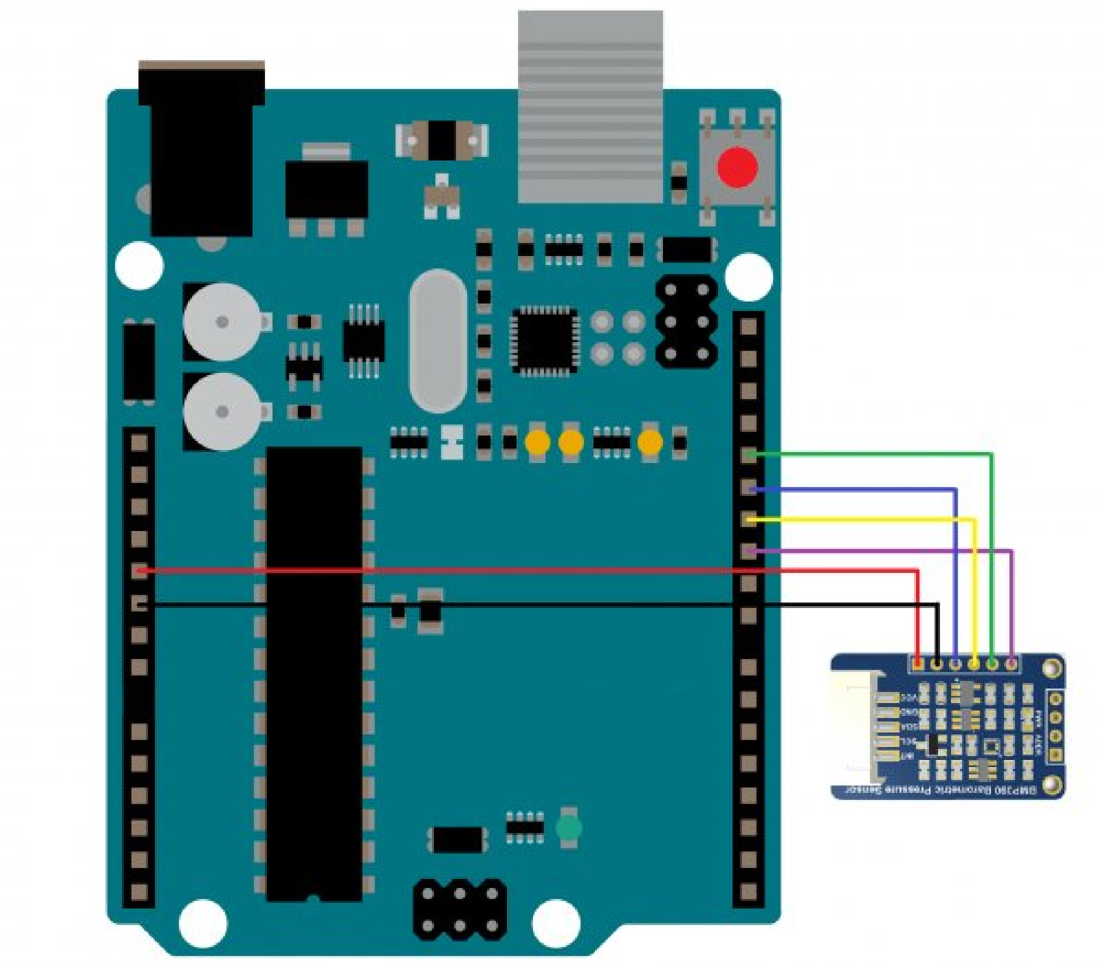
```
10:37:59.680 -> T:27.49 deg C,P:100917.10 Pa
10:38:00.162 -> T:27.48 deg C,P:100913.10 Pa
10:38:00.671 -> T:27.48 deg C,P:100918.96 Pa
10:38:01.200 -> T:27.48 deg C,P:100917.76 Pa
10:38:01.683 -> T:27.48 deg C,P:100916.58 Pa
10:38:02.195 -> T:27.48 deg C,P:100916.67 Pa
```

Wait a moment to get stable data. From left to right, the BMP390 sensor is showing the measured atmospheric pressure (in Pascals) and temperature (in Celsius). 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.

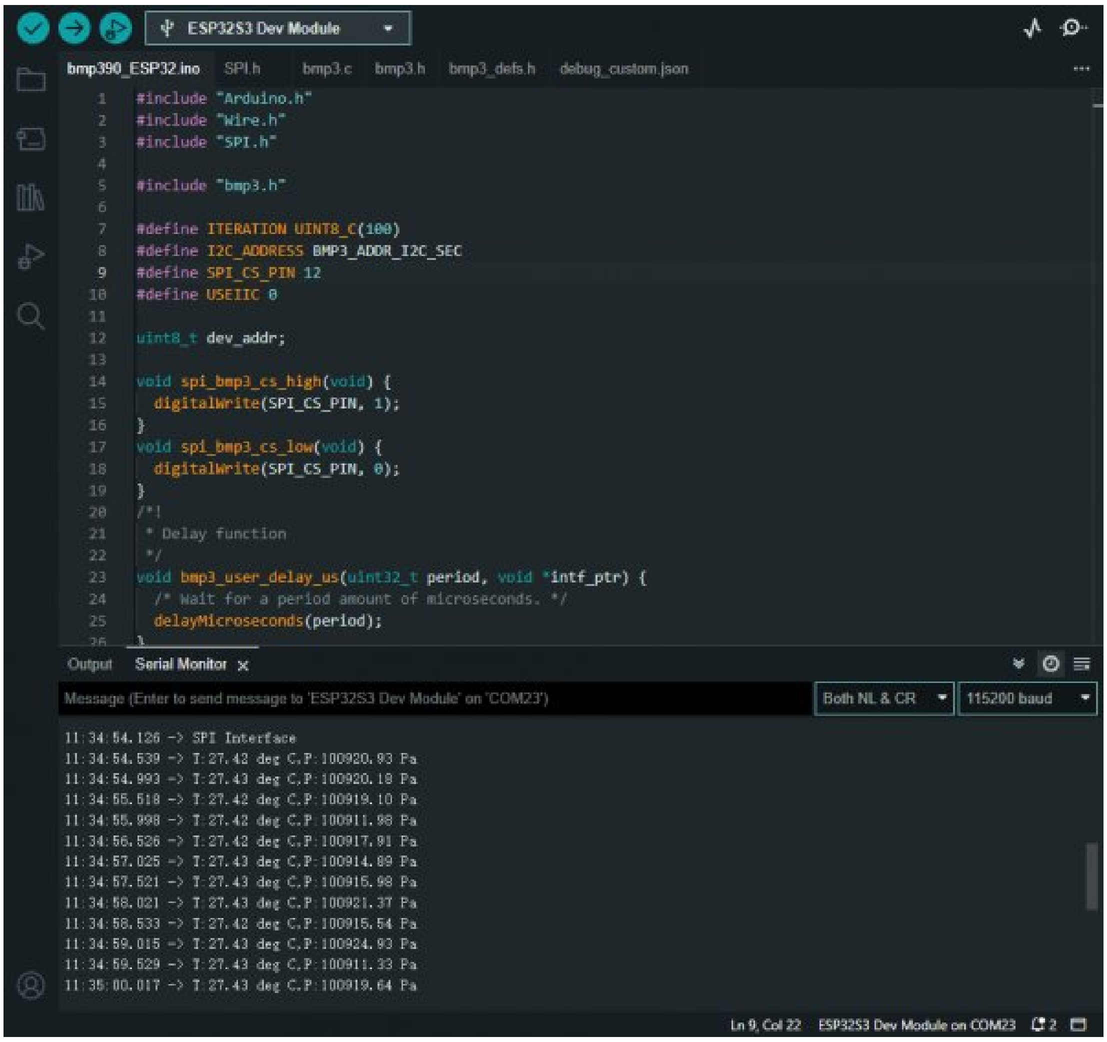
SPI

- Connect the module to the Arduino development board according to the following ways:

Module	Arduino
VCC	5V
GND	GND
SDO	12
SDI	11
SCK	13
CS	10 (SS)



- After correctly connecting, check the communication method (The following figure confirms that SPI mode is used when the macro definition USE_IIC is 0) and the device address, compile and download to Arduino.
- Open Tool -> SSCOM, choose the baud rate as 115200, and get the following information:



```
#include "Arduino.h"
#include "Wire.h"
#include "SPI.h"

#include "bmp3.h"

#define ITERATION UINT8_C(100)
#define I2C_ADDRESS BMP3_ADDR_I2C_SEC
#define SPI_CS_PIN 12
#define USEIIC 0

uint8_t dev_addr;

void spi_bmp3_cs_high(void) {
  digitalWrite(SPI_CS_PIN, 1);
}
void spi_bmp3_cs_low(void) {
  digitalWrite(SPI_CS_PIN, 0);
}
/*!
 * Delay function
 */
void bmp3_user_delay_us(uint32_t period, void *intf_ptr) {
  /* Wait for a period amount of microseconds. */
  delayMicroseconds(period);
}
```

Message (Enter to send message to 'ESP32S3 Dev Module' on 'COM23')

```
11:34:54.126 -> SPI Interface
11:34:54.539 -> T:27.42 deg C,P:100920.93 Pa
11:34:54.993 -> T:27.43 deg C,P:100920.18 Pa
11:34:55.518 -> T:27.42 deg C,P:100919.10 Pa
11:34:55.998 -> T:27.42 deg C,P:100911.98 Pa
11:34:56.526 -> T:27.42 deg C,P:100917.91 Pa
11:34:57.025 -> T:27.43 deg C,P:100914.89 Pa
11:34:57.521 -> T:27.43 deg C,P:100915.98 Pa
11:34:58.021 -> T:27.43 deg C,P:100921.37 Pa
11:34:58.533 -> T:27.42 deg C,P:100915.54 Pa
11:34:59.015 -> T:27.43 deg C,P:100924.93 Pa
11:34:59.529 -> T:27.43 deg C,P:100911.33 Pa
11:35:00.017 -> T:27.43 deg C,P:100919.64 Pa
```

Wait a moment to get stable data. From left to right, the BMP390 sensor is showing the measured atmospheric pressure (in Pascals) and temperature (in Celsius). 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.

Working with ESP32

Take ESP32-S3-DEV-KIT-N8R8 as an example.

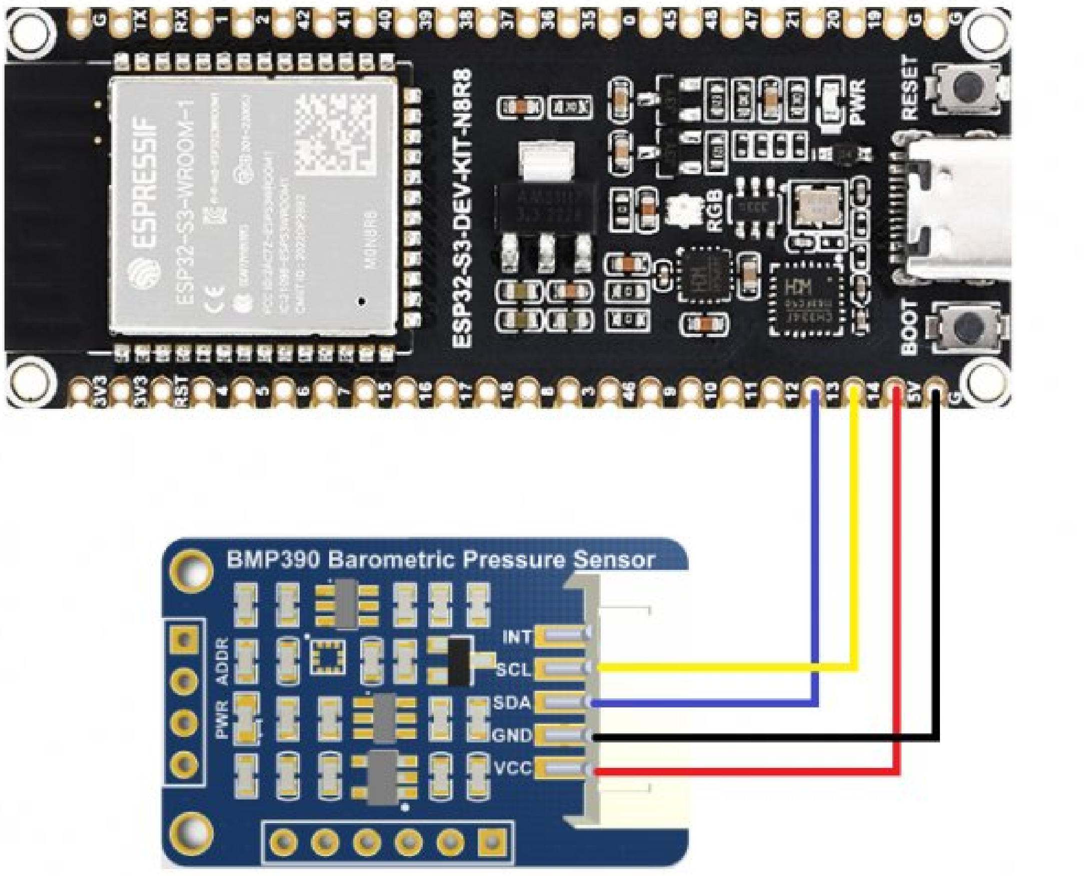
I2C

Connect the module to the ESP32 development board as shown below:

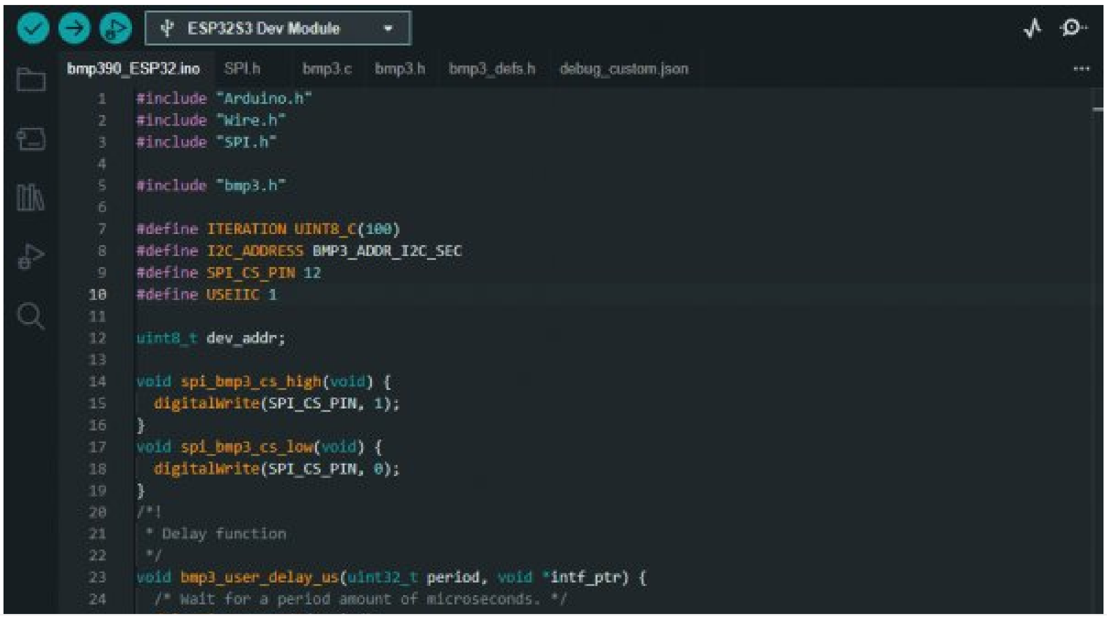
Module	ESP32
VCC	5V
GND	GND
SDA	13

SCL	14
INT	NC

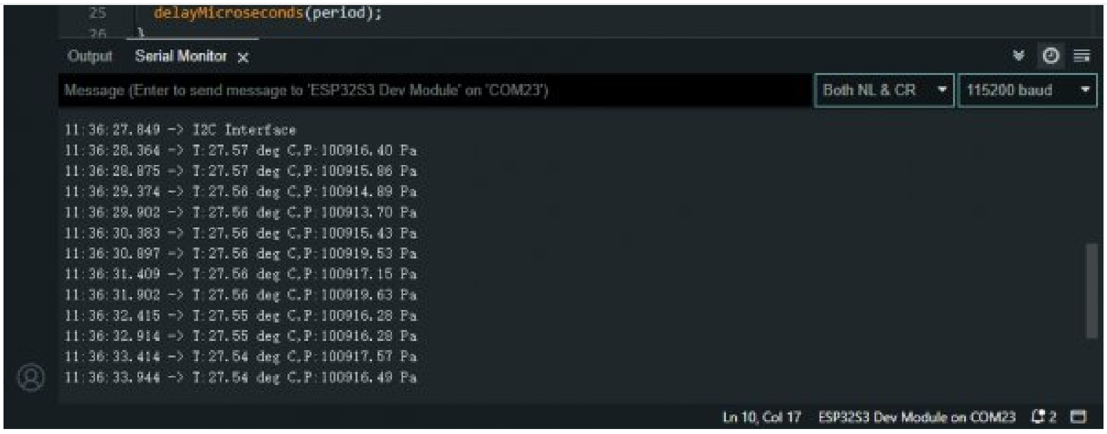
The ESP32's I2C bus can be mapped to most external pins and can be configured to suit your needs.



- After wiring the device correctly and determining the communication method and device address, compile and download it to Arduino.
- Open: Tools -> Serial Monitor, select baud rate 115200, and you can get the following information.



```
#include "Arduino.h"
#include "Wire.h"
#include "SPI.h"

#include "bmp3.h"

#define ITERATION UINT8_C(100)
#define I2C_ADDRESS BMP3_ADDR_I2C_SEC
#define SPI_CS_PIN 12
#define USEIIC 1

uint8_t dev_addr;

void spi_bmp3_cs_high(void) {
  digitalWrite(SPI_CS_PIN, 1);
}
void spi_bmp3_cs_low(void) {
  digitalWrite(SPI_CS_PIN, 0);
}
/*!
 * Delay function
 */
void bmp3_user_delay_us(uint32_t period, void *intf_ptr) {
  /* Wait for a period amount of microseconds. */
```



```
    delayMicroseconds(period);
}
```

Output   Serial Monitor

Message (Enter to send message to 'ESP32S3 Dev Module' on 'COM23')   Both NL & CR   115200 baud

```
11:36:27.849 -> I2C Interface
11:36:28.364 -> T:27.57 deg C,P:100916.40 Pa
11:36:28.875 -> T:27.57 deg C,P:100915.86 Pa
11:36:29.374 -> T:27.56 deg C,P:100914.89 Pa
11:36:29.902 -> T:27.56 deg C,P:100913.70 Pa
11:36:30.383 -> T:27.56 deg C,P:100915.43 Pa
11:36:30.897 -> T:27.56 deg C,P:100919.53 Pa
11:36:31.409 -> T:27.56 deg C,P:100917.15 Pa
11:36:31.902 -> T:27.56 deg C,P:100919.63 Pa
11:36:32.415 -> T:27.55 deg C,P:100916.28 Pa
11:36:32.914 -> T:27.55 deg C,P:100916.28 Pa
11:36:33.414 -> T:27.54 deg C,P:100917.57 Pa
11:36:33.944 -> T:27.54 deg C,P:100916.49 Pa
```

Ln 10, Col 17   ESP32S3 Dev Module on COM23   2

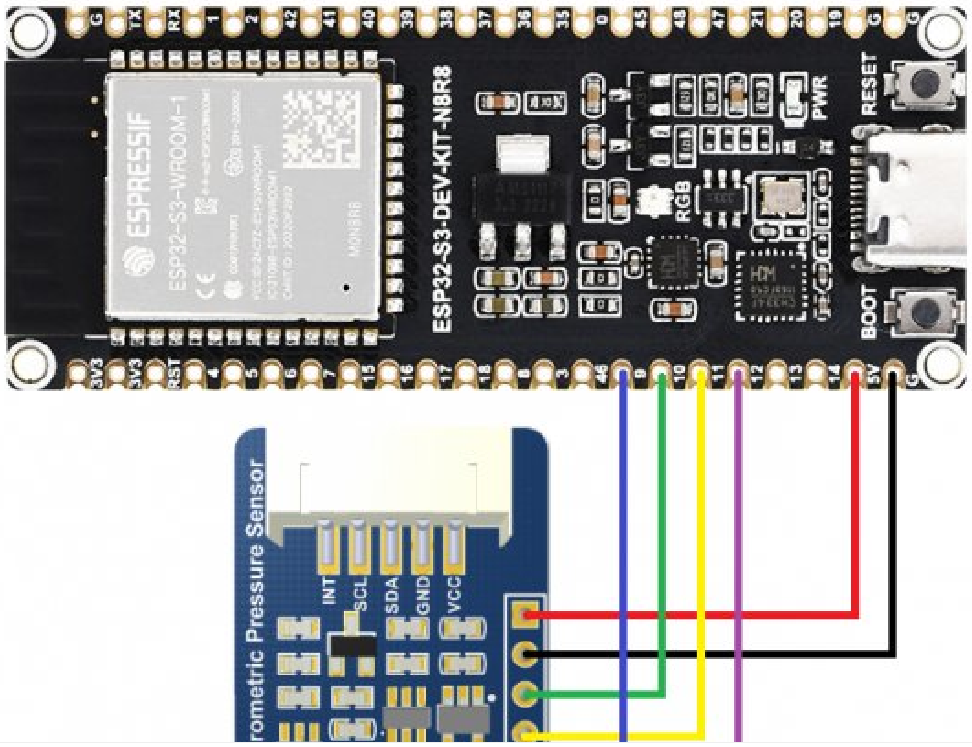
Wait a moment to get stable data. From left to right, the BMP390 sensor is showing the measured atmospheric pressure (in Pascals) and temperature (in Celsius). If the data is not displayed successfully, or if the data is not displayed properly, please check the connection, communication method, and device address for errors.

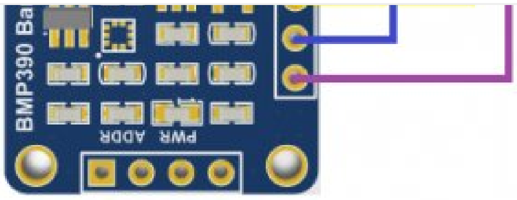
SPI

- Connect the module to the ESP32 development board.

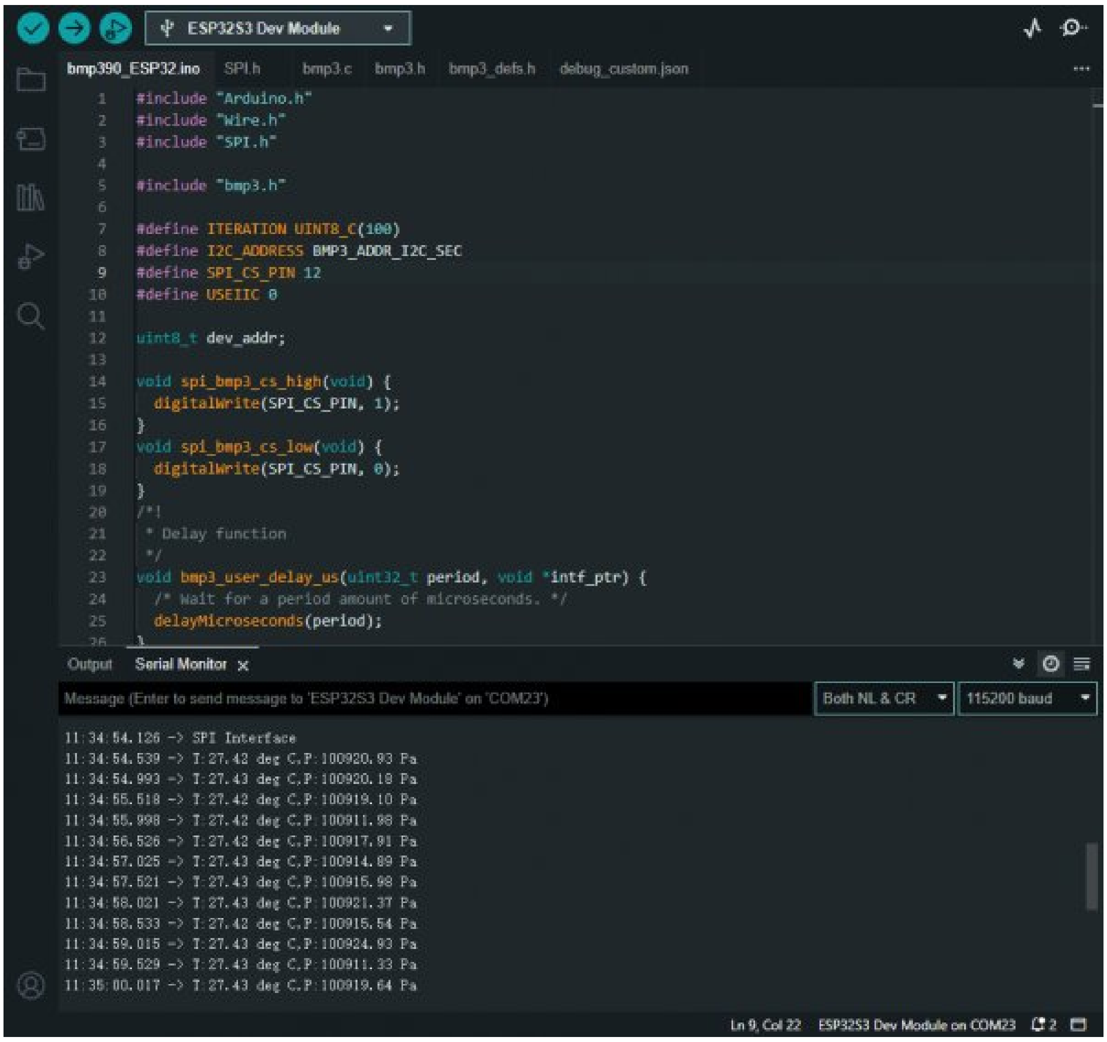
Module	ESP32
VCC	5V
GND	GND
SDO	10
SDI	11
SCK	9
CS	12(SS)

(The ESP32's SPI bus can be mapped to most external pins and can be configured to suit your needs):





- After correctly connecting, check the communication method (The following figure confirms that SPI mode is used when the macro definition USE_IIC is 0) and the device address, compile and download to Arduino.
- Open Tool -> SSCOM, choose the baud rate as 115200, and get the following information:



```
#include "Arduino.h"
#include "Wire.h"
#include "SPI.h"

#include "bmp3.h"

#define ITERATION UINT8_C(100)
#define I2C_ADDRESS BMP3_ADDR_I2C_SEC
#define SPI_CS_PIN 12
#define USEIIC 0

uint8_t dev_addr;

void spi_bmp3_cs_high(void) {
  digitalWrite(SPI_CS_PIN, 1);
}
void spi_bmp3_cs_low(void) {
  digitalWrite(SPI_CS_PIN, 0);
}
/*!
 * Delay function
 */
void bmp3_user_delay_us(uint32_t period, void *intf_ptr) {
  /* Wait for a period amount of microseconds. */
  delayMicroseconds(period);
}
```

```
11:34:54.126 -> SPI Interface
11:34:54.539 -> T:27.42 deg C,P:100920.93 Pa
11:34:54.993 -> T:27.43 deg C,P:100920.18 Pa
11:34:55.518 -> T:27.42 deg C,P:100919.10 Pa
11:34:55.998 -> T:27.42 deg C,P:100911.98 Pa
11:34:56.526 -> T:27.42 deg C,P:100917.91 Pa
11:34:57.025 -> T:27.43 deg C,P:100914.89 Pa
11:34:57.521 -> T:27.43 deg C,P:100915.98 Pa
11:34:58.021 -> T:27.43 deg C,P:100921.37 Pa
11:34:58.533 -> T:27.42 deg C,P:100915.54 Pa
11:34:59.015 -> T:27.43 deg C,P:100924.93 Pa
11:34:59.529 -> T:27.43 deg C,P:100911.33 Pa
11:35:00.017 -> T:27.43 deg C,P:100919.64 Pa
```

You can get stable data after waiting for a moment. From left to right, you can see the barometric pressure (in Pascals) and temperature (in Celsius) tested by the BMP390 sensor. If it fails to display the data, or the data is abnormal, you should check the cable connection, communication way, and the device address.

Working With Raspberry Pi Pico

Software Preparation

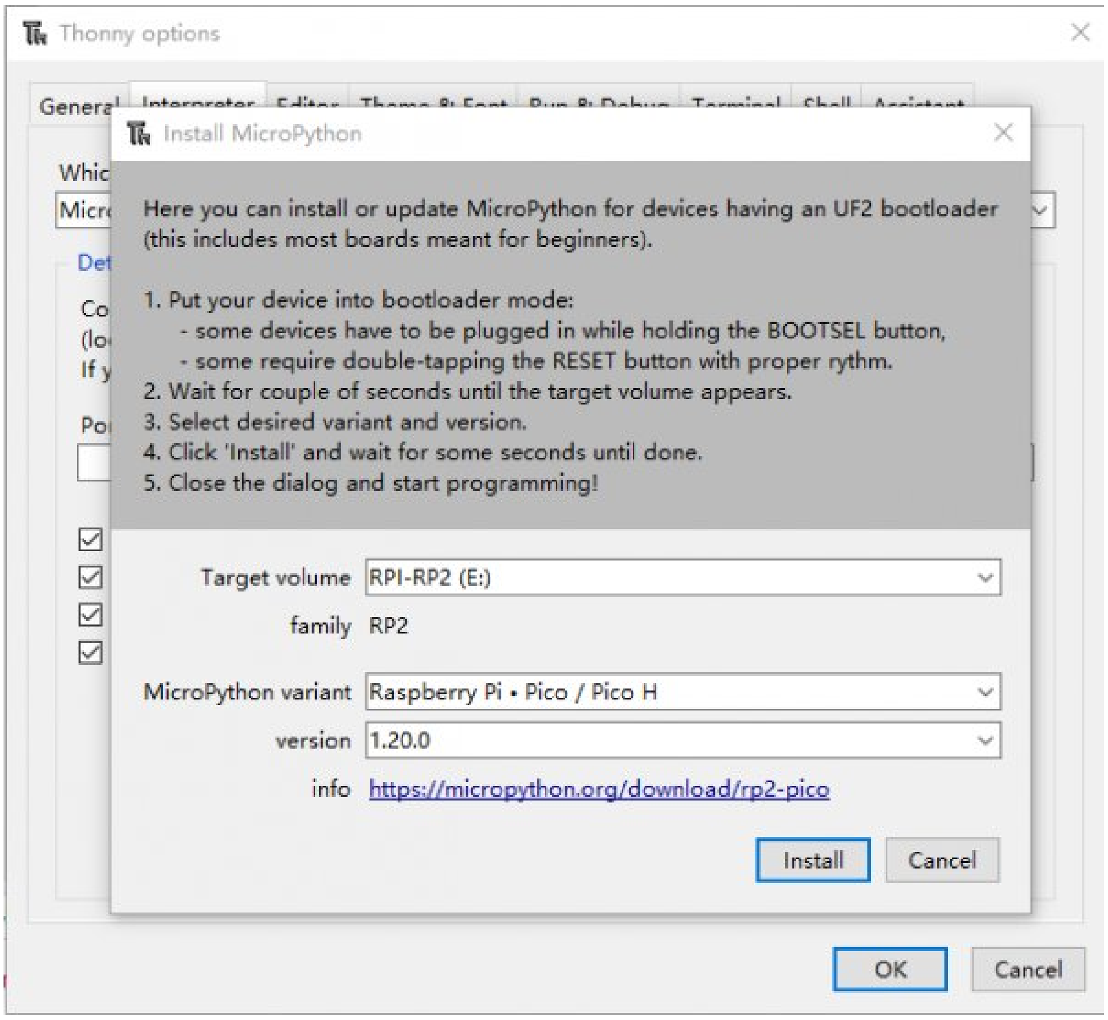
Install Thonny

- Directly enter Thonny to download.
- Also, you can click here to download.

Program MicroPython Firmware

- Online installation: the newest Thonny version can automatically download the firmware.

1. First, you need to press and hold the boot button of the Pico while connecting the Pico to the PC using the USB cable.
2. After that, open Thonny and click Tool -> Options... -> Interpretere -> INstall or update MicroPython.
3. In the pop-up window, select the corresponding development board and MicorPython version, and click Install, then wait for the installation to finish.



Here you can install or update MicroPython for devices having an UF2 bootloader (this includes most boards meant for beginners).

1. Put your device into bootloader mode:
   - some devices have to be plugged in while holding the BOOTSEL button,
   - some require double-tapping the RESET button with proper rythm.
2. Wait for couple of seconds until the target volume appears.
3. Select desired variant and version.
4. Click 'Install' and wait for some seconds until done.
5. Close the dialog and start programming!

Target volume: RPI-RP2 (E:)
family: RP2
MicroPython variant: Raspberry Pi • Pico / Pico H
version: 1.20.0
info: https://micropython.org/download/rp2-pico

If the software installation MicroPython window keeps prompting downloading variants info... and there is no change for a long time, you can choose to use a proxy or use the following offline installation method instead.

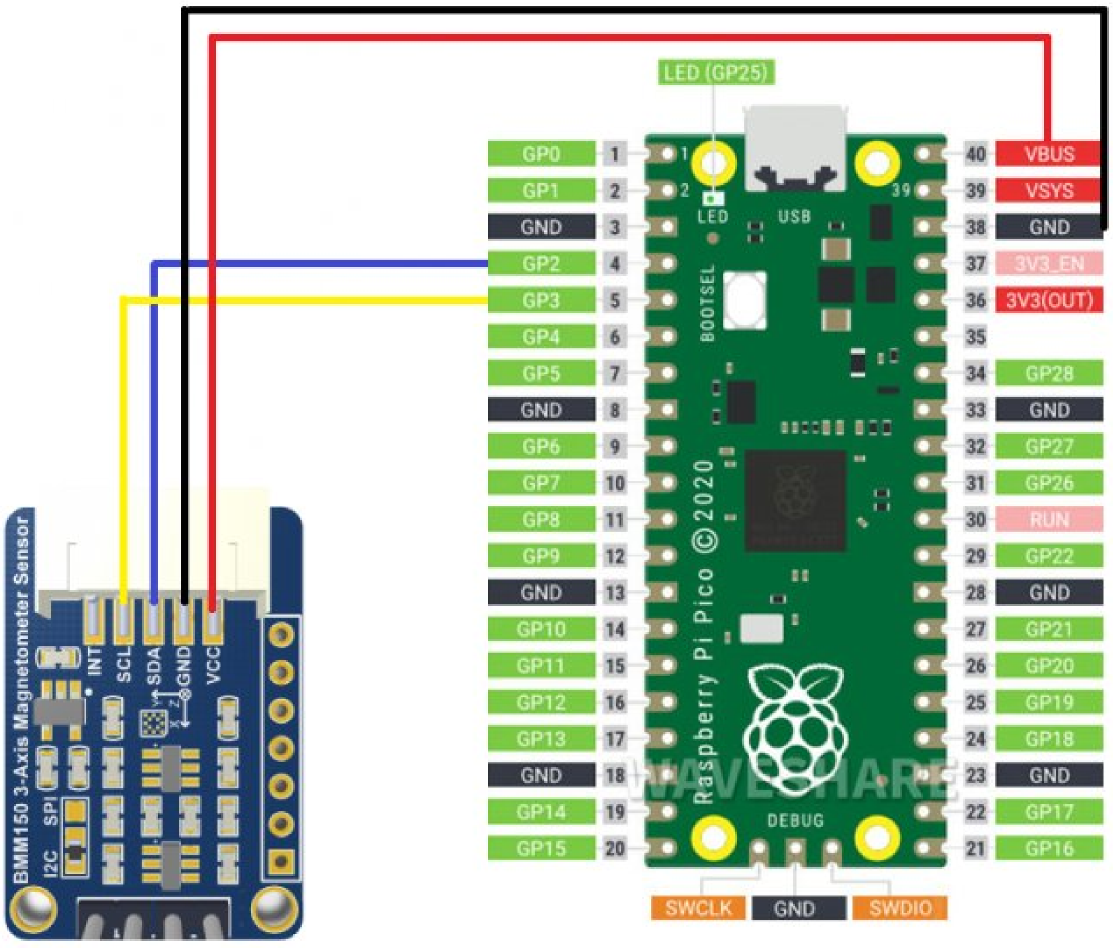
- Off-line installation method:

1. Download the firmware and get a .uf2 file.
2. Press the boot button of the Pico, connect the Pico to the PC with a USB cable, and then a USB storage device appears on the computer.

3. Open the device folder, and drag the ".uf2" file directly into the folder, after that, the storage device automatically popped up, and the development board reboot, at this time represents the installation is complete, you can open Thonny to see whether the successful recognition of Pico's micro python environment. For more details, please refer to Pico's website.

Hardware Connection

Module	Pico
VCC	5V
GND	GND
SDA	GP2
SCL	GP3
INT	NC

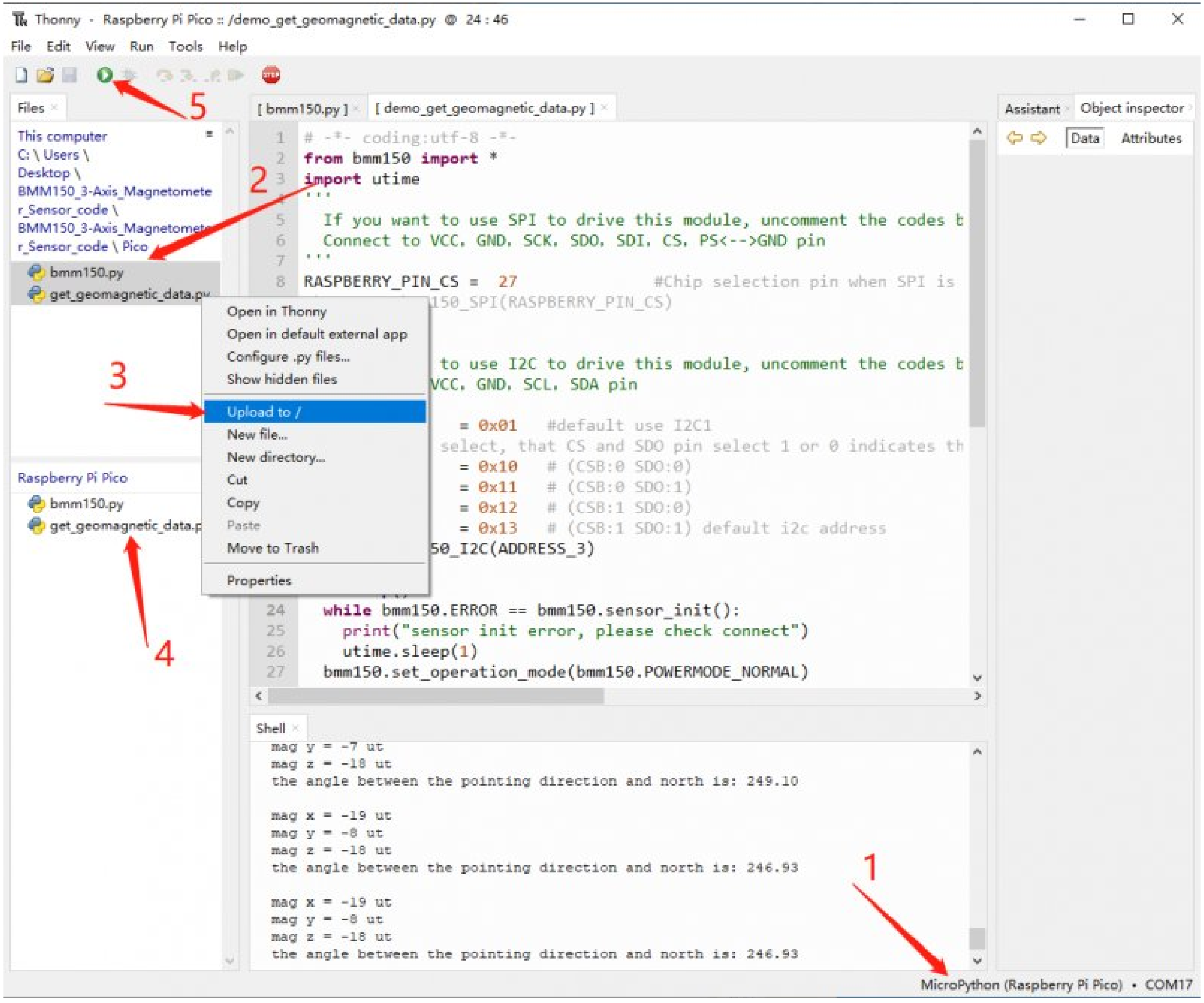


Demo Use

As shown in the picture:

1. Verify that a Raspberry Pi Pico development board with Micropython firmware written to it is connected.
2. Open the unzipped sample program folder in the upper left corner of the file screen and open the Pico folder.

3. Select the two .py files and right-click on them and select Upload.
4. Open the example program get_geomagnetic_data.py by double-clicking on the in-chip file screen at the bottom left after uploading.
5. Click the button above to run.



```
# -*- coding:utf-8 -*-
from bmm150 import *
import utime
'''
  If you want to use SPI to drive this module, uncomment the codes b
  Connect to VCC, GND, SCK, SDO, SDI, CS, PS<-->GND pin
'''
RASPBERRY_PIN_CS =  27          #Chip selection pin when SPI is
                 150_SPI(RASPBERRY_PIN_CS)

              to use I2C to drive this module, uncomment the codes b
            VCC, GND, SCL, SDA pin

              = 0x01   #default use I2C1
            select, that CS and SDO pin select 1 or 0 indicates th
              = 0x10   # (CSB:0 SDO:0)
              = 0x11   # (CSB:0 SDO:1)
              = 0x12   # (CSB:1 SDO:0)
              = 0x13   # (CSB:1 SDO:1) default i2c address
            50_I2C(ADDRESS_3)

  while bmm150.ERROR == bmm150.sensor_init():
    print("sensor init error, please check connect")
    utime.sleep(1)
  bmm150.set_operation_mode(bmm150.POWERMODE_NORMAL)
```

```
mag y = -7 ut
mag z = -18 ut
the angle between the pointing direction and north is: 249.10

mag x = -19 ut
mag y = -8 ut
mag z = -18 ut
the angle between the pointing direction and north is: 246.93

mag x = -19 ut
mag y = -8 ut
mag z = -18 ut
the angle between the pointing direction and north is: 246.93
```

Resource

Document

- Schematic

Demo

- Demo

Software

- Arduino IDE
- SSCOM Serial Assistant

Related Resource

- BMP390 Datasheet

FAQ

Question: There is a problem with the altitude value detected by the barometer, for example, if I am at an altitude of 100 meters above sea level, the altitude detected by the barometer is indeed a negative value, what could be the reason?

Answer:

The height calculated by the air pressure sensor is generally used as a relative value in a short period. For example, if you take a vertical elevator, record a height before the elevator starts, and record a height when the elevator rises to the 3rd floor, the height difference between the two heights is accurate.

If you need to use it on absolute occasions, you can enter the height of the current position as the initial value of the calculation, and then the movement to observe the height change is accurate. However, if the time is long, it will also be easy to have the problem of height drift.

If the project requires a long time to obtain accurate altitude values and high-frequency requirements, it is necessary to be operated with other sensors for processing, such as GPS.

Support