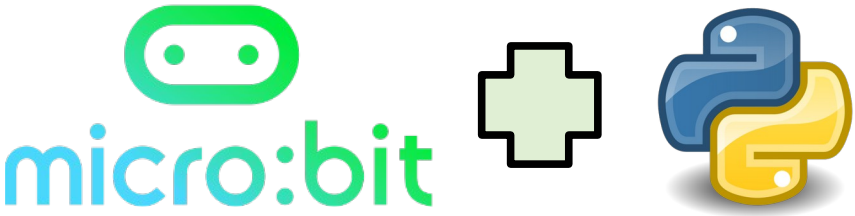
Retos de Micro:bit + Python



Por: *Pedro Ruiz Fernández*

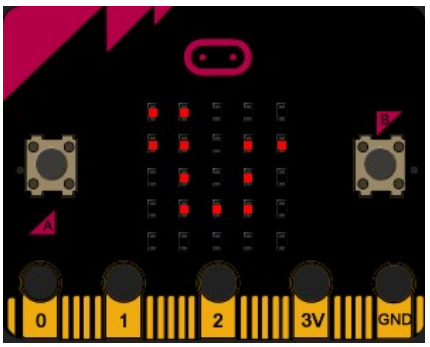
Versión 07/11/2021

Licencia

Reto 1. Hola Mundo
Es la primera práctica o reto en programación, en nuestro caso mostramos con ayuda de la matriz de leds la frase "hola mundo" en modo texto con desplazamiento (scroll). Al objeto display se le aplica un método o procedimiento (acción) llamada scroll.

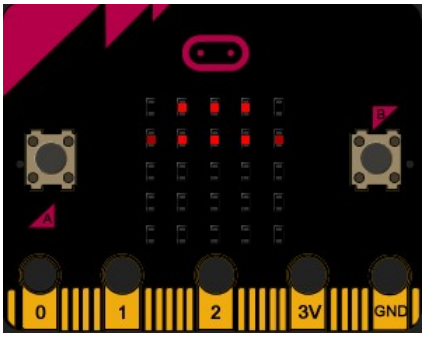
```
from microbit import *
display.scroll("Hola Mundo")
```

Reto 2. Iconos en display
En este reto al objeto display se le aplica el procedimiento (acción) show que nos permite mostrar imágenes prediseñadas, en esta dirección podéis encontrar otras imágenes prediseñadas.




```
from microbit import *
display.show(Image.SNAKE)
```

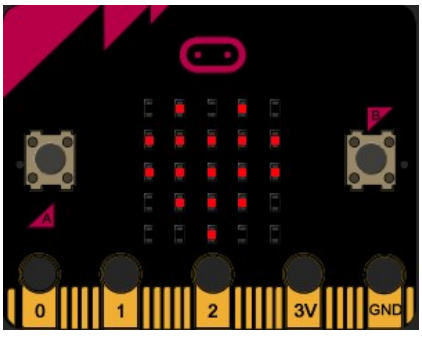
Reto 3. Imágenes DIY
También podemos hacer imágenes en la matriz de leds DIY (Di It Yourself), para ello creamos una variable a la que asociamos una imagen compuesta a base de las diferentes filas (5) compuestas por 5 leds cada una indicando en la posición de cada led su luminosidad de 0 (apagado) a 9 intensidad luminosa máxima. En el ejemplo dibujamos un sombrero.




```
from microbit import *

hat = Image("09990:"
            "59995:"
            "00000:"
            "00000:"
            "00000")

display.show(hat)
```

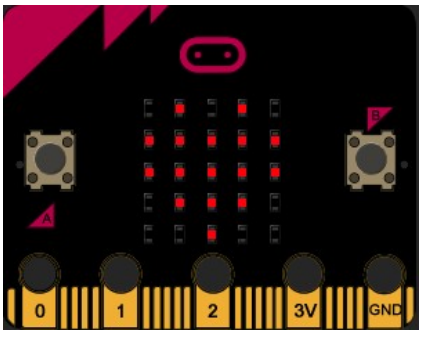
Reto 4. Animaciones
En este reto creamos un array (matriz) de imágenes, en nuestro caso de corazones de tamaño diferente, para animarlos simulando el pálpito del mismo.

```
from microbit import *

while True:
        display.show (Image.HEART)
        sleep(500)
        display.show(Image.HEART_SMALL)
        sleep(500)
```



Otra solución :

```
from microbit import *
imagenes=[Image.HEART,Image.HEART_SMALL]

while True:
        display.show (imagenes[0])
        sleep(500)
        display.show(imagenes[1])
        sleep(500)
```



Otra solución:

```
from microbit import *
imagenes=[Image.HEART,Image.HEART_SMALL]

while True:
        display.show (imagenes,delay=500)
```

Reto 5. Animaciones DIY

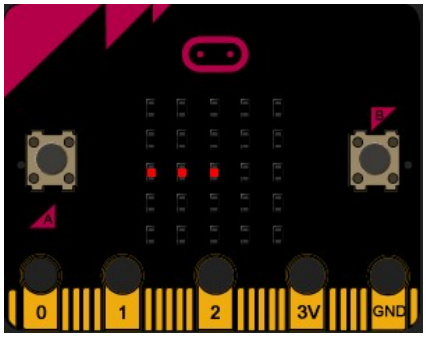
Este reto trata sobre la animación de un sombrero cayendo, para ello lo que hacemos es dibujar 6 sombreros en 6 posiciones diferentes desapareciendo hacia abajo, por tanto lo último que se verá será la parte alta del sombrero. Para ello además de crear las 6 imágenes, creamos una lista con todos los sombreros, en nuestro caso “hats”, para hacer efectiva la animación a través de la orden display.show (hats, delay=150), lo que hacemos es ejecutar la animación con un tiempo de retardo entre imagen de 150 ms.

```
from microbit import *

hat1 = Image("09990:"
        "59995:"
        "00000:"
        "00000:"
        "00000")

hat2 = Image("00000:"
        "09990:"
        "59995:"
        "00000:"
        "00000")

hat3 = Image("00000:"
        "00000:"
        "09990:"
        "59995:"
        "00000")

hat4 = Image("00000:"
        "00000:"
        "00000:"
        "09990:"
        "59995")

hat5 = Image("00000:"
        "00000:"
        "00000:"
        "00000:"
        "09990")

hat6 = Image("00000:"
        "00000:"
        "00000:"
        "00000:"
        "00000")

hats = [hat1, hat2, hat3, hat4, hat5, hat6]
display.show(hats, delay=150)
```

Reto 6. Animación para siempre
En este caso se trata de realizar la animación anterior pero introduciendo la misma en un bucle infinito realizado con “while True”.

```
from microbit import *

hat1 = Image("09990:"
             "59995:"
             "00000:"
             "00000:"
             "00000")

hat2 = Image("00000:"
             "09990:"
             "59995:"
             "00000:"
             "00000")

hat3 = Image("00000:"
             "00000:"
             "09990:"
             "59995:"
             "00000")

hat4 = Image("00000:"
             "00000:"
             "00000:"
             "09990:"
             "59995")

hat5 = Image("00000:"
             "00000:"
             "00000:"
             "00000:"
             "09990")

hat6 = Image("00000:"
             "00000:"
             "00000:"
             "00000:"
             "00000")

hats = [hat1, hat2, hat3, hat4, hat5, hat6]
while True:
    display.show(hats, delay=150)
```

Reto 7. Contador de pulsaciones de botones
En este reto vamos a contar el número de veces que pulsamos el botón a de microbit en un tiempo de 12 segundos, para ello. Usamos la función sleep que deja en modo pausa a microbit durante el número de milisegundos indicado. Además para contar el número de veces que es pulsado el objeto botón a usamos la función buton_a.get_presses().




```
from microbit import *

sleep(12000)
```

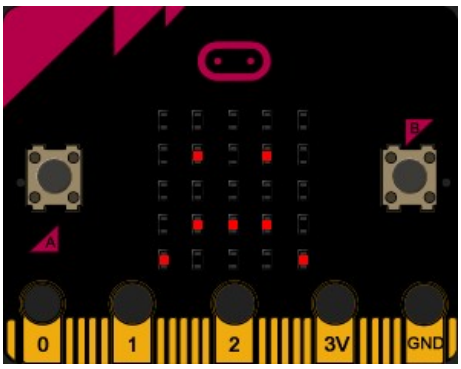
```
display.scroll(str(button_a.get_presses()))
```

Reto 8. While
Este reto muestra la imagen de Reloj a las 12 mientras el tiempo de funcionamiento de microbit sea inferior a 9 segundos, pasado ese tiempo muestra la imagen Reloj a las 9.

```
from microbit import *

while running_time() < 9000:
        display.show(Image.CLOCK12)

display.show(Image.CLOCK9)
```

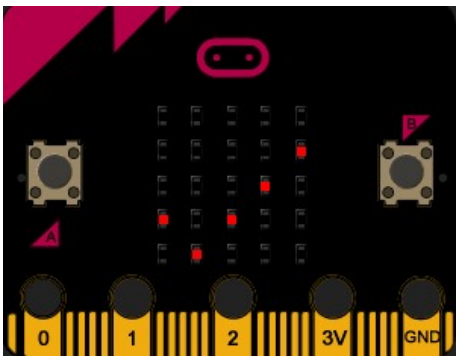
Reto 9. Bucle infinito e if
En este reto si pulsas el botón “A” muestra la imagen SMILE si lo sueltas se muestra la imagen SAD, sale del bucle infinito (while True) cuando pulsas el botón “B”.

```
from microbit import *

while True:
        if button_a.is_pressed():
                display.show(Image.SMILE)
        elif button_b.is_pressed():
                break
        else:
                display.show(Image.SAD)
display.clear()
```



Reto 10. Bucle infinito e if con control de variable booleana
Se trata de mostrar una imagen u otra (YES, NO) pulsando un sólo botón, en este caso el botón “B”.

```
from microbit import *
on= True

while True:
        if button_b.is_pressed():
                on = not on
                sleep (200)
        if on==True:
                display.show(Image.YES)
        else:
                display.show(Image.NO)
```



Reto 11. Bucle for (repeticiones)
Se trata de realizar la animación del corazón palpitando, alternando las imágenes de corazón normal y pequeño 5 veces usando for.

```
from microbit import *

for i in range (5):
        display.show(Image.HEART)
        sleep (500)
        display.show(Image.HEART_SMALL)
        sleep (500)
display.clear()
```

Reto 12. Bucle for (repeticiones) con matrices de leds, enciende primera fila
Reto consistente en encender la primera fila de leds, secuencialmente de izquierda a derecha cada medio segundo.

```
from microbit import *

for x in range (5):
        display.set_pixel(x,0,9)
        sleep (500)
        display.set_pixel (x,0,0)
        sleep (500)

display.clear()
```

Reto 13. Bucle for (repeticiones) con matrices de leds, enciende todos los leds
Se trata de encender y apagar todos los leds del display secuencialmente de izquierda a derecha y de arriba a abajo cada 50 milisegundos.

```
from microbit import *

for y in range (5):
        for x in range (5):
                display.set_pixel(x,y,9)
                sleep (50)
                display.set_pixel (x,y,0)
                sleep (50)

display.clear()
```

Igual que el anterior pero además hace el recorrido contrario (de abajo a arriba y de derecha a izquierda) y todo indefinidamente.

```
from microbit import *
tiempo=50
while True:
  for y in range (0,5,1):
    for x in range (0,5,1):
      display.set_pixel(x,y,9)
      sleep (tiempo)
      display.set_pixel (x,y,0)
      sleep (tiempo)

  for y in range (4,-1,-1):
    for x in range (4,-1,-1):
      display.set_pixel(x,y,9)
      sleep (tiempo)
      display.set_pixel (x,y,0)
      sleep (tiempo)
```

Reto 14. Bucle for (repeticiones) con lista (array)
Reto consistente en encender de la primera fila de leds los leds impares (posiciones 1, 3 y 5), secuencialmente de izquierda a derecha cada 200 milisegundos.

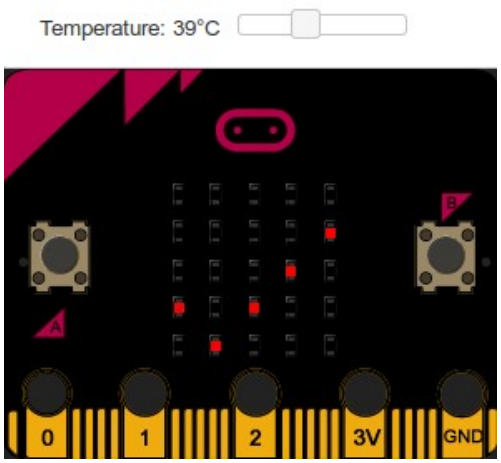
```
from microbit import *

pixels=[0,2,4]
```

```
while True:
    for i in range (3):
        display.set_pixel (pixels[i],0,9)
        sleep (200)
        display.set_pixel(pixels[i],0,0)
        sleep (200)
```

Reto 15. Muestra de temperatura.
Reto consistente en mostrar por pantalla la temperatura de microbit cada medio segundo.

```
from microbit import *

while True:
    temp=temperature()
    display.scroll(str(temp)+" C")
    sleep(500)
```

Reto 16. Termostato
Si la temperatura de microbit supera el umbral de 32ºC se muestra la imagen YES, si es menor se muestra la imagen NO.




```
from microbit import *

while True:
    if temperature()>32:
        display.show(Image.YES)
    else:
        display.show(Image.NO)
```

Reto 17. Encendido gradual de filas de leds en función de luz.
Se trata de encender filas de leds de la pantalla en función de la luz, a más luz más filas encendidas empezando de arriba (menor cantidad de luz) hacia abajo (mayor cantidad de luz).

```
from microbit import *

while True:
    luz=display.read_light_level()
    nivel=round((luz*5)/255)
    #display.scroll(nivel)
    for y in range (nivel):
        for x in range(5):
            display.set_pixel (x,y,9)
    sleep(500)
    display.clear()
```

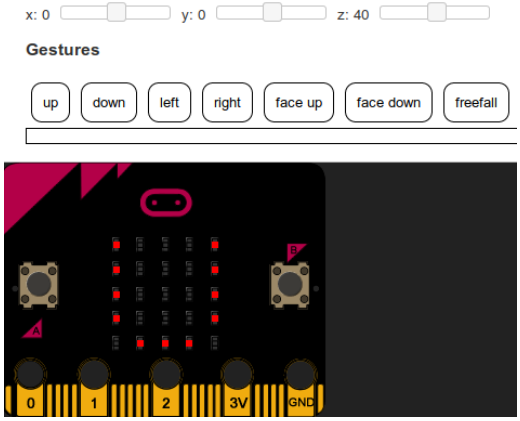
En el siguiente código enciende las filas de abajo hacia arriba:

```
from microbit import *

while True:
    luz=display.read_light_level()
    nivel=round((luz*5)/255)
    #display.scroll(nivel)
    for y in range (4,(4-nivel),-1):
```

```
            for x in range(5):
                display.set_pixel (x,y,9)
    sleep(500)
    display.clear()
```

Reto 18. Arriba (UP) o Abajo (DOWN). Jugando con el acelerómetro

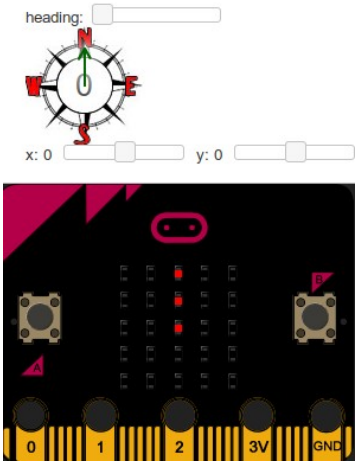
Leemos la aceleración en el eje Z y si su valor es mayor a 200 mg marcará en la pantalla la letra “U”, si es menor que -200 mg indicará la letra “D”, en caso contrario indicará una línea horizontal.




```
from microbit import *

while True:
    lectura = accelerometer.get_z()
    if lectura > 200:
        display.show("U")
    elif lectura < -200:
        display.show("D")
    else:
        display.show("-")
        sleep(500)
```

Reto 19. Brújula

Se trata de dar con las manecilla de la hora del reloj la dirección que marca la brújula previamente calibrada.




```
from microbit import *
compass.calibrate()
relojes=[Image.CLOCK1,Image.CLOCK2,Image.CLOCK3,Image.CLOCK4,Image.CLOCK5,Image.CLOCK6,Image.CLOCK7,Image.CLOCK8,Image.CLOCK9,Image.CLOCK10,Image.CLOCK11,Image.CLOCK12]
while True:
    rumbo=compass.heading()
    hora=int(round((rumbo/360)*12))
    display.show(relojes[hora-1])
    sleep(1000)
```

Reto 20. Botones y radio.

Se trata de enviar el texto “A” a los demás microbits del mismo grupo cuando se pulsa el botón A, y de enviar el texto “B” cuando se pulsa el botón B.

Código para emisor y receptor:

```
from microbit import *
import radio

radio.on()
radio.config(group=5)

while True:
        if button_a.is_pressed():
```

```
        radio.send (“A”)

    if button_b.is_pressed():
        radio.send(“B”)

    dato=radio.receive()

    if dato is not None:
        display.show(dato)
        sleep(500)
    display.clear()
```

Reto 21. Gestos y radio
Se trata de enviar el mensaje “shake” a los demás microbits del mismo grupo cuando hace el gesto “shake”. El botón “b” apaga la radio y sale del programa.

Código para emisor y receptor:

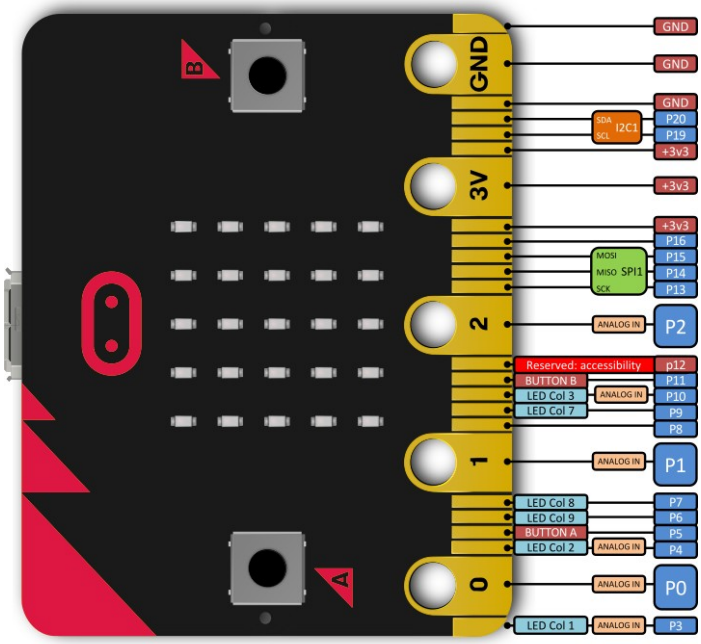
```
from microbit import *
import radio

radio.on()
radio.config(group=5)

while True:
    if button_b.is_pressed():
        display.show(Image.SNAKE)
        radio.off()
        break

    if accelerometer.is_gesture(“shake”):
        radio.send (“shake”)
    gesto=radio.receive()

    if gesto is not None:
        display.show(gesto)
        sleep (500)
    display.clear()
```

Reto 22. Registrador de datos por puerto serie (datalogger)
Se trata de llevar los datos de microbit a PC por la comunicación de puerto serie en este caso entre microbit y PC. En el ejemplo trasladamos al PC los datos de tiempo transcurridos en ms, temperatura y luminosidad, en el PC tenemos que tener un programa en formato terminal que muestre la conexión con el puerto serie, en mi caso en GNU/LINUX es Serial port terminal.

```
def grabadora():
    buffer=str(running_time())+","+ str(temperature())+","+str(display.read_light_level())+"\r\n"
    uart.write(buffer)
    sleep (250)

from microbit import *
uart.init(115200)
buffer=("tiempo"+","+"temperatura"+","+"luz"+"\r\n")
uart.write (buffer)
graba=False
display.show (Image.NO)
```

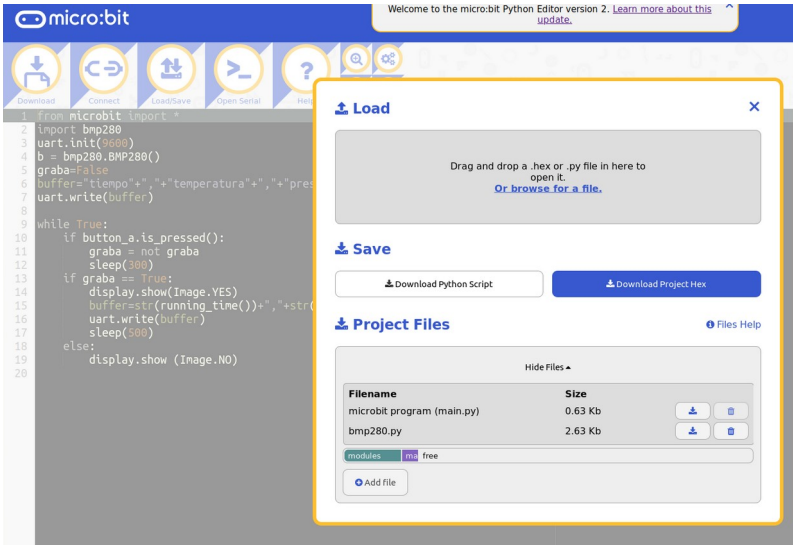
```
while True:
    if button_a.is_pressed():
        graba=not graba
        sleep(300)
    if button_b.is_pressed():
        display.show(Image.SNAKE)
        break
    if graba==True:
        display.show (Image.YES)
        grabadora()
    else:
        display.show (Image.NO)
```

¡¡¡ Próximamente retos con: pines. !!!



Glosario de instrucciones en Python y microbit

- Importación de librerías
  - from microbit import * : importa todas las librerías de microbit para poder usar sus objetos y funciones
  - Import nombre_libreria: para importar otras librerías hay que incluirlas en la carpeta del proyecto y llamarlas con import nombre_de libreria



- Objetos y procedimientos con microbit
  - display (matriz de leds de 5x5)
    - display.scroll (“texto” o str(variable)): muestra en modo texto en desplazamiento texto o variables
    - display.show (“texto” o str(variable)): muestra un texto o una variable, carácter a carácter
    - display.show (Image.NOMBRE_IMAGEN): muestra una imagen predeterminada
    - display.set_pixel (x,y,luminosidad): controla individualmente cada uno de los leds dónde x=posición en x del led de 0 a 4, y=posición en y del led de 0 a 4, luminosidad del led de 0 a 9
    - display.clear(): apaga todos los leds
    - display.read_light_level(): nos devuelve el nivel de luminosidad que detectan los leds de 0 a 255 (8 bits)
  - Pulsadores “a” y “b”
    - button_a.get.presses(): nos da el número de pulsaciones que ha tenido el botón correspondiente (a o b)
    - button_a.is_pressed(): nos indica si un pulsador (a o b) es pulsado (toma valores True o False)
  - Acelerómetro
    - accelerometer.get_x(): nos da el valor de la aceleración en el eje x en milig (puede llegar hasta +-2g (+-2000 mg))
    - accelerometer.get_y(): nos da el valor de la aceleración en el eje y en milig (puede llegar hasta +-2g (+-2000 mg))
    - accelerometer.get_z(): nos da el valor de la aceleración en el eje z en milig (puede llegar hasta +-2g (+-2000 mg))
    - accelerometer.get_values(): nos da una lista (array) con los valores de las tres aceleraciones
    - accelerometer.current_gesture(): devuelve el gesto actual, tenemos gestos como :"up", "down","left", "right", "face up", "face down", "freefall", "3g", "6g", "8g", "shake". Los gestos son cadena de texto.
    - accelerometer.is_gesture (“gesto”): nos comprueba un gesto concreto, devolviendo True o False

        - accelerometer.was_gesture (”gesto”): comprueba el último gesto que se produjo, devolviendo True o False.
        - accelerometer.get_gestures(): devuelve un historial de gestos en forma de lista (array)
    - Brújula
        - compass.calibrate(): realiza el proceso de calibración de la brújula
        - compass.is_calibrated(): nos devuelve True en caso de estar calibrada la brújula y False en caso contrario
        - compass.heading(): nos da la dirección de la brújula de 0º a 360º en sentido horario y en enteros, siendo 0º la dirección norte.
    - Radio
        - radio.on(): enciende la radio
        - radio.off(): apaga la radio
        - radio.config (group=0 a 255): configura varios aspecto de la comunicación radio. En este manual sólo nos centraremos en el grupo, que define al grupo establecido para enviar y recibir información a los microbits que compartan ese grupo
        - radio.send (“mensaje”): envia una cadena de texto
        - radio.recieve(): recibe una cadena de texto que le enviaron
    - UART

- Otas funciones de sistema
    - sleep (tiempo en milisegundos): crea una pausa de un tiempo en milisegundos
    - running_time(): devuelve el tiempo de funcionamiento del dispositivo
    - temperature(): devuelve la temperatura del sensor de temperatura

Referencias usadas:

- https://microbit.org/es/guide/python/