



# MicroPython - Afficheurs à circuit SSD1306



[Mise à jour le : 28/4/2023]

- **Ressources**

- [Getting started with MicroPython on the ESP32, RP2, etc.](#)
- [MicroPython.org](#)
- [MicroPython documentation](#)
- [IDE Thonny](#)

- **Lectures connexes**

- **Installer MicroPython** - [MicroPython - Les modules Espressif ESP32 et ESP8266](#)
- **Raspberry Pi Pico** - [Les modules Raspberry Pi Pico et Pico W](#)
- **Programmez !** Juillet/Août 2019
- **Elektor 489** Mai/Juin 2021

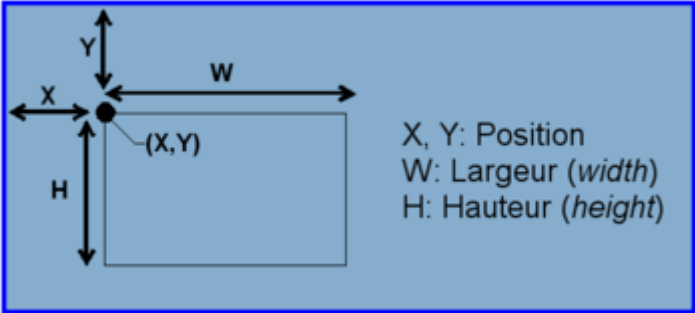


## 1. Présentation

Les exemples de code de cette page ont été testés sur un afficheur [Gravity OLED-2864 \(128x64\)](#) de [DFROBOT](#) et un [Module afficheur OLED 0,96" TF052](#) connectés à un **ESP32** ou à un **Raspberry Pi Pico**. Ces afficheurs sont pilotés par un circuit SOLOMON SYSTECH [SSD1306](#). Sa mise en oeuvre nécessite une bibliothèque (téléchargeable à partir du lien ci-dessous).

**Télécharger** la bibliothèque SSD1306 pour MicroPython et la copier dans un fichier nommé `ssd1306.py` à installer dans la carte à  $\mu$ C. Cette bibliothèque hérite de [framebuf](#). On dispose donc des méthodes ci-dessous pour dessiner des formes sur le buffer d'impression.

• Organisation de l'écran



- **x** : position du point par rapport au côté gauche de l'écran.
- **y** : position du point par rapport au dessus de l'écran.
- **w** : largeur (du mot Width).
- **h** : hauteur (du mot Height).
- **c** : couleur (1=point allumé, 0=point éteint)

• Méthodes de la bibliothèque **SSD1306** (hérite de framebuf)

Prototype	Description
<b>fill</b> ( c )	Remplit l'écran en noir (c=1) ou en blanc (c=0)
<b>rect</b> ( x , y , w , h , c[, f])	Dessine un rectangle de largeur w et de hauteur h dans la couleur c au point (x,y). Le paramètre facultatif f peut être défini sur True pour remplir le rectangle. Sinon, seul un contour d'un pixel est dessiné.
<b>hline</b> (x,y,w,c)	Dessine une ligne horizontale de longueur w dans la couleur c au point (x,y)
<b>vline</b> (x,y,h,c)	Dessine une ligne verticale de longueur h dans la couleur c au point (x,y)
<b>line</b> (x1,y1,x2,y2,c)	Dessine une ligne dans la couleur c entre les points (x1,y1) et (x2,y2)
<b>pixels</b> ( x , y[, c])	Affiche un pixel dans la couleur c au point (x,y). Si c n'est pas donné, obtient la valeur de couleur du pixel spécifié. Si c est donné, définit le pixel spécifié sur la couleur donnée.
<b>scroll</b> (depl.horiz, depl. vert.)	Déplace le contenu de l'écran de n points
<b>show</b> ()	Transfert le contenu du buffer d'affichage sur l'écran
<b>text</b> ("text",x,y,c)	Affiche le texte "text" dans la couleur c au point (x,y). En noir (c=1) ou en blanc (c=0). La hauteur d'un caractère occupe 8px.
<b>poly</b> ( x , y , coordonnées , c[, f])	Étant donné une liste de coordonnées, dessine un polygone fermé arbitraire (convexe ou concave) à l'emplacement x, y donné en utilisant la couleur donnée. Les coordonnées doivent être spécifiées sous forme de tableau d'entiers, par exemple .array('h', [x0, y0, x1, y1, ... xn, yn]) Le paramètre facultatif f peut être défini sur True pour remplir le polygone. Sinon, seul un contour d'un pixel est dessiné.
<b>ellipse</b> (ellipse ( x , y , xr , an , c[, f , m])	Dessine une ellipse à l'emplacement donné. Les rayons xr et yr définissent la géométrie ; des valeurs égales entraînent le dessin d'un cercle. Le paramètre c définit la couleur. Le paramètre optionnel f peut être défini sur True pour remplir l'ellipse sinon, seul un contour d'un pixel est dessiné.

2. Programmation

A l'exception de `scroll()`, les méthodes ci-dessus "écrivent" dans le tampon d'affichage.

## 2.1 Configurations

Dans les exemples de cette page, les microcontrôleurs accèdent à l'afficheur via le bus I2C. Il faut au préalable le configurer et créer une instance de la classe SSD1306 comme ci-dessous.

Exemple pour un **ESP32 Feather Huzzah**

`main.py`

```
from machine import Pin, SoftI2C
import ssd1306
import time
import urandom

# Configuration du bus i2c sur l'ESP32 Feather Huzzah
i2c = SoftI2C(scl=Pin(22), sda=Pin(23), freq=100000)

# Dimension de l'afficheur oled (ssd1306)
oled_width = 128 # px
oled_height = 64 # px

# Construction de l'objet oled
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)
```

Exemple pour un **Raspberry Pi Pico**

`main.py`

```
# Modifications à apporter au code ci-dessus
from machine import Pin, I2C
...

# Configuration du bus i2c sur le Raspberry Pi Pico
# Accès au connecteur I2C0 du shield Grove
i2c = I2C(0, sda=Pin(8), scl=Pin(9), freq=400_000)
# Accès au connecteur I2C1 du shield Grove
# i2c = I2C(1, sda=Pin(6), scl=Pin(7), freq=400_000)
...
```

Une fois créée, l'instance de la classe SSD1306 est utilisable pour afficher du texte et des graphiques

sur l'écran.

## 2.2 Démonos

- **Démono 1** : affichage d'un texte

[main.py](#)

```
# A ajouter au code du §2.1
# Division de l'afficheur en 8 lignes et 9 colonnes
lin_hight = 9
col_width = 8
def text_write(text, lin, col):
    oled.text(text,col*col_width,lin*lin_hight)

oled.fill(0) # Extinction de l'afficheur
text_write("MicroPython",1,2);
text_write("sur",3,6)
text_write("ESP32",5,5)
oled.show()

time.sleep(1)
```

- **Démono 2** : affichage de lignes

[main.py](#)

```
# A ajouter au code du §2.1
# ligne horizontale : hline(pos_x1,pos_y1,pos_x2,pos_y2,c)
# ligne verticale : vline(pos_x1,pos_y1,pos_x2,pos_y2,c)
# c=0=>noir, c=1=>blanc

oled.hline(0,0,oled_width-1,1)
oled.hline(0,oled_height-1,oled_width-1,1)
oled.vline(0,0,oled_height,1)
oled.vline(oled_width-1,0,oled_height,1)
oled.show()

time.sleep(3)
```

- **Démono 3** : affichage aléatoire de pixels

[main.py](#)

```
# A ajouter au code du §2.1
oled.fill(0) # Ecran noir
```

```

for n in range(50):
    pos_x = urandom.randint(1,oled_width)
    pos_y = urandom.randint(1,oled_height)
    oled.pixel(pos_x,pos_y,1)
oled.show()

time.sleep(3)

```

- **Démo 4** : affichage d'une icône

main.py

```

# A ajouter au code du §2.1
ICON = [
    [0,0,0,0,1,1,1,0,0,0,0],
    [0,0,0,1,0,0,0,1,0,0,0],
    [0,0,1,0,0,0,0,0,1,0,0],
    [0,1,0,0,0,0,0,0,0,1,0],
    [1,0,0,0,0,0,0,0,0,0,1],
    [1,1,1,1,1,1,1,1,1,1,1],
    [0,0,0,0,1,1,1,0,0,0,0],
    [0,0,0,1,0,1,0,1,0,0,0],
    [0,0,1,0,0,1,0,0,1,0,0],
    [0,1,0,0,0,1,0,0,0,1,0],
    [0,0,0,0,1,1,1,0,0,0,0],
]

# Démo 4a : affichage d'une icône au centre
oled.fill(0) # Extinction de l'afficheur
for y in range (11):
    for x in range (11):
        oled.pixel(x+64,y+32,ICON[y][x])
oled.show()

time.sleep(2)

# Démo 4b : affichage d'icônes au hazard
for n in range(12):
    pos_x = urandom.randint(1,oled_width-12)
    pos_y = urandom.randint(1,oled_height-12)
    for y, ligne in enumerate(ICON):
        for x, c in enumerate(ligne):
            oled.pixel(x+pos_x,y+pos_y,c)
oled.show()

time.sleep(3)

```



**Télécharger** le projet MICROPYTHON\_ESP32\_SSD1306\_DEMO pour Thonny et la **vidéo** des démos.

- **Démo 5** : affichage d'une icône avec canal alpha  
Modifier l'exemple du site [MCHobby](#) en prenant en compte le code du §2.1 pour un ESP32 ou un Raspberry Pi Pico.

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

Permanent link:  
<https://webge.fr/dokuwiki/doku.php?id=python:micropython:materiel:ssd1306&rev=1684509098>

Last update: **2023/05/19 17:11**

