



# 嵌入式系統總整與實作

曾煜棋、吳昆儒

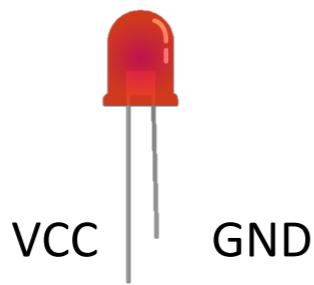
**National Yang Ming Chiao Tung University**



# Last week

- How to use wire to connect devices?
- How to write code to read sensing data on GPIO?

**LED**

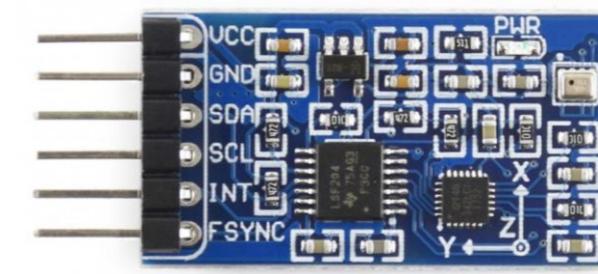


**HC-SR04**



**Ultrasonic**

**ICM20948**



**IMU**  
(Inertial measurement unit)



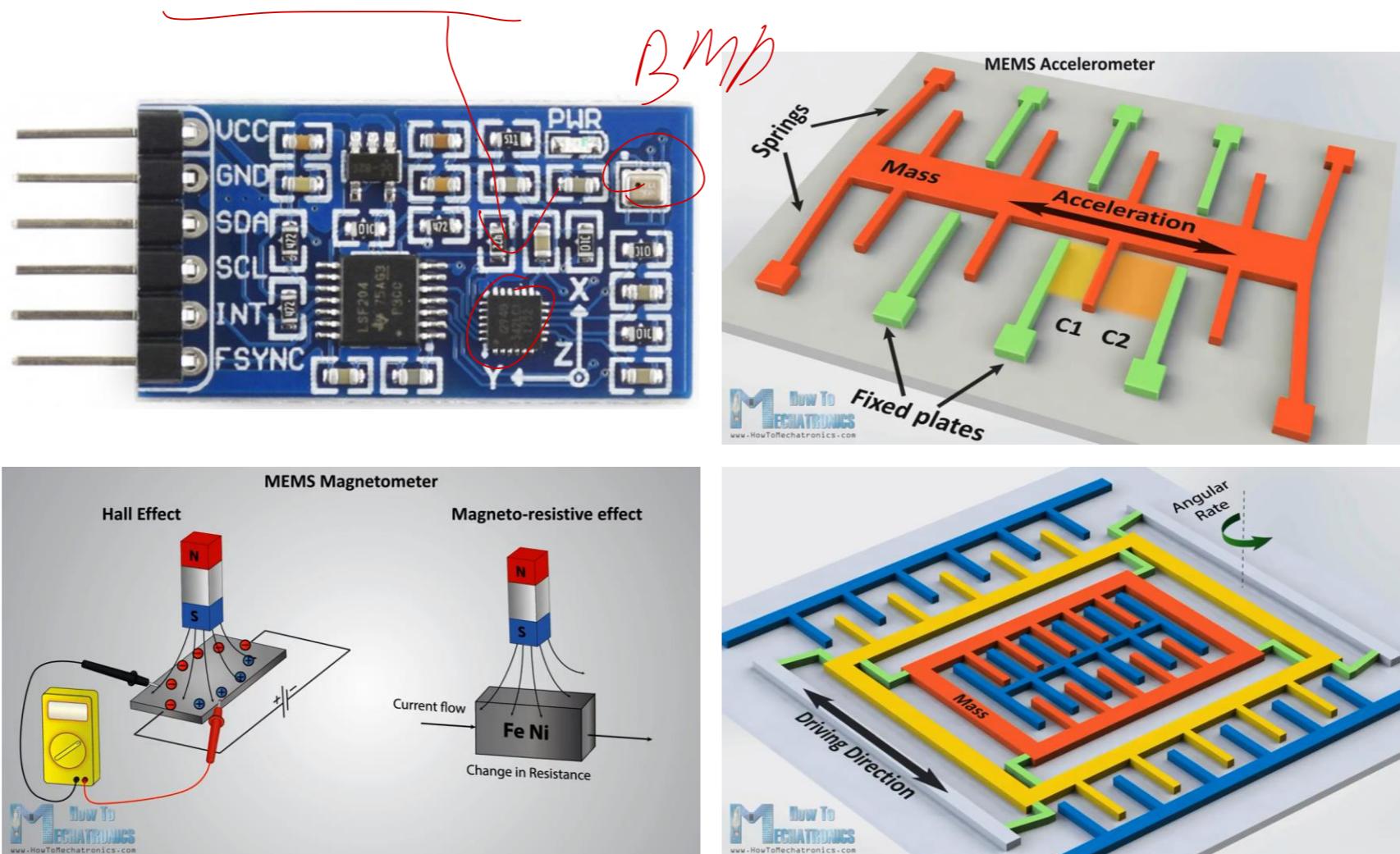
# This week

- Process sensing data
  - The information of IMU
  - Datasheet and code configuration
  - IMU applications
    - Calibration  $a$
    - Calculate distance, rotation angle and heading  $g$
    - Fall detection  $m$

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



## How MEMS Work (Accelerometer, Gyroscope, Magnetometer)



IMU. How MEMS Accelerometer Gyroscope Magnetometer Work & Arduino Tutorial



# Barometer

Fail



## 1. Altitude (BMP085)

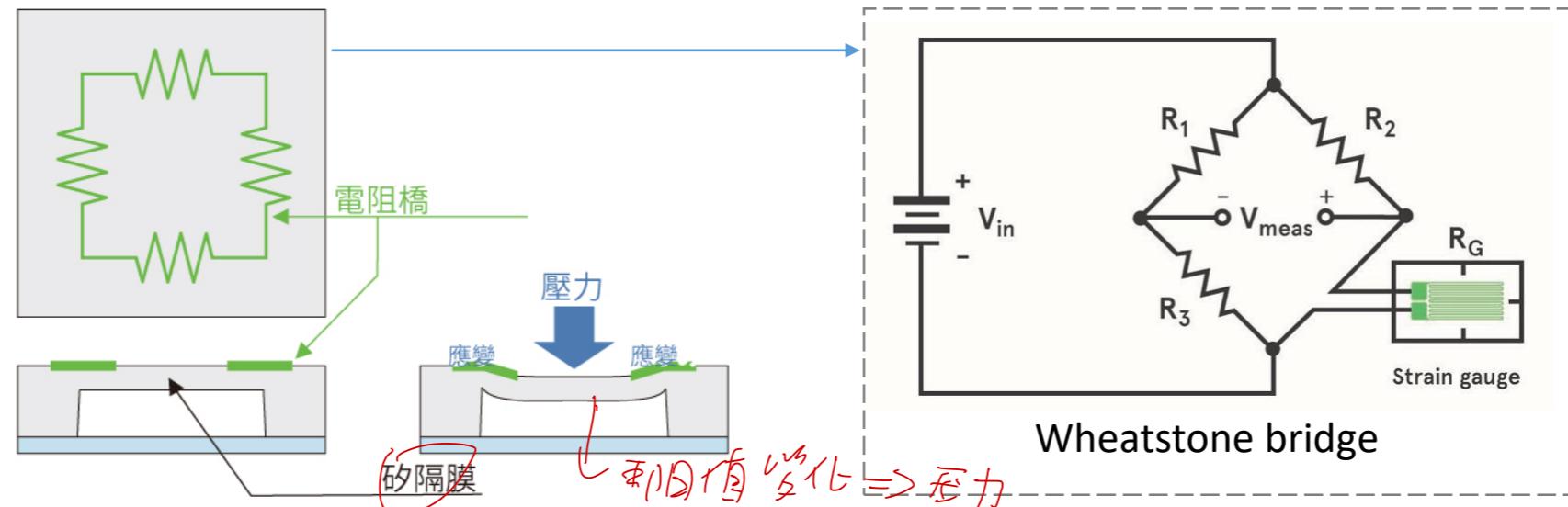
- Pressure sensing range: 300-1100 hPa
  - (9000m to -500m above sea level)
- Up to 0.03hPa / 0.25m resolution
- -40 to +85°C operational range
- +-2°C temperature accuracy
  - Temperature measurement included
- 2-pin i2c interface on chip





# Sensor - Barometer

當  $R_3/R_1 = R_G/R_2$  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。



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

Robert Bosch is the world market leader for pressure sensors in automotive and consumer applications. Bosch's proprietary APSM (Advanced Porous Silicon Membrane) MEMS manufacturing process is fully CMOS compatible and allows a hermetic sealing of the cavity in an all silicon process. The BMP280 is based on Bosch's proven Piezo-resistive pressure sensor technology featuring high EMC robustness, high accuracy and linearity and long term stability.



# BMP280 measurement cycle

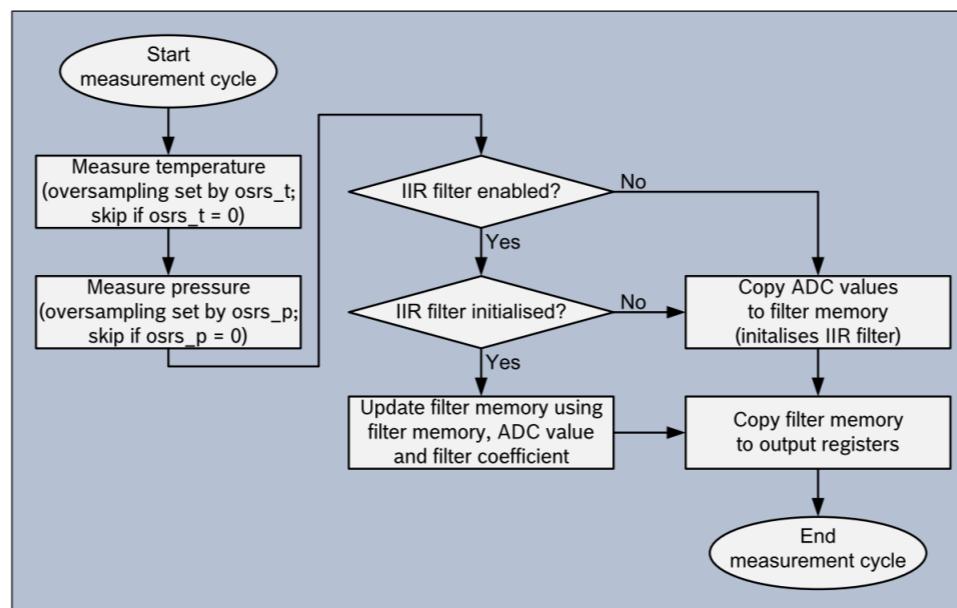


Figure 2: BMP280 measurement cycle

## 3.11 Output compensation

The BMP280 output consists of the ADC output values. However, each sensing element behaves differently, and actual pressure and temperature must be calculated using a set of calibration parameters. The recommended calculation in chapter 3.11.3 uses fixed point arithmetic.

→ 李本友



# Read calibration data

```
def load_calibration_params():
    # Read calibration data from EEPROM of BMP280
    calib = []
    for i in range(0x88, 0x88+24):
        calib.append(bus.read_byte_data(BMP280_ADDRESS, i))
    calib.append(bus.read_byte_data(BMP280_ADDRESS, 0xA1))
    return calib
```

T<sub>1</sub> T<sub>2</sub> T<sub>3</sub> P<sub>1</sub> P<sub>2</sub> P<sub>3</sub>

Table 17: Compensation parameter storage, naming and data type

| Register Address<br>LSB / MSB | Register content | Data type      | Register Address<br>LSB / MSB | Register content | Data type    |
|-------------------------------|------------------|----------------|-------------------------------|------------------|--------------|
| 0x88 / 0x89                   | dig_T1           | unsigned short | 0x94 / 0x95                   | dig_P4           | signed short |
| 0x8A / 0x8B                   | dig_T2           | signed short   | 0x96 / 0x97                   | dig_P5           | signed short |
| 0x8C / 0x8D                   | dig_T3           | signed short   | 0x98 / 0x99                   | dig_P6           | signed short |
| 0x8E / 0x8F                   | dig_P1           | unsigned short | 0x9A / 0x9B                   | dig_P7           | signed short |
| 0x90 / 0x91                   | dig_P2           | signed short   | 0x9C / 0x9D                   | dig_P8           | signed short |
| 0x92 / 0x93                   | dig_P3           | signed short   | 0x9E / 0x9F                   | dig_P9           | signed short |
|                               |                  |                | 0xA0 / 0xA1                   | reserved         | reserved     |



# Read data

- Read raw data
  - Temperature
  - Pressure

```
def read_bmp280_data():
    # Read temperature and pressure raw data from BMP280
    # Temperature
    msb = bus.read_byte_data(BMP280_ADDRESS, 0xFA)
    lsb = bus.read_byte_data(BMP280_ADDRESS, 0xFB)
    xlsb = bus.read_byte_data(BMP280_ADDRESS, 0xFC)
    adc_T = (msb << 16) | (lsb << 8) | xlsb
    adc_T >>= 4

    # Pressure
    msb = bus.read_byte_data(BMP280_ADDRESS, 0xF7)
    lsb = bus.read_byte_data(BMP280_ADDRESS, 0xF8)
    xlsb = bus.read_byte_data(BMP280_ADDRESS, 0xF9)
    adc_P = (msb << 16) | (lsb << 8) | xlsb
    adc_P >>= 4

    return adc_T, adc_P
```

Table 18: Memory map

| Register Name     | Address     | bit7        | bit6            | bit5 | bit4             | bit3           | bit2      | bit1         | bit0 | Reset state |
|-------------------|-------------|-------------|-----------------|------|------------------|----------------|-----------|--------------|------|-------------|
| temp_xlsb         | 0xFC        |             | temp_xlsb<7:4>  |      | 0                | 0              | 0         | 0            | 0    | 0x00        |
| temp_lsb          | 0xFB        |             |                 |      | temp_lsb<7:0>    |                |           |              |      | 0x00        |
| temp_msb          | 0xFA        |             |                 |      |                  | temp_msb<7:0>  |           |              |      | 0x80        |
| press_xlsb        | 0xF9        |             | press_xlsb<7:4> |      | 0                | 0              | 0         | 0            | 0    | 0x00        |
| press_lsb         | 0xF8        |             |                 |      | press_lsb<7:0>   |                |           |              |      | 0x00        |
| press_msb         | 0xF7        |             |                 |      |                  | press_msb<7:0> |           |              |      | 0x80        |
| config            | 0xF5        | t_sb[2:0]   |                 |      | filter[2:0]      |                |           | spi3w_en[0]  |      | 0x00        |
| ctrl_meas         | 0xF4        | osrs_t[2:0] |                 |      | osrs_p[2:0]      |                | mode[1:0] |              |      | 0x00        |
| status            | 0xF3        |             |                 |      | measuring[0]     |                |           | im_update[0] |      | 0x00        |
| reset             | 0xE0        |             |                 |      | reset[7:0]       |                |           |              |      | 0x00        |
| id                | 0xD0        |             |                 |      | chip_id[7:0]     |                |           |              |      | 0x58        |
| calib25...calib00 | 0xA1...0x88 |             |                 |      | calibration data |                |           |              |      | individual  |



# Compensation formula

## 3.11 Output compensation

The BMP280 output consists of the ADC output values. However, each sensing element behaves differently, and actual pressure and temperature must be calculated using a set of calibration parameters. The recommended calculation in chapter 3.11.3 uses fixed point arithmetic.

```
def compensate_temperature(adc_T):
    # Compensation algorithm from BMP280 datasheet
    var1 = (((adc_T >> 3) - (dig_T1 << 1)) * dig_T2) >> 11
    var2 = (((((adc_T >> 4) - dig_T1) * ((adc_T >> 4) - dig_T1)) >> 12) *
            dig_T3) >> 14
    t_fine = var1 + var2
    temperature = (t_fine * 5 + 128) >> 8
    return temperature, t_fine

def compensate_pressure(adc_P, t_fine):
    # Compensation algorithm from BMP280 datasheet
    var1 = t_fine - 128000
    var2 = var1 * var1 * dig_P6
    var2 = var2 + ((var1 * dig_P5) << 17)
    var2 = var2 + (dig_P4 << 35)
    var1 = ((var1 * var1 * dig_P3) >> 8) + ((var1 * dig_P2) <> 12)
    var1 = (((1 << 47) + var1) * dig_P1) >> 33

    if var1 == 0:
        return 0 # Avoid division by zero

    p = 1048576 - adc_P
    p = (((p << 31) - var2) * 3125) // var1
    var1 = (dig_P9 * (p >> 13) * (p >> 13)) >> 25
    var2 = (dig_P8 * p) >> 19

    pressure = ((p + var1 + var2) >> 8) + (dig_P7 << 4)
    return pressure
```

*Datasheet*

```
// Returns temperature in DegC, resolution is 0.01 DegC. Output value of 't_fine' carries fine temperature as global value
BMP280_S32_t bmp280_compensate_T_int32(BMP280_S32_t adc_T)
{
    BMP280_S32_t var1, var2, T;
    var1 = (((adc_T>>3) - ((BMP280_S32_t)dig_T1<<1)) * ((BMP280_S32_t)d
    var2 = (((((adc_T>>4) - ((BMP280_S32_t)dig_T1)) * ((adc_T>>4) - ((BMP
        ((BMP280_S32_t)dig_T3))) >> 14);
    t_fine = var1 + var2;
    T = (t_fine * 5 + 128) >> 8;
    return T;
}

// Returns pressure in Pa as unsigned 32 bit integer. Output value of "96"
BMP280_U32_t bmp280_compensate_P_int32(BMP280_S32_t adc_P)
{
    BMP280_S32_t var1, var2;
    BMP280_U32_t p;
    var1 = (((BMP280_S32_t)t_fine)>>1) - (BMP280_S32_t)64000;
    var2 = (((var1>>2) * (var1>>2)) >> 11) * ((BMP280_S32_t)dig_P6);
```



# Results

- Run sample code:

```
pi@raspberrypi:~/lab3 $ python bmp_280.py
Temperature: 23.64 C
Pressure: 835.54 hPa
Temperature: 23.64 C
Pressure: 835.54 hPa
```

气压采集

```
while True:
    # Read raw temperature and pressure data
    adc_T, adc_P = read_bmp280_data()

    # Compensate raw temperature and pressure
    temperature, t_fine = compensate_temperature(adc_T)
    pressure = compensate_pressure(adc_P, t_fine)

    print(f"Temperature: {temperature / 100.0:.2f} C")
    print(f"Pressure: {pressure / 25600.0:.2f} hPa")

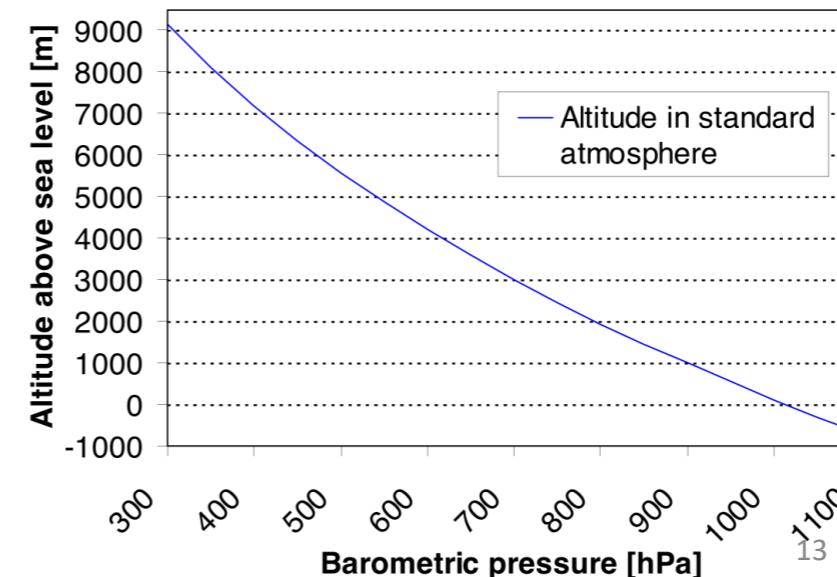
    time.sleep(1)
```



# Quiz 1

- Convert pressure to altitude  $\text{压 力} \rightarrow \text{高 度}$
- Formula:
  - $\text{altitude} = (1 - (\text{pressure}_\text{Pa} / \text{sea_level_pressure}_\text{Pa})^{(1/5.255)}) * 44330$
  - $\text{sea_level_pressure}$ : 海平面的標準氣壓值
    - 通常使用 101325 Pa

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





# IMU configuration



# 10 DOF IMU Sensor (D)

- 10 DOF:
  - 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer, and barometric altimeter
- Chip module
  - ICM20948 (加速度+陀螺儀+AK09916磁力計), BMP280 (氣壓計)
- Accelerometer Features:
  - Measurement range (configurable):  $\pm 2, \pm 4, \pm 8, \pm 16g$
- Gyroscope Features:
  - Measurement range (configurable):  $\pm 250, \pm 500, \pm 1000, \pm 2000^{\circ}/sec$
- Compass/Magnetometer Features:
  - Measurement range:  $\pm 4900\mu T$
- Barometric pressure sensor Features:
  - Measurement range: 300~1100hPa (Altitude: +9000m ~ -500m)



# Datasheet and code configuration

- Sample code: 10dof\_imu\_reading.py (from last week)
- Accelerometer
  - self.set\_accelerometer\_sample\_rate(125)
  - self.set\_accelerometer\_low\_pass(enabled=True, mode=5) ↳ i/B
  - self.set\_accelerometer\_full\_scale(16)
- Gyroscope
  - self.set\_gyro\_sample\_rate(100)
  - self.set\_gyro\_low\_pass(enabled=True, mode=5)
  - self.set\_gyro\_full\_scale(250)
- Magnetometer
  - self.mag\_write(AK09916\_CNTL2, 0x01)



# Accelerometer config

- 加速度計設定 (sample\_rate)

```
ICM20948_ACCEL_SMPLRT_DIV_1 = 0x10  
ICM20948_ACCEL_SMPLRT_DIV_2 = 0x11
```

```
def set_accelerometer_sample_rate(self, rate=125):  
    self.bank(2)  
    rate = int((1125.0 / rate) - 1) # 125Hz - 1.125 kHz / (1 + rate) => 8  
    self.write(ICM20948_ACCEL_SMPLRT_DIV_1, (rate >> 8) & 0xff)  
    self.write(ICM20948_ACCEL_SMPLRT_DIV_2, rate & 0xff)
```



00001000  
LSB

| 10.12 ACCEL_SMPLRT_DIV_2 |                       |   |
|--------------------------|-----------------------|---|
| Name: ACCEL_SMPLRT_DIV_2 |                       |   |
| Address: 17 (11h)        |                       |   |
| Type: USR2               |                       |   |
| Bank: 2                  |                       |   |
| Serial IF: R/W           |                       |   |
| Reset Value: 0x00        |                       |   |
| BIT                      | NAME                  | FUNCTION  |
| 7:0                      | ACCEL_SMPLRT_DIV[7:0] | LSB for ACCEL sample rate div.<br>ODR is computed as follows:<br>1.125 kHz/(1+ACCEL_SMPLRT_DIV[11:0]) |

| 10.11 ACCEL_SMPLRT_DIV_1 |                        |                                |
|--------------------------|------------------------|--------------------------------|
| Name: ACCEL_SMPLRT_DIV_1 |                        |                                |
| Address: 16 (10h)        |                        |                                |
| Type: USR2               |                        |                                |
| Bank: 2                  |                        |                                |
| Serial IF: R/W           |                        |                                |
| Reset Value: 0x00        |                        |                                |
| BIT                      | NAME                   | FUNCTION                       |
| 7:4                      | -                      | Reserved.                      |
| 3:0                      | ACCEL_SMPLRT_DIV[11:8] | MSB for ACCEL sample rate div. |



# Accelerometer config

- 加速度計設定 (full scale range)

```
ICM20948_ACCEL_CONFIG = 0x14
```

```
def set_accelerometer_full_scale(self, scale=16):
    self.bank(2)
    value = self.read(ICM20948_ACCEL_CONFIG) & 0b11111001
    value |= {2: 0b00, 4: 0b01, 8: 0b10, 16: 0b11}[scale] << 1
    self.write(ICM20948_ACCEL_CONFIG, value)
```

## 10.15 ACCEL\_CONFIG

| Name: ACCEL_CONFIG |                    | 7   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------------|--------------------|---|---|---|---|---|---|---|---|
| Address: 20 (14h)  |                    |   |   |   |   |   |   |   |   |
| Type: USR2         |                    |   |   |   |   |   |   |   |   |
| Bank: 2            |                    |   |   |   |   |   |   |   |   |
| Serial IF: R/W     |                    |   |   |   |   |   |   |   |   |
| Reset Value: 0x01  |                    |   |   |   |   |   |   |   |   |
| BIT                | NAME               | FUNCTION  |   |   |   |   |   |   |   |
| 7:6                | -                  | Reserved.   |   |   |   |   |   |   |   |
| 5:3                | ACCEL_DLPFCFG[2:0] | Accelerometer low pass filter configuration as shown in Table 18. |   |   |   |   |   |   |   |
| 2:1                | ACCEL_FS_SEL[1:0]  | Accelerometer Full Scale Select:                                  |   |   |   |   |   |   |   |
|                    |                    | 00: ±2g<br>01: ±4g<br>10: ±8g<br>11: ±16g                         |   |   |   |   |   |   |   |
| 0                  | ACCEL_FCHOICE      | 0: Bypass accel DLPF.<br>1: Enable accel DLPF.                    |   |   |   |   |   |   |   |



# Accelerometer config

- 加速度計設定 (low pass filter): 減少輸出雜訊，提高資料穩定性

```
def set_accelerometer_low_pass(self, enabled=True, mode=5):
    self.bank(2)
    value = self.read(ICM20948_ACCEL_CONFIG) & 0b10001110
    if enabled:
        value |= 0b1
    value |= (mode & 0x07) << 3
    self.write(ICM20948_ACCEL_CONFIG, value)
```

| ACCEL_FCHOICE | ACCEL_DLPCFG | OUTPUT      |          |   |
|---------------|--------------|-------------|----------|---|
|               |              | 3DB BW [HZ] | NBW [HZ] | RATE [HZ]   |
| 0             | x            | 1209        | 1248     | 4500  |
| 1             | 0            | 246.0       | 265.0    | 1125/(1+ACCEL_SMPLRT_DIV)Hz where ACCEL_SMPLRT_DIV is 0, 1, 2,...4095 |
| 1             | 1            | 246.0       | 265.0    | 1125/(1+ACCEL_SMPLRT_DIV)Hz where ACCEL_SMPLRT_DIV is 0, 1, 2,...4095 |
| 1             | 2            | 111.4       | 136.0    | 1125/(1+ACCEL_SMPLRT_DIV)Hz where ACCEL_SMPLRT_DIV is 0, 1, 2,...4095 |
| 1             | 3            | 50.4        | 68.8     | 1125/(1+ACCEL_SMPLRT_DIV)Hz where ACCEL_SMPLRT_DIV is 0, 1, 2,...4095 |
| 1             | 4            | 23.9        | 34.4     | 1125/(1+ACCEL_SMPLRT_DIV)Hz where ACCEL_SMPLRT_DIV is 0, 1, 2,...4095 |
| 1             | 5            | 11.5        | 17.0     | 1125/(1+ACCEL_SMPLRT_DIV)Hz where ACCEL_SMPLRT_DIV is 0, 1, 2,...4095 |
| 1             | 6            | 5.7         | 8.3      | 1125/(1+ACCEL_SMPLRT_DIV)Hz where ACCEL_SMPLRT_DIV is 0, 1, 2,...4095 |
| 1             | 7            | 473         | 499      | 1125/(1+ACCEL_SMPLRT_DIV)Hz where ACCEL_SMPLRT_DIV is 0, 1, 2,...4095 |

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| / | / | 0 | 0 | 0 | / | / | 0 |



# Gyroscope config

- 陀螺儀設定 (sample\_rate)

```
ICM20948_GYRO_SMPLRT_DIV = 0x00
```



```
def set_gyro_sample_rate(self, rate=125):
    self.bank(2)
    rate = int((1125.0 / rate) - 1) # 125Hz - 1.125 kHz / (1 + rate)
    self.write(ICM20948_GYRO_SMPLRT_DIV, rate)
```

## 10.1 GYRO\_SMPLRT\_DIV

Name: GYRO\_SMPLRT\_DIV

Address: 0 (00h)

Type: USR2

Bank: 2

Serial IF: R/W

Reset Value: 0x00

| BIT | NAME                 | FUNCTION  |
|-----|----------------------|---|
| 7:0 | GYRO_SMPLRT_DIV[7:0] | Gyro sample rate divider. Divides the internal sample rate to generate the sample rate that controls sensor data output rate, FIFO sample rate, and DMP sequence rate.<br><b>NOTE:</b> This register is only effective when FCHOICE = 1'b1 (FCHOICE_B register bit is 1'b0), and (0 < DLPF_CFG < 7).<br>ODR is computed as follows:<br>1.1 kHz/(1+GYRO_SMPLRT_DIV[7:0]) |



# Gyroscope config

- 陀螺儀設定 (full scale range)

```
ICM20948_GYRO_CONFIG_1 = 0x01
```

```
def set_gyro_full_scale(self, scale=250):
    self.bank(2)
    value = self.read(ICM20948_GYRO_CONFIG_1) & 0b11111001
    value |= {250: 0b00, 500: 0b01, 1000: 0b10, 2000: 0b11}[scale] << 1
    self.write(ICM20948_GYRO_CONFIG_1, value)
```

## 10.2 GYRO\_CONFIG\_1

| BIT | NAME             | FUNCTION  |
|-----|------------------|---|
| 7:6 | -                | Reserved.   |
| 5:3 | GYRO_DLPCFG[2:0] | Gyro low pass filter configuration as shown in Table 16.                                      |
| 2:1 | GYRO_FS_SEL[1:0] | Gyro Full Scale Select:<br>00 = ±250 dps<br>01 = ±500 dps<br>10 = ±1000 dps<br>11 = ±2000 dps |
| 0   | GYRO_FCHOICE     | 0 – Bypass gyro DLPF.<br>1 – Enable gyro DLPF.  |

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|



# Gyroscope config

- 陀螺儀設定 (low pass filter): 減少輸出雜訊，提高資料穩定性

```
def set_gyro_low_pass(self, enabled=True, mode=5):
    self.bank(2)
    value = self.read(ICM20948_GYRO_CONFIG_1) & 0b11000110
    if enabled:
        value |= 0b1
        value |= (mode & 0x07) << 3
    self.write(ICM20948_GYRO_CONFIG_1, value)
```

5=101

7 6 5 4 3 2 1 0

5:3 = GYRO\_DLPFCFG[2:0]

2:1 = GYRO\_FS\_SEL[1:0]

0 = GYRO\_FCHOICE

| GYRO_FCHOICE | GYRO_DLPFCFG | OUTPUT      |          |  |
|--------------|--------------|-------------|----------|--|
|              |              | 3DB BW [Hz] | NBW [Hz] | RATE [Hz]  |
| 0            | x            | 12106       | 12316    | 9000   |
| 1            | 0            | 196.6       | 229.8    | 1125/(1+GYRO_SMPLRT_DIV)Hz where GYRO_SMPLRT_DIV is 0, 1, 2,...255 |
| 1            | 1            | 151.8       | 187.6    | 1125/(1+GYRO_SMPLRT_DIV)Hz where GYRO_SMPLRT_DIV is 0, 1, 2,...255 |
| 1            | 2            | 119.5       | 154.3    | 1125/(1+GYRO_SMPLRT_DIV)Hz where GYRO_SMPLRT_DIV is 0, 1, 2,...255 |
| 1            | 3            | 51.2        | 73.3     | 1125/(1+GYRO_SMPLRT_DIV)Hz where GYRO_SMPLRT_DIV is 0, 1, 2,...255 |
| 1            | 4            | 23.9        | 35.9     | 1125/(1+GYRO_SMPLRT_DIV)Hz where GYRO_SMPLRT_DIV is 0, 1, 2,...255 |
| 1            | 5            | 11.6        | 17.8     | 1125/(1+GYRO_SMPLRT_DIV)Hz where GYRO_SMPLRT_DIV is 0, 1, 2,...255 |
| 1            | 6            | 5.7         | 8.9      | 1125/(1+GYRO_SMPLRT_DIV)Hz where GYRO_SMPLRT_DIV is 0, 1, 2,...255 |
| 1            | 7            | 361.4       | 376.5    | 1125/(1+GYRO_SMPLRT_DIV)Hz where GYRO_SMPLRT_DIV is 0, 1, 2,...255 |

Table 16. Gyroscope Configuration 1



# Magnetometer config

- 磁力計設定

```
def mag_write(self, reg, value):
    self.bank(3)
    self.write(ICM20948_I2C_SLV0_ADDR, AK09916_I2C_ADDR)
    self.write(ICM20948_I2C_SLV0_REG, reg)
    self.write(ICM20948_I2C_SLV0_DO, value)
    self.bank(0)
    self.trigger_mag_io()
```

self.mag\_write(AK09916\_CNTL2, 0x01) *→ 2(0)1 植<sub>220</sub> ←*

## 13.5 CNTL2: CONTROL 2

| ADDR                | REGISTER NAME | D7 | D6 | D5 | D4    | D3    | D2    | D1    | D0    |
|---------------------|---------------|----|----|----|-------|-------|-------|-------|-------|
| READ/WRITE REGISTER |               |    |    |    |       |       |       |       |       |
| 31H                 | CNTL2         | 0  | 0  | 0  | MPDE4 | MODE3 | MODE2 | MODE1 | MODE0 |
|                     | Reset         | 0  | 0  | 0  | 0     | 0     | 0     | 0     | 0     |



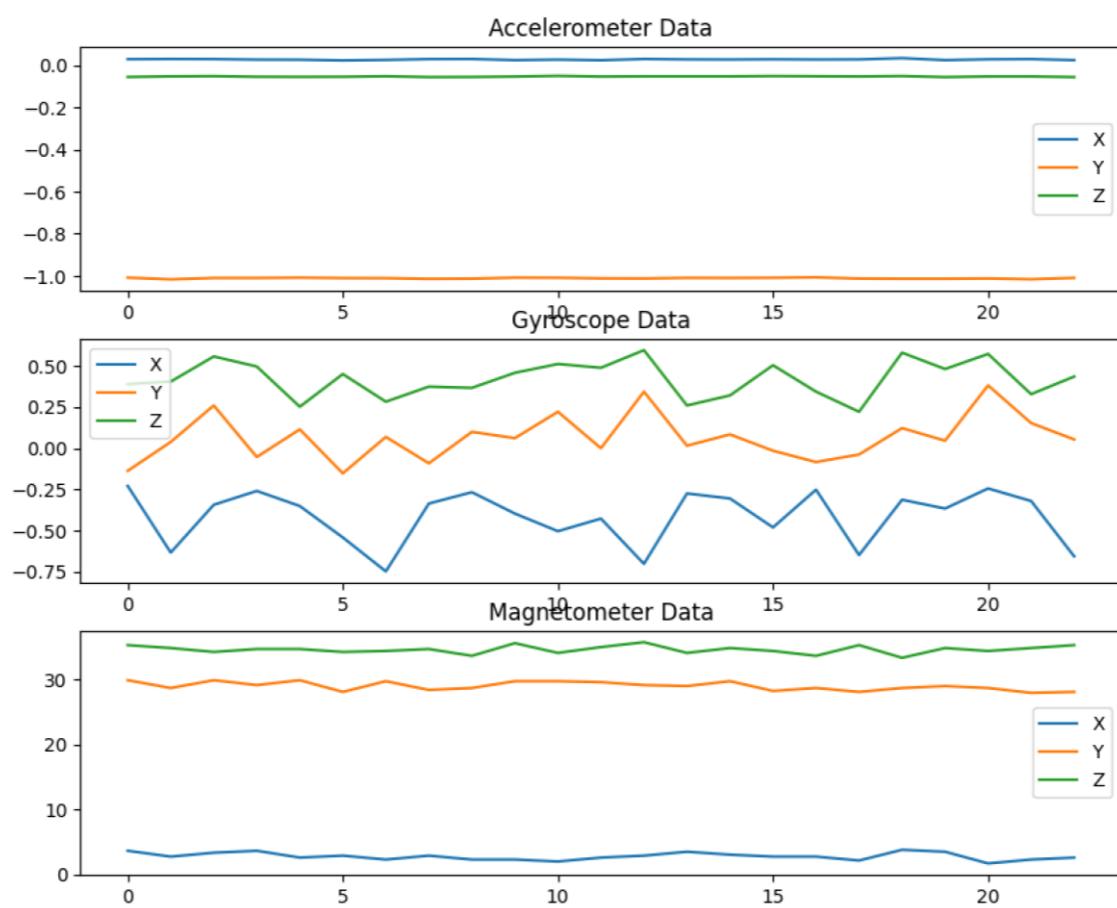
MODE[4:0] bits: Operation mode setting  
“00000”: Power-down mode  
“00001”: Single measurement mode  
“00010”: Continuous measurement mode 1  
“00100”: Continuous measurement mode 2  
“00110”: Continuous measurement mode 3  
“01000”: Continuous measurement mode 4  
“10000”: Self-test mode



# Calibration



# IMU results





# Calibration

- Why calibration?

- 靜置狀態的讀數，理論上應該是固定值
- 但實際上會有飄動，通常稱為漂移drift錯誤 (存在offset需要修正)

- 加速度計 (Accelerometer) :

- 一般可藉由靜置狀態的量測數值做校正（只顯示重力）。

- 陀螺儀 (Gyroscope) :

- 在靜置狀態，通常也會有非零的讀數

- 磁力計 (Magnetometer) :

- 受到環境磁場影響，有硬鐵和軟鐵影響，需要進行磁場校正



# Accelerometer calibration

- 加速度計的output理論上在靜置時應符合：
  - 當IMU水平放置且靜止時，其測到的數值理應為： $(0, 0, \pm 1)$  G

- 校正步驟：

- 將IMU擺放在桌上，確認XYZ的方向與相應的理論值(ex: 0, 0, +1)  
• 靜置數秒取得大量的讀數  
• 針對XYZ軸的讀數各自取平均值，再減去作為offset  
• 即： $\text{acc\_x\_calibrated} = \text{accel\_x\_raw} - \text{offset\_x}$

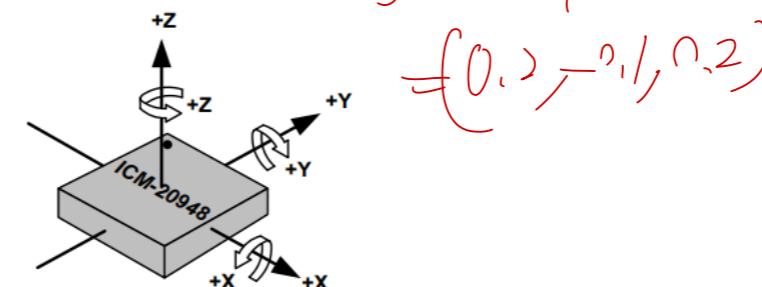
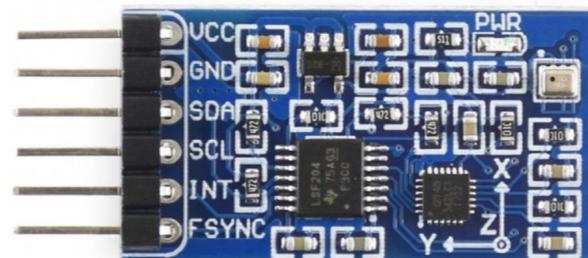
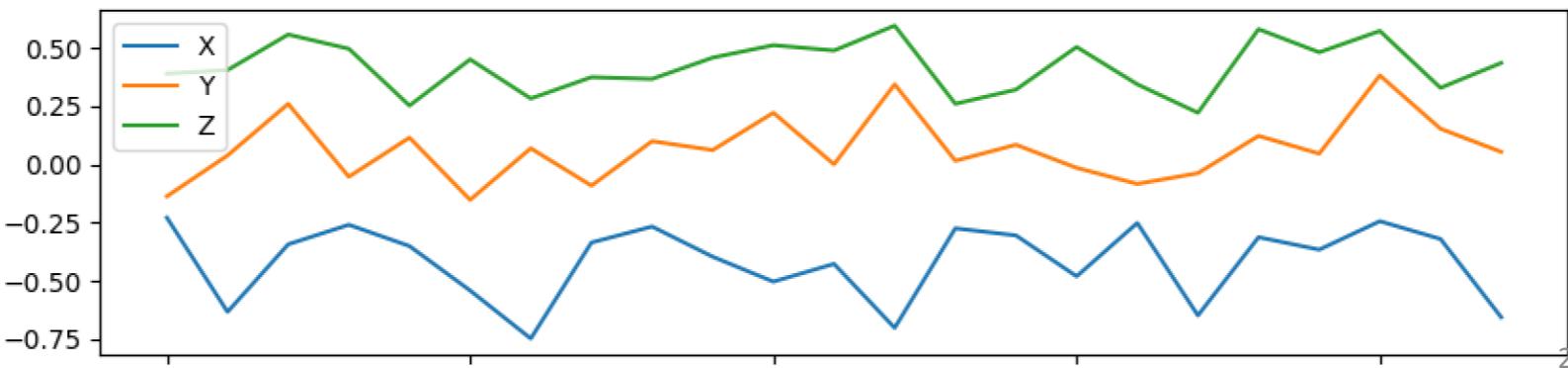


Figure 12. Orientation of Axes of Sensitivity and Polarity of Rotation



# Gyroscope calibration

- 陀螺儀通常有bias (漂移) , 即使靜置也會有讀數
- 校正步驟 : (跟上頁流程一樣)
  - 將IMU擺放在桌上，~~靜置數秒取得大量的讀數~~
  - 將XYZ的讀數取平均值當作offset
  - 量測值須扣除offset，即： $\text{gyro\_x\_calibrated} = \text{gyro\_x\_raw} - \text{offset\_x}$





## 計算偏移量

- 先收集靜態資訊 (EX: 1000筆 or 記錄一分鐘)
- 偏移量(offset)
  - $\text{offset\_x} = \text{average}(\text{accel\_x\_data}); \text{average}(\text{gyro\_x\_data})$
  - $\text{offset\_y} = \text{average}(\text{accel\_y\_data}); \text{average}(\text{gyro\_y\_data})$
  - $\text{offset\_z} = \text{average}(\text{accel\_z\_data}); \text{average}(\text{gyro\_z\_data})$

- 程式碼:

```
acc_x_calibrated = accel_x_raw - offset_x  
acc_y_calibrated = accel_y_raw - offset_y  
acc_z_calibrated = accel_z_raw - offset_z  
  
gyro_x_calibrated = gyro_x_raw - offset_x  
gyro_y_calibrated = gyro_y_raw - offset_y  
gyro_z_calibrated = gyro_z_raw - offset_z
```

; offset



## 儲存量測結果

- python 10dof\_imu\_reading.py >> result.txt

pi@raspberrypi:~/lab3 \$ python 10dof\_imu\_reading.py

|       |       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 00.03 | -1.01 | -0.05 | -0.02 | 00.02 | 00.83 | 02.70 | 29.70 | 35.10 |
| 00.03 | -1.01 | -0.05 | -0.31 | 00.16 | 00.50 | 01.65 | 28.65 | 34.80 |
| 00.03 | -1.01 | -0.05 | -0.35 | 00.01 | 00.54 | 02.70 | 29.40 | 33.30 |
| 00.03 | -1.01 | -0.05 | -0.58 | -0.08 | 00.45 | 03.45 | 28.20 | 35.85 |
| 00.03 | -1.01 | -0.06 | -0.33 | -0.05 | 00.35 | 02.25 | 28.95 | 34.05 |
| 00.03 | -1.01 | -0.06 | -0.62 | 00.13 | 00.33 | 03.00 | 28.95 | 35.10 |
| 00.03 | -1.01 | -0.05 | -0.54 | 00.05 | 00.53 | 02.85 | 28.35 | 34.65 |

acc      gyro      mag

pi@raspberrypi:~/lab3 \$ cat results.txt

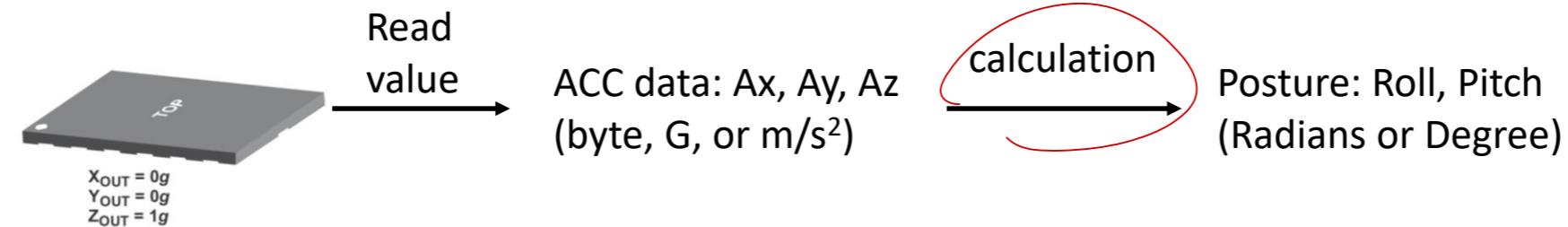
|    | acc               | gyro              | mag               |
|----|-------------------|-------------------|-------------------|
| 1  | 00.03 -1.01 -0.06 | -0.40 00.02 00.73 | 02.85 28.80 34.65 |
| 2  | 00.02 -1.01 -0.06 | -0.29 00.17 00.43 | 02.25 28.65 35.85 |
| 3  | 00.03 -1.01 -0.06 | -0.46 -0.15 00.47 | 02.10 29.55 34.65 |
| 4  | 00.03 -1.01 -0.06 | -0.38 00.12 00.29 | 02.10 28.05 34.95 |
| 5  | 00.02 -1.01 -0.06 | -0.29 -0.11 00.52 | 02.55 29.85 34.20 |
| 6  | 00.03 -1.01 -0.06 | -0.25 -0.06 00.60 | 02.25 29.70 35.10 |
| 7  | 00.03 -1.01 -0.05 | -0.44 00.02 00.44 | 01.80 28.80 34.95 |
| 8  | 00.03 -1.01 -0.06 | -0.48 00.06 00.28 | 03.30 29.85 34.65 |
| 9  | 00.03 -1.01 -0.05 | -0.39 00.18 00.49 | 02.25 28.20 34.80 |
| 10 | 00.03 -1.01 -0.06 | -0.57 00.05 00.27 | 01.35 27.30 34.65 |

平均  
offset



# ACC Brief summary

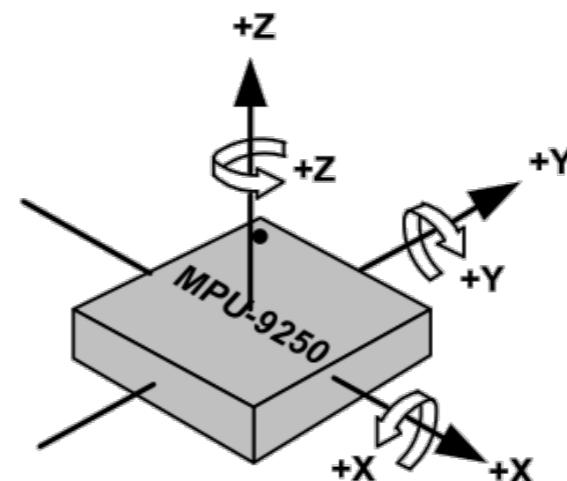
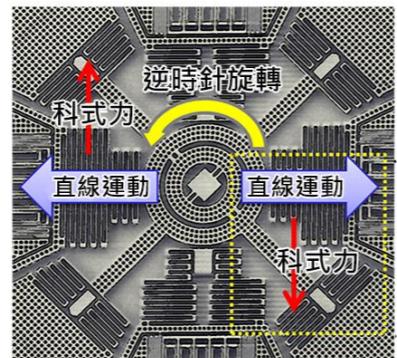
- What can we obtain from accelerometer?
  - 1. Movement along with xyz-axis
    - Unit: byte, G, or m/s<sup>2</sup>
  - 2. Posture (Roll, Pitch)
    - Unit: Radians or Degree





# Gyro Brief summary

- What can we obtain from gyroscope?
  - 1. rotation (angular velocity)
    - Unit: byte, DPS (Degree per Second) *角速度*
  - 2. rotated angle (integral “angular velocity” to “angle”)
    - Unit: degree. You can convert it to rad.





# Quiz 2-1

- Acceleration vs velocity vs distance

$$v = v_0 + at$$

$$s = v_0 t + \frac{1}{2} a t^2$$

- Try to calculate distance based on accelerometer

- With calibration

```
307 if __name__ == "__main__":
308     imu = ICM20948()
309
310     while True:
311         mx, my, mz = imu.read_magnetometer_data()
312         ax, ay, az, gx, gy, gz = imu.read_accelerometer_gyro_data()
313
314         print(f"""
315 Accel: {ax:05.2f} {ay:05.2f} {az:05.2f}
316 Gyro:  {gx:05.2f} {gy:05.2f} {gz:05.2f}
317 Mag:   {mx:05.2f} {my:05.2f} {mz:05.2f}""")
318
319         time.sleep(0.25)
```



## Quiz 2-2

- Rotation (angular velocity) vs rotated angle
$$\nu = \nu_0 + at$$
- Try to calculate rotation angle based on gyroscope
  - With calibration

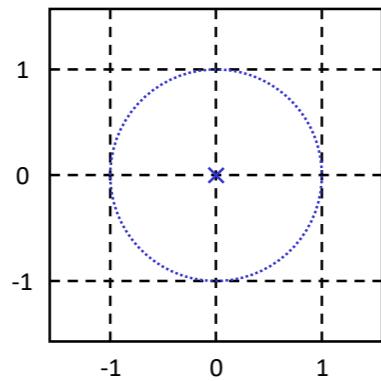
```
307 if __name__ == "__main__":
308     imu = ICM20948()
309
310     while True:
311         mx, my, mz = imu.read_magnetometer_data()
312         ax, ay, az, gx, gy, gz = imu.read_accelerometer_gyro_data()
313
314         print(f"""
315 Accel: {ax:05.2f} {ay:05.2f} {az:05.2f}
316 Gyro:  {gx:05.2f} {gy:05.2f} {gz:05.2f}
317 Mag:   {mx:05.2f} {my:05.2f} {mz:05.2f}""")
318
319         time.sleep(0.25)
```



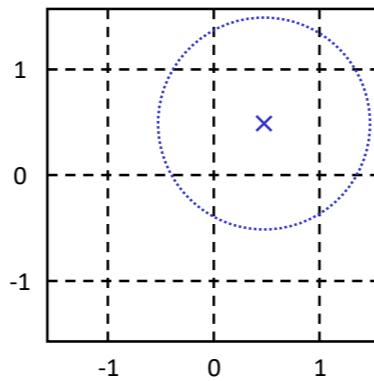
# Magnetometer Calibration

- 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)

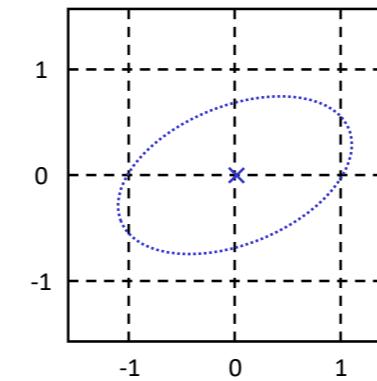
(Perfect)



No distortion

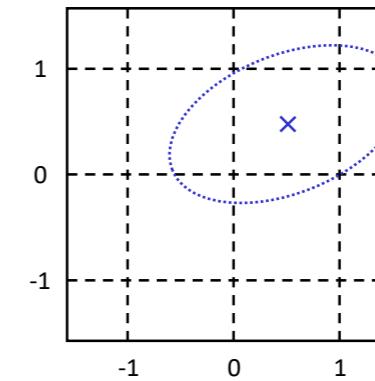


Hard Iron  
distortion



Soft Iron  
distortion

(Reality)



Soft & Hard  
Iron distortion

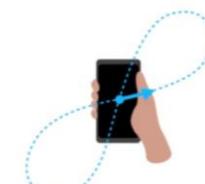
# Compass Calibration

```
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
```

執行期間, 請將磁力計朝向xyz三周轉動360度

```
... -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
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$
```

你可隨時校準藍點的指南針  
像這樣傾斜並移動手機 3 次：



指南針精準度：低

完成

```
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
```



# Discussion 1

- Does your magnetometer have distortion?
  - Record the raw data and plot it. (for XY, YZ, XZ)



Fig source: <https://makersportal.com/blog/calibration-of-a-magnetometer-with-raspberry-pi>



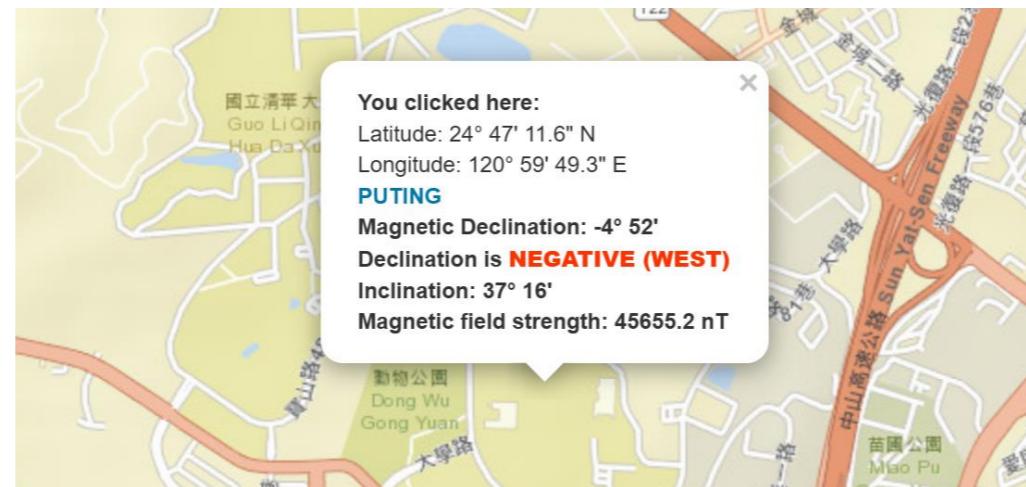
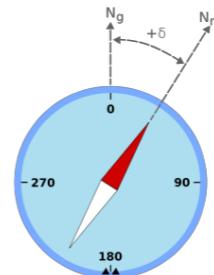
# APP: Compass

- Declination angle

```
194  
195  
196  
197  
198  
199
```

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

-4°32''



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



## Compass code: calculate heading

```

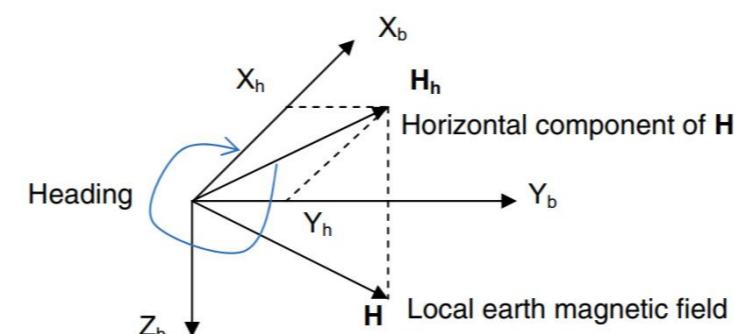
220     def getX(self):
221         self.X = (self.read_word_2c(HMC5883L_DO_X_H, rf=0) - self.Xoffset) * self.scale
222         return self.X
223
224     def getY(self):
225         self.Y = (self.read_word_2c(HMC5883L_DO_Y_H, rf=0) - self.Yoffset) * self.scale
226         return self.Y
227
228     def 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
    
```

Declination angle

### □ Heading:

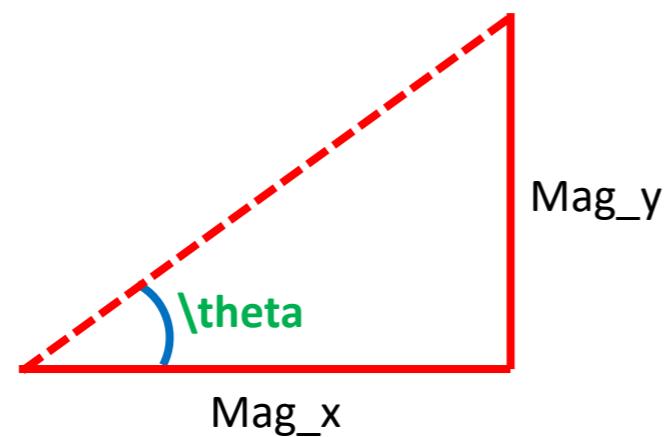
- 1. measure **two orthogonal components** of the magnetic vector (**mx, my**)
- 2. calculate the heading as the **arctangent** of their ratio

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

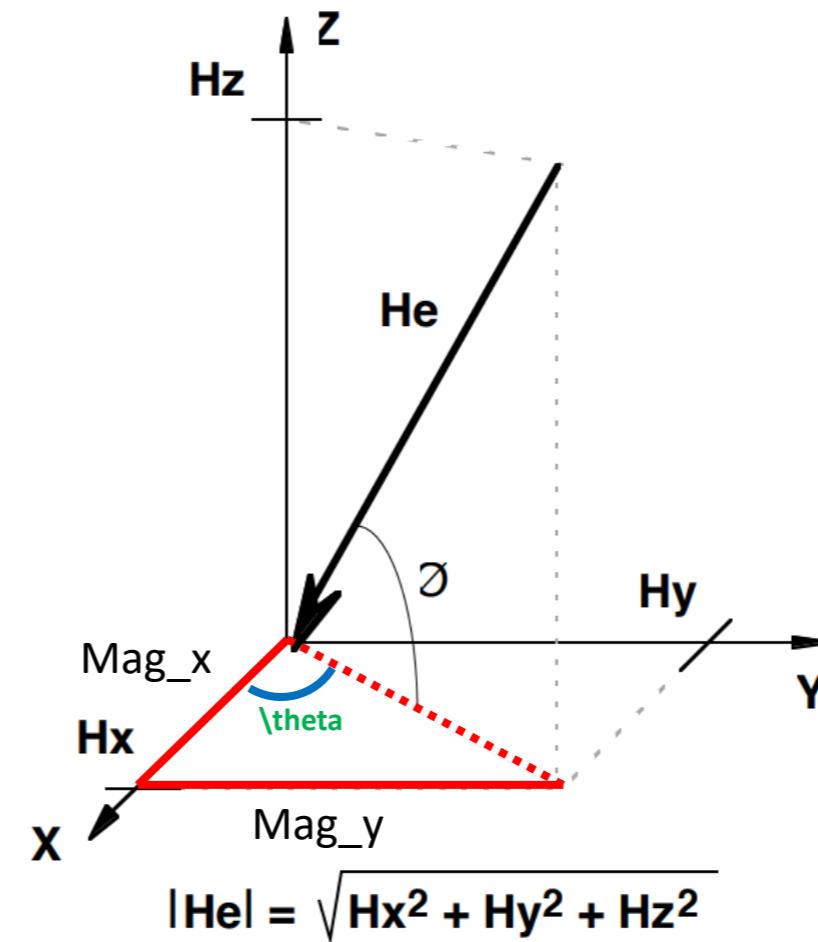




## Compass heading



$$\theta = \arctan(y/x)$$



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



# App: Fall detection

## Tracking of fall detection using IMU sensor: An IoHT application

[VB Semwal, A Kumar, P Nargesh, V Soni](#) - Machine Learning, Image ..., 2023 - Springer

... developed the methodology to detect the fall of a person ... fall recovery capability & fall diagnosis information can be utilized to avoid the fall and to provide assistant after fall detection of ...

☆ 儲存 引用 被引用 7 次 相關文章 全部共 2 個版本 ☰

## Synthetic IMU datasets and protocols can simplify fall detection experiments and optimize sensor configuration

[J Tang, B He, J Xu, T Tan, Z Wang...](#) - IEEE transactions on ..., 2024 - ieeexplore.ieee.org

... As per our investigation, existing IMU datasets in the field of fall detection suffer from issues such as limited data volume and inconsistent experimental paradigms. In contrast, an ...

☆ 儲存 引用 相關文章 全部共 7 個版本

## Development of LSTM Model for Fall Prediction Using IMU

[V Chandramouleesvar, ME Swetha...](#) - Advances in Science ..., 2023 - Trans Tech Publ

... Existing fall detection and prediction systems focus on physiological factors such as gait, vision, cognition, and medical history of individuals. Serious injuries such as broken bones or a ...

☆ 儲存 引用 被引用 1 次 相關文章

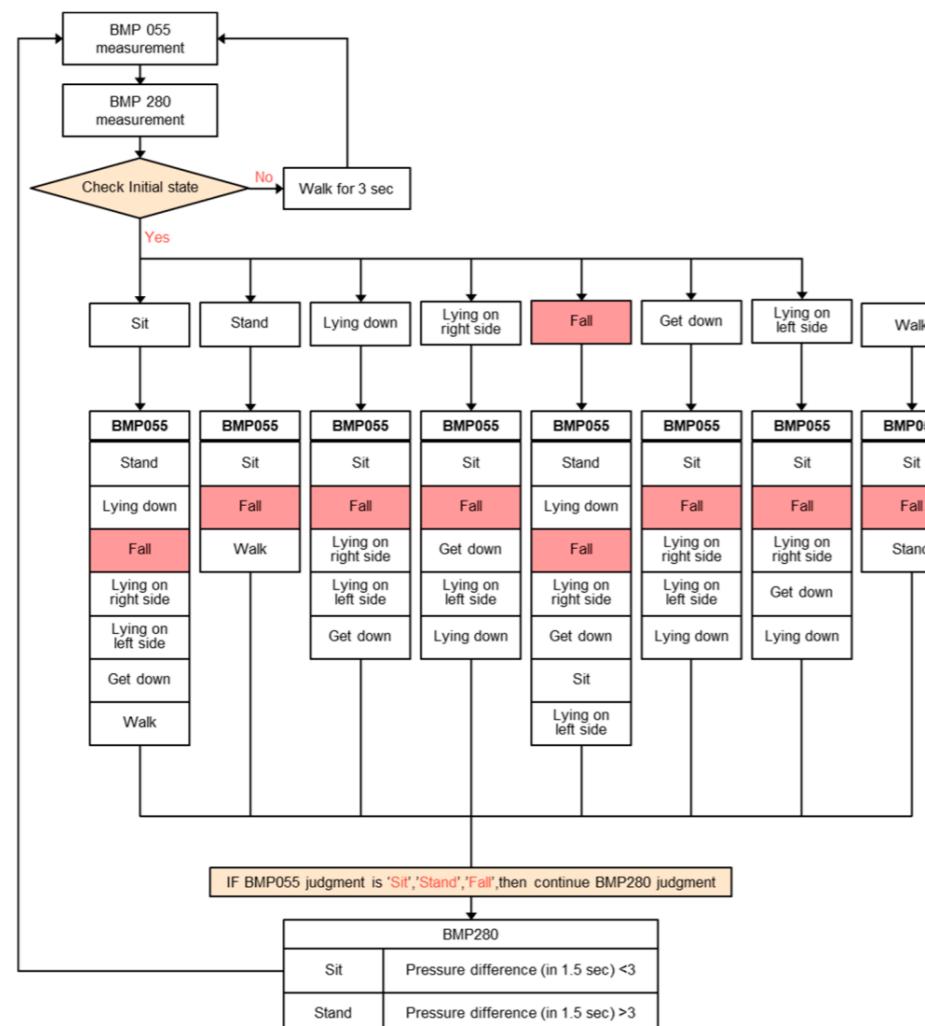
## [HTML] TinyFallNet: a lightweight pre-impact fall detection model

[B Koo, X Yu, S Lee, S Yang, D Kim, S Xiong, Y Kim](#) - Sensors, 2023 - mdpi.com

... -classification models to preimpact fall detection using IMU and showed that additional tuning ... learning models based on IMU and the development of applications based on IMU data. ...

☆ 儲存 引用 被引用 2 次 相關文章 全部共 10 個版本 ☰

# Fall detection paper



*Paper: Fall Recognition Based on an IMU Wearable Device and Fall Verification through a Smart Speaker and the IoT*

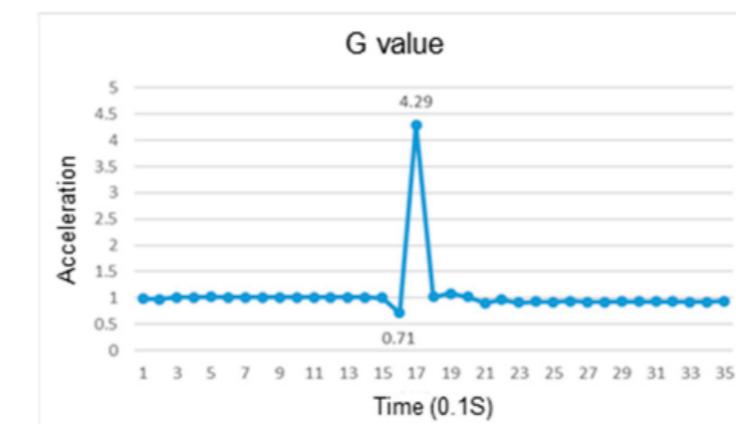
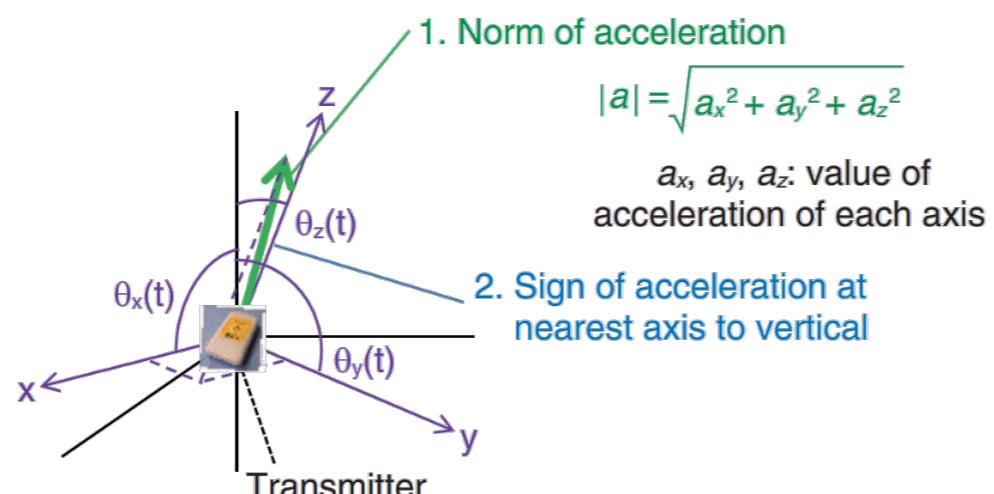


Fig. The change in the resultant force of falling backward while sitting on a chair.



# Quiz 3

- Fall detection with notification
  - Use the total acceleration:
    - $\text{total\_acc} = \sqrt{\text{acc}_x^2 + \text{acc}_y^2 + \text{acc}_z^2}$
  - Observe the G value when falling down
  - Turn on LED if occurred

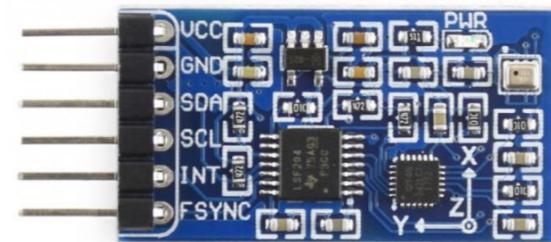


Hint: Use python to calculate pow and sqrt

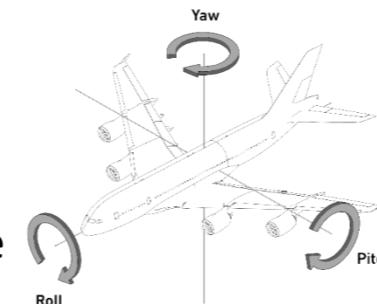
```
import numpy  
x = 4  
  
y = pow(x,2)  
# y=16  
  
z = numpy.sqrt(x)  
# z=2  
  
print y  
print z
```



# IMU summary

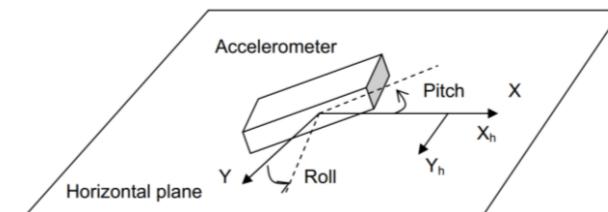


1. Accelerometer: Acceleration(加速度) -> Roll, Pitch (傾斜姿態)
2. Gyroscope: Angular velocity(角速度) -> Rotated angle (旋轉角度)
3. Magnetometer: Magnetic field(磁場) -> Heading (方位)
4. Barometer: Pressure(大氣壓力) -> Altitude (高度)



## IMU information

- Roll, Pitch: Accelerometer + Gyroscope
- Heading: Magnetometer
- Altitude: Barometer



## Sensor Fusion:

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



# Summary

- Labs: process IMU data
  - The information of IMU
  - Datasheet and code configuration
  - IMU applications
- Write down the answer for discussion, upload to e-campus.  
Deadline (before next class): 13:10, 3/28(Fri.)
  - Discussion1: Does your magnetometer have distortion?
  - 書面問答, 請上傳到e3
- Quiz 1-3:
  - 1. Convert pressure to altitude
  - 2. Calculate distance and rotated angle
  - 3. Fall detection with notification
  - 請於課堂上完成, 找助教demo