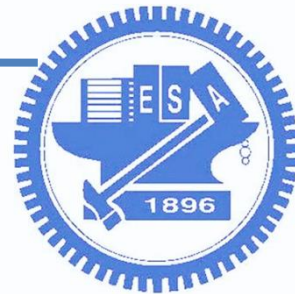


Outline

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

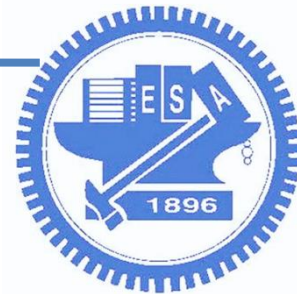


Last week

- 1. 寫Python程式控制GPIO介面
- 2. 練習超音波、溫溼度、電子零件的使用方式
- 3. 超音波、溫溼度的運作原理

This 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



嵌入式系統設計概論與實作

曾煜棋、吳昆儒、張凌燕

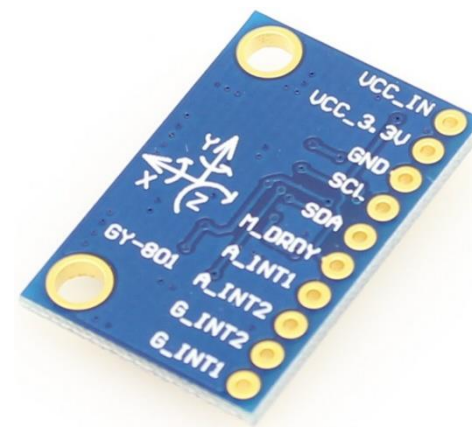
National Chiao Tung University

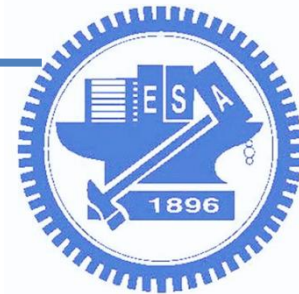


Outline

- 嵌入式應用: 人體活動偵測
 - 加速度、陀螺儀...等

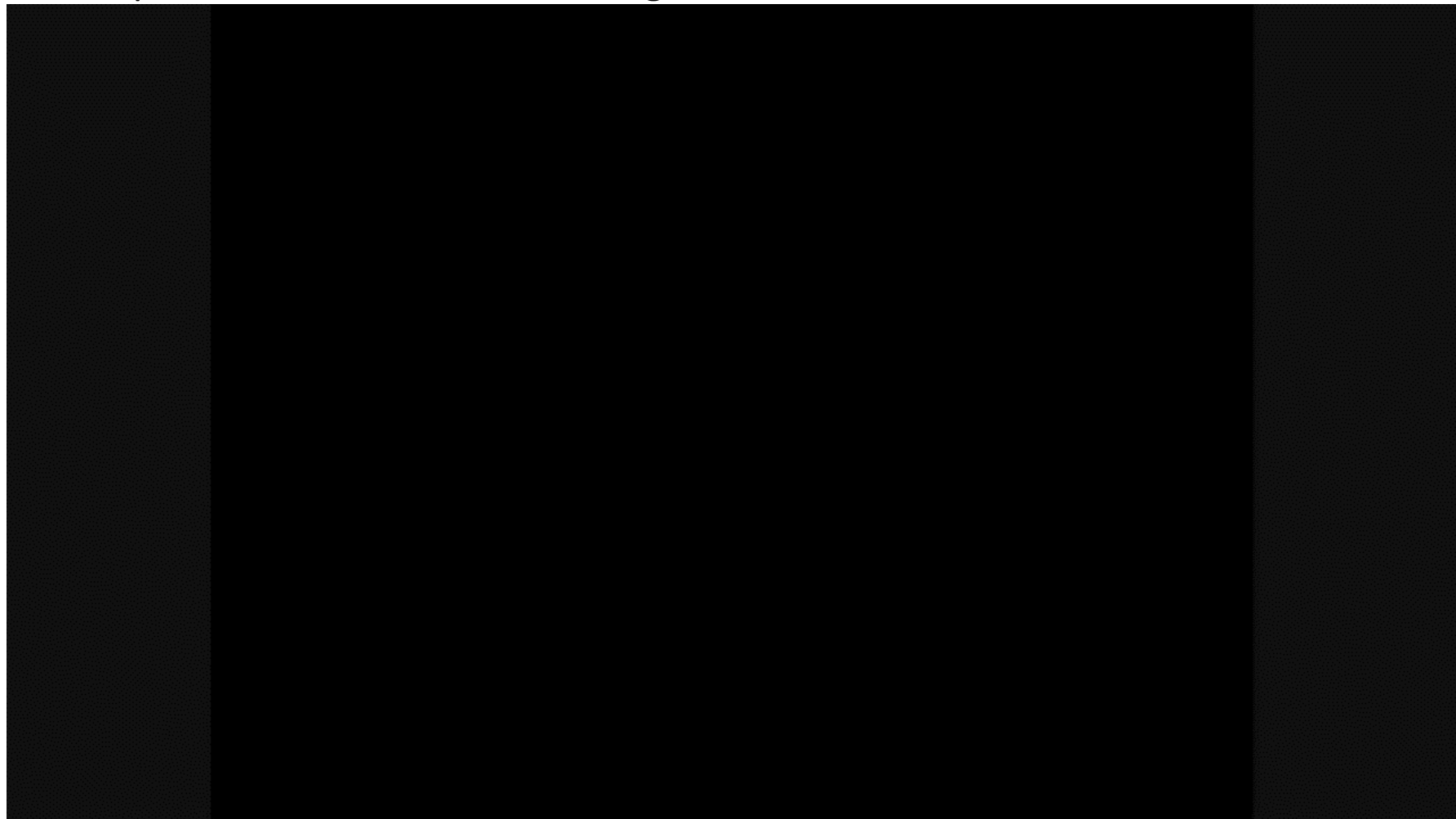
- GY801 (I2C sensor)
 - 3-axis Accelerometer, Gyroscope, magnetometer and pressure
 1. ADXL345 : Accelerometer
 2. L3G4200 : Gyroscope
 3. HMC5883 : Magnetometer
 4. BMP085 : Pressure

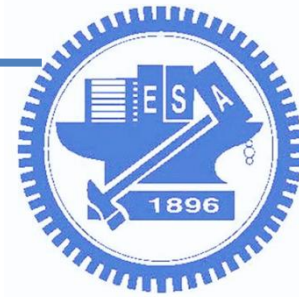




活動偵測應用

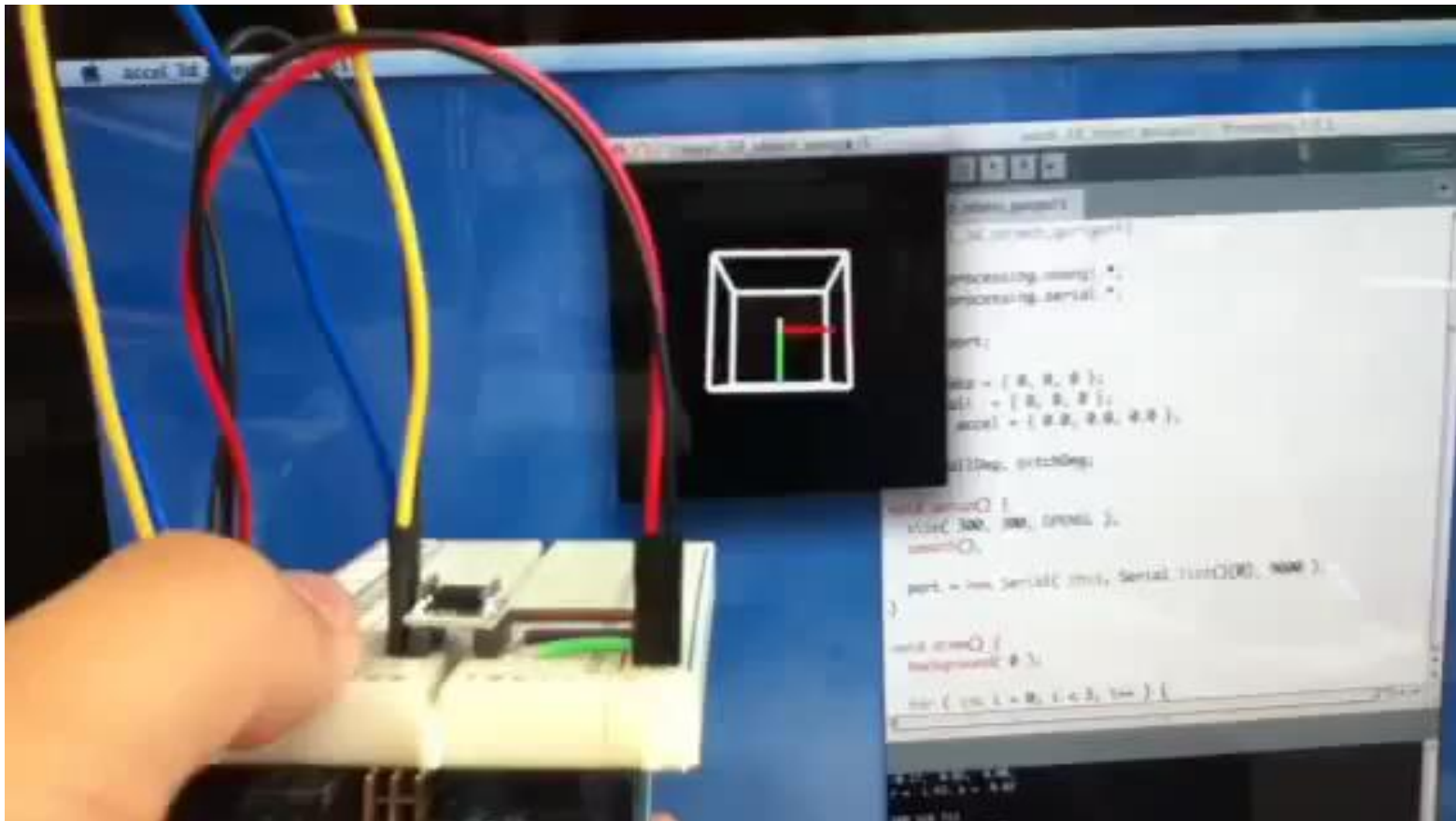
- Open Source IMU and AHRS Algorithm with x-IMU





活動偵測應用

- 3D Box on screen rotated around its roll and pitch axes through an acceleration sensor

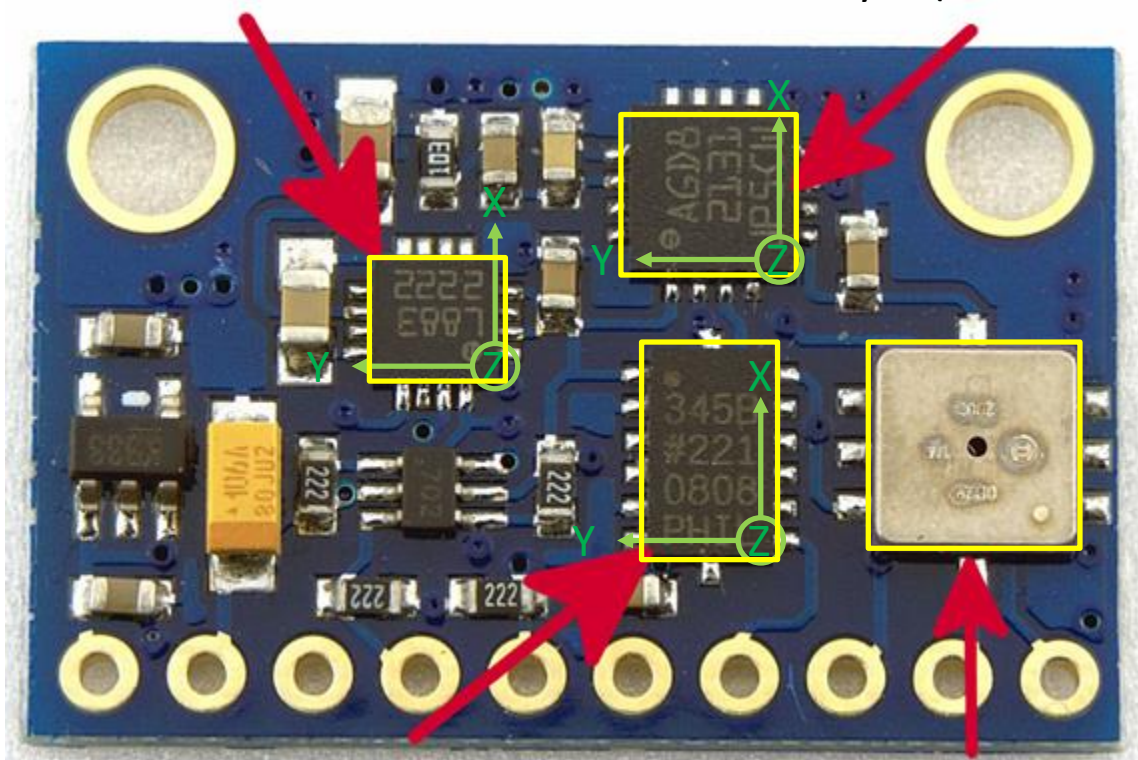




GY-801 (10 DoF sensor)

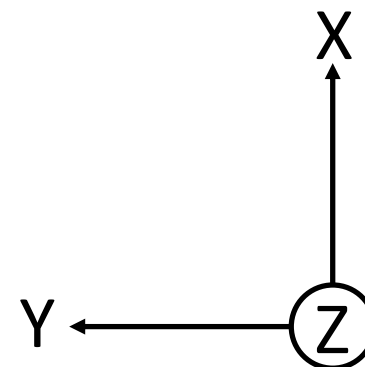
Compass (HMC5883L)

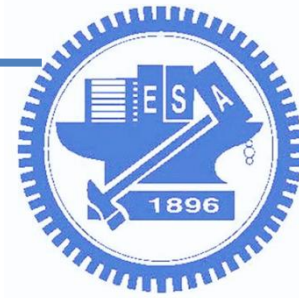
Gyro (L3G4200D)



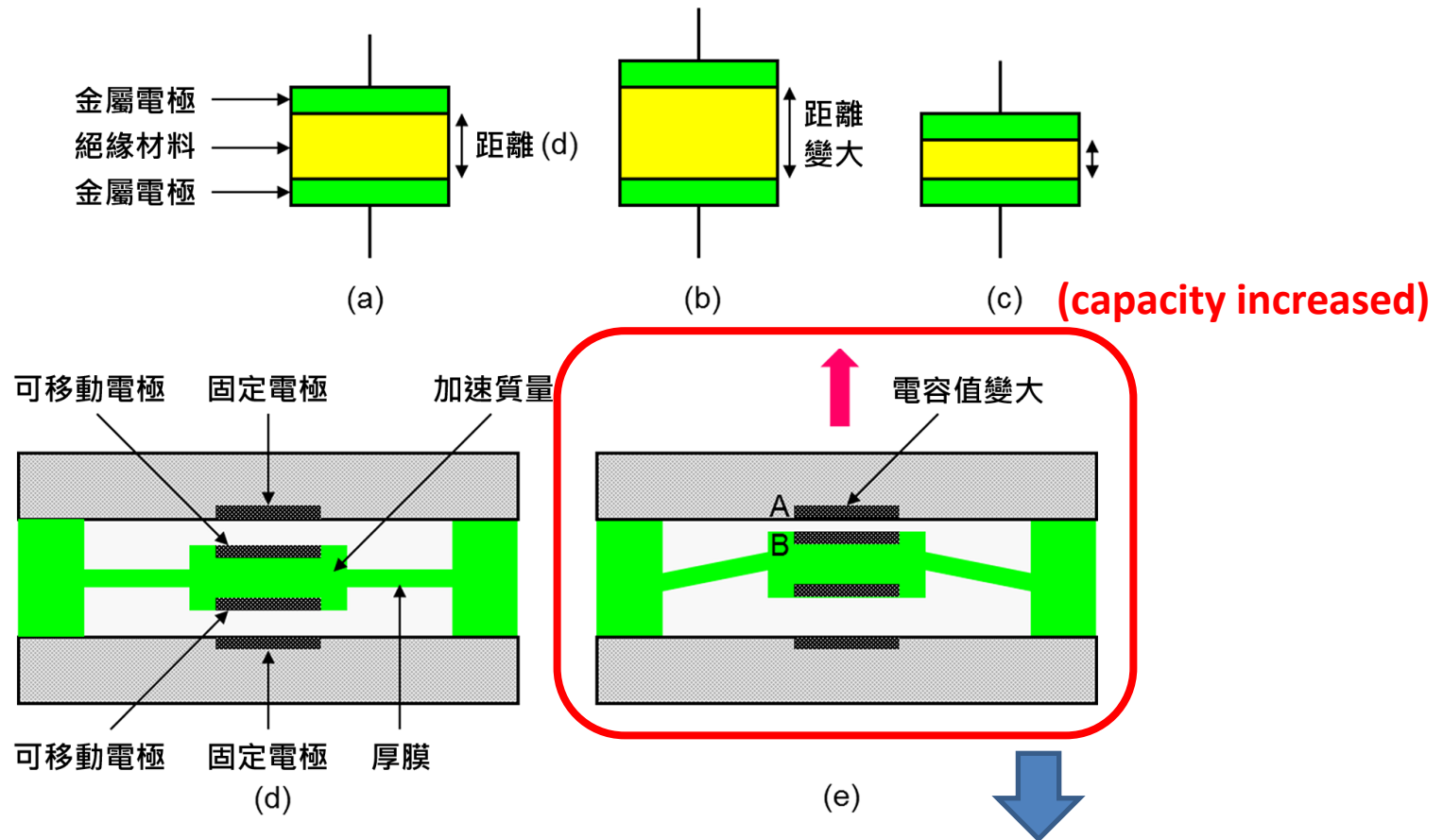
G-sensor (ADXL345)

Pressure (BMP085)



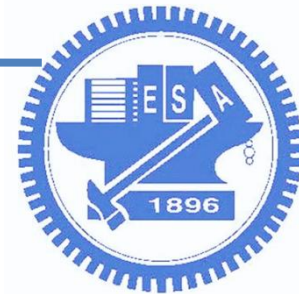


Sensor - Accelerometer

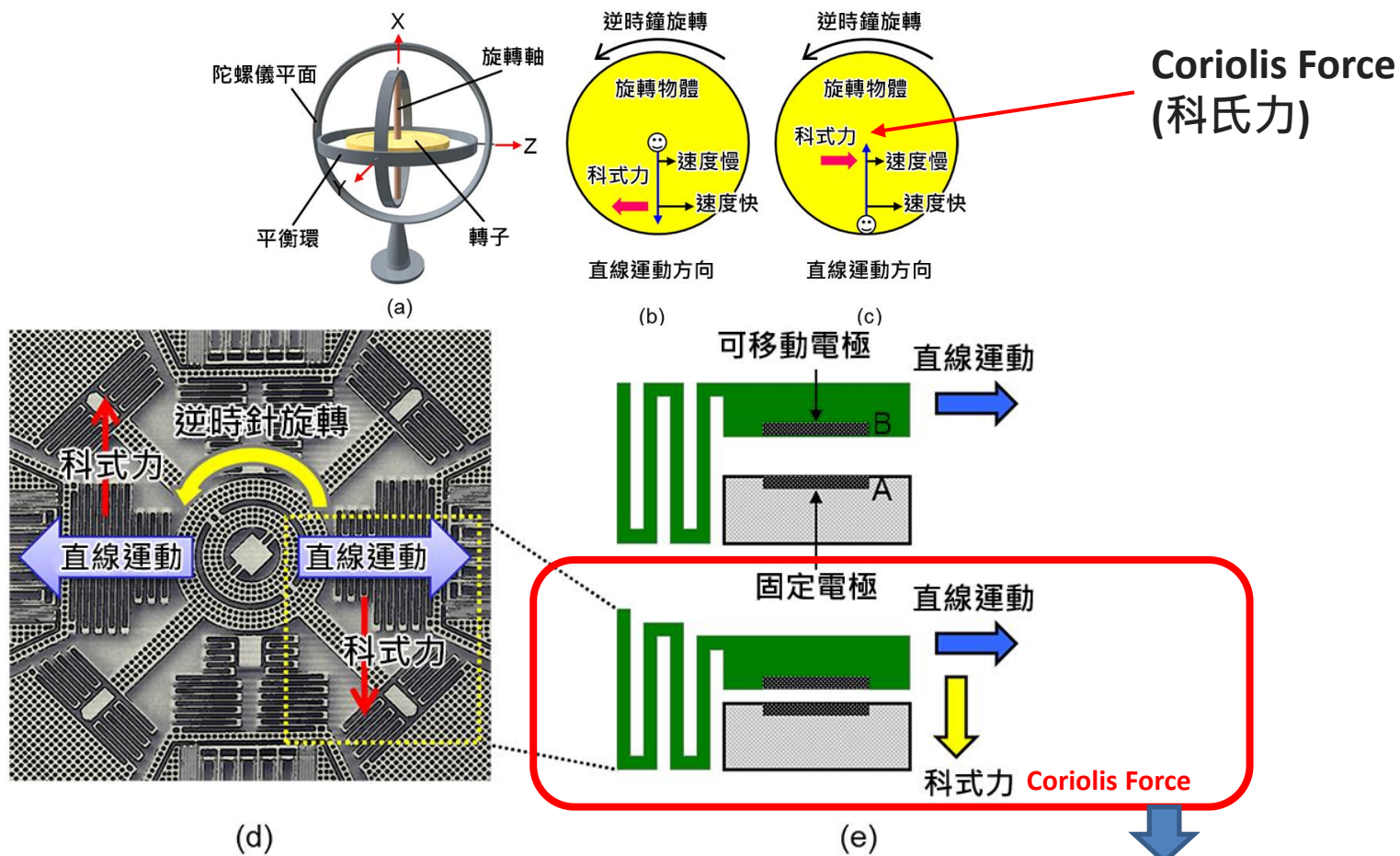


When you move the sensor, the distance (**capacity**) is changed.

We **use capacity to calculate acceleration**.

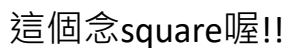


Sensor - Gyro

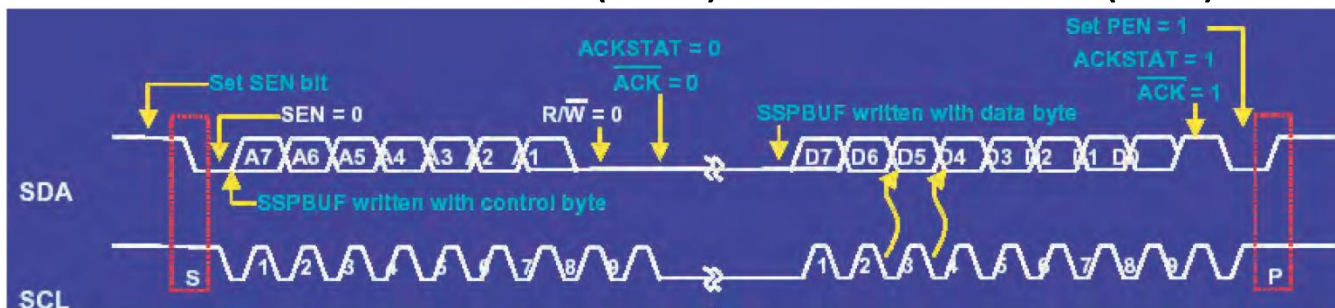


When you rotate the sensor, Coriolis Force change the distance (**capacity**).

We **use capacity to calculate Angular velocity (角速度)**.



- I2C全名為Inter-IC，它是一種半雙工同步多組設備匯流排，只需要兩條信號線：串列資料線 (SDA) 及串列時脈線 (SCL)。



- 在傳送資料過程中共有三種類型信號，分別是：開始信號、結束信號和應答信號。
 - 開始信號：SCL為高電位時，SDA由高電位降為低電位，開始傳送資料。
 - 結束信號：SCL為高電位時，SDA由低電位升為高電位，結束傳送資料。
 - 應答信號：收到 8bit 資料後，向發送資料的IC發出特定的低電位脈衝
- 資料讀取方式：
 - 當 SCL 由低電位升為高電位時，讀取 SDA 的資料。



- [illegible]



The screenshot shows the Raspberry Pi Software Configuration Tool (raspi-config) running in a terminal window titled "(COM8) [80x24]". The menu options are: 連線(C), 編輯(E), 檢視(V), 視窗(W), 選項(O), 說明(H). The main menu lists several configuration options:

- P1 Camera
- P2 SSH
- P3 VNC
- P4 SPI
- P5 I2C** (highlighted)
- P6 Serial
- P7 1-Wire
- P8 Remote GPIO

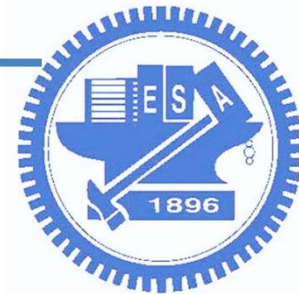
Each option has a corresponding "Enable/Disable" status. The "P5 I2C" option is currently selected, and its status is "g". At the bottom of the screen, there are two navigation options: "<Select>" and "<Back>".



Enable I2C on RaspPI

- Choose Yes





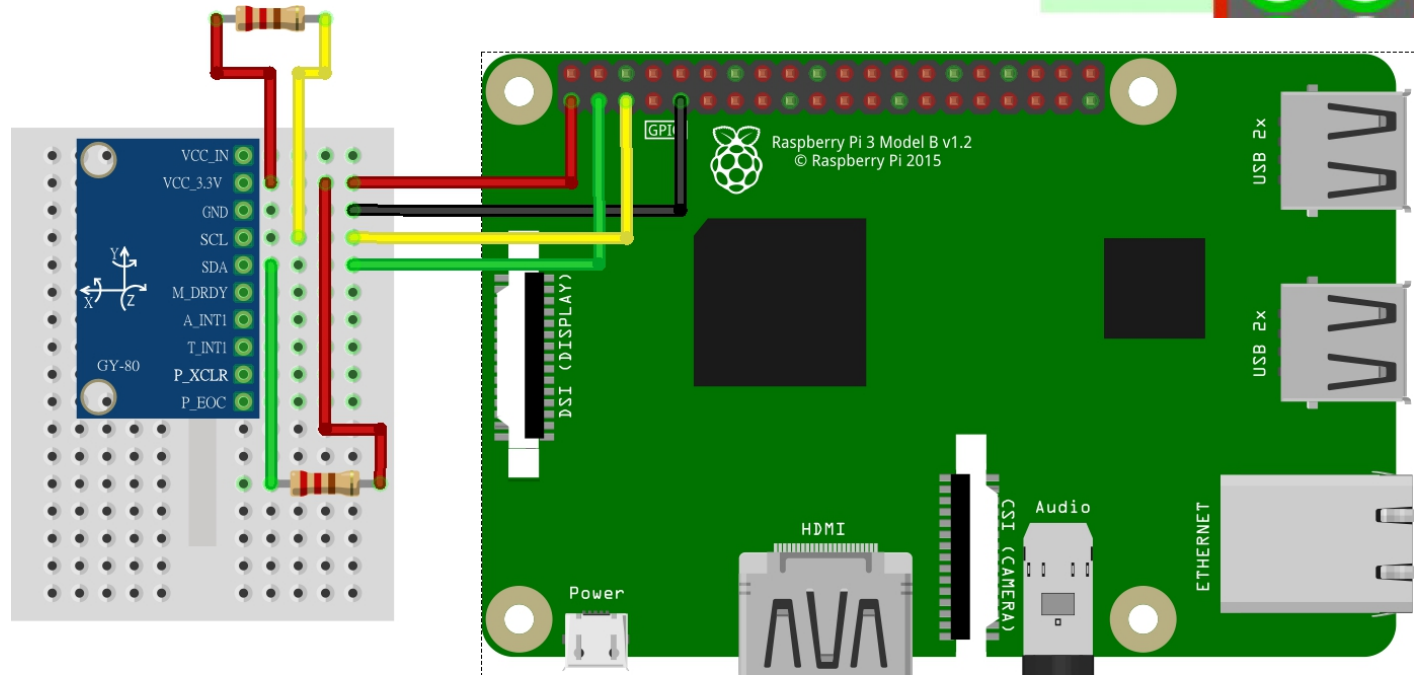
Enable I2C on RaspPI

- I2C is enabled



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



Pi Model B/B+	
3V3 Power	1 2
GPIO2 SDA1 I2C	3 4
GPIO3 SCL1 I2C	5 6
GPIO4	7 8
Ground	9 10
GPIO17	11 12
5V Power	
5V Power	
Ground	
GPIO14 UART0_TXD	
GPIO15 UART0_RXD	
GPIO18 PCM_CLK	

fritzing

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*`

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

□ Command 2

■ `sudo i2cdetect -y 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

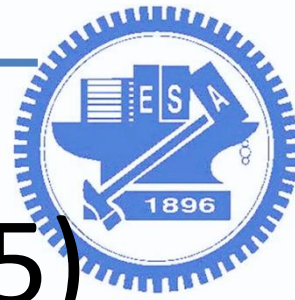
■ `sudo ls -al /dev/*i2c*`

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

□ Command 2

■ `sudo i2cdetect -y 1`

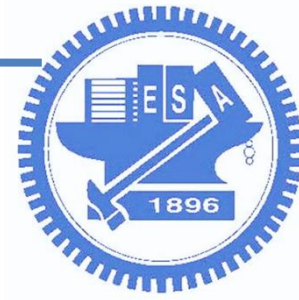
```
pi@raspberrypi:~$ sudo i2cdetect -y 1  
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
30: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



1. Accelerometer (ADXL345)

- ADXL345
 - Chipset : ADXL345
 - Power : 3 ~ 5V
 - Protocol : I2C/SPI
 - Range : $\pm 2g$ 、 $\pm 4g$ 、 $\pm 8g$ 、 $\pm 16g$

Parameter	Test Conditions	Min	Typ ¹	Max	Unit
OUTPUT RESOLUTION	Each axis				
All g Ranges	10-bit resolution		10		Bits
$\pm 2 g$ Range	Full resolution		10		Bits
$\pm 4 g$ Range	Full resolution		11		Bits
$\pm 8 g$ Range	Full resolution		12		Bits
$\pm 16 g$ Range	Full resolution		13		Bits
SENSITIVITY	Each axis				
Sensitivity at X_{OUT} , Y_{OUT} , Z_{OUT}	All g -ranges, full resolution	230	256	282	LSB/ g
	$\pm 2 g$, 10-bit resolution	230	256	282	LSB/ g
	$\pm 4 g$, 10-bit resolution	115	128	141	LSB/ g
	$\pm 8 g$, 10-bit resolution	57	64	71	LSB/ g
	$\pm 16 g$, 10-bit resolution	29	32	35	LSB/ g
Sensitivity Deviation from Ideal	All g -ranges		± 1.0		%
Scale Factor at X_{OUT} , Y_{OUT} , Z_{OUT}	All g -ranges, full resolution	3.5	3.9	4.3	mg/LSB
	$\pm 2 g$, 10-bit resolution	3.5	3.9	4.3	mg/LSB
	$\pm 4 g$, 10-bit resolution	7.1	7.8	8.7	mg/LSB
	$\pm 8 g$, 10-bit resolution	14.1	15.6	17.5	mg/LSB
	$\pm 16 g$, 10-bit resolution	28.6	31.2	34.5	mg/LSB
Sensitivity Change Due to Temperature			± 0.01		%/ $^{\circ}C$



1. Accelerometer code

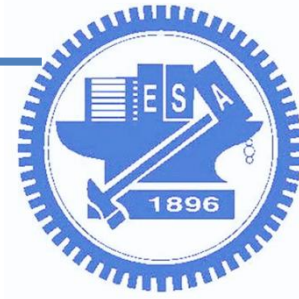
□ Download ADXL library

```
sudo apt-get install git build-essential python-dev
git clone https://github.com/adafruit/Adafruit_Python_ADXL345.git
cd Adafruit_Python_ADXL345
sudo python setup.py install

cd examples/
sudo python simpletest.py
```

□ Execution results

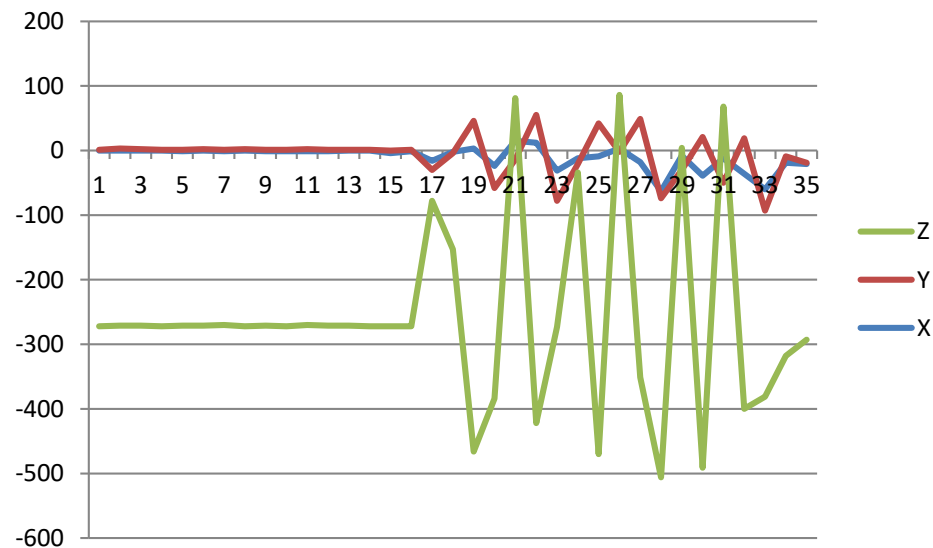
```
pi@raspberrypi:~/gy80/1.acc/examples$ sudo python simpletest.py
Printing X, Y, Z axis values, press Ctrl-C to quit...
X=-13, Y=-2, Z=-272
X=-12, Y=-3, Z=-272
X=-12, Y=-2, Z=-272
X=-11, Y=-3, Z=-271
X=-12, Y=-3, Z=-270
X=-12, Y=-1, Z=-272
```

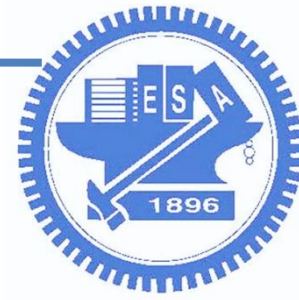


1. Accelerometer code

Execution results (by excel)

```
pi@raspberrypi:~/gy80/1.acc/examples$ sudo python simpletest.py
Printing X, Y, Z axis values, press Ctrl-C to quit...
X=-13, Y=-2, Z=-272
X=-12, Y=-3, Z=-272
X=-12, Y=-2, Z=-272
X=-11, Y=-3, Z=-271
X=-12, Y=-3, Z=-270
X=-12, Y=-1, Z=-272
```





1. Accelerometer code

□ Sample code (simplectest.py)

Adafruit_ADXL345/ADXL345.py

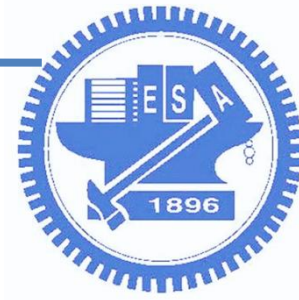
```
import time
import Adafruit_ADXL345
accel = Adafruit_ADXL345.ADXL345()

print('Printing X, Y, Z axis values, press Ctrl-C to quit...')

while True:
    # Read the X, Y, Z axis acceleration values and print them.
    x, y, z = accel.read()
    print('X={0}, Y={1}, Z={2}'.format(x, y, z))

    # Wait half a second and repeat.
    time.sleep(0.5)
```

print the X, Y, Z axis acceleration values every half second.



1. Accelerometer code

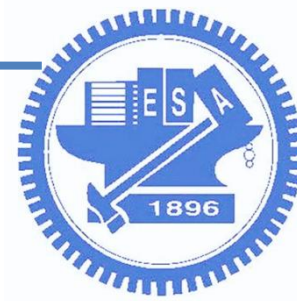
□ Adafruit_ADXL345/ADXL345.py

```
ADXL345_ADDRESS    = 0x53
ADXL345_REG_DEVID   = 0x00 # Device ID
ADXL345_REG_DATA0   = 0x32 # X-axis data 0 (6 bytes for X/Y/Z)
ADXL345_REG_POWER_CTL = 0x2D # Power-saving features control
...
```

```
def read(self):
    """Read the current value of the accelerometer and return it as a tuple
    of signed 16-bit X, Y, Z axis values.
    """
    raw = self._device.readList(ADXL345_REG_DATA0, 6)
    return struct.unpack('<hhh', raw)
```

Referring to datasheet, read 6 bytes on register location
(ADXL345_REG_DATA0, 0x32) then transform to x/y/z value

**When accessing I2C sensor, we have to write code based on datasheet.
The code is already done for you.**



1. Accelerometer spec.

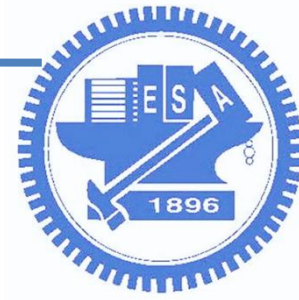
REGISTER MAP

Table 19.

Address		Name	Type	Reset Value	Description
Hex	Dec				
0x32	50	DATAx0	R	00000000	X-Axis Data 0
0x33	51	DATAx1	R	00000000	X-Axis Data 1
0x34	52	DATAY0	R	00000000	Y-Axis Data 0
0x35	53	DATAY1	R	00000000	Y-Axis Data 1
0x36	54	DATAz0	R	00000000	Z-Axis Data 0
0x37	55	DATAz1	R	00000000	Z-Axis Data 1

Register 0x32 to Register 0x37—DATAx0, DATAx1, DATAY0, DATAY1, DATAz0, DATAz1 (Read Only)

These six bytes (Register 0x32 to Register 0x37) are eight bits each and hold the output data for each axis. Register 0x32 and Register 0x33 hold the output data for the x-axis, Register 0x34 and Register 0x35 hold the output data for the y-axis, and Register 0x36 and Register 0x37 hold the output data for the z-axis. The output data is twos complement, with DATAx0 as the least significant byte and DATAx1 as the most significant byte, where x represent X, Y, or Z. The DATA_FORMAT register (Address 0x31) controls the format of the data. It is recommended that a multiple-byte read of all registers be performed to prevent a change in data between reads of sequential registers.

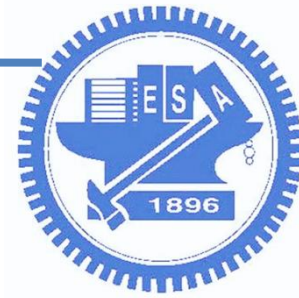


1. Accelerometer code

- In default measurement range, it should report +/-2G (default, range=2), but the code shows the following data:

```
pi@raspberrypi:~/gy80/1.acc/examples$ sudo python simpletest.py
Printing X, Y, Z axis values, press Ctrl-C to quit...
X=-13, Y=-2, Z=-272
X=-12, Y=-3, Z=-272
X=-12, Y=-2, Z=-272
X=-11, Y=-3, Z=-271
X=-12, Y=-3, Z=-270
X=-12, Y=-1, Z=-272
```

- **How to transform the output to +/-2G?**

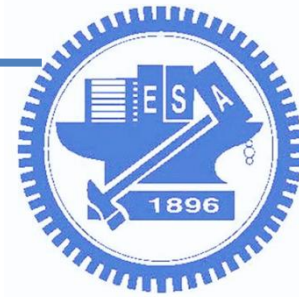


1. Accelerometer code

- Hint: according to datasheet, we can use the following formula to convert output (Raw_{Accel}) to $\pm 2G$.

Parameter	Test Conditions	Min	Typ ¹	Max	Unit
OUTPUT RESOLUTION	Each axis				
All g Ranges	10-bit resolution		10		Bits
$\pm 2 g$ Range	Full resolution		10		Bits
SENSITIVITY	Each axis				
Sensitivity at X_{OUT} , Y_{OUT} , Z_{OUT}	All g -ranges, full resolution	230	256	282	LSB/ g
	$\pm 2 g$, 10-bit resolution	230	256	282	LSB/ g

$$G_{Accel} = Raw_{Accel} / Sensitivity$$

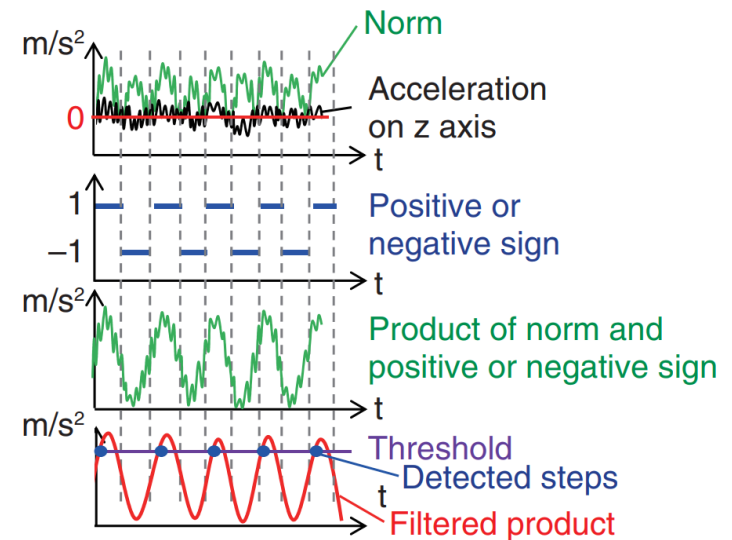
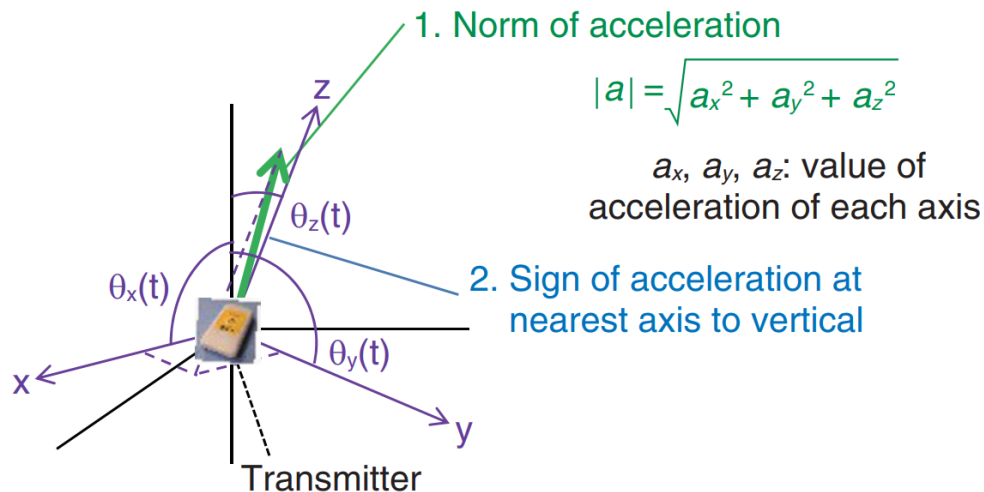


Quiz 1

□ Calculate the norm of acceleration $|a|$

□ Formula:

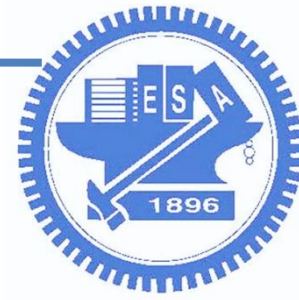
$$|a| = \sqrt{x^2 + y^2 + z^2}$$



Application: step detection

Gait Analysis Using a Wearable T-shirt Type Sensor

<https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201604ra1.html>

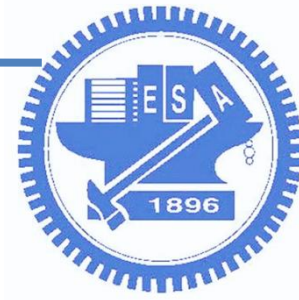


Quiz 1

- Hold the sensor, do not move for 5 seconds, then **turn on LED!**
Hint: use the norm of acceleration $|a|$, and set a variance.
 - If the norm does not exceed the variance for 5 second, we can treat it as stable.

Requirements

1. Convert output to +/-2G
2. Calculate norm



Quiz 1

- Use python to calculate pow and sqrt

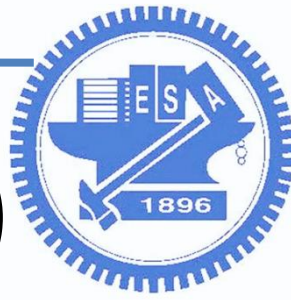
```
import numpy
x = 4

y = pow(x,2)
# y=16

z = numpy.sqrt(x)
# z=2

print y
print z
```

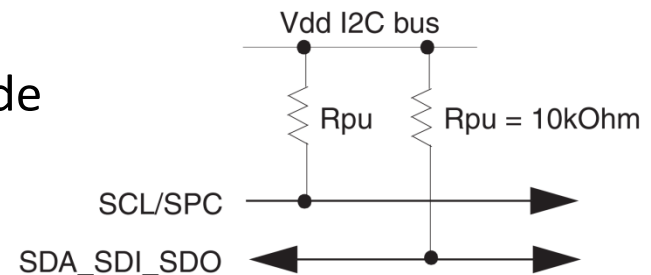
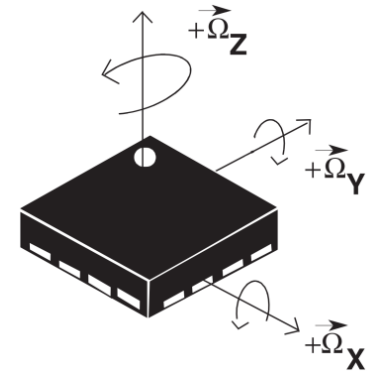
```
(COM4) [80x24]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)
pi@raspberrypi:~$ python norm_a.py
16
2.0
pi@raspberrypi:~$
```



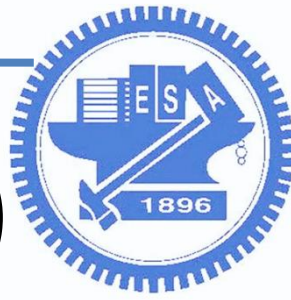
2. Gyroscope (L3G4200D)

□ Features

- 3-Axis angular rate sensor (yaw, pitch, and roll)
- Supports I2C and SPI communications
- Three selectable scales:
250/500/2000 degrees/sec (dps)
- High shock survivability
- Embedded temperature sensor -40 to +185 °F
(-40 to + 85 °C)
- Embedded power-down and sleep mode
- 16 bit-rate value data output
- 8-bit temperature data output



Pull-up to be added when I2C interface is used



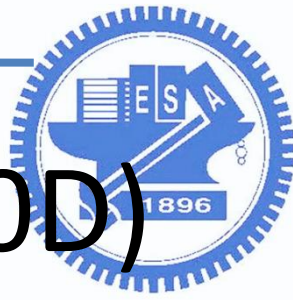
2. Gyroscope (L3G4200D)

- Download the library

```
cd ~  
git clone https://github.com/bashardawood/L3G4200D-Python  
cd L3G4200D-Python  
sudo python gyro.py
```

- Execution

```
pi@raspberrypi:~/L3G4200D-Python$ sudo python gyro.py  
-1013      -256      1602  
  946      -683      -195  
  943      -681      -198  
  947      -683      -193  
  940      -679      -194  
  947      -672      -192  
  935      -680      -193  
  948      -514      -192
```



2. Gyroscope code (L3G4200D)

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

```
from time import sleep  
import smbus  
import string
```



Load library

```
def getSignedNumber(number):  
    if number & (1 << 15):  
        return number | ~65535  
    else:  
        return number & 65535
```

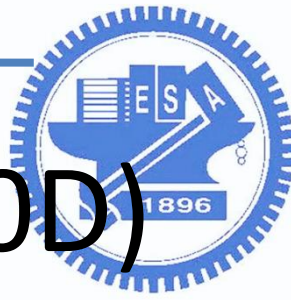


2's complement

```
i2c_bus=smbus.SMBus(1)  
i2c_address=0x69  
i2c_bus.write_byte_data(i2c_address,0x20,0x0F)  
i2c_bus.write_byte_data(i2c_address,0x23,0x20)
```



Set register address
(based on datasheet)



2. Gyroscope code (L3G4200D)

```
while True:
```

```
    i2c_bus.write_byte(i2c_address,0x28)
    X_L = i2c_bus.read_byte(i2c_address)
    i2c_bus.write_byte(i2c_address,0x29)
    X_H = i2c_bus.read_byte(i2c_address)
    X = X_H << 8 | X_L
```

```
    ....
```

```
    X = getSignedNumber(X)
    Y = getSignedNumber(Y)
    Z = getSignedNumber(Z)
```

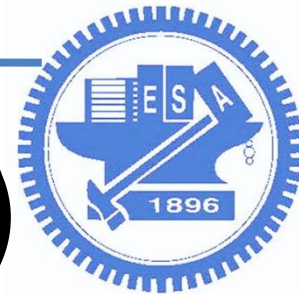
```
    print string.rjust(`X`, 10),
    print string.rjust(`Y`, 10),
    print string.rjust(`Z`, 10)
    sleep(0.02)
```

Read 1st and 2nd byte, then
combine byte value

2's complement
(def getSignedNumber(number):)

Put space before value

**When accessing I2C sensor, we have to write code based on datasheet.
The code is already done for you.**



2. Gyroscope (L3G4200D)

```
i2c_bus.write_byte_data(i2c_address,0x20,0x0F)
```

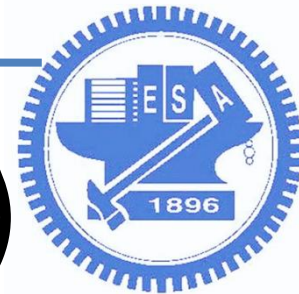
CTRL_REG1 (20h)

Table 20. CTRL_REG1 register

DR1	DR0	BW1	BW0	PD	Zen	Yen	Xen
-----	-----	-----	-----	----	-----	-----	-----

Table 21. CTRL_REG1 description

DR1-DR0	Output Data Rate selection. Refer to Table 22
BW1-BW0	Bandwidth selection. Refer to Table 22
PD	Power down mode enable. Default value: 0 (0: power down mode, 1: normal mode or sleep mode)
Zen	Z axis enable. Default value: 1 (0: Z axis disabled; 1: Z axis enabled)
Yen	Y axis enable. Default value: 1 (0: Y axis disabled; 1: Y axis enabled)
Xen	X axis enable. Default value: 1 (0: X axis disabled; 1: X axis enabled)



2. Gyroscope (L3G4200D)

```
i2c_bus.write_byte_data(i2c_address,0x23,0x20)
```

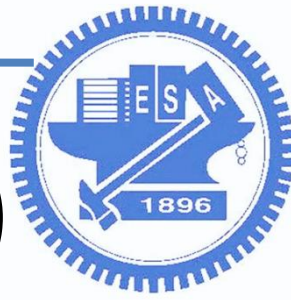
CTRL_REG4 (23h)

Table 30. CTRL_REG4 register

BDU	BLE	FS1	FS0	-	ST1	ST0	SIM
-----	-----	-----	-----	---	-----	-----	-----

Table 31. CTRL_REG4 description

BDU	Block Data Update. Default value: 0 (0: continuous update; 1: output registers not updated until MSB and LSB reading)
BLE	Big/Little Endian Data Selection. Default value 0. (0: Data LSB @ lower address; 1: Data MSB @ lower address)
FS1-FS0	Full Scale selection. Default value: 00 (00: 250 dps; 01: 500 dps; 10: 2000 dps; 11: 2000 dps)
ST1-ST0	Self Test Enable. Default value: 00 (00: Self Test Disabled; Other: See Table)
SIM	SPI Serial Interface Mode selection. Default value: 0 (0: 4-wire interface; 1: 3-wire interface).



2. Gyroscope (L3G4200D)

```
i2c_bus.write_byte(i2c_address,0x28)
X_L = i2c_bus.read_byte(i2c_address)
i2c_bus.write_byte(i2c_address,0x29)
X_H = i2c_bus.read_byte(i2c_address)
X = X_H << 8 | X_L
```

.

.

.

OUT_X_L (28h), OUT_X_H (29h)

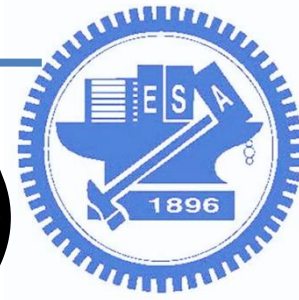
X-axis angular rate data. The value is expressed as two's complement.

OUT_Y_L (2Ah), OUT_Y_H (2Bh)

Y-axis angular rate data. The value is expressed as two's complement.

OUT_Z_L (2Ch), OUT_Z_H (2Dh)

Z-axis angular rate data. The value is expressed as two's complement.



2. Gyroscope (L3G4200D)

- The output is bit value, how to convert it to angular velocity?

```
pi@raspberrypi:~/L3G4200D-Python$ sudo python gyro.py
-1013      -256      1602
  946      -683      -195
  943      -681      -198
  947      -683      -193
  940      -679      -194
  947      -672      -192
  935      -680      -193
  948      -514      -192
```

- Change output to DPS (degrees Per Second)

- **DPS = (Raw * 70) / 1000**

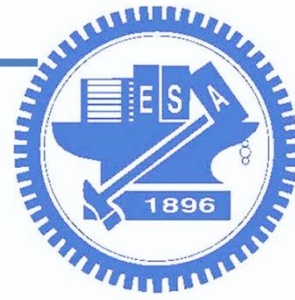
Symbol	Parameter	Test condition	Min.	Typ. ⁽²⁾	Max.	Unit
So	Sensitivity	FS = 250 dps		8.75		mdps/digit
		FS = 500 dps		17.50		
		FS = 2000 dps		70		

The mdps/digit stands for Milli Degrees Per Second



Quiz 2

- Rotate the sensor to control the LED
 - **Left: turn on LED; Right: turn off LED**



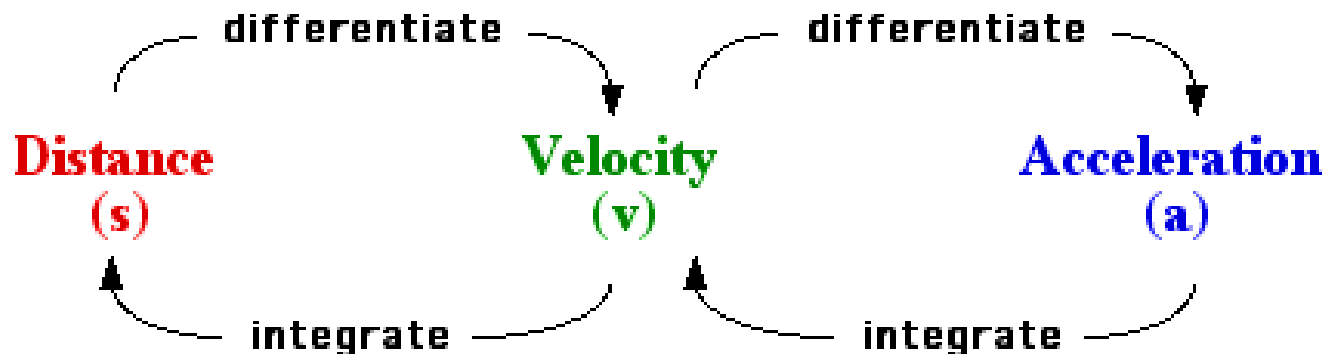
Discussion

- We can obtain $a_x/a_y/a_z$ from sensor.
- How to calculate the distance based on accelerometer?
- **Is it possible? Why?**

$$\mathbf{s}(t) = \mathbf{s}_0 + \mathbf{v}_0 t + \frac{1}{2} \mathbf{a} t^2 = \mathbf{s}_0 + \frac{\mathbf{v}_0 + \mathbf{v}(t)}{2} t$$

$$\mathbf{v}(t) = \mathbf{v}_0 + \mathbf{a} t$$

$$v^2(t) = v_0^2 + 2\mathbf{a} \cdot [\mathbf{s}(t) - \mathbf{s}_0]$$





Summary

- Practice Lab (accelerometer, gyroscope)
- Write down the answer for discussion
 - Discussion: How to calculate the distance based on accelerometer?
- Write code for **Quiz 1 - 2**, then **demonstrate it to TAs**
 - Quiz1: Hold the sensor, do not move for 5 seconds.
 - Quiz2: Rotate the sensor to control the LED