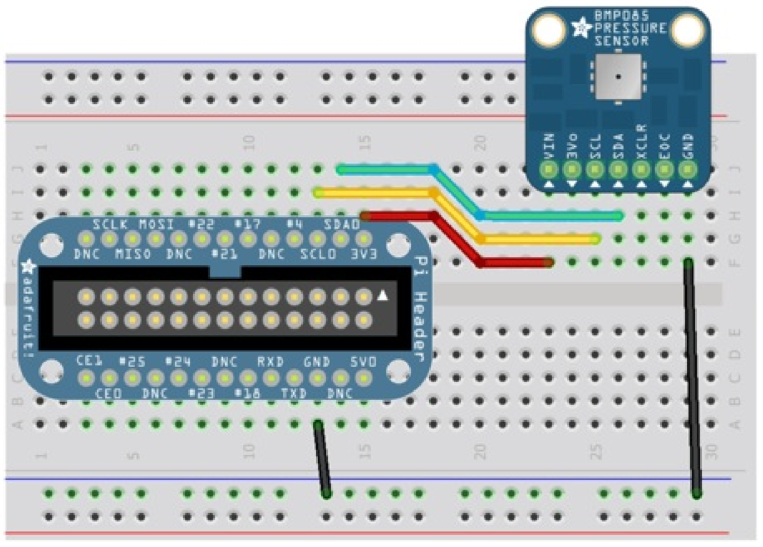
adafruit learning system

# Using the BMP085/180 with Raspberry Pi or Beaglebone Black

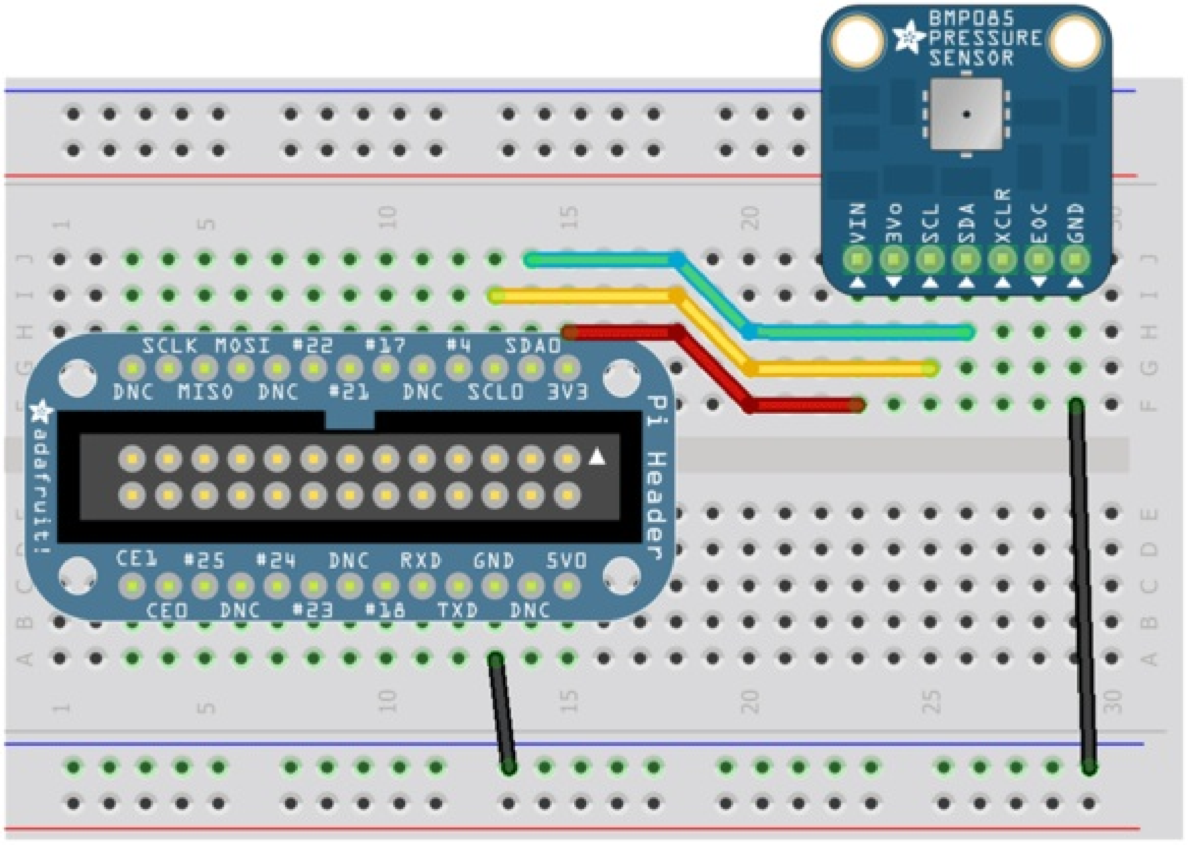
Created by Kevin Townsend



Last updated on 2015-08-16 07:10:13 PM EDT

## Guide Contents

# Overview



The Raspberry Pi and Beaglebone Black include support for Python, which makes it easy to get access to a lot of low-level hardware and software stacks -- USB, TCP/IP, multiple file systems etc. This is a good thing since it means you don't need to wrap your head around all the obscure details that go along with these complex stacks or the implementation details of various serial buses: you can focus on getting your data off your sensor and into your project as quickly as possible. Hurray for abstraction!

Most sensors tend to communicate with other devices based on one of three well-defined mechanisms: **I2C**, **SPI** or good old **analog output**. There are dozens of other serial buses and communication protocols out there (CAN, 1-Wire, etc.), and they all have their strengths and weaknesses, but I2C, SPI and analog cover the overwhelming majority of sensors you're likely to hook up to your development board.

I2C is a particularly useful bus with the for two main reasons:

- It only requires two shared lines: **SCL** for the clock signal, and **SDA** for the bi-direction data transfers.

- Each I2C device uses a unique 7-bit address, meaning you can have more than 120 unique I2C devices sharing the bus, and you can freely communicate with them one at a time on an as-needed basis.

This tutorial will show you how you can read data from the I2C-based BMP085 or BMP180 Barometric Pressure Sensor using Python on a Raspberry Pi or Beaglebone Black.

## A Note on Distributions

Please note for the Raspberry Pi that this tutorial is based on Occidentalis (http://adafru.it/aNv), Adafruit's own educational Linux distro for Pi. It should work just as well with the latest Wheezy distro, etc., but it hasn't yet been tested on anything else.

For the Beaglebone Black this tutorial is based on the Debian distribution (http://adafru.it/dvh) that's shipping with recent Beaglebone Black boards. If you're using an older Beaglebone Black with the Angstrom distribution it's highly recommended that you grab a micro SD card and load it with Debian!

## Configuring the Pi for I2C

**If you're using a Raspberry Pi, follow the steps below to configure it to use the I2C interface. If you're using a Beaglebone Black with its standard Debian distribution, you can skip this page and move on to the next step.**

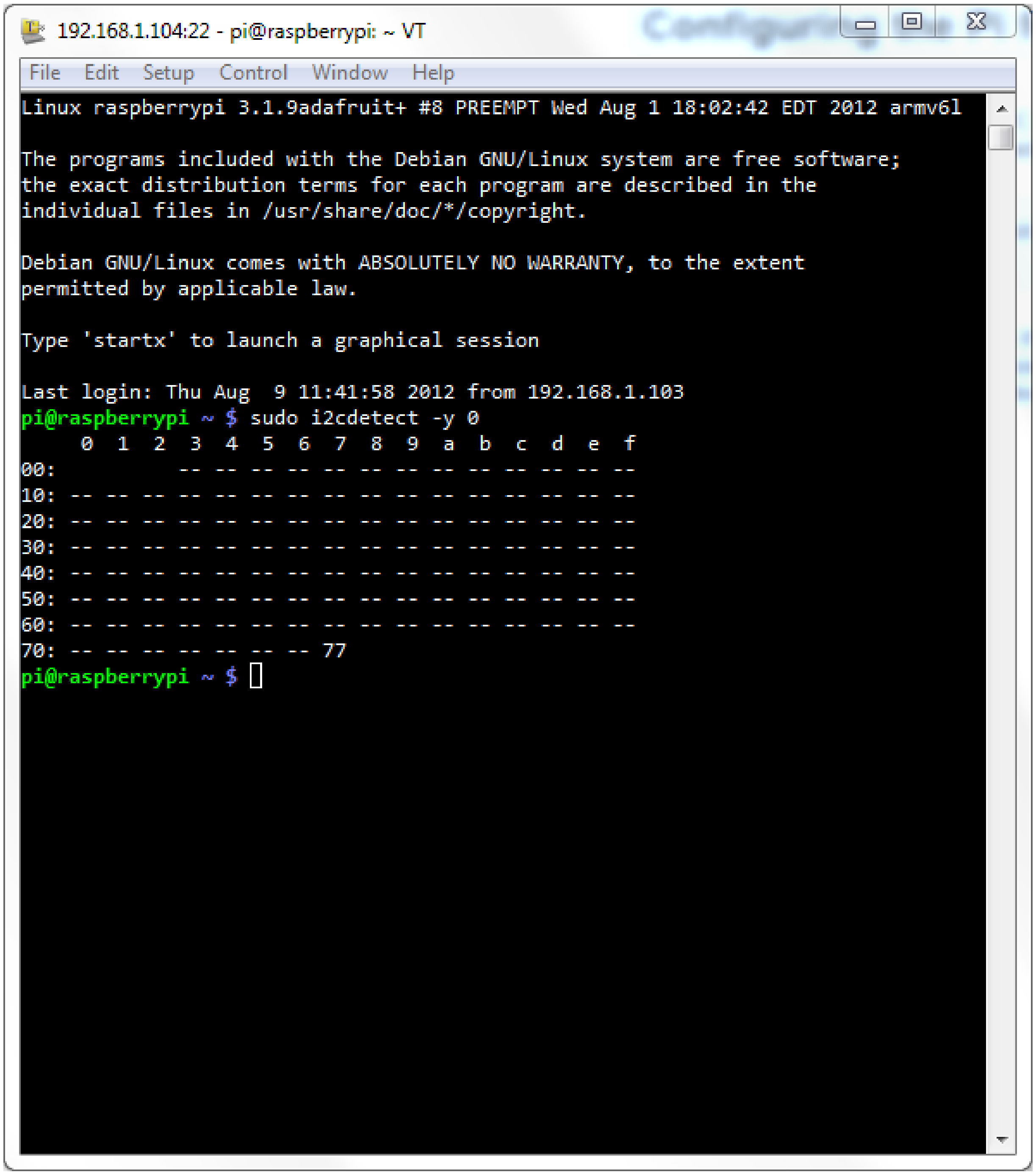
Before you can get started with I2C on the Pi, you'll need to run through a couple quick steps from the console.
Check out this tutorial for more details and follow it completely

http://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c (http://adafru.it/aTI)

When you're done, run

```
sudo i2cdetect -y 0 (if you are using a version 1 Raspberry Pi)
sudo i2cdetect -y 1 (if you are using a version 2 Raspberry Pi)
```

This will search /dev/i2c-0 or /dev/i2c-1 for all address, and if an Adafruit BMP085 Breakout is properly connected it should show up at 0x77 as follows:


192.168.1.104:22 - pi@raspberrypi: ~ VT
File Edit Setup Control Window Help

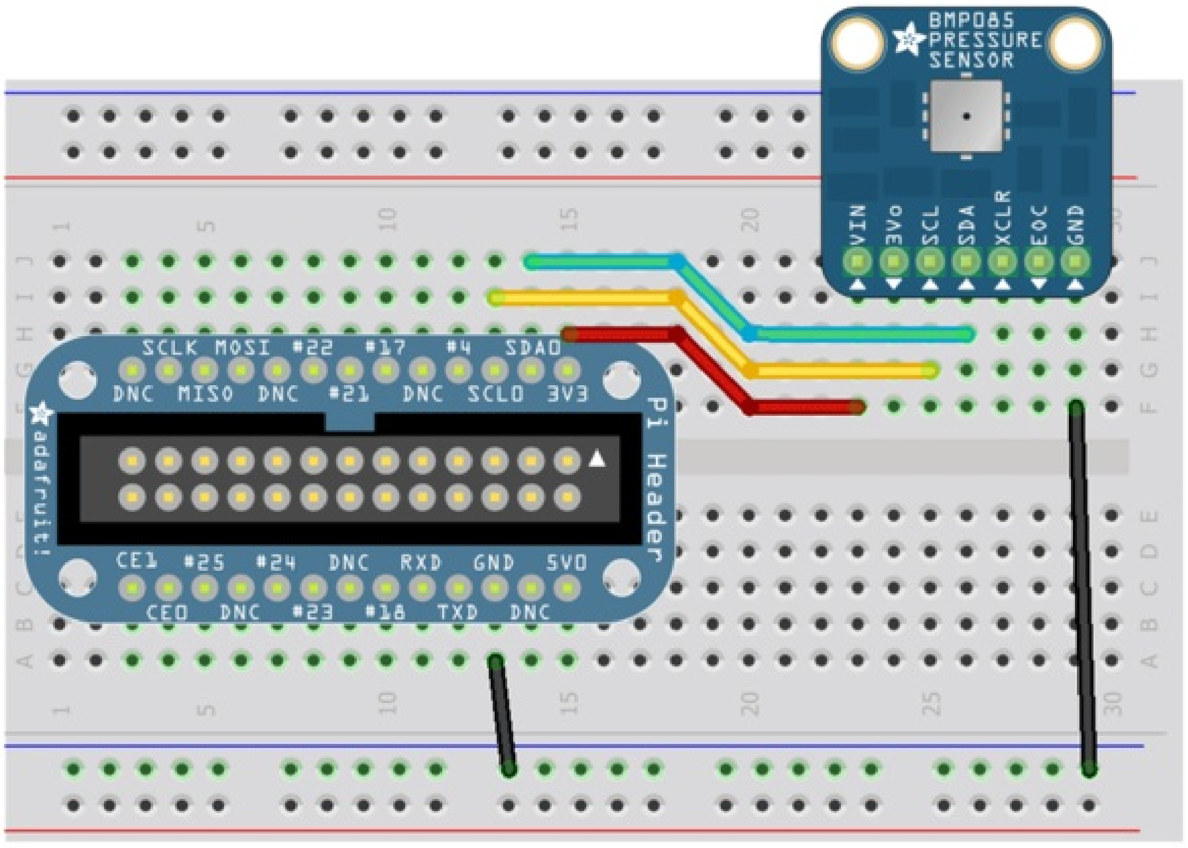
```
Linux raspberrypi 3.1.9adafruit+ #8 PREEMPT Wed Aug 1 18:02:42 EDT 2012 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Type 'startx' to launch a graphical session

Last login: Thu Aug  9 11:41:58 2012 from 192.168.1.103
pi@raspberrypi ~ $ sudo i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
pi@raspberrypi ~ $
```

Once both of these packages have been installed, you have everything you need to get started accessing I2C and SMBus devices in Python.

## Hooking Everything Up

### Raspberry Pi

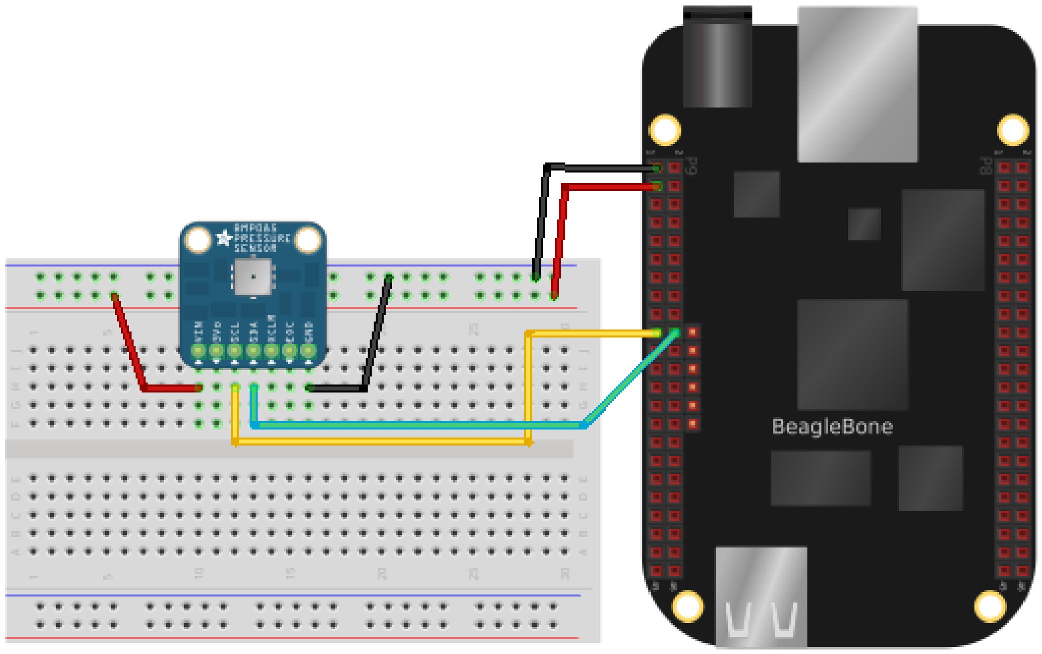
To hook your Adafruit BMP085/BMP180 Breakout up to the Pi, you can use a Pi Cobbler as seen in the following wiring diagram:



Make sure that you connect the VIN pin on the BMP sensor to 3V3, NOT 5V0! Connecting VIN to the 5V supply will cause the board to use 5V logic, which is perfect for the Arduino, but may damage the sensitive 3.3V inputs on the Raspberry Pi.

### Beaglebone Black

To connect the BMP sensor to a Beaglebone Black you can use any of the available I2C buses (http://adafru.it/dvk), but the library will default to using I2C bus 1 with pin P9_19 as SCL and P9_20 as SDA as shown below:



Make sure you aren't using any device tree overlays (http://adafru.it/dp6) which might interfere with the I2C pins. By default the standard device tree setup of the Debian distribution should expose the I2C bus 1 connections above without any extra work required.

Make sure you connect the VIN pin of the BMP sensor to the Beaglebone Black's 3.3 volt power rail, NOT the 5 volt power rail!

# Using the Adafruit BMP Python Library (Updated)

Using the BMP sensor with a Raspberry Pi or Beaglebone Black is easy with the Adafruit Python BMP sensor library (http://adafru.it/dCL). First make sure your device is powered on and has access to the internet (through a wired or wireless connection). Then connect to your device in a terminal and navigate to a directory where you want to download the library (like /home/pi on a Raspberry Pi or /root on a Beaglebone Black). Finally execute the following commands to download dependencies and install the library:

```
sudo apt-get update
sudo apt-get install git build-essential python-dev python-smbus
git clone https://github.com/adafruit/Adafruit_Python_BMP.git
cd Adafruit_Python_BMP
sudo python setup.py install
```

If you already have git or python-smbus installed you can ignore the message about the package already being installed.

Once the library is installed it will be accessible to any Python script on your device. You can see a few example scripts included in the library source's **examples** folder. Try running the **simpletest.py** example which grabs a single reading from the BMP sensor and displays it by executing:

```
cd examples
sudo python simpletest.py
```

If you receive an error message, carefully check that the library was installed correctly in the previous steps and try again. Note that the command needs to be run as root with sudo so that it can access the hardware's I2C bus.

After running the script you should see an output such as:

```
Temp = 20.20 *C
Pressure = 101667.00 Pa
Altitude = -28.27 m
Sealevel Pressure = 101665.00 Pa
```

Open the **simpletest.py** code in a text editor to see how to use the library to read the BMP sensor. First the library is imported with this command:

```
import Adafruit_BMP.BMP085 as BMP085
```

Next a BMP085 sensor instance is created with this command:

```
# Default constructor will pick a default I2C bus.
#
# For the Raspberry Pi this means you should hook up to the only exposed I2C bus
# from the main GPIO header and the library will figure out the bus number based
# on the Pi's revision.
#
# For the Beaglebone Black the library will assume bus 1 by default, which is
# exposed with SCL = P9_19 and SDA = P9_20.
sensor = BMP085.BMP085()

# Optionally you can override the bus number:
#sensor = BMP085.BMP085(busnum=2)

# You can also optionally change the BMP085 mode to one of BMP085_ULTRALOWPOWER,
# BMP085_STANDARD, BMP085_HIGHRES, or BMP085_ULTRAHIGHRES.  See the BMP085
# datasheet for more details on the meanings of each mode (accuracy and power
# consumption are primarily the differences).  The default mode is STANDARD.
#sensor = BMP085.BMP085(mode=BMP085.BMP085_ULTRAHIGHRES)
```

You can see from the comments there are a few ways to create the sensor instance. By default if you pass no parameters the library will try to find the right I2C bus for your device. For a Raspberry Pi the library will detect the revision number and use the appropriate bus (0 or 1). For a Beaglebone Black there are multiple I2C buses so the library defaults to bus 1, which is exposed with pin P9_19 as SCL clock and P9_20 as SDA data. You can explicitly set the bus number by passing it in the busnum parameter.

**Note if you're using a BeagleBone Black with the Ubuntu operating system you might need to change busnum to 2 to use the P9_19 & P9_20 pin I2C connection.** Just change the line to look like: **sensor = BMP.BMP085(busnum=2)**

The library will also choose by default to use the BMP sensor's standard operation mode. You can override this by passing a mode parameter with an explicit mode value--check the BMP datasheet (http://adafru.it/aKE) for more information on its modes.

Once the BMP sensor instance is created, you can read its values by calling the **read_temperature**, **read_pressure**, **read_altitude**, and **read_sealevel_pressure** functions like below:

```
print 'Temp = {0:0.2f} *C'.format(sensor.read_temperature())
print 'Pressure = {0:0.2f} Pa'.format(sensor.read_pressure())
print 'Altitude = {0:0.2f} m'.format(sensor.read_altitude())
print 'Sealevel Pressure = {0:0.2f} Pa'.format(sensor.read_sealevel_pressure())
```

That's all you need to do to read BMP sensor values using the Adafruit Python BMP library!

For another example of using the BMP library, check out the **google_spreadsheet.py** example. This code is similar to the DHT sensor Google Docs spreadsheet logging code (http://adafru.it/dCM), but is modified to use the BMP sensor and write the temperature, pressure, and altitude to a Google Docs spreadsheet. Check out the page on configuring Google Docs (http://adafru.it/dCN) to see more details on how to create the spreadsheet and configure the username, password, and spreadsheet name.

## Using the Adafruit BMP085 Python Library

Note this page shows how to use an older version of the BMP Python code and is only for historical purposes!

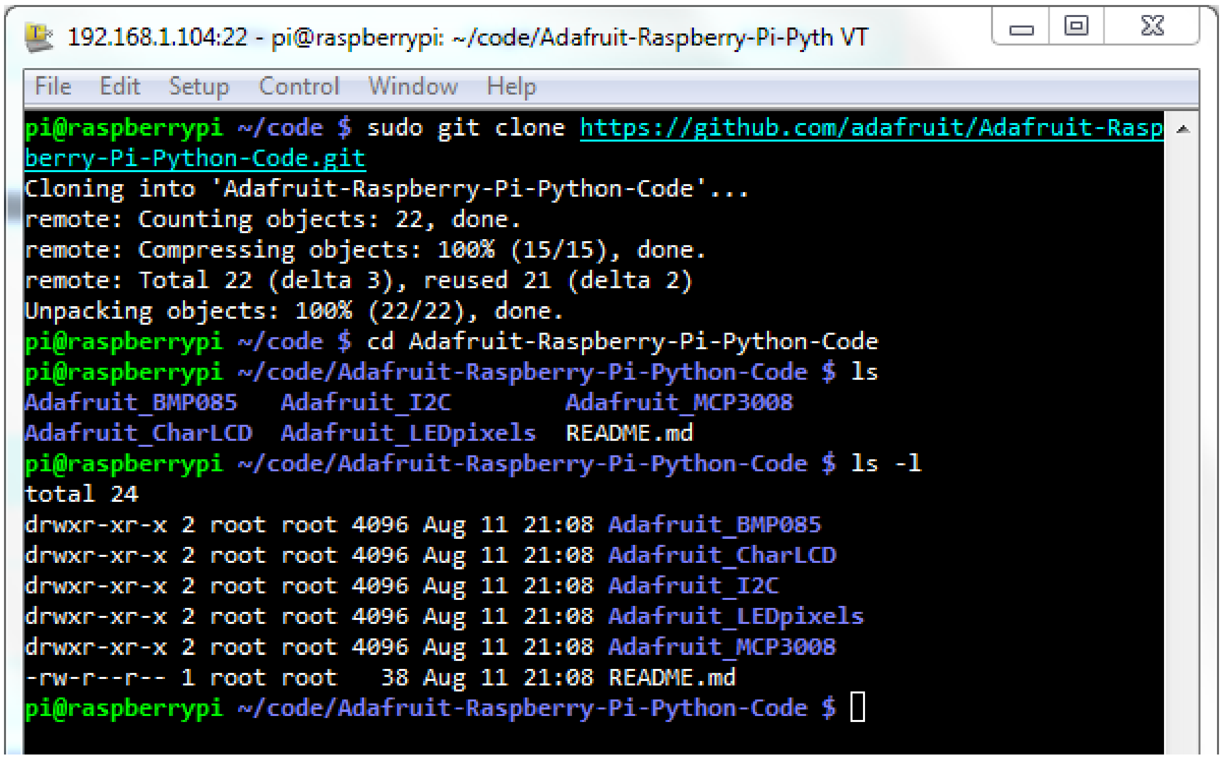
The BMP085 Python code for Pi is available on Github at https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code (http://adafru.it/aOg)

While many of these drivers and classes are works in progress -- we're still trying to figure out how we can make accessing HW as painless as possible on the Pi -- the current code should serve as a good starting point to understanding how you can access SMBus/I2C devices with your Pi, and getting some basic data out of your BMP085.

## Downloading the Code from Github

The easiest way to get the code onto your Pi is to hook up an Ethernet cable, and clone it directly using 'git', which is installed by default on most distros.  Simply run the following commands from an appropriate location (ex. "/home/pi"):

```
$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
$ cd Adafruit-Raspberry-Pi-Python-Code
$ cd Adafruit_BMP085
```


192.168.1.104:22 - pi@raspberrypi: ~/code/Adafruit-Raspberry-Pi-Pyth VT
File Edit Setup Control Window Help

```
pi@raspberrypi ~/code $ sudo git clone https://github.com/adafruit/Adafruit-Rasp
berry-Pi-Python-Code.git
Cloning into 'Adafruit-Raspberry-Pi-Python-Code'...
remote: Counting objects: 22, done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 22 (delta 3), reused 21 (delta 2)
Unpacking objects: 100% (22/22), done.
pi@raspberrypi ~/code $ cd Adafruit-Raspberry-Pi-Python-Code
pi@raspberrypi ~/code/Adafruit-Raspberry-Pi-Python-Code $ ls
Adafruit_BMP085    Adafruit_I2C        Adafruit_MCP3008
Adafruit_CharLCD   Adafruit_LEDpixels  README.md
pi@raspberrypi ~/code/Adafruit-Raspberry-Pi-Python-Code $ ls -l
total 24
drwxr-xr-x 2 root root 4096 Aug 11 21:08 Adafruit_BMP085
drwxr-xr-x 2 root root 4096 Aug 11 21:08 Adafruit_CharLCD
drwxr-xr-x 2 root root 4096 Aug 11 21:08 Adafruit_I2C
drwxr-xr-x 2 root root 4096 Aug 11 21:08 Adafruit_LEDpixels
drwxr-xr-x 2 root root 4096 Aug 11 21:08 Adafruit_MCP3008
-rw-r--r-- 1 root root   38 Aug 11 21:08 README.md
pi@raspberrypi ~/code/Adafruit-Raspberry-Pi-Python-Code $ 
```

## Testing the Library

If you're using a version 2 Pi (512 M) then you'll have to change the I2C bus as it flipped from #0 to #1 in the version 2.
Edit Adafruit_I2C.py with **nano Adafruit_I2C.py** and change this line

> **def __init__(self, address, bus=smbus.SMBus(0), debug=False):**

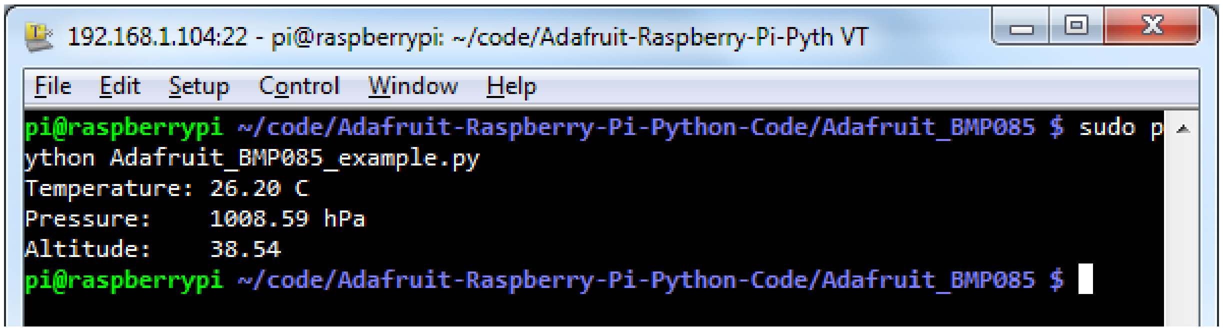
to

> **def __init__(self, address, bus=smbus.SMBus(1), debug=False)**

Once the code has be downloaded to an appropriate folder, and you have your BMP085 properly connected, you can start reading some data via the following command (the driver includes a simple demo program):

```
sudo python Adafruit_BMP085_example.py
```

Which should give you something similar to the following:


192.168.1.104:22 - pi@raspberrypi: ~/code/Adafruit-Raspberry-Pi-Pyth VT
File Edit Setup Control Window Help

```
pi@raspberrypi ~/code/Adafruit-Raspberry-Pi-Python-Code/Adafruit_BMP085 $ sudo p
ython Adafruit_BMP085_example.py
Temperature: 26.20 C
Pressure:    1008.59 hPa
Altitude:    38.54
pi@raspberrypi ~/code/Adafruit-Raspberry-Pi-Python-Code/Adafruit_BMP085 $
```

## Modifying the Code

The BMP085 library is organized as two seperate classes.  There is one class to handle the low-level SMBus/I2C calls (Adafruit_I2C), and another class that handles the BMP085-specific functionality.

The library includes the basic example shown above, but you can also customize the code a bit to provide full debug output if you're having any problems, change the address, or use the BMP085 in one of it's four different modes (ULTRALOWPOWER, STANDARD, HIRES, and ULTRAHIRES), as seen in the commented out initializors in the sample code below:

```python
#!/usr/bin/python

from Adafruit_BMP085 import BMP085

# ===========================================================================
# Example Code
# ===========================================================================

# Initialise the BMP085 and use STANDARD mode (default value)
# bmp = BMP085(0x77, debug=True)
bmp = BMP085(0x77)

# To specify a different operating mode, uncomment one of the following:
# bmp = BMP085(0x77, 0)  # ULTRALOWPOWER Mode
# bmp = BMP085(0x77, 1)  # STANDARD Mode
# bmp = BMP085(0x77, 2)  # HIRES Mode
# bmp = BMP085(0x77, 3)  # ULTRAHIRES Mode

temp = bmp.readTemperature()
pressure = bmp.readPressure()
altitude = bmp.readAltitude()

print "Temperature: %.2f C" % temp
print "Pressure:    %.2f hPa" % (pressure / 100.0)
print "Altitude:    %.2f" % altitude
```

## FAQs

### Can I use multiple BMP sensors on the same board?

Because each I2C device on the bus needs to have it's own unique address, you normally can only have one device at address 0x77 (etc.). If you require several I2C devices at the same address, and if the devices have a reset pin (like the BMP085 does), then you CAN use multiple devices at the same address ... but at the expense of one GPIO pin per device. What you can do is hold the other devices in reset by pulling the XCLR (Reset) pin low, and letting XCLR go high on the one device that you do want to read, releasing it from reset and causing it to respond to any request on the I2C bus.

Note that on the Beaglebone Black there are 2 I2C buses, so you can in theory run one sensor on each bus.