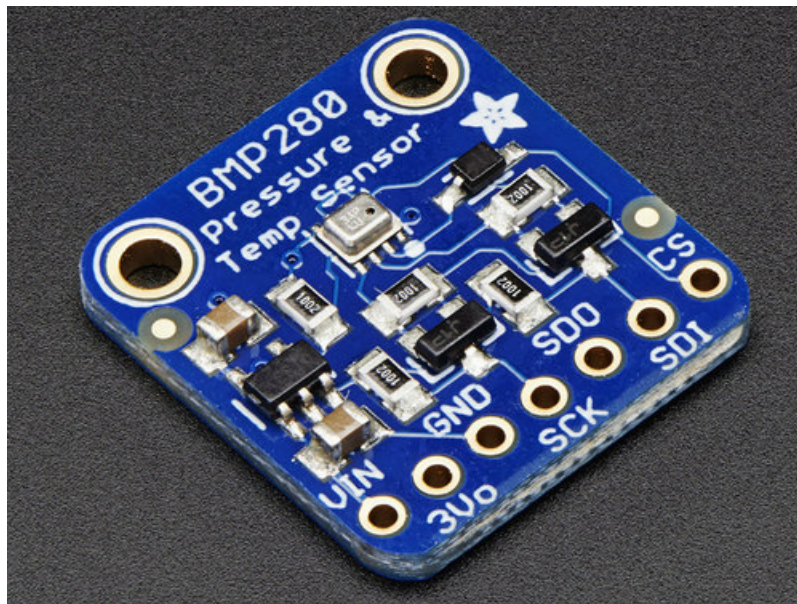




## Adafruit BMP280 Barometric Pressure + Temperature Sensor Breakout

Created by lady ada

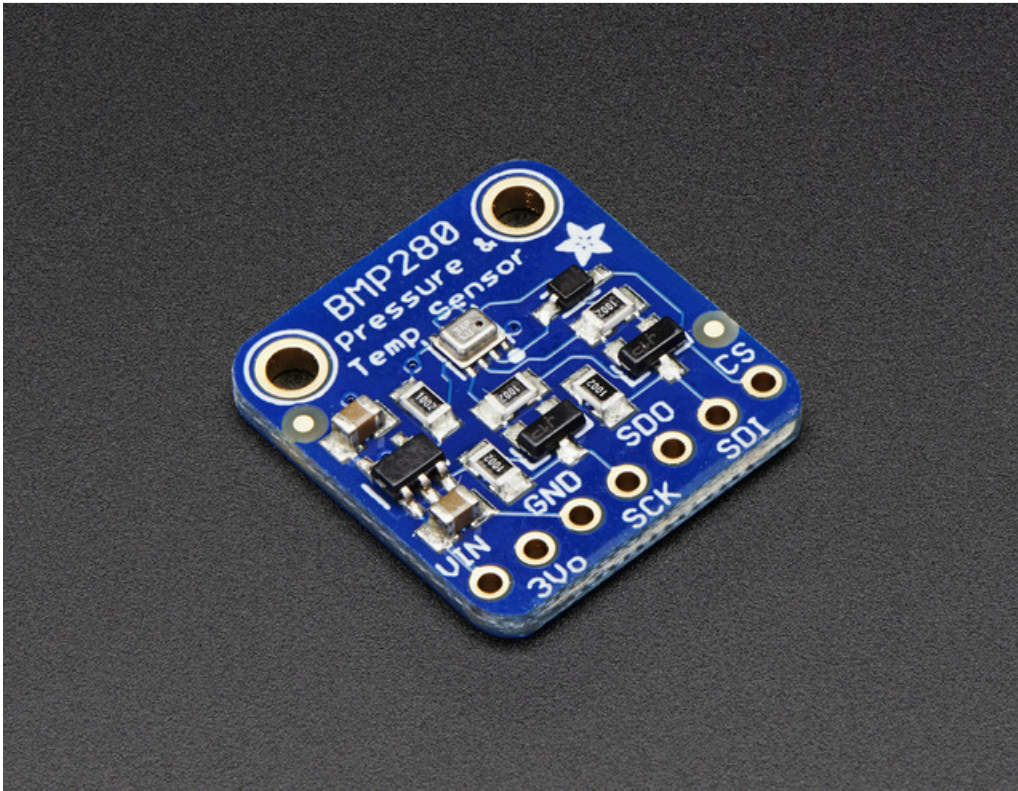


Last updated on 2018-05-07 06:09:39 PM UTC

## Guide Contents

Guide Contents	2
Overview	3
Pinouts	6
Power Pins:	6
SPI Logic pins:	6
I2C Logic pins:	6
Assembly	8
Prepare the header strip:	8
Add the breakout board:	9
And Solder!	10
Arduino Test	12
I2C Wiring	12
SPI Wiring	12
Download Adafruit_BMP280 library	12
Load Demo	13
Library Reference	15
CircuitPython Test	17
Usage	19
F.A.Q.	21
How come the altitude calculation is wrong? Is my sensor broken?	21
If I have long delays between reads, the first data read seems wrong?	21
Downloads	22
Documents	22
Schematic	22
Dimensions	22

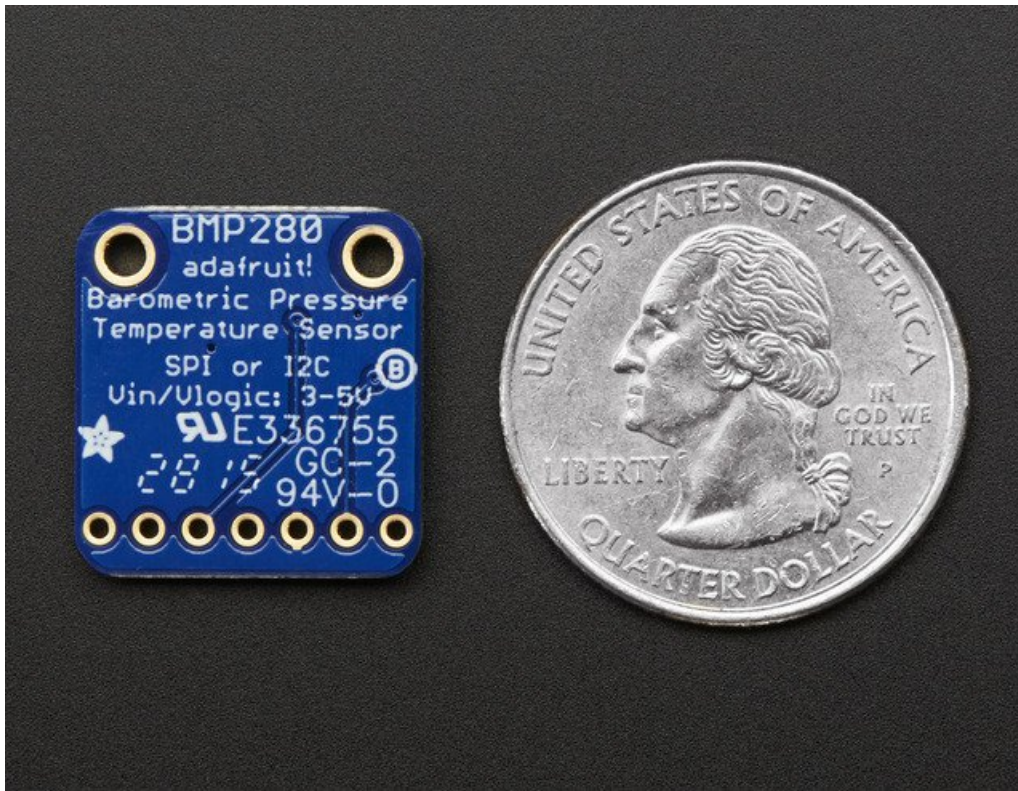
## Overview



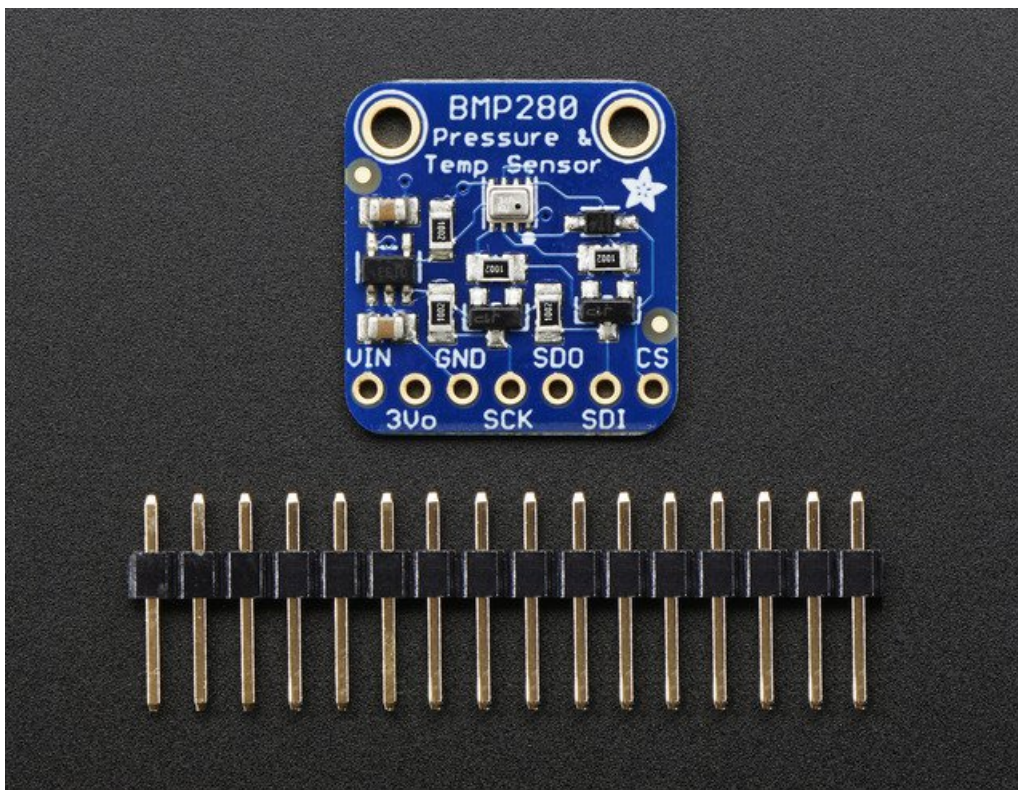
Bosch has stepped up their game with their new BMP280 sensor, an environmental sensor with temperature, barometric pressure that is the next generation upgrade to the BMP085/BMP180/BMP183. This sensor is great for all sorts of weather sensing and can even be used in both I2C and SPI!

This precision sensor from Bosch is the best low-cost, precision sensing solution for measuring barometric pressure with  $\pm 1$  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  $\pm 1$  meter accuracy





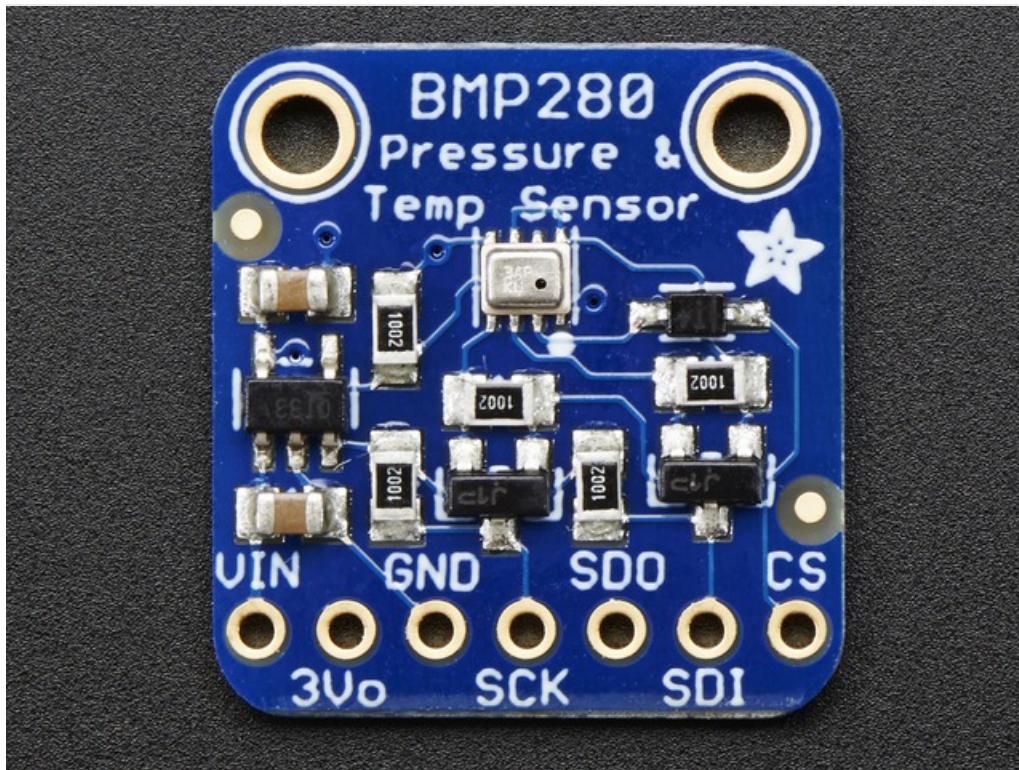
The BME280 is the next-generation of sensors from Bosch, and is the upgrade to the BMP085/BMP180/BMP183 - with a low altitude noise of 0.25m and the same fast conversion time. It has the same specifications, but can use either I2C or SPI. For simple easy wiring, go with I2C. If you want to connect a 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 **SPI Clock** pin, its an input to the chip
- **SDO** - this is the **Serial Data Out / Master In Slave Out** pin, for data sent from the BMP280 to your processor
- **SDI** - this is the **Serial Data In / Master Out Slave 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.

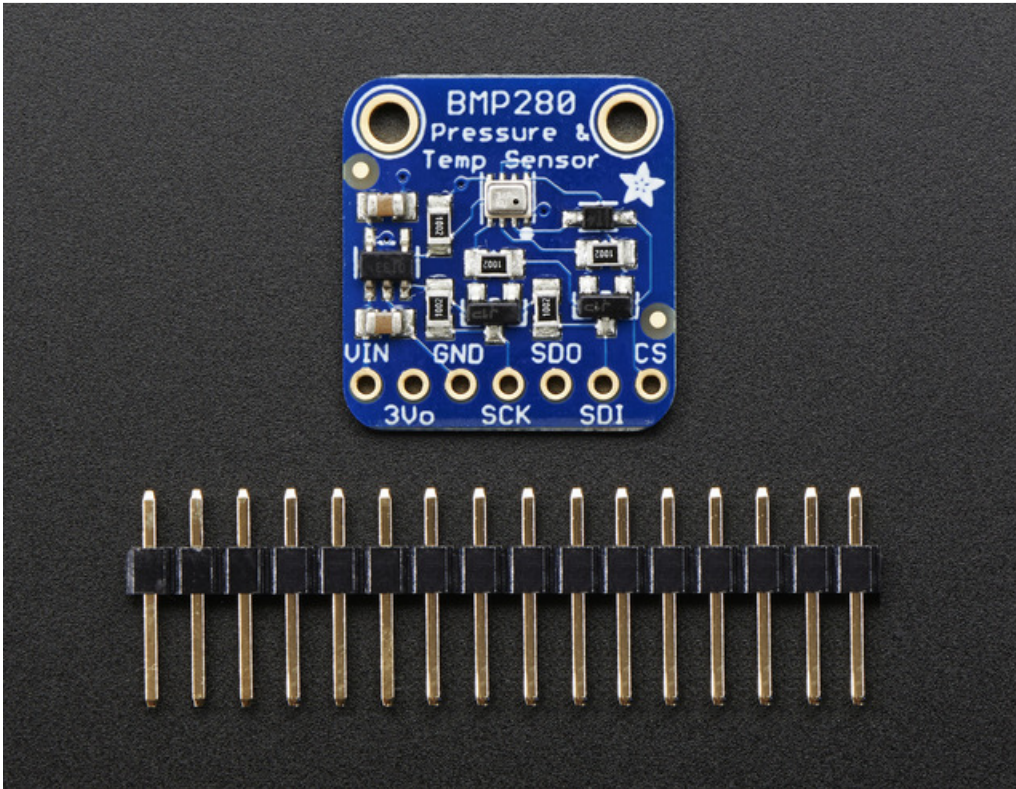
### I2C Logic pins:

- **SCK** - this is *also* the I2C clock pin, connect to your microcontrollers I2C clock line.
- **SDI** - this is *also* the I2C data pin, connect to your microcontrollers I2C data line.

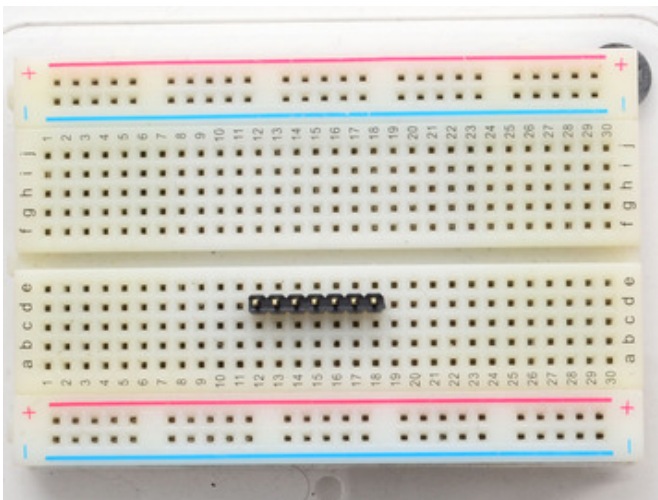
Leave the other pins disconnected



## Assembly



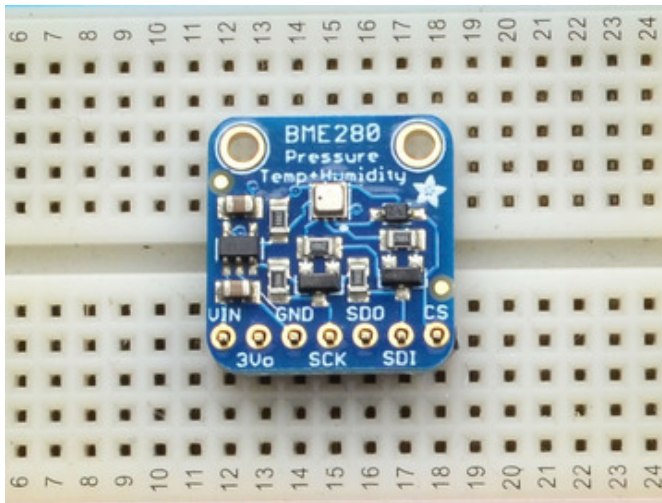
The assembly pix use the BME280 but it is identically shaped/sized as the BMP280



### Prepare the header strip:

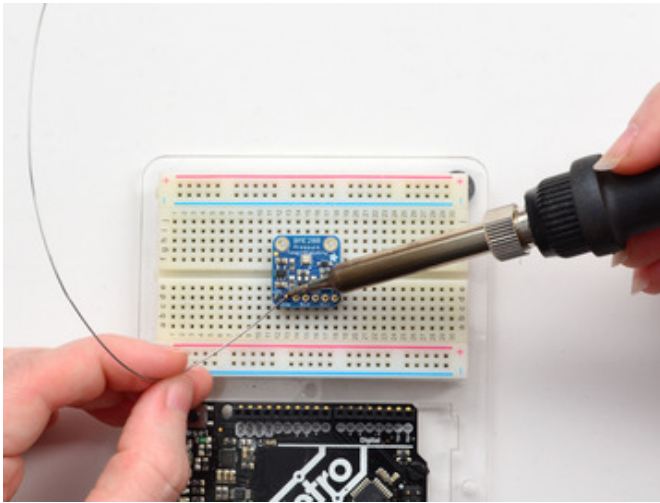
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**





### Add the breakout board:

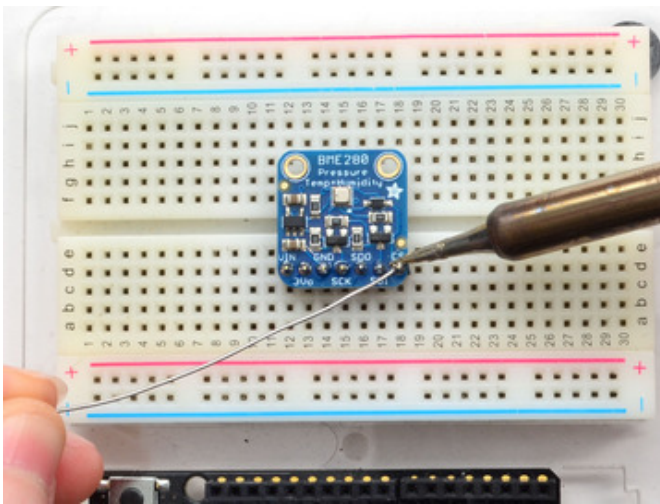
Place the breakout board over the pins so that the short pins poke through the breakout pads

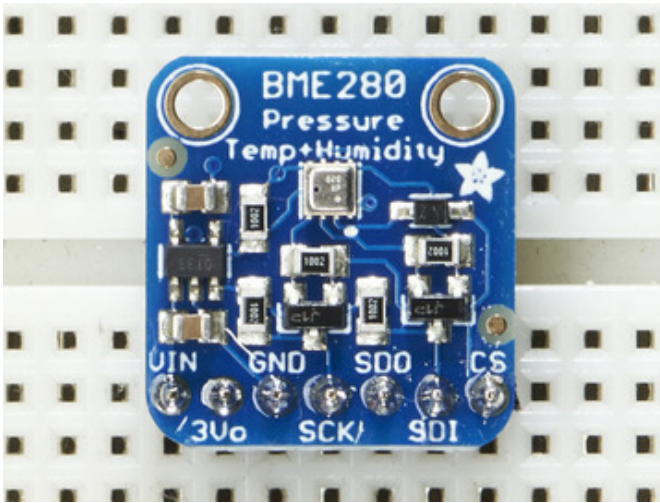


## And Solder!

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://adafruit.it/aTk) (<https://adafruit.it/aTk>)).





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

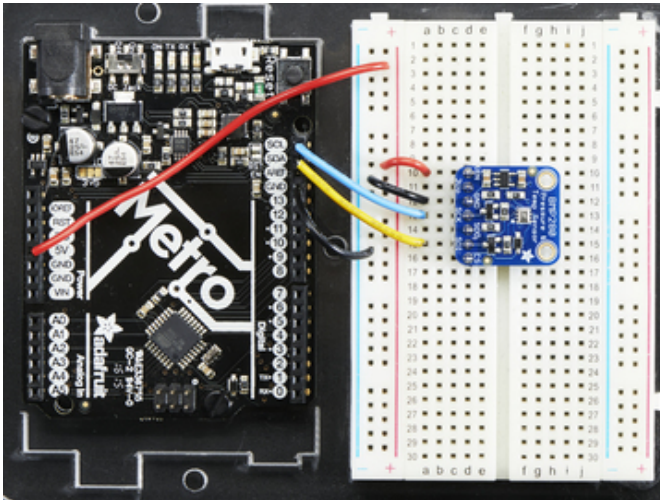


## Arduino Test

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

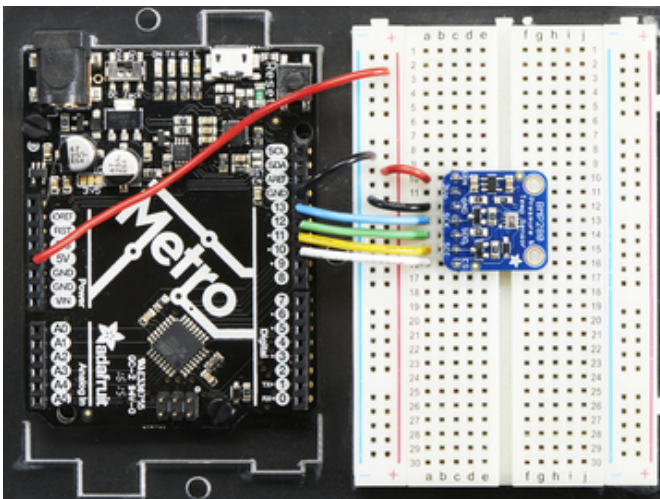
Use this wiring if you want to connect via I2C interface



- 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**

## 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:



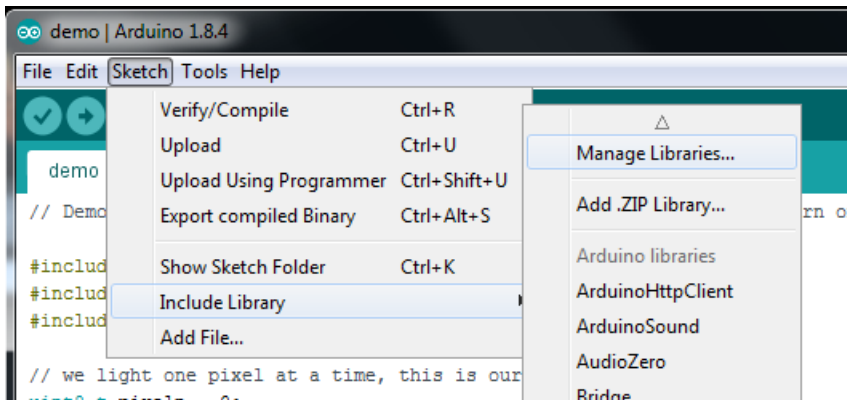
- Connect **Vin** to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect **GND** to common power/data ground
- Connect the **SCK** pin to **Digital #13** but any pin can be used later
- Connect the **SDO** pin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

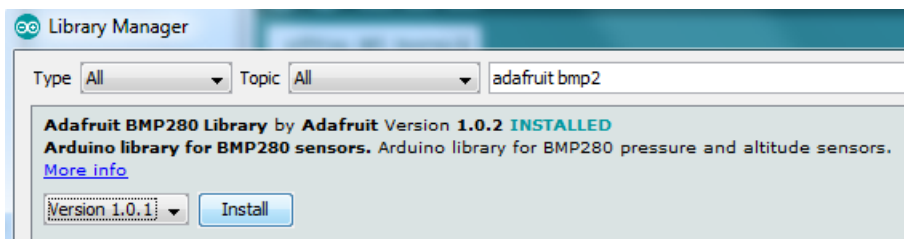
## Download Adafruit\_BMP280 library

To begin reading sensor data, you will need to [install the Adafruit\\_BMP280 library \(code on our github repository\)](#). It is available from the Arduino library manager so we recommend using that.

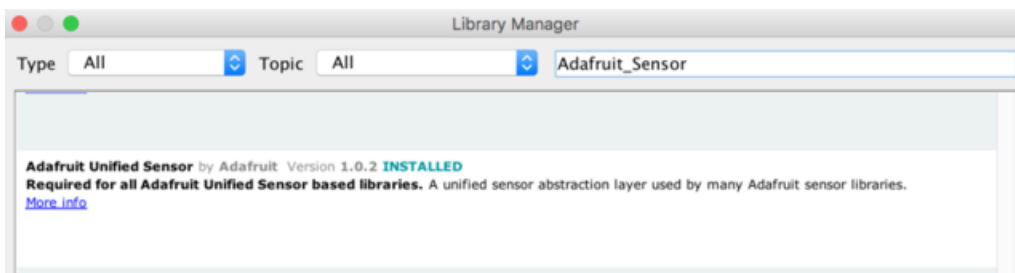
From the IDE open up the library manager...



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



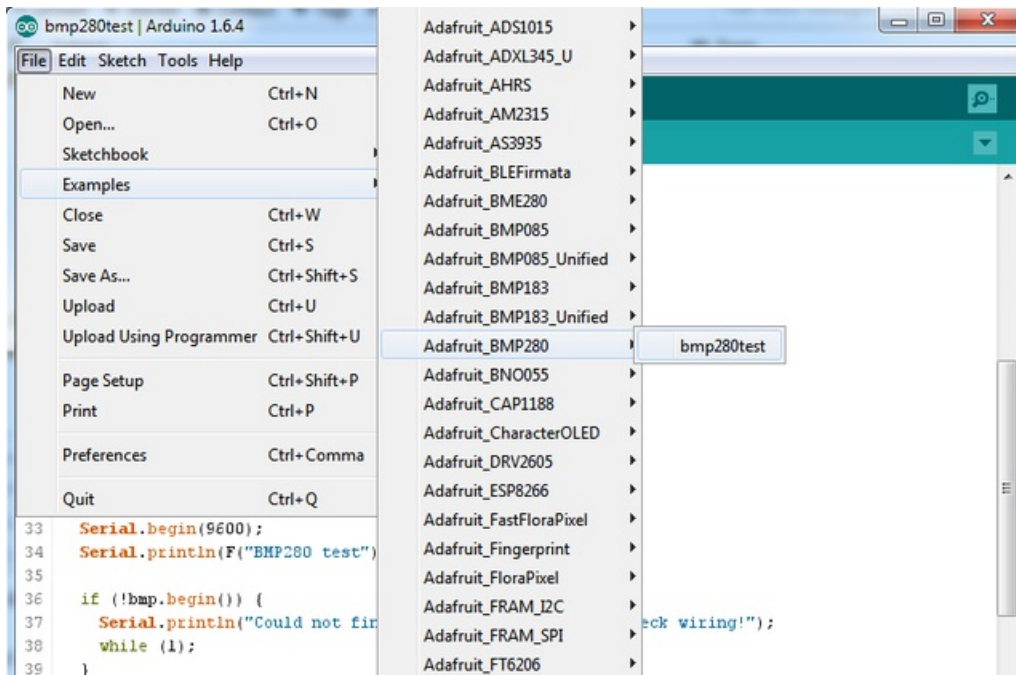
You'll also need to install the **Adafruit Unified Sensor** library



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

## Load Demo

Open up **File->Examples->Adafruit\_BMP280->bmp280test** and upload to your Arduino wired up to the sensor



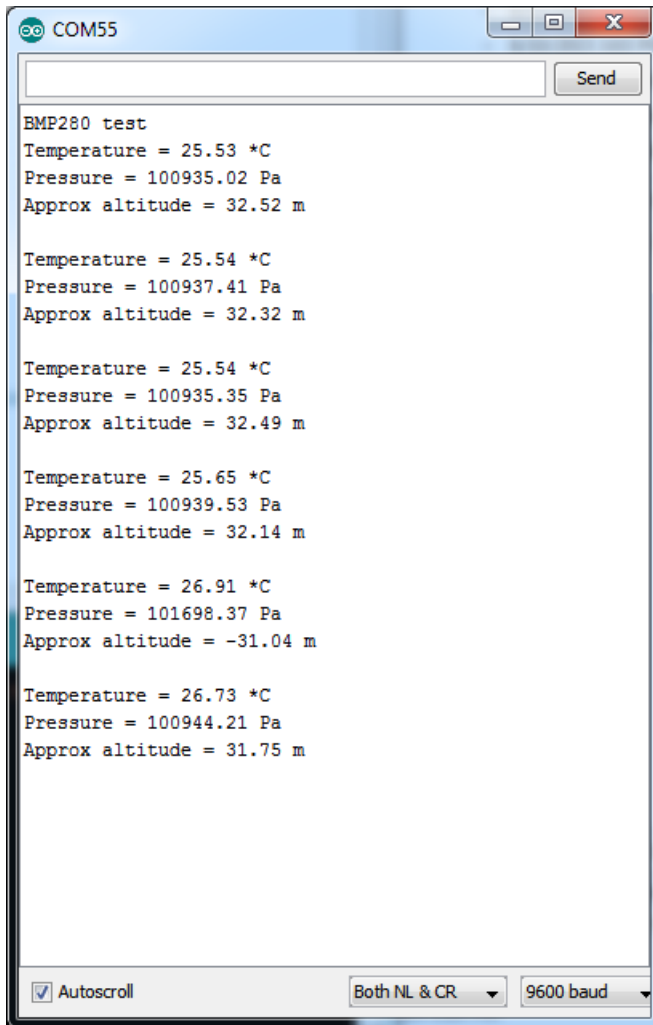
Depending on whether you are using I2C or SPI, change the pin names and comment or uncomment the following lines.

```
#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  $F = C * 9/5 + 32$  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.](#)

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()
```

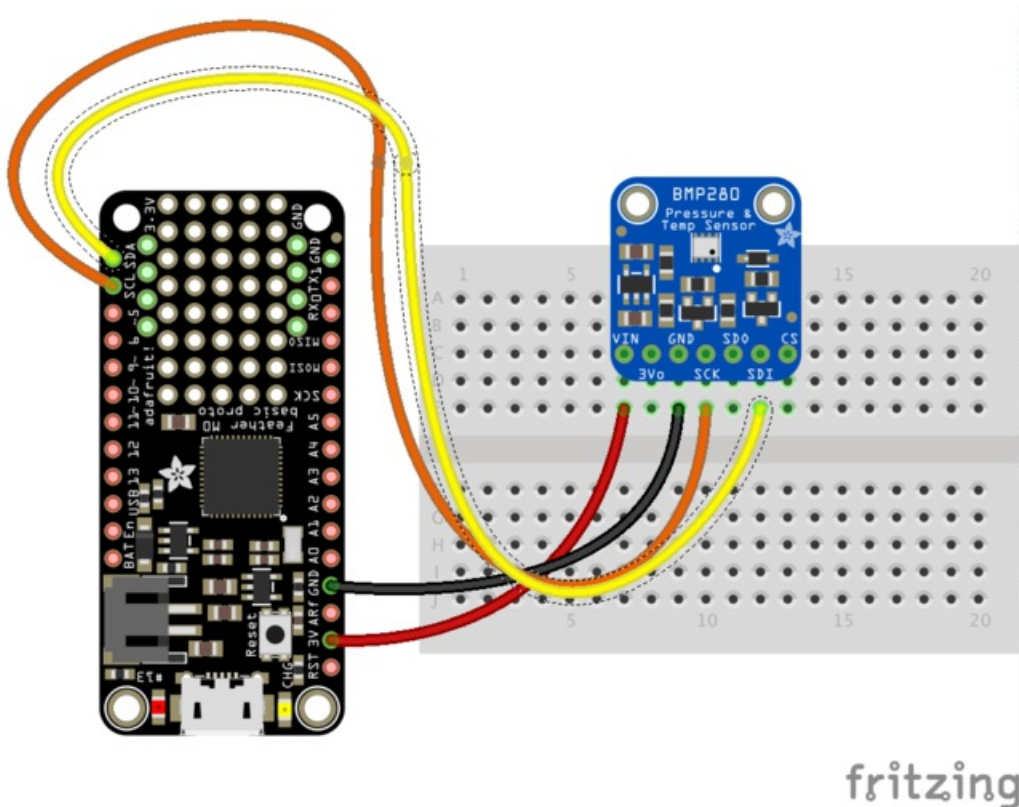
Temperature is always a floating point, in Centigrade. Pressure is a 32 bit integer with the pressure in Pascals. You may need to convert to a different value to match it with your weather report.

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

## CircuitPython Test

It's easy to use the BMP280 sensor with CircuitPython and the [Adafruit CircuitPython BMP280](#) module. This module allows you to easily write Python code that reads the temperature and pressure from the sensor.

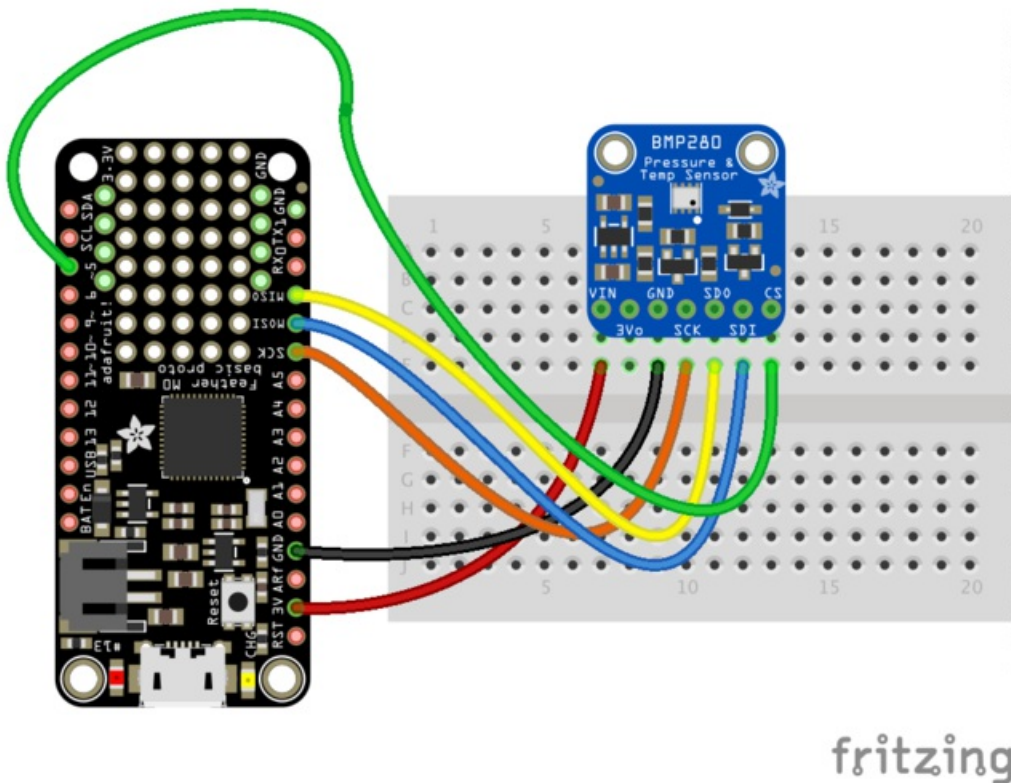
First wire up a BMP280 to your board exactly as shown on the previous pages for Arduino. You can use either I2C or SPI wiring, although it's recommended to use I2C for simplicity. Here's an example of wiring a Feather M0 to the sensor with I2C:



- 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)

Next you'll need to install the [Adafruit CircuitPython BMP280](#) library on your CircuitPython board. **Remember this module is for Adafruit CircuitPython firmware and not MicroPython.org firmware!**

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](#). For example the Circuit Playground Express guide has [a great page on how to install the library bundle](#) for both express and non-express boards.

Remember for non-express boards like the Trinket M0, Gemma M0, and Feather/Metro M0 basic you'll need to manually install the necessary libraries from the bundle:

- `adafruit_bmp280.mpy`
- `adafruit_bus_device`

You can also download the `adafruit_bmp280.mpy` from [its releases page on Github](#).

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](#) so you are at the CircuitPython `>>>` prompt.

## 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_bmp280
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)
```

Remember if you're using a board that doesn't support hardware I2C (like the ESP8266) you need to use the **bitbangio** module instead:

```
import board
import bitbangio
import adafruit_bmp280
i2c = bitbangio.I2C(board.SCL, board.SDA)
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 busio
import digitalio
import adafruit_bmp280
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
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
>>> █
```

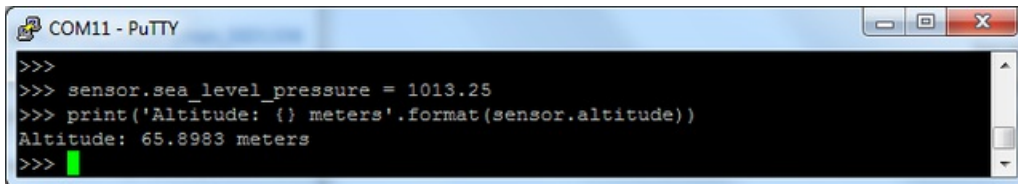
For altitude you'll want to set the pressure at sea level for your location to get the most accurate measure (remember

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))
```



```
COM11 - PuTTY
>>>
>>> sensor.sea_level_pressure = 1013.25
>>> print('Altitude: {} meters'.format(sensor.altitude))
Altitude: 65.8983 meters
>>>
```

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:

```
import time

import board
# import digitalio # For use with SPI
import busio

import adafruit_bmp280

# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)

# OR create library object using our Bus SPI port
#spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
#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)
```



## 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?**

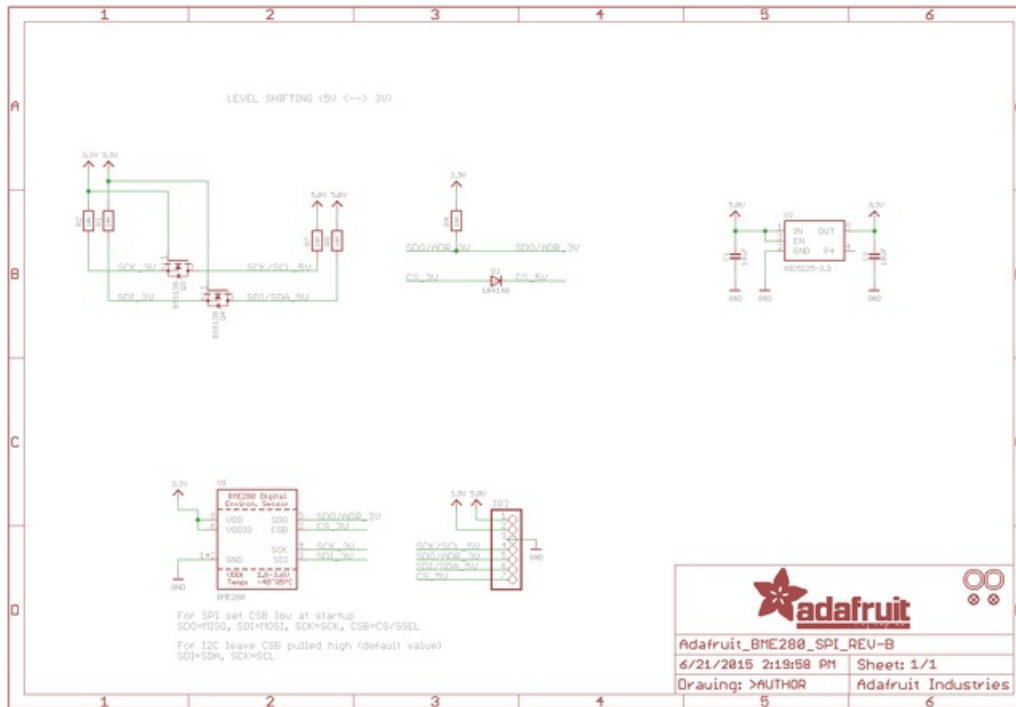
The BMx280 'saves' the last reading in memory for you to query. Just read twice in a row and toss out the first reading!

## Documents

- [Datasheet for the BMP280 sensor used in the breakout](#)
- [Arduino BMP280 Driver](#)
- [Fritzing object in the Adafruit Fritzing Library](#)
- [EagleCAD PCB files on GitHub](#)

## Schematic

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



## Dimensions

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

