http://www.didel.com/ info@didel.com

www.didel.com/dixi/DixiDoc.pdf

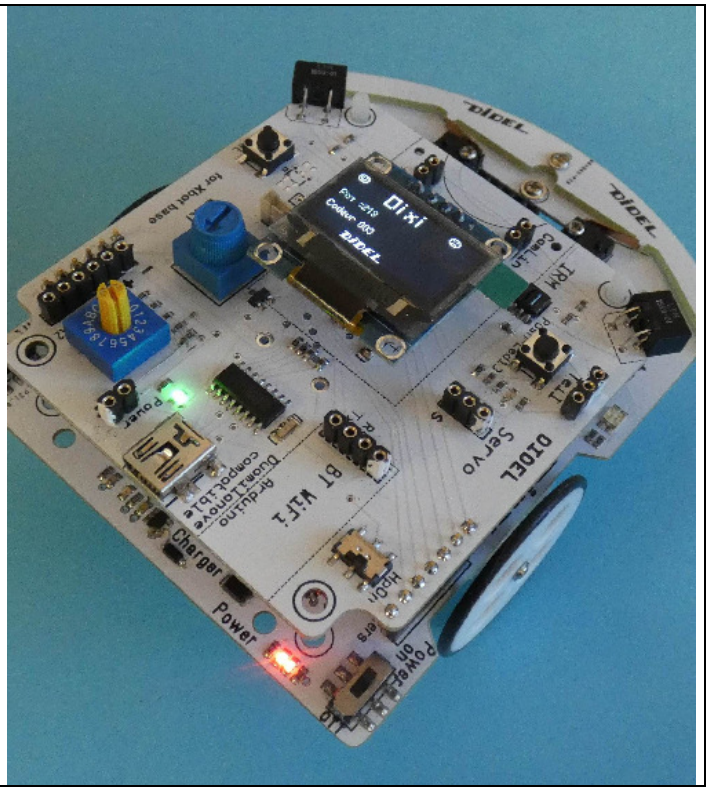
Dixi – carte "Arduino" pour la base Xbot

Note: Les liens aux fichiers pdf ne sont probablement pas cliquables dans votre browser. Liste sous www.didel.com/Dixi

Introduction

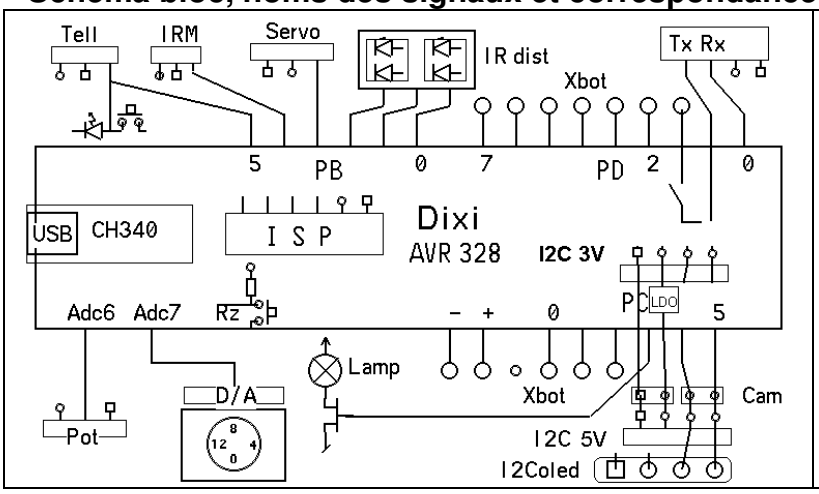
Dixi est une carte Arduino Uno/Duemilanove compatible qui s'enfiche sur la base Xbot et ajoute des fonctionnalités très intéressantes:

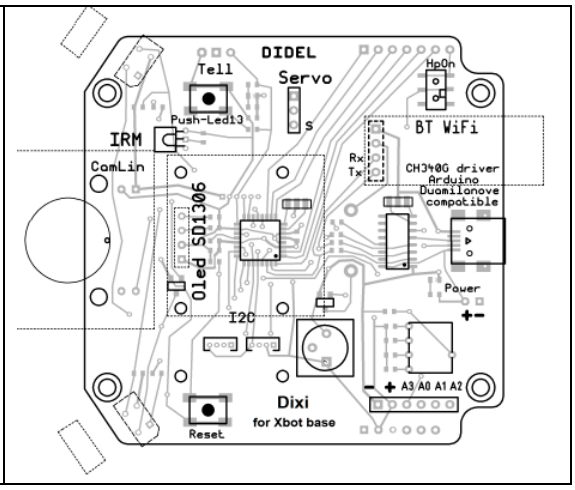
- Poussoir et Led sur la pin 13 comme pour le Diduino2
- Capteurs de distance pour éviter les murs (Dist2Ir)
- Capteur de télécommande infrarouge IRM
- Connecteur analogique en parallèle avec le connecteur avant du Xbot
- Lampe pour suivi de lumière
- Potentiomètre pour saisir une valeur
- Commutateur rotatif pour choisir 16 valeurs ou actions
- Connecteur pour Oled SSD1306
- Connecteur I2C pour ajouter des capteurs
- Connecteur Tell
- Connecteur série pour Bluetooth ou Wifi
- Connecteur pour un servo
- Interrupteur pour le buzzer du Xbot



Le DixiBot est une base Xbot avec la carte Dixi à califourchon.

Schéma bloc, noms des signaux et correspondance Arduino





Pin	Port		Pin	Port		Pin	Port	
0	PD0	Rx Conn	8	PB0	IrG	14	PC0	A0 base+con
1	PD1	**Tx Hp**	9	PB1	IrD	15	PC1	A1 base+con
2	PD2	ObsG	10	PB2	IrLed	16	PC2	A2 base+con
3	PD3	ObsD	11	PB3	Servo	17	PC3	Lamp
4	PD4	RecG	12	PB4	IrModule	18	PC4	I2C SDA
5	PD5	AvG	13	PB5	Led/PushTell	19	PC5	I2C SCL
6	PD6	AvD					An6	Pot
7	PD7	RecD					An7	Rot switch

En jaune, signaux vers la base Xbot

Fichier de définitions et librairies

La librairie `Dixi.h` définit les fonctions de base associées aux pins ci-dessus. Les librairies `OledTwi,h, OledPix.h, DistIr.h, Tell.h, Pfm.h et Inter2.h`, expliqués plus loin, sont préparés dans la base de programme www.didel.com/dixi/RefDixi.zip
On complète cette base avec ses programmes personnels, dont on peut avoir récupéré des parties dans le programme www.didel.com/dixi/Demo.zip
Le capteur IRM et les connecteurs Tell, Servo, Tx/Tx et I2C sont mentionnés plus loin.
Les librairies .documentées sous www.didel.com/dixi/LibX.pdf sont utilisables avec éventuellement une adaptation des définitions.

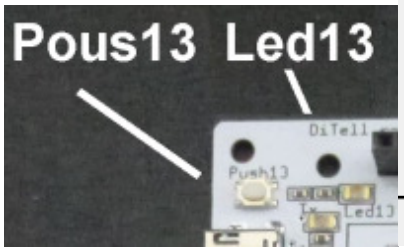
Delais

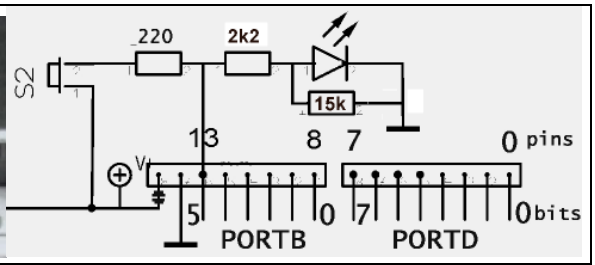
Nous définissons dans Dixi.h deux fonctions de retard qui bloquent comme le `delay ();` d'Arduino, mais ne lancent pas l'interruption. **`Delus(v16);`** est un délai de v16 µS et `DelMs(v16);` remplace `delay (v16);` .

```
void Delus (int uu) {  // approximatif
   for (volatile int i=0; i<uu; i++) {}
}void DelMs (int dm) {
   for (volatile int i=0; i<dm; i++) {
     for (volatile int j=0; j<800; j++) {} // 16MHz
   }
}
```

Led et Poussoir

La Pin13 (PortB bit5) est traditionnellement réservée pour l'aide à la mise au point. Dixi a comme pour la carte Diduino un poussoir en parallèle.





Pédagogiquement, la combinaison permet de présenter et tester la Led, puis le poussoir, puis les contraintes imposées par le passage de l'utilisation de la Led au poussoir. Bon pour former des techniciens et ingénieurs qui doivent maîtriser le matériel. Les fonctions peuvent s'utiliser sans entrer dans ces détails.

En mode ***Led*** la pin 13 est en sortie et une résistance de 220 Ohm neutralise l'effet du poussoir, la sortie est dominante. La Led s'allume si on programme l'état "1".
En mode ***Push***, la pin 13 est en entrée, le poussoir envoie du courant pour allumer la Led, et on peut calculer la tension sur l'entrée dans les 2 modes.
Par défaut, on est en mode ***Led***, Led allumée ou éteinte. On passe en mode ***Push*** automatiquement si on appelle la macro `PousOn;` qui rend l'état du poussoir et revient en mode Led. Documentation complémentaire sur www.didel.com/diduino/Poussoir.pdf
(en anglais: www.didel.com/diduino/PushButton.pdf)

Le fichier de définition Dixi.h contient les définitions de ce premier tableau. Le 2e tableau montre les notations Arduino équivalentes, dont l'exécution est beaucoup plus lente.

C

```
#define bLP 5  // LED/Push bit on PORTB
#define PushMode  bitSet (DDRB,bLP)
#define LedMode   bitClear  (DDRB,bLP)
#define  LEDOn  bitSet  (PORTB,bLP)
#define  LEDOff LedMode; bitClear(PORTB,bLP)
#define  push PushMode; Del60; PINB&(1<<bLP); LedMode
void SetupLed { bitSet (DDRB,bLed); }
```

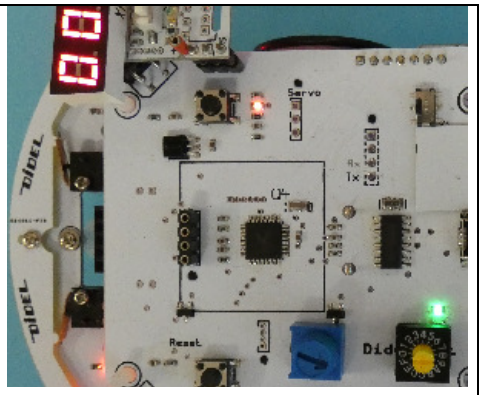
Arduino

```
#define LP 13 // LED/Push pin
#define PushMode  pinMode (LP,INPUT)
#define LedMode   pinMode (LP,OUTPUT)
#define  push    PushMode; delayMicroseconds(60); digitalRead(LP);
#define  LedOn     LedMode; digitalWrite (LP,HIGH)
#define  LedOff    LedMode; digitalWrite (LP,LOW)
void SetupLed { pinMode (Led,OUTPUT); }
```

Tell

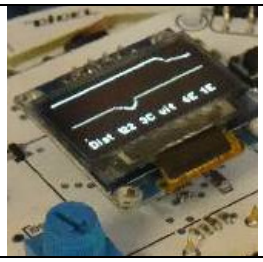
La pin 13 peut commander un affichage qui complète et remplace le terminal série quand la carte est autonome. C'est plus rapide et souvent bien plus pratique que ce terminal. La fonction Tell(v16); envoie 16-bit data sur l'affichage Tell.
Tell a un poussoir pour afficher en hexadécimal ou décimal sans modifier le programme ! L'affichage en décimal est limité à 9999. En affichage de deux bytes, les points décimaux codent les centaines. Voir www.didel.com/diduino/DiTell.pdf
Dixi a un pot en A6, voila un programme de test simple:

```
#include "Tell"
void setup() { setupTell(); }
void loop() {
   Tell (analogRead(A6)); delay (100);
}
```

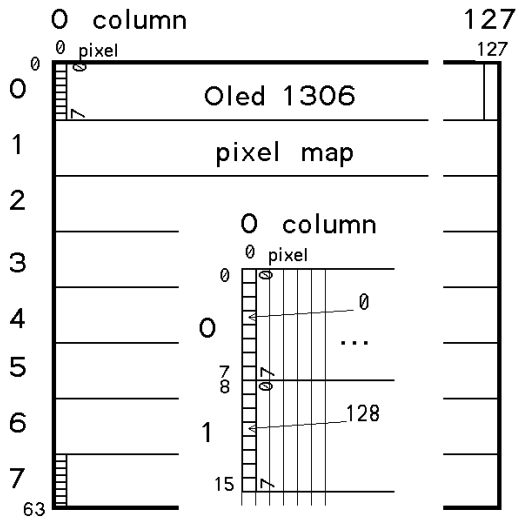


Oled

Le module Oled SSD1306 est standard sur les cartes Dixi. La doc est sous www.didel.com/Oled1306.pdf avec en particulier comment dessiner des Sprite(); qui ne sont pas mentionnés ci-dessous.



```
//TestOled
#include "Oled.h"
void setup(){
  SetupOled ();
}
byte var = 33; // -- variable globale
void loop(){
  LiCol(0,0);  Sprite(smile);  // top
  LiCol(4,0);  Dec8(var);
  LiCol(4,30); Hex8(var);
  (var++);  // on compte  var++;     // on
compte
  LiCol(7,118); Sprite(sad); // bas d'écran
  delay(500);
}
```



Les fonctions à disposition sont les suivantes:

Clear();	. Efface l'écran et place le pointeur en (0,0).
Licol(li,co);	. Place le pointeur ligne li (0 à 7) et en colonne co (0 à 127, mais il faut la place pour finir)
Bin8(v);	. Affiche la variable 8 bits (byte, char, int8_t, uint8_t) en binaire
Hex8(v);	. Affiche la variable 8 bits (byte, char, int8_t, uint8_t) en binaire
Hex16(v);	. Affiche la variable 16 bits (int, int16_t uint6_t) en binaire
Dec8(v);	. Affiche la variable 8 bits (byte, char, int8_t, uint8_t) en binaire
Dec9999(v);	. 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.

En double taille (utilise la ligne en-dessus, donc lignes 1..7) on a:

Big(); (ou BigCar) **BigBin8(); BigHex8(); BigHex16(); BigDec8(); BigDec9999();**

Dot (x,y); DDot (x,y); Origine en haut à gauche.

L'écran 32x128 n'a que les 4 lignes 4,5,6,7. Il faut l'initialiser dans le setup avec l'instruction **DoubleH();** Sur l'affichage 64x128, cette instruction étale les lignes 4 à 7 sur les lignes écran 0 à 7 et donne une bonne lisibilité (et plus de rapidité d'écriture que le mode Big, qui dédouble péniblement des pixels.

Moteurs du Xbot

Les moteur sont câblés de façon à pouvoir programmer en Arduino pour faire avancer le robot en PWM. Plusieurs librairies pour des shields moteur sont utilisables.
Exemple simple:

```
void setup() { } // Initialisation par défaut
byte PwmValue;
void loop(){
  digitalWrite (4,LOW);
  analogWite (5,pwmValue);
  analogWite (6,pwmValue);
  digitalWrite (4,LOW)
}
```

TestMotOled.ino affiche en plus

Les primitives dans Dixi.h traitent chaque moteur séparément :
AvG; AvD; pour avancer
RecD; RecG; pour reculer
FreeG; FreeD; pour s'arrêter
A noter que FreeG,D mettent le moteur en roue libre, les deux sorties à zéro.
BlockG; BlockD; mettent les deux sorties à un, les leds rouges et vertes sont donc allumées, le moteur est court-circuité et se freine plus vite. Dans ce mode, on a plus de peine pour tourner le moteur à la main.

Programme de test en tout-ou rien

Le robot doit être positionné avant de démarrer son programme. Testons l'utilité du poussoir et la subtilité de la commutation Led/Poussoir.
Le programme attend que l'on presse, et clignote pour le signaler.
Si on presse, il y a une attente pour relâcher le poussoir avant de démarrer les mouvements.
Notez l'utilité de donner un nom aux délais pour pouvoir les changer à un seul endroit pour ralentir ou accéplérer la séquence.
Mettre plusieurs instructions par lignes n'est pas recommandé dans les programmes C complexes, mais ici c'est tellement plus clair !
A l'arrêt, le moteur est libre. Avec Block au lieu de Free. il est bloqué. Le moteur et l'ampli moteur ne sont pas assez bon pour que la différence soit bien marquée.

```
//TestMotSimple.ino ok
//Clignote au reset. Presser pour démarrer
#include "Dixi.h"
void setup() {
  SetupDixi();
}
 #define Bdel DelMs(300) // demi-période clignotement
void loop() {
  PushMode;
  while (!PushOn) {
    LedMode; LedOn; Bdel; LedOff; Bdel;
    PushMode;
  }
  #define D DelMs(2000)
  D;
  byte var=3;
  while(var--){  // on répète 3 fois
    AvD; AvG;  D;     // Avance
    FreeG; FreeD; D;
    AvD;   D;          //TourneG"
    FreeG; FreeD; D;
    AvG;  D;          // TourneD
    FreeG; FreeD; D;
    RecD; RecG;  D;  // Recule
    FreeG; FreeD; D;
  }
  for(;;);
}
```

PFM

Le PFM est trop peu connu. Il est géré par l'interruption du timer2 et permet des vitesses très lente (avantages sous didel.com/PFMversusPWMforRobots.pdf). Les vitesses vont de -100 à +100 les valeurs plus grande sont saturées à 100 ou -100. (anciennement de -80 à +80). Si la vitesse passe de 127 (saturé à 100) à 128 (négatif, saturé à -100), il y a une brutale inversion de vitesse.
Les macros Free ou Block n'ont pas d'effet, l'interruption remet tout de suite la vitesse selon les variables globales pfmG et pfmD. Voir les exemples de programme.
Une librairie Vit.h sera proposée. Elle propose 16 vitesses logiques et le passage d'une vitesse à l'autre est progressif (accélération constante.

Moustaches du Xbot

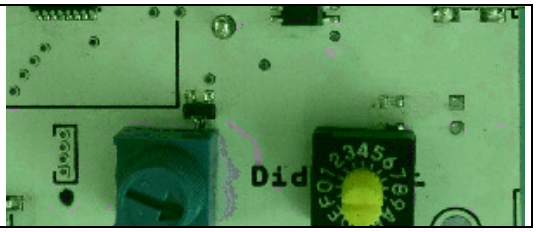
Les fonctions bool=ObsG(); et bool=ObsD(); lisent les moustaches (détecteur d'obstacle). Il peut y avoir des rebonds de contact.

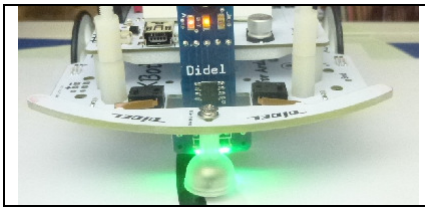
Modules Xsens

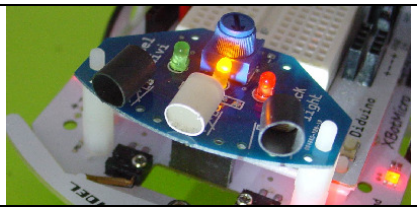
Voir la documentation listée dans https://www.didel.com/prof/Xbot.html

Entrées A0 A1 A2 (A3)

Un connecteur à 6 broches reçoit le 3.7V du Lipo de la base Xbot et transmet le 5V pour charger ce LiPo. Les signaux A0, A1, A2 de la carte Dixi sont transmis au connecteur avant du Xbot, voir www.didel.com/xbot/Xsensors.pdf. Ces signaux, complété par A3, sont disponibles avec le Gnd et Vcc sur un connecteur femelle.





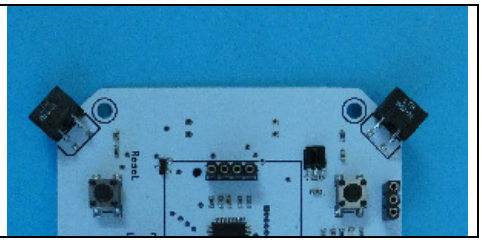


A3 est partagé avec la commande de la lampe.

Ampoule (Lamp)
La lampe se commande avec les macros LampOn, LampOff. Si on veut utiliser la pin A3, il faut dévisser l'ampoule commandée pas un transistor à forte impédance d'entrée.

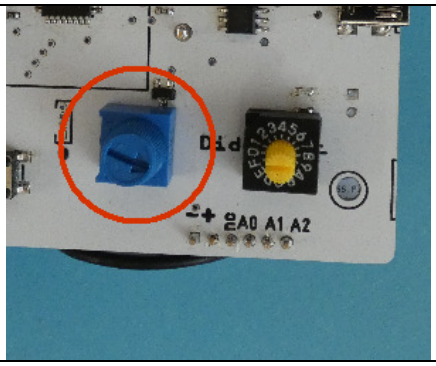
Capteurs de distance (DistIr)

Le capteur de distance Dist2Ir est sur la carte, avec les mêmes spécifications que le module :
https://www.didel.com/prof/Xdist2Ir.pdf
La librairie DistIr.h fournit une fonction bloquante GetDistIr(); et une machine d'état DoDistIr(); appelée par interruption.



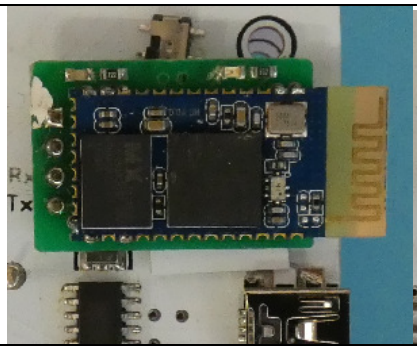
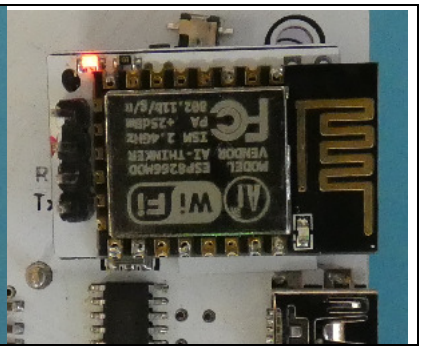
Potentiomètre et commutateur rotatif

Un potentiomètre est câblé sur l'entrée AN6 de l'AVR 328; cette pin n'est pas accessible sur les cartes Arduino. Le potentiomètre se lit avec `val10bits= analogRead(A6);` ou avec la fonction `v8=GetPot();` qui donne une valeur bien assez précise.
Le commutateur rotatif à 16 position utilise AN7 via un convertisseur D/A. Les 16 positions correspondent à 16 tensions et la fonction GetRotSw(); rend 16 valeurs à utiliser dans un switch/case.



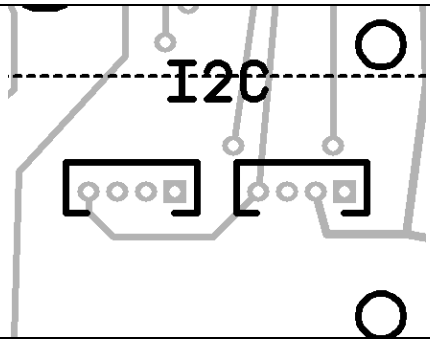
Connecteur série UART/Bt/Wifi

Les signaux Rx et Tx de l'AVR328, utilisés lors de la programmation via USB, son disponible sur un connecteur à 4 compatible avec des modules Bt et Wifi.
Tx est aussi envoyé via un interrupteur vers le haut-parleur de la base Xbot.
Doc selon demande.

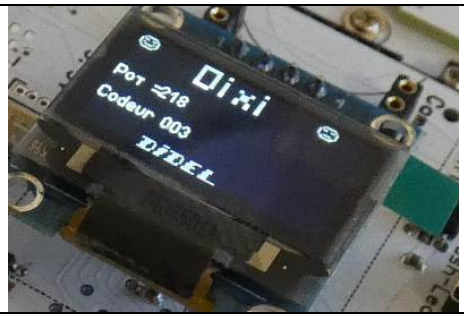



Connecteurs I2C

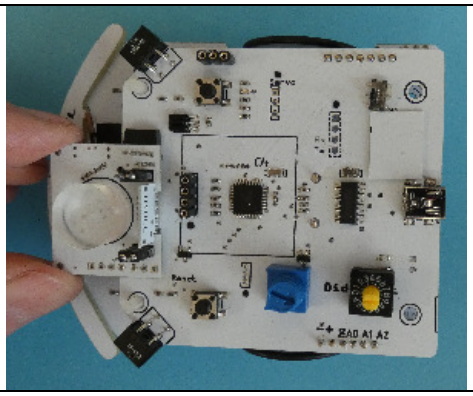
Les capteurs I2C existant sont annoncés 5V ou 3.3V, mais les modules existant sont tous compatible Arduino, donc 5V.
Le brochage est Gnd 5V Sda Sck. Led harness Molex compatible ont les couleurs rouge/gnd, noir/+5V, jaune/Sda, vert/Sck.
Pour faciliter les projets éventuels, deux connecteurs sont prévus. Un connecteur est soudé dessous, où il y a de la place pour caser des modules définitifs.



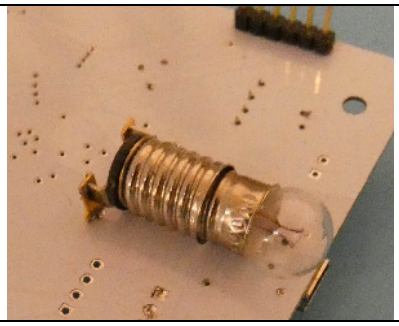
Connecteur I2C pour Oled SSD1306

Un connecteur pour Oled.html 64x128, type SSD1306 est placé au centre de la carte. Le module de 32x128 est compatible.	

Connecteur I2C pour Caméra linéaire

La caméra linlaire en développement (96 pixels) sera un exclave I2C avec un Oled local pour afficher la courbe d'intensité lumineuse et des paramètres.	

Lampe sous le PCB

Une application motivante pour un groupe qui a des Dixibots est de programmer des chenilles processionaires. Une ampoule a été prévue à l'arrière, pour avoir une lumière plus omnidirectionnelle qu'une Led. Il faut utiliser des ampoules 5V 50mA, et pas des ampoules de lampe de poche qui consomment trop.	

Connecteur servo ou RGB strip

Un connecteur avec alimentation et un signal connecté à la pin Arduino PB3 peut être utilisé pour lire un signal en tout-ou-rien, ou écrire une valeur sur une Led, un servo, un strip de leds RGB, etc..	

Récepteur IR (IRmodule)

Un récepteur IR compatible avec toutes les télécommandes ex lié à la pin PB4. Une solution simple compatible avec toutes les télécommandes est bien documentée : www.didel.com/TelecommandeIrSimple.pdf