Bitify

Tinkering with the Raspberry Pi and other geeky stuff

Sunday, 24 November 2013

Interfacing a BMP085 Digital Pressure sensor to the Raspberry Pi

I recently bought a sensor with a BMP085 Digital Pressure sensor on it so I thought I'd write a post on how to read the data from the Raspberry Pi in Python over I2C.

Python code

Below is simple test code to initialise the sensor and then continuously loop around reading the temperature and air pressure.

```
#!/usr/bin/python
import smbus
import time

bus = smbus.SMBus(0) # or bus = smbus.SMBus(1) if you have a revision 2 board
address = 0x77

def read_byte(adr):
    return bus.read_byte_data(address, adr)

def read_word(adr):
    high = bus.read_byte_data(address, adr)
    low = bus.read_byte_data(address, adr+1)
    val = (high << 8) + low
    return val

def read_word_2c(adr):
    val = read_word(adr)
    if (val >= 0x8000):
        return -((0xffff - val) + 1)
    else:
        return val

def write_byte(adr, value):
    bus.write_byte_data(address, adr, value)

def twos_compliment(val):
    if (val >= 0x8000):
        return -((0xffff - val) + 1)
    else:
        return val

def get_word(array, index, twos):
    val = (array[index] << 8) + array[index+1]
    if twos:
        return twos_compliment(val)
    else:
        return val

def calculate():
    # This code is a direct translation from the datasheet
    # and should be optimised for real world use

    #Calculate temperature
    x1 = ((temp_raw - ac6) * ac5) / 32768
    x2 = (mc * 2048) / (x1 + md)
    b5 = x1 + x2
    t = (b5 + 8) / 16

    # Now calculate the pressure
    b6 = b5 - 4000
    x1 = (b2 * (b6 * b6 >> 12)) >> 11
    x2 = ac2 * b6 >> 11
    x3 = x1 + x2
    b3 = (((ac1 * 4 + x3) << oss) + 2) >> 2

    x1 = (ac3 * b6) >> 13
    x2 = (b1 * (b6 * b6 >> 12)) >> 16
    x3 = ((x1 + x2) + 2) >> 2
    b4 = ac4 * (x3 + 32768) >> 15
    b7 = (pressure_raw - b3) * (50000 >> oss)
    if (b7 < 0x80000000):
        p = (b7 * 2) /b4
    else:
        p = (b7 / b4) *2
    x1 = (p >> 8) * (p >> 8)
    x1 = (x1 * 3038) >> 16
    x2 = (-7357 * p) >> 16
    p = p + ((x1 + x2 + 3791) >> 4)
    return(t,p)

calibration = bus.read_i2c_block_data(address, 0xAA, 22)

oss = 3                 # Ultra high resolution
temp_wait_period = 0.004
pressure_wait_period = 0.0255 # Conversion time
```

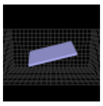
Search

Search

Popular Posts

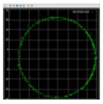
Interfacing Raspberry Pi and MPU-6050

I wanted to interface my Pi to a Six-Axis Gyro + Accelerometer sensor and the one I settled on was based on a MPU-6050 chip. I went for thi...



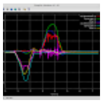
Reading data from the MPU-6050 on the Raspberry Pi

In a previous post I showed how to connect an Accelerometer & Gyro sensor to the Raspberry Pi, in this post I'll show some simple P...



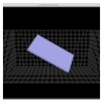
Connecting and calibrating a HMC5883L Compass on the Raspberry Pi

Here is how to connect a HMC5883L Compass to the Raspberry Pi, calibrate it and read the data. Connecting the compass is simple enough, fo...



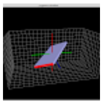
Using a complementary filter to combine Accelerometer and Gyroscopic data

This post shows how to combine data from the accelerometer and gyroscope using a complementary filter to produce a better readings from the...



3D OpenGL visualisation of the data from an MPU-6050 connected to a Raspberry Pi

In this post I'll show how to serve the data over http and display a 3D representation in OpenGL extending on a previous blog post det...



Pitch, Roll and Yaw using MPU6050 & HMC5883L (with tilt compensation and complementary filter)

Combining the data from an MPU605 and a HMC5883L to give tilt compensated pitch, roll and yaw. Pitch, roll and yaw (with tilt compensati...

GY80 (L3G4200D, ADXL345, HMC5883L, BMP085) Python library for Raspberry Pi

A while back I bought a GY80 board, which comprises of: L3G4200D - Three axis Gyroscope ADXL345 - Three axis accelerometer HMC5883L - C...

Temperature logging with a DS18B20 and a Raspberry Pi

I wanted to do some temperature logging so I hooked up a DS18B20 temperature sensor to a Raspberry Pi. About the DS18B20 Dallas DS18B...

```
077
078 # The sensor has a block of factory set calibration values we need to read
079 # these are then used in a length calculation to get the temperature and pressure
080 ac1 = get_word(calibration, 0, True)
081 ac2 = get_word(calibration, 2, True)
082 ac3 = get_word(calibration, 4, True)
083 ac4 = get_word(calibration, 6, False)
084 ac5 = get_word(calibration, 8, False)
085 ac6 = get_word(calibration, 10, False)
086 b1 =  get_word(calibration, 12, True)
087 b2 =  get_word(calibration, 14, True)
088 mb =  get_word(calibration, 16, True)
089 mc =  get_word(calibration, 18, True)
090 md =  get_word(calibration, 20, True)
091
092
093 while True:
094     # Read raw temperature
095     write_byte(0xF4, 0x2E)          # Tell the sensor to take a temperature reading
096     time.sleep(temp_wait_period)    # Wait for the conversion to take place
097     temp_raw = read_word_2c(0xF6)
098
099     write_byte(0xF4, 0x34 + (oss << 6)) # Tell the sensor to take a pressure reading
100     time.sleep(pressure_wait_period)    # Wait for the conversion to take place
101     pressure_raw = ((read_byte(0xF6) << 16) \
102                      + (read_byte(0xF7) << 8) \
103                      + (read_byte(0xF8)) ) >> (8-oss)
104
105
106     temperature, pressure = calculate()
107     print time.time(), temperature / 10., pressure / 100.
108     time.sleep(1)
```

To get a reading out of the sensor you first have to read the factory set calibration block (lines 080-090). This is different for each device and is used in the lengthy calculations for both temperature and pressure. The function **calculate()** is just a direct translation of the code presented in the datasheet, I don't understand what it's doing but it gives us the required values.

Testing the sensor and the code

To test everything was working OK I saved the above code to a file called read-pressure.py, ran it and re-directed the output to a file

```
sudo ./read-pressure.py > pressure-test.dat
```

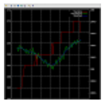
I then slowly walked up and down the stairs in my house to get some data. Then plotted the data with the following gnuplot program

```
set terminal wxt persist size 800,800 background '#000000'
set style line 99 linecolor rgb "#ffffff" linetype 0 linewidth 2
set key top right textcolor linestyle 99
set grid linestyle 99
set border linestyle 99

set yrange  [16.4:17.2]
set y2range [1003.5:1005]
set y2tics

plot filename using 1:2 axes x1y1 title "True temp" w l ,\
     filename using 1:3 axes x1y2 title "True pressure" w l, \
     filename using 1:3 axes x1y2 title "Smoothed" smooth bezier
```

Here is the command to generate the plot below

```
gnuplot -e "filename='pressure-test.dat'" gnuplot-pressure.plg
```

You can see the pressure dropping as I went up the stairs and then back down again. You can see the temperature went up slightly too which I think was just heat from my hand slowly raising it.



Interfacing a BMP085 Digital Pressure sensor to the Raspberry Pi

I recently bought a sensor with a BMP085 Digital Pressure sensor on it so I thought I'd write a post on how to read the data from the R...

Labels

- 1-wire (1)
- ADXL345 (1)
- BMP085 (2)
- DS18B20 (1)
- gnuplot (3)
- GY80 (1)
- HMC5883L (3)
- L3G4200D (1)
- MPU-6050 (6)
- OpenGL (2)
- Python (8)
- Raspberry Pi (8)
- Raspbian (4)
- temperature (1)

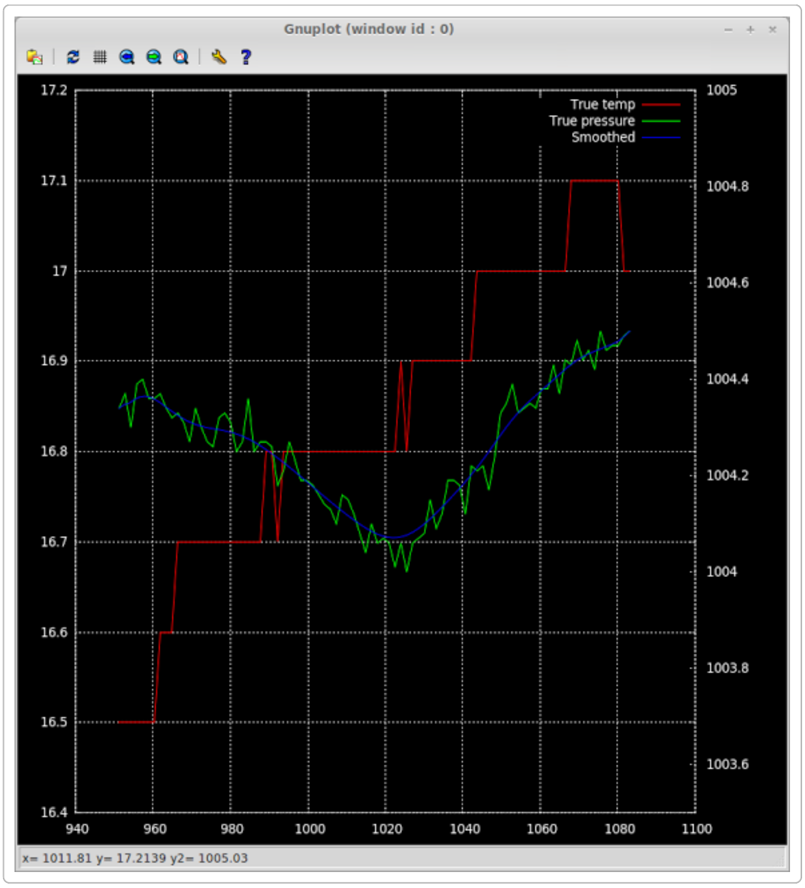
Blog Archive

- ► 2014 (2)
- ▼ 2013 (7)
  - ► December (1)
  - ▼ November (6)
    - Interfacing a BMP085 Digital Pressure sensor to th...
    - Connecting and calibrating a HMC5883L Compass on t...
    - Using a complementary filter to combine Accelerome...
    - 3D OpenGL visualisation of the data from an MPU-60...
    - Reading data from the MPU-6050 on the Raspberry Pi...
    - Interfacing Raspberry Pi and MPU-6050

About Me

Andrew Birkett

View my complete profile



Sensor response as I walked up and down the stairs

To calculate altitude (height above ground) I used the first (p_0) and the lowest (p) readings from the output and plugged them into the following formula, again this is taken from the datasheet.

$$altitude = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right)$$

This gave me a height of 2.86m, I was surprised to get a significant reading by just walking up and down the stairs so when I finally add it to a quad-copter I should get good results.

Posted by Andrew Birkett at 21:02 G+1 +3 Recommander ce contenu sur Google

Labels: BMP085, gnuplot, Python, Raspberry Pi

3 comments

Add a comment as Julien Louette

Top comments

Andrew Birkett via Google+ 2 years ago - Shared publicly
Reading air pressure in Python on a **#raspberrypi**

1 · Reply

ngc405 ngc405 2 years ago - Shared publicly
Thank you for sharing your code

To avoid changing code when moving from Raspberry V1 to newer version (I got several, including red ones), I changed line 5 above (bus = smbus.SMBus(0) # or bus = smbus.SMBus(1) if you have a revision 2 board) to :

1

Andrew Birkett 2 years ago
Thanks for the code, I've just got a brand new pi which is a later revision so I'll test it out and then update the code in the blog post.

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)