Luxmealex
Un photovore

1. Introduction

Afin de réaliser un petit robot pour apprendre les techniques de base, j'ai décidé de me lancer dans un projet de type petit mobile cherchant la lumière. Il n'a pas besoin d'être compliqué, et même plus il sera simple, mieux cela sera.
Ce robot fera appel aux trois composants majeurs de la robotique : de la mécanique, de l'électronique et un peu d'informatique.

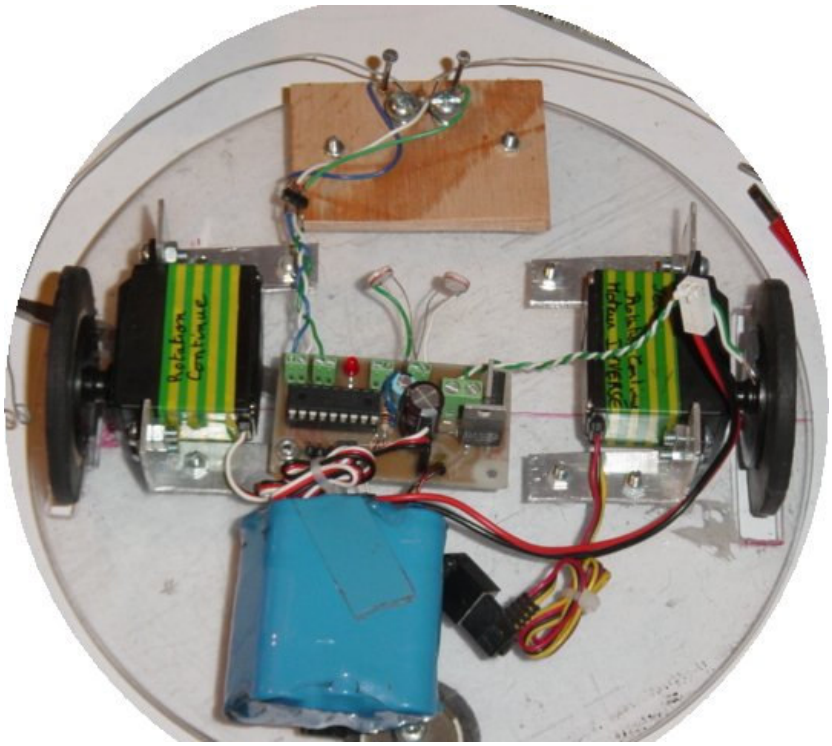
1.1 Prérequis

Vous aurez besoin de

1. savoir scier, limer, percer, bricoler
2. savoir réaliser ou faire réaliser un circuit imprimé ou utiliser une plaque à trou type veroboard,
3. savoir souder les composants et mettre les diodes/capa dans le bon sens
4. avoir un programmateur de PIC.
5. moins de 50€, 100 avec tout l'outillage.
6. savoir utiliser internet pour chercher les réponses aux questions que l'on a. (www.google.fr, le site des fribottes : http://fribotte.free.fr, un forum http://www.planete-sciences.org/forums et sa FAQ : http://tcremel.free.fr/doc/wiki.htm)

2. Mécanique.

Pour faire avancer ce module sans qu'il reste accroché, j'ai opté pour une plaque en plexiglas ronde sur laquelle tout viendra s'attacher. remplaçable par n'importe quelle planche légère

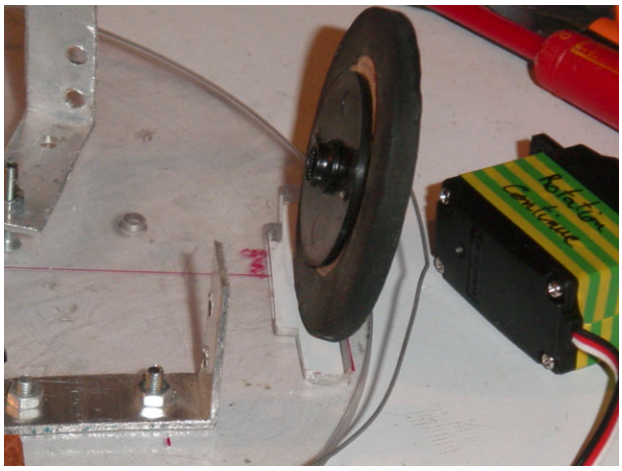


Comme base de motorisation, je suis parti sur des servomoteurs (Futaba S3003) à rotation continue. (il faut modifier un servo standard cf http://tcremel.free.fr/doc/bidouille_servo.pdf).

Pour attacher ces servomoteurs, j'ai utilisé de la cornière en aluminium que l'on trouve dans tout magasin de bricolage 30mm*20mm. J'ai coupé des petits morceaux et j'ai réalisé le perçage en face des trous du servomoteur. Vous pouvez remplacer la cornière par un bloc de bois plein.

Dans une plaque de contreplaqué de 5mm, j'ai découpé à la scie cloche une roue de 50mm. Si vous n'avez pas de scie cloche, il vous faut couper un cercle à la scie et à la lime en prévoyant un trou au centre de la roue pour visser sur le servomoteur. J'ai coupé une chambre à air de vélo sur 30mm et en l'étirant au maximum j'ai fait passer la roue dans la chambre. Ceci donne une forte adhérence à la roue.

Ensuite j'ai vissé le palonnier du servomoteur bien centré sur cette roue et enfin la roue sur le servomoteur.



Dans la plaque de base, j'ai placé les centres des roues sur un diamètre et marqué la position des équerres de fixation.

J'ai également creusé une fente dans la plaque pour laisser passer la roue. Ceci évitera à la roue de se prendre les pieds dans le tapis.

Une fois les servomoteurs fixés sur la base, il faut prévoir une attache pour les batteries. J'ai mis du velcro.

Il faut aussi prévoir une attache pour la plaque électronique de commande.

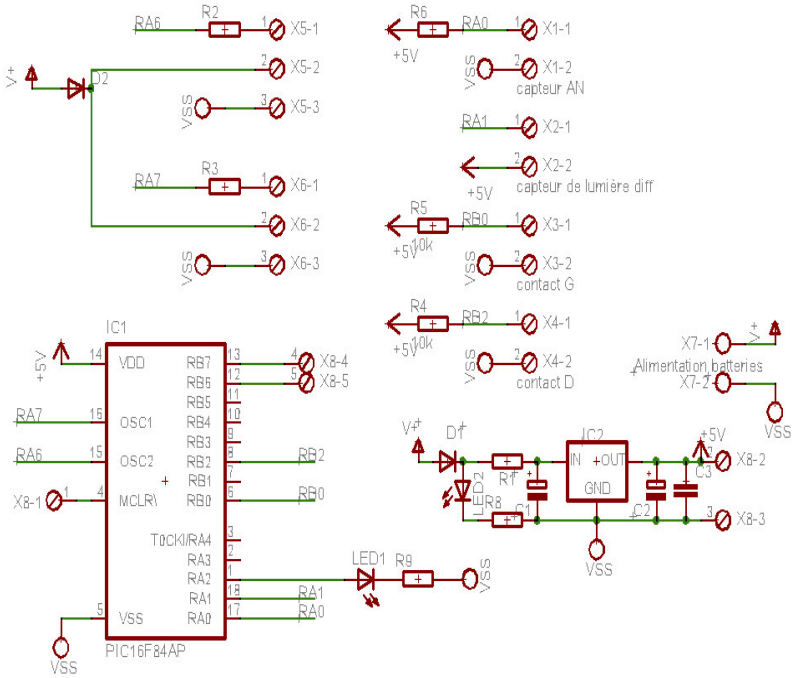
Enfin, j'ai réalisé des moustaches de contact en entortillant un pauvre fil de fer autour d'une vis et mis un clou pour faire le retour d'information. Ceci est a améliorer, mais pour l'instant c'est efficace…
Sous la batterie, j'ai ajouté un patin (un bouton de porte en laiton, mais les patins en téflon qu'on utilise pour faire glisser les meubles sont parfaits).

2.1 Liste des courses

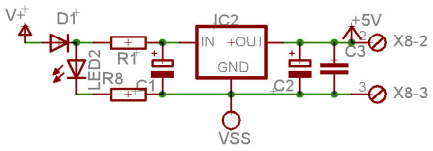
1. un peu de matière de 20cm de diamètre 4-5 mm d'épaisseur.
2. fil de fer,
3. vis et écrou de 3mm de diamètre
4. petite vis à bois
5. scie, lime, tournevis, fer à souder, perceuse avec foret de 3.5mm
6. velcro
7. 2 servo moteurs modifiés (donc avec 4 résistances de 2.2k)
8. de la cornière 20*30 ou un bloc de bois
9. 6 batteries NiCd
10. un bouton de porte plat ou un patin en téflon…bref un trainard.

3. Electronique.

Une fois la mécanique réalisée, il faut passer à l'électronique.
Le montage est assez simple et les fonctions de base seront réalisées par un PIC16F88.
On met deux photorésistances entre le 0V et le 5V. Au milieu des deux résistances, si la lumière est égale sur les deux, on trouve 2.5V. Si la lumière baisse sur une photorésistance, sa résistance augmente et la tension aux bornes de cette résistance augmente. En mesurant la tension via la patte RA1 on peut déterminer de quel côté on doit aller et commander les moteurs en conséquence.

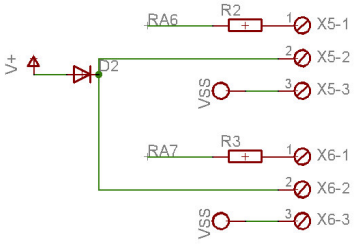


- L'alimentation du PIC est réalisée par un régulateur 5V 78L05. pour alimenter le 7805, j'ai ajouté avant le régulateur un circuit D-RC (D1 R1C1). La diode D1 mise sur l'arrivée batterie limite les erreurs en cas d'inversion de sens de la batterie. Elle élimine (grâce à la Capa du R1 C1 qui suit) également les baisses de tension. Ensuite, le filtre passe-bas R1 C1 limite les pics de tension.



  Les capa C2 et C3 servent à filtrer la sortie. C3 sert surtout a filtrer l'alimentation du PIC
  La LED et sa résistance (LED2-R8) servent d'indicateur de tension batterie.
- La recherche de la lumière se fera en mettant deux photo-résistances en pont diviseur. A éclairage équivalent, on doit avoir la même résistance et donc une tension moyenne de 2.5V au milieu des deux résistances. Ensuite, le côté le plus lumineux sera moins résistant et donc la tension va varier entre 0 et 5V. En mesurant cette valeur on saura de quel côté est la lumière. Il faut donc entrer cette information sur une des pattes réalisant la Conversion Analogique Numérique du PIC

- La détection de contact se fera sur réalisée en faisant toucher à une des pattes la masse. Sans contact une résistance de 10k viendra apporter le 5V sur cette patte. On pourra utiliser la résistance de pull up intégrée dans le pic en positionnant le bon bit (RBPU).
- Une LED avec sa résistance est branchée directement sur une des pattes du PIC



- La commande des servomoteurs sera réalisée directement sur les pattes du PIC avec une résistance de protection de 1k pour le cas où on pourrait avoir un retour. (R2 et R3)
- Les servomoteurs sont alimentés directement depuis la batterie 7.2V avec une diode de protection (D2) permettant de baisser la tension dans le servomoteur à 6.5V Cette tension un peu élevée peut-être limitée par l'utilisation d'une autre diode en série.

Le schéma sous Eagle (www.cadsoft.de) se trouve dans
http://tcremel.free.fr/doc/luxmealex_eagle.zip

3.1 Liste des courses

1. soit du veroboard, soit une plaque imprimée – gravée - percée
2. 2 LEDs
3. 2 résistances de 1k 1/4W (pour la commande des servo en tant que protection)
4. 2 résistances 1k (pour les LEDs).
5. 2 résistances de 10k
6. 1 78L05 avec deux capa chimiques de 10µF au moins (plus c'est mieux) et une petite capa genre 100nF
7. 2 diodes genre 1N4001 passant respectivement 0.1A (côté pic) et 1A (côté moteurs) sans griller
8. 2 photorésistances.
9. des borniers ou des fils soudés en direct.
10. un interrupteur pour couper le courant de la batterie. Sinon il faudra débrancher la batterie pour l'arrêter.
11. 1 support pour le pic 18 pattes et le pic lui même. 16F88

4. Informatique

Dans les fonctionnalités utilisées, on aura : une conversion analogique - numérique (CAN), l'utilisation d'un timer pour commander les moteurs, l'allumage d'une LED et la lecture des capteurs de choc.
Il faut un programmateur de PIC ou avoir accès a ce genre d'outil. (chercher kudelsko, ou JDM ou programmateur de pic…)
Le code est réalisé avec boostC. Par contre les délais cours ne sont pas acceptés par le compilateur, je lui ai donc indiqué que mon PIC tournait à 32 MHz au lieu de 8 puis j'ai donné des temps qui sont donc multipliés par 4. 250ms demandé devient en fait 1s.
Le programme se décompose en 3 parties. Une partie d'initialisation, une partie de choix de la direction (fonction main) et une partie de gestion du timer0 pour générer les signaux des servomoteurs.
Dans le timer0 on utilisera des variables globales pour régler la position des servos (sensgauche, sensdroit) ou la luminosité de la LED (flashspeed).

4.1 Initialisation

Lors de l'initialisation, on commence par passer l'horloge à 8MHz.
Ensuite on configure le convertisseur CAN.
Ensuite on configure les TRIS pour choisir si les pattes sont en entrée ou en sortie.
On fait clignoter la LED pour signifier que le robot viens de démarrer. (a noter que l'on ne pourra plus faire clignoter la led ultérieurement puisqu'elle est contrôlée par le timer0
Enfin, on configure le timer0 et on le met en route.

4.2 Interruption pour timer0

Chaque fois que timer0 arrive à 255 il génère une interruption. Dans cette interruption, on incrémente un compteur (mytimer). (un incrément toute les 78µs)
Ce compteur va retomber à 0 toutes les 20ms (temps de commande pour les servo moteurs). Dès que le compteur est sous la valeur demandée, on met la sortie de commande du servo à 1, Dès qu'il dépasse, on la remet à 0. Ainsi le servo moteur se contrôle avec des valeurs de 13 (1ms) à 25 (2ms)
On réalise le même raisonnement pour la LED pour choisir sa luminosité.

4.3 Fonction Main

Le 'main' commence par réaliser la conversion CAN. Entre 2 conversions, on impose un petit délai.
Ensuite on utilise la valeur lue lors de la conversion pour choisir la direction.
Si on est environ au milieu (même luminosité à gauche et a droite), on va tout droit, sinon, on choisit de tourner vers la lumière.
Enfin, on lit sur le PORTB (masqué par 5 pour garder RB2 et RB0 les détecteurs de contacts). Et s'il y a contact, on force un recul et une rotation du côté opposé au contact. Tant que le PORTB n'est pas en situation où il n'y a pas contact, on recule et on s'oriente. Je n'ai pas traité le cas où on touche avec les deux moustaches.
Mais il y a plein d'autres cas que je n'ai pas traité…

4.4 Liste des courses

1. un programmateur de PIC

2. un soft pour utiliser le programmateur (winpicpr / icprog …)
3. Un compilateur C boostC (http://www.picant.com/c2c/download.html)

5. Evolutions

Afin de ne pas laisser ce robot fini tout à fait, libre à vous de modifier le code et l'électronique.
Je vous propose de :

- Mettre un récepteur Infrarouge pour le commander par une télécommande (code RC5)
- Adjoindre des capteurs de distances style GP2D12 ou ultrason pour éviter les chocs.

Tout élément permettant de jouer avec sera pour vous une source de joies et de richesses infinies.

6. ANNEXE A Code informatique

```
#pragma CLOCK_FREQ 32000000
  // defini comme 4 fois plus rapide que la réalité pour pouvoir accéder a delay_us
  // les delays sont donc 4 fois plus long que ceux indiqués dans la ligne de code.
#include <p16f88.h>
#include <boostc.h>

#define VERSION 0x01
#define GRANDELUMINOSITE 20
#define TOURNESENESTRE 25 //25 = 2ms
#define TOURNEDEXTRE 13 // 13 = 1ms
char value_a1=128;
char restart_CAN=0;
char sens_gauche;
char sens_droit;
char mytimer;
char flashspeed;
char mytimerplus;
```

```
void interrupt()
{
        if ( intcon & 0x04  )              // si l'interruption TIMER 0 a lieu
                {                           // alors
                tmr0 = 179;  // soit 77 impulsions avant interruption  => 256 int=20ms
                clear_bit( intcon, T0IF );   // Effacement du drapeau interruption
                mytimer++;  // incrémente toutes les 78µs

                // positionne les servos 0=arret
                // positions entre 13(1ms) et 25(2ms)
                if (mytimer<sens_gauche) set_bit(porta,6); else clear_bit(porta,6);
                if (mytimer<sens_droit) set_bit(porta,7); else clear_bit(porta,7);

                if (mytimer==0) mytimerplus++;  //mytimerplus fait le tour en 5 secondes
                if (mytimer<flashspeed) set_bit(porta,2); else clear_bit(porta,2);
                }
}
```

```
void initialisation()
{
osccon=0x7C;  //initialisation de l'horloge interne à 8Mhz
ansel=3; // AN0 et AN1 en Convertisseurs
adcon1=0; set_bit(adcon1,6);
adcon0=0x41; // clock à 16 + mise en route ADC
set_bit(adcon0,3); //utiliser RA1

porta=0;
trisa=0x3B; //A0 et A1 en entrée, A6, A7 en sortie (servomoteurs) A2=LED
portb=0;
trisb=0xFF; // B6 et B7 en entrée au début. B0 et B2 sur capteur de contact.
set_bit(porta,2); delay_ms(250);
clear_bit(porta,2); delay_ms(250);
set_bit(porta,2); delay_ms(250);
clear_bit(porta,2); delay_ms(250);

option_reg=0x80; // timer 0 et prescaler à 0 + pull-up non

intcon=0xE0; //GIE, PEIE, TOIE
}
```

```
void tourneg(void)
{       sens_gauche=TOURNESENESTRE; sens_droit=TOURNEDEXTRE;}
void tourned(void)
{       sens_gauche=TOURNEDEXTRE; sens_droit=TOURNESENESTRE;}
void avance(void)
{       sens_gauche=TOURNESENESTRE; sens_droit=TOURNESENESTRE;}
void recule(void)
{       sens_gauche=TOURNEDEXTRE; sens_droit=TOURNEDEXTRE;}
void stop(void)
{       sens_gauche=0; sens_droit=0;}

void main()
{
char i,j;
initialisation();

while (1)
{
```

```
// gestion du CAN
if (adcon0&0x04) // CAN en cours
        {
        restart_CAN=0; //conversion en cours stopper le compteur de delay entre les conversions
        }
else {
        if (restart_CAN<10)
                {
                restart_CAN++; // relancer la conversion après un certain temps
                }
        else if (restart_CAN<200)
                {
                value_a1=adresh;
                delay_us(8);
                set_bit(adcon0,2);  // relancer la conversion
                }
        }
```

```
if (value_a1>128)  // plus lumineux a droite
        {
        if (value_a1 > 128+GRANDELUMINOSITE ) // grand écart de luminosité
                {
                flashspeed=1;
                tourned();
                }
        else {
                flashspeed=50;
                avance();
                }
        }
else { // plus lumineux a gauche
        if (value_a1< 128-GRANDELUMINOSITE ) // grand écart de luminosité
                {
                flashspeed=254;
                tourneg();
                }
        else {
                flashspeed=50;
                avance();
                }
        }
```

```
while ((portb&5) != 5)
        {
        i=portb&5;
        recule();delay_ms(125);
        if (i == 1) tourneg();
        if (i == 4) tourned();
        delay_ms(30);
        }
```

```
}
}
```