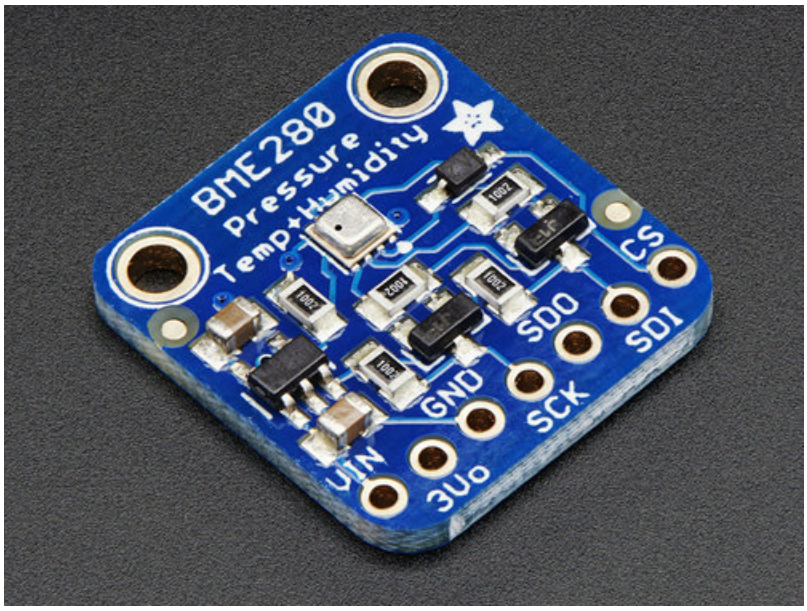
adafruit learning system

Adafruit BME280 Humidity + Barometric Pressure + Temperature Sensor Breakout

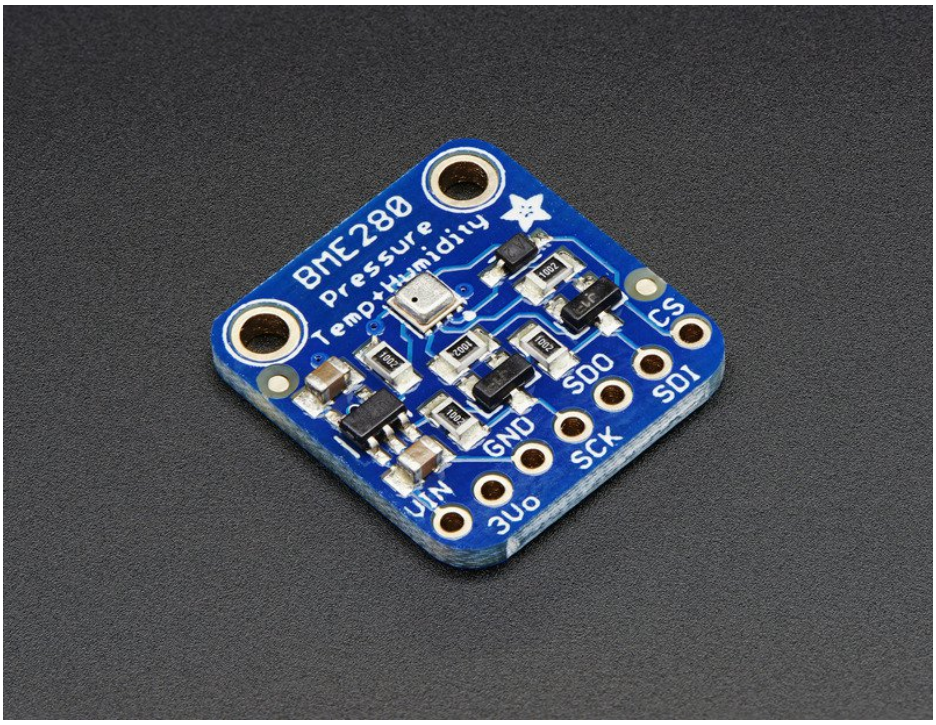
Created by lady ada



Last updated on 2018-09-25 04:11:33 PM UTC

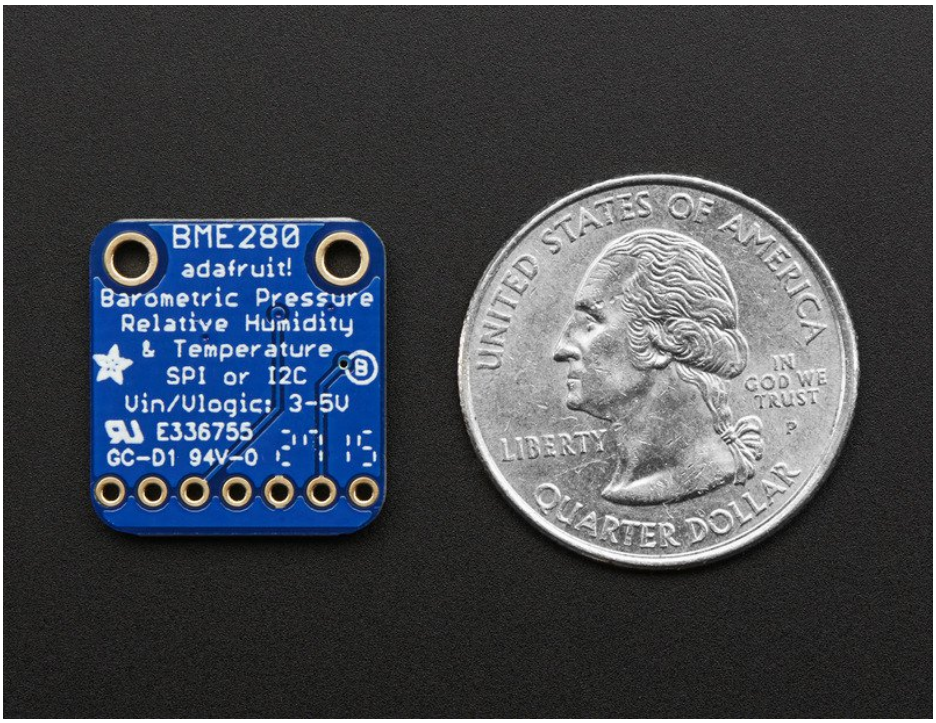
Guide Contents

Overview

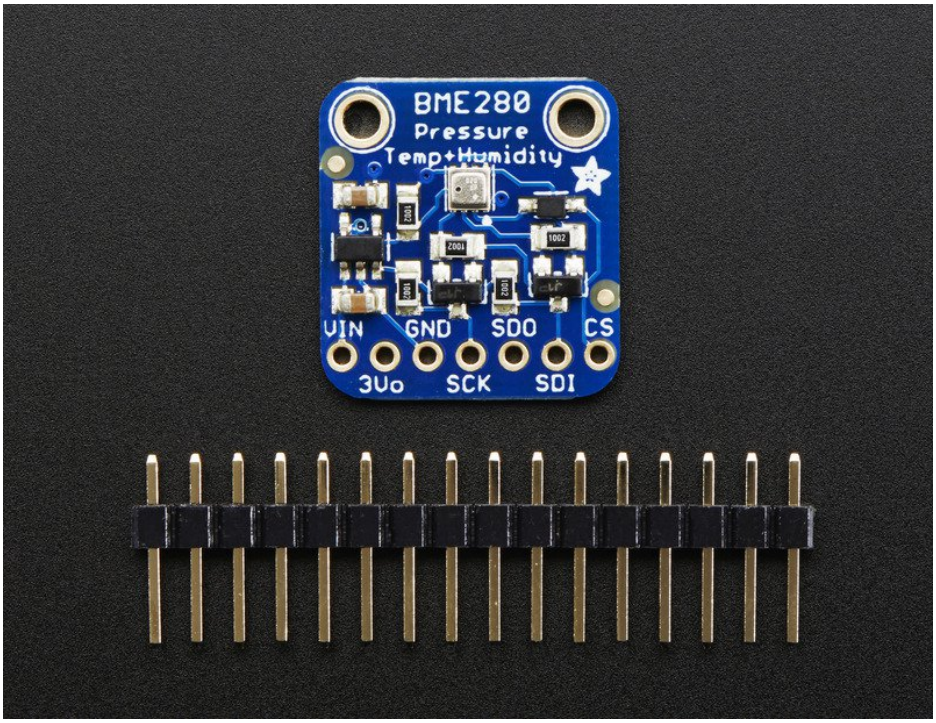


Bosch has stepped up their game with their new BME280 sensor, an environmental sensor with temperature, barometric pressure *and* humidity! This sensor is great for all sorts of weather/environmental sensing and can even be used in both I2C and SPI!

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!

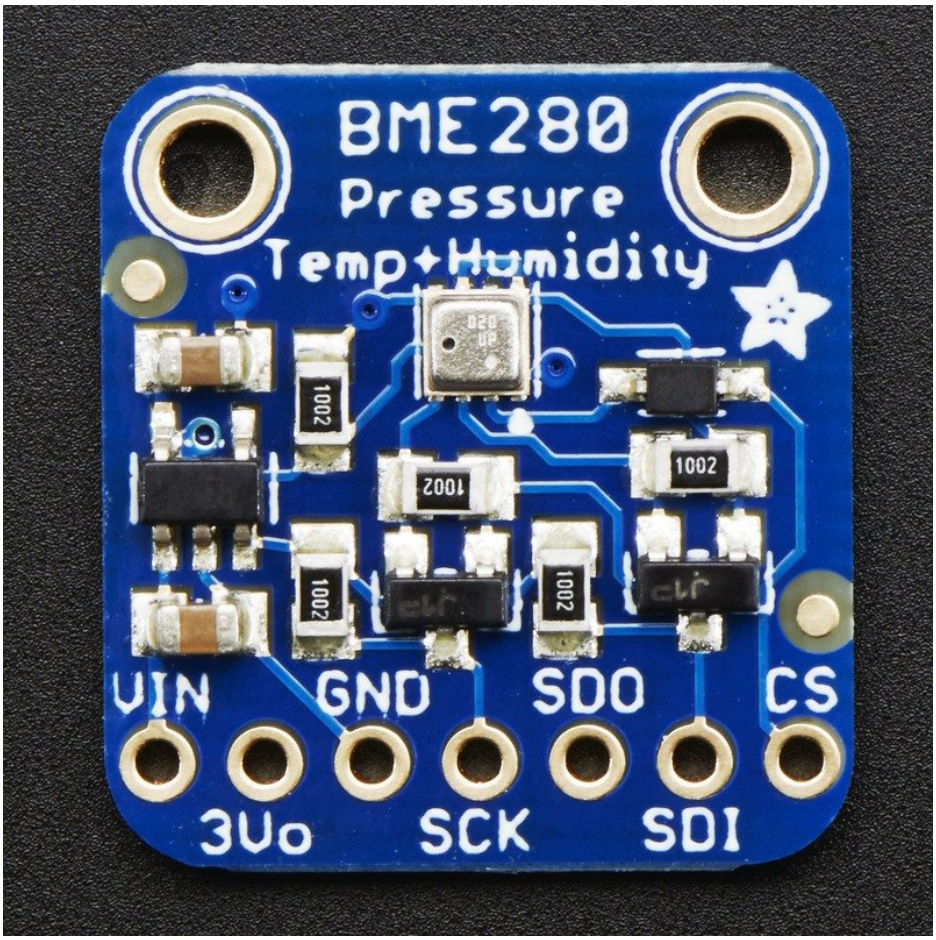


The BME280 is the next-generation of sensors from Bosch, and is the upgrade to the BMP085/BMP180/BMP183 - with a low altitude noise of 0.25m and the same fast conversion time. It has the same specifications, but can use either I2C *or* SPI. For simple easy wiring, go with I2C. If you want to connect 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. We even wrote up a nice tutorial with wiring diagrams, schematics, libraries and examples to get you running in 10 minutes!

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 **S**PI **C**lo**c**k pin, its an input to the chip
- **SDO** - this is the **S**erial **D**ata **O**ut / **M**aster **I**n **S**lave **O**ut pin, for data sent from the BMP183 to your processor
- **SDI** - this is the **S**erial **D**ata **I**n / **M**aster **O**ut **S**lave **I**n pin, for data sent from your processor to the BME280
- **CS** - this is the **C**hip **S**elect pin, drop it low to start an SPI transaction. Its an input to the chip

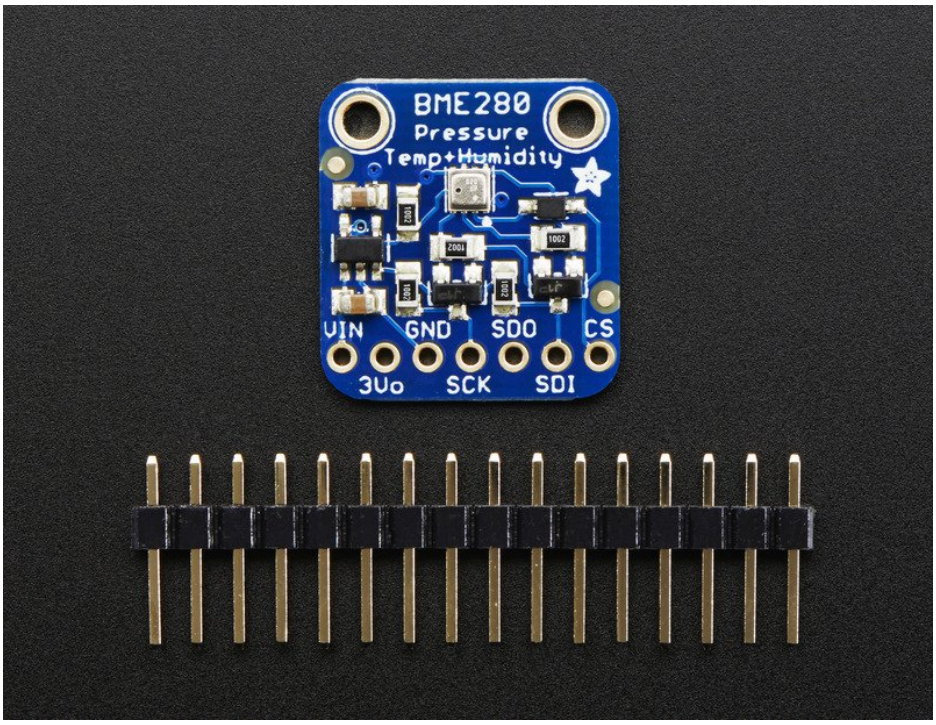
If you want to connect multiple BME280's to one microcontroller, have them share the SDI, SDO and SCK pins. Then assign each one a unique CS pin.

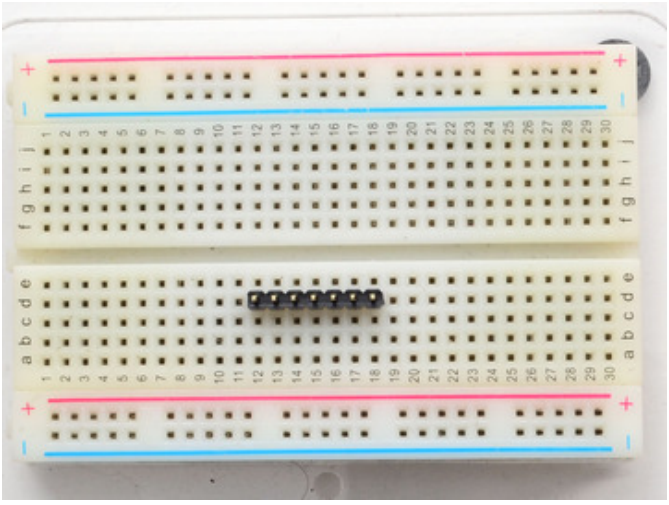
I2C Logic pins:

- **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.

Leave the other pins disconnected

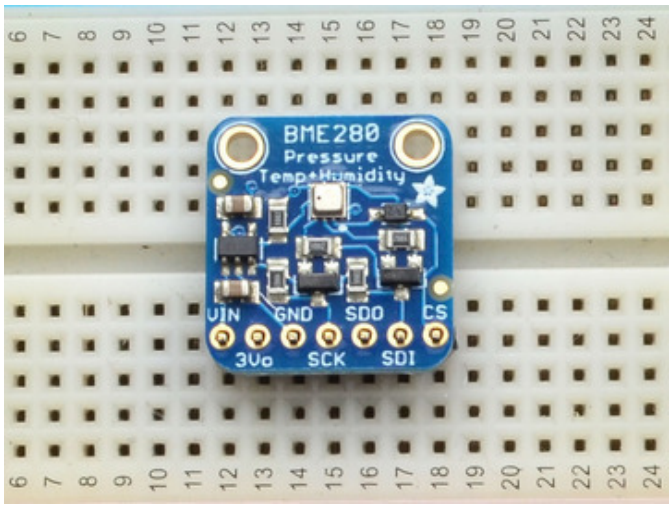
Assembly





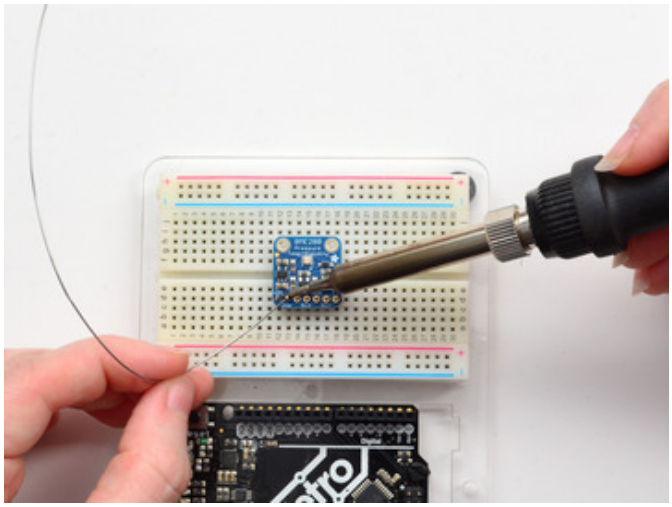
Prepare the header strip:

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:

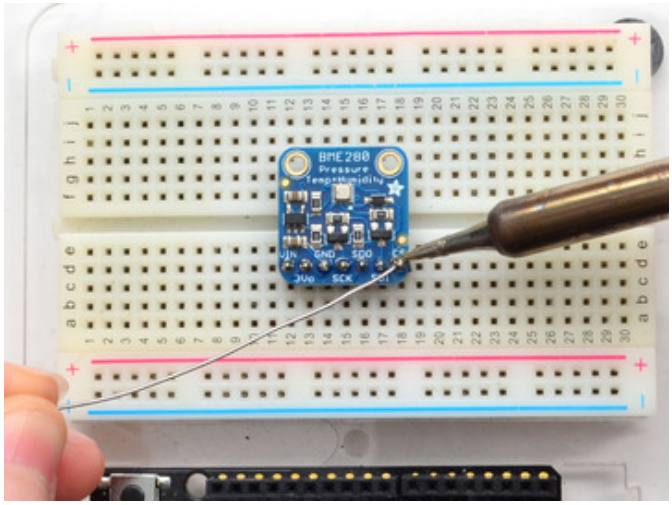
Place the breakout board over the pins so that the short pins poke through the breakout pads



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).



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

Arduino Test

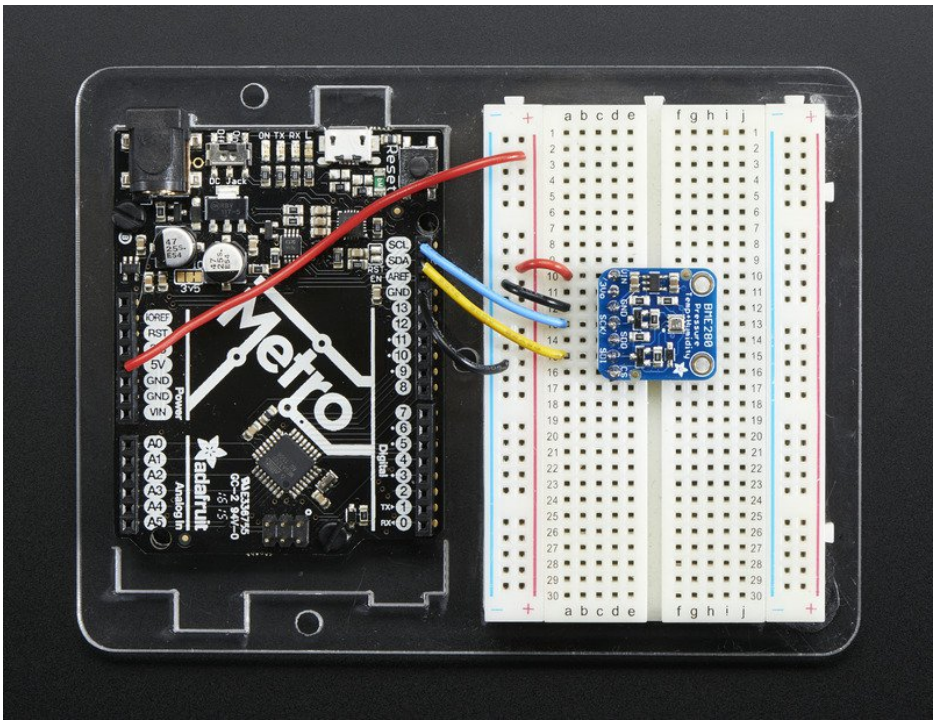
You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, as long as you have 4 available pins it is possible to 'bit-bang SPI' or you can use two I2C pins, but usually those pins are fixed in hardware. Just check out the library, then port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



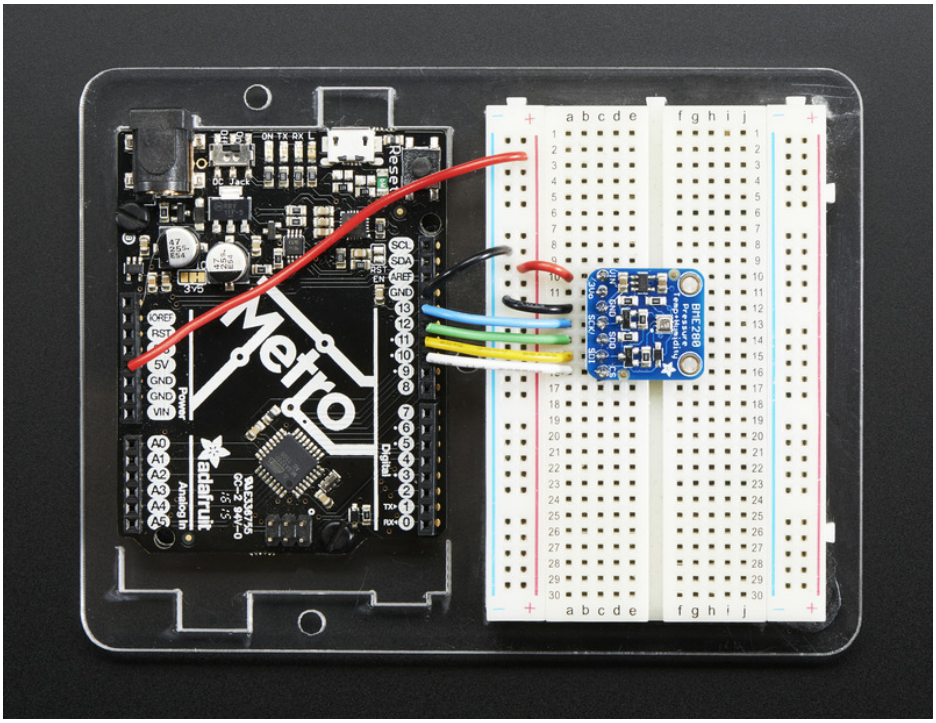
SPI Wiring

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

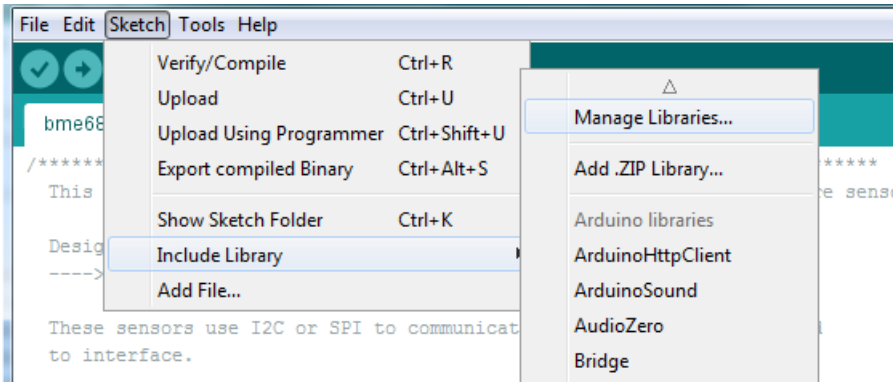
Later on, once we get it working, we can adjust the library to use hardware SPI if you desire, or change the pins to other



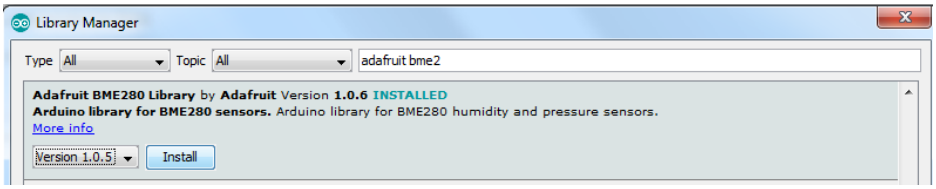
Install Adafruit_BME280 library

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

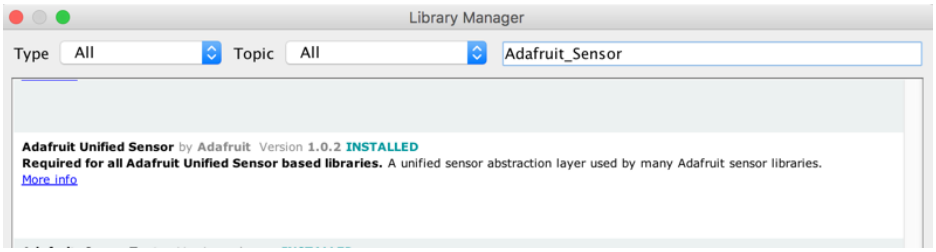
From the IDE open up the library manager...



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



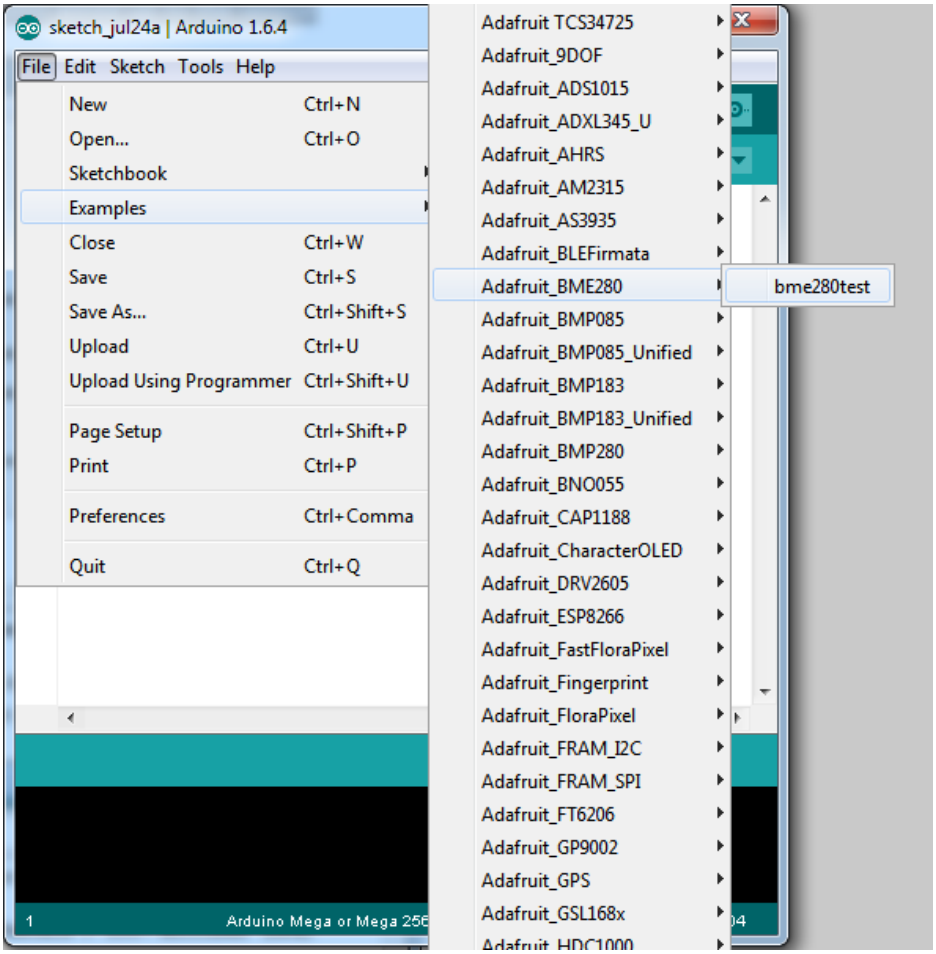
Also add the **Adafruit Unified Sensor** library



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

Load Demo

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



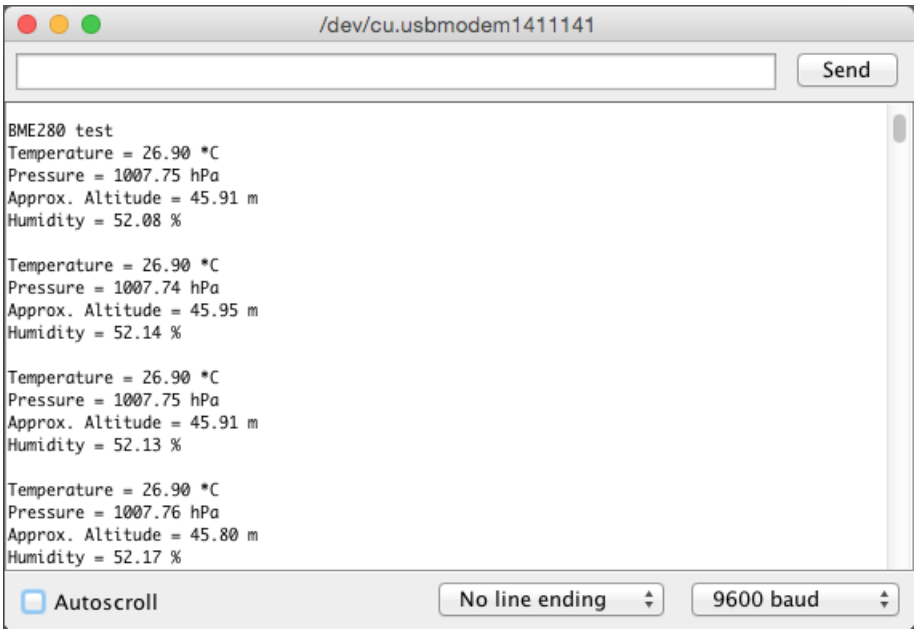
Depending on whether you are using I2C or SPI, change the pin names and comment or uncomment the following

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. (https://adafru.it/d5h)
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.

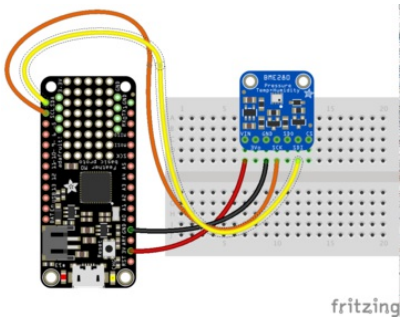
Python & CircuitPython Test

It's easy to use the BME280 sensor with Python or CircuitPython and the Adafruit CircuitPython BME280 (https://adafru.it/BfX) 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 (https://adafru.it/BSN).

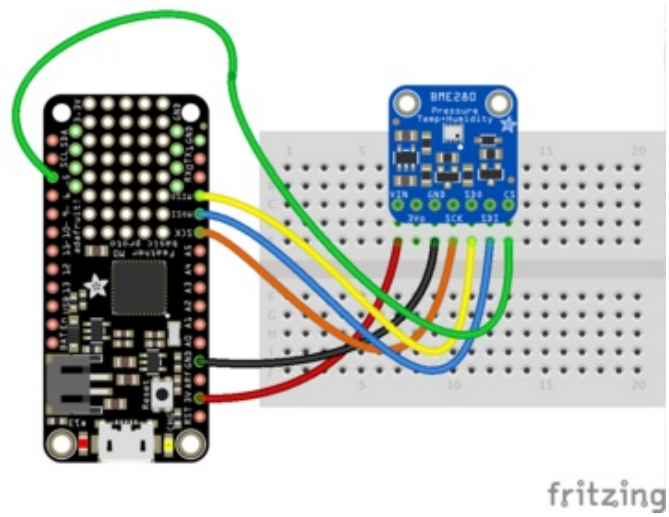
CircuitPython Microcontroller Wiring

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:

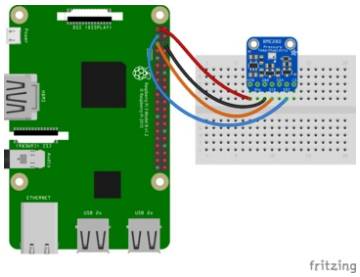


- **Board 3V** to **sensor VIN**
- **Board GND** to **sensor GND**
- **Board SCK** to **sensor SCK**
- **Board MOSI** to **sensor SDI**
- **Board MISO** to **sensor SDO**
- **Board D5** to **sensor CS** (or use any other free digital I/O pin)

Python Computer Wiring

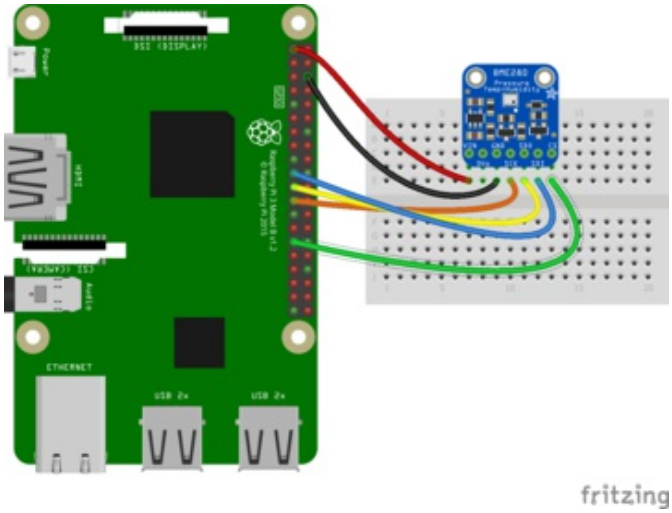
Since there's *dozens* of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

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 (https://adafru.it/BfX) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (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). Our CircuitPython starter guide has a great page on how to install the library bundle (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_bme280.mpy**
- **adafruit_bus_device**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_bme280.mpy,** and **adafruit_bus_device** files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz) 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 (https://adafru.it/BSN)!

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
import busio
import adafruit_bme280
i2c = busio.I2C(board.SCL, 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 busio
import digitalio
import adafruit_bme280
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
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)
```

```
>>> print("Temperature: %0.1f C" % bme280.temperature)
Temperature: 22.0 C
>>> print("Humidity: %0.1f %%" % bme280.humidity)
Humidity: 35.9 %
>>> print("Pressure: %0.1f hPa" % bme280.pressure)
Pressure: 1007.7 hPa
>>>
```

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)
```

```
>>> bme280.sea_level_pressure = 1013.25
>>> print("Altitude = %0.2f meters" % bme280.altitude)
Altitude = 47.14 meters
```

You can use the BME280 temperature and humidity to calculate the dew point using the Magnus formula (https://adafru.it/C7u)! 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

```
import time

import board
import busio
import adafruit_bme280

# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# OR create library object using our Bus SPI port
#spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
#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.humidity)
    print("Pressure: %0.1f hPa" % bme280.pressure)
    print("Altitude = %0.2f meters" % bme280.altitude)
    time.sleep(2)
```

Python Docs

Python Docs (https://adafru.it/BK2)

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 BME280 sensor used in this breakout (https://adafru.it/fG0)
- Arduino BME280 Driver (https://adafru.it/fFZ)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/aP3)
- EagleCAD PCB files on GitHub (https://adafru.it/rCW)
- K&R Smith calibration notes (https://adafru.it/BfU)

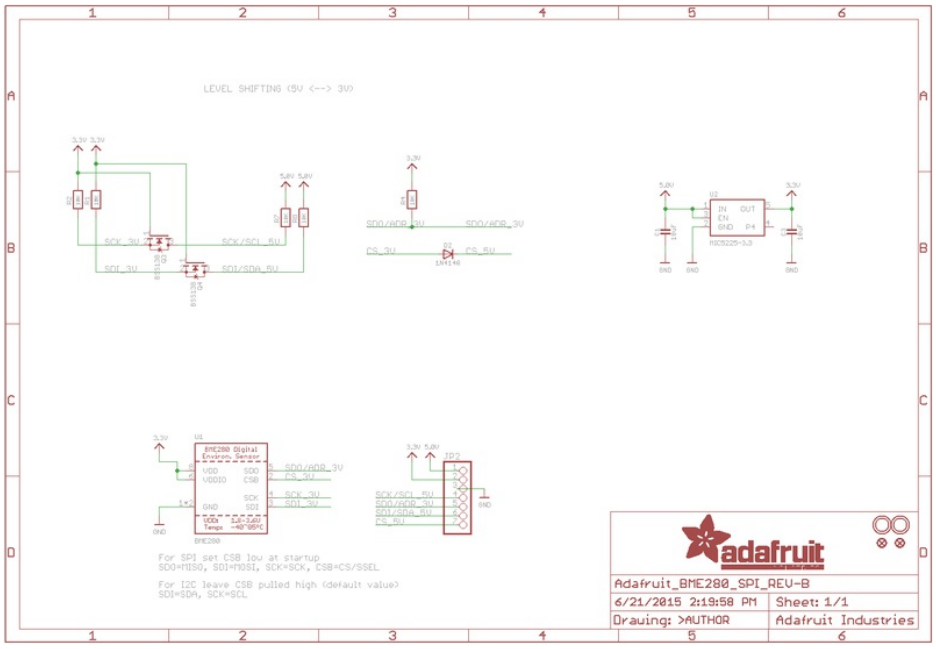
Alternative Driver (Python)

If you are using this breakout with a Raspberry Pi or Pi2, you can also look at the Adafruit_Python_BME280 (https://adafru.it/fX4) 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.

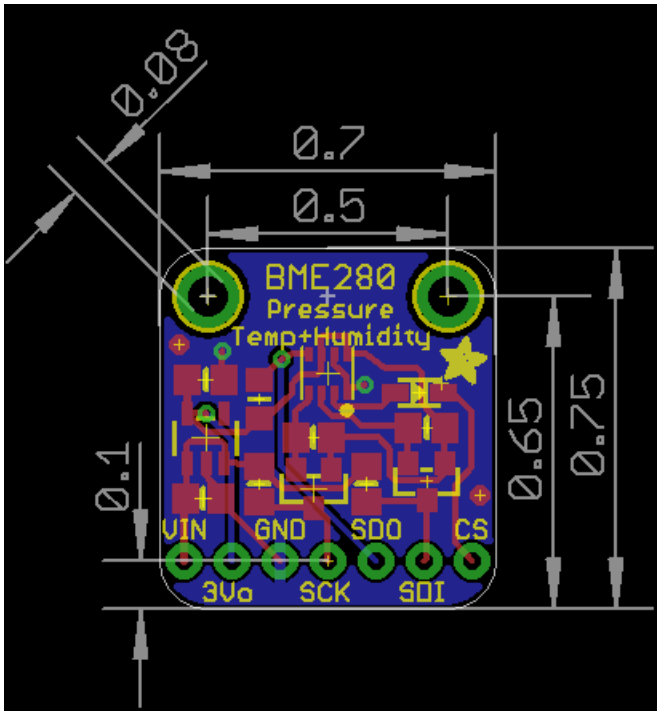
Schematic

Click to enlarge



Dimensions

In inches


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