

Acelerómetro

Página extraída de Federico Coca [Guía de Trabajo de Microbit](#) CC-BY-SA

Este objeto permite acceder al acelerómetro de la placa.

Por defecto MicroPython establece el rango del acelerómetro en ± 2000 mg (siendo g una unidad de aceleración basada en la gravedad estándar), que configura los valores máximo y mínimo devueltos por las funciones del acelerómetro. El rango puede cambiarse mediante `microbit.accelerometer.set_range()`.

El acelerómetro también proporciona funciones de conveniencia para detectar gestos. Los gestos reconocidos se representan como cadenas: arriba (up), abajo (down), izquierda (left), derecha (right), boca arriba (face up), boca abajo (face down), caída libre (freefall), 3g, 6g, 8g, sacudida (shake).

Nota: Los gestos no se actualizan en segundo plano por lo que es necesario realizar llamadas constantes a algún método del acelerómetro para realizar la detección de gestos. Normalmente los gestos pueden ser detectados usando un bucle con un pequeño retardo `microbit.sleep()`.

Sus funciones son:

- `microbit.accelerometer.get_x()`
- `microbit.accelerometer.get_y()`
- `microbit.accelerometer.get_z()`

Retorna como un entero positivo o negativo la aceleración medida en el eje correspondiente en mili-g.

- `microbit.accelerometer.get_values()`

Devuelve las medidas de aceleración en todos los ejes a la vez, como una tupla de tres elementos de enteros ordenados como X, Y, Z.

- `microbit.accelerometer.get_strength()`

Obtiene la medida de la aceleración de todos los ejes combinados, como un entero positivo. Es la suma pitagórica de los ejes X, Y y Z. Devuelve la fuerza de aceleración combinada de todos los ejes, en mili-g.

- `microbit.accelerometer.current_gesture()`

Devuelve una cadena con el nombre del gesto actual.

- `microbit.accelerometer.is_gesture(name)`

El parámetro `name` es una cadena con el nombre del gesto a comprobar. Devuelve un valor booleano que indica si el gesto nombrado está activo actualmente.

- `microbit.accelerometer.was_gesture(name)`

El parámetro `name` es una cadena con el nombre del gesto a comprobar. Devuelve un valor booleano que indica si el gesto nombrado ha estado activo desde la última vez.

- `microbit.accelerometer.get_gestures()`

Se usa para obtener una lista histórica de los gestos registrados. Al llamar a esta función se borra el histórico de gestos antes de devolver el valor. Devuelve una tupla del historial de gestos, el más reciente aparece en último lugar.

- `microbit.accelerometer.set_range(value)`

Ajusta el rango de sensibilidad del acelerómetro, en g (gravedad estándar), a los valores más cercanos soportados por el hardware, de forma que redondee a 2, 4 u 8 g. El parámetro `value` establece el nuevo rango para el acelerómetro, un entero en g.

A continuación vamos a ver los ejemplos que aparecen en la documentación oficial traducidos.

Trazador gráfico de aceleraciones

El programa lo vamos a hacer en el editor Mu para aprovechar su trazador gráfico.

```
from microbit import *

while True:
    uart.write(str(accelerometer.get_values()) + "\r\n")
    sleep(1000)
```

Ponemos en marcha el trazador, y la comunicación serie REPL después de flashear el programa. Hacemos reset de la micro:bit y podemos ver la evolución de las lecturas con los movimientos de la placa.

A11_brujula_MC.png

Gradiometro

En este programa se ve como si la luz pesara:

<https://www.youtube.com/embed/JR8h-O11QbY>

El programa es:

```
from microbit import *

# El brillo estara entre 0 y 9

brillo = 9
```

```
# La funcion mapear ajusta los valores
# leidos y los lleva al rango 0-4.
def mapear(valor):
    if valor < -500:
        valor=-500
    elif valor > 500:
        valor=500
    valor=(valor+500)/250
    return int(valor)

while True:
    # Lee la aceleración en x e y con un
    #rango que va de -2000 a 2000.
    roll_x = accelerometer.get_x()
    pitch_y = accelerometer.get_y()
    # No necesitamos un rango tan amplio por
    # eso lo bajamos de -500 a 500
    x=mapear(roll_x)
    y=mapear(pitch_y)
    display.clear()
    display.set_pixel(x, y, brillo)
    sleep(500)
```

Bola mágica

Una bola 8 mágica que adivina el futuro. Haz una pregunta y agita el dispositivo para obtener una respuesta.

```
# Magic 8 ball by Nicholas Tollervey. February 2016.
# Bola mágica 8 por Nicholas Tollervey. Febrero 2016.
# Ask a question then shake.
# Haz una pregunta y agita la micro:bit
# This program has been placed into the public domain.
# Este programa es de dominio público.

from microbit import *
import random
```

```
respuestas = [  
    "Es cierto",  
    "Es decididamente así",  
    "Sin duda alguna",  
    "Sí, definitivamente.",  
    "Puedes confiar en ello",  
    "Como yo lo veo, sí",  
    "Lo más probable",  
    "Buenas perspectivas",  
    "Si",  
    "Los indicios apuntan a que sí",  
    "Respuesta dudosa inténtalo de nuevo",  
    "Vuelve a preguntar más tarde",  
    "Mejor no te lo digo ahora",  
    "No se puede predecir ahora",  
    "Concéntrate y vuelve a preguntar",  
    "No cuentes con ello",  
    "Mi respuesta es no",  
    "Mis fuentes dicen que no",  
    "Perspectivas no tan buenas",  
    "Muy dudoso",  
]  
  
while True:  
    display.show('8')  
    if accelerometer.was_gesture('shake'):  
        display.clear()  
        sleep(1000)  
        display.scroll(random.choice(respuestas))  
    sleep(10)
```

Juego evita obstáculos

Un juego consistente en evitar obstáculos moviendo la micro:bit.

```
# Simple Slalom by Larry Hastings, September 2015  
# Eslalon simple de Larry Hastings, septiembre de 2015.  
# This program has been placed into the public domain.  
# Este programa es de dominio público.
```

```

import microbit as m
import random

p = m.display.show
min_x = -1024
max_x = 1024
range_x = max_x - min_x
wall_min_speed = 400
player_min_speed = 200
wall_max_speed = 100
player_max_speed = 50
speed_max = 12

while True:

    i = m.Image('00000:'*5)
    s = i.set_pixel
    player_x = 2
    wall_y = -1
    hole = 0
    score = 0
    handled_this_wall = False
    wall_speed = wall_min_speed
    player_speed = player_min_speed
    wall_next = 0
    player_next = 0

    while True:
        t = m.running_time()
        player_update = t >= player_next
        wall_update = t >= wall_next
        if not (player_update or wall_update):
            next_event = min(wall_next, player_next)
            delta = next_event - t
            m.sleep(delta)
            continue

        if wall_update:
            # calculate new speeds
            speed = min(score, speed_max)

```

```

wall_speed = wall_min_speed + int((wall_max_speed - wall_min_speed) * speed / speed_max)
player_speed = player_min_speed + int((player_max_speed - player_min_speed) * speed / speed_max)
wall_next = t + wall_speed
if wall_y < 5:
    # erase old wall
    use_wall_y = max(wall_y, 0)
    for wall_x in range(5):
        if wall_x != hole:
            s(wall_x, use_wall_y, 0)
wall_reached_player = (wall_y == 4)
if player_update:
    player_next = t + player_speed
    # find new x coord
    x = m.accelerometer.get_x()
    x = min(max(min_x, x), max_x)
    # print("x accel", x)
    s(player_x, 4, 0) # turn off old pixel
    x = ((x - min_x) / range_x) * 5
    x = min(max(0, x), 4)
    x = int(x + 0.5)
    # print("have", position, "want", x)
    if not handled_this_wall:
        if player_x < x:
            player_x += 1
        elif player_x > x:
            player_x -= 1
    # print("new", position)
    # print()
if wall_update:
    # update wall position
    wall_y += 1
    if wall_y == 7:
        wall_y = -1
        hole = random.randrange(5)
        handled_this_wall = False
if wall_y < 5:
    # draw new wall
    use_wall_y = max(wall_y, 0)
    for wall_x in range(5):
        if wall_x != hole:

```

```
s(wall_x, use_wall_y, 6)
if wall_reached_player and not handled_this_wall:
    handled_this_wall = True
    if (player_x != hole):
        # collision! game over!
        break
    score += 1
if player_update:
    s(player_x, 4, 9) # turn on new pixel
p(i)
p(i.SAD)
m.sleep(1000)
m.display.scroll("Score:" + str(score))
while True:
    if (m.button_a.is_pressed() and m.button_a.is_pressed()):
        break
    m.sleep(100)
```

<https://www.youtube.com/embed/a0f75gyTm5U>

Página extraída de Federico Coca [Guia de Trabajo de Microbit](#) CC-BY-SA

Revision #5

Created 27 September 2024 11:53:53 by Javier Quintana

Updated 1 October 2024 19:42:14 by Javier Quintana