TM1638

Descripción de métodos

1. **tm1638.TM1638(stb, clk, dio):** Constructor que inicializa una instancia de la clase TM1638. Los parámetros son:

- **stb (Pin):** Pin de selección de chip (strobe).
- **clk (Pin):** Pin de reloj.
- **dio (Pin):** Pin de entrada/salida de datos.

Estos pines son utilizados para la comunicación con el chip TM1638.

2. **tm.number(value):** Este método muestra un número en el display de 7 segmentos. Toma como argumento:

value: El número a mostrar en el display (hasta 8 dígitos). Internamente convierte el número en un formato que el TM1638 puede mostrar en sus 8 posiciones.

3. **tm.led(position, state):** Controla el estado de un LED individual. Toma dos argumentos:

position: La posición del LED (entre 0 y 7).
state: Un valor booleano (True para encender el LED, False para apagarlo).

4. **tm.leds(pattern):** Controla los 8 LEDs simultáneamente usando un patrón de bits. Toma un argumento:

pattern: Un número binario de 8 bits donde cada bit representa el estado de un LED (1 para encendido, 0 para apagado). Por ejemplo, 0b01010101 encendería los LEDs en posiciones alternas.

5. **tm.show(text):** Muestra una cadena de caracteres en el display. Toma un argumento:

text: La cadena de caracteres (hasta 8 caracteres) a mostrar en el display.

6. **tm.keys():** Lee el estado de los botones conectados al TM1638. Devuelve un entero de 8 bits, donde cada bit representa el estado de un botón (1 si está presionado, 0 si no lo está).

Códigos TM1638

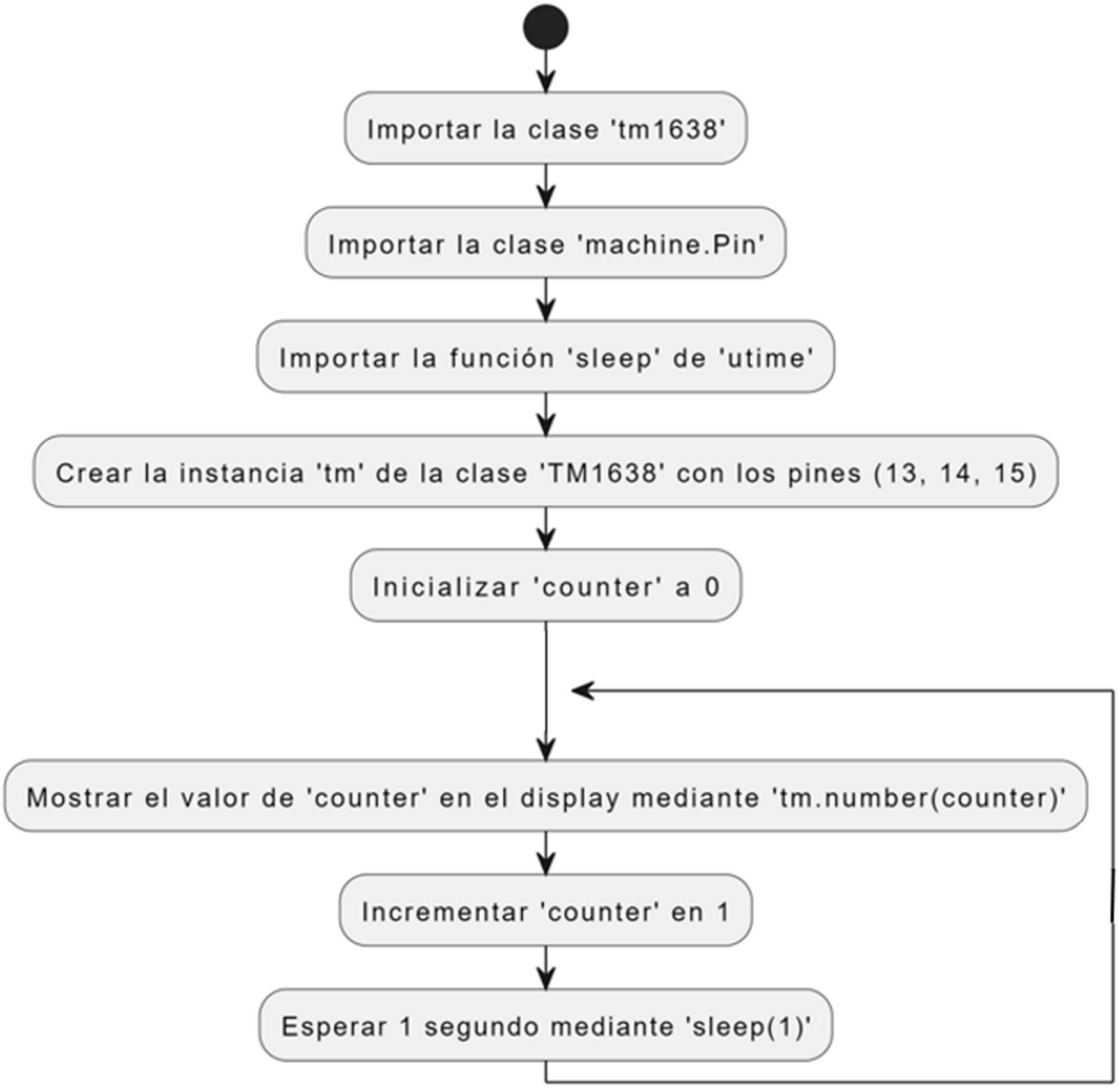
- ## Contador

```
import tm1638
# Aquí se importa la clase 'TM1638' desde la biblioteca 'tm1638'.
# Esta clase representa un objeto que controla un módulo TM1638.
# En programación orientada a objetos, 'TM1638' es la clase que define el comportamiento del display.
# Como clase, define cómo se puede interactuar con el display TM1638.

from machine import Pin
# 'Pin' es una clase del módulo 'machine'. Esta clase representa la interfaz de control para los pines físicos del microcontrolador.
# Cada objeto que crees de esta clase será una instancia de un pin en específico. Es parte del hardware abstracto que controlas.

from utime import sleep
# 'sleep' es una función externa (no un método ni parte de una clase) que permite pausar la ejecución del programa.

# A continuación, se crea una instancia de la clase TM1638, con varios atributos configurados mediante el constructor de la clase.

tm = tm1638.TM1638(stb=Pin(13), clk=Pin(14), dio=Pin(15))
# Se está creando una instancia de la clase 'TM1638'.
# El objeto se llama 'tm', y se inicializa utilizando el constructor de la clase 'TM1638'.
# En este caso, se le están pasando tres pines al constructor como atributos (stb, clk, dio).
# 'stb', 'clk', y 'dio' son parámetros del constructor, y cada uno se configura con un objeto de la clase 'Pin'.
# Estos pines son atributos que se almacenarán en el objeto 'tm' y se usarán para comunicarse con el display.
# El método __init__ de 'TM1638' es el constructor que inicializa la comunicación con el display a través de los pines dados.

counter = 0
# 'counter' es una variable simple (no parte de una clase).
# En POO, podría ser considerado un atributo si fuera parte de una clase.
# En este caso, está fuera de cualquier clase y es simplemente un valor que se incrementa en el ciclo.

while True:
    # Aquí se inicia un bucle infinito. No está vinculado a ninguna clase, pero sería parte de un flujo de control en POO.

    tm.number(counter)
    # 'tm' es la instancia de la clase 'TM1638'.
    # Se llama al método 'number()' de la clase 'TM1638' usando la instancia 'tm'.
    # 'number()' es un método público que toma un argumento (counter) y lo muestra en el display.
    # Este método es parte de la interfaz pública de la clase, y su función es mostrar un número en el display de la instancia.

    counter += 1
    # Aquí se incrementa el valor de 'counter'. En una implementación orientada a objetos,
    # 'counter' podría ser un atributo de la clase y este incremento podría suceder dentro de un método.
```

sleep(1)

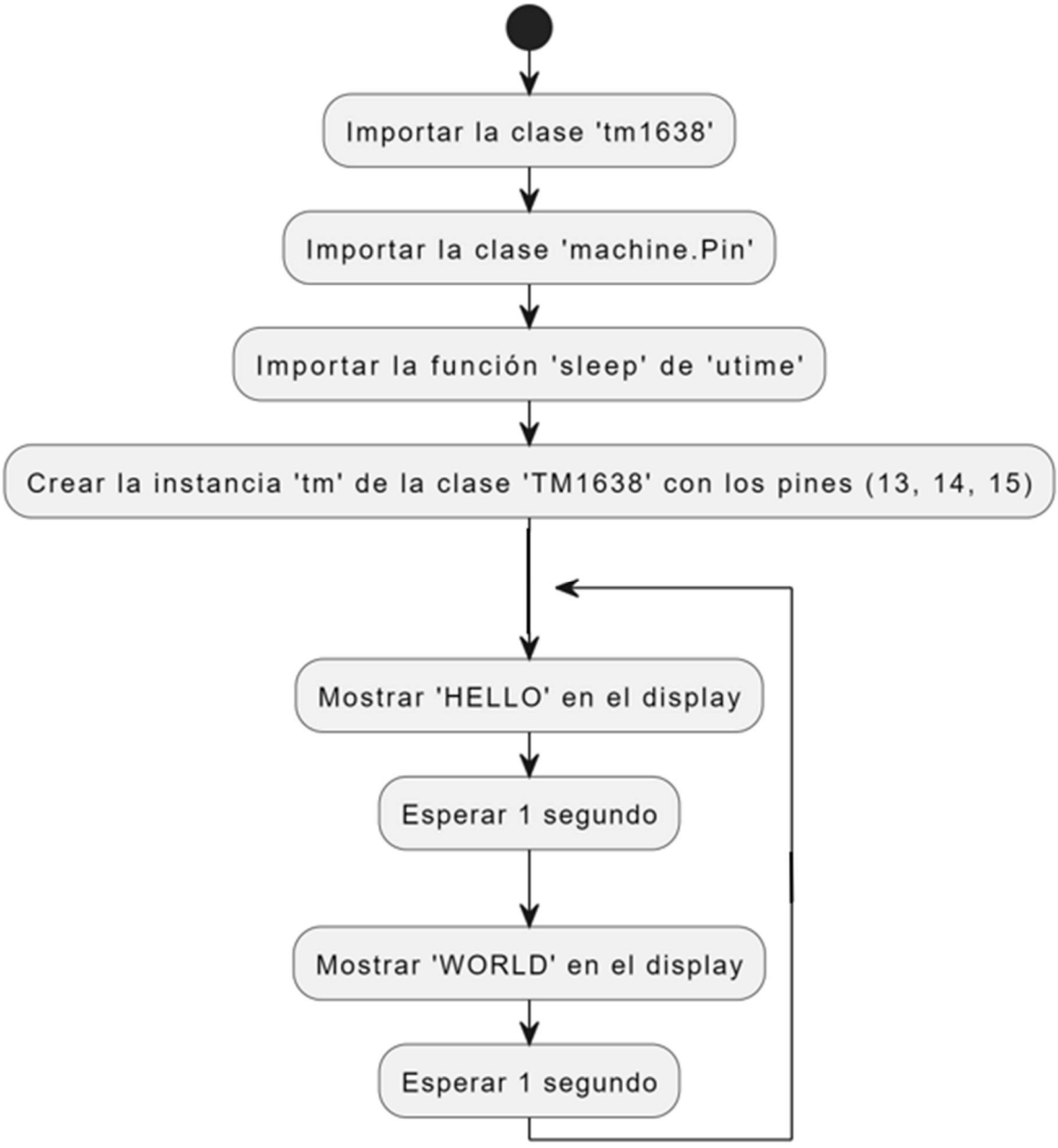
Se invoca la función 'sleep()' para pausar la ejecución del programa durante un segundo.

En POO, esto podría ser parte de la lógica interna de un método de una clase que controla la frecuencia de actualización.



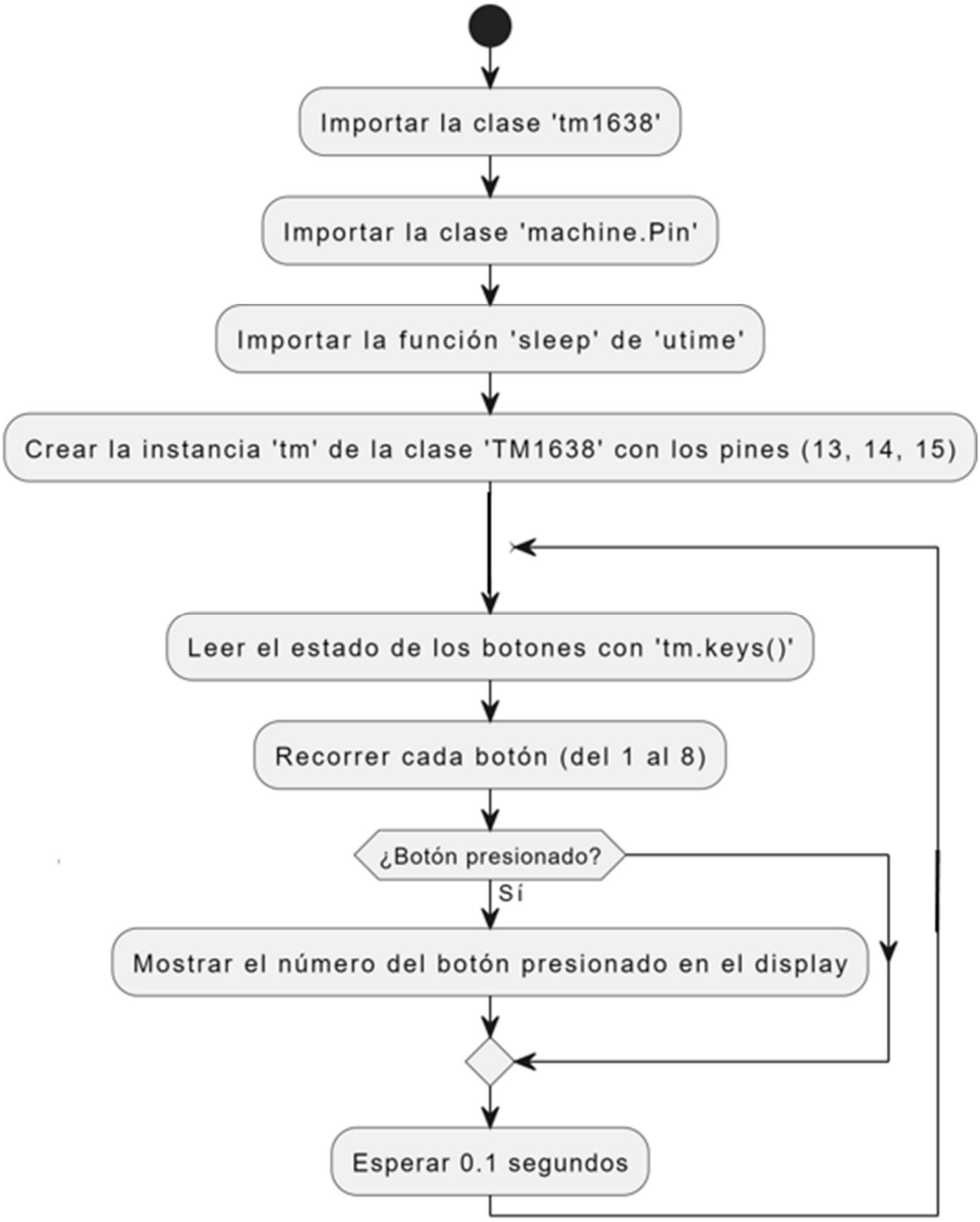
- **holamundotm1638**

```
import tm1638
# Aquí se importa la clase 'TM1638' desde la biblioteca 'tm1638'.
# Esta clase representa un objeto que controla un módulo TM1638.
# Como clase, define cómo interactuar con el display TM1638, que es una pantalla de 7 segmentos.

from machine import Pin
# 'Pin' es una clase del módulo 'machine'. Esta clase representa la interfaz de control para los pines físicos del microcontrolador.
# Cada objeto que crees de esta clase será una instancia que representa un pin del microcontrolador.

from utime import sleep
# 'sleep' es una función externa, que no pertenece a ninguna clase. Pausa la ejecución del programa durante un tiempo determinado.

# A continuación, se crea una instancia de la clase 'TM1638', con varios atributos configurados mediante su constructor.
tm = tm1638.TM1638(stb=Pin(13), clk=Pin(14), dio=Pin(15))
# Se está creando una instancia de la clase 'TM1638'.
# La instancia se llama 'tm', y se inicializa utilizando el constructor de la clase 'TM1638'.
# El constructor recibe tres objetos de la clase 'Pin' (stb, clk, dio) como atributos, que representan los pines físicos necesarios para controlar el display.
# 'stb', 'clk', y 'dio' son parámetros del constructor que definen los pines utilizados para la comunicación con el módulo TM1638.

while True:
    # Se inicia un bucle infinito, no asociado directamente a ninguna clase.
    # Este bucle es parte de la lógica de control que se repetirá indefinidamente mientras el programa esté en ejecución.

    tm.show('  HELLO  ')
    # 'tm' es la instancia de la clase 'TM1638'.
    # Se llama al método 'show()' de la clase 'TM1638' usando la instancia 'tm'.
    # El método 'show()' recibe un argumento, que es una cadena de texto ('HELLO'), y su función es mostrar ese texto en el display TM1638.
    # Este método es parte de la interfaz pública de la clase 'TM1638', y su función es permitir que el display muestre el texto pasado como parámetro.

    sleep(1)
    # Se llama a la función 'sleep()', que pausa la ejecución del programa durante 1 segundo.
    # Esto permite que el texto 'HELLO' se mantenga en la pantalla del display TM1638 durante ese tiempo.

    tm.show(' WORLD ')
    # Se llama nuevamente al método 'show()' de la instancia 'tm', esta vez con el argumento 'WORLD'.
    # El método 'show()' actualizará el display TM1638 para mostrar el nuevo texto ('WORLD').

    sleep(1)
    # Nuevamente, se pausa la ejecución durante 1 segundo para que el texto 'WORLD' permanezca visible en el display por ese tiempo.
```


Importar la clase 'tm1638'
Importar la clase 'machine.Pin'
Importar la función 'sleep' de 'utime'
Crear la instancia 'tm' de la clase 'TM1638' con los pines (13, 14, 15)
Mostrar 'HELLO' en el display
Esperar 1 segundo
Mostrar 'WORLD' en el display
Esperar 1 segundo

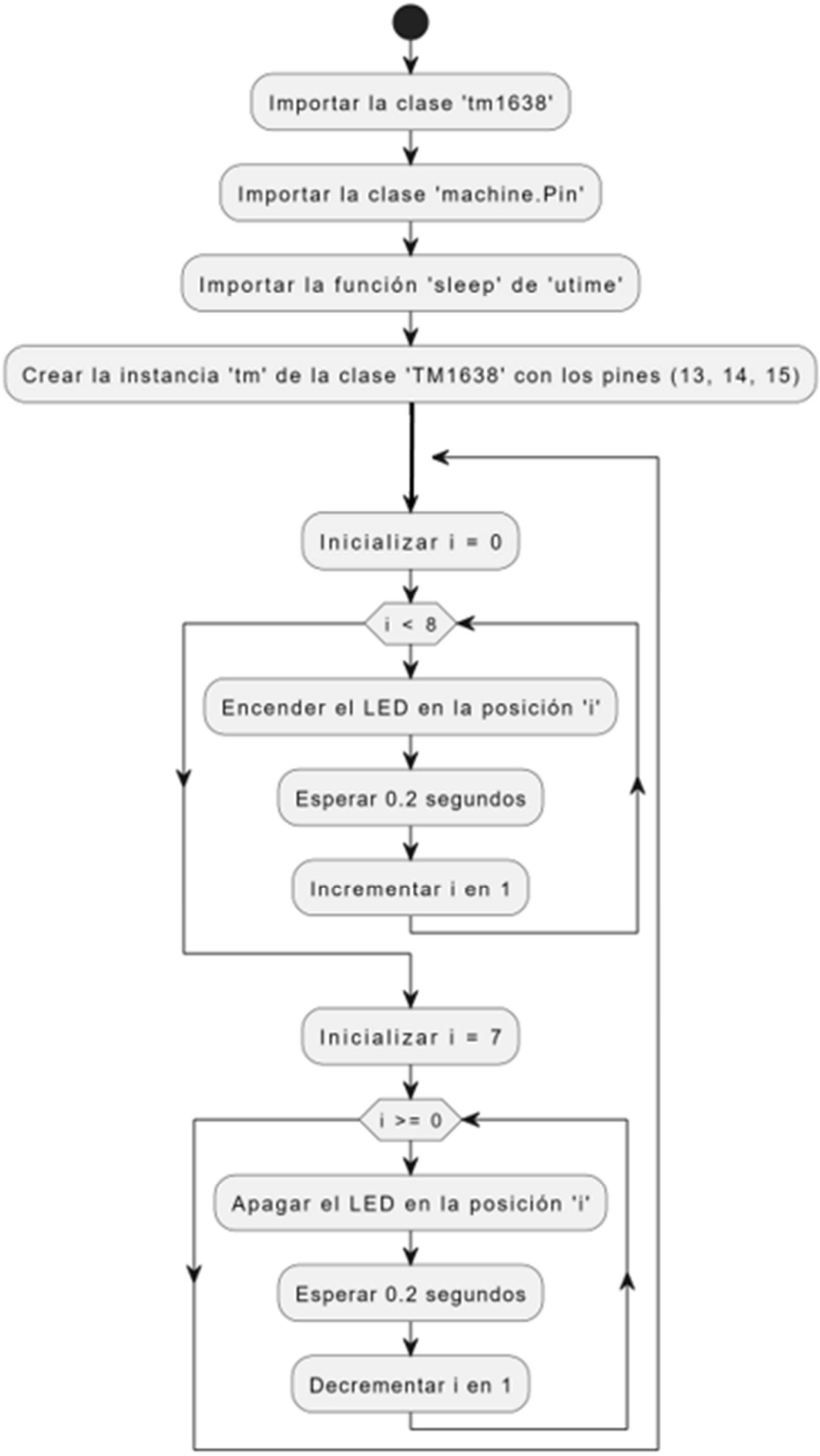
- **lecturabotones**

```
import tm1638
# Se importa la clase 'TM1638' desde la biblioteca 'tm1638'.
# Esta clase representa un objeto que controla un módulo TM1638, que incluye botones, LEDs y un display de 7 segmentos.

from machine import Pin
# 'Pin' es una clase del módulo 'machine'. Esta clase permite controlar los pines de entrada/salida en un microcontrolador.

from utime import sleep
# 'sleep' es una función externa que no pertenece a ninguna clase. Pausa la ejecución del programa por el tiempo especificado (en segundos).

# Se crea una instancia de la clase 'TM1638' con los pines de control del display.
tm = tm1638.TM1638(stb=Pin(13), clk=Pin(14), dio=Pin(15))
# Se está creando una instancia de la clase 'TM1638' llamada 'tm'.
# Esta instancia utiliza tres pines: 'stb', 'clk', y 'dio', que son configurados como objetos de la clase 'Pin'.
# Estos pines controlan la comunicación entre el microcontrolador y el módulo TM1638.

while True:
  # Inicia un bucle infinito, que repetirá continuamente la lectura de los botones y mostrará el número del botón presionado.

  pressed = tm.keys()  # Leer los botones
  # 'pressed' es una variable que almacena el valor retornado por el método 'keys()' de la clase 'TM1638'.
  # El método 'keys()' devuelve un valor entero donde cada bit representa el estado de un botón.
  # Si un bit es 1, significa que el botón correspondiente está presionado.

  for i in range(8):
    # Se inicia un bucle 'for' que recorre los 8 botones disponibles en el módulo TM1638.

    if ((pressed >> i) & 1):
      # Se utiliza un desplazamiento de bits (right shift) y una operación AND bit a bit para comprobar si el botón en la posición 'i' está presionado.
      # Si el bit en la posición 'i' es 1, significa que el botón correspondiente ha sido presionado.

      tm.number(i + 1)
      # Se llama al método 'number()' de la instancia 'tm' para mostrar el número del botón presionado en el display.
      # 'i + 1' es el número que se mostrará en el display, representando el botón presionado (ya que los botones están numerados de 1 a 8).

  sleep(0.1)
  # Se pausa la ejecución del programa durante 0.1 segundos antes de volver a leer los botones.
  # Esto ayuda a evitar que el programa responda demasiado rápido o muestre el número del botón múltiples veces por una sola pulsación.
```


Importar la clase 'tm1638'
Importar la clase 'machine.Pin'
Importar la función 'sleep' de 'utime'
Crear la instancia 'tm' de la clase 'TM1638' con los pines (13, 14, 15)
Leer el estado de los botones con 'tm.keys()'
Recorrer cada botón (del 1 al 8)
¿Botón presionado?
Sí
Mostrar el número del botón presionado en el display
Esperar 0.1 segundos

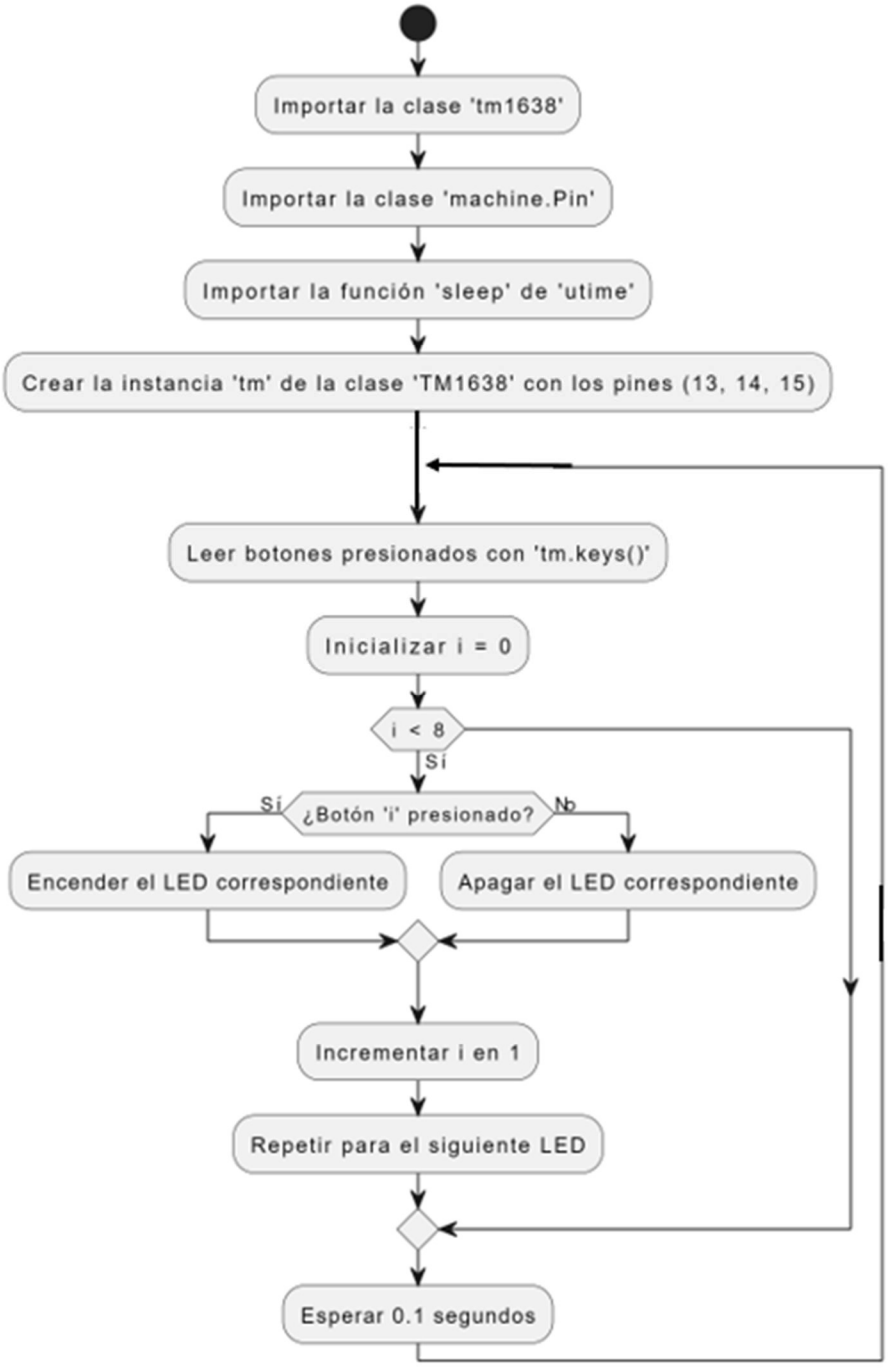
- **ledsintermitentes**

```
import tm1638
# Se importa la clase 'TM1638' desde la biblioteca 'tm1638'.
# Esta clase permite controlar un módulo TM1638, que incluye LEDs, botones y un display de 7 segmentos.

from machine import Pin
# 'Pin' es una clase que representa los pines de entrada/salida del microcontrolador.
# Se utiliza para configurar los pines que se usan para la comunicación con el módulo TM1638.

from utime import sleep
# 'sleep' es una función externa, no parte de ninguna clase.
# Esta función pausa la ejecución del programa durante el tiempo especificado (en segundos).

# Se crea una instancia de la clase 'TM1638', que manejará los LEDs y el display.
tm = tm1638.TM1638(stb=Pin(13), clk=Pin(14), dio=Pin(15))
# Se está creando una instancia llamada 'tm' de la clase 'TM1638'.
# Esta instancia utiliza los pines 13, 14 y 15 para comunicarse con el módulo TM1638.
# Los pines se configuran mediante objetos de la clase 'Pin'.

while True:
  # Inicia un bucle infinito que repetirá continuamente la secuencia de encender y apagar los LEDs.

  # Encender LEDs de izquierda a derecha
  for i in range(8):
    # Este bucle 'for' recorre los LEDs de izquierda a derecha (del 0 al 7).
    tm.led(i, True)
    # Se llama al método 'led()' de la instancia 'tm' para encender el LED en la posición 'i'.
    # El segundo argumento 'True' indica que el LED debe encenderse.

    sleep(0.2)
    # Pausa la ejecución durante 0.2 segundos antes de encender el siguiente LED.

  # Apagar LEDs de derecha a izquierda
  for i in range(7, -1, -1):
    # Este bucle 'for' recorre los LEDs de derecha a izquierda (del 7 al 0).
    tm.led(i, False)
    # Se llama nuevamente al método 'led()', pero esta vez con el segundo argumento 'False' para apagar el LED.

    sleep(0.2)
    # Pausa la ejecución durante 0.2 segundos antes de apagar el siguiente LED.
```


Importar la clase 'tm1638'
Importar la clase 'machine.Pin'
Importar la función 'sleep' de 'utime'
Crear la instancia 'tm' de la clase 'TM1638' con los pines (13, 14, 15)
Inicializar i = 0
i < 8
Encender el LED en la posición 'i'
Esperar 0.2 segundos
Incrementar i en 1
Inicializar i = 7
i >= 0
Apagar el LED en la posición 'i'
Esperar 0.2 segundos
Decrementar i en 1

- **ledsybotones**

```
import tm1638
# Se importa la clase 'TM1638' desde la biblioteca 'tm1638'.
# Esta clase permite controlar un módulo TM1638, que incluye LEDs, botones y un display de 7 segmentos.

from machine import Pin
# 'Pin' es una clase que representa los pines de entrada/salida del microcontrolador.
# Se utiliza para configurar los pines que se usan para la comunicación con el módulo TM1638.

from utime import sleep
# 'sleep' es una función externa, no parte de ninguna clase.
# Esta función pausa la ejecución del programa durante el tiempo especificado (en segundos).

# Se crea una instancia de la clase 'TM1638', que manejará los LEDs, botones y el display.
tm = tm1638.TM1638(stb=Pin(13), clk=Pin(14), dio=Pin(15))
# Se está creando una instancia llamada 'tm' de la clase 'TM1638'.
# Esta instancia utiliza los pines 13, 14 y 15 para comunicarse con el módulo TM1638.
# Los pines se configuran mediante objetos de la clase 'Pin'.

while True:
  # Inicia un bucle infinito que repetirá continuamente la lectura de los botones y el control de los LEDs.

  pressed = tm.keys()  # Leer botones presionados
  # 'pressed' es una variable que almacena el valor retornado por el método 'keys()' de la clase 'TM1638'.
  # El método 'keys()' devuelve un valor entero donde cada bit representa el estado de un botón.
  # Si un bit es 1, significa que el botón correspondiente está presionado.

  for i in range(8):
    # Este bucle 'for' recorre los 8 botones y LEDs disponibles en el módulo TM1638.

    if ((pressed >> i) & 1):
      # Se utiliza un desplazamiento de bits (right shift) y una operación AND bit a bit para comprobar si el botón en la posición 'i'
está presionado.
      # Si el bit en la posición 'i' es 1, significa que el botón correspondiente ha sido presionado.

      tm.led(i, True)  # Encender el LED correspondiente
      # Se llama al método 'led()' de la instancia 'tm' para encender el LED en la posición 'i'.

    else:
      tm.led(i, False)  # Apagar el LED si no está presionado
      # Si el botón no está presionado, se apaga el LED correspondiente usando el método 'led()'.

  sleep(0.1)
  # Se pausa la ejecución del programa durante 0.1 segundos antes de volver a leer los botones.
```


Importar la clase 'tm1638'
Importar la clase 'machine.Pin'
Importar la función 'sleep' de 'utime'
Crear la instancia 'tm' de la clase 'TM1638' con los pines (13, 14, 15)
Leer botones presionados con 'tm.keys()'
Inicializar i = 0
i < 8
Sí
¿Botón 'i' presionado?
Sí
No
Encender el LED correspondiente
Apagar el LED correspondiente
Incrementar i en 1
Repetir para el siguiente LED
Esperar 0.1 segundos