

Echiquier Automatisé

Projet personnel



01 janvier 2024

Xavier AIGUAVIVA

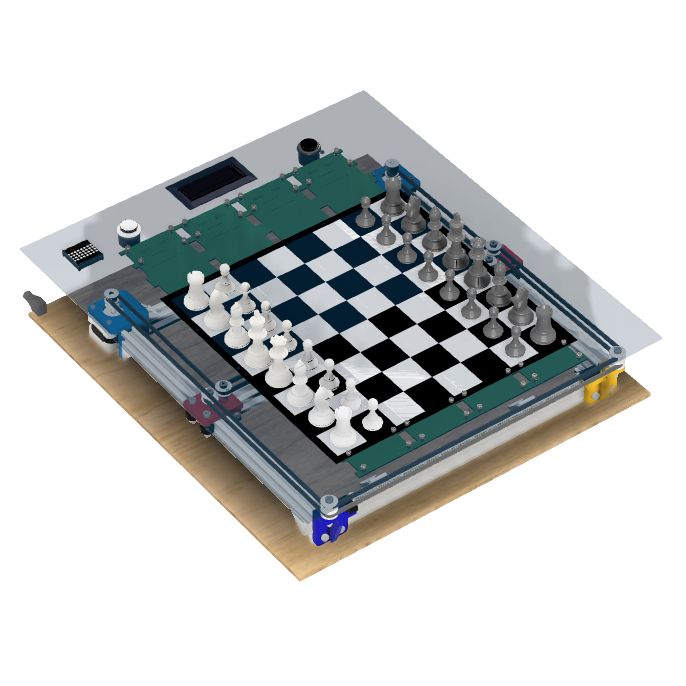


Table des matières

[2. Présentation 2](#_Toc188431839)

[3. Electronique 4](#_Toc188431840)

[1. Déplacer les pièces 0](#_Toc188431841)

[1. Moteurs NEMA 17 0](#_Toc188431842)

[2. Drivers Moteurs DRV8825 2](#_Toc188431843)

[3. Electroaimant 5](#_Toc188431844)

[4. Driver Electroaimant L298N 7](#_Toc188431845)

[2. Détecter la position des pièces et l’afficher 8](#_Toc188431846)

[3. Interface utilisateur 14](#_Toc188431847)

[4. Gestion les fonctions 14](#_Toc188431848)

[5. Alimentations 16](#_Toc188431849)

[4. Table XY 18](#_Toc188431850)

[6. Fonctionnement 21](#_Toc188431851)

[7. Table de vérité 22](#_Toc188431852)

[1. Intégration 24](#_Toc188431853)

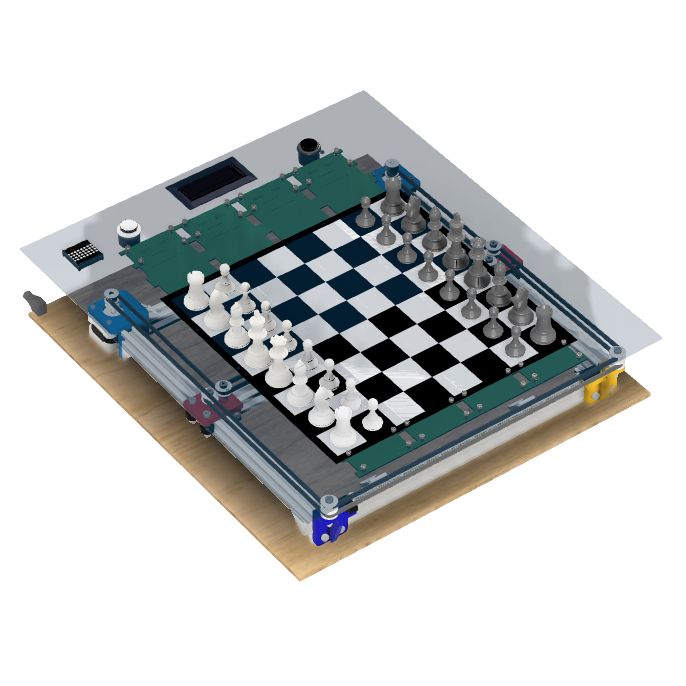
[2. Intelligence Artificielle 25](#_Toc188431854)

[8. IA Micro-Max 26](#_Toc188431855)

[3. Code 26](#_Toc188431856)

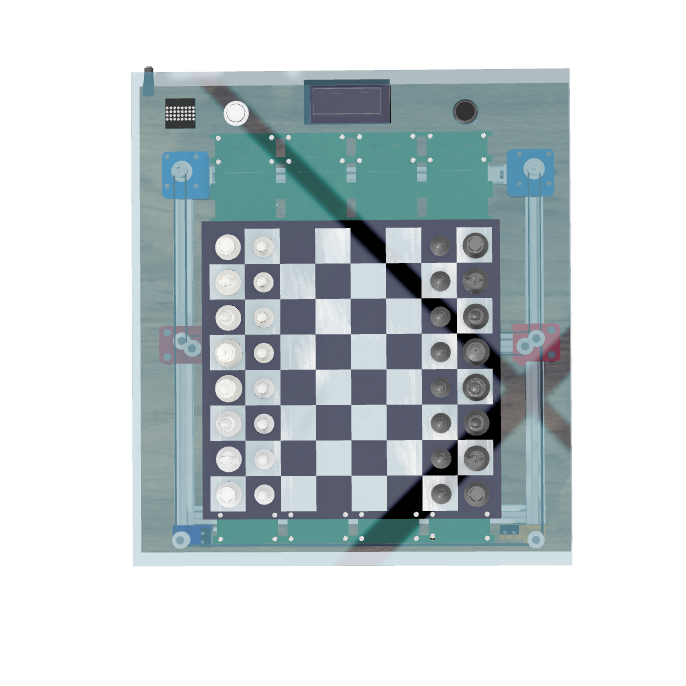
# Présentation

Le but de ce document de présenter la conception d’un système d’échiquier automatisé.



Cet échiquier devrait intégrer les fonctions suivantes :

* Timer,
* Détection de la position des pièces,
* Mouvement automatique de spièces,
* Intelligence artificielle.



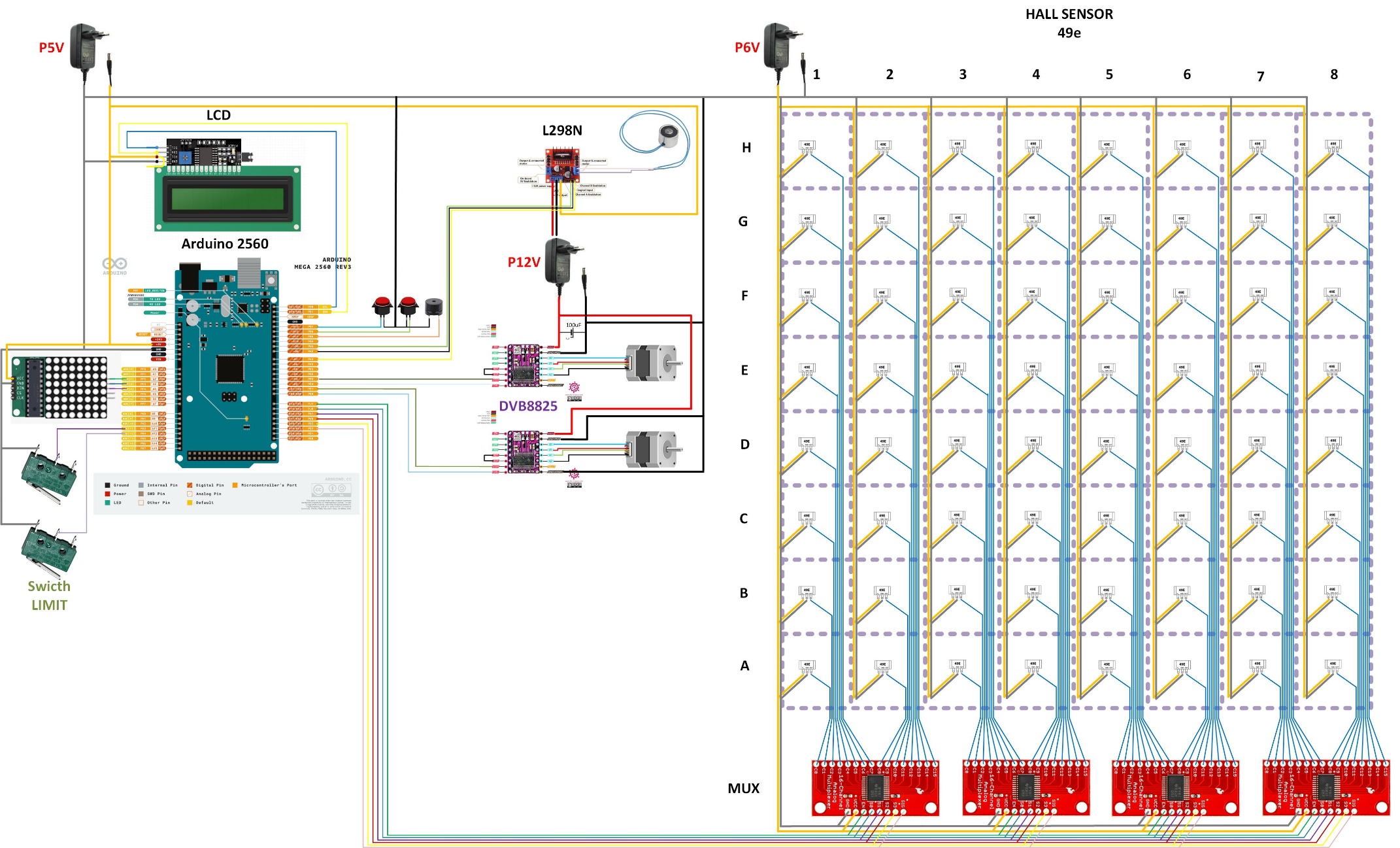
# Electronique

Pour le bon fonctionnement de l’échiquier, il va s’agir d’assurer les fonctions suivantes :

* Détecter la position des pièces et l’afficher,
* Avoir une interface utilisateur,
* Déplacer les pièces,
* Gérer les fonctions,
* Alimentations.

Dans le but de faciliter la phase de prototypage, nous allons utiliser des modules préconçus disponibles dans le commerce.

Le schéma du système est indiqué ci-dessous :



## Déplacer les pièces

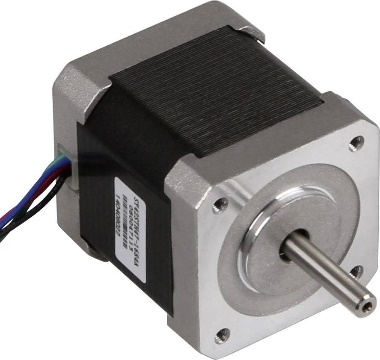
Electroniquement, pour déplacer les pièces, nous allons avoir besoin de :

* Deux moteurs,
* Deux hacheurs 4 quadrants pour piloter les moteurs,
* Un électroaimant pour déplacer les pièces,
* Un hacheur 4 quadrant pour piloter l’électroaimant.

### Moteurs NEMA 17

Le moteur utilisé doit compacte et être doté de suffisamment de couple pour assureur le mouvement du chariot.

Les caractéristiques du moteur **NEMA 17** correspond aux critères de choix.

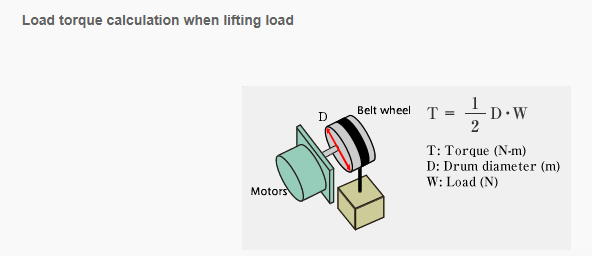


La tension d’alimentation de ce moteur est de **12V** pour un courant consommé de **0.4A**.

Son couple est de **0.26Nm.**

Le diamètre de la roue utilisée est de **6mm**.

Nous estimons la charge (chariot + rail Vslot + electroaimant, …) de manière conservative à 1kg.



Le couple du moteur devrait permettre de gérer le poids du chariot avec suffisamment de marge.

Nonobstant, il faut prendre en compte la résolution du driver (Micropas) qui va influencer le couple disponible.

Pour estimer le couple disponible à une résolution donnée, nous utiliserons l’approximation suivante :

L’approximation est essentiellement due aux non linéarités du moteur et du driver ainsi qu’à la non prise en compte des pertes mécaniques.

Cette approximation est acceptable car nous allons travailler à vitesse faible, avec des charges faibles et que nous souhaitons uniquement une tendance.

La fonction de transfert est la suivante :

Afin de trouver un compromis entre la précision, la vitesse, le couple et le bruit/vibration, nous pouvons prendre une résolution de 8 micros pas, donnant un couple de **5.07Ncm > 2.94Ncm.**

La marge du couple semble acceptable.

La vitesse max du moteur est défini par :

Le temps de commande du moteur entre chaque pas ne doit pas inférieur à :

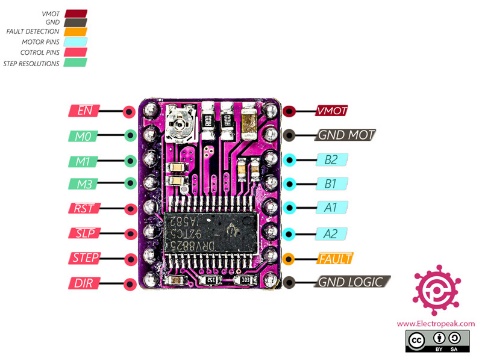
Ajouter le cablage du moteur

### Drivers Moteurs DRV8825

Le composant DRV8825 est un driver de moteur pas à pas utilisant un H-bridge.

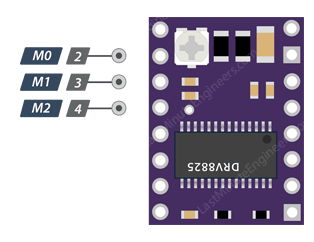
Il peut être alimenté en 12V, tolère le microstepping et supporte jusqu’à 2.5A de courant max consommé.

Il semble correspondre aux besoin du moteur.



Afin de régler la résolution du driver, il faut trouver un compromis entre la précision, la vitesse, le couple et le bruit/vibration.

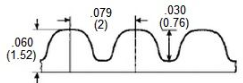
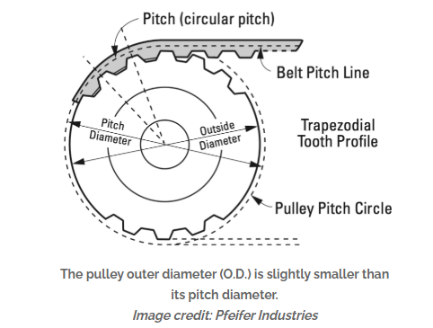
Le réglage de la résolution du driver peut être réalisé via 3 proches du driver :



Comme indiqué ci-dessus, la résolution a été choisie à 8 micros pas est réglable via 3 broches du driver.



Nous allons confirmer que la précision de déplacement reste correcte sur l’échiquier et nous allons évaluer la résolution de déplacement, horizontal et vertical :



Puis la résolution par case de l’échiquier :

Pour en déduire le coefficient qui permettra le déplacement des pièces de case en case, horizontalement et verticalement :

Ce coefficient doit être utilisé pour tout déplacement d’une case à une autre, dans les directions horizontales et verticales.

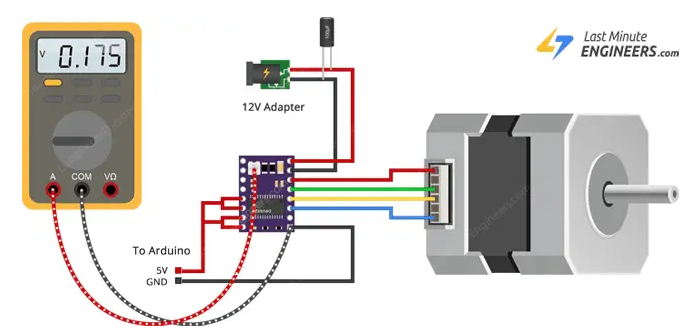
Ce coefficient pourra être ajustée en fonction des résultats des tests.

Nous allons maintenant évaluer la résolution de déplacement diagonale :

Diagonal Move

Ce coefficient doit être utilisé pour tout déplacement d’une case à une autre, dans les directions diagonales.

La limitation du courant se règle via le potentiomètre du module



La valeur de limitation en courant est définie par la relation suivante :

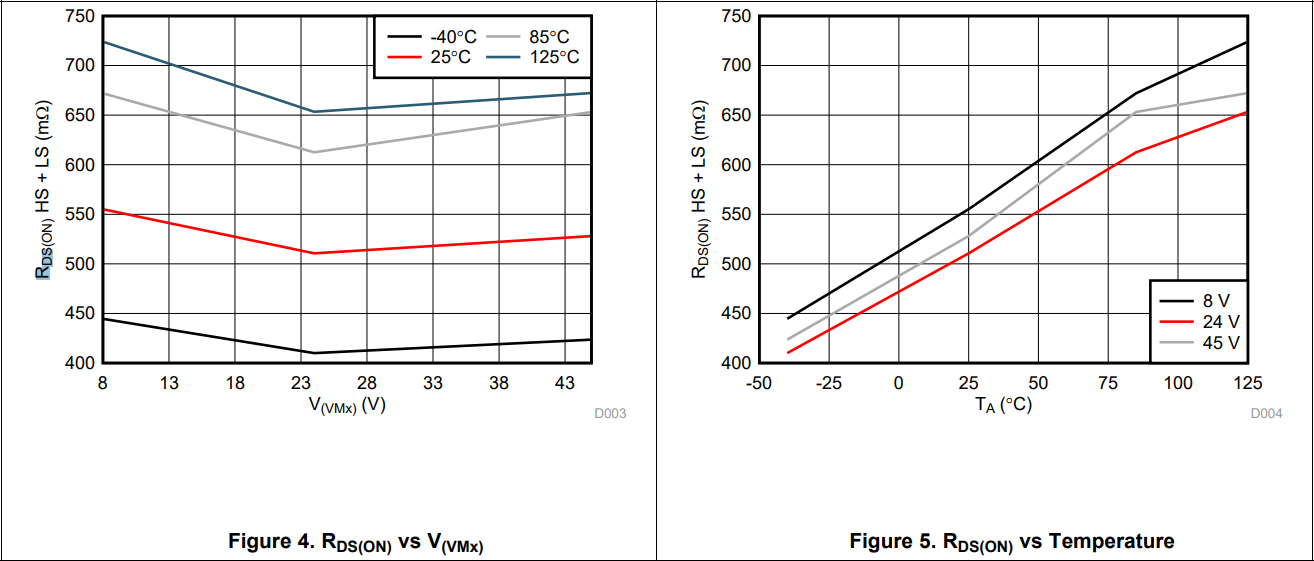
Le moteur pas à pas requiert 0.4A par phase soit 0.8Amax, nous devrons régler la tension de référence sur le module à :

En prenant une marge de 20% :

Nous allons estimer la puissance totale dissipée dans le boitier :

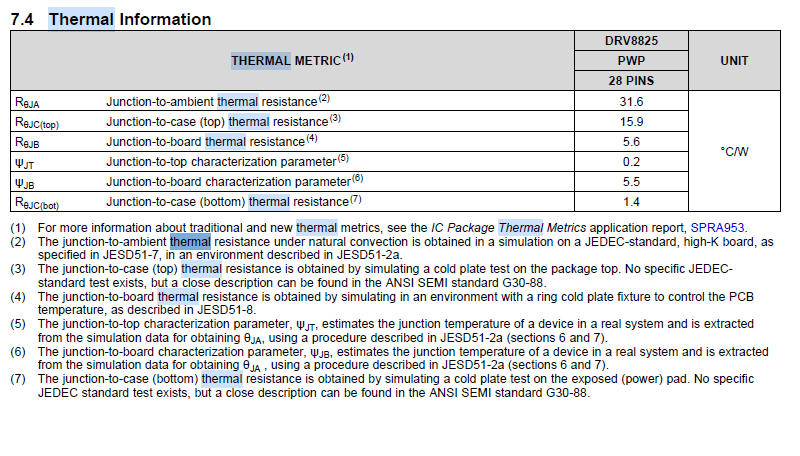
La valeur du courant RMS équivaut à 0.7 fois le courant max.

La valeur de la résistance série des MOSFETs est le pire cas de la datasheet.



Nous pouvons en déduire la température max du composant :

Sachant que le module est fourni avec un dissipateur thermique, nous prendrons une résistance thermique de Rthjc de 15.9°C/W.



### Electroaimant

Dans un premier, nous utiliserons des aimants néodyme rond N35 NdFeB, 8x3mm, intégrés dans les pièces d’échecs.



Il faut alors choisir un électro aimant capable de déplacer ces aimants plus la pièce d’échec associée.

Calculons la force de l’aimant. Pour cela, nous avons besoin de du champ magnétique de l’aimant (1.2T pour un NE35), la surface de l’aimant en m² ainsi que la perméabilité magnétique du vide.

Nous obtenons :

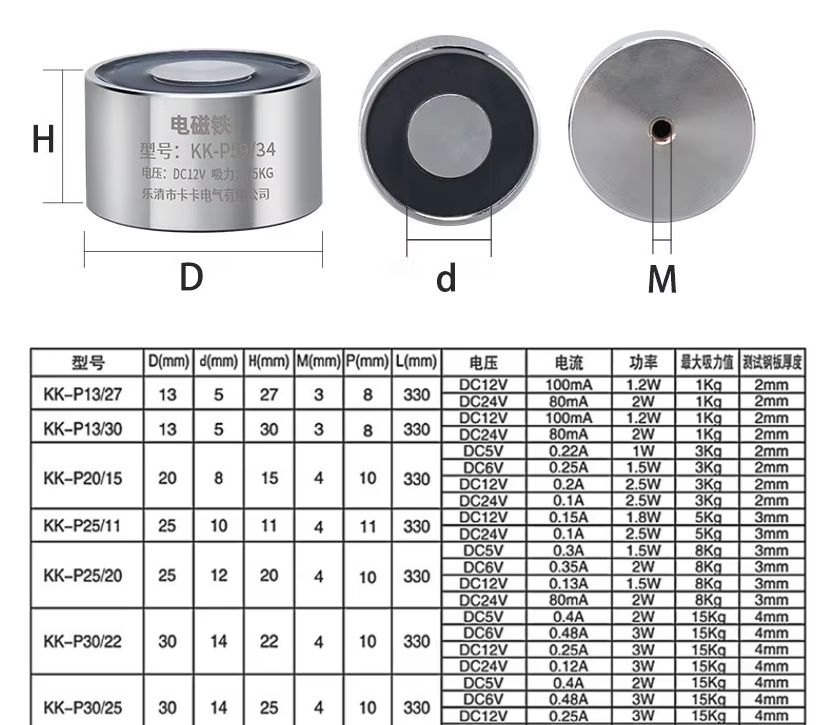
Cette force est valable uniquement au contact de l’aimant, elle diminue fortement avec la distance.

On peut approximer la valeur pour une distance d de 1cm avec la formule empirique suivante :

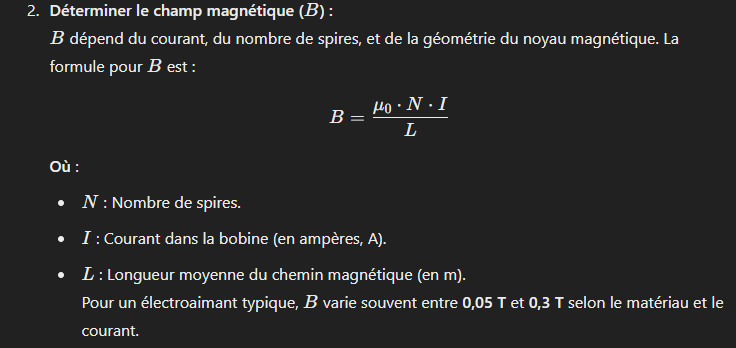
L’électro aimant doit avoir une force comprise entre .

Un électro aimant compact est nécessaire et nous nous tournerons vers un de la série KK-Pxx/yy.

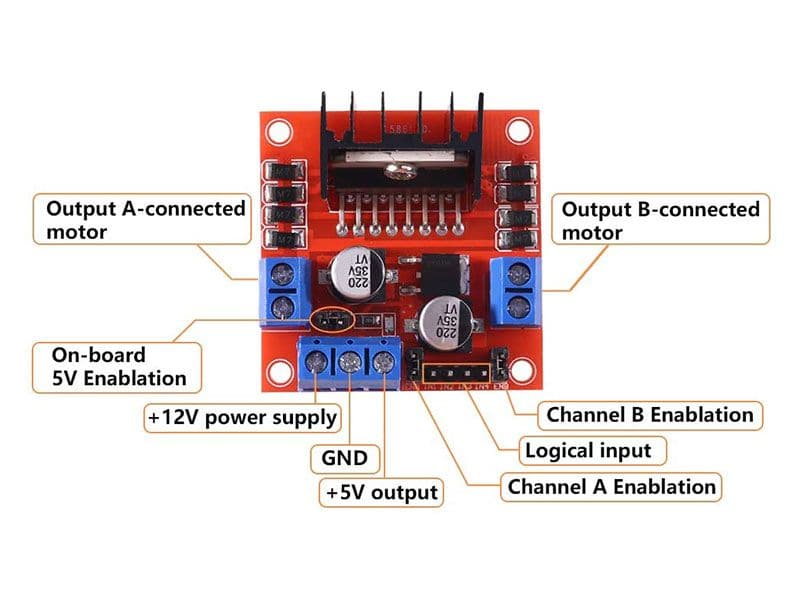




A reprendre



### Driver Electroaimant L298N



## Détecter la position des pièces et l’afficher

La détection des pièces se fait via les composants suivants :

* Détecteur de champ magnétique,
* Multiplexeur (MUX).

La détection choisie se fera via des aimants, intégrés dans les pièces d’échecs.

Deux solutions s’offrent à nous pour assurer cette détection de champ magnétique :

* Les relais « REED »,
* Les capteurs à effet HALL.

Les relais REED sont des capteurs TOR et permettent de remplir la fonction attendue et l’intégration et simple et rapide.

Cependant, les capteurs à effet HALL sont analogiques et permettent une plus grande plage de détection de champ magnétique.

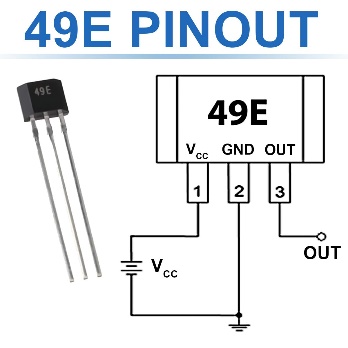
De plus, ils permettent de détecter les deux polarités de l’aimant, permettant d’utiliser une polarité pour les pièces blanches et une pour les pièces noires.

Nous obtiendrions alors ce type de matrice, avec :

* 0 : absence de pièce,
* -1 : pièce noire présente,
* 1 : pièce blanche présente.



Pour ces raisons et malgré une intégration plus complexe, nous choisissons les capteurs à effet hall **SS49e** en boitier TO92.



Nous n’avons pas besoin d’une grande précision de mesure ni d’une fonction de transfert. La variation de valeur relative devrait être suffisante pour assurer la détection des pièces aimantées.

Pour cette raison, nous n’utiliserons pas de chaine d’acquisition pour améliorer la précision du signal analogique et câblerons directement le capteur sur l’ADC du microcontrôleur.

Pour information, la fonction de transfert du capteur est définie par :

L’alimentation du capteur peut être réalisé de 2.7V à 6.5V.

Nous n’avons pas les informations pour extrapoler la fonction de transfert en fonction de la tension d’alimentation.

Nous en déduirons, dans premier temps, que la tension de sortie du capteur est linéairement proportionnelle à la tension d’alimentation sur toute la plage de mesure (ainsi que la sensibilité).

La plage de mesure correspond aux spécifications des entrées analogiques du microcontrôleur analogiques pour une tension de 6V.

Par conséquent, afin de compenser les éventuelles chutes de tension et d’augmenter la plage de mesure (et donc la précision), nous utiliserons une tension d’alimentation de 6V.

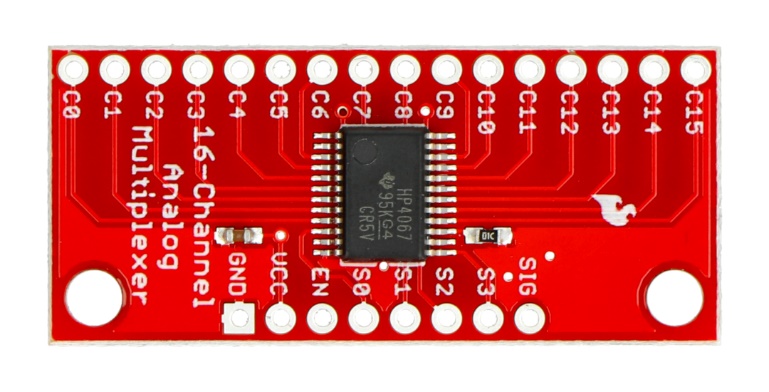
La précision de l’ADC est de 10 bits et en prenant l’hypothèse d’une plage de mesure de 4.6V, nous obtenons :

Réaliser des mesures pour valider cette hypothèse.

L’échiquier possédant 64 cases, il est donc nécessaire d’implémenter 64 capteurs.

Nous utiliserons 4 MUX de 16 voies pour assurer l’acquisition des 64 signaux analogiques.

Le MUX choisie sera le **CD4067B** et peut également être alimenté en 6V.



Les MUX partagerons les signaux d’adressage, S0 à S1 (adressage de la voie lue), pour économiser les broches du microcontrôleur.

Le temps de transition Max donnée par la datasheet est de 600ns. Ce temps est largement inférieur au besoin de temps d’acquisition des capteurs.

Nous prendrons un temps d’acquisition arbitraire de **5** ms par voie, ce qui conduit à un temps de **64x5=320ms** pour l’acquisition de l’intégralité des capteurs

La résistance série du MUX est de 1050 Ohm. En prenant en compte un courant aborsbé par l’ADC de 50nA, on obtient :

La tension d’offset est totalement négligeable dans notre application.

Les MUX sont réparties de la manière suivante sur la matrice de l’échiquier :



La matrice de commande des MUX est présentée ci-dessous :



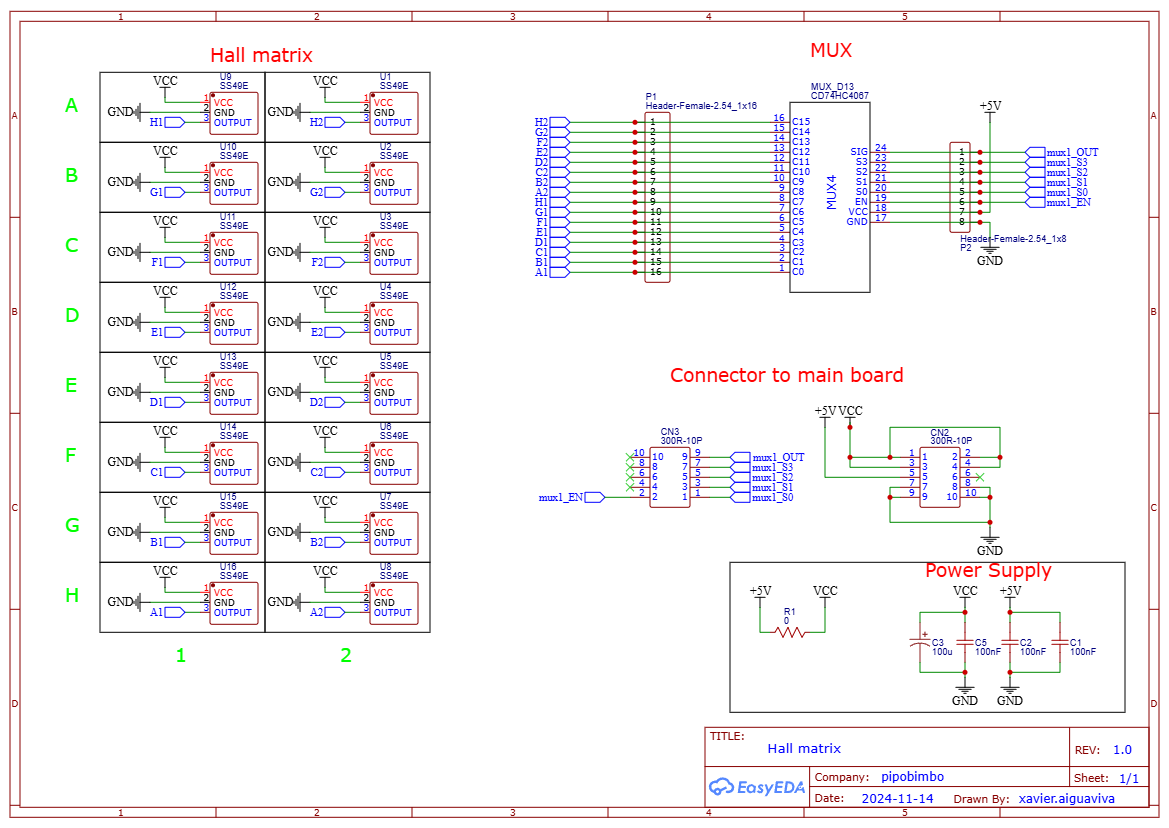






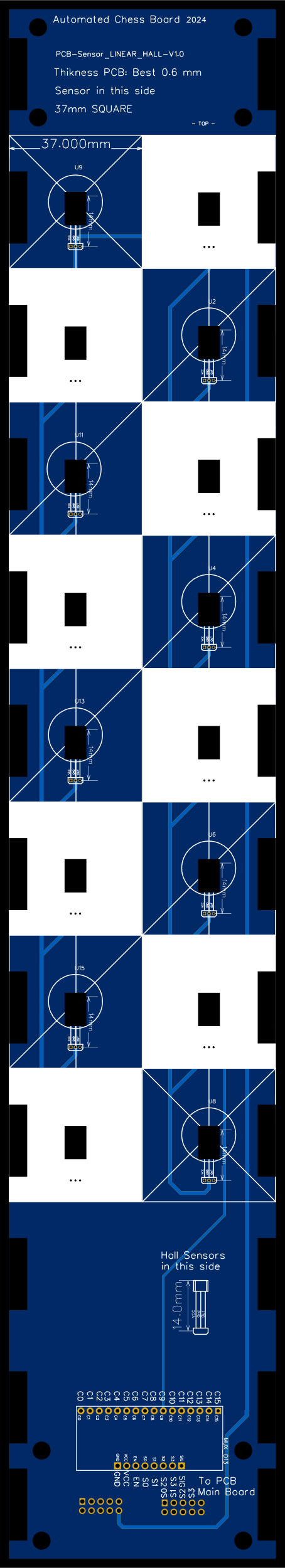
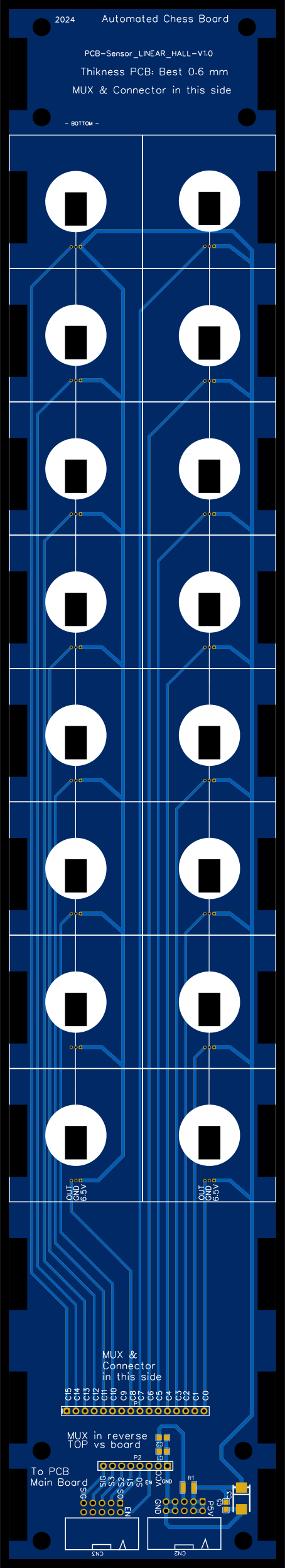
Afin de faciliter l’intégration, il est souhaitable de concevoir un PCB adapté au besoin.

La carte électronique se composera de 16 capteurs et du MUX associé :



Un strap permet de dissocier les alimentations des capteurs et du MUX si besoin.

Des condensateurs de découplage, chimique et céramique, sont ajoutés sur chacune des cartes (les calculs n’ont pas été réalisés, leur valeur sera ajustée au besoin).



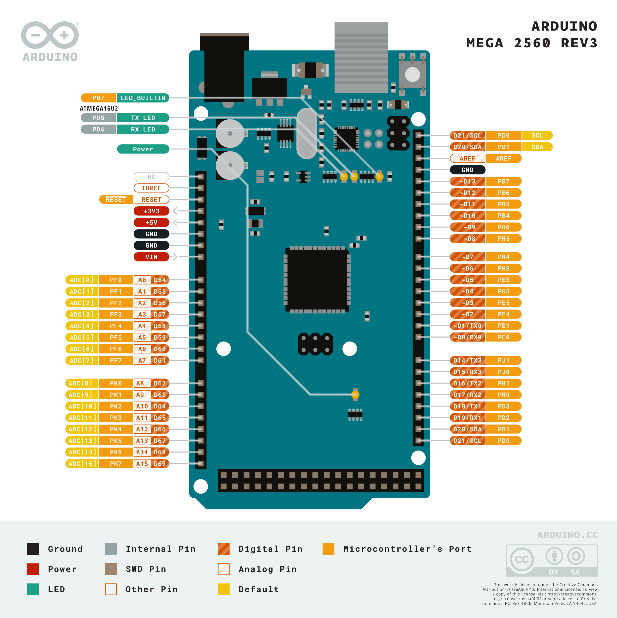
## Interface utilisateur

Pour l’interface utilisateur, pour utiliserons :

* Un écran LCD,
* Boutons.

## Gestion les fonctions

Pour gérer toutes les fonctions, nous choisirons arbitrairement une carte Arduino 2560 V3 ainsi qu’une carte mezzanine RAMP V1.4.

Il est nécessaire d’avoir un nombre suffisant de broches disponibles ainsi qu’une performance suffisante de mémoire vive pour permettre l’utilisation d’une IA d’échec.

La carte Arduino possède seulement 8 kB de RAM, 256 kB de flash, et une vitesse de 16 MHz.



## Alimentations

Les alimentations nécessaires au bon fonctionnement du système sont les suivantes :

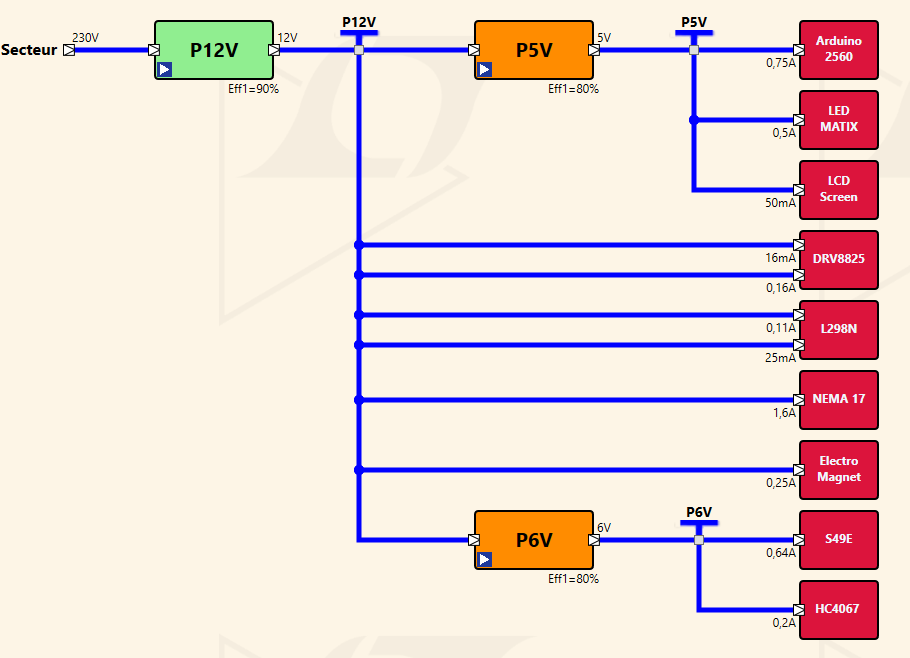
* 12V,
* 6V
* 5V.

Les estimations des consommations max sont présentées ci-dessous :



Le rendement des drivers est ici négligé car les données sont manquantes.

Des mesures seront à réalisées.

L’arbre de puissance de décompose de la manière suivante : 

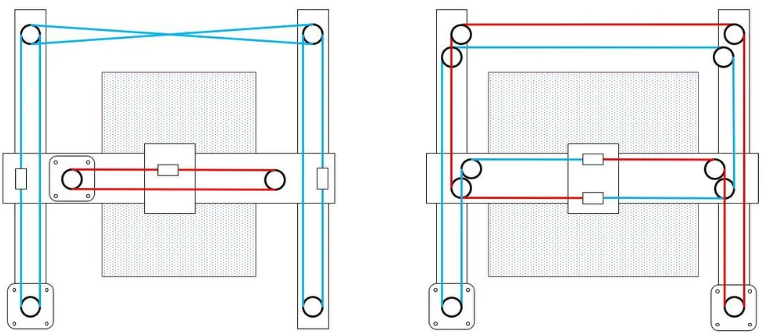
Le 12V sera généré à partir de l’alimentation secteur.

Les rendements des alimentations et drivers sont estimés pour le moment et devront être ajustés.

# Table XY

Deux types de tables peuvent être utilisées pour réaliser les mouvements du chariot :

* Table type cartésienne,
* Table type XY.



Cartésien XY

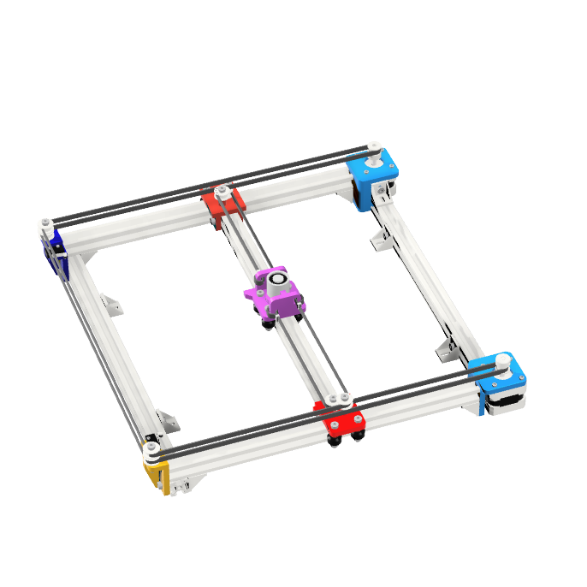
La table cartésienne offre plus de simplicité à mettre en œuvre en termes de fabrication de d’utilisation mais l’encombrement est important du au moteur installé sur le chariot.

Tandis que la table XY permet une le déplacement du chariot dans une zone plus importante et encombrement plus faible mais le câblage est plus complexe et la précision de déplacement est plus faible.

La Table XY répond davantage au besoin et sera donc sélectionnée.



La table se présente sous la forme suivante :



Et est constitué des éléments suivants :

* 4x V-slot,
* 2x courroies,
* 2x moteurs pas à pas,
* 8x poulies,
* 2x poulies dentées,
* 2x capteurs fins de courses
* 4x équerres de fixation,
* 6x supports spécifiques,
* 8x roues,
* Visseries, écrous, Spacers.

La modélisation a d’abord été réalisé sur FREECAD puis migrer sur FUSION 360.

Ajouter images de tous les éléments

Vue de côté, le système devrait se composer comme suit :



## Fonctionnement

Les déplacements en diagonal sont initiés par la rotation d’un SEUL moteur. La modification de la rotation du moteur entraine un déplacement inverse.

Si deux moteurs tournent dans la même direction, un déplacement horizontal est généré.

Si deux moteurs tournent la direction opposée, un déplacement vertical est généré.

Au démarrage, les coordonnées X et Y du chariot sont inconnus, il est donc nécessaire de calibrer le chariot via les deux capteurs fins de courses (position 0).

Ces capteurs fins de courses sont fixés sur le chassis, la position 0 est donc inaltérable et sert de référence.



## Table de vérité

Nous obtenons donc la table de vérité suivante :



Elle présente les commandes à appliquer pour chacune des variables afin d’obtenir les déplacements voulus.

# Intégration

# Intelligence Artificielle

Concernant la gestion des coups des pièces adverses, il est nécessaire d’intégrer une IA d’échec dans le code adapté aux microcontrolleur Arduino.

Voici une sélection de candidat potentiel :

1. **Stockfish pour microcontrôleurs**

* **Stockfish Lite** ou versions ultra-légères de Stockfish.
* Stockfish est l’un des moteurs d’échecs les plus puissants et il existe des forks optimisés pour les microcontrôleurs.
* Ces versions allégées se concentrent sur des algorithmes simplifiés et utilisent moins de mémoire.

1. **Micro-Max**

* **Micro-Max** est un moteur d’échecs ultra-léger écrit en C.
* Taille : Environ 1-2 kB de mémoire.
* Caractéristiques :
  + Implémente un moteur d'échecs fonctionnel avec un algorithme de recherche alpha-bêta.
  + Gère les règles standard des échecs, mais avec des fonctionnalités limitées (sans prise en charge des règles complexes comme la sous-promotion personnalisée).

1. **TSCP (Tiny Simple Chess Program)**

* **TSCP** est un moteur d’échecs écrit en C, conçu pour être simple et éducatif.
* Taille et ressources :
  + Consomme peu de mémoire (moins de 20 kB après simplification).
  + Parfait pour des projets nécessitant une IA basique d’échecs.

1. **Chess for Arduino (Chess4Arduino)**

* Spécialement conçu pour les microcontrôleurs, ce projet est disponible sur GitHub.
* Il implémente un moteur d’échecs léger, prenant en charge les mouvements légaux, les échecs et les mats.

Dans le but de faire un choix, nous allons nous appuyer sur les critères suivants :

* **Mémoire disponible** : La **RAM** de l'Arduino Mega 2560 (8 kB) limite le moteur que nous pouvons utiliser. Un moteur léger avec des tables de transposition minimales est indispensables.
* **Temps de calcul** : Les moteurs d’échecs fonctionnent en évaluant des positions via des recherches dans un arbre. Avec un Arduino Méga, il faudrait limiter la profondeur de calcul à 2-4 coups pour maintenir un temps de réponse acceptable.
* **Fonctionnalités** : Si l’on souhaite que l’IA joue à un niveau acceptable, il faut s’assurer qu’elle gère les règles de base (roque, promotion, prise en passant).

Il sera éventuellement nécessaire d’adapter le code de l’IA afin qu’il soit compatible avec l’environnement Arduino et que son utilisation des ressources soient le faible possible tout en gardant un niveau de jeu correct.

Pour un Arduino Méga 2560, le choix d'un moteur comme **Micro-Max** ou **Chess4Arduino** semble idéal.

Ces moteurs sont suffisamment légers pour fonctionner dans l’environnement contraint de l’Arduino, tout en offrant des performances correctes pour une application comme un échiquier automatique.

Dans un premier temps, nous essaierons l’IA Micro-Max :

<https://gist.github.com/purplejacket/ca8b9e53b5dddcad15cb8ee659260fff>

<https://home.hccnet.nl/h.g.muller/max-src2.html>

## IA Micro-Max

Il intègre Negamax, un algorithme permettant d’accélérer les calculs.

La table de hachage à une taille originelle de 192Mb. Il est possible de la diminuer afin d’améliorer les performances de l’algortithme dans l’arduino.

Il est possible de réaliser cette diminution via la ligne de code suivante :

* #define U 0x100008 : pour une table de 12Mb

La valeur doit être une puissance de 2 plus 8 et d’entrer la valeur en haxadecimal.

# Code