ArtBot

Rapport de Projet

Jaime Alba, Adrien Boichis

15.03.2022

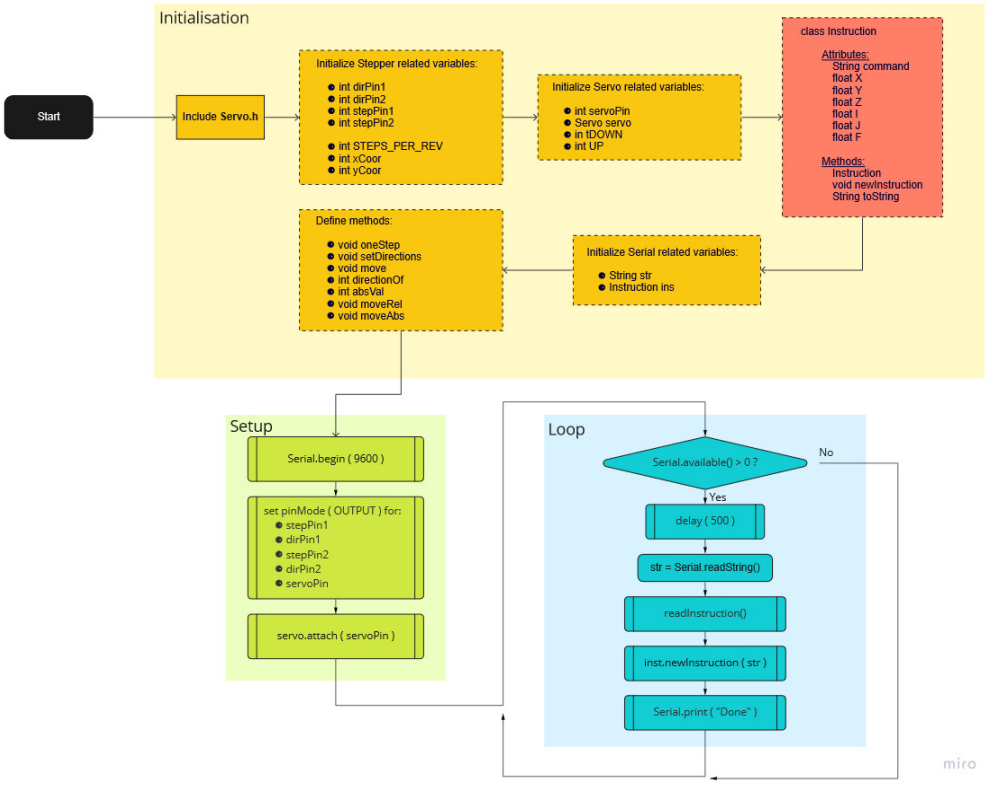
Électronique avec Arduino

1. Présentation du projet et des problématiques

L'objectif du projet ArtBot était de construire un robot avec la capacité de dessiner n'importe quelle image.

L'utilisation de celui-ci est très simple mais nécessite le téléchargement de quelques applications. En effet, pour dessiner une image, celle-ci doit d'abord être transformée en format G-Code puis envoyée vers l'arduino grâce à l'application ***Serial Processor*** que nous avons codée.

Tout d'abord, on utilise l'application ***Inkscape*** pour exporter l'image en fichier G-Code (d'extension .txt). Puis on ouvre ***Serial Processor***, on sélectionne le fichier et on l'envoie via le port USB à l'arduino. Ce dernier va finalement stocker les informations pour les lire puis les passer comme paramètres dans les fonctions de mouvement.



Nous distinguerons par la suite les x grandes parties du projet auxquelles nous avons consacré le plus de temps et que nous considérons les plus importantes:

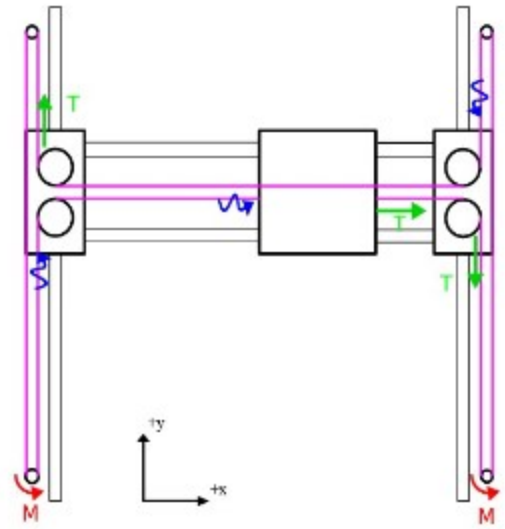
- Le Robot (partie mécanique et composants électroniques)
- Les moteurs Pas à Pas

- Transformation d'images
- Communication Serial via port USB

Le problème le plus important face à ce projet a été la précision du mouvement du stylo qui dépend fortement de la quantité de frottement et structure du robot. Nous y reviendrons plus en détail dans le prochain aparté.

2. Robot (partie mécanique)

2.1. Structure générale



Le robot ArtBot met en relation trois domaines: la mécanique, l'électronique et l'informatique.

L'idéal serait que ces trois soient le plus optimisés possible pour gagner en vitesse d'exécution et en précision et réduire la consommation et l'espace occupé

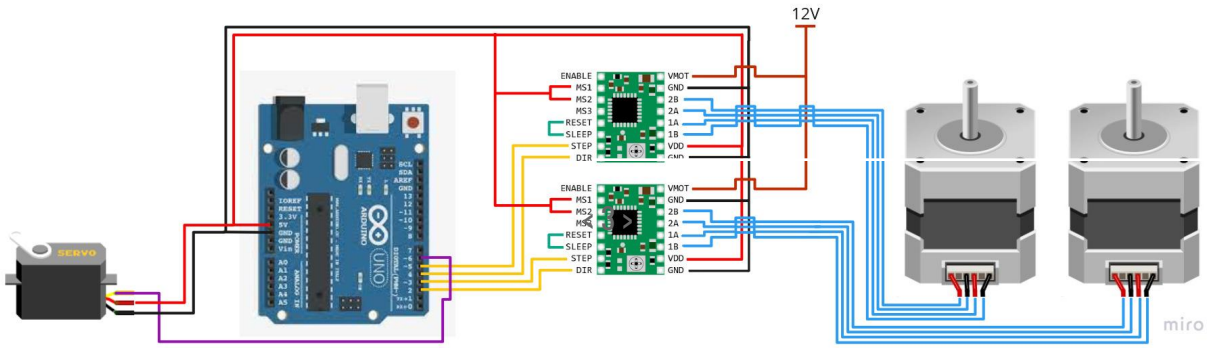
2.2. Matériel

Composants électroniques:

- Une carte Arduino Uno
- Deux moteurs pas à pas **NEMA17** (Stepper Motor)
- Deux drivers **A4988**
- Un Micro Servo moteur
- Prise 12V

Pièces:

- Trois boites découpées au laser
- 4 tubes de fer
- Pièces de bois
- Roulements
- Roulements à mouvement linéaire



2.3 Travail effectué

Nous avons commencé par établir l'idée de fonctionnement de notre projet. Puis nous avons commencé la modélisation des pièces grâce au logiciel onshape.

2.2.1. Création des pièces

Au début nous avions prévu de créer les pièces avec l'imprimante 3D. Toutefois, cette solution ne s'est pas avérée utile et nous a pris trop de temps. Nous avons donc créé les pièces à l'aide de la découpeuse laser et avec des bouts de bois. Nous avons créé deux boîtes qui tiennent les deux moteurs pas à pas. Puis pour les pièces qui bougent sur l'axe Y, on a tout simplement utilisé deux bouts de bois auxquels nous avons fixé deux roulements à mouvement linéaire pour qu'ils puissent glisser sans frottement.

Nous avons fixé les tubes de fer centraux sur lesquels repose la pièce centrale qui tient le servo moteur et le stylo. La pièce centrale peut alors bouger sur les deux axes X et Y. Et grâce au servo moteur on peut lever et descendre le stylo.

2.2.2. Assemblage des pièces

Concernant l'assemblage, nous avons tout d'abord pris la planche de bois la plus plate possible que nous avons trouvée et ensuite fait des calculs pour bien placer nos différentes boîtes contenant les moteurs et les différents bouts de bois qui permettaient de fixer les roulements aux différentes extrémités de notre dispositif. Ensuite nous avons fixé les pièces entre elles. Nous les avons vissées ou collées en fonction de leur usage.

2.3. Problèmes rencontrés

Nous avons rencontré plusieurs types de problèmes sur la partie mécanique. Le premier s'est présenté lors du fonctionnement des moteurs ou la force produite sur les roulements était trop forte pour ceux-ci. En effet nous n'avons eu que de petits roulements qui permettaient d'être fixés qu'à l'aide de petites visses. Toutefois la force était conséquente et les visses qui tiennent les roulements ne supportent pas cette force.

Ensuite on a eu des problèmes sur la hauteur de ces roulements et sur leurs positionnement en effet il fallait que ceux-ci soient à la même hauteur pour que le câble puisse coulisser de manière fluide et que les frottements ne soient pas trop importants.

Enfin la planche que nous avions à disposition n'était pas totalement droite ce qui

induit que le stylo à ne pas exercer la même pression en différents points, en effet à certains endroits sa position est trop proche de la plaque et à d'autres elle est trop loin ce qui engendre des dessins pas très précis.

2.4. Bilan

Pour conclure cette partie mécanique, nous pouvons nous rendre compte que pour avoir un tracé précis et fluide on requiert des pièces extrêmement précises sur tout le dispositif. En effet, une seule pièce mal conçue ou pas aux bonnes dimensions va engendrer un déséquilibre et un mal fonctionnement. Et ainsi le bois s'avère être la solution la plus pratique et la plus facile d'accès mais pas forcément la plus précise et fiable, en effet l'utilisation de fer par exemple nous aurait permis d'avoir des pièces plus précises.

3. Moteurs Pas à Pas

3.1. Présentation des modules

Les moteurs sont les éléments les plus importants pour le mouvement du robot. C'est pour cela que nous avons choisi des moteurs pas à pas **NEMA 17** qui produisent un mouvement extrêmement précis. Ceux-ci sont alimentés avec 12V.

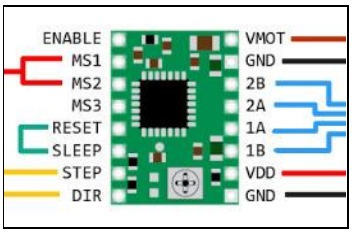
Pour les contrôler nous utilisons des drivers **A4988** qui nous permettent de produire 1 pas ou ½, ¼, ⅛ ou 1/16 de pas par pulsation envoyée. L'ampleur du pas peut donc varier en fonction de la précision voulue.

MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

Par exemple, lorsqu'on veut dessiner des petits dessins, on utilise une fraction de pas très petite, mais lorsqu'on doit se déplacer au-dessus de la feuille sans dessiner on peut utiliser un pas plus grand pour augmenter la vitesse du robot.

3.2. Travail effectué

Tout d'abord, il faut savoir que chaque moteur pas à pas bipolaire a 4 câbles reliés deux à deux. Pour localiser les paires il y a deux méthodes, soit utiliser un voltmètre, soit faire un circuit fermé avec une led qui s'allumera en présence du courant généré en tournant le moteur manuellement.



Après ça, le câblage des drivers selon le cours sur le moteur a été très rapide et n'a pas posé de problème particulier. Ce schéma le résume simplement:

Finalement au niveau du code, on a fait "évoluer" les fonctions. C'est à dire que nous avons commencé par une simple fonction ***oneStep(int stepPin)***, qui réalise une pulsation au pin '*step*' du driver, puis on a appelé cette fonction dans la fonction ***move(int steps1, int steps2)*** qui réalise un certain nombre de pas pour le moteur nº1 et nº2 passés en paramètres. Ainsi de suite, on a continué avec ***moveRel(int x, int y)*** en introduisant les coordonnées cartésiennes et finalement ***moveAbs(int x, int y)*** qui bouge vers un point précis du repère avec un mouvement rectiligne.

3.3. Problèmes rencontrés

3.3.1. Bruitage

Lorsqu'on connectait les deux moteurs à la fois, on constatait qu'ils bougeaient tout seuls. En effet, en passant la main au-dessus du câblage ils s'arrêtaient, puis en l'enlevant ils continuaient. Ce bruit provenait d'une erreur de code et non pas de câblage. La librairie qu'on utilisait (AccelStepper) n'initialise pas le ***pinMode(OUTPUT)***, et donc l'arduino n'isolait pas les pins et laissait passer des bruits.

3.3.2. Rotation simultanée

Un autre problème qui provient de la librairie AccelStepper (et la raison principale pour qu'on arrête de l'utiliser) et l'impossibilité d'utiliser les deux moteurs à la fois. Après recherche, on décide de créer les fonctions depuis le début. Grâce à ça nous gagnons aussi en espace mémoire.

3.3.3. Rotation pas fluide

La rotation des moteurs était très brusque et pas du tout fluide. Même en

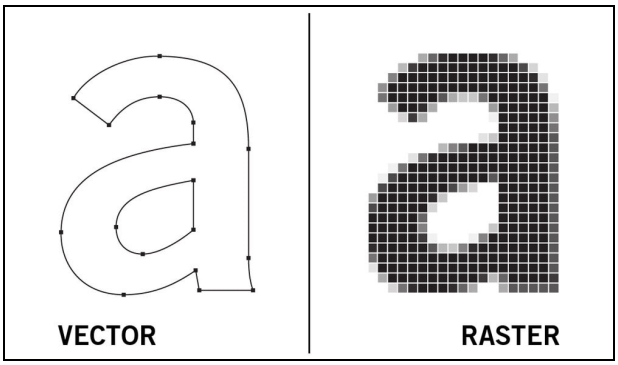
réduisant le délai entre chaque pas et en ajustant correctement le potentiomètre du driver le problème persistait. Finalement, on réalisa qu'on appelait la fonction ***Serial.print()*** afin de faire des tests. Cependant celle ci prend un temps considérable à s'exécuter, ce qui ralentissait les moteurs. En enlevant tous les appels à cette fonction le problème se résout.

3.4. Bilan

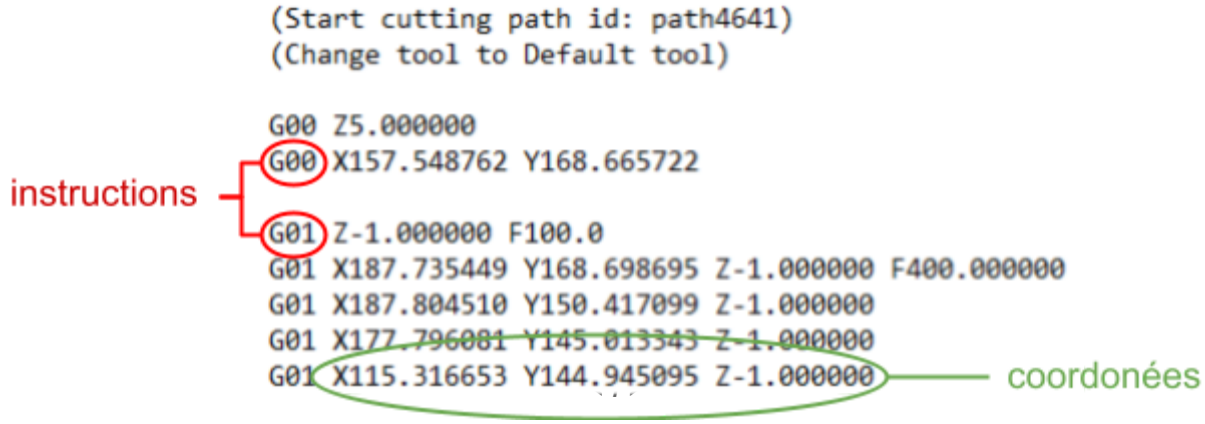
On a pu apprendre qu'il vaut mieux commencer par des fonctions précises du code, vérifier qu'elles fonctionnent correctement, puis construire la structure générale autour de celles-ci. De cette façon on peut réaliser des tests plus précis et les erreurs ne s'accumulent pas.

4. Transformation d'Images

Afin que le robot puisse savoir comment dessiner une image, il faut d'abord la traiter et la transformer dans un langage qu'il puisse comprendre. Cette partie est dédiée à ce processus : transformer un fichier jpeg, png, … , en image vectorielle puis en G-Code.



Le G-Code est composé d'instructions et de coordonnées et est utilisé dans la majorité des machines CNC. Voici un exemple :

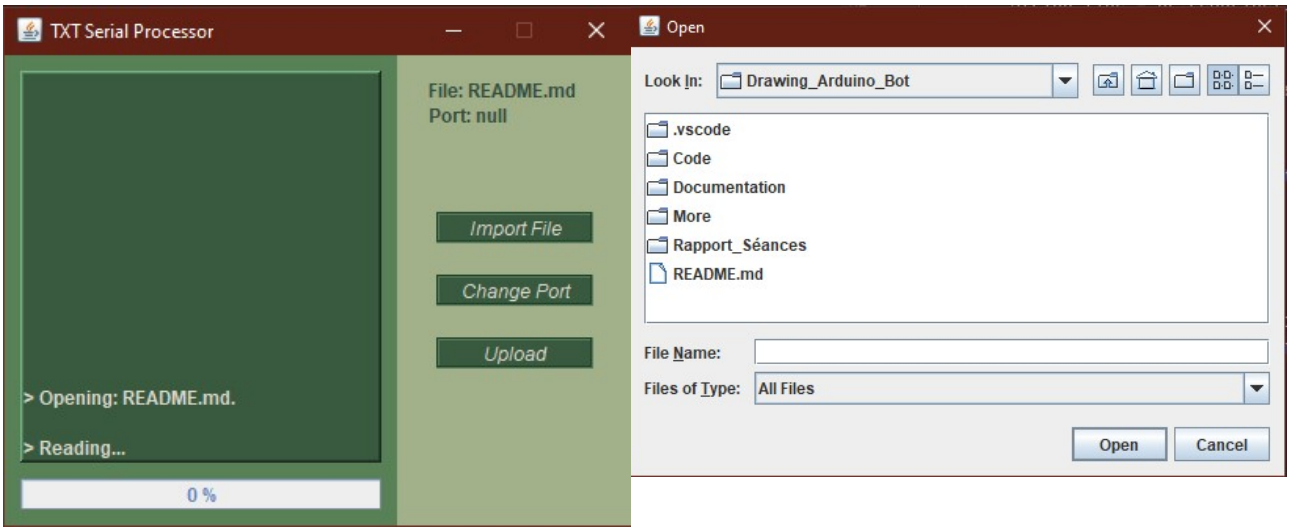


Pour obtenir ce fichier à partir d'une image on utilise l'application **Inkscape** et on peut facilement exporter l'image. Voici un petit tutoriel fait par nous au cas où quelqu'un serait intéressé.

5. Communication Serial

5.1. Présentation des modules

Afin d'envoyer le fichier G-Code vers l'arduino on a codé une application Java qui lit un fichier local et envoie l'information à travers le port USB.



5.2. Travail effectué

Afin de créer cette interface on a utilisé la librairie **Java Swing** qui est très utilisée. La partie front-end n'a pas été compliquée, avec quelques tutoriels les éléments principaux étaient codés.

La communication serial est plus complexe. D'abord on lit et stocke dans une liste toutes les lignes du fichier G-Code en supprimant les lignes innécessaires comme les commentaires entre parenthèses. Puis, grâce à la librairie **gnu.io** on utilise le **outputStream** du port connecté à l'arduino. On va pouvoir alors envoyer les lignes de commandes par intervalles.

Puis, la dernière étape a été de créer un parser dans le code de l'arduino qui

puisse convertir les lignes de texte G-Code en instructions. On a donc créé une classe **Instruction** qui est instanciée avec une ligne de texte. Puis les attributs coordonnées sont sauvegardés et peuvent ainsi être utilisés par la fonction ***moveAbs()***.

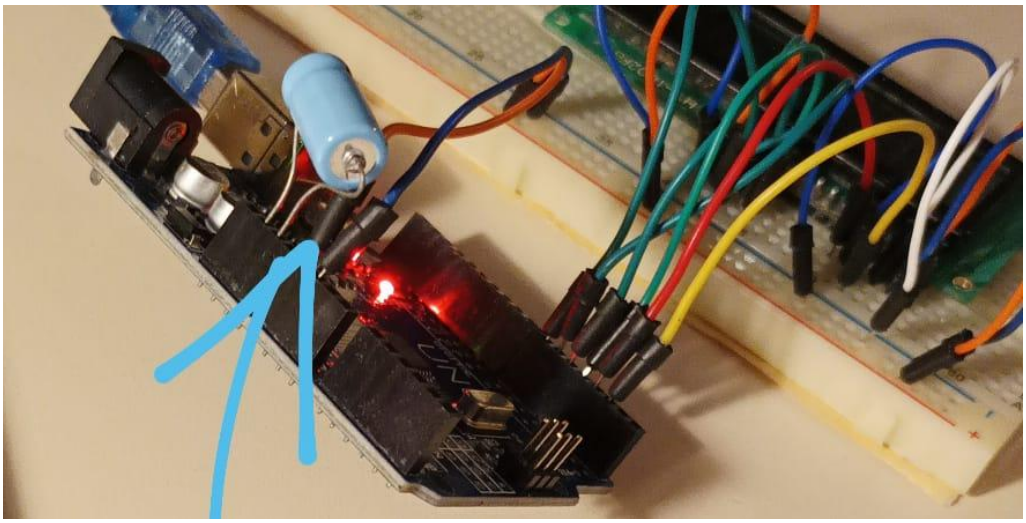
5.3. Problèmes rencontrés

5.3.1. Librairies

Afin d'envoyer les lignes du fichier G-Code à travers le port USB, il a fallu utiliser une librairie Java pour se connecter plus facilement. Le problème a été que beaucoup des librairies soit ne fonctionnent uniquement qu'avec une certaine version windows (**RXTX**), soit l'installation des dépendances en Java crée des bugs entre les différent fichiers sources du JDK (**JSerialComm**), soit les ports USB ne sont pas détectés (aussi **JSerialComm).** Avec l'aide d'un camarade SI5 qui m'a aidé avec l'installation, j'ai réinstallé une version stable du JDK Java puis on a importé la librairie **gnu.io** à travers **Maven**. La librairie a finalement bien fonctionné et on pouvait envoyer du texte vers l'arduino.

5.3.2. Reset de l'Arduino

Chaque fois qu'on envoie du texte vers l'arduino, l'application Java envoie un signal LOW puis commence le transfert de données. L'arduino comprend ce signal LOW comme une instruction pour reset et recommencer le programme. Ceci est problématique puisque l'information n'est donc pas transmise et le dessin devrait recommencer à partir de zéro. En cherchant on trouva qu'il y avait un pin *reset* sur l'Arduino Uno, on a finalement juste mis un condensateur entre le 5V et le pin *reset*, ce qui résout le problème.



6. Conclusion

Pour conclure, nous aurions dû avancer plus rapidement sur la partie mécanique en se concentrant uniquement sur celle-ci pour pouvoir ensuite trouver et résoudre les problèmes liés au code auxquels nous allions faire face. Ne pas faire ceci a provoqué que les délais soient largement dépassés et qu'on ai pris beaucoup de retard au début.

Même si le projet n'a pas pu aboutir à ce dont on avait initialement pensé, nous avons pû tirer beaucoup de leçons à un niveau aussi technique que personnel.

Impressions de JAIME : Ce projet m'a permis de me rendre compte non seulement de l'importance d'être organisé, mais aussi de l'importance d'une cohésion d'équipe. En effet, chacun peut apporter un point de vue différent qui peut améliorer le projet. Je me suis rendu compte que ces projets où chacun développe un aspect me passionne et me donne envie de m'investir dans d'autres projets plus grands. Ce projet n'a pas été comme ça mais j'espère en trouver un où l'équipe soit aussi motivée que moi.

Impressions d'ADRIEN : Pour conclure, je dirais que j'ai découvert un plaisir de construction et de création. En effet j'ai beaucoup aimé cette recherche et ce développement et notamment le fonctionnement de la partie mécanique. Voir comment les moteurs marchaient et comment le système allait fonctionner. J'ai aussi découvert un certain intérêt à la partie électronique avec le fonctionnement de la carte arduino sur les moteurs, comment le servomoteur fonctionne pour actionner la montée et la descente du stylo.