الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة
République Algérienne Démocratique et Populaire
وزارة التعليـم العالـي والبحـث العلمـي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Centre Universitaire Abdelhafid Boussouf - Mila

Institut des Sciences et de Technologie

Département des Sciences et Techniques

Nº Réf :……

Projet de Fin d'Etude préparé En vue de l'obtention du diplôme de MASTER

Spécialité : Electromécanique

Conception et réalisation d'une machine CNC

Réalisé par :
- Mohammed Raïd ABDEMEZIANE
- Djamal BENABIED

Soutenu devant le jury :

Dr.	**Hocine BENSLIMANE**	**Président**
Dr.	**Hichem BOUCHENITFA**	**Examinateur**
Dr.	**Farida MEDJANI**	**Promoteur**

Année universitaire : 2019/2020

Résumé

Les machines CNC jouent un rôle important dans plusieurs domaines de l'industrie moderne. Elles sont largement utilisées dans la production grâce à leur précision et haute qualité, ce que nous a encouragés à choisir ce projet. Nous avons présenté certaines notions principales à propos des machines CNC et du moteur pas à pas. Ce dernier est programmé avec G-code et le microcontrôleur MSP430G2553.

Dans la partie mécanique, nous avons conçu et fabriqué notre propre modèle d'une machine CNC. Des tests ont été effectués sur la machine afin de valider et de vérifier les algorithmes précédemment développés. Les résultats obtenus sont considérés meilleurs et encourageants.

Mots clés : Machine CNC, Microcontrôleur, G-code, moteur pas à pas, SolidWorks.

ملخص

تلعب آلات CNC دورًا مهمًا في العديد من مجالات الصناعة الحديثة. تستخدم على نطاق واسع في الإنتاج بفضل دقتها وجودتها العالية، وهذا ما شجعنا على اختيار ي هذا المشروع. قدمنا من خلاله بعض المفاهيم الرئيسية حول آلات CNC والمحرك الخطوي. تمت برمجة هذا الأخير باستخدام G–code والميكروكونترولر MSP430G2553.

فيما يتعلق بالجانب الميكانيكي، قمنا بتصميم وتصنيع نموذج آلة CNC خاص بنا. تم اجراء اختبارات مختلفة من اجل التحقق من الخواروميات المطورة في الأقسام السابقة والتحقق منها حيث تعتبر النتائج التي تم الحصول عليها مشجعة ومثمرة للغاية.

الكلمات المفتاحية: آلة التحكم الرقمي، آلة CNC، G–code، المتحكم الرقمي، المحرك الخطوي، SolidWorks.

Abstract

CNC machines play an important role in several fields of the modern industries. They are widely used in production because of their precision and their high quality, which encouraged us to choose this project. We have presented some main notions about CNC machines and stepper motor. The latter is controlled programming with the G code software and MSP430G2553 microcontroller.

In the mechanical part, we designed and manufactured our own CNC machine model. Tests were carried out on the machine in order to validate and verify the algorithms previously developed. The results obtained are considered better and encouraging.

Keywords: CNC machine, microcontroller, G-code, stepper motor, SolidWorks.

Remerciements

Nous tenons avant tous à remercier Dieu le clément, le Tout-Puissant de m'avoir donné la santé, le courage, la volonté et la patience pour mener ce travail.

Au terme de ce travail, il nous est agréable d'exprimer notre profonde reconnaissance et notre gratitude les plus sincères à notre promotrice ***Madame Farida MEDJANI*** *Maitre de Conférence au Centre Universitaire Abdelhafid Boussouf Mila qui de longues dates nous a incités et nous a encouragés. Sans la confiance que nous avons témoignée au moment de la réalisation de ce travail, elle a été la plus grande aide pour concevoir ce travail et à le mener à terme, nous la remercions profondément pour le temps précieux qu'elle a consacré pour suivre notre travail et de nous avoir guidé. Merci madame pour votre disponibilité, votre bonne humeur et toutes les connaissances que vous nous avez apportées, quel que soit ce que nous allons écrire malheureusement qu'il ne sera possible d'exprimer toute l'estime que nous vous portons-madame.*

Nous tenons à remercier particulièrement ***Mr Hocine BENSLIMANE*** *Maître de conférence au centre universitaire BOSSOUF Abdelhafid de Mila, pour l'honneur qu'il nous fait en présidant le jury de ce Mémoire.*

Nous remercions particulièrement Mr ***Hichem BOUCHENITFA****, Maître de conférence au centre universitaire BOSSOUF Abdelhafid de Mila, pour l'intérêt qu'il a porté à ce travail et pour avoir accepté d'être examinateur de ce Mémoire.*

Un immense merci pour ***Pr. Tahar KEZAI*** *qui nous a permet de réaliser notre expérimentation au niveau du local de Inovvation Acadmey Mila, sans oublie toute l'équipe d'innovation acadmey :* ***Ghoulem Chelat*** *et* ***Ahmed Ben Messoud****.*

Nos vifs remerciements vont également aux enseignants de la classe des sciences technique pour leurs remarques pertinentes et constructives ainsi leurs conseils et leurs propositions pour l'enrichissement de ce travail.

Nos chaleureux remerciements s'adressent à tous les enseignants qui nous ont accompagnés durant tout notre cursus universitaire

Une grosse pensée pour nos collègues spécialement pour leurs aides et leur accompagnement tous le long de la réalisation de ce travail.

Cette période particulière n'aurait pas pu être aussi agréable sans l'amitié et le soutien de nos amies qui étaient toujours à notre côté pour nos prêter une mainforte et un soutien sans failles.

Nous souhaitons à remercier les personnes qui nous sommes les plus chères, toutes les personnes de nos familles et de notre entourage qui nous ont soutenues tout au long de ce projet de fin d'études

Il n'y a pas un mot qui peut exprimer nos immenses remerciements à notre cher parent, merci pour l'amour et le soutien constant et votre encouragement à distance, vous êtes et vous restez le lompe qui éclair notre avoie, merci pour tous vos sacrifices pour nous, merci d'être toujours notre support qui nous aide à grimpé toujours vers le haut, merci d'être notre premier modèle de la patience, de persévérance et de sacrifice, merci de nous mettre toujours fière de vous. Merci également à nos chers frères et sœurs pour leur encouragement tout le long de notre cursus universitaire.

Dans l'impossibilité de citer tous les noms, nous adressons nos plus sincères remerciements à toutes les personnes qui ont contribué de près ou de loin à la réalisation et l'aboutissement de ce travail.

Raid et Djamel

Dédicaces

Nous dédions ce modeste travail à :

À nos très chères et douces mères, nos très chers pères qui ont été tous jours auprès de nous, merci de nous avoir permis de réaliser notre parcours sans de ne jamais manquer de rien. Nous adressons au ciel les vœux les plus ardents pour la conservation de leur santé et de leur vie.

À nos frères et nos sœurs, les mots sont faibles pour exprimer la force de nos sentiments et la reconnaissance que nous vous portons. Que Dieu vous garde pour nous.

Et à ceux qui partagent nos bonheurs et malheurs nos chers amis

À nos collègues au Centre Universitaire Abdelhafid Boussouf- Mila et nos camarades de promotion 2019-2020

Nous voudrons avoir pitié du professeur FARDI Brahim et nous lui dédierons ce travail

À tous nos enseignants de notre cursus éducatif

À tous ceux qui nous sont chers et que nous aimons et qui nous aiment

Raid et Djamel

Sommaire

Chapitre 3 : Programmation

Chapitre 4 : Partie mécanique : conception et réalisation de la machine CNC

Liste des abréviations

MIT : Massachusetts institute of technology

MOCN : Machine outille a commande numerique

APT : Automatic programed tools.

CNC : Commande numerique par calculateur

FAO : La fabrication assistee par ordinateur

DVD : Digital versatile disc

RAM : Memoire vive ou memoire pc (random access memory)

AVR : Academie des renseignements exterieurs (agrupacio valencianista republicana)

MSP : Mixed signal microcontroller

RISC : Reduced instruction set compute

UART : Universal asynchronous receiver-transmitter

I2C : Inter-integrated circuit

SPI : Serial peripheral interface

CMOS : Complementary metal oxide semi-conductor

GPIO : General purpose input/output

ADC : Analog to digital converter (convertisseur analogique a numerique)

NMI : Non-maskable interrupt

PWM : Pulse width modulation

CAO : Conception assistee par ordinateur

CFAO : Conception et Fabrication Assistées par Ordinateur

EIA : Electronic Industries Alliance

Liste des tableaux

Liste des figures

Introduction générale

Introduction générale

L'automatisation des machines industrielles permet : l'amélioration de la qualité des produits, la simplification du travail humain et l'amélioration de la vitesse de production. Parmi les machines automatisée on trouve les machines CNC (machines commande numérique par calculateur).

Les machines CNC permettent de réaliser des tâches qui sont très compliquées à effectuer manuellement d'une façon économique et rentable à savoir: la fabrication des pièces mécaniques de précision (coupe, usinage), la couture et la broderie, métier du plâtre, la minisérie …etc. En analysant les machines CNC existantes dans le marché, on peut constater que :

- Ces machines CNC sont des boites noires (les programmes sources ne sont pas commercialisés) ;
- Le cout de la machine CNC est élevé.
- L'indispensabilité de la machine dans le marché durant la période de besoin.
- La nécessité de quelques fonctions spécifiques qui ne sont pas disponibles dans la machine CNC existant dans le marché (selon le cahier des charges rétabli par l'industriel).

Les raisons présentées ci-dessus nous ont motivé à réaliser une machine CNC à faible cout programmable directement en langage C. De ce fait, nous avons pu programmer directement notre propre algorithme, bibliothèque (library) et le programme d'interprétation et d'envoi de commande (G-code) vers le microcontrôleur.

Notre projet de fin d'études consiste à concevoir, réaliser et tester une machine CNC G-code, dont les mouvements des axes sont assurés par des moteurs pas à pas commandés par le microcontrôleur MSP430. La commande de moteur pas à pas est assurée par PWM et GPIO. Afin que nous puissions accéder et programmer directement les registres des périphériques du MSP430G2553, tous nos programmes sont écrits en langage C.

La conception et la réalisation de la machine demande des compétences en mécanique et électronique, des compétences acquit lors de notre cursus et qui ont été renforcées grâce à ce projet. De plus, la commande des moteurs en langage C demande des connaissances en informatique. Il a été pour nous un grand plaisir de travailler sur une activité possédant des enjeux aussi passionnants et pluridisciplinaire, et nous à donner la motivation pour mener à terme ce projet qui est conforme avec notre formation d'électromécanicien.

Ce mémoire est structuré comme suit :

- Le premier chapitre présente des généralités et les principales descriptions sur les machines CNC et les moteurs pas à pas.
- Le deuxième chapitre est consacré pour la partie électronique. Cette partie est dédiée à la description des composants électroniques et leur utilisation pour pouvoir réaliser notre commande de la machine (commande de deux moteurs pas à pas).
- Le troisième chapitre concerne la programmation de notre machine CNC à base de G-CODE et comment l'utiliser avec le microcontrôleur MSP430G2553.
- Dans le quatrième chapitre nous présentons les pièces mécaniques utilisées pour la réalisation de notre machine et nous décrivons la conception et la réalisation de la dite machine en utilisant le logiciel SolidWorks. Finalement, des tests des algorithmes de commande développés dans les parties précédentes sont effectués.

Ce mémoire comporte, en outre, une introduction générale et une conclusion générale.

Chapitre 1

Généralités sur les machines CNC et les moteurs pas à pas

1.1. Introduction

Une machine CNC c'est une machine qui peut être pilotée par les moteurs pas à pas, Ces derniers permettent de convertir directement un signal électrique numérique en un positionnement angulaire de caractère incrémental. Dans ce chapitre nous présentons un aperçu général sur les machines CNC et les moteurs pas à pas.

1.2. Généralités sur les machines CNC

1.2.1. Historique [1]

L'exploitation industrielle de la CN (**C**ommande **N**umérique) est liée au développement de l'électronique. En 1947, à Traverse City dans l'État du Michigan, John Parsons fabrique pour le compte de l'US Air Force des pales d'hélicoptère par reproduction. Pour façonner ses gabarits, il utilise une méthode consistant à percer plusieurs centaines de trous faiblement espacés de manière à approcher le profil théorique. L'emplacement et la profondeur de chaque trou sont calculés avec précision par un ordinateur IBM à cartes perforées. La finition de la surface est obtenue par des opérations manuelles de polissage.

Mais, lorsque l'US Air Force confie à ce même Parsons la réalisation de pièces de formes encore plus complexes pour ses futurs avions supersoniques, celui-ci réalise que sa méthode est trop approximative et que seul un usinage continu en 3 dimensions sera en mesure de donner satisfaction.

Au printemps 1949, il confie alors au Massachusetts Institute of Technology (MIT) le soin de développer des asservissements capables de piloter une machine qui recevra des instructions intermittentes à partir d'un lecteur de cartes. Cette machine, une fraiseuse prototype Cincinnati à broche verticale, conçue pour exécuter des déplacements simultanés suivant 3 axes, est officiellement présentée en septembre 1952 dans le Servomechanisms Laboratory du MIT. L'information mathématique étant la base du concept, on lui donne le nom de Numerical Control (NC).

Il faut encore attendre quelques années de vastes fonds de l'US Air Force et l'appui des chercheurs du MIT pour rendre la première Machine-outil à commande numérique(MOCN) réellement opérationnelle.

Les différentes étapes de développement de la CN sont les suivantes.

- 1954 : Bendix acquiert le brevet de Parsons et fabrique la première CN industrielle.
- 1955 : à Font du Lac (Wisconsin), le constructeur américain Giddins & Lewis commercialise la première MOCN.
- 1959 : apparition de la CN en Europe (foire de Hanovre). Le MIT annonce la création du langage de programmation APT (Automatic Programed Tools).
- 1960 : apparition du système DNC (Direct Numerical Control)
- 1964 : en France, la Télémécanique Éléctrique lance la CN NUM 100 conçue à base de relais Téléstatic.
- 1968 : la CN adopte les circuits intégrés ; elle devient plus compacte et plus puissante. Le premier centre d'usinage est mis en vente par Kearney & Trecker (USA).
- 1972 : les minicalculateurs remplacent les logiques câblées ; la CN devient CNC.
- 1976 : développement des CN à microprocesseurs.
- 1984 : apparition de fonctions graphiques évoluées et du mode de programmation conversationnel, début de l'ère de la fabrication assistée par ordinateur (FAO).
- 1986 : les CN s'intègrent dans les réseaux de communication, début de l'ère de la fabrication flexible (CIM : Computer Integrated Manufacturing).
- 1990 : développement des CN à microprocesseurs 32 bits

1.2.2. Définition de la commande numérique

La commande numérique (acronyme CN) est le système de contrôle qui coordonne les mouvements d'une machine-outil de manière à ce que l'outil suit des trajectoires prédéfinies sur des axes spécifiques sans que l'opérateur ait à intervenir directement. Les ordres de mouvement ou de déplacement, la vitesse de ces déplacements et leur précision, sont donnés à partir d'informations numérique (code numérique). Ces informations sont codées sur des supports tel que : rubans perforés, cassettes ou disquettes magnétiques ou simplement sauvegardés.

1.2.3. Définition d'une machine CNC

La machine-outil à commande numérique CNC (**C**omputer **N**umerical **C**ontrol en anglais) est une machine-outil dotée d'une commande numérique assurée par un ordinateur. C'est une machine totalement ou partiellement automatique à laquelle les ordres sont communiqués grâce à des codes qui sont porté sur un support matériel. Le premier rôle d'une machine CNC est de générer des mouvements, elle recevra des valeurs de positionnement de vitesse et d'accélération et générera suite à un traitement des consignes numériques en sortie [2].

1.2.4. Domaine d'utilisation

L'utilisation de la commande numérique ne se limite pas aux machines-outils travaillant par enlèvement de la matière avec des outils coupants. Elle est présente sur des installations de découpe par faisceau laser, en électroérosion que ce soit en défonçage ou en découpe par fil, en poinçonnage ou pliage de produits en feuille, pour la mise en place des composant, lors des opérations d'assemblage,…. Elle sert aussi à piloter des tables traçantes, les machines à mesurer tridimensionnelles, les robots.

La machine à outil à commande numérique représentent le moyen de production le plus important des pièces mécaniques. Elle nécessite des gestes précis et/ou répétitifs pour effectuer diverses opérations : percer, scier, rectifier, découper, fraiser, plier, graver, tarauder, souder, visser, déposer un matériau. Les matériaux qui peuvent être usinés sont très divers, la caractéristique principale qui les différencie est la dureté. Plus le matériau est dur, plus il faudra que la CNC, ainsi que l'outil qui l'équipe aient la qualité et la puissance nécessaires pour usiner le matériau. D'où de nombreux dispositifs CNC diffèrent entre eux principalement par l'outil qui est utilisé.

Les machines CNC sont employées dans de nombreux secteurs industriels : métallurgies, bois, textiles... Elles sont associées à des nouvelles technologies ; laser, électroérosion, jet d'eau.

1.2.5. Principe de fonctionnement d'une machine CNC

Les machines à commande numérique sont devenues des moyens de production incontournables dans l'industrie. Elles permettent des cadences de production importantes et facilitent l'obtention de surfaces complexes (formes arrondies …) [3]. Ce type de machine se compose ainsi de deux parties complémentaires (figure 1) :

- La partie opérative.
- La partie commande.

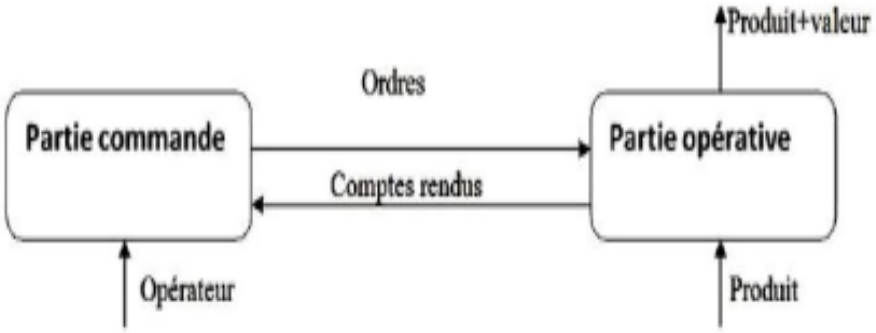


Figure 1 : Décomposition d'une machine numérique [2].

La partie commande est composée d'un calculateur ou d'un ordinateur et d'éléments électroniques capables de piloter les moteurs, cette partie permet de piloter la partie opérative.

La partie opérative comporte des axes de déplacement, la tête de l'outil et les actionneurs (les moteurs).

Des ordres vont être générés vers la commande par le biais d'un code machine ou par action manuelle de l'opérateur. La commande va traiter ces informations et générer des consignes afin d'obtenir les déplacements voulus par le biais des moteurs d'axes. Des contrôles de vitesse et de position seront alors effectués de manière continue par la machine.

1.2.6. Classification des machines CNC

Les machines à commande numérique sont classées selon :

- Le mode de fonctionnement de la machine.
- le mode d'usinage.

1.2.6.1. Classification des CNC selon le mode de fonctionnement

a) Fonctionnement en boucle ouvert

En boucle ouverte (figure 2), le système assure le déplacement du chariot mais ne le contrôle pas.

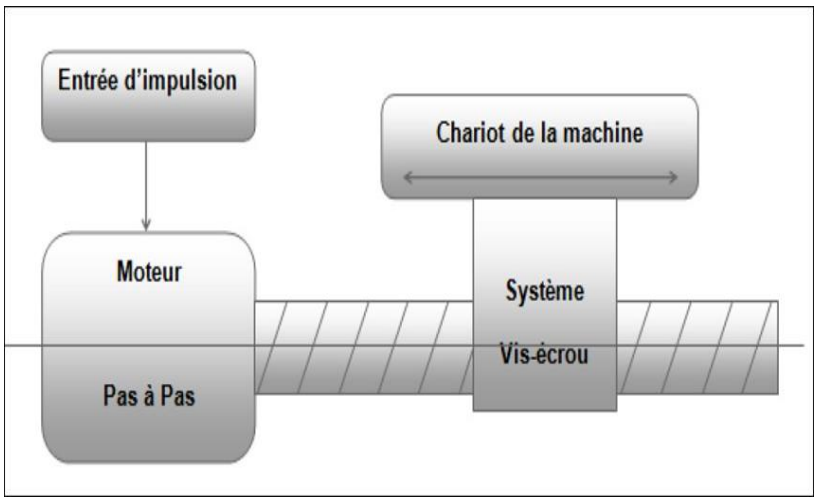


Figure 2 : Fonctionnement en boucle ouverte [4]

b) Fonctionnement en boucle fermé

En boucle fermée (figure 3) le système contrôle le déplacement ou la position jusqu'à égalité des grandeurs entrée (***E***) désirée et celui mesuré (***Gm***).

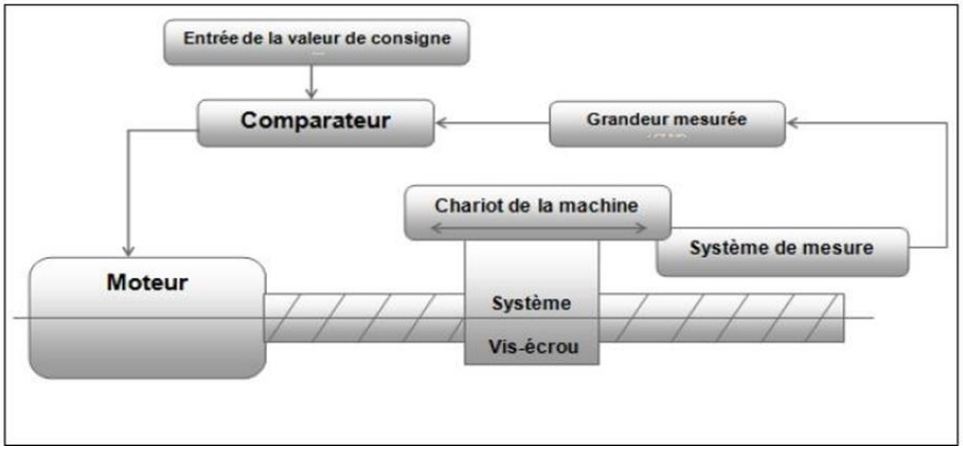


Figure 3 : Fonctionnement en boucle fermée. [4]

c) Fonctionnement avec commande adaptative

La commande adaptative (figure 4) réalise d'une façon continue et automatique l'adaptation des conditions de coupe. Des capteurs relève les valeurs de couple de la broche, l'amplitude de vibration de la broche, la température au point de coupe. Ces information sont transmise à une unité spéciale qui les envoi vers le directeur de commande numérique qui agit selon l'analyse des informations sur les condition de coupe pour permettre une meilleur qualité de travail, une meilleur productivité et une plus grande sécurité.

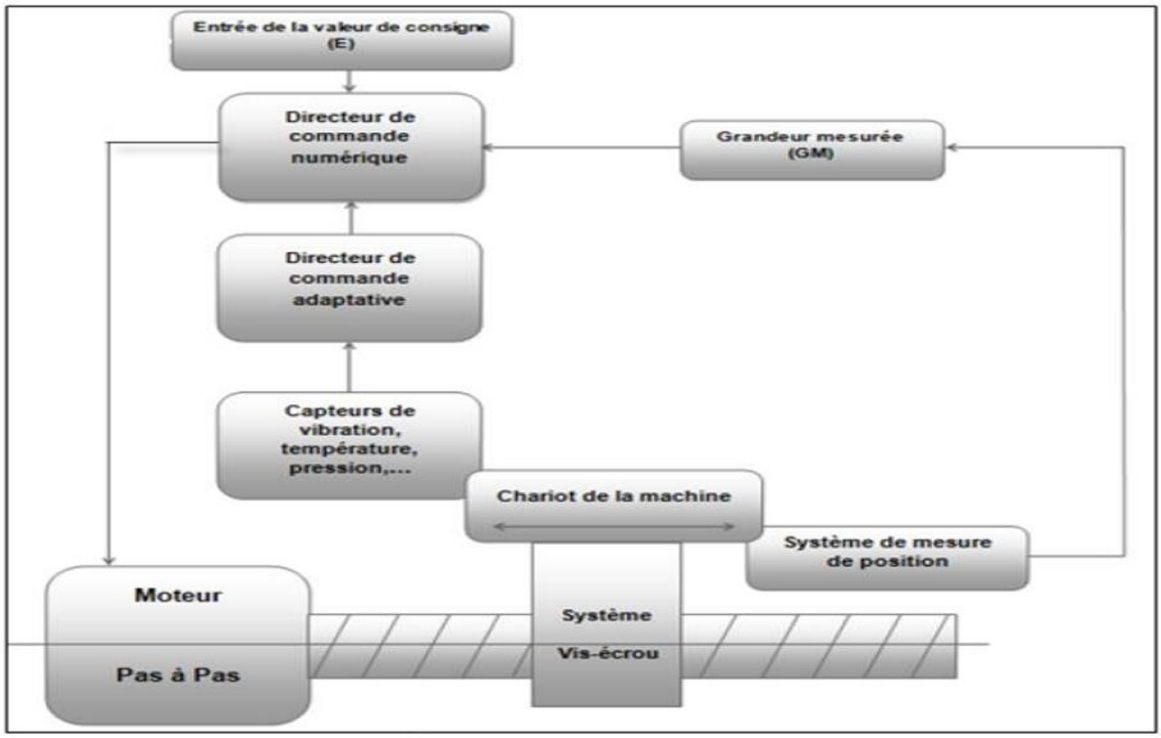


Figure 4 : Fonctionnement avec commande adaptative. [4]

1.2.6.2. Classification des CNC selon le mode d'usinage

Selon le mode d'usinage on peut classer les MOCN en trois catégories :

- Commande numérique par axiale (Tournage, fraisage, alésage.)
- Commande numérique de contournée (Toute opération possible sur un centre de tournage, ou centre d'usinage).

1.3.Les moteurs pas à pas

Le moteur pas à pas est un moteur électrique qui converti des impulsions électrique en mouvement de rotation avec nombre de pas qui corresponde au nombre des impulsions de commande. Le moteur pas à pas a été inventé par Marius Lavet en 1936. Mis au point pour la fabrication de montres, on le retrouve dans presque tous les modèles à quartz. Le moteur pas à pas est également utilisé dans le secteur de l'informatique, notamment dans les imprimantes reliées à un ordinateur et le lecteur DVD. Il est également utilisé dans le domaine des caméras robotiques et de surveillance.

Les moteurs pas à pas sont des moteurs spéciaux, composés simplement d'un stator réunissent des pièces polaires et des bobinages, et utilisés pour commander avec grande précision le déplacement et la position d'un objet [5]. Comme leur nom l'indique, ces moteurs tournent par

incrément discret. Chaque incrément de rotation est provoqué par une impulsion de courant fournie à l'un des enroulements du stator. Le moteur pas à pas est l'organe de positionnement et de vitesse travaillant généralement en boucle ouverte [5]. Le principe de base est donc toujours la création d'un champ tournant comme dans les moteurs triphasés industriels ou dans les petits moteurs équipant les programmateurs mécaniques : les pôles magnétiques de rotation de même nom se repoussent et les pôles des noms contraires s'attirent, le champ magnétique entraînera le rotor alimenté dans le même sens. Ceci traduit le fait qu'on transforme une grandeur numérique en une grandeur analogique. La fréquence de rotation, ou vitesse est donc commandée par des impulsions (consigne de rotation) contrôlées elle-même par un dispositif électronique en technologie câblée programmée [5].

1.3.1. Les différents types des moteurs pas à pas

Il existe Trois catégories de moteurs :

- Les moteurs à aimant permanent
- Les moteurs à réluctance variable
- Les moteurs hybrides

a) Le moteur à réluctance variable

Le moteur à réluctance variable fonctionne selon un principe différent de celui des moteurs à aiment permanent. Ils possèdent bien évidemment un stator, mais ce dernier est fabriqué en acier doux non magnétique. Il n'est pas lisse et possède plusieurs dents. Ce type de moteur est représenté en (figure 5). On peut voir, dans cet exemple, que le stator est composé de 8 plots sur lesquels enroulés les bobinages, ce qui donne 4 phase. Le rotor, quant à lui, ne comporte que 6 dents [6].

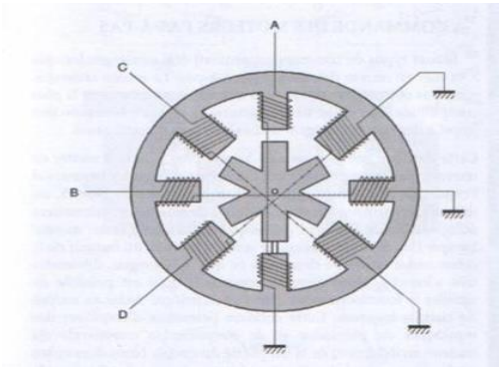


Figure 5 : Moteur à réluctance variable [7].

- **Le principe de fonctionnement :**

Le principe de fonctionnement est très simple, ces moteurs sont composés d'un barreau de fer doux et d'un certain nombre de bobines. Lorsqu'on alimente une bobine, elle devient un électroaimant et le barreau de fer cherche naturellement à s'orienter suivant le champ magnétique. On alimente la phase 1, puis la phase 2, puis la phase 3, puis phase 4. Si on veut changer le sens du moteur, il suffit de changer l'ordre d'alimentation des bobines. Dans la pratique, le barreau de ferrite a plusieurs dents (dans notre exemple est 6). Dès qu'on alimente la phase 2, il y a une rotation de 15° (i.e. 60° - 45° = 15°), puis la phase 3, etc. Donc le moteur tourne de 15° dès qu'on alimente une phase [8]. Il faut 24 impulsions pour faire un tour complet. C'est un moteur 24 pas par tour. Comme le montre figure suivant :

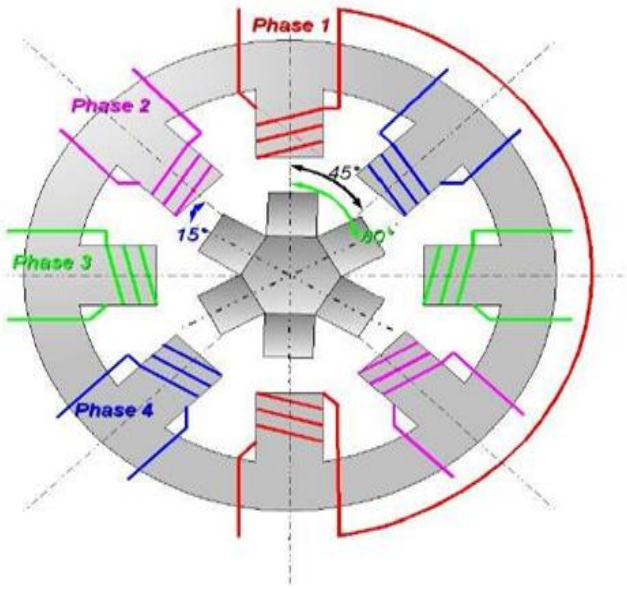


Figure 6 : Principe de fonctionnement moteur à réluctance variable. [8]

Le mode de commande peut dans ce cas être :

* Mode à pas entier, une phase alimentée à la fois.
* Mode à pas entier, deux phases alimentées en même temps.
* Mode demi-pas.

Les séquences de commandes sont présentées dans le tableau suivant :

Tableau 1 : Le mode de commande moteur réluctance variable.

Mode pas entier et (Une phase On)	**Mode pas entier et (Deux phases On en même temps.)**	**Mode demi-pas**
Ph1 On Ph2 On Ph3 On Ph4 On	Ph1 On et Ph 3 On Ph2 On et Ph 3 On Ph2 On et Ph 4 On Ph1 On et Ph 4 On	Ph1 On et Ph 3 On Ph 3 On Ph2 On et Ph 4 On Ph 4 On Ph 1 On et Ph 4 On

b) Le moteur à aimants permanent

Les moteurs à aimants permanents sont semblables aux moteurs à réluctance variable, sauf que le rotor possède deux pôles NORD et SUD. À cause des aimants permanents, le rotor reste freiné à sa dernière position lorsque le bloc d'alimentation cesse de fournir des impulsions. Une façon simple de voir le système, est de placer une boussole entre deux aimants. Suivant la bobine qui est alimentée et le sens du courant, l'aiment va s'aligner avec le champ.

b.1) Moteur à aimant permanent bipolaire

Le courant de commande est bidirectionnel et l'avance d'un pas s'effectue par une séquence de commutation des enroulements statorique [9].

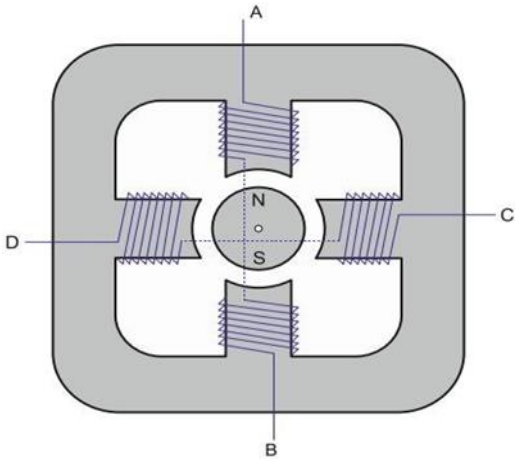


Figure 7 : Représentation schématique d'un moteur bipolaire : commande à trois modes [7]

b.1.1) Fonctionnement à pas complet

Dans ce mode de fonctionnement, les bobines sont alimentées l'une après l'autre dans un sens puis dans l'autre. L'aimant permanent suit le déplacement du champ magnétique créé par ces bobines et s'oriente selon une de ses 4 positions stables [10].

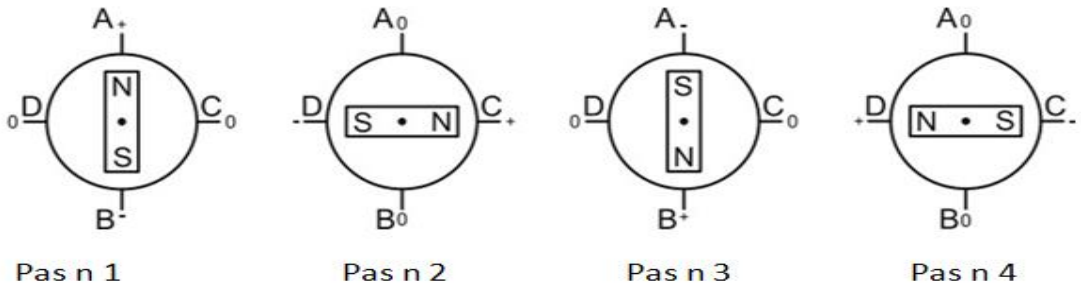


Figure 8 : Fonctionnement à pas complèt. [7].

b.1.2) Fonctionnement avec couple maximal

Dans ce mode de fonctionnement, alimenter une paire de phase en même temps et donc augmentera l'intensité du flux magnétique créé par le stator et le couple moteur, Le rotor prendra donc également l'une des 4 positions de figure suivant, suivant le sens d'alimentation de chacune des bobines.

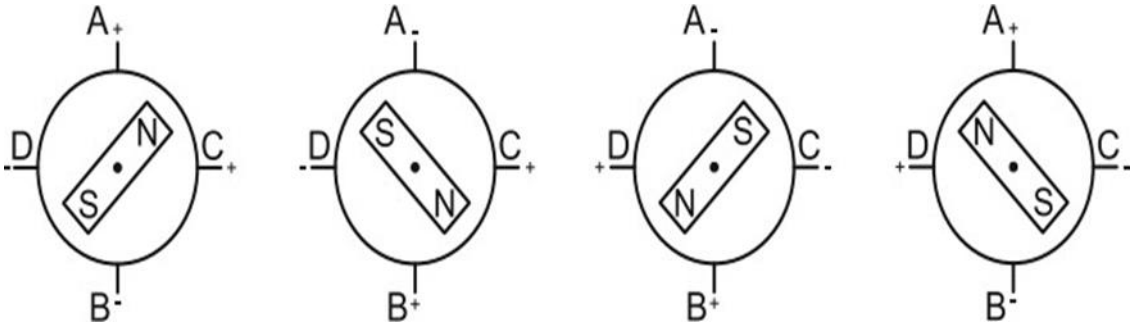


Figure 9 : Fonctionnement avec couple maximal [7].

b.1.3) Fonctionnement à demi-pas

Le troisième mode est combine les 2 modes précédent alimenter une phase puis deux puis une, est donc augmente le nombre de positions stables et le nombre de pas par tour ce mode de commande est appelée «demi-pas». La figure suivante montre le fonctionnement de ce mode.

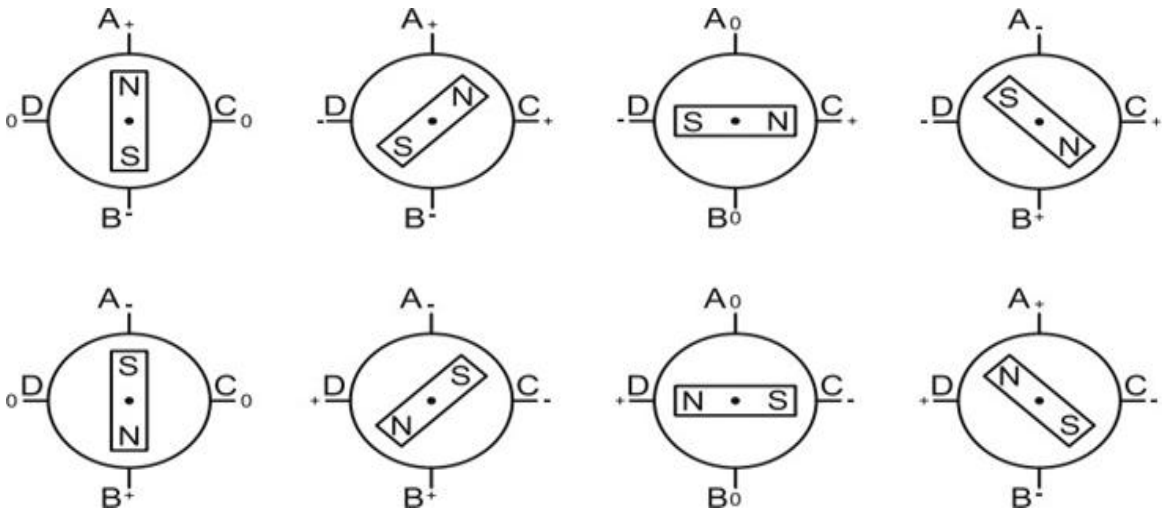


Figure 10 : Fonctionnement à demi-pas [7].

b.2) Moteur à aimant permanent unipolaire

Les bobinages d'un moteur unipolaire sont alimentés toujours dans le même sens par une tension unique d'où le nom d'unipolaire. Représenté par la figure suivant [11].

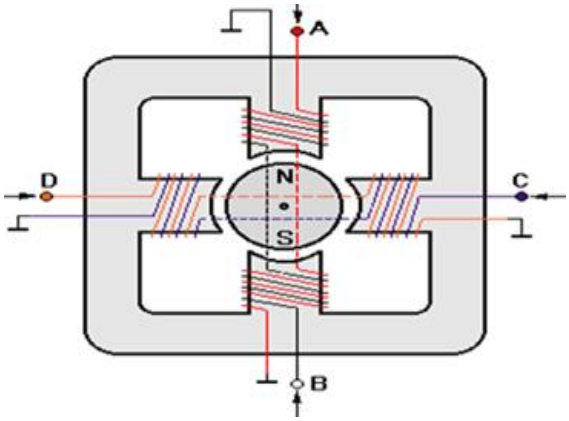


Figure 11 : Représentation schématique d'un moteur unipolaire.

b.2.1) Fonctionnement d'un moteur pas-a-pas unipolaire

On partira du principe que la rotation d'un moteur pas à pas s'effectue en 4 étapes, dans la réalité, le moteur est constitué d'une succession d'alternance de pôles : ainsi, l'axe du modèle dont nous disposons dans notre réalisation fait un tour complet en 48 pas (un pas correspond donc à 360°/48 = 7,5°) [5]

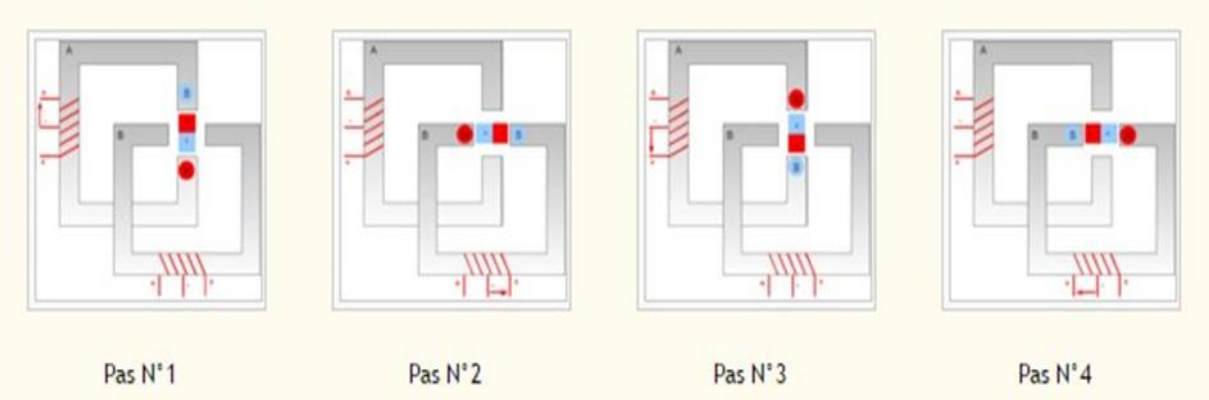


Figure 12 : Fonctionnement d'un moteur pas-a-pas unipolaire.

c) Moteur hybride

En mélange les structures des deux moteurs précédents, c'est à dire en plaçant les aimants du moteur à aimants permanents dans un circuit ferromagnétique on crée un nouveau type de moteur appelé moteur hybride. Dans ce cas, il existe un couple réluctant provoqué par la variation de perméances propres associées à chaque aimant et à chaque bobine.

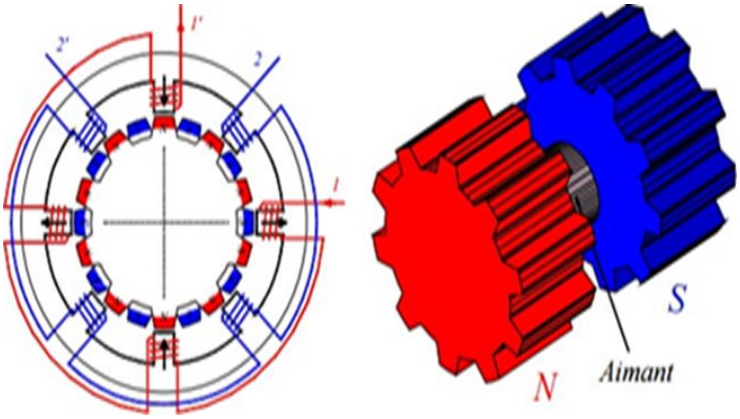


Figure 13 : Moteur hybride.

1.3.2. Comparaison des 3 types de moteurs

Le tableau ci-dessous présente les avantages de l'un par rapport à l'autre :

Tableau 2 : Comparaison entre les différents types des moteurs pas à pas.

Type De moteur	**Moteur a aimant permanent**	**Moteur a réluctance variable**	**Moteur hybride**
Résolution (nb de pas/tour**)**	Moyenne	Bonne	Elevée
Sens de rotation	dépend : - du sens du courant pour les moteurs bipolaires - L'ordre d'alimentation des bobines pour les moteurs unipolaires	dépend : - uniquement de l'ordre d'alimentation des bobines	dépend : - du sens du courant pour les moteurs bipolaires - L'ordre d'alimentation des bobines pour les moteurs unipolaires
Fréquence de travail	Faible	Grande	Grande
Couple de travaille	Elevé	Faible	Elevé

1.3.3. Les Avantages et les inconvénients de moteur pas à pas

Avantage :

- ✓ Sa facilité de mise en œuvre pour avoir un système précis et répétable.
- ✓ le moteur pas à pas ne nécessite pas d'entretien.
- ✓ Contrôle de la position, de la vitesse et synchronisation de plusieurs moteurs (de besoin de contre-réaction).
- ✓ Moteur sans balais.

Inconvénients :

- ✓ Vitesse et couple relativement faible.
- ✓ Couple décroissant rapidement lorsque la vitesse augmente.

1.4. Conclusion

Dans ce chapitre les machines CNC et les moteurs pas à pas sont exposés d'une manier générale. Cependant le problème qui se présente maintenant est comment commander les moteurs pas à pas autrement dit comment peut- on faire varier le sens de rotation du moteur pas à pas, le nombre de pas, et comment utiliser ces moteurs pour contrôler la machine CNC. Les réponses à ces questions seront détaillées dans le chapitre suivant.

Chapitre 2

Partie électronique

2.1. Introduction

La machine CNC peut être divisée en trois parties. Le système mécanique qui reçoit des signaux de commande nécessaire du système électronique qui permet finalement à actionnement souhaité des moteurs. Les systèmes électroniques obtiennent une commande ou un ensemble de commandes du système logiciel et génère des commandes pour le système mécanique. Les principes et les généralités sur des machines à commande numérique et les moteurs pas à pas avaient été décrit dans le chapitre précèdent. Dans ce chapitre nous allons détailler le système électronique qui est responsable de la génération du signal de commande pour les moteurs qui guide le mouvement de la trajectoire de l'outil dans chaque direction ou axe.

Le système électronique est composé de :

- L'alimentation
- La carte microcontrôleur
- La carte de commande des moteurs pas à pas

Nous allons décrire les composants électroniques utilisés dans notre projet à savoir le microcontrôleur MSP430G2553 et le driver A4988 pour la commande de moteur et nous montrons comment brancher ces composants ensemble pour pouvoir réaliser notre commande de machine.

2.2.Microcontrôleur

Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur

- **La mémoire morte** :(ROM = Read Only Memory) qui contient les instructions. Son contenu est permanent, et reste intact lorsque le système n'est plus alimenté, contient généralement d'un à quelques centaines de kilooctets.
- **Le processeur :** il interprète les instructions et traite les données d'un programme. est cadencé à des fréquences de quelques mégahertz ou dizaines de mégahertz. Il ne consomme généralement qu'une fraction de watt. Son jeu d'instructions est plus simple.
- **La mémoire vive** : (mémoire vive RAM) de la mémoire pour stocker les variables durant l'exécution du programme. est généralement très limitée : de quelques centaines d'octets (ou en anglais byte, abrégé "B") à quelques dizaines de kilooctets selon les modèles.
- **Les circuits d'entrée-sortie** sont simplement des entrées logiques, pour lire une valeur binaire, comme par exemple un interrupteur, ainsi que des sorties logiques, capables de fournir quelques milliampères, par exemple pour commander une LED.

L'intérêt des microcontrôleurs réside dans leur :

- plus haut degré d'intégration,
- Plus faible consommation électrique.

Bien que des microcontrôleurs existent depuis les années 1970, ils se sont développés de manière spectaculaire depuis quelques années. Les microcontrôleurs actuels contiennent de la mémoire flash, qui facilite l'écriture du programme qui doit s'exécuter. Ils sont principalement conçus pour des applications embarquées et sont utilisés dans des dispositifs à commande automatique, tels que des dispositifs médicaux implantables, des outils électriques, des jouets, des machines de bureau et d'autres systèmes embarqués.

Au niveau du marché, de nombreux microcontrôleurs sont proposés par les fabricants (Microchip-Atmel, Texas Instrument, NXP,). Chaque fabricant offre souvent plusieurs familles de microcontrôleurs et chaque famille comporte souvent des dizaines, voire des centaines de modèles qui diffèrent les uns des autres principalement par les tailles mémoires (RAM et ROM) et le nombre de broches d'entrée-sortie.

Dans notre projet, nous avons utilisé le microcontrôleur MSP430G2553 de Texas Instruments, notre choix est basé sur sa faible consommation d'énergie et en particulier à son faible coût.

2.2.1. Le microcontrôleur MSP430G2553

Les microcontrôleurs de la famille MSP430G2553 de Texas Instruments sont disponibles dans plusieurs boîtiers, dont le DIL20 (Dual-In-Line 20 pins), facile à mettre en œuvre. Le MSP430G2253 contient :

- **Un processeur** 16 bits, avec une centaine d'instructions, cadencé au maximum à 16 MHz, avec 16 registres de 16 bits.
- **Une mémoire morte** de type flash de 16 ko. Cette mémoire peut être effacée et écrite un très grand nombre de fois.
- **Une mémoire vive** de 512 oct.
- **16 broches d'entrée-sorties**. Huit d'entre-elles peuvent être connectées à un convertisseur analogique-numérique de 10 bits de résolution. Certaines de ces broches ont également d'autres fonctions spécifiques (génération de signaux, capture d'événements, utilisation d'un quartz, etc.).

Les MSP430 acceptent une tension comprise entre 1,8 et 3,6 V et caractérisé par :

- Ultra faible consommation d'énergie :
    - Mode actif : 230 µA à 1 MHz et 2,2 V.
    - Mode veille : 0,5 µA.
    - Mode "arrêt" : 0,1 µA.
- Architecture 16 bits RISC.
- Fréquences internes jusqu'à 16 MHz avec quatre fréquences calibrées.
- Oscillateur interne en Crystal à puissance très-faible et à fréquence basse 32 kHz.
- Source d'horloge numérique externe.

Le composant est décliné en une série de configurations comprenant les périphériques usuels :

- convertisseurs A/D 10/12/14/16 bits,
- convertisseurs D/A 12 bits,
- interfaces UART, SPI, I2C
- pilote LCD
- Watchdog
- multiplicateur hardware
- oscillateur interne,….

2.2.1.1. Les brochages de MSP430G2553 :

Chaque broche est désignée par un ou plusieurs noms (Fig 14). Trois groupes de broches sont identifiés pour la réalisation du montage fonctionnel de microcontrôleur :

- Les alimentations : Gnd et +3 V.
- Les broches de programmation : Rst et TESTl.
- Les broches d'entrées-sorties pour l'application proprement dite, c'est-à-dire toutes les autres broches.

A. Les alimentations :

Les microcontrôleurs sont presque toujours réalisés en technologie CMOS et nécessitent une alimentation unique. La borne négative de l'alimentation est souvent désignée par ***Gnd*** (Ground : masse). C'est la broche 20 du MSP430G2553. La borne positive est appelée DVCC c'est la broche 1[12].

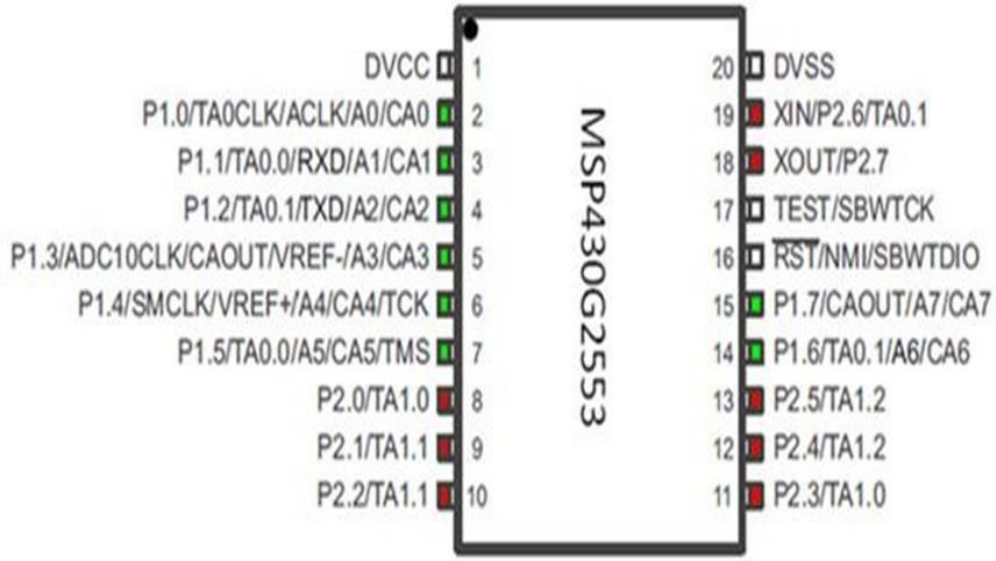


Figure 14 : Brochages de MSP430G2553.

B. Les broches de programmation :

La fonction Rst ou Reset est câblée sur la broche 16. Le Rst est une entrée du circuit. Notez la barre sur son nom : cela signifie que cette entrée est active à 0. On le note aussi parfois avec le signe barre oblique (ou slash) / précédant le nom : /Rst (Fig 2.1). C'est une bonne habitude de mettre en fonctionnement normal du microcontrôleur. C'est une résistance reliée au +3 V qui joue ce rôle. On parle de résistance de tirage (en anglais pull-up : tirer vers le haut). Une valeur d'environ 47 kΩ convient dans ce cas. La programmation des MSP430 peut se faire par l'intermédiaire de deux signaux : le RST et un signal nommé TEST (broche 17) [12].

C. Les broches d'entrée-sortie :

C'est-à-dire toutes les autres broches sauf les broches d'alimentations et de programmation. Elles sont regroupées en 2 ports : P1 et P2. Ces deux ports sont complets : ils comportent chacun 8 broches [12].

2.2.1.2. Carte de développement MSP430 LaunchPad :

Pour pouvoir programmer le MSP430, nous avons utilisé LaunchPad MSP-EXP430G2. C'est un outil de débogage et de programmation Flash facile à utiliser mettant à la disposition de l'utilisateur tous les éléments dont il a besoin pour procéder au développement de dispositifs de la gamme MSP430G2XX. Il comporte une carte cible à prises DIP avec émulation intégrée pour programmer et déboguer rapidement les MSP430 au sein du système par le biais du protocole Spy Bi-Wire (JTAG à 2 fils).

Le LaunchPad MSP430 comprend :

1 Support de processeur DIP à 20 broches.

2 Emulation Flash intégrée pour programmation et débogage.

3 Deux LED utilisateur.

4 LED d'alimentation

5 Bouton utilisateur.

6 Bouton Reset.

7 Une embase à 10 broches pour une connexion à un circuit externe.

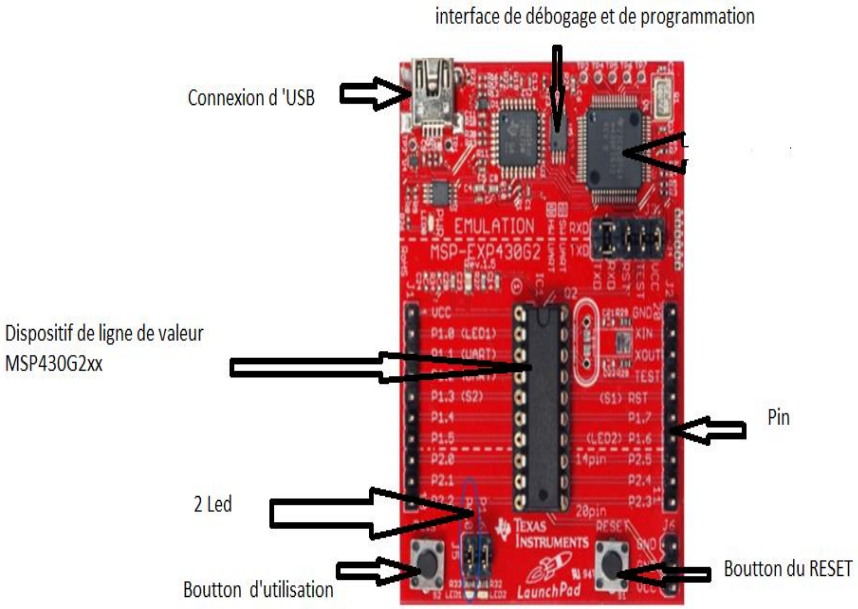


Figure 15 : Carte de développement MSP430G2553 LaunchPad

2.2.1.3. Programmations MSP430G2553

La mémoire Flash peut être effacée et programmée en quelques secondes sans aucune alimentation externe grâce à la technologie Flash à ultra-faible consommation du MSP430. Le LaunchPad sert d'interface entre les circuits MSP430 et un environnement logiciel intégré tel que Code Composer Studio version 4 ou IAR Embedded Workbench. . Ces IDE sont gratuits et illimités sur les systèmes MSP430GEXX.

Pour la programmation de notre MSP430G2553 nous avons opté pour IDE IAR embedded workbench en utilisant le langage C. Une fois le programme compilé, le fichier binaire est envoyé au microcontrôleur via le câble USB qui connecte le Lunchpad à notre PC.

- **Présentation de langage C**

C'est un langage de programmation impératif généraliste, de bas niveau. Inventé au début des années 1970 pour réécrire UNIX. C'est devenu un des langages les plus utilisés, encore de nos jours. De nombreux langages plus modernes comme C++, C#, Java et PHP ou Javascript ont repris une syntaxe similaire au C et reprennent en partie sa logique. C offre au développeur une marge de contrôle importante sur la machine (notamment sur la gestion de la mémoire) et est de ce fait utilisé pour réaliser les « fondations » (compilateurs, interpréteurs…) de ces langages plus modernes [13].

Pour nous faciliter la programmation nous avons élaboré un Template en langage C qui à la structure suivante :

1- inclusion de fichiers et bibliothéque:

```
#include <fichier1.h>
#include <bibliothèque1 .h>
```

2- déclarations des variables sous différents types, syntaxe utilisée est la suivante :

```
char  val1;
 int   val2;
float  val3;
```

3- déclarations de toutes les fonctions (sous-programme) utilisées :

```
void fonction1(void) ;
void fonction2(void) ;
```

3- programme principal : cette partie est utilisée par le système pour exécuter le programme, syntaxe utilisée est la suivante :

```
 void main(void)
{
  Apelle fonction1() ;
 Apelle fonction2 () ;
}
```

4- détail des fonctions déclarées précédemment, syntaxe utilisée est la suivante :

```
void fonction1(void)
{
    …
    …
}
void fonction2(void)
{
  …
  …
}
```

2.2.1.4. Les Modules de msp430G2553

A) Les entrées-sorties programmables (GPIO)

GPIO signifie broche d'entrée/sortie à usage général (ou simplement «IO»). Chacune des petites broches métalliques d'une puce de microcontrôleur a un objectif. Elle peut avoir un seul objectif dédié comme connecter la puce à l'alimentation ou à la terre, ou il peut s'agir d'une broche GPIO que le programmeur peut configurer de différentes manières. Les GPIO peuvent être configurés comme entrées ou sorties :

* **En entrée :** une broche GPIO indique au microcontrôleur quelle tension est présente sur la broche. Par exemple, une configuration d'entrée GPIO peut être utilisée pour indiquer au microcontrôleur si un bouton est enfoncé.
* **En sortie :** le microcontrôleur choisit de régler la broche GPIO pour émettre une tension haute ou basse.

MSP430G2253 dispose de deux ports GPIO, chaque port contient 8 broches numéroté comme suit : Broches du port 1 P1.0 à P1.7 et broches du port 2 P2.0 à P2.7. La plupart des broches du MSP430 fonctionnent comme des broches GPIO, avec la possibilité de fonctionner comme broche spécialisée dans la bonne configuration. En tant que broches GPIO, chaque broche est indépendamment.

A.1) Les Registres GPIO de MSP430G2553 :

La série MSP430x2xx possède neuf bits pour contrôler chaque broche GPIO, chaque bit correspond à un registre de contrôle GPIO. Chaque registre à une largeur de 8 bits et chaque bit du registre correspond à une broche GPIO. Ainsi, le bit 0 de P1IN correspond à la broche 0 du port 1, le bit 1 correspond à la broche 1 du port 1, et ainsi de suite jusqu'au bit 7 correspond à la broche 7 du port P1.

Chaque port se voit attribuer plusieurs registres 8 bits qui contrôlent la fonction des broches et fournissent des informations sur leur état actuel. Voici la liste des registres GPIO MSP430 et leurs opérations spécifiques :

- **PxDIR Direction Registre :** Le registre PxDIR sélectionne la direction de la broche d'E / S, que ce soit une entrée ou une sortie. Ceci indépendamment de la fonction sélectionnée de la broche.
    - Bit = 0 : la broche du port est commutée dans le sens d'entrée,
    - Bit = 1: la broche du port est commutée dans le sens de sortie.

❖ **PxOUT Registre de sortie :** Le registre PxOUT détermine la valeur de sortie vers la broche GPIO, lorsque la broche est configurée en tant que fonction d'E / S, la direction de sortie et la résistance pull up / down est désactivée.

- Bit = 0 : la sortie est faible.
- Bit = 1 : la sortie est haute.
- Si la résistance pull up / pull down de la broche est activée, le bit correspondant dans le registre PxOUT sélectionne pull up ou pull down.
- Bit = 0 : la broche est tirée vers le bas.
- Bit = 1 : la broche est tirée vers le haut.

* **PxIN Registre d'entrée :** Le registre PxIN reflète la valeur du signal entrant dans la broche GPIO, lorsqu'il est configuré en tant que fonction d'E / S. Ainsi, en lisant cette valeur, vous pouvez déterminer s'il y a une 0 logique ou une 1 logique sur l'entrée.

- Bit = 0 : l'entrée est faible
- Bit = 1 : l'entrée est élevée

* **PxREN Registre de résistance Pull up / Pull down :**

Le registre PxREN active ou désactive la résistance de pull-up / pull -down interne, qui correspond à la broche d'E / S individuelle.

- Bit = 0: résistance pull-up / pull-down désactivée.
- Bit = 1: résistance Pull-up / Pull-down activée.

- **Résistance Pull Up :**

Bouton poussé -> le bit dans le registre d'entrée (PxIN) est faible (0).

Bouton relâché -> le bit sélectionné dans le registre PxIN est élevé (1).

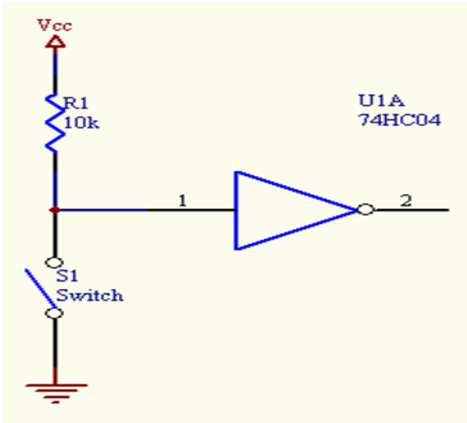


Figure 16 : Schémas des résistances Pull up.

- **Résistance Pull Down :**

- Bouton poussé -> le bit dans le registre d'entrée (PxIN) est élevé (1).
- Bouton relâché -> le bit sélectionné dans le registre PxIN est bas (0).

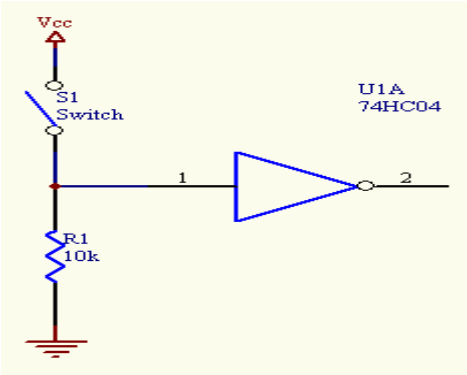


Figure 17 : Schémas des résistances Pull Down.

Afin de sélectionner la résistance Pull-up ou Pull-down, nous devons utiliser en conjonction avec le registre PxREN, le registre PxOUT comme suit :

Lorsque la résistance Pullup / pulldown est activée (PxREN = 1) :

- ✓ **PxOUT = 0** : la broche est tirée vers le bas.
- ✓ **PxOUT = 1** : la broche est tirée vers le haut.

1. **Registres de sélection de fonction PxSEL et PxSEL2 :**

Les registres **PxSEL** et **PxSEL2** permettent d'associer les broches GPIO individuelles aux fonctions du module périphérique interne, ou simplement de les laisser comme ports d'E/S standard.

Tableau 3 : Configuration de PxSEL et PxSEL2

PxSEL2	PxSEL	Fonction de la broche
0	0	La fonction est sélectionnée I/O
0	1	La fonction du module périphérique principal est sélectionnée
1	0	Réservé (voir fiche technique spécifique de msp430G2553)
1	1	La fonction du module périphérique secondaire est sélectionnée

2. les registres PxIE et PxIES et PxIFG :

Ces registres seront détaillés dans la partie des interruptions.

Afin de se familiariser avec le msp430 nous allons allumer la LED vert connecté au pin P1.0 et la LED rouge connecter au pin P1.6 de msp430 lorsque le bouton s2 (P1.3) est enfoncé.

```
//=========================includesblock=======================//
#include <msp430.h>
//============================ mainblock=======================//
void main(void) {
//============================initializing   registers===============//
WDTCTL = WDTPW | WDTHOLD;    // Stop watchdog timer
 P1DIR &=~ 0x08;   //set P1.3 to input--
 P1DIR|= 0X41 ;      // set the red led and green pin to output direction
 P1REN |= 0X08 ;   //resistor enabled
 P1OUT |= 0X08 ;   // pull up resistor selected
  //========================== infinite loop =======================//
  while(1){
if((P1IN & BIT3)==0){        //  if the button is pushed, enter
P1OUT |=0X41 ;                  // turn on the LED green and red
__delay_cycles(500000);     // dealy 0.5 s
P1OUT &=~ 0x41 ;            //   turn off the LED green and red
}
}
}
```

Les étapes suivies dans ce programme sont :

- ✓ inclusion de la bibliothèque msp430,

- ✓ arrêt du timer du chien de garde,
- ✓ initialisation des registres utilisés,
- ✓ dans la boucle nous créons notre scénario : lorsque nous appuyons sur le bouton, la LED rouge et la LED VERT s'allume en même temps pendant une demi-seconde. Vu que nous utilisons la résistance de tirage vers le haut, la valeur dans le registre P1IN est 1 lorsque le bouton est relâché, sinon elle est 0, ce qui explique notre utilisation de la condition : if ((P1IN & BIT3) == 0).

B) Les interruptions

C'est un signal qui informe notre microcontrôleur qu'un certain événement s'est produit, provoquant l'interruption du flux normal du programme principal et l'exécution d'une "routine d'interruption", qui gère l'événement et prend une action spécifiée. Prenons un exemple : Je suis en train de travailler à mon bureau. Le téléphone sonne. Je vais répondre au téléphone. Après la conversation, je reprends mon travail là où je l'avais laissé.

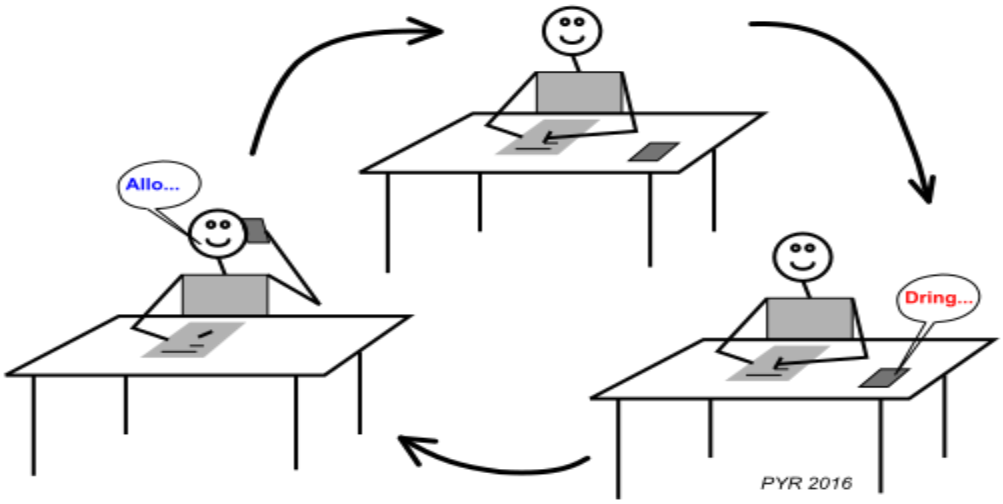


Figure 18 : Exemple d'Interruption dans la vie courante [15]

La principale caractéristique du MSP430 est sa faible consommation d'énergie en utilisant des modes de faible consommation, ce qui nous oblige à apprendre comment le mettre en veille et comment le réveiller. C'est l'une des principales raisons pour lesquelles nous devrions connaître les interruptions dans msp430.

Les événements qui vont produire une interruption sont soit de nature extérieurs au microcontrôleur, par exemple changement sur une entrée, ou peuvent être de nature intérieurs. Dans la catégorie des interruptions intérieures au microcontrôleur, les plus importantes sont celle liées aux Timers.

B.1) Les sources d'interruption [15]

Il existe généralement plusieurs sources interruptions sur un microcontrôleur. Lorsqu'une interruption se produit, le système doit être capable d'en savoir la source. Si rien n'est prévu au niveau matériel, la routine d'interruption doit consulter les registres pour chaque interruption, pour connaître celle qui a été activée.

Les vecteurs d'interruption (*interrupt vectors*) permettent d'être plus efficace : une adresse différente est réservée pour le début de la routine de chaque interruption. Souvent ces deux mécanismes vont être utilisés successivement, comme nous le verrons plus tard lors d'une interruption produite par une entrée sur un MSP430.

Voici la table résumée des vecteurs d'interruption pour un MSP430G2553, y compris l'adresse pour le Reset :

- **0xFFFE** : Reset
- **0xFFFC** : NMI
- **0xFFFA** : Timer1 CCR0
- **0xFFF8** : Timer1 CCR1, CCR2, TAIFG
- **0xFFF6** : Comparator_A
- **0xFFF4** : Watchdog Timer
- **0xFFF2** : Timer0 CCR0
- **0xFFF0** : Timer0 CCR1, CCR2, TAIFG
- **0xFFEE** : USCI status
- **0xFFEC** : USCI receive/transmit
- **0xFFEA** : ADC10
- **0xFFE6** : Port P2
- **0xFFE4** : Port P1

Les adresses se trouvent en mémoire flash, ce sont les dernières adresses de l'espace d'adressage de 16 bits.

Plusieurs sources d'interruptions nécessitent la scrutation pour déterminer la cause exacte de l'interruption. C'est le cas par exemple des interruptions sur les ports : chaque bit peut produire une interruption.

B.2) Mise en œuvre d'une interruption

Plusieurs étapes sont nécessaires pour mettre en œuvre une routine d'interruption :

- Enclencher l'interruption qui nous intéresse. Par exemple une interruption sur une entrée.
- Préciser comment cette interruption doit fonctionner. Par exemple dire sur quel flanc l'interruption doit se produire.
- Enclencher globalement les interruptions. Les microprocesseurs disposent d'un fanion général qui autorise les interruptions. Il est souvent utilisé pour pouvoir éviter les interruptions dans certaines parties critiques au programme, qui ne doivent pas être interrompues.

Le schéma logique ci-dessous montre la logique qui permet de générer les interruptions et les fanions qu'il faut ajuster.

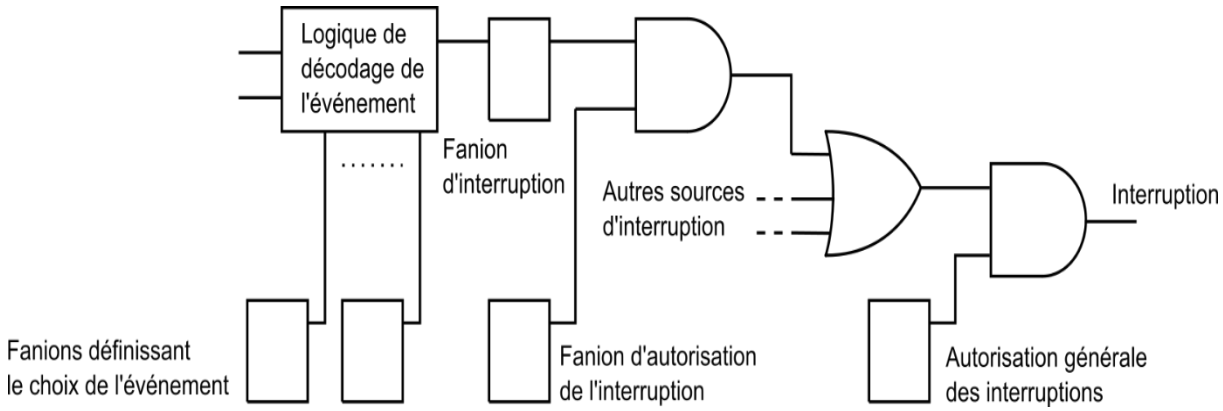


Figure 19 : Logique de génération des interruptions [15]

On y trouve :

A. la logique qui permet de saisir un événement.

B. les fanions qui règlent la manière dont l'événement est décodé.

C. le fanion qui enclenche cette interruption particulière.

D. la porte ET associée à ce fanion pour produire cette interruption.

E. la porte OU qui permet à toutes les interruptions d'être prises en compte

F. le fanion général d'autorisation des interruptions

G. la porte ET qui produit finalement les interruptions.

B.3) SYNTAXE DES ROUTINES D'INTERRUPTIONS

Le langage C ne définit pas la syntaxe des routines d'interruptions. Plusieurs notations sont utilisées, dépendant des compilateurs. Nous présenterons ici une des syntaxes supportées par les compilateurs [15].

```
#pragma vector=NUMERO_DU_VECTEUR
__interrupt void Nom_de_la_routine (void) {
 ...
}
```

La première ligne indique au compilateur à quel vecteur d'interruption la routine sera associée. La seconde ligne est une déclaration de procédure presque habituelle, avec un nom et aucun paramètre d'entrée (void). L'indication ***__interrupt*** permet au compilateur d'utiliser les instructions correspondant à une routine d'interruption, en particulier le ***Reti*** final (instruction de retour) [15].

B.4) Interruption produite par une entrée

Certains GPIO du MSP430 ont la capacité de générer une interruption et d'informer le CPU lorsqu'une transition s'est produite. Le MSP430 permet une flexibilité dans la configuration du GPIO qui générera l'interruption et sur quel front (montant ou descendant). Les registres contrôlant les interruptions sont les suivants :

- **Registres *P1IFG, P2IFG (Interrupt Flag)*** : Seuls les ports GPIO 1 et 2 ont une fonctionnalité d'interruption. Les registres ***P1IFG et P2IFG*** contiennent le drapeau (flag) d'interruption pour la broche d'*E/S* correspondante, le drapeau (flag) d'interruption est défini lorsque le front de signal d'entrée sélectionné se produit sur la broche :

- Bit = 0 : aucune interruption n'est en attente.
- Bit = 1 : une interruption est en attente.

- **Registres *P1IE, P2IE (Interrupt Enable) :*** Le bit de registre ***P1IE et P2IE*** active l'indicateur d'interruption ***PxIFG*** associé :
  - Bit = 0 : l'indicateur (flag) d'interruption est désactivé.
  - Bit = 1 : l'indicateur (flag) d'interruption est activé.
- **Registres *P1IES, P2IES (Interrupt Edge Select)*** : permet de choisir pour chaque bit le flanc qui va produire l'interruption :
  - Bit = 0 : l'indicateur PxIFG est défini avec une transition de bas en haut **(↑).**
  - Bit = 1 : l'indicateur PxIFG est défini avec une transition de haut en bas (↓**).**

Pour davantage de clarification, nous avons élaboré un programme qui fonctionne avec des interruptions, chaque fois que nous appuyons sur le bouton, une interruption est générée, nous basculons entre la LED vert et LED rouge pour le visualiser.

```
//=========================includes block=================//
#include <msp430.h>
//====================== main-block====================//
void main(void) {
//=========================initializingregisters==============//
  WDTCTL = WDTPW | WDTHOLD;    // Stop watchdog timer
  P1DIR &=~ 0x08;                        //set P1.3 to input
  P1DIR |= 0x41;                          // set the leds to output direction
  P1OUT &=~ 0x41;                        // turn off the Leds
   P1REN |=0x08;                      //resistor enabled for p1.3
    P1OUT |= 0x08;                         //   pull up resistor is selected-
     //==========interrupt configuration block=====================//
     P1IE |= 0x08;                        //enble interrupt for P1.3
     P1IES |=0x08;                      // edge select is falling
    P1IFG &=~ 0x08;                   //clear the flag
    __enable_interrupt() ;             //enable global interrupts//
 //=================infinite loop=========================//
While(1) {
 }
}
//=======interrupt service routine block============================//
 #pragma vector = PORT1_VECTOR                // Interrupt routine associated with Port
P1
  __interrupt void PORT_1 (void) {                // Interrupt routine associated with Port
P1
P1OUT ^=0x41 ;                                    // toggle the p1.0 and p1.6
_delay_cycles  (500000) ;                          //wait for 0.5s
P1IFG &=~ 0x08 ;                                  //clear the flag
}
```

Après l'inclusion de la bibliothèque MSP430 et l'arrêt du watchdog, les étapes suivies lors de la programmation sont :

- **La configuration des broches d'*E/S*** : réglage de la direction des broches, sélection de la résistance pull-up / pull-down

- **Bloc de configuration d'interruption** : dans la première ligne de celui-ci, dire au MCU d'écouter la broche **P1.3** pour les changements d'état logique (activant effectivement l'interruption sur cette broche particulière).
- Ensuite, nous sélectionnons le bord lorsque l'interruption est levée (de Haut à Bas ou Bas à Haut) ; N'oublions pas que le bouton du LaunchPad connecte la broche d'entrée à ***GND*** lorsqu'il est enfoncé et à ***VCC*** dans le cas contraire. Pour cette raison, nous sélectionnons le front ***Hi/Lo*** (front descendant).
- Enfin, nous effaçons l'indicateur (***flag***) d'interruption pour cette broche. Le registre d'indicateur d'interruption ***P1IFG*** signale lorsqu'une interruption est déclenchée, et il doit être effacé à la fin de la routine de service d'interruption.
- **boucle infinie :** il n'y a rien à faire dans la boucle principale
- **Interrupt** service routine block : C'est la routine de service d'interruption Port1. Chaque fois que nous appuyons sur le bouton P1.3, clignoter les LEDs (LED verte et rouge sur LaunchPad '0x41'),
- Effacez le drapeau (flag) d'interruption ***P1.3*** (très important) et nous revenons à l'exécution normale.

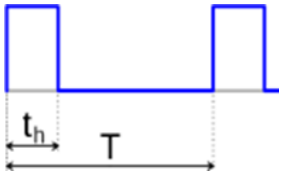
C) Modulation de largeur d'impulsions (PWM)

La modulation de largeur d'impulsions (MLI : en anglais Pulse Width Modulation, soit PWM), est une technique couramment utilisée pour synthétiser des signaux pseudo analogiques à l'aide de circuits à fonctionnement tout ou rien, ou plus généralement à états discrets. Elle sert à générer un signal pseudo analogique à partir d'un environnement numérique ou analogique pour permettre un traitement de ce signal par des composants en commutation (se comportant comme des interrupteurs ouverts ou fermés). Le principe général est qu'en appliquant une succession rapide d'états discrets avec des ratios de durée bien choisis, on peut obtenir en ne regardant que la valeur moyenne du signal n'importe quelle valeur intermédiaire [16].

C.1) Le rapport cyclique (Duty cycle)

Le rapport cyclique ou Duty cycle est la fraction d'une période pendant laquelle un signal ou un système est actif. Le cycle de service est généralement exprimé en pourcentage ou en rapport. Une période est le temps qu'il faut à un signal pour terminer un cycle marche-arrêt [17]. En tant que formule, le rapport cyclique (%) peut être exprimé comme suit :

$$a = 100 \times \frac{th}{T}$$



- a est le rapport cyclique
- T est la période du signal rectangulaire
- th est la durée d'une pulsation
- Si th =0 alors a = 0% et la tension moyenne de sortie est nulle.
- Si th =$T/2$ alors a =50% et la tension moyenne de sortie est égal a ***Vcc/2***.
- Si th =T alors a =100% et la tension moyenne de sortie est égal a ***Vcc***.

C. 3) PWM et MSP430G2553

La modulation de largeur d'impulsion est une méthode de génération d'un signal analogique à partir de ses homologues numériques. Le signal numérique de forme carrée est contrôlé en ajustant sa période de temps de marche/arrêt avec un rapport cyclique. Sur les MCU, cela se fait en utilisant des compteurs. Le timer (compteur) comptera jusqu'à une certaine valeur (puis reviendra à zéro, encore et encore), cette valeur est la période, et certains registres conserveront une autre valeur pour comparer avec le compteur, cette valeur contrôlera/basculera le haut (on) et l'a verrouillé jusqu'à ce que la comparaison suivante corresponde, cette autre valeur représente notre rapport cyclique.

Le MSP430G2253 dispose de deux timers 16 bits, timer0 et timer1 sont toutes deux **Timer_A**, qui a cinq blocs de capture/comparaison à l'intérieur du contrôleur. L'un quelconque des peut être utilisé pour capturer des données de timer ou générer des intervalles. Le mode de comparaison est utilisé pour générer des signaux de sortie PWM. Chaque bloc de capture/comparaison a une unité de sortie, qui est utilisée pour générer des signaux de sortie PWM.

Le Timer A est utilisé pour générer le signal PWM .Dans le Timer A, le TACCR0 est stocké pour la valeur de période totale de PWM et TACCR1 est utilisé pour stocker sur la période de (rapport cyclique).

Le registre de comptage du Timer_A (TAR) est en mode up et est incrémenté pour chaque horloge donnée par la source d'horloge sélectionnée. Le signal OUT1 du module de comparaison du Timer_ A est configuré sur une broche de sortie et la valeur de broche est démarrée avec la logique 1.

Le circuit de comparaison compare TAR et TACCR1, lorsque TAR atteint TACCR1, le signal OUT1 est changé en logique 0. Le TAR compte toujours et compare avec TACCR0. Lorsque 'il correspond, le signal OUT1 passe à la logique 1 et TAR est remis à 0.

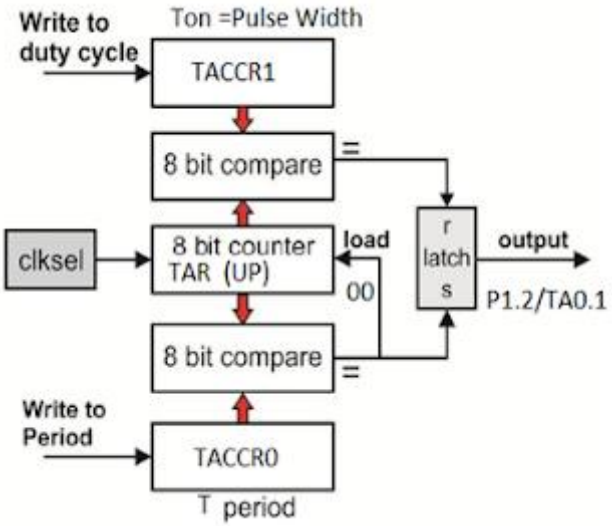


Figure 20 : Génération de signal en PWM dans MSP430 [18].

Remarque :

Le MSP430 est livré avec une configuration variée pour contrôler le comportement du signal PWM. y compris le mode de comptage : Up, continu, Up-Down et mode de sortie : Set, Reset, Toggle, Set/Reset, Reset/Set. Une référence complète est disponible sur le guide de la famille MSP430G2553.

Nous présentons en dessous notre exemple de programmation pour génère une sortie PWM sur P2.1 en utilisant Timer_A1

```
#include <msp430.h>

 WDTCTL = WDTPW + WDTHOLD;                    // stop watchdog

 DCOCTL = 0;                                  // Select lowest DCOx and MODx settings

BCSCTL1 = CALBC1_1MHZ;                        // Set DCO

DCOCTL = CALDCO_1MHZ;

P2DIR |= 0x02;                                // set step pin to output

P2SEL |= 0x02;                                // enable select register for PWM
```

```
TA1CCTL1 = OUTMOD_7;              //set PWM output mode to OUTMOD_7 (check datasheet)

TA1CTL = TASSEL_2 + MC_1;    // selct SMCLK(1mhz)and the count-mode  up mode,

For (int i=0 ; i<10 ;i++){

TA1CCR0 = 10000;                    // set PWM Period

 TA1CCR1 = TA1CCR0/2;             //set Duty cycle to 50%

 __delay_cycles(100000);

 TA1CCR0=0;                          //turne of PWM

}

}
```

Nous avons créé un signal PWM dans un délai d'ordre de 0.1s, avec une période 0.01 s (TA1CCR0 = 10000) et de rapport cyclique 50%. Lorsque ce délai expire, nous désactivons PWM (TA1CCR0=0) et on fait répéter ce scénario pour dix itération en utilisant la boucle for.

Le résultat obtenu est vérifié on le visualisant par une LED externe connectée sur la broche P2.1 de MSP430 :

- La LED est allumé à 50% de son intensité maximale. C'est-à-dire que la sortie P2.1 est la moitié de la valeur Vcc donnée au MCU ($V_{moy} = \frac{V_{cc}}{2}$).On remarque que lorsque on augmente la valeur de la période la LED clignote.
- lorsque nous mettons 0 dans le registre TA1CCR0, après la fin de délai du signale PWM nous remarquons que le PWM est bloqué mais la LED parfois reste allumé à 100 % de son intensité maximale. C'est-à-dire que la sortie à P2.1 égale la valeur de V_{cc} et parfois la LED s'éteint, cela signifie que la sortie à P2.1 égale 0.

Pour vérifier l'état de la LED allumée ou éteint après la fin de délai de signal PWM, nous avons fait plusieurs tests avec un délai fixe et à chaque fois on change la valeur de la période. On trace la forme de signal pour chaque test d'une itération, nous obtenons les résultats suivants :

Test 1 : la fin de délai de signale a coïncidé avec la dernière impulsion et ça nous a donné un single continue, ou la LED reste allumé.

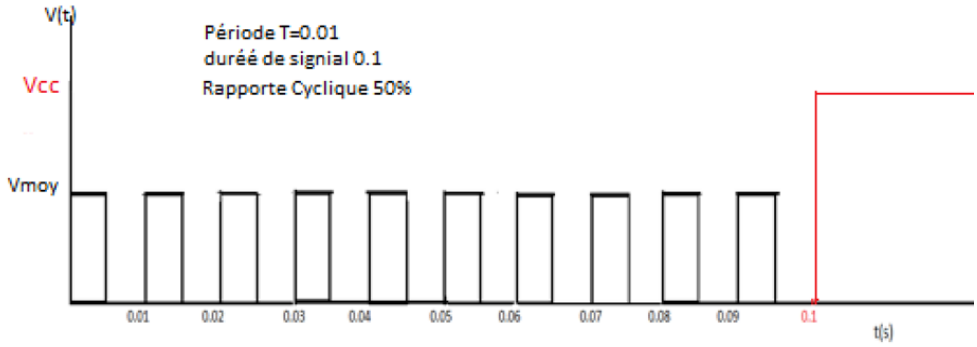


Figure 21 : Resultat du Test 1

Test 2 : le délai de signal s'est terminé pendant la phase on de la dernière impulsion, donc un single continue est engendré, et la LED reste éteint (la sortie égal 0 V)

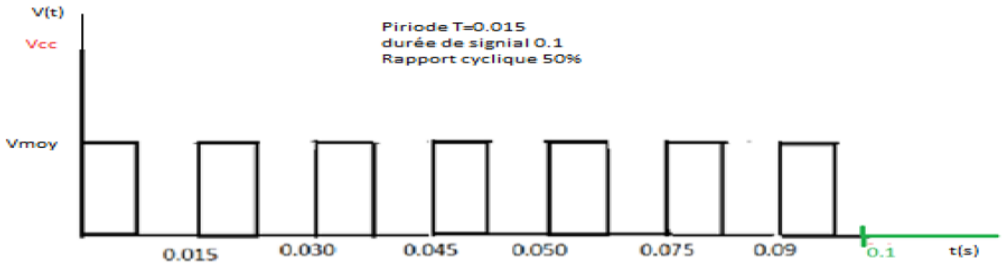


Figure 22 : Résultat Test2

Test 3 : le délai de signal s'est terminé pendent la phase on de la dernière impulsion donc un single continue est engendré, ou la LED reste allumé (la sortie égal V_{cc}).

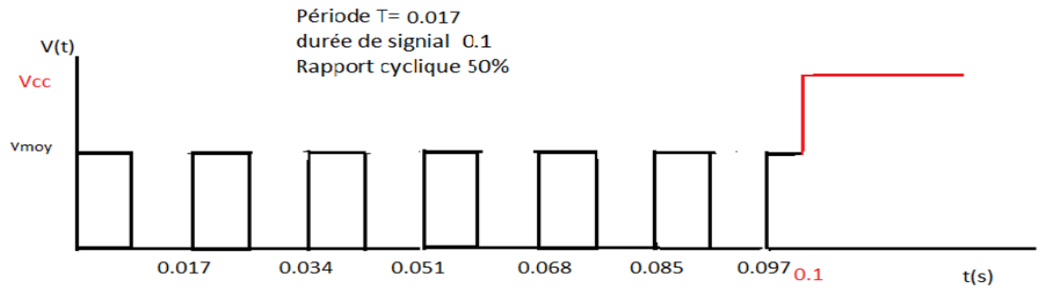


Figure 23 : Résultat Test3

D) UART (Universal Asynchronous Receiver/Transmitter)

D.1) Présentation de Protocole UART : [17]

L'UART (Universal Asynchronous Receiver/Transmitter) est un protocole de communication série intégré dans la pluparts des microcontrôleurs modernes, il permet la communication entre deux microprocesseurs. Chaque UART dispose d'une entrée RX et d'une sortie TX, le transmetteur envoie une trame de données bit par bit au récepteur, cette trame est composée :

- D'un bit de début toujours à 0 (Low level).
- D'un octet de données (08 bits).
- D'un bit de parité (paire ou impaire).
- D'un ou deux bits d'arrêt «stop» toujours à 1 (High level).

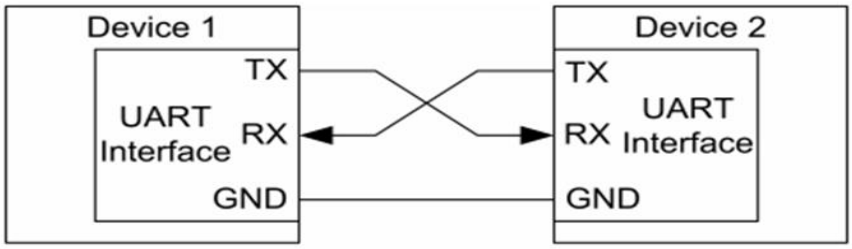


Figure 24 : Communication entre deux équipements

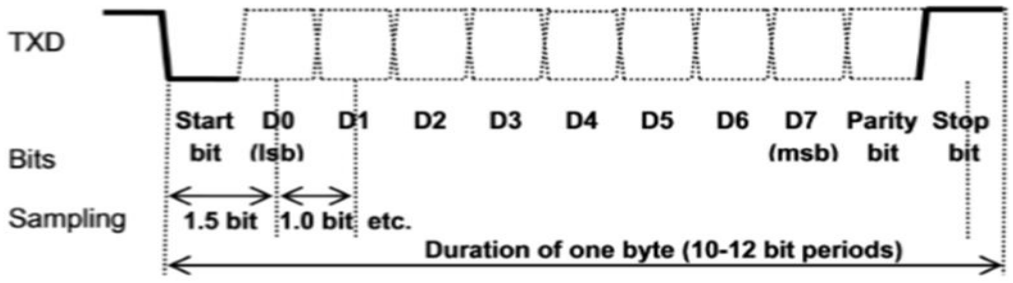


Figure 25 : Format de tram de données [17].

La transmission des données est faite d'une façon asynchrone, cela signifie qu'il n'y a pas de signal d'horloge (CLK) entre l'émetteur et le récepteur, d'où le récepteur se base sur la durée de ses bits qui doit être fixé et connue de l'émetteur et du récepteur pour identifier les différents bits d'un octet. La durée d'un bit est définie par le BAUDRATE, nombre de bits par secondes. Le

BAUDRATE peut avoir les valeurs suivantes : 300 bps, 1200 bps, 2400 bps, 4800 bps, 9600 bps, 19200 bps, 38400 bps, des vitesses rapides sont parfois aussi possibles.

L'interfaçage du microcontrôleur avec le PC nécessite l'utilisation d'un circuit intermédiaire dit « convertisseur USB to UART » pour connecter le microcontrôleur directement au PC via le port série.

D.2) L'UART de MSP430G2553

Dans MSP430, la communication en série est assurée par un périphérique sur puce appelé **USCI** (**U**niversal **S**erial **C**ommunications **I**nterface). Le périphérique est conçu de manière à pouvoir gérer plusieurs formats de communication série, synchrones et asynchrones comme SPI, I2C, IrDA, UART, etc. MSP430G2553 a deux noms de modules USCI comme **USCI_A0** et **USCI_B0** pour gérer plusieurs formats de communication. USCI_A0 peut être configuré pour gérer LIN, IrDA, SPI et la communication série asynchrone (UART) tandis que USCI_B0 peut gérer SPI et I2C.

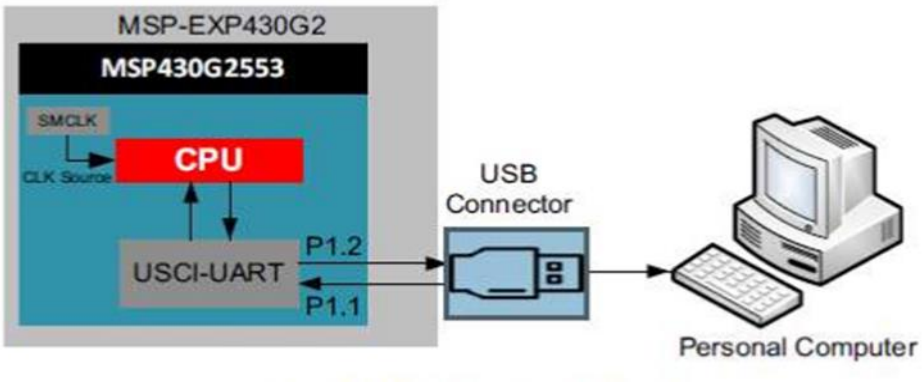


Figure 26 : Communication entre MSP430G2553 et PC.

D.2.1) Les Registres

Tableau 4 : Les registres de l'UART de MSP430G2553 [20]

Registre	Forme courte	Type de registre	Adresse	Etat initial
USCI_A0 Control registre 0	UCA0CTL0	Read/write	060h	Reset with PUC

USCI_A0 Control registre 1	UCA0CTL1	Read/write	061h	001h with PUC
USCI_A0 Baud rate Control register 0	UCA0BR0	Read/write	062h	Reset with PUC
USCI_A0 Baud rate Control register 1	UCA0BR1	Read/write	0.63h	Reset with PUC
USCI_A0 Modulisation contrôle register	UCA0MCTL	Read/write	064h	Reset with PUC
USCI_A0 status register	UCA0STAT	Read/write	065h	Reset with PUC
USCI_A0 receive buffer register	UCA0RXBUF	Read	066h	Reset with PUC
USCI_A0 Transmit buffer register	UCA0TXBUF	Read/write	067h	Reset with PUC
USCI_A0 Auto Baud control register	UCA0ABCTL	Read/write	05Dh	Reset with PUC
USCI_A0 IrDA transmit control register	UCA0IRTCTL	Read/write	05Eh	Reset with PUC
USCI_A0 IrDA receive control register	UCA0IRRCTL	Read/write	05Fh	Reset with PUC
SFR interrupt enable register 2	IE2	Read/write	001h	Reset with PUC
SFR interrupt flag register 2	IFG2	Read/write	003h	00Ah with PUC

D.2.2) Configuration de Baud rate [21]

La fiche technique du MSP430 contient une description détaillée de la configuration du débit en bauds pour différents modes. La manière la plus simple de configurer la vitesse de transmission du MSP430 consiste à utiliser le tableau de réglage de la vitesse de transmission répertorié par l'utilisateur. Découvrez simplement quelles valeurs UCBRx sont compatibles avec la fréquence BRCLK et la vitesse de transmission que vous souhaitez utiliser. Après avoir connu les valeurs, mettez-les simplement dans UCA0BR0 et UCA0BR1 pour configurer la vitesse de transmission.

Tableau 5 : Configuration de Baud rate

Fréquence BCLCK(HZ)	**Baud Rate [Baud]**	**UCBRx**	**UCBRSx**	**UCBRFx**	**Erreur TX maximale [%]**		**Erreur TX maximale [%]**	
12,000,000	115200	104	1	0	-0.5	0.6	-0.9	1.2
12,000,000	128000	93	6	0	-0.8	0	-1.5	0.4
12,000,000	256000	46	7	0	-1.9	0	-2.0	2.0
16,000,000	9600	1666	6	0	-0.05	0.05	-0.05	0.1
16,000,000	19200	833	2	0	-0.1	0.05	-0.2	0.1
16,000,000	38400	416	6	0	-0.2	0.2	-0.2	0.4
16,000,000	56000	285	6	0	-0.3	0.1	-0.5	0.2
16,000,000	115200	138	7	0	-0.7	0	-0.8	0.6
16,000,000	128000	125	0	0	0	0	-0.8	0
16,000,000	256000	62	4	0	-0.8	0	-1.2	1.2

Nous présentons en dessous notre exemple de programmation UART :

```
#include"msp430g2553.h"
void main (void){
 WDTCTL = WDTPW + WDTHOLD; // Stop the Watch dog
//------------------- Configure the Clocks ------------------//
DCOCTL  = 0;            // Select lowest DCOx and MODx settings
```

```
BCSCTL1=CALBC1_1MHZ   //Set range
COCTL  = CALDCO_1MHZ;      // Set DCO step + modulation
 //---------------- Configuring the LED's ----------------------//
  P1DIR  |=  BIT0 + BIT6;  // P1.0 and P1.6 output
  P1OUT  &= ~BIT0 + BIT6;             // P1.0 and P1.6 = 0
 //--------- Setting the UART function for P1.1 & P1.2 --------//
   P1SEL  |=  BIT1 + BIT2;  // P1.1 UCA0RXD input
   P1SEL2 |=  BIT1 + BIT2;  // P1.2 UCA0TXD output
  //------------ Configuring the UART(USCI_A0) ----------------//
  UCA0CTL1 |=  UCSSEL_2 + UCSWRST;  // USCI Clock = SMCLK,USCI_A0
disabled
  UCA0BR0   =  104;              // 104 From datasheet table-
  UCA0BR1   =  0;              // -selects baudrate =9600,clk = SMCLK
  UCA0MCTL  =  UCBRS_1;            // Modulation value = 1 from datasheet
  UCA0STAT |=  UCLISTEN;           // loop back mode enabled
  UCA0CTL1 &= ~UCSWRST;            // Clear UCSWRST to enable USCI_A0
   //---------------- Enabling the interrupts ------------------//
    IE2  |=  UCA0TXIE;                          // Enable the Transmit interrupt
     IE2  |=  UCA0RXIE;                          // Enable the Receive  interrupt
  _BIS_SR(GIE);                // Enable the global interrupt
UCA0TXBUF = 'G';                // Transmit a byte
  _BIS_SR(LPM0_bits + GIE);        // Going to LPM0
}
//   ===== =====   Transmit and Receive interrupts======      //
#pragma vector = USCIAB0TX_VECTOR
 __interrupt void TransmitInterrupt(void)
 {
   P1OUT  ^= BIT0;//light up P1.0 Led on Tx
  }

 #pragma vector= USCIAB0RX_VECTOR
 __interrupt voidReceiveInterrupt(void)
```

```
{
  P1OUT  ^= BIT6;     // light up P1.6 LED on RX
  IFG2 &= ~UCA0RXIFG;} // Clear RX flag
```

Cet exemple configure un UART et transmet la lettre ASCII «G». UART est configuré en mode de récupération pour que «G» soit reçu et stocké dans UCA0RXBUF. La LED connectée à P1.0 s'allume lorsque «G» est transmis et la LED est connectée à P1.6 lorsque «G» est à nouveau reçu.

2.3. Le circuit de puissance (driver) A4988 :

L'A4988 est un circuit intégré ou un contrôleur dédié à la commande des moteurs pas à pas bipolaires, Ce contrôleur, aussi communément appelé "StepStick" ou "Stepper Driver", permet de contrôler des moteurs pas-à-pas bipolaires en micro-stepping avec un maximum de 2 ampères par bobine. Il y'a 5 résolutions différentes de pas :

1- Full-step: Pas complet

- Half-step: 1/2 pas
- Quarter-step : 1/4 de pas
- eighth-step : 1/8 ième de pas
- sixteenth-step : 1/16 ième de pas

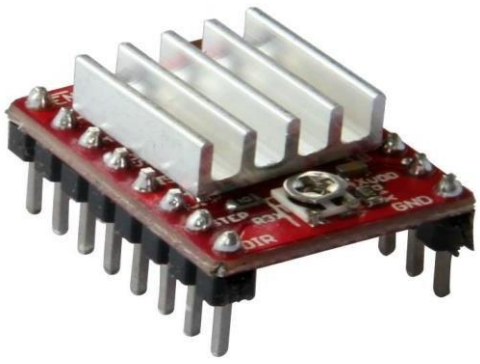


Figure 27 : Driver A4988 [22]

2.3.1. Caractéristiques et détails techniques de circuit A4988 :

Voici quelques caractéristiques essentielles de ce driver :

- Tension de fonctionnement minimum : 8 V.
- Tension maximale de fonctionnement : 35 V.

- Courant par phase (en continu) : 1,2 A.
- Courant maximum par phase : 2.3 A.
- Logique de commande :
- Tension minimale : 3 V
- Tension maximale : 5.5 V
- Une commande de hachage intelligente qui sélectionne automatiquement le bon mode.
- Protection (arrêt) en cas de surchauffe, sous tension, surcharge/sur-courant/court-circuit. Utile en cas de surcharge/blocage moteur (car le courant va monter en flèche, ce qui doit provoquer la mise en protection).
- Composé de 2 pont H chaque pont H constitué de 4 transistors MOSFET.

2.3.2. Brochage et description des pins de driver A4988

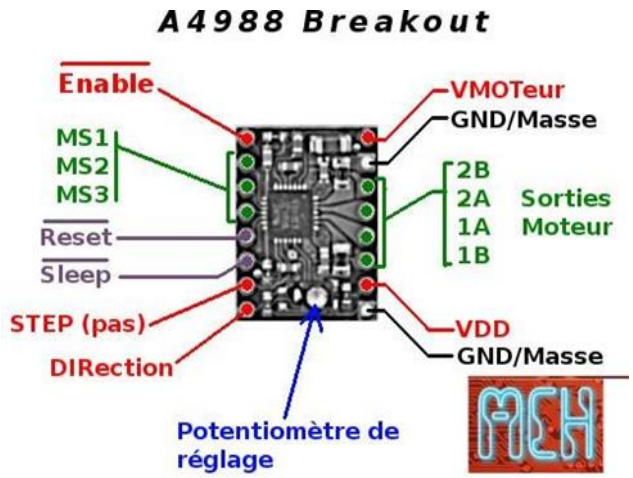


Figure 28 : Branchement A4988 [23]

- **Enable :** Logique Inversée, ce pin permet d'activer ou désactivé le moteur, quand il est mis dans un état haut (en logique 1) le moteur est désactivé et quand il est mis en état bas(en logique 0) le moteur est activé.
- **MS1, MS2, MS3 :** Ces pins permettent de sélectionner la configuration de mode de résolution de moteur pas à pas.
- **RESET :** Logique inversée. Permet de faire une réinitialisation du module. Généralement connecté sur la broche "SLEEP".
- **SLEEP :** Logique inversée. Généralement connecté sur la broche "Reset" du module
- **STEP :** En l'envoyant un signal d'horloge (haut puis bas) le moteur avancera d'un pas
- **DIR** : Permet d'indiquer la direction de rotation du moteur. Etat Haut=High pour tourner dans un sens, Etat bas=Low pour tourner dans l'autre sens

- **VMOT :** Tension d'alimentation du moteur. Habituellement 12V pour les moteurs pas à pas. Tension entre 8 et 12v.
- **GND** : Sous "VMOT", masse pour l'alimentation du moteur.
- **1B 1A** : Première bobine du moteur pas à pas bipolaire.
- **2A 2B** : Deuxième bobine du moteur pas à pas bipolaire.
- **VDD** : Alimentation de la logique de commande entre 3 et 5.5v.
- **GND** : Sous "VDD", masse de la logique de commande.

2.3.3. Ajustement de Driver A4988 [24]

2.3.3.1. Ajustement des entrées du contrôle :

Chaque impulsion sur l'entrée STEP correspond à un pas ou micro pas du moteur pas à pas dans la direction sélectionnée sur la broche DIR. Les broches STEP et DIR ne disposent pas de résistances pull-down ou pull-up interne. Donc le contrôle de deux états Haut et Bas de ces entrées est fait directement par le microcontrôleur. La tension sur ces broches ne doit jamais être flottante.

2.3.3.2. Sélection de mode de pas

Les moteurs pas-à-pas disposent de leur propre spécification physique de "pas" connu comme un "pas complet". Un pilot Micro-pas tel que l'A4988 permet d'obtenir une plus grande résolution en autorisant des positions intermédiaires dans un pas. Cela est rendu possible en modulant intelligemment la quantité de courant dans les bobines du moteur pas-à pas. Par exemple, piloter un moteur en mode "1/4 de pas" permet d'obtenir 800 micro-pas sur un moteur prévu pour 200 pas par révolution (tour) et cela en utilisant 4 niveau de courants différents pour chacun des Microsteps. La configuration de mode de résolution se fait à l'aide des broches MS1, MS2 et MS3.

- MS1 et MS3 disposent d'une résistance pull-down interne de 100kΩ.Une résistance pull-down signifie que si on ne place pas ces broches au niveau logique haut : VDD (la tension choisie pour la logique de commande) alors elles seront automatiquement ramenées au niveau logique bas.
- MS2 dispose d'une résistance pull-down de 50kΩ :

Si les broches MS1, MS2 et MS3 sont laissées flottantes, l'A4988 fonctionnera en mode "pas complet". Pour que le Microstepping fonctionne correctement, il faut que le courant soit limité d'une façon à ne pas activer la protection en sur-courant. Sinon, les niveaux des courants intermédiaires ne seront pas correctement maintenus et le moteur pourrait sauter des microsteps.

2.4.Commande moteur pas à pas :

Dans projet nous avons utilisé des moteurs pas à pas NEMA 17 42BYGH811. Nous exposons dans cette partie comment commander ces moteurs avec MSP430G2553 et A4988.

2.4.1. Spécifications de moteur NEMA17 42BYGHW811

- **Spécifications générales :**
  - Précision de l'angle de marche : + - 5% (pas complet, pas de charge).
  - Précision de résistance : + - 10%.
  - Précision d'inductance : + - 20%.
- **Spécifications techniques :**
  - Angle de marche : 1.8 degrés
  - 200 pas par tour (1.8 deg/pas).
  - Tension : 3.1 V.
  - 2 phases (bipolaires) et 4 fils.
  - Courant 2.5 A/phase.
  - Résistance: 1.25 Ohm/phas. Inductance: 1.8 mH/phase .
  - Diamètre de l'arbre : 5.0mm. Longueur : 48mm. Poids : 0.37 kg.

Pour quoi utilisons le moteur NEMA17 42BYGHW811 ?

Moteur pas à pas parfait pour la construction des machines CNC ou nombreux autres applications. Le modèle 42BYGHW811 est recommandé par beaucoup de professionnels. Ainsi le moteur du fabricant Wantai avec 4800g*cm est presque 20% plus puissant que les autres modèles de la classe Nema 17, la figure suivant représenter la photo de ce moteur

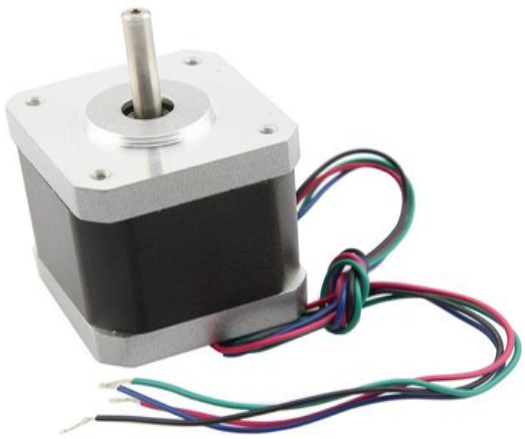


Figure 29 : Moteur NEMA17 42BYGHW811

2.3.2 Circuit de commande moteur pas à pas

La figure suivant représente le circuit électrique de la commande de deux moteurs pas à pas de type NEMA17 42BYGHW811 à l'aide de microcontrôleur msp430G2553 et les drivers A4988.

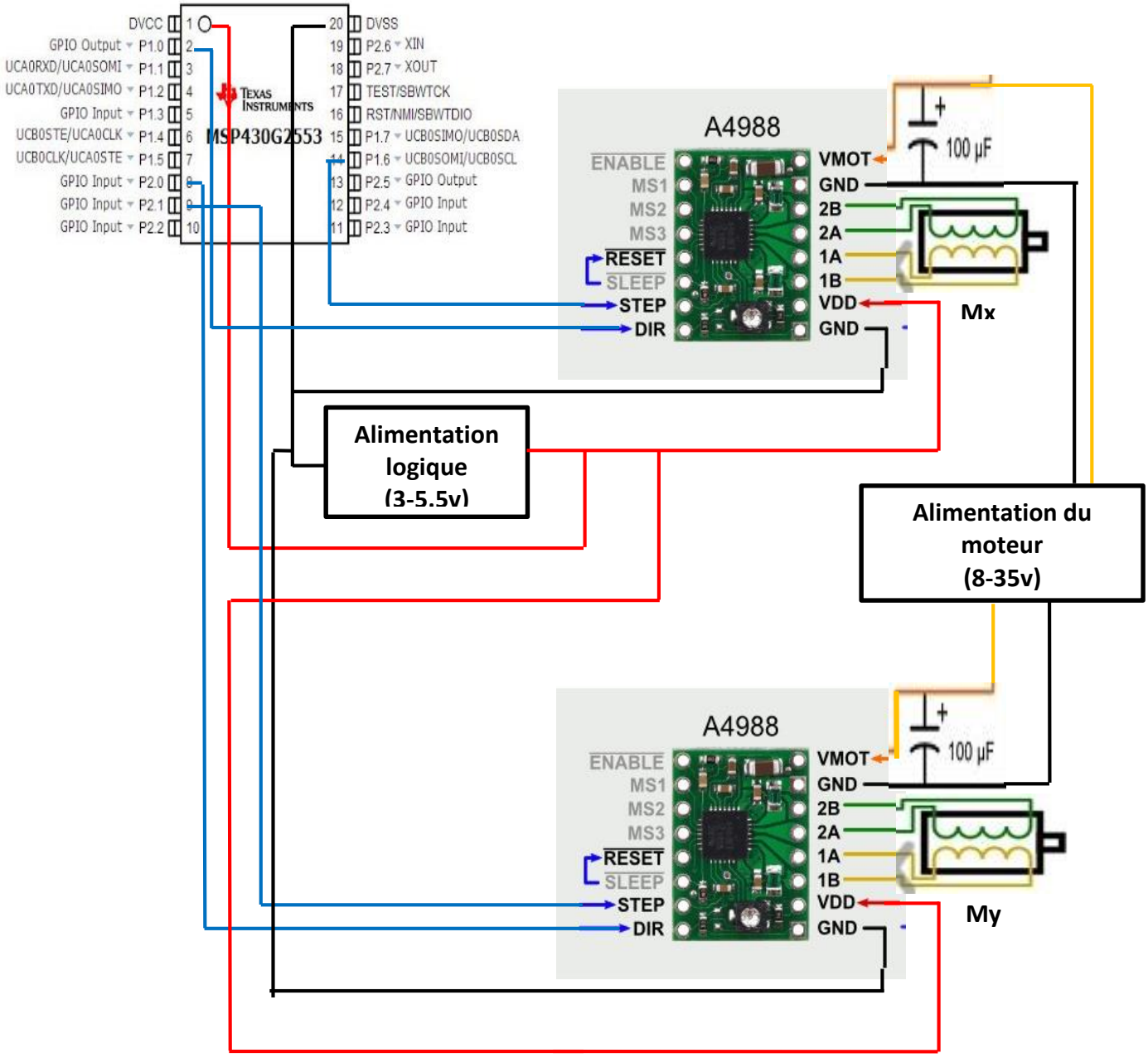


Figure 30 : Circuit de commande de deux moteurs pas à pas.

2.4.2. Résultats et discussion

Nous avons élaboré deux programmes pour commander deux moteurs pas à pas avec le circuit de la figure 30. Le premier programme pour contrôler les deux moteurs par PWM et le deuxième programme pour les contrôler par GPIO.

Premier programme : commande de deux moteurs par PWM

```
/*==========This program1   Control Stepper Motor with PWM ====*/
void main(void) {
int m = 0;
WDTCTL = WDTPW + WDTHOLD;    // Stop WDT
DCOCTL = 0;                              // Select lowest DCOx and MODx settings
BCSCTL1 = CALBC1_1MHZ;             // Set DCO
DCOCTL = CALDCO_1MHZ;
// initialiser the registers of all pins
P1DIR = 0x00;
P2DIR = 0x00;
P2OUT = 0x00;
P1OUT = 0x00;
// Set PWM  Mx
TA0CCTL1 = OUTMOD_7;          // set OUTMOD _ 7
TA0CTL = TASSEL_2 + MC_1;     // TASSEL_2 = SMCLK, MC_1= up mode
// Set PWM  My
TA1CCTL1 = OUTMOD_7;          // set OUTMOD _ 7
TA1CTL = TASSEL_2 + MC_1;     // TASSEL_2 = SMCLK, MC_1= up mode
tournMx(400, 1);                        //    Motor  Mx rotates 400 steps in cw direction
tournMy ( 400,0) ;}                  //    Motor  My rotates 400 steps in ccw direction


void tournMx (int n,int m){     // n = the number of steps that the motor must turn;
                                        // m = sens of rotation (1 cw, 0 ccw)
```

```
  P1DIR |= 0x01;                    // P1.0 as OUTPUT to control the driver direction of Mx
   if (m == 1) {
    P1OUT |= 0x01;                  // direction cw of rotation of Mx
}
  else {
    P1OUT &= ~ 0x01;          // direction ccw of rotation of Mx
  }
   P1DIR |= 0x40;               // P1.6 as output to control the motor Mx steps
   P1SEL |= 0x40;               // P1.6 selects the PWMx function of pin P1.6
  TA0CCR0 = 10000;              // set of PWM Period
  TA0CCR1 = TA0CCR0/2;          // set of Duty cycle 50%
for(int i=0;i<n;i++) __delay_cycles(10000);}
  TA0CCR1 = 0;}     //turne of PWMx
     void tournMy (int n,int m){     // n = the number of steps that the motor must turn;
     P2DIR |= 0x01;           // P2.0 as OUTPUT to control the driver direction of My
   if (m == 1) {
    P2OUT |= 0x01;              // direction cw of rotation of My
}
  else {
    P2OUT &= ~ 0x01;          // direction ccw of rotation of My
  }
   P2DIR |= 0x02;               // P2.1 as output to control the motor steps
   P2SEL |= 0x02;               // P2.1 selects the PWMx function of pin P2.1
  TA1CCR0 = 10000;              // set of PWMy Period
  TA1CCR1 =  TA1CCR0/2;          // set of Duty cycle 50%
for(int i=0;i<n;i++) __delay_cycles(10000);
  TA1CCR1 = 0;}      //turne of PWMy
```

Deuxième programme : commande de deux moteurs pas à pas, par GPIO

```
/*==========This program2   Control Stepper Motor with GPIO ====*/
void main(void) {
int m = 0;
WDTCTL = WDTPW + WDTHOLD;    // Stop WDT
DCOCTL = 0;                                  // Select lowest DCOx and MODx settings
BCSCTL1 = CALBC1_1MHZ;              // Set DCO
DCOCTL = CALDCO_1MHZ;
// initialiser the registers of all pins
P1DIR = 0x00;
P2DIR = 0x00;
P2OUT = 0x00;
P1OUT = 0x00;
tournMx(400, 1);                          //    Motor  Mx rotates 400 steps in cw direction
tournMy ( 400,0) ;}                    //    Motor  Mx rotates 400 steps in ccw direction
void tournMx (int n,int m){     // n = the number of steps that the motor must turn;
                                          // m = sens of rotation (1 cw, 0 ccw)
  P1DIR |= 0x01;           // P1.0 as OUTPUT to control the driver direction of Mx
 if (m == 1) {
   P1OUT |= 0x01;              // direction cw of rotation of Mx
}
  else {
    P1OUT &= ~ 0x01;          // direction ccw of rotation of Mx
  }
P1DIR |=0x40 ;    // P1.6 as output to control the motor Mx steps
For (i=0,i<n,i++) {            // loop to create pulse signals the Mx
```

```
 P1OUT |=0x40;                //    P2.1 set to low (0)
 __delay_cycles(10000);  // wait 0.5 s
P1OUT &= ~ 0x40;      // P1.6 set to low(0)
  __delay_cycles(10000); // wait 0.5 s
}}
void tournMy (int n,int m){     // n = the number of steps that the motor must turn;
                                // m = sens of rotation (1 cw, 0 ccw)
  P2DIR |= 0x01;           // P2.0 as OUTPUT to control the driver direction of My
   if (m == 1) {
    P2OUT |= 0x01;              // direction cw of rotation of My
}
  else {
    P2OUT &= ~ 0x01;          // direction ccw of rotation of My
  }
P2DIR |=0x02 ;    // P2.1 as output to control the motor My steps
For (i=0,i<n,i++) {       // loop to create pulse signals the My
 P2OUT |=0x02;             // P2.1 set to high (1)
 __delay_cycles(10000);   // wait 0.5 s
P2OUT &= ~ 0x02;            // P2.1 set to low(0)
  __delay_cycles(10000);  // wait 0.5 s
}}
```

- Premièrement, nous arrêtons le timer du chien de garde et nous initialisons l'horloge de tous les registres de tous les pins.
- Ensuite nous configurons PWM pour les deux moteur dans le premier programme, où nous sélectionnons le mode de sortie (Reset/set), et nous sélectionnons le sous-main

clock « SMCLK » : a une fréquence par défaut de 1 Mhz, et nous sélectionnons le mode de comptage (Up mode).

- Nous appellons la fonction (tournMx) pour faire tourner le moteur Mx 400 pas, dans le sens de rotation d'une montre (cw) et après nous appellons la fonction (tournMy) pour faire tourner le moteur My 400 pas, dans le sens contraire de rotation d'une montre ccw. Pour contrôler le sens de rotation du moteur Mx et le nombre de pas nous utilisons les broches P1.0 et P1.6 respectivement et les broche P2.0 et P2.1 pour le moteur My.
- Dans le premier programme nous a créons un signal de commande PWM pour contrôler les deux moteurs sur les sorties des broches P1.6 et P2.1 à l'aide des registres TIMER.
- Dans le deuxième programme nous créons un signale carrée pour contrôlé les deux moteurs sur les sortie des broches P1.6 et P2.1 à l'aide des registres GPIO.

2.4.2.1. La déférence entre command par PWM et par GPIO :

* La consommation de la puissance est moins dans le cas de commande par PWM que dans la commande par GPIO. Cela est dû au fait que la commande PWM utilise en sortie de la broche la moitié de la valeur V_{cc} donnée au MCC tandis que la commande par GPIO utilise toute la valeur V_{cc} en sortie de la broche

* Un gain important du temps alloué par le microcontrôleur est observé dans le cas de la commande par PWM

* Le MSP430 contient deux TIMER (TIMER A0, TIMER A1) cela impose que par la commande par PWM on ne peut contrôler que deux moteurs. Tandis que la commande par GPIO permet de contrôler plus que de deux moteurs

2.5.Conclusion

Dans ce chapitre nous avons présentés les composant électronique utilisés dans notre projet à savoir : le microcontrôleur MSP430, LaunchPad MSP430, le driver de commande de moteurs le A4988 et le moteur pas à pas type Nema17.

Nous avons réussi à contrôler les moteurs pas à pas par la commande PWM et GPIO. C'est notre premier pas vers la commande de notre machine CNC.

Chapitre 3

Programmation

3.1.Introduction :

Notre modèle CNC se déplace selon le G-Code qui provient de l'ordinateur via le port série. G- Code est le langage de programmation de machine à commande numérique qui décrit la direction et la quantité de mouvement de l'axe à déplacer. Dans ce chapitre, Nous discuterons certaines commandes G-Code et la signification de chaque commande. Nous exposons comment écrire un programme en utilisant le langage G-CODE, comment interprété G-code avec MSP430G2553

3.1.1. Historique du langage G-code :

Le G-code est un langage de programmation, était développé par l'EIA au début des années 1960, et finalement normalisé par l'ISO en février 1980 sous la référence RS274D/ (ISO 6983).

Compte tenu de l'absence de développement ultérieur, une grande variété des configurations de machines-outils, et du peu de demande pour une réelle interopérabilité, peu de contrôleurs à commande numérique respectent ce standard. Des extensions et variantes ont été ajoutées indépendamment par divers fabricants, ce qui fait que les opérateurs doivent connaitre les différents dialectes et particularité des machines qu'ils utilisent, et les systèmes de CFAO (Conception et Fabrication Assistées par Ordinateur) doivent se limiter au plus petit dénominateur commun des machines qu'ils commandent.

Beaucoup de fabricants ont essayé de contourner cette difficulté et rester compatible en suivant la route tracée par la firme Fanuc. Malheureusement, Fanuc n'est pas conforme à la norme RS- 274 ou à ses précédents standards, et a été lent à ajouter de nouvelles fonctionnalités et à utiliser la puissance croissante des ordinateurs. Par exemple, ils ont transformé la commande g70/g71 en g20/g21 ; ils ont utilisé des parenthèses pour les commentaires, ce qui a causé des problèmes lors de l'introduction des calculs mathématiques ; ils n'ont commencé à utiliser les nanomètres que récemment (ce qui requiert 64 bits) ; ils ont introduit les nurbs pour compenser le faible débit des blocs depuis la mémoire (au lieu de mettre en place un cache).

Depuis l'établissement de la norme ISO 6983, et les technologies évoluant rapidement, de nombreuses extensions ont été ajoutées pour tenir compte des nouveautés et des nouvelles capacités des machines-outils. Ces extensions, bien que souvent utiles chez des constructeurs différents de directeur de commande numérique, n'entrent pas dans la norme et compliquent la tâche des logiciels de fabrication assistée par ordinateur, qui doivent créer les lignes de ce langage pour un DCN particulier. Pour analyser la syntaxe des programmes ISO, des éditeurs de logiciels ont mis au point des solutions de simulation d'usinage qui ont permis d'émuler le comportement

d'un DCN sur un PC. Cela permit de mettre au point les programmes sans mobiliser la machine-outil à commande numérique. On peut citer des solutions comme Ncsimul, Roboris, Ncview, etc… A côté de l'ISO, des langages propres aux constructeurs de DNC ainsi que les interfaces de programmation conversationnelle sont destinées à simplifier la programmation.

Le principe est simple : on va écrire une suite de lignes comportant quelques instructions et commandes qui vont demander à la machine de réaliser des mouvements dans l'espace en 3 dimensions. On va pour cela utiliser un repère cartésien dont les 3 axes de références sont les axes X-Y-Z [25].

3.1.2. Définition de G-code

Le G-code est le langage plus utilisé pour contrôler une machine à commande numérique. Il s'agit bien d'un langage de programmation, qui nous sert donc à programmer les mouvements que la machine va effectuer, et le fichier contenant la suite d'instruction s'appelle, en toute logique, un programme. Il s'agit de simple fichier texte, humainement lisible, au même titre que du code en C, Pascal ou Basic. Il se compose d'un certain nombre de ''commandes'' spécifiques indiquant à la machine quel type de mouvement elle doit exécuter (droite, arc de cercle, etc.…), et l'indication de coordonnées sur les axes X, Y et Z. [26]

3.1.3. Les usages de G-code

Le G-code Développé à l'origine pour des machines-outils par enlèvement de matière**,** il est désormais utilisé dans un domaine très vaste de la fabrication, avec des adaptations :

- Usinage par enlèvement de matière : tournage, fraisage, perçage, gravure, défonçage ;
- Découpe avec : couteau, laser, jet d'eau, plasma, flamme ou oxydation.
- Poinçonnage
- Impression 3D : par dépôt de matière, durcissement d'une résine, solidification de poudre.

3.2.Programmation de la machine à commande numérique

Un programme CN comporte des caractères obligatoires de début et fin. L'affectation des blocs situés entre ses caractères est dans l'ordre de les écrire et leurs numérotation n'intervient pas dans l'ordre de déroulement du programme donc il est conseillé de les numéroter dans l'ordre d'écriture [25]. On peut programmer manuellement ou assistée par un logiciel de FAO.

3.2.1. Structure d'un programme

Un programme CN se compose d'une suite de blocs. Chaque bloc se compose des mots et chaque mot contient les données et les ordres l'exécution d'une opération requis. Il est divisé en 3 parties : tête de programme, corps de programme et fin de programme.

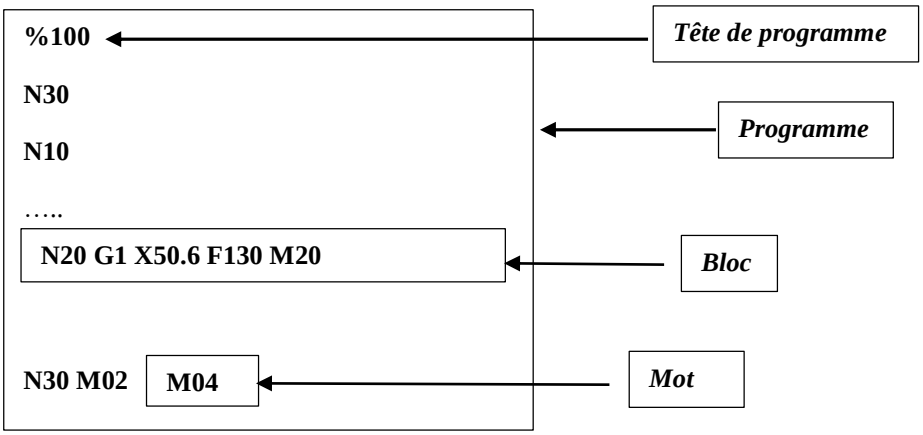


Figure 31 : Structure d'un programme. [27]

3.2.2. Format d'un bloc

On définit une ligne d'instruction composée de mots codés par un bloc à transmettre au système de commande. Le format de bloc définit la syntaxe des mots de fonction et de dimension composant chaque bloc de programme [25].

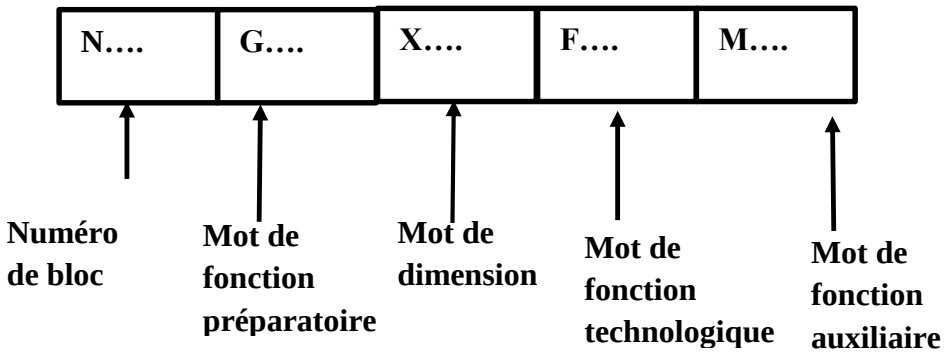


Figure 32 : Format d'un bloc [27]

3.2.3. Format d'un mot

Une instruction ou un ordre à transmettre au système de commande est définit par un mot.

Un mot est composé d'une lettre et d'une valeur numérique, entière ou décimale.

Adresse (*1 ou 2 lettre / 1 caractère*)	**Signe algébrique** (+ / -)	**Données numériques** (*Chiffres lié à l'adresse*)

Figure 33 : Format d'un mot

On distingue deux types de mot :

- Mots définissant des dimensions :

X	+	5.3

- Mots définissant des fonctions :

G	**0**	**2**

3.2.3.1. Les différents formats des mots :

Dans un bloc, les mots peuvent avoir plusieurs fonctions, chaque fonction de mot est affectée en tant qu'adresse identifiée par un code de lettre dans le programme. Il existe différents fonction des mots, dans ce qui suit nous décrivons les fonctions de base les plus utilisés dans les programmes :

- **Fonctions préparatoires (G) :** Commande G-Code Standard, Déplacement d'un ou plusieurs axes, cycles machine.
  - **La fonction G00 (déplacement rapide) :**

Cette commande G00 s'utilise pour le déplacement des axes de la machine à vitesse rapide. Elle est utilisée principalement pour positionner rapidement la machine sur un point donné avant chaque commande d'avance (coupe ou dessiner).

En général. Chaque axe spécifié est déplacé à la même vitesse mais les axes ne finiront pas nécessairement leurs mouvements tous en même temps. La machine attendra que tous les mouvements soient terminés avant de lancer la commande suivante.

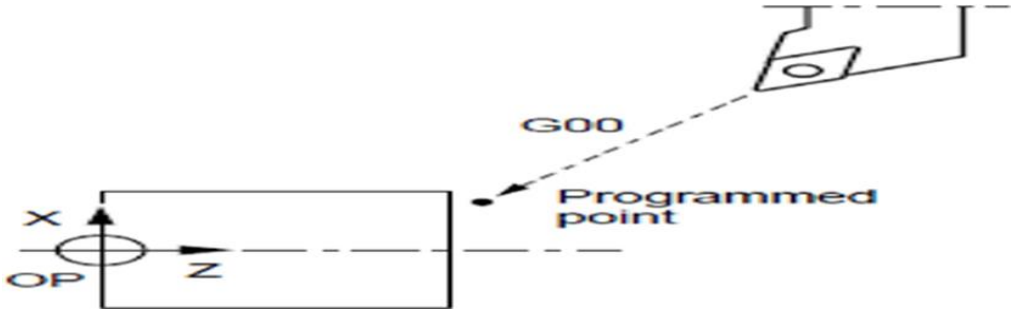


Figure 34 : Fonction G00 [28]

- **La fonction G01 (interpolation linéaire)**

Cette commande G01 assure un mouvement en ligne droite (linéaire) de point à point. Le mouvement peut avoir lieu sur 1 ou plusieurs axes. Vous pouvez commander un G01 avec 3 axes ou plus ; tous les axes vont se déplacer, du départ à l'arrivée, en même temps. La trajectoire est la résultante de tous les déplacements d'axes programmés dans le bloc.

Figure 35 : Fonction G01 [28]

- **Les fonctions G02 et G03 (interpolation circulaire)**

G02 : interpolation circulaire sens horaire à vitesse d'avance programmée.

G03 : interpolation circulaire sens antihoraire à vitesse d'avance programmée.

- Cette commande (G02 ou G03) assure une un mouvement circulaire des axes linéaires.
- Les valeurs X et Y et Z sont utilisées pour spécifier le point final du mouvement.

Il y a deux façons de spécifier le centre du mouvement circulaire :

- la première utilise I ou K ou J pour spécifier la distance depuis le point de démarrage jusqu'au centre de l'arc.
- la deuxième emploie R pour spécifier le rayon de l'arc.

I, J, K : Si on emploie I, J ou K pour spécifier le centre de l'arc, R ne peut pas être utilisé. Seulement I, J ou K spécifique pour le plan sélectionné (I, J pour G17, I, K pour G18, J, K pour G19) sera permis. Si seulement l'une des valeurs I, J, K est spécifiée, les autres sont considérées zéro. I, J ou K c'est la distance à signe entre le point de départ et le centre du cercle.

Figure 36 : Fonction G02 et G03 [28]

- **Les fonctions G90 et G91 (mode de déplacement de l'outil) :**

On distingue deux modes de commandes de déplacement de l'outil : en absolues et en relatives.

- **G90 :** programmation en absolue par rapport à l'origine programme.

Les valeurs des coordonnées de la position d'arrivée sont programmées.

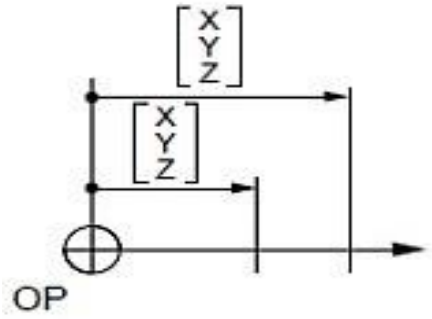


Figure 37 : Fonction G90 [28]

- **G91 :** programmation en relative par rapport au point de départ du bloc.

La valeur programmée sur un axe est repérée par rapport à la dernière position programmée.

La valeur est égale au déplacement à réaliser.

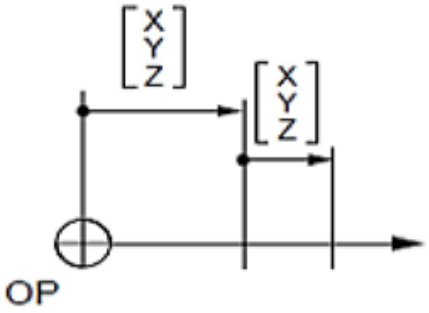


Figure 38 : Fonction G91 [28]

D'autres fonctions préparatoires fréquemment utilisées sont regroupées dans le tableau ci-dessous

Tableau 5 : Fonctions préparatoires fréquemment utilisées.

Code	Fonction
G17	Sélection du plan x-y
G18	Sélection du plan x-z
G19	Sélection du plan y-z
G20	Programmation en pouce
G21	Programmation en mm
G40	Désactivation de la compensation de rayon d'outil
G41	Compensation de rayon d'outil à gauche
G42	Compensation de rayon d'outil à droite
G28	Retour au point de référence
G94/G95	Déplacement en pouces par minute/pouce par tour
G96/G97	Vitesse de coupe constant /vitesse de rotation constant Ou annulation de G96
G80	Annulation du cycle fixe
G4	Pause (1s)

- **Coordonnées de points (X, Y, Z, I, J, K,)**

Tableau 6 : Signification des fonctions des coordonnées

Code	Fonction
X	Coordonnée X : peut être un nombre entier ou décimal
Y	Coordonnée Y : peut être un nombre entier ou décimal
Z	Coordonnée Z : peut être un nombre entier ou décimal
I	Paramètre - Décalage X dans le déplacement de l'arc
J	Paramètre - Décalage Y dans le déplacement de l'arc
K	Paramètre - Décalage Z dans le déplacement de l'arc

- **Fonctions auxiliaires (M) :**

Tableau 7 : Signification des fonctions auxiliaires.

code	Fonction
M00	Programme optionnel arrêt automatique.
M01	Demande d'arrêt de programme en option.
M02/M30	Fin du programme.
M03	Rotation de la broche sens horaire.
M04	Rotation de la broche sens anti-horaire.
M05	Arrêtez la broche
M06	Changement d'outil

- **Vitesses, avances, outils, BLOC... (S, F, T, N...) :**

Tableau 8 : Signification des fonctions

code	Fonction
T	L'appel de l'outil désigné par le nombre qui suit.
F	Avance de travail en millimètres/minute ou en millimètres/tour
S	Vitesse de rotation de la broche en tr/min
N	Numéro de bloc

3.2.4. Format d'un programme

Afin d'écrire un programme (CN) nous suivrons les étapes suivant :

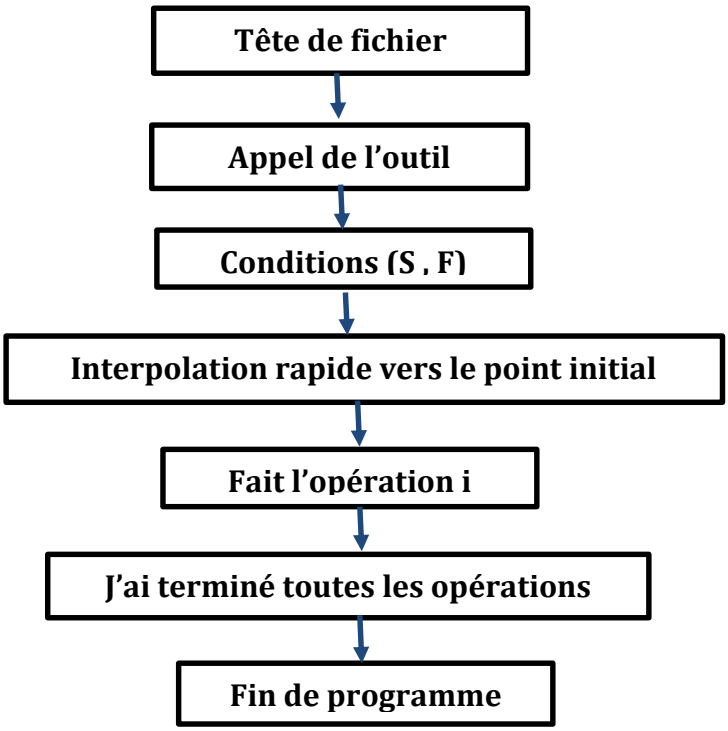


Figure 39 : Format d'un programme.

Dans l'exemple suivant nous avons élaboré un programme G-CODE en utilisant l'émulateur **Ncviewer** pour dessiner l'image suivante.

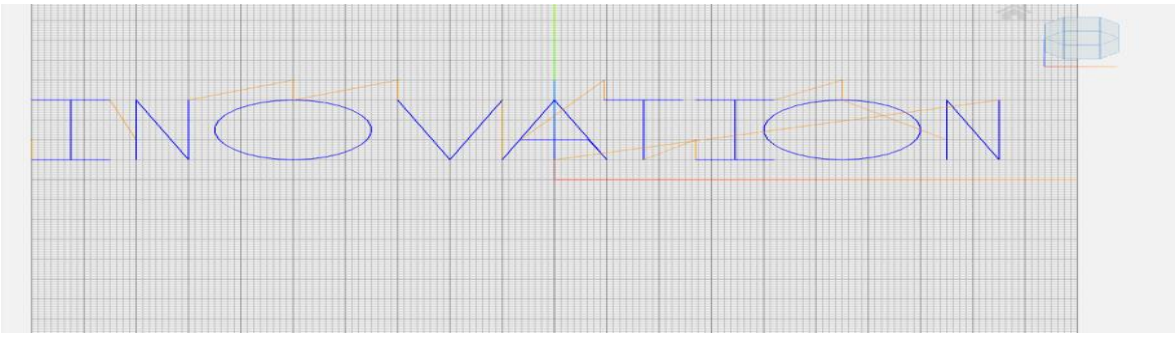


Figure 40 : Dessin testé par Ncviewer.

PROGRAMME G-CODE

```
%2020
G91 G17 G21
G0  X-100 Y10 Z10
G0 X0  Y0  Z-10
G1 X15  Y0
G1 X-7.5  Y0
G1 X0   Y30
G1 X-7.5 Y0
G1 X15 Y0
G0 X5 Y-30 Z10
```

```
G0 X0  Y0   Z-10
G1 X0  Y30
G1 X10 Y-30
G1 X0 Y30
G0 X20 Y0 Z10
G0 X0  Y0 Z-10
G2 X0 Y-30 J-15
G2 X0  Y30 J15
G0  X20  Y0 Z10
G0  X0   Y0  Z-10
G1  X10 Y-30
G1  X10 Y30
G0  X0 Y-30 Z10
G0 X0 Y0  Z-10
G1 X10 Y30
G1 X10 Y-30
G1 X-3.5 Y10
G1  X-13 Y0
G0 X16 Y20 Z10
G0 X0 Y0 Z-10
G1 X15 Y0
G1 X-7.5 Y0
G1 X0  Y-30
G0 X10 Y0 Z10
G0 X0 Y0 Z-10
G1 X15 Y0
G1 X-7.5 Y0
G1  X0 Y30
G1 X-7.5 Y0
G1 X15 Y0
G0 X13 Y0 Z10
G0 X0 Y0 Z-10
G03 X0 Y-30 J-15
G03 X0 Y30 J15
G0 X20 Y-30 Z10
G0 X0 Y0 Z-10
G1 X0 Y30
G1 X10 Y-30
G1 X0 Y30
G0 X-85 Y-40 Z10
G0 X0 Y0 Z-10
```

3.2.5. Interprétation de G-CODE avec MSP430G2553

Pour la programmation de notre machine on utilise les commandes g-code avec le microcontrôleur MSP430G2553. Pour utiliser le g-code avec MSP430G2553, on suit les étapes indiquées dans l'organigramme suivant :

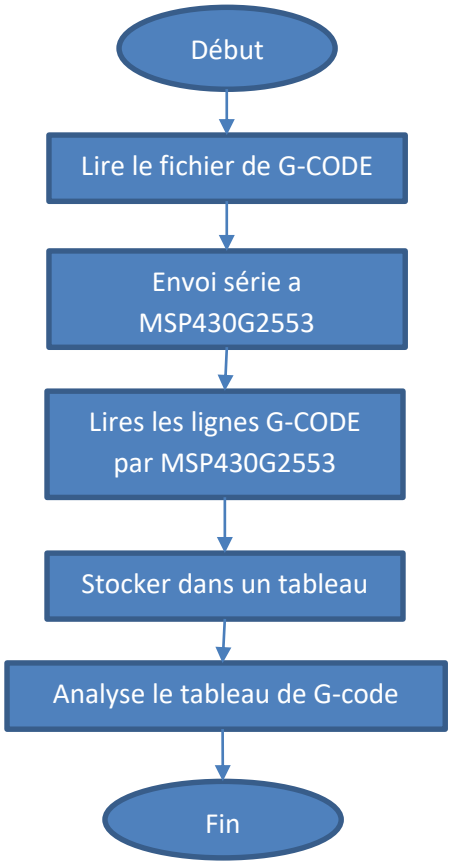


Figure 41 : Les étapes d'utilisation du G-code avec MSP430G2553.

3.2.5.1. Lire le fichier G-code

Nous allons créer un fichier texte (.txt) qui contient des lignes de G-code et lire ce fichier ligne par ligne puis le stoker dans un tableau. L'organigramme ci-dessous nous montre comment lire ce fichier texte.

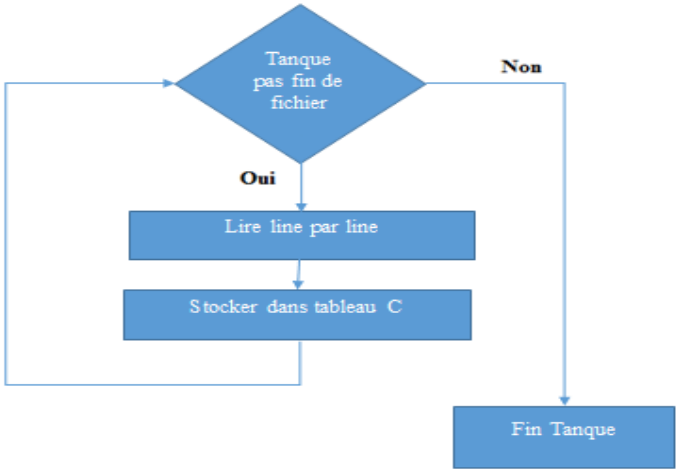


Figure 42 : Organigramme : lire un fichier de G-code

Le code ci-dessous nous montre comment lire ce fichier texte à l'aide du langage C:

```
#include <stdio.h>
int main(){
FILE *fp;
  fp = fopen("G-CODE.txt", "r"); // read mode
 if (fp == NULL)
   {
      printf("Error while opening the file.\n") ;
 }
 else {
      char c[100] ;
      while ((fgets(c,100,fp))!=NULL){
  printf("%s",c);
      }
 }
   fclose(fp);
 return 0;}
```

Les étapes de programme peuvent être résumées comme suit :

- Inclure la bibliothèque ***stdio.h***
- Déclarer un pointer **fp** de type fille et initialiser le pointer à null.
- Appeler la fonction d'ouverture de fichier **fopen** et récupérer la valeur qu'elle renvoie dans le pointeur **fp** et écrire dans la fonction **fopen** le nom de fichier **G-code.txt** et le mode d'ouverture lecture (r). Donc le pointer **fp** devient pointer sur **G-code.txt**.
- Vérifier si l'ouverture a réussi : on peut lire le fichier (Figure 3.11), sinon : erreur d'ouverture de fichier on affiche ce message **("Error while opening the file").**
- Déclarer un tableau de chaine de caractère **c** de taille **100**.
- Appel de la fonction **fgets** pour lire une line de caractère et stoker le contenu de la ligne lu **(**lire tableau **c[100]**), pour lire tous les ligne du fichier en fait une boucle **while,** cette dernière s'arrêter si **fgets** récupéré **null.**
- Afficher la ligne qu'on vient de lire
- Fermer le fichier **fp** avec la fonction **fclose**.

Résultat sont affiché sur Terminale I/O de IAR (Figure 44) :

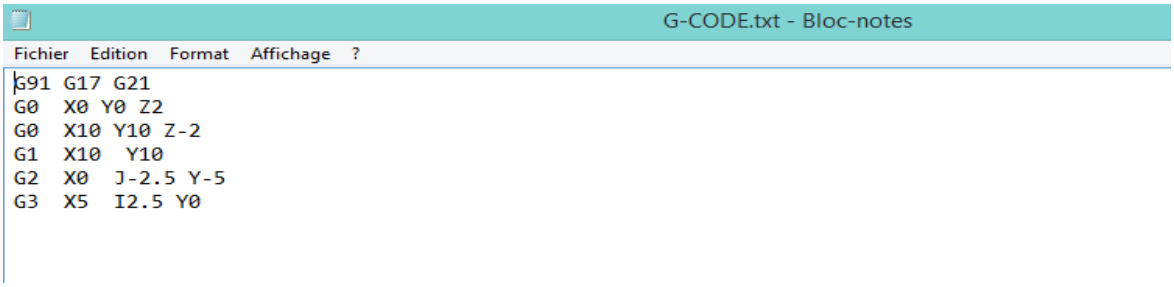


Figure 43 : Fichier txt de G-code crée.

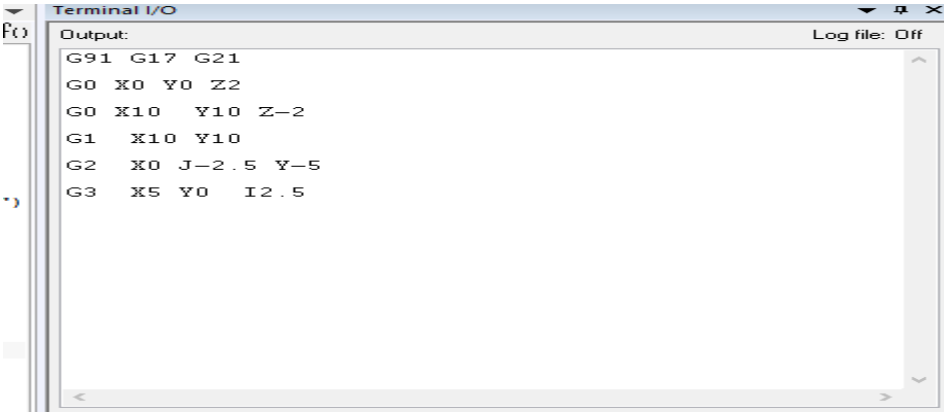


Figure 44 : Résultat de lecture du fichier G-Code.

3.2.5.2. Envoi série (SERIAL SEND) :

Notre projet dépend des commandes G-Code qui proviennent de l'ordinateur en série, donc nous allons écrire un programme en langage ***C*** qui ouvre le port série, et renvoie les lignes de G-code en série au microcontrôleur MSP430G2253.

L'organigramme ci-dessous décrit notre algorithme de programme :

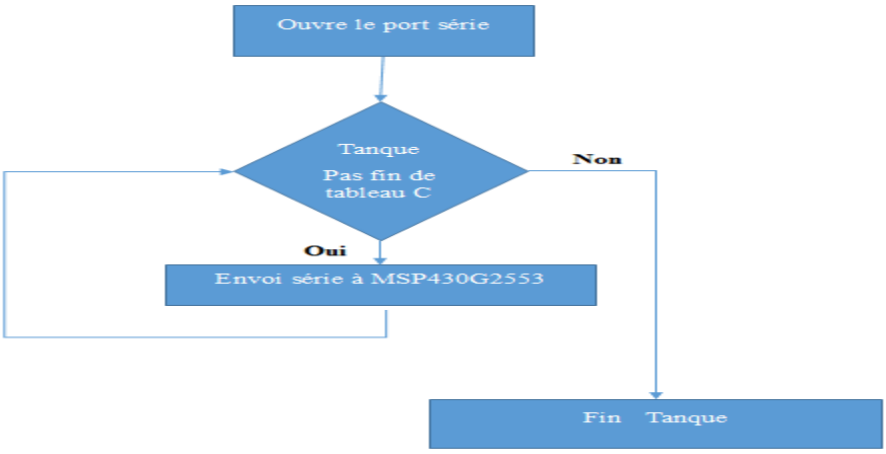


Figure 45 : Organigramme : envoi série à MSP430G2553.

3.2.5.3. Lire les linges par MSP430G2553 et les stocker dans un tableau :

1. 1e MSP430G2553 écoute le port série, attend la réception du caractère.
2. A la réception du caractère il le stocke dans buffer jusqu'à la du caractère de fin ligne (**"/n**), il arrête la réception des caractères.
3. Stocke le contenu de buffer dans un tableau.
4. Vérifie si il s'agit de la dernière ligne ou Non, lorsqu'il est découve que :

- c'est la dernière ligne, le programme s'arrête.
- Si ce n'est pas la dernière ligne de G-code, le processus de réception des caractères se poursuit.

L'organigramme ci-dessous décrit notre algorithme de programme :

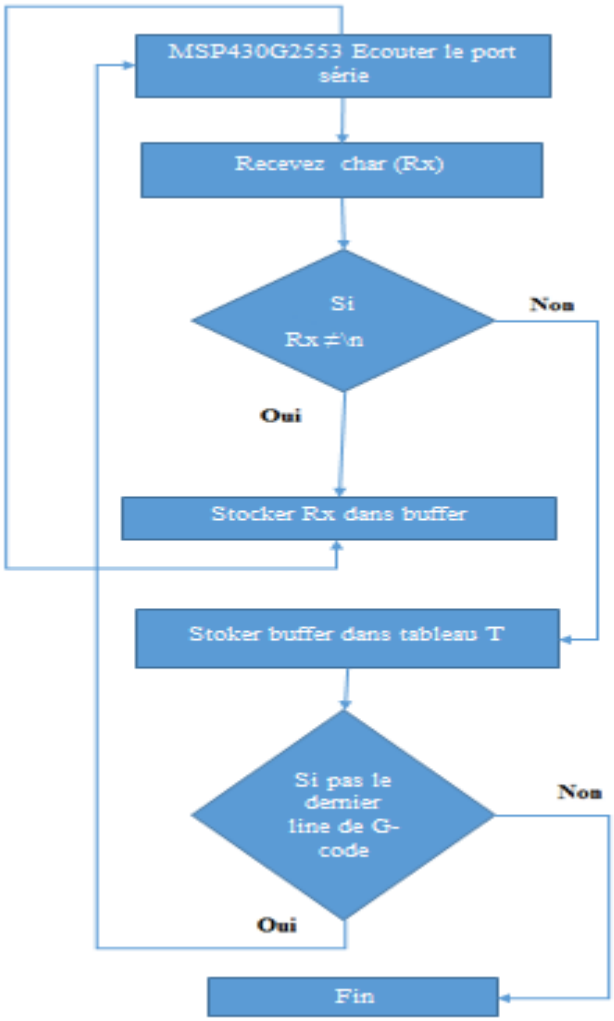


Figure 46 : Organigramme : lecture des linges par MSP430G2553 et leur stockage dans un tableau.

3.2.5.4. Analyser le tableau de G-code par MSP43G2553 :

Après la réception de toutes les lignes de G-code et que le tableau soit remplis, nous montrons, maintenant comment analyser et traiter les lignes de G-code et dire à la machine quoi faire. Ceci est illustré via notre code **« procesecommande »** en langage **C** suivant qui explique comment analyser et traiter G-code ligne par ligne

```
Void procesecommande{
  char str[20];
char T[7][20]={"G91 G17 G21","G0  X0 Y0 Z2","G0 X10 Y10 Z-2","G1 X10 Y10",
"G2 X0 Y-5 J-2.5","G3 X5 Y0 I2.5","M30"} ;
for (int i=0;i<7;i++){
  strcpy(str,T[i]);
   char* token=strtok(str," ");
    int cmd=atoi(token+1);
token= strtok(NULL," ");
if(str[0] == 'G')
                        {
  if (cmd<4){
  while(token != NULL)
                   {
 switch(*token)
                    {
case 'X': Xgcode = atof(token+1);   printf("XgCODE %f\n", Xgcode);");      break;
case 'Y': Ygcode = atof(token+1);   printf("YgCODE %f\n", Ygcode);break;
case 'Z': Zgcode = atof(token+1);   printf("Zgcode %f\n", Zgcode);break;
case 'J': J = atof(token+1);   printf("J %f\n", Jgcode); break;
 case 'I':  I= atof(token+1);   printf("J %f\n", Igcode);   break;
}
   token = strtok(NULL," ");    }
switch(cmd) {
     case  0:    Deplcment_Rapide() ; printf    (  deplcment Rapide ()   :\n");   break;
   case 1 :  Interpolation_linEaire ()   ; printf    (   Interpolation_ linEaire ()   :\n"); break;
   case 2:    Interpolation_arc_cw();  printf  (   Interpolation_arc_ccw()  :\n");   break;
```

```
   case 3: Interpolation_arc_ccw();   printf    (   Interpolation_arc_ccw()  :\n");     break;
                }
      if (cmd >3 ){
   switch (cmd)   {
case 90: abslu=1;    relatv=0;  printf (" selct cord absolue  :\n");
   while( token != NULL ){
     cmd1 [0] = atoi(token+1);
switch(cmd1[0]){
  case 17:        printf    ( selction plan xy  :\n");     break;
   case 18:           printf  (" selction plan xz  :\n");      break;
   case 19:           printf   (" selction plan zy  :\n");       break;
                 }
  cmd1 [1] = atoi(token+1);
      switch(cmd1[1]){
    case 20:         inch=1 ;mm=0 ;    printf (" selct unité en pouce :\n") ;      break;
    case 21:         inch=0;mm=1 ;      printf (" selct unité en mm :\n") ;     break;
                      }
       token = strtok(NULL," ");
              }   break;
  case 4:        _pose_progrmme ();printf    (   pose progrmme ();     break;
}}}
  switch (cmd){
if(str[0] == 'M')   {
  case 2: case 30:    _fin_progrmme ();          break;
}}}}
```

Dans ce code **« procesecommande »** nous avons procédé comme suit :

- Copier une ligne de G-code et la coller en vecteur **str** à l'aide de fonction **strcpy**
- Séparer le vecteur (bloc g-code) **str** en mots à l'aide de la fonction **strtok,** renvoie le premier mot dans **tokene**
- Déterminer le numéro de commande à l'aide de la fonction **atoi**
- Aller au mot suivant à l'aide de code : **token= strtok(NULL," ")**

– Vérifier le type de commande qui représente le premier caractère de vecteur **str** :

  - Si le type commande est **G** et le numéro de commande inférieure à **4,** déterminer les cordonnés **(x,y,z,j,i)** à l'aide de fonction **atof**,, puis appeller la fonction possédant chaque numéro de commande:

- Pour 0 : appeler fonction de déplacement rapide.
- Pour 1 : appeler fonction d'interpolation linéaire.
- Pour 2 : appeler fonction d d'interpolation arc cw.
- Pour 3 : appeler fonction d d'interpolation arc ccw.

  - Si le type commande est **G** et le numéro de commande supérieure à **4,** appeler les autres fonctions préparatoires (sélection plan, sélection type de cordonnée, sélection unité…).
  - Si le type commande est M et numéro de commande est **30** ou **3,** appeler la fonction fin de programme.

– Utiliser une **boucle for** pour refaire les mêmes étapes pour chaque ligne.

– Pour vérifier que le code prossecommande est correct nous effectuons un test d'impression de (**Xgcode, Ygcode, Zgcode, j et i)** déterminé, et impression de chaque fonction appelée dans le code sur terminale I\O d'IAR.

Résultat (Figure 47).

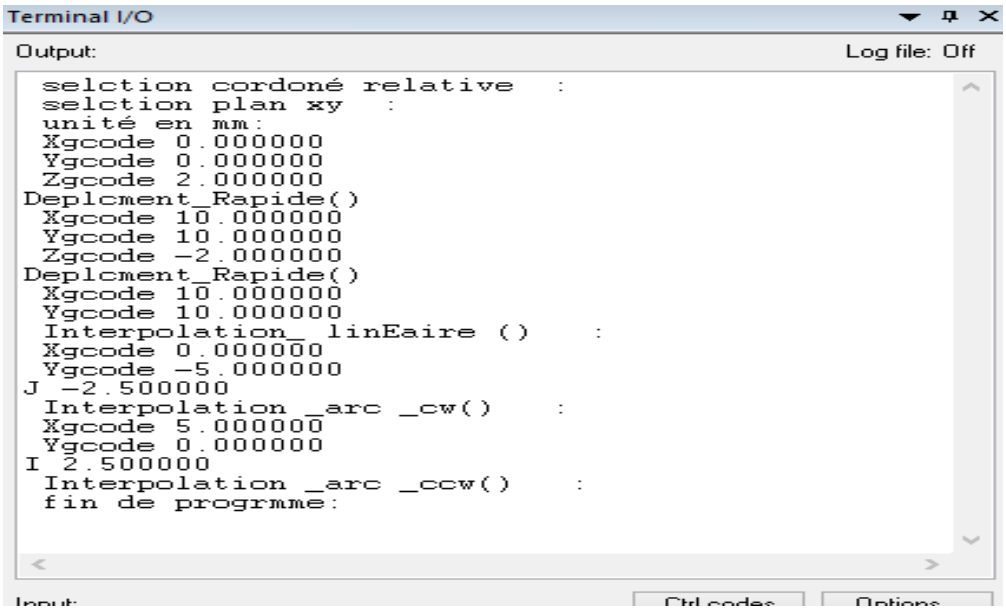


Figure 47 : Résultat du test d'impression du code « **procesecommande** ».

A.1 Algorithme de déplacement rapide

Le déplacement rapide est le déplacement des axes ***(x, y, z)*** de la machine à vitesse rapide, pour notre machine, nous procédons successivement, suivant que si le cas est :

- Le changement de l'outil de la machine :
- Déplacement de l'axe z (reculer)
- Déplacements de l'axe x
- Déplacement de l'axe y.
- La descente de l'outil de la machine :
- Déplacement de l'axe x,
- Déplacement l'axe y,
- Déplaçons l'axe z (avancer).

L'algorithme suivant explique la procédure de déplacement rapide des axes de la machine.

Algorithme : Procédure de déplacement rapide

```
procédure de déplacement rapide
détermine xi,yi,zi,xf,yf,zf       a partire de g-cod    // xi,yi,zi (point intiale) ;xf,yf,zf(point final)
 dx=xf-xi ; si dx>0  sens Mx=1  ;    'déterminer la sens de rotation de Mx(moteur de la axe x)
                sinon  sens Mx=0 ; fin si ;   'Mx=0 :sens négative ;Mx=1 :sens rotation positive
 dy= yf-yi; si  dy>0 sens My=1 ;   'déterminer la sens de rotation de My (moteur de la axe y)
              sinon sens My=0 ; fin si;    'My=0 :sens négative ;My=1 :sens rotation positive
dz= zf-zi ;
si dz>0                                  '  le cas  l'essor de l'outil de la machine
Mz Reculer ;                                  'appler fonction de moteur de la axe z reculer
Mx tourne (npas) ;
My tourne (npas) ;
 Sinon                                  ' le cas descendre De l'outil de la machine
  Mx tourne (npas)
  My  tourne (npas)
   Mz avancé                             appler fonction  moteur de la axe z avancé
Fin si
fin procédure
```

A.2 Algorithme d'interpolation linéaire

Se basant sur l'interpolation de Bressenham [29], nous avons créé l'algorithme de l'interpolation linéaire afin de tracer une ligne entre deux points (x_i, y_i) et (x_f, y_f). Pour se faire, nous traçons des petits segments entre les deux points (x_i, y_i) et (x_f, y_f), comme illustré sur la figure suivant.

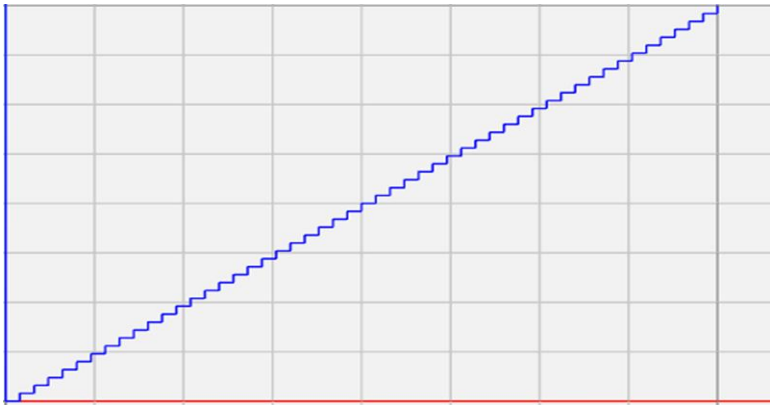


Figure 48 : Procédure : tracer une ligne (interpolation linéaire).

Pour tracer une ligne les changements des petits segments varie selon le type de la ligne, pour :

1. Une ligne horizontale les changements concerne la direction des X
2. Une ligne verticale les changements concerne la direction des Y
3. Une ligne diagonale les changements concerne les deux directions X et Y

Ces petits segments ont la même longueur qui est contrôlé par le pas de deux moteurs pas à pas. Dans l'algorithme suivant nous expliquons la procédure de tracer une droite (interpolation Linéaire) :

Algorithme : Procédure d'interpolation linéaire (tracer une ligne).

```
procédure Tracer_ligne    ''procédure d'interpolation Linéaire''
 Etap1 :   déterminé  (xi, yi) ;(xf,yf) .     ' déterminer à partir de g-code
 Etap2 : dx= xf-xi ; dy= yf-yi ;
  Etap 3 :
Si  dx>0 ; Sens Mx=1 ;            ' Sens de rotation du moteur x positif
Pas.x=  +p ;                      'p : longueur de petite segment
  Sinon   Sens Mx=0 ;             'Sens de rotation du moteur x négative
   Pas.x= -p .                    'p : longueur de petite segment
   Fin si
```

```
si  dy>0 ;    Sens My=1 ;            'Sens de rotation du moteur y positif
  Pas.y= +p.                         'p : longueur de petite segment
Sinon Sens My=0;                   'Sens de rotation du moteur y négative
Pas.y=  -p.                        'p : longueur de petite segment
Fin si
 Etap4 :
  Si |dx|>|dy|
    Acm=0 ;
      Faire :
    xi=xi+Pas.x .
    Incrémenter(Mx)                ' moteur x tour par n pas
  Acm=Acm +|dy| ;
    si (Acm>=|dx|)
    yi=yi+Pas.y ;
    Incrémenter(My) ;              ' moteur x tour par n pas
   Acm=Acm- |dx|;
Fin si
Tant que ( xi≠xf)
Sinon
    Acm=0 ;
      Faire :
    yi=yi+Pas.y .
    Incrémenter(My)
     Acm=Acm +fabs(dx)
    si (Acm>=fabs(dy)
    xi=xi+Pas.x
    Incrémenter(Mx)
   Acm=Acm-fabs(dy);
  Fin si
  Tant que (yi≠yf)
  Fin si
Fin procédure
```

A.3 Algorithme d'interpolation arc

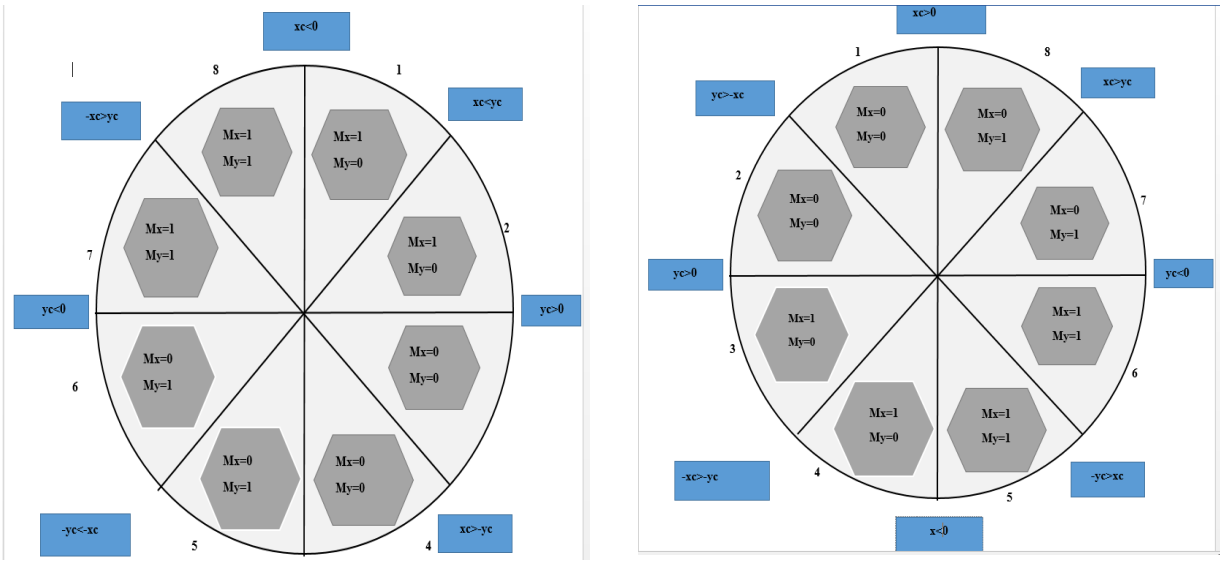
Toujours, en se basant sur l'interpolation de Bressenham [29], nous avons créé l'algorithme de l'interpolation arc afin de tracer un arc entre deux points (x_i, y_i) et (x_f, y_f). La figure 49 illustre

des exemples pour tracer des arcs entre deux points avec des rayons différents en utilisant l'algorithme de l'interpolation arc.

Figure 49 : Procédure : tracer un arc

Nous avons procédé comme suit tracer l'arc avec l'algorithme de l'interpolation arc :

- **Etape 1 :** nous déterminons (x_i, y_i) et (x_f, y_f), les coordonnées du cercle (x_c, y_c) et le rayon à partir de G-code. nous définissons une valeur de **p** qui représente la valeur d'incrémentation et décrémentation entre les cordonnés. Elle représente également la longueur des petits segments.
- **Etape 2 :** nous divisons le cercle en huit octets et nous désignons le sens de rotation de deux moteurs pour l'axe x et l'axe y sur chaque octets (**Figure 50)**.



1. Interpolation arc cw

Mx =1, My=1 : sens de rotation deux moteur cw

2. Interpolation arc ccw

Mx=0, My=0 : sens de rotation deux moteur ccw

Figure 50 : Définition du sens de rotation de deux moteurs dans chaque octet

- **Etape 3 :** nous calculons les valeurs de variable **d** sur un octet pour vérifier la position du point M.

- Si ***d > 0*** **M** au-dessus du cercle idéal.
- Si ***d < 0*** **M** au-dessous du cercle idéal.

Les valeurs de la variable **d** sont égales dans tous les octets du cercle, il suffit donc de calculer la valeur de la variable **d** d'un octet.

❖ **Etape 4 :** nous appelons les fonctions de deux moteur de l'axe x et l'axe y pour tracer un arc :

1. pour interpolation arc **cw (clock wise) :**

```
Tant que (xi ≠xf ou yi ≠ yf)
Tan que (xc<yc)                    '1er octets
si d<0 xc=+p ;xi=+p ; Incrémenter(Mx) ;
sinon  xc=+p ;xi =+p ;yc=-p ;yi=-p ; Incrémenter(Mx) ; Incrémenter(My) ;fin si ;
fin Tan que
Tan que (yc>0)                     '2em octets
si d<0  yc=-p ;yi=-p ; Incrémenter(My) ;
sinon  yc=-p ;yi=-p ; xc=+p ;xi =+p ; Incrémenter(My) ; Incrémenter(Mx) ;fin si;
fin Tan que
Tan que (xc>-yc)                   '3em octets
si d<0  yc=-p ;yi=-p ; Incrémenter(My) ;
 sinon  yc=-p ;yi=-p ; xc=-p ;xi =-p ; Incrémenter(My) ; Incrémenter(Mx) ;fin si ;
fin Tan que
Tan que (xc>0)                     '4em octets
si d<0 xc=-p ;xi=-p ; Incrémenter(Mx) ;
sinon  xc=-p ;xi =-p ;yc=-p ;yi=-p ; Incrémenter(Mx) ; Incrémenter(My) ;fin si ;
fin Tan que
Tan que (-yc>-xc)                  '5em octets
si d<0 xc=-p ;xi=-p ; Incrémenter(Mx) ;
sinon  xc=-p ;xi =-p ;yc=+p ;yi=+p ; Incrémenter(Mx) ; Incrémenter(My) ;fin si ;
fin Tan que
```

```
Tan que (yc>0)                    ‘6em octets
si d<0  yc=+p ;yi=+p ; Incrémenter(My) ;
sinon  yc=+p ;yi=+p ; xc=-p ;xi =-p ; Incrémenter(My) ; Incrémenter(Mx) ;fin si ;
fin Tan que
Tan que (-xc>yc)                  ‘7em octets
si d<0  yc=+p ;yi=+p ; Incrémenter(My) ;
sinon  yc=+p ;yi=+p ; xc=+p ;xi =+p ; Incrémenter(My) ; Incrémenter(Mx) ;fin si ;

fin Tan que
Tan que (xc<0)                    ‘8em octets
si d<0 xc=+p ;xi=+p ; Incrémenter(Mx) ;
sinon  xc=+p ;xi =+p ;yc=+p ;yi=+p ; Incrémenter(Mx) ; Incrémenter(My) ;fin si ;
fin Tan que
fin tan que
```

2. pour interpolation **arc ccw (contre clock wise)** : nous suivrons la même méthode utilisée pour l’interpolation **arc cw**.

A-4 Résultats et discussion :

- **Test des algorithmes**

Pour tester les trois algorithmes précédents, nous implémentons ces algorithmes en langage C. Puis nous les introduisons dans le code via « procesecommande » que a été expliqué précédemment, où nous utilisons les mêmes lignes G-code qui nous avons analysé.

Afin de valider les tests, nous suivons les étapes suivantes :

- Emulation des lignes de G-code en utilisant l'émulateur **Ncviewer**

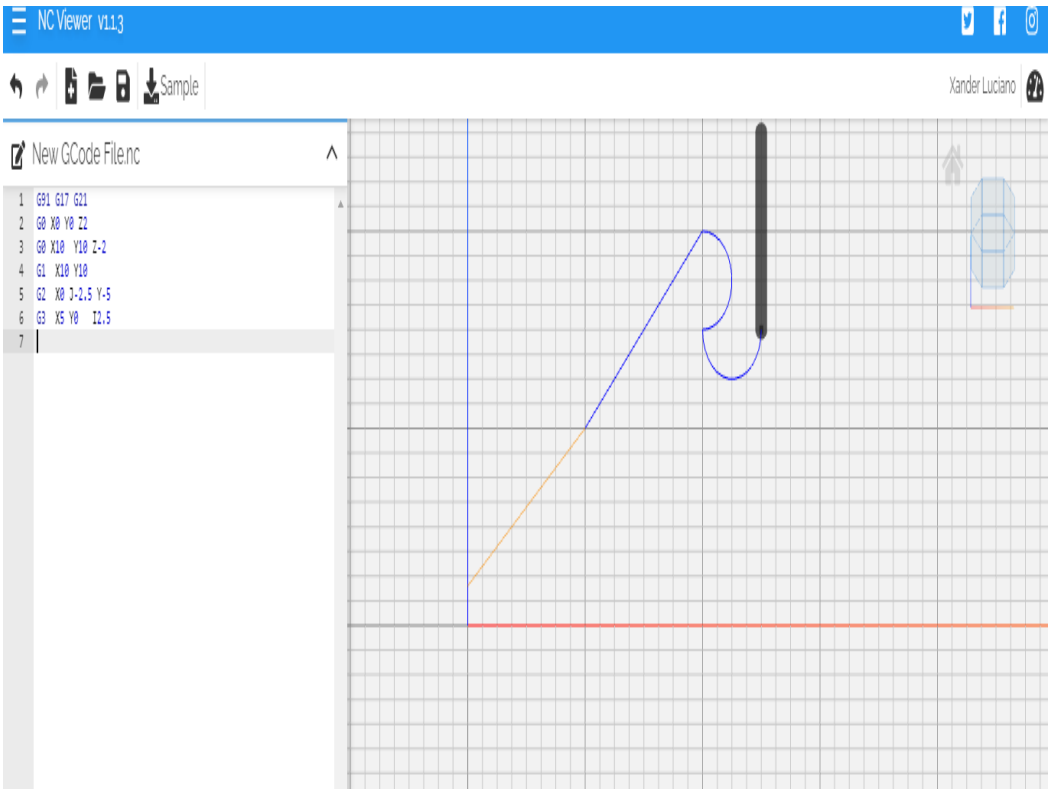


Figure 51 : Résultat de l'émulation des lignes de G-code

Remarque : A cause du confinement suite à la propagation du Covid 19 et l'arrêt total de moyens de transport publique les tests sur la machine étaient impossibles donc nous avons opté pour l'impression des instructions de déplacement et rotation des moteurs (axes).

- **Test de l'algorithme de déplacement rapide** : impression de :

a- ***(x_i,y_i,z_i) (x_f,y_f,z_f)***

b- le sens de rotation de deux moteur (**M_x,M_y**)

c- les fonctions des moteurs qui déplacent les axes (**X, Y, Z**).

Sachant que : 1mm = 25 pas

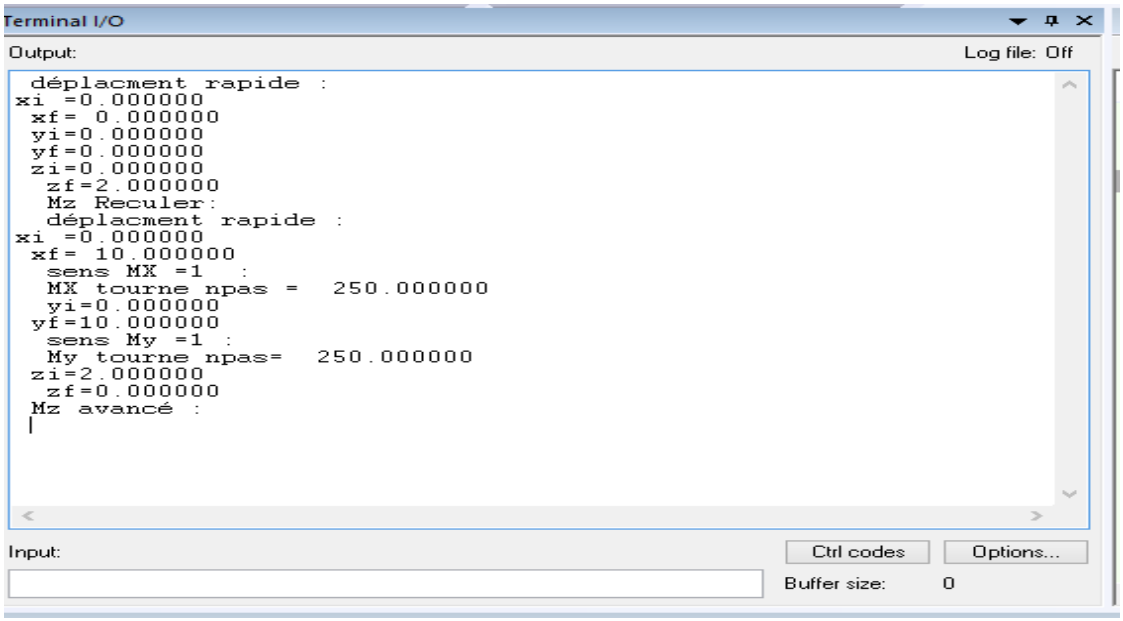


Figure 52 : Résultat du test d'algorithme de déplacement rapide.

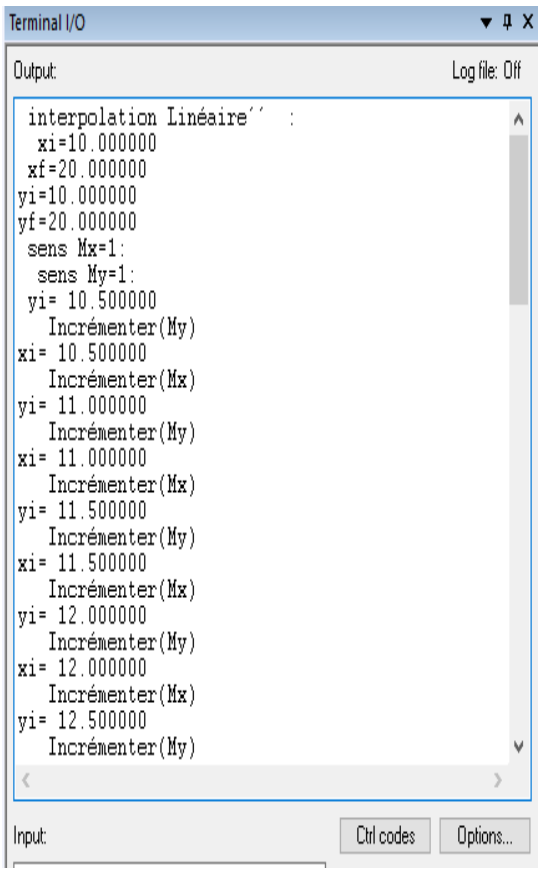
- **Test de l'algorithme d'interpolation linéaire** : impression de :

❖ (x_i,y_i) (x_f,y_f)

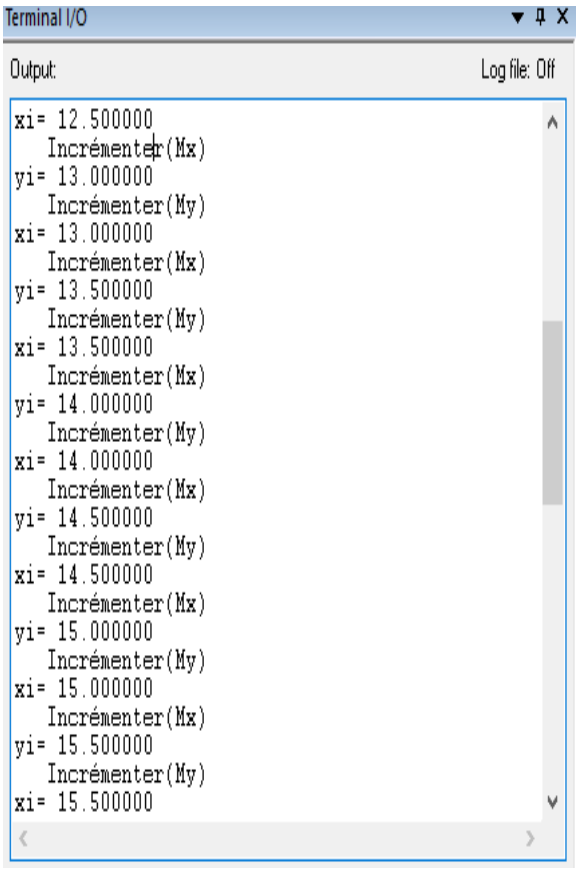
❖ le sens de rotation de deux moteurs (M_x,M_y)

❖ la fonction d'incrémentation des moteurs.

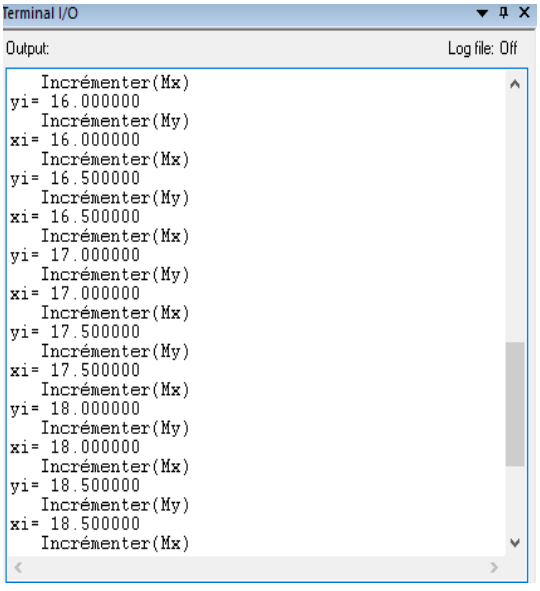
Sachant que : **p**=0.5mm, la longueur de petite segment



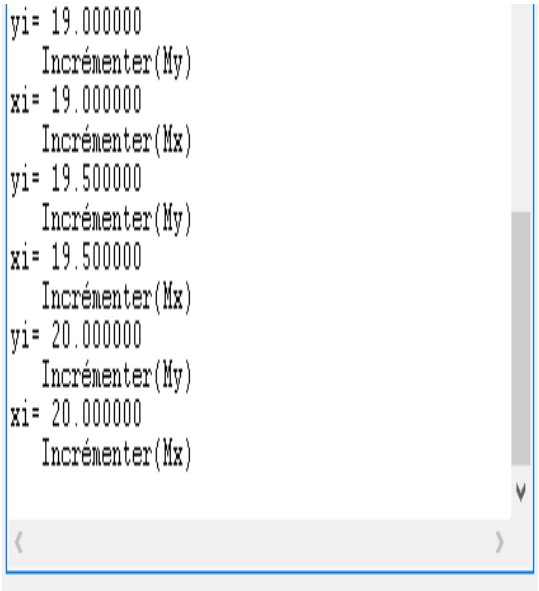
(1)



(2)



(3)



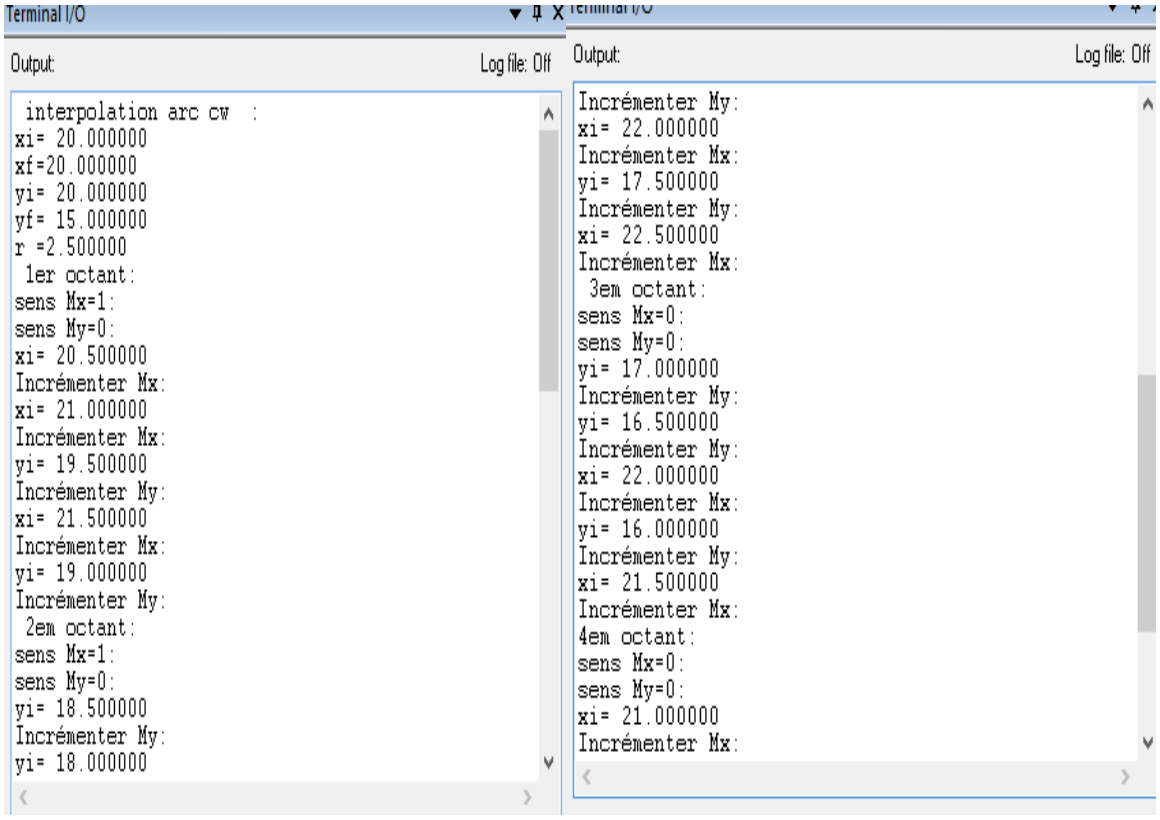
(4)

Figure 53 : Résultat de test d'algorithme d'interpolation linéaire.

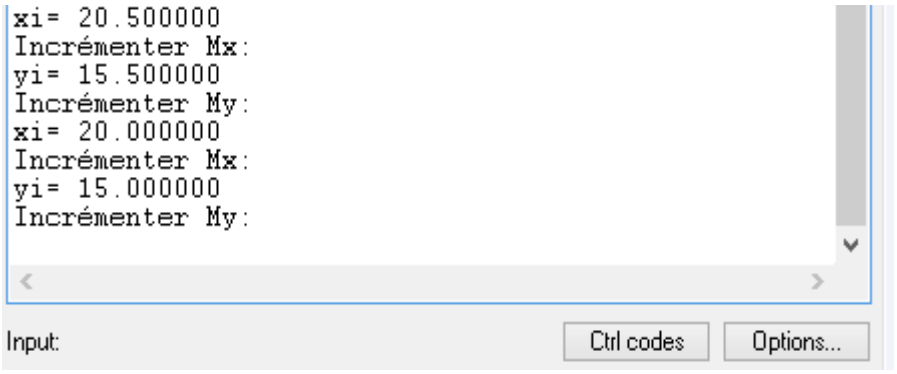
4 - Test de l'algorithme d'interpolation arc cw : impression de :

- ***(x_i,y_i) (x_f,y_f)***
- le sens de rotation de deux moteurs **(M_x,M_y)**
- la fonction d'incrémentation des moteurs et le rayon.

Sachant que : **p**=0.5mm, la longueur de petite segment



(1) (2)



(3)

Figure 54 : Résultat de test d'algorithme d'interpolation arc cw

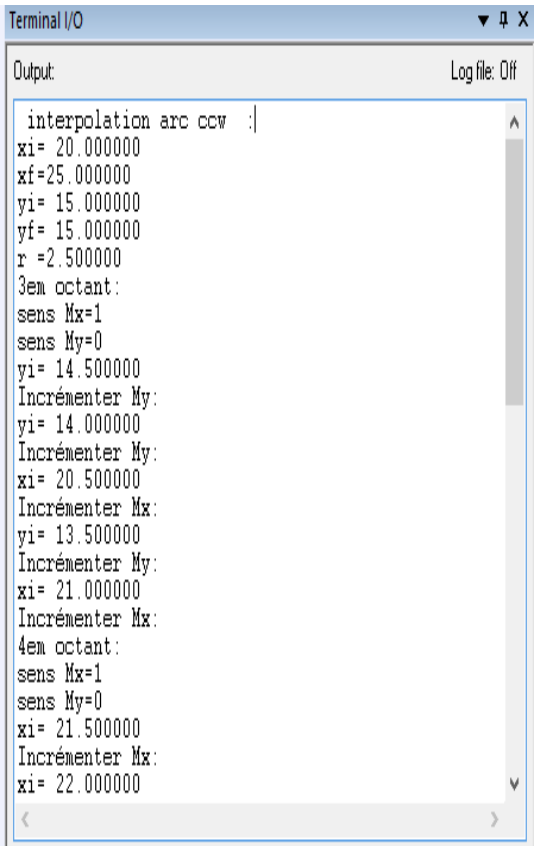
- **Test de l'algorithme d'interpolation arc ccw** : impression de :

❖ (x_i,y_i) (x_f,y_f)

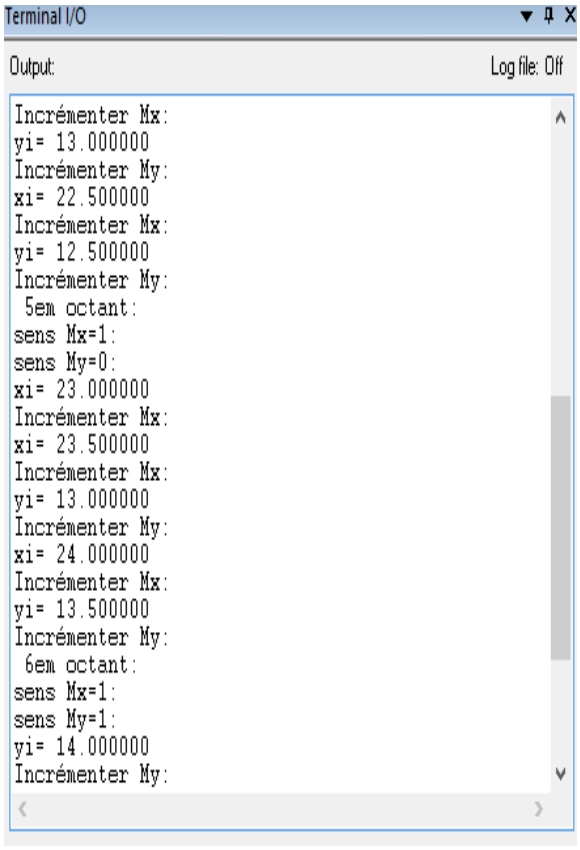
❖ le sens de rotation de deux moteurs (M_x,M_y)

❖ les fonctions d'incrémentation des moteurs.

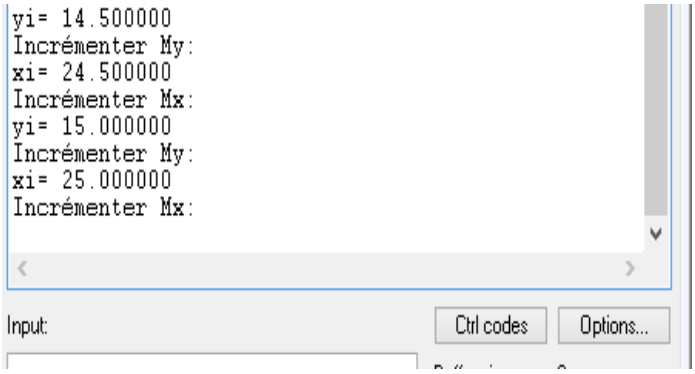
Sachant que : **p** = 0.5mm, la longueur de petite segment



(1)



(2)



(3)

Figure 55 : Résultat de test d'algorithme d'interpolation arc ccw.

❖ **Optimisations de programme**

Durant le développement de notre programme principal, on a rencontré le problème de dépassement de la capacité de stockage (RAM), pour cela nous proposons les solutions suivantes pour l'optimiser :

1. Utilisation des pointeurs
2. Favoriser l'utilisation des variables locales par rapport aux variables globales pour réduire l'espace dans la RAM.
3. Au lieu d'utiliser intégralement les bibliothèques de langage C nous utilisons les fonctions requises seulement e.g. Créé la fonction **fabs()**, au lieu d'appeler toute la bibliothèque math.h .

3.3.Conclusion

Dans ce chapitre nous nous somme intéressé à la programmation des machines de commandes numériques à l'aide du langage de programmation G-CODE. Nous avons expliqué la structure d'un programme et les fonctions des mots les plus utilisés. Après nous avons réussi la commande de moteurs par la génération des programmes en langage C qui contiennent des fonctions pour tracer des lignes ou des arcs. De plus une optimisation de programme final est effectuée pour surmonter le dépassement de la capacité de stockage.

Chapitre 4

Partie mécanique : conception et réalisation de la machine CNC

4.1.Introduction :

Ce chapitre est consacré à la conception et la réalisation de la machine CNC. Nous exposerons les composants de la machine à réaliser et nous présentons la conception des modèles des pièces et leur assemblage à l'aide du logiciel **solidworks.** Une fois la conception est vérifiée, on passe à l'étape de réalisation de la machine et des tests.

4.2.Description du projet :

Notre projet consiste à fabriquer un modèle simple à trois axes d'une machine CNC. Le travail est reparti en trois parties comme suit :

- La partie mécanique
- La partie électronique
- La partie programmation

Dans cette partie nous nous intéressons à la partie mécanique, les deux autres parties ont été détaillées dans les chapitres précédents.

Notre machine ressemblera à une fraiseuse CNC comme celle de la figure 56.

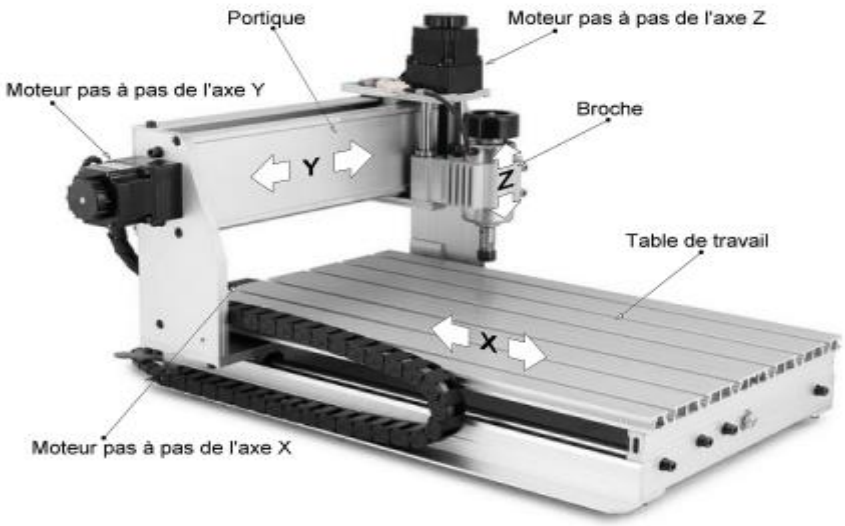


Figure 56 : Modèle d'une fraiseuse. [30]

Pour nous simplifier la conception et la fabrication, notre machine fonctionnera par déplacement de la table de travail en dessous de la broche. Malheureusement, la situation de confinement à cause du covid 19 et la fermeture des commerces nous a obligés de modifier la broche par un simple stylo pour vérifier le bon fonctionnement de notre machine.

Le système mécanique est assemblé de telle manière que le mouvement des 3 axes soit réalisé en utilisant des rails linéaires assemblés avec des roulements linéaires. Les moteurs sont montés chacun à l'axe qui est source de mouvement et qui agissent en fonction du signal de commande à partir du circuit électronique. Chaque moteur est relié à une vis sans fin pour chaque axe qui est chargé de transformer le mouvement du moteur de rotation en mouvement linéaire.

Le mouvement contrôlé de chaque axe est obtenu directement par la commande de la rotation du moteur. La vitesse du mouvement de chaque axe peut également être contrôlée par le contrôle direct de la vitesse du moteur, en donnant des signaux de commande nécessaires. Ainsi, la trajectoire de l'outil fixée à l'organe terminal est contrôlée dans chaque axe pour une action bien déterminée.

4.3. Composants mécaniques de la machine CNC

Dans cette partie, nous passons en revue les différents éléments fonctionnels de notre appareil qui assurent sa bonne performance.

4.3.1. Système vis-écrou :

Les trois axes X, Y et Z, permettant le déplacement de l'outil dans les six directions, sont entraînés grâce à la rotation de vis tournant dans l'un ou l'autre sens, selon la direction désirée. La vis sans-écrou est une tige filetée qui permet à l'aide d'un écrou de transformer la vitesse de rotation de moteur en un mouvement de translation et assure des déplacements de grandes précision.

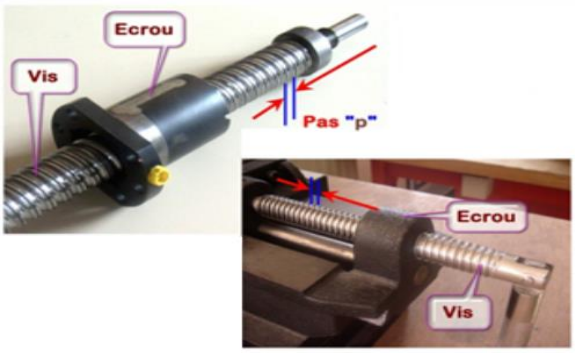


Figure 57 : Vis et écrou à billes.

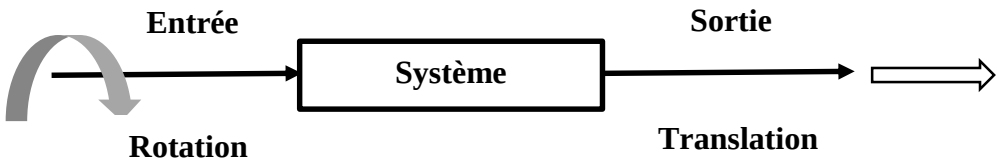


Figure 58 : Transformation du mouvement.

La cinématique du système vis-écrou est définie par le pas de la vis ***P*** en millimètre (mm) qui est la distance parcourue le long de l'axe de l'hélice pour un tour [31].

a. Vis à un filet **b.** Vis à plusieurs filets

Figure 59 : Types de vis à écrou

(1) Pour une vis a un seul filet (**Figure 59.a**) : ***P* = Pas de l'hélice = Pas du filetage**

(2) Pour une vis a plusieurs filets (**Figure 59 b**) : ***P_a* : Pas apparent** et le pas de l'hélice est : ***P* = *n*P_a** (n est le nombre des filets)

Un tour correspond à un déplacement de 1 pas de l'hélice.

Le déplacement est exprimé par la relation : ***L* = N_v. *n*. *Pa***

Où : ***Nv*** : étant le nombre de tours effectue

Dans notre projet, nous avons utilisé une tige avec 4 filets et un pas de 2mm. La distance parcourue le long de l'axe de l'hélice pour un tour est :

$$L = 1 \times 4 \times 2 = 8mm$$

Sachant que le moteur pas à pas utilisé fait 200 pas par rotation, la distance pour un seul pas est égale à la distance parcourue le long de l'axe de l'hélice pour un tour divisé par 200 :

$$L_{moteur} = \frac{L}{200} = \frac{8}{200} = 0{,}04\, mm/pas$$

4.3.2. Accouplement :

L'accouplement est un dispositif de liaison entre deux arbres rotatifs, permettant la transmission du couple. Il existe des accouplements dits « semi-élastiques » qui permettent de

rattraper de petits défauts d'alignement (typiquement les défauts d'usinage). Ces accouplements sont généralement constitués de deux parties rigides solidaires des arbres et d'une partie légèrement flexible qui rattrape les défauts d'alignement. [32]

Figure 60 : Accouplement.

Nous avons utilisé 3 accouplements pour accoupler les arbres de moteurs pas à pas (axes X, Y, Z) et les vis à écrous

4.3.3. Rail de guidage V-slot :

C'est un profil de rail linéaire de haute qualité en aluminium avec une rainure en V. extrêmement lisse sur les 4 côtés. Il est précis, facile à travailler avec et permet un contrôle illimitée de la conception par sa nature modulaire. Elle permet de guider le mouvement de translation des roulements selon les trois axes de la machine. L'axe Y est constitué de deux rails parallèles tandis que l'axe X et Z contiennent un seul rail chaque un. [33]

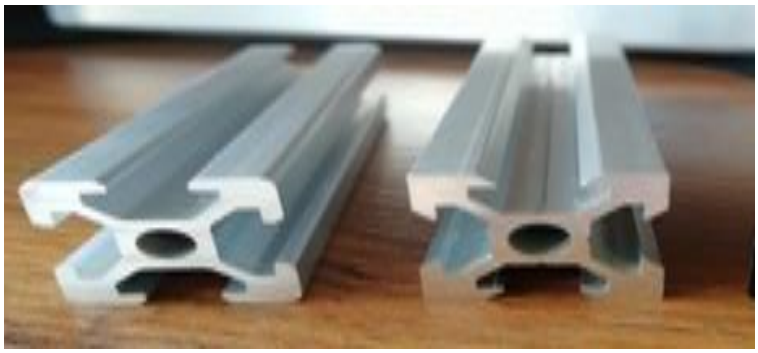


Figure 61 : Rail de guidage V-slot.

4.3.4. Support des axes de guidage :

Se relie aux rails pour fixer les axes :

Figure 62 : Support des axes de guidage.

4.3.5. L'axe de guidage :

Permet de guider le mouvement de translation des roulements selon les deux axes de la machine. Les deux axe X et Y est constitué de deux rails parallèles.[34]

Figure 63 : Axe de guidage**.**

4.3.6. Les roulements :

Les roulements sont des organes mécaniques servant à guider un arbre de transmission en rotation (l'axe X et Y). Ils assurent principalement [34] :

- Le positionnement de l'arbre par rapport à l'alésage.
- Une rotation précise avec un minimum de frottement

Figure 64 : Roulement

4.4.Cinématique d'un axe :

Chaque table présente quatre liaisons mécaniques qui sont :

- Liaison encastrement qui représente l'accouplement du moteur.
- Deux liaisons pivots qui assurent la rotation de la vis dans les paliers à roulements.
- Liaison hélicoïdale qui assure la rotation et la translation du système vis à billes.
- Quatre liaisons glissières qui assurent la translation des charriots de guidage.

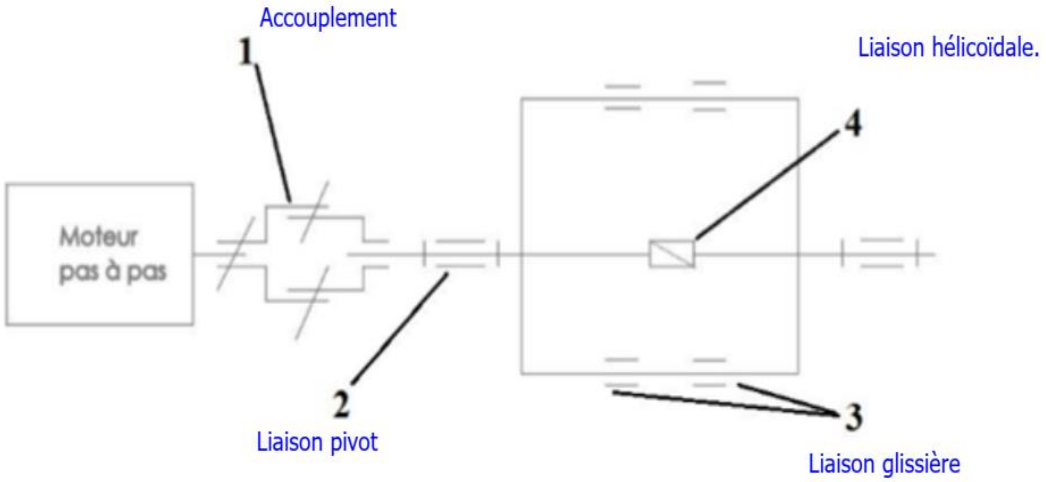


Figure 65 : Dessin représentant les liaisons sur l'axe. [35]

4.5.La Conception assistée par ordinateur (CAO) :

La CAO est un ensemble d'outils et de programmes informatiques permettant d'assister l'ingénieur dans la conception et la mise au point d'un produit. Un système de CAO permet de représenter et d'étudier le fonctionnement d'un objet sans l'avoir fabriqué réellement, c'est-à-dire en virtuel. Il existe un grand nombre de logiciels de CAO, par exemple [36] :

- SolidWorks.
- FreeCAD.
- OpenCASCADE.
- QCAD..

Vu que nous somme familiarisé durant notre formation avec SolidWorks, nous avons décidé de l'utilisé.

4.5.1. SolidWorks:

Créé en 1993 par l'éditeur américain éponyme, SolidWorks a été acheté le 24 juin 1997 par la société Dassault Systèmes [37].

SolidWorks est un logiciel de CAD (computer Aided Design) très puissant et facile à manipuler, dédié au design et la modélisation 3D. Il est très utilisé dans l'industrie pour la fabrication de différentes pièces mécaniques, hydrauliques, électriques ainsi que l'assemblage des pièces et la conception des machines et des appareils de A à Z.

Figure 66 : Logiciel SOLIDEWORKS

4.5.1.1. Fonctionnement :

SOLIDWORKS est un modeleur 3D utilisant la conception paramétrique. Il génère 3 types de fichiers relatifs à trois concepts de base : la pièce, l'assemblage et la mise en plan. Ces fichiers sont en relation. Toute modification à quelque niveau que ce soit est répercutée vers tous les fichiers concernés [37].

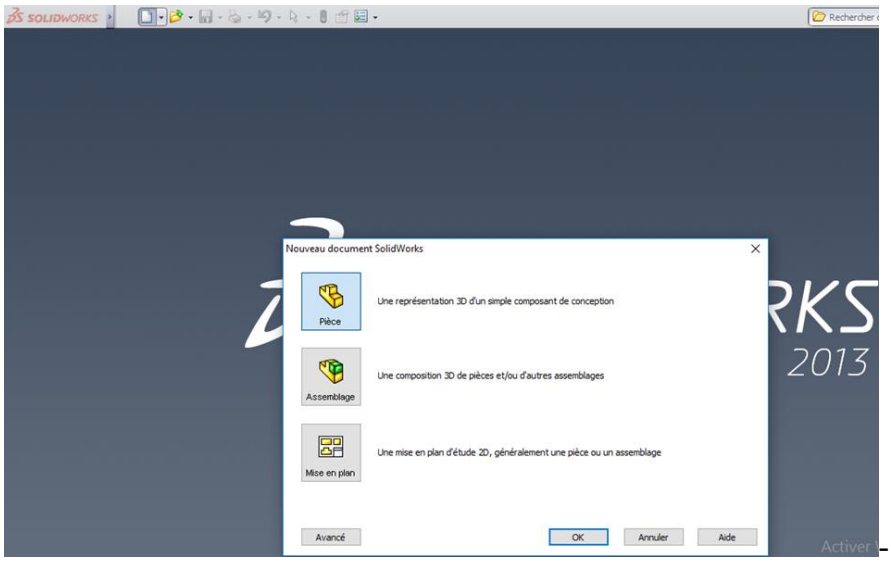


Figure 67 : Une fenêtre du logiciel SOLIDWORKS

4.5.1.2. Pièce :

La pièce est l'objet 3D monobloc. La modélisation d'une telle entité dépendra de la culture de l'utilisateur. Comme de nombreux logiciels conviviaux, SolidWorks permet d'aboutir à un même résultat apparent par des voies souvent différentes. C'est lors de la retouche de ces fichiers ou de leur exploitation qu'on appréciera la bonne méthode.

Une pièce est la réunion d'un ensemble de fonctions volumiques avec des relations d'antériorité, des géométriques, des relations booléennes (ajout retrait)... Cette organisation est rappelée sur l'arbre de construction. Chaque ligne est associée à une fonction qu'on peut renommer à sa guise [37]. Pour obtenir un volume, suivre les étapes :

- Définir une origine
- Choisir un plan
- Tracer une esquisse
- Générer un volume

4.5.1.3 Assemblages

Les assemblages sont obtenus par la juxtaposition de pièces. La mise en position de pièces est définie par un ensemble des contraintes d'assemblage associant, deux entités respectives par une relation géométrique (coïncidence, tangence, coaxialité...).

Dans une certaine mesure, ces associations de contraintes s'apparentent aux liaisons mécaniques entre les pièces. Le mécanisme monté, s'il possède encore des mobilités, peut être manipulé virtuellement. On peut alors aisément procéder à des réglages à l'aide des différents outils disponibles (déplacement composants, détection de collision ou d'interférence, mesure des jeux, etc.) [37].

Étapes de l'assemblage

- Ajouter des pièces dans un assemblage
- Déplacer et faire pivoter des composants dans un assemblage
- On peut positionner et orienter les composants à l'aide de contraintes qui créent des relations entre les composants [38]

4.5.1.3. Mise en plan :

Une fois les pièces ou assemblages créés, il est possible de générer automatiquement les mises en plan (représentation 2D) avec insertion automatique des côtes et liaisons entre les vues

2D et le modèle 3D. De plus, des fonctions d'habillage (texte, hachure, cotation…) permettent à l'utilisateur d'annoter rapidement un plan. Pour faire des mises en plan, il est tout d'abord nécessaire d'avoir des fonds de plan pour y projeter les dessins. Ces fonds de plans ont un format (A4, A3,…), une orientation (portrait ou paysage) et contiennent éventuellement un cartouche. Un certain nombre de fonds de plan de base sont proposés à l'origine, mais il est préférable, avant de commencer, de personnaliser les fonds de plan que l'on utilisera par la suite. [38]

Étapes de création d'une mise en plan

- Ouvrir un modèle de mis en plan et éditer un fond de plan
- Insérer des vues standards d'un modèle de pièce
- Ajouter des annotations de modèle et de référence
- Ajouter une autre feuille de mise en plan e. insérer une vue nommée f. imprimer la mise en plan [38]

Pour la conception de notre machine CNC nous utilisant SolidWorks en modélisation 3D. On modifie chacune des pièces afin de les assembler pour obtenir la forme finale de la machine.

Figure 68 : Fenêtre principale de SolidWorks

4.5.2. La conception de notre machine CNC avec SolidWorks

4.5.2.1. Partie pièces

La conception des pièces de la machine, avec SolidWorks, se déroule en 3 étapes :

- Sélection du plan de face,
- Conception de la forme en 3D en utilisant l'option « bossage extrudé »
- Création des différents détails qui caractérisent chacune de pièces.

- **Rail de guidage**

Nous avons défini la même base pour tous les rails mais avec des longueurs différentes.

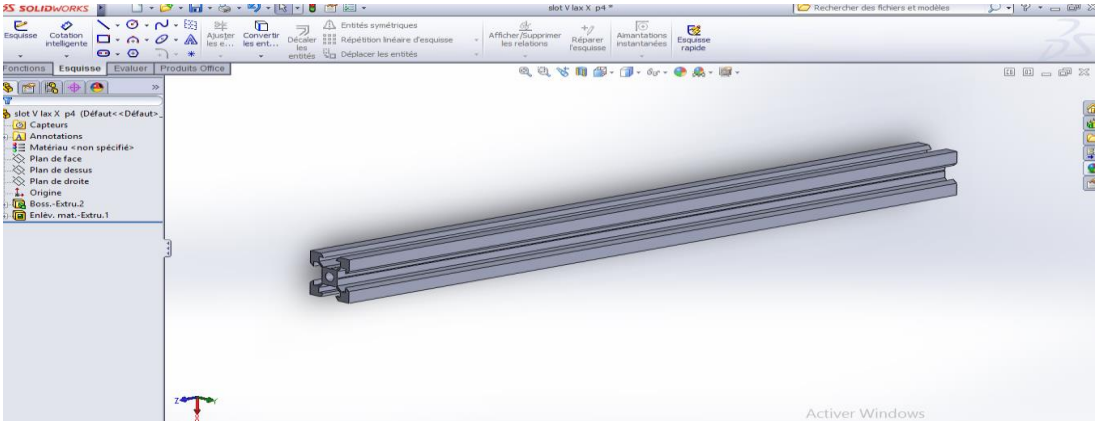


Figure 69 : Rail de guidage.

Trois pièces de rail de type V 2020 sont conçus. Les dimensions de base de chacune d'elles étant 20 x 20, et les longueurs sont les suivantes :

Tableau 9 : Caractéristiques de rails de guidage utilisés.

Axe	**Dimensions (**mm)	**Nombre**
Ox	20 ×20 ×280	5
Oy	20× 20× 320	2
Oz	20× 20 ×220	2

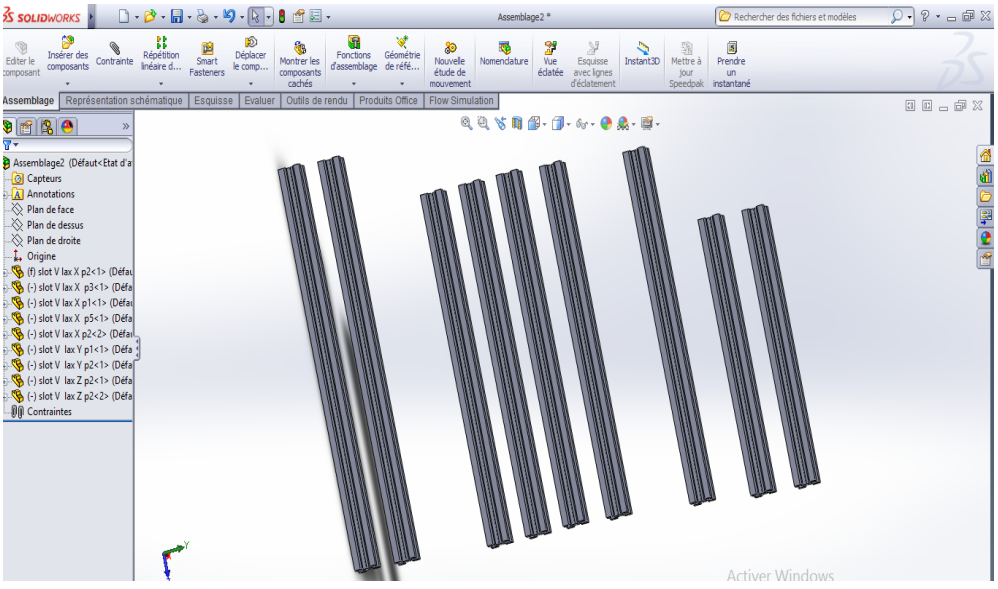


Figure 70 : Nombres des rails de guidage pour assemblage.

- **Support de cadre**

Des petits triangles sur la face latérale et deux trous pour les vis de fixation sont créés sur la forme en 3D de la pièce du Support de cadre.

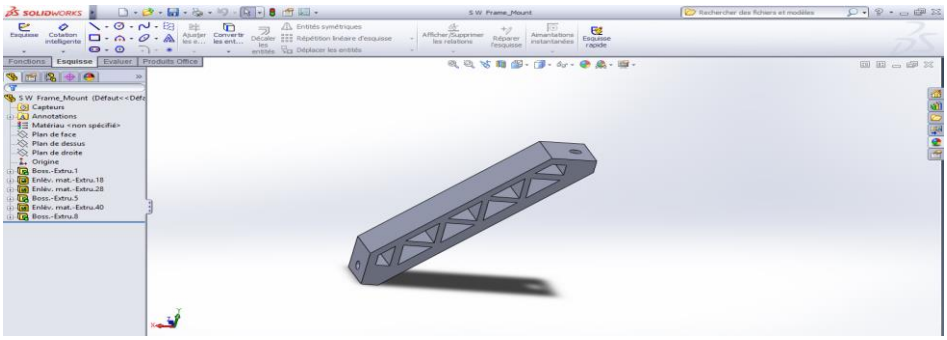


Figure 71 : Support de cadre.

- **Les axes du guidage**

Nous avons dessiné un cercle de 8mm de diamètre, après avoir défini la longueur de l'axe, nous avons obtenu la forme finale en 3D. Pour faciliter l'assemblage de la pièce avec le support, nous avons minimisé les extrémités de l'axe.

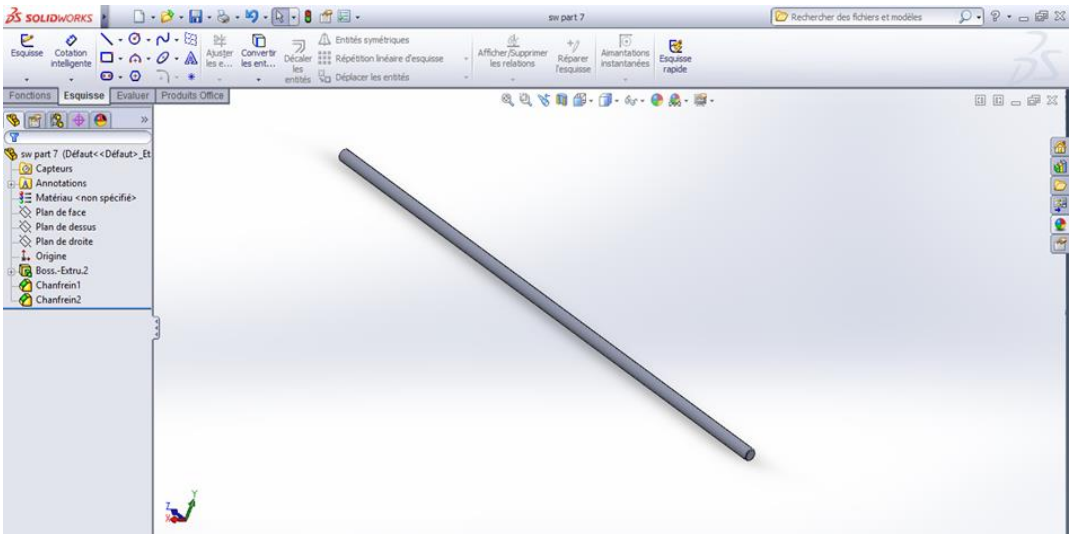


Figure 72 : Axe de guidage.

- **Support des axes**

Le support des axes contient une entrée pour l'axe, deux trous pour le fixer sur le rail et un canal pour limiter le mouvement de l'axe.

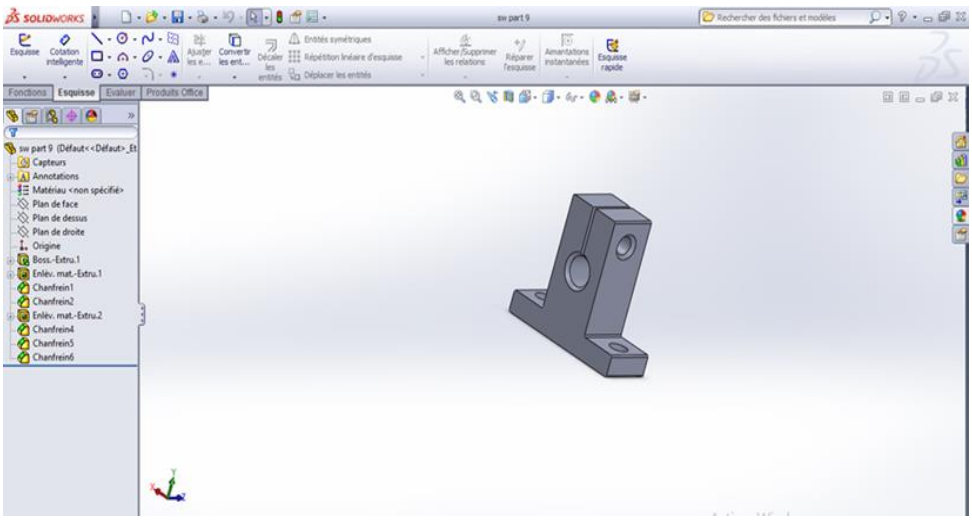


Figure 73 : Support des axes.

- **Support de moteur**

Il contient deux trous pour le fixer sur le rail et 5 points pour fixer le moteur dessus.

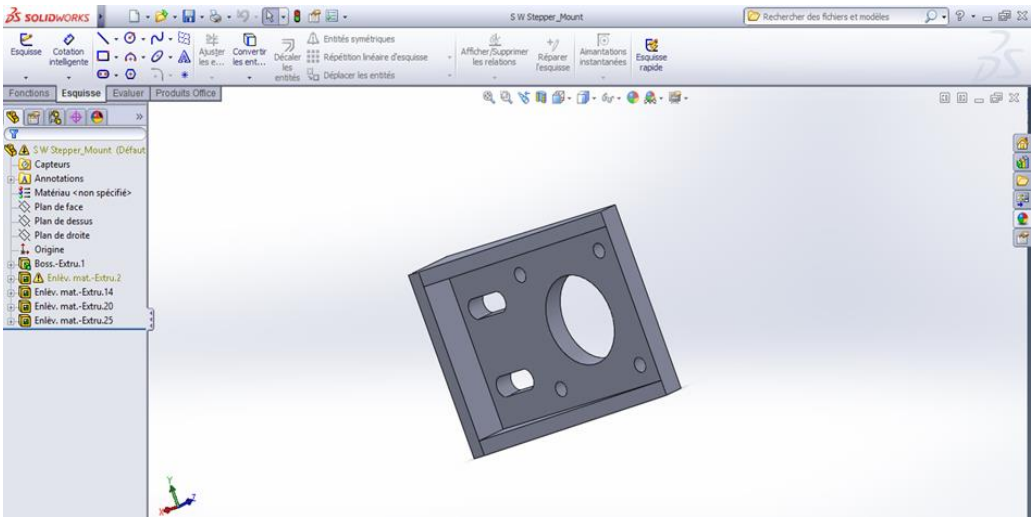


Figure 74 : Support de moteur pas-à-pas.

- **Moteur pas à pas Nema 17**

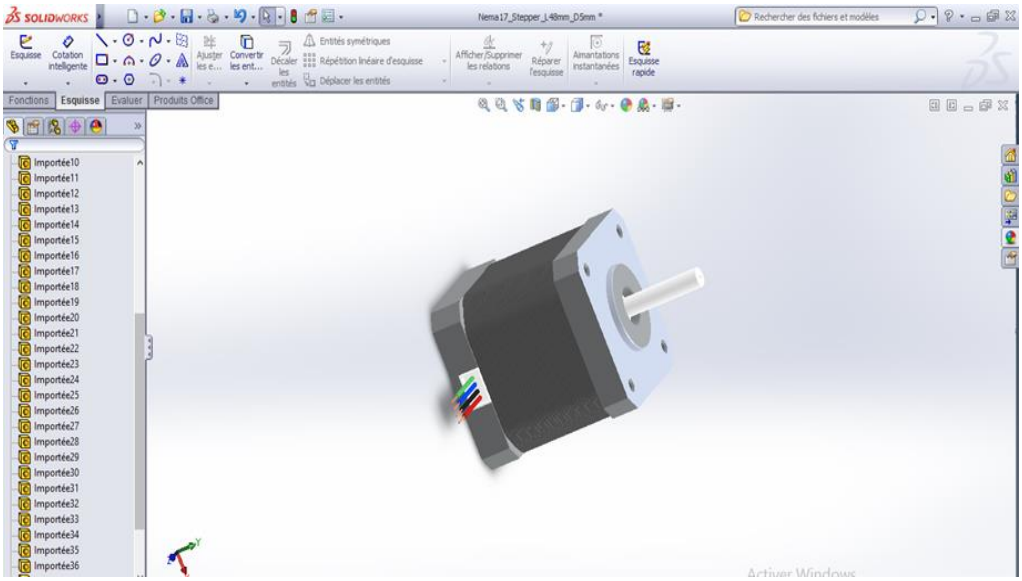


Figure 75 : Moteur pas-à-pas Nema 17.

– **Accouplement**

Cette pièce cylindrique assure une liaison entre le moteur et la vis sans fin-écrou, ceci permet la transmission de rotation.

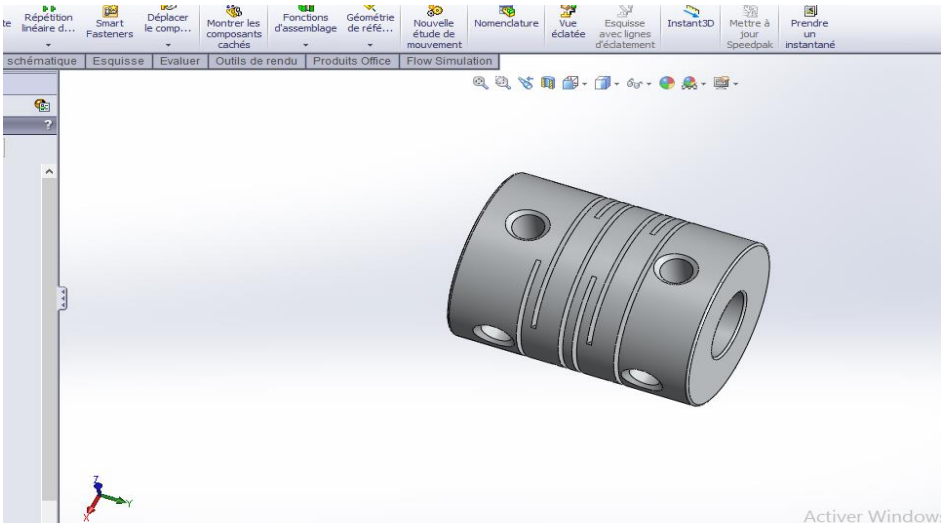


Figure 76 : Accouplement.

– **Vis sans fin (tige) :**

Cette vis est dessinée selon les caractéristiques présentées dans le tableau suivant

Tableau 10 : Caractéristiques de la vis utilisée.

Type	**Trapezoidal (mm)**
Nombre de filet	4

Diamètre de vis	8
Pas	2
Longueur	300

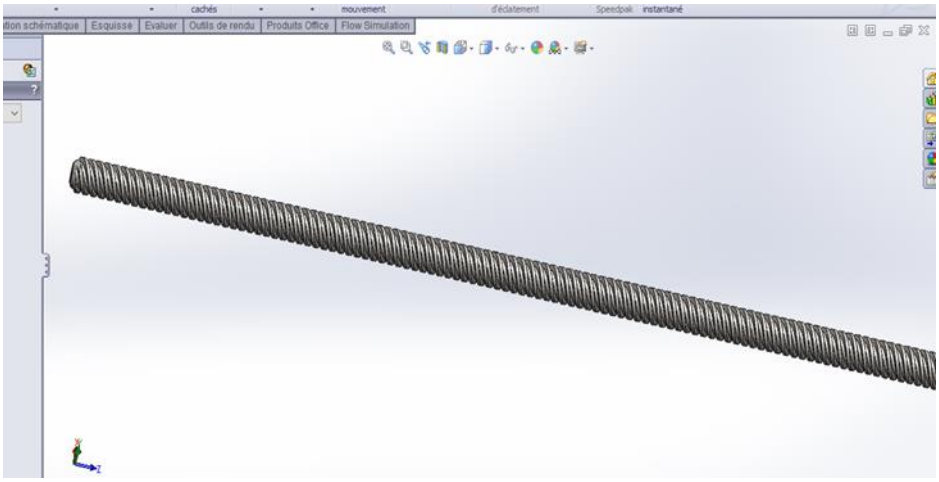


Figure 77 : Vis sans fin.

Dans la figure suivante, le trait en bleu représente la période de rotation de vis lié au moteur, cette période exprime en même temps la distance de déplacement de l'écrou.

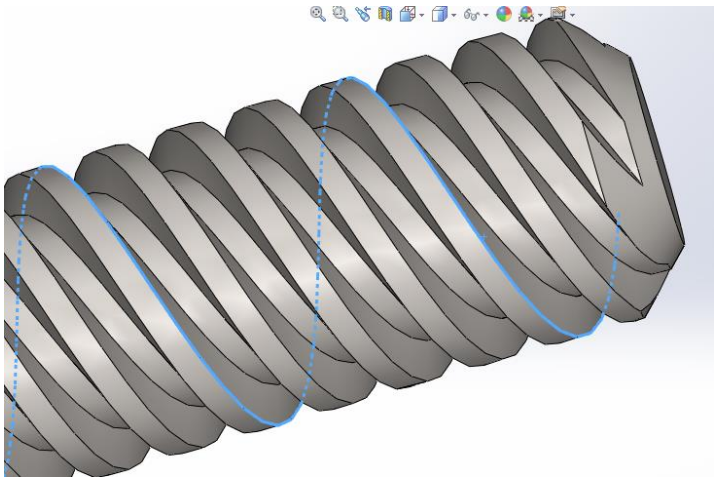


Figure 78 : Zoom de vis sans fin.

- **Écrou**

Conception de l'écrou a placé sur la vis sans fin. Deux écrous conçus un pour l'axe X et l'autre pour l'axe Y

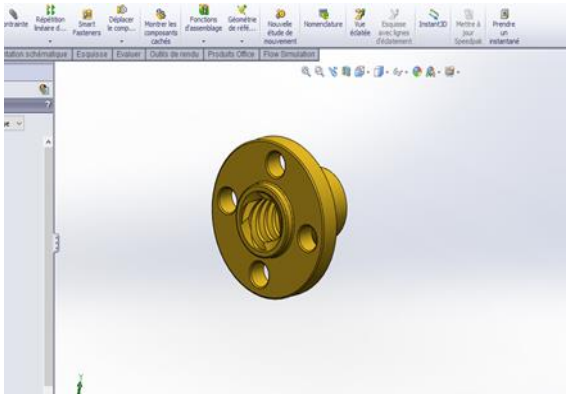


Figure 79 : Ecrou.

- **Support de vis sur l'axe X**

Conçu de manière à se fixer sur le rail et contient une entrée afin de fixer l'ensemble vis sans fin-écrou sur l'axe X.

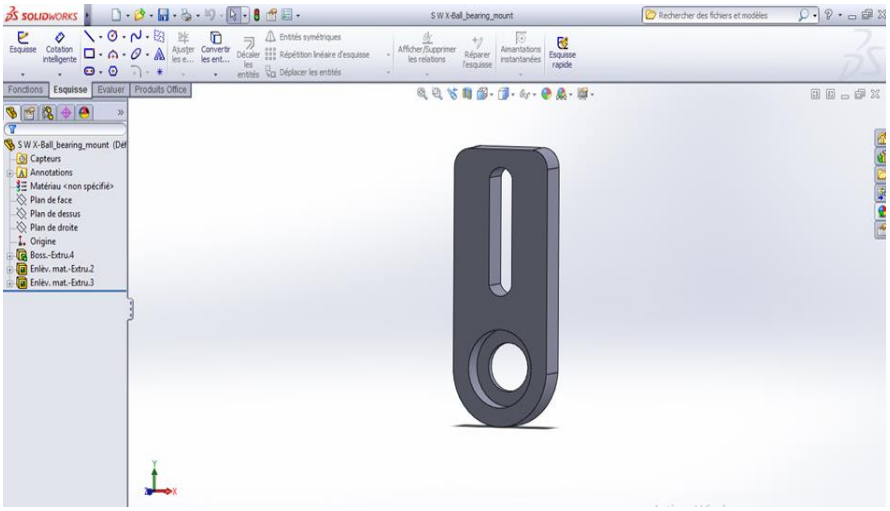


Figure 80 : Support de vis sur l'axe X.

- **Support de vis dans l'axe Y**

Conçu de manière à se fixer sur le rail et contient une entrée afin de fixer l'ensemble vis sans fin-écrou sur l'axe Y.

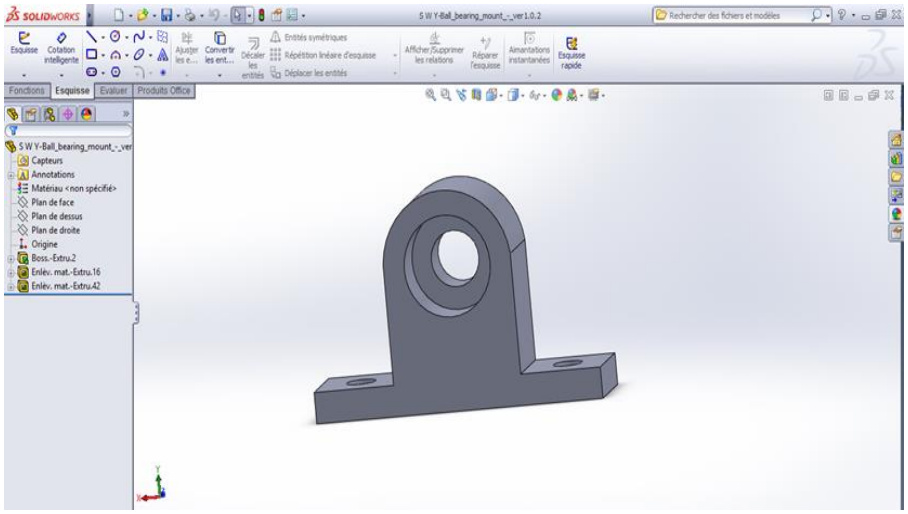


Figure 81 : Support de vis dans l'axe Y.

- **Carriage**

La pièce la plus compliquée, contient un trou pour la vis sans fin dans lequel se place l'écrou avec 4 points de fixation. De plus, elle contient deux entrées permettant le glissement des axes et des points de fixation latéraux pour le stylo.

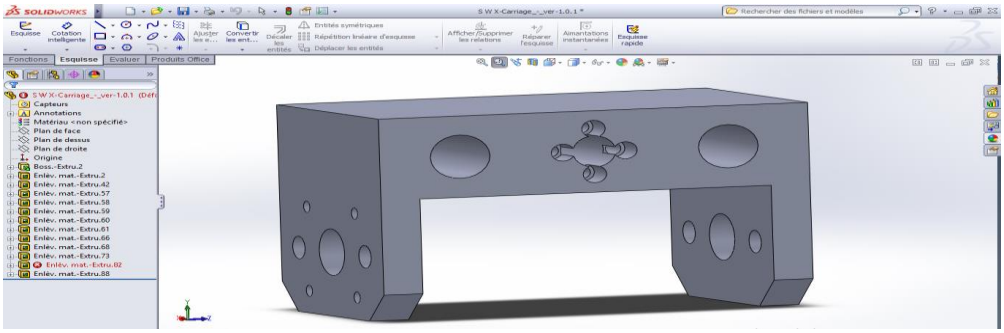


Figure 82 : Carriage.

- **Routeur Motion**

Conçu pour se placer sur les axes et permet le déplacement de la table sur l'axe Y.

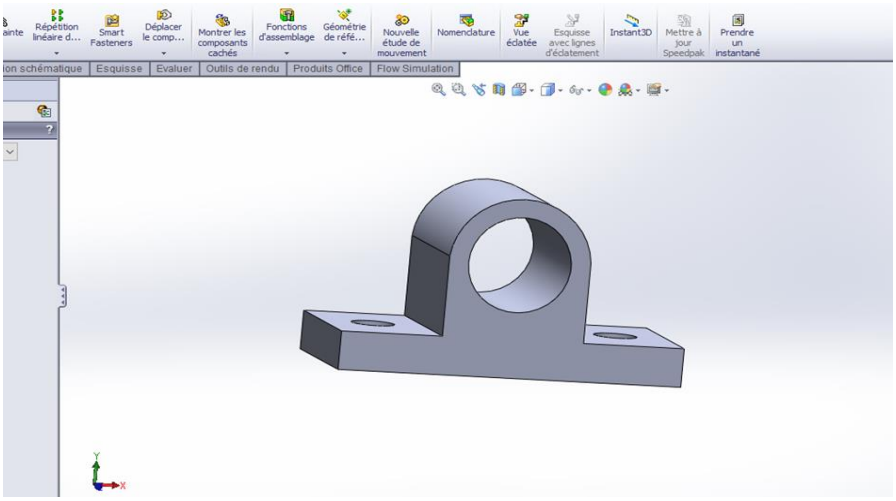


Figure 83 : Routeur Motion.

- **Palier linéaire**

Se place dans le routeur motion afin de permettre aux axes de se déplacer sans frottement.

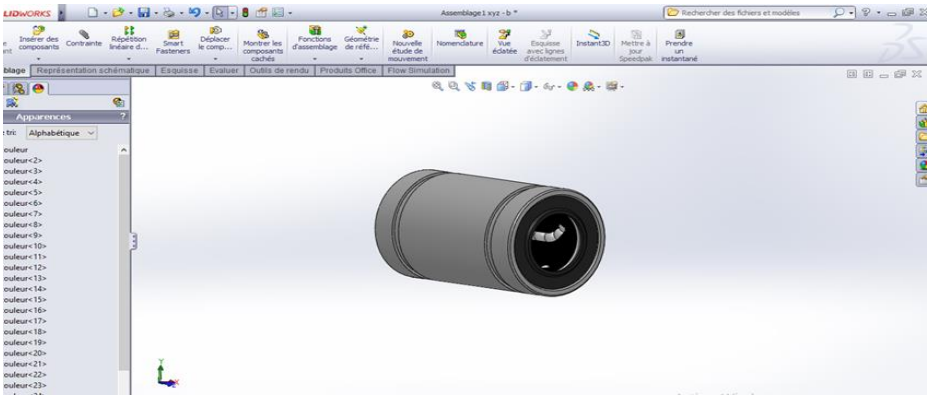


Figure 84 : Palier linéaire.

- **Nut (écrous)**

Contient un trou pour assurer le mouvement de la vis sans fin et quatre points pour la fixation de l'écrou à son intérieur.

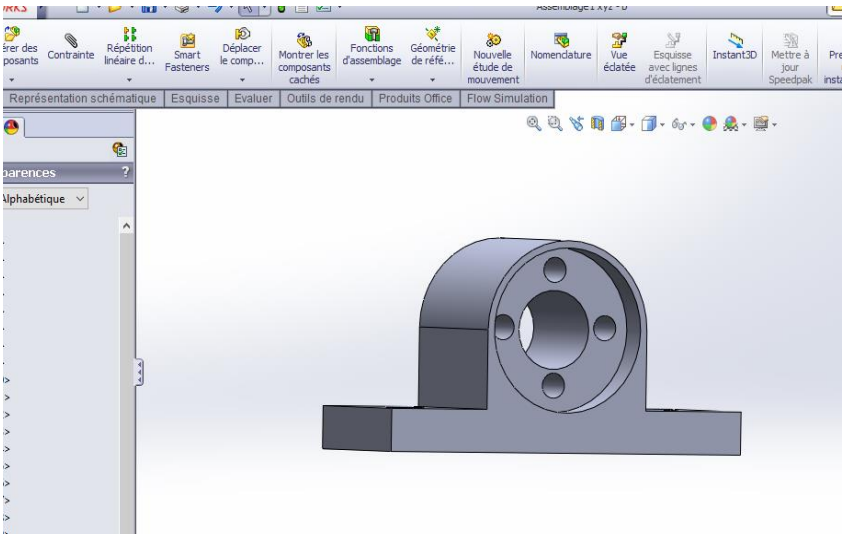


Figure 85 : Nut.

– **La table**

Pour qu'elle assure un mouvement sur l'axe Y, elle est conçue de façon à se fixer sur les quatre routeurs motion et les nuts.

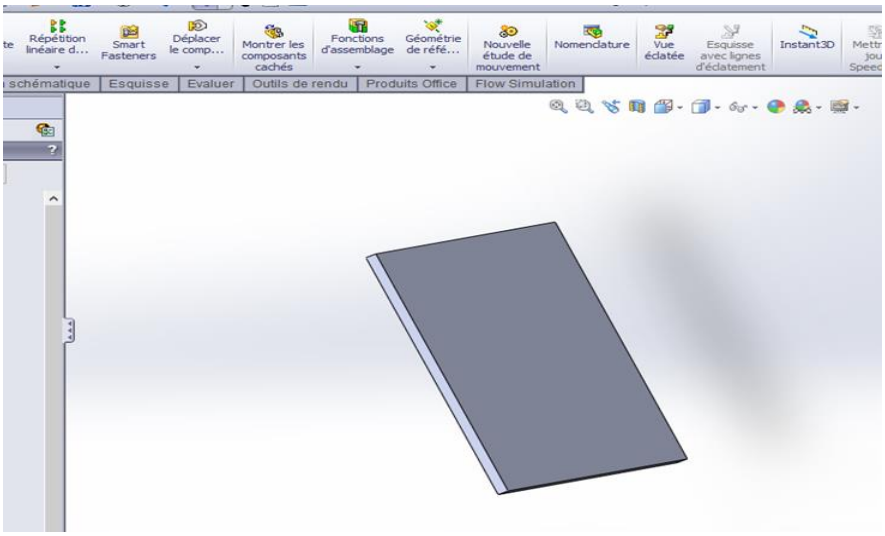


Figure 86 : La table du CNC.

4.5.2.2 Partie de l'assemblage

Etape 1 : le cadre du CNC

1. L'assemblage du cadre est constitué de neuf rails, tous reliés entre eux.

2. Deux supports cadre situées aux deux côtés.

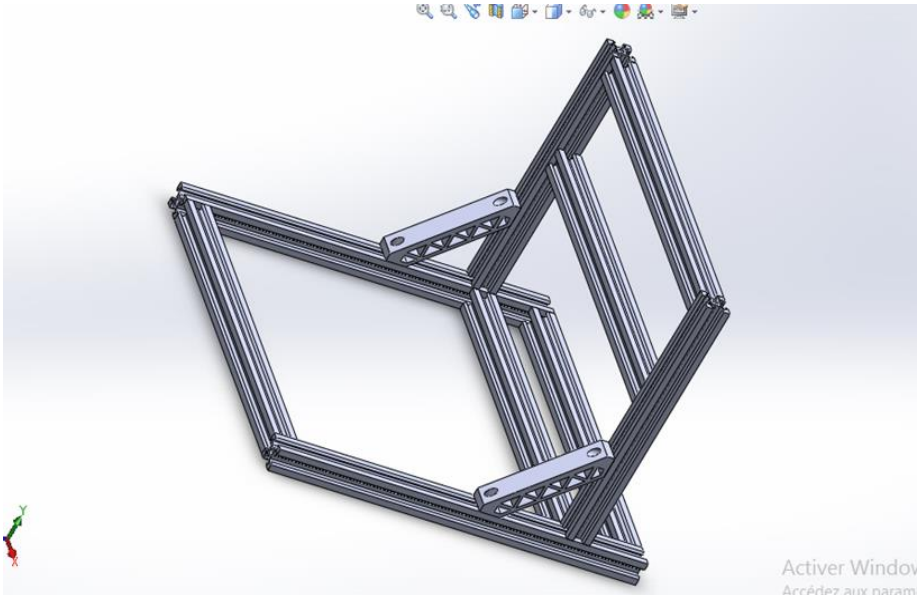


Figure 87 : Forme d'assemblage du cadre de CNC.

Etape 2 : **l'assemblage sur l'axe X**

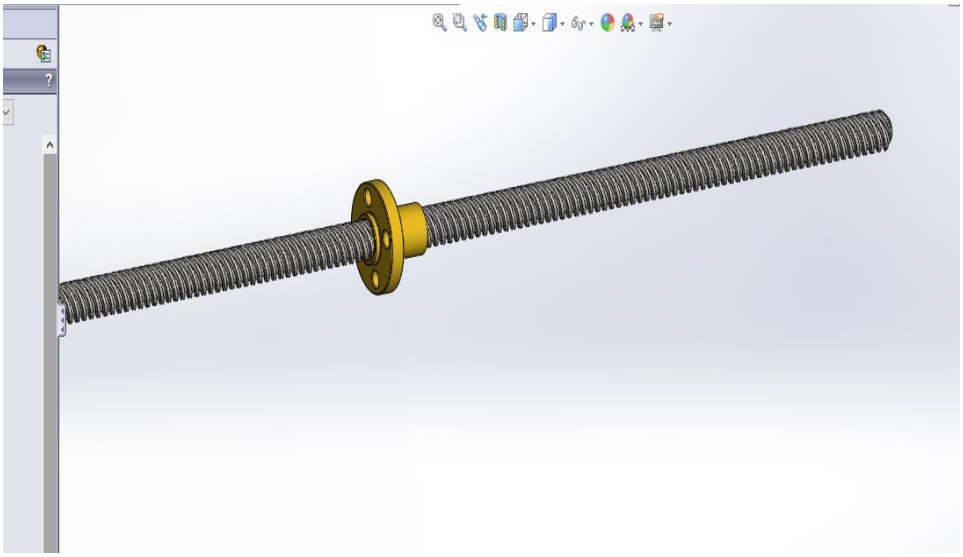


Figure 88 : Assemblage de vis-écrou dans l'axe X.

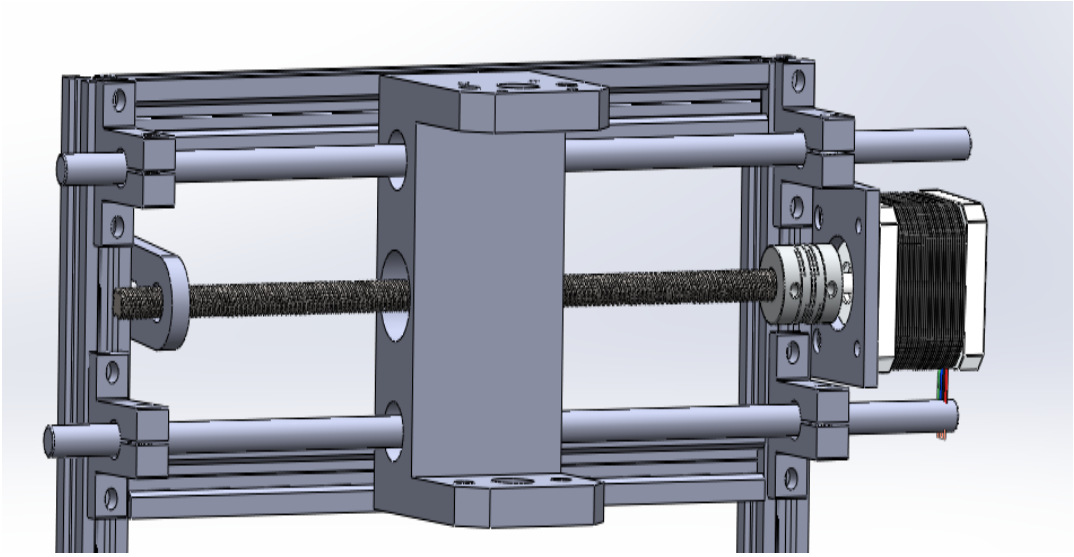


Figure 89 : Assemblage final dans l'axe X.

Étape 3 : l'assemblage sur l'axe Y

Nous procédons comme suit

- Assemblage de vis-écrous (de même que pour l'axe X : Figure 89)
- Assemblage Routeur Motion avec Palier linéaire

Le résultat final de l'assemblage sur l'axe Y est illustré sur la figure suivante :

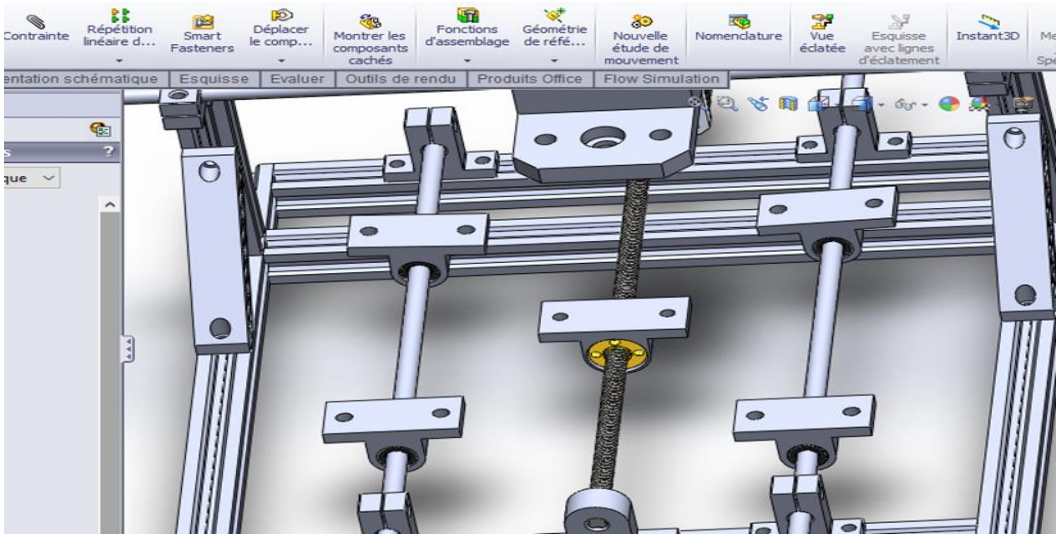


Figure 90 : Assemblage final sur l'axe Y.

Étape 3 : Assemblage du Moteur pas à pas nema17 sur les axes :

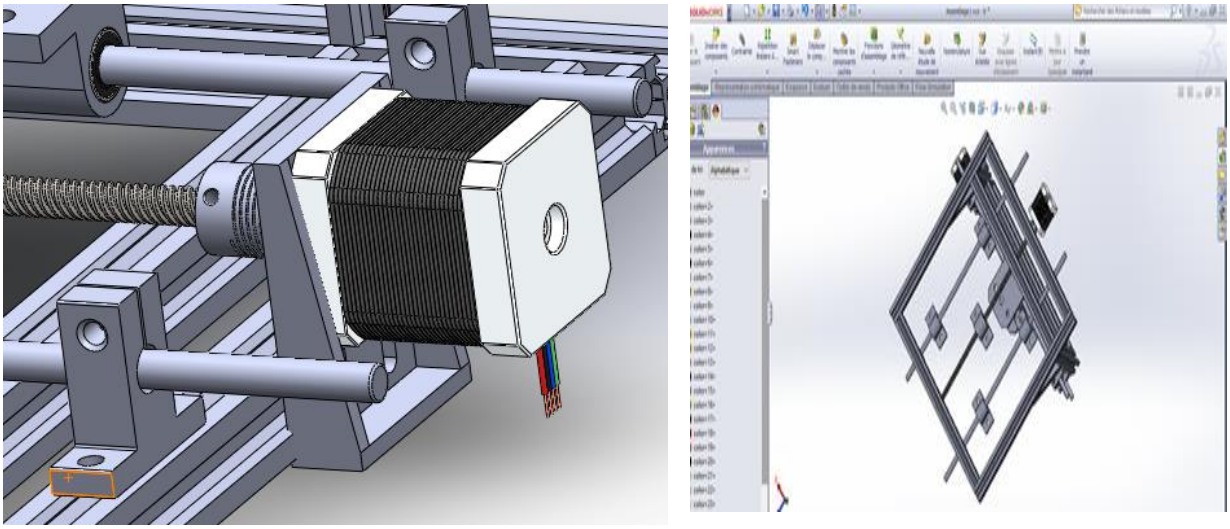


Figure 91 : Moteur pas à pas nema17 dans l'axe Y.

Etape 4 : l'assemblage final du CNC : Dans cette étape, nous plaçons la table du CNC sur l'axe Y

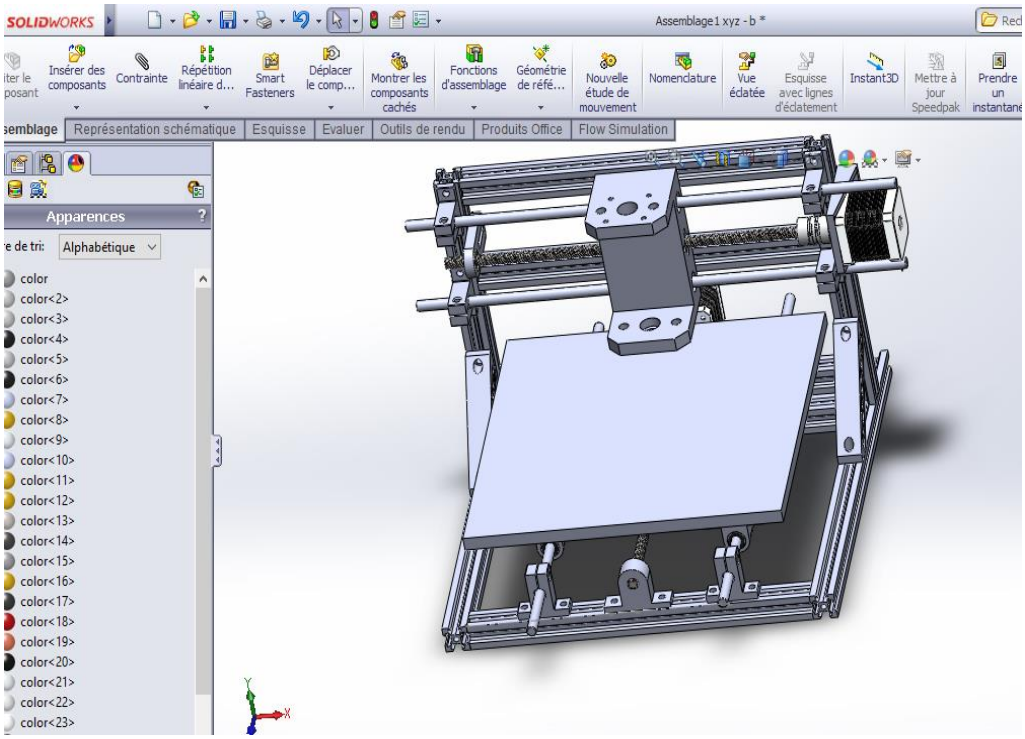


Figure 92 : Forme finale de la machine CNC.

4.6. La réalisation de la machine CNC

4.6.1. Problème de fabrication

Les principaux problèmes que nous avons rencontrés sont dus principalement à la propagation de la pandémie du covid 19. Suite au confinement nous avons rencontré différents difficultés :

- Difficulté d'obtenir les différentes pièces suite à la fermeture de commerces.
- Difficultés pour usiner nos pièces suite à la fermeture des tours et ateliers.
- Difficultés de déplacement suite à l'arrêt total des moyens du transport public

Malgré toutes ces difficultés, nous étions déterminés à réaliser notre machine. Pour cela nous avons apporté des modifications sur certaines pièces, achetés avant le confinement. Avec la collaboration **d'INNOVATION ACADMY** qui nous accueille dans son local, nous assistés par son staff et nous donnés accès à ses équipements nous avons réussi à réaliser notre machine.

Figure 93 : Pièces utilisées dans la réalisation.

4.6.2. Composants supplémentaires :

Comme mentionné plus haut, il existe de légères différences entre la forme de la machine en CAO et la forme réelle, en raison de l'addition de quelques pièces dont nous n'avions pas besoin dans la conception au niveau de SolidWork.

Pièce 1 : corner pour fixer les rails de guidage avec les vis.

Figure 94 : Corner.

Pièce 2 : Les rondelles sont utilisées pour supporter la pression des vis.

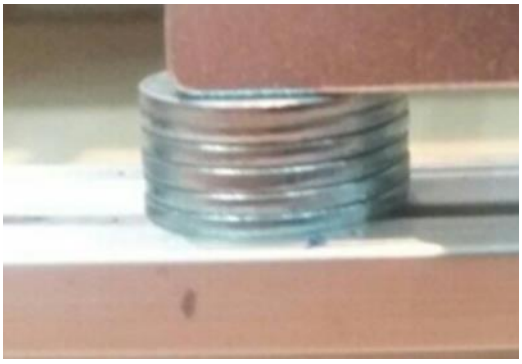


Figure 95 : Rondelles.

Pièce 3 : Routeur Motion SC8UU : nous avons remplacé le Routeur Motion imprimé en 3D par le Routeur Motion SC8UU, et nous avons réduit le nombre de pièces de 4 à 2.

Figure 96 : Routeur motion SC8UU.

4.6.3. La réalisation de l'axe X :

Fixation des différentes pièces de la partie fonctionnant sur l'axe X.

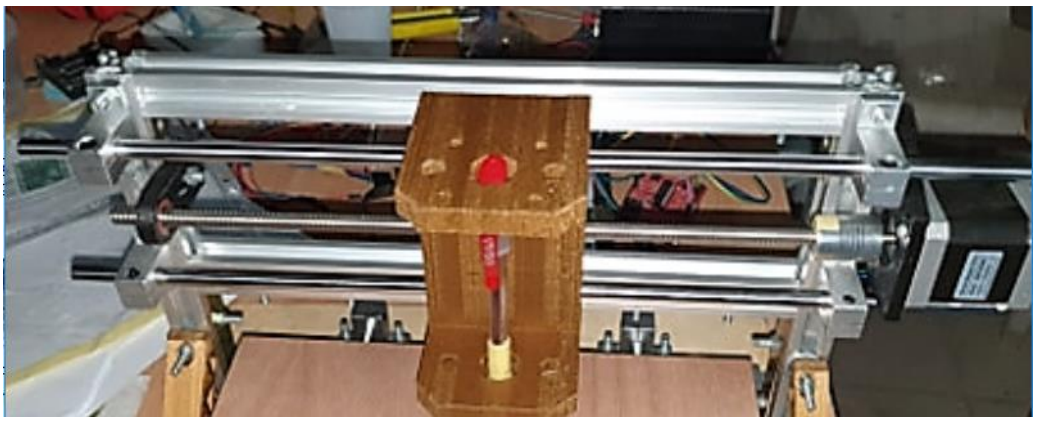


Figure 97 : Réalisation de la machine sur l'axe X.

4.6.4. La réalisation de l'axe Y :

Fixation des différentes pièces de la partie fonctionnant sur l'axe Y

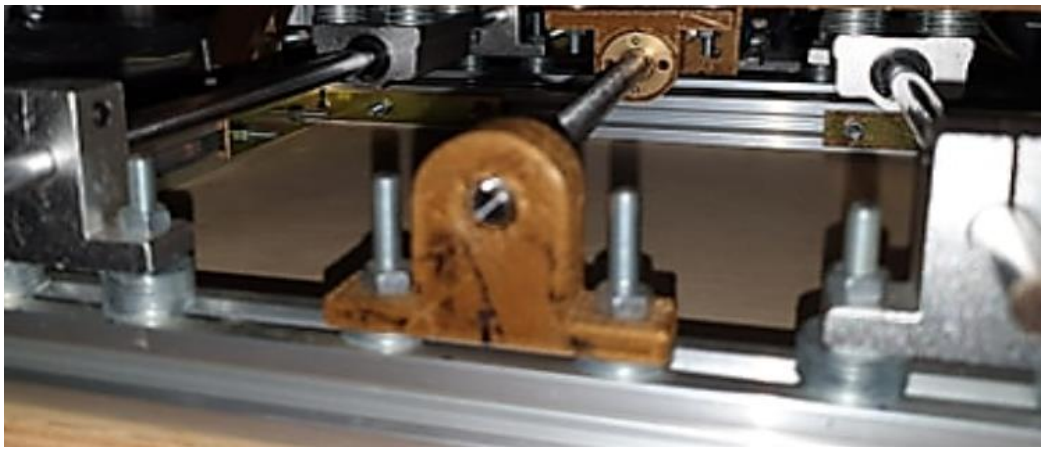


Figure 98 : Réalisation de la machine sur l'axe Y (vue de face).

Figure 99 : Moteur pas à pas dans l'axe Y (vue de l'arrière).

4.6.5. Circuit de montage

La dernière phase de notre réalisation et l'assemblage de la partie mécanique et la partie électrique. Le schéma de branchement ci-dessous représente le circuit électrique qui assure le fonctionnement de la machine, via :

a) Le branchement des moteurs pas à pas avec les drivers A4988.
b) Le branchement des moteurs avec microcontrôleur MSP430G2553,
c) Une source d'alimentation.

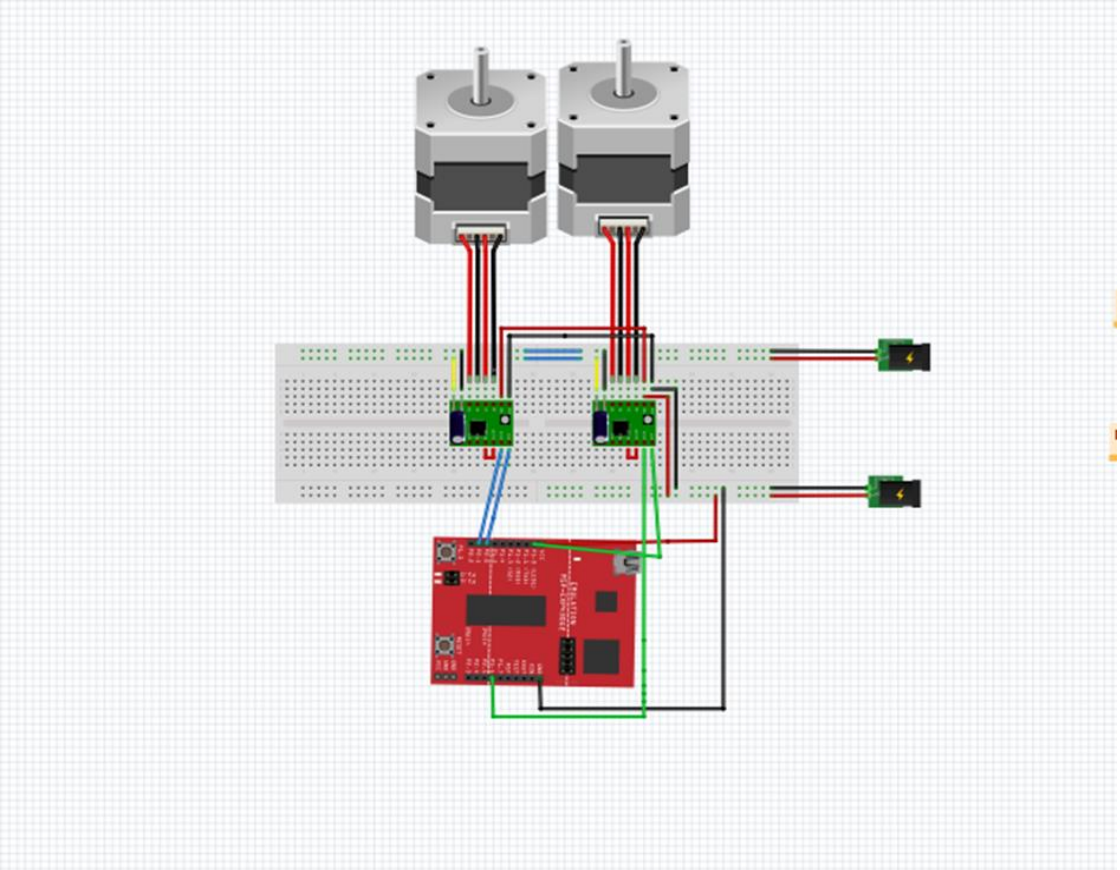


Figure 100 : Circuit de montage dans la machine CNC.

Figure 101 : Réalisations de la machine CNC et circuit de montage.

4.7.Test du fonctionnement de la machine

Pour assurer le bon fonctionnement de notre machine nous allons tester des algorithmes élaborés dans le chapitre 3.

La figure 102 montres bien que ces algorithmes sont valides. La distance que la machine a dessinée en ligne droite pour un cycle du moteur pas à pas est d'environ 8 mm, qui est la distance théoriquement calculée.

La légère différence entre eux est due à des défauts dans le moteur pas à pas, au driver A4988 ou à certains défauts mécaniques.

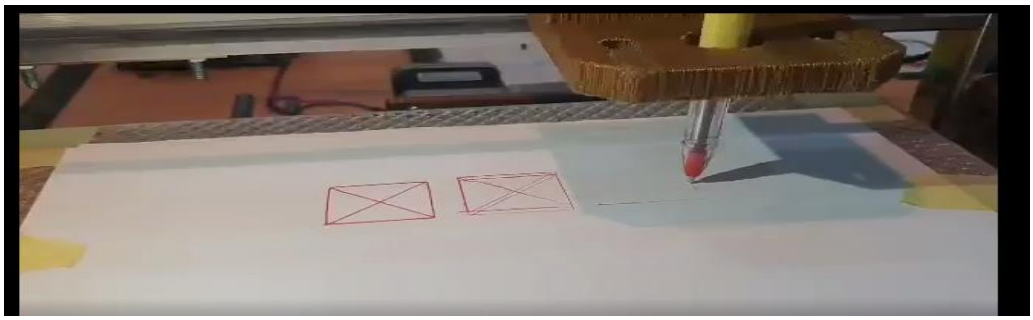


Figure 102 : Test du fonctionnement de la machine.

4.7.1. L'erreur mécanique :

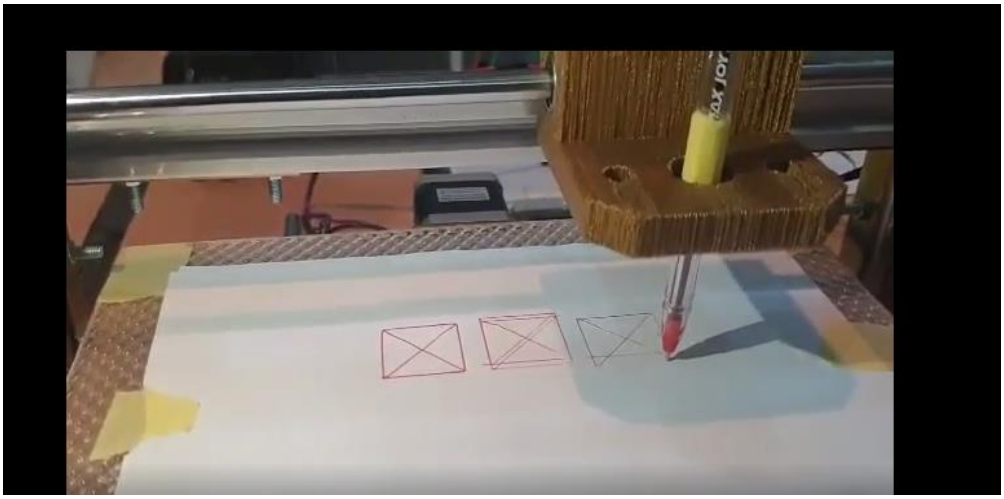


Figure 103 : Fonctionnement de la machine dans sur 3 zones.

On remarque à travers l'image que la précision du premier dessin est beaucoup plus élevée que la précision du deuxième et du troisième dessin, cela est dû à plusieurs raisons telles que :

- l'effet de la direction du dessin à cause des mouvements du charriot qui conduit à un changement de position et cela affecte la stabilité du dessin.
- La table n'est pas parallèle à l'axe OX, à cause de l'inclinaison tout long de l'axe X à cause des erreurs de serrage ou d'équilibre entre les deux côtés.
- Les vibrations jouent également un rôle, elles affectent négativement les mouvements du charriot sur l'axe x et sur la table.
- La feuille sur laquelle la machine écrit peut possiblement glisser ou bouger causant des erreurs.

4.8. Conclusion :

Malgré les difficultés rencontrées, notre objectif de concevoir et réaliser une machine CNC est attient. Nous avons réussi à faire fonctionner la machine on la commandant par nos propres programmes. Des améliorations sont nécessaires pour corriger les erreurs mécaniques que nous considérons comme une perspective d'une recherche ultérieure.

Conclusion générale

Conclusion générale

Notre projet de fin d'études a porté sur la conception et la réalisation d'un modèle de machine à commande numérique (CNC) en utilisant le langage de programmation numérique G-Code et le microcontrôleur MSP430. Cette machine est capable de réaliser un calcul des coordonnées des points définissant une Trajectoire.

En concluant, les acquis de ce projet ainsi que les étapes suivies lors de sa réalisation sont énumérées comme suit :

- La maitrise de la programmation de G-code,
- La maitrise de la programmation des microcontrôleurs de type MSP430G2553 en utilisant le langage C.
- La maitrise de fonctionnement et de la commande des moteurs pas à pas,
- Résoudre quelques contraintes techniques liées à la limitation de mémoire du microcontrôleur.
- La maitrise de la conception mécanique ainsi que l'animation de la machine CNC à base du SolidWorks.
- Fabrication et réalisation de la machine en regroupant déférents aspects (hardware et software).

Il est à noter que nous avons rencontré plusieurs problèmes durant la conception de la commande et la réalisation pratique à savoir : la capacité de stockage de ram du MSP430G2553. Mais grâce à ce que nous avons appris durant nos études académiques et pratiques et l'assistance de nos encadreurs, nous avons pu confronter ou surmonter ces difficultés.

Ce projet a servi de source pour la découverte et l'apprentissage des plusieurs domaines d'études tels que l'informatique, la programmation langage C, la réalisation des circuits électriques, la commande des moteurs avec les microcontrôleurs, la programmation numérique avec G-code et la conception et assemblage de différentes pièces mécanique à base de SOLID WORKS.

Puisque notre projet est démarré à zéro, nous avons une source d'erreurs comme :

- La précision des mesures du modèle,
- L'arrondissement des nombres flottants,
- Les erreurs mécaniques.

Le travail réalisé pourrait être complété en considérant les différents aspects suivants :

- Installation de notre machine CNC dans une unité de production,
- Application des algorithmes d'intelligence artificielle sur notre machine CNC.

Références bibliographiques

Références

[1] Boulsane Mehdi ,Mézâche Mohammed ,Ghiti Mohamed Taher.2016. CNC machine à 2 axes. Mini projet, Université de constantine1,P.3-6.

[2] Chanez Guerrouabi , Roza Ait Rahmane.2018. Etude et conception d'une machine CNC (Découpeuse laser). Mémoire de Master, Université Mouloud Mammeri, Tizi-Ouzou,3p.

[3] Abdennadji Youssef, Ahmadi Anouar.juin 2010, Conception et réalisation d'une commande numérique d'une machine de decouplaser , Ingénieurd'école nationale de Gabés.

[4] D. Gelin, M. Vincent. Mars 1995. Éléments des fabrications, Edition marketing, paris.

[5] Bentaleb mokhtar,gasmi elhadi.2016. Réalisation et commande d'une machine cnc à base des moteurs pas à pas. Mémoire de Master, universite kasdi merbah, ouargla,7p.

[6] Takashi Kenjo . Akira Sugawara, stepping motors and their microprocessor controls Oxford University Press , ISBN 0-19-859385-6.

[7]Patrice,O 2004.Moteurs pas -a -pas et pc.paris :Donod,Paris,7p, 9p, 10p.

[8] Rafik Ben Kad, Djamel Smaini.2015. Conception et réalisation d'une machine CNC.mémoire de Master, universite mouloud mammeri, tizi-ouzou, 28p.

[9] http://www.positron-libre.com/ elctronique/moteur-pas-a-pas/sequence-commande

[10] http://moteur-industrie.com/moteurs-pas-a-pas/technique.html

[11] http://sam.electroastro.pagesperso-orange.fr/dossiers/pasapas/moteurpas2.htm

[12] Delfouf Ouafia, / Fekir Hanane .2015. Etude Et réalisation d'une prothèse auditive numérique.mémoire de Master, universite mouloud mammeri ,tizi-ouzou,65p

[13] https://fr.wikipedia.org/wiki/C_(langage).

[14] http://ftp.iar.se/WWWfiles/msp430/webic/doc/EW430_CompilerReference.pdf

[15] http://pyrochat.github.io/mooc-led/cours/501/inter.html

[16] :https://fr.wikipedia.org/wiki/Modulation_de_largeur_d%27impulsion

[17] https://en.wikipedia.org/wiki/Duty_cycle

[18] http://learncontrollers.blogspot.com/2018/02/pulse-width-modulation.html

[19] KARA ALI REDHA Etude, Conception et Réalisation d'une Imprimante 3D à dépôt de la matière

[20] http://www.simplyembedded.org/tutorials/msp430-uart/

[21]https://www.xanthium.in/Serial-Communication-MSP430-UART-USCI_

[22] http://www.3dprintingdatabase.org/sites/default/files/20137114837820.jpg

[23] http://mchobby.be/wiki/index.php?title=Fichier:A4988-PinOut.jpg

[24] « https://www.pololu.com/product/2128».

[25] Sources et contributeurs de l'article, Programmation de commande source :

http://fr.wikipedia.org/w/index.php?oldid=73873596contributeurs:Anthere,Aristote2,Arnaud.Serander,Badmood,Bapti,Betbert

[26] Le Brear CNC, https://lebearcnc.com/,25 06 2018

[27] Kessai Sonia,Saich saida.2018 « élaboration d'un simulateur pour une machine-outil a command numérique ». mémoire de Master, Université Mouloud Mammeri ,Béjaia,30p,31p.

[28] http://tu-quincy .fr /codes-de-programtion-iso-machine-num/,04/04/2018,11 :38

[29] http://vernier.frederic.free.fr/Teaching/IGSD/Cours_01_algorithmique_graphique.pdf

[30] https://elbarnas.wordpress.com/category/les-principes-de-fonctionnement-dune-machine-cnc/

[30]https://elearning.gov.mr/storage/app/public/html/DessinTransformatic_web_gen_auroraW/co/module_Dessin_Transformatic_1.html.

[32] BOUYAHIA YESSINE . 2019. Réalisation d'un prototype d'une machine CNC 3 axes. Mémoire de Master . Université Aboubakr Belkaïd . Tlemcen ,27p

[33] MOUZAOUI Melissa , TAZAMOUCHT Yanis. Réalisation et automatisation d'une machine à commande numérique . Mémoire de Master. .Université A.MIRA-BEJAIA ,42p

[34] MESBAH Farouk,,ADDOUN Abdellah .Session Juin 2019 . .Mémoire de Master Plateforme IIoT pour le suivi et la commande numérique des machines .Université Constantine 2- Abdelhamid Mehri . Constantine ,110p, 111p

[35] KHELIFI Fateh . 2018. Conception et réalisation d'une machine à commande numérique fraiseuse . Mémoire de master. Université SAAD DAHLAB de BLIDA

[36] Patrick BLAIN, technique de l'ingénieur (CAO et méthodologie de conception). Document B 2 810

[37] https://fr.wikipedia.org/wiki/SolidWorks

[38] solidworks.fr Profil de la société Purdue. Université Purdue Research and Education Centre for Information Systems en génie 1997.

Annexes

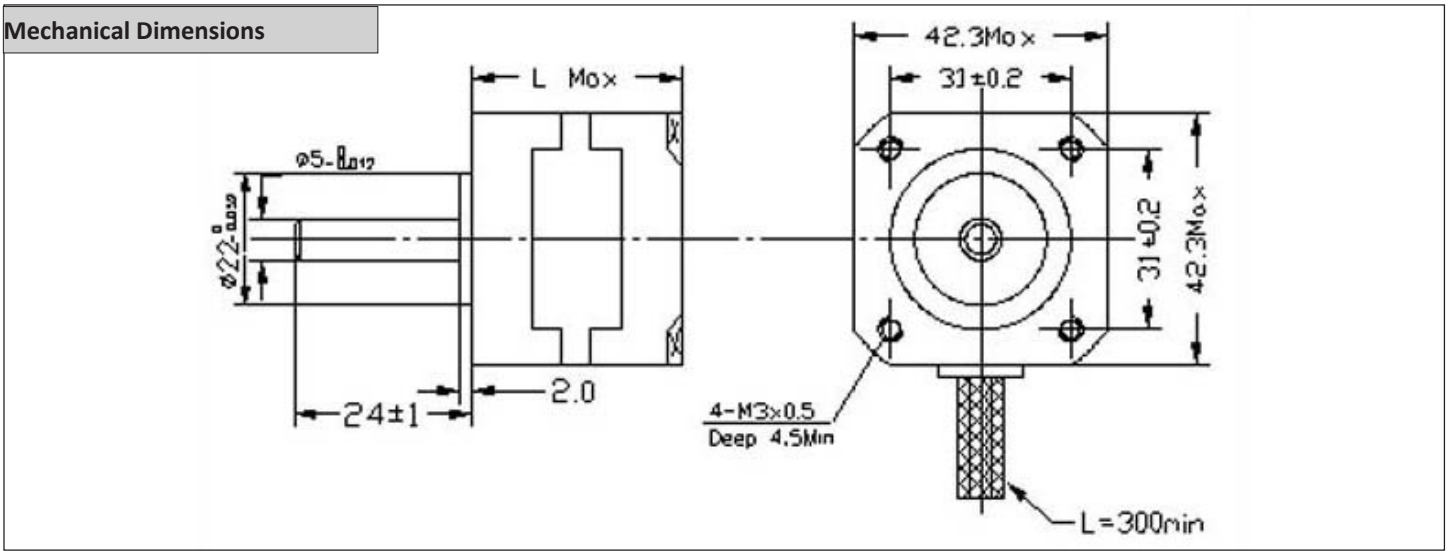
Annexe 1 : Moteur pas à pas Nema 17 42BYGHW811

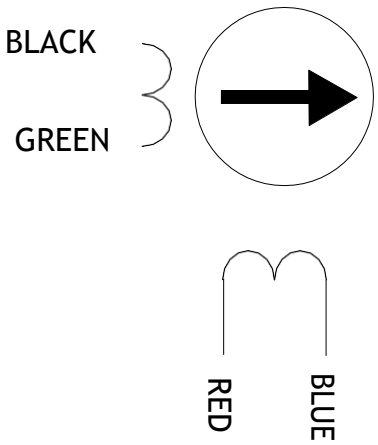
General Specifications	
Item	Specification
Step Accuracy	5%
Temperature Rise	80°C Max
Ambient Temperature	-20°C~+50°C
Insulation Resistance	100MΩ Min., 500VDC
Dielectric Strength	500 VAC for one minute

Electrical Specifications

Model No.	Step Angle	Motor Length	Rated Voltage	Rated Current	Phase Resistance	Phase Inductance	Holding Torque	# of Leads	Rotor Inertia	Detent Torque	Weight
	°	mm	V	A	Ω	mH	g-cm		$g\text{-}cm^2$	g-cm	kg
42BYGHW811	1.8	48	3.1	2.5	1.25	1.8	4800	4	68	280	0.34

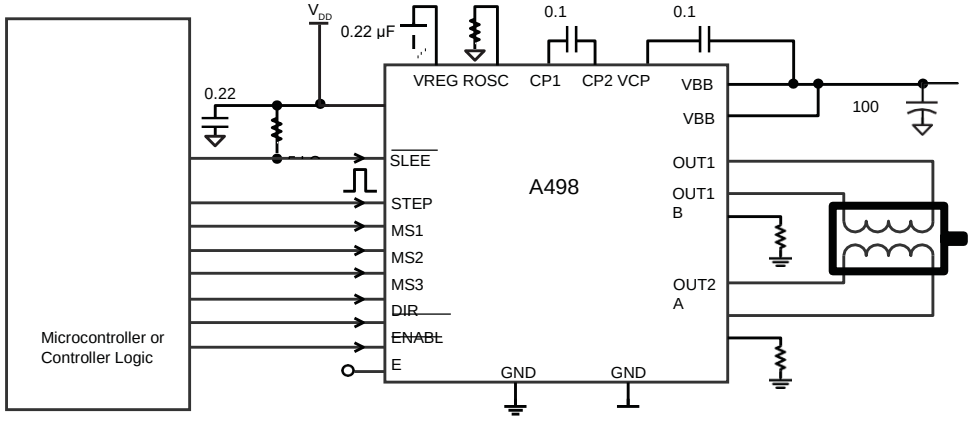
Mechanical Dimensions



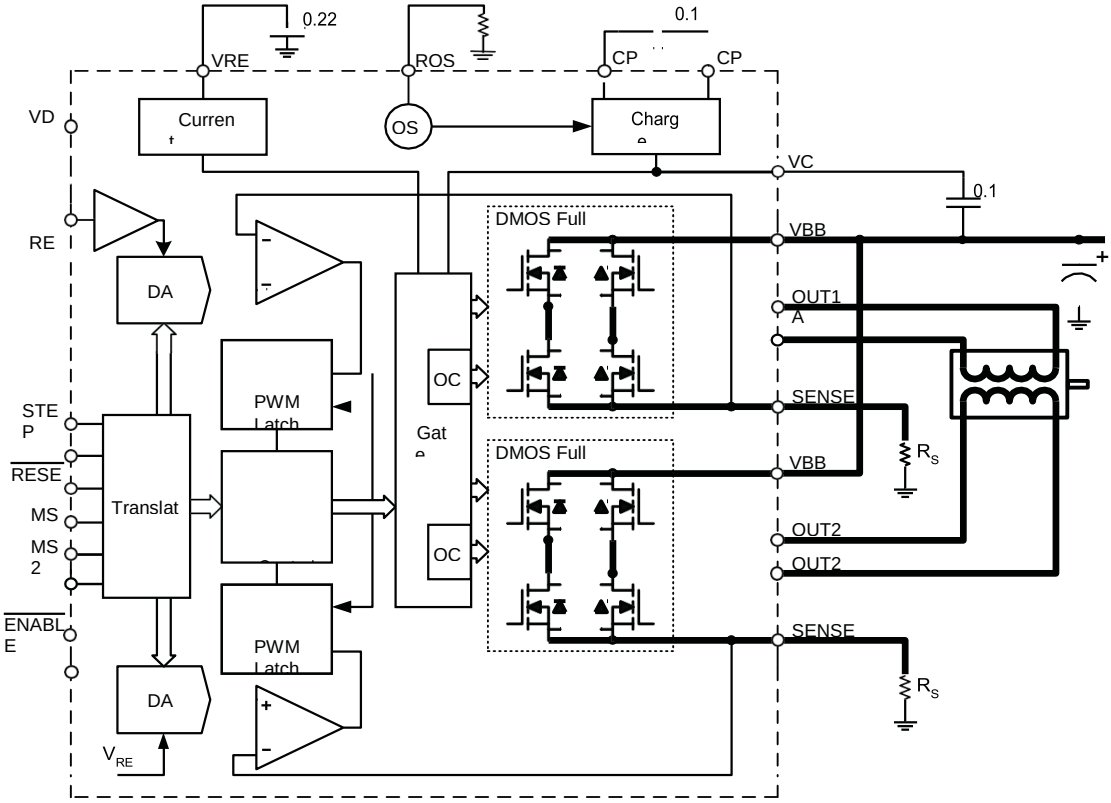


Annexe 2: DRIVER A4988

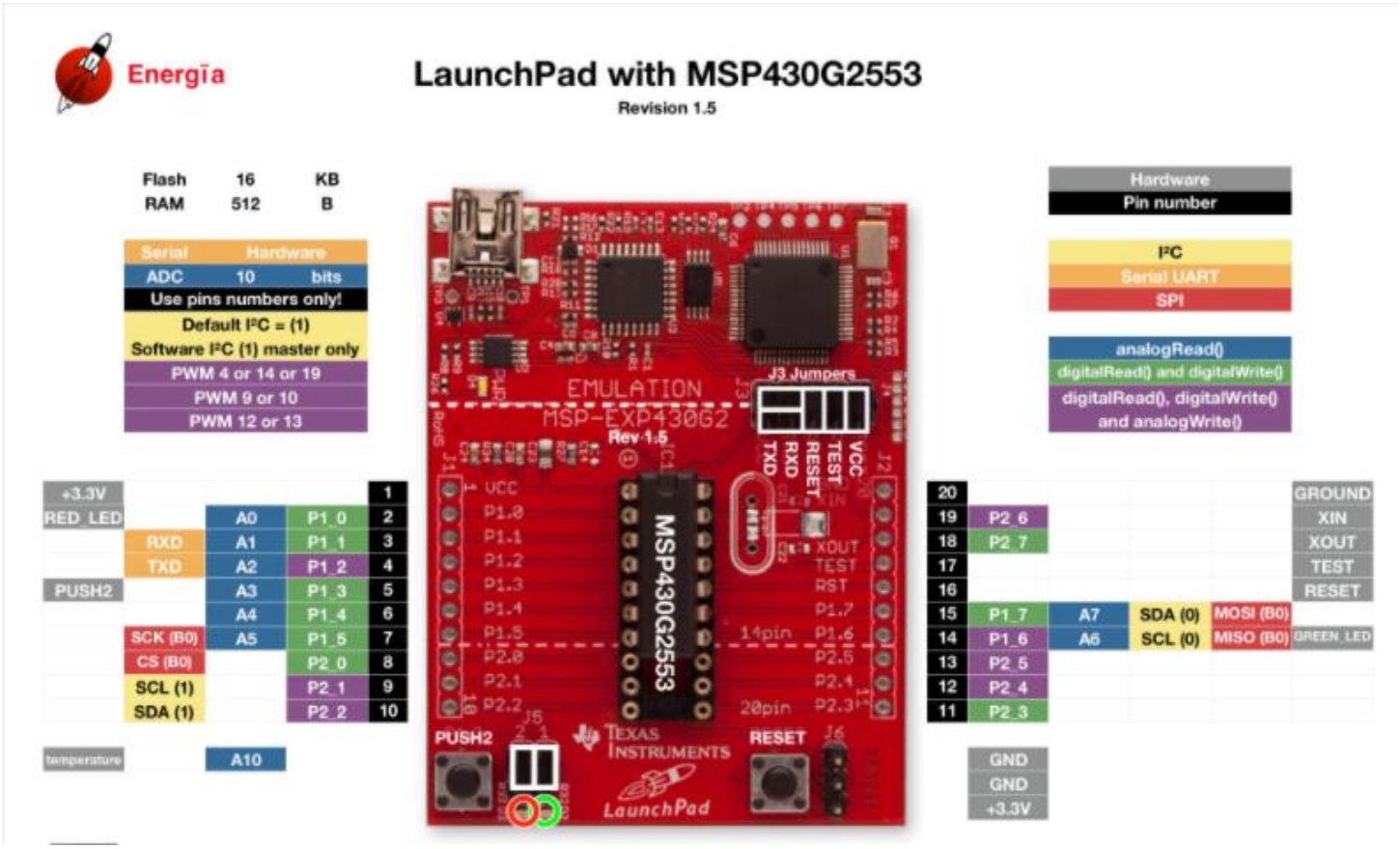
Typical Application Diagram



Functional Block Diagram



Annexe 3: MSP430G2553



Annex 4: Tableau G-code

Common G Codes (avec certaines extensions non standardisées)	
G00	Déplacement rapide
G01	Interpolation linéaire
G02	Interpolation circulaire (sens horaire, anti-trigo)
G03	Interpolation circulaire (sens antihoraire, trigo)
G04	Arrêt programme et ouverture carter (pour nettoyer) (temporisation - suivi de l'argument F ou X en secondes)
G10/G11	Écriture de données / Effacement de données (suivi de l'argument L suivant le type de données à écrire)

G17	Sélection du plan X-Y
G18	Sélection du plan X-Z
G19	Sélection du plan Y-Z
G20	Programmation en pouces
G21	Programmation en mm
G28	Retour à la position d'origine
G31	Saute la fonction (mode *Interrupt* utilisé pour les capteurs et les mesures pièces et de longueur d'outil)
G33	Filetage à pas constant
G34	Filetage à pas variable
G40	Pas de compensation de rayon d'outil
G41	Compensation de rayon d'outil à gauche
G42	Compensation de rayon d'outil à droite
G54 à G59	Activation du décalage d'origine pièce (*Offset*)
G68 / G68.1	Activation du mode "Plan incliné" (*Tilted plane working*) pour les centres d'usinage 5 axes
G70	Cycle de finition
G71 / G71.7	Cycle d'ébauche suivant l'axe Z (appel de profil balisé entre les arguments P et Q)
G75	Cycle de gorge
G76 / G76.7	Cycle de filetage
G83	Cycle de perçage débourrage
G69	Annulation du mode *Tilted plane working* (Plan incliné)

G84	Cycle de taraudage rigide
G90	Déplacements en coordonnées absolues
G91	Déplacements en coordonnées relatives
G94/G95	Déplacement en pouces par minute/pouce par tour
G96 ; G97	Vitesse de coupe constante (vitesse de surface constante) ; Vitesse de rotation constante ou annulation de G96

Annexe 5 : Forme finale de notre machine CNC