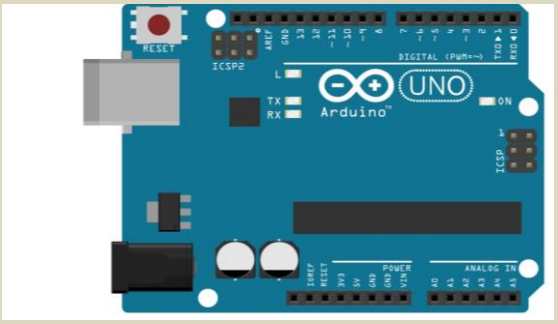
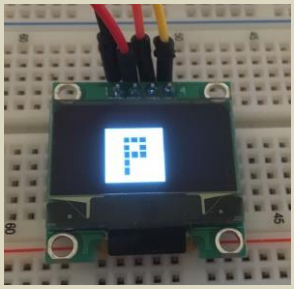
Mon Arduino veut Un Écran OLED 124x68 Sur Arduino

Présentation du matériel.





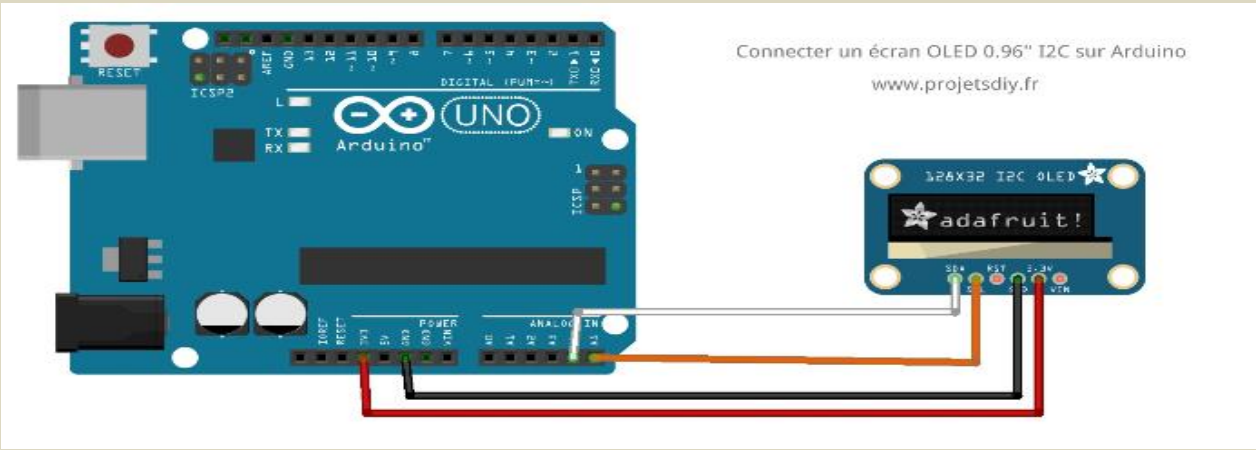
- Citer les différentes technologies d'écrans.
- Donner la définition d'OLED.

- Trouver les caractéristiques de l'écran OLED I2C SSD1306.

- Expliquer succinctement le bus I2C.
- Donner les avantages et inconvénients de l'I2C.

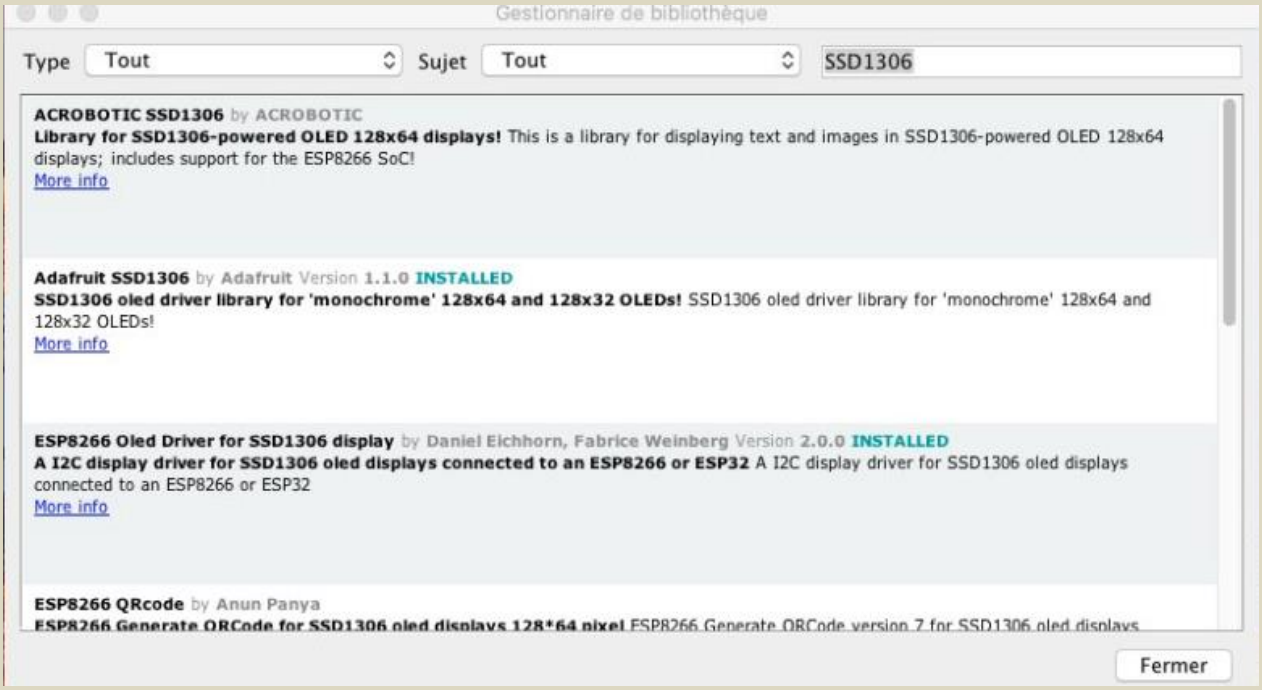
Câblage de l'écran OLED I2C SSD1306 128x64p 0.96" sur Arduino

Vous disposez d'un Arduino Uno, relié le Pin A4 sur le Pin SDA (Serial DAta) de l'écran et le port A5 sur le Pin SCL (Serial CLock).
Donner le rôle de SDa et Clock.



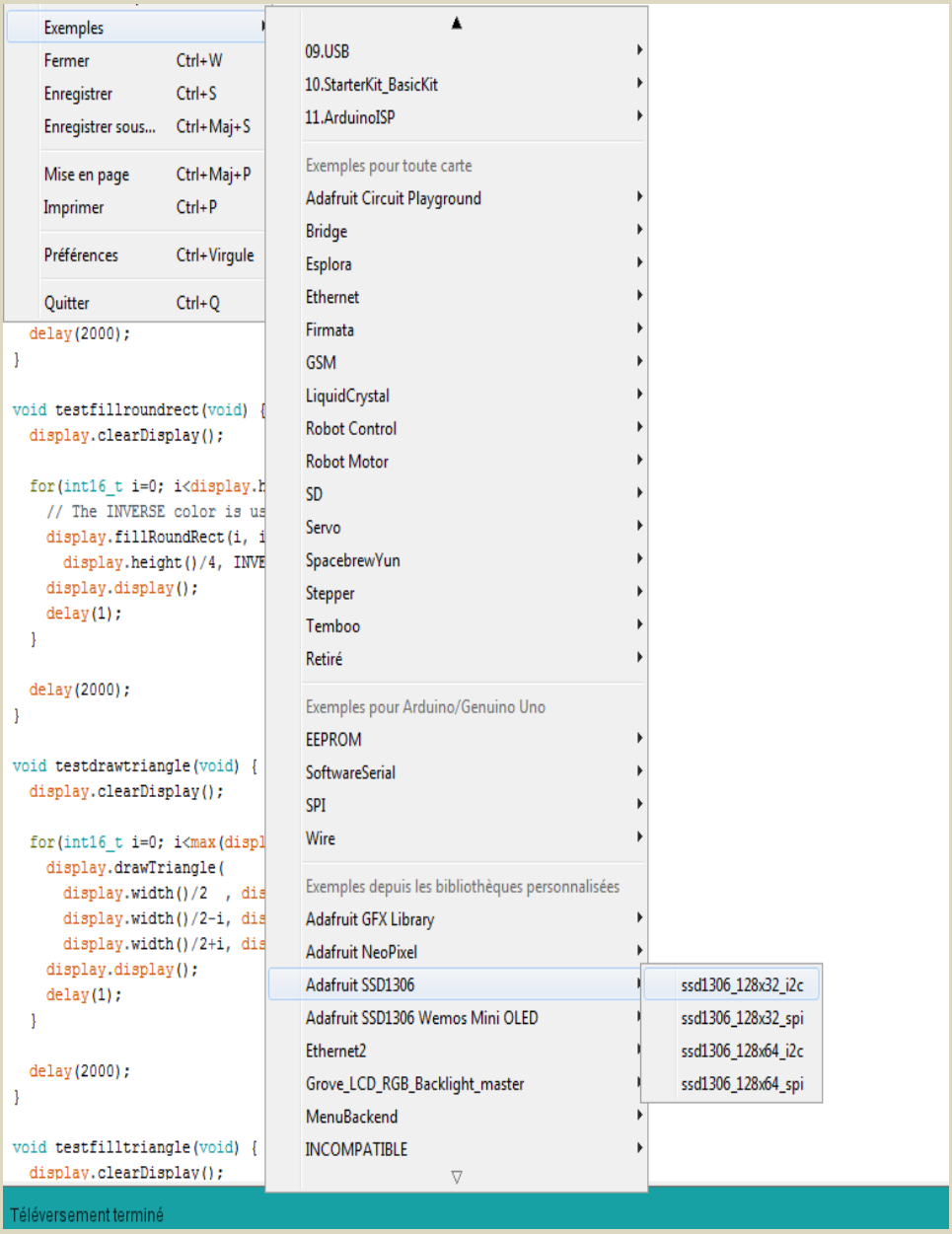
Comment installer une librairie sur l'IDE Arduino ?

Pour ceux qui découvrent l'IDE Arduino, vous aurez besoin d'ajouter des librairies pour faire fonctionner l'écran OLED. Certaines librairies sont directement disponibles depuis le gestionnaire de librairies. C'est le cas par exemple de la librairie Adafruit pour le SSD1306. Dans le menu croquis, aller dans Inclure une bibliothèque puis Gérer les bibliothèques.



Adafruit a développé une librairie très puissante qui va nous permettre de gérer l'affichage de notre mini écran mais aussi de tracer plein de chose très facilement grâce à la librairie dédiée, GFX Library.

- Installer les 2 librairies « Adafruit_GFX et Adafruit_SSD1306 ».
- Exécuter le programme suivant : ssd1306 128*32 i2c.ino et vérifier son fonctionnement.

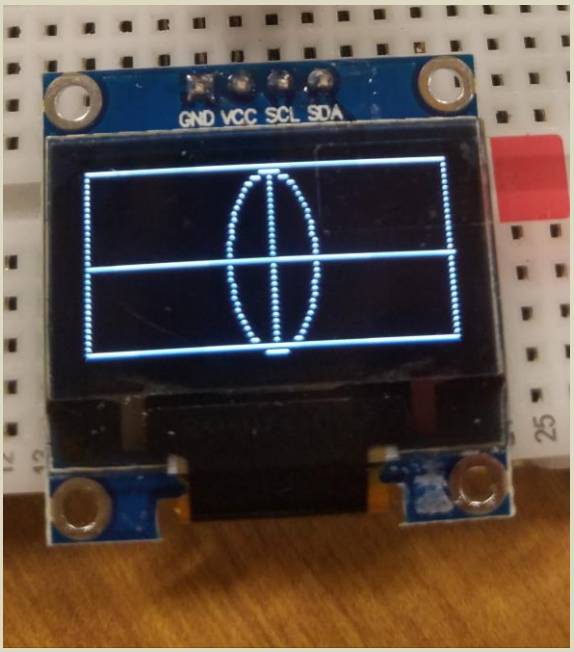

Exemples
Fermer Ctrl+W
Enregistrer Ctrl+S
Enregistrer sous... Ctrl+Maj+S
Mise en page Ctrl+Maj+P
Imprimer Ctrl+P
Préférences Ctrl+Virgule
Quitter Ctrl+Q
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Exemples pour toute carte
Adafruit Circuit Playground
Bridge
Esplora
Ethernet
Firmata
GSM
LiquidCrystal
Robot Control
Robot Motor
SD
Servo
SpacebrewYun
Stepper
Temboo
Retiré
Exemples pour Arduino/Genuino Uno
EEPROM
SoftwareSerial
SPI
Wire
Exemples depuis les bibliothèques personnalisées
Adafruit GFX Library
Adafruit NeoPixel
Adafruit SSD1306
Adafruit SSD1306 Wemos Mini OLED
Ethernet2
Grove_LCD_RGB_Backlight_master
MenuBackend
INCOMPATIBLE
ssd1306_128x32_i2c
ssd1306_128x32_spi
ssd1306_128x64_i2c
ssd1306_128x64_spi
delay(2000);
}
void testfillroundrect(void) {
display.clearDisplay();
for(int16_t i=0; i<display.h
// The INVERSE color is us
display.fillRoundRect(i, i
display.height()/4, INVE
display.display();
delay(1);
}
delay(2000);
}
void testdrawtriangle(void) {
display.clearDisplay();
for(int16_t i=0; i<max(displ
display.drawTriangle(
display.width()/2 , dis
display.width()/2-i, dis
display.width()/2+i, dis
display.display();
delay(1);
}
delay(2000);
}
void testfilltriangle(void) {
display.clearDisplay();
Téléversement terminé

Liste des fonctions de la librairie disponibles

Fonctions de la librairie Adafruit_SSD1306	
Adafruit_SSD1306 display(OLED_RESET)	initialise l'objet display(Pin pour le reset)
display()	Actualise l'affichage
clearDisplay()	Efface l'écran et le buffer
invertDisplay(bool)	inverse l'affichage (true ou false)
Fonctions Adafruit_GFX	
drawPixel(uint16_t x, uint16_t y, uint16_t color)	Dessine un pixel en X,Y de la couleur color
drawLine(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t color)	Dessine une ligne de X1,Y1 à x2,Y2 de la couleur color
drawFastVLine(uint16_t x0, uint16_t y0, uint16_t length, uint16_t color) **drawFastHLine(uin86_t x0, uin86_t y0, uint8_t length, uint16_t color);**	Tracé optimisé de lignes horizontales et verticales
drawRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color) **fillRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color)**	Dessine un rectangle depuis X,Y de largeur w et hauteur h idem mais plein
drawCircle(uint16_t x0, uint16_t y0, uint16_t r, uint16_t color) **fillCircle(uint16_t x0, uint16_t y0, uint16_t r, uint16_t color)**	Dessine un cercle de centre X,Y et de rayon r idem mais le cercle est plein
drawRoundRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t radius, uint16_t color) **fillRoundRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t radius, uint16_t color)**	Idem rectangle mais avec un arrondi de rayon raduis aux angles
drawTriangle(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color) **fillTriangle(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)**	Dessine un triangle en spécifiant les coordonnées de chaque sommets (x0,y1), (x2,y2), (x3,y3)
drawChar(uint16_t x, uint16_t y, char c,	Dessine un caractère en x,y

uint16_t color, uint16_t bg, uint8_t size)	
drawBitmap(int16_t x, int16_t y, uint8_t *bitmap, int16_t w, int16_t h, uint16_t color)	Affiche un bitmap en x,y de largeur w et hauteur t
fillScreen(uint16_t color);	Colorie entièrement l'écran dans la couleur spécifiée
setRotation(uint8_t rotation)	Rotation de l'affichage : 0 -> 0°, 1 -> 90°, 2 -> 180°, 3 -> 270°

Réaliser un programme en vous aidant du programme test et de la librairie Adafruit_SSD1306, qui affiche l'écran suivant :



Programme a compléter :

```
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#define OLED_RESET 4
Adafruit_SSD1306 display( OLED_RESET );
#if( SSD1306_LCDHEIGHT != 32 )
#error( "Height incorrect, please fix
Adafruit_SSD1306.h!" );
#endif
void setup()
{
 // Initialise la communication I²C à l'adresse 0x3C.
 display.begin( SSD1306_SWITCHCAPVCC, 0x3C );
 display.clearDisplay();

 // Affiche des textes
 affiche();
}

void loop()
{

}

void affiche( void )
{
```

Programme à compléter…

```
}
```

Faire valider par le Duce de l'électronique.

Réaliser un programme pour afficher un texte

L'affichage d'un texte demande un peu plus de travail. Il est nécessaire de modifier les paramètres d'affichage paramètre par paramètre. Voici un petit exemple pour afficher « Vive les sin « en 4,0:

```
display.setTextSize( 1 );
 display.setTextColor( WHITE );
 display.clearDisplay();

 display.setCursor( 4, 0 );
 display.println( "Vive les  sin" );
```

Compléter les commentaires.

Ecrire le programme complet et faire valider par El professor.

Comment préparer et afficher des images Bitmaps

Vous voudrez très certainement afficher plus qu'un simple texte sur votre petit écran. Voici une solution pour convertir vos Bitmaps en chaine Hexa.

LCD Assistant est un petit utilitaire bien pratique qui va vous permettre de convertir d'importe quel image au format Bitmap en un tableau de caractères pouvant être affiché sur votre mini écran OLED. LCD Assistant est recommandé par Sparkfun mais vous en trouverez d'autres sur internet avec les mots clés "convert bitmap to graphic lcd".
Pour commencer vous devez disposer d'une image au format Bitmap. Après quelques tests, je vous conseil de redimensionner votre image pour quelle soit compatible avec l'écran cible. LCD Assistant est un programme bien pratique mais déjà ancien, il ne faut pas trop lui en demander et corriger à la main les petits défauts. J'ai pris pour cet exemple le logo de Projets DIY que j'ai redimensionné en 64 x 64 pixels (le logo est carré). Exportez l'image au format BMP monochrome. Si votre image dépasse la résolution demandée, elle sera tronquée et seule la partie en haut à gauche de l'image importée sera générée. J'ai obtenu le meilleur résultat en prenant la taille inférieure de 48×48 pixels.

Ouvrez l'image dans LCD Assistant en laissant 8 pixels/byte, orientation verticale pour Byte Orientation. Exportez le tableau via File -> Save Output. Donnez d'importe quel nom avec une extension .txt par exemple pour pouvoir ouvrir facilement le fichier avec un petit éditeur de texte. Vous obtiendrez une variable contenant votre image convertie. La conversion n'est pas toujours parfaite, bien souvent (toujours !!) vous devrez supprimer la première ligne du tableau [0x40, 0x00, 0x40, 0x00,] (attention de ne pas oublier la virgule).

Affichage d'un bitmap à l'aide de la librairie Sparkfun

On commence par nettoyer l'écran

```
oled.clear(ALL);
```

En imaginant que votre écran OLED soit déclaré avec l'objet oled, il suffit d'appeler la fonction drawBitmap en lui passant en paramètre la variable contenant l'image à afficher.

```
oled.drawBitmap(bender);
```

Maintenant on actualise l'écran. C'est tout

```
oled.display();
```

Affichage d'un bitmap avec la librairie Adafruit_GFX

La librairie Adafruit_GFX utilise une variable PROGMEM pour stocker l'image. La variable suivante a été générée à l'aide d'image2cpp présenté au prochain paragraphe

```
const unsigned char myBitmap [] PROGMEM = {
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x10,
0xff, 0xff, 0xe0, 0x10, 0x08, 0x07, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x10,
0xff, 0xff, 0xe0, 0x10, 0x08, 0x07, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xe0, 0x1f, 0xf8, 0x07, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x1f,
0xff, 0xff, 0xe0, 0x1f, 0xf8, 0x07, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xe0, 0x10, 0x08, 0x07, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x10,
0xff, 0xff, 0xe0, 0x10, 0x08, 0x07, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x10, ,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x1f,
0xff, 0xff, 0xe0, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x1f,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x1f,
0xff, 0xff, 0xe0, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x1f,
0xff, 0xff, 0xe0, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,};
```

Les commandes sont très similaires à celles de Sparkfun. On commence par nettoyer l'écran

```
display.clearDisplay();
```

La commande drawBitmap nécessite les paramètres suivants : position x, position y, variable contenant le bitmap, largeur, hauteur, couleur. Pour mes tests, j'ai un écran de 128×32 pixels. Pour obtenir un logo carré, j'ai étendu l'image en doublant le nombre de pixels de x (64×32).

```
display.drawBitmap(32,0,myBitmap,64,32,WHITE);
```

On actualise l'affichage

```
display.display();
Ecrire le programme complet et faire valider  par grand Guana.
```