**République Algérienne Démocratique et Populaire**
**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Ecole Nationale Polytechnique**
Département d'Automatique
Laboratoire de commande des processus

# Mémoire de Magister

Spécialité : Automatique
Option : Robotique et Productique

Réalisé par
**BENAHMED Safia**
Ingénieur d'état en Automatique

Thème

# CONCEPTION ET ÉTUDE DE LA MAIN DLR/HIT ET EMBARQUEMENT DE LA COMMANDE SUR FPGA

soutenu publiquement le 12/01/2011

| | | |
|---|---|---|
| Président : | Mr.H. Boudjema | Professeur à ENP |
| Directeur de thèse : | Mr.R. Sadoun | Chargé de cours à ENP |
| Examinateurs : | Mr.H. Chekireb | Professeur à ENP |
| | Mr.M. Hemici | Maitre de conférences |

Année 2009/2010
**Ecole Nationale Polytechnique**
**ENP, 10 Avenue Hassen Badi, El-Harrach, Alger**

ملخص

في هذا العمل نقوم بإنشاء عن طريق البرنامج SolidWorks اليد الإصطناعية DLR/HIT التي نقوم بدراستها من الناحية النظرية، ثم نقوم بتطبيق التحكم PID الكلاسيكي بمواقع المحركات من دون مشط ذو التيار المستمر المستعملة.

التطبيق يكمن في التحكم باليد الثلاثية الأبعاد بالكترونيك قابلة للبرمجة بوسطة رقاقة FPGA مكنا نحقق الوصل بين الحقيقة والوهم؛ و.. لذالك يملئ جدول عن طريق التمثيل بقيم زوايا المفاصل التي سوف ترسل من بعد إلى البرنامج SolidWorks من رقاقة FPGA عن طريق API مبرمجة بـ Visual Basic

المفتاح: اليد DLR/HIT، الأيادي الإصطناعية، ثلاثي الأبعاد، SolidWorks، FPGA، Spartan 3 E، Soc، محركات من دون مشط التيار المستمر، التضمين السريع، النماذج الروبوتيكية، API، PID، Visual Basic

**Résumé**

Dans ce travail nous concevons dans l'environnement SolidWorks la main DLR/HIT. Les différents modèles :Géométrique, cinématique et dynamique sont calculés puis une commande PID classique de position est appliquée aux moteurs sans balai utilisés. La mise en œuvre consiste à commander la main en 3D par une électronique flexible via l'FPGA, ainsi nous établirons une connexion entre le virtuel et la réalité. Pour se faire, nous avons chargé un tableau par simulation, qui contiendra les valeurs des angles des articulations, qui seront communiquées par la suite à SolidWorks par la carte FPGA via son API en Visual Basic.

**Mots clés :** La main DLR/HIT, Mains artificielles, 3D, SolidWorks, FPGA, Spartan 3E, Soc, MPSoc, Prototypage rapide, Modélisation robotique, Moteurs sans balai à courant continu, API, Visual Basic, PID.

**Abstract**

In this work we design in the SolidWorks environment DLR/HIT hand, which we study its various models; kinematics, Jacobian matrix and dynamic, a conventional PID control of position is applied to the BLDC motors used.

The implementation consists to control the hand in 3D by a flexible electronics via an FPGA card, thus we will establish a connection between virtual and reality; to be done, we load a table by simulation, which contains the values of the angles of the joints, which will be communicated to SolidWorks by the FPGA via the API in Visual Basic.

**Key words:** The DLR/HIT hand, Artificial ands, 3D, SolidWorks, FPGA, Spartan 3E, Soc, MPSoc, Rapid prototyping, Robotic modelling, BLDC motors, API, Visual Basic, PID.

*« A tous ceux qui me sont chers »*

*Safia*

# Remerciements

Mes premiers remerciements vont à monsieur R.Sadoun chargé de cours à l'Ecole Nationale Polytechnique pour avoir encadré ce travail.

Un spécial remerciement pour Issaad Massinissa et Boushaba Mohammed Houssam Eddine avec qui j'ai travaillé pour la programmation de la carte FPGA.

Je remercie monsieur H. Boudjema Professeur à l'ENP pour avoir accepté de présider le jury.

Je remercie également monsieur H. Chekireb Professeur à l'ENP et monsieur M. Hemici Maitre de conférences pour avoir accepté de juger ce présent travail.

Un grand merci pour tous ceux qui m'ont soutenue pour ne pas laisser tomber cette recherche.

# Liste des figures

# Liste des tableaux

# Table des matières

# *Abréviation*

| | |
|---|---|
| *BSB* | Base System Builder |
| *BRAM* | Block of Random Access Memory |
| *BLDC* | Brush less Direct current |
| *RRRR* | Chaine simple robotique à quatre rotations |
| *EDK* | Embedded Development Kit |
| *FPGA* | Field Programmable Gate Array |
| *HIT* | Harbin Institute of Technology |
| *DLR* | Institute of Robotics and Mechatronics, German Aerospace Center |
| *ISE* | Integrated Software Environment |
| *LCP* | Lois de commande en position |
| *MHS* | Microprocessor Hardware Specification |
| *MSS* | Microprocessor Software Specification |
| *MGD* | Modèle géométrique direct |
| *MGI* | Modèle géométrique indirect |
| *PlatGen* | Platform Generator |
| *PI* | Proportionnel intégral |
| *PID* | Proportionnel intégral dérivé |
| *SDK* | Software Development Kit |
| *SoC* | System on Chip |
| *XPS* | Xilinx Platform Studio |

# Introduction

Cela fait plusieurs décennies que les centres de recherche s'intéressent aux prothèses intelligentes [1], telles que celles des mains ou celles des jambes, dont les résultats étaient utilisés dans plusieurs domaines, où nous citons par exemple le domaine spatial, la chirurgie endoscopique ou même en guise de prothèse. L'idée de ce travail est d'arriver à réaliser et commander une telle main robotique, pour y arriver deux façons auraient été possibles, la première est de passer à la réalisation physique des différentes parties de cette main ainsi le coût sera élevé et l'architecture figée, la deuxième façon est de choisir une réalisation virtuelle qui peut communiquer avec une électronique externe, ainsi et selon les résultats observés, l'architecture proposée de cette main pourrait être améliorée et validée : C'est le concept du prototypage rapide.

Pour se faire, nous avons choisi la main DLR/HIT comme prototype, le choix d'une main qui existe déjà et qui a été sujet de nombreuses améliorations nous permet de gagner un pas d'avance dans la conception, le choix spécial de cette main n'est pas arbitraire car son principe fondamental est la modularité, du fait la réalisation de la main revient principalement à la réalisation d'un seul doigt. Toujours dans l'esprit du prototypage rapide le

choix d'une carte FPGA nous permet de changer tout au long de nos applications les périphériques ou l'architecture de notre carte de commande.

La conception a été établie sous SolidWorks, puis commandée par la carte FPGA Spartan-3E, en conséquence nous avons pu créer une connexion entre réalité (la commande sur FPGA) et virtuel (la conception), l'outil qui nous a servi pour établir la communication entre ces deux environnements est l'API via Visual Basic. Pour se faire, nous avons partagé le travail en cinq chapitres présentés comme suit :

Le premier chapitre porte sur l'état de l'art où nous effectuons une étude des mains robotiques existantes, puis nous avons choisi la main DLR/HIT dont les principales caractéristiques sont détaillées. Aussi le choix de l'environnement de travail SolidWorks est justifié.

Le deuxième chapitre porte sur la conception mécanique de la main, sous SolidWorks, un des avantages de travailler dans cet environnement est qu'il calcule la masse, le centre de masse et le tenseur d'inertie des pièces ; ainsi nous détournons une phase de calculs fastidieux pour la modélisation.

Le troisième chapitre : La modélisation où nous étudions et validons les différents modèles : Géométrique direct et indirect, cinématique direct et indirect et le modèle dynamique. Les calculs ont été établis par programmation à l'aide des variables symboliques sous MATLAB.

Dans le quatrième chapitre ou chapitre commande, nous proposons d'abord une commande décentralisée par PID classique, puis nous introduisons le concept de la commande en effort d'où la commande en impédance.

Nous avons consacré le cinquième chapitre à l'FPGA,la carte de commande. La configuration et la programmation de la carte que nous avons choisi dépend de plusieurs paramètres telle que l'architecture proposée pour l'électronique où nous évoquons le concept du Soc et de l'MPSoc.

# CHAPITRE I
# L'ETAT DE L'ART

## *I.1 Introduction*

Le développement d'une main habile robotisée est un sujet de défi qui a été poursuivi par plusieurs équipes de recherche. Beaucoup de mains adroites de robot ont été conçues pendant les dernières trois décennies. [ 1 ]

Les premières recherches remontent à la 2ème guerre mondiale, où plusieurs soldats étaient amputés du bras. Ce qui a fait croître la demande des prothèses dites primaires, d'une forme très modeste ; elles n'avaient pas d'articulation et étaient équipées d'un très simple crochet, et qui en réalité ne représentaient qu'une consolation morale ; ainsi la fabrication des mains artificielles est devenue un thème très intéressant pour plusieurs centres de recherches à travers le monde. L'objectif final de ces efforts est de réaliser une main capable d'imiter le plus possible les mouvements naturels d'un être humain. Un autre objectif majeur dans ce domaine est de promouvoir les caractéristiques tel que réduire la distorsion et de rendre l'organe intelligent capable de comprendre pour réagir. [ 2 ]

La cause humaine est l'intérêt principal de toute prothèse, entre autre, plusieurs objectifs sont ciblés tel que l'étude de l'intelligence naturelle pour construire une main

artificielle autonome apte à travailler dans des zones hostiles, citons comme exemple les robots destinés à l'exploration spatiale ou plutôt pour perfectionner le réalisme des animations en trois dimensions ou la simulation des accidents de véhicules, des opérations chirurgicales, des applications sportives ou autres.

## *I.2 L'état de l'art*

Plusieurs recherches ont tenté de créer une main robotique semblable, plus ou moins à la main humaine, selon les applications et les besoins visés. Nous présentons dans ce qui suit un ensemble de mains déjà réalisées ; Dont la ressemblance avec la main humaine, le nombre de doigts, de degrés de liberté, d'articulation et d'actionneur et la stratégie de commande diffèrent.

### I.2.1 Modèle de Barrett

**L**a main de Barrett, produite par Barrett Technology, est basée sur une conception développée à l'université de Pennsylvanie. C'est une main mécanique composée de trois doigts. Chaque doigt possède deux articulations rotoïdes (Figure 1-3). Un doigt est fixe et les deux autres peuvent s'écarter jusqu'à 180 degrés par rapport à la paume (le doigt 3 est le doigt fixe et les doigts 1 et 2 tournent autour de la paume), et ceci d'une manière synchrone. La main est commandée par quatre moteurs qui activent sept liaisons au total.[ 3 ]

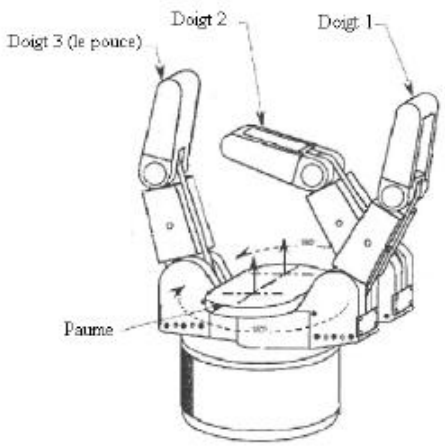



**Figure I.1 Modèle de Barrette**

### I.2.2 La main SARAH

**L**a main SARAH (Self-Adaptive Robotic Auxiliary Hand), est une main à trois doigts comme celle de la main de Barrett. Cette main est installée sur la station spatiale

internationale. Le but de cette main est la saisie des objets dans l'espace. Les trois doigts sont synchronisés de telle sorte qu'ils s'ouvrent ensemble ou se ferment ensemble. Cette main est motorisée par seulement deux moteurs indépendants, un pour l'ouverture/fermeture et un autre pour l'orientation des doigts. Cette main possède 10 articulations, [ 3 ].

**Figure I.2 La Main de SARAH**

### I.2.3 La main Robonaut

**L**a main Robonaut a été développée par la NASA au centre de l'espace Johnson. Cette main possède 5 doigts et un total de 14 degrés de liberté, 12 pour les 5 doigts et 2 pour le poignet Sa taille est équivalente à 95% à celle d'une main humaine.

L 'index, le majeur et le pouce sont considérés comme les doigts primaires de manipulation. Et ont trois degrés de liberté chacun. L'annulaire et l'auriculaire sont décalés par rapports aux autres doigts ce qui leurs donnent la capacité de s'enrouler autour de l'objet à saisir, ils n'ont qu'un seul degré de liberté chacun. Ils sont capables d'exercer des efforts de serrage importants. Le degré de liberté qui reste est pour la paume pour réaliser la saisie. Cette main comprend 14 moteurs au total situés tous au niveau de l'avant-bras. [3] [4]

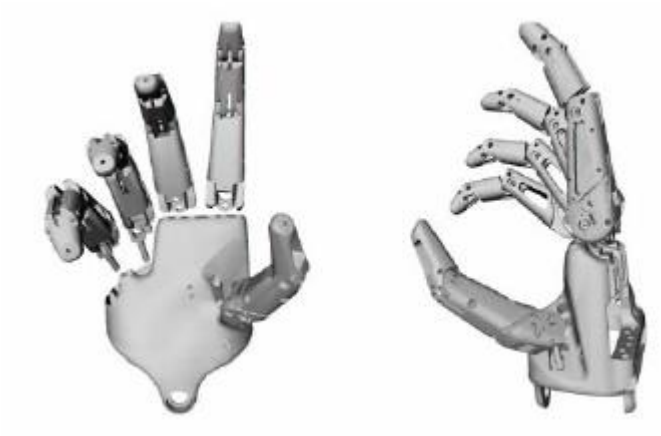


**Figure I.3 La main Robonaut**

### I.2.4 La main de Shadow

**Figure I.4 La main de Shadow**

Cette main créée par la compagnie Shadow Robot en Angleterre est, à ce jour**,** la main robotique la plus proche de la main humaine. Elle contient 24 degrés de liberté qui sont contrôlés par des muscles pneumatiques situés tous au niveau de l'avant-bras ; ces 24 degrés de liberté sont distribués comme suit : 4 DOF pour les trois premiers doigts(l'index le majeur et l'annulaire), 5 pour l'auriculaire et 5 autres pour le pouce et 2 pour le poignet[4]. Cette main robotique peut être utilisée dans plusieurs applications : télé présence, mobilité virtuelle, industrie etc. Les liaisons des doigts sont actionnées par des muscles pneumatiques antagonistes. Par contre, quelques articulations sont actionnées par un seul muscle et le mouvement opposé se fait à l'aide d'un ressort. [3]

### I.2.5 La main RTR de l'école supérieure Sant'Anna

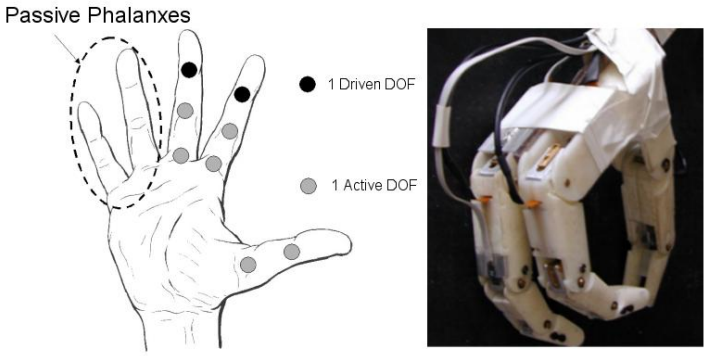



**Figure I.5 La main RTR**

Elle comporte trois doigts et 6 degrés de liberté, deux pour chacun, la structure du pouce est différente de celle de l'index et du majeur. Les actionneurs sont des micro- moteurs sans bagues à courant continu, le mouvement est converti en un mouvement linéaire à l'aide d'un système de vis sans fin. Elle est caractérisée par son poids et son volume semblable à celui d'une main humaine de taille moyenne. [5]

### I.2.6 La main de Belgrade/USC

Avec 5 doigts et un actionneur électrique unique qui met tous les doigts à la même position, également caractérisé par son système de contrôle simple.

### I.2.7 La main d'Utah/MIT

Avec 4 doigts et 4 d.o.f. chacun, le poignet avec 3 DOF, une transmission faite avec des tendons, une mise en fonction pneumatique. Son système de contrôle peut accomplir des tâches par la planification de trajectoire et l'intégration de la vision.

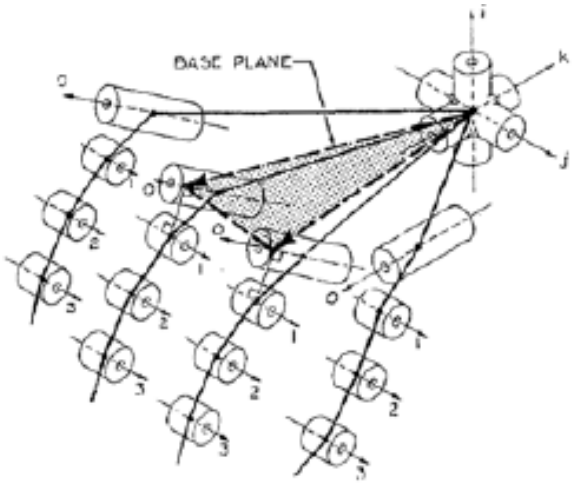



**Figure I.6 Modèle de la main d'Utah/MIT**

### I.2.8 La main de Rovetta

De la polytechnique de Milano a seulement 3 doigts et elle est télécommandée par un utilisateur portant un gant. Le gant fournit également le feedback de force à la main de l'utilisateur.

### I.2.9 La main de Grasper

Avec 3 doigts et 9 d.o.f., a seulement 3 actionneurs et les joints sont actionnés par ordre. [5]

### I.2.10 Comparaison entre quelques mains robotiques

Voici un tableau qui récapitule toutes les caractéristiques fondamentales qui différencient et caractérisent les mains robotisées déjà réalisées dont certaines sont déjà citées au-dessus.

| Main | Nombre de doigts | Taille de la main par rapport à la main humaine | DOF | Nombre d'actionneurs | Pois en Kg | Force (N) |
|---|---|---|---|---|---|---|
| **Main humaine** | 5 | 1 | 22+2 poignet | 38 ext+int | Environ 0.4 | >300 |
| **Salisbury** | 3 | 1.2 +contrôle | 9 | 12 ext | 1.1 | 44 |
| **Utah/MIT** | 4 | 2 +contrôle | 16 | 32 ext | | 31 |
| **DIST** | 4 | 1.5+contrôle | 16 | 20 ext | 1 | |
| **Sugiuchi** | 5 | | 17 | | | |
| **Anthrobot** | 5 | 1 +contrôle | 20 | 16 ext | 4.5 | |
| **SDSU** | 5 | 1 +contrôle | 15 | 6 ext | 2 | 15/doigt |
| **NTU** | 5 | | 17 | | 1.57 | |
| **Shadow** | 4 | 1.2 +contrôle | 22+2 poignet | | 4 | |
| **BUAA** | 4 | 1 +contrôle | 16 | | 1.4 | |
| **Goldfinger** | 4 | 1.5 +contrôle | 12 | | 2.27 | |
| **Barrett** | 3 | 1.2 | 8 | 4 int | 1.18 | 20/doigt |
| **Laval 1** | 3 | 2 | 12 | 6 ext | 9 | 687 |
| **Laval 2** | 3 | 1.5 | 10 | 2 ext | | |
| **Gifu II** | 5 | 1.5 +contrôle | 20 | 16 int | | 4.9 |
| **Robonaut** | 5 | 1.2 +contrôle | 12+2 poignet | 12+2 ext | | |
| **DLR II** | 4 | 1.5 +contrôle | 13 | 13 int | 1.8 | 30/doigt |
| ***Grasper*** | 3 | | 9 | 3 | | |
| ***Rovetta*** | 3 | | | | | |
| **Belgrade/USC** | 5 | | 1 | 1 | | |
| **SARAH** | 3 | | 2 | 2 | | |
| **RTR II** | 3 | 1 | 9 | 2 int | 0.32 | 16 |

**Tableau I.1 Analyse comparative -Etat de l'art- [6]**

## *I.3 La main DLR/HIT*

Depuis 1997, le centre aérospatial Allemand (DLR, *Institute of Robotics and Mechatronics, German Aerospace Center)* a développé deux générations de mains adroites multi sensorielles de robot : la main DLR I [ 7 ] et II [ 8 ].

**Figure I.7 DLR I- 1997**

**Figure I.8 DLR II- 2001**

**a. L'historique**

Basé sur l'expérience de la main DLR I, la main DLR II a été conçue pour qu'elle soit plus puissante et plus fiable. Le nombre de câbles entre la main et le microprocesseur principal a été considérablement réduit plus de 400 à seulement 12. La combinaison optimale des moteurs BLDC, du réducteur harmonique, des courroies et des pignons différentiels ont fait que la force du bout du doigt atteint 30 N. Le degré de liberté supplémentaire du pouce a augmenté non seulement la puissance de la saisie mais également a permis les fines manipulations.

De l'autre côté, il n'est pas facile de fabriquer une telle main, particulièrement pour les actionneurs où tous les moteurs sont particulièrement conçus et les sondes analogues de hall doivent être collées et calibrées prudemment. Depuis 2001, basés sur l'expérience de la main DLR II, le HIT *(Harbin Institute of Technology)*et le DLR ont conjointement développé une main

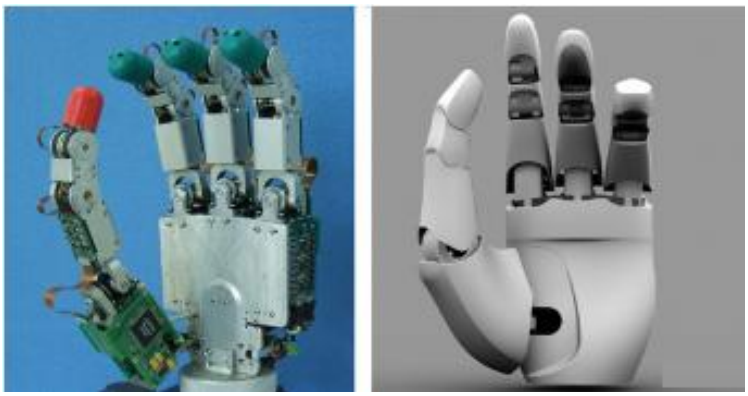


**Figure I.9 DLR/HIT-2005**

adroite à quatre-doigt modulaire :La main de DLR/HIT. Le but du projet est d'établir une plus petite main que la main DLR II qui peut être fabriquée en petite série. Le prix devrait être aussi bas que possible et la fiabilité aussi haute que possible.

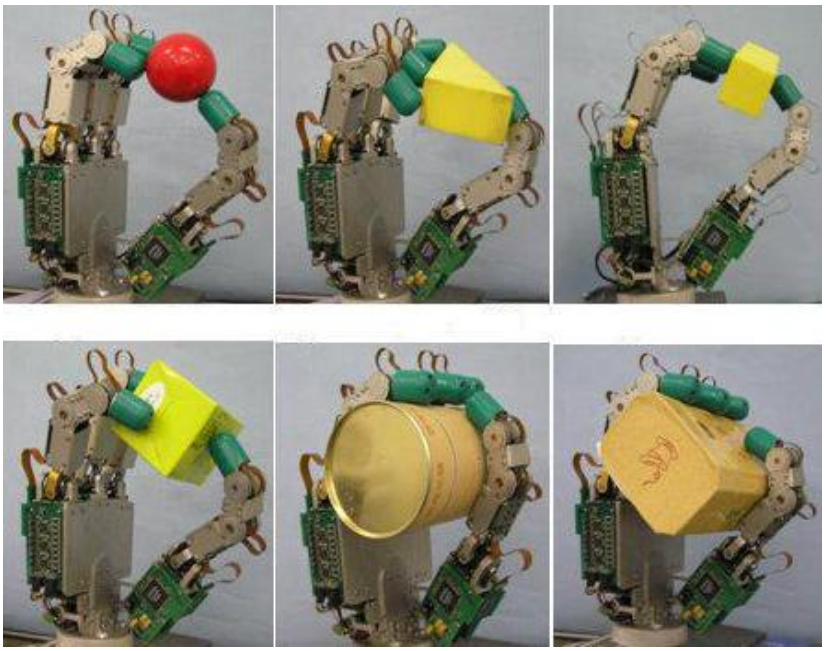


**Figure I.10 Exemples de saisies de différents objets**

Pour réaliser un degré élevé de modularité, les quatre doigts sont identiques. Chaque doigt a trois degrés de liberté et quatre articulations, les deux dernières sont mécaniquement couplées par une barre rigide. Le pouce a un degré de liberté additionnel pour réaliser le mouvement relatif à la paume, ce qui donne le nombre total de 13 degrés de liberté. Tous les actionneurs sont intégrés dans la base du doigt ou le corps du doigt directement, l'électronique et les contrôleurs de transmission sont entièrement intégrés dans la base du doigt afin de réaliser la modularité de la main et réduire au minimum le poids et la quantité de câbles requis.

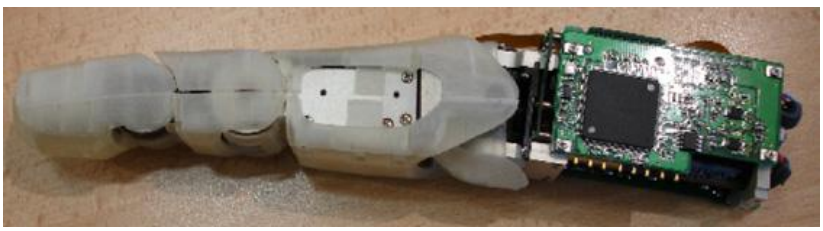


**Figure I.11 Un doigt de la main DLR/HIT**

## b. L'électronique

### Les entrées/ sorties

La définition des entrées nous permettra de réserver les périphériques d'entrées de la carte spartan3 utilisée (que nous définissons plus tard) ainsi que le protocole utilisé pour la lecture des données, l'FPGA calculera la commande et envoie ces signaux aux cartes de puissance des moteurs ; les périphériques d'envoi doivent être aussi réservés. La figure suivante définie l'architecture électronique du doigt DLR/HIT. Des circuits de traitement des signaux et des convertisseurs ANALOGIQUE-NUMÉRIQUE périodique à grande vitesse (bit 12) sont également intégrés pour convertir les signaux analogiques provenant des capteurs, la figure V.3 résume le branchement électronique et la localisation de ces capteurs.

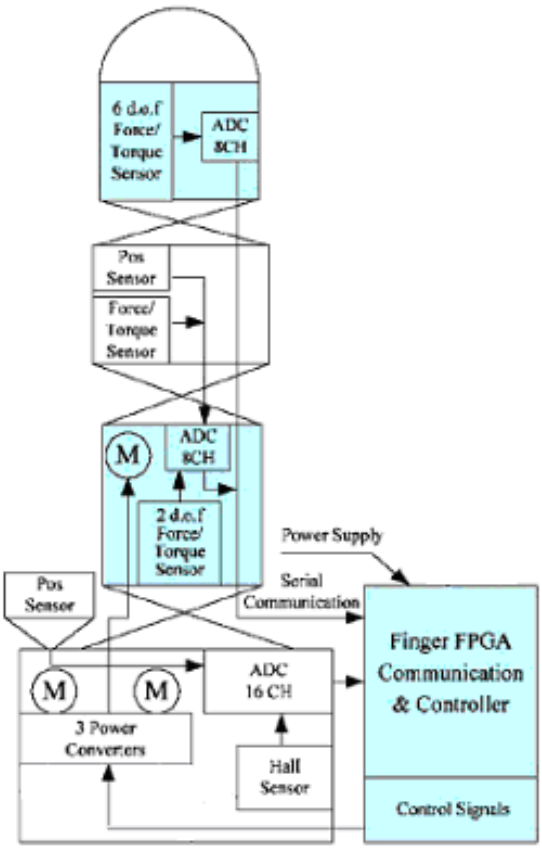



**Figure 1.12 L'architecture électronique d'un doigt DLR/HIT**

### Les capteurs [1]

Pour réaliser une simple commande en position ou une commande en impédance l'observation de la sortie sera nécessaire et va être traduite dans notre cas par des capteurs. Le tableau (V.1) résume l'ensemble des capteurs utilisé pour un seul doigt

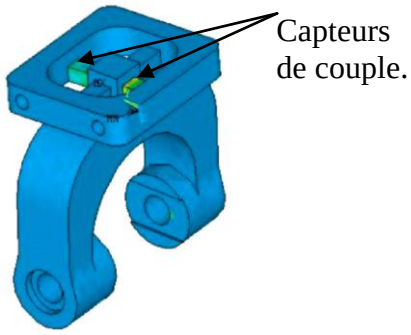
.

| Type des capteurs | Nombre de capteur par doigt |
|---|---|
| Couple d'articulation | 3 |
| Position de l'articulation | 3 |
| Position du moteur | 3 |
| Force/couple | 1 |
| Température | 2 |

.

Tableau 1.2 les capteurs du doigt DLR/HIT [1]

Les capteurs de couple : deux types de capteur de couple sont utilisés, le premier conçu spécialement pour l'articulation de la base avec deux degrés de liberté (Figure I.13).
Le deuxième capteur de couple relie l'articulation moyenne et distale (figure I.14)



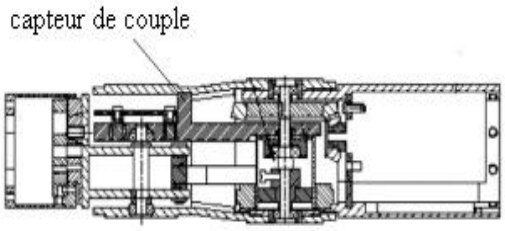




**Figure 1.13 Design du pignon différentiel** **Figure 1.14 L'unité du doigt**

Un capteur miniaturisé composé de six capteurs (force/couple) (20 millimètres de diamètre et 16 millimètres de hauteur) (figure I.15), avec sortie numérique a été développée pour la fin du doigt, composé de six fils et une vitesse de transmission des données de 15.6 kilohertz. Les gammes de mesure de force et de couple sont 30 N et 600 N millimètre respectivement.

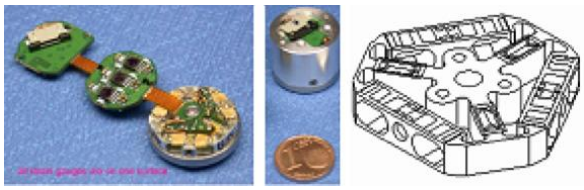


**Figure 1.15 Capteur de force à six degrés de liberté**

Différents capteurs de température sont utilisés dans la structure réelle de la main DLR/HIT pour la surveillance des températures des moteurs.

### Les moteurs

Chaque moteur contient huit câbles au total, trois pour le signal de commande, trois pour les capteurs à effet Hall et un pour l'alimentation et un autre pour la masse.

## *I.4 Les outils de conception assistée par ordinateur*

Il existe dans le commerce un multiple choix de logiciel de conception mécanique assistée par ordinateur (CAO)tel que AutoCAD, SolidWorks ou même d'autres logiciels orientés architecture ou conceptions plus complexes tel que 3DS Max Studio conçu spécialement pour les

vidéos, les jeux vidéos et les effets spéciaux où plusieurs paramètres sont pris en considération tel que la nature climatique et la gravité ; Tous ces logiciels pourraient être choisis comme outil de conception, mais pour plusieurs raisons qui vont être citées plus tard notre choix porte sur SolidWorks.

### I.4.1 Le choix de SolidWorks

SolidWorks est un logiciel commercial largement utilisé pour la modélisation et la conception assistée par ordinateur. Il est basé sur la définition des paramètres des composants et des fonctionnalités et il peut être utilisé d'une façon très intuitive. [ 9 ]

La figure I-16 donne une vue d'ensemble de l'environnement SolidWorks ainsi que ses différentes parties

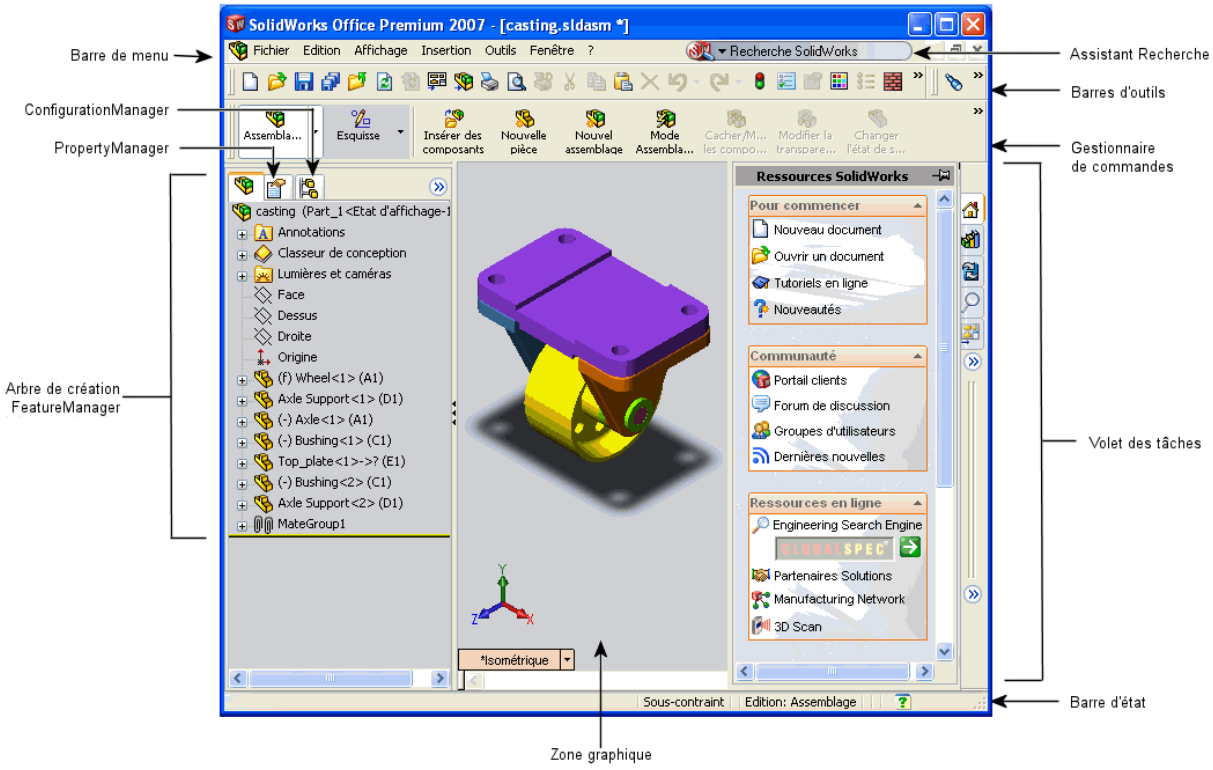



**Figure I-16 Vue d'ensemble de l'environnement SolidWorks.** [ 10 ]

Plusieurs raisons justifient le choix de SolidWorks comme environnement de conception pour notre projet qui sont comme suit :

- Une bibliothèque assez riche pour le choix des matériaux utilisés pour les différentes pièces.
- Des outils de simulation performants.

- La simulation des mouvements des pièces les unes par rapport aux autres, sans oublier la détection de collision et les contraintes mécaniques entre les volumes et la simulation des frottements.
- Le calcul de la masse, le calcul des centres d'inerties et la simulation des forces appliquées.

Toutes ces fonctionnalités permettent de s'approcher le plus possible du modèle réel souhaité.

## *I.5 Conclusion*

A travers ce chapitre nous avons fait défini et comparé plusieurs mains robotiques par rapport à leurs nombres de doigts, de degrés de liberté (dextérité) d'actionneurs et leurs emplacements, leurs volumes et leurs poids, et dont la stratégie des tâches est totalement différente entre la plupart d'entre elles. Par la suite, nous avons choisi un modèle qui est le DLR/HIT développé et amélioré depuis une douzaine d'année, en premier lieu par l'institut aérospatial Allemand puis en collaboration avec le HIT chinois.

La main DLR/HIT, est définie par :

- La modularité : ce qui signifie que tous les doigts sont identiques et qui va par la suite faciliter la fabrication
- Un total de 4 doigts et 13 degrés de liberté.
- Chaque doigt comporte 3 degrés de liberté et 4 articulations, les deux premières se trouvent à l'extrémité de la paume et assurent le mouvement d'abduction/adduction et flexion/extension, et les deux autres articulations sont mécaniquement couplées et ont un seul degré de liberté.
- Les actionneurs sont des moteurs sans balais à courant continu et sont directement intégrés soit à la structure du doigt soit au niveau de la paume. Ce qui est une caractéristique importante en comparant aux structures qui emploient des actionneurs pneumatiques dont l'emplacement est au niveau de l'avant bras ce qui mène à utiliser plus de volume.

La conception de la main DLR/HIT fera l'objet du prochain chapitre est va être conçue comme prévue dans l'environnement SolidWorks. Il nous reste à définir toutes

l’électroniques y compris les capteurs, que nous allons introduire dans le chapitre 5 pour ne pas encombrer le premier chapitre.

# CHAPITRE II
# CONCEPTION

## *II.1 Introduction*

Dans ce chapitre nous nous intéresserons à la partie mécanique de la main ce qui inclue la conception des pièces tel que : les moteurs, les supports moteurs, les courroies les engrenages et leurs montages pour la réalisation d'un doigt, puis nous proposons un positionnement pour tous les doigts ainsi que l'ajout de la cinquième articulation pour le pouce.

Pour se faire, nous avons choisi SolidWorks comme environnement de travail, SolidWorks nous permettra non seulement la conception mais il sert aussi comme outil de vérification, validation, animation et ce qui est aussi très important, il calculera le tenseur d'inertie, la masse et la position du centre de masse pour chaque partie du doigt qui vont être utilisés pour le calcul du modèle dynamique lors du prochain chapitre.

Le doigt de la main DLR-Hit- se compose de deux parties indépendantes: la première qu'on appellera la base du doigt et qui inclura les deux premiers moteurs, leurs supports et le pignon différentiel. La deuxième partie est appelée l'unité du doigt, elle comportera les trois phalanges et elle constituera la partie motrice du doigt. Tous les mécanismes utilisés vont être introduis et définis au fur et à mesure.

## *II.2 La base du doigt DLR/HIT* [ 1 ]

Avec ses deux degrés de liberté elle assure au doigt les mouvements d'abduction/adduction et flexion/extension qui seront assurés par le couplage de deux moteurs sans bagues à courant continu identiques (à l'aide d'un pignon différentiel). Les moteurs sont de dimensions de 16 millimètres de diamètre et 28 millimètres de longueur. Ils sont également

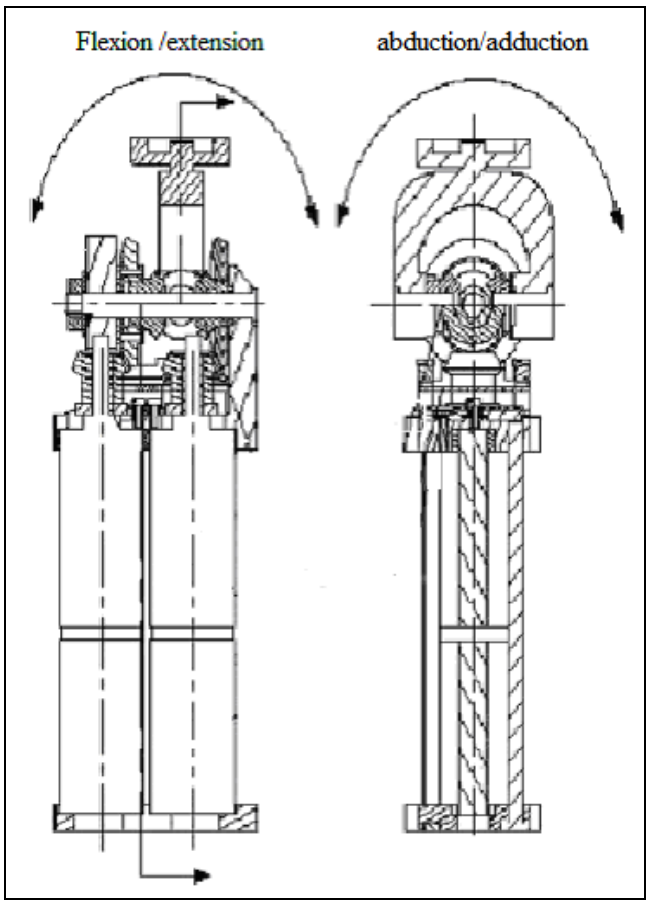



**Figure II-1 Unité : Base du doigt**

menés d'un réducteur intégré de rapport 159:1 ce qui leurs donnera une nouvelle dimension qui sera 16mm de diamètre et 67mm de longueur (moteur+ réducteurs intégrés)

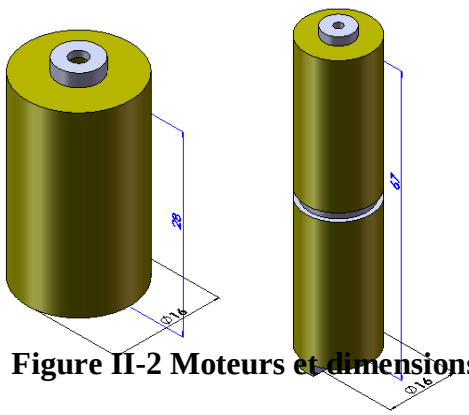



**Figure II-2 Moteurs et dimensions**

Les vitesses planétaires d'entraînement avec le rapport 159:1 de réduction sont directement couplées aux moteurs, et les pignons coniques sont reliés aux vitesses planétaires par l'intermédiaire de la réduction additionnelle de vitesse de 2:1.

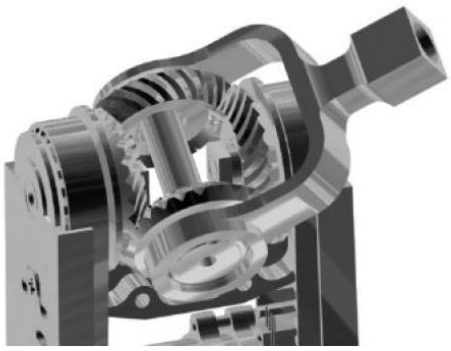
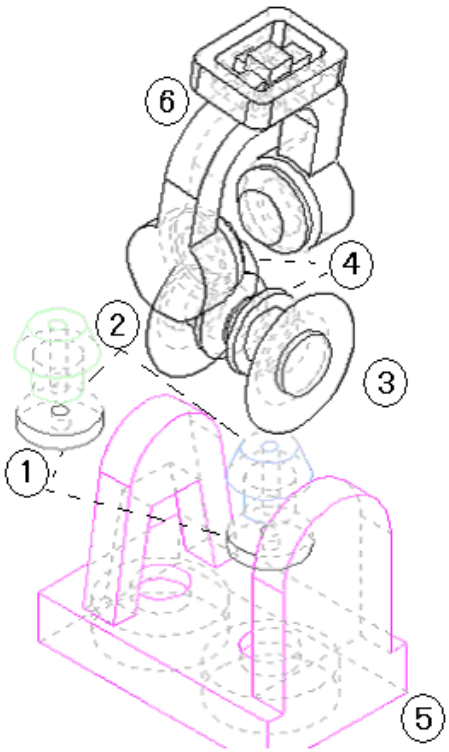


**Figure II-3 Pignon différentiel**

Pour le mouvement d'abduction/adduction les moteurs appliquent un mouvement synchrone aux pignons coniques en utilisant le couple des deux moteurs. Pour le mouvement de flexion/extension les moteurs tournent dans des directions contraires. Ceci cause un mouvement de flexion sur le bout du doigt en utilisant le couple des deux moteurs et signifie que nous pouvons employer de petits moteurs et réducteurs tout en atteignant la double force de rendement sur le bout du doigt.




1-Joints
2-Pignons moteurs
3-Pignons planétaire
4-Pignons
5-Support supérieur des moteurs
6-Pignons satellitaires.

**Figure II-4 Constituants d’un pignon différentiel**

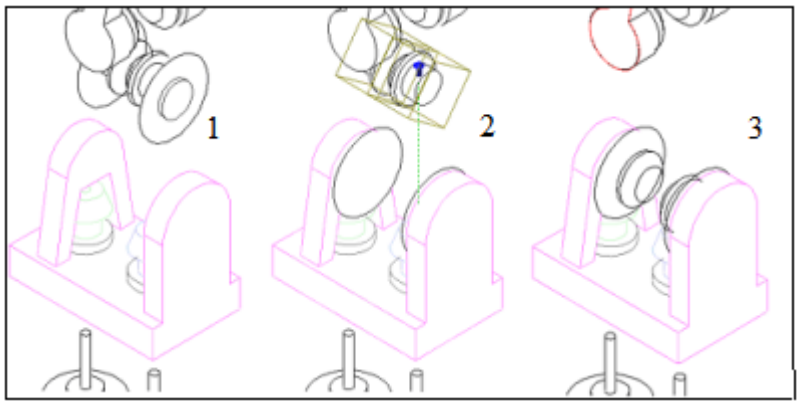



**Figure II-5 Montage du pignon différentiel**

Le pignon différentiel est fixé au support supérieur des moteurs, ces derniers sont aussi fixés par le support inférieur pour les empêcher de tourner sur eux-mêmes une fois sous tension. Figure(II.6) ; la figure(II.7) défile les étapes du montage des différents éléments, le pignon satellitaire est fait de la sorte  pour assurer la fixation de l'unité du doigt.

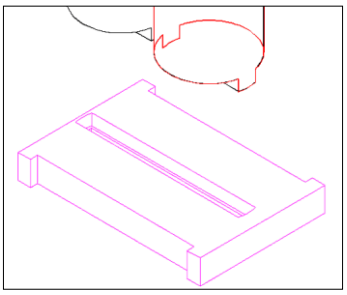


**Figure II-6 Support inférieur des moteurs**

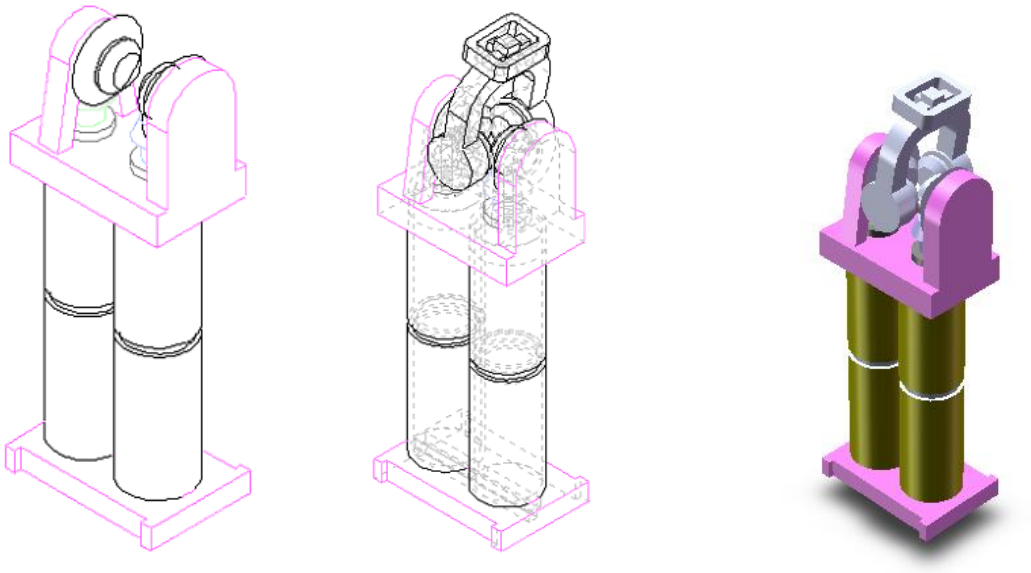


**Figure II-7 Aspect final de la base du doigt**

## *II.3 L'unité du doigt*

L'unité du doigt a un degré de liberté et deux articulations, l'actionneur est identique à ceux utilisés dans la base du doigt sauf que pour des raisons de contraintes d'espace, le réducteur n'est pas intégré, pour remédier à cette contrainte, l'utilisation d'un

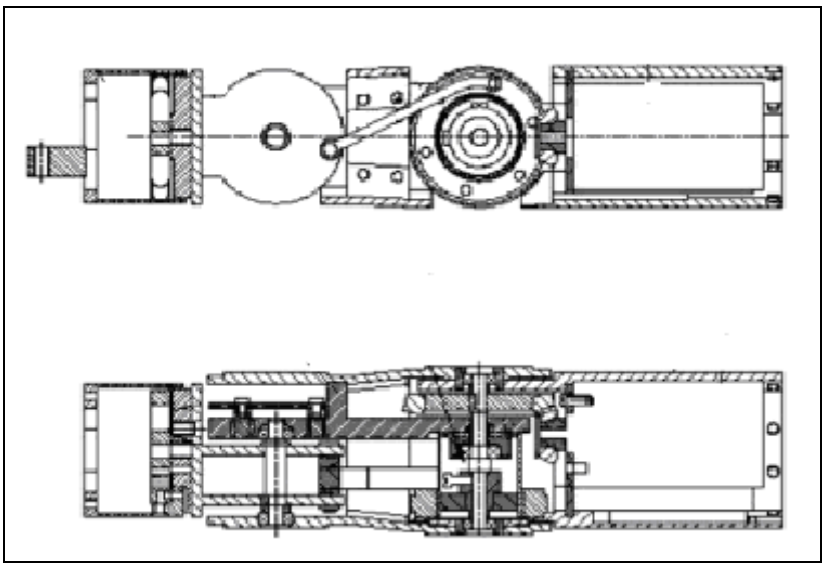


**Figure II-8 L'unité du doigt[ 1 ]**

réducteur harmonique serai indispensable. La caractéristique fondamentale d'un réducteur harmonique c'est qu'il réduit les vitesses à l'aide seulement de trois pièces à axe creux, la première est fixe, les deux autres tournent sur le même axe à des vitesses différentes, les trois sont imbriquées l'une dans l'autre d'où l'espace nécessaire est réduit.

**Figure II-9 Réducteur harmonique**

Notre réducteur harmonique est de rapport 100:1 de réduction, il mesure 20 millimètres de diamètre et 13.4 millimètres de longueur. La vitesse du moteur est transmise au réducteur harmonique à l'aide du pignon noir, le réducteur harmonique réduit la vitesse et elle sera transmise au mécanisme par l'axe jaune qui est à l'intérieur de la première transmission.

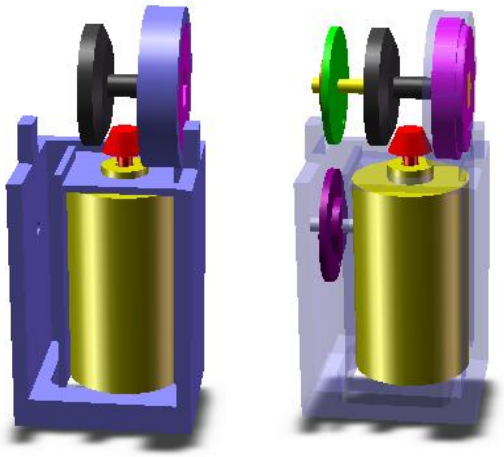


**Figure II-10 Le pignon de transmission et le réducteur harmonique**

Le mouvement de la phalange distale est piloté par le mouvement de l'articulation de la phalange moyenne à l'aide d'une barre rigide. Si nous couplons ces deux dernières directement nous aurons deux sens inversés de mouvement. Ce qui nous a mené à inverser le sens de la vitesse transmise à la phalange distale avant la transmission, et pour effectuer cette opération l'utilisation d'une courroie dentée directe (le pignon marron même vitesse que l'axe jaune vers la pièce mauve) en premier lieu et puis une autre croisée pour retransmettre la vitesse à la barre qui est soudé sur un roulement (la pièce verte)du même axe que le réducteur harmonique. Le rapport de réduction de tout le mécanisme d'inversion est de 1 : 1.

Un autre mécanisme pour l'inversion de la vitesse peut être envisagé en utilisant deux autres pignons intermédiaires fixés sur l'architecture de la phalange proximale, cela est pour remplacer la courroie croisée.

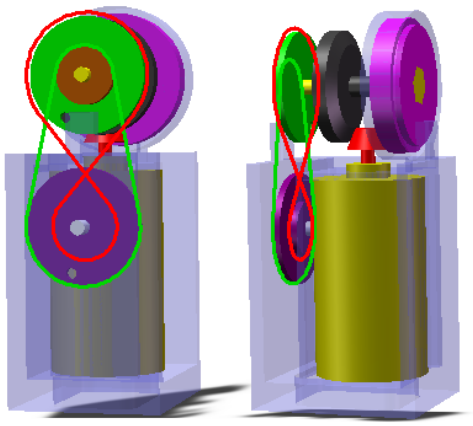


**Figure II-11 Les courroies**

La structure et les paramètres de la barre de transmission sont optimisés par la simulation.

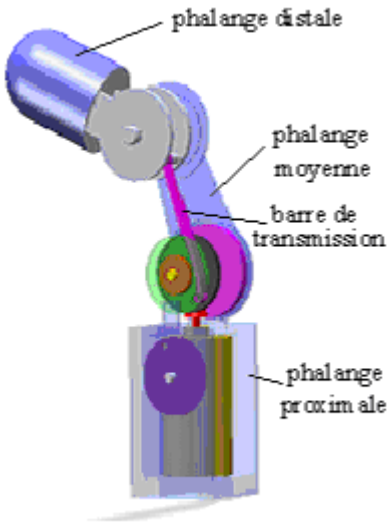



**Figure II-12 Placement de la barre de transmission**

Une fois que les phalangettes sont placées, il ne nous reste que la fixation de cette unité à l'unité de base, grâce à la structure du pignon satellitaire.

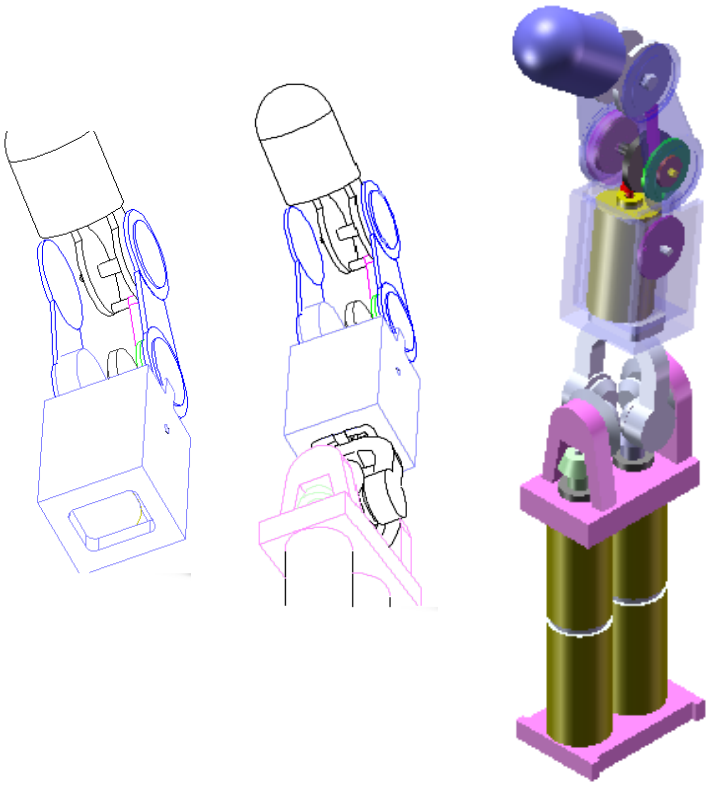


**Figure II-13 Fixation des deux unités et aspect final.**

## *II.4 La conception de la main DLR/HIT*

Le mouvement du pouce, par rapport à la paume est absolument nécessaire pour améliorer l'exécution de la saisie.

Par conséquent la main sera conçue avec un degré de liberté additionnel afin de réaliser le mouvement du pouce relatif à la paume. Les doigts seront fixés à la paume à l'aide des vis. Le pouce est équipé d'une nouvelle articulation qui précède sa base qui fera le mouvement de la pronation et la supination, en d'autre terme le mouvement de rotation si nous prenons l'avant bras comme axe de rotation. La structure de la paume permet de configurer la main soit comme une main droite ou comme une main gauche.

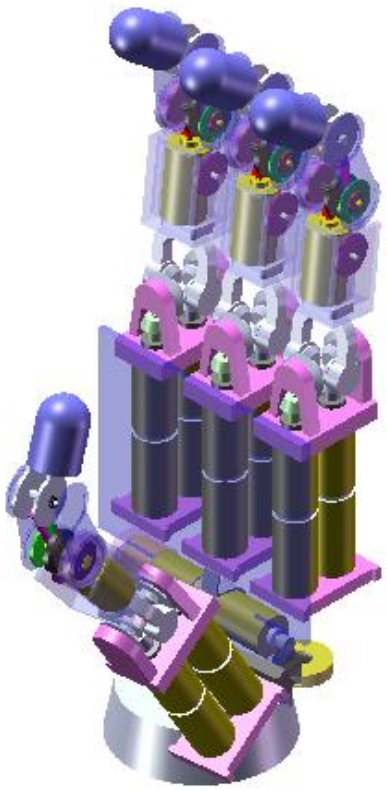


**Figure II-14 La Main DLR/HIT proposée**

## *II.5 Les caractéristiques de masse*

Nous proposons de préparer les données pour les calculs qui vont être au prochain chapitre pour le calcul du modèle dynamique. Ces données concerneront la masse, les coordonnées du centre de masse et le tenseur d'inertie. Il est à noter que SolidWorks calcule le tenseur d'inertie par rapport à son centre de masse au même coordonnées, et aussi il le calcule par rapport aux coordonnées de l'articulation appelée aussi repère de sortie, la loi de Huygens lie les deux données.

La masse totale de la main est de 1761.07 grammes (avec 373.74 grammes pour chaque doigt) son centre de masse se trouve au niveau de la paume ce qui va assurer l'équilibre. Si dans des travaux futurs, la main ferai l'objet d'une extension d'un bras manipulateur il sera important de donnée son tenseur d'inertie par rapport à son centre de masse, donnée en grammes x millimètres carrés.

| | | |
|---|---|---|
| Lxx = 12437974.55 | Lxy = -262327.77 | Lxz = 1228344.80 |
| Lyx = -262327.77 | Lyy = 5036258.19 | Lyz = -711343.32 |
| Lzx = 1228344.80 | Lzy = -711343.32 | Lzz = 10573187.92 |

### III.5.1 Caractéristiques de la base du doigt

Les caractéristiques de masse du premier corps (la base du doigt) serons nécessaires seulement dans le calcule du modèle dynamique du pouce puisque c'est la seule position où il sera en mouvement

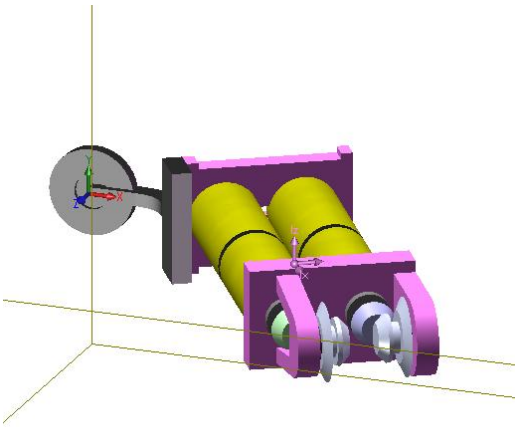


**Figure II-15 Positionnement du premier corps**

Masse = 202.26 grammes

Centre de gravité: ( millimètres )

X = 58.42

Y = -0.58

Z = 25.77

Moments d'inertie: ( grammes x millimètres carrés )

Pris au système de coordonnées de sortie.

| | | |
|---|---|---|
| Ixx = 224265.36 | Ixy = -6830.96 | Ixz = 347449.03 |
| Iyx = -6830.96 | Iyy = 962684.80 | Iyz = -3014.96 |
| Izx = 347449.03 | Izy = -3014.96 | Izz = 749236.67 |

**Remarque :** le pois des pignons satellitaires sera porté par la première articulation le fait pour lequel nous l'avons supprimé du corps. Pour avoir des coordonnées correctes il faut positionner le corps de telle sorte qu'il concorde avec les repères proposés dans le chapitre III.

### III.5.2 Caractéristique de la phalange proximale

La phalange proximale comportera le support du moteur les pignons de la partie unité du doigt, ainsi que les pignons satellitaire. Voir figure (II.16)

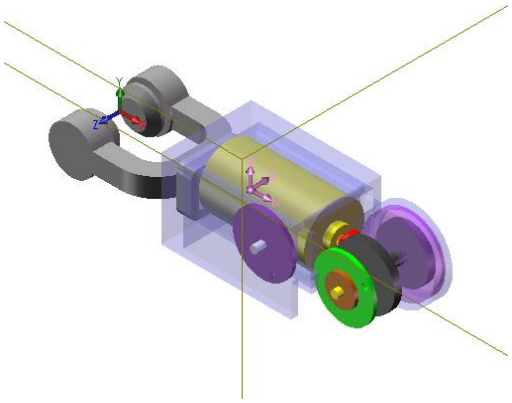


**Figure II-16 Phalange proximale**

Ses caractéristiques de masse sont :

Masse = 93.86 grammes

Centre de gravité: ( millimètres )
X = 33.32
Y = -1.94
Z = -1.30
Moments d'inertie: ( grammes * millimètres carrés )

Pris au système de coordonnées de sortie.

| | | |
|---|---|---|
| Ixx = 7279.47 | Ixy = -5585.17 | Ixz = -5479.27 |
| Iyx = -5585.17 | Iyy = 141018.41 | Iyz = 247.12 |
| Izx = -5479.27 | Izy = 247.12 | Izz = 138954.92 |

### III.5.3 Caractéristiques de la phalange moyenne

Cette partie comporte juste les parois de la phalangette et la barre de transmission que nous avons négligée par rapport au pois 3 grammes et par rapport au mouvement, sinon les moments d'inerties serons en fonction des thêtas.

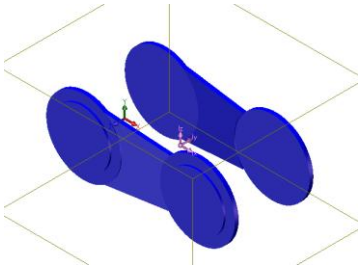


**Figure II-17 Phalange moyenne**

Caractéristiques :

Masse = 58.87 grammes

Centre de gravité: ( millimètres )

X = 14.91

Y = 1.25

Z = 0.00

Moments d'inertie: ( grammes x millimètres carrés )

Pris au système de coordonnées de sortie.

| | | |
|---|---|---|
| Ixx = 7733.12 | Ixy = 1094.95 | Ixz = 0.00 |
| Iyx = 1094.95 | Iyy = 31519.46 | Iyz = -0.00 |
| Izx = 0.00 | Izy = -0.00 | Izz = 26594.98 |

### III.5.4 Caractéristiques de la phalange distale

Elle est constituée d'un seul élément présenté comme suit :

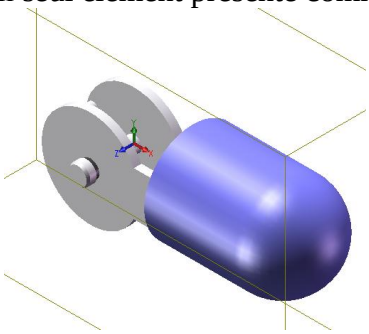


**Figure II-18 Phalange distale**

Masse = 21.07 grammes

Centre de gravité: ( millimètres )

X = 20.88

Y = 0.01

Z = 0.91

Moments d'inertie: ( grammes x millimètres carrés )

Pris au système de coordonnées de sortie.

| | | |
|---|---|---|
| Ixx = 925.35 | Ixy = -1.01 | Ixz = 224.83 |
| Iyx = -1.01 | Iyy = 11996.62 | Iyz = 0.44 |
| Izx = 224.83 | Izy = 0.44 | Izz = 11958.41 |

## *II.6 Conclusion*

La réalisation de la main DLR/HIT revient principalement à la réalisation d'un seul doigt, d'où son principe fondamental : La modularité. Le positionnement des doigts est

optimisé par simulations. La structure de la paume est symétrique ce qui permettra de la décrire soit comme une main droite ou soit comme une main gauche, en positionnant le pouce.

L'utilisation de SolidWorks nous a permis d'avoir les moments d'inerties est cela sans passer par les calculs fastidieux, d'autant plus que les structures sont assez complexes. Ainsi nous avons préparé une bonne partie pour le chapitre prochain.

# CHAPITRE III

# MODELISATION

## *III.1 Introduction*

Si on se réfère à sa définition ; la modélisation consiste à traduire les phénomènes physiques sous formes d'équations plus maniables pour réaliser la commande ; ce qui rend cette étape très importante à savoir la plus importante puisqu'elle est à la base de la commande ainsi que ses paramètres.

Plusieurs niveaux de modélisation sont possibles selon les objectifs, les contraintes de la tâche et les performances recherchées :Modèle géométrique, cinématique et dynamique.

Ces modèles peuvent être définis comme suit :

- Les modèles de transformation entre l'espace opérationnel et l'espace articulaire ; on distingue :
  - Les modèles géométriques direct et inverse qui expriment la situation de l'organe terminal en fonction des variables articulaires du mécanisme et inversement.
  - Les modèles cinématiques direct et inverse qui expriment la vitesse de l'organe terminal en fonction des vitesse articulaires et inversement.

- Les modèles dynamiques définissant les équations du mouvement du robot qui permettent d'établir les relations entre couples et forces exercés par les actionneurs et les positions, vitesses et accélérations des articulations. [11]

Une autre modélisations s'impose pour établir la commande et simuler le comportement du doigt : C'est celle des actionneurs qui sont dans notre cas des moteurs sans balai à courant continu.

## *III.2 Description de la géométrie du doigt DLR/HIT*

La cinématique du doigt de la main DLR/HIT est du type RRRR,(figure III.1) à noter que la quatrième articulation est pilotée par la troisième ce qui ramène le nombre de degrés de liberté à trois.

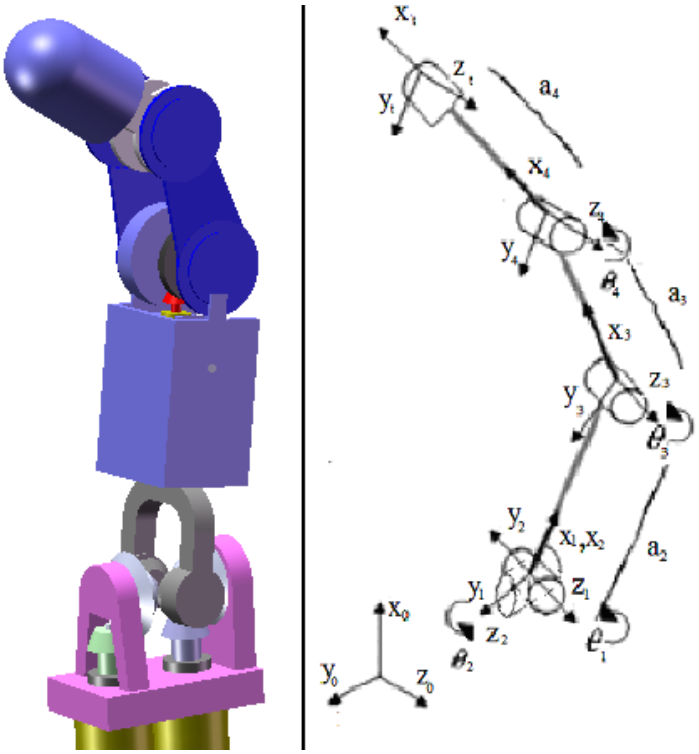



**Figure III.1 Cinématique du doigt de la main DLR/HIT**

D'un point de vue méthodologique nous plaçons d'abord les axes $z_j$ sur les axes articulaires, puis les axes $x_j$ qui seront portés par la perpendiculaire commune aux axes $z_j$ et $z_{j+1}$ et puisque dans notre cas tous les axes z sont parallèles nous avons considéré les axes x parallèle aux corps $C_j$. Les origines des repères sont prises aux centres des articulations et le repère $R_0$ est confondu avec $R_1$ aux positions du repos ; C'est-à-dire quand tous les $\theta_j$ sont

nuls. Nous déterminons ensuite les paramètres géométriques du doigt, présentés au tableau suivant :

| j | $\alpha_j$ | $d_j$ | $\theta_j$ | $r_j$ |
|---|---|---|---|---|
| 1 | 0° | 0 | $\theta_1$ | 0 |
| 2 | 90° | 0 | $\theta_2$ | 0 |
| 3 | -90° | $a_2$ | $\theta_3$ | 0 |
| 4 | 0° | $a_3$ | $\theta_4$ | 0 |
| 5 | 0° | $a_4$ | 0 | 0 |

**Tableau III.1 Paramètres géométriques du doigt de la main DLR/HIT**

| Les angles articulations (degrés) | | Les distances entre les articulations (mm) | |
|---|---|---|---|
| $\theta_1$ | [0° 90°] | $a_1$ | 0.0 |
| $\theta_2$ | [-20° 20°] | $a_2$ | 67 |
| $\theta_3$ | [0° 90°] | $a_3$ | 30 |
| $\theta_4$ | [0° 90°] | $a_4$ | 30 |

**Tableau III.2 Contraintes géométriques des paramètres géométriques**

## *III.3 Modèle géométrique direct*

Comme déjà mentionné le modèle géométrique direct permet d'exprimer les coordonnées opérationnelles de l'organe terminal, (dans notre cas c'est la fin du doigt) en fonction des coordonnées articulaires du robot dans le repère de base $R_0$. Ce qui revient à exprimer le repère $R_t$ dans le repère $R_0$. Pour cela ; nous faisons appel à la matrice de transformations homogènes notée ${}^0T_t$.

Tel que :

$$\begin{cases} {}^0T_t = {}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4\,{}^4T_t & \text{(III.3.1)} \\ \quad et \\ {}^{j-1}T_j = Rot(x,\alpha_j)\ Trans(x,d_j)\ Rot(z,\theta_j)\ Trans(z,r_j) \qquad 0 \le j \le 5 \end{cases}$$

calcul de ${}^0T_t$

$${}^0T_t = \begin{pmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} C2 & -S2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S2 & C2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} C3 & -S3 & 0 & 67 \\ 0 & 0 & 1 & 0 \\ -S3 & -C3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} C4 & -S4 & 0 & 0 \\ S4 & C4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 30 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Notre système est à 3 degrés de liberté nous n'effectuerons que la commande en position, l'orientation ne sera qu'observée et est représentée par le cosinus directeur de la

matrice ${}^0T_t$ (les trois premières lignes et les trois première colonnes de ${}^0T_t$). La vecteur position est représenté par la 4ème colonne de ${}^0T_t$.

$$\begin{cases} s_x = C1C2C34 - S1S34 \\ s_y = S1C2C34 + C1S34 \\ s_z = S2C34 \end{cases}$$

$$\begin{cases} n_x = -C1C2S34 - S1C34 \\ n_y = -S1C2S34 + C1C34 \\ n_z = -S2S34 \end{cases} \quad (III.3.2)$$

$$\begin{cases} a_x = -C1S2 \\ a_y = -S1S2 \\ a_z = C2 \end{cases}$$

$$\left.\begin{array}{l} Px = 30C1C2C34 - 30S1S34 + 30C1C2C3 - 30S1S3 + 67C1C2 \\ Py = 30S1C2C34 + 30C1S34 + 30S1C2C3 + 30C1S3 + 67S1C2 \\ Pz = 30S2C34 + 30S2C3 + 67S2 \end{array}\right\} (III.3.3)$$

*Et : $Ci = \cos(\theta_i)$ , $Si = \sin(\theta_i)$, $Cij = \cos(\theta_i + \theta_j)$, $Sij = \sin(\theta_i + \theta_j)$*

**Remarque:**

Les calculs sont effectués par l'intermédiaire du fichier transformation.m (Matlab) ci-joint dans le document numérique accompagné. Vous pouvez exécuter la commande dlrhitdemo.

## *III.4 Modèle cinématique*

Le modèle cinématique direct d'un robot manipulateur décrit les vitesses des coordonnées opérationnelles en fonction des vitesses articulaires il est noté :

$$\dot{X} = J(q)\ \dot{q} \quad (III.4.1)$$

Ou J(q) désigne la matrice jacobienne égale à dX/dq (III.4.2)

**Calcul de la matrice jacobienne**

On peut obtenir la matrice jacobienne par une méthode de calcul direct, fondée sur la relation entre les vecteurs des vitesses de translation et de rotation $V_n$ et $\omega_n$.

$$\dot{X} = \begin{bmatrix} V_n \\ \\ \omega_n \end{bmatrix} = J_n\ \dot{q} \quad (III.4.3)$$

La matrice J est de dimension 6x4, les trois premières lignes sont obtenues en dérivant l'expression du vecteur P(Expression III.3.3) par rapport aux variables articulaires $q_1$,$q_2$,$q_3$ et $q_4$. Les trois dernières lignes sont obtenues par colonne décrivant le vecteur a de chacune des matrices ${}^0T_1$, ${}^0T_1{}^1T_2$, ${}^0T_1{}^1T_2{}^2T_3$ et enfin ${}^0T_1{}^1T_2{}^2T_3{}^3T_4$, dans l'ordre. Les calculs sont effectués par des fichiers.m sous Matlab donnés en annexe ainsi que la matrice jacobienne J(q) à cause de l'importance volumique de ses paramètres.

## *III.5 Modèle cinématique inverse*

L'objectif de ce modèle est de calculer à partir d'une configuration q donnée, les vitesses articulaires $\dot{q}$ assurant au repère terminal une vitesse opérationnelle $\dot{X}$ imposée. Pour obtenir le modèle cinématique inverse on inverse le modèle cinématique direct.

### III.5.1 Calcul du modèle cinématique inverse

Nous avons constaté que le volume de l'expression de la matrice jacobienne est très important du fait la recherche de la solution numérique nous paraissait plus adéquate pour ce genre de problème. A noté que la matrice J(q) n'est pas inversible raison pour laquelle nous avons eu recours à la pseudo inverse de J(q).De la relation (III.4.1)

$$\dot{X} = J(q)\ \dot{q} \text{ : la matrice J(q) est de dimension 6x4}$$

$$J'(q)\ \dot{X} = J'(q)J(q)\ \dot{q} \Rightarrow \dot{q} = (J'(q)J(q))^{-1}J'(q)\ \dot{X}$$

$$\text{On note } J^+(q) = (J'(q)J(q))^{-1}J'(q) \qquad \text{(III.5.1)}$$

La matrice $J^+(q)$ est appelée la pseudo inverse de J(q), le modèle cinématique inverse peut s'écrire sous la forme :

$$\dot{q} = J^+(q)\ \dot{X} \qquad \text{(III.5.2)}$$

Comme pour la matrice jacobienne J(q), la pseudo inverse $J^+(q)$ demande des calculs fastidieux, vous pouvez consulter le document numérique.

### III.5.2 Calcul du modèle géométrique inverse

Parmi les intérêts du modèle cinématique inverse est le calcul du modèle géométrique inverse dont la solution analytique parait compliquée à trouver à savoir impossible. De la relation (III.5.2) nous avons :

$$q = J^+(q)\ \dot{X} \Leftrightarrow dq/dt = J^+(q)\ \dot{X}$$

$$dq/dt = J^+(q)\ \dot{X} \Rightarrow (q(i+1)-q(i))/\Delta t = J^+(q)\ \dot{X}$$

$$q(i+1) = q(i) + \Delta t\, J^{+}(q)\, \dot{X} \qquad \text{(III.5.3)}$$

On suppose que la vitesse opérationnelle est donnée, sinon on la calcule tel que :

$$\dot{X} = dX/dt \Rightarrow \dot{X} \equiv (X(i)-X(i-1))/\Delta t \qquad \text{(III.5.4)}$$

Cette méthode permet le calcul des solutions locales des variables articulaires q connaissant les coordonnées opérationnelles X.

**Remarque**

Pour chaque itération on doit imposer $q_4=q_3$ pour satisfaire la contrainte du pilotage de $q_4$ par $q_3$.

## *III.6 Modèle dynamique direct*

Le modèle dynamique est la relation entre les couples (et ou forces) appliqués aux actionneurs et les positions, vitesses et accélération articulaires [11]; le modèle dynamique peut s'écrire sous la forme :

$$\Gamma = f(q, \dot{q}, \ddot{q}, \Gamma_e) \qquad \text{(III.6.1)}$$

### III.6.1 Calcul du Modèle dynamique direct

Il existe plusieurs formalismes utilisés pour la détermination du modèle dynamique dont les plus utilisés sont le formalisme de Lagrange et le formalisme de Newton-Euler.

La méthode de Lagrange est considérée comme étant la plus simple la raison pour laquelle nous l'avons adopté, son principal avantage, est que les termes obtenus dans les équations finales du modèle dynamique ont une signification physique (inerties, frottements, gravité, forces de Coriolis et forces centrifuges)

Ce formalisme décrit les équations du mouvement en terme de travail et d'énergie du système par l'équation suivante :

$$\frac{d}{dt}\frac{\partial}{\partial \dot{q}_i} L(q,\dot{q}) - \frac{\partial}{\partial q_i} L(q,\dot{q}) = F_i \qquad 1 \le i \le n \qquad \text{(III.6.2)}$$

avec :

- L : lagrangien du système égale à T-U (III.6.2.a)
- T : Energie cinétique totale du système
- U : Energie potentielle totale du système
- $F_i$ : La force généralisée agissant sur la $i^{ème}$ articulation, $\Gamma = F_i$ *; sans frottement.*

*Le couple Γ peut se mettre à partir des équations (III.6.2) et (III.6.2.a) sous la forme :*

$$\Gamma = D(q)\ \ddot{q} + C(q,\dot{q})\,\dot{q} + Q(q) \qquad \text{(III.6.3)}$$

D(q) : Matrice d'inertie du système

C(q, $\dot{q}$ ) : Matrice des forces /couples de Coriolis et de forces centrifuges.

Q(q) : vecteur des forces de gravité

Nous pourrons calculer les éléments de la matrice C(q,q) à partir du symbole de Christophell $c_{i,j,k}$ tel que :

$$\begin{cases} C_{ij} = \sum_{k=1}^{n} c_{ijk}\dot{q}_k \\ c_{ijk} = \frac{1}{2}\left[\frac{\partial A_{ij}}{\partial q_k} + \frac{\partial A_{ik}}{\partial q_j} - \frac{\partial A_{jk}}{\partial q_i}\right] \end{cases} \qquad \text{(III.6.4)}$$

Les éléments du vecteur Q se calculent en dérivant U, l'énergie potentielle par rapport à q :

$$Q_i = \frac{\partial U}{\partial q_i} \qquad \text{(III.6.5)}$$

### III.6.1.1 Détermination de la matrice d'inertie D

Le calcul de la matrice d'inertie revient au calcul de l'énergie cinétique, qui est considérée comme étant la partie la plus difficile à déterminer pour la modélisation des robots, prenons par exemple le schéma suivant :

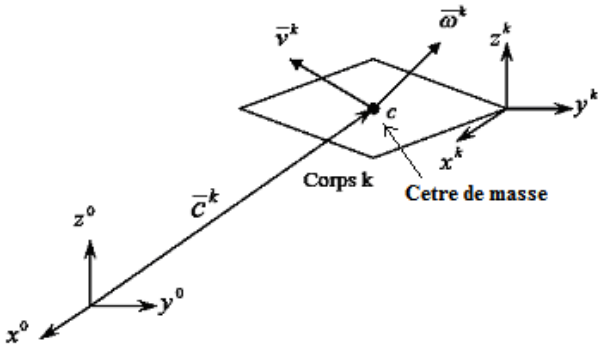



**Figure III.2 Mouvement du corps k**

L'énergie cinétique totale peut s'écrire sous la forme suivant :

$$T(q,\dot{q}) = \sum_{k=1}^{n} \frac{\left(\bar{v}^k\right)^T \cdot m_k \cdot \bar{v}^k + \left(\bar{\omega}^k\right)^T \cdot D_k \cdot \bar{\omega}^k}{2} \qquad \text{(III.6.6)}$$

$v_k$ : Vitesse de translation de c par rapport à $R_0$. de dimension (3,1)

$\omega_k$ : Vitesse de rotation autour de c par rapport à $R_0$. de dimension (3,1)

$m_k$ : Masse du corps k.

$D_k$ : La matrice d'inertie du corps k autour de son centre de masse, exprimée dans le repère absolu $R_0$.

Le but de cette étape est d'exprimer cette expression en fonction de q et q, les vitesses $v_k$ et $\omega_k$ *sont fonction des vitesses articulaires q en fonction de la* $k^{ième}$ *jacobienne :*

$$\begin{pmatrix} v_k \\ \\ \omega_k \end{pmatrix} = J_k(q)\ \dot{q} \qquad \text{(III.6.7)}$$

$J_k$ est calculée de la même façon que pour le modèle cinématique tel que nous prenons à chaque k centre de gravité comme fin de la chaine du modèle. Nous notons aussi :

$$J_k = \begin{pmatrix} A_k(q) \\ \\ B_k(q) \end{pmatrix} \qquad \text{(III.6.8)}$$

Nous obtenons le tenseur d'inertie par rapport au repère $R_0$ de l'expression :

$$D_k(q) = R_0^k(q)\,\bar{D}_k\left[R_0^k(q)\right]^T \qquad \text{(III.6.9)}$$

$\bar{D}_k$ : Le tenseur d'inertie du corps k par rapport à son centre de masse exprimé dans le repère $R_k$. Il est directement donné par Solidworks dans notre étude ce qui nous évitera son calcul qui serait très fastidieux vu la complexité des pièces du modèle.

A ce moment nous pourrons noter l'énergie cinétique totale sous l'expression suivante :

$$T = \frac{1}{2}\dot{q}^T D\,\dot{q} \qquad \text{(III.6.10)}$$

et :

$$D(q) \triangleq \sum_{k=1}^{n} \left\{ \left[A^k(q)\right]^T m_k A^k(q) + \left[B^k(q)\right]^T D_k B^k(q) \right\} \qquad \text{(III.6.11)}$$

### III.6.1.2 Détermination de l'énergie potentielle

L'énergie potentielle emmagasinée dans le k$^{ème}$ corps d'un manipulateur est la quantité de travail nécessaire pour déplacer le centre de masse du corps k en présence de la force de gravité.

L'énergie potentielle totale est donc donnée par :

$$U(q) = \sum_{k=1}^{n} m_k \cdot g^T \cdot \overline{c}^k(q) \qquad \text{(III.6.12)}$$

$\overline{c}^k(q)$ : Sont les coordonnées du centre de masse du corps k dans le repère $R_0$.

**Remarque :** Tous les calculs sont effectués par Matlab, à l'aide du fichier dynadlrhit.m ci-joint dans le document numérique. Vous pouvez l'exécuter à partir de la commande dlrhitdemo.

## III.6.2 Prise en compte des frottements

De nombreuses études ont été réalisées afin de mieux analyser les frottement au niveau des articulations, des réducteurs et des transmissions, les frottements non compensés provoquent en effets des erreurs statiques, des retards et des cycles limites.[11-p229].

Des textes expérimentaux ont prouvé que le couple de frottement au démarrage décroit exponentiellement à faible vitesse. Puis croit ensuite proportionnellement à la vitesse. Le modèle correspond est donné par :

$$b_k(\dot{q}) = b_k^v \dot{q}_k + sign(\dot{q}_k)\left[ b_k^d + \left(b_k^s - b_k^d\right)\exp\frac{-|\dot{q}_k|}{\varepsilon}\right] \quad 1 \le k \le n \qquad \text{(III.6.13)}$$

## III.6.3 Prise en compte des efforts exercés par l'organe terminal

Le couple ou la force que doit fournir un actionneur pour que l'organe terminal d'un robot puisse exercer un effort statique $F_{en}$ sur l'environnement s'écrie :

$$\Gamma_e = J_n^T F_{en} \qquad \text{(III.6.14)}$$

On ajoute ce terme $\Gamma_e$ au deuxième membre de l'expression (III.6.3)

## *III.7 Relation entre couples articulaires et couples moteurs*

Notre doigt comporte en premier lieu un couplage au niveau de la double articulation $q_1$ et $q_2$ et une transmission au niveau de l'articulation représentée par $q_4$ en

deuxième lieu. Et pour remédier à ce problème nous utiliserons le principe des travaux virtuels pour passer d'un espace à l'autre. Le travail virtuel des actions s'écrit après changement de coordonnées :

$$\Gamma^T dq = \tau_m{}^T dq_m \qquad \text{(III.7.1)}$$

$q_m$ représente le vecteur des variables moteurs et $\tau_m$ celui des couples moteurs, les relations entre les deux systèmes peuvent aussi être exprimées par :

$$dq = J_{qm}\, dq_m \qquad \text{(III.7.2)}$$

$J_{qm}$ est le jacobien de q par rapport à $q_m$ égale à $\frac{\partial q}{\partial q_m}$

$$q_1 = \frac{q_{m1} - q_{m2}}{N_1}$$

$$q_2 = \frac{q_{m1} + q_{m2}}{N_1} \quad => \quad J_{qm} = \begin{pmatrix} 1/N_1 & -1/N_1 & 0 \\ 1/N_1 & 1/N_1 & 0 \\ 0 & 0 & 1/N_3 \\ 0 & 0 & 1/N_3 \end{pmatrix}$$

$$q_3 = \frac{q_{m3}}{N_3}$$

$$q_4 = \frac{q_{m3}}{N_3}$$

*N1= 159 x 2 et N3= 100*

A partir des relations (III.7.1) et (III.7.2) on déduit que :

$$\tau_m = J_{qm}{}^T \Gamma \qquad \text{(III.7.3)}$$

## *III.8 Modélisation des moteurs*

IL y a principalement deux types de moteurs à courant continu utilisés dans l'industrie, Le premier est le moteur conventionnel à courant continu où le flux est produit par le stator à l'excitation des enroulements, qui se trouvent au niveau du rotor, du fait ce dernier se met à tourner, des bagues sont utilisées pour assurer l'alimentation. Le deuxième type est le moteur sans bagues (sans balais, BLDC motor) dont les enroulements sont au niveau du stator ; à l'excitation du stator par un courant i le rotor crée un flux magnétique qui le met à tourner puisque le stator est fixé, par conséquent l'utilisation des bagues sera suspendue.

Non seulement la suspension des bagues diminue l'entretien mais aussi les moteurs sans balais ont des avantages plus tentant tel qu'un plus petit volume, une force élevée et une structure plus simple, Ainsi ils sont largement utilisés dans les secteurs qui ont besoin d'une commande à rendement élevé tels que : La robotique, les imprimantes laser les disques durs, etc.

Les caractéristiques dynamiques des moteurs sans balais sont similaires à celles des moteurs conventionnels à courant continu, dont les équations sont représentées comme suit :

$$v_{app}(t) = L\frac{di(t)}{dt} + R.i(t) + v_{emf}(t) \quad \text{(III.8.1)}$$

$$v_{emf} = K_b.\omega(t) \quad \text{(III.8.2)}$$

$$T(t) = K_t.i(t) \quad \text{(III.8.3)}$$

$$T(t) = J\frac{d\omega(t)}{dt} + D.\omega(t) \quad \text{(III.8.4)}$$

Tel que :

| | |
|---|---|
| $v(t)_{app}$ | est la tension appliquée |
| $\omega\ (t)$ | est la vitesse du moteur |
| L | est l'inductance du stator |
| $i(t)$ | est le courant du circuit |
| R | est la résistance du stator |
| $v(t)_{emf}$ | est la force électromotrice |
| T | est le couple du moteur |
| D | est le coefficient visqueux |
| J | est le moment d'inertie |
| $Kt$ | est la constante du couple du moteur |
| Kb | est la constante de la force électromotrice. |

La figure (III.2) représente le schéma fonctionnel du moteur ; dont la fonction du transfère est :

$$\frac{\omega(s)}{v_{app}(s)} = \frac{K_t}{LJ.S^2 + (LD + RJ)S + K_tK_b} \quad \text{(III.8.5)}$$

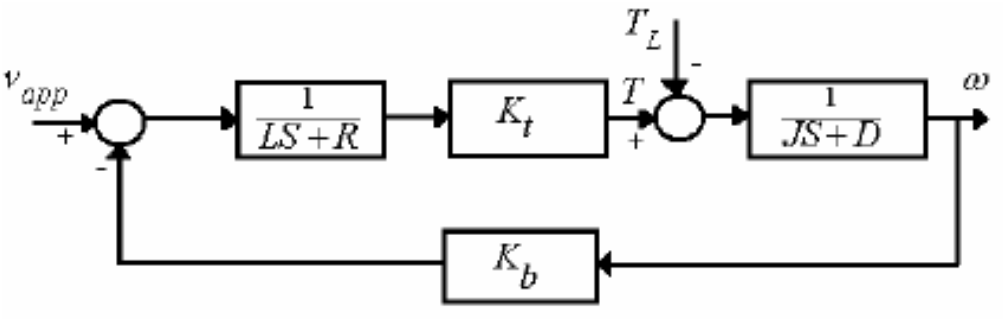



**Figure III.3 Schéma bloc d'un moteur sans bagues à courant continu (BLDC motor)**

## *III.9 Conclusion*

L'utilisation de l'arbre de transmission a augmenté le degré de difficulté des calculs, ce qui nous a mené à utiliser la programmation pour établir les différents modèles qui seront utilisés pour la commande du robot qui fera l'objet du prochain chapitre. Le modèle géométrique direct et indirect et le modèle cinématique direct et indirect ont été validés sous Matlab.

# CHAPITRE IV
# LA COMMANDE

## *IV.1 Introduction*

Commander la main DLR/HIT revient à commander les moteurs qui la gèrent, rappelons que ces moteurs sont des moteurs sans bagues à courant continu. Nous avons choisi une commande en position réalisée par un contrôleur PID classique.

Ainsi nous pouvons réaliser plusieurs tâches de position telle que l'ouverture et la fermeture de la main et la saisie d'objets de diverses formes géométriques, à condition de connaitre les positions exactes de ces objets.

Pour se faire, la génération des trajectoires désirées sera recommandable, et fera l'objet du prochain point.

## *IV.2 Génération de trajectoires*

Le mouvement peut être généré dans les deux espaces, articulaire ou opérationnel, chaque approche peut être choisie pour des raisons bien distinguées qui sont définies par le choix des tâches à effectuer par le robot. Analysons maintenant ces deux approches :

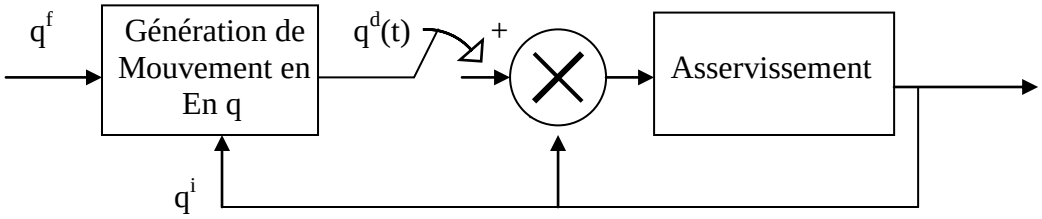



**Figure IV.1 Génération de mouvement dans l'espace articulaire**

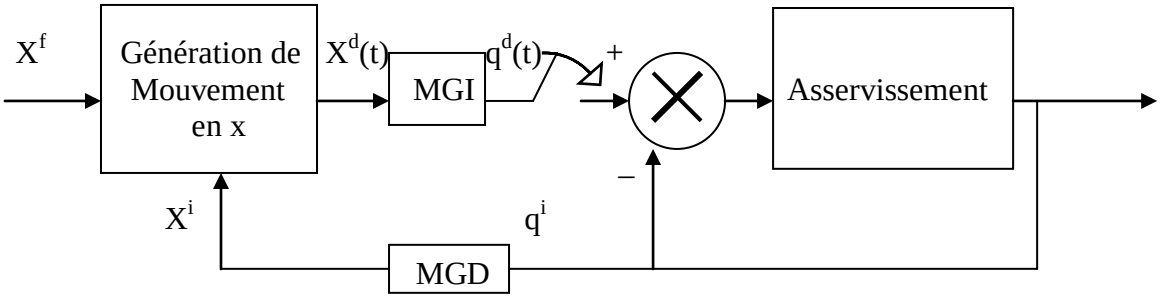



**Figure IV.2 Génération de mouvement dans l'espace opérationnel**

La génération de mouvement dans l'espace opérationnel fait appelle aux deux modèles, le modèle géométrique direct et le modèle géométrique indirect qui, dans notre cas, fera appel au modèle cinématique indirect, par conséquent le volume de calcul sera plus important, la raison pour laquelle nous avons opté pour la génération du mouvement dans l'espace articulaire, pour plus de détail consulter [11].

### VI.2.1 Génération de mouvement entre deux points dans l'espace articulaire

Le mouvement de $q^i$ et $q^f$ en fonction du temps t est décrit par l'équation suivante :

$$q(t) = q^i + r(t)\, D \qquad \text{pour } 0 \le t \le t_f \qquad \text{VI.2.1}$$

$$\dot{q}(t) = \dot{r}(t)\, D \qquad \text{VI.2.2}$$

avec $D = q^f - q^i$

La fonction r(t) est appelée fonction d'interpolation, les valeurs aux limites de cette fonction sont données par :

$$\begin{cases} r(0) = 0 \\ r(t_f) = 1 \end{cases}$$

pour assurer la continuité des accélérations afin d'éviter d'exciter la mécanique donc nous ajoutons quatre autre contraintes :

$$\begin{cases} \dot{q}(0) = 0 \\ \dot{q}(t_f) = 0 \end{cases} \qquad \begin{cases} \ddot{q}(0) = 0 \\ \ddot{q}(t_f) = 0 \end{cases}$$

Nous avons choisi r(t) comme étant un polynôme ; et avec ces six conditions à satisfaire r(t) sera de degré cinq.

On peut démontrer que :

$$r(t) = 10\left(\frac{t}{t_f}\right)^3 - 15\left(\frac{t}{t_f}\right)^4 + 6\left(\frac{t}{t_f}\right)^5 \qquad \text{VI.2.3}$$

$\theta^i = [0 \;\; 0 \;\; 0 \;\; 0]$

$\theta^f = [\pi/3 \;\; \pi/12 \;\; \pi/6 \;\; \pi/6]$

$t_f = 10$ s, et un $\Delta t = 0.04$.

Les trajectoires désirées sont illustrées ci dessous.

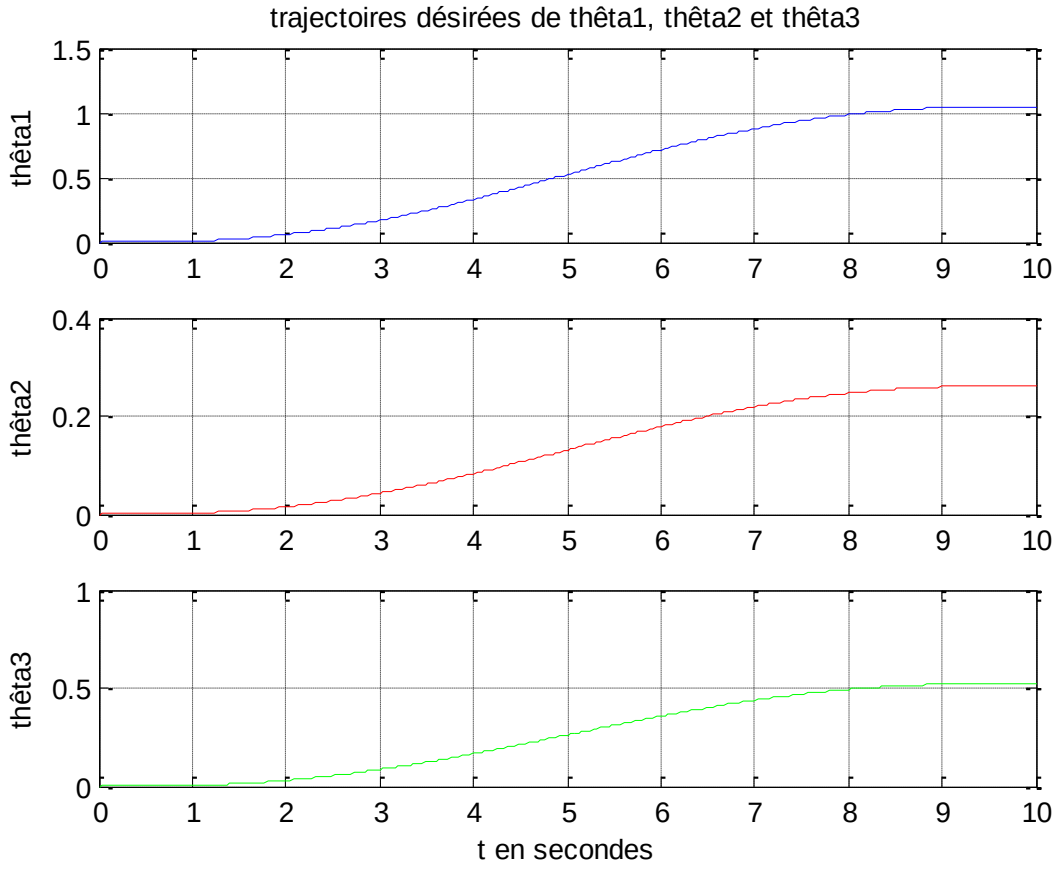



**Figure IV.3 Profil des positions articulaires désirées**

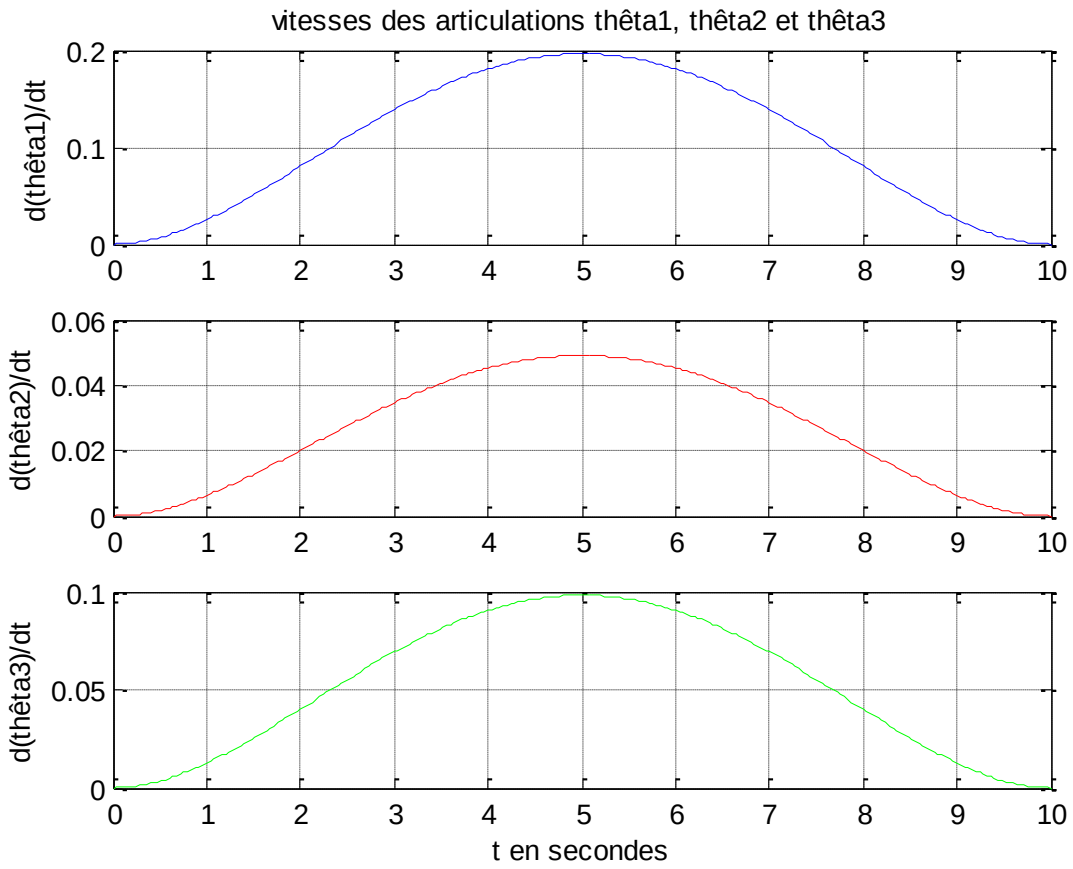



**Figure IV.4 Profil des vitesses articulaires désirées**

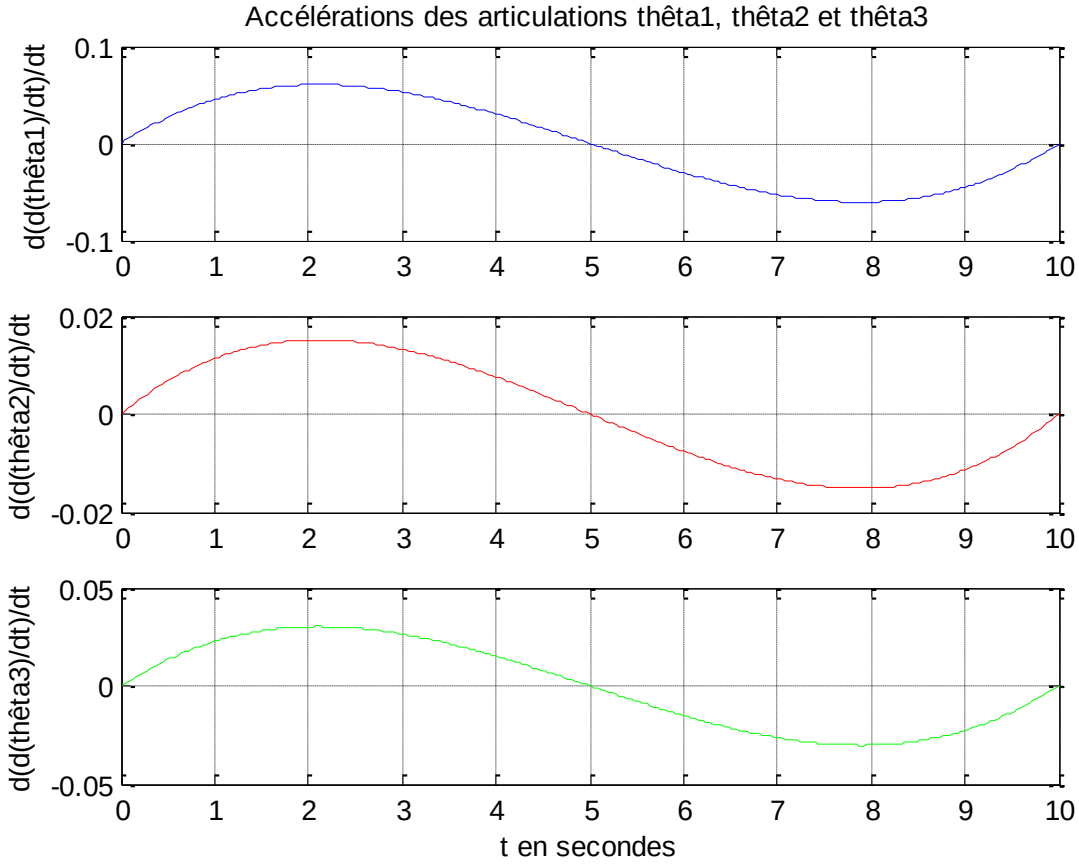



**Figure IV.5 Profil des accélérations articulaires désirées**

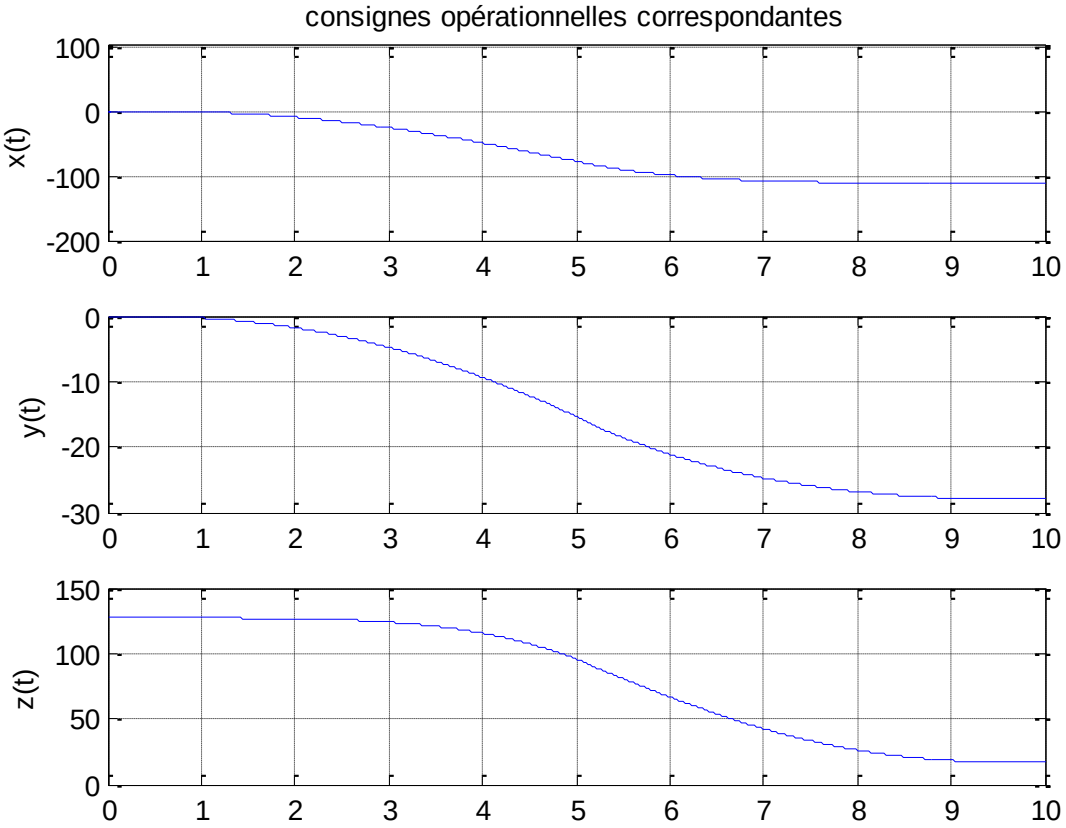



**Figure IV.6Trajectoires opérationnelles correspondantes**

## *VI.3 La commande classique par PID décentralisée*

La commande décentralisée est la commande la plus utilisée dans le monde industriel [11], pour sa simplicité d'implémentation et son faible temps de calcul. Le terme décentralisé implique que chaque actionneur est commandé à part, sans tenir compte des autres actionneurs. Son principe de commande peut être illustré comme montré au schéma suivant :

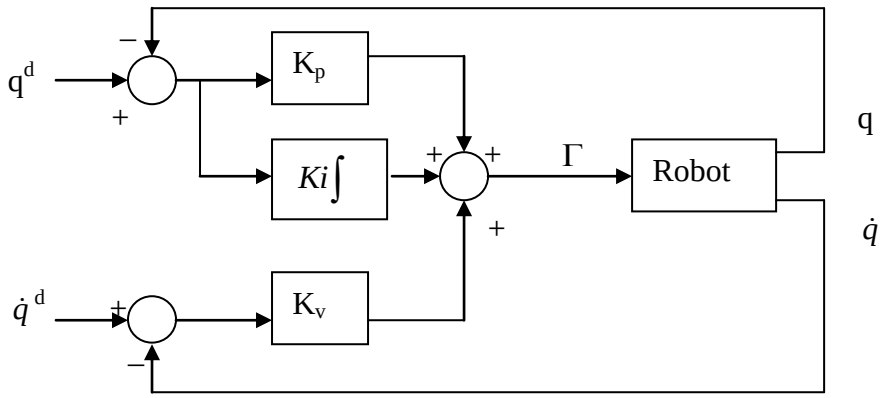



**Figure IV.7 Schéma classique d'une commande PID**

Où $\dot{q}^d$ et $q^d$ désignent vitesse et position désirées dans l'espace articulaire et où Kv, Ki et $K_p$ sont des matrices diagonales définies positives.

**Implémentation de la commande**

Pour que la simulation reflète le plus possible le comportement de tout le mécanisme, nous avons introduit les couples de charge de chaque moteur, ce couple est calculé à l'aide de l'expression dynamique est sera multiplié par l'inverse du rapport de réduction de chaque actionneur. Le suivi des trajectoires est très satisfaisant comme le montre la figure suivante.

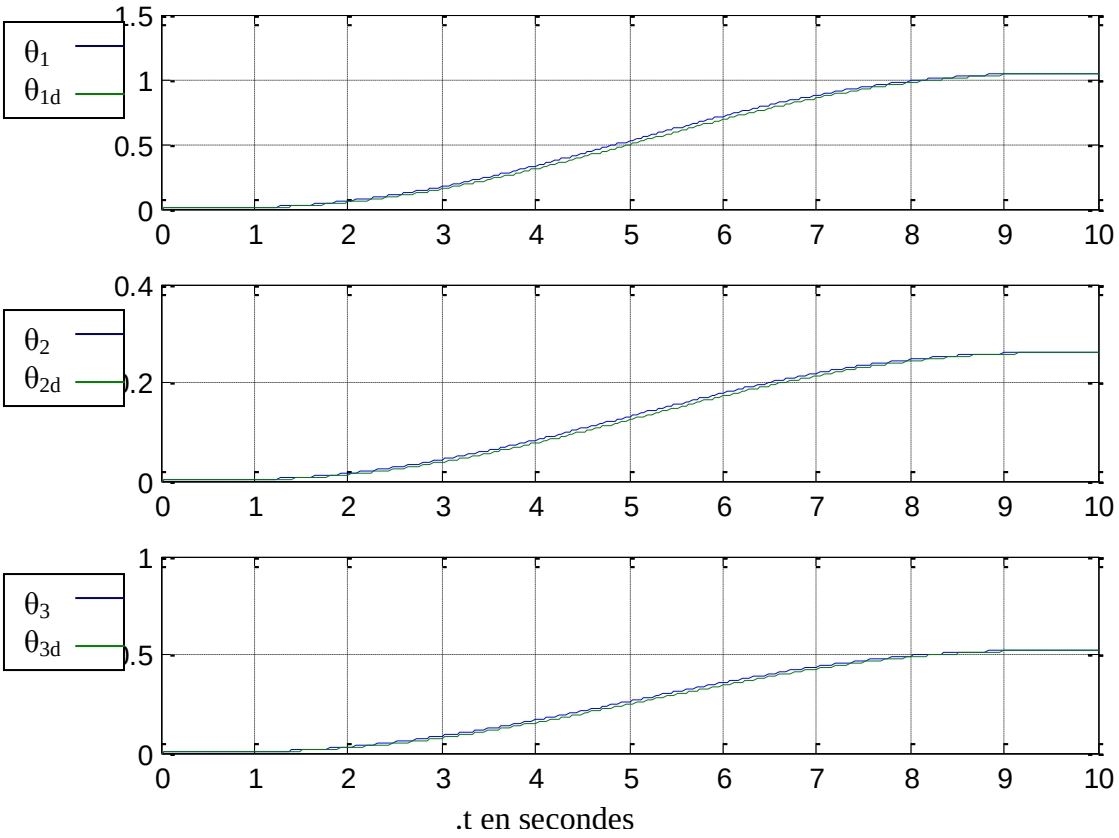



**Figure IV.8 Trajectoires des positions articulations par rapport aux trajectoires désirées**

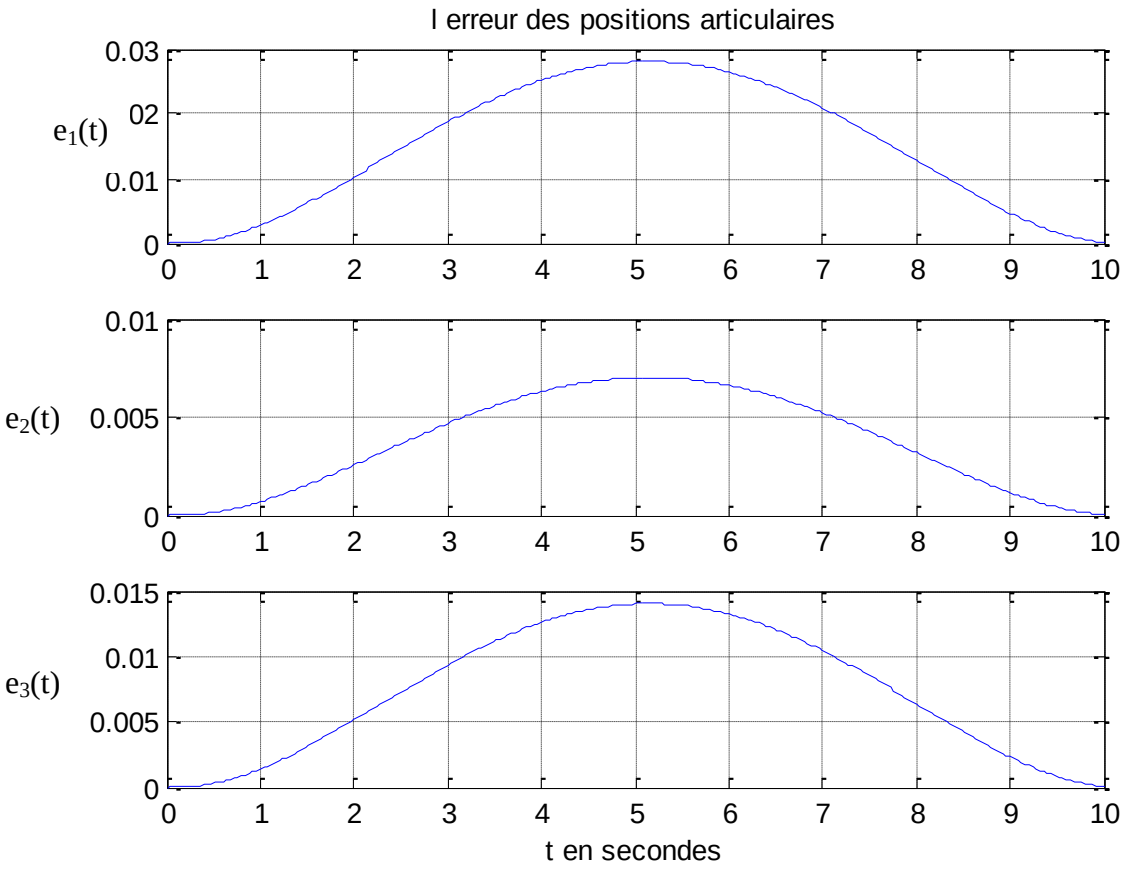



**Figure IV.9 L'erreur des positions articulaires**

L'erreur de la trajectoire par rapport à la trajectoire des moteurs désirée définie aussi la courbe de la commande à un facteur prés, qui peut être introduit au niveau des cartes de puissance des moteurs.

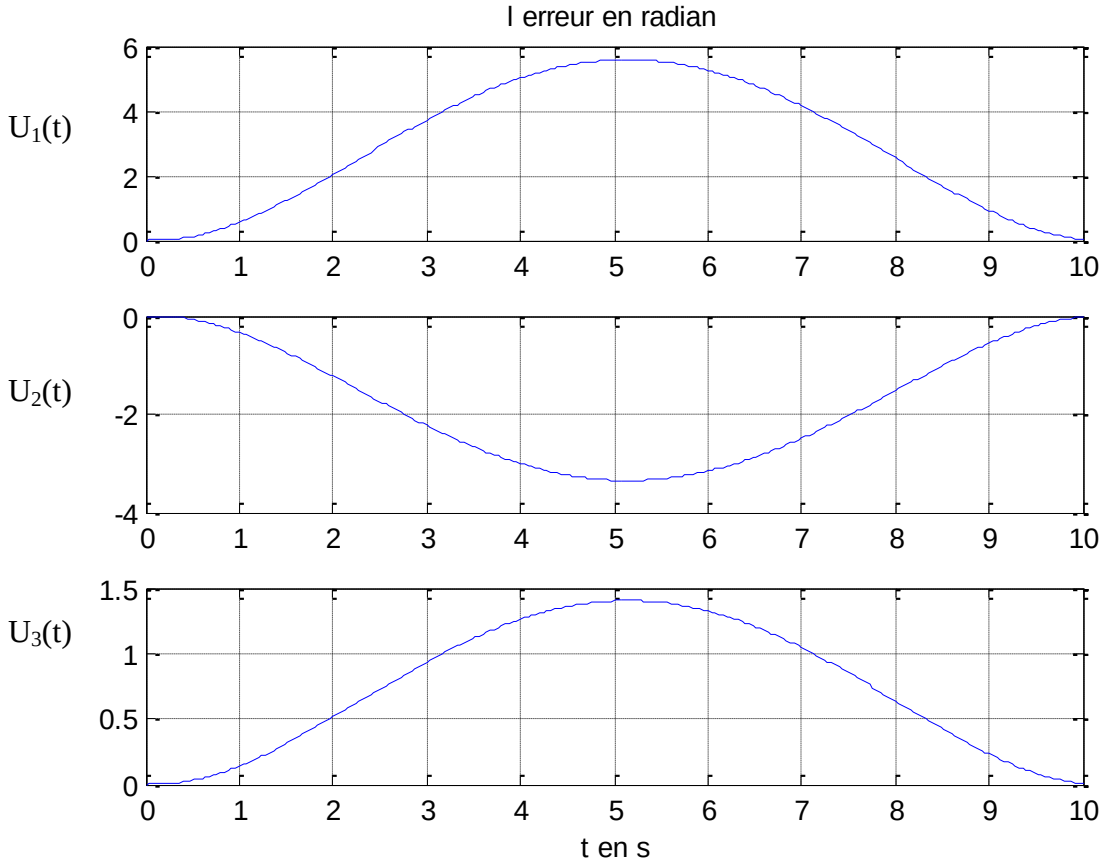



**Figure IV.10 les commandes à introduire au système**

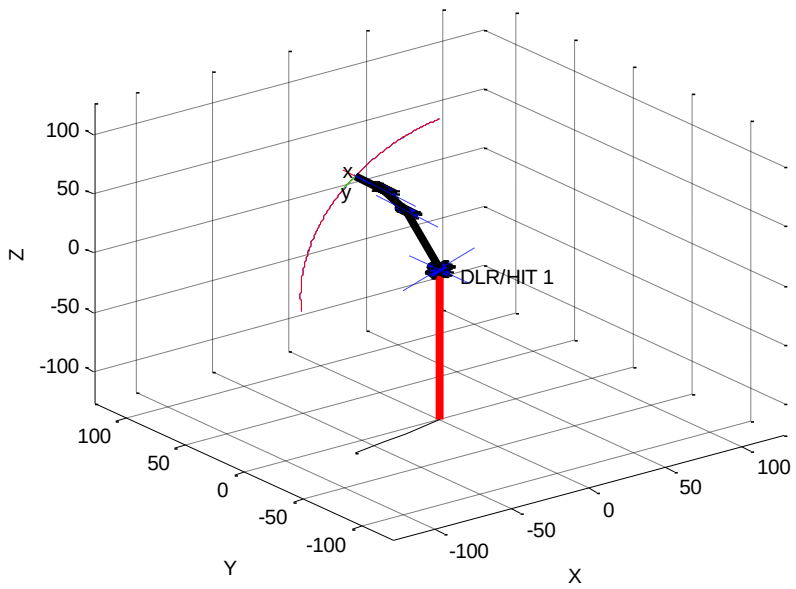



**Figure IV.11 Trajectoire dans le domaine opérationnel**

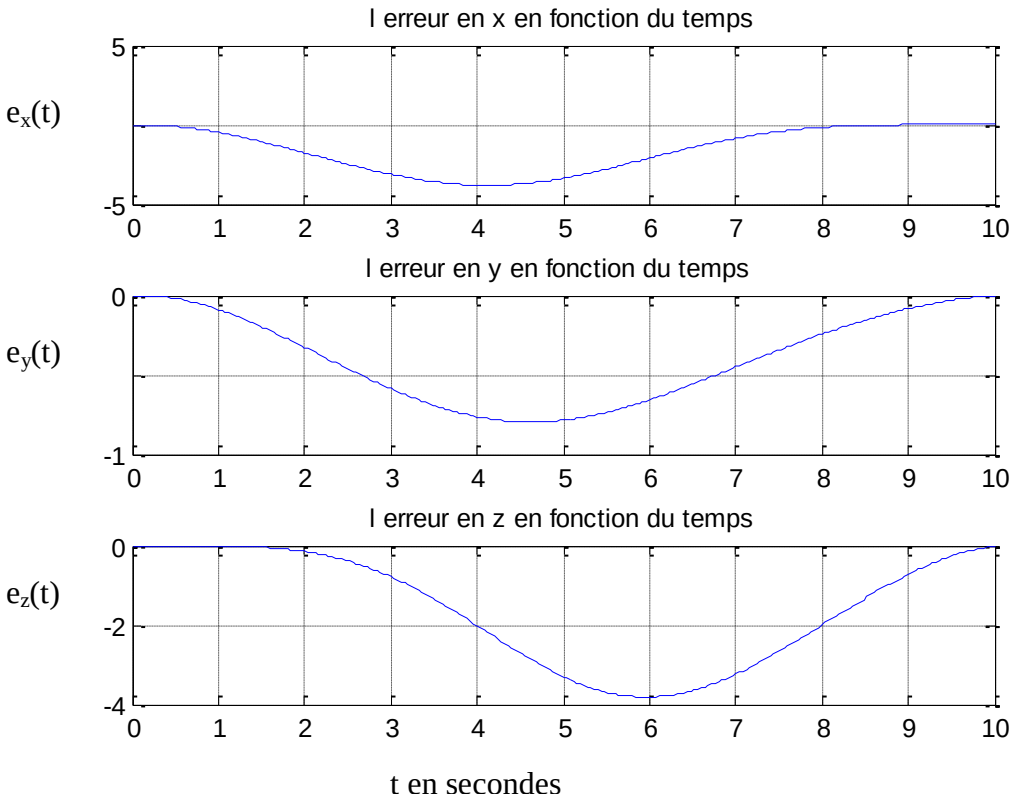



**Figure IV.12 L'erreur dans le domaine opérationnel**

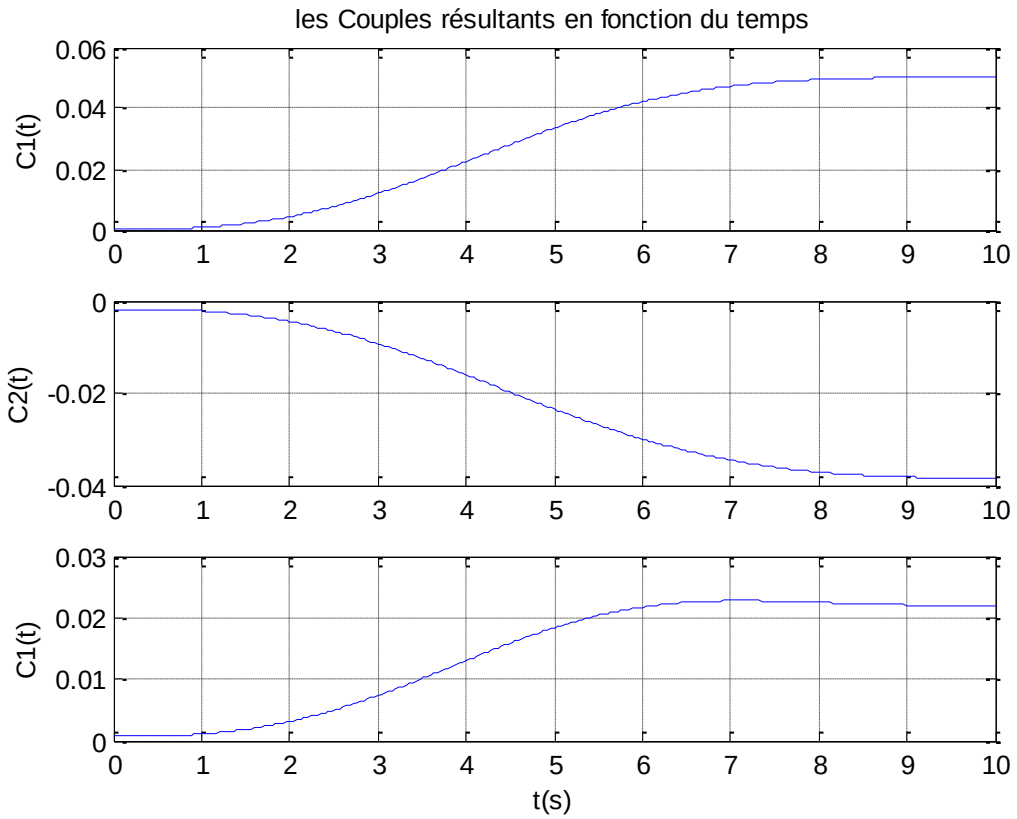



**Figure IV.13 Les couples résultants au niveau des articulations**

Nous avons pris $K_p = \begin{bmatrix} 10000,0,0 \\ 0,10000,0 \\ 0,0,100000 \end{bmatrix}$, $K_v$ est Ki sont des matrices nulles.

**Remarque**

Vous pouvez visualiser ces résultats en tapant la commande dlrhitdemo sous Matlab.

## *IV.4 Conclusion*

Nous avons bien vu que la commande classique par PID donne de très bons résultats pour une simple commande en position. Nous proposons, pour d'autres travaux, d'appliquer des commandes en efforts et cela pour la réalisation des saisies ou le contact avec l'environnement.

# CHAPITRE V

# ELECTRONIQUE ET IMPLEMENTATION

## V.1 Introduction

Le but de ce chapitre est d'embarquer la commande sur une carte électronique, Pour ce faire, nous proposons le passage par trois étapes. La première concerne les différentes configurations matérielles où l'on doit définir les entrées sorties (les capteurs et signaux de commande), et justifier aussi le choix de la carte de commande.

Nous appelons la deuxième étape, Architectures de la partie matérielle et l'outil de programmation API, plus précisément l'API de SolidWorks ainsi la communication entre la commande matérielle et la main virtuelle sera possible.

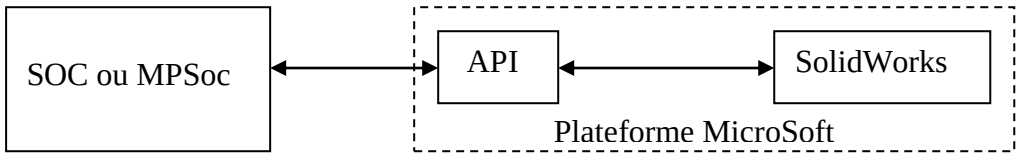



**Figure V.1 Approche de conception**

La troisième partie fera l'objet de la mise en œuvre ; où on configure d'abord la carte, puis on définit la tâche par une lecture d'un tableau contenant des angles θ déjà simulés

sous Matlab chargé dans la carte de commande et qui sera traduit par la suite, par des mouvements des doigts de la main sous SolidWorks via l'API. La figure V.1 illustre notre approche de conception.

## V.2 Première partie : Configuration matérielle

Cette étape comportera l'étude théorique des différentes architectures électroniques proposées, qu'on aura besoin pour définir les outils qui vont être utilisés pour l'implémentation.

### V.2.1 Architecture de commande proposée

La figure suivante illustre l'architecture de commande que nous proposons pour la main DLR/HIT, la commande des quatre doigts est effectuée par un seul circuit programmable, cette structure sera à la base des architectures que nous proposons plus tard pour la configuration.

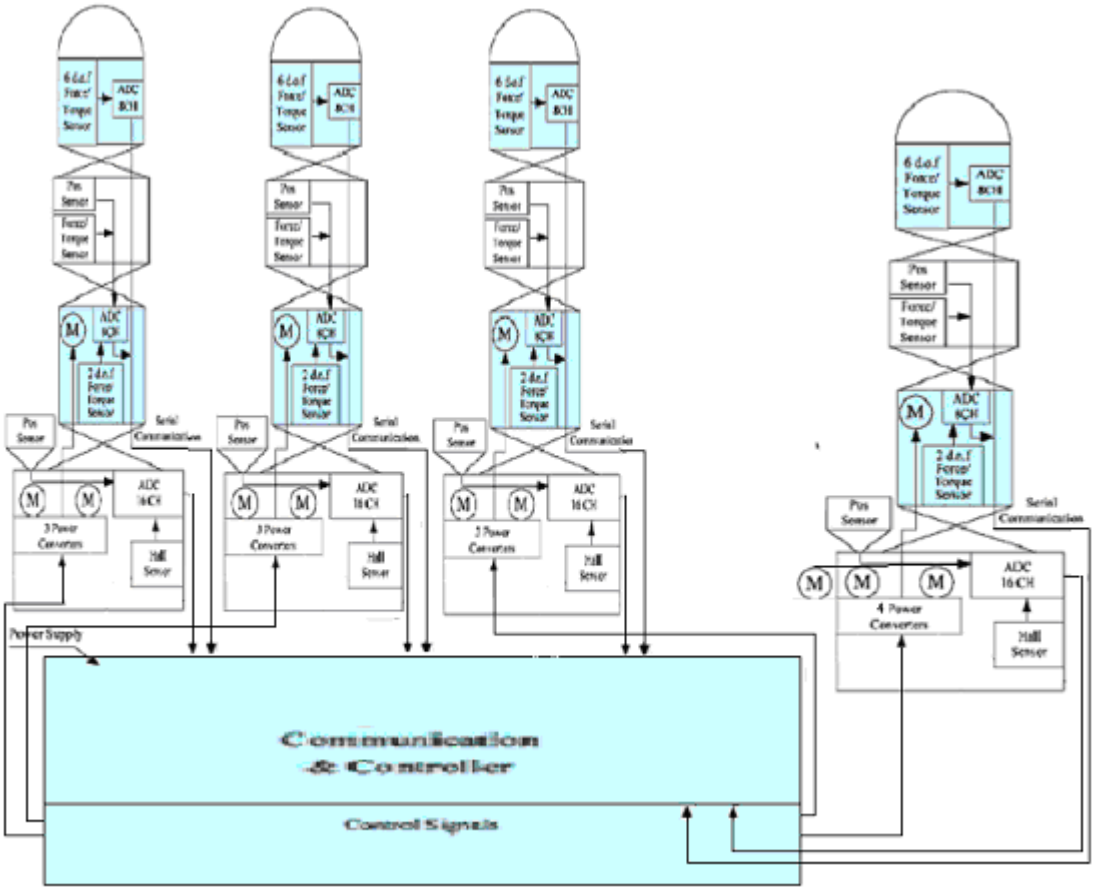



**Figure V.2 Configuration proposée pour la main DLR/HIT**

Il est possible de choisir une architecture parfaitement modulaire, c'est-à-dire concevoir un circuit programmable pour chaque doigt et un autre pour établir la communication entre les différents doigts.

Bien que notre mécanique soit virtuelle, nous jugeons utile d'étudier l'électronique de la main, d'autant plus qu'elles représentent les périphériques du circuit de commande.

### V.2.2 Le choix de la carte de commande

Différentes approches possibles pour un système logique existent, [12] elles consistent à choisir entre l'utilisation de la logique standard (le choix microcontrôleur), et l'utilisation de la solution On Chip qui elle-même offre la possibilité de choisir entre les ASICs (spécifiques à l'application pour laquelle ils ont été conçus), et les PLDs (logique programmable) et plus précisément les FPGAs.

Le microcontrôleur est un circuit intégré dédié aux applications embarquées. A la différence du microprocesseur, sa puce comporte un certain nombre d'interfaces, une RAM et PROM, des timers et des ports d'entrée/sortie. Il est vrai que l'aspect Standard des microcontrôleurs favorise leur utilisation dans tout type d'application, mais il peut être très pénalisant pour ce qui est des performances.

## V.2.2.1 La solution On Chip [13]

Un system en SOC[1] est un dispositif caractérisé par l'intégration de toutes les parties du système informatique en un seul circuit. En effet, cela permet de réduire :

- La dissipation de l'énergie.
- Les interconnections.
- La taille du dispositif.

Un system en SOC typique contient un ou plusieurs noyaux de processeurs, un nombre arbitraire de périphériques, une mémoire on chip, et un bus qui interconnecte tout ces dispositifs.

#### V.2.2.1.a Les ASICs (Application Specific Integrated Circuit) [13]

Par définition, les circuits ASIC regroupent tous les circuits dont la fonction peut être personnalisée d'une manière ou d'une autre en vue d'une application spécifique, par opposition aux circuits standards dont la fonction est définie et parfaitement décrite dans le catalogue des composants. Les ASICs sont généralement plus convenables pour les productions en série de

[1] Solution on chip.

conceptions déjà vérifiées et non pour les prototypes. L'inconvénient majeur réside dans le fait du *passage obligatoire chez le fondeur* ce qui implique des frais de développement élevés du circuit.

#### V.2.2.1.b Les PLD (Programmable Logic Device) [13]

Ce sont des chips qui peuvent être programmés pour se comporter comme une conception arbitraire. Un PLD peut être programmé pour une implémentation aussi simple qu'une opération en logique combinatoire comme il pourrait l'être pour des conceptions beaucoup plus importantes.

#### V.2.2.1.c Les FPGAs (Field programmable gate array) [13]

Ce sont des circuits de type PLD. Un FPGA possède une architecture générique qui consiste en un ensemble de blocs logiques et séquentiels configurables et d'un ensemble d'interconnections programmables.

Les circuits FPGAs ne sont pas optimisés pour une application bien déterminée, par conséquent ils consomment plus d'énergie que les ASICs. Par contre ils sont beaucoup plus simples à programmer et à reprogrammer, ce qui raccourcit les cycles de conception et permet de suivre l'évolution de l'application pour laquelle, ils ont été conçus.

Les FPGAs sont plus convenables pour les prototypes et pour les productions en série limitées qui ne sont pas de la qualité des ASICs.

Suivant toujours le principe du prototypage rapide, nous avons utilisé un FPGA du constructeur Xilinx, implanté sur une carte Spartan 3E.

## *V.3 Deuxième partie :Architectures proposées et l'outil API*

### V.3.1 La commande centralisée

Dans ce type de commande, les doigts de notre main artificielle seront pilotés avec un seul processeur implémenté sur SoC.

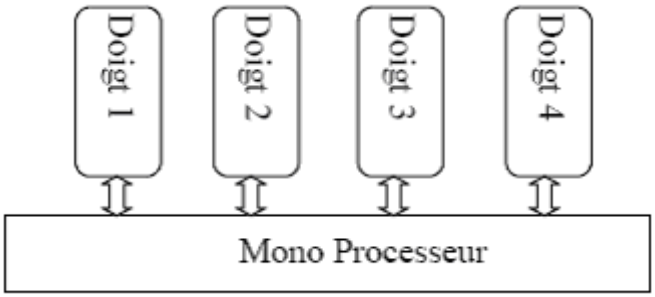



**Figure V.3 Commande centralisée - Architecture hardware**

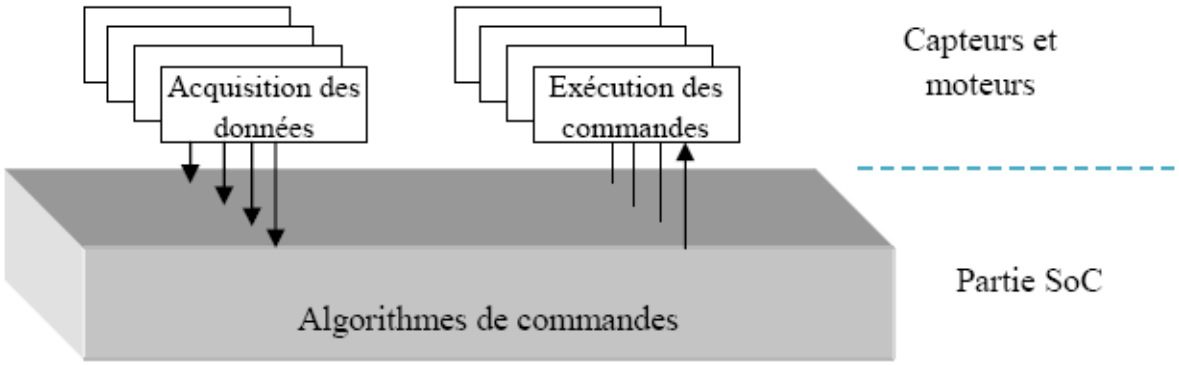



**Figure V.4 Commande centralisée - Architecture software.**

### V.3.2 La commande répartie

La deuxième approche est la commande repartie dans laquelle en implémente pour chaque doigt un processeur qui traitera les données de son doigt indépendamment des autres processeurs.

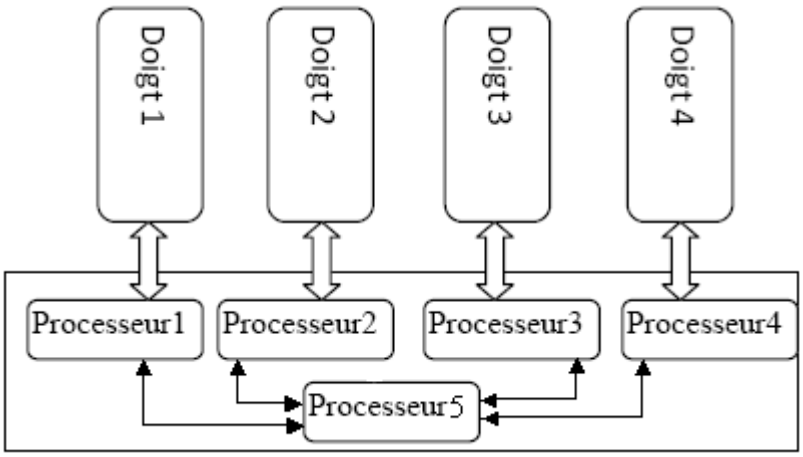



**Figure V.5 Commande répartie - Architecture hardware**

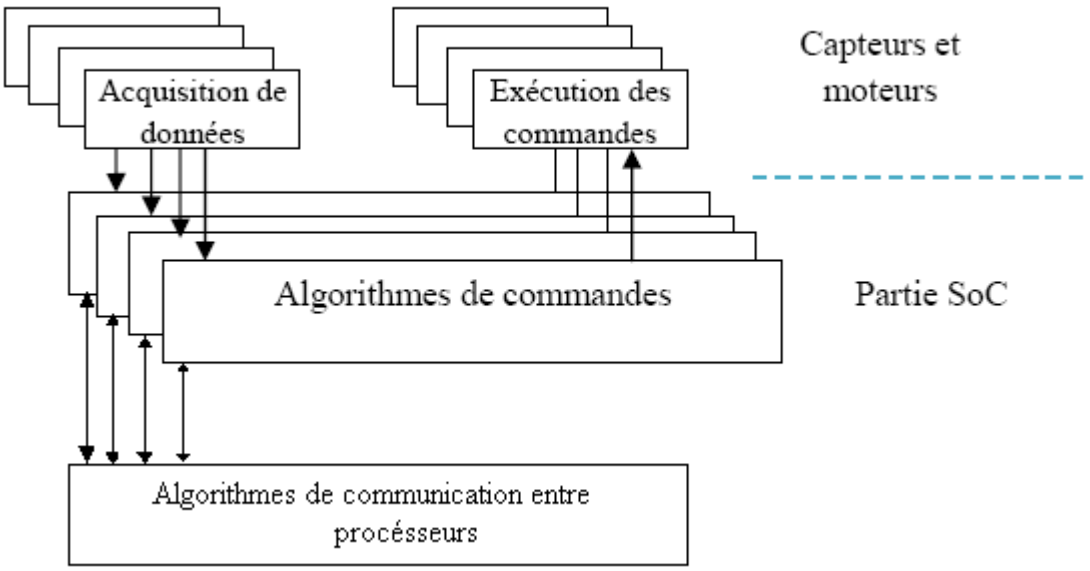



**Figure V.6 Commande répartie - Architecture hardware**

## V.3.3 L'API

Une **API** (Application Programming Interface, traduisez « interface de programmation » ou « interface pour l'accès programmé aux applications) est un ensemble de fonctions permettant d'accéder aux services d'une application, par l'intermédiaire d'un langage de programmation.

### V.3.3.1 Utilité de l'API

Une API permet de fournir un certain niveau d'abstraction au développeur, c'est-à-dire qu'elle lui masque la complexité de l'accès à un système ou à une application en proposant un jeu de fonctions standard dont seuls les paramètres et les valeurs retournées sont connus.

Ainsi, par analogie avec une voiture, le conducteur n'a pas à connaître le fonctionnement mécanique du moteur d'un véhicule pour pouvoir le conduire, il s'agit d'une certaine façon de l'interface proposée à l'utilisateur.

Grâce aux API, un développeur n'a donc pas à se soucier de la façon dont une application distante fonctionne, ni de la manière dont les fonctions ont été implémentées pour pouvoir l'utiliser dans un programme. Une API peut être disponible pour un langage particulier ou bien être disponible pour plusieurs langages de programmation.

### V.3.3.2 Illustration pour l'animation

Le but de notre utilisation de l'API est d'arriver à animer la structure selon nos données, pour cela nous présentons dans ce qui suit, en détail ;un petit exemple illustratif pour la recherche des noms de fonction que nous utiliserons plus tard sous Visual Basic.

La méthode la plus simple est l'utilisation des macros, les macros dans SolidWorks sont sous Visual basic qui est recommandé par le constructeur pour la programmation des APIs, on peut enregistrer une macro de la façon suivante :

On lance l'enregistrement sous SolidWorks dans la barre d'outils Macro (voir figure ci-opposée-dont les outils sont exécuter-arrêter- nouvelle et éditer macro), par exemple pour effectuer une flexion de l'articulation moyenne d'un doigt, on sélectionne l'articulation puis dans la Barre d'outils Gestionnaire de commande, on sélectionne rotation du composant (voir figure), puis on arrête l'enregistrement, on enregistre et on édite la macro, pour on visualise le nom de fonction activée ; Rotatecomponent. (voir la figure ci-dessous)

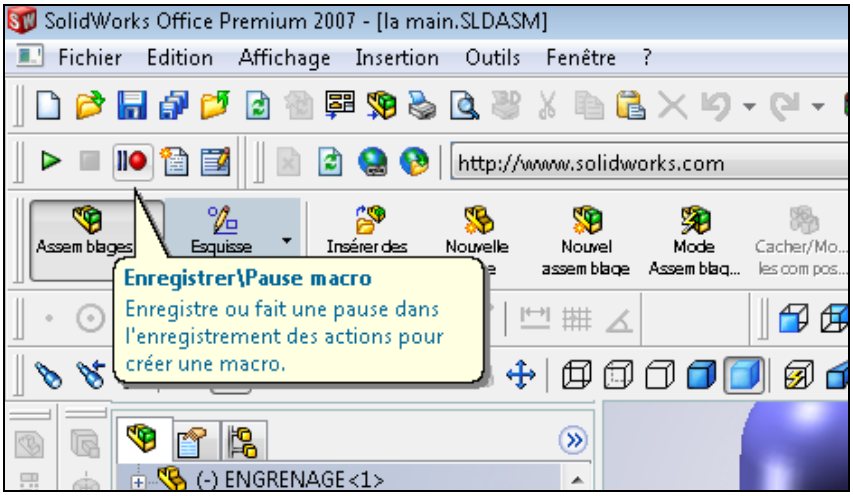



**Figure V.7 Début d'enregistrement d'une macro**

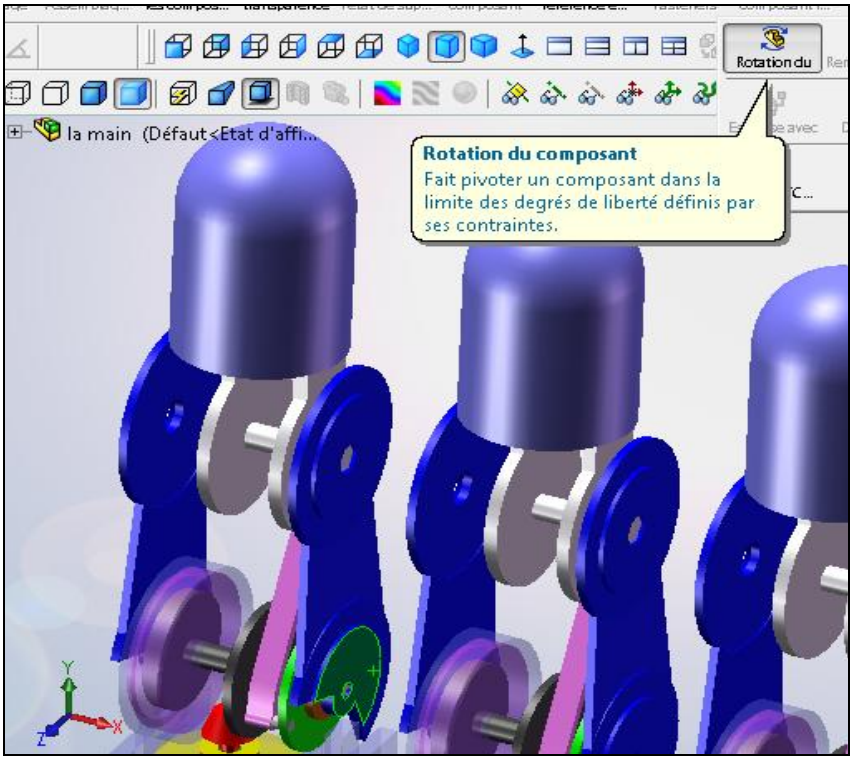



**Figure V.8 Sélection de la fonction rotation d'un composant**

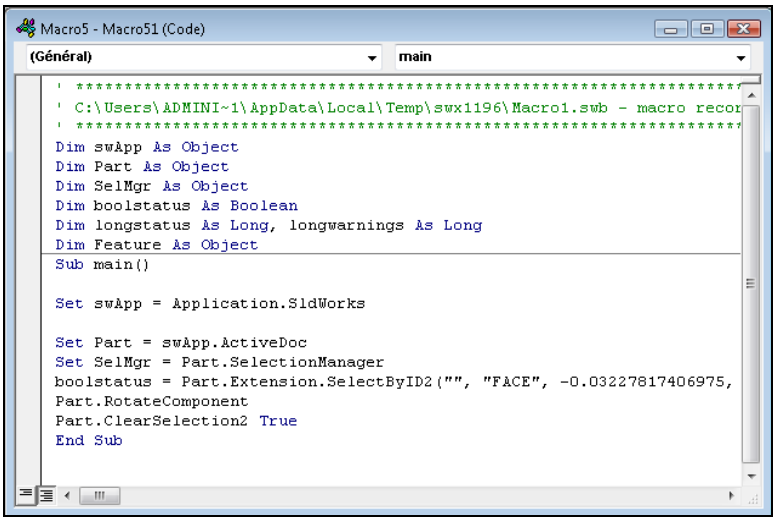



**Figure V.9 La macro obtenue**

De la même façon on peut obtenir les noms de plusieurs fonctions de SolidWorks.

## *V.4 Troisième partie : Mise en œuvre*

### V.4.1 Introduction

Notre travail consiste à interfacer SolidWorks avec une électronique externe qui se chargera de piloter le prototype virtuel de la main DLR. Un des grands avantages du prototypage rapide est sa flexibilité qui nous a permis une dissociation entre la partie mécanique et la partie motorisation de la main ce qui est impossible dans une réalisation physique du prototype, ce concept est illustré dans les figures V.14 et V.15

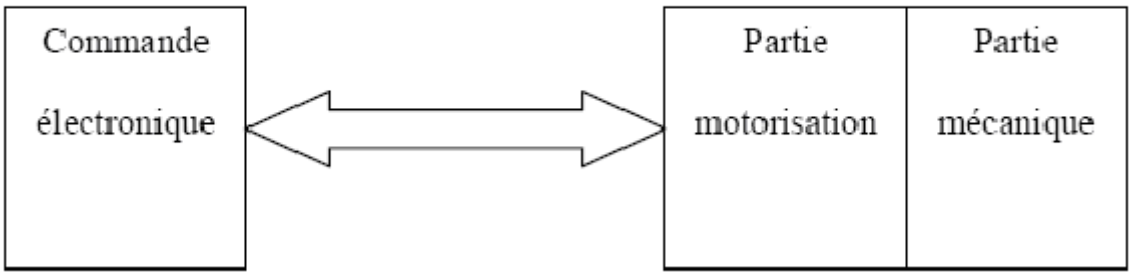



**Figure V.10 Cas d'une réalisation physique**

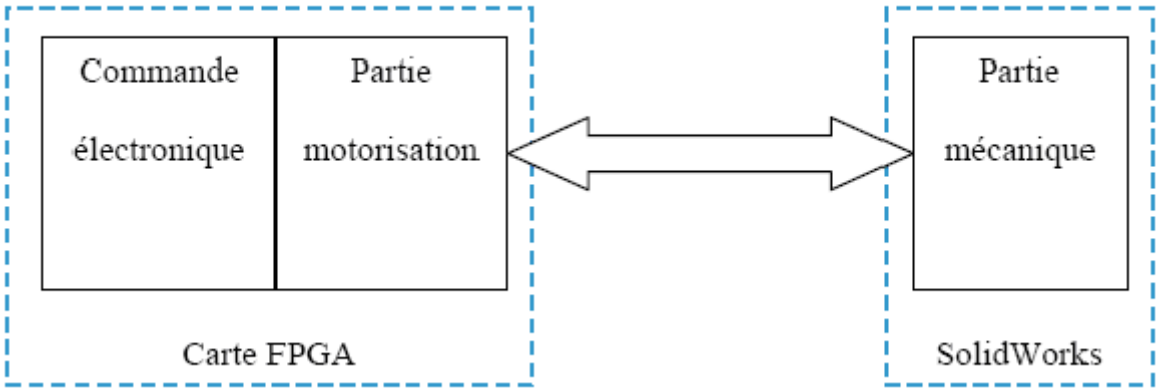



**Figure V.11 Cas d'une réalisation virtuelle**

Dans notre cas, cette approche se traduit par l'intégration de la partie motorisation dans le circuit de commande sous forme de tableaux contenant les angles décrivant la trajectoire du mouvement du doigt. Ces angles sont les résultats d'une simulation sous MATLAB des moteurs.

La description de notre travail est la suivante :

Pour chaque doigt de la main, on a des tableaux de valeurs décrivant une trajectoire pour ce dernier tel que le mouvement de chaque doigt est géré par un *thread* généré par le système d'exploitation Xilkernel et qui est indépendant des autres *thread*[2]

Le déclenchement du mouvement d'un doigt est fait avec l'un des boutons poussoirs disponibles sut la carte FPGA.

Le travail comporte les parties suivantes :

· La mise en oeuvre de la plateforme matérielle.

· La réalisation de la partie software sous EDK.

· La réalisation de la partie software sous Windows.

· Résultat et conclusion.

Nous soulignons que nous avons choisi la configuration matérielle de type architecture centralisée, cela pour simplifier l'étape de la programmation d'autant plus que le temps d'étude est une contrainte à ne pas négligée.

## V.4.2 La mise en œuvre de la plateforme matérielle

### V.4.2.1 Configuration de la carte

La plateforme matérielle est créée à l'aide du BSB en suivant les étapes suivantes :

[2] Un *thread* ou tâche ou processus léger, est similaire à un processus car tous deux représentent l'exécution d'un ensemble d'instructions du langage machine d'un processeur. Contrairement aux processus, les threads partagent le même espace mémoire ce qui rend le basculement d'un thread à un autre beaucoup plus rapide dans un système d'exploitation, on aura en onséquence une réduction dans la consommation du temps CPU et de la mémoire utilisée.

· Lancement du logiciel XPS.

· Sélection du modèle de la carte qui est dans notre cas Spartan 3E.

· Choix du processeur MICROBLAZE car c'est le seul processeur disponible pour ce modèle de la carte, ainsi que la fréquence d'horloge qu'on la prend égale a 50 Mhz, et la taille de la BRAM qui sera 8Ko.

La sélection et la configuration des interfaces d'entrée/sortie du système sont :

_ RS232_DCE qui sera utilisé comme périphérique d'entrée/sortie standard.

_ BUTTON_4BIT qui sera utilisé pour déclencher les différents processus.

_ DDR_SRAM, son utilisation est nécessaire pour charger notre application vue sa taille importante.

_ Un *timer* qui va générer les ticks pour le MicroBlaze.

· L'étape finale de cet assistant nous donne un résumé pour tous les périphériques choisis ainsi que leur adressage au niveau du processeur. Après confirmation, le BSB se ferme et nous renvoi vers l'interface XPS.

La figure IV-3 représente le diagramme block de la plateforme crée.

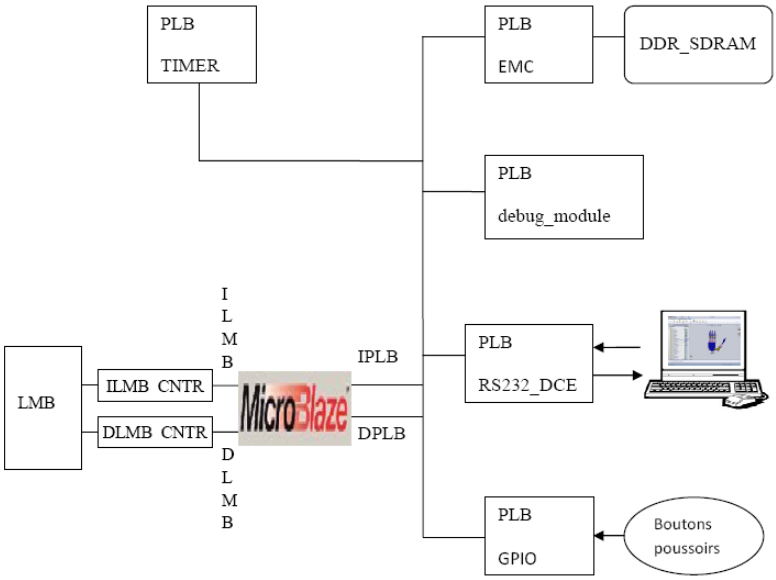



**Figure V.12 Diagramme Block de la plateforme matérielle**

Pour l'adressage, EDK se charge d'attribuer à chaque périphérique l'espace mémoire approprie. Le tableau VI-1 résume le système d'adressage de la plateforme adoptée ainsi que les bus de connexions.

| Instance | Name | Base address | High address | Bus connexion |
|---|---|---|---|---|
| dlmb_cntlr | C-BASEADDR | 0x00000000 | 0x00001fff | dlmb |
| ilmb_cntlr | C_BASEADDR | 0x00000000 | 0x00001fff | ilmb |
| debug_module | C_BASEADDR | 0x84400000 | 0x8440ffff | mb_plb |
| button_4Bit | C_BASEADDR | 0x81400000 | 0x8140ffff | mb_plb |
| xps_timer_1 | C_BASEADDR | 0x83c00000 | 0x83c0ffff | mb_plb |
| RS232_DCE | C_BASEADDR | 0x84000000 | 0x8400ffff | mb_plb |
| DDR_SDRAM | C_MPMC_BASEADDR | 0x8c000000 | 0x8fffffff | mb_plb |

**Tableau V.1 Résumé du système d'adressage**

### V.4.2.2 Génération de la plateforme matérielle

La plateforme matérielle est créée en utilisant l'outil de génération de plateforme PlatGen. Cet outil a pour entrée le fichier MHS qui définit l'architecture du système, les périphériques, les processeurs enfouis, les connections du système, la configuration des adresses de chaque périphérique dans le système et les options de configuration de chaque périphérique. PlatGen crée donc une Netlist qui décrit le matériel utilisé en plusieurs formes (NGC, EDIF). Après l'exécution de cet outil, les outils d'implémentation sur FPGA (ISE) s'exécutent pour compléter l'implémentation du matériel. A la fin du flot ISE, un fichier de configuration (Bitstream) est généré. Le tableau suivant donne un résume sur les ressources utilisées par la plateforme ainsi générée

| Type | Utilisée | Disponible | % |
|---|---|---|---|
| Slices | 2579 | 4656 | 55 |
| Slice Flip Flops | 3083 | 9312 | 33 |
| 4 input LUTs | 3492 | 9312 | 37 |
| IOs | 15751 | NA | NA |
| bonded IOBs | 44 | 232 | 18 |

**TableauV.2 Ressources utilisées par la plateforme matérielle**

### V.4.3 La réalisation de la partie software sous EDK

Après la création de la partie hardware et la génération du Bitstream, on lance le SDK directement à partir de l'XPS qui nous permettra de créer notre partie software à savoir la plateforme software, la configuration et l'implémentation du noyau XilKernel et l'écriture de l'application software.

### V.4.3.1 La plateforme software

La plateforme logicielle est spécifiée par le fichier MSS (Microprocessor Software Specification) qui définit les librairies, les pilotes, les paramètres de personnalisation du processeur, les dispositifs d'entrées/sorties, les routines de traitement d'interruptions et d'autres

caractéristiques du software. Le fichier MSS est utilisé comme entrée pour l'outil de génération de librairies (LibGen). Cet outil nous permet de configurer les librairies et les pilotes avec les adresses des périphériques du processeur enfoui. Les librairies et les pilotes configurés et les programmes sources (d'extension .c ou bien .h) seront compilés grâce à mb-gcc qui est un compilateur GNU adapté à MicroBlaze.

Le résultat est un fichier exécutable qui sera chargé dans la BRAM ou la SRAM de MicroBlaze pour être exécuté.

### V.4.3.2 Configuration et Implémentation de XilKernel

L'un des avantages de l'utilisation du noyau XilKernel est sa disponibilité dans l'environnement EDK. Pour la configuration du noyau XilKernel on a suivi les étapes suivantes :

· Dans la première fenêtre de l'assistant "Software Platform Settings" on choisie XilKernel comme système d'exploitation. Et on ajoute –lxilkernel comme extra_compiler_flag nécessaire pour l'utilisation des fonctionnalités de XilKernel.

· Dans la fenêtre "OS and libraries" on configure le système d'exploitation comme suit :

_ On choisi RS232_DCE pour les paramètres de configuration stdin and stdout.

_ On choisi "xps_intc_0 comme le "sysintc_spec pour spécifier le nom de l'instance du périphérique de contrôle d'interruption qui pilotera les interruptions du système.

_ Dans le menu systmr_spec on choisie xps_timer_1 comme systmr_dev pour spécifier le timer utiliser pour l'OS.

_ Dans le menu config_pthread_support on choisi static_pthread_table et on ajoute la fonction test_start_func avec une priorité de niveau 1, cette fonction contiendra les threads crées lors du lancement de Xilkernel.

_ Dans le menu config_sched on sélectionne le mode tourniquet (SCHED_RR) comme mode de fonctionnement de l'ordonnanceur de taches.

_ Pour avoir la capacité d'arrêter le fonctionnement (KILL) d'un thread on doit mettre l'option enhanced_features à la valeur true.

_ On quitte l'assistant "Software Platform Settings" en cliquant sur le bouton OK pour revenir à la fenêtre principale de SDK.

Une fois la configuration du Xilkernel est terminée, on doit générer le Linker Script qui sera utilisé pour lancer notre application directement à partir de la mémoire DDR-SDRAM et ajouter la librairie Xilkernel aux librairies du compilateur gcc puis charger le Bitstream généré sur la carte FPGA pour avoir l'architecture suivante :

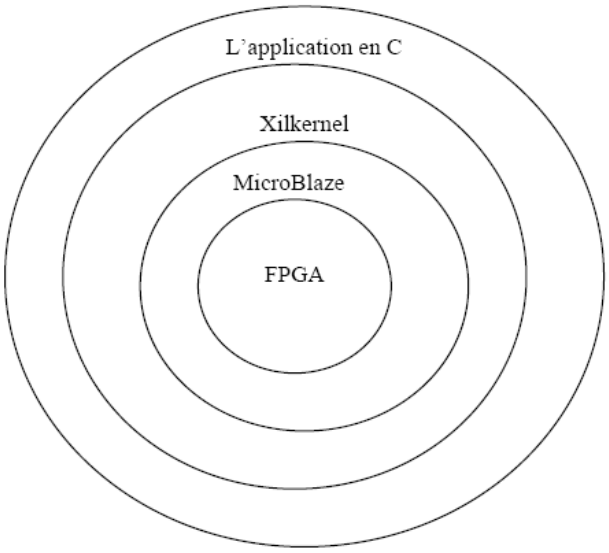



**Figure V.13 L'architecture de la partie software**

## V.4.3.3 L'application software sous EDK

### V.4.3.3.2 Architecture de l'application

Cette application doit assurer l'échange de données avec la partie software sous Windows, ce qui nous a menés à l'utilisation des threads gérés par l'OS Xilkernel.

Le fonctionnement sera comme suit :

Après avoir adapté la carte, compilé le noyau temps réel, on procède à l'étape finale qui est l'écriture du programme qui doit s'exécute comme suit :

Au début, la fonction principale main() est exécutée automatiquement, dans laquelle on fait appel à la fonction xilkernel_main(), qui ce chargera de lancer le système d'exploitation Xilkernel.

Une fois le noyau appelé et initialisé, il appel son Ordonnanceur pour ordonnancer les différentes tâches du programme.

La première fonction qui sera exécuté dans notre programme est la fonction start_test_func qui a la priorité la plus grande et qui sera donc exécutée dés le démarrage de Xilkernel. Pour chaque doigt de notre main un thread est crée tout en lui affectant une fonction approprié (lanched_pthread1, lanched_pthread2 ou lanched_pthread3). Cette dernière contient des instructions qui traduisent lors de leurs exécutions l'organigramme suivant :

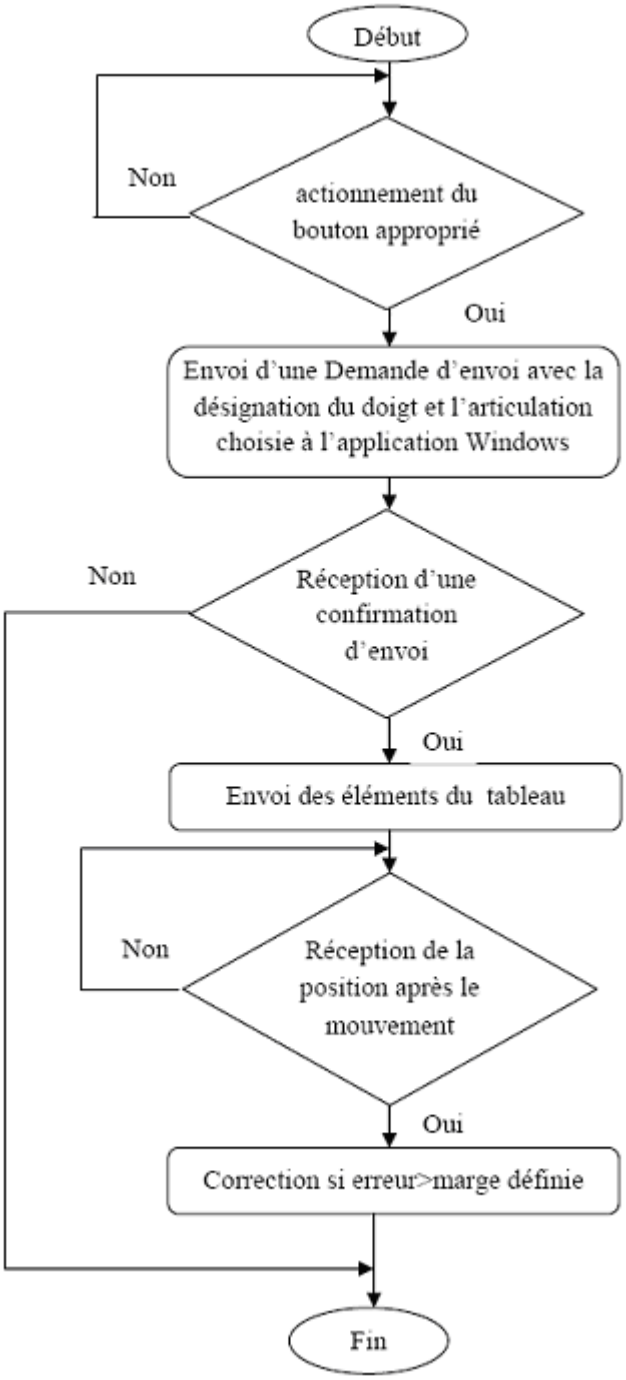



**Figure V.14 l'organigramme de la fonction gérante du doigt**

Une priorité entre les fonctions fingerX_func a été établit pour contourner le problème du temps de réponse relativement long du logiciel SolidWorks.

Finalement on aboutira à l'architecture suivante :

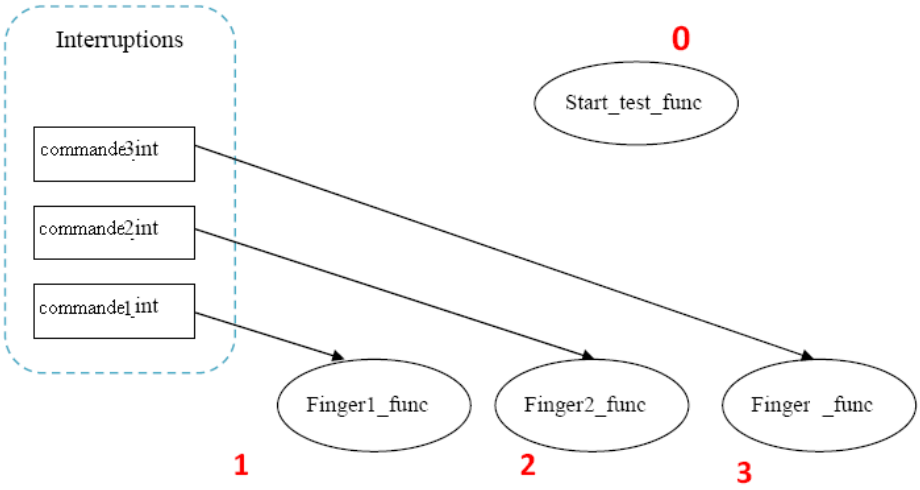



**Figure V.15 Architecture de l'application sous EDK**

### V.4.4 La réalisation de la partie soft sous Windows

L'application réalisée sous Visual Basic est du type *Windows Form*, donc elle comporte une interface graphique qui permet une meilleure interactivité avec l'utilisateur et une partie code pour rendre cette dernière fonctionnelle.

Le code est structuré sous forme de fonctions et de subroutines permettant une modularité qui a pour avantage de ne pas changer tout le code lors de la modification d'une fonctionnalité.

#### V.4.4.1 Le fonctionnement de l'application

Lors du lancement de l'application, une subroutine est chargée d'inspecter les ports accessibles sur la machine et les afficher a l'utilisateur afin qu'il choisi le port sur lequel la carte est branchée en plus de la configuration de ce dernier.

· La grande partie du programme est écrite sous la subroutine qui se déclenche automatiquement lors de la réception des données sur le port RS232. Elle fait appel aux fonctions suivantes :

_ SelComponent() : Cette fonction permet de sélectionner le composant dont le nom est transmis comme paramètre de type chaine de caractères. Le type d'objet de retour est Component2.

_ Origine() : Cette fonction a aussi comme paramètre le nom du composant, est donne en retour les coordonnées du point d'origine du composant voulu. Le retour est du type tableau de valeurs.

_ createTrans() : Cette fonction a comme retour un objet du type *MathTransform* qui sera crée à partir des paramètres suivants : coordonnées de l'origine, Axe de rotation et de l'angle de rotation autour de cet axe.

_ Difference() : Cette fonction permet d'avoir en sortie un tableau de valeurs qui donne les différences entre deux éléments successifs du tableau en entrée.

L'Algorithme de la partie soft sous Windows est conçu comme suit :

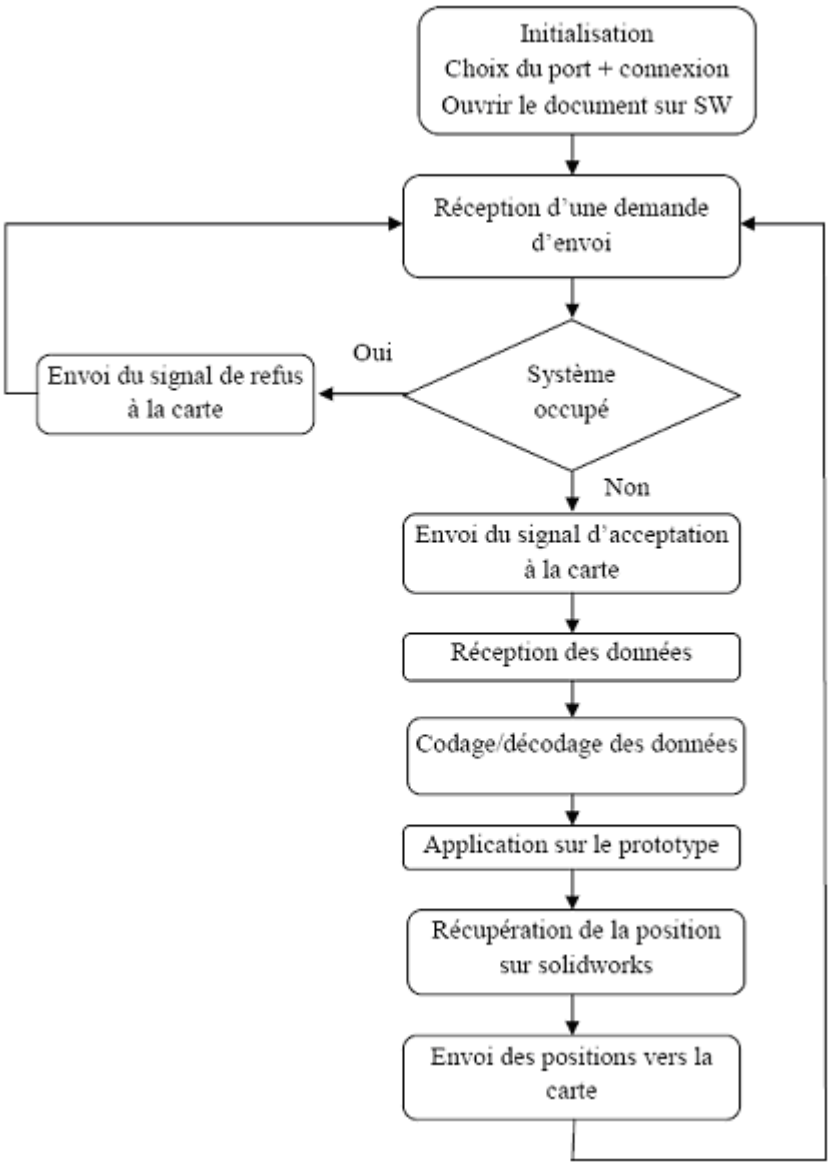



**Figure V.16 L'organigramme de la réception des données au niveau de la partie soft sous Windows**

Le codage de l'information qui circule entre les deux parties software respecte les règles suivantes

·Le transfert de données se fait par octets.

·Un octet envoyé par l'application sous EDK égal à 'D' est décodé comme demande d'envoi au niveau de l'application sous Windows ,et doit être suivi par deux octets portants le numéro du doigt et le numéro de l'articulation a piloter respectivement.

·Un octet envoyé par l'application sous Windows égal à 'O' est décodé comme confirmation d'envoi.

·Tous les octets différents de 'D' et 'O' seront considérés comme données et vont êtres ajoutés respectivement au tableau de données qui contient la trajectoire de l'articulation du doigt choisis.

### V.4.5 Résultat de l'exécution

Une fois que la carte est mise sous tension en cliquant sur le bouton SolidWorks dans l'interface graphique créée par l'API, ainsi le logiciel est lancé, puis sur connexion, on choisi l'un des boutons dédiés pour commander les doigts sur la carte FPGA, une fois que cela fait on visualise sur l'écran le mouvement du doigt désigné, des leds feront témoins de l'état de la carte de commande.

## *V.5 Conclusion*

Le défi de ce chapitre est d'arriver à commander la main virtuelle que nous avons conçue sous SolidWorks par une électronique sous forme d'un Soc. Pour ce faire, nous avons établi une étude du matériel disponible d'où le choix de la carte FPGA Spartan 3E XC3S500E. Pour sa configuration et sa programmation l'étude de l'environnement EDK nous a été indispensable, une autre étude du Soft était établie ; celle de l'API de SolidWorks pour l'interfaçage entre l'FPGA et la main virtuelle sous SolidWorks.

Pour l'application, nous avons choisi de faire une lecture des données d'un tableau contenant les angles des articulations d'un doigt, qui sera traduite par les mouvements de ces articulations sous SolidWorks.

# Conclusion et perspectives

Deux outils du prototypage rapide ont été évoqués, le premier est SolidWorks pour la conception mécanique, le deuxième est l'FPGA ; SolidWorks nous a permis de créer la main DLR/HIT puis à simuler en 3 dimensions son comportement via une commande provenant du circuit FPGA.

L'utilisation d'une barre de transmission a compliqué les équations pour le calcul du modèle géométrique inverse, la matrice pseudo inverse, le modèle cinématique et cinématique inverse raison pour laquelle nous avons choisi la solution numérique. Aussi, le volume des différentes expressions nous a menés à utiliser les variables symboliques sous Matlab pour le calcul du modèle dynamique.

Pour la commande nous avons choisi une commande classique par PID.

Plusieurs thèmes peuvent faire l'objet de nos perspectives tel que :

La modélisation dédiée au pouce puisqu'il a une articulation de plus.

L'étude énergétique pour réaliser l'autonomie en ce sens.

La réalisation d'une interface capable de traduire des signaux neuronaux ou musculaires pour réaliser la connexion entre la pensée et la main DLR/HIT pour un projet de prothèse ou même une prothèse virtuelle.

La réalisation d'une commande en boucle et la simulation du comportement des capteurs sous SolidWorks.

L’application d’autres commandes telle que la commande en effort d’autant plus que chaque doigt est équipé d’un capteur de force à 6 degrés de liberté.

# Bibliographie

[1] The modular multisensory DLR-HIT-Hand, H. Liu a, P. Meusel a, N. Seitz a, B. Willberg a, G. Hirzinger a, M.H. Jin b, Y.W. Liu b, R. Wei b, Z.W. Xie b. Proceedings of the 2005 IEEE/ASME, International Conference on Advanced Intelligent Mechatronics Monterey, California, USA, 24-28 July, 2005

[2] A "Wearable" Artificial Hand for Prosthetics and Humanoid Robotics
Applications. M. C. Carrozza, B. Massa, S. Micera, M. Zecca, P. Dario_Scuola Superiore Sant'Anna, Pisa, Italy and Centro INAIL RTR, Viareggio (Lu), Italy

[3] Prédiction des efforts musculaires dans le système main avant-bras : Modélisation, simulation, optimisation et validation. Joe CHALFOUN. 29 avril 2005, thèse de doctorat.

[4] Robotic Hands: Design Review and Proposal of New Design Process,Jimmy W. Soto Martell, and Giuseppina Gini

[5] Blackfingers: an Artificial Hand that Copies Human Hand in Structure, Size, and Functions. Michele Folgheraiter, Giuseppina Gini DEI, Politecnico di Milano,

[6] THE ADAH PROJECT: AN ASTRONAUT DEXTEROUS ARTIFICIAL HAND TO RESTORE THE MANIPULATION ABILITIES OF THE ASTRONAUT, M.C. Carrozza, F.Vecchi, S. Roccella, L. Barboni, E. Cavallaro, S. Micera, P. Dario

[7] Learning Techniques in a Dataglove Based Telemanipulation System for the DLR Hand, M_ Fischer_ P_ van der Smagt_ and G_ Hirzinger

[8 DLR-Hand II: Next Generation of a Dextrous Robot Hand, J. Butterfa_, M. Grebenstein, H. Liu and G. Hirzinger.. Proceedings of the 2001 IEEE International Conference on Robotics and Automation, Seoul, Korea, May 21-26, 2001.

[9] Maxime Nicole, « SolidWorks-Bases_et_Pieces»,Cours master , Université de Sherbrooke, 2005, CANADA.

[10] Vue d'ensemble de l'interface utilisateur, guide de l'utilisateur en ligne SolidWorks 2007.

[11] Modélisation identification et commande des robots, 2ème edition; Wisama Khalil, Etienne Dombre.1999.

[12] « Circuits logiques programmables », maitrise EEA, N. JULIEN ,*septembre 1999 Universitéde bretagne sud lorient*

[13] "Evaluating Xilinx MicroBlaze for Network SoC solutions"; *thése de magister en the Computer Engineering*,Peter Magnusson, 10th January 2004.

[14] Spartan-3E FPGA Starter Kit Board User Guide UG230 (v1.1) June 20, 2008.

[15] MicroBlaze Processor Reference Guide Embedded Development Kit EDK 10.1i UG081 (v9.0).

[16] "EDK OS and libraries Reference Manual", E*mbedded Development Kit 6.3i*. UG 114(V3.0), 20 aout 2004.

[17] SolidWorks API Help 2008.