



## SmartElex BME680 - Temperature, Humidity, Pressure and Gas Sensor Breakout Board



The long awaited BME680 from Bosch gives you *all the environmental sensing you want* in one small package. This little sensor contains **temperature, humidity, barometric pressure** and **VOC gas** sensing capabilities. All over SPI or I2C, at a great price!

Like the BME280 & BMP280, this precision sensor from Bosch can measure humidity with  $\pm 3\%$  accuracy, 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 or better accuracy!

The BME680 takes those sensors to the next step in that it contains a small MOX sensor. The heated metal oxide changes resistance based on the volatile organic compounds (VOC) in the air, so it can be used to detect gasses & alcohols such as Ethanol, Alcohol and Carbon Monoxide and perform air quality measurements. Note it will give you one resistance value, with overall VOC content, it cannot differentiate gasses or alcohols.

**Please note, this sensor, like *all* VOC/gas sensors, has variability and to get precise measurements you will want to calibrate it against known sources!** That said, for general environmental sensors, it will give you a good idea of trends and comparisons. We recommend that you run this sensor for 48 hours when you first receive it to "burn it in", and then 30 minutes in the desired mode every time the sensor is in use. This is because the sensitivity levels of the sensor will change during early use and the resistance will slowly rise over time as the MOX warms up to its baseline reading.

For your convenience we've pick-and-placed the sensor on a PCB with a 3.3V regulator and some level shifting so it can be easily used with your favorite 3.3V or 5V microcontroller.

## 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 BME680 to your processor
- **SDI** - this is the **Serial Data In / Microcontroller Out Sensor In** pin, for data sent from your processor to the BME680
- **CS** - this is the **Chip Select** pin, drop it low to start an SPI transaction. Its an input to the chip

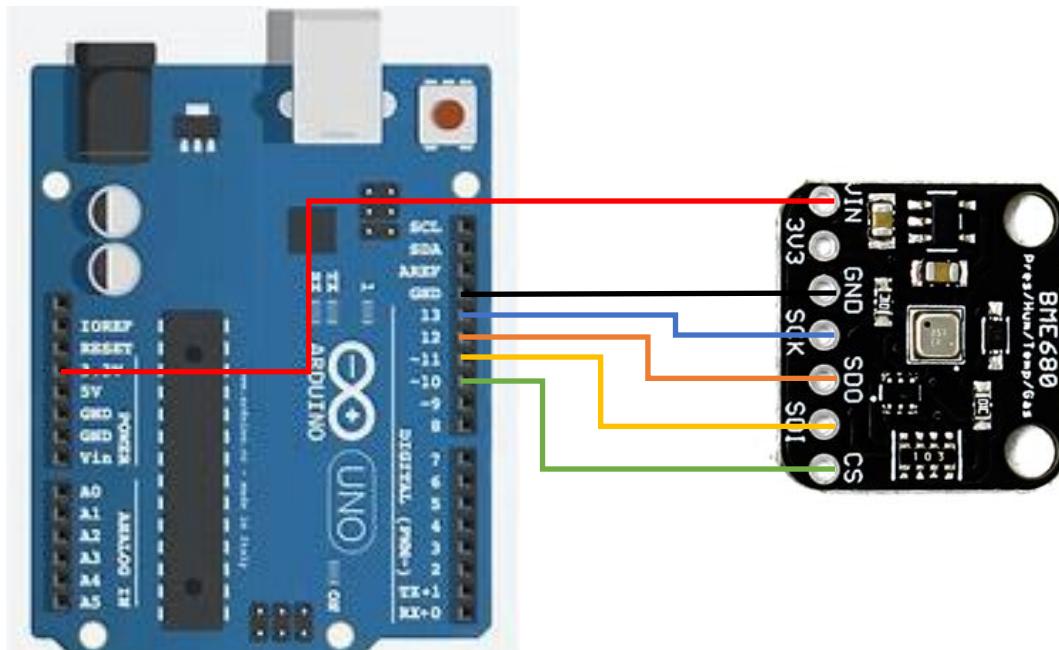
## Wiring

You can easily wire this breakout to any microcontroller, we'll be using an Arduino compatible. 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.

## SPI Wiring

Since this is a SPI-capable sensor, we can use hardware or 'software' SPI. To make wiring identical on all microcontrollers, we'll begin with 'software' SPI. The following pins should be used:



Arduino	BME680
D13	SCK
D12	SDO
D11	SDI
D10	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
- 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 others.

## Install Adafruit\_BME680 library

To begin reading sensor data, you will need to install the Adafruit\_BME680 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 bme680** to locate the library. Click **Install**

## Load Demo

Open up **File->Examples->Adafruit\_BME680->bmp680test** and upload to your microcontroller wired up to the sensor

```
#include <Wire.h>

#include <SPI.h>

#include <Adafruit_Sensor.h>

#include "Adafruit_BME680.h"

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

#define SEALEVELPRESSURE_HPA (1013.25)

// Adafruit_BME680 bme; // I2C
//Adafruit_BME680 bme(BME_CS); // hardware SPI
Adafruit_BME680 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);

void setup() {
    Serial.begin(9600);
```

```
while (!Serial);

Serial.println(F("BME680 test"));

if (!bme.begin()) {
    Serial.println("Could not find a valid BME680 sensor, check wiring!");
    while (1);
}

// Set up oversampling and filter initialization
bme.setTemperatureOversampling(BME680_OS_8X);
bme.setHumidityOversampling(BME680_OS_2X);
bme.setPressureOversampling(BME680_OS_4X);
bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
bme.setGasHeater(320, 150); // 320*C for 150 ms
}

void loop() {
    if (! bme.performReading()) {
        Serial.println("Failed to perform reading :(");
        return;
    }
    Serial.print("Temperature = ");
    Serial.print(bme.temperature);
    Serial.println(" *C");

    Serial.print("Pressure = ");
}
```

```
Serial.print(bme.pressure / 100.0);
Serial.println(" hPa");

Serial.print("Humidity = ");
Serial.print(bme.humidity);
Serial.println(" %");

Serial.print("Gas = ");
Serial.print(bme.gas_resistance / 1000.0);
Serial.println(" KOhms");

Serial.print("Approx. Altitude = ");
Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
Serial.println(" m");

Serial.println();
delay(2000);
}
```

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_BME680 bme; // I2C
//Adafruit_BME680 bme(BME_CS); // hardware SPI
//Adafruit_BME680 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);
```

Once uploaded, open up the serial console at 9600 baud speed to see data being printed out

```
BME680 test
Temperature = 22.97 *C
Pressure = 1023.02 hPa
Humidity = 34.57 %
Gas = 0.00 KOhms
Approx. Altitude = -79.71 m

Temperature = 23.06 *C
Pressure = 1022.88 hPa
Humidity = 34.60 %
Gas = 158.76 KOhms
Approx. Altitude = -77.56 m

Temperature = 23.18 *C
Pressure = 1022.68 hPa
Humidity = 34.51 %
Gas = 180.01 KOhms
Approx. Altitude = -76.32 m
```

Autoscroll      No line ending      9600 baud      Clear output

**Temperature** is calculated in degrees C, you can convert this to F by using the classic  $F = C \times 9/5 + 32$  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.

**Humidity** is returned in Relative Humidity %

**Gas** is returned as a resistance value in ohms. This value takes up to 30 minutes to stabilize! Once it stabilizes, you can use that as your baseline reading. Higher concentrations of VOC will make the resistance lower.

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.