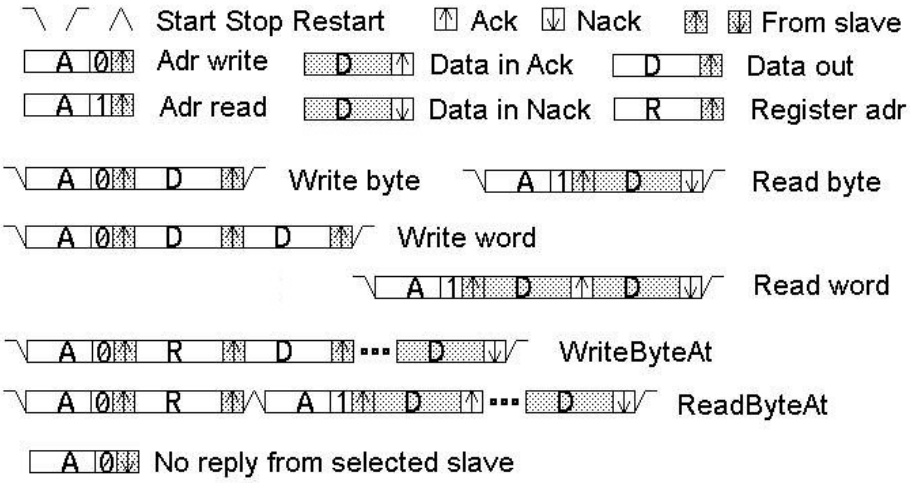
Librairies I2C

La librairie Wire est lourde à utiliser.
On a avantage à la remplacer par la librairie I2Ctwi qui traduit directement le protocole I2C et ne passe pas par un tampon qui ne se justifie qu'exceptionnellement.



	I2Ctwi 688b 12v	Wire 2814b 221v
Setup	SetupI2Ctwi();	Wire.begin();
Adresse	ad8 8 bits, double de l'adresse 7 bits	ad7 7 bits
Paramètres	ad8, rr registre, dd data 8 ou 16	ad7, rr registre, dd data 8 ou 16
Ecriture 8b	WriteByte (ad8,dd8);	Wire.beginTransmission(ad7); Wire.write(dd); Wire.endTransmission(true);
Ecriture 16b	WriteWord (ad8,dd16);	Wire.beginTransmission(ad7); Wire.write(dd16>>8); Wire.write(dd16&0xFF); Wire.endTransmission(true); Wire.endTransmission(true);
Lecture 8b	ReadByte (ad8);	Wire.requestFrom(ad7,1,true); dd8=Wire.read();
Lecture 16b	ReadWord(ad8);	Wire.requestFrom(ad7,2,true); dd16 =Wire.read()<<8|Wire.read();
Ecriture 8b dans le reg rr	WriteByteAt (ad8,rr,dd8);	
Ecriture 16b dans les reg rr et rr+1	WriteByteAt (ad8,rr,dd8);	
Lecture 8b dans le reg rr	dd8=ReadByteAt (ad8,rr);	
Lecture 16b dans les reg rr et rr+1	dd16=ReadWordAt(ad8,rr);	Wire.beginTransmission(ad7); Wire.write(rr); Wire.endTransmission(false); Wire.requestFrom(ad7,true); dd16 =Wire.read()<<8|Wire.read();

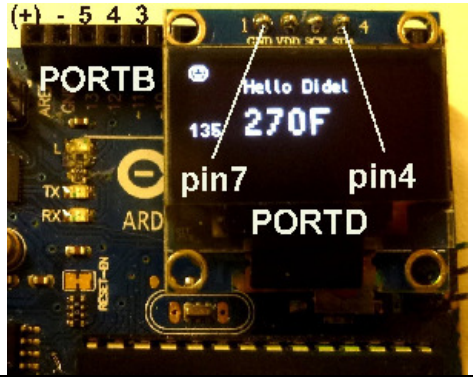
Les transferts de bloc sont faciles à rajouter.

Des librairies simplifiées on été définies pour commander les Oled SSD1306. On ne peut que leur écrire des bytes sur un Oled, inutile de trainer toute la fonctionnalité ci-dessus. On a un seul ordre:
WriteByteOled(dd); (a différents noms dans nos programmes suite à l'évolution historique).

I2C en bitbang

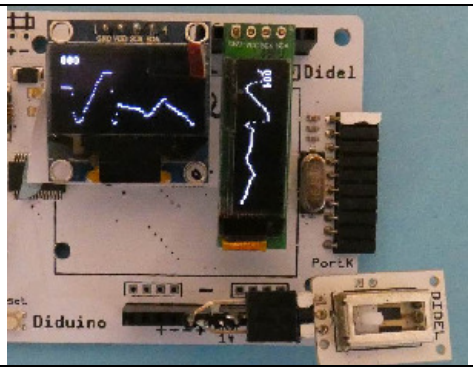
On programme facilement I2C en "bitbang". Ceci permet de mettre un ou plusieurs Oled sur n'importe quelles pins, mais il faut éviter les collisions et bien définir les bits concernés et les alimentation si elles sont programmées sur des pins.
Les fonctions bitbang n'ont été définies que pour le Oled, mais on pourrait les étendre pour toutes les fonctions. C'est seulement si on utilise le flag d'interruption de TWI que l'on gagne en disponibilité du processeur. Rappelons qu'un transfert élémentaire I2C dure 100µs. Servir l'envoi d'un byte TWI prend 2-3 µs et l'interruption 2 µs; on gagne donc un temps important avec TWI si le flag est testé tous les 20 µs (ralentit un peu le transfert à cause du jitter) ou déclenche une interruption.

I2Cbb pour Arduino
La consommation du 1306 est faible, on peut l'alimenter par 2 pins programmées à l'état 0 et 1. Les versions sont OledI2Cbb7.h, pour la variante qui utilise les pins 7,6,5,4. Si on a compris les opérations logiques, c'est facile de modifier les10 lignes de définition.
Attention, les noms utilisés sont Start Stop Write; ce sont de noms qu'ils ne faut pas utiliser ailleurs; c'est aussi facile de changer, et si le .h reste associé au programme, il n'y aura pas d'incompatibilité.



Si on veut programmer en BB un esclave I2C qui a une autre adresse sur les même pins, il faut utiliser des fonctions avec l'adresse explicite en premier paramètre, ou redéfinit l'adresse avant chaque utilisation (dangereux) ou activer un flag qui sélectionne l'adresse avec laquelle on travaille.

Si on veut câbler un 2e Oled, qui a la même adresse, il faut utiliser d'autres pins. On écrit sur un affichage ou sur l'autre après l'avoir sélectionné Voir
https://www.didel.com/OledBbLib.pdf
On pourrait aussi définir un autre .h qui définit Write1(), Write2(), mais il faut alors aussi dédoubler les programmes qui utilisent un affichage ou l'autre.



Dans Pégase, le capteur est sur les pins I2C, on utilise la librairie I2Ctwi.h avec les appels vus plus haut. Oled est sur les pins Arduino 2,3, accédé par I2Cbb.h avec les bonnes définitions.

Comparaison (un accès WriteByte(adr,data);)

WriteByteWire.ino	WriteByteTwi.ino	WriteByteBB.ino
2594b 190v	508b 10v	572b 10v

jdn 181022