Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

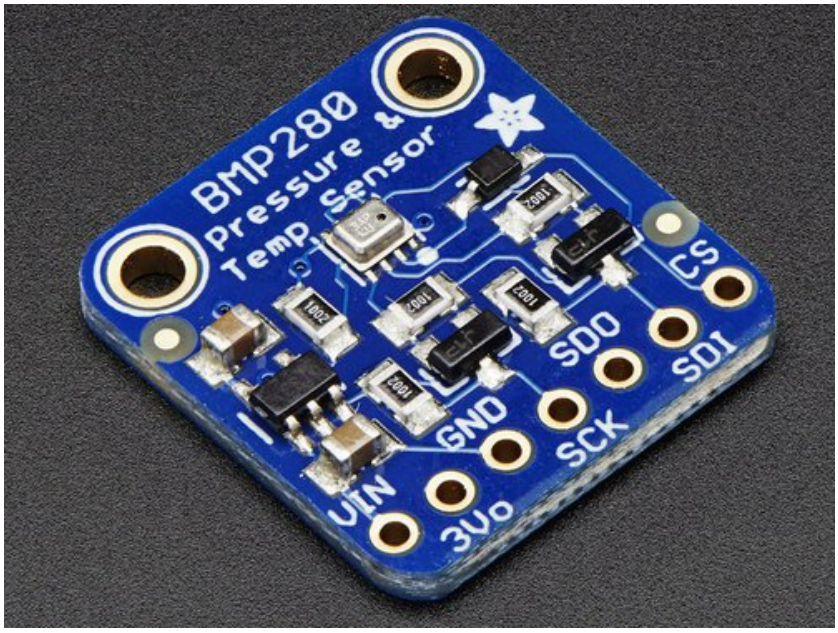
We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Adafruit BMP280 Barometric Pressure + Temperature Sensor Breakout

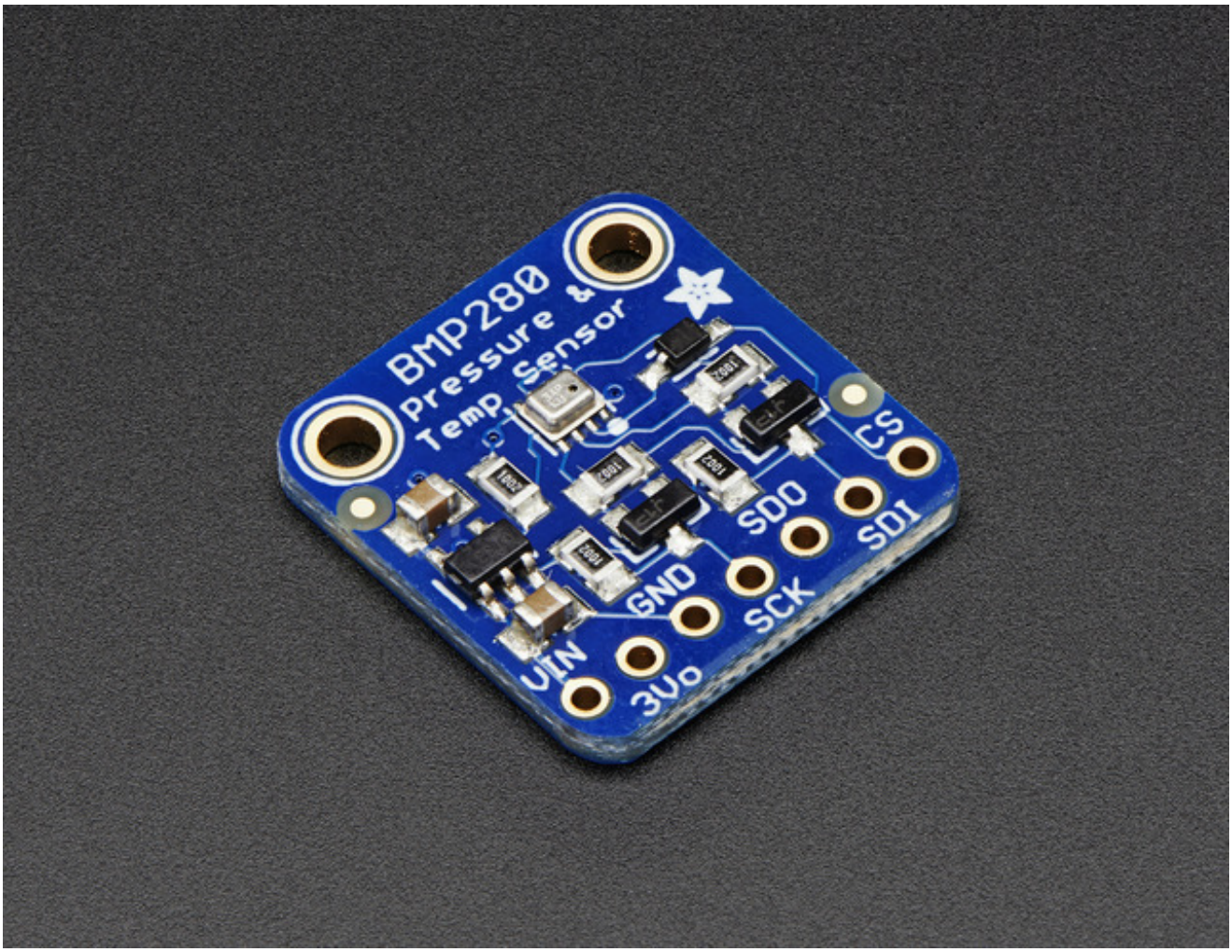
Created by lady ada



Last updated on 2017-01-12 10:40:25 PM UTC

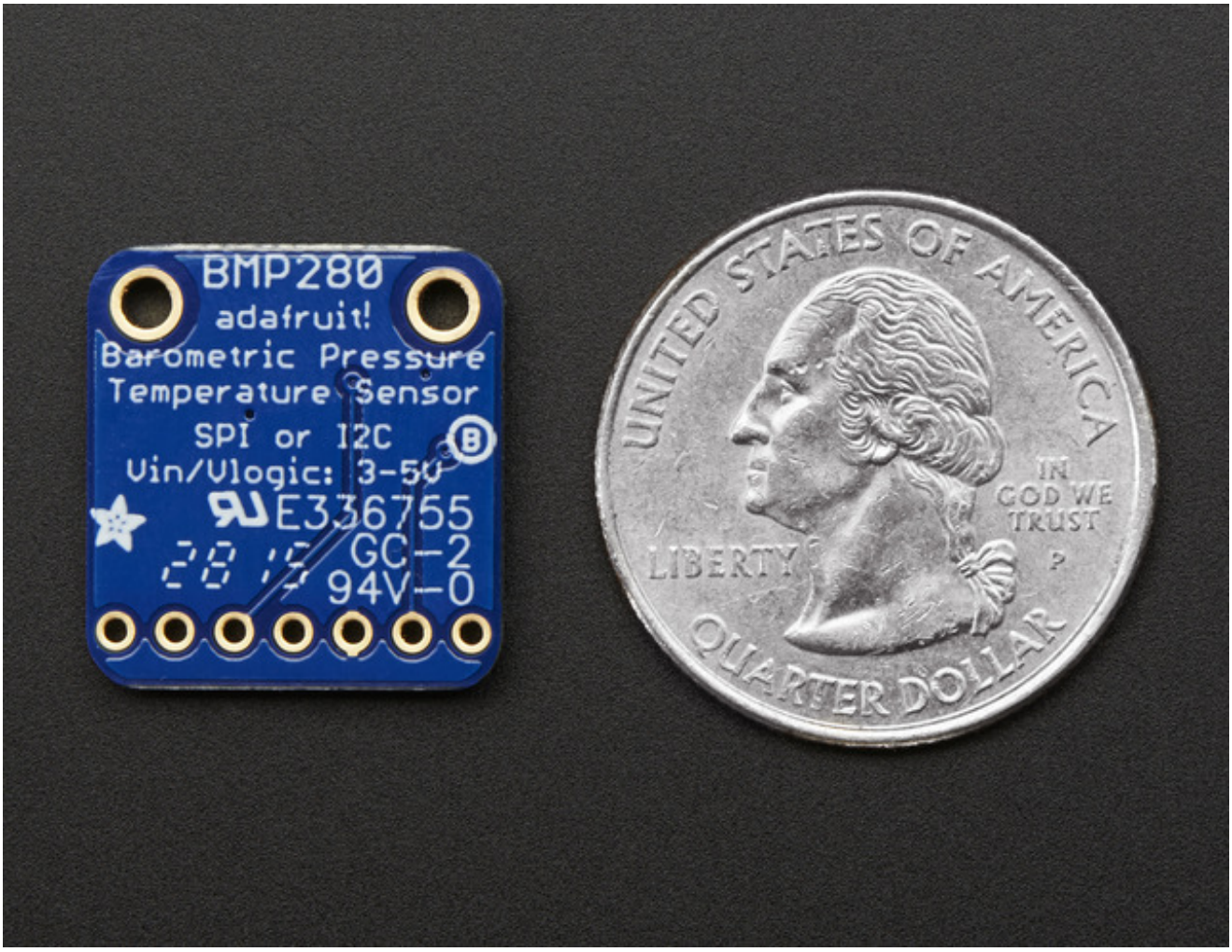
Guide Contents

Overview

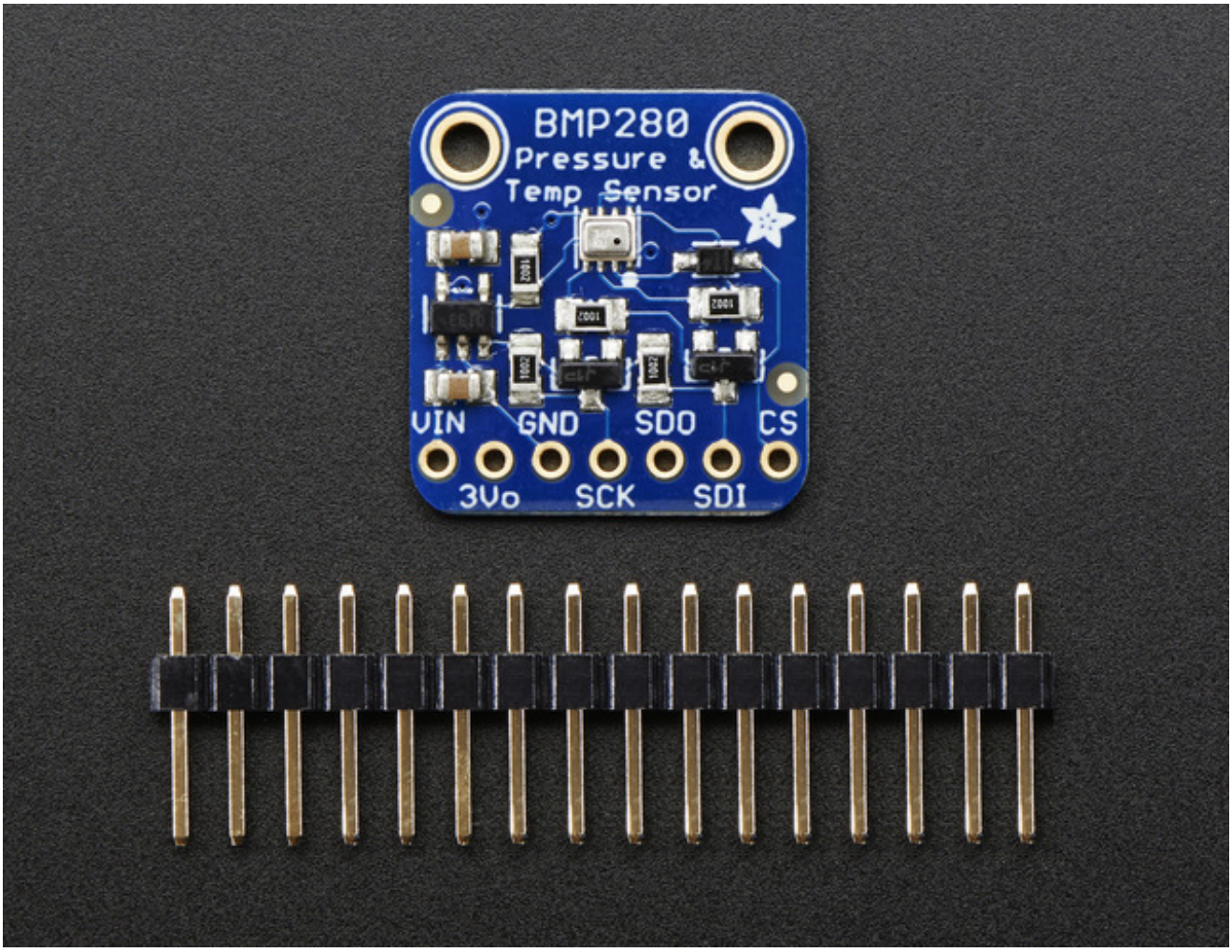


Bosch has stepped up their game with their new BMP280 sensor, an environmental sensor with temperature, barometric pressure that is the next generation upgrade to the BMP085/BMP180/BMP183. This sensor is great for all sorts of weather sensing and can even be used in both I2C and SPI!

This precision sensor from Bosch is the best low-cost, precision sensing solution for measuring barometric pressure with ±1 hPa absolute 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

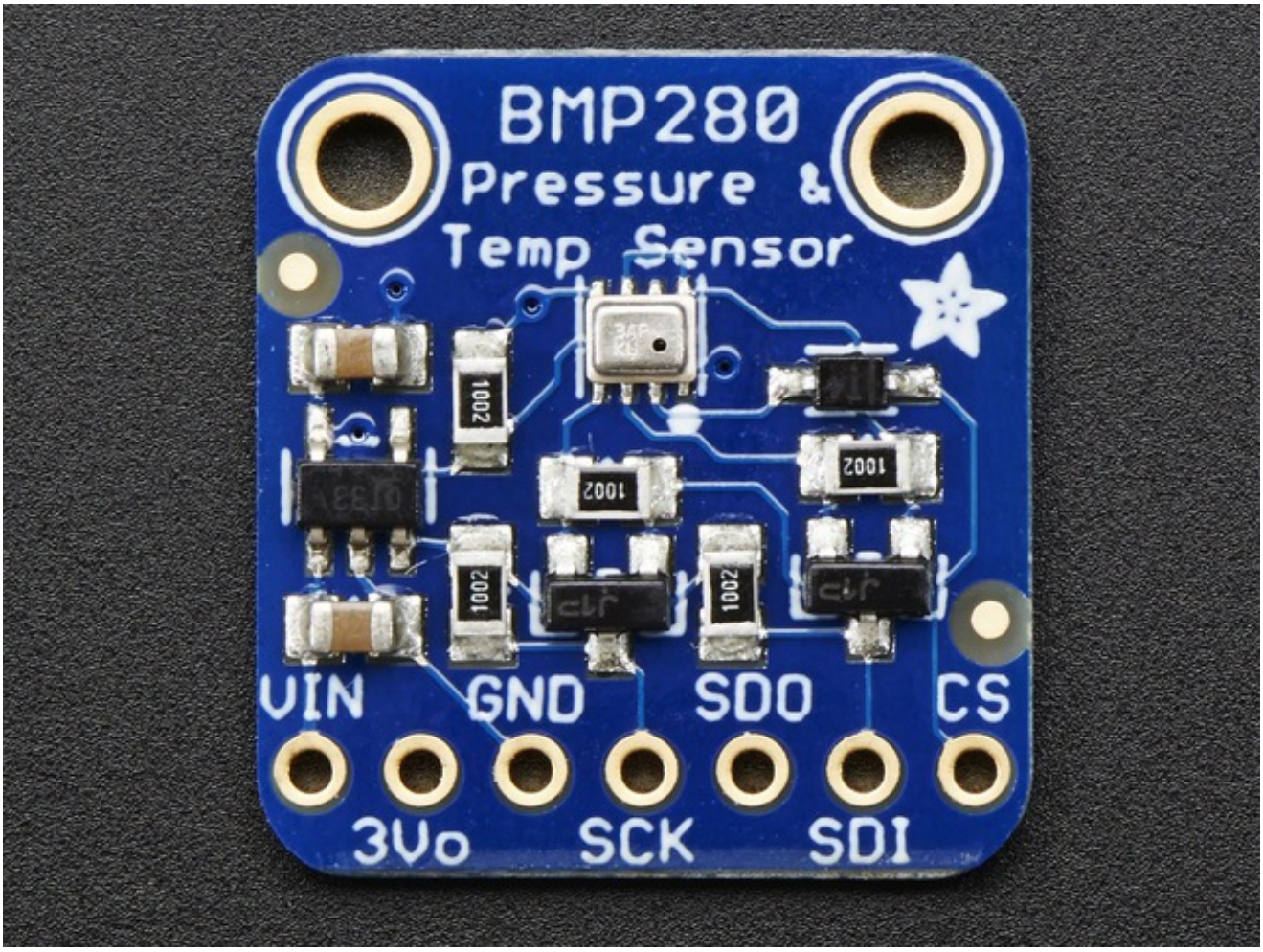


The BME280 is the next-generation of sensors from Bosch, and is the upgrade to the BMP085/BMP180/BMP183 - with a low altitude noise of 0.25m and the same fast conversion time. It has the same specifications, but can use either I2C *or* SPI. For simple easy wiring, go with I2C. If you want to connect 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**loc**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

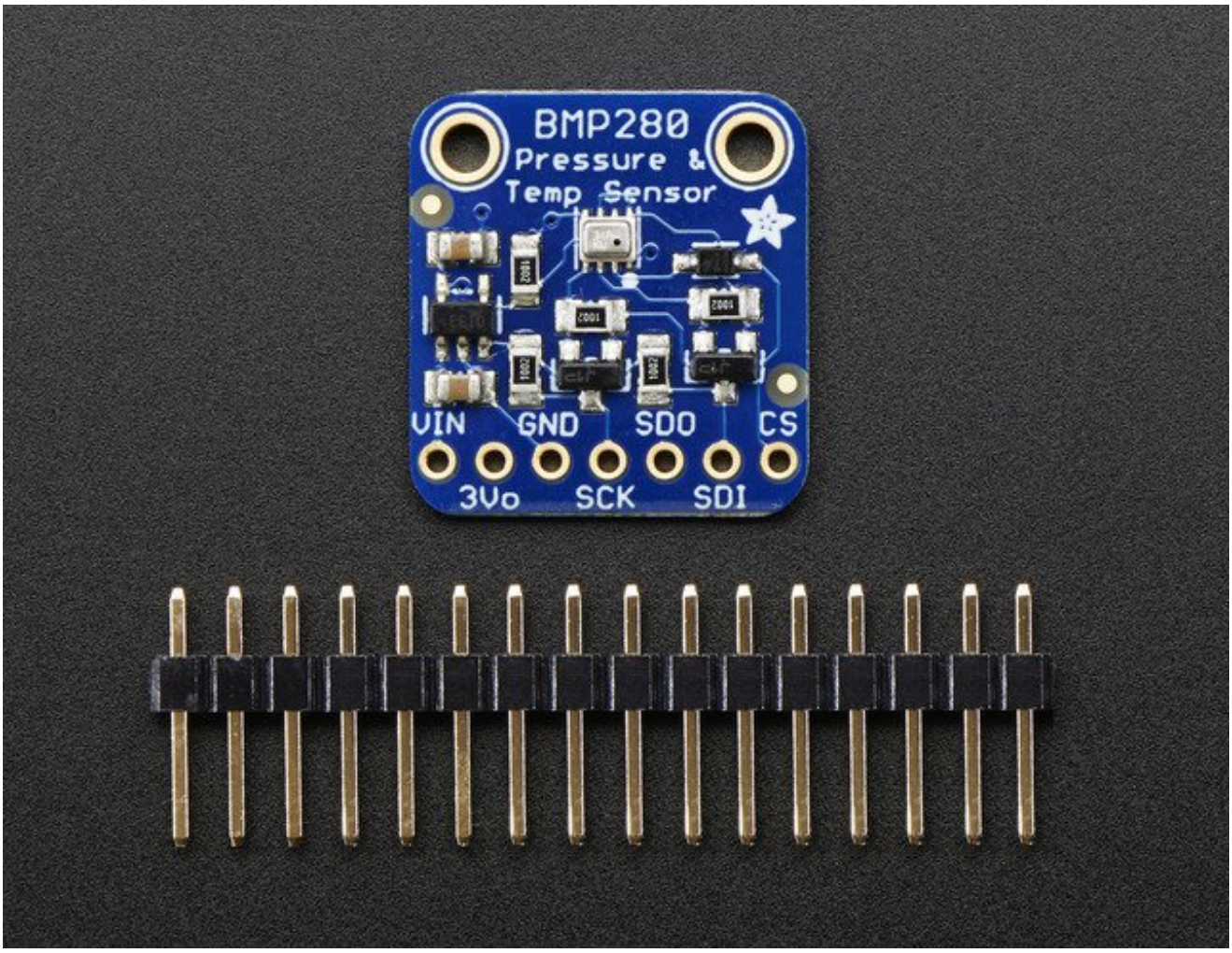
If you want to connect multiple BME280's to one microcontroller, have them share the SDI, SDO and SCK pins. Then assign each one a unique CS pin.

I2C Logic pins:

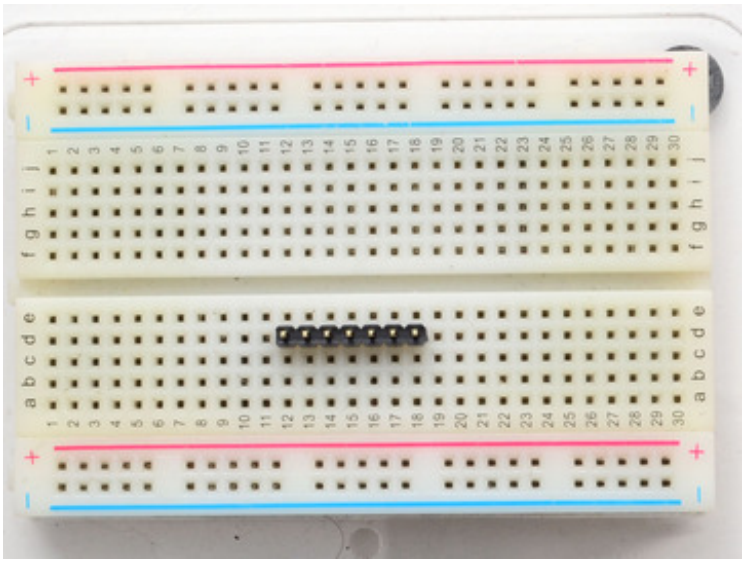
- **SCK** - this is *also* the I2C clock pin, connect to your microcontrollers I2C clock line.
- **SDI** - this is *also* the I2C data pin, connect to your microcontrollers I2C data line.

Leave the other pins disconnected

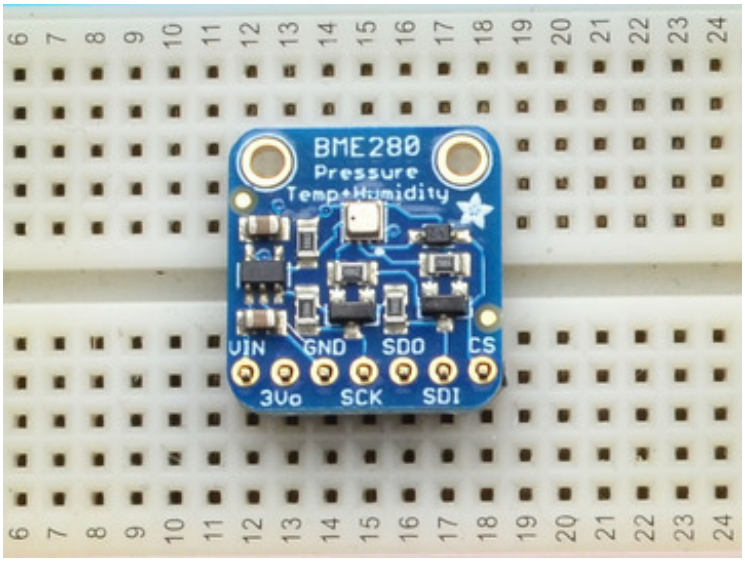
Assembly



Prepare the header strip:

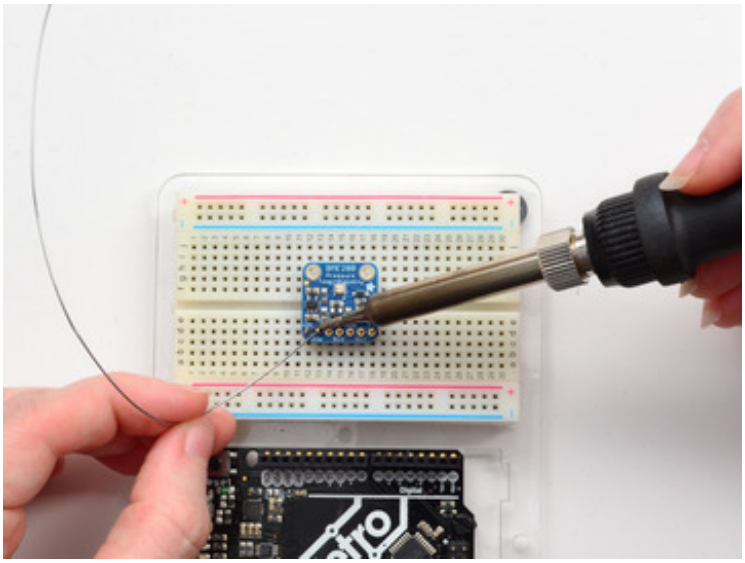
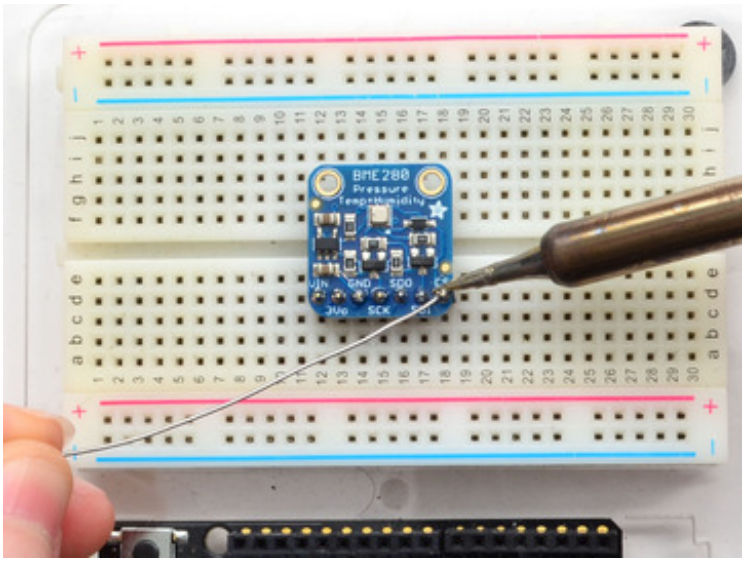
- 

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](http://adafru.it/aTk)).

- 

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

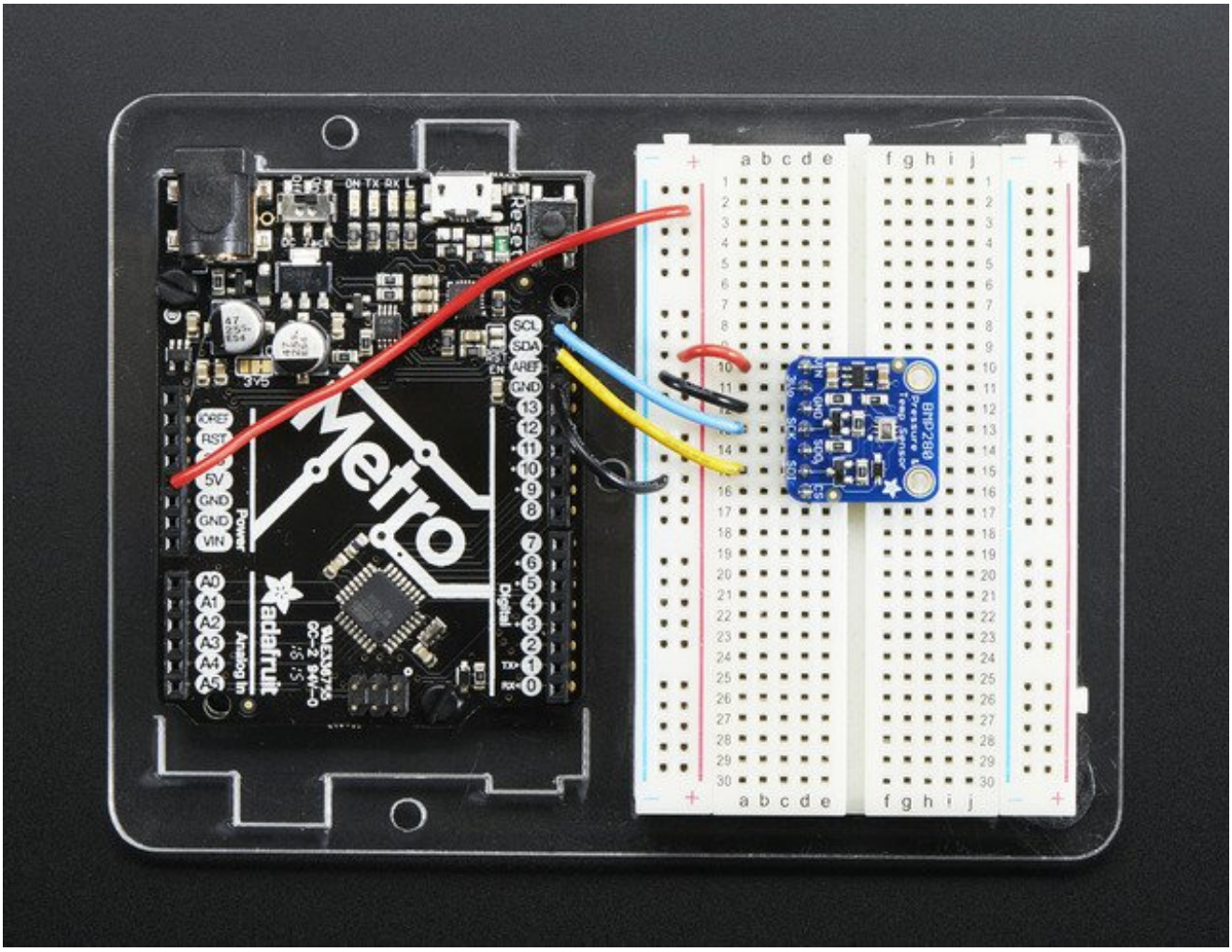
Wiring & Test

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**

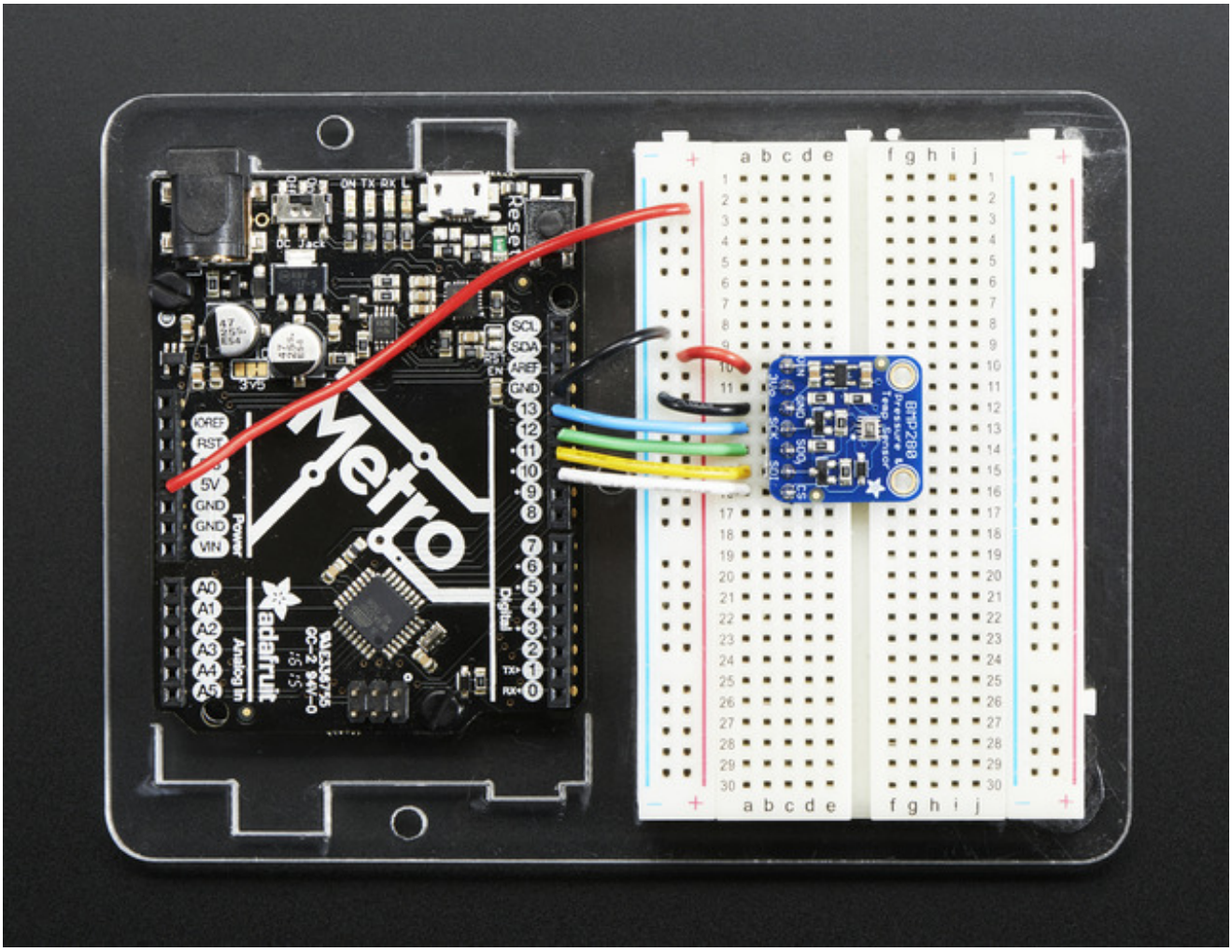


SPI Wiring

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[download Adafruit_BMP280 from our github repository](http://adafru.it/fIK). You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

[Download Adafruit BMP280 Library](http://adafru.it/fIL)
http://adafru.it/fIL

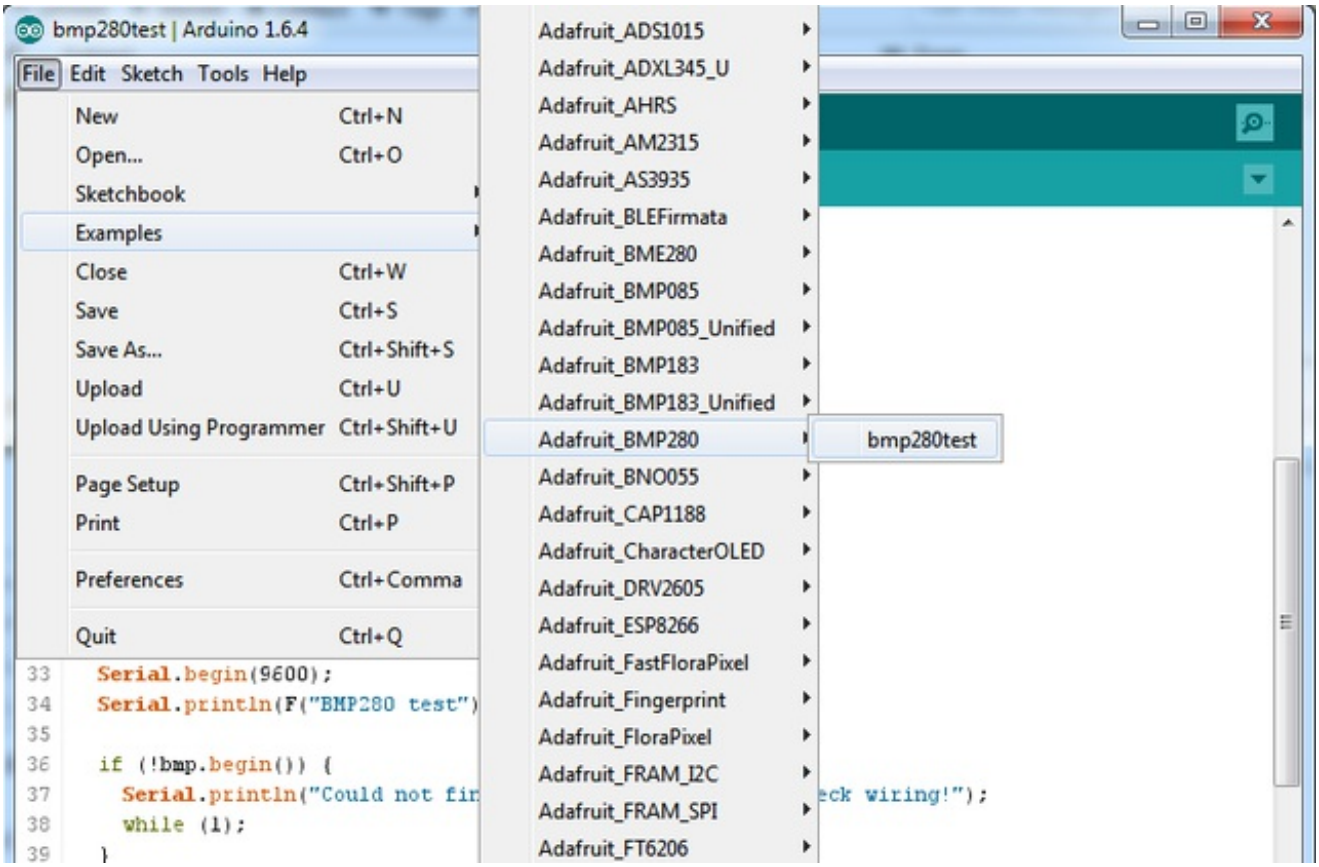
Rename the uncompressed folder **Adafruit_BMP280** and check that the **Adafruit_BMP280** folder contains **Adafruit_BMP280.cpp** and **Adafruit_BMP280.h**

Place the **Adafruit_BMP280** library folder your **arduinosketchfolder/libraries/** folder. You may need to create the**libraries** subfolder if its your first library. Restart the IDE.

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

Load Demo

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

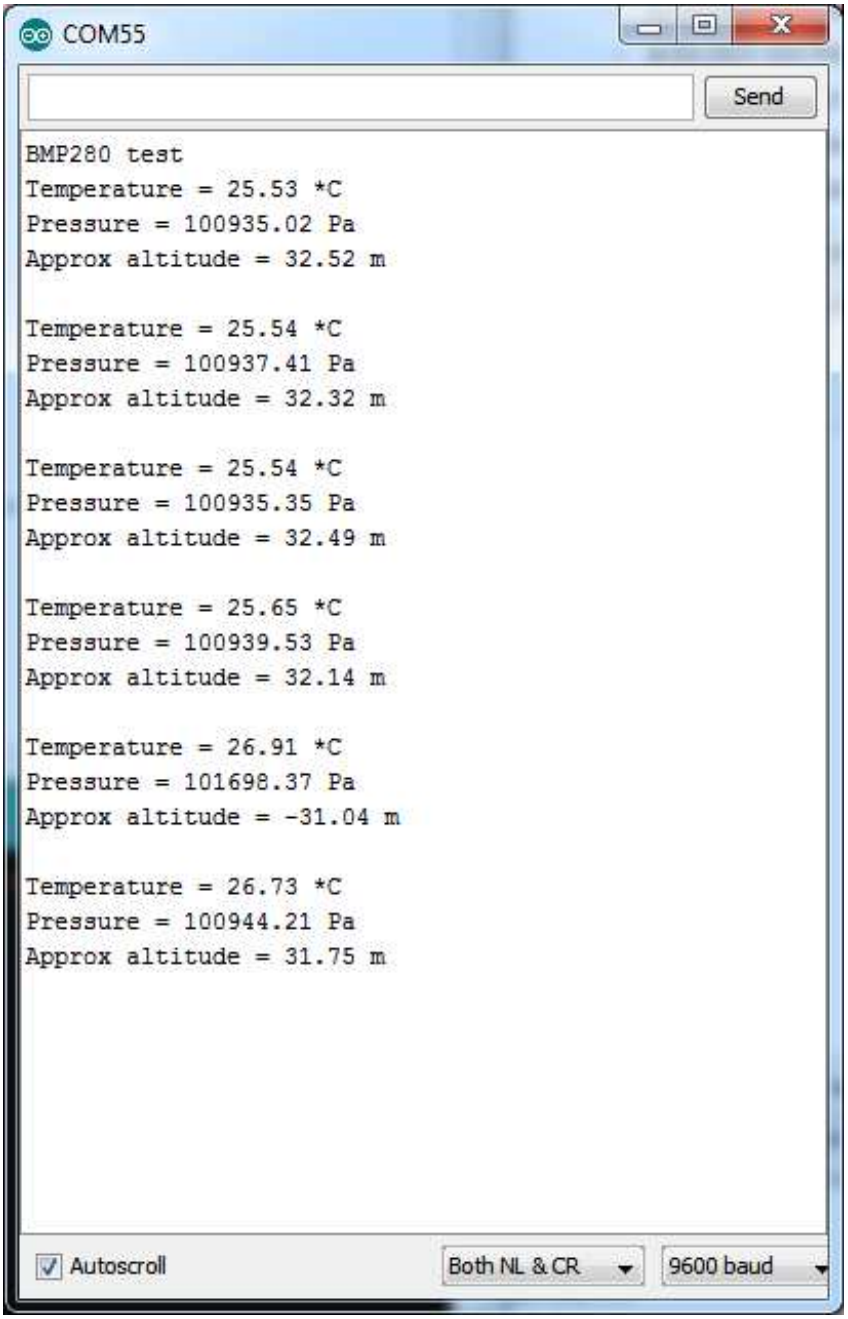


Depending on whether you are using I2C or SPI, change the pin names and comment or uncomment the following lines.

```
#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 * 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. (http://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

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!

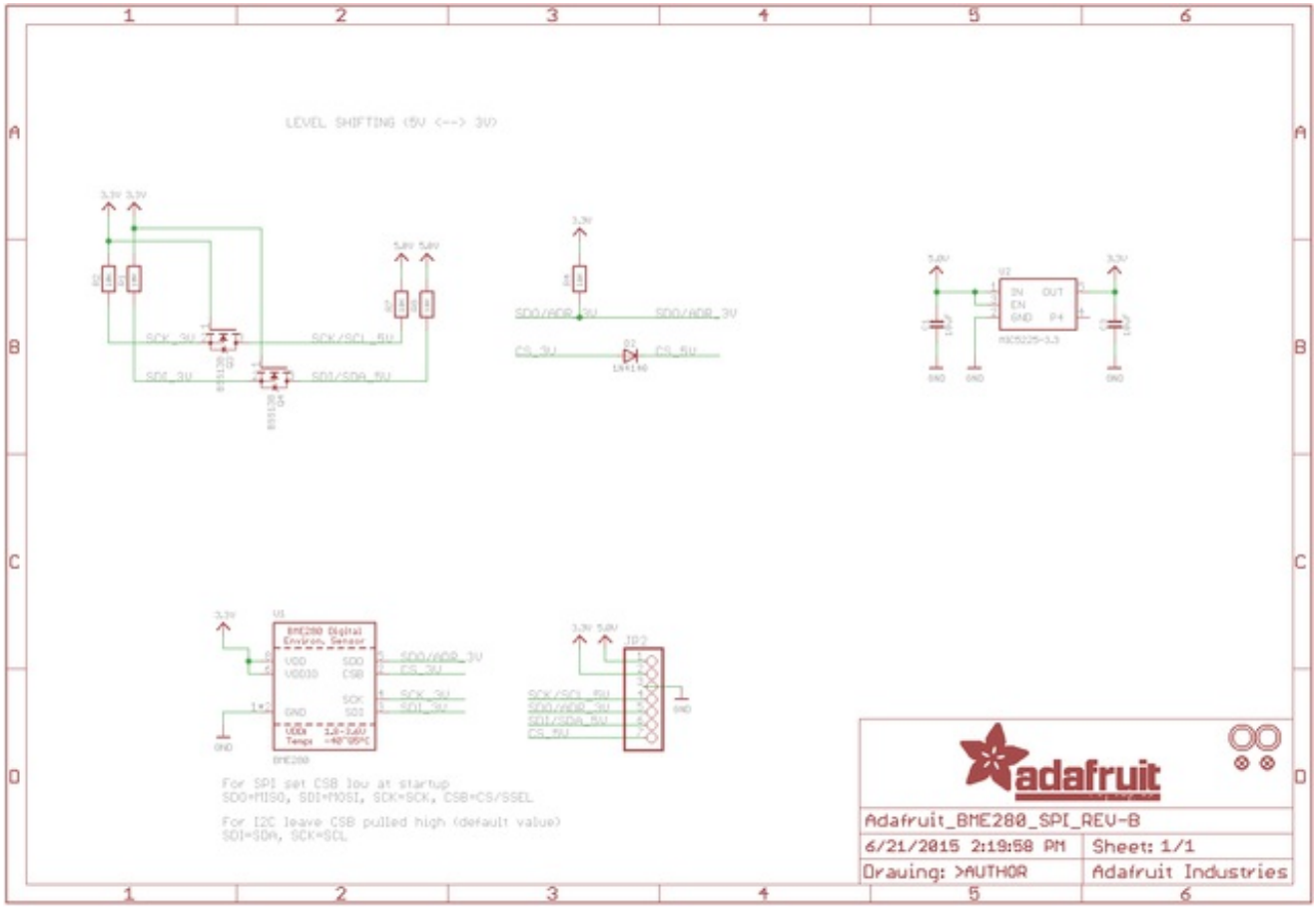
Downloads

Documents

- Datasheet for the BMP280 sensor used in the breakout(http://adafru.it/fIO)
- Arduino BMP280 Driver (http://adafru.it/fIK)
- Fritzing object in the Adafruit Fritzing Library(http://adafru.it/aP3)
- EagleCAD PCB files on GitHub (http://adafru.it/rDq)

Schematic

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



Dimensions

Dimensions

In inches