utilisation d'écrans dans les projets Raspberry Pi

Extrait : écrans à diodes électroluminescentes organiques (OLED)

Dogan Ibrahim (Royaume-Uni)

Cet article est un extrait du livre (en anglais) de Dogan Ibrahim intitulé *Using Displays in Raspberry Pi Projects* et publié par Elektor. Depuis leur apparition, les écrans OLED ont toujours eu les faveurs des makers et des ingénieurs spécialistes des systèmes embarqués. Dans cet article, nous verrons comment incorporer des écrans OLED dans des projets avec un minimum d'efforts et de programmation – des exigences qui, disons-le, font clairement du Raspberry Pi le candidat idéal. C'est ce que nous vous proposons de découvrir !

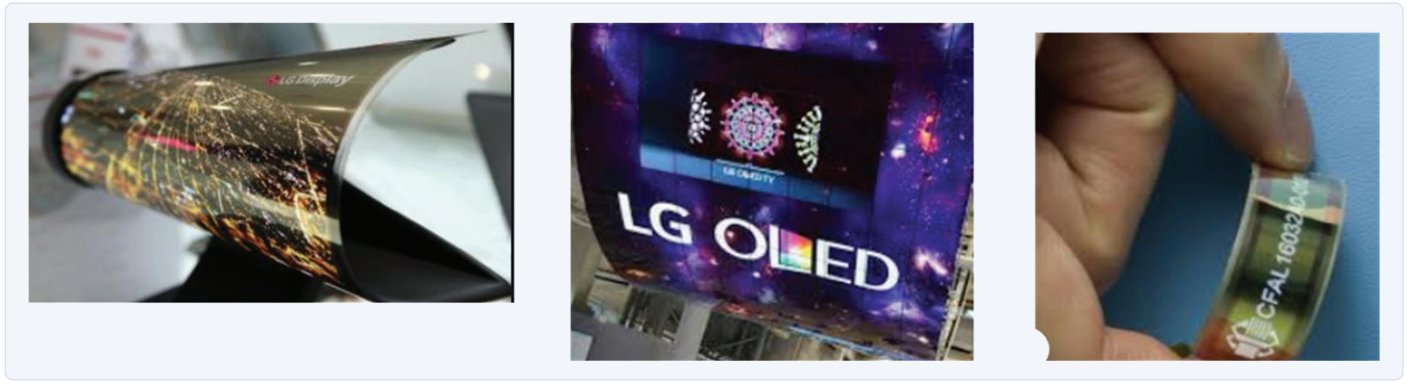


Figure 1. Quelques écrans OLED.

Note de l'éditeur : cet article est un extrait du livre « Using Displays in Raspberry Pi Projects » formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine *Elektor*. Puisque cet article est extrait d'une publication plus vaste, certains termes peuvent faire référence à des passages du livre d'origine situés ailleurs. L'auteur et l'éditeur ont fait de leur mieux pour l'éviter et seront heureux de répondre aux questions – pour les contacter, voir l'encadré « Des questions, des commentaires ? ».

Une diode électroluminescente organique est un composant d'affichage à base de LED, dans lequel la couche électroluminescente émissive est un film de composé organique qui émet de la lumière en présence d'un courant électrique. La couche organique est située entre deux électrodes dont au moins une est transparente. Les écrans OLED sont utilisés dans de nombreuses applications commerciales, telles que les écrans de télévision, les écrans d'ordinateur, les ordiphones, les consoles de jeux et autres écrans numériques. Les OLED peuvent être pilotés par des contrôleurs à matrice passive (PMOLED) ou à matrice active (AMOLED). Généralement, dans les écrans de type PMOLED, toutes les lignes et colonnes sont commandées séquentiellement, une par une. Dans les écrans de type AMOLED, on utilise un fond de panier de transistors à couches minces pour accéder directement à chaque pixel et l'activer ou le désactiver.
En général, les écrans OLED présentent les avantages suivants par rapport aux écrans LCD :

- La consommation d'énergie des OLED est très faible, ce qui les rend très intéressants pour les applications d'affichage mobile.
- Les écrans OLED ont des qualités d'image améliorées, un meilleur contraste, une plus grande luminosité, des angles de vision plus larges et des taux de rafraîchissement plus rapides.
- Les écrans OLED sont plus résistants, ce qui les rend aptes à une utilisation dans une plus large gamme de températures.
- Les écrans OLED peuvent être fabriqués pour être flexibles. On peut par exemple les plier, les porter au poignet, etc.

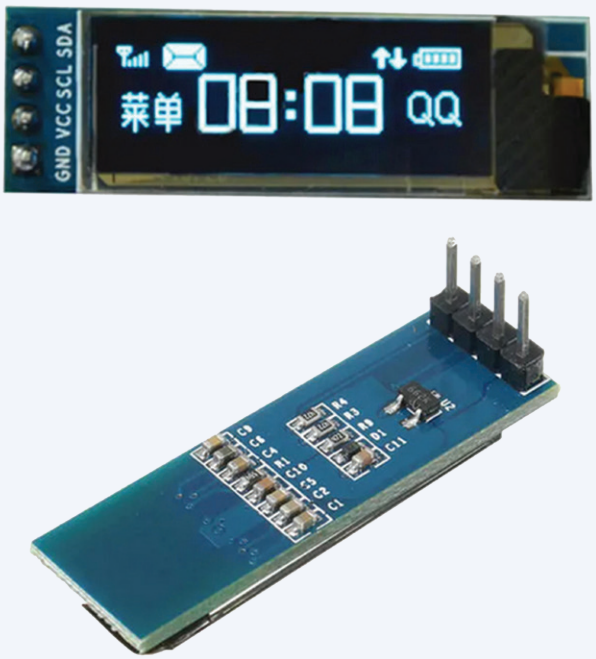


Figure 2. OLED de 128×32 pixels. Source : https://uk.banggood.com

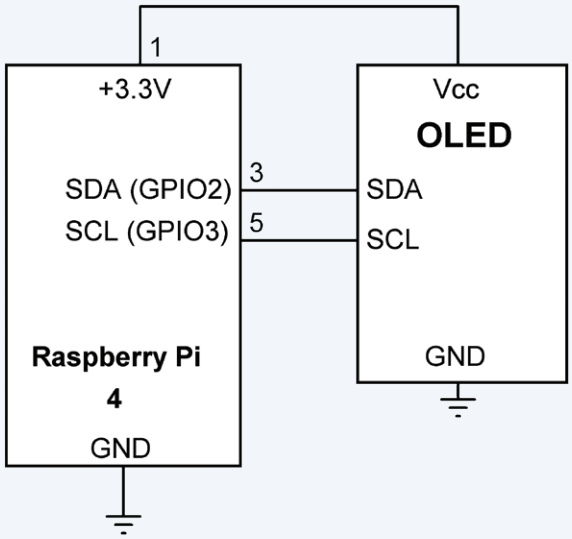


Figure 3. Le schéma du circuit.

Les écrans OLED ne sont cependant pas parfaits et présentent certains inconvénients :

- Leur coût est considérablement plus élevé.
- Leur durée de vie est limitée, bien que cette situation se soit récemment améliorée.
- Ils peuvent poser problème à la lumière directe du soleil en raison de leur nature émissive.

La **figure 1** montre quelques exemples d'écrans OLED.

Utilisation des écrans OLED

Il existe de nombreuses tailles d'écrans OLED avec différentes matrices de pixels. Il n'est évidemment pas possible de couvrir tous les types d'écrans OLED ici. Dans cet article, nous utiliserons le très populaire écran OLED de 128×32 pixels, piloté par la puce intégrée SSD1306 (voir **figure 2**). Il est peut-être opportun de passer en revue les caractéristiques de cet écran avant de l'incorporer à nos projets. Les spécifications de cet écran OLED sont les suivantes :

- Résolution : 128×32 pixels
- Couleur du pixel : blanc (le bleu est également disponible)
- Taille : 2,3 cm
- Interface : I²C
- Pilote : SSD1306
- 4 broches : GND, VCC, SCL (I2C), SDA (I2C)
- Angle de vue : plus de 160°
- Tension de fonctionnement : +3,3 V / +5 V

L'écran OLED piloté par la puce SSD1306 communique avec l'ordinateur hôte via l'interface I2C (c'est-à-dire que deux broches suffisent pour l'interface, en plus des broches d'alimentation et de masse). L'adresse I2C du pilote est 0x3C par défaut. La **figure 3** montre la connexion entre le Raspberry Pi et l'écran OLED. La puce fonctionne normalement en +3,3 V, mais un régulateur de tension sur la puce permet également un fonctionnement en +5 V.

Port Raspberry Pi	N° broche Raspberry Pi	Broche OLED
SDA	(GPIO2)	2
SCL	(GPIO3)	3
GND	39	GND
+3,3V	1	V_{cc}

Installation de la bibliothèque OLED

Il faut activer l'interface I2C sur le Raspberry Pi avec l'outil de configuration `sudo raspi-config`. Vous devez ensuite installer la bibliothèque OLED. Ici, nous utilisons la bibliothèque *Adafruit_Python_SSD1306*. Les étapes sont les suivantes (certaines de ces bibliothèques peuvent être déjà installées sur votre Raspberry Pi) :

```
pi@raspberrypi:~ $ sudo apt install -y python3-dev
python3- pil python3- pip python3-setuptools python3-
rpi.gpio
pi@raspberrypi:~ $ sudo pip3 install
adafruit-circuitpython-ssd1306
```

Avant d'utiliser un dispositif connecté à l'interface I2C, il faut d'abord détecter son adresse. Pour trouver l'adresse I2C de l'écran,

```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

Figure 4. Détection de l'écran OLED.

Figure 5. Coordonnées de l'écran.

Listage 1. OLEDCorners.py.

```
#-------------------------------------------------------------
#        MONTRE LES PIXELS AUX COINS DE L'ECRAN
#        ======================================
#
# Ce programme affiche les pixels aux quatre coins de l'écran
#
# Author: Dogan Ibrahim
# File   : OLEDCorners.py
# Date   : November 2020
#-------------------------------------------------------------
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)

display.fill(0)              # Efface les pixels
display.show()               # Affiche les données

display.pixel(0,0,1)         # Pixel en (0,0)
display.pixel(127,0,1)       # Pixel en (127,0)
display.pixel(0,31,1)        # Pixel en (0,31)
display.pixel(127, 31, 1)    # Pixel en (127,31)
display.show()               # Affiche les données
```

Listage 2. OLEDText.py.

```
#-------------------------------------------------------
#            AFFICHAGE DE TEXTE
#            ==================
#
# Ce programme affiche le texte E L E K T O R en (15,5)
#
# Author: Dogan Ibrahim
# File   : OLEDText.py
# Date   : November 2020
#-------------------------------------------------------
from PIL import Image,ImageDraw,ImageFont
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
font = ImageFont.load_default()          # police par défaut

display.fill(0)                          # efface l'écran
display.show()

width = display.width                    # Largeur
height = display.height                  # Hauteur
image = Image.new('1',(width,height))
draw = ImageDraw.Draw(image)
draw.text((15, 5), "E L E K T O R", font = font, fill = 255)
display.image(image)
display.show()
```

il faut exécuter la commande suivante dans le terminal :

pi@raspberrypi:~ $ `i2cdetect -y 1`

Le résultat devrait ressembler à la **figure 4**.

Projet 1 : afficher des pixels aux quatre coins de l'écran

Les coordonnées de l'écran OLED sont indiquées dans la **figure 5**. Le coin supérieur gauche est en (0,0). La coordonnée X traverse l'écran de 0 à 127, tandis que la coordonnée Y le descend de 0 à 31.

Listage du programme : le **listage 1** présente le programme *OLEDCorners.py*. Ce programme est inclus dans le fichier d'archive des logiciels du livre qu'on trouve à l'adresse [1], sous la rubrique *Téléchargements*. Au début du programme, on importe dans le programme la bibliothèque OLED d'Adafruit. Les pixels sont ensuite effacés en appelant la fonction `display.fill(0)`. Il est recommandé de toujours effacer les pixels au début d'un programme. Les pixels aux quatre coins de l'écran sont ensuite allumés. Notez qu'il faut appeler la fonction `display.show()` pour afficher les données sur l'écran.
La **figure 6** montre l'écran avec les quatre pixels d'angle activés.

Projet 2 : afficher du texte

Dans ce projet, on affiche le texte « ELEKTOR » à partir de la position (15,5) de l'écran.
Listage du programme : le **listage 2** montre le programme *OLEDText.py*. Il utilise les fonctions de la bibliothèque PIL [2]. Au début du programme, on importe les modules *image* et *font* de la bibliothèque PIL. La bibliothèque de polices par défaut est ensuite chargée. Enfin, le programme efface les pixels. Les variables `width` et `height` contiennent la largeur (128) et la hauteur (32) de l'écran.
Pour créer une image, il faut d'abord créer une image vide aux dimensions de l'écran, comme ceci :

```
image = Image.new('1', (width, height))
```

Avec cet objet 'image', on crée l'objet 'draw' (dessin) qui sert à dessiner des formes et du texte :

```
draw = ImageDraw.Draw(image)
```

Dans ce programme, nous voulons afficher un texte aux coordonnées (15,5), et on utilise donc la fonction `draw.text` :

```
draw.text((15, 5), "E L E K T O R", font = font, fill
   = 255)
```

Remarquez que `fill = 255` dessine l'image en blanc, et `0` la dessine en noir. Ensuite, nous devons afficher l'image en appelant les fonctions `display.image(image)` et `display.show()`. La **figure 7** montre le texte affiché.

Projet 3 : afficher de formes

On peut dessiner diverses formes sur l'écran. Pour une liste complète des fonctions, consultez la documentation de la bibliothèque PIL – il y a beaucoup plus de fonctions que celles décrites brièvement ici.

Rectangle :
(x,y) est le coin supérieur gauche du rectangle. `Outline` est la couleur du contour de la forme (255 pour le blanc, 0 pour le noir), `fill` est l'intérieur de la forme (255 pour le blanc, 0 pour le noir).

```
draw.rectangle((x, y, width, height), outline = nn,
   fill = mm)
```

Ellipse :

```
draw.ellipse((x, y, width, height), outline = nn,
   fill = mm)
```

Ligne :
(x1,y1) est la première coordonnée, (x2,y2) est la seconde.

```
draw.line((x1, y1, x2, y2), fill = mm)
```

Arc :

```
draw.arc((x1,x2,y1,y2), s1, s2, fill = mm)
```

Trace un arc entre les angles de départ (s1) et d'arrivée (s2), à l'intérieur de la boîte (x1,x2,y1,y2).

Corde :

```
draw.chord((x1,x2,y1,y2), s1, s2, fill = mm)
```

Trace une ligne entre les angles de départ (s1) et d'arrivée (s2), à l'intérieur de la boîte (x1,x2,y1,y2).

Camembert :

```
draw.pieslice((x1,y1,x2,y2), s1, s2, fill = mm)
```

Identique à `arc`, mais dessine également des lignes droites entre les points d'extrémité et le centre de la boîte englobante.

Point :

```
draw.point((x1,x2), fill = mm)
```

Dessine des pixels aux coordonnées données. Il est possible de dessiner plus d'un pixel, comme dans l'exemple ci-dessous où trois pixels individuels sont dessinés aux coordonnées données :

```
draw.point([(x1,y1), (x2,y2), (x3,y3)], fill = mm)
```

Figure 6. Nous avons des pixels aux quatre coins de l'écran OLED !

Figure 7. Le texte affiché : « E L E K T O R » – quoi d'autre ?

Listage 3. LEDRect.py.

```
#-----------------------------------------------------------------
#
#        AFFICHE UN RECTANGLE AVEC DU TEXTE À L'INTÉRIEUR
#        ================================================
#
# Dans ce programme, on affiche un rectangle dans un coin de
# l'écran et le texte R E C T A N G L E est inscrit à l'intérieur
#
# Author: Dogan Ibrahim
# File   : OLEDRect.py
# Date   : November.py
#-----------------------------------------------------------------
from PIL import Image,ImageDraw,ImageFont
from board import SCL, SDA
import busio
import adafruit_ssd1306

 i2c = busio.I2C(SCL, SDA)
 display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
 font = ImageFont.load_default()

 display.fill(0)
 display.show()

 width=(display.width)
 height=(display.height)
 image=Image.new('1',(width,height))
 draw=ImageDraw.Draw(image)
 draw.rectangle((0,0, 127, 31),outline = 255, fill = 0)
 draw.text((15, 12),"R E C T A N G L E", font = font, fill = 255)
 display.image(image)
 display.show()
```

Listage 4. OLEDShape1.py.

```
#---------------------------------------------------------------
#
#    AFFICHE QUATRE RECTANGLES AVEC LES CHIFFRES 1,2,3,4
#    ===================================================
#
# Dans ce programme, on affiche 4 rectangles. On inscrit le chiffre 1
# à l'intérieur du rectangle 1, le chiffre 2 à l'intérieur du
# rectangle 2, etc.
#
# Author: Dogan Ibrahim
# File   : OLEDShape1.py
# Date   : November 2020
#---------------------------------------------------------------
from PIL import Image,ImageDraw,ImageFont
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
font = ImageFont.load_default()

display.fill(0)
display.show()

width=(display.width)
 height=(display.height)
image=Image.new('1',(width,height))
draw=ImageDraw.Draw(image)
draw.rectangle((0,0, 127, 31),outline = 255, fill = 0)
draw.line((64, 0, 64, 31), fill = 255)
draw.line((0, 16, 127, 16), fill = 255)
draw.text((32, 4), "1", font = font, fill = 255)
draw.text((94, 4), "2", font = font, fill = 255)
draw.text((32, 17),"3", font = font, fill = 255)
draw.text((94, 17),"4", font = font, fill = 255)
display.image(image)
display.show()
```

Listage 5. LEDShape2.py.

```
#---------------------------------------------------------------
#
#        AFFICHE UNE CORDE ET UN POLYGONE
#        ================================
#
# Dans ce programme, on dessine une corde et un polygone.
# Le polygone est rempli en blanc.
#
# Author: Dogan Ibrahim
# File   : OLEDShape2.py
# Date   : November 2020
#---------------------------------------------------------------
from PIL import Image,ImageDraw,ImageFont
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
font = ImageFont.load_default()

display.fill(0)
display.show()

width=(display.width)
height=(display.height)
image=Image.new('1',(width,height))
draw=ImageDraw.Draw(image)
draw.chord((0, 0, 90, 30), 10, 180, fill = 255)
draw.polygon([(100,5), (120,9), (120,25),
(100,30)],outline=255,fill=255)
display.image(image)
display.show()
```

Figure . L'affichage obtenu en exécutant OLEDRect.py.

Figure 9. L'affichage obtenu en exécutant OLEDShape1.py (rectangles).

Figure 10. L'affichage obtenu en exécutant OLEDShape2.py (corde et polygone).

Des questions, des commentaires ?
Envoyez un courriel à l'auteur (d.ibrahim@btinternet.com) ou contactez Elektor (redaction@elektor.fr).

Contributeurs
Texte : **Dogan Ibrahim**
Rédaction : **Jan Buiting**
Mise en page : **Harmen Heida**
Traduction : **Denis Lafourcade**

Polygone :

```
draw.polygon([(x1,y1), (x2,y2), (x3,y3), ......],
   outline = nn, fill = mm)
```

Dessine un polygone dont le contour est constitué de lignes droites entre les coordonnées données, plus une ligne droite entre la dernière et la première coordonnée. La liste des coordonnées peut être un objet séquentiel contenant soit des couples [(x, y), ...], soit des valeurs numériques [x, y, ...]. Elle doit contenir au moins trois coordonnées.

Le programme *OLEDRect.py* (**listage 3**) montre le dessin d'un rectangle aux coins de l'écran et l'écriture du texte « RECTANGLE » à l'intérieur de ce rectangle. La **figure 8** montre le résultat.

Le programme *OLEDShape1.py* (**listage 4**) affiche quatre rectangles. Dans le rectangle 1, on inscrit le chiffre « 1 ». Dans le rectangle 2, on inscrit le chiffre « 2 », et ainsi de suite. La **figure 9** montre le résultat.

Le programme *OLEDShape2.py* (**listage 5**) affiche une corde et un quadrilatère. Le paramètre de remplissage du polygone est fixé à 255, il est donc rempli d'un fond blanc. La **figure 10** montre le résultat.

Listage 6. OLEDBitmap.py.

```
# ---------------------------------------------------
#
#        AFFICHE UN BITMAP
#        =================
#
# Ce programme affiche une image bitmap
#
# Author: Dogan Ibrahim
# File  : OLEDBitmap.py
# Date  : November 2020
#----------------------------------------------------
from PIL import Image,ImageDraw,ImageFont
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32,
i2c)

display.fill(0)
display.show()

width=(display.width)
height=(display.height)
image=Image.new('1',(width,height))
image=Image.open('LetterA.png').
resize((width,height),\
Image.ANTIALIAS).convert('1')
display.image(image)
display.show()
```

Publicité

Capteurs d'humidité de la série SHT

Mesure précise de l'humidité
basée sur la technologie CMOSens®.

Série SHT2x
solutions éprouvées pour votre application

Plage d'humidité mesurée	0-100% RH
Interface de communication	I2C, PWM, SDM

Série SHT3x
mesure d'humidité fiable et précise

Plage d'humidité mesurée	0-100% RH
Interface de communication	En fonction du modèle : sortie analogique ou I2C

Série SHT40 - une nouvelle génération
de mesure d'humidité encore plus précise

Plage d'humidité mesurée	0-100% RH
Interface de communication	I2C

Allez sur na tme.eu et découvrez-le !

T M E ®
Electronic Components

TRANSFER MULTISORT ELEKTRONIK
GLOBAL DISTRIBUTEUR DE COMPOSANTS ÉLECTRONIQUES

Ustronna 41, 93-350 Łódź, Pologne
+48 42 645 54 44, export@tme.eu, tme.eu

tme.eu
facebook.com/TME.eu
youtube.com/TMElectroniComponent
instagram.com/tme.eu

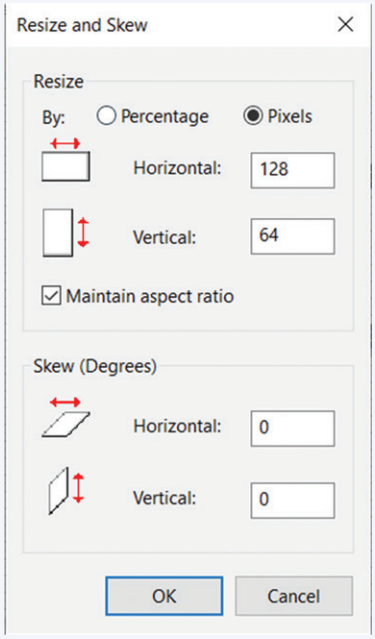


Figure 11. Dans MS Paint, réglage du nombre de pixels sur 128×64.

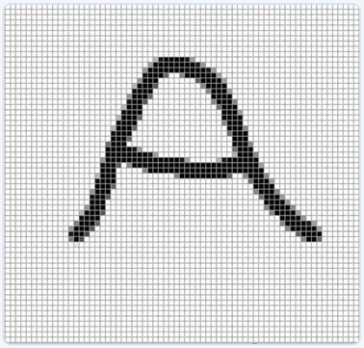


Figure 12. Dessinez la forme requise dans MS Paint.

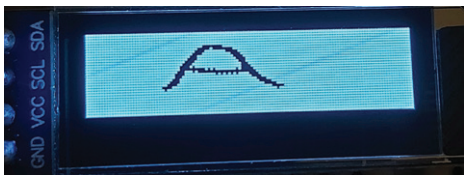


Figure 13. Gagné ! Une lettre quasi manuscrite sur l'écran OLED.

Projet 4 : création et affichage d'un bitmap

Ici, nous créons une image bitmap de 128×32, puis l'affichons à l'écran. L'image sera la lettre « A ». Les étapes sont détaillées ci-dessous.

Création de l'image : nous utilisons le programme *Paint* de Microsoft pour créer notre image.

Lancez le programme *MS Paint*.

Cliquez sur *Accueil Redimensionner*, puis sur *Pixels*. Réglez *Horizontal* sur 128 pixels et *Vertical* sur 64 pixels comme indiqué sur la **figure 11**. Cliquez sur *OK*.
Cliquez sur *Affichage*, puis sur *Zoom avant* pour agrandir la grille. Dessinez la forme que vous souhaitez à l'aide de la souris. Dans ce projet, la lettre *A* est dessinée à la main, comme le montre la **figure 12**.
Enregistrez le fichier, par exemple sous le nom de *LetterA.png*. Copiez le fichier dans le répertoire personnel (*/home/pi*) de Raspberry Pi. Vous pouvez utiliser le programme *WinSCP*, disponible gratuitement, pour copier le fichier.

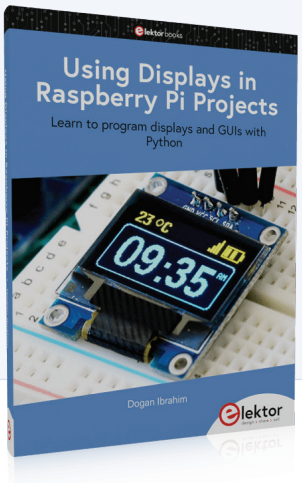
Le programme final, *OLEDBitmap.py*, figure dans le **listage 6**. On utilise la fonction `Image.open` pour ouvrir le fichier image, le fichier est redimensionné et converti en couleur sur 1 bit comme requis par la bibliothèque. La **figure 13** montre le résultat.

Tous les programmes abordés dans cet article sont disponibles sur la page web Elektor du livre [1]. Sur cette page, allez jusqu'à *Téléchargements* pour trouver le fichier qui regroupe tous les programmes. Téléchargez le fichier .zip, extrayez-le sur votre ordinateur, puis localisez et ouvrez les six programmes Python dont il est question ici.

(210197-04)

PRODUITS

- **Livre en anglais, D. Ibrahim,** *Using Displays in Raspberry Pi Projects* **(Elektor 2021)**
- **Version papier :**
  www.elektor.fr/using-displays-in-raspberry-pi-projects
- **Version numérique :**
  www.elektor.fr/using-displays-in-raspberry-pi-projects-e-book



LIENS

[1] **Téléchargement des programmes** : http://www.elektor.fr/using-displays-in-raspberry-pi-projects
[2] **Bibliothèque LIP** : http://effbot.org/imagingbook/imagedraw.htm