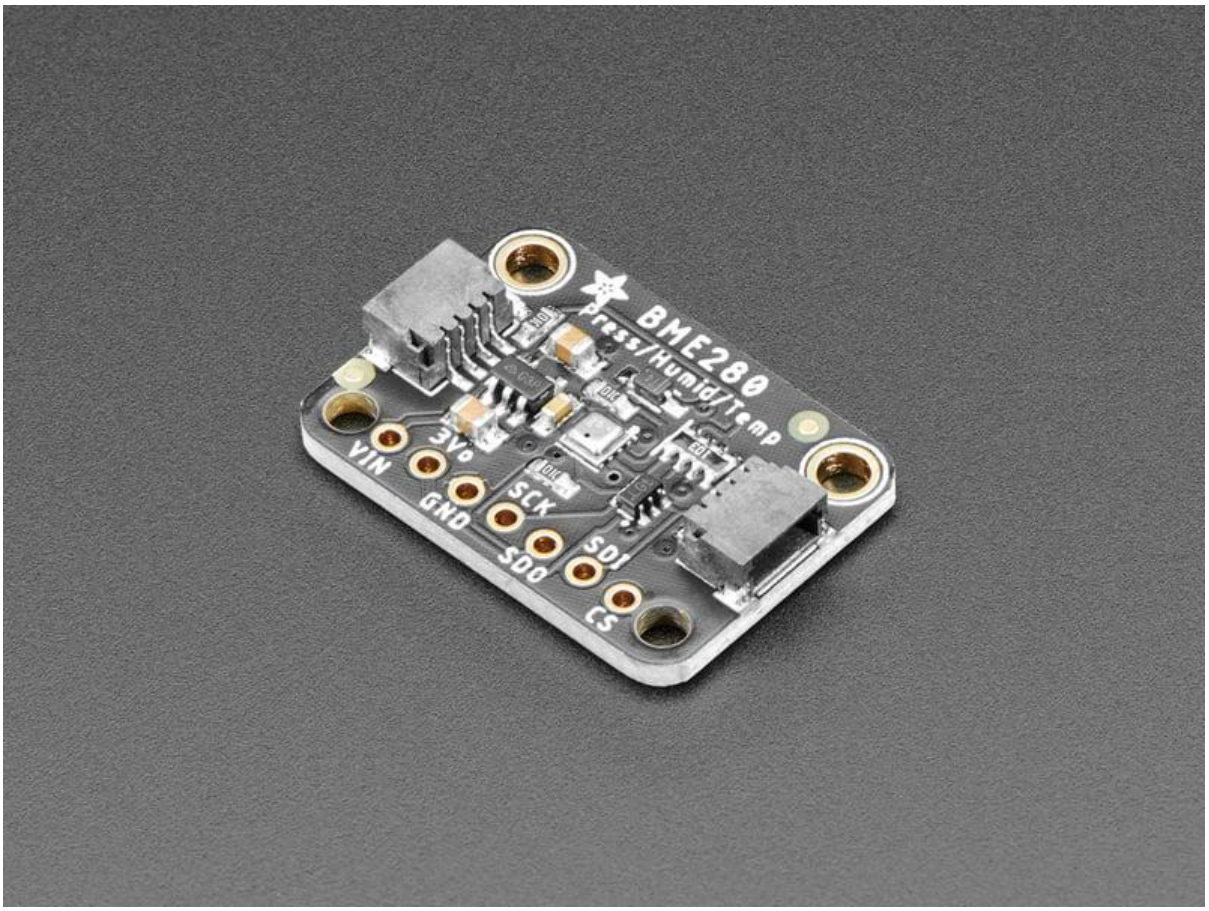
Adafruit BME280 Humidity + Barometric Pressure + Temperature Sensor Breakout

Created by lady ada



https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout

Last updated on 2023-08-29 02:58:21 PM EDT

Table of Contents

Overview 3

Pinouts 5

- Power Pins:
- SPI Logic pins:
- I2C Logic pins:
- I2C Address

Assembly 7

- Prepare the header strip:
- Add the breakout board:
- And Solder!

Arduino Test 9

- I2C Wiring
- SPI Wiring
- Install Adafruit_BME280 library
- Load Demo
- Library Reference

Python & CircuitPython Test 15

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- CircuitPython Installation of BME280 Library
- Python Installation of BME280 Library
- CircuitPython & Python Usage
- Full Example Code

Python Docs 23

F.A.Q. 23

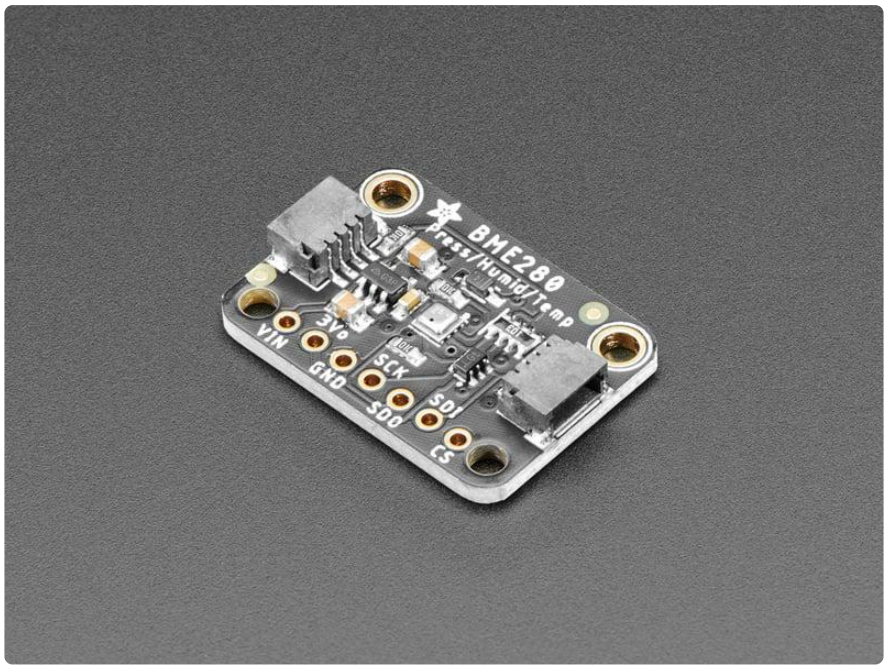
WipperSnapper 24

- What is WipperSnapper
- Wiring
- Usage

Downloads 31

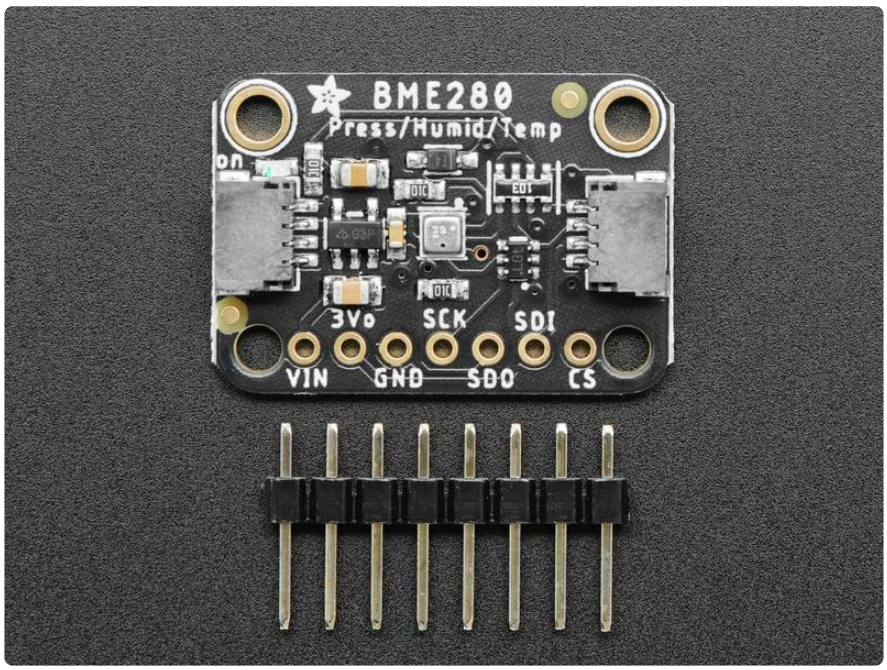
- Documents
- Alternative Driver (Python)
- Schematic and Fab Print for STEMMA QT Version
- Schematic and Fab Print for Original Version

Overview

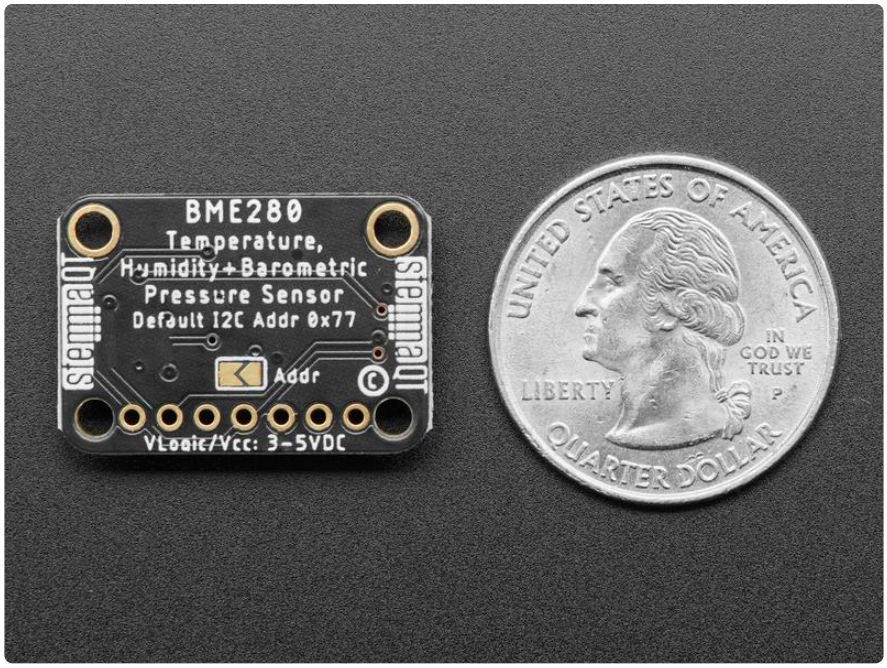


Bosch has stepped up their game with their new BME280 sensor, an environmental sensor with temperature, barometric pressure and humidity! This sensor is great for all sorts of weather/environmental sensing and can even be used in both I2C and SPI!

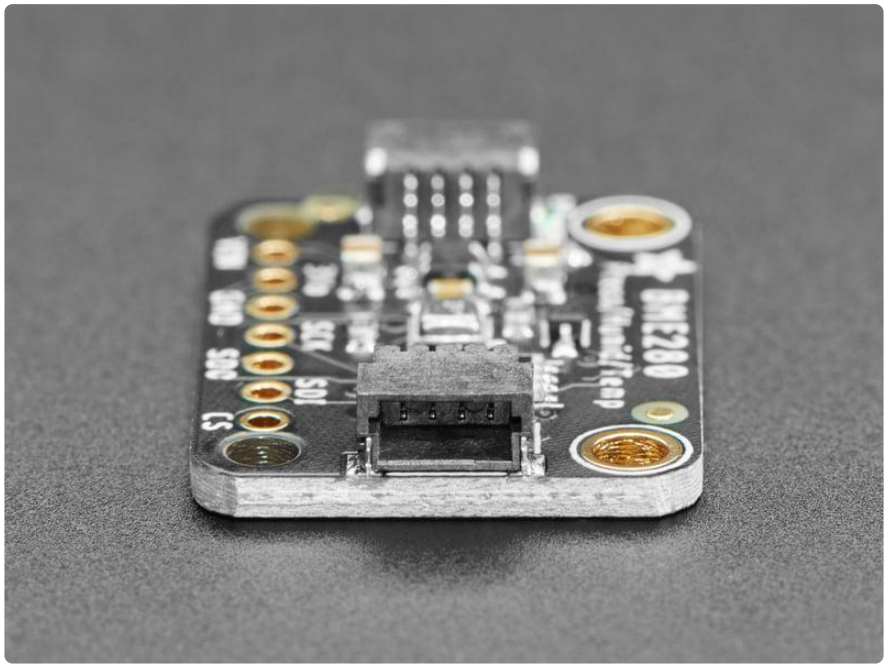
This precision sensor from Bosch is the best low-cost sensing solution for measuring humidity with ±3% accuracy, barometric pressure with ±1 hPa absolute accuraccy, and temperature with ±1.0°C accuracy. Because pressure changes with altitude, and the pressure measurements are so good, you can also use it as an altimeter with ±1 meter accuracy!



The BME280 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.



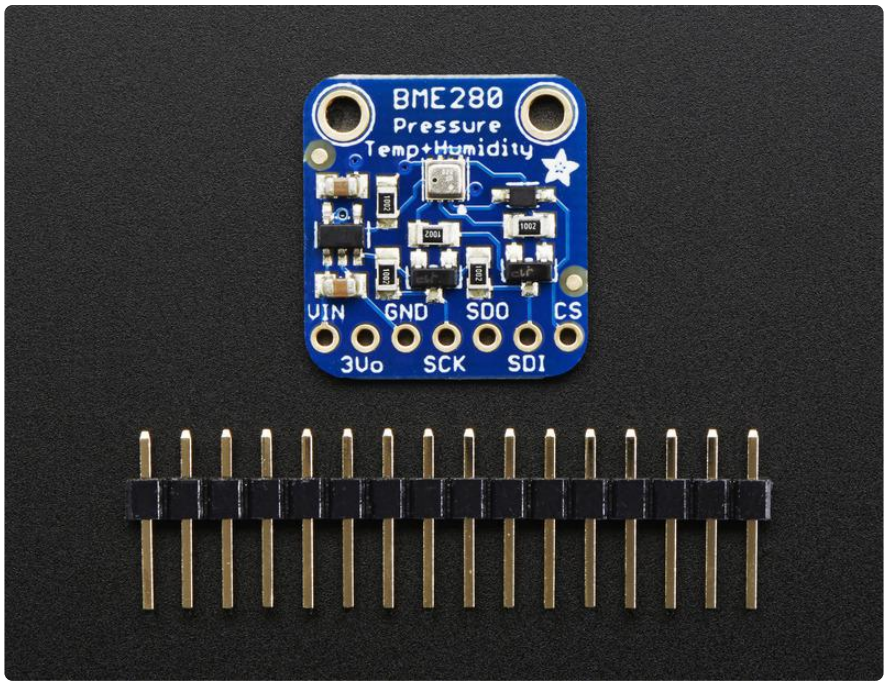
Nice sensor right? So we made it easy for you to get right into your next project. The surface-mount sensor is soldered onto a PCB and comes with a 3.3V regulator and level shifting so you can use it with a 3V or 5V logic microcontroller without worry.



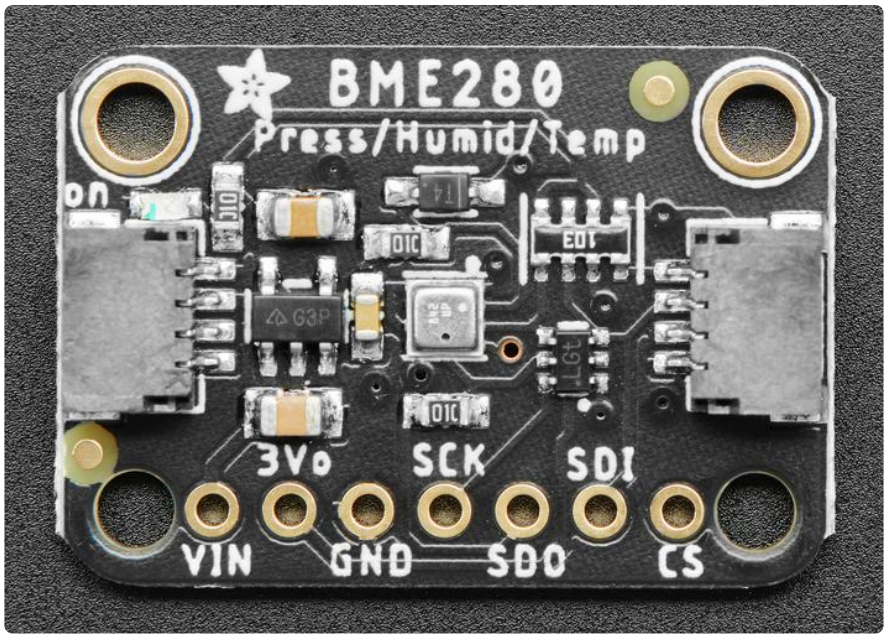
The breakout is made in the STEMMA QT form factor (), making them easy to interface with. The STEMMA QT connectors () on either side are compatible with the SparkFun

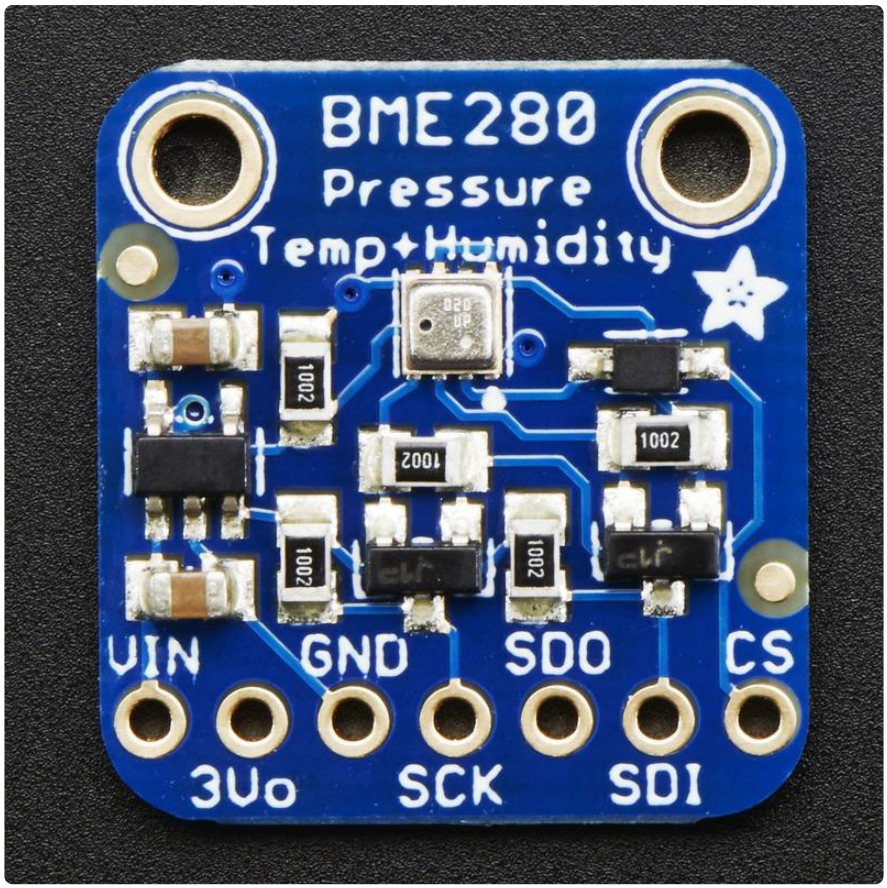
Qwiic () I2C connectors. This allows you to make solderless connections between your development board and the BME280 or to chain it with a wide range of other sensors and accessories using a compatible cable ().

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
- 3Vo - 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 BMP183 to your processor
- SDI - this is the Serial Data In / Microcontroller Out Sensor In pin, for data sent from your processor to the BME280

- 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 BME280'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, connect to your microcontrollers I2C clock line.
- SDI - this is also the I2C data pin, connect to your microcontrollers I2C data line.
- STEMMA QT () - These connectors allow you to connectors to dev boards with S TEMMA QT connectors or to other things with various associated accessories ()

Leave the other pins disconnected.

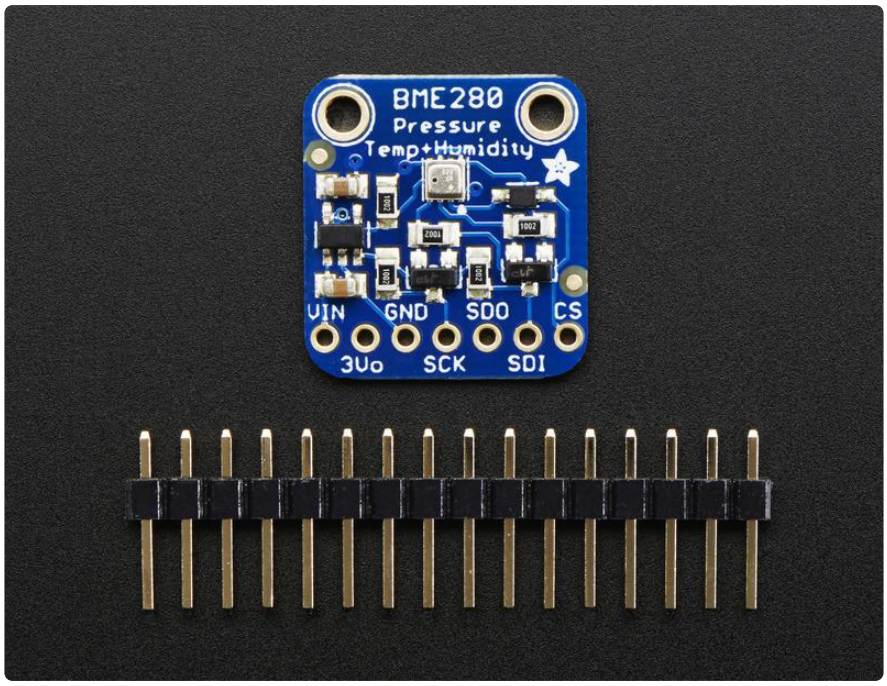
I2C Address

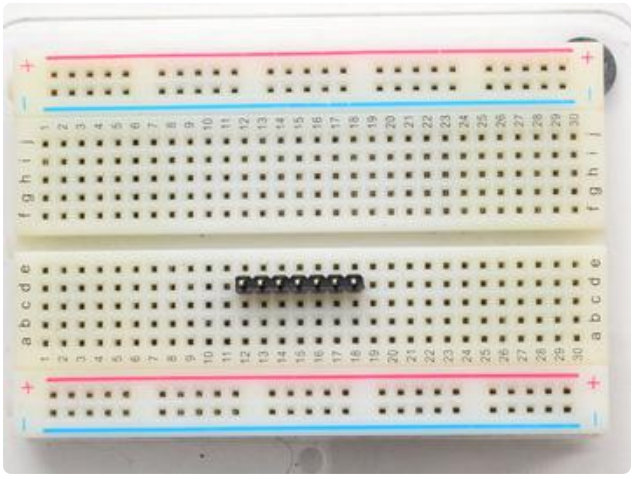
By default, the I2C address is 0x77. If you add a jumper from SDO to GND with a wire, or by soldering the ADDR jumper on the back closed, the address will change to 0x76. This non-default address will need to be specified in user code. See library docs for specifics.

Don't forget to update your user code to also specify the non-default I2C address if it is used.

Assembly

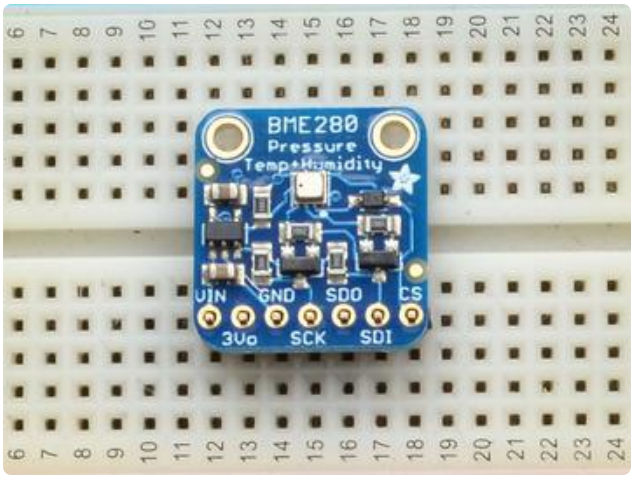
Your board may look a little different - the assembly process is the same!





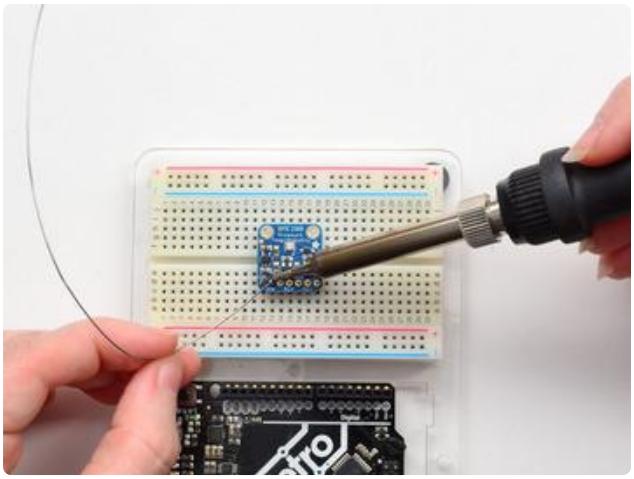
Prepare the header strip:

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



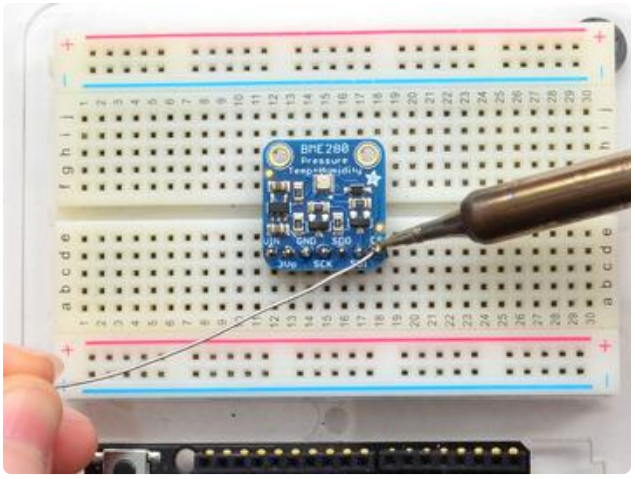
Add the breakout board:

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 ()).

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

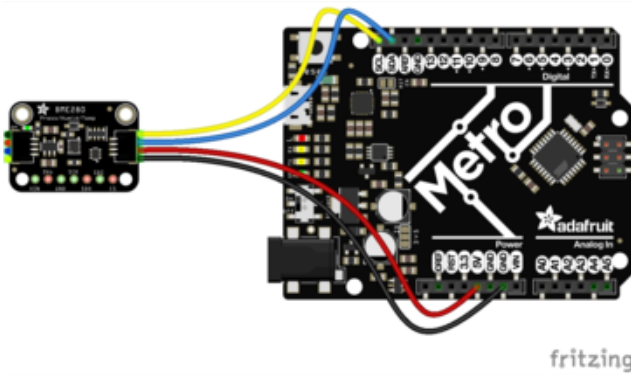
---

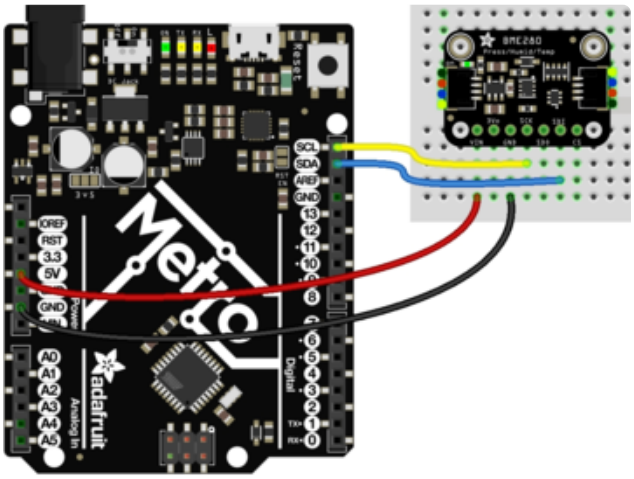
Arduino Test

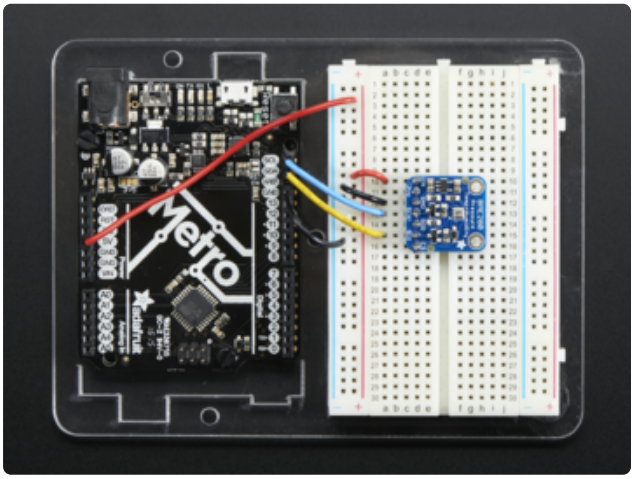
You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, as long as you have 4 available pins it is possible to 'bit-bang SPI' or you can use two I2C pins, but usually those pins are fixed in hardware. Just check out the library, then port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

I2C Wiring

Use this wiring if you want to connect via I2C interface





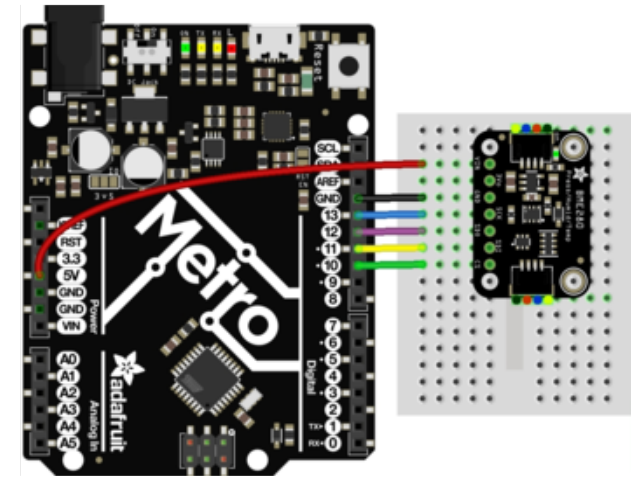


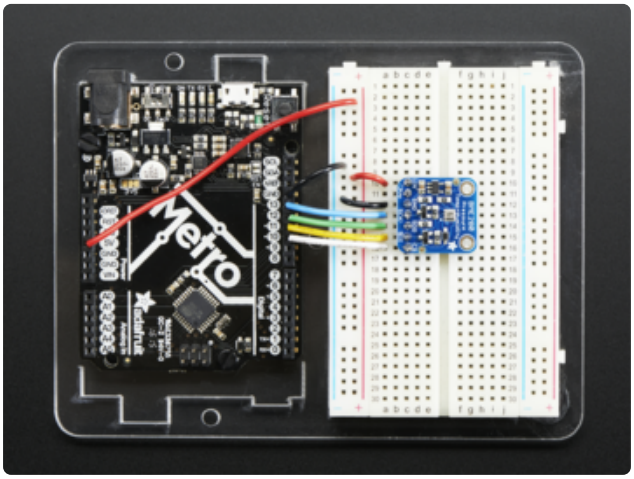
Connect Vin to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
Connect GND to common power/data ground
Connect the SCK pin to the I2C clock SCL pin on your Arduino. On an UNO & '328 based Arduino, this is also known as A5, on a Mega it is also known as digital 21 and on a Leonardo/Micro, digital 3
Connect the SDI 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
By default, the i2c address is 0x77. If you add a jumper from SDO to GND, the address will change to 0x76.

In you are having intermittent issues with I2C on the original non-STEMMA version of the board, try also jumpering CS to Vin.

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:



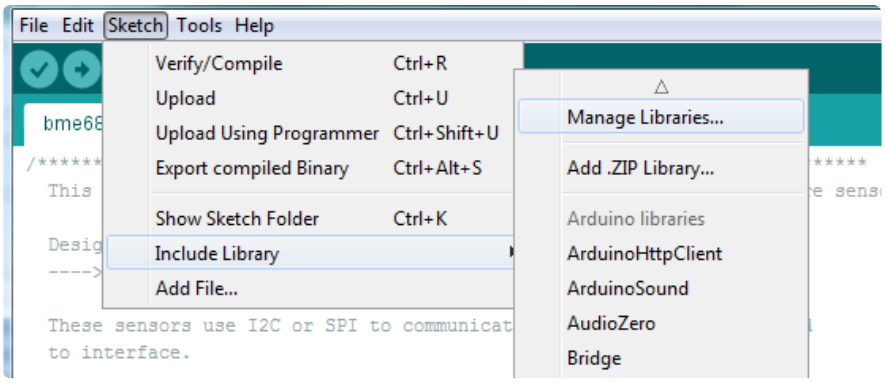


Connect Vin to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
Connect GND to common power/data ground
Connect the SCK pin to Digital #13 but any pin can be used later
Connect the SDO pin to Digital #12 but any pin 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

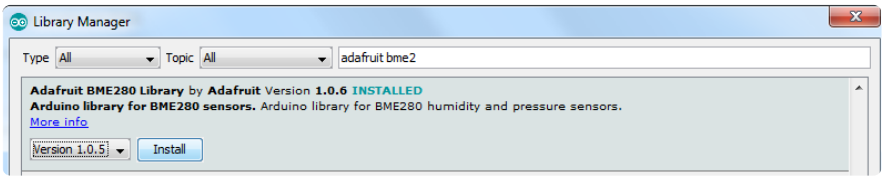
Install Adafruit_BME280 library

To begin reading sensor data, you will need to install the Adafruit_BME280 library (code on our github repository) (). It is available from the Arduino library manager so we recommend using that.

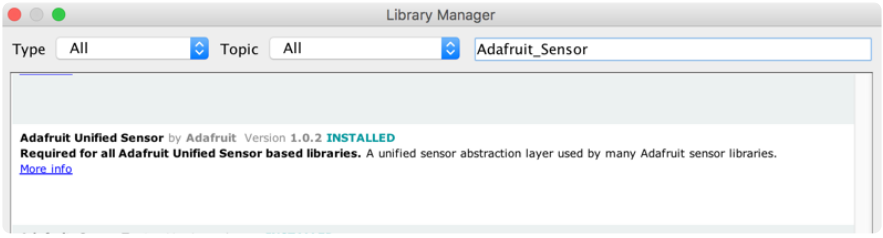
From the IDE open up the library manager...



And type in adafruit bme280 to locate the library. Click Install



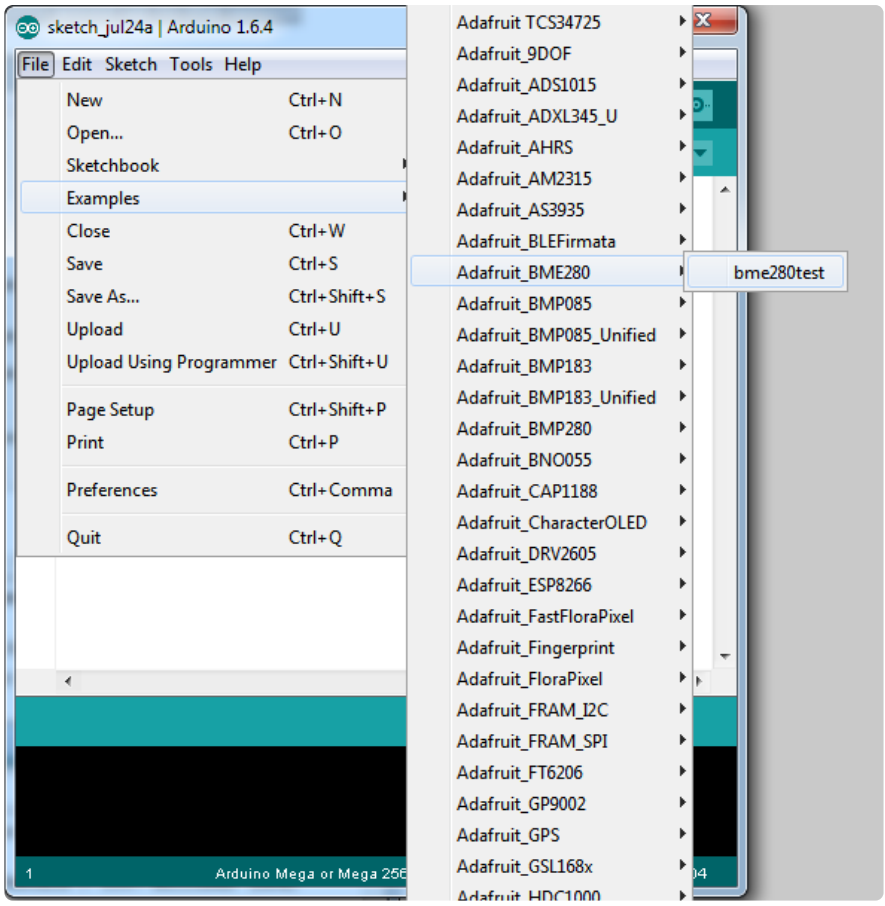
Also add the Adafruit Unified Sensor library



We also have a great tutorial on Arduino library installation at: http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use ()

Load Demo

Open up File->Examples->Adafruit_BME280->bmp280test and upload to your Arduino wired up to the sensor

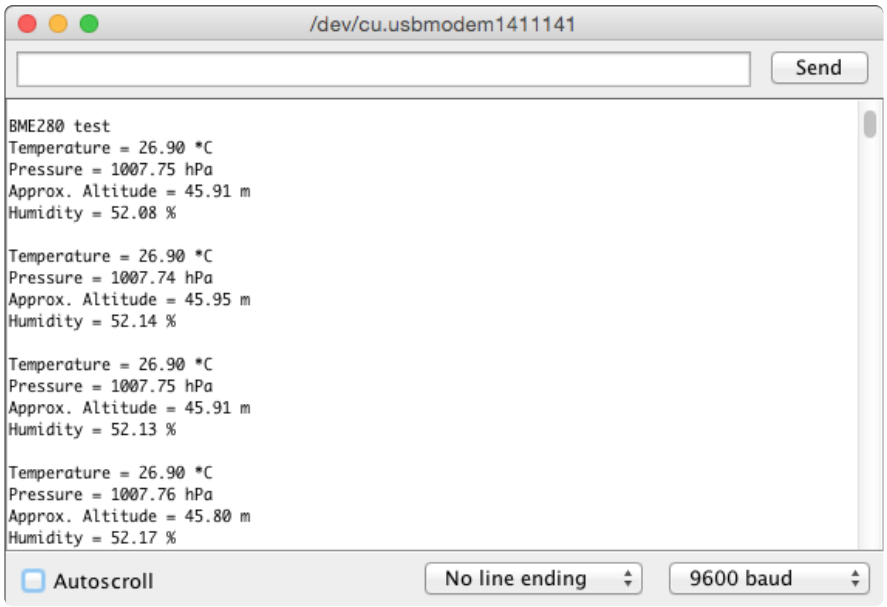


Depending on whether you are using I2C or SPI, change the pin names and comment or uncomment the following lines.

```
#define BME_SCK 13
#define BME_MISO 12
#define BME_MOSI 11
#define BME_CS 10

Adafruit_BME280 bme; // I2C
//Adafruit_BME280 bme(BME_CS); // hardware SPI
//Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO,  BME_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 * 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 3386.39 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 BME280 object with either software SPI (where all four pins can be any I/O) using

```
Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO,  BME_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. ()
In this case, you can use any CS pin, but the other three pins are fixed

```
Adafruit_BME280 bme(BME_CS); // hardware SPI
```

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

```
Adafruit_BME280 bme; // I2C
```

Once started, you can initialize the sensor with

```
  if (!bme.begin()) {
    Serial.println("Could not find a valid BME280 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 humidity, temperature and pressure is easy, just call:

```
bme.readTemperature()
bme.readPressure()
bme.readHumidity()
```

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. Humidity is in % Relative Humidity

It's also possible to turn the BME280 into an altimeter. If you know the pressure at sea level, the library can calculate the current barometric pressure into altitude

```
bmp.readAltitude(seaLevelPressure)
```

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.

Pass in the current sea level pressure in hPa - so the value will be somewhere around ~1000. You can also test with the generic 1013.25 value.

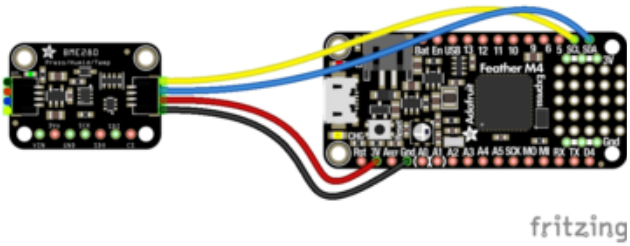
Python & CircuitPython Test

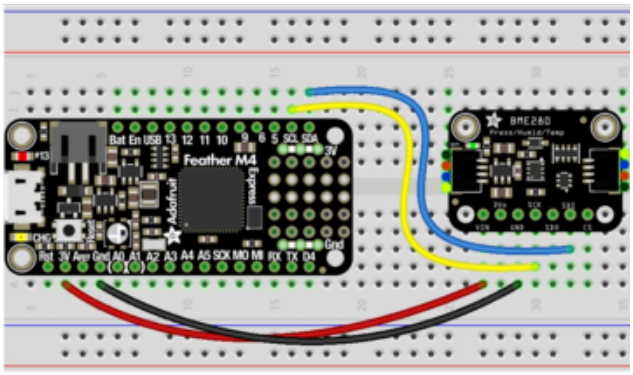
It's easy to use the BME280 sensor with Python or CircuitPython and the Adafruit CircuitPython BME280 () module. This module allows you to easily write Python code that reads the humidity, temperature, pressure, and more from the sensor.

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 ().

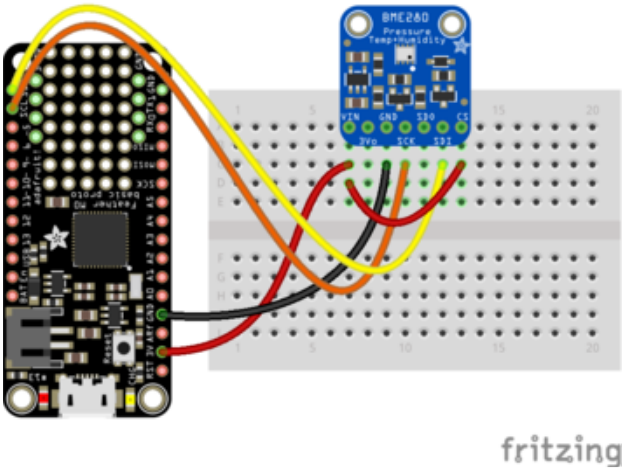
CircuitPython Microcontroller Wiring

First wire up a BME280 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 M0 to the sensor with I2C:

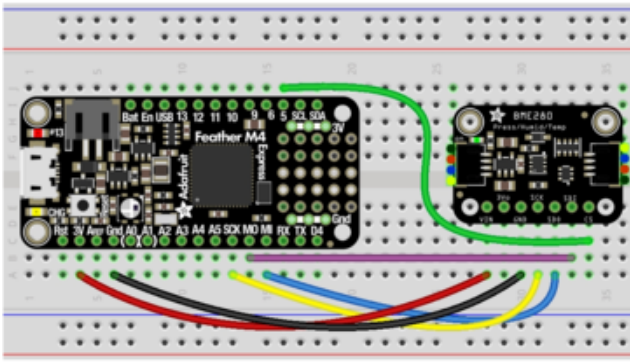


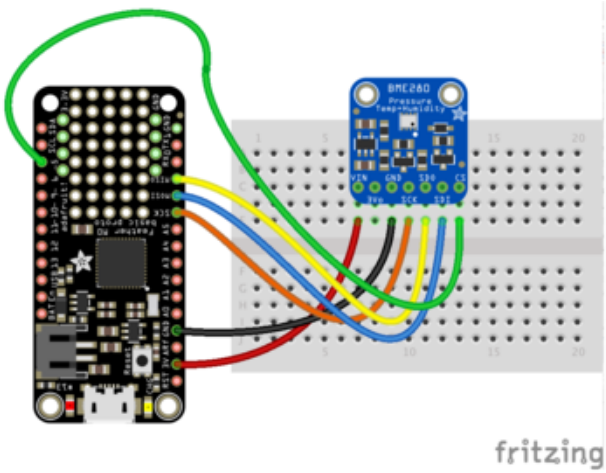


Board 3V to sensor VIN
Board GND to sensor GND
Board SCL to sensor SCK
Board SDA to sensor SDI



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



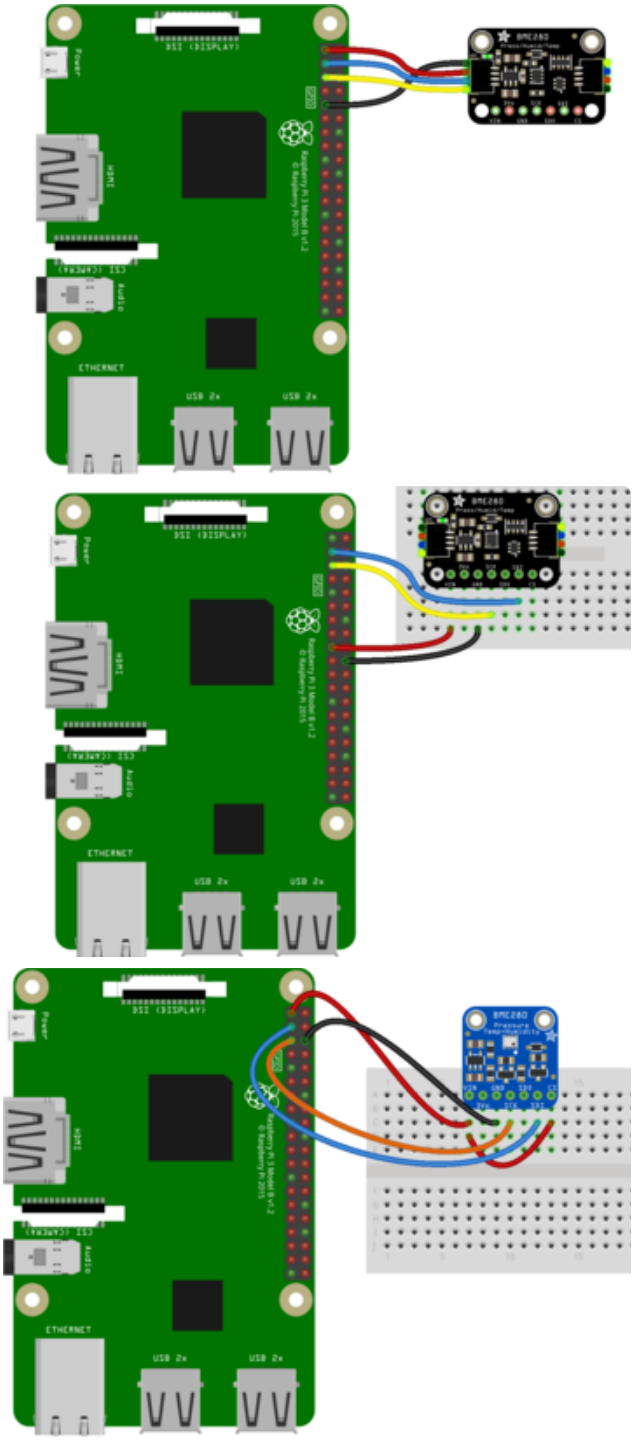


Board 3V to sensor VIN
Board GND to sensor GND
Board SCK to sensor SCK
Board MOSI to sensor SDI
Board MISO to sensor SDO
Board D5 to sensor CS (or use any other free digital I/O pin)

Python Computer Wiring

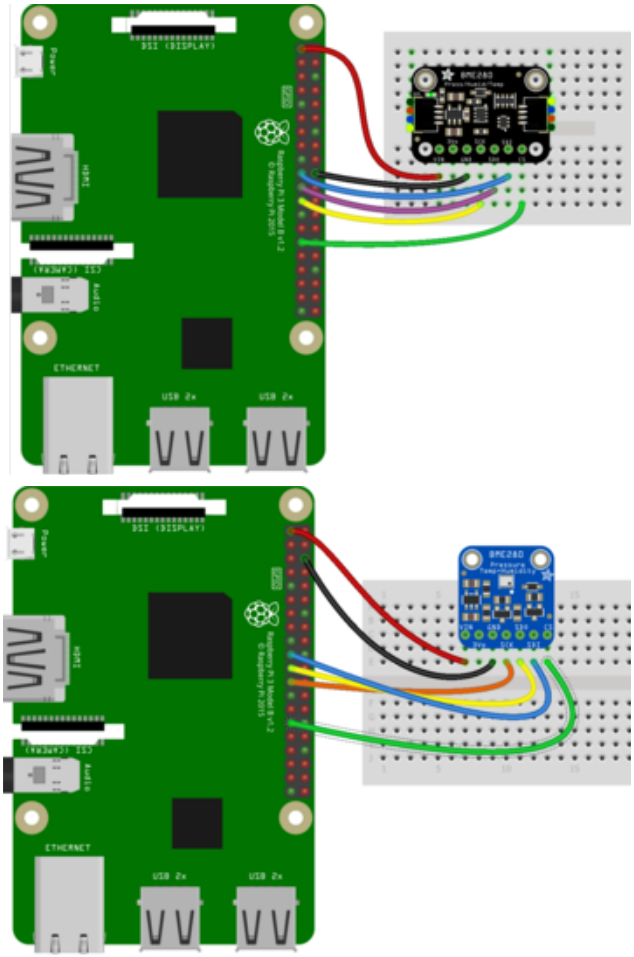
Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported ().

Here's the Raspberry Pi wired with I2C:



Pi 3V3 to sensor VIN
Pi GND to sensor GND
Pi SCL to sensor SCK
Pi SDA to sensor SDI

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 BME280 Library

You'll need to install the Adafruit CircuitPython BME280 () library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython () 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 (). Our CircuitPython starter guide has a great page on how to install the library bundle ().

Before continuing make sure your board's lib folder or root filesystem has the adafruit_bme280, and adafruit_bus_device folders copied over. Both of these are folders with library files in them. Make sure to copy the whole folder for both into your lib folder!

Next connect to the board's serial REPL () so you are at the CircuitPython >>> prompt.

Python Installation of BME280 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 ()!

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

- `sudo pip3 install adafruit-circuitpython-bme280`

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
from adafruit_bme280 import basic as adafruit_bme280
i2c = board.I2C()  # uses board.SCL and board.SDA
bme280 = adafruit_bme280.Adafruit_BME280_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
from adafruit_bme280 import basic as adafruit_bme280
spi = board.SPI()
cs = digitalio.DigitalInOut(board.D5)
bme280 = adafruit_bme280.Adafruit_BME280_SPI(spi, cs)
```

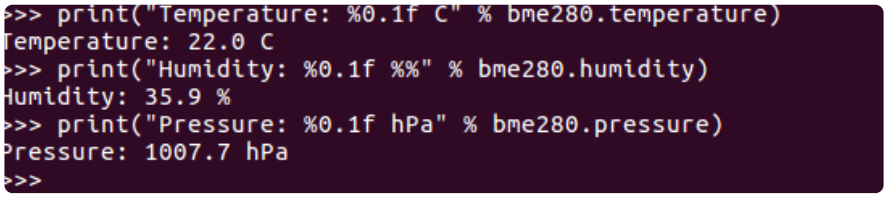
Now you're ready to read values from the sensor using any of these properties:

- temperature - The sensor temperature in degrees Celsius.
- humidity - The percent humidity as a value from 0 to 100%.

- pressure - The pressure in hPa.
- altitude - The altitude in meters.

For example to print temperature, humidity, and pressure:

```
print("\nTemperature: %0.1f C" % bme280.temperature)
print("Humidity: %0.1f %%" % bme280.humidity)
print("Pressure: %0.1f hPa" % bme280.pressure)
```

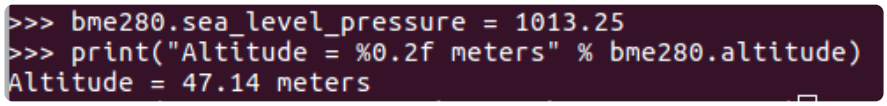


For altitude you'll want to set the pressure at sea level for your location to get the most accurate measure (remember these sensors can only infer altitude based on pressure and need a set calibration point). Look at your local weather report for a pressure at sea level reading and set the sea_level_pressure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

```
bme280.sea_level_pressure = 1013.4
```

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

```
print("Altitude = %0.2f meters" % bme280.altitude)
```



You can use the BME280 temperature and humidity to calculate the dew point using the Magnus formula ()! For this example, you'll need to `import` an additional library: `math`. Run the following code:

```
import math
b = 17.62
c = 243.12
gamma = (b * bme280.temperature /(c + bme280.temperature)) +
math.log(bme280.humidity / 100.0)
dewpoint = (c * gamma) / (b - gamma)
print(dewpoint)
```

That's all there is to using the BME280 sensor with CircuitPython!

Full Example Code

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

import time
import board
from adafruit_bme280 import basic as adafruit_bme280

# Create sensor object, using 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
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# OR create sensor object, using the board's default SPI bus.
# spi = board.SPI()
# bme_cs = digitalio.DigitalInOut(board.D10)
# bme280 = adafruit_bme280.Adafruit_BME280_SPI(spi, bme_cs)

# change this to match the location's pressure (hPa) at sea level
bme280.sea_level_pressure = 1013.25

while True:
    print("\nTemperature: %0.1f C" % bme280.temperature)
    print("Humidity: %0.1f %%" % bme280.relative_humidity)
    print("Pressure: %0.1f hPa" % bme280.pressure)
    print("Altitude = %0.2f meters" % bme280.altitude)
    time.sleep(2)
```

Python Docs

Python Docs ()

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!

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 (), a web platform designed (by Adafruit! ()) 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.

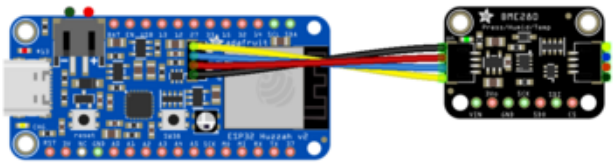
From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

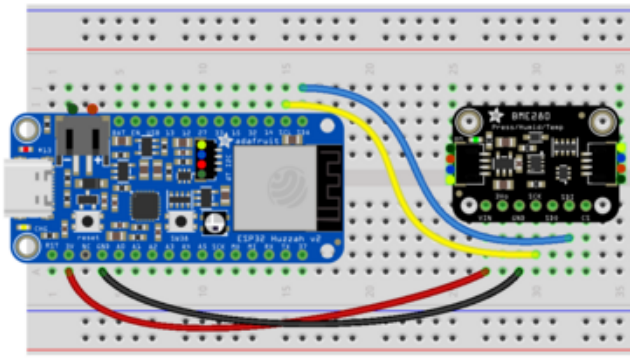
If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

Quickstart: Adafruit IO WipperSnapper

Wiring

First, wire up an BME280 to your board exactly as follows. Here is an example of the BME280 wired to an Adafruit ESP32 Feather V2 () using I2C with a STEMMA QT cable (no soldering required) ()



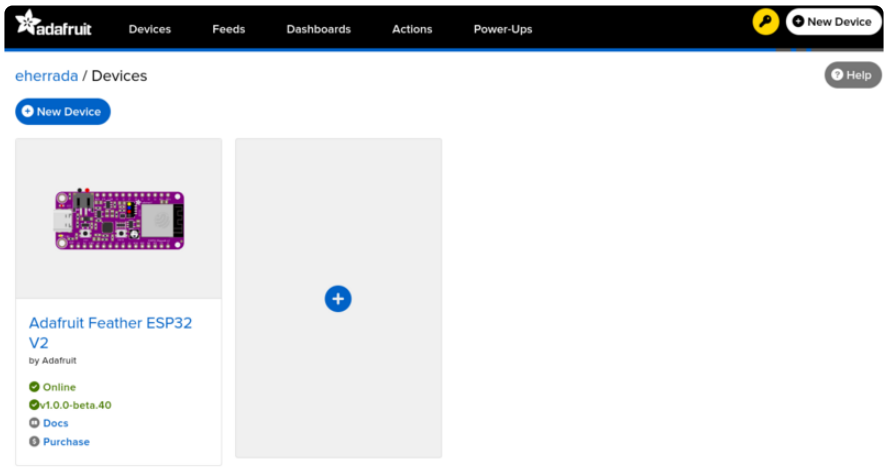


Board 3V to sensor VIN (red wire on STEMMA QT)
Board GND to sensor GND (black wire on STEMMA QT)
Board SCL to sensor SCL (yellow wire on STEMMA QT)
Board SDA to sensor SDA (blue wire on STEMMA QT)

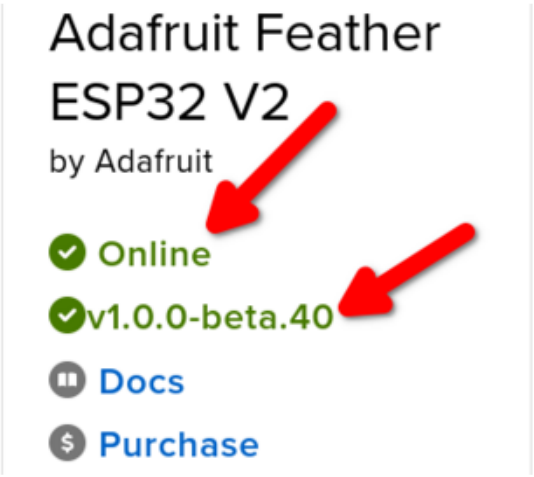
Usage

Connect your board to Adafruit IO Wippersnapper and navigate to the WipperSnapper board list ().

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



If you do not see your board listed here - you need to connect your board to Adafruit IO () first.

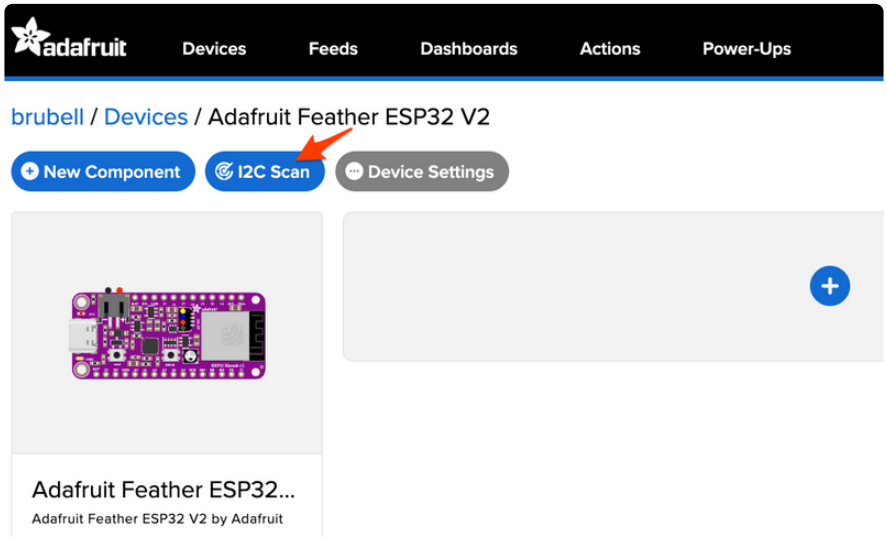


On the device page, quickly check that you're running the latest version of the WipperSnapper firmware.

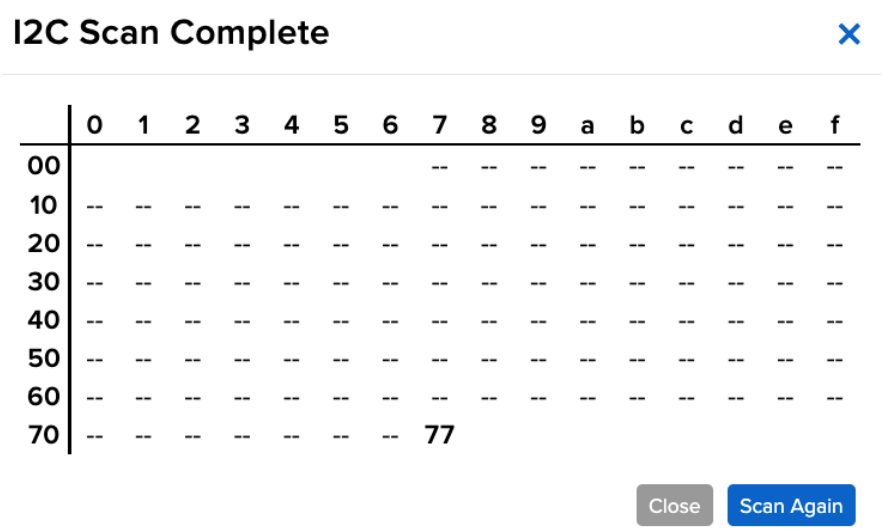
The device tile on the left indicates the version number of the firmware running on the connected board.

If the firmware version is green with a checkmark - continue with this guide.
If the firmware version is red with an "X" - update to the latest WipperSnapper firmware () on your board before continuing.

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



You should see the BME280's default I2C address of `0x77` pop up in the I2C scan list.



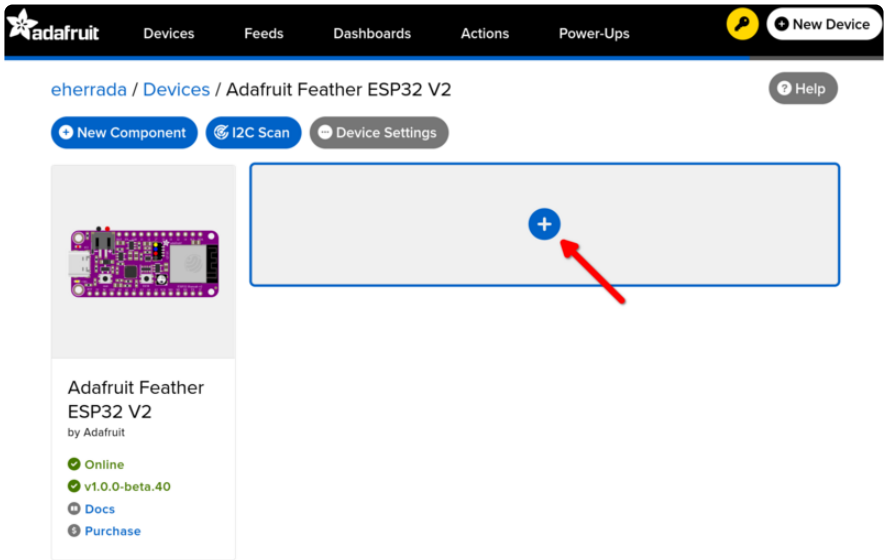
I don't see the sensor's I2C address listed!

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

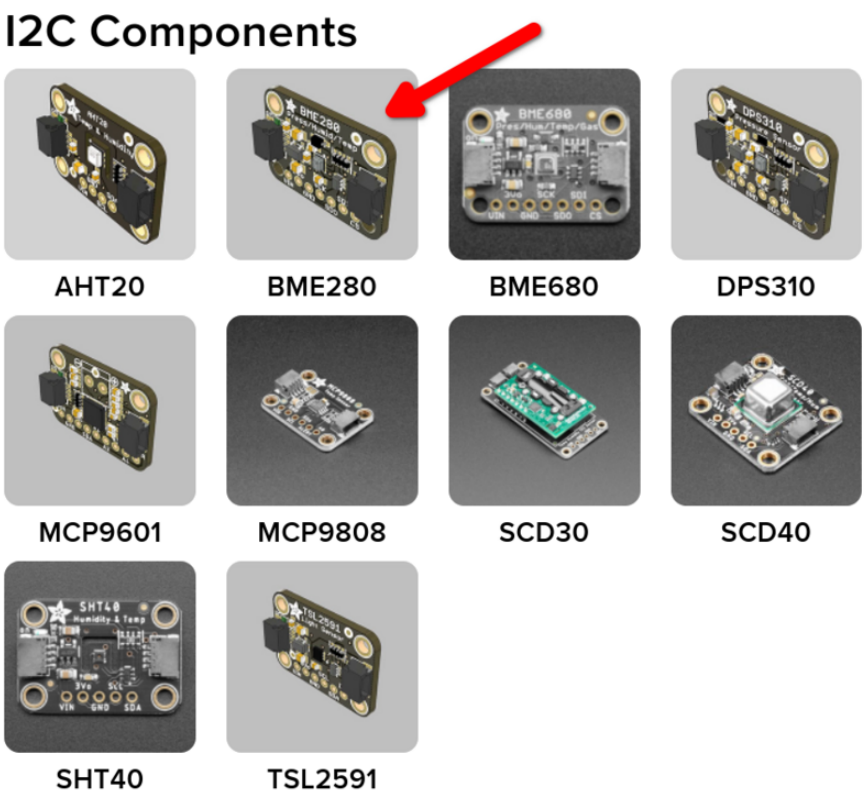
Then, reset the board and let it re-connect to Adafruit IO WipperSnapper.

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.



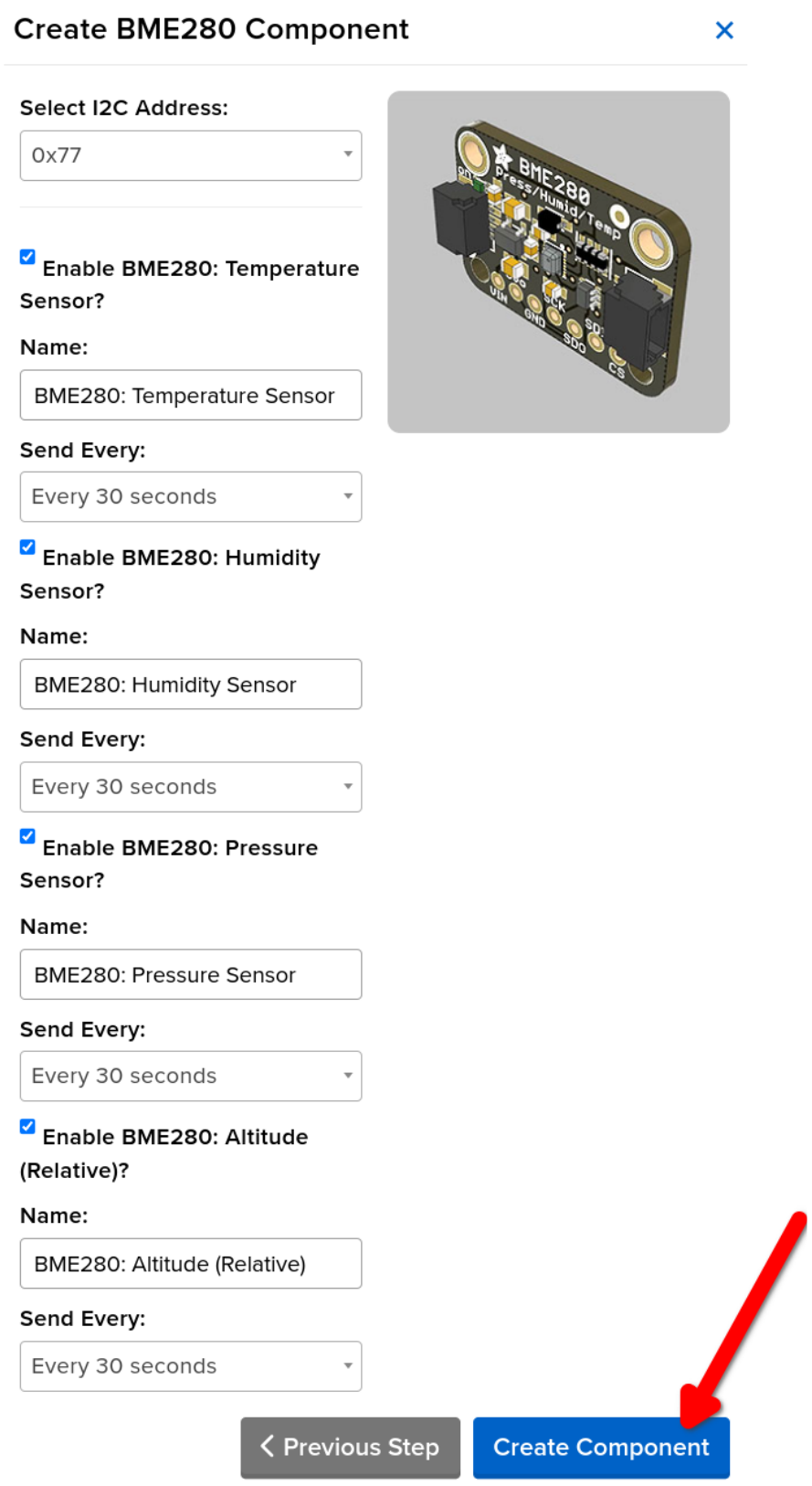
Select the BME280 from the component picker.



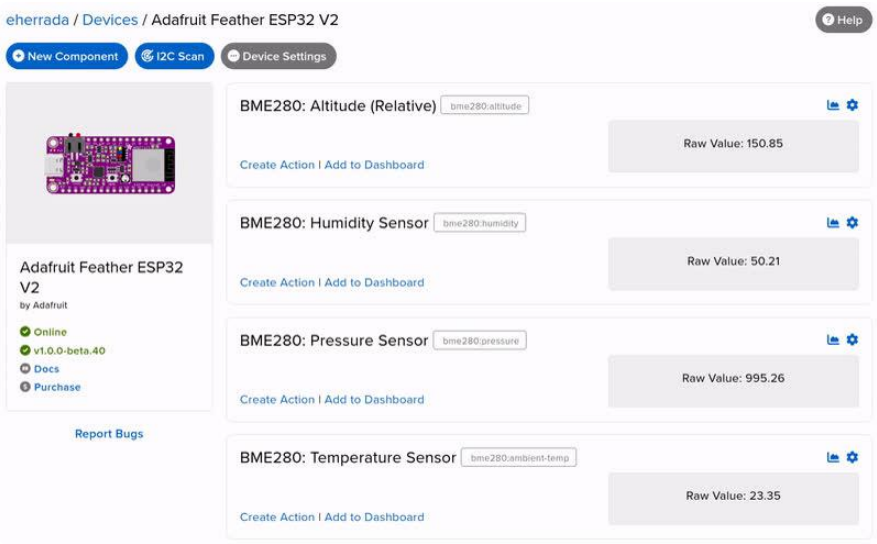
On the component configuration page, the BME280's sensor address should be listed along with the sensor's settings.

The Send Every option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the BME280 sensor 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 for each sensor to every 30 seconds.

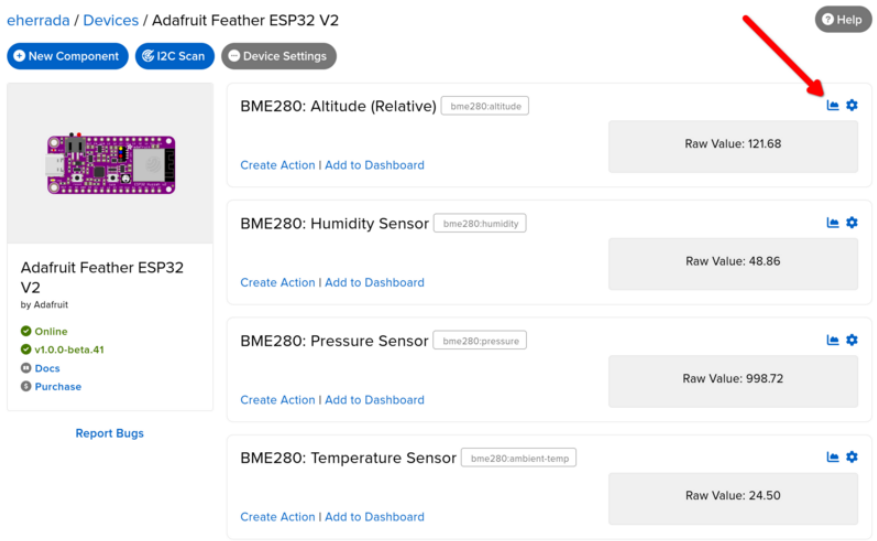


Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.



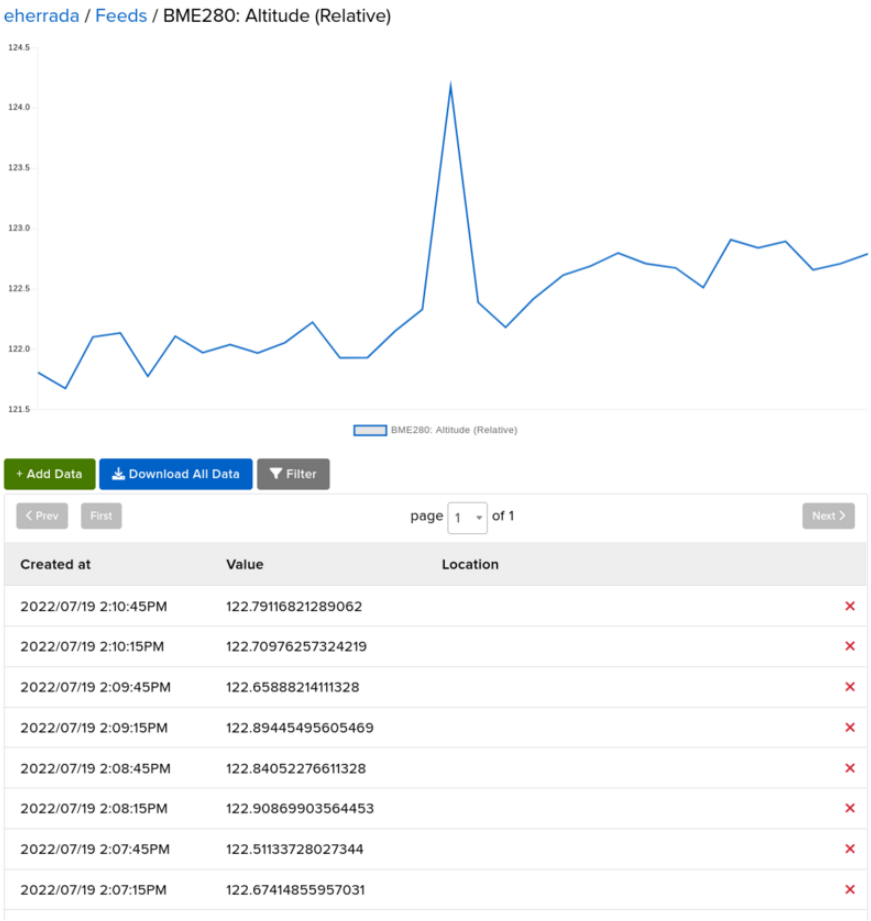
To view the data that has been logged from the sensor, click on the graph next to the sensor name.

The BME280 has four sensors that each have their own feeds. In this picture, we're looking at the altitude sensor, but if you click on the graph icon for the different sensors you'll see their feed history.



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 ().

For IO Free accounts, feed data is stored for a maximum of 30 days and there's a maximum of 10 feeds. In this guide, you created four feeds (one for each of the BME280's sensors). If you'd like to store data for more than 30 days, increase the number of feeds (components) you can use with WipperSnapper, or increase your data rate to send more sensor measurements to Adafruit IO - upgrade your account to Adafruit IO Plus ().



Downloads

Documents

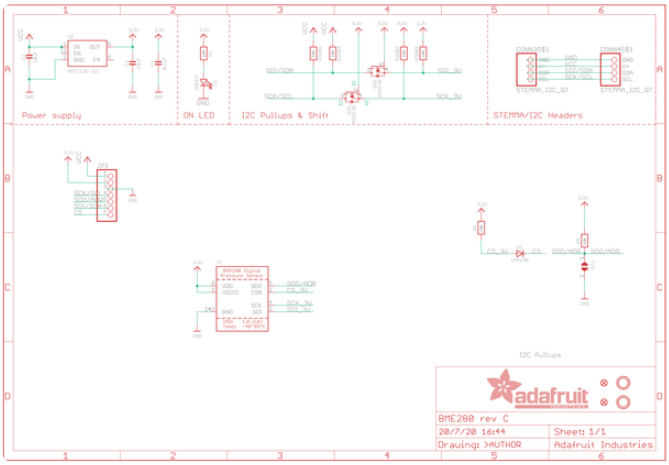
- Datasheet for the BME280 sensor used in this breakout ()
- Arduino BME280 Driver ()
- Fritzing object in the Adafruit Fritzing Library ()
- EagleCAD PCB files on GitHub (original, non QT version) ()
- EagleCAD PCB files for QT version, which is the same exact PCB as the BMP280 ()(sensors are pin/size compatible)
- K&R Smith calibration notes ()

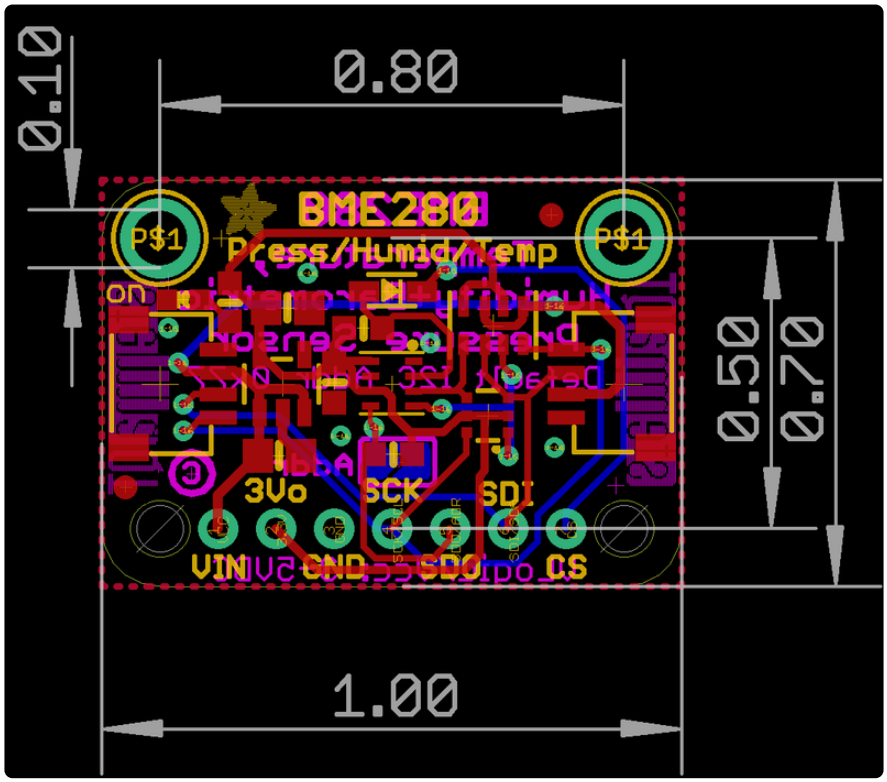
Alternative Driver (Python)

If you are using this breakout with a Raspberry Pi or Pi2, you can also look at the Adafruit_Python_BME280 () driver.

This alternative driver uses I2C to communicate with the BME280, so connect SCL on the Pi to SCK on the BME, and SDA to SDI, along with power (3.3V to VIN) and GND.

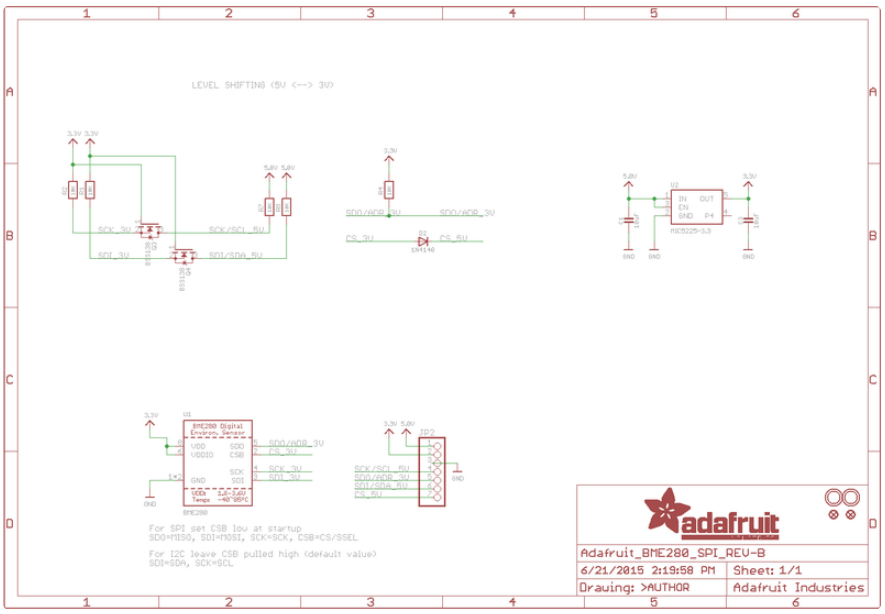
Schematic and Fab Print for STEMMA QT Version

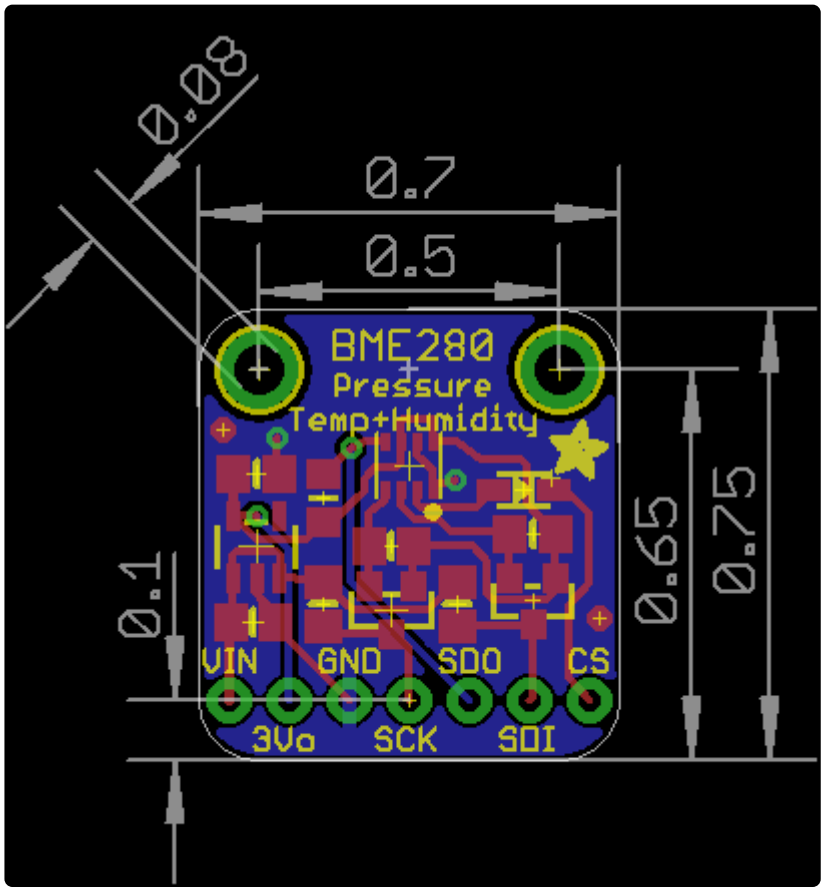




Schematic and Fab Print for Original Version

Click to enlarge


LEVEL SHIFTING (5V <--> 3V)
adafruit
Adafruit_BME280_SPI_REV-B
6/21/2015 2:19:58 PM
Sheet: 1/1
Drawing: >AUTHOR
Adafruit Industries


0.08
0.7
0.5
BME280
Pressure
Temp+Humidity
0.1
0.65
0.75
VIN
GND
SDO
CS
3Vo
SCK
SDI