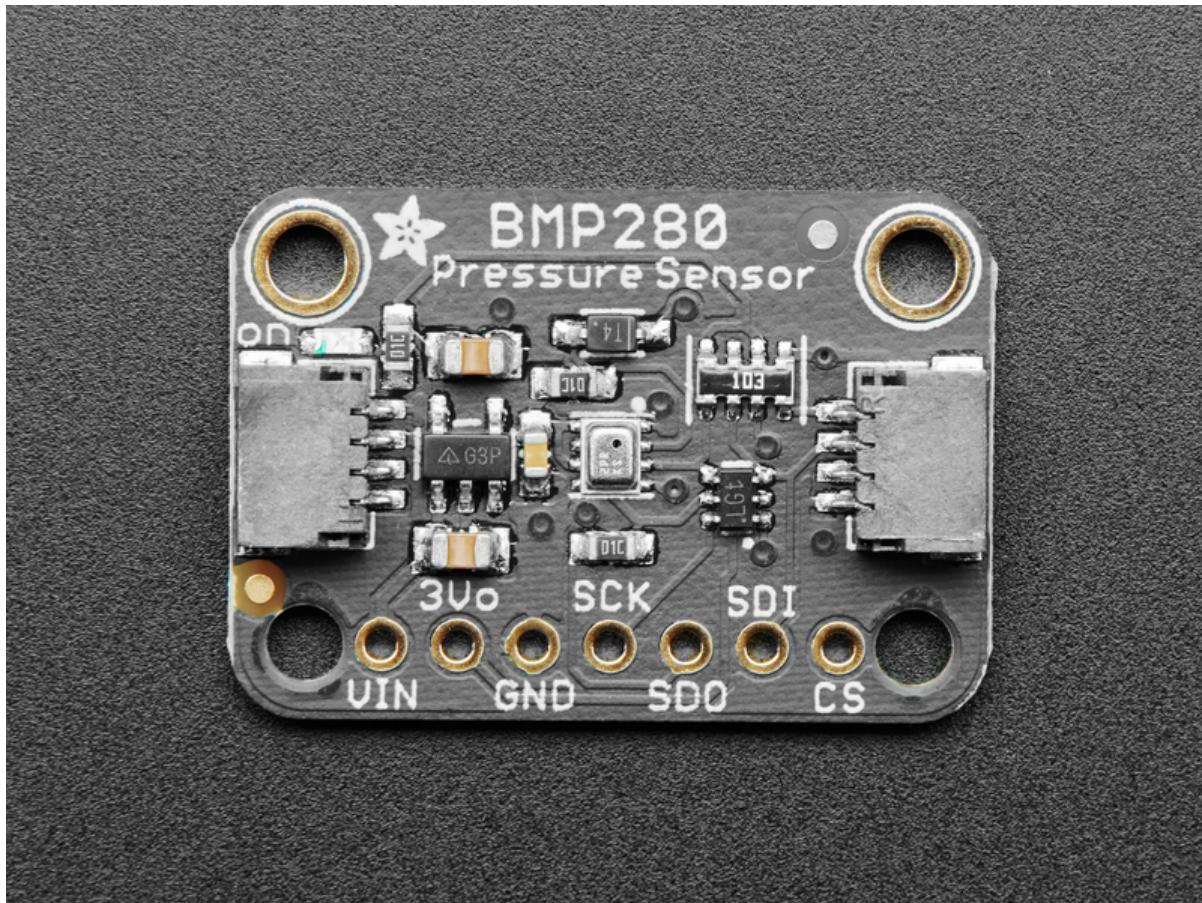




Adafruit BMP280 Barometric Pressure + Temperature Sensor Breakout

Created by lady ada



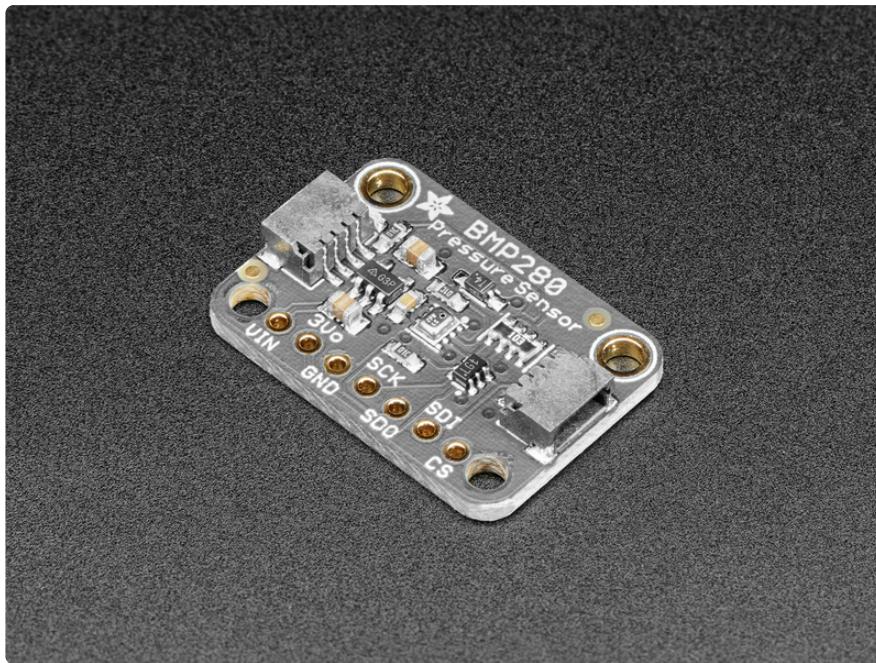
<https://learn.adafruit.com/adafruit-bmp280-barometric-pressure-plus-temperature-sensor-breakout>

Last updated on 2024-06-03 01:46:24 PM EDT

Table of Contents

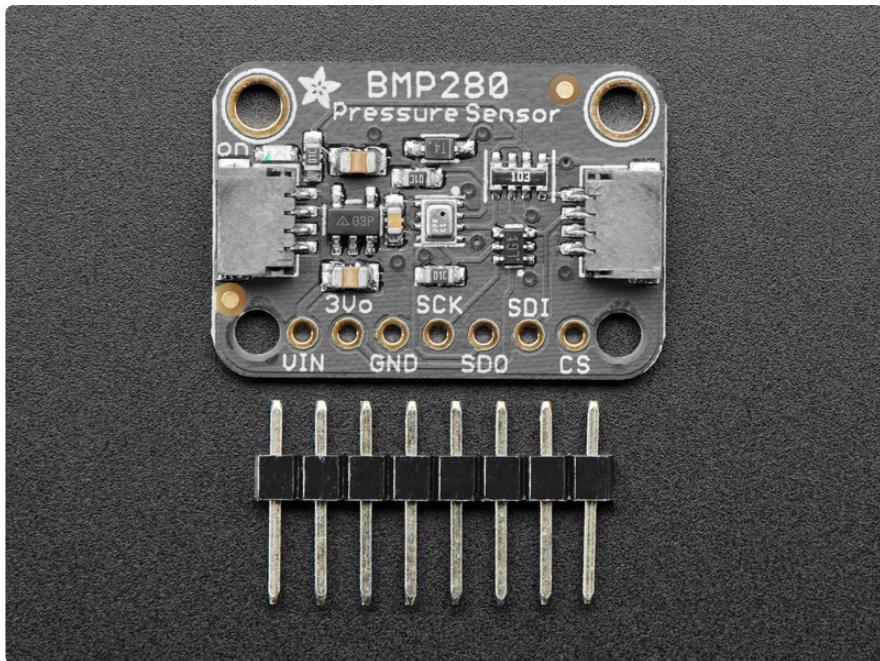
Overview	3
Pinouts	6
• SPI Logic pins: • Power Pins: • I2C Logic pins:	
Assembly	8
• Prepare the header strip: • Add the breakout board: • And Solder!	
Arduino Test	10
• I2C Wiring • SPI Wiring • Download Adafruit_BMP280 library • Load Demo • Library Reference	
Python & CircuitPython Test	16
• CircuitPython Microcontroller Wiring • Python Computer Wiring • CircuitPython Installation of BMP280 Library • Python Installation of BMP280 Library • CircuitPython & Python Usage	
Python Docs	23
WipperSnapper	24
• What is WipperSnapper • Wiring • Usage	
F.A.Q.	31
Downloads	32
• Documents • Schematic - STEMMA QT version • Fab Print - STEMMA QT version • Schematic - Original version • Fab Print - Original version	

Overview

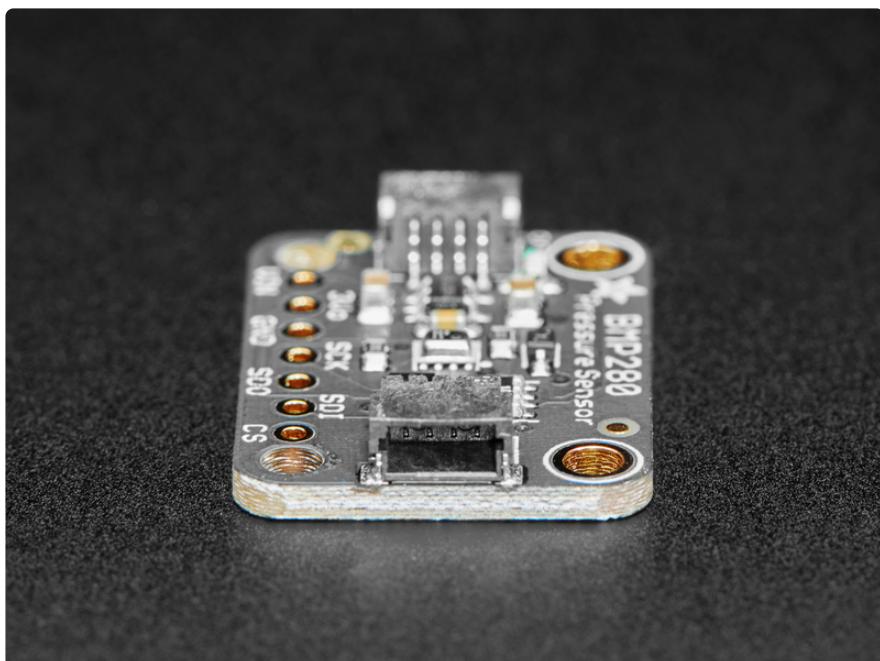


Bosch has stepped up their game with their new BMP280 sensor, an environmental sensor with temperature, barometric pressure that is the next generation upgrade to the BMP085/BMP180/BMP183. This sensor is great for all sorts of weather sensing and can even be used in both I2C and SPI!

This precision sensor from Bosch is the best low-cost, precision sensing solution for measuring barometric pressure with ± 1 hPa absolute accuracy, and temperature with $\pm 1.0^\circ\text{C}$ accuracy. Because pressure changes with altitude, and the pressure measurements are so good, you can also use it as an altimeter with ± 1 meter accuracy.



The BMP280 is the next-generation of sensors from Bosch, and is the upgrade to the BMP085/BMP180/BMP183 - with a low altitude noise of 0.25m and the same fast conversion time. It has the same specifications, but can use either I2C or SPI. For simple easy wiring, go with I2C. If you want to connect a bunch of sensors without worrying about I2C address collisions, go with SPI.



Nice sensor right? So we made it easy for you to get right into your next project. The surface-mount sensor is soldered onto a custom made PCB in the [STEMMA QT form factor](https://adafru.it/LBQ) (<https://adafru.it/LBQ>), making them easy to interface with. The [STEMMA QT connectors](https://adafru.it/JqB) (<https://adafru.it/JqB>) on either side are compatible with the [SparkFun Qwiic](https://adafru.it/Fpw) (<https://adafru.it/Fpw>) I2C connectors. This allows you to make solderless connections between your development board and the BMP280 or to chain it with a

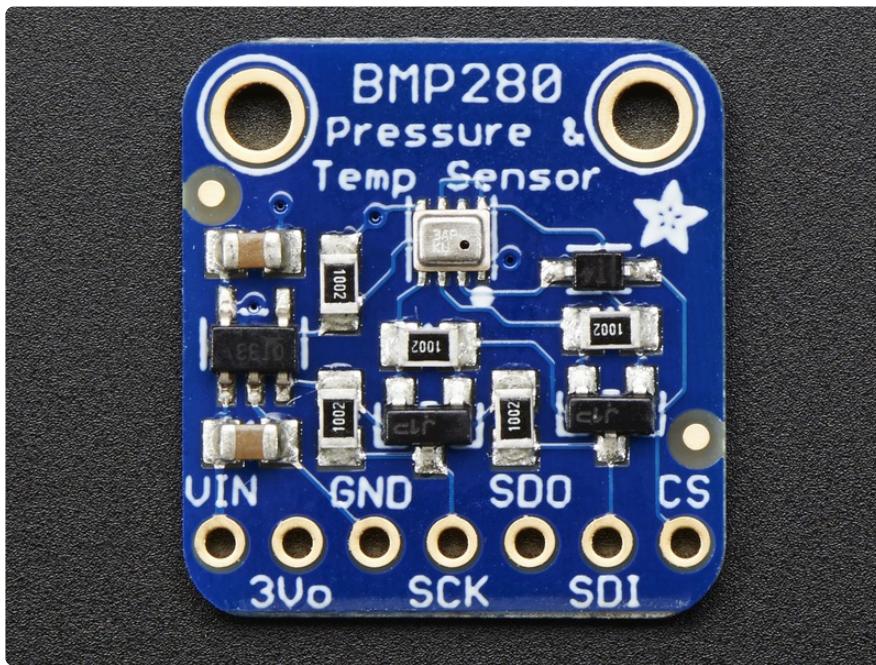
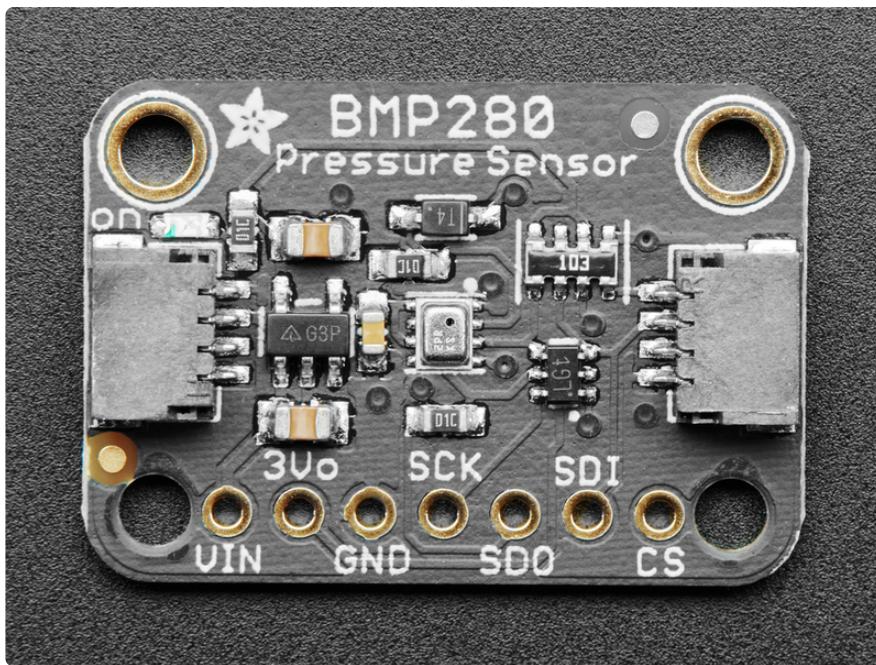
wide range of other sensors and accessories using a [compatible cable](https://adafru.it/JnB) (<https://adafru.it/JnB>). We've of course broken out all the pins to standard headers and added a voltage regulator and level shifting so allow you to use it with either 3.3V or 5V systems such as the Metro M4 or Arduino Uno respectively.

We even wrote up a nice tutorial with wiring diagrams, schematics, libraries and examples to get you running in 10 minutes! [Make sure to check the tutorial for example code for Arduino and CircuitPython, pinouts, assembly, wiring, downloads, and more!](https://adafru.it/MbA) (<https://adafru.it/MbA>)

There are two versions of this board - the STEMMA QT version shown above, and the original header-only version shown below. Code works the same on both!



Pinouts



Power Pins:

- **Vin** - this is the power pin. Since the sensor chip uses 3 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
- **3V0** - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like

- **GND** - common ground for power and logic

SPI Logic pins:

All pins going into the breakout have level shifting circuitry to make them 3-5V logic level safe. Use whatever logic level is on **Vin**!

- **SCK** - This is the **SPI Clock** pin, its an input to the chip
- **SDO** - this is the **Serial Data Out / Microcontroller In Sensor Out** pin, for data sent from the BMP280 to your processor
- **SDI** - this is the **Serial Data In / Microcontroller Out Sensor In** pin, for data sent from your processor to the BMP280
- **CS** - this is the **Chip Select** pin, drop it low to start an SPI transaction. Its an input to the chip

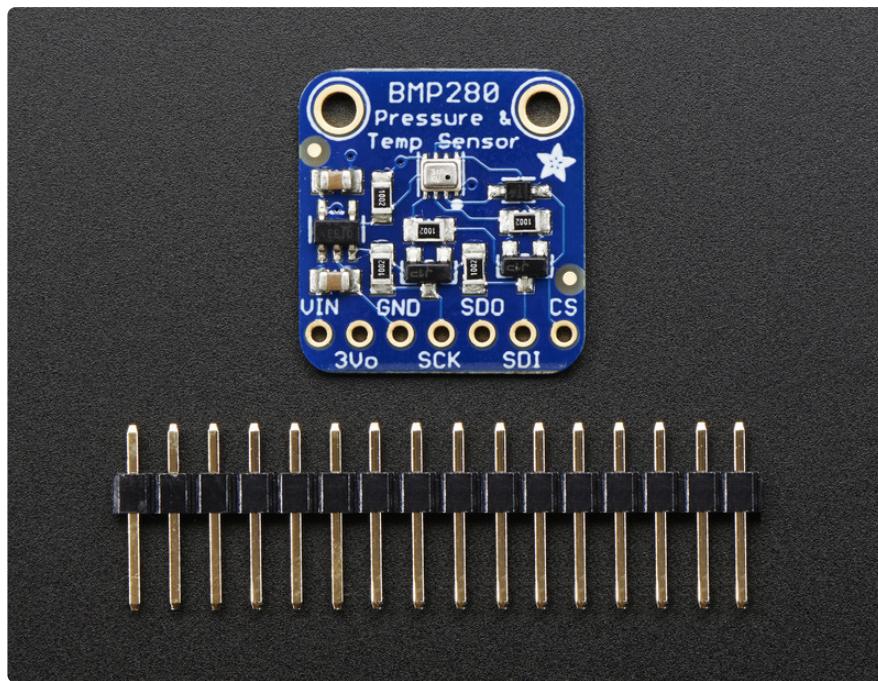
If you want to connect multiple BMP280's to one microcontroller, have them share the SDI, SDO and SCK pins. Then assign each one a unique CS pin.

I2C Logic pins:

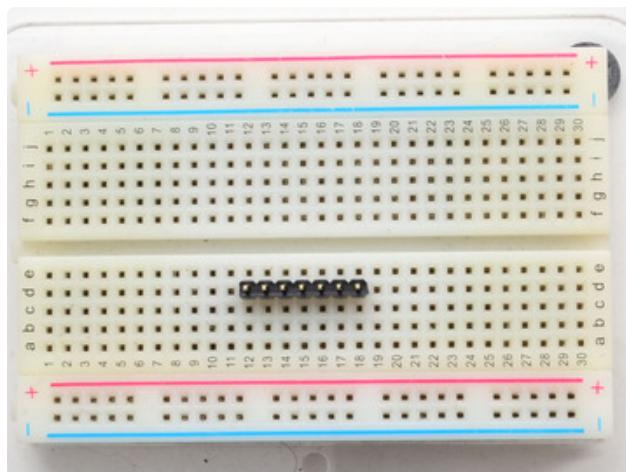
- **SCK** - this is also the I2C clock pin (**SCL**), connect to your microcontroller's I2C clock line.
- **SDI** - this is also the I2C data pin (**SDA**), connect to your microcontroller's I2C data line.

Leave the other pins disconnected

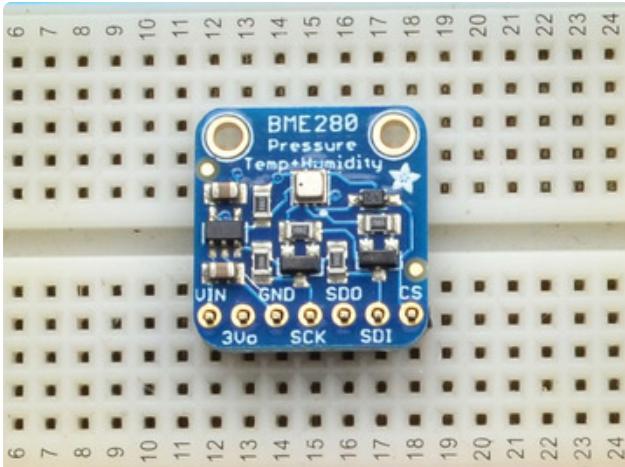
Assembly



The assembly pix use the BME280 but it is identically shaped/sized as the BMP280

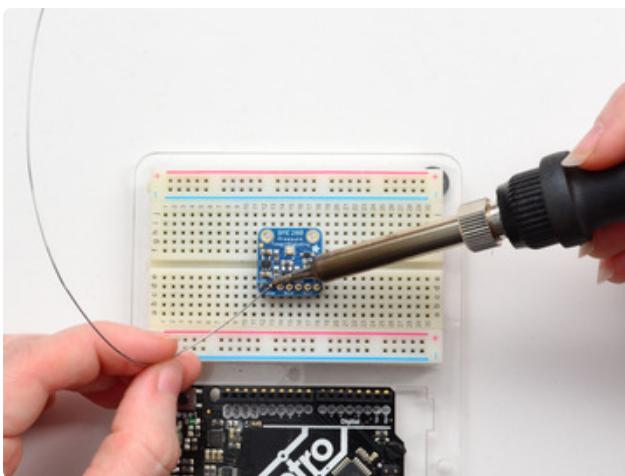


Prepare the header strip:
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**



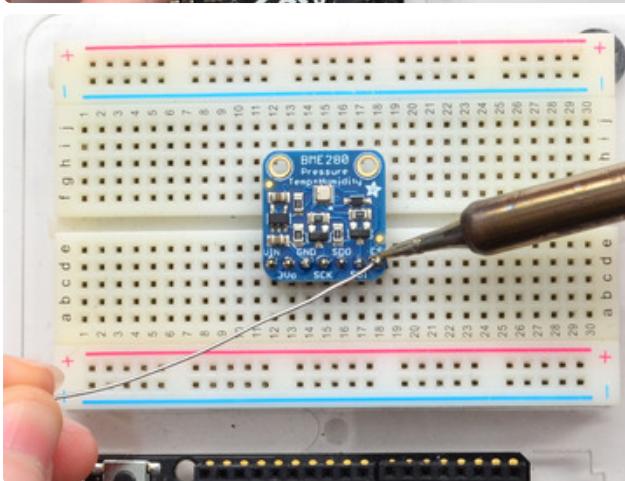
Add the breakout board:

Place the breakout board over the pins so that the short pins poke through the breakout pads



And Solder!

Be sure to solder all pins for reliable electrical contact.



(For tips on soldering, be sure to check out our [Guide to Excellent Soldering](https://adafru.it/aTk) (<https://adafru.it/aTk>)).



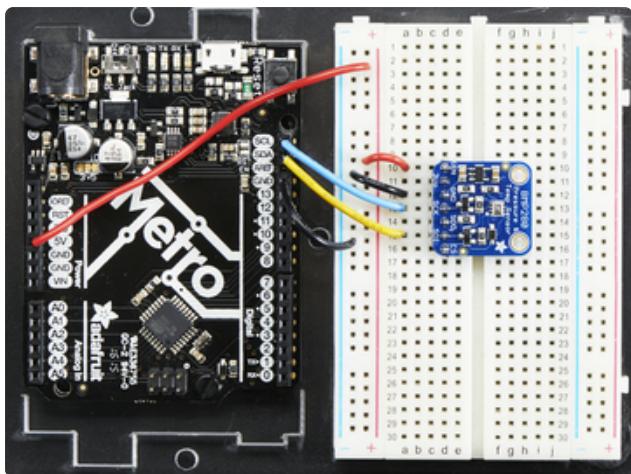
You're done! Check your solder joints visually and continue onto the next steps

Arduino Test

You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, as long as you have 4 available pins it is possible to 'bit-bang SPI' or you can use two I2C pins, but usually those pins are fixed in hardware. Just check out the library, then port the code.

I2C Wiring

Use this wiring if you want to connect via I2C interface

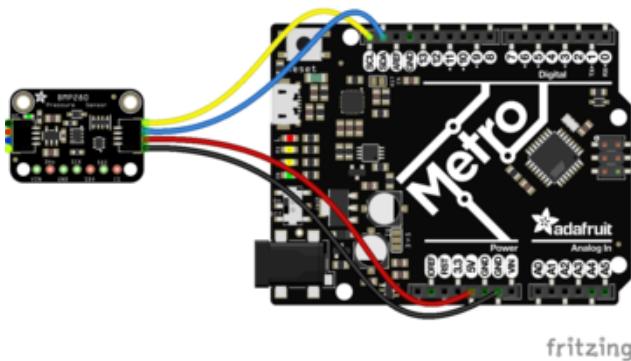
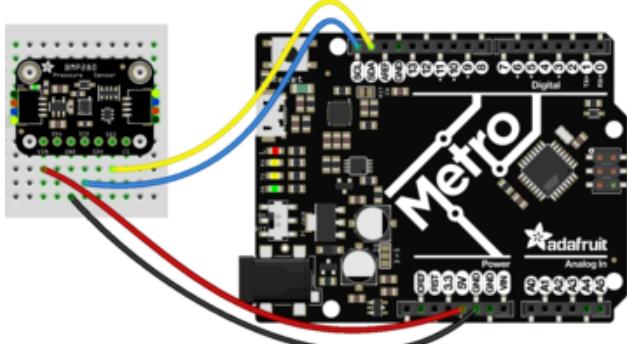


Connect **Vin** (red wire on **STEMMA version**) to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V

Connect **GND** (black wire on **STEMMA version**) to common power/data ground

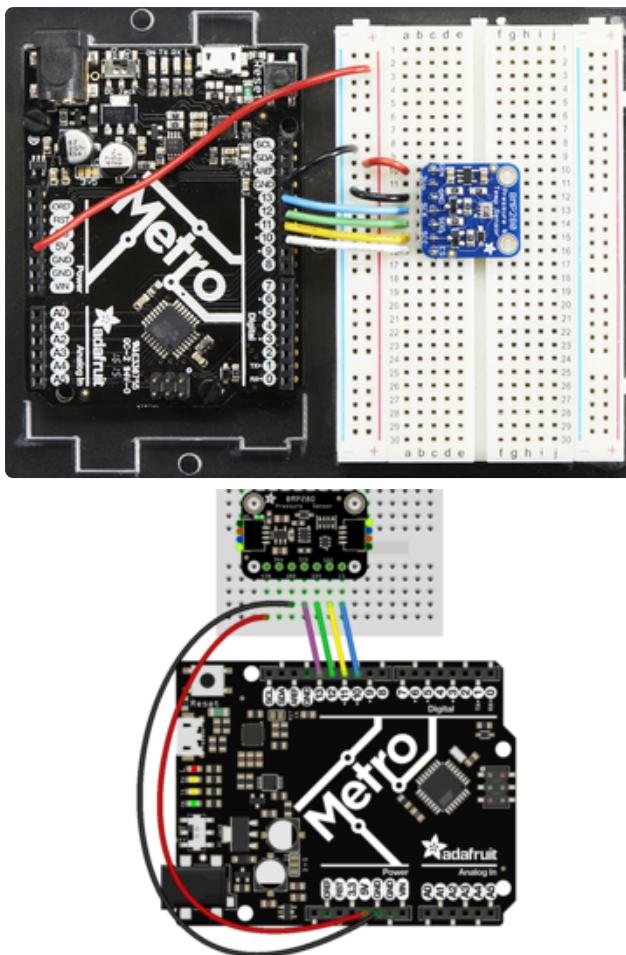
Connect the **SCK** (yellow wire on **STEMMA version**) pin to the I2C clock **SCL** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A5**, on a Mega it is also known as **digital 21** and on a Leonardo/Micro, **digital 3**

Connect the **SDI** (blue wire on **STEMMA version**) pin to the I2C data **SDA** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A4**, on a Mega it is also known as **digital 20** and on a Leonardo/Micro, **digital 2**



SPI Wiring

Since this is a SPI-capable sensor, we can use hardware or 'software' SPI. To make wiring identical on all Arduinos, we'll begin with 'software' SPI. The following pins should be used:



Connect **Vin** to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V

Connect **GND** to common power/data ground

Connect the **SCK** pin to **Digital #13** but any pin can be used later

Connect the **SDO** pin to **Digital #12** but any pin can be used later

Connect the **SDI** pin to **Digital #11** but any pin can be used later

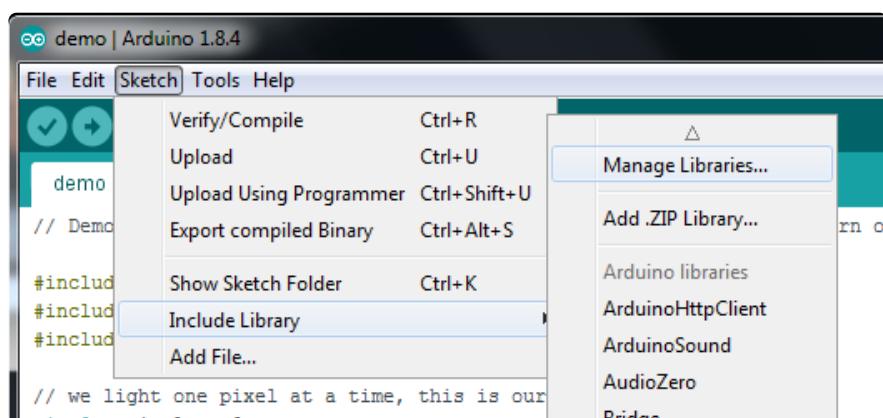
Connect the **CS** pin **Digital #10** but any pin can be used later

Later on, once we get it working, we can adjust the library to use hardware SPI if you desire, or change the pins to other

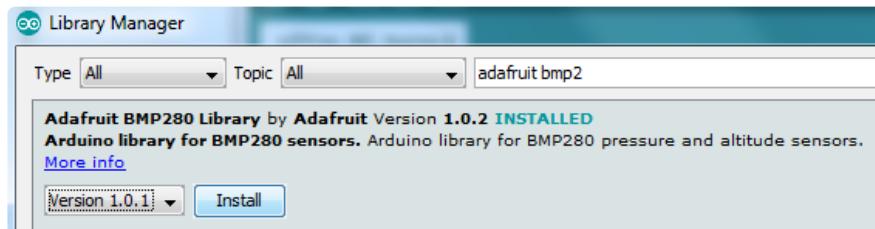
Download Adafruit_BMP280 library

To begin reading sensor data, you will need to [install the Adafruit_BMP280 library \(code on our github repository\)](#) (<https://adafru.it/fIK>). It is available from the Arduino library manager so we recommend using that.

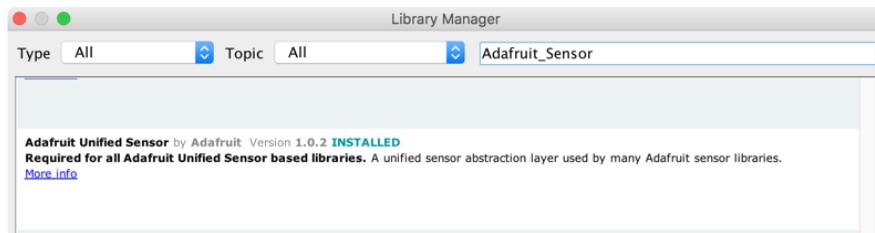
From the IDE open up the library manager...



And type in **adafruit bmp280** to locate the library. Click **Install**



You'll also need to install the **Adafruit Unified Sensor** library

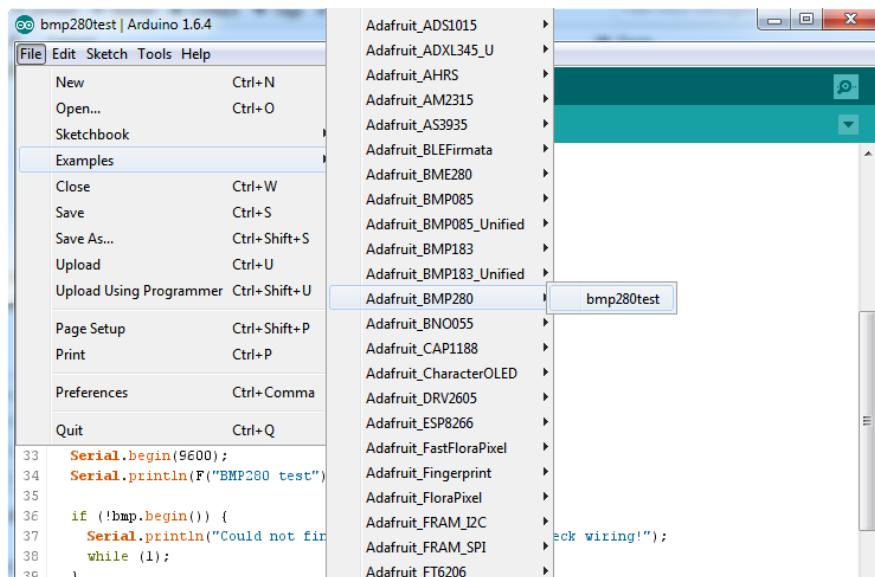


We also have a great tutorial on Arduino library installation at:

<http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use> (<https://adafru.it/aYM>)

Load Demo

Open up **File->Examples->Adafruit_BMP280->bmp280test** and upload to your Arduino wired up to the sensor



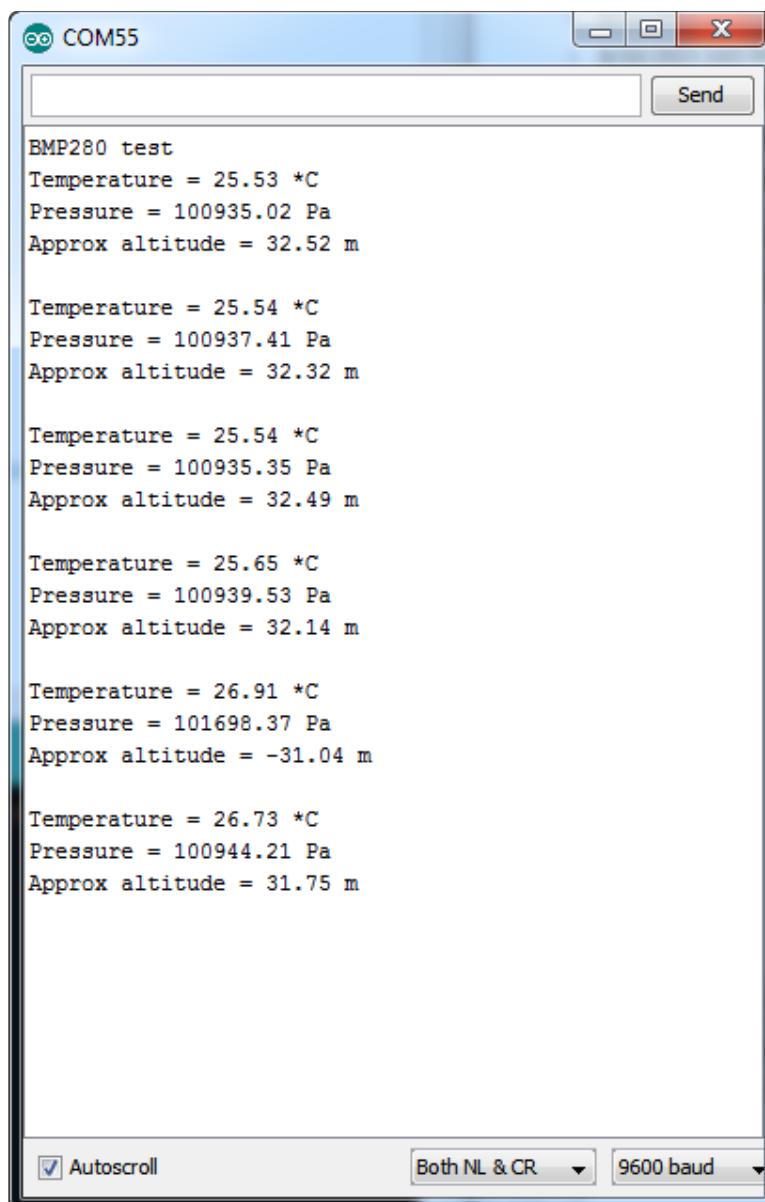
Depending on whether you are using I2C or SPI, change the pin names and comment or uncomment the following lines.

```
#define BMP_SCK 13  
#define BMP_MISO 12
```

```
#define BMP_MOSI 11
#define BMP_CS 10

Adafruit_BMP280 bmp; // I2C
//Adafruit_BMP280 bmp(BMP_CS); // hardware SPI
//Adafruit_BMP280 bmp(BMP_CS, BMP_MOSI, BMP_MISO, BMP_SCK);
```

Once uploaded to your Arduino, open up the serial console at 9600 baud speed to see data being printed out



Temperature is calculated in degrees C, you can convert this to F by using the classic $F = C \times 9/5 + 32$ equation.

Pressure is returned in the SI units of **Pascals**. 100 Pascals = 1 hPa = 1 millibar. Often times barometric pressure is reported in millibar or inches-mercury. For future reference 1 pascal = 0.000295333727 inches of mercury, or 1 inch Hg = 3386.39 Pascal. So if you take the pascal value of say 100734 and divide by 3389.39 you'll get

29.72 inches-Hg.

You can also calculate Altitude. **However, you can only really do a good accurate job of calculating altitude if you know the hPa pressure at sea level for your location and day!** The sensor is quite precise but if you do not have the data updated for the current day then it can be difficult to get more accurate than 10 meters.

Library Reference

You can start out by creating a BMP280 object with either software SPI (where all four pins can be any I/O) using

```
Adafruit_BMP280 bmp(BMP_CS, BMP_MOSI, BMP_MISO, BMP_SCK);
```

Or you can use hardware SPI. With hardware SPI you must use the hardware SPI pins for your Arduino - and each arduino type has different pins! [Check the SPI reference to see what pins to use. \(<https://adafru.it/d5h>\)](#)

In this case, you can use any CS pin, but the other three pins are fixed

```
Adafruit_BMP280 bmp(BMP_CS); // hardware SPI
```

or I2C using the default I2C bus, no pins are assigned

```
Adafruit_BMP280 bmp; // I2C
```

Once started, you can initialize the sensor with

```
if (!bmp.begin()) {  
    Serial.println("Could not find a valid BMP280 sensor, check wiring!");  
    while (1);  
}
```

begin() will return True if the sensor was found, and False if not. If you get a False value back, check your wiring!

Reading temperature and pressure is easy, just call:

```
bmp.readTemperature()  
bmp.readPressure()
```

Temperature is always a floating point, in Centigrade. Pressure is a 32 bit integer with the pressure in Pascals. You may need to convert to a different value to match it with your weather report.

It's also possible to turn the BMP280 into an altimeter. If you know the pressure at sea level, the library can calculate the current barometric pressure into altitude

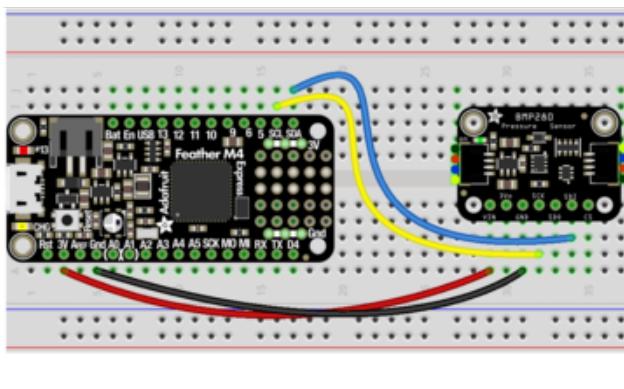
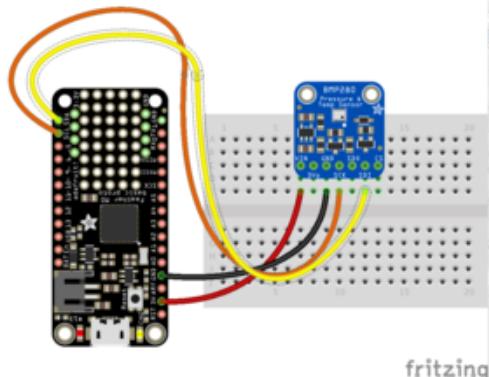
Python & CircuitPython Test

It's easy to use the BMP280 sensor with CircuitPython and the [Adafruit CircuitPython BMP280 \(<https://adafru.it/C0x>\)](#) module. This module allows you to easily write Python code that reads the temperature and pressure from the sensor.

You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python [thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library \(<https://adafru.it/BSN>\)](#).

CircuitPython Microcontroller Wiring

First wire up a BMP280 to your board exactly as shown on the previous pages for Arduino. You can use either I2C or SPI wiring, although it's recommended to use I2C for simplicity. Here's an example of wiring a Feather to the sensor with I2C:

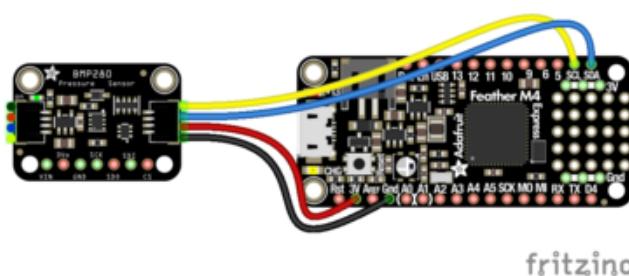


Board 3V to sensor VIN (red wire on STEMMA version)

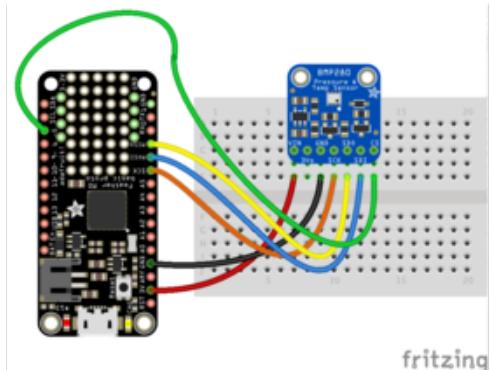
Board GND to sensor GND (black wire on STEMMA version)

Board SCL to sensor SCK (yellow wire on STEMMA version)

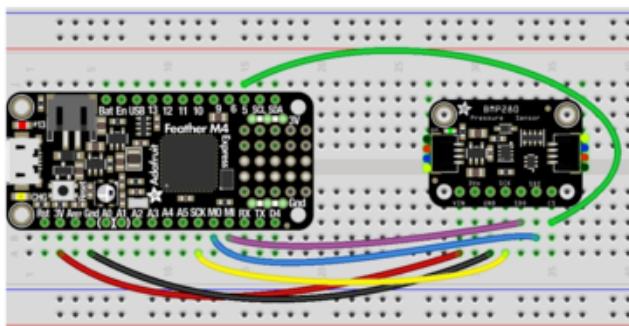
Board SDA to sensor SDI (blue wire on STEMMA version)



And an example of a Feather M0 wired with hardware SPI:



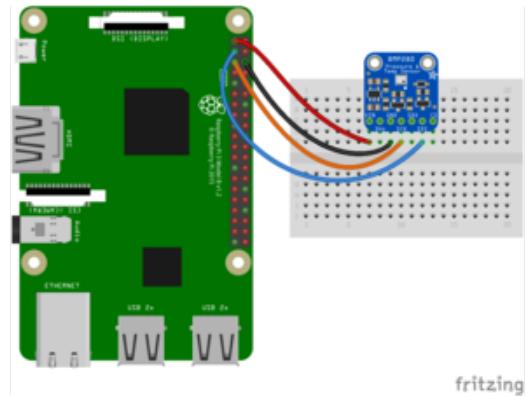
Board 3V to sensor VIN
Board GND to sensor GND
Board SCK to sensor SCK
Board MOSI to sensor SDI
Board MISO to sensor SDO
Board D5 to sensor CS (or use any other free digital I/O pin)



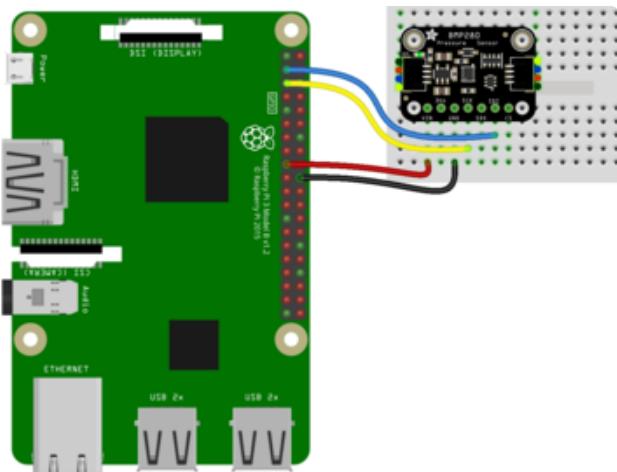
Python Computer Wiring

Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported \(<https://adafruit.it/BSN>\)](#).

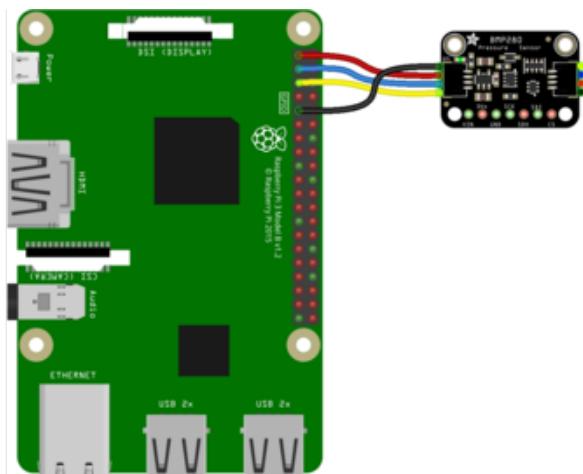
Here's the Raspberry Pi wired with I2C:



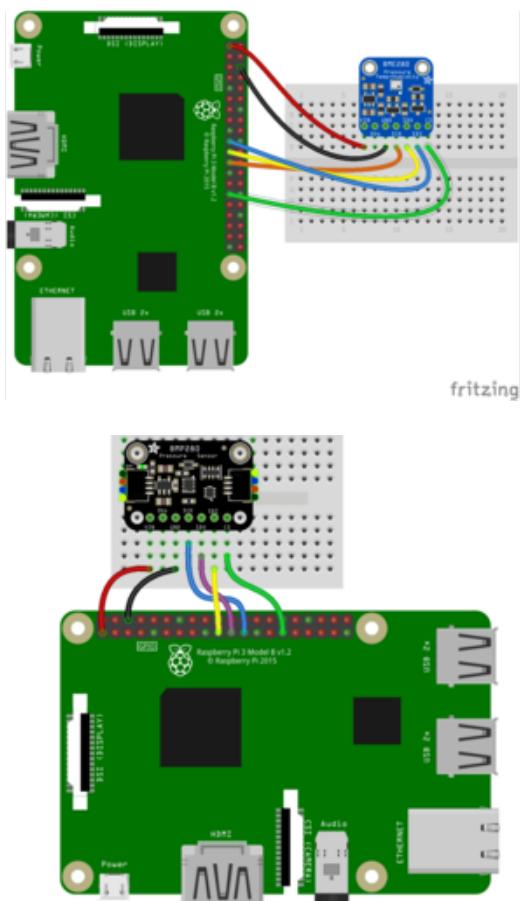
fritzing



Pi 3V3 to sensor VIN (red wire on STEMMA version)
Pi GND to sensor GND (black wire on STEMMA version)
Pi SCL to sensor SCK (yellow wire on STEMMA version)
Pi SDA to sensor SDI (blue wire on STEMMA version)



And an example on the Raspberry Pi 3 Model B wired with SPI:



Pi 3V3 to sensor VIN
Pi GND to sensor GND
Pi MOSI to sensor SDI
Pi MISO to sensor SDO
Pi SCLK to sensor SCK
Pi #5 to sensor CS (or use any other free GPIO pin)

CircuitPython Installation of BMP280 Library

You'll need to install the [Adafruit CircuitPython BMP280](https://adafru.it/C0x) (<https://adafru.it/C0x>) library on your CircuitPython board.

First make sure you are running the [latest version of Adafruit CircuitPython](https://adafru.it/Amd) (<https://adafru.it/Amd>) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from [Adafruit's CircuitPython library bundle](https://adafru.it/uap) (<https://adafru.it/uap>). Our CircuitPython starter guide has [a great page on how to install the library bundle](https://adafru.it/ABU) (<https://adafru.it/ABU>).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- `adafruit_bmp280.mpy`
- `adafruit_bus_device`

Before continuing make sure your board's lib folder or root filesystem has the `adafruit_bmp280.mpy`, and `adafruit_bus_device` files and folders copied over.

Next [connect to the board's serial REPL](https://adafru.it/Awz) (<https://adafru.it/Awz>) so you are at the CircuitPython >>> prompt.

Python Installation of BMP280 Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. [Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready](https://adafru.it/BSN) (<https://adafru.it/BSN>)!

Once that's done, from your command line run the following command:

- `sudo pip3 install adafruit-circuitpython-bmp280`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython & Python Usage

To demonstrate the usage of the sensor we'll initialize it and read the temperature, humidity, and more from the board's Python REPL.

If you're using an I2C connection run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import board
import adafruit_bmp280
i2c = board.I2C()
sensor = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)
```

Or if you're using a SPI connection run this code instead to setup the SPI connection and sensor:

```
import board
import digitalio
import adafruit_bmp280
spi = board.SPI()
cs = digitalio.DigitalInOut(board.D5)
sensor = adafruit_bmp280.Adafruit_BMP280_SPI(spi, cs)
```

Now you're ready to read values from the sensor using any of these properties:

- **temperature** - The sensor temperature in degrees Celsius.
- **pressure** - The pressure in hPa.
- **altitude** - The altitude in meters.

For example to print temperature and pressure:

```
print('Temperature: {} degrees C'.format(sensor.temperature))
print('Pressure: {}hPa'.format(sensor.pressure))
```

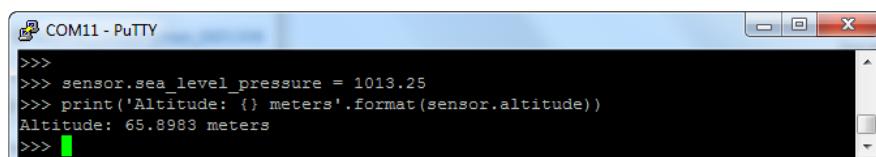
```
>>> print('Temperature: {} degrees C'.format(sensor.temperature))
Temperature: 21.0874 degrees C
>>> print('Pressure: {}hPa'.format(sensor.pressure))
Pressure: 1012.32hPa
>>> █
```

For altitude you'll want to set the pressure at sea level for your location to get the most accurate measure (remember these sensors can only infer altitude based on pressure and need a set calibration point). Look at your local weather report for a pressure at sea level reading and set the **seaLevelhPA** property:

```
sensor.sea_level_pressure = 1013.25
```

Then read the altitude property for a more accurate altitude reading (but remember this altitude will fluctuate based on atmospheric pressure changes!):

```
print('Altitude: {} meters'.format(sensor.altitude))
```



That's all there is to using the BMP280 sensor with CircuitPython!

Here's a starting example that will print out the temperature, pressure and altitude every 2 seconds:

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""Simpletest Example that shows how to get temperature,
   pressure, and altitude readings from a BMP280"""
import time
import board

# import digitalio # For use with SPI
import adafruit_bmp280

# Create sensor object, communicating over the board's default I2C bus
i2c = board.I2C()    # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
# microcontroller
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)

# OR Create sensor object, communicating over the board's default SPI bus
# spi = board.SPI()
# bmp_cs = digitalio.DigitalInOut(board.D10)
# bmp280 = adafruit_bmp280.Adafruit_BMP280_SPI(spi, bmp_cs)

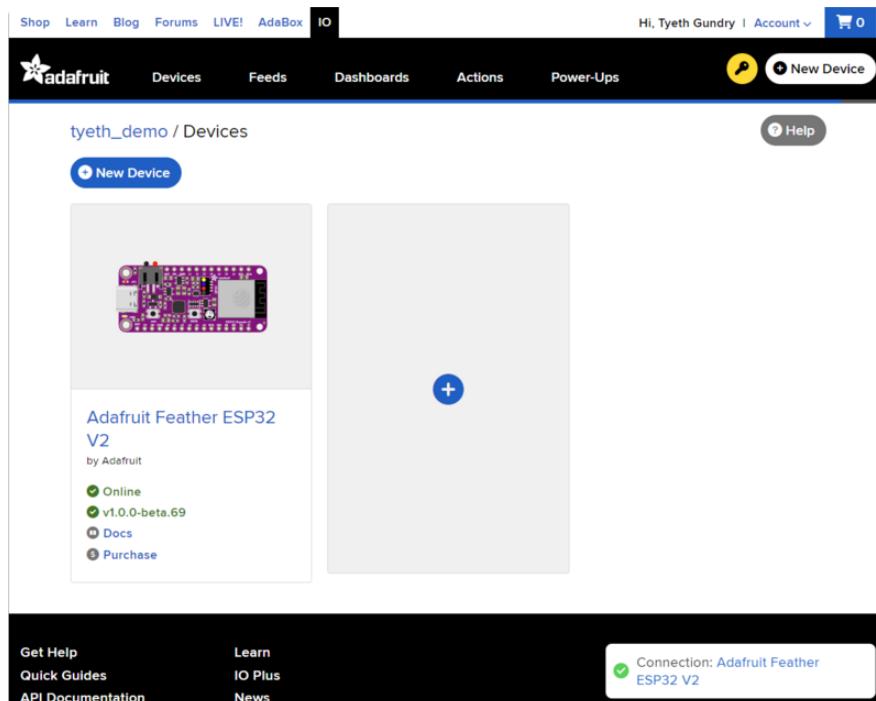
# change this to match the location's pressure (hPa) at sea level
bmp280.sea_level_pressure = 1013.25

while True:
    print("\nTemperature: %0.1f C" % bmp280.temperature)
    print("Pressure: %0.1f hPa" % bmp280.pressure)
    print("Altitude = %0.2f meters" % bmp280.altitude)
    time.sleep(2)
```

Python Docs

[Python Docs \(https://adafru.it/C0B\)](https://adafru.it/C0B)

WipperSnapper



What is WipperSnapper

WipperSnapper is a firmware designed to turn any WiFi-capable board into an Internet-of-Things device without programming a single line of code. WipperSnapper connects to [Adafruit IO](https://adafru.it/fsU) (<https://adafru.it/fsU>), a web platform designed ([by Adafruit!](https://adafru.it/Bo5) (<https://adafru.it/Bo5>)) to display, respond, and interact with your project's data.

Simply load the WipperSnapper firmware onto your board, add credentials, and plug it into power. Your board will automatically register itself with your Adafruit IO account.

From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

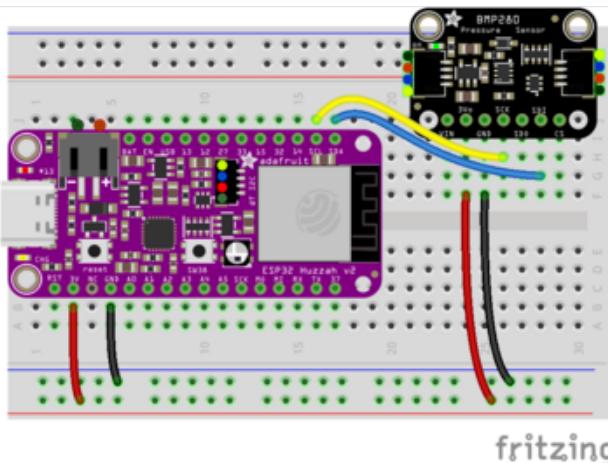
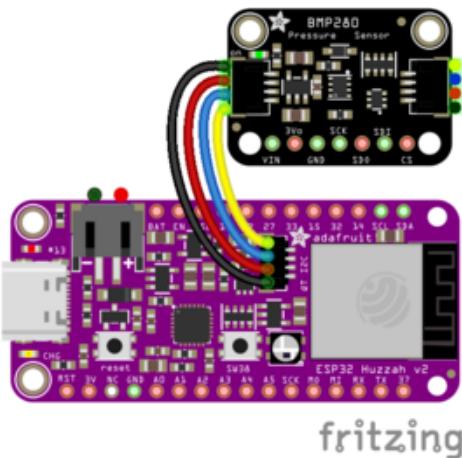
If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

Quickstart: Adafruit IO
WipperSnapper

<https://adafru.it/Vfd>

Wiring

First, wire up a BMP280 to your board exactly as follows. Here is an example of the BMP280 wired to an [Adafruit ESP32 Feather V2](http://adafru.it/5400) (<http://adafru.it/5400>) using I2C with a [STEMMA QT cable](http://adafru.it/4210) (no soldering required) (<http://adafru.it/4210>)



Board 3V to sensor VIN (red wire on STEMMA QT)

Board GND to sensor GND (black wire on STEMMA QT)

Board SCL to sensor SCK (yellow wire on STEMMA QT)

Board SDA to sensor SDI (blue wire on STEMMA QT)

Usage

Connect your board to Adafruit IO Wippersnapper and [navigate to the Wippersnapper board list](https://adafru.it/TAu) (<https://adafru.it/TAu>).

On this page, **select the Wippersnapper board you're using** to be brought to the board's interface page.

The screenshot shows the Adafruit IO web interface. At the top, there's a navigation bar with links for Shop, Learn, Blog, Forums, LIVE!, AdaBox, and IO. On the right, it says "Hi, Tyeth Gundry | Account" and shows a cart icon with "0". Below the navigation is a header with the Adafruit logo and menu items: Devices, Feeds, Dashboards, Actions, Power-Ups, a key icon, and a "New Device" button. The main area is titled "tyeth_demo / Devices". A "New Device" button is at the top left. On the right, there's a "Help" link. In the center, there's a device tile for an "Adafruit Feather ESP32 V2" by Adafruit. The tile shows a small image of the board, its name, the fact that it's online, its firmware version (v1.0.0-beta.71), a "Docs" link, and a "Purchase" link. To the right of the tile is a large empty space with a blue plus sign.

If you do not see your board listed here - you need [to connect your board to Adafruit IO](#) (<https://adafru.it/Vfd>) first.

This screenshot shows the detailed view of the Adafruit Feather ESP32 V2 device page. The title is "Adafruit Feather ESP32 V2" and it's "by Adafruit". It has status indicators: "Online" (green checkmark) and "v1.0.0-beta.69" (green checkmark). There are also "Docs" and "Purchase" links. A red arrow points to the firmware version "v1.0.0-beta.69".

This screenshot shows the same device page, but the board is now disconnected. The "Online" status is now red with an exclamation mark, and the firmware version "v1.0.0-beta.68" is red with an exclamation mark. There is a blue "Update" button next to it. A hand cursor icon is over the "Update" button. The "Docs" and "Purchase" links are also present.

On the device page, quickly **check that you're running the latest version of the WipperSnapper firmware**.

The device tile on the left indicates the version number of the firmware running on the connected board.

If the **firmware version is green with a checkmark** - continue with this guide.
If the **firmware version is red with an exclamation mark "!"** - [update to the latest WipperSnapper firmware](#) (<https://adafru.it/Vfd>) on your board before continuing.

Next, make sure the sensor is plugged into your board and click the **I2C Scan** button.

The screenshot shows the Adafruit IO web interface. At the top, there's a navigation bar with links for Devices, Feeds, Dashboards, Actions, and Power-Ups. Below the navigation, the URL is shown as brubell / Devices / Adafruit Feather ESP32 V2. There are three buttons at the top: 'New Component' (blue), 'I2C Scan' (blue with a red arrow pointing to it), and 'Device Settings' (grey). On the left, there's a thumbnail image of the Adafruit Feather ESP32 V2 board. To the right of the board is a large empty area with a blue plus sign. Below the board thumbnail, the text 'Adafruit Feather ESP32...' and 'Adafruit Feather ESP32 V2 by Adafruit' is displayed.

You should see the BMP280's default I2C address of `0x77` pop-up in the I2C scan list.

The screenshot shows a modal dialog titled 'I2C Scan Complete'. It contains a table with rows labeled from 00 to 7F and columns labeled from 0 to f. The cell at row 77, column 7 is highlighted with a yellow background. At the bottom of the dialog are two buttons: 'Close' and 'Scan Again'.

	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

Close **Scan Again**

I don't see the sensor's I2C address listed!

First, double-check the connection and/or wiring between the sensor and the board.

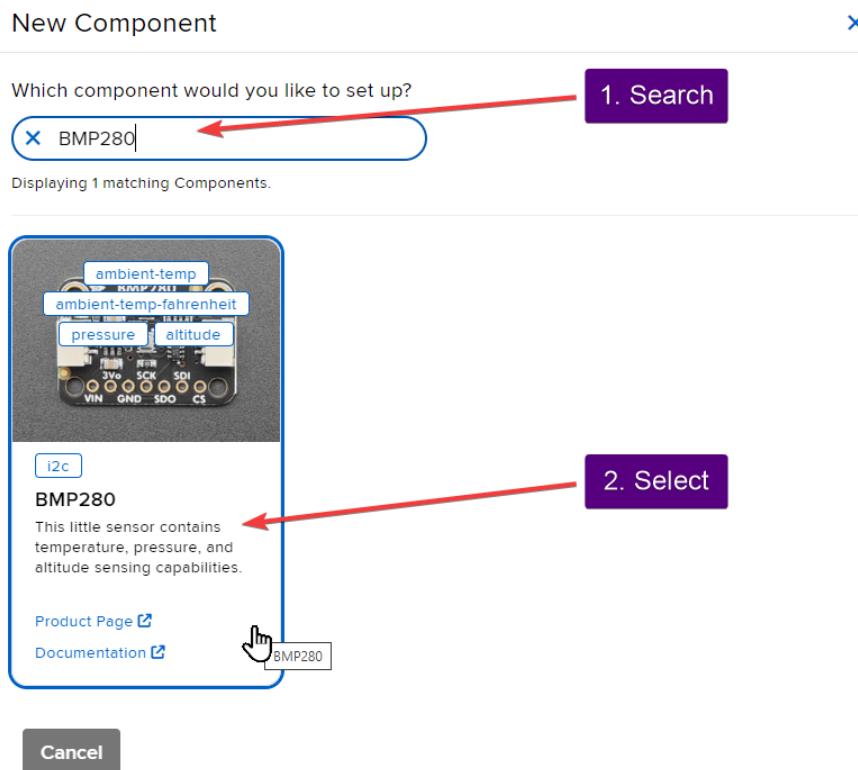
Then, reset the board and let it re-connect to Adafruit IO WipperSnapper.

With the sensor detected in an I2C scan, you're ready to add the sensor to your board.

Click the New Component button or the + button to bring up the component picker.



Adafruit IO supports a large amount of components. To quickly find your sensor, type **BMP280** into the search bar, then select the **BMP280** component.



On the component configuration page, the BMP280's sensor address should be listed along with the sensor's settings.

The **Send Every** option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the BMP280's sensors and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

For this example, set the **Send Every** interval to every 30 seconds. Don't forget to scroll down as there are 4 sensor metrics / feeds to select. On a small screen you may only see the first 3 and then wonder why altitude isn't updating (it will still be set to the default of every 15 minutes)

Create BMP280 Component

X

Select I2C Address:

0x77

Enable BMP280: Temperature Sensor (°C)?

Name:

BMP280: Temperature Sensor (°C)

Send Every:

Every 30 seconds

Enable BMP280: Temperature Sensor (°F)?

Name:

BMP280: Temperature Sensor (°F)

Send Every:

Every 30 seconds

Enable BMP280: Pressure Sensor?

Name:

BMP280: Pressure Sensor

Send Every:

Every 30 seconds

Enable BMP280: Altitude (Relative)?

Name:



[← Back to Component Type](#)

Create Component

Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.

The screenshot shows the Adafruit IO device configuration interface for an Adafruit Feather ESP32 V2. On the left, there's a sidebar with the device's name and status (Online, v1.0.0-beta.70, Docs, Purchase). Below that is a 'Report Bugs' link. The main area displays four sensor feeds:

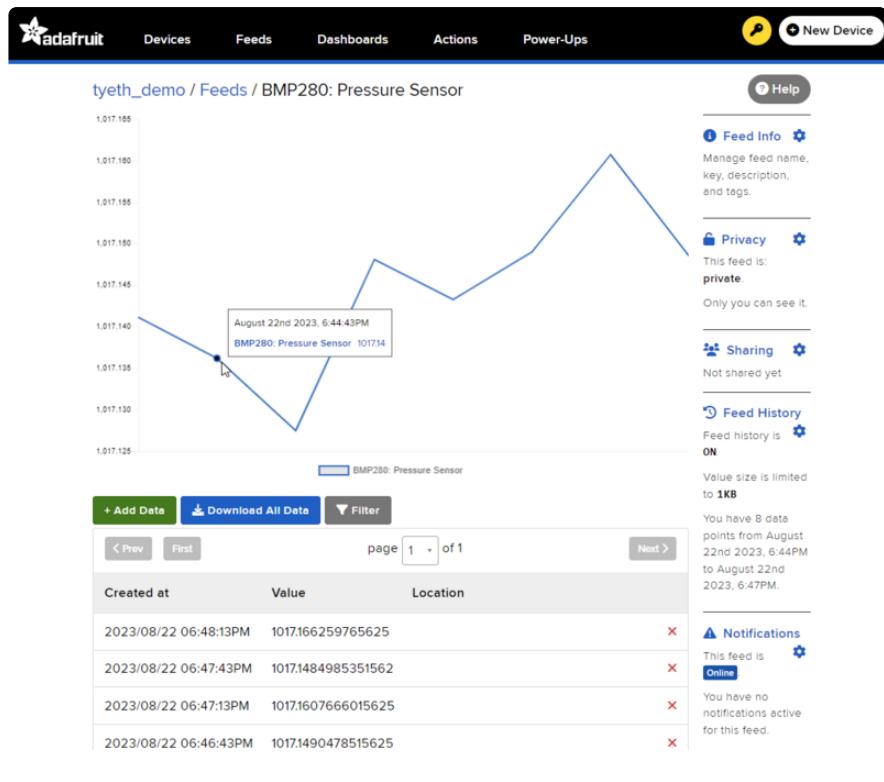
- BMP280: Altitude (Relative) [bmp280:altitude]: Shows a graph with a value of -32.35m.
- BMP280: Pressure Sensor [bmp280:pressure]: Shows a graph with a value of 1017.14hPa.
- BMP280: Temperature Sensor (°C) [bmp280:ambient-temp]: Shows a graph with a value of 26.32°C.
- BMP280: Temperature Sensor (°F) [bmp280:ambient-temp-fahrenheit]: Shows a graph with a value of 79.38°F.

Each feed has a 'Create Action' and 'Add to Dashboard' link below it, along with edit and settings icons.

To view the data that has been logged from the sensor, click on the graph next to the sensor name.

This screenshot is similar to the one above, showing the Adafruit IO device configuration for an Adafruit Feather ESP32 V2. The sidebar and feeds are identical. A red arrow points to the graph icon next to the 'BMP280: Pressure Sensor' feed, highlighting it.

Here you can see the feed history and edit things about the feed such as the name, privacy, webhooks associated with the feed and more. If you want to learn more about how feeds work, [check out this page \(https://adafru.it/10aZ\)](https://adafru.it/10aZ).



F.A.Q.

How come the altitude calculation is wrong? Is my sensor broken?

No, your sensor is likely just fine. The altitude calculation depends on knowing the barometric pressure at sea level

If you do not set the correct sea level pressure for your location FOR THE CURRENT DAY it will not be able to calculate the altitude accurately

Barometric pressure at sea level changes daily based on the weather!

If I have long delays between reads, the first data read seems wrong?

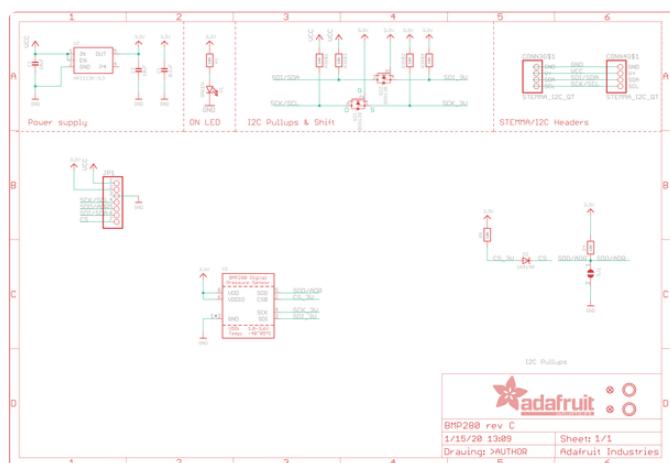
The BMx280 'saves' the last reading in memory for you to query. Just read twice in a row and toss out the first reading!

Downloads

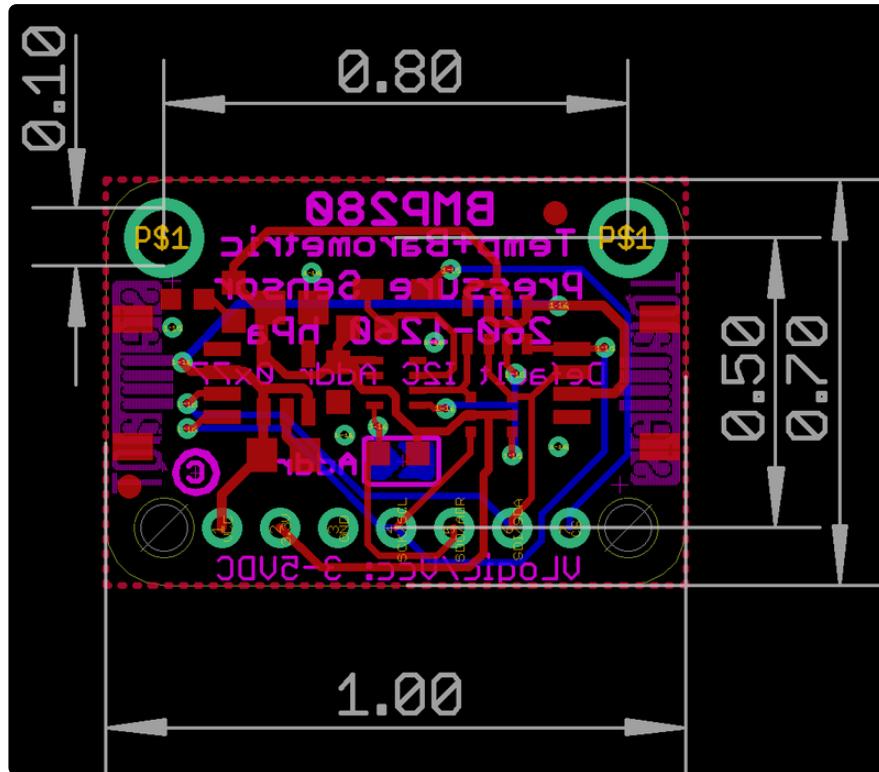
Documents

- [Datasheet for the BMP280 sensor used in the breakout](https://adafru.it/fIO) (<https://adafru.it/fIO>)
- [Arduino BMP280 Driver](https://adafru.it/fIK) (<https://adafru.it/fIK>)
- Fritzing object in the Adafruit Fritzing Library (<https://adafru.it/Mbx>)
- [EagleCAD PCB files on GitHub](https://adafru.it/rDq) (<https://adafru.it/rDq>)

Schematic - STEMMA QT version

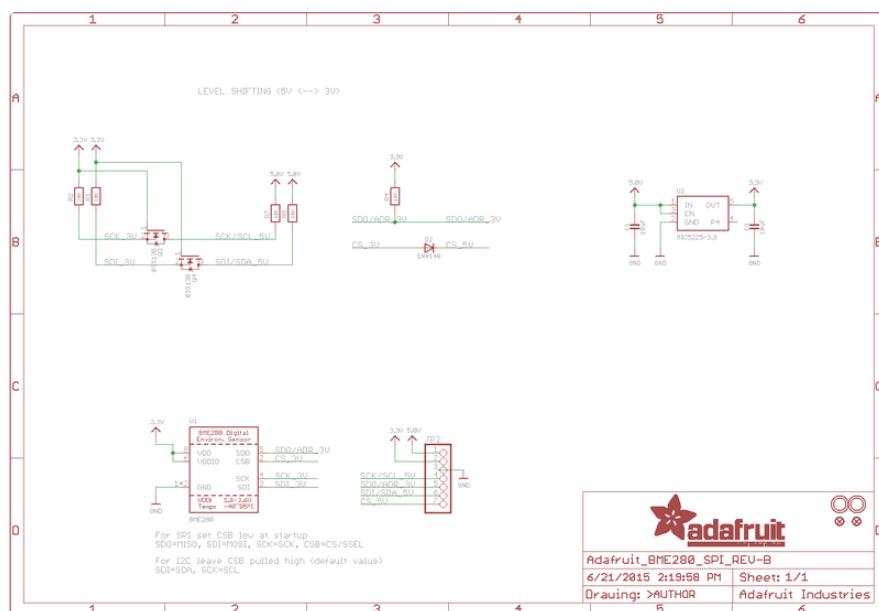


Fab Print - STEMMA QT version



Schematic - Original version

Click to enlarge. BMP280 shares the same package & pinout as the BME280 so the schematic is the same



Fab Print - Original version

In inches. BMP280 shares the same package & pinout as the BME280 so the layout is the same

