République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Bejaia

Faculté de Technologie

Département de GENIE ELECTRIQUE

Mémoire de fin d'étude

En Vue de l'obtention du diplôme de Master en Electrotechnique

Option : AUTOMATISMES INDUSTRIELS

Thème

Système de positionnement 2D/3D :

Réalisation et Commande

Présenté par :

- ✓ MIOUCHE Sofiane
- ✓ SERIKMA Yacine

Promoteur:

Mr MELAHI Ahmed

Promotion : 2013 - 2014

Remerciements

Nous remercions le bon dieu, le tout puissant de nous avoir accordé santé, force et courage afin d'Arrivés au terme de ce modeste travail.
Nous tenons à exprimer notre profonde gratitude à notre Promoteur « Mr : MELAHI. Ahmed » pour son aide et suivi pour l'élaboration de ce travail.
Nos remerciements s'adressent aux membres du jury d'avoir accepté d'évaluer notre travail.

Nous tenons également, à remercier tout le corps enseignant du département Génie électrique qui a contribué à notre formation.

En fin, nous tenons de remercier vivement tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Dédicaces

Au nom du tout puissant

Je dédie ce travail de fin d'étude aux êtres les plus chères ;
Mon père et ma mère qui ont fait de moi ce qui je suis aujourd'hui et qui ont veillé de guider mes pas durant toute ma vie par leurs aides, leur grands émotions et leur sacrifice.
A mes frères et sœurs
A ma famille
A mon binôme YACINE et sa famille
A tous mes amis (es)

A toutes les personnes qui m'ont encouragé tout au long de mon cursus.

Spéciales dédicaces pour le H211

A tous les êtres chers dont le soutient m'a été indispensable

Sofiane

Dédicaces

Au nom du tout puissant

Je dédie ce travail de fin d'étude aux êtres les plus chères ;
Mon père et ma mère qui ont fait de moi ce qui je suis aujourd'hui et qui ont veillé de guider mes pas durant toute ma vie par leurs aides, leur grands émotions et leur sacrifice.
A mes frères et sœurs
A ma famille
A mon binôme SOFIANE et sa famille
A tous mes amis (es)

A toutes les personnes qui m'ont encouragé tout au long de mon cursus.

Spéciales dédicaces pour le H211, B16, G105

A tous les êtres chers dont le soutient m'a été indispensable

Yacine

Sommaire

Chapitre III : Réalisation et premier tests

Chapitre IV : Programmation

Chapitre V : Tests finaux et validation

Liste des figures

Liste des Tableaux

INTRODUCTION GENERALE

Introduction générale :

En raison de leurs performances, les systèmes de positionnement sont considérés, de nos jours, comme une solution intéressante dans plusieurs applications. En effet, ces mécanismes pourraient atteindre de meilleures performances dynamiques en raison de leurs rigidités structurelles.

Dans les applications industrielles, l'espace de travail est une caractéristique importante d'un système de positionnement. Il représente un critère de performance dans l'explication de ces systèmes, ce qui rend la synthèse géométrique d'un mécanisme dans un espace de travail requis.

C'est au denier siècle que les systèmes de positionnement ont amorcés l'explosion des thèmes de recherche. A cette époque les systèmes du positionnement étaient conçus en respectant les contraintes imposées par le milieu industriel, comme la répétition, la précision dans la réalisation des taches, le respect des cadences de production, etc.

Les mouvements devant êtres effectués par les systèmes de positionnement pour réaliser les tâches désirées exigent plus de précision et des vitesses parfois variables. Ceci à conduit à la mise au point d'algorithmes de commande sophistiqués. Parmi ces algorithmes, on peut citer les algorithmes basés sur les modèles dynamiques.

Les modèles dynamiques des systèmes de positionnement sont exprimés en termes de paramètres (masse, inertie, frottement…) et géométriques des segments qui les constituent. Ces paramètres dépendent de la géométrie des segments qui sont généralement complexes et de la structure d'outil mouvant (actionneur, capteur…), ceci rend leurs mesures physiques avant assemblages non précises ou impossibles.

Etant donné que les tâches à exécuter sont naturellement définies dans notre espace (espace opérationnel), ce qui signifie la description en fonction du temps de la position et de l'orientation de la commande en position d'un outil mouvant, dont l'application sera faite sur un système de deux degrés de liberté.

Dans notre mémoire, on s'intéresse au système de positionnement 2D/3D, par l'étude technique du système, et la réalisation d'une carte commande qui pilote ce système.

Pour cela on a réparti ce travail en cinq chapitres :

- Dans le premier chapitre, on présente l'état de l'art pour les systèmes de positionnement.

- Le deuxième chapitre montre différentes trajectoires pour le balayage d'une surface ensuite l'élaboration d'une analyse fonctionnelle du système de positionnement.

- Le troisième chapitre est dédié aux tests des composants de la carte de commande avec un enchainement de réalisation.

- Quand au quatrième chapitre, il est réservé à la programmation du système (contrôle et trajectoires), ainsi qu'à la manière dont on peut développer le programme étape par étape.

- Dans le cinquième chapitre, on procède à la validation de notre projet réalisé, par un programme final et le schéma de simulation sur ISIS PROTEUS.

CHAPITRE 1 :
SYSTEME DE POSITIONNEMENT ETAT DE L'ART

I.1. Introduction:

Déplacer les éléments d'un système de positionnement d'un point à un autre le plus rapidement possible est une des principales fonctions de tout système de positionnement.

L'utilisation des systèmes de positionnement est aujourd'hui couramment envisagée pour l'automatisation de nombreuses tâches. Celles-ci sont particulièrement diversifiées : le nettoyage, le transport dans les ateliers automatisés, écriture, lecture…

Nous présentons dans ce chapitre le système de positionnement à développer et nous introduisons son principe de fonctionnement.

I.2. Système de positionnement à 3D :

I.2.1. Degrés de liberté :

Un système libre, situé dans un espace à trois dimensions, peut réaliser six mouvements. A chaque mouvement, est associé un degré de liberté. Un degré de liberté traduit donc une liberté de mouvement (rotation ou translation). Une pièce libre dans l'espace dispose de 6 degrés de liberté : 3 rotations (Rx, Ry, Rz) et 3 translations (Tx, Ty, Tz) **[1].**

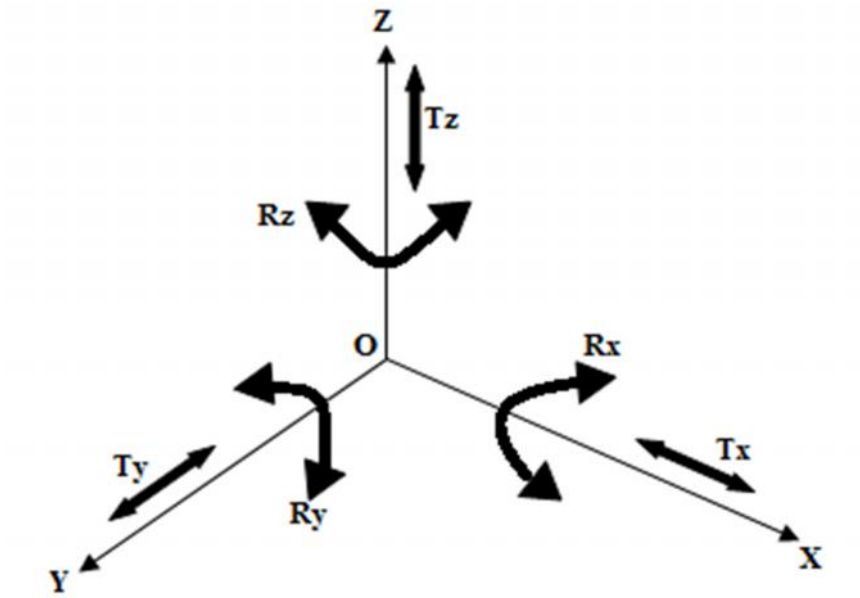


Figure I.1 Mouvement, associé un degré de liberté dans un système à 3D

I.2.2. Technologie et fonction d'un produit de positionnement en chaines fonctionnelles :

La figure suivante représente la structure générale d'un système de positionnement :

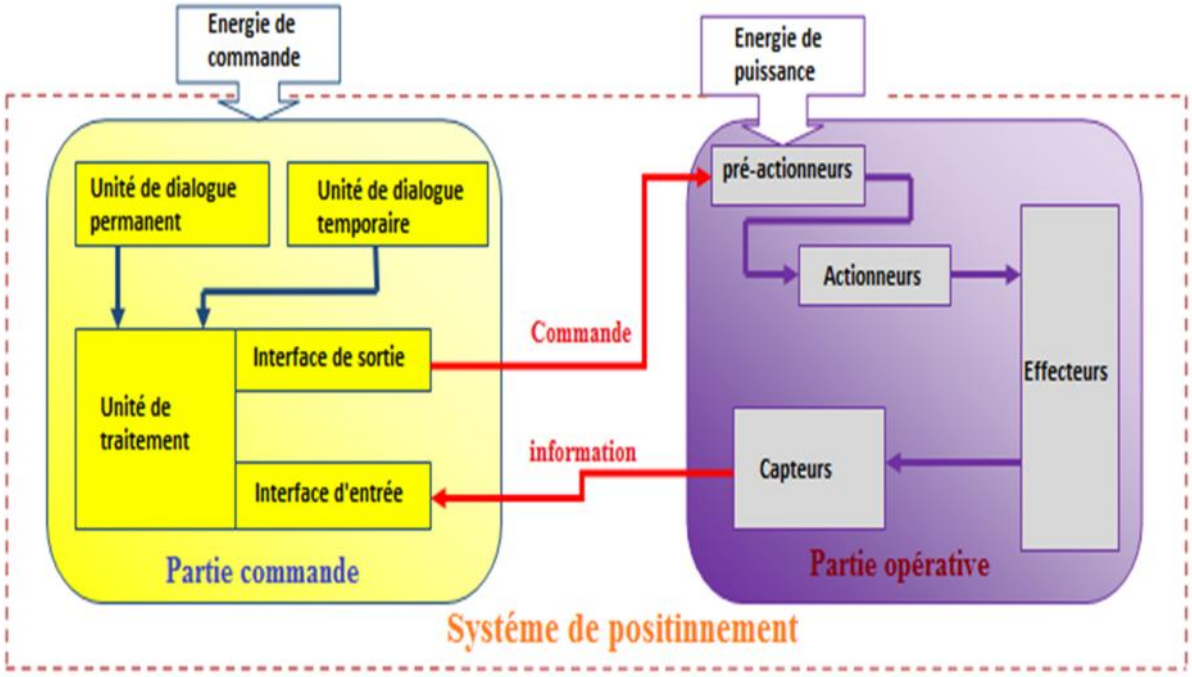


Figure I.2 Structure d'un système de positionnement

Les systèmes de positionnement possèdent une structure de base identique. Ils sont constitués de plusieurs parties reliées entre elles.

A. Une partie opérative :

C'est la partie visible du système de positionnement. Elle comporte l'élément mécanique du mécanisme avec :

A.1. Les pré-actionneurs :

C'est les éléments qui reçoivent des ordres venant de la partie commande, généralement sous forme de signaux électriques, dans le but de commander les actionneurs. **[2]**
Les pré-actionneurs utilisés dans les systèmes de positionnement sont :

- **Distributeur :**

Il est chargé d'alimenter les vérins en énergie pneumatique ou hydraulique dans le cas d'un système de positionnement fort puissance (ex : grue).

- **Contacteur :**

Il est un pré-actionneur destiné à ouvrir ou fermer un circuit électrique par l'intermédiaire d'un circuit de commande. Il alimente le moteur électrique en énergie de puissance en fonction d'une consigne opérative issue de la partie commande.

A.2. Les actionneurs :

Ils sont des éléments chargés de convertir de l'énergie pneumatique, hydraulique, ou électrique en énergie mécanique, afin de l'adapter au besoin de la partie opérative. Les actionneurs principaux utilisés dans les systèmes de positionnement sont :

- **Vérin :**

Il fait partie des actionneurs pneumatique ou hydraulique, l'énergie mécanique est produite sous forme d'un mouvement permettant un déplacement ou crée une force.

- **Moteur :**

Les moteurs électriques sont des actionneurs chargés de transformer l'énergie électrique en énergie mécanique de rotation. Le mouvement de rotation à l'intérieur d'un moteur est engendré grâce à des phénomènes magnétiques **[3].**

A.3. Les capteurs :

Un capteur est un dispositif soumis à l'action d'une grandeur physique, il informe la partie commande de l'exécution du travail. Ce signal est pneumatique mais dans la grand majorité des cas cette information se fait par l'intermédiaire d'un signale électrique. La famille utilisée dans les systèmes de positionnement est celle des capteurs de positions.

B. La partie commande :

Ce secteur de l'automatisme gère selon une suite logique le déroulement ordonnée des opérations à réaliser. Il reçoit des informations en provenance des capteurs de la partie opérative, et les restitue vers cette même partie opérative en direction des pré-actionneurs et actionneurs **[4].**

C. La partie relation :

Sa complexité dépend de l'importance du système. Elle regroupe les différentes commandes nécessaires au bon fonctionnement du procédé, c'est-à-dire marche/arrêt.

I.3. Modélisation du système de positionnement :

Pour concevoir, simuler ou commander un système de positionnement, il est nécessaire de disposer d'un modèle du mécanisme. Plusieurs niveaux de modélisation sont utilisés

I.3.1. Modélisation géométrique :

Dans la modélisation géométrique, on s'intéresse à la configuration géométrique du système sans tenir compte des forces qui la provoque. La géométrie du système et donnée par ses paramètres géométriques : positions et orientations de ses composants.

La modélisation des systèmes de positionnement de façon systématique exige une méthode adéquate pour la description de leurs structures. Plusieurs méthodes et notions ont été proposées telles que : la convention de Denvit-Hartenberg (D-H) et celle de Khalil-Kleifinger (K-K). Ces méthodes sont les plus utilisées en système de positionnement pour la définition de l'orientation et de la position des différents éléments d'un système mécanique articulé**[5].**

I.3.2. Modélisation cinématique :

Le modèle cinématique est, littéralement, un modèle des vitesses. Il exprime les relations entres les vitesses articulaires de chaque articulation et les vitesses cartésiennes d'un point de la chaîne cinématique. Ce modèle est donc un modèle par accroissement élémentaires : chaque variation élémentaire de la grandeur d'une articulation implique une variation de position de l'organe terminale et inversement **[6].**

Lorsque ces variations infinitésimales sont exprimées par rapport au temps, on peut les considérés comme des vitesses.

Le modèle cinématique permet donc non seulement de compléter éventuellement le modèle géométrique en tenant compte des vitesses, mais aussi de remplacer le modèle géométrique. En agissant par accroissements successifs, on peut se déplacer d'un pont donné à un autre.

I.3.3. Modélisation dynamique :

Afin de pouvoir commander ou simuler les systèmes de positionnement, il est nécessaire d'établir le modèle dynamique du système, c'est-à-dire les équations liant les forces exercées par les actionneurs et ceux dues aux interactions avec l'environnement aux déplacements des axes. On obtient ainsi un système d'équations différentielles non linéaires **[6].**

L'ensemble des équations dynamiques peuvent êtres déterminées à partir des lois de la mécanique classique newtonienne ou lagrangienne. Les approches d'Euler-Lagrange et de Newton-Euler permettent d'aboutir aux équations du mouvement du système de positionnement.

I.4. Génération des trajectoires :

I.4.1. Génération de la trajectoire dans l'espace articulaire :

La génération de mouvements dans l'espace articulaire applique à chaque articulation du système de positionnement une loi de mouvement dont les contraintes sont définies dans l'espace articulaire.

Dans le cas le plus fréquent où la durée du mouvement n'est pas imposée, chaque articulation a une durée propre de mouvement déduite de la satisfaction des contraintes cinématiques ou dynamiques. Il est alors nécessaire, dans une seconde étape, de synchroniser l'ensemble des articulations sur l'articulation maître. A l'issue de leur synchronisation, les mouvements recalculés sont directement les consignes de commande **[6].**

I.4.2. Mouvement point à point et mouvement à trajectoire continue :

De façon générale, la génération de mouvement construit la trajectoire de l'outil du robot à partir de la donnée des contraintes spatiales sous la forme de points intermédiaires qui sont soit de véritables ponts de passage, soit des points précisant les directions successives du mouvement grâce à la notion de polygone directeur. La génération de mouvement est alors envisagée selon deux modes **[6].**

- Un mode **''**point à point**''** qui impose l'arrêt du robot à chaque point intermédiaire. Il est souvent associé à une génération de mouvement dans l'espace articulaire.
- Un mode **''**trajectoire continue**''** qui impose une continuité de la vitesse, il est souvent associé à une génération de mouvement dans l'espace opérationnel.

I.4.2.1. Méthode de base :

Ces méthodes de base définissent le mouvement d'un point de départ x^d à un point d'arrivée x^a à l'aide d'une spline unique ou d'une composition de splines élémentaires qui définit soit un profil de vitesse spécifique, soit un profil d'accélération spécifique. Sachant que la vitesse est nulle aux points de départ est d'arrivée, la spline de plus faible degré assurant la continuité de la vitesse lors du mouvement est la spline cubique d'équation.

$$X(t)=at^3 + bt^2 + ct + d \ ,0 \leq t \leq T \qquad (I.1)$$

La variable T désigne la durée du mouvement, les quatre coefficients a, b, c, d sont déterminés à partir des quatre contraintes de définition de mouvement :

- Les contraintes initiales :

  $x(0)= x^d$ (I.2)

  $\dot{x}(0)=0$ (I.3)
- Les contraintes finales :

  $x(T) = x^a$ (I.4)

  $\dot{x}(T) = 0$ (I.5)

En combinant les équations (I.2), (I.3), (I.4) et (I.5) en aura :

$a=-2(x^a - x^d)/T^3$ (I.6)

$c=0$ (I.7)

$b=3(x^a - x^d)/T^2$ (I.8)

d=x^d (I.9)

il résulte de ce calcul que la spline cubique a un maximum de vitesse et un maximum d'accélération en valeur absolu donnés par :

$$\dot{x}(t)_{max}=3|x^a - x^d| /2T \quad (I.10)$$

$$\ddot{x}(t)_{max}=6|x^a - x^d| /T^2 \quad (I.11)$$

I.4.2.2. Mouvement point à point à profil de vitesse trapézoïdal :

Si l'on souhaite disposer d'une phase de vitesse constante, on considère le profil trapézoïdal de vitesse qui modifie le profil d'accélération nulle en lui intégrant une phase d'accélération nulle. La loi de mouvement est composée de trois splines successives d'ordre 2, 1 et 2, dont on détermine les huit coefficients à partir des huit contraintes de position et de vitesse initiales et finales. Le générateur de mouvement à profil trapézoïdal de vitesse est donné par le système suivant :

$$\begin{cases} x(t)=\frac{x^a-x^d}{2\tau(T-\tau)}t^2 + x^d, & 0 \le t \le \tau \\ x(t)=\frac{x^a-x^d}{T-\tau}\left(t-\frac{\tau}{2}\right) + x^d, & \tau \le t \le T-\tau \\ x(t)=\frac{x^a-x^d}{2\tau(T-\tau)}(t-T)^2+ x^a, & T-\tau \le t \le T \end{cases} \quad (I.12)$$

La variable T désigne la durée totale du mouvement et τ celle des phases d'accélération initiale et de freinage final.

Les expressions de vitesse et accélération maximales sont :

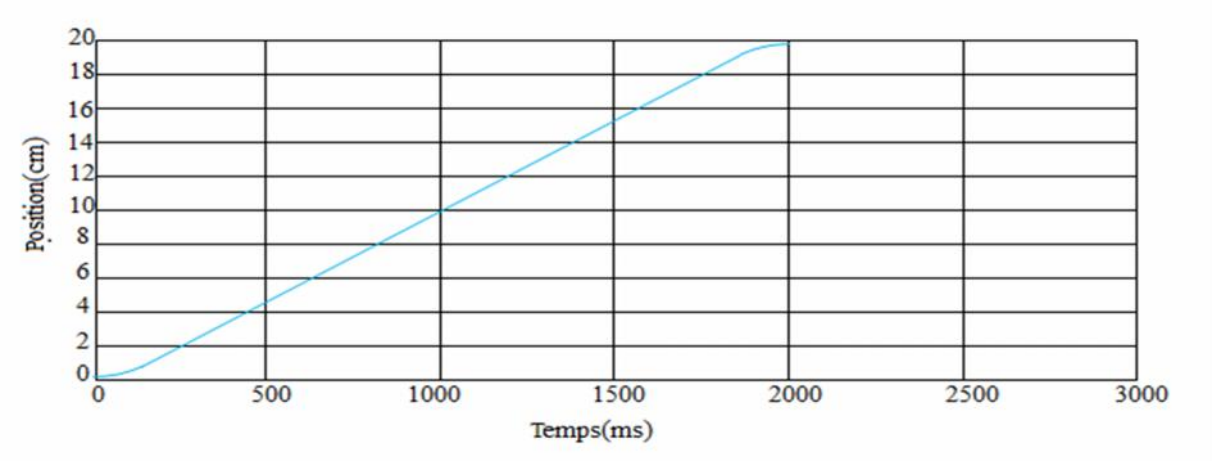
$$\dot{x}_{max}=\frac{|x^a-x^d|}{T-\tau} \quad (I.13)$$

$$\ddot{x}_{max}=\frac{|x^a-x^d|}{\tau(T-\tau)} \quad (I.14)$$

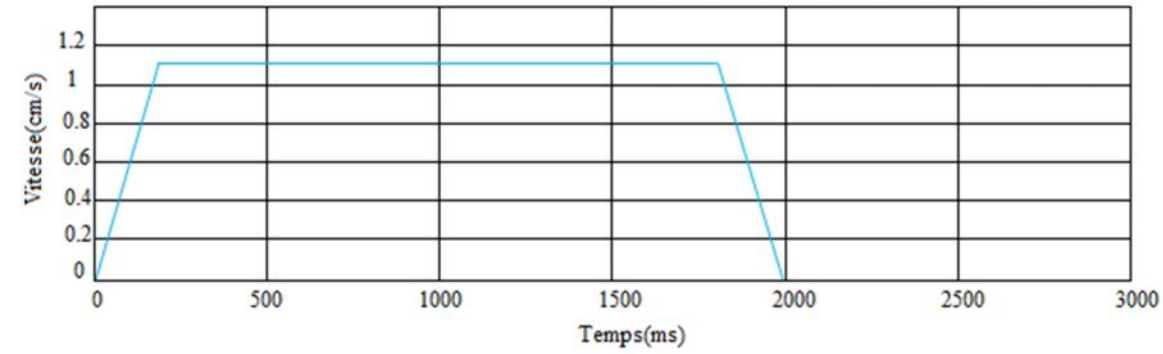
On détermine la durée optimale T réalisant la satisfaction des contraintes de vitesse $\dot{x}_{max}$ et d'accélération $\ddot{x}_{max}$ en considérant τ et T comme deux variable indépendantes, qui permettent d'assurer à la fois l'accélération $\ddot{x}_{max}$ et la vitesse de croisière $\dot{x}_{max}$ par les relations :

$$\tau = \frac{\dot{x}_{max}}{\ddot{x}_{max}} \quad (I.15)$$

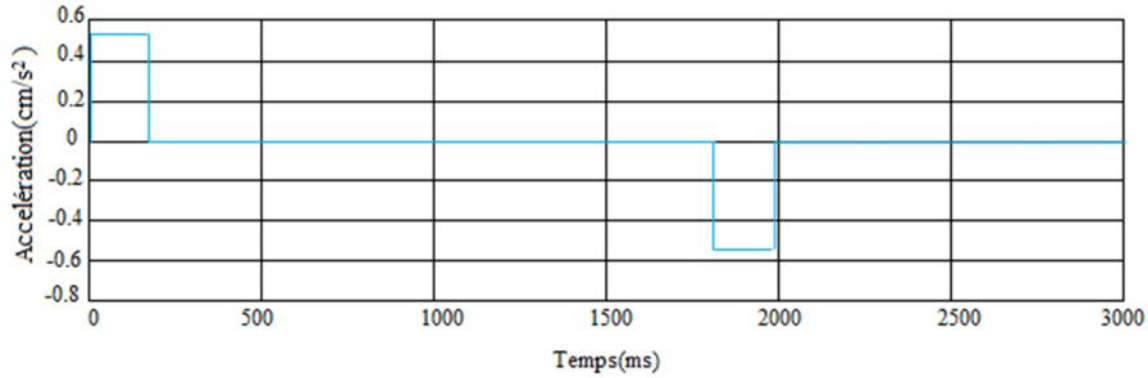
$$T=\frac{|x^a-x^d|}{\dot{x}_{max}}+\tau \quad (I.16)$$



(a) position



(b) vitesse



(c) accélération

Figure I.3 Mouvement point à point à profil de vitesse trapézoïdal.

I .5. Problématique :

L'objectif de notre problématique consiste à réaliser un système de positionnement 2D/3D qui fait le balayage d'une surface.

L'utilisation de ce système est orientée vers la mesure des distributions de charges électriques (électrets) sur une surface. Ces charges ont été préalablement déposées sur cette surface.

Pour faire le balayage on a besoin des trajectoires. Ces trajectoires peuvent être obtenues en utilisant des moteurs (MCC, MPP) qui fonctionnent selon deux axes (X et Y).

Plusieurs types de commandes de moteurs peuvent être envisagés. Les plus sophistiquée font appel à des circuits intégrés dédiés à ce type d'application. Parmi ces circuits intégrés on trouve les microprocesseurs, microcontrôleurs,…

I.6. Conclusion :

Ce premier chapitre, a permis de présenter brièvement les différentes notions nécessaires pour l'étude du système de positionnement et les types de modèles utilisés (modèle géométrique, modèle cinématique, modèle dynamique).

Ensuite nous avons vu la notion de génération de trajectoire qui permet de réaliser la tache désirée.

La problématique a été pose est une première analyse donnée, le détail est dans la suite du mémoire.

CHAPITRE II :

TRAJECTOIRES ET ANALYSE FONCTIONNELLE

II.1 Introduction :

La trajectoire est l'évolution de la position, et ses dérivés en fonction du temps pour chacune des articulations.

Le mouvement le plus simple est d'aller d'un point initial à un point final. Ce type de mouvement convient aux tâches de transfert d'objet quand l'espace de travail ne comporte aucun obstacle. Pour certaines raisons telles qu'éviter les obstacles, le chemin à suivre par l'élément terminal peut être contraint par l'addition des points intermédiaires aux configurations initiale et finale.

II.2. La génération des différentes trajectoires :

On utilise sept types de trajectoires de référence. Les trajectoires générées sont données dans le plan (x, y). Les deux moteurs ont pour but de faire un balayage.

II.2.1. Présentations des différentes trajectoires :

II.2.1.1. La première trajectoire :

La **figure (II.1)** montre la première trajectoire de référence. Les positions minimales et maximales sont :

- Pour l'articulation prismatique selon l'axe X : $x_{min} = 5cm;\ x_{max} = 15cm.$
- Pour l'articulation prismatique selon l'axe Y : $y_{min} = 0cm;\ y_{max} = 7cm.$

Le balayage se fait comme suit : d'abord les deux moteurs sont à la position initiale (15cm, 0cm), ensuite ils commencent à tracer des rectangles, et à chaque fois ils diminuent les dimensions du rectangle de 1mm à chaque extrémité pour arriver en fin à la position final.

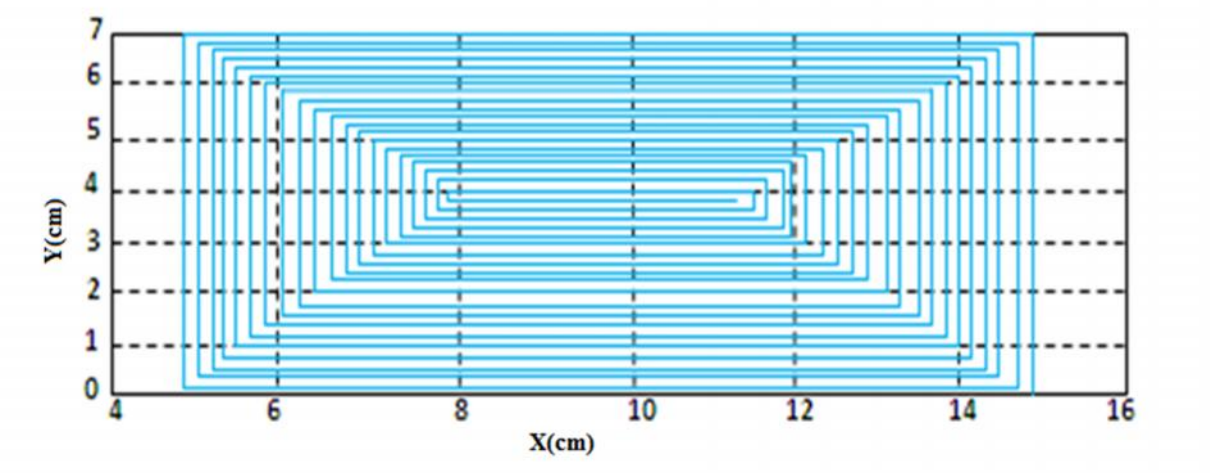


Figure II.1 Première trajectoire de consigne.

II.2.1.2. La deuxième trajectoire :

La **figure (II.2)** montre la deuxième trajectoire de référence. Les positions minimales et maximales pour les deux axes sont :

- Pour l'articulation prismatique selon l'axe X : $x_{min} = 0cm;\ x_{max} = 10cm.$
- Pour l'articulation prismatique selon l'axe Y : $y_{min} = 0cm;\ y_{max} = 7cm.$

Dans ce cas le balayage ce fait en va et vient à partir de la position initiale (0cm, 0cm) pour arriver à la position finale (10cm, 0cm).

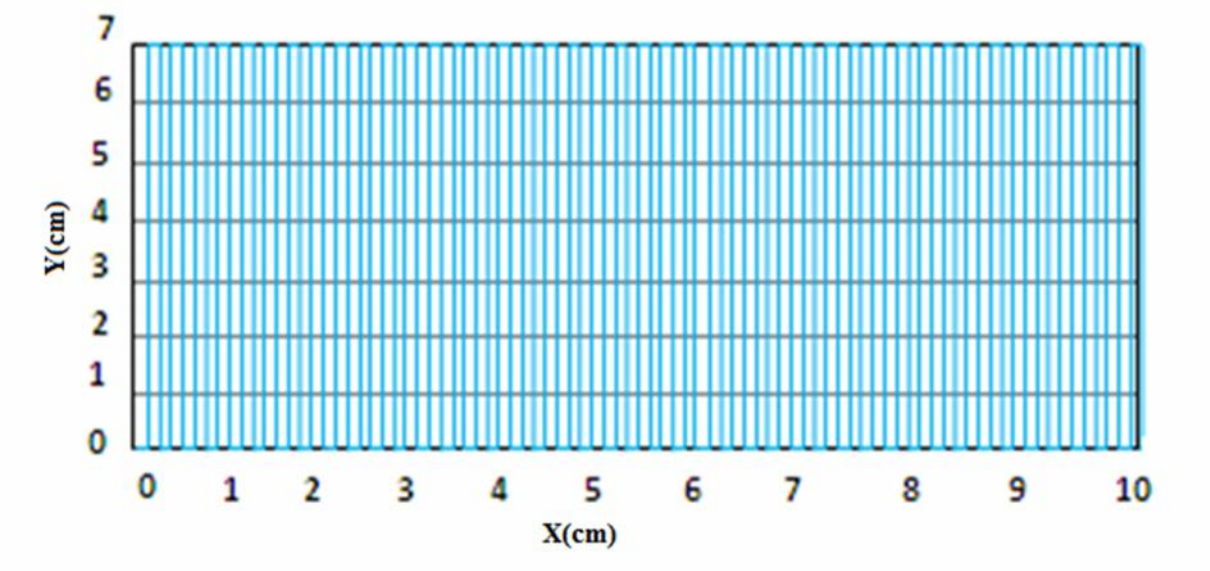


Figure II.2 Deuxième trajectoire de consigne.

II.2.1.3. La troisième trajectoire :

La **figure (II.3)** montre la troisième trajectoire de référence. Les positions minimales et maximales sont :

- Pour l'articulation prismatique selon l'axe X : $x_{min} = 0cm;\ x_{max} = 18cm.$
- Pour l'articulation prismatique selon l'axe Y : $y_{min} = 0cm;\ y_{max} = 16cm.$

Le balayage est décrit comme suit : les deux moteurs commencent ses opérations à partir de la position initiale (0cm, 0cm), puis ils commencent à tracer des lignes suivant l'axe X et à chaque fois ils montent de 1cm suivant l'axe Y pour arriver à la position finale (0cm, 0cm).

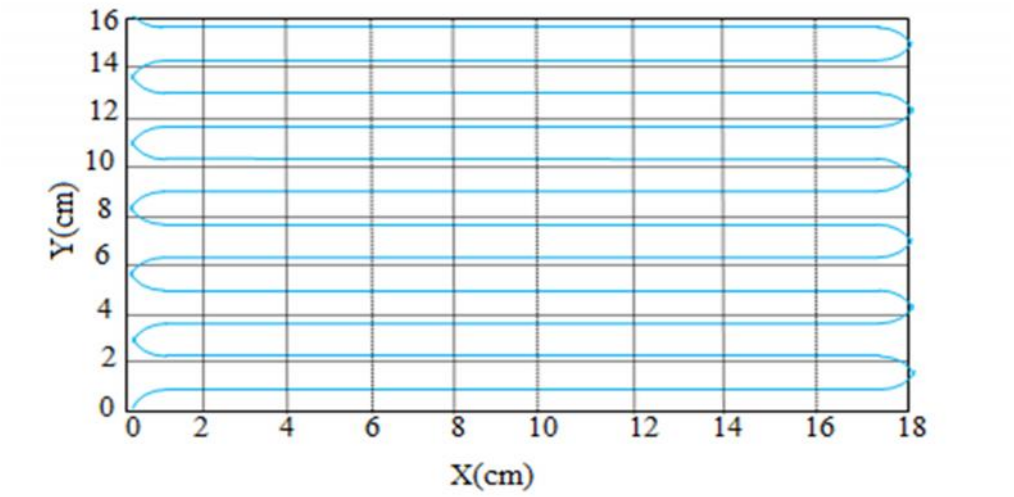


Figure II.3 Troisième trajectoire de consigne.

II.2.1.4. La quatrième trajectoire :

La **figure (II.4)** montre la quatrième trajectoire de référence. Les positions minimales et maximales sont :

- Pour l'articulation prismatique selon l'axe X : $x_{min} = 0cm;\ x_{max} = 20cm.$
- Pour l'articulation prismatique selon l'axe Y : $y_{min} = 0cm;\ y_{max} = 20cm.$

Les deux moteurs dessinent des cercles de même rayon 10cm et de centre (10cm, 10cm) parcouru plusieurs fois.

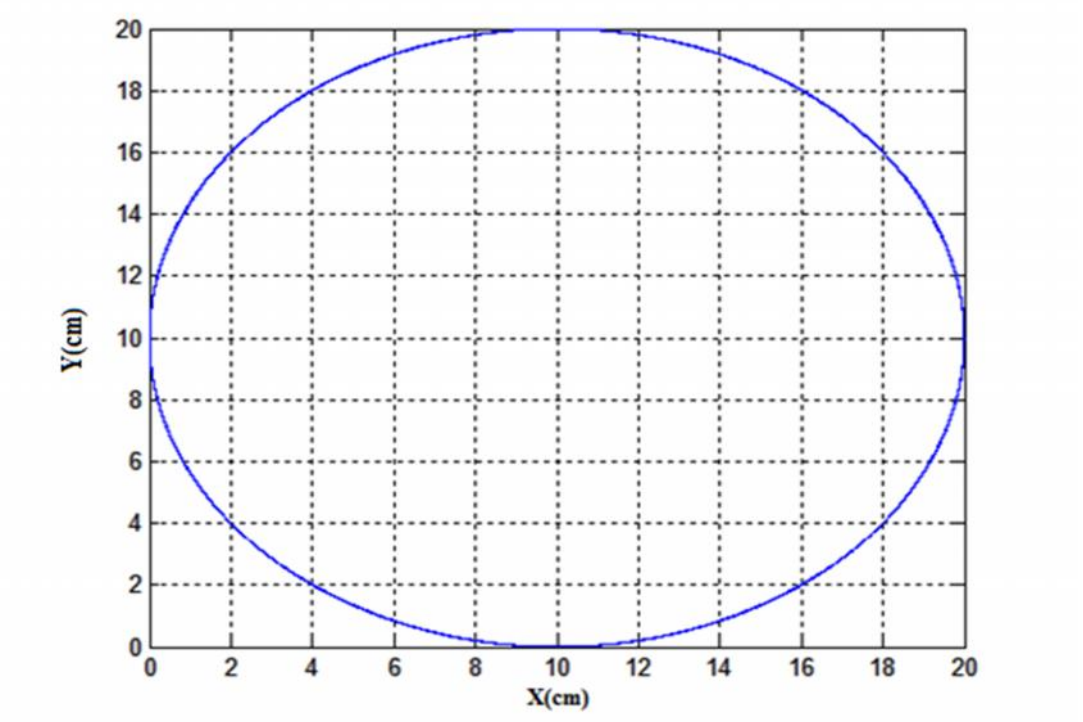


Figure II.4 Quatrième trajectoire de consigne.

II.2.1.5. La cinquième trajectoire :

La **figure (II.5)** montre la cinquième trajectoire de référence. Les positions minimales et maximales suivantes :

- Pour l'articulation prismatique selon l'axe X : $x_{min} = 0cm;\ x_{max} = 20cm.$
- Pour l'articulation prismatique selon l'axe Y : $y_{min} = 0cm;\ y_{max} = 20cm.$

La trajectoire est une série de cercles concentrique de centre (10cm, 10cm). Les deux moteurs commencent par le cercle extrême de rayon 10cm, et à chaque fois qu'il termine un cercle, ils passent au cercle suivant en diminuant le rayon de 1cm.

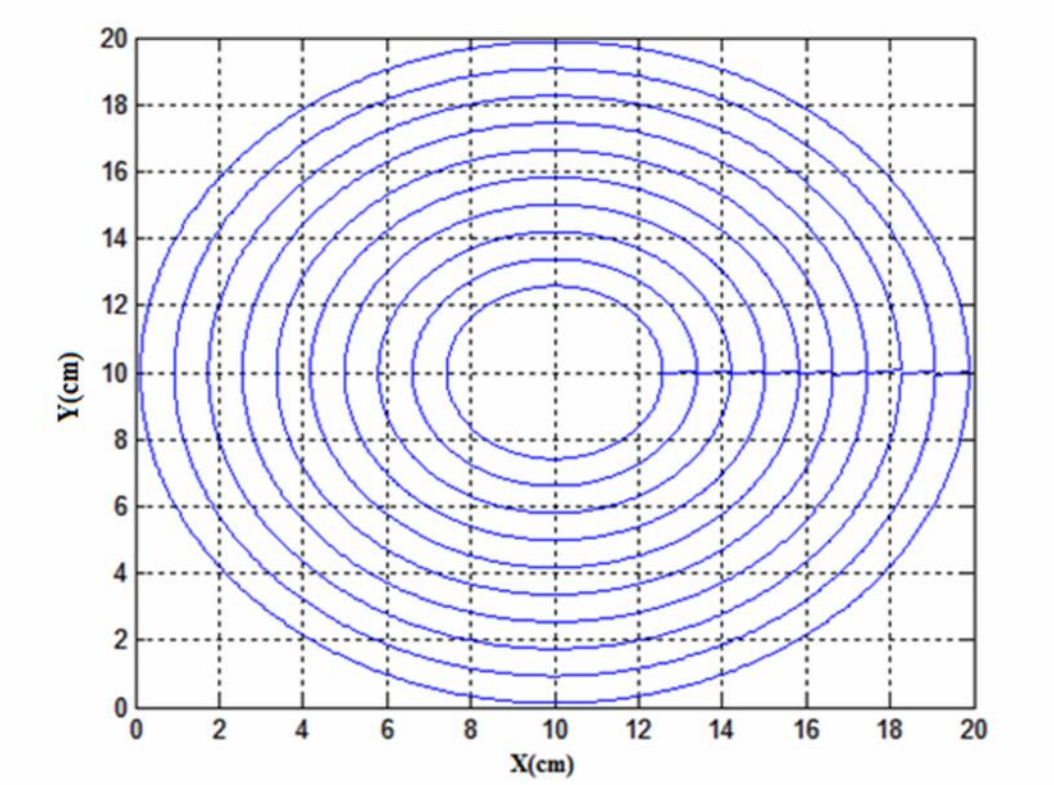


Figure II.5 Cinquième trajectoire de consigne.

II.2.1.6. La sixième trajectoire :

La **figure (II.6)** montre la sixième trajectoire de référence. Les positions minimales et maximales suivantes :

- Pour l'articulation prismatique selon l'axe X : $x_{min} = 5cm;\ x_{max} = 15cm.$
- Pour l'articulation prismatique selon l'axe Y : $y_{min} = 5cm;\ y_{max} = 15cm.$

Le balayage ce fait comme suit : les deux moteurs sont à la position initial (10cm, 10cm). Ensuite ils commencent à tracer une spirale à pas constant.

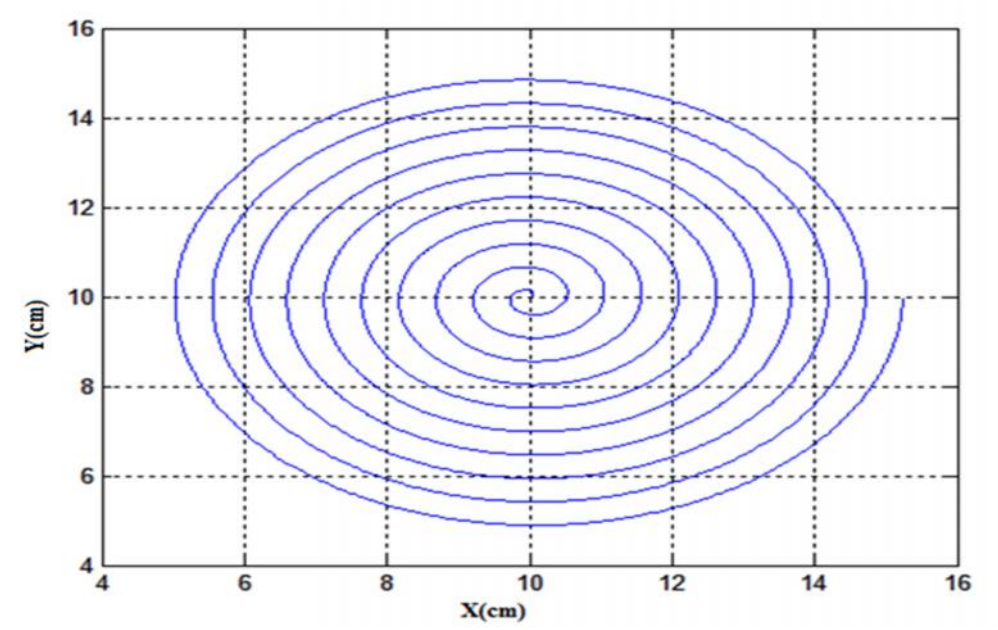


Figure II.6 Sixième trajectoire de consigne.

II.2.1.7. La septième trajectoire :

La **figure (II.7)** montre la septième trajectoire de référence. Les positions minimales et maximales suivantes :

- Pour l'articulation prismatique selon l'axe X : $x_{min} = 5cm;\ x_{max} = 15cm.$
- Pour l'articulation prismatique selon l'axe Y : $y_{min} = 5cm;\ y_{max} = 15cm.$

Le balayage ce fait comme suit : les deux moteurs sont à la position initial (11.5cm, 10cm). Ensuite ils commencent à tracer une spirale logarithmique.

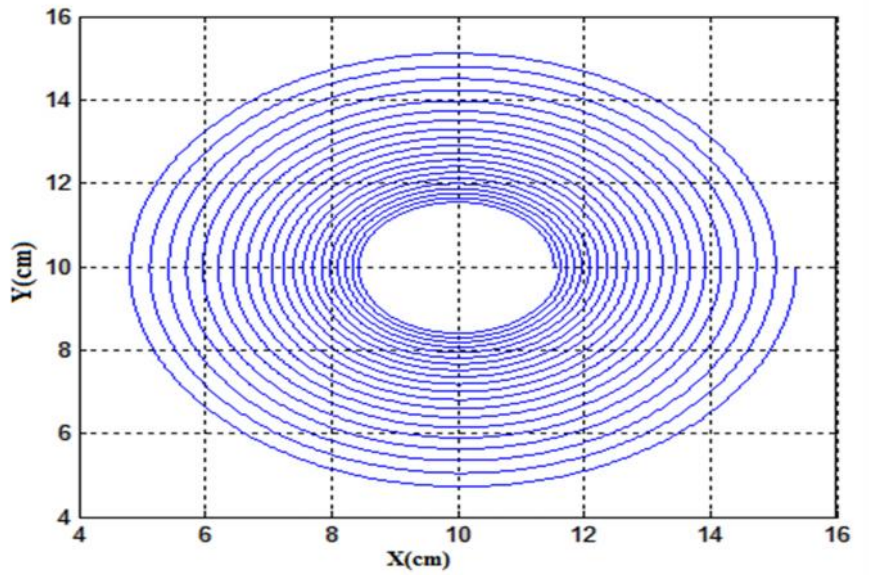


Figure II.7 Septième trajectoire de consigne.

II.3. Analyse fonctionnel :

L'analyse fonctionnelle est un outil performant pour recenser, caractériser, ordonner, hiérarchiser et valoriser les fonctions d'un produit. Elle permet d'avoir une vision claire des exigences attendues du produit. Ceci permet :

- D'aboutir sur un cahier des charges du produit attendu.
- De démarrer une analyse des risques afin d'éradiquer tout défaut potentiel engendrant une non-tenue des spécifications, avant que la conception ne soit figée et part la même avant que le retour en arrière ne coute trop cher.
- De démarrer une analyse de la valeur afin d'obtenir le meilleur rapport Qualité/prix.
- C'est-à-dire obtenir un produit qui ne répondre qu'aux spécifications demandées. Il ne sert à rien d'avoir un produit ou composants de produit ayant plus de fonctions que nécessaire car celles-ci auront un cout **[7].**

II.3.1. Structured Analysis and Design Technic (SADT):

A. Définition:

SADT est un langage multidisciplinaire, favorise la communication entre utilisateurs et concepteurs. Le concept est simple, basé sur un formalisme graphique et textuel facile. Permet de modéliser le problème, puis expose la solution et assure une communication efficace entre les différentes personnes concernées par le système, il existe 7 concepts fondamentaux :
1. SADT analyse un système : en construisant un modèle dont le but est d'exprimer une compréhension complète et de le situer dans son contexte. Plusieurs modèles selon diffèrent points de vue peuvent s'avérer nécessaires.
2. Analyse du système : descendante, hiérarchique, modulaire, structurée
3. SADT décrit le « QUOI » où la méthode est efficace et non le « COMMENT », diffèrent solutions envisagées pour la réalisation du QUOI.
4. SADT modélise à la fois les choses, données, objets, noms, produits, personnes, programmes, machines, et les événements, activités, verbes effectués par ces objets.
5. SADT est un langage semi-formel c'est-à-dire {texte + graphiques (boîtes, flèches)}
6. SADT favorise le travail en équipe discipliné et coordonné.
7. SADT oblige à consigner par écrit tous les choix effectués pendant l'analyse **[8].**

B. Langage SADT :

Le modèle SADT est un ensemble de diagrammes ordonnés hiérarchiquement. Au niveau le plus général : diagramme de niveau «A-1». Il montre le contexte du système à analyser : les sources et destinations des informations arrivant et sortant de la «boîte» à analyser, c'est à dire ses interfaces avec l'extérieur.
Au niveau inférieur, on montre juste la boîte principale (appelée boîte 0) dans un diagramme, appelé «A-0». Cette boîte se décompose elle même en un diagramme de niveau inférieur niveau «A0», qui contient les boîtes 1, 2, 3,4, ….i. Chaque boîte i se décompose elle même en un diagramme i1, i2, i3, i4, ….

Un diagramme peut être un actigrammes (activités : Ai) ou un datagramme (données: Di) **[8]**.

C. Schématisation :

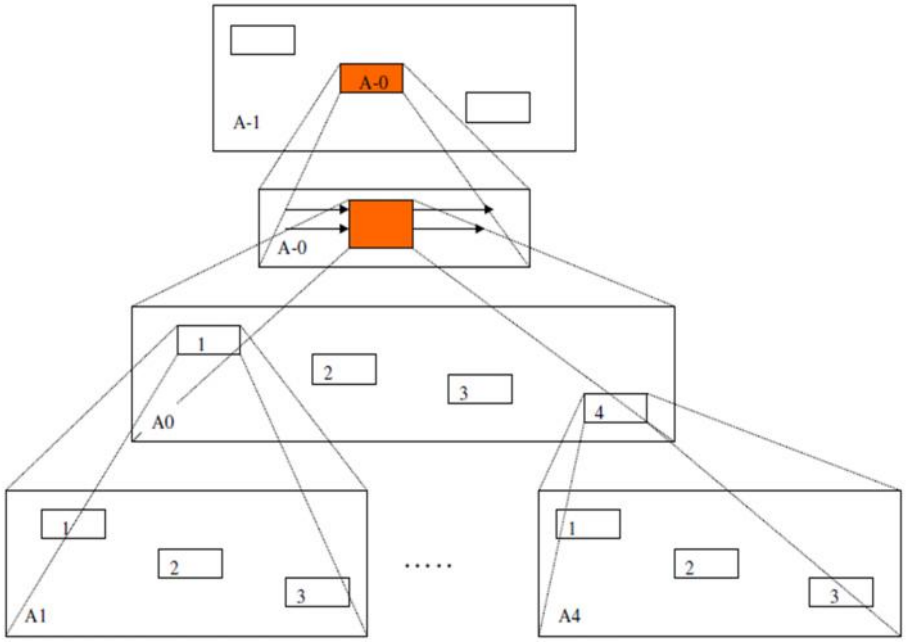
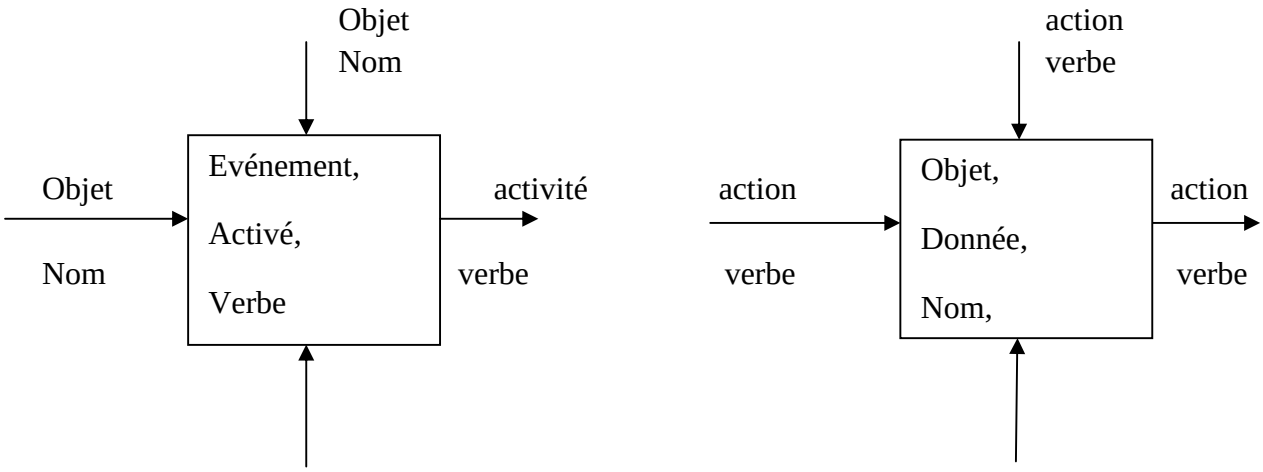


Figure II.8 Hiérarchisation des diagrammes SADT.

Chaque diagramme de niveau inférieur apporte un nombre limité de détails sur un sujet bien délimité. Une boîte est décomposée en un diagramme comportant entre 3 boîtes au minimum et 6 boîtes au maximum (sauf A1 ou D1 et quelques cas particuliers).
Selon SADT, le modèle est composé de données (objets, substantifs) et d'actions (événements activités, verbes). En SADT, il y a dualité entre activités et données **[8].**



La figure ci-dessous représente le SADT de notre système de positionnement :

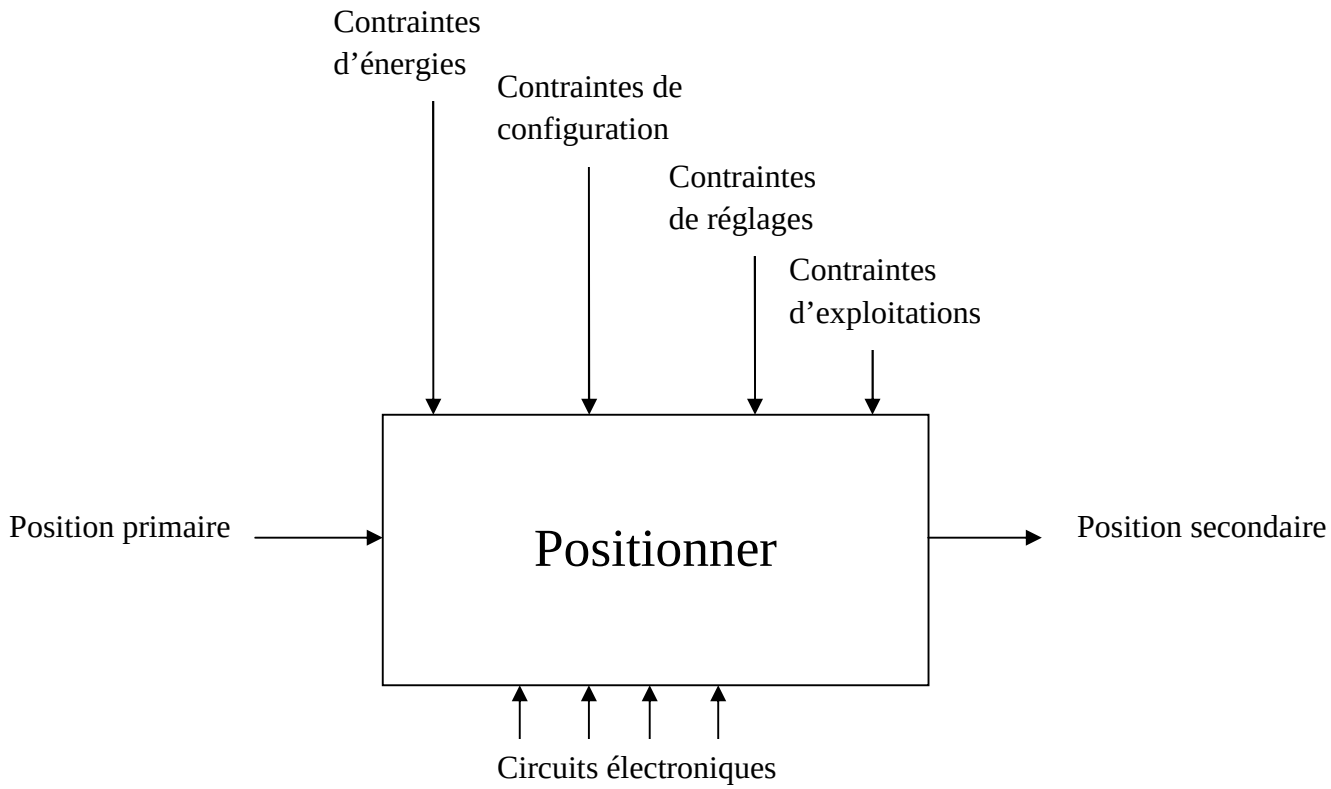


Figure II.9 Schématisation de notre SADT

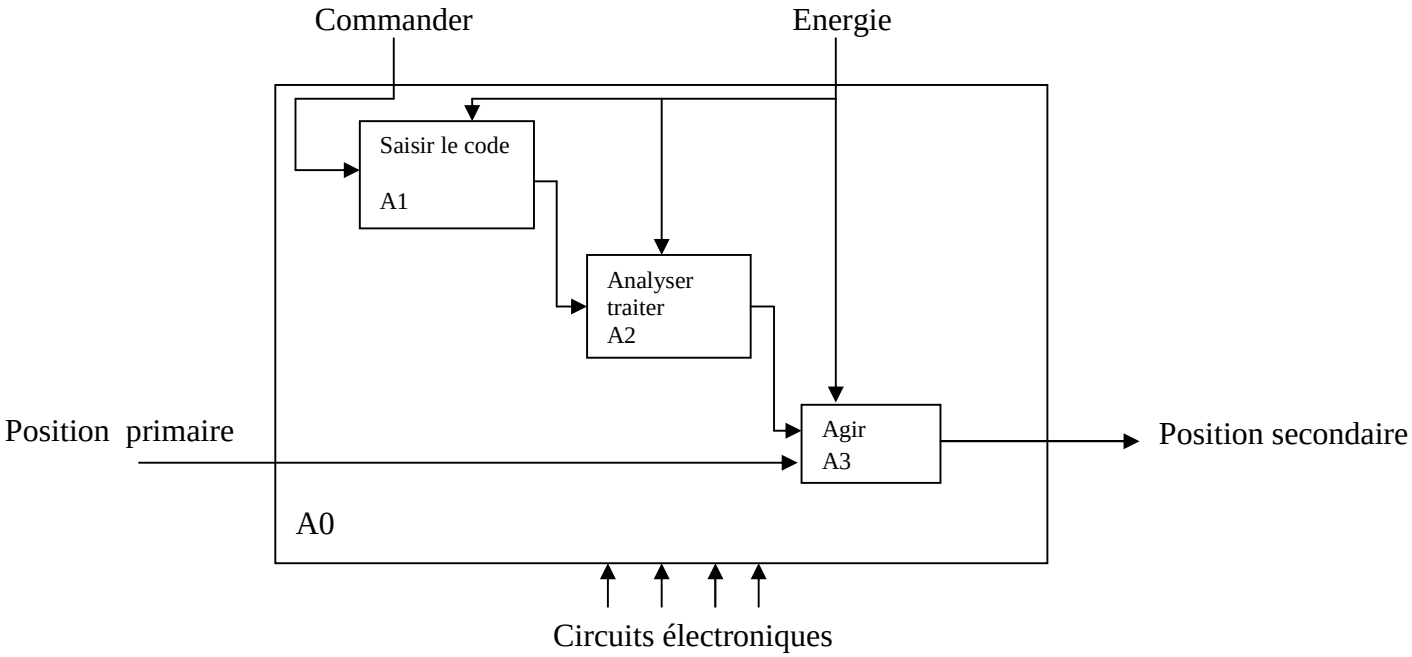


Figure II.10 Le SADT de notre système de positionnement

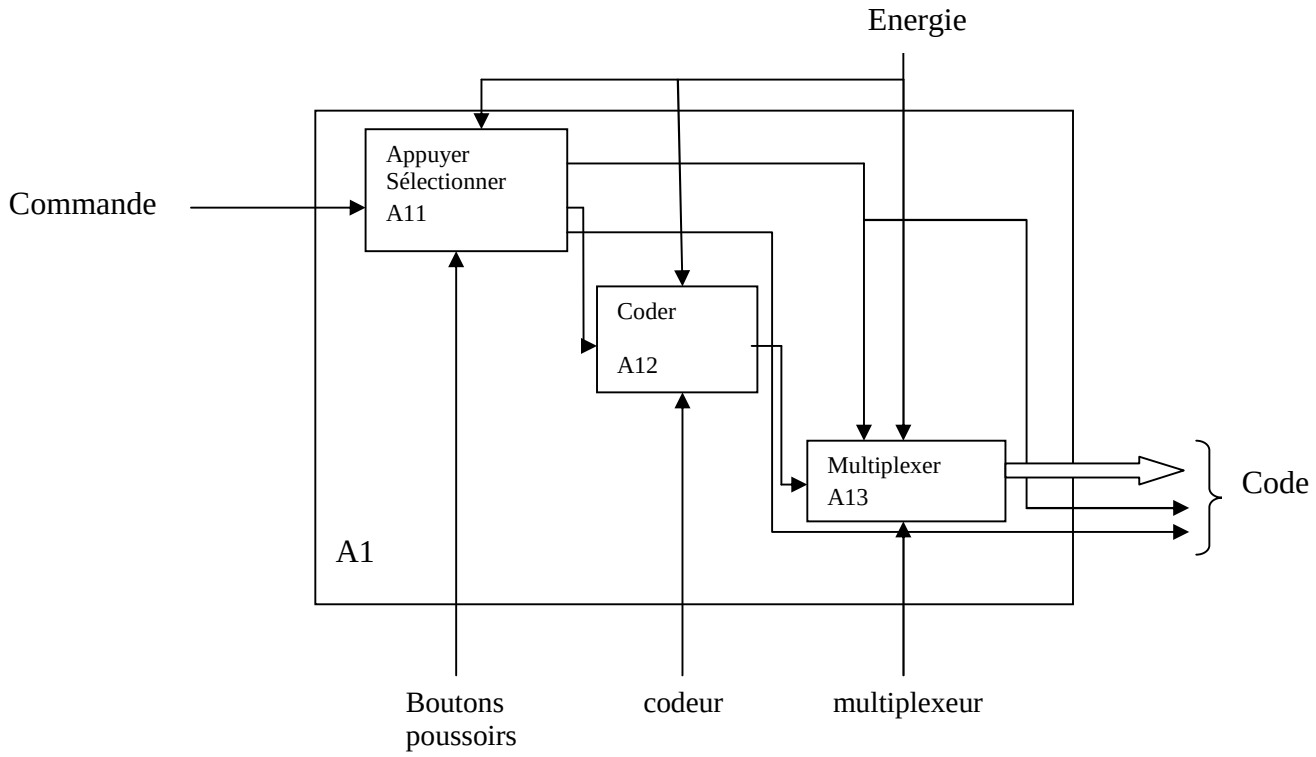


Figure II.11 Le SADT du premier actigrammes (saisir le code)

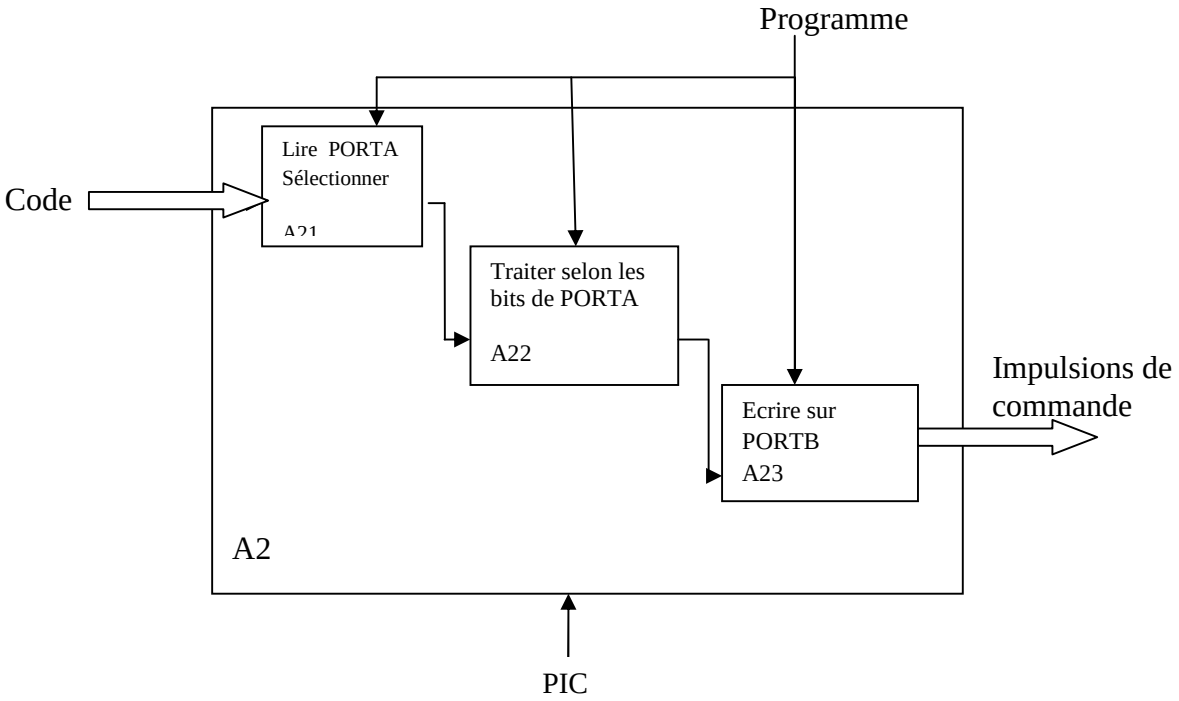


Figure II.12 Le SADT du deuxième actigrammes (analyser et traiter)

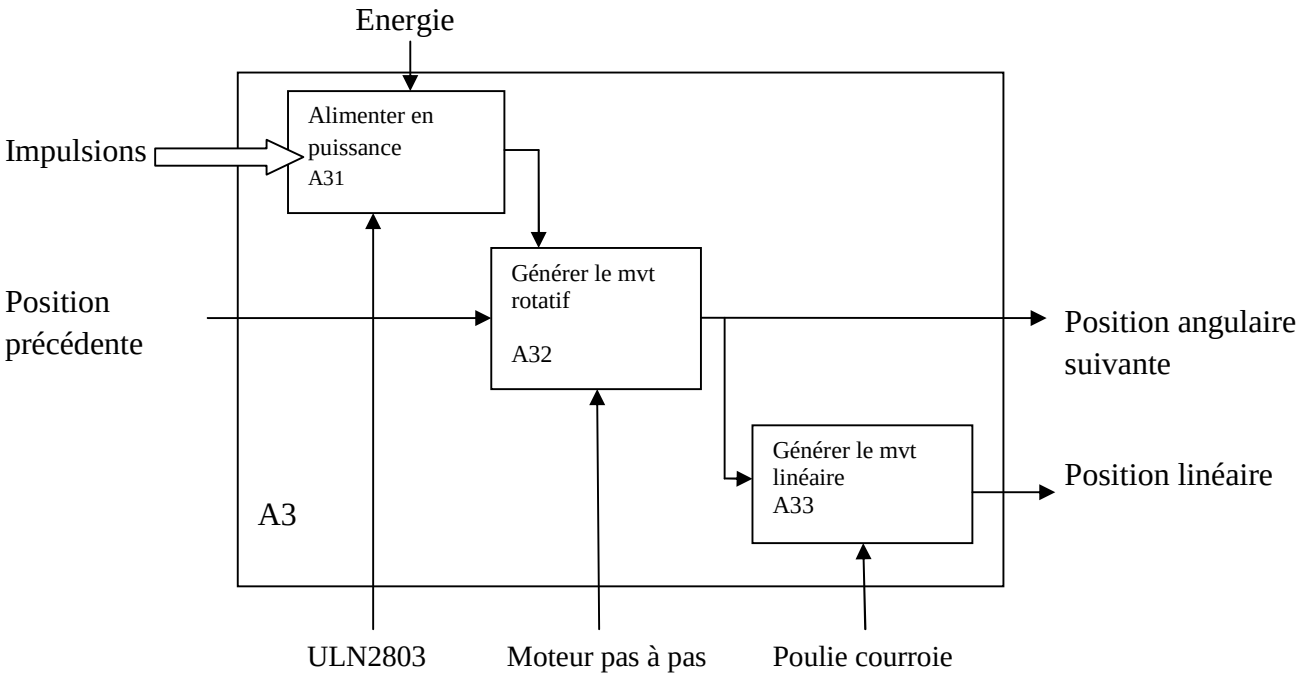


Figure II.13 Le SADT du troisième actigrammes (agir)

II.3.2. Function Analysis System Technique (FAST):

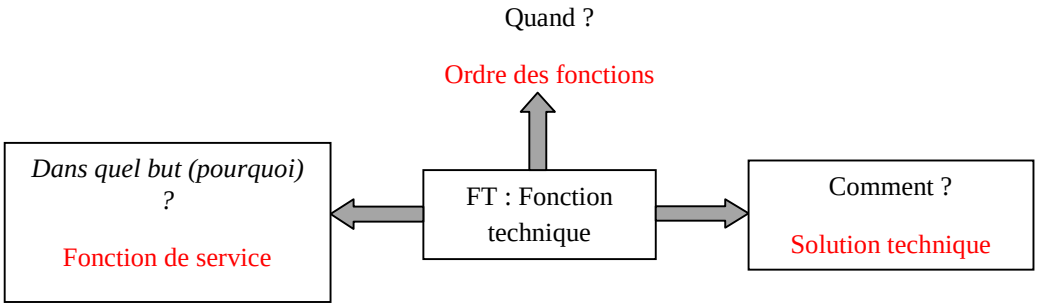
A. Définition :

La méthode FAST (Function Analysis System Technique) permet de relier et d'ordonner toutes les fonctions techniques assurées par les éléments du système.

Un diagramme FAST présente une traduction rigoureuse de chacune des fonctions de service en fonctions techniques, puis matériellement en solutions techniques. Il constitue alors un ensemble de données essentielles permettant d'avoir une bonne connaissance d'un système et ainsi de pouvoir améliorer la solution proposée.

Pour chaque fonction technique, la méthode FAST répond aux 3 questions :

1. Pourquoi ? Pourquoi une fonction doit-elle être assurée ? Accès à une fonction technique d'ordre supérieur, on y répond en lisant le diagramme de droite à gauche.
2. Comment ? Comment cette fonction doit-elle être assurée ? On décompose alors la fonction, et on peut lire la réponse à la question en parcourant le diagramme de gauche à droite.
3. Quand ? Quand cette fonction doit-elle être assurée? Recherche des simultanéités, qui sont alors représentées verticalement.



La figure ci-dessous représente le FAST de notre système de positionnement :

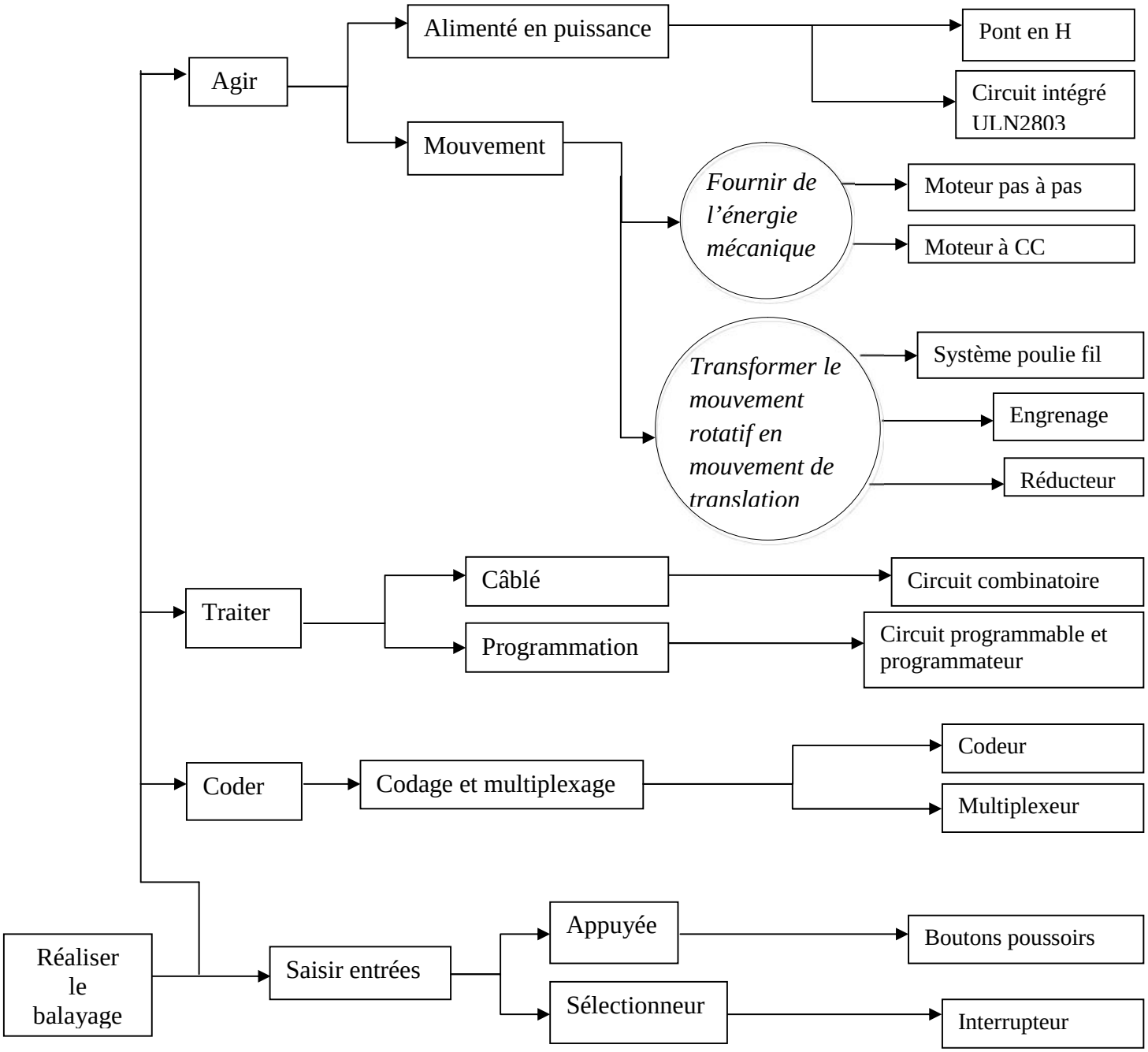


Figure II.14 Le FAST des systèmes de positionnement.

II.4. Conclusion :

Dans ce chapitre, nous avons effectué la présentation de différentes trajectoires de référence, suivie d'une analyse fonctionnelle technique pour le fonctionnement (SADT) et pour le choix du matériel (FAST). Ceci nous a permis de comprendre d'avantage le principe de fonctionnement du système et de proposer une solution technique permettant la réalisation de notre système de positionnement. En fait, cette solution peut être changée lors du développement et la réalisation du système qui fait l'objet des chapitres suivants.

CHAPITRE III :
REALISATION ET PREMIER TESTS

III.1 Introduction :

Ainsi, après quelques rappels et notions fondamentales sur les moteurs pas à pas, ce chapitre présente la réalisation de la carte de commande.
La réalisation est faite sur une plaquette d'essai d'une manière progressive. Dans ce chapitre on présente ce processus de réalisation progressif avec les tests nécessaires à chaque étape.

III.2. Test des composants de la carte de commande :

III.2.1 Identification du moteur pas à pas :

Le moteur pas à pas unipolaire utilisé possède 6 fils de différentes couleurs (rouge, bleu, jaune, noir, marron, blanc), et deux bobines ayant des points milieux (Voir l'annexe1).

La figure ci-dessous nous montre la photo réelle du moteur pas à pas :

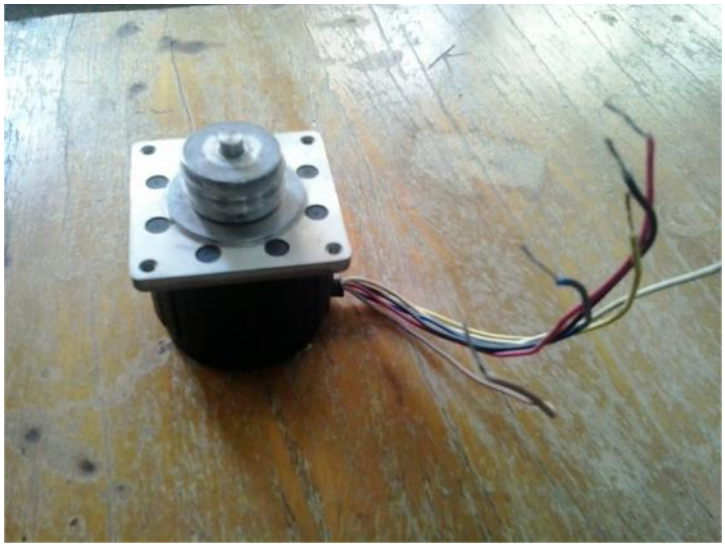


Figure III.1 Moteur pas à pas unipolaire

III.2.1.1 Identification des fils du moteur pas à pas :

Pour identifier les fils on a utilisé un multimètre, on a mesuré les résistances entre chaque deux fils, et on a obtenue les valeurs dans le tableau suivant :

Couleur	Jaune	Rouge	Bleu	Noir	Blanc	Marron
Jaune	0	5 Ω	10 Ω	∞	∞	∞
Rouge	5 Ω	0	5 Ω	∞	∞	∞
Bleu	10 Ω	5 Ω	0	∞	∞	∞
Noir	∞	∞	∞	0	5 Ω	10 Ω
Blanc	∞	∞	∞	5 Ω	0	5 Ω
Marron	∞	∞	∞	10 Ω	5 Ω	0

Tableau III.1 Différentes valeurs des résistances mesurées

D’après le tableau on a déduit que les fils de la première bobine sont de couleurs Jaune, Noir et Bleu. Les extrémités sont le jaune et le bleu, et le commun c’est le noir.

Les fils de la deuxième bobine sont de couleurs Rouge, Blanc et Marron. Le commun est le blanc, et les extrémités sont le rouge et le marron.

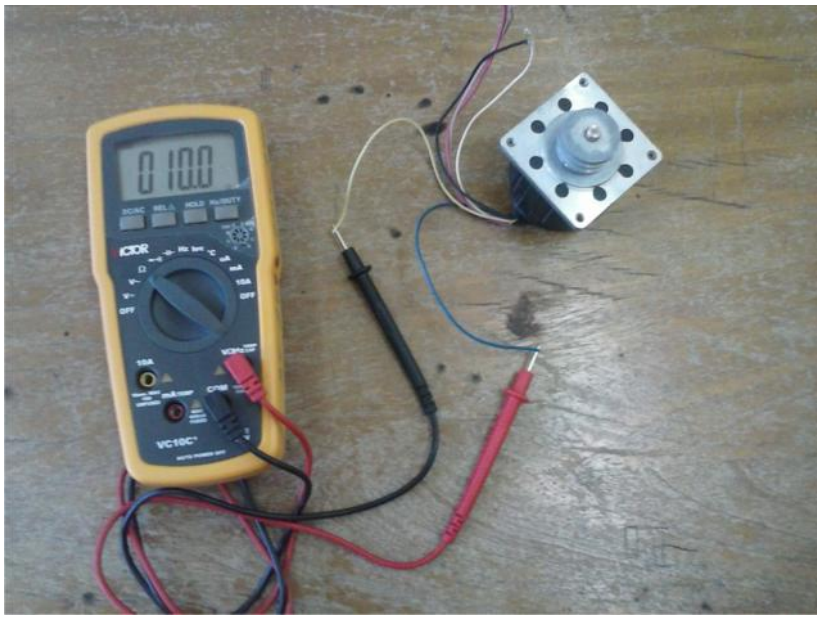


Figure III.2 Mesure des résistances

III.2.1.2. Rotation du moteur pas à pas :

Le montage ci contre consiste à identifier la succession des impulsions qui fait tourner le moteur dans un sens et dans l’autre.

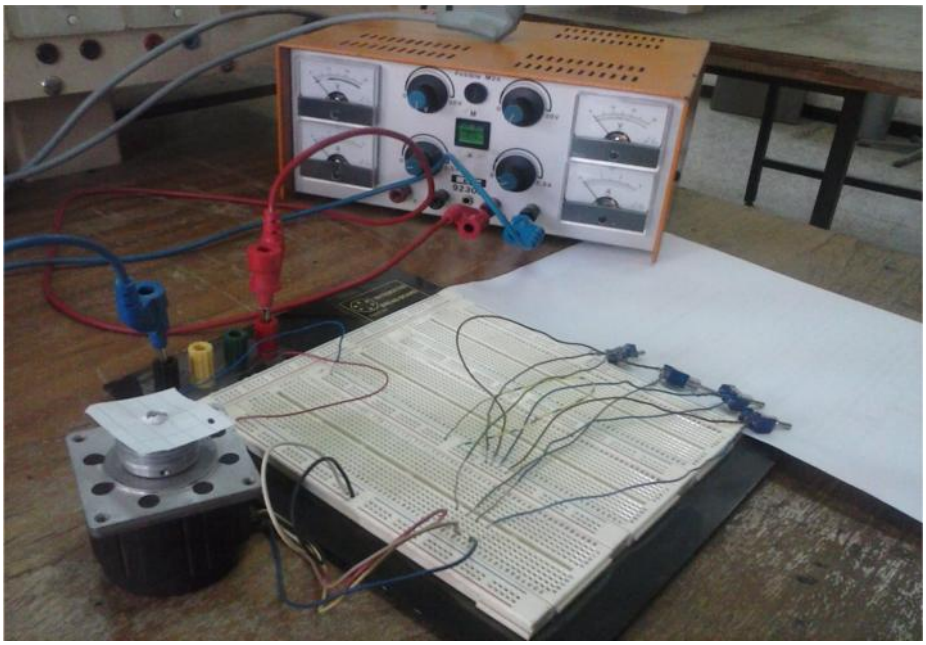


Figure III.3 Maquette d’essai de succession et sens.

Ce montage est équipé de :

- Une alimentation variable.
- Une plaquette d'essai.
- Un moteur pas à pas (unipolaire).
- Fils électriques.
- quatre interrupteurs.

Ce montage permet d'identifier la succession des impulsions et le sens de rotation de notre moteur pas à pas (unipolaire), à l'aide d'une feuille marqué avec un stylo, fixé sur l'arbre de ce dernier.

Donc, pour pouvoir identifier cette succession, il suffit d'introduire la commande à travers les quatre Interrupteurs.

Sens1 : la succession est 0001, 0010, 0100, 1000
Sens2 : la succession est 1000, 0100, 0010, 0001

III.2.2. Réalisation de l'alimentation stabilisée 5V :

L'alimentation stabilisée est constituée de deux condensateurs (0.33µF, 150nF) et un régulateur de tension LM7805. La figure suivante montre le circuit électronique de cette alimentation stabilisée.

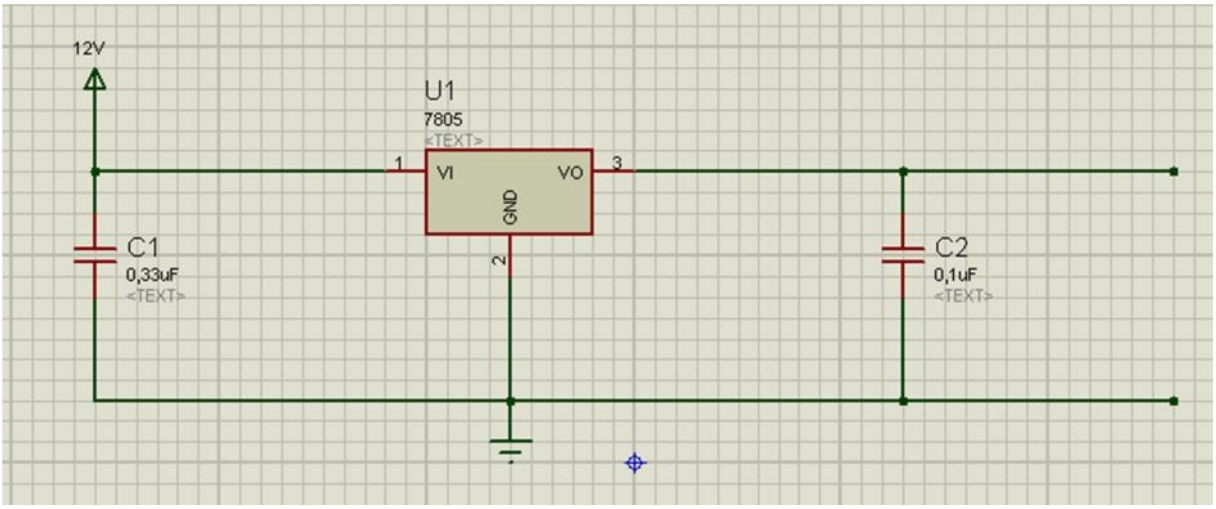


Figure III.4 Le schéma du circuit d'alimentation stabilisée

On alimente le circuit avec une tension continue supérieure à 5V (par exemple de 12V). On aura en sortie une tension stabilisée de 5V. C'est cette tension qu'on va utiliser dans toute la suite de notre réalisation.

La figure ci-contre nous montre la photo réelle du circuit qu'on a réalisé au laboratoire :

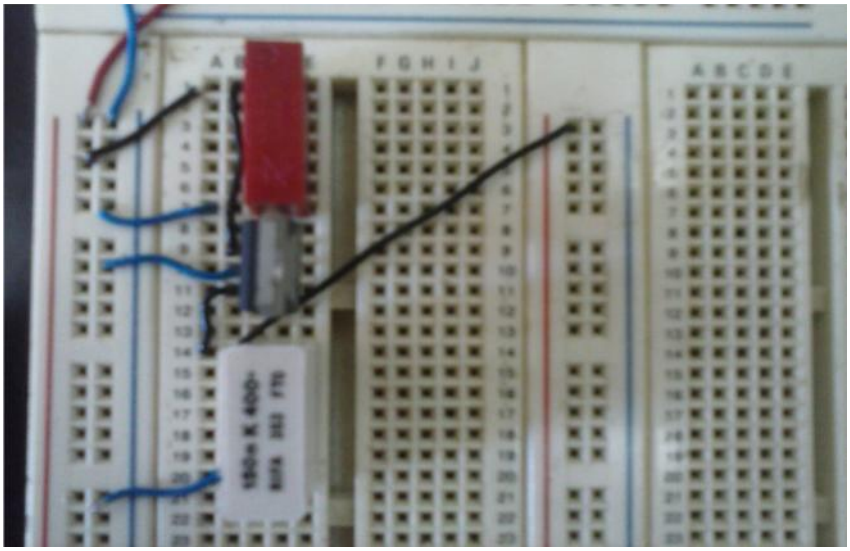


Figure III.5 Circuit de l'alimentation stabilisée

III.2.3. Test du circuit intégré ULN2803 :

Dans ce montage qu'on a réalisé sur la plaquette d'essai, l'allumage d'une LED (LED-red) à travers du circuit intégrée ULN2803 permet de tester (identifier, vérifier) les pins (les pattes, les broches) de ce dernier (Voir l'annexe2).

Le test se résume en l'alimentation de la LED à travers les pins (entrée/sortie). L'allumage de la LED montre le bon fonctionnement de la paire de pins testée.

Ce montage est commandé par un bouton poussoir avec une alimentation de 5V. Pour réduire le courant qui passe par la LED, on a introduit une résistance de 1KΩ en série avec la LED.

La figure ci-dessous montre le premier test :

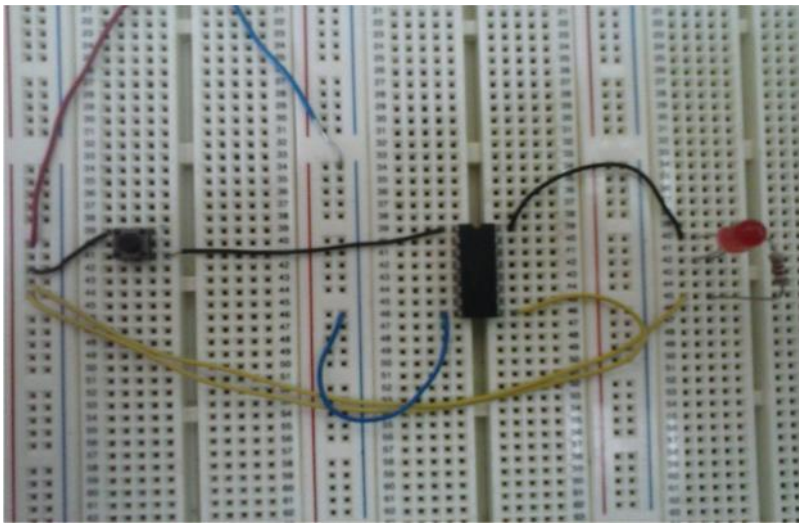


Figure III.6 Test de l'ULN2803.

III.2.4. Test d'un ULN2803 avec un seul moteur pas à pas :

D'après l'identification qu'on a faite sur le moteur pas à pas unipolaire et le fonctionnement du circuit intégrée ULN2803, on a pu réaliser la commande d'un seul moteur pas à pas.

Ce montage est commandé par quatre boutons poussoirs avec une alimentation de 5V. Pour minimiser le courant on a introduit des résistances de 1KΩ en série avec les boutons.

On clique sur les boutons poussoirs du haut vers le bas respectivement et le moteur pas à pas tourne dans un seul sens (sens1).

La figure ci-dessous nous montre le test :

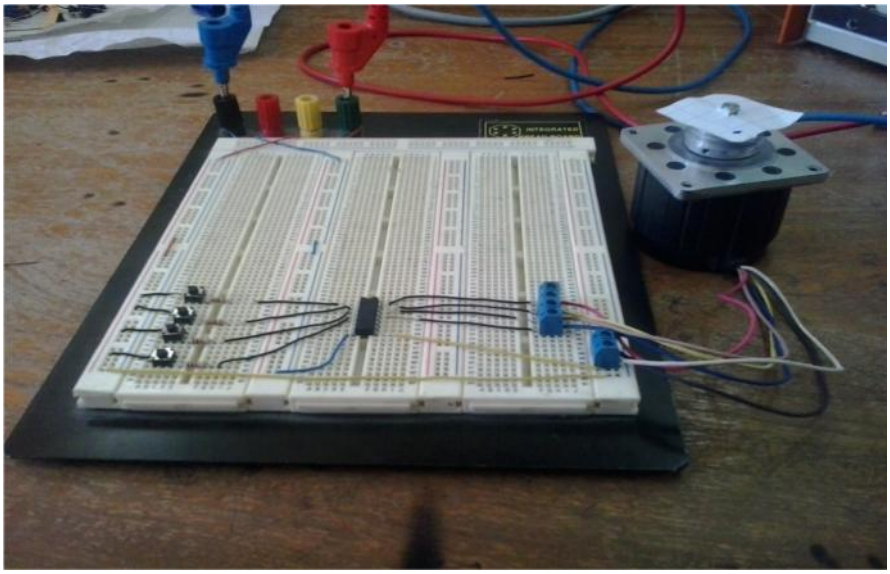


Figure III.7 Commande d'un seul moteur pas à pas

III.2.5. Test de L'ULN2803 avec deux moteurs pas à pas :

Après le test qu'on a fait sur un seul moteur pas à pas unipolaire, on passe à l'application avec un seul ULN2803 et deux moteurs pas à pas.

Ce montage est commandé par huit boutons poussoirs avec une alimentation de 5V. Pour minimiser le courant on a introduit des résistances de 1KΩ en série avec les boutons.

On clique sur les boutons poussoirs du haut vers le bas respectivement les deux moteurs pas à pas tournent successivement dans le même sens.

N.B. voir l'annexe2

La figure ci-dessous nous montre le test :

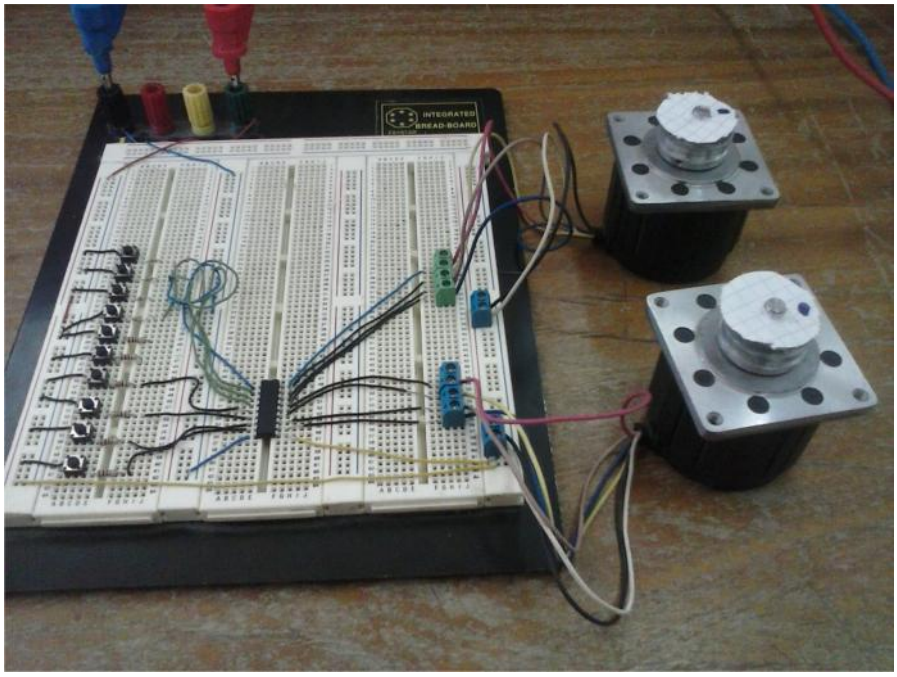


Figure III.8 Commande de deux moteurs pas à pas

III.2.6. Test du microcontrôleur (PIC 16F84) :

Le montage qui est représenté sous la figure ci-dessous consiste à tester le PIC 16F84. Pour cela, on a réalisé un circuit électronique du PIC 16F84 avec un signal d'horloge à base de deux condensateurs de 15pf et un quartz (4Mhz). La LED qui apparait s'allume grâce à un programme qu'on a injecté dans le PIC. Le changement d'état de la LED est commandé par un bouton poussoir; si on appuie sur le bouton, la LED s'allume et si on lâche le bouton elle s'éteint (Voir l'annexe2).

Le programme qu'on a injecté dans le PIC est :

```
Void main ()
{
 TRISA=0xFF;  //le PORTA en entrée
 TRISB =0x00;  //le PORTB en sortie
 while(1)
 {
 if (PORTA.B0==1) PORTB.RB4=0; //si le bouton est appuyé alors la LED s'allume
   else PORTB.RB4=1; //si non la LED s'éteint
}
}
```

La figure ci-dessous montre le circuit électronique du PIC 16F84:

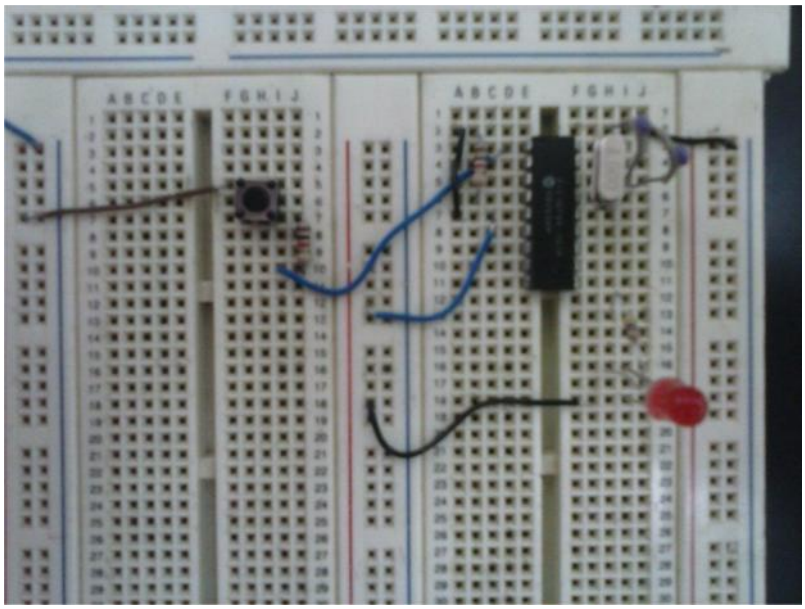


Figure III.9 Circuit électronique de test du PIC 16F84

III.2.7. Test du codeur :

Dans ce montage on a réalisé le schéma pour le test du fonctionnement du codeur 74148N, Ce codeur est de type 8/3, il a huit entrées et trois sorties inversées (voir l'annexe2).

Le test de ce codeur est effectué par six boutons poussoirs. Ils sont connectés à la masse et aux entrées du codeur. Les sorties du codeur sont reliés à trois LED. L'appuie sur un bouton poussoir nous donne un code en sortie qui est lu à travers les LED.

Exemple : Si on appuie sur le premier bouton poussoir les LED affiche le code 001.

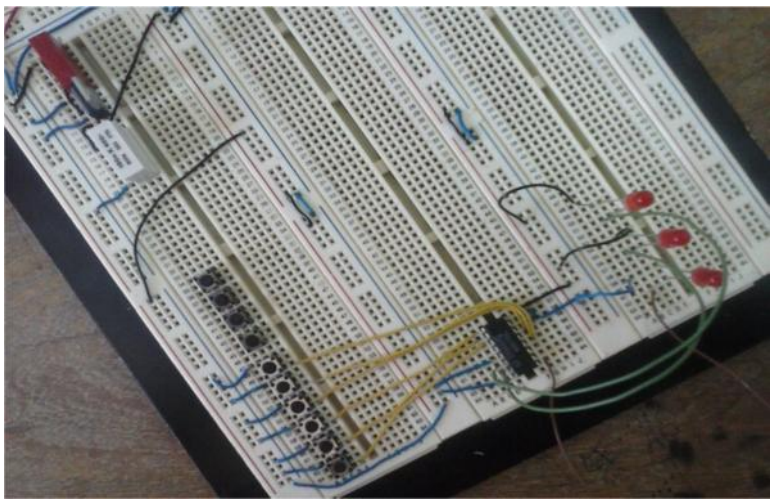


Figure III.10 Circuit du codeur

III.2.7. Test du multiplexeur :

Dans ce montage on a réalisé le schéma pour tester le fonctionnement du multiplexeur 74LS158N. Ce circuit contient 4 multiplexeurs 2/1 (voir l'annexe2).

On veut avoir un multiplexeur de mots (2 mots à l'entrée/1 mot à la sortie). Chaque mot contient 3bits.

D'après l'expérience qu'on a faite sur le codeur 74148N, on a ajouté un autre codeur identique au premier pour avoir 2 mots de 3 bits chacun à l'entrée du multiplexeur. La sortie du multiplexeur est le mot sélectionné par le bit de sélection du multiplexeur. On utilise seulement 3 MUX 2/1 pour les trois bits des mots. Le quatrième n'est pas utilisé.

Les sorties des deux codeurs sont relies aux entées du multiplexeur, les sorties de ce dernier sont testées par cinq LED, dont trois LED sont connectés à chaque pin de sorties de multiplexeur et les deux autres aux interrupteurs (commutateurs) de sélections. Une entrée de sélection est connecté au select de multiplexeur pour choisir soit le premier codeur soit le deuxième codeur; la dernière est connecté au commutateur indépendant qui désigne le mode de précision soit haute précision (un seul pas) soit basse précision (quatre pas).

Ce test nous a permis de coder les entrées de notre réalisation par le système binaire sur cinq bits pour chaque entrée. C'est ce code là qui sera lu par le PIC à travers le PORTA.

La figure ci-dessous nous montre le montage :

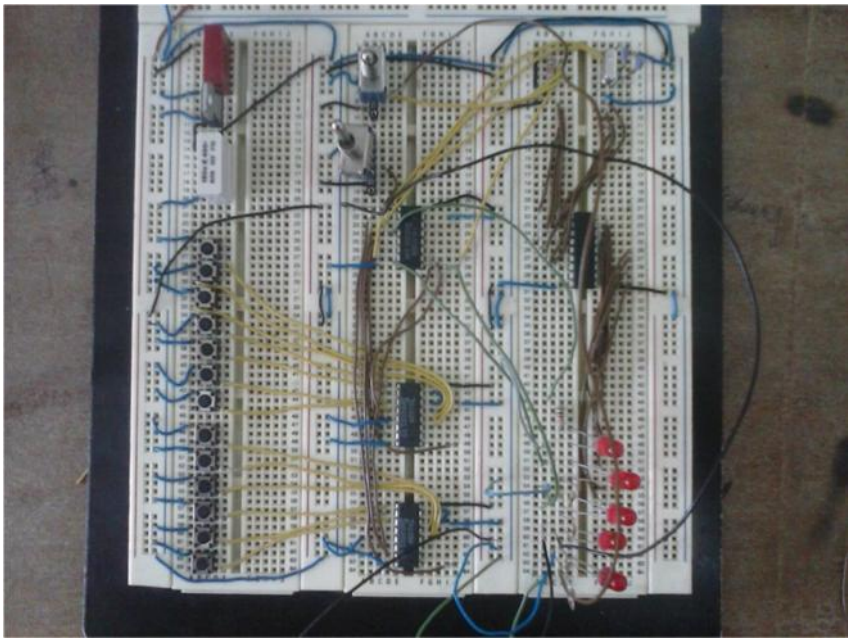


Figure III.11 Circuit des deux codeurs et le multiplexeur

III.3. Réalisation de la carte de commande pour les deux moteurs pas à pas:

Pour réaliser la carte de commande des deux moteurs pas à pas, on est passé par plusieurs tests principaux.

Pour pouvoir commander ces deux moteurs, on a réalisé la carte de commande sur la plaquette d'essai progressivement suivant les tests précédents (étapes précédentes).

La **Figure III.11** illustre une photo réelle de la carte de commande réalisée au laboratoire pour les deux moteurs pas à pas.

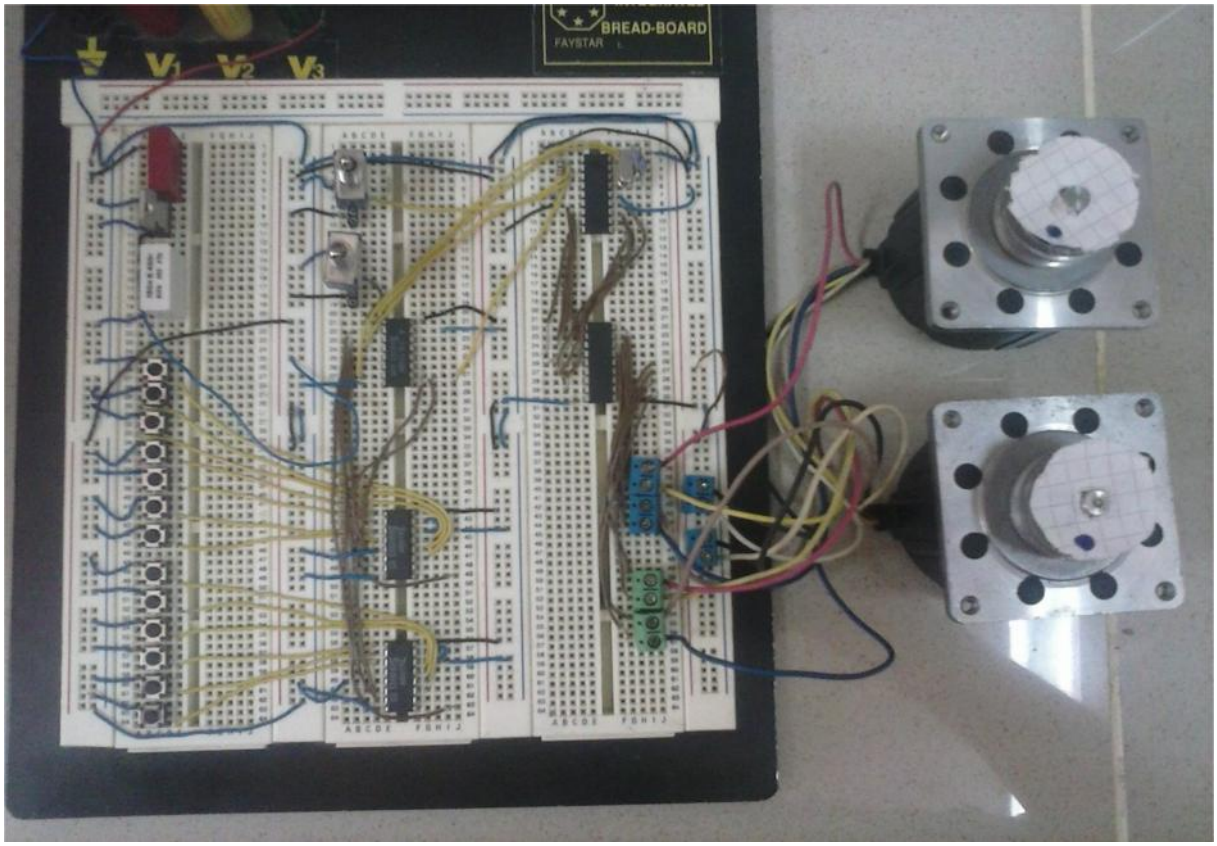


Figure III.12 Photo réelle de la carte de commande finale réalisée au laboratoire

III.4. Conclusion :

Nous avons présenté dans ce chapitre les différents composants électroniques permettant la réalisation de notre projet.
La carte de commande est constituée essentiellement d'une alimentation stabilisée, des boutons poussoirs, deux codeurs, un multiplexeur, un circuit intégré ULN2803 et deux moteur pas à pas unipolaire, le tous piloté par un microcontrôleur : le **PIC16F84.**
Nous avons également présenté les schémas de branchement détaillés de chaque composant constituant cette partie matérielle de notre travail.
Nous avons procédé pas à pas (progressivement) pour réaliser ce travail.

CHAPITRE IV : PROGRAMMATION

IV.1. Introduction:

Dans ce chapitre, nous présentons la procédure qu'on a suivie pour la création et la programmation des trajectoires. Le passage par plusieurs étapes nous a facilité la rédaction et la compréhension du programme.

IV.2. Programmation des trajectoires:

Pour le développement de nos programmes nous avons subdivisé notre travail en plusieurs étapes.

IV.2.1. Etape1 (test de rotation des deux moteurs pas à pas dans les deux sens) :

Au départ on a écrit un programme qui permet de tourner un seul moteur pas à pas dans un sens, ensuite dans le sens contraire et enfin, on a ajouté le deuxième moteur qui réalise les mêmes fonctions que le premier.

La tâche est réalisée par le programme suivant :

```
void main()   //sous programme de moteur1 sens1
{
int i;   //initialiser i
const int dly=100;    //initialiser le delay à 100ms
TRISA=0xFF;    ;     //le port A en entrée
TRISB=0x00; ;     //le port B en sortie
for(;;)
{
while(PORTA.B0==0)
{
for(i=1;i<=n;i++)
{
for(j=1;j<=3;j++)
{
for(i=1;i<=3;i++)
{
PORTB=0b00000100;delay_ms(dly);    //le 3éme bit du port B est mise à 1 (RB2=1)
PORTB=0b00001000;delay_ms(dly);    //le 4éme bit du port B est mise à 1 (RB3=1
PORTB=0b00000001;delay_ms(dly);    //le 1er bit du port B est mise à 1 (RB0=1)
PORTB=0b00000010;delay_ms(dly);    //le 2éme bit du port B est mise à 1 (RB1=1)
      }
for(i=1;i<=5;i++)
{
PORTB=0b00000001;delay_ms(dly);          //le 1er bit du port B est mise à 1 (RB0=1)
PORTB=0b00001000;delay_ms(dly);          //Le 4éme bit du port B est mise à 1 (RB3=1)
PORTB=0b00000100;delay_ms(dly);          //le 3éme bit du port B est mise à 1 (RB2=1)
PORTB=0b00000010;delay_ms(dly);          //le 2éme bit du port B est mise à 1 (RB1=1)
}
for(i=1;i<=3;i++)
{
 PORTB=0b00010000;delay_ms(dly);        //le 5éme bit du port B est mise à 1 (RB4=1)
 PORTB=0b00100000;delay_ms(dly);        //le 6éme bit du port B est mise à 1 (RB5=1)
 PORTB=0b01000000;delay_ms(dly);        //le 7éme bit du port B est mise à 1 (RB6=1)
```

```
PORTB=0b10000000;delay_ms(dly);        //le 8éme bit du port B est mise à 1 (RB7=1
}
for(i=1;i<=5;i++)
{
PORTB=0b01000000;delay_ms(dly);   //le 7éme bit du port B est mis à 1 (RB6=1)
PORTB=0b00100000;delay_ms(dly);   //le 6éme bit du port B est mis à 1 (RB5=1)
PORTB=0b00010000;delay_ms(dly);   //le 5éme bit du port B est mis à 1 (RB4=1)
PORTB=0b10000000;delay_ms(dly);   //le 8éme bit du port B est mis à 1 (RB7=1)
}
}
```

IV.2.2. Etape2 (première trajectoire):

Le balayage se fait en va et vient à partir de la position initiale (0cm, 0cm) pour arriver à la position finale.

La **figure (IV.1)** montre la manière de balayage de la première trajectoire

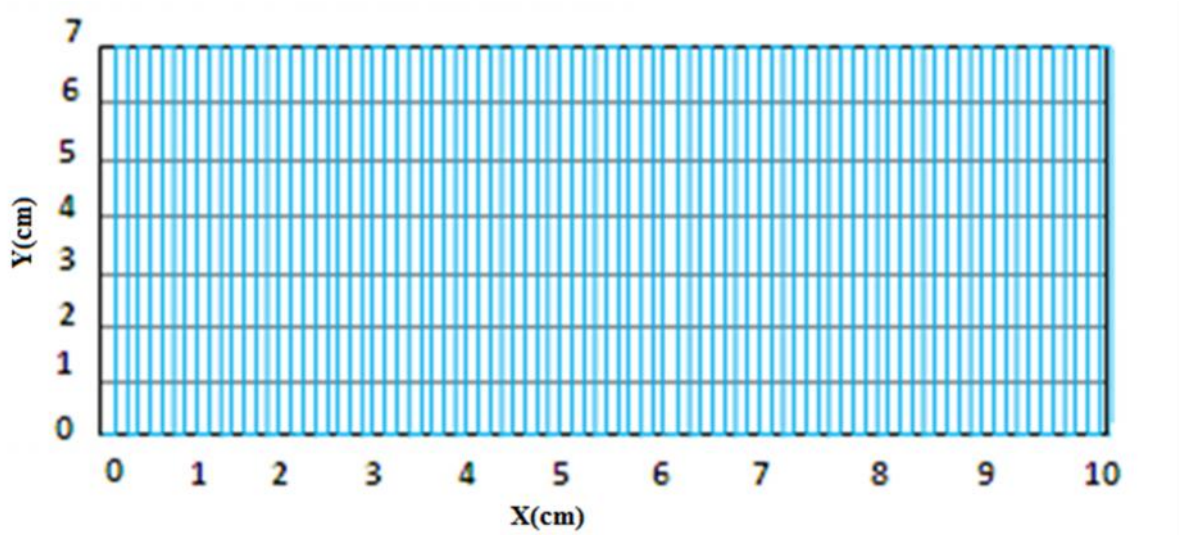


Figure IV.1 Première trajectoire.

La programmation de la première trajectoire nécessite la mise du fichier précédent sous formes de fonctions appelées par le programme principale.

La trajectoire est programmée comme suit:

```
void traj1()   //sous programme de la trajectoire1

{
int j;    //initialiser j
for(j=1;j<=3;j++)

{
m1s1(3);     //rotation du moteur1 dans le sens1
```

```
m2s1(1);   //rotation du moteur2 dans le sens1
m1s2(3);    //rotation du moteur1 dans le sens2
m2s1(1);   //rotation du moteur2 dans le sens1

}
}
```

IV.2.3. Etape3 (deuxième trajectoire):

Dans ce cas le balayage se fait comme suit : d'abord les deux moteurs sont à la position initiale (10cm, 0cm), ensuite ils commencent à tracer des rectangles, et à chaque fois ils diminuent les dimensions du rectangle de 1mm à chaque extrémité pour arriver en fin à la position final.

La **figure (IV.2)** montre la manière de balayage de la deuxième trajectoire :

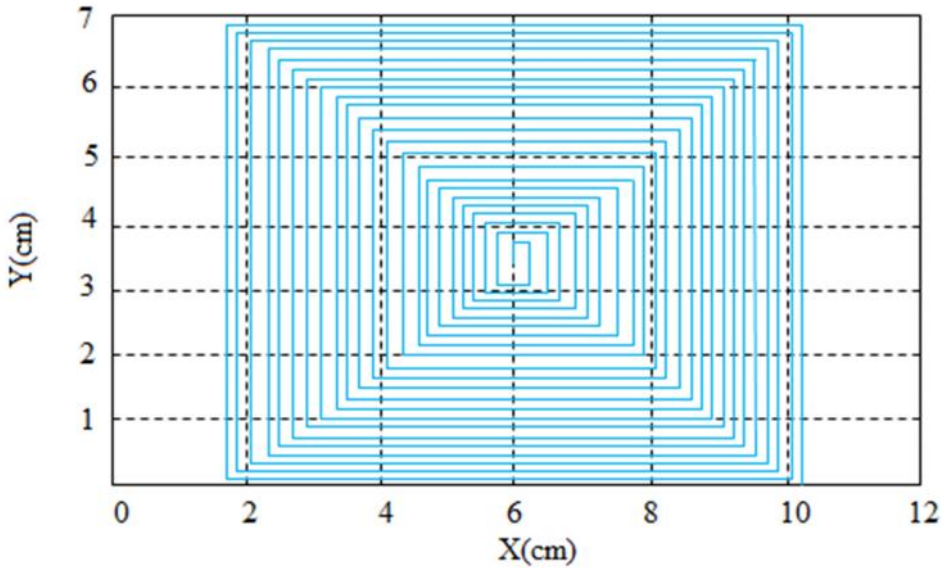


Figure IV.2 deuxième trajectoire.

On rajoute le sous programme qui décrit le fonctionnement de la deuxième trajectoire:

```
void traj2()   //sous programme de la trajectoire2

{
int j;   //initialiser j
for(j=5;j>0;j=j-2)

{
m1s1(j);    //rotation du moteur1 dans le sens1
m2s1(j);   //rotation du moteur2 dans le sens1
m1s2(j-1);   //rotation du moteur1 dans le sens2
m2s2(j-1);   //rotation du moteur2 dans le sens2
```

```
}
}
```

IV.2.4. Etape4 (la troisième trajectoire) :

Dans ce cas le balayage se fait comme suit : d'abord les deux moteurs sont à la position initiale (3.5cm, 6cm), ensuite ils commencent à tracer des rectangles, et à chaque fois ils augmentent les dimensions du rectangle de 1mm à chaque extrémité pour arriver en fin à la position final.

La **figure (IV.3)** montre la manière de balayage de la troisième trajectoire :

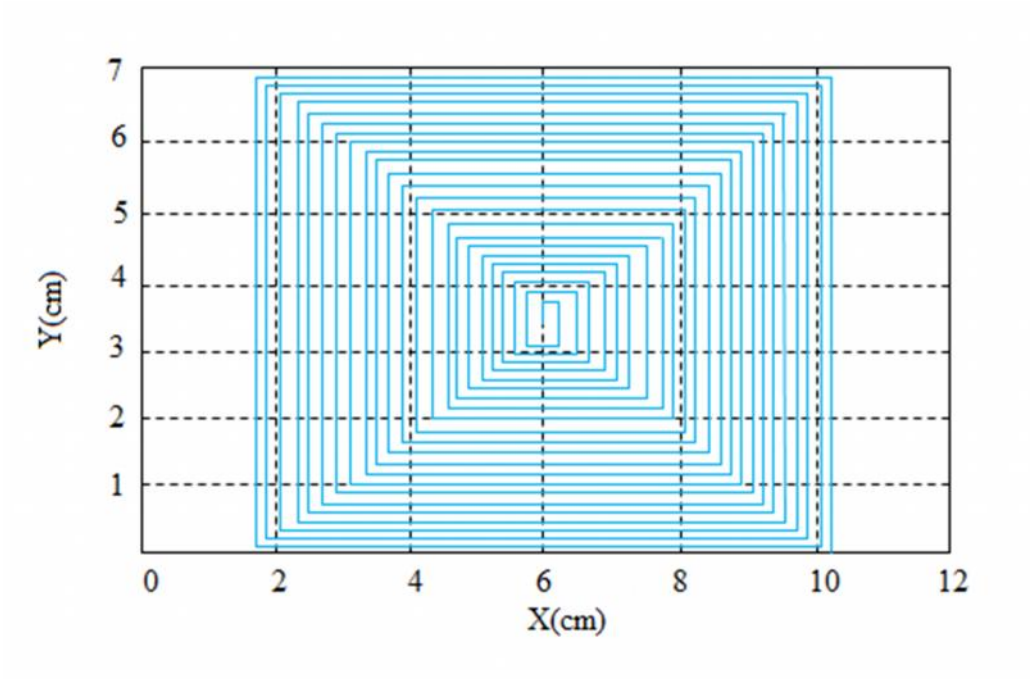


Figure IV.3 Troisième trajectoire.

Le sous programme de la troisième trajectoire est comme suit :

```
void traj3()  //sous programme de la trajectoire3

{
int j;  //initialiser j
for(j=1;5>=j;j=j+2)

{
m1s1(j);    //rotation du moteur1 dans le sens1
m2s1(j);    //rotation du moteur2 dans le sens1
m1s2(j+1);   //rotation du moteur1 dans le sens2
m2s2(j+1);   //rotation du moteur2 dans le sens2

}
}
```

IV.2.5. Etape5 (réalisation des différents sens pour que les deux moteurs fonctionnent au même temps):

Pour que nous puissions réaliser la quatrième trajectoire, on doit d'abord introduire les sous programme qui font tourner les deux moteur ensemble.

Les sous programmes sont :

```
void ms11(int n)    //sous programme de moteur1 sens1 et moteur2 sens1

 {
int i; //initialiser i
const int dly=100;   //initialiser le delay à 100ms
for(i=1;i<=n;i++)

 {

If (PORTA==0b00010) return  ;      //si le bouton poussoir2 est différent de 1 alors stop
PORTB=0b00010100;delay_ms(dly);   //le 3éme et le 5éme bit du port B sont mis à1 (RB2=1,RB4=1)
PORTB=0b00101000;delay_ms(dly);   //le 4éme et le 6éme bit du port B sont mis à 1(RB3=1,RB5=1)
PORTB=0b01000001;delay_ms(dly);   //le 1er et le 7éme bit du port B sont mis à 1 ( RB0=1,RB6=1)
PORTB=0b10000010;delay_ms(dly);   //le 2éme et le 8éme bit du port B sont mis à 1(RB1=1,RB7=1)

 }
}

void ms12(int n)  //sous programme de moteur1 sens1 et moteur2 sens2

{
int i;  //initialiser i
const int dly=100;  //initialiser le delay à 100ms
for(i=1;i<=n;i++)

{

if (PORTA==0b00010) return  ;  //si le bouton poussoir2 est différent de 1 alors stop
PORTB=0b01000100;delay_ms(dly);    //le 3éme et le 7éme bit du port B sont mis à1(RB2=1,RB6=1)
PORTB=0b00101000;delay_ms(dly);  //le 4éme et le 6éme bit du port B sont mis à 1 (RB3=1,RB5=1)
PORTB=0b00010001;delay_ms(dly);     //le 1er et le 5éme bit du port B sont mis à 1 (RB0=1,RB4=1)
PORTB=0b10000010;delay_ms(dly);  //le 2éme et le 8éme bit du port B sont mis à 1 (RB1=1,RB7=1)

}
}



void ms22(int n)  //sous programme de moteur1 sens2 et moteur2 sens2

{
int i;   //initialiser i
```

```
const int dly=100;   //initialiser le delay à 100ms
for(i=1;i<=n;i++)

{

 if (PORTA==0b00010) return  ;    //si le bouton poussoir2 est différent de 1 alors stop
PORTB=0b01000001;delay_ms(dly);    //le 1er et le 7me bit du port B sont mis à 1 (RB0=1,RB6=1)
PORTB=0b00101000;delay_ms(dly);  //le 4éme et le 6éme bit du port B sont mis à 1 (RB3=1,RB5=1)
PORTB=0b00010100;delay_ms(dly);   //le 3éme et le 5éme bit du port B sont mis à 1(RB2=1,RB4=1)
PORTB=0b10000010;delay_ms(dly);   //le 2éme et le 8éme bit du port B sont mis à 1(RB1=1,RB7=1)

}
}
void ms21(int n)  //sous programme de moteur1 sens2 et moteur2 sens1

{
int i;  //initialiser i
const int dly=100;  //initialiser le delay à 100ms
for(i=1;i<=n;i++)

{

 if (PORTA==0b00010) return  ;  //si le bouton poussoir2 est différent de 1 alors stop
PORTB=0b00010001;delay_ms(dly);    //le 1er et le 5éme bit du port B sont mis à 1 (RB0=1,RB4=1)
PORTB=0b00101000;delay_ms(dly);  //le 4éme et le 6éme bit du port B sont mis à 1 (RB3=1,RB5=1)
PORTB=0b01000100;delay_ms(dly);  //le 3éme et le 7éme bit du port B sont mis à 1 (RB2=1,RB6=1)
PORTB=0b10000010;delay_ms(dly);  //le 2éme et le 8éme bit du port B sont mis à 1 (RB1=1,RB7=1)

}
}
```

IV.2.6. Etape6 (quatrième trajectoire) :

La **figure (VI.4)** montre la manière de balayage de la quatrième trajectoire :

Le balayage se fait comme suit : d'abord les deux moteurs tracent le premier octogone, ensuite ils commencent à tracer les autres octogones dont les longueurs des cotés (haut, bas, gauche, droite) sont gardés fixes. Les autres cotés augmentent de 1mm à chaque extrémité et à chaque pas pour arriver en fin à la position final.

Cette trajectoire remplace la cinquième trajectoire qu'on a présentée dans le chapitre II.

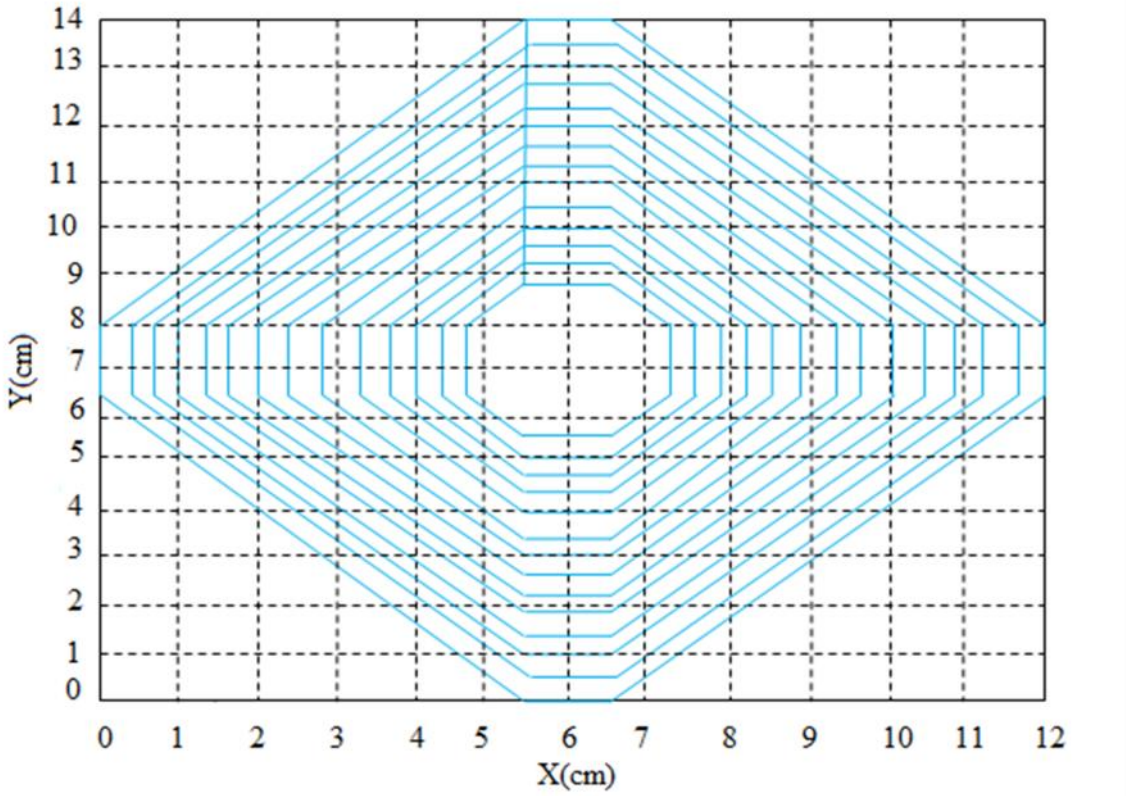


Figure IV.4 Quatrième trajectoire.

```
void traj4()    //sous programme de la trajectoire4

{
int j;   //initialiser j
for(j=1;j<=3;j=j+1)

{
m1s1(1);   //rotation du moteur1 dans le sens1
ms12(j);    //rotation du moteur1 dans le sens1 et le moteur2 dans le sens2
m2s2(1);   //rotation du moteur2 dans le sens2
ms22(j);   //rotation du moteur1 dans le sens2 et le moteur2 dans le sens2
m1s2(1);   //rotation du moteur1 dans le sens2
ms21(j);   //rotation du moteur1 dans le sens2 et le moteur2 dans le sens1
m2s1(1);   //rotation du moteur2 dans le sens1
ms11(j);    //rotation du moteur1 dans le sens1 et le moteur2 dans le sens1
m2s1(1);   //rotation du moteur2 dans le sens1

}
}
```

IV.3. Conclusion:

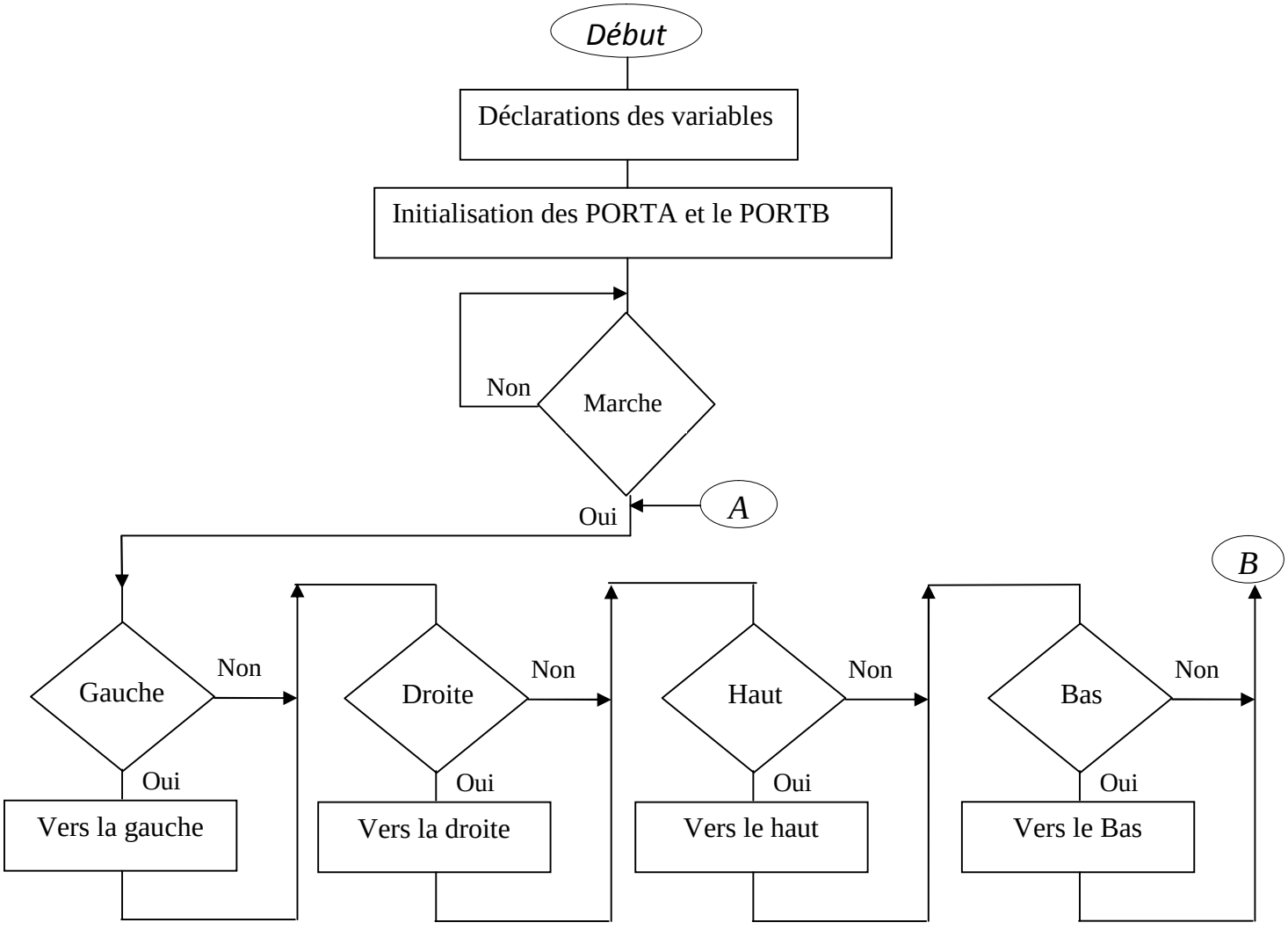
La programmation du PIC est une tâche relativement facile, surtout avec L'utilisation de langage évolué approprié comme le MikroC. L'architecture interne du PIC à encore faciliter cette tache grâce à sa structure en registre ce qui permet de configurer chaque périphérique séparément.
Le PIC offre une multitude de solutions pour un problème donné : choix de timers, plusieurs modes de fonctionnement du module, cela rend la tache plus facile.

CHAPITRE V :
TESTS FINAUX ET VALIDATION

V.1.Introduction :

Le PIC est le cœur de la commande dans notre travail. C'est à lui que le fonctionnement de tous les autres périphériques est lié, et c'est lui qui traite les données envoyées par l'operateur, et c'est lui qui pilote les moteurs, et c'est lui qui communique avec l'operateur. Mais comment assure-t-il cette commande ? Pour répondre à cette question nous allons nous intéresser dans ce chapitre au programme qu'on a élaboré pour le contrôle de la carte de commande par le PIC.

V.2. Organigramme



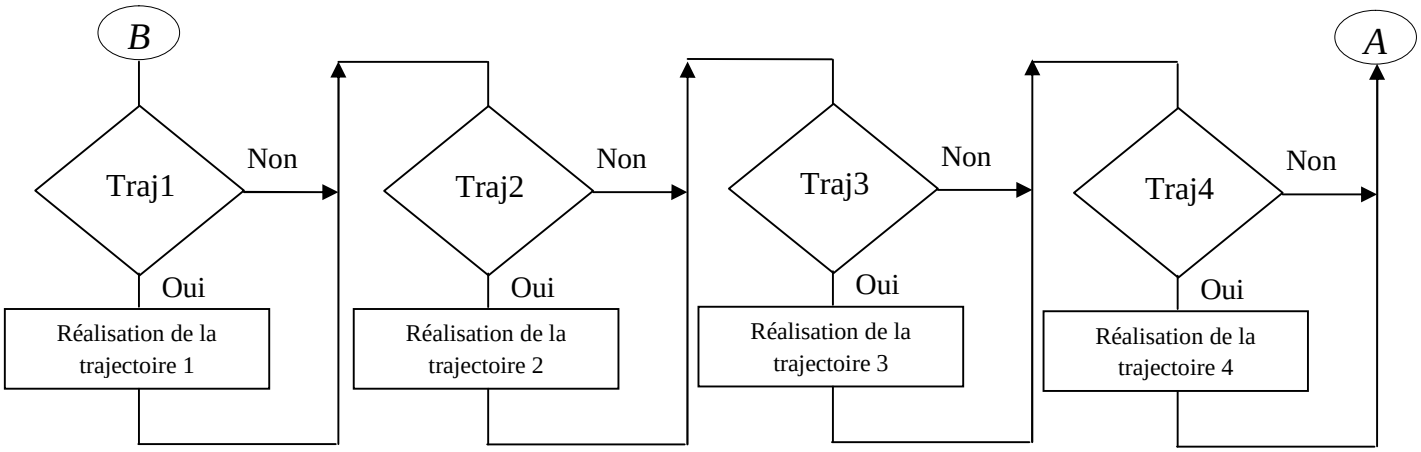


Figure V.1 Organigramme

V.3. Le programme final:

Le programme qu'on a construit pour réaliser toutes les fonctions est le suivant :

```
//prototypes des fonctions

void m1s1(int);     //déclaration de la fonction moteur1 sens1
void m1s2(int);     //déclaration de la fonction moteur1 sens2
void m2s1(int);     //déclaration de la fonction moteur2 sens1
void m2s2(int);     //déclaration de la fonction moteur2 sens2
void ms11(int);     //déclaration de la fonction moteur1 sens1 et moteu2 sens1
void ms12(int);     //déclaration de la fonction moteur1 sens1 et moteu2 sens2
void ms22(int);     //déclaration de la fonction moteur1 sens2 et moteu2 sens2
void ms21(int);     //déclaration de la fonction moteur1 sens2 et moteu2 sens1
void traj1();        //déclaration de la fonction trajectoire1
void traj2();       //déclaration de la fonction trajectoire2
void traj3();       //déclaration de la fonction trajectoire3
void traj4();       //déclaration de la fonction trajectoire4
void traj5();       //déclaration de la fonction trajectoire5
void traj6();       //déclaration de la fonction trajectoire6
void traj7();       //déclaration de la fonction trajectoire7

void m1s1H(int);     //déclaration de la fonction moteur1 sens1 (haute précision)
void m1s2H(int);    //déclaration de la fonction moteur1 sens2 (haute précision)
void m2s1H(int);    //déclaration de la fonction moteur2 sens1 (haute précision)
void m2s2H(int);    //déclaration de la fonction moteur2 sens2 (haute précision)
void ms11H(int);    //déclaration de la fonction moteur1 sens1 et moteu2 sens1 (haute précision)
void ms12H(int);    //déclaration de la fonction moteur1 sens1 et moteu2 sens2 (haute précision)
void ms22H(int);    //déclaration de la fonction moteur1 sens2 et moteu2 sens2 (haute précision)
void ms21H(int);    //déclaration de la fonction moteur1 sens2 et moteu2 sens1 (haute précision)
void traj1H();       //déclaration de la fonction trajectoire1 (haute précision)
```

```
void traj2H();        //déclaration de la fonction trajectoire2 (haute précision)
void traj3H();        //déclaration de la fonction trajectoire3 (haute précision)
void traj4H();        //déclaration de la fonction trajectoire4 (haute précision)
void traj5H();        //déclaration de la fonction trajectoire5 (haute précision)
void traj6H();        //déclaration de la fonction trajectoire6 (haute précision)
void traj7H();        //déclaration de la fonction trajectoire7 (haute précision)

const int dly=500; //initialiser le delay à 500ms

 void main()    //programme principale

{
int i,j;    //initialiser i,j
int PA=0 ;
TRISA=0xFF;    //le port A en entrée
TRISB=0x00;    //le port B en sortie

for(;;)                //la suite de programme s'effectue en boucle

{
if (PORTA==1) PA=1;
while(PA==0b00001)            //en appuyant sur le bouton poussoir1
{
{
if(PORTA==0b00011) m1s1(1); //si le bouton poussoir3 est appuyé alors marcher le moteur1 dans le sens1 avec 4 pas
if(PORTA==0b00100) m1s2(1);   //si le bouton poussoir4 est appuyé alors marcher le moteur1 dans le sens2 avec 4 pas
if(PORTA==0b00101) m2s1(1);   //si le bouton poussoir5 est appuyé alors marcher le moteur2 dans le sens1 avec 4 pas
if(PORTA==0b00110) m2s2(1);   //si le bouton poussoir6 est appuyé alors marcher le moteur2 dans le sens2 avec 4 pas
if(PORTA==0b01001) traj1();    //si le bouton poussoir7 est appuyé alors construire la trajectoire1
if(PORTA==0b01010) traj2();   //si le bouton poussoir8 est appuyé alors construire la trajectoire2
if(PORTA==0b01011) traj3(); //si le bouton poussoir9 est appuyé alors construire la trajectoire3
if(PORTA==0b01100) traj4();   //si le bouton poussoir10 est appuyé alors construire la trajectoire4
//if(PORTA==0b01101) traj5();   //si le bouton poussoir11 est appuyé alors construire la trajectoire5
//if(PORTA==0b01110) traj6();   //si le bouton poussoir12 est appuyé alors construire la trajectoire6
//if(PORTA==0b01111) traj7();   //si le bouton poussoir13 est appuyé alors construire la trajectoire7

// haute précision

 //if(PORTA==0b10011) m1s1H(1); //si le bouton poussoir3 est appuyé alors marcher le moteur1 dans le sens1 d'un seul pas (haute précision)

//if(PORTA==0b10100) m1s2H(1); //si le bouton poussoir4 est appuyé alors marcher le moteur1 dans le sens2 d'un seul pas (haute précision)

//if(PORTA==0b10101) m2s1H(1); //si le bouton poussoir5 est appuyé alors marcher le moteur2 dans le sens1 d'un seul pas (haute précision)
```

```
//if(PORTA==0b10110) m2s2H(1); //si le bouton poussoir6 est appuyé alors marcher le moteur2 dans le sens2 d'un seul pas (haute précision)

//if(PORTA==0b11001) traj1H();     //si le bouton poussoir7 est appuyé alors construire la  trajectoire1 (haute précision)

//if(PORTA==0b11010) traj2H();   //si le bouton poussoir8 est appuyé alors construire la trajectoire2 (haute précision)

//if(PORTA==0b11011) traj3H();   //si le bouton poussoir9 est appuyé alors construire la trajectoire3 (haute précision)

//if(PORTA==0b11100) traj4H();   //si le bouton poussoir10 est appuyé alors construire la trajectoire4 (haute précision)

//if(PORTA==0b11101) traj5H();   //si le bouton poussoir11 est appuyé alors construire la trajectoire5 (haute précision)

//if(PORTA==0b11110) traj6H();   //si le bouton poussoir12 est appuyé alors construire la trajectoire6 (haute précision)

//if(PORTA==0b11111) traj7H();   //si le bouton poussoir13 est appuyé alors construire la trajectoire7 (haute précision)

}
}
}
}

void m1s1(int n)   //sous programme de moteur1 sens1

{
int i;   //initialiser i
for(i=1;i<=n;i++)
{
if(PORTA==0b00010) return ; //si le bouton poussoir2 est différent de 1 alors stop

if (PORTA==0b10010) return ; //si le bouton poussoir2 est différent de 1 alors stop

PORTB=0b00000100;delay_ms(dly);    //le 3éme bit du port B est mise à 1 (RB2=1)
PORTB=0b00001000;delay_ms(dly);    //le 4éme bit du port B est mise à 1 (RB3=1
PORTB=0b00000001;delay_ms(dly);    //le 1er bit du port B est mise à 1 (RB0=1)
PORTB=0b00000010;delay_ms(dly);    //le 2éme bit du port B est mise à 1 (RB1=1)

 }
}

void m1s2(int n) //sous programme de moteur1 sens2

{
int i; //initialiser i
```

```
for(i=1;i<=n;i++)

{

 if(PORTA==0b00010) return ; //si le bouton poussoir2 est différent de 1 alors stop

if (PORTA==0b10010) return ; //si le bouton poussoir2 est différent de 1 alors stop

PORTB=0b00000001;delay_ms(dly);          //le 1er bit du port B est mise à 1 (RB0=1)
PORTB=0b00001000;delay_ms(dly);          //Le 4éme bit du port B est mise à 1 (RB3=1)
PORTB=0b00000100;delay_ms(dly);          //le 3éme bit du port B est mise à 1 (RB2=1)
PORTB=0b00000010;delay_ms(dly);          //le 2éme bit du port B est mise à 1 (RB1=1)

}
}

void m2s1(int n)      //sous programme de moteur2 sens1

{
int i;    //initialiser i
for(i=1;i<=n;i++)

{

if(PORTA==0b00010) return ; //si le bouton poussoir2 est différent de 1 alors stop

if (PORTA==0b10010) return ; //si le bouton poussoir2 est différent de 1 alors stop


PORTB=0b00010000;delay_ms(dly);        //le 5éme bit du port B est mise à 1 (RB4=1)
PORTB=0b00100000;delay_ms(dly);        //le 6éme bit du port B est mise à 1 (RB5=1)
PORTB=0b01000000;delay_ms(dly);        //le 7éme bit du port B est mise à 1 (RB6=1)
PORTB=0b10000000;delay_ms(dly);        //le 8éme bit du port B est mise à 1 (RB7=1)

 }
}
void m2s2(int n)    //sous programme de moteur2 sens2

{
int i; //initialiser i
for(i=1;i<=n;i++)

{
if(PORTA==0b00010) return ; //si le bouton poussoir2 est différent de 1 alors stop

if (PORTA==0b10010) return ; //si le bouton poussoir2 est différent de 1 alors stop

PORTB=0b01000000;delay_ms(dly);    //le 7éme bit du port B est mis à 1 (RB6=1)
PORTB=0b00100000;delay_ms(dly);    //le 6éme bit du port B est mis à 1 (RB5=1)
PORTB=0b00010000;delay_ms(dly);    //le 5éme bit du port B est mis à 1 (RB4=1)
PORTB=0b10000000;delay_ms(dly);    //le 8éme bit du port B est mis à 1 (RB7=1)
```

```
}
}

 void ms11(int n)    //sous programme de moteur1 sens1 et moteur2 sens1

 {
int i; //initialiser I
for(i=1;i<=n;i++)

{

if(PORTA==0b00010) return ; //si le bouton poussoir2 est différent de 1 alors stop

if (PORTA==0b10010) return ; //si le bouton poussoir2 est différent de 1 alors stop

PORTB=0b00010100;delay_ms(dly);    //le 3éme et le 5éme bit du port B sont mis à (RB2=1,RB4=1)
PORTB=0b00101000;delay_ms(dly);   //le 4éme et le 6éme bit du port B sont mis à 1(RB3=1,RB5=1)
PORTB=0b01000001;delay_ms(dly);   //le 1er et le 7éme bit du port B sont mis à 1 ( RB0=1,RB6=1)
PORTB=0b10000010;delay_ms(dly);   //le 2éme et le 8éme bit du port B sont mis à 1(RB1=1,RB7=1)

 }
}

void ms12(int n)  //sous programme de moteur1 sens1 et moteur2 sens2

{
int i;  //initialiser i
for(i=1;i<=n;i++)

{

if(PORTA==0b00010) return ; //si le bouton poussoir2 est différent de 1 alors stop

if (PORTA==0b10010) return ; //si le bouton poussoir2 est différent de 1 alors stop

PORTB=0b01000100;delay_ms(dly);    //le 3éme et le 7éme bit du port B sont mis à1(RB2=1,RB6=1)
PORTB=0b00101000;delay_ms(dly);  //le 4éme et le 6éme bit du port B sont mis à 1 (RB3=1,RB5=1)
PORTB=0b00010001;delay_ms(dly);     //le 1er et le 5éme bit du port B sont mis à 1 (RB0=1,RB4=1)
PORTB=0b10000010;delay_ms(dly);  //le 2éme et le 8éme bit du port B sont mis à 1 (RB1=1,RB7=1)

 }
}



void ms22(int n)  //sous programme de moteur1 sens2 et moteur2 sens2

{
int i;   //initialiser i
for(i=1;i<=n;i++)

 {
```

```
if(PORTA==0b00010) return ; //si le bouton poussoir2 est différent de 1 alors stop

if (PORTA==0b10010) return ; //si le bouton poussoir2 est différent de 1 alors stop

PORTB=0b01000001;delay_ms(dly);    //le 1er et le 7me bit du port B sont mis à 1 (RB0=1,RB6=1)
PORTB=0b00101000;delay_ms(dly);  //le 4éme et le 6éme bit du port B sont mis à 1 (RB3=1,RB5=1)
PORTB=0b00010100;delay_ms(dly);   //le 3éme et le 5éme bit du port B sont mis à 1(RB2=1,RB4=1)
PORTB=0b10000010;delay_ms(dly);   //le 2éme et le 8éme bit du port B sont mis à 1(RB1=1,RB7=1)

 }
}
void ms21(int n)  //sous programme de moteur1 sens2 et moteur2 sens1

{
int i;  //initialiser i
for(i=1;i<=n;i++)

{

if(PORTA==0b00010) return ; //si le bouton poussoir2 est différent de 1 alors stop

if (PORTA==0b10010) return ; //si le bouton poussoir2 est différent de 1 alors stop

PORTB=0b00010001;delay_ms(dly);    //le 1er et le 5éme bit du port B sont mis à 1 (RB0=1,RB4=1)
PORTB=0b00101000;delay_ms(dly);  //le 4éme et le 6éme bit du port B sont mis à 1 (RB3=1,RB5=1)
PORTB=0b01000100;delay_ms(dly);  //le 3éme et le 7éme bit du port B sont mis à 1 (RB2=1,RB6=1)
PORTB=0b10000010;delay_ms(dly);  //le 2éme et le 8éme bit du port B sont mis à 1 (RB1=1,RB7=1)

}
}

void traj1()   //sous programme de la trajectoire1

{
int j;    //initialiser j
for(j=1;j<=6;j++)

{
m1s1(60);     //rotation du moteur1 dans le sens1
m2s1(5);   //rotation du moteur2 dans le sens1
m1s2(60);    //rotation du moteur1 dans le sens2
m2s1(5);   //rotation du moteur2 dans le sens1

 }
}

void traj2()   //sous programme de la trajectoire2

{
int j;   //initialiser j
```

```
for(j=60;j>0;j=j-20)

{
m1s1(j);     //rotation du moteur1 dans le sens1
m2s1(j);    //rotation du moteur2 dans le sens1
m1s2(j-10);    //rotation du moteur1 dans le sens2
m2s2(j-10);    //rotation du moteur2 dans le sens2

}
}

void traj3()  //sous programme de la trajectoire3

{
int j;  //initialiser j
for(j=10;60>=j;j=j+20)

{
m1s1(j);     //rotation du moteur1 dans le sens1
m2s1(j);     //rotation du moteur2 dans le sens1
m1s2(j+10);    //rotation du moteur1 dans le sens2
m2s2(j+10);    //rotation du moteur2 dans le sens2

}
}

void traj4()    //sous programme de la trajectoire4

{
int j;   //initialiser j
for(j=10;j<=35;j=j+10)

{
m1s1(10);   //rotation du moteur1 dans le sens1
ms12(j);    //rotation du moteur1 dans le sens1 et le moteur2 dans le sens2
m2s2(10);   //rotation du moteur2 dans le sens2
ms22(j);   //rotation du moteur1 dans le sens2 et le moteur2 dans le sens2
m1s2(10);   //rotation du moteur1 dans le sens2
ms21(j);   //rotation du moteur1 dans le sens2 et le moteur2 dans le sens1
m2s1(10);   //rotation du moteur2 dans le sens1
ms11(j);    //rotation du moteur1 dans le sens1 et le moteur2 dans le sens1
m2s1(5);   //rotation du moteur2 dans le sens1

}
}

(
```

V.4. Carte de commande :

La figure ci-dessous nous montre la réalisation finale de la carte de commande:

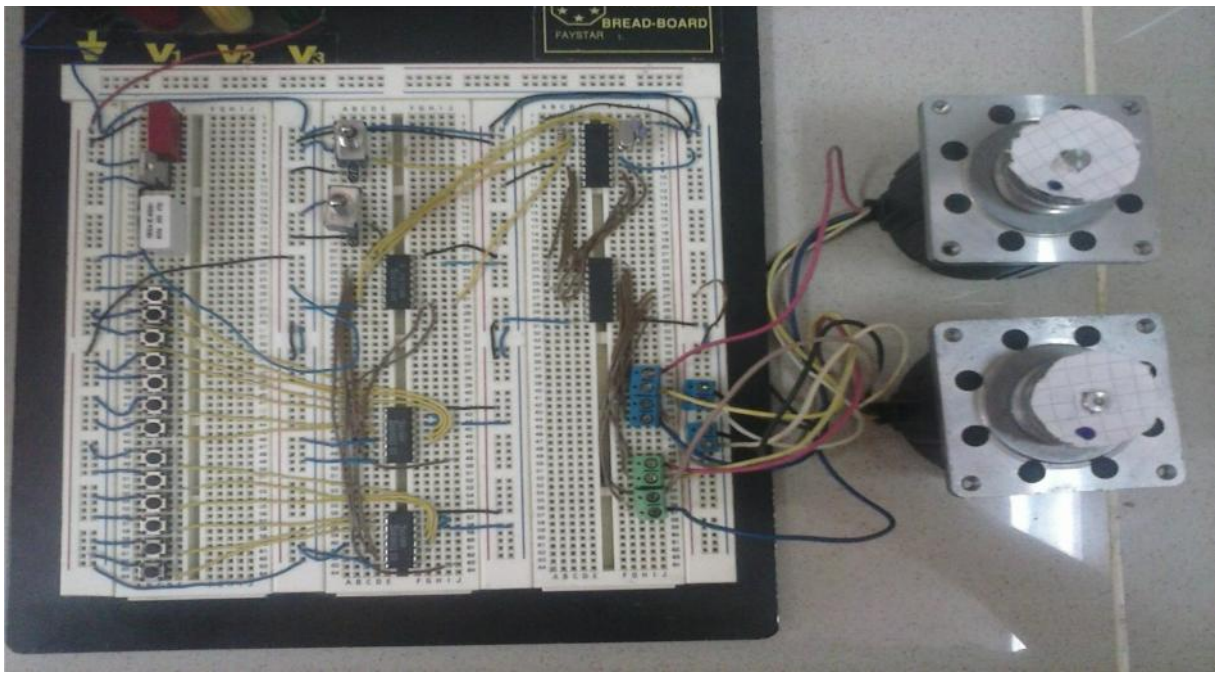


Figure V.2 Photo réel de la carte commande réalisée au laboratoire

V.5. Schéma de simulation :

ISIS **(Intelligent Schematic Input System)** est le module de saisie des schémas électroniques et de simulation de la suite logicielle de PROTEUS.

Le schéma réalisé sous PROTEUS ISIS est comme suit :

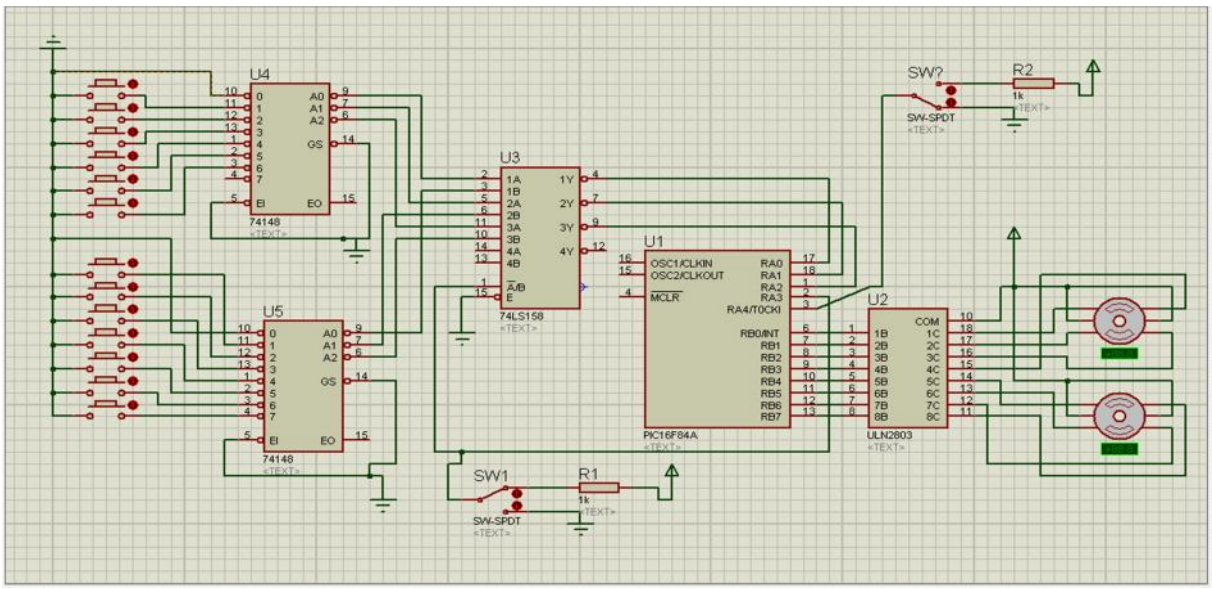


Figure V.3 Schéma simplifié de la carte de commande réalisé sous ISIS

V.6 Conclusion :

Dans ce chapitre nous avons finalisé notre travail, en faisant apparaitre les fonctions des circuits électroniques grâce aux programmes établit avec le MikroC, qui est simple et facile à comprendre ainsi son manuel est complet, il nous a permis de créer le code machine qui sera chargé dans la mémoire du microcontrôleur.
Nous avons évité la programmation par l'assembleur compte-tenu de la complexité du jeu d'instructions, et notre programme nécessite d'élaborer des fonctions particulières.
Nous avons utilisé Proteus ISIS comme logiciel de simulation, pour présenter les mouvements des moteurs a travers la carte de commande qu'on a détaillé dans le chapitre précédent.

CONCLUSION GENERALE

Conclusion générale :

A travers ce mémoire, nous avons rappelé les différentes notions nécessaires pour l'étude des systèmes de positionnents. Nous avons précisé la définition de certains termes qui sont nécessaires, pour leur étude, les notions et outils mathématiques mis en œuvre pour la modélisation, basée sur la structure mécanique, les actionneurs et les capteurs.

La génération des trajectoires permet d'avoir les consignes que le système doit poursuivre, ces trajectoires sont destinées à positionner le dispositif pour réaliser certaines tâches.
Pour notre travail, on s'est intéressé à la génération de quatre trajectoires pour obtenir différentes manières de faire le balayage d'une surface.

L'idée de l'utilisation des microcontrôleurs est née de la nécessité de disposer pour certaines applications d'une commande avec des performances assez élevées. L'étude qui vient d'être présentée permet de donner une vue globale sur la commande par un PIC qui présente des avantages immenses, ainsi les techniques de leurs applications.
Dans la réalisation de notre projet, nous avons suivi une méthode progressive. A chaque étape nous avons vérifié le bon fonctionnement des composants et des circuits jusque là réalisés. Le passage à l'étape suivante ne peut se faire que par la validation de l'étape précédente.

Les tests lors de l'élaboration du travail valident pas à pas le système réalisé.

Le programme injecté dans le PIC est basé sur une approche fonctionnelle dans laquelle on raisonne par les tâches à réaliser (mouvements de base, mouvements combinés et les trajectoires).

Notre travail est fait pour réaliser des balayages d'une surface plane et peut être utilisé comme plateforme pour les robots mobiles et autres applications. Il suffit de changer les positions des moteurs et le programme injecté dans le PIC pour réaliser d'autres types de trajectoires.

En perspectives, on propose de faire des améliorations (rajout de la haute précision), de perfectionnements du fonctionnement ainsi une étude concernant le coté puissance.

Bibliographie

[1] GUILLAUME SABATIER, FRANCOIS RAGUSA et HUBERT ANTZ, « Manuel de Technologie Mécanique » Edition DUNOD 2006.

[2] ANTHONY JUTON « Automatisme Industriels I » Janvier 2007.

[3] un article de WIKIPEDIA, L'encyclopédie libre, conception assiste par ordinateur, consulté en Mai 2014

[4] L.BERGOUGNOUX, « Automates Programmables Industriels » Polytech Marseille département de mécanique énergétique 2eme année option S.I.I.C 2004 /2005.

[5] W.KHALIL, ETIENNE DOMBRE « Modélisation, identification et commandes des robots » 2^{éme} Edition hermes, 1999

[6] B.Tackfarinas, B.Madjid « Etude et Réalisation d'un Système de Positionnement à 2D /3D à l'aide d'un logiciel de réalité virtuelle » mémoire de fin d'étude en vue d'obtention d'un diplôme master en AI, 2011 /2012.

[7] Cours CAO 2013/2014

[8] Quelque transparent de cours sur la méthode SADT 1995/1996

[9] PATRICE OGUIC « Moteur pas à as et PC » ETSF 2004

[10] PIERE MAYE « Moteur électrique pour la robotique » Edition DUNOD Paris 2002

[11] GERAND LACROUX « les actionneurs électrique pour la robotique et les asservissement »Technique et Documentation Lavoisier 1994

[12] site internet, http: //www.xcotton.fr

[13] PASCAL MAYEUX « «Apprendre la programmation des pic par l'expérimentation et la simulation » 3^{éme} Edition DUNOD Paris 2005

[14] www.datasheetcalog.com/datasheet: Electronique composants, consulté en juin 2014

ANNEXES

ANNEXE 1

A1.1. Définition et description :

Ces moteurs comportent un rotor et un stator. Le rotor est soit muni d'aimants permanents (structure dite polarisée ou active), soit constitué par des pièces ferromagnétiques dentées (structure dite à reluctance variable ou passive). La troisième structure est le mélange des deux structures précédentes (structure hybride) le stator quant à lui, est constitue d'un certain nombre de bobines d'excitation.

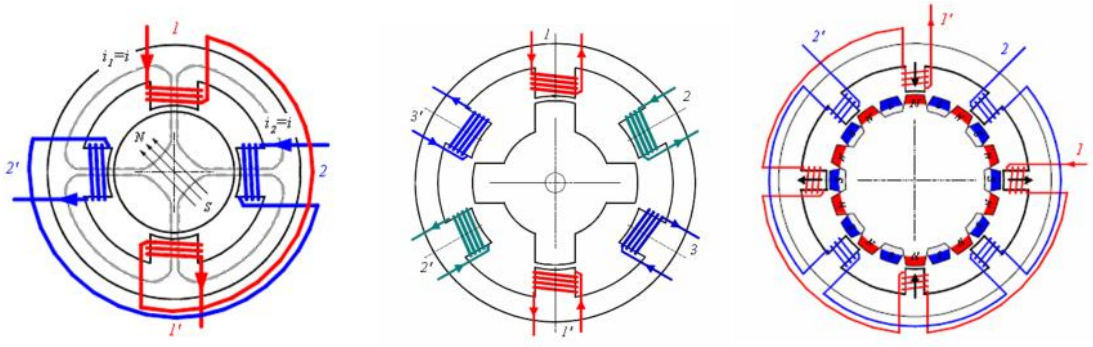
L'alimentation de chaque bobinage du moteur pas à pas par une tension provoque l'apparition d'un courant qui engendre un champ magnétique de direction précise.

Le changement séquentiel des tensions appliquées à chaque bobinage permet de déplacer la position du rotor selon une résolution élémentaire appelée pas. La succession des configurations d'alimentation à une fréquence donnée, impose un champ statoriques tournant, avec une résolution d'un pas complet, d'un demi pas ou d'un micro pas.

A1.2. Types des moteurs pas à pas:

Diverses technologies de moteur pas à pas sont disponibles, la seule différence entre elles résident dans la forme du rotor. On distingue trois types :

- Moteur à aimants permanents
- Moteur à reluctance variable
- Moteur hybride



(a) Aimants permanents (b) reluctance variable (c) Hybride

A1.1 représentation des différents types de moteurs pas à pas.

A1.3. Les caractéristiques des moteurs pas à pas :

Il existe différents types de moteur pas à pas mais ils présentent tous un certain nombre de caractéristiques communes tant en fonctionnement statique que dynamique dues à leur alimentation particulière **[11].**

A1.3.1. Caractéristiques statiques :

Les caractéristiques statiques sont celles obtenues à partir du courant établi dans une ou plusieurs phases de la machine.

- **Pas angulaire :** c'est la valeur de l'angle de rotation effectué par le moteur sous l'action d'une commutation. On définit le moteur par le nombre de pas nécessaire pour effectuer un tour complet. Ce nombre de pas dépend à la fois du type de construction et du mode d'alimentation.
- **Vitesse de rotation :** elle est proportionnelle à la fréquence de commutation avec nombre de pas par tour est une fréquence f, la vitesse de rotation est f/n_p en tours par seconde.
- **Couple moteur :** lorsque le moteur est alimenté, son rotor tend à s'aligné sur l'axe du champ ainsi crée. De la tentative de l'écarter de cette position, nait un couple croissant avec le décalage angulaire. Le couple passe par une valeur maximale désignée sous le nom de « couple statique ».
- **Couple de détente :** c'est le couple maximal qui peut être appliqué sur l'arbre du moteur non alimenté sans qu'il y ait rotation d'un pas. Ce couple n'existe que sur les machines à aimants.
- **Erreur sur le pas :** les imperfections de la réalisation mécanique sont la cause d'un écart de position par rapport à la position théorique définie par le nombre de pas par tour. Cette erreur n'excède généralement par 5%.

A1.3.2. Caractéristiques dynamiques :

Les caractéristiques dynamiques sont celles que l'on peut obtenir pendant le fonctionnement pas à pas, Elles sont déterminées par :

- des facteurs mécaniques comme l'inertie du rotor et de la charge appliquée. Il est évident qu'un moment d'inertie élevé s'oppose au déplacement rapide du rotor ou qu'il peut permettre de dépasser la valeur du pas. Lorsque l'inertie croît, la fréquence limite de fonctionnement correcte décroit.
- des facteurs électriques. L'augmentation de l'impédance du bobinage avec la fréquence s'oppose à l'établissement du courant dans l'enroulement que l'on doit alimenter pour passer au pas suivant. Le mode d'alimentation a donc une influence considérable

Le passage d'un pas au suivant s'accompagne donc d'oscillation plus au moins amortit.

Dans le plan défini par la fréquence d'alimentation au nombre de pas par seconde et par le couple utile, on trouvera, pour une valeur donné de l'inertie additionnel, deux courbes C=f(f). La première C_1, **(Figure A1.5)** est la limite de fonctionnement en moteur pas-a-pas. La seconde, C_2 interne a la précédente, est la limite de démarrage qui défini la zone dans laquelle le moteur pourra toujours démarrer en partant du repos pour atteindre directement la fréquence de fonctionnement désiré. Entre ces deux courbes, le moteur ne peut être alimenté directement sous cette fréquence. Il ne peut accéder à cette fréquence que par une montée

lente de l'accroissement du nombre de pas par seconde au dessus de la fréquence f2. De même, l'arrêt ne peut être obtenu que par une réduction suffisamment lente de la fréquence. Les courbes C_1 et C_2 dépendent du mode d'alimentation avec des caractéristiques supérieures à courant constant.

On notera que le moteur peut fonctionner en mode synchrone a des fréquences élevées comme un moteur synchrone classique avec le rotor lié a la vitesse du champ tournant, mais ce n'est plus un moteur pas a pas. Cette propriété peut être utilisée dans des machines spéciales.

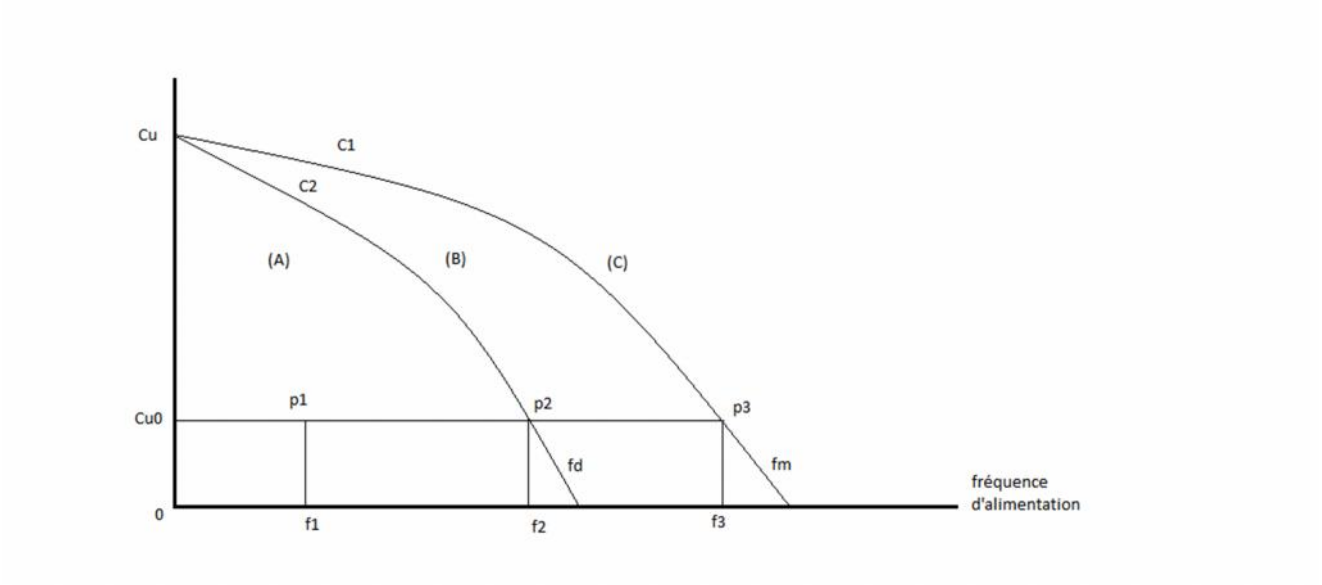


Figure A1.2 Zones de fonctionnement du moteur pas à pas

- **Zone (A) :** le moteur peut démarrer directement à sa fréquence d'utilisation
- **Zone(B) :** il doit d'abord démarrer a la fréquence maximale définie par la courbe C_2
- **Zone(C) :** le moteur ne peut plus suivre la fréquence

A1.4. Commande des moteurs pas à pas :

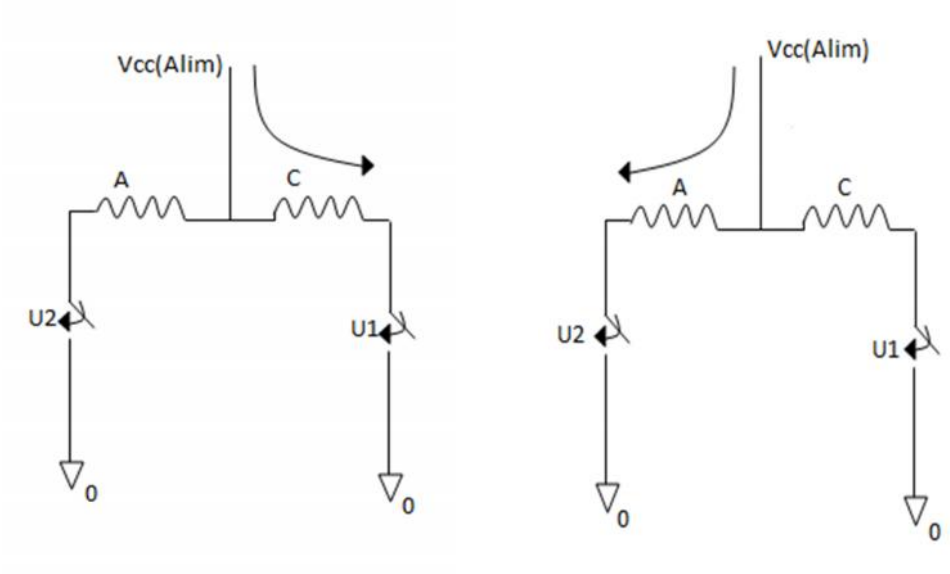
Plusieurs types de commandes peuvent être envisages lorsque l'on met en œuvre des moteur pas à pas. La moins onéreuse, mais pas obligatoirement la plus efficace et certainement la plus compliquée, fera usage de transistors. La plus sophistiquée fera appel à des circuits intégrés dédiés à ce type d'application.

Cette dernière solution sera la plus simple à mettre en œuvre et facilitera grandement l'interfaçage entre les moteurs et l'organe de commande, que ce soit un ordinateur de type PC ou un microcontrôleur **[9]**.

A1.4.1. Commande unipolaire de moteur pas à pas :

Une phase est alimentée si son bobinage est traversé par un courant nominal. Selon la configuration, le sens du courant traversant la moitié du bobinage donné peut être fixe **(Figure A1.6-a)** ou peut être inversé par la commande adéquate **(Figure A1.6-b).**

Dans le cas ou le courant de phase ne peut pas changer de sens (unidirectionnel) le moteur est dit unipolaire. Le circuit de commande est facile à réaliser. Mais dans ce cas, seulement la moitié du bobinage est alimentée à la fois ce qui donne un couple moins important **[10]**.



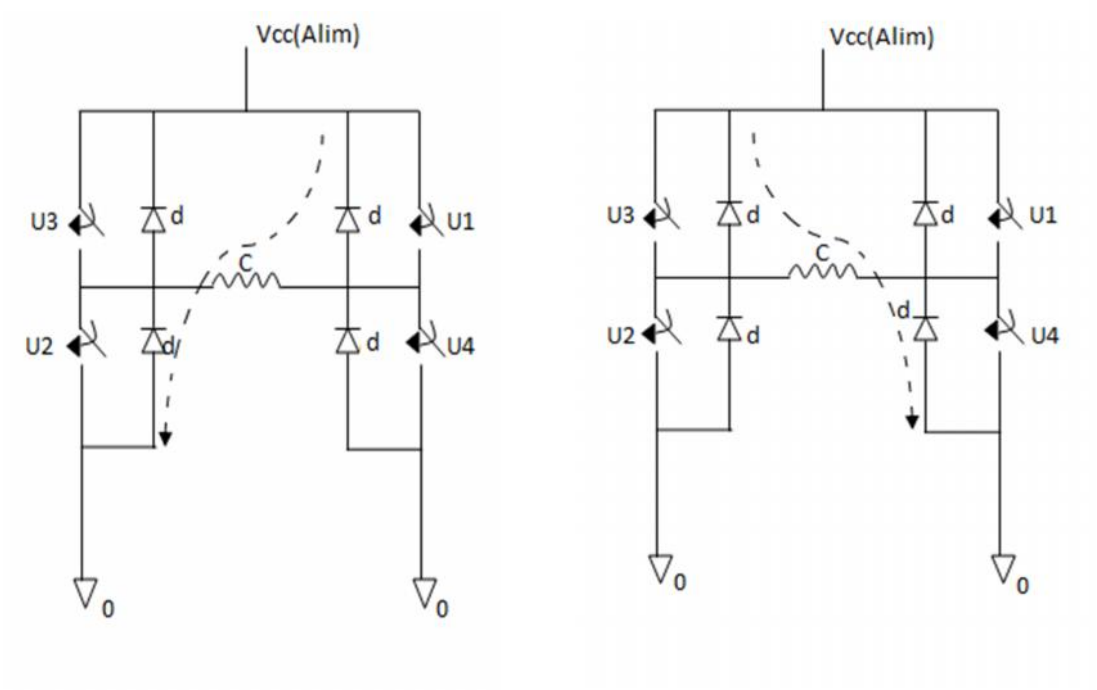
(a) Si U1 est fermé (b) Si U2 est fermé

Figure A1.3 Commande unipolaire de moteur pas à pas

A1.4.2. La commande bipolaire de moteur pas à pas :

Le circuit de commande des moteurs bipolaires est plus compliqué, mais on obtient un couple plus important. Pour pouvoir changer le sens de courant de phase, on utilise un circuit de commande en pont appelé pont H. le circuit typique d'une telle commande est représenté sur la **(Figure A1.7).** Les interrupteurs de la figure sont remplacés généralement par des transistors. Il faut veiller à ce que deux interrupteurs dans un même bras du pont ne soient pas fermés en même temps pour éviter le court circuit de l'alimentation. Dans ce cas le couple est plus important car la totalité d'un bobinage est alimentée à la fois.

La vitesse max de commutation ou bien le nombre de pas par seconde doit prendre en considération le temps nécessaire pour l'établissement de courant de phase **[10]**.



(a) Si U1 et U2 sont fermés et U3 et U4 sont ouvertes

(b) Si U3 et U4 sont fermés et U1 et U2 sont ouvertes

Figure A1.4 Commande bipolaire de moteur pas à pas

Dans le schéma de ce pont les interrupteurs U1, U2, U3 et U4 sont commandés, respectivement, par les signaux I1, I2, I3 et I4. Les commandes I1 et I4 ne doivent jamais être à l'état haut en même temps. Ainsi les deux transitoires d'un même bras ne sont jamais fermés en même temps. Il en est de même pour les commandes des interrupteurs U3 et U2.

Pour éviter d'avoir des courants inverse dans les transistors on monte des diodes de roues libres.

ANNEXE 2

A2.1. Les microcontrôleurs :

Un microcontrôleur ou PIC (**periphecal interface contrôler)** est une unité de traitement de l'information de type microprocesseur contenant tout les composants d'un système informatique, à savoir microprocesseur, des mémoires et des périphériques (ports, timers, convertisseurs…). Chaque fabricant à sa ou ses familles de microcontrôleur. Une famille se caractérise par un noyau commun (le microprocesseur, le jeu d'instruction …) **[12].**

A2.2. Le microcontrôleur PIC 16F84A :

Nous allons maintenant s'intéresser à la structure interne du PIC16F84A, avec lequel nous avons travaillé. Le PIC16F84A est un microcontrôleur de **MICROCHIP**, fait partie intégrante de la famille des Mid-Range dont la mémoire programme est de type flash, capable d'accepter une fréquence d'horloge maximale de 20Mhz **[13].**

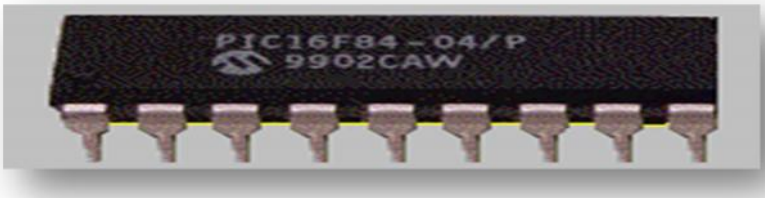


Figure A2.1 Microcontrôleur PIC 16F84A.

A2.2.1. Structure externe du PIC 16F84A :

Le PIC 16F84A est logé dans un boitier de 18 broches, la figure ci-dessous nous décrit le brochage du PIC 16F84A **[14].**

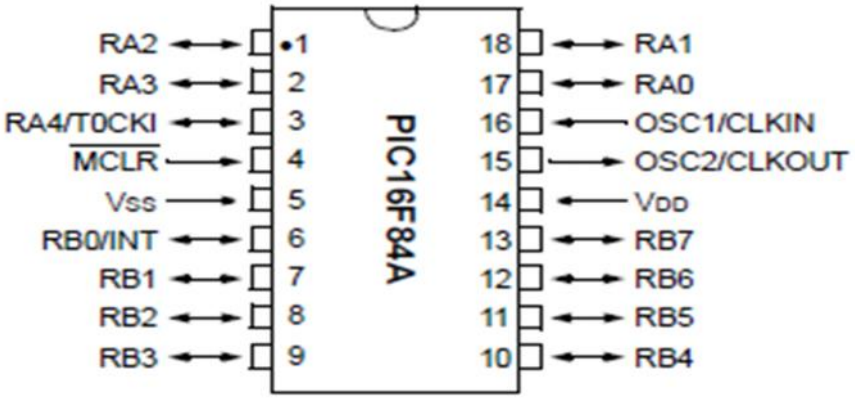


Figure A2.2 Schéma externe du PIC 16F84A.

- Vss et Vdd : broches d'alimentation et de masse.
- OSC1 et OSC2 : broches pour le signal horloge. elles peuvent recevoir un circuit RC ou un résonateur à quartz.
- MCLR : Reset (master clear).
- RA0 à RA4 : 5 entrées/sorties du port A.
- RB0 à RB7 : 8 entrées/sorties du port B.
- T0CKI : Entrée d'horloge externe du timer TMR0.
- INT : entrée d'interruption externe.

A2.2.2. Structure interne du PIC 16F84A:

La **Figure II.2** montre les différents composants du PIC 16F84A :

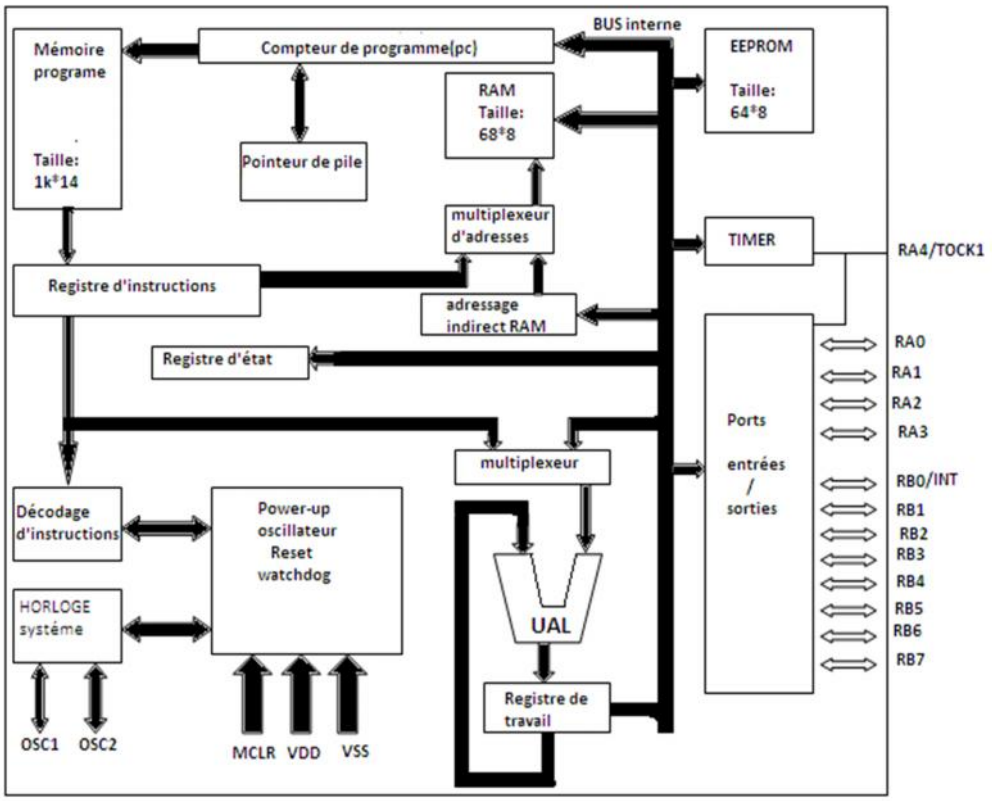


Figure A2.3 Structure interne du PIC16F84A.

La mémoire programme de ce PIC est de type flash et s'étend sur 1K mots. Chaque mot est sur 14 bits (Taille du code instruction). Ainsi on peut écrire des programmes allant à 1024 instruction dans ces PIC. La mémoire de donnés de type RAM et contient 68 octets. L'EEPROM contient 64 octets.
Il y a 13 pins d'entrée/sortie. Ces pins sont configurable en entrée et en sortie chacune a part.

Quelques pins sont multiplexées avec d'autres fonctions. Ces fonctions sont :

- Interruption extérieure.
- L'interruption sur changement sur PORTB.
- L'entrée horloge du TIMER0.

Les autres blocks fonctionnels du PIC sont, un :

- système d'initialisation à la mise sous tension (Power-Up Timer, …).
- circuit de génération d'horloge à partir du quartz externe (timing génération).
- compteur de programme (Program counter) et une pile (stack).
- bus spécifique pour le programme (Program Bus).
- registre contenant le code de l'instruction à exécuter.
- bus spécifique pour les données (Data Bus).
- Une unité arithmétique et logique(ALU) **[13].**

A2.3. Présentation du circuit intégré ULN2803 :

Des différents types de montage d'interfaces de moteur pas à pas sont existée pour leurs commande, nous intéressons à la commande par circuit intégré ULN2803.
Le circuit intégré ULN2803 comportant un octuple réseau de transistors Darlington connectés en parallèle deux à deux. Ce circuit supporte une tension maximale de 50v et un courant de 500mA par transistor. Le schéma interne de circuit est donné par la figure suivante : **[9]**

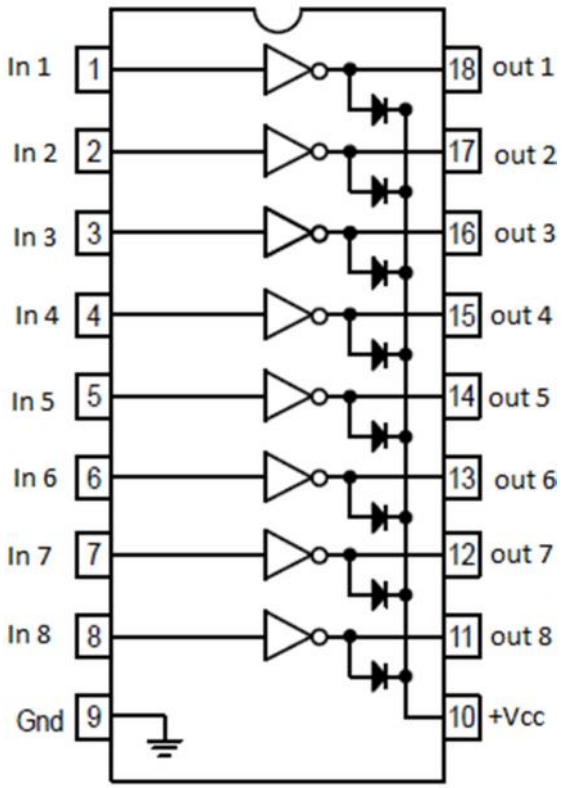


Figure A2.4 Structure interne du circuit intégré ULN2803.

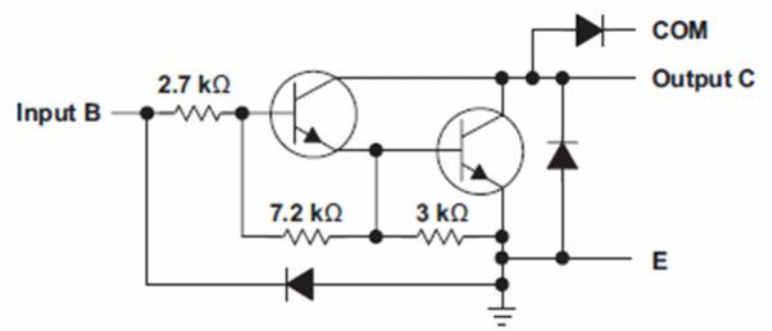


Figure A2.5 Constitution interne de l'un des huit éléments d'ULN2803.

Le montage devra être alimenté par une tension de 12v et sous un ampérage minimum de 1A.

A2.4. Présentation du codeur 74148N :

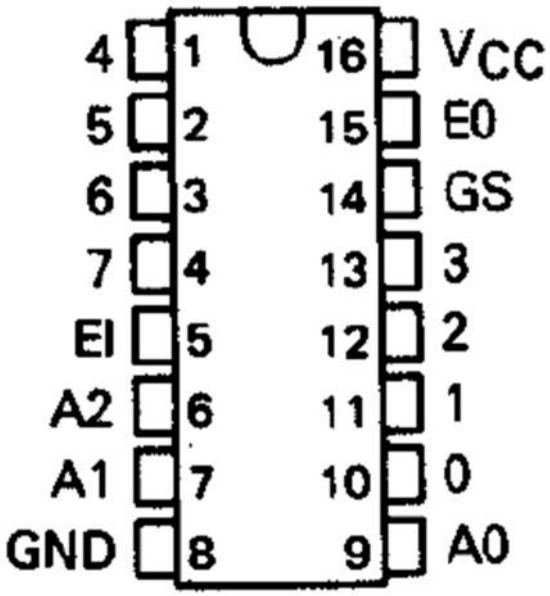


Figure A2.6 Schéma externe du codeur 74148N.

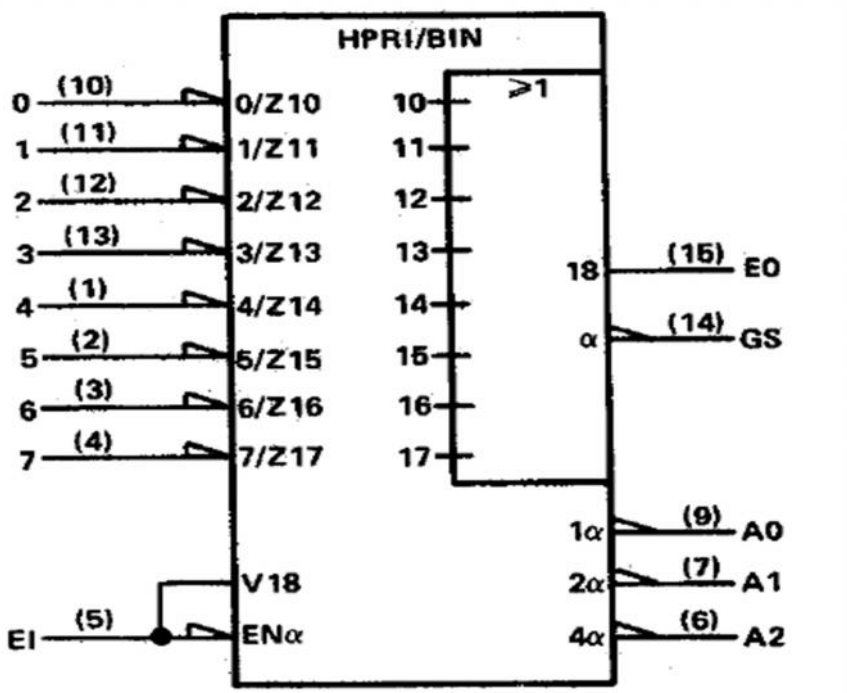


Figure A2.7 Structure interne du codeur 74148N

A2.5. Présentation du multiplexeur 74LS158N :

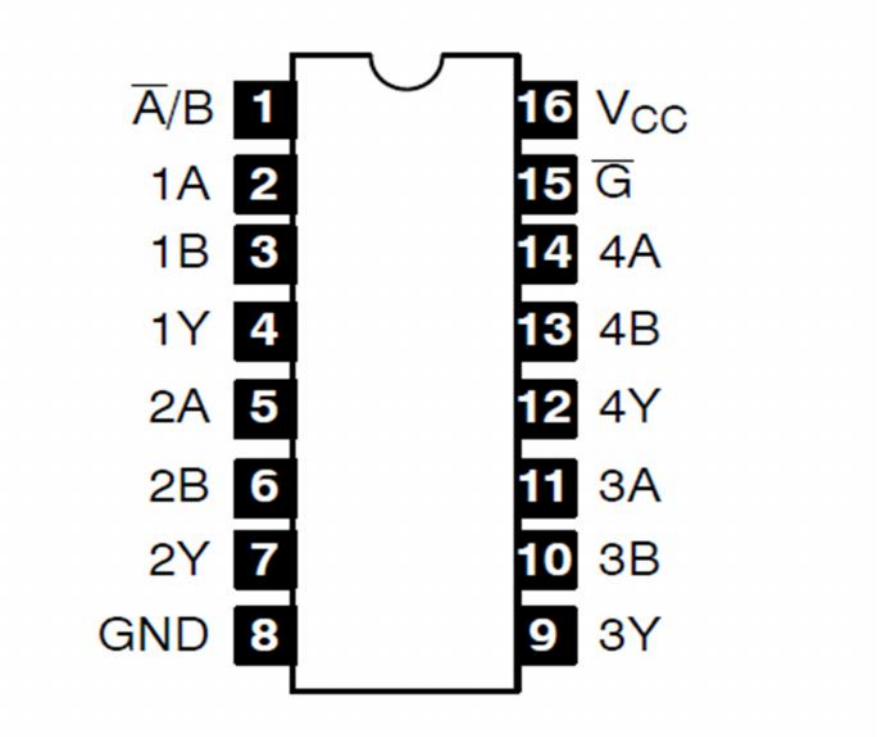


Figure A2.8 Schéma externe du multiplexeur 74LS158N

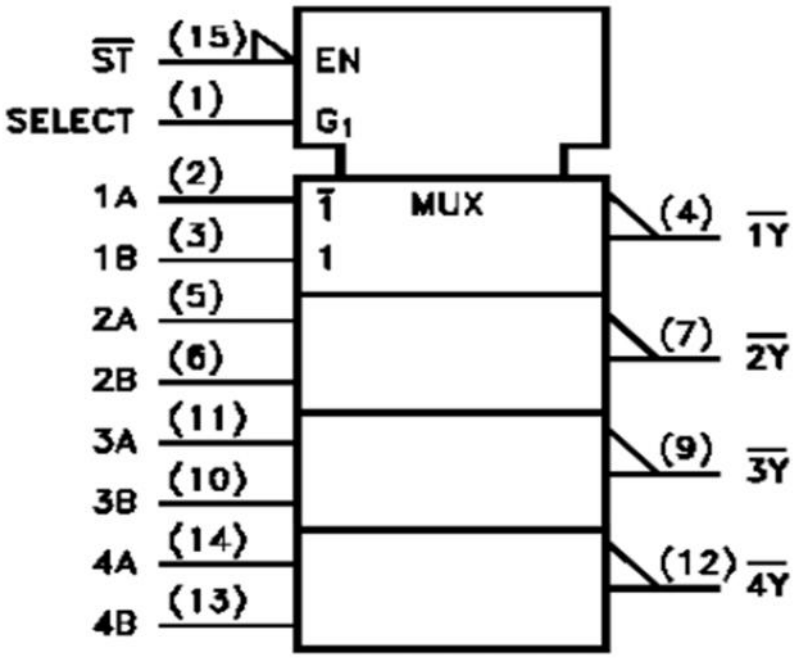


Figure A2.9 Structure interne du multiplexeur 74LS158N.

ANNEXE 3

A3.1 MikroC PRO:

MikroC est un puissant outil pour les micros PIC. Il est conçu pour fournir au client la solution la plus simple possible pour développer des applications pour systèmes embarqués sans compromettre les performances ou le contrôle.

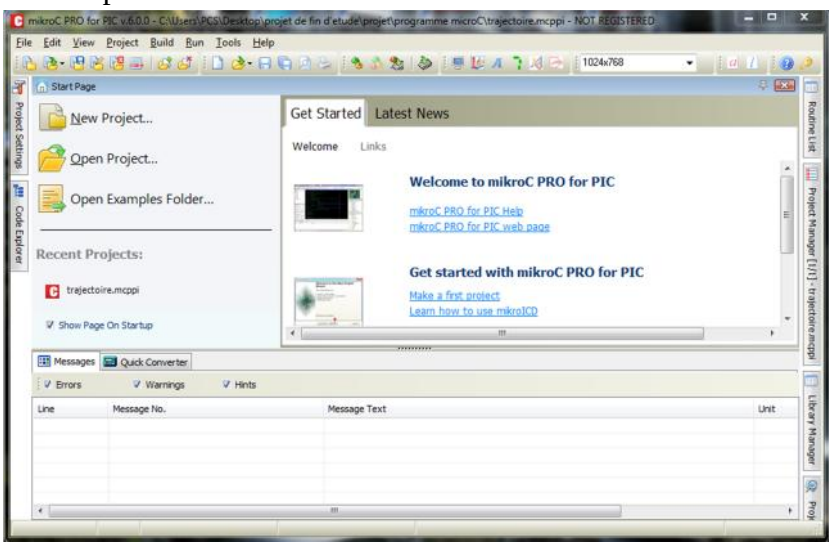


Figure A3.1 La fenêtre d'accueil de MikroC PRO.

A3.1 Création d'un projet sur MikroC PRO:

Dans la barre de menu, Project; on sélectionne New Project qui est un guide pour créer un projet.

Figure A3.2 Guide pour nouveau projet.

La première étape consiste à choisir le composant voulu dans la liste déroulante(PIC16F84), le nom du projet, la fréquence du projet.

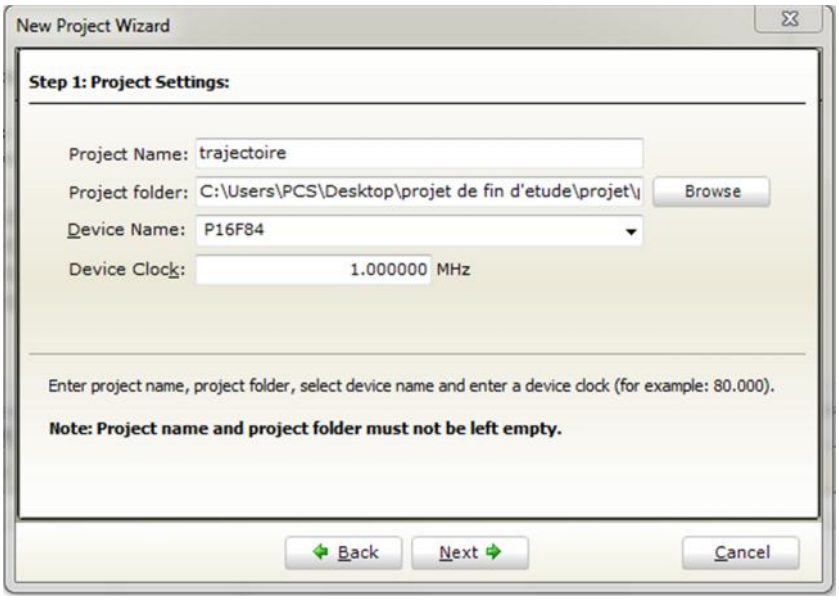


Figure A3.3 Choix du PIC et ses paramètres.

La deuxième étape consiste à sélectionner les fichiers que vous voulez ajouter au projet.

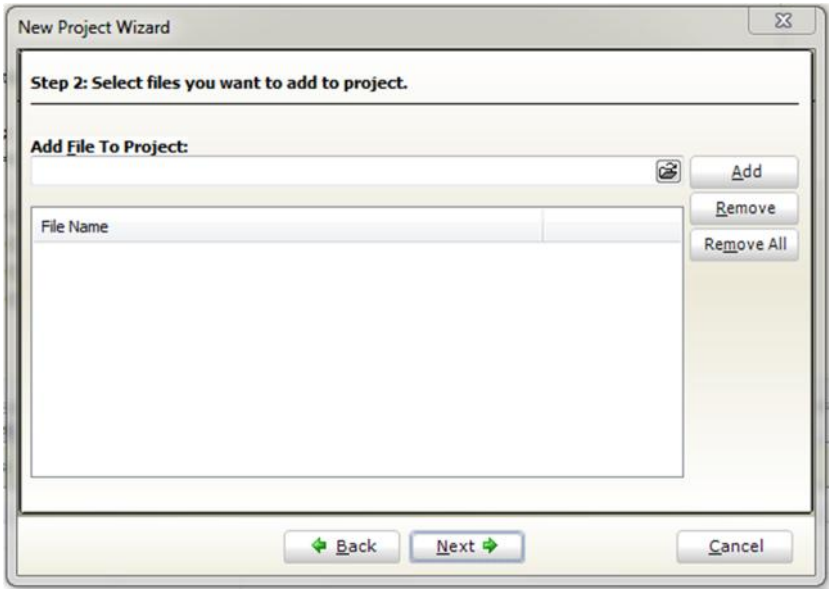


Figure A3.4 Inclure d'autres fichiers au projet.

La troisième étape consiste à sélectionner l'état initiale pour la bibliothèque manager.

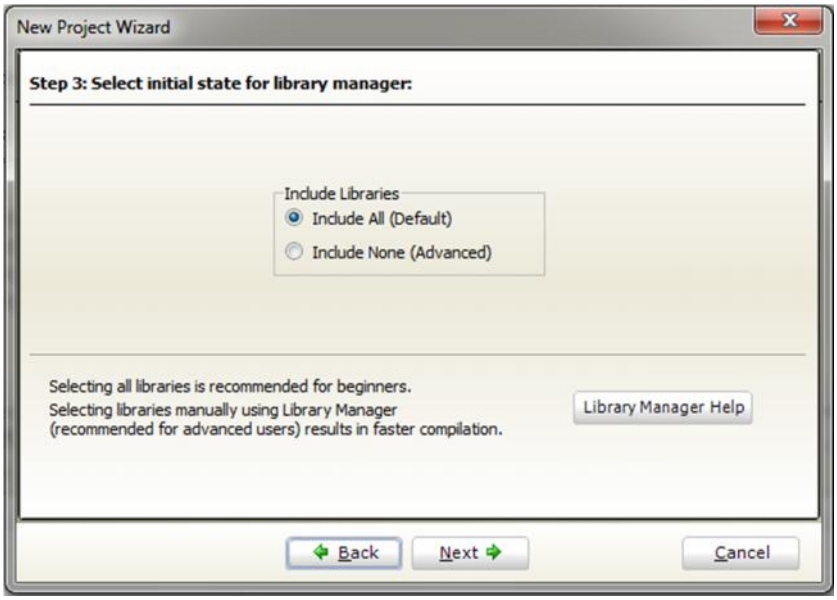


Figure A3.5 Sélection d'état initiale.

La quatrième étape nous confirme que le projet est crée avec succès, ainsi on clique sur finish.

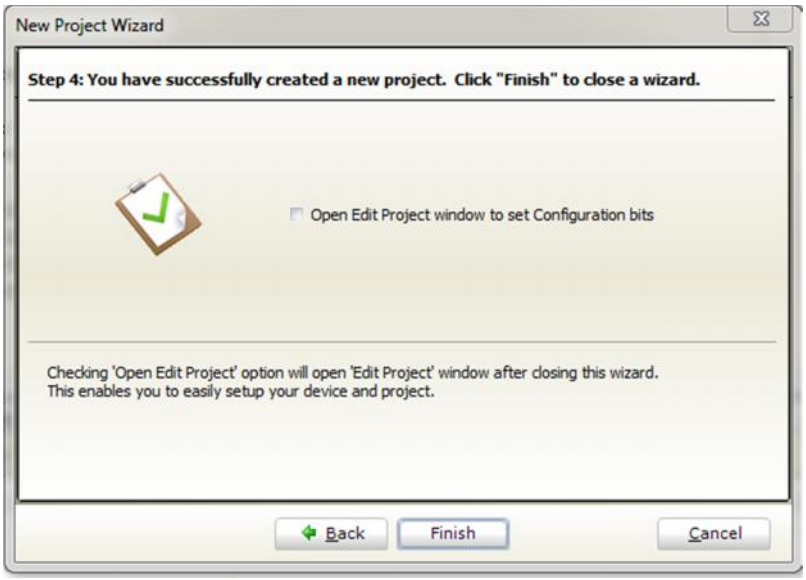


Figure A3.6 Création du projet avec succès.

Voici l'espace de travail de MikroC PRO :

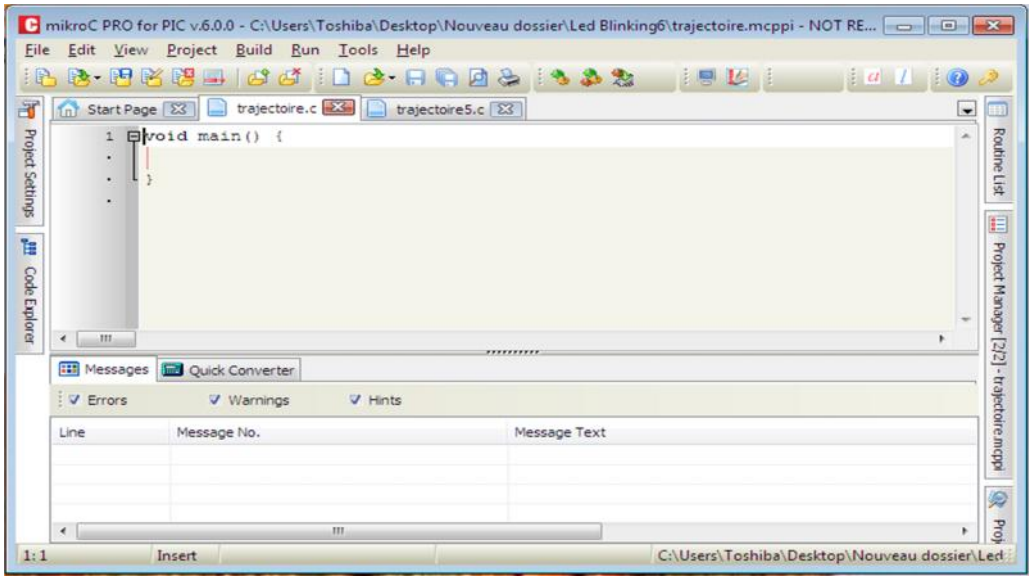


Figure A3.7 L'espace de travail de MikroC PRO.

Premièrement on doit écrire un code source C sur l'espace de travail, après on construit le projet en appuyant sur le bouton Build, et enfin on doit corriger les erreurs éventuelles que le compilateur détecte. Cela permet de générer un fichier en extension (.hex) à partir du code source. C'est ce dernier fichier qu'on va injecter au microcontrôleur.

A3.2 Isis proteus:

Le logiciel ISIS **(Intelligent Schematic Input Système)** de PROTEUS est principalement connu pour éditer des schémas électroniques. Par ailleurs, le logiciel permet également de simuler ces schémas ce qui permet de déceler certaines erreurs.

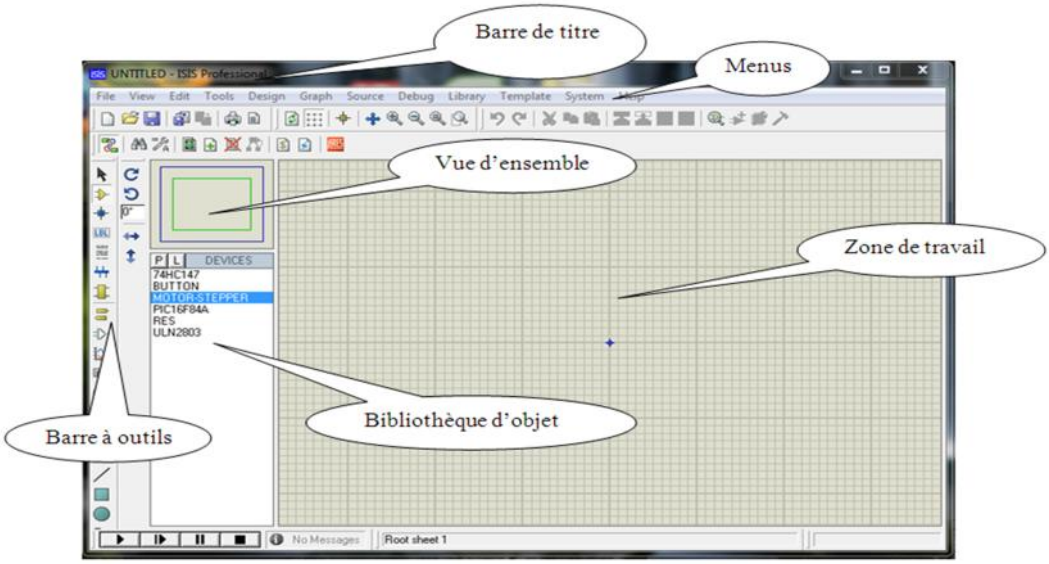


Figure A3.8 La fenêtre principale de travail sur ISIS.

A3.2.1 Sélection des composants à utiliser

Pour faire la sélection des éléments qu'on veut utiliser:

Un clique sur l'icone  (Component Mode) puis sur bouton parcourir la bibliothèque  (figure

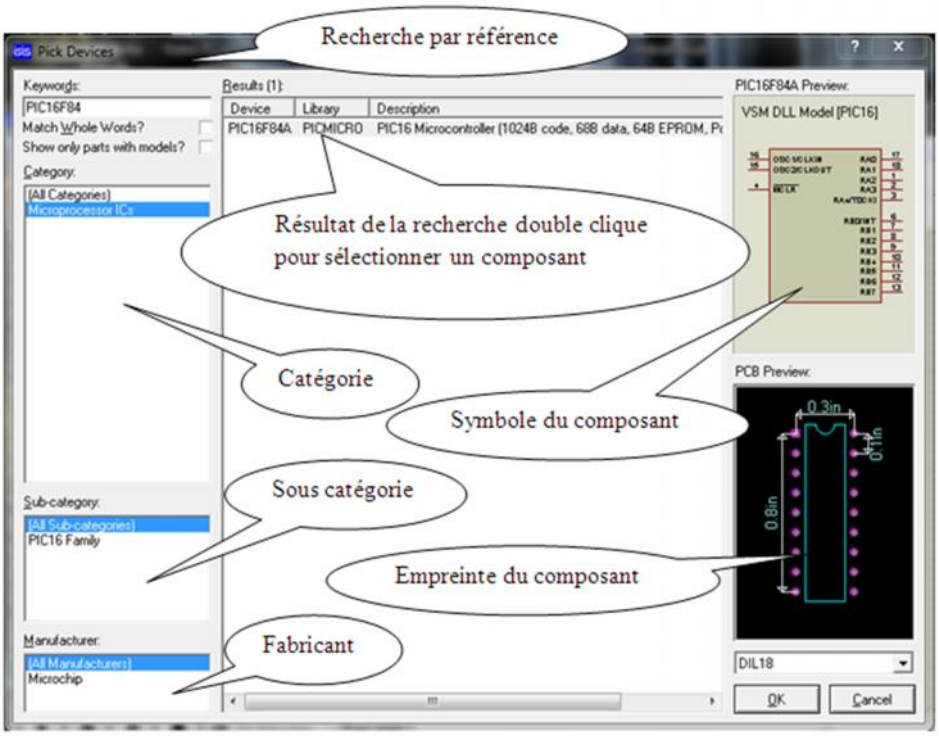


Figure A3.9 Bibliothèque ISIS.