الجمهـوريـــة الجزائـــريـــة الديمقـراطيـــة الشـعبيـــة
République Algérienne Démocratique et Populaire
وزارة التـــعلـيـــم العـــالـــي والبـحـــث العـلمـــي
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

جامعة وهران 2 محمد بن أ حمد
Université d'Oran 2 Mohamed Ben Ahmed

معهد الصيانة و الأمن الصناعي
Institut de Maintenance et de Sécurité Industrielle

Département de Maintenance en Instrumentation

MÉMOIRE

Pour l'obtention du diplôme de Master

Filière : Génie Industriel
Spécialité : Génie Industriel

Thème

Simulation d'une cellule robotisée pour coopération de robots industriels par le logiciel d'ABB RobotStudio

Présenté et soutenu publiquement par :

Nom : Younes
Prénom : Oussama

et

Nom : Chebbah
Prénom : Oussama

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
Mekki Ibrahim El Khalil	MCA	IMSI-Univ. D'Oran2	**Président**
Kacimi abderrahmane	MCB	IMSI-Univ. D'Oran2	**Encadreur**
Titah Mawloud	MAA	IMSI-Univ. D'Oran2	**Examinateur**

Année 2019/2020

Remerciements

C’est avec une profonde reconnaissance et considération particulière que je remercie :

- Allah, le miséricordieux le très miséricordieux qui nous a donné la force, la patience, le courage et la volonté pour élaborer ce travail.

- Mon père, ma mère, mes frères et sœurs qui m'ont toujours entouré et motivé sans cesse à devenir meilleur et à me surpasser.

- Mon encadreur Mr A .Kacimi d’avoir accepté de superviser ma thèse, pour vos conseils votre disponibilité et votre soutien dans les moments délicats.

- Tous mes enseignants qui ont contribué à ma formation depuis l’école primaire jusqu’aux études universitaires.

- Toutes les personne qui m’ont soutenu, ma famille, mes amis, mes camarades que je ne peux nominativement citer , qu’ils trouvent ici l’expression de ma profonde reconnaissance …

Dédicace

Je dédie ce travail: À mes chers parents, pour tout leurs sacrifies, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études.

À mes chers freres pour leur encouragements permanents et leur soutien moral. Àtoute ma famille et mes amis pour leur soutien tout au long de mon parcours universitaire.

À tous mes professeurs :Leur générosité et leur soutien m'oblige de leurs témoigner mon profound respect et ma loyale considération.

À mon binôme :Younes Oussama

Chebbah Oussama

Dédicace

Je dédie ce travail à :

A mes chers parents

A mes frères et sœurs

A toute ma famille

A tous mes professeurs

À mon binôme

A tous mes camarades

A tous mes amis

Younes Oussama

Résumé :

Notre travail consiste à faire la simulation d'une cellule robotisée , permettant la coopération de plusieurs robots industriels, commandée par le logiciel ABB RobotsStudio .

Au premier lieu nous s'intéressons aux définitions et généralités sur les robots ainsi que leurs caractéristiques , paramètres et domaines d'application .

Dans une deuxieme etape on a abordé les outils mathématiques pour la planification des taches industrielles,

Ainsi on a fini par la planification des taches industrielles de notre cellule robotisée et puis l'application sous le logiciel et la conception des taches.

Mots-clés : RobotStudio, cellule,ABB, synchronisation, programmation,industrie,articulation,apprentissage,commande

Abstract :

Our work consists of simulating a robotic cell, allowing the cooperation of several industrial robots, controlled by the ABB RobotsStudio software.

First, we are interested in definitions and generalities about robots as well as their characteristics, parameters and fields of application.

In a second step we approached the mathematical tools for the planning of industrial tasks,

So we ended up planning the industrial tasks of our robotic cell and then the application under the software and the design of the spots.

Keywords: RobotStudio, cell, ABB, synchronization, programming, industry, articulation, learning, control

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Chapitre II

Abréviations :

PPP : 3 articulations prismatiques

DDL : degré de liberté

RPP : articulation rotoide avec 2 articulations prismatiques

RRP : deux articulations rotoides avec une articulation prismatique

RRR : 3 articulations rotoides

SCARA : sélective compliance assembly robot arm

TRANS : translation

ROT : rotation

SMA : système mécanique articulé

MGD : modèle géométrique direct

MGI : modèle géométrique inverse.

Introduction générale :

Quand on parle de robotique, plusieurs idées viennent à l'esprit de chacun de nous , nous pourrions nous référer aux premiers concepts et automates de l'antiquité ou aux premiers robots comme à des personnages de la mythologie, c'est à siècle dernier que l'éclatement, de la robotique industrielle a amorcé l'explosion des thèmes de recherche, a cette époque les robots étaient conçus en respectant les contraintes imposées par le milieu industriel, comme la respectabilité et la précision des tâches est avec les développements scientifiques, spécifiquement de l'électronique et de l'informatique mais aussi automatique, mathématique, mécanique, matériaux, que la technologie robotique a progressé, les robots actuels sont dotés d'une intelligence qui leur donne une certaine autonomie qui va leur permettre de se diffuser dans de nouveaux domaines, la robotique est officiellement comme un contrôle automatique, reprogrammable, polyvalent manipulateur programmable dans trois axes, sont considérés précise du robot est un système automatique mécanisé cabale d'effectuer une ou plusieurs taches dans un environnement donné, c'est une machine programmable capable d'exécuter diverses taches répétitives, et en principe, d'adapter son comportement a certains et évènements perturbant le fonctionnement nominal prévu : les taches peuvent être de type point a point ou de type continu : suivie de courbes, suivi de surfaces, les premières correspondent aux procèdes les plus répondus dans l'industrie manufacturière dont le soudage par résistance, la palettisation, le perçage et la manutention, tandis que les secondes, elles correspondent aux procèdes de soudage a l'arc, encollage, découpe, projection peinture.

Les robots se substituent à l'homme ou prolongent son action en apportant précision, rapidité ou capacité à appliquer d'importants efforts .

Dans ce sens, notre travail consiste à faire la simulation d'une cellule robotisée , permettant la coopération de plusieurs robots industriels par le logiciel ABB RobotsStudio

Ce mémoire est décomposé en quatre chapitres :

- Dans le premier chapitre, nous intéressons aux définitions et généralités sur les robots ainsi que leurs caractéristiques, paramètres et domaines d'application.

- Le deuxième chapitre est dédié aux outils mathématiques pour la planification des taches industrielles, pour ce faire nous passerons en revue les différents modèles géométriques et cinématiques qui expriment respectivement la situation et les vitesses de l'organe terminal en fonction des variables articulaires et inversement.

- Le troisième chapitre est consacré à la planification des taches industrielles de notre cellule robotisée .

- Ainsi on a fini par la simulation sous le logiciel RobotStudio™ .

Chapitre 1

Généralités sur la robotique

I.1 Introduction :

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

Dans ce premier chapitre, on va donner un aperçu non exhaustive sur les robots, un bref historique sur l'évolution de la robotique industrielle et présenter les différents types de robots et les éléments constitutifs de ces derniers.

I.2 Définitions générales :

I.2.1 La robotique :

C'est une science qui s'intéresse aux robots. En fait, il s'agit d'un ensemble de disciplines techniques (mécanique, électronique, automatique, informatique) articulées autour d'un objectif et d'un objet communs. Cet objectif est l'automatisation flexible de nombreux secteurs de l'activité humaine réputés jusqu'à très récemment comme ne pouvant se passer de la présence de l'homme, et l'objet est le robot, sorte de machine universelle dont l'homme rêve depuis toujours pour le remplacer dans les tâches difficiles .

I.2.2 Le robot industriel :

Le terme robot a été introduit pour la première fois par l'auteur Tchèque Capek en 1920 dans sa pièce de théâtre R.U.R. (Rossum's Universal Robot) et il est dérivé du mot robota qui signifie travailleur (de force). [2]

Un robot (figure 1.1) est un dispositif mécatronique (alliant mécanique, électronique et informatique) accomplissant automatiquement des tâches diverses. C'est une machine intelligente fonctionnelle qui nécessite une autonomie de mouvements.

L'Organisation Internationale de Normalisation définit le robot comme étant un manipulateur à plusieurs degrés de liberté, à commande automatique, reprogrammable, multi- applications, mobile ou non, destiné à être utilisé dans les applications d'automatisation industrielle. [3]

I.2.3 La composition d'un robot :

* Le mécanisme :

Structure plus au moins proche de celle du bras humain, on dit aussi manipulateur quand il ne s'agit pas d'un robot mobile. Sa motorisation est réalisée par des actionneurs électriques, pneumatiques ou hydrauliques qui transmettent leur mouvement aux articulations par des systèmes appropriés.

* La perception :

Permet de gérer les relations entre le robot et son environnement. Les organes de perception sont des capteurs dits « proprioceptifs » lorsqu'ils mesurent l'état interne du robot (position et vitesses des articulations) ou « extéroceptifs » lorsqu'ils recueillent des informations sur l'environnement (détection de présence, mesure de distance, vision artificielle).

* La commande :

Qui synthétise les consignes des asservissements pilotant les actionneurs. A partir de la fonction de perception et des ordres de l'utilisateur, elle permet d'engendrer les actions du robot.

* L'interface homme-machine :

A travers laquelle l'utilisateur programme les taches que le robot doit exécuter.

* Le poste de travail et les dispositifs péri-robotique :

Qui constituent l'environnement dans lequel évolue le robot.

Figure 1.1 : Robots à six et cinq ddl: (a) robot sériel à six articulation et (b) robot sériel à cinq articulation.

I .3 Historique :

La robotique industrielle a connu un essore entre 1950-1970.

Elle a vu le jour en 1954 lorsque Georges DEVOL a pu réaliser son brevet sur la robotique. Dans ce brevet Devol a conçu un robot qu'il a intitulé Unimate.

En 1961, le premier Unimate fut utilisé dans les usines de GENERAL MOTORS.

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

En 1978 un nouveau robot est conçu par Unimation Inc avec l'aide de General Motors. Ensemble ils conçurent le robot PUMA 500. Le robot PUMA (Programmable Universal Machine for Assembly) a été conçu par Vic Schienman et fut financé par General Motors et par The Massachussets Institute of Technology au milieu des années 70. Le système de ce robot est composé d'un bras manipulateur permettant d'assembler des composants industriels et de son ordinateur de commande. Ce robot est le robot d'assemblage le plus rependu dans l'industrie des années 70 .

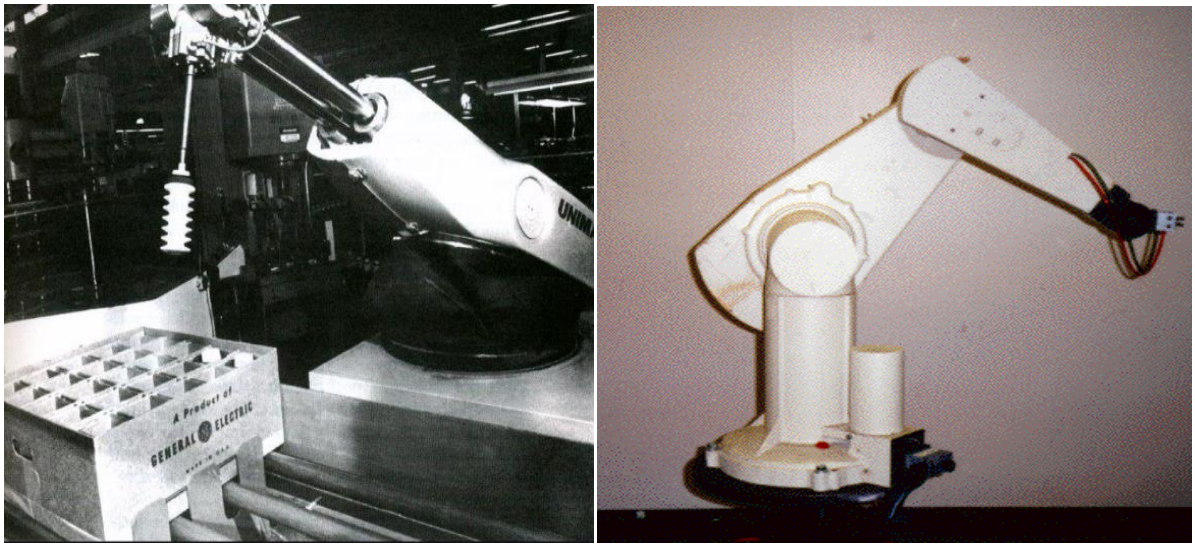

Figure 1.2 : Robotique industrielle (unimate , puma)

En 1985, Reymond Clavel a imaginé le Robot Delta qui possède un bras de manipulation formé de 3 parallélogrammes. Son brevet tombe dans le domaine public en 2007 et différents constructeurs devraient alors sortir leur propre robot delta.

Le Jet Propulsion Laboratory (JPL) développe un robot industriel hexapode (à 6 pattes) du nom de Lemur. Lemur aura pour mission de monter, assembler et réparer des installations spatiales. Pesant moins de 5 kg, il offre la possibilité innovante d'adapter différents outils sur chacun de ses membres.

Selon l'étude robotique de la Fédération Internationale de Robotique (IFR) en 2012, il y a au moins 1 153 000 robots industriels opérationnels fin 2011 dans le monde.

Avec l'apparition de la robotique industrielle, les robots étaient conçus pour remplacer les ouvriers dans les tâches pénibles, répétitives ou dangereuses (peinture, soudure…). Aujourd'hui avec le développement de l'électronique, de l'informatique, de la mécanique et aussi de l'automatique, la technologie robotique a progressé. La recherche dans le domaine de la robotique est dirigée vers le développement de robots dévoués à des tâches bien différentes que celles demandées par l'industrie. Par exemple des robots travaillant en mode automatique ou semi-automatique et qui ont souvent pour objectif d'interagir avec des humains et de les aider dans leurs tâches (surveillance, manutention d'objets lourds…). Ils sont dotés d'une intelligence qui leur donne une certaine autonomie.

Ainsi donc, le développement important de l'intelligence artificielle et de la robotique font que de nouveaux robots apparaissent constamment et l'utilisation de systèmes robotiques apparait aujourd'hui dans plusieurs domaines d'activité : la médecine, la défense, la recherche etc.… [4]

I.4 Architecture du robot :

1. Actionneur = moteur, vérin…

2. Axe = articulation.

3. Corps = segment.

4. Organe terminal.

5. Effecteur = outil.

6. Base

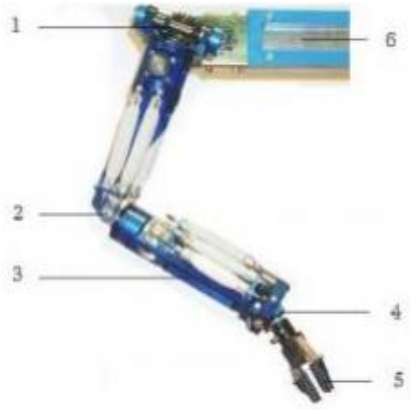

Figure 1.3 : Architecture d'un robot

I.4.1 La base :

La base du manipulateur est fixée sur le lieu du travail. Ceci est le cas de la quasi-totalité des robots industriels.

I.4.2 Le porteur :

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

I.4.2.1 Segment :corps solides rigides susceptibles d'être en mouvement par rapport à la base du porteur, et les uns par rapport aux autres,

I.4.2.2 Articulation : Une articulation lie deux corps successifs en limitant le nombre de degré de liberté, de l'un par rapport à l'autre , et on cite deux types d'articulations :

Articulation rotoïde : Il s'agit d'une articulation de type pivot, notée R, réduisant le mouvement entre deux corps à une rotation autour d'un axe commun. La situation relative entre les deux corps est donnée par l'angle autour de cet axe (figure 1.4)

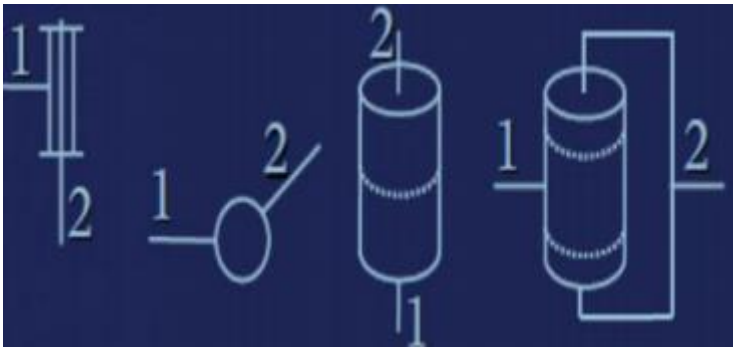

Figure.1.4 : Représentation d'une articulation rotoide .

Articulation prismatique : Il s'agit d'une articulation de type glissière, notée P, réduisant le mouvement entre deux corps à une translation le long d'un axe commun. La situation relative entre les deux corps est mesurée par la distance le long de cet axe (figure 1.5)

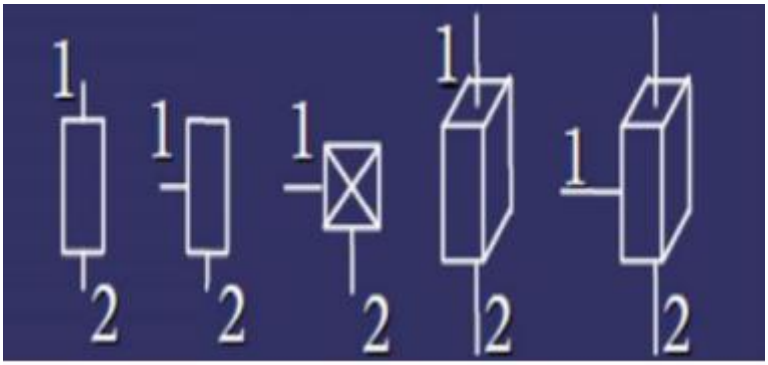

Figure 1.5 : Représentation d'une articulation prismatique.

I.4.3 L'actionneur :

Pour être animé, la structure mécanique articulée comporte des moteurs le plus souvent associés à des transmissions (courroies crantées), l'ensemble constitue les actionneurs. Les actionneurs utilisent fréquemment des moteurs électriques à aimant permanent, à courant continu, à commande par l'induit. On trouve de plus en plus de moteurs à commutation électronique (sans balais), ou, pour de petits robots, des moteurs pas à pas. Pour les robots devant manipuler de très lourdes charges (par exemple, une pelle

mécanique), les actionneurs sont le plus souvent hydrauliques, agissant en translation (vérin hydraulique) ou en rotation (moteur hydraulique). (Les actionneurs pneumatiques sont d'un usage général pour les manipulateurs à cycles (robots tout ou rien). Un manipulateur à cycles est une structure mécanique articulée avec un nombre limité de degrés de liberté permettant une succession de mouvements contrôlés uniquement par des capteurs de fin de course réglables manuellement à la course désirée (asservissement en position difficile dû à la compressibilité de l'air). [5]

I.4.4 L'organe terminal :

On regroupe tout dispositif destiné à manipuler des objets (dispositifs de serrage, dispositifs magnétiques, à dépression, …), ou à les transformer (outils, torche de soudage, pistolet de peinture, …).
En d'autres termes, il s'agit d'une interface permettant au robot d'interagir avec son environnement. Un organe terminal peut être multifonctionnel, au sens où il peut être équipé de plusieurs dispositifs ayant des fonctionnalités différentes. Il peut aussi être monofonctionnel, mais interchangeable. Un robot, enfin, peut-être multi-bras, chacun des bras portant un organe terminal différent. On utilisera indifféremment le terme organe terminal, préhenseur, outil ou effecteur pour nommer le dispositif d'interaction fixé à l'extrémité mobile de la structure mécanique, exemple : pistolet pour la soudure dans les robots industriels. [5]

I.5 Caractéristiques d'un robot :

I.5.1. Géométriques :

• Nombre d'axes (mus par un actionneur)

• Architecture (série ou parallèle)

• Chaînage des articulations.

• Nombre de degrés de liberté.

Exemples (P : Prismatique / R : Rotation / DL : Degrés de liberté) :

- 3 axes, série, RRR, 3DL.

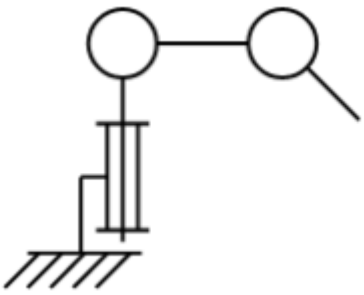

Figure 1.6 : Robot avec 3 axes, série, RRR et 3DL.

- 3 axes, série, PPP, 3DL.

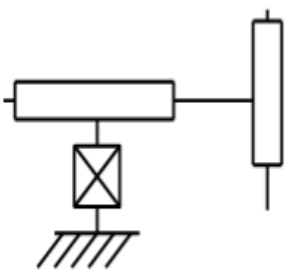

Figure 1.7: Robot avec 3 axes, série, PPP, 3DL.

- 4 axes, parallèle, RP+RP, 3DL.

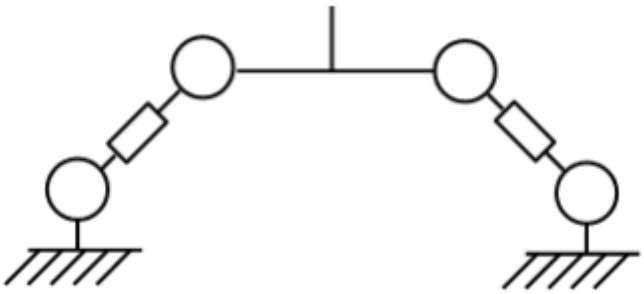

Figure 1.8 : Robot avec 4 axes, parallèle, RP+RP, 3DL.

I 5.2 Volume de travail :

Le volume de travail est accessible par l'outil du robot est le volume qui peut balayer l'outil. Le volume dépend de la géométrie du robot, de la longueur des segments et du débattement des articulations [6].

I 5.3 Précision / Répétabilité :

On dit que le positionnement absolu est imprécis lorsqu'il dépasse 1mm, alors on a l'une des erreurs suivantes:
- Erreurs de modèle géométrique.
- Erreurs de quantification de la mesure de position.
- Flexibilité.

Répétabilité : la répétabilité d'un robot est l'erreur maximale de positionnement répété de l'outil en tout point de son espace de travail. En général, la répétabilité ne doit pas dépasser 0.1 mm [6].

I.5.4. Performances dynamiques :

5.5 Vitesse maximale : C'est la vitesse de translation ou de rotation de chaque axe du robot. Souvent les constructeurs donnent une vitesse maximale de l'outil ou de l'organe terminal [6].

5.5 Accélération maximale : Elle dépend de l'inertie, donc de la position du robot. Pour chaque axe, cette accélération est donnée dans la configuration la plus défavorable (inertie maximale, charge maximale) [6].

I.5.5. Charge utile :

C'est la charge qui garantie une durée de vie la plus longe possible. Donc, c'est la charge maximale que peut porter le robot sans perturber ni la répétabilité ni les performances dynamiques du robot. Cette

charge utile est nettement inférieure à la charge maximale que peut porter le robot et est directement dépendante des actionneurs [6].

I.6 Avantages et inconvénients des robots :

Un système robotique consiste non seulement des robots mais au10ssi d'autres dispositifs et systèmes qui sont utilisés avec le robot pour effectuer la tâche nécessaire.

I.6.1. Avantages :

- Robotique et automatisation peut dans de nombreuses situations d'accroître la productivité, la sécurité, l'efficacité, la qualité et la cohérence des produits ;
- Robots peuvent travailler dans un environnement dangereux, sans le besoin de soutien de la vie, ou les préoccupations concernant la sécurité ;
- Robots n'ont pas besoin de l'éclairage, la climatisation, de ventilation et de protection contre le bruit ;
- Robots travaillent continuellement, sans ressentir une fatigue ou l'ennui, et ne nécessitent pas une assurance médicale ou de vacances ;
- Robots sont de précision répétable à tous les moments, sauf si quelque chose arrive à eux ou ils s'usent ;
- Robots peuvent être beaucoup plus précis que les humains. Précision linéaire d'un robot typiquement est de 20 à 10 microns ;

I.6.2. Inconvénients :

L'inconvénient des robots est qu'ils manquent de capacité de réagir en cas d'urgence, à moins que les situations comprises et les réponses sont inclut dans le système. Les mesures de sécurité nécessaires pour s'assurer qu'ils ne lèsent pas les opérateurs et n'endommagent les machines qui travaillent avec eux.

- Réponse inadéquate ou mal ;
- Manque de pouvoirs prendre une décision ;
- Consommation de l'énergie ;
- Peuvent causer des dommages à des autres appareils, et la blessure de l'homme ;

I.7Types des robots :

Dans l'espace tridimensionnel, un corps rigide libre peut se déplacer selon six degrés de libertés (ddl) :

Trois translations et trois rotations. On utilise le terme pose pour désigner la localisation du corps par rapport à un référentiel. Une pose est composée d'une position et d'une orientation.

Pour placer un corps rigide n'importe où dans l'espace tridimensionnel, on a besoin d'un robot avec minimum six articulations motorisées, c.à.d. d'un robot à six ddl figure (1.1.a) . La grande majorité des robots industriels sont de type sériel.

I.7.1 Le robot sériel : Un robot sériel est composé d'une série de segments reliés par des articulations motorisées rotoïde (en rotation) ou prismatiques (en translation). Dans certaines applications, on n'a pas besoin de déplacer les objets selon six ddl mais seulement selon cinq ou même quatre ou trois ddl. La figure (1.1.b) illustre un robot sériel à cinq ddl utilisé pour la palettisation. Le robot de la figure (1.9) est lui aussi utilisé pour la palettisation, mais il a seulement quatre ddl.

Figure 1.9 : Robot sériel pour palettisation à quatre ddl

I.7.2. Le robot parallèle : Dans un robot parallèle, l'effecteur est relié à la base via plusieurs < bras >, et la plupart des articulations ne sont pas motorisées. Les robots parallèles peuvent eux aussi avoir six, cinq, quatre, trois ou même deux ddl. Les robots parallèles à six ddl les plus connus sont les hexapodes, comme ceux qui déplacent les cockpits des simulateurs de vol. Les robots parallèles sont généralement plus rigides et plus rapides que les robots sériels. En revanche, ils sont beaucoup plus difficiles à étudier et il en existe de milliers d'architectures différentes.

Figure 1.10: Un robot parallèle

I.7.3. Le robot cartésien et le robot SCARA : De retour aux robots sériels, le robot illustré à la figure (6.3) (c) est un robot cartésien alors que le robot de la figure (1.11) (d) est un robot SCARA. Ces robots sont tous à quatre ddl (trois translations et une rotation autour d'un axe vertical) et servent à déplacer rapidement des petits objets.

Figure 1.11 : (c) est un robot cartésien, (d) est un robot SCARA

I.7.4. Les robots redondants à sept ddl :

Enfin, il existe aussi des robots à sept ddl (c.à.d. avec sept articulations motorisées) comme le robot illustré à la figure (1.12) (a), mais ils sont rarement utilisés. L'avantage d'un tel robot redondant est l'existence d'une infinité de possibilités pour atteindre une pose désirée, ce qui permet au robot de contourner des obstacles. Beaucoup plus souvent, on monte un robot a six articulations sur un guide linéaire la figure (1.12) (b) ou sur une table pivotante, ce qui résulte aussi en un système robotique redondant. Cependant, la raison principale n'est pas de contourner des obstacles mais d'augmenter l'espace de travail du robot (l'ensemble de poses que l'effecteur du robot peut atteindre).

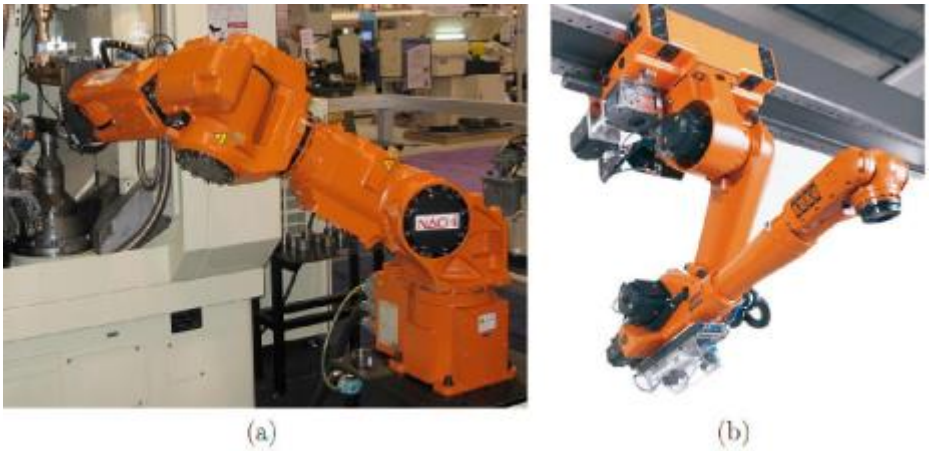

Figure 1.12: Robots redondants.

I.8. Domaine d'application :

Le but premier d'un robot est d'effectuer des tâches répétitives et/ou précises. Les robots permettent également d'effectuer des tâches dans des environnements de travail trop dangereux pour l'Homme. On peut les regrouper par Dull-Dirty-Dangerous. Les robots sont généralement présents dans différents domaines d'application tels que l'industriel, médical, médical, militaire et spatial.

I.8.1 Domaine industriel :L'objectif de ces robots est de remplacer l'homme dans des activités fastidieuses ou onéreuses pour l'employeur. Les robots ont donc commencé à être utilisés dans les chaînes d'assemblage industrielles. Dans ces chaînes d'assemblage, on retrouve des robots soudeurs, manipulateurs, peintres … (Fig.1.13)

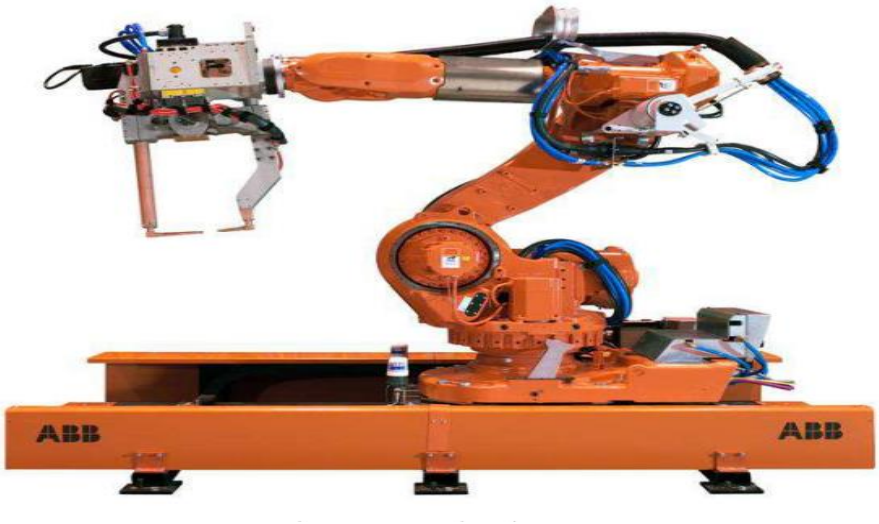

Fig.1.13 : Bras soudeur d'ABB

I.8.2 Domaine médical : Les robots commencent à être de plus en plus dans le domaine médical, qu'il s'agisse de « simples » échographies ou d'opérations chirurgicales plus délicates. En fait ces robots ne sont pas complètement autonomes mais ils assistent les médecins ou chirurgiens, jusqu'à permettre des opérations médicales à distance (télémédecine). On parle de surgétique c'est-à dire tout ce qui consiste à introduire les derniers outils des technologies informatiques et robotiques dans la pratique médico-chirurgicale.
Cette pratique de « chirurgie assistée » est émergente donc bien que peu répandue, elle est en phase de devenir la du futur. (Fig1.14) [7]

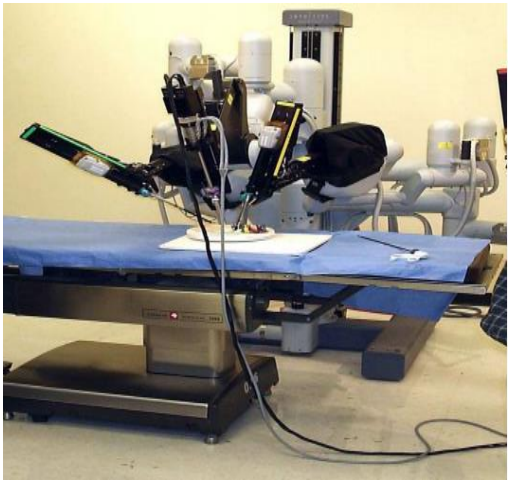

Fig.1.14 : Robot Da Vinci

I.8.3Domaine agricole : Après des décennies d'expérimentation et de tâtonnements, les robots ont enfin fait leur entrée à la ferme. Cette machine totalement autonome fonctionne grâce à l'énergie solaire et circule dans les rangées de plantations pour surveiller et analyser les plants (voir figure 1.15).

Ce robot a déjà passé avec succès de nombreux tests réalisés dans des champs de légumes mais se contente de surveiller la « bonne santé » et plantations. Grâce à ces nombreux capteurs, senseurs et caméras, il détecte rapidement d'éventuelles anomalies (présence de mauvaises herbes, animaux nuisibles, croissance trop faible) et avertit l'exploitant agricole qui peut ainsi prendre immédiatement les mesures appropriées.

Fig.1.15 : robot utilisé en agriculture

I.8.4Domaine militaire : Les robots sont de plus en plus utilisés dans le domaine militaire. En effet, la miniaturisation permet aujourd'hui de créer des robots discrets mais dotés de nombreux capteurs, ce qui est idéal pour des missions d'espionnage ou d'éclairement, comme le montre la figure(1.16) suivante. De plus, certains robots sont équipés d'un armement pour évoluer en milieu hostile, dans le but de remplacer les soldats pour limiter les pertes humaines.

Figure 1.16: Robot utilisé dans le domaine militaire

I.8.5 Domaine spatial : Les robots spatiaux sont destinés à explorer des environnements où l'homme ne peut pas se rendre. C'est-à-dire des environnements souvent mortels pour l'homme.

Aujourd'hui, l'histoire de la conquête spatiale est devenue indissociable de celle de la robotique et à ce moment même plusieurs robots sont en activité, aussi bien sur la station spatiale internationale que sur la planète Mars. Les futures missions d'exploration feront elles aussi appel aux robots et à leur IA , que ce soit pour relever de nouveaux défis ou pour mieux connaître notre système solaire. Mais ces machines s'avèreront être de plus en plus variées puisque toutes les tailles, toutes les fonctions et tous les modes de déplacements sont à l'étude afin de développer des robots capables d'assurer des tâches très distinctes, de façon autonome ou en parfaite synergie avec les humains.

I.8.6Domaine des services : La révolution de la robotique a conduit ces dernières années à voir de nombreux robots s'installer chez les particuliers pour effectuer des tâches à la place de leur possesseur, la figure (1.17) illustre le robot de service ASIMO. En effet, ceux-ci sont capables de faire le ménage, tondre la pelouse, nettoyer la piscine etc. Ce qui conduit certains clients (aisés) à se procurer ces domestiques contemporains. Enfin, la robotique autrefois réservée à des applications précises ou coûteuses, est aujourd'hui de plus en plus utilisée à titre ludique. En effet, les robots compagnons par exemple sont des objets de plus en plus convoités : les applications « basiques » de jouet pour enfant, jusqu'à l'humanoïde destiné à remplacer une présence humaine.

Figure 1.17 : Robot de service ASIMO

Enfin, la robotique autrefois réservée à des applications précises ou coûteuses, est aujourd'hui de plus en plus utilisée à titre ludique. En effet, les robots compagnons par exemple sont des objets de plus en plus convoités : les applications « basiques » de jouet pour enfant, jusqu'à l'humanoïde destiné à remplacer une présence humaine.

I.9 Conclusion :

Dans ce chapitre on a présenté les robots industriels les plus utilisés à travers le monde entier ainsi que leurs fabricants, leur historique en industrie et les différents champs d'applications. Quelques propriétés des ces robots sont évoqués dans ce chapitre qui vont servir dans les prochains chapitres.

.

Chapitre 2

Planification mathématique des taches

II.1 Introduction :

Pour commander ou simuler le comportement d'un système mécanique articulé (robot), on doit disposer d'un modèle. Plusieurs niveaux de modélisation sont possibles selon les objectifs, les contraintes de la tache et les performances recherchées.

Les modèles géométriques direct et inverse qui expriment la situation de l'organe terminal en fonction des variables articulaire et inversement.

Les modèles cinématiques direct et inverse qui expriment les vitesses de l'organe terminal en fonction des variables articulaires et inversement

Les modèles dynamiques direct et inverse définissant les équations du mouvement du robot qui permettent d'établir les relations entre les couples ou forces exercées par les actionneurs et les positions, vitesses, accélérations des articulations. [8]

II.2 Modèle Géométrique des robots :

La Conception et la commande des robots nécessitent le calcul de certains modèles mathématiques tel que le modèle géométrique direct qui expriment la situation de l'organe terminal en fonction des vitesses articulaires du mécanisme et inversement.

La modélisation du robot de façon systématique et automatique exige une méthode adéquate pour la description de leur morphologie. Plusieurs méthodes et notion on été proposées, la plus répandue est celle de Denavit-Hartenberg mais cette méthode, développée pour des structures ouvertes simples, présente des ambigüités lorsqu'elle est appliquées sur des robots ayant des structures fermées ou arborescente. C'est pourquoi, on utilise la notion de Khalil et Kleinfinger qui permet la description homogène, et avec un nombre minimum de paramètres, des architectures ouvertes simples et complexes des systèmes mécaniques articulés. [8]

II.2.1.Convention de Denavit-Hartenberg :

Méthodologie à suivre pour décrire les robots à structure ouverte simple.

Une structure ouverte simple est composée de n+1 corps noté C0 .....Cn et de n articulations, le corps C0 désigne la base du robot et le corps Cn le corps qui porte l'organe terminal.

L'articulation j connecte le corps Cj au corps Cj-1 figure (2.1). [8]

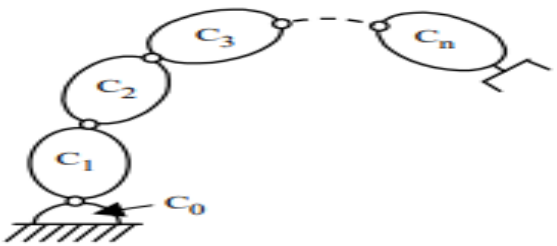

Figure 2.1 : Robot à structure ouverte simple

La méthode de description est basée sur le principe suivant:

II.2.1.1. Principe :

-Fixer des repères à chaque corps du robot.

-Calculer les matrices homogènes entre chaque corps.

-Calculer la matrice homogène entre base et organe terminal

II.2.1.2 Hypothèses :

On suppose que le robot est constitué d'un chainage de n+1 corps liés entre eux par n articulations rotoides ou prismatiques. A chaque corps, on associe un repère Ri. Les repères sont numérotes de 0 à n. La i ème articulation, dont la position est notée qi, est le point qui relie les corps i-1 et i.

Le repère Rj fixé au corps Cj, est défini de sorte que :

-L'axe Zj est porté par l'axe de l'articulation j.

-L'axe Xj est porté par la perpendiculaire commune aux axes Zj et Zj+1 .Si les axes Zj et Zj+1 sont parallèles ou colinéaires, le choix de Xi n'est pas unique. [8]

II.2.2 Les paramètres de Denavit-Hartenberg :

Le passage du repère Rj-1 eu repère Rj s'exprime en fonction des quatre paramètres géométrique suivants: figure (2.2)

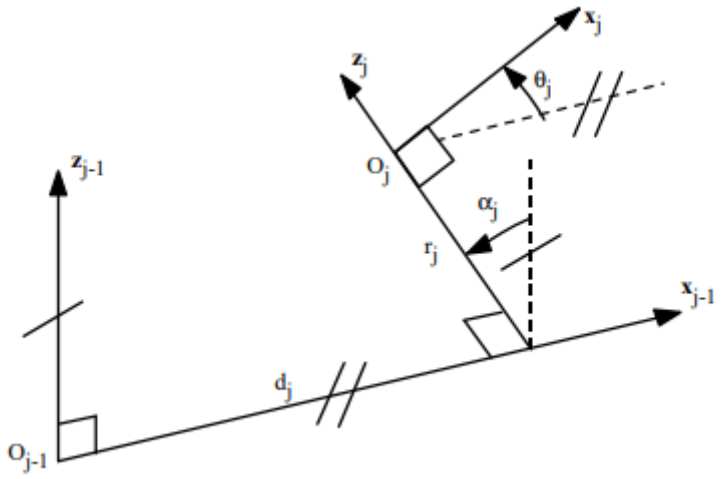

Figure (2.2) : Passage du repère.

1/ ∝j : Angle entre les axes Zj-1 et Zj correspondant à une rotation autour de Xj-1

2/ dj : Distance entre Zj-1 et Zj le long de Xj-1

3/ θj : Angle entre l'axe xj-1 et x et xj correspondant à une rotation autour de zj.

4/ rj : Distance entre xj-1 et x et xj le long de zj.

-Si l'articulation i est de type prismatique, alors di est variable

-Si l'articulation i est de type rotoide, alors θi est variable.

La variable articulaire qj associée à la j ième articulation est définie par:

$$q_j = \bar{\sigma}_j \theta_j + \sigma_j r_j$$

Avec:

σ j = 0 si l'articulation j est rotoide

σ j = 1 si l'articulation j est prismatique

A partir de cette description on peut définir la matrice de transformation homogène définissant le repère Rj dans le repère Rj-1

On a

$$^{j-1}T_j = \mathbf{Rot}(\mathbf{x}, \alpha_j)\,\mathbf{Trans}(\mathbf{x}, d_j)\,\mathbf{Rot}(\mathbf{z}, \theta_j)\,\mathbf{Trans}(\mathbf{z}, r_j)$$

$$= \begin{bmatrix} C\theta_j & -S\theta_j & 0 & d_j \\ C\alpha_j S\theta_j & C\alpha_j C\theta_j & -S\alpha_j & -r_j S\alpha_j \\ S\alpha_j S\theta_j & S\alpha_j C\theta_j & C\alpha_j & r_j C\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

:

On remarque que la matrice de rotation (3x3) j- 1 Aj peut être obtenue par :

$$^{j-1}A_j = rot(x,\alpha_j)rot(z,\theta_j)$$

La matrice de transformation définissant R j-1 dans R j est donnée par :

$$^{j}T_{j-1} = \begin{pmatrix} ^{j-1}A_j & -{}^{j-1}A_j^{T}\,{}^{j-1}P_j \\ 0_{1,3} & 1 \end{pmatrix}$$

$$ {}^{j}T_{j-1} = \begin{pmatrix} & & & -d_j C\theta_j \\ & {}^{j-1}A_j{}^{T} & & d_j S\theta_j \\ & & & -r_j \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

II.3 Le modèle géométrique direct :

Le modèle géométrique direct (MDG) est l'ensemble des relations qui permettent d'exprimer la situation de l'organe terminal, c.à.d les coordonnées opérationnelles du robot, en fonction des coordonnées articulaires. Dans le cas d'une chaine ouverte simple, il peut être représenté par la matrice de passage 0 T n : [8]

$$ {}^{0}T_n = {}^{0}T_1\,(q_1)\,{}^{1}T_2\,(q_2)\,\ldots\ldots\,{}^{n-1}T_n\,(q_n) $$

Le modèle géométrique direct du robot peut être représenté par la relation:

$$ X = f(q) $$

q étant le vecteur des variables articulaires tel que :

$$ q = [q_1 q_2 \ldots\ldots q_n]^T $$

Les coordonnées opérationnelles sont définies par:

$$ X = [X_1 X_2 \ldots\ldots X_n]^T = [P_x P_y P_z S_x S_y S_z\; n_x n_y n_z a_x a_y a_z]^T $$

On a S= n x a :

$$ X = [P_x P_y P_z\; n_x n_y n_z a_x a_y a_z]^T $$

II.4Le modèle géométrique inverse :

Le modèle géométrique direct d'un robot permet de calculer les coordonnées opérationnelles donnant la situation de l'organe terminal en fonction des coordonnées articulaire, le modèle inverse consiste à calculer les coordonnées articulaires correspondant à une situation données de l'organe terminal. Lorsqu'elle existe, la forme explicite qui donne toutes les solutions possibles constitue le modèle géométrique inverse.

$$ q = f^{-1}(x) $$

Pas de méthode analytique systématique pour calculer le MGI. Le mieux est de reprendre les équations du MGD et de mener le calcul à l'envers. Dans le cas où n = 6, l'existence d'un poignet sphérique permet de débuter la résolution par :

$$P_x = x_1 \ an_{xx} \ m + 1_{zx};$$

$$P_y = x_2 \ an_{xy} \ m + 1_{zy};$$

$$P_z = x_3 \ an_{xz} \ m + 1_{zz};$$

Ensuite résolution au cas par cas pour exprimer les qi, pour

i = 1,2,…, n en fonction de px, py, pz et des cosinus directeurs. [8]

II.5 Modèle cinématique du robot :

II.5.1 Modèle cinématique direct :

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

$$\dot{X} = J(q)\dot{q}$$

Ou J(q) désigne la matrice jacobéenne de dimension (m×n) du mécanisme, égale à ax/aq et fonction de la configuration articulaire q. La même matrice jacobéenne intervient dans le calcul du modèle différentiel direct qui donne la variation élémentaire dX des coordonnées opérationnelles en fonction des variations élémentaires des coordonnées articulaires dq, soit : [8]

$$dX = J(q)\, dq$$

$$J(q) = \frac{\partial f_i(q)}{\partial q_j}$$

avec i=1,.....,m et j=1,.....,n

Le modèle cinématique direct peut être mis sous la forme :

$$\dot{X} = \begin{pmatrix} \Omega_p & 0_3 \\ 0_3 & \Omega_r \end{pmatrix} \begin{pmatrix} {}^0A_j & 0_3 \\ 0_3 & {}^0A_j \end{pmatrix} \begin{pmatrix} I_3 & -{}^i\widetilde{L}_{j,n} \\ 0_3 & I_3 \end{pmatrix} {}^i j_{n,j}\dot{q}$$

Ou sous forme condensée:

$$\dot{X} = {}^0J_n\dot{q}$$

II.5.1.1 Intérêts de la matrice jacobéenne :

a/ Elle est la base du modèle différentiel inverse, permettant de calculer une solution local des variables articulaires q connaissant les coordonnées opérationnelles X.

b/ En statique, on utilise le jacobéen pour établir la relation liant les efforts exercés par l'organe terminal sur l'environnement aux forces et couples des actionneurs.

c/ Elle facilite le calcul des singularités et de la dimension de l'espace opérationnel accessible du robot.

II.5.2. Modèle cinématique inverse :

L'objectif du modèle cinématique inverse est de calculer, à partir d'une configuration q donnée, les vitesses articulaires q qui assurent au repère terminal une vitesse optimale X imposée. Cette définition est analogue à celle du modèle différentiel inverse : ce dernier permet de déterminer la différentielle articulaire dq correspondant à une différentielle des coordonnées opérationnelles dX spécifiée.

Pour obtenir le modèle cinématique inverse, on inverse le modèle cinématique direct en résolvant un système d'équation linéaires ; la mise en œuvre peut être faite de façon analytique ou numérique;

- la solution analytique a pour avantage de diminuer considérablement le nombre d'opération, mais on doit traiter tous les cas singulier

- Les méthodes numériques sont plus générales, la plus répandue étant fondée sur la notion de pseudo-inverse : les algorithmes traitent de façon unifiée les cas réguliers, singuliers et redondants ; elles nécessitent un temps de calcul relativement important .

Dans ce cas, la matrice jacobéenne J est carrée d'ordre n et son déterminant est non nul:

On calcule J-1 , la matrice inverse de J, qui permet de déterminer les vitesses articulaire q grâce à la relation:

$$\dot{q} = J^{-1}\dot{X}$$

Lorsque la matrice J a la forme suivante:

$$J = \begin{pmatrix} A & 0 \\ B & C \end{pmatrix}$$

Les matrices A et C étant carrées inversibles, il est facile de montrer que l'inverse de cette matrice s'écrit:

$$J^{-1} = \begin{pmatrix} A^{-1} & 0 \\ -C^{-1}BA^{-1} & C^{-1} \end{pmatrix}$$

A et C étant de dimension (3x3) .

II.6 L'orientation :

On peut représenter une orientation (rotation) par plusieurs représentations mathématiques, les plus utilisées sont:

- Les trois angles d'EULER
- Les trois angles de CARDAN
- Les matrices des cosinus directeurs
- Les quaternions

II.7 Principe de la description des tâches :

Dans la plupart des contrôleurs de robot actuels, la géométrie des trajectoires est décrite par une succession de repères.

Suivre une trajectoire consiste alors à amener en coïncidence sur ces repères un référentiel lié à l'organe terminal. Lorsque le nombre de composantes utilisées pour spécifier la tâche est inférieur au nombre de degrés de liberté du robot, il y a redondance de celui-ci vis-à-vis de la tâche et donc, une infinité de solution pour la réaliser. Cette description, minimale en ce sens qu'elle ne contraint que les degrés de liberté de la tâche ayant un rôle fonctionnel, est intéressante car elle permet de satisfaire des critères d'optimisation supplémentaires lors de son exécution.

Le formalisme retenu est celui du contact des surfaces usuelles de la mécanique (plan, cylindre, sphère) qui décrit les liaisons mécaniques simples tableau (2.2) illustrées par la figure (2.3). A ces six liaisons , on adjoint la liaison totale correspondant à un encastrement et les deux liaisons composées à un degré de mobilité :la liaison rotoïde et la liaison prismatique figure (2 .4).

La description des situations successives du repère terminal lié au robot est réalisée par une séquence de liaisons simples et composées, le choix d'une liaison étant dicté par les contraintes locales associe à la tâche.

	Plan	Cylindre	Sphère
Plan	Appui plan	Appui linéaire	Appui simple
Cylindre		Verrou	Anneau
Sphère			Rotule

tableau 2.1 : Liaisons mécaniques simples

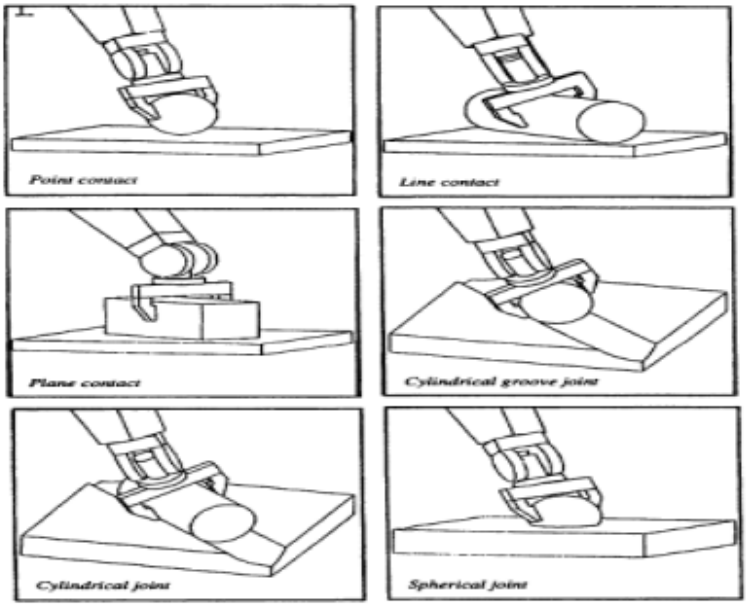

Figure 2.3 : Liaisons mécaniques simples

Le formalisme des liaisons mécaniques peut paraître difficile à utiliser. Une solution plus ergonomique consiste à décrire un déplacement en termes de liaison entre deux entités géométriques simples (point, droite, plan) appartenant l'une au robot, l'autre à l'environnement :

Une liaison rotule, par exemple, correspond à la mise en coïncidence de deux points. De même, les liaisons composées seront spécifiées par deux combinaisons simultanées d'éléments géométriques.

Le choix n'est pas unique : ainsi, une liaison rotoïde par exemple peut être issue de la réalisation d'une liaison droite sur droite et d'une liaison point sur plan ou d'une liaison droite sur droite et d'une liaison point sur plan ou d'une liaison droite sur droite et d'une liaison point sur point.

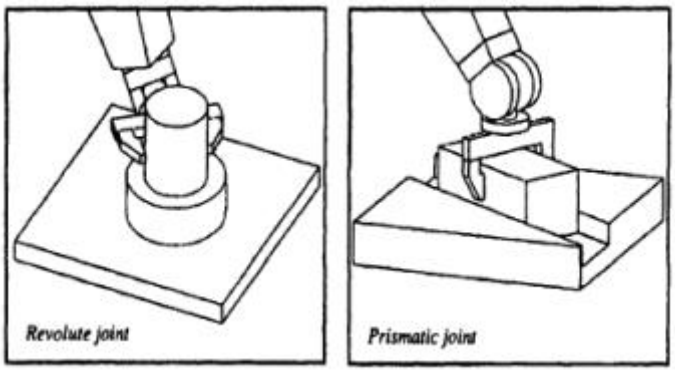

Figure 2.4 : Liaison rotoide et prismatique

Cette description géométrique est particulièrement bien adaptée à une programmation graphique des tâches. La figure (2 .5) montre l'exemple d'un assemblage cylindrique, réalisé avec l'application robotique du logiciel de CFAO CATIA dans lequel est implanté ce formalisme. Les différentes étapes sont les suivants :

a) Etat initial : sélection d'un point du robot et d'un point de l'environnement (rotule)

b) Le cylindre est positionné, avec une orientation quelconque, au-dessus du site d'assemblage ; sélection d'une droite du robot et d'une droite de l'environnement (verrou)

c) Les axes du tampon cylindrique et de l'alésage sont alignés ; sélection d'un point et d'une droite du robot d'une part, de l'environnement d'autre part (rotoïde)

d) Etat final : la tâche d'assemblage est terminée. On peut remarquer la rotation laissée libre autour de l'axe principal du cylindre entre les états c) et d) .

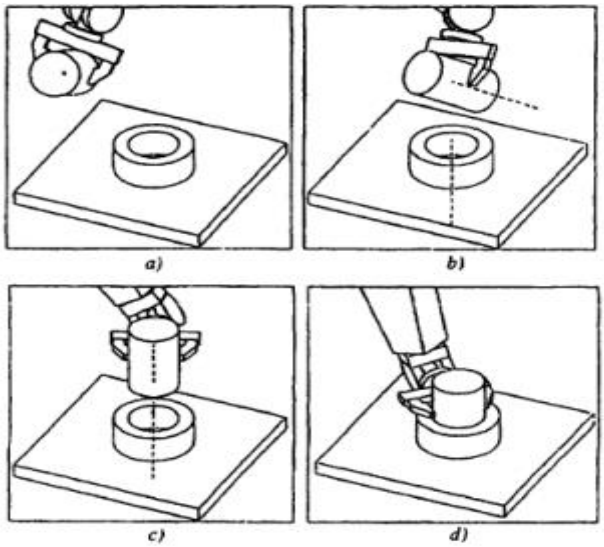

Figure 2.5 : Programmation graphique d'un assemblage à partir d'une description minimal des tâches.

II.8Modèles différentiels associés à une description minimale :

Pour prendre en compte ce type de description de tâche, on écrit le modèle différentiel définissant les relations différentielles du repère outil RE de la façon suivante :

$$\begin{pmatrix} {}^0dP_E \\ {}^0\delta_E \end{pmatrix} = \begin{pmatrix} {}^0A_n & 0_3 \\ 0_3 & {}^0A_n \end{pmatrix} \begin{pmatrix} {}^ndP_E \\ {}^n\delta_E \end{pmatrix} = \begin{pmatrix} {}^0A_n & 0_3 \\ 0_3 & {}^0A_n \end{pmatrix} \begin{pmatrix} I_3 & -{}^n\hat{P}_E \\ 0_3 & I_3 \end{pmatrix} \begin{pmatrix} {}^ndP_n \\ {}^n\delta_n \end{pmatrix}$$

$$= \begin{pmatrix} {}^0A_n & -{}^0A_n\hat{P}_E \\ 0_3 & {}^0A_n \end{pmatrix} {}^nJ_n dq$$

Le terme n PE définissant l'origine du repère RE exprimée dans Rn

Avec une description minimale de la tâche le modèle différentiel s'écrit :

$$dX = H\,{}^nJ_n dq$$

Où nJ n est de dimension (6 x n) et où H est une matrice de dimension (c x 6).

La caractérisation de la liaison c dépend du nombre d'équations de certaine associées à la tâche. On va commencer par la description de l'exemple le plus simple qui est le suivant.

II.9 Appui simple (point. Plan) :

Cette liaison consiste à amener un point OE de l'outil dans le plan Q. sa position dans le plan Q étant quelconque figure (2.6). Soit N le vecteur unitaire selon la normale au plan Q et soit OD un point arbitraire du plan Q. Le déplacement global à effecteur pour réaliser l'appui simple s'exprime dans le repère R0 par :

$$r = {}^0N^T \left({}^0P_d - {}^0P_E \right)$$

0 Pd et 0 Pe définissant les points OD et OE dans le repère R0.

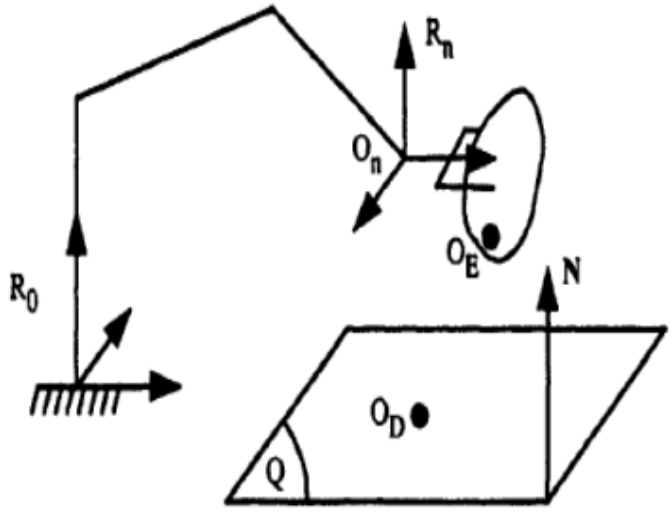

Figure 2.6 : Réalisation d'un appui simple

Le déplacement global r est réalisé par une succession de déplacements élémentaires à une seule composante tels que :

$$dX = dr = {}^0N^T \, {}^0dP_E = \left({}^0N^T \, {}^0A_n \quad -{}^0N^T \, {}^0A_n \, {}^n\hat{P}_E \right) {}^nJ_n dq$$
$$= {}^0N^T \, {}^0A_n \left(I_3 \quad -{}^n\hat{P}_E \right) {}^nJ_n dq$$

La matrice H est alors identifiée à 0 N t 0An (I3-nPe) et n'a qu'une seule ligne

.L'expression constitue le modèle différentiel de l'appui simple.

Remarque 1:

On peut résumer les équivalence liaison mécanique/spécification géométrique par le tableau ci-dessous :

Type de liaison	Elément de l'effecteur	Elément de l'environnement	Nombre d'équations indépendantes	Nombre total d'équations
Appui simple	Point	Plan	1	1
Appui linéaire	Droite	Plan	2	2
Appui plan	Plan	Plan	3	3
Anneau	Point	Droite	2	2
Rotule	Point	Point	3	3
Verrou	Droite	Droite	4	4
Rotoïde	Droite-Point	Droite-Point	5	7
prismatique	Plan-Plan	Plan-Plan	5	6

tableau 2.2 : Equivalence liaison mécanique/spécification géométrique

Remarque 2:

L'inconvénient majeur du modèle cinématique pour le transformateur de coordonnées est qu'il ne traduit qu'un comportement local du robot autour d'une configuration articulaire donnée. La trajectoire engendrée dépend donc de la configuration initiale du mécanisme.

II.10Conclusion :

Ce chapitre a été consacré aux outils mathématiques utilisés dans la modélisation, la simulation et la planification des robots manipulateurs.

Les programmes des différentes taches planifiées dans la robotique industrielle, que ça soit graphique ou par langage procédurale sont développés via ces outils mathématiques.

Nous avons montré comment, à partir d'une description utilisant le formalisme des liaisons mécaniques, il était possible de contraindre uniquement les degrés de liberté fonctionnels de la tâche.

.

Chapitre 3

Programmation des taches avec RobotStudio

III.1 Introduction :

Dans le processus continu, le robot est manipulé par une série de commandes et de fonctions stockées dans un fichier dit << programme robot>>. La trajectoire du robot doit être programmée avant son exécution. Deux modes de programmation des trajectoires robot sont couramment utilisés [9]

- la programmation en ligne (PEL)
- la programmation hors-ligne (PHL).

III.2 Programmation en ligne :

La programmation en ligne est aujourd'hui le mode de programmation le plus utilisé industriellement, il existe deux méthodes de programmation en ligne: l'apprentissage direct et l'apprentissage indirect point à point.

Figure 3.1: Programmation en ligne.

III.2.1 L'apprentissage direct :

Le principe de cette méthode de programmation consiste à mémoriser, lors de l'apprentissage, l'ensemble des configurations articulaires du robot pendant l'exécution de la tache ou pendant une simulation de celle-ci. L'opérateur manipule directement le robot à l'aide d'un pupitre d'apprentissage pour lui "apprendre" la tache à accomplir. L'acquisition de la trajectoire est continue. Ce type de programmation est simple et demande peu de formation pour le programmeur. Cette méthode est plutôt utilisée pour des applications qui exigent des mouvements de précision limitée, dans la projection sur les pièces simples ou le domaine de la peinture.

III.2.2 L'apprentissage indirect point par point :

Le principe utilisé par cette méthode est également basé sur une démonstration matérielle de la tache à réaliser. L'opérateur déplace le robot en mode manuel, utilisant pour cela le pupitre de commande à touche. Cependant, la configuration du robot n'est enregistrée qu'en certains points caractéristiques de la trajectoire. La liaison entre ces différents points est spécifiée par les caractéristiques de la trajectoire entre

ces points (linéaire, articulaire voire circulaire) et par la définition d'une vitesse de déplacement spécifiée par l'opérateur [10]

III.3 La programmation hors-ligne :

La programmation hors-ligne permet la génération des programmes avec une précision beaucoup plus importante. Le temps consacré sur site est fortement réduit. Le positionnement des trajectoires ainsi que la conception des outils peuvent être optimisés.

Un des principaux intérêts de ce type de méthode est qu'il ne nécessite aucune immobilisation de l'outil de production durant la phase de programmation du robot. On distingue 2 approches principales pour la programmation hors-ligne des robots: la programmation par langage et la programmation graphique à partir d'un système de CFAO (Conception et Fabrication Assistée par Ordinateur).

III.3.1 Programmation par langage robotique :

Les langages de programmation des robots sont proches des langages informatique classique. Ils proposent des instructions particulières pour commander les mouvements des robots. Ils sont performants pour définir des positions par calculs ou pour implanter des instructions conditionnelles ou des boucles.

Le principal inconvénient de ces langages est qu'ils restent généralement spécifiques à chaque marque de robot; il n'existe pas de standard reconnu. Cela oblige l'opérateur à connaitre un langage pour communiquer avec le robot. Les langages suivants peuvent être cités: RAPID (ABB); VAL II (UNIMATION), RAIL (AUTOMATIX), PALAW (KOMATSU). Dans notre étude nous allons utiliser le langage RAPID.

Ce type de programmation nécessite de connaitre à l'avance les points de passage, l'orientation de l'outil ainsi que la vitesse locale de déplacement pour chaque point.

III.3.2. Programmation graphique :

Cette méthode de programmation est une méthode plus pratique pour programmer des trajectoires robot sur les pièces de forme complexe. Les logiciels de CFAO actuels permettent de représenter et de modéliser les robots, les pièces à revêtir et l'environnement [11]

La figure (3.2) présente la programmation hors-ligne à l'aide d'un logiciel CFAO robotique.

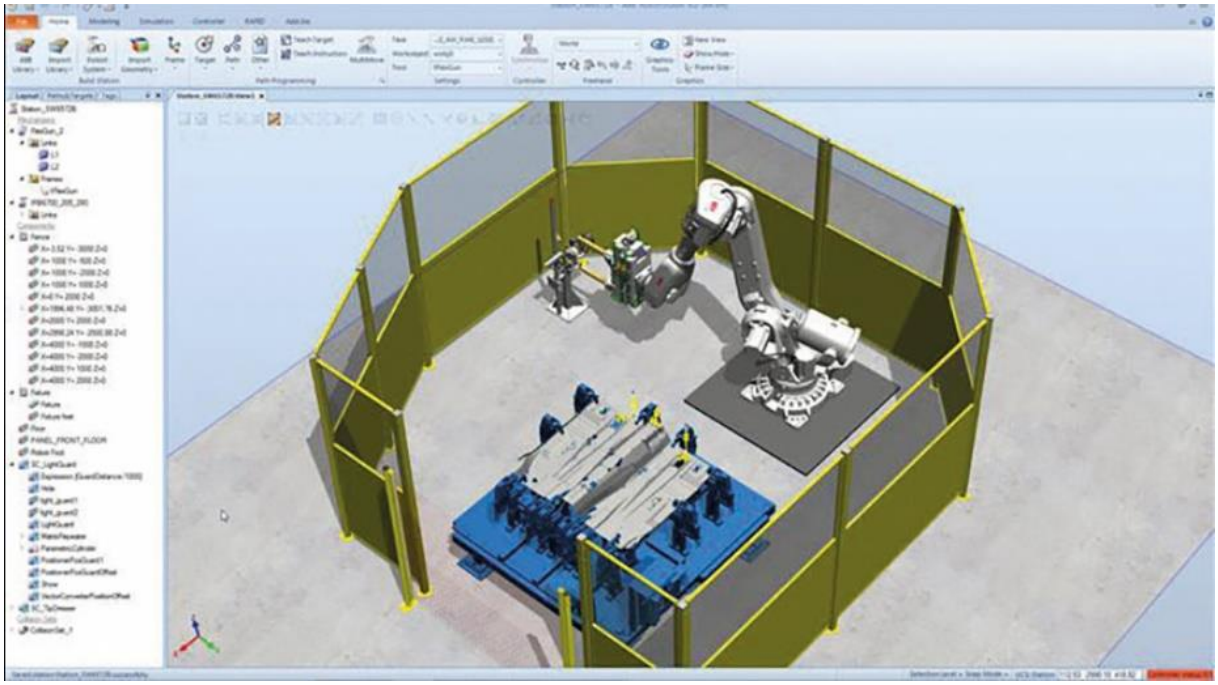

Figure 3.2: Programmation graphique.

Dans le cadre de cette étude, le logiciel de CFAO robotique dit "RobotStudio™ " est utilisé en raison de sa bonne compatibilité entre les robots virtuels et les robots réels.

RobotStudio™ est un logiciel d'ABB qui permet de simuler et de programmer hors-ligne des systèmes robots à l'aide d'un PC standard fonctionnant sous Windows.

III.4 RobotStudio ™ :

RobotStudio™ est fondé sur le " VIRTUAL CONTROLLER" d'ABB qui est une copie exacte du logiciel réel qui commande le robot en production. Il est ainsi possible d'effectuer des simulations très réalistes à l'aide de programmes de robots réels et de fichiers de configuration identiques à ceux utilisés dans l'atelier. Le Virtual Controller contient ainsi un " pupitre mobile d'apprentissage" virtuel qui permet d'utiliser le robot simulé exactement comme un vrai robot.

RobotStudio™ augmente la rentabilité du système robot en permettant d'effectuer des taches comme la formation, la programmation et l'optimisation sans gêner la production.

RobotStudio™ offre ainsi plusieurs avantages comme :

- Réduction des risques- vérification de nouvelles installations en créant rapidement sa propre station, en menant des études de faisabilité précises, et en vérifiant l'accessibilité, les collisions et les problèmes de singularité.

- Mise en service plus rapide- le temps consacré à la programmation peut être réduit et, plus important encore, cette tache peut être effectuée avant l'installation du système

- Changement plus rapide- il est possible de tester les modifications apportées à des installations existantes ou de mettre à jour les programmes de production de nouvelles pièces

- Optimisation des programmes- disposition de plusieurs outils qui facilitent l'optimisation des programmes de robot

- Environnement de simulation 3D- représentation 3D complète des systèmes robotiques

- Système de commande virtuel- système de commande de robot intégré à un PC standard

- Bibliothèque de robots- modèles exacts de la gamme de produit complète ABB

- Pupitre mobile virtuel d'apprentissage- parfait pour la formation hors -ligne des opérateurs

- Editeur de programmes RAPID- vérification automatique des erreurs de programme

- Simulateur des E/S- interaction avec des entrées/sorties simulées
- VBA(Visual Basic for Application)- permet de développer des fonctionnalités particuliers .

Son inconvénient principal est l'impossibilité de choisir les vitesses et les accélérations optimales de l'exécution des différentes taches planifiées, Robot studio permet le choix de vitesses prédéfinies, néanmoins une solution consiste à lui ajouter des extensions sous forme de programmes.

III.4.1 Station de RobotStudio™ :

D'une certaine façon, une station de RobotStudio™ correspond à une cellule de robot dans une usine. La station contient le robot, l'outil, les pièces de travail, les programmes et tout ce qui peut être utile lors de la simulation. Une seule station peut être ouverte lors de l'utilisation du logiciel.
Cette station peut être enregistrée dans un fichier. Les fichiers de station portent l'extension <<.stn>>. Un fichier de station contient:

- Un système de coordonnées dans lequel sont positionnés les objets de la station.

- Des références aux objets de la station.

- Des informations sur le système de commande virtuel utilisé dans la station.

III.5 La programmation :

III.5.1 Le langage RAPID :

Le langage de programmation des robots ABB s'appelle RAPID. Un programme en RAPID est un ou plusieurs fichiers de type texte (ASCII), non compilées. Le contrôleur du robot interprète le code, tout en validant la syntaxe du programme en entier. Un programme contenant des erreurs de syntaxe peut être chargé dans le contrôleur, mais ne peut pas être exécuté.

Il existe deux types de fichiers possibles. Le premier type est le plus important : les modules (.mod) qui contiennent le programme. Cela ne fait aucune différence pour le contrôleur du robot si le programme est écrit dans un seul ou dans plusieurs modules. L'utilisation de plusieurs modules se justifie uniquement dans la plus grande facilité de saisi et de réutilisation des modules par le programmeur. Par contre, il ne peut y avoir qu'un seul programme actif dans le contrôleur du robot ; c'est-à-dire qu'un seul des modules peut contenir une procédure appelée < main >.

Lors du chargement dans le contrôleur des différents modules, le contrôleur assumera automatiquement l'ensemble de modules comme un programme. Les modules peuvent être chargés séparément, un par un, ou en utilisant le deuxième type de fichier : le programme (.pgf), qui est simplement un fichier de type XML qui spécifie la liste des modules à charger.

Le contrôleur amorce le programme en partant par la procédure < main >. Celle-ci doit être au début d'un module, juste après la déclaration des variables globales à la tâche.

Le langage RAPID possède tous les types de données atomiques classiques : booléen, numérique, chaine de caractères, etc.

III.5.1.1 Syntaxe d'une déclaration :

Il est important de noter que le nom d'une procédure, d'une fonction ou d'une interruption ne peut pas avoir plus de 32 caractères, et ne peut par commencer par un caractère numérique ou par un caractère spécial. Cependant, noter que le tiret bas (<_ >) est toléré.

Il est également important de noter que les noms des modules, procédures, fonctions et registres mémoires doivent être uniques. Exemple, le contrôleur ne sera pas utiliser un module portant le même nom d'une routine ou encore comprendre l'assignation si une fonction porte le même nom d'une variable.

Le langage RAPID offre la possibilité au programmeur de se créer des registres mémoires afin de stocker des informations pertinentes au déroulement d'un projet. Son approche est similaire à un langage de programmation informatique. La déclaration est structurée et les registres que l'on désire accessibles de partout doivent être déclarés dans le début du module avant toutes instructions.

Il est impossible de déclarer une variable à mi-programme. L'ordre de classement des registres dans la déclaration n'est pas imposé. Il est possible de les regrouper par type, besoin, équipement, etc.

III.5.1.2 Le registre mémoire :

La définition d'un registre mémoire peut être de trois types :

***Constante (CONST) :**

Une constante représente une valeur statique lors de l'exécution du programme. Elle peut seulement recevoir une nouvelle valeur manuellement que ce soit lors d'une programmation en ligne ou hors-ligne.

*** Variable (VAR) :**

Une variable représente une valeur dynamique que le programme peut modifier lors de son exécution. Elle peut recevoir une nouvelle valeur en tout temps. Celle-ci n'est réinitialisée que lorsqu'on demande au contrôleur de réinitialiser un programme (opération appelée < PP to Main >).

*** Persistante (PERS) :**

Une persistante peut être décrite comme étant une variable < persistante > dans le temps. Lorsqu'on modifie sa valeur, sa définition est automatiquement mise à jour par le contrôleur afin de retenir toujours la dernière valeur. Il est important de noter que pour faire cette rétention, la variable doit-être déclarée avec une initialisation Les persistantes ne peuvent être déclarées qu'au niveau d'un module, et non à l'intérieur d'une procédure.

III.5.1.3 Les enregistrements :

Le langage RAPID possède également le type enregistrement. Un enregistrement est composé d'un mélange de données atomiques et/ou d'enregistrements. Chaque composante d'un enregistrement a un nom et peut être adressée en utilisant le nom de l'enregistrement suivi par le séparateur < . > suivi par le nom de la composante.

Les enregistrements les plus importants sont les poses (pose), les référentiels (wobjdata), la pose de l'effecteur combinée à la configuration du robot (robtarget) et les définitions d'outils (tooldata).

***Pose :**

Une pose est une représentation mathématique d'un référentiel par rapport à un autre. Dans RAPID, un enregistrement de type pose est composé d'un enregistrement de type position (pos) et d'un enregistrement de type orientation (orient), par exemple [[0; 0; 0]; [1; 0; 0; 0]]. Le langage RAPID utilise la représentation par quaternions .

***Wobjdata :**

L'enregistrement wobjdata (< work object data>) permet définir un référentiel de travail. Il est très rare, voire quasi impossible, d'avoir un référentiel de robot parfaitement aligné avec son milieu de travail il est donc souvent nécessaire de se définir un référentiel par rapport à un objet et/ou une surface de travail. De plus, les cellules étant parfois complexes, il n'est pas rare d'avoir plus d'un lieu de travail. Dans le cas d'un robot ABB, les enregistrements de type wobjdata doivent toujours être déclarés globaux et

persistantes (PERS). L'enregistrement wobjdata est une structure complexe composée des éléments suivants :

- robhold (bool) : Sert à définir si le robot tient l'outil, ou si l'outil est fixe.

-ufprog (bool) : Sert à définir si le référentiel se déplace dans l'espace ou s'il est fixe.

- ufmec (string) : Sert à définir le mécanisme à suivre si ufprog est faux.

-uframe (pose) : Sert à définir la pose du référentiel du lieu de travail (< user frame >) par rapport au référentiel de l'atelier wobj0 (figure 3.4). La valeur de cette composante provient le plus souvent d'un enseignement manuel à l'aide du FlexPendant ou de la fonction DefFrame.

- oframe (pose) : Sert à définir la pose de l'objet sur lequel on va travailler (< object frame >) par rapport au uframe (figure 3.3). Souvent, cette composante est laissée à sa valeur par sans effet, soit [[0; 0; 0]; [1; 0; 0; 0]] (c'est-à-dire que le référentiel oframe coïncide avec le référentiel uframe).

La valeur de cette composante, si modifiée, provient le plus souvent d'un enseignement manuel à l'aide du FlexPendant ou de la fonction DefFrame.

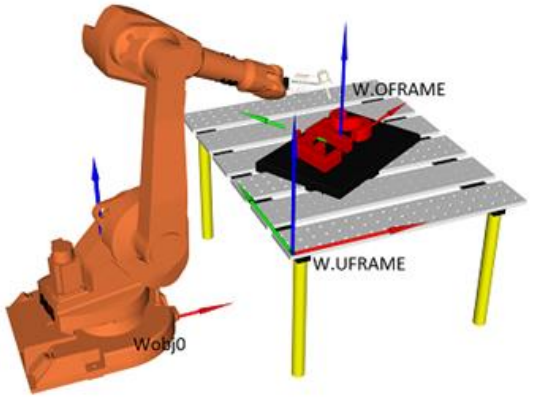

Figure 3.3: Référentiels faisant partie d'un enregistrement de type workoject,dont le nom est W.

***Robtarget :**

L'enregistrement de type robtarget est utilisé pour représenter une pose de l'outil du robot par rapport à un référentiel (wobjdata) non spécifié, ainsi que la configuration du robot et les positions des moteurs supplémentaires (tel qu'un guide linéaire ou une table pivotante). La structure de l'enregistrement robtarget se définit comme suit :

-trans (pos) : Contient les coordonnées x, y et z (en mm) d'un référentiel d'outil non spécifié par rapport à un référentiel de travail (wobjdata) non spécifié.

-rot (orient) : Contient le quaternion exprimant l'orientation du même référentiel d'outil par rapport au même référentiel de travail (wobjdata).

-robconf (confdata) : Permet de définir la valeur du cadran des articulations 1, 4 et 6, ainsi que le numéro de configuration (de 0 à 7, dans le cas des robots sériels à 6 ddl).

-extax (extjoint) : Permets d'interagir avec les articulations externes du robot. [3]

***Tooldata :**

L'enregistrement de type tooldata permet de définir un référentiel sur un outil ainsi que les caractéristiques physiques de l'outil.

L'outil de chaque robot est fixé à la bride du robot. En plus de contenir un référentiel physique, l'enregistrement va également contenir le poids, le centre de gravité et les moments d'inertie de l'outil. Il est important de bien définir ceux-ci si on veut optimiser la précision et la vitesse du robot. L'enregistrement de type tooldata est une structure composée principalement des trois niveaux suivants :

-robhold (bool) : Sert à définir si le robot tient l'outil, ou si l'outil est fixe. Dans le cas ou le robot tient l'outil cette valeur sera toujours égale à false.

-tframe (pose) : Permet de définir la pose du référentiel de l'outil par rapport au référentiel de la bride, appelé tool0.

- tload (loaddata) : Permet de définir la charge physique de l'outil et ainsi permettre au robot de connaitre les forces physiques générées par l'outil lors de son déplacement et ainsi optimiser sa vitesse et sa trajectoire.

III.5.1.4 Les commandes de déplacement :

Le but premier d'une commande de déplacement est de déplacer l'outil du robot d'une manière spécifique. Dans les commandes de déplacements les plus utilisées, on doit toujours spécifier un robtarget comme cible. Il existe trois types de déplacements dont la cible est un robtarget.

III.5.1.4.1 Les types de déplacements :

*** Déplacement linéaire :**

Dans un déplacement linéaire, l'origine du référentiel de l'outil spécifié suit une trajectoire linéaire (figure 3.4). La réorientation de l'effecteur, si nécessaire, se fait de façon automatique (l'opérateur ne peut ne pas influencer cette réorientation). Ce type de déplacement est très contraignant, car il oblige souvent le robot à réaliser des mouvements complexes et parfois brusques.

Figure(3.4): Déplacement linéaire de l'outil

Un déplacement linéaire peut également être limité parce qu'on appelle des singularités. Quand un programmeur rencontre une singularité dans un mouvement linéaire, il est fréquent que la seule solution soit un changement physique de la cellule de travail afin que

le robot ne repasse plus sur cette singularité. Les singularités ne sont pas liées à une marque de robot, mais bien au design mécanique ce qui signifie que certains robots en possèdent plus que d'autres et que toutes les marques de robot ont ce problème avec certains de leurs modèles. Par contre, chaque fabricant du robot a des limitations spécifiques additio-nnelles lors d'un déplacement linéaire.

Les paramètres de base d'une commande de linéaire sont comme suit :
MoveL robtarget, speeddata, zonedata, tooldata, \wobj:=wobjdata;
\wobj:=wobjdata;

1. MoveL : Commande pour un déplacement linéaire de l'endroit ou il se trouve vers le robtarget spécifié.

2. robtarget : Valeur venant d'une variable, fonction ou autre qui indique la destination de l'outil.

3. speeddata : Vitesse de croisière maximale que l'outil peut atteindre durant le déplacement.

4. zonedata : Précision d'arrêt ou d'approche de l'origine du référentiel de l'outil sur un robtarget à atteindre.

5. tooldata : Le référentiel de l'outil et les caractéristiques physiques de l'outil.

6. wobjdata : Paramètre optionnel qui défini par rapport à quel référentiel de travail on veut positionner le référentiel de l'outil à la pose définie dans le robtarget. Si ce paramètre est omis, le référentiel par défaut est celui de l'atelier (appelée wobj0).

*** Déplacement articulaire**

Dans un déplacement articulaire, la trajectoire suivie par l'effecteur du robot est difficilement prévisible (figure 3.5). Ce type de déplacement offre une grande liberté de mouvement sans risque de singularité durant le déplacement. Le robot réalise un déplacement articulaire en définissant les valeurs des articulations à la configuration ou il se trouve, ensuite en calculant les valeurs de celles-ci à la destination finale. Une fois le déplacement nécessaire de chaque articulation connu, le robot démarre toutes les articulations en même temps, à différentes vitesses, afin de toutes les l'outil arrêter en même temps, à la destination finale (définie par un robtaget).

Figure 3.5: Déplacement articulaire de (commande MoveJ).

et

Les paramètres de base d'une commande de déplacement articulaire sont comme suit :
MoveJ robtarget, speeddata, zonedata, tooldata, \wobj:=wobjdata;

1. MoveJ : Commande indiquant un déplacement articulaire de l'endroit ou il se trouve vers le robtarget spécifié.

2. robtarget : Valeurs venant d'une variable, fonction ou autre qui indique la destination de l'outil.

3. speeddata: Vitesse de croisière maximale que l'outil peut atteindre durant le déplacement.

4. zonedata : Précision d'arrêt ou d'approche de l'origine du référentiel de l'outil sur un robtarget à atteindre.

5. tooldata : Le référentiel de l'outil et les caractéristiques physiques de l'outil.

6. wobjdata : Paramètre optionnel qui défini par rapport à quel référentiel de travail on veut positionner le référentiel de l'outil à la pose définie dans le robtarget. Si ce paramètre est omis, le référentiel par défaut est celui de l'atelier (appelée wobj0).

*** Déplacement circulaire**

Dans un déplacement circulaire, l'origine du référentiel de l'outil spécifié suit une trajectoire circulaire passante par trois robtargets (le robtarget actuel, un intermédiaire et un final). Un déplacement circulaire a les mêmes contraintes qu'un déplacement linéaire puisque l'on force le trajet. Il est donc à risque pour ce qui concerne les singularités.

Les paramètres de base d'une commande de déplacement articulaire sont comme suit :

MoveC robtarget, robtarget, speeddata, zonedata, tooldata, \wobj:=wobjdata;

1. MoveC : Commande indiquant un déplacement circulaire par calcul de trois points (position courante, une valeur intermédiaire et une valeur finale).

2. robtarget (1^{er}) : Valeur venant d'une variable, fonction ou autre qui indique le robtarget intermédiaire par lequel l'outil doit passer.

3. robtarget (2^{e}) : Valeur venant d'une variable, fonction ou autre qui indique la destination finale de l'outil.

4. speeddata : Vitesse de croisière maximale que l'outil peut d'atteindre durant le déplacement.

5. zonedata : Précision d'arrêt ou d'approche de l'origine du référentiel de l'outil sur un robtarget à atteindre.

6. tooldata : Le référentiel de l'outil et les caractéristiques physiques de l'outil.

7. wobjdata : Paramètre optionnel qui défini par rapport à quel référentiel de travail on veut positionner le référentiel de l'outil à la pose définie dans le robtarget. Si ce paramètre est omis, le référentiel par défaut est celui de l'atelier (appelée wobj0).

III.5.1.5 Configuration mécanique :

Quand un robot déplace sont outil vers la pose d'un robtarget, il doit normalement respecter la configuration physique (configuration moteur) imposée par ce robtarget pour atteindre l'objectif. Il est possible d'avoir un problème à définir la bonne configuration physique avec des robtargets calculés ou même pour un point décalé à partir d'un autre. Il est donc possible de spécifier au robot de ne plus tenir compte de cette imposition pour solutionner la pose d'un robtarget, ce qui signifie que le robot choisi lui-même sa configuration mécanique pour atteindre la pose.

Afin d'éviter les mouvements imprévus, l'approche abordée dans le contrôleur IRC5 est de faire le plus petit déplacement possible. Ce qui ne laisse pas le robot choisir son sens de rotation des articulations, donc si l'une d'entre elles atteint une limite durant le déplacement, le robot tombe en faute.

Les paramètres de base d'une commande de configuration moteur sur les mouvements de type articulaire :

ConfJ \On | \Off;

1. ConfJ : Commande définissant si le robot respecte ou non les configurations moteurs imposées durant les mouvements joints.

2. \On | \Off : On, active l'imposition des configurations moteurs. Off, désactive l'imposition des configurations moteurs.

Les paramètres de base d'une commande de configuration moteur sur les mouvements de type linéaire et circulaire :

ConfL \On | \Off;

1. ConfL : Commande définissant si le robot respecte ou non les configurations moteurs imposées durant les mouvements linéaires et circulaires.

2. \On | \Off : On, active l'imposition des configurations moteurs. Off, désactive l'imposition des configurations moteurs.

III.5.1.6 Vitesse et précision du déplacement :

*Vitesse :

La vitesse est un enregistrement de type speeddata qui s'évalue toujours au niveau du référentiel de l'outil. Il existe dans le contrôleur plusieurs constantes de type speeddata déjà définies (v1,v10, v20, v50, v100, etc.).

L'enregistrement de type speeddata est une structure composée des éléments suivants :

1. v-tcp : vitesse linéaire de l'origine du référentiel de l'outil en mm/s ;

2. v-ori : vitesse angulaire du référentiel de l'outil, exprimée en degrés/s ;

3. v-leax : vitesse des articulations prismatiques externes en mm/s ;

4. v-reax : vitesse des articulations rotoides des externes en mm/s.

III.5.1.7 L'interpolation :

L'interpolation est un enregistrement de type zonedata et définit la tolérance en position à respecter par rapport au robtarget spécifié, dans un déplacement, lorsqu'il y a un autre déplacement tout de suite après. Cette fonctionnalité a pour but d'assurer la fluidité des parcours et l'optimisation des temps de cycles. On peut faire le parallèle avec la conduite d'automobile. En effet, quand un conducteur doit changer de direction dans un milieu piéton, il va décélérer, s'immobiliser complètement, puis tourner. à l'opposé, lorsqu'il prend une sortie d'autoroute, il suivra un arc de cercle lui permettant de ne pas top ralentir.

L'enregistrement de type zonedata est composé de plusieurs valeurs qui définissent le comportement de l'outil lors de l'arrivée dans la zone d'interpolation .

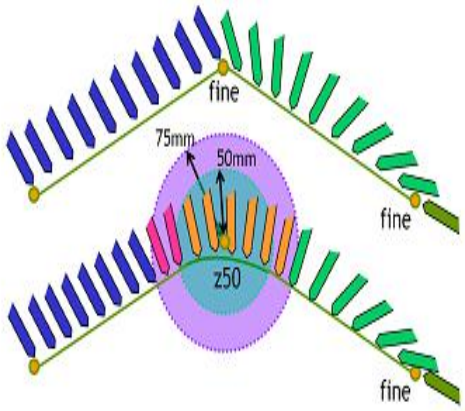

Figure 3.6 : Interpolation entre deux déplacements linéaires

1. finep : Variable booléenne définissant si le robot doit s'arrêter ;

2. pzone-tcp : rayon en mm de la sphère ou l'origine du référentiel de l'outil peut commencer l'interpolation vers le prochain trajet ;

3. pzone-ori : rayon en mm de la sphère ou l'origine du référentiel doit se trouver pour que l'orientation de l'outil puisse commencer l'interpolation vers le prochain trajet. Ce rayon doit être égal ou plus grand que le rayon précédant.

III.5.1.8 Fonctions de calculs de robtargets et de poses :

Souvent, nous avons besoin de calculer un nouveau robtarget à partir d'un robtaget donné. Il existe deux fonctions à cet effet :

***RelTool** :

Cette fonction retourne un robtarget en faisant une transformation (par exemple, une rotation) par rapport au référentiel d'un robtarget donné.

Exemple : r1:=RelTool(robtarget, Dx, Dy, Dz, \Rx, \Ry, \Rz);

1. RelTool : Fonction qui retourne un robtarget calculé à partir du système d'axes d'un robtarget donné, mais exprimé dans le référentiel du robtarget donné ;

2. robtarget : valeur à partir duquel on trouve un nouveau robtarget ;

3. Dx : déplacement en mm dans l'axe x du robtarget donné ;

4. Dy : déplacement en mm dans l'axe y du robtarget donné;

5. Dz : déplacement en mm dans l'axe z du robtarget donné ;

6. \Rx : rotation en degré autour de l'axe x du robtarget donné.

7. \Ry : rotation en degré autour de l'axe y du robtarget donné.

8. \Rz : rotation en degré autour de l'axe z du robtarget donné.

Si deux ou trois rotations sont spécifiées en même temps, elles sont effectuées tout d'abord autour de l'axe x, puis autour du nouvel axe y et ensuite autour du nouvel axe z.

*** Offs**

Cette fonction retourne un robtarget décalé dans le repère du robtarget donné.

Exemple: r1:=Offs(robtarget, Dx, Dy, Dz);

1.Offs: Fonction qui retourne un robtarget décalé dans le référentiel du robtarget donné ;

2. robtarget : valeur à partir duquel on trouve un nouveau robtarget ;

3. Dx : déplacement en mm dans l'axe x du repère ;

4. Dy : déplacement en mm dans l'axe y du repère ;

5. Dz : déplacement en mm dans l'axe z du repère ;

*** DefFrame**

Cette fonction retourne une pose à partir de l'origine des trois robtargets. Il faut comprendre que lors de l'enseignement de robtargets, l'humain arrive à avoir une erreur relativement faible au niveau du positionnement en translation, mais que l'orientation est souvent de piètre qualité. Donc la fonction < DefFrame > définit une pose avec orientation contrôlée à partir de la translation des robtargets.

Exemple: r1:=DefFrame(robtarget (1), robtarget (2), robtarget (3),\Origin:=[1,2 ou 3]);

1. Offs : Fonction qui retourne une pose à partir de trois robtargets ;

2. robtarget (1) : premier robtarget ;

3. robtarget (2) : deuxième robtarget ;

4. robtarget (3) : troisième robtarget ;

5. \Origin : paramètre optionnel définissant la méthode utilisée pour solutionner la pose résultante ;

(a) \Origin :=1 ou paramètre omis: Le premier robtarget défini l'origine de nouveau système d'axe, l'origine du robtarget 2 est nécessairement sur le vecteur x de la pose calculée et le troisième robtarget positionne le plan XY (Y positif) ;

(b) \Origin :=2 : Le premier robtarget est sur l'axe x négatif de la pose calculée, l'origine du robtarget 2 est l'origine de la pose calculée et le troisième robtarget positionne le plan XY (Ypositif) ;

(c) \Origin :=3 : Méthode utilisée lors de l'enseignement avec la méthode de 3 points dans le boitier de commande.

Le premier robtarget est sur l'axe x proche de l'origine de la pose calculée, l'origine du robtarget 2 est sur l'axe x de la pose calculée, mais loin de l'origine et le troisième robtarget positionne le vecteur Y positif qui sera perpendiculaire à l'axe x, fixant par le même fait l'origine de la pose calculée.

III.6 Conclusion

La programmation graphique sous tous ses aspects, ses avantages, ses inconvénients et ses contraintes on été évoqués dans ce chapitre. Les commandes de déplacement ont une importance capitale dans la programmation et la planification d'une tâche, dans ce chapitre une attention particulière a été donnée à ce sujet.

Chapitre 4

Application sous RobotStudio

IV.1Introduction

Dans notre projet on a travaillé avec les robots ABB type IRB 1200 et IRB 1410 .
Premièrement, on va définir ces derniers en donnant leurs caractéristiques importantes .
Deuxièmement , on va présenter la planification, programmation et simulation des taches réalisées, en montrant a quoi servent les différentes fenêtres avec lesquelles on a travaillé .

IV.2 Robot type IRB 1410 :

IV 2.1. Description:

Figure 4.1 : Robot IRB 1410

L'IRB 1410 vous offre des cycles de travail rapides et fiables qui augmentent la productivité. Le robot a fait ses preuves dans les applications de soudage à l'arc et offre des performances et une valeur exceptionnelles, garantissant des temps de retour sur investissement courts. Le robot a une capacité de manipulation de 5 kg au poignet avec une charge supplémentaire unique de 18 kg pour les équipements d'application sur le bras supérieur. Des niveaux supérieurs de contrôle et de précision de trajectoire assurent une excellente qualité de travail.

La possibilité d'ajuster la vitesse et la position du processus vous permet d'obtenir une précision de fabrication optimale avec peu ou pas de rejets.

L'IRB 1410 est connu pour sa construction rigide et robuste. Cela se traduit par de faibles niveaux de bruit, de longs intervalles entre les entretiens de routine et une longue durée de vie.

Le robot dispose d'une grande surface de travail et d'une grande portée. La conception compacte, le poignet très fin et le fonctionnement haute performance, même dans les endroits difficiles et restreints.

L'IRB 1410 dispose d'un câblage d'alimentation en fil et de trous de montage intégrés pour un montage optimisé de l'équipement process sur le bras. Les fonctions de soudage à l'arc faciles à utiliser sont incluses en standard dans la commande du robot IRC5 et sont disponibles via l'unité d'interface de programmation et de commande brevetée - le FlexPendant.

Pour un fonctionnement sans souci, ABB propose également RemoteService, qui donne accès à distance aux équipements de surveillance et de support. De plus, les clients d'ABB peuvent profiter de

l'organisation de service de l'entreprise ; avec plus de 35 ans d'expérience dans le secteur du soudage à l'arc, ABB fournit une assistance technique dans plus de 100 sites dans 53 pays.[12]

IV.2.2 Caractéristiques :

Type	articulé
Nombre d'axes	6 axes
Fonction	de soudage à l'arc, de manutention
Autres caractéristiques	pour temps de cycle rapide, au sol
Charge maximale	5 kg (11,023 lb)
Rayon d'action	1 440 mm (56,69 in)
Répétabilité	0,02 mm (0,00079 in)

IV.3 Robot type IRB 1200 :

IV.3.1 Description:

Figure 4.2 : Robot IRB 1200

Souhaitez-vous réduire de 15 % la taille de vos cellules robotisées tout en augmentant leur vitesse d'exécution de 10 % ? C'est désormais possible avec le nouveau robot IRB 1200 d'ABB. Il répond plus particulièrement aux besoins des industries spécialisées dans la manutention et le service des machines, notamment en termes de flexibilité, de facilité d'utilisation, de compacité et de temps de cycle réduit, tout en garantissant d'importantes enveloppes de travail.

L'IRB 1200 est adapté aux situations nécessitant l'intervention d'un robot compact, sans sacrifier pour autant ni son enveloppe de travail ni ses fonctionnalités. Les courtes distances entre les opérations contribuent à réduire les temps de cycle au sein de la cellule, aussi petite soit-elle, vous permettant ainsi d'en faire plus avec moins.

Son design très fonctionnel de l'IRB 1200 est une caractéristique qui n‘est pas seulement esthétique. Ses surfaces lisses facilitent son nettoyage et son entretien dans différentes applications de service de machines ou dans la manutention de l‘industrie agroalimentaire. Cette caractéristique de conception se retrouve dans toute la gamme des fonctionnalités de l'IRB 1200, et contribue à réduire de 15 % la taille des cellules robotisées et de 10 % leurs temps de cycle. Compacité L’absence de décalage mécanique entre l’axe 1 et l’axe 2 assure à l’IRB 1200 une course horizontale plus longue que celle des autres robots. Autrement dit, l'IRB 1200 peut être positionné à proximité immédiate de la pièce à traiter. L'un des principaux avantages de cette course allongée est qu'elle permet une installation nettement plus compacte lorsque le robot est monté en position suspendue ou au mur à l'intérieur d'une cellule de petite taille, comme dans le cas des applications d'usinage, de polissage ou de manutention de composants électroniques.
Avec ses dimensions compactes, il offre une excellente enveloppe de travail contribuant à réduire les temps de cycle, et à accroître la compacité des cellules robotisées.

Il est équipé de quatre alimentations séparées en air comprimé, de 10 signaux client et d‘un port Ethernet, l'IRB 1200 est conçu dans un objectif d'intégration maximale. Ainsi, les raccordements électriques et d'air comprimé se font sur le robot via les connecteurs latéraux ou sur la platine de connexion intégrée à l'embase (disponible en option). Le port Ethernet facilite l'intégration avec d'autres équipements. Le câblage est réalisé à l'intérieur même du robot, de la bride du poignet jusqu'à l'embase, afi n de garantir une plus grande compacité. [13]

IV.3.2 Caractéristiques :

Type	articulé
Nombre d'axes	6 axes
Fonction	de manutention
Applications	pour l'industrie agroalimentaire, pour l'industrie de la boisson, pour salle blanche, pour l'industrie de la fonderie
Autres caractéristiques	pour temps de cycle rapide, compact, de sécurité
Charge maximale	5 kg, 7 kg (11,023 lb)
Rayon d'action	703 mm, 901 mm (27,68 in)
Répétabilité	0,02 mm, 0,025 mm (0,00079 in)

IV.4 La réalisation des taches :

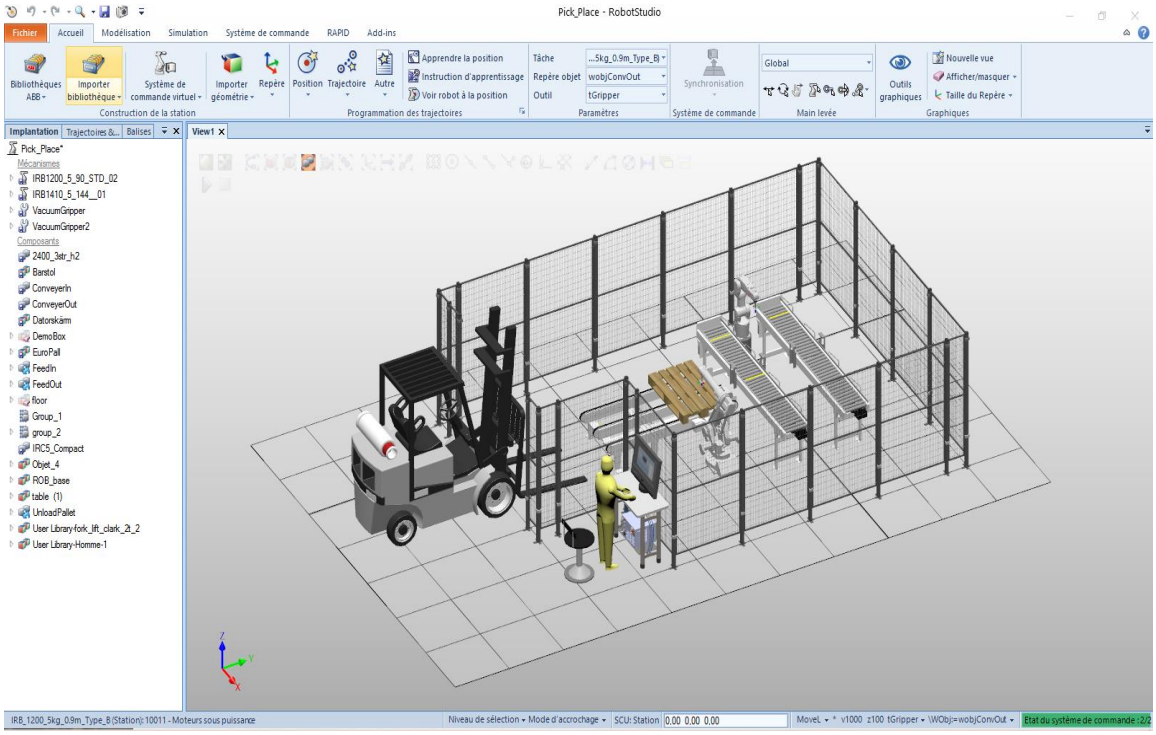

Figure 4.3 : La base du logiciel

La fenêtre principale du logiciel se compose de plusieurs touches et onglets, et c'est a partir de ces derniers qu'on a fait nos instructions principales .

Ces instructions dans chacun des onglets sont les suivants :

IV.4.1 Accueil :

- Importer depuis la bibliothèque les robots , les tapis roulants ainsi que la palette .
- Chercher une géométrie spécifique .
- Créer la trajectoire des différents équipements .
- Pivoter , déplacer et piloter les éléments de la cellule .

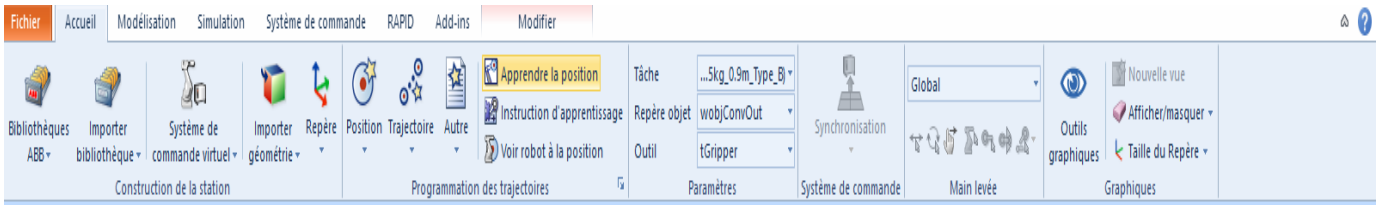

Figure 4.4 : L'onglet accueil

IV.4.2 Modélisation :

- Créer les repère de modélisation, les axes de rotation , les bordure entre les corps .
- Mesurer les distances.
- Créer des mécanismes, outils ou des convoyeur .

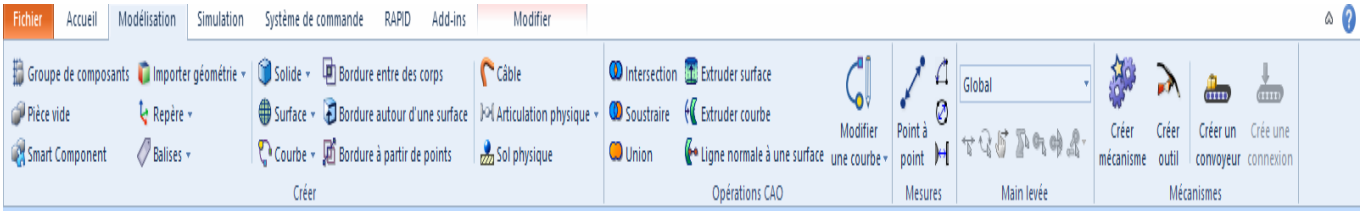

Figure 4.5 : L'onglet modélisation

IV.4.3 Simulation :

- Configurer la simulation et ses paramètres.
- Enregistrer une vidéo pendant la simulation.

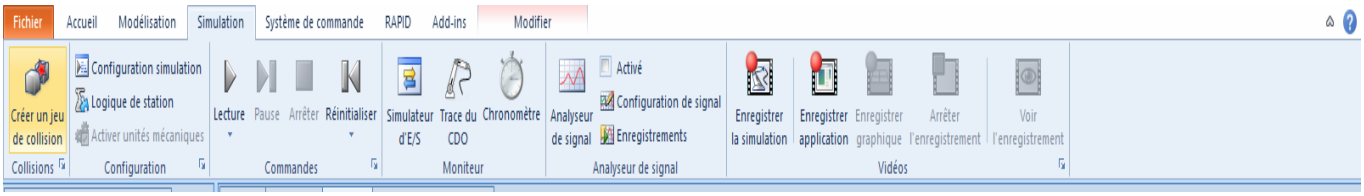

Figure 4.6 : L'onglet simulation

IV.4.4 Rapid :

- Introduire et programmer les différentes instructions

Figure 4.7 : L'onglet RAPID

IV.4.5Add-ins :

- Installer et migrer les différentes applications et commandes nécessaires ou manquantes pour le bon fonctionnement du logiciel.

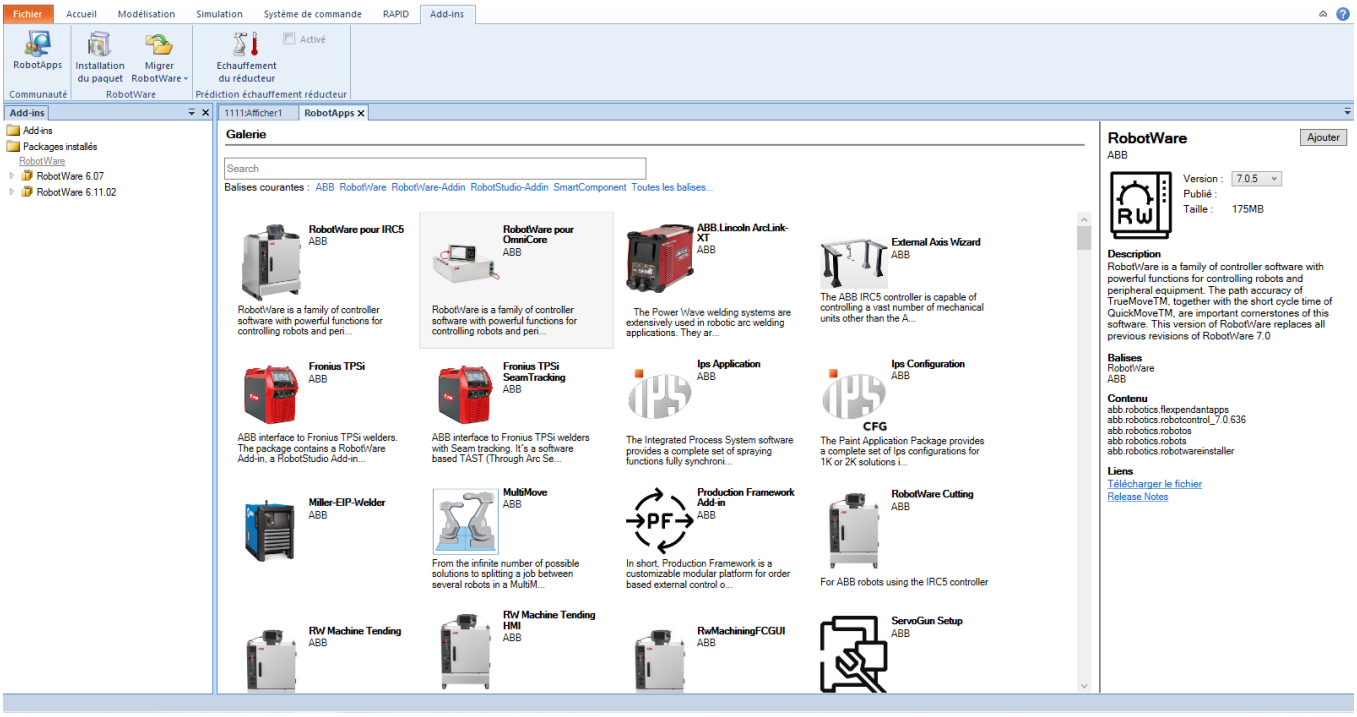

Figure 4.8 : L'onglet Add-ins

IV.4.6 Implantation :

C'est une mini fenêtre qui nous facilite l'accès pour configurer les mécanismes et les composants utilisées.

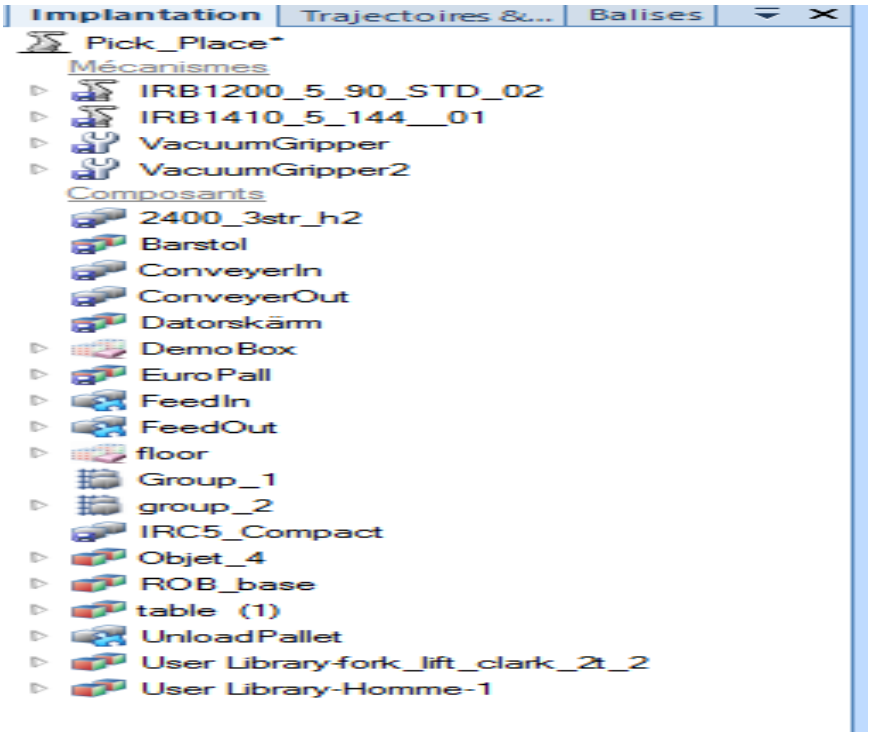

Figure 4.9 : Fenêtre implantation

IV.4.7 Journal :

C'est un afficheur des événements et des messages qui se trouve en bas.

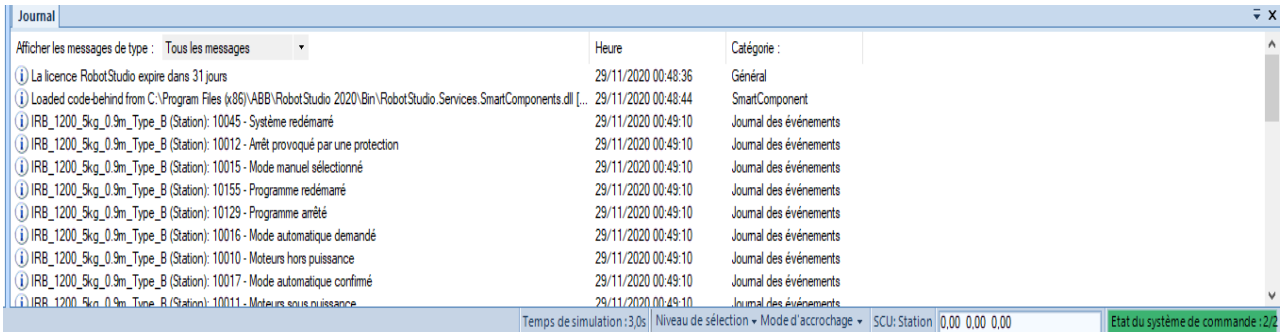

Figure 4.10 : Fenêtre journal

IV.4.8 Trajectoires et positions :

C'est une mini fenêtre pour l'accès direct vers la trajectoire et positions.

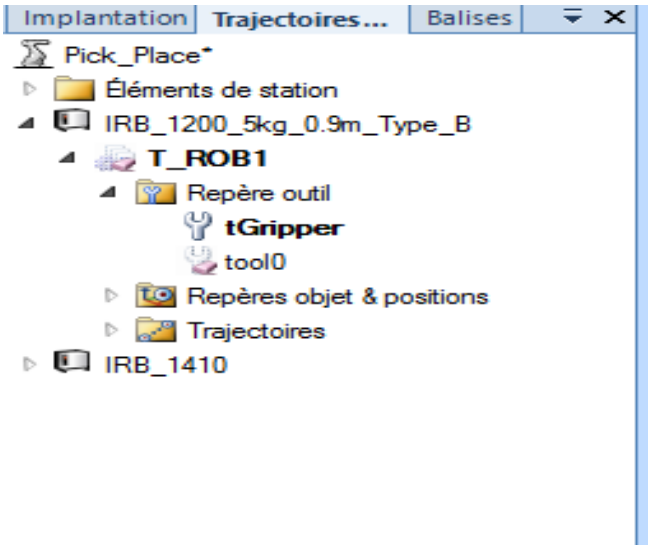

Figure 4.11 : Fenêtre des trajectoires

IV.4.9 Systèmes de commandes :

Une base appartient a l'onglet RAPID la ou on introduit les différents enregistrements et instructions.

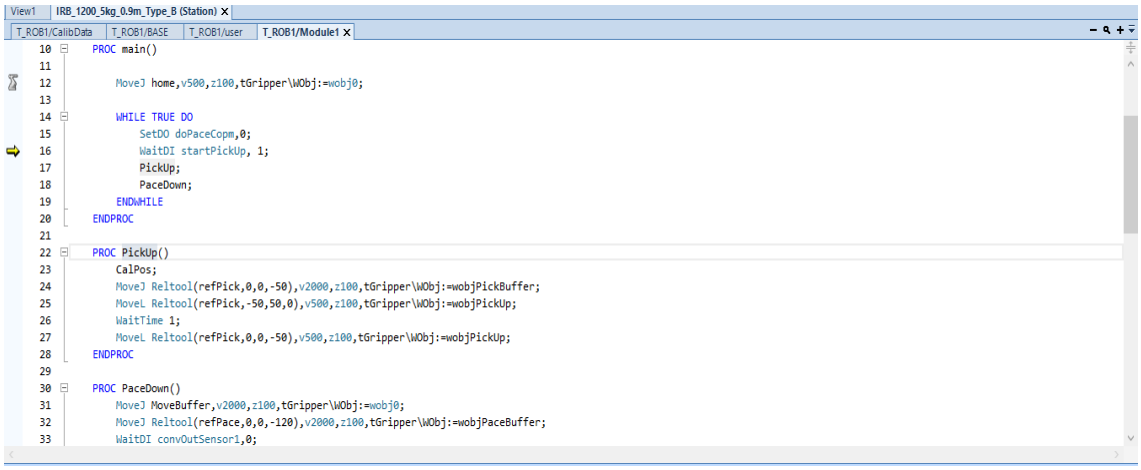

Figure 4.12 : Fenêtre des commandes

IV.4.10 Composants intelligents :

C'est le cerveau de chaque élément qui organise le temps , la vitesse , le chainage ainsi que les instructions a accomplir (attacher une pièce / détacher la pièce …etc.)

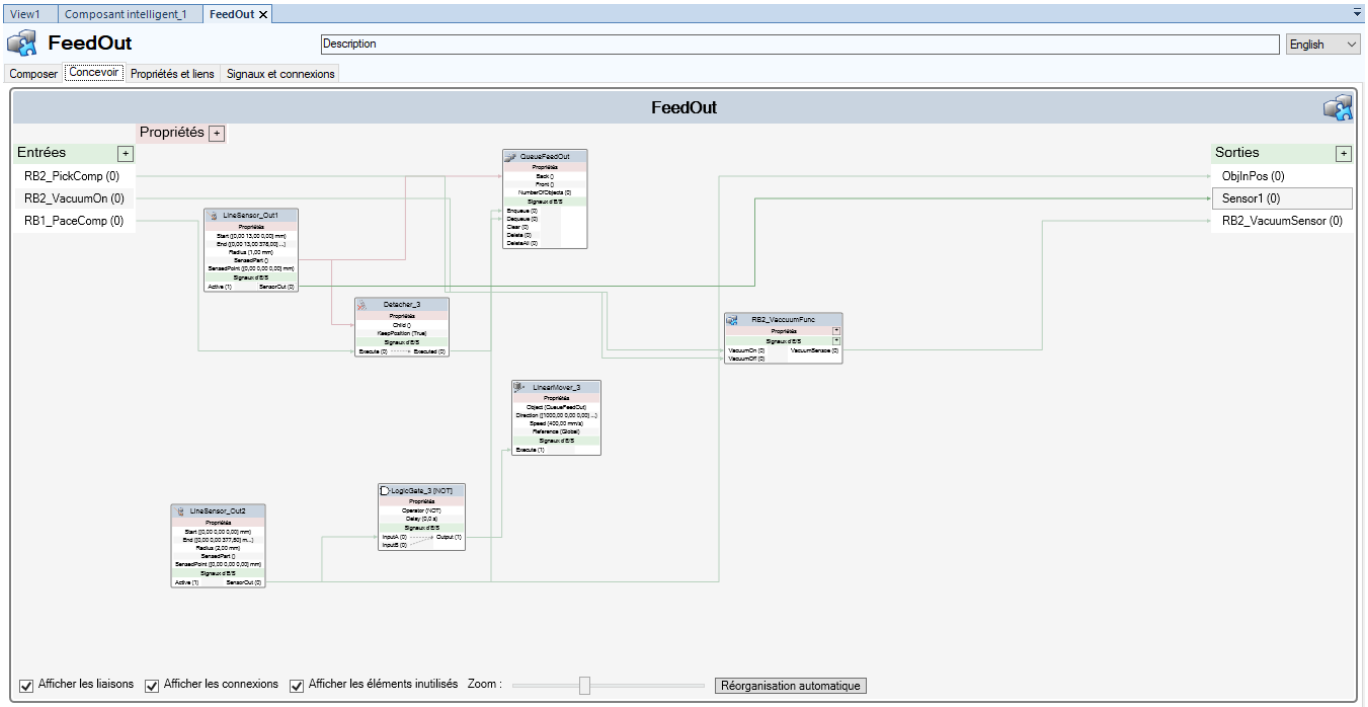

Figure 4.13 : Grafcet qui montre le chainage des instructions

IV.5 Programme des robots :

On a fait la déclaration des variables en respectant l'ordre de classement des registres mémoires ainsi que les enregistrements .

IV.5.1 Robot IRB 1200 :

Pour les modules systèmes , on a le programme celui de la base et l'utilisateur (voir annexe 1)

IV.5.2 Robot IRB 1410 :

Pour les modules systèmes , on a le programme celui de la base et l'utilisateur (voir annexe 2)

IV.6 Conclusion

La programmation hors ligne de la robotique est le meilleur moyen d'augmenter le retour sur investissement des systèmes robots. Sur ce terme on a travaillé avec ce logiciel, offrant une réplique numérique complète des actifs physiques ou des systèmes afin que nous pouvons voir à distance ce qui se passe dans notre ligne de production à partir d'un ordinateur de bureau sans pour autant l'interrompre .

Conclusion générale :

La robotique peut être définie comme l'ensemble des techniques et études tendant à concevoir des systèmes mécaniques, informatiques ou mixtes, capables de se substituer à l'homme dans ses fonctions motrices, sensorielles et intellectuelles.

En effet , La programmation hors ligne de la robotique est le meilleur moyen d'augmenter le retour sur investissement des systèmes robots. Sur ce terme on a travaillé avec le logiciel RobotStudio qui est un logiciel de simulation et de programmation hors ligne d'ABB , offrant une réplique numérique complète des actifs physiques ou des systèmes afin que nous pouvons voir à distance ce qui se passe dans notre ligne de production à partir d'un ordinateur de bureau sans pour autant l'interrompre .

Il fournit aussi les outils qui augmentent la rentabilité de notre système par la réalisation de tâches telles que la formation, la programmation et l'optimisation ce qui procure de nombreux avantages cités dans le 3eme chapitre.

Ainsi nous avons pu simuler une cellule pour coopérer plusieurs robots qu'on a simulé avec ce logiciel.

Ce travail nous a permis de mettre en valeur nos connaissances cumulées pendant toute la durée de nos études, il nous a permis aussi d'acquérir une expérience nouvelle : comment bien maitriser la simulation et la programmation sous ce logiciel.

Perspectives :

A la lumière des résultats obtenus , plusieurs perspectives s'ouvrent à nous pour bien maitriser l'application sous le RobotStudio pour simuler d'autres cellules vastes et plus compliquées qui faitent plusieurs taches a la fois .

D'une façon générale, c'est bien évident que notre travail est loin d'être parfait, mais il peut constituer une base très intéressante et un stand expérimental pour les promotions à venir qui voudront travailler sur ce type de sujet.

Bibliographie :

[1] Negrache Bensaouag « Commande dynamique et Adaptative des robots manipulateurs rigides en utilisant l'algorithme des moindres carrés et du gradient application à un robot à 3ddl, Puma »université d'Oran Es-Sénia, 23 octobre 2004 ,

https://espace.etsmtl.ca/id/eprint/710/1/LE_BOUDEC_Brice.pdf

consulté le 29-07-2020

[2] BOUZIANE Fatima Zohra « Rétro-conception du bras horizontal de robot manipulateur de la cellule flexible (Tlemcen)», UNIVERSITE ABOU BEKR BELKAID-TLEMCEN Option Ingénierie des systèmes mécaniques productiques,(2013) ,

http://dspace.univ-tlemcen.dz/handle/112/3857

consulté le 17-08-2020.

[3] PRIEL MARC «les robots industriels: caractéristiques, performances et choix »: Edition AFNOR (1990).

https://data.bnf.fr/fr/12181087/marc_priel/

consulté le 17-08-2020.

[4] F.Z. BOUZIANE 'Rétro-conception du bras horizontal de robot manipulateur de la cellule flexible', Université de Tlemcen, option Ingénierie des systèmes mécaniques productiques, 2013

http://dspace.univ-tlemcen.dz/handle/112/3857

consulté le 17-08-2020.

[5] CHAAL Merouane, « Modélisation cinématique d'un robot manipulateur à chaine continue ouverte », UNIVERSITE KASDI MERBAH OUARGLA Option Maintenance Industrielle , (2013).

https://dspace.univ-ouargla.dz/jspui/handle/123456789/1570

consulté le 05-09-2020.

[6] Jacques Gangloff « Cours de Robotique » Université de strasbourg 2010.

https://www.gdr-robotique.org/cours_de_robotique/?id=42d45e5762b80afd925329f86b992718

consulté le 06-09-2020.

[7] Site Wikipédia – Da vinci medical robot , consulté le 25-09-2020

[8] Khalil W., Dombre E., Modelisation, identification and control of robots, Hermes Penton Science, London, ISBN 1-90399-613-9, 2002, 480 p ,

elearn.univ-tlemcen.dz

consulté le 01-10-2020

[9] Thèse de doctorat , université de technologies de Belfort-Montbéliard France,2000,

https://www.utbm.fr/formations/doctorats/

consulté le 01-10-2020

[10] Thermal spray 2006 ; building on 100 of succes , ASM internationnel,materiels , ohio USA,2006 , https://www.worldcat.org/title/thermal-spray-2006-building-on-100-years-of-success-proceedings-of-the-2006-international-thermal-spray-conference-may-15-18-2006-seattle-washington-usa/oclc/145554939

consulté le 03-10-2020 .

[11] Thermal spray 2005 ;explore its potential , E.lugsheider ASM international , materiel park Ohoi USA , 2005. ,

https://www.researchgate.net/publication/226052604_From_Powders_to_Thermally_Sprayed_Coatings

consulté le 03-10-2020 .

[12] https://www.directindustry.fr/ (site web) Consulté le 08-11-2020

[13] https://new.abb.com/ (site web) Consulté le 09-11-2020

Annexes

Annexe 1 :

Base :

```
MODULE BASE (SYSMODULE, NOSTEPIN, VIEWONLY)

  ! System module with basic predefined system data
  !************************************************

  ! System data tool0, wobj0 and load0
  ! Do not translate or delete tool0, wobj0, load0
  PERS tooldata tool0 := [TRUE, [[0, 0, 0], [1, 0, 0, 0]],
                         [0.001, [0, 0, 0.001],[1, 0, 0, 0], 0, 0, 0]];

  PERS wobjdata wobj0 := [FALSE, TRUE, "", [[0, 0, 0],[1, 0, 0, 0]],
                         [[0, 0, 0],[1, 0, 0, 0]]];

  PERS loaddata load0 := [0.001, [0, 0, 0.001],[1, 0, 0, 0], 0, 0, 0];

ENDMODULE
```

Utilisateur :

```
MODULE user (SYSMODULE)

  ! Predefined user data
  !*********************

  ! Declaration of numeric registers reg1...reg5
  VAR num reg1 := 0;
  VAR num reg2 := 0;
  VAR num reg3 := 0;
  VAR num reg4 := 0;
  VAR num reg5 := 0;

  ! Declaration of stopwatch clock1
  VAR clock clock1;

  ! Template for declaration of workobject wobj1
  !TASK PERS wobjdata wobj1 := [FALSE, TRUE, "", [[0, 0, 0],[1, 0, 0, 0]],[[0, 0, 0],[1, 0, 0, 0]]];

ENDMODULE
```

Pour le module programme :

```
    PROC main()

        MoveJ home,v500,z100,tGripper\WObj:=wobj0;

        WHILE TRUE DO
            SetDO doPaceCopm,0;
            WaitDI startPickUp, 1;
            PickUp;
            PaceDown;
        ENDWHILE
    ENDPROC

    PROC PickUp()
        CalPos;
        MoveJ Reltool(refPick,0,0,-50),v2000,z100,tGripper\WObj:=wobjPickBuffer;
        MoveL Reltool(refPick,-50,50,0),v500,z100,tGripper\WObj:=wobjPickUp;
        WaitTime 1;
        MoveL Reltool(refPick,0,0,-50),v500,z100,tGripper\WObj:=wobjPickUp;
    ENDPROC

    PROC PaceDown()
        MoveJ MoveBuffer,v2000,z100,tGripper\WObj:=wobj0;
        MoveJ Reltool(refPace,0,0,-120),v2000,z100,tGripper\WObj:=wobjPaceBuffer;
        WaitDI convOutSensor1,0;
        MoveL Reltool(refPace,0,0,0),v500,z100,tGripper\WObj:=wobjPaceBuffer;
        WaitTime 1;
        SetDO doPaceCopm,1;
        MoveL Reltool(refPace,0,0,-120),v500,z100,tGripper\WObj:=wobjPaceBuffer;
        MoveJ home,v2000,z100,tGripper\WObj:=wobj0;
    ENDPROC

    PROC CalPos()
        wobjPickUp:=wobjPickBuffer;
        wobjPickUp.uframe.trans.x := aXpos;
        wobjPickUp.uframe.trans.y := aYpos;
        wobjPickUp.oframe.rot := OrientZYX(-aZori,0,0);
    ENDPROC
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    !!!!!!!!!!!!!!!
    !!!!!!!
ENDMODULE
```

Annexe 2

Base :

```
MODULE BASE (SYSMODULE, NOSTEPIN, VIEWONLY)

  ! System module with basic predefined system data
  !************************************************

  ! System data tool0, wobj0 and load0
  ! Do not translate or delete tool0, wobj0, load0
  PERS tooldata tool0 := [TRUE, [[0, 0, 0], [1, 0, 0, 0]],
                         [0.001, [0, 0, 0.001],[1, 0, 0, 0], 0, 0, 0]];

  PERS wobjdata wobj0 := [FALSE, TRUE, "", [[0, 0, 0],[1, 0, 0, 0]],
                         [[0, 0, 0],[1, 0, 0, 0]]];

  PERS loaddata load0 := [0.001, [0, 0, 0.001],[1, 0, 0, 0], 0, 0, 0];

ENDMODULE
```

Utilisateur :

```

MODULE user (SYSMODULE)

  ! Predefined user data
  !*********************

  ! Declaration of numeric registers reg1...reg5
  VAR num reg1 := 0;
  VAR num reg2 := 0;
  VAR num reg3 := 0;
  VAR num reg4 := 0;
  VAR num reg5 := 0;

  ! Declaration of stopwatch clock1
  VAR clock clock1;

  ! Template for declaration of workobject wobj1
  !TASK PERS wobjdata wobj1 := [FALSE, TRUE, "", [[0, 0, 0],[1, 0, 0, 0]],[[0, 0, 0],[1, 0, 0, 0]]];

ENDMODULE
```

Pour le module programme :

```
MODULE Module1
    CONST robtarget home_RB2:=[[600,0,995],[0,0,1,0],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget pPickBufferRB2:=[[571.999951894,-947.999987031,1025.000049229],[-0.000000013,0,1,-0.000000022],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget pPickRB2:=[[573.156,-948.021,875],[0,0,1,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget pRefLoadPallet:=[[300,160,100],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS robtarget pPaceLoadPallet:=[[600,640,200],[0,0,1,0],[0,0,0,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
    VAR num xpos_load_pallet := 0;
    VAR num ypos_load_pallet := 0;
    CONST num x_pitch := 150;
    CONST num y_pitch := 120;
    CONST num z_pitch := 100;
    VAR num load_counter := 0;
    CONST num x_offset := 300;
    CONST num y_offset := 160;
    !***********************************************************
    !
    ! Module:  Module1
    !
    ! Description:
    !   <Insert description here>
    !
    ! Author: LenovoDEV
    !
    ! Version: 1.0
    !
    !***********************************************************


    !***********************************************************
    !
    ! Procedure main
    !
    !   This is the entry point of your program
    !
    !***********************************************************
    PROC main()

        Loader;

    ENDPROC

    PROC PickUp()
        MoveJ home_RB2,v2000,z100,tGripper\WObj:=wobj0;
        MoveJ Reltool(pPickBufferRB2,0,0,0),v2000,z100,tGripper\WObj:=wobj0;
        MoveL pPickRB2,v500,z100,tGripper\WObj:=wobj0;
        WaitTime 1;
        SetDO RB2_VacuumCmdOn, 1;
        WaitDI RB2_VacummSensorOn, 1;
        MoveL pPickBufferRB2,v500,z100,tGripper\WObj:=wobj0;
    ENDPROC
    PROC PaceDown()
        MoveJ RelTool(pPaceLoadPallet,0,0,-150),v5000,z10,tGripper\WObj:=wobjLoadPallet;
        MoveL RelTool(pPaceLoadPallet,0,0,0),v1000,z100,tGripper\WObj:=wobjLoadPallet;
        WaitTime 1;
        SetDO RB2_PaceComp, 1;
        SetDO RB2_VacuumCmdOn, 0;
        MoveL RelTool(pPaceLoadPallet,0,0,-150),v1000,z100,tGripper\WObj:=wobjLoadPallet;
        MoveJ home_RB2,v5000,z100,tGripper\WObj:=wobj0;
        SetDO RB2_PaceComp,0;
    ENDPROC
```

```
61
62      PROC Loader()
63
64          pPaceLoadPallet := pRefLoadPallet;
65
66          FOR n FROM 1 TO 2 DO
67              pPaceLoadPallet.trans.z := z_pitch*(n-1) + 100;
68              FOR col FROM 1 TO 3 DO
69                  pPaceLoadPallet.trans.x := (x_pitch*(col-1)) + x_offset;
70                  FOR row FROM 1 TO 5 DO
71                      pPaceLoadPallet.trans.y := y_pitch*(row-1) + y_offset;
72                      WaitDI RB2_start, 1;
73                      PickUp;
74                      PaceDown;
75                  ENDFOR
76              ENDFOR
77          ENDFOR
78
79          SetDO LoadComp, 1;
80          WaitDI diUnloadFinish, 1;
81          SetDO LoadComp, 0;
82      ENDPROC
83
84      PROC LoadCounter()
85
86      ENDPROC
87  ENDMODULE
```