Hes·so VALAIS WALLIS
School of Engineering

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Filière Systèmes industriels

Orientation Power & Control

Travail de bachelor
Diplôme 2021

Yohan Aymon

Robot SCARA

- *Professeur*
  Prof. Fariba Moghaddam
- *Expert*
  Serge Lillo
- *Date de la remise de rapport*
  20.08.2021

HES-SO Valais

SYND	ETE	TEVI
X	X	X

Données du travail de diplôme
Daten der Diplomarbeit

FO 1.2.02.07.CB
haj/11/01/2016

Filière / Studiengang **SYND**	Année académique / *Studienjahr* **2020/21**	No TD / *Nr. DA* **PC/2021/62**
Mandant / *Auftraggeber* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*	Etudiant / *Student* **Yohan Aymon** Professeur / *Dozent* **Fariba Moghaddam**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja [1] ☒ non / nein	Expert / *Experte (données complètes)*	

Titre / *Titel*

Robot Scara

Description / *Beschreibung*

Dans le cadre d'un travail de diplôme en 2020, un modèle de robot SCARA capable de dessiner sur une feuille A4 a été modélisé sous forme de jumeau numérique à l'aide des logiciels Siemens PLCSIM, SIMIT et NX MCD. Une fois simulé, l'homologue réel du robot SCARA a été construit et monté.

Ce projet de diplôme propose d'améliorer le robot SCARA au niveau mécanique : châssis, motorisation, capteurs, préhenseur, etc. On demande également d'améliorer son interface IHM et d'optimiser les différents algorithmes d'initialisation et de régulation du robot. Enfin, on vise la gestion dynamique des bras du robot pour obtenir un mouvement fluide et rapide tout en tenant compte des paramètres de sécurité.

Objectifs / *Ziele*

- Se familiariser avec le robot Scara existant et ses différents outils de commande et régulation.
- Améliorer certains éléments de construction de la maquette (Châssis, support des moteurs, courroies, amplificateurs 4 quadrants, ...) et optimiser le fonctionnement du robot an ajoutant de nouveaux composants (capteurs, préhenseur, actionneur, ...).
- Optimiser l'initialisation du robot ainsi que les différents régulateurs de vitesse et de position.
- Développer des algorithmes pour la gestion dynamique et fluide des bras du robot par rapport à la trajectoire à dessiner.
- Améliorer l'interface IHM du robot.

Signature ou visa / *Unterschrift oder Visum*	Délais / *Termine*
Responsable de l'orientation / filière *Leiter der Vertiefungsrichtung / Studiengang:* F. Moghaddam	Attribution du thème / *Ausgabe des Auftrags:* **10.05.2021** Présentation intermédiaire / *Zwischenpräsentation* **07 – 08.06.2021** Remise du rapport / *Abgabe des Schlussberichts:* **20.08.2021, 12:00** Exposition / *Ausstellung der Diplomarbeiten:* **25 – 27.08.2021** (si autorisé / *falls genehmigt*) Défense orale / *Mündliche Verfechtung:* **30.08 – 09.09.2021**
[1] Etudiant / *Student* :	

[1] *Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme.*
Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.

Rapport reçu le / *Schlussbericht erhalten am* ..... Visa du secrétariat / *Visum des Sekretariats* .....

Robot SCARA

Diplômant Yohan Aymon

Objectif du projet

Le but de ce travail de bachelor est de reprendre le robot SCARA réalisé lors d'un travail de diplôme en 2020 et de l'optimiser au niveau mécanique, système d'entraînement, commande et interface homme-machine. Ce bras articulé a pour but d'effectuer des dessins sur une feuille A4.

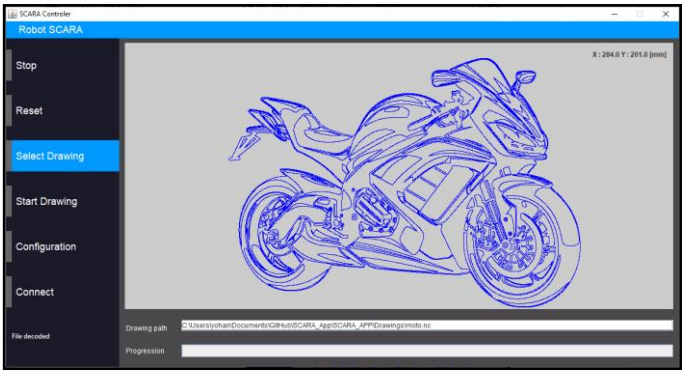
Méthodes | Expériences | Résultats

Un robot SCARA (**S**elective **C**ompliance **A**ssembly **R**obot **A**rm) est un type de robot industriel reprenant la morphologie d'un bras humain. Ces machines sont régulièrement retrouvées dans de nombreux domaines où il est nécessaire d'effectuer des mouvements Pick-and-place ou encore des trajectoires précises.

L'installation existante a été améliorée en créant un support pour le robot, en réduisant les jeux mécaniques, ainsi qu'en concevant un préhenseur motorisé. Le système d'entraînement du bras a également été revu en modifiant le type de moteur utilisé ainsi que son électronique de commande. Le nouveau système assemblé est ainsi plus précis que le précédent.

Au niveau logiciel, un premier programme, fonctionnant sur un ordinateur, permet de piloter et superviser le robot avec une interface homme-machine. Ce programme décode le dessin sélectionné par l'utilisateur et transmet toutes les informations essentielles pour commander le robot et suivre son fonctionnement.

Un second programme sur un automate de chez Siemens, reçoit les commandes du PC et pilote le robot en conséquence.



Interface homme-machine permettant de piloter et superviser le robot SCARA.

Travail de diplôme
| édition 2021 |

Filière
Systèmes industriels

Domaine d'application
Power & Control

Professeure responsable
Fariba Moghaddam
Fariba.Moghaddam@hevs.ch

Table des matières

Table des figures

Liste des tableaux

Listes des acronymes et glossaire

Acronymes

CAO Conception Assitée par Ordinateur 40

CNC Computer Numerical Control 38

HMI Human-Machine Interface 2

IP Internet Protocol (voir glossaire IP) 30

MGI Modèle géométrique inverse 12, 13, 61

NEMA National Electrical Manufacturers Association 11

PC Personal Computer 29

POO Programmation Orientée-Objet xiv

PTO Pulse Train Output 10

PWM Pulse Width Modulation 22

SCARA Selective Compliance Assembly Robot Arm 1

TCP Transmission Control Protocol (voir glossaire TCP) 30

Glossaire

Classe En POO, structure permettant de stocker en mémoire des attributs et des méthodes 43

Closed loop En français boucle fermée, se dit d'un système régulé à l'aide d'un retour d'information sur la sortie du système, comme par exemple un capteur 11, 15

Driver Terme anglais pour électronique de commande de moteur 7

GCode Langage de programmation pour machine-outil à commande numérique 39

IP *Une adresse IP (avec IP pour Internet Protocol) est un numéro d'identification qui est attribué de façon permanente ou provisoire à chaque périphérique relié à un réseau informatique qui utilise l'Internet Protocol. L'adresse IP est à la base du système d'acheminement (le routage) des paquets de données sur Internet.* [1] xiii

Microstepping Mode de pilotage des moteurs pas à pas 10, 34

Méthodes En POO, nom donné à une fonction membre d'une classe, parfois utilisé comme synonyme de fonction 51

Objet En POO, un objet est une classe qui a été instanciée. On utilise ce mot comme synonyme d'instance. 43

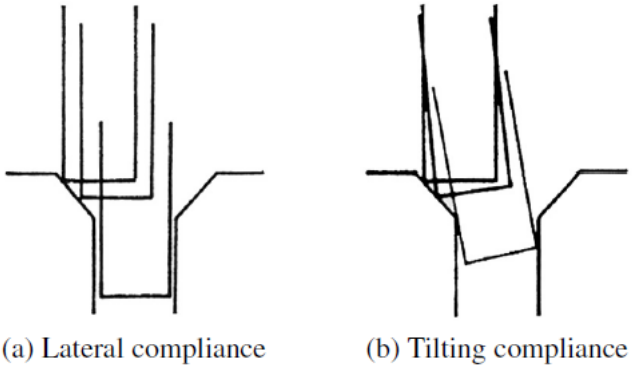
TCP *Transmission Control Protocol (littéralement, « protocole de contrôle de transmissions »), abrégé TCP, est un protocole de transport fiable, en mode connecté.* [2] xiii

Thread En informatique, fil d'exécution ou processus 51

1 Introduction

1.1 Le robot SCARA

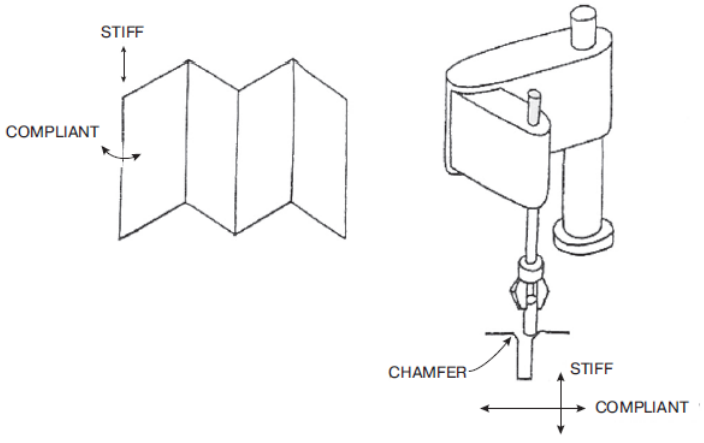
Les SCARA sont un type de robot industriel inventé en 1978 par le professeur japonais Hiroshi Makino [3]. Ce système devait, initialement, résoudre un problème précis : « Comment déposer une pièce dans un trou si cette dernière n'est pas bien alignée ? ».



(a) Lateral compliance (b) Tilting compliance

Figure *1.1 – Type de défauts d'alignement [3]*

Ce professeur est venu avec l'idée de développer un robot rigide sur l'axe 'Z', c'est-à-dire l'axe vertical, mais flexible sur les deux autres axes. Son premier design a été une structure de la forme d'un « byobu », une sorte de paravent japonais. Dans ce cas-là les charnières de la paroi représentaient les articulations et les cadres les bras. Avec ce concept, le robot pouvait ainsi effectuer un positionnement précis sur un plan en deux dimensions tout en ayant une possibilité de réalignement du bras en cas de décalage.



(a) A Japanese *byobu* (b) SCARA

Figure *1.2 – Comparaison byobu et robot SCARA [3]*

Un robot dit SCARA vient de l'acronyme anglais « Selective Compliance Articulated Robot Arm ». Cette définition pourrait être traduite en français par : bras robotisé ayant un système adaptable aux différents défauts d'alignement (flexibilité mécanique). Aujourd'hui, ce type de robot est largement utilisé dans les chaînes d'assemblage. Il a comme principal avantage de pouvoir réaliser un positionnement rapide tout en ayant une fixation au sol minime. De plus, il permet d'obtenir une très bonne répétabilité du mouvement par rapport aux autres types de robot comme les cartésiens [4]. Ces différents points ont également permis à ce concept de se développer dans d'autres domaines comme l'impression 3D, la soudure et la gravure laser.

1.2 But du projet

Dans le cadre d'un travail de diplôme en 2020 de la filière systèmes industriels P&C, un robot SCARA a été modélisé en jumeau numérique [5]. Après avoir terminé le travail de simulation, l'étudiant a monté le robot réel et validé son projet par différents essais. Ce robot doit être capable de dessiner sur une feuille de papier A4. Il aura comme objectif principal d'être utilisé comme démonstrateur pour la HES-SO Valais/Wallis lors de portes ouvertes par exemple.

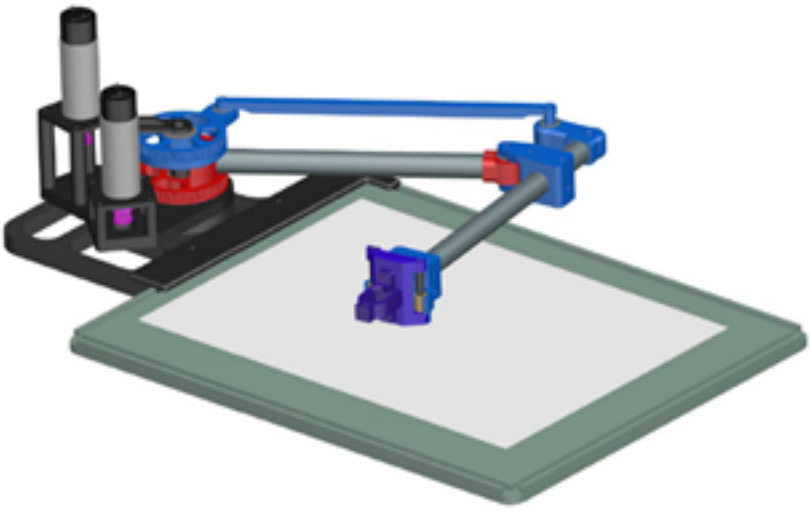


Figure *1.3 – Robot SCARA modélisé sous NX [5]*

Le travail précédent n'ayant pas abouti à un robot entièrement fonctionnel, ce projet reprend donc la suite de la conception. Ce diplôme aura comme objectif d'améliorer le système existant.
Plusieurs éléments seront à revoir : les jeux mécaniques, la motorisation, les capteurs et le préhenseur du robot. Au niveau software les algorithmes de régulation, l'interface HMI et l'initialisation du robot devront être également modifiés, le but étant d'obtenir un robot effectuant des mouvements fluides et rapides en dessinant sur une feuille A4.

Concrètement les objectifs du projet seront :

- se familiariser avec le robot SCARA existant et ses différents algorithmes de commande et régulation ;
- améliorer les éléments du robot (Châssis, support des moteurs, courroies, amplificateurs 4 quadrants, etc. . .) ;
- optimiser l'initialisation du robot et ses algorithmes de régulation de vitesse et position ;
- développer des algorithmes pour la gestion dynamique et fluide des bras du robot par rapport à la trajectoire à dessiner ;
- améliorer l'interface homme machine.

Les éléments mentionnés ci-dessus ont évidemment évolué durant l'avancement du projet. N'ayant pas un cahier des charges fixe, plusieurs choix arbitraires ont été réalisés. Les différentes modifications et décisions seront énoncées au cours du travail.

1.3 Organisation du projet

Le projet a été organisé en fonction des délais de livraisons. Les envois étant relativement lents chez certains fabricants, l'électronique et les moteurs ont été choisis en premier. La partie mécanique ayant des délais plus courts, a été placée en seconde. Finalement, la programmation du robot ainsi que son interface seront réalisées en fin de projet. Suivi d'une description générale de l'installation initiale, c'est dans cet ordre que sera présenté ce rapport. Le planning complet peut être trouvé en annexe B.

2 Description du système initial

Ce projet reprenant un travail précédent, une description du système à améliorer permettra de comprendre la base du travail.

2.1 Concept mécanique

Le robot SCARA monté à la HES s'inspire du modèle équivalent trouvé sur le site Thingverse [1]. Il consiste en un bras mobile pouvant effectuer des mouvements dans un plan en deux dimensions. A l'extrémité du bras, un préhenseur permet de fixer un stylo ou un crayon offrant ainsi la possibilité de réaliser des traits sur une feuille. Deux moteurs DC, montés sur la base principale, entraînent chacun une partie du bras du robot. Au centre de cette dernière, un axe central constitué d'une vis et de deux roues dentées montées sur roulement, crée la liaison entre le bras et les pignons des moteurs.

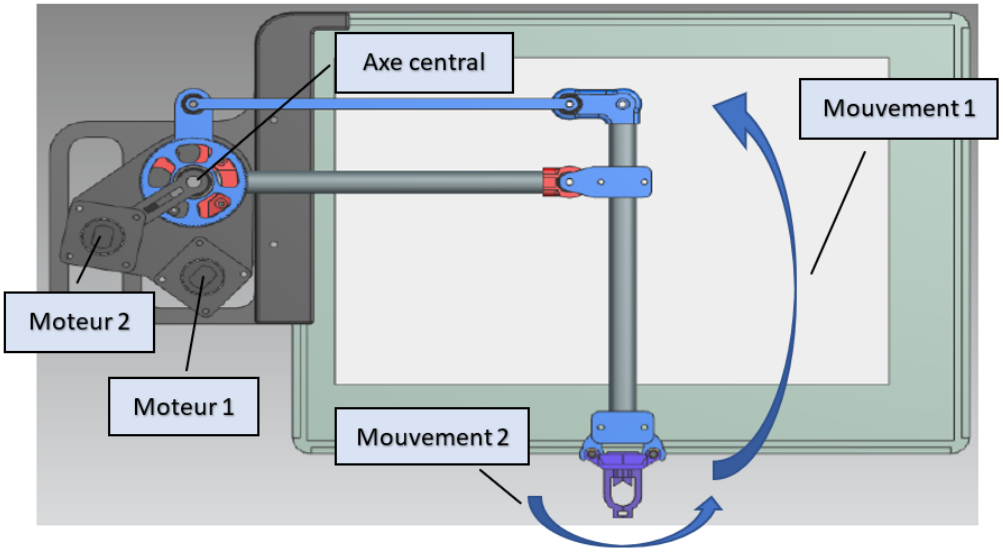


Figure *2.1 – Robot SCARA, vue de dessus (ancien montage) [5]*

2.1.1 Motorisation

Initialement, durant le travail de diplôme précédent, les moteurs DC avaient été achetés chez Faulhaber. Le choix s'était porté sur les SR2232. Ces derniers étaient combinés avec le réducteur du même fournisseur. Ce réducteur, ne « possédant pas » de jeu angulaire, permettait d'effectuer un positionnement précis du bras robotisé en cas de mouvement aller-retour.

1. ThingVerse https://www.thingiverse.com/thing:3096135 (accessed : 19.05.2021)

Figure *2.2 – Moteur DC et réducteur*

Moteur	Fabricant	Faulhaber
	Dénomination	2232 SR moteur DC
	Couple nominal	10 [mNm]
Réducteur	Fabricant	Faulhaber
	Dénomination	Réducteur à étage 22/5
	Rapport de réduction	1/69.2

Table *2.1 – Moteur DC travail de bachelor 2020*

Arbitrairement, le moteur numéro 1 définit la machine tournante faisant effectuer au préhenseur un arc de cercle vertical. De ce fait, le moteur numéro 2 est celui permettant de réaliser des arcs horizontaux (cf. figure 2.1). En combinant les 2 positions des moteurs, il est possible d'accéder à toute la surface d'une feuille A4. En revanche, avec la configuration comme présentée dans la figure n°2.3 plusieurs zones de la feuille restaient inaccessibles [5].

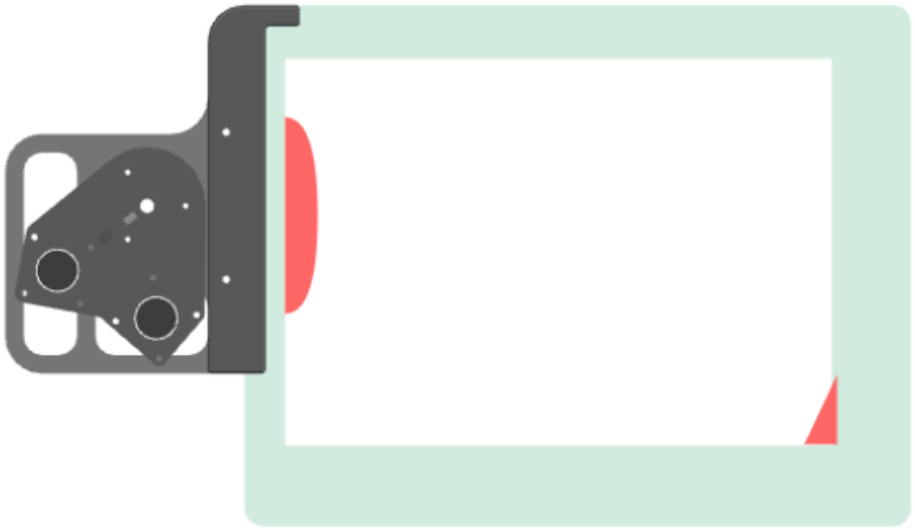


Figure *2.3 – En rouge, positions inatteignables [5]*

2.2 Régulation et interface

Au niveau de la commande, le système motorisé initial était piloté par deux drivers deux quadrants de chez Faulhaber. Dans ce rapport, le terme Driver définira l'électronique de commande permettant de piloter un moteur.
Une régulation en cascade avait été réalisée sur les deux moteurs entraînant le bras. Le courant et la vitesse étaient régulés à l'interne de l'électronique. L'automate programmable S7-1500 de chez Siemens supportait la régulation de position.

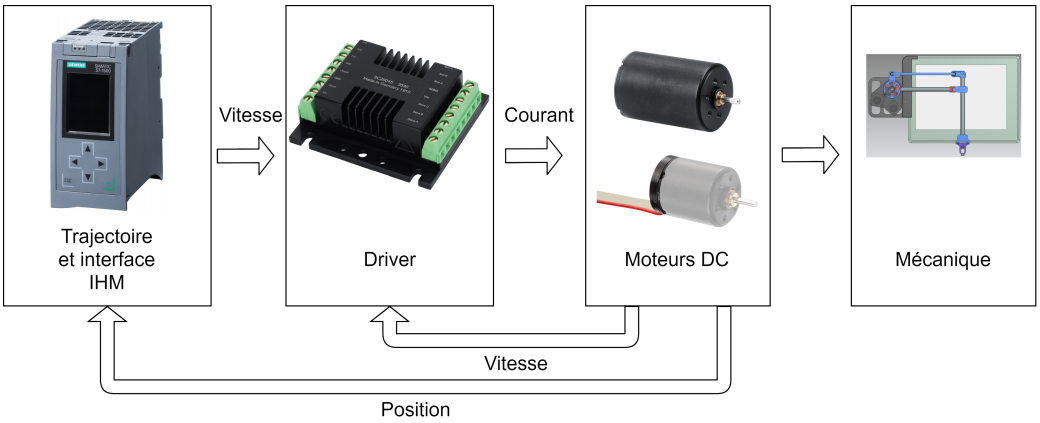


Figure *2.4 – Schématisation de l'ancien système SCARA*

L'interface IHM permettait à l'utilisateur de pouvoir observer le comportement du robot mais également de choisir le type de dessin à réaliser. Ainsi, des coordonnées de dessin étaient générées puis transformées en coordonnées polaires pour les moteurs.

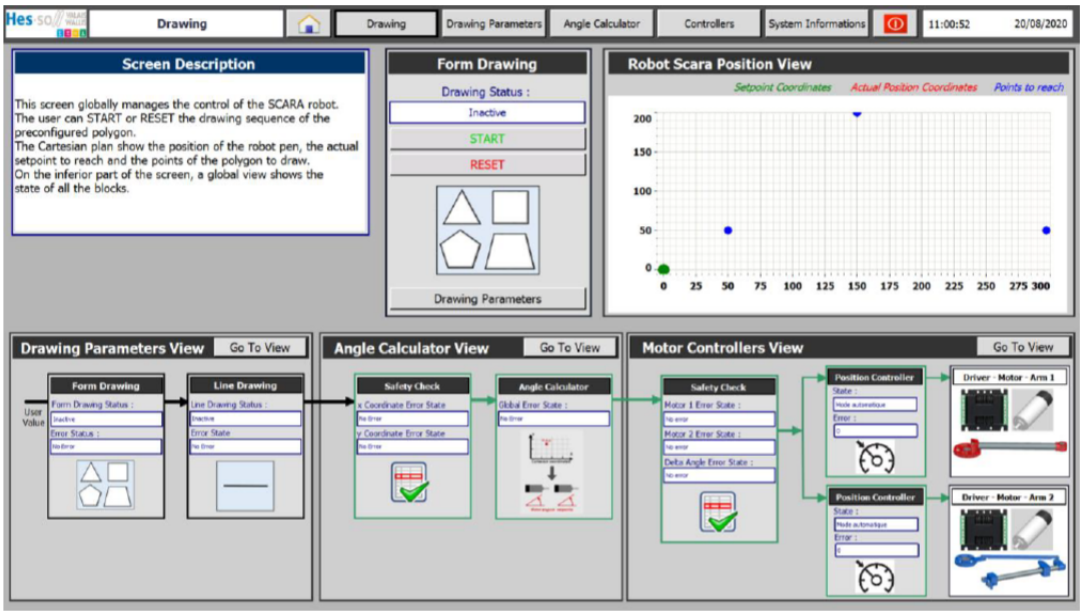


Figure *2.5 – Ancien IHM*

3 Motorisation et électronique de commande

La première amélioration à effectuer concernait le changement des drivers deux quadrants de chez Faulhaber. Deux possibilités principales ont été envisagées. La première était le changement des drivers, la seconde le remplacement des moteurs.

Raisons du changement

Les drivers du projet précédant n'offraient pas assez de dynamisme pour l'application choisie et devaient donc être remplacés. En effet, les contrôleurs de vitesse ne pouvaient que produire des mouvements dans deux quadrants c'est-à-dire des vitesses et couples de même signe. En étudiant le fait que l'on voudrait ralentir sans changer de sens un moteur, il est impossible à un système deux quadrants de l'effectuer de manière fluide. Le driver n'appliquera simplement plus de couple et le moteur freinera avec les frottements. L'autre possibilité est d'appliquer sur le moteur un couple négatif créant alors de grandes variations de vitesses et de couple sur le moteur. En résumé, il est bien plus difficile d'effectuer un contrôle du système sans moyen de freinage maitrisé du moteur.

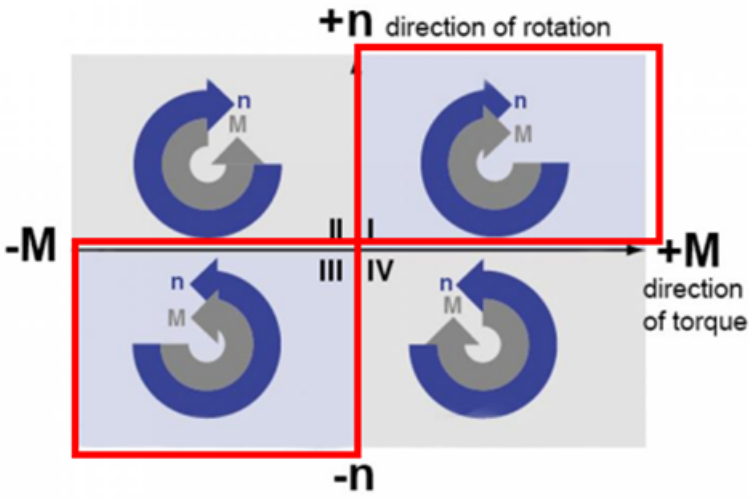


Figure *3.1 – Quadrants de couple et vitesse [6]*

Il a donc été proposé d'utiliser des convertisseurs fonctionnant sur quatre quadrants permettant alors le contrôle total de la vitesse et du couple dans les deux sens de rotation. Étant très onéreux et pour d'autres avantages cités dans le point suivant, cette idée a rapidement été abandonnée. Il a donc été décidé de changer de fournisseur pour les moteurs.

3.1 Motorisation

Un nouveau choix de moteur a donc été envisagé. Dans le cadre de ce projet, le concurrent principal de la machine DC était le moteur pas à pas. Une rapide comparaison des avantages de chacun, permet aisément de comprendre que ces derniers sont un très bon compromis.

	Moteur pas à pas	Moteur DC
Positionnement précis	Aucune régulation nécessaire et microstepping	Régulation complexe et réducteur
Erreur de positionnement	Boucle fermée	Boucle fermée
Coûts	Bon marché	Onéreux
Mesure de position	Non nécessaire	Avec observateur
Mouvement fluide	Microstepping	Driver 4 quadrants
Génération de trajectoire	Simple (listes de positions)	Complexe (fonctions continues)

Table *3.1 – Comparaison moteur pas à pas - moteur DC*

Les moteurs pas à pas possèdent de nombreux avantages simplifiant notamment la partie programmation du robot. De plus, les pièces mécaniques ne nécessitaient que peu d'adaptations. C'est cette possibilité qui a donc été retenue.
Pour dimensionner correctement le moteur, le choix a été basé sur la machine utilisée dans le travail précédent. En reprenant les paramètres du système moteur-réducteur Faulhaber, on obtient :

$$M_{requis} = M_{moteur} * r_{reducteur} = 10 * 10^{-3}\,[Nm] * 69.2 = 69.2 * 10^{-2}\,[Nm]$$

Il reste donc à trouver un moteur adapté ainsi que son électronique de commande.

3.2 Électronique de commande

Afin de bien comprendre le choix des drivers moteurs, deux notions doivent être développées :

— Le mode de pilotage des machines pas à pas

— Le Microstepping

Pilotage

La commande standard pour ces drivers est le train d'impulsions ou Pulse Train Output (PTO) en anglais. En recevant une impulsion, le driver fera effectuer une rotation de 1.8° à l'axe du moteur. Cette valeur de 1.8° est le pas standard pour ces machines tournantes. Afin de déplacer le moteur en continu, des séries d'impulsions seront donc générées.

Microstepping

La précision du positionnement sera créée par le microstepping. Il s'agit d'un mode de pilotage des moteurs pas à pas qui sera combiné à la commande standard. Initialement ces machines auront $360°/1.8° = 200[pas/revolution]$. En utilisant le microstepping, le driver réduira électriquement l'angle de chaque pas par un coefficient de division (par exemple 2, 4, 8 etc. . .). En prenant un cas réel, un moteur avec un coefficient 10 de microstepping n'aura plus 200 pas par révolution mais : $200*10 = 2000[pas/revolution]$ donnant un angle de pas de $1.8°/10 = 0.18°$. Ce mouvement sera effectué en pilotant le courant dans les deux bobines du moteur en même temps, créant ainsi un champ magnétique orienté entre les deux pas du moteur [7]. Cf. Figure 3.2

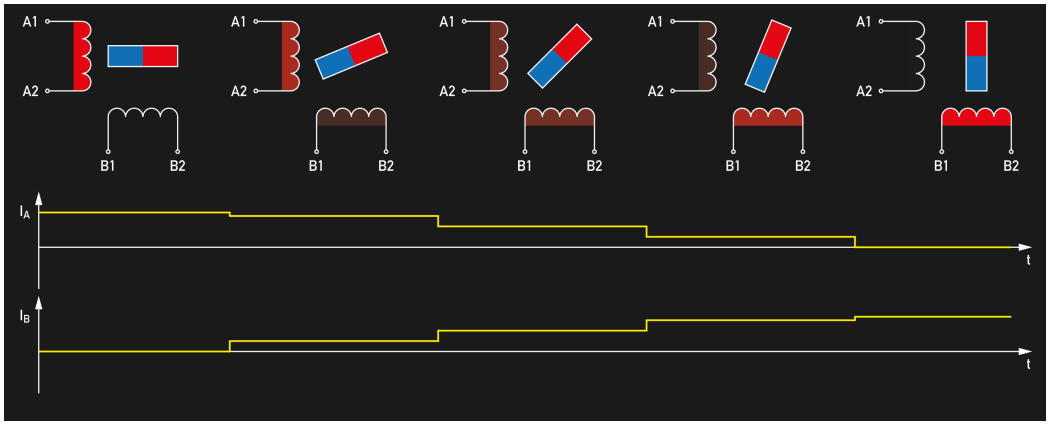


Figure *3.2 – Schématisation microstepping [7]*

Cette manière de piloter le moteur remplacera l'utilisation d'un réducteur évitant ainsi un achat supplémentaire et des jeux mécaniques indésirables. En revanche, un défaut important est à noter. Le microstepping ralentira et réduira le couple du moteur. Un compromis entre vitesse et précision devra être trouvé. Pour toutes ces raisons le moteur NEMA 17 72 Ncm de chez StepperOnline a été choisi (D.3).

Figure *3.3 – Moteur Nema 17 et driver CL42T*

Accordé au moteur, le choix du driver s'est porté sur le modèle CL42T de chez StepperOnline. Son interface permet la commande du moteur pas à pas par impulsions tout en effectuant du microstepping. Additionné au moteur, un encodeur de 1000 impulsions par tour est monté sur l'axe de la machine tournante. Le système Closed

loop du driver permet avec ce retour d'information d'empêcher le système de perdre des pas. Le tout étant livré en kit, de simples vérifications comme le courant de sortie du système et les moyens de communication ont été effectuées.

Calcul du coefficient de microstepping

Afin de bien comprendre le paramétrage des drivers, un aparté sur la partie logiciel doit être fait.
En omettant toute la programmation, le système sera géré de la manière suivante : Premièrement, des coordonnées cartésiennes représentant le dessin seront les consignes de trajectoire du robot ; Ensuite, ayant une construction mécanique différente d'un robot cartésien, une transformation mathématique devra être effectuée afin de trouver les angles de consigne à appliquer sur les moteurs. En robotique, cette partie-là est appelée modèle géométrique inverse abrégé MGI ; Finalement, ces angles seront transformés en train d'impulsions par un automate S7-1200 de chez Siemens. Ces dernières seront directement appliquées sur le driver. Comme expliqué précédemment, le CL42T s'occupera ensuite d'entraîner le moteur en fonction du nombre d'impulsions reçues.

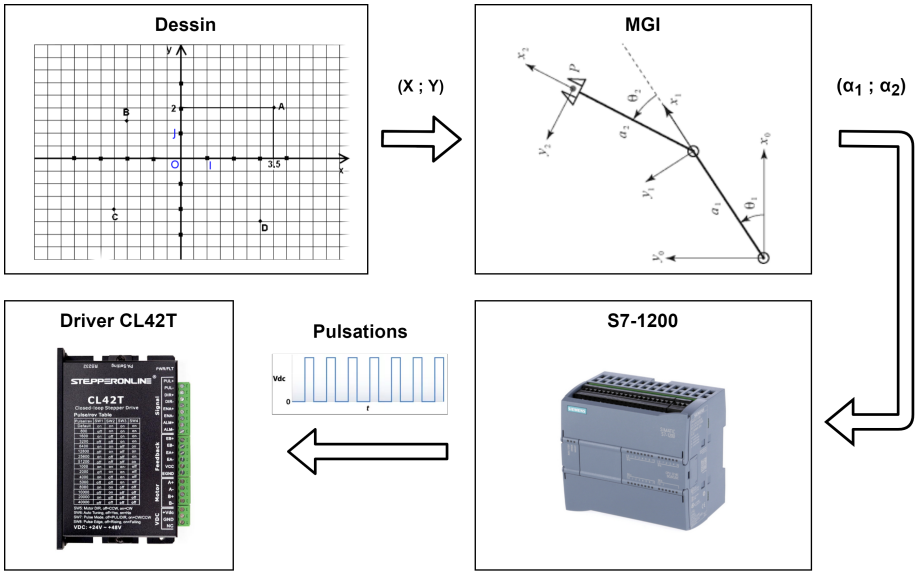


Figure *3.4 – Schéma de principe pilotage robot*

Si l'on fixe arbitrairement la résolution du dessin à un trait minimal de 0.5 mm, le modèle géométrique inverse doit être développé pour connaître le facteur de microstepping minimal qui doit être paramétré sur les drivers. Cette résolution a été choisie à l'aide d'une mesure approximative de la largeur d'un trait effectué par un stylo.

Le calcul du micro-pas minimal a été réalisé très tôt dans le projet. En effet, il devait permettre de vérifier si l'électronique de commande était correctement choisie. Le modèle géométrique inverse n'étant pas encore réalisé, une méthode équivalente a été utilisée. Les résultats suivants sont donc issus du développement en annexe D.1 et D.2. Le MGI sera développé au chapitre 5.5.2.

Le facteur de microstepping minimal calculé est de 550 pulses/rev. Plus tard dans le projet, après différents essais, il est apparu qu'un facteur plus important améliorait la qualité du dessin. C'est donc la configuration finale qui est présentée en paramétrage.

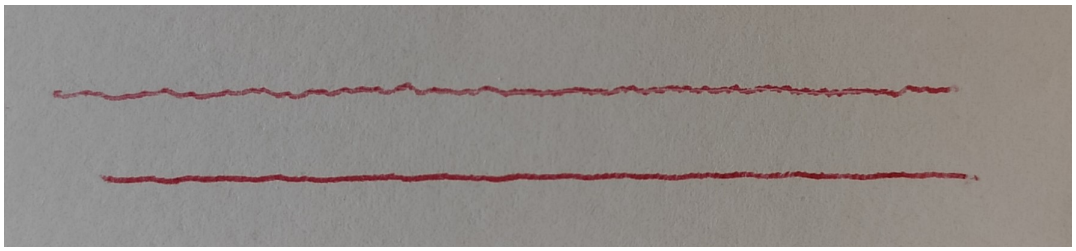


Figure *3.5 – Traits de 5 cm (non à l'échelle ici)*
1000 pulses/rev. (haut) 5000 pulses/rev. (bas)

La configuration finale choisie est donc de :

— 5000 pulsations / révolution ⇔ 0.072°/ pas du moteur

Cette valeur peut être immédiatement paramétrée dans les drivers contrôlant les moteurs pas à pas.

3.2.1 Paramétrage des drivers

Afin de régler correctement ces drivers, plusieurs éléments sont à modifier. Ces réglages sont effectués de deux façons, la première par des contacts sur le driver (PA settings) et la seconde avec le programme fourni chez StepperOnline et un câble RS232-USB (fourni avec les moteurs).

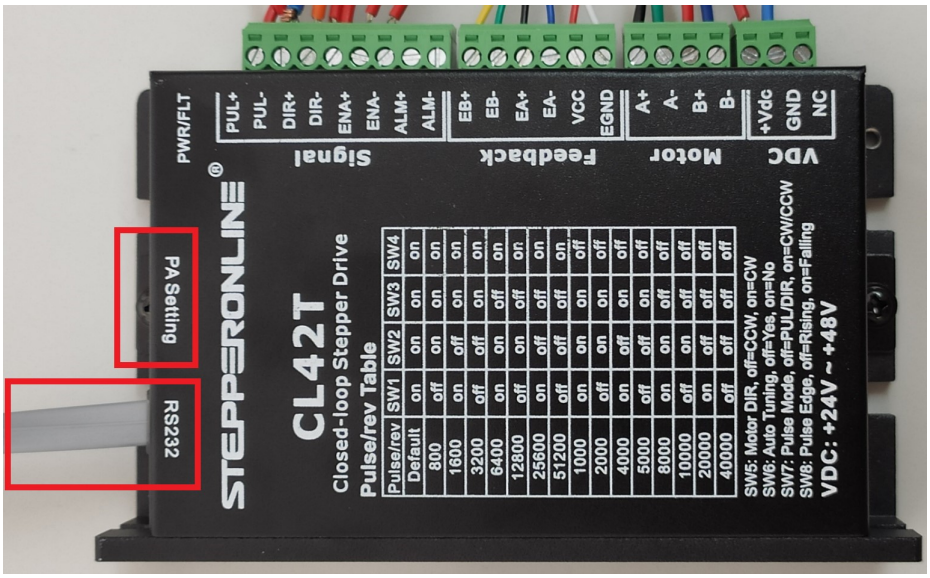


Figure *3.6 – Drivers et ports de paramétrage*

Contacts

Pour ce robot les drivers devront être paramétrés comme suit :

— Pulses/rev. 5000 : SW1 OFF // SW2 OFF // SW 3 ON // SW 4 OFF
  Le micro pas sera réglé sur 5000 (choix arbitraire)

— Mode CCW : SW 5 OFF
  Le moteur démarrera dans le sens contraire des aiguilles d'une montre

— Auto Tuning : SW 6 OFF
  Automesure des paramètres du moteur

— Pulse mode : SW 7 OFF
  Une entrée direction et une entrée déplacement

— Pulse edge : SW 8 OFF
  Déplacement du moteur sur flanc montant

Logiciel de configuration StepperOnline

Le logiciel fourni par StepperOnline [logiciel A] permet d'ajuster les derniers paramètres. La communication avec le driver est effectuée en branchant le câble RS232 vers USB fourni avec les drivers. Un onglet de navigation donne accès aux paramètres du système {n°1 et 2}. Ces derniers peuvent être modifiés directement mais devront impérativement être sauvegardés sur le driver en appuyant sur le bouton Write {n°4}.

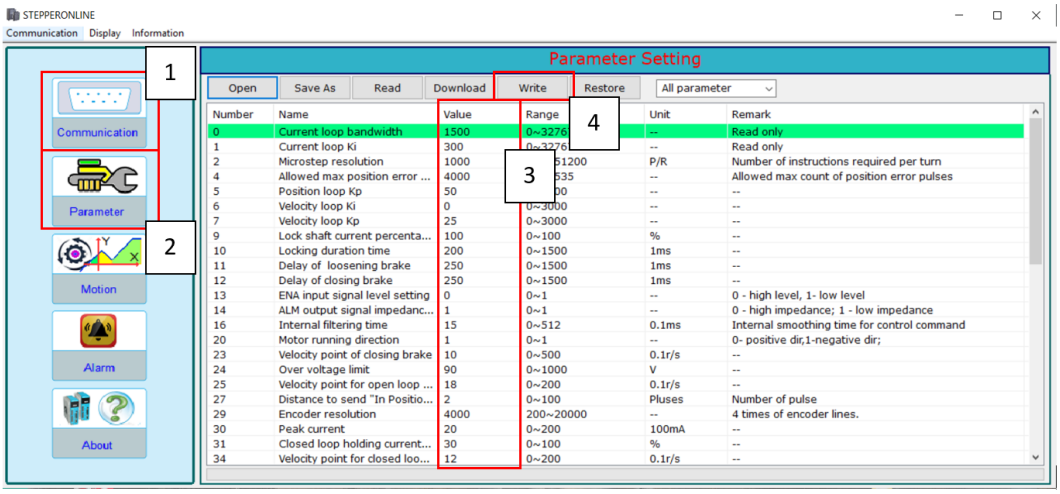


Figure *3.7 – Vue paramètre du programme StepperOnline*

Les paramètres {n°3} à modifier seront :

- **No 2** Micro-step resolution : 5000
- **No 13** Ena input signal level settings : 0 (High level)
  Enclenchement des moteurs à 24 V
- **No 30** Peak current 20
  Courant maximal de 2 A
- **No 31** Closed loop holding current percentage : 30
  Un courant 30% du courant maximal circule à l'arrêt des moteurs (blocage des moteurs)
- **No 59** Control mode : 2 (closed-loop)
- **No 61** Auto Tuning at power on : 1
- **No 63** Motion Type : 2 (Low speed circular interpolation motion)

Suite à une question posée chez StepperOnline, il s'est avéré que les autres paramètres de réglage affichés sur le programme, comme les coefficients de régulation, ne peuvent pas être modifiés et sont là à titre d'information.

4 Mécanique

La seconde partie est le design et l'amélioration mécanique du système. Les objectifs principaux sont de :

- créer un support pour le robot ;
- créer un système de mise à zéro ;
- créer un préhenseur motorisé ;
- optimiser la précision du système (jeux mécaniques).

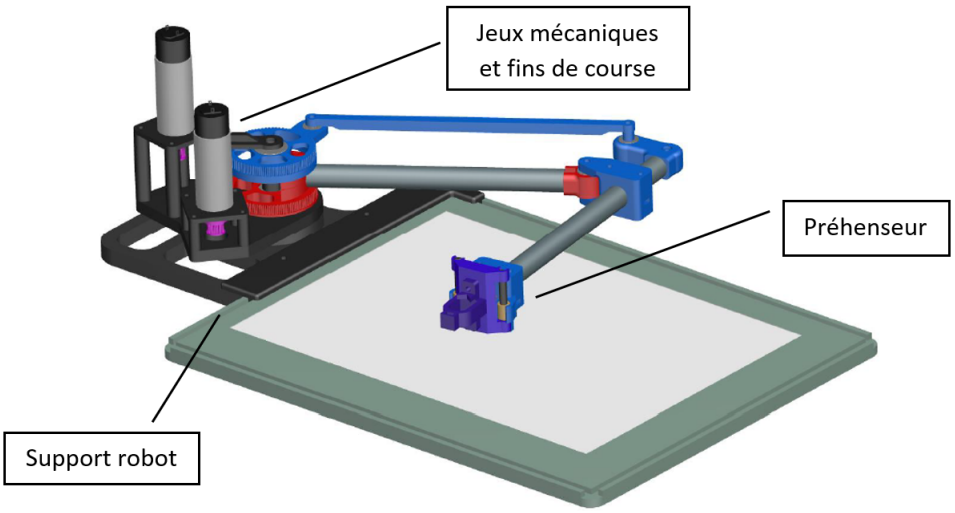


Figure *4.1 – Améliorations du robot*

4.1 Support robot

Toujours dans le but de l'organisation du travail, les pièces devant être réalisées en dehors de l'HES ont été priorisées. C'est pourquoi le boîtier du robot a été conçu en premier. Plusieurs idées de design ont donc été analysées. Celle retenue a été le boîtier en profilé Item intégré au support du robot. En effet, il présentait l'avantage d'être entièrement personnalisable en dimensions sans pour autant être imposant. De plus, tous les composants seraient intégrés, y compris l'automate pilotant le robot, proposant ainsi une solution compacte et aisée à déplacer. Sur la face supérieure 4 angles viendront bloquer la feuille A4 de dessin.
La position du robot par rapport à la feuille A4 a été modifiée. En effet, lors de son travail, l'ancien étudiant avait indiqué que certaines parties de la feuille étaient inaccessibles mécaniquement. Le robot a été donc positionné sur le haut afin de pouvoir atteindre l'ensemble de la zone de dessin. (Ancien emplacement cf.figure 4.1 ; nouvel emplacement cf. figure 4.3)

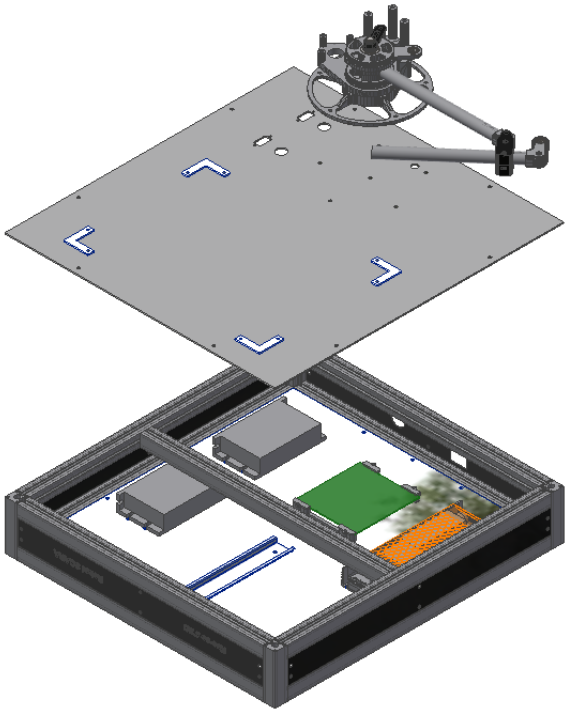


Figure *4.2 – Vue éclatée support robot*

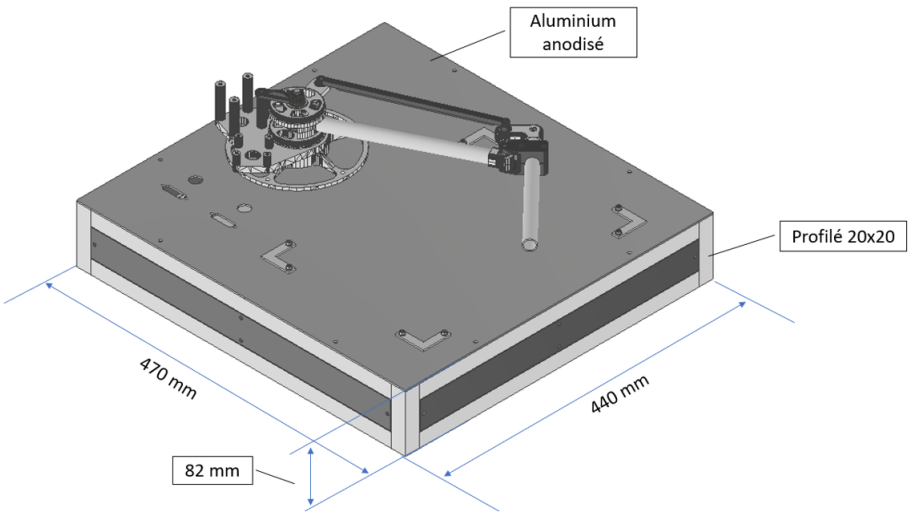


Figure *4.3 – Vue générale support robot*

4.2 Mise à zéro

Pour la mise à zéro du robot, les capteurs montés sur les moteurs ne peuvent pas être utilisés comme référence. En effet, ces derniers ne sont pas des codeurs absolus et nécessitent donc une prise de référence. De plus, ils ne seront pas branchés à l'automate qui pilotera les bras. Ainsi, un autre moyen de mise à zéro doit être trouvé.

Deux supports prévus pour des capteurs étaient imprimés sur la base du robot. Ainsi, le choix s'est porté sur des capteurs pouvant être montés sur cette dernière. Deux fin de course à galet (voir annexe E) ont été choisis pour la mise à zéro du robot. Ce type de détecteur présente les avantages de n'être pas cher et de réaliser une mise à zéro simple du système. Ainsi les deux bras devront être totalement rétractés pour prendre leur position de référence.

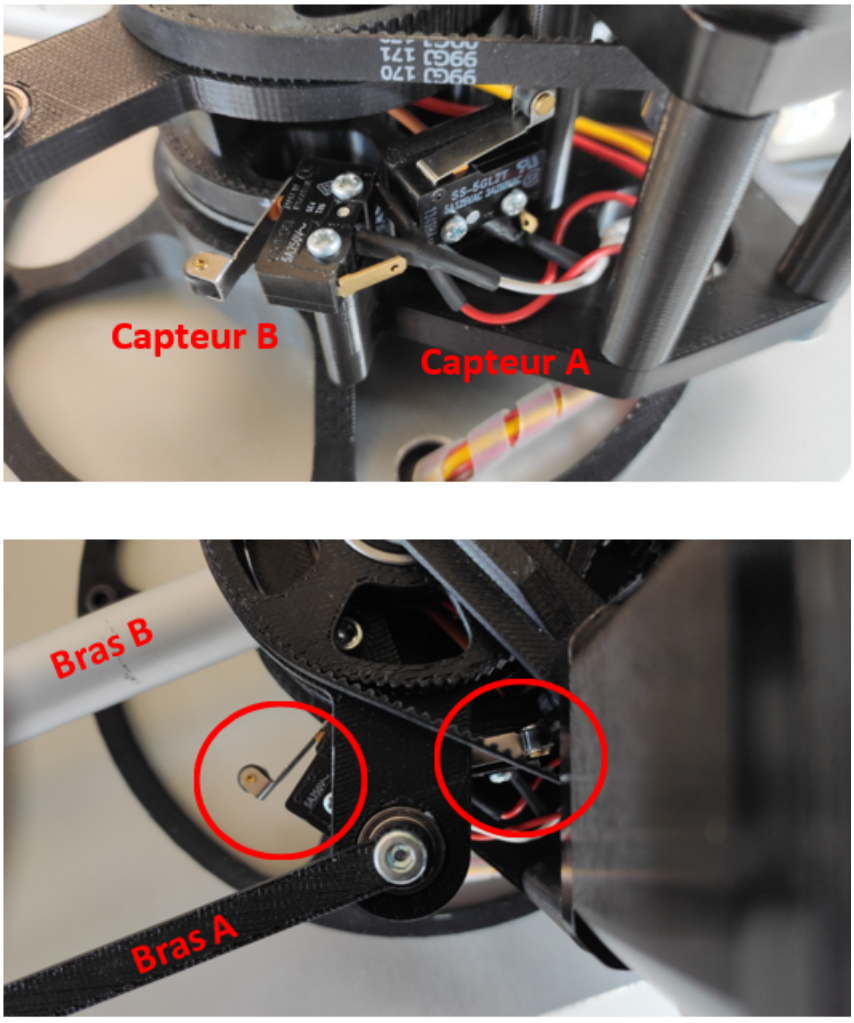


Figure *4.4 – Fins de course A et B*

Plus tard dans le projet, une modification des supports des capteurs a été nécessaire. En effet, leurs positions empêchaient d'atteindre l'extrémité bas droite de la feuille. La figure 4.4 est une image du montage final du projet.

4.3 Préhenseur

Le préhenseur fixé à l'extrémité du robot permet de maintenir le stylo tout en appuyant ou non sur la feuille. Des pièces déjà imprimées créaient la base du système.

Figure *4.5 – Pièces préhenseur déjà réalisées*

L'idée a été d'installer autour des axes (goupilles) deux ressorts de compression. Ainsi, le stylo serait toujours appuyé sur la feuille même en cas de changement de hauteur verticale. Le retour serait effectué par un servomoteur de modélisme. Ce dernier appuierait, avec un bras de levier, sur la partie plastique centrale du système créant un mouvement linéaire vers le haut.

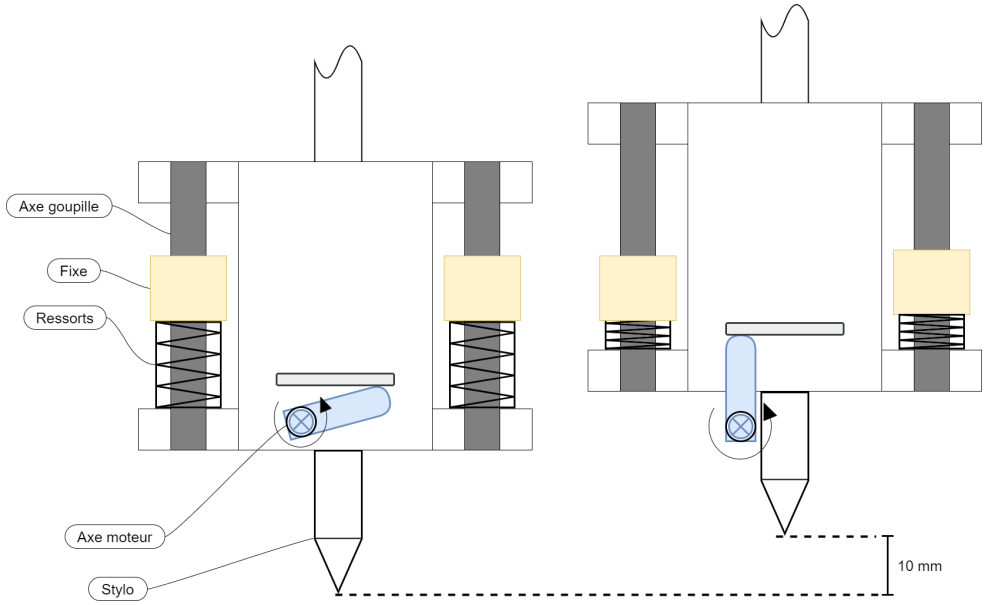


Figure *4.6 – Amélioration préhenseur schématisée*

La course verticale du stylo a été définie à 10 mm. Ce choix de déplacement vient de l'addition du jeu mesuré (1 – 2 mm) entre la position du bras déployé et rétracté mais également de la condition d'évitement des obstacles. En effet, au moment de la mise à zéro du robot le bras devra être complètement rétracté. Cette course devant être réalisée à l'aveugle il est possible qu'un stylo accroché au préhenseur entre en collision avec une vis fixant les angles de maintien de la feuille à dessin.

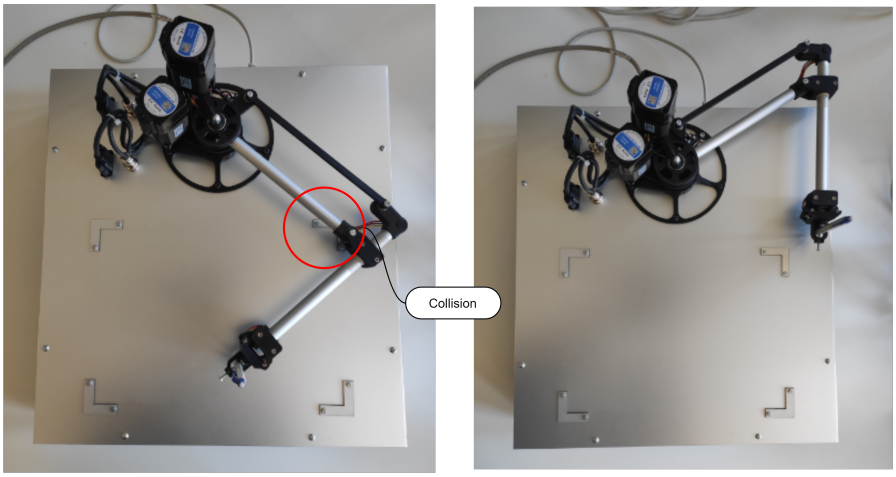


Figure *4.7 – Collision à la mise à zéro (gauche) et position de mise à zéro (droite)*

4.3.1 Ressorts et servomoteur

Pour le dimensionnement des ressorts, deux autres mesures ont été faites :

- poids d'écriture 80 à 200 g ;
- poids préhenseur 40 g.

Afin de correctement écrire sur la feuille la force à appliquer est de :

$$F_{feuille} = m_{ecriture} * g = (80 -> 200g) * 9.81 = 0.79 -> 1.96[N]$$

Ayant deux ressorts, la force minimale d'un ressort doit donc être de :

$$F_{ressort} = \frac{F_{feuille}}{2} - m_{prehenseur} * g = 0.0981 -> 0.6867\,[N]$$

En prenant arbitrairement le double pour la force du ressort comprimé, on obtient une constante de rigidité de :

$$c = \frac{\Delta F}{d} = \frac{0.6867}{10} = 0.06867\,[\frac{N}{mm}]$$

Les ressorts choisis chez Ressorts du Léman sont donnés en annexe C.2.

Grâce à cette rigidité, il possible d'en déduire le couple nécessaire au moteur pour effectuer le mouvement vers le haut. Il est à noter que le bras de levier L raccourcira en même temps que le système monte en fonction du cosinus de l'angle développé par le moteur (0° <=> préhenseur bas). L'équation du mouvement est donc :

$$M(\alpha) = (m_{prehenseur} * g + 2 * F_{ressort}) * L * \cos(\alpha)$$

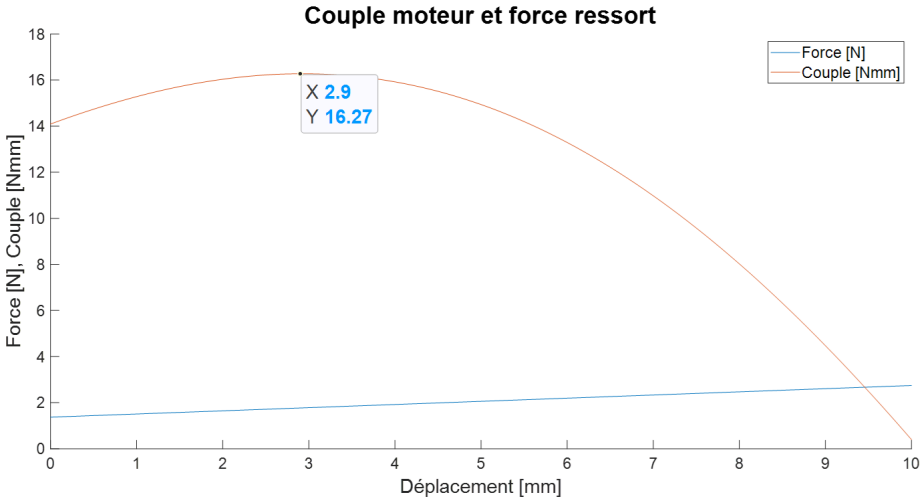


Figure *4.8 – Évolution du couple moteur en fonction du déplacement vertical*

Le moteur a donc été choisi en conséquence de ce résultat. Le couple nécessaire minimum doit obligatoirement être plus grand que 16.27 [Nmm]. Le choix s'est porté sur le modèle SER0049 de chez DFRobot. Ses caractéristiques principales sont :

Torque	44.13 à 53.9 [Nmm]
Current	0.11 à 0.12 [A]
Operating angle	180°
PWM	500 à 2000 μs

Les pulsations PWM seront également générées par l'automate S7-1200. Ces dernières seront cadencées à 50 [Hz]. Une largeur de 2 [ms] correspondra à 0° et 0.5 [ms] 180°. Dans le cadre de ce projet, une variation de 0 à 90° sera suffisant.

Figure *4.9 – Servomoteur choisi [8]*

4.3.2 Autres améliorations

Après montage du préhenseur, des améliorations de pièces ont été nécessaires. En effet, les douilles en laiton ne tenaient pas bien dans leur logement. La pièce supérieure a donc été à nouveau imprimée en ajoutant une partie empêchant la sortie des douilles.

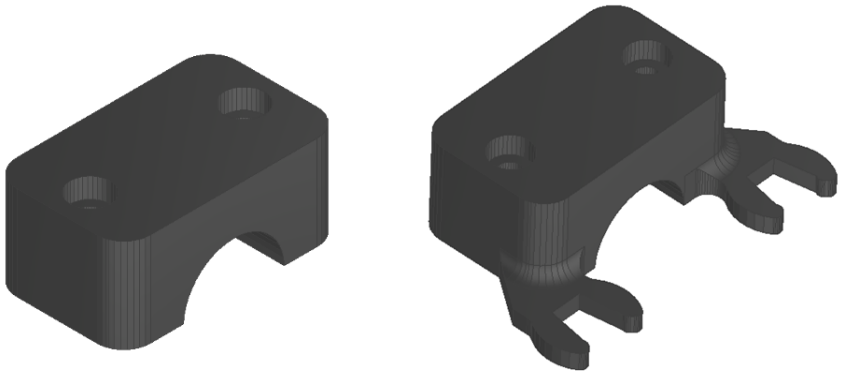


Figure *4.10 – Pièce supérieure avant (gauche) et après (droite) amélioration*

De plus, l'entraxe entre les goupilles n'était pas identique entre la pièce coulissante (slider) et le support du préhenseur. Ce mauvais alignement bloquait la pièce slider forçant sur le moteur et nécessitant une grande force de ressort pour descendre le stylo. Ainsi le slider a été modifié en améliorant par la même occasion le maintien des axes.

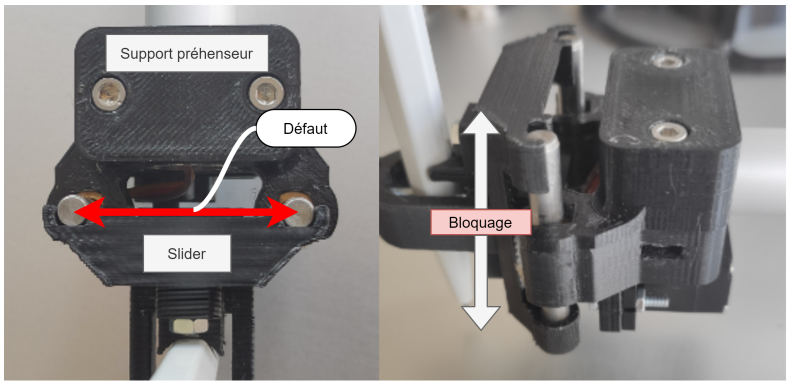


Figure *4.11 – Amélioration et défaut slider*

4.4 Jeux mécaniques

La dernière partie de ce domaine visait à améliorer la précision mécanique du bras robotisé. Les points principaux identifiés comme ayant un impact négatif sur le système sont les suivants :

— pignon moteur ;

— tendeur courroie ;

— axe central.

4.4.1 Pignon et tendeur

Le pignon et la courroie sont, sur ce robot, la liaison avec l'axe central supportant le bras robotisé et la motorisation du système. Sur ce montage, un premier problème a été identifié : des sauts de courroie crantée se généraient à l'inversion de sens des moteurs. L'origine du problème était multifactorielle. Premièrement, la courroie n'était pas assez tendue et ne pouvait pas l'être plus (mécanique sans possibilité de réglage). Ensuite, le pignon avait été réalisé en impression 3D plastique et était de mauvaise qualité.

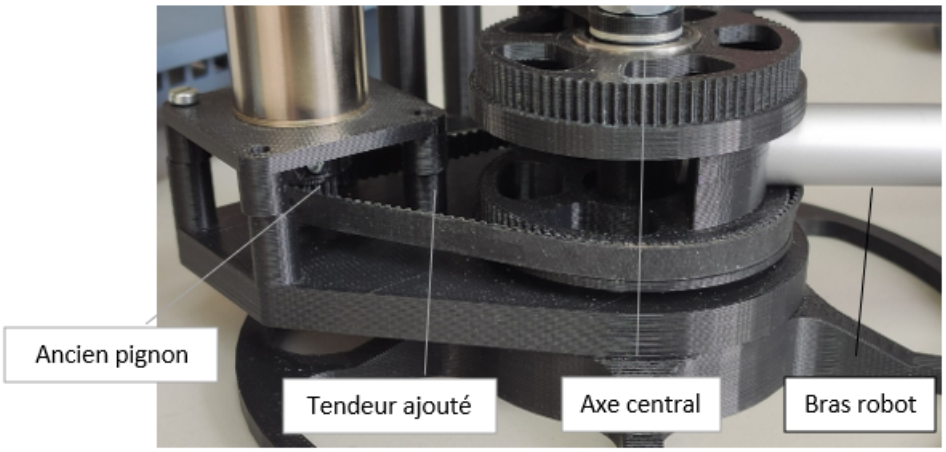


Figure *4.12 – Position des éléments améliorables*

Afin de pallier ce problème, plusieurs actions ont été réalisées. Premièrement, deux pignons en aluminium ont été achetés. Le diamètre qui a été choisi est légèrement supérieur à ceux précédemment imprimés, dans le but d'obtenir une meilleure accroche avec plus de dents reposant sur la courroie (de Ø10 à Ø12 mm).

Figure *4.13 – Pignon 2GT aluminium*

Ensuite, un tendeur a été réalisé en élargissant les trous de passage des vis fixant les moteurs à la base principale du robot. La course choisie pour ces tendeurs est de 2 mm. En addition avec l'augmentation de diamètre du pignon, cette distance est suffisante pour tendre correctement la courroie. Cette dimension a été choisie simplement en effectuant différents essais en déplaçant manuellement le moteur et en mesurant la course nécessaire pour obtenir une tension suffisante.

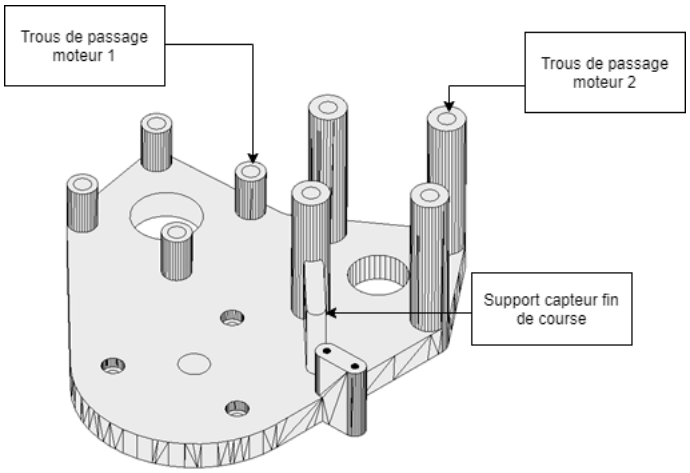


Figure *4.14 – Base bras robotisé*

Après différents essais du robot, un défaut supplémentaire a été identifié. Comme expliqué au chapitre du support robot (4.1), l'ensemble du montage avait été déplacé sur la partie supérieur de la feuille A4. En revanche, une partie de la feuille A4 restait inaccessible à cause d'un capteur fin de course placé trop en avant. La base du robot (figure 4.14) a donc dû être réimprimée avec un nouvel emplacement pour le fin de course du moteur 2.

4.4.2 Axe central

Cette partie mécanique est le support principal du bras robotisé (illustration 4.15). Elle consiste en deux roues dentées (turquoises) séparées par une entretoise (noire). Le tout est centré par un axe traversant deux roulements internes aux roues susmentionnées. En appuyant sur la feuille avec le stylo, le bras était poussé vers le haut provoquant ainsi une perte de précision. Ce défaut a été résolu en collant la cage extérieure des roulements et en créant un axe central plus ajusté pour la bague intérieure. La tolérance qui a été donnée pour le diamètre de l'axe où sont montés les roulements, a été choisie en g6 correspondant à un montage libre mais ajusté.

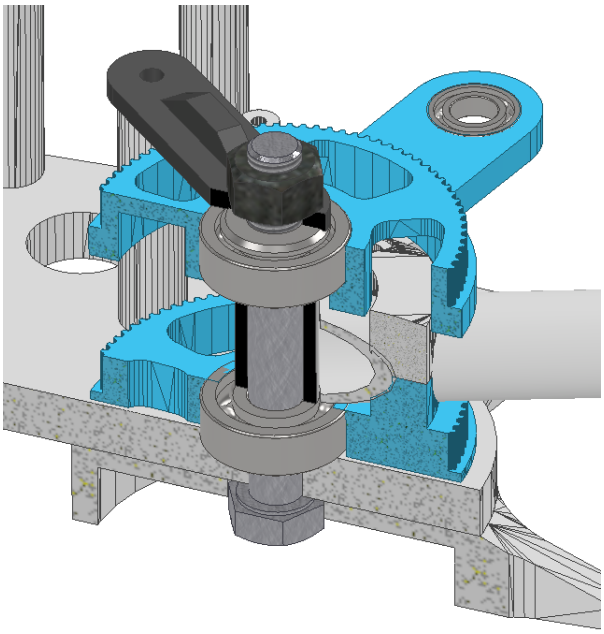


Figure *4.15 – Axe central montage*

Figure *4.16 – Axe central 3D*

4.5 Résultat

Le projet final, au niveau mécanique, est présenté sur l'illustration ci-dessous.

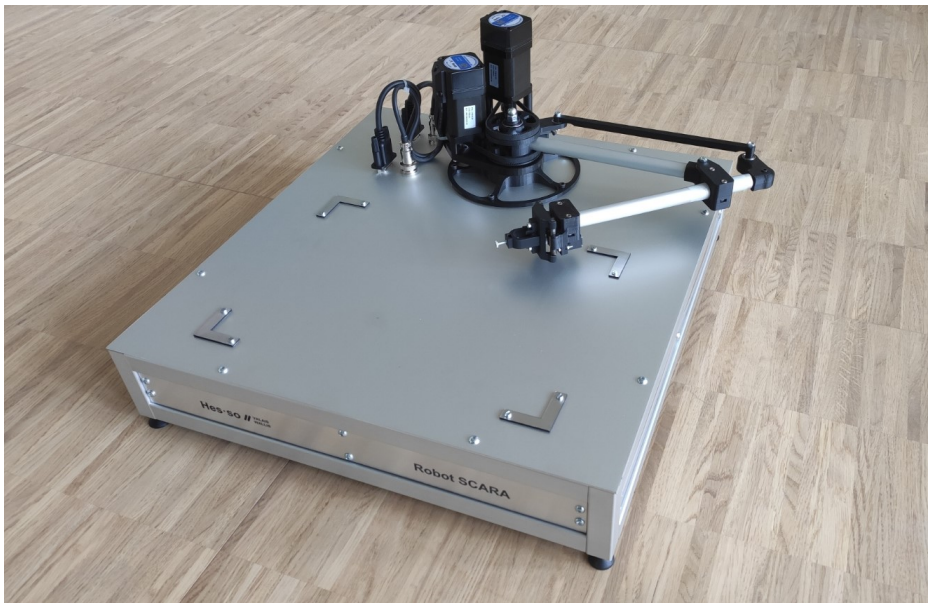


Figure *4.17 – Robot SCARA monté*

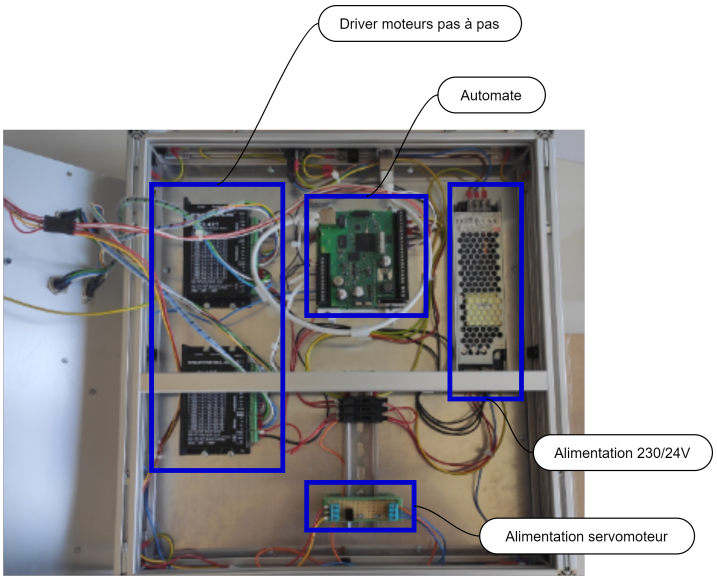


Figure *4.18 – Intérieur du coffret*

L'ensemble des mises en plan des pièces réalisées durant ce projet sont ajoutées en annexe C. Dans ce même chapitre, la totalité des pièces achetées sont listées sous C.2.

5 Programmation

La dernière partie de ce projet a été la programmation logicielle du robot. Elle consiste en trois points principaux :

- la génération de trajectoire ;
- l'interface utilisateur ;
- le pilotage des moteurs.

5.1 Concept général

Pour ce système, il a été décidé de travailler avec deux programmes en parallèle. Le premier, fonctionnant sur un PC, devra produire *l'interface utilisateur et la génération de trajectoires* et le second, sur un automate Siemens, *récupérera les consignes moteurs et pilotera le bras robotisé*.
Ce choix de fonctionner avec deux logiciels permet d'étendre la capacité de dessin du robot. En effet, avec un programme unique sur automate, le choix des dessins serait limité par la capacité de mémoire du système. Pour démontrer ce choix de manière concrète, un dessin évolué peut atteindre plus de 70'000 coordonnées alors que l'automate ne peut en stocker au maximum que 2'000. C'est aussi un des points qui limitait l'ancien étudiant (Sébastien Debons) dans son travail de diplôme. Ne pouvant générer qu'un nombre de points limité, son projet a été confiné au dessin de formes extrêmement simples comme des rectangles ou des losanges.
Le fait de fonctionner avec une génération de trajectoires externe à l'automate permet également de diviser les tâches et de ne pas surcharger l'automate en calculs. Malgré tout, il aurait tout de même été possible de travailler avec une mémoire externe branchée à ce dernier. Un second programme effectuant la conversion des coordonnées en angles pour les moteurs aurait quand même été nécessaire.

C'est pour toutes ces raisons que le logiciel a été divisé en deux parties.

Ce chapitre sera organisé en présentant en premier le travail réalisé sur l'automate, puis celui sur Java pour le PC de contrôle.

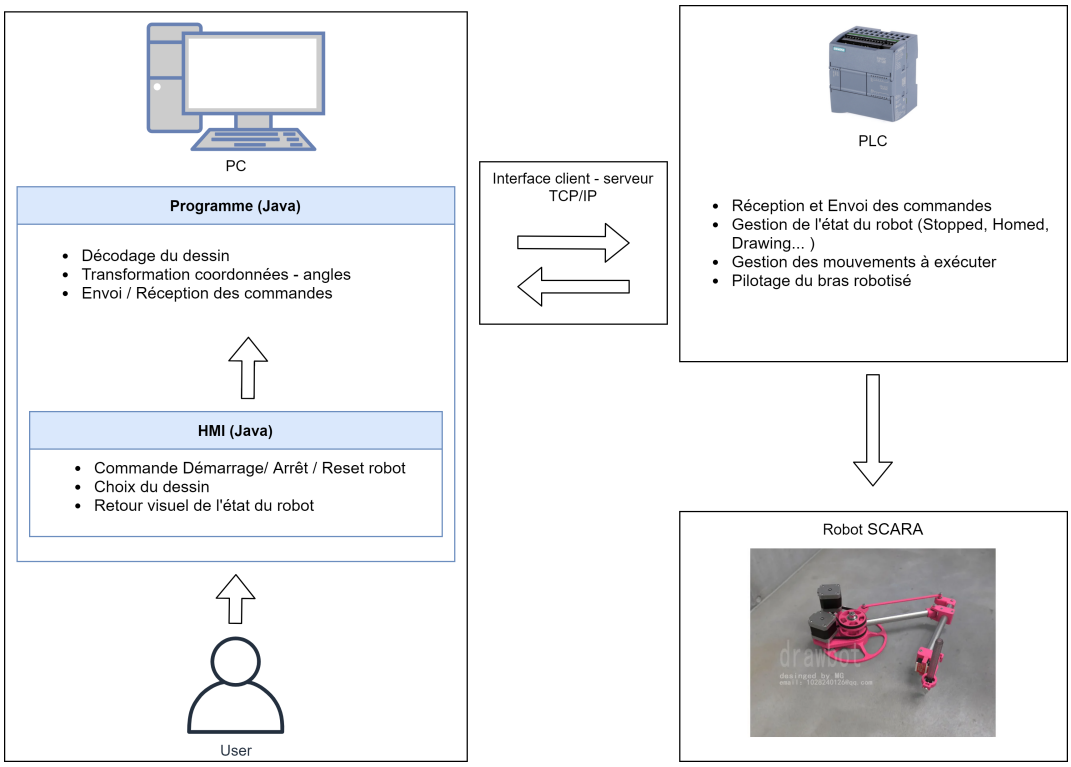


Figure *5.1 – Organisation logicielle*

5.2 Programme automate

La partie automate s'occupe essentiellement du pilotage des moteurs. Sa fonction principale est donc de récupérer les commandes envoyées par le PC et de l'informer quand l'ordre est terminé. Trois points sont donc développés :

— la machine d'état du système ;

— la communication TCP/IP ;

— le pilotage des moteurs.

Concrètement, le système sera séparé en deux blocs principaux. Le premier "SCARA control" implémentera la machine d'état et le fonctionnement du robot. Le second "Communication control" s'occupera de la communication TCP/IP avec le PC.

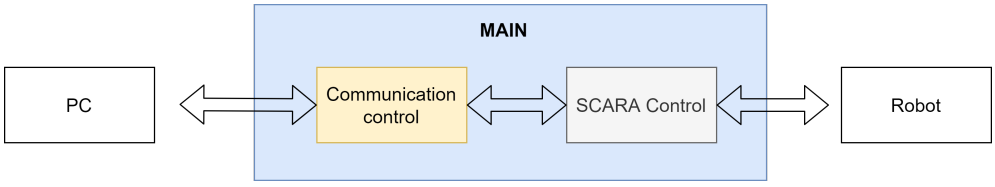


Figure *5.2 – Organisation générale du programme automate*

5.2.1 Machine d'état du système

Le logiciel fonctionne sur la base d'une machine d'état. Ce système structure le fonctionnement du robot. Dans le bloc SCARA Control deux sous systèmes interagiront. La machine d'état sera la logique et le "Response Controler" "dialoguera" avec le bloc de communication.

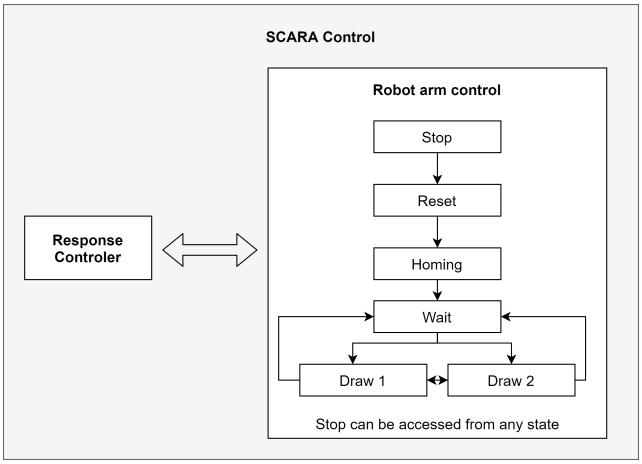


Figure *5.3 – Organisation du programme automate*

Chaque état (cf. figure 5.3) pourra être passé automatiquement ou après avoir reçu la commande correspondante.

État	Fonction	Passage au suivant
Stop	• Arrêt robot	CMD Reset
	• Puissance coupée	
Reset	• Ré-initialisation blocs moteurs	Reset validé
Homing	• Mise à zéro robot	Fins de course activés
Wait	• Attente démarrage • Écriture des nouvelles consignes	CMD Start draw
Draw 1-2	• Dessin • Écriture des nouvelles consignes	Liste de consignes terminée

Table *5.1 – Résumé des états du système*

Draw 1 et 2

Comme mentionné précédemment, un problème de capacité de mémoire a rapidement été découvert. En effet, l'automate ne pouvait stocker qu'un dessin partiel. Pour résoudre le problème, le dessin sera transmis par "Ligne" $\approx$ groupe de coordonnées ou segment de dessin. Au moment de la réception d'une "Ligne" complète, l'ensemble des mouvements sera effectué. De plus, le traitement (écriture et confirmation) des points étant lente, un système de buffer a également été mis en place. Au moment de l'exécution de la première ligne une seconde sera enregistrée en parallèle, réduisant ainsi le temps d'attente entre chaque dessin de segment.

Chaque buffer aura une capacité de 1000 coordonnées au maximum, correspondant à la moitié du stockage maximal du système (chapitre 5.1). Une possibilité aurait été d'utiliser la pleine capacité et de ne pas créer de tampon. En revanche, l'écriture de 2000 coordonnées aurait pris un temps considérable. De plus, la plupart des segments de dessins ne dépassent pas les 1000 coordonnées ce qui aurait rendu inutile une partie de la mémoire de travail.
Il est à noter que la structure de dessin mentionnée ici est développée plus en détail dans le chapitre 5.5.1.

5.2.2 Réception des commandes

Présenté au chapitre 5.2, le bloc "Communication control" s'occupera de la communication TCP/IP. Chez Siemens la liaison est déjà prête et a simplement besoin d'être configurée. Les paramètres principaux sont les suivants :

— IP PLC (statique) 192.168.0.2

Une adresse fixe permet d'éviter plus de travail à la découverte de l'adresse IP du PLC

— Local PORT PLC : 2000

Le PLC aura le port 2000 ouvert à la communication

— Partner : non spécifié

Non spécifié est nécessaire dans le cadre d'une communication avec un autre système qu'un automate Siemens.

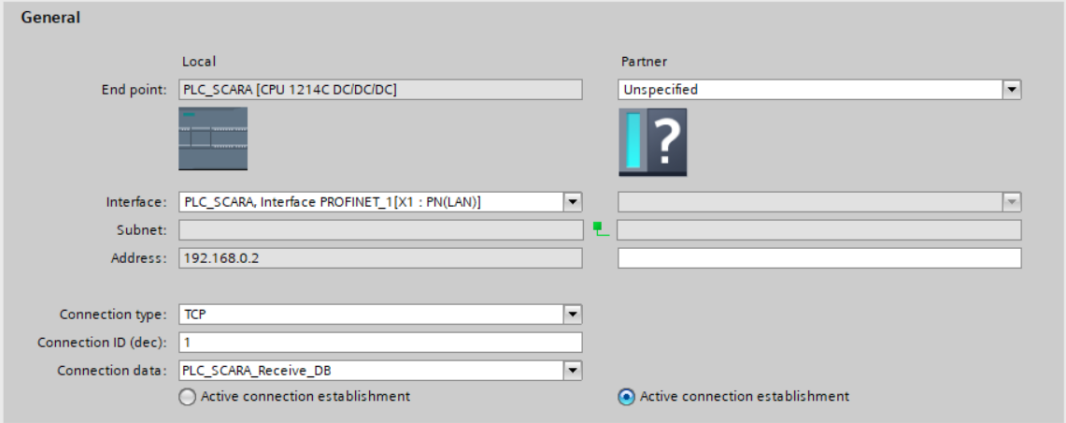


Figure *5.4 – Configuration bloc de communication*

Plus concrètement, ce système effectuera la liaison avec le PC. Dès qu'une requête de connexion sera effectuée, la communication sera automatiquement établie et maintenue jusqu'à déconnexion du client (PC). Les informations nécessaires à transmettre tout au long des transactions sont :

— des ordres généraux :

le type d'ordre à effectuer (arrêt, démarrage,etc...) ;

— des ordres de mouvement :

le type de mouvement à réaliser ;

la position de consigne des deux moteurs ;

la vélocité des deux moteurs ;

la prochaine étape.

Le PLC ne pouvant recevoir qu'un type de message dans un format très précis, ces commandes ont été concaténées afin de toujours envoyer un message de même longueur. Le type de donnée Char a été choisi pour transmettre ces commandes car il permettait d'effectuer simplement une mise en forme d'une longueur spécifiée et avait également la possibilité d'être facilement convertible.

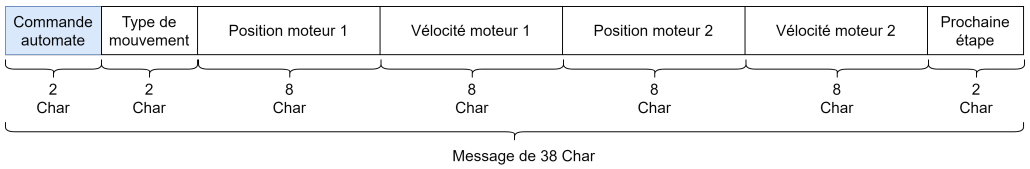


Figure *5.5 – Format de message*

Après avoir été converti de Char à entier ou nombre à virgule, cette commande est traitée par le bloc "SCARA Control". Il est à noter que dans le cas d'une simple commande d'arrêt du robot ou reset, il est indispensable que le message soit également long de 38 caractères. Dans ce cas, la suite du message sera remplie par des 0.

Le choix de 8 caractères pour transférer la vitesse et la position vient de deux critères. Les valeurs seront au maximum dans les centaines et la précision est de 3 digits car l'angle minimal pouvant être réalisé par le moteur est de $360°/5000 = 0.072°$. La valeur de 5000 est le facteur de Microstepping calculé précédemment.

5.2.3 Pilotage des moteurs

Cette partie est la pièce la plus importante à la réalisation de mouvements fluides et précis. La première étape est la configuration des moteurs et la seconde le pilotage.

Configuration

Les objets technologiques de chez Siemens ont été utilisés pour paramétrer les deux moteurs pas à pas. Ces objets sont simplement des interfaces utilisateur permettant d'avoir un paramétrage plus aisé et visuel de ces moteurs.

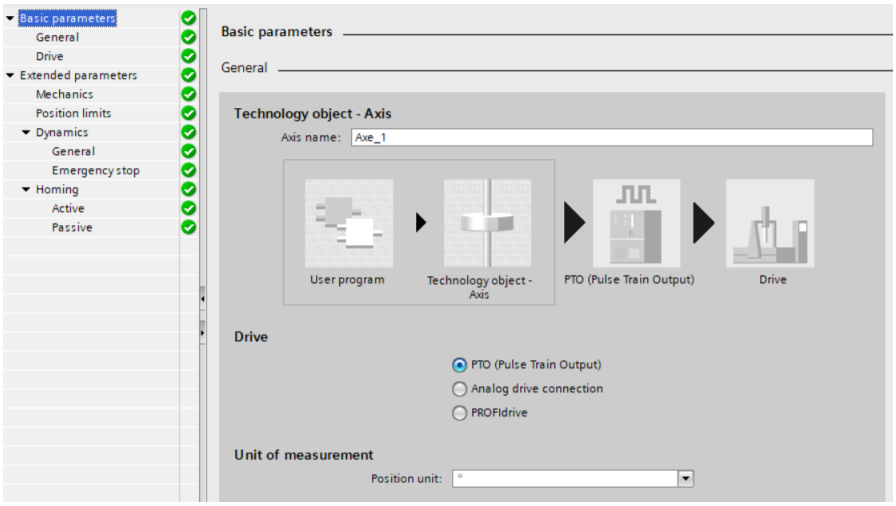


Figure *5.6 – Interface de l'objet technologique axe*

Tout d'abord, dans l'onglet Basic parameters / General (cf. figure 5.6) permet de configurer le type de sortie et l'unité des consignes données. Pour ce projet, la commande PTO sera utilisée et par simplicité, les angles seront donnés en degrés.

Ensuite, sous Drive, l'interface permet de choisir quelle sortie physique sera utilisée pour les impulsions données aux drivers. Dans ce cas, une sortie sera configurée pour les pulsations et une seconde pour le sens. Sur la figure 5.7 la configuration est donnée pour le moteur 1. Un paramétrage équivalent sera exécuté pour le moteur 2 avec, évidemment, des sorties physiques différentes.

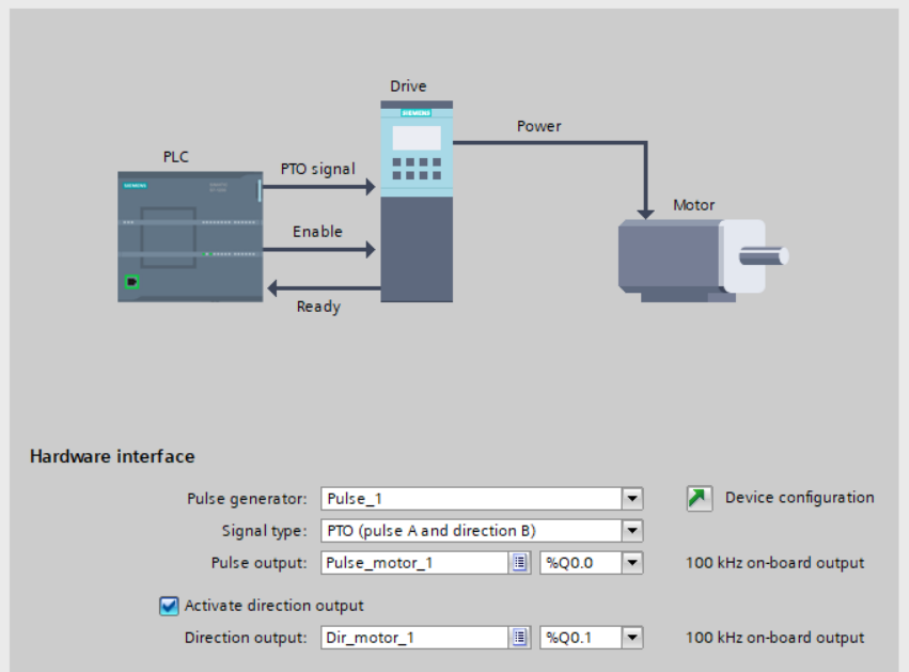


Figure *5.7 – Onglet Drive*

Puis, l'onglet Mechanics permet de configurer le facteur de microstepping choisi. Dans le cas du projet, une valeur de 5000 a été définie comme suffisante. Ce nombre est le coefficient calculé au chapitre 3.2.

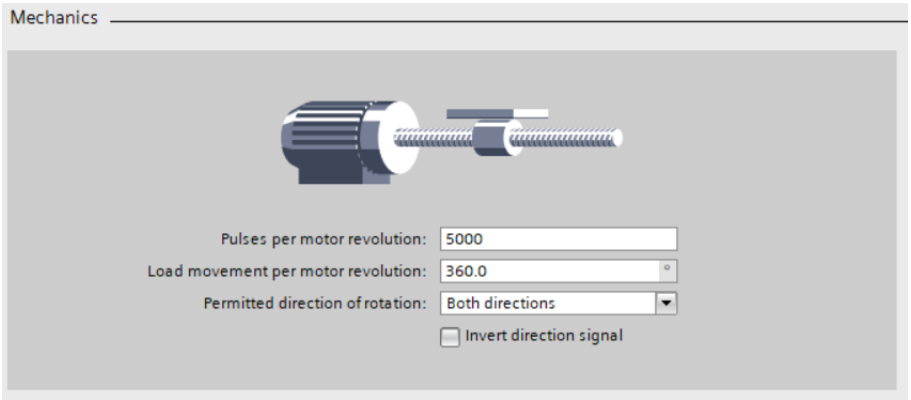


Figure *5.8 – Onglet Mechanics*

Après cela vient la configuration des limites mécaniques du système. Ici, uniquement les fins de course logiciels (Software limit switches) ont été utilisés. Le robot sera immédiatement arrêté et mis hors puissance en cas de dépassement de ces limites.
Au moment de la mise à zéro du robot (rencontre avec le fin de course mécanique), la position atteinte par le moteur sera définie comme 0°. La position supérieure a donc été choisie très proche de 0° pour empêcher la collision avec les fins de course mécaniques. Pour la limite basse (low limit switch), la valeur a été choisie en pilotant manuellement le moteur jusqu'à atteinte de la position maximale nécessaire au dessin sur une feuille A4.

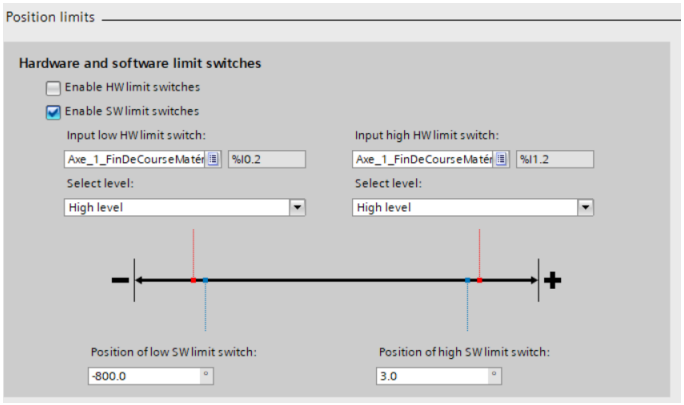


Figure *5.9 – Onglet Mechanics*

Finalement, la configuration de la vitesse et de l'accélération maximale a été réalisée. Les valeurs finales ont été trouvées par essais en combinaison avec la vitesse fixée au chapitre 5.5.1. La vitesse minimale a été imposée par le programme et les accélérations devront être maximales à tout moment afin de rendre le mouvement le plus fluide possible.

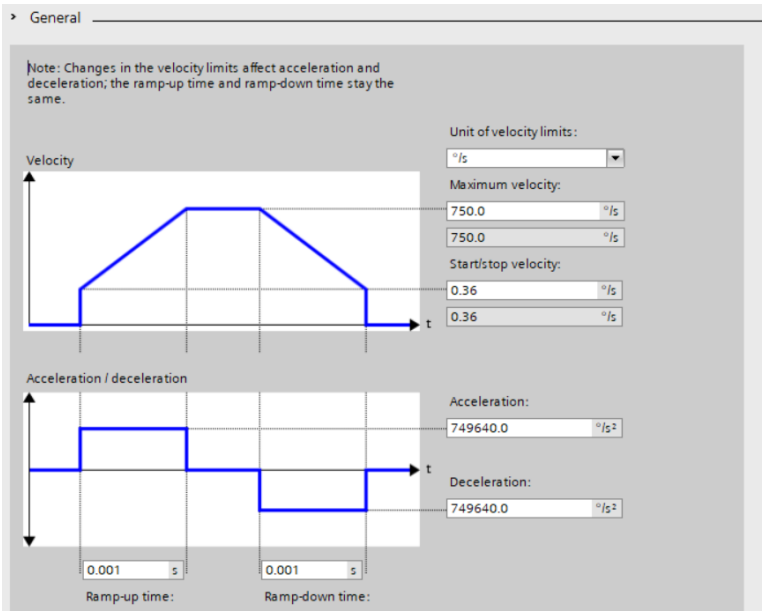


Figure *5.10 – Onglet Position limits*

Commande des moteurs

Pour utiliser les moteurs avec des objets technologiques, 3 blocs doivent être combinés. Le premier est le MC_Power permettant de couper la puissance sur le moteur, le second MC_Reset pour réinitialiser le système et MC_CommandTable pour démarrer les mouvements. L'ordre de démarrage est donc :

1. ré-initialisation du système (MC_Reset) ;
2. allumage de la puissance (MC_Power) ;
3. démarrage du mouvement (MC_CommandTable).

Initialement il avait été prévu de fonctionner avec des "commande tables" c'est à dire un set de mouvements pouvant être réalisé sans arrêter les moteurs. Ces derniers permettaient également de calculer automatiquement les accélérations à donner aux moteurs. L'interface (cf. figure 5.11) se présente sous la forme d'une liste de 32 mouvements et leurs paramètres. Dans notre cas, ce qui aurait été utilisé est le positionning absolute (positionnement absolu d'un moteur) avec comme prochaine étape le blending motion (pas de passage à 0 de la vitesse). Sur le bas, un graphe permet d'observer concrètement le mouvement qui sera effectué.

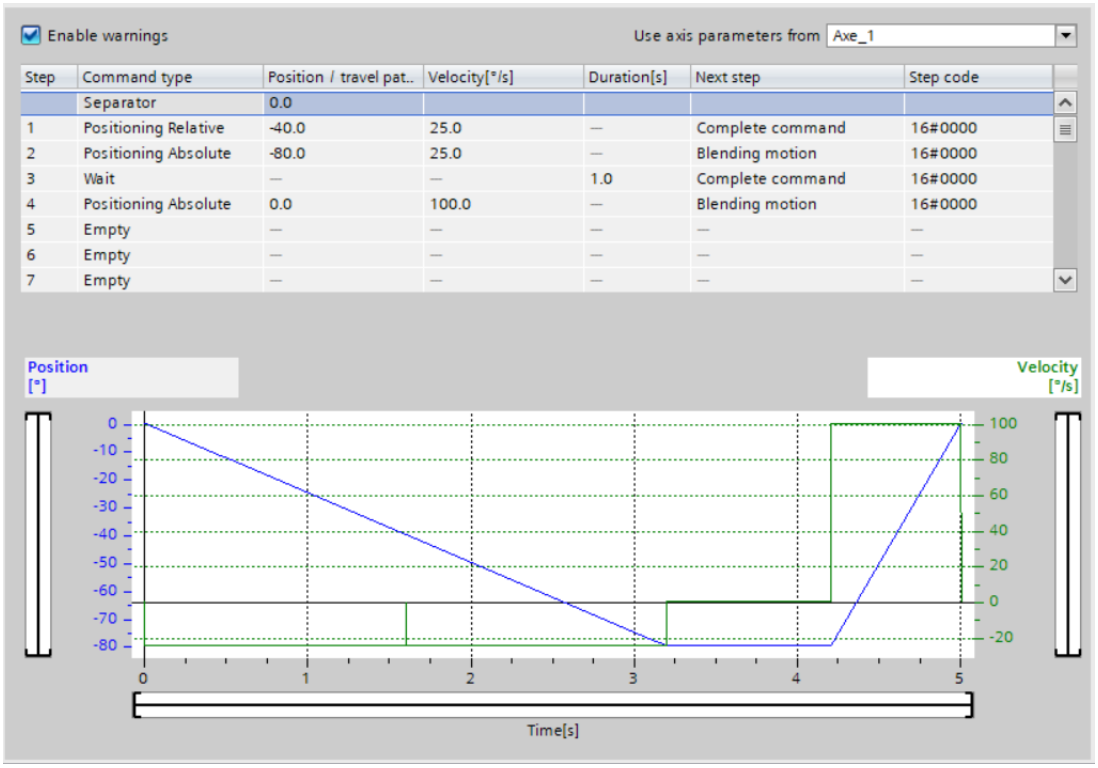


Figure *5.11 – Interface de l'objet CommandTable*

A l'utilisation de ces blocs, plusieurs problèmes sont apparus. Premièrement, la liste des mouvements est non extensible, limitant à uniquement 32 mouvements sans pauses. Ensuite, un important défaut de synchronisation a été remarqué entre les axes. En effet, malgré un calcul avec un temps égal entre chaque point pour les deux moteurs et un démarrage synchronisé, le système se décalait systématiquement. Un second essai en utilisant la commande permettant de faire fonctionner un moteur pendant un temps

précis a été effectué. Le défaut était à nouveau présent. Ce dernier a été identifié comme un problème de ressource de l'automate. Le PLC n'étant pas très puissant, les démarrages et enchaînements puis arrêts des deux "command tables" (moteurs 1 et 2) n'étaient pas toujours effectués au même cycle, posant ainsi un problème direct sur la synchronisation des moteurs donc sur le dessin réalisé.
Ce problème important a été résolu en ralentissant le système par l'écriture d'une commande unique à l'objet CommandTable (cf. figure 5.12). Cette nouvelle manière de piloter les moteurs a permis de réduire très fortement le problème de synchronisation des axes, mais a comme défaut d'impacter négativement la vitesse d'exécution du dessin. En revanche, en cherchant différents moyens de piloter des moteurs pas à pas, il est apparu que cette méthode est très régulièrement utilisée et convient parfaitement pour ce type de moteur. [9]

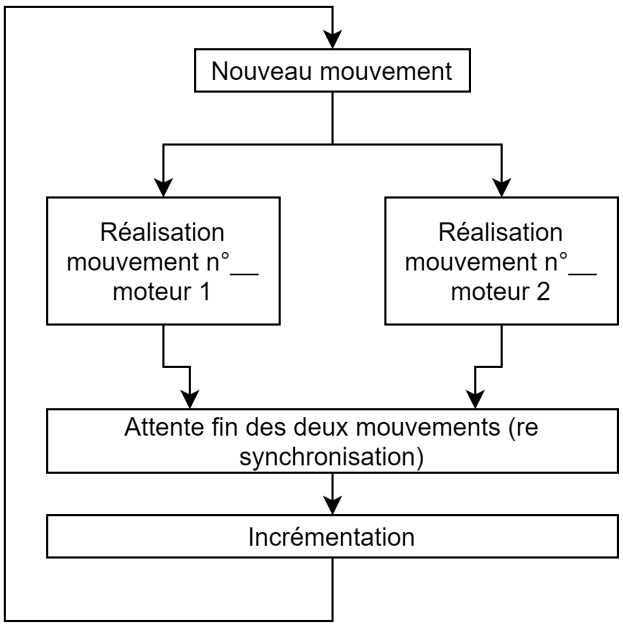


Figure *5.12 – Nouveau pilotage*

Une manière d'accélérer le mouvement serait de se munir d'un micro contrôleur plus puissant. En effet, sur beaucoup d'appareils différents (imprimantes laser, 3D, CNC), ce type de commande a été retrouvé. Le temps de cycle est simplement plus rapide, permettant ainsi de piloter de manière plus efficace de multiples axes.

5.3 Programme PC de contrôle

Le logiciel réalisé sur le PC de contrôle a plusieurs objectifs. Il doit pouvoir décoder un dessin, se connecter à l'automate, envoyer des consignes et produire une interface utilisateur. Plusieurs choix étaient envisageables : Matlab ou Java. Le premier permettait d'implémenter extrêmement simplement une interface utilisateur et une communication TCP. Son principal défaut était que le système HMI ne fonctionnait que sur un système d'évènements. En d'autres termes, des scripts peuvent être lancés uniquement par un appui sur un bouton. Le problème est que l'automate peut envoyer parfois des données de manière asynchrone. Ainsi, le programme doit réagir en conséquence sans appui sur un nouveau bouton. De plus, le système de construction d'interface Matlab étant encore très récent, de très nombreux bugs sont apparus à l'utilisation. Finalement, la seconde possibilité, Java, nécessitait plus de travail mais avait un environnement de travail beaucoup moins restrictif. Pour toutes ces raisons, le langage Java a été choisi.

Ensuite, pour le choix du dessin, l'ancien étudiant avait rentré manuellement dans l'automate un set de points permettant de réaliser une forme. L'objectif serait de s'affranchir de ce système afin d'augmenter le choix de dessins disponibles, voire de réaliser des dessins personnels. Pour interpréter une image, deux méthodes principales existent : les coordonnées cartésiennes ou le GCode. Ce dernier système, choisi pour ce projet, sera expliqué plus en détail dans le chapitre suivant (5.3.1).

Au final, le flux du choix de l'image au PLC sera le suivant :

1. choix de l'image ;
2. transformation de l'image en GCode ;
3. lecture du fichier et génération de trajectoire ;
4. envoi des consignes ligne par ligne.

Concrètement, en utilisation courante, la personne choisira un dessin déjà converti en GCode. Le programme LaserGRBL permettra simplement de rajouter des illustrations supplémentaires.

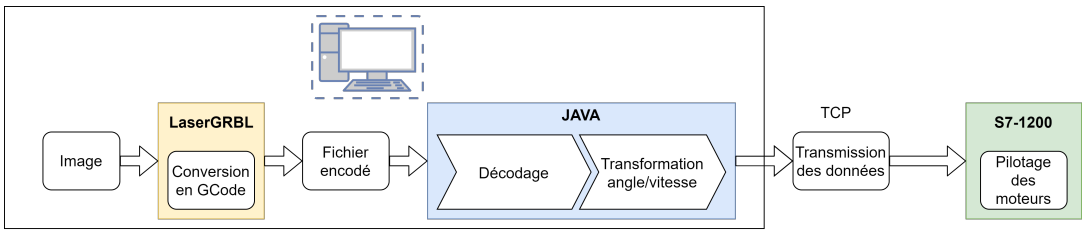


Figure *5.13 – Flux de l'image au dessin*

Cette section sera la description du GCode et de LaserGRBL. Les suivantes décriront le programme Java réalisé.

5.3.1 GCode

Le GCode est un code sous forme de liste d'instruction permettant de piloter des machines-outils à commande numérique [9]. Ce format de fichier est utilisé dans de nombreuses installations comme par exemple, les CNC, les découpeuses laser/plasma/jet d'eau ou encore l'impression 3D. Généralement, dans ces applications, le fichier sera généré par un système externe qui enverra par la suite ces listes d'instructions à une carte de pilotage. Une ligne de ce fichier encodé contiendra toutes les informations nécessaires pour réaliser une trajectoire. En effet, ce code permet de décomposer les instructions d'un dessin, CAO par exemple, en type de mouvements. Pour le cas de ce projet le GCode est donc tout indiqué. Avec une bonne interprétation du fichier, la génération de trajectoires est déjà partiellement réalisée. En opposition, le système de coordonnées qui aurait pu être utilisé pour ce travail ne contient pas toutes les informations requises pour le robot. Une seconde transformation du fichier aurait été nécessaire pour choisir, par exemple, le bon type de mouvement.

Les principaux codes

Une liste d'instructions de GCode se génère, dans la majorité des cas, de la façon du tableau 5.2. L'instruction de gauche donnera l'information sur le type de commande et la suite sur la position.

Code	Signification
M3 **S**0	Démarrage de la broche
F500	Vitesse de déplacement 500 mm/min
G*0* **X**126 **Y**30	Positionnement absolu sans contrôle de trajectoire à X126 et Y30 (plan cartésien)
S1000	Vitesse de broche = 1000
G*2* **X**126.426 **Y**30.71 **I**0 **J**16.735	Mouvement circulaire dans le sens horaire
G*3* **X**126.217 **Y**30.726 **I**0.078 **J**2.305	Mouvement circulaire dans le sens anti-horaire
G*1* **X**2.046 **Y**30.78	Mouvement linéaire
S0	Vitesse de la broche = 0
M5	Arrêt de la broche

Table *5.2 – Exemple de commandes générées*

Dans le cas de ce travail, les commandes seront interprétées différemment. La M3 indiquera le départ d'un dessin et la M5 la fin. Les codes G (0 à 3) devront également être transformés à l'aide du modèle géométrique inverse du robot. (Voir chapitre 5.5.2). Le S0 qui normalement indique un changement de mouvement, sera interprété comme une nouvelle liste d'instruction ≈ segment de dessin (voir figure 5.14 [1]).
Le robot SCARA fonctionnera par exécution d'un set de mouvements démarrant par cette dernière commande. Ne pouvant pas envoyer ces données sans interprétation, le programme développé sur Java s'occupera d'effectuer la lecture du fichier, la transformation et le calcul des vitesses.
En réalité, de nombreuses autres commandes existent. Elles seront simplement ignorées car inutiles dans le cas d'un dessin en deux dimensions.

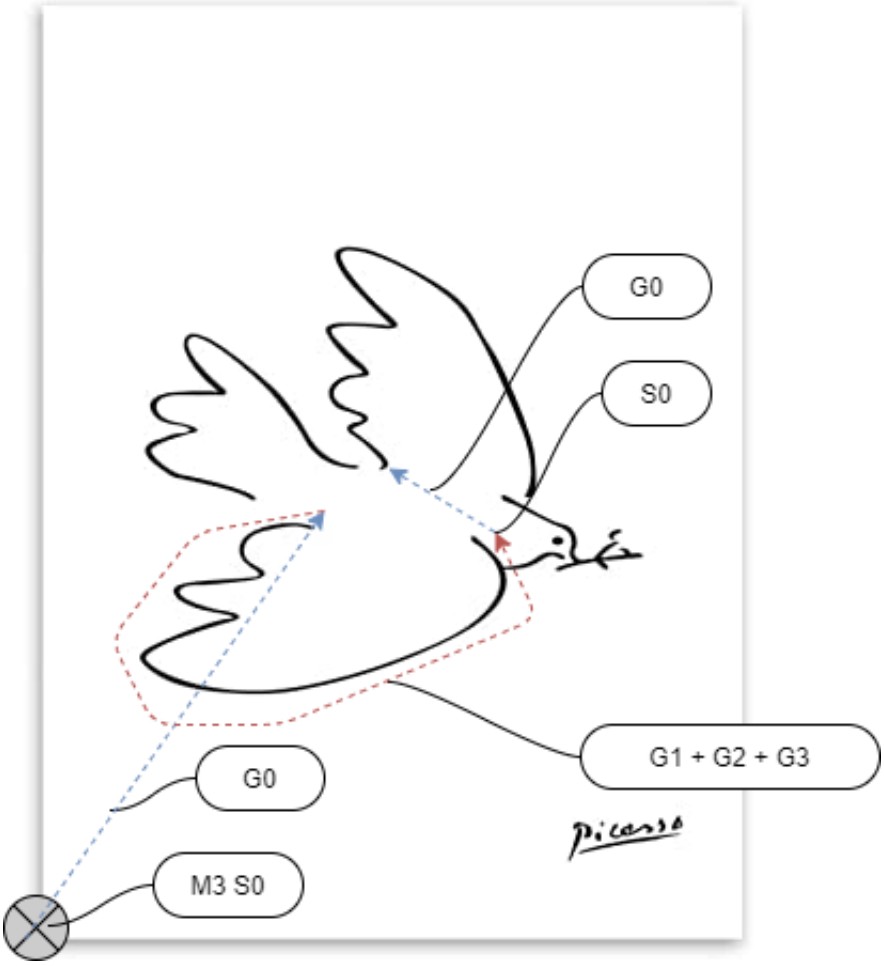


Figure *5.14 – Schématisation G-code*

5.3.2 Logiciel LaserGRBL

[Logiciel A]

LaserGRBL est le programme opensource qui a été choisi pour transformer des images en GCode par vectorisation [10]. Ce système a également la possibilité de piloter des graveuses laser, mais dans ce projet cette fonctionnalité ne sera pas exploitée. L'interface se présente sous la forme de l'illustration 5.15. Les traits bleus continus représentent un dessin et les traitillés seront le déplacement du robot sans dessin. L'intégration d'une image se fait par un "drag and drop". Pour obtenir le GCode de ce fichier, il suffit simplement d'enregistrer le projet depuis l'onglet Fichier $\longrightarrow$ Sauver le programme. A l'importation d'un nouveau fichier, une nouvelle interface (figure 5.16) s'ouvre et permet de configurer la vectorisation du dessin. La luminosité et le contraste permettront de régler la qualité du dessin et l'option de lissage de vectoriser avec des interpolations circulaires. Une dernière option permet de choisir le type de remplissage.

1. Image for G-code
https://www.amazon.fr/Picasso-Abstrait-Peinture-Tableaux-Decoration/dp/B08MQ741CM (accessed: 25.07.2021)

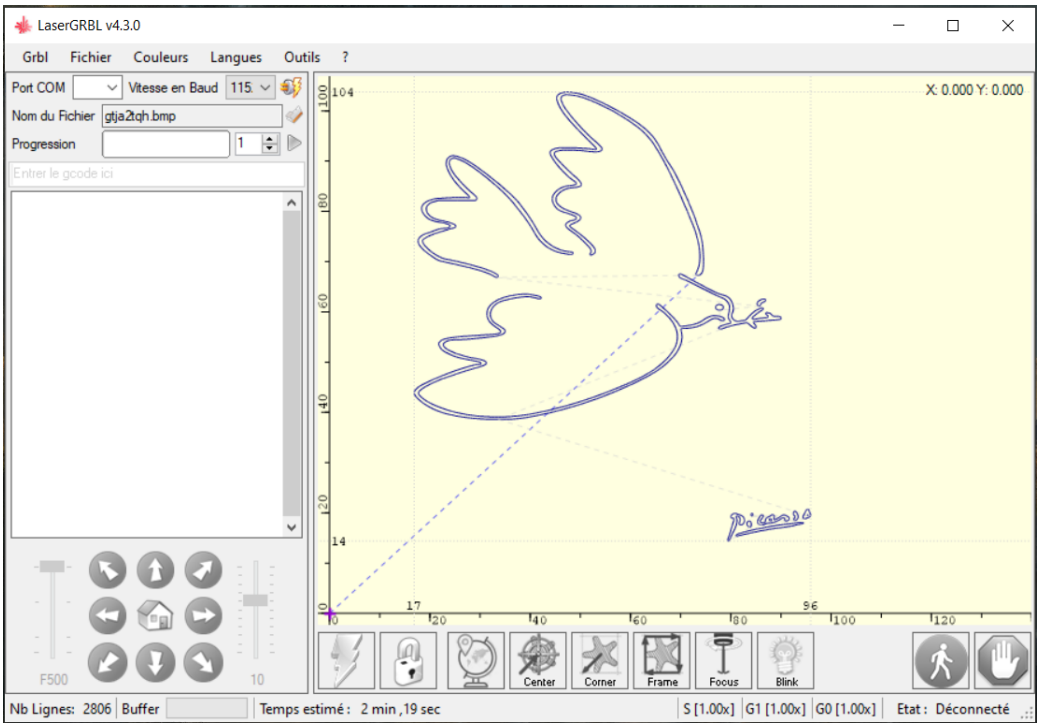


Figure *5.15 – Interface utilisateur LaserGRBL*

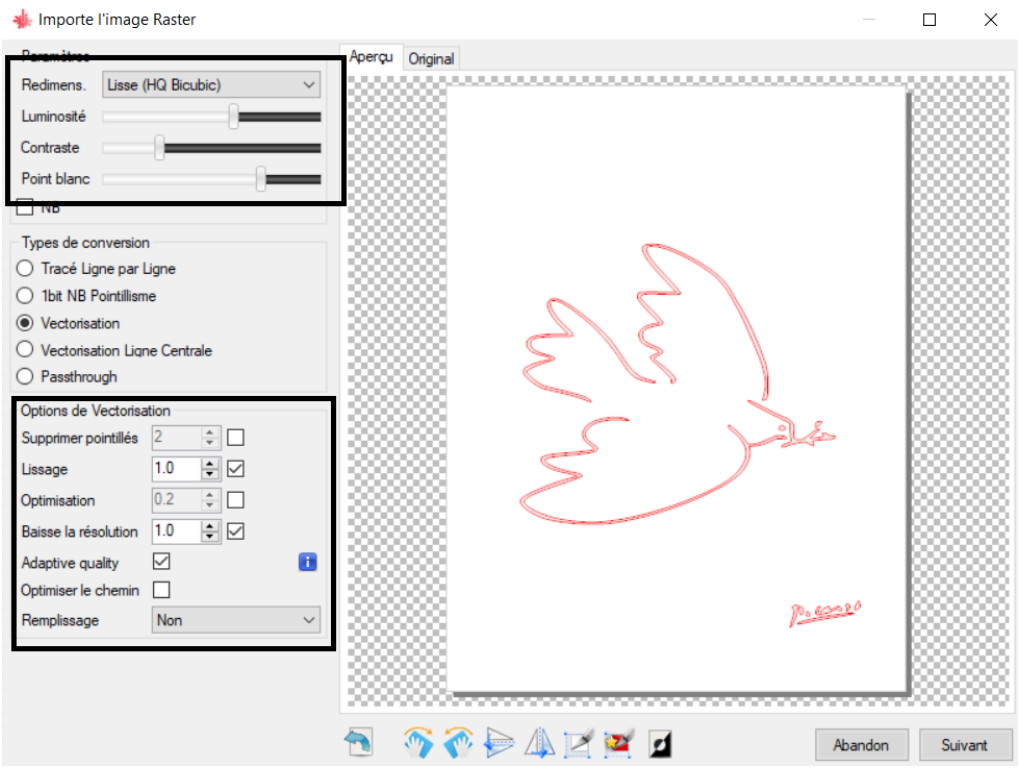


Figure *5.16 – Interface d'importation*

5.4 Programme Java

Le logiciel développé sur Java doit donc permettre de décoder les fichiers GCode et de les transmettre à l'automate. Le système est construit sur deux éléments principaux : le programme cyclique (logique du système) et l'interface utilisateur. La Classe principale où les deux sont appelés est UIClient. Pour l'ensemble de cette section, les cadres noirs représenteront des Objet Java et les cadres traitillés l'organisation du système.

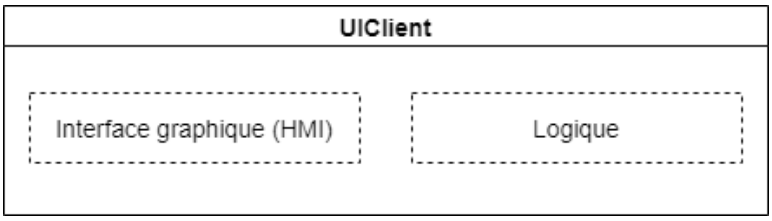


Figure *5.17 – Organisation générale du logiciel*

5.5 Logique

Dans la partie logique, le code est séparé en deux fonctions principales, la partie communication TCP et la partie lecture/décodage d'un fichier d'instruction. L'ensemble des classes créées pour ce projet sont répertoriées dans la figure 5.18. Chaque classe appellera une sous-classe permettant ainsi de structurer l'organisation du système.

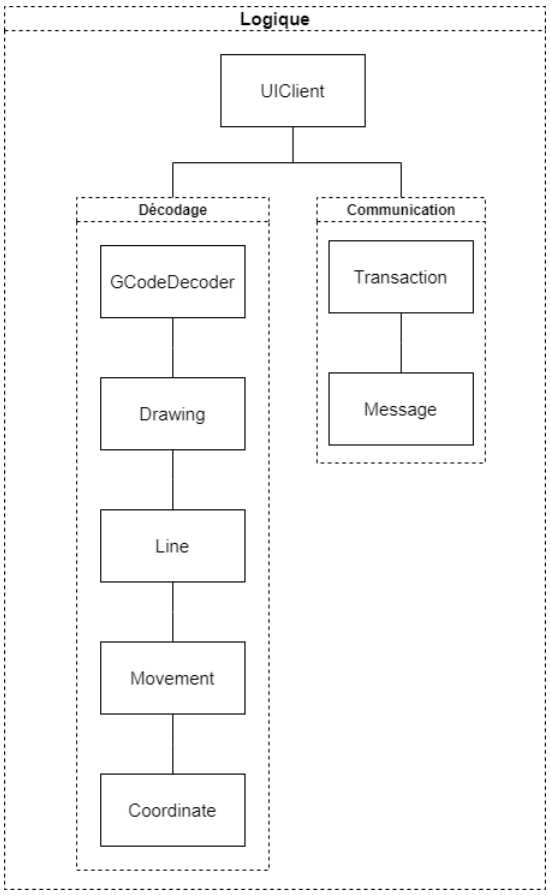


Figure *5.18 – Organisation générale partie logique*

Cette partie s'occupera donc d'interpréter le GCode afin de le transformer en coordonnées polaires pour le robot. Ainsi deux processus doivent être effectués :

1. interprétation du GCode ;
2. transformation par modèle géométrique inverse.

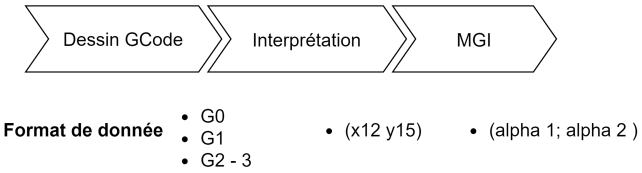


Figure *5.19 – Flux du GCode aux coordonnées*

5.5.1 Décodage et trajectoire

La partie lecture de fichier sera organisée par la classe GCodeDecoder. Celle-ci, à son appel créera un nouveau dessin (Drawing). Ce dernier contiendra des groupes de lignes c'est-à-dire à chaque fois que le stylo changera de segment de dessin (chapitre 5.3.1). Ces lignes (Line) contiendront elles-même des groupes de mouvements. Ces derniers stockeront uniquement deux coordonnées en angle (moteur 1 et moteur 2) ainsi que leurs vélocités. Ils représenteront donc un mouvement d'un point au suivant. L'objet Coordinate stocke simplement les coordonnées cartésiennes du mouvement.

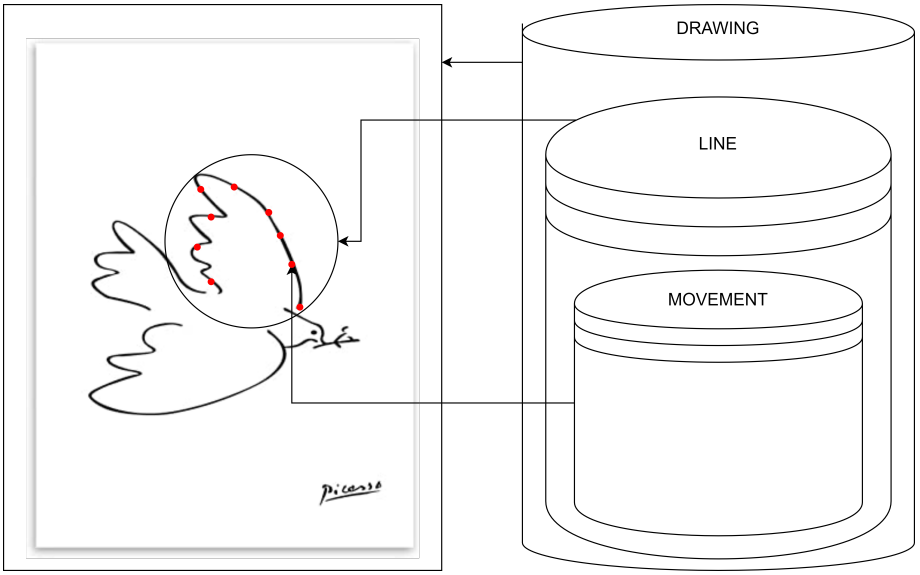


Figure *5.20 – Organisation logicielle d'un dessin*

Le décodeur de GCode appliquera automatiquement cette structure à chaque nouveau dessin donné. Ce dernier lira instruction par instruction le fichier et appliquera la transformation requise. Il restera donc au GCodeDecoder à interpréter correctement les différentes commandes existantes. Quatre instructions principales ont été étudiées :

— G0 positionnement direct ;

— G1 interpolation linéaire ;

— G2-G3 interpolation circulaire.

Positionnement direct

Exemple de commande : G0 X1 Y2

Cette première est la plus basique et ne nécessite pas d'interpolation. La coordonnée cartésienne reçue sera simplement transformée à l'aide du modèle géométrique inverse. Les vitesses n'étant pas importantes elles seront fixées à 75°/s permettant ainsi de réduire fortement le temps de déplacement entre chaque segment du dessin.

Interpolation linéaire

Exemple de commande : G1 X1 Y2

Cette commande est un mouvement linéaire partant du point précédent et arrivant au point noté sur l'instruction. N'étant pas forcément à distance égale, un pas maximum de 0.4 mm a arbitrairement été défini. Plus long, le mouvement sera divisé en plusieurs points intermédiaires. La forme de l'interpolation linéaire sera une droite décrivant la trajectoire :

$$Y = a * (X_{start} + n * \frac{\Delta X}{i}) + b$$

La pente a sera trouvée par :

$$a = \frac{\Delta Y}{\Delta X} = \frac{Y_{end} - Y_{start}}{X_{end} - X_{start}}$$

L'offset b en utilisant le point d'arrivée :

$$b = Y_{end} - \frac{\Delta Y}{\Delta X} * X_{end}$$

Il reste donc à définir l'incrément i entre chaque point. Ce dernier sera défini par :

$$i = \frac{\sqrt{(\Delta X)^2 + (\Delta Y)^2}}{pas_{max}}$$

Le tout sera itéré n fois jusqu'à atteindre le point d'arrivée.

Interpolation circulaire

Exemple de commande : G2 X1 Y2 I5 J8

Le même principe sera appliqué pour l'interpolation circulaire. Ne pouvant pas réaliser de cercle le système sera divisé en petits segments linéaires de longueurs identiques. La commande s'interprète de la manière suivante : X et Y seront les points d'arrivée et I J représente la distance relative au point de départ et le centre du cercle qui devra être réalisé. Pour ce type d'interpolation le pas minimum sera défini à 0.1 mm. La qualité d'interpolation change donc en fonction du mouvement à réaliser.

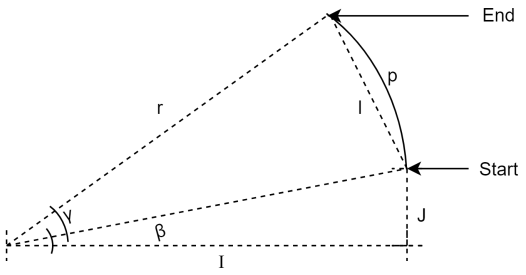


Figure *5.21 – Schématisation de l'interpolation circulaire anti-horaire*

La première étape est de trouver le rayon du cercle et la longueur de la corde :

$$r = \sqrt{I^2 + J^2} \qquad\qquad l = \sqrt{(\Delta X)^2 + (\Delta Y)^2}$$

Avec cette information il est possible d'en déduire la longueur de l'arc :

$$p = 2 * r * \pi * \frac{\gamma}{360} = 2 * r * \pi * \frac{2 * \arcsin\left(\frac{l}{2 * r}\right)}{360}$$

Le principe précédent est réutilisé pour connaître le facteur de division qui devra être appliqué.

$$i = \frac{p}{pas_{max}}$$

Ce facteur est ensuite appliqué non plus à un segment mais à l'angle total permettant ainsi de trouver l'incrément :

$$\delta\gamma = \frac{\gamma}{i}$$

Il faut ensuite trouver l'offset β qui permettra de donner le point de départ de l'incrémentation. Avec ce calcul un cas particulier pourrait arriver. Si I est égal à 0 dans ce cas l'offset est de 90° positif si J est plus grand que 0 si non -90°.

$$\beta = \arctan\left(\frac{J}{I}\right)$$

Un problème similaire doit être résolu pour la partie du calcul de l'angle total. En effet, si le mouvement se trouve dans le 1er ou le 4ème quadrant du cercle trigonométrique, c'est à dire I < 0, l'angle peut être calculé par :

$$Sens\ horaire\ \alpha = \beta - n * \delta\gamma$$

$$Sens\ anti - horaire\ \alpha = \beta + n * \delta\gamma$$

Si ce n'est pas le cas, l'offset doit être augmenté de 180° afin de se retrouver dans le 2ème ou le 3ème quadrant du cercle trigonométrique.
Finalement le système pourra être incrémenté *n* fois à travers les fonctions :

$$X = X_{start} + I + r * \cos(\alpha)$$

$$Y = Y_{start} + J + r * \sin(\alpha)$$

Ces dernières coordonnées seront ensuite transformées en angles par le MGI du robot.

Calcul de vitesse

Pour tous ces mouvements, la vitesse angulaire devra être calculée. La méthode utilisée est appliquée à la transformation des coordonnées cartésiennes en angles. Une vitesse de dessin fixe a été définie pour la trajectoire entre deux points du plan cartésien. Ne pouvant pas aisément transformer les vitesses linéaires en vitesses angulaires, le temps a été pris en référence. Ainsi la première étape est de calculer ce dernier avec la base définie :

$$t_{start-end} = \frac{s}{v_{fixe}} = \frac{\sqrt{(X_{end} - X_{start})^2 + (Y_{end} - Y_{start})^2}}{v_{fixe}}$$

Le temps étant identique entre deux points du plan cartésien ou deux points du plan du robot correspondant, les vitesses angulaires sont donc trouvées par :

$$\omega_{M1} = \frac{\alpha_{end_1} - \alpha_{start_1}}{t_{start-end}} \qquad \omega_{M2} = \frac{\alpha_{end_2} - \alpha_{start_2}}{t_{start-end}}$$

Pour notre application, le signe des vitesses n'est pas important. En effet, les blocs de chez Siemens utilisés pour le pilotage des moteurs choisissent automatiquement si la vitesse donnée doit être positive ou négative.

Après interprétation de ces commandes, le modèle géométrique doit être utilisé.

5.5.2 Modèle géométrique inverse

Données générales du modèle

Le modèle est construit sur les dimensions suivantes :

- $arm_{a_{short}} = 29mm$
- $arm_{a_{long}} = 193mm$
- $arm_b = 200mm$

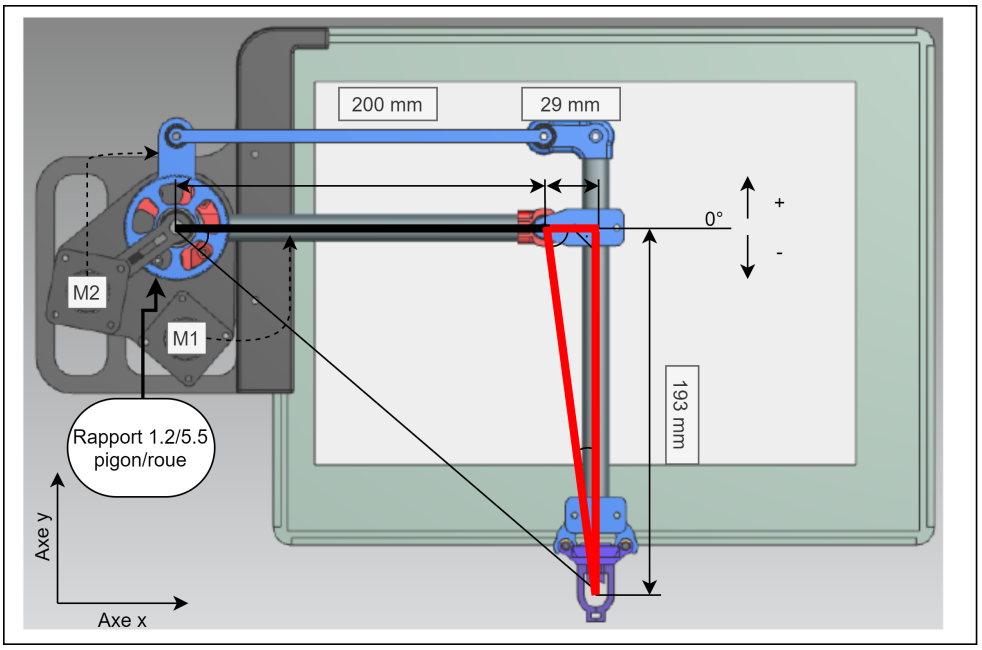


Figure *5.22 – Dimension robot SCARA, ancienne position*

Un rapport de $r_{reduc} = d_{pignon}/d_{roue} = 1.2/5.5$ est appliqué comme réduction entre le pignon et la roue dentée montée sur l'axe central. Les angles moteurs seront donnés positifs si le bras dépasse la ligne horizontale cf. figure 5.22. Sur l'illustration présentée ci-dessus, le moteur de gauche est positionné à $90/r_{reduc} = 412.5°$ positif et le second, à droite, 0°. Il est à noter que cette figure représente l'ancien montage mécanique. Le nouveau sera positionné sur le haut de la feuille A4 impliquant simplement un offset différent avec l'origine (bas gauche feuille A4). Avec ces données le développement du modèle peut commencer.

Les angles des moteurs seront définis par α_1 et α_2 respectivement pour les moteurs 1 et 2.

Moteur 1

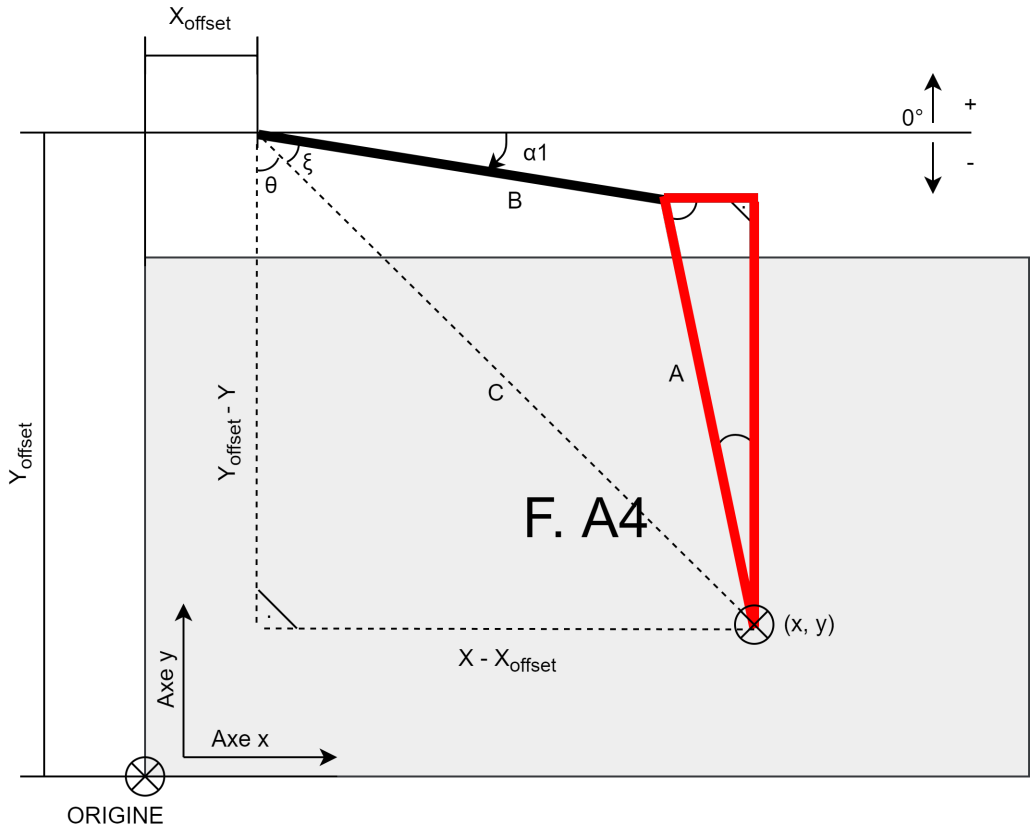


Figure *5.23 – Schéma robot SCARA avec nouveau montage moteur 1*

L'angle développé par le moteur 1 sera exprimé par :

$$\alpha_1 = \frac{\theta + \xi - 90}{r_{reduc}}$$

L'angle ξ définira l'angle entre le bras B du préhenseur et le segment C. θ sera l'angle entre une droite parallèle à l'axe y et C.

C représente le segment entre le centre (axe central) du robot et le préhenseur. Il sera exprimé de la manière suivante :

$$C = \sqrt{(x - x_{offset})^2 + (y_{offset} - y)^2}$$

Avec le théorème du cosinus, ce dernier permet ensuite de trouver ξ :

$$A^2 = B^2 + C^2 - 2 * B * C * \cos(\xi) \Leftrightarrow \xi = \arccos\left(\frac{A^2 - B^2 - C^2}{-2 * B * C}\right)$$

Finalement, θ sera calculé en développant le triangle formé par c et l'axe y.

$$\theta = \arctan\left(\frac{x - x_{offset}}{y_{offset} - y}\right)$$

Moteur 2

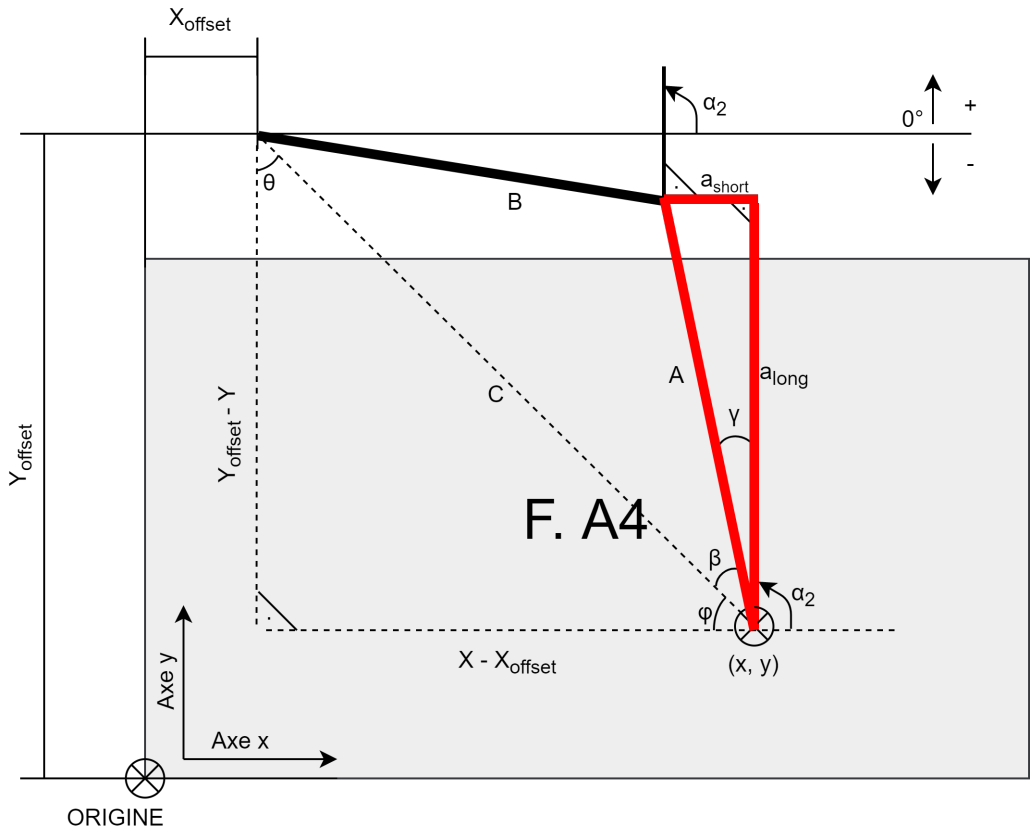


Figure *5.24 – Schéma robot SCARA avec nouveau montage moteur 2*

Le même développement doit être effectué pour le moteur 2. Cette fois, l'angle sera défini par :

$$\alpha_2 = \frac{180 - \phi - \beta - \gamma}{r_{reduc}}$$

En effet, l'angle donné par le triangle rouge formé par les pièces mécanique cf. illustration 5.24 sera identique à celui donné par le moteur 2. Cette transformation peut être faite car le bras a_{long} est relié par une tige rigide forçant le parallélisme des deux pièces.

Connaissant l'angle θ, ϕ peut immédiatement être trouvé :

$$\phi = 180 - 90 - \theta$$

Ensuite, γ est donné par la tangente du triangle rouge :

$$\gamma = \arctan\left(\frac{arm_{a_{short}}}{arm_{a_{long}}}\right) = 8.55°$$

Il reste β qui sera exprimé, comme précédemment, par le théorème du cosinus :

$$B^2 = A^2 + C^2 - 2 * A * C * \cos(\beta) \Leftrightarrow \beta = \arccos\left(\frac{B^2 - A^2 - C^2}{-2 * A * C}\right)$$

Après avoir développé le modèle géométrique inverse, il reste à transmettre ces données à l'automate de commande.

5.5.3 Communication

La partie TCP/IP du système sera gérée par l'objet java UIClient. Cette classe contiendra l'ensemble des méthodes permettant de se connecter et envoyer des messages au PLC. Dans le cadre de ce projet, le PC de contrôle sera le client et l'automate le serveur. En plus de cette classe principale, deux objets additionnels ont été créés.

Message

L'objet Message a comme utilité de construire la partie data de la trame qui sera envoyée. Expliqué au chapitre 5.2.2, les messages devant être envoyés devront respecter une longueur définie. Cette classe a comme objectif principal de construire et concaténer les données dans cette configuration.

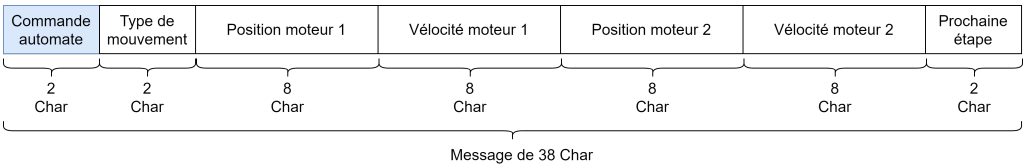


Figure *5.25 – Format de message*

Transaction

Cette classe est simplement une structure permettant de gérer automatiquement toutes les commandes à envoyer pour l'exécution complète d'un dessin.

Une notion essentielle est à noter à ce stade du travail. Certaines Méthodes en Java sont appelées bloquantes. Autrement dit, à l'appel de cette fonction, le programme dans lequel elles sont appelées sera figé jusqu'à la fin de l'exécution. Par exemple, dans ce travail, la méthode readLine() est appelée. Elle permet de lire ce qui est arrivé comme message TCP depuis l'automate. Ainsi, si aucune trame n'arrive, le programme sera bloqué.
Dans le cadre du logiciel développé pour ce travail, il n'est pas envisageable que le système reste figé. Par exemple, le robot doit pouvoir être arrêté à tout moment et l'interface doit régulièrement être mise à jour. Un système d'exécution parallèle ou Thread a donc été mis en place.
L'objet Transaction implémentera l'interface Runnable. Expliqué différemment, une Transaction sera un nouveau processus exécuté parallèlement au principal permettant ainsi de toujours avoir la logique principale fonctionnelle. Dans celle-ci, toutes les commandes pour démarrer et écrire des nouvelles consignes seront générées.

Le même principe a été utilisé pour l'interface graphique développée au chapitre suivant. Au final, trois Thread seront donc utilisés.

Au final, la figure 5.26 présentée ci-dessous, résume la totalité du processus d'un dessin. La partie de droite donne simplement une information complémentaire sur quelle doit être la commande envoyée et quelle est la réponse. Chaque bloc bleu sera l'appel d'une méthode bloquante pour le système. Il devient évident que ce processus doit être traité en dehors du Thread principal. Il est à noter que le système de buffer implémenté dans l'automate n'est pas présent sur ce diagramme et est "invisible" pour le programme développé sur Java. En effet, l'automate enverra simplement une commande "fin de dessin" au début de l'exécution de la première ligne relançant ainsi le cycle pour le remplissage du buffer.

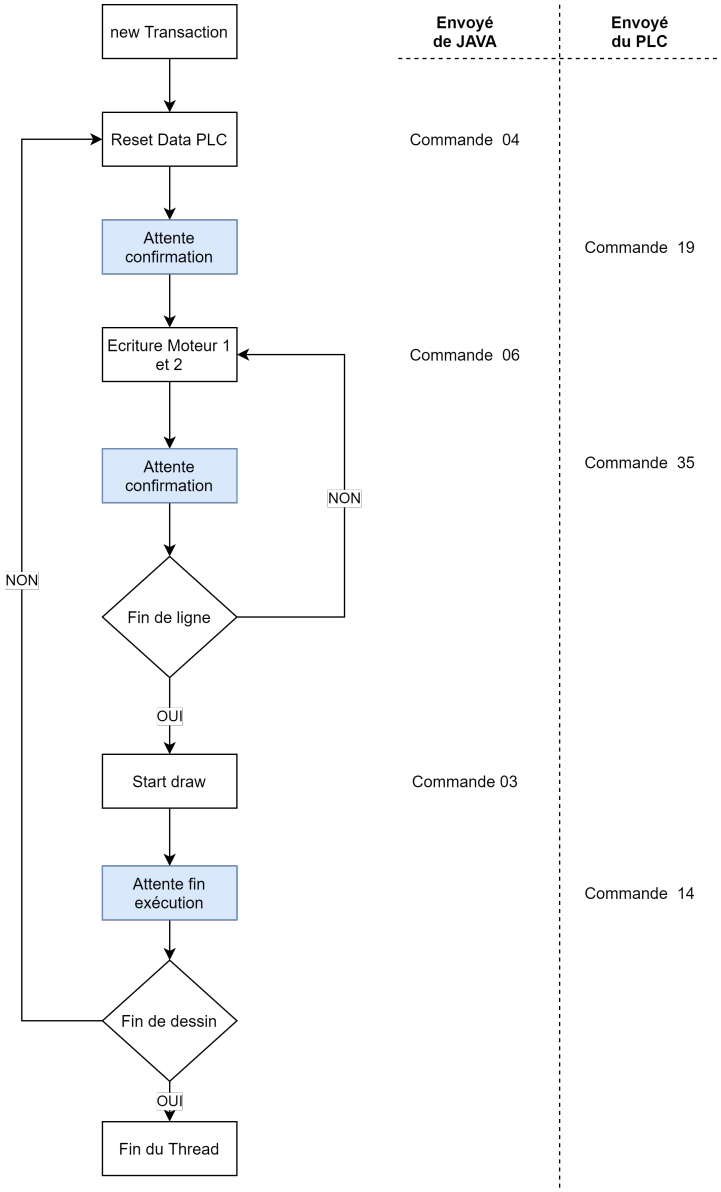


Figure *5.26 – Exécution d'une transaction complète*

5.6 Interface utilisateur

Pour la partie interface utilisateur, aucune demande particulière n'a été énoncée. La condition principale devait être que cette dernière soit simple et compréhensible.

Arbitrairement, plusieurs fonctions ont donc été définies pour cette interface. L'utilisateur doit pouvoir piloter le robot et choisir un dessin le tout en étant le plus visuel possible. Différentes informations doivent être affichées à l'écran comme : l'étape d'exécution du programme, le dessin choisi, la progression du dessin.

L'HMI développée pour ce projet est présentée à la figure ci-dessous.

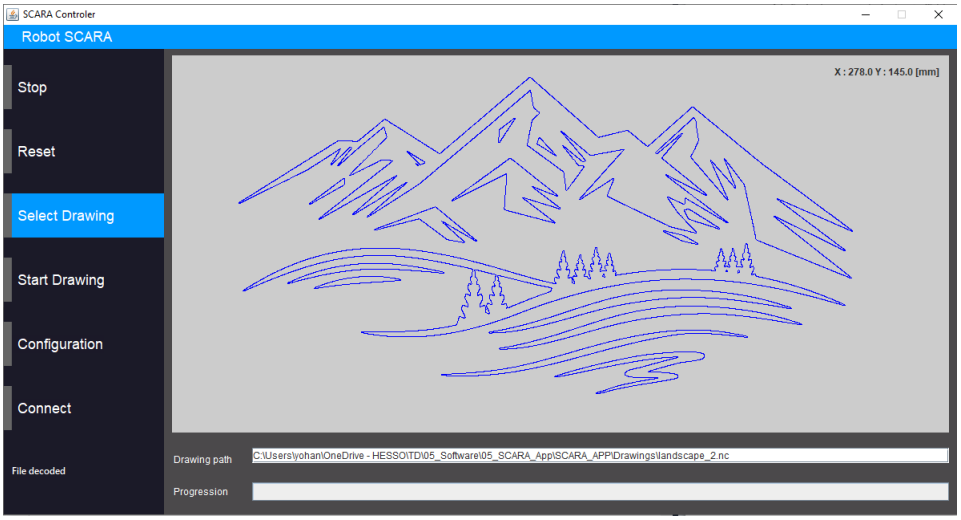


Figure *5.27 – Interface utilisateur*

Cette dernière est séparée en trois zones. La première à gauche est le panneau de commande permettant de piloter le robot. La seconde à droite informe l'utilisateur du dessin sélectionné. Et la dernière, sur le bas, affichera les différentes informations mentionnées précédemment. Afin d'éviter les répétitions, la section suivante présentera non seulement l'utilisation de l'interface mais également le fonctionnement du robot.

5.6.1 Utilisation du robot

Le SCARA doit être initialement démarré en branchant l'alimentation 230V à la prise présente à l'arrière du coffret. Un second câble, RJ45, devra être relié au PC de contrôle avec le port présent également sur l'arrière de l'installation.

Figure *5.28 – RJ45 et alimentation*

La première étape est de configurer manuellement l'adresse IP du port ethernet utilisé sur PC pour la communication avec le robot. Le format à respecter est : "192.168.0.x." avec *"x"* prenant une valeur différente de celle de l'automate. Le programme de contrôle peut ensuite être exécuté. Initialement, le robot sera déconnecté. L'action de départ sera d'appuyer sur le bouton Connect (cf. figure 5.27). Un texte apparaîtra sur le bas gauche informant l'utilisateur à quel IP et PORT (logiciel) le programme essaie de se connecter.

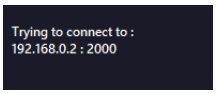


Figure *5.29 – Information de connexion*

Par défaut, l'adresse IP sera 192.168.0.2 et le port 2000. En revanche, si l'automate a été reconfiguré différemment le bouton Configuration (cf. figure 5.27) permet de changer les paramètres du système. Un second panneau (illustration 5.30) s'ouvre et permet de modifier ces valeurs.

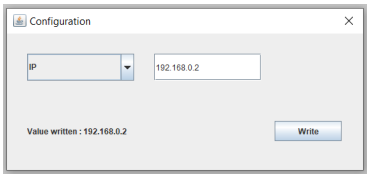


Figure *5.30 – Paneau de configuration*

Les variables pouvant être modifiées sont :

Paramètre	Description
IP	Adresse IP à laquelle se trouve l'automate
PORT	Port de l'automate
REDUCTOR	Coefficient de réduction pignon/roue
LENGHTA_{LONG}	Bras A partie longue [mm] (voir chapitre 5.5.2)
LENGHTA_{SHORT}	Bras A partie courte [mm] (voir chapitre 5.5.2)
LENGTHB	Bras B [mm] (voir chapitre 5.5.2)
OFFSETX	Offset X depuis l'origine de la feuille [mm] (voir chapitre 5.5.2)
OFFSETY	Offset Y depuis l'origine de la feuille [mm] (voir chapitre 5.5.2)
OFFSETM1	Offset de mise à zéro moteur 1 [°]
OFFSETM2	Offset de mise à zéro moteur 2 [°]
MAXLENGHTLINEAR	Distance maximale entre deux points mouvement linéaire [mm] (Qualité d'interpolation)
MAXLENGTHCIRCULAR	Distance maximale entre deux points mouvement circulaire [mm] (Qualité d'interpolation)
SPEED	Vitesse d'écriture [mm/s]

Table *5.3 – Paramètres modifiables*

Dès que le robot est connecté, l'utilisateur en est informé avec le texte en bas à gauche.

Figure *5.31 – Étapes de connection*

Après cela, soit le dessin peut être choisi, soit le robot mis à zéro. L'ordre des actions n'étant pas important, la mise à zéro est présentée par principe en premier.
Pour initialiser le robot, il est conseillé d'enlever le stylo afin de minimiser les risques de collision. Dès que le préhenseur est libre, le bouton Reset peut être appuyé. Le bras sera démarré et complètement rétracté afin de connaître la position initiale du système. En cas de problème, le bouton stop peut être appuyé à tout moment pour arrêter le robot. Une fois la mise à zéro terminée, un texte indique à l'utilisateur que le système est correctement arrivé au point initial. Ensuite, le stylo peut être monté. Le choix du dessin peut, par la suite, être effectué.

L'illustration peut être sélectionnée avec le bouton Select Drawing. Une nouvelle fenêtre (cf. figure 5.36) s'ouvre et permet de choisir le fichier préalablement transformé en GCode par LaserGRBL (fichier en .nc).

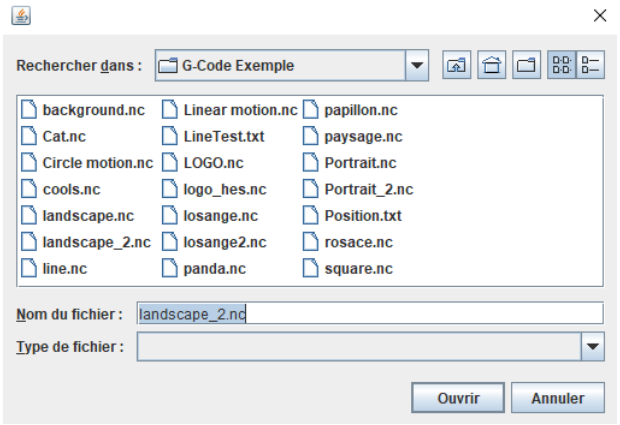


Figure *5.32 – Fenêtre de sélection de dessin*

Après avoir sélectionné le dessin, le bouton "Ouvrir" permet de valider le choix. Immédiatement après, une barre de chargement (figure 5.33) indique que le système est en train de décoder les commandes. Pour des dessins complexes, (plus de 100'000 pts) ce chargement peut atteindre les 5 secondes.

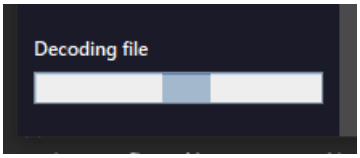


Figure *5.33 – Animation de décodage du dessin*

A la fin du chargement, le dessin est affiché sur la partie de droite de l'écran. Tout en haut à droite, le point maximal qui sera atteint sur la feuille est donné en mm.

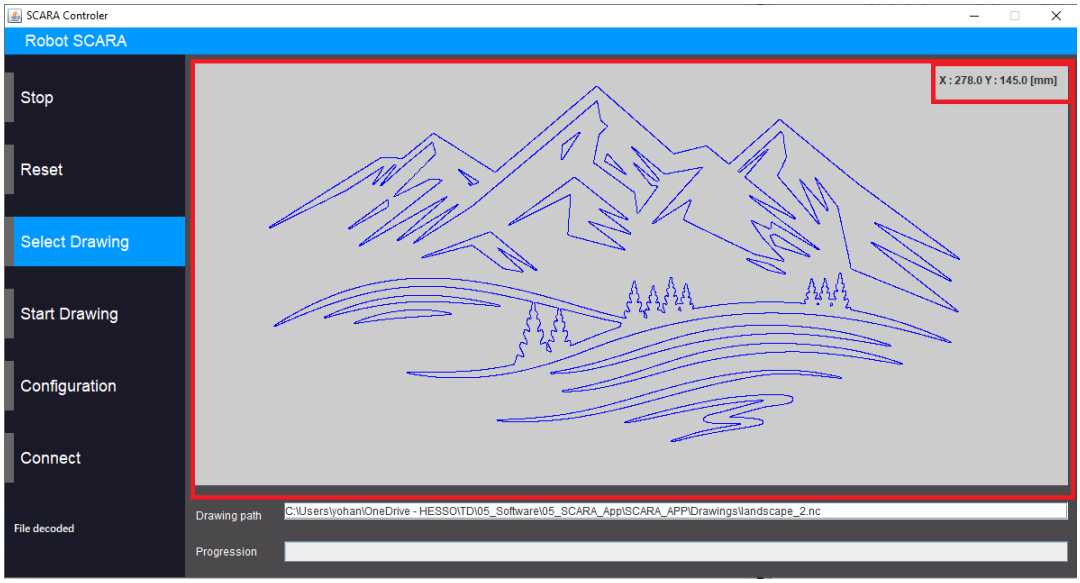


Figure *5.34 – Zone d'affichage dessin décodé*

Finalement, le dessin peut être démarré en appuyant sur le bouton Start Drawing. La progression totale est affichée immédiatement sous l'affichage de l'illustration choisie. (Cf. figure 5.35)

Figure *5.35 – Différents états de progression (25-30-100%)*

En cas de long transfert, par exemple pour des lignes de 1000 points, une seconde barre de chargement sur le bas gauche permet de visualiser l'avancement de l'envoi de la ligne.

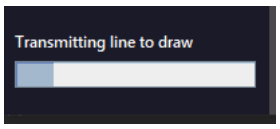


Figure *5.36 – Barre de chargement pour transfert de ligne*

Le dessin sera terminé au moment où le robot se met en position en dehors de la feuille et que l'interface indique en bas à gauche "Draw done". Pour redémarrer un nouveau processus, il suffit alors de choisir un nouveau dessin et d'appuyer sur start drawing. Il est à noter que l'arrêt immédiat du robot peut être fait avec le bouton stop à tout moment.

Pour l'arrêt complet du système, il est conseillé d'appuyer sur le bouton Stop afin de couper la puissance sur les moteurs avant d'éteindre l'installation.
En cas de déconnexion de l'installation (coupure de l'alimentation), l'application sera automatiquement déconnectée. Il suffit alors d'appuyer à nouveau sur le bouton Connect afin de redémarrer le processus. Il est à noter que l'automate n'accepte qu'une connexion unique. Si l'application est déjà démarrée et connectée une seconde instance ne pourra pas piloter le robot.

6 Résultats et observations

Afin d'évaluer le projet, la qualité du dessin a été inspectée. Arbitrairement, plusieurs points seront étudiés :

- la précision de dimension (longueur, angle) ;
- la précision de positionnement ;
- la répétabilité ;
- la qualité de la ligne.

Ces critères seront utilisés comme validation pour toutes les commandes du GCode devant effectuer un trait, c'est-à-dire G1, G2, G3. Toutes les mesures effectuées dans ce chapitre seront faites à la règle ou au pied à coulisse.

Pour le test de dimension et positionnement, un dessin aux géométries connues sera effectué puis mesuré. La répétabilité sera évaluée en réalisant cinq fois cette même trajectoire.

6.1 Mouvement linéaire G1

Mesures complètes en annexe F.1.

Pour ce mouvement un set de consignes aux dimensions suivantes a été donné. Il est à noter que tous les angles sont de 90°.

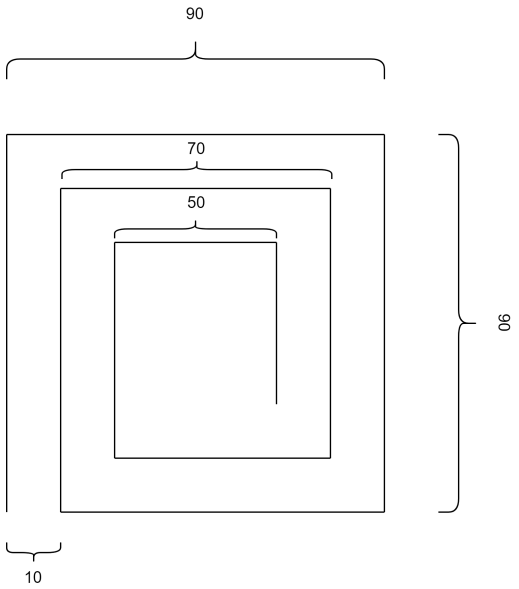


Figure *6.1 – Mouvement de test linéaire*

En général le dessin est de la bonne dimension. L'erreur obtenue pour les segments de la figure 6.3 est d'environ :

Mesures	Erreur
90 mm	±1 mm
70 mm	±0.5 mm
50 mm	±0.5 mm
90°	±0.5 °

Table *6.1 – Mesures mouvement linéaire*

On remarque qu'en règle générale le dessin est dans les bonnes dimensions, avec probablement une erreur croissante en augmentant la longueur des traits. Après inspection de ces derniers, on observe que certains angles ne sont pas parfaitement droits (cf. figure 6.2 agrandissement bas). Ce dernier défaut est probablement dû aux mouvements d'inclinaisons que peut subir le préhenseur à l'appui sur la feuille. En revanche, la précision du positionnement de la figure complète ainsi que la répétabilité sont très bonnes (< 0.5 mm d'erreur).

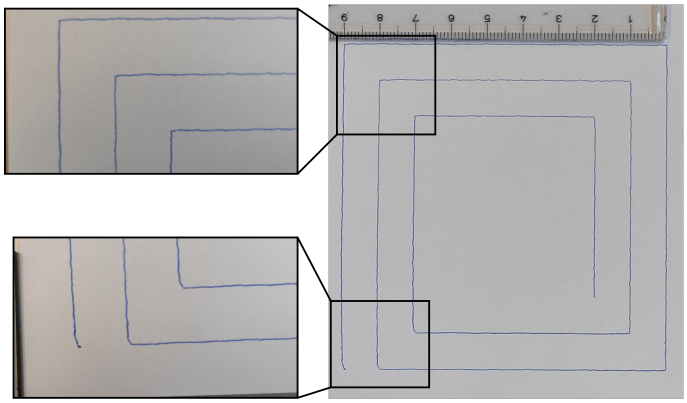


Figure *6.2 – Mouvement linéaire réalisé*

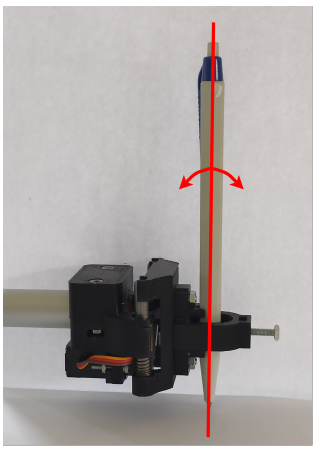


Figure *6.3 – Mouvement du préhenseur*

Afin d'évaluer au mieux la précision sur de longues lignes, un second essai en dessinant un rectangle de 285x200 mm a été réalisé. Contre toute attente, ce second test a été relativement précis. L'erreur obtenue a été mesurée à :

Mesures	Erreur
200 mm	±2 mm
285 mm	±1 mm
90°	±1.5 °

Table *6.2 – Mesures rectangle*

Cet essai a cependant mis en évidence deux zones de dessin critiques (cf. figure 6.4 (cadres noirs)). En effet, ces surfaces atteignent les limites du modèle (MGI) et physiques du robot donnant lieu à de grandes vibrations et à des défauts plus importants. Ces problèmes altèrent alors la qualité de la ligne (ondulations plus importantes). Le reste de la trajectoire est bonne avec des ondulations sur la ligne d'environ ± 0.3 mm.

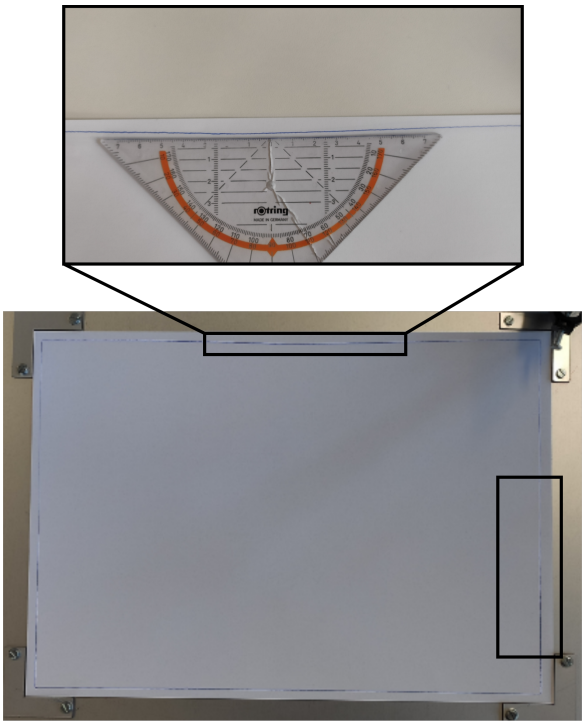


Figure *6.4 – En noir, zones critiques*

En revanche, en compilant toutes les mesures , l'erreur relative moyenne est d'environ de 0.84% ce qui est tout de même un bon résultat.

$$\epsilon_{rel} = \frac{\frac{1}{90} + \frac{0.5}{70} + \frac{0.5}{50} + \frac{2}{200} + \frac{1}{285}}{5} = \pm 0.84\%$$

6.2 Mouvement circulaire G2 - G3

Mesures complètes en annexe F.2.

Un essai équivalent a été réalisé pour les commandes des mouvements circulaires. Dans ce cas-là, la géométrie donnée sera un arc de cercle de 90° d'un rayon de 90 mm. Ayant dessiné sur le mouvement linéaire, les quarts de cercles auront comme point de centre les angles du carré.

Les résultats sont également concluants. Une unique erreur de positionnement a été retrouvée (agrandissement gauche figure 6.5). Comme précédemment, ce défaut vient très probablement du mouvement du préhenseur.

Mesures	Erreur
90 mm	±0.5 mm
90°	±0.5 °

Table *6.3 – Mesures cercles*

$$\epsilon_{rel} = \frac{0.5}{90} = \pm 0.56\%$$

La compilation des mesures donne ainsi une erreur relative d'environ ± 0.56%.

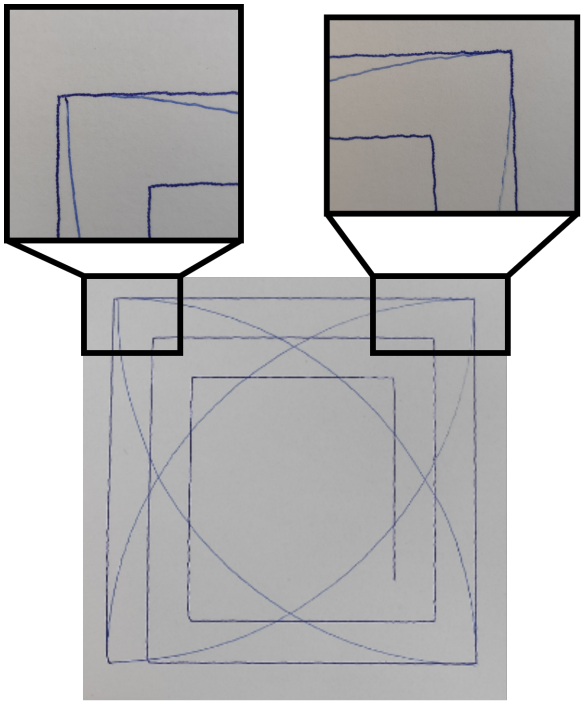


Figure *6.5 – Quatre quarts de cercle effectués par les commandes G2 et G3*

7 Conclusion

Ce travail avait pour but d'améliorer le robot SCARA assemblé l'année passée pour le travail de bachelor Digital Twin. Différents éléments devaient être retravaillés dans les domaines de l'électronique, de la mécanique et de la programmation.

7.1 Résumé des travaux accomplis

Premièrement, au niveau électronique, les drivers moteurs devaient être changés car inadaptés. Après évaluation des possibilités, il s'est avéré qu'un changement de type de moteur apportait une solution plus simple au niveau logiciel et, de surcroît, moins chère. Des moteurs pas à pas et une nouvelle électronique de commande adaptée ont donc été choisis.

Ensuite, au niveau mécanique, plusieurs points devaient être retravaillés. Un coffret de commande combiné au support du robot a été conçu. Une solution de préhenseur motorisé a été développée et améliorée en reprenant les pièces du travail précédent. Puis, des jeux mécaniques impactant la précision du robot ont été recherchés. Il s'est avéré que deux pièces principales devaient être remplacées. De ce fait, les pignons des moteurs ont été changés ainsi que l'axe central (une simple vis) troquée par une pièce usinée spécialement pour ajuster le montage. La mise à zéro a également été réalisée par l'utilisation de capteurs à galet.

Puis, au niveau logiciel, deux programmes ont été développés. Initialement, l'ancien programme, créé par l'étudiant précédent, devait être réutilisé. En revanche, avec le changement de moteur, seules les transformations mathématiques ont été réutilisées. Ce premier programme, fonctionnant sur un PLC, s'occupe de la gestion de l'état du robot et du pilotage des moteurs.

Finalement, un second logiciel a été développé avec le langage de programmation Java. Ce dernier avait pour but, en combinaison avec LaserGRBL, de permettre à l'utilisateur de superviser le robot et de choisir un dessin personnalisé depuis un PC. Ainsi, un fichier en GCode généré sera interprété puis envoyé par séquence à l'automate. En combinaison avec ce programme, une interface utilisateur a été conçue afin d'obtenir un retour visuel clair de l'état du robot et du dessin sélectionné.

7.2 Améliorations

En revanche plusieurs points restent améliorables. Ces derniers concernent principalement le pilotage du robot.

En effet, comme mentionné durant le rapport, un automate de chez Siemens avait été choisi car le programme écrit par l'étudiant précédent devait être réutilisé. Cependant, les S7-1200 (gamme d'automates choisie) ne sont pas optimums pour l'application de ce projet. En effet, le traitement des données par le PLC étant lent, pour de longs traits le système peut rester figer jusqu'à 20 secondes avant d'entreprendre le segment suivant. Une amélioration serait donc de ne pas effectuer les interpolations sur le PC mais sur la carte de pilotage (actuellement l'automate), réduisant ainsi grandement le temps de transmission car bien moins de points seraient envoyés. Ce système nécessiterait en revanche une carte processeur plus puissante permettant d'effectuer très rapidement la conversion de GCode aux angles des moteurs.

Toujours sur le PLC, un problème similaire est ressorti pour le pilotage des moteurs. En effet, l'enchaînement des points ne peut être effectué extrêmement rapidement limitant ainsi la vitesse et la fluidité de dessin du robot. Ce défaut pourrait être également résolu par l'utilisation d'une carte plus adaptée et expressément dédiée au pilotage des deux axes en simultané. Pour ces deux améliorations, un produit pouvant être utilisé pour ce type de commande serait, par exemple, la carte Mega 2560 [1].

Ensuite, pour l'utilisateur, une petite lampe ou un retour visuel pourrait être installé sur le robot afin d'indiquer si l'installation est allumée. Dans le même principe, l'interface homme-machine pourrait être améliorée en créant un affichage visuel plus clair sur l'avancement du dessin. Par exemple, un changement de couleur sur les traits déjà réalisés serait une solution.

Finalement, au niveau mécanique, un jeu a été sous-estimé. En effet, au moment où le bras est totalement déplié ce dernier est très bas en comparaison à la position rétractée. La course du préhenseur pourrait être augmentée pour rattraper cette différence de hauteur, mais il serait préférable de retravailler l'axe central en changeant les roulements par des butées à billes, annulant ainsi totalement le jeu restant au niveau de l'axe central.

1. Carte de commande : https://store.arduino.cc/arduino-mega-2560-rev3 (consulté le 16.08.2021)

7.3 Date et signature

Sion, le 20 août 2021 Aymon Yohan

7.4 Remerciements

Je souhaite exprimer ma gratitude aux personnes qui m'ont aidé dans la réalisation du travail de diplôme :

- Madame **Fariba Moghaddam** *professeure à la HES-SO Valais, pour le suivi du projet et ses conseils/avis techniques*
- Monsieur **Christophe Truffer**, *adjoint scientifique à la HES-SO Valais, pour le suivi du projet et les conseils techniques sur le logiciel TIA Portal*
- Monsieur **Jérôme Darbellay** *collaborateur technique à la HES-SO Valais, pour ses nombreux conseils dans le domaine mécanique*
- Monsieur **Pascal Sartoretti** *adjoint scientifique à la HES-SO Valais pour les commandes et l'impression de pièces 3D*
- L'atelier mécanique *pour la réalisation très rapide des pièces du projet*

Je remercie également toutes les personnes qui auront pris le temps de lire ce rapport et également celles m'ayant soutenu durant ce travail.

A | Logiciels utilisés

TIA Portal

Nom	Totally Integrated Automation Portal
Version	V16
Développeur	Simens AG, 2008-2019
Utilisation	Programmation PLC S7-1200 12/DC/DC/DC
URL	https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html (consulté le 11.08.2021)

StepperOnline

Nom	StepperOnline Launcher
Version	2.0
Développeur	omc-StepperOnline.com
Utilisation	Configuration des drivers
URL	https://www.omc-stepperonline.com/download/STEPPERONLINE_v2.0.0.exe (consulté le 11.08.2021)

Apache Netbeans IDE

Nom	Apache Netbeans IDE
Version	12.4
Développeur	Apache
Utilisation	Programmation Java
URL	https://netbeans.apache.org/ (consulté le 11.08.2021)

LaserGRBL

Nom	LaserGRBL
Version	v1.1
Développeur	LaserGRBL
Utilisation	Transformation de dessin vers GCode
URL	https://lasergrbl.com/ (consulté le 11.08.2021)

Inventor

Nom	Autodesk Inventor professionnal
Version	2020
Développeur	Autodesk
Utilisation	Conception mécanique
URL	https://www.autodesk.ch/fr/products/inventor/overview?panel=buy (consulté le 11.08.2021)

Matlab

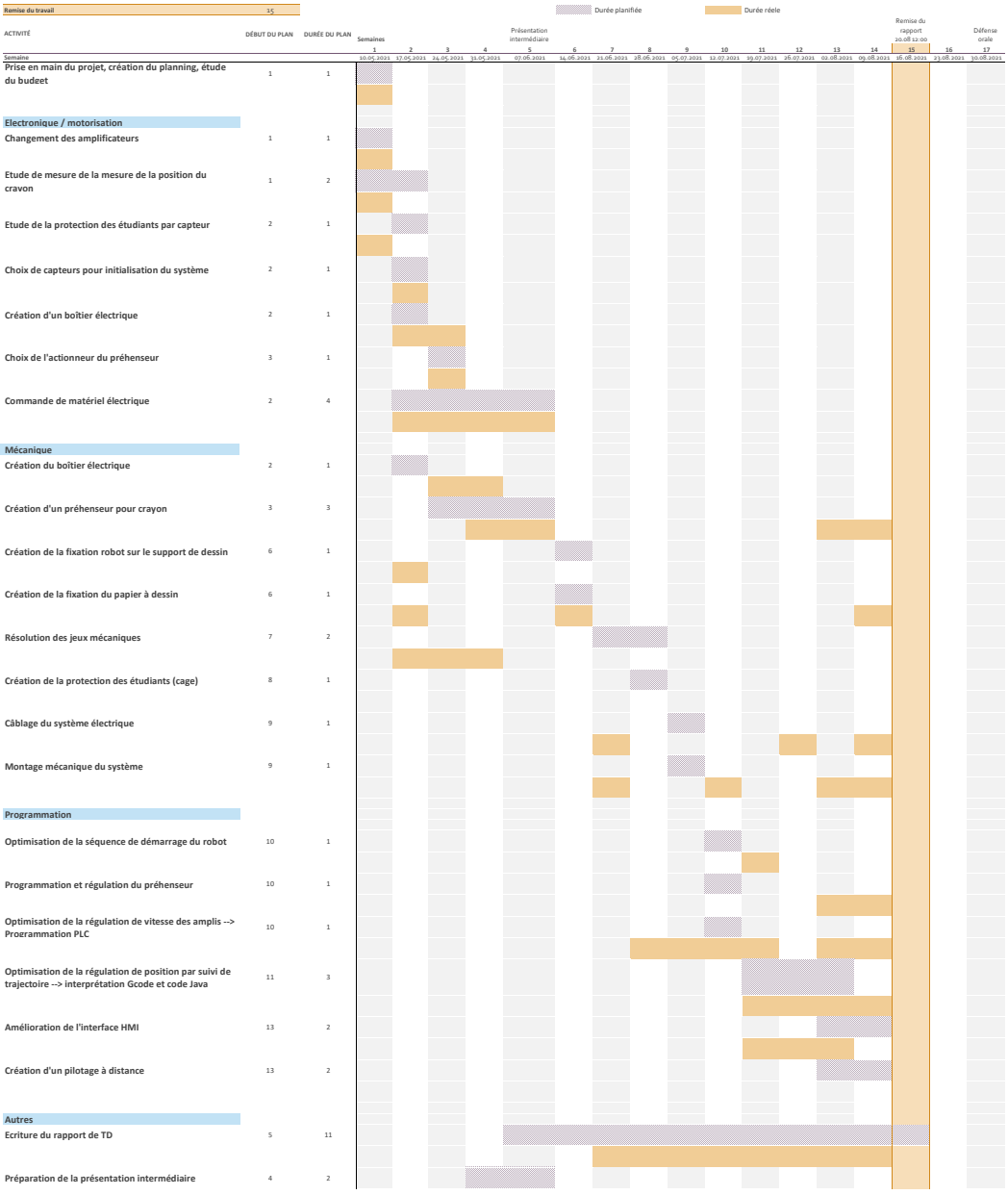
Nom	Matlab
Version	R2019b
Développeur	Mathworks
Utilisation	Vérifications et MGI
URL	https://www.mathworks.com/products/matlab.html (consulté le 11.08.2021)

DrawIO

Nom	DrawIO/diagrams.net
Version	14.6.13
Développeur	JGraph Ltd.
Utilisation	Schématisation et diagrammes
URL	https://www.diagrams.net/ (consulté le 11.08.2021)

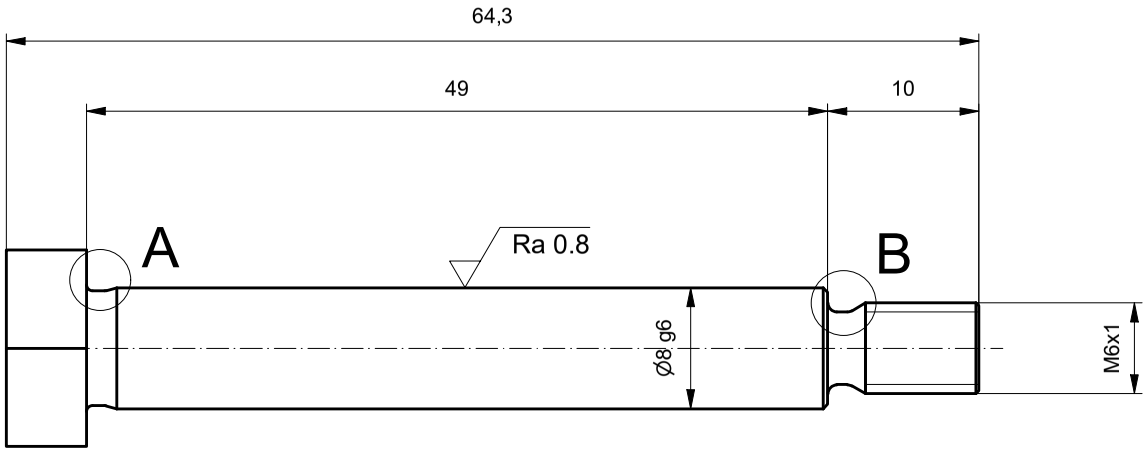
B | Planning

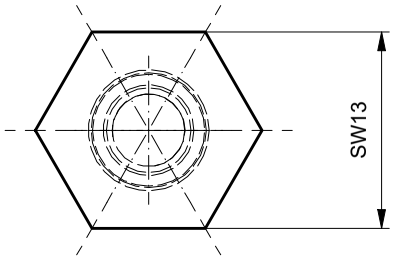
TD Robot SCARA



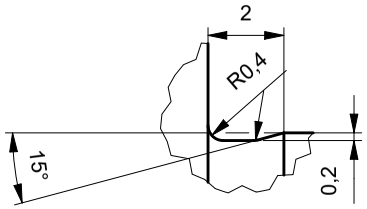
C | Mécanique

C.1 Mises en plan



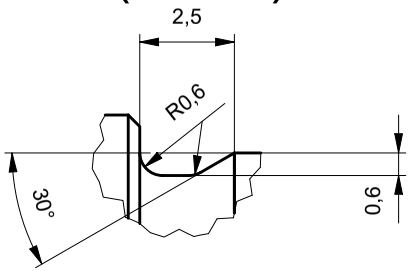


A (5 : 1)



Saignée DIN 509 Forme E

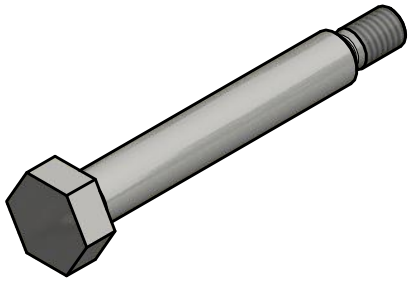
B (5 : 1)



Gorge pour filetage

Ra 6.3

ISO2768-M

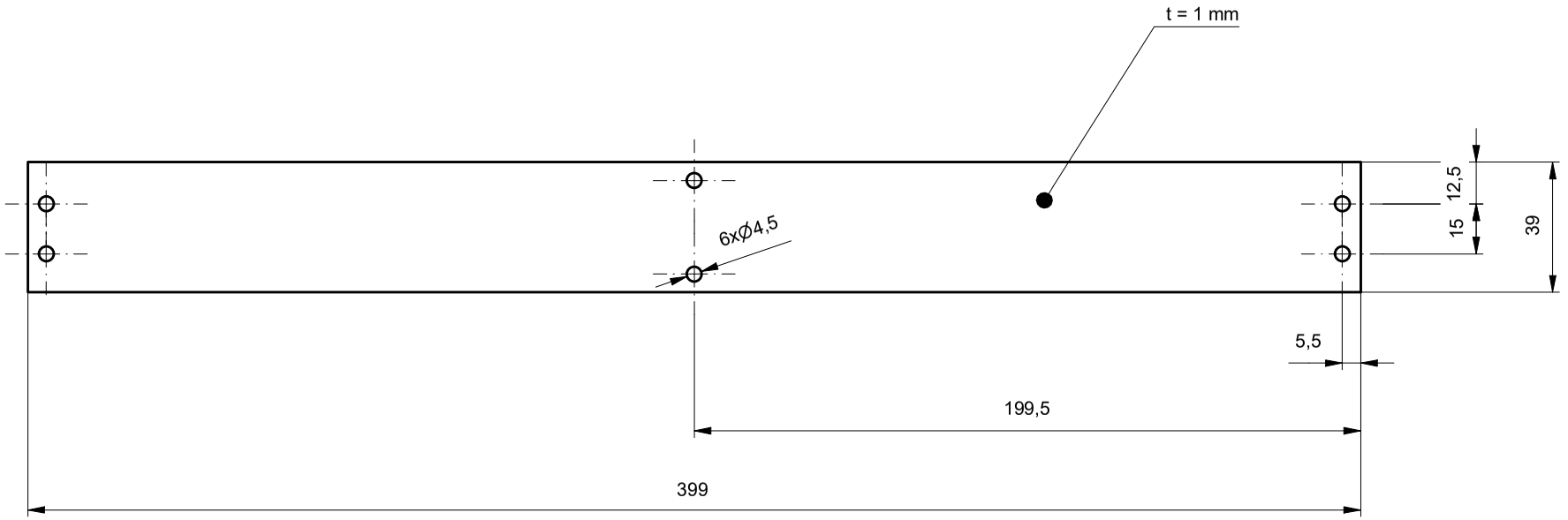


008	1	Acier Inox / Inox Stahl	Axe central
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale

Axe central	Dessiné Gezeichnet	Yohan Aymon	10.06.2021	Echelle Massstab
	Contrôlé Geprüft			2:1

Fichier
Datei C:\Users\yohan\OneDrive - HESSO\TD\03_Mécanique\3_Projet_Inventor\SCARA_Robot\Mise en plan\Axe_central.idw

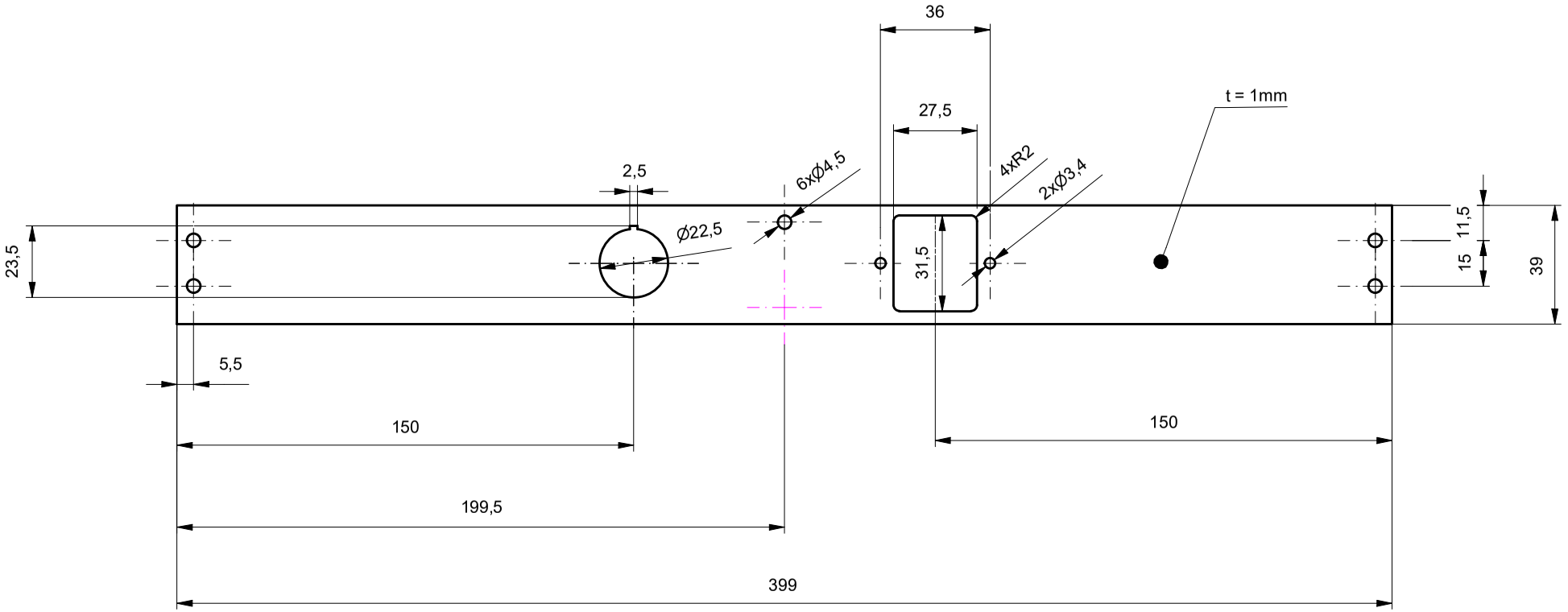
Hes·so VALAIS WALLIS



ISO2768-M

003	1	Aluminium	Côté 1
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale

Face n°1	Dessiné Gezeichnet	Yohan Aymon	29.05.2020	Echelle Massstab
	Contrôlé Geprüft			1:2
Fichier Datei	C:\Users\yohan\OneDrive - HESSO\TD\03_Mécanique\3_Projet_Inventor\SCARA_Robot\Mise en plan\Face_coté_2.id			
Hes·so VALAIS WALLIS				



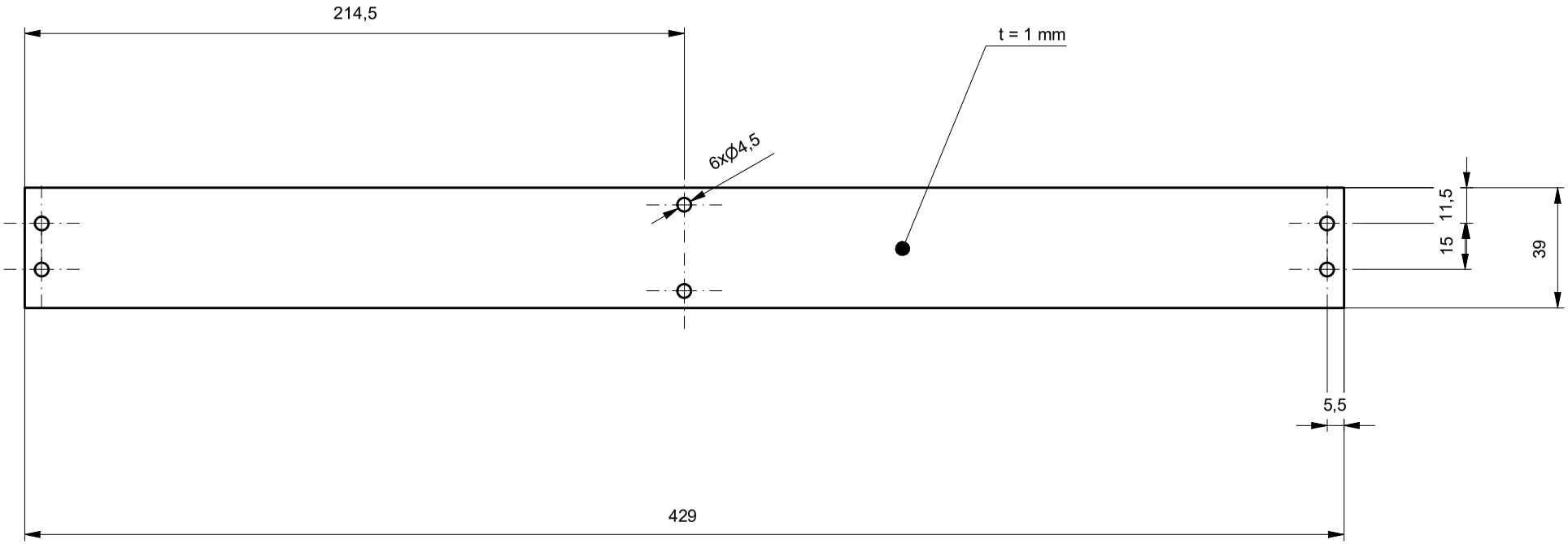
ISO2768-M

005	1	Aluminium 6061	Côté 3
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale

Face n°3	Dessiné Gezeichnet	Yohan Aymon	29.05.2020	Echelle Massstab
	Contrôlé Geprüft			1:2

Fichier
Datei C:\Users\yohan\OneDrive - HESSO\TD\03_Mécanique\3_Projet_Inventor\SCARA_Robot\Mise en plan\Face_coté_3.id

Hes·so VALAIS WALLIS



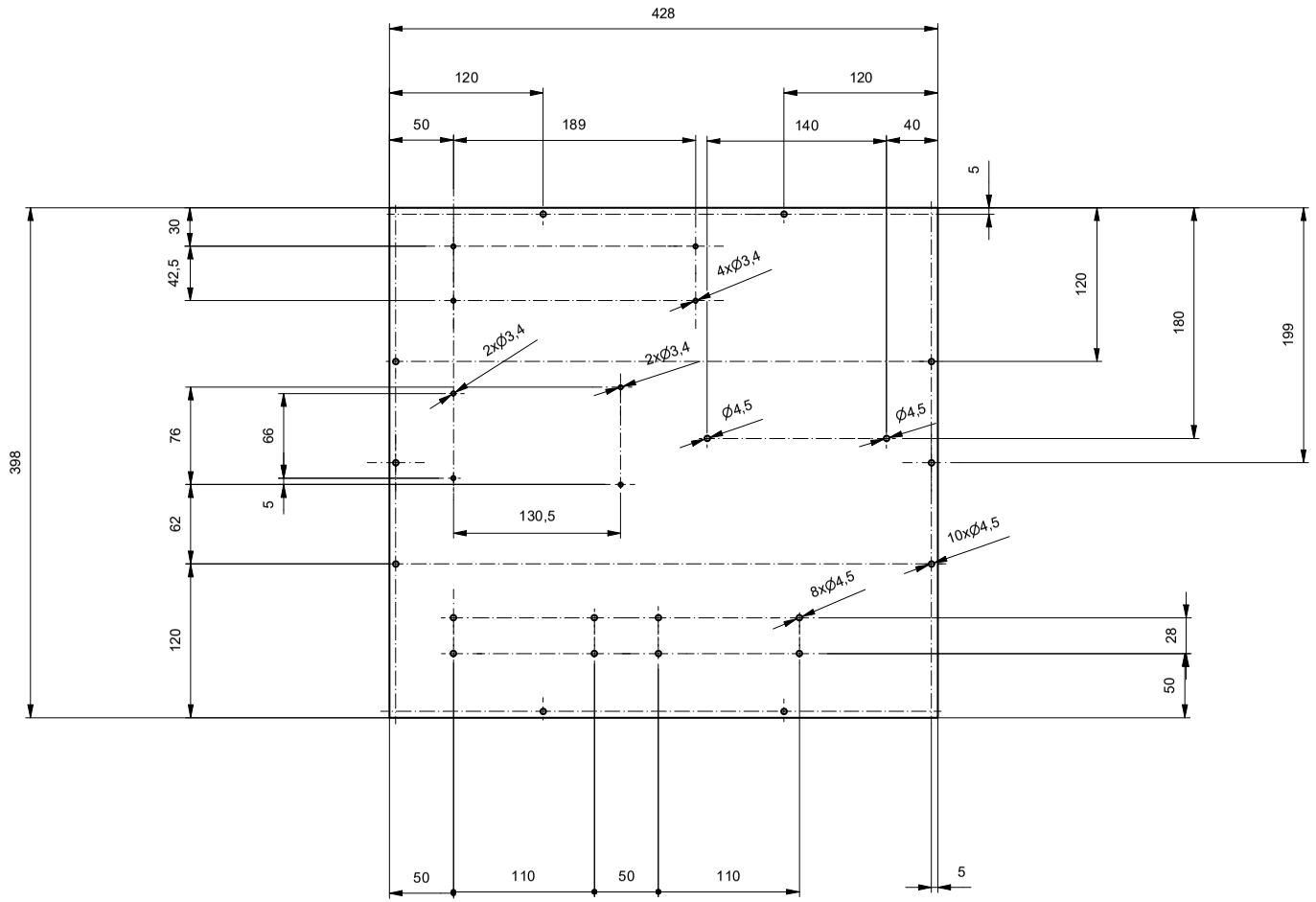
ISO2768-M

004	1	Aluminium 6061	Côté 2
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale

Face n°2	Dessiné Gezeichnet	Yohan Aymon	29.05.2020	Echelle Massstab
	Contrôlé Geprüft			1:2

Fichier
Datei C:\Users\yohan\OneDrive - HESSO\TD\03_Mécanique\3_Projet_Inventor\SCARA_Robot\Mise en plan\Face_coté.idw

Hes·so VALAIS WALLIS



t = 1.5mm

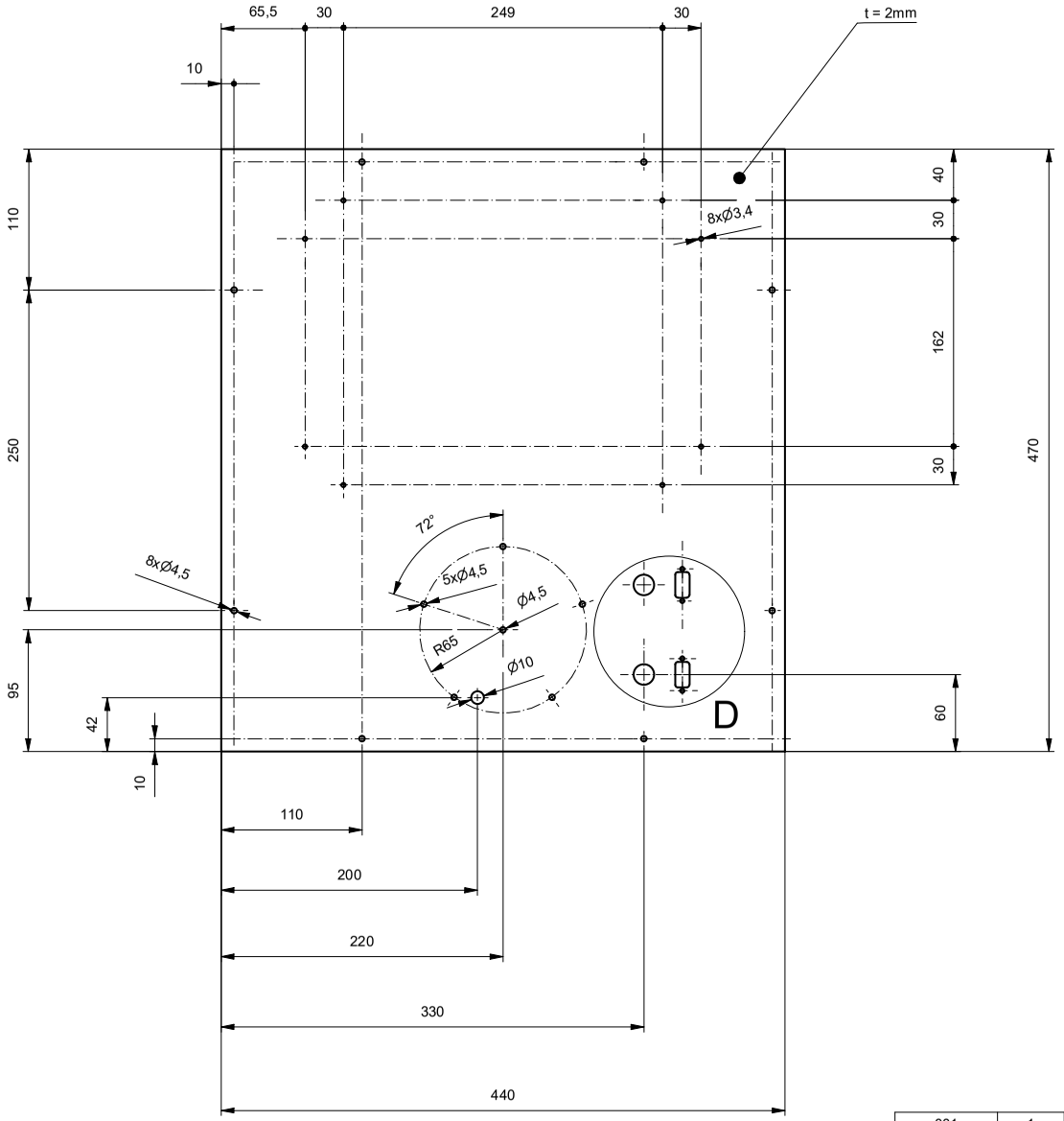
ISO2768-M

002	1	Aluminium 6061	Support composant
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale

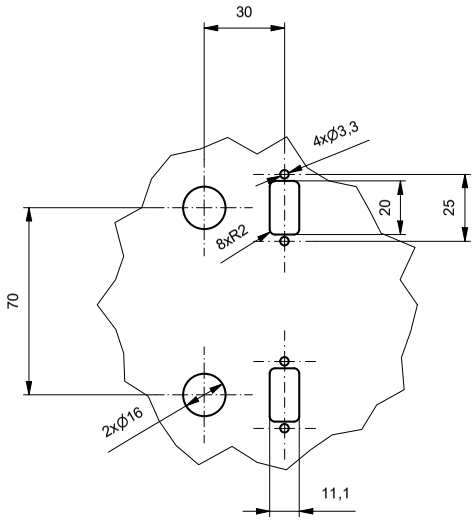
Support électronique	Dessiné Gezeichnet	Yohan Aymon	29.05.2020	Echelle Massstab
	Contrôlé Geprüft			1:4

Fichier
Datei C:\Users\yohan\OneDrive - HESSO\TD\03_Mécanique\3_Projet_Inventor\SCARA_Robot\Mise en plan\Face Fond.idw

Hes·so VALAIS WALLIS



D (1 : 2)



Anodisé

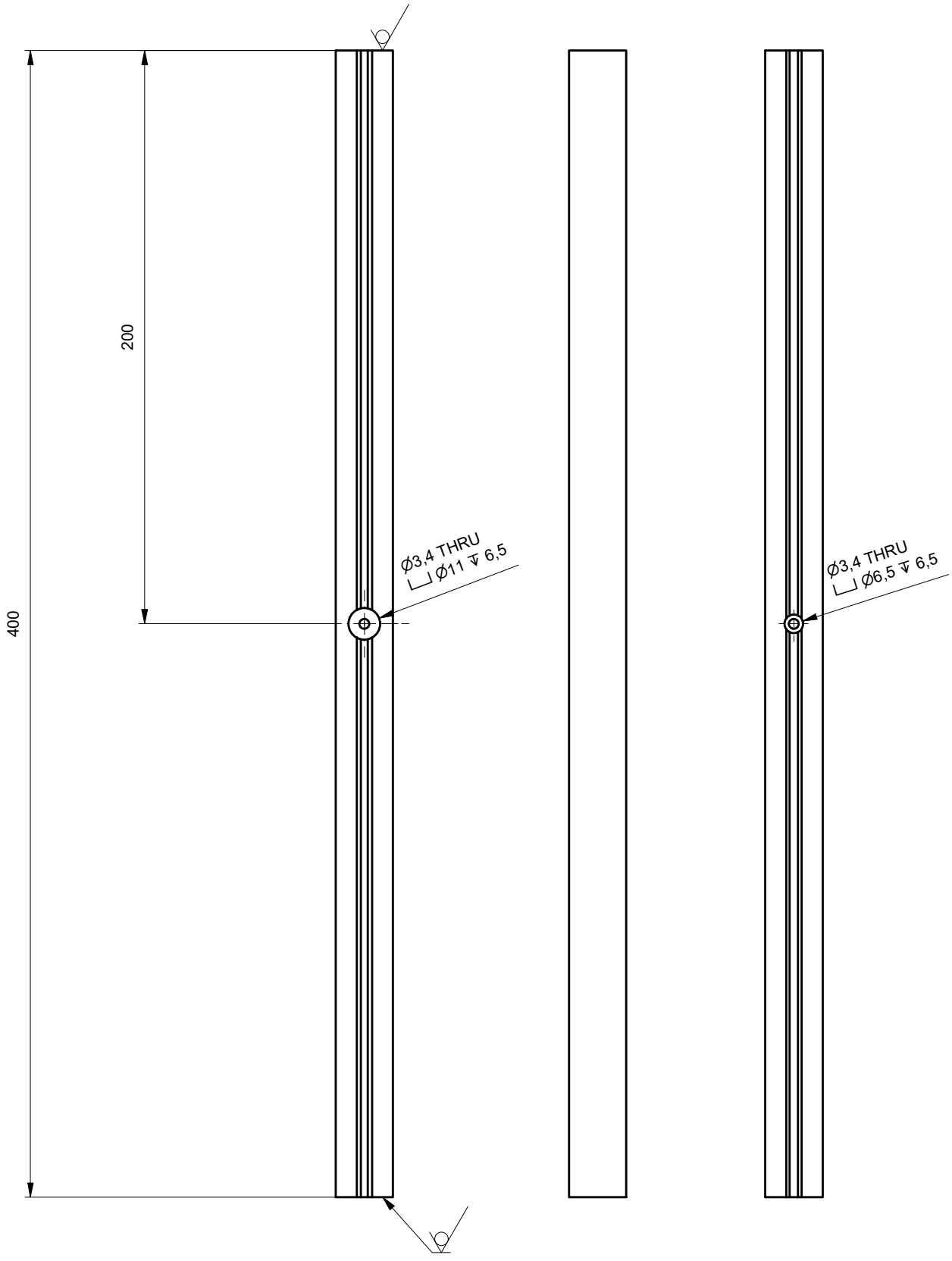
ISO2768-M

001	1	Aluminium 6061	Support robot
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale

Support robot	Dessiné Gezeichnet	Yohan Aymon	29.05.2020	Echelle Massstab
	Contrôlé Geprüft			1:4

Fichier
Datei C:\Users\yohan\OneDrive - HESSO\TD\03_Mécanique\3_Projet_Inventor\SCARA_Robot\Mise en plan\Face_superieur

Hes·so VALAIS WALLIS

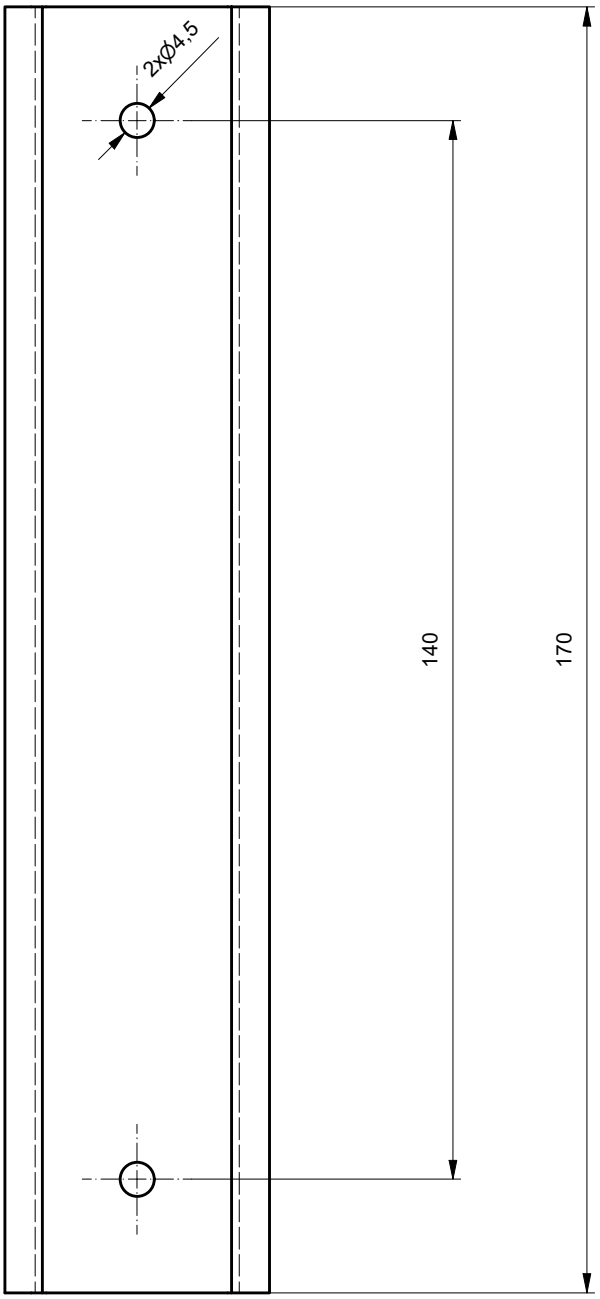


007	1	Aluminium	Profilé lamage

Profilé	Dessiné Gezeichnet		09.06.2021	Echelle Massstab
	Contrôlé Geprüft			1:2

Fichier Datei C:\Users\yohan\OneDrive - HESSO\TD\03_Mécanique\3_Projet_Inventor\SCARA_Robot\Mise en plan\Profile_Base_lo

Hes·so VALAIS WALLIS



006	1	Aluminium 6061	Rail DIN
Pos. Pos.	Quantité Menge	Matière Werkstoff	Dénomination / Caractéristiques Benennung / Merkmale

Rail DIN	Dessiné Gezeichnet		09.06.2021	Echelle Massstab
	Contrôlé Geprüft			1:1

Fichier
Datei C:\Users\yohan\OneDrive - HESSO\TD\03_Mécanique\3_Projet_Inventor\SCARA_Robot\Mise en plan\Rail_DIN.idw

Hes·so VALAIS WALLIS

V1 profilés 30x30

Fournisseur	Description	Qty	No article	Dimension	prix unitaire	prix tot
Item	Multiblock 6 PA, noir	24	0.0.419.58			
Item	Profilé 6 30x30 1N léger, naturell	4	0.0.439.43	380		
Item	Profilé 6 30x30 1N léger, naturell	4	0.0.439.44	410		
Item	Profilé 6 30x30 2N90 léger, naturel	4	0.0.439.45	120		
Item	Fixation automatique 6 30	16	0.0.672.86			
Item	Ecrou 6 St M4, zingué	8	0.0.419.46			
Item	Vis Chc tête bombée M5x10, zingué	24	8.0.000.06			
Item	Vis Chc tête bombée M4x8, zingué	10	8.0.001.98			

V2 Profilé 20x20 (Variante choisie)

Fournisseur	Description	Qty	No article	Dimension	prix unitaire	prix tot
Item	Multiblock 5 PA, noir	24	0.0.370.71			
Item	Profilé 5 20x20 1N, naturel	5	0.0.437.74	400		
Item	Profilé 5 20x20 1N, naturel	4	0.0.437.74	430		
Item	Profilé 5 20x20 2N90, naturell	4	0.0.437.66	80		
Item	Fixation automatique 5 20	18	0.0.672.88			
Item	Ecrou 5 St M4, zingué	8	0.0.370.06			
Item	Vis Chc tête bombée M4x8, zingué	8	8.0.001.98			
Item	Vis Chc tête bombée M4x10, zingué	24	8.0.002.01			
Item	Pied D20, M5x45, noir	4	0.0.464.75			
Item	Tôle Al 2mm, anodisé naturel	1	0.0.473.08	440x470		
Item	Tôle Al 2mm, brut (non dégraissé)	1	0.0.428.27	398x428		
Total					357.1	

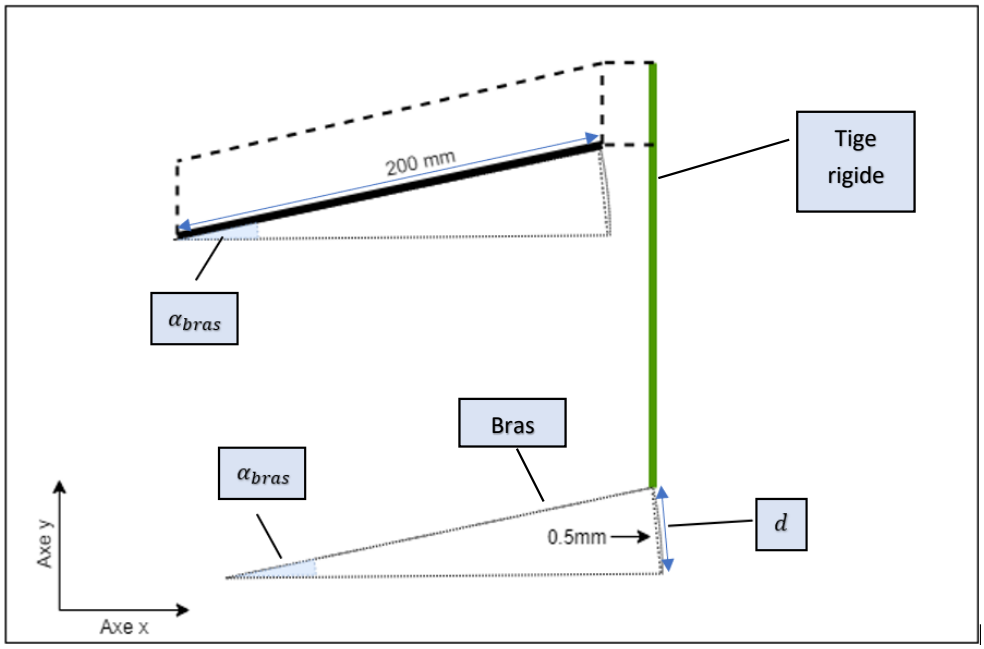
Votre panier

Des frais supplémentaires sont ajoutés pour atteindre le montant minimum de commande de CHF 30.00

Qté	Description	Supprimer	Prix	Sous-total
2	CA 0.45/45.10	✖	4.00	CHF 8.00
2	CA 0.50/65.00	✖	4.00	CHF 8.00
2	CA 0.45/32.00	✖	4.00	CHF 8.00
2	C02400162000M	✖	7.10	CHF 14.20
Modifier			**Sous-total**	CHF 38.20
			Frais de port	CHF 14.00
			TVA 7.7%	CHF 4.00
			Total	CHF 56.20

D | Microstepping et motorisation

D.1 Micro pas M1



Le pas définit l'angle entre chaque position stable du moteur.

Déplacement minimal d donne un angle de :

$$\alpha_{bras} = 2 * \arcsin\left(\frac{\frac{d}{2}}{Bras}\right) = 2 * \arcsin\left(\frac{0.25}{200}\right) = 0.143 \left[\frac{°}{pas}\right]$$

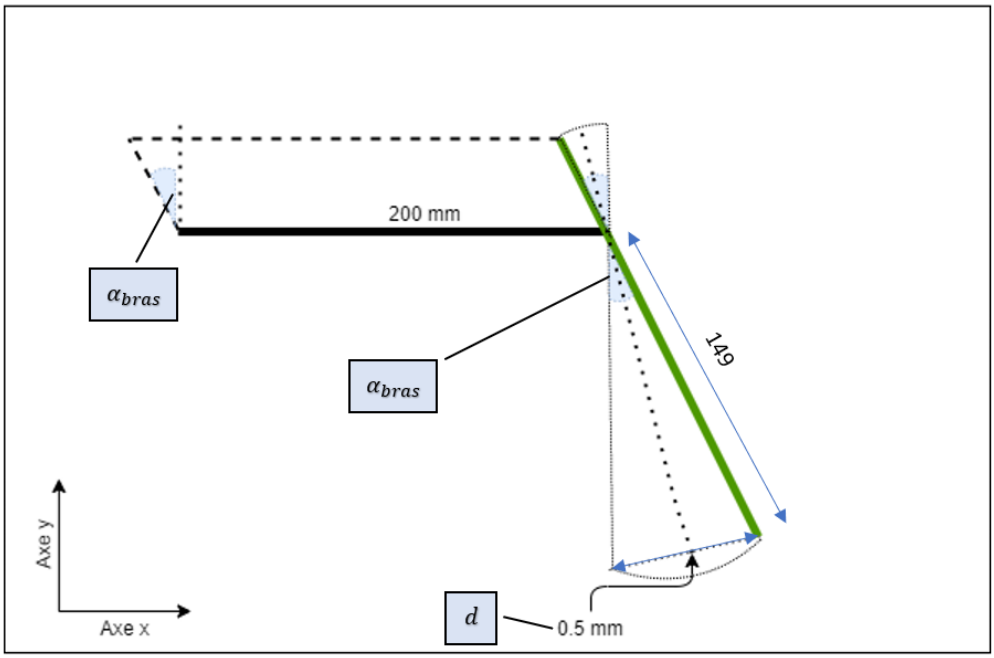
Cet angle pour un pas est directement appliqué par la grande roue dentée (angles opposés par le sommet). L'angle effectué sur le pignon sera l'angle de la grande roue multiplié par le rapport entre le diamètre de la roue et le pignon :

$$\alpha_{pignon} = \alpha_{bras} * \frac{D_{roue}}{D_{pignon}} = 0.143 * \frac{55\ [mm]}{12\ [mm]} = 0.6554[°]$$

Les pulsations **minimums** par révolution seront données par :

$$\frac{Pulse}{rev} = \frac{360\ °}{\alpha_{pigon}} = \frac{360°}{0.6554°} = \mathbf{549.28}\left[\frac{\boldsymbol{puls.}}{\boldsymbol{rev.}}\right]$$

D.2 Micro pas M2



Le pas définit l'angle entre chaque position stable du moteur.

Déplacement minimal d donne un angle de :

$$\alpha_{bras} = 2 * \arcsin\left(\frac{\frac{d}{2}}{Bras}\right) = 2 * \arcsin\left(\frac{0.25}{149}\right) = 0.1922 \left[\frac{^\circ}{pas}\right]$$

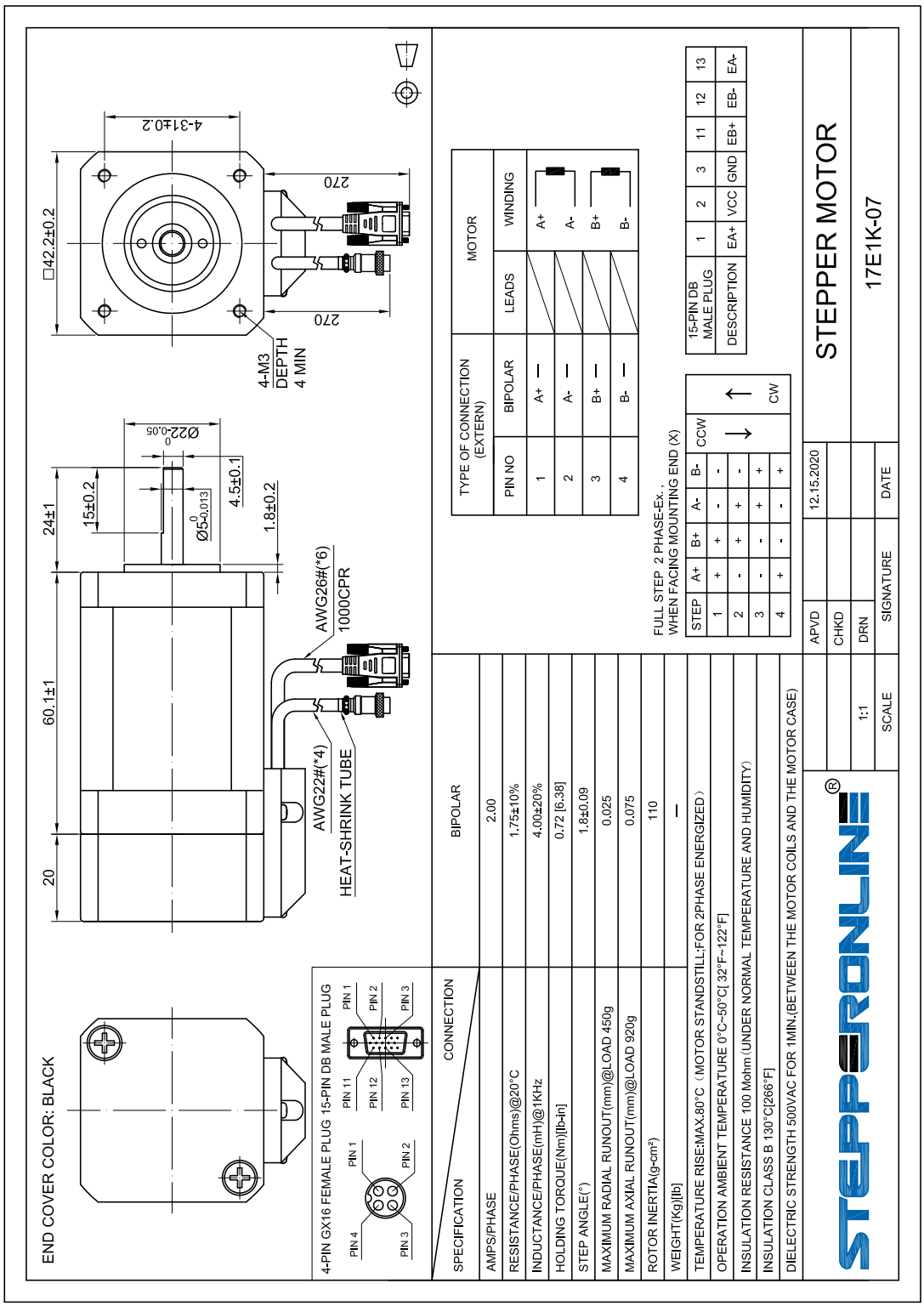
Cet angle pour un pas est directement appliqué par la grande roue dentée (angles opposés par le sommet). L'angle effectué sur le pignon sera l'angle de la grande roue multiplié par le rapport entre le diamètre de la roue et le pignon :

$$\alpha_{pignon} = \alpha_{bras} * \frac{D_{roue}}{D_{pignon}} = 0.1922 * \frac{55\ [mm]}{12\ [mm]} = 0.881[^\circ]$$

Les pulsations **minimums** par révolution seront données par :

$$\frac{Pulse}{rev} = \frac{360\ ^\circ}{\alpha_{pigon}} = \frac{360^\circ}{0.881^\circ} = \mathbf{408.6} \left[\frac{\boldsymbol{puls.}}{\boldsymbol{rev.}}\right]$$

D.3 Moteur pas à pas



E | Achats - électronique

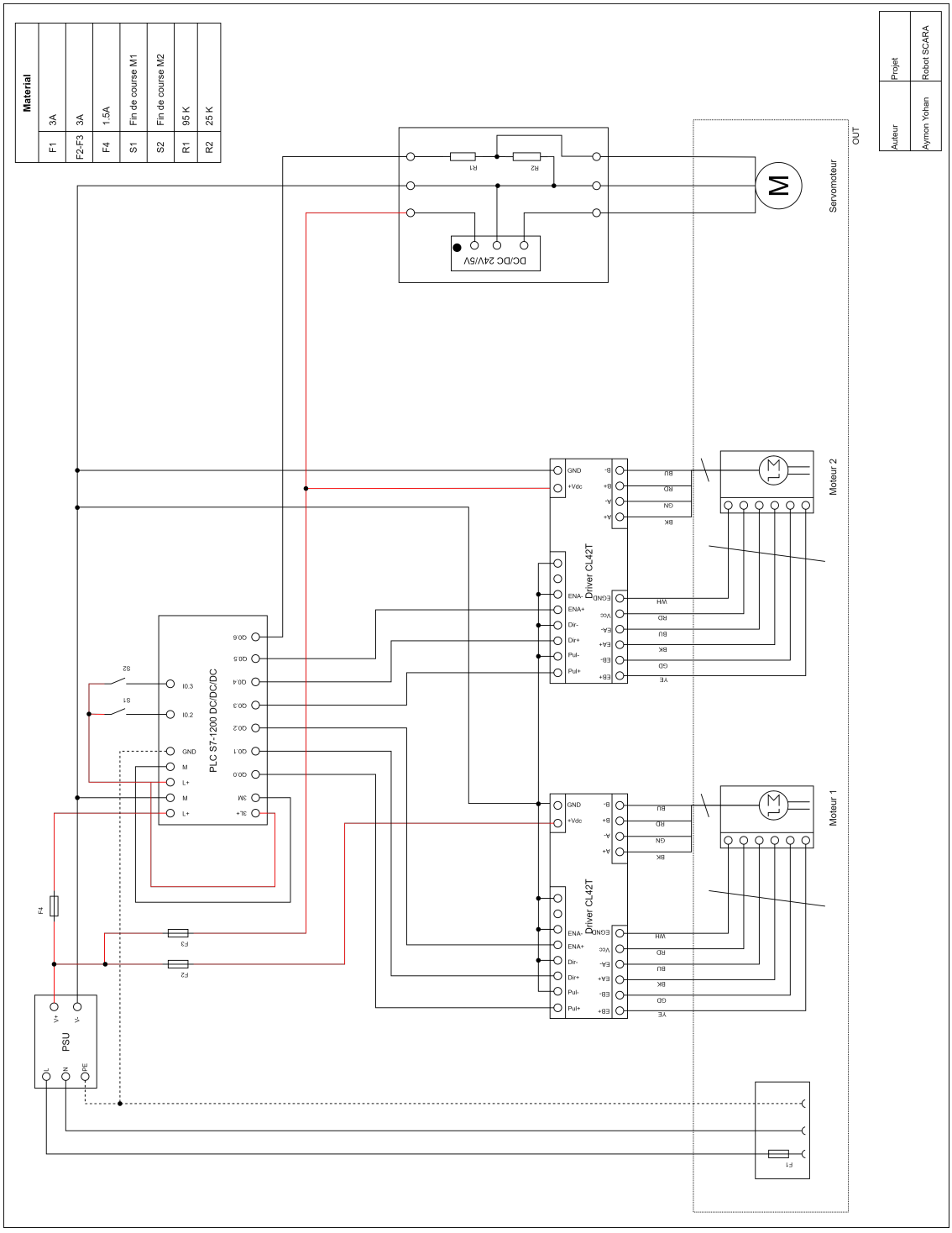
Variante 1

Fournisseur	Type	Qty	No art.	Prix / pièce	Prix total
Distrelec	Fin de course	2	135-72-450	2.35	4.7
Distrelec	DC/DC 1W 24/5 V	1	301-13-061	4.55	4.55
Fauhlaber	Carte MC 3001 P RS	2			0
Fauhlaber	Accessoire carte 6500.01807	2			0

Variante 2 (Variante choisie)

Fournisseur	Type	Qty	No art.	Prix / pièce	Prix total	A commander	Site
Distrelec	Fin de course	2	135-72-450	2.35 CHF	4.7	oui	
Distrelec	Prise D-Sub 15 Femelle	2	144-04-547	13.60 CHF	27.2	oui	
Distrelec	Alimentation à découpage, 24V, 8.4A	1	301-34-213	54.08 CHF	54.08	oui	
Distrelec	Connecteur RJ45	1	110-25-868	16.87 CHF	16.87	oui	
Distrelec	Câble RJ45	1	301-66-882	6.10 CHF	6.1	oui	
Distrelec	Bornier fusible	3	301-65-551	7.20 CHF	21.6	oui	
Distrelec	Fusible 5x20 3 A	5	301-71-816	0.36 CHF	1.8125	non	
Distrelec	Fusible 5x20 1.25A	5	301-71-803	0.63 CHF	3.15	non	
Distrelec	Connecteur DIN	2	301-73-780	0.77 CHF	1.54	non	
Mouser	Servomoteur 0-180°	1	426-SER0049	4.69 CHF	4.69	oui	https://www.dfrobot.com/product-2120.html
Mouser	DC/DC converter	1	495-TSR-1-2450	5.01 CHF	5.01	oui	
Mouser	2GT Pulley	2	485-1251	7.44 CHF	14.88	oui	
Mouser	Connecteur avec fusible C14	1	693-6200.2300	3.24 CHF	3.24	oui	
Mouser	Borniers fixes Terminal block, PCB mount	2	490-TB001-500-03BE	0.81 CHF	1.61	oui	
Fruugo	Connecteur gx16 4 pins	2	52179481-105042429	3.95	7.9	oui	
StepperOnline	Stepper motor + Electronique	2	1-CL42T-P05	$ 150.00	300	oui	
Total					**474.3825**		

E.1 Electrical Wiring



F | Essais et observations

Mesures linéaires [mm] set 1			
90	70.5	50	
90.5	70.5	50.5	
90.1	70.5	50.4	
90	70.3	50.3	
91	70.3		
90.8	70.5		
91			
90.5			

Mesures linéaires [mm] set 2
285.5
286
284.5
285.7
284.5
284.5

Mesures circulaires [mm]			
90.1	89.5	90.5	90.4
90.5	90.3	90	90.5
90.4	90.3	90.2	90
89.7	90.3	90.15	89.8
89.5	90.3	90.4	90.5

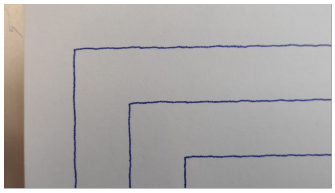
F.1 Essais G0 - G1

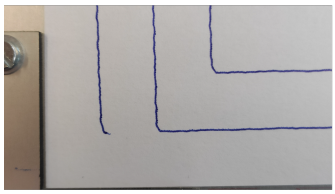
Liste d'instructions donnée

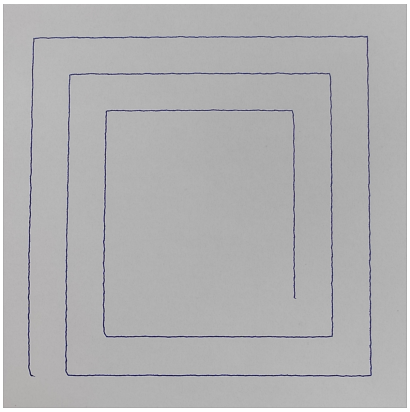
```
M3
S1000
G0 X10 Y10
G1 X10 Y100
G1 X100 Y100
G1 X100 Y10
G1 X20 Y10
G1 X20 Y90
G1 X90 Y90
G1 X90 Y20
G1 X30 Y20
G1 X30 Y20
G1 X30 Y80
G1 X80 Y80
G1 X80 Y30
S0
M5
```

Répétabilité

Traits parfaitement superposés.

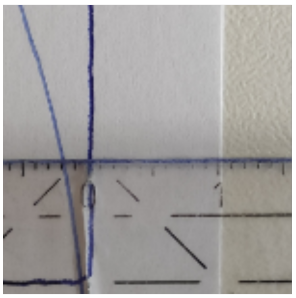


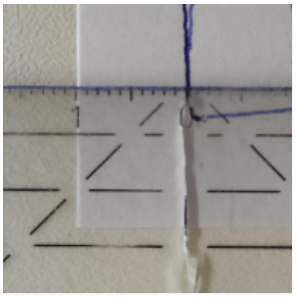




Précision de positionnement

Résultats d'erreur inférieurs à 0.5 mm





F.2 Essais G2 - G3

Liste d'instructions donnée

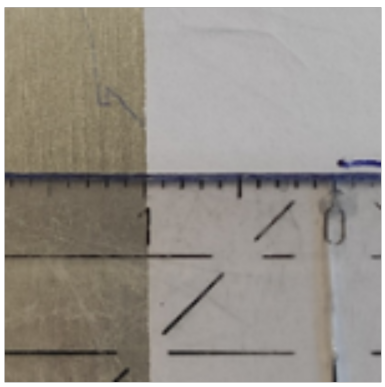
```
M3
S1000
G0 x10 y10
G2 x100 y100 I90 J0
S1000
G0 X10 Y10
G3 x100 y100 I0 J90
S1000
G0 X10 Y100
G3 x100 y10 I90 J0
S1000
G0 X10 Y100
G2 X100 Y10 I0 J-90
M5
```

Répétabilité

Traits parfaitement superposés.

Précision de positionnement

Résultats d'erreur inférieurs à 0.5 mm



Bibliographie

[1] Wikipedia. *IP description*. url : https://fr.wikipedia.org/wiki/Adresse_IP. (accessed : 12.08.2021).

[2] Wikipedia. *TCP description*. url : https://fr.wikipedia.org/wiki/Transmission_Control_Protocol. (accessed : 12.08.2021).

[3] Hiroshi Makino. « Development of the SCARA ». In : *Journal of Robotics and Mechatronics* 26.1 (2014), p. 5-8. doi : 10.20965/jrm.2014.p0005.

[4] Fanuc. *SCARA Avantages*. url : https://www.fanuc.eu/ch/fr/robots/page-de-filtre-des-robots/scara-series/selection-support. (accessed : 18.08.2021).

[5] Debons Sébastien. « Jumeau numérique (Digital Twin) ». 2020.

[6] drive.tech. *4 quadrants*. url : https://drive.tech/fr/stream-content/controleurs-1-q-contre-controleurs-4-q. (accessed : 30.05.2021).

[7] hackaday.com. *Microstepping*. url : https://hackaday.com/2016/08/29/how-accurate-is-microstepping-really/. (accessed : 30.05.2021).

[8] dfrobot.com. *servomotor*. url : https://www.dfrobot.com/product-2120.html. (accessed : 25.07.2021).

[9] Wikipedia. *Programmation de commande numérique*. url : https://fr.wikipedia.org/wiki/Programmation_de_commande_num%5C%C3%5C%A9rique. (accessed : 25.07.2021).

[10] laserGRBL. url : https://lasergrbl.com/. (accessed : 25.07.2021).