**Puissance 4 Mécanique**

Projet IETI 2020-2021

****

Thomas ESPINASSE & Bora SURMELI & Georges LECONTE & Paul MATHIEU

**Introduction :**

Nous avons créé le puissance 4 mécanique afin d’exploiter toutes les possibilités de la table 2D (déplacement, bouger un stylo, capteurs de bout de course …). Ce sujet nous a aussi permis aussi de hiérarchiser facilement les différentes expectatives du cahier des charges et de donner la priorité aux fonctionnalités essentielles tout en nous laissant la possibilité d’ajouter des fonctionnalités supplémentaires. Le projet a été divisé en trois grands axes : l’élaboration d’une intelligence artificielle, le développement d’un logiciel de traitement d’image et le contrôle de la table mécanique. On a pu ainsi travailler en parallèle sans se gêner.

1. **Intelligence et façon de jouer de l’IA :**

**Présentation :**

On a entraîné l’IA avec une technique de “machine learning”, cette technique s’est avérée au final un peu faible du fait du nombre important de possibilités importantes a chaque configuration du plateau de jeu,rendant le calcul de toutes les possibilités possibles trop long et important pour notre puissance de calcul disponible. On est partis sur une base ou chacune des 7 colonnes possibles possède 100 points de chances de gagner pour chacune des configurations possibles. Malgré que c’était peut être pas la meilleure façon de programmer l’IA on remarque qu’elle acquiert néanmoins quelques réflexes, par exemple lorsqu’elle commence à jouer ses possibilités de jouer dans la colonne centrale sont beaucoup plus élevées que dans le autres colonnes, ce qui est le meilleur coup à jouer dans cette configuration.

**Entrainement :**

Pour commencer l’IA joue au hasard et son contre aussi.Chaque partie joue avec l’IA (contre elle même) est enregistrée car si elle gagne, elle remonte le cours de la partie et pour chaque situation qu’elle a rencontrée elle ajoute +1 dans la colonne dans laquelle elle a joué et la donc fait gagner. (Voir schéma de gameplay de l’IA). On utilise cette technique d'entraînement afin d’éviter que l’IA s’habitue à jouer d’une certaine façon et qu' elle soit plus flexible et pas bloqué sur un seul gameplay style. Notre IA s’est entraînée de cette façon environ 30 heures.

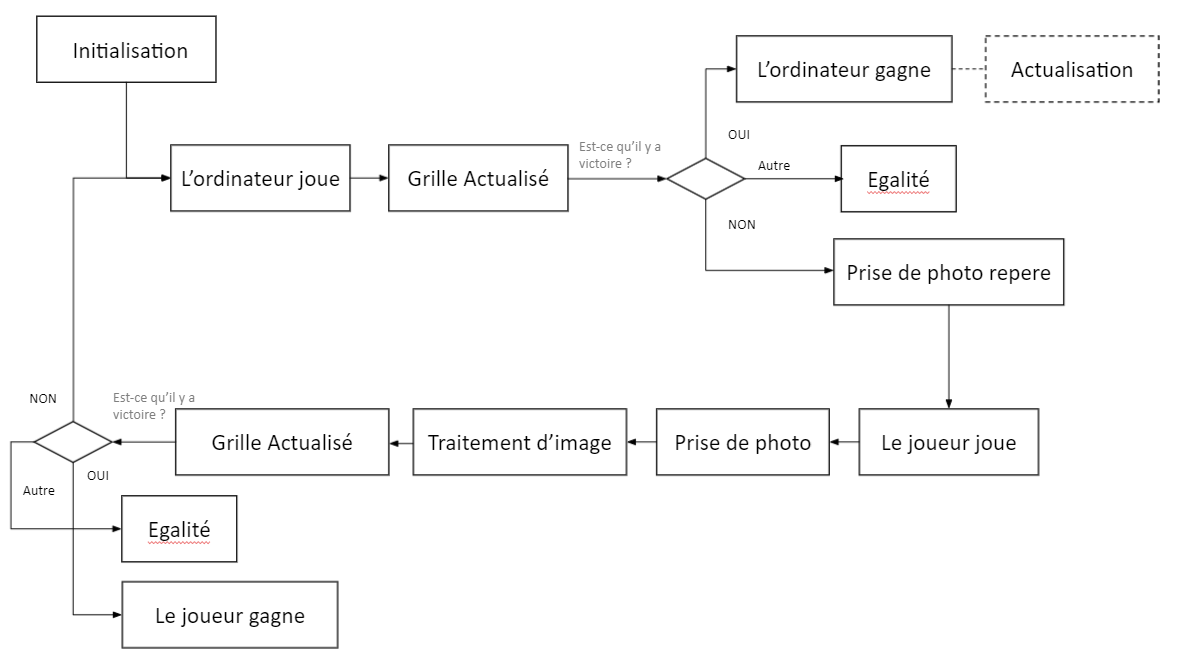
Après un certain de temps avec cet entraînement on décide de passer à un entraînement dans lequel l’IA joue de façon probabiliste en fonction de sa base de données, c'est-à-dire qu’elle joue selon une colonne en fonction de la probabilité que cette colonne a de la rapprocher de la victoire (mais le “joueur” continue de jouer au hasard).Notre IA s’est entraînée de cette façon environ 15 heures.

Un autre des aspects qui a rendu l'entraînement de l’IA plus complexe que prévu était que le temps pour que l’IA joue une partie était variable. On a pu remarquer que ça dépendait de si l’ordinateur était connecté à internet, de ce qui tournait en arrière plan …. On a même remarqué un ralentissement au fur et à mesure du temps, c’est peut être dû au grandissement de la base de données ce qui rendait son actualisation de plus en plus lente.

On remarque souvent que l’IA a du mal à faire de choses simples comme contrer l’adversaire lorsqu’il a aligné 3 fiches ou de prendre la victoire lorsqu’elle a déjà 3 fiches alignées et qu’elle peut. Le programme d'entraînement s'appelle “IAPuissance\_4”

**Au moment de jouer :**

Au moment d' une vraie partie contre l’IA, cette dernière continue à fonctionner de façon probabiliste comme expliquée ci- dessous. C'est-à-dire qu'elle regarde l’état de la grille de jeu puis elle va chercher dans sa base de données les données disponibles sur cette configuration. Avec ces données elle choisit une des colonnes de façon pondérée en fonction du nombre relatif de points de la colonne. Cette méthode bien que n’étant pas la plus performante pour obtenir la victoire a l’avantage de laisser à l'IA explorer de nouvelles possibilités et de ne pas enfermer l’IA dans un seul style de jeu. Le programme qui permet cela s'appelle “Programme\_Principal”.

****

*Schéma du fonctionnement du Programme Principal de l’IA*

**Liaison Caméra :**

La liaison entre la caméra et son logiciel de traitement d’images est facilitée par le fait que tout a été codé sous Matlab pour la caméra. Il nous a donc suffit de mettre les deux programmes (celui de traitement d’image et celui de jouer de l’IA) l’un à la suite de l’autre dans une boucle qui gère le déroulement de la partie. La partie traitement d’image renvoie à chaque fois la ligne et la colonne ou le joueur a joué et l’IA actualise alors sa grille pour répondre de façon adéquate.

**Liaison Table :**

La liaison entre l’IA et la table est une liaison série qui a été implémentée du côté table 2D mais on a pas eu le temps de l'implémenter du côté IA. Donc on a pas complètement réussi à boucler la boucle, mais chaque composant fonctionne individuellement.

**Possibilitées d’amélioration :**

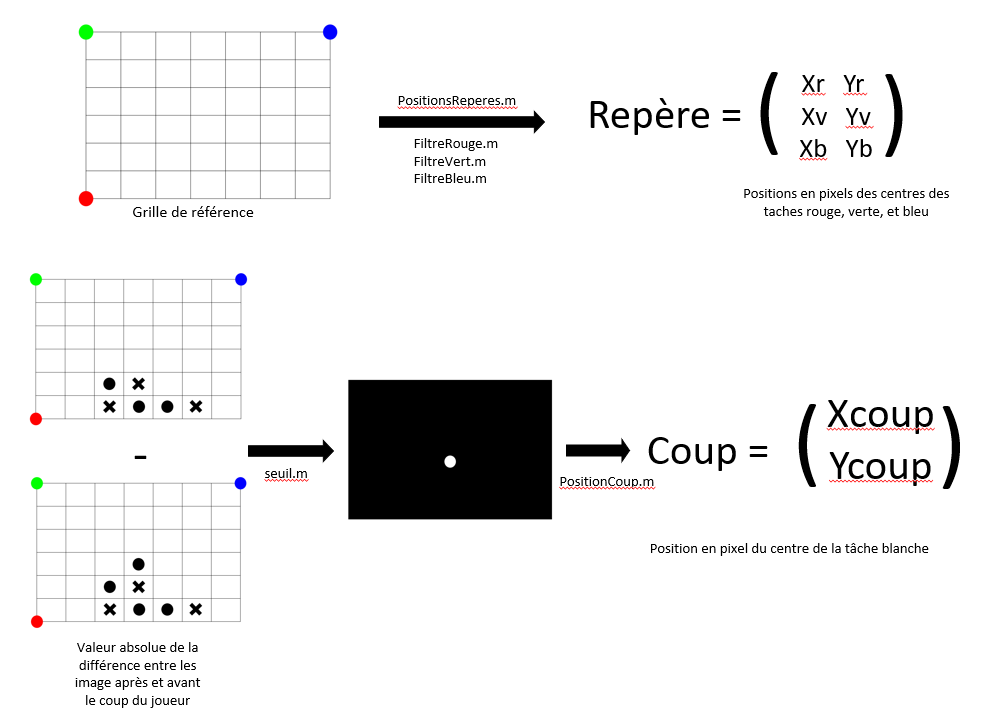
* Il faudrait laisser à l'IA davantage de temps pour s'entraîner
* Pour accroître la difficulté il faudrait que l’IA choisisse obligatoirement de jouer dans la colonne qui a le plus de possibilités de lui apporter la victoire.
* Une autre solution serait d’utiliser le fait que le puissance 4 est un problème résolu, on pourrait donc donner directement les coups a jouer sans avoir à entraîner l’IA (ce qui est très énergivore).
* Obliger l’IA à contrer et à prendre la victoire lorsqu’elle peut.

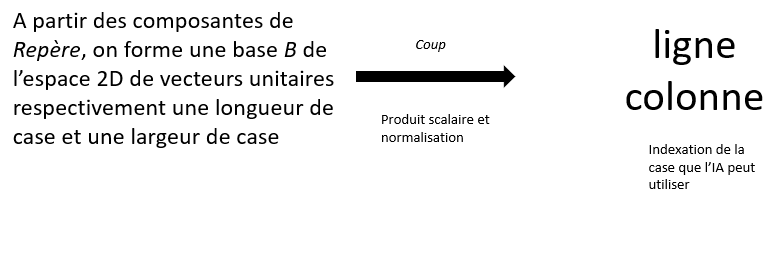
1. **Utilisation de la webcam :**

**Problématique matérielle :**

Pour la partie visant à détecter les positions jouées par l’humain, une solution optique a été retenue. Nous avons choisi de relier une WebCam à Matlab, ce langage permettant une gestion très pratique des périphériques externes, ainsi que des images.

Voici un schéma représentant le travaille effectué sur l’image :





*Diagramme du fonctionnement de l’algorithme de traitement d’image*

**Prise de repère et création d’une base du plan :**

La “grille de référence“ mentionnée est une feuille A4 imprimée et placée au centre de la table de jeu, les coups de l’IA étant tracés par cette même table. Cette grille nous permet, grâce aux pastilles des 3 couleurs primaires utilisées par la caméra, de déterminer un repère. En effet la première partie du programme défini les vecteurs reliant la pastille bleue à la pastille verte, et la pastille verte à la pastille rouge. Ces vecteurs sont alors respectivement divisés par 7 et 6 pour obtenir une base *B* de l’espace 2D de vecteurs unitaires correspondant à la largeur et la longueur d’une case. Les positions repérées par la suite en pixels seront ramenées au vecteur les reliant à la pastille verte, origine de notre repère, et projetées à l’aide d’un produit scalaire dans la base *B*. (on prendra garde d’ajouter une demi case horizontalement et verticalement pour bien faire correspondre les centres des cases jouées, et pas les intersections du maillage).

**Détection d’un coup :**

Il reste à détecter la position jouée par l’humain, car l’ordinateur sait ou il joue sans avoir besoin de vérifier à l’aide de la caméra. Pour cela, on effectue deux prises de vue, avant et après le coup du joueur. La valeur absolue de la différence entre ces deux matrices est ramenée à une image en noir et blanc par la fonction *rgb2grey*. On applique alors à cette image en niveau de gris la fonction *seuil*, qui ne garde en blanc que les pixels de grande valeur (superieur à *s,* argument de *seuil*, et correspondant donc à de gros écarts entre les deux images), tout le reste est en noir. Le barycentre des pixels de l’image résultante est alors pris comme centre de la tâche tracée par le joueur. La méthode précédemment détaillée, ainsi que les arrondis par la fonction *round* donnent un numéro de ligne et de colonne, interprétable par l’algorithme de jeu présenté dans la partie précédente.

Cette méthode consistant à passer la différence en niveaux de gris et à seuiller le résultat a été mise en oeuvre car nous avons constaté expérimentalement que sans cela, les variations d’éclairement du champ de la caméra dues à nos propres ombres, jamais exactement au même endroit, prenaient trop d’ampleur dans le calcul du barycentre des variations.

**Possibilitées d’amélioration :**

La méthode utilisée pour définir la base *B* grâce à la feuille imprimée avec les trois pastilles de couleur pose pour l’instant un problème. Le filtre utilisé pour isoler la composante verte ou rouge ou bleu de l’image pour detecter le point de couleur est très sensible aux conditions d’éclairage. De plus un simple barycentre n’aurai pas donné grand chose car l’ensemble de la feuille est blanc et donc contient des valeurs importzntes dans chaque composante. Ainsi les fonctions *FiltreRouge.m FiltreVert.m*  et  *FiltreBleu.m*, doivent être étalonnées au début de chaque partie, en vérifiant manuellement sur une image de la grille prise par la webCam, les valeurs effectives des différentes composantes de chaque pastille de couleur, pour “forcer” le programme à ne considerer comme “vert” (ou rouge ou bleu) que les pixels s’approchant fortement du vert manuellement identifié comme celui du repère.

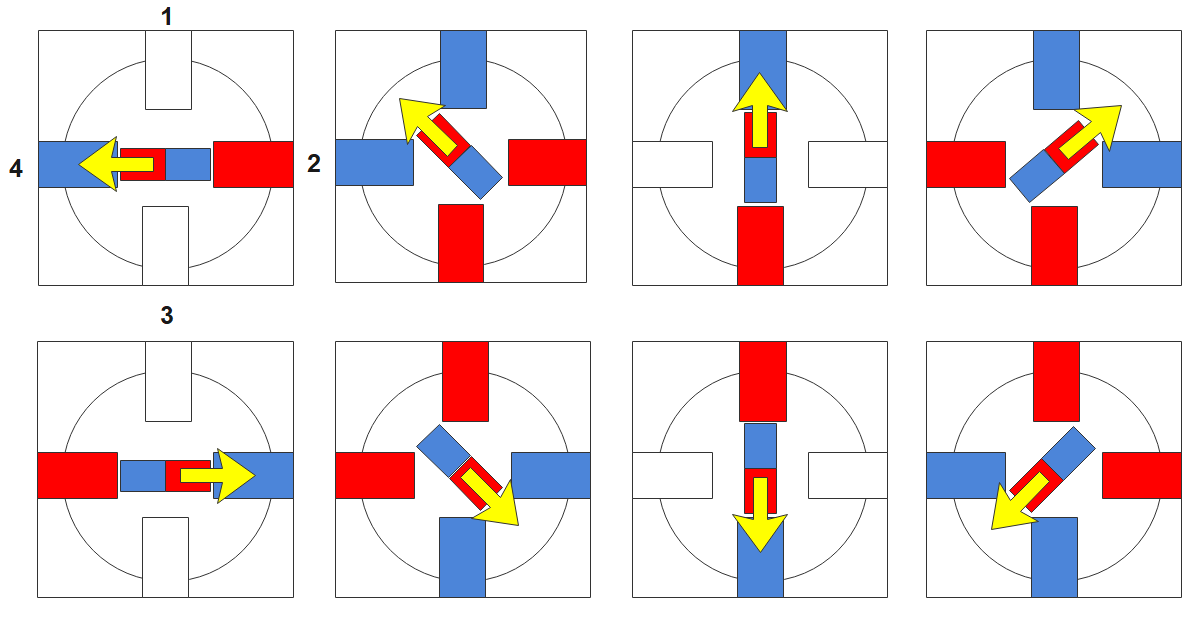
1. **Contrôle de la table 2D :**

**Présentation des composants utilisés:**

La plateforme de la table 2D est en liaison glissière avec des guides le long des axes de la table. La translation le long de ces axes est assurée via un système pignon-courroie commandé par des moteurs pas à pas.

Les moteurs pas à pas ont une précision supérieure aux moteurs à courant continu et sont donc bien adaptés pour des projets qui nécessitent de commander des moteurs avec une certaine finesse. C’est bien évidemment notre cas puisque le choix du moteur a une incidence directe sur la précision du mouvement de translation de la plateforme. Le principe du moteur pas à pas est assez simple. Le stator est composé d’un certain nombre de paires de bobines placées en regard. Le rotor est lui composé d’aimants permanents. En changeant l’alimentation de chacune des bobines dans un certain ordre, on change le champ magnétique et le stator tourne d’un pas afin que les aimants permanents s’alignent avec le nouveau champ. Le pas du moteur est donc défini par le nombre de paires de bobines sur le stator. Les bobines sont alimentées en tout ou rien.

Pour mieux comprendre le principe de fonctionnement, on prend un cas simplifié, où le stator est composé de deux paires de bobines et le rotor un pôle nord et un pôle sud. En rouge sont les pôles sud et en bleu sont les pôles nord, la flèche jaune représente le champ total resultant.



*Schéma de la rotation d’un moteur pas à pas*

Pour effectuer un tour complet, on doit donc commander les bobines dans un ordre particulier. La table logique correspondante est donnée ci-dessous :

|  |  |  |  |
| --- | --- | --- | --- |
| Etat bobine 1 | Etat bobine 2 | Etat bobine 3 | Etat bobine 4 |
| 0 | 1 | 0 | -1 |
| -1 | 1 | 1 | -1 |
| -1 | 0 | 1 | 0 |
| -1 | -1 | 1 | 1 |
| 0 | -1 | 0 | 1 |
| 1 | -1 | -1 | 1 |
| 1 | 0 | -1 | 0 |
| 1 | 1 | -1 | -1 |

*Table logique correspondant à l’état des bobines lors de la rotation du moteur*

Les états des bobines 1 et 2 sont les complémentaires logiques des états des bobines 3 et 4. Il suffit donc de commander 2 entrées pour faire tourner le moteur comme on le souhaite.

Le moteur dont on dispose a un nombre de bobines plus important, ce qui permet d’avoir un pas angulaire plus petit mais ce qui nécessite de commander plus d’entrées, en l'occurrence 4. Coder un programme déterminant la nouvelle combinaison pour tourner le moteur d’un pas dans un sens ou dans l’autre en fonction de la combinaison précédente est extrêmement fastidieux. On utilise donc un circuit imprimé dont c’est le rôle de la carte L297. Le composant doit être alimenté en 5V et relié à la masse. Il comporte deux entrées digitales : l’état de la première définit le sens de rotation du moteur et un changement d’état sur la seconde entrée correspond à l’instruction tourner d’un pas. Le composant est constitué d’un ensemble de portes logiques qui permettent de déterminer l’état des sorties qui correspondent aux valeurs de la table logique pour effectuer la rotation demandée en fonction de la combinaison précédente.

On souhaite donc connecter ces sorties aux entrées du moteur. La puissance de la carte n’est cependant pas suffisante pour contrôler directement les moteurs à partir des sorties de la cartes L297 : il faut donc utiliser un circuit de puissance distinct du circuit de commande. On utilise pour cela un relais qui permet d’amplifier les signaux de commandes issus de la carte L297 afin qu’ils soient conformes à l’ampérage et au voltage attendu par le moteur.

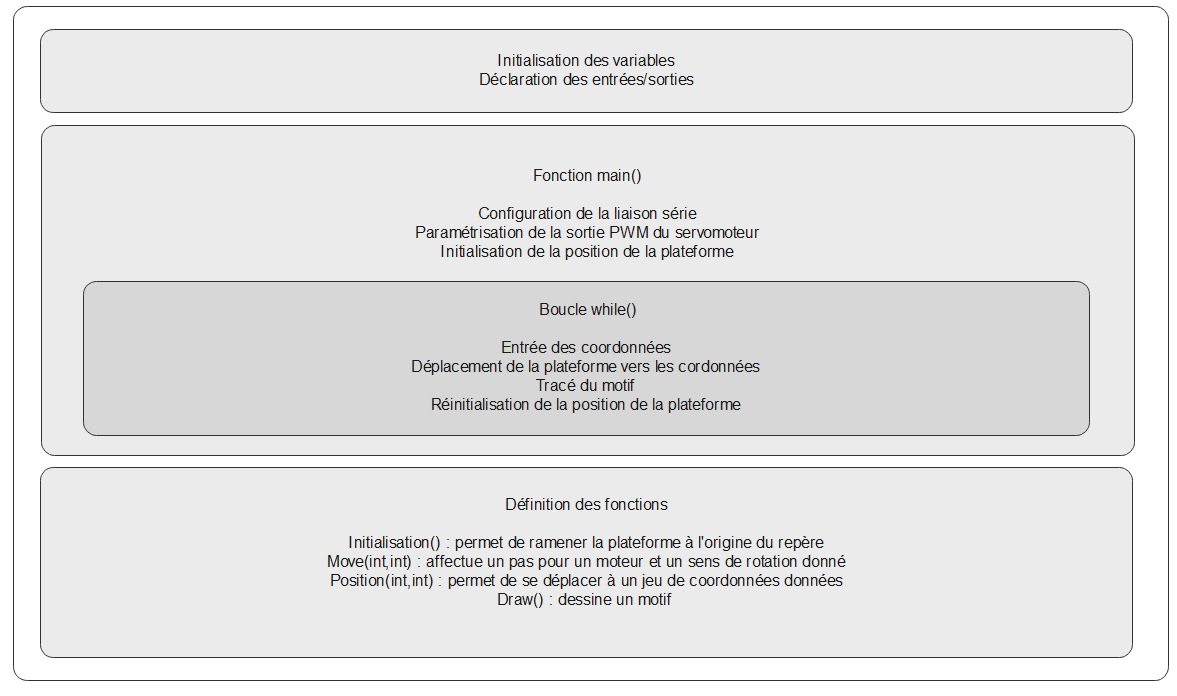
Si la puissance fournie au moteur est trop faible, le champ magnétique à l’intérieur du moteur le sera aussi et le couple du moteur sera trop faible pour pouvoir actionner le pignon correctement.

Notre projet demande également l’utilisation d’un servomoteur. Ce type de moteur désigne un moteur à courant continu commandé par un potentiomètre. Il s’agit donc d’un système asservi en position. On s’en sert pour commander la hauteur du stylo par rapport à la table, un premier angle correspond à un stylo abaissé et un second au cas où le stylo est relevé. Un servomoteur a trois entrées : une pour l’alimentation, une pour la masse et une dernière pour un signal PWM dont le rapport cyclique caractérise l’angle que l’on veut obtenir.

Les derniers éléments du montage sont les interrupteurs placés au bout des axes de la table 2D. Leur état permet de savoir quand la plateforme est en butée. C’est notamment très utile dans la phase d’initialisation pour positionner la plateforme sur l’origine du repère de notre système de coordonnées.

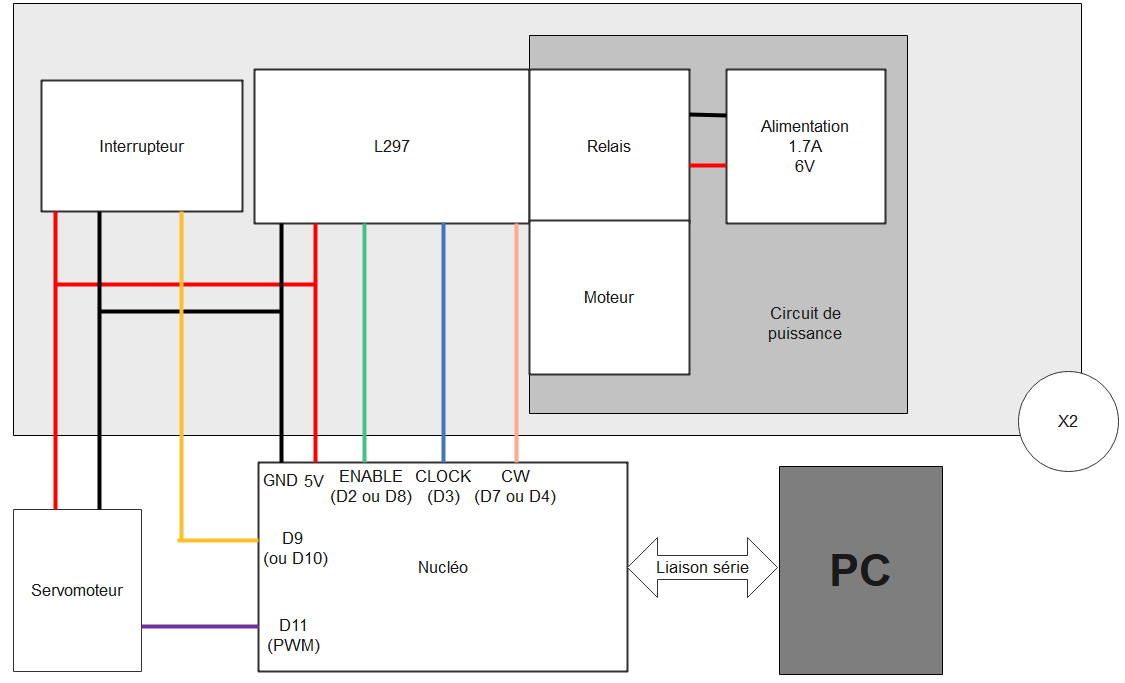
**Structure du code :**

Le programme implémenté sur la carte Nucléo est codé en C. Une première version du programme permettait de fixer la fréquence de l’horloge à l’aide d’un PWM, on commandait le déplacement de la plateforme en choisissant le temps pendant lequel on activait l’entrée ENABLE. Le programme joint en annexe permet de commander les déplacements de la plateforme en terme de nombre de pas à effectuer, ce qui présente un double intérêt : d’une part un gain en précision puisqu’on supprime une incertitude sur la distance parcourue qui vaut le double du pas; d’autre part la possibilité de faire varier au besoin la vitesse de translation de la plateforme facilement ce qui n’est pas possible avec l’autre méthode.



*Structure du code*

**Câblage :**



*Schéma de câblage*

***Détermination du pas de translation* :**

Quelque soit la méthode choisie pour commander le déplacement de la plateforme, on a besoin de connaître la valeur de la distance parcourue rectiligne par la plateforme pour une rotation d’un pas du moteur. La méthode utilisée pour le déterminer comporte deux étapes.

On commence par l’estimer en s’appuyant sur la formule avec le rayon approximatif du pignon et le pas angulaire fourni par la documentation.

On calcule alors le nombre de pas qu’il faudrait pour effectuer une distance fixée la plus grande possible mais pas trop non plus pour que la distance parcourue reste inférieure à la course totale de la plateforme. On mesure la distance réellement parcourue et on la divise par le nombre de pas pour avoir la valeur d’un pas latéral.

AN : En première approximation, on a . Une commande pour un déplacement de 200mm correspond à un déplacement réel de 137mm donc on en déduit que la valeur du pas est

**Conclusion :**

Au final notre système est quasi fonctionnel en boucle fermée, a une liaison série prête.Or chaque une des différentes parties fonctionne bien de façon autonome et le traitement d’images réussi à travailler avec l’IA. La plupart des expectatives du cahier des charges essentiel pour le projet ont été atteintes (sauf la communication IA-Table 2D) et même beaucoup des expectatives optionnelles ont été atteintes. Un code couleur a été ajouté au tableau du cahier des charges pour rendre compte de la réalisation de chacune des expectatives.

Le projet a bien fonctionné pour la majorité des aspects. Les grands points à améliorer seraient de prévoir plus de temps pour faire les liaisons entre les différentes parties et de mieux prévoir le développement de l’IA.

L’ensembles de codes utilisées lors du projet sonts fournis en annexe (Voir dossier Code)