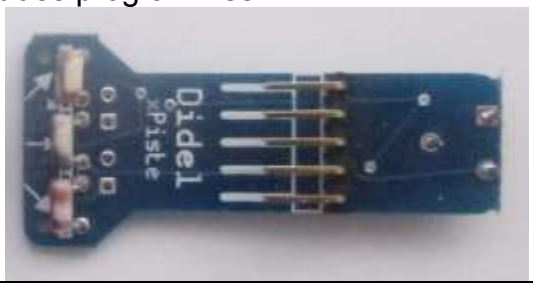
Suivi de piste avec le Xbot/Diduibot

1. Introduction

Le document www.didel.com/diduino/SuiviPisteEt Lum.pdf donne les explications générales sur ce problème intéressant de suivi de lumière. Il présente la problématique et donne les schémas électroniques, rappelle les fréquences lumineuse utilisées et évoque des objectifs à suivre.
Le présent document se concentre sur la mise en oeuvre du capteur optimisé pour le Xbot.

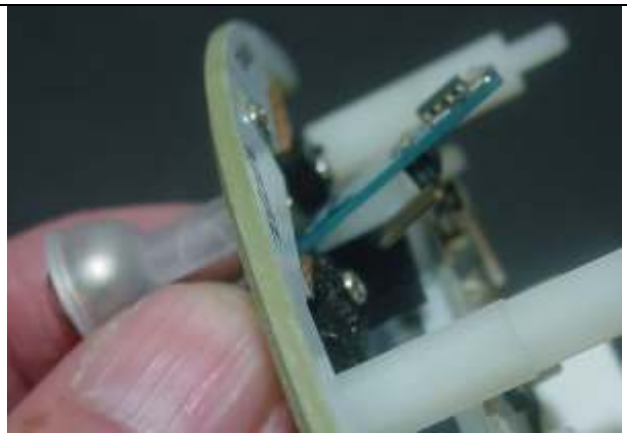
2. Circuit Xpiste

Le circuit a 3 photocapteurs et des Leds de couleur, pour montrer approximativement ce que les capteurs voient, et faciliter la mise au point des programmes.



3. Connecteur

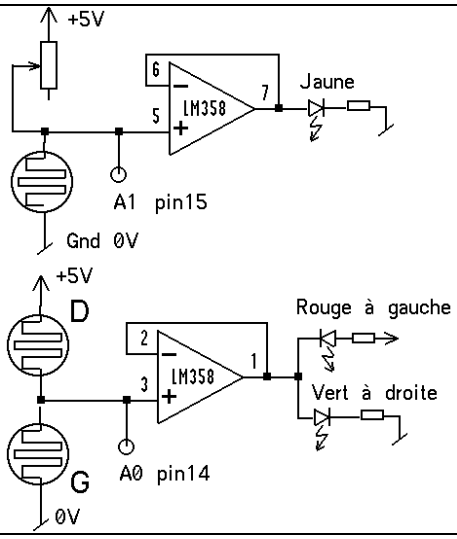
Le connecteur à l'avant du xBot comporte 5 broches, soit l'alimentation, et 3 signaux. Les signaux A0, A1, A2 sont câblés entre la base Xbot et la carte Diduino. Dans le cas du module piste il faut câbler A0 (fil jaune de la photo) et A1 (fil supplémentaire à droite).



L'insertion du Xpiste se fait par dessous.

4. Schéma

Le circuit a 2 parties indépendantes. Les deux LDR de chaque côté, distantes de 12mm, sont branchées en série. La tension est de ~2.5V lorsque l'éclairage est symétrique, quelque soit l'intensité lumineuse. La LDR centrale donne une tension qui dépend de l'intensité lumineuse, et que l'on peut ajuster avec un potentiomètre, pour que la tension moyenne soit proche de 2.5V.
Des ampli-op commandent des Leds qui donnent une idée de la tension transmise au microcontrôleur: vert tension >2V, rouge <3V. C'est évidemment peu précis, mais cela montre bien si les signaux sont bons. Une piste bien contrastée pour notre oeil ne l'est pas nécessairement pour les capteurs LDR.



Le module Piste s'insère depuis le dessous du robot. Deux leds verte (pour une bonne efficacité lumineuse) éclairent la piste.
Les capteurs ont un angle de vue que l'on peut adapter en les entourant d'un cylindre plus ou moins long.

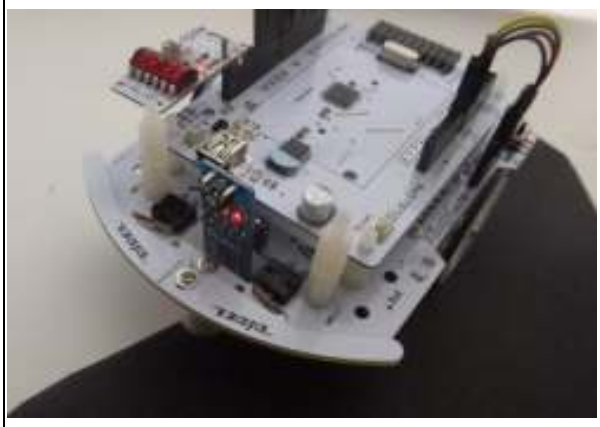
5. Valeur des capteurs

La tension sur les lignes A0 et A1 dépends de la nature du réflecteur et de la luminosité. Il faut la mesurer avant chaque expérience pour connaître les limites et les valeurs dans la zone de réglage, et adapter le logiciel. Le programme suivant est le plus simple à utiliser. Il se trouve avec les autres programmes sous www.didel.com/xbotsens/Piste.zip

```
//TestA0A1SerialPrint.ino| durée 110 us
void setup() {
Serial.begin(9600);
}
void loop() {
  Serial.print ("Différence  ");
  Serial.print (analogRead (A0)) ;
  Serial.print ("  Centre  ");
  Serial.println (analogRead (A1)) ;
  delay (1000);
}
```

6. Suivi d'un bord

Suivre le bord d'une zone ou d'une piste large est plus facile et plus intéressant pédagogiquement car on peut observer l'effet des paramètres sans perdre le bord.
`LimiteNB` est la moyenne entre le niveau clair et sobre mesuré, après division par 4.
`Vitesse` est une vitesse de consigne, moyenne entre la vitesse des deux roues.
L'écart et les vitesses sont calculées au plus simple. En changeant la formule pour réagir plus fortement, il faut faire attention de ne pas dépasser les valeurs acceptables : les vitesses doivent être entre 0 et 255.

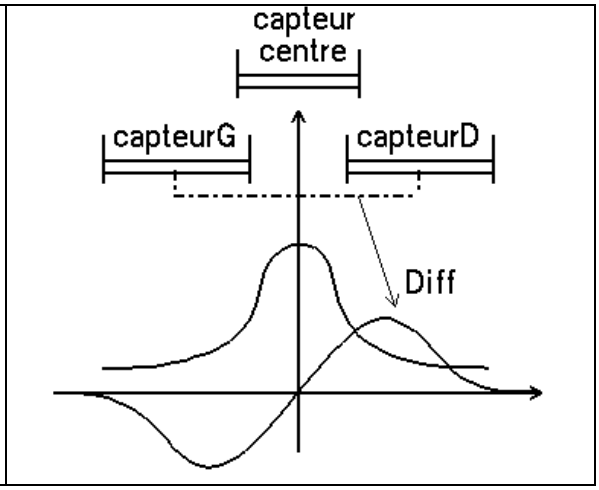


```
// SuitBordMin.ino  - n'utilise pas Xbot.h
// En continu, on modifie le pwm des deux moteurs selon l'écart.
void setup() {
   DDRD  |=  0b11110000 ; // force les 1 (motors out)
   DDRD  &=  0b11110011 ; // force les 0 (moustaches in)
   PORTD &= 0x0F ; // --> 0 Mot stop
}
#define PinAvG 5
#define PinAvD 6
#define Centre A1    // entrée analo
#define LimiteNB 80  // Selon mesure préalable avec Serial.print
#define Vitesse 20   // max127
int caPiste, ecart;
int vitG, vitD;
void loop()  {
  caPiste = ((analogRead (Centre))/4); //  valeur 53--110
  ecart = caPiste - LimiteNB;
  vitG =  Vitesse+ecart; if (vitG <0) {vitG=0;}
  vitD =  Vitesse-ecart; if (vitD <0) {vitD=0;}
   analogWrite (PinAvG, vitG); // 0-256
   analogWrite (PinAvD, vitD); // 256-0
}
```

Le programme utilisant Xbot.h et incluant l'option affichage série est dans le dossier `SuitBord` du zip

7 Suivre une piste

Suivre une piste avec 2 capteurs est difficile à bien programmer. Il faut rester dans la zone ou les capteurs donnent une information utilisable. En dehors de la piste, on ne sait pas comment corriger.
La LDR centrale dit si on est sur la piste, mais ne permet pas de savoir dans quel sens on la quitte. Il faut travailler sur la différence, qui n'existe plus si on s'écarte trop.
Pour la mise au point, on affichera avec des Serial.print les capteurs et les consignes moteur, sans bouger le robot. Avec l'avance moteur, le comportement est plus difficile à maîtriser.

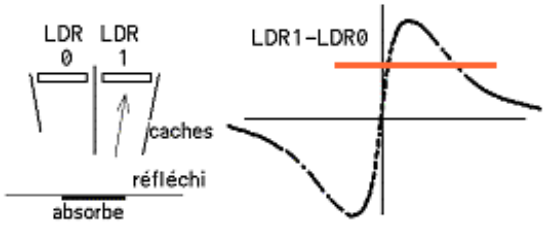


Choisir la piste n'est pas facile. Elle doit bien contraster avec le sol, et ce que voit notre œil n'est pas ce que voit le capteur LDR. Elle doit être assez large (2 fois le diamètre d'une LDR), un trait de marker feutre ne se voit pas bien. De la bande d'électricien isolante (tissu et non pas plastique) n'as pas la souplesse souhaitée, mais offre un bon contraste.

8 Exemples de programme

Les programmes se trouvent sous www.didel.com/xbot/Piste.zip , Leur bon fonctionnement n'est pas garanti et il peut y avoir des différences avec les programmes imprimés ici.! Les paramètres doivent être ajustés et l'algorithme (formule) est rarement le meilleur. On utilise que les deux LDR extérieures, la tension lue est la différence relative, ce qui est très intéressant car c'est indépendant de l'ambiance lumineuse.

Il faut bien mesurer ce que voit le
On place une règle tranparente sur la piste et on lit la valeur tous les 2mm de déplacement. La valeur lue est 512 si les deux LDR reçoivent la même intensité lumineuse (tension moitié, 2.5V).



On comprend que l'asservissement doit se faire entre les 2 bosses. Si on dépasse la bosse, la correction doit se faire en sens inverse et la détection du passage par dessus la bosse n'est pas évidente.
Pour une première étape dans la programmation, il faut trouver une piste qui donne une bonne distance entre les bosses avec des bonnes différences de valeur. L'algorithme est théoriquement simple: Pour une valeur proche de 512, le robot va en ligne droite. Si la valeur augmente ou diminue, il faut corriger dans le bon sens, et il y a inévitablement des oscillation si on ne tient pas compte de la vitesse du changement..
Le problème est donc de corriger suffisamment en évitant le développement d'oscillations.

Avant de passer aux programmes suivants, il faut avoir joué avec les moustaches et appris à faire des programmes propres, utilisant le fichier de définition `xbot.h` (www.didel.com/xbot/EvitObstacle.pdf)

On peut essayer en tout-ou-rien, en bloquant un moteur pendant un temps proportionnel à l'écart.

Pour la mise au point, c'est utile d'afficher la valeur utilisée pour l'asservissement.
Pour ne pas afficher trop souvent, le code

```
// SuitPisteToutOuRien.ino
// Toutes les 100 ms, on décide s'il faut freiner une roue
// si l'erreur est trop grande – a réfléchir
#include "XBotDef.h"

void setup() {
   XBotSetup();
}
int caPiste;
byte cc;
void loop() {
```

```
 if (cc++>20) {cc=0; .. }
```
n'imprime qu'une fois tous les 20 passages.

Le robot est rapide, il faut que la piste soit longue. Pour ralentir, on peut pulser l'avance:
```
Avance; delay (2);Stop; delay(20);
```

```
 Avance; delay (100);
 caPiste = ((analogRead (A0))/4)-128;
 if (cc++>20) {cc=0; Serial.println (caPiste);}
  if (caPiste>10)  { //tourne selon erreur)
    TourneD; delay (caPiste/2);
  }
  if (caPiste<-10)  {
    TourneG; delay (-caPiste/2);
  }
}
```

L'astuce de ce programme est de donner un PWM égal à la valeur analogique centre piste divisée par 4. Mais l'un des moteur a la valeur complémentaire. Il ralentit quand l'autre accélère.
Le réglage doit naturellement être amélioré avec éventuellement une zone morte et des facteurs correctifs adéquats.
En cas de perte de la piste, la retrouver est hasardeux.
(programme non testé)

```
// SuitPiste.ino
// En continu, on modifie le pwm des deux moteurs selon l'écart.
#include "XBotDef.h"
void setup() {
  XBotSetup ();
}
#define PinAvG 5
#define PinAvD 6
int caPiste;
void loop()  {
 caPiste = ((analogRead (A0))/4); // 0-256
 analogWrite (PinAvG, caPiste); // 0-256
 analogWrite (PinAvD, 256-caPiste); // 256-0
}
```

Suivi d'un bord

Suivre le bord d'une piste large est plus facile et plus intéressant pédagogiquement car on peut observer l'effet des paramètres sans perdre le bord.

```
// SuitPiste.ino
// En continu, on modifie le pwm des deux moteurs selon l'Ã©cart.
#include "XBotDef.h"
//#define Serie
void setup() {
  XBotSetup ();
#ifdef Serie
  Serial.begin(9600);
#endif
}
#define PinAvG 5
#define PinAvD 6
#define Centre A1    // entrÃ©e analo
#define LimiteNB 80  // Selon mesure prÃ©alable avec Terminal
#define Vitesse 20   // max127
int caPiste, ecart;
int vitG, vitD;
void loop()  {
 caPiste = ((analogRead (Centre))/4); // 53--110
 ecart = caPiste - LimiteNB;
 vitG =  Vitesse+ecart; if (vitG <0) {vitG=0;}
 vitD =  Vitesse-ecart; if (vitD <0) {vitD=0;}
   analogWrite (PinAvG, vitG); // 0-256
   analogWrite (PinAvD, vitD); // 256-0
#ifdef Serie
  delay (1000);
  Serial.print ("piste "); Serial.print (caPiste);
  Serial.print ("  ecart "); Serial.print (ecart);
  Serial.print ("  vitG "); Serial.print (vitG);
  Serial.print ("  vitD "); Serial.println (vitD);
#endif
}
```

jdn 140213/150710