



Outline

日期	(週數) 主題
2/22	1. 嵌入式應用介紹
3/8	3. 樹莓派介紹與設定
3/15	4. 樹莓派應用(倒車雷達)
3/22	5. 樹莓派應用(人體活動偵測)
3/29	6. 樹莓派應用(人體活動偵測)
4/12	8. 網路攝影機 IP cam
4/19	9. 網路攝影機 + 影像辨識
4/26	10. 網路攝影機 + 機器學習影像辨識
5/3	11. Midterm (Project分組)
5/10	12. 語音助理. Google assistant
5/17	13. 網路應用、推播廣告(beacon)
5/24	14. 其他嵌入式系統
5/31	15. Final Project – Proposal
6/14	17. Final Project prepare, Q&A, 補demo
6/21,28	18,19. Final Project demonstration (陽明, 交大) TBD



Last week

- 1. 領取GY801感測器
- 2. 連接I2C sensor
- 3. 加速度計、陀螺儀的運作原理
- 4. 寫Python程式讀取數值並分析

This week

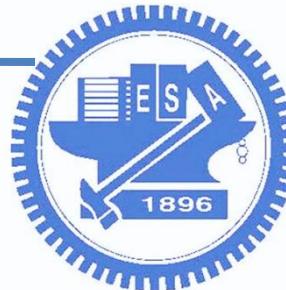
- 1. 介紹電子羅盤、氣壓計
- 2. 寫Python程式讀取數值並分析



嵌入式系統設計概論與實作

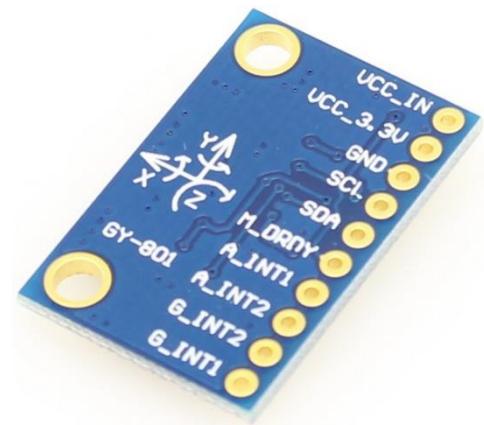
曾煜棋、吳昆儒、張凌燕

National Chiao Tung University



Outline

- 嵌入式應用: 人體活動偵測
 - 加速度、陀螺儀、電子羅盤、氣壓計...等
 - 感測數值分析
- GY801 (I2C sensor)
 - 3-axis Accelerometer, Gyroscope, magnetometer and pressure
 - ① ADXL345 : Accelerometer
 - ② L3G4200 : Gyroscope
 - ③ HMC5883 : Magnetometer
 - ④ BMP085 : Pressure

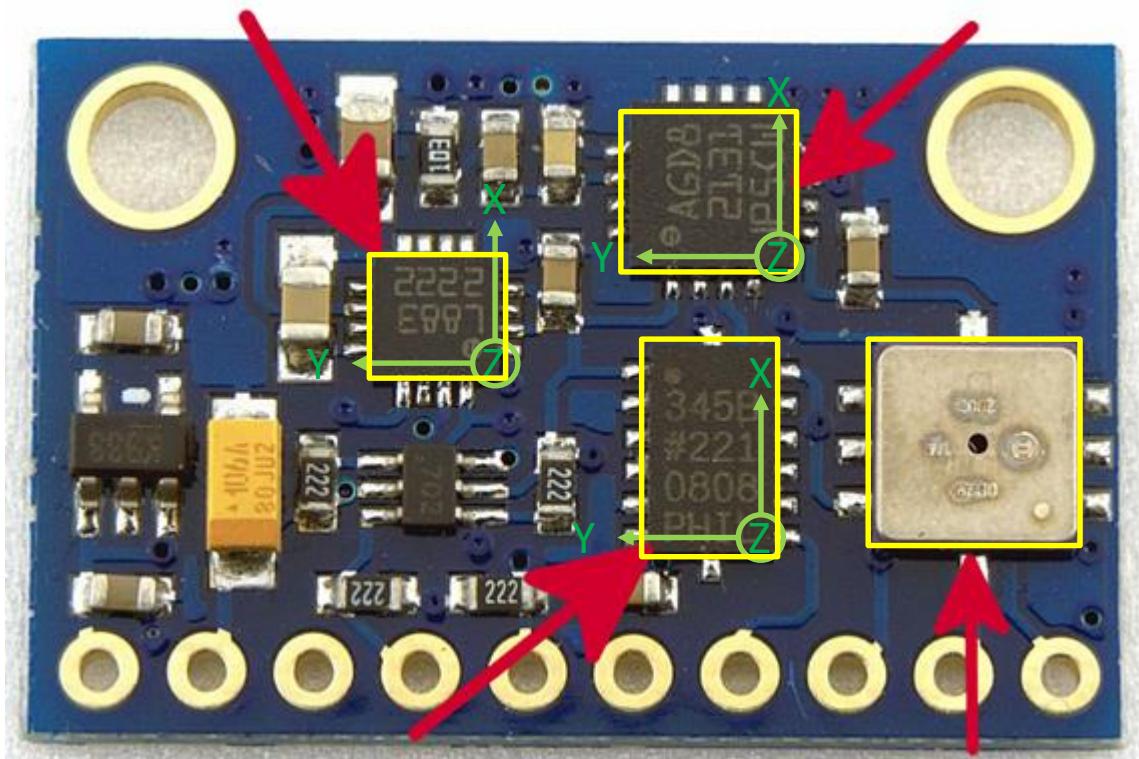




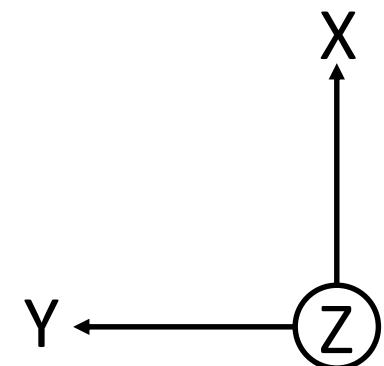
GY-801 (10 DoF sensor)

Compass (HMC5883L)

Gyro (L3G4200D)

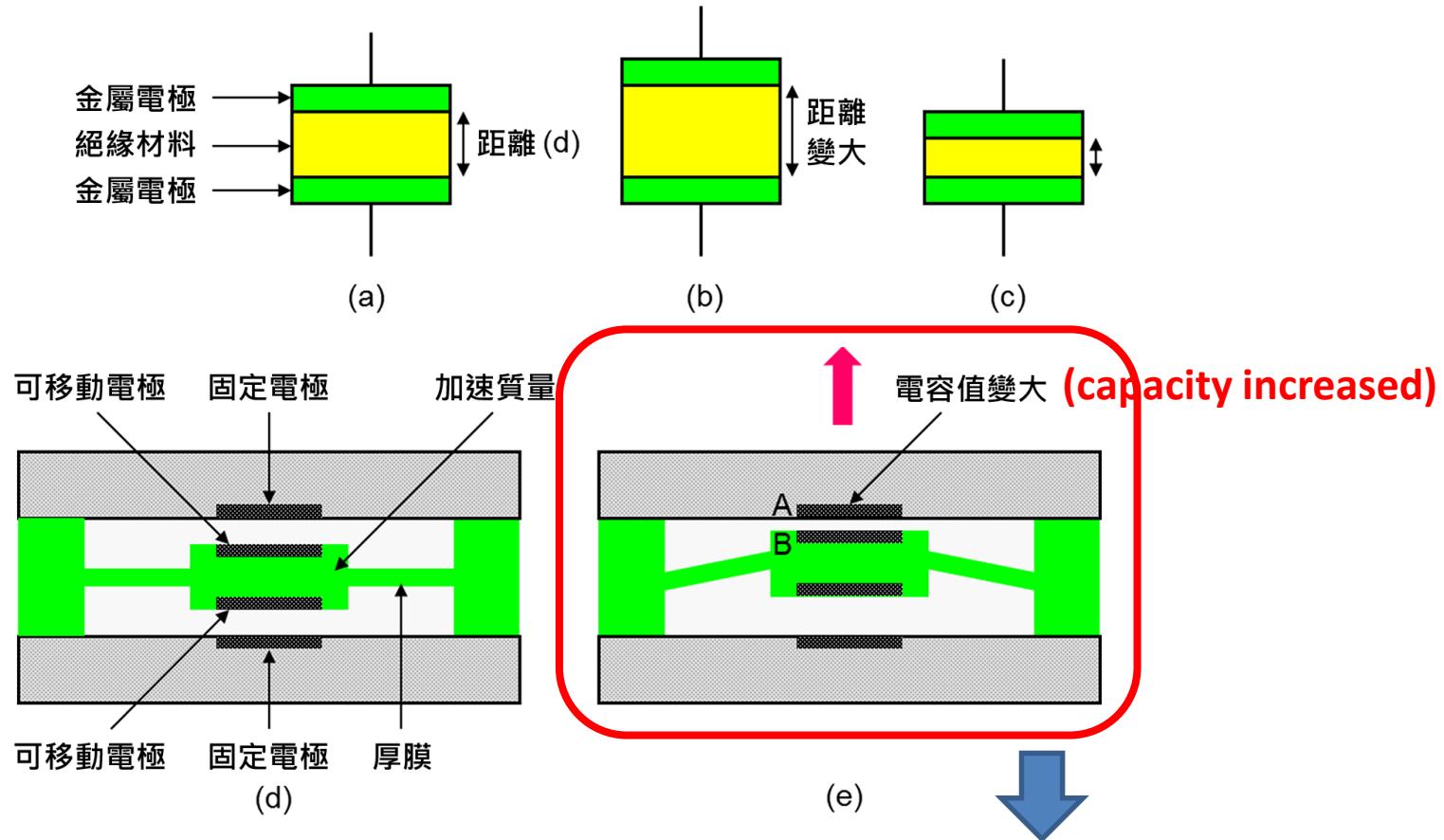


Pressure (BMP085)





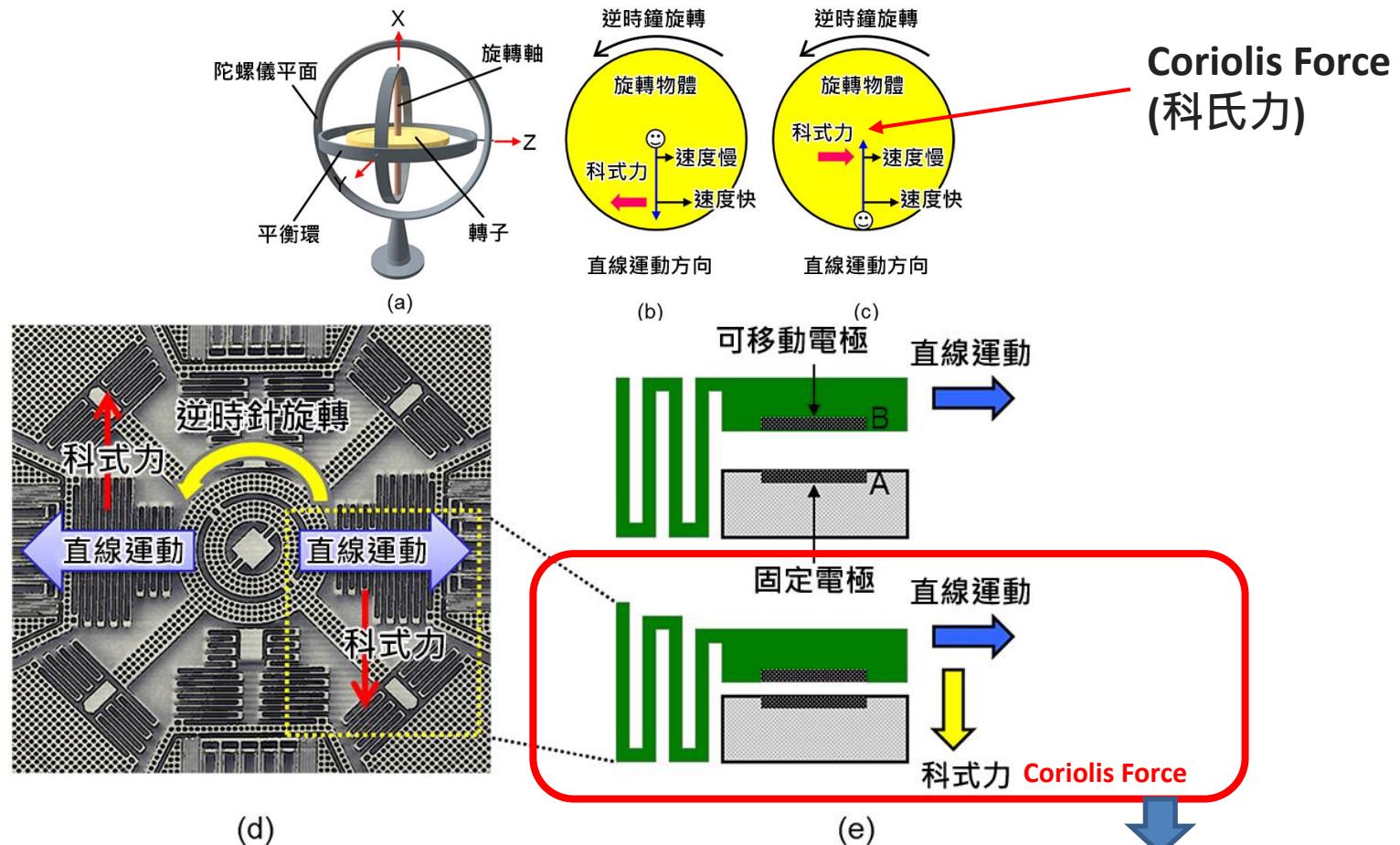
Sensor - Accelerometer



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



Sensor - Gyro



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

Sensor - Magnetomet

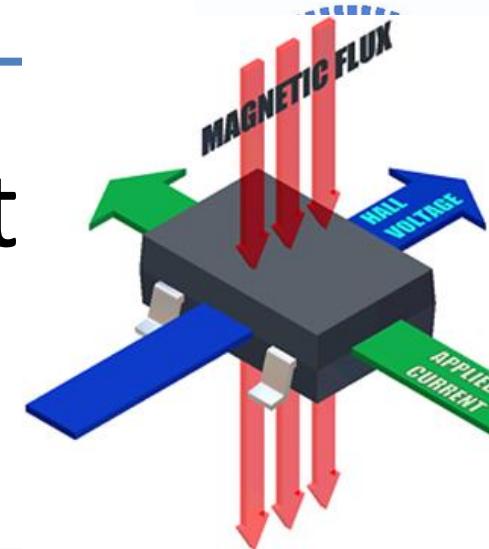
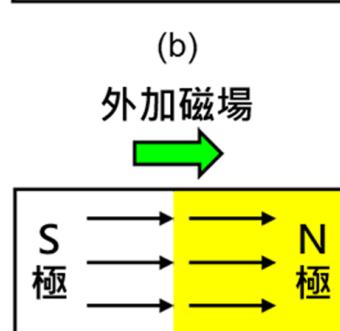
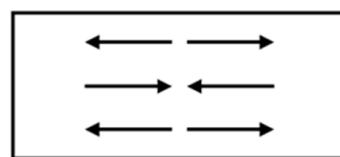
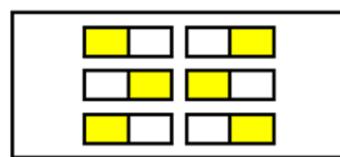
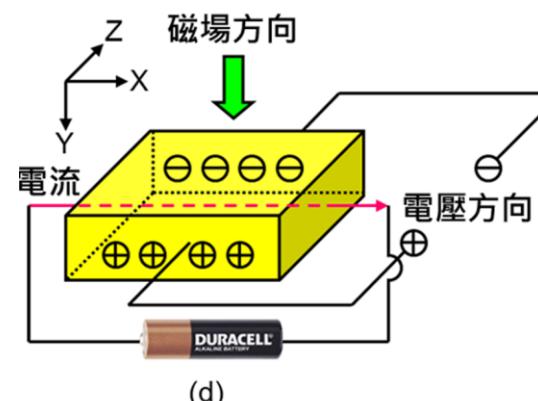


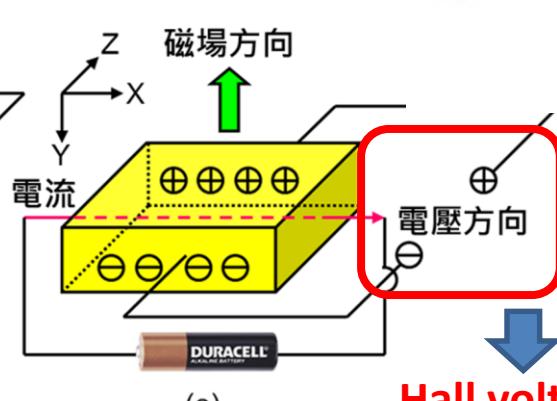
Fig: <https://www.ecnmag.com/article/2009/03/sensor-zone-april-2009>



(a)

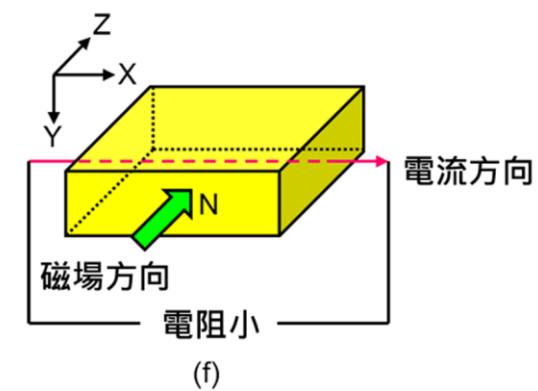


(d)

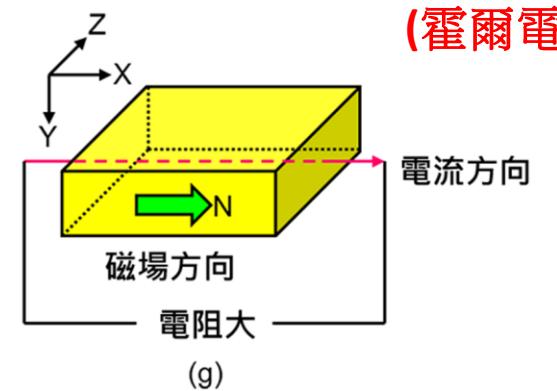


(e)

Hall voltage
(霍爾電壓)



(f)

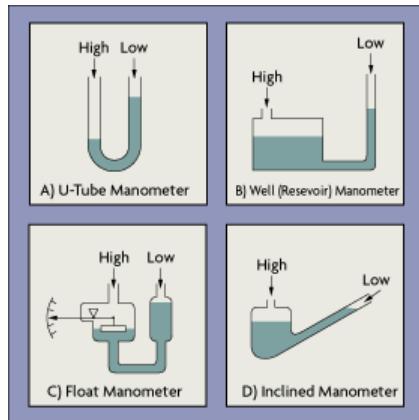


(g)

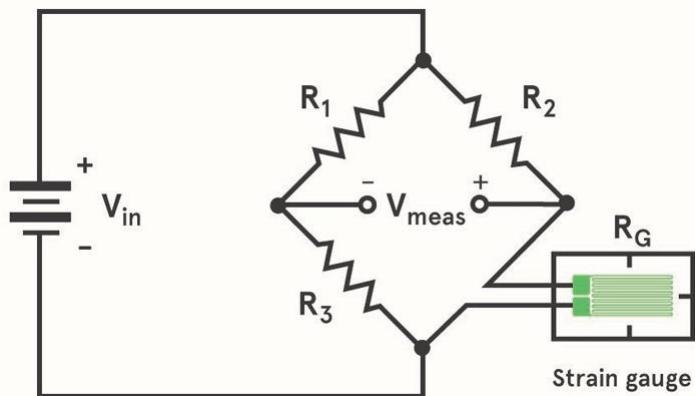
Measure the absolute **magnetic intensity** based on **Hall voltage** (霍爾電壓)



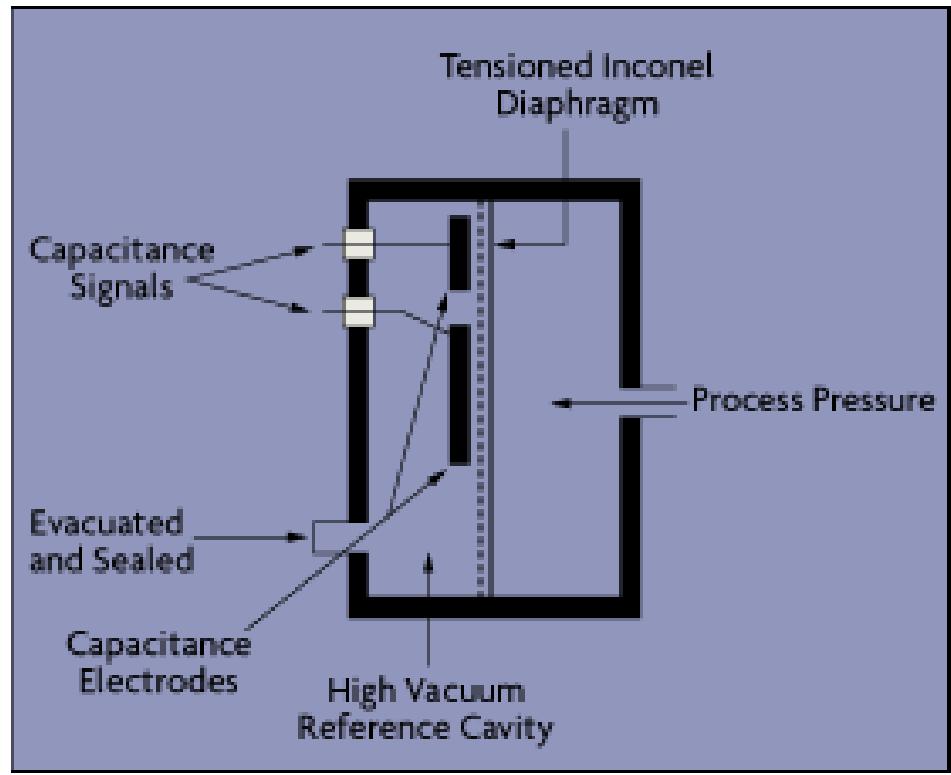
Sensor - Pressure



Manometer

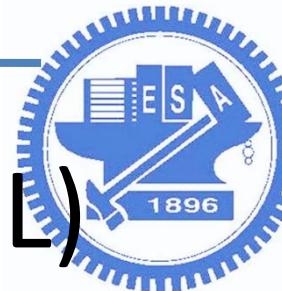


Use Wheatstone bridge to measure the resistance.



Capacitance Vacuum Manometer

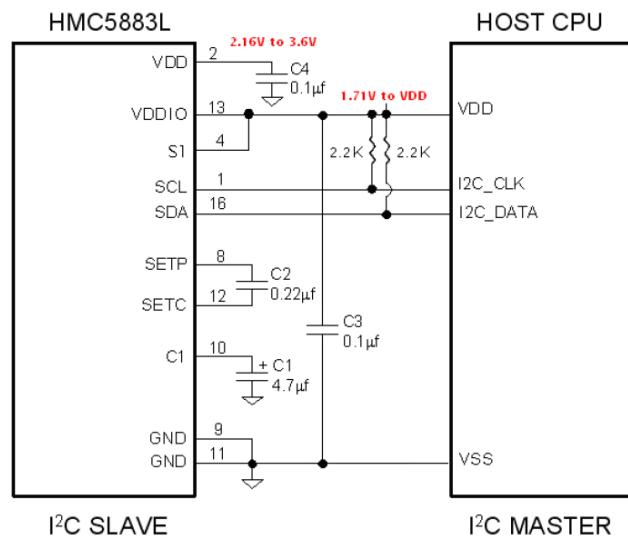
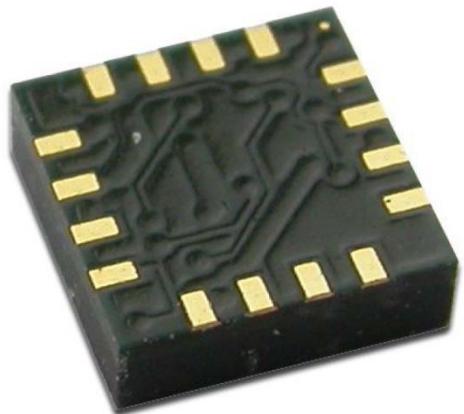
Measure the **varying capacity** based on **pressure**.

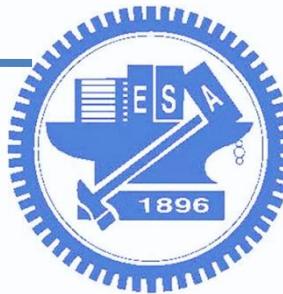


3. Magnetometer (HMC5883L)

□ 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 code

- Sample code:

```
wget https://goo.gl/c6iD08 -O compass.py  
python compass.py
```

- Execution

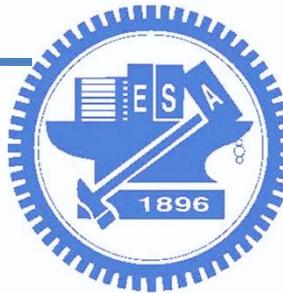
```
pi@raspberrypi:~$ python compass.py  
-148.12 , -38.64 , -348.68  
-147.2 , -37.72 , -348.68  
-147.2 , -37.72 , -348.68  
-148.12 , -37.72 , -347.76  
-148.12 , -37.72 , -348.68  
-149.04 , -38.64 , -347.76  
-148.12 , -37.72 , -347.76  
-149.04 , -37.72 , -347.76  
-148.12 , -37.72 , -347.76  
-148.12 , -37.72 , -347.76
```



3. Magnetometer code

```
1 import sys, os, math, time, thread, smbus, random, requests
2
3 bus = smbus.SMBus(1)
4 addrHMC = 0x1e
5
6 def read_word(address, adr):          Read register value
7     high = bus.read_byte_data(address, adr)
8     low = bus.read_byte_data(address, adr + 1)
9     val = (high << 8) + low
10    return val
11
12 def read_word_2c(address, adr):        2's complement
13     val = read_word(address, adr)
14     if (val >= 0x8000):
15         return -((65535 - val) + 1)
16     else:
17         return val
18
19 def main():
20
21     bus.write_byte_data(addrHMC, 0, 0b01110000) # Set to 8 samples @ 15Hz
22     bus.write_byte_data(addrHMC, 1, 0b00100000) # 1.3 gain LSb / Gauss 1090 (default)
23     bus.write_byte_data(addrHMC, 2, 0b00000000) # Continuous sampling
24
25     while True:
26         x = read_word_2c(addrHMC, 3)           Read mx/my/mz
27         y = read_word_2c(addrHMC, 7)
28         z = read_word_2c(addrHMC, 5)
29
30         print x,"",y,"",z
31         time.sleep(0.1)                      Display sensing value
32
33 if __name__ == "__main__":
34     main()
```

Download sample code : [wget https://goo.gl/c6iD08 -O compass.py](https://goo.gl/c6iD08)



3. HMC5883L Datasheet

□ Set register address based on datasheet

Register Access

This section describes the process of reading from and writing to this device. The device uses an address pointer to indicate which register location is to be read from or written to. These pointer locations are sent from the master to this slave device and succeed the 7-bit address (0x1E) plus 1 bit read/write identifier, i.e. 0x3D for read and 0x3C for write.

```
bus = smbus.SMBus(1)
addrHMC = 0x1e

def init_imu():
    # HMC setting
    bus.write_byte_data(addrHMC, 0, 0b01110000) # Set to 8 samples @ 15Hz
    bus.write_byte_data(addrHMC, 1, 0b00100000) # 1.3 gain LSb / Gauss 1090 (default)
    bus.write_byte_data(addrHMC, 2, 0b00000000) # Continuous sampling
```



3. HMC5883L Datasheet

□ Set register address

```
bus.write_byte_data(addrHMC, 0, 0b01110000)  
bus.write_byte_data(addrHMC, 1, 0b00100000)  
bus.write_byte_data(addrHMC, 2, 0b00000000)
```

Address Location	Name	Access
00	Configuration Register A	Read/Write
01	Configuration Register B	Read/Write
02	Mode Register	Read/Write
03	Data Output X MSB Register	Read → <code>read_word_2c(addrHMC, 3)</code>
04	Data Output X LSB Register	Read
05	Data Output Z MSB Register	Read → <code>read_word_2c(addrHMC, 5)</code>
06	Data Output Z LSB Register	Read
07	Data Output Y MSB Register	Read → <code>read_word_2c(addrHMC, 7)</code>
08	Data Output Y LSB Register	Read
09	Status Register	Read
10	Identification Register A	Read
11	Identification Register B	Read
12	Identification Register C	Read

Table2: Register List



3. HMC5883L Datasheet

- Set operation mode (Configuration Register A)

```
bus.write_byte_data(addrHMC, 0, 0b01110000) # Set to 8 samples @ 15Hz
```

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

15 Hz

Table 5: Data Output Rates



3. HMC5883L Datasheet

□ Set operation mode (Configuration Register B)

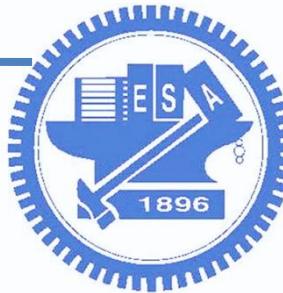
```
bus.write_byte_data(addrHMC, 1, 0b00100000) # 1.3 gain LSb / Gauss 1090 (default)
```

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)

+/- 1.3Ga; Gain 1090



3. HMC5883L Datasheet

□ Set operation mode (Mode Register)

```
bus.write_byte_data(addrHMC, 2, 0b00000000) # Continuous sampling
```

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

Table 10: Mode Register

Continuous
Measurement
Mode

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.

You can also change operation mode based on your need and datasheet.
The code is already set default for you.



3. HMC5883L Datasheet

- In the code, we also obtain raw data (bit value, LSB).
How to convert it to gauss?

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

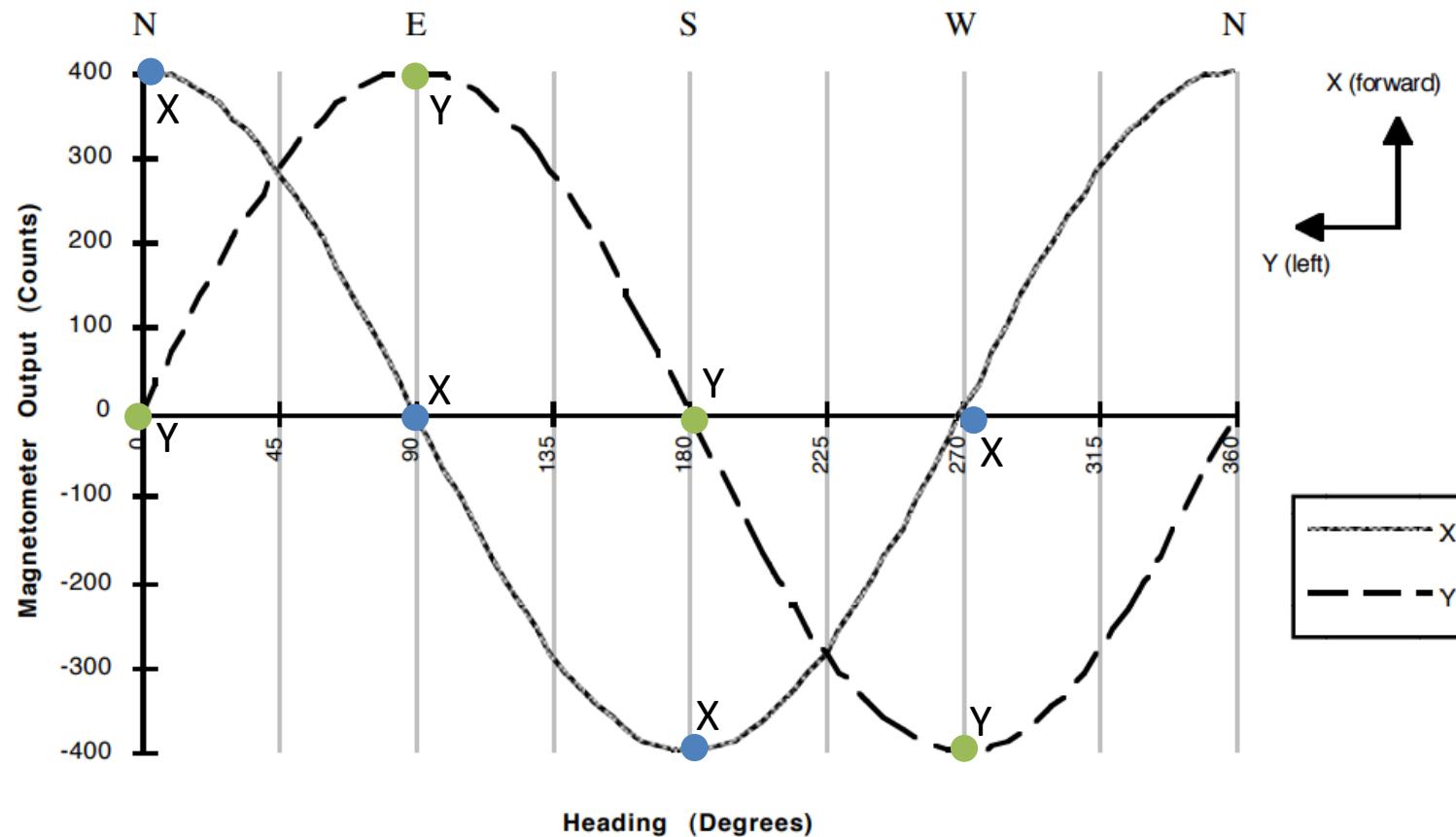
GN2	GN1	GN0	Recommended Sensor Field Range	Gain (LSb/Gauss)	Digital Resolution (mG/LSb)	Output Range
0	0	0	± 0.88 Ga	1370	0.73	0xF800–0x07FF (-2048–2047)
0	0	1	± 1.3 Ga	1090 (default)	0.92	0xF800–0x07FF (-2048–2047)

- When it report -4096, the overflow occurs.

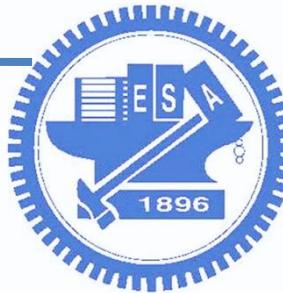
In the event the ADC reading overflows or underflows for the given channel, or if there is a math overflow during the bias measurement, this data register will contain the value -4096. This register value will clear when after the next valid measurement is made.



Magnetometer reading



Hx and Hy Magnetometer Readings for Different Compass Headings

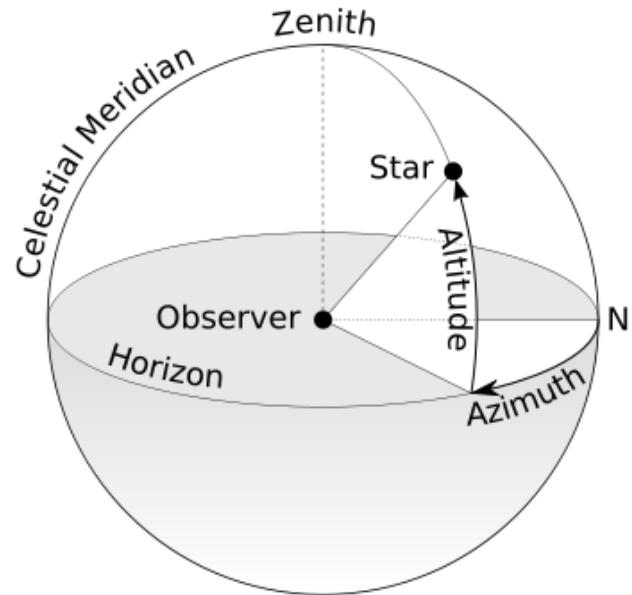


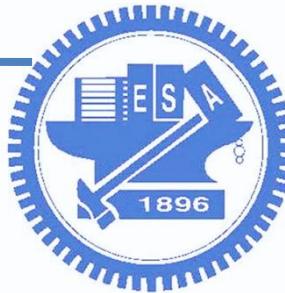
Quiz 1

□ How to estimate the heading?

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

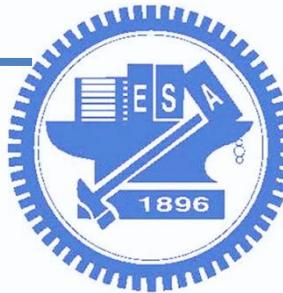
$$\text{atan2}(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$





Mathematical functions

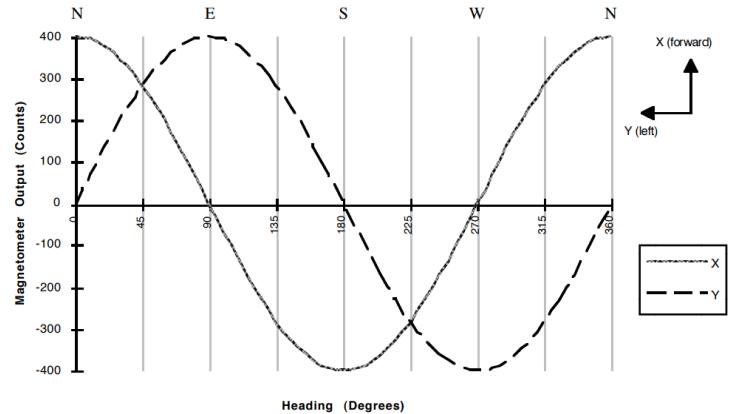
- Trigonometric functions
 - **math.atan(x)**
 - Return the arc tangent of x, in radians.
 - **math.atan2(y, x)**
 - Return $\text{atan}(y / x)$, in radians. The result is between $-\pi$ and π . The vector in the plane from the origin to point (x, y) makes this angle with the positive X axis. The point of $\text{atan2}()$ is that the signs of both inputs are known to it, so it can compute the correct quadrant for the angle. For example, $\text{atan}(1)$ and $\text{atan2}(1, 1)$ are both $\pi/4$, but $\text{atan2}(-1, -1)$ is $-3\pi/4$.
- Angular conversion
 - **math.degrees(x)**
 - Convert angle x from radians to degrees.
- Constant
 - **math.pi**
 - $\pi = 3.141592\dots$

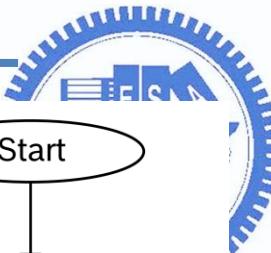


Discussion 1

- Rotate the magnetometer for 360 degree. Record the value for x and y, then **draw it. Write down your observation.**
 - python compass.py > **result.txt** # press **ctrl + c** to stop
 - Copy the result.txt to your PC. You can use Excel or Matlab to draw fig.

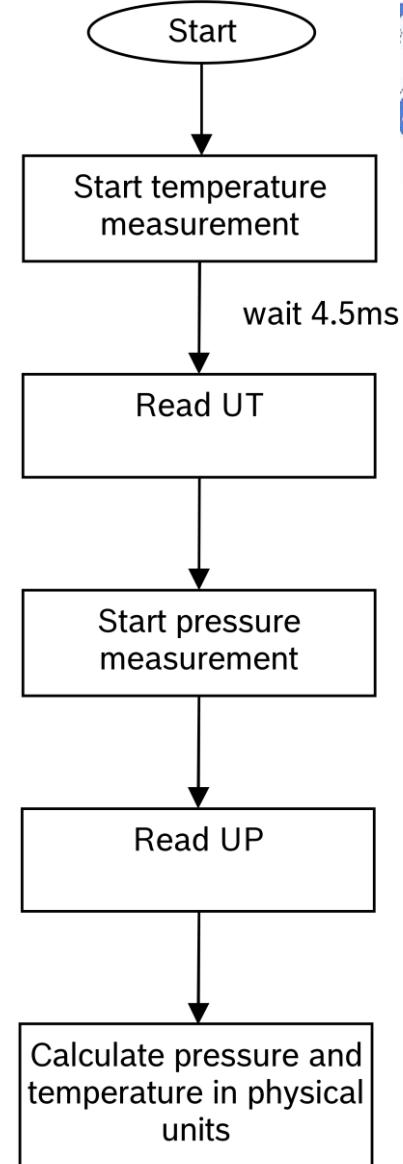
```
(COM8) [80x24]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)
pi@raspberrypi:~$ python compass.py > compass_result.txt
```





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





4. Altitude (BMP085)

- BMP085 sample code:

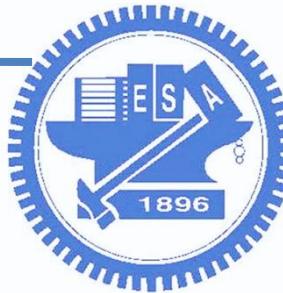
- https://github.com/adafruit/Adafruit_Python_BMP

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

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

- Execution

```
pi@raspberrypi:~/gy80/4.bmp/examples$ sudo python simpletest.py  
Temp = 25.50 *C  
Pressure = 100536.00 Pa  
Altitude = 65.90 m  
Sealevel Pressure = 100536.00 Pa
```



4. Altitude code (BMP085)

- Source code: examples/simpletest.py

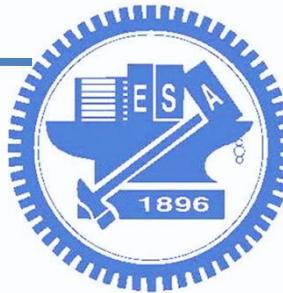
```
#!/usr/bin/python
import Adafruit_BMP.BMP085 as BMP085
sensor = BMP085.BMP085()

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()))
```

Load library

Print sensing value

More information: Adafruit_BMP/BMP085.py



4. BMP085 Datasheet

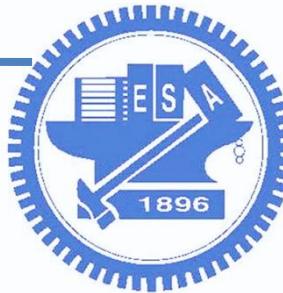
- Define address in Adafruit_BMP/BMP085.py

```
# BMP085 default address.  
BMP085_I2CADDR      = 0x77  
  
# Operating Modes  
BMP085_ULTRALOWPOWER  = 0  
BMP085_STANDARD       = 1  
BMP085_HIGHRES        = 2  
BMP085_ULTRAHIGHRES   = 3
```



The operation mode is set to BMP085_STANDARD. (refer to datasheet)

Mode	Parameter <i>oversampling_setting</i>	Internal number of samples	Conversion time pressure max. [ms]	Avg. current @ 1 sample/s typ. [μ A]	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

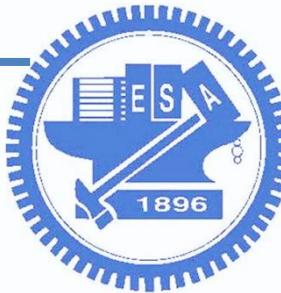


4. BMP085 Datasheet

- Define Calibration coefficients from datasheet

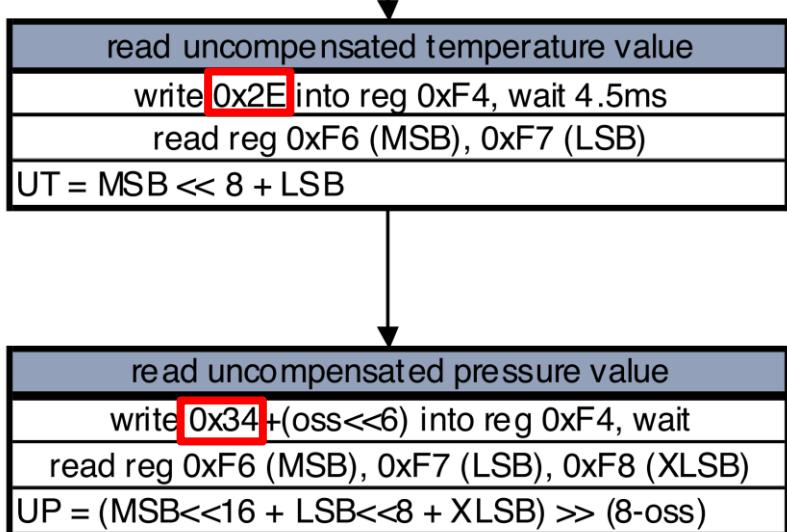
	BMP085 reg adr	
Parameter	MSB	LSB
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

```
# BMP085 Registers
BMP085_CAL_AC1      = 0xAA # R Calibration data (16 bits)
BMP085_CAL_AC2      = 0xAC # R Calibration data (16 bits)
BMP085_CAL_AC3      = 0xAE # R Calibration data (16 bits)
BMP085_CAL_AC4      = 0xB0 # R Calibration data (16 bits)
BMP085_CAL_AC5      = 0xB2 # R Calibration data (16 bits)
BMP085_CAL_AC6      = 0xB4 # R Calibration data (16 bits)
BMP085_CAL_B1       = 0xB6 # R Calibration data (16 bits)
BMP085_CAL_B2       = 0xB8 # R Calibration data (16 bits)
BMP085_CAL_MB       = 0xBA # R Calibration data (16 bits)
BMP085_CAL_MC       = 0xBC # R Calibration data (16 bits)
BMP085_CAL_MD       = 0xBE # R Calibration data (16 bits)
```



4. BMP085 Datasheet

- How to read temperature and pressure referred by datasheet



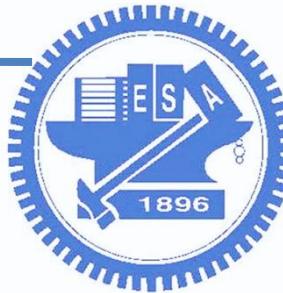
Commands

BMP085_READTEMPCMD = 0x2E
BMP085_READPRESSURECMD = 0x34

BMP085 Registers

BMP085_CONTROL = 0xF4
BMP085_TEMPDATA = 0xF6
BMP085_PRESSUREDATA = 0xF6

Fig. Calculation of pressure and temperature for BMP085



4. BMP085 Datasheet

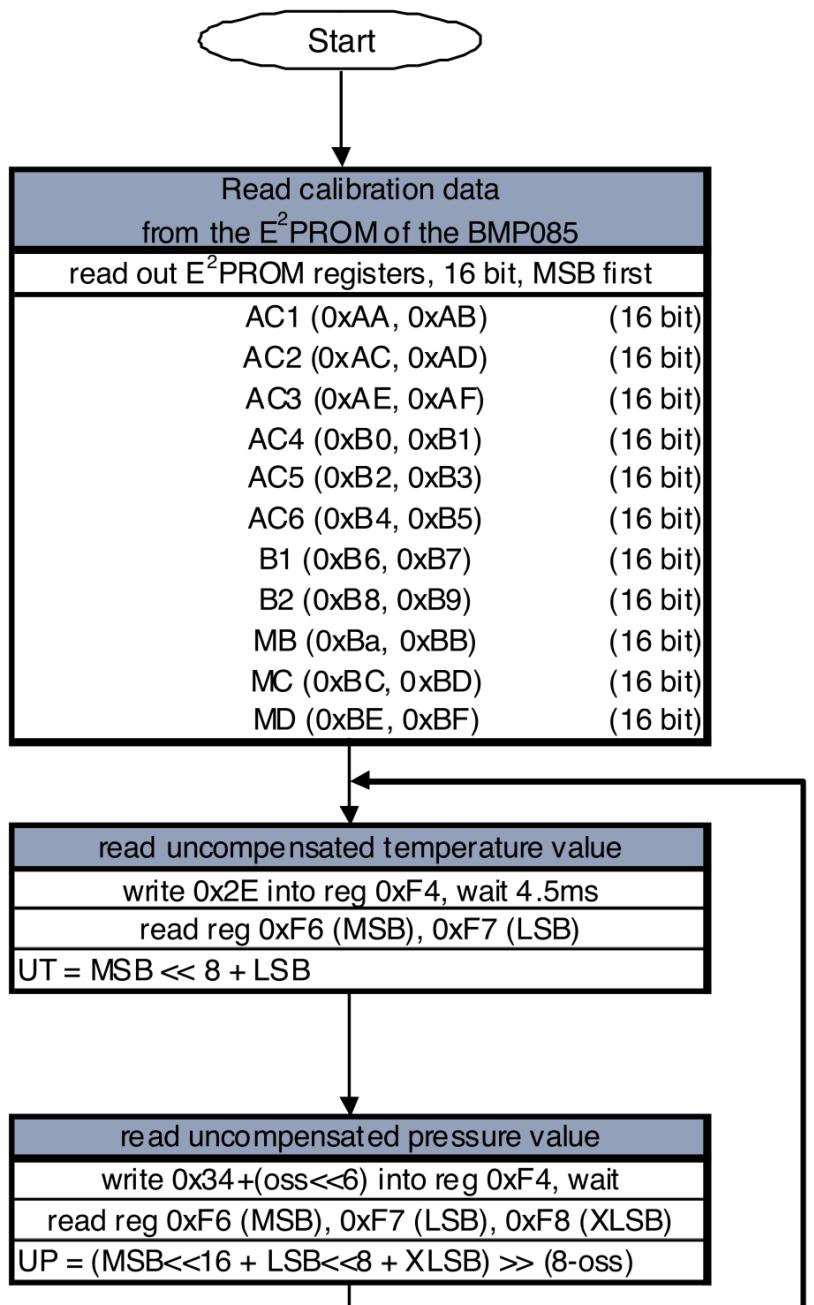
□ Debug coefficients (datasheet)

```
def _load_datasheet_calibration(self):
    # Set calibration from values
    # in the datasheet example.
    # Useful for debugging the
    # temp and pressure calculation accuracy.
    self.cal_AC1 = 408
    self.cal_AC2 = -72
    self.cal_AC3 = -14383
    self.cal_AC4 = 32741
    self.cal_AC5 = 32757
    self.cal_AC6 = 23153
    self.cal_B1 = 6190
    self.cal_B2 = 4
    self.cal_MB = -32767
    self.cal_MC = -8711
    self.cal_MD = 2868
```

AC1 =	408
AC2 =	-72
AC3 =	-14383
AC4 =	32741
AC5 =	32757
AC6 =	23153
B1 =	6190
B2 =	4
MB =	-32768
MC =	-8711
MD =	2868



Datasheet example



example:

C code function: type:

bmp085_get_cal_param

AC1 (0xAA, 0xAB)	(16 bit)	AC1 = 408	short
AC2 (0xAC, 0xAD)	(16 bit)	AC2 = -72	short
AC3 (0xAE, 0xAF)	(16 bit)	AC3 = -14383	short
AC4 (0xB0, 0xB1)	(16 bit)	AC4 = 32741	unsigned short
AC5 (0xB2, 0xB3)	(16 bit)	AC5 = 32757	unsigned short
AC6 (0xB4, 0xB5)	(16 bit)	AC6 = 23153	unsigned short
B1 (0xB6, 0xB7)	(16 bit)	B1 = 6190	short
B2 (0xB8, 0xB9)	(16 bit)	B2 = 4	short
MB (0xBA, 0xBB)	(16 bit)	MB = -32768	short
MC (0xBC, 0xBD)	(16 bit)	MC = -8711	short
MD (0xBE, 0xBF)	(16 bit)	MD = 2868	short

bmp085_get_ut

UT = 27898

long

OSS = 0
= oversampling_setting
(ultra low power mode)

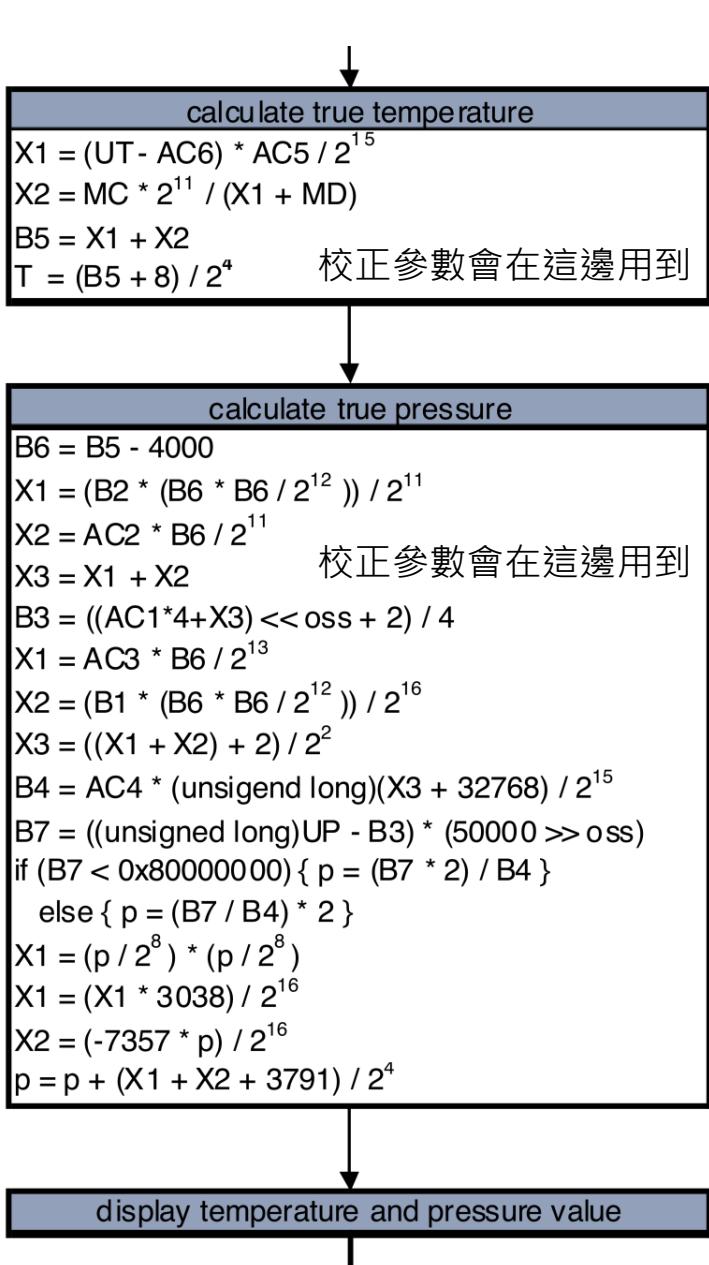
short (0 .. 3)

bmp085_get_up

UP = 23843

long

Measurement flowchart

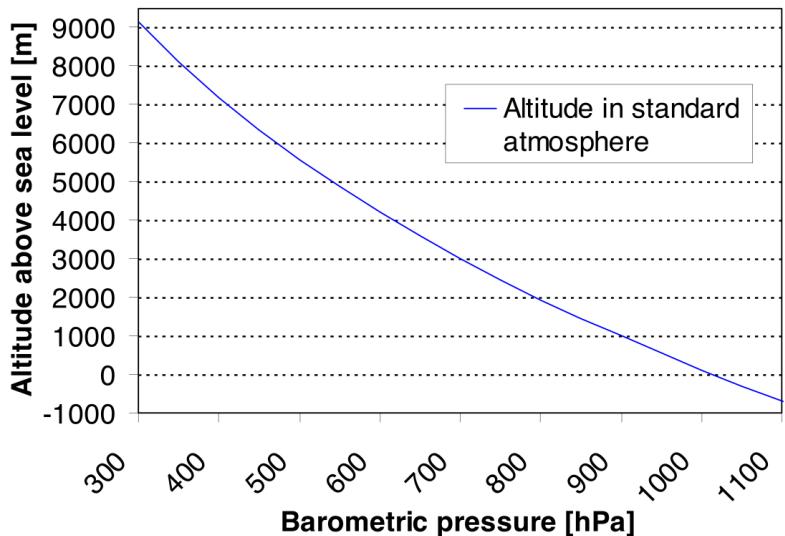


$X1 =$ $X2 =$ $B5 =$ $T =$	4743 -2344 2399 150	bmp085_get_temperature long long long temp in 0.1°C long
		BMP085_calpressure
	$B6 =$	long
	$X1 =$	long
	$X2 =$	long
	$X3 =$	long
	$B3 =$	long
	$X1 =$	long
	$X2 =$	long
	$X3 =$	long
	$B4 =$	long
	$B7 =$	long
	$p =$	long
	$X1 =$	long
	$X1 =$	long
	$X2 =$	long
	$p =$	long
	74529	long
	3454	long
	-7859	long
	69964	press. in Pa long



4. Altitude (BMP085)

- How to calculate altitude?



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

```
print('Altitude = {0:0.2f} m'.format(sensor.read_altitude()))
```

The code is already done for you.

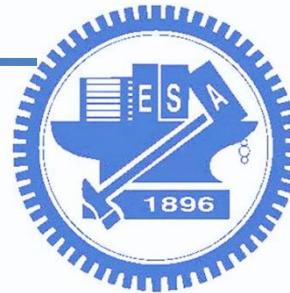


Todo: Integrate code

- GY801 can provide 3-axis Accelerometer, Gyroscope, magnetometer and pressure.
- **Integrate the codes, print accelerometer, gyroscope, magnetometer and altitude at the same time.**
 - The sensing value should be converted to corresponding unit.
 - Ex: accel = +/-2G

ACC: x: ? ,y: ? ,z: ?; GYRO: x: ? ,y ?,z: ?; MAG: x: ?, y: ?, z: ?; Alti: ?.

We will use this for Quiz 2.



After obtain sensing value

- How to smooth the sensing data? -> Use filter

1. Moving average
2. Low-pass filter
3. Kalman filter
4. Complementary filter

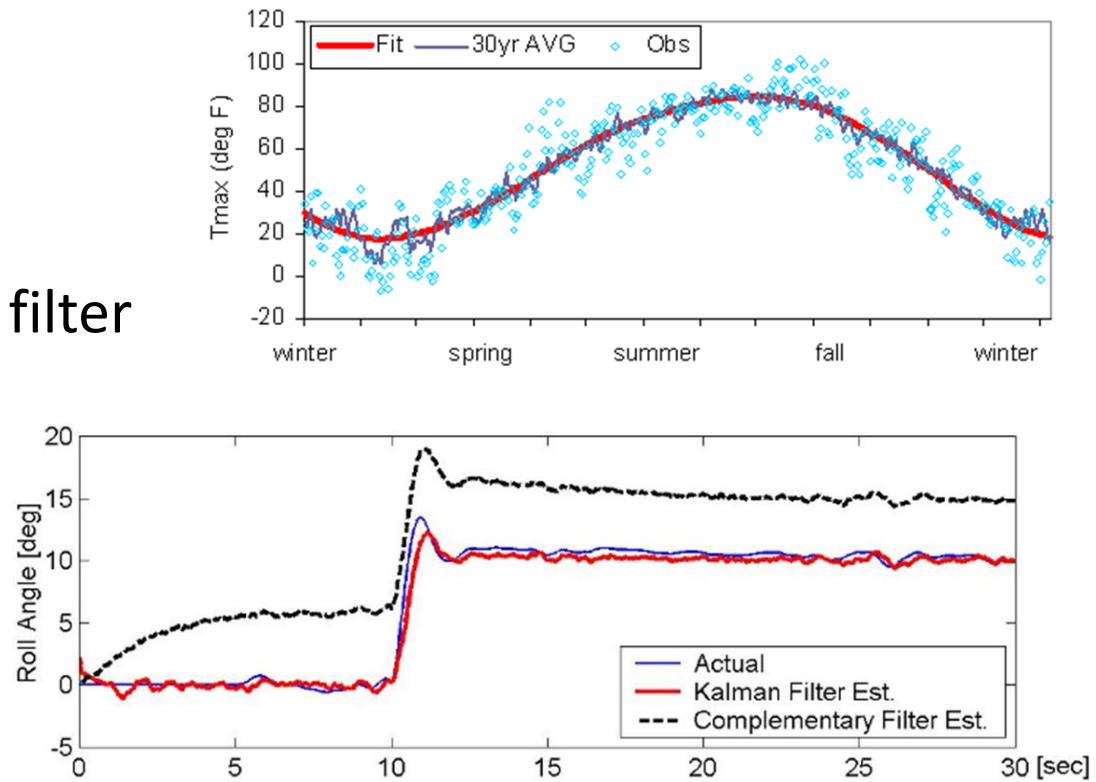
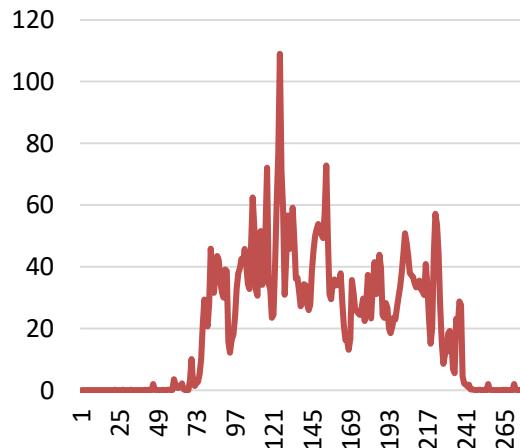
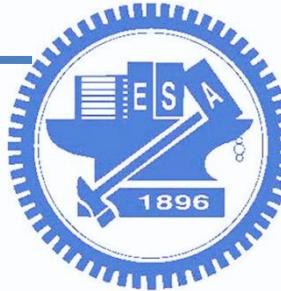


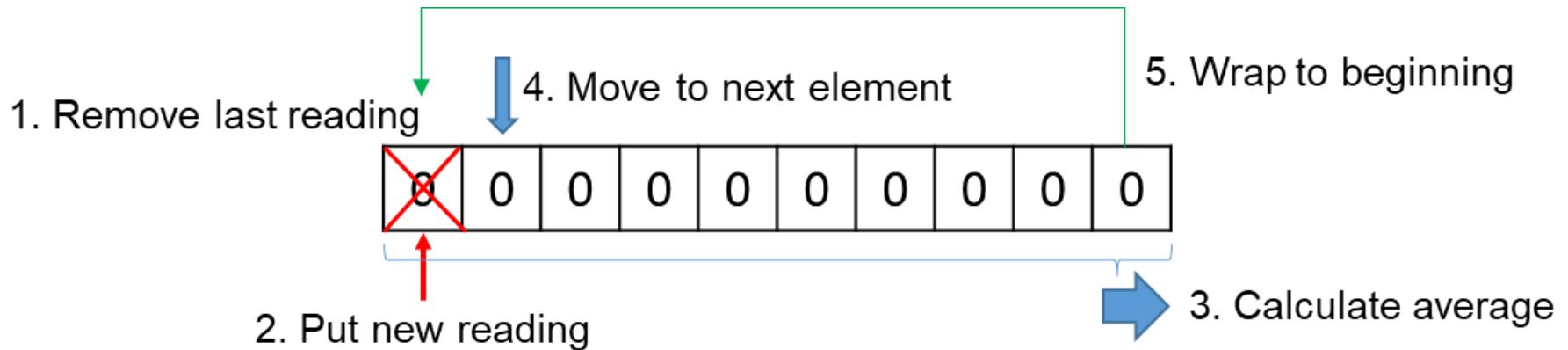
Fig source: <https://github.com/snowpuppy/augreality/wiki/Alec-green-notebook>

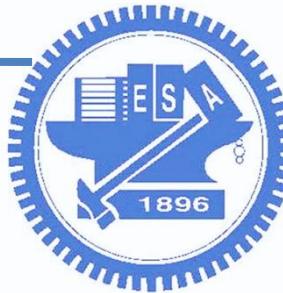
Fig source: http://www.nws.noaa.gov/om/csd/pds/PCU2/statistics/Stats/part2/Filter_LP.htm



1. Moving average

□ Basic concept:





2. Low-pass Filter

□ Features

- letting the low frequencies pass and attenuating the high frequencies
- Simple to implement

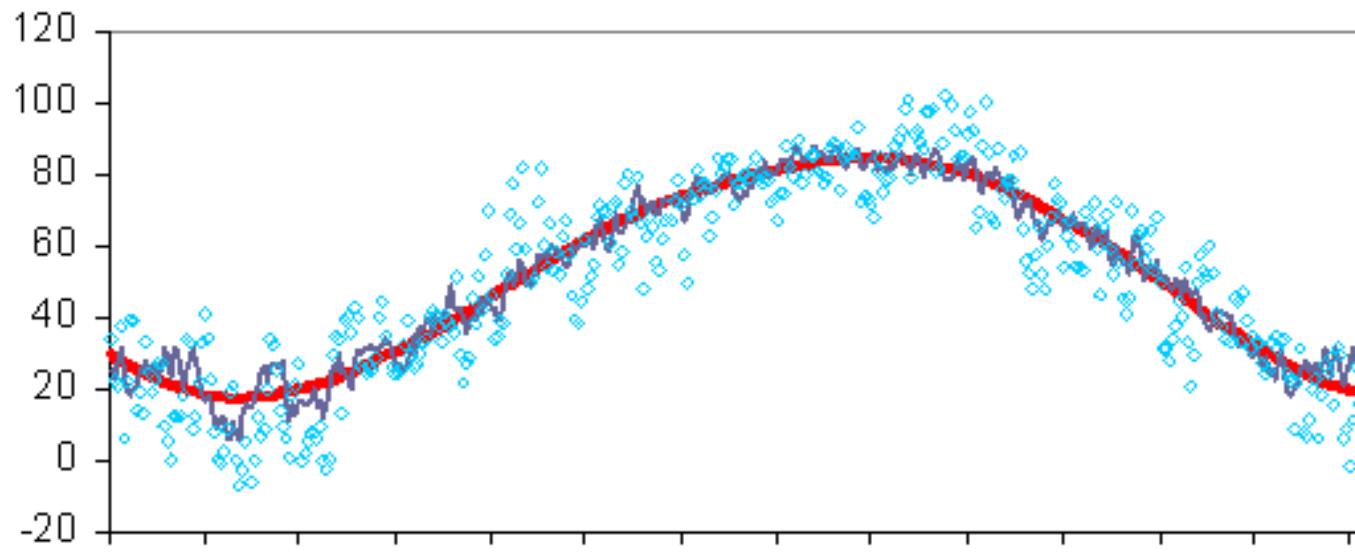
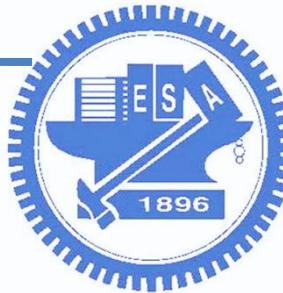


Fig source: http://www.nws.noaa.gov/om/csd/pds/PCU2/statistics/Stats/part2/Filter_LP.htm



2. Low-pass Filter

- Implement low-pass filter in the code

- $x[]$: current reading
- $y[]$: output value
- α : smoothing factor (0-1)

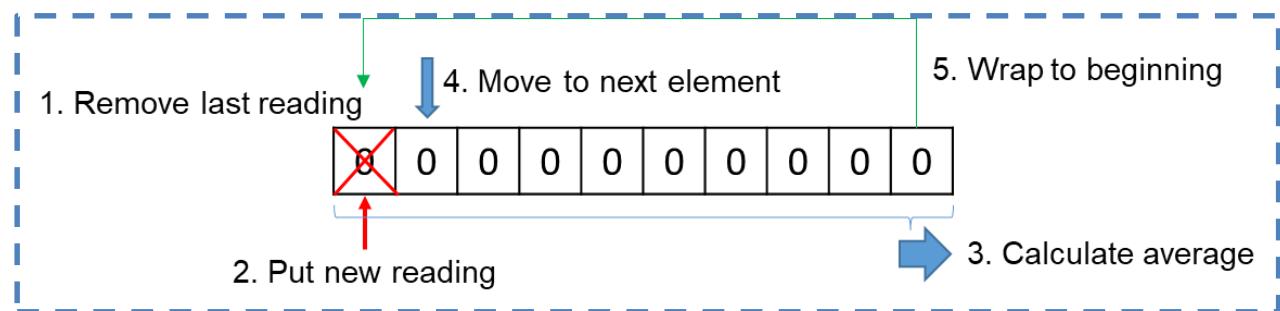
$$y[t] := \alpha * x[t] + (1-\alpha) * y[t-1]$$

```
// Pseudocode algorithm
// Return RC low-pass filter output samples, given input samples,
// time interval dt, and time constant RC
function lowpass(real[0..n] x, real dt, real RC)
    var real[0..n] y
    var real α := dt / (RC + dt)
    y[0] := x[0]
    for i from 1 to n
        y[i] := α * x[i] + (1-α) * y[i-1]
    return y
```



Quiz 2

- How to smooth the sensing data? Try to write these two methods for the norm of acceleration $|a|$.
 1. Moving average
 2. Low-pass filter



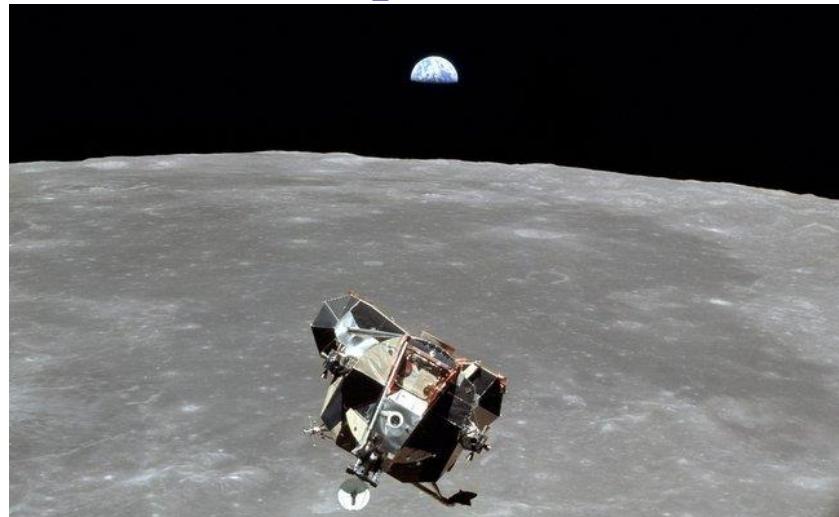
$$y[t] = \alpha * x[t] + (1-\alpha) * y[t-1], \text{ where } \alpha = 0.5$$



3. Kalman filter

- Applications
 - Determination of planet orbit parameters from limited earth observations

A Kalman filter was used to assist with navigation in the Apollo 11 moon landing.



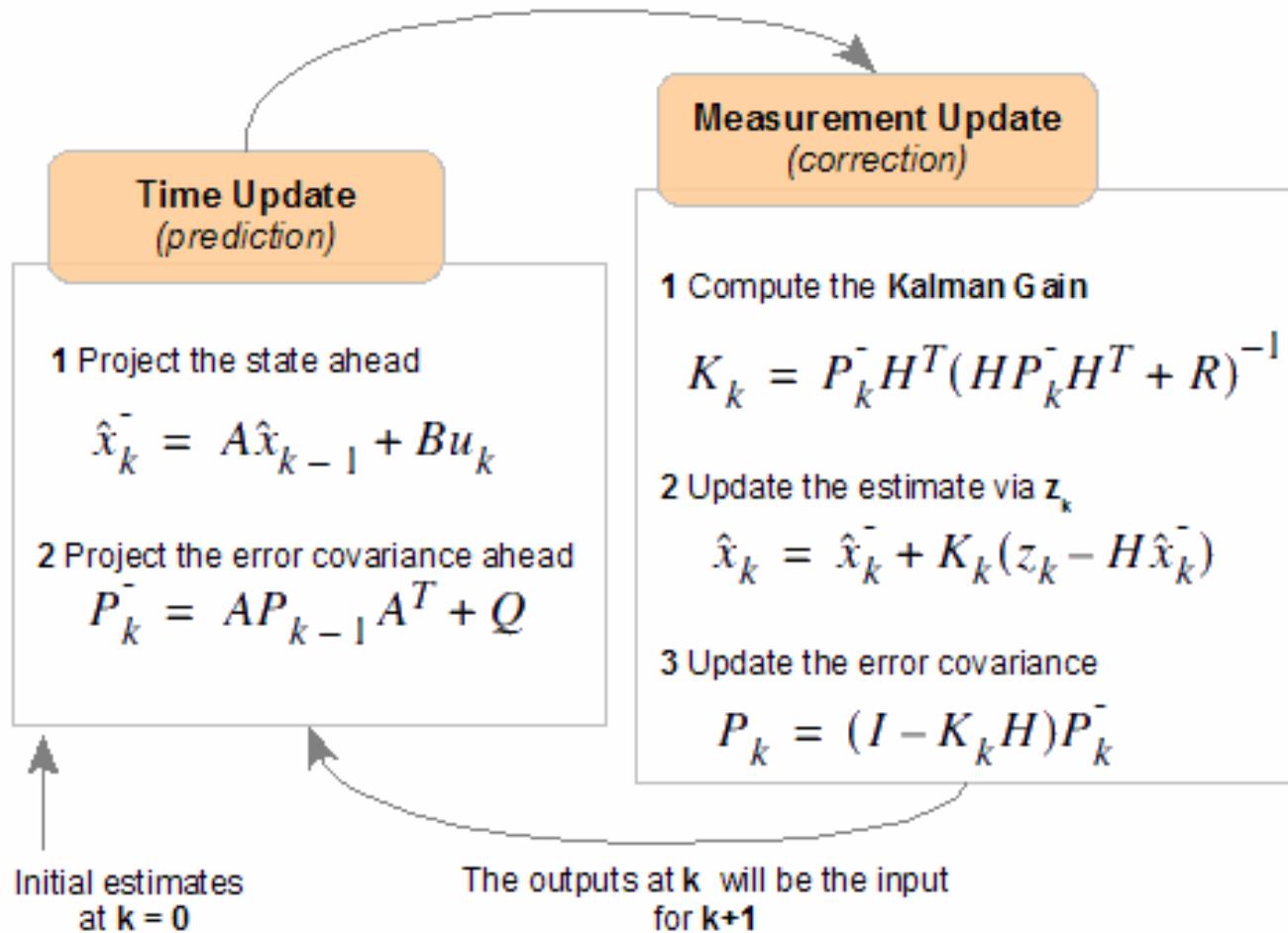


3. Kalman filter

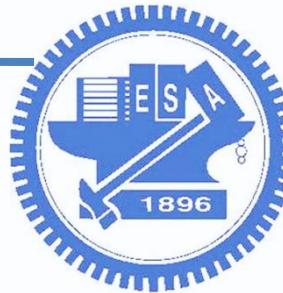
- Kalman filter is an optimal estimator
 - infer parameters of interest from **indirect**, **inaccurate** and **uncertain observations**
 - produce **estimates** of unknown variables that tend to be more accurate
- Why Kalman filter?
 - Good results in practice due to optimality and structure
 - Convenient form for online real time processing
 - Easy to formulate and implement given a basic understanding



3. Kalman filter



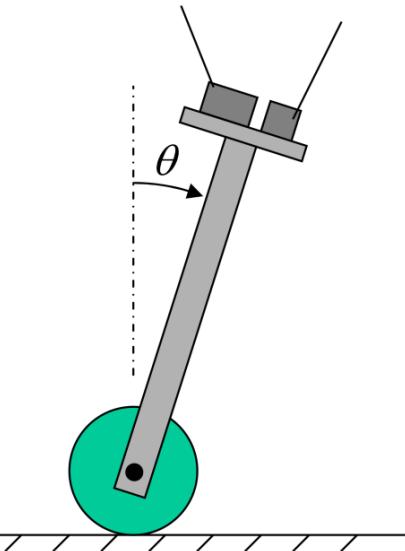
We can use Kalman filter to combine the accelerometer and gyro values.



4. Complementary filter

- Accelerometer is good in long term.
- Gyroscope is good in short term.
 - Use Complementary filter to integrate them

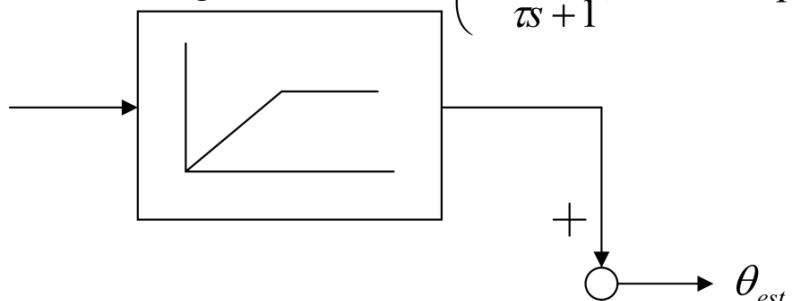
accelerometer rate gyro



$$\theta \approx \int (\text{angular rate}) dt$$

- not good in long term
due to integration

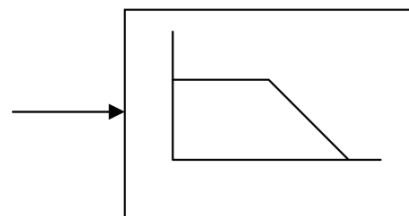
High Pass Filter $\left(= \frac{\tau s}{\tau s + 1}, \text{ for example} \right)$



$$\theta \approx \sin^{-1} \left(\frac{\text{accel. output}}{g} \right)$$

- only good in long term
- not proper during fast motion

Low Pass Filter $\left(= \frac{1}{\tau s + 1} \right)$



We can use Complementary filter to combine the accelerometer and gyro values.



BerryIMU (for 3 and 4)

- The code currently performs **angle measurements** using the **gyroscope** and **accelerometer**, which are fused using a **complementary filter** and **kalman filter**.

```
#Complementary filter used to combine the accelerometer and gyro values.
```

```
CFangleX=AA*(CFangleX+rate_gyr_x*LP) +(1 - AA) * AccXangle
```

```
CFangleY=AA*(CFangleY+rate_gyr_y*LP) +(1 - AA) * AccYangle
```

```
#Kalman filter used to combine the accelerometer and gyro values.
```

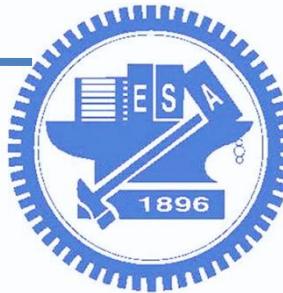
```
kalmanY = kalmanFilterY(AccYangle, rate_gyr_y,LP)
```

```
kalmanX = kalmanFilterX(AccXangle, rate_gyr_x,LP)
```

- Original source code:
 - <https://github.com/mwilliams03/BerryIMU/blob/master/python-BerryIMU-gryo-accel-compass/berryIMU.py>

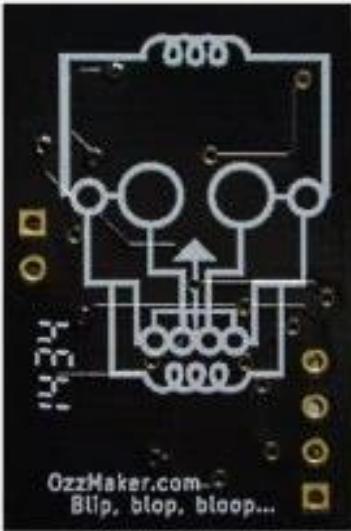
Download code to your PI:

```
wget http://140.113.144.127/~wufish/12kalman_and_comp__blank.py
```



BerryIMU

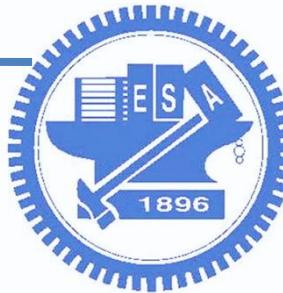
- The code is based on BerryIMU
 - 3D accelerometer, 3D gyroscope, 3D magnetometer
 - Different sensor module, I2C address ...



BerryIMU

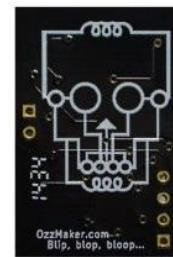
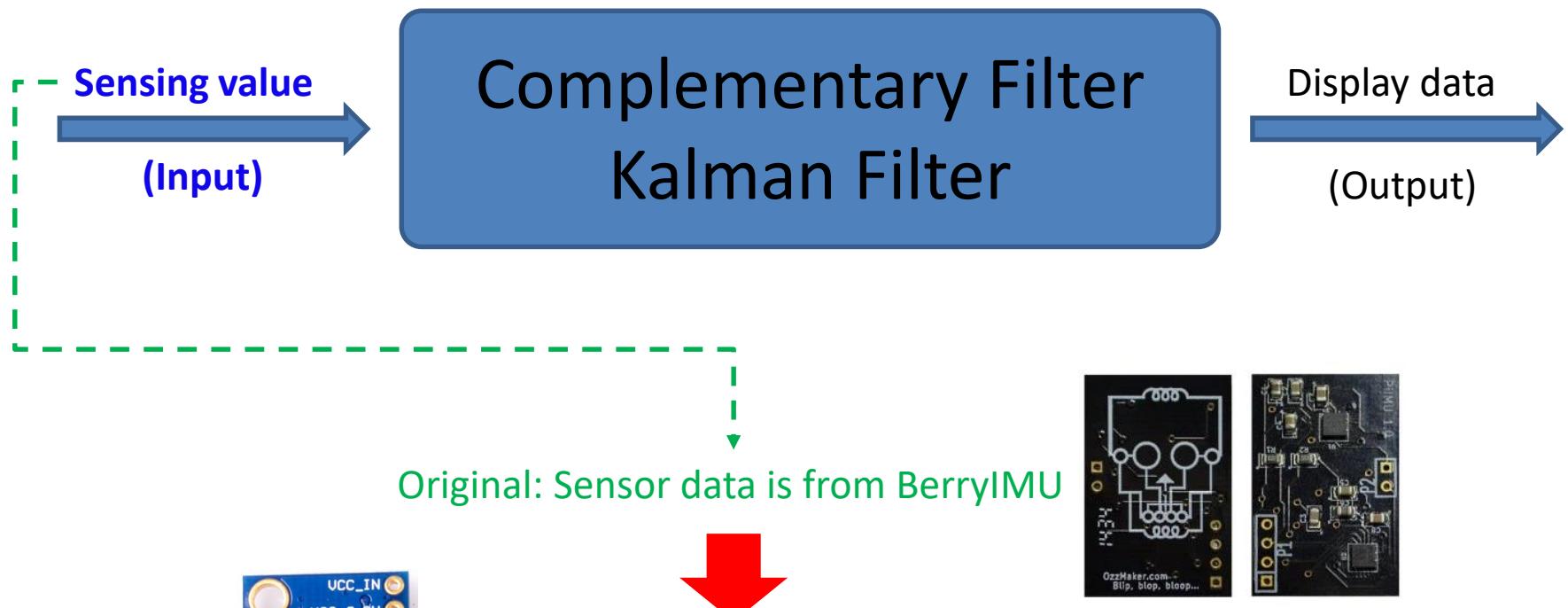


GY801

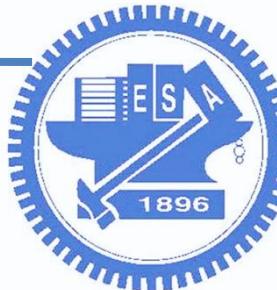


BerryIMU

- Basic flowchart of code



Todo: Sensor data is from GY801

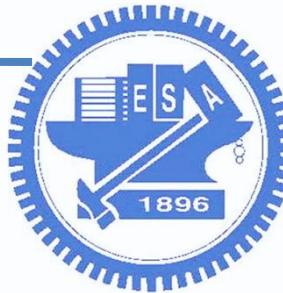


BerryIMU code

□ Import libraries and define parameters

```
import Adafruit_ADXL345
import string
import sys, os, math, time, thread, smbus, random, requests
import Queue
from signal import signal, SIGPIPE, SIG_DFL
import smbus
import time
import math
import datetime
bus = smbus.SMBus(1)

RAD_TO_DEG = 57.29578
M_PI = 3.14159265358979323846
G_GAIN = 0.070 # [deg/s/LSB] If you change the dps for gyro,
AA = 0.40      # Complementary filter constant
```



BerryIMU code

□ Kalman filter

```
#Kalman filter variables
Q_angle = 0.02
Q_gyro = 0.0015
R_angle = 0.005
y_bias = 0.0
x_bias = 0.0
XP_00 = 0.0
XP_01 = 0.0
XP_10 = 0.0
XP_11 = 0.0
YP_00 = 0.0
YP_01 = 0.0
YP_10 = 0.0
YP_11 = 0.0
KFangleX = 0.0
KFangleY = 0.0
```

```
def kalmanFilterY ( accAngle, gyroRate, DT):
    ...
    ...

def kalmanFilterX ( accAngle, gyroRate, DT):
    ...
    ...

while True:
    # Read sensor data
    ...
    ...

    # Kalman filter used to combine
    # the accelerometer and gyro values.
    kalmanY = kalmanFilterY(AccYangle, rate_gyr_y,LP)
    kalmanX = kalmanFilterX(AccXangle, rate_gyr_x,LP)
    ...
```



BerryIMU code

□ Complementary filter

...

```
while True:
```

```
    # Read sensor data
```

...

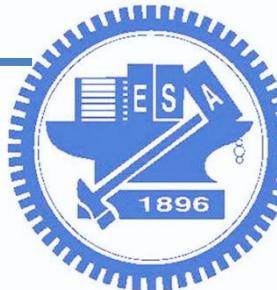
```
# Complementary filter used to combine
```

```
# the accelerometer and gyro values.
```

```
CFangleX=AA*(CFangleX+rate_gyr_x*LP)+(1 - AA) * AccXangle
```

```
CFangleY=AA*(CFangleY+rate_gyr_y*LP)+(1 - AA) * AccYangle
```

...



Quiz 3

- Download and modify the BerryIMU code, the sensor data is from GY801.
Show the filter results on your screen.

```
wget http://140.113.144.127/~wufish/12kalman_and_comp__blank.py
```

```
while True:
```

```
# Read the accel, gyro and compass values
```

```
ACCx = ??
```

```
ACCy = ??
```

```
ACCz = ??
```

```
GYRx = ??
```

```
GYRy = ??
```

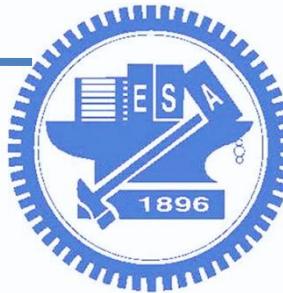
```
GYRz = ??
```

```
MAGx = ??
```

```
MAGy = ??
```

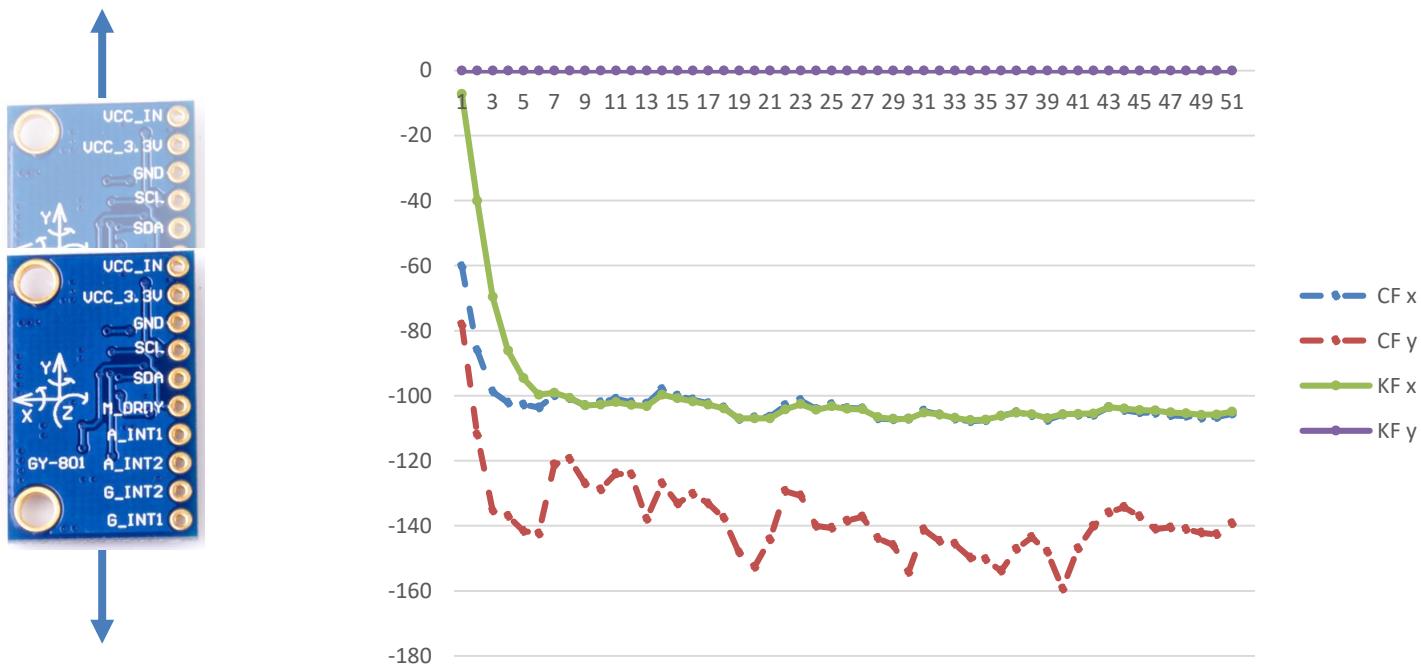
```
MAGz = ??
```

Sensor data is from GY801



Discussion

- Based on Quiz 3, wave the sensor and record the output value of filters, then draw it. Write down your observation.





Summary

- Practice Lab (magnetometer, pressure and filter)
- Write down the answer for discussion
 - Discussion:
 - 1. Rotate the magnetometer for 360 degree. Record the value for x and y.
 - 2. Wave the sensor and record the output value of filters.
 - Draw them and write down your observation.
 - Deadline: Before 4/5, 12:00
- Write code for **Quiz 1 - 3**, then demonstrate it to TAs
 - **Quiz1: How to estimate the heading?**
 - **Quiz2: Moving average and LPF for $|a|$.**
 - **Quiz3: Combine BerryIMU with GY801.**
 - Deadline: Before 3/29, 15:10
 - Late Demo: Before 4/12, 15:10