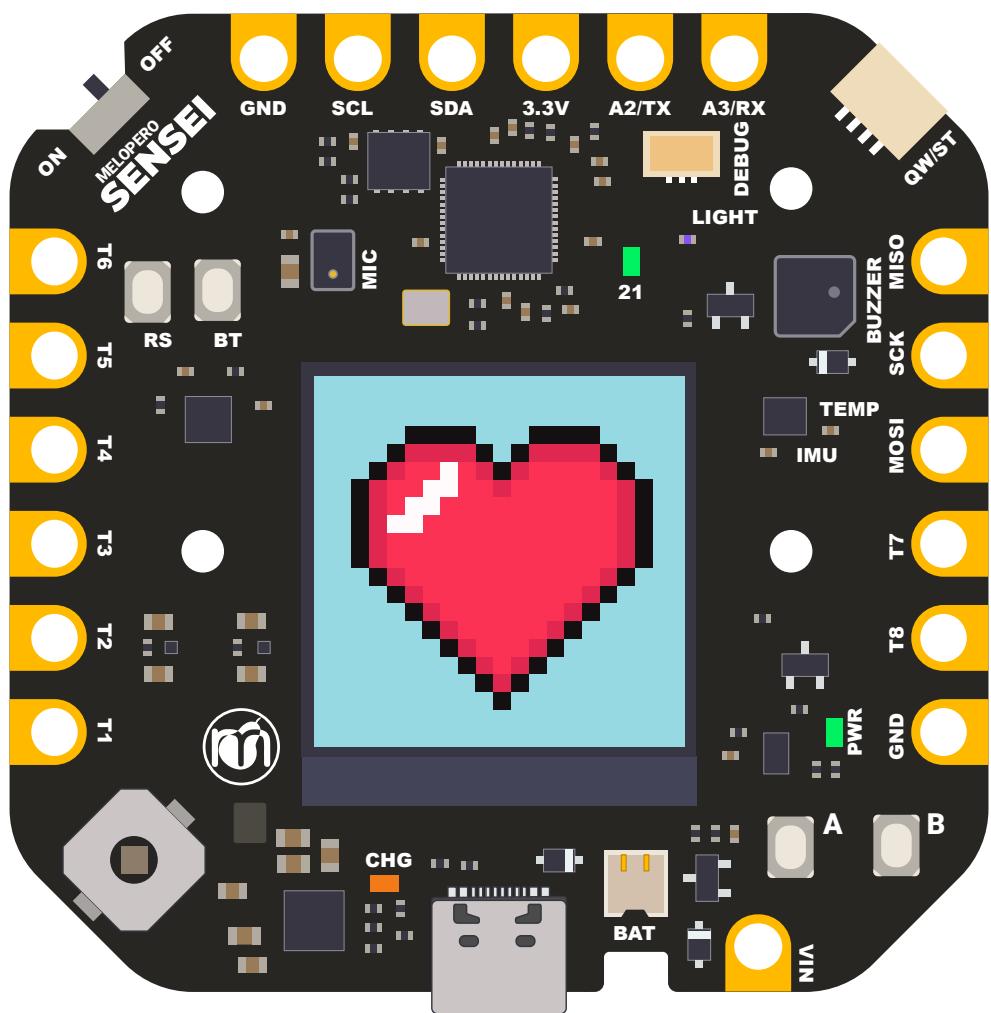


# Guida rapida

# Melopero SENSEI

v0.9.1



Questa guida è ancora in versione beta e viene costantemente aggiornata con correzioni e nuovi contenuti.

Versione **0.9.1**  
13 Dicembre 2023

# Contents

1. Panoramica Hardware .....	4
2. Pinout .....	5
3. Alimentazione e ricarica della batteria .....	6
5. Sensei software pack .....	7
5.1 Download .. . . . .	7
6. MicroPython .....	8
6.1 Installare MicroPython . . . . .	8
6.2 Installazione di Thonny. . . . .	9
6.3 Thonny - avvio rapido. . . . .	10
6.4 The REPL. . . . .	10
6.5 Libreria Sensei per MicroPython . . . . .	11
7. Esempi MicroPython .....	15
7.1 Display basics 1 . . . . .	15
7.2 Display basics 2 . . . . .	16
7.3 Input basics . . . . .	17
7.4 Touch interrupt . . . . .	18
7.5 Read battery . . . . .	19
7.6 Read light sensor . . . . .	20
7.7 IMU read values . . . . .	21
7.8 Freefall and tap detection . . . . .	22
7.9 Step counter . . . . .	23
7.10 Fitness device . . . . .	24
8. Reset della Flash Memory .....	25

# 1. Panoramica Hardware

Melopero Sensei è una scheda di sviluppo basata sul microcontrollore Raspberry Pi RP204 e dotata di display, sensori, buzzer, led e pulsanti.

Può essere programmata in C/C++, MicroPython, Arduino Language (presto disponibile) e altro (non supportato).

Specifiche:

- memoria flash 8MB
- pulsanti di reset e boot
- connettore Qwiic/Stemma
- connettore SWD per il debugging
- interruttore ON/OFF
- Display TFT 1.3" 240x240pixel a colori
- Buzzer
- microfono PDM
- sensore di luminosità
- accelerometro/giroscopio/sensore di temperatura
- sensore touch fino a 12 canali
- joystick e due pulsanti programmabili
- circuito ricarica della batteria
- connettore posteriore per moduli sensei backpack (presto disponibili)
- connettore USB-C per alimentare, programmare e ricaricare.
- Led verde programmabile sul pin 21
- Led di accensione (verde)
- Led di ricarica in corso della batteria (verde)
- 4 fori per il montaggio
- Programmabile in C/C++, MicroPython, Arduino language (presto disponibile).

Dimensioni: 65mm x 65mm

## 2. Pinout

# MELOPERO SENSEI

## PINOUT



Raspberry Pi  
Approved Reseller

⚠ GPIO Max Current 12mA per pin

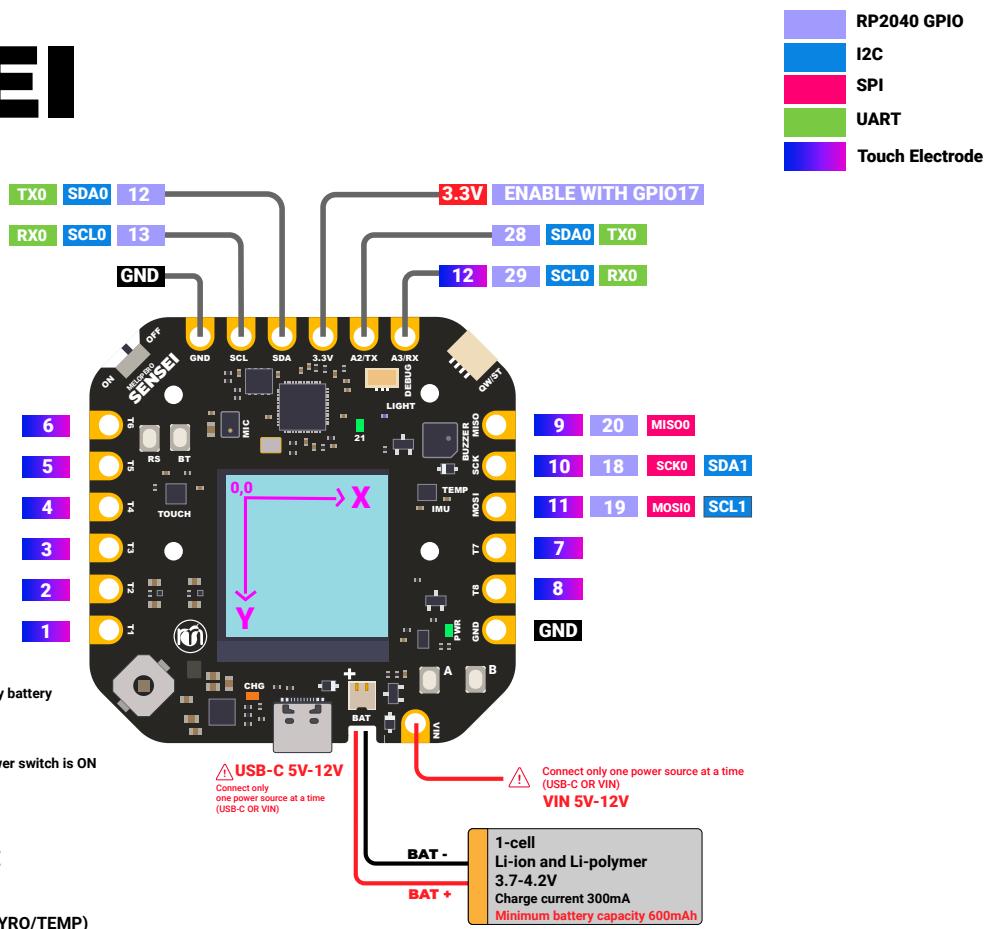
█ User LED **21**

█ CHG LED

Blinks in case of battery disconnected or faulty battery  
Fixed while charging  
OFF if charge is completed

█ PWR LED

ON if a power source is connected and the power switch is ON



### SENSORS:

INTERRUPT		5	I2C 1 0x5A	MPR121 (TOUCH)
ENABLE POWER WITH GPIO17		22 23	I2C 1 0x6A	LSM6DSL (ACC/GYRO/TEMP)
2 / CLK	3 / DATA			MP34DT06J (PDM MICROPHONE)
		15		BUZZER
		27/A1		ALS-PT19-315C/L177/TR8 (AMBIENT LIGHT)
		26/A0		BATSENSE

### 3. Alimentazione e ricarica della batteria

Melopero Sensei è dotata di un circuito di ricarica per batterie Li-Ion Li-Poly a 1-cell con tensione 3.7-4.2V. La corrente di ricarica è di circa 300mA, quindi la capacità minima della batteria deve essere di almeno 600mAh. Sullo schema pinout è indicata la polarità della batteria.

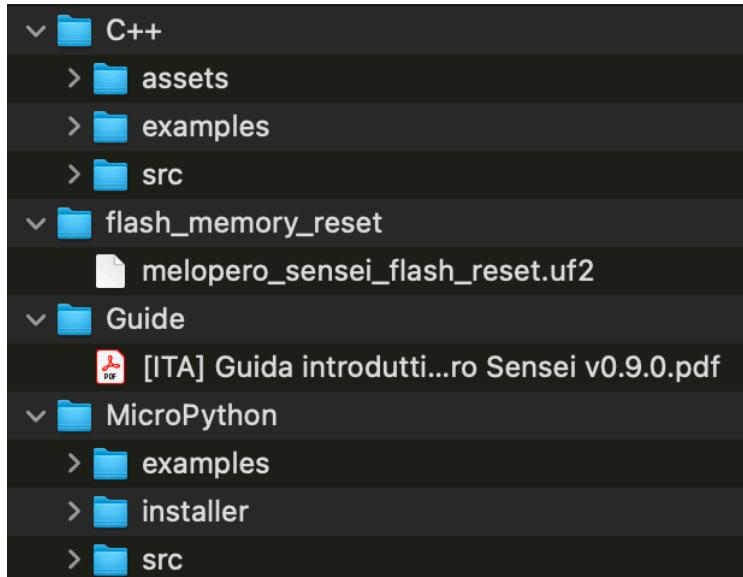
Il charger si occuperà di ricaricare la batteria quando sarà presente una fonte di alimentazione sufficiente. Tramite la porta USB-C è possibile alimentare e programmare la Sensei e ricaricare contemporaneamente la batteria con un cavo USB direttamente dal proprio PC.

Il charger è impostato per richiedere al massimo 500mA dalla USB-C o VIN

**IMPORTANTE: Nonostante la presenza di diodi di protezione, raccomandiamo fortemente di non alimentare mai la scheda utilizzando insieme sia la porta USB-C che il pad VIN.**

# 5. Sensei software pack

Il Sensei software pack contiene il file di installazione di MicroPython per la Sensei con la libreria integrata e i file sorgente e gli esempi, la libreria C++ con gli esempi, la guida introduttiva, un file per resettare la memoria flash. La libreria Arduino (non ancora disponibile) potrà essere scaricata direttamente dal gestore delle librerie dell'IDE Arduino.



## 5.1 Download

Per scaricare il software pack andare al seguente indirizzo:

[https://github.com/melopero/Melopero\\_Sensei/archive/refs/heads/main.zip](https://github.com/melopero/Melopero_Sensei/archive/refs/heads/main.zip)

# 6. MicroPython

MicroPython comprende un sottoinsieme di funzioni del linguaggio di programmazione Python 3 ottimizzato per essere eseguito su microcontrollori.

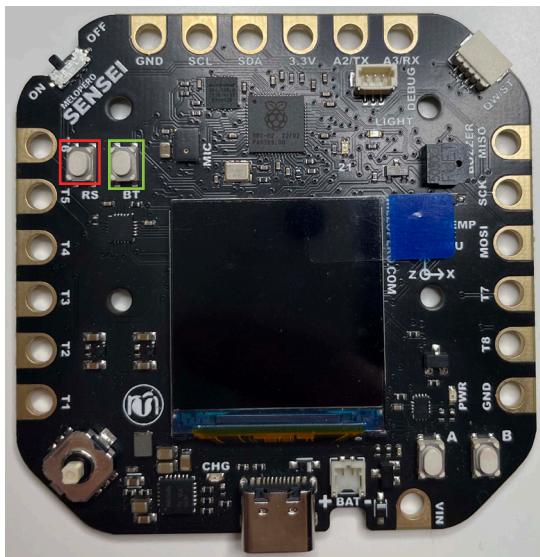
All'interno del Sensei software pack è presente il file per installare una versione di MicroPython che include già la libreria software per gestire i sensori e il display.

## 6.1 Installare MicroPython

Dopo aver scaricato l'ultima versione del Sensei software pack (vedi capitolo dedicato), attiva la modalità di avvio del bootloader sulla tua scheda e trascina (o copia e incolla) il file di installazione .uf2 di MicroPython al suo interno.

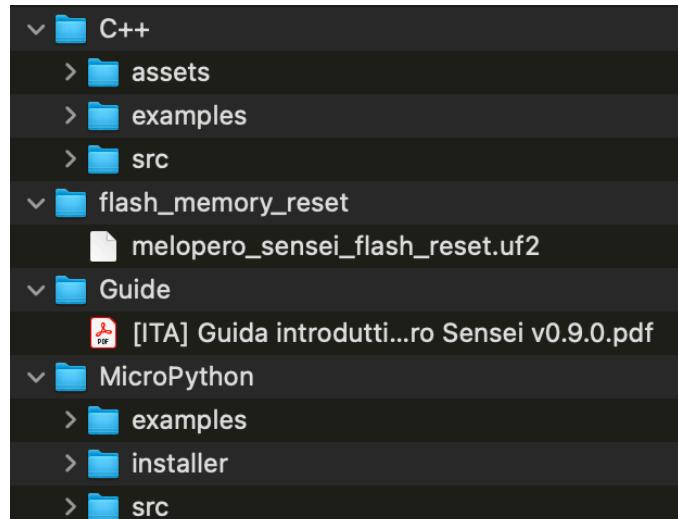
Abbiamo incluso anche la libreria melopero\_sensei nella versione che installerai, quindi non sono necessarie installazioni aggiuntive.

Per attivare la modalità bootloader, quando la Sensei è già collegata alla porta USB del tuo computer, premi e mantieni premuto il pulsante BOOT / BT (cerchiato in verde nell'immagine sotto), quindi premi e rilascia il pulsante di reset (cerchiato in rosso). Continua a tenere premuto il pulsante BOOT / BOOTSEL finché compare il drive RPI-RP2.



Trascina sul drive RPI-RP2 (o copia e incolla) il file installer di MicroPython che trovi nella cartella MicroPython/installer.

L'installazione durerà qualche secondo



## 6.2 Installazione di Thonny

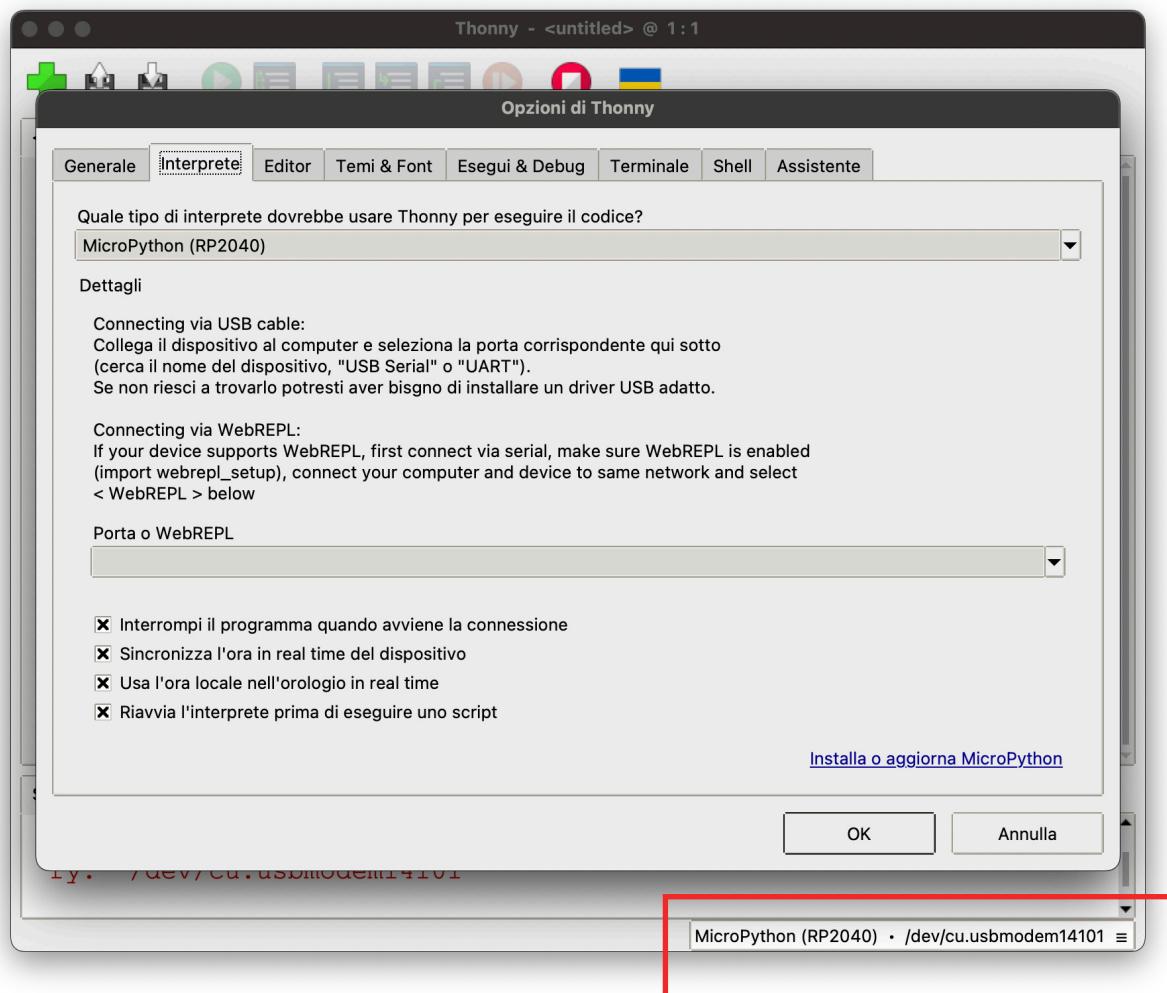
Thonny è un editor di codice Python per programmatori principianti ed è l'editor consigliato per la programmazione su schede basate su RP2040 con MicroPython.

È disponibile per Windows, Mac e Linux all'indirizzo seguente: <https://thonny.org>

La prima volta che eseguirai l'editor Thonny, dovrà impostare **MicroPython (RP2040)** e selezionare la porta giusta per il tuo dispositivo collegato (attenzione questa porta potrebbe cambiare a seguito di nuove installazioni di MicroPython sulla Sensei)

Vai su Preferenze (potrebbe essere Opzioni/Impostazioni su Windows), fai clic sulla scheda "Interprete" e seleziona quindi Micropython (Raspberry Pi Pico) e la porta corretta dal menu delle porte (la tua scheda deve essere collegata e con MicroPython già installato).

Se stai usando Thonny in modalità semplice, puoi accedere al pannello di configurazione dell'interprete facendo clic sull'angolo in basso a destra della finestra di Thonny (cerchiato in rosso sotto).



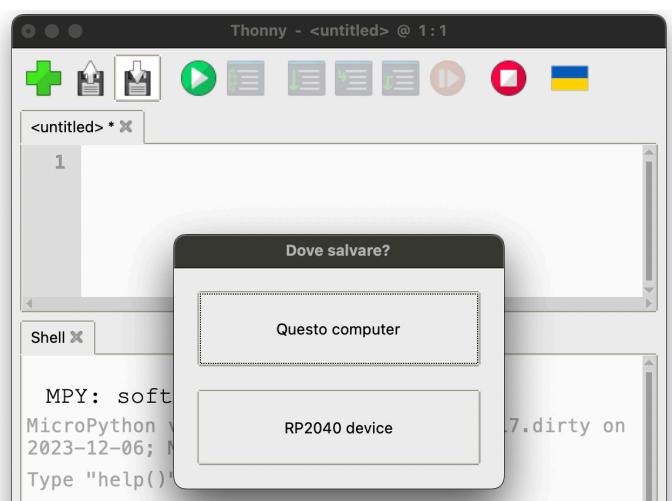
## 6.3 Thonny - avvio rapido

Dopo aver installato MicroPython sulla Sensei, apri Thonny e fai clic su STOP/RESTART. Leggerai il seguente testo nella shell nella parte inferiore della finestra quando Cookie sarà pronto a ricevere istruzioni:

```
MicroPython v1.22.0-preview.132.g958c6d917.dirty on 2023-12-06; Melopero Sensei wi  
th RP2040  
Type "help()" for more information.  
>>>
```

Scrivi del codice nell'area di script di Thonny o apri l'esempio di codice MicroPython incluso nel pacchetto software (vedi capitolo 3 per scaricare il pacchetto software). Quando sei pronto, fai clic su RUN o SAVE. Thonny ti chiederà dove vuoi eseguire o salvare lo script: seleziona "RP2040 Device". Nel caso in cui stai salvando il file, dagli un nome che includa l'estensione .py, ad esempio myfile.py. Se salvi il file come main.py, il tuo codice verrà eseguito automaticamente ogni volta che alimenti il Cookie.

Ora che il tuo file è stato salvato, esegui il programma facendo clic su RUN (pulsante PLAY verde). Se il tuo codice include delle stampe, le vedrai nella Shell. Prima di salvare un nuovo file, ricorda di fare clic sul pulsante "stop/restart" se il tuo Cookie sta già eseguendo un programma.



## 6.4 The REPL

Il REPL, acronimo di Read-Evaluate-Print-Loop, ti consente di eseguire direttamente nella console le righe di codice e ottenere un risultato immediato. Nella Shell, prova a eseguire il comando `print("hello world")` e premi invio. Il REPL interpreterà la riga di codice e ti restituirà il risultato, in questo caso stamperà "hello world".

```
Shell >  
MicroPython v1.16-341-g215292166-dirty on 2021-10-06; Melopero Shake RP2040 with RP2040  
Type "help()" for more information.  
>>> print("Hello World")  
Hello World  
>>> |
```

MicroPython (Raspberry Pi Pico)

# 6.5 Libreria Sensei per MicroPython

Il modo più veloce per iniziare a programmare la Melopero Sensei in MicroPython è tramite l'esempio principale, che utilizza tutte le funzioni disponibili e contiene dei commenti al codice.  
La libreria per la Sensei include funzioni per controllare il display, i sensori touch, accelerometro, giroscopio e temperatura, il sensore di luminosità, il buzzer, il led, il joystick e i pulsanti A/B.

## **MeloperoSensei()**

Crea un nuovo oggetto Sensei

### **update\_display()**

Questa funzione è fondamentale per rendere effettive le modifiche applicate al display. Deve essere chiamata per ultima dopo aver apportato le modifiche che si desidera vengano visualizzate sullo schermo, incluso la cancellazione del display.

### **clear\_display()**

Questa funzione cancella tutto il display, impostando il colore 0,0,0 (nero). Anche in questo caso deve essere chiamata la funzione update\_display() per rendere effettive le modifiche.

### **set\_display\_color(red, green, blue)**

Questa funzione cancella tutto il display, impostando il colore definito dalle componenti rosso, verde e blu. Tali componenti possono avere valori che vanno da 0 a 255.

### **set\_text\_font(font,size)**

Imposta il font e la dimensione del testo

Come font è possibile passare alla funzione uno tra i seguenti:

TEXT\_MONO, TEXT\_MONO\_BOLD, TEXT\_SANS, TEXT\_SANS\_BOLD

Le dimensioni impostabili sono:

TEXT VERY\_SMALL, TEXT\_SMALL, TEXT\_REGULAR, TEXT\_BIG

### **set\_text\_color(red, green, blue)**

Imposta il colore del font

prende come parametri le componenti rosso, verde e blu.

per componente i valori permessi sono 0-255

### **write\_text(text,x,y)**

Scrive del testo nella posizione dello schermo (x,y).

prende come parametri il testo e le coordinate x e y del punto dove iniziare a scrivere.

Per impostare il font, la dimensione e il colore del testo, è necessario chiamare prima le funzioni apposite

### **draw\_pixel(x, y, red, green, blue)**

Colora il pixel nella posizione x, y.

Il colore è definito nelle sue componenti rosso, verde e blu.

I valori delle componenti possono variare da 0 a 255.

### **draw\_line(startx, starty, endx, endy, red, green, blue)**

Disegna una linea che parte dal punto con coordinate (startx, starty) e arriva fino al punto (endx, endy).

Il colore è definito nelle sue componenti rosso, verde e blu.

I valori delle componenti possono variare da 0 a 255.

### **draw\_rect(startx, starty, width, height, red, green, blue)**

Disegna un rettangolo vuoto che parte dal punto con coordinate (startx, starty) con una larghezza width e un'altezza height. Il colore è definito nelle sue componenti rosso, verde e blu.

I valori delle componenti possono variare da 0 a 255.

### **draw\_fill\_rect(startx, starty, endx, endy, red, green, blue)**

Disegna un rettangolo pieno che parte dal punto con coordinate (startx, starty) con una larghezza width e un'altezza height.

Il colore è definito nelle sue componenti rosso, verde e blu.

I valori delle componenti possono variare da 0 a 255.

### **get\_cpu\_temp()**

Questa funzione restituisce la temperatura misurata dal sensore interno microcontrollore RP2040.

### **is\_button\_pressed(button)**

Restituisce true/1 se il pulsante è premuto nell'istante in cui viene chiamata la funzione. I pulsanti che possono essere passati in ingresso alla funzione sono JOYSTICK\_UP, JOYSTICK\_DOWN, JOYSTICK\_LEFT, JOYSTICK\_RIGHT, JOYSTICK\_CENTER, BUTTON\_A, BUTTON\_B.

### **enable\_button\_irq\_state(enable)**

Abilita l'invio di un segnale di interrupt sul pin GPIO4 quando viene premuto un pulsante. Se vengono premuti più pulsanti solo il primo genererà l'invio del segnale. Per riattivare la funzione è necessario leggere prima il registro degli interrupt chiamando la funzione get\_interrupt

### **get\_button\_irq\_state(button)**

Quando un pulsante viene premuto, se abilitato, viene inviato un segnale sul pin GPIO4 della Sensei da parte dell'expander a cui sono connessi tutti i pulsanti. Chiamando questa funzione è possibile sapere se un pulsante specifico è stato premuto a partire dall'ultimo interrupt (incluso quello che ha generato il segnale). I pulsanti che possono essere passati in ingresso alla funzione sono JOYSTICK\_UP, JOYSTICK\_DOWN, JOYSTICK\_LEFT, JOYSTICK\_RIGHT, JOYSTICK\_CENTER, BUTTON\_A, BUTTON\_B.

### **get\_interrupt\_register()**

I pulsanti e il joystick sono connessi ad un expander. Quando un pulsante viene premuto, se abilitato, viene inviato un segnale sul pin GPIO4 della Sensei. Chiamando questa funzione è possibile sapere quali pulsanti sono stati premuti a partire dall'ultimo interrupt (incluso quello che ha generato il segnale).

### **play\_note(frequency, duration, volume, sweep\_direction, sweep\_time)**

Questa funzione consente di emettere un suono o un effetto sonoro utilizzando il buzzer. I parametri in ingresso sono la frequenza, la durata, il volume, la sweep direction, ovvero la direzione in cui la frequenza o il tono del suono cambia nel tempo e la sweep time, ovvero il tempo necessario affinché il tono del suono cambi da una frequenza all'altra, in altre parole il tempo necessario per completare la sweep.

### **get\_interrupt\_register()**

I pulsanti e il joystick sono connessi ad un expander. Quando un pulsante viene premuto, se abilitato, viene inviato un segnale sul pin GPIO4 della Sensei. Chiamando questa funzione è possibile sapere quali pulsanti sono stati premuti a partire dall'ultimo interrupt (incluso quello che ha generato il segnale).

### **read\_battery()**

Restituisce il livello di tensione della batteria. Se il cavo USB è connesso il valore restituito da questa funzione sarà sempre 4.2V, pari al massimo della carica della batteria.

### **read\_light()**

Restituisce il valore misurato dal sensore di luminosità

## **touch\_init()**

Attiva il sensore touch su 12 elettrodi (vedi pinout della scheda)

## **get\_touch()**

Restituisce una tupla di 12 elementi, corrispondenti allo stato degli elettrodi 0-11. Se il valore tupla[i] è pari a 1, l'elettrodo i-esimo corrispondente è stato toccato.

## **imu\_init()**

Attiva il sensore accelerometro/giroscopio.

## **get\_acceleration()**

Restituisce una tupla di 3 elementi, corrispondenti all'accelerazione sugli assi x, y e z. Il primo elemento, tupla[0] è l'asse x.

## **get\_rotation()**

Restituisce una tupla di 3 elementi, corrispondenti alla velocità angolare sugli assi x, y e z. Il primo elemento, tupla[0] è l'asse x.

## **enable\_pedometer()**

Abilita la funzione pedometro.

## **reset\_steps()**

Resetta il conteggio dei passi.

## **get\_steps()**

Restituisce il numero di passi rilevato.

## **get\_freefall()**

Restituisce "True" se il sensore ha rilevato una caduta libera, dopo la chiamata cambia il dato a "False".

## **get\_single\_tap()**

Restituisce "True" se il sensore ha rilevato un singolo tocco, dopo la chiamata cambia il dato a "False".

## **enable\_mic()**

funzione non ancora disponibile, servirà per abilitare il microfono in modalità rilevatore di rumore.

## **read\_mic()**

funzione non ancora disponibile, servirà per leggere l'intensità di suono/rumore captato dal microfono

## **enable\_usb\_mic()**

funzione non ancora disponibile, servirà per abilitare il microfono in modalità dispositivo-audio-usb

# 7. Esempi MicroPython

## 7.1 Display basics 1

```
display_basics_1.py
1 from melopero_sensei import *          A
2 from time import sleep
3
4
5 sensei = MeloperoSensei()              B
6
7 sensei.set_display_color(180,0,0)       C
8 sensei.update_display()
9
10 sleep(2)                             D
11
12 sensei.set_display_color(0,180,0)      E
13
14 sensei.set_text_color(0,0,255)
15 sensei.set_text_font(TEXT_MONO_BOLD, TEXT_BIG)
16
17 sensei.write_text("Hello", 50,130)     F
18
19 sensei.update_display()
20
21 sleep(2)
22
23 sensei.clear_display()               G
24 sensei.update_display()
```

- (A) dalla libreria melopero\_sensei importa tutte le funzioni  
dalla libreria time importa la funzione sleep
- (B) crea un nuovo oggetto Melopero Sensei e assegnagli il  
nome sensei
- (C) imposta il colore del display con R, G, B rispettivamente pari a 180, 0, 0  
(sarà un colore rosso). Applica le modifiche chiamando update\_display()
- (D) attendi 2 secondi con la funzione sleep. Senza questo comando il codice verrebbe eseguito troppo velocemente e non vedresti nulla
- (E) imposta un nuovo colore di sfondo, il colore del testo, il font e la dimensione.  
Le nuove modifiche non sono ancora visibili
- (F) scrivi il testo "hello" in posizione 50, 130.  
Applica tutte le modifiche precedenti chiamando update\_display()
- (G) cancella tutto il display e applica il comando utilizzando update\_display().

## 7.2 Display basics 2

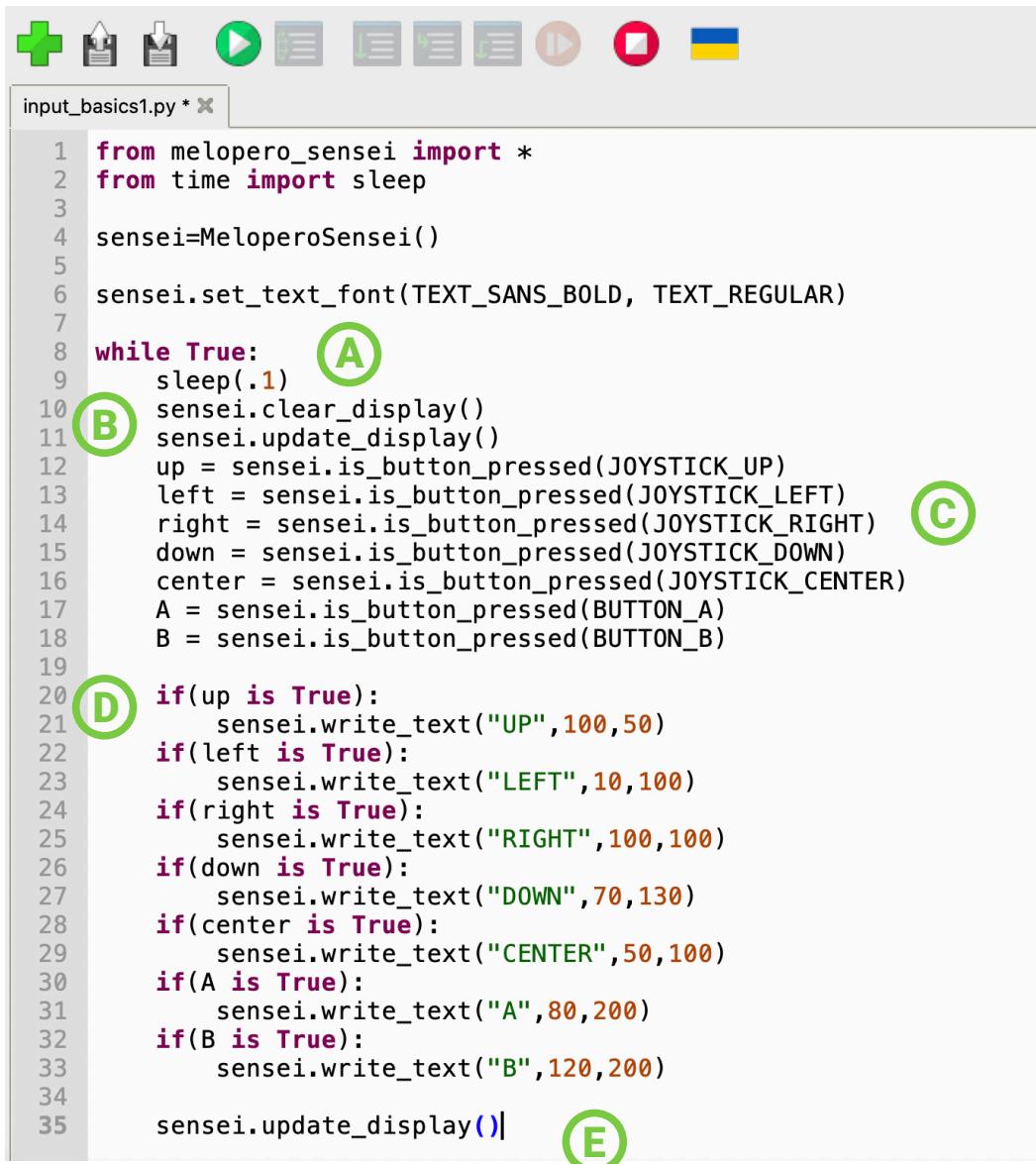


display\_basics\_2.py

```
1 from melopero_sensei import *
2
3
4 sensei=MeloperoSensei()
5
6
7 sensei.draw_line(10,10,60,60,200,0,0) (A)
8
9
10 sensei.draw_line(60,60,200,60,0,255,0) (B)
11
12
13 sensei.draw_pixel(100,100,255,255,0) (C)
14
15
16 sensei.draw_rect(120,120,100,30,200,0,255) (D)
17
18
19 sensei.draw_fill_rect(120,170,100,30,0,100,255) (E)
20
21
22 sensei.update_display() (F)
```

- (A) disegna una linea dal punto x=10, y=10 al punto x=60, y=60 (sarà diagonale) utilizzando il colore R=200, G=0, B=0 (un rosso scuro)
- (B) disegna una linea dal punto 60,60 al punto 200, 60 (sarà orizzontale) utilizzando il colore R=0, G=255, B=0 (un verde)
- (C) imposta il colore del pixel 120,120 con R, G, B rispettivamente pari a 300, 0, 255
- (D) disegna un rettangolo vuoto di larghezza 100 e altezza 30, partendo dal punto 120,120. Utilizza il colore R=0, G=100, B=255 per il contorno
- (E) disegna un rettangolo pieno di larghezza 100 e altezza 30, partendo dal punto 120,170. Utilizza il colore R=0, G=100, B=255 per il riempimento
- (F) Applica al display tutte le modifiche precedenti chiamando update\_display()

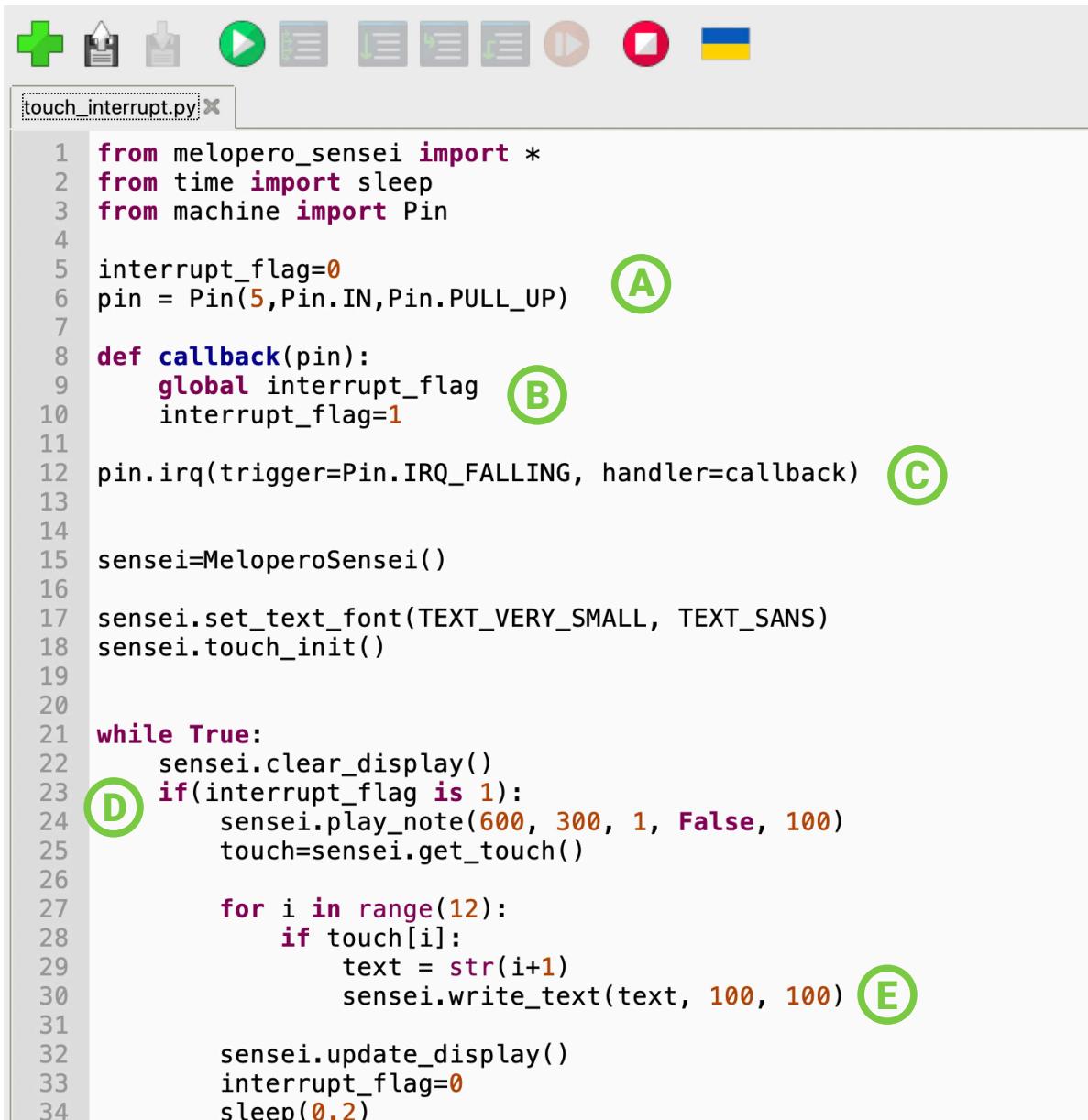
## 7.3 Input basics



```
1 from melopero_sensei import *
2 from time import sleep
3
4 sensei=MeloperoSensei()
5
6 sensei.set_text_font(TEXT_SANS_BOLD, TEXT_REGULAR)
7
8 while True: A
9     sleep(.1)
10    sensei.clear_display() B
11    sensei.update_display()
12    up = sensei.is_button_pressed(JOYSTICK_UP)
13    left = sensei.is_button_pressed(JOYSTICK_LEFT)
14    right = sensei.is_button_pressed(JOYSTICK_RIGHT)
15    down = sensei.is_button_pressed(JOYSTICK_DOWN)
16    center = sensei.is_button_pressed(JOYSTICK_CENTER)
17    A = sensei.is_button_pressed(BUTTON_A)
18    B = sensei.is_button_pressed(BUTTON_B)
19
20    if(up is True): D
21        sensei.write_text("UP",100,50)
22    if(left is True):
23        sensei.write_text("LEFT",10,100)
24    if(right is True):
25        sensei.write_text("RIGHT",100,100)
26    if(down is True):
27        sensei.write_text("DOWN",70,130)
28    if(center is True):
29        sensei.write_text("CENTER",50,100)
30    if(A is True):
31        sensei.write_text("A",80,200)
32    if(B is True):
33        sensei.write_text("B",120,200)
34
35    sensei.update_display() E
```

- (A)** Esegui il blocco di codice sottostante ciclicamente per sempre (ovvero fino a quando True è maggiore di zero)
- (B)** Attenzione all'indentazione, ovvero questa rientranza che fa capire a Thonny qual è il blocco di codice da eseguire sotto while True:
- (C)** leggi lo stato dei pulsanti e memorizza il valore (premuto = 1, non premuto =0) nelle rispettive variabili up, left, right, down, center, A e B.
- (D)** controlla ogni variabile, se è uguale a 1 il corrispondente tasto è premuto e quindi scrivi sul display il testo relativo nella posizione indicata.
- (E)** applica tutte le modifiche al display con update\_display()

## 7.4 Touch interrupt



The image shows a screenshot of a MicroPython code editor. At the top, there are several icons: a green plus sign, a file icon, a download icon, a play icon, a memory card icon, a left arrow, a right arrow, a red square, and a blue and yellow flag icon. Below the icons, the file name "touch\_interrupt.py" is displayed in a tab. The code itself is as follows:

```
1 from melopero_sensei import *
2 from time import sleep
3 from machine import Pin
4
5 interrupt_flag=0
6 pin = Pin(5,Pin.IN,Pin.PULL_UP) A
7
8 def callback(pin):
9     global interrupt_flag B
10    interrupt_flag=1
11
12 pin.irq(trigger=Pin.IRQ_FALLING, handler=callback) C
13
14
15 sensei=MeloperoSensei()
16
17 sensei.set_text_font(TEXT VERY_SMALL, TEXT_SANS)
18 sensei.touch_init()
19
20
21 while True:
22     sensei.clear_display()
23     if(interrupt_flag is 1): D
24         sensei.play_note(600, 300, 1, False, 100)
25         touch=sensei.get_touch()
26
27         for i in range(12):
28             if touch[i]:
29                 text = str(i+1)
30                 sensei.write_text(text, 100, 100) E
31
32             sensei.update_display()
33             interrupt_flag=0
34             sleep(0.2)
```

(A)

il sensore touch avviserà la Sensei di un nuovo touch sul pin 5  
definisci una variabile flag (bandiera) e impostala a 0

(B)

la callback è una funzione che viene chiamata ogni volta che la Sensei rileva un segnale di interrupt (una sorta di avviso) sul pin 5. In particolare questa funzione cambia il valore della variabile flag a 1.

(C)

abilita il controllo dei segnali di interrupt sul pin 5 e, quando ricevuti, esegui la funzione callback

(D)

se la flag è uguale ad 1, esegui il codice sottostante. Attenzione qui ci sono due livelli di indentazione, il secondo blocco, quello più rientrante, viene eseguito solo se la condizione if() è verificata.

(E)

emetti un suono, poi leggi e memorizza la lista contenente lo stato dei 12 elettrodi.  
**for i in range(12):**: facendo variare il valore di "i" da 0 a 11, controlla, uno alla volta, se l'elemento i-esimo touch[i] è pari a 1 (1=toccato, 0=non toccato). Se si, stampane il numero corrispondente sullo schermo.

**if touch[i]:** è un modo alternativo di scrivere **if touch[i] is 1:**

## 7.5 Read battery

The image shows the ScratchJr interface. At the top, there is a toolbar with various icons: a green plus sign, a pencil, a document, a play button, a list, a list with a checkmark, a list with a question mark, a red square, and a blue and yellow flag. Below the toolbar, the script editor window is open, showing a script titled "read\_battery.py". The script contains the following Python code:

```
1 from melopero_sensei import *
2 from time import sleep
3
4 sensei=MeloperoSensei()
5
6 sensei.set_text_font(TEXT_BIG, TEXT_SANS_BOLD)
7
8 while True:
9     sensei.clear_display()
10    bat = sensei.read_battery() (A)
11
12    sensei.write_text("BAT", 60,80) (B)
13    sensei.write_text(str(bat), 60,150)
14
15    sensei.update_display()
16    sleep(1) (C)
17
18
19
```

The code uses the Melopero Sensei library to read the battery level and display it on a screen. It includes a loop that updates the display every second.

**(A)**

la funzione `read_battery()` restituisce la tensione della batteria. Il valore massimo è 4.2V. Se si chiama questa funzione quando il cavo USB è connesso, il valore restituito sarà sempre 4.2 a causa della tensione di ricarica applicata dal charger. Per leggere la tensione della batteria bisogna disconnettere il cavo USB.

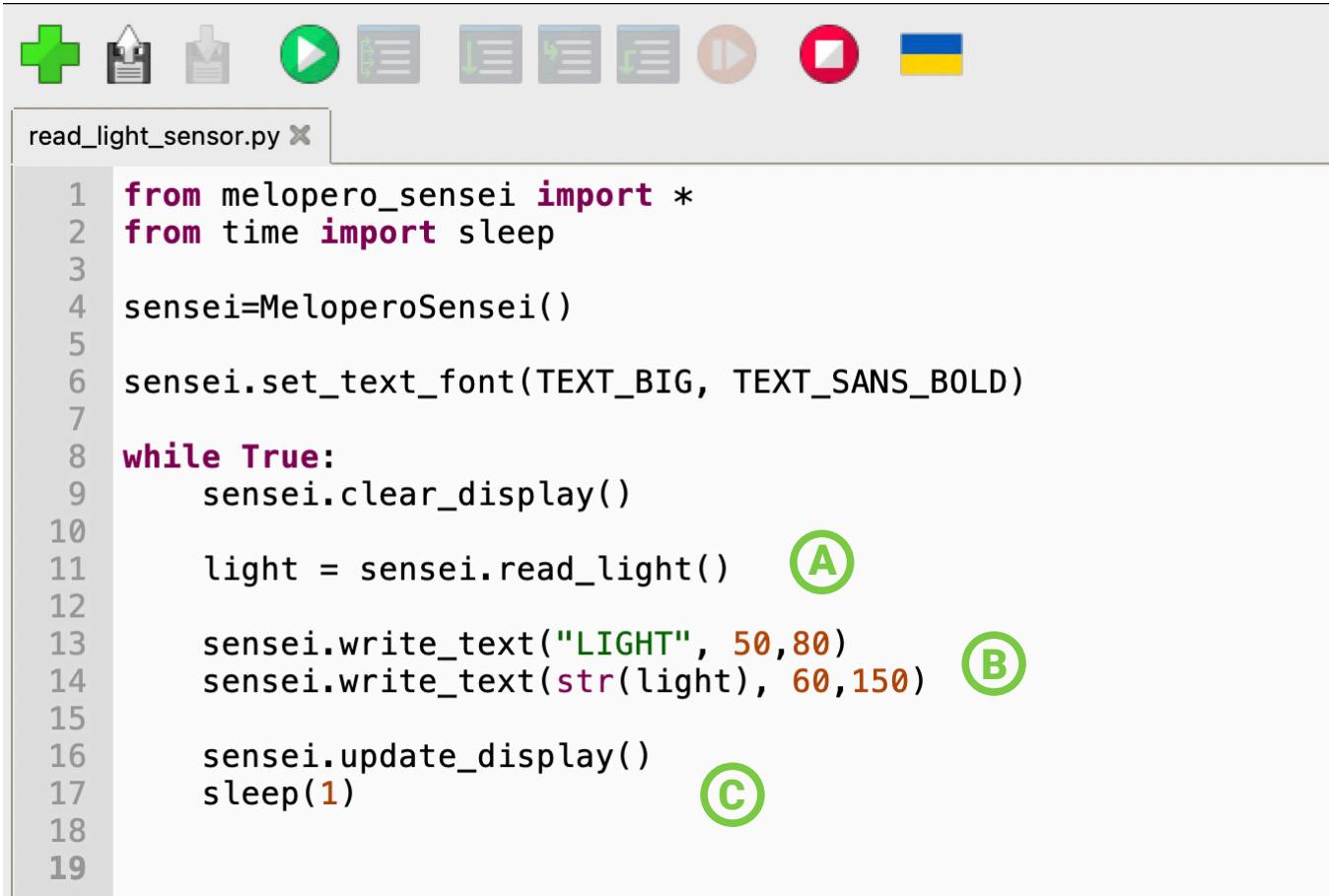
**(B)**

Scrivi sul display "BAT" in posizione 60, 80.  
Converti il valore `bat` in una stringa utilizzando la funzione `str()` e visualizzalo sul display nella posizione 60, 150.

**(C)**

Aggiorna il display per applicare le modifiche e attendi 1 secondo prima di eseguire un nuovo ciclo while.

## 7.6 Read light sensor



```
1 from melopero_sensei import *
2 from time import sleep
3
4 sensei=MeloperoSensei()
5
6 sensei.set_text_font(TEXT_BIG, TEXT_SANS_BOLD)
7
8 while True:
9     sensei.clear_display()
10
11     light = sensei.read_light() (A)
12
13     sensei.write_text("LIGHT", 50, 80)
14     sensei.write_text(str(light), 60, 150) (B)
15
16     sensei.update_display()
17     sleep(1) (C)
18
19
```

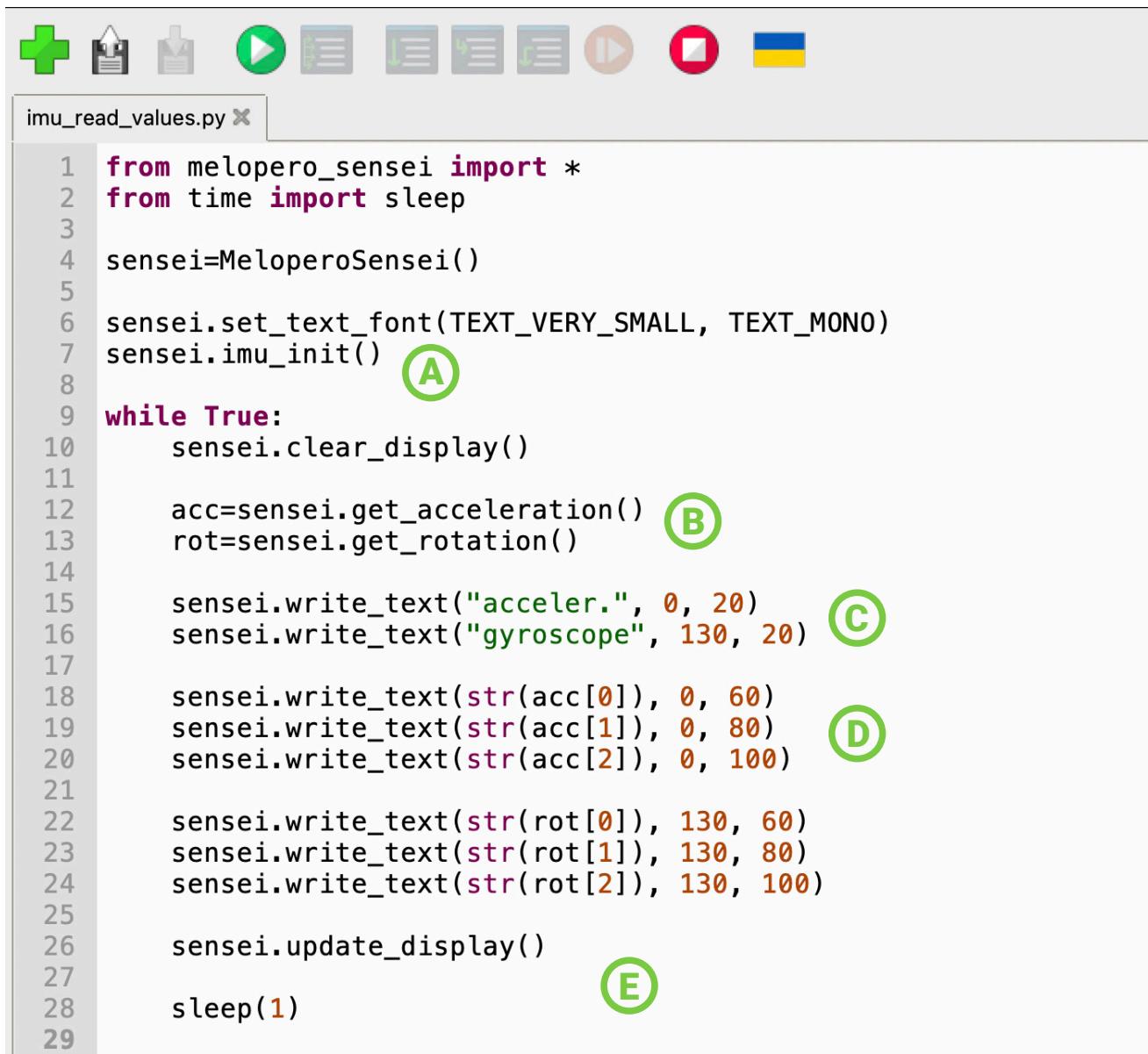
A la funzione `read_light()` restituisce un numero che varia a seconda della luminosità percepita dal sensore. Il massimo e il minimo sono quindi relativi all'ambiente in cui si esegue l'esperimento.

Ad esempio, prova a posizionare un dito sul sensore di luminosità e memorizza il valore riportato sul display. Dopodichè osserva e memorizza il valore riportato dal sensore senza che essere coperto. Adesso puoi utilizzare questi due valori per capire se la Sensei è al buio oppure no.

B Scrivi sul display "LIGHT" in posizione 50, 80.  
Converti il valore `light` in una stringa utilizzando la funzione `str()` e visualizzalo sul display nella posizione 60, 150.

C Aggiorna il display per applicare le modifiche e attendi 1 secondo prima di eseguire un nuovo ciclo `while`.

## 7.7 IMU read values



```
imu_read_values.py
1 from melopero_sensei import *
2 from time import sleep
3
4 sensei=MeloperoSensei()
5
6 sensei.set_text_font(TEXT VERY_SMALL, TEXT_MONO)
7 sensei.imu_init() A
8
9 while True:
10     sensei.clear_display()
11
12     acc=sensei.get_acceleration() B
13     rot=sensei.get_rotation()
14
15     sensei.write_text("acceler.", 0, 20) C
16     sensei.write_text("gyroscope", 130, 20)
17
18     sensei.write_text(str(acc[0]), 0, 60)
19     sensei.write_text(str(acc[1]), 0, 80)
20     sensei.write_text(str(acc[2]), 0, 100) D
21
22     sensei.write_text(str(rot[0]), 130, 60)
23     sensei.write_text(str(rot[1]), 130, 80)
24     sensei.write_text(str(rot[2]), 130, 100)
25
26     sensei.update_display()
27
28     sleep(1) E
```

**(A)** attiva il sensore IMU (Intertial Measurement Unit), composto da accelerometro e giroscopio

**(B)** leggi i valori dell'accelerometro e del giroscopio.

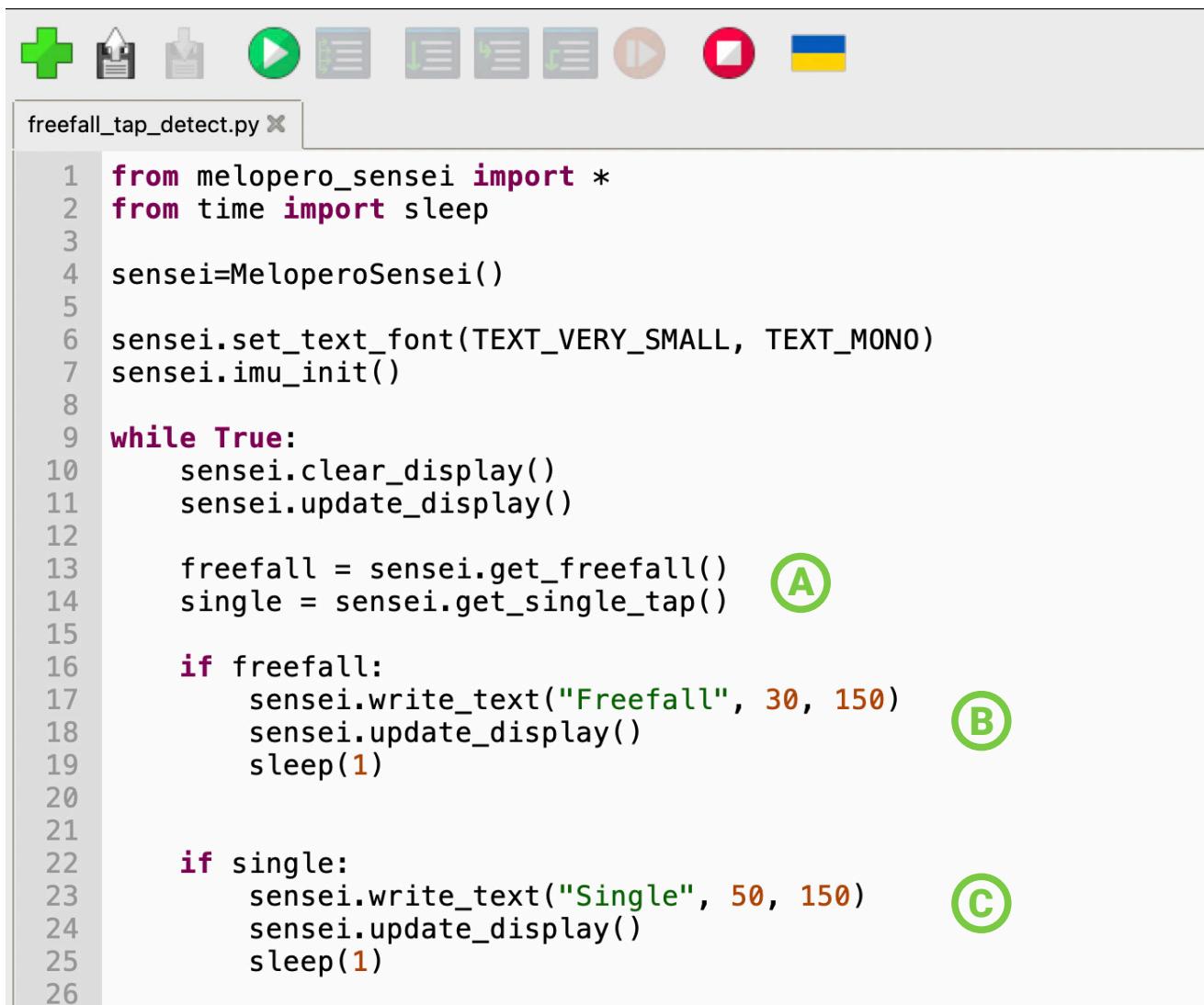
Ognuna di queste funzioni restituisce una tupla di 3 elementi, contenente i valori misurati sull'asse x, y e z. Per accedere al primo elemento, ovvero il valore misurato sull'asse x, si può scrivere acc[0] e rot[0].

**(C)** scrivi sul display i nomi dei due sensori, uno a sinistra e l'altro a destra.

**(D)** scrivi sullo schermo i valori rilevati da entrambi i sensori, in modo che non si sovrappongano. La funzione write\_text richiede come primo parametro una stringa. Utilizzando str(acc[0]) si può convertire il numero acc[0] in stringa a passarlo poi a write\_text(...)

**(E)** applica le modifiche al display chiamando update\_display()

## 7.8 Freefall and tap detection



```
1 from melopero_sensei import *
2 from time import sleep
3
4 sensei=MeloperoSensei()
5
6 sensei.set_text_font(TEXT VERY_SMALL, TEXT_MONO)
7 sensei.imu_init()
8
9 while True:
10     sensei.clear_display()
11     sensei.update_display()
12
13     freefall = sensei.get_freefall() A
14     single = sensei.get_single_tap()
15
16     if freefall:
17         sensei.write_text("Freefall", 30, 150) B
18         sensei.update_display()
19         sleep(1)
20
21
22     if single:
23         sensei.write_text("Single", 50, 150) C
24         sensei.update_display()
25         sleep(1)
26
```

A

La IMU è in grado di rilevare la caduta libera e un colpetto dato sulla Sensei, come si fa su alcuni smartphone per accendere lo schermo e leggere l'ora.

Per simulare la caduta libera puoi lasciarti cadere sul palmo della mano la Sensei, anche se attaccata al cavo USB e da un piccola distanza.

Se il sensore rileva una caduta o un tap lo registra in memoria.

Le funzioni `get_freefall()` e `get_single_tap()` restituiscono "True" se il relativo evento è stato rilevato, False se non è stato rilevato.

Ad ogni chiamata di queste funzioni il dato memorizza dal sensore viene resettato a False.

B

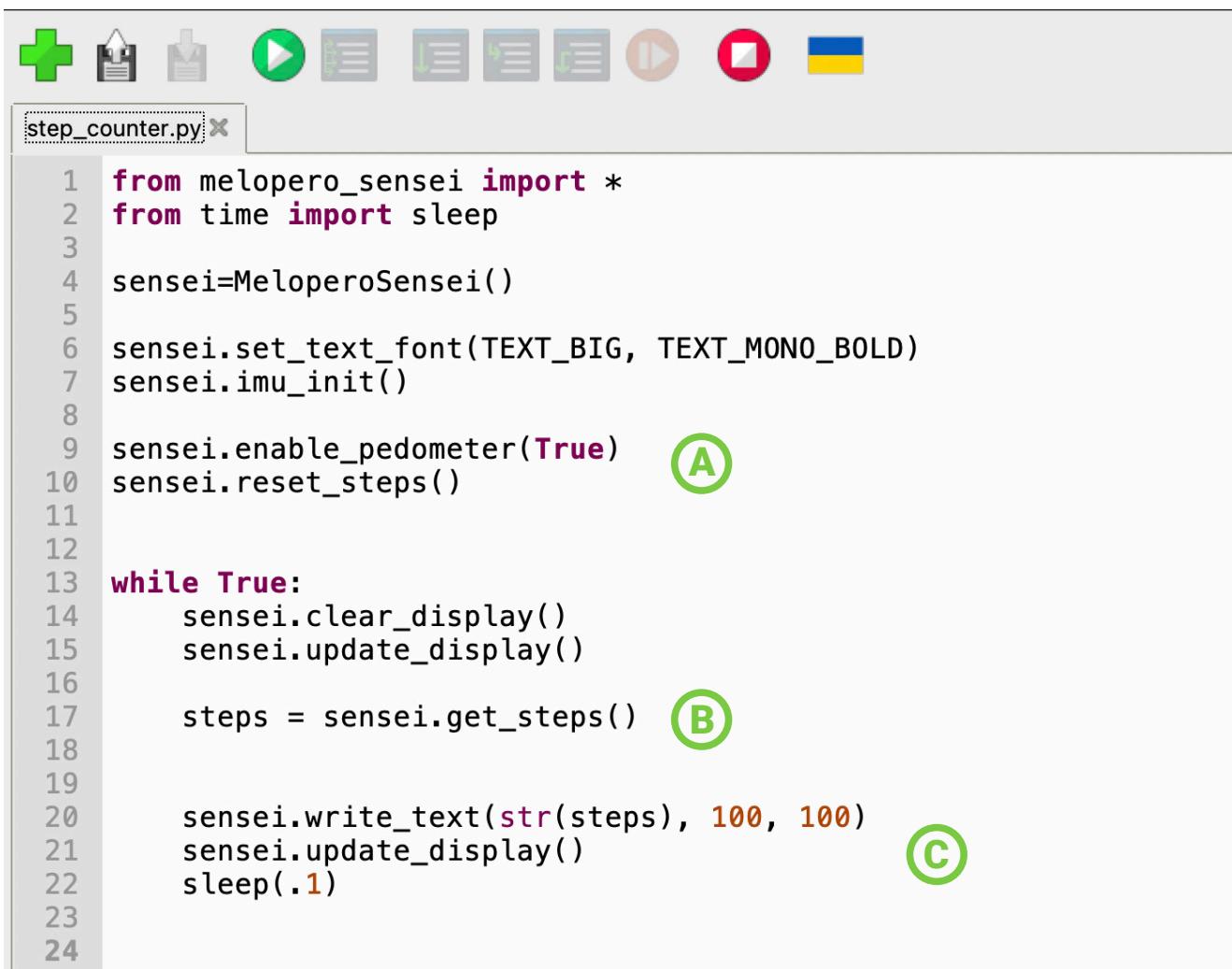
`if freefall:` è un'abbreviazione di `if freefall is True:`

se la variabile `freefall` è "True", scrivi "Freefall" sullo schermo, aggiorna il display e attendi 1 secondo.

C

se la variabile `single` è "True", scrivi "Single" sullo schermo, aggiorna il display e attendi 1 secondo.

## 7.9 Step counter



```
1 from melopero_sensei import *
2 from time import sleep
3
4 sensei=MeloperoSensei()
5
6 sensei.set_text_font(TEXT_BIG, TEXT_MONO_BOLD)
7 sensei.imu_init()
8
9 sensei.enable_pedometer(True)      A
10 sensei.reset_steps()
11
12
13 while True:
14     sensei.clear_display()
15     sensei.update_display()
16
17     steps = sensei.get_steps()      B
18
19
20     sensei.write_text(str(steps), 100, 100)
21     sensei.update_display()
22     sleep(.1)                     C
23
24
```

(A)

La IMU è in grado di rilevare e contare i passi.

Per utilizzare il sensore come contapassi bisogna abilitare la funzionalità chiamando `enable_pedometer()` e cancellare l'eventuale memoria di passi precedenti con `reset_steps()`.

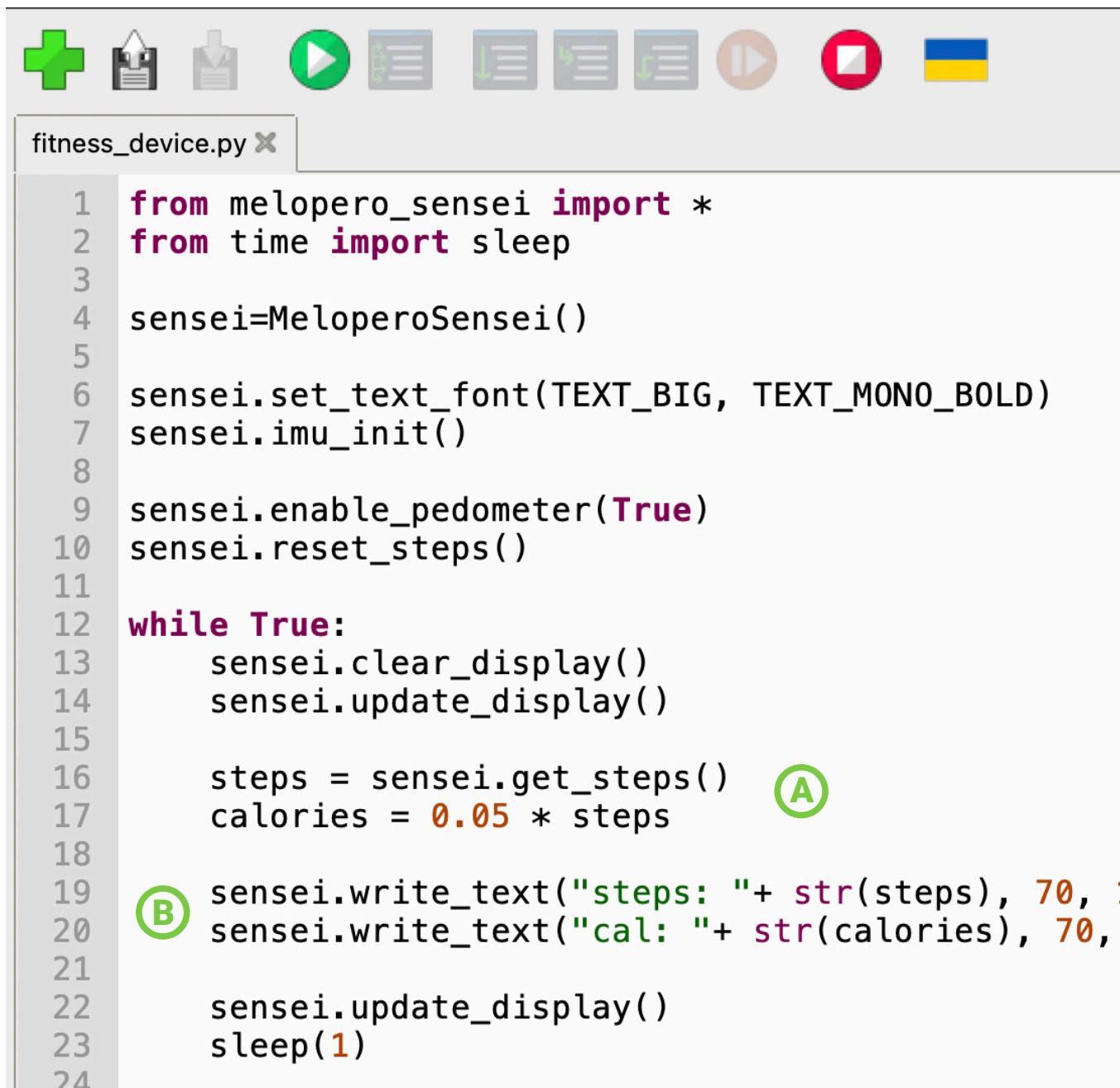
(B)

`get_steps()` restituisce il numero di passi effettuati.

(C)

converti il numero di passi in testo con `str(steps)` e scrivi sullo schermo nella posizione 100,100.

## 7.10 Fitness device



```
fitness_device.py ✘

1 from melopero_sensei import *
2 from time import sleep
3
4 sensei=MeloperoSensei()
5
6 sensei.set_text_font(TEXT_BIG, TEXT_MONO_BOLD)
7 sensei imu_init()
8
9 sensei.enable_pedometer(True)
10 sensei.reset_steps()
11
12 while True:
13     sensei.clear_display()
14     sensei.update_display()
15
16     steps = sensei.get_steps()      A
17     calories = 0.05 * steps
18
19     B sensei.write_text("steps: "+ str(steps), 70, :
20     sensei.write_text("cal: "+ str(calories), 70,
21
22     sensei.update_display()
23     sleep(1)
24
```

**A** grazie al conteggio dei passi, è possibile stimare il numero di calorie consumate e realizzare così un vero e proprio dispositivo fitness indossabile. In questo esempio la formula utilizzata è molto approssimativa, semplicemente aggiunge 0,05 calorie per ogni passo effettuato e non tiene in considerazione il peso, l'età o altri fattori che consentirebbero sicuramente di giungere a calcoli più precisi.

**B** scrivi sullo schermo i passi e le calorie.  
nella riga di codice **"steps: " + str(steps)**, il simbolo "+" consente di concatenare la stringa "steps: " e quella generata dalla conversione str(step).

# 8. Reset della Flash Memory

Se hai bisogno di ripristinare la memoria flash alle condizioni iniziali, attiva la modalità bootloader sulla Sensei e trascina e rilascia (o copia e incolla) il file di cancellazione flash disponibile nel pacchetto software sull'unità RPI-RP2, come hai fatto per installare MicroPython.

Per attivare la modalità bootloader, quando la Sensei è già collegata alla porta USB del tuo computer, premi e tieni premuto il pulsante BOOT / BT, quindi premi e rilascia il pulsante di RESET / RT . Continua a tenere premuto il pulsante BOOT / BOOTSEL finché compare l'unità RPI-RP2.

