嵌入式系統設計概論與實作

曾煜棋、吳昆儒

National Yang Ming Chiao Tung University

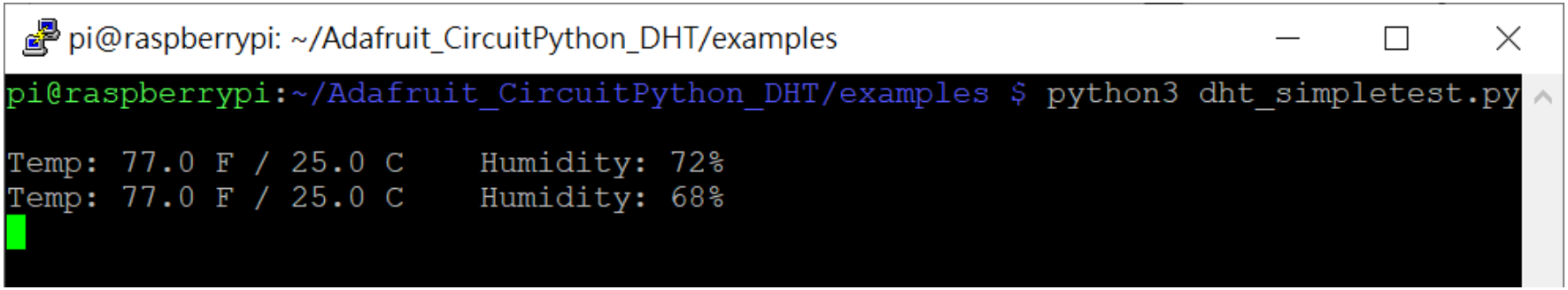
補充: 嵌入式系統設計概論與實作.0312update

周次	日期	主題
1	2/26	0.嵌入式課程介紹
2	3/5	1.嵌入式開發板 - 樹莓派介紹與設定
3	3/12	2.感測器應用(溫溼度、超音波)
4	3/19	3.人體活動偵測
5	3/26	4.人體活動偵測
~~6~~	~~4/2~~	~~兒童節及民族掃墓節調整放假(2日-5日)~~
7	4/9	5.網路攝影機 IP cam
8	4/16	6. 網路攝影機 + 機器學習影像辨識 (NYCU:期中考試(12日-16日)) By 台灣樹莓派的講師!!
9	4/23	7. 網路攝影機 + 影像辨識
10	4/30	Midterm, Project分組
11	5/7	8.推播廣告(beacon)應用
12	5/14	9.語音助理
13	5/21	Final Project – Proposal
14	5/28	10.樹莓派核心編譯 (Cross compile, Kernel)
15	6/4	Final Project prepare, Q&A, 補demo
16	6/11	Final Project demonstration (NYCU:學期考試(7日-11日))
17	6/18	(暫定)Final Project demonstration part 2

補充-DHT11 problem

- 已知問題: 部分的DHT11模組, 使用現有的sample code無法讀取資料
- 解決方法: 使用新版的Adafruit_CircuitPython_DHT, 可順利讀到資料

  - pip3 install adafruit-circuitpython-dht
  - sudo apt-get install -y libgpiod-dev
  - wget https://raw.githubusercontent.com/adafruit/Adafruit_CircuitPython_DHT/master/examples/dht_simpletest.py
  - *# 預設是DHT22, 記得改成DHT11.*
  - *# ex: dhtDevice = adafruit_dht.DHT**11**(board.D**4**)*
  - python3 dht_simpletest.py



補充-陀螺儀的參數(DPS/digit)

EX: FS=250dps, Sensitivity=8.75 mbps/digit

- Raw data = 1(bit value) = 0.00875 degree per second
- Raw data = 16(bit) = 0.00875*16 = 0.14 dps
- # Two's Complement Integer Ranges:
  - -2^15 (-32,768) to +2^15-1 (32,767)
- Raw data = 32767 = 286.71125 dps
- Raw data = -32,768 = -286.72 dps
  - 實際可測到286, 說明書寫250比較保守

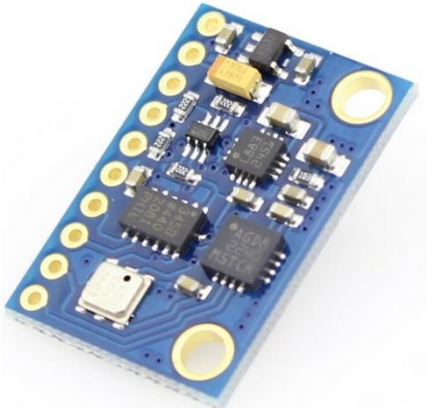
		32,767	-32,768
FS	sensitivity	max	min
250	8.75	286.7113	-286.72
500	17.5	573.4225	-573.44
2000	70	2293.76	2293.76

Table 4. Mechanical characteristics @ Vdd = 3.0 V, T = 25 °C, unless otherwise noted(1)

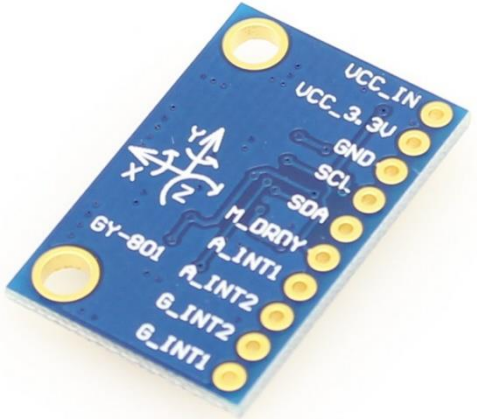
Symbol	Parameter	Test condition	Min.	Typ.(2)	Max.	Unit
FS	Measurement range	User-selectable		±250		dps
				±500		
				±2000		
So	Sensitivity	FS = 250 dps		8.75		mdps/digit
		FS = 500 dps		17.50		
		FS = 2000 dps		70		

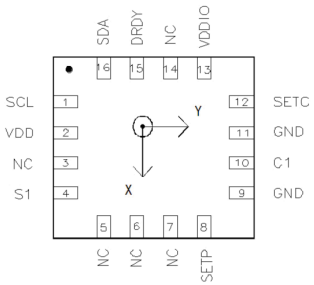
Last week & This week

- 嵌入式應用: 人體活動偵測
  - 加速度、陀螺儀...等



- GY801 (I2C sensor)
  - 3-axis Accelerometer, Gyroscope, magnetometer and pressure
  1. ADXL345：Accelerometer
  2. L3G4200：Gyroscope
  3. HMC5883：Magnetometer
  4. BMP085：Pressure

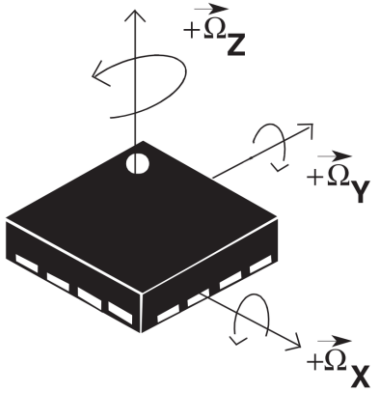


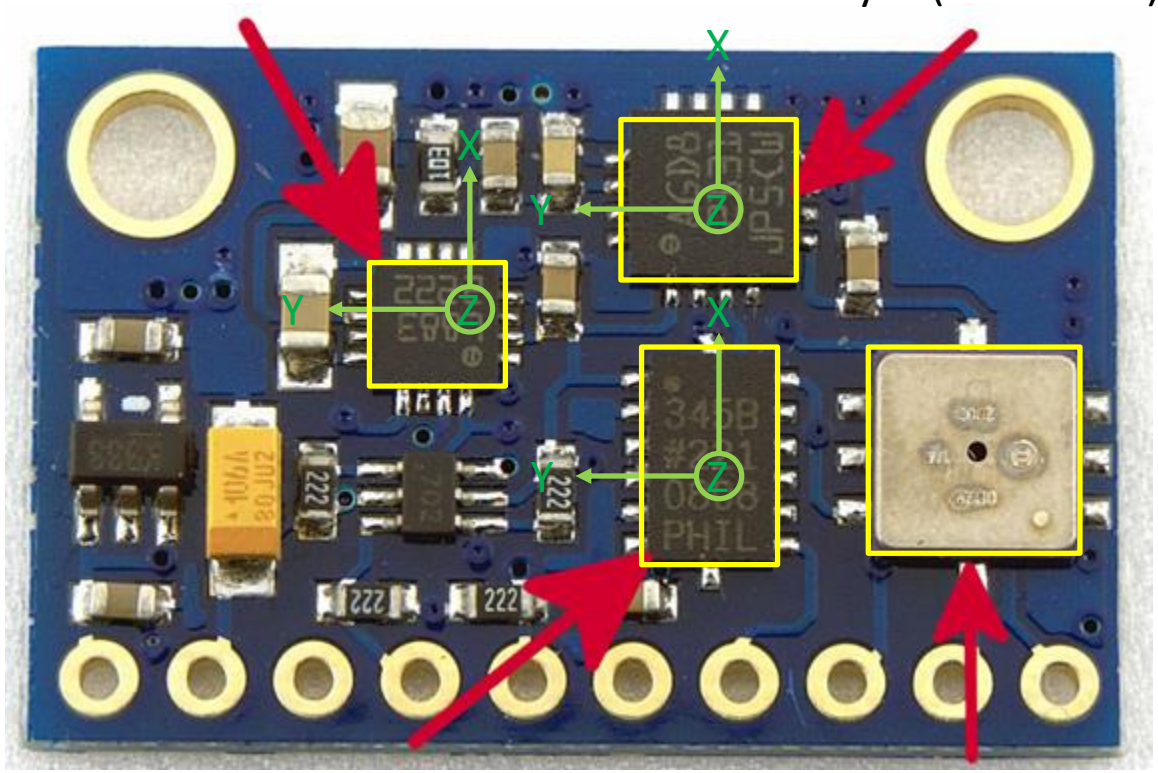


GY-801 (10 DoF sensor)

Compass (HMC5883L)

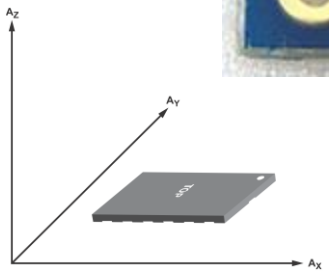
Gyro (L3G4200D)

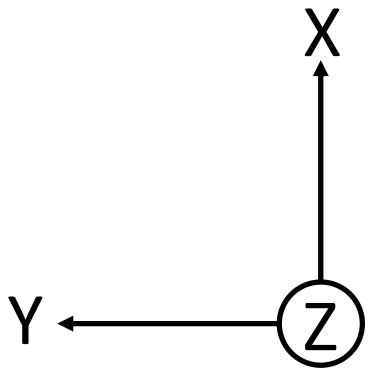




G-sensor (ADXL345)

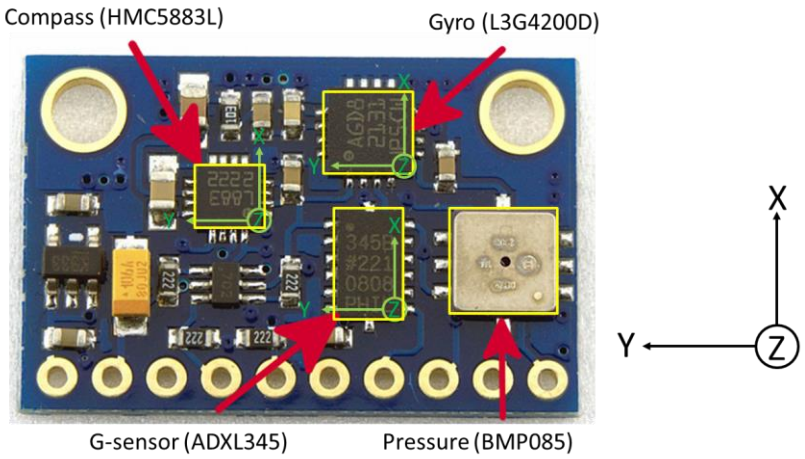
Pressure (BMP085)

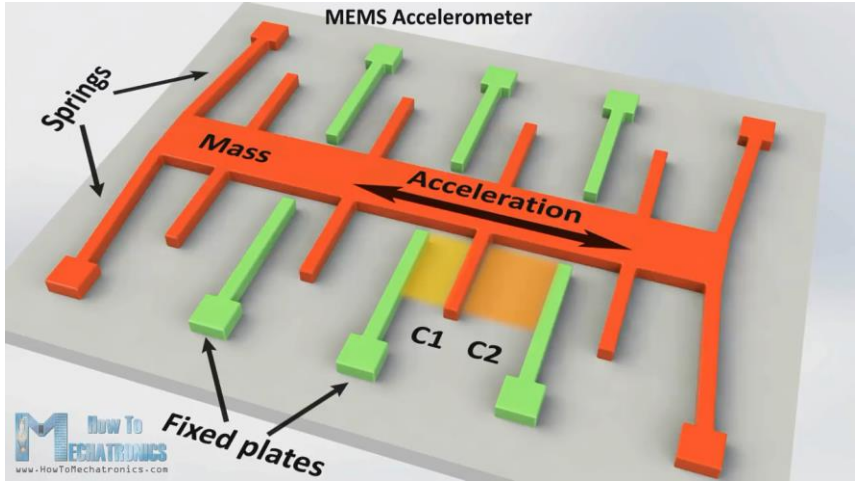


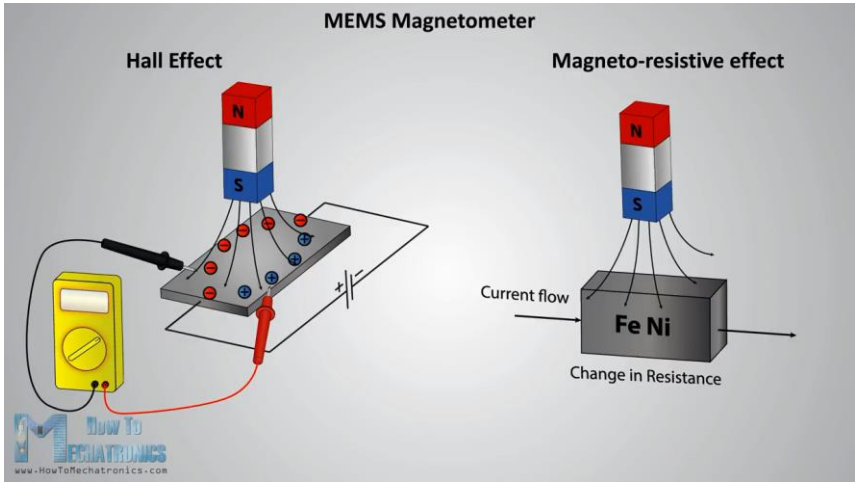


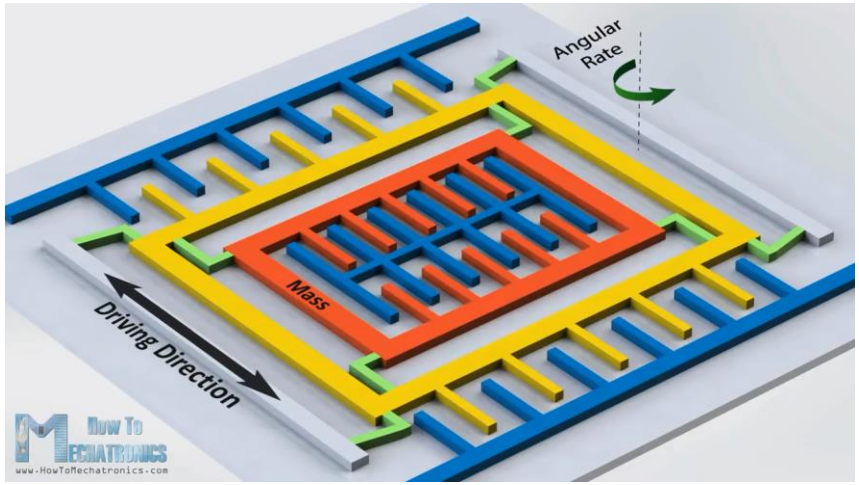
https://www.youtube.com/watch?v=eqZgxR6eRjo

How MEMS Accelerometer, Gyroscope, Magnetometer Work





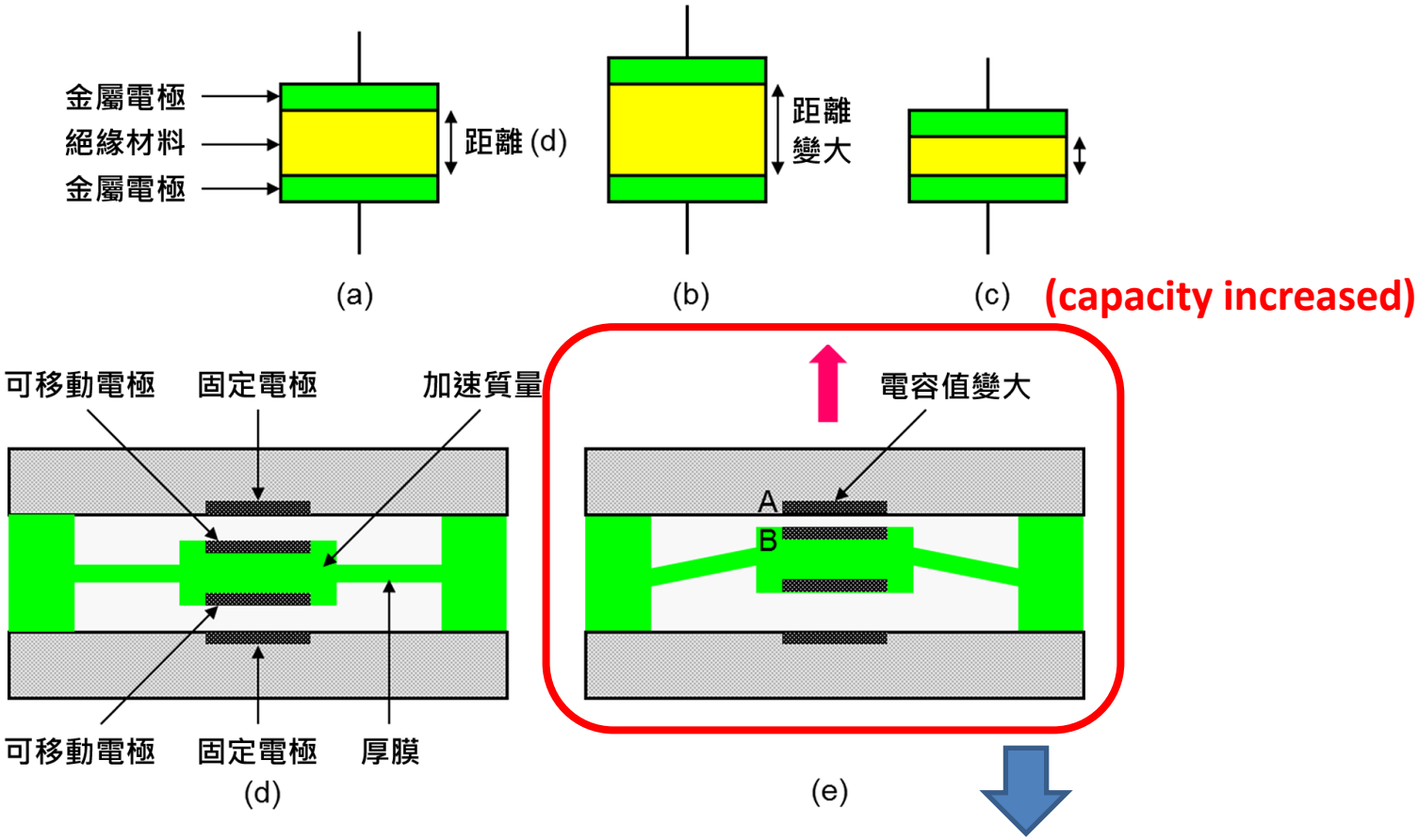




IMU. How MEMS Accelerometer Gyroscope Magnetometer Work & Arduino Tutorial

電容微加速度器的構造與原理
http://scimonth.blogspot.tw/2014/07/blog-post_94.html

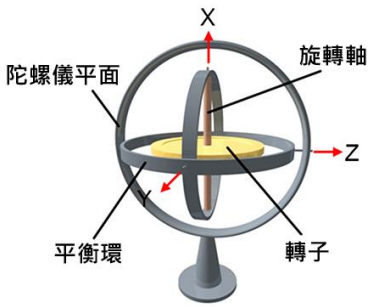
Sensor - Accelerometer



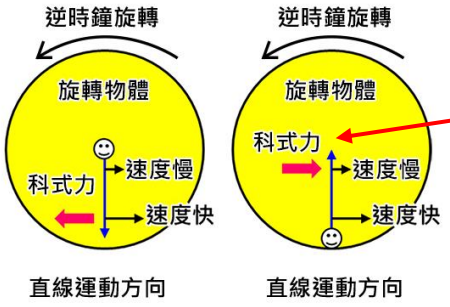
When you move the sensor, the distance (**capacity**) is changed.
We **use capacity to calculate acceleration**.

微陀螺儀的構造與原理
http://scimonth.blogspot.tw/2014/07/blog-post_94.html

Sensor - Gyro

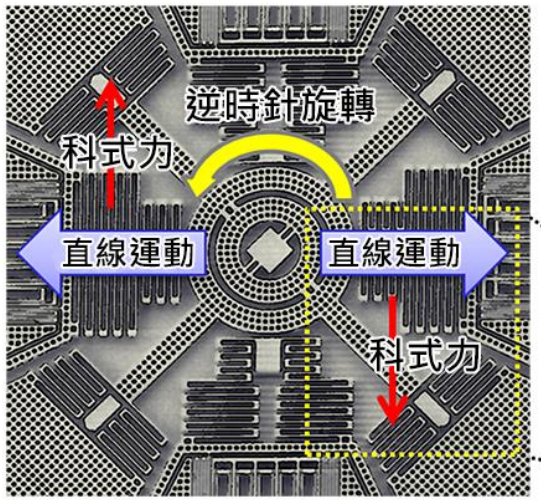


(a)

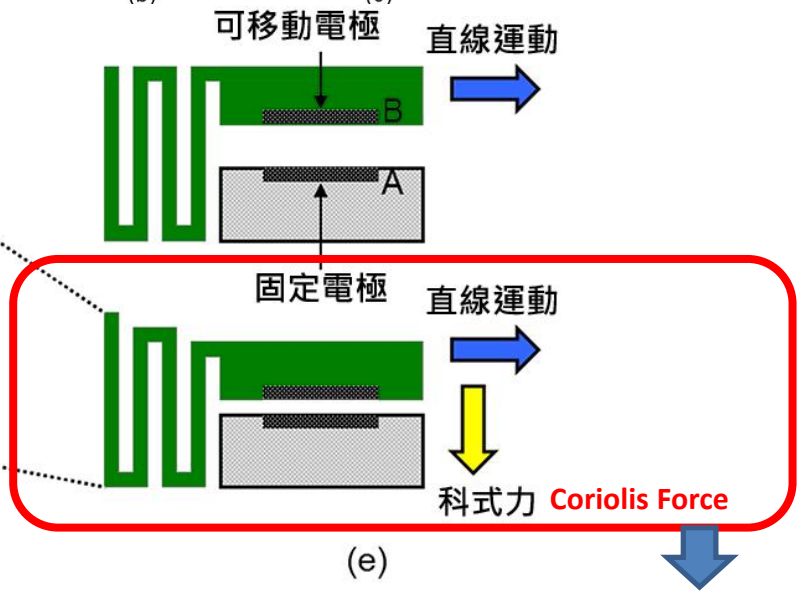


(b) (c)

Coriolis Force (科氏力)



(d)

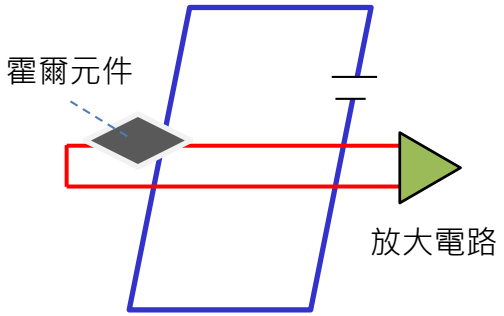


(e)

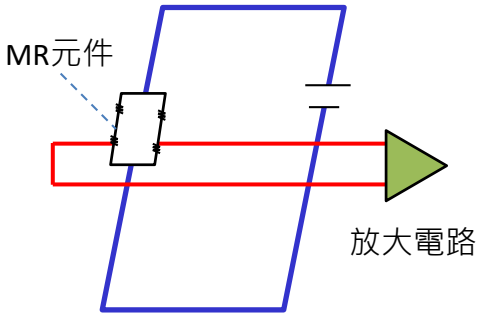
When you rotate the sensor, Coriolis Force change the distance (**capacity**).
We **use capacity to calculate Angular velocity (角速度).**

地磁感測器
https://www.rohm.com.tw/electronics-basics/sensors/sensor_what2

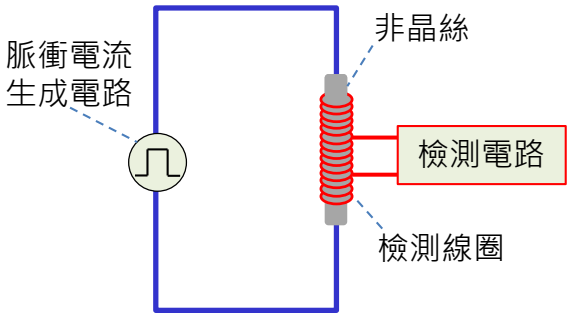
Sensor - Magnetometer



基於磁場變化
電壓隨著霍爾效應發生變化



基於磁場變化
MR元件的電阻發生變化



基於磁場變化
利用脈衝電流檢測線圈電壓

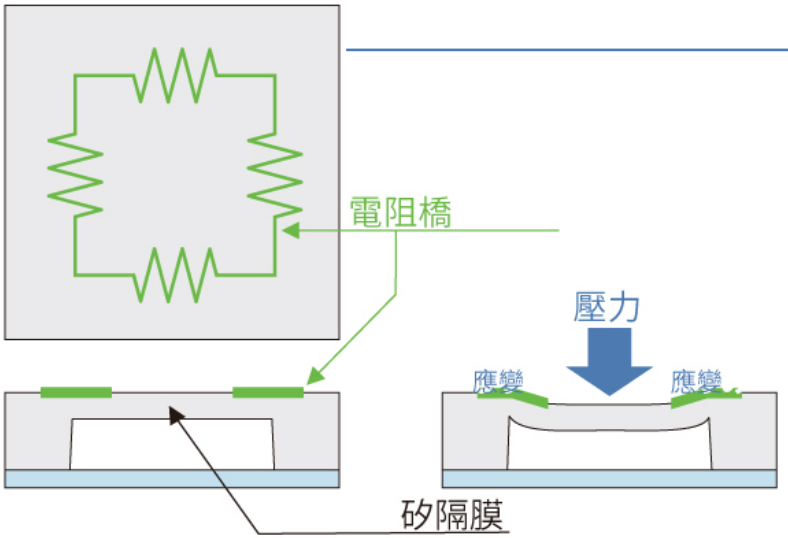
Magneto Resistance 磁阻效應感測器

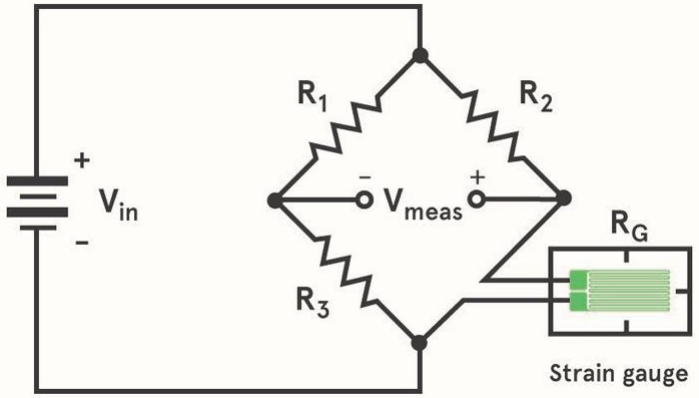
cost compassing and magnetometry. The HMC5883L includes our state-of-the-art, high-resolution HMC118X series magneto-resistive sensors plus an ASIC containing amplification, automatic degaussing strap drivers, offset cancellation, and a 12-bit ADC that enables 1° to 2° compass heading accuracy. The I^2C

壓阻式氣壓感測器
https://www.rohm.com.tw/electronics-basics/sensors/sensor_what4

Sensor - Pressure

當R3/R1 = RG/R2時，電橋平衡，檢流計無電流通過。
對於電流是否經過非常敏感，可以獲取頗精確的測量。





Wheatstone bridge

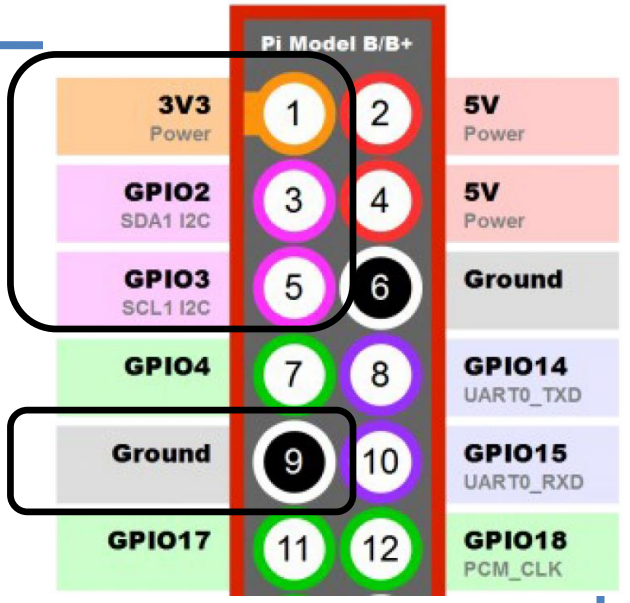
表面擴散雜質形成電阻橋電路(Wheatstone bridge)，
施加壓力產生的變形會影響電阻值，進而來計算壓力（氣壓）。

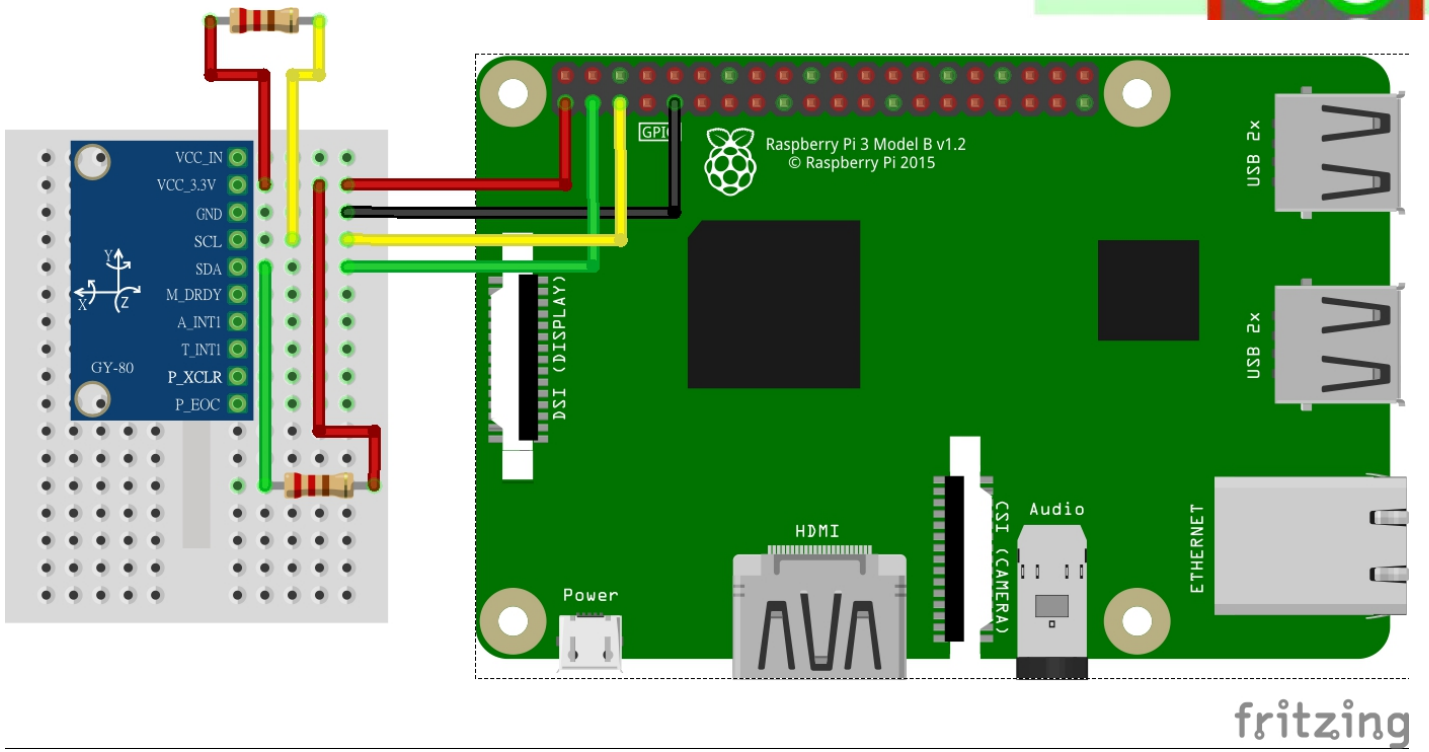
Piezo-Resistive 壓阻式感測器

The BMP085 is based on piezo-resistive technology for EMC robustness, high accuracy and linearity as well as long term stability.

Connect GY801

VCC	**Pin 1 (3.3V), Red line**
GND	**Pin 9 (Ground), Black line**
SCL	**Pin 5 (SCL), Yellow line**
SDA	**Pin 3 (SDA), Green line**





The resistors are build-in on circuit. (you can skip them)

After connecting GY801

- Install I2C tool to check the state:
  - **sudo apt-get install i2c-tools**
  - Command 1
    - **sudo ls -al /dev/*i2c***
  - Command 2
    - **sudo i2cdetect -y 1**

```
pi@raspberrypi:~$ sudo ls -al /dev/*i2c*
crw-rw---- 1 root i2c 89, 1 Mar 7 03:39 /dev/i2c-1
```

Show the I2C address

```
pi@raspberrypi:~$ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- 1e --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- 53 -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- 69 -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
```

After connecting GY801

- If no I2C device
  - Command 1
    - sudo ls -al /dev/*i2c*

```
pi@raspberrypi:~$ ls -al /dev/*i2c*
ls: cannot access /dev/*i2c*: No such file or directory
```

  - Command 2
    - sudo i2cdetect -y 1

```
pi@raspberrypi:~/gy801$ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Check your GY801

注意! 檢查磁力計的晶片!!

```
pi@raspberrypi:~$ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- 1e --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- 53 -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- 69 -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
```

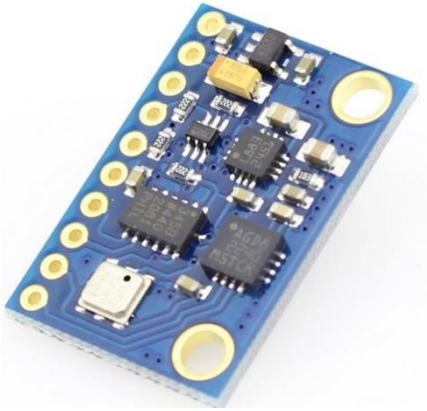
晶片組: HMC5883
(address: 0x1e)

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: 30 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- 53 -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- 69 -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
```

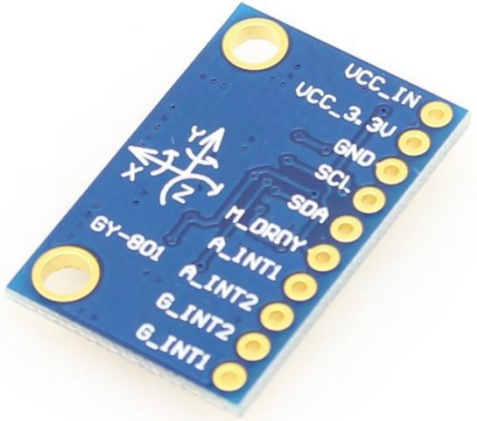
晶片組: QMC 或 MMC
(address: 0x30)

Outline

- 嵌入式應用: 人體活動偵測
  - 加速度、陀螺儀...等



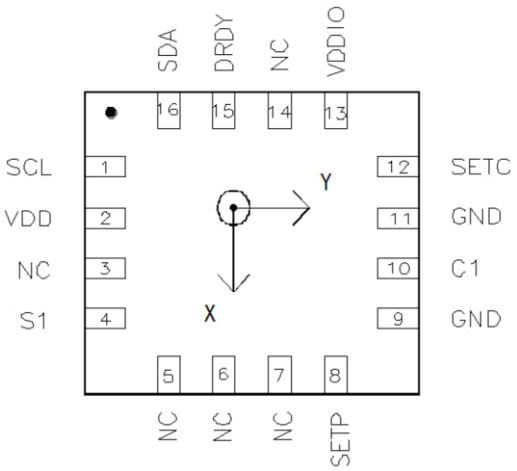
- GY801 (I2C sensor)
  1. 3-axis Accelerometer, Gyroscope, magnetometer and pressure
  2. ADXL345：Accelerometer
  3. L3G4200：Gyroscope
  4. HMC5883：Magnetometer
  5. BMP085：Pressure

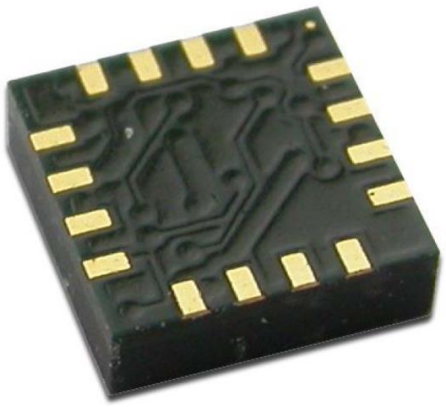


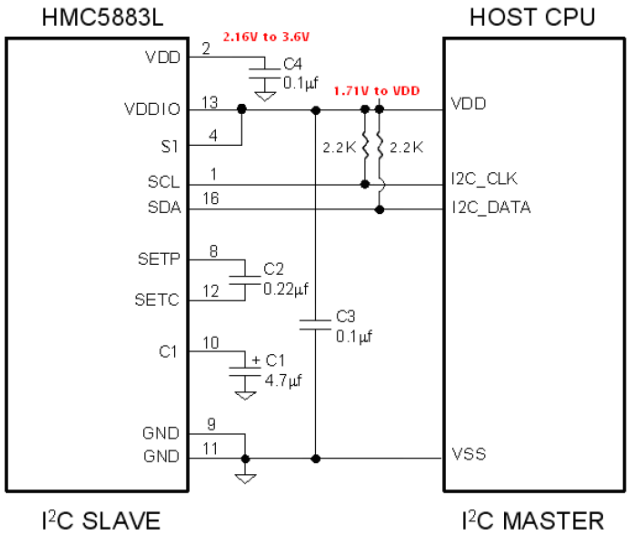
https://sites.google.com/site/playrobotquickstart/home/hmc583l-dian-zi-luo-pan
https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf
http://web.sensor-ic.com:8000/ZLXIAZAI/20130122/HMC5883L中文规格书-Feb%202011.pdf

3. Magnetometer (HMC5883L)

- HMC5883L
  - 3-Axis Magnetoresistive Sensors and ASIC in a 3.0x3.0x0.9mm LCC Surface Mount Package
  - 12-Bit ADC Coupled with Low Noise AMR Sensors Achieves 2 milli-gauss Field Resolution in ±8 Gauss Fields
  - Low Voltage Operations (2.16 to 3.6V) and Low Power Consumption (100 μ A)
  - I2C Digital Interface
  - Wide Magnetic Field Range (+/-8 Oe)







3. Magnetometer (HMC5883L)

- HMC5883L procedure

Below is an example of a (power-on) initialization process for “continuous-measurement mode”:

1. Write CRA (00) – send **0x3C 0x00 0x70** (8-average, 15 Hz default, normal measurement)
2. Write CRB (01) – send **0x3C 0x01 0xA0** (Gain=5, or any other desired gain)
3. Write Mode (02) – send **0x3C 0x02 0x00** (Continuous-measurement mode)
4. Wait 6 ms or monitor status register or DRDY hardware interrupt pin
5. Loop

   Send **0x3D 0x06** (Read all 6 bytes. If gain is changed then this data set is using previous gain)
   Convert three 16-bit 2’s compliment hex values to decimal values and assign to X, Z, Y, respectively.
   Send **0x3C 0x03** (point to first data register 03)
   Wait about 67 ms (if 15 Hz rate) or monitor status register or DRDY hardware interrupt pin

   End_loop

The default (factory) HMC5883L 8-bit slave address is 0x3C for write operations, or 0x3D for read operations.

3. Magnetometer (HMC5883L)

- Sample code results:

COM6 - PuTTY

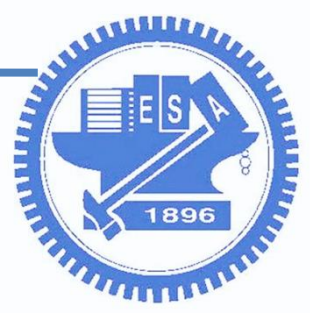
```
pi@raspberrypi:~/gy801$ python 3compasswu.py
X = 240 , Y = -43 , Z = 411 (gauss)
X = 240 , Y = -43 , Z = 411 (gauss)
X = 240 , Y = -43 , Z = 411 (gauss)
X = 240 , Y = -43 , Z = 411 (gauss)
X = 240 , Y = -43 , Z = 411 (gauss)
X = 240 , Y = -43 , Z = 411 (gauss)
X = 240 , Y = -43 , Z = 411 (gauss)
X = 240 , Y = -43 , Z = 411 (gauss)
X = 240 , Y = -43 , Z = 411 (gauss)
X = 240 , Y = -43 , Z = 411 (gauss)
```

3. Compass code: define address, read byte data


```
12  # the following address is defined by datasheet
13  #HMC5883L (Magnetometer) constants
14  HMC5883L_ADDRESS          =     0x1E  # I2C address
15
16  HMC5883L_CRA              =     0x00  # write CRA(00), Configuration Register A
17  HMC5883L_CRB              =     0x01  # write CRB(01), Configuration Register B
18  HMC5883L_MR               =     0x02  # write Mode(02)
19  HMC5883L_DO_X_H           =     0x03  # Data Output
20  HMC5883L_DO_X_L           =     0x04
21  HMC5883L_DO_Z_H           =     0x05
22  HMC5883L_DO_Z_L           =     0x06
23  HMC5883L_DO_Y_H           =     0x07
24  HMC5883L_DO_Y_L           =     0x08
```

```
56      def read_word(self,adr,rf=1):
57          # rf=1 Little Endian Format, rf=0 Big Endian Format
58          if (rf == 1):
59              # acc, gyro
60              low = self.read_byte(adr)
61              high = self.read_byte(adr+1)
62          else:
63              # compass
64              high = self.read_byte(adr)
65              low = self.read_byte(adr+1)
66          val = (high << 8) + low
67          return val
```

a. Write CRA(00)



Below is an example of a (power-on) initialization process for "continuous-measurement mode":

1. Write CRA (00) – send **0x3C 0x00 0x70** (8-average, 15 Hz default, normal measurement)
2. Write CRB (01) – send **0x3C 0x01 0xA0** (Gain=5, or any other desired gain)
3. Write Mode (02) – send **0x3C 0x02 0x00** (Continuous-measurement mode)
4. Wait 6 ms or monitor status register or DRDY hardware interrupt pin
5. Loop

   Send **0x3D 0x06** (Read all 6 bytes. If gain is changed then this data set is using previous gain)
   Convert three 16-bit 2's compliment hex values to decimal values and assign to X, Z, Y, respectively.
   Send **0x3C 0x03** (point to first data register 03)
   Wait about 67 ms (if 15 Hz rate) or monitor status register or DRDY hardware interrupt pin

   End_loop

```
194         # Declination Angle
195         self.angle_offset = ( -4 + (32/60)) / (180 / pi)
196         # Formula: (deg + (min / 60.0)) / (180 / M_PI);
197         # ex: Hsinchu = Magnetic Declination: -4 deg, 32 min
198         # declinationAngle = ( -4 + (32/60)) / (180 / M_PI)
199         # http://www.magnetic-declination.com/
200
201         self.scale = 0.92 # convert bit value(LSB) to gauss. DigitalResolution
202
203         # Configuration Register A, write value: 0111 0000
204         self.write_byte(HMC5883L_CRA, 0b01110000)
205         # CRA6-CRA5 = 11 -> 8 samples per measurement
206         # CRA4-CRA2 = 100 -> Data Output Rate = 15Hz
207         # CRA1-CRA0 = 00 -> Normal measurement configuration (Default)
208
209
210         # Configuration Register B , write value: 0010 0000
211         self.write_byte(HMC5883L_CRB, 0b00100000)
212         # CRB7-CRB5 = 001 (Gain Configuration Bits) -> Gain=1090(LSb/Gauss), default
213         # ps. output range = -2048 to 2047
214
215
216         # Mode Register, write value: 0000 0000
217         self.write_byte(HMC5883L_MR, 0b00000000)
218         # MR1-MR0 = 00 (Mode Select Bits) -> Continuous-Measurement Mode.
```

a. Write CRA(00)

```
203         # Configuration Register A, write value: 0111 0000
204         self.write_byte(HMC5883L_CRA, 0b01110000)
205         # CRA6-CRA5 = 11 -> 8 samples per measurement
206         # CRA4-CRA2 = 100 -> Data Output Rate = 15Hz
207         # CRA1-CRA0 = 00 -> Normal measurement configuration (Default)
```

CRA7	CRA6	CRA5	CRA4	CRA3	CRA2	CRA1	CRA0
(0)	MA1(0)	MA0(0)	DO2 (1)	DO1 (0)	DO0 (0)	MS1 (0)	MS0 (0)

Table 3: Configuration Register A

Location	Name	Description
CRA7	CRA7	Bit CRA7 is reserved for future function. Set to 0 when configuring CRA.
CRA6 to CRA5	MA1 to MA0	Select number of samples averaged (1 to 8) per measurement output. 00 = 1(Default); 01 = 2; 10 = 4; 11 = 8

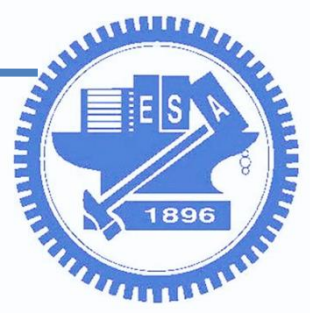
8 samples

DO2	DO1	DO0	Typical Data Output Rate (Hz)
0	0	0	0.75
0	0	1	1.5
0	1	0	3
0	1	1	7.5
1	0	0	15 (Default)
1	0	1	30
1	1	0	75
1	1	1	Reserved

Table 5: Data Output Rates

15 Hz

b. Write CRB(01)



Below is an example of a (power-on) initialization process for "continuous-measurement mode":

1. Write CRA (00) – send **0x3C 0x00 0x70** (8-average, 15 Hz default, normal measurement)
2. Write CRB (01) – send **0x3C 0x01 0xA0** (Gain=5, or any other desired gain)
3. Write Mode (02) – send **0x3C 0x02 0x00** (Continuous-measurement mode)
4. Wait 6 ms or monitor status register or DRDY hardware interrupt pin
5. Loop

   Send **0x3D 0x06** (Read all 6 bytes. If gain is changed then this data set is using previous gain)
   Convert three 16-bit 2's compliment hex values to decimal values and assign to X, Z, Y, respectively.
   Send **0x3C 0x03** (point to first data register 03)
   Wait about 67 ms (if 15 Hz rate) or monitor status register or DRDY hardware interrupt pin

   End_loop

```
194         # Declination Angle
195         self.angle_offset = ( -4 + (32/60)) / (180 / pi)
196         # Formula: (deg + (min / 60.0)) / (180 / M_PI);
197         # ex: Hsinchu = Magnetic Declination: -4 deg, 32 min
198         # declinationAngle = ( -4 + (32/60)) / (180 / M_PI)
199         # http://www.magnetic-declination.com/
200
201         self.scale = 0.92 # convert bit value(LSB) to gauss. DigitalResolution
202
203         # Configuration Register A, write value: 0111 0000
204         self.write_byte(HMC5883L_CRA, 0b01110000)
205         # CRA6-CRA5 = 11 -> 8 samples per measurement
206         # CRA4-CRA2 = 100 -> Data Output Rate = 15Hz
207         # CRA1-CRA0 = 00 -> Normal measurement configuration (Default)
208
209
210         # Configuration Register B , write value: 0010 0000
211         self.write_byte(HMC5883L_CRB, 0b00100000)
212         # CRB7-CRB5 = 001 (Gain Configuration Bits) -> Gain=1090(LSb/Gauss), default
213         # ps. output range = -2048 to 2047
214
215
216         # Mode Register, write value: 0000 0000
217         self.write_byte(HMC5883L_MR, 0b00000000)
218         # MR1-MR0 = 00 (Mode Select Bits) -> Continuous-Measurement Mode.
```

b. Write CRB(01)

注意! Datasheet寫0xA0, 這邊是0x20

```
210         # Configuration Register B , write value: 0010 0000
211         self.write_byte(HMC5883L_CRB, 0b00100000)
212         # CRB7-CRB5 = 001 (Gain Configuration Bits) -> Gain=1090(LSb/Gauss), default
213         # ps. output range = -2048 to 2047
```

CRB7	CRB6	CRB5	CRB4	CRB3	CRB2	CRB1	CRB0
GN2 (0)	GN1 (0)	GN0 (1)	(0)	(0)	(0)	(0)	(0)

Table 7: Configuration B Register

GN2	GN1	GN0	Recommended Sensor Field Range	Gain (LSb/ Gauss)	Digital Resolution (mG/LSb)	Output Range
0	0	0	± 0.88 Ga	1370	0.73	0xF800–0x07FF (-2048–2047)
0	0	1	± 1.3 Ga	1090 (default)	0.92	0xF800–0x07FF (-2048–2047)
0	1	0	± 1.9 Ga	820	1.22	0xF800–0x07FF (-2048–2047)
0	1	1	± 2.5 Ga	660	1.52	0xF800–0x07FF (-2048–2047)
1	0	0	± 4.0 Ga	440	2.27	0xF800–0x07FF (-2048–2047)
1	0	1	± 4.7 Ga	390	2.56	0xF800–0x07FF (-2048–2047)
1	1	0	± 5.6 Ga	330	3.03	0xF800–0x07FF (-2048–2047)
1	1	1	± 8.1 Ga	230	4.35	0xF800–0x07FF (-2048–2047)

Table 9: Gain Settings

3. Compass code: Write value to specific address

```
210     # Configuration Register B , write value: 0010 0000
211     self.write_byte(HMC5883L_CRB, 0b00100000)
212     # CRB7-CRB5 = 001 (Gain Configuration Bits) -> Gain=1090(LSb/Gauss), default
213     # ps. output range = -2048 to 2047
```

CRB7	CRB6	CRB5	CRB4	CRB3	CRB2	CRB1	CRB0
GN2 (0)	GN1 (0)	GN0 (1)	(0)	(0)	(0)	(0)	(0)

Table 7: Configuration B Register

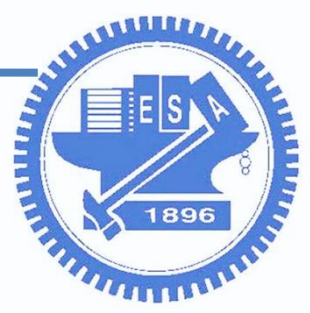
GN2	GN1	GN0	Recommended Sensor Field Range	Gain (LSb/ Gauss)	Digital Resolution (mG/LSb)	Output Range
0	0	0	± 0.88 Ga	1370	0.73	0xF800–0x07FF (-2048–2047)
0	0	1	± 1.3 Ga	1090 (default)	0.92	0xF800–0x07FF (-2048–2047)

$$m_{gauss} = m_{raw} * \mathbf{DigitalResolution}$$

When it report -4096, the overflow occurs.

- In the event the ADC reading overflows or underflows for the given channel, or if there is a math overflow during the bias measurement, this data register will contain the value -4096. This register value will clear when after the next valid measurement is made.

c. Write Mode(02)



Below is an example of a (power-on) initialization process for "continuous-measurement mode":

1. Write CRA (00) – send **0x3C 0x00 0x70** (8-average, 15 Hz default, normal measurement)
2. Write CRB (01) – send **0x3C 0x01 0xA0** (Gain=5, or any other desired gain)
3. Write Mode (02) – send **0x3C 0x02 0x00** (Continuous-measurement mode)
4. Wait 6 ms or monitor status register or DRDY hardware interrupt pin
5. Loop

   Send **0x3D 0x06** (Read all 6 bytes. If gain is changed then this data set is using previous gain)
   Convert three 16-bit 2's compliment hex values to decimal values and assign to X, Z, Y, respectively.
   Send **0x3C 0x03** (point to first data register 03)
   Wait about 67 ms (if 15 Hz rate) or monitor status register or DRDY hardware interrupt pin

   End_loop

```
194         # Declination Angle
195         self.angle_offset = ( -4 + (32/60)) / (180 / pi)
196         # Formula: (deg + (min / 60.0)) / (180 / M_PI);
197         # ex: Hsinchu = Magnetic Declination: -4 deg, 32 min
198         # declinationAngle = ( -4 + (32/60)) / (180 / M_PI)
199         # http://www.magnetic-declination.com/
200
201         self.scale = 0.92 # convert bit value(LSB) to gauss. DigitalResolution
202
203         # Configuration Register A, write value: 0111 0000
204         self.write_byte(HMC5883L_CRA, 0b01110000)
205         # CRA6-CRA5 = 11 -> 8 samples per measurement
206         # CRA4-CRA2 = 100 -> Data Output Rate = 15Hz
207         # CRA1-CRA0 = 00 -> Normal measurement configuration (Default)
208
209
210         # Configuration Register B , write value: 0010 0000
211         self.write_byte(HMC5883L_CRB, 0b00100000)
212         # CRB7-CRB5 = 001 (Gain Configuration Bits) -> Gain=1090(LSb/Gauss), default
213         # ps. output range = -2048 to 2047
214
215
216         # Mode Register, write value: 0000 0000
217         self.write_byte(HMC5883L_MR, 0b00000000)
218         # MR1-MR0 = 00 (Mode Select Bits) -> Continuous-Measurement Mode.
```

c. Write Mode(02)

```
216         # Mode Register, write value: 0000 0000
217         self.write_byte(HMC5883L_MR, 0b00000000)
218         # MR1-MR0 = 00 (Mode Select Bits) -> Continuous-Measurement Mode.
```

MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
HS(0)	(0)	(0)	(0)	(0)	(0)	MD1 (0)	MD0 (1)

Table 10: Mode Register

MD1	MD0	Operating Mode
0	0	Continuous-Measurement Mode. In continuous-measurement mode, the device continuously performs measurements and places the result in the data register. RDY goes high when new data is placed in all three registers. After a power-on or a write to the mode or configuration register, the first measurement set is available from all three data output registers after a period of $2/f_{DO}$ and subsequent measurements are available at a frequency of f_{DO}, where f_{DO} is the frequency of data output.
0	1	Single-Measurement Mode (Default). When single-measurement mode is selected, device performs a single measurement, sets RDY high and returned to idle mode. Mode register returns to idle mode bit values. The measurement remains in the data output register and RDY remains high until the data output register is read or another measurement is performed.
1	0	Idle Mode. Device is placed in idle mode.
1	1	Idle Mode. Device is placed in idle mode.

Table 12: Operating Modes

3. Compass code: offset

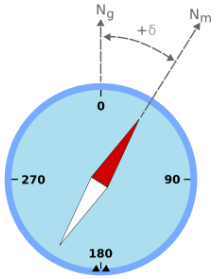

```
194     # Declination Angle
195     self.angle_offset = ( -4 + (32/60)) / (180 / pi)
196     # Formula: (deg + (min / 60.0)) / (180 / M_PI);
197     # ex: Hsinchu = Magnetic Declination: -4 deg, 32 min
198     # declinationAngle = ( -4 + (32/60)) / (180 / M_PI)
199     # http://www.magnetic-declination.com/
200
201     self.scale = 0.92 # convert bit value(LSB) to gauss. DigitalResolution
202
203     # Configuration Register A, write value: 0111 0000
204     self.write_byte(HMC5883L_CRA, 0b01110000)
205     # CRA6-CRA5 = 11 -> 8 samples per measurement
206     # CRA4-CRA2 = 100 -> Data Output Rate = 15Hz
207     # CRA1-CRA0 = 00 -> Normal measurement configuration (Default)
208
209
210     # Configuration Register B , write value: 0010 0000
211     self.write_byte(HMC5883L_CRB, 0b00100000)
212     # CRB7-CRB5 = 001 (Gain Configuration Bits) -> Gain=1090(LSb/Gauss), default
213     # ps. output range = -2048 to 2047
214
215
216     # Mode Register, write value: 0000 0000
217     self.write_byte(HMC5883L_MR, 0b00000000)
218     # MR1-MR0 = 00 (Mode Select Bits) -> Continuous-Measurement Mode.
```

https://en.wikipedia.org/wiki/Magnetic_declination

3. Compass code: Declination angle

https://www.rapidtables.com/convert/number/degrees-minutes-seconds-to-degrees.html

```
194         # Declination Angle
195         self.angle_offset = ( -1 * (4 + (32/60))) / (180 / pi)
196         # Formula: (deg + (min / 60.0)) / (180 / M_PI);
197         # ex: Hsinchu = Magnetic Declination: -4 deg, 32 min
198         # declinationAngle = ( -1 * (4 + (32/60))) / (180 / pi)
199         # http://www.magnetic-declination.com/
```



http://www.magnetic-declination.com/

Latitude: 24.4700° North
Longitude: 120.5900° East
Date: 2020-02-17

Magnetic declination: 4° 23.11' West

Annual Change (minutes/year): 4.8 '/y West

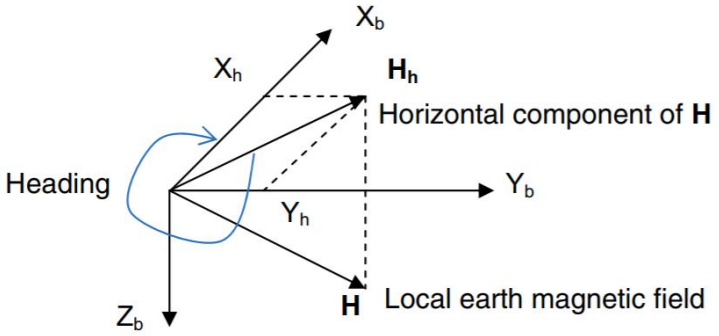
https://www.geomag.nrcan.gc.ca/calc/mdcal-en.php

3. Compass code: calculate heading

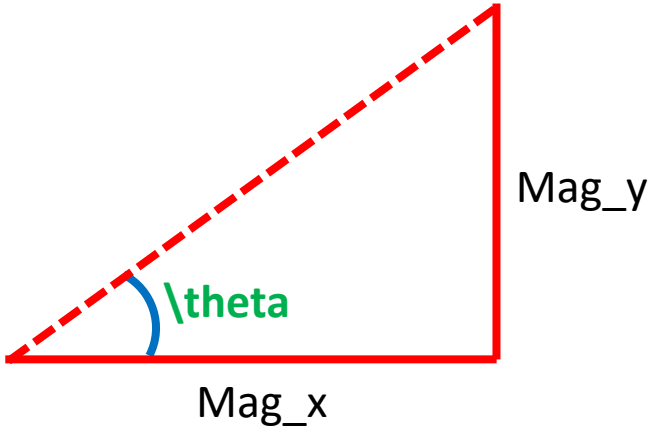
```
220     def getX(self):
221         self.X = (self.read_word_2c(HMC5883L_DO_X_H, rf=0) - self.Xoffset) * self.scale
222         return self.X
223
224     def getY(self):
225         self.Y = (self.read_word_2c(HMC5883L_DO_Y_H, rf=0) - self.Yoffset) * self.scale
226         return self.Y
227
228     def getZ(self):
229         self.Z = (self.read_word_2c(HMC5883L_DO_Z_H, rf=0) - self.Zoffset) * self.scale
230         return self.Z
231
232     def getHeading(self):
233         bearing  = degrees(atan2(self.getY(), self.getX()))
234
235         if (bearing < 0):
236             bearing += 360
237         if (bearing > 360):
238             bearing -= 360
239         self.angle = bearing + self.angle_offset
240         return self.angle
```

- Heading:
  - 1. measure **two orthogonal components** of the magnetic vector (**mx, my**)
  - 2. calculate the heading as the **arctangent** of their ratio

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$



3. Compass heading



\theta = arctan(y/x)

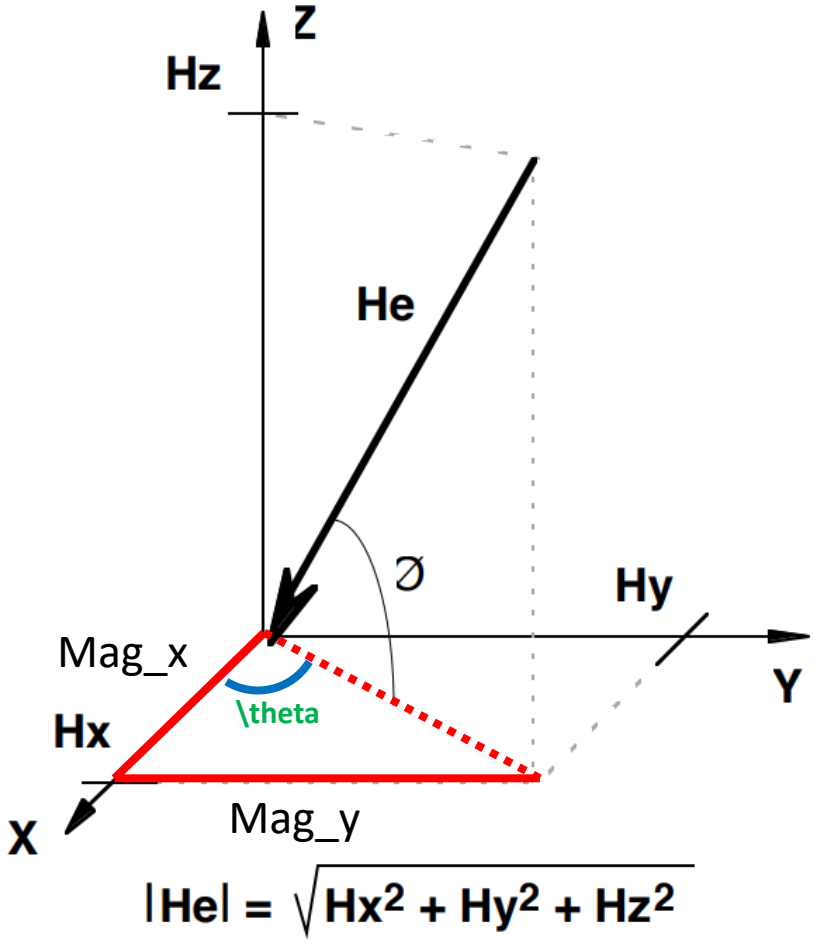


Figure 2 - Earth's Field (He) in 3 Axis

Discussion 1

- 1. Based on the requirements, set correct value in the sample code
  - 8 samples per measurement
  - Data Output Rate = 15Hz
  - Gain=1090(LSb/Gauss)
  - Convert LSB to Gauss (by using self.scale)

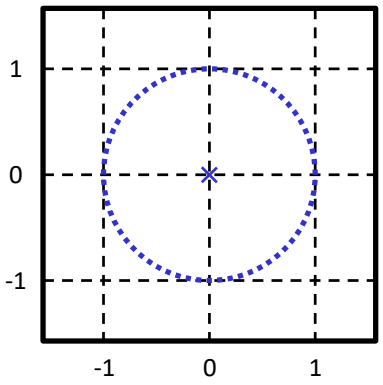
```
self.scale = ?? # convert bit value(LSB) to gauss. DigitalResolution

# Configuration Register A
self.write_byte(HMC5883L_CRA, 0b????????)

# Configuration Register B
self.write_byte(HMC5883L_CRB, 0b????????)

# Mode Register
self.write_byte(HMC5883L_MR, 0b????????)
```

- 2. Continuously measurement(infinite loop)
- 3. Calibrate your sensor (see next page)

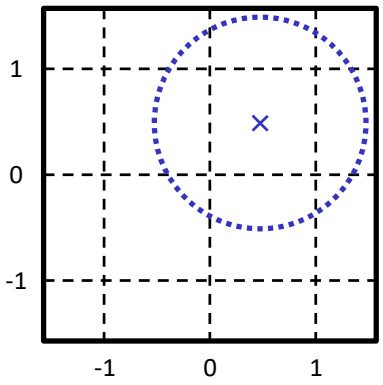
Compass Calibration

- Compass distortion
  - Hard Iron (硬磁干擾: 固定強度的磁干擾物)
  - Soft Iron (軟磁干擾: 會改變強度及方向, 可扭曲磁力線的干擾物)

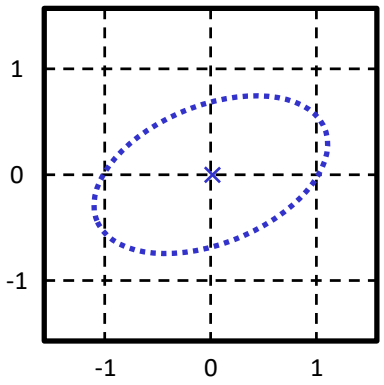
(Perfect)



No distortion

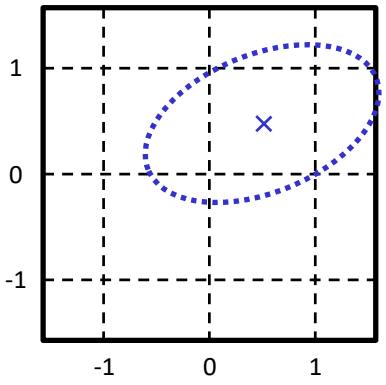


Hard Iron distortion



Soft Iron distortion

(Reality)



Soft & Hard Iron distortion

Compass Calibration

3calibrate-hmc5883l.py and i2cutils.py


```
42  print "Now repeatedly rotate the hmc5883l around all three axes"
43
44  for i in range(0,100):
45      x_out = read_word_2c(3)
46      y_out = read_word_2c(7)
47      z_out = read_word_2c(5)
48
49      if x_out < minx:
50          minx=x_out
51
52      if y_out < miny:
53          miny=y_out
54
55      if z_out < minz:
56          minz=z_out
57
58      if x_out > maxx:
59          maxx=x_out
60
61      if y_out > maxy:
62          maxy=y_out
63
64      if z_out > maxz:
65          maxz=z_out
66
67  print "x offset: ", (maxx + minx) / 2
68  print "y offset: ", (maxy + miny) / 2
69  print "z offset: ", (maxz + minz) / 2
```

```
x: -514 - 387, y: -577 - 0, z: -267 - 777
x: -514 - 387, y: -577 - 0, z: -267 - 777
x: -514 - 387, y: -577 - 0, z: -267 - 777
x: -514 - 387, y: -577 - 0, z: -267 - 777
x: -514 - 387, y: -577 - 0, z: -267 - 777
results:
x: min, max = -514, 387
y: min, max = -577, 0
z: min, max = -267, 777
x offset:  -64
y offset:  -289
z offset:  255
pi@raspberrypi:~/gy801$
```

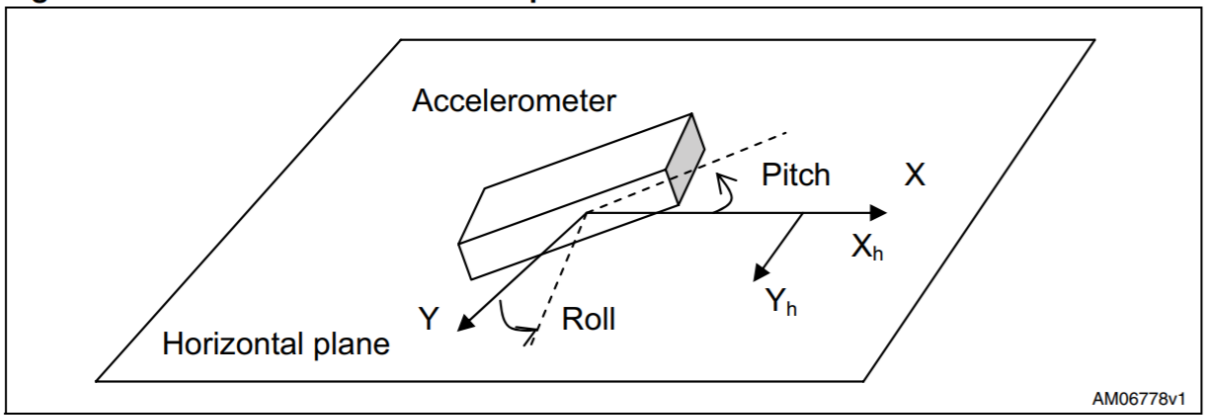
```
184     def __init__(self) :
185         #Class Properties
186         self.X = None
187         self.Y = None
188         self.Z = None
189         self.angle = None
190         self.Xoffset = 0
191         self.Yoffset = 0
192         self.Zoffset = 0
```

Quiz 1

- Tilt-compensate compass

If the handheld device is tilted, then the pitch and roll angles are not equal to 0°, where the pitch and roll can be measured by a 3-axis accelerometer. Therefore, the magnetic sensor measurements XM, YM, and ZM need to be compensated to obtain Xh and Yh.

Figure 3. Handheld device at tilted position



Equation 2

$$X_h = X_M \cos\text{Pitch} + Z_M \sin\text{Pitch}$$
$$Y_h = X_M \sin\text{Roll}\sin\text{Pitch} + Y_M \cos\text{Roll} - Z_M \sin\text{Roll}\cos\text{Pitch}$$

Where, X_M, Y_M, and Z_M are magnetic sensor measurements.

Quiz 1

$$CF\_pitch[t] = (CF\_pitch[t-1] + Xangle_{gyro}[t]) * 0.98 + pitch_{acc}[t] * 0.02$$
$$CF\_roll[t] = (CF\_roll[t-1] + Yangle_{gyro}[t]) * 0.98 + roll_{acc}[t] * 0.02$$

- Tilt-compensate compass
  1. Calibrate your compass
  2. Get **Roll, Pitch** from accelerometer & gyroscope
  3. **Compensated** magnetic sensor measurements

     $X_h = X_M \cos(pitch) + Z_M \sin(pitch)$

     $Y_h = X_M \sin(roll)\sin(pitch) + Y_M \cos(roll) - Z_M \sin(roll)\cos(pitch)$

  4. Calculate tilt-compensated **heading**

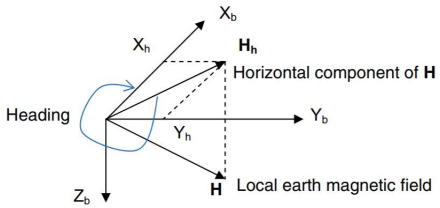
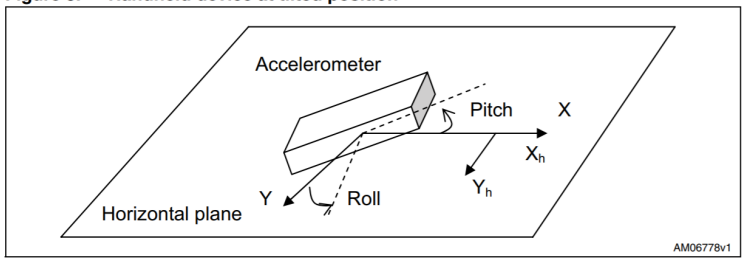


Figure 3. Handheld device at tilted position



Equation 2

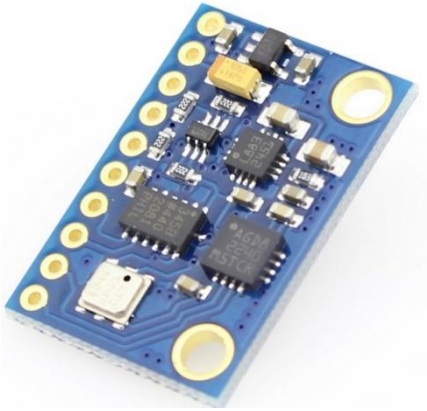
$$X_h = X_M \cos Pitch + Z_M \sin Pitch$$
$$Y_h = X_M \sin Roll \sin Pitch + Y_M \cos Roll - Z_M \sin Roll \cos Pitch$$

Where, X_M, Y_M, and Z_M are magnetic sensor measurements.

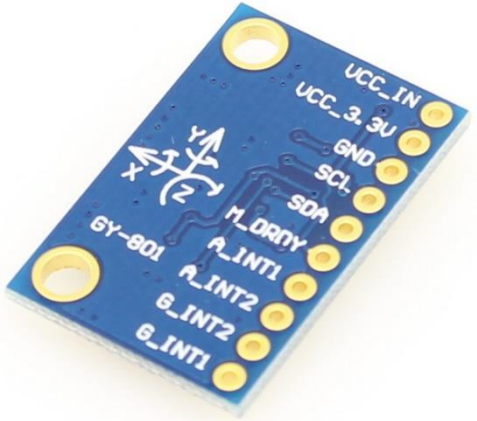
$$\operatorname{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$

Outline

- 嵌入式應用: 人體活動偵測
  - 加速度、陀螺儀...等

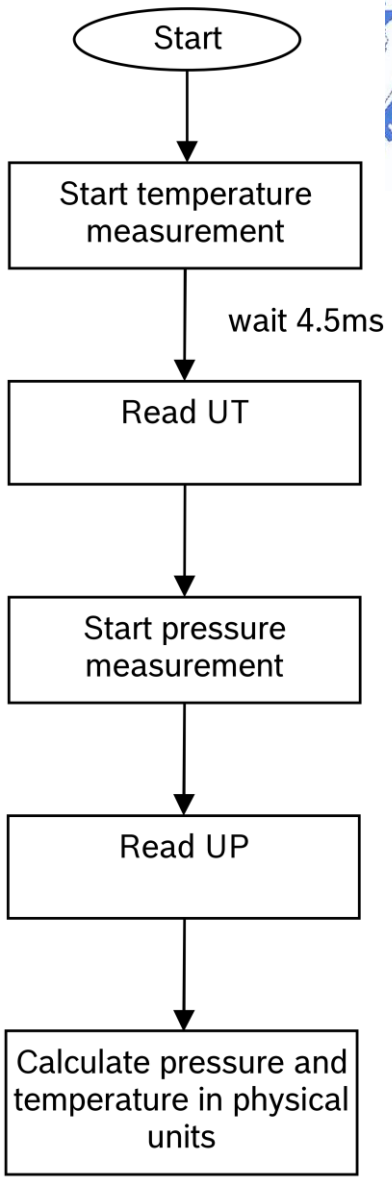


- GY801 (I2C sensor)
  1. 3-axis Accelerometer, Gyroscope, magnetometer and pressure
  2. ADXL345：Accelerometer
  3. L3G4200：Gyroscope
  4. HMC5883：Magnetometer
  5. BMP085：Pressure



4. Altitude (BMP085)

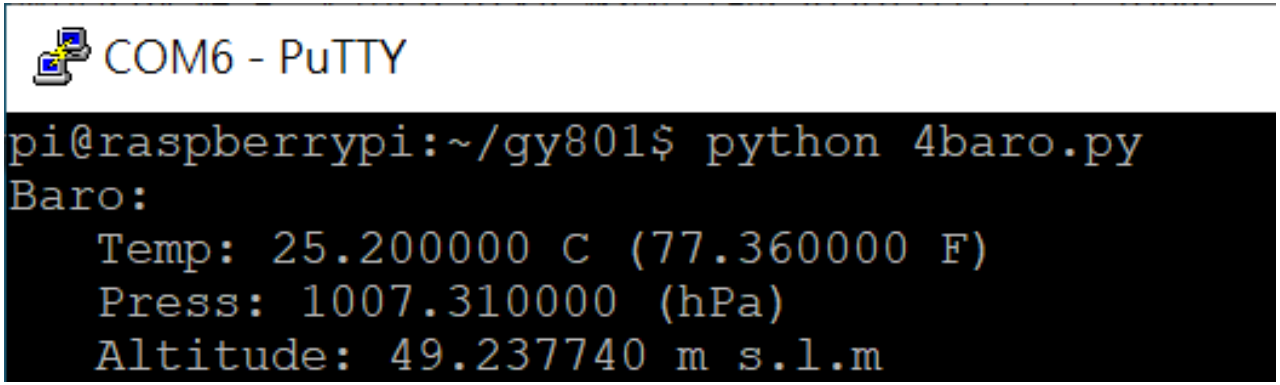
- Pressure sensing range: 300-1100 hPa
  - (9000m to -500m above sea level)
- Up to 0.03hPa / 0.25m resolution
- -40 to +85°C operational range
- +-2°C temperature accuracy
  - Temperature measurement included
- 2-pin i2c interface on chip



https://www.sparkfun.com/datasheets/Components/General/BST-BMP085-DS000-05.pdf
https://www.sparkfun.com/tutorials/253

4. Altitude (BMP085)

- Sample code



4. Barometer code: define address

```
 9  #BMP180 (Barometer) constants
10  BMP180_ADDRESS              = 0x77
11
12  # Calibration coefficients
13  BMP180_AC1                  = 0xAA
14  BMP180_AC2                  = 0xAC
15  BMP180_AC3                  = 0xAE
16  BMP180_AC4                  = 0xB0
17  BMP180_AC5                  = 0xB2
18  BMP180_AC6                  = 0xB4
19  BMP180_B1                  = 0xB6
20  BMP180_B2                  = 0xB8
21  BMP180_MB                  = 0xBA
22  BMP180_MC                  = 0xBC
23  BMP180_MD                  = 0xBE
```

	BMP085 reg adr	
Parameter	MSB	LSB
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

4. Barometer code: read data

```
71      # read calibration data
72      def _read_calibratio_params(self) :
73          self.ac1_val = self.read_word_2c(BMP180_AC1,0)
74          self.ac2_val = self.read_word_2c(BMP180_AC2,0)
75          self.ac3_val = self.read_word_2c(BMP180_AC3,0)
76          self.ac4_val = self.read_word(BMP180_AC4,0)
77          self.ac5_val = self.read_word(BMP180_AC5,0)
78          self.ac6_val = self.read_word(BMP180_AC6,0)
79          self.b1_val = self.read_word_2c(BMP180_B1,0)
80          self.b2_val = self.read_word_2c(BMP180_B2,0)
81          self.mc_val = self.read_word_2c(BMP180_MC,0)
82          self.md_val = self.read_word_2c(BMP180_MD,0)
83
84      # read uncompensated temperature value
85      def getTempC(self) :
86          # print ("Calculating temperature...")
87          self.write_byte(0xF4, 0x2E)
88          time.sleep(0.005)
89
90          ut = self.read_word(0xF6,0)
91
92          # calculate true temperature
93          x1 = ((ut - self.ac6_val) * self.ac5_val) >> 15
94          x2 = (self.mc_val << 11) // (x1 + self.md_val)
95          b5 = x1 + x2
96          self.tempC = ((b5 + 8) >> 4) / 10.0
97
98          return self.tempC
```

Read calibration data from the E^2PROM of the BMP085	
read out E^2PROM registers, 16 bit, MSB first	
AC1 (0xAA, 0xAB)	(16 bit)
AC2 (0xAC, 0xAD)	(16 bit)
AC3 (0xAE, 0xAF)	(16 bit)
AC4 (0xB0, 0xB1)	(16 bit)
AC5 (0xB2, 0xB3)	(16 bit)
AC6 (0xB4, 0xB5)	(16 bit)
B1 (0xB6, 0xB7)	(16 bit)
B2 (0xB8, 0xB9)	(16 bit)
MB (0xBa, 0xBB)	(16 bit)
MC (0xBC, 0xBD)	(16 bit)
MD (0xBE, 0xBF)	(16 bit)

read uncompensated temperature value
write 0x2E into reg 0xF4, wait 4.5ms
read reg 0xF6 (MSB), 0xF7 (LSB)
UT = MSB << 8 + LSB

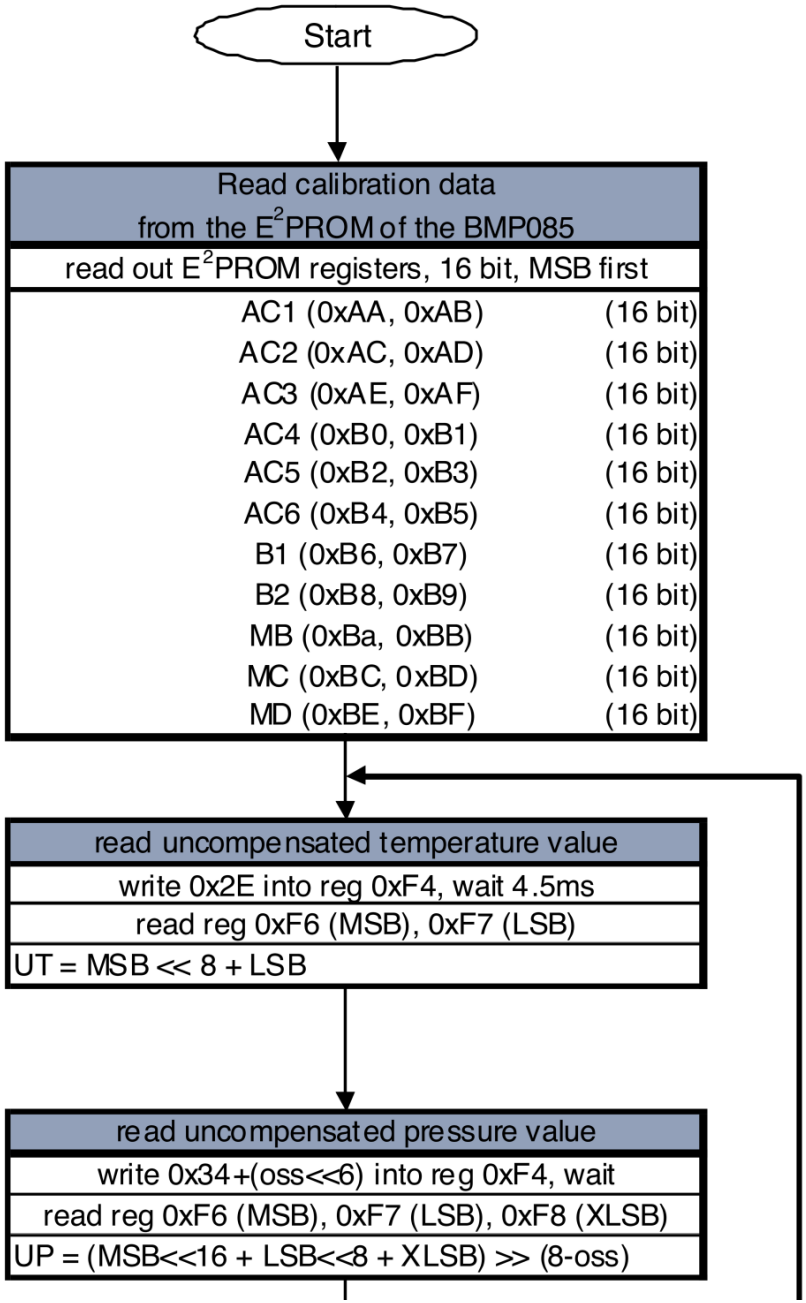
calculate true temperature
$X1 = (UT - AC6) * AC5 / 2^{15}$
$X2 = MC * 2^{11} / (X1 + MD)$
B5 = X1 + X2
$T = (B5 + 8) / 2^4$

4. Barometer code: read data

```
106     # read uncompensated pressure value
107     def getPress(self) :
108         # print ("Calculating temperature...")
109         self.write_byte(0xF4, 0x2E)
110         time.sleep(0.005)
111
112         ut = self.read_word(0xF6,0)
113
114         x1 = ((ut - self.ac6_val) * self.ac5_val) >> 15
115         x2 = (self.mc_val << 11) // (x1 + self.md_val)
116         b5 = x1 + x2
117
118         #print ("Calculating pressure...")
119         self.write_byte(0xF4, 0x34 + (self.oversampling << 6))
120         time.sleep(0.04)
121
122         msb = self.read_byte(0xF6)
123         lsb = self.read_byte(0xF7)
124         xsb = self.read_byte(0xF8)
125
126         up = ((msb << 16) + (lsb << 8) + xsb) >> (8 - self.oversampling)
127
128         # calculate true pressure
129         b6 = b5 - 4000
130         b62 = b6 * b6 >> 12
131         x1 = (self.b2_val * b62) >> 11
132         x2 = self.ac2_val * b6 >> 11
133         x3 = x1 + x2
134         b3 = (((self.ac1_val * 4 + x3) << self.oversampling) + 2) >> 2
135
136         x1 = self.ac3_val * b6 >> 13
137         x2 = (self.b1_val * b62) >> 16
138         x3 = ((x1 + x2) + 2) >> 2
139         b4 = (self.ac4_val * (x3 + 32768)) >> 15
140         b7 = (up - b3) * (50000 >> self.oversampling)
141
142         press = (b7 * 2) // b4
143         #press = (b7 / b4) * 2
144
145         x1 = (press >> 8) * (press >> 8)
146         x1 = (x1 * 3038) >> 16
147         x2 = (-7357 * press) >> 16
148         self.press = ( press + ((x1 + x2 + 3791) >> 4) ) / 100.0
149
150         return self.press
```

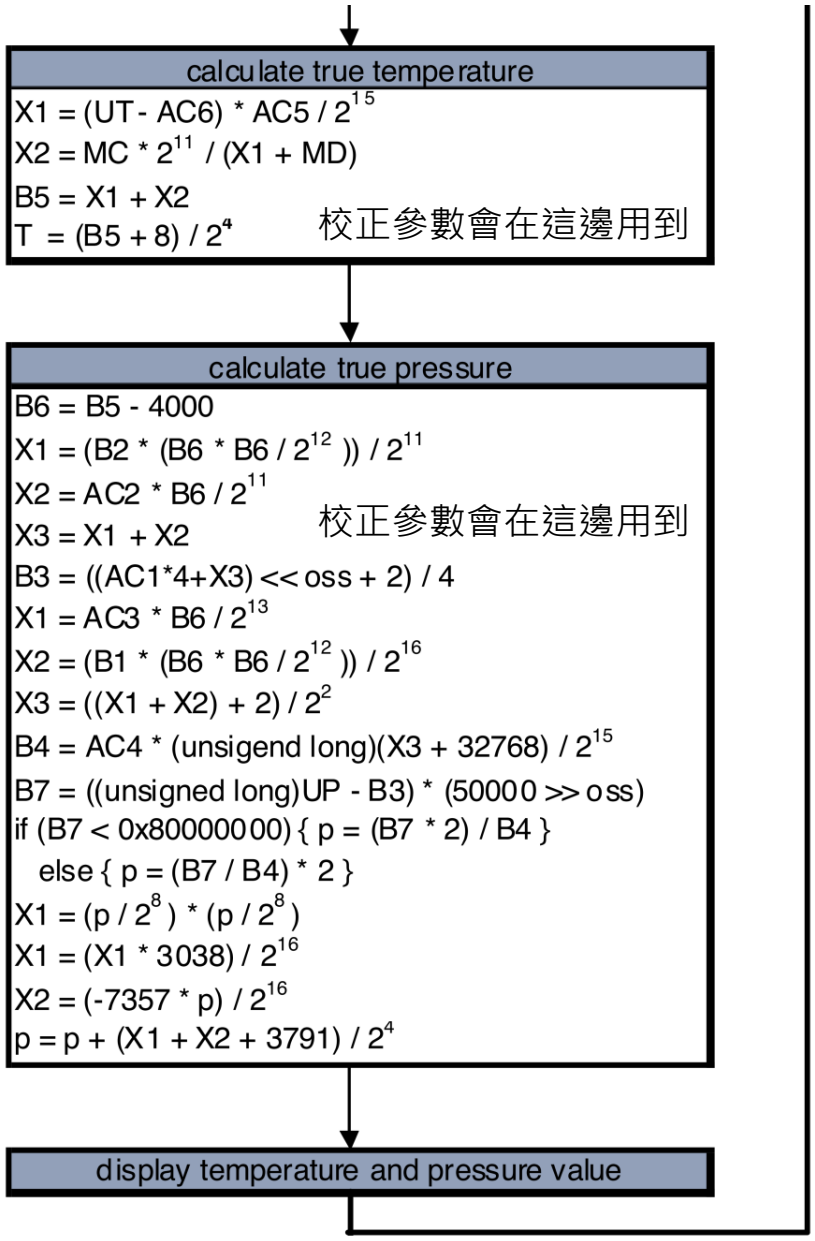
read uncompensated pressure value
write 0x34+(oss<<6) into reg 0xF4, wait
read reg 0xF6 (MSB), 0xF7 (LSB), 0xF8 (XLSB)
UP = (MSB<<16 + LSB<<8 + XLSB) >> (8-oss)

calculate true pressure
B6 = B5 - 4000
$X1 = (B2 * (B6 * B6 / 2^{12})) / 2^{11}$
$X2 = AC2 * B6 / 2^{11}$
X3 = X1 + X2
B3 = ((AC1*4+X3) << oss + 2) / 4
$X1 = AC3 * B6 / 2^{13}$
$X2 = (B1 * (B6 * B6 / 2^{12})) / 2^{16}$
$X3 = ((X1 + X2) + 2) / 2^{2}$
$B4 = AC4 * (unsigend\ long)(X3 + 32768) / 2^{15}$
B7 = ((unsigned long)UP - B3) * (50000 >> oss)
if (B7 < 0x80000000) { p = (B7 * 2) / B4 }
else { p = (B7 / B4) * 2 }
$X1 = (p / 2^{8}) * (p / 2^{8})$
$X1 = (X1 * 3038) / 2^{16}$
$X2 = (-7357 * p) / 2^{16}$
$p = p + (X1 + X2 + 3791) / 2^{4}$



	example:	C code function:	type:
		bmp085_get_cal_param	
AC1 =	408		short
AC2 =	-72		short
AC3 =	-14383		short
AC4 =	32741		unsigned short
AC5 =	32757		unsigned short
AC6 =	23153		unsigned short
B1 =	6190		short
B2 =	4		short
MB =	-32768		short
MC =	-8711		short
MD =	2868		short
		bmp085_get_ut	
UT =	**27898**		long
oss =	0		short (0 .. 3)
= oversampling_setting			
(ultra low power mode)		bmp085_get_up	
UP =	**23843**		long

Measurement flowchart



bmp085_get_temperature

X1 =	4743		long
X2 =	-2344		long
B5 =	2399		long
T =	**150**	***temp in 0.1℃***	long

BMP085_calpressure

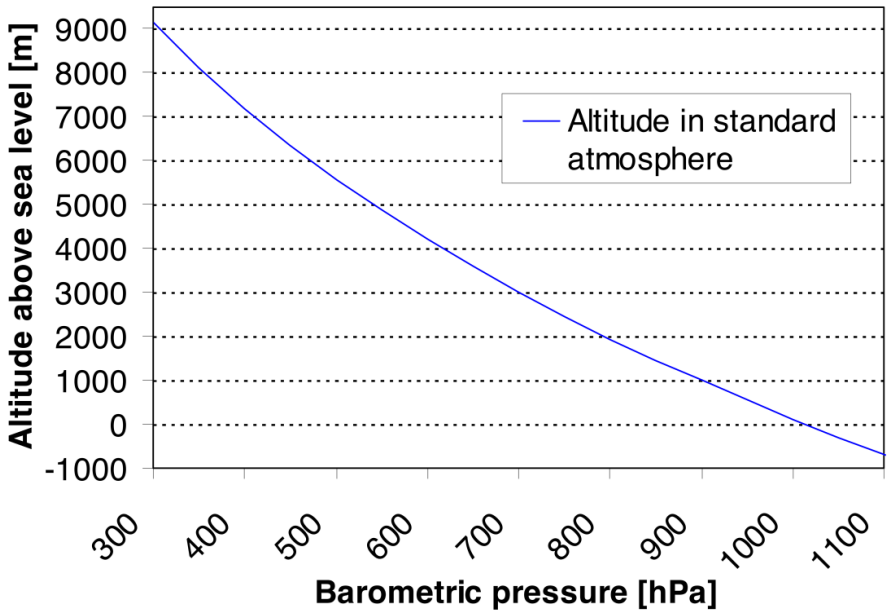
B6 =	-1601		long
X1 =	1		long
X2 =	56		long
X3 =	57		long
B3 =	422		long
X1 =	2810		long
X2 =	59		long
X3 =	717		long
B4 =	33457		unsigned long
B7 =	1171050000		unsigned long
p =	70003		long
			long
X1 =	74529		long
X1 =	3454		long
X2 =	-7859		long
p =	**69964**	***press. in Pa***	long

Measurement flowchart

4. Barometer code: calculate altitude

```
152     # calculate absolute altitude
153     def getAltitude(self) :
154         #    print ("Calculating altitude...")
155         self.altitude = 44330 * (1 - ((self.getPress() / STANDARD_PRESSURE) ** 0.1903))
156         return self.altitude
```

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



4. Barometer code: set operation mode

Control registers

read uncompensated temperature value
write 0x2E into reg 0xF4, wait 4.5ms
read reg 0xF6 (MSB), 0xF7 (LSB)
UT = MSB << 8 + LSB

read uncompensated pressure value
write 0x34+(oss<<6) into reg 0xF4, wait
read reg 0xF6 (MSB), 0xF7 (LSB), 0xF8 (XLSB)
UP = (MSB<<16 + LSB<<8 + XLSB) >> (8-oss)

Measurement	Control register value (register address 0xF4)	Max. conversion time [ms]
Temperature	0x2E	4.5
Pressure (osrs = 0)	0x34	4.5
Pressure (osrs = 1)	0x74	7.5
Pressure (osrs = 2)	0xB4	13.5
Pressure (osrs = 3)	0xF4	25.5

Overview of BMP085 modes, selected by driver software via the variable *oversampling_setting*:

Mode	Parameter *oversampling_setting*	Internal number of samples	Conversion time pressure max. [ms]	Avg. current @ 1 sample/s typ. [µA]	RMS noise typ. [hPa]	RMS noise typ. [m]
ultra low power	0	1	4.5	3	0.06	0.5
standard	1	2	7.5	5	0.05	0.4
high resolution	2	4	13.5	7	0.04	0.3
ultra high resolution	3	8	25.5	12	0.03	0.25

Discussion 2

- 1. Based on the datasheet, set correct value in the sample code
  - Standard mode

read uncompensated pressure value
write 0x34+(oss<<6) into reg 0xF4, wait
read reg 0xF6 (MSB), 0xF7 (LSB), 0xF8 (XLSB)
UP = (MSB<<16 + LSB<<8 + XLSB) >> (8-oss)

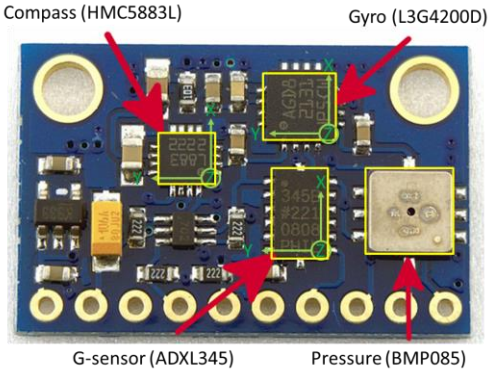
read uncompensated temperature value
write 0x2E into reg 0xF4, wait 4.5ms
read reg 0xF6 (MSB), 0xF7 (LSB)
UT = MSB << 8 + LSB

```
def getPress(self) :
    # print ("Calculating temperature...")
    self.write_byte(0xF4, 0x??)
    time.sleep(0.005)
```

```
# read uncompensated temperature value
def getTempC(self) :
    # print ("Calculating temperature...")
    self.write_byte(0xF4, 0x??)
    time.sleep(0.005)
```

- 2. Continuously measurement(infinite loop)

GY801 summary



1. Accelerometer: Acceleration(加速度) -> Roll, Pitch (傾斜姿態)
2. Gyroscope: Angular velocity(角速度) -> Rotated angle (旋轉角度)
3. Magnetometer: Magnetic field(磁場) -> Heading (方位)
4. Barometer: Pressure(大氣壓力) -> Altitude (高度)

Sensor Fusion:

- Accelerometer + Gyroscope: Enhanced Rotation angle
- Accelerometer + Gyroscope + Magnetometer: Tilt-compensate compass

Quiz 2

- A drone is flying, it needs the following information:

  Roll: xxx, Pitch: xxx, Heading: xxx, Altitude: xxx
  (unit: Degree and Meter)

- Hint: Combine the above code to achieve this
  - Roll, Pitch: Accelerometer + Gyroscope
  - Heading: Accelerometer + Gyroscope + Magnetometer
  - Altitude: Barometer

Summary

- Practice Lab (magnetometer, pressure and filter)
- Write down the answer for discussion
  - **Discussion 1&2:**
    - **Based on the requirements, set correct value in the sample code**
    - (放code截圖)
    - Deadline: Before 11:59, 4/17
- Write code for Quiz 1 - 2, then demonstrate it to TAs
  - **Quiz1: Tilt-compensate compass**
  - **Quiz2: Drone's information**
    - Deadline: Before 15:10, 4/10
    - Late Demo: Before 15:10, 4/17