

[Home](#)**Projects****Arduino**

[MAX7219 Dotmatrix](#)
[MCP23017 I/O Expander](#)
[BMP280 Pressure Sensor](#)
[DS18B20 Thermometer](#)
[Humidity DHT22/11](#)
[Rotary Encoder](#)
[MAX6675 0 to 1024 DegC](#)
[Battery Charger](#)
[SSD1306 OLED](#)
[74HC595 I/O Expander](#)
[Hitachi HD44780](#)
[Logic Level Converter](#)
[DigitalWrite](#)

Ebooks[Digital download eBooks](#)**PIC**

[LED Dot Matrix 8x8 Direct drive](#)
[Serial LCD](#)
[LCD Keypad](#)
[Frequency Counter LCD](#)
[Frequency Counter 7seg](#)
[DS1307 Clock](#)
[Binary Clock 8x8 matrix](#)
[Infrared Receiver](#)
[A PIC 12F675 tutorial.](#)
[LCD Volt Meter](#)
[PIC Sonar](#)
[PIC Projects](#)
[Project Ideas](#)

Interfaces

[SPI Interface](#)
[I2C Interface](#)

PIC Resources

[PIC Introduction](#)
[PIC ICSP](#)
[Programmer Types](#)

Become a subscriber (Free)

Join 29,000 other subscribers to receive subscriber sale discounts and other free resources.

Name:

E-Mail:

JOIN NOW

Don't worry -- your email address is totally secure. I promise to use it **only** to send you MicroZine.

How to use the BMP280 Barometric Pressure Chip

The BMP280 is an insanely small SMD device (2.0mm x 2.5mm) so you'll need to get hold of a breakout board to use it on the bench. It is small enough that it can be used within a mobile phone. That may seem a bit unusual as you would probably think of this device used only in a weather station environment until you realise that pressure can be used to measure altitude.

In fact the applications that are suggested by Bosch Sensortech (manufactures the BMP280) are:

- Enhancement of GPS navigation (e.g. time-to-first-fix improvement, dead-reckoning, slope detection).
- Indoor navigation (floor detection, elevator detection).
- Outdoor navigation, leisure and sports applications.
- Weather forecast.
- Health care applications (e.g. spirometry).
- Vertical velocity indication (e.g. rise/sink speed).

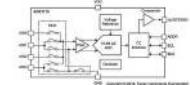
It is also ideally suited and for use as a highly accurate Arduino barometric pressure sensor since it employs an SPI interface.

 [report this ad](#)
[Like 2](#)

Visit our Facebook Page:

[To Visit Click Here](#)**Recent Articles**

[How to use the ADS1115](#)



A tutorial on using the ADS1115 precision 16 bit ADC for low power use.

[Read more](#)
[BACK](#)
[TOP](#)

Resources

Tips & Techniques
PIC Tutorials
Microcontroller Blog
Site map

Other

Articles
Books



Lcd Display Controllers

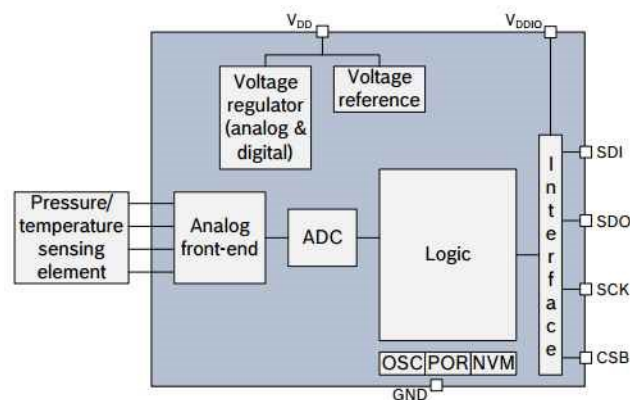
Over 500,000 Products Online

Order In The Next {PH_0}. Get It By {PH_1}. Track & Trace Delivery Available!

uk.rs-online.com

OPEN

[Source: https://www.bosch-sensortec.com/bst/products/all_products/bmp280]



Source: *datasheet Figure 1*



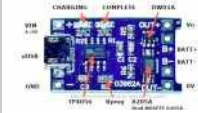
report this ad

Initially it would seem that measuring height using a BMP280 chip is a little redundant since a GPS system is far more accurate and also gives you height information as well. Now, I did a little research on Google and that last statement may not be so accurate!

We all tend to assume that any system we have is absolutely accurate and I thought GPS was in that category (based on the fact that my satnav shows me exactly where I am - but how accurate is that?). It turns out to be about $\pm 10\text{m}$!

Despite the fact that GPS satellites have atomic clocks on board and even after adjustment for relativistic changes due to speed of travel and distance above the earth - there are other factors that influence GPS accuracy including the atmosphere, satellite position inaccuracy and other factors as well - and these can add up to quite a large error.

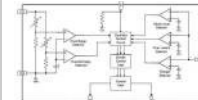
The TP4056: Lithium Ion/polymer Battery Charger IC



Learn how to use the TP4056 properly. There's a right, and a wrong way, to use it to safely charge Lithium Ion batteries.

[Read more](#)

DW01A Battery Protector IC



The DW01A chip is a Lithium Ion battery protector commonly used on TP4056 boards. Find out Exactly how it works and how to use it the correct way.

[Read more](#)

Arduino String: How to read commands from the serial port.

For Arduino string operations you can use Object Class Strings or C style strings but which should you use? Also find out how to decode commands and control variables in your programs using strings.

[Read more](#)

A Real Time Clock design (DS1307) with a PIC microcontroller

Real Time Clock Design (FREE): A Free and Complete RTC design using the DS1307 and a PIC micro (16F88) also re-targetable. This PIC project uses an I2C Clock chip and

BACK
TOP



Garmin GPS receivers are typically accurate to 10m [horizontal measurement]
[Source: Garmin.com/aboutGPS].

Did you know that the military did not believe that Einstein's theory of relativity would apply and built in a software switch to inside the satellite code to stop adjustment for that science (they had to turn it on when they found Enistein's relativity theorem did apply even to their satellites!).

Why a GPS height Measurement is not that accurate

Another website (see link below) discusses the problem more fully. In summary, it is a comparison of GPS vs Barometric Pressure measurement. In that article it discusses how hang gliders and paragliders must fly within a specific defined volumetric space so that they are not eliminated from the competition. To do that it is important that they can measure height accurately using their own instruments and you would have thought that the GPS system is ideal for that; Maybe not!

The thing about GPS is that the GPS receiver knows where the satellites are so it knows its position relative to 4 or more satellites with extreme accuracy (more satellites give increasing positional accuracy). The problem is that the orbit of the satellites does not mean that the satellite or GPS receiver knows where the ground is located!

""Your GPS unit has multiple models (datums) of the Earth in its memory."

[source <http://www.xcmag.com/2011/07/gps-versus-barometric-altitude-the-definitive-answer/>]

In fact, each GPS unit has multiple models of the earth's surface e.g. WGS 84 (World Geodetic System - created in 1984). Other datums can be more accurate for specific countries e.g. in the UK OSGB36 (ordnance survey data). There are quite a few other datum sets but WGS 84 is commonly used for GPS systems.

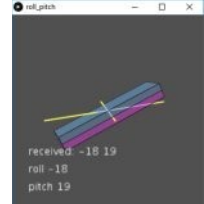
The WGS 84 system defines an oblate spheroid that approximates the shape of the Earth and this is where errors can occur depending on where you are located. In the UK you would be 70m above the true height, and in India 100m below! (See the link above for a heatmap of height variation around the globe). Depending on your GPS the error may be calculated and compensated out (geoid correction).

Additionally there are other horizontal error sources due to:

7-segment display to create a fou...

[Read more](#)

[How to use the ADXL345 for movement sensing and more.](#)



With the ADXL345 acellerometer you can detect up to 16g! You can also find out how to use it for tap detection and more.

[Read more](#)

[Readers Comments](#)

"I wanted to thank you so so so much for all the information you have provided in your site it's

SUPERB and **FANTASTIC**."

- Ranish Pottath

"This site really is the best and my favorite. I find here many useful projects and tips."

- Milan

bursach<at>gmail.com<

"Awesome site, very, very easy and nice to navigate!"

- Matt

matt_tr<at>
wolf359.cjb.net

[Learn Microcontrollers](#)

"Interested in Microcontrollers?"

Sign up for The Free 7 day guide:

[FREE GUIDE :](#)
[CLICK HERE](#)

BACK
TOP

"I am a newbie to PIC

*and I wanted to say
how great your
site has been for me."*

- Dave

de_scott<at>bellsouth.net

*"Your site is a great
and perfect work.
congratulations."*

- Suresh

integratredinfosys<at>
yahoo.com

*"I couldn't find the correct
words to define
yourweb site."*

*Very useful, uncovered,
honest and clear."*

*Thanks so much for
your time and works.
Regards."*

- Anon



- Ionospheric effect ($\pm 5\text{m}$)
- Satellite position error (± 2.5)
- Satellite clock error ($\pm 2\text{m}$)
- Multipath interference ($\pm 1\text{m}$)
- Tropospheric effect ($\pm 0.5\text{m}$)
- Maths and rounding errors ($\pm 1\text{m}$)

Total horizontal error: $\pm 15\text{m}$. It is estimated that the **vertical error** is 3 times this: $\pm 45\text{m}$.

Another error source is from averaging the information over time e.g. over a few seconds so there will be a delay in output reading while climbing or falling.

The section above is a summary from this link:

<http://www.xcmag.com/2011/07/gps-versus-barometric-altitude-the-definitive-answer/>

Using Barometric Pressure for Height Measurement

In fact all aviation uses a Barometric pressure measurement converted to show a height measurement, since pressure decreases with height. Factors that affect the reading are temperature and so altimeters are temperature compensated. Another factor is the ground level pressure measurement (defines zero height) which varies with the weather so altimeters allow setting of the ground level pressure reading (obtained from weather services for aviation) to eliminate local pressure changes. lookup QNH and QNE

With the Bosch 280 chip the pressure range is from 300 to 1100 hPa which means height can be measured from +9000m to -500m. It also has an relative accuracy of ± 0.12 hPa. This equates to a relative accuracy of $\pm 1\text{m}$.

 [report this ad](#)

BACK
TOP



Note: This is probably the maximum inaccuracy - see later for actual measurements that show the sensor (the specific one I am using) is far more sensitive to relative height change.

Even if this inaccuracy is correct across the range of batches of all BMP280's then it still represents a hugely more accurate reading than using a GPS for height measurement.

The absolute accuracy is $\pm 1\text{hPa}$ equivalent to 11.9m [$9500.0/(1100-300)$] - this is still quite a bit better than GPS vertical readings. To get a more accurate height reading the unit would have to be calibrated i.e. set the output reading to zero on the ground (or get the current ground level pressure reading for the area from an aviation service and input it to the microcontroller).

The BMP280 (all electronic barometers probably) can react very quickly to pressure changes - there is no waiting around. The datasheet indicates that you can read the device 157 times a second (for the BMP180 it is 120Hz). This means variations in height can be rapidly detected. This is a bit too fast for height measurements - you won't need that many data points - although they can be used when employing the IIR filter i.e. oversampling and then filtering to get a stable output. The rapid measurement rate would be more appropriate for spirometry (measurement of breathing) or for rapid height change measurement such as in a hang glider competition..

Difference between BMP180 and BMP280

The following table shows the **difference between the older BMP180 and the replacement part BMP280**. Note how the BMP280 has far better specifications and has multiple interfaces including high speed SPI, whereas the BMP180 only has I2C (you could have ordered a separate part with SPI). The other major difference is that the BMP280 has a selectable filter operation with 5 different bandwidths resulting in measurements with higher accuracy.

Parameter	BMP180	BMP280
Footprint	3.6 × 3.8 mm	2.0 × 2.5 mm
Minimum V _{DD}	1.80 V	1.71 V
Minimum V _{DDIO}	1.62 V	1.20 V
Current consumption @3 Pa RMS noise	12 µA	2.7 µA
RMS Noise	3 Pa	1.3 Pa
Pressure resolution	1 Pa	0.16 Pa
Temperature resolution	0.1°C	0.01°C
Interfaces	I ² C	I ² C & SPI (3 and 4 wire, mode '00' and '11')
Measurement modes	Only P or T, forced	P&T, forced or periodic
Measurement rate	up to 120 Hz	up to 157 Hz
Filter options	None	Five bandwidths

▶ ×

Note: Temperature accuracy data is (buried in the data sheet):

- ±0.5°C (at 25°C) and
- ±1.0°C (over the full temperature range 0°C ~ 65°C).

Pressure Measurement Units

Unit	Symbol	No. Pascals
bar	bar	1 × 10 ⁵ Pa (exactly)
millibar	mbar	100 Pa (exactly)
hectopascal	hPa	100 Pa (exactly)
conventional millimetre of mercury	mmHg	133.322... Pa
conventional inch of mercury	inHg	3386.39... Pa
pound-force per square inch	lbf/in ²	6894.76... Pa

[Source: <http://www.npl.co.uk/reference/faqs/pressure-units>]

Note: hPa is numerically identical to mbar (millibar).

Measuring Altitude

To measure altitude you have to know the pressure reading at sea level. The commonly used value is 101325 Pa or 1013.25mb - this was intended to represent the mean atmospheric pressure at sea level. Unfortunately the value is not representative of the pressure at sea level in many different countries.

BACK
TOP

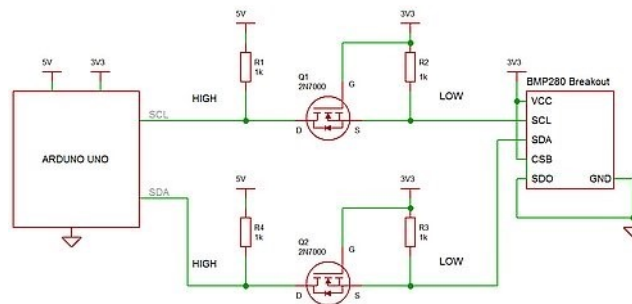
Really, the only way for accurate height measurement, is to obtain the pressure value for sea level on the day from a reliable forecast.

Note: Even the ground level value will change through out the day anyway, so you really need a radio link transmitting the current pressure on the ground to your measuring device. This would make the altitude measurement as accurate as possible even with local pressure variations.

Schematic diagram

Use the level translator parts if your breakout doesn't have them on-board (some boards come with level translator components). If it does have them then connect from the (5V) uno to the breakout board directly. Alternatively use a 3V3 arduino for direct connection.

Arduino Uno BMP280 wiring



The two [level converter circuits](#) above convert voltage levels to and from 5V and 3V3. You can find out ore about it [here](#).

Note: The I2C connections are the two top left ones on an Arduino Uno if the usb connector is at the left of the board. They are not marked. The left one is SCL and the right one is SDA. They are the two shielded pins opposite the 6 pin connector the next labelled pin is AREF.

Typical pressure Results through the day

These reading are taken from the serial monitor output from the Arduino serial port for both Adafruti BMP280 library and I2C library (I2C library 1st). Height is not calibrated = wrong (except where noted).

10.15 3/8/17 avg 3-4th aug 1007mbar sea level pressure forecast

Probe BMP280: Sensor found

HeightPT1: 7.41 m; Height: 81.49 Pressure: 100350.00 Pa; T: 22.97 C

HeightPT1: 14.10 m; Height: 80.99 Pressure: 100356.00 Pa; T: 23.03 C

HeightPT1: 80.79 m; Height: 80.74 Pressure: 100359.00 Pa; T: 23.14 C

16:37 3/8/17

Probe BMP280: Sensor found

HeightPT1: 6.81 m; Height: 74.95 Pressure: 100428.00 Pa; T: 23.43 C

HeightPT1: 13.01 m; Height: 74.95 Pressure: 100428.00 Pa; T: 23.50 C

HeightPT1: 18.62 m; Height: 74.78 Pressure: 100430.00 Pa; T: 23.55 C

17.06 3/8/17

HeightPT1: 73.82 m; Height: 73.61 Pressure: 100444.00 Pa; T: 24.29 C

HeightPT1: 73.82 m; Height: 73.86 Pressure: 100441.00 Pa; T: 24.29 C

HeightPT1: 73.80 m; Height: 73.61 Pressure: 100444.00 Pa; T: 24.29 C

17:13 3/8/17

HeightPT1: 72.57 m; Height: 72.77 Pressure: 100454.00 Pa; T: 24.50 C

HeightPT1: 72.60 m; Height: 72.94 Pressure: 100452.00 Pa; T: 24.50 C

HeightPT1: 72.63 m; Height: 72.85 Pressure: 100453.00 Pa; T: 24.50 C

HeightPT1: 72.65 m; Height: 72.85 Pressure: 100453.00 Pa; T: 24.51 C

22:18 3/8/17

Probe BMP280: Sensor found

HeightPT1: 5.31 m; Height: 58.44 Pressure: 100625.00 Pa; T: 27.80 C

HeightPT1: 10.11 m; Height: 58.11 Pressure: 100629.00 Pa; T: 27.85 C

HeightPT1: 14.51 m; Height: 58.53 Pressure: 100624.00 Pa; T: 27.89 C

MP280 test 17.06 3/8/17 Adafruit library output (when SDO pulled to 3V3)

Temperature = 24.61 *C

Pressure = 100444.17 Pa

Approx altitude = 73.60 m

3/8/17 Adafruit library output (when SDO pulled to 3V3)

Temperature = 24.99 *C

Pressure = 100462.96 Pa

Approx altitude = 72.02 m

3/8/17 Adafruit library output (when SDO pulled to 3V3)

Temperature = 25.03 *C

Pressure = 100458.71 Pa

Approx altitude = 72.38 m

07:09 4/8/17 forecast pressure 1009mbar avg sea level pressure

Probe BMP280: Sensor found

HeightPT1: 5.08 m; Height: 55.85 Pressure: 100656.00 Pa; T: 21.30 C

HeightPT1: 9.68 m; Height: 55.68 Pressure: 100658.00 Pa; T: 21.36 C

HeightPT1: 13.87 m; Height: 55.85 Pressure: 100656.00 Pa; T: 21.42 C

09:04 4/8/17

HeightPT1: 49.79 m; Height: 49.57 Pressure: 100731.00 Pa; T: 22.46 C

HeightPT1: 49.79 m; Height: 49.74 Pressure: 100729.00 Pa; T: 22.46 C

HeightPT1: 49.80 m; Height: 49.91 Pressure: 100727.00 Pa; T: 22.45 C

HeightPT1: 49.80 m; Height: 49.82 Pressure: 100728.00 Pa; T: 22.44 C

Different libraries give the same result

Here the adafruit library was given the actual forecast sea level pressure to get an altitude. I don't think it is too accurate because you have to have the actual ground level pressure value at your location. where I am is about 23m above sea level not 2m see the readings below!

The I2C Sensor library does not let you change the default value of 1013.25mb so you would need to go and write some code to allow accurate height measurement adjustment.

Different libraries Adafruit and I2C library (same result)

07:41 13/8/17 forecast pressure 1020mbar avg sea level pressure

I2C Sensor LIB: Probe BMP280: Sensor found

HeightPT1: 1.82 m; Height: 20.03 Pressure: 101758.00 Pa; T: 23.27 C

HeightPT1: 3.50 m; Height: 20.28 Pressure: 101755.00 Pa; T: 23.34 C

HeightPT1: 5.01 m; Height: 20.12 Pressure: 101757.00 Pa; T: 23.33 C

07:42 13/8/17 Adafruit library forecast pressure 1020mbar

Temperature = 23.70 *C

Pressure = 101755.11 Pa

Approx altitude = 20.27 m

Note: The I2C sensor library code averages the 1st 10 readings (see example 2 below) so the left hand values are not stable (until 10 values are gathered) the next measurement is the unaveraged height reading (20.03).

Interestingly there is a difference in temperature readings; Maybe one of the library code implementations is not right.

Warning: Getting a calibrated height using a predicted forecast pressure reading is not accurate as it will be slightly different at your location, so it is best to design code to store the value (when you are at ground level - by pressing a dedicated button, or gather the data on power up). Then use this value as the calibration value for measuring relative height from ground level.

[BACK](#)
[TOP](#)

Why your BMP280 won't start

If you're like me, and you have a breakout board without level shifters, and you create some using a few components (see the schematic on this page), to get I2C mode going, and then burn the example code into the Arduino uno r3. You then wonder why it's doing absolutely nothing!

As a typical engineer you give it a go and then - it's time to read the manual! - this is always a last resort unless there is a danger of blowing up a \$800 component! In which case read the manual first.

It turns out that the communication mode of the BMP280 is set on power up by the state of various control inputs.

One "gotcha" is that an OUTPUT in SPI mode turns into an INPUT in I2C mode!

SPI mode uses more pins to define its interface and one of these unused the outputs (in SPI mode) is used as an input in I2C mode!. If you have a breakout board then these control inputs are left floating, meaning its just not going to work unless you set them before power up.

Warning: Holding any interface pin high while V_{DDIO} is switched off will damage the chip due to current flow through protection diode. This could happen if you are designing a complex power management system where control pins are connected to the processor and power to the chip is controlled separately.

Conversely, if V_{DD} is off and V_{DDIO} is supplied, control pins at the chip are held at high-z (inactive).

I2C Mode Selection

To get [I2C mode](#) going the Chip Select pin (CS) must be held at the V_{DDIO} level as the BMP280 powers up.

If CS is held low during power up SPI mode is active.

I2C Address Selection

In addition to that the SDO pin (used as an output in SPI mode) is used as an input that selects one of two I2C addresses. This pin must be held low to get the lower address.

If you don't want to continuously change from high to low on SDO when changing which library you use just go into the Adafruit library (or I2C Sensor library) and change the BMP280_address from 0x77 to 0x76 (Adafruit library). The file is at:

```
C:\Users\<user
name>\Documents\Arduino\libraries\Adafruit_BMP280_Library\Adafruit_BMP280.h
```

I2C pullups

SDI is a bi directional open drain pin and must have a pullup attached.

There is no need for a pullup on SCK as it is input; In I2C mode the device is only ever a slave and does not do clock stretching. I tested this using a resistor drop down (5V to 3V) using 1k connected to a 1k5 then to ground. With the SCK input connected to the top of the 1k5 - it works fine.

Note: You can only leave out one pull up resistor if you use the BMP280 on its own since I2C requires pullups so that the bus can be released for other devices.

Note: The SDI and SCK pins are not true open drain pins and only SDI and SDO are implemented as bi-directional.

Table 29 in the datasheet (Pin description) shows what is going on:

Table 29: Pin description

Pin	Name	I/O Type	Description	Connect to		
				SPI 4W	SPI 3W	PC
1	GND	Supply	Ground	GND		
2	CSB	In	Chip select	CSB	CSB	V _{DDIO}
3	SDI	In/Out	Serial data input	SDI	SDI/SDO	SDA
4	SCK	In	Serial clock input	SCK	SCK	SCL
5	SDO	In/Out	Serial data output	SDO	DNC	GND for default address
6	V _{DDIO}	Supply	Digital interface supply	V _{DDIO}		
7	GND	Supply	Ground	GND		
8	V _{DD}	Supply	Analog supply	V _{DD}		

Adafruit Library just won't go

I found that the ADAAfruit BMP280 Library would just not work - the reason was that the SDO pin (used as an input in I2c mode) has been defaulted to 0x77 in software - that means the SDO pin needs to be pulled high (3V3 high) to work with this library (or change the default address in the library to 0x76).

I2C Address : For the BMP280

The device can be set to one of two I2C addresses, so you can have two separate pressure sensors on one I2C bus. The selection is made using the SDO pin as an input, and when low selects address 0x76, and when high selects address 0x77.

BMP280 Datasheet

[Click here to download the datasheet](#)

Software Library and versions

Arduino IDE Version

Version : 1.6.4

Bosch BMP280 sensor Libraries:

The libraries used are:

- Adafruit BMP280 Library 1.0.2
- I2C-Sensor-Lib Ver 0.8.2

...both can easily installed from the Arduino IDE.

If you do not see the library as an entry when you click the menus:

Sketch-->Include Library

Then select manage libraries :

Sketch-->Include Library -->Manage Libraries...

Search for and install <lib name> using the "Filter Your Search" form.

Code size results from libraries used:

For the simple example code used there is a large difference in code use as shown below.

The Adafruit BMP280 uses 1334 bytes more than I2C Sensor Lib - this is something to note if you are running out of code space.

Code use I2C-Sensor-Lib Library:

Sketch uses 8,830 bytes (27%) of program storage space. Maximum is 32,256 bytes.

Global variables use 563 bytes (27%) of dynamic memory, leaving 1,485 bytes for local variables.

BACK
TOP

Code use Adafruit BMP280 Library:
Sketch uses 10,164 bytes (31%) of program storage space. Maximum is 32,256 bytes.
Global variables use 563 bytes (27%) of dynamic memory, leaving 1,485 bytes for local variable

Differences between the libraries

The adafruit BMP280 library comes with a member function (or method) that allows entry of the sea level altitude and returns the altitude based on this value and your current barometric reading (See example 1 below). This is a sparse library and does not give any other methods to control the register settings within the device.

On the other hand I2C Sensor lib does have methods to set the internal parameters of the device but does not have a sea level altitude correction function. You really need a corrector adjustment to get more accurate altitude. The code below shows you how to update the library to allow entry of the sea level pressure reading.

Adding a method to the I2C Sensor lib

This requires a simple addition to the library file:

- C:\Users\<username>\Documents\Arduino\libraries\I2C-Sensor-Lib\Lib\src\i2c_BMP280.h

Find the top method

```
"void getAltitude(float& meter)"
```

...and add in the one shown below:

```
"void getAltitude(float& meter, float seaLevelmbar)"
```

```
/**< gives the number of meters above sea level */
void getAltitude(float& meter)
{
    uint32_t iPascal;
    getPressure(iPascal);

    meter = 44330.0*(1-pow(float(iPascal)/101325.0,1.0/5.255));
};

/**< gives the number of meters above sea level. JFM parameterised sea level*/
void getAltitude(float& meter, float seaLevelmbar)
{
    uint32_t iPascal;
    getPressure(iPascal);

    meter = 44330.0*(1-pow(float(iPascal)/(seaLevelmbar*100),1.0/5.255));
};
```

This just adds a public method that allows the sea level pressure (forecast) to be entered instead of the default 1013.25mbar value which gives a more accurate height reading.

Example 1 Adafruit Library : BMP280

The 1st arduino example sketch. This example uses the "Adafruit BMP280" Library.

Click in the code below to copy it to the clipboard.

```
/*
 * This is a library for the BMP280 humidity, temperature & pressure sensor
 *
 * Designed specifically to work with the Adafruit BMEP280 Breakout
 * ----> http://www.adafruit.com/products/2651
 */
```

BACK
TOP

These sensors use I2C or SPI to communicate, 2 or 4 pins are required to interface.

Adafruit invests time and resources providing this open source code, please support Adafruit and open-source hardware by purchasing products from Adafruit!

Example 2 - I2C Lib : BMP280

The 2nd arduino example sketch. This example uses the "I2C-Sensor-Lib" library.

Note: This example requires the additional method shown above for accurate altitude which is not included in the original bmp280 arduino library (instructions for adding it are [here](#)):

```
void getAltitude(float&meter, float seaLevelmbar).
```

```
#include <Wire.h>
#include "i2c.h"

#include "i2c_BMP280.h"
BMP280 bmp280;

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

    Serial.print("I2C Sensor LIB: Probe BMP280: ");
    if (bmp280.initialize()) Serial.println("Sensor found");
    else
    {
        Serial.println("Sensor missing");
        while (1) {}
    }

    // onetime-measure:
    bmp280.setEnabled(0);
```

Ideas for using Altitude Sensors

A table in the datasheet repeated below gives an indication of how to setup the device for different applications which range from indoor navigation (can't use the GPS here as there is no signal indoors- it would also require accelerometer and perhaps a gyroscope chip, for dead reckoning movement detection). Other usage ideas are: Drop detection. Elevator floor change detection, Weather monitoring (e.g. Arduino barometric pressure monitor : lowest power setting), Use in a mobile phone.

Use case	Mode	Over-sampling setting	osrs_p	osrs_t	IIR filter coeff. (see 3.3.3)	I _{DD} [µA] (see 3.7)	ODR [Hz] (see 3.8.2)	RMS Noise [cm] (see 3.5)
handheld device low-power (e.g. Android)	Normal	Ultra high resolution	×16	×2	4	247	10.0	4.0
handheld device dynamic (e.g. Android)	Normal	Standard resolution	×4	×1	16	577	83.3	2.4
Weather monitoring (lowest power)	Forced	Ultra low power	×1	×1	Off	0.14	1/60	26.4
Elevator / floor change detection	Normal	Standard resolution	×4	×1	4	50.9	7.3	6.4
Drop detection	Normal	Low power	×2	×1	Off	509	125	20.8
Indoor navigation	Normal	Ultra high resolution	×16	×2	16	650	26.3	1.6

BACK
TOP

Source: Datasheet Table 7: Recommends filter settings based on use cases.

Note: ODR is Typical Output Data Rate i.e. the maximum you can read the device for the given settings.

What's the IIR filter for?

The BMP280 chip has a built in **I**nfinite **I**mpulse **R**esponse filter (which is a type of DSP filter) that is useful for improving the output result from the sensor (or ignoring short term changes). For instance if using the sensor indoors, the sensor is so fast and sensitive that any small pressure change will be recorded e.g. the slamming of a door.

Using the IIR is a trade off between sensor response time and the need to ignore short term changes in pressure - outside you would not need to use the IIR filter since there won't be any doors slamming to cause sudden pressure changes (see Weather monitoring/Indoor Navigation in the table above (IIR=0)). For drop detection the situation is the opposite to indoor navigation use - you want to detect sudden pressure changes for instant height change detection.

For indoor navigation use you can setup the BMP280 so that small changes are ignored by setting appropriate values for the IIR filter - that would be the maximum value i.e. 16 - see table above (Indoor Navigation and hand held dynamic device).

Update rate vs oversampling

Oversampling is simply how many times the device reads a sensor and averages out the result and then allows you to read it. The more oversampling that you set up the more time it takes for a reading to become available.

The table below (table 14 - again from the datasheet) shows the relationship between a specific standby time (that you can control: t_{sb}) versus the oversampling setting (ovsr_p and ovsr_t). For the above settings table 7 shows that the oversampling setting is "Standard resolution" so you can set t_{sb} to 0.5 and read the device at 83.33Hz i.e. the table below tells you the maximum achievable output rate for the given settings.

Table 14: typical output data Rate (ODR) in normal mode [Hz]

Oversampling setting	$t_{standby}$ [ms]							
	0.5	62.5	125	250	500	1000	2000	4000
Ultra low power	166.67	14.71	7.66	3.91	1.98	0.99	0.50	0.25
Low power	125.00	14.29	7.55	3.88	1.97	0.99	0.50	0.25
Standard resolution	83.33	13.51	7.33	3.82	1.96	0.99	0.50	0.25
High resolution	50.00	12.20	6.92	3.71	1.92	0.98	0.50	0.25
Ultra high resolution	26.32	10.00	6.15	3.48	1.86	0.96	0.49	0.25

Example 3 Height Change Code

I have this odd idea that I want to find out the speed of the lift in the local sports center, and this device is the ideal tool to measure height changes and rate of height change. To do this you need to setup the BMP280 for elevator floor detection (see table 7 above) - actually drop detection provides more data and an ODR of 125Hz - you can average out the data by processing..

Library code that can be found at:

- C:\Users\<username>\Documents\Arduino\libraries\Adafruit_BMP280_Library
- C:\Users\<username>\Documents\Arduino\libraries\I2C-Sensor-Lib_iLib

Currently in the Adafruit Library software the Control register that sets the mode is fixed with no provision for re-programming it (within method: begin. Its value is set to 3F). For that reason the following code uses the I2C Sensor library as this has a few more functions that allow changing the mode of the BMP280 (see i2c_BMP280.h).

For elevator movement detection you need the following setup:

Drop detection			
Description	parameter	Value	I2C library method

Pressure oversampling	osrs_p	x2	setPressureOversampleRatio
Temperature oversampling	osrs_t	x1	setTemperatureOversampleRatio
IIR	IIR	0	setFilterRatio
Standby time	t_sb	0 = 0.5ms	setStandby

I selected a standby time as '0' for a fast update rate from the sensor. It may need to be adjusted for your code i.e. increase standby to reduce power consumption.

```
#include <Wire.h>
#include "i2c.h"

#include "i2c_BMP280.h"
BMP280 bmp280;

void setup()
{
    Serial.begin(115200);
    Serial.println("BMP280 elevator");
    Serial.print("Probe BMP280: ");
    if (bmp280.initialize()) Serial.println("Sensor found");
    else
    {
        Serial.println("Sensor missing");
        while (1) {}
    }

    // Setup for elevator floor change measurement.
    bmp280.setPressureOversampleRatio(2);
```

Now you can use the BMP280 for more than just weather monitoring!

This is the typical output you get when lifting or lowering the sensor (using the serial monitor in the IDE).

Note: The 76 value is as fast as you can go round the loop. In theory the sample rate should be set to 8 since the max you should go at is 125Hz so the sample rate should be 1/125=8ms. Then the repeat data rate in 1 sec would be 125 but the code is only capable of going at 76 in a sec. This will be due to using floats and serial output - think about using fixed point and it will go faster. The question is do you need it to go faster = design decision (the library would have to be re-written as well!).

You can see that increases in height result in positive values while decreases in height result in negative values and anything below 0.2m is ignored. The output below is obtained just from moving the sensor ~30cm up and down at a desk so it is very sensitive and ideal for height change detection (building navigation).

```
BMP280 elevator
Probe BMP280: Sensor found
Velocity: -3812.96 m/s ;debug: 73
Velocity: -0.31 m/s ;debug: 76
debug: 76
debug: 76
debug: 76
debug: 76
Velocity: 0.32 m/s ;debug: 76
Velocity: 0.24 m/s ;debug: 76
debug: 76
debug: 76
debug: 76
debug: 76
Velocity: 0.54 m/s ;debug: 76
debug: 76
Velocity: -0.47 m/s ;debug: 76
debug: 76
debug: 76
Velocity: 0.35 m/s ;debug: 76
debug: 76
Velocity: -0.20 m/s ;debug: 76
Velocity: -0.28 m/s ;debug: 76
```

```
Velocity: 0.27 m/s ;debug: 76
debug: 76
debug: 76
Velocity: -0.45 m/s ;debug: 76
debug: 76
debug: 76
debug: 76
```

The above code is only a first go at obtaining velocity. You should change it to get a reading every second using either "indoor navigation" or "handheld device" BMP280 modes and stop using the complete data set for averaging i.e. rely on the internal modes of the BMP280. Then display velocity based on height changes from one sample to the next at 1 sec intervals. In a handheld device you would want to save power and also do other tasks! Probably you would need a sample rate greater than 1 second so as not to miss slow changes;.

Other ideas for use of Altitude Sensors : BMP280

The BMP280 could be used in a drone for measuring height e.g. for building mapping (or just to figure out how high something is - given that the device seems fairly accurate (approx. 30cm). The stated relative accuracy seems a bit pessimistic at $\pm 0.12\text{hPa}$ ($\pm 0.12\text{mbar}$) for a $\pm 1\text{m}$ resolution and in practice seems a lot better than this (although where you need to claim accuracy you would have to state that the measurement is $\pm 1\text{m}$).

The results in the Serial Monitor output (above) show that the device is capable of detecting a 30cm change in height (for this specific device).


Show Index

Like 2 people like this. Be the first of your friends.


New! Comments

Have your say about what you just read! Leave me a comment in the box below.


3 Comments Sort by Oldest



Add a comment...



Tom Dawson
Hi John, A great article.
My question...
You mention the use of the module to determine altitude. To do this, knowledge of QNH is needed. If this is obtained from a local airfield for example, how would you then set QNH. For example, if you knew that QNH was 1000hPa how would you set that figure and how would you adjust it on a daily or more frequent basis? Cheers, Tom
[Like](#) · [Reply](#) · 2y



John Main
just to remind myself
QNH pressure above sea level (Nil Height).
QFE pressure at airport altitude (Field Elevation).

To do it you would

BACK
TOP

[Privacy Policy](#) | [Contact](#) | [About Me](#)

[Site Map](#) | [Terms of Use](#)

[BACK](#)
[TOP](#)

Copyright © 2005- 2019 Best-Microcontroller-Projects
All Rights Reserved.
No reproduction without permission.