MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI, TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET DE L'INFORMATIQUE
DEPARTEMENT D'ELECTRONIQUE

Mémoire de fin d'études

Présenté en vue de l'obtention
du Diplôme d'Ingénieur d'Etat en Electronique

Option : **Communication** et instrumentation

Thème:

Réalisation d'une carte de commande d'un robot manipulateur

Proposé et dirigé par :

M^{r} : LAGHROUCHE.M
M^{r} : TRIKI.A

Présenté par :

M^{r} : FAKHEUR RACHID
BELAID ADEL

Année universitaire 2008/2009

Soutenu le : 01/07/2009

Dédicace

Dieu merci, Dieu merci Dieu merci… !
Je dédie ce modeste travail à mes très chers parents pour leur aide et leur soutien tout au long de mes études, et qui ont fait de moi ce que je suis aujourd'hui et j'espère qu'un jour je serai capable de leur donner au moins le minimum car quoiqu'on face on arrivera jamais à leurs rendre tout.

A mes très chers frères : Kamel, Abderzek et Lounis à qui je Souhaite le succès dans leur vie.

A ma très chère sœur Cilia.

A la mémoire de mon cher grand père Said et son frére Hocine. « Que DIEU les accueille dans son vaste paradis ».

A tout mes cousins (es), tantes et oncles et leurs enfants.

A tout mes amis(es).

A toute la famille ***FAKHEUR***

FAKHEUR RACHID

Introduction Générale

Regroupant l'électronique, l'électrotechnique, la mécanique, l'automatique et l'informatique, la robotique est devenue depuis les années soixante-dix une discipline à part entière. Avec les grands progrès qu'elle a connus, la robotique s'impose de plus en plus au point ou il est très difficile d'imaginer une industrie moderne élaborée sans robots. Actuellement, les chercheurs parviennent à réaliser des robots légers, rapides, flexibles et pouvant même prendre des formes complexes et des dimensions réduites. Ceci montre que la robotique commence à devenir un art.

Les robots manipulateurs sont largement utilisés dans les systèmes industriels où ils jouent l'un des rôles les plus important, pour augmenter la productivité, réduire les coûts de production et amélioré la qualité de la production. Nous trouvons également l'utilisation des robots manipulateurs dans les hôpitaux où ils assistent dans de difficiles procédures chirurgicales ainsi que dans les milieux hostiles à l'homme, telle que les hautes températures et la radioactivité.

Commander un robot manipulateur consiste à lui permettre d'atteindre un état désiré ou de réaliser une tache bien spécifique avec une grande précision dans un environnement réel.

Le but de notre travail justement est de réaliser un système qui commande un bras manipulateur à 3 degrés de liberté (robot SCARA RP3) en utilisant des moteurs pas à pas permettant ainsi au robot manipulateur de se déplacer avec une grande précision. Le contenu de ce mémoire est présenté sous forme de quatre chapitres :

Le premier chapitre est consacré tout d'abord aux moteurs pas à pas. Nous avons présenté les différents types de moteurs pas à pas ainsi que leur principe de fonctionnement.

Le second chapitre est consacré à la modélisation des robots manipulateurs. Nous avons donné dans ce chapitre les notions de base de la robotique et de la modélisation géométrique qui a été retenu pour modéliser le robot SCARA RP3.

Le troisième chapitre est consacré à présenter la démarche que nous avons suivie pour la réalisation de la carte de commande du bras manipulateur ainsi que la partie soft du montage, nous y introduisons tous les organigrammes relatifs au déroulement du programme du PIC.

Ainsi après avoir établie un schéma synoptique du montage, nous développons un à un chacun de ses blocs avec toutes les informations nécessaires et les organigrammes correspondant.

Dans le dernier chapitre nous commençons par rappeler une à une les principales étapes à suivre pour la réalisation d'un circuit imprimé.

Nous donnons ensuite les figures montrant les tracés, coté cuivre et coté composants de la carte que nous avons réalisé et nous terminons par un tableau affichant la liste par catégories de tous les composants utilisé dans la réalisation de notre carte de commande.

Nous terminerons ce travail par une conclusion générale.

Chapitre I :

Généralités sur les moteurs pas à pas

Chapitre I : Généralités sur les moteurs pas à pas

I.1.Introduction :

La commande d'un robot a pour but de contrôler le mouvement des actionneurs selon une trajectoire programmée.

Les actionneurs peuvent être de nature électrique, hydraulique où pneumatique.Les actionneurs électriques les plus utilise en robotique sont : les moteurs pas à pas, les moteurs synchrone et les moteurs asynchrone.

Les moteurs pas à pas sont généralement les plus utilisé en robotique car leurs principal avantage est qu'ils sont facile à commander.

I.2. Moteurs pas à pas :

Les moteurs pas à pas sont des moteurs spéciaux utilisés pour commander avec une grande précision le déplacement d'un objet. Ils permettent de convertir directement un signal électrique numérique en un positionnement angulaire de caractère incrémental.

Comme leur nom l'indique, ces moteurs tournent par incréments discrets. Chaque incrément de rotation est provoqué par une impulsion de courant fournie à l'un des enroulements du stator.

La valeur de leur tension d'alimentation varie dans des grandes proportions, elle peut être comprise entre 3 Volt et plusieurs dizaines de volts. De même, selon la résistance ohmique de leurs bobinages, le courant consommé s'étendra dans une gamme allant de quelques dizaines de milliampères à plusieurs ampères.

Selon sa construction, un moteur pas à pas peut avancer de 90°,45°,18°, 7,5° où d'une fraction de degré seulement par impulsion.

I.3. Structure des moteurs pas à pas :

La **figure I.1** représente la structure d'un moteur pas à pas, Il est composé d'un :

- **Stator** : C'est la partie fixe du moteur, Il est constitué d'enroulements disposés sur son périmètre.
- **Rotor** : C'est la partie tournante du moteur, il est selon le type de moteur, constitué soit d'une pièce ferromagnétique, soit d'un aimant permanent, soit de la combinaison des deux.

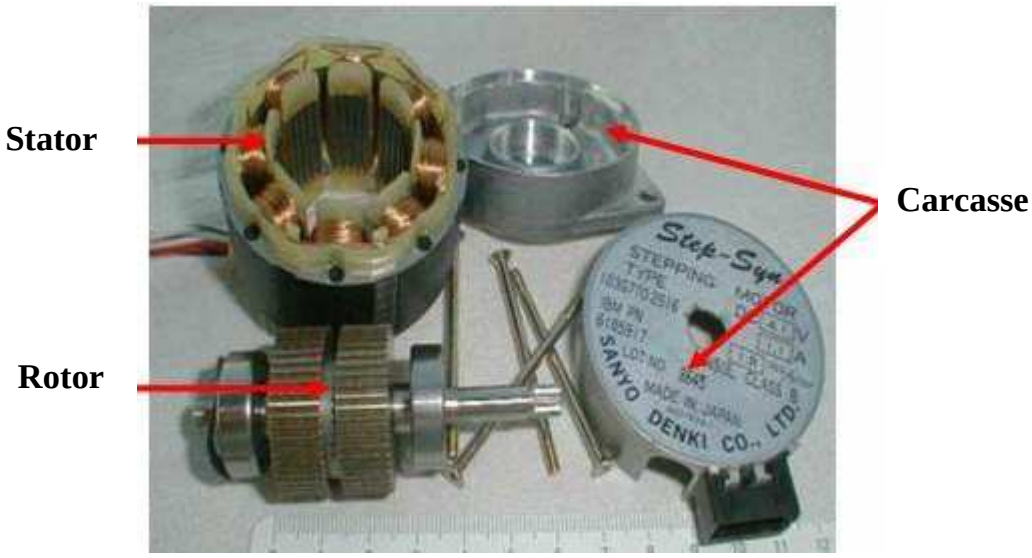


Figure I.1. Structure des moteurs pas à pas.

I.4. Principe de fonctionnement :

Le principe de fonctionnement d'un moteur pas à pas dépend de la source d'alimentation .Cette dérniere va alimenter les bobines du stator l'une après l'autre en génerant des impulsions .Lorsqu'une bobine est parcourue par un courant électrique, elle va s'aimanté c'est-à-dire elle va produire un pole nord et un pole sud.Le pole nord du rotor sera attiré par le pole sud du stator,pole crée par la circulation du courant dans le bobinage.Donc on aura une rotation du rotor d'un pas à chaque fois qu'on alimente le bobinage suivant.

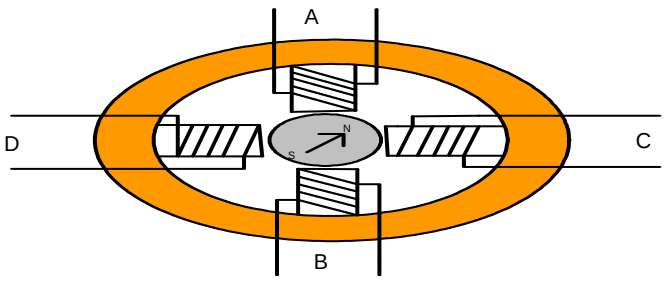


Figure I.2. Principe de fonctionnement d'un moteur pas à pas.

I.5. les modes de fonctionnement d'un moteur pas à pas :

Le moteur pas à pas peut fonctionner selon 3 modes :

1. mode monophasé. (**Figure I.3**)

2. mode biphasé. (**Figure I.4**)

3. mode demi-pas. (**Figure I.5**

Dans le mode monophasé, le couple n'est pas très important puisqu'un seul enroulement est alimenté pour effectuer un pas.

C'est dans le mode biphasé que le moteur développera la plus grande puissance (couple élevé) car les deux phases sont alimentées dans le même temps.

Le mode demi-pas permet de doubler le nombre de pas qu'un moteur peut effectuer par tour .En effet, dans ce mode la commande du moteur est un mélange de mode biphasé et monophasé.

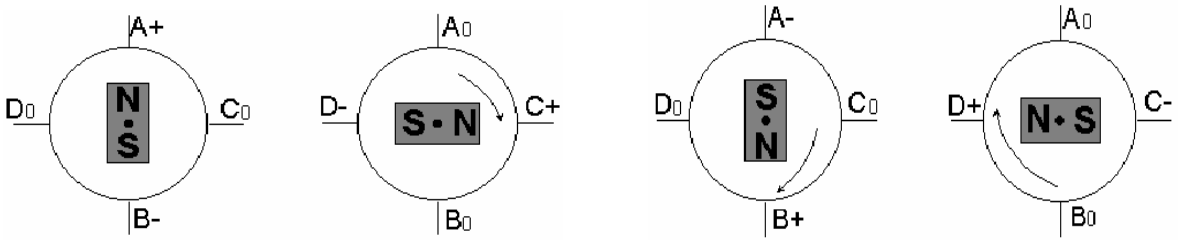


Figure I.3 Mode monophasé.

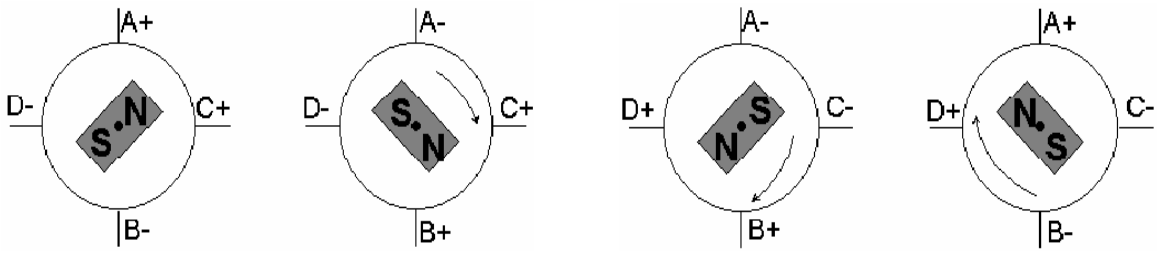


Figure I.4. Mode biphasé

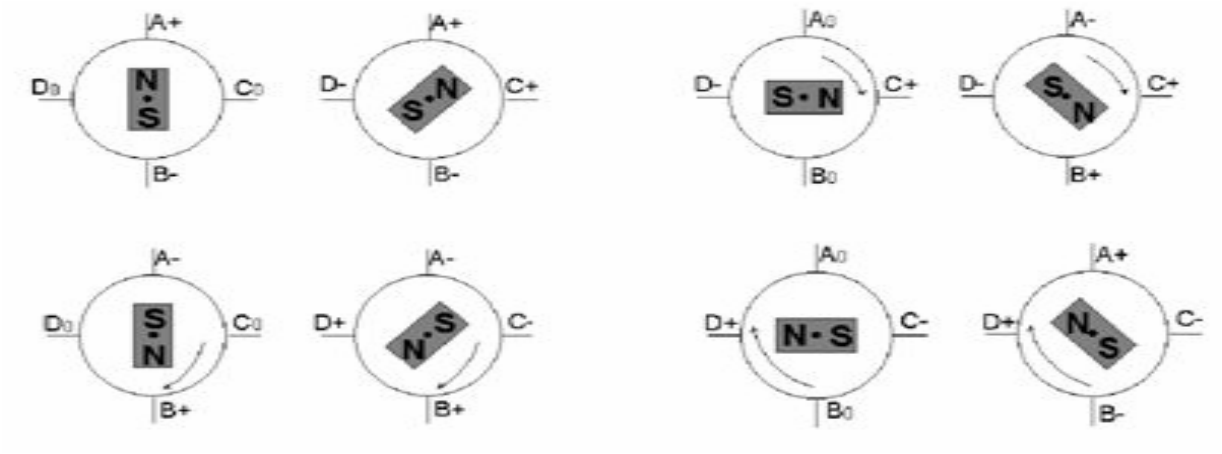


Figure I.5. Mode demi-pas.

I.6. Les différents types de moteurs pas à pas :

Il existe trois types de moteur pas à pas :

- ✓ Moteur pas à pas à aimant permanent.
- ✓ Moteur pas à pas à réluctance variable.
- ✓ Moteur pas à pas hybride.

I.6.1. Les moteurs à aimant permanent :

Les moteurs à aimant permanant sont constitués d'un stator supportant les bobinages et d'un rotor magnétique (aiment bipolaire).

Cette catégorie de moteurs se subdivise en deux types :

a) Le moteur unipolaire.

b) Le moteur bipolaire.

I.6.1.a. Moteur pas à pas unipolaire :

Les enroulements sont constitués de deux fils dont les extrémités opposées sont continuellement reliées au plus de l'alimentation, c'est-à-dire que les bobines sont à point milieu contrairement aux moteurs bipolaire. (Voir **figure I.6**)

Les enroulements à point milieu sont alimentés avec une polarité de même signe d'où la signification du terme "unipolaire". Le nombre de phases est égal au nombre de demi-enroulements.

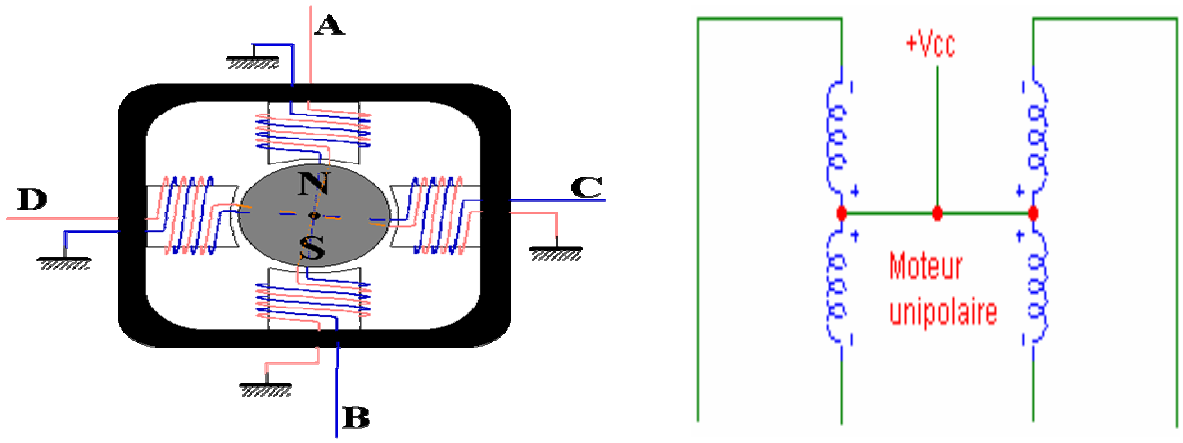


Figure I.6. Moteur unipolaire.

Comme pour chaque type de moteur, le moteur unipolaire peut être commandé en 3 modes :

1. Mode monophasé *:*

On alimente successivement chaque demi-enroulement (voir **figure I.7** et **tableau 1**)

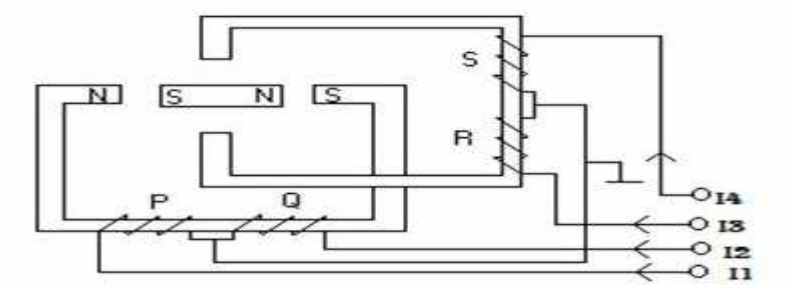


Figure I.7. Modes de fonctionnement du moteur unipolaire.

Séquences	Phase P I1	Phase Q I2	Phase R I3	Phase S I4	Angle
1	1	0	0	0	**0**
2	0	0	1	0	**90°**
3	0	1	0	0	**180°**
4	0	0	0	1	**270°**

Tableau 1.Séquences du mode monophasé.

2. Mode biphasé :

On alimente successivement 2 demi-enroulements *:* voir **figure I.7** et **tableau 2.**

Séquences	Phase P I1	Phase Q I2	Phase R I3	Phase S I4	Angle
1	1	0	1	0	**45°**
2	0	1	1	0	**135°**
3	0	1	0	1	**225°**
4	1	0	0	1	**315°**

Tableau 2. Séquences du mode biphasé

Remarque :

Pour inverser le sens de rotation d'un moteur pas à pas unipolaire il suffit d'inverser l'ordre d'alimentation des bobinages (les séquences)

3. Mode demi pas :

En associant le mode 1 et 2, on obtient le mode demi-pas, c'est à dire ici 8 pas par tour.

I.6.1.b. Moteur pas à pas bipolaire :

Pour un moteur bipolaire, les enroulements du stator n'ont pas de point milieu. Chaque borne de chaque enroulement est alimenté soit positivement, soit négativement, d'où la signification du terme : "bipolaire"

En inversant les polarités des enroulements statoriques, on inverse les pôles nord et sud du stator. Le nombre de phases est égal au nombre d'enroulements.

Pour un moteur bipolaire, le nombre de pas par tour est donné par :**Np=nombre de phases × nombre de pôles au rotor.**

Le sens de rotation d'un moteur pas à pas bipolaire dépend du sens du courant et de l'ordre d'alimentation des bobinages.

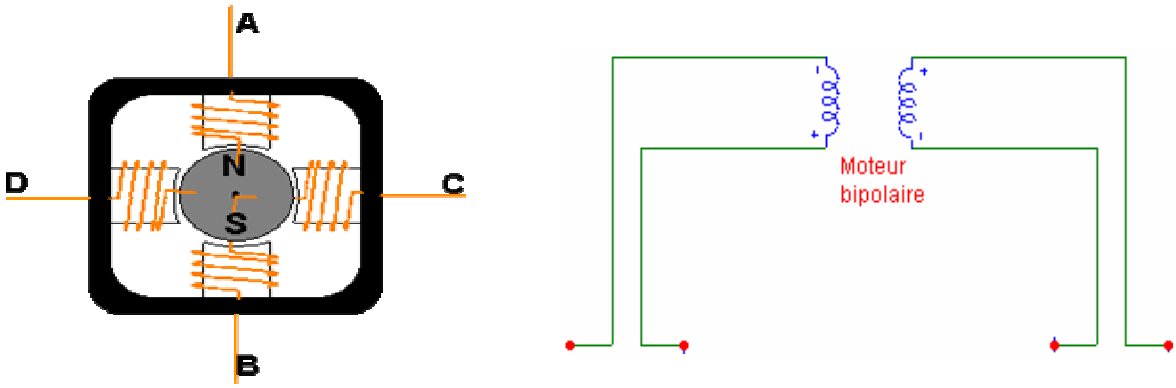


Figure I.8. Moteur bipolaire.

Comme pour le moteur unipolaire, le moteur bipolaire peut être commandé en 3 modes :

1. Mode monophasé*:*

On alimente un seul enroulement à la fois, voir **Figure I.9** et **Tableau 4.**

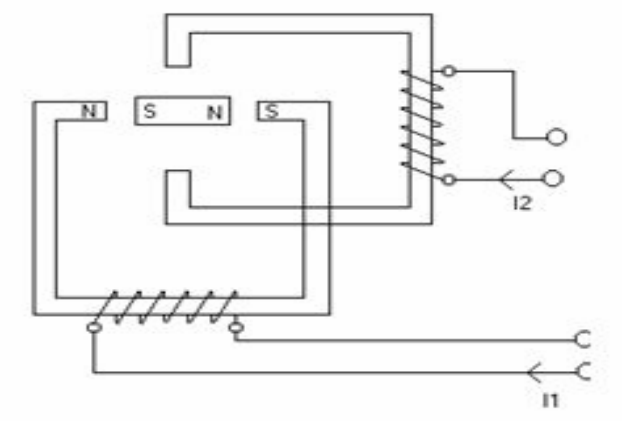


Figure I.9. Modes de fonctionnement du moteur unipolaire.

Séquences	I1>0	I1<0	I2>0	I2<0	Angle
1	1	0	0	0	**0**
2	0	0	1	0	**90°**
3	0	1	0	0	**180°**
4	0	0	0	1	**270°**

Tableau 4. Séquences du mode monophasé

2. Mode biphasé:

On alimente 2 enroulements à la fois, voir **tableau 5**:

Séquences	I1>0	I1<0	I2>0	I2<0	Angle
1	1	0	1	0	**45°**
2	0	1	1	0	**135°**
3	0	1	0	1	**225°**
4	1	0	0	1	**315°**

Tableau 5. Séquences du mode biphasé.

3. Mode demi pas :

En associant le mode 1 et 2, on obtient comme précédemment un séquencement en 1/2 pas, c'est à dire ici 8 pas par tour.

I.6.2. Les moteurs à réluctance variable :

Il s'agit d'un moteur qui comporte un rotor à encoches constitué par un matériau ferromagnétique non aimanté, se positionnant dans la direction de la plus faible réluctance et d'un stator qui comporte des encoches (dents) sur lesquels sont enroulés des bobinages. Le nombre de dents au rotor (Nr) et au stator (Ns) est obligatoirement différent (Voir **figure I.10**).

Lorsqu'on alimente une paire de bobines, le rotor se place de façon à ce que le flux qui le traverse soit maximal(ou reluctance minimale) car les lignes du champ passent plus facilement par le circuit ferromagnétique que dans l'air ; la perméabilité étant supérieur à 1000.

Remarque :

- Le rotor étant en fer doux, son mouvement est indépendant du sens d'alimentation des différentes phases, le choix de la séquence d'alimentation détermine son sens de rotation.
- Le nombre de pas par tour est donné par : **Np=Ns*Nr\(Ns-Nr)**

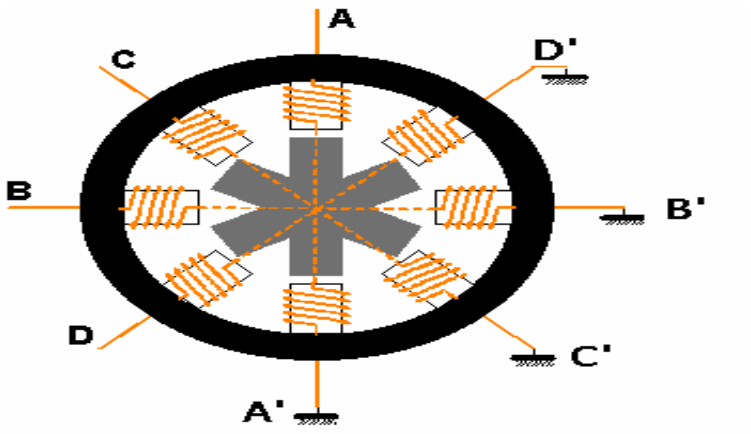


Figure I.10. Moteur à réluctance variable.

Comme pour le moteur à aimant permanant, le moteur à reluctance variable peut être commandé en 3 modes :

1. Mode monophasé:

Dans ce mode, on alimente une seul phase à la fois.

On alimentant successivement les phases AA', BB', CC', DD', AA'... ; le moteur va avancer de 24 pas par tour soit une avance de 15° par pas (voir **figure I.11**)

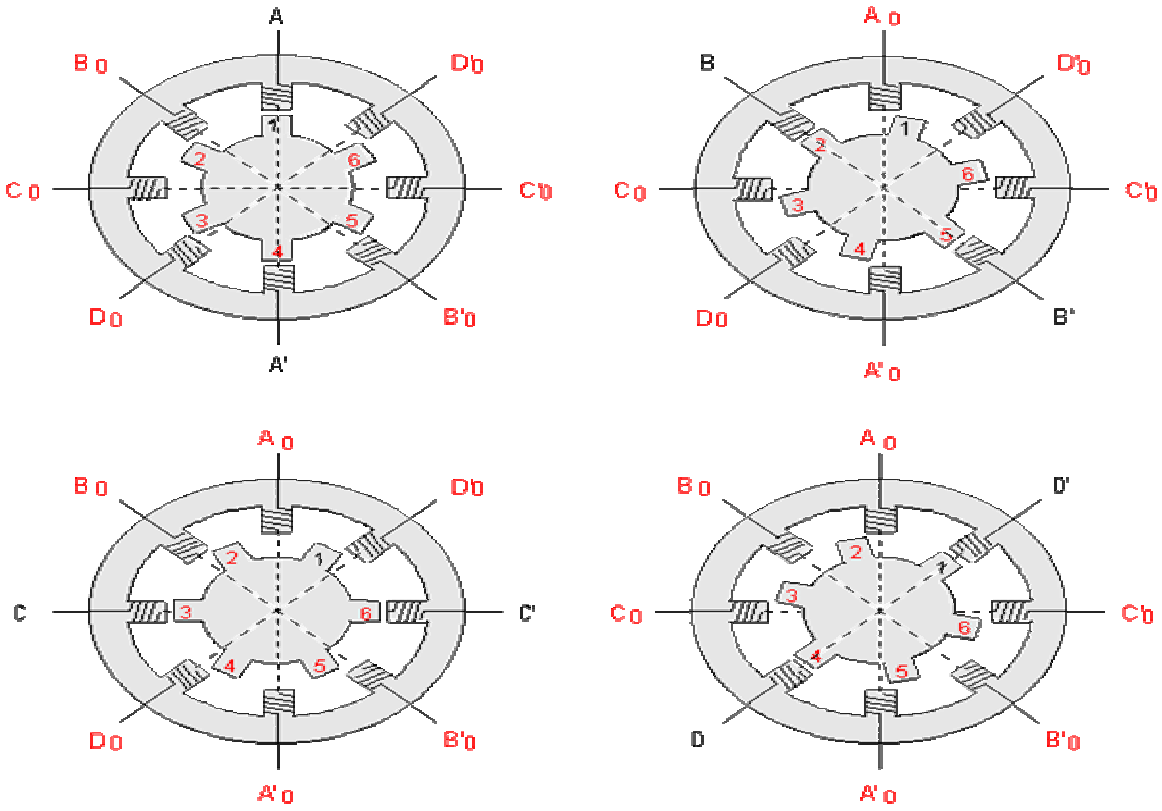


Figure I.11. Mode monophasé.

2. Mode biphasé:

On alimente 2 phases à la fois. (**Figure I.12**)

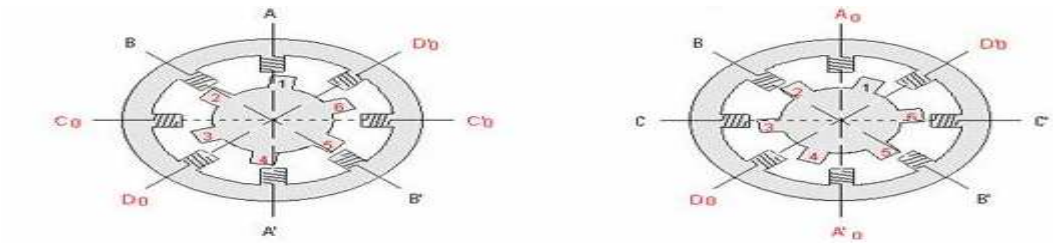


Figure I.12. Mode biphasé.

3.Mode demi-pas*:*

On combinant les séquences des deux modes, on obtient le fonctionnement du moteur en demi-pas .Dans ce mode le moteur va tourner de 48 demi-pas par tour, soit une avance de 7,5° par demi-pas. (Voir **figure I.13**)

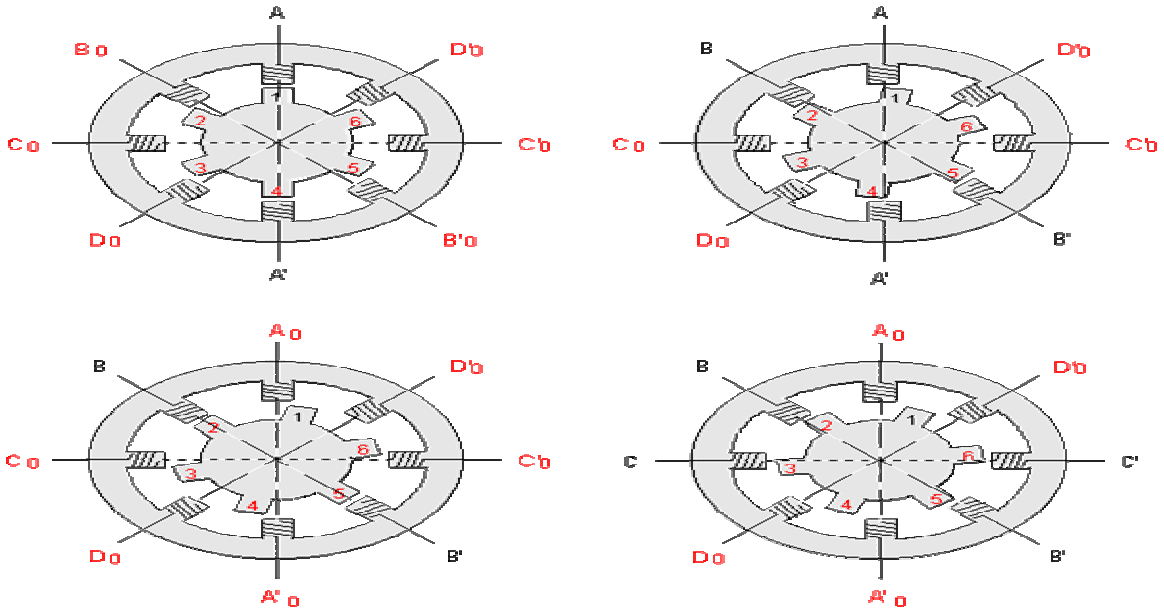


Figure I.13. Mode demi-pas.

I.6.3. Les moteurs hybrides :

Le moteur hybride est de loin le plus répandu des moteurs dans le domaine industriel. Le moteur hybride est une fusion des moteurs à aimant permanant et à reluctance variable.

Le rotor consiste en deux pièces ayant chacune trois dents entre les deux pièces se trouve un aimant permanent.

Le stator est constitue de quatre dents sur lesquelles sont enroulée les bobines. (Voir **figure I.14).**

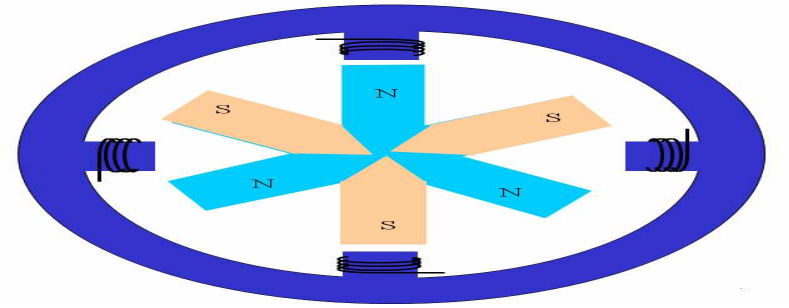


Figure I.14. Moteur hybride.

Son principe de fonctionnement repose sur l'effet d'un aimant permanant et d'une reluctance variable.

Il utilise alors la règle du champ maximum car l'aimant permanant du rotor essaie de minimisé la reluctance du champ magnétique lorsque l'une des bobines du stator est alimenté.

Les moteurs hybrides permettant d'avoir un couple plus important grâce aux aimants ainsi qu'un nombre de pas élevée. La majorité des moteurs hybrides ont 2 phases.

I.7. Comparaison des trois catégories de moteurs pas à pas :

Avant de conclure ce chapitre, on a jugé utile de dresser ce tableau qui compare entre les trois types de moteur pas à pas.

Type de moteur	Moteur à aimant permanent	Moteur à réluctance variable	Moteur hybride
Résolution (Nombre de pas/tour)	Moyenne	Bonne	Elevée
Couple moteur	Elevé	Faible	Elevé
Sens de rotation	Il dépend : - du sens du courant pour le moteur bipolaire - de l'ordre d'alimentation des bobines	Il dépend uniquement de l'ordre d'alimentation des bobines	Il dépend : - du sens du courant pour le moteur bipolaire - de l'ordre d'alimentation des bobines

I.8. Conclusion :

A travers ce chapitre on a exposé les différents types de moteurs pas à pas ainsi que leur fonctionnement .Cela nous a permis de mieux comprendre leurs principe de fonctionnement et ainsi d'en tirer profit pour aborder l'objectif principal de notre

travail qu'est la réalisation d'une carte de commande d'un robot manipulateur en utilisant comme articulateurs :les moteurs pas à pas.

Chapitre II :

Modélisation géométrique des robots manipulateurs

II.1. Introduction :

La commande d'un robot manipulateur consiste à asservir la situation (position et orientation) de son organe terminal à une situation imposée ,en agissant sur les actionneurs dont il est doté et modifier ainsi sa configuration.

Afin de contrôler et de commander correctement les actionneurs (les moteurs), il est impérative de faire une bonne modélisation qui consiste à représenter le comportement de la structure mécanique articulée par des équations algébriques.

A travers ce chapitre, on va présenter toutes les notions de base de la robotique qui nous permettrons de réaliser une bonne modélisation du robot manipulateur (robot SCARA RP3).

II.2. Morphologie générale d'un robot manipulateur:

Les robots manipulateurs se présentent en générale sous forme de structure mécanique poly-articulée se terminant par un organe terminal (poignet). Comme le montre la **figure II-1**

Le robot manipulateur est destiné à réaliser certaines opérations, sa commande est basée sur la spécification de la tache à effectuer. Dans ce cas, le problème se ramène à la formulation des ordres qu'il doit réaliser avec une grande précision et en un temps suffisamment court.

Le robot manipulateur est généralement constitué de trois éléments essentiels :

a)La Base :

C'est le socle sur lequel repose les différentes parties du robot.

b) Les Articulateurs :

C'est l'organe qui anime la structure mécanique par la conversion de l'énergie source en énergie mécanique.

Le rôle des articulateurs est d'amener et positionner l'organe terminal du robot en un lieu précis.

c) L'organe Terminal (poignet) :

C'est la partie essentielle du robot destinée à manipuler des objets ; et c'est l'outil qui lui permet d'atteindre tous les points de son environnement. On utilise généralement une pince adapté à la géométrie de l'objet qu'on veut saisir.

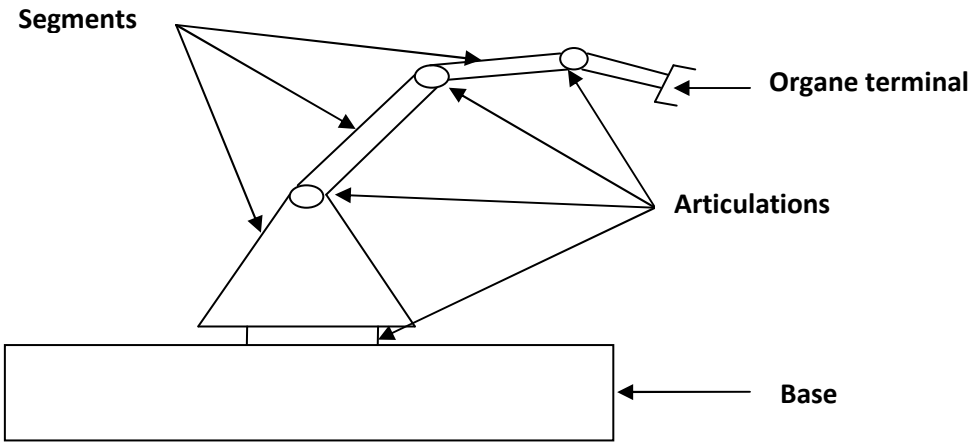


Figure II.1. Morphologie générale d'un robot manipulateur.

II.3. Description géométrique d'un bras manipulateur :

Un robot manipulateur à structure ouverte simple (**Figure II.2**) est composé de **n+1** corps rigides notés : $\mathbf{C_0 \ldots C_j \ldots C_n}$ et de **n** articulations où liaisons. Le Corps $\mathbf{C_0}$ constitue la base du robot tandis que le corps $\mathbf{C_n}$ porte l'organe terminal. L'articulation **J** relie le corps $\mathbf{C_j}$ au corps $\mathbf{C_{j-1}}$, le mouvement de chaque articulation produit le mouvement relatif du corps.

Les articulations peuvent être de deux types :

✓Articulation prismatique notée **(P)** : lorsque le mouvement effectué est une translation.

✓Articulation rotoide notée **(R)** : lorsque le mouvement effectué est une rotation.

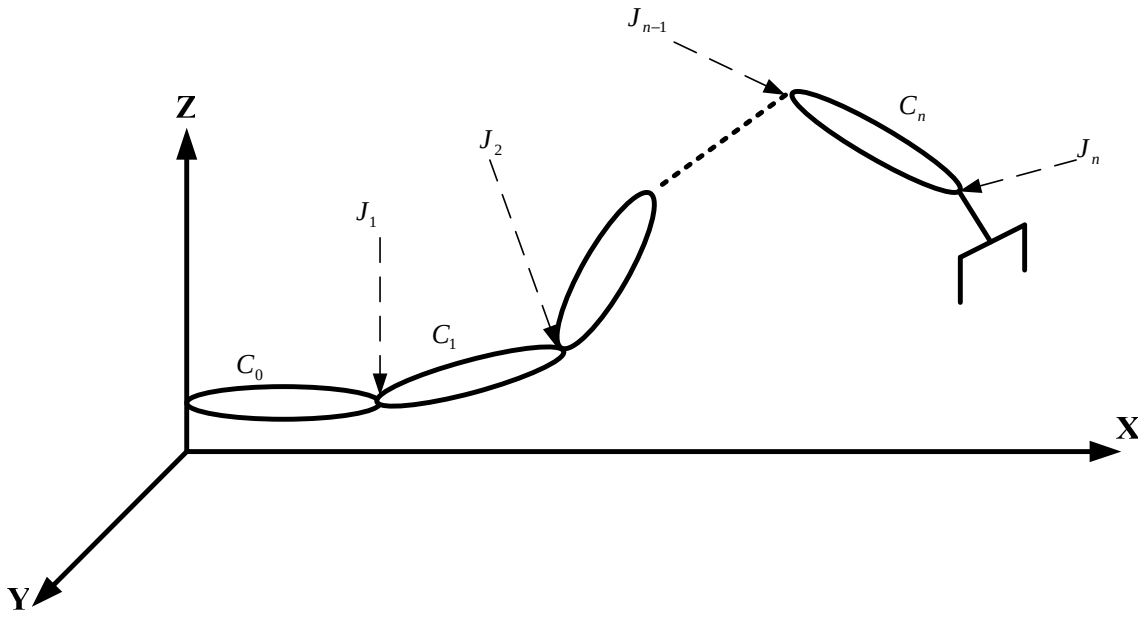


Figure II.2. Description géométrique d'un bras manipulateur

II.4. Mouvement du robot :

II.4.1 Degrés de liberté :

En robotique, le nombre de degrés de liberté (N.D.L) d'un robot manipulateur est le nombre de paramètres indépendants nécessaires pour décrire la situation de l'organe terminal. Ils représentent les mouvements élémentaires (rotations où translations) pour situer un objet lié à l'extrémité de la structure mécanique.

II.4.2 Espace des coordonnées :

On distingue deux types de coordonnées :

a)Coordonnées opérationnelles :

Elles permettent de caractériser et de définir la position de l'organe terminal, pour cela on définit un espace appelé espace opérationnel qui a pour référence un repère (R_{op}) orthonormé ($o_1.x_1.y_1.z_1$) lié généralement à la base du robot manipulateur, de même il est nécessaire d'introduire un autre repère (R_n) orthonormé (o_n,x_n,y_n,z_n) lié à l'organe terminal afin de mieux définir la situation (position, orientation de l'organe terminal). Les coordonnées associées sont donc appelées coordonnées opérationnelles.

b) Coordonnées généralisées :

On appelle coordonnées généralisées la quantité définissant le mouvement relatif du corps (segment) C_i par rapport au corps C_{i-1} , on la note q_i .Elle peut être un angle autour d'un axe (liaison de rotation),distance (longueur) le long d'un axe (liaison de translation) .

II.4.3. Espace de travail d'un robot manipulateur :

L'espace de travail d'un robot manipulateur est défini comme étant l'ensemble des positions et des orientations accessibles par un repère particulier, lié en général à son organe terminal. Il est entièrement déterminé par la structure du bras manipulateur : longueurs des éléments rigides, amplitude maximale de la rotation et de la translation des articulations et du déplacement des bras

II.5. Notations :

Afin de faciliter la compréhension des différentes écritures mathématiques utilisées dans ce chapitre, nous avons défini les notations suivantes :

$\mathbf{R_i}$ **:** repère **i** .

P: point.

$^{i}\mathbf{P}$**:** Coordonnées de point dans le repère $\mathbf{n^o}$ **. i.**

$\vec{\mathbf{v}}$ **:** Vecteur

$^{i}\mathbf{V}$**:** Coordonnées de $\vec{\mathbf{v}}$ dans le repère **i.**

$\overrightarrow{OP}$ **:** Vecteur

$\mathbf{I_{op}}$**:** Coordonnées de $\overrightarrow{\mathbf{OP}}$ dans le repère **i.**

$\mathbf{R_{01}}$ **:** rotation du repère $\mathbf{R_0}$ vers le repère $\mathbf{R_1}$.

$\mathbf{M_{01}}$ **:** Matrice de passage homogène du repère $\mathbf{R_0}$ vers $\mathbf{R_1}$. Où bien :

$^{i-1}\mathbf{M_i}$ **:** Matrice de passage homogène du repère $\mathbf{R_i}$ vers le repère $\mathbf{R_{i-1}}$

II.6. Orientation d'un solide :

On définit l'orientation d'un solide **(S)** associé à un repère $\mathbf{R_1}$ **(voir figure II.3)** par rapport à un repère $\mathbf{R_0}$ comme suit :

$$\vec{\imath_1} = (\vec{\imath_1} \cdot \vec{\imath_0})\vec{\imath_0} + (\vec{\imath_1} \cdot \vec{J_0})\vec{J_0} + (\vec{\imath_1} \cdot \vec{k_0})\vec{k_0}$$

$$\vec{J_1} = (\vec{J_1} \cdot \vec{\imath_0})\vec{\imath_0} + (\vec{J_1} \cdot \vec{J_0})\vec{J_0} + (\vec{J_1} \cdot \vec{k_0})\vec{k_0}$$

$$\vec{k_1} = (\vec{k_1} \cdot \vec{\imath_0})\vec{\imath_0} + (\vec{k_1} \cdot \vec{J_0})\vec{J_0} + (\vec{k_1} \cdot \vec{k_0})\vec{k_0}$$

D'où :

$$\begin{pmatrix} \vec{\imath_1} \\ \vec{J_1} \\ \vec{k_1} \end{pmatrix} = \begin{bmatrix} \vec{\imath_1} \cdot \vec{\imath_0} & \vec{\imath_1} \cdot \vec{J_0} & \vec{\imath_1} \cdot \vec{k_0} \\ \vec{J_1} \cdot \vec{\imath_0} & \vec{J_1} \cdot \vec{J_0} & \vec{J_1} \cdot \vec{k_0} \\ \vec{k_1} \cdot \vec{\imath_0} & \vec{k_1} \cdot \vec{J_0} & \vec{k_1} \cdot \vec{k_0} \end{bmatrix} \begin{pmatrix} \vec{\imath_0} \\ \vec{J_0} \\ \vec{k_0} \end{pmatrix}$$

$$\begin{pmatrix} \vec{\imath_1} \\ \vec{J_1} \\ \vec{k_1} \end{pmatrix} = R_{10} \begin{pmatrix} \vec{\imath_0} \\ \vec{J_0} \\ \vec{k_0} \end{pmatrix}$$

Avec $\mathbf{R_{10}}$ matrice de rotation de $\mathbf{R_1}$ vers $\mathbf{R_0}$

De la même manière :

$$\begin{pmatrix} \vec{\imath_0} \\ \vec{J_0} \\ \vec{k_0} \end{pmatrix} = \begin{bmatrix} \vec{\imath_1} \cdot \vec{\imath_0} & \vec{J_1} \cdot \vec{\imath_0} & \vec{k_1} \cdot \vec{\imath_0} \\ \vec{\imath_1} \cdot \vec{J_0} & \vec{J_1} \cdot \vec{J_0} & \vec{k_1} \cdot \vec{J_0} \\ \vec{\imath_1} \cdot \vec{k_0} & \vec{J_1} \cdot \vec{k_0} & \vec{k_1} \cdot \vec{k_0} \end{bmatrix} \begin{pmatrix} \vec{\imath_1} \\ \vec{J_1} \\ \vec{k_1} \end{pmatrix}$$

$$= R_{01} \begin{pmatrix} \vec{\imath_1} \\ \vec{J_1} \\ \vec{k_1} \end{pmatrix}$$

Avec $\mathbf{R_{01}}$ matrice de rotation de $\mathbf{R_0}$ vers $\mathbf{R_1}$

Comme $\mathbf{R_{01}}$ et $\mathbf{R_{10}}$ sont des transformations inverses, il vient : $\mathbf{R_{01} = R_{10}^{-1}}$ de plus on remarque que $\mathbf{R_{01} = R_{10}^{T}}$ **(matrice transposée).**

II.6.1 Orientation d'un point P appartenant au solide S :

Soit **P** un point du solide **(S)** de coordonnée $(x_{p0} \quad y_{p0} \quad z_{p0})$ et $(x_{p1} \quad y_{p1} \quad z_{p1})$ dans les repères $\mathbf{R_0}$ et $\mathbf{R_1}$ respectivement (**Voir figure II.3).**

On déduit que :

$$(x_{p0} \quad y_{p0} \quad z_{p0}) = (x_{p1} \quad y_{p1} \quad z_{p1})R_{10}$$

D'où :

$$\begin{pmatrix} x_{p0} \\ y_{p0} \\ z_{p0} \end{pmatrix} = R_{01} \begin{pmatrix} x_{p1} \\ y_{p1} \\ z_{p1} \end{pmatrix} \Leftrightarrow {}^0P = R_{01}\,{}^1P$$

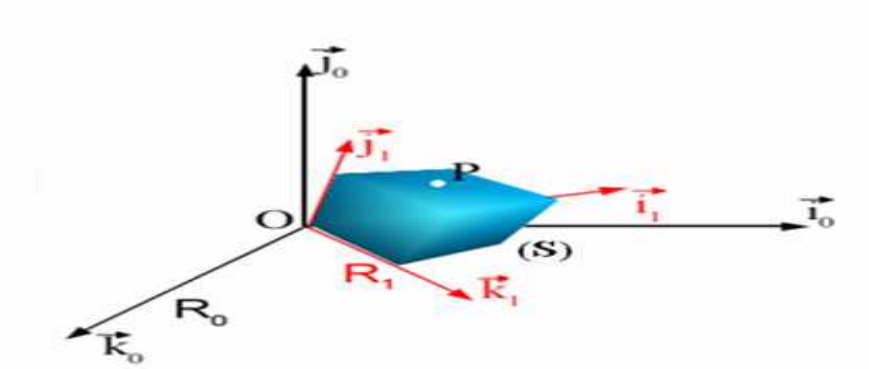


Figure II.3. Orientation d'un solide.

Remarque :

Par ailleurs si on considère plusieurs rotations:

- $\mathbf{R_{01}}$ rotation de $\mathbf{R_0}$ vers $\mathbf{R_1}$
- $\mathbf{R_{12}}$ rotation de $\mathbf{R_1}$ vers $\mathbf{R_2}$

On aura donc comme précédemment :

$$\mathbf{{}^1P = R_{12}\,{}^2P}$$

$$\mathbf{{}^0P = R_{01}\,{}^1P}$$

D'où :

$^0P = R_{01}\, R_{12}\, {}^2P = R_{02}\, {}^2P$

II.7. Transformation des repères:

Pour transformer (translation où rotation) un repère $\mathbf{R_i}$ et l'amener vers un autre repère $\mathbf{R_j}$, on fait appel à une matrice dite : **matrice de transformation homogène** ou bien **matrice de passage homogène** notée ^iT_j. Soit un repère $\mathbf{R_j}$ défini par trois vecteurs unitaires $\mathbf{a_j}$, $\mathbf{b_j}$, $\mathbf{k_j}$ et son origine $\mathbf{P_j}$ comme le montre la **figure II.4.**

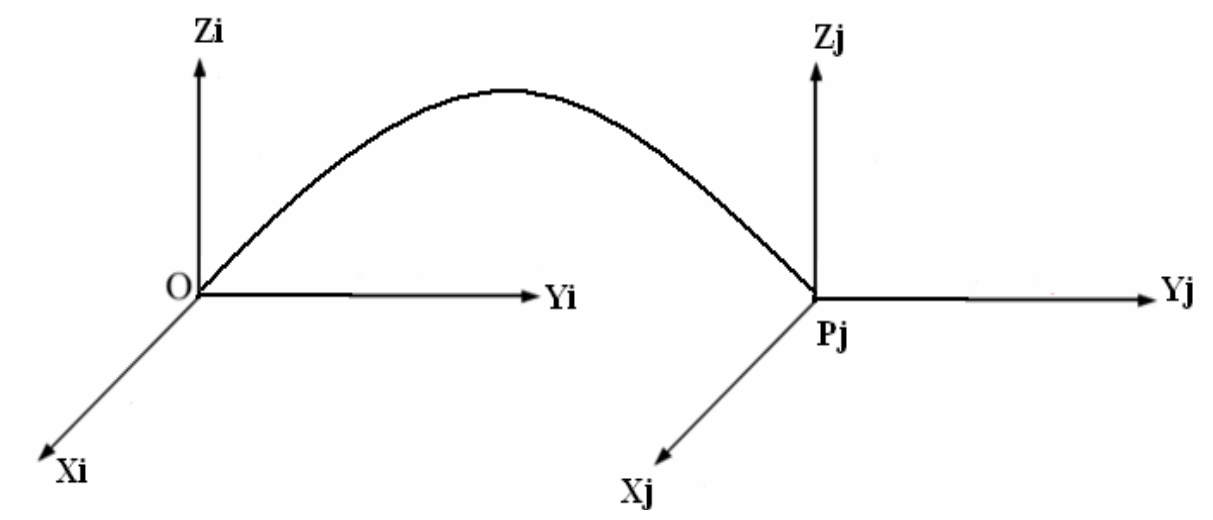


Figure II.4. Transformation de repére.

Les coordonnées des vecteurs unitaires et de l'origine du repère $\mathbf{R_j}$ dans le repère $\mathbf{R_i}$ sont respectivement données comme suite :

$$^ia_j = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}, {}^ib_j = \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}, {}^ic_j = \begin{pmatrix} c_i \\ c_j \\ c_z \end{pmatrix}, {}^ip_j = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$$

La matrice de transformation homogène est donnée comme suit :

$$^iT_j = \left(^ia_j\ {}^ib_j\ {}^ic_j\ {}^ip_j\right) = \begin{bmatrix} a_x & b_x & c_x & p_x \\ a_j & b_j & c_j & p_j \\ a_z & b_z & c_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

II.8. Les matrices homogènes :

C'est un outil mathématique permettant de calculer un changement de base en une seule opération matricielle. Pour cela, on rajoute une 4^{éme} coordonnée.

$$M_{01} = \begin{bmatrix} & R_{01} & & T_x \\ & & & T_y \\ & & & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^1P = \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} \leftarrow$$

Ajout d'une coordonnée supplémentaire

En utilisant les matrices homogènes le calcul des coordonnées d'un point P par rapport à un autre repère sera donné par la relation :

$$^0P = M_{01}\ ^1P$$

Remarque :

$^0P = M_{01}\ ^1P$ **:** est une relation de changement de coordonnées pour un point.

$^0V = R_{01}\ ^1V$ **:** est une relation de changement de coordonnées pour un vecteur.

$M_{02} = M_{01}\ M_{12}$: est une relation de composition entre plusieurs matrices homogènes.

$M_{01}M_{12} \neq M_{12}M_{01}$: la matrice homogène n'est pas commutatif

L'inverse d'une matrice de passage $\mathbf{M}^{-1}$ est donnée par :

$$M = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \qquad M^{-1} = \begin{bmatrix} R^T & -R^T T \\ 0 & 1 \end{bmatrix}$$

II.8.1. Application de la matrice de transformation homogène :

II.8.1.a. Matrice de transformation de translation :

Faisons subir au repère $\mathbf{R_i}$ une translation donnée par $\mathbf{d} = [\mathbf{a} \quad \mathbf{b} \quad \mathbf{c}]^t$

avec **d** coordonnées de l'origine du nouveau repère dans le repère $\mathbf{R_i}$ comme le montre la **figure II.5**.

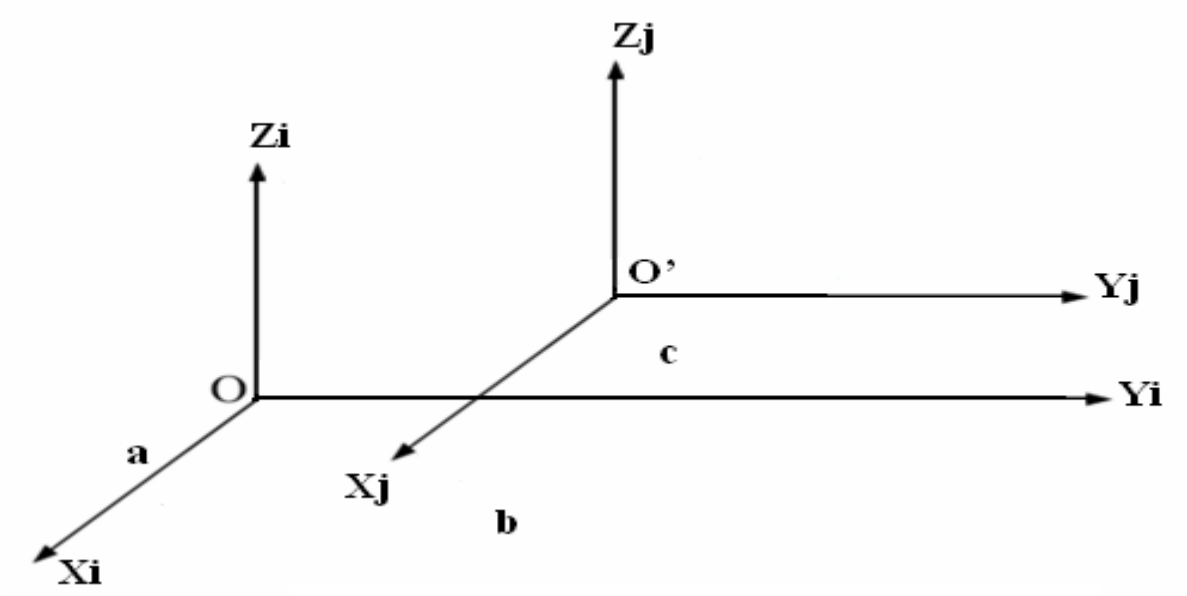


Figure II.5. Translation de repère

Cette translation notée **Trans (a, b, c)** est donnée par :

$$Trans(a\ ,b\ ,c) = {}^{i}T_{j} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Remarque : Toute translation sera représentée par une matrice unité de changement de repère. On la note : $\boldsymbol{M_i^{i+1} = [I]}$

II.8.1.b. Matrice de transformation de rotation :

La transformation d'un repère R_i vers un autre repère R_{i+1} dans le cas d'une chaine articulaire formée par des corps supposés indéformables s'effectue à l'aide d'une rotation d'angle θ de R_i autour de l'un de ses trois axes (x_i,y_i,z_i).

- **Rotation autour de l'axe X_i :**

Faisons subir au repère $\mathbf{R_i}$ une rotation $\boldsymbol{\theta}$ autour de l'axe $\mathbf{x_i}$ **(figure II.6)**

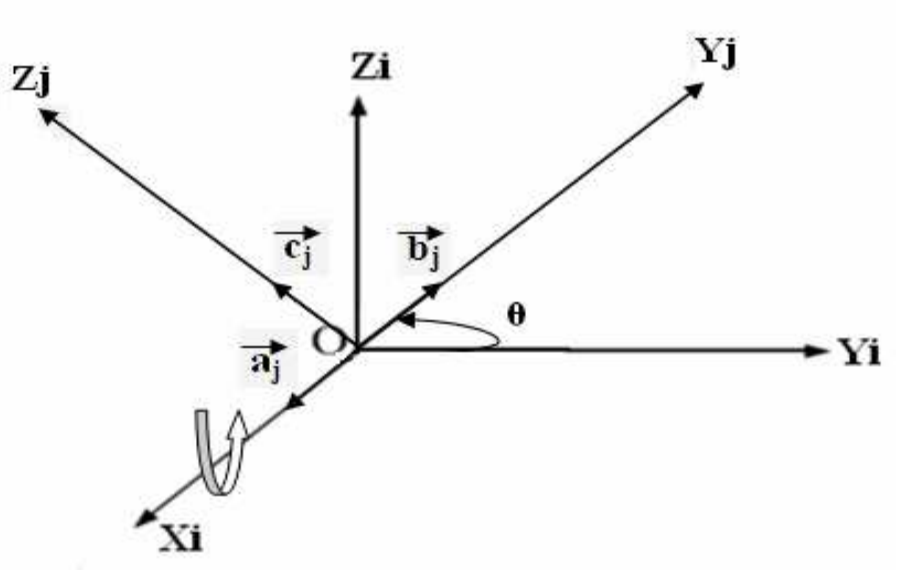


Figure II.6. Rotation autour de X_i

On trouve par projection :

$$\begin{cases} \vec{a_j} = \vec{a_i} \\ \vec{b_j} = \cos\theta.\ \vec{b_i} + \sin\theta.\ \vec{c_i} \\ \vec{c_j} = \cos\theta.\ \vec{c_i} - \sin\theta.\ \vec{b_i} \end{cases}$$

Pour simplifier l'écriture matricielle, posons :

$\cos\theta = C(\theta)$

Sin θ = S(θ)

La matrice de transformation de rotation autour de l'axe **X** notée **Rot (x , θ)** est donnée par :

$$Rot(x,\theta) = {}^{i}T_{j} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C(\theta) & -S(\theta) & 0 \\ 0 & S(\theta) & C(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De la même manière, on trouve les matrices de transformation de rotation autour de l'axe **Y et Z** qui sont donnée respectivement par :

$$Rot(y,\varphi) = {}^{i}T_{j} = \begin{bmatrix} C(\varphi) & 0 & S(\varphi) & 0 \\ 0 & 1 & 0 & 0 \\ -S(\varphi) & 0 & C(\varphi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Et

$$Rot(z,\varphi) = {}^{i}T_{j} = \begin{bmatrix} C(\varphi) & -S(\varphi) & 0 & 0 \\ S(\varphi) & C(\varphi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

II.9. Représentation de la rotation:

La rotation peut être définie par 3 rotations successives autour des axes $\mathbf{X_0}$, $\mathbf{Y_0}$, $\mathbf{Z_0}$, d'un repère $\mathbf{R_0}$.(voir **figure II.9**).

$$R_{01} = R_{(x_0,\theta_e)} . R_{(y_0,\theta_r)} . R_{(z_0,\theta_t)}$$

$$R_{01} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\theta_r & -S\theta_r \\ 0 & S\theta_r & C\theta_r \end{bmatrix} \begin{bmatrix} C\theta_t & 0 & S\theta_t \\ 0 & 1 & 0 \\ -S\theta_t & 0 & C\theta_t \end{bmatrix} \begin{bmatrix} C\theta_e & -S\theta_e & 0 \\ S\theta_e & C\theta_e & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{01} = \begin{bmatrix} C\theta_e C\theta_t & -S\theta_e C\theta_r + C\theta_e S\theta_t S\theta_r & S\theta_e S\theta_r + C\theta_e S\theta_t C\theta_r \\ S\theta_e C\theta_t & C\theta_e C\theta_r + S\theta_e S\theta_t S\theta_r & -C\theta_e S\theta_r + S\theta_e S\theta_t C\theta_r \\ -S\theta_t & C\theta_t S\theta_r & C\theta_t C\theta_r \end{bmatrix}$$

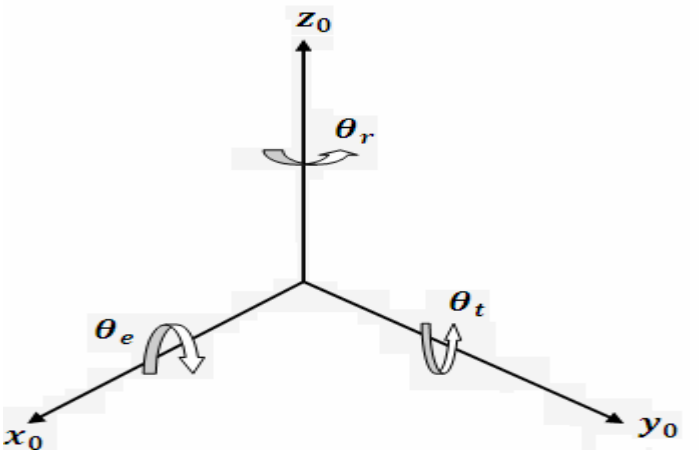


Figure II.9. Représentation de la rotation

II.10. Attitude d'un repère :

L'attitude du repère R_1 par rapport au repère R_0 est définie par la translation des origines et par la rotation R_{01} : $(\overrightarrow{O_0O_1}, R_{01})$

Les coordonnées du point P peuvent être exprimées dans R_0 à partir de ces coordonnées dans R_1 grâce à la connaissance de l'attitude de R_1 par rapport à R_0.

$$^0P = {}^0O_1 + R_{01}\ {}^1P$$

Avec ${}^0O_1$: coordonnées de O_1 dans le repère R_0

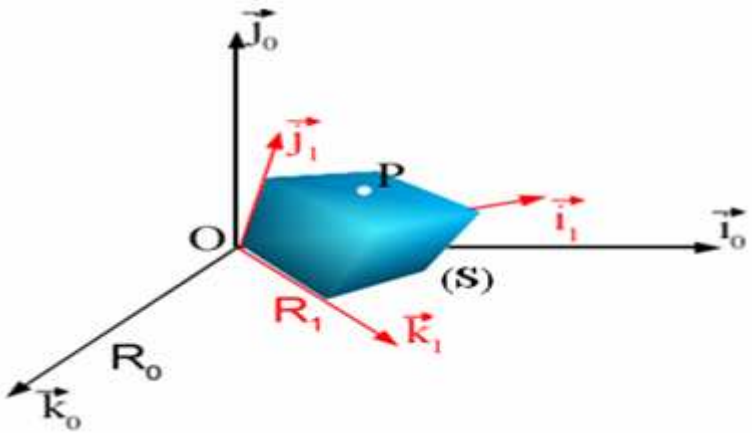


Figure II.10. Attitude d'un repère.

II.11. Modélisation des robots manipulateurs:

Pour exécuter une tache en un point donné de l'espace, le système artificiel combine ses performances matérielles et logicielles, pour que son organe terminal puisse prendre une succession de configurations (position, orientation, mouvement de l'outil) nécessaire à l'exécution de cette tache.

Afin de contrôler et de commander correctement les actionneurs (les moteurs), il est impérative de faire une bonne modélisation qui consiste à représenter le comportement de la structure mécanique articulée par des équations algébriques. On a trois types de modélisation :

Modélisation géométrique : du point de vue des positions.
Modélisation cinématique : du point de vue des vitesses.
Modélisation dynamique : considérant les efforts mis en jeu.
Ces modélisations permettent :

- D'exprimer les déplacements relatifs des différents éléments du mécanisme articulé les uns par rapport aux autres (matrices homogènes élémentaires).
- De spécifier les situations successives que doit prendre le repère associé à l'organe terminal du robot pour réaliser une tache donnée, ainsi que les vitesses correspondantes.
- De décrire et de contrôler les efforts mis en jeu lorsque le robot interagit avec son environnement.

II.11.1. Modélisation géométrique :

Qu'elle que soit la tache que doit réaliser un robot manipulateur, il est important de maitriser la position et le déplacement de son outil de travail (organe terminal).La modélisation géométrique ,qui est une configuration définie par un ensemble de variables articulaires traduisant les déplacements relatifs d'un corps par rapport au précédant,

Pour faciliter les calculs du modèle géométrique, on doit suivre ces étapes essentielles dans l'ordre suivant :

1. Fixer des repères à chaque corps du robot.

2. Calculer les matrices homogènes entre chaque corps.

3. Calculer la matrice homogène entre la base et l'organe terminal.

Avant d'expliquer les deux types de modélisation géométrique, on va d'abord parler de la convention de Denavite-Hertenberg qui est très importante pour réaliser une bonne modélisation du robot c'est-à-dire bien positionner les repères de chaque corps du robot (satisfaire l'étape 1 ci-dessus).

II.12. Convention de Denavite-Hertenberg (DH) :

C'est une méthode destinée à systématiser la modélisation de n'importe quel type de robot série, en représentant l'attitude d'un repère $\mathbf{R_i}$ par rapport à un repère $\mathbf{R_{i-1}}$.

Ses principaux avantages sont :

- Simplification maximale du modèle géométrique.
- Etablissement d'une norme reconnue par tous.

On peut représenter l'attitude d'un repère R_i par rapport à un repère R_{i-1} à l'aide de 4 paramètres uniques à condition de fixer les deux conditions de Denavite-Hertenberg (DH) suivante:

1. L'axe X_i de R_i est perpendiculaire à l'axe Z_{i-1} de R_{i-1}.

2. L'axe X_i coupe l'axe Z_{i-1}.

Afin de satisfaire ces deux conditions fondamentales, on doit suivre ces quatre décompositions élémentaires dans l'ordre suivant :

1. Rotation autour de z d'un angle $\boldsymbol{\theta_i}$

2. Translation le long de z d'une longueur $\mathbf{d_i}$

3. Translation le long de x d'une longueur $\mathbf{a_i}$

4. Rotation autour de x d'un angle $\boldsymbol{\alpha_i}$,

Avec :

θ_i **:** angle entre les axes x_{i-1} et x_i correspondant à une rotation autour de z_{i-1} .

α_i **:** angle entre les axes z_{i-1} et z_i correspondant à une rotation autour de x_i

a_i : distance entre z_{i-1} et z_i le long de x_i

d_i **:** distance entre x_{i-1} et x_i le long de z_{i-1} .

Les paramètres θ_i, α_i, a_i, d_i sont appelés : les paramètres de Denavite-Hertenberg.

La figure ci-dessous illustre les différentes décompositions élémentaires .

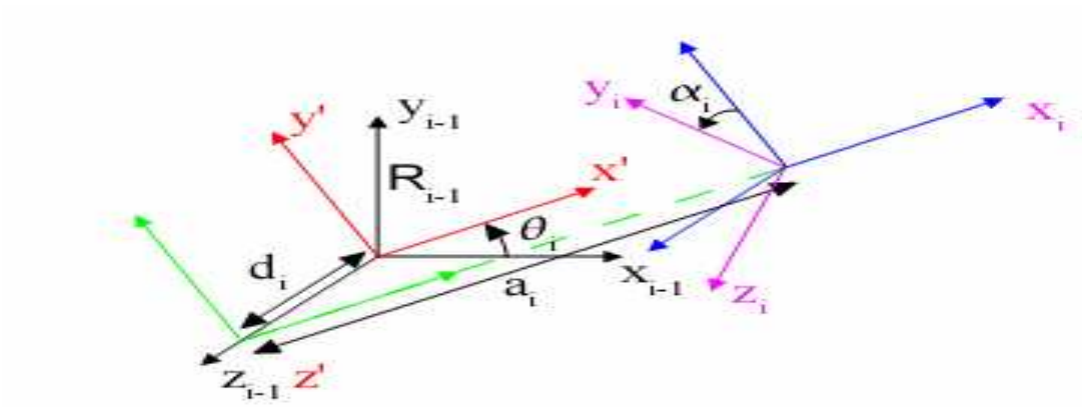


Figure II.11. Les décompositions élémentaires.

On peut facilement vérifier que les deux conditions de Denavite-Hertenberg sont ainsi respectées.

La matrice de transformation définissant le repère R_i dans le repère R_{i-1} est donnée par :

$$^{i-1}T_i = R_{(z_{i-1},\theta_i)}\, T_{(z_{i-1},d_i)} T_{(x_i,a_i)}\; T_{(x_i,\alpha_i)}$$

$$^{i-1}T_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{i-1}T_i = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

✓ **Cas particuliers :**

1. Les axes z_{i-1} et z_i sont parallèles :

Il y à une infinité de normales communes entre z_i et z_{i-1} . On place θ_i tel que :

$d_i = 0$; $\alpha_i = 0$ (figure II.12.a).

2. Les axes z_{i-1} et z_i se coupent :

Le point O_i est placé à l'intersection des axes z_{i-1} et z_i **(figure II.12.b).**

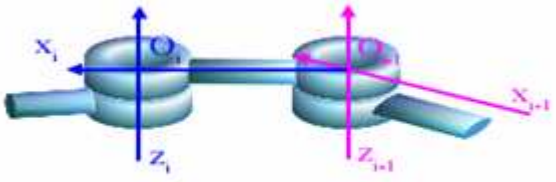


Figure II.12. a

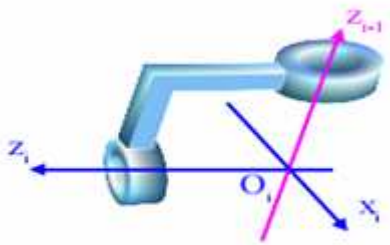


Figure II.12.b

II.13. Modélisation géométrique direct :

Le modèle géométrique direct permet de connaitre l'emplacement de l'effecteur (organe terminal) par rapport à un repère de référence en fonction des variables articulaires.

Autrement dit, le modèle géométrique direct est un système d'équation donnant d'une façon explicite les coordonnées opérationnelles en fonction des coordonnées généralisées.

L'équation 1 montre d'une manière générale cette relation existante entre ces deux types de coordonnées.

$$ X = f(q) \quad \text{.....(1).} $$

Tel que :

X: represente les coordonnées coordonnées opérationnelles

q : represente les coordonnées coordonnées généralisées.

II.14. Modèle géométrique inverse du robot (MGI) :

On appelle modèle géométrique inverse la relation réciproque du modèle géométrique direct, c'est-à-dire il permet la synthèse d'un système d'équations exprimant explicitement les coordonnées généralisées en fonction des coordonnées opérationnelles. L'équation 1 devient alors :

$$\mathbf{q} = \mathbf{f^{-1}(X)}$$

On remarque que ce modèle donne les valeurs des variables articulaires en fonction de la position et l'orientation (la situation) de l'organe terminal.

Remarque :

Pour un même robot, il y a plusieurs modèles géométriques inverses alors qu'il n'y a qu'un seul modèle géométrique direct.

II.15. Modélisation du robot SCARA (RP3) à commander :

II.15.1.Description du robot à commander:

Le robot **SCARA RP3** est un bras manipulateur planaire ayant une structure ouverte simple, constitué principalement de deux segments S1 et S2 qui peuvent tourner respectivement autour des axes Z_1 et Z_2 ; S2 comporte l'organe terminal qui est une pince à deux doigts.

Pour le robot SCARA, les rotations θ_1 et θ_2 ainsi que la translation d de la pince sont assurées par les actionneurs électrique (moteur pas à pas) tandis que, l'ouverture et fermeture de la pince elles sont assurées par les actionneurs pneumatiques fonctionnant en tout où rien.

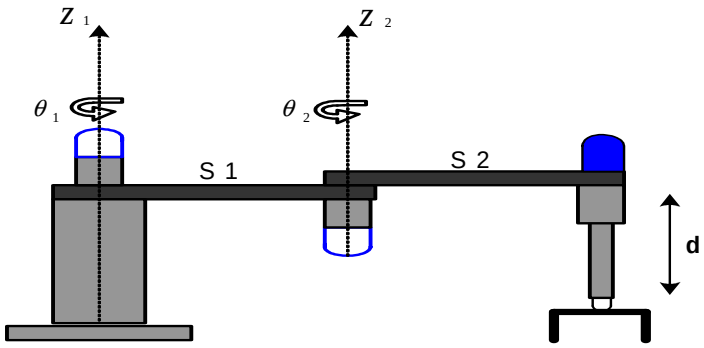


Figure II.13. Robot SCARA RP3.

II.15.2.Espace de travail du robot:

Comme représenté sur la **figure II.14** l'espace de travail du robot correspond a un espace fermé par le cylindre de :

- Rayon R= L2 + L3
- axe Z1
- hauteur h = dmax

Les longueurs des éléments rigides du robot sont :

L_1=20cm.

L_2=15cm.

L_3=10cm.

dmax=20cm

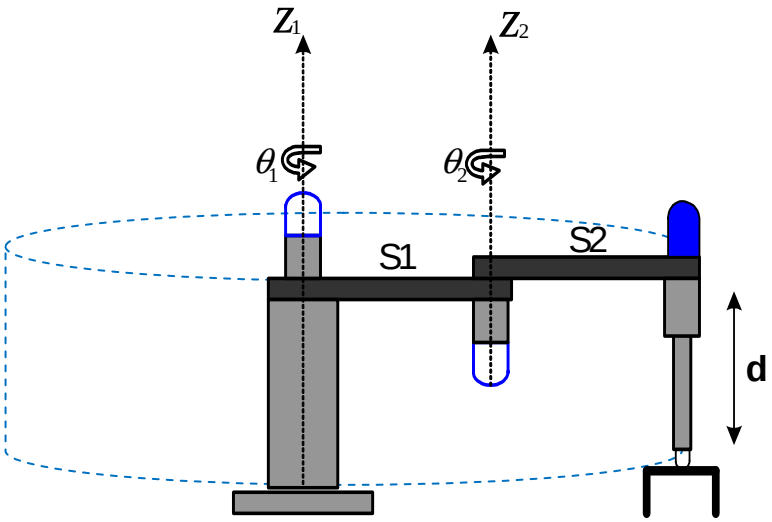


Figure II.14.Espace de travail du robot SCARA

II.15.3.Modèle géométrique du robot SCARA RP3:

La figure ci-dessus donne une description géométrique du robot SCARA RP3. Le placement des repères et la détermination des paramètres géométriques sont effectués selon la méthode décrite précédemment.

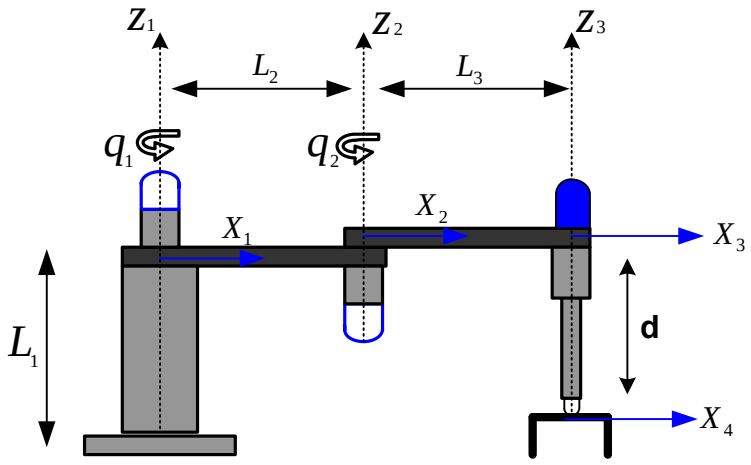


Figure II.15. Modélisation du robot SCARA RP3.

Les paramètres géométriques de DENAVIT-HARTENBERG du robot SCARA sont donnés dans le tableau ci-dessus.

Corps	a_i	d_i	α_i	θ_i
1	0	L_1	0	0
2	L_2	0	0	q_1
3	L_3	0	0	q_2
4	0	d	0	0

Les matrices de transformations homogènes sont données comme suit :

$$ {}^{i-1}T_i = \begin{bmatrix} C\theta_i & -S\theta_i\, C\alpha_i & S\theta_i\, S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i\, C\alpha_i & -C\theta_i\, S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Pour i=1 :

$$^{0}T_{1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pour i=2 :

Pour i=3 :

$$^{1}T_{2} = \begin{bmatrix} \ddot{Sq_1} & \ddot{Cq_1} & 0 & L_2Cq_1 \\ 0 & 0 & 1 & L_2Sq_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}T_{3} = \begin{bmatrix} Cq_2 & -Sq_2 & 0 & L_3Cq_2 \\ Sq_2 & Cq_2 & 0 & L_3Sq_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pour i=4 :

$$^{3}T_{4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matrice de transformation homogène de l'organe terminal par rapport à la base du robot est donnée par :

$^{0}\mathbf{T_4}={}^{0}T_1\ {}^{1}T_2\ {}^{2}T_3\ {}^{3}T_4$, ce qui nous donne :

$$ {}^{0}T_4 = \begin{bmatrix} C(q_1+q_2) & -S(q_1+q_2) & 0 & L_3C(q_1+q_2)+L_2Cq_2 \\ S(q_1+q_2) & C(q_1+q_2) & 0 & L_3S(q_1+q_2)+L_2Sq_1 \\ 0 & 0 & 1 & L_1+d \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Le modèle géométriques direct du robot est déduit directement à partir de la matrice de transformation homogène de l'organe terminal par rapport à la base ; il est donné par :

$$ \begin{cases} X = L_3\ C(q_1+q_2) + L_2\ C(q_2) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{équation (1).} \\ Y = L_3\ S(q_1+q_2) + L_2\ S(q_1) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{équation (2).} \\ Z = L_1 + d \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{équation (3).} \end{cases} $$

II.15.4.Calcule du modèle géométrique inverse du robot SCARA RP3:

A partir des équations du modèle géométrique direct , on a :

$Z=L_1+d$ donc on aura **:** $\mathbf{d= Z - L_1}$

L'équation $(1)^2 + (2)^2$ nous donne:

$X^2+Y^2= (L_3)^2 + (L_2)^2 + 2\ L_3\ L_2\ [C(q_1+q_2)\ C\ q_1 + S(q_1+q_2)\ S\ q_1]$

$X^2+Y^2= (L_3)^2 + (L_2)^2 + 2\ L_3\ L_2\ C\ q_2$ donc on aura :

$$ C\,q_2 = \frac{X^2 + Y^2 - L_3^2 - L_2^2}{2\ L_3\ L_2} $$

Ce qui implique que :

$$q_2 = \text{Arc cos}\left(\frac{X^2 + Y^2 - L_3^2 - L_2^2}{2\,L_3\,L_2}\right)$$

La multiplication de l'équation (1) par C (q1) et l'équation (2) par S (q1) nous donne :

$X\,C(q_1) = L_3\,C(q_1+q_2)\,C\,q_1 + L_2\,C^2(q_1)$ …… (a)

$Y\,S(q_1) = L_3\,S(q_1+q_2)\,S\,q_1 + L_2\,S^2(q_1)$ ……(b)

a + b nous donne :

$X\,C(q_1) + Y\,S(q_1) = L_3\;C\,q_2 + L_2$. En remplaçant q2 par son expression, on aura :

$$X\,C(q_1) + Y\,S(q_1) = L_3\,\frac{X^2 + Y^2 - L_3^2 + L_2^2}{2\,L_3\,L_2} + L_2$$

$$X\,C(q_1) + Y\,S(q_1) = \frac{X^2 + Y^2 - L_3^2 + L_2^2}{2\,L_2}$$

On divisant tous par $\sqrt{X^2 + Y^2}$, on aura :

$$\frac{X}{\sqrt{X^2+Y^2}}\,C(q_1) + \frac{Y}{\sqrt{X^2+Y^2}}\,S(q_1) = \frac{X^2 + Y^2 - L_3^2 + L_2^2}{2\,L_2\,\sqrt{X^2+Y^2}}$$

$$C(\varepsilon)C(q_1) + S(\varepsilon)S(q_1) = \frac{X^2 + Y^2 - L_3^2 + L_2^2}{2\,L_2\,\sqrt{X^2 + Y^2}} \qquad avec\ \varepsilon = Arctg\frac{Y}{X}$$

$$C(q_1 - \varepsilon) = \frac{X^2 + Y^2 - L_3^2 + L_2^2}{2\,L_2\,\sqrt{X^2 + Y^2}}$$

Ce qui nous donne :

$$q_1 = \mathbf{Arc\cos}\left(\frac{X^2 + Y^2 - L_3^2 + L_2^2}{2\,L_2\,\sqrt{X^2 + Y^2}}\right) + \varepsilon$$

Les équations du modèle géométrique inverse du robot SCARA RP3 sont données alors par :

$$q_1 = \mathbf{Arc\cos}\left(\frac{X^2 + Y^2 - L_3^2 + L_2^2}{2\,L_2\,\sqrt{X^2 + Y^2}}\right) + \varepsilon$$

$$q_2 = \mathbf{Arc\cos}\left(\frac{X^2 + Y^2 - L_3^2 - L_2^2}{2\,L_3\,L_2}\right)$$

$d = Z - L_1$

Ces trois équations seront programmées dans le PIC afin de permettre au robot de réaliser les taches désirées.

II.16. Conclusion :

Au cours, de ce chapitre on a exposé les concepts de base utilisés en robotique. Ce chapitre comporte aussi les outils mathématiques nécessaires pour la modélisation géométrique qui a été retenue pour modéliser notre robot manipulateur.

Après la détermination des équations mathématiques définissant le modèle géométrique inverse du robot, ces équations seront programmées dans le PIC afin de permettre au robot de réaliser les taches désirées.

Chapitre III :

Conception logicielle et matérielle

III.1.Introduction :

Dans cette partie de notre travail, nous allons réaliser le circuit qui va commander les articulations du robot SCARA.

Pour ce faire, nous avons établit un schéma synoptique **(Figure III.1)** d'un montage à base d'un microcontrôleur PIC 16F877A, à qui nous avons ajouté une interface utilisateur, et des circuits de puissance (L293D).

L'interface utilisateur est constituée de 4 boutons poussoirs pour l'introduction des coordonnées de l'objet que doit déplacer le robot, et d'un afficheur LCD 2 lignes, 16 caractères qui permet de visualiser en instantanée les informations relatives à ses paramètres.

Le circuit de puissance L293D permet l'amplification du courant de sortie du PIC (25mA) pour commander le moteurs pas à pas qui nécessite un courant plus élève.

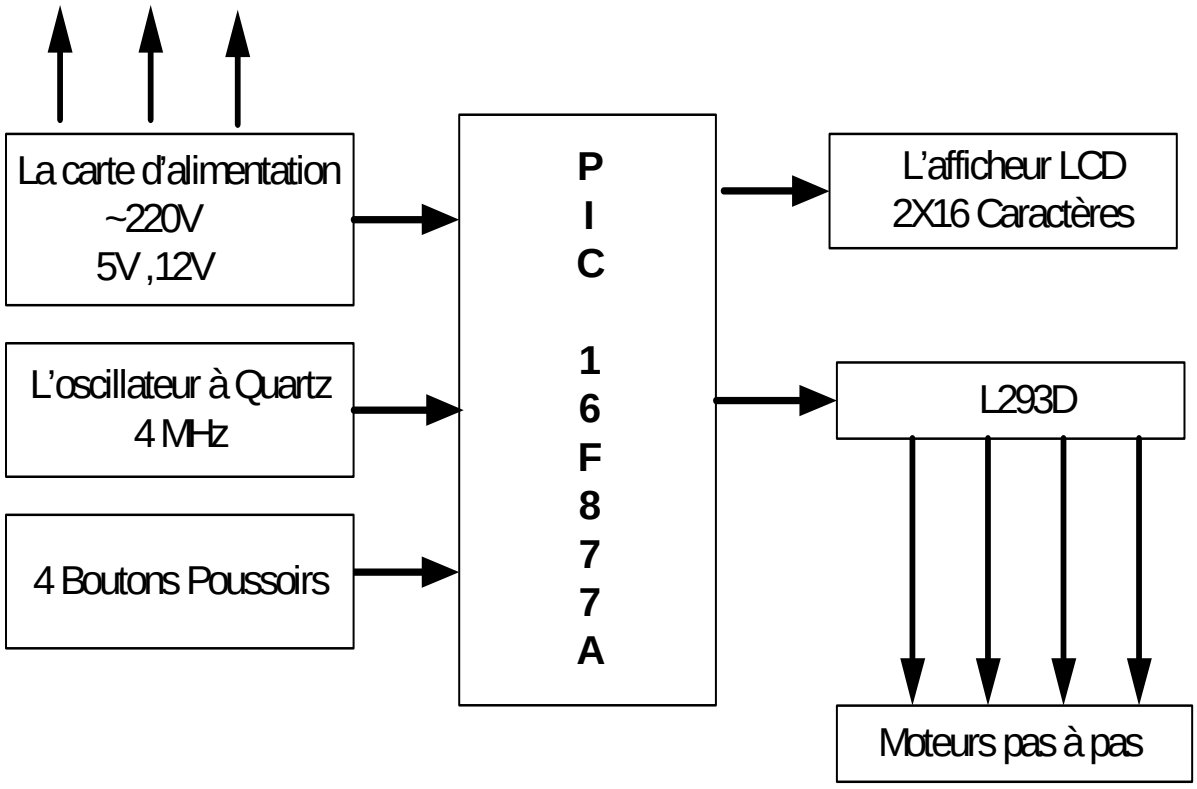


Figure III.1. Schéma synoptique du montage.

III.2.Microcontrôleur:

III.2.1.Introduction :

Notre montage est architecturé autour du microcontrôleur PIC 16F877A (comme on peut le voir sur le synoptique de fonctionnement (**Figure III.1**). Ce dernier nous a permis de générer les séquences de commende des moteurs pas à pas afin d'assurer l'orientation du bras du robot (organe terminal) vers la position exacte de l'objet qu'on veut déplacer.

L'intégration du pic 16F877 dans notre carte de commande nous a permis d'utiliser que peu de composants. En effet, en ayant recours à la logique programmée, le fonctionnement du robot manipulateur repose essentiellement sur le programme embarqué sur le PIC (Le Software).Ce qui confère au robot l'avantage d'être évolutif par soft contrairement à la logique câblée.

III.2.2.Présentation du PIC 16F877 :

Le pic 16F877 est un composant de Microchip, il fait partie de la famille mid-range, sa mémoire programme est de type Flash, il est capable de fonctionner à une fréquence d'horloge de 20Mhz.

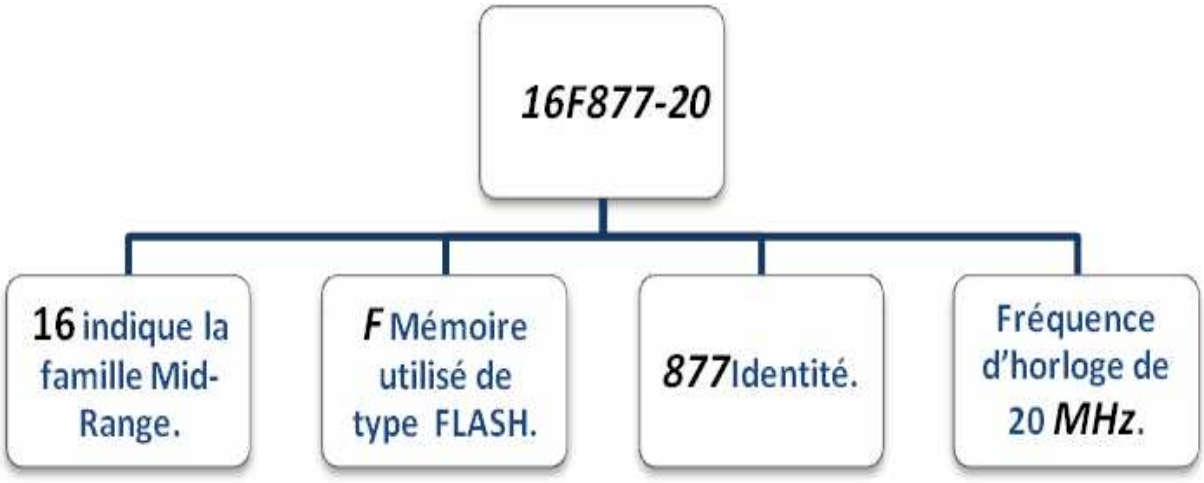


Figure III.2 : configuration du pic 16F877

Le pic 16F877 présente toute une série de composants, ces principales caractéristique sont :

- Mémoire programme de type flash de 8Kmots,
- RAM de 368 octets,
- EPROM de 256 octets,
- 33 E /S reparties en 5 ports bidirectionnels.
- 8 convertisseurs A /N de 10 bits.
- 14 sources d'interruption.
- Chien de garde WDT.
- Chaque sortie présente un courant max de 25 mA.
- Une interface de communication série synchrone (SSP/SPI).
- Le compteur de programme est sur 13 bits.
- Jeu de 35 instructions.
- Toutes les instructions n'ont besoin que d'un cycle d'horloge sauf pour les sauts qui nécessitent deux cycles d'horloge,
- Une interface de communication série asynchrone et synchrone (USART/SCI),
- Une tension d'alimentation entre 4,5v et 6v,
- Trois temporisations (TIMER0, TIMER1, TIMER2).

III.2.3. Organisation externe du PIC 16F877 :

Les entrées sorties du microcontrôleur peuvent être groupées par fonction comme le montre la **figure III.3.**

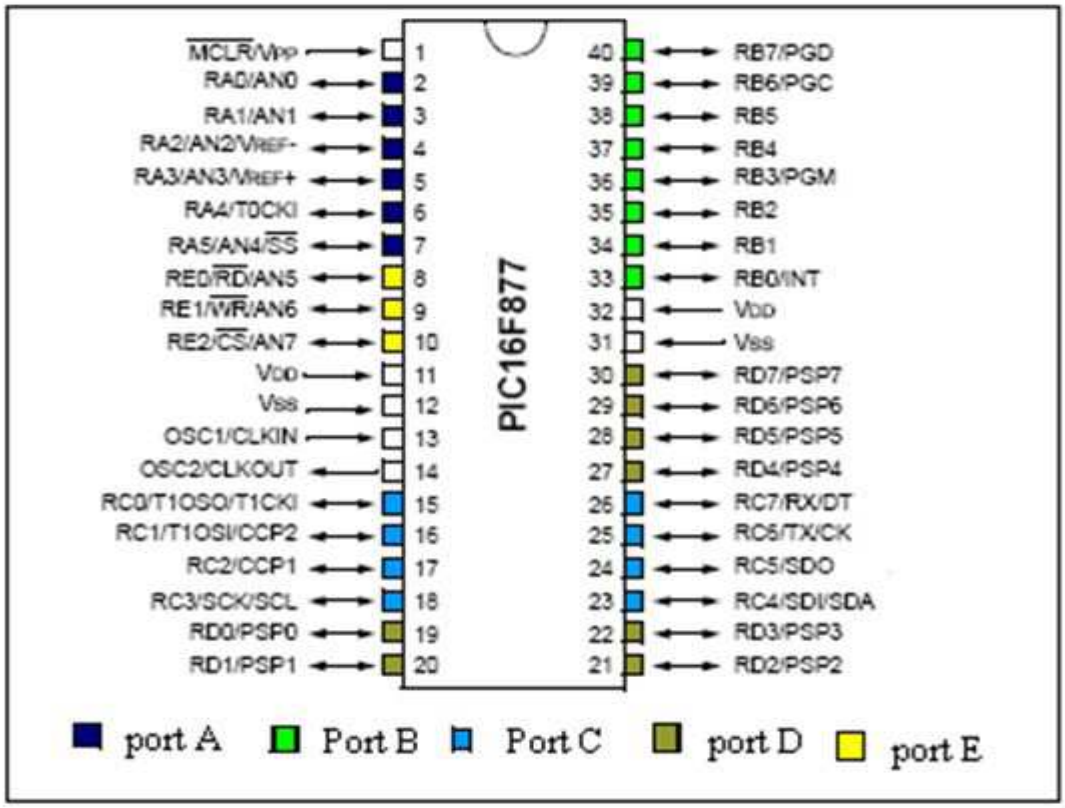


Figure III.3: Brochage du PIC 16F877

Le boitier du PIC 16F877 décrit par la figure ci-dessus comprend 40 pins :

33 pins d'entrées/sorties, 4 pins pour l'alimentation, 2 pins pour l'oscillateur et une pin pour le Reset (MCLR).

- **Les ports :**

Le Pic 16F877 dispose de 33PINS d'E/S pour communiqué avec l'environnement extérieure. Celles-ci sont reparties en 5 ports parallèles bidirectionnels :

- ✓ 6 lignes pour le port A.
- ✓ 8 lignes pour le port B.
- ✓ 8lignes pour le port C.
- ✓ 8 lignes pour le port D.
- ✓ 3 lignes pour le port E.

III.2.4. Configuration minimale du PIC:

Pour réaliser une carte à base d'un pic 16F877, il faut au moins configurer son alimentation, le MCLR et l'horloge (voir **figure III.4).**

1. L'alimentation :

Le PIC fonctionne dans la plage de tension qui s'étend de 4,5 à 6V

2. Le circuit d'initialisation :

Le MCLR est habituellement relié au 5V. Cette pin étant utilisée pour effectuer un reset du composant en cas de connexion à la masse.

3. L'Oscillateur :

L'oscillateur est le cœur du microcontrôleur : c'est lui qui cadence le déroulement du programme. Il doit fournir un signal carré périodique au microcontrôleur.

Pour notre PIC 16F877A nous avons optés pour l'une des méthodes fournie avec le Datasheet du composant qu'est l'oscillateur à Quartz.

Pour le 16F877A, nous allons considérer plusieurs types d'oscillateurs :

- LP : les oscillateurs basse fréquence, en dessous de 200 KHz.
- XT : les oscillateurs moyenne fréquence, entre 200 KHz et 4MHz.
- HS : les oscillateurs haute fréquence, entre 4MHz et 20 MHz.

Dans le cas de notre application nous utiliserons l'oscillateur de type HS avec un quartz de 4 MHz

Sur la **figure III.4,** l'oscillateur à Quartz est câblé au PIC sur les broches OSC1/CLKIN et OSC2/CLKOUT comme il a été recommandé par Microchip. Pour le choix des capacités le constructeur préconise également, dans le Datasheet du composant, l'utilisation de ces valeurs :

Type osc	Fréq quartz	Capacite1	Capacite2
HC	4Mhz	15Pf	15Pf
	8Mhz	15-33Pf	15-33Pf
	20Mhz	15-33Pf	15-33Pf

D'où notre oscillateur comporte :

- Un quartz 4 Mhz.
- Deux capacités de15PF.

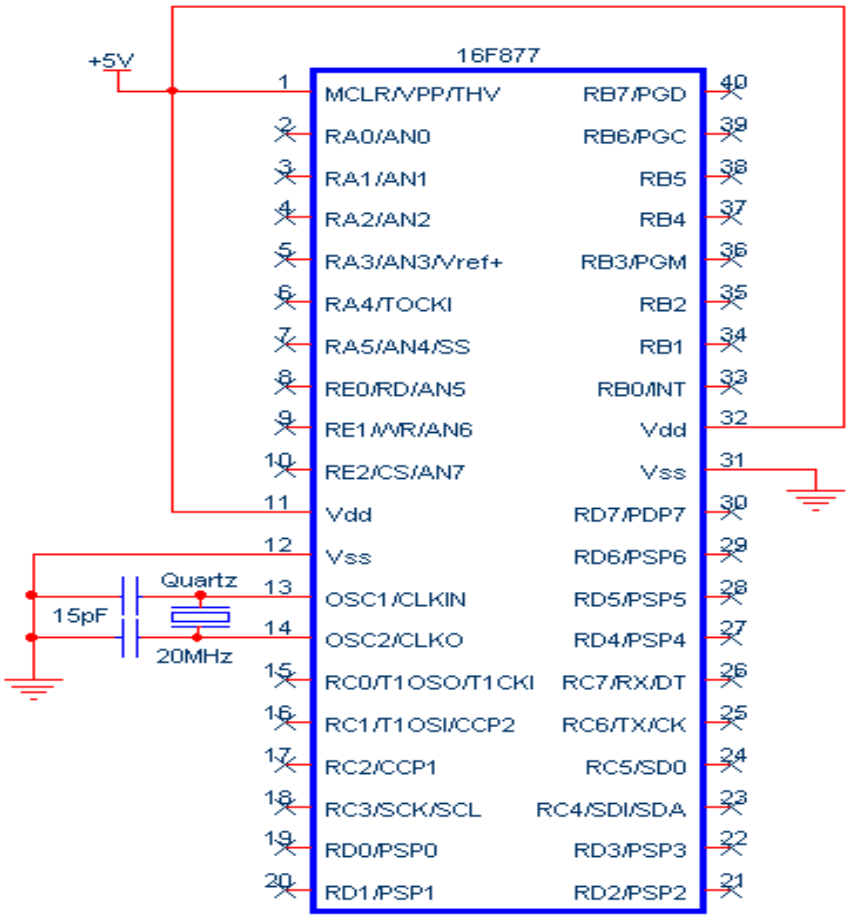


Figure III.4 : configuration minimale du PIC 16F877

III.3. L'afficheur LCD :

Le module LCD 2 lignes de 16 caractères est un GDM1602A de chez XIAMEN OCULAR qui intègre son propre module de gestion, il est donc directement connecté aux broches du PIC et alimenté en 5V.

C'est un afficheur LCD à accès parallèle, c'est-à-dire qu'il reçoit les données à afficher sous forme parallèle avec possibilité (programmable) de transmettre en mode 8 bits ou en mode 4 bits, dans ce dernier cas on économise quatre lignes du PIC mais le transfert se fera en deux fois . Ainsi pour notre application le GDM1602A communiquera avec le PIC via le PORTB en mode 4 bits (**Figure III.5**).

III.3.1.Le brochage de l'afficheur LCD :

VSS	0 volts de l'alimentation
VDD	+5V
VEE	Tension à appliquer pour gérer le contraste
RS	Registre Selecte (mode instruction ou donnée)
R/W	Read/Write (écriture ou lecture)
E	Enable (sélection de l'afficheur)
DB-DB7	Data 0 à 7 (données où instructions)

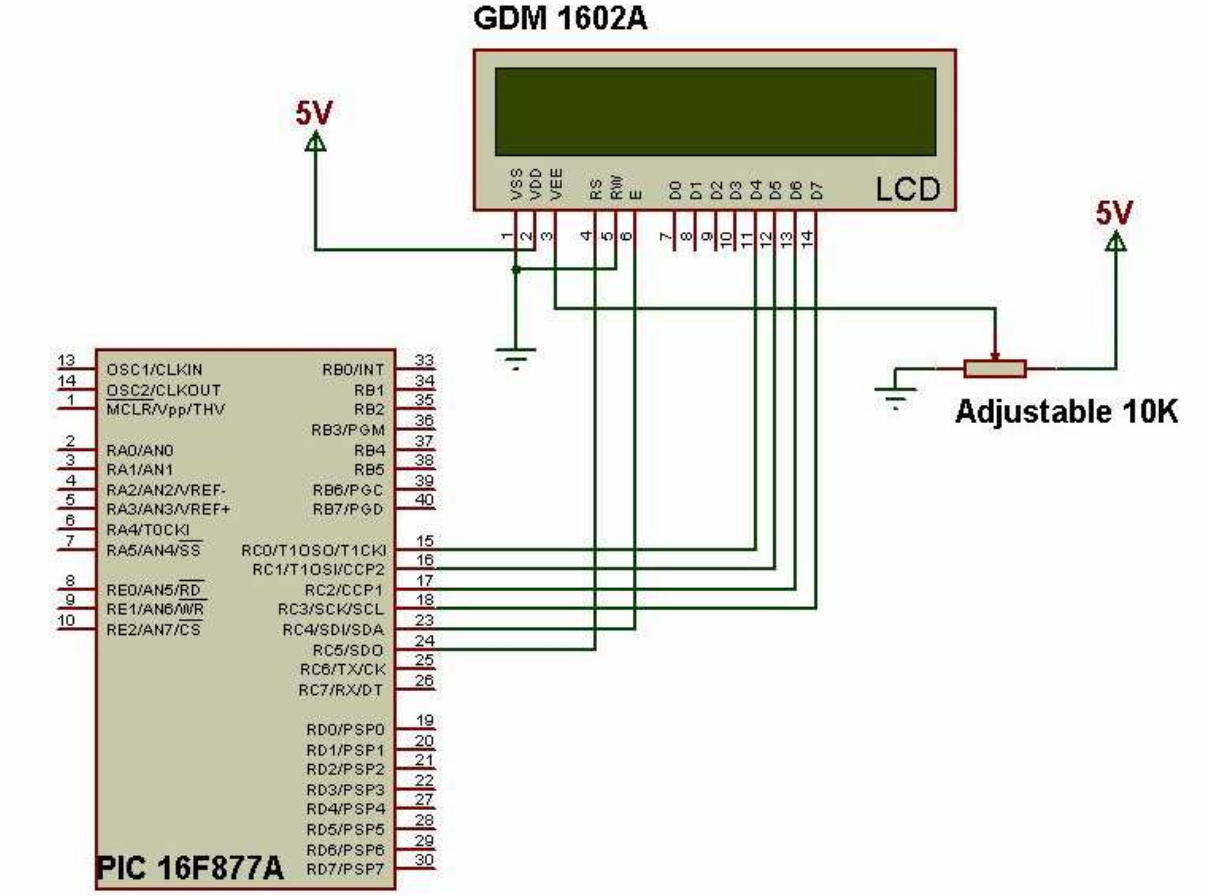


Figure III.5. Câblage de l'Afficheur LCD

III.3.2. Fonctionnement de l'afficheur LCD :

L'afficheur dans notre montage n'a été câblé qu'en écriture (RW=0), nous nous limiterons donc à ce type de fonctionnement. L'afficheur proprement dit (effacement de l'écran, affichage du curseur etc.) soit des données qui seront affichées à la position courante du curseur.

La sélection du mode instruction est réalisée en mettant la broche RS à 0 (Register Select) et en envoyant ensuite l'octet de commande sur le port de données (DB4 à DB7). Lorsque la broche RS est à 1 les valeurs envoyées sur les lignes DB4 à DB7 seront affichées sous forme de caractère ASCII à la position courante du curseur.

La valeur du potentiomètre servant au réglage du contraste de l'afficheur recommande de prendre un potentiomètre de valeurs comprises entre 10K et 20k.

III.4. La commande des moteurs pas à pas :

Afin de simplifier la commande des moteurs pas à pas et utilisé le minimum de circuit intégré dans notre carte de commande, on a programmé dans le PIC 16F877A les séquences nécessaires pour générer la rotation du moteur soit en mode pas entier ou bien en mode demi pas ensuite pour pouvoir faire tourner le moteur pas à pas qui nécessite une certaine puissance ,on a utilisé un circuit de puissance qui est : le circuit L293D qu'on a choisit parmi plusieurs autres circuits de puissance.

III.4.1. Les moteurs utilisés dans le montage :

Après avoir testé les moteurs pas à pas à l'aide d'un multimètre, on a déduit qu'ils s'agissent des moteurs de type unipolaire qui nécessitent un courant nominal de 360mA. En fonction de ceux deux critère, on a choisie le circuit adéquat pour réaliser la commande de ces derniers ; ce circuit n'est autre que le L293D qui est très utilisé.

III.5. Présentation du L293D:

Le circuit intégré L293D est un circuit de puissance qui sert à amplifier les signaux pour commander les moteurs pas à pas.

Il est composé de 4 transistors Darlington (en emeteur commun) pour amplifier le courant de sortie nécessaire pour le bon fonctionnement du moteur.

Chaque pont de transistor est connecter à une entré de validation pour valider où pas le fonctionnement des deux transistors Darlington.

Le circuit L293D est composé de 16 broches comme indiqué dans la figure ci-dessous.

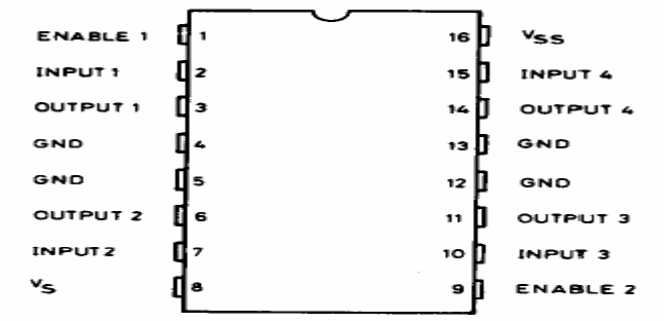


Figure III.6. Brochage du L293D.

III.5.1. Brochage du L293D avec le PIC:

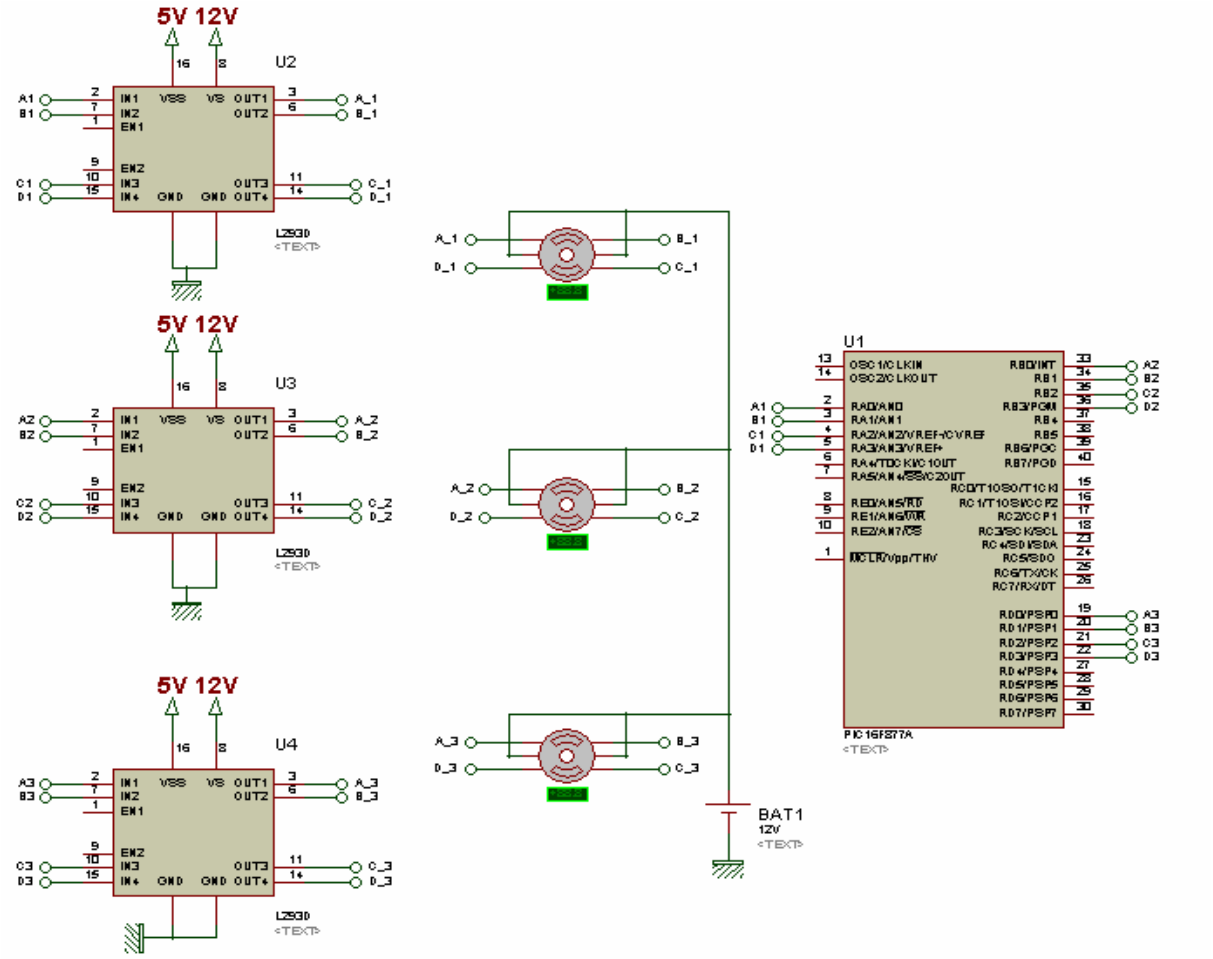


Figure III.7. Brochage du L293D avec le PIC 16F877

III.5.2. Caractéristiques principales du L293D:

- ✓ Courant de sortie de 600 mA.
- ✓ Tension d'alimentation 5V et 12V.
- ✓ Prix dérisoire.

III.6. Brochage des boutons poussoirs avec le PIC :

Notre montage comporte quatre boutons poussoirs qui ont des fonctions différentes **(**voir **figure III.8).**

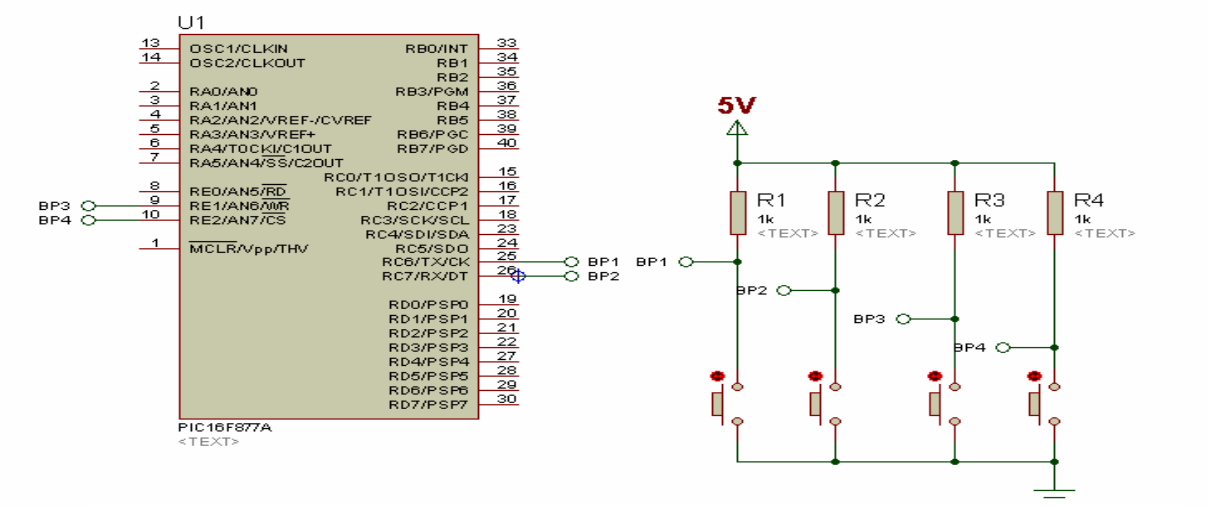


Figure III.8.Brochage des boutons poussoirs avec le PIC 16F877

Le bouton poussoir BP1 permet d'incrémenter la variable sélectionné.

Le bouton poussoir BP2 permet de décrémenter la variable sélectionner.

Le bouton poussoir BP3 permet de sélectionner une variable (x,y,z).

Le bouton poussoir BP4 permet de lancer les calculs des angles pour déterminer l'angle de rotation de chaque moteur.

III.7.Les Organigrammes :

Les organigrammes ont pour but de faciliter la compréhension et le déroulement du programme, et ne représentent que les actions principales effectués par le microcontrôleur. Ils ne traduisent pas intégralement le code en C et ne reflètent pas les particularités du PIC.

Le programme peut se diviser en plusieurs fonctions :

- La fonction *Main* : Le Programme principal.
- Les fonctions de la gestion de l'afficheur LCD (initialisation, effacement de l'écran, commande…).
- Les fonctions du menu pour le pilotage du microcontrôleur via les 4 boutons poussoirs.

III.7.1.La Fonction Main (le programme principal) :

La figure III.9 represente l'organigramme du programme principal.

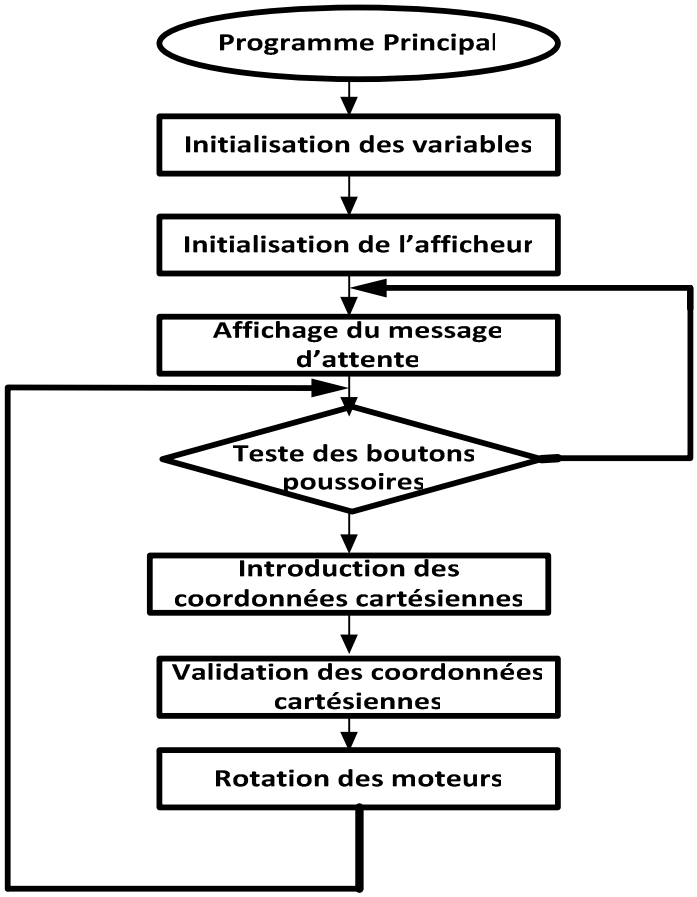


Figure III.9. Organigramme : Programme Principale

III.7.2. La Gestion de l'Afficheur LCD :

L'afficheur LCD nécessite une phase d'initialisation avant de pouvoir remplir sa fonction. L'afficheur est initialisé lors de l'appel de la fonction *init_lcd()* dans le programme principal. La procédure d'initialisation est la suivante :

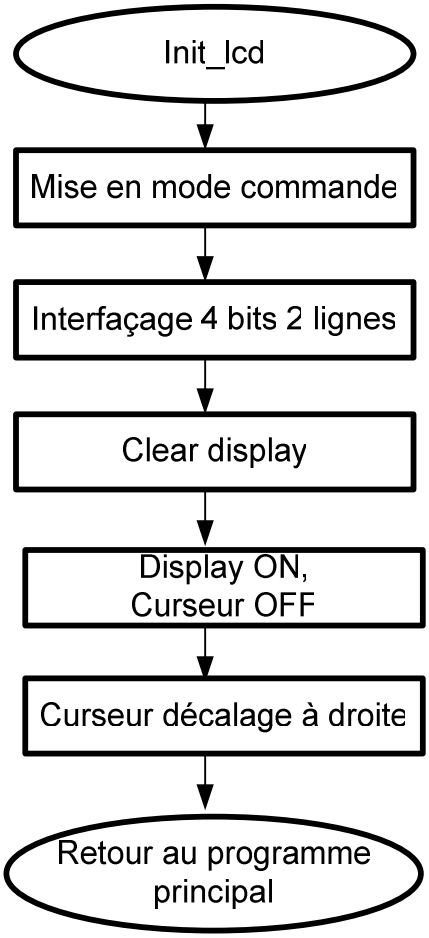


Figure III.10.Organigramme de la fonction init_lcd ()

Une fois initialisé, l'afficheur est près à recevoir les informations à afficher. Pour simplifier l'écriture, nous avons fait quatre fonctions : **lcd_gotoxy()**, **printf(LCD_PUTC," "), lcd_putc('\f')**;pour transférer les caractères à afficher et pour spécifier l'adresse (la position sur l'écran) où l'on désire écrire.

III.7.3.La Gestion des boutons poussoirs :

Le clavier (les boutons poussoirs) est géré dans la boucle principale (Loop (;;)). A chaque redondance de la boucle, les ports d'entrés (RC6, RC7, RE1, RE2) sont testés.

Ces quatre boutons poussoirs nous permettent de sélectionner et d'introduire les coordonnées cartésiennes pour faire tourner les moteurs à la position désirée.

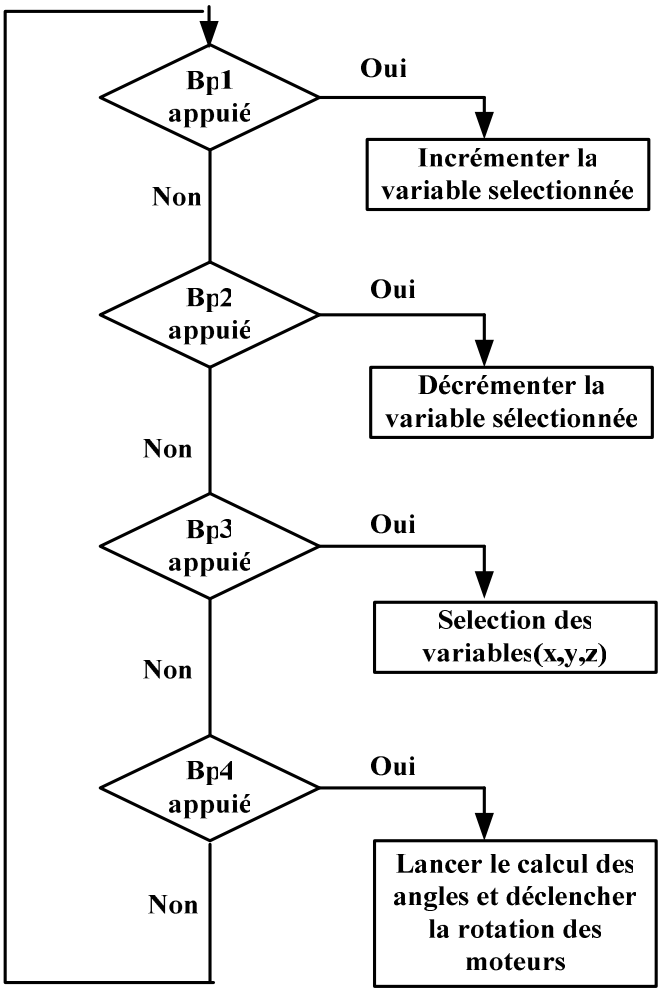


Figure III.11.Organigramme de la gestion des boutons poussoirs

Chapitre IV :

Réalisation pratique

IV.1. La réalisation du circuit imprimé :

Le circuit imprimé (CI) est le support de tout montage électronique : il est le lien aussi bien mécanique qu'électrique entre les différents composants. Le circuit imprimé est une platine de matériaux composite recouverte d'une couche très fine de cuivre métallique.

Le cuivre assure les liens électriques entre les composants, pour que ceux-ci soient conformes au plan du montage.

La réalisation du circuit imprimé peut se décomposer en les étapes suivantes :

IV .1.1. La réalisation du typon :

Le dessin du typon est réalisé à l'aide du logiciel **PROTEUS** .L'intérêt de ce logiciel est qu'il vérifie la cohérence entre le typon réaliser et le schéma. Il possède également des fonctions « d'autoroutage » qui permettent de laisser le logiciel faire les pistes entre les composants. Cette méthode est rapide mais donne souvent des résultats moyens qu'il faut corriger à la main.

1. Enregistrement du typon en format Bitmap où PCB pour transfert sur imprimante Laser.
2. Transfert du dessin sur papier transparent.
3. Vérification et éventuelle retouche manuelle : Le résultat obtenu n'est pas toujours celui escompté au départ. En effet, le transfert sur papier transparent via l'imprimante donne parfois une opacité imparfaite des pistes .Il faut alors contrôler les pistes et opacifier les trous à l'aide d'un feutre pour papier transparent.

IV .1.2. La découpe :

Au départ, une plaque vierge est une plaque composée de trois couches :

- ✓ Une couche d'époxy ou de Bakélite isolant.
- ✓ Une couche de cuivre conducteur.
- ✓ Une couche de résine pré sensibilisées.

- ✓ Un film protecteur opaque autoadhésif recouvre la résine pré sensibilisée.

Cette étape consiste à découper l'époxy à la taille du typon à l'aide d'une scie.

IV .1.3. Insolation du circuit :

L'insolation consiste à envoyer des UV à l'aide d'une insoleuse sur certaines parties de la plaque d'Epoxy. Les parties isolées disparaitront lors de la révélation et il ne restera plus que les parties qui étaient cachées par un masque**.**

Pour insoler une platine nous allons suivre ces étapes :

1. Eviter de travailler en plein soleil.

2. Décoller le film protecteur.

3. Placer le typon sur le verre de l'insoleuse (attention à l'orientation) et le coté cuivré sur le typon .

4. Allumer l'insoleuse pendant environ 2 minutes et recouvrir l'insoleuse de façon à éviter les fuites des UV. Cette étape est très importante, une surexposition aux UV de la plaque provoquera des trous dans les pistes, et une sous exposition provoquera des faux contacte entre les pistes .Une série de teste s'impose donc pour arriver à trouver le juste milieu ente « sur » et « sous » exposition du circuit aux UV.

IV .1.4. La révélation :

Après avoir insolé notre plaque, nous allons maintenant la révéler. Pour ce faire, il faut placer la plaque dans un bac de révélateur qui est en fait une soude caustique (produit hautement corrosif), à la température inscrite sur son paquet, et agiter jusqu'à ce que le dessin apparaisse nettement sur un fond cuivré, puis rincer la plaque avec de l'eau.

IV .1.5. La gravure :

La gravure se fait avec du perchlorure de fer. Elle sera d'autant plus rapide que celui-ci est chaud et agité. Nous pouvons placer la platine dans un bac de perchlorure et attendre sans y toucher quelques heures, mais il est nettement mieux et plus rapide d'agiter au moins de temps en temps. A la fin, bien rincer le circuit.

IV.1.6. Le perçage :

A l'aide d'une micro perceuse munie d'un foret de diamètre de 0,8 mm en général, on perce soigneusement le support et chaque pastille du circuit imprimé pour y implanter les pattes des composants.

IV .2. Développement de notre maquette :

Pour développer le typon de notre montage nous avons opté d'utiliser le logiciel de simulation PROTEUS, qui intègre un environnement de développement des circuits imprimés ARES .En effet, après avoir crée et simulé notre réalisation sous ce même logiciel nous avons réussi aisément à passer directement du schéma électrique au PCB en se servant principalement de la fonction d'autoroutage.

IV .2.1. Le circuit imprimé de la carte principale :

Les deux faces du circuit imprimé de la carte principale, coté cuivre et coté composants sont illustrés respectivement par la **figure IV.1 et la figure IV.2**.

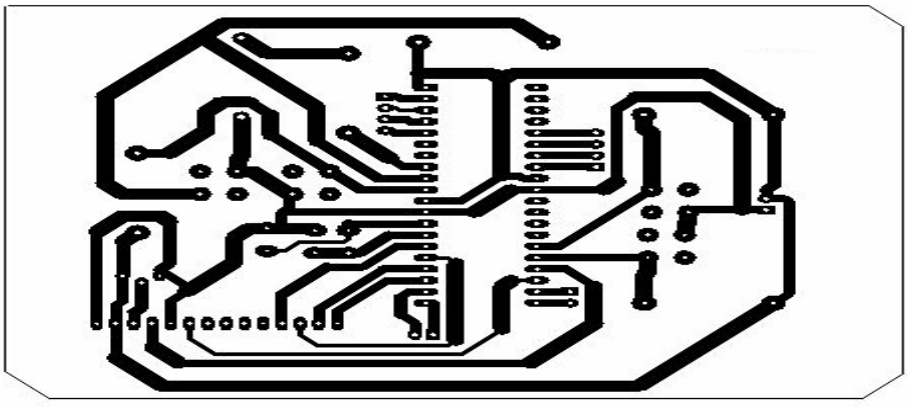


Figure IV.1.Coté cuivre du circuit imprimé (carte principale)

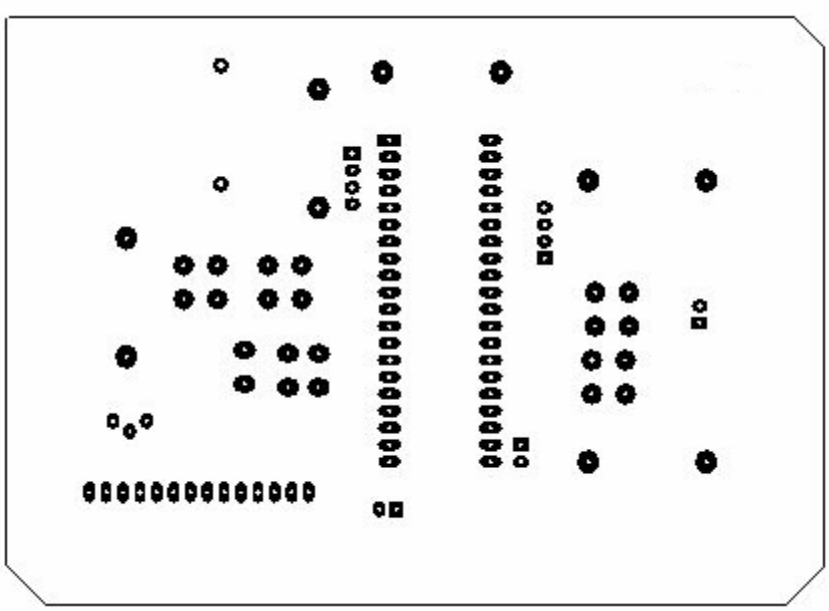


Figure IV.2.Coté composants du circuit imprimé (carte principale)

IV .2.2.Implantation des composants :

La figure V.3 indique l'implantation de chaque composant sur la surface du circuit imprimé.

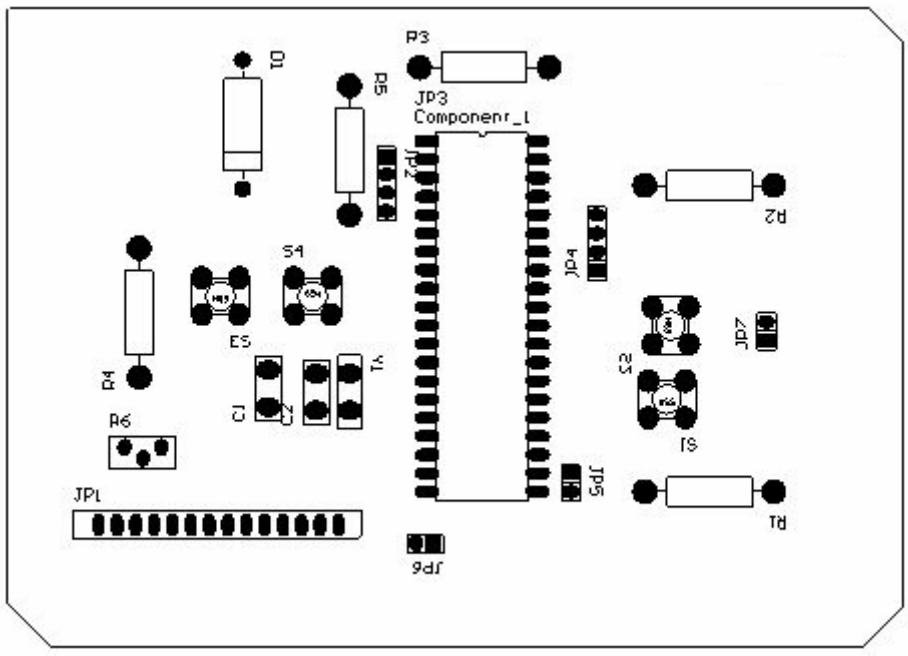


Figure IV.3.Implantation des composants (carte principale)

IV .2.3. Vue en 3 Dimensions du circuit imprimé :

La figure IV.4 représente le circuit imprimé en 3 dimensions.

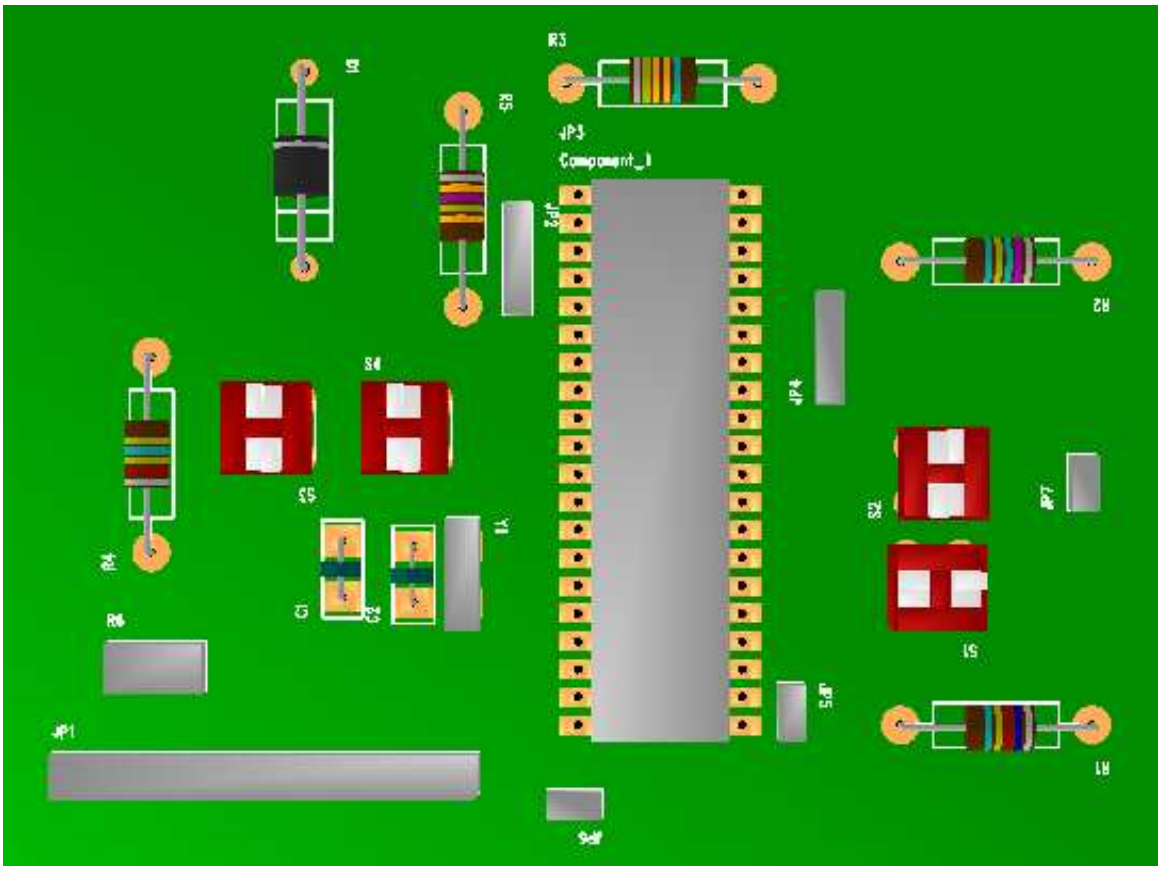


Figure IV.4 .Vue du circuit imprimé en 3Dimension.

IV.3. Nomenclature des composants :

Les tableaux ci-dessus affichent pour chaque type tous les composants utilisés dans notre application.

Résistance		
Quantités	Référence	Valeur
4	**R1, R2, R3, R4**	**1K**

Capacités		
Quantités	Référence	Valeur
2	**C1, C2**	**15PF**

Circuits intégrés		
Quantités	Référence	Valeur
1	**U1**	**PIC 16F877A**
3	**U2, U3, U4**	**L293D**

Autres composants		
Quantités	Référence	Valeur
4	**Boutons Poussoirs**	
1	**RV1 (Potentiomètre)**	**10K**
1	**Quartz**	**4MHZ**
1	**LCD**	**GDM 1602A**
1	**LED**	

IV.4.Les photos du projet réalisé :

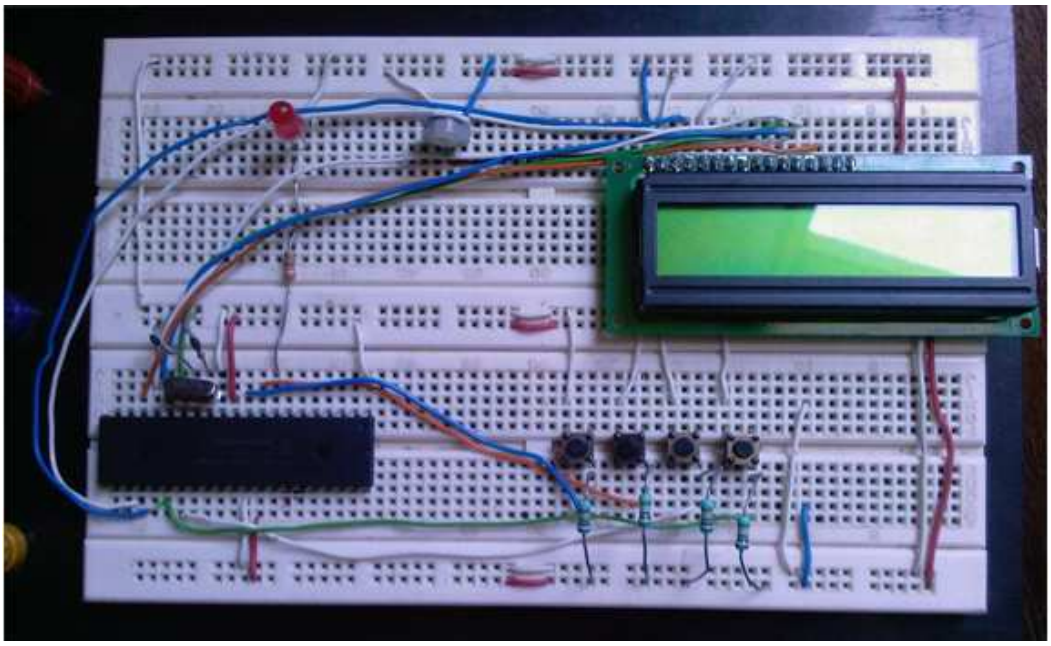


Image1. Partie commande du circuit

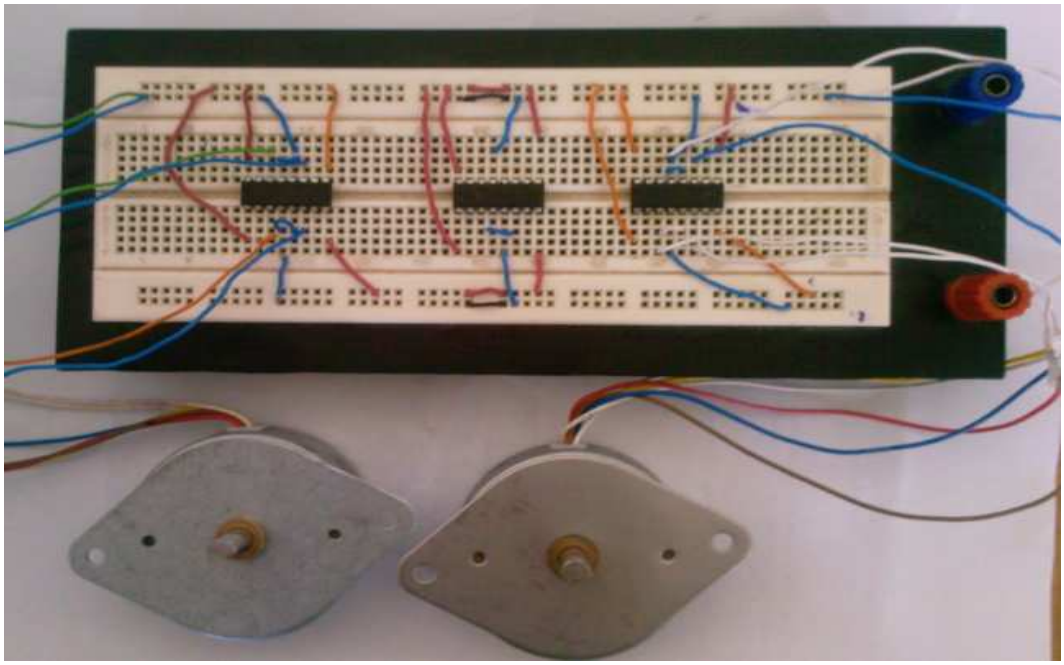


Image 2 . Partie puissance du montage

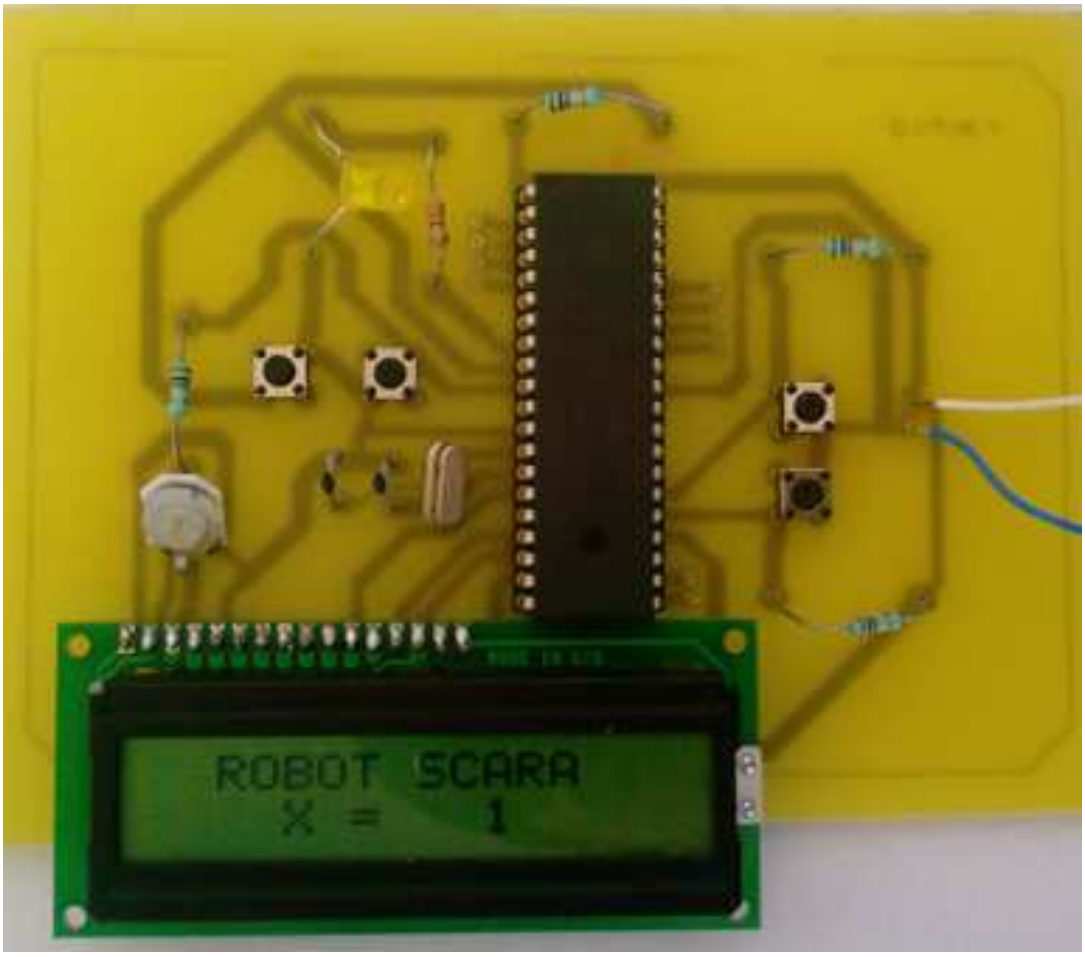


Image 3. La carte commande du montage

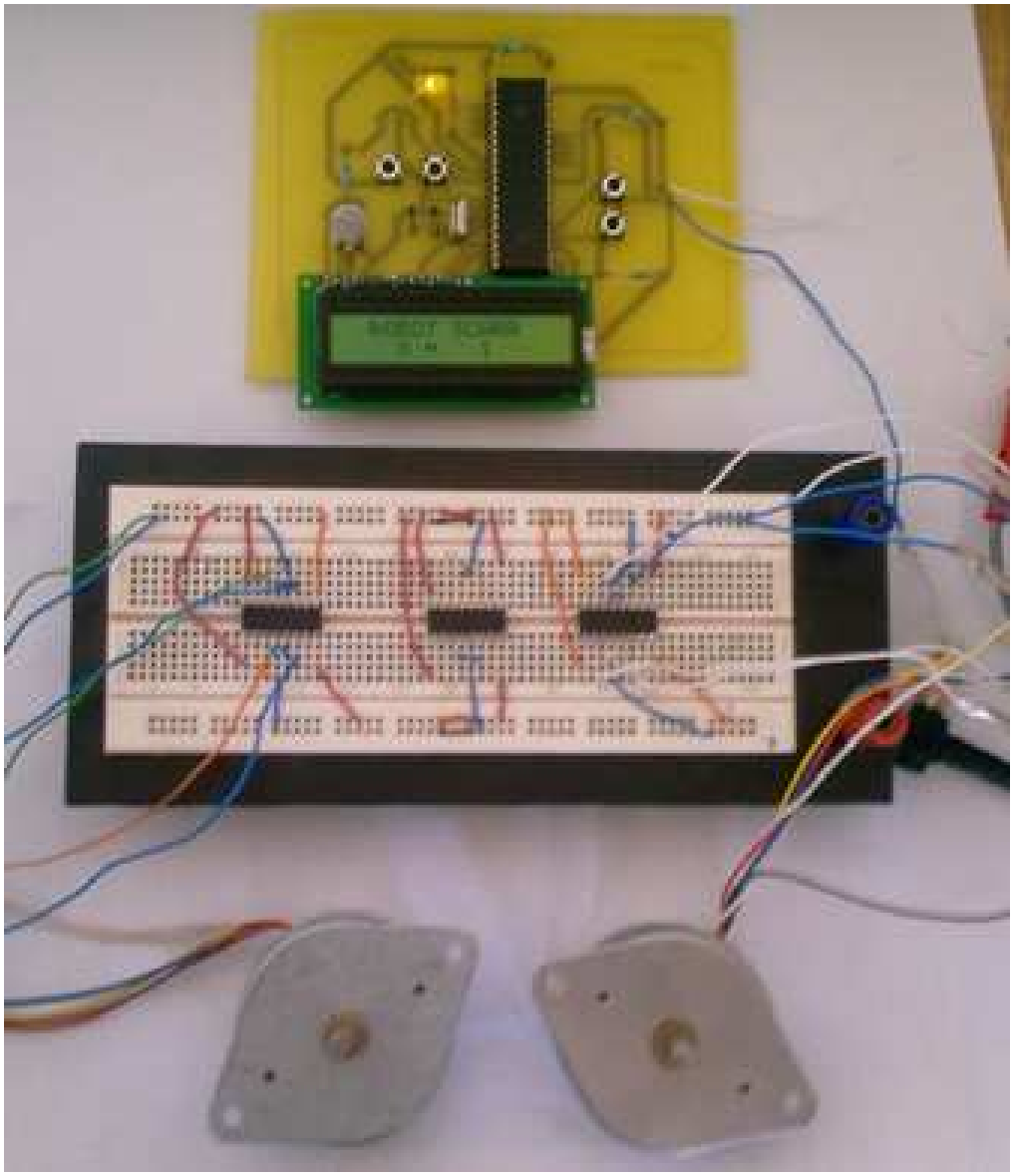


Image 4 .La carte complète du projet réalisé

Conclusion Générale

Conclusion Générale

Au terme de ce travail, nous avons pu réaliser une carte de commande d'un rebot manipulateur architecturé autour du microcontrôleur PIC16F877A.

Ce travail nous a permis de mettre en valeur une partie importante des connaissances théoriques que nous avons accumulées pendant toute la durée de nos études. Il nous a permis également d'acquérir une expérience nouvelle dans l'électronique pratique et la programmation des PIC.

Au cours de la réalisation de notre projet, nous avons utilisé des logiciels de conception, simulation et programmation des circuits électroniques. En effet, nous avons utilisé MPLAB et le PIC C pour la simulation du programme, IC-PROG pour la programmation du microcontrôleur, PROTEUS pour le dessin du circuit électrique, le circuit imprimé et l'implantation des composants et pour la simulation de manière interactive notre projet.

Nous avons aussi acquis quelques notions de base sur la robotique et la modélisation des robots manipulateurs.

Enfin, nous souhaitons dans l'avenir rajouté à notre maquette une commande en temps réel en utilisant avec un transfert vers un calculateur maitre puissant.

Annexes

Annexe I :

Afficheur LCD

XIAMEN OCULAR

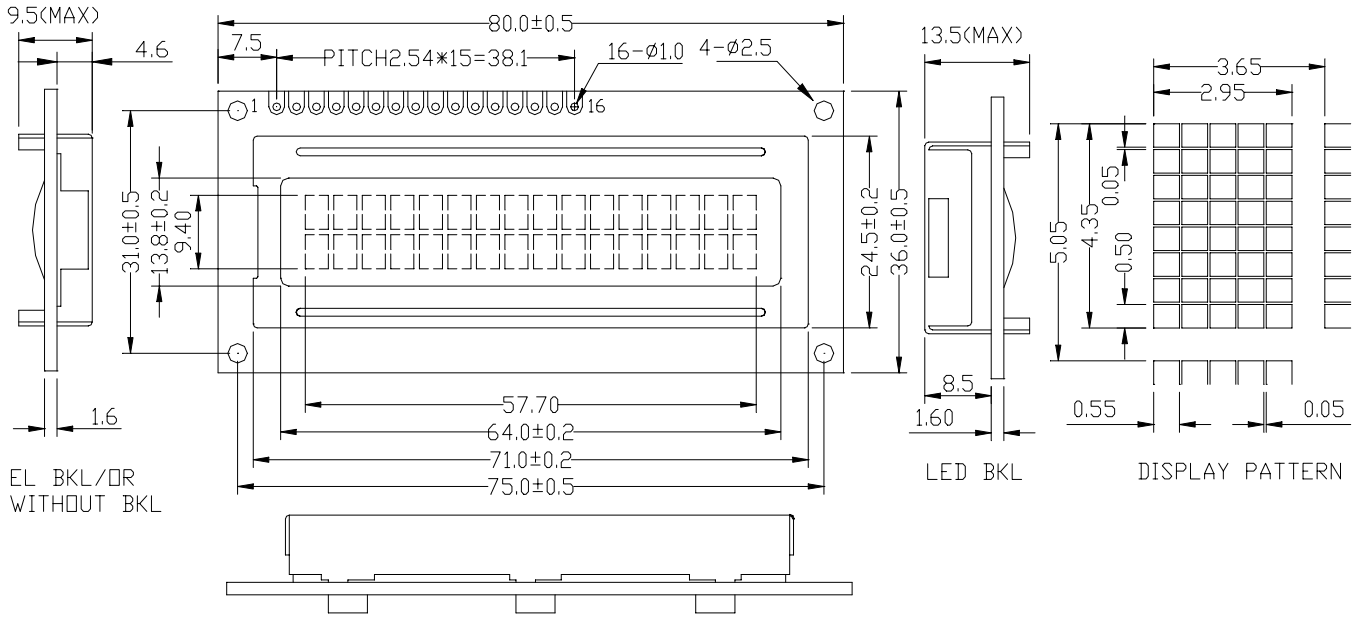
GDM1602A

SPECIFICATIONS OF LCD MODULE

Features

1. 5x8 dots
2. Built-in controller (S6A0069 or Equivalent)
3. Power supply: Type 5V
4. 1/16 duty cycle
5. LED backlight
6. N.V. option

Outline dimension

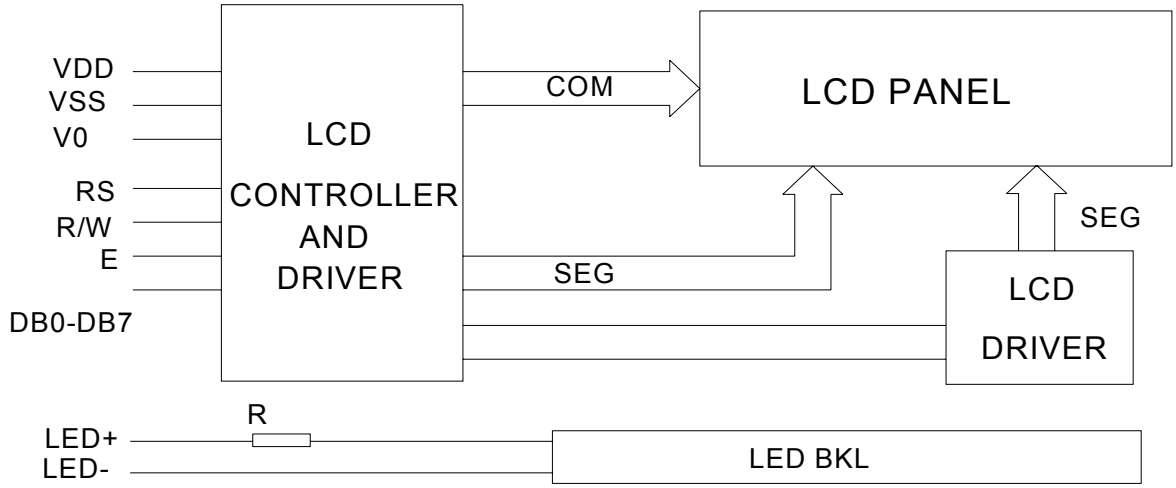


Absolute maximum ratings

Item	Symbol	Standard			Unit
Power voltage	V_{DD}-V_{SS}	0	-	7.0	V
Input voltage	VIN	VSS	-	VDD	
Operating temperature range	Top	0	-	+50	℃
Storage temperature range	Tst	-10	-	+60	

***Wide temperature range is available**
(operating/storage temperature as –20~+70/-30~+80℃)

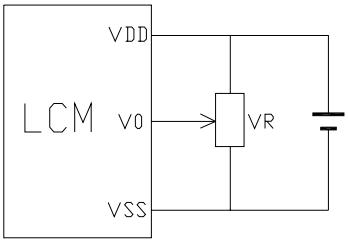
Block diagram



Interface pin description

Pin no.	Symbol	External connection	Function
1	V_{SS}	Power supply	Signal ground for LCM (GND)
2	V_{DD}		Power supply for logic for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL power supply	Power supply for BKL
16	LED-		Power supply for BKL (GND)

Contrast adjust



V_{DD}-V_0: LCD Driving voltage VR: 10k~20k

Optical characteristics

STN type display module (Ta=25℃, VDD=5.0V)

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Viewing angle	θ	$C_r \geqslant 2$	-60	-	35	deg
	Φ		-40	-	40	
Contrast ratio	C_r		-	10	-	-
Response time (rise)	T_r	-	-	300	-	ms
Response time (fall)	T_r	-	-	300	-	

Electrical characteristics

LED ratings

Item	Symbol	Min	Typ.	Max	Unit
Forward Voltage	VF	3.8	4.2	4.4	v
Forward current	If		-	160	mA
Power	P			0.12	W
Peak wave length	λp				nm
Luminance	Lv		100		Cd/m2
Operating temperature range	VOP	-20	-	+70	℃
Storage temperature range	VST	-25	-	+80	

DC characteristics

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Supply voltage for LCD	V_{DD}-V_0	Ta =25℃	-	4.5	-	V
Input voltage	V_{DD}		-	5.0	-	
Backlight supply voltage	V_F		-	-	-	
Supply current	I_{DD}	Ta=25℃, V_{DD}=5.0V	-	1.5	2.5	mA
Backlight supply current	I_F	V_{LED}=5.0V R=56Ω	-		-	
Input leakage current	I_{LKG}		-	-	1.0	uA
“H” level input voltage	V_{IH}		2.2	-	V_{DD}	V
“L” level input voltage	V_{IL}	Twice initial value or less	0	-	0.6	
“H” level output voltage	V_{OH}	LOH=-0.25mA	2.4	-	-	
“L” level output voltage	V_{OL}	LOH=1.6mA	-	-	0.4	

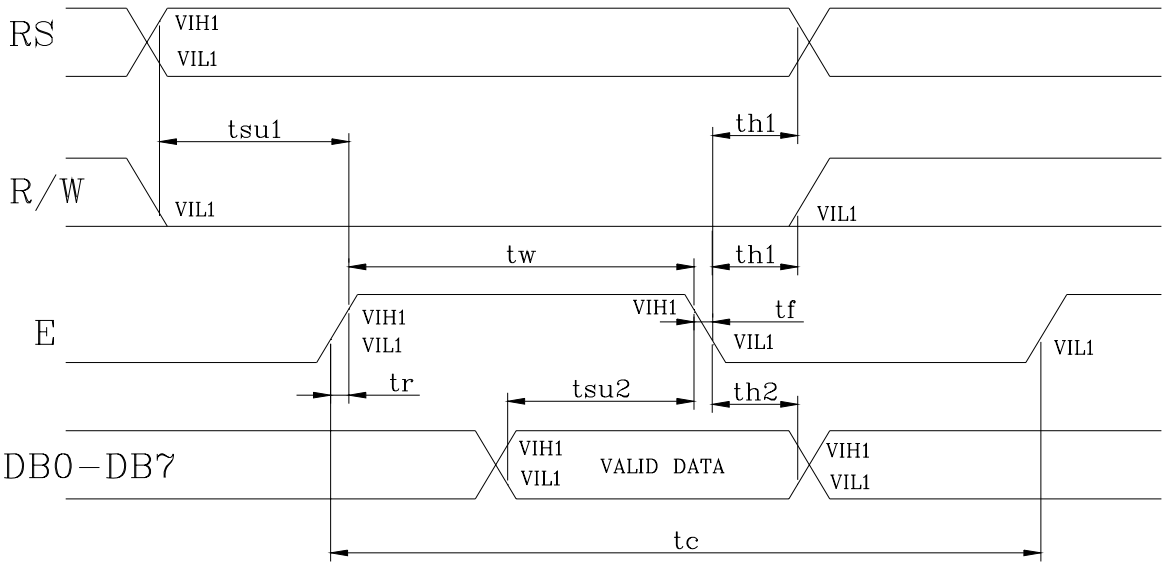
Read cycle (Ta=25℃, VDD=5.0V)

Parameter	Symbol	Test pin	Min.	Typ.	Max.	Unit
Enable cycle time	t_c	E	500	-	-	ns
Enable pulse width	t_w		300	-	-	
Enable rise/fall time	t_r, t_f		-	-	25	
RS; R/W setup time	t_{su}	RS; R/W	100	-	-	
RS; R/W address hold time	t_h	RS; R/W	10	-	-	
Read data output delay	t_d	DB0~DB7	60	-	90	
Read data hold time	t_{dh}		20	-	-	

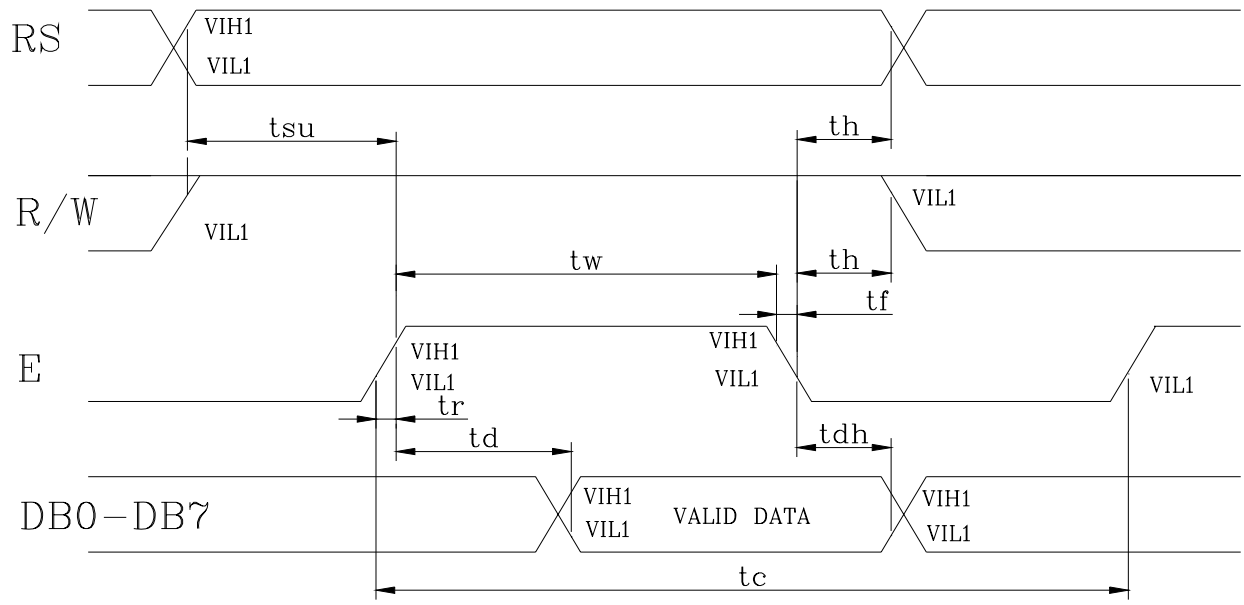
Write cycle (Ta=25℃, VDD=5.0V)

Parameter	Symbol	Test pin	Min.	Typ.	Max.	Unit
Enable cycle time	t_c	E	500	-	-	ns
Enable pulse width	t_w		300	-	-	
Enable rise/fall time	t_r, t_f		-	-	25	
RS; R/W setup time	t_{su1}	RS; R/W	100	-	-	
RS; R/W address hold time	t_{h1}	RS; R/W	10	-	-	
Read data output delay	t_{su2}	DB0~DB7	60	-	-	
Read data hold time	t_{h2}		10	-	-	

Write mode timing diagram



Read mode timing diagram



Instruction description

Outline

To overcome the speed difference between the internal clock of KS0066U and the MPU clock, KS0066U performs internal operations by storing control in formations to IR or DR. The internal operation is determined according to the signal from MPU, composed of read/write and data bus (Refer to Table7). Instructions can be divided largely into four groups:

1) KS0066U function set instructions (set display methods, set data length, etc.)
2) Address set instructions to internal RAM
3) Data transfer instructions with internal RAM
4) Others

The address of the internal RAM is automatically increased or decreased by 1.

Note: during internal operation, busy flag (DB7) is read “High”.
Busy flag check must be preceded by the next instruction.

Instruction Table

Instruction	Instruction code										Description	Execution time (fosc= 270 KHZ
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write “20H” to DDRA and set DDRAM address to “00H” from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to “00H” From AC and return cursor to Its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction And blinking of entire display	39us
Display ON/ OFF control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and Blinking of cursor (B) on/off Control bit.	
Cursor or Display shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display Shift control bit, and the Direction, without changing of DDRAM data.	39us
Function set	0	0	0	0	1	DL	N	F	-	-	Set interface data length (DL: 8- Bit/4-bit), numbers of display Line (N: =2-line/1-line) and, Display font type (F: 5x11/5x8)	39us
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address Counter.	39us
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address Counter.	39us
Read busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal Operation or not can be known By reading BF. The contents of Address counter can also be read.	0us
Write data to Address	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43us
Read data From RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43us

NOTE:

When an MPU program with checking the busy flag (DB7) is made, it must be necessary 1/2fosc is necessary for executing the next instruction by the falling edge of the "E" signal after the busy flag (DB7) goes to "Low".

Contents

1) Clear display

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

Clear all the display data by writing "20H" (space code) to all DDRAM address, and set DDRAM address to "00H" into AC (address counter).

Return cursor to the original status, namely, bring the cursor to the left edge on the fist line of the display.

Make the entry mode increment (I/D="High").

2) Return home

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	-

Return home is cursor return home instruction.

Set DDRAM address to "00H" into the address counter.

Return cursor to its original site and return display to its original status, if shifted.

Contents of DDRAM does not change.

3) Entry mode set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	SH

Set the moving direction of cursor and display.

I/D: increment / decrement of DDRAM address (cursor or blink)

When I/D="high", cursor/blink moves to right and DDRAM address is increased by 1.
When I/D="Low", cursor/blink moves to left and DDRAM address is increased by 1.
*CGRAM operates the same way as DDRAM, when reading from or writing to CGRAM.

SH: shift of entire display

When DDRAM read (CGRAM read/write) operation or SH="Low", shifting of entire display is not performed. If SH ="High" and DDRAM write operation, shift of entire display is performed according to I/D value. (I/D="high". shift left, I/D="Low". Shift right).

4) Display ON/OFF control

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

Control display/cursor/blink ON/OFF 1 bit register.

D: Display ON/OFF control bit

When D="High", entire display is turned on.
When D="Low", display is turned off, but display data remains in DDRAM.

C: cursor ON/OFF control bit

When D="High", cursor is turned on.
When D="Low", cursor is disappeared in current display, but I/D register preserves its data.

B: Cursor blink ON/OFF control bit

When B="High", cursor blink is on, which performs alternately between all the "High" data and display characters at the cursor position.
When B="Low", blink is off.

5) Cursor or display shift

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	-	-

Shifting of right/left cursor position or display without writing or reading of display data.
This instruction is used to correct or search display data.
During 2-line mode display, cursor moves to the 2nd line after the 40th digit of the 1st line.
Note that display shift is performed simultaneously in all the lines.
When display data is shifted repeatedly, each line is shifted individually.
When display shift is performed, the contents of the address counter are not changed.

Shift patterns according to S/C and R/L bits

S/C	R/L	Operation
0	0	Shift cursor to the left, AC is decreased by 1
0	1	Shift cursor to the right, AC is increased by 1
1	0	Shift all the display to the left, cursor moves according to the display
1	1	Shift all the display to the right, cursor moves according to the display

6) Function set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	-	-

DL: Interface data length control bit

When DL="High", it means 8-bit bus mode with MPU.

When DL="Low", it means 4-bit bus mode with MPU. Hence, DL is a signal to select 8-bit or 4-bit bus mode.
When 4-but bus mode, it needs to transfer 4-bit data twice.

N: Display line number control bit

When N="Low", 1-line display mode is set.
When N="High", 2-line display mode is set.

F: Display line number control bit

When F="Low", 5x8 dots format display mode is set.
When F="High", 5x11 dots format display mode.

7) Set CGRAM address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

Set CGRAM address to AC.
The instruction makes CGRAM data available from MPU.

8) Set DDRAM address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Set DDRAM address to AC.
This instruction makes DDRAM data available form MPU.
When 1-line display mode (N=LOW), DDRAM address is form "00H" to "4FH".In 2-line display mode (N=High), DDRAM address in the 1st line form "00H" to "27H", and DDRAM address in the 2nd line is from "40H" to "67H".

9) Read busy flag & address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0

This instruction shows whether KS0066U is in internal operation or not.
If the resultant BF is "High", internal operation is in progress and should wait BF is to be LOW, which by then the nest instruction can be performed. In this instruction you can also read the value of the address counter.

10) Write data to RAM

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D7	D6	D5	D4	D3	D2	D1	D0

Write binary 8-bit data to DDRAM/CGRAM.
The selection of RAM from DDRAM, and CGRAM, is set by the previous address set

instruction (DDRAM address set, CGRAM address set).
RAM set instruction can also determine the AC direction to RAM.
After write operation. The address is automatically increased/decreased by 1, according to the entry mode.

11) Read data from RAM

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D7	D6	D5	D4	D3	D2	D1	D0

Read binary 8-bit data from DDRAM/CGRAM.

The selection of RAM is set by the previous address set instruction. If the address set instruction of RAM is not performed before this instruction, the data that has been read first is invalid, as the direction of AC is not yet determined. If RAM data is read several times without RAM address instructions set before, read operation, the correct RAM data can be obtained from the second. But the first data would be incorrect, as there is no time margin to transfer RAM data.

In case of DDRAM read operation, cursor shift instruction plays the same role as DDRAM address set instruction, it also transfers RAM data to output data register.

After read operation, address counter is automatically increased/decreased by 1 according to the entry mode.

After CGRAM read operation, display shift may not be executed correctly.

NOTE: In case of RAM write operation, AC is increased/decreased by 1 as in read operation.

At this time, AC indicates next address position, but only the previous data can be read by the read instruction.

Display character address code:

Display position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DDRAM address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
DDRAM address	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Standard character pattern

Annexe II :

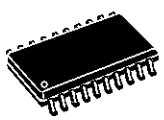
L269D

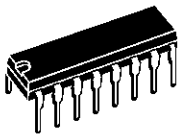
SGS-THOMSON MICROELECTRONICS

L293D
L293DD

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

- 600mA OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (non repetitive) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES





SO(12+4+4) **Powerdip (12+2+2)**

ORDERING NUMBERS:

L293DD L293D

DESCRIPTION

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoides, DC and stepping motors) and switching power transistors.

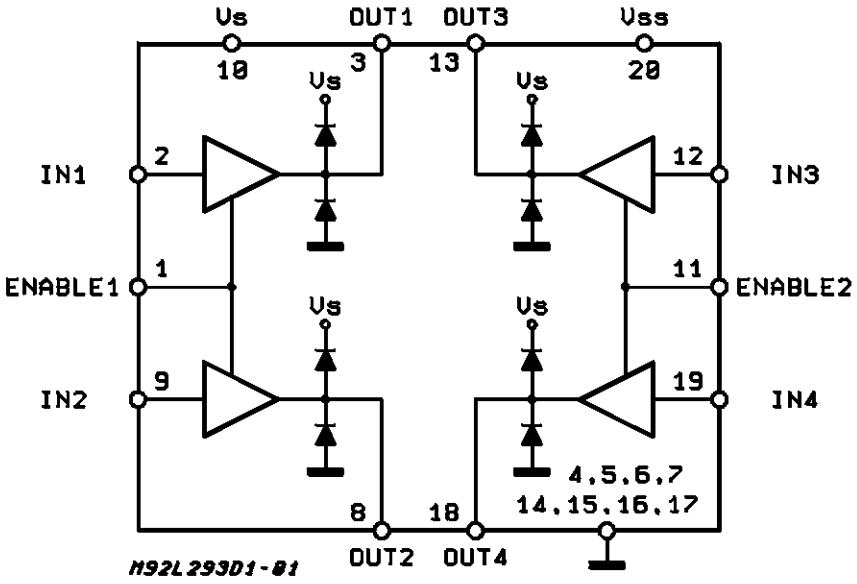
To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz.

The L293D is assembled in a 16 lead plastic packaage which has 4 center pins connected together and used for heatsinking

The L293DD is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heatsinking.

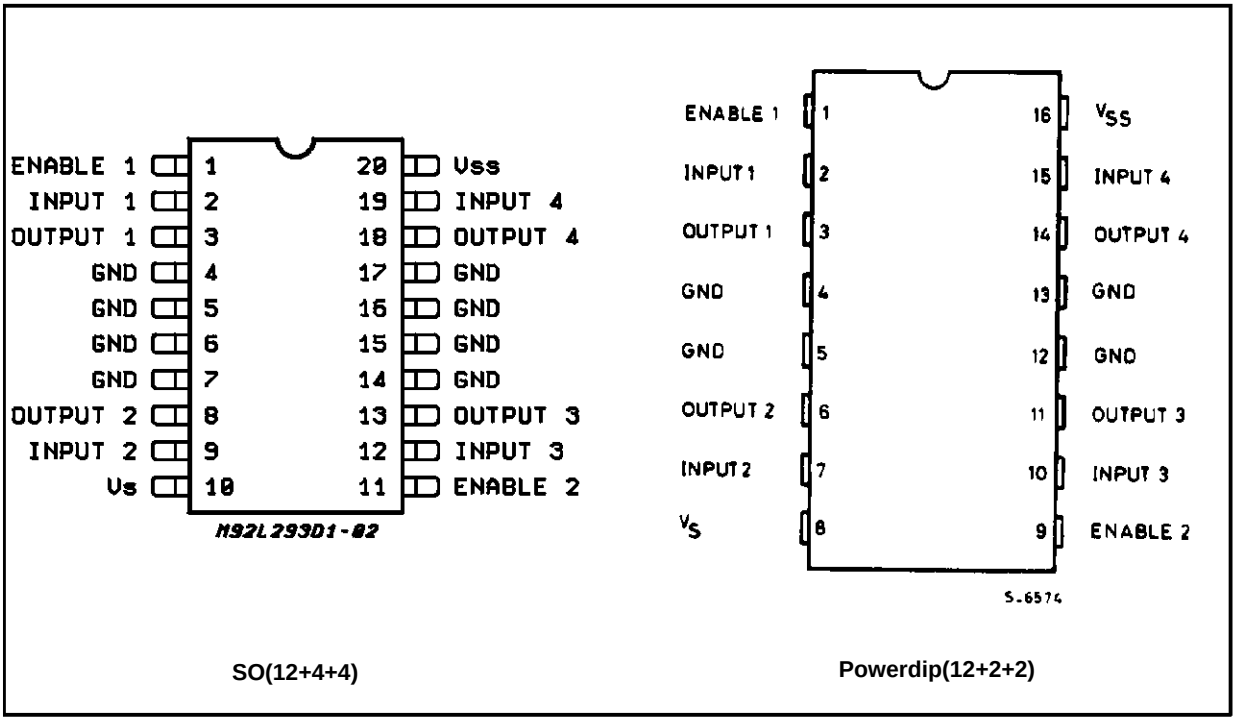
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{en}	Enable Voltage	7	V
I_o	Peak Output Current (100 µs non repetitive)	1.2	A
P_{tot}	Total Power Dissipation at T_{pins} = 90 °C	4	W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to 150	°C

PIN CONNECTIONS (Top view)



SO(12+4+4) **Powerdip(12+2+2)**

THERMAL DATA

Symbol	Decription		DIP	SO	Unit
$R_{th\ j\text{-}pins}$	Thermal Resistance Junction-pins	max.	–	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance junction-ambient	max.	80	50 (*)	°C/W
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	max.	14	–	

(*) With 6sq. cm on board heatsink.

ELECTRICAL CHARACTERISTICS (for each channel, V_S = 24 V, V_{SS} = 5 V, T_{amb} = 25 °C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 10)		V_{SS}		36	V
V_{SS}	Logic Supply Voltage (pin 20)		4.5		36	V
I_S	Total Quiescent Supply Current (pin 10)	V_i = L ; I_O = 0 ; V_{en} = H		2	6	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	24	mA
		V_{en} = L			4	mA
I_{SS}	Total Quiescent Logic Supply Current (pin 20)	V_i = L ; I_O = 0 ; V_{en} = H		44	60	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	22	mA
		V_{en} = L		16	24	mA
V_{IL}	Input Low Voltage (pin 2, 9, 12, 19)		– 0.3		1.5	V
V_{IH}	Input High Voltage (pin 2, 9, 12, 19)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
I_{IL}	Low Voltage Input Current (pin 2, 9, 12, 19)	V_{IL} = 1.5 V			– 10	µA
I_{IH}	High Voltage Input Current (pin 2, 9, 12, 19)	$2.3\ V \leq V_{IH} \leq V_{SS} - 0.6\ V$		30	100	µA
$V_{en\,L}$	Enable Low Voltage (pin 1, 11)		– 0.3		1.5	V
$V_{en\,H}$	Enable High Voltage (pin 1, 11)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
$I_{en\,L}$	Low Voltage Enable Current (pin 1, 11)	$V_{en\,L}$ = 1.5 V		– 30	– 100	µA
$I_{en\,H}$	High Voltage Enable Current (pin 1, 11)	$2.3\ V \leq V_{en\,H} \leq V_{SS} - 0.6\ V$			± 10	µA
$V_{CE(sat)H}$	Source Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = – 0.6 A		1.4	1.8	V
$V_{CE(sat)L}$	Sink Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = + 0.6 A		1.2	1.8	V
V_F	Clamp Diode Forward Voltage	I_O = 600nA		1.3		V
t_r	Rise Time (*)	0.1 to 0.9 V_O		250		ns
t_f	Fall Time (*)	0.9 to 0.1 V_O		250		ns
t_{on}	Turn-on Delay (*)	0.5 V_i to 0.5 V_O		750		ns
t_{off}	Turn-off Delay (*)	0.5 V_i to 0.5 V_O		200		ns

(*) See fig. 1.

TRUTH TABLE (one channel)

Input	Enable (*)	Output
H	H	H
L	H	L
H	L	Z
L	L	Z

Z = High output impedance
(*) Relative to the considered channel

Figure 1: Switching Times

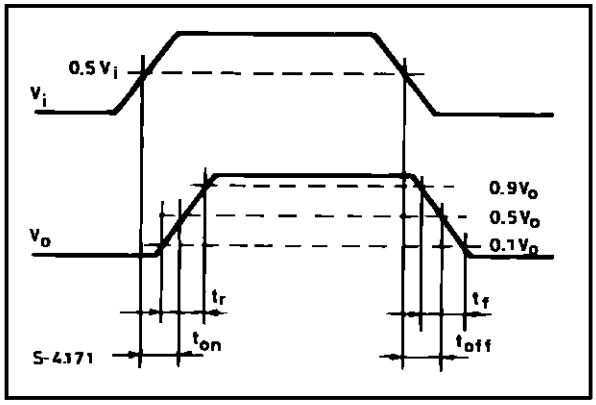
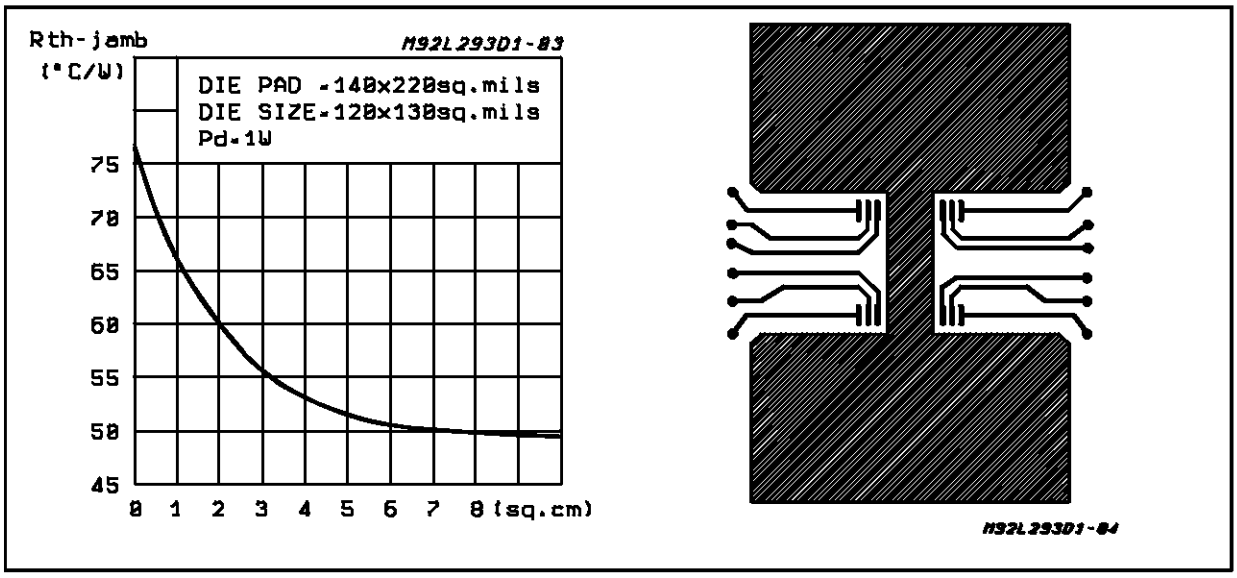
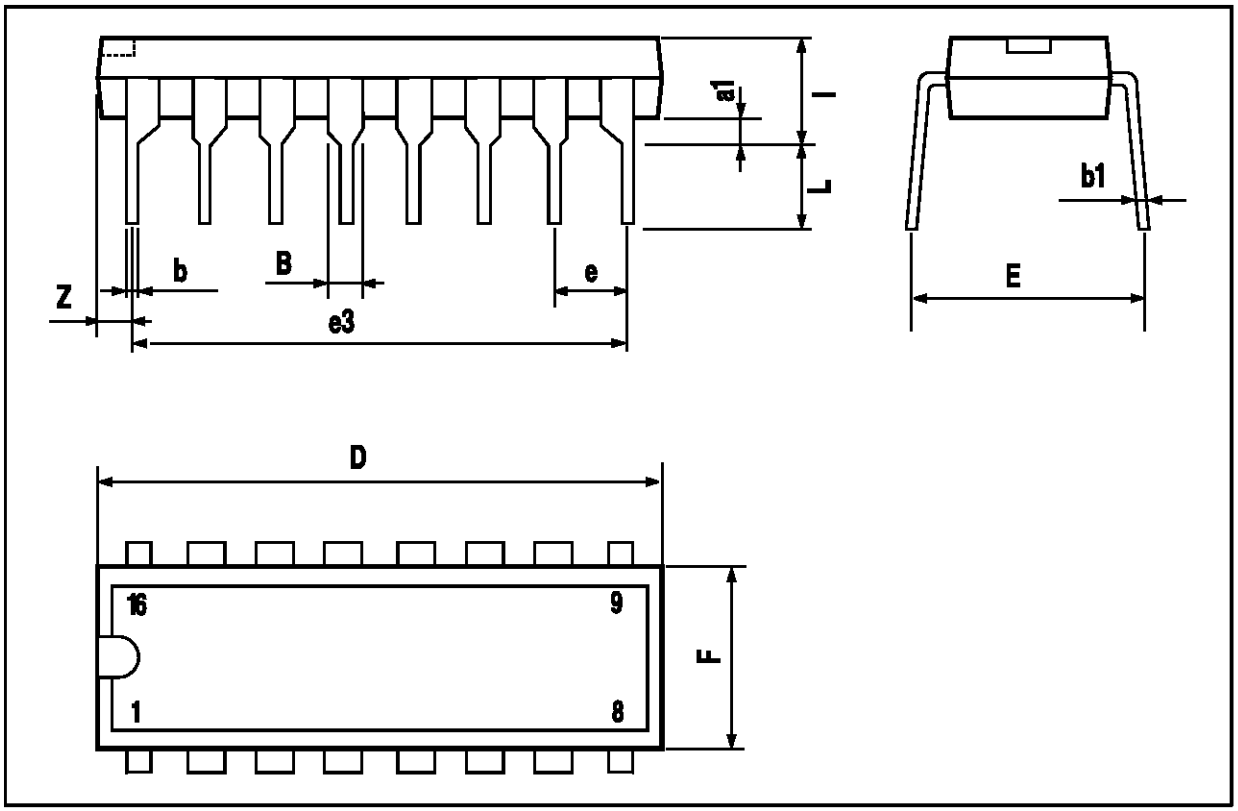


Figure 2: Junction to ambient thermal resistance vs. area on board heatsink (SO12+4+4 package)



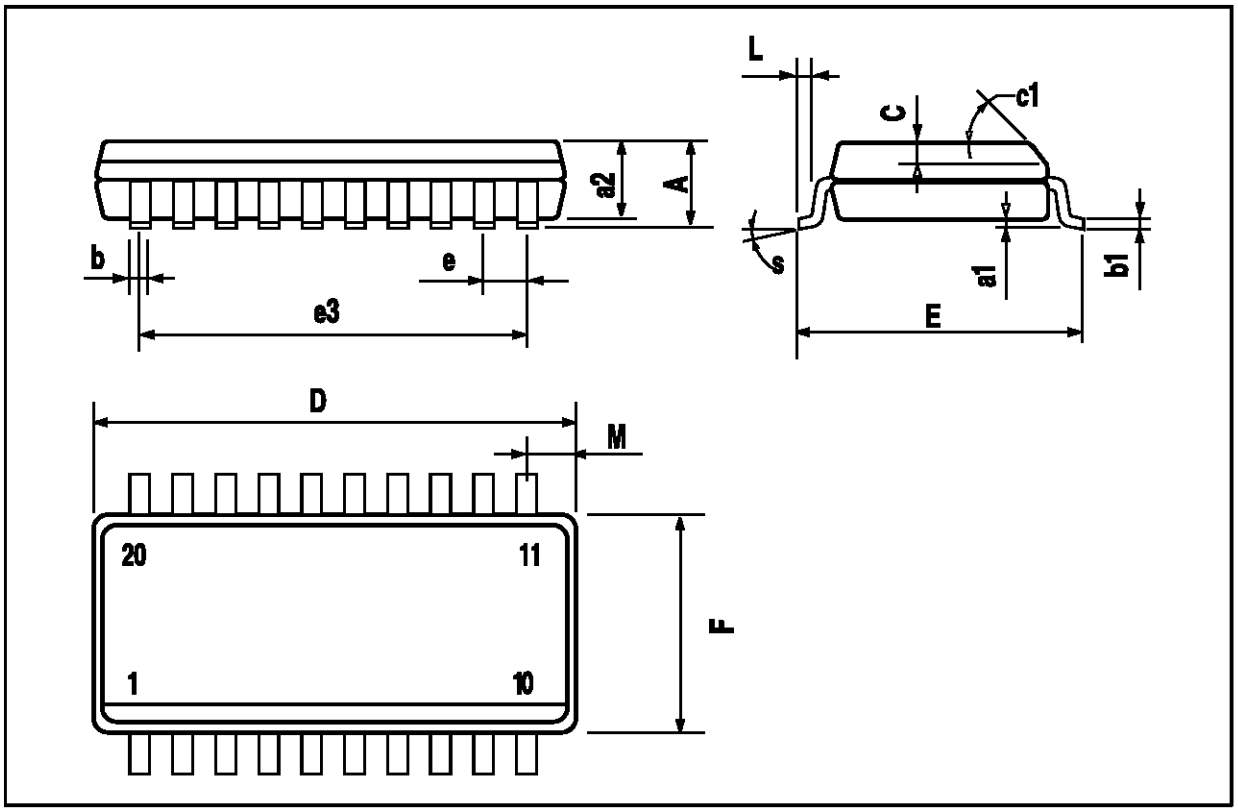
POWERDIP16 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.51			0.020		
B	0.85		1.40	0.033		0.055
b		0.50			0.020	
b1	0.38		0.50	0.015		0.020
D			20.0			0.787
E		8.80			0.346	
e		2.54			0.100	
e3		17.78			0.700	
F			7.10			0.280
I			5.10			0.201
L		3.30			0.130	
Z			1.27			0.050



SO20 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			2.65			0.104
a1	0.1		0.2	0.004		0.008
a2			2.45			0.096
b	0.35		0.49	0.014		0.019
b1	0.23		0.32	0.009		0.013
C		0.5			0.020	
c1		45			1.772	
D		1	12.6		0.039	0.496
E	10		10.65	0.394		0.419
e		1.27			0.050	
e3		11.43			0.450	
F		1	7.4		0.039	0.291
G	8.8		9.15	0.346		0.360
L	0.5		1.27	0.020		0.050
M			0.75			0.030
S	8° (max.)					



Annexe III :

La programmation du PIC

Pour la programmation de notre PIC, on a choisie de le programmer en langage C, qui offre beaucoup d'avantages par rapport à l'autre langage (Assembleur).
Pour le choix du compilateur ; on choisie le compilateur C de CCS qui est très utilisé et offre beaucoup d'avantages par rapport aux autres compilateurs. Ce dernier nous offre la possibilité de configurer les ports E/S et l'afficheur … sans pour autant écrire le programme de configuration des ports E/S et l'afficheur car c'est lui qui va le faire.

Le compilateur C (CCS : Custom Computer Services) :

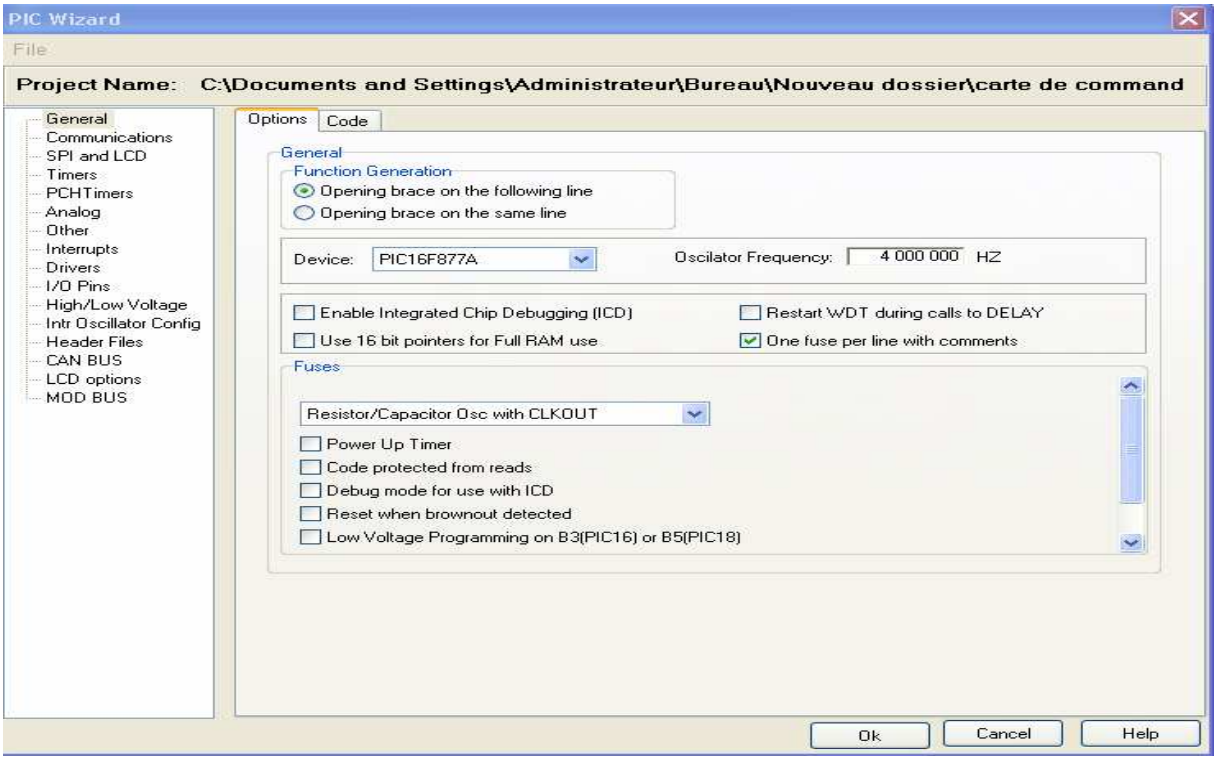
Le système de développement comporte un compilateur adapté au langage C qui traduit les instructions écrite en langage évolué qui constituent le code source, en code binaire exécutable par le microcontrôleur qui constitue le code objet voir figure.

Le CCS est un compilateur C pour les processeurs de la famille MicroChip PIC 16.Des fonctions intégrées qui permettent de développer le code de manière très aisée. L'environnement intégré de développement C donne à l'utilisateur une méthode rapide de produire un code efficace par le biais du langage évolué C

Création du projet :

Dans le menu « Projet »,cliquez sur « Pic Wizard»,la fenêtre suivante apparaît.

Après avoir nommé le projet et l'enregistré, la fenêtre suivante apparaît.



A partir de cette fenêtre, on peut configurer les ports en E/S, l'afficheur LCD,.....etc.

En validant cette fenêtre, la fenêtre principale du compilateur réapparaît, pour écrire le programme principal.

Le programme principal :

```
**********************************************************************

//                Déclaration des variables

//**********************************************************************
*****

int   coordonnees=0;

int   stepper_state_1 = 0;

int   stepper_state_2 = 0;
```

```
int   stepper_state_3 = 0;

float   x = 0;

float   y= 0;

float   z= 0;

float l1=10,l2=15,l3=20;

float q1,q2,q3;

float qq1=0,qq2=0,qq3=0;

float calcul;

int   tempo=0;

int   nbr_pas_1=0,nbr_pas_2=0,nbr_pas_3=0;


//*****************************************************************
*****

//                    Définitions

//*****************************************************************
*****

#bit bp1 = PORTC.6

#bit bp2 = PORTC.7

#bit bp3 = PORTE.1

#bit bp4 = PORTE.2
```

```
//-------------------------
// initialisation du système
//-------------------------
   PORTA=0;
   PORTB=0;
   PORTC=0;
   PORTD=0;
   PORTE=0;

//---------------------------------
// initialisation de l'afficheur lcd
//---------------------------------
   lcd_init();
//---------------------------------
```

Programme principal :

```
lcd_gotoxy(3,1);
   printf(LCD_PUTC,"ROBOT SCARA ");
   lcd_gotoxy(5,2);
   printf(LCD_PUTC,"X = %3.0f", xx);
loop:
   bp1c();
```

```
  bp2c();
  bp3c();
  bp4c();
//*****************************************************************
***************
//    fonction pour réaliser le front montant des boutons poussoirs
//*****************************************************************
***************
int button (int1 pin, int bp)

{
  static int flag_bp = 0;
  static int etat = 0;
  if(pin == 0)
  {
    bit_set(flag_bp,bp);
    delay_ms(20);
  }
  else
  {
    if(bit_test(flag_bp,bp))
```

```
    {
      bit_clear(flag_bp,bp);
      etat = 1;
    }
    else
      etat = 0;
  }

}
//*****************************************************************
***************
//            sous programmes de gestion de bp1
//*****************************************************************
***************
void bp1c(void)
{
   if(button(bp1,0))
   { lcd_putc('\f');
     lcd_gotoxy(3,1);
     printf(LCD_PUTC,"ROBOT SCARA ");
autoo:switch (coordonnees)
```

```
{
 case 0:{
  if(xx < 100.0)
  {
   xx=xx+1;
   lcd_gotoxy(5,2);
   printf(LCD_PUTC,"X = %3.0f", xx);
  }
 break;}
 case 1:{
  if(yy < 100.0)
  {
   yy=yy+1;
   lcd_gotoxy(5,2);
   printf(LCD_PUTC,"Y = %3.0f", yy);
  }
 break;}
 case 2:{
  if(zz < 100.0)
  {
   zz=zz+1;
```

```
          lcd_gotoxy(5,2);
          printf(LCD_PUTC,"Z = %3.0f", zz);
         }
       break;}
     }
   }
   ////////////////////////////////////////////////////////////////////////////
//         sous programme de gestion de bp2
////////////////////////////////////////////////////////////////////////////

void bp2c(void)
{
  if(button(bp2,1))
  {lcd_putc('\f');
   lcd_gotoxy(3,1);
   printf(LCD_PUTC,"ROBOT SCARA ");
autoo1:switch (coordonnees)
     {
      case 0:{
        if(xx > 0.0)
        {
```

```
    xx=xx-1;
    lcd_gotoxy(5,2);
    printf(LCD_PUTC,"X = %3.0f", xx);
  }
break;}
case 1:{
  if(yy > 0.0)
  {
   yy=yy-1;
   lcd_gotoxy(5,2);
   printf(LCD_PUTC,"Y = %3.0f", yy);
  }
break;}
case 2:{
  if(zz > 0.0)
  {
   zz=zz-1;
   lcd_gotoxy(5,2);
   printf(LCD_PUTC,"Z = %3.0f", zz);
  }
break;}
```

```
    }
  }
//*****************************************************************
//              sous programmes de gestion de bp3
//*****************************************************************
void bp3c(void)
{
  if(button(bp3,2))
  {lcd_putc('\f');
   lcd_gotoxy(3,1);
   printf(LCD_PUTC,"ROBOT SCARA ");
    switch (coordonnees)
    {
      case 0:{
         lcd_gotoxy(5,2);
         printf(LCD_PUTC,"Y = %3.0f", yy);
         coordonnees++;
      break;}
      case 1:{
         lcd_gotoxy(5,2);
         printf(LCD_PUTC,"Z = %3.0f", zz);
```

```
         coordonnees++;
       break;}
       case 2:{
         lcd_gotoxy(5,2);
         printf(LCD_PUTC,"X = %3.0f", xx);
         coordonnees = 0;
       break;}
     }
  }
}
//*****************************************************************
//              sous programmes de gestion de bp4
//*****************************************************************
void bp4c(void)
{
   if(button(bp4,3))
   {
     tempo=1;
     q1 = xx*xx + yy*yy + l2*l2 - l3*l3;
     calcul = 2*l2*sqrt(xx*xx + yy*yy);
     q1 = q1/calcul;
```

```
      q1 = acos(q1);
      q1 = q1 + atan (yy/xx);
      q1 = q1*180/PI;
      q1 = q1/3.75;
      q2 = xx*xx + yy*yy - l2*l2 - l3*l3;
      calcul = 2*l2*l3;
      q2 = q2/calcul;
      q2 = acos(q2);
      q2 = q2*180/PI;
      q2 = q2/3.75;
      q3 = zz - l1;
      lcd_putc('\f');
      lcd_gotoxy(1,1);
   printf(LCD_PUTC,"moteur en marche");
    }
}
if(tempo == 1)
   {
      tempo=0;
      if(q1 > qq1)
      {
```

```
    nbr_pas_1=(int)q1 - (int)qq1;
    qq1=q1;
    drive_stepper_1(50, 1, nbr_pas_1);
  }
  else
  {
    nbr_pas_1=(int)qq1 - (int)q1;
    qq1=q1;
    drive_stepper_1(50, 0, nbr_pas_1);
  }
  if(q2 > qq2)
  {
    nbr_pas_2=(int)q2 - (int)qq2;
    qq2=q2;
    drive_stepper_2(50, 1, nbr_pas_2);
  }
  else
  {
    nbr_pas_2=(int)qq2 - (int)q2;
    qq2=q2;
    drive_stepper_2(50, 0, nbr_pas_2);
```

```
    }
    if(q3 > qq3)
    {
      nbr_pas_3=(int)q3 - (int)qq3;
      qq3=q3;
      drive_stepper_3(50, 1, nbr_pas_3);
    }
    else
    {
      nbr_pas_3=(int)qq3 - (int)q3;
      qq3=q3;
      drive_stepper_3(50, 0, nbr_pas_3);
    }
    goto  loop;
  }
  else
    goto loop;
}
```

Bibliographies

Références Bibliographiques :

Mémoires d'Ingénieurs :

- A.ALILI, Conception & Réalisation d'un Automate *Programmable pour* Contrôle *Commandé autour d'un Réacteur Nucléaire à base d'un PIC16F877,* Département électronique, FGEI, UMMTO, 2002
- F.CHALLALI, F.KARAOUI, *Etude & Réalisation d'une Carte d'acquisition & de Transmission des Données à base du PIC16F877*, Département électronique, FGEI, UMMTO, 2004
- H.BELLABIOUD, A.OUMAOUCHE, *Conception & Réalisation d'une Centrale de Détection Automatique d'Incendie & d'Intrusion à base du Microcontrôleur PIC16F877,* Département électronique, FGEI, UMMTO, 2002
- L.LAMARI, N.NECHAF, *Conception & Réalisation d'un Générateur de Signaux Chaotiques à Base de Microcontrôleur PIC16F876,* Département électronique, FGEI, UMMTO, 2006
- DJ.DJELID, S.FERHAT, *Conception & Réalisation d'un Générateur d'impulsion programmable .Application à la technique de pompage de charge,* Département électronique, FGEI, UMMTO, 2008.
- S.ABAZIZ ,A.ASMANI , *Conception & Réalisation d'une commande de moteur pas à pas unipolaire piloté par microordinateur ;* FGEI, UMMTO, 2004.
- K.SMAIL,H.BELGAID, *Conception & Réalisation d'une carte de commande de 3 moteurs pas à pas piloté par microordinateur ;* FGEI, UMMTO, 2000.
- A. BENMISRA**,** Programmation des robots industriels et application sur le robot manipulateur, Blida 2007

Livres:

- CHRISTIAN TAVERNIER, *Programmation en C des PIC*, PARIS, L'USINE NOUVELLE, DUNOD, 2005.
- GERARD SAMBLANCAT, *Progresser avec Microcontrôleurs PIC*, PARIS, 2006
- J.C.CHAUVEAU, G.CHAVALIER, B.CHEVALIER *Mémotech Electronique (Circuits & Composants)*, PARIS, CASTEILLA, 2003
- PASCAL MAYEUX, *Apprendre la Programmation des PIC*, 3ème Edition, PARIS, DUNOD, 2005.
- PATRICE OGUIC , Moteurs pas à pas. L'USINE NOUVELLE, DUNOD, 2004.
- PHILIPPE COIFFET, La robotique : Principes et Applications.

Revues:

ELECTRONIQUE PRATIQUE Edition N°285 Juillet/Aout2004

Sites Internet :

***ANNE CANTEAUT* :** Cours sur le langage C :
http://www-rocq.inria.fr/secret/Anne.Canteaut/COURS_C/

BIGONOFF : Incontournable pour l'apprentissage de la programmation des PIC. ***BIGONOFF*** présente dans son site l'intégralité de ses cours pour les PIC (part1, part2, pat3, part4 & part5) ainsi que beaucoup d'autres outils tous très intéressants :
http://abcelectronique.com/bigonoff

***FABER FREDERIC* :** Un manuel pour études d'autodidacte pour le C comprenant exercices et solutions :
http://www.ltam.lu/cours-c//prg-c.htm

Forums de discussions:

http://abcelectronique.com

http://forums.futura-sciences.com

http://www.planete-sciences.org/forums/

http://www.electroforum.info/f/

http://www.edaboard.com/

IUFM de l'académie d'Aix-Marseille : un site qui regorge de cours et d'exemples d'applications pour la programmation des PIC en C :

http://www.aix-mrs.iufm.fr/formations/filieres/ge/data/PIC/PICC/indexPIC_C.htm

MICROCHIP :

http://www.microchip.com

SHANE TOLMIE**:** Il présente dans son site beaucoup d'outils idéals pour la programmation des PIC en C :

www.microchipc.com