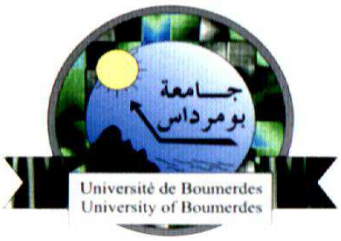
N° Ordre……../DGM/FSI/UMBB/2018

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE M'HAMED BOUGARA BOUMERDES



Faculté des Sciences de L'Ingénieur
Département Génie Mécanique

Mémoire de Master

En vue de l'obtention du diplôme de **MASTER** en **:**

Filière : Génie Mécanique
Option : Mécatronique

THEME
Etude et réalisation d'un bras manipulateur Commandé par le microcontrôleur 16F877A

Présenté par :

Kherchi Mohamed

Rial Sofiane

Promoteur : Mr.Benazzouz Djamel

Promotion 2017- 2018

REMERCIEMENTS

❖ *Avant tout, on remercie Dieu pour tout le courage et la force qu'il m'a donné pour éditer ce travail.*

❖ *On remercié* ***Mr Benazzouz Djamel*** *celui qui a réalisent notre idée en PFE, pour nous encourager et ouvert son laboratoire pour nous conseiller et nous guider.*

❖ *Nous remercions Mr* ***Kherchi Hamid*** *et* ***Cash Process*** *qui nous a ouvert son atelier pour ce projet.*

❖ *Tous nos professeurs de master, de la formation mécatronique pour leurs disponibilité et conseils.*

❖ *Nos pères et mères qui nous ont soutenue durant ce travail, nos frères et sœurs nous ont toujours entouré et motivé à sans cesse devenir meilleur.*

❖*Nombreuses sont les personnes qui nous ont soutenu, nos famille, nos amies, on ne peut tous les citer, qu'ils trouvent ici l'expression de notre profonde reconnaissance.*

Table des matières

Introduction Générale

INTRODUCTION GENERALE

L'être humain a besoin de plusieurs outils pour faciliter la vie, des moyens sophistiqués, des moyens touchés par la technologie, un de ces moyens est la robotique.

Le domaine qui nous permet d'effectuer des tâches et de manipuler des objets selon un programme de façon automatique. Ce domaine est généralement utilisé pour remplacer les humains dans des situations ou ces derniers sont incapables d'effectuer le travail, des situations plus dangereuses, de haute précision ou répétitive.

Etant donné que ce domaine est très important et intéressant, Il est devenu le devoir de chaque chercheur ou étudiant, dans le domaine de sciences technique, de comprendre le fonctionnement de cette technologie, d'où notre intérêt. Nous avons donc, décidé de réaliser un bras robot.

Ce dernier est à trois degrés de liberté. Ce bras manipulateur est un système de positionnement. Ces positions sont produites par des actionneurs, pour la réalisation de ce bras nous nous intéressons aux**Moteur pas à pas.**

L'objectif principal de notre travail est de réaliser et commander un bras manipulateur à 3 degrés de liberté commandé par le microcontrôleur 16F877A. Ce Microcontrôleur a pour fonction de traiter les informations entrantes pour émettre des ordres de sorties en fonction d'un programme, Ce bras est manipulé par ordinateur avec une interface graphique.

Description du mémoire

Ce mémoire est structuré en une introduction générale ainsi que cinq chapitres et des références bibliographiques.

Le chapitre 1 : Généralités sur les robots industriels qu'on pourrait appliquer lors de la conception et de l'utilisation de robots industriels ont été résumées

Le chapitre 2 : Uneétudesur les actionneurs et capteurs en robotique a été développée

Le Chapitre 3 ; Donne une bref présentation sur le système commande microcontrôleur

Le Chapitre 4 : Présente tous les outils de conception utilisés dans le projet.

Le chapitre 5 : Ce dernier chapitre présente la conception et réalisation d'un bras manipulateur commandé par des microcontrôleurs de type (16f877a)

Enfin, nous terminons par une conclusion générale qui illustre les principaux résultats obtenus sur le modèle de conception développé dans ce mémoire

Chapitre 1 :

Robot industriel

1 ROBOTS INDUSTRIELS

1.1. Introduction

L'utilisation des systèmes robotiques apparaît aujourd'hui dans plusieurs domaines d'activités : la médecine, la défense, la recherche scientifique etc.… Les robots sont utilisés de manière privilégiée pour des missions où les objectifs sont quantifiables et clairement définis. Ils sont destinés à faciliter les tâches pour l'homme et à amplifier le rendement.

Un aperçu sur les différents types de robots et un bref historique sur l'évolution de la robotique industrielle est à présenter dans ce chapitre, ainsi que les éléments constitutifs de ces derniers.

1.2. Histoire des robots industriels

Définitions :

La robotique : La robotique est la science qui s'intéresse aux robots. En fait, il s'agit d'un domaine multidisciplinaire : on y trouve des aspects concernant la mécanique, l'informatique, l'électronique, … Le terme robot est apparu pour la première fois vers 1920 dans une pièce de théâtre du tchèque K. Schappe où il désignait de petits êtres artificiels anthropomorphes répondant parfaitement aux ordres de leur maître ("robota" signifie travail en tchèque)

Robot industriel : Un robot est un dispositif mécatronique (alliant mécanique, électronique et informatique) accomplissant automatiquement des tâches diverses. C'est une machine intelligente fonctionnelle qui nécessite une autonomie de mouvements, qui est destiné à effectuer une grande variété de tâches**.**

La robotique industrielle a connu un essor entre 1950-1970. Elle a vu le jour en 1954lorsque Georges DEVOL a pu réaliser son brevet sur la robotique. Dans ce brevet, Georges DEVOLaconçu un robot qu'il a intitulé Unimate.En 1961, le premier Unimate fut utilisé dans les usinesdeGeneral Motors.

En 1966, l'entreprise Unimation continue de développer des robots et élaborent notammentdes robots permettant de faire d'autres tâches, comme des robots de manipulation matérielleou encore des robots conçus pour la soudure ou pour d'autres applications de ce genre.

En 1978 un nouveau robot est conçu par UnimationInc avec l'aide de General Motors. Ensemble ils conçurent le robot PUMA 500. Le robot PUMA (Programmable UniversalMachine for Assembly) a été conçu par Vic Schienman et fut financé par General Motors etpar The Massachussetts Institute of Technology au milieu des années 70.

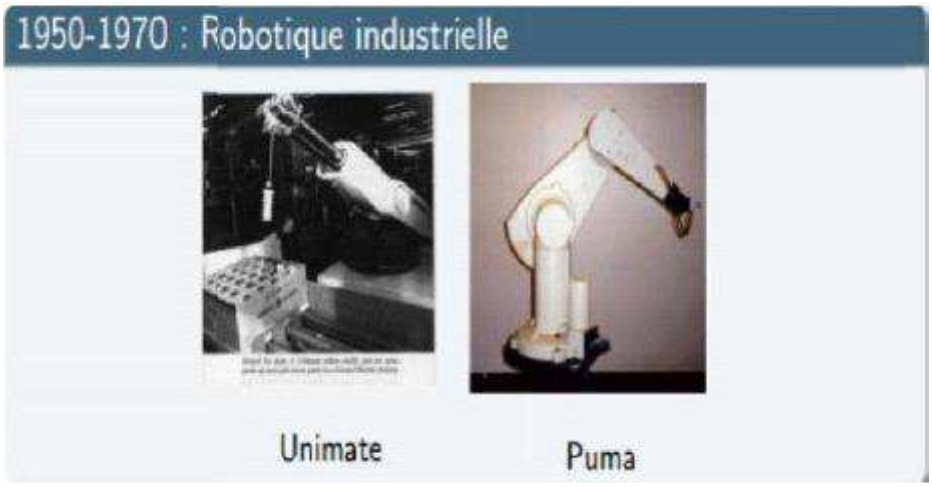


Figure 1.1:les robots Puma et Unimate

En 1985, Reymond Clavel a imaginé le Robot Delta qui possède un bras de manipulationformé de 3 parallélogrammes. Son brevet tombe dans le domaine public en 2007 et différentsconstructeurs devraient alors sortir leur propre robot Delta. (**JPL**) développe un robot industriel hexapode (à 6 pattes) du nom de Lemur. Lemuraurapour mission de monter, assembler et réparer des installations spatiales. Pesant moins de 5 kg, il offre la possibilité innovante d'adapter différents outils sur chacun de ses membres.

Figure1.2: robotlemur

Avec l'apparition de la robotique industrielle, les robots étaient conçus pour remplacerles ouvriers dans les tâches pénibles, répétitives ou dangereuses (peinture, soudure…).Aujourd'hui avec le développement de l'électronique, de l'informatique, de la mécanique et aussi de l'automatique, la technologie robotique a progressé. La recherche dans le domaine de la robotique est dirigée vers le développement de robots dévoués à des tâches bien différentes que celles demandées par l'industrie. Par exemple des robots travaillant en

mode automatique ou semi-automatique et qui ont souvent pour objectif d'interagir avec des humains et de les aider dans leurs tâches (surveillance, manutention d'objets lourds…). Ils sont dotés d'une intelligence qui leur donne une certaine autonomie. Ainsi donc, le développement important de l'intelligence artificielle et de la robotique font que de nouveaux robots apparaissent constamment et l'utilisation de systèmes robotiques apparait aujourd'hui dans plusieurs domaines d'activité : la médecine, la défense, la recherche etc.

1.3. Types de robot industriel

Il y a deux types de robots : robots mobiles et robots manipulateurs.

1. Robots mobile

Un robot mobile est un système mécanique, électronique et informatique agissant physiquement sur son environnement en vue d'atteindre un objectif qui lui a été assigné. Cette machine est polyvalente et capable de s'adapter à certaines variations de ses conditions de fonctionnement. Elle est dotée de fonctions de perception, de décision et d'action. Ainsi, le robot devrait être capable d'effectuer des tâches diverses, de plusieurs manières, et accomplir correctement sa tâche, même s'il rencontre de nouvelles situations inattendues, il doit être :

- capable de choisir ses actions pour atteindre son but.
- capable d'accomplir correctement sa tâche même s'il rencontre de nouvelles situations inattendues sans intervention humaine.

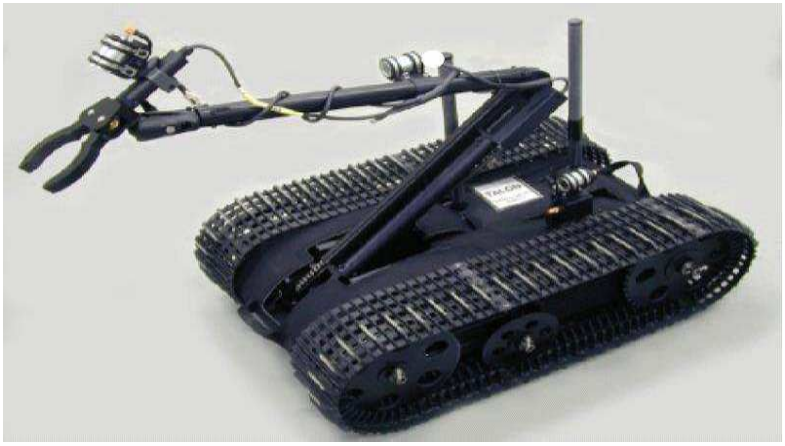


Figure 1.3: Robot Mobile

2. Robot manipulateur

Un robot est une machine capable d'effectuer des tâches et de manipuler des objets selon un programme de façon automatique. Ils sont généralement utilisés pour remplacer les humains dans des situations ou ces derniers sont incapables d'effectuer le travail, des situations plus dangereuses, de haute précision ou répétitives. Les robots industriels se distinguent souvent par un système articulé semblable à un bras humain. Ce bras peut se déplacer, dans certains cas, sur plusieurs axes et est muni d'un outil propre à l'opération à effectuer : l'organe effecteur. Les applications typiques de ces robots en industrie sont le soudage, la peinture, la manipulation, l'assemblage et l'inspection. Les robots sont de plus en plus utilisés en industrie, car ils sont très rapides, précis et ne se fatiguent jamais.

Figure 1.4 : robot manipulateur

1.4. Composition d'un robot

Un robot est un assemblage complexe de pièces mécaniques et de pièces électroniques, le tout pouvant être piloté par une intelligence artificielle. Disposées comme suit:

1. Partie opérationnelle

L'unité opérationnelle contient :

Source d'énergie : Ils possèdent également une source d'énergie embarquée : généralement une batterie d'accumulateurs électriques

Capteurs : Un capteur est un dispositif transformant l'état d'une grandeur physique observée en une grandeur utilisable, telle qu'une tension électrique, Cette information est utilisée pour calculer l'ordre approprié à envoyer aux actionneurs.

Actionneurs : Les actions des robots sont réalisées à l'aide d'actionneurs. Ce sont des organes qui transforment l'énergie qui leur est fournie en un phénomène physique utilisable comme des mouvements.

2. Partie informationnelle

Les systèmes de traitement de l'information, généralement sont des microprocesseurs à basse consommation d'énergie ou des microcontrôleurs.

3. L'interface de communication

Ils possèdent également un transformateur d'information : envoyer et recevoir des informations d'après les interfaces entrée / sortie (liaison filaire, Sans fil, Ecran tactile...)

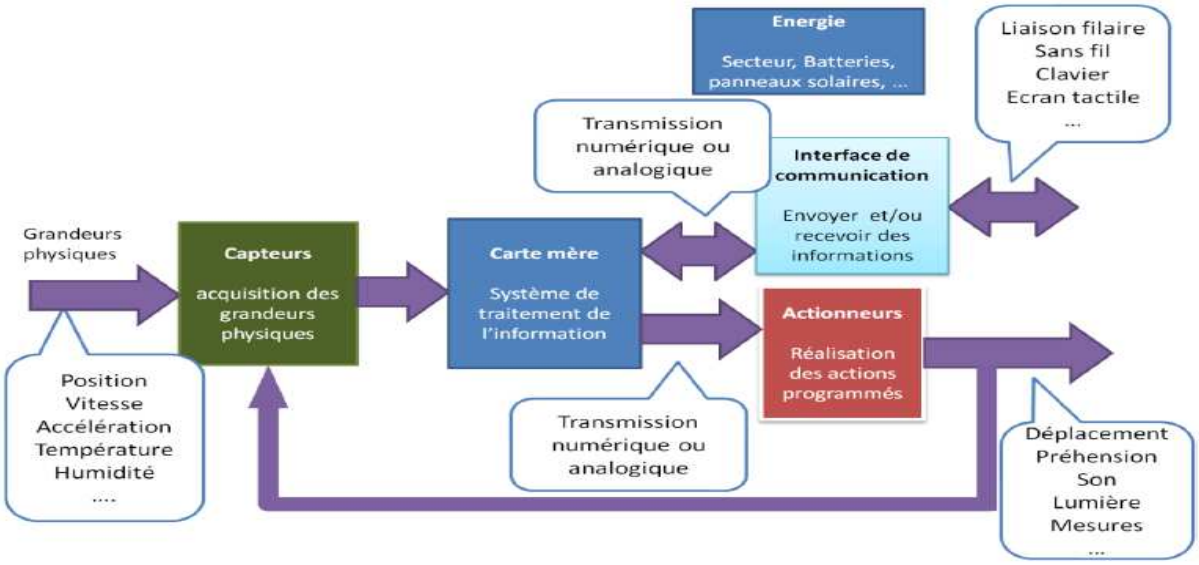


Figure 1.5 : Unité opérationnelle et fonctionnelle d'un robot

1.5. Architecture de manipulateur :

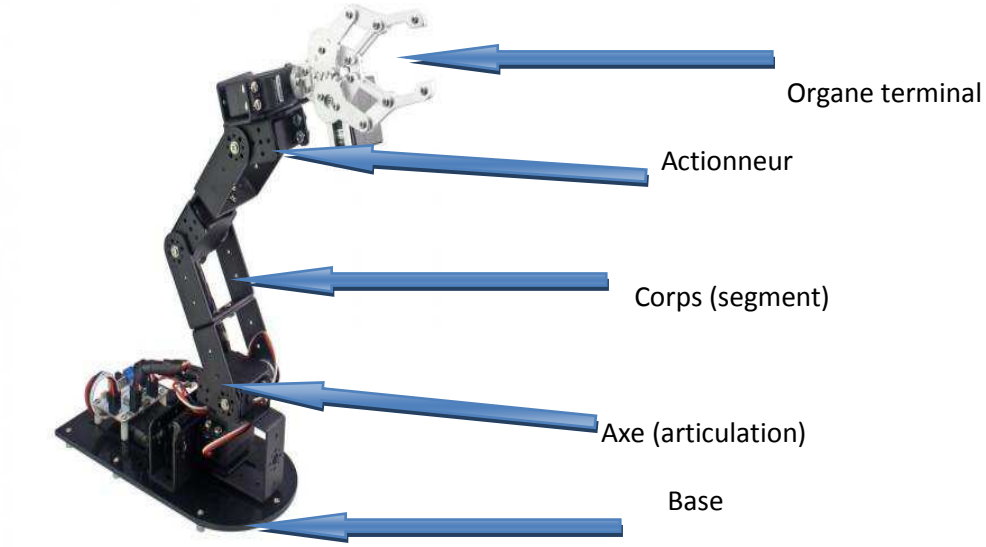


Figure 1.6 : Architecture des robots

Base : La base du manipulateur est fixée sur le lieu du travail. Ceci est le cas de la majorité des robots industriels.

Actionneur : Les actionneurs sont les «muscles» de manipulateurs qui transforment l'énergie qui lui est fournie en un phénomène physique qui fournit un travail. Les types

communs des actionneurs sont les servomoteurs, les moteurs pas à pas, les actionneurs pneumatiques et les vérins hydrauliques. Les actionneurs sont sous le contrôle du contrôleur.

Porteur : Le porteur représente l'essentiel du système mécanique articulé (segment, articulation, actionneur, l'organe terminal), il a pour rôle d'amener l'organe terminal dans une situation donnée imposée par la tâche. Il est constitué de :

Segment : Corps solides rigides susceptibles d'être en mouvement par rapport à la base du porteur, et les uns par rapport aux autres.

Articulations : Une articulation lie deux corps successifs en limitant le nombre de degré de liberté, de l'un par rapport à l'autre.

a) ***Articulation rotoïde*:**Il s'agit d'une articulation de type pivot, notée R, réduisant le mouvement entre deux corps à une rotation autour d'un axe commun. La situation relative entre les deux corps est donnée par l'angle autour de cet axe (Voir la figure ci-dessus)

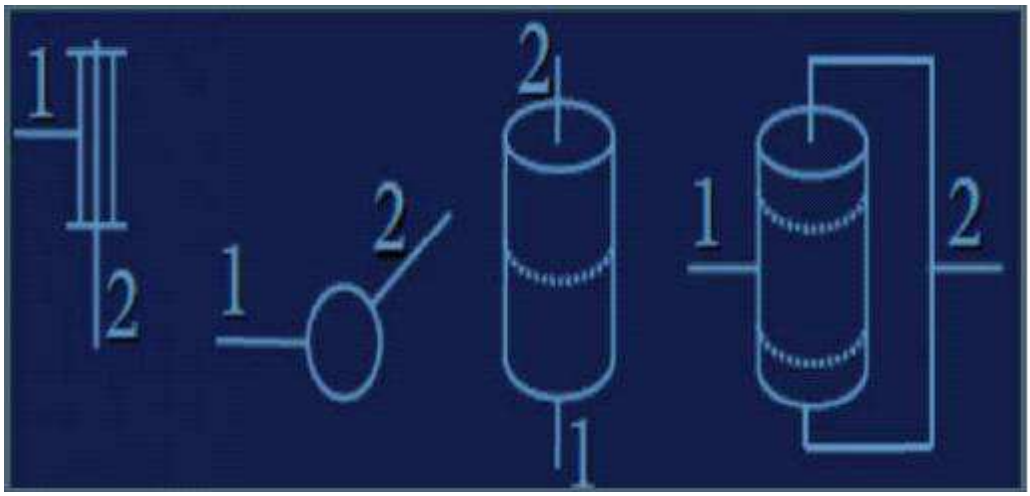


Figure 1.7 : ArticulationRotoïde

b) **Articulation prismatique :** Il s'agit d'une articulation de type glissière, notée P, réduisant le mouvement entre deux corps à une translation le long d'un axe commun. La situation relative entre les deux corps est mesurée par la distance le long de cet axe (voir la figure si dessus).

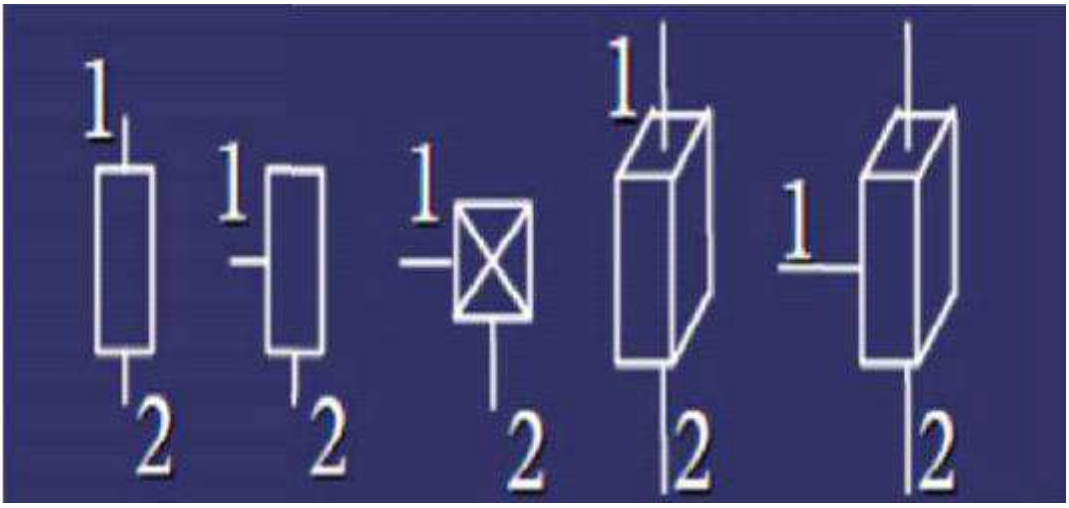


Figure 1.8: Articulation prismatique

L'organe terminal : On regroupe tout dispositif destiné à manipuler des objets (dispositifs de serrage, Dispositifs magnétiques, à dépression, …), ou à les transformer (outils, torche de soudage, Pistolet de peinture, …). En d'autres termes, il s'agit d'une interface permettant au robotd'interagir avec son environnement. Un organe terminal peut être multifonctionnel, au sens oùil peut être équipé de plusieurs dispositifs ayant des fonctionnalités différentes. Il peut aussi être monofonctionnel, mais interchangeable. Un robot, enfin, peut-être multi-bras, chacun des bras portant un organe terminal différent. On utilisera indifféremment le terme organe terminal, préhenseur, outil ou effecteur pour nommer le dispositif d'interaction fixé àl'extrémité mobile de la structure mécanique, exemple : pistolet pour la soudure dans les robots industriels.

1.6. Classification des robots manipulateurs

Le nombre de classe et les distinctions entre celles-ci varient de pays à pays (6 classes Japon, 4 en France).

1. Classification fonctionnelle

L'A.F.R.I. distingue 4 classes illustrées ci-dessous :

- **Manipulateur à commande manuelle**

La figure ci-dessous représente les manipulateurs à commande manuelle :

Figure 1.9: manipulateurà commande manuelle

- **Manipulateur automatique**

La figure montre un bras manipulateur qui exerce des mouvements de soudure sansl'intervention de l'homme.

Figure 1.10:manipulateur automatique

- **Manipulateur programmables**

Ils répètent les mouvements qu'on leur a appris ou programmés sans informations sur l'environnement ou la tâche effectué.

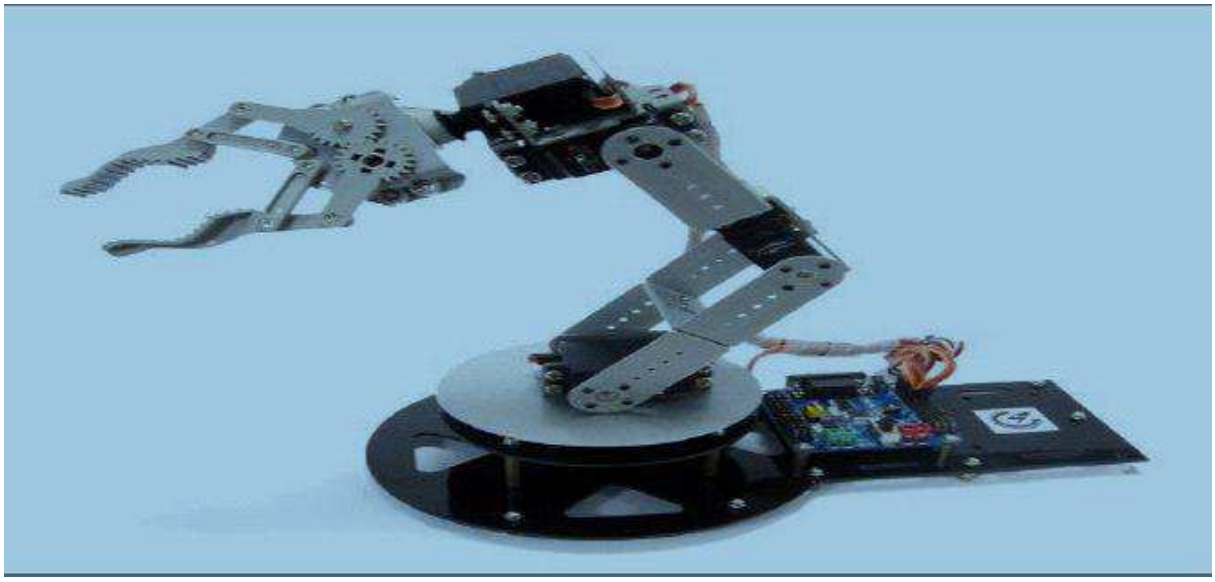


Figure1.11: Manipulateur programmable

- **Robots intelligents**

On trouve actuellement des robots de seconde génération qui sont capables d'acquérir etd'utiliser certaines informations sur leur environnement (systèmes de vision, détecteurs deproximité, capteurs d'efforts,...) comme le montre la Figure I.12.Les robots de troisièmegénération sont capables de comprendre un langage oral proche du langage naturel et de sedébrouiller de façon autonome dans un environnement complexe grâce à l'utilisation del'intelligence artificielle

Figure 1.12: robot intelligent

2. Classification géométrique

On peut aussi classer les robots suivant leur configuration géométrique

- **Structure cartésienne (PPP)**

C'est une structure à trois liaisons prismatiques et est la plus ancienne. Historiquement, elle découle logiquement de la conception traditionnelle d'une machine-outil à trois axes, typerectifieuse ou fraiseuse par exemple. Cette structure est relativement peu utilisée sauf dans quelques applications particulières telles que robots pratiques, robots de magasinage.

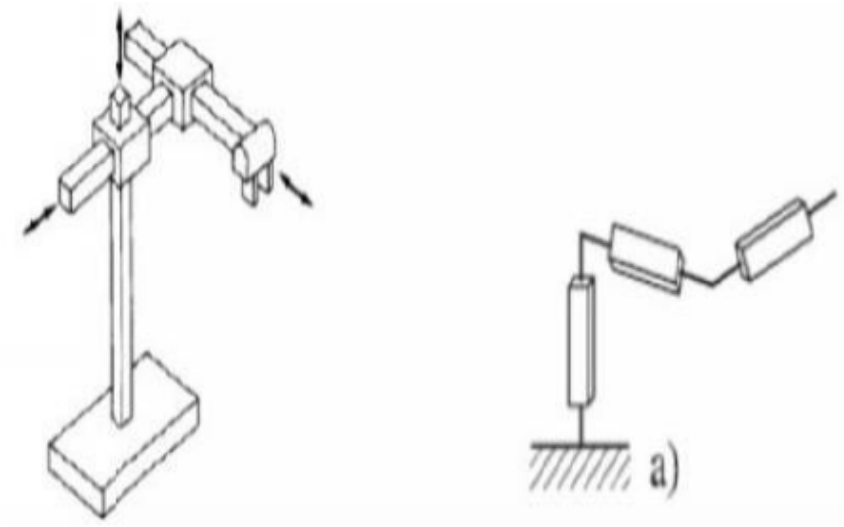


Figure 1.13: Structure cartésienne (PPP)

- **Structure cylindrique (RPP) ou (PRP)**

Cette structure associe une rotation et deux translations. Elle présente l'inconvénientd'offrir un volume de travail faible devant un encombrement total important. Elle n'estpratiquement plus utilisée.

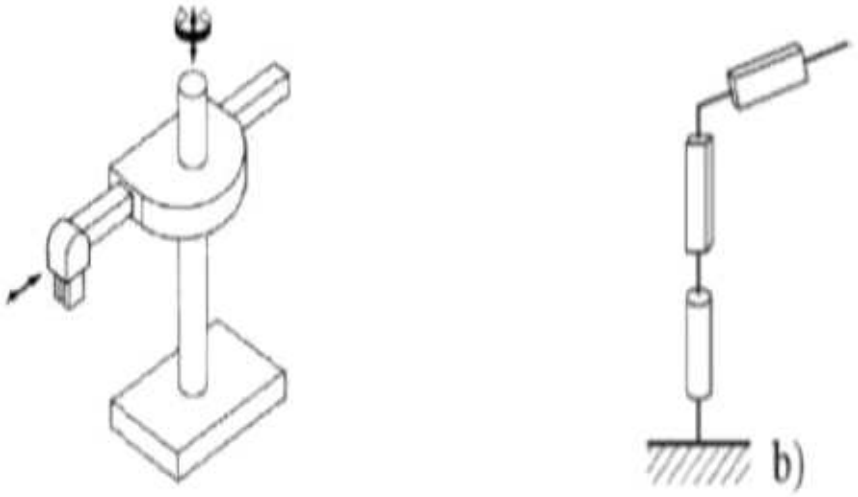


Figure 1.14 : Structure cylindrique

- **Structure sphérique ou polar**

C'est une structure quasiment abandonnée pour des raisons similaires à l'abandon de la structure cylindrique

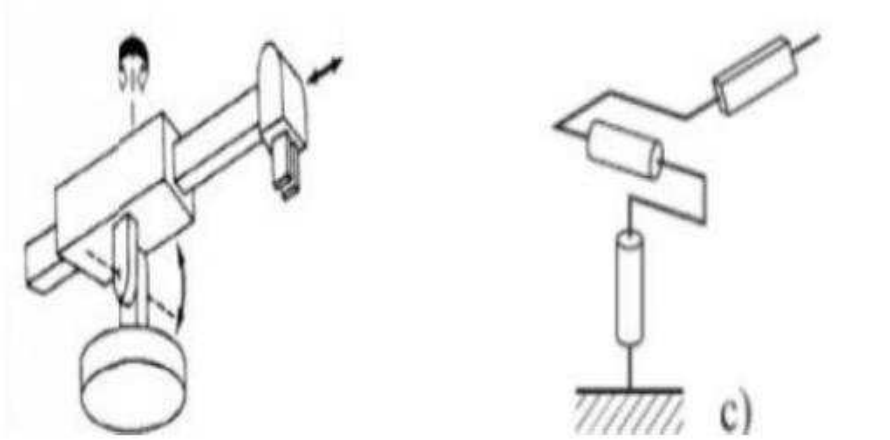


Figure 1.15: Structure polar

- **Structure dite Scara**

A axes de rotation parallèles, elle est l'une des plus utilisées en particulier pour des tâches de manutention ou d'assemblages très fréquents dans l'industrie. Ce succès commercial est lié au fait que le ratio entre le volume de travail et l'encombrement est trèsfavorable et aussi au fait que la structure SCARA est très adaptée à ce type de tâches.

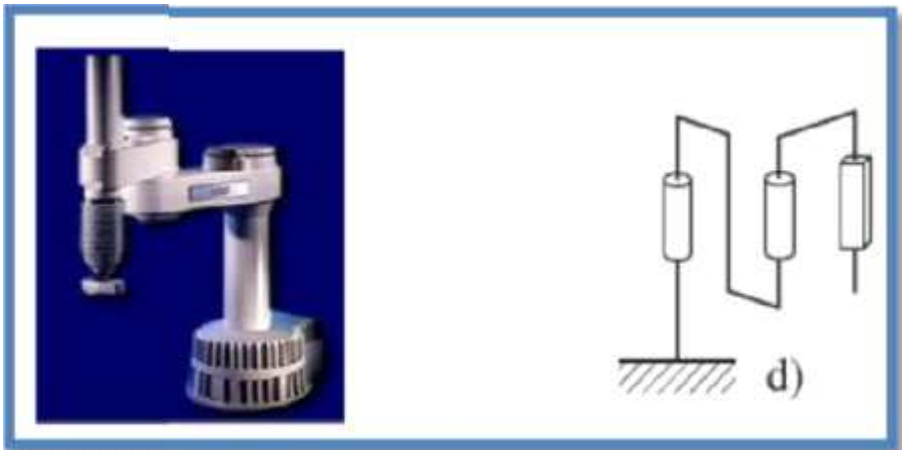


Figure 1.16: Structures Scara

- **Structure 3R (anthropomorphe)**

Elle permet d'amener un solide en un point de l'espace par trois rotations, généralement une à axe vertical et deux à axes horizontaux et parallèles. C'est le porteur «généraliste par excellence pouvant se programmer facilement pour différents types de tâches et disposant d'un volume de travail conséquent.

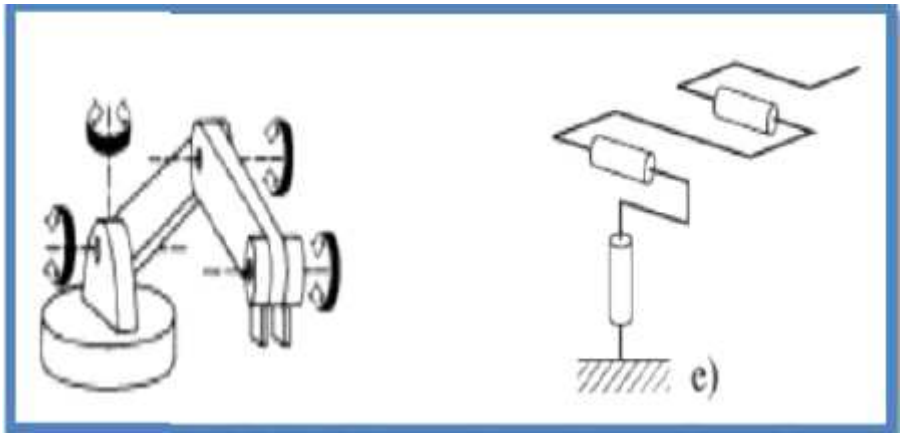


Figure I.17: structureAnthropomorphe(3R)

1.7. Conclusion

Dans ce chapitre, nous avons donné une idée générale sur la robotique, l'historique des robots, leurs structures, leurs utilisations et les différents types de robots ainsi que leurs classifications et leurs domaines d'applications.

Les robots industriels peuvent être très utiles dans plusieurs domaines. Ils sont rapides, très fiables et très précis. On les retrouve dans des domaines comme le soudage, l'emballage, la peinture, le transport, l'inspection …. Etc.

Avantages et inconvénients d'un bras manipulateur

- **Avantages**
- Augmente la productivité
- Utilise efficacement les équipements
- Réduire les coûts de travail
- Flexibilité au travail
- Obtenir le produit final dans un délai minimum
- Meilleure précision dans l'exécution
- **Inconvénients**
- Provoque du chômage (remplace l'être humain)
- Coût d'achat élevé
- Difficulté a programmée pour exécuter des tâches précises
- Besoin de haute précision et d'un nombre important de capteurs pour effectuer les tâches complexes
- La défaillance de bras robotique induit l'arrêt de la chaîne de production

Chapitre 2 :

Instrumentation

1 INSTRUMENTATION

2.1. Introduction

Les performances des robots dépendent fortement de celles des actionneurs, des chaînes cinématiques associées, et de système sensoriel associe, qui ont une importance primordiale. Les capteurs constituent la source des données qui permettent l'élaboration des commandes pilotant le robot, pour exercer les actions matérielles désirées.

Définition:

Actionneur : est un dispositif qui transforme l'énergie délivrée par l'interface de puissance, en énergie utilisable par les effecteurs de processus. Le schéma fonctionnel d'un actionneur ainsi que les schémas des principaux actionneurs sont représentés à la figure ci-dessous. Les moteurs électriques, les vérins pneumatiques ainsi que les éléments chauffants sont des exemples typiques d'actionneurs utilisés en automatisation industrielle

Effecteurs:un effecteur est un dispositif qui transforme l'énergie délivrée par un actionneur, en valeur ajoutée. Selon le cas, un effecteur peut être séparé ou non del'actionneur. Quelques exemples d'effecteurs rencontrés en milieu industriel sont : lesventilateurs, les broyeurs, les pinces à outils de robots articulés, dispositifs de transfert de chaleur...

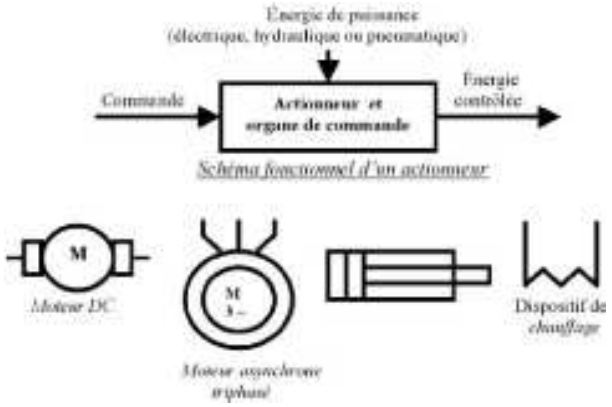


Figure 2.1 : Schéma fonctionnel et symboles des actionneurs

Famille d'actionneur : Le mouvement est imposé au robot par un ou plusieurs actionneurs par transformation d'une énergie source en énergie de base utilisée peut être.

a) Pneumatique
b) Électrique
c) Hydraulique
d) Mécanique (dispositif à câbles et poulies)

2.2. Moteurs pas à pas :

Un moteur pas à pas transforme une impulsion électrique en une énergie mécanique permettant le déplacement angulaire du rotor appelé " pas ".

On doit alimenter leurs bobines dans un ordre bien précis, faisant appel à une logique de commande déterminée.

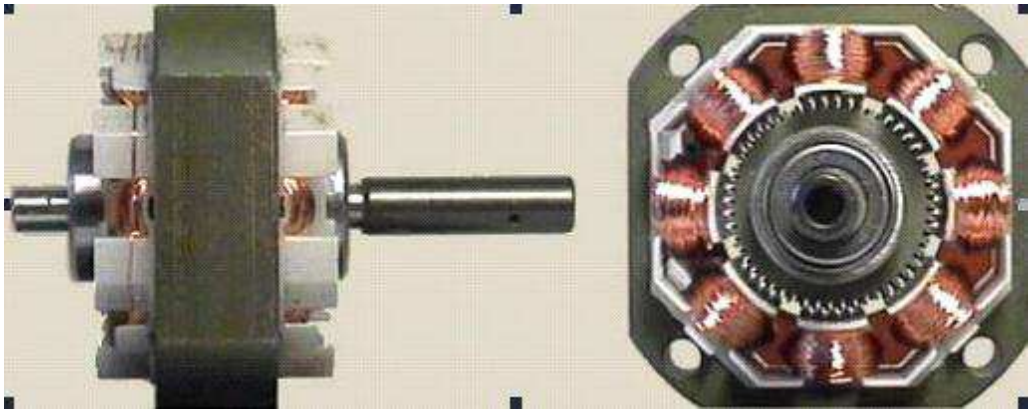


Figure 2.2 : moteur pas à pas

Fonctionnement :

Toute application impliquant l'utilisation d'un moteur pas à pas nécessite de collecter les informations indispensables à un bon dimensionnement :

- la masse de la charge à entraîner (en kg) ;
- son inertie (en $kg \cdot m^2$) ;
- le type d'entraînement mécanique (vis, courroie crantée, crémaillère, etc.) ;
- le type de guidage, afin d'estimer les frottements (secs et visqueux) ;
- les efforts de travail (en N) ;
- le déplacement le plus critique (distance en fonction d'un temps).

L'influence de la charge est directement liée au calcul du couple moteur via les paramètres du calcul inertiel (en $kg \cdot m^2$) et de l'accélération (en $m \cdot s^{-2}$). Pour des paramètres d'accélération et de chaîne cinématique identiques, un moteur pas à pas n'aura pas besoin du même couple selon la charge mise en jeu.

Pour une application industrielle, le dimensionnement d'un moteur pas à pas doit être calculé de façon rigoureuse ou être surdimensionné afin d'éviter tout problème de glissement par « perte de pas ». Le moteur pas à pas fonctionnant en boucle ouverte (sans asservissement), il ne récupère pas sa position de consigne en cas de glissement.

On trouve trois types de moteurs pas à pas :

1. moteur à réluctance variable
2. moteur à aimants permanents
3. moteur hybride, qui est une combinaison des deux technologies précédentes.

1. Moteurs à réluctance variable :

Les moteurs à réluctance variable (moteurs MRV) doivent leur nom au fait que le circuit magnétique qui les compose s'oppose de façon variable à sa pénétration par un champ magnétique.

Ces moteurs sont composés d'un barreau de fer doux et d'un certain nombre de bobines. Lorsque nous alimentons une bobine, le champ magnétique cherche à minimiser le passage dans l'air. Ainsi l'entrefer entre la bobine et le barreau se réduit. Le barreau s'aligne avec le

champ magnétique pour obtenir une réluctance minimale. Nous alimentonsla phase 1, puis la phase 2, puis la phase 3... Si nous souhaitons 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 (ici 6). Dès que nous alimentons 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 que nous alimentons une phase. Il faut 24 impulsions pour faire un tour complet. C'est un moteur 24 pas.

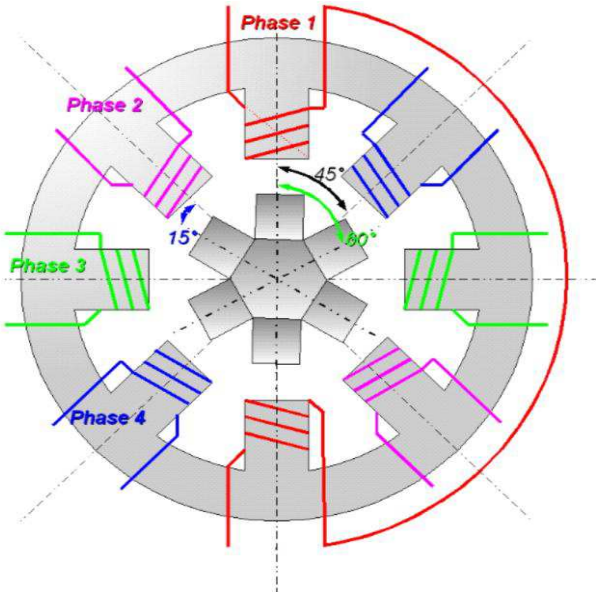


Figure 2.3: moteur a réluctance variable

a) Avantage :

Peu coûteux, d'une bonne précision. Dans l'exemple, avec seulement 4 enroulements, on obtient 24 pas (on peut facilement obtenir 360 pas). Le sens du courant dans la bobine n'a aucune importance.

b) Inconvénients :

Nécessite au moins trois bobinages, pour obtenir un cycle complet, pas de couple résiduel, c'est-à-dire que hors tension, le rotor est libre, ce qui peut être problématique pour ce genre de moteur. La fabrication est assez délicate, les entrefers doivent être très faibles.

2. Moteur a aimant permanent :

Les moteurs à aimants permanents sont semblables aux moteurs à réluctance variable, sauf que le rotor possède des 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'aimant va s'aligner avec le champ.

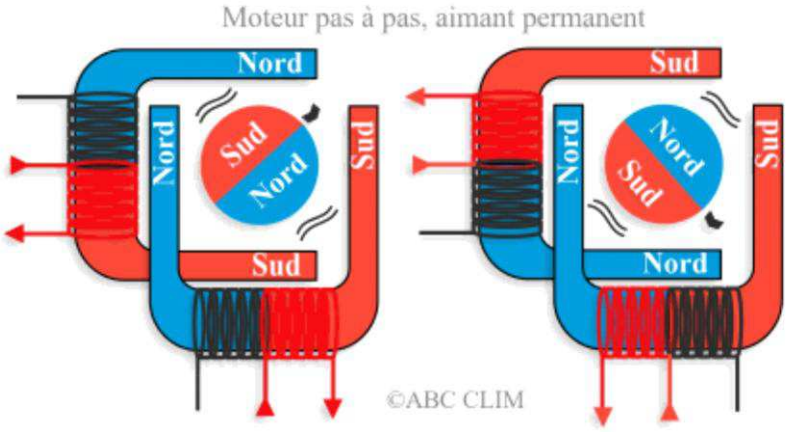


Figure 2.4: Moteurà aimant permanant

a. Moteur a aimant permanent bipolaire :

Chaque enroulement est séparé et il est alimenté de façon séquentielle soit en pôle positif ou négatif (bipolaire).

Fonctionnement :

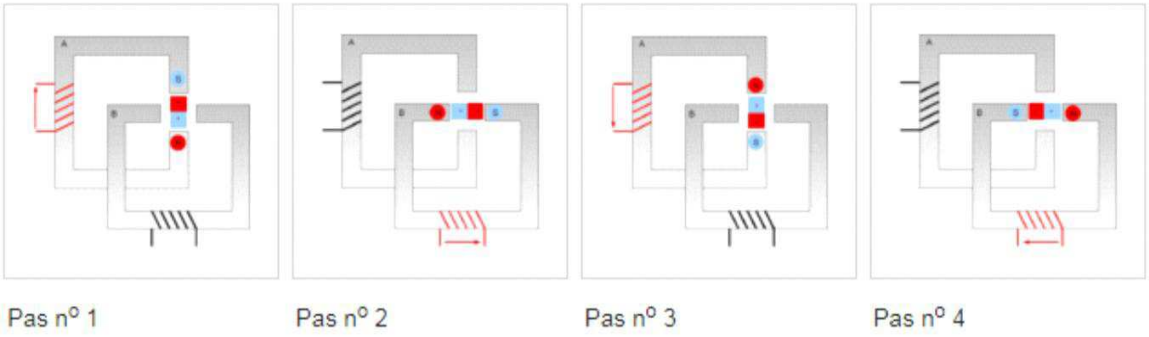


Figure 2.5: Fonctionnement d'un moteur à aimant permanant

Tableau 2.1 : récapitulatif de l'ordre des phases

Impulsion	Bobine A	Bobine A	Bobine B	Bobine B
T2	+	-		
T2			+	-
T3	-	+		
T4			-	+

Exemple avec un couple maximale :

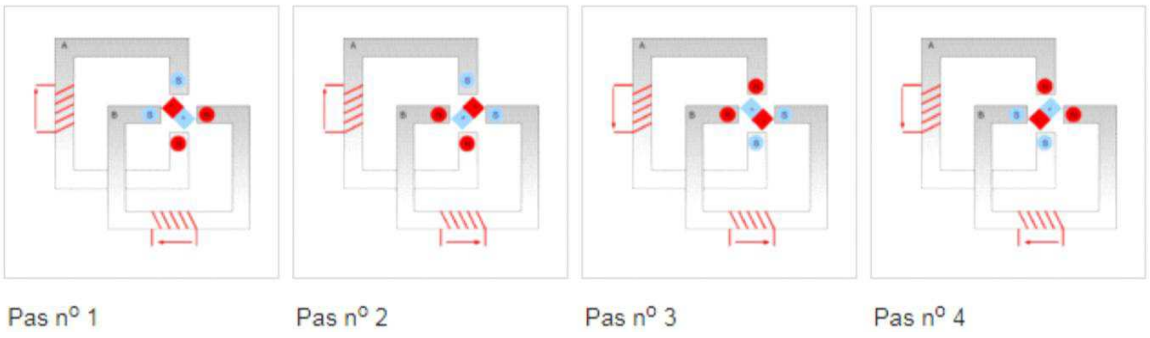


Figure 2.6 : couple maximal

Tableau 2.2 : Alimentation des bobines

Impulsion	Bobine A	Bobine A	Bobine B	Bobine B
T2	**+**	**-**	**+**	**-**
T2	**+**	**-**	**-**	**+**
T3	**-**	**+**	**-**	**+**
T4	**-**	**+**	**+**	**-**

Exemple avec demi-pas :

Figure 2.7: Demi -pas

b. Moteur a aimant permanent unipolaire :

Les enroulements sont raccordés avec un commun ou point milieu et ils sont alimentées avec des impulsions de même polarité (unipolaire).Dans les exemples précédents, nous avons vu que nous alimentons les enroulements dans les deux sens de courant, il existe des versions avec des demi-bobines (avec un point milieu). L'avantage est que l'on n'inverse jamais le sens du courant,

Donc la commande est plus simple. Tout le problème est que l'on « double » le nombre d'enroulements, donc le moteur est plus coûteux et encombrant, néanmoins cela reste très courant pour les petites puissances

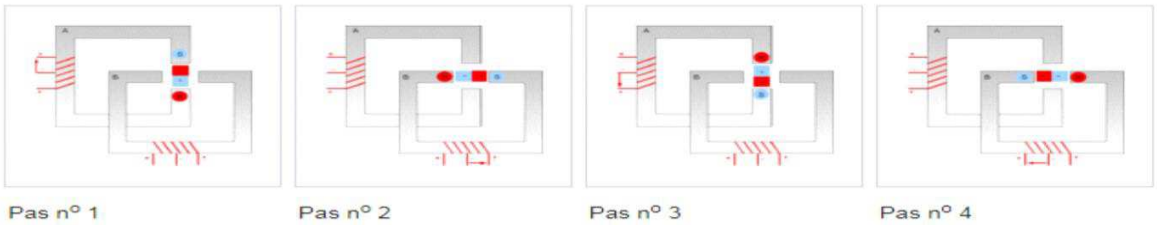


Figure 2.8: Moteur unipolaire

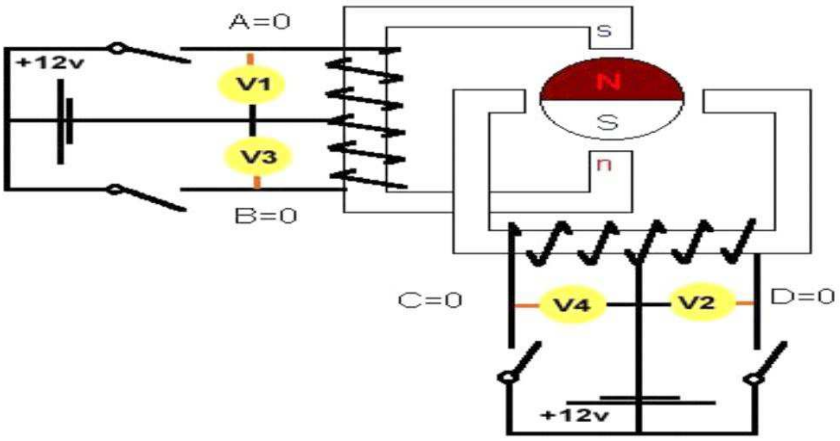


Figure 2.9 : schéma électrique d'un M.P.P unipolaire

2. Moteur pas à pas hybride :

Le moteur pas à pas hybride emprunte du moteur à aimant permanent et de la machine à réluctance variable. Il est donc à réluctance variable mais avec un rotor à aimants permanents. L'avantage est un nombre de pas très élevé.

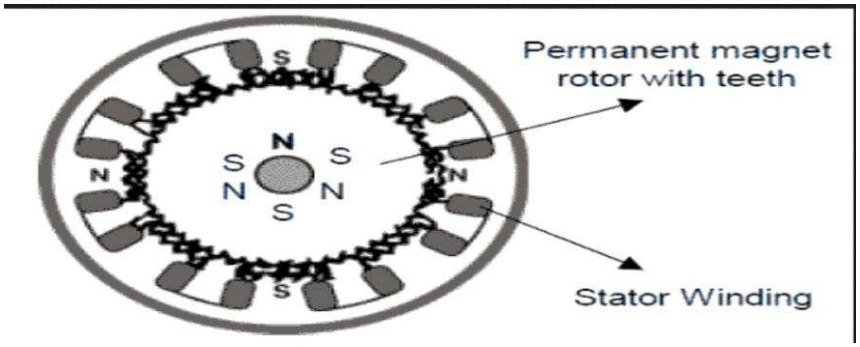


Figure 2.10: Moteur pas à pas hybride

Exemple : moteur pas à pas àaimant permanent bipolaire :

Pour commander ce type de moteur il ne faut 8 transistors :

- si on commande T1 et T4, alors nous alimentons dans un sens, soit nous alimentons en T2 et T3, nous changeons le sens de l'alimentation, donc le sens du courant.

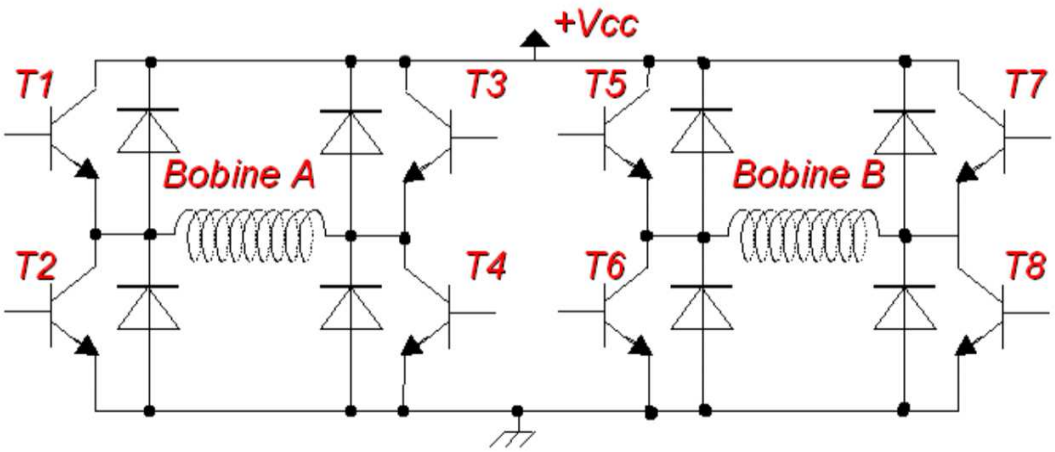


Figure 2.11 : schéma électrique M.P.P hybride

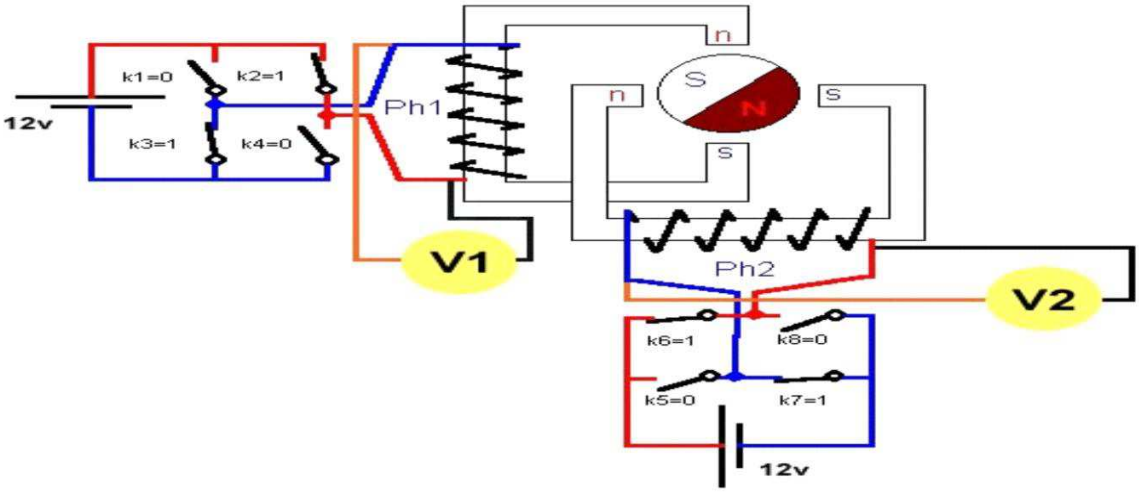


Figure 2.12 : Schéma fonctionnel d'un M.P.P Hybride

2.3. Actionneurs choisis pour exécuter la rotation :

Nous avons choisis les moteurs pas à pas pour leurs conceptions qui permettent d'avoir une précision dans la rotation du moteur, grâce à son déplacement de 200pas avec 1,8degré on peut atteindre n'importe quel angle.

1. Moteur pas à pas Nema23 :

Ce moteur pas à pas bipolaire (2303HS280AW) est un moteur nema23 choisis pour :

- Son couple de 1,5N.m qui est nécessaire pour faire pivoter la base du bras manipulateur ;
- son poids de 0,6kg.
- un angle de rotation de 1,8degré (200pas=360°).
- une alimentation qui varié entre 5 à 48 volt.

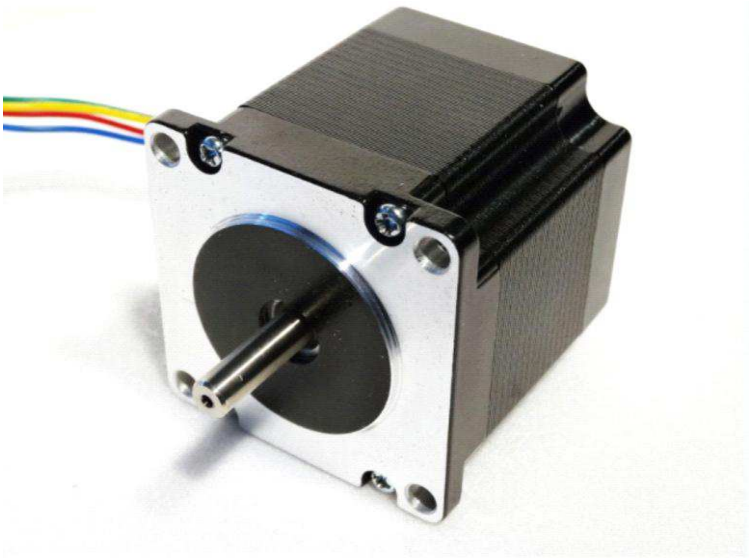


Figure 2.13 : Moteur pas à pas nema23 (2303HS280AW)

Description	length	Mounted rated current	Mounted holding current	Detent torque	Rotor inertia	Motor weight
stack	L «max »	amps	Nm	Nm	G.cm^2	Kg
	57mm	2,8	1.24	1.05	300	0.72

Tableau 2.3: Caractéristique du Moteur pas à pas nema23

2. Moteur pas à pas nema17 :

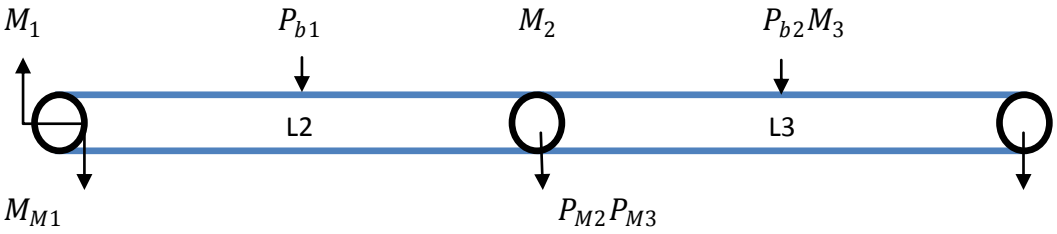
- le moteur Nema 17 est le plus petit moteur de la catégorie des moteurs pas à pas, ce moteur a été choisi pour :
- son poids qui ne dépasse pas les 280 g parce que plus on s'éloigne de la base avons besoin de plus de légèreté par rapport à la base du bras manipulateur ;
- son couple de 48mN.m (sois 1,20 kg.cm) qui est assez nécessaire pour faire pivoter le dernier axe du bras manipulateur ;
- son alimentation de 5v qui ne consomme pas beaucoup d'énergie.

Figure 2.14:Aperçu du moteur pas à pas nema17

Description	length	Mounted rated current	Mounted holding current	Detent torque	Rotor inertia	Motor weight
stack	L «max »	amps	Nm	mNm	G.cm^2	Kg
	39mm	2	0.48	15	57	0.16

Tableau 2.4: Caractéristique du moteur pas à pas nema17

Calcule couple nécessaire :



Le couple minimal pour supporté le bras :

$$\sum M_{ext} = 0$$

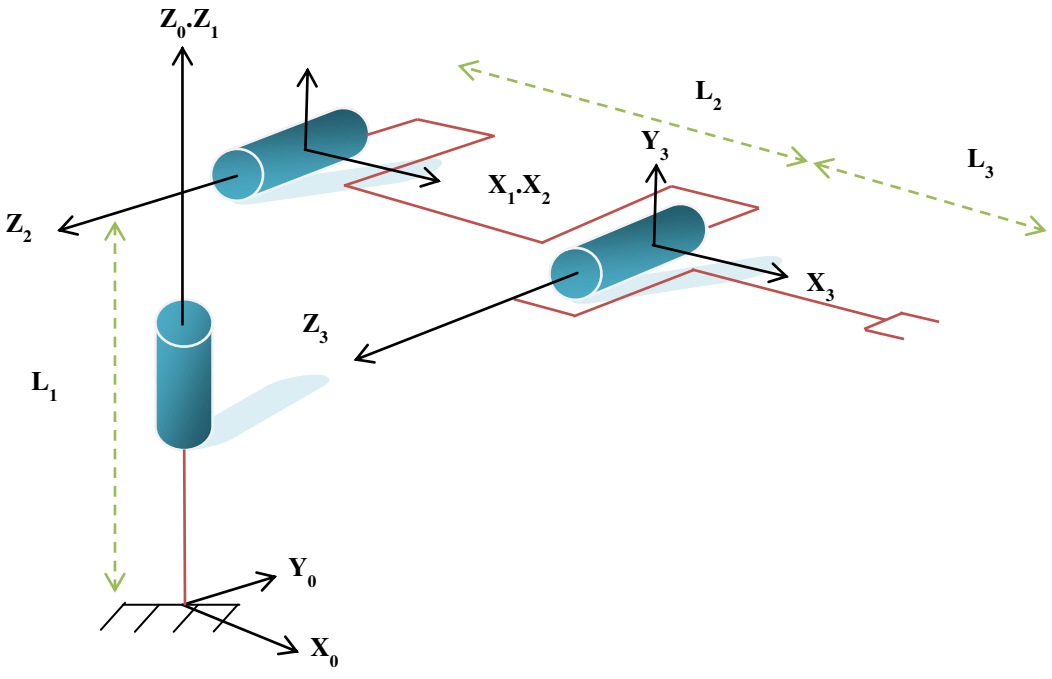
$$\left(P_{b1} * \frac{L_2}{2}\right) + (P_{M2} * L_2) + \left(P_{b2} * \left(L_2 + \frac{L_3}{2}\right)\right) + \left(P_{M3} * (L_2 + L_3)\right) = M_{M1} \quad [1]$$

Données: $\boldsymbol{L_2 = L_3 = 0.3m}$, $\boldsymbol{m_{M2} = m_{M3} = 0.16kg}$, $\boldsymbol{m_{b1} = m_{b2} = 0.182\ kg}$, $\boldsymbol{g = 9.81}$m/s2

Application numérique :

$$M_{M1} > 2.48N.m$$

Projection du bras manipulateur :



2.4. Calcule des angles de rotation des Moteur

Model géométrique direct (MGD)

	θ_j	r_j	α_j	d_j
R_0 - R_1	θ_1	L_1	0	0
R_1 - R_2	θ_2	0	$\frac{\pi}{2}$	0
R_2 - R_3	θ_3	0	0	L_2

Le remplacement des paramètres entre chaque deux repères successifs nous a donné Les matrices de passage suivantes :

$$^{i-1}T_i=\begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & d_i \\ \cos\alpha_i \sin\theta_i & \cos\theta_i \cos\alpha_i & -\sin\alpha_i r_i \sin\theta_i \\ \sin\alpha_i \sin\theta_i & \cos\theta_i \sin\alpha_i & \cos\alpha_i\ r_i \cos\theta_i \\ 0 & 0 & 0 & 1 \end{bmatrix} [2]$$

- après avoir remplacer les paramétre dans la matrice de passage on a les trois matrice suivante :

$$^{O}T_1=\begin{bmatrix} \text{Cos}\,\theta_1 & -\text{Sin}\,\theta_1 & 0 & 0 \\ \text{Sin}\,\theta_1 & \text{Cos}\,\theta_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} [3] \quad ^{1}T_2=\begin{bmatrix} \text{Cos}\,\theta_2 & -\text{Sin}\,\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \text{Sin}\,\theta_2 & \text{Cos}\,\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} [4]$$

$$^{2}T_3=\begin{bmatrix} \text{Cos}\,\theta_3 & -\text{Sin}\,\theta_3 & 0 & L_2 \\ \text{Sin}\,\theta_3 & \text{Cos}\,\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} [5]$$

- on multiplié les trois matrices 3,4 et 5 pour trouver la matrice homogène $^{0}T_3$:

$$^{0}T_3 = {}^{O}T_1 . {}^{1}T_2 . {}^{2}T_3 .$$

$$^{O}T_3=\begin{bmatrix} \text{Cos}\,\theta_1 & -\text{Sin}\,\theta_1 & 0 & 0 \\ \text{Sin}\,\theta_1 & \text{Cos}\,\theta_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \text{Cos}\,\theta_2 & -\text{Sin}\,\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \text{Sin}\,\theta_2 & \text{Cos}\,\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \text{Cos}\,\theta_3 & -\text{Sin}\,\theta_3 & 0 & L_2 \\ \text{Sin}\,\theta_3 & \text{Cos}\,\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- On trouve donc la matrice de passage du 1er repére au 3éme repére :

$$^{0}T_3=\begin{bmatrix} \text{Cos}\theta_1\text{Cos}(\theta_2+\theta_3) & -\text{Cos}\,\theta_1\,\text{Sin}(\theta_2+\theta_3) & \text{Sin}\,\theta_1 & L_2\,\text{Cos}\,\theta_2\,\text{Cos}\,\theta_1 \\ \text{Sin}\theta_1\text{Cos}(\theta_2+\theta_3) & -\text{Sin}\theta_1\text{Sin}(\theta_2+\theta_3) & -\text{Cos}\,\theta_1 & L_2\,\text{Cos}\,\theta_2\,\text{Sin}\,\theta_1 \\ \text{Sin}(\theta_2+\theta_3) & \text{Cos}(\theta_2+\theta_3) & 0 & L_2\,\text{Sin}\,\theta_2+L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} [6]$$

- Pour obtenir P_x , $P_y et\ P_z$ on multiplié la dérniere colone de la matrice ${}^0T_3$ avec le vecteur de l'organe terminal :

  Vecteur de l'organe terminal : $\begin{bmatrix} L_3 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ [7]

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Cos}\theta_1\text{Cos}(\theta_2+\theta_3) & -\text{Cos}\,\theta_1\,\text{Sin}(\theta_2+\theta_3) & \text{Sin}\,\theta_1 & L_2\,\text{Cos}\,\theta_2\,\text{Cos}\,\theta_1 \\ \text{Sin}\theta_1\text{Cos}(\theta_2+\theta_3) & -\text{Sin}\theta_1\text{Sin}(\theta_2+\theta_3) & -\text{Cos}\,\theta_1 & L_2\,\text{Cos}\,\theta_2\,\text{Sin}\,\theta_1 \\ \text{Sin}(\theta_2+\theta_3) & \text{Cos}(\theta_2+\theta_3) & 0 & L_2\,\text{Sin}\,\theta_2+L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} L_3 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} (L_2\text{Cos}\theta_2 + L_3\text{Cos}(\theta_2+\theta_3))\text{Cos}\,\theta_1 \\ (L_2\text{Cos}\theta_2 + L_3\text{Cos}(\theta_2+\theta_3))\text{Sin}\,\theta_1 \\ L_1 + L_2\text{Sin}\theta_2 + L_3\,\text{Sin}(\theta_2+\theta_3) \\ 1 \end{bmatrix} \quad [8]$$

Alors :

$$P_x = (L_2\text{Cos}\theta_2 + L_3\text{Cos}(\theta_2+\theta_3))\text{Cos}\,\theta_1 \quad [9]$$

$$P_y = (L_2\text{Cos}\theta_2 + L_3\text{Cos}(\theta_2+\theta_3))\text{Sin}\,\theta_1 \quad [10]$$

$$P_z = L_1 + L_2\text{Sin}\theta_2 + L_3\,\text{Sin}(\theta_2+\theta_3) \quad [11]$$

Modèle géométrique inverse

Soit :

$$P = [Px, Py, Pz]^T$$

$$P_x = [L_2\ \cos(\theta_2) + L_3\cos(\theta_2+\theta_3)]\cos\theta_1 \quad (12)$$

$$P_y = [L_2\ \cos(\theta_2) + L_3\cos(\theta_2+\theta_3)]\sin\theta_1 \quad (13)$$

$$P_z = L_1 + L_2\text{Sin}\theta_2 + L_3\,\text{Sin}(\theta_2+\theta_3) \quad (14)$$

$$P_x^2 = \cos^2\theta_1\,[L_2\cos\theta_1 + L_3\cos(\theta_1+\theta_2)]^2 \quad (15)$$

$$P_y^2 = \sin^2\theta_1[L_2\cos\theta_2 + L_3\cos(\theta_2+\theta_3)]^2 \quad (16)$$

Utilisant :

$$\cos(a-b) = \sin(a)\sin(b) + \cos(a)\cos(b)$$
$$\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$$
$$\sin(a+b) = \sin(a)\cos(b) - \cos(a)\sin(b)$$

$(1)/(2)$ Donne :

$$\frac{P_y}{P_x} = \frac{\sin\theta_1}{\cos\theta_1} = \tan\theta_1 \Longrightarrow \boldsymbol{\theta_1} = \tan^{-1}\frac{P_y}{P_x}$$

(4)+(5) Donne :

$$\cos\theta_3 = \frac{P_x^2 + P_y^2 + (P_z - L_1)^2 - L_2^2 - L_3^2}{2L_2L_3}$$

$$\theta_{3=} \cos^{-1} \frac{P_x^2 + P_y^2 + (P_z - L_1)^2 - L_2^2 - L_3^2}{2L_2L_3}$$

Nous posons :

$$\begin{Bmatrix} K_1 = L_2 + L_3\cos\theta_3 \\ K_2 = L_3\sin\theta_3 \end{Bmatrix}$$

$$\frac{P_x}{\cos\theta_1} = K_1\cos\theta_2 - K_2\sin\theta_2 \quad (17)$$

$$(P_z - L_1) = K_1\sin\theta_2 - K_2\cos\theta_2 \quad (18)$$

$$(6) \Longrightarrow \begin{Bmatrix} \cos\theta_2 = \frac{{}^{P_x}/_{\cos\theta_1} - K_2\sin\theta_2}{K_1} \ldots . (a) \\ \sin\theta_2 = \frac{{}^{P_x}/_{\cos\theta_1} - K_1\cos\theta_2}{K_2} \ldots . (b) \end{Bmatrix}$$

En remplaçant : (a) ou (b) dans (18) :

$$\boldsymbol{\theta_2} = \begin{Bmatrix} \cos^{-1} \frac{K_1\,{}^{P_x}/_{\cos\theta_1} - K_2(P_z - L_1)}{K_1^2 + K_2^2} \\ \sin^{-1} \frac{K_1(P_z - L_1) + K_2\,{}^{P_x}/_{\cos\theta_1}}{K_1^2 + K_2^2} \end{Bmatrix}$$

2.5. Capteurs :

Un capteur est un organe de prélèvement d'informations qui élabore à partir d'une grandeur physique (Information entrante) une autre grandeur physique de nature différente (Information sortante : très souvent électrique). Cette grandeur, représentative de la grandeur prélevée, est utilisable à des fins de mesure ou de commande. Les capteurs sont composés de deux éléments : corps d'épreuve, élément sensible.

Capteur passif : Il s'agit généralement d'impédances (résistance, inductance, capacité) dont l'un des paramètres déterminants est sensible à la grandeur mesurée.

Capteur actif : Fonctionnant en générateur, un capteur actif est généralement fondé dans son principe sur un effet physique qui assure la conversion en énergie électrique de la forme d'énergie propre à la grandeur physique à mesurer (énergie thermique, mécanique ou de rayonnement)

Dans la majorité des cas, les signaux issus d'un capteur seront électriques, ce qui veut dire qu'ils peuvent être des tensions comme des courants. Il peut y avoir trois types de signaux de sortie différents :

- Signal binaire TOR (Tout Ou Rien).
- Signal analogique.
- Signal numérique.

Les effets physiques les plus rencontrés sont :

1. Effet thermoélectrique : Un circuit formé de deux conducteurs de nature chimique différente, dont les jonctions sont à des températures T1 et T2, est le siège d'une force électromotrice d'origine thermique (T1, T2).

2. Effet piézo-électrique : L'application d'une contrainte mécanique à certains matériaux dits Piézo- Electriques (le quartz par exemple) entraîne l'apparition d'une déformation et d'une même charge électrique de signe différent sur les faces opposées.

3. Effet d'induction électromagnétique : La variation du flux d'induction magnétique dans un circuit électrique induit une tension électrique (détection de passage d'un objet métallique).

4. Effet photo-électrique : La libération de charges électriques dans la matière sous l'influence d'un rayonnement lumineux ou plus généralement d'une onde électromagnétique.

5. Effet Hall : Un champ magnétique B et un courant électrique I créent dans le matériau une différence de potentiel UH.

6. Effet photovoltaïque : Des électrons et des trous sont libérés au voisinage d'une jonction PN illuminée, leur déplacement modifie la tension à ses bornes.

1. Capteur Optique :

- Un capteur optique est un dispositif capable de détecter l'intensité ou la longueur d'onde des photons.

On les utilise pour détecter un grand nombre de phénomène :

- l'intensité lumineuse bien-sûr
- la chaleur (capteur pyrométrique) :
- la présence
- la couleur (et donc certains gaz ou produits chimiques)

Mais aussi pour: acquérir des informations numériques transmises par des conducteurs (fibres) optiques.

Nous avons mentionné au pare avant deux catégories de capteur actif et passif, pour les capteurs passif nous avons des capteurs optiques à photorésistance, dans la catégorie des capteurs actif nous avons 3 types :

- capteur photodiodes
- capteur photovoltaïque
- capteur phototransistor

Le choix c'est porté sur un capteur optique phototransistor (actif) qui va nous permettre d'indiquer les positions des différents moteurs du bras manipulateur.

2. Capteur Phototransistor :

Un phototransistor un composant semi-conducteur ayant la capacité de détecter un rayonnement du domaine optique et de la transformer en signal électrique.

➢ **Fonctionnement:** Un phototransistor est un composant qui possède la même structure qu'un transistor bipolaire classique, mais dont la jonction collecteur - base peut être éclairée par un rayonnement lumineux.

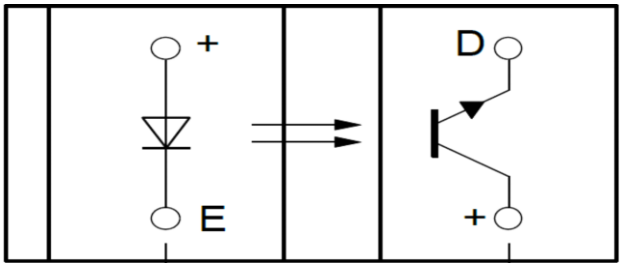


Figure 2.17: Schéma du fonctionnement de l'effet phototransistor

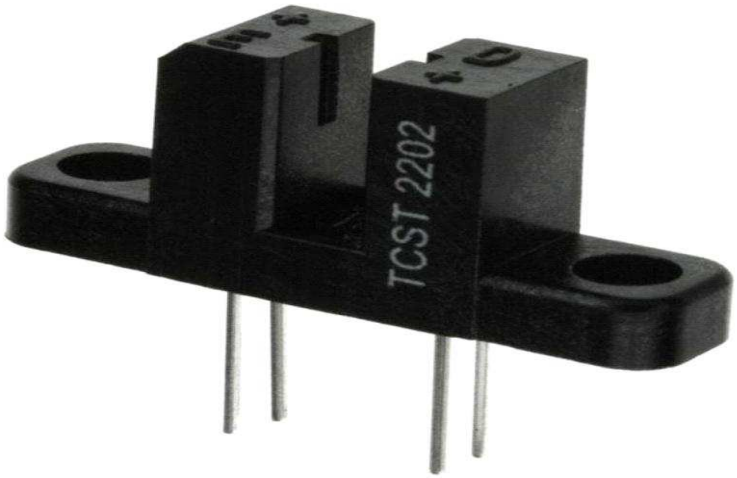


Figure 2.18: Aperçu du capteur optique à fourche à effet phototransistor

2.6. Conclusion

Dans ce chapitre nous avons vu les capteurs et les actionneurs utilisés dans le robot manipulateur, leur différent type ainsi que les calcule des angles de rotation qui vont être utilisée dans le programme du bras manipulateur.

Chapitre 3 :

Microcontrôleur

1. MICROCONTROLEURS

1.1. Introduction :

La plus grande partie des systèmes électroniques complexes utilisés de nos jours sont des systèmes embarqués : téléphones mobiles, horloges, baladeurs, récepteurs GPS, électroménager, automobile, transport aérien/maritime/fluvial. Les systèmes embarqués se démarquent des systèmes informatiques traditionnels selon plusieurs aspects :

Ils sont soumis à des contraintes de taille (intégration), de consommation électrique (autonomie) et de coût importantes (grande série)

Ils doivent communiquer avec des dispositifs d'entrées-sorties (IO) : boutons, relais, résistances variables, moteurs électriques, LED, circuits intégrés logiques, etc.

Ils n'ont parfois aucun dispositif d'interface homme-machine : ni clavier, ni écran, ni disque, ni imprimante, etc. Par exemple, le contrôleur d'injection de carburant du moteur d'une automobile est totalement invisible pour le conducteur.

Figure 3.1: Aperçu d'un calculateur moteur (système embarqué)

1.2. Microcontrôleur :

Définition

Un microcontrôleur se présente comme étant une unité de traitement de l'information de type microprocesseur contenant tous les composants d'un système informatique, à savoir microprocesseur, des mémoires et des périphériques (ports, timers, convertisseurs…). Chaque fabricant a sa ou ses familles de microcontrôleurs.

Une famille se caractérise par un noyau commun (le microprocesseur, le jeu d'instruction…). Ainsi les fabricants peuvent présenter un grand nombre de pins qui s'adaptent plus au moins à certaines tâches. Mais un programmeur connaissant une famille n'a pas besoin d'apprendre à utiliser chaque membre, il lui faut connaître juste ces différences par rapport au père de la famille. Ces différences sont souvent, la taille des mémoires, la présence ou l'absence des périphériques et leurs nombres.

Caractéristiques :

- **Séparation des mémoires de programme et de données:** On obtient ainsi une meilleure bande passante et des instructions et des données pas forcément codées sur le même nombre de bits.
- **Communication avec l'extérieur seulement par des ports :** il ne possède pas de bus d'adresses, de bus de données et de bus de contrôle comme la plupart des microprocesseurs.
- **Utilisation d'un jeu d'instructions réduit** : d'où le nom de son architecture : RISC (Reduced Instructions Set Construction). Les instructions sont ainsi codées sur un nombre réduit de bits, ce qui accélère l'exécution (1 cycle machine par instruction sauf pour les sauts qui requirent 2 cycles). En revanche, leur nombre limité oblige à se restreindre à des instructions basiques, contrairement aux systèmes d'architecture CISC (Complex Instructions Set Construction) qui proposent plus d'instructions donc codées sur plus de bits mais réalisant des traitements plus complexes.

Figure 3.2 : Aperçu d'un microcontrôleur

Domaine d'utilisation d'un microcontrôleur :

Les microcontrôleurs sont fréquemment utilisés dans les systèmes embarqués, comme les contrôleurs des moteurs automobiles, les télécommandes, les appareils de bureau, l'électroménager, les jouets, la téléphonie mobile.

1.3. Différente partie du microcontrôleur :

Le processeur est l'élément central d'un système informatique : il interprète les instructions et traite les données d'un programme. Il a besoin de certains éléments externes pour fonctionner :

- ✓ Une horloge pour le cadencer (en général à quartz).
- ✓ De la mémoire pour stocker les variables durant l'exécution du programme (mémoire vive RAM) et le programme d'une mise sous tension à l'autre (mémoire morte ROM). Si l'on

conçoit un système assigné à une tâche bien particulière (ce qui est généralement le cas des systèmes embarqués), le programme n'est pas amené à changer. Il peut donc être stocké dans une mémoire morte (ROM) ;

- ✓ Des périphériques (pour interagir avec le monde extérieur).

Ces éléments sont reliés par 3 bus :

- ✓ le bus d'adresse qui permet au microprocesseur de sélectionner la case mémoire ou le périphérique auquel il veut accéder pour lire ou écrire une information (instruction ou donnée) ;
- ✓ le bus de données qui permet le transfert des informations entre les différents éléments ; ces informations seront soit des instructions, soit des données en provenance ou à destination de la mémoire ou des périphériques.
- ✓ le bus de contrôle qui indique si l'opération en cours est une lecture ou une écriture, si un périphérique demande une interruption pour faire remonter une information au processeur.

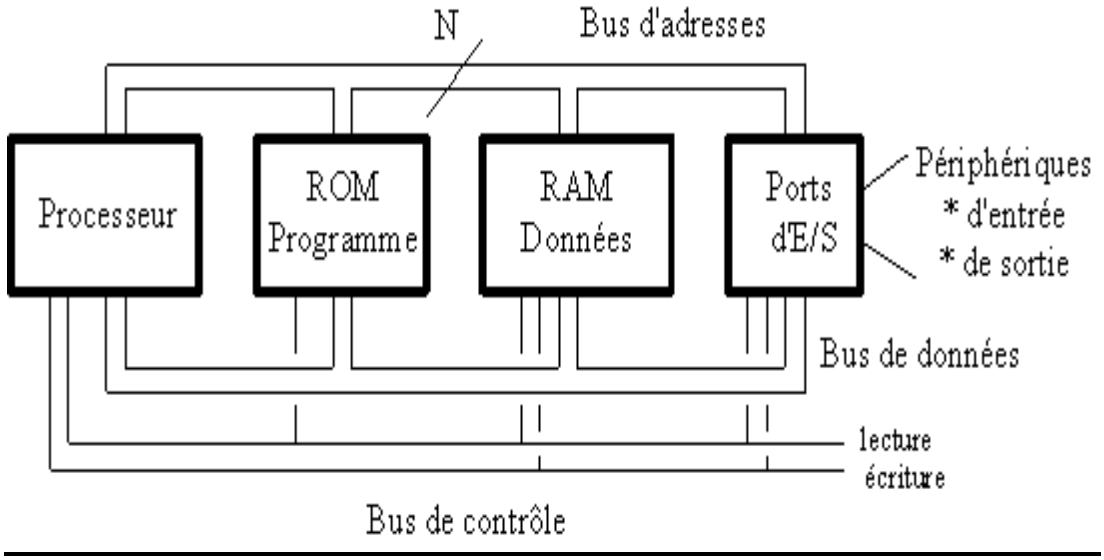


Figure 3.3. : Structure interne d'un microcontrôleur

1.4. Famille des Microcontrôleur PIC :

Il existe deux familles opposées, Les RISC et les CISC (Complexe Instruction SET Computer), chez les CISC, on diminue la vitesse de traitement mais les instructions sont plus complexes, plus puissantes et donc plus nombreuses. Il s'agit donc d'un choix de stratégie.

Les PICs sont des composants RISC (Reduce Instructions Construction Set), ou encore composant à jeu d'instructions réduit. L'avantage est que plus on réduit le nombre d'instructions, plus facile et plus rapide en est le décodage, et plus vite le composant fonctionne.

La famille des PICs est subdivisée en 3 grandes familles :

- ✓ La famille BaseLine, qui utilise des mots d'instructions de 12 bits
- ✓ la famille Mid-Range, qui utilise des mots de 14 bits
- ✓ la famille High-End, qui utilise des mots de 16 bits (18FXXX).

.

Identification d'un microcontrôleur pic :

Un PIC est identifié par un numéro de la forme suivant : xx(L)XXyy –zz

- **xx** : Famille du composant (12, 14, 16, 17, 18)
- **L** : Tolérance plus importante de la plage de tension
- **XX** : Type de mémoire de programme
- **C** - EPROM (**E**rasable **P**rogrammable **R**ead **O**nly **M**emory) ouEEPROM (**E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory).
- **CR**–PROM (**P**rogrammable **R**ead **O**nly **M**emory).
- **F** - FLASH
- **yy** : Identification
- **zz** : Vitesse maximum du quartz

Définition des types de mémoire :

- **Mémoire EPROM :** sont effaçables et programmables par l'utilisateur.
- **Mémoire EEPROM :** sont effaçables et programmables par l'utilisateur. Elles sont plus faciles à effacer que les **EPROM** car elles sont effaçables électriquement donc sans manipulations physiques.
- **Mémoire Flash :** est un compromis entre les mémoires RAM et ROM : c'est une mémoire non volatile, comme les mémoires mortes, mais qui possède par ailleurs les caractéristiques d'une mémoire vive.

Exemple sur l'identification des microcontrôleurs PIC :

Un microcontrôleur Pic est identifié de cette manière : PIC 16F84-10

16	**F**	**84**	**-10**
Famille des Mid-Range	Une mémoire de type flash	identifiant du PIC	Quartz à10MHz max

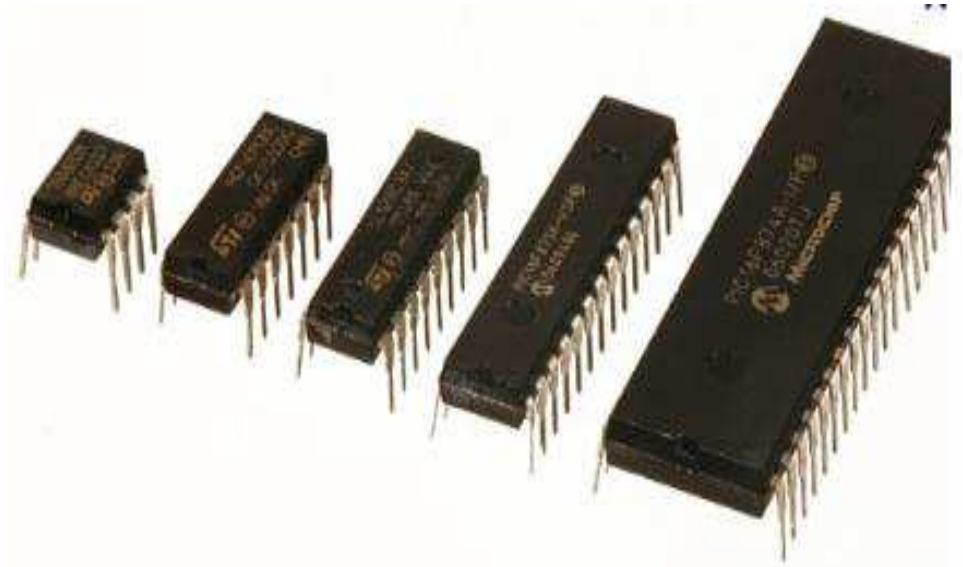


Figure 3.4 : différents microcontrôleurs PIC

1.5. Microcontrôleurs PIC utilisé :

Nous avons choisi d'utiliser le microcontrôleur PIC 16F877 vue a ça caractéristiques adéquate à notre réalisation de ce bras manipulateur.

Microcontrôleur PIC 16F877 :

- **Structure interne :** caractéristique CPU

CPU à architecture RISC (8 bits)

1) Mémoire programme de 8 Kmots de 14 bits (Flash),
2) Mémoire donnée de 368 Octets,
3) EEPROM donnée de 256 Octets,
4) 14 sources interruptions.
5) Générateur d'horloge de type RC ou quartz (jusqu'à‟ à 20 MHz).
6) 05 ports d'entrée sortie
7) Fonctionnement en mode sleep pour réduction de la consommation,
8) Programmation par mode ICSP (In Circuit Serial Programming) 12V ou 5V,
9) Possibilité aux applications utilisateur d'accéder à la mémoire programme.

- **Caractéristique des périphériques :**

1) Timer0 : Timer/Compteur 8 bits avec un prédiviseur 8 bits
2) Timer1 : Timer/Compteur 16 bits avec une prédivision de 1, 2, 4, ou 8 ; il peut être incrémenté en mode veille (Sleep), via une horloge externe,
3) Timer2 : Timer 8 bits avec deux diviseurs (pré et post diviseur)
4) Deux modules « Capture, Compare et PWM » : Module capture 16 bits avec une résolution max. 12,5 ns, Module Compare 16 bits avec une résolution max. 200 ns, Module PWM avec une résolution max. 10 bits,
5) Convertisseur Analogiques numériques multicanal (8 voies) avec une conversion sur 10 bits, Synchronous Serial Port (SSP) SSP, Port série synchrone en mode I2C (mode maitre/esclave),
6) Universel SynchronousAsynchronousReceiverTransmitter (USART) : Port série universel, mode asynchrone (RS232) et mode synchrone.

Brochage du microcontrôleur PIC 16F877A :

- 2 broches Vss (Gnd 0volt) et 2 broches Vdd (5volt)
- 4 broches pour le port A
- 8 broches pour le port B
- 8 broches pour le port C
- 5broches pour le convertisseur ANL/NUM (en configurant le port A et E)
- 8 broches pour le port D
- 3 broches pour le port E
- Entrée RX et TX

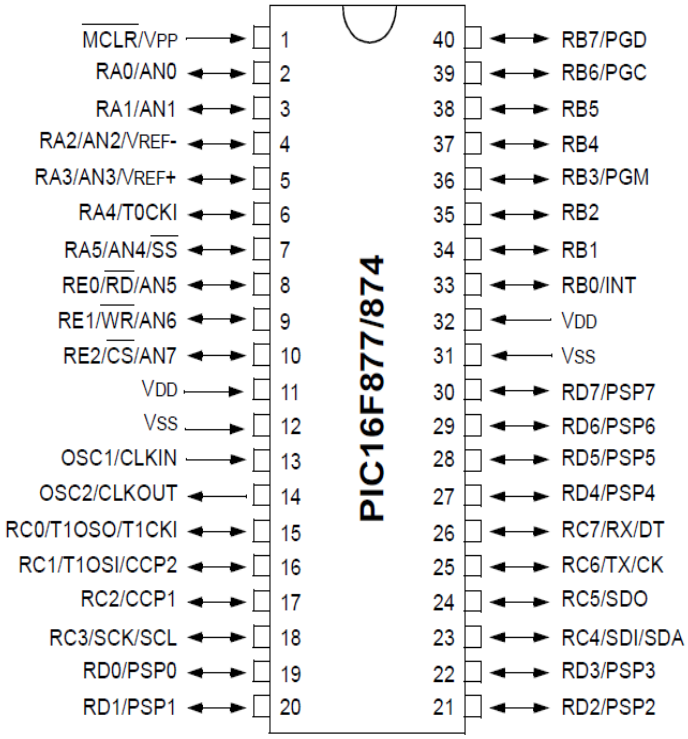


Figure 3.5 : brochage Du microcontrôleur PIC16F877

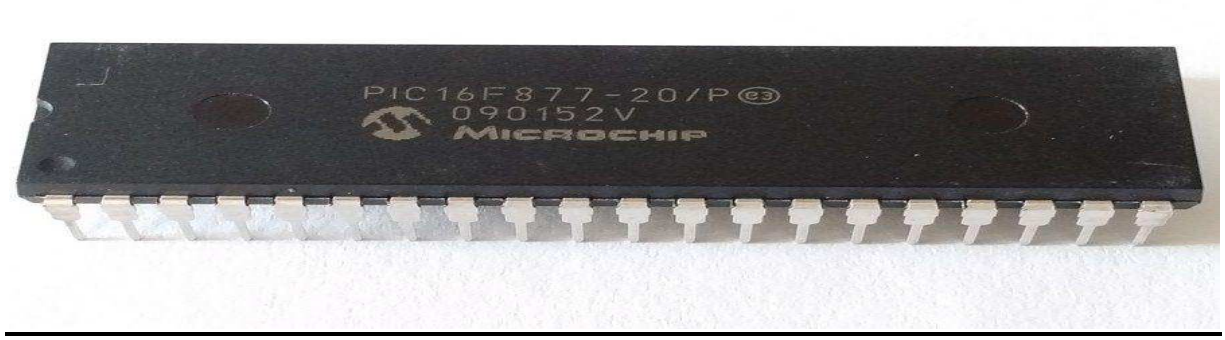
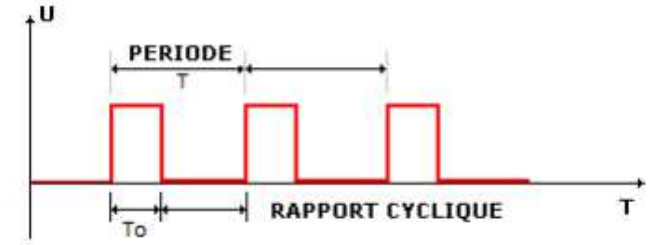


Figure 3.6 : Aperçu du microcontrôleur PIC 16F877

Il y'a deux méthodes pour contrôler la vitesse d'un moteur pas à pas sur le microcontrôleur PIC : La méthode de la Modulation de Largeur d'Impulsions (MLI) et Contrôler la vitesse sur le programme utilisé.

Pulse Width Modulation(PWM) : La **modulation de largeur d'impulsions** (**MLI**) est une technique couramment utilisée pour synthétiser des signaux continus à l'aide de circuits à fonctionnement tout ou rien, ou plus généralement à états discrets.
Cela consiste à alimenter le moteur avec une tension en créneaux, la tension moyenne dépend alors du rapport cyclique T_0/T donc la vitesse varie en fonction de cette tension moyenne.



$$Umoy = \frac{1}{T}\int_0^T u(t)dt = \frac{1}{T}[Umax.t]_0^{To} = Umax.\frac{To}{T}$$

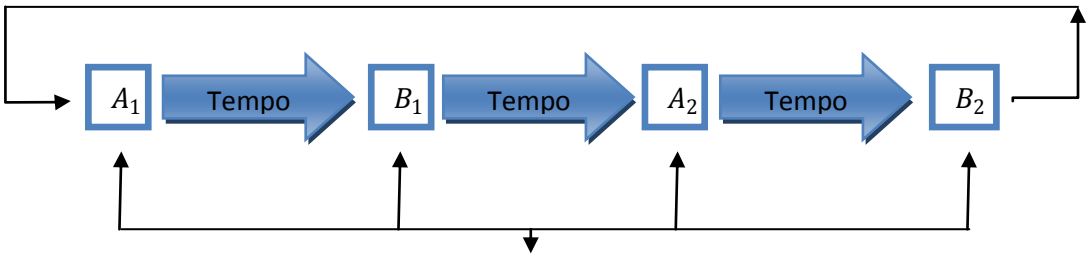
Le rapport cyclique est exprimé en %

Figure **3.7**: le rapport cyclique

Contrôler la vitesse sur le programme : Dans notre cas nous avons utilisé cette méthode, en programment notre software de façon à pouvoir contrôler la vitesse du moteur pas à pas en varient le temps de l'envoi des impulsions en boucle de ces quatre séquence l'une après l'autre, c'est-à-dire à chaque fois qu'on envoi une impulsion à ce moteur nous avons une temporisation de temps qu'on commande, quand en diminue cette temporisation la vitesse du moteur augmente dés qu'en augmente la temporisation la vitesse du moteur diminue.

La temporisation utilisé est sous forme de boucle qui permet au microcontrôleur de faire des calculs quelconque, comme décrémenter un compteur jusqu'à est ce qu'il devienne nul ce qui permettra d'ajouter un temps voulu entre ces séquences du moteur.
Pour l'envoi des séquences on doit alimenter une bobine a la fois, pour ce moteur on a 4

Voici un schéma synoptique qui définitcette méthode qui est envoyé en boucle
Séquences nommées: $A_1,\ A_2,\ B_1,\ B_2$.
Temporisation = Tempo



Chaque commande envoyée par le microcontrôleur est une impulsion

1.6. Langage assembleur :

Un **langage d'assemblage** ou **langage assembleur** est, en programmation informatique, un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain. Les combinaisons de bits du langage machine sont représentées par des symboles dits « mnémoniques » (du grec *mnêmonikos*, relatif à la mémoire), c'est-à-dire faciles à retenir. Le programme assembleur convertit ces mnémoniques en langage machine, ainsi que les valeurs (écrites en décimal) en binaire et les libellés d'emplacements en adresses, en vue de créer par exemple un fichier objet ou un fichier exécutable.

Dans la pratique courante, le même terme *assembleur* est utilisé à la fois pour désigner le langage d'assemblage et le programme assembleur qui le traduit. On parle ainsi de « programmation en assembleur ».

Notre outil choisi pour accomplir cette fonction est le logiciel : MPLAB IDE (v8.00) car c'est un environnement de développement dédié aux microcontrôleurs PIC de Microchip.

Logiciel MPLAB :

Mpblab IDE est un logiciel qui fonctionne sur un PC (windows,MacOS,Linux) pour développer des applications pour les microcontrôleurs Microchip et les contrôleurs de signaux numériques. C'est ce que nous appelons un environnement de développement intégré(IDE), car il fournit un environnement intégré unique pour développer du code pour les microcontrôleurs intégrés.

Différente fonction du MPLAB :

- Un éditeur (afin de créer les fichiers de programme en langage C ou en assembleur).
- Un compilateur (qui dépend de l'installation effectuée, ainsi que de la famille de microcontrôleurs PIC utilisée).
- Un assembleur (qui dépend de l'installation effectuée, ainsi que de la famille de microcontrôleurs PIC utilisée).
- Un linker (qui dépend de l'installation effectuée, ainsi que de la famille de microcontrôleurs PIC utilisée).
- Un simulateur.

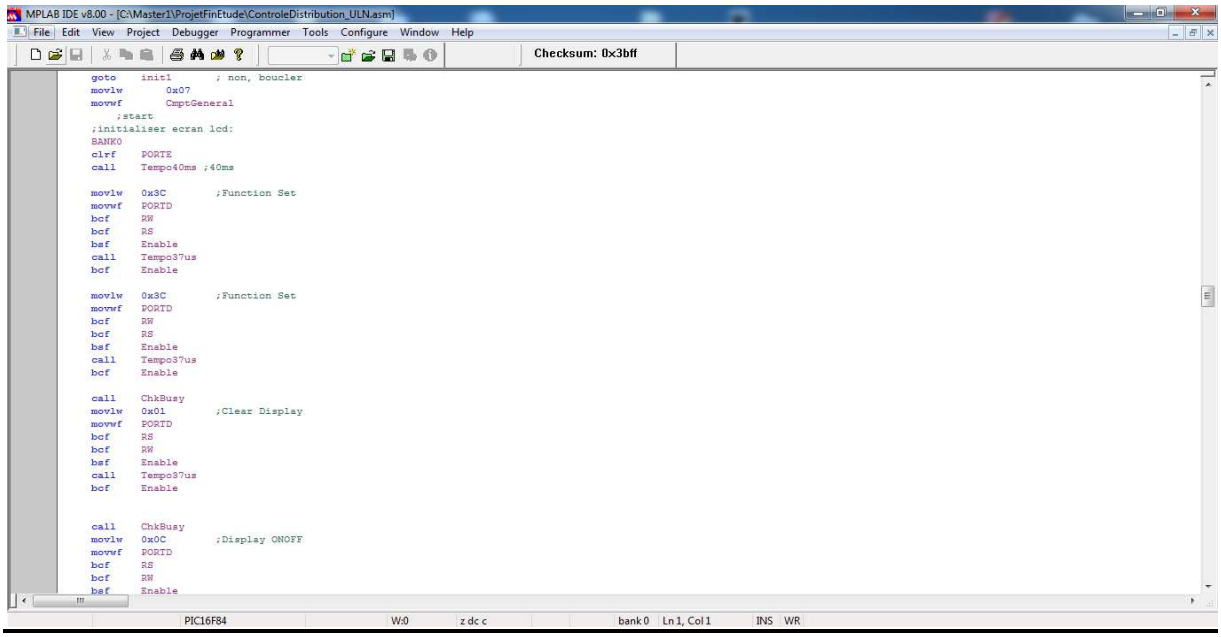


Figure 3.8 : Aperçu du logiciel MPLAB

Le langage machine est le seul langage qu'un processeur puisse exécuter. Or chaque famille de processeurs utilise un jeu d'instructions différent, voici un exemple des différentes instructions en langage mnémonique (Hexadécimal ou) et qui est assemblé en langage machine Ci-dessous :

Exemple :

En langage mnémonique (ou appeler langage assembleur) l'écriture d'un programme se fait en Hexadécimal or un microcontrôleur exécute que le langage machine, voici ci-dessous un exemple :

L'instruction BCF (Bit Clear F) est en langage mnémonique et en Hexadécimal ça nous donne : 4BF

Avec le logiciel MPLAB qui va assembler ce langage en langage machine (en Binaire), le résultat de cette instruction en binaire : 4BF ⟶ 0100 1011 1111

HEX	Mnemonic		Description	Fucntion
1EF	ADDWF	F,d	Add W and F	Wreg+F ► Dest
040	CLRW		Clear W	0 ► Wreg
0EF	DECF	F,d	Decrement F	F-1 ► Dest
2EF	DECFSZ	F,d	Decrement F, Skip if 0	F-1►Dest,skip if 0
2AF	INCF	F,d	Increment F	F+1 ►Dest

Tableau 3.1 : exemple d'instruction en langage mnémonique

1.7. Conclusion :

Par rapport à des systèmes électroniques à base de microprocesseurs et autres composants séparés, les microcontrôleurs permettent de diminuer la taille, la consommation électrique et le coût des produits. Ils ont ainsi permis de démocratiser l'utilisation de l'informatique dans un grand nombre de produits et de procédés.

Chapitre 4 :

Outils de conception

1. OUTILS DE CONCEPTION

1.1. SolidWorks

Le logiciel de CAO (Conception Assisté par Ordinateur) SolidWorks® est une application de conception mécanique 3D paramétrique qui permet aux concepteurs d'esquisser rapidement des idées, d'expérimenter des fonctions et des cotes afin de produire des modèles et des mises en plan précises.

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.

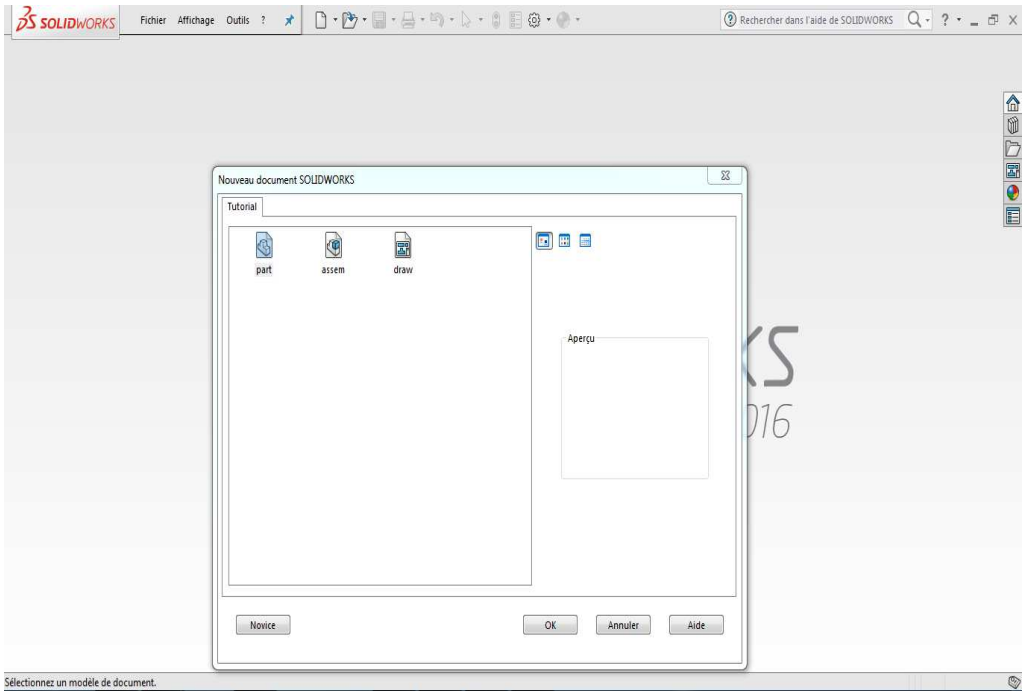


Figure 4.1: Aperçu du logiciel Solidworks avec ces différents fonctions

1.2. Méthode de conception :

Avant de procéder réellement à la conception du modèle, il est utile de planifier sa méthode de création. Une fois les besoins identifiés et les concepts appropriés élaborés, vous pouvez développer le modèle :

Esquisses : Créez les esquisses et décidez du mode de cotation et des emplacements d'application des relations.

Fonctions : Sélectionnez les fonctions appropriées, comme les extrusions et les congés, déterminez les meilleures fonctions à appliquer et l'ordre de leur application.

Assemblages: Sélectionnez les composants à contraindre et les types de contraintes à appliquer.

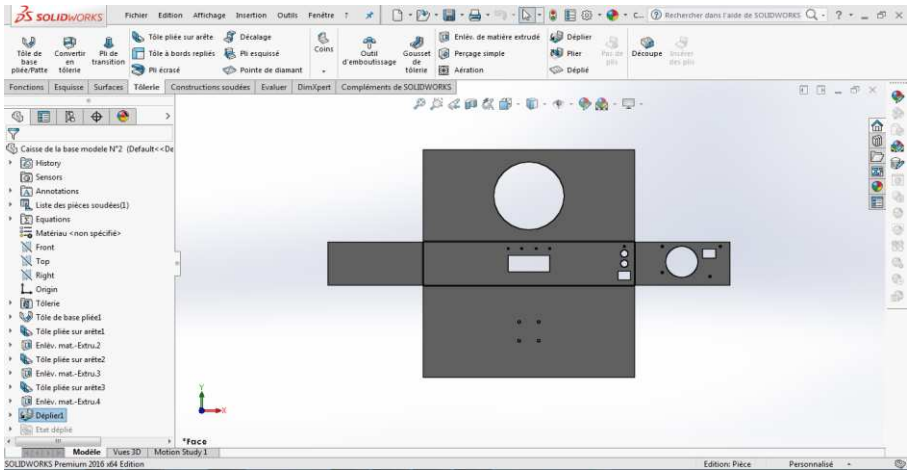


Figure 4.2 : Aperçu du logiciel Solidworks vue 2D (Esquisse)

1.3. Fonctionnement du logiciel SolidWorks :

Les pièces constituent les éléments de base du logiciel SOLIDWORKS. Les assemblages contiennent des pièces ou d'autres assemblages, appelés des sous-assemblages.

Un modèle SOLIDWORKS est constitué de géométrie 3D qui définit ses arêtes, faces et surfaces. Le logiciel SOLIDWORKS vous permet de concevoir rapidement des modèles précis. Les modèles SOLIDWORKS sont :

- ✓ Basés sur la modélisation 3D.
- ✓ Basés sur les composants.

1.4. Modélisation 3D :

SOLIDWORKS adopte l'approche de modélisation 3D. Lorsque vous concevez une pièce, vous créez un modèle 3D, de l'esquisse initiale au résultat final. A partir de ce modèle, vous pouvez créer des mises en plan 2D ou contraindre des composants constitués de pièces ou de sous-assemblages afin de créer des assemblages 3D. Vous pouvez aussi créer des mises en plan 2D d'assemblages 3D.

Un modèle conçu à l'aide de SOLIDWORKS peut être visualisé dans ses trois dimensions, c'est-à-dire dans son aspect final après fabrication.

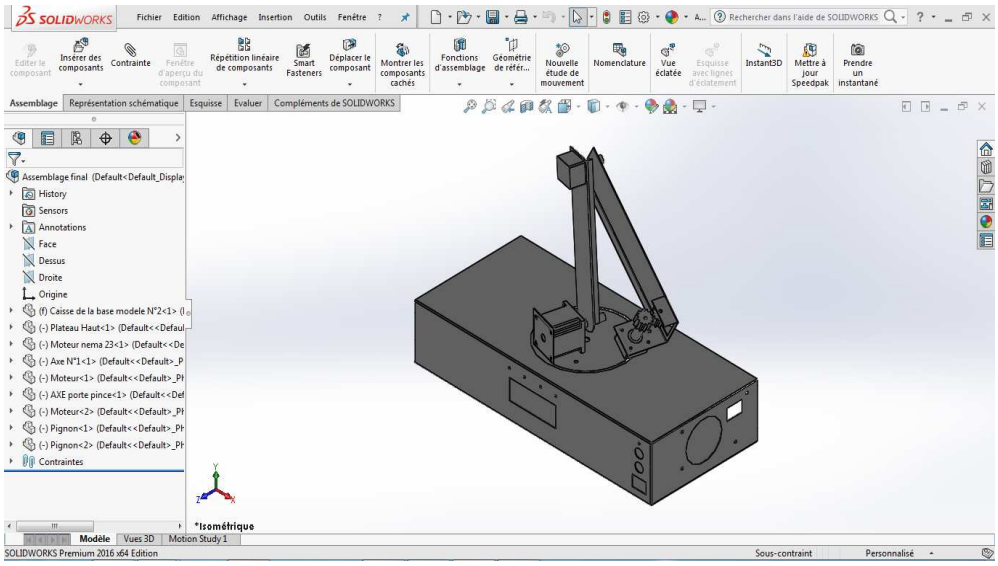


Figure 4.3 : Aperçu du logiciel SolidwWorks avec un assemblage

1.5. Logiciels :

1. EAGLE (Easily Applicable GraphicalLayout Editor) est un logiciel de conception assistée par ordinateur de circuits imprimés.

Il comprend un éditeur de schémas, un logiciel de routage de circuit imprimé avec une fonction d'auto routage, et un éditeur de bibliothèques. Le logiciel est fourni avec une série de bibliothèques de composants de base. C'est un logiciel multiplateforme.

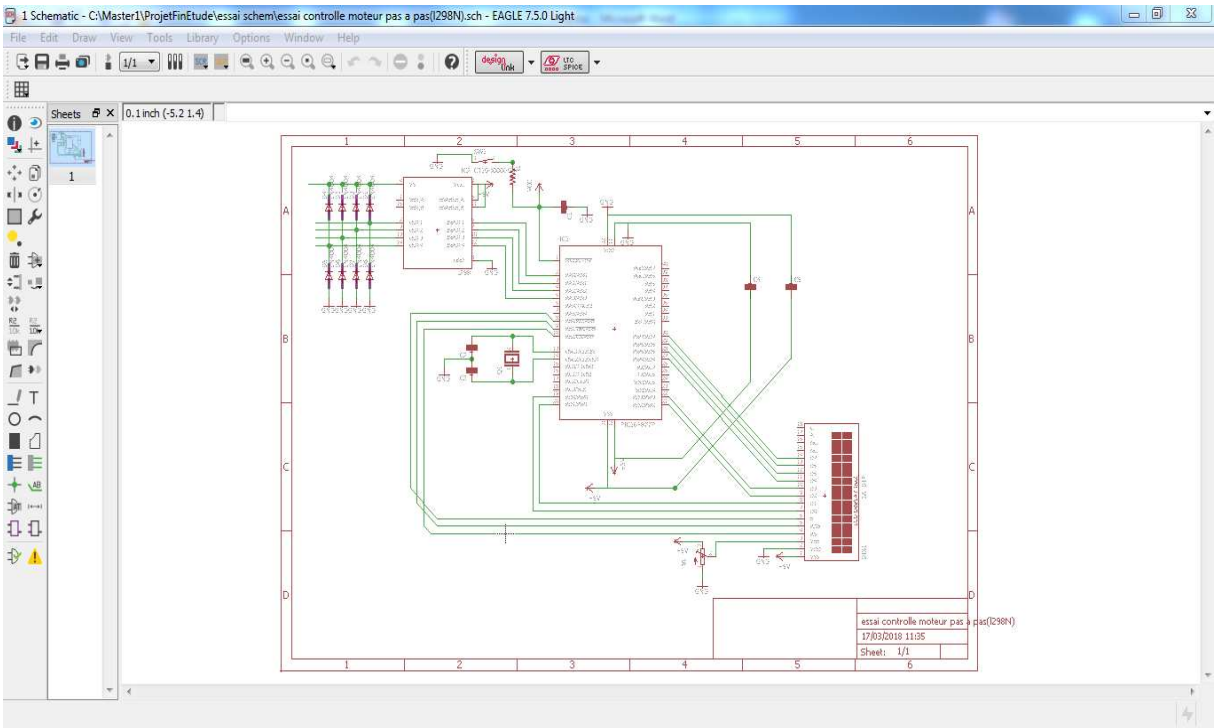


Figure 4.4:Aperçu du logiciel Eagle

2. Visual Basic (VB)

Est un langage de programmation événementielle de troisième génération ainsi qu'un environnement de développement intégré, créé par Microsoft pour son modèle de programmation COM[1]. Visual Basic est directement dérivé du BASIC et permet le développement rapide d'applications, la création d'interfaces utilisateur graphiques, l'accès aux bases de données en utilisant les technologies DAO, ADO et RDO, ainsi que la création de contrôles ou objets ActiveX.

Visual Basic a été conçu pour être facile à apprendre et à utiliser. Le langage permet de créer des applications graphiques de façon simple, mais également de créer des applications véritablement complexes. Programmer en VB est un mélange de plusieurs tâches, comme disposer visuellement les composants et contrôles sur les formulaires, définir les propriétés et les actions associées à ces composants, et enfin ajouter du code pour ajouter des fonctionnalités.

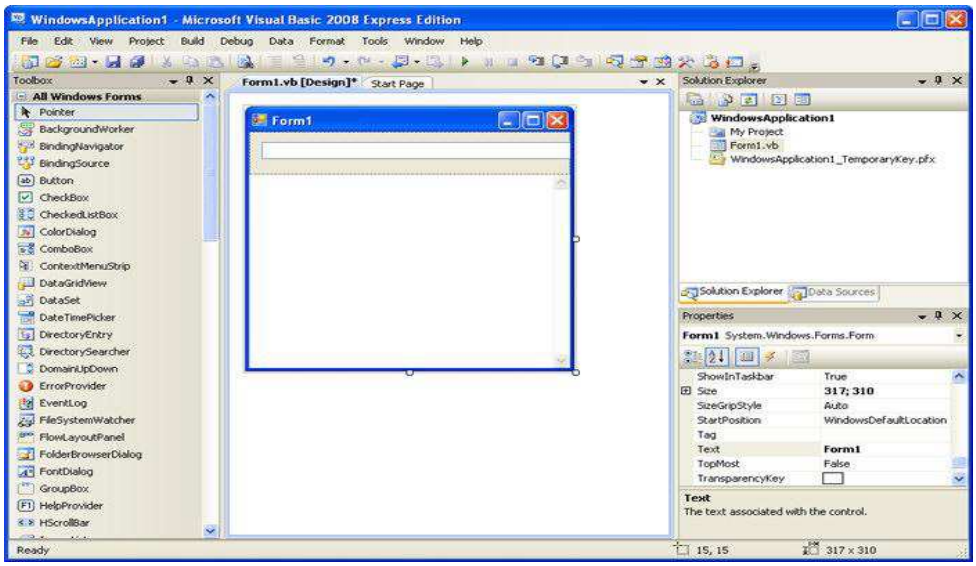


Figure 4.5: Aperçu du logiciel Visual Basic

3. PICkit :

PICkit est une famille de programmeurs pour microcontrôleur PIC de MicrochipTechnology. Ils permettent de programmer les microcontrôleurs et de déboguer les programmes in situ, ainsi que de programmer certaines mémoires EEPROM. Certains modèles proposent également des fonctions d'analyseur logique et de terminal série.

Nous avons trois générations :

- PICkit 1
- PICkit 2 (c'est la version utilisée)
- PICkit 3

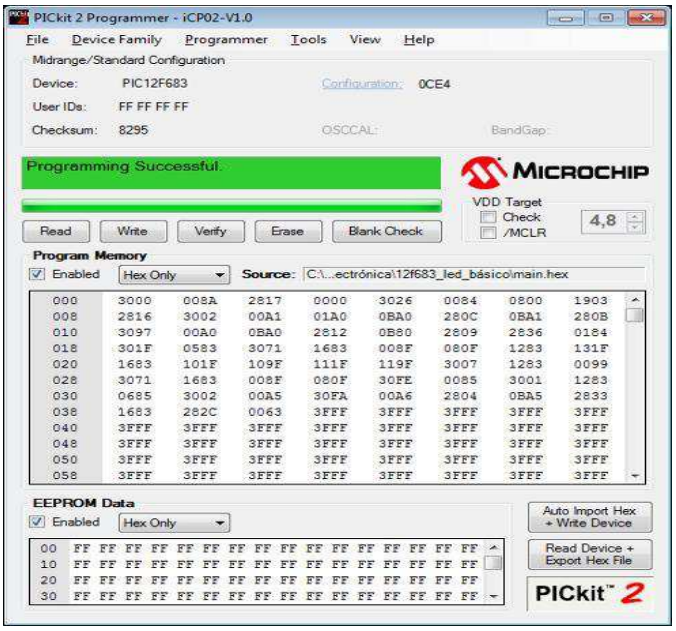


Figure 4.6: Aperçu du logiciel PICkit 2

4. **Ucancam :** C'est un outil qui permet de convertir les fichiers importés par n'importe quel autre outil de conception assisté par ordinateur comme dans notre cas Solidworks en fichier NC i.e. en G code pour la machine découpe plasma.

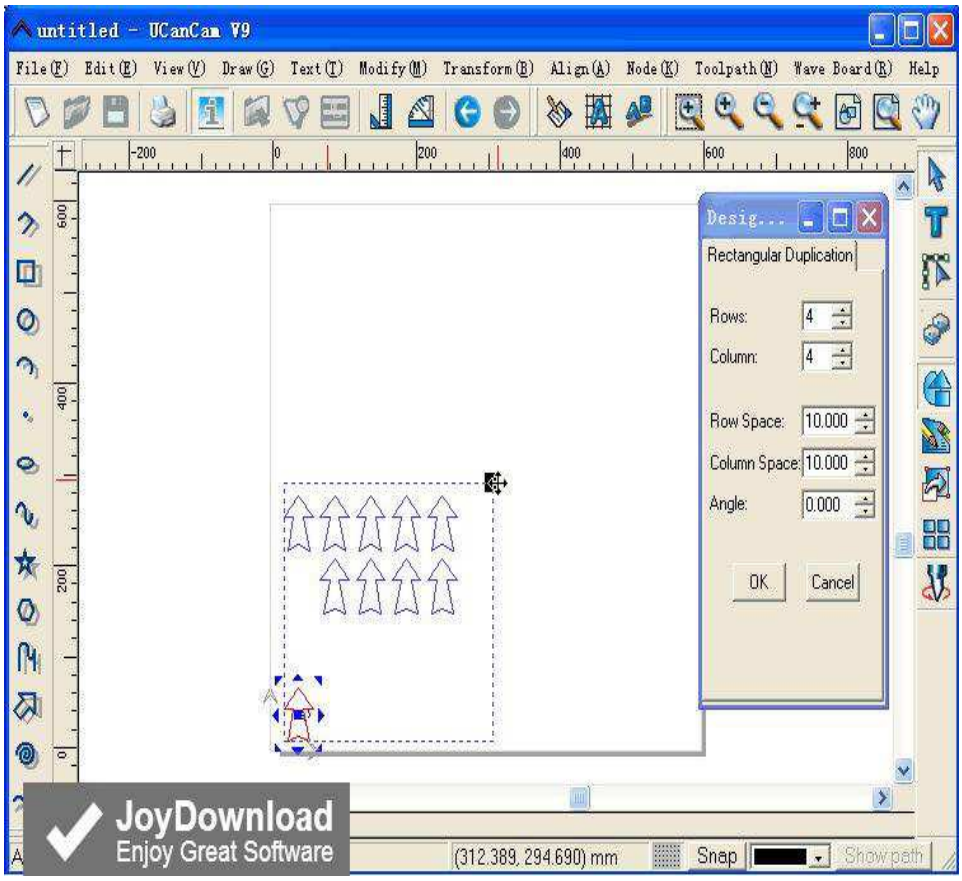


Figure 4.7:Aperçu du logiciel Ucancam

1.6. Conclusion

Nous venons de voir dans ce chapitre les différents outils (logiciel) utilisés pour réaliser ce bras manipulateur, chaque outil utilisé est essentiel et a une fonction bien définie.

Pour n'importe quelle conception dans le monde de l'industrie, il faudra passer par tous ces outils cités au paravent parce qu'ils sont lié l'un a l'autre pour arriver à un état réel final.

Chapitre 5 :

Conception du bras manipulateur et montage

1. CONCEPTION DU BRAS MANIPULATEUR ET MONTAGE

1.1. Introduction

Dans les chapitres précédents nous avons cite les différents partie de la base de la robotique pour réaliser ce bras manipulateur, Dans ce quatrième chapitre on va présenter la structure du bras manipulateur et les différentes étapes pour réaliser ce bras manipulateur.

1.2. Structure du bras manipulateur :

Notre bras manipulateur est un robot automatisé c'est-à-dire il fonctionne sans l'intervention de l'utilisateur (humain) ça permet sert de diminuer la main d'œuvre au sein d'une usine quelconque mais aussi de préserver l'humain à effectuer des taches a risque comme dans des usines de déchet toxique.

Il est composé d'une base qui englobe a l'intérieure : une alimentation, les 2 circuits imprimé, le premier moteur qui permet la rotation de la base. Et de deux axes qui permettent d'avoir une portée de (0.6m).

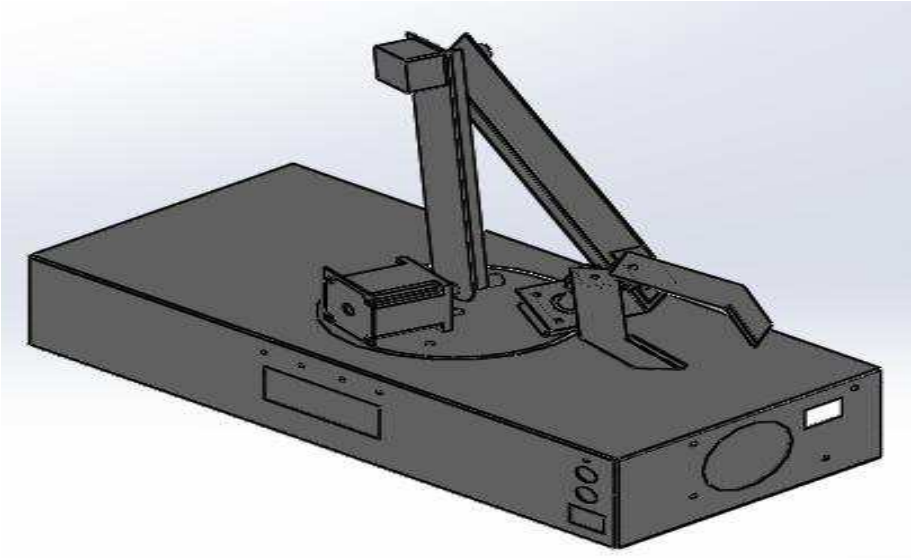


Figure 5.1 : Aperçu du bras manipulateur vue 3D Solidworks

Caisse de la base du bras manipulateur :

La caisse a était conçue par du fer doux d'une épaisseur de 0.9mm (Tôle 9/10), d'une longueur de 510mm, largeur de 230mm et une hauteur de 110mm.

Cette caisse a était modéliser pour pouvoir englober une alimentation d'un ordinateur, 2 circuit imprimée et l'élément de rotation de la base le moteur plus un roulement de 110mm de diamètre et le plateau haut qui est fixé au roulement.

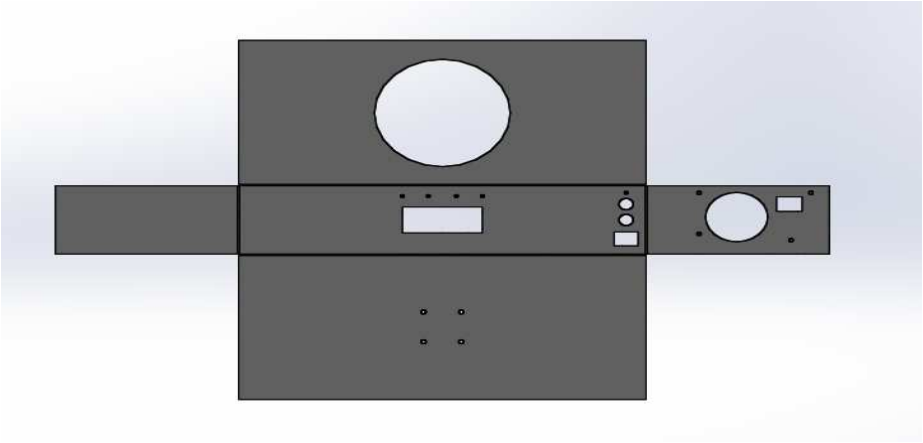


Figure 5.4: Aperçu de la caisse en état déplié

Cette figure nous montre le schéma effectuer sur le l'outil SolidWorks, pour avoir la pièce final donc réel il faut passer par plusieurs étapes :

- Esquisser la pièce voulue sur l'outil SolidWorks.
- Importer le fichier requit de la pièce en format DXF
- Importer le fichier de la pièce sur le l'outil Ucancam
- Convertir le fichier en format NC (en G code)
- Découper la pièce sur la machine découpe plasma
- Plier les différentes parties de la pièce pour obtenir la figure ci-dessous (V.6)

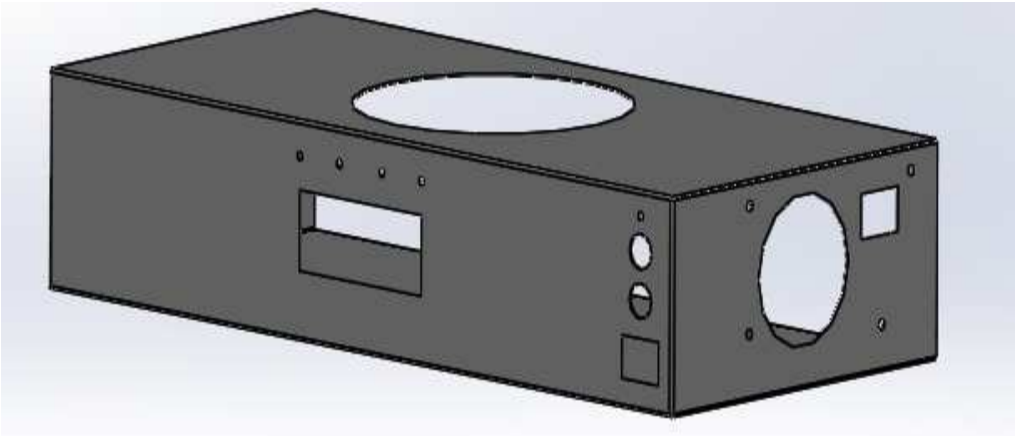


Figure 5.5: Aperçu de la caisse en état plié

1.3. Axes du bras manipulateur :

Les deux axes ont était conçues par le même fer doux et la même procédure de conception cité précédemment.se sont des axes de 300mm de longueur, environ 65mm de largeur comme nous le montre ces figures ci-dessous.

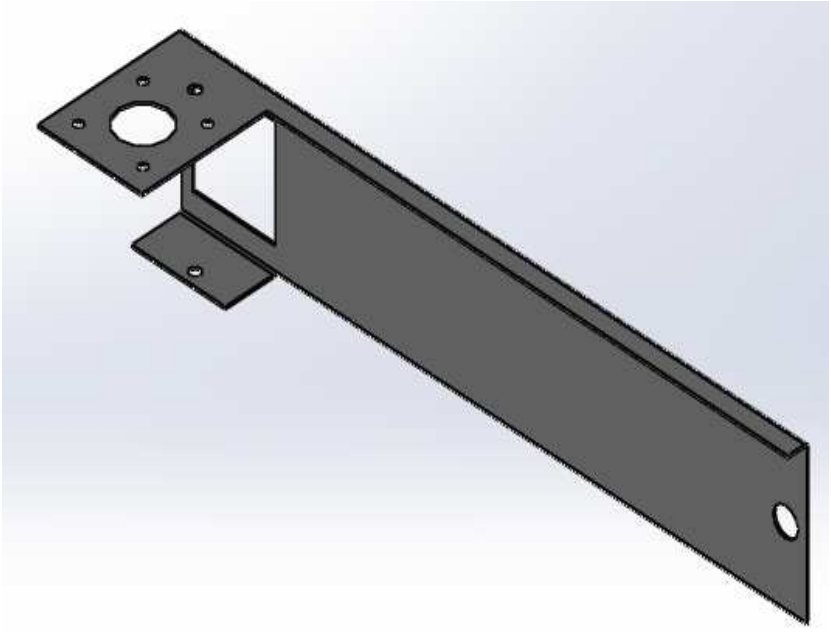


Figure 5.6 : Aperçu de l'axe N°1 porte pince sur Solidworks

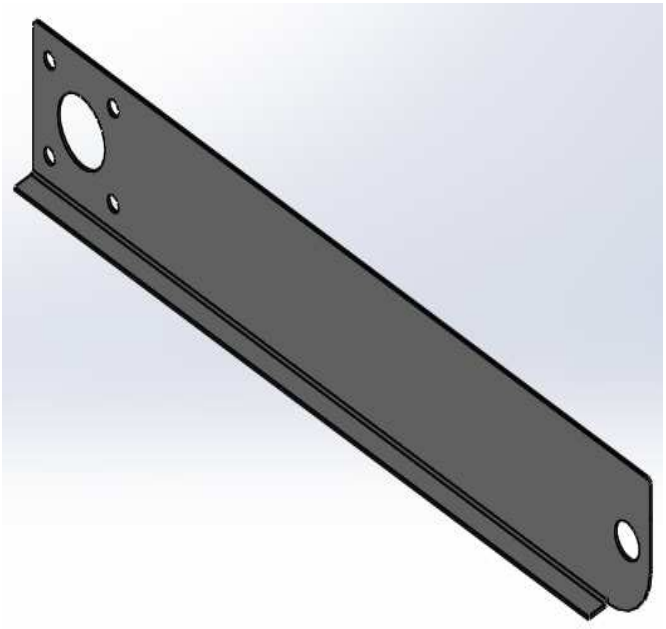


Figure 5.7 : Aperçu de l'axe N°2

1.4. Pince du Bras manipulateur :

La pince a était conçue par une forme non droite avec un pli à l'extrémité pour avoir une meilleure prise quel que soit la forme de l'objet choisis arrondi ou carré, elle serait capable de l'agrippé avec cette forme.

Les deux parties de la pince doivent avoir une rotation inverse ainsi, nous avons choisi d'ajouter des pignons pour permettre d'inverser le sens de rotation.

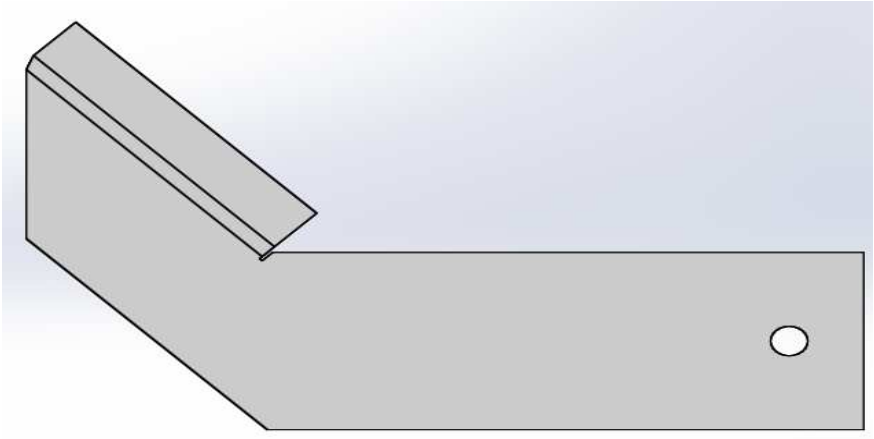


Figure 5.8 : Aperçu de la pince état déplié

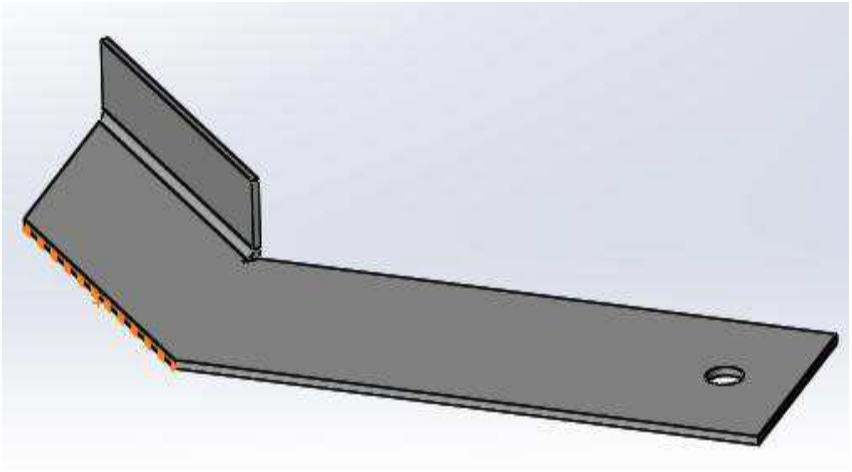


Figure 5.9 : Aperçu de la pince état plié

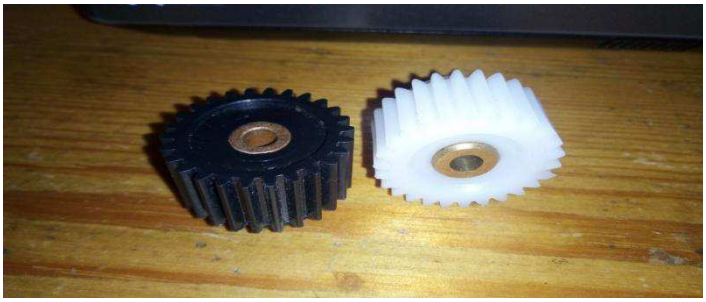


Figure 5.10: Aperçu des pignons choisis

Donc nous avons les deux pince, les deux pignons et l'extrémité de l'axe port pince qui constitue un support pour le moteur et la pince, voici ci-dessous le résultat final de la pince après l'assemblage :

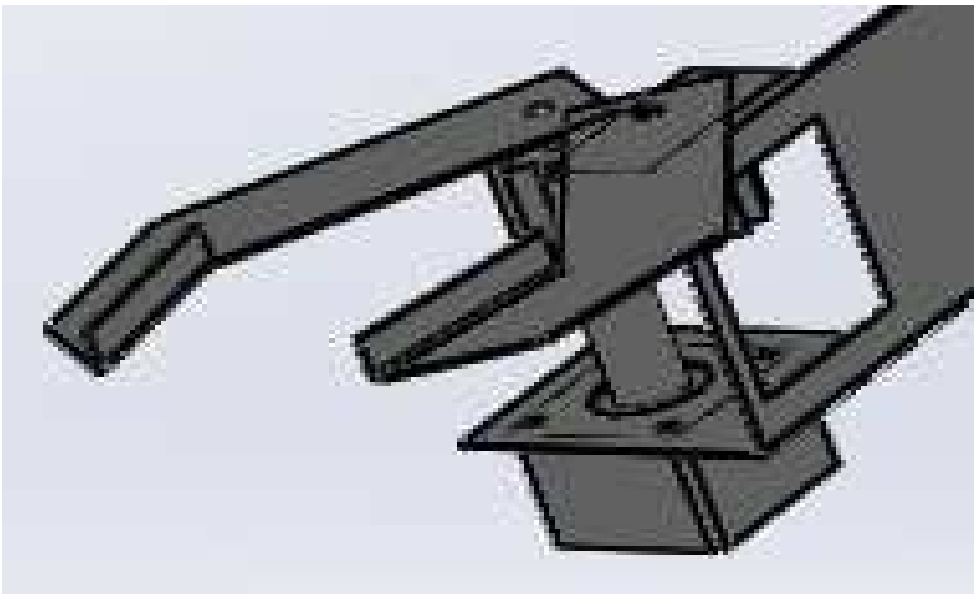


Figure 5.11 : Aperçu de la pince sur l'outil SolidWorks

Figure 5.12 : Aperçu de la pince vue réelle

1.5. Application électronique et commande

Notre bras manipulateur est un bras automatisé, dans notre ca c'est un bras manipulateur porteur, on lui donne une tache quelconque c'est-à-dire on lui envoie juste l'information de l'emplacement de l'objet voulue et l'exécute sans aucune intervention de l'humain, pour cela nous avons choisi le microcontrôleur Pic16F877 pour effectuer cette fonction de commander les quatre moteur pas à pas.

Il fallait pour cela élaborer un circuit électronique capable de subvenir au besoin de contrôler les moteur pas à pas, l'écran LCD et les LED.

Pour arriver à ce circuit imprimé on doit passer par plusieurs étapes :

Tout d'abord choisir les différents composants électroniques adéquats a notre fonction requise comme le microcontrôleur mentionné auparavant ou les composants requis pour une connexion via l'ordinateur. Ensuite faire un schéma électronique de tout ces composants sur l'outil de conception EAGLE, c'est-à-dire de faire tout le brochage du circuit.

Mais ces différents composants mentionnés ont besoin de plusieurs autres composant comme les résistances ou les capacités requise pour ce circuit, donc la 1ère étape est de réaliser plusieurs circuits électroniques pour arriver à un résultat final.

Pour l'alimenter ce circuit on utilise une alimentation (ATX) qui nous donne des différentes sortie : +3V,-3V, +5V,-5V, +12V,-12V. Nous avons plusieurs composants principaux à alimenter comme le microcontrôleur PIC, le MAX 232, le capteur optique et le L298N plus les moteurs pas à pas.

Donc le premier circuit du microcontrôleur pic (16F877A) a besoin d'une capacité de 100uF pour stabiliser le courant continue a une valeur fixe de 5volt, remarquons que nous avons deux broches connectées au VCC(5V), les broches 11 et 32, car une seule broche ne suffit pas à alimenter tous le microcontrôleurs. La résistance de 10Kohm qui est en parallèle avec la capacité, a pour but de diminuer le courant sur l'alimentation du microcontrôleur et le bouton poussoir permet d'avoir un reset, c'est-à-dire de redémarrer le microcontrôleur en cas d'un problème quelconque.

Voici- ci-dessous la figure montrant ce schéma électronique :

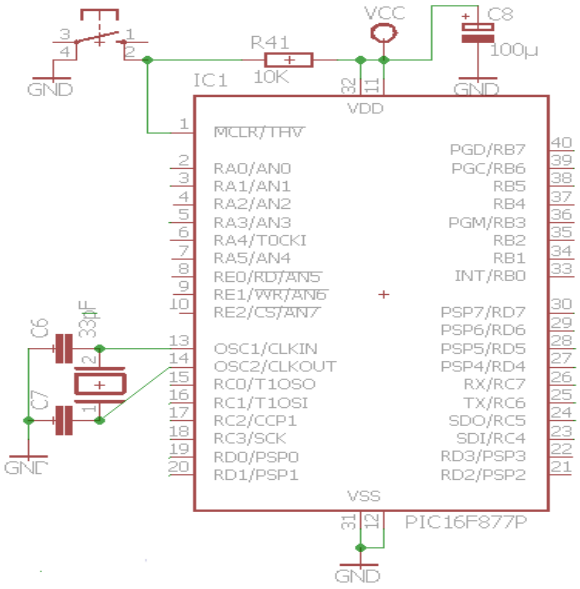


Figure **5.13** : schéma électronique de l'alimentation du microcontrôleur PIC

Pour le deuxième circuit c'est l'alimentation du composant **L298N** et les moteurs pas à pas, ici avons besoin de deux alimentations différentes, la première 1ère c'est du +5volt qui est connecté à la broche 9 du **L298N**, pour alimenter le composant afin qu'ils deviennent fonctionnel et le second 2ème c'est du +12volt pour avoir une augmentation de puissance pour les moteurs pas à pas.

Donc le +12volt alimente la 4ème broche du **L298N** qui est la broche Voltage Supply qui a une tension maximale de 50volt. La capacité de 100nF est mise en parallèle pour avoir une tension de 5volt stable et diminuer les parasites du circuit.

Les 8 diodes utilisées et connectées entre les sorties du **L298N** et aux fils des moteurs pas à pas ont pour fonction de donner un sens aux courant et d'interdire son retour dans le sens inverse et de dissiper la chaleur et la tension restante lors de la coupure de l'alimentation, le bout des ces diodes est connecté au 12volt pour avoir la puissance nécessaire pour ces moteurs.

Donc le deuxième circuit qu'on peut appeler circuit de puissance est représenté dans la figure ci-dessous :

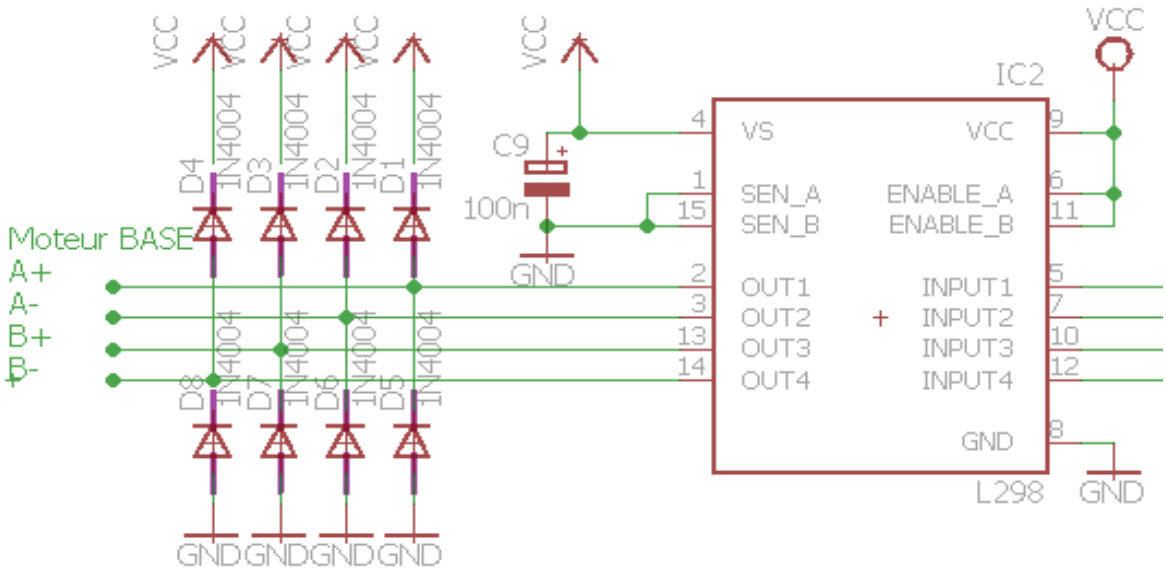


Figure **5.14** : schéma électronique pour l'alimentation du L298N et le moteur pas à pas

Le Troisième circuit est celui du composant **MAX232** qui est nécessaire pour convertir la tension de 5V, 0V en +12V,-12V. Ce convertisseur est utile pour la norme S.I de la connexion ordinateur, or la sortie du microcontrôleur **Rx** (réception) **Tx** (transmission) est de 0v ou 5v et la connexion avec un ordinateur par le port DB9 est de +12v ou -12v.

Donc ce circuit, est un circuit pour convertir les sorties des ports du microcontrôleur :

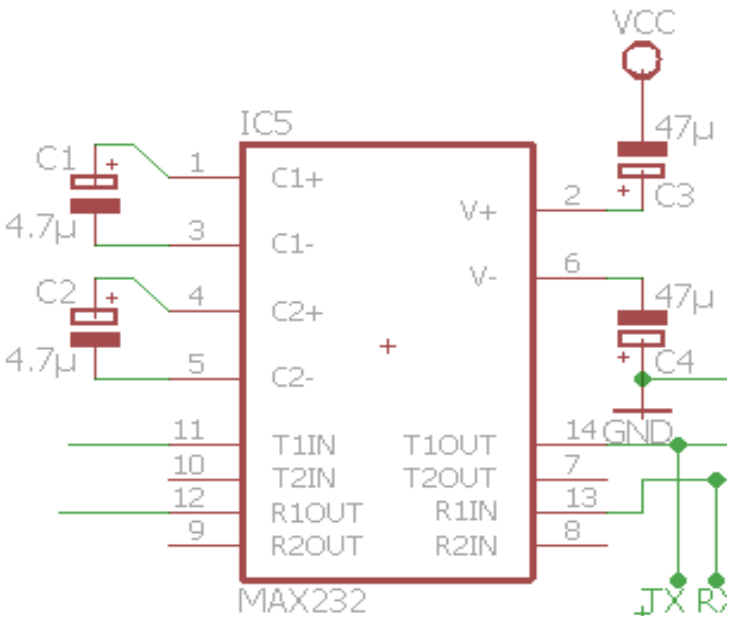


Figure **5.15** : schéma électronique du MAX232

Le 4ème circuit est le circuit qui nous démontre le schéma électronique du capteur optique qui a pour fonction de détecter la position initial du moteur pas à pas, c'est un capteur donc il à besoin d'une alimentation de 5v pour qu'il fonctionne, en plus des résistances de 2,2 KOhm et 120 Ohm qui sont connectées a la masse pour diminuer le passage du courant en parallèle avec l'entrée du port B6 du microcontrôleur qui réceptionne la donnée (5v,Ov) du capteur optique. Lors du passage entre la diode et le transistor nous aurons 0v au niveau du port B6, donc l'interruption est détectée au niveau du microcontrôleur qui arrête la rotation du moteur lors de son initialisation.

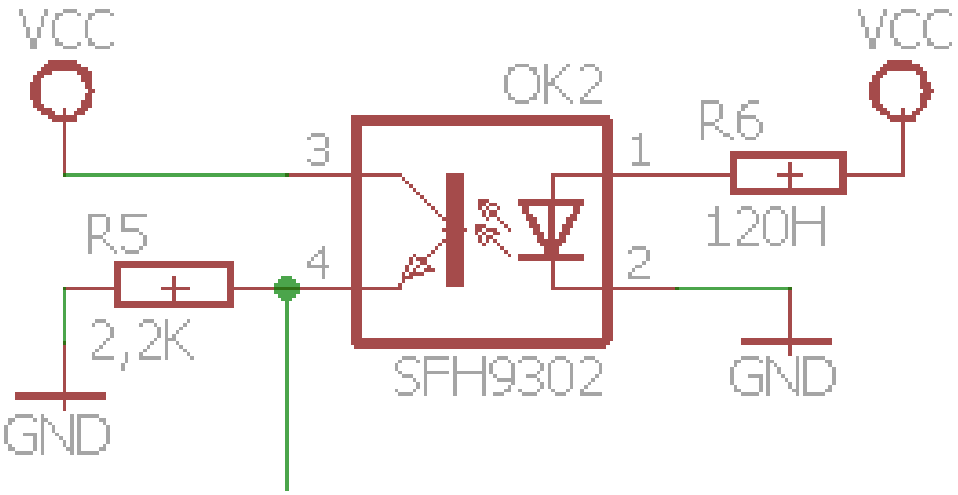


Figure **5.16** : schéma électronique du capteur optique TCST-2202

Après avoir réalisé ces circuits des composants principaux pour la commande de ce bras manipulateur on déduira un schéma électronique qui est représenté ci-dessous :

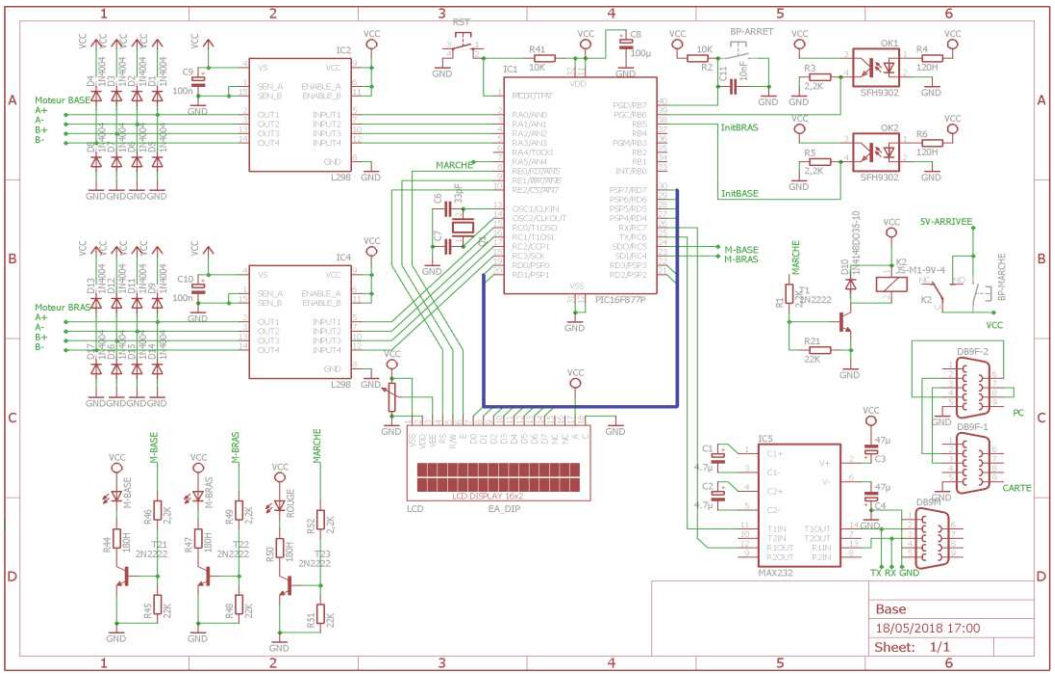


Figure 5.17: Aperçu du schéma électronique sur l'outil EAGLE

1.6. Différents composants de ce schéma électronique :

- ✓ Microcontrôleur PIC16F877
- ✓ Quartz 20 Mhz
- ✓ Le composant Max 232(pour convertir 0 et 5volt en 12 ou -12volt)
- ✓ Adaptateur asynchrone R232 (DB9)
- ✓ Le composant L298
- ✓ Relais 5v
- ✓ Capteur a fourche optique.
- ✓ Les différentes résistances et capacités pour le filtrage électronique.
- ✓ Bouton poussoir et LED.
- ✓ L'écran LCD (20*4).

Notre bras se constitue de deux circuits imprimés identiques :

1er circuit : - Contrôler les deux premiers moteurs celui de la base et de la première rotation.

- ✓ Il gère la connexion entre l'ordinateur et le bras manipulateur.
- ✓ Il commande l'allumage du bras manipulateur.
- ✓ Il contrôle les différents led des deux moteurs et celle de l'allumage.
- ✓ Contrôler l'affichage de l'LCD.
- ✓ intercepter l'information obtenue du capteur optique des moteurs.

2éme circuit : -Contrôler les deux moteurs restants celui de la deuxième rotation et de la pince.

- ✓ Il contrôle les deux LED des deux moteurs.
- ✓ intercepter l'information obtenue du capteur optique des moteurs.

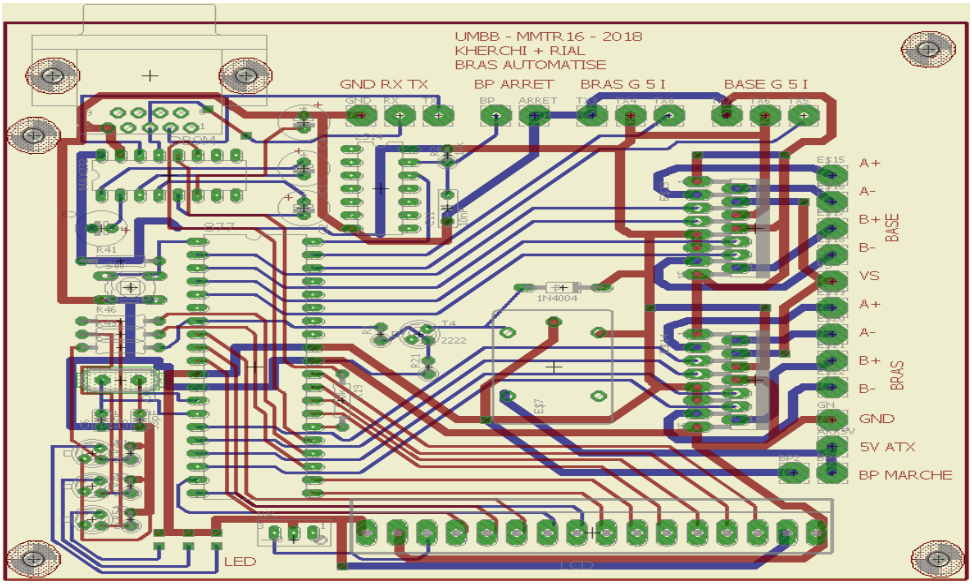


Figure 5.18: Aperçu du circuit imprimé sur l'outil EAGLE

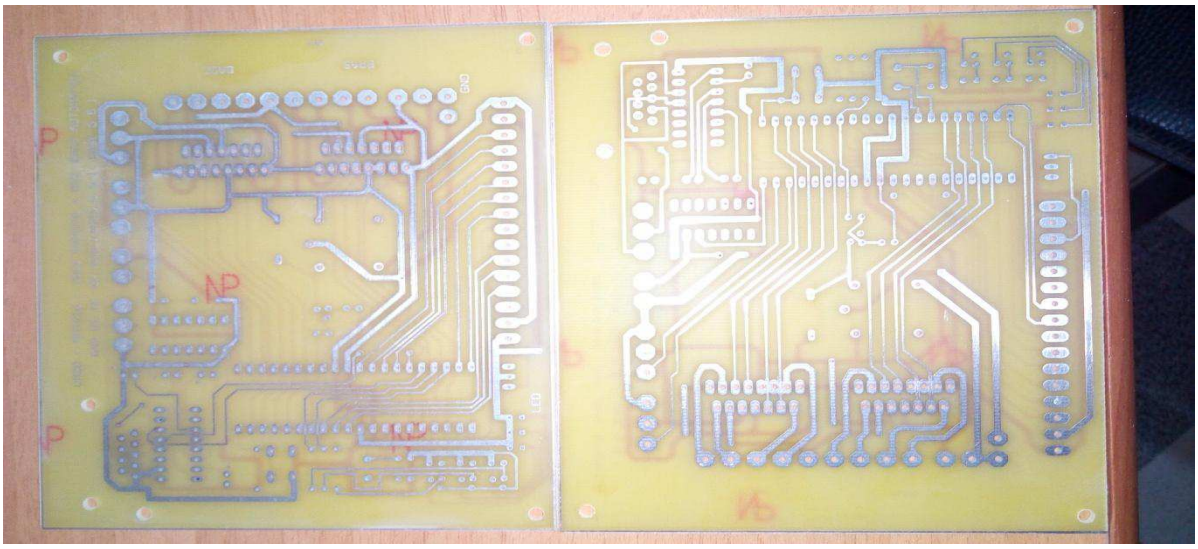


Figure 5.19: Aperçu des deux faces du circuit en état réel

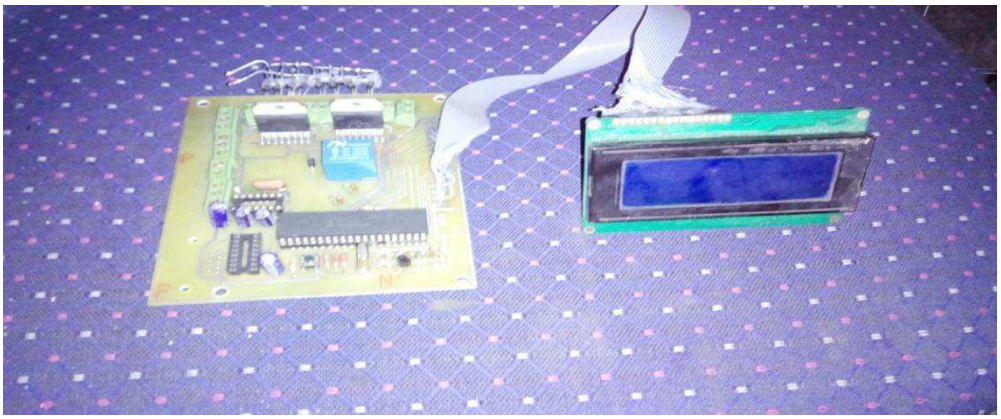
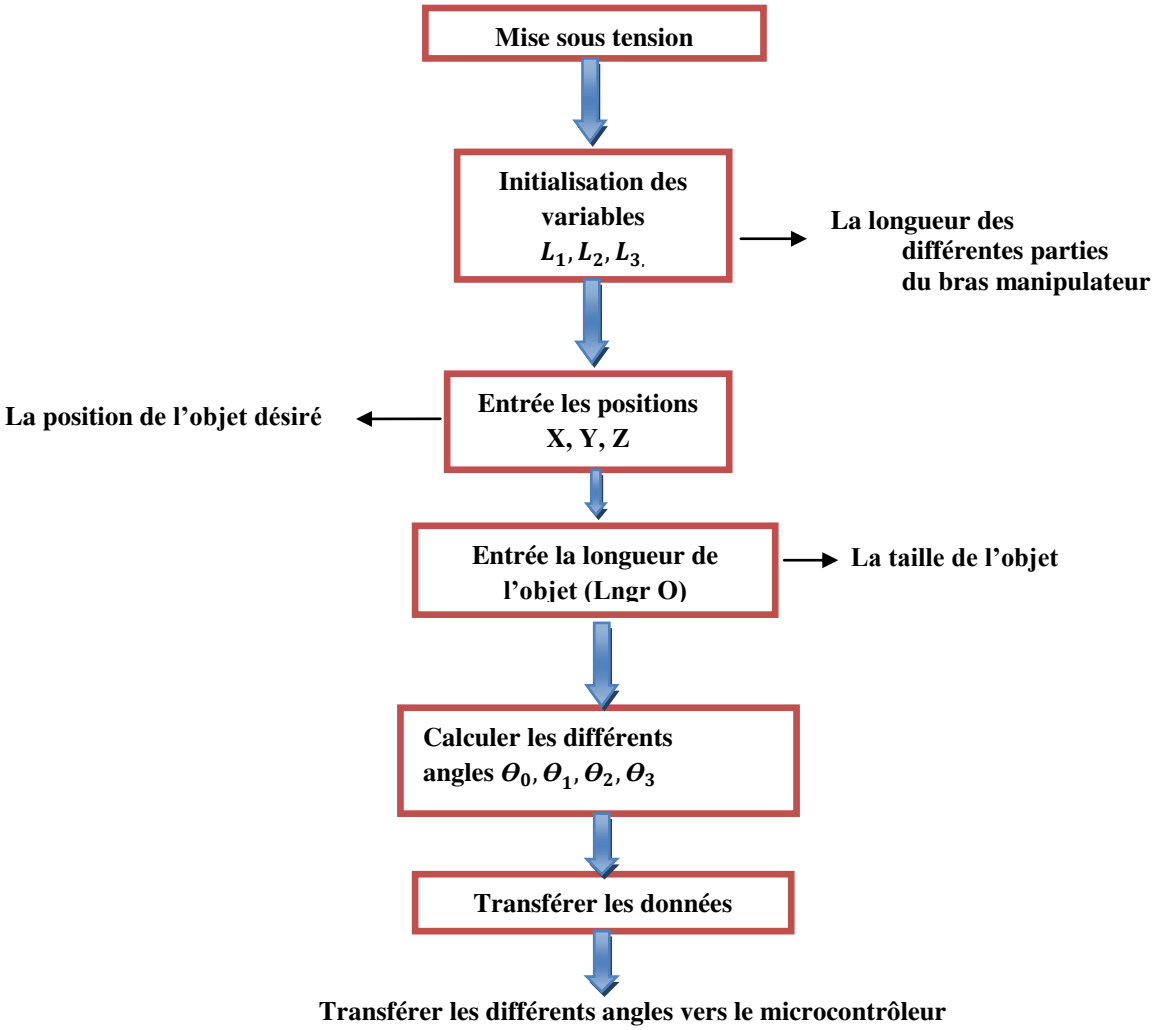


Figure 5.20: Aperçu du circuit après avoir soudé tout les composants électroniques

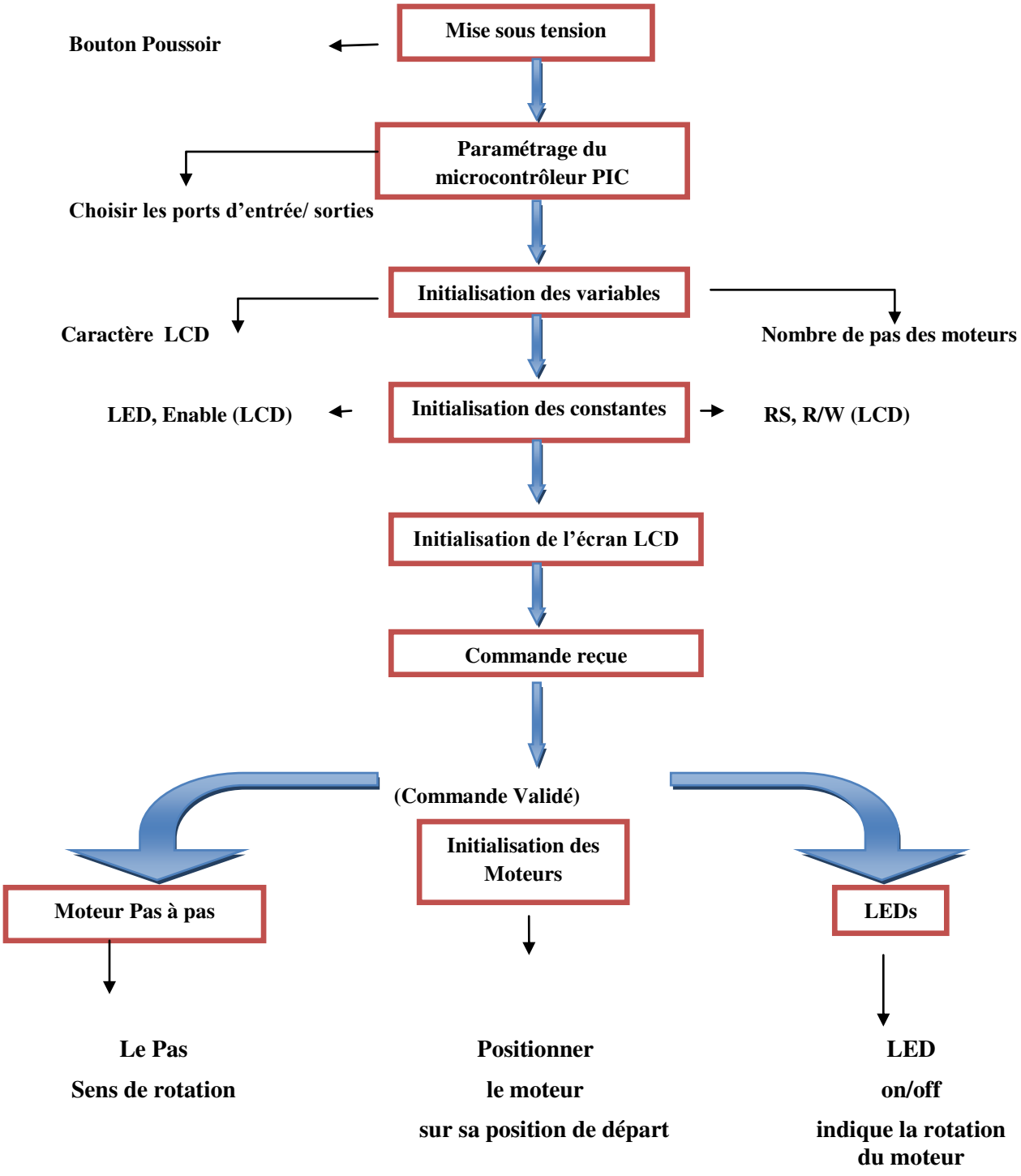
1.7. Organigramme du programme pour la commande du bras manipulateur :

Deux programmes sont conçus pour la manipulation du bras, le premier pour le microcontrôleur et le deuxième pour l'interface sur l'ordinateur qui va permettre de communiquer avec les différents moteurs pas à pas utilisés.

Voici ci-dessous un organigramme du programme écrit sur l'outil Visual Basic :



Le deuxième organigramme du programme écrit sur l'outil Mplab pour les deux microcontrôleurs pour la commande des moteurs, LCD et les LEDs écrit sur l'outil de Mplab en langage d'assembleur :



1.8. Conclusion

Dans ce chapitre nous avons présenté la réalisation du bras manipulateur qui a nécessité deux phases de réalisation, une phase mécanique et une phase électronique.
La partie mécanique a été réalisée en atelier en rassemblant les différents outils cité au paravent et la partie électronique a été réalisée grâce aux connaissances électroniques et informatiques acquises durant le cursus universitaire.

Figure **5.21**: Le bras manipulateur dans l'état final

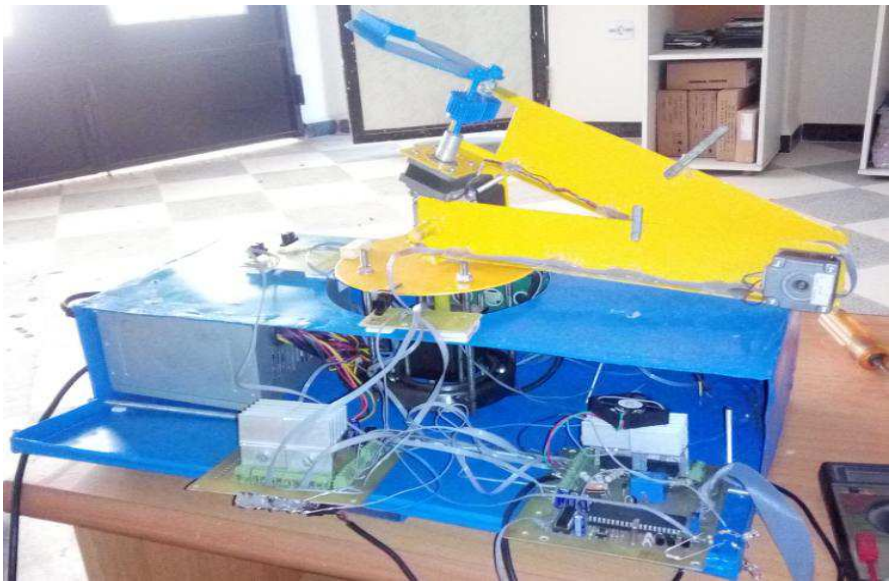


Figure **5.22**: Le bras manipulateur dans l'état final

Conclusion Générale

CONCLUSION GENERALE

Ce mémoire que nous avons conclu est une étude et réalisation d'un bras manipulateur, cette conception a été mise en œuvre grâce à la formation reçue durant notre cursus universitaire en Génie Mécanique (Mécatronique).

En utilisant ces différentes connaissances en Mécatronique (Mécanique, Electronique, Programmation, Informatique), nous avons pu élaborer une conception de ce bras manipulateur à trois degrés de liberté (RRR).

Dans ce projet nous avons présenté une méthode de réalisation en utilisant une commande avec le microcontrôleur PIC et des actionneurs tels que les moteurs pas à pas, tous ces éléments sontcontrôlésgrâceà une interface sur un ordinateur géré par l'utilisateur (humain) loin du champ de travail.

Le but de ce projet est de réaliser un bras manipulateur qui se déplace avec précision pour faciliter et automatiser le travail au sein d'une industrie, aussi depréserver l'humain contre les risque du travail.

Durant la réalisation de ce projet, nous avonsfait face à plusieurs obstacles tels que manque de puissance dans les moteurs qui a engendré des problèmes lors du soulèvement de la structure de ce robot ou dans la conception de la structure du bras manipulateur et sa pince. Mais malgré cela nous sommes arrivés à ce résultat final, un bras autonome qui est capable de se déplacer avec précision d'un point à un autre dans l'espace.

Perspective

Ce projet de fin d'étude est une réalisation d'une maquette d'un bras manipulateur c'est-à-dire à une échelle réduite, le but à atteindre est d'avoir une précision lors de son déplacement, or dans l'industrie il nous faut des robots plus grand donc une plus grande structure solide et de plus gros actionneurs. Les moteurs utilisés, dans ce projet, sont des moteurs Nema 17,23 qui sont utilisés pour les imprimantes 3D par exemple, ils possèdent un couple très petit même en Mili Newton.

Un robot destiné pour une industriedoit être capable de supporter des éléments avec un poids important, donc le principe pour cette réalisation reste le même. Notre robot peut être amélioré en ajoutant un capteur de pression et un détecteur d'objet, il serait plus autonome en détectant l'objet et pourrait fermer la pince par rapport à la taille de l'objet.

Le principe de la réalisation industrielle est le même, la même méthode utilisée, soit dans la commande et/ou la programmation, la différence réside dans la structure et le circuit de puissance adéquat pour alimenter ses actionneurs.

Références Bibliographiques

REFERENCES BIBLIOGRAPHIQUES

[1] Ait Dahmane Kahina et Ait Ziane Meziane, 2015 « **Conception et Réalisation d'un Bras Manipulateur Commandé par API »,** Thèse Master, Université de KHEMIS MILIANA.

[2] Al Teef Noor ali, Jghef,YousefSofyanHasanHeddeh,Mohamed Mohamed Zaki Hani Ibrahim, Mohamed Ali Ismail Abdul Rahman, 2015 « **Design and the mechanism of controlling a robotic arm,** memoire ….

[3] Arian Faravar, 2014 **Design,Implementation and control of a robotic arm using PIC16F877A Microcontroller**

[4] MoussaouiAmira, 2017 «**Conception et réalisation d'un bras manipulateur commandé par l'arduinomega 2560** », Thèse Master, Université M'hamedBougara BOUMERDES.

[5] RichaBraham et BouyekhfMouhamed ,2017 « **Etude et réalisation d'un bras robot à 2DDL** », ThéseMaster,Université Djilali Bounaama KHEMIS MILIANA.

[6] Saadi Ramzy et SalhiNassereddine, 2010 « **Réalisation de carte a microcontrôleur pour le contrôle de bras manipulateur via un Pc Mémoire »,** Thèse Master, Université Mohamed Khider BISKRA.

[7] V.Tourtchine, 2009 « **Microcontrôleur de la famille PIC (support de cours et prise en main du logiciel Mplab) »,** Université M'hamedBougara BOUMERDES.

[8] https://www.apprendre-en-ligne.net/crypto/passecret/ohm.html

[9] https://www.rs-online.com/designspark/arduino-stepper-motor-control-fr

[10] http://www.f-legrand.fr/scidoc/docimg/sciphys/arduino/paspas/paspas.html

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

[12] http://eskimon.fr/290-arduino-603-petits-pas-le-moteur-pas-pas

[13] http://etronics.free.fr/dossiers/num/num50/mpap.htm

[14] https://www.youtube.com/watch?v=e1PAHEeJZTA

[15] http://www.jsfrance.com/fr/capteurs-de-position-aimants/3920-capteur-optique-a fourche-tcst-1103-neuf-2100000047314.html

[16] https://fr.wikiversity.org/wiki/Capteur/Capteurs_optiques

[17] https://fr.wikipedia.org/wiki/Opto-%C3%A9lectronique

[18] https://fr.wikipedia.org/wiki/Capteurhttps://fr.wikipedia.org/wiki/Capteur

[19] https://fr.slideshare.net/guest1e7b02/microcontroleur-pic16-f84

[20] https://fr.wikipedia.org/wiki/Assembleur

[21] https://tronixmaker.com/fr/electronique/64-moteur-pas-a-pas-nema-23-518.html

[22] http://gte.univ-littoral.fr/sections/documents-pdagogiques/chapitre-8-mesure/downloadFile/file/Les_capteurs.pdf?nocache=1289041293.82

[23] https://fr.wikipedia.org/wiki/Capteur

[24] http://sti.tice.ac-orleans-tours.fr/spip2/IMG/pdf/Les_capteurs.pdf

[25] http://www.technologuepro.com/cours-capteurs-actionneurs-instrumentation-industrielle/ch12-les-differents-types-de-capteurs.pdf

[26] https://www.digchip.com/datasheets/parts/datasheet/513/TCST-2202-pdf.php

[27] http://jerome.hennecart.free.fr/APL_photoT.htm

[28] https://www.jeulin.fr/media/pim/assets/DocumentsPDF/std.lang.all/1-/fr/Notice-283171-FR.pdf

[29] http://dspace.univ-tlemcen.dz/bitstream/112/5074/3/Chapitre%202.pdf

[30] http://kiattisakbp1.blogspot.com/2015/07/45-fibo.html

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

[32] https://fr.wikipedia.org/wiki/Presse_plieuse

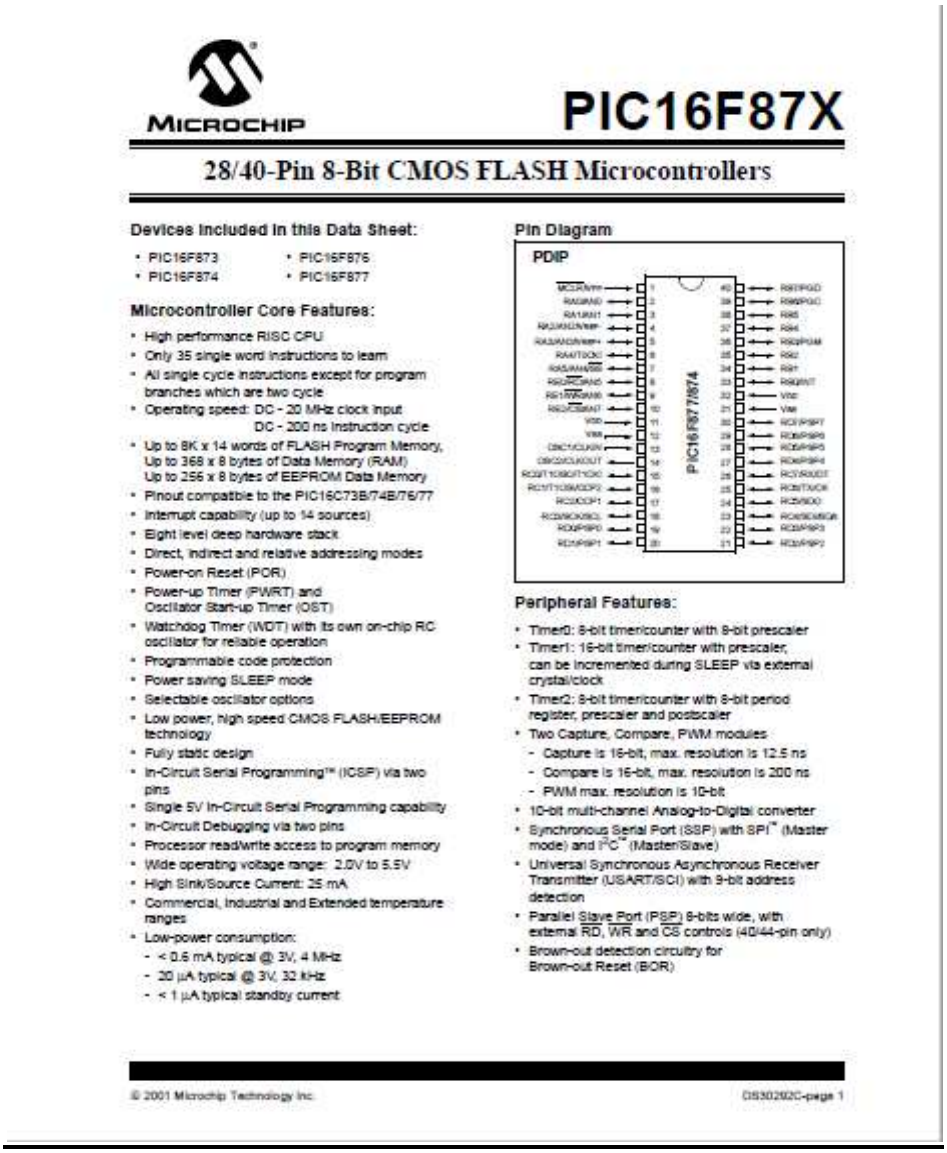
[33] https://fr.wikipedia.org/wiki/Robot

[34] https://sites.google.com/site/gpa141ginfberm/qu-est-ce-qu-un-robot-industriel
[35] https://fr.wikipedia.org/wiki/Capteur
[36] https://fr.wikipedia.org/wiki/Actionneur
[37] Theory of Automatic Robot Assembly and Programming by By B.O. Nnaji
[38] https://fr.wikipedia.org/wiki/Capteur
[39] http://dondon.vvv.enseirb-matmeca.fr/RSIcapteur/a.pdf

Annexes

ANNEXES

Annexe 1 : Microcontrôleur PIC 16F877A



MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

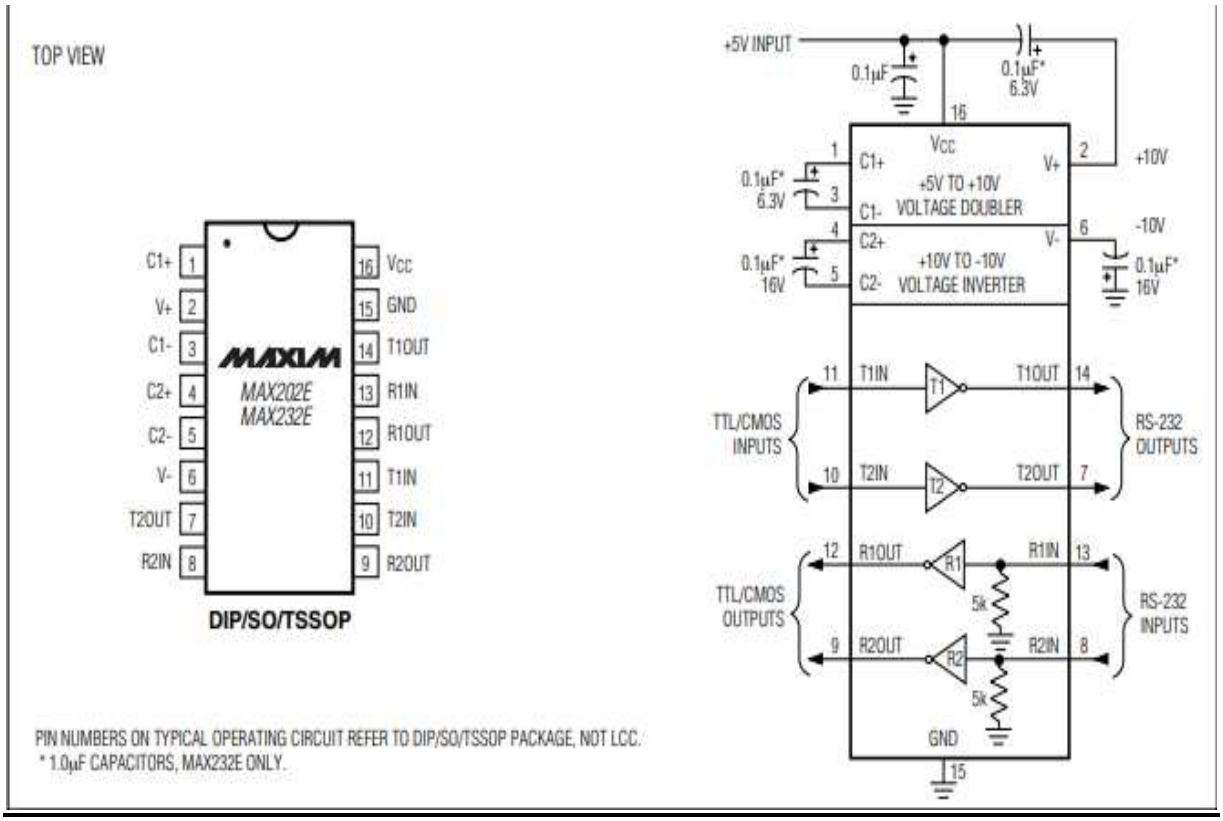
PDIP

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

Annexe 2-Max 232



MAX202E/MAX232E

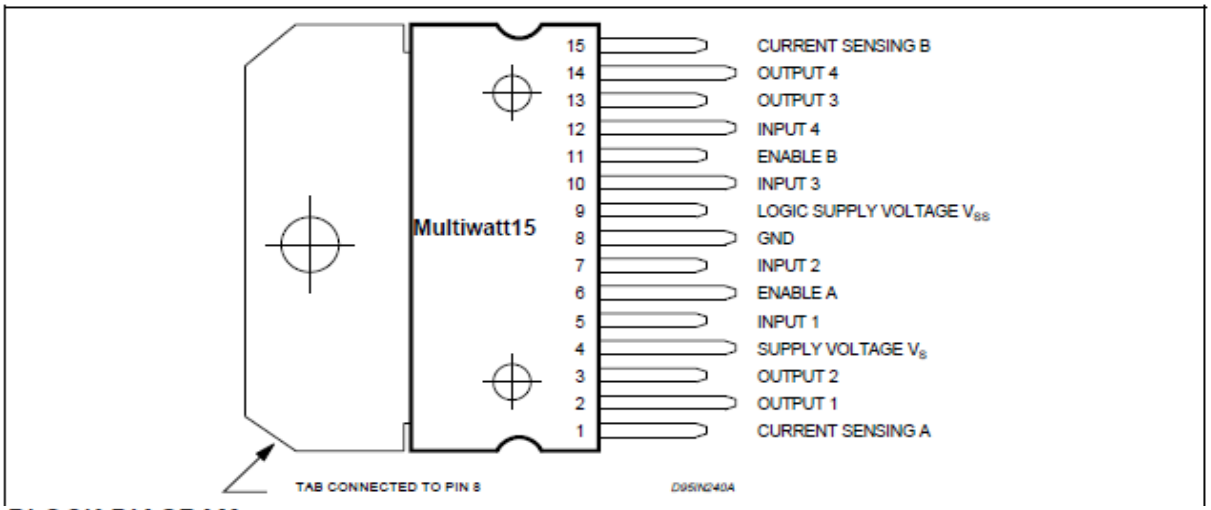
PIN		NAME	FUNCTION
DIP/SO/TSSOP	LCC		
1, 3	2, 4	C1+, C1-	Terminals for positive charge-pump capacitor
2	3	V+	+2VCC voltage generated by the charge pump
4, 5	5, 7	C2+, C2-	Terminals for negative charge-pump capacitor
6	8	V-	-2VCC voltage generated by the charge pump
7, 14	9, 18	T_OUT	RS-232 Driver Outputs
8, 13	10, 17	R_IN	RS-232 Receiver Outputs
9, 12	12, 15	R_OUT	RS-232 Receiver Outputs
10, 11	13, 14	T_IN	RS-232 Driver Inputs
15	19	GND	Ground
16	20	VCC	+4.5V to +5.5V Supply-Voltage Input
—	1, 6, 11, 16	N.C.	No Connect—not internally connected.

Annexe 3-L298N

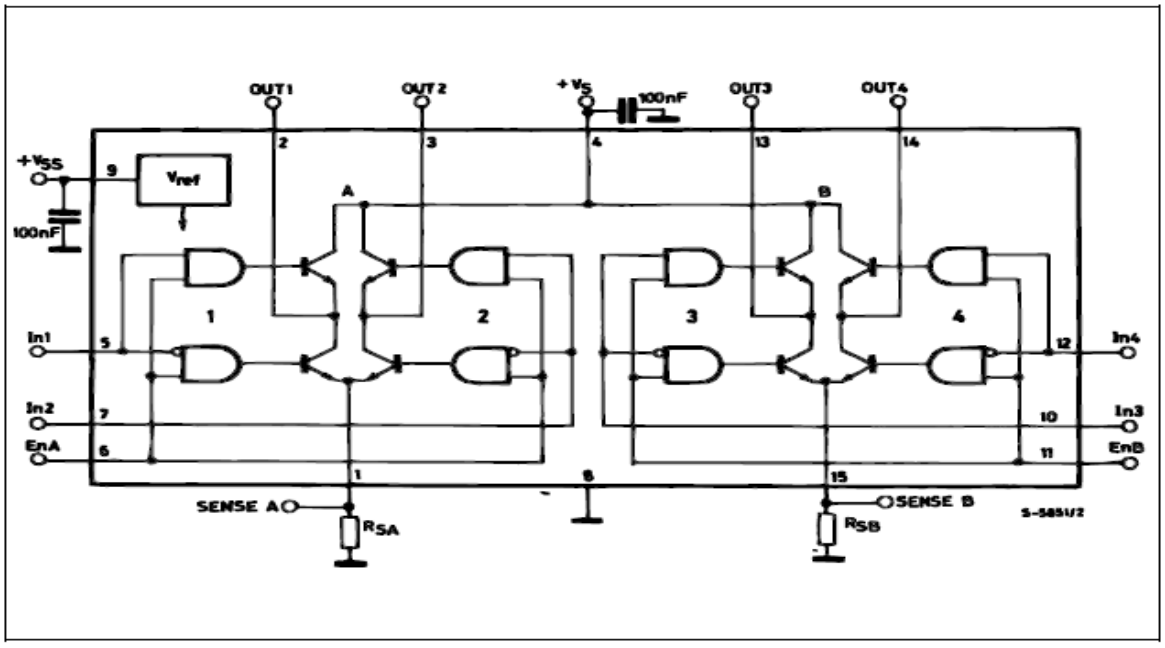
ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

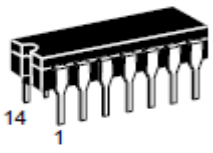
PIN CONNECTIONS (top view)



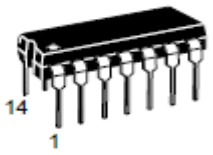
BLOCK DIAGRAM



Annexe 4-74LS14



J SUFFIX
CERAMIC
CASE 632-08



N SUFFIX
PLASTIC
CASE 646-06

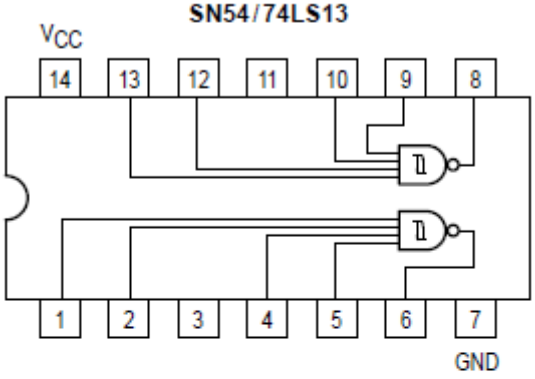
D SUFFIX
SOIC
CASE 751A-02

ORDERING INFORMATION

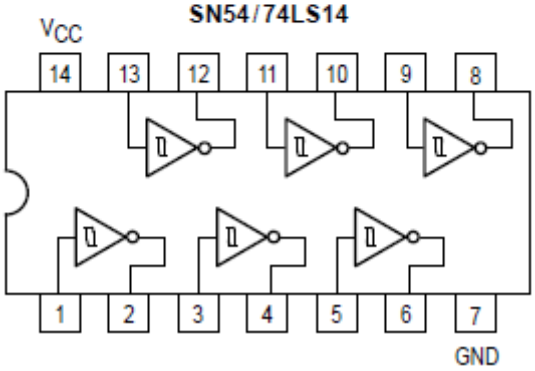
SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

LOGIC AND CONNECTION DIAGRAMS

SN54/74LS13



SN54/74LS14



Annexe 5-Programme du microcontroleur PIC16F877 :

```
;*********************************************************************
;   Ce fichier est le programme de la carte de contrôlre du bras manipulateur*
;*********************************************************************
;   NOM:    Code Carte Controle Proncipal
;   Date:    06/2018
;   Circuit:  Carte Controle Principal
;   Auteur:   Rial. Kherchi
;
;*********************************************************************
;   Fichier requis: P16F877.inc                                 *
;*********************************************************************
      LIST      p=16F877         ; Définition de processeur
      #include <p16F877.inc>       ; fichier include
      radix dec                                   ; on travaille en décimal par défaut

      __CONFIG    _CP_OFF & _DEBUG_OFF & _WRT_ENABLE_OFF & _CPD_OFF
& _LVP_OFF & _BODEN_OFF & _PWRTE_ON & _WDT_OFF & _HS_OSC


;*********************************************************************
;                       ASSIGNATIONS SYSTEME                *
;*********************************************************************
; REGISTRE OPTION_REG (configuration)
; ----------------------------------
OPTIONVAL       EQU    B'00000111'
                ; RBPU     b7 : 1= Résistance rappel +5V hors service
                ; INTEDG   b6 : 1= Interrupt sur flanc montant de RB0
                ;              0= Interrupt sur flanc descend. de RB0
                ; TOCS      b5 : 1= source clock = transition sur RA4
                ;              0= horloge interne
                ; TOSE      b4 : 1= Sélection flanc montant RA4(si B5=1)
                ;              0= Sélection flanc descendant RA4
                ; PSA       b3 : 1= Assignation prédiviseur sur Watchdog
```

```
                    ;               0= Assignation prédiviseur sur Tmr0
                    ; PS2/PS0   b2/b0 valeur du prédiviseur
                ;         000 =  1/1 (watchdog) ou 1/2 (tmr0)
                    ;         001 =  1/2            1/4
                    ;         010 =  1/4                  1/8
                    ;         011 =  1/8                  1/16
                    ;         100 =  1/16                 1/32
                    ;         101 =  1/32                 1/64
                    ;         110 =  1/64                 1/128
                    ;         111 =  1/128       1/256


; REGISTRE INTCON (contrôle interruptions standard)
; -------------------------------------------------
INTCONVAL       EQU    B'00000000'
                    ; GIE       b7 : masque autorisation générale interrupt
                ;             ne pas mettre ce bit à 1 ici
                ;             sera mis en temps utile
                    ; PEIE      b6 : masque autorisation générale périphériques
                    ; T0IE      b5 : masque interruption tmr0
                    ; INTE      b4 : masque interuption RB0/Int
                    ; RBIE      b3 : masque interruption RB4/RB7
                    ; T0IF      b2 : flag tmr0
                    ; INTF      b1 : flag RB0/Int
                    ; RBIF      b0 : flag interruption RB4/RB7

; REGISTRE PIE1 (contrôle interruptions périphériques)
; ----------------------------------------------------
PIE1VAL             EQU    B'00000000'
                    ; PSPIE     b7 : Toujours 0 sur PIC 16F786
                    ; ADIE      b6 : masque interrupt convertisseur A/D
                    ; RCIE      b5 : masque interrupt réception USART
                    ; TXIE      b4 : masque interrupt transmission USART
                    ; SSPIE     b3 : masque interrupt port série synchrone
                    ; CCP1IE    b2 : masque interrupt CCP1
```

```
                    ; TMR2IE    b1 : masque interrupt TMR2 = PR2
                    ; TMR1IE    b0 : masque interrupt débordement tmr1

  ; REGISTRE PIE2 (contrôle interruptions particulières)
  ; ------------------------------------------------
  PIE2VAL           EQU   B'00000000'
                    ; UNUSED    b7 : inutilisé, laisser à 0
                    ; RESERVED  b6 : réservé, laisser à 0
                    ; UNUSED    b5 : inutilisé, laisser à 0
                    ; EEIE      b4 : masque interrupt écriture EEPROM
                    ; BCLIE     b3 : masque interrupt collision bus
                    ; UNUSED    b2 : inutilisé, laisser à 0
                    ; UNUSED    b1 : inutilisé, laisser à 0
                    ; CCP2IE    b0 : masque interrupt CCP2

  ; REGISTRE ADCON1 (ANALOGIQUE/DIGITAL)
  ; ------------------------------------
  ADCON1VAL         EQU   B'00000110' ; PORTA en mode digital

  ; DIRECTION DES PORTS I/O
  ; ----------------------
  ;le controle des moteurs se fait par RA0-5, RC0-5 et RD0-7
  ;le controle des LED se fait par RE0-2
  DIRPORTA          EQU   B'00000000'  ; Direction PORTA (1=entrée, 0=sortie)5 bits pour controle moteur)
  DIRPORTBEQU   B'11110000'   ; Direction PORTB en entrée, les 4 non utilisées sont en
                                                          ;sortie pour eviter parasite (pour le paiement on utilise 2 x boutons et 2 x portes seulement)
  DIRPORTCEQU   B'11000000'   ; Direction PORTC (en sortie pour controle moteur sauf 6 et 7 en entrée car RX/TX)
  DIRPORTEEQU   B'11100000'   ; Direction PORTE 3 Bitrs pour controle LED et bit 4 à 0 pour mettre le port D en IO
  DIRPORTD          EQU   B'00000000'  ; Direction PORTD (en sortie pour controle moteur)

  ; CONFIGURATION DU PORT SERIE
  ; ----------------------
```

```
   TX_CONFIG       EQU   B'00000100'  ; Configuration du registre de controle de
l'emission du port serie
   RX_CONFIG       EQU   B'10000000'  ; Configuration du registre de controle de la
reception du port serie
   BAUDRATE        EQU   D'129'       ; fixer la vitesse à 9600

   ;*****************************************************************************
   ;                    ASSIGNATIONS PROGRAMME                    *
   ;*****************************************************************************

   ; exemple
   ; -------
   MASQUE          EQU   H'00FF'
   STX EQU   "["
   ETX EQU   "]"



   ;*****************************************************************************
   ;                          DEFINE                          *
   ;*****************************************************************************
; exemple
   ; -------
   #DEFINE LED_Ok              PORTE,0          ; LED Ok Verte (Position 1)
   #DEFINE LED_Warning  PORTE,1          ; LED Warning Jaune (Position 2)
   #DEFINE LED_Fault    PORTE,2          ; LED Fault Rouge (Position 3, donc 4
pour faciliter test))
   #DEFINE   LED_Pwr           PORTA,5          ; LED Alimentation Rouge
   #DEFINE LED_Bras     PORTC,5          ;LED Bras verte
   #DEFINE LED_Base     PORTA,5          ;LED Base verte
```

```
;*****************************************************************************
;                    MACRO                              *
;*****************************************************************************

                    ; Changement de banques
                    ; ----------------------

BANK0     macro                    ; passer en banque0
          bcf     STATUS,RP0
          bcf     STATUS,RP1
     endm

BANK1     macro                    ; passer en banque1
          bsf     STATUS,RP0
          bcf     STATUS,RP1
     endm

BANK2     macro                    ; passer en banque2
          bcf     STATUS,RP0
          bsf     STATUS,RP1
     endm

BANK3     macro                    ; passer en banque3
          bsf     STATUS,RP0
          bsf     STATUS,RP1
     endm

                    ; Chargement caractéres dans buffer d'emission
                    ; ----------------------

MESS_TX  macro  a1,a2,a3,a4,a5,a6
     BANK0
     MovlwSTX
     MovwfTxCaract1
     Movlwa1
```

```
        MovwfTxCaract2
        Movlwa2
        MovwfTxCaract3
        Movlwa3
        MovwfTxCaract4
        Movlwa4
        MovwfTxCaract5
        Movlwa5
        MovwfTxCaract6
        Movlwa6
        MovwfTxCaract7
        MovlwETX
        MovwfTxCaract8
        endm

MESS_RX macro   a1,a2,a3,a4,a5,a6,a7,8,a9
        BANK0
        MovlwSTX
        MovwfRxCaract1
        Movlwa1
        MovwfRxCaract2
        Movlwa2
        MovwfRxCaract3
        Movlwa3
        MovwfRxCaract4
        Movlwa4
        MovwfRxCaract5
        Movlwa5
        MovwfRxCaract6
        Movlwa6
        MovwfRxCaract7
        Movlwa7
        MovwfRxCaract8
        Movlwa8
        MovwfRxCaract9
```

```
        Movlw a9
        MovwfRxCaract10
        MovlwETX
        MovwfRxCaract8
        endm

SequenceMoteur macro a1,a2,a3,a4
        BANK0
        movlw a1
        movwf Pas1
        movlw a2
        movwf Pas2
        movlw a3
        movwf Pas3
        movlw a4
        movwf Pas4
        endm

;*****************************************************************************
;                    VARIABLES BANQUE 0                          *
;*****************************************************************************

; Zone de 6 bytes pour mettre les caractéres récus (buffer reception)
; ----------------

        CBLOCK      0x20            ; Début de la zone (0x20 à 0x27)

                RxCaract1 :1        ; Zone de 1 byte
        RxCaract2 :1            ; Zone de 1 byte
        RxCaract3 :1            ; Zone de 1 byte
        RxCaract4 :1            ; Zone de 1 byte
        RxCaract5 :1            ; Zone de 1 byte
        RxCaract6 :1            ; Zone de 1 byte
        RxCaract7 :1            ; Zone de 1 byte
        RxCaract8 :1            ; Zone de 1 byte
```

```
        RxCaract9 :1        ; Zone de 1 byte
        RxCaract10 :1       ; Zone de 1 byte
        RxCaract11:1        ; Zone de 1 byte
        RxPtr : 1           ; pointeur du buffer de reception

        ENDC                ; Fin de la zone

    ; Zone de 6 bytes pour mettre les caractéres à emettre (buffer emission)
    ; ----------------

        CBLOCK      0x30        ; Début de la zone (0x30 à 0x37)

        TxCaract1 :1        ; Zone de 1 byte
        TxCaract2 :1        ; Zone de 1 byte
        TxCaract3 :1        ; Zone de 1 byte
        TxCaract4 :1        ; Zone de 1 byte
        TxCaract5 :1        ; Zone de 1 byte
        TxCaract6 :1        ; Zone de 1 byte
        TxCaract7 :1        ; Zone de 1 byte
        TxCaract8 :1        ; Zone de 1 byte
        TxPtr : 1           ; pointeur du buffer d'emission

        ENDC                ; Fin de la zone

    ;
        CBLOCK      0x40        ; Début de la zone (0x30 à 0x37)

        NbrPasBase : 1          ;contient le nombre de pas à effectuer par le moteur de
base
        NbrPasBras : 1          ;contient le nombre de pas à effectuer par le moteur du
bras
        SensBase : 1    ;contient le sns de rotation du moteur base
        SensBras : 1    ;contient le sns de rotation du moteur bras
        Pas1 : 1            ;séquence 1 = 0A
        Pas2 : 1            ;séquence 1 = 09
        Pas3 : 1            ;séquence 1 = 05
```

```
        Pas4 : 1                ;séquence 1 = 06
        PasPtrBase      : 1             ;contient la derniére séquence effectuée pour base
        PasPtrBras      : 1             ;contient la derniére séquence effectuée pour bras
        CmptPtrBase : 1                 ;contient le nombre de seéquence (4) pour
réinitialiser le pts de pas
        CmptPtrBras : 1                 ;contient le nombre de seéquence (4) pour
réinitialiser le pts de pas
        SaveCmptPtr  :          1       ; sauvegarde de CmptPtr
        cmpt1:1
        cmpt2:1
        cmpt3:1
        cmpt4:1
        Speed  :        1       ;contient la vitesse
;       Flags : 1

                ENDC

;*****************************************************************************
;                       VARIABLES ZONE COMMUNE                          *
;*****************************************************************************

; Zone de 16 bytes
; ----------------

        CBLOCK 0x70             ; Début de la zone (0x70 à 0x7F)
        w_temp : 1              ; Sauvegarde registre W
        status_temp : 1         ; sauvegarde registre STATUS
        FSR_temp : 1            ; sauvegarde FSR (si indirect en interrupt)
        PCLATH_temp : 1         ; sauvegarde PCLATH (si prog>2K)

        CmptTmr0 : 1 ;multiplicateur pour le timer0 (temps timer avec prescalaire
                                ;de 256 est de 13,107 ms
        SaveTempoDebounce : 1   ; sauvegarde de la tempo de l'antirebond vanant avec
commande R
        SaveTempoMoteur : 1     ;sauvegarde de la tempo des moteurs venant avec la
commande M
```

```
        SaveNbrTempoMoteur : 1    ;sauvegarde du nombre de fois à attendre la tempo des moteurs (temps = SaveNbrTempoMoteur * SaveTempoMoteur)


        ActualBtnPrt : 1        ;sauvegarde du portB actuel contenant l'état des boutons et porte
        LastBtnPrt : 1          ;sauvegarde du dernier portB lu contenant l'état des boutons et porte
        Flags : 1       ;registre de flag pour usage general (voir DEFINE pour sens flags)
        CmptGeneral : 1 ; cmptr à usage general


        ENDC
        #DEFINE EnvoiMessFlags,0        ; si à 1 envoyer message
        ;#DEFINE    ShutterPos    Flags,1        ; si à 0 donc le shutter est à sa position normale
        ;#DEFINE    ShutterTime   Flags,2        ; si à 1 donc on a lancé le timer pour temps Shutter
        #DEFINE Debounce  Flags,3       ; si à 1 donc on a lancé timer anti rebond
        ;#DEFINE    Appuyer1      Flags,4        ; à 1 si touche 1 appuyée
        #DEFINE    MoteurTime   Flags,2        ; si à 1 donc on a lancé le timer pour temps moteur
        #DEFINE    FlagMoteurBase      Flags,6        ; si à 1 alors moteur base a deja tourné
        #DEFINE    FlagMoteurBras      Flags,7        ; si à 1 alors loteur bras a deja tourné
        #DEFINE    FlagPstMotBase      Flags,5        ;si moteur base sur position initiale alors à 1
        #DEFINE    FlagPstMotBras      Flags,4        ;si moteur bras sur position initiale alors à 1
        #DEFINE    FlagEteindre   Flags,2        ;flag eteindre
        #DEFINE    FlagInitMoteur      Flags,1        ;flag pour demander initialisation moteur
   ;*****************************************************************************
   ;               DEMARRAGE SUR RESET                     *
   ;*****************************************************************************


        org 0x000               ; Adresse de départ après reset
        goto    init            ; Initialiser
```

```
; ////////////////////////////////////////////////////////////////////

;                    I N T E R R U P T I O N S

; ////////////////////////////////////////////////////////////////////

;*********************************************************************
;                  ROUTINE INTERRUPTION                              *
;*********************************************************************
;---------------------------------------------------------------------
; Si on n'utilise pas l'adressage indirect dans les interrupts, on se passera
; de sauvegarder FSR
; Si le programme ne fait pas plus de 2K, on se passera de la gestion de
; PCLATH
;---------------------------------------------------------------------
                        ;sauvegarder registres
                        ;---------------------
      org 0x004                 ; adresse d'interruption
      movwf   w_temp            ; sauver registre W
      swapf  STATUS,w           ; swap status avec résultat dans w
      movwf status_temp         ; sauver status swappé
      movf   FSR , w                    ; charger FSR
      movwf FSR_temp            ; sauvegarder FSR
      movf   PCLATH , w         ; charger PCLATH
      movwf PCLATH_temp         ; le sauver
      clrf     PCLATH                   ; on est en page 0
      BANK0                             ; passer en banque0

                        ; switch vers différentes interrupts
                        ; inverser ordre pour modifier priorités
                        ; mais attention alors au test PEIE
                        ; effacer les inutiles
                        ;--------------------------------------

                        ; Interruption réception USART
```

```
                ; ----------------------------
int_rx
        bsf     STATUS,RP0      ; sélectionner banque1
        btfss   PIE1,RCIE       ; tester si interrupt autorisée
        goto    int_tx          ; non sauter
        bcf     STATUS,RP0      ; oui, sélectionner banque0
        btfss   PIR1,RCIF       ; oui, tester si interrupt en cours
        goto    int_tx          ; non sauter
        call    Caract_Recu     ; oui, traiter interrupt
                                ; LE FLAG NE DOIT PAS ETRE REMIS
A 0
                                ; C'EST LA LECTURE DE RCREG QUI
LE PROVOQUE
        goto    restorereg      ; et fin d'interrupt


                ; Interruption transmission USART
                ; ------------------------------
int_tx
        bsf     STATUS,RP0      ; sélectionner banque1
        btfss   PIE1,TXIE       ; tester si interrupt autorisée
        goto    int_tmr0        ; non sauter
        bcf     STATUS,RP0      ; oui, sélectionner banque0
        btfss   PIR1,TXIF       ; oui, tester si interrupt en cours
        goto    int_tmr0        ; non sauter

        movf    TxPtr,W
        movwf   FSR
        movf    INDF,W
        movwf   TXREG
        incf    TxPtr,f; pointer sur octet suivant

                ; tester si dernier octet
                ; -----------------------
        xorlw   ETX             ; comparer octet envoyé avec Line-feed
        btfss   STATUS,Z    ; égalité?
```

```
        goto    restorereg      ; non, retour d'interruption

                        ; traiter fin d'émission
                        ; ----------------------
        bsf             STATUS,RP0          ; passer en banque 1
        bcf             PIE1,TXIE           ; fin des interruptions émission USART
        bcf             STATUS,RP0          ; repasser banque 0
        movlw TxCaract1             ; adresse du buffer de sortie
        movwf TxPtr                 ; prochain caractère = premier
                                                ; fin d'interruption
                                                ; oui, traiter interrupt
                                                ; LE FLAG NE DOIT PAS ETRE REMIS
A 0
                                                ; C'EST L'ECRITURE DE TXREG QUI
LE PROVOQUE
        goto    restorereg          ; et fin d'interrupt

                        ; Interruption TMR0
                        ; -----------------
  int_tmr0
        btfsc   INTCON,T0IE             ; tester si interrupt timer autorisée
        btfss   INTCON,T0IF             ; oui, tester si interrupt timer en cours
        goto    int_rb          ; non test suivant
                                                ; oui, traiter interrupt tmr0
        movlw 0
        xorwf  CmptGeneral
        btfss   STATUS,Z

        ;goto   ProchainPassageCmptGeneral
        decfsz  CmptTmr0
        goto    ProchainPassage
        btfss   MoteurTime
        goto    AntiRebond
        BANK0
        clrf    PORTA                   ;arreter les moteurs aprés un temps
        clrf    PORTD
```

```
        bcf     PORTC,0
        bcf     PORTC,1
        bcf     PORTC,2
        bcf     PORTC,3
        bcf     PORTC,4
        bcf     PORTC,5
        bcf     MoteurTime
        ;envoi message fin de distribution
        MESS_TX "F","x","x","x","x","x"
        bsf     EnvoiMess
        movlw RxCaract1
        movwf RxPtr
SortiePourDebounce
        bcf     INTCON,T0IE             ; devalider interrupt tmr0
ProchainPassage
        bcf     INTCON,T0IF             ; effacer flag interrupt tmr0
        goto    restorereg              ; et fin d'interruption
                                        ; SUPPRIMER CETTE LIGNE POUR
                                        ; TRAITER PLUSIEURS INTERRUPT
                                        ; EN 1 SEULE FOIS
ProchainPassageCmptGeneral
        decf    CmptGeneral
        bcf     INTCON,T0IF             ; effacer flag interrupt tmr0
        goto    restorereg

AntiRebond
        btfss   Debounce
        goto    ProchainPassage
        bcf     Debounce
        ;bcf     Appuyer1
        BANK0
        movf    PORTB,W
        bcf     INTCON,RBIF
        bsf     INTCON,RBIE             ; valider interruption RB4/7
```

```
        goto    SortiePourDebounce

                        ; interruption RB4/RB7
                        ; --------------------
int_rb
        btfsc   INTCON,RBIE             ; tester si interrupt RB4/7 autorisée
        btfss   INTCON,RBIF             ; oui, tester si interrupt RB4/7 en cours
        goto    int_rb0                 ; non sauter
                                                ; oui traiter interruption
        bcf             INTCON,RBIE             ; dévalider interruption RB4/7

        BANK0

        ;movf   PORTB,W
        ;bcf            INTCON,RBIF
        call    CapteurArret            ; oui, traiter interrupt RB4/7
        ;movf   PORTB,W                 ;lecture avant clr de RBIF
        ;bcf            INTCON,RBIF
        ;bsf            INTCON,RBIE             ; valider interruption RB4/7
        goto    restorereg              ; et fin d'interrupt

                        ; Interruption RB0/INT
                        ; --------------------
int_rb0
        btfsc   INTCON,INTE             ; tester si interrupt RB0 autorisée
        btfss   INTCON,INTF             ; oui, tester si interrupt RB0 en cours
        goto    restorereg              ;intsw2                 ; non sauter au test suivant
                                                ; oui traiter interruption
        bcf             INTCON,INTF             ; effacer flag interupt RB0
        bcf             INTCON,INTE             ;devalider l'interruption RB0
        ;call   PositionShutter                 ; oui, traiter interrupt RB0
        goto    restorereg              ; et fin d'interruption
                                                ; SUPPRIMER CETTE LIGNE POUR
                                                ; TRAITER PLUSIEURS INTERRUPT
                                                ; EN 1 SEULE FOIS
```

```
                    ;restaurer registres
                    ;-------------------
  restorereg
      movf   PCLATH_temp , w    ; recharger ancien PCLATH
      movwf PCLATH              ; le restaurer
      movf   FSR_temp , w ; charger FSR sauvé
      movwf FSR             ; restaurer FSR
      swapf  status_temp,w ; swap ancien status, résultat dans w
      movwf   STATUS            ; restaurer status
      swapf   w_temp,f      ; Inversion L et H de l'ancien W
              ; sans modifier Z
      swapf   w_temp,w      ; Réinversion de L et H dans W
                            ; W restauré sans modifier status
      retfie            ; return from interrupt

  ;*****************************************************************************
  ;                   INTERRUPTION TIMER 0                        *
  ;*****************************************************************************
  Tempo
      return                ; fin d'interruption timer
                            ; peut être remplacé par
                            ; retlw pour retour code d'erreur

  ;*****************************************************************************
  ;                SOUS PROGRAMME DE RECEPTION USART              *
  ;*****************************************************************************
  ;_____________________________________________voir astuce de programmation
pour comparaison (xorlw 1^2)
  Caract_Recu

      movf   RxPtr,W
      movwf         FSR
      movf   RCREG,W
      movwf INDF            ;sauvegarde dans buffer
```

```
        xorlw   ETX             ;comparer avec fin de message
        btfsc   STATUS,Z
        goto    ExeCmd

        incf    RxPtr           ;incrementer buffer

        return                  ; fin d'interruption
                                ; peut être remplacé par
                                ; retlw pour retour code d'erreur
ExeCmd
        BANK1
        bcf     PIE1,RCIE       ;bloquer la reception aprés la reception compléte du message
        BANK0
;       call    CheckCRC        ;verification du CRC
        movlw   RxCaract1       ;initialiser pointeur
        movwf   RxPtr
        movf    RxPtr,W
        movwf   FSR
        movf    INDF,W          ;lire caractere recu se trouvant dans buffer reception
        xorlw   STX             ;comparer avec code debut de chaine
        btfss   STATUS,Z
        goto    ErrCmd

;       incf    RxPtr
;       movf    RxPtr,W
;       movwf   FSR
;       movf    INDF,W
;       xorlw   "2"             ;verifier si commande destinée à cette carte
;       btfss   STATUS,Z
;       goto    ErrCarte
;       goto    SortieInt2

        incf    RxPtr
        movf    RxPtr,W
```

```
movwf FSR
movf  INDF,W
xorlw "M"                  ;comparer avec code moteur
btfss STATUS,Z
goto  SiInitMoteur

incf  RxPtr
incf  RxPtr
incf  RxPtr
incf  RxPtr

incf  RxPtr          ;charger le nombre de pas base
movf  RxPtr,W
movwf FSR
movf  INDF,W                    ;nbr pas base chargé
movwf NbrPasBase

incf  RxPtr          ;charger le nombre de pas bras
movf  RxPtr,W
movwf FSR
movf  INDF,W                    ;nbr pas bras chargé
movwf NbrPasBras

incf  RxPtr          ;charger le sens du moteur base
movf  RxPtr,W
movwf FSR
movf  INDF,W                    ;sens base chargé
movwf SensBase

incf  RxPtr          ;charger le sens du moteur bras
movf  RxPtr,W
movwf FSR
movf  INDF,W                    ;sens bras chargé
movwf SensBras
```

```
        ;Testtttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
;       goto    MoteurBase
        call    MoteurBase
        goto    SortieInt

SiInitMoteur
        movf    RxPtr,W
        movwf   FSR
        movf    INDF,W
        xorlw   "I"                     ;comparer avec code moteur
        btfss   STATUS,Z
        goto    SiVitesse
;testtttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
;       goto    InitMoteur
;       Call    InitMoteur
        bsf             FlagInitMoteur
;       goto    start
        goto    SortieInt

SiVitesse
        movf    RxPtr,W
        movwf   FSR
        movf    INDF,W
        xorlw   "V"                     ;comparer avec code moteur
        btfss   STATUS,Z
        goto    SiRebond

        incf    RxPtr           ;charger la Vitesse
        movf    RxPtr,W
        movwf   FSR
        movf    INDF,W                          ;nbr pas base chargé
        movwf   Speed
;testttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
;       goto    MoteurBase
        goto    SortieInt
```

```
SiRebond
        movf    RxPtr,W
        movwf   FSR
        movf    INDF,W
        xorlw   "R"
        btfss   STATUS,Z
        goto    SiLED
        incf    RxPtr           ;charger le compteur avec la valeur venant avec la commande
        movf    RxPtr,W
        movwf   FSR
        movf    INDF,W                  ;valeur du compteur chargée
        movwf   SaveTempoDebounce       ;charger nombre de passe pour avoir n x 13,107
ms
        goto    SortieInt

SiLED
        movf    RxPtr,W
        movwf   FSR
        movf    INDF,W
        xorlw   "L"
        btfss   STATUS,Z
        goto    ErrCmd
        ;Allumer ou eteindre
        incf    RxPtr
        incf    RxPtr
        incf    RxPtr
        movf    RxPtr,W
        movwf   FSR
        movf    INDF,W
        andlw   B'00000001'             ;test LED Base
        btfsc   W,0
        goto    AlumLedBase             ;Allumer LED_Base
        bcf             LED_Base
        goto    TestLedBras   ;         return
```

```
AlumLedBase
        bsf             LED_Base
TestLedBras
        incf    RxPtr
        movf    RxPtr,W
        movwf   FSR
        movf    INDF,W
        andlw   B'00000001'             ;test Led Bras
        btfsc   W,0
        goto    AlumLedBras
        bcf             LED_Bras
        goto    SortieInt
AlumLedBras
        bsf             LED_Bras
;       goto    SortieInt

SortieInt
        MESS_TX     "C","r","t","2","O","k"
SortieInt1
        bsf             EnvoiMess
SortieInt2
        movlw   RxCaract1
        movwf   RxPtr
        return

ErrCmd
        ;envoyer message d'erreur NACK ???
        movlw   RxCaract1
        movwf   RxPtr
        MESS_TX     "N","A","K","D","2","x"
        bsf             EnvoiMess
        return

ErrCarte
        ;envoyer message d'erreur NACK ???
```

```
        movlw RxCaract1
        movwf RxPtr
        MESS_TX     "C","r","t","2","N","o"
        bsf         EnvoiMess
        return

;*****************************************************************************
;       SOUS PROGRAMME DE TRAITEMENT DES BOUTONS ET CAPTEURS   *
;*****************************************************************************

CapteurArret
        movf   PORTB,W
        bcf         LED_Pwr
        bsf         FlagPstMotBase
        bsf         FlagPstMotBras
        bcf         INTCON,RBIF       ; RAZ flag int RB4-7
        bsf         INTCON,RBIE       ;valider l'interruption RB4-7 aprés le status
        ;goto   InitMoteur
        return

        ;movf  PORTB,W
        ;MESS_TX    "P","0","0","0","0","x"
;       movf   PORTB,W
;       movwf TxCaract3
;       bsf         EnvoiMess     ;envoyer etat des portes et boutons
;       movlw RxCaract1
;       movwf RxPtr
;       bcf         INTCON,RBIF       ; RAZ flag int RB4-7
;       bsf         INTCON,RBIE       ;valider l'interruption RB4-7 aprés le status

;       return
        movlw 0xF0
        btfsc   W,5             ;test si moteur base sur position 0
        goto    TestPstBras
;testttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
```

```
;       bsf             LED_Base
        bsf             FlagPstMotBase
TestPstBras
        btfsc   W,6             ;test si moteur bras sur position 0
        goto    TestBpArret
;testttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
;       bsf             LED_Bras
        bsf             FlagPstMotBras
TestBpArret
        btfsc   W,7
        bsf             FlagEteindre

;       bsf             Debounce                ; initilaiser falg anti rebond
;       movf    SaveTempoDebounce,W
;       movwf   CmptTmr0                ;charger nombre de passe pour avoir n x 13,107 ms
        bcf             INTCON,RBIF         ; RAZ flag int RB4-7
        bsf             INTCON,RBIE         ;valider l'interruption RB4-7 aprés le status
        return

;*****************************************************************************
;       SOUS PROGRAMME DE TRAITEMENT DES BOUTONS ET PORTES     *
;*****************************************************************************
BoutonPorte

        BANK0

        MESS_TX     "P","0","0","0","0","x"

        movf    PORTB,W
        movwf   TxCaract3
        movf    PORTC,W
        movwf   TxCaract4
        bsf             EnvoiMess      ;envoyer etat des portes et boutons
        movlw   RxCaract1
        movwf   RxPtr
```

```
        bsf             Debounce                ; initilaiser falg anti rebond
        movf    SaveTempoDebounce,W
        movwf CmptTmr0              ;charger nombre de passe pour avoir n x 13,107 ms
        bsf             INTCON,T0IE                 ;demarrer delai avec timer 0

        return                  ; fin d'interruption
                                ; peut être remplacé par
                                ; retlw pour retour code d'erreur

; ////////////////////////////////////////////////////////////////////////////

;                       P R O G R A M M E

; ////////////////////////////////////////////////////////////////////////////

;*****************************************************************************
;                       INITIALISATIONS                         *
;*****************************************************************************
init

                        ; initialisation PORTS (banque 0 et 1)
                        ; ----------------------------------

        bsf             STATUS,RP0 ; passer en banque1

        movlw ADCON1VAL
        movwf ADCON1                ;mettre le port A en mode IO

        movlw DIRPORTB    ; Direction PORTB (en entrée)
        movwf TRISB       ; écriture dans registre direction
        movlw DIRPORTC    ; Direction PORTC (en sortie sauf RC/TX)
        movwf TRISC       ; écriture dans registre direction
        movlw DIRPORTA    ; Direction PORTA (en sortie)
        movwf TRISA       ; écriture dans registre direction
```

```
movlw DIRPORTD   ; Direction PORTD (en sortie)
movwf TRISD      ; écriture dans registre direction
movlw DIRPORTE   ; Direction PORTE (en sortie)
movwf TRISE      ; écriture dans registre direction

;initialiser les ports
BANK0                    ; sélectionner banque0
clrf  PORTA              ; Sorties PORTA à 0
clrf  PORTB              ; sorties PORTB à 0
clrf  PORTD
clrf  PORTE
clrf  PORTC



                ; initialisation DU PORT SERIE
                ; ----------------------------------

BANK0
movlw RX_CONFIG ;charger config emission serie
movwf RCSTA
BANK1
movlw TX_CONFIG ;charger config transmission serie
movwf TXSTA
movlw BAUDRATE  ;fixer la vitesse
movwf SPBRG
clrf  Flags

                ; Registre d'options (banque 1)
                ; ----------------------------
movlw OPTIONVAL ; charger masque
movwf OPTION_REG         ; initialiser registre option avec prescalaire TMR0 = 256

                ; registres interruptions (banque 1)
                ; ---------------------------------
```

```
        movlw INTCONVAL; charger valeur registre interruption
        movwf INTCON            ; initialiser interruptions
        movlw PIE1VAL           ; Initialiser registre
        movwf PIE1          ; interruptions périphériques 1
        movlw PIE2VAL           ; initialiser registre
        movwf PIE2          ; interruptions périphériques 2

                        ; Effacer RAM banque 0
                        ; --------------------
        bcf             STATUS,RP0 ; sélectionner banque 0
        movlw 0x20          ; initialisation pointeur
        movwf FSR           ; d'adressage indirect
init1
        clrf    INDF        ; effacer ram
        incf    FSR,f       ; pointer sur suivant
        btfss   FSR,7       ; tester si fin zone atteinte (>7F)
        goto    init1       ; non, boucler

                        ; initialiser les pointeurs des buffers
                        ;-------------------------------------
        movlw RxCaract1
        movwf RxPtr
        movlw TxCaract1
        movwf TxPtr

                        ;initialiser les paramtéres moteur
                        ;---------------------------------
        call    InitParamMoteur
;fermer relais alim
        bsf             LED_Pwr

                        ;initialiser timer 0
                        ;-------------------------------------
        clrf    TMR0
```

```
                    ; autoriser interruptions (banque 0)
                    ; ---------------------------------
        clrf    PIR1            ; effacer flags 1 des interruptions
        clrf    PIR2            ; effacer flags 2 des interruptions
        bsf             INTCON,PEIE         ; autoriser interruptions périphériques
        bsf             RCSTA,CREN          ;autoriser reception en continu

        bsf             INTCON,GIE ; valider interruptions

;testttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
;       MESS_RX     "1","I",0x33,0x05,"G","G"
;       goto    ExeCmd
;       goto    CapteurArret
;testttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt

        movlw TxCaract1
        movwf TxPtr
;_____________________________________________________________________
        MESS_TX     "A","C","K","D","2","x"
;_____________________________________________________________________
        BANK1
        bsf             TXSTA,TXEN
        bsf             PIE1,TXIE     ;autoriser interruption emission
                                      ; attendre fin de l'émission

                                      ; ----------------------
loop
        BANK1
        btfsc   PIE1,TXIE     ; reste des caractères à envoyer?
        goto    $-2             ; oui, attendre
        btfss   TXSTA,TRMT          ; buffer de sortie vide?
        goto    $-4             ; non, attendre

        BANK1
        bsf             PIE1,RCIE     ;autoriser interruption reception
```

```
        bcf     INTCON,RBIF     ; RAZ flag int RB4-7
        bsf     INTCON,RBIE     ;valider l'interruption RB4-7 aprés le status



        goto    start           ; programme principal


;*****************************************************************************
;               PROGRAMME PRINCIPAL                     *
;*****************************************************************************
start
        BANK0
        clrwdt                  ; effacer watch dog
        btfss   FlagInitMoteur
        goto    SiEteindre
        call    InitMoteur
SiEteindre
        btfss   FlagEteindre
        goto    SiEnvoiMess
        call    Eteindre
SiEnvoiMess
        btfss   EnvoiMess
        goto    start
        bcf     EnvoiMess
        movlw TxCaract1
        movwf TxPtr
        BANK1
        bsf     TXSTA,TXEN
        bsf     PIE1,TXIE       ;autoriser interruption emission
                                ; attendre fin de l'émission
                                ; ----------------------
        BANK1
        btfsc   PIE1,TXIE       ; reste des caractères à envoyer?
```

```
        goto    $-2                     ; oui, attendre
        btfss   TXSTA,TRMT              ; buffer de sortie vide?
        goto    $-4                     ; non, attendre
        BANK1
        bsf             PIE1,RCIE       ;re valider la reception
        goto    start

CheckCRC
        return




;*********************************************************************
;                       SOUS PROGRAMME DE GESTION MOTEUR                *
;*********************************************************************
;Sequence moteur pas a pas
MoteurBase
;appeler la routine G ou D selon commande
        movf    SensBase,W
        xorlw   "G"
        btfss   STATUS,Z
        goto    SiDroiteBase
;testttttttttttttttttttttttttttttttttttttttttttttt
;       goto    MoteurBaseG
        call    MoteurBaseG
        return
SiDroiteBase
        movf    SensBase,W
        xorlw   "D"
        btfss   STATUS,Z
        goto    MoteurBras
;testttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
        ;goto   MoteurBaseD
        call    MoteurBaseD
        return
```

```
MoteurBras
;appeler la routine G ou D selon commande
        movf    SensBras,W
        xorlw   "G"
        btfss   STATUS,Z
        goto    SiDroiteBras
;testttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
;       goto    MoteurBrasG
        call    MoteurBrasG
        return
SiDroiteBras
        movf    SensBras,W
        xorlw   "D"
        btfss   STATUS,Z
        return
        call    MoteurBrasD
        return

;rotation moteur base vers la droite
MoteurBaseD

        bsf             FlagMoteurBase
SequenceSuivanteD
PasSuivantD
        movf    PasPtrBase,W
        movwf   FSR
        movf    INDF,W                  ;lire caractere recu se trouvant dans buffer reception
        iorlw   0x10            ;allumer LED bras
        movwf   PORTC
;testttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
        ;goto   TEMPO
        call    TEMPO
        decfsz  NbrPasBase
        goto    ProchaineSeqD
        goto    SortieMoteurBaseD
```

```
ProchaineSeqD
        decfsz  CmptPtrBase,f
        goto    ProchainPasD
        movlw   Pas1
        movwf   PasPtrBase
        movlw   0x04
        movwf   CmptPtrBase
        goto    SequenceSuivanteD
ProchainPasD
        incf    PasPtrBase
        goto    PasSuivantD
SortieMoteurBaseD
        bcf             LED_Base
        goto    MoteurBras

;rotation moteur base vers la gauche
MoteurBaseG
SequenceSuivanteG
        btfsc   FlagMoteurBase
        goto    PasSuivantG
        movlw   Pas4
        movwf   PasPtrBase

PasSuivantG
        movf    PasPtrBase,W
        movwf   FSR
        movf    INDF,W                  ;lire caractere recu se trouvant dans buffer reception
        iorlw   0x10            ;allumer LED bras
        movwf   PORTC
        call    TEMPO
        decfsz  NbrPasBase
        goto    ProchaineSeqG
        goto    SortieMoteurBaseG
ProchaineSeqG
        decfsz  CmptPtrBase,f
```

```
        goto    ProchainPasG
        movlw Pas4
        movwf PasPtrBase
        movlw 0x04
        movwf CmptPtrBase
        goto    PasSuivantG
ProchainPasG
        decf    PasPtrBase
        goto    PasSuivantG
SortieMoteurBaseG
        bsf             FlagMoteurBase
        bcf             LED_Base
        goto    MoteurBras

;---------------
;rotation moteur bras vers la droite
MoteurBrasD
        bsf             FlagMoteurBras
        bsf             LED_Bras
SequenceSuivanteD1
PasSuivantD1

        movf    PasPtrBras,W
        movwf FSR
        movf    INDF,W                  ;lire caractere recu se trouvant dans buffer reception
        iorlw   0x20            ;forcer à 1 PORTA.5 pour garder le relais alim fermé
        movwf PORTA
        call    TEMPO
        decfsz  NbrPasBras
        goto    ProchaineSeqD1
        goto    SortieMoteurBrasD
ProchaineSeqD1
        decfsz  CmptPtrBras,f
        goto    ProchainPasD1
        movlw Pas1
```

```
        movwf PasPtrBras
        movlw 0x04
        movwf CmptPtrBras
        goto  SequenceSuivanteD1
ProchainPasD1
        incf  PasPtrBras
        goto  PasSuivantD1
SortieMoteurBrasD
        bcf           LED_Bras
        return

;rotation moteur base vers la gauche
MoteurBrasG
        bsf           LED_Bras
        btfsc FlagMoteurBras
        goto  PasSuivantG1
SequenceSuivanteG1
        movlw Pas4
        movwf PasPtrBras

PasSuivantG1
        movf  PasPtrBras,W
        movwf FSR
        movf  INDF,W            ;lire caractere recu se trouvant dans buffer reception
        iorlw 0x20          ;forcer à 1 PORTA.5 pour garder le relais alim fermé
        movwf PORTA
        call  TEMPO
        decfsz NbrPasBras
        goto  ProchaineSeqG1
        goto  SortieMoteurBrasG
ProchaineSeqG1
        decfsz CmptPtrBras,f
        goto  ProchainPasG1
        movlw Pas4
        movwf PasPtrBras
```

```
        movlw 0x04
        movwf CmptPtrBras
        goto    PasSuivantG1
ProchainPasG1
        decf    PasPtrBras
        goto    PasSuivantG1
SortieMoteurBrasG
        bsf             FlagMoteurBras
        bcf             LED_Bras
        return

;programme temporisation
TEMPO
        movf    Speed,W
;       movlw 0x50
        movwf cmpt3
boucle3
;       movlw 0x50
        movf    Speed,W
        movwf cmpt2
boucle2
;       movlw 0x50
        movf    Speed,W
        movwf cmpt1
boucle1
        nop
        decfsz  cmpt1,f
        goto    boucle1
        decfsz  cmpt2,f
        goto    boucle2
        decfsz  cmpt3,f
        goto    boucle3
        return
```

```
;*****************************************************************************
;                    SOUS PROGRAMME D'INITIALISATION MOTEUR                  *
;*****************************************************************************
InitMoteur
;       bsf             FlagMoteurBase
        movlw 0xC8  ;charger le nbr de pas à 200 pour faire un tour complet et attendre position initiale
        movwf NbrPasBase
        movwf NbrPasBras
PasSuivantDD
        btfsc   FlagPstMotBase
        goto    SortieMoteurBaseDD
        movf    PasPtrBase,W
        movwf FSR
        movf    INDF,W              ;lire caractere recu se trouvant dans buffer reception
        iorlw   0x10            ;allumer LED bras
        movwf PORTC
        call    TEMPO
        decfsz  NbrPasBase
        goto    ProchaineSeqDD
        goto    SortieMoteurBaseDD
ProchaineSeqDD
        decfsz  CmptPtrBase,f
        goto    ProchainPasDD
        movlw Pas1
        movwf PasPtrBase
        movlw 0x04
        movwf CmptPtrBase
        goto    PasSuivantDD
ProchainPasDD
        incf    PasPtrBase
        goto    PasSuivantDD
SortieMoteurBaseDD
        bcf             LED_Base
```

```
;MoteurBrasD
    bsf         LED_Bras
PasSuivantD1D
    btfsc   FlagPstMotBras
    goto    SortieMoteurBrasDD
    movf    PasPtrBras,W
    movwf   FSR
    movf    INDF,W              ;lire caractere recu se trouvant dans buffer reception
    iorlw   0x20            ;forcer à 1 PORTA.5 pour garder le relais alim fermé
    movwf   PORTA
    call    TEMPO
    decfsz  NbrPasBras
    goto    ProchaineSeqD1D
    goto    SortieMoteurBrasDD
ProchaineSeqD1D
    decfsz  CmptPtrBras,f
    goto    ProchainPasD1D
    movlw   Pas1
    movwf   PasPtrBras
    movlw   0x04
    movwf   CmptPtrBras
    goto    PasSuivantD1D
ProchainPasD1D
    incf    PasPtrBras
    goto    PasSuivantD1D
SortieMoteurBrasDD
    bcf         LED_Bras

    call    InitParamMoteur
    return


InitParamMoteur
    movlw   Pas1
    movwf   PasPtrBase
```

```
        movwf PasPtrBras
        movlw 0x04
        movwf CmptPtrBase
        movlw 0x04
        movwf CmptPtrBras
        bcf         FlagMoteurBase
        bcf         FlagMoteurBras
        bcf         FlagPstMotBase
        bcf         FlagPstMotBras
        bcf         FlagInitMoteur
        SequenceMoteur 0x0A, 0x09, 0x05, 0x06
        movlw 0x50
        movwf Speed
        return

Eteindre
        call    InitMoteur
        bcf         LED_Pwr
        return

        END                 ; directive fin de programme
```

Annexe 6-Programme de l'interface pour le calcul des angles de rotation :

```
Public Teta(3) As Double    'Rotation Moteur 1, 2 et 3
Public SaveTeta(3) As Double  'sauvegarde des rotation moteur
Public TetaPas(3) As Integer   'rotation des moteurs en nombre de pas
Public Lgr(2) As Double  'Longueur des bras 1, 2 et 3
Public PsObj(2) As Double  'Position X, Y, Z de l'objet
Public TlObj As Double   'Taille de l'objet
Public RyPince As Double   'Rayon de la pince
Public CmdCtrl(1) As String 'contient les paramétres à envoyer au moteurs
Public SensRot(3) As String  'sens de rotation des moteurs
Public FlagMoteurBase As Boolean 'indique sue le moteur base a deja tourne
Public FlagMoteurBras As Boolean 'indique sue le moteur bras a deja tourne
Public LedBaseOn As Boolean
Public LedBras1On As Boolean
Public LedPinceOn As Boolean
Public LedBras2On As Boolean
Public CmdLed   As String   'contient la commande des leds
'Public CmdLed2   As String   'contient la commande des leds
Public Debounce As Integer   'contient la valeur de l'anti rebond
Public Speed    As Double   'contient la vitesse

'Public OnLineOn As Boolean

Public a As Double



Sub Main()
InitVar
LedBaseOn = False
LedBras1On = False
LedBras2On = False
LedPinceOn = False
```

```
CmdLed = "[L0000xxxx]"
'CmdLed2 = "[L00xx]"

'initialisation port serie
'Ouvrir Port LED
Robot.ComBras.CommPort = 16
Robot.ComBras.Settings = "9600,N,8,1"
Robot.ComBras.RThreshold = 1
Robot.ComBras.SThreshold = 1
Robot.ComBras.InputLen = 0
Robot.ComBras.InputMode = comInputModeText

If Not Robot.ComBras.PortOpen Then Robot.ComBras.PortOpen = True

'Robot.RotationMoteur(0).Text = Robot.AnimBras.Width
'Robot.RotationMoteur(1).Text = Robot.AnimBras.Height
Robot.Show
End Sub

Public Sub InitVar()
'initialisation des variables
   For i = 0 To 2
      Teta(i) = 0
      SaveTeta(i) = 0
      Lgr(i) = 0
      PsObj(i) = 0
Robot.LongueurBras(i).Text = 0
      Robot.Objet_XYZ(i).Text = 0
Robot.RotationMoteur(i).Text = 0
   Next i
   Teta(3) = 0
   SaveTeta(3) = 0
   Robot.Lgr_Obj.Text = 0
   Robot.Ry_Pince.Text = 0
TlObj = 0
```

```
    RyPince = 0

    FlagMoteurBras = False
    FlagMoteurBase = False

End Sub
Public Sub CalculRotation()
'calcul des rotations des différents moteurs
    'charger les valeurs vers les variables en les verifiant si numeriques
For i = 0 To 2
        If IsNumeric(Robot.LongueurBras(i).Text) Then
            Lgr(i) = Robot.LongueurBras(i).Text
            Lgr(i) = Lgr(i) / 1000

        Else
            Robot.LongueurBras(i).Text = 0
        End If

        If IsNumeric(Robot.Objet_XYZ(i).Text) Then
            PsObj(i) = Robot.Objet_XYZ(i).Text
            PsObj(i) = PsObj(i) / 1000
        Else
            Robot.Objet_XYZ(i).Text = 0
        End If
    Next i

    If IsNumeric(Robot.Lgr_Obj.Text) Then
        TlObj = Robot.Lgr_Obj.Text
        TlObj = TlObj / 1000
    Else
        Robot.Lgr_Obj.Text = 0
    End If

        If IsNumeric(Robot.Ry_Pince.Text) Then
        RyPince = Robot.Ry_Pince.Text
```

```
        RyPince = RyPince / 1000
    Else
        Robot.Lgr_Obj.Text = 0
End If

    'si un paramétres a été remis à zero pour cause d'erreur numérique alors sortir

For i = 0 To 2
        If Robot.LongueurBras(i).Text = 0 Then Exit Sub
        If Robot.Objet_XYZ(i).Text = 0 Then Exit Sub
    Next i
    If Robot.Lgr_Obj.Text = 0 Then Exit Sub
    If Robot.Ry_Pince.Text = 0 Then Exit Sub

'calculer moteur 1
    Teta(0) = Atn(PsObj(0) / PsObj(1))

    'calculer moteur 3
    Teta(2) = (PsObj(0) ^ 2) + (PsObj(1) ^ 2) + ((PsObj(2) - Lgr(0)) ^ 2) - (Lgr(1) ^ 2) -
(Lgr(2) ^ 2)
    Teta(2) = Teta(2) / (2 * Lgr(1) * Lgr(2))
    Teta(2) = Atn(-Teta(2) / Sqr(-Teta(2) * Teta(2) + 1)) + 2 * Atn(1)
'Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
'calculer moteur 2
    Teta(1) = Lgr(1) + (Lgr(2) * Cos(Teta(2)) * (PsObj(2) - Lgr(0)))
Teta(1) = Teta(1) - (Lgr(2) * Sin(Teta(2)) * (PsObj(1) / (Cos(Teta(0)))))
Teta(1) = Teta(1) / ((Lgr(1) + (Lgr(2) * Cos(Teta(2)))) + (Lgr(2) * Sin(Teta(2))))
    'calculer arccos de Teta(1)
    Teta(1) = Atn(-Teta(1) / Sqr(-Teta(1) * Teta(1) + 1)) + 2 * Atn(1)

    'Calculer la rotation du moteur pince
Teta(3) = (180 - (TlObj * (180 / RyPince))) / 2


    Exit Sub
```

```
End Sub
Public Sub FormatData()
   'formattage des données

For i = 0 To 2
        Teta(i) = ((Teta(i) * 180) / 3.14128) 'en °
TetaPas(i) = Teta(i) / 1.8 'en pas
TetaPas(i) = Abs(TetaPas(i))
        If FlagMoteurBase And i < 2 Then TetaPas(i) = TetaPas(i) + 1
        If FlagMoteurBras And i > 1 Then TetaPas(i) = TetaPas(i) + 1
        'Teta(i) = Teta(i) - SaveTeta(i)
        If (Teta(i) - SaveTeta(i)) <= 180 Then SensRot(i) = "G" Else SensRot(i) = "D"
      Next i
      TetaPas(3) = Teta(3) / 1.8
      TetaPas(3) = Abs(TetaPas(3))

   'formatter teta pince

End Sub
Public Sub TransfertCoordonnees()
'format message = STX, 1, M, Pas M1, Pas M2, Sens M1, Sens M2, ETX
TetaPas(0) = 12
TetaPas(1) = 12
TetaPas(2) = 20
TetaPas(3) = 20
SensRot(0) = "G"
SensRot(1) = "G"
SensRot(2) = "G"
```

```
CmdCtrl(0) = "[" & "M" & Chr(TetaPas(0)) & Chr(TetaPas(1)) & SensRot(0) & SensRot(1) & Chr(TetaPas(2)) & Chr(TetaPas(3)) & SensRot(2) & "G" & "]"
CmdCtrl(1) = "[" & "2M" & Chr(TetaPas(3)) & Chr(TetaPas(2)) & SensRot(2) & "G" & "]"
```

```
Robot.ComBras.Output = CmdCtrl(0)
FlagMoteurBase = True
'temporisation
'Robot.MsgRecu.Text = "tempo"
'Tempo (0.1)
'Robot.ComBras.Output = CmdCtrl(1)
FlagMoteurBras = True

End Sub

Public Sub TransfertParametres()
Dim a As String
Dim b As String
'charger les valeurs
If IsNumeric(Robot.AutreParam(0).Text) Then
   Speed = Robot.AutreParam(0).Text
   Else
   Robot.AutreParam(0).Text = 0
End If
If IsNumeric(Robot.AutreParam(1).Text) Then
   Debounce = Robot.AutreParam(1).Text
   Else
   Robot.AutreParam(1).Text = 0
End If

If Debounce > 10 Then Robot.ComBras.Output = "[R" & Chr(Debounce) & "xxxxxxx]"

'If Speed > 6 Then
a = "[V" & ChrW(Speed) & "xxxxxxx]"
b = Chr(Speed)

   Robot.ComBras.Output = "[V" & Chr(Speed) & "xxxxxxx]"
   'Tempo (5)
   'Robot.ComBras.Output = "[2V" & Chr(Speed) & "xxx]"
```

```
'End If



End Sub

Public Sub InitMoteur()
   Robot.ComBras.Output = "[Ixxxxxxxx]"
   'Tempo (5)
   'Robot.ComBras.Output = "[2Ixxxx]"

End Sub
Public Sub Tempo(Pause)
Rem sous programme pour une temporisation, pause est la durée d'attente en ms
Start = Timer
Do While Timer < Start + Pause
DoEvents
Loop
End Sub
```

Annexe 7- Programme de l'interface :

```
Public Teta(3) As Double    'Rotation Moteur 1, 2 et 3
Public SaveTeta(3) As Double  'sauvegarde des rotation moteur
Public TetaPas(3) As Integer   'rotation des moteurs en nombre de pas
Public Lgr(2) As Double  'Longueur des bras 1, 2 et 3
Public PsObj(2) As Double  'Position X, Y, Z de l'objet
Public TlObj As Double   'Taille de l'objet
Public RyPince As Double   'Rayon de la pince
Public CmdCtrl(1) As String 'contient les paramétres à envoyer au moteurs
Public SensRot(3) As String  'sens de rotation des moteurs
Public FlagMoteurBase As Boolean 'indique sue le moteur base a deja tourne
Public FlagMoteurBras As Boolean 'indique sue le moteur bras a deja tourne
Public LedBaseOn As Boolean
Public LedBras1On As Boolean
Public LedPinceOn As Boolean
Public LedBras2On As Boolean
Public CmdLed   As String   'contient la commande des leds
'Public CmdLed2   As String   'contient la commande des leds
Public Debounce As Integer   'contient la valeur de l'anti rebond
Public Speed    As Double   'contient la vitesse

'Public OnLineOn As Boolean

Public a As Double


Sub Main()
InitVar
LedBaseOn = False
LedBras1On = False
LedBras2On = False
LedPinceOn = False
CmdLed = "[L0000xxxx]"
'CmdLed2 = "[L00xx]"
```

```
'initialisation port serie
'Ouvrir Port LED
Robot.ComBras.CommPort = 16
Robot.ComBras.Settings = "9600,N,8,1"
Robot.ComBras.RThreshold = 1
Robot.ComBras.SThreshold = 1
Robot.ComBras.InputLen = 0
Robot.ComBras.InputMode = comInputModeText

If Not Robot.ComBras.PortOpen Then Robot.ComBras.PortOpen = True

'Robot.RotationMoteur(0).Text = Robot.AnimBras.Width
'Robot.RotationMoteur(1).Text = Robot.AnimBras.Height
Robot.Show
End Sub

Public Sub InitVar()
'initialisation des variables
   For i = 0 To 2
      Teta(i) = 0
      SaveTeta(i) = 0
      Lgr(i) = 0
      PsObj(i) = 0
Robot.LongueurBras(i).Text = 0
      Robot.Objet_XYZ(i).Text = 0
Robot.RotationMoteur(i).Text = 0
   Next i
   Teta(3) = 0
   SaveTeta(3) = 0
   Robot.Lgr_Obj.Text = 0
   Robot.Ry_Pince.Text = 0
TlObj = 0
   RyPince = 0
```

```
    FlagMoteurBras = False
    FlagMoteurBase = False

End Sub
Public Sub CalculRotation()
'calcul des rotations des différents moteurs
    'charger les valeurs vers les variables en les verifiant si numeriques
For i = 0 To 2
        If IsNumeric(Robot.LongueurBras(i).Text) Then
            Lgr(i) = Robot.LongueurBras(i).Text
            Lgr(i) = Lgr(i) / 1000

        Else
            Robot.LongueurBras(i).Text = 0
        End If

        If IsNumeric(Robot.Objet_XYZ(i).Text) Then
            PsObj(i) = Robot.Objet_XYZ(i).Text
            PsObj(i) = PsObj(i) / 1000
        Else
            Robot.Objet_XYZ(i).Text = 0
        End If
    Next i

    If IsNumeric(Robot.Lgr_Obj.Text) Then
        TlObj = Robot.Lgr_Obj.Text
        TlObj = TlObj / 1000
    Else
        Robot.Lgr_Obj.Text = 0
    End If

        If IsNumeric(Robot.Ry_Pince.Text) Then
        RyPince = Robot.Ry_Pince.Text
        RyPince = RyPince / 1000
    Else
```

```
        Robot.Lgr_Obj.Text = 0
    End If

      'si un paramétres a été remis à zero pour cause d'erreur numérique alors sortir

    For i = 0 To 2
        If Robot.LongueurBras(i).Text = 0 Then Exit Sub
        If Robot.Objet_XYZ(i).Text = 0 Then Exit Sub
      Next i
      If Robot.Lgr_Obj.Text = 0 Then Exit Sub
      If Robot.Ry_Pince.Text = 0 Then Exit Sub

    'calculer moteur 1
      Teta(0) = Atn(PsObj(0) / PsObj(1))

      'calculer moteur 3
      Teta(2) = (PsObj(0) ^ 2) + (PsObj(1) ^ 2) + ((PsObj(2) - Lgr(0)) ^ 2) - (Lgr(1) ^ 2) -
(Lgr(2) ^ 2)
      Teta(2) = Teta(2) / (2 * Lgr(1) * Lgr(2))
      Teta(2) = Atn(-Teta(2) / Sqr(-Teta(2) * Teta(2) + 1)) + 2 * Atn(1)
    'Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
    'calculer moteur 2
      Teta(1) = Lgr(1) + (Lgr(2) * Cos(Teta(2)) * (PsObj(2) - Lgr(0)))
    Teta(1) = Teta(1) - (Lgr(2) * Sin(Teta(2)) * (PsObj(1) / (Cos(Teta(0)))))
    Teta(1) = Teta(1) / ((Lgr(1) + (Lgr(2) * Cos(Teta(2)))) + (Lgr(2) * Sin(Teta(2))))
      'calculer arccos de Teta(1)
      Teta(1) = Atn(-Teta(1) / Sqr(-Teta(1) * Teta(1) + 1)) + 2 * Atn(1)

      'Calculer la rotation du moteur pince
    Teta(3) = (180 - (TlObj * (180 / RyPince))) / 2


      Exit Sub

    End Sub
```

```
Public Sub FormatData()
   'formattage des données

For i = 0 To 2
        Teta(i) = ((Teta(i) * 180) / 3.14128) 'en °
TetaPas(i) = Teta(i) / 1.8 'en pas
TetaPas(i) = Abs(TetaPas(i))
        If FlagMoteurBase And i < 2 Then TetaPas(i) = TetaPas(i) + 1
        If FlagMoteurBras And i > 1 Then TetaPas(i) = TetaPas(i) + 1
        'Teta(i) = Teta(i) - SaveTeta(i)
        If (Teta(i) - SaveTeta(i)) <= 180 Then SensRot(i) = "G" Else SensRot(i) = "D"
      Next i
      TetaPas(3) = Teta(3) / 1.8
      TetaPas(3) = Abs(TetaPas(3))

   'formatter teta pince

End Sub
Public Sub TransfertCoordonnees()
'format message = STX, 1, M, Pas M1, Pas M2, Sens M1, Sens M2, ETX
TetaPas(0) = 12
TetaPas(1) = 12
TetaPas(2) = 20
TetaPas(3) = 20
SensRot(0) = "G"
SensRot(1) = "G"
SensRot(2) = "G"



CmdCtrl(0) = "[" & "M" & Chr(TetaPas(0)) & Chr(TetaPas(1)) & SensRot(0) & SensRot(1) & Chr(TetaPas(2)) & Chr(TetaPas(3)) & SensRot(2) & "G" & "]"
CmdCtrl(1) = "[" & "2M" & Chr(TetaPas(3)) & Chr(TetaPas(2)) & SensRot(2) & "G" & "]"

Robot.ComBras.Output = CmdCtrl(0)
```

```
FlagMoteurBase = True
'temporisation
'Robot.MsgRecu.Text = "tempo"
'Tempo (0.1)
'Robot.ComBras.Output = CmdCtrl(1)
FlagMoteurBras = True

End Sub

Public Sub TransfertParametres()
Dim a As String
Dim b As String
'charger les valeurs
If IsNumeric(Robot.AutreParam(0).Text) Then
   Speed = Robot.AutreParam(0).Text
   Else
   Robot.AutreParam(0).Text = 0
End If
If IsNumeric(Robot.AutreParam(1).Text) Then
   Debounce = Robot.AutreParam(1).Text
   Else
   Robot.AutreParam(1).Text = 0
End If

If Debounce > 10 Then Robot.ComBras.Output = "[R" & Chr(Debounce) & "xxxxxxx]"

'If Speed > 6 Then
a = "[V" & ChrW(Speed) & "xxxxxxx]"
b = Chr(Speed)

   Robot.ComBras.Output = "[V" & Chr(Speed) & "xxxxxxx]"
   'Tempo (5)
   'Robot.ComBras.Output = "[2V" & Chr(Speed) & "xxx]"
'End If
```

```
End Sub

Public Sub InitMoteur()
   Robot.ComBras.Output = "[Ixxxxxxxx]"
   'Tempo (5)
   'Robot.ComBras.Output = "[2Ixxxx]"

End Sub
Public Sub Tempo(Pause)
Rem sous programme pour une temporisation, pause est la durée d'attente en ms
Start = Timer
Do While Timer < Start + Pause
DoEvents
Loop
End Sub
```

ملخص

يحتوي هذا العمل على اربعة فصول
الفصل الاول نبذة شاملة حول الروبوتات الصناعية
الفصل الثاني تقديم محركات و ملتقطات الروبوت
الفصل الثالث اختيار نظام التحكم وتعيين عناصر دارة التحكم
الفصل الرابع الادوات المستعملة في التصميم
الفصل الخامس تصميم و انجاز يد المعالجة باستعمال **PIC 16F877A**

Abstract

This these contain four parts

- ❖ The objective in the first part is general approach about industrial robots
- ❖ The second part presents the actuators and sensors in robotics
- ❖ The third part is about choosing the microcontroller
- ❖ The fourth presents the tools used
- ❖ The fifth is the realization of the robotic arm

Résumé

Le travail abordé dans ce mémoire comporte quatre parties :

- ❖ L'objet de la première partie est une généralité sur les robots industriels.
- ❖ La seconde Présente les actionneurs et capteurs en robotique.
- ❖ La troisième partie est le choix du composant principal pour la commande du bras manipulateur.
- ❖ La quatrième partie consiste à présenter les outils de conception
- ❖ La cinquième la réalisation du bras manipulateur