الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR. YAHIA FARES DE MEDEA

FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

Département du Génie Électrique et de l'Informatique



Projet de Fin d'Etudes

Présenté pour l'obtention du Diplôme
de Master en Génie Électrique

Spécialité : *Systèmes Intelligents et Communicants*

THEME

Contrôle d'un chariot par la technique neuro-floue: Étude, conception et implémentation autour d'un PIC32

Proposé et dirigé par,

Dr. BOUHEDDA Mounir

Préparé et présenté par,

M. BENAIMECHE Imad Eddine

Melle. BELKACEMI Hanane

Remerciements

Merci à dieu le tout puissant et le tout miséricordieux pour nous avoir donné la santé, la volonté et le courage sans les quels ce travail n'aurait pas été réalisé.

Nous exprimons aussi nous remerciements à notre promoteur Dr BOUHEDDA Mounir pour la confiance qu'il nous a témoignée et pour avoir proposé ce sujet et qui n'a cessé de nous guider et nous diriger.

Ainsi que tous les honorables membres de jury qui nous ont fait l'honneur d'examiner et critiquer ce travail, nous comptons beaucoup sur vos apports de tout genre pour parfaire ce mémoire.

Mes sincères remerciements iront aussi à tous nos enseignants de l'Université Dr Yahia FARES de Médéa pour la qualité de l'enseignement qu'ils ont bien voulu nous prodiguer durant nos études sans ceux ce travail n'aurait certainement jamais pu voir le jour.

Nous ne manquerons pas l'occasion de remercier très chaleureusement, tous eux et à toutes celles qui nous ont encouragés et aidés de près ou de loin pendant la réalisation de ce travail.

Merci à nos camarades et particulièrement à Yazid, Mohamed, Ibrahim, Sidali, Sabrine, Karima, Kawther, Nessrine et Soumia et à tout le monde pour l'honneur qu'ils ont fait en participant à cette soutenance, nous les saluons aimablement pour le respect qu'ils nous portent et qui restera gravé toujours dans notre mémoire.

Merci à toutes nos familles pour leur amour et leur soutien constant, ainsi que tout le personnel de l'administration ainsi les responsables du laboratoire surtout Mr Yahia, Mr Skander.

A la fin à notre pays l'Algérie, nous espérons avoir accompli quelque chose pour toi.



Dédicace

A mes très chers parents

Ce modeste travail est dédié à mes très chers parents et Grande père pour leurs sacrifices pour moi avec acharnements durant toute ma carrière.

Pour leur bien vaillance leurs effort Constants et leur encouragement qu'Allah me les gardes.

A mes très chers sœurs et frères

A ceux qu'ont supportés avec nos les aléas de la vie : ma sœur Hadjer, et mes frères « Ayoub, Mohamed et Isslem »

A mes très chères familles

A tous mes oncles, surtout Djamel qui nous a quittés que dieu ait son âme et à son épouse et Kamel, et toute mes tantes.et mes cousins et mes cousines

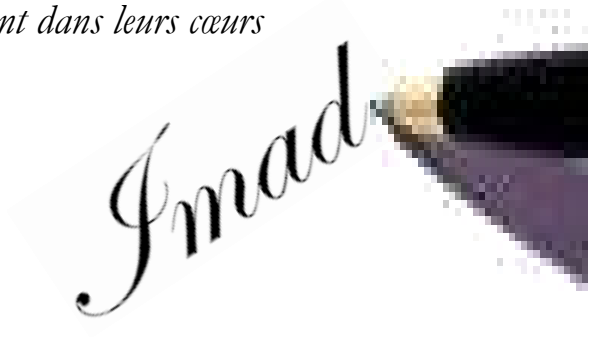
Son oublié toute ma famille qui m'a aidé de près ou de loin même avec un mot encouragement qui m'a donnée de la morale.

Et mon binôme Hanane pour l'encouragement et l'aide jusqu'à la finition de notre travail et les belles journées que j'ai passées avec elle. Et sa famille surtout sa mère pour leur confiance.

A mes très chers amis

A mes cher amis : « Yazid, Bilel, Noureddine, Oussama, Amin, Mustapha, Omar, Hocine, Mohamed »
A tous ma promotion SIC et tous les étudiants de filière Electronique
A tous mes amis que j'aime et qui m'aiment

A tous ceux qui connait Imad et me portent dans leurs cœurs



Dédicace

Je dédie avec grand plaisir ce modeste travail à tous ce qui témoigne qu'il n'y a de dieu qu'Allah et que Mohamed est le prophète de dieu.

Aux prunelles de mes yeux, à ma raison d'être à ma mère que dieu la préserve bonne santé et donne longue vie pour ses encouragements et ses prières tout au longs mes études.
A mon frère Abdelbaki, mon trésor qui a été la cause de ma réussite par son soutien moral et financier.

A ma sœur chahrazed et son partenaire dans la vie son marie Rachid et sa famille.
Merci pour votre soutien
A wassim mon neveu, reçois ici tout mon amour.

A mon oncle Dahmen et Fodhil à mes tantes surtout Aicha et Laila à mes cousines et cousins principalement Youness, Ridha, Nessrine, Yasmine, Abir, Najet, Fairouz, Khouloud, Bouchra.

A toi Imad mon binôme pour la compréhension et pour ton soutien moral surtout dans les moments difficiles pendants la réalisation de cette thèse ainsi qu'à ta famille et surtout Islam et Hadjer.

A mes amis avec lesquels j'ai partagé mes meilleurs moments surtout Sabrine et Karima et Fatima, Nessrine, Soumia, Houda, Nawal, Kawther, Sihem, Fatiha, Jamila , Nouha , Hala pour la bonne ambiance et l'entente qui a règne durant les années d'etude.

A mes camarades et tous ceux de la faculté 2 année SIC et tous ceux de djamaa el Nour.
Tous ceux qui connait « Hanane » et me portent dans leurs cœurs

Hanane



ملخص:

الهدف من هذا العمل، أن نقوم بتنفيذ أدوات التحكم في المنطق الضبابي- العصبي من أجل التحكم في توجيه عربة.

وتحقيقا لهذه الغاية، قمنا بتطبيق نموذج المنطق الضبابي- العصبي وتنفيذه بواسطة PIC32 (بطاقة Digilent CEREBOT 32MX4) وأظهرت الاختبارات التي تجرى على النظام المطبق صحة التحكم المنفذ.

كلمات مفتاحية : العصبية غامض، الضبابي، عربة, PIC32

Abstract:

The aim of this work is the hardware implementation of neuro-fuzzy technique for a buck control

For this purpose, a neuro-fuzzy model is synthesized and optimized in order to hardware implementation on a PIC32 (CEREBOT 32MX4 of Digilent).

Performed tests on the system show the validity of the implemented controller.

Keywords: buck, fuzzy, neuro-fuzzy, PIC32.

Résumé :

Le but ce travail est l'implémentation matérielle d'un contrôleur neuro-flou pour l'orientation d'un chariot.

A cet effet, un modèle neuro-flou est synthétisé et optimisé en vue d'une implémentation sur un PIC32 (carte CEREBOT 32MX4 de Digilent).

Des tests sur le système réalisé ont montré la validité du contrôleur implémenté.

Mots clés : chariot, flou, neuro-flou, PIC32.

Table de matières

LISTE DES FIGURES ..... **I**

LISTE DES TABLEAUX ..... **III**

LISTE DES ACRONYMES ..... **IV**

INTRODUCTION GENERALE **1**

CHAPITRE I : LES SYSTEMES NEURO-FLOUS **2**

1. Introduction **3**

2. Rappels sur RNA **3**

2.1. Définition **3**

2.2. Caractéristiques **4**

2.3. Apprentissage **4**

3. Présentation des systèmes flous **6**

3.1. Ensembles flous **6**

3.2. Variables linguistiques **7**

3.3 Règles et opérateur flous **8**

3.4. Structure générale d'un système flou **9**

4. Le système Neuro-flou **11**

4.1 Définition **11**

5. Le système neuro floue hybride **12**

5.1. Définition **12**

5.2. Méthode de calcul **13**

5.3. Représentation d'un système Takagi-Sugeno (TS) **14**

5.4. Représentation du modèle Takagi-Sugeno dans Matlab **16**

6. Conclusion **20**

CHAPITRE II : CONTEXTE D'ETUDE SUR LES OUTILS UTULISES **21**

1. Introduction **22**

2. Microcontrôleur PIC **22**

2.1. Microcontrôleur **22**

2.2. Définition d'un PIC **22**

2.3. Structure d'un PIC **23**

2.4. Identification d'un PIC **23**

3. PIC32 ..... **24**

3.1. Définition **24**

3.2. La raison de choix du PIC32 24
3.3. CEREBOT 32MX4 25
3.4. Présentation de la carte CEREBOT 32MX4 25
3.5. Les dispositifs incluent 26
3.6. Composition de CEREBOT 32MX4 26
3.7. Utilisation de CEREBOT 32MX4 27
4. Moteur à courant continu (CC) 32
4.1. Définition 32
4.2. Avantages 32
4.3. Inconvénients 33
5. Moteur pas à pas 33
5.1. Définition 33
5.2. Avantages 33
5.3. Inconvénients 33
5.4. Constitution 33
5.5. Les différents types des moteurs 33
6. Commande des moteurs 35
6.1 Moteur pas à pas 35
6.2. Moteur continu 36
7. Conclusion 37

CHAPITRE III : CONCEPTION ET REALISATION 38

1. Introduction 39
2. Cahier des charges 39
3. Synthèse du correcteur flou 41
3.1. Les fonctions d'appartenances 41
3.2. Les règles floues 42
3.3. Test et validation des contrôleurs flous 43
4. Synthèse du contrôleur neuro-flou 45
5. Conception 45
5.1. Partie électronique 45
5.2. Partie mécanique 49
5.3. Partie contrôle 52
6. Réalisation 55
7. Conclusion 61

CONCLUSION GENERALE **63**

ANNEXE 1 ..... **65**

1. Caractéristique **65**

2. Dimension **65**

ANNEXE 2 ..... **66**

1. Caractéristiques **66**

2. Dimension **66**

ANNEXE 3 ..... **67**

RÉFÉRENCES BIBLIOGRAPHIQUE **69**

LISTE DES FIGURES

CHAPITRE I

Figure I.1. *Représentation du neurone artificiel* 3

Figure I.2. *Illustration d'apprentissage supervisé* 5

Figure I.3. *Illustration d'apprentissage non supervisé* 5

Figure I.4. *Différents types de fonction d'appartenance* 7

Figure I.5. *Exemple d'un variable linguistique* 8

Figure I.6. *Structure d'un système flou* 9

Figure I.7. *Méthode d'inférence MAX-MIN* 11

Figure I.8. *Notion de système neuro-flou* 12

Figure I.9. *Principe de fonctionnement du neuro-flou hybride.* 13

Figure I.10. *Le réseau de neuro-flou de Takagi-Sugeno* 14

Figure I.11. *Le TOOLBOX ANFIS* 16

Figure I.12. *Fenêtre de chargement de donnée* 17

Figure I.13. *Présentation des informations chargées* 17

Figure I.14. *Menu d'explicite de la fonction* 18

Figure I.15. *Surface de décision du système neuro-flou* 19

Figure I.16. *Structure de notre système neuro-flou* 19

CHAPITRE II

Figure II.1. *Famille PIC* 23

Figure II.2. *Microcontrôleur PIC32* 25

Figure II.3. *Architecture de circuit CEREBOT 32MX4* 25

Figure II.4. *Les deux types de Pmod de CEREBOT 32MX4* 27

Figure II.5. *Création d'un nouveau projet* 28

Figure II.6. *Le choix de référence de PIC* 29

Figure II.7. *Choix dans MPLAB* 29

Figure II.8. *Fenêtre de configuration bits* 30

Figure II.9. *Mode de moteur pas à pas* 34

Figure II.10. *Représentation schématique d'un moteur unipolaire* 34

Figure II.11. *Montage en pont de H* 35

Figure II.12. *Montage en pont* 36

Figure II.13. *Montage adapté au moteur CC.* 37

CHAPITRE III
Figure III.1. *La zone de charge* 39
Figure III.2. *Conception primaire de chariot* 40
Figure III.3. *Diagramme du contrôleur* 40
Figure III.4. *Organigramme du projet* 41
Figure III.5. L*es fonctions d'appartenance de* x 41
Figure III.6. *Les fonctions d'appartenance* ϕ 42
Figure III.7. *Les fonctions d'appartenance de sortie* θ 42
Figure III.8. *Exemple 1 : test de la simulation* 43
Figure III.9. *Exemple 2 : test de la simulation* 44
Figure III.10. *Exemple 3 : test de la simulation* 44
Figure III.11. *Architecture de montage moteur cc* 46
Figure III.12. *Architecture de montage moteur pas à pas* 47
Figure III.13. *La source tension GwINSTEK GPS-2303* 47
Figure III.14. *La carte CEREBOT utilise* 48
Figure III.15. *Circuit de commande* 48
Figure III.16. *Schéma synoptique du chariot* 50
Figure III.17. *Mécanisme d'orienter la roue* 51
Figure III.18. *Mécanisme d'engrenage des roues d'arrières* 52
Figure III.19. *schéma pour contrôler le moteur cc* 53
Figure III.20. *schéma pour contrôler le moteur cc* 54
Figure III. 21. *Organigramme de fonctionne le moteur pas à pas* 54
Figure III.22. *Diagramme de notre application* 56
Figure III.23. *Exemple 1 : test de la réalisation* 57
Figure III.24. *Résultat de l'exemple 1 dans Matlab* 57
Figure III.25. *Exemple 2 : test de la réalisation* 58
Figure III.26. *Résultat de l'exemple 2 dans Matlab* 58
Figure III.27. *Exemple 3 : test de la réalisation* 59
Figure III.28. *Résultat de l'exemple 3 dans Matlab* 59
Figure III.29. *exemple de résultat non valide* 60
Figure III.30. *Résultat de l'exemple 3 dans Matlab* 60

LISTE DES TABLEAUX

CHAPITRE I

Tableau I.1. *Règles et opérations flous* 8

Tableau I.2. *Comparaison des systèmes neuro-flous* 12

Tableau I.3. *L'évolution de l'erreur au cours de l'apprentissage de la fonction* 18

CHAPITRE II

Tableau II.1. *Contrôle de registre de T1CON* [13] 31

Tableau II.2. *Les Différents commandes du moteur pas à pas unipolaire* 34

CHAPITRE III

Tableau III.1. *Les variables linguistiques* 42

Tableau III.2. *Les règles floues utilise pour logique flou* 43

Tableau III.3. *L'évolution de l'erreur au cours de l'apprentissage de la fonction* 45

LISTE DES ACRONYMES

RNA	Réseaux de Neurones Artificiels
FA	Fonction d'Activation,
NF	Neuro Flou
SIF	Système d'Inférence Floue
TS	Takagi-Sugeno
ANFIS	Adaptive Network based Fuzzy Inference System
MFs	Member ship Functions
Trimpf	Triangular-shaped built-in membership function
PIC	Programmable Interface Contrôler
RISC	Reduce Instructions Construction Set
EEPROM	Electrically-Erasable Programmable Read-Only Memory
RAM	Random Access Memory
OTP	One-Time Programmable
MIPS	Microprocessor without Interlocked Pipeline Stages
LED	Light-Emitting Diode
PLL	Phase-Locked Loop
CC	Courant Continu

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Dans la plupart des systèmes de contrôle, l'introduction des techniques intelligentes est de plus en plus utilisée.

Le problème majeur de l'utilisation de ces techniques autour de solution numérique est la difficulté de l'implémentation.

Montré par l'implémentation d'un contrôleur intelligent pour l'orientation et le déplacement d'un chariot, nous avons synthétisé un contrôleur neuro-flou.

Plusieurs solutions peuvent être utilisées pour réaliser a un contrôleur à cet effet mais qui utilisé généralement des capteurs afin de diviser on orienter le chariot.

La problématique de l'étude décrite dans ce mémoire réside dans la synthèse d'une architecture neuro-flou optimal en vue d'une implémentation autour d'un pic à 32 bits.

Pour reprendre aux objectifs visés On se basera sur les moyens théoriques et méthodes logiques de conception. En effet, la demande d'implémentation des architecture neuro-flou sur un circuit à base de PIC nécessite une parfaite maitrise de la programmation hardware en C ou assembleur. Pour la conception et l'optimisation de l'architecture intelligente, le logiciel Matlab a été utilisé ; et pour l'implémentation hardware le logiciel MPLAB a été utilisé bien sûr avec un compilateur C adéquat.

Le travail décrit dans ce mécanisme a aboutie à un développement d'une architecture optimisé neuro-flou par le contrôle du déplacement à l'orientation d'un chariot de charge

Pour cela et dans le cadre de la présente étude, notre mémoire a été scindé en trois parties principales.

La première partie est consacrée à étude de l'architecture floue et les principes de fonctionnement des systèmes neuro flous, dans la deuxième partie nous présentons la description des outils utilisés qui nous ont à obtenir à la fin de notre travail, la troisième partie est consacrée à la conception et la réalisation pratique.

A la fin de notre mémoire est achevée par une par une conclusion générale et perspective.

CHAPITRE I : LES SYSTEMES NEURO-FLOUS

1. Introduction

Dans ce chapitre, on va présenter un rappel sur les réseaux de neurones, et nous allons présenter brièvement les concepts fondamentaux de la logique floue dans une perspective industrielle, définir et expliquer la terminologie utilisée en logique floue, la théorie des ensembles flous ainsi que le mode de raisonnement propre aux variables floues. Puis une vue globale sur les systèmes neuro-flous.

2. Rappels sur RNA

2.1. Définition

Les RNA abréviation de « réseaux de neurones artificiels »sont des réseaux fortement composés de plusieurs processeurs élémentaires fonctionnant en parallèle. Chaque processeur calcule une sortie unique sur la base des informations qu'il reçoit, la Figure I.1 représenté la structure générale de RNA. [1]

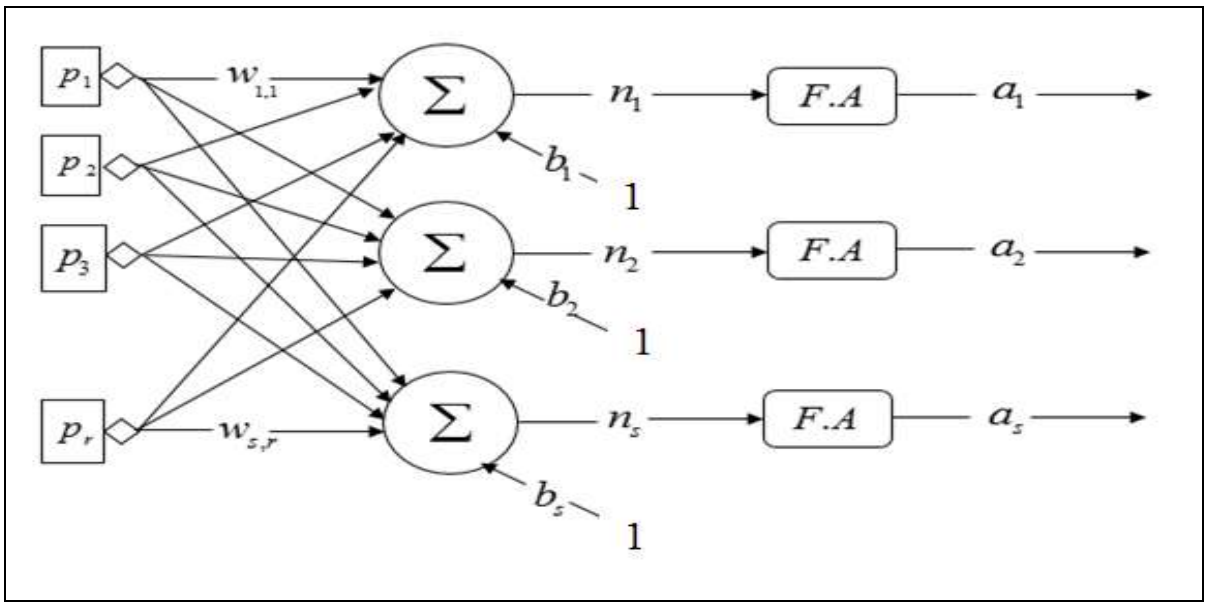


Figure I.1. *Représentation du neurone artificiel*

Avec :

- $p_1 ... p_r$: Entrées.
- $w_{1,1} ... w_{s,r}$: Poids sur les entrées.
- b_s : Biais.
- $n_1 ... n_s = net$: Niveau d'activation.
- $F.A$: Fonction d'activation, où les fonctions les plus utilisées sont la fonction linéaire et la fonction sigmoïde.
  - fonction tangente hyperbolique

$$f(x) = \tanh(\frac{x}{2}) \quad \text{(I. 1)}$$

  - fonction sigmoïde

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{(I. 2)}$$

- $a_1 ... a_s = out$: Sortie du neurone.

2.2. Caractéristiques

Les caractéristiques de RNA sont :

- Un nombre fini de processeurs élémentaires (neurones).
- Liens pondérés passant un signal d'un neurone vers d'autres.
- Plusieurs signaux d'entrée par neurone.
- Un seul signal de sortie.

2.3. Apprentissage

2.3.1. Définition

L'apprentissage est donc défini comme étant tout algorithme d'ajustement des coefficients synaptiques (poids). Donc, les variables modifiées pendant la phase d'apprentissage sont généralement les poids des connexions entre neurones. Le réseau est testé plusieurs fois au fur et à mesure que l'on ajuste les poids, avant qu'il satisfasse à une réponse désirée. Une fois le but est réalisé les poids seront fixé et on peut alors passer à la phase d'utilisation du réseau, principalement il existe deux types : [2]

- Apprentissage supervise

Permet de déterminer les poids synaptiques à partir d'exemples étiquetés aux quels un expert a associé des réponses du réseau. Les paramètres du réseau sont donc modifiés de manière à minimiser l'erreur entre la sortie cible (fournie par l'expert) et la sortie réelle du réseau. La Figure I.2 illustre, un exemple d'apprentissage supervisé.

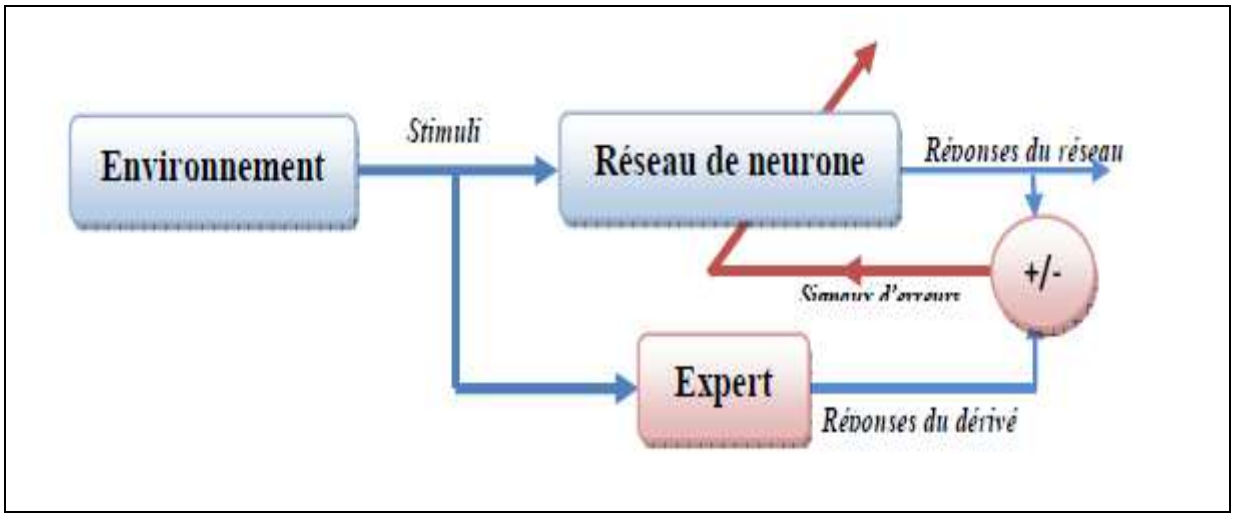


Figure I.2. *Illustration d'apprentissage supervisé*

- Apprentissage non supervise :

Les données fournies en entrée ne contiennent pas d'information sur la sortie désirée. L'apprentissage est réalisé à l'aide des règles qui modifient les paramètres du réseau en fonction des exemples fournis en entrée. La Figure I.3 illustre, un exemple d'apprentissage non supervisé.

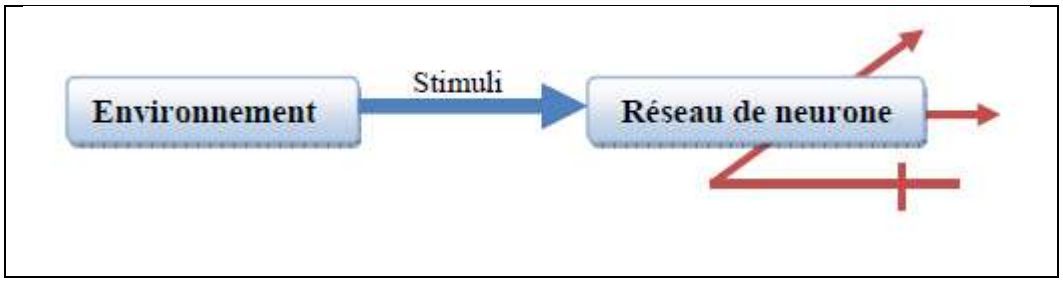


Figure I.3. *Illustration d'apprentissage non supervisé*

2.3.2. Principe de l'apprentissage

Cet algorithme est basé sur l'algorithme de la rétro-propagation du gradient descendant dont les entrées du neurone chargé de la défuzzification sont normalisées. Cette normalisation est nécessaire car elle permet de manipuler des valeurs inférieures ou égales à l'unité. Une telle approche est réalisée par la division de chaque entrée de ce neurone, sur la somme de toutes ses entrées. [2]

2.3.3. L'avantage de l'apprentissage

L'avantage d'un tel algorithme d'apprentissage est que les poids sont modifiées un grand nombre de fois au cours de chaque époque. Et comme chaque modèle d'apprentissage modifiée les poids dans des directions légèrement différentes, l'apprentissage ne se cantonnera pas facilement aux minima locaux, ou à un état dans lequel tous les petits changements effectués dans les poids ne feront qu'empirer l'erreur quadratique moyenne, même si la solution optimale n'a pas encore été trouvée.

3. Présentation des systèmes flous

Les systèmes flous peuvent être considérés comme des systèmes logiques qui utilisent des règles linguistiques pour établir des relations entre leurs variables d'entrée et de sortie. Ils sont apparus pour la première fois dans les années soixante-dix avec des applications dans le domaine du contrôle des processus.

Aujourd'hui, les applications des systèmes flous sont très nombreuses outre la commande, ils sont utilisés aussi pour la modélisation. Nous présentons brièvement quelques notions de base de ces systèmes. [3]

3.1. Ensembles flous

La notion d'ensemble flou a été proposée par Zadeh [3], en introduisant un caractère graduel de l'appartenance d'un élément à un ensemble donné. Cela permet une meilleure représentation des termes et des connaissances vagues que nous manipulons dans notre vie quotidienne.

Mathématiquement, un ensemble flou A d'un univers de discours μ, est caractérisé par une fonction d'appartenance (Figure I.4), notée μ_A, à une valeur dans l'intervalle [0,1] et qui associe à chaque élément x de μ un degré d'appartenance x indiquant le niveau d'appartenance de x a A.

$\mu(x)=1$ Et $\mu(x)=0$ correspondent respectivement à l'appartenance et la non-appartenance.

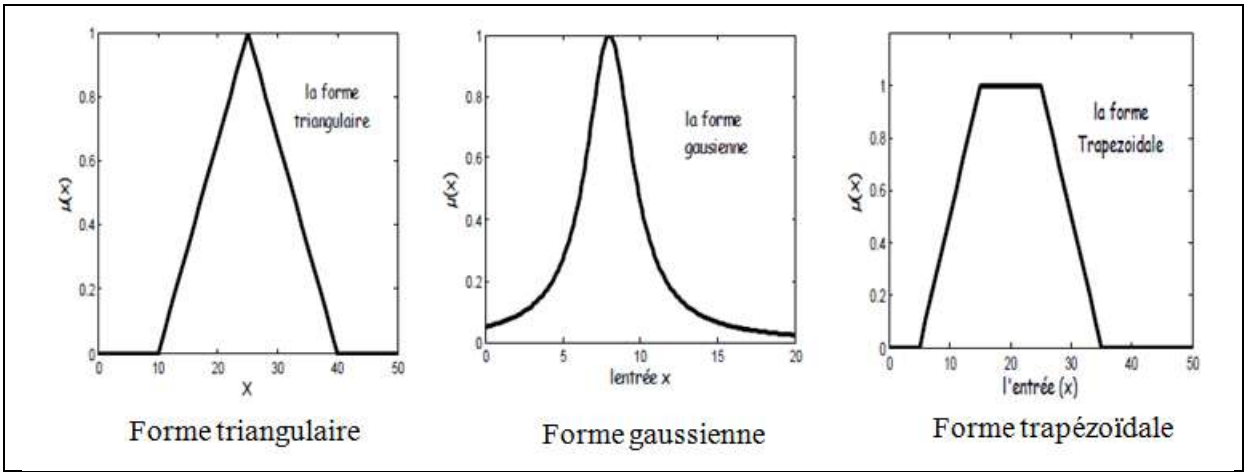


Figure I.4. *Différents types de fonction d'appartenance*

L'allure de la fonction d'appartenance à été choisis selon l'application traitée. La Figure I.4 illustre les différentes formes des fonctions d'appartenance les plus utilisées.

3.2. Variables linguistiques

Une variable linguistique appelée aussi attribut linguistique peut être définie à partir du triplet (X, U, T_x) où X est une variable définie sur l'univers de discours U et $T_x = A_1, A_2$, est un ensemble composé de sous-ensembles flous de U qui caractérise x. On associe souvent à chaque sous-ensemble flou de T_x d'une valeur ou un terme linguistique (étiquette).

Il est généralement imposé que les ensembles flous A_i doivent satisfaire la condition suivante:

$$\forall x, \exists i, (x) \neq 0 \qquad \text{(I. 3)}$$

Cette condition appelée dans la littérature, propriété d'assurance, exige que chaque élément soit affecté au moins à un ensemble flou avec un degré d'appartenance non nul. À cette condition, on ajoute souvent une propriété supplémentaire qui est le respect de la sémantique: les sous-ensembles doivent interpréter réellement les termes linguistiques qui leur sont associés. La Figure I.5 représenté un exemple sur le variable linguistique. S'arève

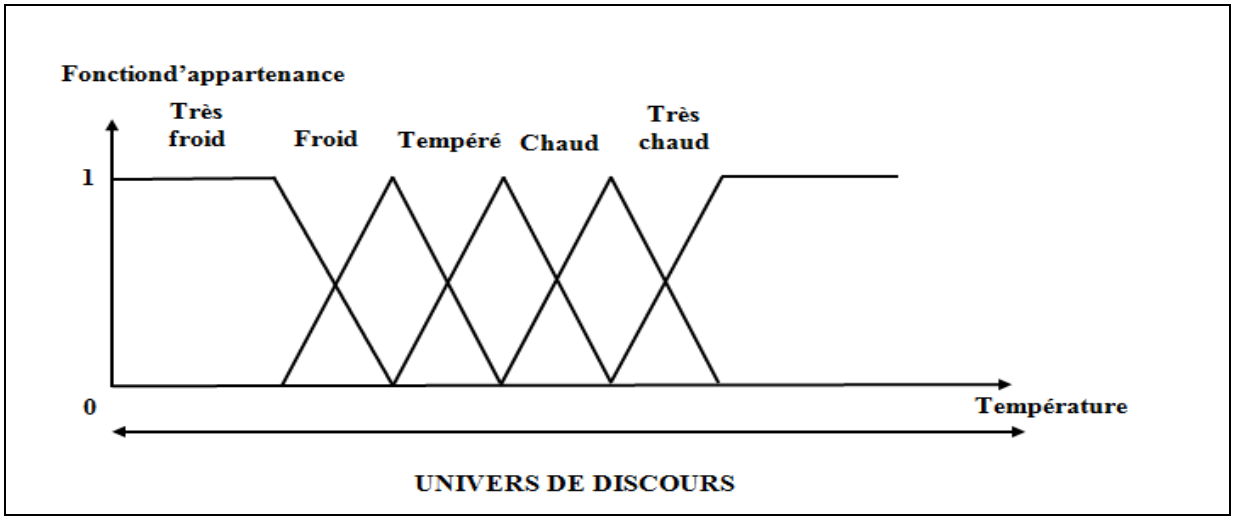


Figure I.5. *Exemple d'un variable linguistique*

- Univers du discours : Gamme de température de 0°C à 200°C.
- Variable linguistique: La température.
- Valeurs linguistiques: « Très froid» «Froid » « Tempéré » « Chaud » «Très Chaud»

3.3 Règles et opérateur flous

On appelle proposition floue élémentaire, une proposition de type X est A où (X,U,T_x) est une variable linguistique et A un sous-ensemble de T_x. Une telle proposition possède un degré de vérité égal à x où x est une valeur réelle de X. D'une manière générale, on peut combiner ces propositions élémentaires à l'aide des opérateurs logiques de conjonction et de disjonction ('et' et 'ou') mis en œuvre respectivement par des T-normes et T-conormes [1] .Le degré de vérité des nouvelles propositions obtenues peut être calculé entre autres par les équations de Tableau I.1

Tableau I.1. *Règles et opérations flous*

Dénomination	*Intersection ET (t-norme)*	*Réunion OU (t-conorme)*	*Complément NON*
Opérateurs de Zadeh MIN/MAX	$\mu_{A\cap B}(x)=\min(\mu_A(x),\mu_B(x))$	$\mu_{A\cap B}(x)=\max(\mu_A(x),\mu_B(x))$	$\mu_{\bar{A}}(x)=1-\mu_A(x)$
Probabiliste PROD/PROBOR	$\mu_{A\cap B}(x)=\mu_A(x)\times\mu_B(x)$	$\mu_A(x)+\mu_B(x)-\mu_A(x)\times\mu_B(x)$	$\mu_{\bar{A}}(x)=1-\mu_A(x)$

L'opérateur d'implication permet d'introduire la notion de règle floue qui caractérise les relations de dépendance entre plusieurs propositions floues:

$$Si\ (X_1\ est\ A_1)\ ET\ (X_2\ est\ A_2)\ ALORS\ (Y\ est\ B) \qquad (I.4)$$

Où X_1, X_2 et Y sont des variables linguistiques, A_1, A_2 et B sont des sous-ensembles flous, dans cette dernière formulation la partie (X_1 est A_1) ET (X_2 est A_2) est appelée prémisse de la règle et la partie (Y est B) est appelée conclusion (conséquent).

3.4. Structure générale d'un système flou

De manière classique, le fonctionnement interne d'un système flou repose sur la structure présentée ça Figure I.6 qui inclut quatre blocs: [4]

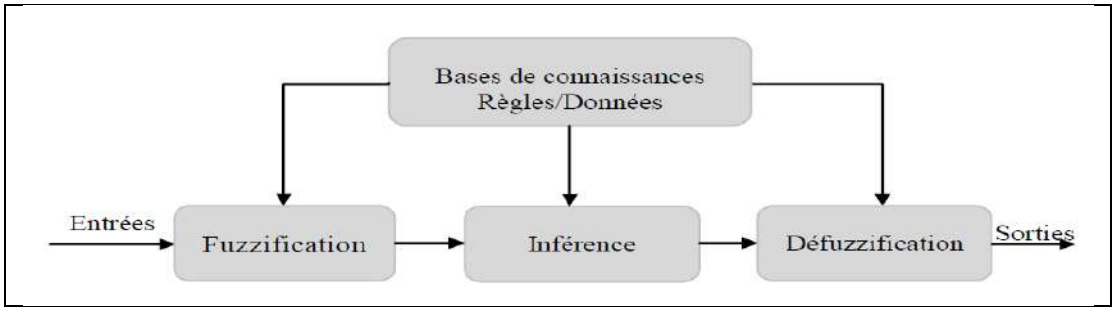


Figure I.6. *Structure d'un système flou*

3.4.1. La base de connaissances:

La conception d'une base de connaissances représente la phase dans la conception des systèmes experts. Elle comprend la base de données et la base des règles floues. [5]

3.4.1.1. La base de données

Contient la définition des ensembles flous, les facteurs d'échelle pour la normalisation des ensembles de référence et la partition de l'espace flou d'entrée et de sortie.

3.4.1.2. La base des règles floues

Le système de contrôle flou comprend un nombre de règles d'inférence reliant les variables floues d'entrée d'un système aux variables floues de sortie de ce système. Ces règles se présentent sous la forme usuelle suivante : [5]

$$SI\ (condition1)\ ET/OU\ (condition2)\ (ET/OU...)\ ALORS\ (action)\ sur\ les\ sorties \qquad (I.5)$$

L'établissement de ces règles est généralement basé sur la connaissance du problème et sur l'expérience de l'opérateur qui peut fixer le nombre de sous-ensembles, leurs fonctions

d'appartenance ainsi que les variables linguistiques. Ils existent plusieurs présentations de la base de règles telles que la description linguistique, symbolique ou par une matrice d'inférence.

3.4.2. La Fuzzification

Dans le domaine du contrôle, les données observées sont des grandeurs physiques générées par des capteurs. Il est nécessaire de convertir ces grandeurs réelles en des variables floues. Pour cela, on fait appel à une opération dite Fuzzification, qui permet de fournir les degrés d'appartenance de la variable floue à ses sous-ensembles flous en fonction de la valeur réelle de la variable d'entrée. [5]

3.4.3. Le mécanisme d'inférence

Dans cette étape, il s'agit de déterminer comment le système interprète les variables linguistiques floues. Les variables linguistiques (entrées et sorties) sont liées par les règles d'inférence. Les variables sont liées par l'opérateur « ET », tandis que les variables de sortie des différentes règles sont liées par l'opérateur « OU » et l'ensemble des règles sont liées par les connecteurs tels que «ET» et « ALORS ». La conséquence d'une règle floue est inférée par l'emploi de règle de composition, en utilisant les fonctions d'implications floues et les connecteurs « ET » et « ALORS». [3]Les méthodes d'inférences se différencient selon la combinaison et l'utilisation des opérateurs « ET » et « OU » dans les règles d'inférence. Parmi ces méthodes on trouve :

- Méthode d'inférence max-min (Mamdani).
- Méthode d'inférence max-prod (Laesen).
- Méthode d'inférence somme-prod (Sugeno).

3.4.3.1. Méthode d'inférence MAX-MIN

Cette méthode représente l'opérateur « ET » par la fonction « Min », la conclusion «ALORS» par la fonction « Max » et l'opérateur « OU » par la fonction « Min ». La représentation graphique de cette méthode d'inférence est illustrée par la Figure I.7

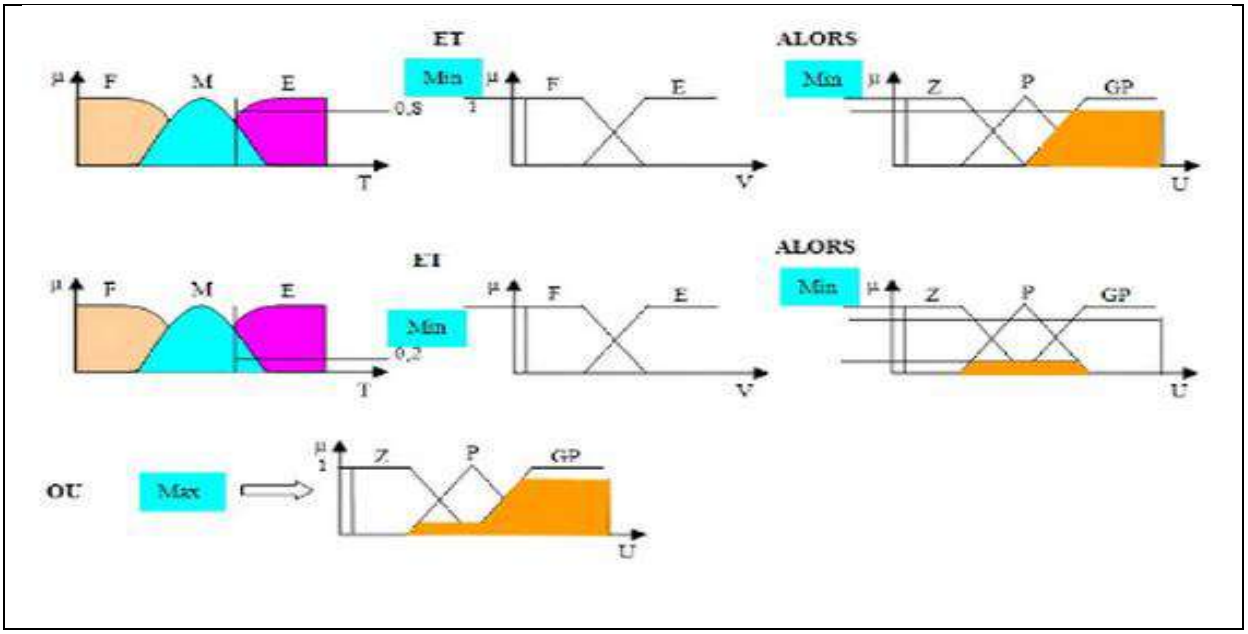


Figure I.7. *Méthode d'inférence MAX-MIN*

3.4.4. La défuzzification

La défuzzification est l'opération qui consiste à extraire d'un ensemble flou une valeur numérique considérée comme valeur significative. Dans la commande en temps réel, un critère de choix de la méthode de défuzzification est la simplicité de calcul, plusieurs méthodes existent [5], les plus utilisées sont :

- Le centre de gravité.
- La méthode de la hauteur.
- La méthode de la hauteur modifiée.
- La méthode de la valeur maximum.
- La méthode de la moyenne des centres.

4. Le système Neuro-flou

4.1 Définition

Les principales propriétés des réseaux neuro flou (NF) permettent de combiner les avantages de deux techniques et d'exploiter les capacités d'apprentissage des NF d'une part et les capacités de raisonnement de logique floue d'autre part donc les NF sont nés de l'association de réseaux de neurones avec la logique floue de manière à trier profit des avantages de chacun de ces deux techniques ,la Figure I.8 représenté cette notion [6]

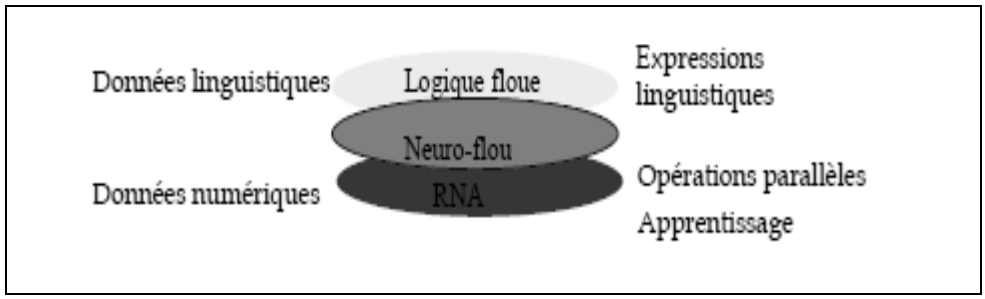


Figure I.8. *Notion de système neuro-flou*

On peut résumer la comparaison entre les réseaux de neurones, et les systèmes d'inférence floue par le Tableau I.2

Tableau I.2. *Comparaison des systèmes neuro-flous*

Système d'inférence Flou	**Système Neuro-flou**	**Les réseaux de neurones artificiels**
La base de règle peut être utilisée	La base de règle peut être utilisée.	La base de règle ne peut être utilisée
Pas d'apprentissage (utilisée la connaissance linguistique).	L'apprentissage peut être démarré de zéro ou avec une base de règles initiale	L'apprentissage peut être démarré de zéro
Interprétable (la règle IFTHEN)	Une interprétation et implémentation simples	Boite noire
Universel et implémentation simple	Le modèle mathématique non requis.	Complexite des algorithmes d'apprentissage
Les connaissances doivent être disponibles	Simple pour extraire de connaissances à partir des données (qualitatif, quantitatif).	Difficulté pour extraire les connaissances

L'intégration des réseaux de neurones et les systèmes d'inférence flous peuvent être formulés en trois principales catégories : coopérative, concurrentes et hybride.

5. Le système neuro floue hybride

5.1. Définition

Dans ce modèle les algorithmes d'apprentissage des systèmes de neurones sont utilisés pour déterminer les paramètres du système d'inférence floue (SIF). Le système Neuro-flou partage la structure des données et la représentation de connaissance [7] .Le SIF peut utiliser l'expertise humaine pour implémenter les composantes essentielles dans la base de

connaissance et la base de données et exécute le raisonnement flou pour inférer la valeur de sortie totale Figure I.9.

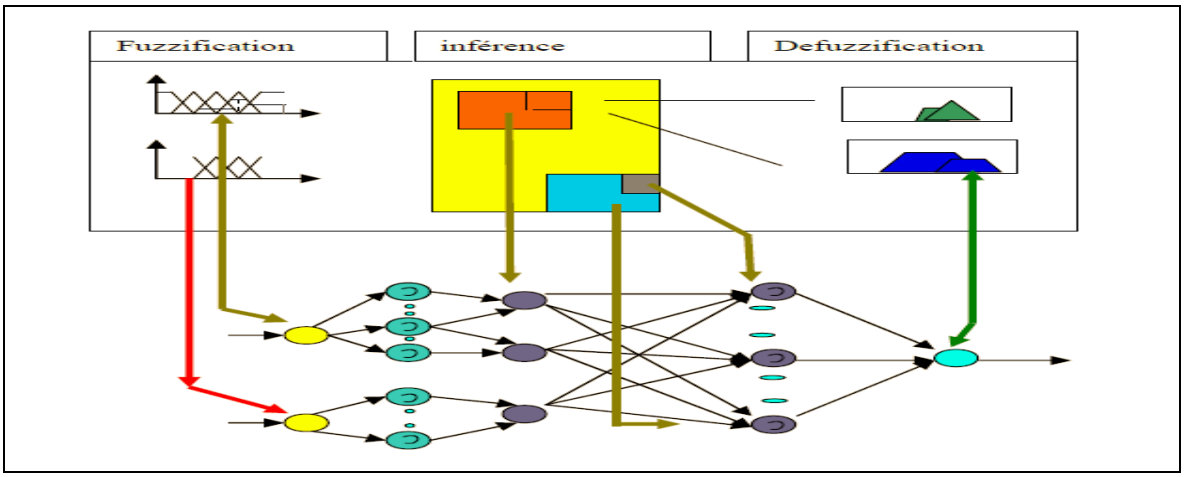


Figure I.9. *Principe de fonctionnement du neuro-flou hybride.*

5.2. Méthode de calcul

Les modèles les plus rencontrés dans la littérature des SIF, sont:

- Les modèles de Mamdani.
- Les modèles de Sugeno.

La différence, entre ces deux raisonnements, réside dans les points suivants :

- Les deux se caractérisent par IF-THEN-ELSE.
- Les deux ont les mêmes structures antérieures. Les fonctions d'appartenance (MFs «Member_ship Fonctions ») sont des ensembles flous.
- Les deux ont les mêmes structures antérieures.
- Mamdani est une structure de conséquent d'un ensemble flou et Sugeno à une fonction.
- L'incertitude de Mamdani peut être attribuée à la fois à des fonctions d'appartenance qui en découlent antécédent.
- L'incertitude de Takagi-Sugeno peut être représentée que dans les fonctions d'appartenance d'antécédent.
- On peut résumer les avantages et les inconvénients de ces modèles à ci-déçus.
  - Avantage de Mamdani
    - Intuitive
    - Acceptation généralisée
    - Bien adapte à l'intervention humaine

- Avantage de Takagi-Sugeno
  - Calcul efficace.
  - Fonctionne bien avec la technique linéaire.
  - Garanti la continuité de la surface de sortie.
  - Bien adaptée à l'analyse mathématique.
  - Plus le nombre de paramètres de conception pour chaque régle.il est possible de besoin de moins de règles.
- Inconvénient de Takagi-Sugeno
  - plus calcul efficace mais perdre l'intelligibilité linguistique.

5.3. Représentation d'un système Takagi-Sugeno (TS)

Les systèmes neuro-flous de type Takagi-Sugeno font usage d'un mélange d'algorithme de retro propagation pour faire un apprentissage des fonctions d'appartenance et la méthode des moindres-carres pour déterminer les coefficients des combinaisons linéaires dans les conclusions des règles.

La procédure d'apprentissage se fait en deux phases : dans la première phase, les modèles de l'entrée sont propagés, et les paramètres de la sortie optimaux sont estimés par la méthode des moindres carres, pendant que les paramètres antérieurs sont supposés être fixes pour le cycle courant à travers l'ensemble de l'apprentissage.

Dans la deuxième phase, la retro propagation est utilisée pour modifier les paramètres antérieurs, pendant que les paramètres de la conclusion « conséquence » restent fixes. Cette procédure sera alors répétée.

L'architecture du système neuro-flou hybride de TS est illustré sur la Figure I.10 Les détails de chaque couche sont ainsi présentés :

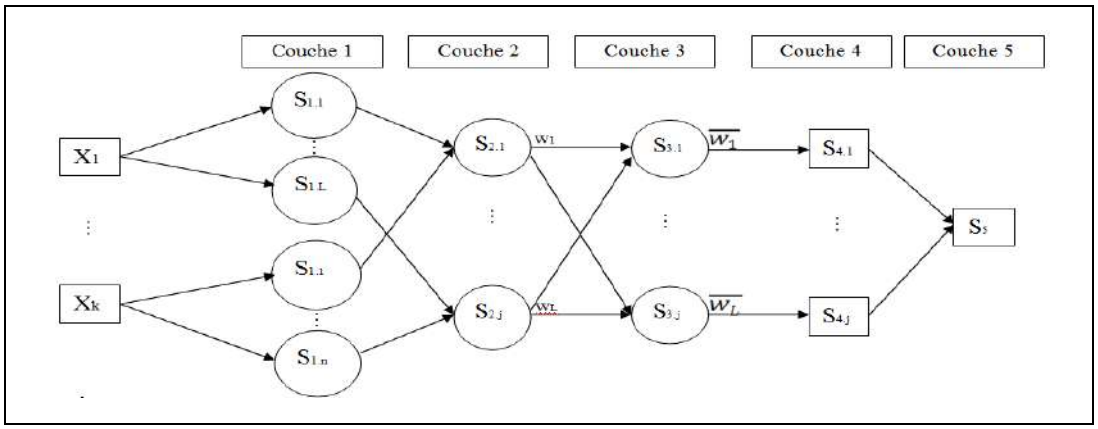


Figure I.10. *Le réseau de neuro-flou de Takagi-Sugeno*

- La couche 1: Fuzzification :

Cette couche permet de convertir les variables d'entrée dans l'espace flou à l'aide des fonctions d'appartenance.

$$S_{1,j} = \mu_{lk}\left(X_K\right) \tag{I. 6}$$

Avec $l = 1,...,nk$ et $k = 1,...,m$, où est le degré d'appartenance de $l_{i^{éme}}$ terme de la partition floue de la variable d'entrée. Différents types de fonction d'appartenance peuvent être utilisés: triangle, trapèze ou gaussienne. Les paramètres définissant ces fonctions sont appelés prémisses [8]

- La couche 2 : pondération des règles floues

Chaque neurone est associé à une règle floue et permet de définir le poids de cette règle à l'aide d'une t-norme. La t-norme la plus utilisé dans ce cas est le produit

$$S_{2,j} = \Pi\mu_{l,k} = w_j \tag{I. 7}$$

Où $u_{lk}(X_k)$ est le degré d'appartenance des termes de la partition floue utilisés dans la $j^{éme}$ règle floue

- La couche 3 : Normalisation

Cette couche a pour but de normaliser les différents poids

$$S_{3,j} = \frac{w_j}{\sum_{k=1}^{k=r} w_k} = \bar{w}_j \tag{I. 8}$$

Pour tous les $j = 1\ldots r$.

- La couche 4 : Défuzzification

Cette couche permet d'obtenir une sortie définie par une fonction du premier ordre des entrées et pondérée

$$S_{4,j} = \bar{w}_j . f_j \tag{I. 9}$$

Avec

$$f_j = \sum_k a_{jk} . X_k + b \tag{I. 10}$$

La fonction f_j peut aussi être constante. Dans ce cas, tous les $a_{jk} = 0$. Les paramètres a_{jk} et b des fonctions f_j sont appelés paramètres conséquents [8], où $k = 1,...,r$.

- La couche 5: Calcul de la sortie

La sortie Y est définie par la somme des sorties de la couches précédentes :

$$S_5 = \sum_{j=1}^{j=r} S_{4,j} = \sum_{j=1}^{j=r} \bar{w}_j . f_j \qquad \text{(I. 11)}$$

5.4. Représentation du modèle Takagi-Sugeno dans Matlab

Le modèle de Takagi-Sugeno nommé par ANFIS (Adaptive Network based Fuzzy Inference System) dans Matlab.

On utilise le TOOLBOX ANFIS de Matlab pour la modélisation neuro-flou, on donne un aperçu général sur l'utilisation de cet outil par les étapes ci-dessus.

Exécuter la commande « anfisedit » dans la zone de travail de Matlab, pour Lancer le TOOLBOX ANFIS. Après l'exécution apparait une fenêtre de travail qu'on peut voire au Figure I. 11.

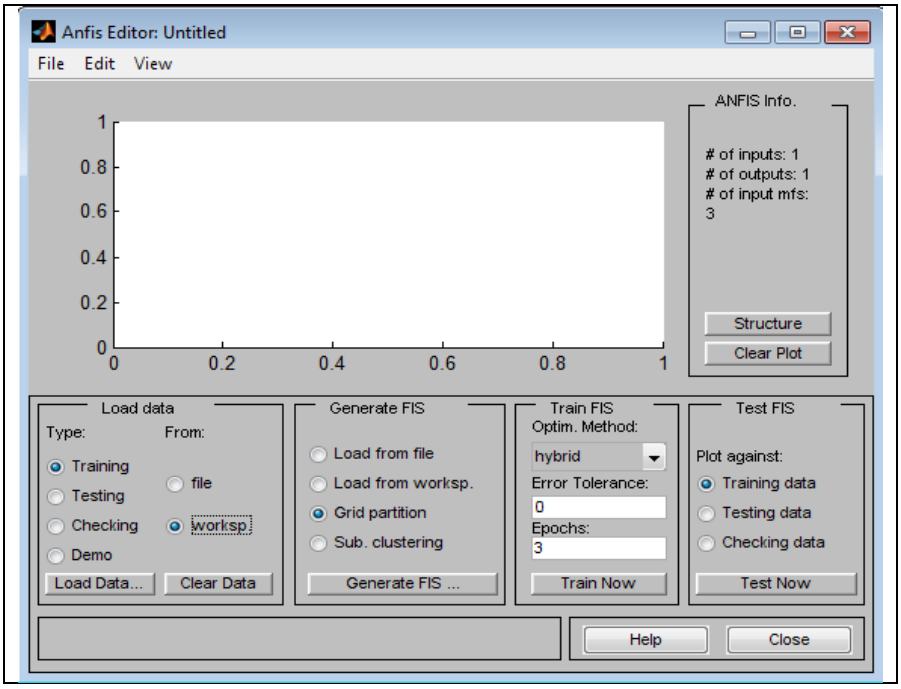


Figure I.11. *Le TOOLBOX ANFIS*

– Clique sur «Load Data» après de choisir le type « training » et de «worksp» Figure I.12

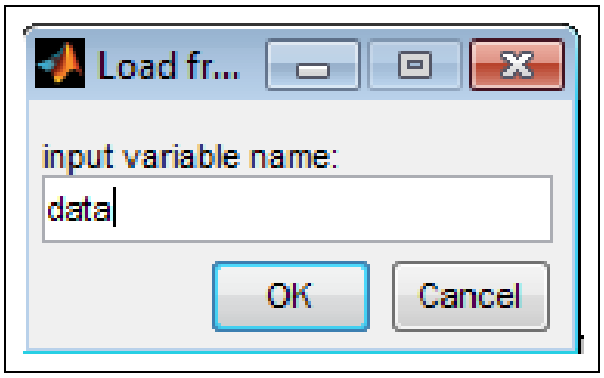


Figure I.12. *Fenêtre de chargement de donnée*

Le variable data est une matrice d'entrés de dimension (M × N) comme celle de notre exemple :

M : est le nombre de lignes qui présenter l'enchantions choisie

N : est le nombre de colonnes qui présenter comme suit (1er input, $2^{ème}$ input… $N^{ème}$ input, output).

Par exemple nous allons faire la fonction sin que : $y = sin(x)$ ou $x = [0:10]$

1er input sont des valeurs de " x " et le output est la orientation " y " met dans la matrice note " s ".

Après avoir cliqué sur le bouton « ok », le chargement de notre matrice " s " se fait, et un graphe ce présente dans la Figure I.13.

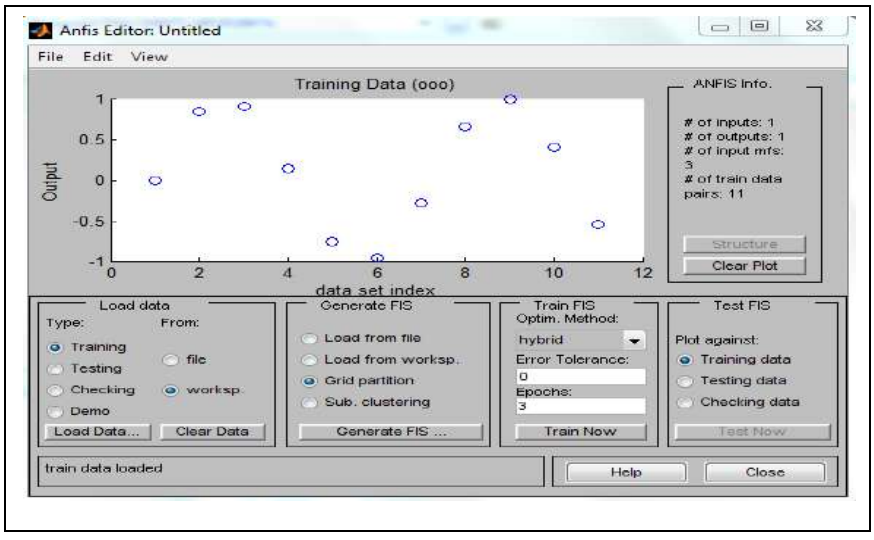


Figure I.13. *Présentation des informations chargées*

– Expliciter les fonctions d'appartenance des entrées, associées à manière de citer les paramètres de nombre de fonctions MFs, et le type des MFs (triangle, trapèze, gaussienne, gbell « sous forme de cloche »).

A cet effet on Clique sur « generate FIS … », ce bouton montre un menu apparait dans la Figure I.14, dont on peut choisir le nombre de MFs et le type, puis on choisit la fonction d'appartenance de output à partir de deux choix : « constant » ou « linéaire ».

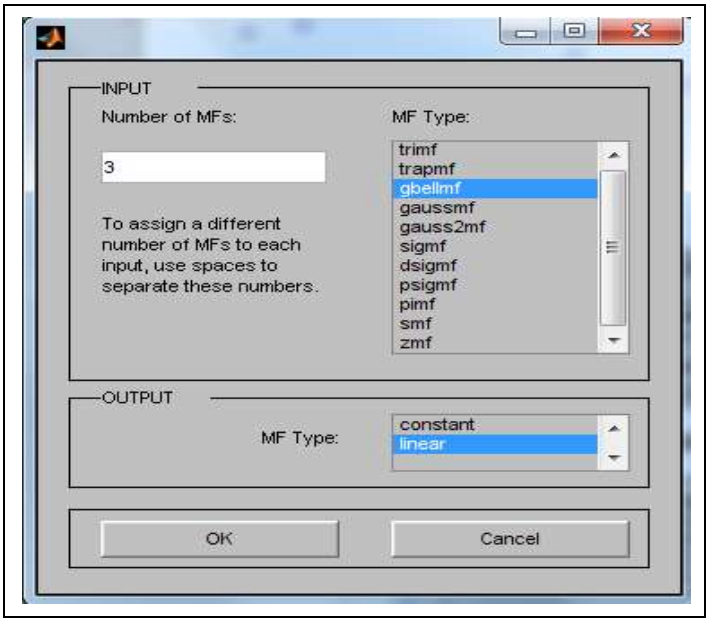


Figure I.14. *Menu d'explicite de la fonction*

A la fin de cette étape on a obtenir l'évolution de l'erreur au cours de l'apprentissage de la fonction.de 100 itérations à partir de « epochs »de « test FIS » Voir Tableau I.3

Par exemple le Tableau I.3 donne les erreurs de MSE (Mean Squared Error) pour des déférents formes des fonctions d'appartenance, le résultat peut être teste avec ce toolbox en utilisant le bouton «Test Now »

Tableau I.3. *L'évolution de l'erreur au cours de l'apprentissage de la fonction*

Fonction	Trimf	Trapmf	Gbellmf	Gaussmf	Gauss2mf
MFs=2	0.609	0.526	2.2952	0.570	0.424
MFs=3	0.136	0.108	1.8045	0.005	0.0006
MFs=4	0.202	0.531	1.8045	0.002	0.0007
MFs=5	**0.0001**	$3.26e^{-6}$	0.00036	0.00016	0.0016

D'après les résultats du Tableau I. 3, la meilleure précession est obtenir avec les fonctions d'appartenance triangulaire ou une erreur de 0.00001.En plus, cette fonction est avantageuse car elle est simple à implémenter.

Testons le résultat de l'apprentissage de notre système avec le bouton « Test Now » de « Test FIS » qui illustre dans Figure I.15.

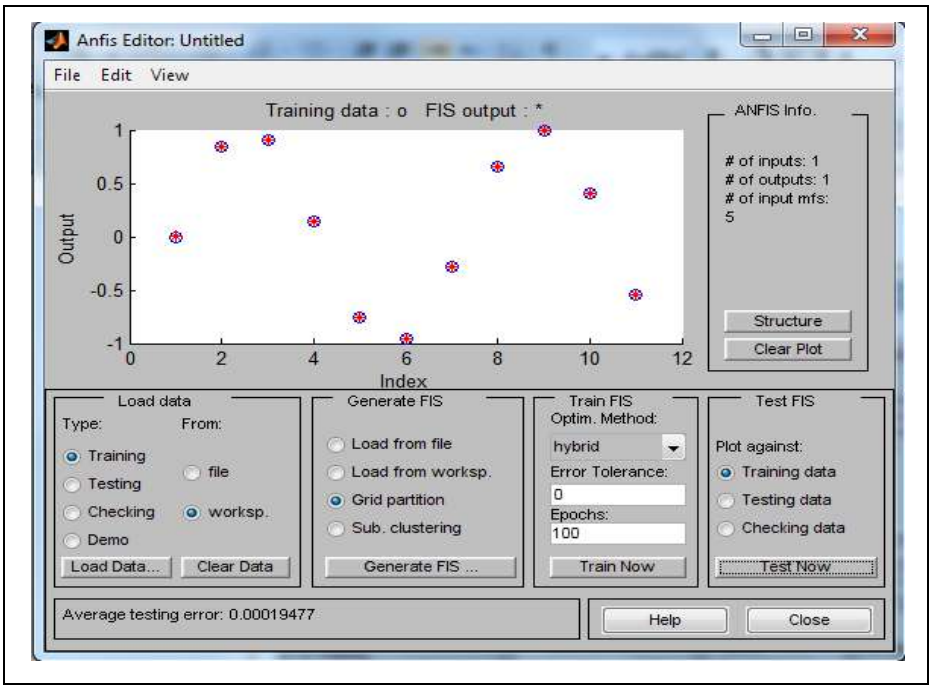


Figure I.15. *Surface de décision du système neuro-flou*

Testons notre ANFIS de structure à partir de « ANFIS info » voir Figure I.16

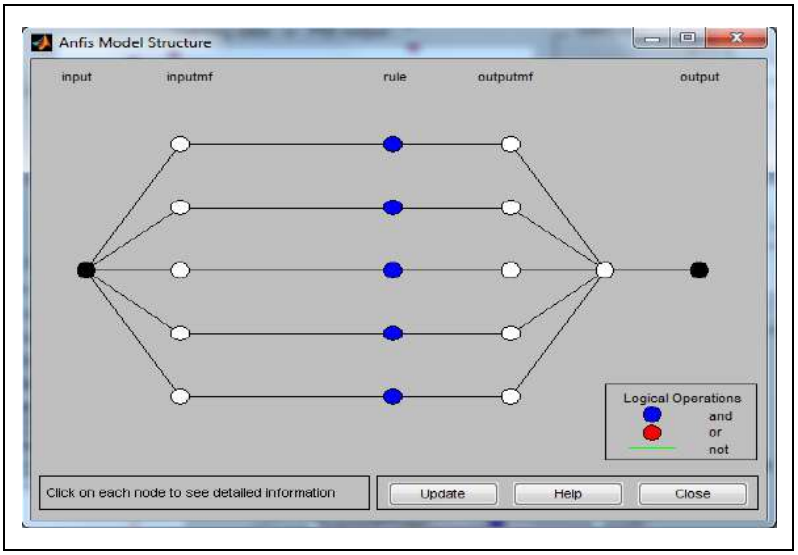


Figure I.16. *Structure de notre système neuro-flou*

6. Conclusion

Dans ce chapitre, nous avons présenté une description générale de réseaux neurones, la théorie floue, et Neuro-floue, Nous avons abordé les définitions et les comparaisons avec autre solutions qui doivent être prisent en compte pour les choix des systèmes Neuro-flous, et nous nous sommes intéressés également à l'outil de synthèse des systèmes neuro-flous en utilisant l'outil ANFIS de Matlab.

CHAPITRE II : CONTEXTE D'ETUDE SUR LES OUTILS UTULISES

1. Introduction

Dans ce chapitre nous allons décrire en détail le fonctionnement des outils qu'on a utilisé dans notre projet, ou nous présenterons les principales caractéristiques de microcontrôleur PIC ainsi que les différents types des moteurs utilisés, et leur principe de fonctionnement.

2. Microcontrôleur PIC

2.1. Microcontrôleur

Un microcontrôleur est une unité de traitement de type microprocesseur contenant tous les composants d'un système informatique, donc c'est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur.

Les microcontrôleurs sont fréquemment utilisés dans les systèmes embarqués, comme les contrôleurs des moteurs automobiles, les téléphones portables les télécommandes, les appareils de bureau, l'électroménager, les jouets,…etc. [9]

2.2. Définition d'un PIC

Les PICs (Pro**g**rammable Interface Contrôler) sont des microcontrôleurs à architecture RISC (Reduce Instructions Construction Set), ou encore composant à jeu d'instructions réduit. Son avantage est que plus on réduit le nombre d'instructions, plus leur décodage sera rapide ce qui augmenté la vitesse de fonctionnement du microcontrôleur. Il existe plusieurs centaines de modèles regroupés dans différentes séries (PIC16, PIC24, PIC32, etc.) en fonction de leurs caractéristiques. [10] La Figure II.1 illustre les déférentes familles des PICs.

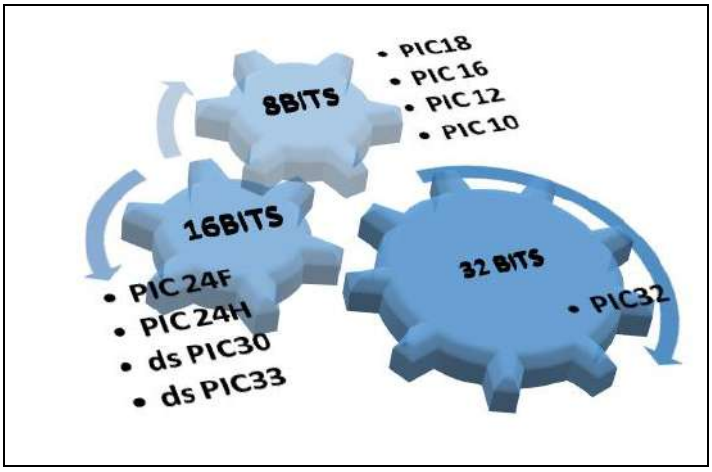


Figure II.1. *Famille PIC*

2.3. Structure d'un PIC

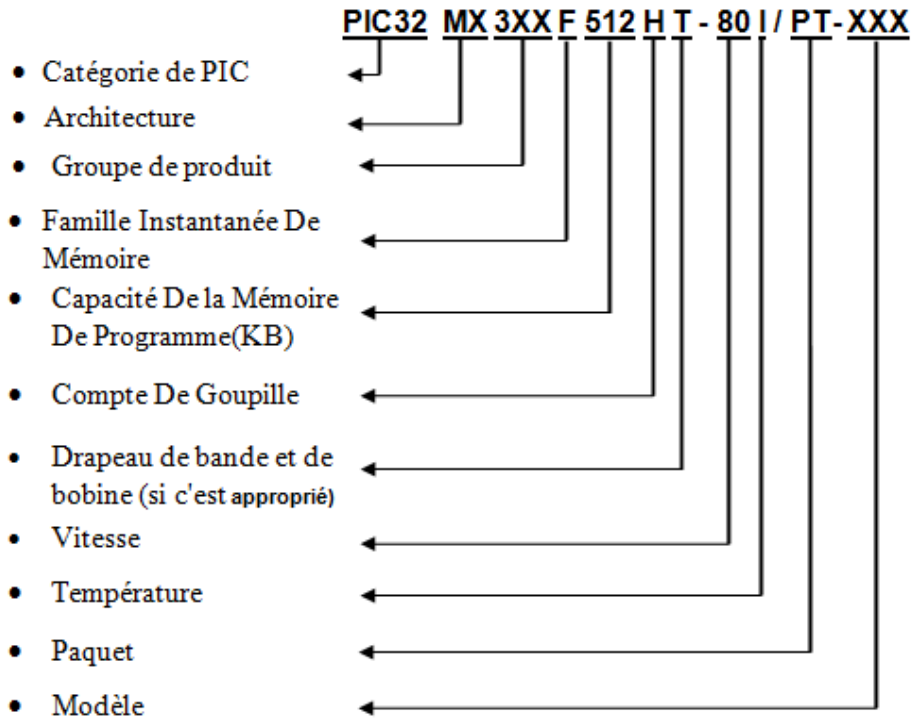
Les PICs, au même titre que les microprocesseurs, sont composés essentiellement de registres ayant chacun une fonction bien définie. Les PICs possèdent également des périphériques intégrés, tels qu'une mémoire EEPROM (Electrically-Erasable Programmable Read-Only Memory), un timer, des ports d'entrées/ sorties ou bien encore un convertisseur analogique/numérique [11].

Selon le type de PIC utilisé, on retrouvera en interne un certain nombre de registres et périphériques possédant des caractéristiques différentes. Les différences de caractéristiques selon le PIC utilisé sont :

- La taille de la RAM (Random Access Memory) interne.
- La mémoire EEPROM intégrée.
- Le type de mémoire programme : FLASH, EPROM ou OTP (One-Time Programmable) et la taille de celle-ci.
- Le timer intégré.
- Les convertisseurs analogique/numérique intégrés.
- La vitesse en MIPS.

2.4. Identification d'un PIC

Pour identifier un PIC, nous utilisons simplement son numéro : les 2 premiers chiffres indiquent la catégorie du PIC, et chaque PIC à son référence structure comme l'exemple suivant :



3. PIC32

3.1. Définition

Le MicroChip PIC32 est une famille de microcontrôleurs complexes, Il est basé sur le cœur MIPS32 M4K (Architecture MIPS « Microprocessor with out Inter locked Pipeline Stages »), Ce microcontrôleur offre de nombreux périphériques utiles pour les systèmes mécatroniques, des E/S numériques, la communication série synchrone et asynchrone et modulation d'impulsions, etc. Pour nos fins, les principaux d'avantages du Pic 32 bits par rapport aux PIC à 8 bits sont : [10]

- La rapidité (vitesse d'horloge maximale de 80 Mhz contre 48 Mhz).
- Périphériques disponibles.

3.2. La raison de choix du PIC32

Notre choix s'est porté sur le pic32, car nous avons à notre porté la carte de développement CEREBOT 32MX4 de Digilent, dont une vue générale peut être donné par la Figure II.2

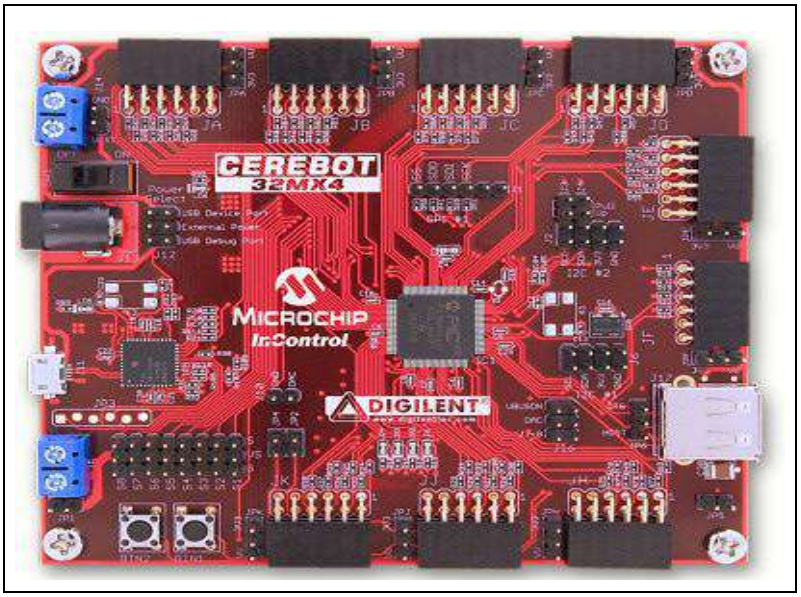


Figure II.2. *Microcontrôleur PIC32*

Plus précisément, le PIC sur lequel les tests ont été effectués est le PIC32. Il a été choisi principalement pour son grand nombre de registres, et la quantité de mémoire dynamique.

3.3. CEREBOT 32MX4

Le CEREBOT 32MX4 fonctionne avec l'environnement de développement MicroChip MPLAB.

3.4. Présentation de la carte CEREBOT 32MX4

La Figure II.3 illustre le diagramme synoptique du CEREBOT [12]

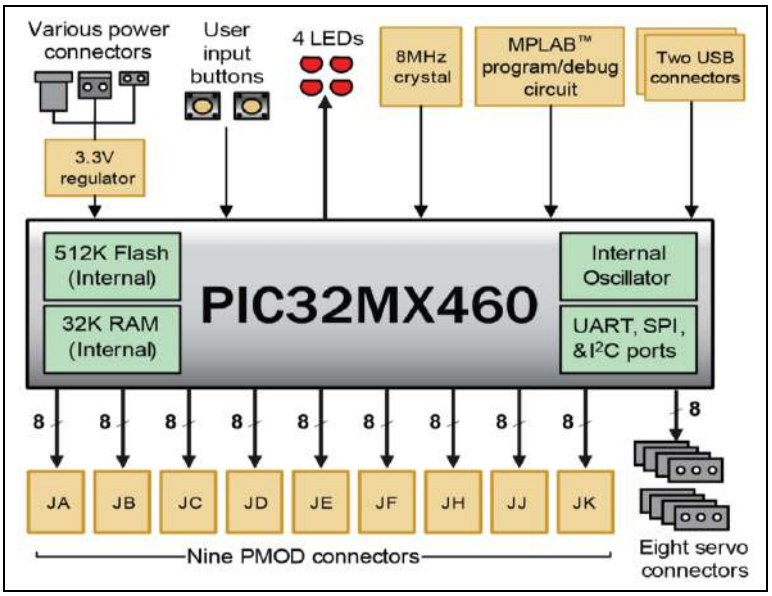


Figure II.3. *Architecture de circuit CEREBOT 32MX4*

3.5. Les dispositifs incluent

Un microcontrôleur de PIC32MX460F512L soutient la programmation dans l'environnement de développement de MPLAB de MicroChip.

- neuf connecteurs Pmod pour les panneaux périphériques du module de Digilent comprend ce CEREBOT 32MX4
- huit connecteurs par les servo moteurs.
- Entre USB.
- deux boutons poussoirs.
- quatre LEDs.
- options multiples d'alimentation, y compris l'USB (sélectionne le mode d'alimenter par le Jumper JP6 dans la carte CEREBOT).

3.6. Composition de CEREBOT 32MX4

Le CEREBOT est composé essentiellement des trois composants fondamentaux à savoir le microcontrôleur, les connecteurs Pmod est la source d'horloge.

3.6.1. Caractéristique

Les caractéristiques générales de PIC32MX460F512L sont :

- 512KB RAM de type flash.
- Mémoire interne d'une taille de 32KB SRAM.
- 2 interfaces périphériques séries.
- Canal d'accès direct à la mémoire.
- deux interfaces périphériques périodiques.
- cinq timer/compteurs de 16 bits.
- cinq sorties de PWM.
- deux comparateurs analogiques.

3.6.2. Connecteur PMOD

Le CEREBOT 32MX4 a neuf connecteurs Pmod pour relier des modules de périphérique Pmod de Digilent. Il y a deux modèles de connecteur de Pmod à six-pins et a douze-pins présenté dans la Figure II.4.

- Les connecteurs de six-pins utilisent la configuration 1x6, fournissant quatre signaux d'E/S, messe et un raccordement de puissance permutable.
- Les connecteurs de douze-pins de configuration 2x6, fournissant huit signaux d'E/S, deux bornes de puissance et deux terre pins.

Le raccordement de l'alimentation peut être permutable entre la source régulée de 3.3v de la carte et la source non régulée, le choix se fait selon la position d'un jumper se retrouvent à cote de chaque Pmod. [12]

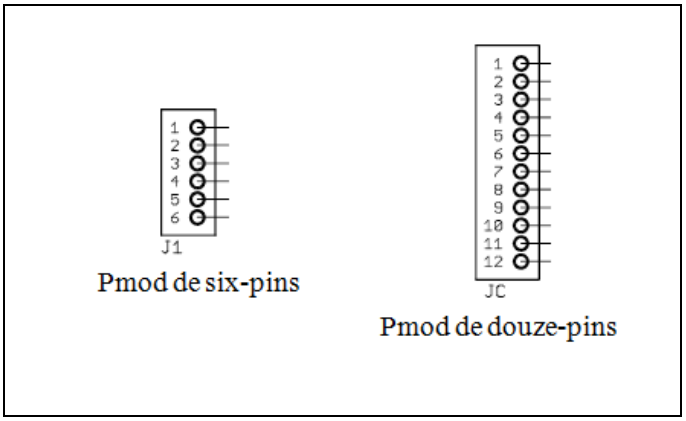


Figure II.4. *Les deux types de Pmod de CEREBOT 32MX4*

3.6.3. Source d'horloge d'unité centrale de traitement

Le microcontrôleur PIC32 supporte plusieurs sources d'horloge pour le fonctionnement du processeur principal, Le CEREBOT 32MX4 peut utiliser différents option d'oscillateur, configure par le pic, de façon de choisirai le système qui veut être convient notre besoin, ces déférent type sont d'utiles avec l'option d'oscillateur de l'EC (External Clock), ou d'utilise avec l'option d'oscillateur de XT, standard source utilise 8MHZ comme la fréquence utilise si choisir l'option d'oscillateur de l'EC.

En utilisant le système interne d'horloge PLL« Phase-Locked Loop », il est possible de choisir des multiples ou divisions de 8Mhz.

3.7. Utilisation de CEREBOT 32MX4

Pour changer une application au CEREBOT 32MX4, on doit tout d'abord préparer la carte, ensuite chargé le programme afin de faire une configuration adapté.

3.7.1. Préparation de la carte

- Branchement d'alimentation.
- Branchement d'USB pour charger notre programme.
- Installation de jumper à partir des dispositifs utilisés.

3.7.2. Chargement

Le panneau de CEREBOT 32MX4 est prévu pour être utilisé avec MPLAB-IDE, Le support de développement choisi est le langage C, qui identifie les ports et les connecteurs, et simule le fonctionnement de la carte.

Pour mise au point avec le starter de kit de PIC32, on choisit «PIC32MX Starter Kit» [12]

Apres la construction de notre projet on charge le programme dans la carte et on observe le résultat, pour voir plus claire, on va présenter un petit exemple. Notre démonstration utilise un programme simple qui gère l'allumage d'une LED de notre carte. Ce travail est basé sur deux étapes :

3.7.3.1. Préparation d'espace de travail

Au début, on ouvre un nouveau projet dans le MPLAB IDE (Figure II.5)

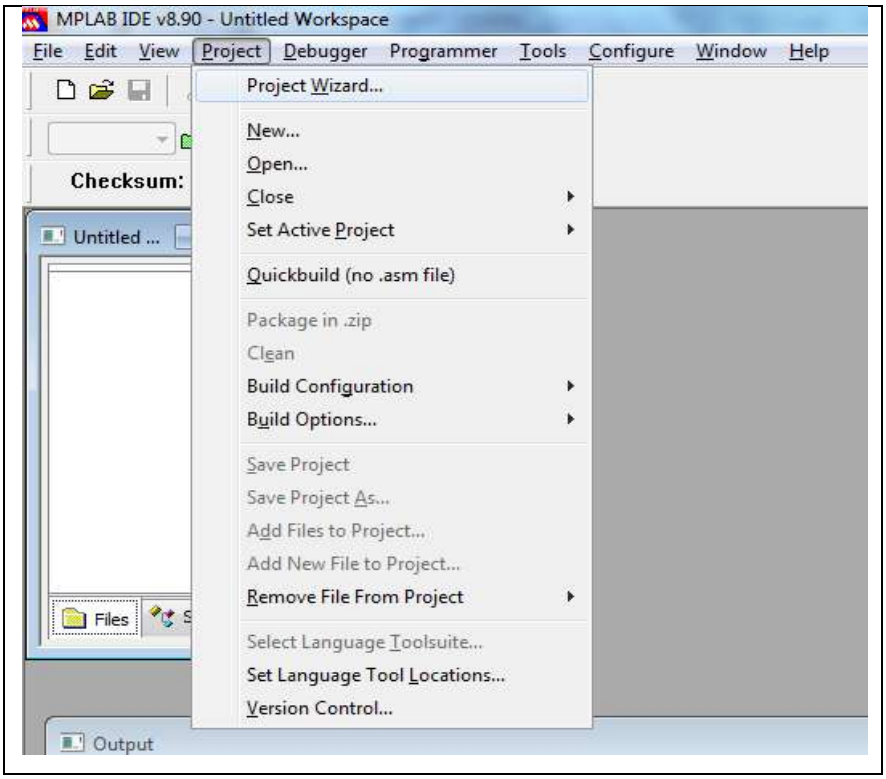


Figure II.5. *Création d'un nouveau projet*

Choisir la référence du PIC (Figure II.6).

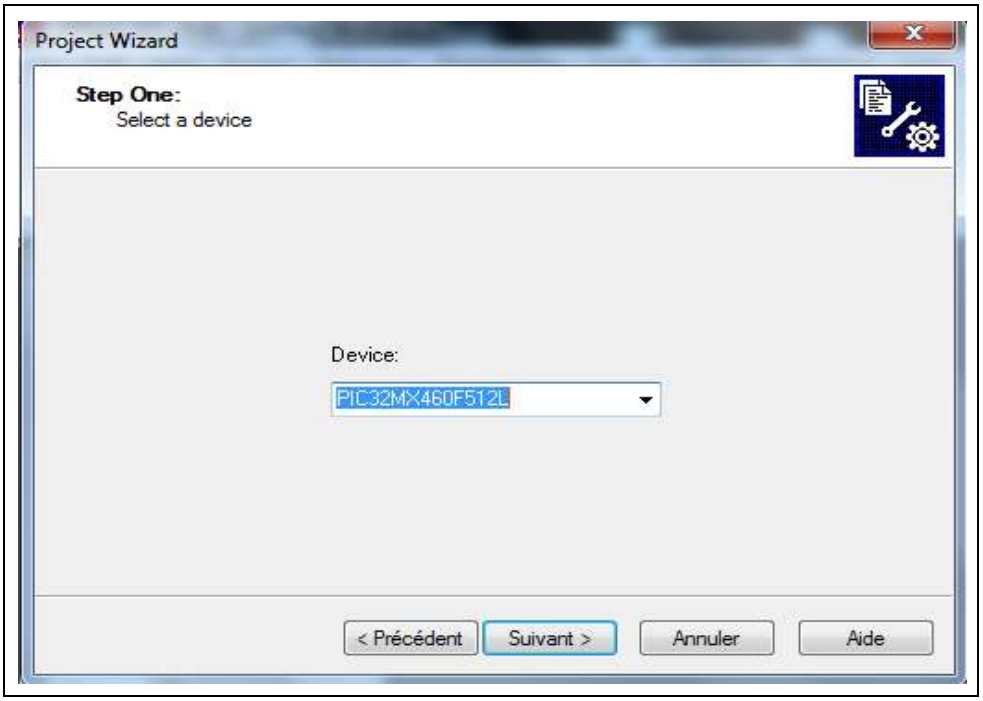


Figure II.6. *Le choix de référence de PIC*

Et enfin, choisir le compilateur du PIC32 bits, dans notre objet le c-compile qui a été préalablement installé. (Figure II.7)

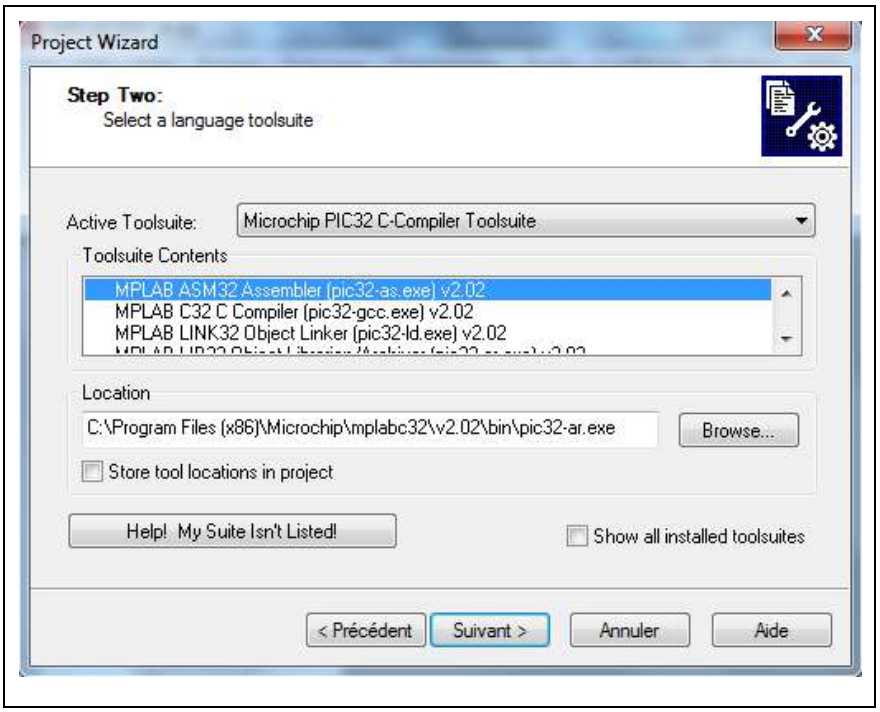
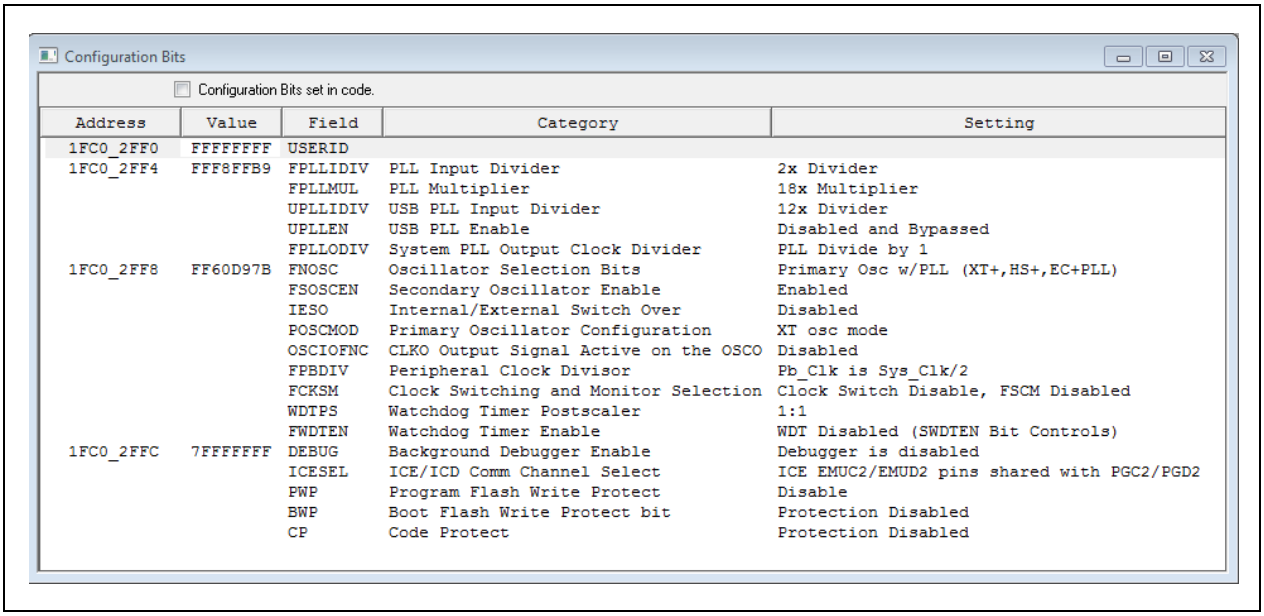


Figure II.7. *Choix dans MPLAB*

Enfin, une configuration du PIC est nécessaire par le MPLAB, on base sur cette configuration sur le PLL diviseur et le mode de temporisation utilise (EC, XT) (Figure II.8).



Address	Value	Field	Category	Setting
1FC0_2FF0	FFFFFFFF	USERID		
1FC0_2FF4	FFF8FFB9	FPLLIDIV	PLL Input Divider	2x Divider
		FPLLMUL	PLL Multiplier	18x Multiplier
		UPLLIDIV	USB PLL Input Divider	12x Divider
		UPLLEN	USB PLL Enable	Disabled and Bypassed
		FPLLODIV	System PLL Output Clock Divider	PLL Divide by 1
1FC0_2FF8	FF60D97B	FNOSC	Oscillator Selection Bits	Primary Osc w/PLL (XT+,HS+,EC+PLL)
		FSOSCEN	Secondary Oscillator Enable	Enabled
		IESO	Internal/External Switch Over	Disabled
		POSCMOD	Primary Oscillator Configuration	XT osc mode
		OSCIOFNC	CLKO Output Signal Active on the OSCO	Disabled
		FPBDIV	Peripheral Clock Divisor	Pb_Clk is Sys_Clk/2
		FCKSM	Clock Switching and Monitor Selection	Clock Switch Disable, FSCM Disabled
		WDTPS	Watchdog Timer Postscaler	1:1
		FWDTEN	Watchdog Timer Enable	WDT Disabled (SWDTEN Bit Controls)
1FC0_2FFC	7FFFFFFF	DEBUG	Background Debugger Enable	Debugger is disabled
		ICESEL	ICE/ICD Comm Channel Select	ICE EMUC2/EMUD2 pins shared with PGC2/PGD2
		PWP	Program Flash Write Protect	Disable
		BWP	Boot Flash Write Protect bit	Protection Disabled
		CP	Code Protect	Protection Disabled

Figure II.8. *Fenêtre de configuration bits*

3.7.3.2. Construction de programme

Dans l'éditeur de programmation, la construction doit être faite selon les champs suivants :

Bibliothèque

Les bibliothèques utilisées dans notre application sont :

- P32xxx.h : définition du µc utilisé, des constantes spécifiques au µc et des paramètres de configuration encodés lors de la programmation.
- Math.h: contient les opérations analogiques et arithmétiques ainsi que les fonctions mathématiques.

Les déclarations et initialisation des variables et les ports

Leur déclarations est sous forme de constantes et variables, les constantes peuvent être déclarées avec la commande « #define cons valeur » et tous les variable sont définis sous forme « type vari = valeur ».

Le mode d'utilisation d'un port peut être en entrée ou sortie.

TRIS nom_de_port_bit = 0 ou 1, 1 pour le mode entrée et 0 pour le mode sortie.

PORT nom_de_port_bit = valeur retenu (valeur à faire entrer un stocker).

Configuration du Timer1

D'un autre cote, la fréquence d'exécution trouve une partie nécessaire dans la programmation, pour contrôler le temps, on utilise un timer qui peut être configuré comme suit :

- TMR1 : un compteur de 16 bits
- T1CON : un registre de 16 bits pour le contrôle de l'activation et le mode d'opération de la temporisation (Tableau II.1).
- PR1 : peut-être utilisé pour produire une remise périodique de la temporisation

Tableau II.1. *Contrôle de registre de T1CON* [13]

adresse virtuelle	Nom		Bit **31/23/15/7**	Bit **30/22/14/6**	Bit **29/21/13/5**	Bit **28/20/12/4**	Bit **27/19/11/3**	Bit **26/18/10/2**	Bit **25/17/9/1**	Bit **24/16/8/0**
BF90_0 600	**T1CON**	31 :24	-	-	-	-	-	-	-	-
		23 :16	-	-	-	-	-	-	-	-
		15 :8	ON	FRZ	SIDL	TMWDIS	TMWP	-	-	-
		7 :0	TGATE	-	TCKPS<1:0>		-	TSYNC	TCS	-

Exemple

Pour utilise le TIMER1, on suive comme le tableau II.1, et initiales les défèrent paramètres décrits comme suivant :

- TON = 1 : active le mode de changement de fréquence
- TCKPS = 11 : le diviseur est placé à la valeur maximum (1:256)
- TGATE = 0, TSYNC = 0 : des fonctions de déclenchement et de synchronisation de l'entrée ne sont pas exigés, puisque nous employons l'horloge interne du MCU directement comme horloge de temporisateur.
- SIDL = 0 (défaut) : nous ne nous inquiétons pas du comportement en mode À VIDE

Dans ce cas, nous allons utiliser des techniques pour retarder la séquence de temporisation, pour obtenir cette valeur on doit utiliser la formule suivante [13]:

$$Temps_délai = Fpb \times 256 \times délai \qquad \text{(II.1)}$$

Le Fpb (peripheral bus clock frequency) supporté dans le pic est 80Mhz, pour obtenir un temps = 256ms on applique la formule II.2 pour calculer le délai.

$$délai = \frac{Temps_délai}{Fpb \times 256} = \frac{256 \times 10^{-3}}{0.125 \times 10^{-6} \times 256} = 8000 \qquad \text{(II. 2)}$$

Pour 256 ms, un comptage jusqu'à 8000, doit être fait.

Cet exemple peut être utilisé pour l'allumage des LEDs à deux séquences gérées par deux boutons.

Quand le bouton (Btn1) est activé, la séquence des 4 LEDs est allumée à une période de 256ms, d'un autre côté si le bouton (Btn2) est activé, la période est réduite à 128ms.

Le code correspondant est donné comme suit :

```
#include <p32xxxx.h>
#define DELAY1 8000 // 256 ms
#define DELAY2 4000 // 128ms
main()
{
TRISA=0x80;;  // configure le bit n°=7 de PORTA comme entrée
TRISB = 0; // configure le PORTB comme SORTIE
PORTA=0;
PORTB=0; // au debut tous les led sont éteint
T1CON = 0x8030; // TMR1 active, deviseur 1:256 horloge de Pb = 8 MHz
PR1 = 0xFFFF; // fixé le période de registre au maximum
// fonctionne pour horloge = 3.9 hz
while (PORTA == 0x40){ // Btn1 activé
TMR1= 0; // initialiser le compteur de TMR1
while ( TMR1 < DELAY1){}//compter jusqu'à le delai DELAY1
PORTB=0xff; // allumé tous les LEDs
TMR1=1;
while ( TMR1 < DELAY1){}//compter jusqu'à le delai DELAY1
PORTB=0; // éteindre tous les LEDs
}
// fonctionne pour horloge = 7.8 hz
while ( PORTA == 0x80){
TMR1 = 0;
while ( TMR1 < DELAY2){}//compter jusqu'a le delai DELAY2
PORTB =  0xff;
TMR1 = 0;
while ( TMR1 < DELAY2){}//compter jusqu'a le delai DELAY2
PORTB =  0;
}
}
```

4. Moteur à courant continu (CC)

4.1. Définition

Une machine à courant continu est une machine électrique. Il s'agit d'un convertisseur électromécanique permettant la conversion bidirectionnelle d'énergie entre une installation électrique par un courant continu et un dispositif mécanique, selon la source d'énergie [14].

4.2. Avantages

- Pas chère.
- Facile à utiliser (2 fils).
- Fonctionnée une tension continue.

- Peut atteindre des vitesses de rotation très élevé.

4.3. Inconvénients

- Peu de couple (car vitesse élevé, pour + de couple => motoréducteur).
- Relativement gourmand en courant.
- Les balais (ou charbons) finissent par s'user.
- Nécessite une boucle d'asservissement pour le contrôle de la vitesse.

5. Moteur pas à pas

5.1. Définition

Le moteur pas à pas est un convertisseur électromécanique qui assure la transformation d'un signal électrique impulsionnel en un déplacement mécanique (angulaire ou linaire). Sa structure de base se présente sous la forme de deux pièces séparées mécaniquement, le stator et le rotor. L'interaction électromagnétique entre ces deux parties s'assure la rotation [15].

5.2. Avantages

- Très précis.
- Beaucoup de couple.

5.3. Inconvénients

- Demande une électronique de contrôle avec un pont en H.
- Couple résiduel sans courant.
- Poids (moteurs lourd).
- Pas d'asservissement (il n'est pas possible de savoir si le moteur est bloqué).

5.4. Constitution

Les moteurs pas à pas sont constitués d'un rotor et d'un stator.

- Un stator composé d'une multitude de bobines alimentées par une tension continue
- D'un rotor

5.5. Les différents types des moteurs

Il existe plusieurs moteurs pas à pas :

- Les moteurs hybrides.
- Les moteurs à reluctance variable.
- Les moteur à aiment permanents à mode unipolaire ou bipolaire représentés dans la Figure II.9.

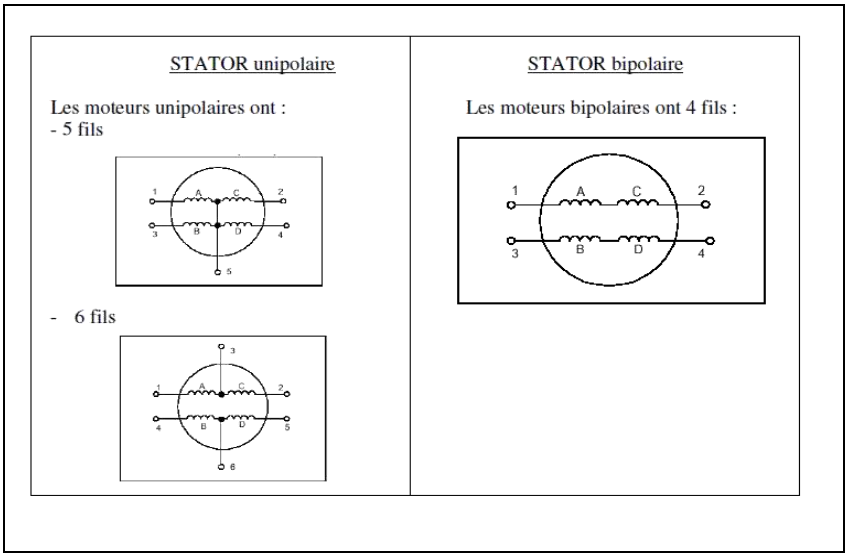


Figure II.9. *Mode de moteur pas à pas*

Dans notre choix, nous avons utilisé le moteur pas à pas au mode unipolaire, illustré dans la Figure II.10.

Afin d'inverser le sens du courant, les enroulements sont réalisés au moyen de deux fils dont l'une des extrémités est reliée au pôle + ou au pôle – de l'alimentation. La commande de ce type moteur est la plus simple de tous les moteurs pas à pas puisqu'il suffira d'alimenter les bobinages à tour de rôle pour faire tourner l'axe d'un pas.

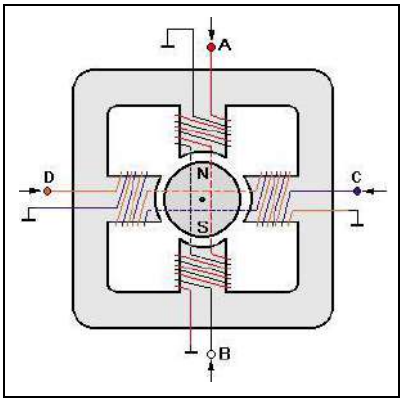


Figure II.10. *Représentation schématique d'un moteur unipolaire*

Comme pour chaque type de moteur, le modèle unipolaire peut être commandé en mode monophasé, biphasé ou demi-pas. Le Tableau II.2 donne l'ordre des différentes commandes.

Tableau II.2. *Les Différents commandes du moteur pas à pas unipolaire*

Mode monophasé	Mode biphasé	Mode demi-pas

A B C D A	AB BC CD DA AB	A AB B BC C CD D DA A

Pour inverse le sens, il suffit d'inverser la séquence.

6. Commande des moteurs

6.1 Moteur pas à pas

Pour utiliser un PIC, il est nécessaire d'ajouter un circuit de puissance entre le PIC et ce moteur, car le PIC donne des tensions et des courants faibles pour faire fonctionner les moteurs surtout à couple élevé.

Pour contrôler le moteur pas à pas ou a pensé à utiliser initialement deux ponts en H (Figure II.11)

Le principe de fonctionnement est décrit comme suite :

Bobine 1 :

- Un premier sens obtenu en saturant T1 et T4 et en bloquant T2 etT3.
- Un second sens correspond à T2 et T3 saturés et T1 et T4 bloqués.

Bobine 2 :

- Un troisième sens obtenu en saturant T5 et T8 et en bloquant T6 etT7.
- Un second sens correspond à T6 et T7 saturés et T5 et T8 bloqués.

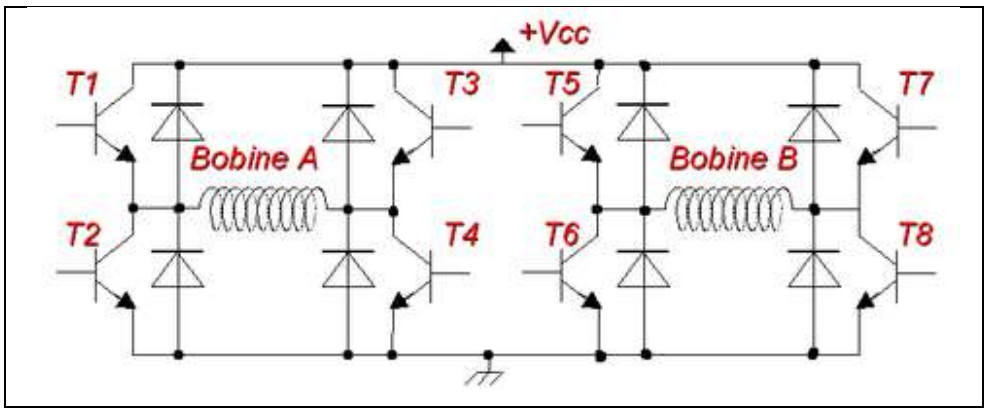


Figure II.11. *Montage en pont de H*

Avec T1, T2, T3 et T4 sont des signaux issus du PIC.

Cette solution a présenté des problèmes relatifs au courant faible au niveau des bobines de moteur, on a opté pour une seconde solution présentée par la Figure II.12.

Le principe de fonctionnement est décrit comme suite :

Bobine 1 :

- Un premier sens obtenu en saturant T1 et en bloquant T2, T3 etT4.
- Un second sens correspond à T2 saturé et T1, T3 et T4 bloqués.

Bobine 2 :

- Un troisième sens obtenu en saturant T3et en bloquant T1, T2 etT4.
- Un second sens correspond à T4 saturé et T1, T2 et T3 bloqués.

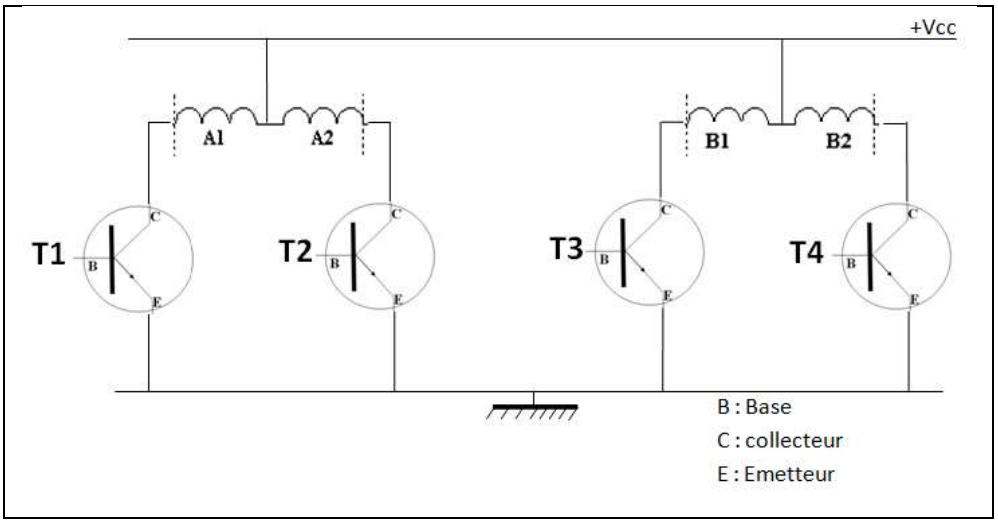


Figure II.12. *Montage en pont*

6.2. Moteur continu

Pour le contrôle de moteur à courant continu, on utilise un circuit de puissance à base transistor (Figure II.13).

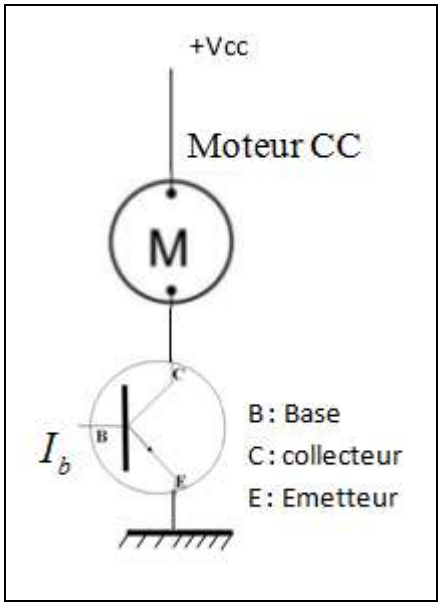


Figure II.13. *Montage adapté au moteur CC.*

Le moteur tourne quand si $I_b \neq 0$ sinon il est à l'arrête.

7. Conclusion

Dans ce chapitre, nous avons adapté l'environnement d'implémentation de l'algorithme de contrôle des moteurs autour d'un PIC 32 bits en utilisant la carte de développement CEREBOT 32MX4 de DIGILENT, nous avons également donné une étude théorique sur les moteur utilisé ainsi que leur circuit de commande.

CHAPITRE III : CONCEPTION ET REALISATION

1. Introduction

Après avoir abordé les outils et dispositifs nécessaires pour notre réalisation, nous traiterons dans ce chapitre les trois volets de notre application à savoir le volet programmation du PIC, circuit électronique et enfin la partie numérique du projet.

2. Cahier des charges

On veut faire le contrôle automatique d'un chariot dans une zone de charge, ce chariot qui ne se déplace qu'en marche avant doit se déplacer de n'importe qu'elle position à la position désirée ou la zone de charge est indiquer par la Figure III.1

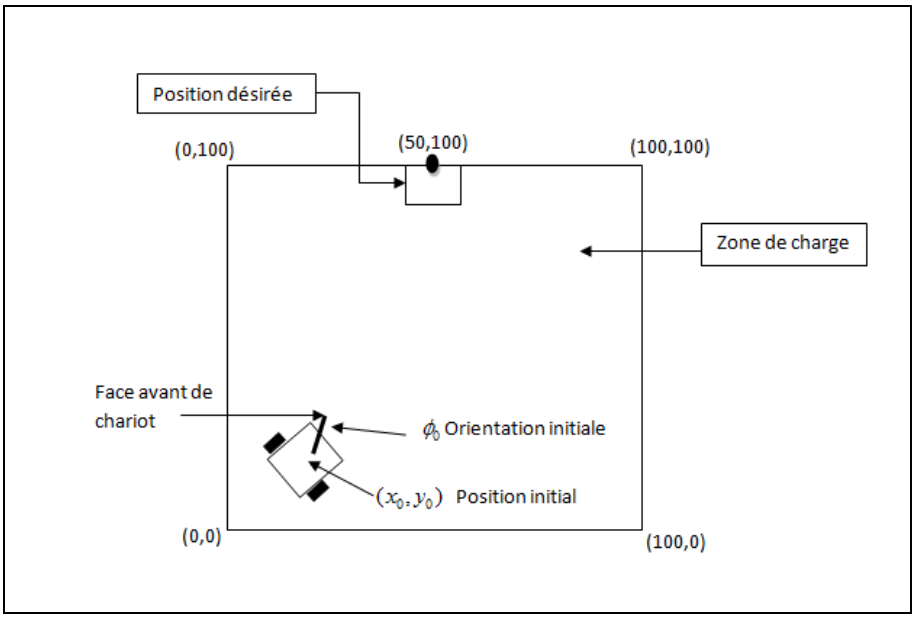


Figure III.1. *La zone de charge*

Le chariot peut se déplacer qu'en avant grâce à un moteur à courant continu installé dans les roues arrière.

L'orientation de la direction se fait grâce à un moteur pas à pas installé dans un mécanisme qui permet de faire tourner la roue d'avant de -30° à 30°.

La position du chariot est identifiée par ses coordonnées cartésiennes et son orientation initiale. Ces explications illustrées pour la Figure III.2

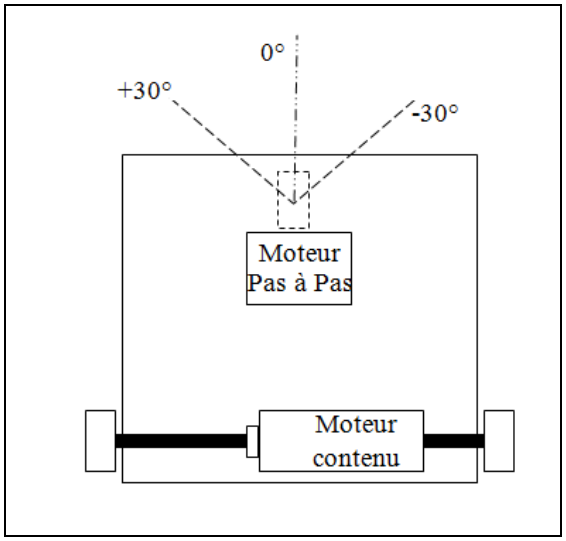


Figure III.2. *Conception primaire de chariot*

Pour le contrôle de ce système, on se propose d'utiliser un contrôleur basé sur la technique neuro-floue (Figure III.3).

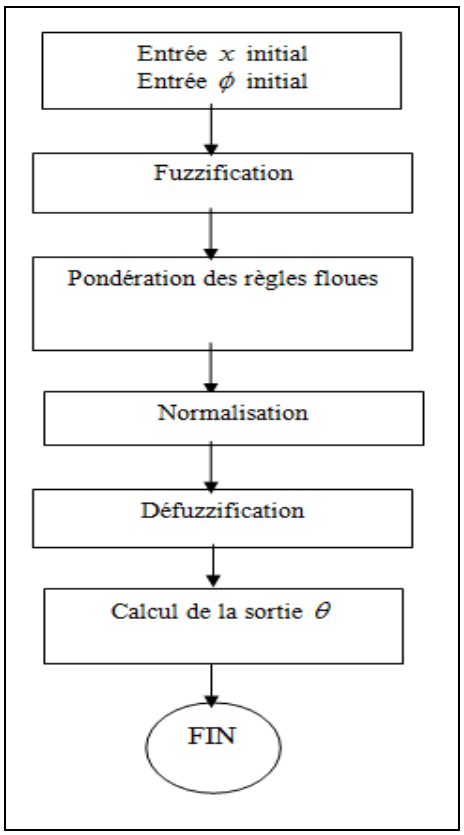


Figure III.3. *Diagramme du contrôleur*

Pour utiliser cette technique, on doit utiliser une base de données des entrées/sorties, cette base de données peut être générée à partir d'un contrôleur flou simple.

La raison d'implémenter un contrôleur neuro-flou au lieu d'implémenter un contrôleur flou est que ce dernier consomme beaucoup moins de ressources mémoire dans le PIC.

La Figure III.4, donné un schéma général et montrant les étapes de synthèse.

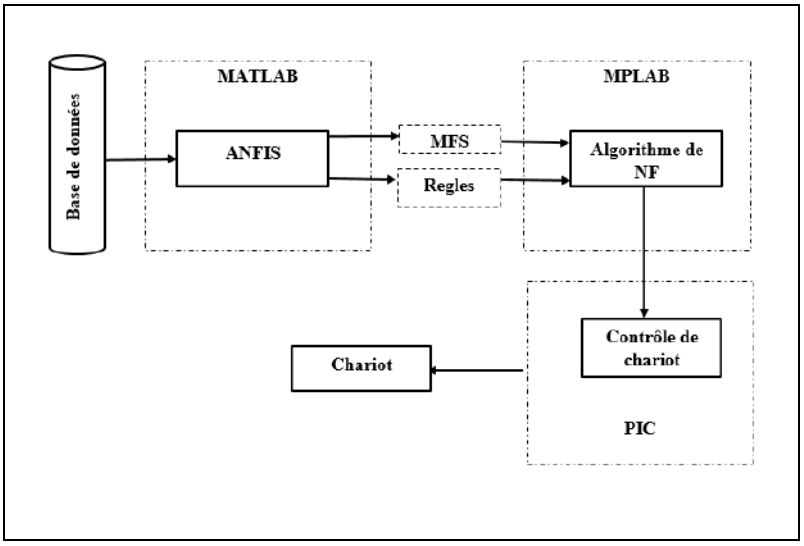


Figure III.4. *Organigramme du projet*

3. Synthèse du correcteur flou

La structure du contrôleur flou possède également deux entrées (x et ϕ) et une sortie θ.

3.1. Les fonctions d'appartenances

Les fonctions d'appartenance des variables x, ϕ et θ sont indiquées par la Figure III.5, Figure III.6 et Figure III.7 respectivement ou on remarque que x, ϕ et θ passe de 5,7 et 7 classes respectivement.

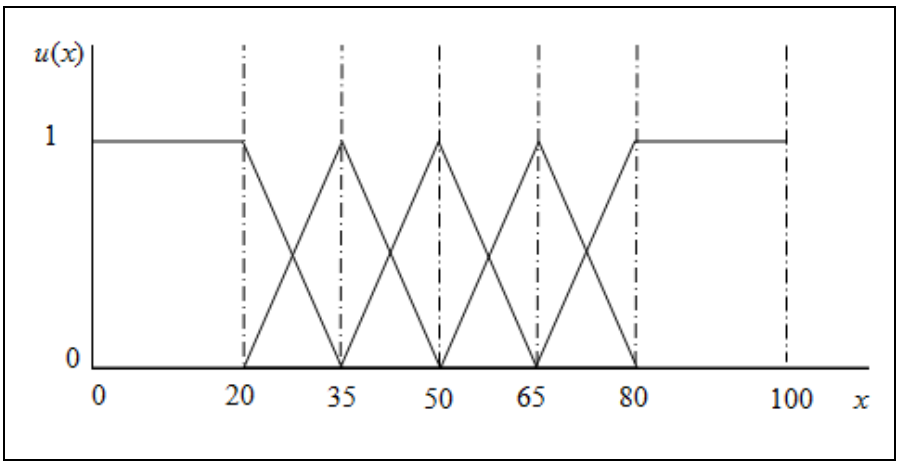


Figure III.5. L*es fonctions d'appartenance de* x

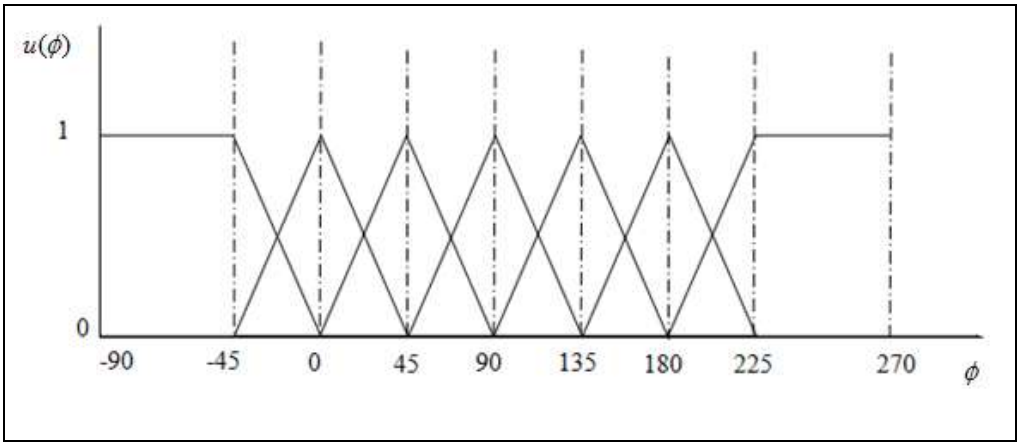


Figure III.6. *Les fonctions d'appartenance* ϕ

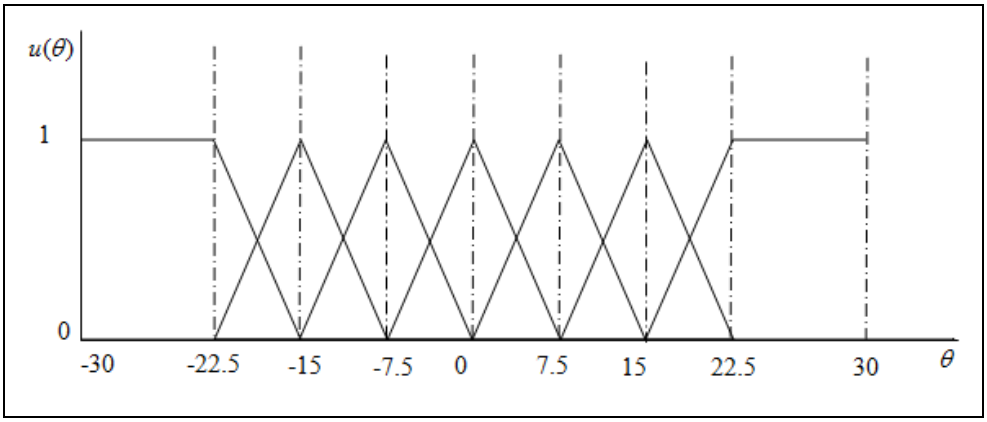


Figure III.7. *Les fonctions d'appartenance de sortie* θ

Où le Tableau III.1indique la nomination de chaque variable linguistique des E/S utilisés.

Tableau III.1. *Les variables linguistiques*

x = **[0 100]**		ϕ = **[-90 270]**		θ = **[-30 30]**	
x = [G, GC, C, DC, D]		ϕ = [DL, DP, DV, V, GV, GP, GL]		θ = [NG, NM, NP, Z, PP, PM, PG]	
G	Gauche	DL	Droite Loin	NG	Négatif Grand
GC	Gauche Centre	DP	Droite Prés	NM	Négatif Moyen
C	Centre	DV	Droite Verticale	NP	Négatif Petit
DC	Droite Centre	V	Verticale	Z	Environ de zéro
D	Droite	GV	Gauche Verticale	PP	Positif Petit
		GP	Gauche Prés	PM	Positif Moyen
		GL	Gauche Loin	PG	Positif Grand

3.2. Les règles floues

À partir de raisonnement humain logique, les règles floues indiquées par le Tableau III.2 sont utilisées pour le contrôleur flou.

Tableau III.2. *Les règles floues utilise pour logique flou*

x \ ϕ	DL	DP	DV	V	GV	GP	GL
G	PP	NP	NM	NG	NG	NG	NG
GC	PM	PP	NP	NM	NM	NG	NG
D	PM	PM	PP	Z	NP	NM	NM
DC	PG	PG	PM	PM	PP	NP	NM
C	PG	PG	PG	PM	PM	PP	NP

3.3. Test et validation des contrôleurs flous

Dans le but de valider, le contrôleur flou synthétisé, plusieurs tests ont été réalisés ou les résultats de simulation sont indiqués par les figures suivantes

- x = 15, ϕ =20, y =35 (Figure III.8).

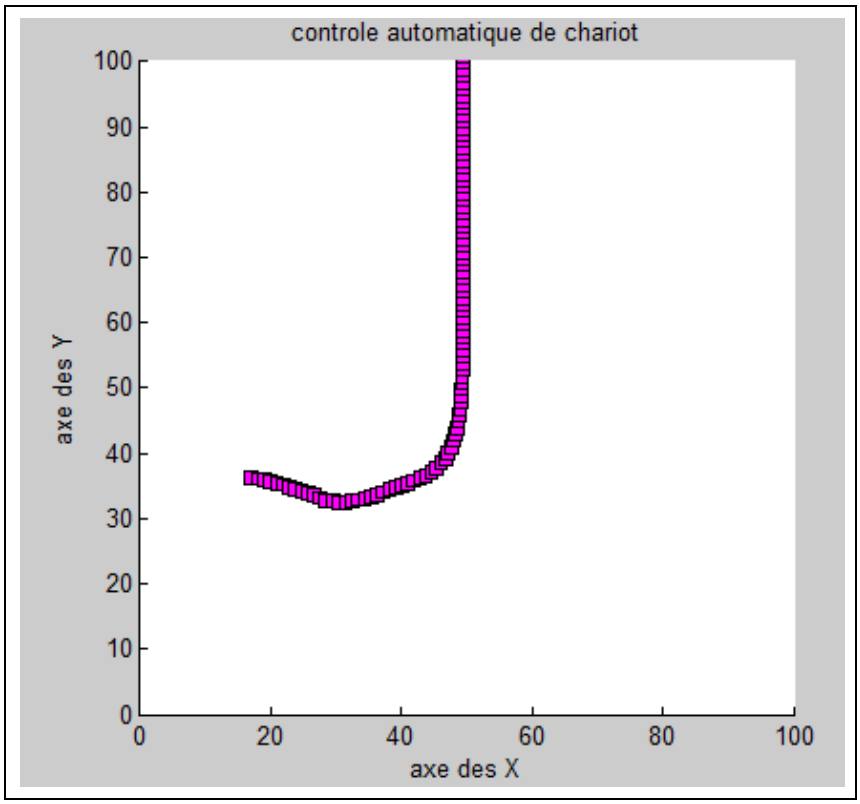


Figure III.8. *Exemple 1 : test de la simulation*

- x = 50, ϕ = 30, y = 60. (Figure III.9).

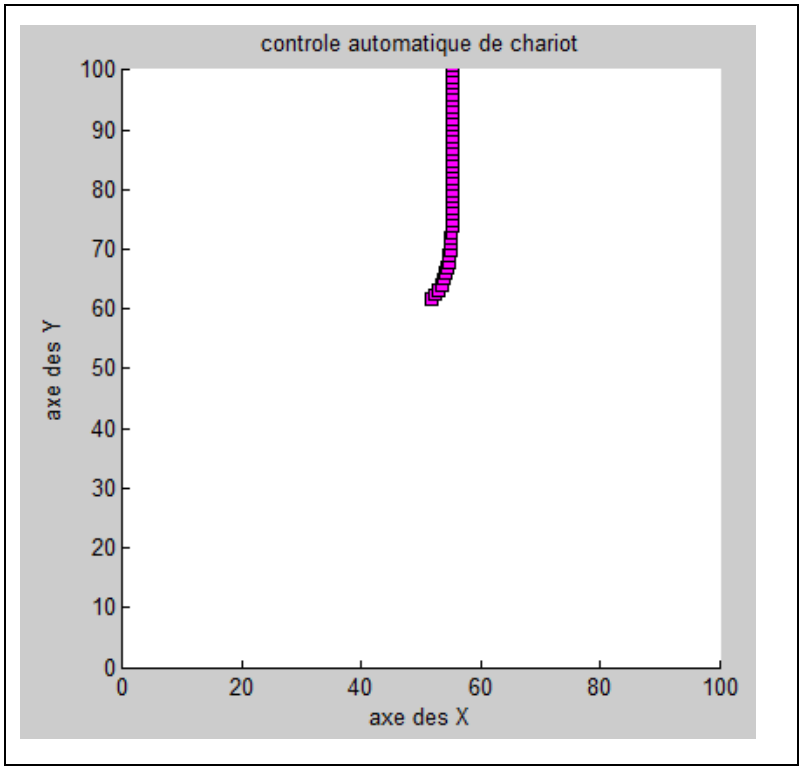


Figure III.9. *Exemple 2 : test de la simulation*

- x = 10, ϕ = -90, y = 15 (Figure III.10).

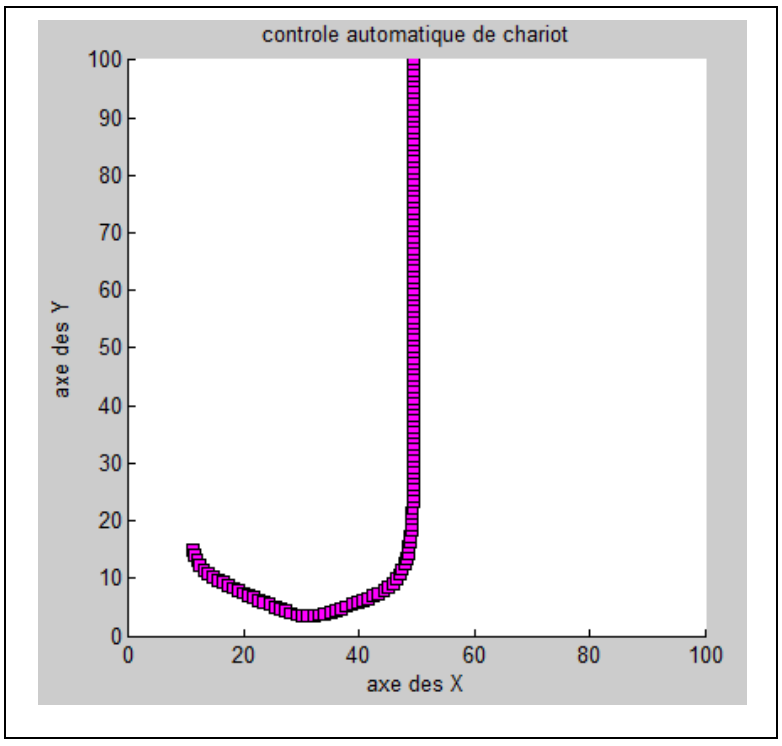


Figure III.10. *Exemple 3 : test de la simulation*

4. Synthèse du contrôleur neuro-flou

Les résultats de contrôleur flou sont utilisés pour l'apprentissage du système neuro flou. Comme on l'a déjà vu dans le chapitre I, on donne le système NF, les données, les types ainsi que le nombre de fonction d'appartenance des entrées et de sortie.

Le résultat d'apprentissage donne les paramètres aussi que règle de contrôleur.

Plusieurs tâtonnements ont été réalisés, dont ils sont résumes dans le Tableau III.3.

Tableau III.3. *L'évolution de l'erreur au cours de l'apprentissage de la fonction*

Fonction	Trimf	Trapmf	gbellmf	gaussmf	gauss2mf
MFs=2 3	3.5747	2.4996	2.2952	2.7220	2.4715
MFs=3 4	1.9658	2.1110	1.8045	1.8882	2.0461
MFs=4 4	1.6485	1.9401	1.8045	1.7504	1.9264
MFs=4 5	1.5629	1.7876	1.6646	1.6136	1.7803
MFs=5 5	**1.3917**	1.6286	1.5736	1.5425	1.6181

Nous concluons à partir du Tableau III.3 la meilleure erreur obtenu de la fonction d'apprentissage Trimf et pour le nombre de MFs est « 5 5 » et pour la sortie de type linéaire.

- Le parcours suivant est basé sur la réalisation de NF, ce dernier et sous forme d'être connu par le pic qui est raisonné par des informations précédentes.
- Dernièrement on a utilisé le pic pour la réalisation de notre programme sur le plan réel, arrivant au chariot qui commence son fonctionnement à l'aide des moteurs et leur commande basés sur des informations déjà noté auparavant, ainsi, on fait de mécanisme qui nous a aidées pour nous donner une orientation performance.

5. Conception

5.1. Partie électronique

Après avoir réalisé le coté programmation on doit abordés le coté conception dont on a besoin d'utiliser les composants suivantes :

- Cerebot 32MX4.
- Cinq transistors BD177.
- Cinq résistances 120 Ω.
- Source de tension.

- Le chariot se compose de deux roues placé en arrière et une seule se trouve en avant, les roues de l'arrière marche par le moteur continu tandis que celle de l'avant marche par le moteur pas à pas.

- **Moteur continu :**

On utilise dans ce travail le moteur à courant continu 1v à 6v (type F130) (Annexe 2), le rôle de ce moteur est de déplacer notre chariot avec l'utilisation de circuit quand on va déjà vu au chapitre II, donc, ce circuit est relié à notre alimentation pour donner une tension et un courant précis, cet alimentation est utilisé pour contrôler la vitesse des roues, le montage est présenter dans la Figure III.11.

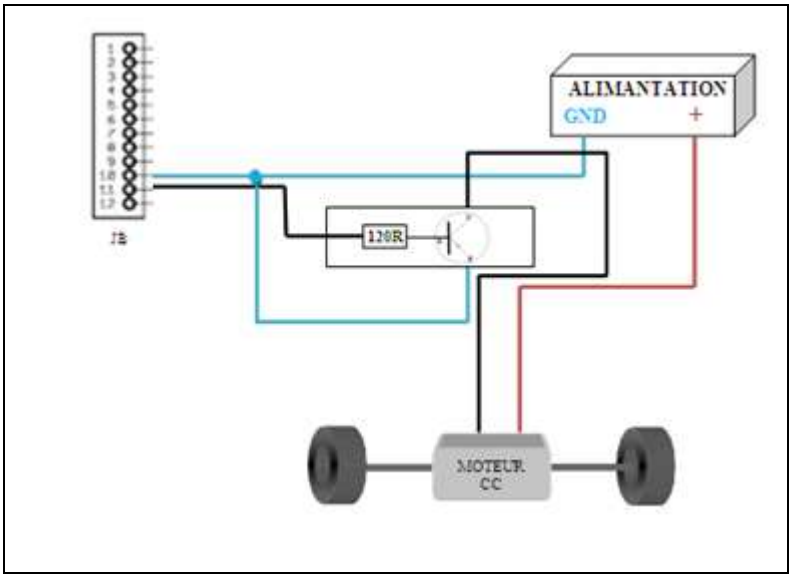


Figure III.11. *Architecture de montage moteur cc*

- **Moteur pas à pas**

On utilise dans ce travail le moteur pas à pas de référencé (PM35S-048) (Annexe 1), on doit faire un montage quand on vu dans le chapitre précédant pour orienter la roue de chariot, et plus précisément ces explications s'illustre dans la réalisation schématisé dans la Figure III.12.

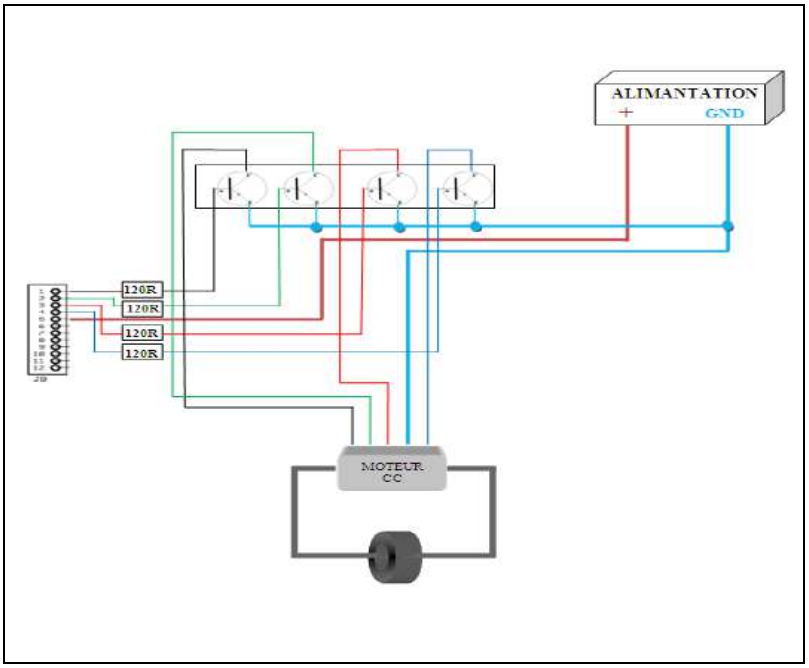


Figure III.12. *Architecture de montage moteur pas à pas*

- **Alimentation**

Nous avons utilisé l'alimentation de référence GwINSTEK GPS-2303, qui contient deux sources présenter dans la Figure III.13, une pour le moteur pas à pas, lui qui support au maximum 12v, et l'autre pour le moteur continu qui peut supporter jusqu'à 4.5v.

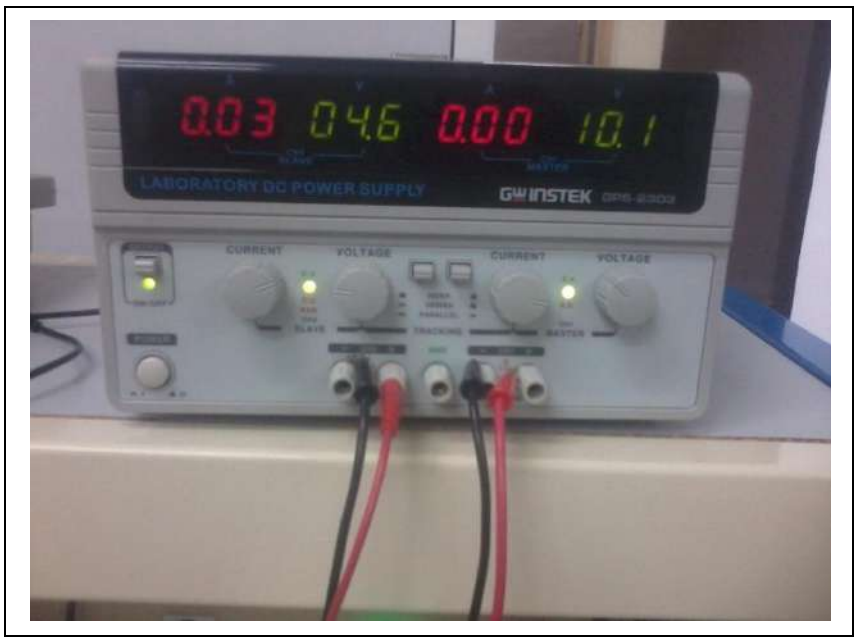


Figure III.13. *La source tension GwINSTEK GPS-2303*

- **CEREBOT 32MX4**

De côté programmation, la carte CEREBOT 32MX4 nous a fournis une tension pour déclencher les moteurs, cette dernière placée dans le circuit de commande utilisé. Après on a utilisé deux port pour donner les tensions conçu, le premier port on a utilisé cinq pins pour contrôler le moteur pas à pas, le deuxième port on a utilisé deux pins pour contrôler le moteur cc. Cette carte est présentée dans la Figure III.14.

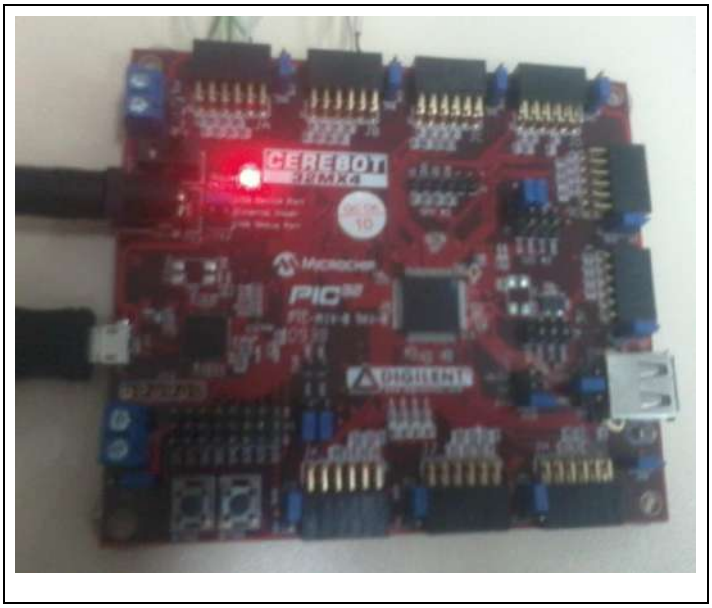


Figure III.14. *La carte CEREBOT utilise*

- **Circuit de commande**

C'est un circuit qui rassemble les transistors de référencé BD177 (Annexe 3) et les résistances 120 Ω. Son rôle est d'amplifié ou bien permet d'utiliser la tension d'alimentation à travers les résultats de la tension du PIC. Ce circuit est présent dans la Figure III.15

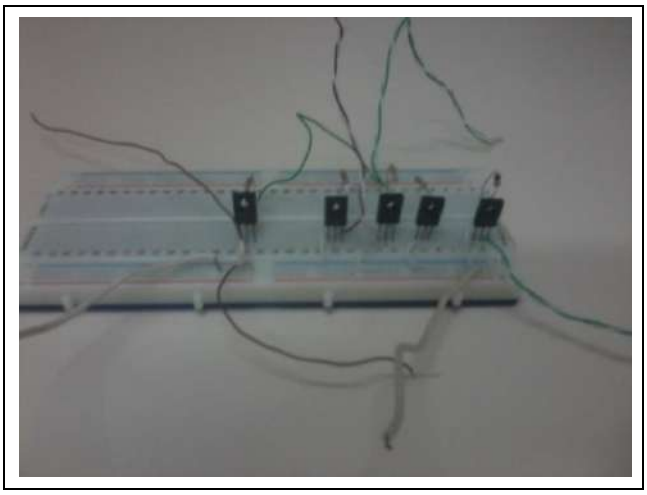
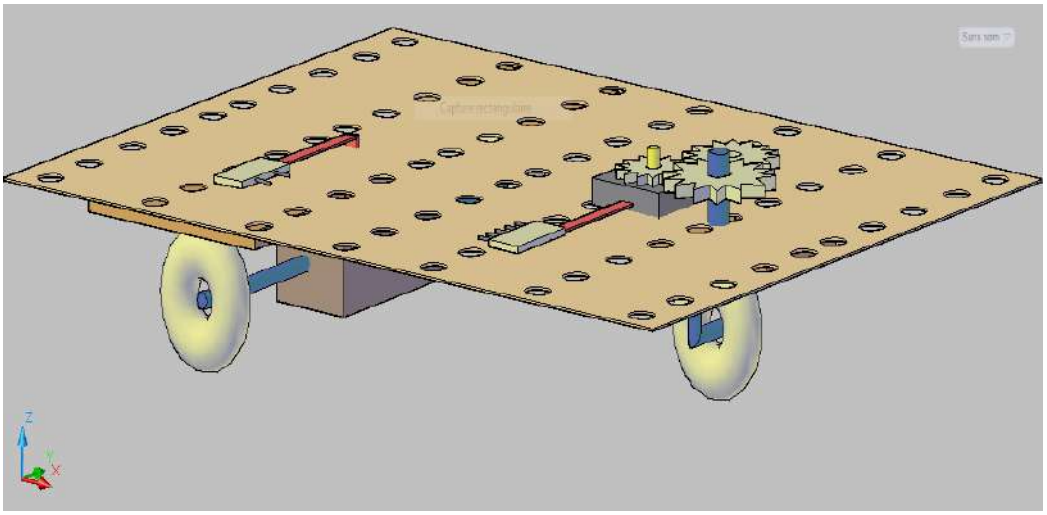


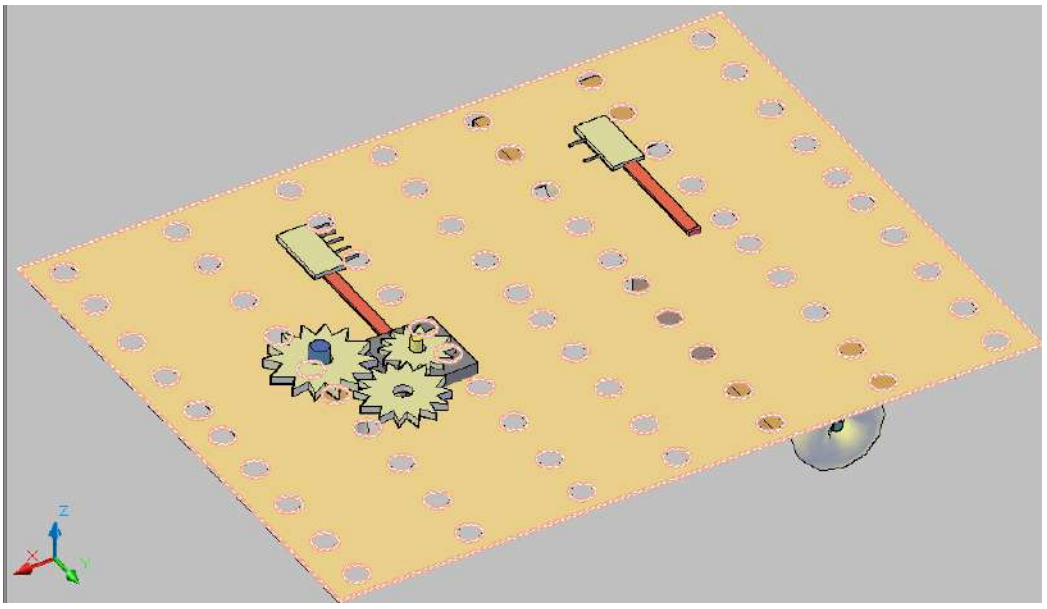
Figure III.15. *Circuit de commande*

5.2. Partie mécanique

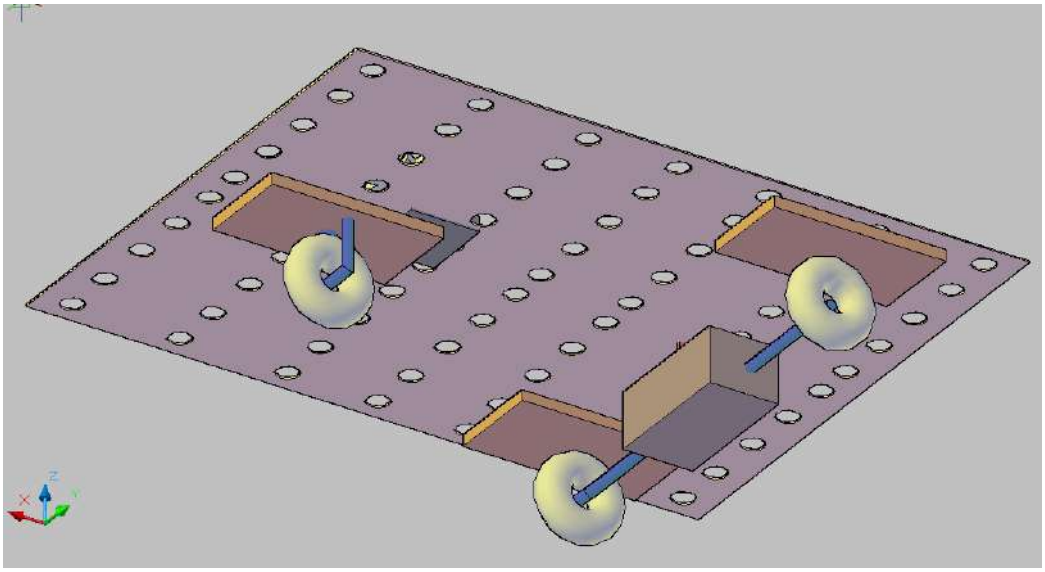
On passe maintenant à la côte mécanique, on a utilisé des engrenages afin d'augmenter la résolution de déplacement, c-à-dire augmenter le déplacement linéaire par rapport à la rotation du l'arbre du moteur. (Figure III.16)



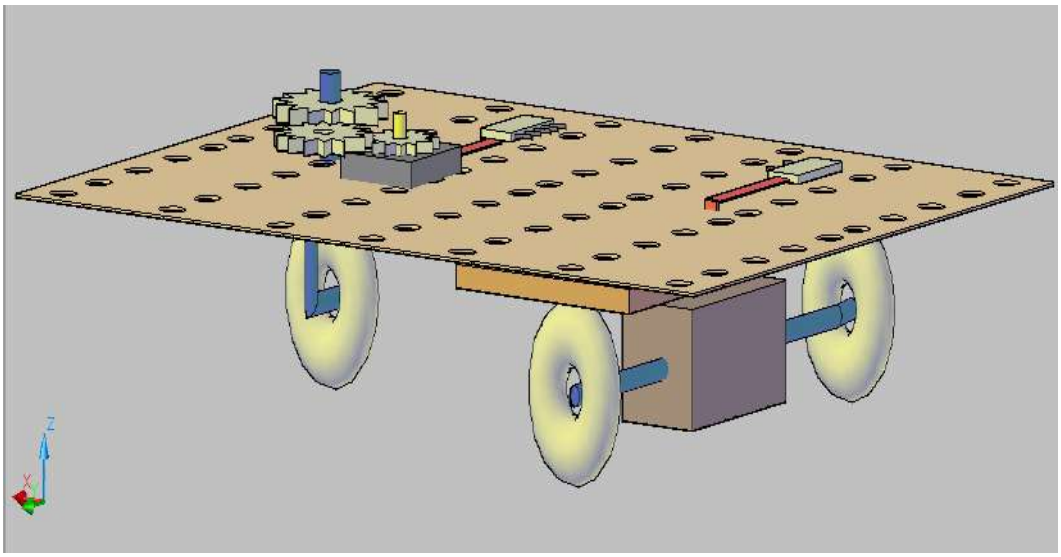
Vu du côté droit



Vu de l'haut



Vu du bas



Vu du cote gauche

Figure III.16. *Schéma synoptique du chariot*

Comme le chariot utilisé possède une roue avant de direction et un autre arrière pour le déplacement, il s'avère nécessaire de détailler les deux mécanismes.

- Mécanisme de la roue de direction

La Figure III.17 donne un schéma synoptique des engrenages utilisés pour la roue de direction.

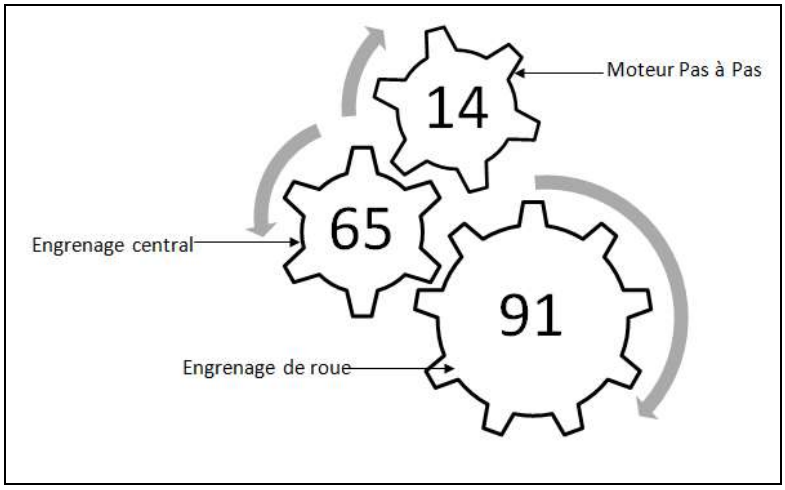


Figure III.17. *Mécanisme d'orienter la roue*

Le jeu des engrenages dans la Figure III.17 qui conçu des précisions varie à partir des formules suivantes

Dans le moteur, on a 48 pas à l'intérieur et 14 à l'extérieur comme suit :

$$48 \text{ pas (à intérieur de Moteur)} \longrightarrow 14 \text{ pas (à extérieur de Moteur)} \longrightarrow 360°$$

$$1 \text{ pas} \longrightarrow 7.5°$$

Entre l'engrenage de moteur et l'engrenage relié à l'arbre ce fonctionnent avec :

$$\left.\begin{array}{l} 65 \text{ pas} \longrightarrow 360° \\ 14 \text{ pas} \longrightarrow x\,? \end{array}\right\} \quad x = \frac{14 \times 360}{65} = 77.53°$$

Et enfin, entre l'engrenage de l'arbre et celui de la roue est donné par :

$$\left.\begin{array}{l} 91 \text{ pas} \longrightarrow 360° \\ 14 \text{ pas} \longrightarrow y\,? \end{array}\right\} \quad y = \frac{14 \times 360}{91} = 55.38°$$

La résolution finale peut être calculée comme suit :

Quand le premier engrenage de moteur tourne un tour complet, donc en marche 14 pas exactement, les autre aussi doit tourner même pas, mais le dégrée de tournage de chaque engrenage est :

Engrenage de moteur ⟶ engrenage central ⟶ engrenage de roue

$$14 \text{ pas} \longrightarrow 14 \text{ pas} \longrightarrow 14 \text{ pas}$$

$$360° \longrightarrow 77.53° \longrightarrow 55.38°$$

- Calcule un seul pas d'engrenage central

$$\left.\begin{array}{l} 360° \longrightarrow 77.53° \\ 7.5° \longrightarrow x\,? \end{array}\right\} \quad x = \frac{7.5 \times 77.53}{360} = 1.61°$$

- Calcule un seul pas de l'engrenage de roue

$$\left.\begin{array}{l} 360° \longrightarrow 55.38° \\ 7.5° \longrightarrow x? \end{array}\right\} \quad x = \frac{7.5 \times 55.38}{360} = 1.15°$$

Finalement, la résolution est 1.15°, c'est une valeur qui peut être obtenue en mode d'utilisation du moteur pas à pas en monophasé ou biphasé ;

En choisissant de travailler avec le mode demi-pas on obtient une résolution de 0.57°/pas.

- Mécanisme de la roue de déplacement

De la même manière que la roue de direction, les roues de déplacement sont placées avec les engrenages de la Figure III.18.

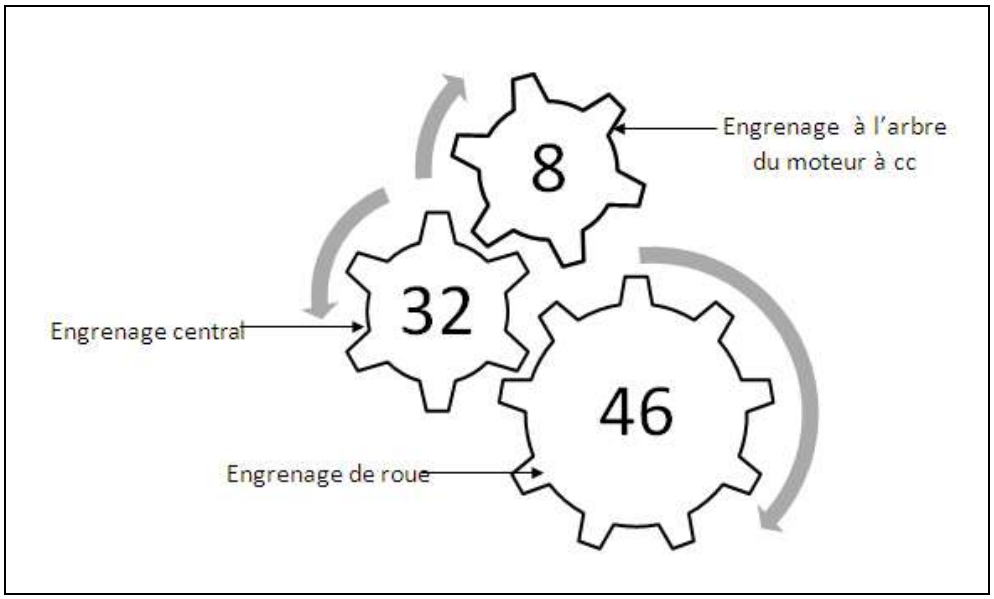


Figure III.18. *Mécanisme d'engrenage des roues d'arrières*

5.3. Partie contrôle

- **Moteur cc**

Pour contrôler le moteur de mode marche/arrêt, le PORTB de CEREBOT 32MX4 doit fournir une séquence de valeurs schématisé (Figure III.19).

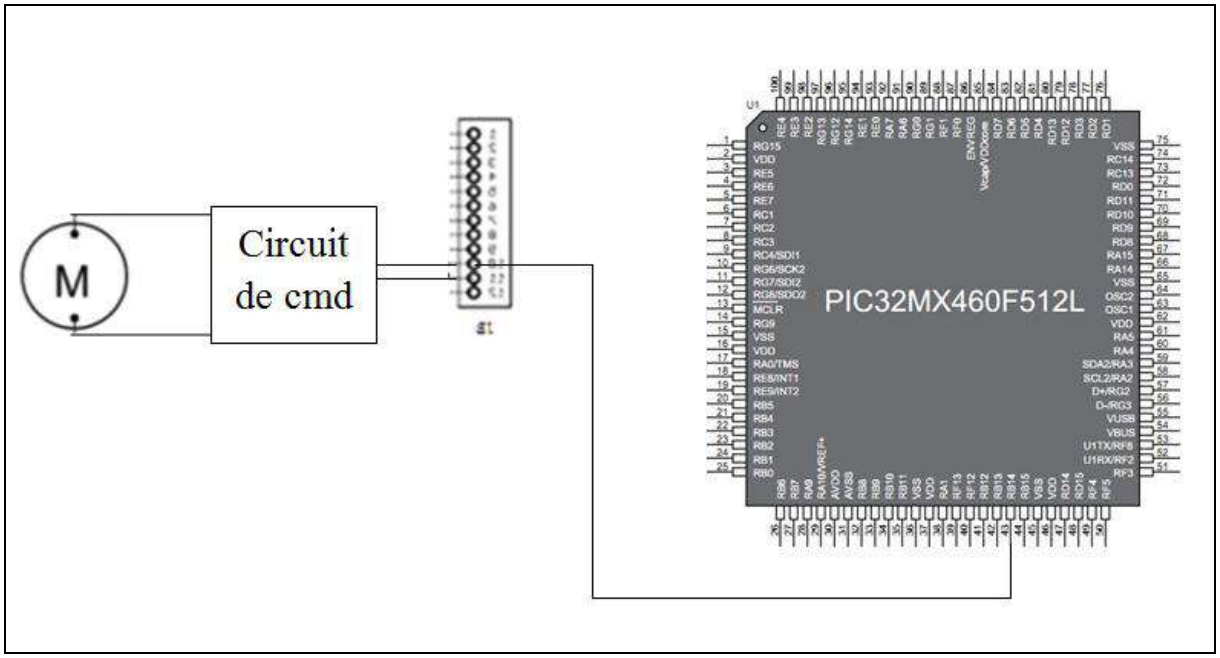


Figure III.19. *schéma pour contrôler le moteur cc*

Et initialisées les valeurs comme les instructions suivantes:

```
PORTB=0x4000;//Démarrer le moteur CC
PORTB=0; // arrêt du moteur CC
```

- **Moteur pas à pas**

L'orientation de la roue d'orientation est contrôlée à partir de séquences des valeurs, ces derniers donnés à travers le PIC dans le but de contrôler le moteur.

Le port utilisé pour fournir la séquence est le port G (PORTG) de CEREBOT, qui possède cinq sorties est comme suit :

- Les quatre premier fils sont des signaux pour commande le contrôle de moteur.
- Le cinquième fil est relié avec l'alimentation.

Ce monté avec la carte et le PIC32 comme le schéma suivant :

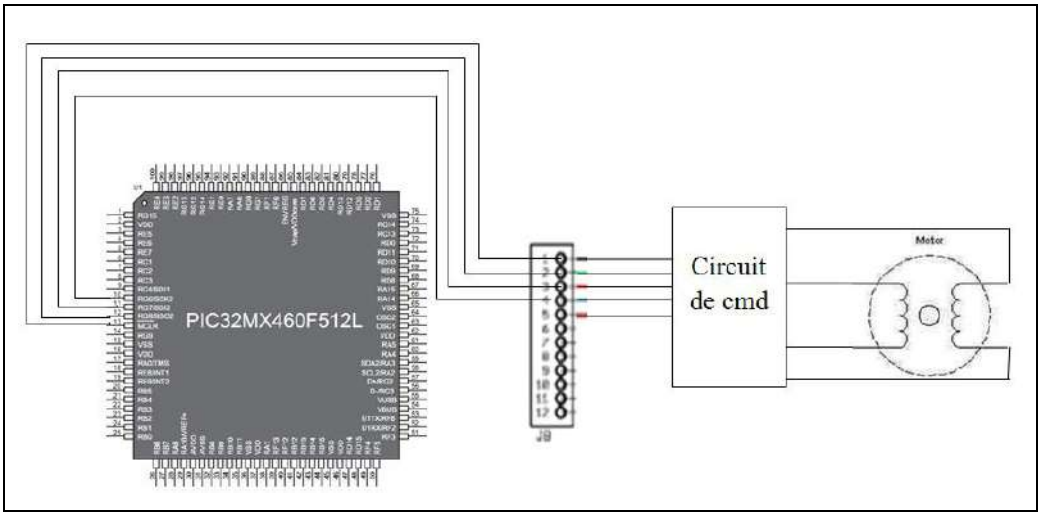


Figure III.20. *schéma pour contrôler le moteur cc*

Pour la commande, une séquence bien déterminée doit être donné. Cette séquence est stockée dans une matrice. Le programme suivant a été implémenté pour contrôler le moteur à travers un organigramme de calcul.

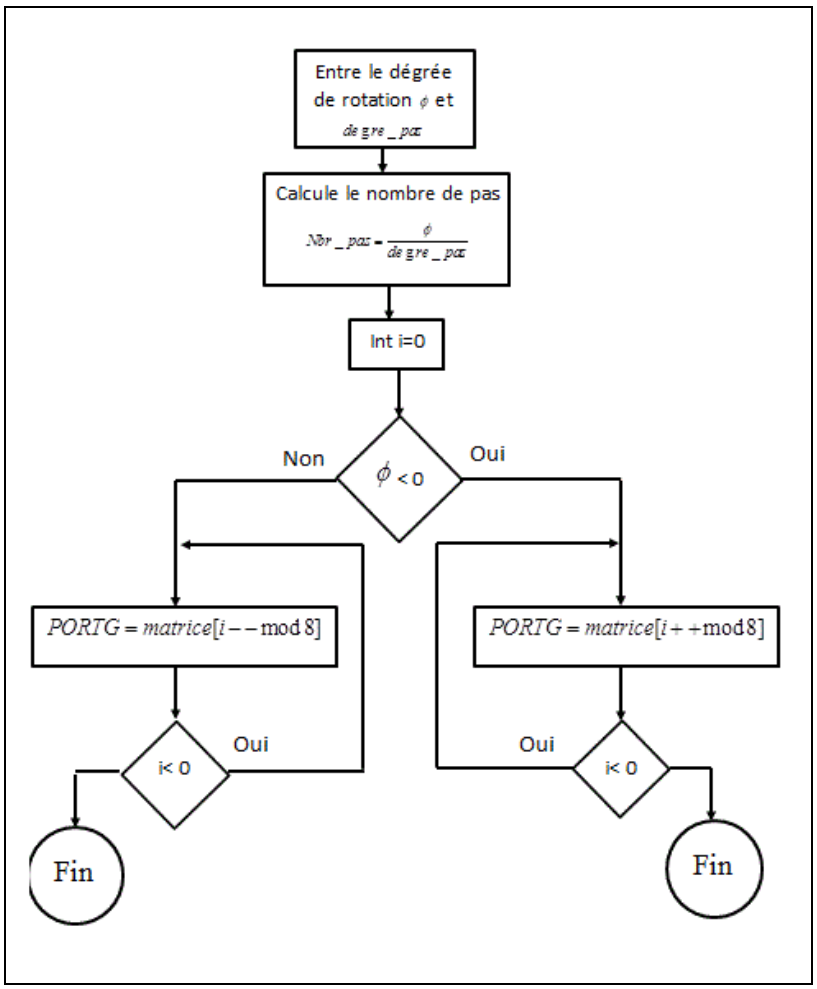


Figure III. 21. *Organigramme de fonctionne le moteur pas à pas*

Et leur programme écrit comme suit :

```
/*
    nous essayons orienté un moteur pa à pas a degrée = ORIENTATION°
    donc, pour faire ce fonctionnement, on a calcule le nombre de pas qui
    peur tourné le moteur pour obtient une orientation° exacte
*/
TRISG=0; // configure le port comme sortie
int matrice = [512,768,256,384,128,192,64,576];//les commande en dicimal pour controler le moteur
double orientation =x; // x est le degres qui donne pour orienter
//calcule le nombre de pas a partir le degree d'orientation
//degre_de_pas calcule a partir la précision obtient a le moteur et les engrenages utilises
int Nombre_de_pas = orientation / degrés_de_pas;
int i=0;
if(x<0){ //pour le sence direct
while (i<Nombre_de_pas) //fonctionne jusqu'a le Nombre de pas
    //pour lire juste les huit valeur de matrice ( i Mod 8 )
    if (i==8) i=0;
    PORTG = matrice[i]; //PORTG prend des valeurs de matrice a l'indice i
    i++; // Incrémente l'indice i
    }
/*
ici pour fonctionne a sence inverce, alors on lire a matrice de commande saise
les valeur de haut au bas, donc de l'indice 8 a le 1er
*/
else{ //pour le sence inverse
while (i>Nombre_de_pas) //fonctionne jusqu'a le Nombre de pas
    //pour lire juste les huit valeur de matrice ( i Mod -1 )
    if(i==-1) i=7;
    PORTE =matrice[i];//PORTG prend des valeurs de matrice a l'indice i
    i--; //décémente l'indice i
}
```

6. Réalisation

La Figure III.19 donné les étapes suivis de direction, pour le contrôle du moteur par la technique neuro-floue.

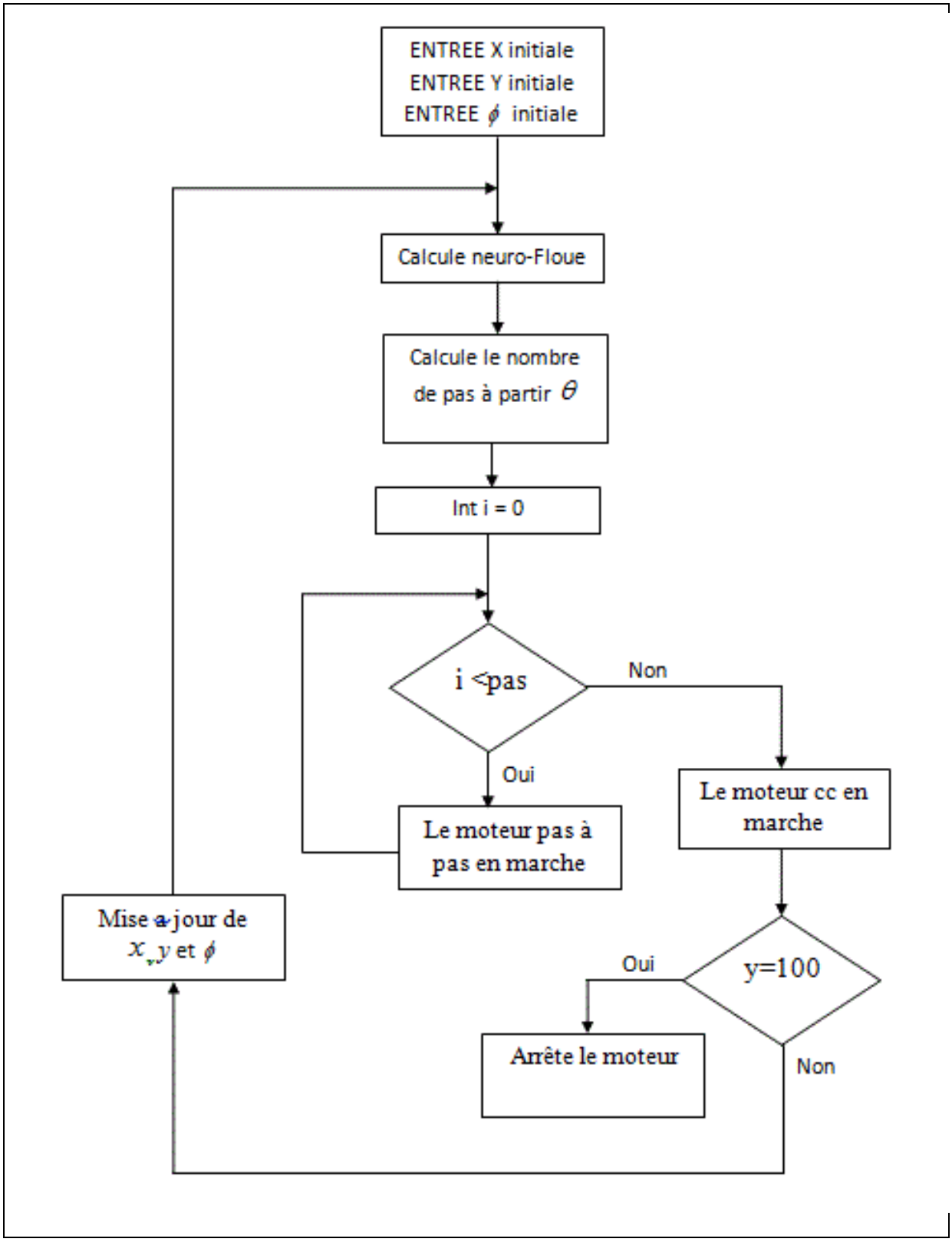


Figure III.22. *Diagramme de notre application*

Les Figure III.20 a III.22, donne des images sur des cas réels avec une comparaison avec les résultats de Matlab

- x = 50, ϕ = 90, y = 50. (Figure III.20).

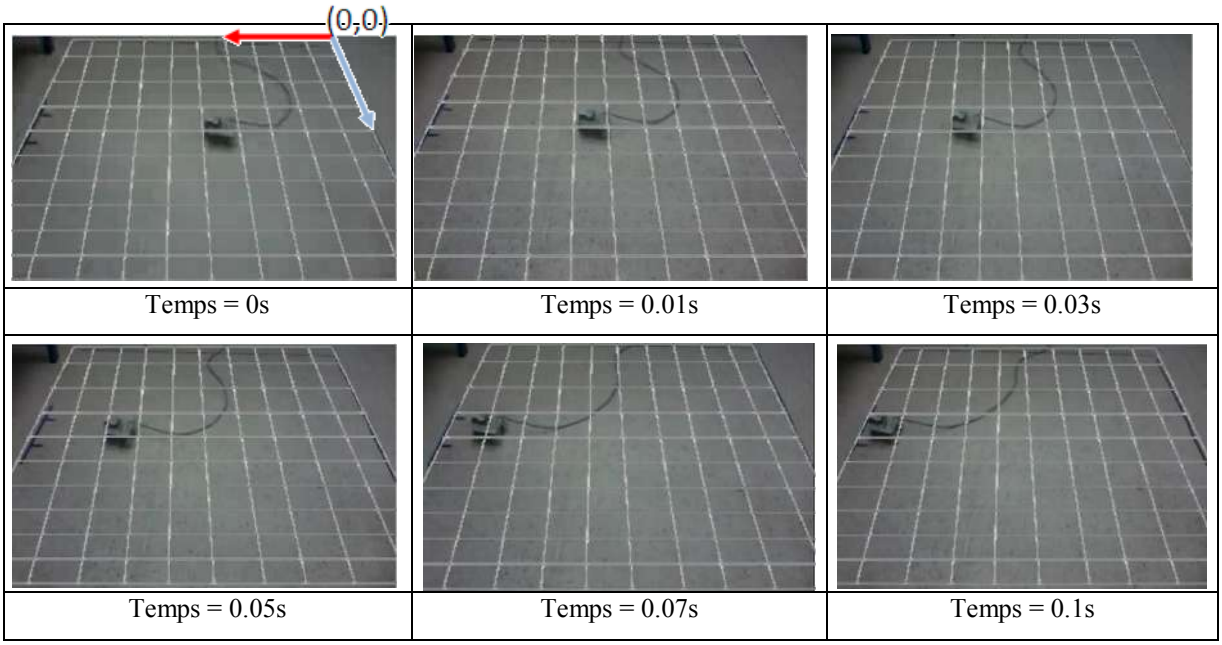


Figure III.23. *Exemple 1 : test de la réalisation*

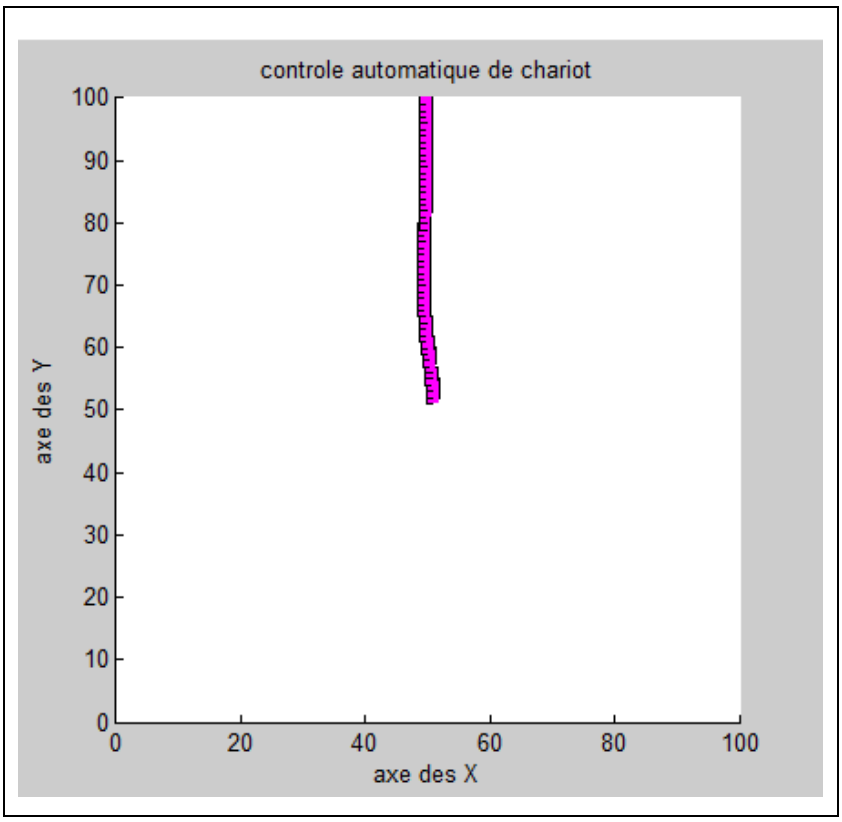


Figure III.24. *Résultat de l'exemple 1 dans Matlab*

- x = 60, ϕ = 135, y = 60. (Figure III.22).

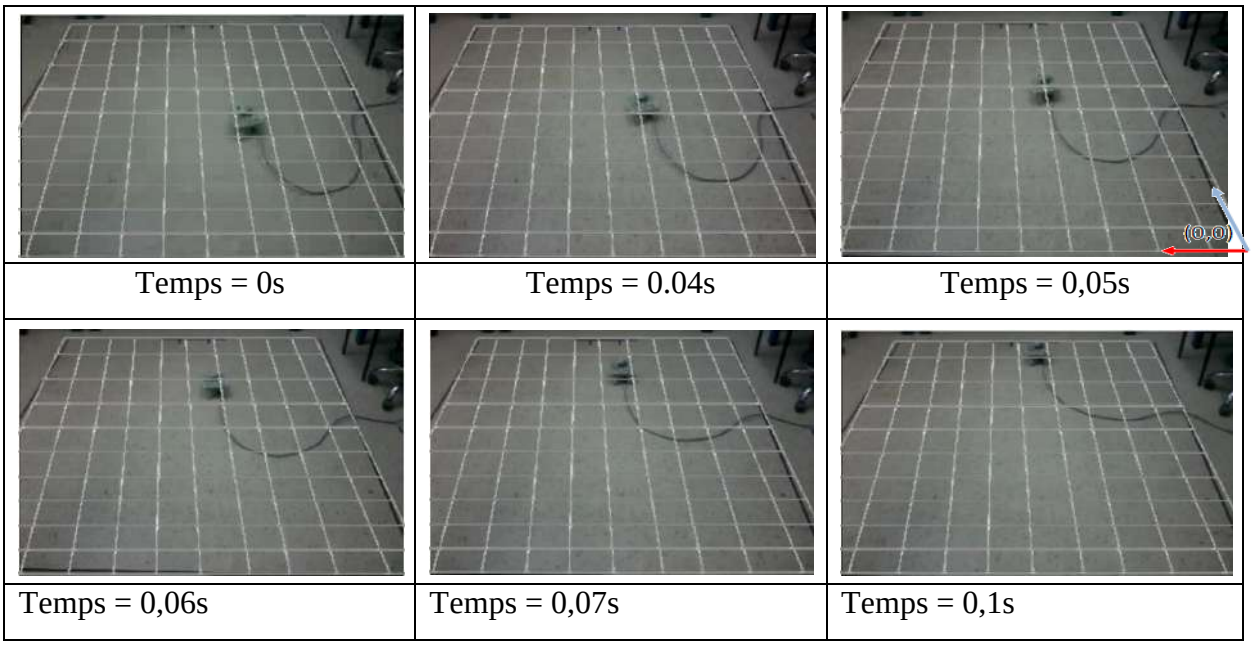


Figure III.25. *Exemple 2 : test de la réalisation*

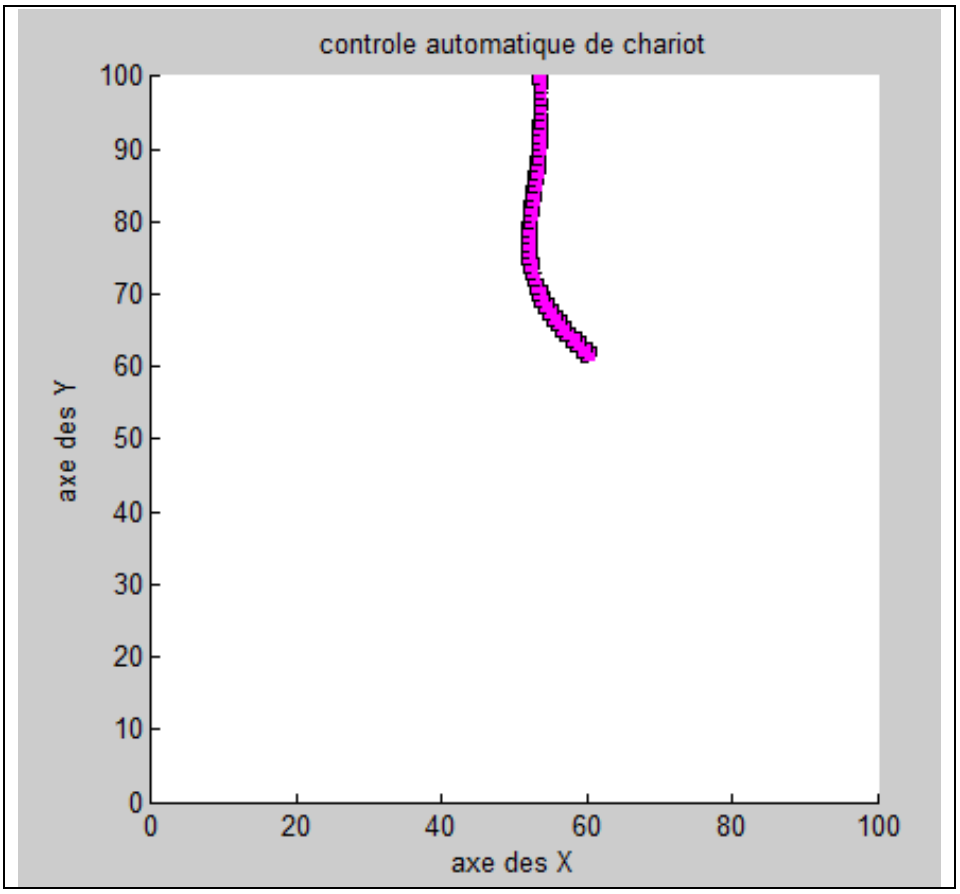


Figure III.26. *Résultat de l'exemple 2 dans Matlab*

- x = 85, ϕ = 180, y = 70. (Figure III.24).

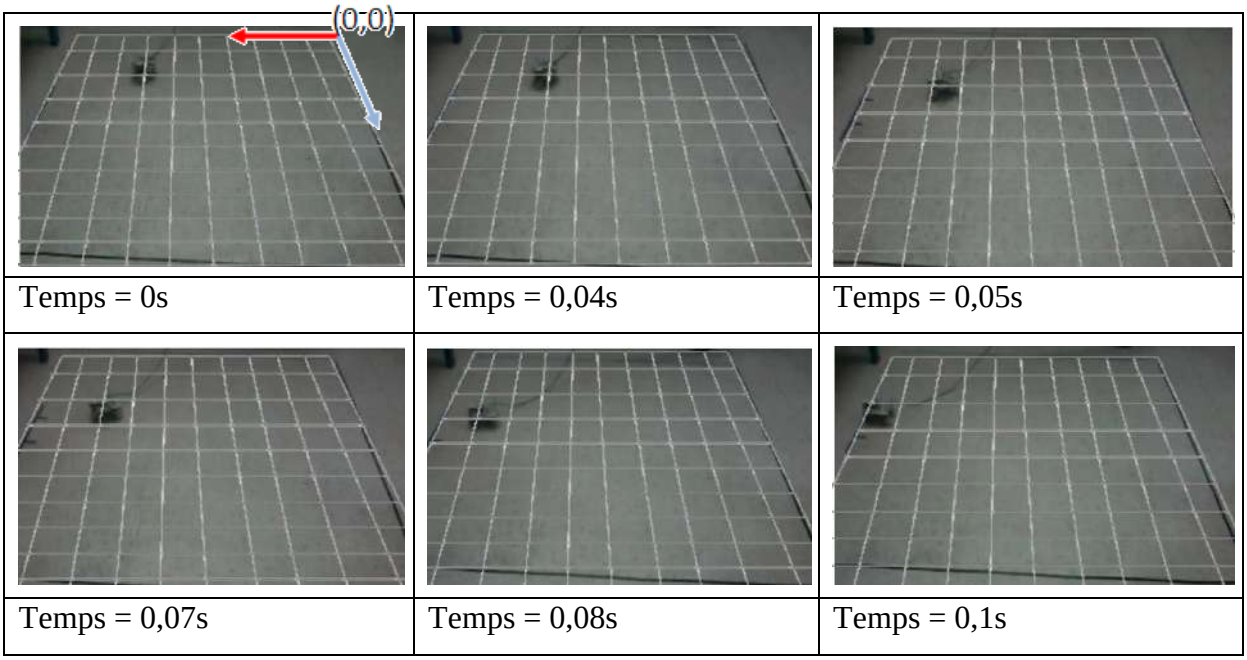


Figure III.27. *Exemple 3 : test de la réalisation*

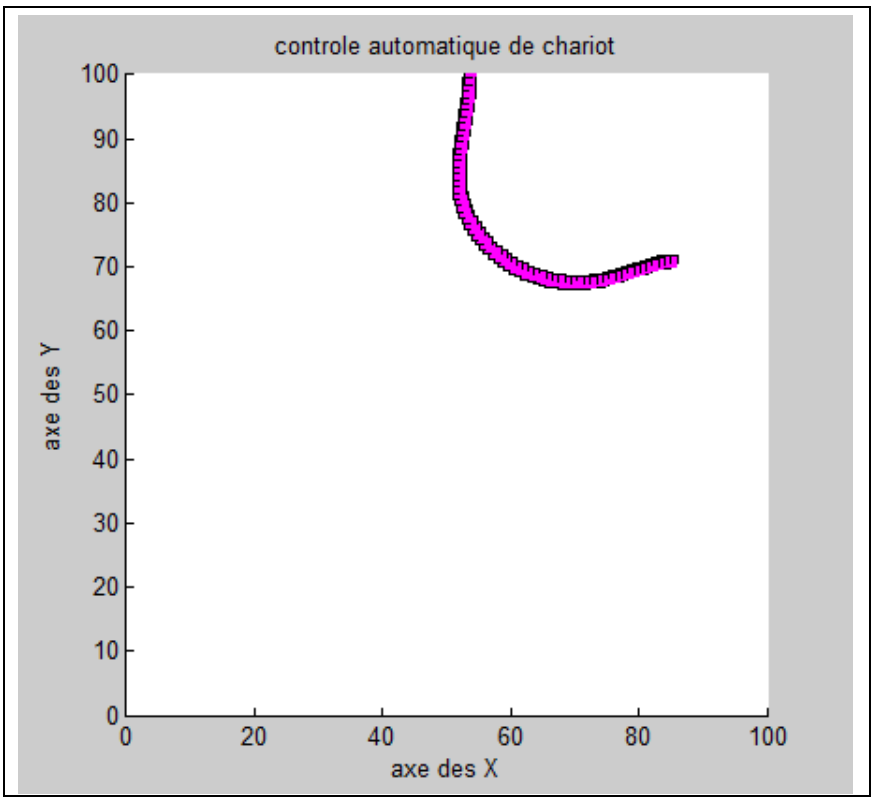


Figure III.28. *Résultat de l'exemple 3 dans Matlab*

On a fait plusieurs tests et on a retenu des solutions bien précises mais ces solutions ne donnent pas toujours de bons résultats car le contrôleur neuro-flou à des valeurs très petites donc nous ne pouvons pas optimiser tous les résultats dans ce moteur.

Alors dans ce cas on a donné un exemple de cas réel avec une comparaison dans Matlab illustrés dans les Figures III.26, III.27 respectivement quand le chariot ne reprend pas à notre besoin

- x = 15, ϕ = 0, y = 50. (Figure III.26).

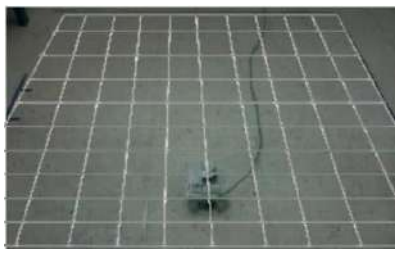
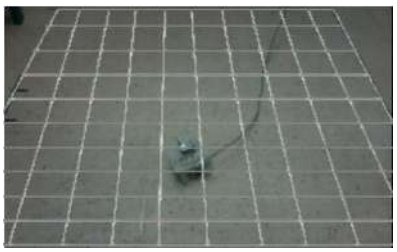
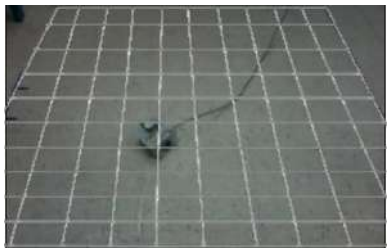
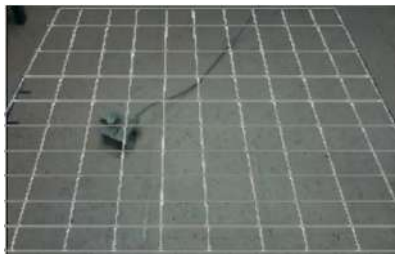
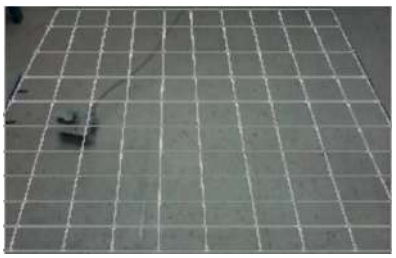
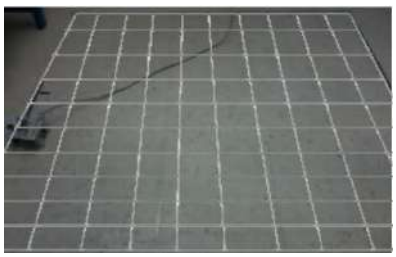
		
Temps = 0s	Temps = 0,03s	Temps = 0,1s
		
Temps = 0,15s	Temps = 0,2s	Temps = 0,25s

Figure III.29. *exemple de résultat non valide*

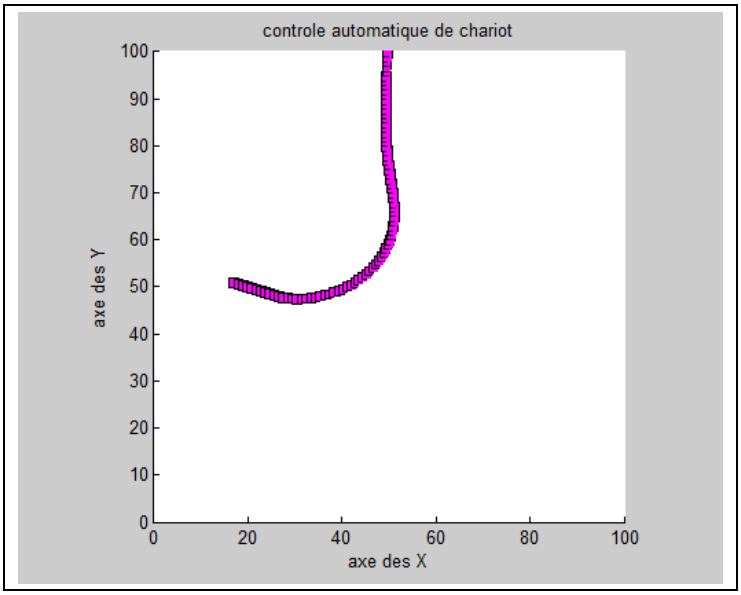


Figure III.30. *Résultat de l'exemple 3 dans Matlab*

7. Conclusion

Dans ce chapitre, le système mécanique conçu a été bien détaillé en insistant sur la roue d'orientation et la roue de déplacement.

L'implémentation de l'algorithme de contrôle neuro-flou de la roue de direction ainsi que du contrôle de la roue de déplacement sont détaillés.

Les résultats du test ont montré une grande similitude entre les résultats pratique et ceux de Matlab d'où la validité des travaux réalisé.

CONCLUSION GENERALE

CONCLUSION GENERALE

Dans ce travail on a développé technique basée neuro-flou pour le contrôle d'un chariot.

On a repondu à la problématique posée en introduction, à savoir la possibilité d'implémenter un contrôleur neuro-flou autour d'une solution numérique hardware.

En effet la logique floue à une grande capacité d'adaptation avec une bonne précision, en plus les réseaux de neurone sont utilisés pour optimiser le contrôleur flou.

Suite à la structure neuro-flou obtenue, un contrôleur autour de PIC32 en utilisant la carte CEREBOT 32MX4 de Digilent a été réalisé.

Le système de contrôle réalisé a permis l'implémentation d'une technique intelligente en utilisant la programmation hardware d'un PIC à 32 bits, ce dernier présente plusieurs avantages.

- Précision : et dépend des composants mécanique utilisés.
- Modularité : le système développé peut être directement embarqués dans une solution numérique complète de contrôle.
- Reconfigurabilité : a amené l'algorithme peut être adaptée à d'autre type de contrôle en changeant des exemples d'apprentissage.

Malgré les bonnes résultats obtenus dans notre travail, à cause des composants mécaniques moteurs, engrenage on n'a pas pour obtenir une très grande précision (ex : moteur pas à pas à 0.57°/pas).

Comme perspective à notre travail, on propose d'adapter notre solution a d'autre type de problème de contrôle connue par exemple contrôle d'un chariot avec marche arrière, dans une zone avec obstacle, ainsi réaliser les dispositifs de chariot avec des composants mécaniques de précisions.

Annexes

ANNEXE 1

Caractéristique de moteur pas à pas

1. Caractéristique

NMB-MAT
Minebea Motor Manufacturing Corporation
eMINEBEA.COM

PM35S-048



PM Motor
PM Type

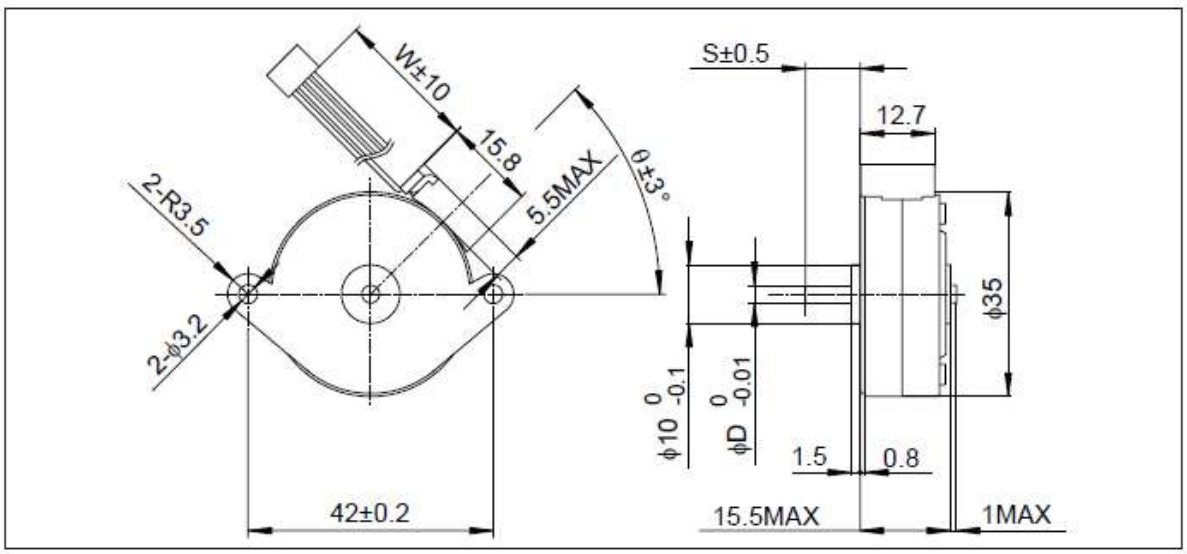
Reference Characteristics

Motor Size	PM35S-048	
Number of Steps per Rotation	48(7.5°/Step)	
Drive Method	2-2 PHASE	
Drive Circuit	UNIPOLAR CONST. VOLT.	BIPOLAR CHOPPER
Drive Voltage	24[V]	24[V]
Current/Phase		600[mA]
Coil Resistance/Phase	50[Ω]	6.6[Ω]
Drive IC	2SC3346	UDN2916B-V
Magnet Material	Ferrite plastic magnet (MSPL) Polar anisotropy ferrite sintered magnet (MS50) Nd-Fe-B bonded magnet (MS70)	
Insulation Resistance	100M[Ω] MIN	
Dielectric Strength	AC 500[V] 1[min]	
Class of Insulation	CLASS E	
Operating Temp.	-10[°C] ~ 50[°C]	
Storage Temp.	-30[°C] ~ 80[°C]	
Operating Hum.	20[%] RH ~ 90[%] RH	

Applications

OA Equipment : Printers / Scanners
Industrial equipment : Flow control valves
Automotive : Meters / Optic axis control device
Home automation appliances : Sewing machines

2. Dimension



ANNEXE 2

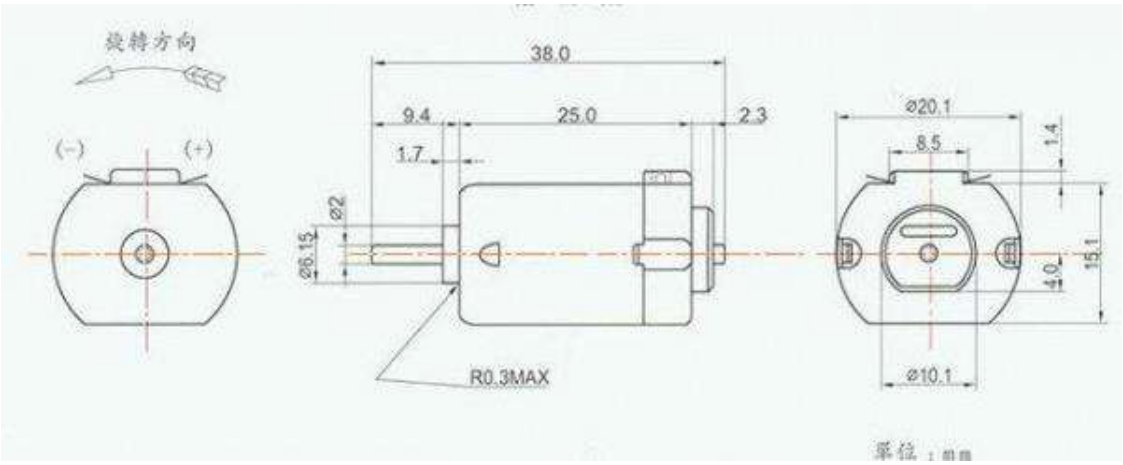
Caractéristique du moteur à courant continu utilisé



1. Caractéristiques

- Petit moteur mais puissant
- Tension d'alimentation de **1V à 6V**, une pile suffit
- Deux côtés plats pour faciliter le montage
- Alimentation nominale sous 3V
- Vitesse de rotation de **13 000 tr/min**, courant de **0.22 A** (sans charge)
- Efficacité maximale à 11 500 tr/min et 0.6A, **couple de 10g.cm**
- Couple moteur arrêté 40g.cm et 2.5A

2. Dimension

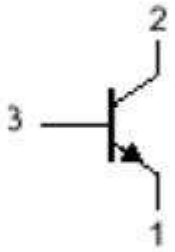


ANNEXE 3

Caractéristique de Transistors BD177



E C B



PINNING

PIN	DESCRIPTION
1	Emitter
2	Collector;connected to mounting base
3	Base

- $V_{CBO} = 60v$.
- $V_{CEO} = 60v$.
- $V_{EBO} = 5v$.
- $I_c = 5A$.
- H_{FE-1} (DC current gain) Min = 40, Max= 250.

Références

Bibliographiques

RÉFÉRENCES BIBLIOGRAPHIQUE

[1] G. Dreyfus, J. Martinez, M. Samuelides, M. Gordon, F. Badran, S. thiria and L. Hérault, réseaux de neurnes, méthodologies et application, Edition Eyrolles, 2010.

[2] A. Abraham, "cerebral quotion of neuro fuzzy techniques-hype or hallelujah," school of computing and information technology,Victoria, Australia, 2000.

[3] N. PALLUAT, Méthodologie de surveillance dynamique à l'aide des réseaux neuro-fous temporels, L'UFR des Sciences et Techniques: Thèse de doctorat, 2006.

[4] L. FOULLOY, S. GALICHET and A. TITLI, Commande floue 1, de la stabilisation à la supervision, Paris: Hermes Science Publications , 02-2003.

[5] W. Théodore and S. Gilbert, Electrotechnique, De Boeck Supérieur, 2000.

[6] M. E. KOUJOK, « Contribution au pronostic industriel :intégration de la confiance à un modèle prédictif neuro-flou,» Thèse de doctorat, Université de FRANCHE-COMTÈ, 2010.

[7] D. Nauck and R. Kruse, NEFCLASS–A neuro-fuzzy approach for the classification of data, Nashville, New York,: ACM Symposium on Applied Computing, February, 1995.

[8] C. P, V. A, L. E and I. B, "Méthodologie pour l'évaluation de perte de performance système dans le cadre d'une maintenance prévisionnelle," Centre de Recherche en automatique de Nancy(CRAN), Nancy-Université, CNRS 2009.

[9] N. PALLUAT, Microprocessor Design, McGraw-Hill Professional, 2006.

[10] encyclopédie WikiBis, "Microcontrôleur PIC," [Online]. Available: http://microprocesseur.wikibis.com/microcontroleur_pic.php. [Accessed 6 16 2014].

[11] V. Benoît, W. Philippe and C. Emmanuel, Les microcontrôleurs PIC programmés, Vingt-

deuxièmes Journées Francophones des Langages Applicatifs: Hernann Informatique. Studia Informatica Universalis, 2011.

[12] digilentinc, "Cerebott 32MX4™ Board Refference Manuall," August 26, 2011.

[13] H. Linacre and H. Jordan, Programming 32-bit Microcontrollers in C, 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA: Elsevier's Science & Technology, 2008.

[14] a. charbonnel, "Le moteur à courant continu," académie caen, Sciences de l'ingénieur en S, france, 28 10 2013.

[15] m. nicolas, "Les moteurs pas-à-pas," [Online]. Available: http://nicolas.marchildon.net/les-moteurs-pas-a-pas/. [Accessed 6 6 2014].