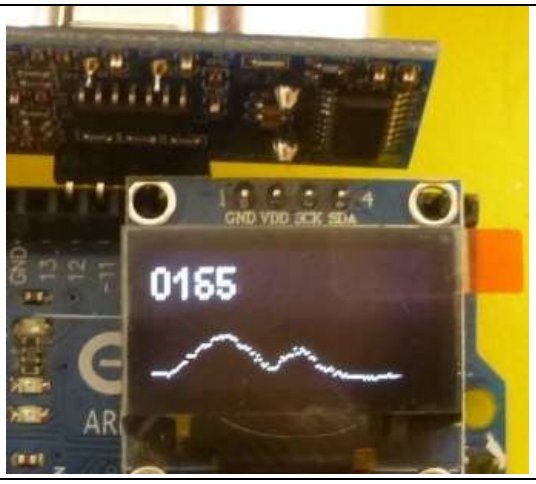
Affichage Oled SSD1306

Liens aux docs sous https://www.didel.com/prof/Oled.html

L'affichage Old SSD1306 peut être utilisé sur les sorties I2C des cartes Arduino avec le logiciel AdaFruit. Ce logiciel est gros et ses fonctions de dessin inutiles pour la mise au point. Didel a séparé la fonction de transfert et l'écriture qui peut se faire soit instantanément, soit globalement (bitmap puis mise à jour en 50ms).
Des variantes de la fonction spécialisée de transfer "bitbang", c'est-à-dire programmée directement avec les instructions de base) définissent sur quelles pins on est câblé. Avec le Xbot, c'est en général les pins 11,10,9,8 et ne nom du fichier à importer est OledI2Cbb11.h.
La librairie graphique ne connaît que les points et les ligne horizontales et verticales. Les autre fonctions dans OledPix.h affichent des nombres.



*Attention: Les OledI2Cbb****.h utilise les noms Write() Startt() Stop() Ack() qu'il ne faut pas utiliser dans d'autres programmes.*

Fonctions dans OledPix****.h

Licol(li,co); **Bin8(v);** **Hex8(v);** **Hex16(v);** **Dec8(v);** **Dec9999(v);**	. Place le pointeur ligne li (0 à 7) et en colonne co (0 à 127, mais il faut la place pour finir) . Affiche la variable 8 bits (byte, char, int8_t, uint8_t) en binaire . Affiche la variable 8 bits (byte, char, int8_t, uint8_t) en binaire . Affiche la variable 16 bits (int, int16_t uint6_t) en binaire . Affiche la variable 8 bits (byte, char, int8_t, uint8_t) en binaire . Affiche la variable 16 bits (int, int16_t uint6_t) en décimal, limite 0x270F=9999 note: Dec16 aurait utilisé 5 chiffres, et c'est rare que l'on doive afficher des nombres plus grands que 9999 en interagissant avec des capteurs.
Car('a'); **Text("txt");**	
Pour décorer	`Sprite (smile); Sprite (sad); Sprite (carre);` MySprite (mondessin);

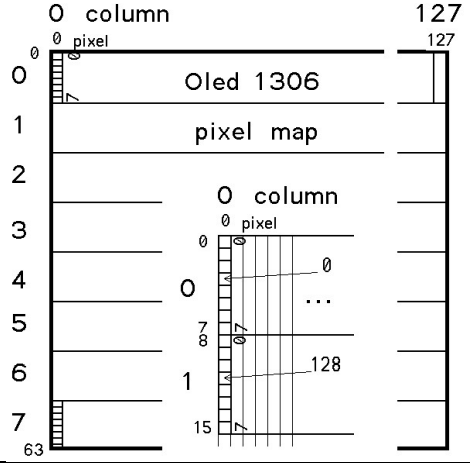
En double taille : (utilise la ligne en-dessus, donc début en lignes 1,3,5,7):

BigBin8(); BigHex8(); BigHex16(); BigDec8(); BigDec9999();
Big(); (ou BigCar();) BigText("txt");

Graphique

L'origine des coordonnées est en haut à gauche
Utiliser l'origine "scolaire" nécessite une simple soustraction : x= (63-x).
Ecrire un point efface les autres pixels du byte écran qu'il contient. Cela n'est pas gênant pour la seule application qui nous intéresse : visualiser des suites de mesures.
Les primitives sont

Dot(x,y); DDot(x,y); Vline(x); Hline(y);
DDot (x,y) affiche 2 points superposés. Cela permet de distinguer 2 courbes.



SSD1306 en format 128x32

En appelant la fonction `DoubleH();` (qui envoie les commandes 0xda et 0x02) on limite le nombre de lignes à 4, mais il faut les numéroter à partir de 4. Si on veut des gros caractères, ils sont en lignes 5 et 7.
Si on utilise `DoubleH();` avec le 128x64, les lignes sont dédoublées, les caractères sont plus lisibles avec le même temps d'écriture. Big est encore plus grand.

Sprite() et mySprite()

L'écran est fait de bytes alignés verticalement. Licol (li,co); positionne le pointeur sur un byte. Facile de copier une table à partir de cette position.
Pour le lutin smile, la table est

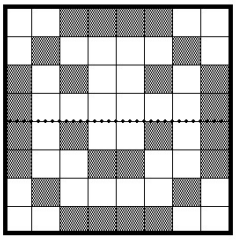
```
  byte  mysmile[]= {0x3c,0x42,0x95,0xa5,0xa1,0xa1,
0xa5,0x95,0x42,0x3c};
```

L'instruction pour copier à partir du pointeur est

```
 MySprite[mysmile];
```

La fonction Sprite utilise le même format, mais la table doit être mise avec le générateur de caractère dans OledPix****.h, comme la table pour ''smile''.
Après le setup, on a

```
 byte  mysmile[]= {0x3c,0x42,0x95,0xa5,0xa1, \
                   0xa1,0xa5,0x95,0x42,0x3c};
void loop(){
  LiCol(0,0); Sprite(smile);
  LiCol(2,0); MySprite(mysmile);
  LiCol(4,0); Text("Hello");
  delay(200);
}
```



Sprite de 2 ligne ou plus

La limitation avec OledPix est que l'on écrit des bytes consécutifs sur une ligne. Un sprite sur plusieurs ligne est écrit comme plusieurs sprite, alignés verticalement avec LeCol(). On peut en faire une fonction.

```
byte bigSup[]={ 00,c0,   } ;
byte bigSdown[]={ 00,03,   } ;

void BigSmile(byte li,byte col) {
  Licol(li,col) ; MySprite (bigSup) ;
  Licol(li+1,col) ; MySprite (bigSdown) ;

}
```