0,96" 128x64 OLED 2864 Display module - SSD1306 (I2C)

[Mise à jour le 18/8/2023]

- **Ressources**
  - **Wiki DFRobot** : Gravity: I2C OLED-2864 Display
  - **Distribué** par Mouser
- **Lectures connexes**
  - Les afficheurs graphiques
  - Bibliothèque - Adafruit GFX Graphics Library
  - Adafruit 1,3" 128x64 OLED FeatherWing - SH1107 + 3 buttons (I2C)
  - Adafruit 1.8" 128x160 Color TFT LCD display with MicroSD Card v2 - ST7735R (SPI)

---

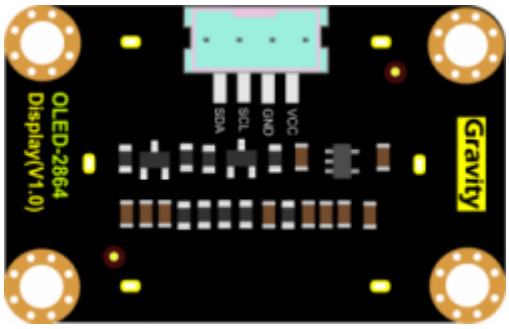
1. Description

Un écran **OLED** fonctionne sans rétroéclairage. Ainsi, il peut afficher des niveaux de **noir profond** et peut être plus mince et plus léger qu'un écran à cristaux liquides (LCD). Dans des conditions de **faible luminosité ambiante**, telles qu'une pièce sombre, un écran OLED peut obtenir un taux de contraste plus élevé qu'un écran LCD.

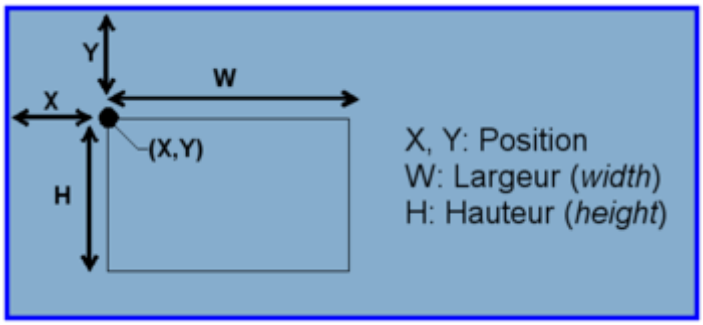
La technologie OLED est utilisée dans des applications commerciales telles que les écrans pour téléphones mobiles et lecteurs multimédias portables, les autoradios et les appareils photo numériques, entre autres.

L'écran **Gravity OLED 2864** est un module d'affichage autolumineux à **fond bleu**. La zone d'affichage est de **0,96“** et utilise une puce SSD1306. Il prend en charge les communications **I2C** et les fréquences de rafraîchissement allant jusqu'à 60 Hz. Le module utilise l’interface commune Gravity I2C pour une utilisation plug and play simplifiée. DFRobot

- **Caractéristiques**
  - **Diagonale** : 0,96”
  - **Luminosité** : 60 (typ.) Cd / m^2
  - **Contrôleur** : SSD1306
  - **Résolution** : 128 x 64
  - **Connectique** : 4 broches (alimentation et bus I2C)
  - **Bus I2C** : adresse 0x3C
  - **Tension d'alimentation** : 3,3V ~ 5V
  - **Consommation maximale** : 20mA @ 3v
  - **Dimensions** : 41.2×26.2x8mm
- **Brochage**



- **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)

2. Programmation

Les programmes suivant sont codés :

- En langage **C** sur Arduino Uno ou compatible
- En langage **MicroPython** sur Raspberry Pi Pico, ESP32 etc.

- Arduino
- Micropython

2.1 La bibliothèque Adafruit GFX Graphics

> **OakOLED**
>
> En programmation C, C++ l'utilisation des méthodes de la classe **Adafruit_GFX** sur cet afficheur peut se faire par l'intermédiaire de la bibliothèque **OakOLED**. Comme OakOLED dérive de Adafruit GFX Graphics il suffit de créer un objet OakOLED pour accéder aux méthodes de Adafruit_GFX.

1. **Installer** OakOLED avec le gestionnaire de bibliothèques.
2. **Tester l'exemple “Hello World”** ci-dessous.

helloWorld.cpp

```
// Exemple OakOLED
// Affiche "hello, world"

#include "Wire.h"
#include "Adafruit_GFX.h"
#include "OakOLED.h"

OakOLED oled;

void setup() {
  Serial.begin(115200);
  oled.begin();

  oled.setTextSize(1);
  oled.setTextColor(1);
  oled.setCursor(0, 0);

  oled.println("Hello, World!");
  oled.display();
}

void loop() {
  delay(10);
}
```

3. **Utiliser la bibliothèque OakOLED dans un programme** (IDE Visual Studio Code)
4. **Inclure** la bibliothèque **Wire.h** au programme en entrant le code ci-dessous.

*.cpp

```
\\ \\
```

```
#include <Wire.h>
```

5. **Inclure** les bibliothèques **Adafruit_GFX** et **OakOLED** dans le programme par :F1 → Arduino:Library Manager → Adafruit_GFX (OakOLED) → Include Library.

*.cpp

```
#include <Adafruit_GFX.h> // Résultat attendu
#include <OakOLED.h>
```

6. **Créer l'objet oled**
Entrer le code ci-dessous.

*.cpp

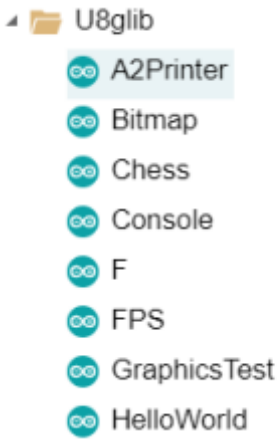
```
OakOLED oled;
```

7. **Utiliser** les méthodes de la classe Adafruit_GFX

2.2 La bibliothèque U8glib

1. **Installer** Arduino U8glib ou U8g2 si vous utilisez Arduino / Genuino 101 avec le gestionnaire de bibliothèques. Ces bibliothèques graphiques prennent en charge un grand nombre d'afficheurs monochromes.

2. **Tester l'exemple “Hello World”**
Arduino Examples → Example from Custom Libraries → U8glib → HelloWorld



> Dans la liste placée entre commentaires, sélectionner le constructeur **u8g** comme ci-dessous.

*.cpp

```
//U8GLIB_SSD1306_128X64 u8g(10, 9);// HW SPI Com: CS = 10, A0 = 9
```

```
(Hardware Pins are  SCK = 13 and MOSI = 11)
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);  // I2C
/ TWI
//U8GLIB_SSD1306_128X64
u8g(U8G_I2C_OPT_DEV_0|U8G_I2C_OPT_NO_ACK|U8G_I2C_OPT_FAST);// Fast I2C
/ TWI
```

3. **Utiliser la bibliothèque U8glib dans un programme** (IDE Visual Studio Code)

- **Inclure** la bibliothèque **u8glib** au programme par F1 → Arduino:Library Manager → U8glib → Include Library

*.cpp

```
#include <U8glib.h> // Résultat attendu
```

- **Initialiser l'objet u8g**

Entrer le code ci-dessous.

*.cpp

```
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);  // I2C
/ TWI
```

- **Utiliser les méthodes de la classe U8GLIB**

Une page lui sera bientôt consacrée.

Exemple

Hello.cpp

```
#include <U8glib.h>

U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);  // I2C
/ TWI

void setup(void){
u8g.drawStr( 0, 22, "hello World!");
}

void loop(void){

}
```