REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

CDTA
Centre de Développement des Technologies Avancées

Département Automatique

Laboratoire de Productique et Roborique

Mémoire de Projet de Fin d'Etudes
Pour l'obtention du diplôme d'Ingénieur d'Etat en Automatique

Mise en place d'un système de commande Automatisé d'une plateforme robotisée d'assemblage

Réalisé par : - Romaissa BELLAL
- Imene AISSIOU

Sous la direction de : -Mr. R.ILLOUL — Chef de département (ENP)
-Mr M.GAHAM — Chef de projet (CDTA)

Présenté et soutenue publiquement le : 03/07/2019

Composition du jury :

Président	Mr	E.M.BERKOUK	Professeur	ENP
Promoteur	Mr	R.ILLOUL	Chef de département	ENP
Promoteur	Mr	M.GAHAM	Chef de projet	CDTA
Examinateur	Mr	H.ACHOUR	MAA	ENP

ENP 2019

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

CDTA
Centre de Développement des Technologies Avancées

Département Automatique

Laboratoire de Productique et Roborique

Mémoire de Projet de Fin d'Etudes
Pour l'obtention du diplôme d'Ingénieur d'Etat en Automatique

Mise en place d'un système de commande Automatisé d'une plateforme robotisée d'assemblage

Réalisé par : - Romaissa BELLAL
- Imene AISSIOU

Sous la direction de : -Mr. R.ILLOUL — Chef de département (ENP)
-Mr M.GAHAM — Chef de projet (CDTA)

Présenté et soutenue publiquement le : 03/07/2019

Composition du jury :

Président	Mr	E.M.BERKOUK	Professeur	ENP
Promoteur	Mr	R.ILLOUL	Chef de département	ENP
Promoteur	Mr	M.GAHAM	Chef de projet	CDTA
Examinateur	Mr	H.ACHOUR	MAA	ENP

ENP 2019

Dédicaces

À nos chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de nos études ;

À nos chers frères et sœurs pour leurs appuis, leurs encouragements permanents, et leur soutien moral ;
À toutes nos familles pour leur soutien tout au long de notre parcours universitaire ;

À tous nos amis et nos camarades cette promotion en particulier Meziane Ismahane et Betit Lilya.

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible

Remerciement

Nous tenons tout d'abord à remercier le bon dieu tout puissant qui nous a donné la volonté et le courage afin de pouvoir présenté ce modeste travail.

Nous voudrions dans un premier temps remercier, notre promoteur M. GAHAM, Chef de projet R&D au centre de développement des technologies avancées, de nous avoir proposé ce sujet mais aussi pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter nos réflexions.
Nous voudrions aussi remercier notre Professeur Mr RACHID ILLOUL, pour avoir accepté de nous encadrer et pour son énorme soutien tout au long de notre projet d'étude.

Nos sincères gratitudes à M. Ricardo JIMENEZ SANCHEZ et à tous les membres du centre culturel universitaire pour leurs conseils et leurs interventions lors de la rédaction de ce mémoire.

Nous tenons à remercier, aussi les membres des jurys pour nous avoir fait l'honneur de juger notre travail. Ainsi que pour avoir consacré une partie de leurs temps précieux pour lire et corriger ce mémoire.

Dans l'impossibilité de citer tous les noms, nos sincères remerciements vont à tous ceux et celles, qui de près ou de loin, ont permis par leurs conseils et leurs compétences la réalisation de ce mémoire.

ملخص:

أصبحت اليوم إدارة الإنتاج ضرورية للشركات التي تسعى لإنتاج السلع و الخدمات بأسعار تنافسية مع احترام الموعد النهائي. هدفها هو أن تكون قادرة على تحقيق التوازن بين الإنتاج وطلبات العملاء ، وتحديد اللوازم ، وتحديد الموارد اللازمة وتحديد الموعد النهائي لاستكمال الطلبات. كجزء من مشروع مقترح من مركز تنمية التكنولوجيات المتطورة وشركة سيمنز الجزائر ، قمنا بتطوير وتنفيذ نهج قائم على المنتجات استنادًا إلى مفهوم العوامل المتعددة الذي يتحكم في نظام آلي.

الدالة الكلمات: المنتج، نظام متعدد الوكلاء، تحكم منطق قابل للبرمجة، تحديد الهوية بموجات الراديو، الأجهزة في الحلقة، محاكاة، تحكم في الوقت الفعلي

ABSTRACT :

Today ,Production management has become a necessity for companies that want to produce goods and services at competitive prices while respecting the deadline. Its goal is to be able to balance production against customer orders, determine supplies, define the necessary resources and set the deadline for completion of orders. As part of a project proposed by the Advanced Technologies Research Center and Siemens Algeria, we have developed and implemented a product-driven control based on the multi-agent concept that controls a robotic system.

Keywords: Smart Product, PLC, multi-agent system, RFID, Hardware-in-the-loop, Simulation, real-time control.

Résumé :

La gestion de la production est devenue aujourd'hui une nécessité pour les entreprises qui doivent produire des biens et des services à des prix compétitifs tout en respectant le délai. Son objectif est de pouvoir équilibrer la production par rapport aux commandes client, de déterminer les approvisionnements, de définir les ressources nécessaires et de fixer les délais de réalisation des commandes. Dans le cadre d'un projet proposé par le Centre de recherche des techniques avancés et Siemens Algérie, nous avons développé et implémenté une commande de pilotage par produit basée sur le concept multi-agent qui commande un système robotisé.

Mots clés : Produit intelligent, Automate pc, système multi-agents, RFID, Hardware-in-the-loop, Simulation, pilotage en temps réel.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L'industrie a connu la succession de trois phases d'évolution majeure qui sont qualifiées de révolution : la mécanisation, l'industrialisation puis l'automatisation, et aujourd'hui une quatrième évolution qualifiée comme la nouvelle génération d'usines connectées, robotisées et intelligentes fait son apparition, c'est l'industrie 4.0.

Dans ce cadre, on propose ce travail concernant la mise en place d'une plateforme d'assemblage robotisée. Nous avons modélisé la plateforme appelée aussi cellule, et nous avons développé une réalisation d'un système à base d'une architecture de commande décentralisée en utilisant l'Open Controller ET200sp.

Ce travail nous l'avons réalisé en collaboration avec l'équipe chargée des systèmes robotiques de production de la division « robotique et productique » au Centre de Recherche de Développement des Technologies Avancées (CDTA) avec des équipements fournis par l'entreprise Siemens.

Le projet a pour objectif de faire évoluer les usines en se basant sur les technologies d'aujourd'hui et les méthodes proposées par l'industrie 4.0. Dans le but de créer une chaîne de production constituée de plusieurs éléments interconnectés et une architecture décentralisée qui répond aux nouvelles exigences du marché en temps réel, l'industrie 4.0 propose des produits uniques et personnalisées en un temps plus réduit, tout en conservant des coûts équivalents.

Dans la pratique, la réalisation de ce concept ne sera réellement possible qu'avec une forte structuration des données sous forme d'objets industriels modélisés, depuis l'échelon le plus bas (données fiabilisées et horodatées des capteurs) jusqu'aux éléments de niveaux les plus élevés (équipements, machines, lignes et sous-ensembles de production complets). Le défi donc, que nous devons relevé est : comment réaliser cette structure ?

Chapitre 1

Chapitre 1

systèmes de production

De nos jours, le développement des systèmes industriels de la production se concentre précisément sur l'automatisation qui est l'une des technologies moderne la plus recherchée dans la rénovation des équipements industriels. Afin de pouvoir rester compétitives sur des marchés de plus en plus incertains, les entreprises ont besoin d'être réactives. Elles doivent souvent faire face à des événements imprévus tels qu'une annulation ou une modification de commande, la prise en compte d'une commande urgente, des aléas du système de production, etc. Ceci nécessite d'avoir des outils de pilotage de la production capables de réagir face aux événements critiques, dans le but de simplifier et d'améliorer les conditions de travail, éliminer les tâches-répétitives, et d'assurer la sécurité et notamment d'augmenter la production. [1]

L'objectif de ce chapitre est de présenter le contexte général dans lequel s'inscrit notre travail : celui du pilotage des systèmes de production. Nous commencerons par parler du pilotage d'un système en général, nous indiquant les différentes fonctions génériques du pilotage de la production et les différentes architectures possibles de pilotage.

1.1 Système de production automatisé :

Un système de production est l'ensemble des pratiques, des règles, des outils et méthodes qui permettent à l'entreprise d'utiliser des ressources, telle la matière première, la main-d'œuvre et les machines pour transformer la matière première en produits finis satisfaisant les clients en leur apportant de la valeur ajoutée. L'automatisation de ce dernier a pour objet d'associer moyens de production et moyens de commande automatique qui permettent d'assurer la reproductibilité du résultat de la manière la plus autonome possible (plus au moins indépendant des interventions humaines).

L'automatisation d'un système de production permet à l'entreprise d'améliorer sa compétitivité (coûts des produits, qualité, adaptabilité à la demande, ...). Elle s'exprime dans les objectifs suivants [2] :

- Augmenter la productivité : fabriquer le maximum de produits pendant le minimum de temps
- Améliorer la flexibilité de production : cela consiste à fabriquer le maximum de variétés de produits, avec le même équipement. Ce qui nécessite l'utilisation de système de production ayant la capacité du s'adapter rapidement aux changements de caractéristiques des produits à fabriquer, en reconfigurant la circulation des produits et des opérations et la

capacité de répondre dans les plus brefs délais aux variations du volume des commandes, sans créer des stocks inutiles

- Contrôler le flux de production, disposer de données technico-économiques sur la production, simuler des programmes de production.
- Améliorer les conditions de travail du personnel en améliorant la sécurité et supprimant la pénibilité.
- Permettre de réaliser des travaux dans des milieux hostiles et suppléer l'homme dans des situations de conduite dangereuses (fond de mer, centrales nucléaires, usines chimiques, domaine spatial,...).

1.2 Pilotage d'un système de production automatisé

Le pilotage d'un système de production consiste à déterminer le processus supérieur de l'entreprise, décliner ce processus en sous processus, assurer que chaque activité participe en cohérence avec les autres à l'atteinte de l'objectif global. Pour cela, il doit assigner à chaque partie du système un ou plusieurs objectifs à attendre. Chaque système doit posséder des dispositifs de retour d'information pour assurer la concordance entre les objectifs assignés et les résultats obtenus, donc il doit avoir une boucle de rétroaction entre le système physique et le système décisionnel qui lui permettra de rectifier les ordres si les résultats obtenus sur le système physique ne sont pas conformes à l'attendu (fig.1). Les décisions sont prises en se basant sur les informations qu'on en a reçues du terrain, donc un système d'information est nécessaire pour faire le lien entre les sous-systèmes physique et de décision (fig.2).

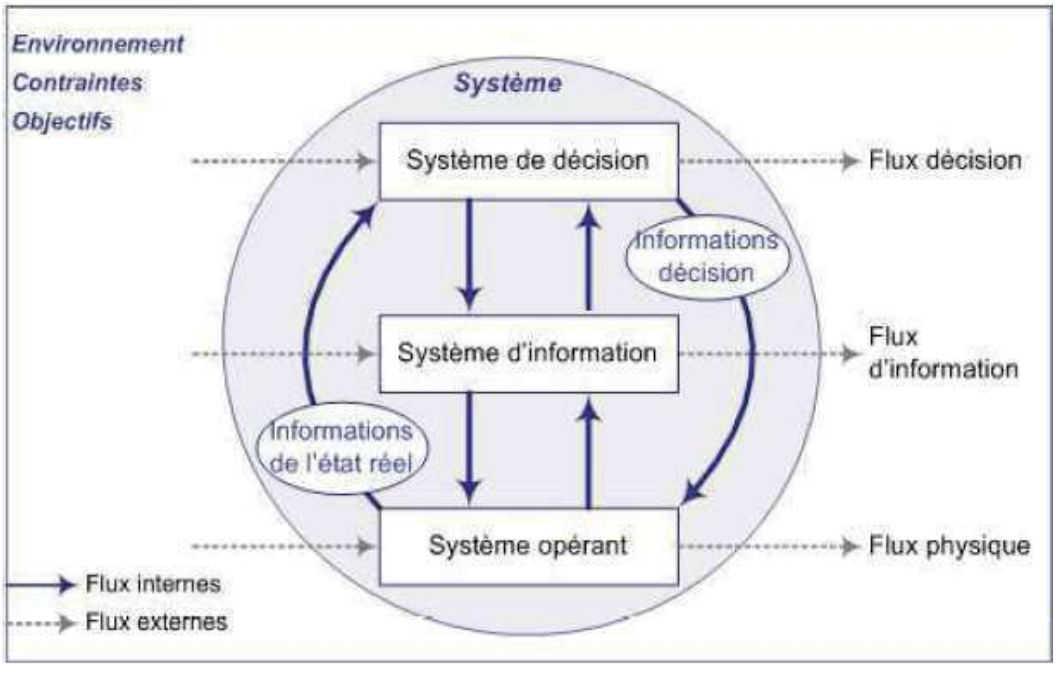

FIGURE 1.1 – Décomposition d'un système de production automatisé.

- **Le système de décision** est la partie commande (PC) d'unsystème de production automatisé. C'est le centre de décision, elle donne les ordres et reçoit les informations de l'extérieur ou de la partie opérative. Cette partie peut être mécanique, électronique ou autre.

Sur de gros systèmes, elle peut se composer de trois parties : un ordinateur, un logiciel et une interface.

- **Le système opérant** est la partie opérative d'un système de production automatisé. C'est le sous-ensemble qui reçoit les ordres de la partie commande et qui les exécute. Il mesure des grandeurs physiques, et rend compte à la partie commande. Il comporte les :
  - **Les capteurs :** sont l'interface entre un processus physique et une information manipulable, ils transforment l'état d'une grandeur physique observée en une grandeur utilisable. Les capteurs recueillent des informations et les transmettent à la partie commande. On les choisit en fonction des informations qui doivent être recueillies.
  - **Les actionneurs** : sont des éléments de la partie opérative qui fournissent la force nécessaire à l'exécution d'un travail ordonné par une unité de commande distante. Ils sont capables de produire des actions physiques à partir de l'énergie qu'ils en ont reçue.

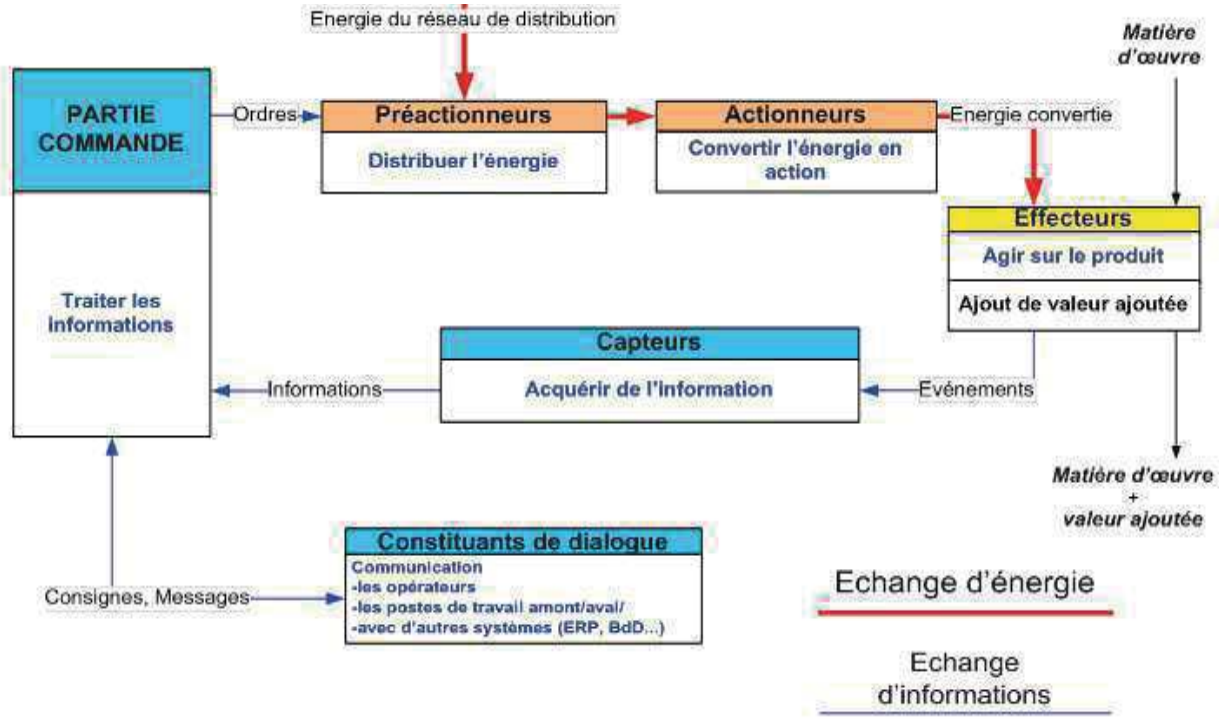

FIGURE 1.2 – Interaction entre les composants d'un système de production automatisé

1.3 Fonctions génériques du pilotage de la production

Pour aboutir à un produit fini, de multiples décisions sont prises quotidiennement par les différents acteurs de l'organisation : dirigeants, cadres, employés, ouvriers, etc. (fig.3). Chaque décision est prise en tenant compte de plusieurs facteurs : caractéristiques de l'organisation, technologie utilisée, évolution du marché, contraintes légales, dynamique des relations sociales,

les risques possible, etc. Ce qui définit l'importance de prendre en compte une quantité croissante d'informations, de connaissances de différentes natures et qualités, de données brutes utilisées dans les diverses activités de l'entreprise telle que les données clients, fournisseurs, statistiques et observation de la production, etc.

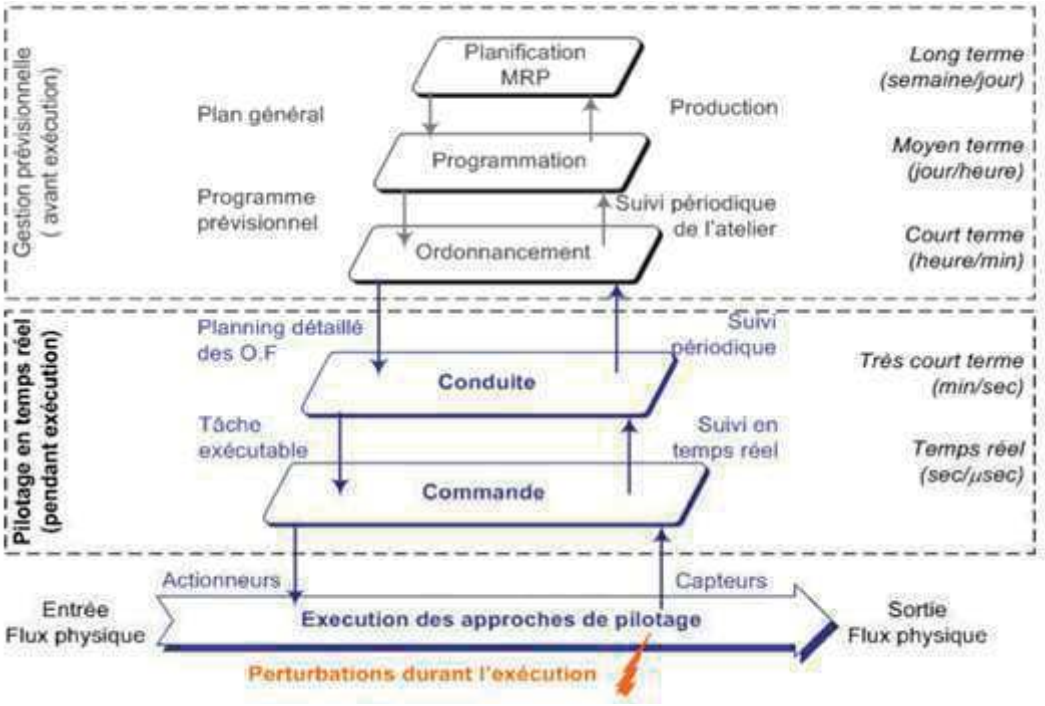

FIGURE 1.3 – Fonctions du pilotage dans la production [MIRD, 09].

1.3.1 Classifications des décisions

On peut classer les décisions selon leur horizon temporel : court terme ou long terme, leur caractère répétitif ou non, le niveau hiérarchique : stratégique, tactique et opérationnel.

— **Les décisions stratégiques ou de planification** sont des décisions uniques et non réversibles prise par le dirigeant et/ou l'équipe de direction d'une entreprise. Ce sont les plus importantes, dans la mesure où elles déterminent l'orientation générale de l'entreprise sur le long terme. Elles sont donc nécessairement porteuses de risques. On peut ranger dans cette catégorie, par exemple, les décisions concernant l'implantation de nouvelles unités de production, la conception d'une nouvelle ligne de production, fusion avec une autre entreprise, etc. Ils reposent sur l'analyse des ressources et compétences pour dégager ses forces et faiblesses et sur l'analyse de l'environnement de l'entreprise pour mettre en lumière les opportunités à saisir et les menaces à éviter.

— **Décision tactique :**sont des décisions prise par le personnel d'encadrement de l'entreprise, applicable sur le moyen terme (de 2 à 5 ans). Ces décisions ont pour but d'organiser les moyens, de développer des différentes tactiques pour accomplir et réaliser les buts et les objectifs fixés par la partie supérieur de la hiérarchie, résoudre les problèmes qui surgissent durant la réalisation des décisions stratégiques.

— **Les Décisions opérationnelles :**sont des décisions à court terme prise par l'individu pour mettre en application le plan d'action prévu par le niveau tactique dans le cadre

du fonctionnement quotidien de l'organisation. Elles interviennent sur l'ordonnancement de la production soit d'une manière périodique où le déclenchement de la procédure de prise de décision est régulier et il est fait à des intervalles de temps réguliers ou d'une manière événementielle où le déclenchement repose sur l'apparition d'événements dans le système.

1.3.2 Modes de la production

En industrie, l'entreprise a le choix de produire ses biens selon différents modes de production :

- **La production à l'unité** est caractérisée par la production d'un produit unique ou en très faible quantité. Ce dernier est fabriqué à la demande du client donc pas de stock. Ce mode est utilisé pour la fabrication des produit soit très volumineux et couteux ou des produits très personnalisés.
- **La production par lots** consiste à produire des petites séries diversifiées de produits identiques. Elle tend le flux de la production en gardant une certaine flexibilité. La planification de ce mode peut être à cours terme (définition et contrôle des priorités) où à moyen terme (planification et régulation des capacités).
- **La production en série** consiste à produire une série de produits identiques standardisés. Selon les débouchés des produits, la fabrication se fera en petite ou grande série uniforme. Ce mode de production nécessite l'implémentation des machines par fonction et une grande flexibilité car les machines ne sont pas spécifiques. Il entraîne la constitution de stocks importants de produits intermédiaires et un délai de production relativement long.
- **La production en continue** concerne la fabrication de produits dont le processus de production ne peut être arrêté, elle est réalisée 24 heures sur 24, 7 jours sur 7, par équipes successives. Ce type de production est caractérisé par un produit unique ou quasi, implantation des machines de façon linéaire, peu de flexibilité, équilibrage de la capacité des machines très bon, investissement important et forte automatisation, des outils de production coûteux et un stockage très réduit.

1.3.3 Les Fonctions du pilotage dans la production

1.3.3.1 Planification

La planification des opérations consiste à définir, en fonction des délais et des priorités, les dates de début des opérations d'un ordre de Travail, fabrication, maintenance, d'achat à fin qu'elle soit terminée dans les délais prévus [2]. Il existe trois niveaux de planification de la production :

- **Plan industriel et commercial** conçu au niveau stratégique de l'entreprise. Il permet de prendre des décisions à long termes sur la gestion de l'ensemble des ressourcesen utilisant des grandes masses d'information. Ce plan sert à :
  - — Effectuer les prévisions de vente par famille de produit.
  - — Intégrer les nouvelles opportunités commerciales décelées grâce aux études de marché (conquête d'un nouveau marché, lancement des nouveaux produits...).

— Etudier l'évolution des ressources d'approvisionnement, de transport, de production, de stockage... et évaluer les besoins matériels, humains et financiers.

— Planifier les investissements futurs.

- **Programme directeur de production** est le plan tactique de l'entreprise qui reprend les données commerciales du Plan industriel et commercial sur un horizon plus court (trois mois par exemple) et les convertit en données de production. Il sert pour chaque référence finale à :
  — Déterminer les besoins bruts et les dates réelles de ces différents besoins sur l'horizon de planification.
  — Calculer les besoin nets.
  — Equilibrer les stocks sur la base de données de planification (stock mini, maxi, délais, stock de sécurité, stratégie de calcul des lots de commande ...).
  — Equilibrer les charges dans l'entreprise sur la base des gammes opératoires.
- **Plan de Charge (plan de fabrication et planning d'atelier)** contient des informations détaillées sur la nature et les quantités de composants à fabriquer, les dates de lancement et de livraison. Le planning est fait sur un horizon (quelques semaines). Sa mise à jour est quotidienne.

1.3.3.2 Programmation :

Elle est chargée d'établir un programme prévisionnel de production àl'atelier, à capacité infinie ou suivant une charge globale admissible par l'atelier. Ce programme prend en compte les besoins dépendants et indépendants et calcule des besoins nets en fonction des stocks, des en-cours, des tailles des lots de fabrication, des taux de rebut... La programmation consiste essentiellement à décliner les objectifs de la planification en ordre de fabrication (OF) sur les différents ateliers et postes de travail et reste dans une logique de définition du «quoi produire».

1.3.3.3 Ordonnancement

Dans le système de production, le problème d'ordonnancement consiste à organiser dans le temps l'exécution d'opérations interdépendantes à l'aide de ressources disponibles en quantités limitées pour réaliser un plan de production [3]. Il se compose de tâches à réaliser et de machines disponibles pour l'exécution de ces tâches. La résolution de ce dernier consiste à : affecter des tâches aux machines, déterminer l'ordre dans lequel chaque machine exécute les tâches, placer des tâches dans le temps endéterminantles instants de début d'exécution de chaque tâche sur chaque machine, optimiser certains critères et tenir compte d'un ensemble de contraintes .[4]

La tâche est une entité élémentaire de travail localisée dans le temps par une date de début et/ou de fin, dont la réalisation nécessite un certain nombre d'unités de temps. On distingue trois types de tâches : des tâches qui peuvent être interrompues au cours de leur exécution, des taches qui doivent être exécutées sans interruption, des tâches indépendantesentre elles c'est-à-dire elles ne sont pas liées entre elles par des contraintes de cohérence technologique.

Les ressources sont de smoyens techniques ou humains utilisés pour la réalisation d'une tâche, elle est disponible en quantité limitée. Il existe plusieurs types des ressources :

— des ressources renouvelables qui sont à nouveau disponibles en même quantité après qu'ils terminent l'exécution des tâche allouer à eux par exemple : les hommes, les machines, l'équipement en général, etc.

— les ressources consommables qui ne sont pas disponibles à nouveau une fois qu'ils terminent l'exécution des tâche allouer à eux par exemple : les matières premières, le budget, etc. On distingue par ailleurs principalement dans le cas de ressources renouvelables :

— les ressources disjonctives qui ne peuvent exécuter qu'une tâche à la fois, par exemple : machine-outil, robot manipulateur, etc.

— les ressources cumulatives qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité, par exemple : équipe d'ouvriers.

L'ordonnancement doit tenir en compte plusieurs critères, par exemple : Le coût et la durée de réalisation, le respect du délai d'exécution, la quantité de moyens nécessaires, la quantité de travail en attente, le temps d'immobilisation des moyens. Pour cela il existe deux niveaux d'ordonnancement l'ordonnancement prévisionnel et l'ordonnancent réactif (fig.4)

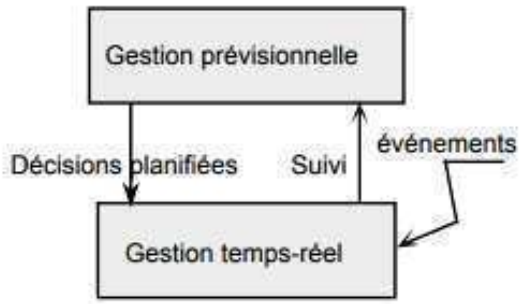

FIGURE 1.4 – interaction entre les deux niveaux d'ordonnancement

L'ordonnancement prévisionnel a pour but de générer un ordonnancement optimal minimisant un critère donné ou une combinaison de plusieurs critères, d'assurer l'anticipation et la programmation d'un ensemble d'actions ou de décisions. La complexité (la diversité des types des ressources composant le système, la multiplicité et parfois la contradiction entre les objectifs de production en terme de coût, délai ou qualité, etc.) et l'inertie (temps de production, de changement d'outil, de calendriers des postes de travail, etc.) du système de production conduisent à mettre en œuvre une gestion prévisionnelle.

L'ordonnancement réactif ou le pilotage en temps réelinclue les méthodes qui servent à réagir en temps réel face aux aléas internes (pannes de ressources, absence de personnel, etc.) ou externes provenant de son environnement (retard d'approvisionnement, arrivée imprévue d'un ordre de fabrication, etc.), il élabore les décisions et les actions qui sont réalisées en temps-réel et qui sont donc déclenchées par un ensemble d'événements liés à l'état courant du système de production.

1.3.3.4 Conduite

La conduite intervienne lors de l'intégration de l'ensemble des décisions et du savoir-faire humain dans la partie décisionnelle pour le pilotage du système.

Il s'avère que l'automatisation des systèmes reste partielle de sorte que certaines tâches doivent être confiées à des intervenants humains dans différentes cas. On note que si l'état du système de production ne permet pas d'appliquer correctement ces planifications, une décision peut être prise localement ou si aucune décision locale ne peut remédier au problème, il sera nécessaire de remettre en question les décisions du niveau supérieur.

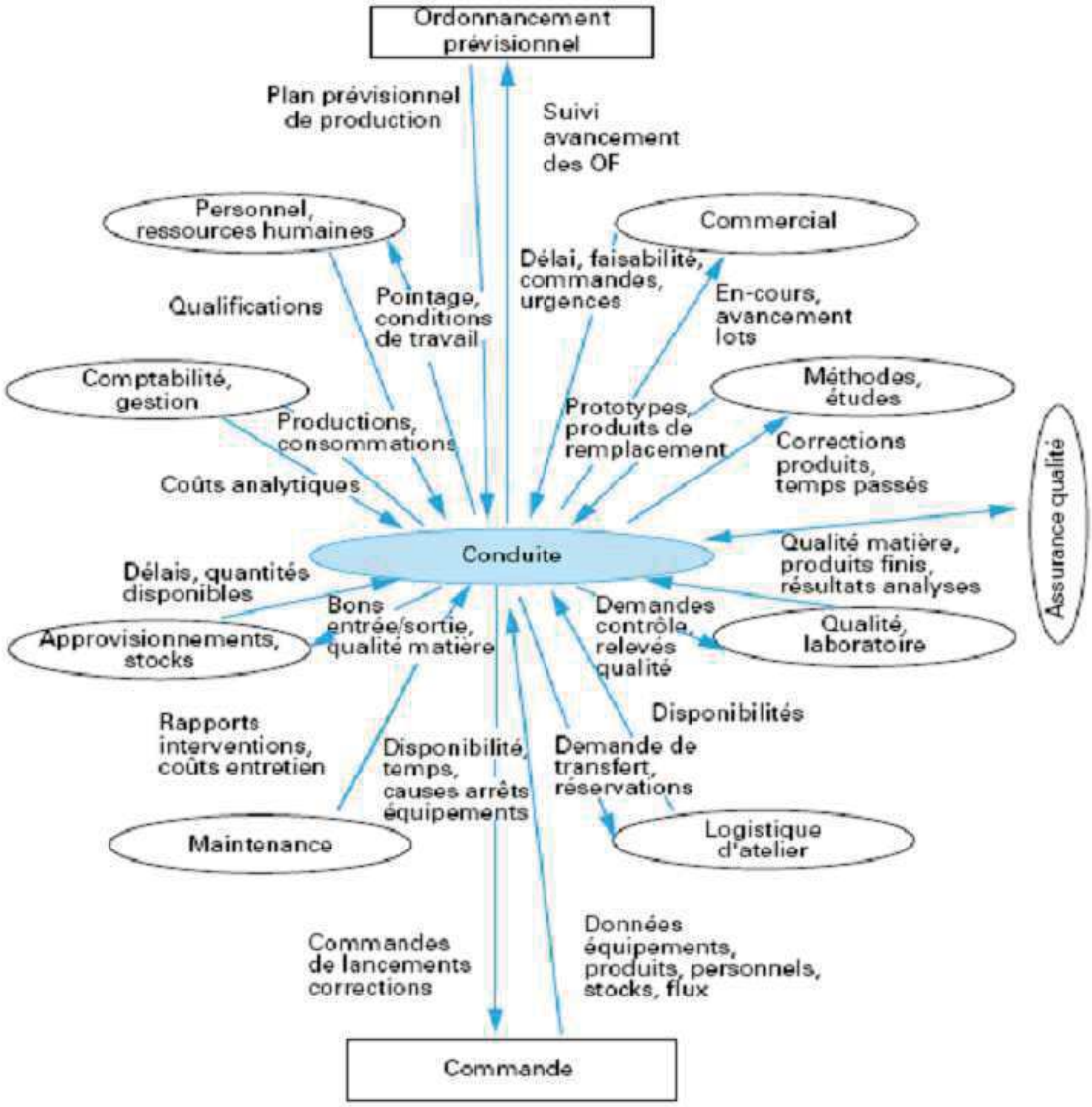

FIGURE 1.5 – La fonction conduite[29]

1.3.3.5 Commande :

Ce niveau, directement en relation avec le système de production, a un rôle d'interfaceet d'interpréteur. Sa tâche essentielle est de traduire un ordre en une séquence d'instructions compréhensibles par la partie opérative. Il est concrétisé soit par un opérateur pilotant une machine et assurant le suivi de réalisation, soit par un automatisme capable d'interpréter unordre et de renseigner la conduite sur l'état d'avancement de celui-ci. C'est le niveau physique, le niveau le plus bas du modèle OSI.

1.4 Typologie des structures de pilotage

1.4.1 Structure centralisée :

L'approche centralisée est très classique (fig.6). Elle est caractérisée par un centre de décision unique qui supervise la production, synchronise et coordonne les différentes ressources. Ce dernier gère seule, en temps réel, les événements qui surviennent tout au long de la production. Le pilotage dans cette structure se fait sur la base d'un ordonnancement prévisionnel où d'un ordonnancement en temps réelle.

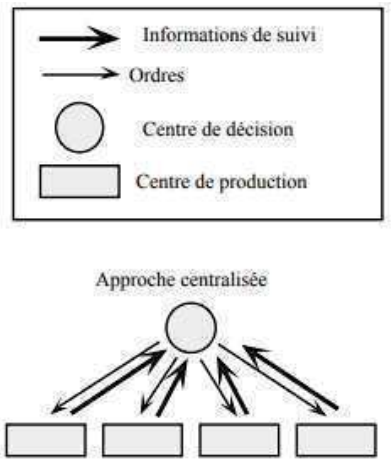

FIGURE 1.6 – Approche centralisée d'un système de production

Parmi les méthodes pour la réalisation temps-réel d'un ordonnancement dans cette approche, on cite :

— L'utilisation d'une méthode multicritère pour la réalisation des tâches les plus contraignantes dans le but de minimiser le temps de cycle [5], cette méthode de pilotage en temps réel dédiée à un atelier flexible robotisé dont elle prend en compte un ensemble de critères conflictuels qui représentent la possibilité, pour une allocation, de libérer un maximum de tâches réalisables. [6]

— L'utilisation des règles de priorité pour la gestion dynamique des files d'attente (le modèle de Merbaki [7], Cette méthode ne concerne que la gestion des files d'attente de

chaque machine et on suppose que les opérations ne peuvent être effectuées sur des machines différentes. Une extension de cette méthode utilise la structuration multi-agents. Ces perspectives vont dans le sens des travaux de Kouiss [8].

— L'utilisation de la simulation dans un but d'anticipation. [9] [10]

L'ordonnancement centralisé permet l'établissement d'un planning d'utilisation des ressources de l'atelier et d'obtenir ainsi des informations quant aux dates de livraison possibles.Ceci ne donne pas toujours des résultats satisfaisant, car on essaie de gérer de façondéterministe un univers aléatoire et dynamique.L'arrivée de nouvelles commandes urgentes, le temps inter-opératoires, le type de fabrication et la charge de l'atelier, la variation des temps opératoires par rapport à l'expérience de l'opérateur, la perturbation et le retard de la fabrication causé par les pièces défectueuses et les pannes de machines, le retard de livraison, ce sont des causes qui perturbent le planning initial .

1.4.2 Structure coordonnée

L'approche coordonnée correspond à un ensemble de structures hiérarchisées où une coopération est possible au sein d'un même niveau (fig 1.7). Ces structures accroissent la capacité de décision au sein de chacun de ces niveaux [6].Cette approche est très utile pour la réalisation d'un ordonnancement prévisionnel, la coopération intra-niveau permet d'avoir une résolution locale des problèmes et des perturbations donc une meilleure intégration temps-réel des perturbations.

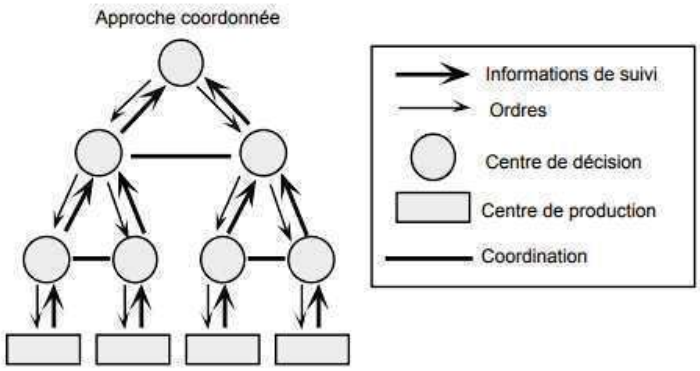

FIGURE 1.7 – approche coordonnée d'un système de production

1.4.3 Structure hiérarchisée :

Etablir un système de gestion de production hiérarchisée permet de simplifier le processus global de décision, elle revient à choisir le nombre de niveaux de prise de décisions, un modèle pour chaque niveau et enfin une méthode pour coordonner le flux des décisions et le retour des informations entre les différents niveaux(fig 1.8). Dans ce système, Les décisions prises à un niveau deviennent des contraintes à satisfaire par les niveaux inférieurs, pour cela on doit vérifier qu'elles soient cohérentes avec les contraintes de ce niveau. Cette approche permet de décomposer le problème global en une succession de sous-problèmes et par la suite de réduire le nombre de variables pour chaque niveau de décision. Les horizons temporels sont longs au

sommet de la hiérarchie. Plus on descend dans la hiérarchie, plus les horizons temporels se raccourcissent et les données manipulées sont détaillées.[11]

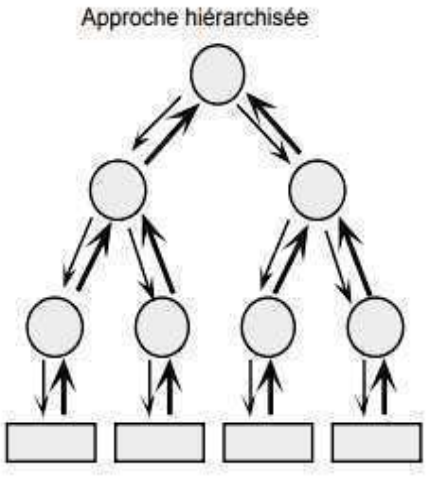

FIGURE 1.8 – Approche hiérarchisée d'un système de production

1.4.4 Structure distribuée :

L'approche distribuée (fig1.9) est fondée sur une distribution totale des capacités de décision parmi un ensemble de ressources pilotant un centre de production [12]. Elle résout la difficulté rencontrée dans la structure coordonnée pour la communication et la circulation d'information entre différents niveaux. Cette approche est particulièrement adaptée dans les productions avec des flux simples, des demandes stables ou à faible variation et l'apparition d'aléas réguliers ou connus.[13]

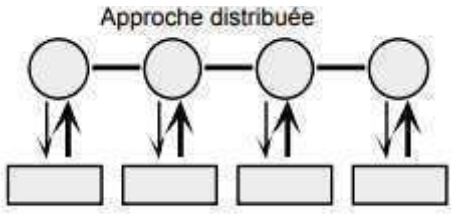

FIGURE 1.9 – structure distribuée d'un système de production

1.4.5 Structure distribuée supervisée

En général, cette structure offre des possibilités d'intercommunication des systèmes de pilotage de différents niveaux permettant l'échange d'informations et la transmission des décisions prises (fig1.10). Cette structure est caractérisée par un ensemble d'entités coopérâtes sous le contrôle d'une entité superviseur dont le rôle est d'imposer, de conseiller ou de modifier une décision afin de respecter un objectif plus global. Ainsi, l'entité superviseur possède une vision plus globale du processus de production [13]. Ce système supérieur sert à commander ou corriger une décision prise afin de satisfaire les objectifs plus globaux.

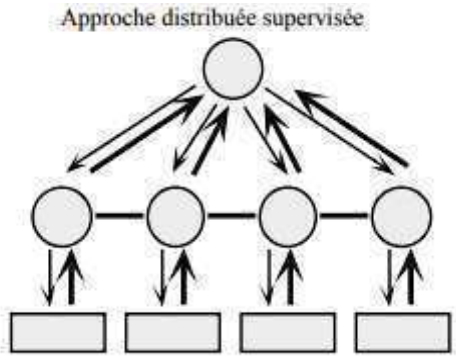

FIGURE 1.10 – Structure distribuée supervisée d'un système de production

1.4.6 Structure décentralisée :

Dans cette structure, tous les centres de pilotage sont au même niveau fonctionnel, ce qui la différencie de la structure distribuée. Il n'existe pas de centre de pilotage de niveau hiérarchique supérieur. Les centres de décision doivent s'auto-organiser pour assurer une gestion cohérente. La plupart du temps, on conserve une certaine structure hiérarchique. Les tâches de suivi de la production ou de lancement des opérations apparaissent comme des entités de haut niveau distribuant le travail aux autres modules. Les centres des différents niveaux sont interconnectés via des bus de communication permettant l'échange d'informations et la transmission des décisions. Ceci n'est rendu possible que par l'arrivée des nouvelles techniques informatiques (système d'information, bases de données partagées, logiciel de type MES) dans les ateliers de production.[13]

L'ordonnancement dans l'approchedécentralisé ne permet pas de faire de prévisions ni de planning car les décisions sont prises au pied de chaque machine. Dès qu'une machine a terminé un lot l'opérateur choisit en fonction d'une règle de priorité définie à l'avance l'OF suivant. Elle a pour objectif d'atteindre les objectifs globaux de l'atelier. Deux règles de priorité sont suivies :

— les règles locales concernent les lots de la file d'attente devant la machine : Premier arrivé premier servi, Priorité de fabrication au lot dont le temps d'opération est le plus petit, la priorité pour les produits dont leur date de livraison est la lus proche pour éliminé le risque d'être en retard, respecter les délais qui consiste à tenir compte le ratio critique temps restant/nombre d'opération restant.

— Les règles de priorité globales nécessitent des informations de la file d'attente mais aussi du reste de l'atelier. Celles-ci devront donc être centralisées même si la décision se répand au niveau local.

1.5 La modèle de kouiss

Kouiss présente une approche distribuée dont le niveau de supervision porte sur les règles d'allocation utilisées par chaque entité. Le modèle exploite la notion d'agent définie par Ferber. Chaque agent possède un soussystème de connaissance, d'expertise et de communication. La séquence des tâches de la file d'attente se fait par l'intermédiaire de règles de décision. Cette séquence se déroule en deux étapes :

— détection des symptômes : des symptômes sont pré-définis en fonction de variables d'états sous la forme d'un ensemble de règles. Ces symptômes sont activés lorsqu'une ou plusieurs variables d'état dépassent un seuil donné. La notion de seuil est décrite de manière approfondie,

— application d'une règle de décision : si nécessaire, un agent applique une règle ou une agrégation de règles pour sélectionner une tâche présente dans sa file d'attente. La décision est donc locale et en temps réel.

Chaque agent utilise les règles indépendamment des autres agents. Le superviseur a pour rôle de faire respecter au mieux les objectifs globaux. (les écarts étant évalués par un jeu de symptômes globaux). Ainsi, il peut imposer une règle particulière à un ensemble d'agents dans le but de respecter une partie de ces objectifs.
Cette approche ne propose pas réellement de méthode d'allocation de tâches,mais plutôt une méthode efficace pour la gestion des files d'attente.

.

Chapitre 2

Chapitre 2

Identification des éléments de la Plateforme

Aujourd'hui, l'automatisation n'est pas une commodité, c'est un réel besoin pour les industriels. Une usine doit autant que faire se peut accroitre de manière continue sa productivité.

L'utilisation d'un automate conduit à une très grande rapidité, une meilleure régularité des résultats et évite à l'homme des tâches pénibles et répétitives.

Après avoir présenté globalement notre plateforme, on passe dans ce chapitre à la présentation des contrôleurs qui seront utilisés pour le pilotage de cette cellule, leurs caractéristiques, et bien sûr le logiciel qui permet leur programmation.

2.1 L'industrie 4.0 et son impact :

Le monde de l'industrie a connu trois révolutions industrielles importantes qui ont changé son fonctionnement. La première révolution apparait au milieu du $18^{ème}$ siècle par le passage d'une production manuelle à une production mécanique avec l'exploitation du charbon et la mise en place des machines à vapeur, ce qui entraine une production plus importante. La deuxième révolution quant à elle apparait à la fin du $19^{ème}$ siècle et introduit du pétrole et de l'électricité afin de moderniser les moyens de production ; et on voit naitre la production en masse du Taylorisme. L'électronique, la télécommunication et les systèmes informatiques sont quant à eux la base de la troisième révolution qui se caractérise par l'automatisation des systèmes (les automates programmables industrielleset les systèmes de supervisions). C'est dans cette période qu'on voit naitre la robotique et la production en grandes masses.Mais ces dernières années, l'apparition et l'évolution des nouvelles technologies numériques a de nouveau bouleversé l'industrie. On parle aujourd'hui des nouveaux modes de fabrications et des interactions reliant les composants, les machines et l'homme à tous les niveaux,et ceci est formulé sous le terme « industrie 4.0 » qui qualifie la quatrième révolution industrielle.

L'industrie 4.0, appelée également usine du futur, se caractérise fondamentalement par une automatisation intelligente et par une intégration de nouvelles technologies à la chaîne de valeur de l'entreprise.

Cette révolution n'est pas technologique car tous les éléments dont on a besoin existent déjà, que ce soit les capteurs intelligents, les automates, les robots collaboratifs … donc la vrai révolution se trouve dans le monde de production qui cherche à construire une nouvelle image différente à celle adaptée pour le moment. L'élément déclencheur de cette révolution est

l'association des capteurs aux algorithmes avec commandes qui permettent de prédire le comportement du produit et influencer son usage courant donc les produits deviennent des produits intelligents et connectés qui fourniront des informations à leurs concepteurs et fabricants.

Il s'agit d'une transformation numérique qui bouleverse l'entreprise manufacturière en apportant des changements radicaux non seulement aux systèmes et processus, mais également aux modes de gestion, aux modèles d'affaires et à la main-d'œuvre.

L'une des premières étapes de cette transformation digitale de l'industrie est souvent l'interconnexion entre les moyens actuels, l'interaction entre les produits et les machines, les machines entre elles, un produit fini qui sera personnalisé et pourra aussi communiquer avec les machines dans sa phase de réalisation. Cette interconnexion nécessite l'utilisation d'une plateforme connectée qui mettra en corrélation le besoin initial du client avec le moyen de production final. [14]

Les défis auxquels font face les entreprises avec l'arrivée de l'industrie 4.0 sont nombreux. Les principaux sont :

— L'entreprise manufacturière doit examiner les nouvelles compétences qui sont requises et le besoin en personnel qualifié.

— La sécurité des données est une préoccupation pour l'ensemble des entreprises avec La multiplication des données et des systèmes qui fait ressortir l'importance de l'aspect sécurité informatique.

— Le besoin d'investissement pour intégrer les nouvelles technologies dans l'usine.

A travers ce projet nous répondront au besoin du client en lui proposant des produits uniques et personnalisés, et en conservant des coûts équivalents. Nous voulons assurer une haute adaptabilité pour répondre à la demande en temps réel. Le passage vers ceci va permettre d'obtenir des pièces à partir de leur définition numérique dans des délais record qui implique une chaîne d'approvisionnement plus réactive et plus efficace. Une bonne gestion des matériaux et de l'énergie et aussi la réduction de temps d'arrêt des machines donc augmentation de la production. L'emploi des capteurs intelligents permet de prévoir les dates d'intervention sur les machines en temps réel, l'état de produit à chaque instant de la production qui permet de connaitre la disponibilité des machines, donc dans le but d'avoir une meilleur gestion et planification de la production, et ainsi avoir une décision locale plus rapide.

2.2 Description du Projet :

La fabrication ou le montage d'un produit nécessite dans la plupart des cas la mise en place d'un processus de fabrication. Ce processus englobe les étapes et les matériels nécessaires afin d'aboutir à un résultat final concluant. Tous ces éléments se regroupent dans une chaîne de production, une ligne d'assemblage composée de postes où plusieurs objets technologiques interagissent les uns avec les autres afin de pouvoir créer les actions que le produit doit subir. Dans notre projet, le but est d'avoir une chaîne de production plus intelligente qui réagit selon le type de produit présent dans son convoyeur principale et qui sera conforme au principe de l'industrie 4.0.

Globalement, notre cahier de charge se compose de deux postes principaux qui vont assembler les pièces dans la palette qui passe dans le convoyeur. Chaque poste possède un robot, un identificateur RF380R, des capteurs de proximité et aussi des convoyeurs qui ramènent les pièces à assembler(deux types de pièces différents pour chaque poste). L'ensemble de ces objets

est commandé par un automate ET200 SP que ce soit le signal des capteurs qui détecte l'arrivée du produit à l'un des emplacements, l'identification qui se fait par le SIMATIC RF380R en lisant les tags et identifiant le type du produit. Ceci donne comme résultat la tâche que l'automate doit envoyer par communication SOCKET au robot, c'est-à-dire le mouvement que doit être appliqué pour récupérer une pièce et la poser en dessus de la palette à une position particulière.

Quant aux automates ET200 SP, chacun travaille indépendamment étant associé à son poste. Cependant, ils sont reliés à l'*opencontroller* par une architecture décentralisée. L'*opencontroller* a pour rôle de définir la tâche aux automates ET200SP à l'aide d'une application basée sur le concept des systèmes multi-agents implémentés sur sa partie PC.

En réalité, nous ne disposons pas de tous les éléments de la chaîne. C'est pour cela qu'on a adopté le principe HARDWARE IN THE LOOP, une méthode de simulation caractérisée par l'association de véritables composants, connectés à une partie temps-réel simulée.Donc, on a construit l'environnement en simulation dans le logiciel V-REP. Nous n'aurons en réalité qu'un seul robot qui est le LBR iiwa 7r800 du premier poste, les RF380r et les capteurs des deux postes et aussi le convoyeur principal.Cependant, le deuxième robot, toutes les pièces, les palettes et les convoyeurs qui ramènent les cubes à assembler seront seulement en simulation.

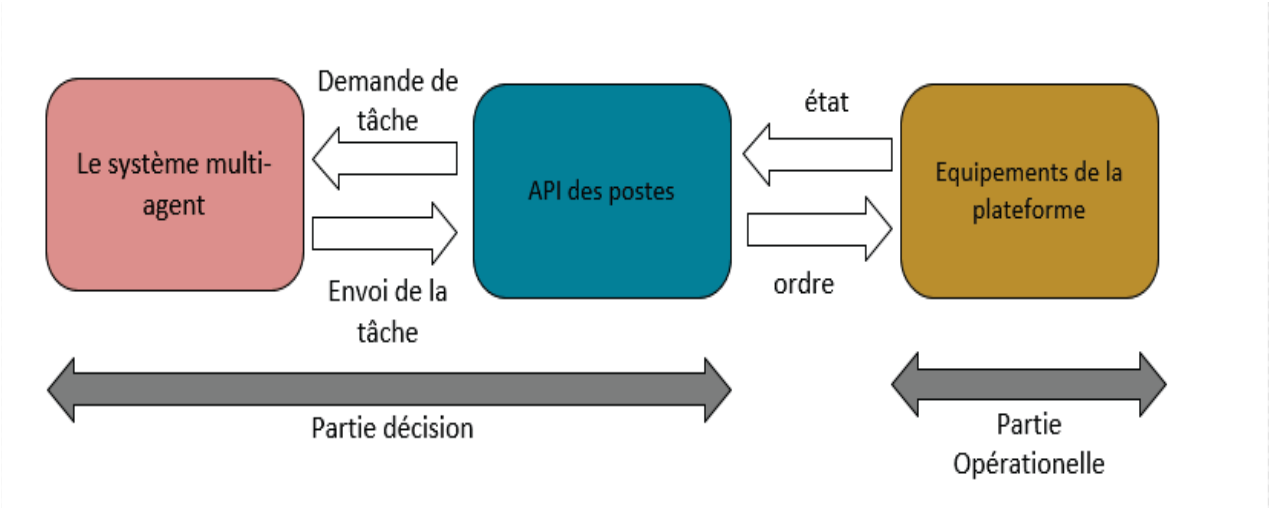

FIGURE 2.1 – Architecture globale du système

2.3 Décomposition de la chaine de production :

La chaine de production est un système automatisé de production dont l'objectif principal est de gérer les flux de matières et d'informations et de de prendre des décisions par rapport aux objectifs prioritaires définis.

2.3.1 . Systèmes de décisions :

Le centre des décisions est la partie commande.Dans ce qui suit, nous allons présenter les composants de ces systèmes de décision et leurs logiciel :

2.3.1.1 Automate SIMATIC ET200SP :

SIMATIC ET 200sp est un système d'entrée/sortie décentralisée qui donne la possibilité d'échange de signaux entre l'automate centrale CPU et les modules extérieurs à travers la liaison PROFINET. Ses systèmes multifonctionnels conviennent pour différents champs d'application et permettent d'aligner plusieurs E /S.

Généralement installé dans un rail, il comprend :

— un module d'interface qui communique avec tous les automates conformes au standard PROFINET IEC 61158 ;

— jusqu'à 64 modules d'E/S, enfichés dans des unités de base passives dans n'importe quelle combinaison ;

— un module serveur qui complète la structure du SIMATIC ET 200SP. cit2ref3

- CPU 1512-PN (6ES7512-1DK01-0AB0) :

  La CPU contient le système d'exploitation et exécute le programme utilisateur. Le programme utilisateur se trouve sur la carte mémoire SIMATIC et il est traité dans la mémoire de travail de la CPU.

  Les interfaces PROFINET se trouvant sur la CPU permettent la communication simultanée avec des appareils PROFINET, des contrôleurs PROFINET, des appareils IHM, des consoles de programmation, d'autres automates et d'autres systèmes. La CPU 1512SP-1 PN prend en charge le fonctionnement comme contrôleur IO, périphérique d'entrée ou comme CPU autonome. Cette interface possède trois ports. Le port 3 se trouve sur la CPU. Le port 1 et le port 2 se trouvent sur le BusAdapter (Adaptateur de bus), on peut aussi raccorder ce système d'E/S à une PG / un PC ou un appareil HMI via le port 3. [18]

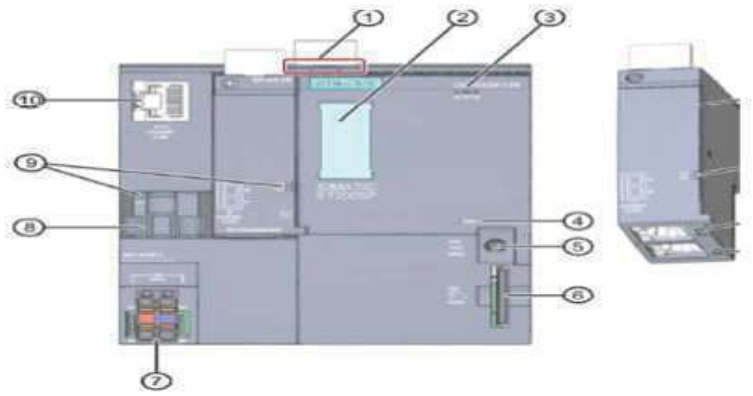

FIGURE 2.2 – Représentation de la CPU avec le module Bus Adaptater [18]

① Déverrouillage du profilé support ;
② Bandes de repérage ;
③ LED d'affichage d'état et de défauts ;
④ LED d'affichage de la tension d'alimentation
⑤ Sélecteur de mode de fonctionnement
⑥ Logement pour la carte mémoire SIMATIC
⑦ Raccordement pour tension d'alimentation

Caractéristique CPU 1512-SP1PN	**CPU 1512-SP1PN**
Tension d'alimentation	24 VCC
Plage admissible, limite inferieur	19.2 V
Plage admissible, limite supérieur	28.8 V
Consommation de courant d'entrer « valeur nominal »	0.6 A
Consommation de courant d'entrer « valeur nominal »	4.7 A (Valeur nominal)
Nombre d'interfaces PROFINET	1
Puissance dissipé	5.6 W

TABLE 2.1 – Tableau des caractéristiques

⑧ Support de câble et fixation pour le port P3 de l'interface PROFINET

⑨ LED pour affichages d'état de l'interface PROFINET : LK1 et LK2 sur Bus Adapter, LK3 sur CPU

⑩ Port P3 de l'interface PROFINET.

- Les modules d'entrée/sortie de l'automate ET 200 SP et son alimentation :
  - **Le module d'entrée numérique :** DI 16x24VDC ST
  - **Le module de sortie numérique :** DQ 16x24VDC/0.5A ST
  - **Module d'alimentation** : BA 2XRJ45

2.3.1.2 SIMATIC ET 200SP Open Controller :

Le SIMATIC ET 200SP Open Controller est le premier automate qui combine les fonctions d'un automate logiciel basé sur PC avec la visualisation, les applications PC et les E/S centrales dans un appareil compact. Il peut être librement combiné à des modules standards de l'ET-200SP. Le contrôleur logiciel SIMATIC S7-1500, déjà installé et préconfiguré, version de SIMATIC S7-1500 pour PC, est utilisé pour le contrôle. Ce contrôleur fonctionne indépendamment de Windows et garantit ainsi une haute disponibilité du système. Cela facilite le démarrage rapide du contrôleur et prend en charge les mises à jour Windows et le redémarrage en cours de fonctionnement. Une panne dans Windows n'a aucun impact sur le contrôle.

Le SIMATIC ET 200SP Open Controller fournit le know-how et la protection d'accès, ainsi que des fonctions d'automatisation importantes, telles que le positionnement des axes, le diagnostic des systèmes intégrés et les interfaces avec Profinet. Il est extensible de manière flexible par les modules ET 200SP et est optimisé pour la fabrication de machines en série ainsi que pour les machines à architecture distribuée. Il permet de réaliser des solutions d'automatisation avec un rapport qualité-prix particulièrement favorable. Par exemple, la visualisation peut être réalisée simplement via un SIMATIC Industrial Flat Panel connecté via l'interface graphique, éventuellement avec une fonctionnalité multitouch. Pour la mise en service, la souris et le clavier peuvent être directement connectés via des interfaces USB. Parallèlement, les interfaces Ethernet Gigabit prennent en charge une connexion haute performance aux réseaux de niveau supérieur. [17]

➤ CPU 1515SP PC (6ES7512-1DK01-0AB0) :

La CPU 1515SP PC est un automate programmable sur base PC sur le modèle duET 200SP. Elle est utilisée pour la commande et la visualisation. Le logiciel IPC DiagBasequi l'accompagne offre des fonctions pour le diagnostic de base et vous aidera à utiliser le
BIOS.
La CPU 1515SP PC possède les caractéristiques techniques suivantes :

- Le support de mémoire est une carte CFast enfichable sur laquelle sont préinstallés :
  - — Système d'exploitation Windows Embedded Standard 7
  - — S7-1500 Software Controller : CPU 1505SP
  - — En option avec IHM : WinCC Runtime Advanced à partir de la version V14 SP1
- Interfaces
  - — Une interface pour les BusAdapters ET 200SP interchangeables en vue du raccordement PROFINET IO (2 ports)
  - — Une interface pour coupler des appareils via l'Ethernet industriel (Gigabit Ethernet)
  - — 3 interfaces pour appareils USB
  - — Une interface DVI-I pour un écran
  - — Un logement plombé pour la carte CFast
  - — Un logement pour carte SD/MMC comme lecteur optionnel supplémentaire
- Tension d'alimentation 1L+ 24 V DC (TBTS/TBTP). Le connecteur fait partie de la fourniture. [19]

➤ Les interfaces de SIMATIC Et200sp Open Controller :

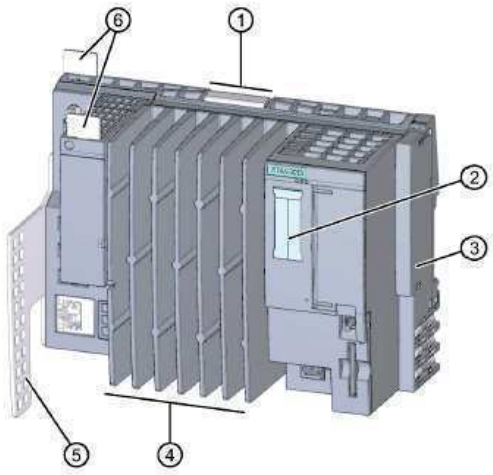

FIGURE 2.3 – Vue de Profil du dispositif
[19]

① Déverrouillage du profilé-support
② Bandes de repérage
③ Module serveur
④ Ailettes de refroidissement
⑤ Décharge de traction
⑥ Etiquettes de repérage de référence

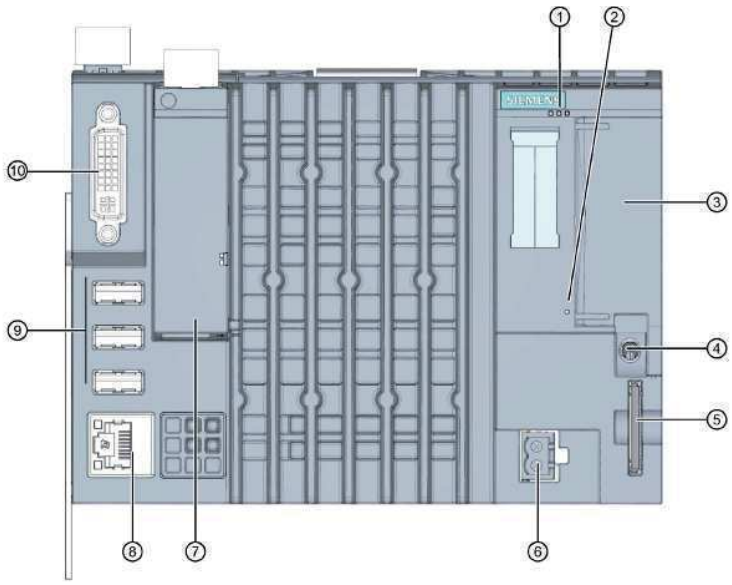

FIGURE 2.4 – Éléments de raccordement et de commande de la CPU 1515sp PC [19]

① LED de signalisation pour le mode de fonctionnement réelle, l'état de diagnostic actuels de la CPU : LED RUN/STOP (vert en mode RUN et jaune en mode STOP),LED ERROR (rouge en cas d'erreur), LED MINT (jaune).Chaque combinaison de couleurs possibles de ces LED donne une signification en rapport avec le Software Controller.

② LED de signalisation Power : (LED éteinte en cas d'absence de l'alimentation ou alimentation faible, LED Jaune allumée dans le cas de la présence de l'alimentation avec l'arrêt de système d'exploitation,LED verte allumée dans le cas de la présence de l'alimentation avec le fonctionnement de système d'exploitation)

③ X50 : Logement pour la carte CFast (mémoire flash, c'est l'unique mémoire de masse de la CPU sur laquelle sont préinstallés le système d'exploitation Windows Embedded Standard 7, S7-1500 Software Controller : CPU 1505SP PC, WinCC Runtime Advanced à partir de la version V14 SP1).

④ Sélecteur de mode : RUN, STOP et MERS pour l'effacement général de la CPU

⑤ X51 : Logement pour une carte SD/MMC optionnelle : utilisé comme une mémoire supplémentaire des données. Les données peuvent être enregistrées sur cette carte par l'intermédiaire de la page Windows. On ne peut pas enregistrer sur cette dernière le système d'exploitation, ni le logiciel exécutif, ni le projet.

⑥ X80 : Raccordement de la tension d'alimentation 24 V CC

⑦ X1 PROFINET (LAN) : emplacement pour BusAdapter pour le raccordement de PROFINET IO affichage d'état pour PROFINET

⑧ X2 PN/IE (LAN) : raccordement GbE Ethernet avec affichage intégré

⑨ X60, X61, X62 : 3 ports USB (pour la souris, le clavier, clé USB, etc.)

⑩ X70 DVI-I : raccordement du moniteur (pour le branchement d'un écran ou HMI)

2.3.1.3 Logiciel de Programmation TIA PORTAL :

Totally Integrated Automation Portal (Tia PORTAL) est un environnement de travail créé par siemens qui permet la configuration, programmation, vérification et diagnostic de tous les

automates SIMATIC et qui combine les logiciels Simatic STEP7 et SIMATIC Wincc.

Le logiciel STEP7 est l'outil de programmation des différents automates de la gamme siemens. Le logiciel Wincc quant à lui est un logiciel de contrôle de donnée pour commander et superviser le système.

Les fonctions suivantes peuvent être utilisées pour automatiser une installation :

— Configuration et paramétrage du matériel.

— Paramétrage de la communication.

— Programmation.

— Test, mise en service et dépannage avec les fonctions d'exploitation et de diagnostic.

— Documentation.

— Génération d'écrans de visualisation pour les Basic Panels SIMATIC avec WinCC Basic intégré.

➤ Environnement de Tia Portal :

Lorsque Tia portal est lancé, on distingue deux parties importantes dans l'environnement du logiciel :

- Vue du projet :

  Cette vue contient l'ensemble des éléments nécessaires pour mettre en œuvre une solution d'automatisation elle comporte une arborescence avec les différents éléments du projet. Les éditeurs requis s'ouvrent en fonction des tâches à réaliser. Données, paramètres et éditeurs peuvent être visualisés dans une seule et même vue. Comme le montre la figure suivante :

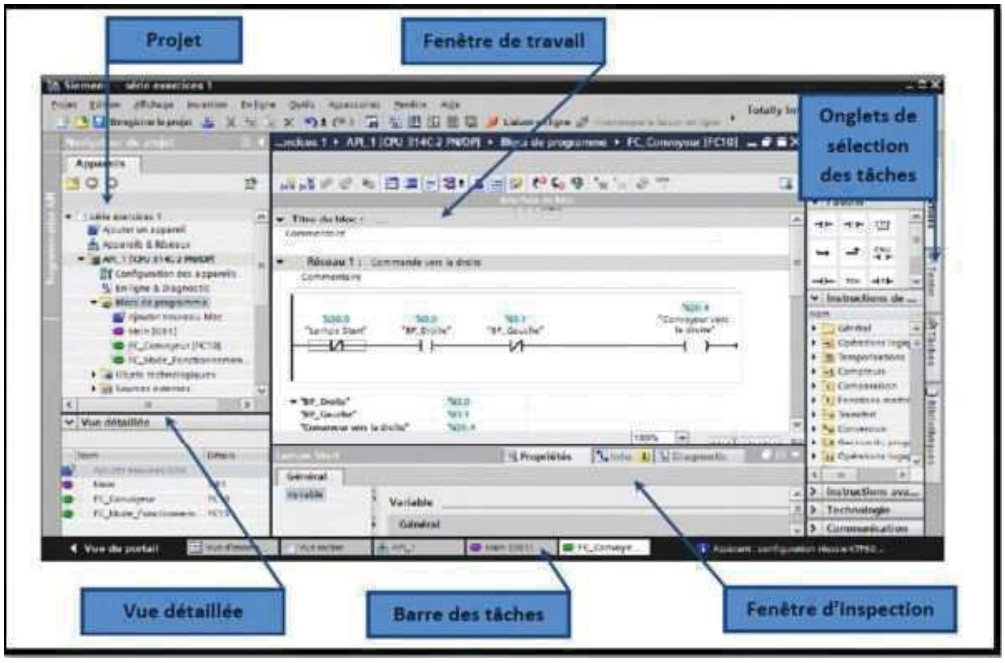

FIGURE 2.5 – Vue du projet

— Projet : permet d'accéder en détailles à tous les objets qui composent la solution d'automatisation à réaliser que ce soit la configuration du matériel, ou encore la programmation.

— Onglets de sélection des tâches : Cet environnent contient énormément de données qui seront utilisés en fonction de l'objet sélectionné.

— La fenêtre de travail : permet de visualiser les objets sélectionnés dans le projet pour être traités. Il peut s'agir des composants matériels, des blocs de programme, des tables des variables, des interfaces homme machin (IHM).

— La fenêtre d'inspection : permet de visualiser des informations complémentaires sur un objet sélectionné où sur les actions en cours d'exécution (propriété du matériel sélectionné, message d'erreur lors de la compilation des blocs de programme,...).

- Vue du portail :

  Il donne la possibilité de traiter une catégorie d'actions, la fenêtre affiche la liste des actions pouvant être réalisé pour la tâche sélectionnée, comme le montre la figure suivante :

FIGURE 2.6 – Vue du portail

➤ Création d'un projet et mettre en œuvre un environnement de travail :

Afin de crée un projet dans la vue de portail il suffit de sélectionner « créer un projet ». On doit par la suite un nom au projet, choisir un emplacement où il sera enregistré, indiqué un commentaire ou encore définir l'auteur du projet. Une fois cette étape passé le projet sera créé.

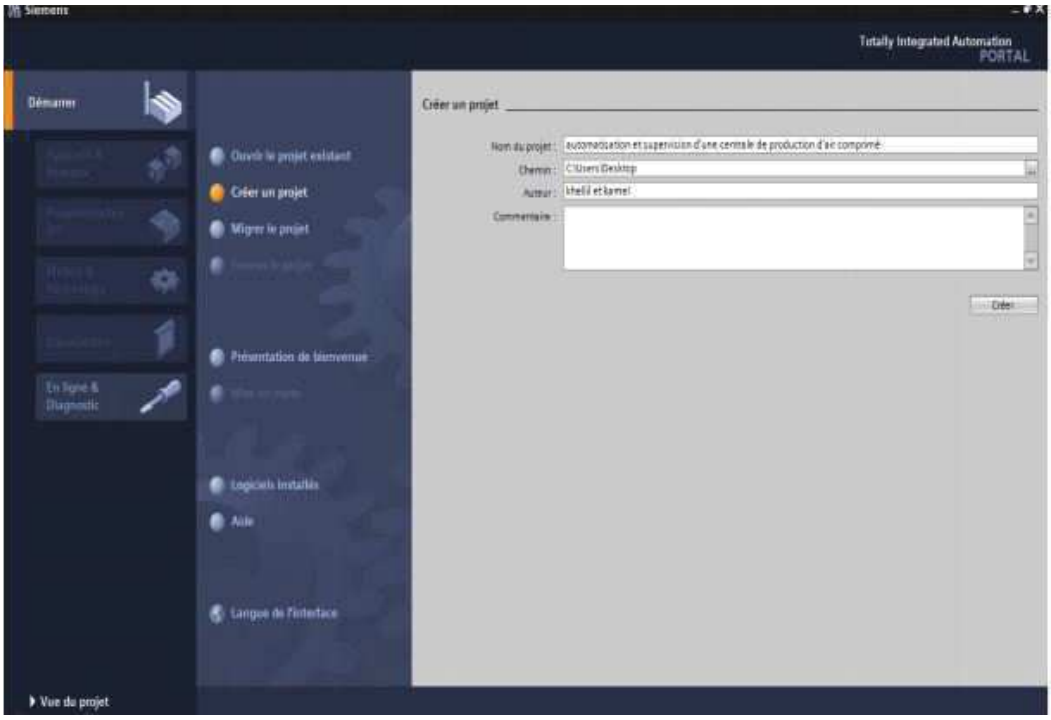

FIGURE 2.7 – Création d'un projet

➤ Configuration du matériel :

Une fois le projet crée, on peut configurer la station de travail. La première étape consiste à définir le matériel existant. Pour cela, on peut passer par la « vue du projet » et cliquer sur «ajouter un appareil » dans le navigateur du projet.

Une liste des éléments apparait pour choisir le matériel qu'on veut utiliser que ce soit API, IHM ou système PC.il se peut de Commencer parfaire le choix de la CPU pour ensuite venir ajouter les modules complémentaires (alimentation, E/S TOR ou analogiques, module de communication, Etc.).

Les modules complémentaires de l'API peuvent être ajoutés en utilisant le catalogue. Si l'on veut ajouter un écran ou un autre API, il faut repasser par la commande « ajouter un appareil » dans le navigateur du projet. Lorsque l'on sélectionne un élément à insérer dans le projet, une description est proposée dans l'onglet information.

➤ Adressage des entrées/sorties et programmation des blocs :

Le logiciel met à disposition différents types de blocs qui contiennent le programme et les données correspondantes. Selon les exigences et la complexité du processus, il est possible de structurer le programme en différents blocs : OB, FB et FC.

- Les blocs d'organisation (OB) :
  Ils constituent l'interface entre le système d'exploitation et le programme utilisateur.
  Les OB sont programmables par l'utilisateur, ce qui permet de déterminer lecomportement de la CPU.
- Les blocs de fonctions (FC) :
  Ce sont des blocs pour la programmation qui n'ont pas de mémoire de données dans laquelle il est possible d'enregistrer les valeurs de paramètres de bloc.

Les données des variables temporaires sont perdues après l'exécution de la fonction.
Si on veut mémoriser ces données, il faut utiliser des opérandes globaux.

- Les blocs fonctionnels (FB) :
  Ce sont des blocs de programmation qui mémorisent durablement leurs paramètres d'entrée, de sortie et d'entrée/sortie dans des blocs de données d'instance afin qu'il soit possible d' y accéder même après le traitement de blocs.

2.3.2 Système opérationnel :

La partie opérative du système est le sous-ensemble qui effectue les actions physiques, mesure des grandeurs physiques et rend compte à la partie commande. Elle se compose de :

2.3.2.1 Capteurs de Proximité :

Ce sont les éléments permettant d'obtenir des informations précises sur l'évolution de système.

Dans notre cas le type de capteurs le plus convenable ce sont des détecteurs photoélectriques qui permettent la détection d'objets de toutes natures, ils réalisent la détection d'une cible (objet ou personne) au moyen d'un faisceau lumineux. Ses deux constituants de base sont un émetteur et un récepteur de lumière, La détection est effective quand la cible pénètre dans le faisceau lumineux et modifie suffisamment la quantité de lumière reçue par le récepteur pour provoquer un changement d'état de la sortie.

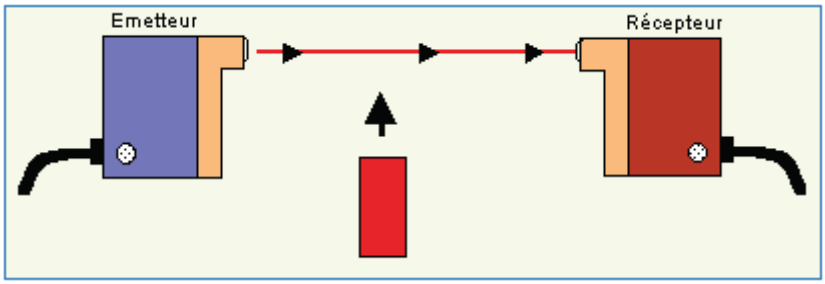

FIGURE 2.8 – Un capteur de proximité photoélectrique

2.3.2.2 SIMATIC RF380R :

Simatic RF 380R est un outil d'identification par radiofréquence de la gamme des RF300 de Siemens. Il est utilisé principalement en industrie afin d'identifier des objets à proximité. Cet appareil identifie seulement des objets à partir d'une certaine distance et doivent obligatoirement avoir un tag RFID.

Pour exécuter le Module RFID, le logiciel TIA Portal possède une bibliothèque de fonctions qui s'appelle "SIMATIC Ident" qui permet la création d'un programme qui pourra contrôler le module à travers le contrôleur. La tâche consiste à lire l'identifiant qui est un code sous forme de tableau de caractère mais aussi l'écriture sur le Tag ce qui veut dire que le code peut être modifié selon l'utilisateur.

Le RF 380R ne peut pas communiquer directement avec l'automate. Il a besoin pour cette tâche d'un module de communication qui est le RF180c. [20]

FIGURE 2.9 – SIMATIC RF380R

① Interface RS-232 Ou RS-422
② Lampe qui informe sur l'état de l'appareil
➤ SIMATIC RF180C :

Le module RF180C est un module indispensable utilisé pour n'importe quel type de contrôleur (automate Siemens) pour l'exploitation des identifiants RFID parmi eux le SIMATIC RF380C.

C'est un module de communication qui a été développé afin de communiquer entre les systèmes d'identification RFID et les automates de type Siemens, à l'aide des blocks de programmation qui sont utilisés afin de contrôler le lecteur. [21]

FIGURE 2.10 – Présentation du RF180C

Toutes les commandes qui seront envoyées de l'automate vers l'identifiant RFID passent par ce module de communication et les réponses qui seront transmises du lecteur RFID vers l'automate empreintes aussi le même chemin.

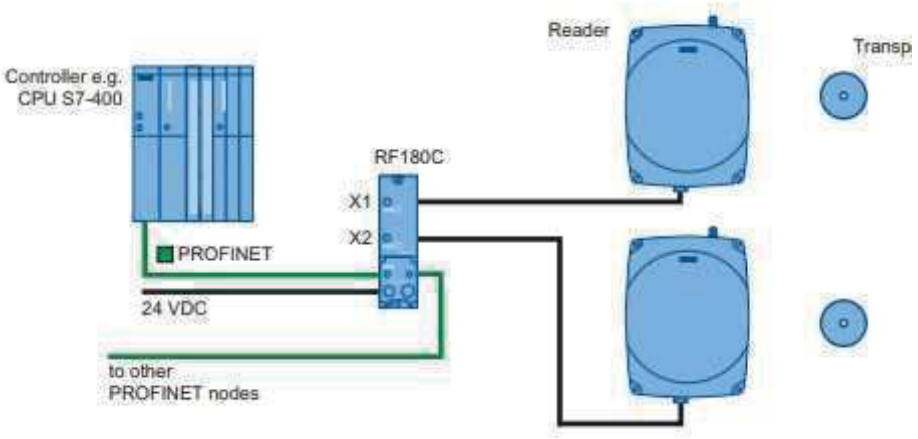

FIGURE 2.11 – Exemple de raccordement entre un lecteur RFID et le module RF180C

2.3.2.3 Le convoyeur :

Un convoyeur est un mécanisme ou une machine qui véhicule des pièces différentes entre plusieurs postes de fabrication ou de montage, dans notre système, il sera basé sur une structure en boucle sur lequel se déplacent des palettes. Son sens de rotation est unidirectionnel. Il sera entrainé par un moteur asynchrone avec un variateur de vitesse qui régulera sa vitesse.

FIGURE 2.12 – Un convoyeur

2.3.2.4 Robot KUKA LBR iiwa 7r800

Le LBR iiwa « assistant de travail industriel intelligent » est un bras robotique léger et performant conçu par l'entreprise allemande KUKA. Le rôle principal de ce robot est d'effectuer des tâches en collaboration avec l'homme, cela permet à l'employé de travailler plus vite et mieux, optimiser la production et la fiabilité d'un poste de travail. Ce rebot est équipé de capteurs de couple intégrés dans chacun de ces axes qui lui permet de détecter un contact imprévu et de réduire immédiatement sa force et sa vitesse. Il intègre aussi une fonction anti-écrasement, cette fonction lui permet de s'arrêter puis reculer s'il y a un contact avec une partie de corps humain ce qui assure la sécurité des personnes travaillant à proximité de ce robot. Ce Cobot peut être programmé par apprentissage, il suffit de lui guider sur une trajectoire et lui apprendre une ou des positions en marquant des pauses et mémoriser ces derniers.

Ce robot industriel peut être utilisé dans les modes suivants :

- mode manuel avec Vitesse réduite (T1) : utilisé pour la programmation, test de programme, et l'enseignement.
- mode manuel avec Haute Vitesse (T2) : utilisé pour le test de programme.
- mode Rétraction de robot contrôlé (CRR) : utilisé lorsque le robot industriel est arrêté par le contrôleur de sécurité.
- mode Automatique (AUT) : utilisé pour l'exécution automatique des programmes pour les robots industriels avec et sans contrôleur de niveau supérieur.

Dans les trois premiers modes le robot ne peut exécuter des programmes qu'en mode JOG, donc il faut maintenir un bouton d'activation et le bouton Démarrer clé pour l'exécution de programme. Un arrêt de sécurité 1 se déclenche si le commutateur d'activation sur le SmartPAD est relâché ou bien il est appuyé à fond. Le relâchement de la touche Démarrer déclenche un arrêt de la catégorie d'arrêt 1. [22]

➤ Caractéristique technique du LBR iiwa :

FIGURE 2.13 – Vue d'ensemble du système de robot
[22]

① Câble de connexion au SmartPad

② KUKA SmartPad control panel

③ Le robot LBR iiwa 7r800

④ Câble de connexion au contrôleur de robot KUKA Sunrise Cabinet

⑤ Contrôleur de robot KUKA Sunrise Cabinet

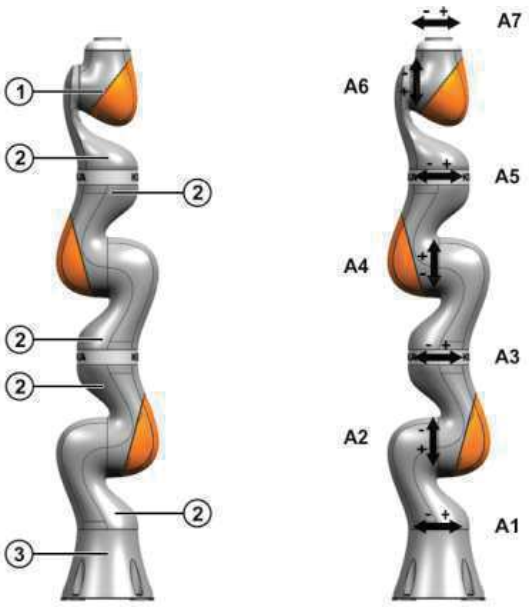

FIGURE 2.14 – Axes de robot et assemblées principales
. [22]

① In-line wrist (poignet)
② Joint module (Ils sont constitués d'une structure en aluminiumles unitésd'entrainement sont situées à l'intérieur de ces modules).
③ Base frame (base de robot).
A1 – A7 Les 7 axes de robot

Catégorie	Petits robots (3Kg-10Kg)
Charge	7 Kg
Charge totale nominale	7 kg
Portée max	800 mm
Nombre d'axes pouvant être commandés	7 axes
Répétabilité de la pose qui est de :	$\pm$ 0.1 mm
Poids	22 Kg
Position(s) de montage	Mur/ Plafond/ Sol
Température ambiante	5 °C à +45 °C
Contrôleur	KUKA Sunrise Cabinet
Mode de protection	IP 54
Degré de protection du poignet en ligne	IP 54
Capteurs	Capteurs de couple articulaires sur chaque axe
Source d'énergie	Alimentation électrique (220V) Connexion électrique ou pneumatique pour les effecteurs ou les préhenseurs
Programmation	Java / Possibilité de programmation par apprentissage

TABLE 2.2 – Caractéristiques Techniques du Robot

2.3.3 Les interfaces

La communication entre la partie opérationnelle et la partie commande nécessite une interface pour transformer ou adapter le message envoyer. Dans notre cas, nous avons utilisé les interfaces suivantes :

2.3.3.1 Réseau Profinet

La communication industrielle joue un rôle important dans l'ensemble des techniques d'automatisation. PROFINET, le standard ouvert basé sur Industrial Ethernet est la technologie de communication adopté par SIEMENS pour relier en réseau ses appareils de différentes gammes.

PROFINET est un réseau de haut niveau utilisé pour les applications d'automatisation industrielle. Il est basé sur des technologies Ethernet standard. Il utilise un matériel et des logiciels Ethernet traditionnels pour définir un réseau qui structure la tâche d'échanger des données des alarmes et des diagnostics avec des contrôleurs programmables et d'autres contrôleurs d'automatisation.

- Ethernet :

  Ethernet est un protocole de réseau local à commutation de paquets permettant aux appareils connectés de communiquer entre eux grâce au protocole TCP / IP dont il représente la couche liaison de données. Ethernet décrit comment les périphériques réseau peuvent formater et transmettre des paquets de données afin que les autres périphériques du même réseau local puissent les reconnaître, les recevoir et les traiter. Un câble Ethernet est le câblage physique encapsulé sur lequel les données transitent.

- TCP/IP :

  TCP/IP est une suite de protocoles. TCP/IP signifie «Transmission Control Protocol/Internet Protocol».C'est en fait une architecture réseau en 4 couches dans laquelle les protocoles TCP et IP jouent un rôle prédominant.

  TCP/IP représente d'une certaine façon l'ensemble des règles de communication sur internet et se base sur la notion adressage IP, c'est-à-dire le fait de fournir une adresse IP à chaque machine du réseau afin de pouvoir acheminer des paquets de données. L'idée de base est de rendre ces réseaux homogènes en leur imposant un protocole commun qui est le TCP. ET pour passer d'un sous-réseau à un autre, il faut passer par le protocole IP qui gère le routage.

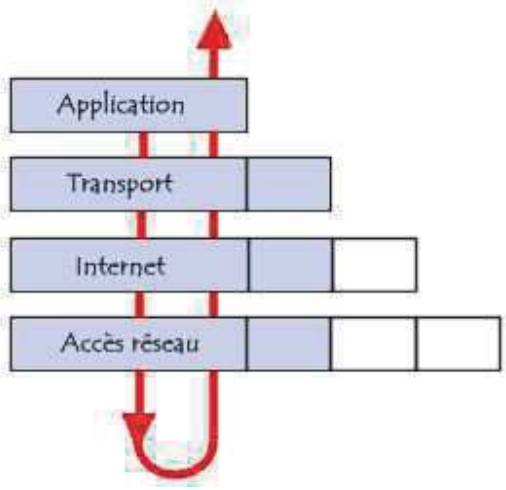

FIGURE 2.15 – Représentation du modèle TCP/IP.

- Profinet IO :
  Le PROFINET IO est un standard Ethernet industriel ouvert et destiné au monde de l'automatisation. Contrairement au PROFINET CBA (Component Based Automation) qui est destiné aux systèmes distribués, le PROFINET IO se concentre sur l'échange de données entre automates programmables. PROFINET IO utilise le canal TCP/IP pour le paramétrage, la configuration et les opérations de lecture / écriture acycliques mais aussi Le canal RT ou Real Time qui est utilisé pour le transfert de données cyclique standard et les alarmes, cette communication contourne l'interface TCP / IP standard pour accélérer l'échange de données avec les automates programmables et Le canal, Isochronous Real Time (IRT).

2.3.3.2 Socket

Dans chaque installation industrielle on trouve plusieurs équipements : des automates programmables, des interfaces hommes /machines, des ordinateurs, des équipements d'entrées /sorties communiquent entre eux grâce à des protocoles de communication. Cependant, un protocole de communication est une spécification de plusieurs règles pour un type de communication particulier, il définit les types de données qui peuvent être envoyés, comment chaque type de message sera identifié, quelles actions peuvent ou doivent être entreprises par les participants à la conversation etc. Ces protocoles doivent respecter le modèle OSI (Open System Interconnexion).

Dans notre projet nous avons utilisé deux types de communication, socket et Profinet. La communication socket assure l'échange d'informations entre l'automate et le système simulé, entre l'automate et le robot réel et aussi entre l'automate et notre application multi-agent implémenté sur l'automate PC. Profinet est utilisé entre l'automate et200sp de chaque poste et le module de lecteur RF180C.

➤La communication par Socket :

La Socket est un point de terminaison mis à l'écoute sur le réseau, afin d'établir une communication serveur/client inter processus via un port bien définit et de faire la transition des données logicielles. Elle se situe juste au-dessus de la couche transport du modèle OSI (protocoles UDP ou TCP) et utilise les services de la couche réseau (protocole IP / ARP).

Modèle des sockets	Modèle OSI
Application utilisant les sockets	Application
	Présentation
	Session
UDP/TCP	Transport
IP/ARP	Réseau
Ethernet, X25, ...	Liaison
	Physique

FIGURE 2.16 – Position du socket dans le modèle OSI

La communication par Socket nous permet de connecter à une machine distante, utiliser un port, attendre une connexion de l'extérieur, recevoir et envoyer des données, écouter les communications entrantes. On distingue deux modes de communication par socket :

- Le mode connecté qui utilise le protocole TCP/IP, l'adresse de destination n'est pas nécessaire à chaque envoie de données car une connexion durable est établie entre les deux processus.
- Le mode non connecté qui utilise le protocole UDP/IP. Ce mode nécessite la définition de l'adresse de destination à chaque envoi, aucun accusé de réception n'est donné.

Le principe de communication par Socket est basé sur 3 étapes essentielles :

- La création d'une Socket coté serveur va spécifier le Port de communication. Cette dernière s'exécute en permanence et attendre l'appel d'une socket client, il accepte ou refuse la communication après la comparaison avec les paramètres de la connexion définis. Plusieurs clients peuvent communiquer à un seul serveur mais l'inverse est impossible.
- La création d'une Socket côté client où on indique le port de communication avec le serveur et l'adresse IP de l'appareil sur laquelle se trouve ce dernier. Le client lance la communication avec le serveur.
- Une fois la connexion entre serveur et client faite, un échange des données à travers des fonctions d'écritures/lecteurs peuvent se faire entre ces deux. à la fin de l'échange, la fermeture de la connexion se fait soit de côté serveur ou côté client. [23]

Chapitre 3

Chapitre 3

Commande de la plateforme

Tout système de production automatisé comprend deux parties coopérants, une partie opératoire et une partie commande qui dialoguent ensemble. La partie opératoire est le processus physique, elle comprend un ensemble d'éléments tels que des capteurs, des actionneurs, des machines, des robots, etc. La partie commande est l'ensemble des moyens de traitement de l'information qui assure la commande et la coordination des tâche.

Dans ce chapitre, on présente le processus mis en place afin de pouvoir faire fonctionner notre projet. On explique le rôle de chaque matériel et logiciel utilisé et on justifie l'utilisation du principe Hardware-In-The-Loop. Dans une autre partie, on énonce l'utilité de chaque élément dans la réalisation de la plateforme réelle et virtuelle.

3.1 Cahier de charge :

Afin de pouvoir faire l'assemblage de plusieurs produits de façon intelligente (connaitre le type et l'état du produit tout au long de la chaine), nous avons mis en place un système automatisée qui suit le diagramme d'activité. (figure 3.1)

Ce diagramme représente le fonctionnement général d'un poste de la chaine. Il fait le lien entre le programme du robot, la simulation V-rep, les lecteurs RFID et l'application multi-agent à travers les blocs fonctions mis en place dans le logiciel TIA pour la programmation de l'automate ET200SP. Il est à noté que la programmation des deux postes sont similaires, la seule différence se présente sur les points de positionnement des pièces dans la palette et que le premier poste 1 communique avec le robot réelle tandis que le deuxième poste communique avec celui de la simulation seulement.

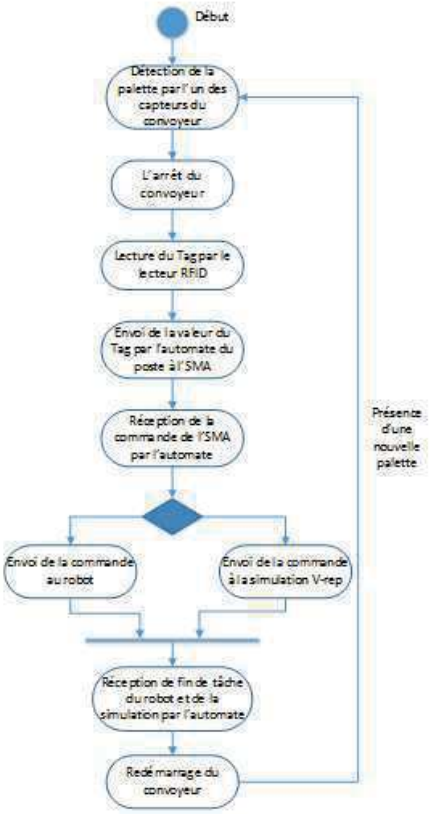

FIGURE 3.1 – Diagramme fonctionnel du poste 1

3.2 Détection de la palette :

La plateforme d'assemblage se compose de plusieurs éléments interconnectés dans le but d'assembler différentes pièces afin de donner un produit prêt. L'une des pièces principales de cette plateforme est le convoyeur, un outil qui sert à véhiculer les pièces entre les deux postes, qui est basé sur une structure en boucle sur laquelle se déplacent les palettes.Son sens de rotation est unidirectionnel et contient près de chaque poste un capteur ultrason et un le module RFID RF380R, un lecteur des tags qui sont implémentés dans les palettes. Chaque tag contient un code qui correspond à un type de produit.

Le capteur de proximité est relié à l'automate directement et procure un signal tout ou rien tandis que le module RFID RF380R est relié au contrôleur par le module de communication RF180C via le protocole de communication PROFINET.

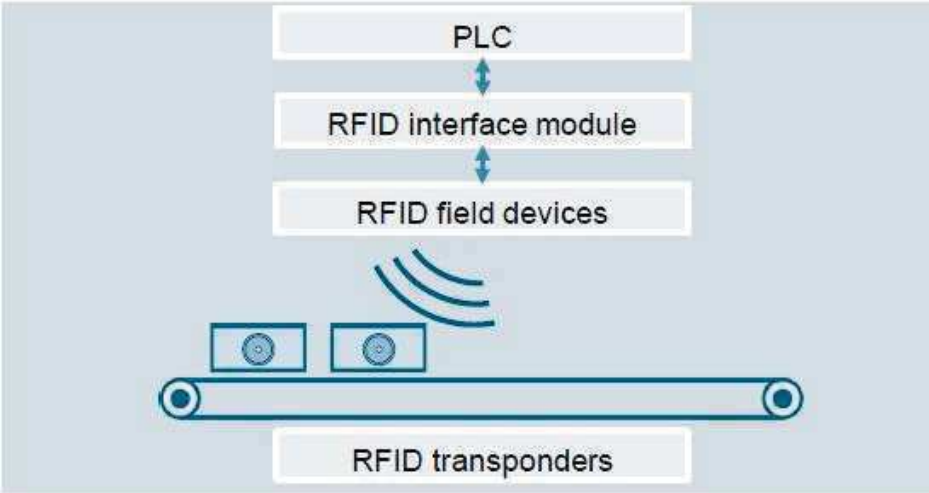

FIGURE 3.2 – fonctionnement du RFID

Pour exécuter le module SIMATIC RF 380R sur l'automate ET200SP, il existe des blocs fonction adaptés à la programmation avec STEP 7 TIA Portal.Il est important d'utiliser ces blocs ainsi qu'une configuration précise afin de pouvoir fonctionner le lecteur RFID.

— **L'objet technologique :**
L'objet technologique est un paramètre d'entrée « HW-CONNECT » des blocs d'identification utilisée pour adresser les modules / lecteurs. Il se présente sous forme de bloc de donnée et contient les informations sur le type et les paramètres du lecteur tel que la configuration du module de communication qui est dans notre cas le RF180C. Il est à noter que chaque objet est associé à un seul lecteur RFID pour assurer un adressage correct. [24]

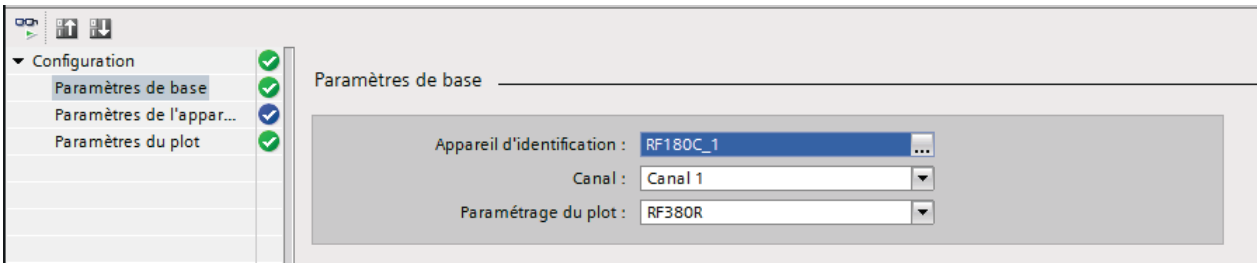

FIGURE 3.3 – Configuration d'un Objet technologique
[24]

3.2.1 Explication des blocs d'identification :

Toutes les blocs d'identifications utilisés ont les paramètres de sorties suivantes :

3.2.2 Fonctionnalité du bloc « RESET-RF300 » :

Le bloc est utilisé pour préparer le lecteur à fonctionner. Lorsque le bloc est exécuté, tous les paramètres définis dans l'objet technologique seront transférés à l'appareil. Aussi dans le cas

Paramètre	Description
DONE	Travail accompli avec succès
BUSY	Travail actif
ERROR	Présence d'une erreur
STATUS	Message d'erreur en cas ou ERROR est actif
PRESENCE	Présence du transpondeur dans le champ RF

TABLE 3.1 – Paramètres du bloc d'identification

du lecteur RF380R l'antenne qui détecte le tag s'allume automatiquement lors de l'exécution du bloc d'où la non nécessité du bloc set-antenna. [24]

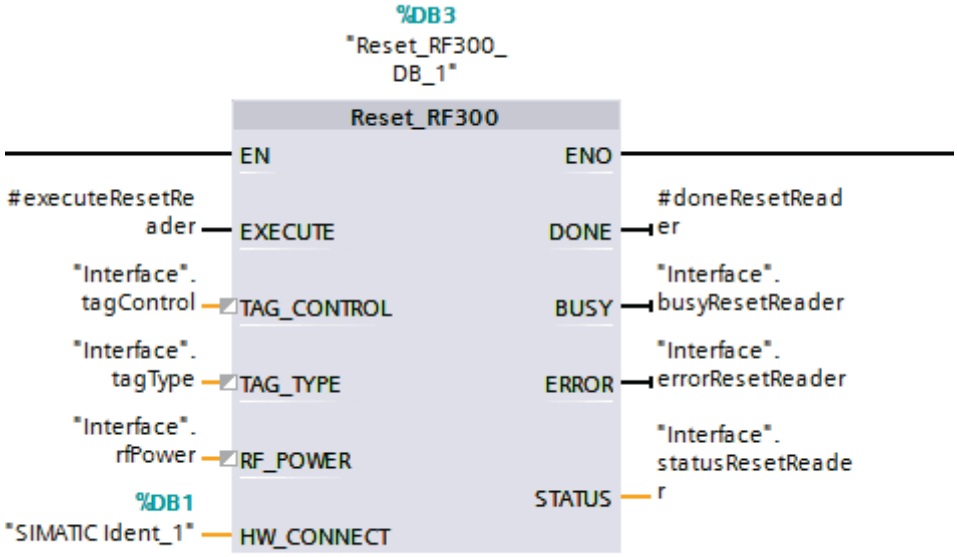

FIGURE 3.4 – Bloc RESET dans le programme

Les paramètres utilisés :

— EXECUTE : pour démarrer le bloc.

— HW-CONNECT : pour l'objet technologique.

3.2.3 Fonctionnalité du bloc « READ » :

Le bloc "READ" permet de lire les données qui sont inscrites dans l'identifiant (tag). Les paramètres du bloc "ADDR_TAG" et "LEN_DATA" permettent de définir la zone mémoire à lire. Les données lues sont fournies via le paramètre "IDENT_DATA". [24]

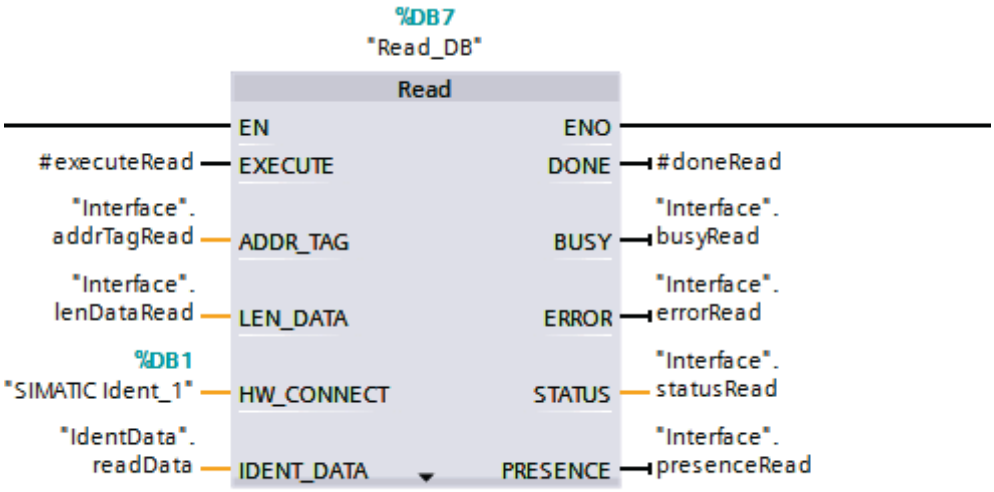

FIGURE 3.5 – Représentation du bloc READ

Les paramètres utilisés :

- EXECUTE : lancer la lecture.
- ADDR_TAG : l'adresse physique à partir du quelle le lecteur commence la lecture .
- HW-CONNECT : l'objet technologique qui fournit l'adresse physique.
- IDENT-DATA : l'identifiant à lire, il se présente sous forme de tableau de BYTES.

3.2.4 Fonctionnalité du bloc « WRITE» :

Le bloc "WRITE" permet d'écrire des données sur l'identifiant à partir du lecteur. Les paramètres "ADDR_TAG" et "LEN_DATA" dans ce cas permettent de définir la zone de mémoire du transpondeur à écrire. Les données à écrire sont fournies par le paramètre "IDENT_DATA". [24]

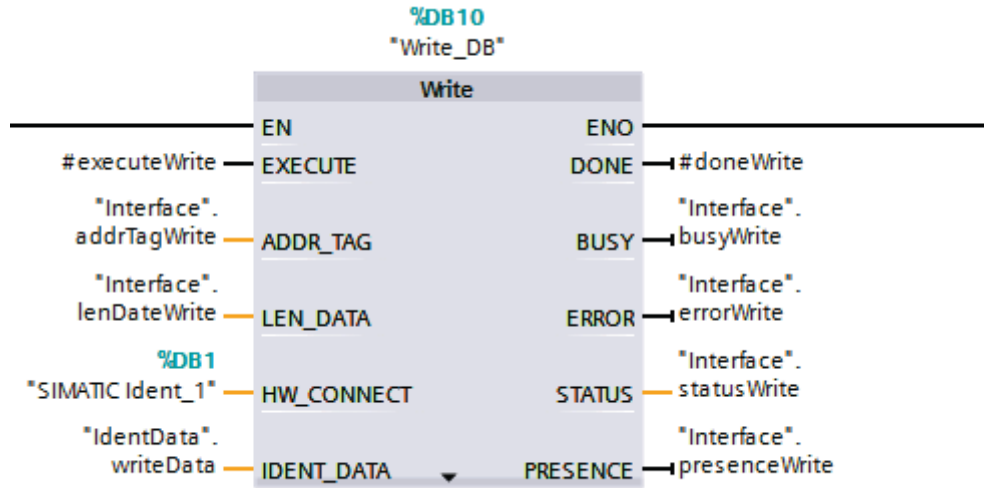

FIGURE 3.6 – Présentation du bloc WRITE

Les paramètres utilisés :

— EXECUTE : lancer la lecture.

— ADDR_TAG : l'adresse physique à partir le lecteur commence l'écriture .

— HW-CONNECT : l'objet technologique qui fournit l'adresse physique.

— IDENT-DATA : l'identifiant que doit le RF écrire sur le Tag, il se présente sous forme de tableau de BYTES.

3.3 Commander le robot kuka lbr iiwa 7 r800 :

Autre élément qui interagit avec les pièces de la plateforme est le robot. Sonfonctionnementen réalité dans notre cas se base sur l'intervention de deux parties : la programmation de ses tâches qui se fait par le biais d'un logiciel associé au robot « SUNRISEWORKBENSH » et la commande à distance par l'automate ET200SP qui définit la tâche.

3.3.1 Commande à distance par l'automate ET200SP :

L'automate ET200SP a la possibilité de communiquer par socket et échanger des données avec d'autres types d'appareil comme le robot en utilisant la bibliothèque de communication de TIA Portal.
Les instructions qu'on a utilisées pour faire la communication sont les suivantes :

- **TSEND-C : pour établir une connexion et émettre des données**
  Lors de l'exécution du bloc « TSEND-c », il établit une communication TCP(ou UDP selon le paramétrage). Une fois la connexion faite, l'automate le maintien automatiquement et envois par la suite les données en direction de la communication existante.

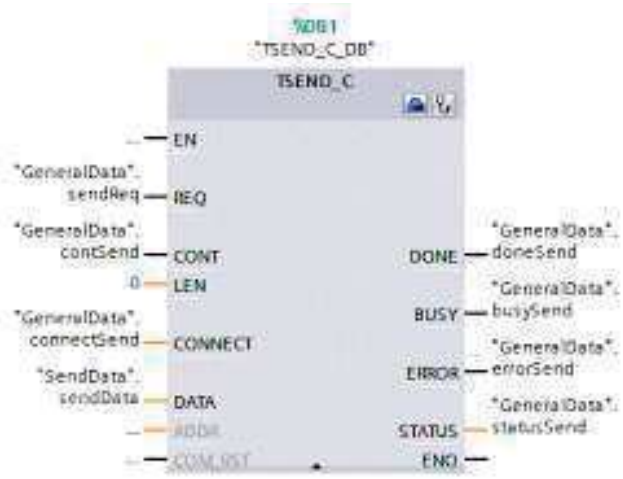

FIGURE 3.7 – Bloc de communication TSEND-C

Les paramètres du bloc :

— CONNECT : Une fois la configuration de la communication est faite, un bloc est généré automatiquement.

— CONT : Pour établir la connexion, le paramètre doit être réglé sur "1" et prend fin lorsque le paramètre est défini sur «0».

— DONE : Si la connexion a été établie avec succès, le paramètre est réglé sur "1" pour un cycle.

— REQ : active la demande d'envois par un front montant.

— LEN :pour spécifier le nombre maximal d'octets transmis lors de l'envoi.

— DATA :Pointeur sur la zone d'envoi qui contient l'adresse et la longueur des données à envoyer.

- **TRCV-C : pour établir une connexion et recevoir des données**

  L'actionnement du bloc TRCV-C permet aussi la mise en place d'une connexion TCP (aussi UDP) et ceci est établi lorsque le paramètre CONT est à 1. Lorsque la connexion est faite, le bloc sera prêt à recevoir les données.

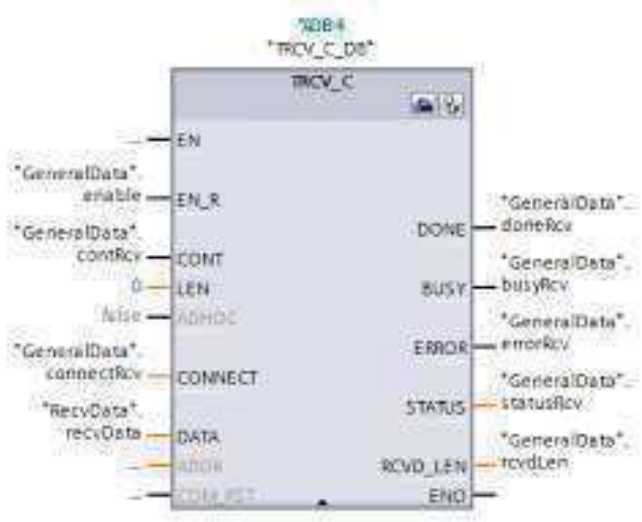

FIGURE 3.8 – Bloc de communication TRCV-C

— CONNECT : Contient le bloc généré lors de la configuration du réseau.

— CONT : Pour établir la connexion, le paramètre doit être réglé sur "1" et prend fin lorsque le paramètre est défini sur «0».

— DONE : Si la connexion a été établie avec succès, le paramètre est réglé sur "1" pour un cycle.

— REQ : active la demande d'envoie par un front montant.

— LEN :pour spécifier le nombre maximal d'octets transmis lors de la réception.

— DATA :Pointeur sur la zone d'envoi qui contient l'adresse et la longueur des données reçues.

Remarque :la communication socket TCP est établie en spécifiant le port et les adresses des deux parties à connecter. Aussi l'automate est configuré en tant que client donc c'est lui qui établit la communication.

3.3.2 Logiciel de Programmation du robot SUNRISE WORKBENSH :

C'est un logiciel de programmation conçu spécialement pour la manipulation des Robots du type KUKA IIWA qui sont des robots industriels dans lequel les tâches de programmation et de commande sont mises en place.
KUKA Sunrise.Workbench est l'environnement de développement de la cellule robotisée (station). Il offre les fonctionnalités suivantes pour le démarrage et le développement d'applications :
Mise en service :

1. Installation du logiciel système
2. Configuration de la cellule robot (station)
3. Modification de la configuration de sécurité
4. Création de la configuration d'E / S
5. Transférer le projet sur le contrôleur de robot

Développement d'applications :

1. Programmation d'applications robotiques en Java
2. Gestion de projets et de programmes
3. Modification et gestion des données d'exécution
4. Synchronisation de projet
5. Débogage distant (localisation et élimination des défauts)
6. Définition des points d'arrêt
7. Exécution du programme en une seule étape (arrêt après chaque ligne de programme)
8. Affichage et modification des variables d'application pendant l'exécution du programme
9. Modification d'un programme en cours d'exécution.

3.3.3 Programmation des tâches du robot :

La programmation par apprentissage est l'une des méthodes utilisées pour faire fonctionner le robot KUKA LBR IIWA. Cette méthode consiste à montrer au robot ce qu'il doit faire, ce qui veut dire amener l'outil sur les poses (les repères) désirées successives représentant la tâche.A travers un boîtier de commande, il était facile de manipuler le robot et ainsi enregistré dans une mémoire la trajectoireà exécuter (qui se compose en une succession des prises et des angles des 7 axes), sous contrôle d'un opérateur humain afin d'assurer une représentation de la tâche unique.

La génération et la gestion des trajectoires et les opérations fait appel quant à elle au logiciel de programmation adapté à ce type de robot SUNRISE WORKBENSH qui, à partir du modèleet des trajectoires à réaliser, permet d'élaborer un programme qui contient la succession des commandes et des actions.Celui-ci est par la suite transféré au boitier de commande (Smart Pad) où il suffit de sélectionner son nom dans la rubrique application puis l'exécuter. Une fois l'exécution faite, Le robot sera prêt à recevoir et à exécuter les consignes.

1	Cette ligne contient le nom du package dans lequel se trouve l'application robot.
2	La section import contient les classes importées nécessaires à la programmation de l'application robot.
3	En-tête de l'application robot (contient le nom de la classe de l'application robot)
4	Les tableaux de données des classes requises dans l'application robotsont déclarés ici. Lors de la création de l'application robot, les instances des classes nécessaires sont automatiquement intégrées. Par défaut, il s'agit de l'instance du robot utilisé, ici un LBR.
5	Les valeurs initiales sont affectées aux tableaux de données créés dans la section de déclaration.
6	La programmation de l'application robot commence par cette méthode.

TABLE 3.2 – les commandes du programme JAVA du robot

Dans le cas de notre travail, nous avons mis en place un programme JAVA (langage de programmation imposé par le logiciel)qui suit une structuration bien précise et se compose de plusieurs méthodes qui sont illustrées dans l'image suivante :

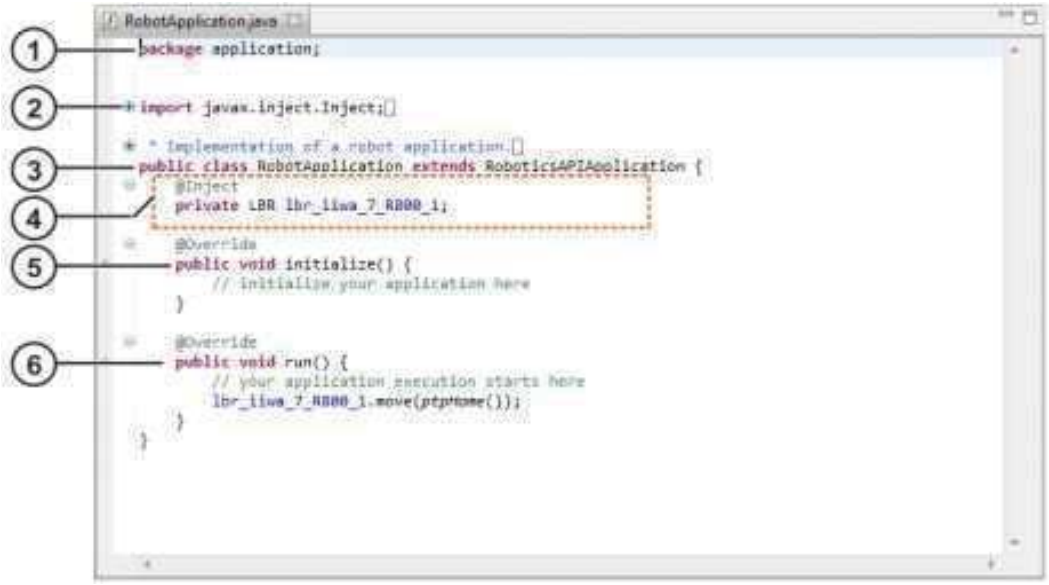

FIGURE 3.9 – Structure de l'application robot

La méthode RUN qui inclue le corps de notre programme contient tout en premier la communication Socket,ou le robot communique directement avec l'automate ET200SP en utilisant le bloc mentionné précédemment par l'envoi d'un message qui contient commande et la réception de la fin de tâche.

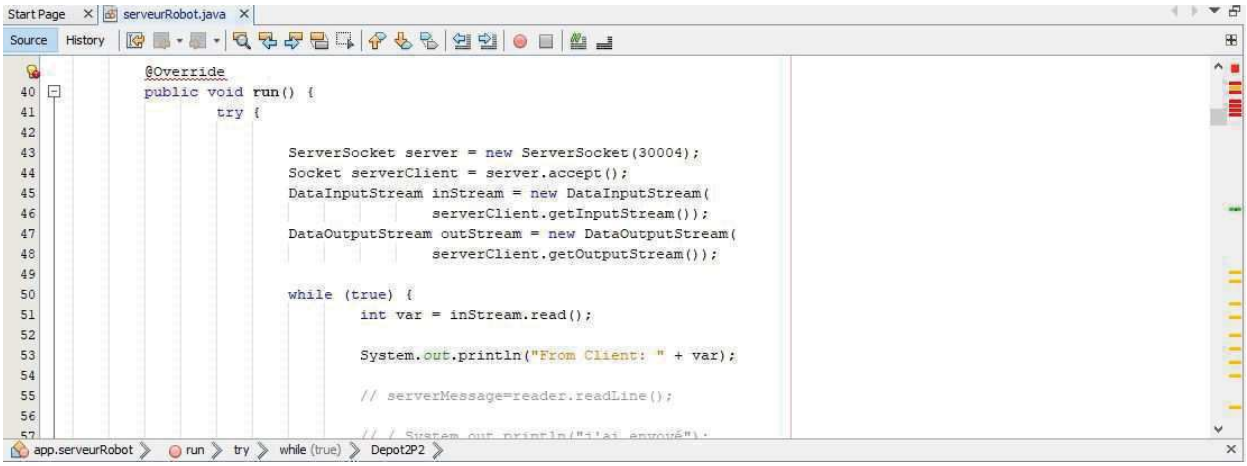

FIGURE 3.10 – Déclaration de la communication Socket

L'élaboration d'une communication par Socket faits appel à des fonctions Java prédéfini :

1. **ServerSocket server = new ServerSocket(30004) :** Créer une Socket sur le port fourni en paramètre.
2. **Socket serverClient = server.accept () :**pour attendre une nouvelle connexion pour que si un client (l'automate) tente de communiquer avec le serveur, la méthode accept() renvoie une Socket qui encapsule la communication avec ce client.
3. **DataInputStreaminStream= newDataInputStream(serverClient.getInputStream()) :**Renvoie un flux en entrée pour recevoir les données de la Socket.
4. **DataOutputStream outStream = new DataOutputStream (serverClient.getOutputStream()) :**Renvoie un flux en sortie pour émettre les données de la Socket.
5. **inStream.read() :** Pour la réception de l'information sur la tâche qui est de type entier.
6. **outStream.writeUTF(serverMessage) :** Pour l'envoi de l'information sur la fin de la tâche qui est de type chaine de caractère**.**
7. **server.close () :** Pour fermer la communication et libérer le port.

Par la suite,les coordonnéesdes points qui ont été enregistrés par apprentissage sont récupérés pour définir les repères avec la fonction **Joint Position**, celle-ci est proposé par la bibliothèque du robot « RoboticsAPIApplication » et prend en argument l'orientation des 7 axes.

Enfin pour faire fonctionner le robot, le programme qui reçoit la tâche par Socket exécute une série de fonctions **move** qui prend en argument le nom du repère auquel le robot doit se déplacer et la vitesse avec laquelle il doit bouger selon l'argument envoyé et à la fin renvoi une fin de tâche.

3.4 Simulation :

Traditionnellement, les essais en contrôle-commande industriel se font directement sur l'équipement physique. Cette pratique a certes l'avantage d'être fidèle à la réalité du terrain, mais elle implique de gros risques pour la partie matérielle sans oublier que la production régulière doit être arrêtée pendant les tests et le débogage. L'absence ou le mal dimensionnement d'un dispositif implique un perdre de temps et d'argent.

Récemment, une nouvelle approche de test de commande apparaître dans le milieu industriel, la simulation hardware in the loop. Cette méthode de test est basée sur la simulation du systéme physique pour protéger les composants de ce dernier en gardant le contrôleur réelle. Elle fournisse une réalité virtuelle à notre système c'est à dire le processus à contrôler (actionneurs, système physique, et ses capteurs) peut être composé soit d'éléments simulés, soit d'éléments réels, soit comme dans notre cas un mixe des deux. Cependant, la simulation HIL est un outil qui permet de relever les défis de disponibilité, coût, sécurité, etc. en rendant les tests de ces systèmes plus efficaces.

Dans notre projet, nous avons réalisé la simulation sur le logiciel de simulation en temps réel V-rep, la commande est centralisée dans une application multi-agent implémentée dans l'automate PC Simatic ET200sp Open Controller, les automates de chaque poste font le lien entre ces deux parties par une communication Socket.

3.4.1 Logiciel de simulation V-rep :

V-REP, le simulateur de CoppeliaRobotics, c'est un simulateur du robot 3D basé sur une architecture de contrôle distribué : les programmes de contrôle (ou les manuscrits) peuvent être directement attachés aux objets de scène et courir simultanément d'un mode *threaded* ou non-threaded. Il propose un mode de fonctionnement en temps-réel. Cela signifie que la simulation est optimisée pour respecter des contraintes de temps-réel et permet d'avoir une simulation fidèle à la réalité, utilisable en combinaison avec un système réel. Il implémente par défaut différents moteurs physiques et de nombreux robots : industriel, mobile, volant, rampant, humanoïde, etc. Il permet aussi de construire un modèle de robot personnalisé (CAO ou via V-REP), d'éditer et simuler des sous-systèmes capteurs, mécanismes, etc.).

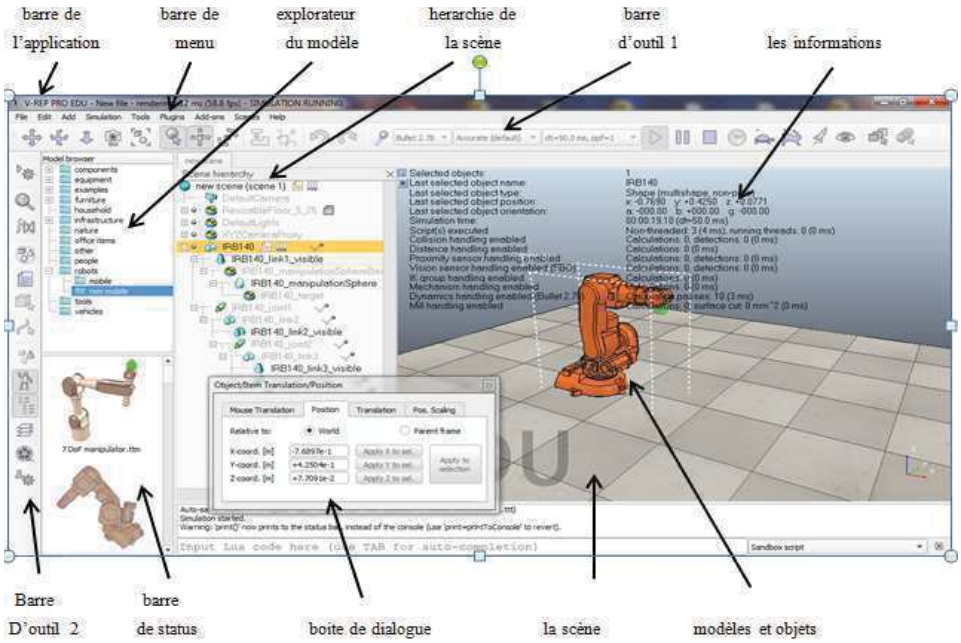

FIGURE 3.11 – Application V-REP

V-REP est une excellente solution pour la Surveillance de sécurité, la présentation de produit, le contrôle à distance, le contrôle des composants, le prototypage et la vérification, la modélisation et simulation des systèmes de production, etc.

Une scène de simulation V-REP contient plusieurs objets de scène ou objets élémentaires qui sont assemblés dans une hiérarchie arborescente (PATH, DUMMY, GRAPH, SHAPE, PROXIMITY SENSOR, VISION SENSOR, JOINT, etc..). Ces objets sont rarement utilisés seuls, ils opèrent plutôt sur (ou sont en connexion avec) d'autres objets de la scène (par exemple, un capteur de proximité détecte des formes ou des modèles qui se croisent avec son volume de détection). De plus, plusieurs modules de calcul V-REP peuvent fonctionner directement sur un ou plusieurs objets de scène. Voici les principaux modules de calcul de V-REP :

— Module cinématique direct et inverse

— Module dynamique ou physique qui permet de manipuler le calcul et l'interaction de la dynamique du corps rigide.

— Module de détection de collision qui permet un contrôle rapide des interférences entre toute forme ou collection de forme et le calcule de contour de la collision.

— Module de calcul de distance minimale qui permet de calculer, visualiser et enregistrer à chaque instant de simulation la distance minimal entre toute forme ou la collection de formes.

— Module de planification de trajectoire qui permet d'effectuer des tâches de planification de parcours holonomes ou non holonomes via une approche dérivée de l'algorithme RRT, basés sur la bibliothèque OMPL.

Tous les modules de calcul nécessitent la définition d'une tâche ou d'un objet de calcul spécifiant la scène sur laquelle le module doit fonctionner sauf les modules dynamique ou physique qui opèrent directement sur tous les objets de scène activés de manière dynamique.

V-REP est un simulateur robotique très complet et aussi très simple d'utilisation. Six approches de programmation ou de codage différentes sont prises en charge, chacune présentant

des avantages particuliers (et évidemment aussi des inconvénients) par rapport aux autres, mais toutes les six sont compatibles les unes des autres (c'est-à-dire qu'elles peuvent être utilisées simultanément).

- **Script intégré** (Embedded Script) c'est l'approche de programmation la plus simple qui consiste à écrire des scripts LUA. Cette méthode permet de personnaliser une simulation (scène ou modèle). Elle contient plus de 500 fonctions API. On trouve plusieurs types de Script : main script, child script et callback script, threaded ou non-threaded. C'est une interface de programmation locale, elle ne peut être utilisée que dans le même processus.
- **Add-on** : cette méthode permet de personnaliser rapidement le simulateur. Ils ne doivent pas être spécifiques à une simulation ou à un modèle donné, ils doivent offrir une fonctionnalité plus générique, liée au simulateur. Il existe deux types de add-on : Fonctionsadd-on qui sont des fonctions qui ne seront exécutées qu'une fois, si elles sont sélectionnées par l'utilisateur. Le deuxième type c'est les Scripts add-on qui sont automatiquement chargés au démarrage du programme et permet à la fonctionnalité de V-REP d'être étendue par des fonctions écrites par l'utilisateur. Ils sont écrits en Lua et exécutés en permanence pendant l'exécution du simulateur et ils exécutent efficacement en arrière-plan.
- **Plugin** : cette méthode consiste essentiellement à écrire un plugin pour V-REP. Elle est utilisée pour fournir une simulation avec des commandes Lua personnalisées, donc utilisable avec la première méthode. Ils fournissent à V-REP une fonctionnalité spéciale nécessitant soit une capacité de calcul rapide (les scripts sont généralement plus lents que les langages compilés), une interface spécifique avec un périphérique matériel (ex un vrai robot), ou une fonction spéciale. Plugin permet aux fonctionnalités de V-REP d'être étendues par des fonctions écrites par l'utilisateur (de la même manière que pour les add-ons). Le langage peut être n'importe quel langage capable de générer une bibliothèque partagée et d'appeler des fonctions C/C++ exportées.
- **Remote API** : (un client API distant) cette méthode permet à une application externe (située par exemple sur un robot, une autre machine, etc.) de se connecter à V-REP en utilisant des commandes API distantes. Il contient environ cent fonctions spécifiques et d'une fonction générique, qui peuvent être appelées à partir d'une application C/C ++, d'un script Python, d'une application Java, d'un programme Matlab / Octave ou d'un script Lua. Les fonctions de l'API distante interagissent avec V-REP via une communication par socket. L'API distante peut permettre à une ou plusieurs applications externes d'interagir avec V-REP de manière synchrone ou asynchrone et même le contrôle à distance du simulateur est pris en charge.
- **ROS Node** (un noeud ROS) : cette méthode permet à une application externe (située par exemple sur un robot, une autre machine, etc.) de se connecter à V-REP via ROS, le Robot Operating Système.
- **BlueZeroNode**(un nœud BlueZero) : cette méthode permet à une application externe (située par exemple sur un robot, une autre machine, etc.) de se connecter à V-REP via BlueZero cite3ref2

3.4.2 La plateforme en simulation :

Notre plateforme simulé en V-rep est constituée d'un convoyeur principale qui transporte les palettes au long de la chaine et de deux post d'assemblage. Chaque poste est équipé d'un robot

kukaiiwa LBR 7r800 similaire au robot réel qui va exécuter des tâches bien définies, de deux convoyeurs qui contient chacun un type d'objet différent qu'on va le placer sur la palette et des captures de proximité qui détecte la présence d'un objet ou d'une palette sur les convoyeurs.

Le contrôle de cette plateforme se fait à travers le système réel. La détection d'une palette sur le convoyeur principal simulé est programmée de tels sorts qu'elle soit synchronisée avec la détection du capteur réel. Cette détection implique l'arrêt du convoyeur principal réel et simulé à travers un ordre envoyé par l'automate à V-rep. L'automate de poste lit l'identifiant du produit stocké dans le Tag qui est porté par la palette à l'aide du lecteur RF380r et l'envoie à l'application multi-agent pour définir son type. Cette dernière définit la tâche à exécuter dans le poste et l'envoie à l'Automate associé qui va la transférer à V-rep par une communication socket. Ces tâches sont préprogrammées dans les deux robots, réel et celui de la simulation. A la fin de chaque tâche un message de fin de tâche est transmis à l'automate du poste, si les deux postes terminent l'exécution de leur tâche le convoyeur redémarre. Le comportement des différents éléments de la simulation est programmé avec le langage Lua dans un Script Threaded.

Les instructions utilisées pour la réalisation de la simulation :

— **sim.getObjectHandle :** cette fonction récupère le Handle de l'objet a manipulé et nous donne la possibilité de l'appeler dans notre programme.

— **sim.readProximitySensor :** Cette fonction permet de connaitre si le capteur de proximité a détecté un objet ou pas. Elle prend comme argument le Handle de capteur et envoie à sa sortie : 0 s'il n y'a pas de détection, 1 : s'il y a une détection d'un objet, -1 : en cas d'erreur. Dans notre programme si le capteur détecte un objet, on arrête le convoyeur en lui donnant une vitesse nulle. Fig 000

```
psensor,dis,pt,detectOH=sim.readProximitySensor(proximity_sensor)

    if(psensor > 0) then
        beltVelocity=0
        sim.setIntegerSignal('b',detectOH)

     end
```

FIGURE 3.12 – Programme Lua du capteur de proximité.

— **sim.rmlMoveToJointPositions :** sert à Déplacer (actionner) plusieurs articulations en même temps à l'aide de Reflexxes Motion Library type II ou IV. Elle nous a permis de programmer les tâches du robot. Cette fonction prend en argument : un tableau des Handles, des articulations à manipuler, un tableau de position des articulations souhaitées, la vitesse et le sens de rotation souhaitée des articulations. Elle donne à sa sortie la nouvelle position, vitesse et accélération de chaque articulation manipulée. Chaque tâche du robot est constituée de plusieurs positions : position de prise, position de dépôt, etc.

```
if(var ==10)then

if(signal1>0) then
sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,Prise1P1,targetVel)
sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,Prise1P2,targetVel)
sim.wait(5)
sim.setIntegerSignal('c',1)
sim.setObjectParent(cube[j],connector,false)
sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,Prise1P1,targetVel)
sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,initialeprise,targetVel)
sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,InitialeDepot,targetVel)
sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,Depot1P1,targetVel)
sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,Depot1P2,targetVel)
sim.setIntegerSignal('c',0)
sim.setObjectParent(cube[j],-1,false)
j=j+1
sim.wait(5)
sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,Depot1P1,targetVel)
sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,InitialeDepot,targetVel)
end
client:send("b")
end
```

FIGURE 3.13 – Exemple de programme d'une tâche Robot

— **sim.setIntegerSignal :** permet de définir la valeur d'un signal entier et créer ce signal s'il n'existe pas. Les signaux crées dans le Main Script ou dans le Child Script sont automatiquement effacés à la fin de la simulation. Elle prend en argument le nom de signal et la valeur à lui ajouter et retourne -1 s'il y a une erreur dans l'exécution. Nous avons utilisé cette fonction pour modifier une valeur dans le Script de GRIPPER de robot qui permet son ouverture ou sa fermeture depuis le Script principal du robot.

— **sim.getIntegerSignal :** cette fonction permet de récupérer la valeur d'un signal entier.

Elle prend en argument le nom de signal et retourne la valeur de signal. Nous l'avons utilisé dans le Script de GRIPPER pour récupérer la valeur de signal.

— **Sim.wait :** cette fonction s'exécute que dans un Threaded Child Script fournit un retard en temps bien précis pour passer à la prochaine étape du programme. Ce dernier peut être exécuté soit en temps de simulation ou en temps réelle.

— **sim.setObjectParent :** permet de définir l"objet parent d'un objet. Elle prend en argument le Handle de l'objet parent, le Handle de l'objet qui sera associé à lui, et False ou True pour indiquer si la position et l'orientation de l'objet associé doivent rester identiques ou pas.

3.4.3 La communication Socket entre l'automate est V-rep :

Pour recevoir les tâches à exécuter du l'automate, nous avons établi une communication par Socket en suivant les étapes indiquées précédemment.

- **Etape 1 : création d'une Socket côté client :**

  Dans cette étape nous avons défini V-rep comme un client, la création du Socket se fait à l'aide des fonctions suivantes :

```
local socket=require("socket")
local client=socket.tcp()
```

Après la création de la Socket, on associe l'adresse IP et le Port sur lesquels elle va communiquer avec le serveur :

```
sim.setThreadIsFree(true)
local result=client:connect('172.31.1.100',3000)
sim.setThreadIsFree(false)
```

Pour que V-rep ne soit pas figé jusqu'au retour des commandes externes de la Socket, on doit déclarer une section non bloquante en appelant la commande sim.setThreadIsFree(true). À l'intérieur de cette section on lance la connexion avec le serveur ensuite on ferme la section à l'aide de la fonction sim.setThreadIsFree (false).

- **Etape 2 : création d'une Socket coté serveur :**

  La socket coté serveur est constitué de deux blocs :Tsend_C envoie la tâche au client et Trcv_C reçoit un message qui indique la fin de l'exécution de la tâche. On indique sur les deux blocs le port sur lequel la communication se fait.

- **Etape 3 : la communication client-serveur :**

Après la création des Sockets cotés serveur et client, le serveur attend une connexion cliente dans le port spécifié, le client demande au serveur s'il accepte la connexion. Ce dernier accepte la connexion, un échange des données s'effectue entre eux. Le serveur (automate) envoie la tâche au client (V-rep) qui va l'exécuter et lui répond lorsqu'il termine l'exécution de la tâche par un message. V-rep ferme la connexion à la fin.

```
if (result==1) then
      print("client connected")
      -- We could connect to the server
   while (sim.getSimulationState()~=sim.simulation_advancing_abouttostop) do

            -- Read the reply from the server:
             sim.setThreadIsFree(true)
              local returnData=client:receive(2)

              if (returnData==nil) then
                   break -- Read error
              end
              if (returnData~=nil) then

                print("socket",returnData)
                var=tonumber(returnData)
                print("socket",var)

              end
              sim.setThreadIsFree(false)

if(var ==10)then
if(var==01)then
if(var ==11)then
if(var ==00)then
end
end

client:close()
end
end
```

FIGURE 3.14 – Programme de lecture/écriture en V-rep

Chapitre 4

Chapitre 4

Pilotage de la plateforme par un système multi-agente

L'usine de future se traduit par la réalisation d'un système de production flexible, capable d'utiliser au mieux les ressources de son système opérant, afin de réagir le plus rapidement possible à la demande et d'offrir des produits personnalisés. Ce système adapte la notion du numérique et introduit les nouvelles technologies qui permette d'obtenir des informations pertinentes et nécessaires, au bon moment pour la réalisation d'un ordonnancement optimale en temps réel.

L'adaptation de ce concept nécessite le développement de nouvelles structures de pilotage (conception, planification et contrôle) des systèmes manufacturiers. Parmi celle-ci, on trouve la structure de pilotage hybride qui est une structure distribuée supervisée du problème avec ordonnancement prévisionnel partiel. La résolution distribuée de problèmes sert à appliquer les principes de l'Intelligence Artificielle Distribuée (I.A.D) dans le but de faire interagir un certain nombre d'entités (agents) qui partagent leurs connaissances et leur savoir-faire afin de résoudre un problème général décomposé en un sous-ensemble de tâches. L'interaction entre ces agents définit le concept d'un système multi-agent.

Dans ce chapitre, on présente l'agent et son rôle en tant qu'entité indépendante et son interaction dans les systèmes multi-agents pour par la suite, introduire ces notions dans la réalisation d'une application qui va contrôler des composants de notre plateforme.

4.1 Agent

4.1.1 Définition d'un agent

Un agent est un système informatique capable d'agir de manière autonome qui peut inclure d'autres agents afin de répondre à ses objectifs de conception. La communauté d'agents en interaction constitue un système multi-agents.

Un agent existe obligatoirement dans un environnement. L'environnement est une structure dans laquelle l'agent évolue et agit et en échange, l'environnement doit réagir à l'action de ces derniers (Interaction entre agent et son environnement). [26]

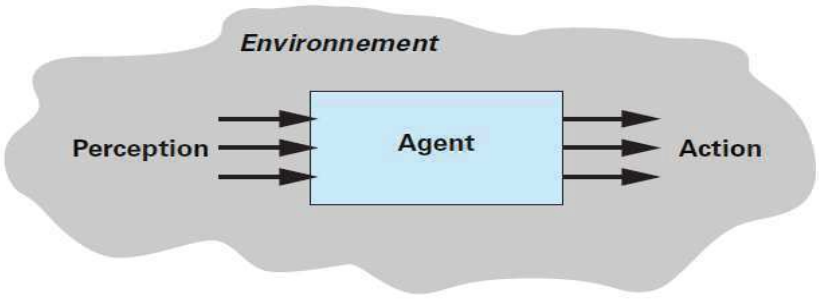

FIGURE 4.1 – représentation d'un agent dans son environnement

En d'autres termes, un agent est une entité physique ou abstraite capable de contrôler partiellement son environnement en ce qu'il peut l'influencer, pouvant communiquer avec d'autres agents et dont le comportement est la conséquence de ses observations, de sa connaissance et des interactions avec les autres agents. [26]

4.1.2 Caractéristique d'un agent

Les agents possèdent plusieurs caractéristiques qui sont :

— Flexible : l'agent doit être capable de réagir dans son environnement selon les ressources qu'il reçoit en entrées.

— Autonome : l'agent est capable de prendre des décisions sur son comportement en se basant sur ses connaissances sans l'intervention d'un tiers

— Proactif : l'agent n'agit pas uniquement en réponse à son environnement mais, il est également capable de prendre l'initiative au "bon" moment. Donc, génère ses buts, prend des initiatives pour satisfaire ses buts.

— Réactif (Capable de répondre à temps) : l'agent doit être capable de percevoir les changements dans son environnement et d'élaborer une réponse dans le temps requis

— Social : l'agent doit être capable d'interagir avec d'autres agents (logiciels ou humains) afin d'accomplir des tâches ou aider ces agents à accomplir les leurs. [27]

4.1.3 Différence entre un agent et un Objet

Plusieurs chercheurs définissent la notion d'agent par rapport à celle d'objet, arguant du fait que la maîtrise d'une nouvelle technologie ou d'un nouveau concept est facilitée par une référence à des technologies ou concepts voisins déjà maîtrisés.Bien qu'il existe plusieurs similitudes entre aux, il n'en existe pas moins d'importantes différences parmi lequel nous allons citer :

— Les agents sont autonomes alors que les objets ne le sont pas ; un agent va décider par son propre processus de décision s'il exécute ou non une action requise ;

— Contrairement aux objets, Les agents agissent d'une manière proactive pour atteindre leurs buts (par exemple, ils saisissent des opportunités).

— Les agents sont capables d'un comportement social : ils peuvent s'engager dans des interactions complexes, par exemple coopération, compétition, négociation, avec d'autres agents ; ce n'est pas le cas des objets .

— Un système multi-agents est, normalement, un système dans lequel les agents correspondent à des chemins d'exécution séparés ; chaque agent a son propre chemin d'exécution alors que les objets, à part les objets concurrents, ne présentent pas cette caractéristique.

— Un objet est défini par l'ensemble des services qu'il offre (ses méthodes) et qu'il ne peut refuser d'exécuter si un autre objet le lui demande. En revanche les agents disposent d'objectifs qui leur donnent une autonomie de décision vis à vis des messages qu'ils reçoivent. D'autre part, ils établissent des interactions complexes qui font intervenir des communications de haut niveau.

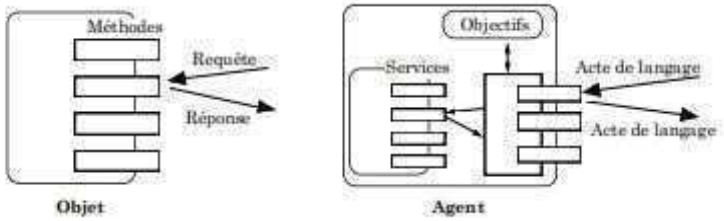

FIGURE 4.2 – Différence entre objet et agent

De ce fait, un agent peut être considéré comme un objet ayant des capacités supplémentaires : recherches de satisfaction (intentions, pulsions) d'une part, et communications à base de langages plus évolués (actes de langages pour les agents cognitifs, propagation de stimuli pour des agents réactifs) d'autre part, ces deux notions étant intrinsèquement liées. Inversement un objet peut être considéré comme un agent « dégénéré » devenu un simple exécutant, tout message étant considéré comme une requête. [27]

4.1.4 Classement des Agents

La classification est un mécanisme important pour comprendre les différents types d'agents. Selon leurs modes de fonctionnement et leurs représentations de leurs environnements, ils peuvent être classés en trois catégories essentielles qui sont : les agents réactifs, lesagents cognitifs et les agents hybrides.

4.1.4.1 Les Agents Ré actifs

Ils sont des composantes très simples qui réagissent directement sur l'environnement perçu. Ils n'ont pas une représentation symbolique de l'environnement ou des connaissances et ils ne possèdent pas de mécanismes d'envoi de messages. Leurs capacités se contentent simplement d'acquérir des perceptions et de réagir à celles-ci qui peuvent être considéré comme une forme de communication. Un SMA constitué d'agents réactifs possède généralement un grand nombre d'agents. [28]

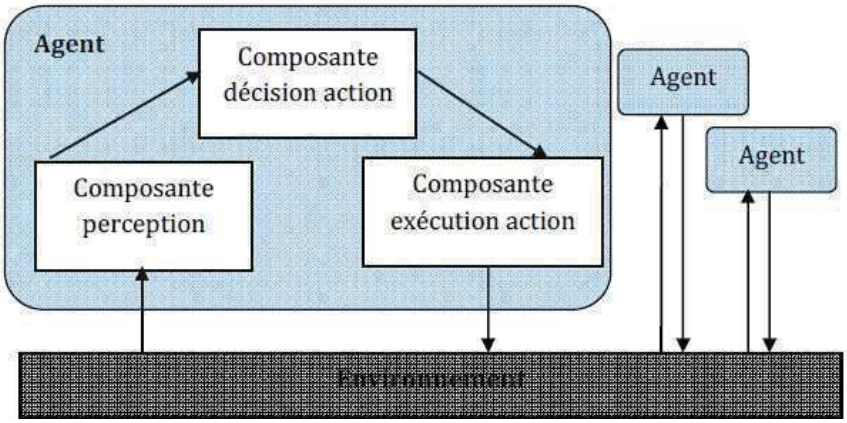

FIGURE 4.3 – Structure générale d'un Agent Réactit

4.1.4.2 Les Agents Cognitifs

Ils sont capables de planifier leur comportement, mémoriser leurs actions passées, communiquer par envoi de messages, négocier, etc. Un SMA constitué d'agents cognitifs possède communément peu d'agents.Ces agents sont capables à eux seuls d'exécuter desopérations complexes, ils peuvent raisonner en s'appuyant sur des bases de connaissances. [28]

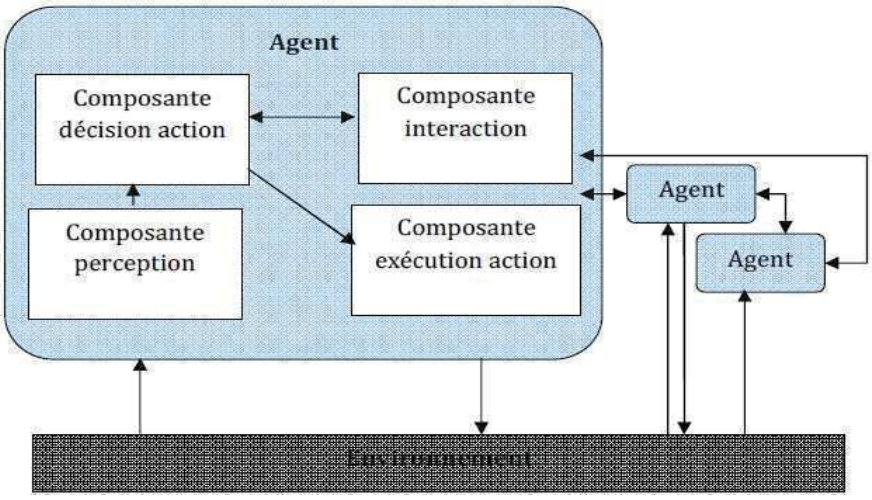

FIGURE 4.4 – Structure générale d'un Agent Cognitifs

4.1.4.3 Les Agents Hybrides

Il est constitué de la combinaison des deux types d'agents mentionnés précédemment, Chaque agent hybride est caractérisé par la notion de couches et chaque couche représente ou les agents cognitifs, ou les agents réactifs. Un agent hybride est composé de plusieurs couches arrangées selon une hiérarchie [27].

4.2 Les Systèmes Multi-Agents

4.2.1 Définition

Un Système Multi-Agent est un ensemble d'agents évoluant dans un environnement commun et interagissant entre eux selon une certaine organisation. Ou encore, c'est un ensemble d'entités qui coordonnent leurs connaissances, buts, expériences et plans pour agir ou résoudre des problèmes, incluant le problème de la coordination inter-agent lui-même. Les agents composent un système, car ils constituent un ensemble cohérent autour d'un objet commun [29]

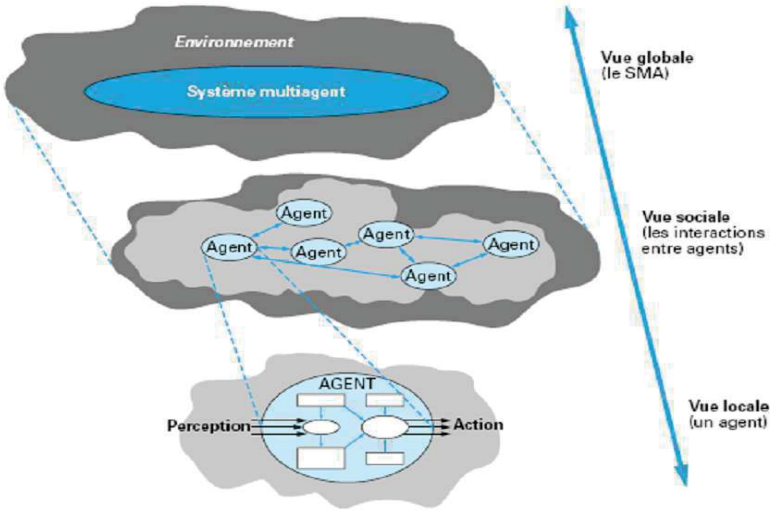

FIGURE 4.5 – Système Multi-Agents

4.2.2 Communication dans les SMA

La mise en œuvre de la communication nécessite un langage de communication compréhensible et commun à tous les agents. Il faut identifier les différents types de communication et définir les moyens permettant non seulement l'envoi et la réception de données mais aussi le transfert de connaissances avec une sémantique appropriée à chaque type de message. Pour expliquer les moyens de communication, on peut définir l'**acte de langage** qui est un moyen mis en œuvre par un locuteur pour agir sur son environnement par ses mots : il cherche à informer, inciter, demander, convaincre, etc. son ou ses interlocuteurs par ce moyen. L'acte de langage désigne donc aussi l'objectif du locuteur au moment où il formule son propos.

De là, on distingue deux modèles de communication qui sont expliqués dans ce qui suit.

4.2.2.1 Communication par partage d'information

Les composants ne sont pas en liaison directe mais communiquent via une structure de données partagée où on trouve les connaissances relatives à la résolution (état courant du problème) qui évolue durant le processus d'exécution. Cette manière de communiquer est l'une des plus utilisées dans la conception des systèmes multi-experts.

4.2.2.2 Communication par envoi de messages

Dans ce type, le mécanisme par transmission de messages suppose une communication directe entre les agents, puisqu'il n'existe pas de modules intermédiaires entre eux, il y'aura un envoi directe et explicite du message au destinataire. Ce mécanisme a été étudié initialement dans le cadre des modèles acteurs. Ce modèle est organisé autour de deux principes élémentaires, à savoir des transmissions de message et un traitement local. Au niveau local, un acteur est constitué de trois composants :

- Une connaissance de son environnement.
- Une connaissance d'un ensemble de noms d'autres acteurs.
- Un ensemble de données.

4.2.3 Plateforme de développement des Systèmes Multi-Agents

La mise en place des Agents nécessite l'utilisation des outils d'élaborations spécifiques. Celles-ci permettent aux développeurs de concevoir et réaliser des applications plus rapidement avec des milliers d'agents qui fonctionnent tous indépendamment en parallèle.

Plusieurs plateformes sont fournies comme logiciels libres, elles sont connues pour avoir été utilisées dans le développement de plusieurs applications comme :MACE, ZEUS, MADKIT, SWORM ou encore JADE, celui qu'on a utilisé dans notre travail.

JADE (Java Agent DEvelopment) est une plateforme de développement de systèmes multi-agents, open-source et basé sur le langage Java.

Chaque instance du JADE est appelée conteneur " container ", et peut contenir plusieurs agents. Un ensemble de conteneurs constituent une plateforme. Chaque plateforme doit contenir un conteneur spécial appelé main-container et tous les autres conteneurs s'enregistrent auprès de celui-là dès leur lancement.
Un main-container se distingue des autres " simples " conteneurs par une autre chose, il contient toujours deux agents spéciaux appelés AMS et DF qui sont lancés automatiquement au lancement du main-container : [30]

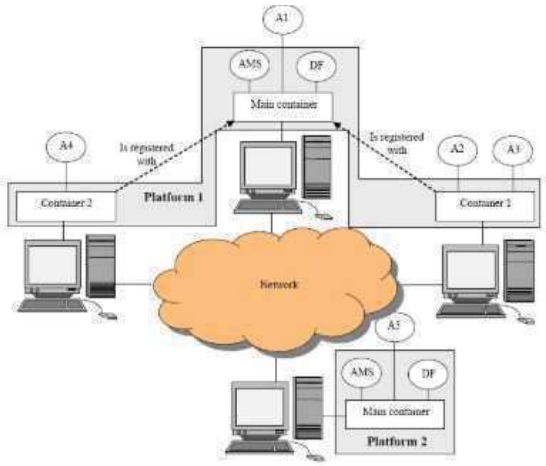

FIGURE 4.6 – Représentation d'un SMA par JADE

— **AMS (Agent Management System)** fournit le service de nommage (pour assurer par exemple que chaque agent possède un identifiant unique dans la plateforme) et repré-

sente l'autorité de la plateforme (par exemple il est possible de créer/arrêter des agents en envoyant des requêtes à l'AMS)

— **DF (Directory Facilitator)**fournit un système de pages jaunes qui permet aux agents de retrouver les agents fournisseurs de services

— **ACC (Agent Communication Channel)** gère la communication entre les agents.

4.2.4 Domaine d'utilisation des Systèmes multi-Agents

L'utilisation des systèmes Multi-Agents varie d'un domaine à un autre, mais elle tournegénéralement autour de trois axes fondamentaux :

- Un système d'agents intelligents dédié à résoudre un certain nombre de problèmes d'une manière distribuée, où chaque agent possède un contrôle total sur son comportement. Pour résoudre un problème complexe, il est plus simple de concevoir des programmes relativement petits (les agents) en interaction.
- Simulations des interactions existantes entre agents autonomes. Le but c'est la mise en œuvre d'un système constitué des agents qui simulent des actions physiques afin de prévoir l'organisation finale. Ce qui importe c'est le comportement d'ensemble et non pas le comportement individuel. L'autonomie permet ici de simuler le comportement exact d'une entité.
- Conception d'un programme constitué d'un ensemble d'agents en interactions afin de réaliser un ensemble de tâches et d'actions réel représentant le but du système. Ce qui est le cas de notre projet.

4.2.5 Comportement d'un Agent

Chaque agent en jade exécute une ou plusieurs tâches qui définissent son comportement (behaviours). Ces tâches sont des instances de la classe **jade.core.Behaviour**. Pour qu'un agent exécute une tâche, on doit lui l'attribuer la méthode **addBehaviour(Behaviourb)** de la classe **jade.core.Agent**.

Chaque Behaviour doit comporter au moins les deux méthodes : **action()** où sont définit les opérations à exécuter par le behaviour et done() qui intervient si le Behaviour a terminé son exécution. Il existe plusieurs types de behaviour : simple, planifié et complexe. [29]

4.2.5.1 simple Behaviour

JADE offre trois types de Behaviours simple :

- One-Shot Behaviour qui a la particularité d'exécuter sa tâche une et une seule fois puis il se termine. Il implémente la méthode **done()**, elle retourne toujours **true** ce qui signifie la fin de la tâche.
- Cyclic Behaviour qui exécute sa tâche d'une manière répétitive. Il implémente la méthode **done(), elle** retourne toujours **false** ce qui permet d'exécuter la tâche plusieurs fois.
- Generic Behaviour qui a un comportement entre le One-shot Behaviour et le Cyclic Behaviour de faite qu'il n'implémente pas la méthode done() et laisse son implémentation au programmeur, donc il peut planifier la terminaison de son Behaviour selon ces besoin.

```
// l'ajout d'un one-shot behaviour pour afficher un Hello world :D
  addBehaviour(new OneShotBehaviour(this){
                public void action(){
                    System.out.println("Bonjour tous le monde je suis l'agent "+getLocalName());
          }
  });

'ajout d'un  CyclicBehaviour pour afficher un message à chaque fois qu'il s'execute
        addBehaviour(new CyclicBehaviour(this) {
                public void action() {
                        System.out.println("cyclique... ");
```

FIGURE 4.7 – Exemple de création d'un One-Shot et Cyclic Behaviours

4.2.5.2 les behaviours planifiés

Pour planifier une tâche d'un agent, JADE offre deux types de Behaviours :

- Waker Behaviour qui est implémenté de façon à exécuter la méthode onWake() après une période de temps passée comme argument au constructeur. Cette période est exprimée en millisecondes. Le Behaviour prend fin juste après avoir exécuté la méthode onWake()
- Ticker behaviour qui est implémenté pour qu'il exécute sa tâche périodiquement par la méthode onTick(). La durée de la période de l éxiicution est passée comme argument au constructeur.

4.2.5.3 les Behaviors composées

Pour exécuter des tâches complexes, JADE offre aussi un ensemble des Behaviours composés qui servent à les présentern, Les sous-Behaviours sont ajoutés au Behaviour composé par la méthode addSubBehaviour().On a 4 type des behaviours composées :

- Behaviour séquentiel : Le behaviour séquentiel est composé de plusieurs sous-behaviour, il commence par exécuter le premier sous-Behaviour et lorsque celui-là termine son exécution, il passe au prochain Behaviour, jusqu'à la fin d'exécution de tous les sous-behaviour.
- Behaviour parallèle :

  Il permet d'exécuter plusieurs Behaviours en parallèle c'est-à-dire en même temps sans attendre que le précédent termine son exécution. Si on veut que le parallelBehaviour termine dès qu'un de ses sous-Behaviours termine alors on doit passer à son constructeur l'argument WHEN_ANY. (Pour attendre la fin de tous les sous-Behaviours on doit lui passer l'argument WHEN_ALL).

```
ParallelBehaviour comportementparallele = new ParallelBehaviour(ParallelBehaviour.WHEN_ANY)
comportementparallele.addSubBehaviour(new WakerBehaviour(this,1000){
        @Override
        protected void handleElapsedTimeout() {
                System.out.println("le temps est écoulé..");
                myAgent.doDelete();
        }
});
comportementparallele.addSubBehaviour(new TickerBehaviour(this,50){

int aleatoire;
@Override
protected void onTick() {
        aleatoire=(int) (Math.random()*100);
        System.out.println("aleatoire =  "+aleatoire);
        if(aleatoire==5)
        {
        System.out.println("Bingo!");
        myAgent.doDelete();
        }
}

addBehaviour(comportementparallele);
```

FIGURE 4.8 – Exemple d'un programme parallèle behaviour

4.3 Conception Multi-agent pour le pilotage de la plateforme réelle

Le principe du pilotage produit se base sur un mode de pilotage distribué en tenant compte du rôle du produit en tant qu'entité actif pour l'échange de l'information entre les systèmes et la prise de décision.

Les systèmes multi-agents est une approche idéale pour représenter des problèmes contrainte à la prise de décisions. Le produit, peut être rendu « intelligent » pour devenir le conducteur de sa propre fabrication. En d'autres termes, le produit devient capable de contrôler son évolution, de dire dans quel état il se trouve et de collaborer avec son environnement.

Ainsi le pilotage par produit est une approche qui est basé sur l'association d'un produit physique à un agent qui est son image dans le système informatique de pilotage.

4.3.1 Structure du système Multi-agent

La structure du système multi-agent se compose d'un conteneur principal ou tous les agents sont définis. A l'aide de la plateforme JADE, on crée des agents JADE comme des classes qui héritent de la classe Agent et qui fait appel à des méthodes qui définissent le cycle de vie de l'agent dans la Plateforme.
L'application Multi-agentse compose de :

— Deux agents postes représentant les postes physiques et en communication directe avec eux

— Cinq agents produit dont chacun contient les informations sur le contenu du produit réel associé à l'agent et communiquent seulement avec les deux agents postes au temps voulu.

4.3.1.1 La mise en place des agents

4.3.1.1.1 Agents Postes : Pour chaque poste est associé un agent. Celui-ci communique directement avec l'objet de commande principale du poste qui est l'automate par une communication socket. Il reçoit par ce dernier l'information sur l'identifiant de produit (par lecture du tag RFID) qui s'est présenté au poste, après avoir traité cette information en sollicitant l'agent produit concerné, il renvoi à l'automate l'ordre de tâche qui doit être exécuter par les différents objets associé au poste (robot, convoyeur et simulation).

Donc la structure générale du programme agent des deux postes se compose de ces parties principales :

- **La communication entre l'agent et l'automate :**

  La communication socket établi entre l'agent et l'automate permet l'échange des informations importantes pour le fonctionnement général.

```
try {
    s = new Socket("172.31.1.140", 1234);
} catch (IOException ex) {
    Logger.getLogger(post1.class.getName()).log(Level.SEVERE, null, ex);
}
```

FIGURE 4.9 – Etablir la communication entre l'automate et l'agent poste

Il est à noter que l'agent poste est programmer en tant que client et reçoit comme entrée l'adresse IP de l'automate et le port de communication.

— Adresse IP :'172.31.1.140' et port : 1234 pour le poste 1

— Adresse IP :'172.31.1.150' et port : 2345 pour le poste 2

```
 DataInputStream dis = new DataInputStream(s.getInputStream());
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
```

FIGURE 4.10 – Initialisation lecture/écriture

— *DataInputStream* est une classe qui permet de lire des lignes de texte et des types de données primitifs Java. Elle contient la méthode *Read* utilisée pour la réception de l'information.

— *DataOutputStream* est une classe qui permet d'écrire des types de données primitifs Java ; nombre de ses méthodes écrivent un seul type primitif Java dans le flux de sortie. La méthode *writeInt* est celle utilisé dans notre cas.

```
int str = dis.read();
while(str==0){
    str = dis.read();

if (str!=0)break;
}
```

FIGURE 4.11 – Lecture de l'information reçu

```
i = Integer.parseInt(Recu.getContent());

dout.writeInt(i);
```

FIGURE 4.12 – Envois de l'information une fois traité.

- **Traitement de l'information reçu par l'automate et communication avec les agents produits :**
  Une fois l'identifiant du produit est reçu, l'agent poste va sélectionner l'agent produit qui correspond à cet identifiant et il établit une communication ACL avec ce dernier.

```
    System.out.println("l'identifiant de l'agent produit est " + str);
myGui.showMessage("l'identifiant de l'agent produit est " + str, true);

                if (str == 1) {
                    iden = "AGENTPRODUIT1";

                }
                if (str == 2) {
                    iden = "AGENTPRODUIT2";
                }
                if (str == 3) {
                    iden = "AGENTPRODUIT3";
                }
                if (str == 4) {
                    iden = "AGENTPRODUIT4";
                }
                if (str == 5) {
                    iden = "AGENTPRODUIT5";
                }
```

FIGURE 4.13 – La sélection de l'agent produit

La communication ACL assure l'échange d'information interne entre ses deux agents, l'agent Poste envoi à l'agent produit concerné un message lui demandant la tâche à exécuter et attend qu'il la reçoit. La réception faite, il renvois cette tâche directement à l'automate du poste concerné.

```
ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);

msg.addReceiver(new AID(iden, AID.ISLOCALNAME));

msg.setContent("envois moi la commande du robot");
myGui.showMessage("la demande de la commande à exécuter est envoyée à l'agent produit : " + str, true);

send(msg);
```

FIGURE 4.14 – Communication ACL avec l'agent produit concerné et envois du message

```
// reception message ACL de agent prod
ACLMessage Recu = receive();
while (Recu == null) {
    MessageTemplate msgR = MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
    Recu = receive(msgR);

    if (Recu != null) {
        System.out.println("commande reçue : " + Recu.getContent());
        myGui.showMessage("commande reçue: " + Recu.getContent(), true);
```

FIGURE 4.15 – Réception du message ALC de l'agent produit

4.3.1.1.2 Agents Produits L'architecture mit en place présente le produit comme une entité autonome qui participe à une prise de décision, concernant le fonctionnement de ce qui compose la partie opérationnelle du système (Robot, Capteurs, simulation et convoyeur). En effet l'assemblage de chaque produit nécessite un comportement particulier de ces machinesqui reçoivent l'information de l'automate en communication avec son agent poste à chaque appel. Pour chaque type de produit correspond un agent et l'information sur l'état d'un produit est requise de l'agent produit concerné par l'agent poste, par l'initiative des messages ACL.

Le programme de un agent produit se décompose en ses parties :

- **Communication de l'agent poste :**

L'agent produit doit identifier avec quel agent poste est-il en contacte pour envoyer l'information, car chaque poste reçoit une information sur le comportement de ses composants selon la composition du produit.

L'agent produit reconnait l'agent poste par le type d'envoi du message :

— L'agent poste 1 envoie un message de type REQUEST.

— L'agent poste 2 envoie un message de type INFORM.

```
final MessageTemplate msgTemplat = MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
final ACLMessage messagRecu = receive(msgTemplat);

if (messagRecu != null) {

    myGui.showMessage("réception d'une commande de  l'agentPOST1: " + messagRecu.getContent(),true);
    System.out.println("réception d'une commande de  l'agentPOST1: " + messagRecu.getContent());
```

FIGURE 4.16 – Communication Message ALC dans un agent produit

Une fois la communication faite entre l'agent poste et l'agent produit, ce dernier reçoit un message et renvois la commande qui permet d'exécuter la tâche au même agent poste.

```
aclMessage2 = messagRecu.createReply();
   aclMessage2.setPerformative(ACLMessage.REQUEST);
    aclMessage2.setContent("1");

     System.out.println("commande envoyée à l'AgentPost 1:" +aclMessage2.getContent());
    myGui.showMessage("commande envoyée à l'AgentPost 1:" +aclMessage2.getContent(),true);
     send(aclMessage2);
}else block(); }}
```

FIGURE 4.17 – Envoi de la commande

➤ **Remarque :**
Les agents postes et produits adoptent un comportement cyclique pour assurer le fonctionnement continu du système.

```
ParallelBehaviour parallelBehaviour=new ParallelBehaviour();
addBehaviour(parallelBehaviour);
parallelBehaviour.addSubBehaviour(new CyclicBehaviour(this) {
    @Override
    public void action() {

        final MessageTemplate msgTemplat = MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
        final ACLMessage messagRecu = receive(msgTemplat);

        if (messagRecu != null) {

            myGui.showMessage("réception d'une commande de  l'agentPOST1: " + messagRecu.getContent(),true);
            System.out.println("réception d'une commande de  l'agentPOST1: " + messagRecu.getContent());


          ACLMessage aclMessage2;

        aclMessage2 = messagRecu.createReply();
           aclMessage2.setPerformative(ACLMessage.REQUEST);
            aclMessage2.setContent("1");

              System.out.println("commande envoyée à l'AgentPost 1:" +aclMessage2.getContent());
             myGui.showMessage("commande envoyée à l'AgentPost 1:" +aclMessage2.getContent(),true);
              send(aclMessage2);
        }else block(); }}
);
```

FIGURE 4.18 – Exemple de comportement cyclique dans un agent

4.3.1.2 La supervision des Agents

Afin de pouvoir suivre le fonctionnement des agents (que ce soit les agents produits ou les agents postes). Nous avons associé à chacun d'entre eux une fenêtre qui affiche des messages au cours de son comportement.

FIGURE 4.19 – Exemple de l'affiche de l'agent poste 1

Notons que tous ces comportements d'agents sont développés et implémentés via la plateforme informatique JADE,ainsi que toutes les communications et interactions des agents qui sont aussi enregistrées.Nous pouvons visualiser ces interactions par la fenêtre de l'agent Sniffer,

où nous mettons en évidence tous les agents de notre prototype implémentés dans la plateforme avec les différentes communications et interactions.

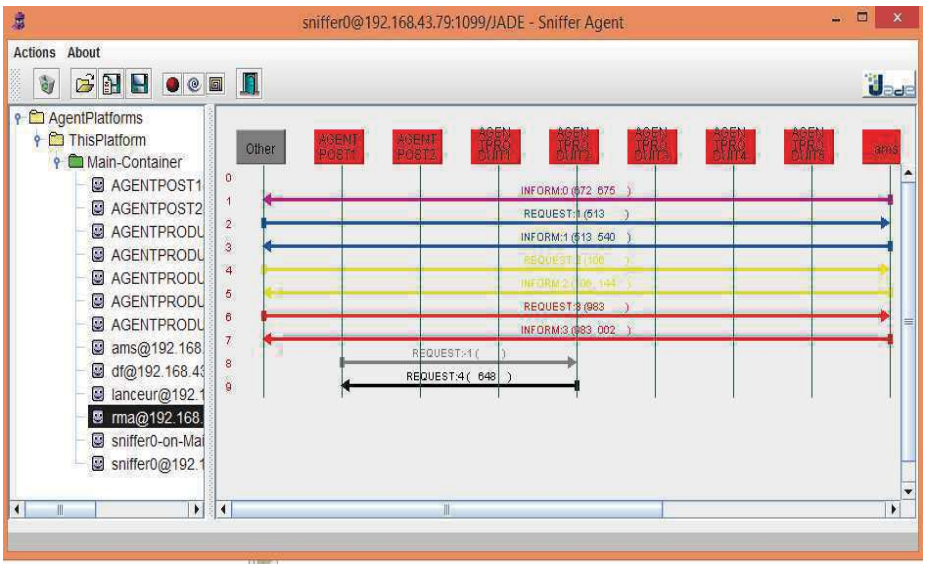

FIGURE 4.20 – Plateforme JADE montrant les intéractions entre les agents

Conclusion Génerale

Dans ce travail, nous nous sommes intéressés aux problèmes de planification et d'ordonnancement d'une chaine flexible d'assemblage robotisé en utilisant la méthode de pilotage par produit, basé sur l'approche multi-agent pour un contrôle en temps réel.

Dans un premier lieu, nous avons présenté les lignes directrices qui constituent un système automatisé de production. Nous avons étudié brièvement quelques structures de pilotage de ce dernier. Cependant, nous avons conclu que la meilleure structure dans notre cas est la structure de pilotage distribué supervisé basé sur un ordonnancement en temps réel. Puis, nous avons décrit le fonctionnement global de notre chaine d'assemblage et les caractéristiques techniques des matériels qui la constitue. Par la suite, nous avons présenté les concepts relatifs à notre étude. Nous avons évoqué la technique RFID comme un élément essentiel pour appliquer le concept relatif à l'industrie 4.0 qui est le contrôle par produit. Nous avons expliqué les programmes que nous avons développés pour le contrôle de chaque dispositif à part et la communication mise en place entre ces derniers en soulignant le rôle de la simulation et le principe Hardware-in-the-loop pour compléter les éléments manquants de la plateforme.

À la fin de ce travail, nous avons décrit une application de l'approche multi-agents au pilotage en temps réel implémenté dans notre système. L'architecture de pilotage choisie est distribuée supervisée. Son fonctionnement est concurrent, intelligent et émergent de l'interaction collective de ses agents. Le contrôle est distribué sur l'ensemble des deux types d'agent, Agent post et Agent produit, chacun à un rôle dans la prise de décision. Enfin, le système intègre une gestion automatique de la configuration globale de la cellule et des agents.

Mais au final, on constate que le système proposé fonctionne globalement, bien qu'il nécessite encore de nombreux efforts de développement. Il permet de mettre en œuvre le concept de pilotage par le produit que nous avons énoncé toute au long de ce mémoire. Cette architecture permet de représenter un fonctionnement hiérarchique, puisque les centres de décision peuvent agir à la fois sur les agents produits et les agents postes. L'accessibilité et la facilité d'utilisation de ce système restent toutefois à améliorer, grâce au développement d'interfaces graphiques, à la simplification des procédures de codage des règles, et en améliorant la stabilité et la robustesse du système multi-agents.

Bibliographie

[1] Sallemi, Anis. Systéme Automatisés de Production. [en ligne]. [consulté le 15 mars 2019]. Disponible sur : http://www.uvt.rnu.tn/resources-uvt/cours/Automates/chap1/co/Module_chap1_5.html.

[2] Le Groupe Logistique Conseil. La planification de la production. [en ligne]. [Consulté le 18 mars 2019]. Disponible sur : http://www.logistiqueconseil.org/Articles/Gestion-production/Planification-production.htm

[3] Erschler J., Fontan G., Mercé C. . Approche par contraintes en planification et ordonnancement de la production. APII, 1993, vol 27, n 6, pp.669-695 .

[4] Kharrat, Samah. Contribution à l'ordonnancement des ateliers de traitement de surface avec deux robots. 191p. Thèse de Doctorat : Automatique .Université de Technologie de Belfort-Montbeliard, 2012.

[5] Djeghaba, Messaoud. Problèmes de décision dans une cellule de production flexible utilisant la coopération entre robots. 122p. Thèse de Doctorat : Automatique. université de Lille Flandres Artois, Lille, France, 1986.

[6] Damien, Trentesaux. Conception d'un système de pilotage distribué, supervisé et multicritère pour les systèmes automatisés de production. 210 p. Thèse de Doctorat : Automatique / Robotique. Institut National Polytechnique de Grenoble - INPG, 1996.

[7] Merbaki. Une approche d'ordonnancement temps-réel basée sur la sélection dynamique de règles de priorité. Thèse de Doctorat .université Claude Bernard Lyon I, Lyon, France, 1995.

[8] Kouiss K., Pierreval H. et Mebarki N.. Towards the use of a multi-agent approach to the dynamicscheduling of flexible manufacturingsystems. International conference on Industrial Engineering and Production Management, Marrakech, Maroc, avril 1995, pp. 118-125.

[9] Arai E., Amnuay S. T. et Uchiyama N.. Real time simulation and monitor forecast for dynamicscheduling in distributed production systems. Proceedings of JSPE-IFIP WG 5.3 Workshop DIISM'93, Japon, 1993, pp. 137-147.

[10] De Smet O., Abou-Kandil H. . Ordonnancement temps-réel pour des systèmes flexibles de production sujets à pannes.revue d'automatique et de productiques appliquées, 1995, vol. 8, n 2-3, pp.291-296.

[11] Trentesaux, Damien. Pilotage hétérarchique des systèmes de production.115p. Habilitation à diriger des recherches : Automatique et Informatique des Systèmes Industriels et Humains. Université de Valenciennes et du Hainaut-Cambrésis, 19 décembre 2002.

[12] Ferber J., Ghallab M. . Problématique des Univers Multi Agents Intelligents. Journées nationales du PRC/IA, Cepadues-Edition, Toulouse, France, mars 1988.

[13] MIRDAMADI, Samieh. Modélisation du processus de pilotage d'un atelier en temps réel à l'aide de la simulation en ligne couplée à l'exécution .174p. Thèse de Doctorat : Systèmes Industriels. l'Institut National Polytechnique de Toulouse, 19 juin 2009.

[14] Gimélec. Industrie 4.0 l'usine du future[en ligne].septembre 2013.[consulté le 24 février 2019].disponible sur : https://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/techniques/1888/1888-gimelec-industrie-4.0-lusine-connectee-septembre-2013.pdf

[15] Kohler Dorothée et Weisz Jean-Daniel. Industrie 4.0 :quelles stratégies numériques ?[en ligne]. novembre 2015. [consulté le 24 février 2019].disponible sur : https://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/techniques/1888/1888-gimelec-industrie-4.0-lusine-connectee-septembre-2013.pdf

[16] SIEMENS. système de périphérique décentralisée ET200SP[en ligne]. juin 2017,[consulté le 06 mars 2019].disponible sur : http://media.automation24.com/manual/fr/58649293_et200sp_system_manual_fr-FR_fr-FR.pdf.

[17] SIEMENS.SIMATIC ET 200SP Open controller.consulté le [06 mars 2019].disponible sur : https://www.industry.siemens.com/topics/global/en/tia/product-innovations/pages/simatic-et200sp-open-controller.aspx

[18] SIEMENS. SIMATIC ET 200SP[en ligne].[consulté le 06 mars 2019]. disponible sur : https://w5.siemens.com/belux/web/fr/industrie/industrie/automatisation/et200sp/et200sp/pages/default.aspx

[19] SIEMENS. ET200SP Open controller[en ligne].juin 2017.[consulté le consulté le 15 mars 2019].disponible sur : https://support.industry.siemens.com/cs/document/109248384/simatic-et-200sp-open-controller-cpu-1515sp-pc-(f)?dti=0&lc=fr-WW.

[20] SIEMENS.Systèmes RFID SIMATIC RF300[en ligne].octobre 2016,[consulté le consulté le 15 mars 2019].disponible sur : https://cache.industry.siemens.com/dl/files/946/21738946/att_900726/v1/SYH_RF300_77_fr-FR.pdf.

[21] SIEMENS.SRFID systems RF180C communication module [en ligne].Décembre 2012,[consulté le consulté le 15 mars 2019].disponible sur : https://cache.industry.siemens.com/dl/files/946/21738946/att_900726/v1/SYH_RF300_77_fr-FR.pdf.

[22] MA KUKA Sunrise Cabinet V5.2[en ligne].[consulté le 18 avril 2019].Disponible sur : http://www.oir.caltech.edu/twiki_oir/pub/Palomar/ZTF/KUKARoboticArmMaterial/MA_KUKA_Sunrise_Cabinet_en.pdf.

[23] krakowiak,sasha.Communication par socket.2006.[consulté le 08 Mai 2019].Univérsité Joseph fourier.

[24] SIEMENS.Industrial Identification with SIMATIC RF300 and RF180C[en ligne].avril 2018.[consulté le consulté le 10 avril 2019].disponible sur : file:///C:/Users/imene/Downloads/109483372_SIMATIC_RF300_RF180C_TO_DOC_V1_1_en.pdf.

[25] coppelia robotics .Virtual Robot Experimentation Platform USER MANUAL[en ligne].2018.[consulté le consulté le 20 avril 2019].disponible sur :http://www.coppeliarobotics.com/helpFiles/

[26] WOOLDRIDGE,michael.an introduction to multi-agent systems . wiley .2012.348p.ISBN 0-471 -49691 -X

[27] MOUSSAOUI MOUHAMED LAMINE. *Système Multi Agent pour planification de la production*[en ligne]. Mémoire :informatique. Université de Djilali BOUNAÂMA Khemis Miliana. disponible dans :`http://dspace.univ-km.dz/xmlui/bitstream/handle/123456789/2010/memoire%20final.pdf?sequence=1&isAllowed=y`

[28] BENOUDINA LAZHAR. *MODÉLISATION ET SIMULATION BASÉES MULTI-AGENTS DU CONTRÔLE DE PROCESSUS INDUSTRIELS*[en ligne]. Mémoire :informatique. Université 20 Aout 1955 Skikda. disponible dans :`http://vrpg.univ-skikda.dz/recherchePG/theses_memoires/fac_sciences/infor/memoire_lazhar.pdf`

[29] FERR,AIDA et BARACHI,FARIDA .*CONCEPTION ET REALISATION D'UN ENVIRONNEMENT D'EMULATION MULTI-AGENT POUR L'EVALUATION DES APPROCHES DE PILOTAGE PAR LE PRODUIT DES SYSTEMES MANUFACTURIERS*[en ligne]. Mémoire :informatique.Université Saad Dahlab.

[30] BENNAL,yougourtha,Créez votre premier agent avec JADE et ECLIPSE[en ligne].2009.[consulté le 19 mai 2019].disponible sur :https ://djug.developpez.com/java/jade/creation-agent/

Annexe A : Grafcet et programme des Automates programmables

Dans la conception d'un système automatisé séquentiel, il faut tenir compte des deux composantes qui le forment. En effet, un automatisme séquentiel ou un système automatisé séquentiel est composé de deux parties complémentaires ; ce sont la partie "opérative" et la partie "commande".

La partie "opérative" est le processus physique à automatiser. Cette partie se compose de trois ensembles qui sont l'unité de production, les actionneurs qui mettent en mouvement la partie mécanique par l'intermédiaire des effecteurs et les capteurs.

La partie "commande" élabore des ordres destinés au processus à automatiser en fonction des informations (comptes rendus) qui lui parviennent de la partie "opérative" et des consignes qu'elle reçoit en entrée. La partie "opérative" est appelée partie puissance. La partie "commande" d'un système automatisé peut être réalisée par l'intermédiaire d'un automate, d'un ordinateur ou par un circuit de logique câblée (séquenceur).

Le Grafcet est le standard qui permet d'expliquer le fonctionnement d'un système automatisé en y incluant toutes les interactions entre la partie "commande" et la partie "opérative". Les étapes à accomplir lors de la réalisation d'un Grafcet vous sont ici présentées.

Pour commander notre système, Nous avons mis en place un Grafcet pour le fonctionnement de chaque poste indépendamment afin que tous les équipements qui appartiennent à ce poste réagissent en temps voulu :

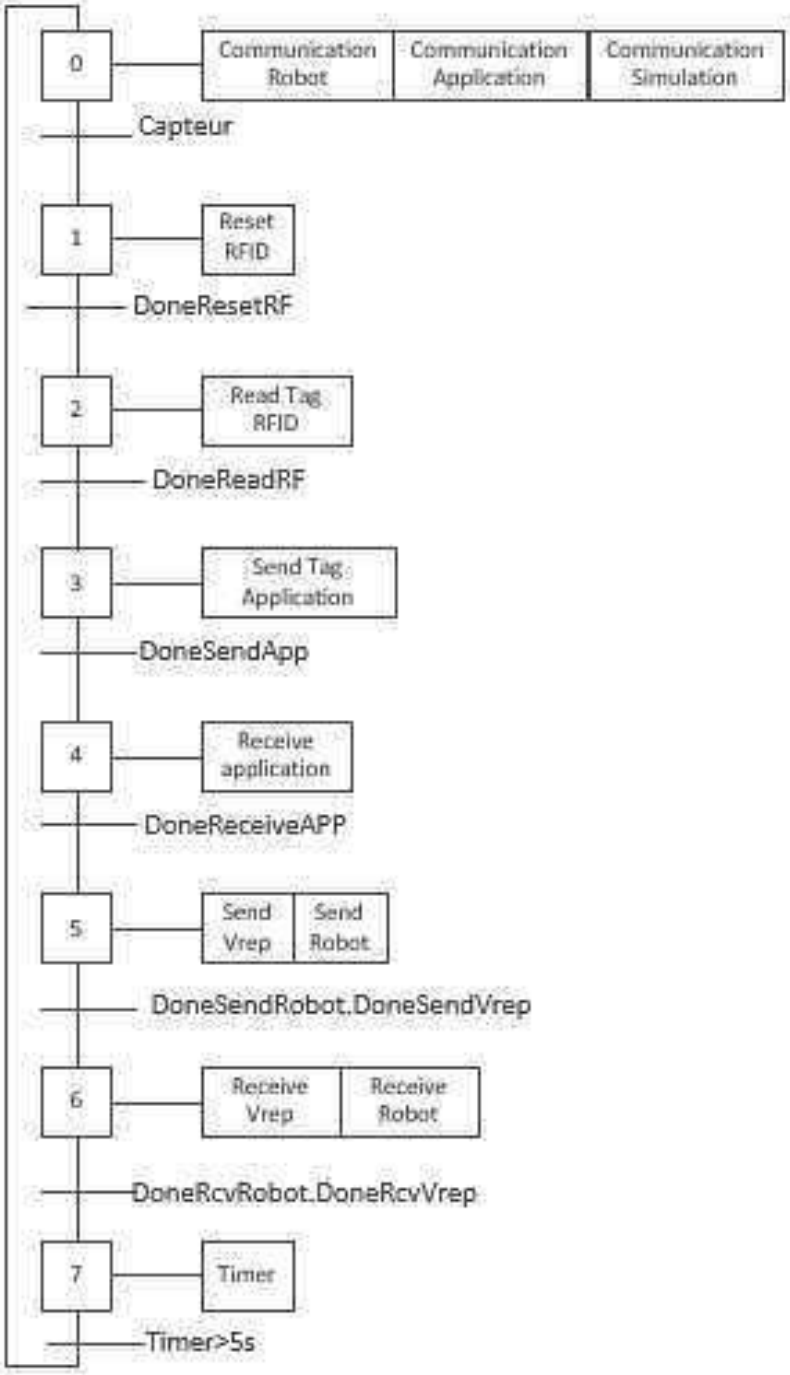

FIGURE 21 – Grafcet du poste 1

A partir de ce grafcet, on extrait les équations suivantes :

$$X_0 \begin{cases} S_O = X_7.Timer + Reset \\ R_0 = X_0.Capteur \end{cases}$$

$$X_1 \begin{cases} S_1 = X_0.Capteur \\ R_1 = X_1.DoneResetRF + Reset \end{cases}$$

$$X_2 \begin{cases} S_2 = X_1.DoneResetRF \\ R_2 = X_2.DoneSendRF + Reset \end{cases}$$

$$X_3 \begin{cases} S_3 = X_2.DoneReadRF \\ R_3 = X_3.DoneSendApp + Reset \end{cases}$$

$$X_4 \begin{cases} S_4 = X_3.DoneSendApp \\ R_4 = X_4.DoneReceiveApp + Reset \end{cases}$$

$$X_5 \begin{cases} S_5 = X_4.DoneReceiveApp \\ R_5 = X_5.DoneSendRobot.DoneSendVrep + Reset \end{cases}$$

$$X_6 \begin{cases} S_6 = X_5.DoneSendRobot.DoneSendVrep \\ R_5 = X_5.DoneRcvRobot.DoneRcvVrep + Reset \end{cases}$$

$$X_7 \begin{cases} S_7 = X_6.DoneRcvRobot.DoneRcvVrep \\ R_5 = X_5.Timer + Reset \end{cases}$$

Remarque :

1. Le grafcet présenté a été appliqué pour les deux postes sauf quelques changements pour le poste 2 à cause du l'absence du robot réelle. Donc, les variables DoneSendRobot, DoneRcvRobot et le bloc de fonction pour la communication robot n'existe pas dans ce cas.
2. Nous avons utilisée à la fin du Grafcet un Timer pour donner le temps à la palette de passer sans que le capteur la détecte une deuxième fois.
3. Le convoyeur est relié directement à l'automate du poste 1 ; mais la logique pour le contrôler a été faite dans les deux postes. Donc, la variable d'entrée du poste1 « Automate2 » a été récupérer de l'automate du poste 2.

Ces équations ont été traduites en ladder, implémenté dans le logiciel TIA Portal et à la fin chargé dans l'automate.
La procédure de programmation c'est faites de la façon suivante :

1. La programmation des équations de déclanchement des états :

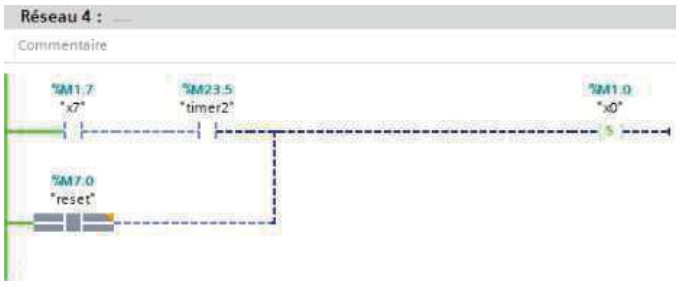

FIGURE 22 – bascule s de l'état X0

FIGURE 23 – bascule s de l'état X1

FIGURE 24 – bascule s de l'état X2

FIGURE 25 – bascule s de l'état X3

FIGURE 26 – bascule s de l'état X4

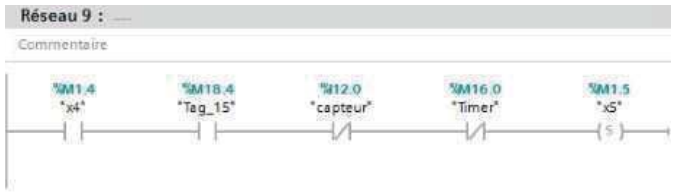

FIGURE 27 – bascule s de l'état X5

FIGURE 28 – bascule s de l'état X6

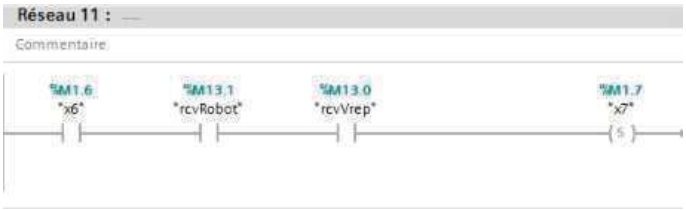

FIGURE 29 – bascule s de l'état X7

2. Viens par la suite les équations de réinitialisation :

FIGURE 30 – bascule R de l'état X0

FIGURE 31 – bascule R de l'état X1

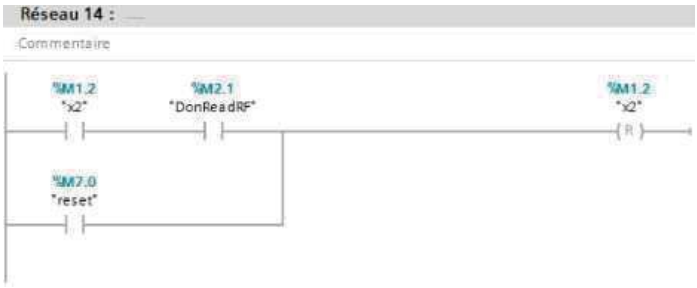

FIGURE 32 – bascule R de l'état X2

FIGURE 33 – bascule R de l'état X3

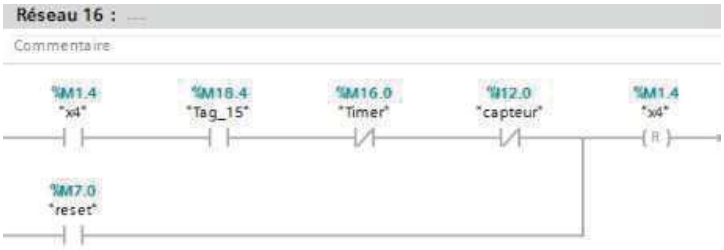

FIGURE 34 – bascule R de l'état X4

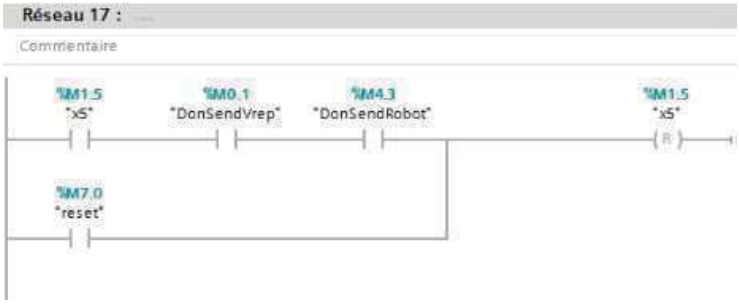

FIGURE 35 – bascule R de l'état X5

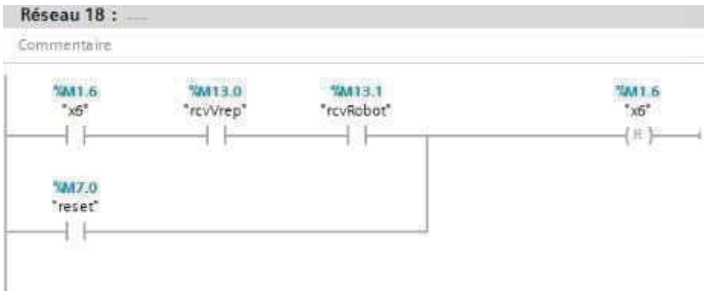

FIGURE 36 – bascule R de l'état X6

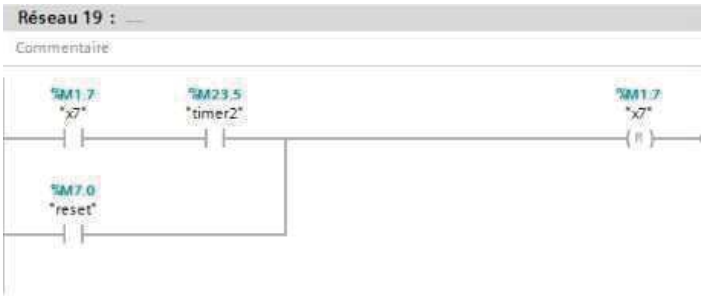

FIGURE 37 – bascule R de l'état X7

3. Le bloc de contient toutes les fonctions pour la lecture RFID :

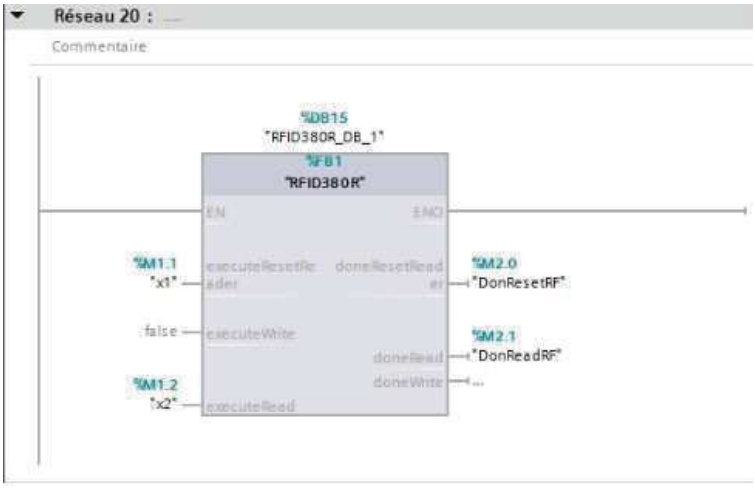

FIGURE 38 – Le bloc de lecture RFID

4. Le bloc qui communique avec le système multi-agent :

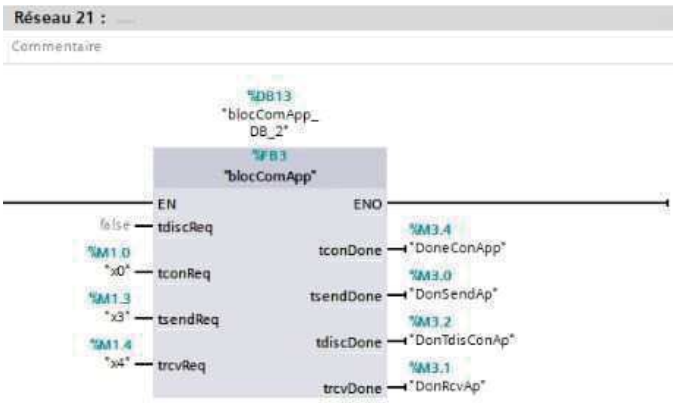

FIGURE 39 – Communication SMA

Qui est accompagné par un bloc de conversion d'un entier (variable récupéré de la SMA) en caractère (pour l'envoyer au Robot) et une chaine de caractère (pour l'envoyer au V-rep)

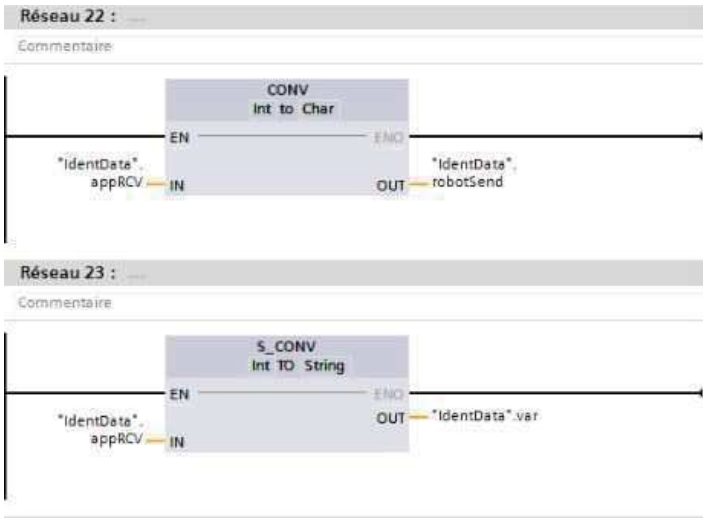

FIGURE 40 – Communication SMA

5. Les blocs de communications avec le robot et le logiciel de simulation V-rep :

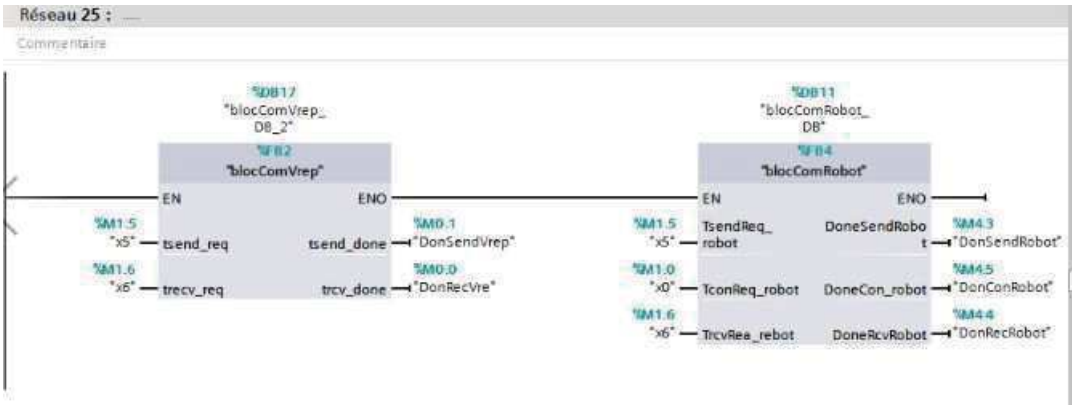

FIGURE 41 – Communication robot et v-rep

6. Le contrôle marche/arrêt du convoyeur pour un seul poste :

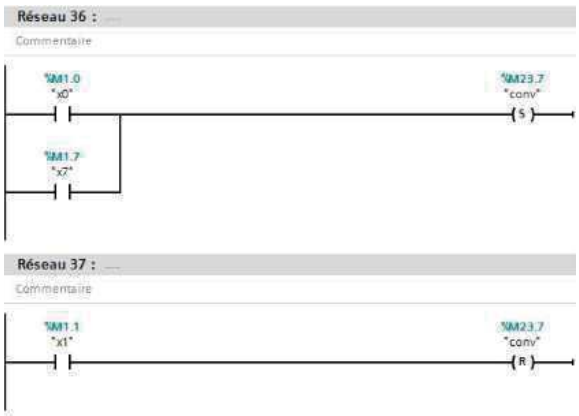

FIGURE 42 – Contrôle du convoyeur

7. Le contrôle marche/arrêt du convoyeur par les deux postes (seulement dans le poste1) :

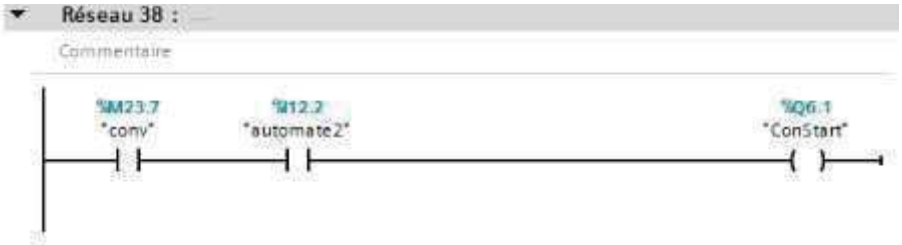

FIGURE 43 – Contrôle du convoyeur par le poste 1

8. Le Timer qui se déclenche par impulsion :

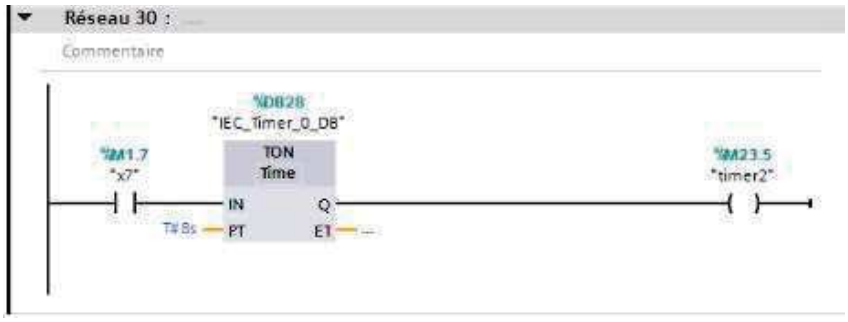

FIGURE 44 – Le Timer

9. Tous les variables peuvent être visualisés dans la table des mnémoniques :

IdentData

	Nom	Type de données	Valeur de départ	Valeur de visualisati.	Rémanence	Accessible ...	Ecritu...	Visible da...
1	Static				☐	☐	☐	☐
2	rcvAutomate	Bool	false	FALSE	☐	☑	☑	☑
3	tsendconv	String	''	' 0'	☐	☑	☐	☑
4	robotSend	Char	' '	'$03'	☐	☑	☑	☑
5	robotRcv	Char	' '	'g'	☐	☑	☑	☑
6	Tagread	Int	0	3	☐	☑	☑	☑
7	startConVrep	Int	0	0	☐	☑	☑	☑
8	tsendStartConVrep	String	''	''	☐	☑	☑	☑
9	appRCV	Int	0	3	☐	☑	☑	☑
10	readData	Array[0..511] of Byte			☐	☑	☑	☑
11	vrepSEND	String	''	''	☐	☑	☑	☑
12	readDataAP	Array[0..1] of Byte			☐	☑	☑	☑
13	var	String	''	' 3'	☐	☑	☑	☑
14	varconv	Int	0	0	☐	☑	☑	☑
15	vrepRCV	Char	' '	'R'	☐	☑	☑	☑
16	writeData	Array[0..511] of Byte			☐	☑	☑	☑

FIGURE 45 – Table des mnémonique du Poste 1

Annexe B : mise en œuvre d'une application multi-agent en utilisant Jade et NetBeans

Un système multi-agent est un système distribué composé d'un ensemble d'agents qui sont capable d'agir d'une façon autonome, de communiquer avec leur environnement dans le but d'atteindre les objectifs pour lesquels ils ont été conçus. Pour chaque agent on doit définir son environnement (espace de vie, ressource, capteur), son organisation (centralisé, décentralisé, hiérarchique) et ces interactions (communication, coordination, etc.)

Il existe plusieurs plateformes de développement Multi-Agent : AnyLogic, CORMAS, JACK, Jadex, JADE, etc. Dans cette annexe, on va présenter les étapes de base pour le lancement d'une application Multi-Agent en utilisant Jade et NetBeans IDE.

NetBeans IDE est un logiciel de développement professionnel open source, adapter aux plusieurs langages de programmation (Javascript, Python, C/C++, java, PHP, etc). Il offre de nombreuses fonctionnalités pratiques : le débogage, l'auto-complétion, etc. Il permit de :

— développer des applications, créer des sites web en utilisant java (des jeux, des animations, etc.)

— développer des applications en glissant et en sélectionnant les items dans l'interface (JLabel, JButton ou JTexteField, etc,).

— Avoir un aperçu du résultat des scripts.

Jade est une Plateforme Multi-Agent crée par le laboratoire TILAB. Il permet de développer des systèmes Multi-Agent et des applications conforme aux normes FIPA. Ces normes établissent les règles normatives qui permettent à une société d'agents d'inter-opérer, ils décrivent le modèle de référence d'une plate-forme multi-agents où ils identifient les rôles de quelques agents clés nécessaires pour la gestion de la plate-forme, et spécifient le contenu du langage de gestion des agents et l'ontologie du langage. Jade possède trois modules importants et nécessaires à la norme FIPA : Directory Facilitator qui fournie une page jaune à la plateforme, Agent Communication Chanel qui gère la communication entre les agents, Agent Management System qui supervise l'enregistrement des agents et supervise leur authentification et leur accès.ces trois module sont activées à chaque à chaque démarrage de la plateforme. Il contient aussi un environnement où les agents peuvent vivre, une librairie de classes et un outil graphique qui facilite la supervision et la gestion de la plateforme.

Une plateforme est un ensemble de conteneurs, dont elle doit contenir un Main container où se trouvent les modules nécessaires de la norme FIPA. Pour lancer une plateforme multi-Agrnt en NetBeans, on doit suivre les étapes suivantes :

- Télécharger Et installer NetBeans
- Télécharger le fiche JADE-all, décompresser le ensuit décompresser les 4 fichiers qui apparaitre.
- Créer une variable Classpath si elle n'existe pas en suivant ces étapes :
  - — par Clic droit sur le poste de travail, choisissez propriétés. La fenêtre propriétés système apparaît, choisissez l'onglet Avancé, puis cliquez sur variables d'environnement. Une petite fenêtre intitulée " variables d'environnement " apparaît.

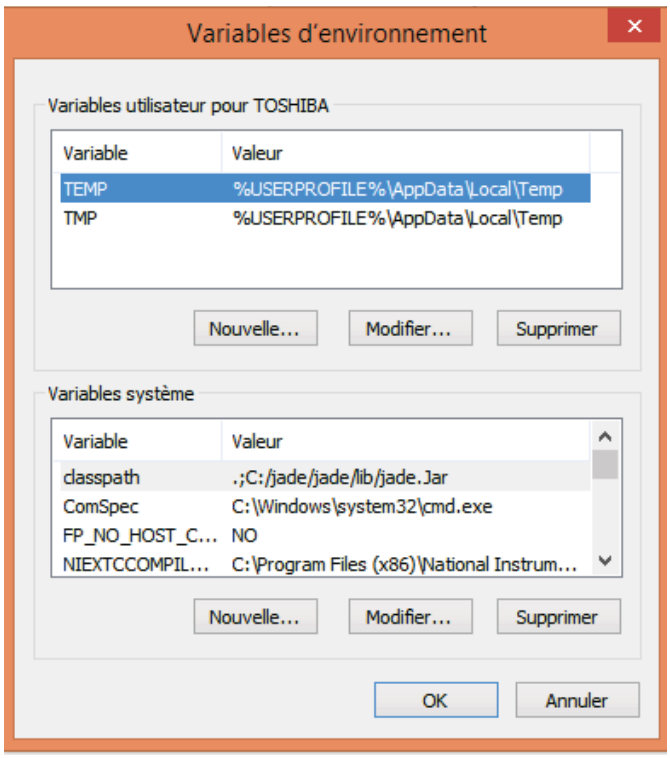

FIGURE 46 – Fenêtre " variables d'environnement.

— Dans la zone variable système, essayez de trouver la variable d'environnement qui porte le nom CLASSPATH. Si vous ne la trouvez pas, il faut la créer, que la variable est trouvée/créée on doit lui attribuer une valeur .cette valeur est la concaténation des chemins des quatre fichiers décompresser précédemment séparer entre elles par un point virgule. A la fin, enregistrez les modifications.

— Pour vérifier que ces étapes sont bien faites, il faut juste ouvrir la fenêtre **Exécuter**, entrer le nom **cmd.** Dans **l'invite de commande** qui apparaitre, tapez la commande suivante : **Java jade.Boot –gui**

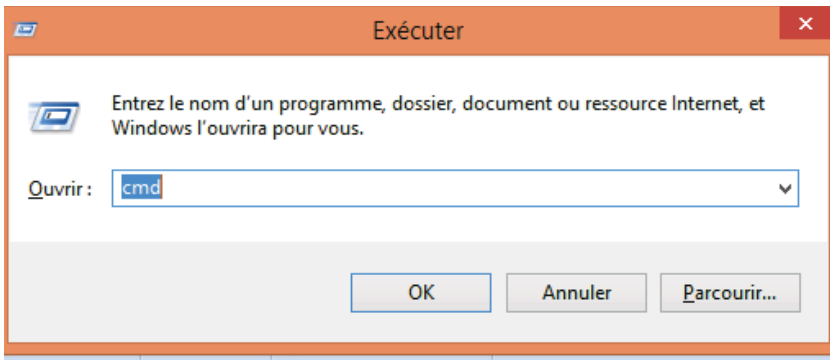

FIGURE 47 – Fenétre " Exécuter.

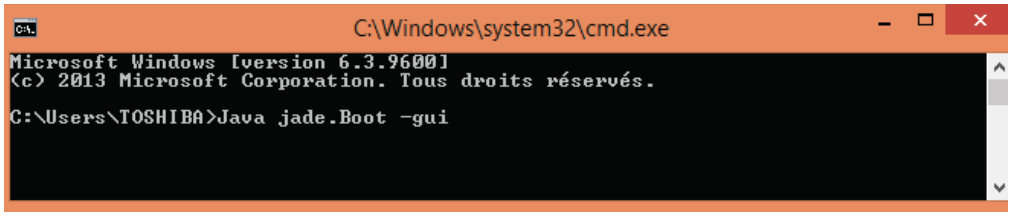

FIGURE 48 – Fenêtre de l'invite de commande.

— Si les étapes précédentes sont bien faites, une fenêtre de lancement de la plateforme Jade va apparaitre.

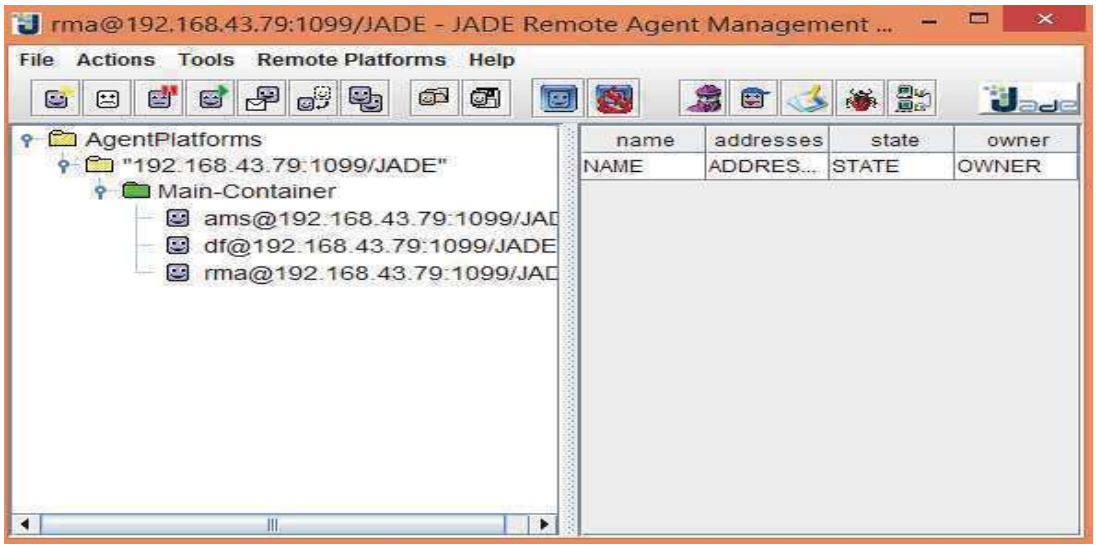

FIGURE 49 – Plateforme Jade.

- créer Main-container avec Jade et Netbeans. Pour cela, vous devez :
  - — Commencer par la création d'un nouveau projet Netbeans, choisissez Java Application, ajoutez un pakage, puis créez une nouvelle classe.

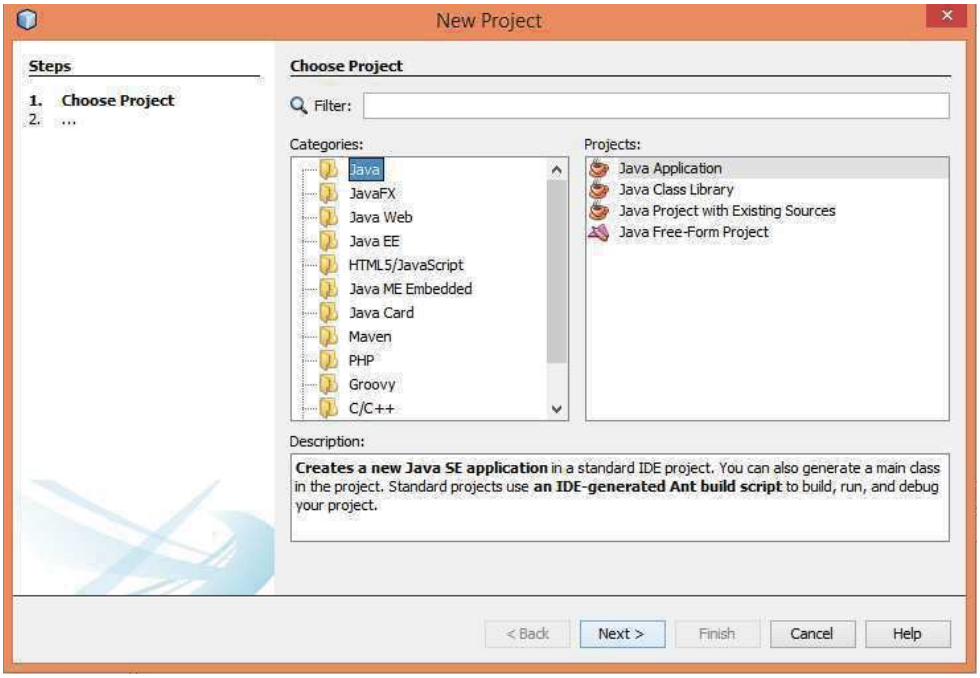

FIGURE 50 – Création d'une application java.

  - — Avant l'écriture du code, vous devez ajouter les librairies à votre projet en effectuant un clic droit sur le nom du projet Puis choisissez propriétés. Cliquez sur java build path >> Libraries>> add external JARs puis ajoutez les trois librairies Jare.

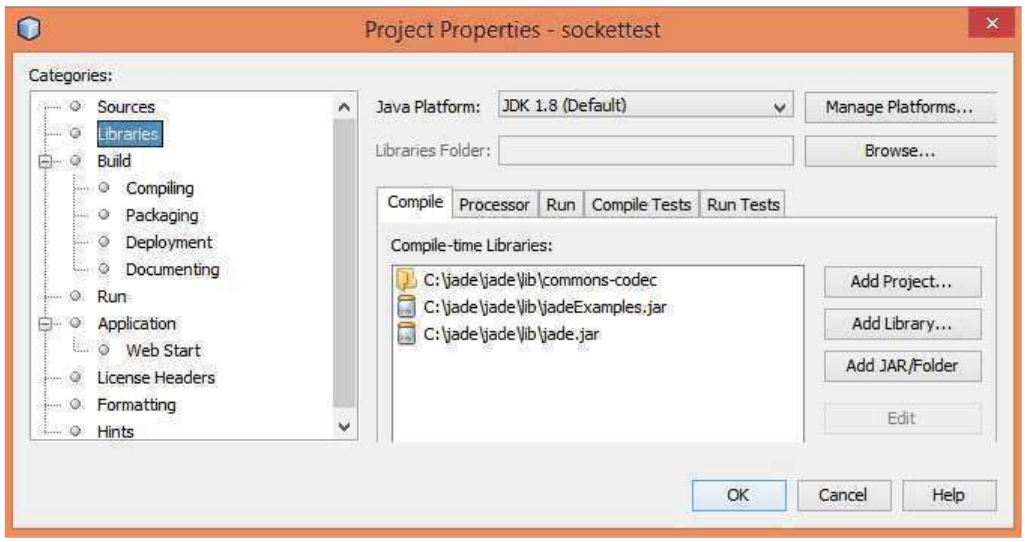

FIGURE 51 – Addition des librairies.

— Pour la compilation et le lancement des agents et des containers, vous devez configurer Run. Pour cela vous devez ajouter jade.Boot à Main Class et entrez la commande suivante dans la case Arguments :

-gui jade.Boot NomDuL'agent :LeNomDuPackage.LeNomDeLaClasse

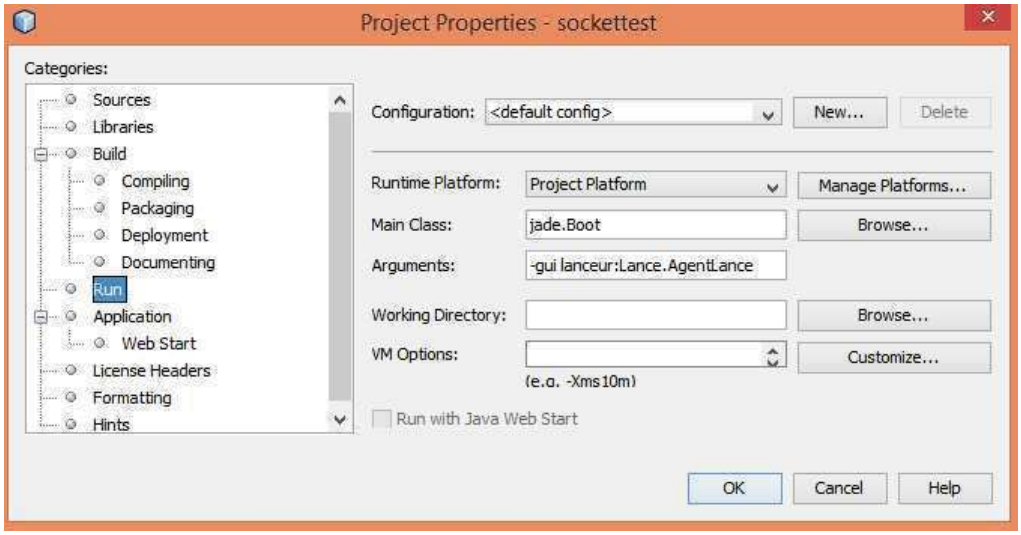

FIGURE 52 – Configuration du Run.

— Maintenant, pour lancer un Main-Container avec une application Java, vous devez écrire le code présenter dans la figure 7 à l'intérieur de la class créer précédemment. Talque au début du programme, vous devez déclarez les méthodes de jade utiliser dans votre programme, ensuit à l'intérieur de Main, créez une instance Runtime, définissez les propriétés de la plateforme et son interface graphique, créez le Main-Container en appelant la fonction createMainContainer, attribuez à cette fonction les propriétés déclarer précédemment et à la fin appelez la fonction start pour que le MainContainer se lance une fois vous compilez et lancez le programme.

```java
package main;
import jade.core.Runtime;
import jade.core.Profile;
import jade.util.leap.Properties;
import jade.core.ProfileImpl;
import jade.wrapper.AgentContainer;
import jade.wrapper.ControllerException;
import jade.util.ExtendedProperties;


public class MainContainer {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        try {
            Runtime runtime= Runtime.instance();
            Properties properties=new ExtendedProperties();
            properties.setProperty(Profile.GUI, "true");
            ProfileImpl profileImpl=new ProfileImpl(properties);
            AgentContainer mainContainer=runtime.createMainContainer(profileImpl);
            mainContainer.start();
        } catch (ControllerException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

FIGURE 53 – Programme de création d'un Main-Container en Java.

Une plateforme est un ensemble des Conteneurs, pour chaque Conteneur on doit spécifier l'adresse IP de MainContainer, pour qu'il puisse communiquer avec elle. Le programme suivant permet de créer un conteneur, à chaque lancement de ce programme un nouveau conteneur apparaitre.

```
import jade.core.Runtime;

import jade.core.ProfileImpl;
import jade.wrapper.AgentContainer;
import jade.wrapper.ControllerException;
import jade.wrapper.AgentController;

public class jadeContainer {
    public static void main(String[] args) {

            try {
                Runtime rt=Runtime.instance();
                ProfileImpl pc=new ProfileImpl(false);
                pc.setParameter(ProfileImpl.MAIN_HOST, "localhost");
                AgentContainer ac =rt.createAgentContainer(pc);

            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

    }
}
```

FIGURE 54 – Programme de création d'un Conteneur en Java.

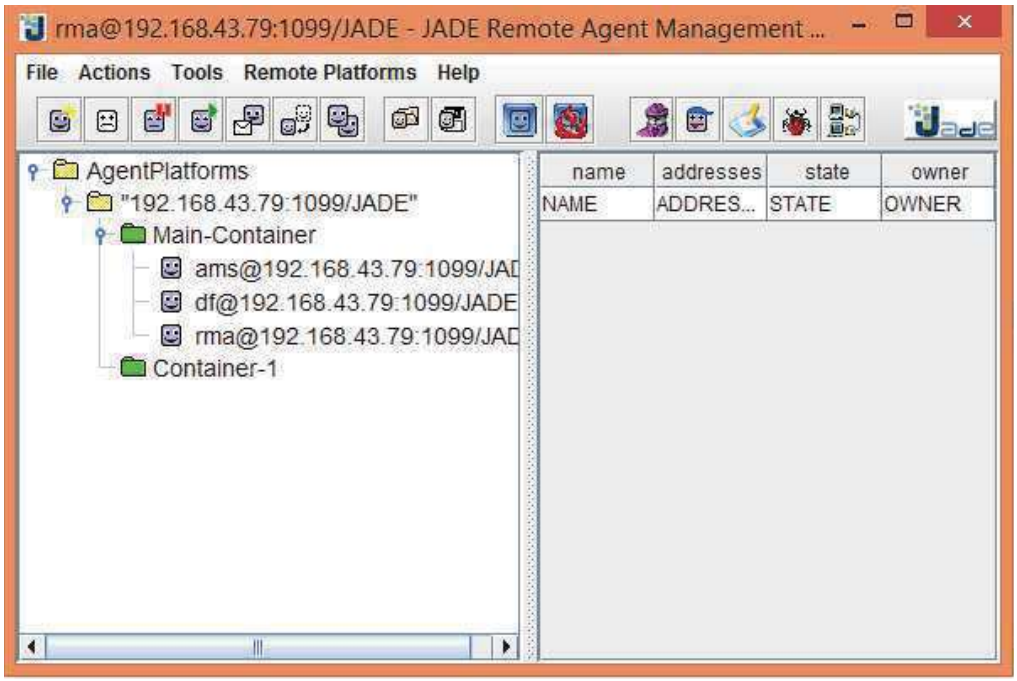

FIGURE 55 – Apparition d'un nouveau conteneur.

Chaque conteneur contient un ou plusieurs Agents, il existe deux formules pour la création de ce dernier :

- En utilisant RMA :

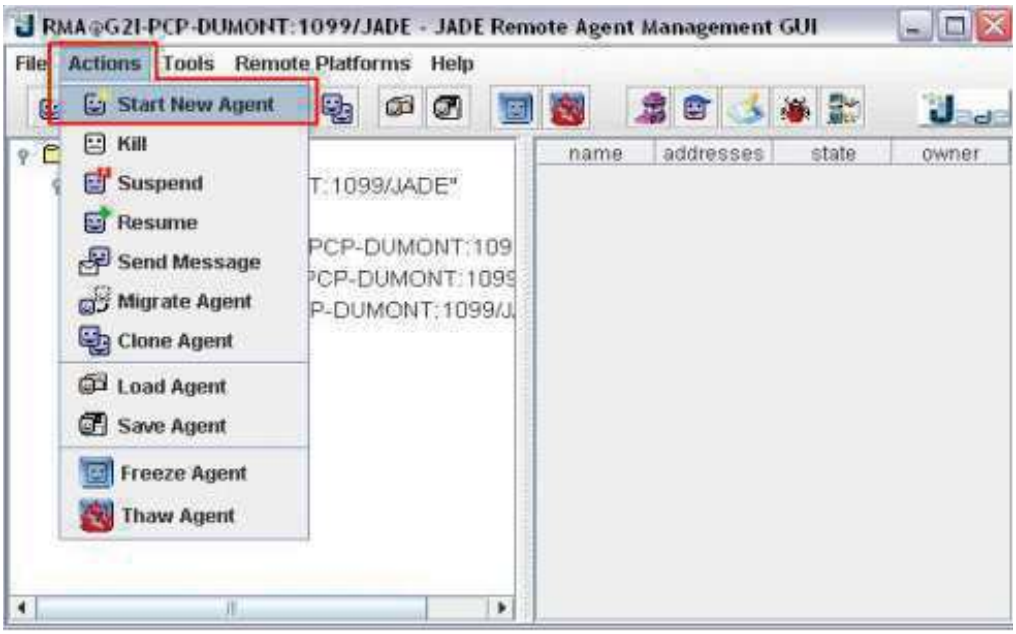

FIGURE 56 – Création d'Agent en utilisant RMA.

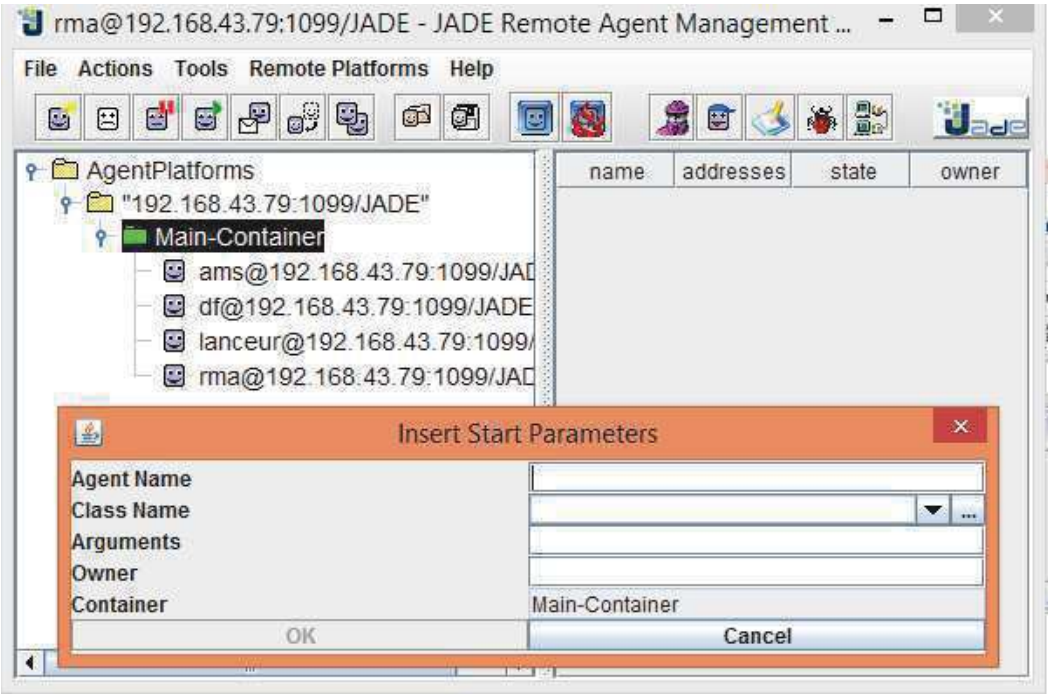

FIGURE 57 – Configuration des paramètres de l'agent en utilisant RMA.

- En programmant un code Java, telle que un agent JADE est une class qui hérite de la classe Agent et qui redéfinie des méthodes qui définissent le cycle de vie de l'agent dans la plateforme. Par exemple : la méthode **setup** qui est la première méthode qui sera appelée après instanciation de l'agent par le conteneur, la méthode **doDelate** qui permet de demander au conteneur de détruire l'agent, etc.

```
AgentContainer container = (AgentContainer) getContainerController();

t1 = container.createNewAgent("AGENTPOST1", "post1.post1", argu);

t1.start();
```

FIGURE 58 – Exemple d'un code java pour la création d'un Agent.

Dans la figure 13, la fonction **creatNewAgent** permet la création d'un nouvel agent sur le nom **AGENTPOST1** et définir sa classe **post1.post1** et ses arguments **argu**.

Chaque plateforme offre une interface graphique utilisateur (GUI) pour la gestion à distance qui permet de contrôler et superviser les états des agents, par exemple arrêter et remettre en marche un agent. L'interface graphique permet aussi de créer et de commencer l'exécution d'un agent sur un hôte éloigné, à condition qu'un réceptacle d'agents s'exécute déjà sur cet hôte. L'interface elle-même a été implémentée comme un agent, appelé RMA (Remote Monitoring Agent). Toute la communication entre les agents et l'interface (GUI) et toute la communication entre cette interface et l'AMS est faite par ACL via une extension ad hoc de l'ontologie des agents de gestion FIPA[2]. NetBeans offre deux possibilités Pour la création d'une interface graphique :

- En glissant et en sélectionnant les items dans l'interface (JLabel, JButton ou JTexteField, etc,). Cette méthode permet une configuration rapide et facile de l'interface,

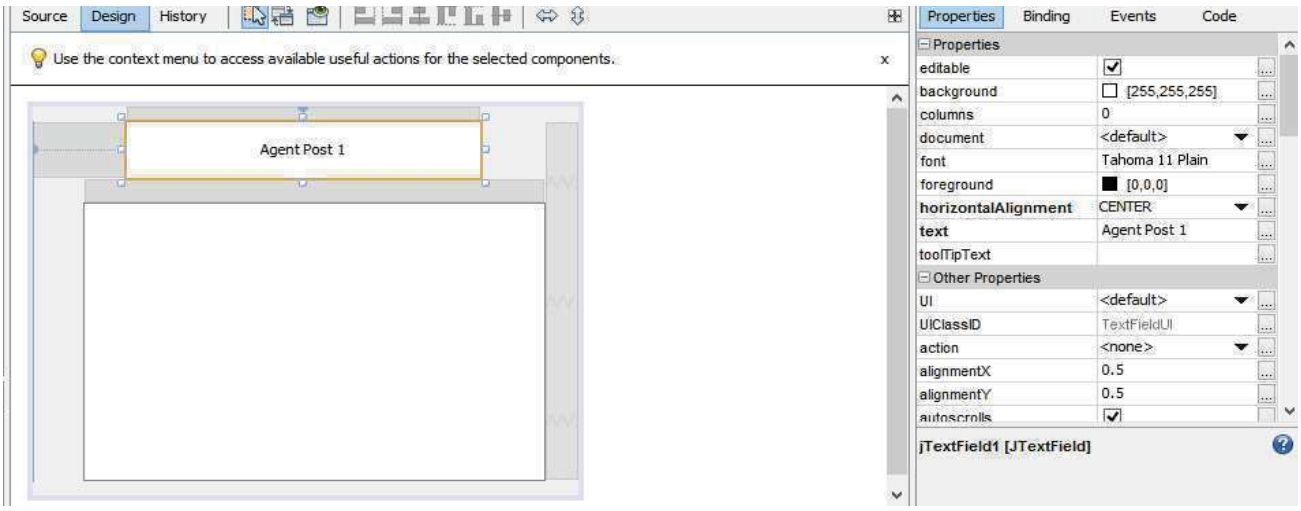

FIGURE 59 – Interface graphique.

- En créant un programme Java qui permet de configurer et lancer l'interface graphique.

```
36    public ConsommateurGui(){
37        jTextAreaMess.setFont(new Font("Arial",Font.BOLD,14));
38        jTextAreaMess.setEditable(false);
39        JPanel jPanelN=new JPanel();
40        jPanelN.setLayout(new FlowLayout());
41        jPanelN.add(jLabelAgent);
42        jPanelN.add(jTextFieldAgent);
43        jPanelN.add(jLabelLivre);
44        jPanelN.add(jTextFieldLivre);
45        jPanelN.add(jButtonEnvoyer);
46        jPanelN.add(jTextAreaMess);
47        this.setLayout(new BorderLayout());
48        this.add(jPanelN,BorderLayout.NORTH);
49        this.add(new JScrollPane(jTextAreaMess), BorderLayout.CENTER);
50        this.setSize(600, 400);
51        this.setVisible(true);
52        jButtonEnvoyer.addActionListener(new ActionListener(){
53
54            @Override
55            public void actionPerformed(ActionEvent arg0) {
56                // TODO Auto-generated method stub
57                String agentName=jTextFieldAgent.getText();
58                String livre=jTextFieldLivre.getText();
59                GuiEvent gev=new GuiEvent(this,1);
60                Map<String,Object> params=new HashMap<>();
61                params.put("AgentAcheteur", agentName);
62                params.put("Le livre", livre);
63                gev.addParameter(params);
64                consommateurAgent.onGuiEvent(gev);
65            }
```

FIGURE 60 – Exemple de création d'une interface Gui par un programme Java.

La visualisation des messages échangés entre les agents, des entrées ou sorties d'agents de la plateforme est possible en Jade, ceci en utilisant l'outil **Sniffer Agent.** Ce dernier peut être lancé en sélectionnant le conteneur, cliquant sur Tools, choisissant **Start Sniffer,** cliquant droit sur l'agent qu'on veut visualiser et à a fin choisissant la commande **Do sniff This agent(s)**

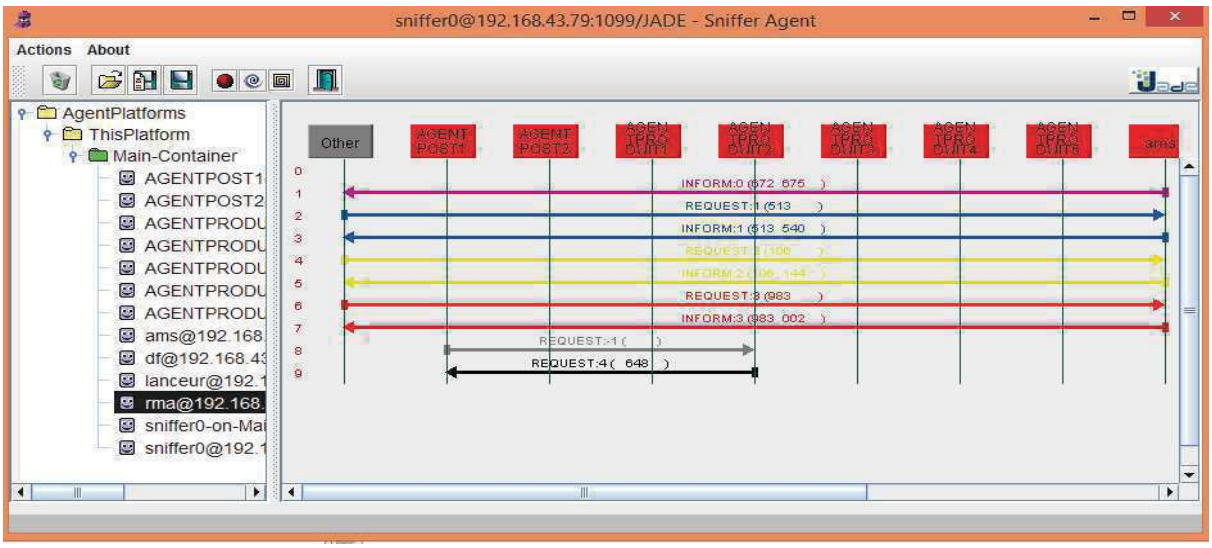

FIGURE 61 – Exemple d'un Sniffer Agent.