La partie électronique et informatique

1)Les moteurs pas à pas

Les moteurs pas à pas sont des moteurs qui font tourner un axe grâce à un aimant et des bobines qui créent un champ électromagnétique.

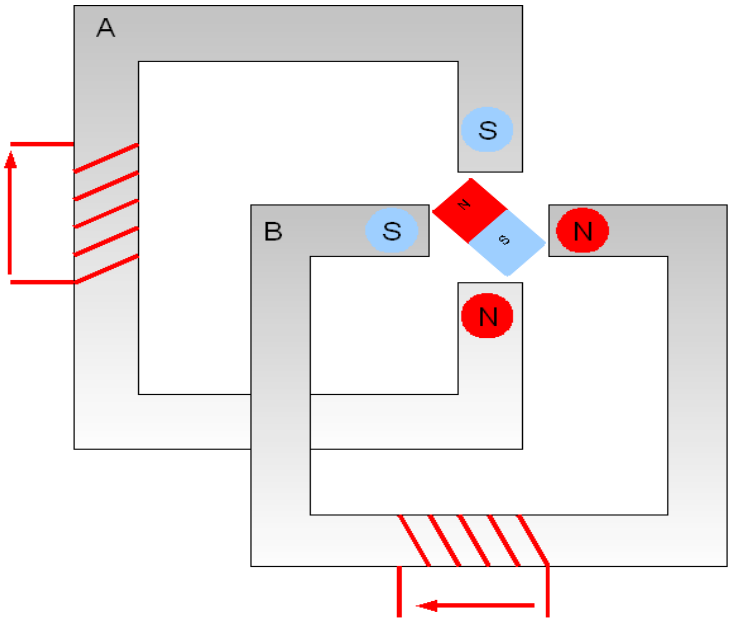


Illustration 11 Schéma du fonctionnement d'un moteur bipolaire demi-pas

L'aimant tourne grâce à la répulsion électromagnétique. Pour que l'aimant tourne, les pôles de l'aimant et de la branche du moteur vers laquelle il est tourné, doivent être inverse pour qu'il y ait répulsion. Quand il ne tourne pas, il faut que l'axe soit bloqué pour ne pas qu'il tourne sans le consentement de l'utilisateur. Le moteur peut tourner de deux façons différentes, par demi-pas comme vu sur le schéma ou par pas entier. Pour les demi-pas, les quatre branches du moteurs sont alimentées de sorte que l'aimant soit bloqué à mi chemin entre deux branches. Il y a donc 8 positions possibles (4 demi-pas et 4 pas entiers) car l'aimant peut être bloqué avec seulement deux branches (par exemple la branche A en haut en sud et la A en bas en nord).

Le choix des moteurs pas à pas c'est fait naturellement car leur nature permet d'avoir une très grande précision sur la rotation de l'axe et un contrôle facile grâce à un microcontrôleur.

Les moteurs que j'ai choisi (fiche technique dans l'annexe) ont 200 pas par révolution, c'est à dire que chaque pas fait 1.8°, étant donné que ma poulie a un diamètre de 30mm chaque pas fait (rayon*angle en radian) $15*1,8*2\pi/360=0,4712$ mm. Si je fais fonctionner mon moteur en demi pas j'ai donc une précision à ±0,2356mm, mais il faut aussi voir la puissance de maintien (holding torque) et la puissance rotative (rotor torque) du moteur.

2) Les drivers et arduino

Les drivers sont des microcontrôleur qui, en interprétant des signaux créneaux (port

3), contrôlent les moteurs en envoyant du courant puiser dans une alimentation (port2) dans les branches du moteur A,A',B,B' (port 1).

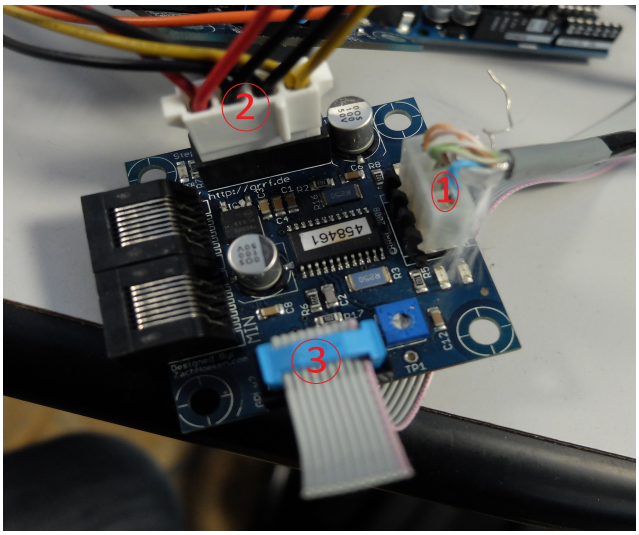


Illustration 12: Driver moteur (ports numérotés)

Cette puce est reliée à Arduino par quatre branches, une branche terre GND, une branche STEP qui à chaque créneau donne l'ordre de faire un pas si la branche EN n'est pas alimentée et une branche DIR qui indique si le moteur va tourner dans le sens des aiguilles d'une montre ou l'inverse.

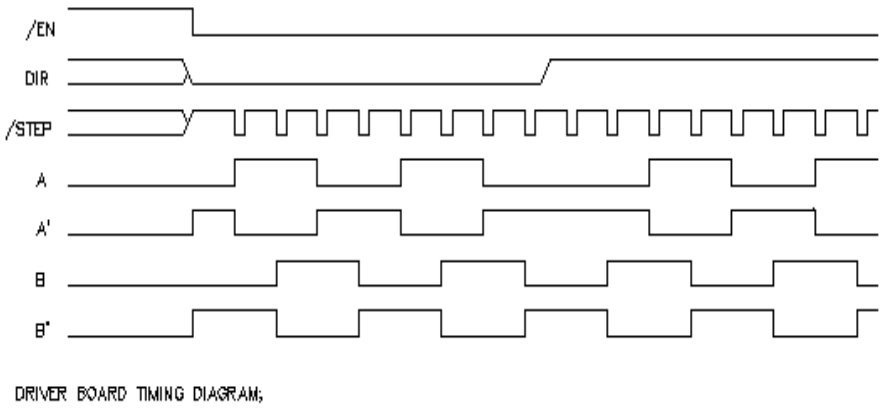


Illustration 13: Transformations des signaux effectuées par le driver

Les drivers sont indispensables pour calculer les envois d'énergie dans chaque branche.

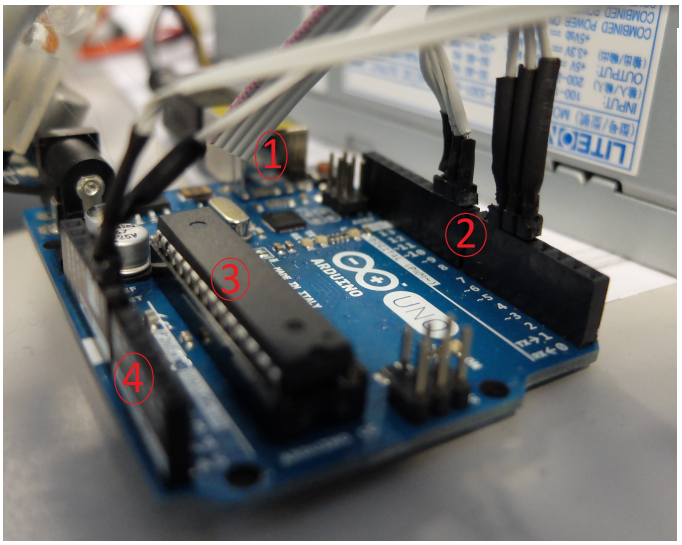


Illustration 14: Arduino

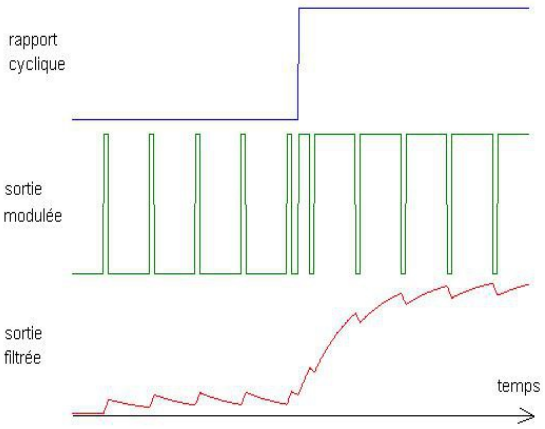


Illustration 15: Signaux PWM

Arduino est un microcontrôleur (3) qui est fixé sur une carte qui est agencée de manière à ce que chaque pin analogique (4) et numérique (2) soit accessible et que le code informatique implanté dans le microcontrôleur soit modifiable par USB (1).

- Les ports numériques peuvent libérer sous forme de signal créneaux des informations binaires par le biais de signaux PWM(Pulse With Modulation qui en fonction de la pulsation des créneaux font un 1 avec 5V ou un 0 avec 0V). On peut voir sur l'illustration 14 que les deux drivers moteurs (3 pins pour EN, DIR et STEP) sont reliés à des pins numériques.
- Les ports analogiques sont des entrées qui traitent des tensions allant de -5V à 5V. Pour pouvoir donner des ordres aux sorties numériques il faut écrire un code dans un langage Arduino développé autour d'une bibliothèque interne du microcontrôleur.

Il existe de nombreuses bibliothèques qui correspondent à tout type de matériel contrôlable de cette manière par exemple je m'en suis servi d'une qui me permettait de contrôler le driver moteur avec des fonctions simples.

3)La partie logiciel

Une question se pose quand il s'agit de retranscrire une image en commande pour une machine est, dans quel ordre et par quelles trajectoires le traceur va-t-il dessiner. Il y a aussi le problème de comment « retranscrire » les niveaux de gris Les solutions existantes proposent trois manières :

- Le traçage linéaire (voir illustration 4) est une méthode qui, pour balayer toute la zone de dessin, trace des lignes horizontales les une en dessous des autres. Pour tracer une ligne droite, il faut faire tourner les deux moteurs à la même vitesse dans le même sens. Étant donné qu'il traverse l'image horizontalement, il fonctionne comme une imprimante. Il utilise donc un système de pixel carré basique. Pour foncer plus ou moins chaque pixel, il faut faire vibrer avec une amplitude proportionnelle soit

aléatoirement soit selon un signal (créneau, sinusoïdale).

- Le traçage radial (image) prend une des poulies pour en faire l'origine du cercle dont le rayon correspond à la rangée de pixel traité. Pour chaque cercle un moteur bloque avec une longueur de fil qui va rester constante pendant que l'autre varie en dessinant

Illustration 16: Traçage radial

un arc de cercle sur la surface voulue. Grâce à cette méthode, on peut atteindre chaque point de l'image en faisant varier la longueur bloquée. L'image utilisée sera découpée en pixel en forme de losange pour que chaque rangée de pixel corresponde à un rayon bloqué. En ce qui concerne les niveaux de gris, la même technique que le traçage linéaire est utilisé.

- Le traçage par chemin lui n'aura pas un balayage de toute la surface de l'image comme les autres mais ne passera que sur des chemins, comme des lettres ou les contours d'une image. Programmer les mouvements est beaucoup plus complexe car il faut travailler avec des images vectorielles et donc transformer les vecteurs en chemin. Il est possible d'utiliser des programmes déjà existant qui transforme les chemins en gcode(langage interprété par les machines qui est basé sur des mouvements cartésiens) mais il faut changer ce système cartésien en un système qui prend en compte les mouvements des poulies et que ce soit interprétable par Arduino.

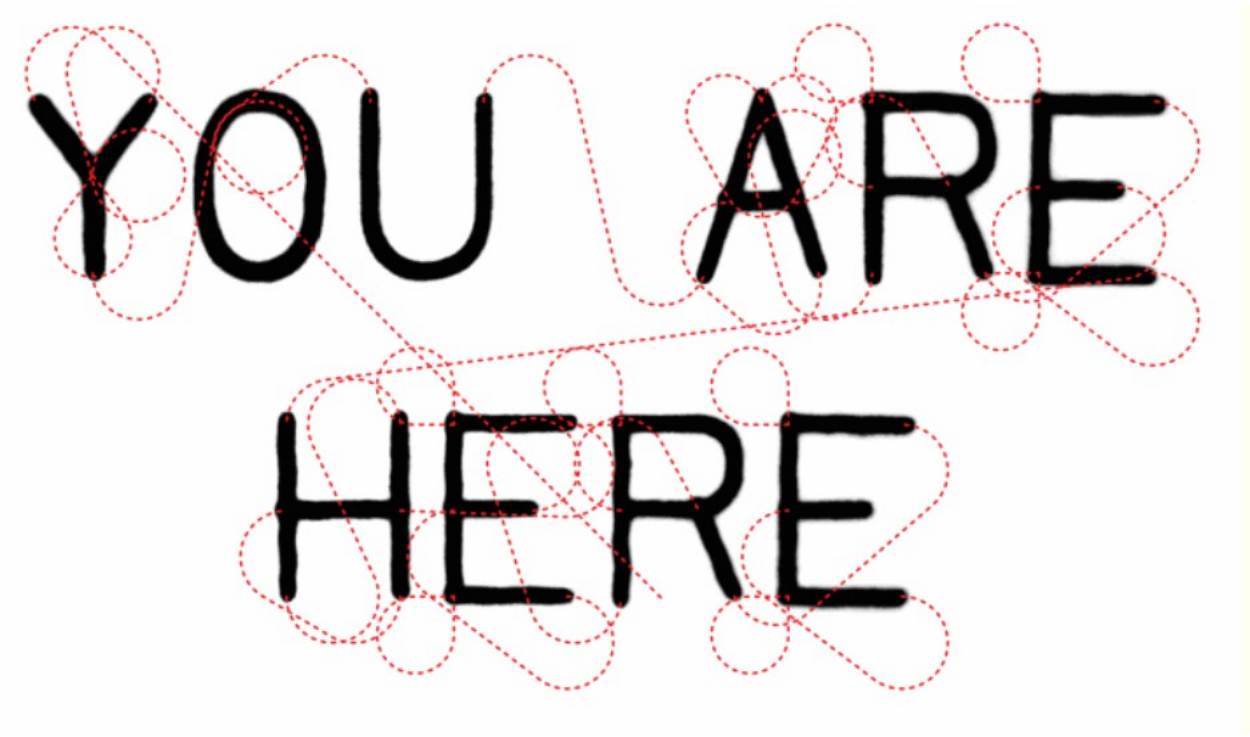


Illustration 17: Traçage par chemin

Par manque de temps je n'ai pas eu l'occasion de développer un programme capable de traiter une image en commande machine. Cependant j'ai fait un programme(annexe) qui dessine un carré pour pouvoir commencer à calibrer la machine.

Conclusion

La conception d'un objet quel qu'il soit amène de nombreux problèmes plus ou moins importants. Le travail d'ingénieur consiste à identifier tout ces problèmes puis à vérifier s'il est indispensable de les résoudre. Il faut ensuite trouver un moyen de les résoudre tout en restant dans le budget alloué au projet et réaliser les solutions avec les moyens auxquels on a accès. Enfin il faut réaliser et tester les prototypes jusqu'à ce que toutes les problématiques soient résolues. J'ai effectué toutes ces étapes en m'aidant de ressources libres trouvées au sein de l'association et sur internet afin de créer mon propre modèle de traceur vertical. A la fin du stage, j'ai fabriqué un système dont la partie mécanique (encore à perfectionner) et électronique fonctionne mais dont la partie informatique n'est pas encore achevée.

Je remercie encore Laurent Berthelot qui m'a toujours apporté une aide pertinente dans mes travaux et qui m'a appris le raisonnement de l'ingénieur.

Bibliographie

Prototypes existants :

http://www.polargraph.co.uk/

http://www.instructables.com/id/Polargraph-Drawing-Machine/

http://makerblock.com/2013/03/a-study-of-drawing-robot-pen-holders-and-design-considerations/

https://www.marginallyclever.com/blog/2012/02/drawing-lines-arc-and-halftones-with-a-wall-hanging-arduino-drawbot/

https://www.adafruit.com/blog/2014/07/04/gocupi-go-raspberry-pi-polargraphdrawbot-piday-raspberrypi-raspberry_pi/

http://www.thingiverse.com/thing:343322

http://stuartchilds.com/drbo/

http://academy.cba.mit.edu/2013/labs/providence/drawbot.html

http://juerglehni.com/works/hektor

http://www.instructables.com/id/SADbot-the-Seasonally-Affected-Drawing-robot/?ALLSTEPS

http://www.as220.org/labs/drawbot/instructions.html

http://tinkerlog.com/2011/09/02/der-kritzler/

http://dealywhopper.com/?p=105

Site de l'association :

http://www.pingbase.net/

http://www.plateforme-c.org/

Wiki de l'association :

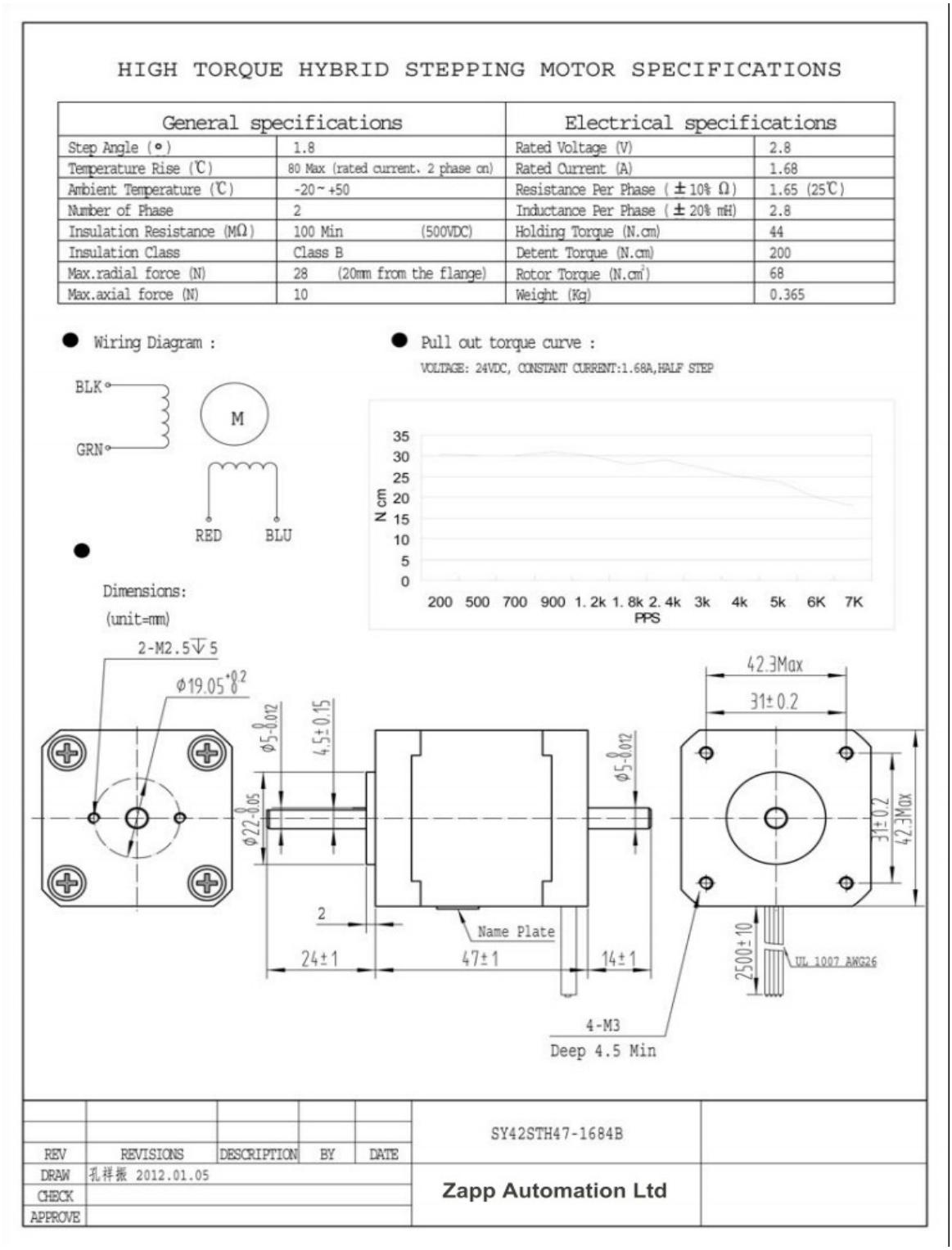
http://fablabo.net/wiki/Accueil

Documentation sur le traceur vertical :
http://plotterbot.com/plotterbot-documentation/plotterbot-prior-art/

Image signal moteur pas à pas :
http://www.hteck.ca/electronics/stepper-motor-drv/sm-driver.html

Image signaux PWM :
http://fr.wikipedia.org/wiki/Modulation_de_largeur_d%27impulsion

Annexe



HIGH TORQUE HYBRID STEPPING MOTOR SPECIFICATIONS

General specifications		Electrical specifications	
Step Angle (°)	1.8	Rated Voltage (V)	2.8
Temperature Rise (℃)	80 Max (rated current, 2 phase on)	Rated Current (A)	1.68
Ambient Temperature (℃)	-20~+50	Resistance Per Phase (±10% Ω)	1.65 (25℃)
Number of Phase	2	Inductance Per Phase (±20% mH)	2.8
Insulation Resistance (MΩ)	100 Min (500VDC)	Holding Torque (N.cm)	44
Insulation Class	Class B	Detent Torque (N.cm)	200
Max.radial force (N)	28 (20mm from the flange)	Rotor Torque (N.cm²)	68
Max.axial force (N)	10	Weight (Kg)	0.365

Illustration 18: Fiche technique des moteurs pas à pas

Code en langage Arduino pour créer un carré avec un traceur vertical :

```
#include <Stepper.h>  //bibliothèque pour moteur pas à pas
#define STEPS 200    //nombre de pas par tour du moteur
const float pi =3.1415;

float r_poulie=15;
float angle_p=1.8;  //angle effectué par pas
//distance d'un pas en mm (angle_p à convertir en radian)
const float d_pas=r_poulie*angle_p*2*pi/360;

Stepper stepper1(STEPS,5,5,6,4);      //Declaration moteur 1
Stepper stepper2(STEPS, 9,9, 10,8);   //Declaration moteur 2

void setup()
{
  stepper1.setSpeed(30);   //Définition de la vitesse des moteurs
  stepper2.setSpeed(30);

}

//dir en h (haut) b (bas) g (gauche) d (droite), l en mm
void ligne(char dir,float l){
   int n;
   n=int(l/d_pas);           //nombre de pas que constitue la distance
   switch(dir){
     case'h':
       for(int i=0;i<n;i++){
         stepper1.step(-1);
         stepper2.step(1);
       }break;
     case 'b':
       for(int i=0;i<n;i++){
```

```
        stepper1.step(1);
        stepper2.step(-1);
      }break;
    case 'g':
      for(int i=0;i<n;i++){
        stepper1.step(-1);
        stepper2.step(-1);
      }break;
    case 'd':
      for(int i=0;i<n;i++){
        stepper1.step(1);
        stepper2.step(1);
      }break;
}}
//l longueur des côtés du carré en mm
void carre(float l){
  ligne('d',l);
  ligne('b',l);
  ligne('g',l);
  ligne('h',l);
}
void loop()
{
 carre(200);
 delay(3000);
}
```

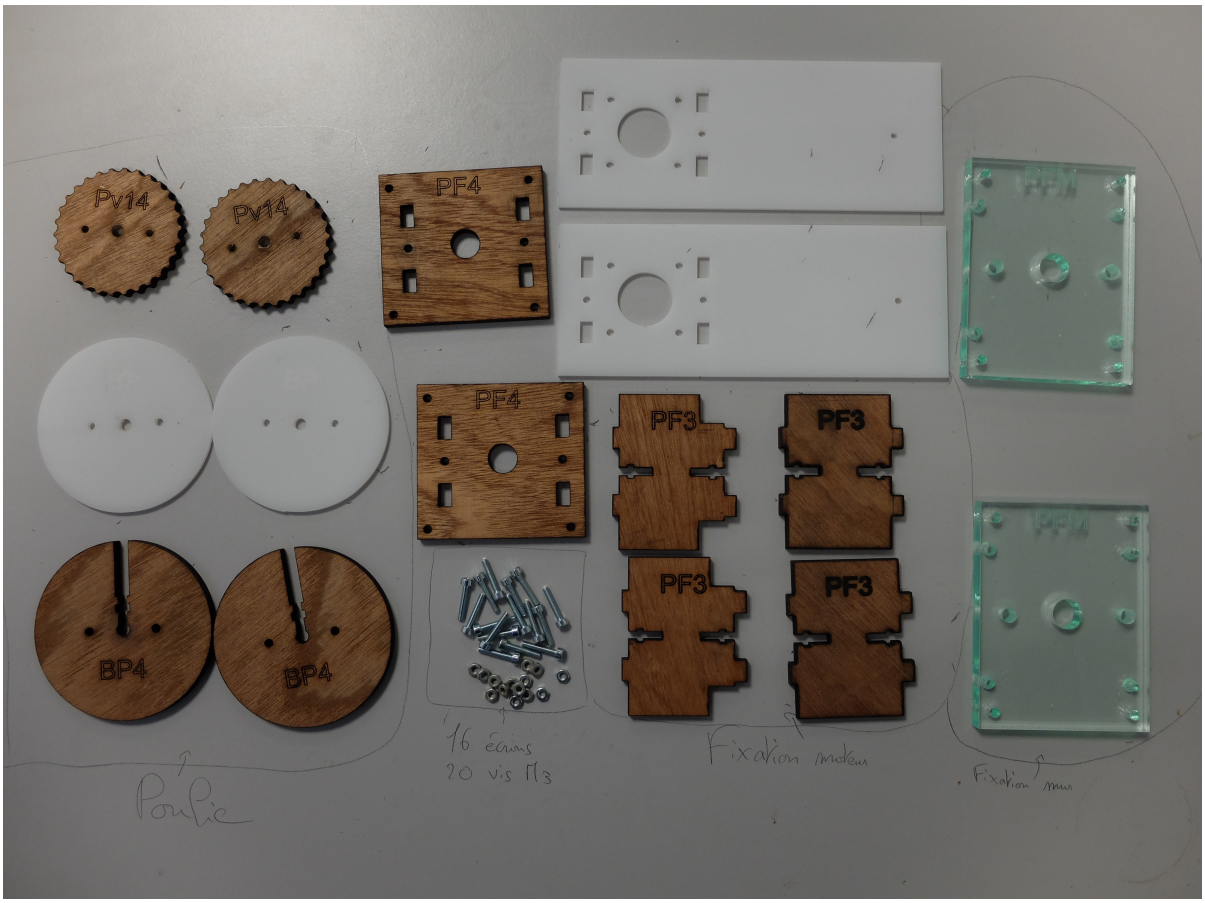


Illustration 19: Pièces poulies et fixation

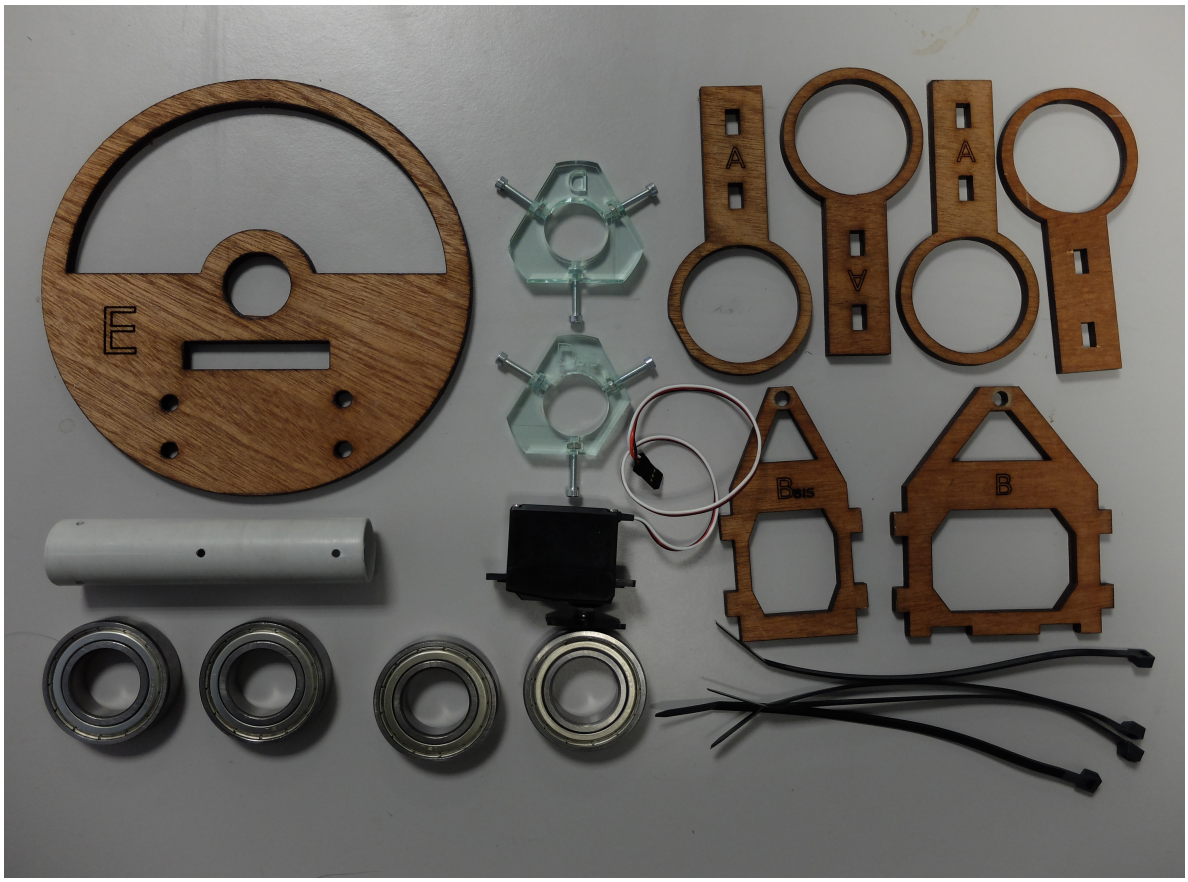


Illustration 20: Pièces porteur de stylo