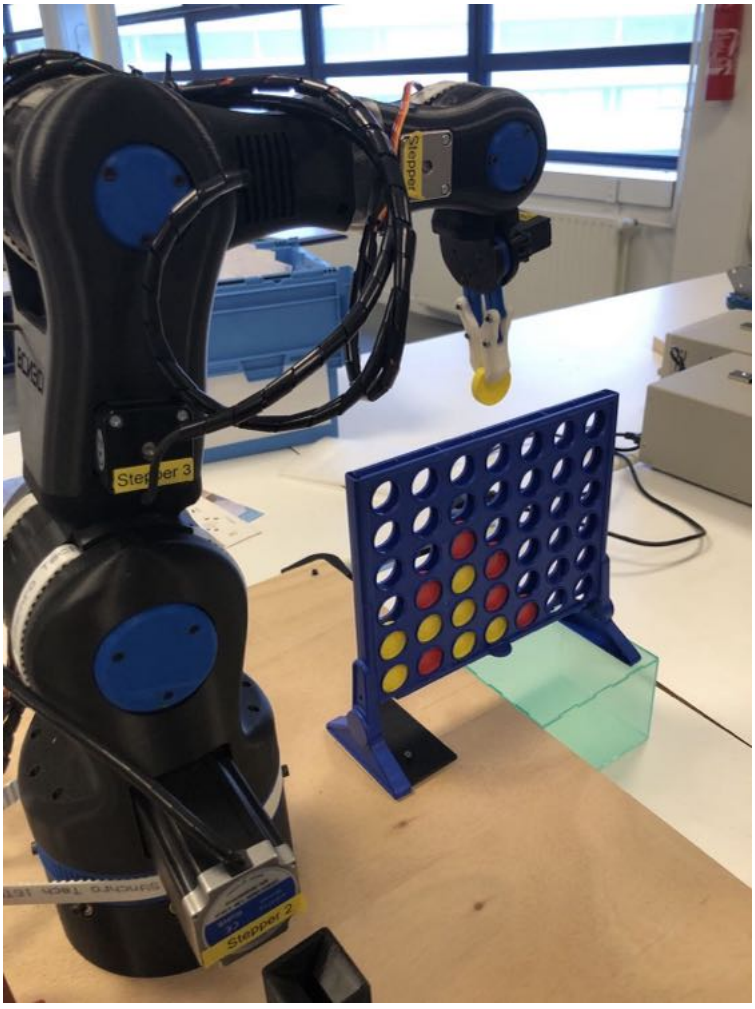
Robots J4

Problématique :

Un bras robotisé peut-il jouer en totale autonomie au puissance 4 ?



Équipe

Notre projet a été mené par une équipe de quatre lycéens aux compétences complémentaires ; Bastien, Kalvyn et Emma. Nous suivons les spécialités mathématiques et physique-chimie, ce qui nous a permis de travailler sur les aspects mécaniques, la modélisation et l'optimisation des mouvements d'un bras robotisé. Gabriel, spécialisé en mathématiques et NSI, a apporté son expertise en programmation et en communication entre les différents systèmes.

En collaborant, nous avons combiné nos connaissances pour concevoir un bras mécanique capable de jouer au Puissance 4 de manière autonome. Ce projet nous a permis d'explorer des domaines variés telles que la robotique, l'intelligence artificielle et la gestion de projet, tout en développant notre capacité à résoudre des problèmes techniques et à travailler en équipe.

La démarche de projet

Ce projet a pour objectif de concevoir un bras robotisé capable de jouer et de gagner au Puissance 4 contre un adversaire humain, démontrant ainsi une maîtrise des domaines de la mécanique, de la programmation et de l'intelligence artificielle. Notre démarche s'inscrit dans une approche scientifique rigoureuse intégrant la modélisation, la fabrication et l'expérimentation. Ce robot utilise des plans open-source modifiés pour répondre à nos besoins spécifiques, tout en combinant des éléments mécaniques, électroniques et informatiques pour aboutir à une machine performante et autonome.

Introduction

Nous avons entrepris ce projet pour explorer les capacités d'un bras robotisé dans un cadre ludique, le jeu de Puissance 4. Notre objectif était de construire une machine capable non seulement de manipuler physiquement les jetons, mais aussi d'analyser la situation du jeu et de prendre des décisions stratégiques.

Table des matières

Équipe....2
Introduction....2
I. Conception et fabrication....4
A. Étude de Faisabilité....4
Exigences Fonctionnelles et Contraintes....4
Contraintes Techniques....4
B. Conception Assistée par Ordinateur (CAO)....6
Logiciels Utilisés....6
Modélisation et Adaptations....7
C. Fabrication....7
Choix des Matériaux....7
D. Distributeur de Jetons....9
Conception....9
Problèmes Rencontrés et Solutions....10
II. Électricité et Électronique....11
A. Câblage....11
B. Alimentation....12
III. Mécanique et calcul....12
A. Usure mécanique 12
B. Calcul théorique....13
C. Précision des mouvements....14
IV. Programmation et algorithme....16
A. Défis rencontrés :....16
B. Solution adoptée :....16
Contrôle des mouvements....17
Intelligence artificielle....17
Communication....20
V. Caméra....21
VII. Perspectives et approfondissements....22
Sources....25

I. Conception et fabrication

A. Étude de Faisabilité

Exigences Fonctionnelles et Contraintes

Pour démarrer notre projet, nous avons défini des critères primordiaux :

- **Taille** : le bras devait mesurer environ 1 mètre pour atteindre et manipuler des jetons dans une grille de Puissance 4. Cette dimension était nécessaire pour une liberté de mouvement suffisante.
- **Précision** : le bras devait être capable de placer un jeton avec exactitude dans la colonne désirée, sans décalage.
- **Articulations** : nous avons déterminé qu'au moins six degrés de liberté (articulations) étaient nécessaires, incluant une pince pour la manipulation.
- **Budget** : les coûts devaient être limités, priorisant des matériaux économiques comme le plastique.
- **Matériaux et outils** : nous avions accès à des imprimantes 3D, mais pas à des outils spécialisés pour travailler le métal ou d'autres matériaux coûteux.

Après plusieurs recherches, nous avons identifié le modèle open-source **BCN3D-Moveo** sur GitHub (lien ici). Ce modèle répondait à nos besoins tout en permettant des modifications. Le choix de ce plan nous a épargné des mois de conception tout en laissant la possibilité d'adaptations spécifiques à notre projet.

Contraintes Techniques

Un défi majeur résidait dans le contrôle des moteurs pour garantir une précision suffisante. De plus, la sélection du matériau pour l'impression 3D devait allier légèreté, durabilité et coût raisonnable. Enfin, il fallait réfléchir à un système simple, mais fiable pour la distribution des jetons.

Afin de condenser ces exigences dans un même document et de mieux s'organiser dans la répartition des tâches, nous les avons réunit dans les diagrammes SYSML *(figure 1 et 2).*

Le **SysML** (Systems Modeling Language) est un outil qui sert à représenter visuellement le fonctionnement et la structure d'un système complexe, comme une voiture, un avion ou un logiciel. Il permet de créer des schémas pour mieux comprendre comment les différentes parties d'un projet interagissent entre elles, repérer d'éventuels problèmes et améliorer la conception avant même de commencer la fabrication. C'est un peu comme un plan détaillé qui aide les équipes à organiser leur travail et à s'assurer que tout fonctionne bien ensemble.

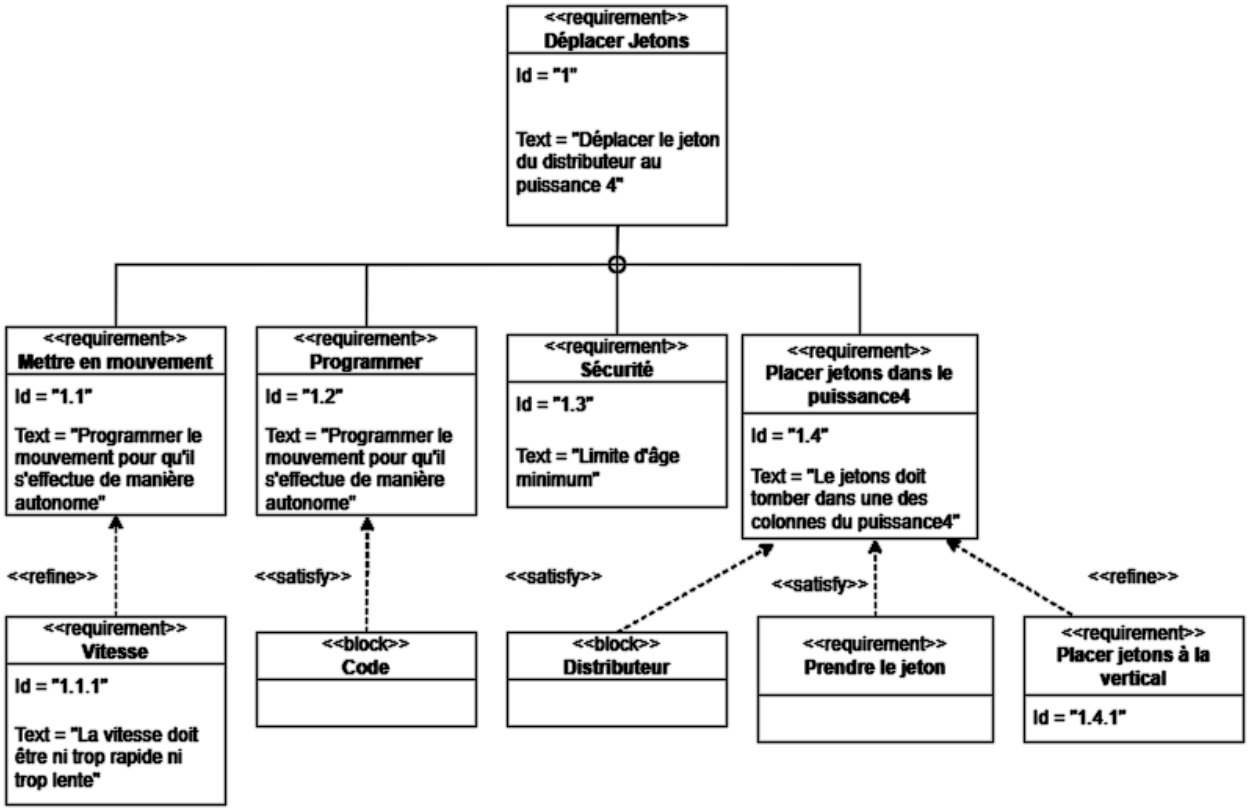


Figure 1: Diagramme d'exigence du bras robot

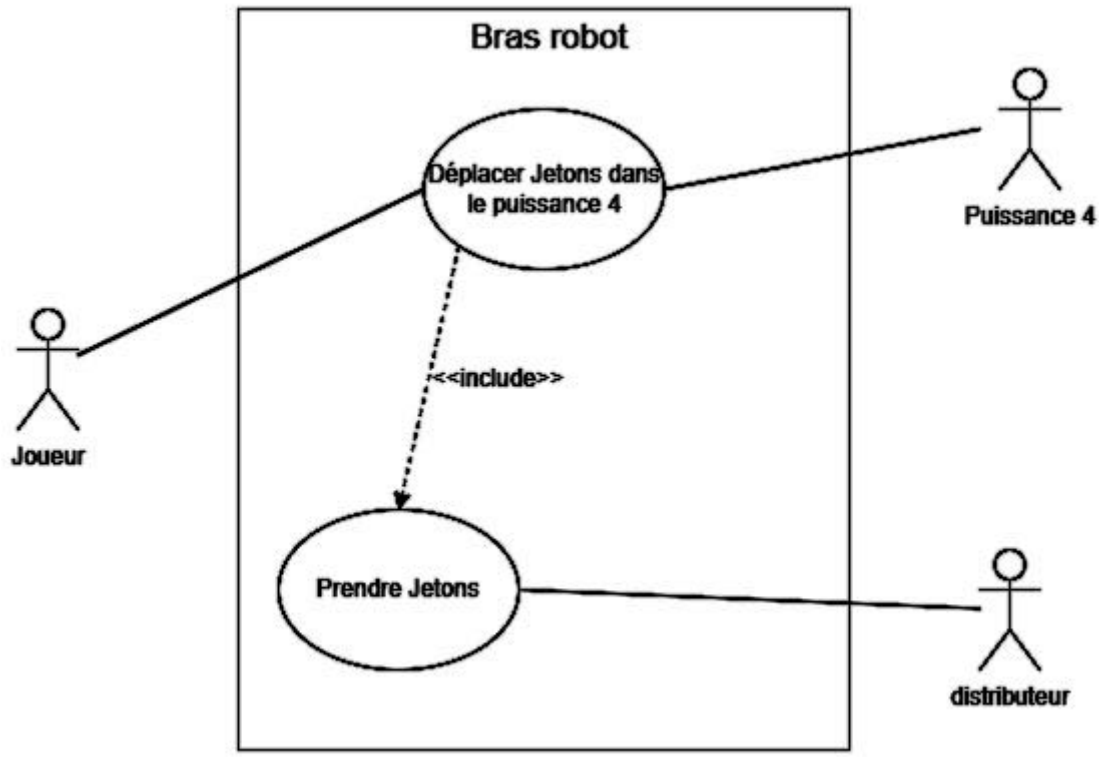


Figure 2: Diagramme des cas d'utilisations du robot

B. Conception Assistée par Ordinateur (CAO)

Logiciels Utilisés

Nous avons utilisé **Onshape** *(voir figure 3, 4 et 5)*, un logiciel de modélisation 3D en ligne, pour personnaliser le modèle BCN3D-Moveo. Ce choix s'est avéré particulièrement pratique pour collaborer à distance et accéder aux fichiers depuis n'importe quel appareil.

Figure 3: Logo du logiciel onshape

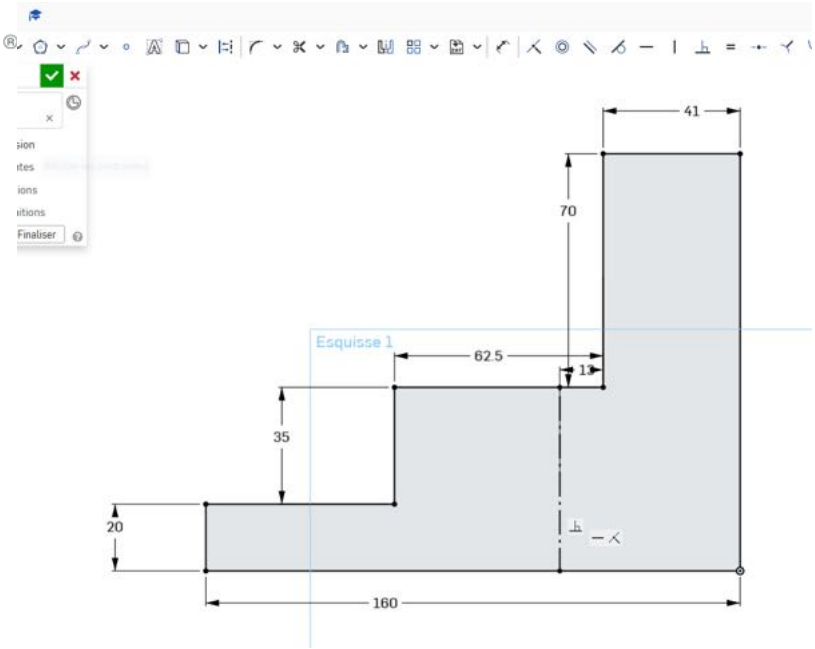


Figure 4: Capture d'écran de la réalisation du distributeur

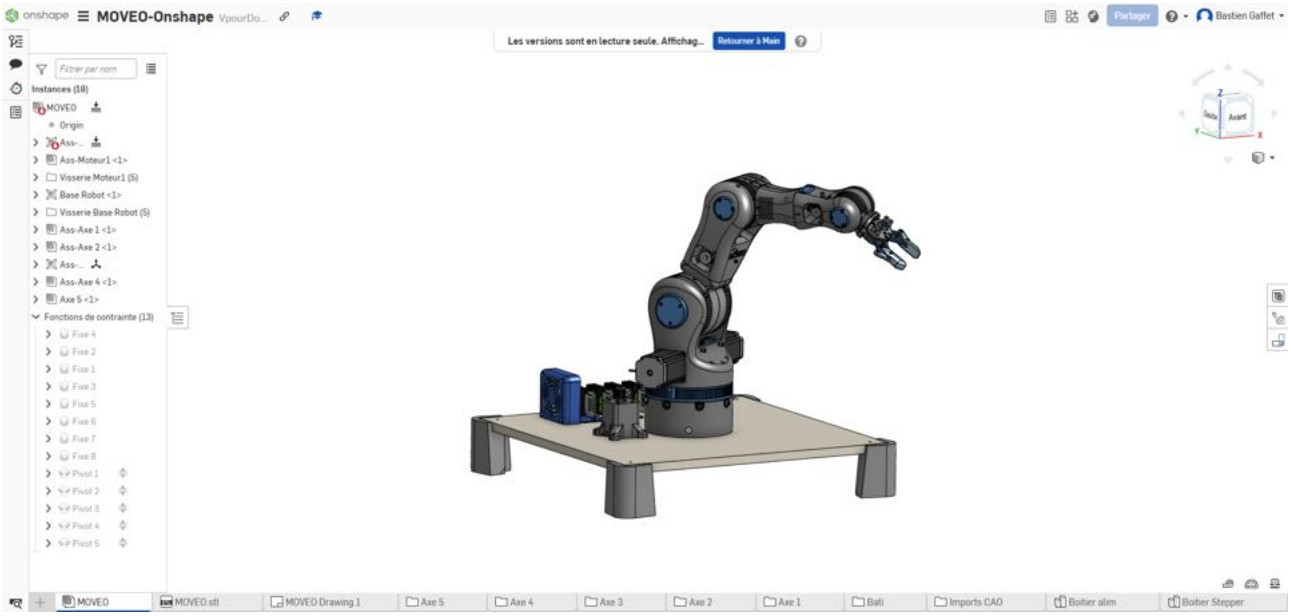


Figure 5: Capture d'écran du modèle du bras robot sur onshape

Modélisation et Adaptations

Des modifications ont été nécessaires pour intégrer une pince capable de saisir des jetons et pour optimiser la stabilité du bras lors de la manipulation des pièces. Ces adaptations ont été modélisées dans Onshape, où nous avons pu visualiser et tester différentes configurations avant impression. La pince *(figure 7)* était lisse ce qui amenait le jeton à glisser ou à se mettre de travers, avec le nouvel embout pour la pince, le jeton est correctement pris à chaque fois.

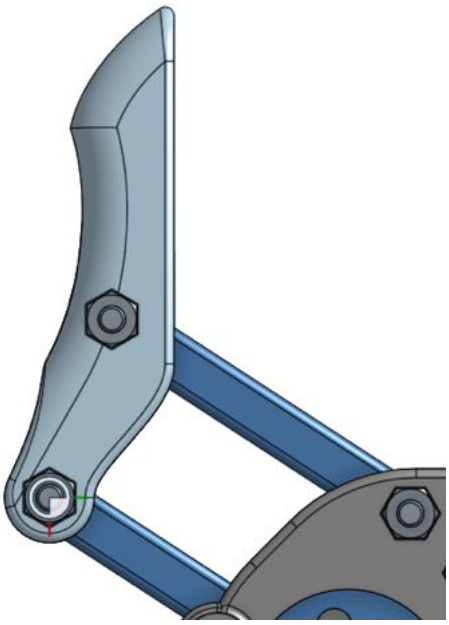


Figure 6: Pince lisse

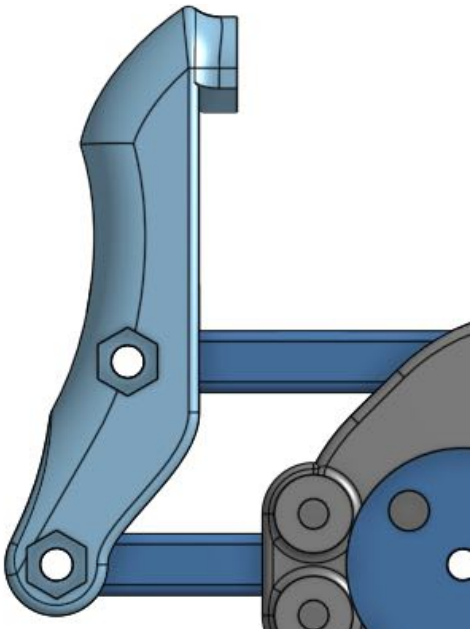


Figure 7: Pince après modification

C. Fabrication

Choix des Matériaux

Nous avons comparé différents plastiques couramment utilisés dans l'impression 3D :

Matériau	Caractéristiques	Applications
ABS	Robuste, résistant à la chaleur et aux impacts	Prototypes fonctionnels
PLA	Facile à imprimer, biodégradable	Prototypes fonctionnels
PETG	Résistant à l'humidité et aux produits chimiques	Composants assemblables
Nylon	Solide, flexible, résistant à l'usure	Prototypes fonctionnels

Nous avons choisi le (**PLA)**, qui présentait le meilleur compromis entre coût, 25€/kg environ, facilité d'impression et rigidité. Ce matériau était également adapté aux imprimantes disponibles dans notre lycée (la Raised3D E2).

L'impression de toutes les pièces a pris environ **100 heures**, réparties sur les deux imprimantes de la salle de Sciences de l'ingénieur. Une attention particulière a été portée aux réglages pour éviter les défauts d'impression susceptibles de compromettre la solidité des pièces. Le remplissage des pièces est de 30 %, ce qui nous fait une masse de PLA de seulement 1,8 kg et donc un coût de 45€ pour l'ensemble du robot. Nous avons donc réussi à allier légèreté et robustesse tout en économisant de la matière.

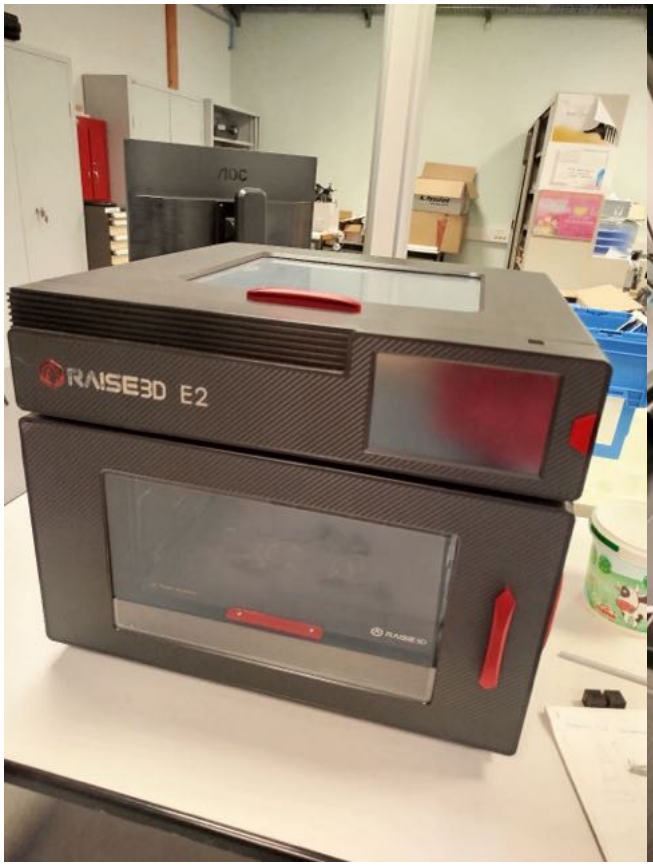


Figure 8: L'imprimante Raised3D E2 qui a servi à imprimer la plupart des pièces.

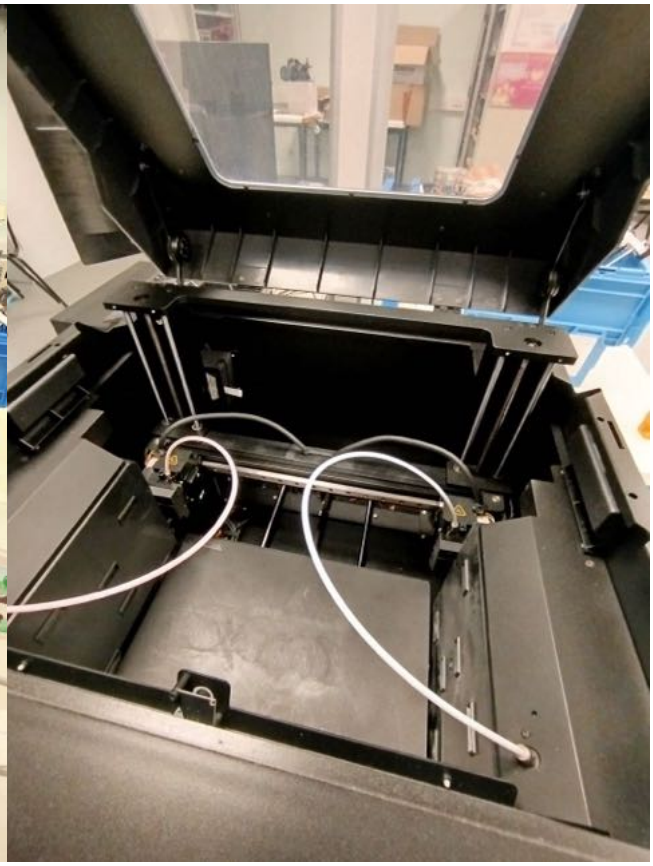


Figure 9: Plateau d'impression de l'imprimante

Moteurs et Électronique

Nous avons sélectionné des moteurs **stepper*** (*moteur pas à pas) pour leur précision et leur capacité à maintenir une position fixe. Ces moteurs fonctionnent par pas, permettant un contrôle détaillé de chaque articulation.

Pour garantir un couple suffisant, nous avons calculé les forces nécessaires en fonction de la masse totale du bras. Ce calcul a été essentiel pour éviter que le bras ne "s'affaisse" sous son propre poids ou perdent des pas lors des mouvements.

Une fois les composants électroniques reçus et l'ensemble des pièces du bras imprimé, nous avons suivi la modélisation et la notice de montage fournie, et assemblé une première version du robot fidèle au plan.

D. Distributeur de Jetons

Conception

Le distributeur de jetons devait répondre aux critères suivants :

1. **Autonomie** : utilisation de la gravité pour éviter tout besoin de motorisation.
2. **Fiabilité** : assurer un flux continu de jetons à la verticale.
3. **Adaptabilité** : être compatible avec la pince du bras robotique.

Après plusieurs itérations, nous avons conçu un distributeur utilisant la gravité simple. Les jetons glissent naturellement le long d'une pente inclinée. Comme pour le bras, nous avons organisé les informations sous formes de diagrammes SysML. *(figure 10 et 11)*

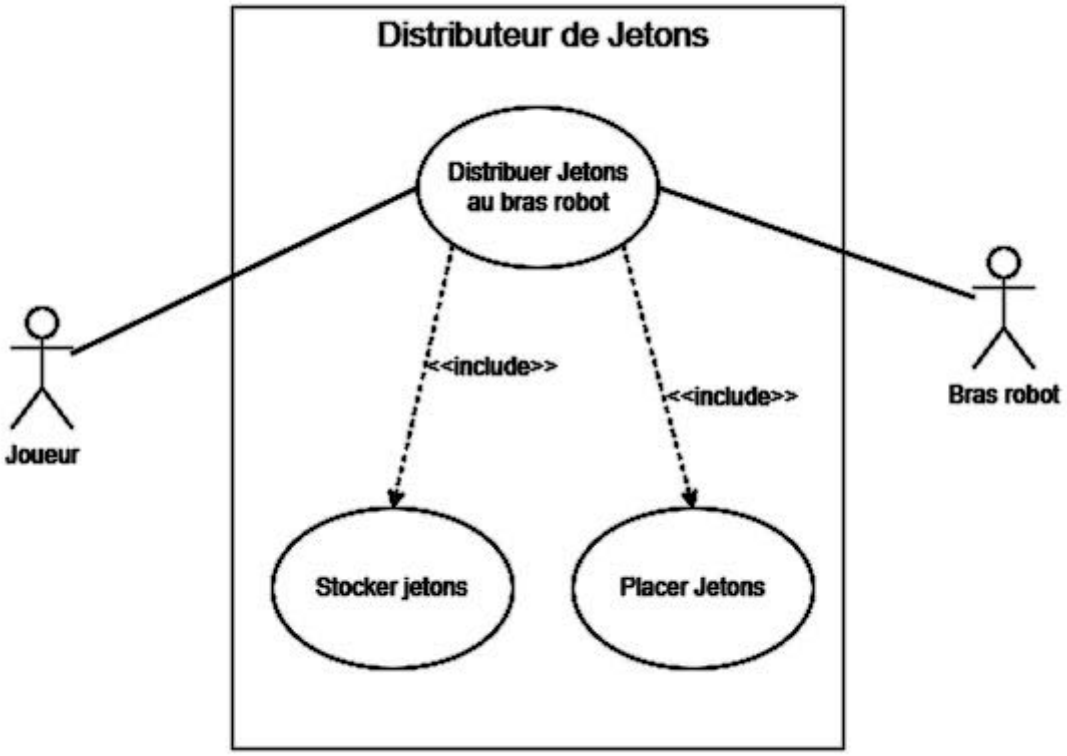


Figure 10: Diagramme des cas d'utilisation du distributeur

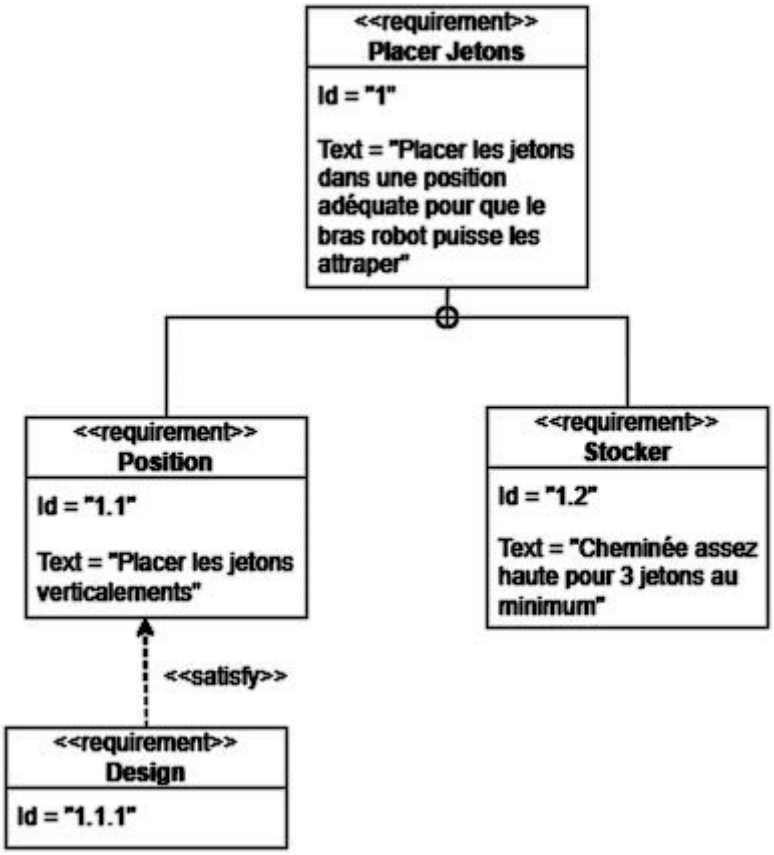


Figure 11: Diagramme d'exigence du distributeur

Problèmes Rencontrés et Solutions

Nous avons imprimé de nombreuses versions différentes du distributeur car certains problèmes n'étaient pas prévisibles sur un simple modèle en trois dimensions. La première version était trop étroite, entraînant des frottements excessifs et un blocage des jetons. Une seconde version a été imprimée, cependant la pente qui permet au jeton de glisser vers l'avant, n'était pas suffisante. Après avoir fait des tests, nous avons conclu qu'une pente de 4° minimum était nécessaire, la version finale inclut donc une pente de 6° qui nous permet d'être d'assurer la chute des jetons., *(Voir figure 12 et 13)*

Un système de fixation dans lequel il est possible de passer des vis afin de le fixer à la planche support à aussi été intégré au distributeur (figure 14). Toutes ces modifications garantissent un flux constant du premier au dernier jeton. Enfin pour des raisons esthétiques, nous avons décidé de garder la même capacité de jeton, c'est à dire 7 jetons sur les 21. Il faut donc charger le distributeur 3 fois en tout, si la partie se poursuit jusqu'au remplissage de la grille du puissance 4.

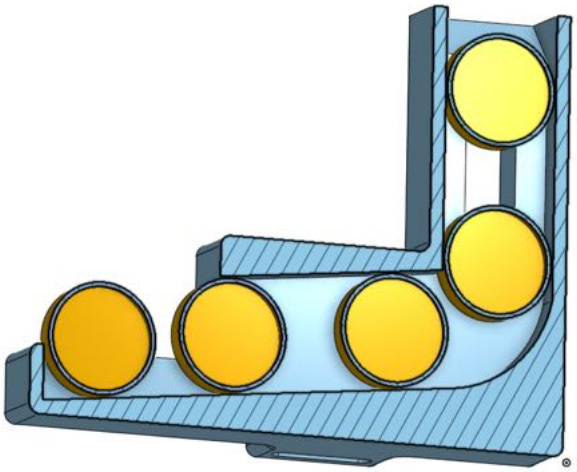


Figure 12: Vue de coupe du distributeur avec une pente de 6°

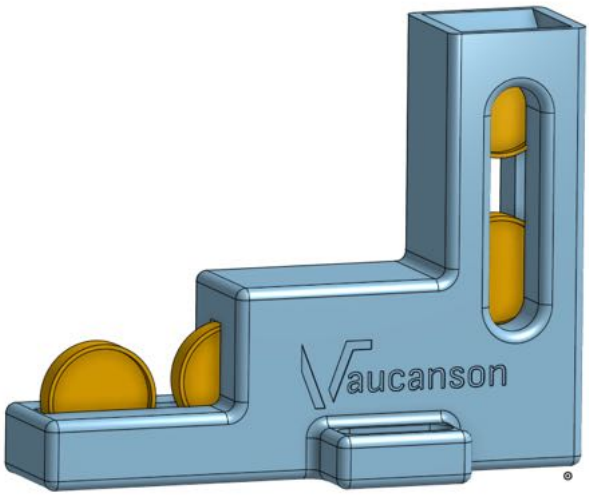


Figure 13: modèle en 3 dimensions de la version finale du distributeur

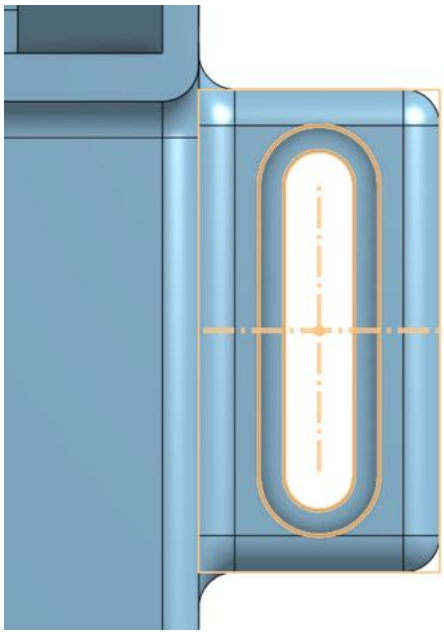


Figure 14: Fixation pour les vis

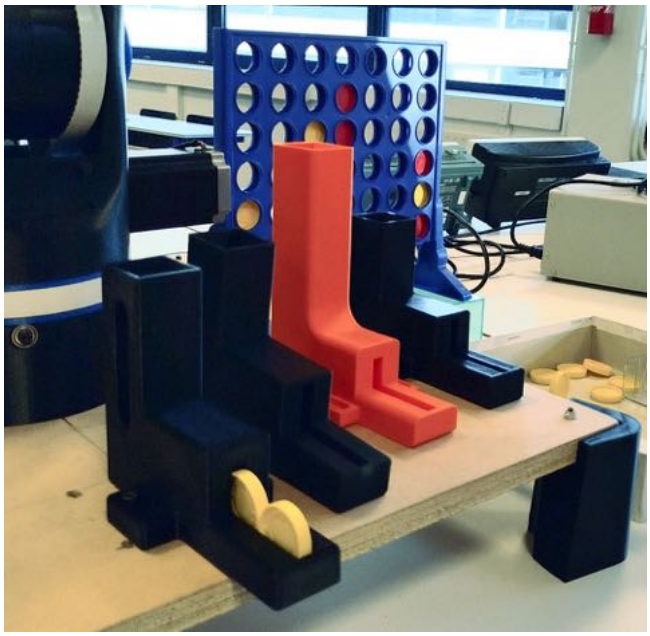


Figure 15: L'ensemble des versions imprimées du distributeur

II. Électricité et Électronique

A. Câblage

Afin de contrôler séparément chaque stepper, nous les avons reliés chacun à un driver séparé nous permettant d'envoyer des ordres ciblés à chacun des steppers. Le modèle que nous avons utilisé est le TB6560 3A Stepper motor driver board WPM333 de chez Whadda. Ces drivers sont

facilement contrôlables avec notre carte arduino mega. Tous les driver sont reliés entre eux par le fil d'alimentation en rouge.

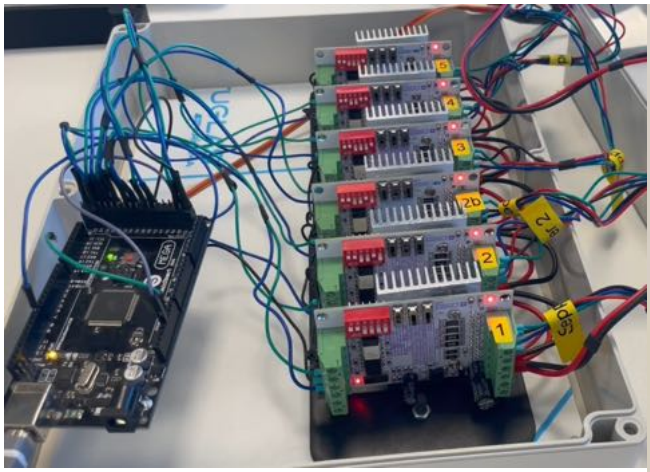


Figure 16: La carte Arduino et les drivers des steppers

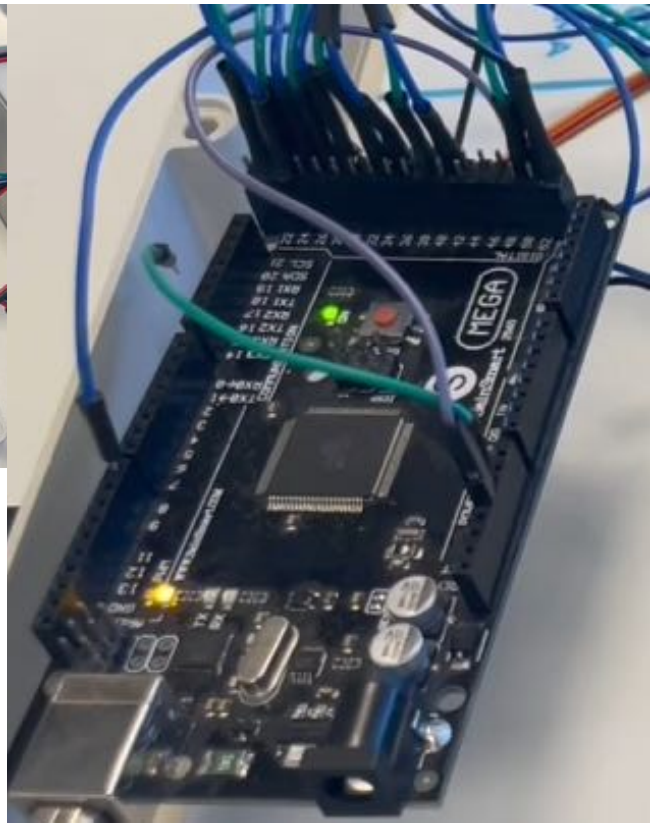


Figure 17: L'arduino mega sous tension

B. Alimentation

Le robot fonctionne grâce à une alimentation 24V continue, branchée sur secteur. Elle convertie donc un courant électrique alternatif avec une tension de 220V en un courant continu sous une tension de 24V. Cette alimentation fournit les steppers et leur driver uniquement. L'alimentation de la carte Arduino Mega se fait via le câble USB type A vers USB type B qui connecte l'ordinateur et la carte arduino ; c'est une alimentation 5V, et pour le servomoteur de simples câbles électriques relient la carte arduino au servomoteur de la pince.

III. Mécanique et calcul

A. Usure mécanique

Quand le bras robot fut enfin assemblé et câblé correctement, nous avons procédé à des tests. Nous avons donc pu identifier de nombreux problèmes dans la conception du robot. Mais ces différents problèmes ne se sont pas tous révélés en même temps, ce qui a ralenti notre capacité à les traiter.

Parmi ces problèmes, une tension trop faible de l'ensemble des courroies faisant le lien entre les steppers et les articulations afin d'augmenter le couple du moteur. Voir l'ensemble des courroies sur la *(figure 18)*. Bien qu'un système de tension était déjà présent dans le design initial, nous avons rajouté une plaque en métal couplée à une vis plus grande pour pouvoir appuyer davantage

sur les courroies en 2a, 2b, 3 et 5 *(figure 21).* Si les courroies sont trop détendues, elles sont susceptibles de sauter des pas ce qui est très problématique. Pour l'articulation 1, un tendeur a été modélisé ; il s'accroche à la planche de bois et maintient la tension de la courroie.

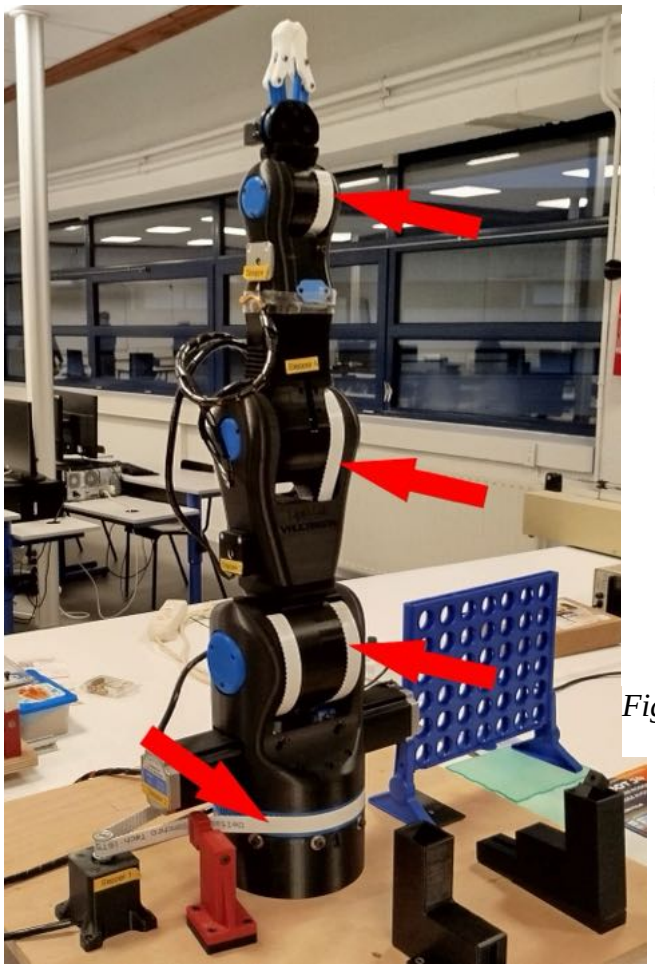


Figure 18: les différentes courroies en blanc et le tendeur en rouge

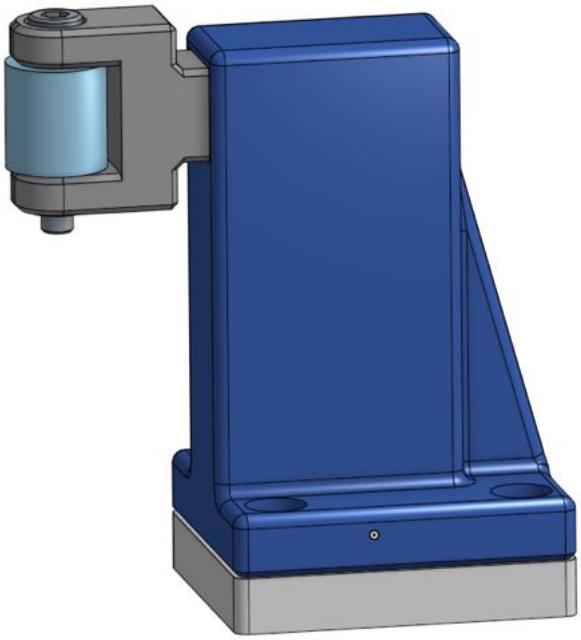


Figure 19: modèle 3D du tendeur pour le stepper 1

B. Calcul théorique

Nous avons tout d'abord, placé à la main le robot hors tension aux différentes positions voulues. Puis une fois la position définie, nous avons remis sous tension le bras pour verrouiller sa position. Ensuite, nous avons relevé les angles de chacune des articulations. À l'aide des formules du rapport réducteur et du rapport entre l'angle et le nombre de pas fournis dans la documentation des steppers. *(exemple documentation figure 22)*.

Z est le nombre de dents du pignon ou de la courroie, r le rapport réducteur, et θ l'angle décrit par le moteur, Les valeurs de $Z_{menée}$ et $Z_{menante}$, ont été définies directement à l'aide du modèle 3D du robot.

$$r = \frac{Z_{menante}}{A_{menée}} \qquad \theta_{moteur} = \frac{\theta_{sortie}}{r}$$

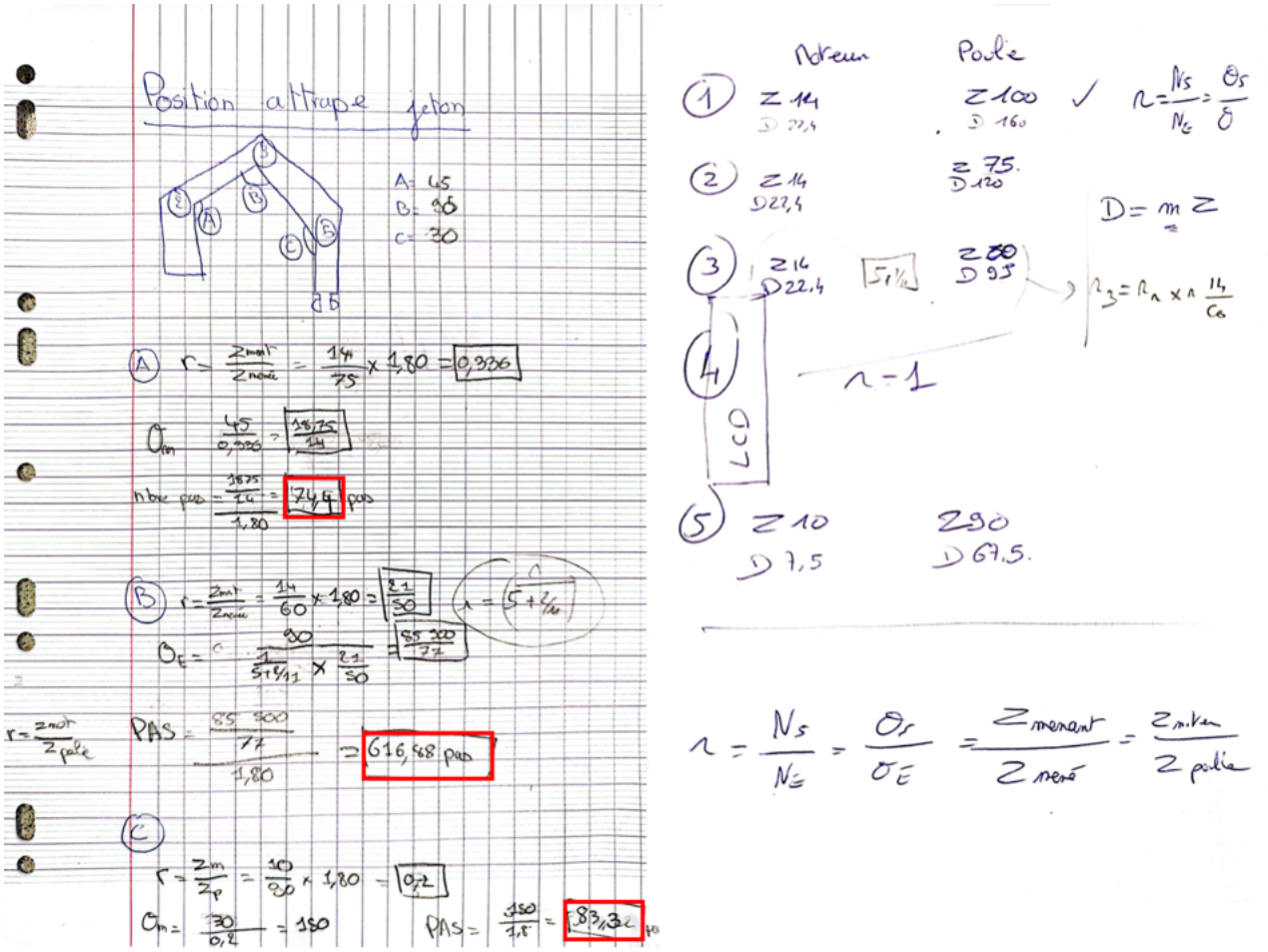


Figure 20: Calculs des positions sur feuilles

Nous avons donc trouvé les positions théoriques voulues en nombre de pas à entrer dans notre programme. Cependant, après exécution du programme et vérification des calculs, l'angle décrit par les articulations n'a pas du tout correspondu aux prévisions. Pour le nom des articulations : se référer à la figure 21. Nous avons trouvé par exemple, pour les articulation 2a et 2b : 74 pas, pour l'articulation 3 : 616 pas et pour la 5 : 83 pas alors qu'en réalité pour atteindre les positions voulues, il faut respectivement 1270, 885 et 680 pas. Nous avons donc rebondit en employant une méthode empirique, qui nous a permis au bout de plusieurs itérations de définir correctement le nombre de pas devant être effectué par chacun des steppers. Ces positions en nombres de pas sont stockées dans notre code arduino et nous permettent de faire facilement des tests.

C. Précision des mouvements

Pourquoi avons-nous besoin d'une précision importante ? Nous avons en effet besoin d'une précision non négligeable, car le jeton du puissance 4 est à peine moins large que la grille <0,5mm, si un quelconque décalage se crée à tout moment le jeton risque de tomber à côté de la colonne choisie. La précision des mouvements est influencée par plusieurs facteurs, dont l'imprécision du pas des moteurs. *(Voir figure 22)*. Celle-ci étant très faible, elle est négligeable. Et le second facteur, le plus important, c'est la résistance des articulations et donc la tension des

courroies. si l'accélération et/ou la vitesse du bras dépasse un certain seuil. Au niveau de l'articulation 2a et 2b. *(Figure 21)*.

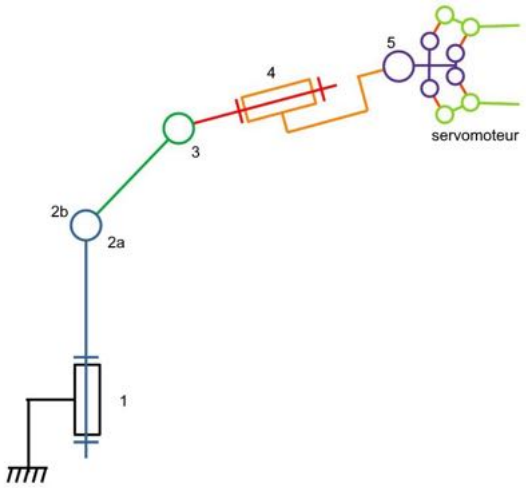


Figure 21: schéma cinématique du bras

SPECIFICATION \ CONNECTION	BIPOLAR
AMPS/PHASE	1.80
RESISTANCE/PHASE(Ohms)@25°C	2.55±10%
INDUCTANCE/PHASE(mH)@1KHz	12.00±20%
HOLDING TORQUE(Nm)[lb-in]	2.40[21.24]
STEP ANGLE(°)	1.80
STEP ACCURACY(NON-ACCUM)	±5.00%
ROTOR INERTIA(g-cm²)	680.00
WEIGHT(Kg)[lb]	—
TEMPERATURE RISE:MAX.80°C（MOTOR STANDSTILL:FOR 2PHASE ENERGIZED）	
AMBIENT TEMPERATURE -10°C~50°C[14°F~122°F]	
INSULATION RESISTANCE 100 Mohm（UNDER NORMAL TEMPERATURE AND HUMIDITY）	
INSULATION CLASS B 130°C[266°F]	
DIELECTRIC STRENGTH 500VAC FOR 1MIN.(BETWEEN THE MOTOR COILS AND THE MOTOR CASE)	
AMBIENT HUMIDITY MAX.85%(NO CONDENSATION)	

Figure 22: notice stepper articulation 2a et 2b

Le moteur n'arrive pas à supporter la charge. Pour que nos mouvements soient précis, nous avons donc réglé la tension et l'intensité apportée au moteur.

On pourrait se demander que se passe-t-il si malgré toutes nos précautions ! Un pas saute et notre mouvement est imprécis ?

Tout d'abord, on pourrait penser à faire un pas en arrière, mais premier problème, on ne peut pas détecter quand un pas est sauté. Cela est dû au fait que la position du robot est définie de façon relative. A chaque mise en tension du robot et lancement du programme, la position zéro de tous les moteurs est définie sur leur position actuelle. En d'autre terme si la position que nous donnons au robot à la mise sous tension du bras est légèrement différente alors toutes les positions seront aussi décalées de cette différence pour l'ensemble de la partie. Pour régler ce problème, nous avons utilisé les repères au niveau de toutes les articulations pour être sûr d'avoir une position de départ fixe. *(Voir figure 23)*. De plus, si c'est la courroie ou le moteur qui saute un pas, l'angle de décalage ne sera pas le même. De plus, une fois le robot sous tension, les moteurs se verrouillent, et il est impossible de les faire bouger à la main en plus du fait que cela forcerait sur l'ensemble du mécanisme. Il est donc très compliqué d'agir une fois le programme en marche.

Ce qui justifie toutes les manipulations sur tous les aspects possibles, aussi bien en programmation qu'en mécanique et sur l'alimentation électrique.

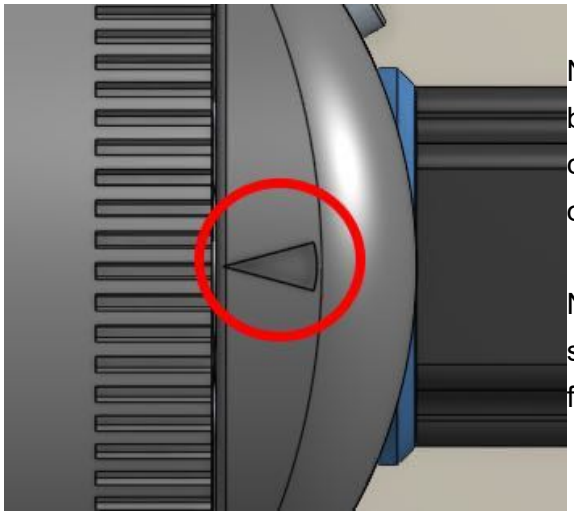


Figure 23: repère sur le modèle onshape

Nous avons recalibré manuellement les positions du bras en testant chaque déplacement pour chaque colonne du Puissance 4, garantissant une précision optimale.

Nous avons remarqué que les steppers 2 et 2b sautaient des pas à cause du poids du robot. Il nous fallait donc augmenter le couple des stepper pour

qu'ils puissent porter la partie supérieure du bras. La formule suivante lie le couple C en Newton-mètre, la puissance en watt et N la vitesse de rotation en tour par minute.

$C = \frac{P}{\frac{2\pi N}{60}}$ Peut être simplifié comme ceci : $C = \frac{30\,P}{\pi N}$ De plus on a : $P = U \times I$

Ce qui nous donne : $C = \frac{30\,U \times I}{\pi N}$

Avec cette relation, on observe que le couple est proportionnel à l'intensité du courant électrique qui parcourt le moteur et qu'il est inversement proportionnel à la vitesse de rotation du moteur. Donc pour augmenter le couple de l'articulation 2a et 2b, nous avons diminué la vitesse de rotation et augmenté la puissance grâce à la documentation des steppers indiquant pour quelles conditions leur couple est maximal *(figure 24)*.

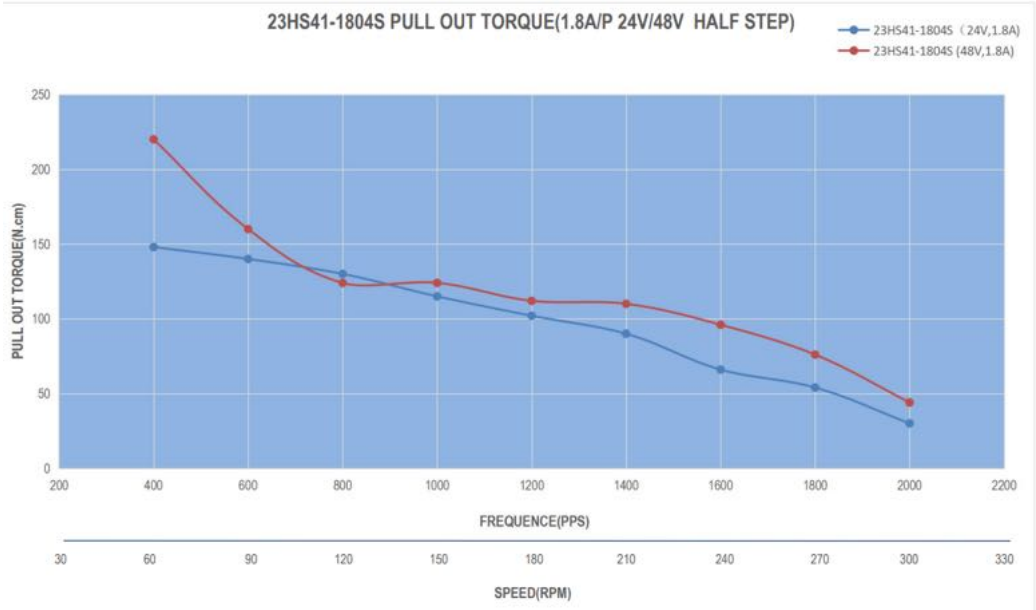


Figure 24: Courbe du couple en fonction de la vitesse et de la fréquence

IV. Programmation et algorithme

A. Défis rencontrés :

1. **Contrôle des mouvements :** comment définir les positions des différentes articulations, et les mouvements simultanés pour des déplacements optimisés.
2. **Intelligence artificielle** : le robot doit être capable de gagner contre un humain avec un temps de calcul raisonnable, mais il doit aussi avoir une difficulté adaptative.
3. **Communication** entre la carte Arduino et le programme Python sur l'ordinateur

B. Solution adoptée :

La programmation du bras a été réalisée pour combiner tous ces aspects :

Contrôle des mouvements

À l'aide de commandes basées sur les positions calibrées, le bras est capable de se déplacer efficacement. Chaque position, après avoir été définie à la main, est enregistrée dans une fonction dans le code arduino afin de pouvoir s'en servir à l'aide d'une simple ligne de code. Il y a donc une position par colonne et une pour la prise du jeton dans le distributeur. Pour exécuter ces mouvements, la position de destination est indiquée pour chaque moteur et donc chaque articulation du bras, ensuite une boucle les fait avancer d'un petit nombre de pas à chaque fois jusqu'à ce que tous les moteurs atteignent la position définie. Ici, la difficulté est la synchronisation des mouvements de plusieurs articulations en même temps. En effet bien que toutes les positions que l'on a définies soient atteignables pour le robot, il y a des obstacles comme le distributeur, la grille du puissance 4 et la caméra qui empêchent le bras de faire le mouvement complet en une fois.

Pour l'ensemble des mouvements, il a donc fallu créer des boucles différentes, mais les plus imbriquées possible pour ne pas perdre trop de temps. En effet, un mouvement désynchronisé vient additionner les temps de déplacement entre eux. Comme nous avons un total de 5 articulations différentes, il faut que les mouvements soient le plus possible en simultané sous peine d'attendre 30 secondes pour chaque jeton joué pour le bras et donc que la partie soit bien trop longue.

La vitesse et l'accélération maximale de chacune des articulations correspondent à un ensemble de valeurs que nous avons définies dans le programme. *(voir figure 25)*.

```
3   #include <Servo.h>
4   const int servPin=8; // PIN servomoteur
5   Servo servo_Pince;
6   const int Maxspeed1=5000.0;     // définition de la vitesse maximale du stepper 1
7   const int Accel1=1000.0;        // définition de l'accélération du stepper 1
8   const int Maxspeed2=2500.0;
9   const int Accel2=500.0;
10  const int Maxspeed3=5000.0;
11  const int Accel3=1000.0;
12  const int Maxspeed4=5000.0;
13  const int Accel4=1000.0;
14  const int Maxspeed5=5000.0;
15  const int Accel5=1000.0;
```

Figure 25: Code arduino définition des vitesses et accélérations

Intelligence artificielle

Nous avons essayé de développer une intelligence artificielle(IA) en deep Q Learning pour permettre au robot de jouer au Puissance 4. Le deep Q Learning est une méthode

```
C:\Windows\System32\cmd.e
Épisode 4305/10000 terminé.
Épisode 4310/10000 terminé.
Épisode 4315/10000 terminé.
Épisode 4320/10000 terminé.
Épisode 4325/10000 terminé.
Épisode 4330/10000 terminé.
Épisode 4335/10000 terminé.
Épisode 4340/10000 terminé.
Épisode 4345/10000 terminé.
Épisode 4350/10000 terminé.
Épisode 4355/10000 terminé.
Épisode 4360/10000 terminé.
Épisode 4365/10000 terminé.
Épisode 4370/10000 terminé.
Épisode 4375/10000 terminé.
Épisode 4380/10000 terminé.
Épisode 4385/10000 terminé.
Épisode 4390/10000 terminé.
Épisode 4395/10000 terminé.
Épisode 4400/10000 terminé.
Sauvegarde effectuée après 4400 épisodes.
```

Figure 26: console windows lors de l'entraînement de l'IA

d'entraînement des IA qui se base sur une répétition de dizaine de milliers de parties contre elle-même avec l'attribution de récompense en fonction de ses actions pour que l'IA crée d'elle-même des stratégies lui octroyant la victoire.

Pour pouvoir créer ses stratégies, elle utilise un réseau de neurones qui s'inspire du fonctionnement du cerveau humain. À chaque action considérée comme positive, des connexions vont se créer entre les différents neurones, c'est ce qui va permettre en théorie à l'intelligence artificielle de prendre une décision instantanée et optimisée à chaque situation qui va lui être présentée lors d'une partie contre un joueur.

L'intelligence artificielle entraînée par Deep Q learning est particulièrement pratique à utiliser une fois le modèle entraîné. En effet son exécution est instantanée, la quantité de données occupée par un modèle comme celui-ci est très faible, mais cela implique donc que tous les calculs ont déjà été faits en avance. C'est sur cette partie que nous n'avons pas réussi à obtenir de résultat satisfaisant avec pas loin de dix-mille parties jouées contre elle-même, l'IA jouait de façon incohérente et ne pouvait absolument pas rivaliser avec un humain.

Nous avons donc opté pour un autre algorithme d'intelligence : le WinMax. Celui-ci attribue à chaque jeton un nombre de points et calcule donc un score pour chaque position du plateau, il simule ensuite le coup suivant, et recalcule le score et ainsi de suite en testant l'intégralité des possibilités. Le nombre de possibilités évolue donc en suivant une progression de forme exponentielle, si bien que pour un arbre de décision de profondeur 8 qui calcule toutes les possibilités de jeu avec huit coups de vision dans le futur, il y a déjà presque six-millions de situations possibles. *(Voir figure 27)*.

Figure 27: Graphique du nombre de possibilité dans l'arbre de décision par rapport à la profondeur de l'algorithme

Nous avons donc utilisé cet algorithme en limitant la profondeur de l'arbre de décision à 6 pour que le temps de calcul soit raisonnable. Il faut que l'ordinateur ne réfléchisse pas plus longtemps qu'un humain. Pour déterminer la profondeur de l'arbre de décision de jeu la plus adaptée, les différentes profondeurs de l'algorithme se sont affrontées, chaque profondeur a affronté 20 fois toutes les autres, pour un total de 100 parties. Nous avons donc écrit un code qui reprend exactement l'algorithme qui joue avec le bras robot et avons simplement supprimé la partie communication

avec le bras pour jouer les parties rapidement uniquement sur l'ordinateur *(figure 29 et 30)*. Pour calculer un score de performance pour les différentes profondeurs, nous avons utilisé cette formule dans un tableur *(figure 31 et 32)* pour trier un grand nombre de données soit environ 7000 valeurs de temps relevées :

Avec les temps de calcul pour un jeton en seconde (s) et i la profondeur de l'algorithme

$$Score\ Perf_i = \alpha \cdot RatioGains_i - \beta \cdot PénalitéTemps_i \quad \text{Avec} \quad \alpha = 60 \quad \beta = 1$$

$$RatioGains_i = \frac{TotalPartiesJouées}{Parties\ gagnées_i} \quad \text{On a :} \quad TotalPartiesJouées = 100$$

`Si` $TempsMax_i > TempsMaxAcceptable$ Avec $TempsMaxAcceptable = 10$

Alors : $PénalitéTemps_i = \frac{TempsMoyen_i}{TempsMaxiObservé_i} + \frac{TempsMaxObservé_i}{TempsMaxAcceptable} \times 5$

Sinon : $PénalitéTemps_i = \frac{TempsMoyen_i}{TempsMaxiObservé_i}$

Cette formule priorise le taux de victoire de l'IA et la pénalise fortement si le temps maximal est supérieur à 10 secondes. On obtient le graphique suivant : *(Figure 28)*.

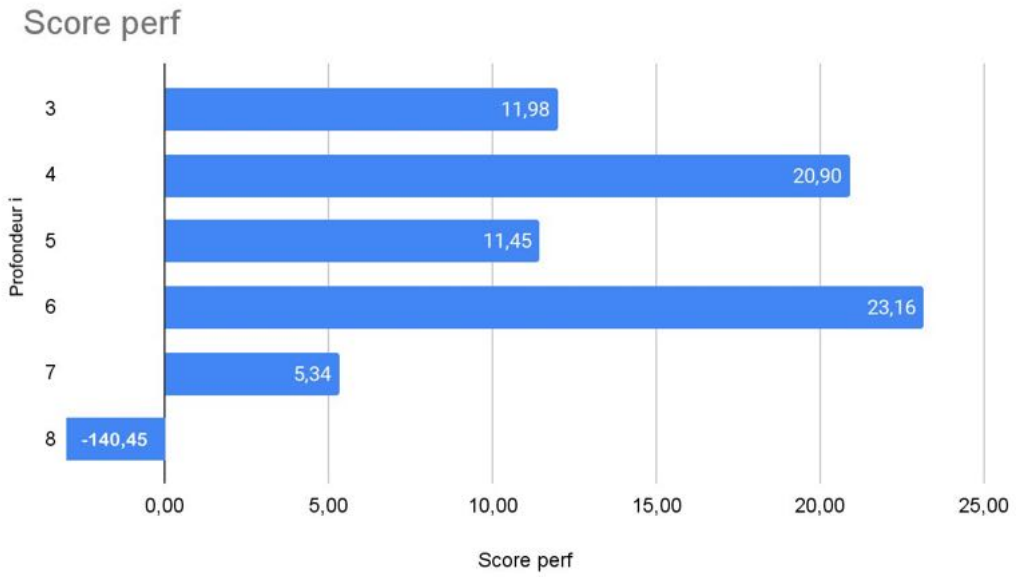


Figure 28: Score de performance en fonction de la profondeur de l'algorithme

Comme on peut le constater, les performances d'une profondeur de 6 sont les plus intéressantes dans une optique de gain de temps pour compenser la vitesse faible du bras mécanique. Nous avons donc choisi une profondeur de 6. On remarque aussi que la profondeur de 5 qui respecte le temps maximal autorisé présente un taux de victoire plus faible que l'algorithme de profondeur 4 qui est de plus plus rapide que le 5, ce qui prouve la nécessité de ce test statistique. Cette incohérence n'était pas détectable sans jouer un nombre important de parties.

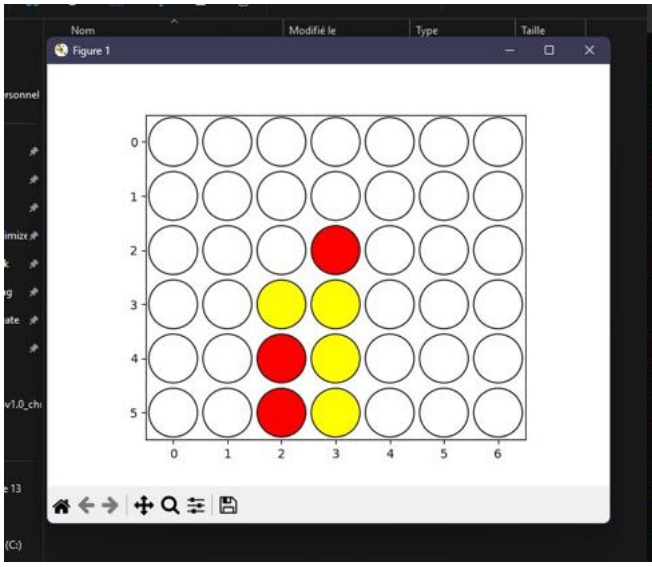


Figure 29: visualisation de la partie virtuelle

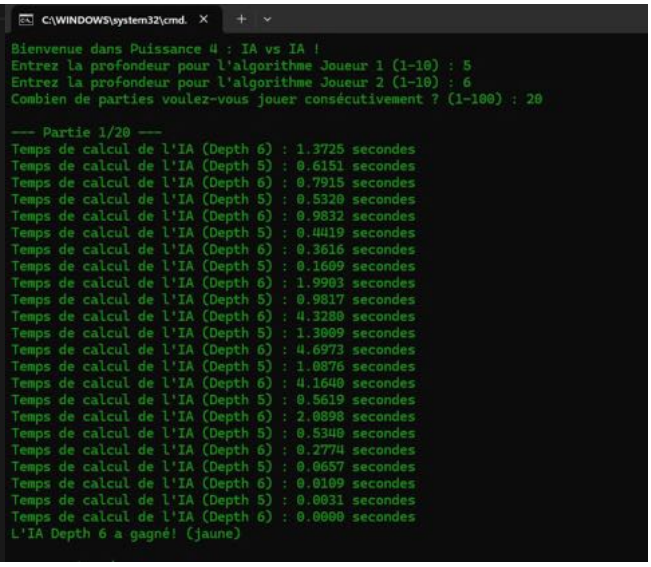


Figure 30: logs de la console pendant le calcul

profondeur		Gagnant	RatioGains	PénalitéTemps	Score perf
3	Victoire	26	0,26	3,62	11,98
	défaite	-23			
4	Victoire	42	0,42	4,30	20,90
	défaite	-39			
5	Victoire	27	0,27	4,75	11,45
	défaite	-24			
6	Victoire	51	0,51	7,44	23,16
	défaite	-48			
7	Victoire	74	0,74	39,06	5,34
	défaite	-71			
8	Victoire	70	0,70	182,45	-140,45
	défaite	-67			

Figure 32: tableur du calcul du Score de Performance

temps de calcul (en s)	profondeur	temps de calcul (en s)	profondeur	temps de calcul (en s)
0,17	5	1,01	6	2,49
0,22		0,98		1,48
0,11		1,61		2,47
0,06		1,66		2,46
0,15		1,48		3,34
0,17		1,37		3,60
0,09		1,86		2,55
0,08		1,68		2,57
0,10		1,44		2,64
0,10		0,36		1,32
0,09		0,12		0,33
0,05		0,13		0,18
0,00		0,06		0,08
0,10		0,03		0,08
0,16		0,01		0,04
0,10		0,01		0,00
0,04		0,00		0,00

Figure 31: Feuille de calcul des valeurs des temps de calcul

Communication

Nous avons choisi de partager le code en un script Python sur un ordinateur et un autre script Arduino exécuté sur la carte qui communiquent entre eux. Le programme Arduino gère tous les mouvements des moteurs du robot avec plusieurs fonctions dédiées à chaque action, tandis que le script Python se concentre sur la logique du jeu et communique les fonctions à exécuter sur la carte Arduino à l'aide du port COM qui est automatiquement détecté à l'aide de la bibliothèque : "serial.tools.list_ports". Dans la boucle Loop de l'arduino, une boucle qui se répète à l'infini tant que l'arduino est sous tension, une fonction permet de détecter si une information est envoyée par le programme python pour ensuite effectuer le mouvement pour placer le jeton dans la bonne colonne de la grille du puissance 4.

```
84  void loop() {
85      String teststr = Serial.readString();  // Lire la chaîne envoyée par le programme extérieur
86
87      if (teststr.length() > 0) {  // Vérifier si la chaîne reçue contient quelque chose
88          teststr.trim();  // Supprimer les espaces ou caractères invisibles
89          int entree = teststr.toInt();  // Convertir la chaîne en entier
90
91          // Vérifier si l'entrée est valide (entre 1 et 7, et pas un zéro dû à un toInt() invalide)
92          if (entree >= 1 && entree <= 7) {
93              prendreJeton(500*8, 1270, 885, 800, 680);
94              poserDansColonne(entree);
95          }
96      }
97  }
```

Figure 33: Boucle loop du code Arduino qui détecte l'envoi d'information en série

```
219     SerialObj.write(entree)
293 arduino_port = detect_arduino()
294 if arduino_port:
295     print(f"Arduino détecté sur {arduino_port}")
296     SerialObj = serial.Serial(arduino_port, baudrate=9600, bytesize=8, parity='N', stopbits=1)
297 else:
298     print("Aucun Arduino détecté.")
```

Figure 34: Code Python ligne219 communication série / ligne293 et + détection de la carte arduino

V. Caméra

Pour rendre notre système autonome, nous avons ajouté une camera qui à l'aide de la bibliothèque python cv2 qui peut traiter jusqu'à 30 images par seconde. Les jetons détectés sont stockés sous la forme d'une liste de la forme : *(figure 35)*

```
[
    [0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0]
]
```

Figure 35: liste qui représente la grille du puissance 4 en python

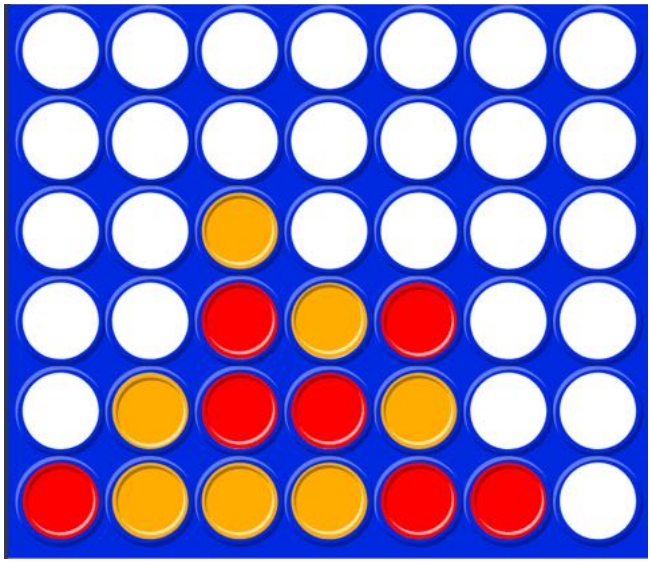


Figure 36: grille d'exemple

```
[
    [0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0],
    [0, 0, 2, 0, 0, 0, 0],
    [0, 0, 1, 2, 1, 0, 0],
    [0, 2, 1, 1, 2, 0, 0],
    [1, 2, 2, 2, 1, 1, 0]
]
```

Figure 37: représentation en liste de la grille de la figure 31

Figure 38: Webcam

Un "0" est une case vide, un "1" représente un jeton de couleur rouge et enfin un "2" représente lui un jeton Jaune. On obtient donc avec l'image *(figure 36)* cette liste *(figure37)* directement exploitable par l'algorithme WinMax.

La caméra utilisée dans ce projet n'a pas besoin d'être particulièrement sophistiquée. Une résolution maximale de 720p est largement suffisante pour détecter les jetons avec précision. Dans un souci de maîtrise des coûts, nous avons opté pour une webcam de récupération, un modèle ancien *(figure 38)* qui répond néanmoins parfaitement aux besoins du système. Afin d'optimiser encore la détection, nous avons ajouté un éclairage permettant de maintenir une luminosité constante et ainsi garantir une reconnaissance fiable des jetons, quelles que soient les conditions d'utilisation.

VII. Perspectives et approfondissements

Nous envisageons plusieurs améliorations pour aller plus loin dans ce projet :

- **Reconnaissance vocale** : Une commande vocale pourrait simplifier l'interaction avec le joueur humain.
- **Miniaturisation :** se débarrasser de la dépendance à un ordinateur encombrant

Nous envisageons d'intégrer un système de reconnaissance vocale afin de rendre notre bras mécanique plus accessible et intuitif. Cette amélioration permettrait au joueur humain de commander le robot simplement en annonçant la colonne dans laquelle il souhaite jouer, supprimant ainsi la nécessité d'une interaction physique avec une interface. Une telle fonctionnalité faciliterait l'utilisation du dispositif pour un plus large public, notamment pour les personnes ayant des difficultés motrices. De plus, cela rendrait l'expérience de jeu plus fluide et naturelle, rapprochant encore davantage l'interaction homme-machine d'un véritable affrontement entre deux joueurs humains.

Nous envisageons d'améliorer la transportabilité de notre installation afin de la rendre plus facile à déplacer et à installer dans différents environnements. Actuellement, notre système repose sur un ordinateur pour traiter les données et contrôler le bras mécanique, ce qui limite sa mobilité. Pour

pallier cette contrainte, nous pourrions remplacer l'ordinateur par un Raspberry Pi, un dispositif compact et performant, capable d'exécuter les algorithmes nécessaires au fonctionnement du bras tout en réduisant l'encombrement de l'installation. Cela offrirait plusieurs avantages : un gain de place significatif, une consommation énergétique réduite et une meilleure adaptabilité du système à divers contextes. Ainsi, cette amélioration renforcerait la flexibilité et la praticité de notre projet, le rendant plus accessible et facile à déployer.

Conclusion

Ce projet incarne une démarche pluridisciplinaire combinant mécanique, programmation et intelligence artificielle. En construisant et adaptant ce bras robotisé, nous avons démontré notre capacité à surmonter des défis techniques complexes et à aboutir à un résultat concret et fonctionnel, tout en rendant l'utilisation d'un tel mécanisme ludique et divertissante.

Durant ce projet, nous avons rencontré plusieurs défis majeurs. Tout d'abord, la précision des mouvements était cruciale pour garantir le placement exact des jetons dans la grille du Puissance 4. Cela nous a poussé à recalibrer les positions des moteurs et à ajuster la tension des courroies pour minimiser les erreurs. Ensuite, nous avons dû adapter notre approche en intelligence artificielle, initialement basée sur le Deep Q Learning, en faveur de l'algorithme WinMax, plus efficace dans notre contexte.

L'un des principaux défis était aussi la synchronisation des mouvements des différentes articulations. Nous avons optimisé les déplacements en réduisant les temps d'attente et en ajustant les paramètres de vitesse et d'accélération des moteurs. De plus, l'intégration de la communication entre la carte Arduino et le script Python a nécessité une gestion rigoureuse du protocole de transmission des commandes.

En termes de compétences acquises, ce projet nous a permis d'approfondir nos connaissances en conception mécanique, en programmation embarquée et en intelligence artificielle. Nous avons également amélioré notre capacité à travailler en équipe et à mener une démarche expérimentale rigoureuse pour résoudre les problèmes rencontrés.

Enfin, ce projet ouvre la voie à de nombreuses perspectives d'amélioration. L'intégration d'une reconnaissance vocale permettrait de rendre l'interaction avec le robot plus intuitive et accessible, notamment pour les personnes en situation de handicap. De même, l'utilisation d'un Raspberry Pi pourrait renforcer la mobilité du système en supprimant la dépendance à un ordinateur externe.

Au-delà de l'aspect technique, ce projet illustre l'importance de la persévérance et de l'adaptabilité face aux obstacles techniques. Il reflète aussi le potentiel qu'offre l'automatisation et l'intelligence artificielle dans des applications interactives et éducatives. En conclusion, nous avons non seulement réussi à concevoir un système fonctionnel, mais nous avons également développé des compétences essentielles qui nous seront précieuses pour de futurs projets technologiques.

Sources

Sources liées à la mécanique et à l’électronique du bras

BCN3D Technologies. *Moveo: A Fully Open-Source 3D Printable Robotic Arm*. GitHub. Disponible sur : https://github.com/BCN3D/BCN3D-Moveo/blob/master/LICENSE

Stepperonline. *site internet et documentation.* Disponible sur : https://www.omc-stepperonline.com/fr

---

Sources liées à la programmation et l’IA

Python Software Foundation. *PySerial Documentation*. Disponible sur : https://pyserial.readthedocs.io/en/latest/

Arduino. *Documentation officielle Arduino*. Disponible sur : https://docs.arduino.cc/

Sources liées à la vision par ordinateur

Bradski, G. (2000). *The OpenCV Library*. Dr. Dobb's Journal of Software Tools. Disponible sur : https://opencv.org/
