



# MicroPython - Entrées, Sorties (GPIO)



[Mise à jour le : 1/6/2023] **En cours de rédaction**

- **Ressources**

- [MicroPython.org](https://micropython.org/)
- [MicroPython documentation](#)
- [IDE Thonny](#)

- **Lectures connexes**

- [MicroPython - Les modules Raspberry Pi Pico et Pico W](#)
- [ESP32/ESP8266 Digital Inputs and Digital Outputs with MicroPython](#)
- [MicroPython with ESP32 and ESP8266: Interacting with GPIOs](#)
- [ESP32/ESP8266 PWM with MicroPython - Dim LED](#)
- [ESP32/ESP8266 Analog Readings with MicroPython](#)
- [MicroPython: Interrupts with ESP32 and ESP8266](#)

## 2. Entrées, sorties numériques



### 2.3 Interruption

- **Matériel** : [ESP32 Feather Huzzah](#) ou [Raspberry Pi Pico](#), [Digilent Pmod BTN: 4 User Pushbuttons](#) [\[Schéma\]](#)

#### 2.3.2 Configuration

La configuration en entrée de la broche destinée à recevoir un évènement est identique à celle du paragraphe précédent.

### 2.3.3 Evènement et gestionnaire d'évènement

Un évènement est attaché à un gestionnaire (service d'interruption) .



#### 2.3.4 Exemples de code

- [RPi Pico](#)
- [ESP32](#)

#### A faire

Exemple de code pour un **Raspberry Pi Pico**

\*.py

- **Ressource** : [Quick reference for ESP32, GPIO Pins, External interrupts](#) sur Micropython.org.

Exemple de code pour un **ESP32 Feather Huzzah**

\*.py

```
# Code partiel du programme HORLOGE

# Réglage de l'heure à la mise sous tension
time_offset=12*3600+0*60+0 # hh+mm+ss

# Routines de service d'interruption (ISR)
def handle_interrupt_min(pin):
    global time_offset
    time_offset+=60
    time.sleep(.2)

def handle_interrupt_hr(pin):
    global time_offset
    time_offset+=3600
    time.sleep(.2)

# Réglage des minutes
# Ajout de 60s à l'heure initiale
```

```
button_min = Pin(25, Pin.IN)
# Gestionnaire d'interruption
button_min.irq(trigger=Pin.IRQ_RISING, handler=handle_interrupt_min)

# Réglage des heures
# Ajout de 3600s à l'heure initiale
button_hr = Pin(26, Pin.IN)
# Gestionnaire d'interruption
button_hr.irq(trigger=Pin.IRQ_RISING, handler=handle_interrupt_hr)
{{ :python:micropython:matériel:thonny.png?nolink&70| }}
```

**Télécharger** le projet MICROPYTHON\_ESP32\_HORLOGE pour Thonny.



A voir : la vidéo de démonstration sur [Youtube](#)



### 3. Entrées analogiques

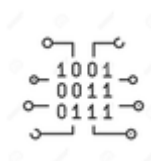
- **Ressource**

- [Quick reference for the RP2, Pins and GPIO](#) sur Micropython.org., potentiomètre 10kOhm.

#### 3.1 Présentation

- **Ressource**

- [Un signal analogique : petits rappels](#) sur le site Zeste de savoir.



#### 3.2 Exemples de code

- [RPI Pico](#)
- [ESP32](#)
- **Ressource**
  - [ADC \(analog to digital conversion\)](#) | [class Pin – control I/O pins](#) sur Micropython.org.

Exemple de code pour un **Raspberry Pi Pico**

\*.py

```
# -----  
-----  
# Lecture et affichage dans la console de la tension issue d'un  
# potentiomètre  
# Date : 22/5/2023  
# Matériels : Raspberry Pi Pico, Shield Grove, pot. 10k  
# ADC accessibles sur le shield Grove pour RP2 :  
# Connecteur: ADC      : GPIO  
#      A0 : ADC0      : 26  
#      A1 : ADC0,ADC1: 26,27  
#      A2 : ADC1,ADC2: 27,28  
# IDE : Thonny  
# -----  
-----  
from machine import ADC, Pin  
import time  
  
# Le potentiomètre 10k0hm est connecté à l'entrée analogique A0 du  
# shield.  
# Attention : La tension doit être comprise entre 0 - 3,3V (3,6V max !)  
# sur une entrée analogique.  
# Configuration  
pot = ADC(Pin(26))  
  
while (True):  
    val=pot.read_u16() # lecture de l'ADC  
    U = val*3.3/65535 # Calcul de la tension  
    print("%.2f" % U) # Affichage dans la console (formaté à 2  
    décimales)  
    time.sleep(1)
```

- **Ressource**
  - [ADC \(analog to digital conversion\)](#) sur Micropython.org.

Exemple de code pour un **ESP32 Feather Huzzah**

\*.py

```
# ADC accessibles en Python sur la carte ESP32 Feather Huzzah :
```

```
# ADC:GPIO
# A2 : 34
# A3 : 39
# A4 : 36
# A7 : 32
# A9 : 33

from machine import ADC, Pin

# Le potentiomètre 10k0hm est connecté à l'entrée analogique A2 de
# l'ESP32.
# Configuration
adc = ADC(Pin(34))
# Sur une entrée analogique, la tension doit
# être comprise entre 0 - 3,3V (3,6V max !)
adc.atten(ADC.ATTN_11DB) # voir doc
# Mesure
value = adc.read()

print(value) # affichage dans la console
```

From:

<http://webge.fr/dokuwiki/> - **WEBGE Wikis**

Permanent link:

<http://webge.fr/dokuwiki/doku.php?id=python:micropython:es&rev=1692468135>

Last update: **2023/08/19 20:02**

