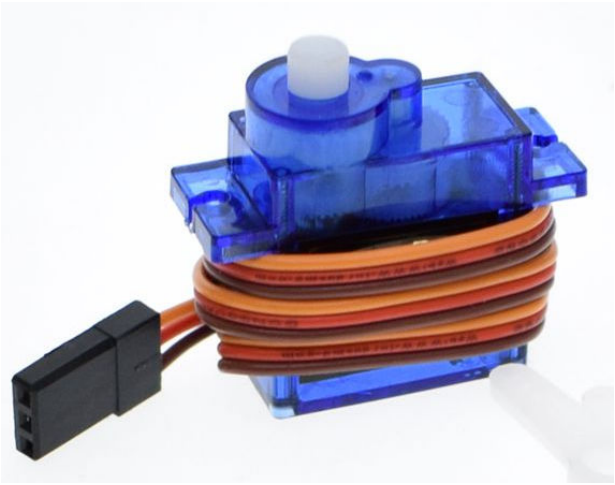
Arduino et les servomoteurs

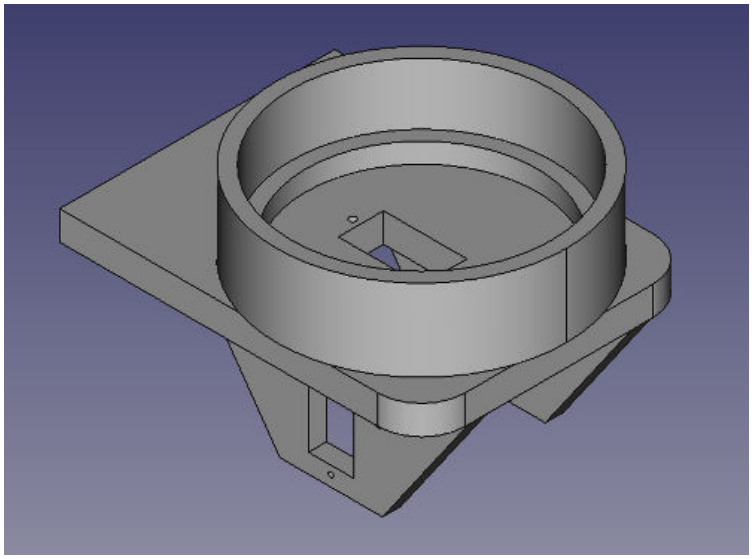
Comme j'ai acquis récemment un peu de matériel : une imprimante 3D ainsi qu'un Arduino et des servomoteurs. Bien sûr quand on a des nouveaux jouets, on joue avec ! Pour commencer j'ai fait tous de sortes de pièces plastiques pour réparer divers objets de la maison ou diverses boîtes de rangement. Pour finir j'ai donc dessiné une pièce qui pouvait recevoir ces deux servomoteurs.



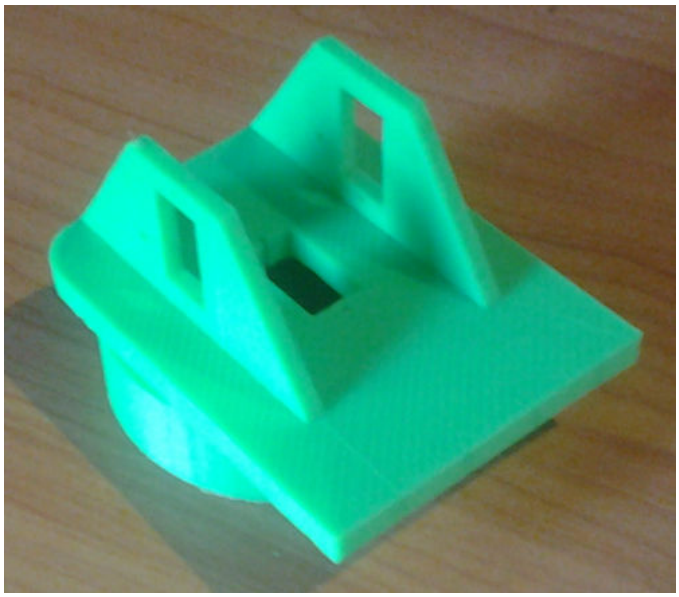
Puis je me suis demandé comment faire bouger quelque chose avec tout cela. Comme vous le comprenez, ce n'est pas un projet structuré ou l'objectif final est bien défini, c'est plutôt au jour le jour, que puis-je faire ?

Les pièces mécaniques

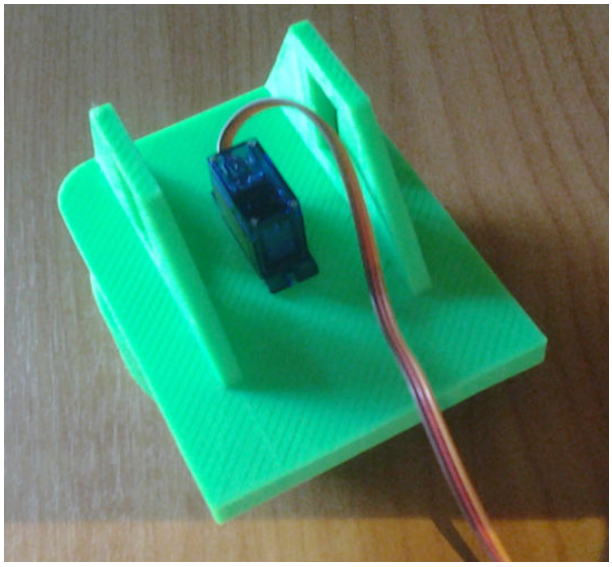
Pour fabriquer les pièces avec l'imprimante 3D, j'utilise le logiciel "Freecad v0.16" et j'imprime avec "Cura v2.5.0". La première pièce dessinée de ce "projet", une base de : "je ne savais pas encore quoi", je voulais simplement deux emplacements pour mes servomoteurs. En fait, comme vous pouvez le voir il y avait 3 emplacements !



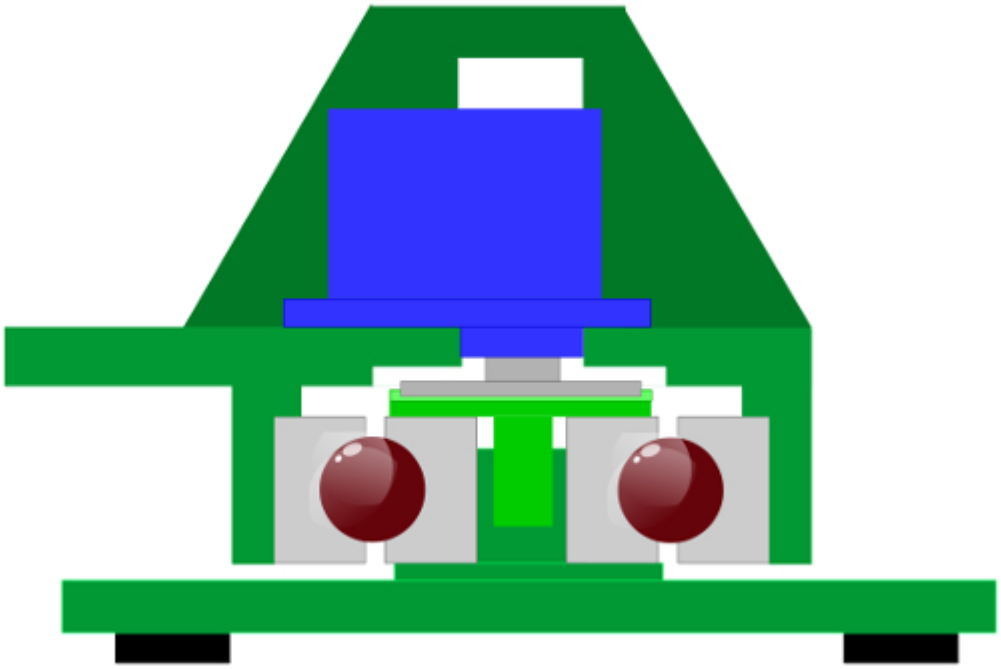
Avec la pièce résultante, pour la couleur c'est le fil plastique reçu avec l'imprimante, donc pas encore de choix. C'est du PLA, un plastique biodégradable et, ma foi, fort solide !



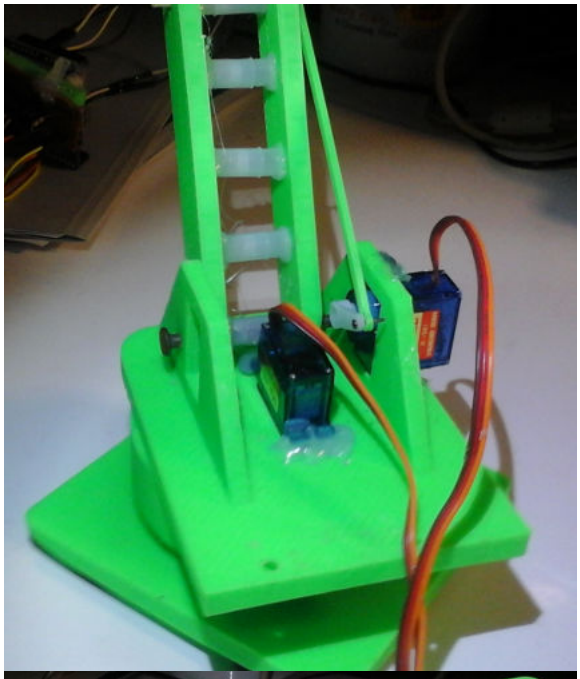
La même pièce avec un servomoteur en position :

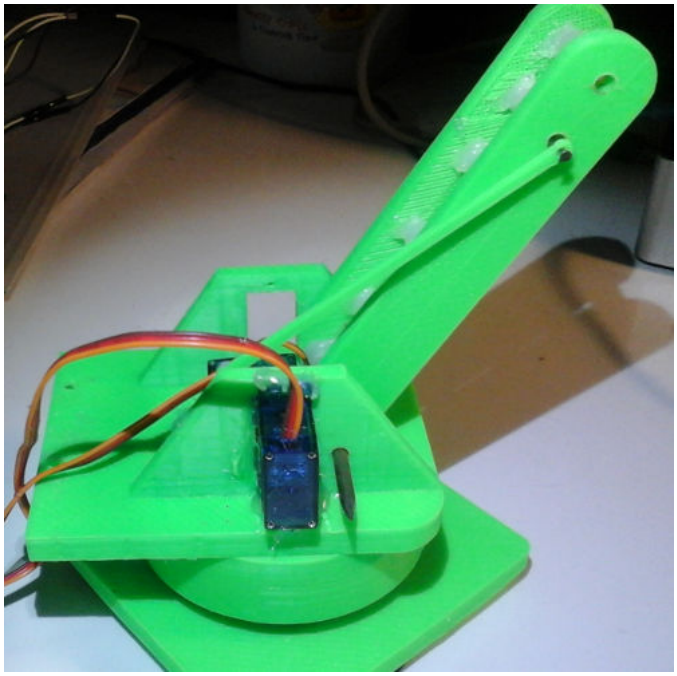


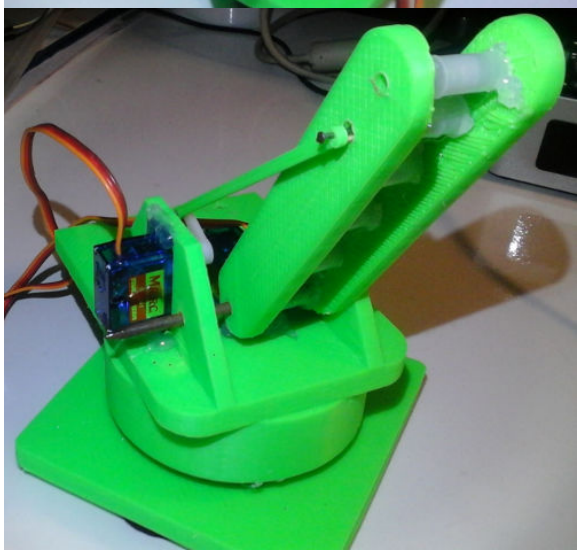
*Le schéma de l'axe de base, en fait il y a plus de pièces, car il y a du rafistolage un peu partout, les pièces n'étant pas vraiment adaptées *



Pour finir cette partie mécanique, voici à coups de colle chaude, de clous et de bout de plastique de récupération, l'allure de l'ensemble. Même le roulement est très grossier et plein de jeu, bref c'est une "honte mécanique", mais ça marche …….







Les pièces ne sont pas très réussies (c'est mes débuts avec l'imprimante) et c'est pour cela qu'en fin de compte, la bidouille à fait son apparition. J'ai remplacé les axes et la visserie par de la colle chaude, des bouts de bois et des clous !

Le programme de pilotage

Dans le monde de l'Arduino et des servomoteurs, la programmation n'est pas un problème compliqué pour peu que vous ayez quelques notions de programmation. L'interface (IDE) d'Arduino, c'est la version 18.3 que j'utilise, est rapidement prise en main et vous avez sur le Net une multitude de d'exemples de code. Il y a bien sûr la bibliothèque "servo.h", mais avec cette librairie (bibliothèque), je n'ai jamais réussi à contrôler convenablement la vitesse des mouvements, c'est pourquoi je ne l'utilise pas. De plus, un problème au moment

de l'initialisation de l'Arduino et des servomoteurs, ces derniers effectuaient un mouvement en position médiane. J'ai donc écrit ma petite routine de création de pulses, très simple 3 ou 4 lignes de programme. C'est adapté à mon Arduino et mes servomoteurs et cela fonctionne pas trop mal.

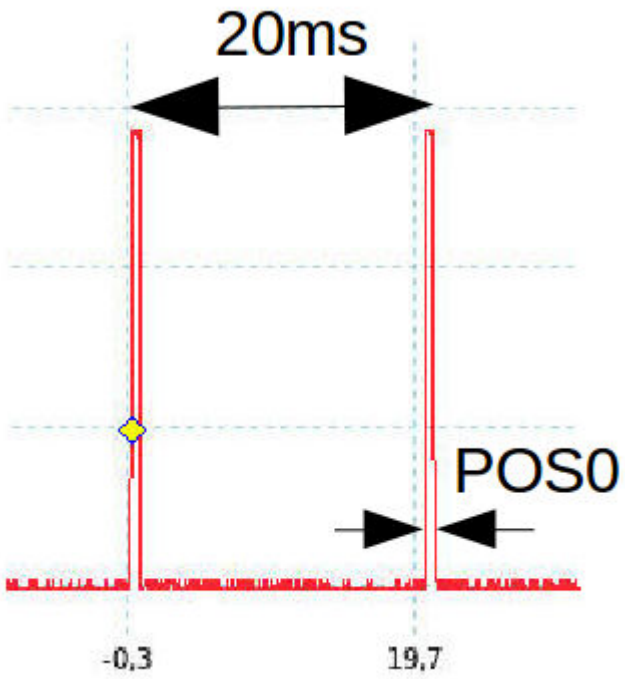
PILOTAGE DES SERVOMOTEURS

Je ne vais pas vous faire un cours sur les servomoteurs, car vous trouverez énormément de documentations et tutoriels sur Internet (site1, site2, ...).

En plus de la puissance électrique, les servomoteurs doivent recevoir un pulse d'une durée définie toutes les 20ms. La durée de ce pulse définit la position du servomoteur. Voilà c'est tout ce qui nécessaire de savoir pour piloter par un Arduino des servomoteurs (un servo en l'occurance) :

```
if (millis() - lastPulse0 >= 20) {
  digitalWrite(PinServo0, HIGH);
  delayMicroseconds(POS0);
  digitalWrite(PinServo0, LOW);
  lastPulse0 = millis();
}
```

Sur le listing ci-dessus, le test du début contrôle que l'on fabrique un pulse chaque 20ms. La construction du pulse sur la "PinServoO" commence par la mise HIGH puis attente, durée du pulse "POS0" (qui correspond à la position) puis passage à LOW donc la fin du pulse. Je pense que vous avez deviné que 0 = servo0, si vous avez d'autre servo 1,2, ... changer les repères. Voici le résultat sur la pin en question.



La définition de POS0 dépend de votre servomoteur, pour les miens, cette valeur varie de 630µs pour 0° à 2430µs pour 180°. Comment trouver ces valeurs ? Là aussi c'est Internet qui est votre ami pour les servomoteurs bon marché, pour les autres (les onéreux) vous avez les valeurs des fabricants. Attention certains servomoteurs ne vont pas au delà de 90°! Renseignez vous avant d'acheter, ne faites pas comme moi, le prix et hop on regarde après ce que l'on peut faire !

Vous pouvez aussi retrouver ces valeurs par tâtonnement avec un petit programme de test sans oublier votre oreille ! Voici le programme que j'utilise, un peu modifié à ma sauce.

```
/*servo-skatch.ino
Example sketch for connecting a hobby servo to a sparkfun redboard
(https://www.sparkfun.com/products/9065)
(https://www.sparkfun.com/products/12757)
Byron Jacquot@ SparkFun Electronics
May 17, 2016
**SparkFun code, firmware, and software is released under the MIT
License(http://opensource.org/licenses/MIT).**
Development environment specifics:
Arduino 1.6.5
*****************************************************************************/
#include <Servo.h>
int temps = 1800;//censé être à mi-chemin entre 1000 et 2000, un bon point de départ
Servo monServo;
boolean ret=false;
int incr=100;
int plus=0;
int inf=630;
void setup()
{
Serial.begin(115200);
Serial.println("Hello World");
monServo.attach(7);
//on démarre à une valeur censé être la moitié de
//l'excursion totale de l'angle réalisé par le servomoteur
monServo.writeMicroseconds(temps);
}
void loop()
{
//des données sur la liaison série ? (lorsque l'on appuie sur 'a' ou 'd')
if(Serial.available())
{
char commande = Serial.read(); //on lit
//on modifie la consigne si c'est un caractère qui nous intéresse
if(commande == 'a')
temps += 50; //ajout de 10µs au temps HAUT
else if(commande == 'd')
temps -= 50; //retrait de 10µs au temps HAUT
//on modifie la consigne du servo
monServo.writeMicroseconds(temps);
//et on fait un retour sur la console pour savoir où on est rendu
Serial.println(temps, DEC);
}
if (ret==true){
monServo.writeMicroseconds(inf);
ret=false;
}else{
if(plus>1800){
plus=0;
}
monServo.writeMicroseconds(inf+plus);
Serial.println(plus, DEC);
plus=plus+incr;
ret=true;
}
delay(2000);
}
```

Le réglage de la position est assez simple il suffit donc de fixer POS0 pour le servo0. Voici le bout de programme qui effectue ce travail.

```
POS0 = map(pos0,0,180,minRot,maxRot);
```

Comme vous le voyez c'est vraiment très simple. Si vous ne connaissez pas l'instruction "map", voici une page d'explications des fonctions Arduino. Je fixe les mouvements en degré, directement transformés en µs pour la largeur du pulse. Reste maintenant la vitesse de déplacement des axes. Là aussi c'est extrêmement simple et j'applique la même recette que pour les pulses. Pour expliquer, rien ne vaut mieux qu'un exemple concret : pour déplacer un axe de la position 45° à la position 90°. Je demande un déplacement degré par degré et je contrôle le temps entre chaque demande.

- *45° position de départ*
- *fait une pause*
- *va à la position 46°*
- *fait une pause*
- *va à la position 47°*
- *fait une pause*
- *.....*
- *fait une pause*
- *va à la position 90°*
- *attends la suite*

Pour le servomoteur il n'y pas de différence entre "fait une pause" et "attends la suite", il reçoit toujours, toutes les 20ms, les pulses de la largeur correspondant à sa position. Par contre, le "fait une pause" permet de ralentir l'enchaînement de l'augmentation (ou la diminution) des degrés. Voici le listing Arduino pour les servomoteurs, j'ai mis uniquement le servo1 (servo0 est similaire)

```
void mvtAxe1(){
  if (mvt1!=pos1)
  {//mouvement demandé
    if (millis() - lastPulVit1 >= VSB1)
    { //speed control
      att1=false;
      if(mvt1>pos1){
        pos1=pos1+1;
      }else{
        pos1=pos1-1;
      }
      VitServo1();
      POS1 = map(pos1,0,180,minRot,maxRot);
      lastPulVit1=millis();
    }
  }else{
    if(att1!=true){
     Serial.print("AXE BRAS position atteinte ");
     Serial.println(mvt1);
     att1=true;
    }
  }
  if (millis() - lastPulse1 >= 20) {
    digitalWrite(PinServo1, HIGH);
    delayMicroseconds(POS1);
    digitalWrite(PinServo1, LOW);
    lastPulse1 = millis();
  }
}
```

Quelques mots sur les variables :

- *mvt1 : position demandée pour aller de 45° à 90°, cela signifie que mvt1 = 90*
- *VSB1 : c'est la vitesse demandée, enfin plutôt le délai entre chaque incrément de degré spécifié par l'appel à la fonction : VitServo1()*

```
void VitServo1()
{
  readVal();
  VSB1 = map(Vit1,0, 225,maxVit,minVit);   //transfert de table
}
```

- *maxVit et minVit : sont respectivement 10ms et 140ms pour mes servos sans qu'ils se perdent ou qu'ils "zoonent" trop. J'ai fixé arbitrairement une échelle de 0 à 225 pour la vitesse. Donc pour fixer un déplacement je donne le lieu d'arrivée(mvt1) et la vitesse (Vit1). C'est le rôle de la fonction readVal(), j'y reviendrais plus tard.*
- *att1 : signal que le but du déplacement est atteint, on se trouve maintenant à 90° dans notre exemple*
- *Le dernier bout de code vous le connaissez c'est la "fabrication par l'Arduino" du pulse*

pour le servomoteur.

Comme vous l'avez deviné, vous avez autant de routine mvtAxe() que de servos …

Voici une petite vidéo de l'écran de l'oscilloscope, regardez le signal rouge, c'est le signal sur l'Arduino du servomoteur, sortie (PinServo1). Vous voyez le pulse s'agrandir puis se rétrécir, c'est normal car le servomoteur fait un mouvement de va-et-vient (0°-180°-0°-180°-…) avec une pause en fin de course. J'espère que comme cela c'est clair cette histoire de pulse. Pour le signal bleu j'y reviens dans la partie électronique.

L'électronique

On a la mécanique, l'Arduino et les servomoteurs avec leur pilotage, il faut maintenant simplement un peu de courant électrique et …… RIEN NE FONCTIONNE !!!!

C'est normal ! Cela vérifie encore, si besoin était la fameuse citation d'Einstein sur la théorie et la pratique.

D'abord, j'avais acheté mes servomoteurs sans trop regarder, à part le prix, et donc je pensais naïvement qu'ils fonctionnaient sous 5V. Mais non c'était sous 3.5V !! Bon pas de panique, j'ai dans ma panoplie d'Arduino, un Olimexino-328 (compatible Arduino) avec une sortie 3.3V cela ira bien. J'ai mis au point mon programme et testé avec uniquement le servomoteur libre sans effort mécanique. Cela fonctionnait malgré quelques fois des comportements du servomoteur pas très clair, mais bon j'ai incriminé d'abord le software puis mes branchements car ils étaient un peu "olé olé", bref, rien ne m'a vraiment inquiété.

J'ai monté le servomoteur avec la mécanique et là, bien sûr, plus rien de correct ; le servomoteur fait vraiment n'importe quoi et je ne pilote plus rien du tout. N'y comprenant plus rien, je reviens à la situation antérieure (sans mécanique), ça marche pas trop mal. Donc le problème c'est la mécanique ! CQFD !

J'ai remarqué que cela marchait mieux (c'est-à-dire que le mouvement était mieux contrôlé) dans un sens que dans l'autre. J'ai recherché une raison mécanique comme source du problème. Rien, je patauge !! Comme presque toujours, une recherche sur le Net montre que d'autres bricoleurs ont les mêmes problèmes et un internaute suggère (je ne me souviens plus du nom de ce "sauveur de l'humanité") que peut-être il s'agit d'un manque puissance électrique. Certains servomoteurs lorsqu'ils n'ont pas assez de puissance à disposition, ont un comportement erratique, ils font n'importe quoi : voir la vidéo ci-dessous, aucune demande de mouvement est faite et ….

En contrôlant la tension sur l'alimentation du servomoteur on voit nettement le problème. C'est le signal bleu sur la vidéo précédente. Bon maintenant, c'est clair il me suffit d'alimenter suffisamment mes servomoteurs et tout rentrera dans l'ordre. J'ai donc installé un petit circuit sur base LM2596 qui permet de délivrer assez de courant (3A max) mais avec mon alimentation générale de 1.5A en 5V (7,5W tout de même), eeehhh ben non ! C'est pas suffisant pour un Arduino et les servomoteurs, deux maintenant, après probablement trois ou quatre !

Voici une vidéo du mouvement et des signaux (pulse et tension d'alimentation). Un pulse juste au-dessous de 3V ne semble pas poser de problème, mais dès que la tension d'alimentation tombe au-dessous de 3.1V c'est l'anarchie. En survoltant (4.2V) on arrive à maintenir la tension d'alimentation au-dessus de 3.2V et donc on contrôle le mouvement.

Donc il me faut une alimentation plus solide, ça c'est pour le prochain épisode ……..

Quelques documents

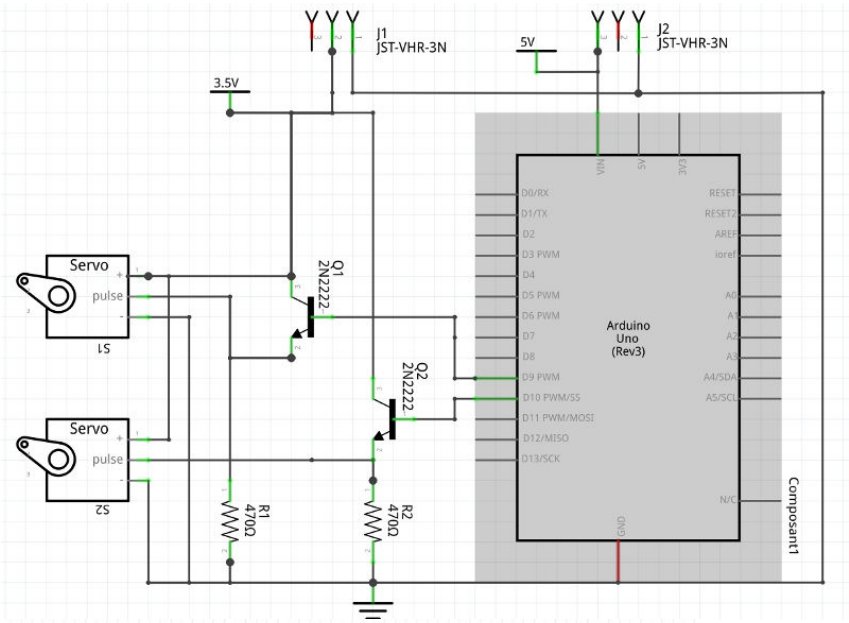
PROGRAMME ARDUINO

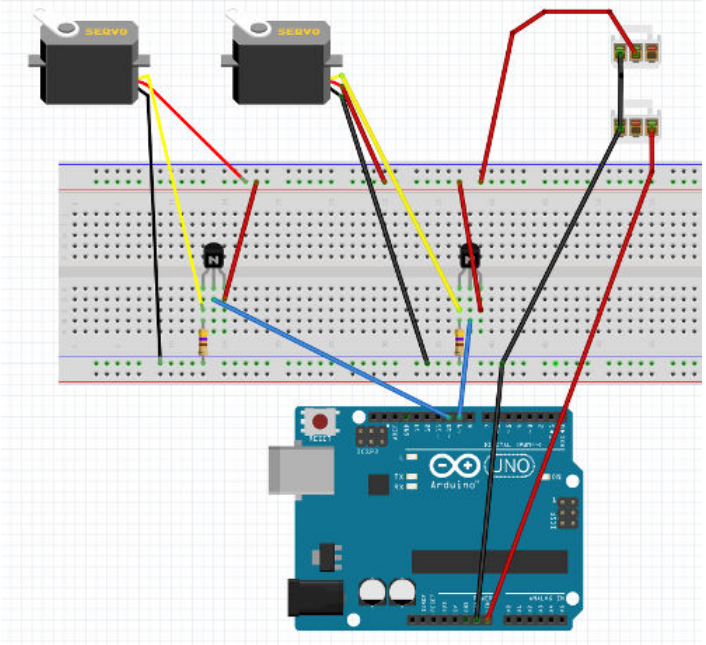
Pour ceux qui veulent : mon programme Arduino (en l'état). Actuellement les commandes proviennent du moniteur en ligne de l'IDE d'Arduino, mais en principe un module Bluetooth (ça fait trois jours que je n'arrive pas à communiquer depuis le smartphone!!) devrait permettre un pilotage depuis un mobile enfin j'espère ….

Il y a aussi un paramètre à ajouter c'est un facteur qui permet de moduler l'accélération (et la décélération) des mouvements. C'est le rôle de la variable "acc0" dans le listing, non utilisé actuellement.

SCHÉMA ÉLECTRONIQUE

Très simple mais fonctionnel. (fichier Fritzing, changez l'extension zip en fzz)

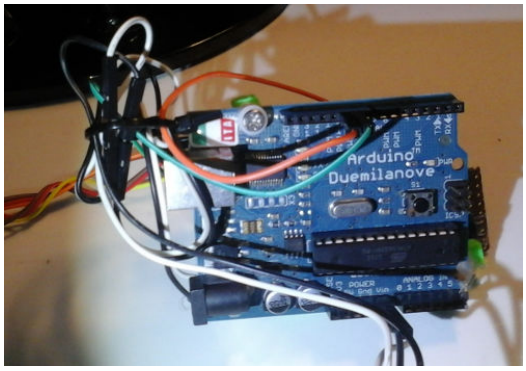


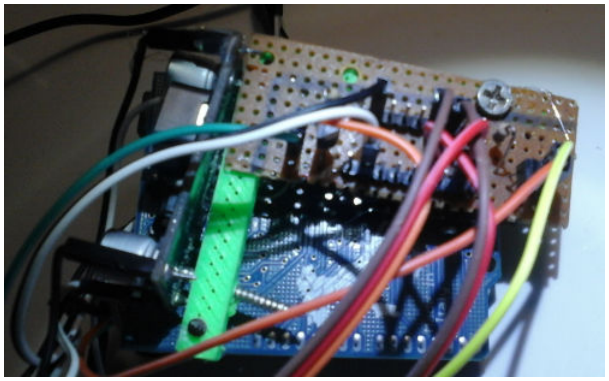


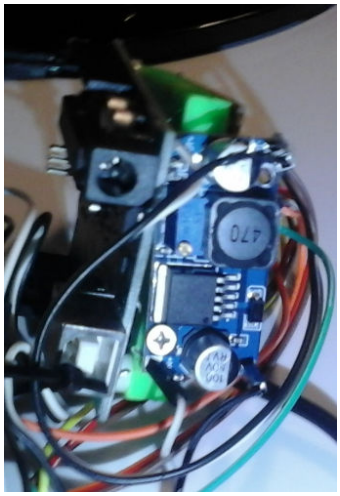
Cette partie est à revoir car il faut ajouter le module Bluetooth, prévoir deux servomoteurs supplémentaires et certainement un ou deux capteurs pour ne pas être "aveugle". Il faut surtout que je regarde comment réduire le poids (l'encombrement) de l'ensemble,

probablement passer à un Nano, Mini ou Micro, à voir .....

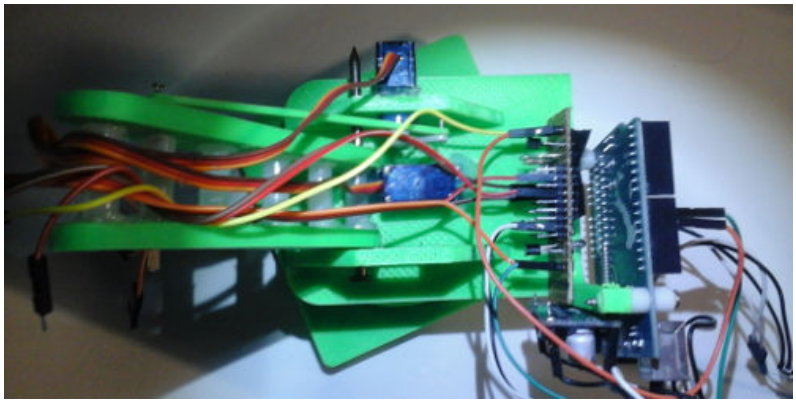
L'électronique d'essai : un Arduino (Duemilanove), un circuit DC-DC (M2596) et deux transistors(2N2222).

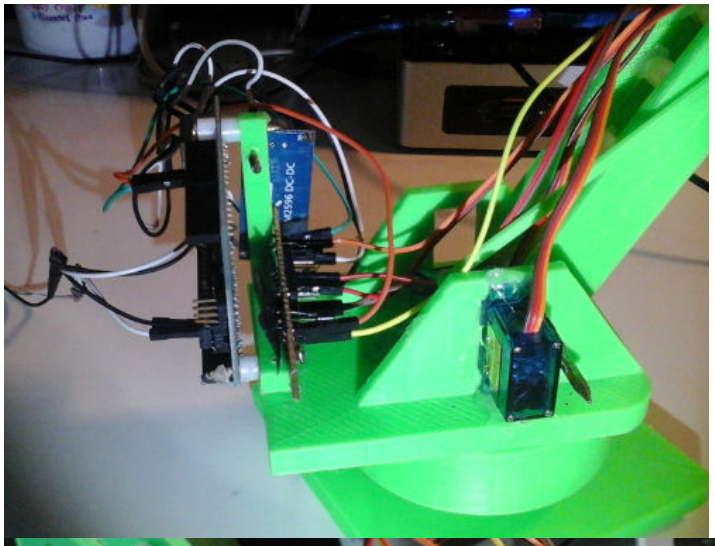


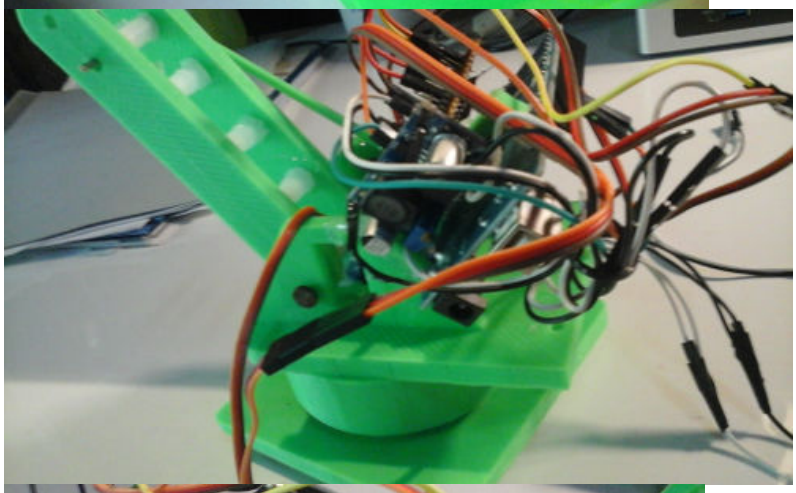


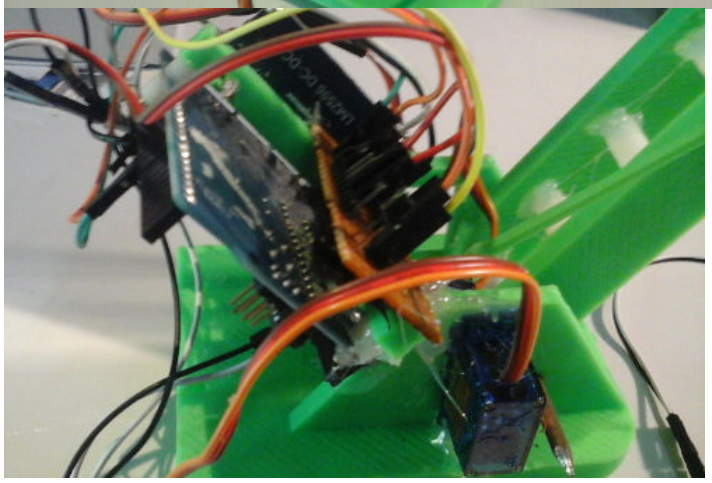


J'ai voulu dans un premier temps fixer cet ensemble (difficilement définissable) sur une équerre sur l'arrière de l'ébauche de la pelle mécanique. Trop lourd et trop de porte à faux. Le servomoteur peinait, j'ai tenté un aguillage à coup de colle chaude pour que l'ensemble soit plus au centre de rotation mécanique, ça donne un vrai embrouillamini !! Jugez vous même :









C’était meilleur, le servomoteur fonctionnait normalement, donc du travail aussi dans cette direction.

Conclusion

Faites des recherches avant de faire, oui : étudiez avant de faire c’est évident !!!