

Asier Marin

Aritz Ibañez

Ane Sanchiz

## Laboratorio 5

### Sensores para Sistemas Embebidos

#### Instrucciones

En este divertido laboratorio vais a aprender cómo comunicarse con sensores que usan el protocolo I2C. Además, aprenderás a buscar información en el *datasheet* del sensor. En estas actividades mostraremos los valores del sensor los mostraremos por pantalla, pero en el último tema del curso aprenderás cómo guardar los valores en una base de datos, o acceder a su lectura desde un navegador web desde tu smartphone.

- Entrega: en ALUD
- Este laboratorio se hará en grupos de 2-3 personas, aunque la entrega será individual.
- Revisad las diapositivas correspondientes a los sensores I2C.
- Antes de comenzar debéis instalar la librería smbus2 para I2C en python en la RPi  
`$ sudo pip3 install smbus2`

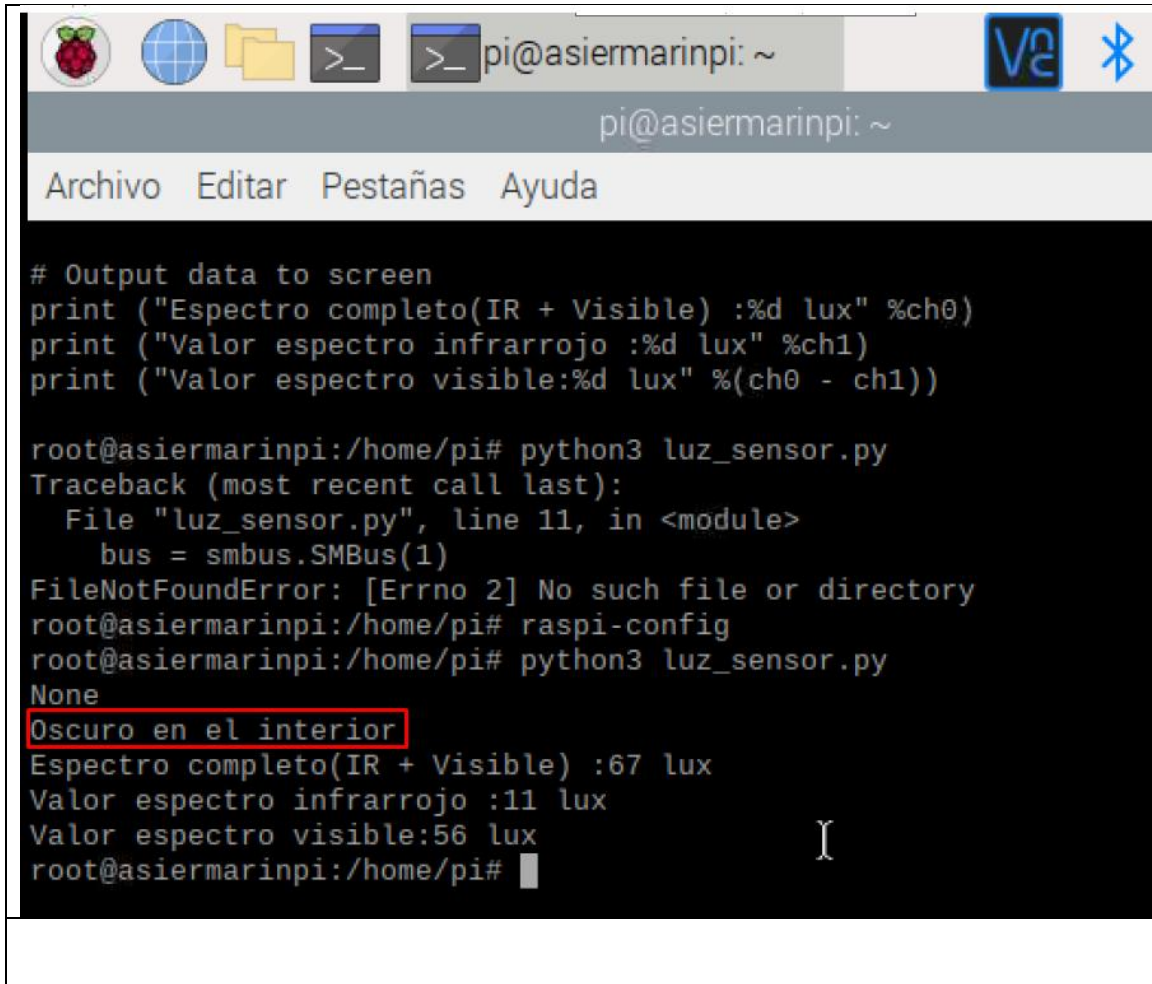
#### Parte 1 (4 puntos): Sensor digital de Luz

**Entrega:** Este mismo documento habiendo respondido a las preguntas, y el fichero `luz_sensor.py` completo.

En esta parte usaréis el sensor digital de Luz TSL2561. El *datasheet* está disponible en ALUD. Se proporciona el programa de ejemplo (`luz_sensor.py`) Debéis conectar el sensor como se muestra en la figura de la siguiente página, y ejecutar el programa de ejemplo anterior. Comprobad que se muestran por pantalla los valores del sensor. A continuación, realizar las siguientes tareas/preguntas

- A. (1 punto) En el programa `luz_sensor.py`, añadir una función que muestre por pantalla los valores de intensidad luminosa de una forma más *legible* o *intuitiva*, en vez de valores LUX. Por ejemplo, cuando el valor de LUX sea 125, se mostraría “Habitación interior bastante oscura”. Usad aquí vuestra creatividad para diseñar la función, y tomad como referencia los valores de la tabla de esta página web <https://docs.microsoft.com/en-us/windows/win32/sensorsapi/understanding-and-interpreting-lux-values>

Como se puede observar en la imagen siguiente se puede ver como se muestra en pantalla la iluminación obtenida. Estas descripciones se basan en el rango de valores obtenidos a partir de la documentación oficial.



```
# Output data to screen
print ("Espectro completo(IR + Visible) :%d lux" %ch0)
print ("Valor espectro infrarrojo :%d lux" %ch1)
print ("Valor espectro visible:%d lux" %(ch0 - ch1))

root@asiermarinpi:/home/pi# python3 luz_sensor.py
Traceback (most recent call last):
  File "luz_sensor.py", line 11, in <module>
    bus = smbus.SMBus(1)
FileNotFoundError: [Errno 2] No such file or directory
root@asiermarinpi:/home/pi# raspi-config
root@asiermarinpi:/home/pi# python3 luz_sensor.py
None
Oscuro en el interior
Espectro completo(IR + Visible) :67 lux
Valor espectro infrarrojo :11 lux
Valor espectro visible:56 lux
root@asiermarinpi:/home/pi#
```

```
if (ch0 < 10):
    print("Negro total")
elif (ch0 >= 11 and ch0 <= 50):
    print(" Muy negro")
elif (ch0 >= 51 and ch0 <= 200):
    print("Oscuro en el interior")
elif (ch0 >= 201 and ch0 <= 400):
    print("Tenue en el interior")
elif (ch0 >= 401 and ch0 <= 1000):
    print("Normal")
elif (ch0 >= 1001 and ch0 <= 5000):
    print("Luminoso")
elif (ch0 >= 5001 and ch0 <= 10000):
    print("Luminoso en el exterior")
elif (ch0 >= 10001 and ch0 <= 30000):
    print("Muy luminoso en el exterior")
elif (ch0 >= 30001 and ch0 <= 100000):
    print("Luz directa")
```

- B. (1 punto) Descubre la dirección I2C del sensor con el comando `$i2cdetect -y 1`  
Escríbela debajo. ¿Coincide con la que se indica en el datasheet?

El comando describe desde qué dirección hexadecimal se obtienen los datos del sensor.  
Al igual que se indica en la documentación oficial, la dirección debe apuntar a 0x29 o 0x49.

```
root@Aritz:/home/pi# i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  29  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
root@Aritz:/home/pi# python3 luz_sensor.py
Espectro completo(IR + Visible):109 lux
valor espectro infrarrojo :49 lux
valor espectro visible:60 lux
```

```
// Read ADC Channels Using Read Word Protocol - RECOMMENDED
Address = 0x39 //Slave addr - also 0x29 or 0x49

//Address the Ch0 lower data register and configure for Read Word
Command = 0xAC //Set Command bit and Word bit

//Reads two bytes from sequential registers 0x0C and 0x0D
//Results are returned in DataLow and DataHigh variables
ReadWord (Address, Command, DataLow, DataHigh)
Channel0 = 256 * DataHigh + DataLow

//Address the Ch1 lower data register and configure for Read Word
Command = 0xAE //Set bit fields 7 and 5

//Reads two bytes from sequential registers 0x0E and 0x0F
//Results are returned in DataLow and DataHigh variables
ReadWord (Address, Command, DataLow, DataHigh)
Channel1 = 256 * DataHigh + DataLow //Shift DataHigh to upper byte

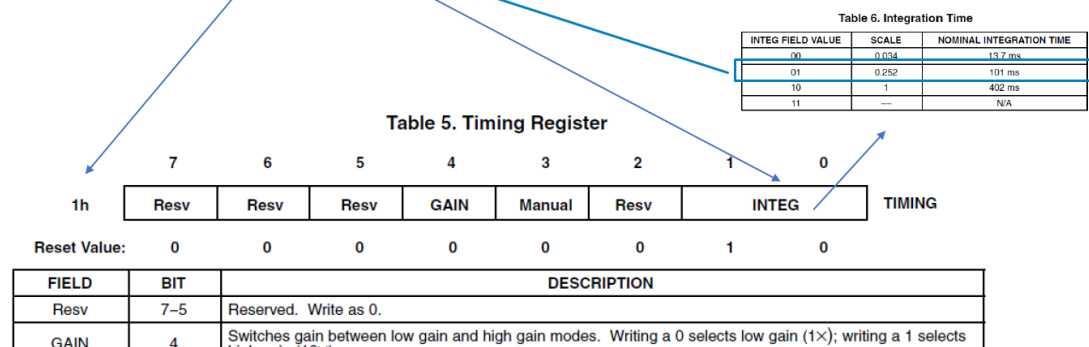
// Read ADC Channels Using Read Byte Protocol
Address = 0x39 //Slave addr - also 0x29 or 0x49
Command = 0x8C //Address the Ch0 lower data register
ReadByte (Address, Command, DataLow) //Result returned in DataLow
Command = 0x8D //Address the Ch0 upper data register
ReadByte (Address, Command, DataHigh) //Result returned in DataHigh
Channel0 = 256 * DataHigh + DataLow //Shift DataHigh to upper byte

Command = 0x8E //Address the Ch1 lower data register
ReadByte (Address, Command, DataLow) //Result returned in DataLow
Command = 0x8F //Address the Ch1 upper data register
ReadByte (Address, Command, DataHigh) //Result returned in DataHigh
Channel1 = 256 * DataHigh + DataLow //Shift DataHigh to upper byte
```

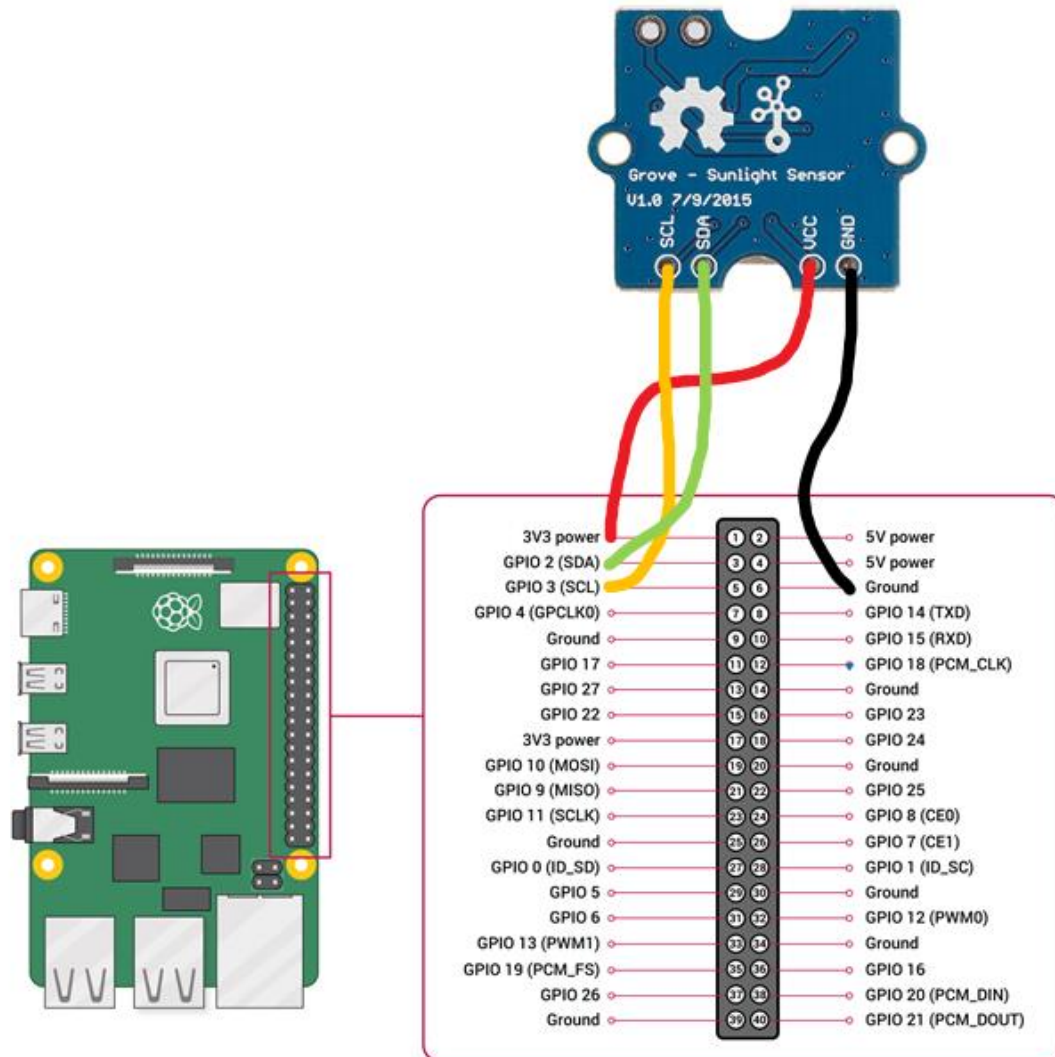
- C. (2 puntos) ¿Cómo cambiarías el valor del tiempo de integración del conversor analógico-digital a un valor de 101 ms? (pista: mirad el registro de timing ó Timing Register). Consultad además las diapositivas de clase. Incluye la línea o líneas de código de Python que usarías.

Para obtener el valor con ese tiempo de respuesta es necesario que se establezca en el valor 0x01.

`bus.write_byte_data(0x29, 0x01 | 0x80, 0x02)`



```
seleccionar el timing register 0x01(01) con el registro de comando  
Escribir el valor 0x02(02)  
bus.write_byte_data(TSL2561_DEFAULT_ADDRESS, 0x01 | 0x80, 0x01)  
nt(x)
```

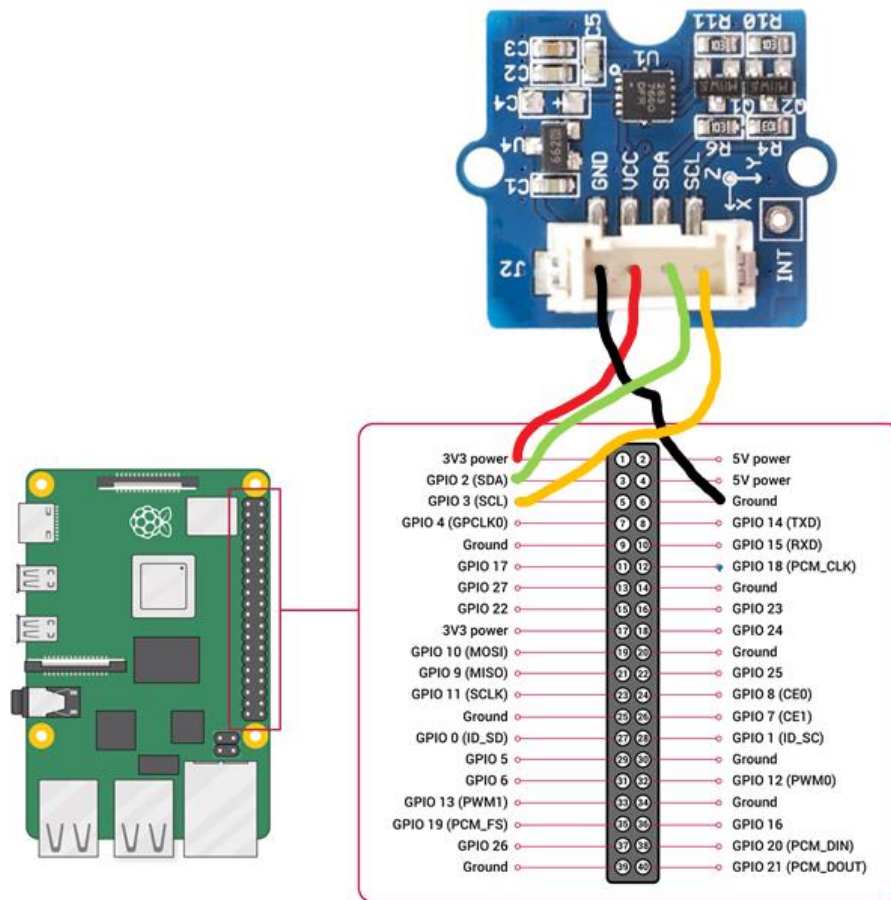


## Parte 2 (6 puntos): Acelerómetro

El acelerómetro que tenéis es del tipo MEMs capacitivo, y una vez obtenido los valores analógicos de aceleración de los ejes x,y,z, los convierte a formato digital de 6 bits, y lo guarda en unos registros internos. Mediante el protocolo I2C y nuestra RPi, es posible leer los valores de dichos registros internos para acceder a los valores de la aceleración en x,y,z.

**Entrega:** Este mismo documento habiendo respondido a las preguntas, y el fichero [accel\\_ejemplo.py](#) completo.

La siguiente figura muestra cómo debemos conectar el sensor a la RPI.



- A. (0.5 puntos) Conecta el sensor a la RPi, y comprueba el funcionamiento con el programa que se proporciona [accel\\_ejemplo.py](#). Se debe mostrar por pantalla los valores de x,y,z. ¿A qué **unidades** corresponde la medida absoluta que devuelve el sensor?

Corresponde a la intensidad gravitacional universal. De forma constante cada unidad corresponde a 9.8 m/s<sup>2</sup>.

$$1 \text{ g} = 9,80665 \text{ m/s}^2$$



```

*****
Acceleration in X-Axis : -0.375059 g
Acceleration in Y-Axis : -0.046882 g
Acceleration in Z-Axis : 0.890764 g
*****

```

B. (0.5 puntos) ¿Por qué se dividen los valores que se leen del registro de datos entre **21.33**?

El valor predeterminado de entrada son 21.33, la idea es dividir por ese valor para obtener una referencia sobre 0. De esa manera es más fácil a la vista percibir los cambios en los diversos ejes.

0g Offset Temperature Variation					
X			-1.3		mg/°C
Y			+1.5		mg/°C
Z			-1.0		mg/°C
Sensitivity <sup>(1)</sup> (T <sub>A</sub> = 25°C, AV <sub>DD</sub> = 2.8 V)					
±1.5g range 6-bit			21.33		count/g
Acceleration Sensitivity at T <sub>AMB</sub>		19.62	21.33	23.04	count/g
Acceleration Sensitivity Temperature Variation			±0.01		%/°C
Input High Voltage	V <sub>IH</sub>	0.7 x V <sub>DD</sub>			
Input Low Voltage	V <sub>IL</sub>			0.35 x V <sub>DD</sub>	
Output Low Voltage (IOL = 6 mA + SDA, INT)	V <sub>OL</sub>			0.5	V
Input Leakage Current	I <sub>IH</sub> , I <sub>IL</sub>		0.025		µA



C. (0.5 puntos) ¿A qué se refiere el valor  $\pm 1.5g$ , que describe el sensor en el datasheet?

Active Mode, ODR = 16	$I_{DD}$		89		$\mu A$
Active Mode, ODR = 32	$I_{DD}$		133		$\mu A$
Active Mode, ODR = 64	$I_{DD}$		221		$\mu A$
Active Mode <sup>(1)</sup> , ODR = 120	$I_{DD}$		294		$\mu A$
Acceleration Range			$\pm 1.5$		g
Operating Temperature Range	$T_A$	-40	25	85	$^{\circ}C$
0g Output Signal ( $T_A = 25^{\circ}C$ , $AV_{DD} = 2.8 V$ )					
0g Offset $\pm 1.5g$ range <sup>(1)</sup>	$V_{OFF}$	-3	0	3	counts

Es el rango de error del sensor.

D. (0.5 puntos) Modifica ligeramente el programa anterior, de forma que se muestren por pantalla los valores en la unidad  $m/s^2$ .

Simplemente se ha multiplicado por su correcta conversión como constante.

```
while True :
    time.sleep(0.1)
    accl = mma7660fc.read_accl()
    print ("Acceleration in X-Axis : %f"%((accl['x']/21.33) * 9.8) + " m/s2")
    print ("Acceleration in Y-Axis : %f"%((accl['y']/21.33) * 9.8)+ " m/s2")
    print ("Acceleration in Z-Axis : %f"%((accl['z']/21.33) * 9.8)+ " m/s2")
    print (" ***** ")
```

```
*****
Acceleration in X-Axis : 1.837787 m/s2
Acceleration in Y-Axis : -4.594468 m/s2
Acceleration in Z-Axis : 9.648383 m/s2
*****
```

- E. (1.5 puntos) Añade una función en el programa anterior ([accel\\_ejemplo.py](#)) que detecte el evento de un “shake”. El sensor que tenéis ya implementa la detección del evento de shake, así que se puede leer el estado de shake o no-shake mediante la lectura del bit 7 del registro 03. Como ayuda, bázate en la función que se incluye en el archivo de Python que lee los bits D4,D3,D2 para detectar la polarización (PoLa) del acelerómetro.

**\$03: Tilt Status (Read only)**

**TILT**

D7	D6	D5	D4	D3	D2	D1	D0
Shake	Alert	Tap	PoLa[2]	PoLa[1]	PoLa[0]	BaFro[1]	BaFro[0]
0	0	0	0	0	0	0	0

**BaFro[1:0]**

00: Unknown condition of front or back  
01: Front: Equipment is lying on its front  
10: Back: Equipment is lying on its back

**PoLa[2:0]**

000: Unknown condition of up or down or left or right  
001: Left: Equipment is in landscape mode to the left  
010: Right: Equipment is in landscape mode to the right  
101: Down: Equipment standing vertically in inverted orientation  
110: Up: Equipment standing vertically in normal orientation

**Tap**

1: Equipment has detected a tap  
0: Equipment has not detected a tap

**Alert**

0: Register data is valid  
1: The register was read at the same time as MMA7660FC was attempting to update the contents. Re-read the register

**Shake**

0: Equipment is not experiencing shake in one or more of the axes enabled by SHINTX, SHINTY, and SHINTZ  
1: Equipment is experiencing shake in one or more of the axes enabled by SHINTX, SHINTY, and SHINTZ

**Note:** When entering active mode from standby mode, if the device is flat ( $\pm 1g$  on Z-axis) the value for BaFro will be back (-1g) or front (+1g) but PoLa will be in unknown condition. if the device is being held in an Up/Down/Right/Left position, the PoLa value will be updated with current orientation, but BaFro will be in unknown condition.

- F. (1 punto) ¿Qué criterio usa el sensor para decidir si ha habido un evento de shake o no? (pista: consultad el datasheet)

```
# Actividad Parte 2 - E
def read_shake(self):
    data = bus.read_i2c_block_data(MMA7660FC_DEFAULT_ADDRESS,MMA7660FC_TILT,1)

    shake = data[0] & 0x80

    if shake == 0x00:
        print("No shake")
    else:
        print("Shake")
```

```
Orientacion derecha
No shake ←
Valores en g:
Acceleration in X-Axis : -0.375059g
Acceleration in Y-Axis : -0.890764g
Acceleration in Z-Axis : -0.140647g
*****
Pasos: 6
```

- G. (1.5 puntos) Realiza una función en el fichero Python anterior que implemente **contador de pasos simple basado en los valores leídos del acelerómetro**. No se pide que sea robusto, sino un programa sencillo. Cuando se llame a la función, el programa debe empezar a detectar pasos. Cada paso detectado, imprime por pantalla el número actual del contador de pasos (desde el comienzo del contador de pasos). Para comprobar el funcionamiento, puedes mover suavemente el acelerómetro con la mano, emulando el movimiento de un paso (andando). Como ayuda, consulta las diapositivas de clase.

La referencia utilizada para contar pasos es 30. Se ha escogido a partir de ir probando poco a poco. La idea es utilizar la magnitud como filtro a la hora de contar pasos.

```
# Actividad Parte 2 - G
def contar_pasos(self, eje_x, eje_y, eje_z):
    mag = math.sqrt((eje_x**2) + (eje_y**2) + (eje_z**2))
    pasos = 0
    if (mag > 30):
        return True
    else:
        return False
```

Este método por lo tanto se encuentra en el *loop* que hace la llamada para el filtrado constantemente.

```
Orientacion abajo
No shake
Valores en g:
Acceleration in X-Axis : -0.562588g
Acceleration in Y-Axis : 0.234412g
Acceleration in Z-Axis : 0.843882g
*****
Pasos: 4
```

```
pasos = 0
while True :
    time.sleep(0.1)
    mma7660fc.read_orientation()
    mma7660fc.read_shake()
    accl = mma7660fc.read_accl()
    print("Valores en g:")
    print ("Acceleration in X-Axis : %f"%(accl['x']/21.33) + "g")
    print ("Acceleration in Y-Axis : %f"%(accl['y']/21.33)+ "g")
    print ("Acceleration in Z-Axis : %f"%(accl['z']/21.33)+ "g")
    print (" ***** ")
    eje_x = accl['x']
    eje_y = accl['y']
    eje_z = accl['z']
    if (mma7660fc.contar_pasos(eje_x, eje_y, eje_z) == True):
        pasos += 1
    print(f"Pasos: {pasos}")
```