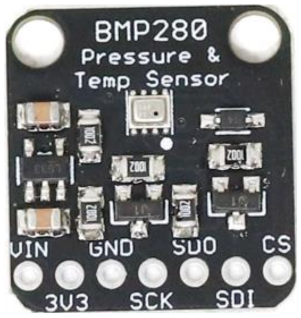
SmartElex BMP280- I2C or SPI Barometric Pressure & Altitude Sensor Breakout Board



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

This precision sensor from Bosch is the best low-cost, precision sensing solution for measuring barometric pressure with ±1 hPa absolute accuracy, and temperature with ±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 BMP280 is the next-generation of sensors from Bosch, and is the upgrade to the BMP085/BMP180/BMP183 - with a low altitude noise of 0.25m and the same fast conversion time. It has the same specifications, but can use either I2C or SPI. For simple easy wiring, go with I2C. If you want to connect a bunch of sensors without worrying about I2C address collisions, go with SPI.

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**icrocontroller **I**n **S**ensor **O**ut pin, for data sent from the BMP280 to your processor
- **SDI** - this is the **S**erial **D**ata **I**n / **M**icrocontroller **O**ut **S**ensor **I**n pin, for data sent from your processor to the BMP280
- **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 BMP280's to one microcontroller, have them share the SDI, SDO and SCK pins. Then assign each one a unique CS pin.

I2C Logic pins:

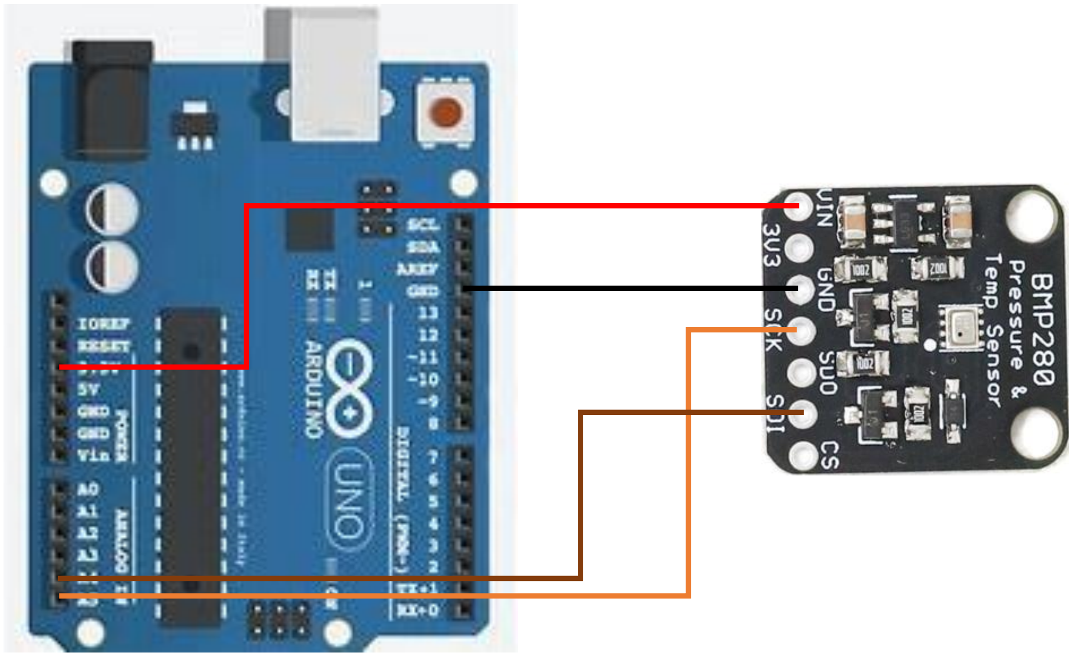
- **SCK** - this is *also* the I2C clock pin **(SCL)**, connect to your microcontroller's I2C clock line.
- **SDI** - this is *also* the I2C data pin **(SDA),** connect to your microcontroller's I2C data line.
- Leave the other pins disconnected

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

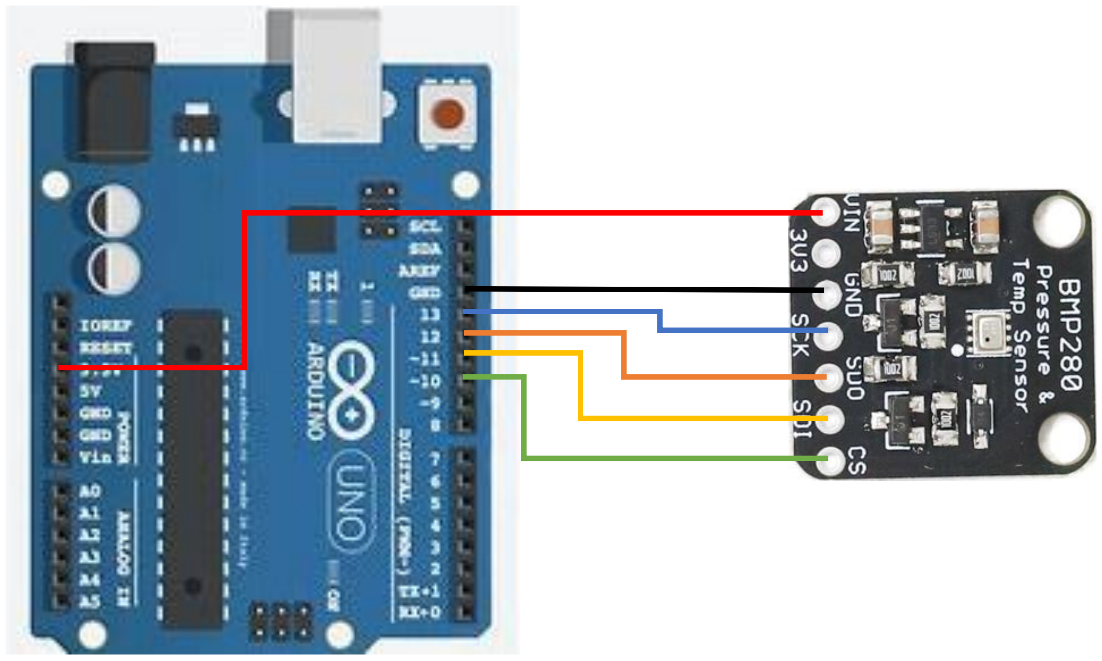


Arduino	BMP280
SCL(A5)	SCK
SDA(A4)	SDI
5v OR 3.3v	VIN
GND	GND

- 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:



Arduino	BMP280
D13(SCK)	SCK
D12(MISO)	SDO
D11(MOSI)	SDI
D10(SS)	CS
5v OR 3.3v	VIN
GND	GND

- 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. 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

Example Code

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

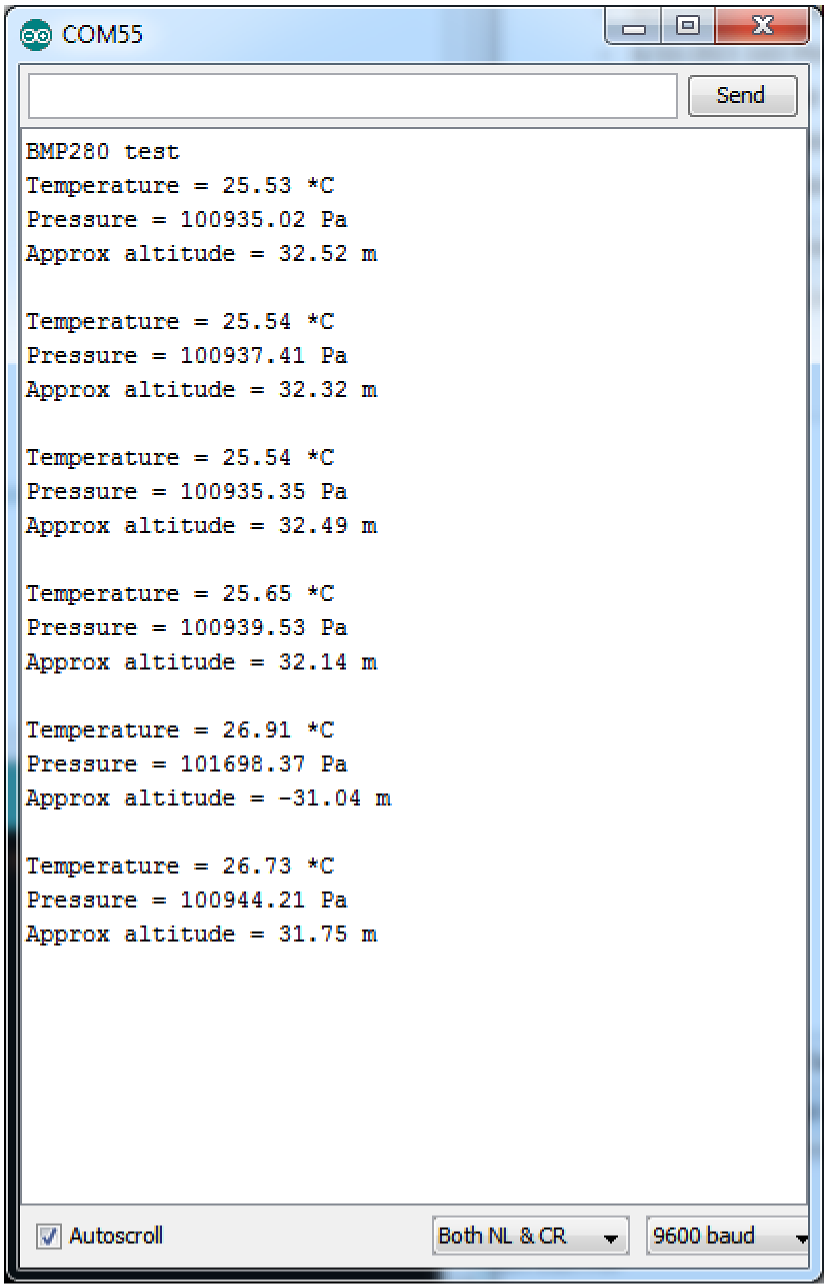
```
#include <Wire.h>

#include <SPI.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_BME280.h>

#define BME_SCK 13

#define BME_MISO 12

#define BME_MOSI 11

#define BME_CS 10

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C

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

//Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // software SPI

unsigned long delayTime;

void setup() {

   Serial.begin(9600);

   while(!Serial);    // time to get serial running

   Serial.println(F("BME280 test"));

   unsigned status;

   // default settings

   status = bme.begin();

   // You can also pass in a Wire library object like &Wire2
```

```
  // status = bme.begin(0x76, &Wire2)
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring, address, sensor
ID!");
    Serial.print("SensorID was: 0x"); Serial.println(bme.sensorID(),16);
    Serial.print("       ID of 0xFF probably means a bad address, a BMP 180 or BMP
085\n");
    Serial.print("   ID of 0x56-0x58 represents a BMP 280,\n");
    Serial.print("       ID of 0x60 represents a BME 280.\n");
    Serial.print("       ID of 0x61 represents a BME 680.\n");
    while (1) delay(10);
  }
  Serial.println("-- Default Test --");
  delayTime = 1000;
  Serial.println();
}
void loop() {
  printValues();
  delay(delayTime);
}
void printValues() {
  Serial.print("Temperature = ");
  Serial.print(bme.readTemperature());
  Serial.println(" °C");
  Serial.print("Pressure = ");
  Serial.print(bme.readPressure() / 100.0F);
  Serial.println(" hPa");
  Serial.print("Approx. Altitude = ");

  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));

  Serial.println(" m");

  Serial.print("Humidity = ");

  Serial.print(bme.readHumidity());

  Serial.println(" %");

  Serial.println();

}
```

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



```
BMP280 test
Temperature = 25.53 *C
Pressure = 100935.02 Pa
Approx altitude = 32.52 m

Temperature = 25.54 *C
Pressure = 100937.41 Pa
Approx altitude = 32.32 m

Temperature = 25.54 *C
Pressure = 100935.35 Pa
Approx altitude = 32.49 m

Temperature = 25.65 *C
Pressure = 100939.53 Pa
Approx altitude = 32.14 m

Temperature = 26.91 *C
Pressure = 101698.37 Pa
Approx altitude = -31.04 m

Temperature = 26.73 *C
Pressure = 100944.21 Pa
Approx altitude = 31.75 m
```

Autoscroll ✓ — Both NL & CR — 9600 baud

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