الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Étude

présenté par

Dirahoui Walid

&

Mouali Oumaïma

pour l'obtention du diplôme de Master en Électronique option Automatique

Thème

Développement de maquettes pédagogiques virtuelles pour la programmation des automates programmables industriels

Proposé par : Salhi Hassen & Tebani Karima

Année Universitaire 2014-2015

Dédicace

Je dédie cet humble travail

A mes chers et respectueux parents.

A mon père mon premier encadrant depuis ma naissance qui a sacrifié sa vie pour ma réussite avec son amour et son encouragement.

A ma mère qu'elle trouve ici l'hommage de ma gratitude qui, si grande qu'elle puisse être ne sera à la hauteur de ses sacrifices et ces prières pour moi.

Puisse Dieu tout puissant vous garder et vous procurer santé et bonheur.

Je dédie aussi ce modeste travail

A mes chères sœur Amel et Hanna et mon frère Amine et sa femme Nadia.

A mes neveux Rassil et Yasser, mes nièces Norhene et Rihem.

A ma grande famille, mes cousines et mes cousins.

A mes amis surtout, Lamia, Manel, Mouniret Kika.

A mes camarades Mohamed Ameziane (halliche), Kahina.

A mon Binôme et sa famille.

Sans oublier tous les enseignants que ce soit du primaire, du moyen, du secondaire ou de l'enseignement supérieur.

Oumaïma

Dédicace

Je dédie ce mémoire

A mes chers parents, ma mère et mon père

Pour leur patience leur amour leur soutiens et leurs encouragements

A mes frères, Nassim et Nihad.

A mes amis et mes camarades

A tous mes cousins et cousines

Sans oublier mes tantes, mes oncles et mes grands parents

Walid

Remerciements

Ce travail a été effectué au sein de laboratoire (salle machine) de la division Productique et Robotique du Centre de Développement des Technologies Avancées (CDTA) de Baba Hassen.

Nous tenons tout d'abord à remercier Dieu le tout Puissant qui nous a donné la force et la volonté pour réaliser ce modeste travail.

Nous tenons à remercier en premier lieu, Monsieur Salhi Hassen, Professeur et directeur de laboratoire des systèmes électriques et de télécommande au département d'électronique à l'Université Saad Dahleb de Blida, d'avoir accepté d'être notre promoteur durant ce travail et pour la confiance qu'il nous a donnée ainsi que pour sa collaboration et ses précieux conseils.

Nous tenons à exprimer notre reconnaissance à Monsieur Kazed Boualem Maitre-assistant et chef d'option de licence Automatique, pour sa patience, son aide et ces remarques pertinentes tout au long de notre travail.

Nos remerciements s'adressent à mademoiselle Tebani Karima, notre co-promotrice au CDTA pour son soutien, sa patience et son encadrement total.

Nous tenons également à remercier tous les membres de la Division de Robotique au CDTA surtout Mr Gaham Mehdi responsable de projet et Mr Mihoubi Bachir ainsi que Mr Kara Khaled et tous nos collègues pour l'excellente ambiance de travail qu'ils ont créée.

Nous remercions également les membres du jury d'avoir accepté de juger notre travail.

Madame Chentir Amina Maitre de conférence à l'université de Blida, Mr Ayad Hussein enseignant à l'université de Blida.

Nous tenons à remercier tout particulièrement et à témoigner toute notre reconnaissance aux personnes suivantes : Acherouf Mohamed Ameziane et Kassa Kahina, pour leur précieuse aide et pour nous avoir épaulé dans des moments délicats.

Nous concluons en remerciant toutes les personnes qui ont participées, de près ou de loin, à la réalisation de ce Mémoire. Merci à nos amis et nos familles qui nous ont supportés, soutenus et encouragés.

ملخص:

عملنا يتمحور حول تنفيذ محاكاة Hardware-In-the-Loop بعبارة اخرى نريد برمجة جزء من محاكاة بواسطة وحدة تحكم الية بهدف تصميم نماذج افتراضية بمعنى ان اجزاء التشغيل المصطنعة متمركزة على اساس انظمة صناعية حقيقية لأهداف بيداغوجية بحتة .

بسبب غياب الوسائل والمعدات الخاصة بهذا المجال نريد تبسيط الاعمال التطبيقية في مجال الالية لأقصى حد وكذا نريد السماح للطالب للتلاعب والتمرن على اجهزة التحكم الالية دون تعريض الطالب و المعدات للخطر.

كلمات المفاتيح: محاكاة Hardware-In-the-Loop, FlexSim, Java, OPC.

Résumé

Ce travail consiste à mettre en œuvre une simulation Hardware-In-the-Loop, en d'autres termes commander une partie simulée par un automate programmable dans le but de concevoir des maquettes virtuelles, c'est-à-dire des parties opératives virtuelles basées sur des systèmes industriels réels, et ce pour des fins pédagogiques et industrielles. En absence de matériel, nous voulons faciliter au maximum la présentation des TP d'automatique, afin de permettre aux étudiants de manipuler et de programmer des parties opératives et de charger les programmes dans l'API sans mettre en danger l'étudiant et le matériel.

Mots-clés : Simulation Hardware-In-the-Loop, Logiciel FlexSim, Java, OPC.

Abstract

This work is to implement a hardware-in-the-loop simulation, in other words controlling a simulated part by a programmable automaton, with the aim to design virtual models, that is to say, a virtual operating parts based on real industrial systems, and this is for purely educational purposes. When no equipment is available, we want to facilitate at maximum the presentation of practical work, in automatics, thus we will allow students to manipulate and program an operating part and load the programs into the PLC, without endangering neither the student nor the equipment.

Key-words : Hardware-in-the-loop simulation, FlexSim software, Java, OPC.

API : Automate Programmable industriel

CEI : Commission Electrotechnique Internationale

DLL: Dynamic Link Library

FBD: Function Block Diagram (*Schéma par blocs*)

GM : General Motors

GSM: Global System for Mobile communication

HIL: Hardware-In-the-Loop (*Objet physique dans la boucle*)

ICSP : In-Circuit Serial Programming.

IDE : Integrated Development Environment (Un environnement de développement libre de la bibliothèque java).

IP : Internet Protocol (*Protocole de communication de réseau informatique conçu pour Internet*).

LED : Light-Emitting Diode (*diode électroluminescente*).

NO : Normalement Ouvert.

OPC : OLE for Process Control.

SD : Secure Digital (*Carte mémoire amovible de stockage*).

SED : Simulation à évènements discrets.

SPA : Systèmes de Production Automatisés

TCP : Transmission Control Protocol (*Protocole de transport utilisé par internet*)

TOR : Tout Ou Rien.

UART : Universal Asynchronous Receiver Transmitter (*Émetteur-récepteur asynchrone universel*)

UDP : User Datagramme Protocol (*Protocole de télécommunication utilisé par Internet*)

USB: Universal Serial Bus (*Bus universel en serie*)

Introduction générale

Dans le domaine de l'automatique, il est difficile de relier théorie et pratique, du moins pour les apprentis que sont les étudiants. Actuellement, ce qui est proposé dans les universités s'avère, plus ou moins, insuffisant dans la mesure où, on veut que l'étudiant en automatique puisse assimiler, voire maitriser, les fondamentaux de l'automatique, surtout en ce qui concerne l'automatique industrielle. À sa sortie de l'université, l'automaticien fraichement diplômé, se retrouve très vite face à ses déficiences : il constate, en effet, que ce qu'il a appris et ce qu'il y a réellement en dehors de l'université, sont deux notions relativement différentes. Cela est dû souvent au manque de pratique proposée à l'université qui n'est, en quelque sorte, pas responsable de ces lacunes car on ne peut pas ramener l'industrie dans les salles de TP ni aller faire les TP dans les usines. On se retrouve là face à un problème d'ordre technique.

L'objectif principal de ce projet est de montrer les intérêts des plateformes de simulation des logiciels de programmation en Automatique Industrielle et de mettre en évidence les qualités des maquettes pédagogiques dans la reproduction du comportement d'un atelier de production.

Lors du *Colloque sur l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes* en 1999 [1], les premiers prototypes de partie opérative simulée, appelés Maquettes Virtuelles, avaient été présentés. Les Maquettes Virtuelles sont une représentation fidèle, tant au niveau fonctionnel que comportemental des systèmes habituellement utilisés en Travaux Pratiques (TP). Elles reposent sur une plateforme initialement développée pour des recherches dans le domaine de la conception et l'évaluation d'outils de supervision adaptés à l'homme [2]. L'idée de base est de permettre aux étudiants de profiter pleinement de la séance de TP. D'une façon très générale, une séance de TP d'Automatique vise à transmettre aux étudiants un savoir-faire concernant les outils et les langages de programmation des automates programmables mais surtout la possibilité d'appliquer les concepts théoriques vus en cours pour aboutir à une « bonne » partie commande.

Pour l'ensemble de ce travail, il est sous-entendu que la simulation citée est à événements discrets. Il faut aussi rappeler que la simulation de flux est un outil informatique qui est de plus en plus utilisé par les industriels et par les chercheurs. A.M. Law et W.D. Kelton citent, dans leur ouvrage « Simulation Modeling and Analysis », plusieurs enquêtes aux Etats-Unis qui montrent la place de la simulation dans l'industrie : l'une d'elles indique que parmi 14 techniques utilisées, la simulation arrive en 2ème position pour 84% des entreprises sondées [3]. Pour les chercheurs,

le principal intérêt est de pouvoir travailler sur un système de production virtuel, dont le comportement peut être très proche du système réel, à moindre coût et sans aucun risque. Dans le domaine de l'optimisation et de la prise de décision, les autres avantages de la simulation font que cet outil permet, depuis une dizaine d'années seulement, de mettre en œuvre des méthodes qu'il était inimaginable d'appliquer sur les systèmes réels ou sur des modèles mathématiques.

Organisation du mémoire

Ce mémoire est composé de quatre chapitres. Les trois premiers chapitres concernent une étude bibliographique sur les différents concepts avancés dans ce travail ainsi que les différents dispositifs et composants électroniques utilisés. Ainsi, le premier chapitre, comporte des généralités sur la simulation avec quelques logiciels de simulation dont celui qui nous intéresse. Ensuite, dans les second et troisième chapitres il est question de l'application à réaliser. Le quatrième chapitre consiste à présenter l'essentiel des résultats obtenus.

Introduction

Aujourd'hui, la simulation est utilisée dans de nombreux domaines de recherche et de développement : mécanique, science des matériaux, aéronautique, météorologie ... Elle intervient aussi dans des secteurs comme celui de la banque et des finances.

Dans la logique de faire plus vite, mieux et, surtout, moins cher, la simulation possède énormément d'atouts et cela est dû aux nombreuses contraintes économiques : réactivité, anticipation et compétitivité (gain de productivité), mais aussi des enjeux de sûreté et de sécurité, par une meilleure compréhension des situations accidentelles.

Ce chapitre consiste à donner un aperçu général sur la simulation en introduisant la simulation à évènements discrets, avec laquelle ont été élaborés les systèmes traités. En dernier lieu, on introduit l'outil de simulation utilisé au cours du projet.

I.1 La simulation

Les chercheurs, les ingénieurs et bien d'autres professionnels se posent souvent la question : quel est le résultat que j'obtiens si j'exerce telle action sur un élément ?

Le moyen le plus simple serait de tenter l'expérience, c'est-à-dire d'exercer l'action souhaitée sur l'élément en cause pour pouvoir observer ou mesurer le résultat. Dans de nombreux cas l'expérience est irréalisable, trop chère ou contraire à l'éthique. Il serait par exemple irrationnel de fermer arbitrairement l'un des guichets d'une banque et de laisser le public s'accumuler dans le hall pour le seul intérêt d'observer le phénomène. Il serait de même difficilement convenable de construire une extension d'un atelier de production pour en étudier les effets et de la détruire ensuite, voyant qu'elle n'a pas apporté les résultats désirés.

C'est à cause des difficultés liées à l'expérimentation directe, dans le but de pouvoir en tirer des résultats concrets, que le recours à la simulation est devenu nécessaire voire indispensable, ainsi la simulation consiste à concevoir un modèle du système (réel) étudié et de mener des expérimentations sur ce dernier afin d'interpréter les résultats fournis par le déroulement du modèle puis formuler des décisions relatives au système, selon nos besoins.

I.1.1 Quelques exemples d'application

Les domaines que touche la simulation sont divers, nous allons en citer quelques exemples :

- Conception et analyse des systèmes de productions.
- Évaluation de la logistique d'organisation de services: hôpitaux, bureau de poste, restaurants.

– Étude des systèmes à grande échelle (économie, finance, biologie).

I.1.2 Les raisons du succès de la simulation

Les raisons du succès de la simulation sont variées : la disponibilité d'une puissance de calcul toujours croissante, la disponibilité d'outils, logiciels, conçus exprès pour la simulation et la nécessité et la volonté de manipuler des modèles de plus en plus détaillés et complexes.

I.2 La simulation à évènements discrets

Une simulation à événements discrets (le mot discret ici ne signifie ni temps discret ni état discret) est une modélisation informatique où l'état d'un système est représenté par une séquence chronologique d'événements discrets. Nous appelons par « évènement » tout changement d'état du système réel se produisant à un instant donné, ainsi que les actions qui accompagnent ce changement. La prise de décision dépend exclusivement de la modification de l'état du système donc des évènements [4].

Afin d'éclaircir l'idée de la SED[1] rien ne vaut un exemple explicatif : ainsi dans le cas d'un péage d'autoroute, si aucune voiture ne se présente, il n'y a pas d'événement et l'état du système ne change pas alors il n'y a rien à observer. Dans ce type de simulation on ne connaît pas à l'avance le nombre d'événements que comportera une simulation.

I.3 Les logiciels de simulation

Pour simuler n'importe quel système il faut avant tout des outils informatiques puissants et l'un des outils les plus importants c'est le logiciel de simulation, nous allons donc citer quelques-uns :

I.3.1 Logiciel Arena

Le logiciel Arena est désigné essentiellement pour la création des modèles animés et pour représenter virtuellement n'importe quel système. La modélisation d'un système se fait en plaçant des objets ou des blocs (appelés modules) représentant ses comportements. Une telle approche de modélisation permet d'obtenir une structure du modèle proche de celle du système réel à simuler. Il a été conçu en 1982 par C.D. Pedgen de *System Modeling Corporation* [5].

[1] Simulation à évènements discrets

I.3.2 Logiciel Witness

WITNESS est un logiciel de simulation qui fournit un ensemble d'outils professionnels pour modéliser et simuler tous les processus d'activité, comme l'industrie manufacturière. Il permet aussi d'élaborer des stratégies d'amélioration continue. Il offre une plate-forme accessible et flexible pour tester et vérifier nos scénarios de changement sans risque et de façon rentable [6].

I.3.3 FlexSim

Le logiciel FlexSim est un outil d'analyse et de simulation de système manufacturier. Il permet, entre autres, de simuler plusieurs alternatives d'aménagement afin de laisser le concepteur choisir celle qui répond à ses besoins. Il s'avère très efficace en ce qui concerne l'optimisation des couts de projets avant leur réalisation, car grâce à la simulation avec FlexSim on peut estimer les dépenses dont on a besoin pour l'aboutissement d'un quelconque projet industriel, en simulant sa production par exemple, on peut aussi estimer le nombre de personnel et de machines de production nécessaires pour arriver à un chiffre de production précis [7].

Pour ce projet, le logiciel **FlexSim** a été choisi comme outil de simulation, particulièrement pour sa flexibilité, car il offre une interface simple d'utilisation qui accélère la prise en main du logiciel et permet, notamment, de faire de la simulation à évènements discrets grâce aux objets discrets que comporte sa bibliothèque. Il est en adéquation avec l'objectif fixé par ce travail qui consiste à réaliser une simulation Hardware-In-the-Loop. En effet, il comporte des fonctionnalités qui permettent de communiquer, dans un même réseau, avec d'autres composants tels que la communication par Sockets[2].

- **Historique**

FlexSim Software Products a été fondé par Bill Nordgren (co-fondateur de Promodel Corporation) en 1993, Roger Hullinger, et Cliff King, à l'origine sous le nom de F&H Simulations. F&H Simulations fournissait des services de vente, support et formation sur le

[2] Un socket est une structure de données abstraite qui est utilisée pour établir un canal de communication permettant l'envoi et la réception d'informations entre des processus qui s'exécutent dans un environnement distribué

logiciel de simulation Taylor II - propriété de la société hollandaise F&H Simulation B.V (F&H Holland).

En 1998, F&H Holland a développé la première génération de moteur de simulation 3D orienté objet : Taylor ED (Enterprise Dynamics). F&H Simulations a contribué au développement d'objets robustes intégrés dans Taylor ED. En complément, F&H Simulations a contribué à vendre, conseiller, et former sur le nouveau logiciel.

En 2000, F&H Holland a été racheté. F&H Simulations a saisi cette opportunité pour devenir indépendant. Dr. Eamonn Lavery et Anthony Johnson ont rejoint la société pour superviser le développement d'un logiciel de simulation 3D orienté objet de nouvelle génération: FlexSim. F&H Simulations a changé son nom en FlexSim Software Products.

La version 1.0 de FlexSim a été délivrée en février 2003, dotée d'un tout nouveau moteur de simulation dernier cri, un environnement de modélisation en 3D, et une intégration en C. Depuis sa première version, FlexSim est devenu un standard de référence dans le domaine de la simulation par événements discrets [8].

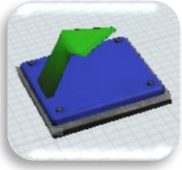
- **La librairie FlexSim**

FlexSim possède une bibliothèque très riche en objets qui permet de faire de la simulation à évènements discrets, grâce aux objets discrets qu'elle comporte, comme elle permet de faire d'autres type de simulation à l'aide de ses objets « fluides » (ex : niveau de remplissage de réservoir). Nous, ce qui nous intéresse c'est évidemment les objets discrets car nous faisons de la simulation à évènements discrets.
Voici les différentes classes de la bibliothèque FlexSim :

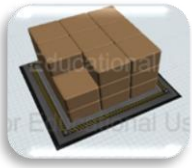
- **Fixed Ressources (les ressources fixes)**

La classe suivante contient, comme son nom l'indique, des ressources fixes : *Source, Sink, Queue, Processor …*

- *Source :* les sources sont l'une des composantes les plus importantes de notre processus, il faudrait au minimum en avoir une dans notre modèle car sans source il n'y aurait pas de produit, donc pas de système simulé. Leur rôle est de créer et de relâcher les produits du système « Flowitems ».

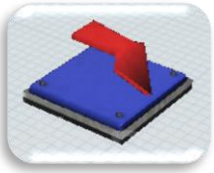


- *Queue :* ce sont des zones tampons, des files d'attentes. Ils gardent temporairement les produits « Items » lorsque les procédés ne peuvent les accepter. Ils peuvent recevoir plusieurs types d'items à la fois jusqu'à ce que leur capacité maximale soit atteinte.

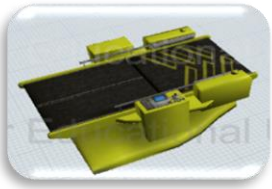


- *Processor :* ce sont les procédés qui transforment les produits « FlowItems » ou force un délai lorsque la capacité est atteinte.

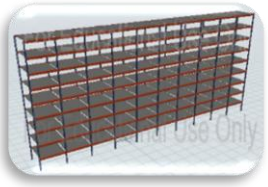
- *Sink :* les sink représentent la fin de notre processus. Ils reçoivent les produits et les enlèvent de la simulation. On peut le considérer comme l'expédition d'une entreprise.



- *Combiner :* le combiner est utilisé pour regrouper plusieurs « Flowitems » ensembles pendant qu'ils le parcourent. Il peut soit les joindre ensembles de façon permanente, ou bien il peut les emballersorte qu'ils peuvent être séparés plus tard.



- *Rack :* le « Rack » est utilisé pour stocker les produits « Flowitems ». On peut aussi modifier les dimensions et le nombre de ses colonnes, et le nombre de niveaux qu'il peut comporter.



- *BasicFR :* c'est un objet construit pour être personnalisé en un objet de la libraire de l'utilisateur Il contient pratiquement toutes les options qu'on trouve dans un objet appartenant aux Ressources Fixes. À noter que sa forme est donnée par défaut sous forme d'un cube gris, on peut facilement la modifier dans l'onglet General → 3D Shapes.

Tous les objets cités ci-dessus possèdent des paramètres pouvant être modifiés. On peut accéder au menu d'édition en cliquant deux fois sur l'objet ou bien en cliquant grâce au bouton droit de la souris sur l'objet et on sélectionne « Properties ».

Chaque classe d'objet possède des caractéristiques qui lui sont propres. La fenêtre de propriétés possède plusieurs onglets reliés à différentes fonctions. Voici par exemple la fenêtre de paramétrage de la source :

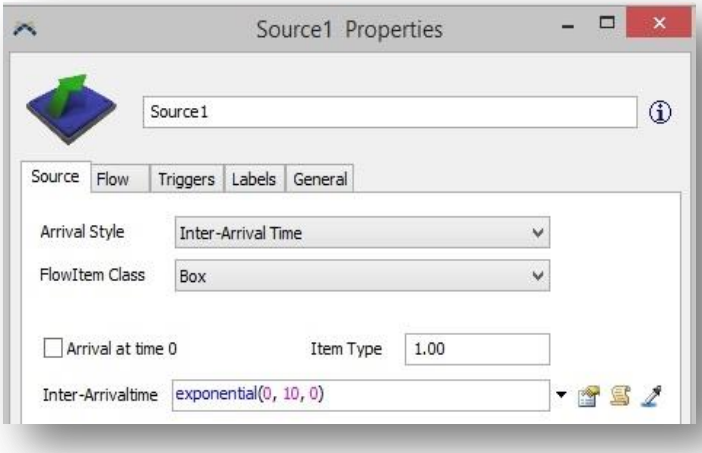


Figure I-1. Propriétés de la Source

Avec :

- *L'onglet Source :* on peut régler le mode d'arrivé « Arrival Style » dont il existe trois modes d'arrivées pour les sources : un temps d'inter arrivé « interarrival time », une planification « arrival schedule » et enfin une séquence « arrival sequence ». On peut choisir aussi le type du produit « FlowItem Class ».

- *L'onglet Flow :* dans cet onglet il s'agit du paramétrage du flux des produits et du choix de leur destination « Send To Port » et on peut aussi utiliser un transporteur tel un robot par exemple, en cochant « Use Transport » et en indiquant le transporteur évidement.

- *Triggers :* on peut définir, dans cet onglet, le comportement de l'Objet en fonctions d'évènements par exemple : lors de la réception d'un message, de la création d'un produit ou bien lors de la sortie du produit, etc.

- *Labels :* dans l'onglet suivant onglet, on peut étiqueter nos produits lors de leur passage par tel ou tel Objet.

- *General :* ici on peut modifier les dimensions, la couleur, la texture, la position, la rotation, de notre Objet.

➢ **Task Executers (Sources Mobiles)**

La classe que nous allons aborder maintenant, contient des exécuteurs de taches tel que le *Transporter, Operator, Robot …* et un *Dispatcher* pour synchroniser leur travail.

Voici quelques-uns des Objets qui composent cette classe :

- *TaskExecuter, Operateu et Transporter :* ce sont des objets qui ont à peu-près les mêmes propriétés, on choisira un en dépend de l'autre en fonction de notre système, pour le rendre plus réaliste.

- *Elevator :* l'élévateur est un type spécial de transporteurs qui déplace de bas en haut ou bien de haut en bas les produits.

- *Robot :* de même que l'élévateur, le robot est type spécial de transporteurs sauf qu'il transporte les produits plus librement par rapport à l'élévateur.

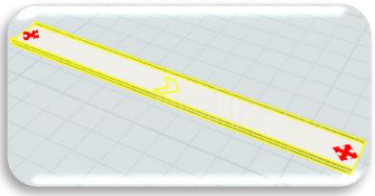
- *BasicTE :* de même que le BasicFR, le BasicTE est un Exécuteur de Taches personnalisables qui contient les mêmes paramètres que n'importe quel objet appartement à la librairie *TaskExecuters.*

Comme pour les objets des Ressources Fixes, les objets des Sources Mobiles possèdent des paramètres pour une configuration de l'objet selon le besoin.

➢ **Conveyors (convoyeurs)**

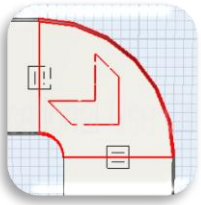
Cette librairie est dédiée à la simulation des systèmes de convoyages. Elle permet de configurer un système de telle sorte qu'il soit le plus proche possible de la réalité et ce grâce aux différents objets qui la composent et qu'on va citer ci-dessous :

– *Straight Conveyor (Convoyeur droit) :* il peut simuler, par exemple, un convoyeur à rouleaux. Comme son nom l'indique c'est un convoyeur droit.

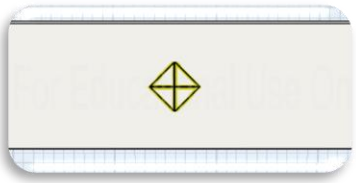


– *Curved Conveyor (Convoyeur courbé) :* la seule différence avec le convoyeur précèdent c'est que ce dernier a une forme courbée.

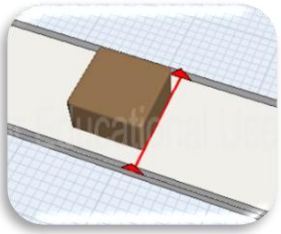
– *Join Conveyors (Jonction de convoyeur) :* c'est plus un outil qu'un objet, il permet de former une jonction courbé entre deux convoyeurs positionnés à 90º l'un par rapport à l'autre.



– *Decision Point (Point de décision) :* les points de décisions peuvent être assimilés à des capteurs dans un système automatisés, ils nous permettent donc de paramétrer la logique de décision.



– *Photo Eye (Cellule photoélectrique) :* les cellules photoélectrique sont similaires aux points de décision dans le sens où leur comportement est celui d'un capteur. Cependant, alors qu'un point de décision se déclenche à chaque fois qu'un article le traverse, la cellule photoélectrique fonctionne comme un vrai capteur photoélectrique c'est-à-dire qu'il n'y aura pas de remise à zéro tant qu'il n'y aura pas d'espace entre les objets dans le convoyeur.



– *Motor (Moteur) :* Les moteurs peuvent être utilisés si les systèmes de convoyage sont allumés ou éteints à un moment donné.

– *Merge Controller (Contrôle des lignes de convoyage) :* c'est un objet qui permet contrôler la façon dont différentes voies de transport se mélangent.

- **Langages de programmation de FlexSim**

FlexSim peut être programmé de trois manières différentes, soit avec le langage C++ ou bien FlexScript[3] qui a une syntaxe de programmation similaire à celle du langage C++ mais qui n'a pas besoin d'être compilé contrairement à ce dernier, sinon on fait appel à la bibliothèque DLL[4].

Conclusion

Ce chapitre contient des généralités sur la simulation, le type de simulation utilisée, et enfin, une vue globale sur le logiciel FlexSim ainsi qu'une esquisse sur les différents éléments qui composent sa bibliothèque.

[3] Langage de programmation spécifique à FlexSim
[4] Dynamic Link Library

Introduction

Dans ce second chapitre, on va rentrer un peu plus dans le vif du sujet, il est question des sujets qui touchent directement le travail pratique comme la simulation Hardware-In-the-Loop qui est le noyau de ce projet. Ce chapitre contient les types de systèmes à simuler en plus d'une introduction sur les automates programmables et leurs caractéristiques. En dernier lieu, une simulation d'un petit système de production est proposée.

II.1 Simulation Hardware-In-the-Loop (HIL)

La simulation Hardware-in-the-Loop est une méthode de simulation qui se distingue par le fait qu'elle associe de véritables composants, connectés à une partie temps-réel simulée.

La partie simulée peut contenir une seule partie du système (système physique seulement), et les actionneurs avec les capteurs peuvent être réels (physiques) comme elle peut contenir tout le système (système physique + capteurs + actionneurs) ce qui est le cas dans ce projet, c'est-à-dire que l'API représente, seul, la partie physique [9].

Les avantages qu'offre la simulation Hardware-in-the-Loop sont multiples, en voici quelques-uns :

- La conception et le test des systèmes de contrôle peuvent être faits sans le système réel (le travail peut s'effectuer en laboratoire).
- Les conditions d'essais sur système de contrôle matériel peuvent être poussées à l'extrême (exemple : haute/basse température, fortes accélérations et choques mécaniques, compatibilité électromagnétique).
- Il est possible d'effectuer des essais sur les effets de défauts et pannes des actionneurs, capteurs et ordinateurs sur l'ensemble du système.

La figure suivante illustre ce qu'est une simulation Hardware-in-the-loop :

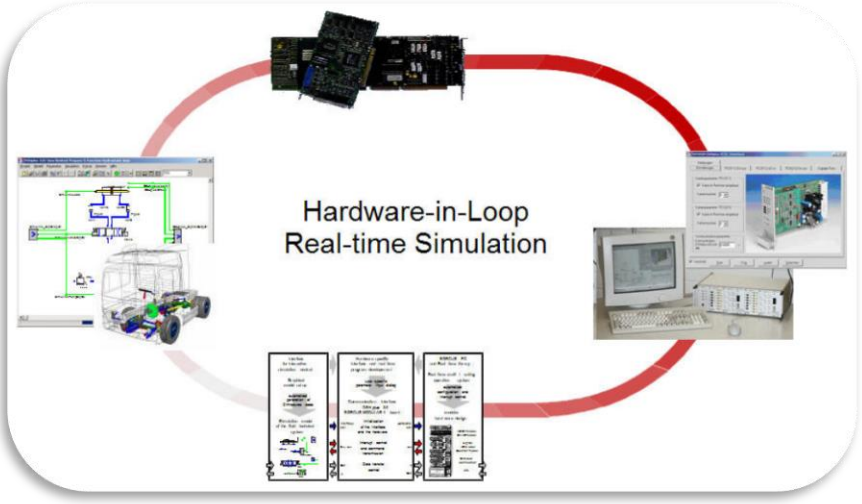


Figure II-1. Simulation Hardware-in-The-Loop temps réel simulé

Le domaine de la simulation est très vaste et les possibilités de simulation avec FlexSim sont infinies. L'un des domaines où l'Automatique est la plus répondue est celui des systèmes de production automatisés (SPA), c'est ce qui explique le choix de simulation de ses systèmes dans ce projet.

II.2 Les systèmes automatisés

"Depuis toujours l'homme est en quête de bien être". Cette réflexion peut paraître bien éloignée d'un cours de Sciences Industrielles, pourtant c'est la base de l'évolution des sciences en général, et de l'automatisation en particulier. L'homme a commencé par penser, concevoir et réaliser. Lorsqu'il a fallu multiplier le nombre d'objets fabriqués, produire en plus grand nombre, l'automatisation des tâches est alors apparue : remplacer l'homme dans des actions pénibles, délicates ou répétitives.

Citons pour exemple quelques grands hommes, avec les premiers développements de l'ère industrielle au XVIIIème siècle, Watt, avec ses systèmes de régulation à vapeur, Jacquard et ses métiers à tisser automatiques... Une liste exhaustive serait bien difficile à établir !

Enfin, le développement des connaissances, et des outils mathématiques, ont conduit à un formidable essor des systèmes automatisés, et des systèmes asservis, dans la deuxième moitié du 20ème siècle.

L'automatisation consiste à « *rendre automatique* » les opérations qui exigeaient auparavant l'intervention humaine, elle est considérée comme l'étape d'un progrès technique ou apparaissent des dispositifs susceptibles de seconder l'homme, non seulement dans ses efforts musculaires, mais également dans son travail intellectuel de surveillance et de contrôle [10].

Dans l'industrie, les automatismes sont devenus indispensables : ils permettent d'effectuer quotidiennement les taches les plus pénibles, répétitives et dangereuses. Parfois, ces automatismes sont d'une telle rapidité et d'une telle précision, qu'ils réalisent des actions impossibles pour un être humain. L'automatisme est donc synonyme de productivité et de sécurité.

Le savoir-faire de l'opérateur est transposé dans le système automatisé, il devient le « Processus ».

Un processus peut être considéré comme un système organisé qui utilise des ressources (personnel, équipement, matériels et machines, matière première et informations) pour transformer des éléments entrants (les intrants) en éléments de sortie (les extrants) dont le résultat final attendu est un produit.

La figure II-2 montre un schéma représentatif d'un système automatisé :

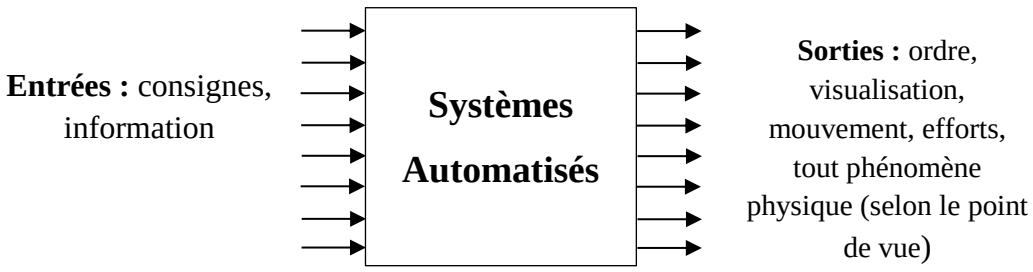


Figure II-2. Schéma représentatif d'un système automatisé

Les systèmes automatisés sont partout, ils font partie de notre quotidien, on ne peut plus s'en séparer car ils représentent une avancée technologique exceptionnelle, qui facilite largement la vie de l'homme. En voici quelques exemples :

Figure II-3. Distributeur automatique de boissons

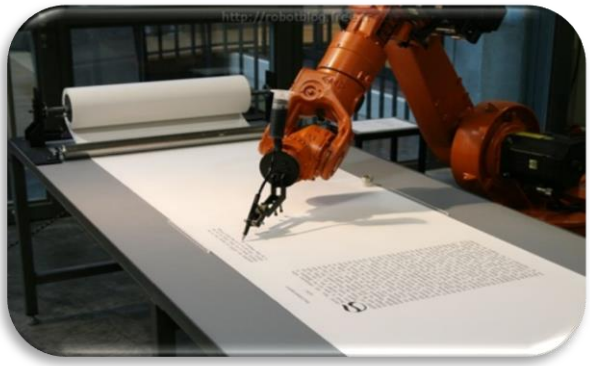


Figure II-4. Robot automatique industriel

Figure II-5. Chaine de production automatisée

Un système automatisé est composé de deux parties, la partie commande et la partie opérative.

- **La partie opérative**

La partie opérative est formée de l'ensemble de divers organes physiques comme les pré-actionneurs, les actionneurs et les capteurs, qui interagissent sur le produit pour lui conférer une valeur ajoutée. Les pré-actionneurs, servent de relais de puissance entre la commande et les actionneurs qui agissent et transforment le produit. Les capteurs recueillent les informations : état ou position du produit, alarmes, etc. traduisant un changement d'état du procédé [11].

- **La partie commande**

La Partie Commande est l'ensemble des moyens logiciels et d'informations concernant le pilotage et la conduite du procédé. Elle donne des ordres à la partie opérative à travers les actionneurs et reçoit ses comptes rendus grâce aux capteurs [12].

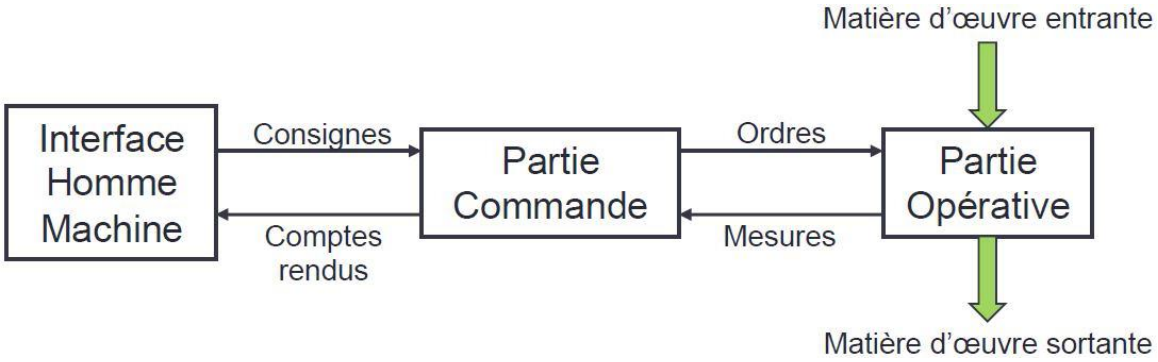


Figure II-6. Schéma explicatif montrant le rôle de la partie commande et la partie opérative dans un système automatisé

II.3 Les Automates Programmables Industriels (API)

Qui dit automatique, ou bien SPA[5], dit forcement API[6]. Dans ce projet, un automate programmable représente la partie physique dans la simulation Hardware-in-The-Loop.

Les automates programmables industriels sont apparus à la fin des années soixante, à la demande de l'industrie automobile américaine GM[7], qui réclamait plus d'adaptabilité de leurs systèmes de commande.

Les coûts de l'électronique permettaient alors de remplacer avantageusement les technologies actuelles.

Un API est une forme particulière de contrôleur à microprocesseur qui utilise une mémoire programmable pour stocker les instructions et qui implémente différentes fonctions, qu'elles soient logiques, de séquencement, de temporisation, de comptage ou arithmétiques, pour commander les machines et les processus. Il est conçu pour être exploité par des ingénieurs, dont les connaissances en informatique et langages de programmation peuvent être limitées [13].

II.3.1 La Structure d'un API

Cet ensemble électronique gère et assure la commande d'un système automatisé. Il se compose de plusieurs parties et notamment d'une mémoire programmable dans laquelle l'opérateur écrit, dans un langage propre à l'automate, des directives concernant le déroulement du processus à automatiser. Son rôle consiste donc à fournir des ordres à la partie opérative en

[5] Systèmes de Production Automatisés
[6] Automate Programmable industriel
[7] General Motors

vue d'exécuter un travail précis comme par exemple la sortie ou la rentrée d'une tige de vérin, l'ouverture ou la fermeture d'une vanne. La partie opérative lui donnera en retour des informations relatives `à l'exécution du travail [11].

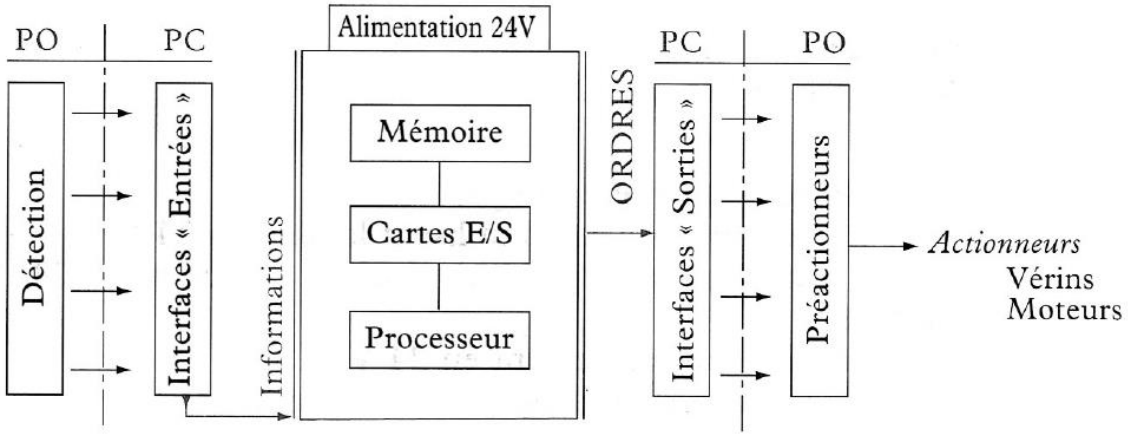


Figure II-7. Structure interne d'un API **[11]**

Un API se compose de quatre parties principales : une mémoire, un processeur, des interfaces d'Entrées/Sorties, une alimentation (240 V_{ac} → 24 V_{cc}) [14].

II.3.2 Description des éléments d'un API

- **Le processeur**

Son rôle consiste d'une part à organiser les différentes relations entre la zone mémoire et les interfaces d'E/S et d'autre part à gérer les instructions du programme.

- **Les modules d'Entrées/Sorties**

L'interface d'Entrées comporte des adresses d'entrée, une pour chaque capteur relié. Tandis que, l'interface de Sorties comporte des adresses de sorties, une pour chaque pré-actionneur. Le nombre d'E/S varie suivant le type d'automate. Les cartes d'Entrées/Sorties ont une modularité de 8, 16 ou 32 voies. Elles admettent ou délivrent des tensions continues entre 0 et 24 V_{cc}.

- **La mémoire**

Elle est conçue pour recevoir, gérer, stocker des informations issues des différents secteurs du système que sont le terminal de programmation (PC ou console) et le processeur, qui lui gère et exécute le programme. Elle reçoit également des informations en provenance des capteurs. Il

existe dans les automates plusieurs types de m´mémoires qui remplissent des fonctions différentes :

- la conception et l'élaboration du programme font appel à la RAM et l'EEPROM.
- la conservation du programme pendant l'exécution de celui-ci fait appel à une EPROM.

- **L'alimentation**

Tous les automates actuels utilisent un bloc d'alimentation alimenté en 240 V_{ac} et délivrant une tension de 24 V_{cc}.

II.3.3 Langages de programmation d'un API

Chaque automate possède son propre langage. Mais par contre, les constructeurs proposent tous une interface logicielle répondant à la norme CEI[8]. Cette norme d´définit cinq langages de programmation utilisables, qui sont [15] :

- **Grafcet :**

Ce langage de programmation de haut niveau permet la programmation aisée de tous les procédés séquentiels.

- **Schéma FBD**

Le langage FBD[9] permet de programmer graphiquement à l'aide de blocs, représentant des variables, des opérateurs ou des fonctions. Il permet de manipuler tous les types de variables.

- **Schéma Ladder**

Ce langage graphique est essentiellement dédié à la programmation d'équations booléennes.

- **Texte structuré**

Ce langage est un langage textuel de haut niveau. Il permet la programmation de tout type d'algorithme plus ou moins complexe.

- **Liste d'instructions**

Ce langage textuel de bas niveau est un langage à une instruction par ligne. Il peut être comparé au langage assembleur.

[8] Commission Electrotechnique Internationale
[9] Function Block Diagram

À présent, la mise en œuvre de la simulation Hardware-in-The-Loop est possible. La partie simulée qui est représentée par un système industriel sur FlexSim tandis que l'automate constitue la partie physique. Il reste juste à établir une communication entre ces deux parties. Ceci se fera grâce aux deux logiciels : NetBeans (sous langage Java) et KEPServerEX (serveur OPC), qui feront le lien entre l'API et le système simulé.

II.4 Le langage Java

Le langage java est un langage très fréquent dans le domaine informatique, il est à la fois un langage de programmation informatique orienté objet et un environnement d'exécution créé par James Gosling et Patrick Naughton employés de Sun Microsystems avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982). Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux…) et il reprend en grande partie la syntaxe du Langage C++. Java donne aussi la possibilité de développer des applications autonomes mais aussi, et surtout, des applications client-Serveur.

Dans ce projet on utilise le logiciel NetBeans comme environnement de programmation dans le but d'établir une communication client-serveur entre FlexSim (Logiciel de Simulation cité dans le chapitre précédant) et l'API avec, NetBeans comme client et FlexSim comme serveur, et pour que cela soit possible il faut une communication avec ce même API, que NetBeans ne peut pas faire, alors on fait appel à un troisième élément, il s'agit du serveur OPC[10] [16, 17].

II.5 Le serveur OPC

L'OPC est un protocole de communication très important dans le domaine de l'informatique industrielle par ses nombreux avantages (Pérenniser les installations, Donner une indépendance à l'utilisateur et aux éditeurs de logiciel, éviter la prolifération de protocoles communicants, accéder aux données depuis n'importe quel point du réseau **…**). Ce protocole est une technique apparue en 1995 conçu par Microsoft pour relier ses applications Windows et les matériels et logiciels du contrôle de processus. Ce serveur consiste à fournir une méthode permettant à différents logiciels d'accéder aux données de dispositifs de contrôle de processus, comme un automate d'où la nécessité d'avoir OPC pour établir la communication FlexSim-Java-API ([18], [19], [20]).

[10] OLE for Process Control

II.6 Système de d'emballage

Voici un système d'emballage de bouteilles d'eau de petit format (30cl) avec lequel une simulation Hardware-In-the-Loop est établie. Dans ce système l'association Java-OPC est utilisée comme intermédiaire entre l'API et FlexSim.

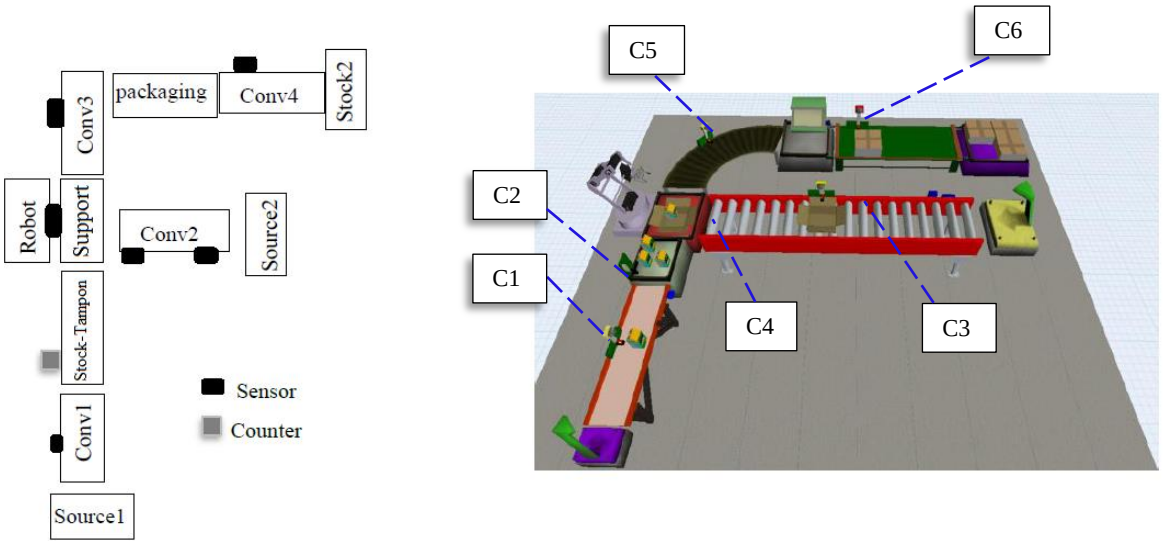


Figure II-8. Système de production et d'emballage

II.6.1 Cahier de charges du système d'emballage

Le système (Figure II-8) se compose de quatre convoyeurs (Conv1→Conv4), de 6 capteurs (C1→C6), d'un robot, d'un stock-tampon, d'une machine de fermeture de boites en carton, de deux sources (vu que notre système est une partie d'un grand système de production, les sources, 1 et 2, représentent juste l'arrivage des bouteilles et des boites en carton depuis leurs machines de fabrication) et enfin, d'un stock final.

Le 'Conv1' permet de transporter les bouteilles d'eau depuis la 'Source1' vers le stock-tampon et le 'Conv2' permet de transporter les boites en carton depuis la 'Source2' vers un support tandis que le 'Conv3' transporte les boites en carton contenant le produit vers la machine de fermeture des boites alors que le 'Conv4' conduit les boites fermées vers le stock final. Le capteur 'C1' détecte la présence des bouteilles au niveau du 'Conv1', le capteur 'C2' incrémente un compteur 'Compt1' qui compte 4 unités (boites de lait) puis, lorsqu'il est activé le convoyeur 'Conv1' s'arrête. C'est le même principe qui est utilisé pour le convoyage des boites en carton : nous avons un capteur 'C3' qui détecte la présence des boites et un autre capteur 'C4' qui incrémente un compteur 'Compt2' qui compte une seule unité puis fait arrêter le convoyeur

'Conv2'. Le capteur 'C5' est un capteur de présence qui lorsqu'il détecte la présence des cartons contenant le produit, fait démarrer la machine à ruban adhésif. Il représente aussi le Reset des deux compteurs, compteur 1 et 2.

Enfin, le capteur 'C6' incrémente un compteur 'Compt3' qui compte le nombre de boites fermés qui se dirigent vers le stock final.

II.6.2 Grafcet du système d'emballage

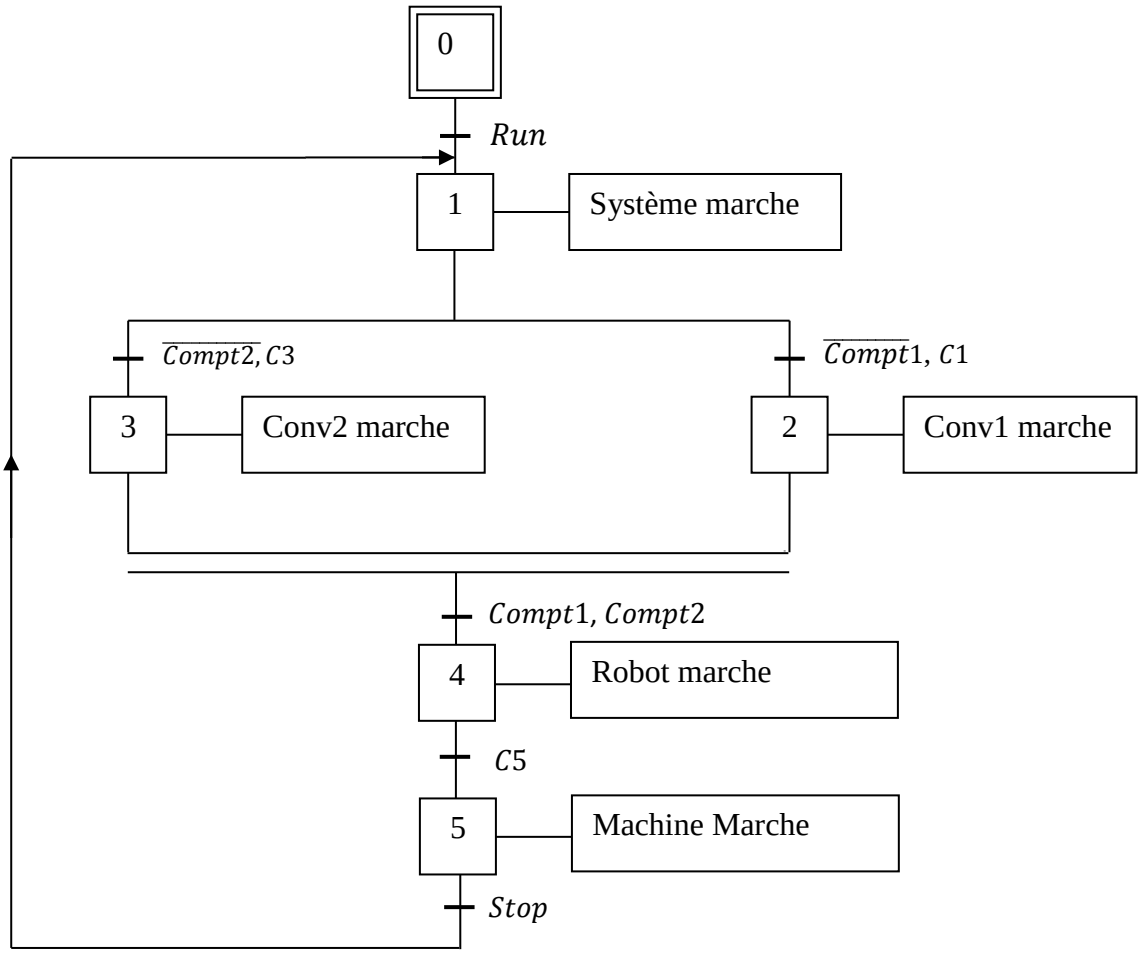


Figure II-9. Grafcet du système de production

II.7 Le Schneider TWDLCAA24DRF

L'automate programmable utilisé pour la simulation des systèmes, est le Schneider TWDLCAA24DRF. C'est un automate compact connu pour sa flexibilité et sa portabilité. Il est programmé grâce au logiciel TwidoSuite. Il dispose de 24 entrées/sorties TOR[11] dont 14 entrées et 10 sorties relais. Il est aussi doté de deux points de réglage analogiques. Il est muni d'un port série intégré et présente un emplacement pour un port série supplémentaire. Il accepte jusqu'à quatre modules d'expansion d'E/S. Sa tension de sortie est de l'ordre de 24V et son alimentation varie entre 100 et 240V. Il est très pratique dans les petites applications, d'où notre choix de cet automate pour mettre en place notre simulation [22].

À noter que pour visualiser le changement d'état des entrées/sorties nous avons dû trouver une astuce, et ce à cause de l'absence de câblage des entrées/sorties de l'API étant donné qu'on utilise un logiciel (OPC) pour communiquer avec le PC (FlexSim). OPC récupère l'information à partir des mémentos que nous avons placé en sortie (bobine) dans le programme Ladder, puis on a placé ces derniers aussi en entrée (contact NO[12]) qui vont exciter la bobine en sortie qui n'est d'autre qu'une une sortie réel de l'automate. Ainsi on a pu visualiser le changement d'état des entrées/sorties de l'API.

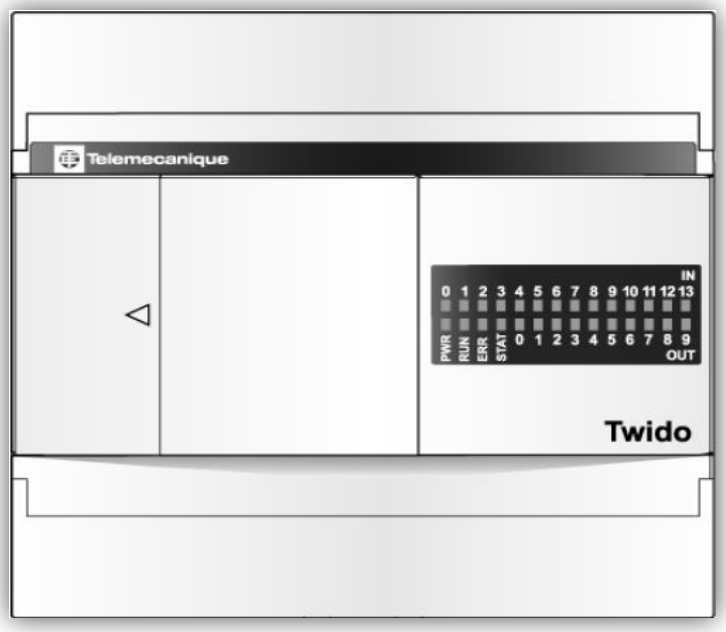


Figure II-10. Schneider TWDLCAA24DRF

[11] Tout Ou Rien
[12] Normalement Ouvert

II.8 Application

Dans cette application on a, tout d'abord, établi la communication entre le logiciel de simulation FlexSim et l'API avec une interface de communication Java-OPC. FlexSim étant programmé comme serveur et Java comme client. Ensuite, on démarre la simulation à partir de FlexSim (serveur à l'écoute) et on lance aussitôt le client (Java), la communication s'établit instantanément. Puis, FlexSim envoie des messages à Java (échange d'informations par socket), pour les traiter et les envoyer à l'API grâce au protocole OPC qui reçoit des tags depuis NetBeans, ce dernier permet de relier l'application Java avec l'API afin d'exécuter les actions liées aux messages. Enfin, L'API envoie ces actions sous forme de signaux au serveur OPC qui va transmettre les informations à Java pour déclencher les actionneurs du système (FlexSim) à titre d'exemple : démarrage ou arrêt d'un ou plusieurs convoyeurs.

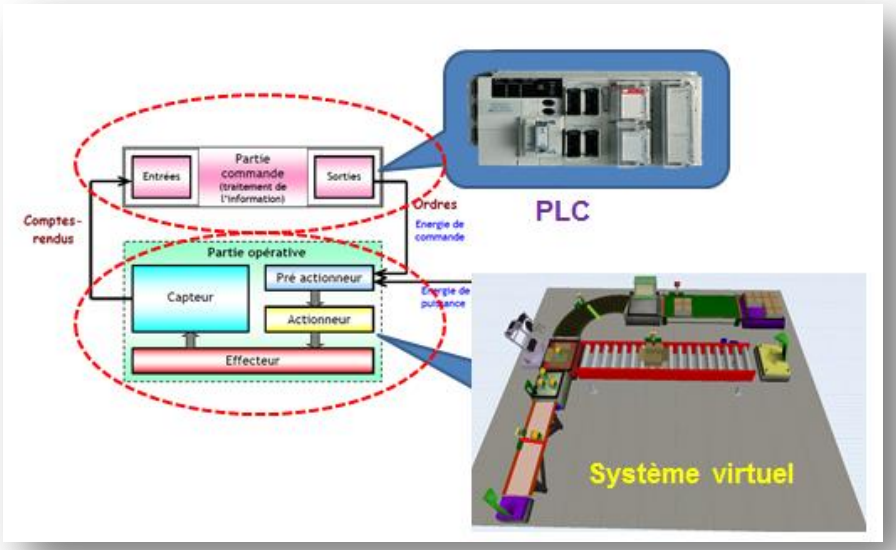


Figure II-11. Simulation Hardware-in-The-Loop du système de production

Résultat

Après la simulation Hardware-in-The-Loop du système précédent, on constate que la simulation s'est bien déroulée et que la communication client-serveur a bien fonctionnée.

Conclusion

Comme cité précédemment, on veut des simulations qui soient le plus proche possible de la réalité. Avec Java-OPC comme intermédiaire, la communication entre le système simulé et l'API est une communication « Soft », c'est-à-dire que les entrées/sorties de l'automate ne sont pas

câblés. Pour une meilleure simulation et afin d'atteindre les buts pédagogiques fixés par ce projet, il a été décidé de changer l'architecture de l'application en remplaçant le serveur OPC par une carte Arduino.

Introduction

Dans l'optique de bonifier la simulation proposé par ce travail et de permettre aux étudiants de bien assimiler les bases de l'automatique, une carte Arduino va remplacer le serveur OPC, ainsi, on aura la possibilité de câbler les entrées/sorties de l'automate, chose qui n'était pas possible précédemment par le fait que Java-OPC soient des logiciels donc il n'y a pas de communication physique avec l'automate programmable. En effet, l'idée de base ne va pas changer, elle passera d'un intermédiaire 100% logiciel (Java-OPC) à une configuration hybride : Java-Arduino.

III.1 La carte Arduino

Le projet Arduino est issu d'une équipe d'enseignants et d'étudiants de l'école d'« Interaction Design Institute d'Ivrea[13] » située, comme son nom l'indique, dans la ville italienne d'Ivrea. Ils ont rencontré un problème majeur à cette période (avant 2003 - 2004) : les outils nécessaires à la création de projets d'interactivité étaient complexes et couteux, ce qui rendaient difficiles le développement, par ces derniers, de nombreux projets et ceci ralentissait la mise en œuvre concrète de leur apprentissage.

Les cartes Arduino sont des circuits électroniques composés de connecteurs divers reliés à un microcontrôleur programmable. Les autres composants de la carte sont dédiés à la gestion et à la distribution de l'alimentation électrique ou à la communication, nous pouvons en citer : ports d'entrée et sortie qui nous serviront à commander des appareils extérieurs ou à recevoir des informations, port USB.

Sans tout connaître ni tout comprendre de l'électronique, l'environnement matériel et logiciel des cartes Arduino, permet à l'utilisateur de formuler ses projets par l'expérimentation directe sur la carte en s'appuyant, par ailleurs, sur plusieurs supports disponibles en ligne [23].

[13] L'école « Interaction Design Institute Ivrea » (IDII) est aujourd'hui située à Copenhague sous le nom de « Copenhagen Institute of Interaction Design »

III.1.1 Logiciel

L'environnement de programmation Arduino (IDE[14] en anglais) est une application écrite en Java inspirée du langage Processing[15]. L'IDE permet notamment d'écrire, de modifier un programme et de le convertir en une série d'instructions compréhensibles pour la carte.

Le logiciel Arduino offre une multitude d'exemples avec des commentaires et selon l'utilité de la carte dans notre projet : on peut trouver des exemples sur l'Ethernet Shield (qu'on va aborder par la suite), sur la programmation du module GSM, des programmes simples comme allumer et éteindre des LEDs. Cela a pour but d'accélérer la prise en main avec la carte [23].

La figure suivante montre la fenêtre de programmation du logiciel Arduino :

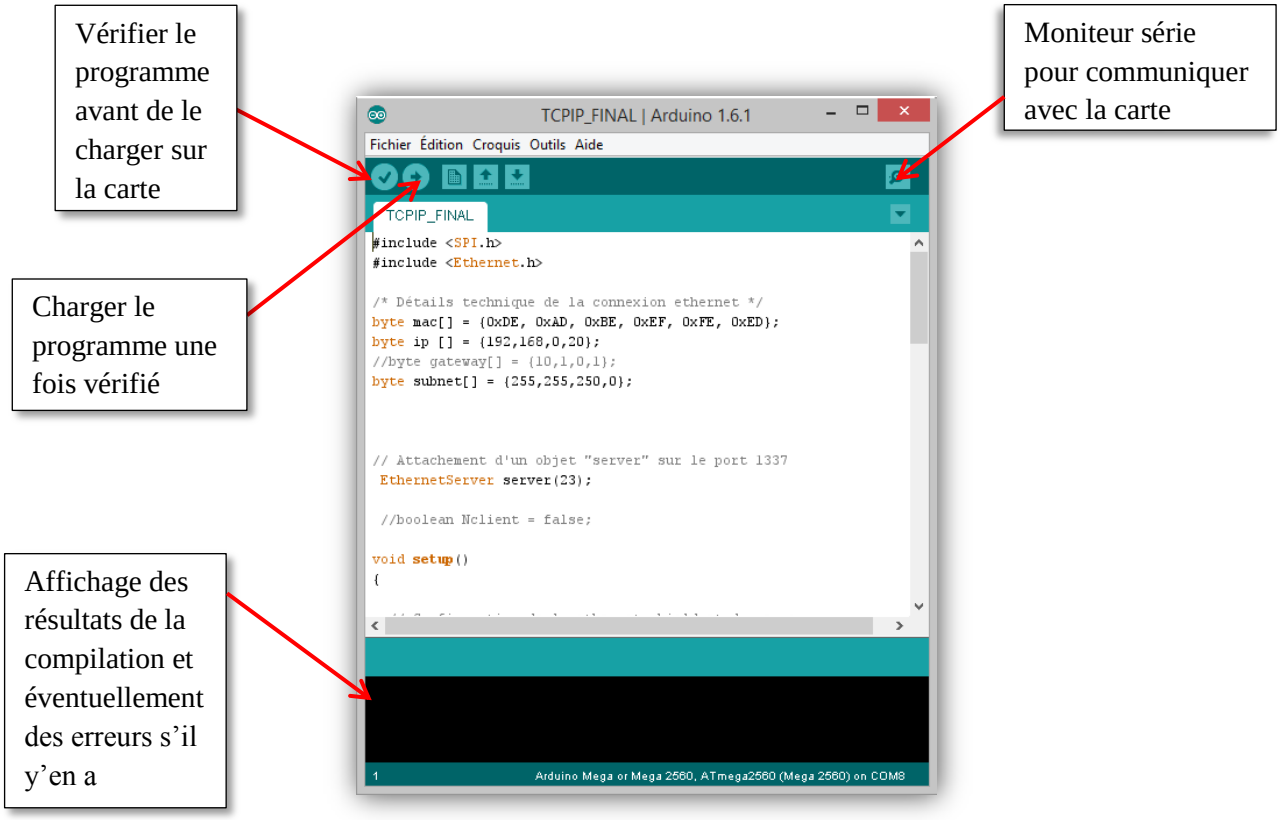


Figure III-1. Fenêtre de programmation du logiciel Arduino

III.1.2 Langage Arduino

Le langage Arduino a été inspiré des langages informatiques C et C++, néanmoins, il se distingue de ces deux langages classiques par le fait que ces commandes sont adaptées aux

[14] Integrated Development Environment
[15] Un environnement de développement libre de la bibliothèque java

besoins de la carte, qui consistent essentiellement à communiquer avec ses différents connecteurs.

III.1.3 Arduino Mega 2560

La carte Arduino Mega 2560 est une carte à microcontrôleur basée sur un ATmega2560[16]. Elle se compose de : de 54 broches numériques d'entrées/sorties dont 14 peuvent être utilisées en sorties PWM, de 16 entrées analogiques qui peuvent également être utilisées en broches entrées/sorties numériques, de 4 UART[17] (port série matériel), d'un quartz[18] 16Mhz, d'une connexion USB, d'un connecteur d'alimentation jack, d'un connecteur ICSP (programmation "in-circuit"), et d'un bouton de réinitialisation (reset) [24].

Pour pouvoir l'utiliser et se lancer, il suffit tout simplement de connecter la Mega 2560 à un ordinateur à l'aide d'un câble USB, de charger le programme à l'aide de l'interface du logiciel Arduino et là voilà fin prête pour accomplir une tache précise.

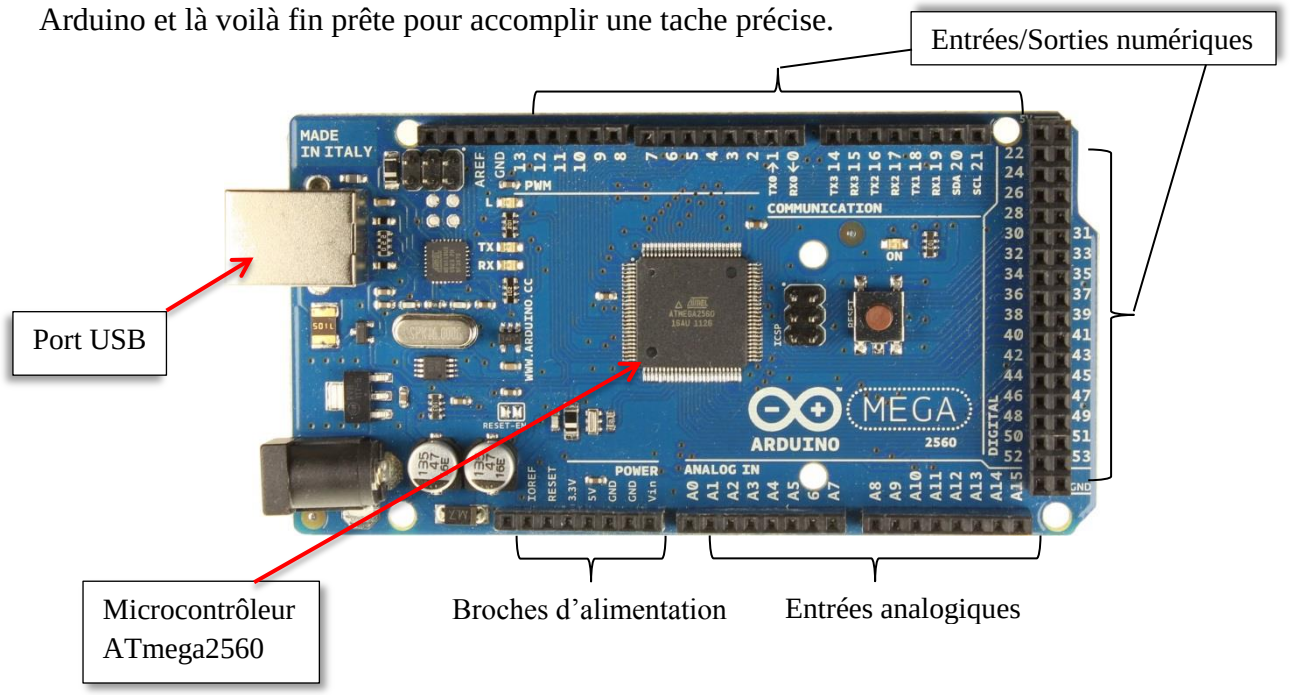


Figure I-2. Carte Arduino Mega 2560

III.2 Ethernet Shield W5100

La plateforme de développement Arduino bénéficie d'un grand nombre de modules, l'un de ses modules : l'Ethernet Shield W5100. Il permet d'ajouter une connexion réseau ethernet. Il

[16] Microcontrôleur d'Atmel®
[17] Universal Asynchronous Receiver Transmitter émetteur-récepteur asynchrone universel)
[18] Un composant qui possède comme propriété utile d'osciller à une fréquence stable lorsqu'il est stimulé électriquement

est basé sur le circuit intégré WIZnet W5100 qui fournit une pile réseau (IP) adapté, à la fois, au protocole de télécommunication TCP[19] et UDP[20]. Il supporte jusqu'à quatre connexions simultanées. Il suffit d'utiliser la librairie Ethernet pour écrire des programmes qui se connectent à internet. On le connecte à une carte Arduino grâce à ses longues broches qui dépassent du circuit imprimé. Ainsi le brochage de la carte Arduino n'est pas modifié et permet d'enficher un autre module par-dessus et laisse l'accès aux broches de la carte Arduino.

La dernière version de ce module ajoute un connecteur pour carte SD, qui pourra être utilisé pour stocker des fichiers afin de les envoyer sur le réseau. [25]

Voici une image de l'Ethernet Shield W5100 :

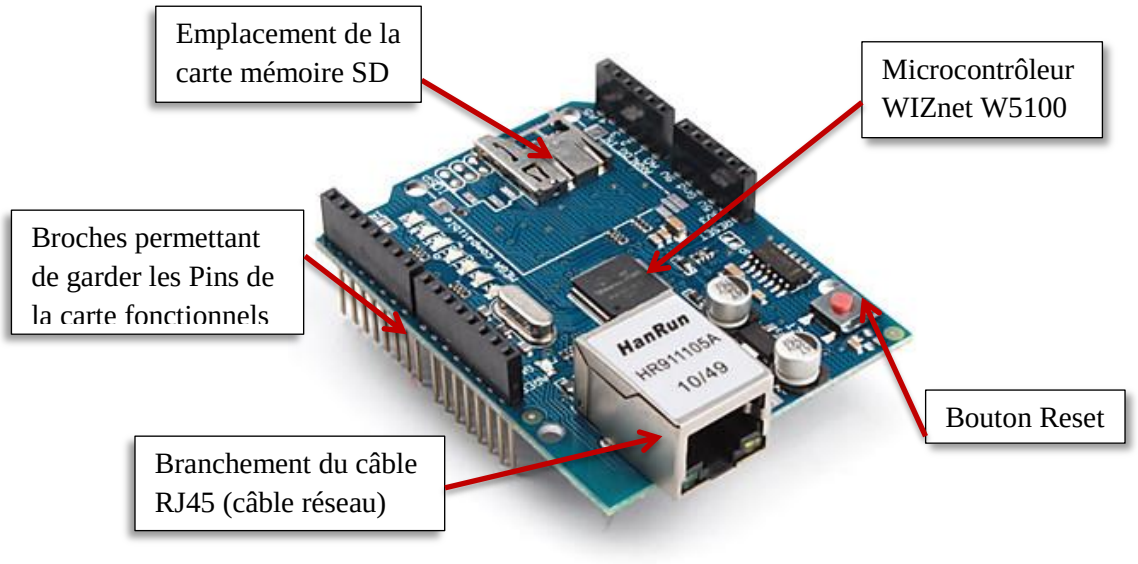


Figure I-3. Ethernet Shield W5100

La carte Arudino Mega 2560 ne prenant pas en charge le protocole TCP/IP, il fallait faire appel à l'Ethernet Shield d'Arduino pour pouvoir établir une communication réseau dont voici le résultat de l'embranchement du Shield au-dessus de la Mega 2560 :

[19] Transmission Control Protocol, c'est un protocole de transport utilisé par internet
[20] User Datagram Protocol, est un des principaux protocoles de télécommunication utilisés par Internet

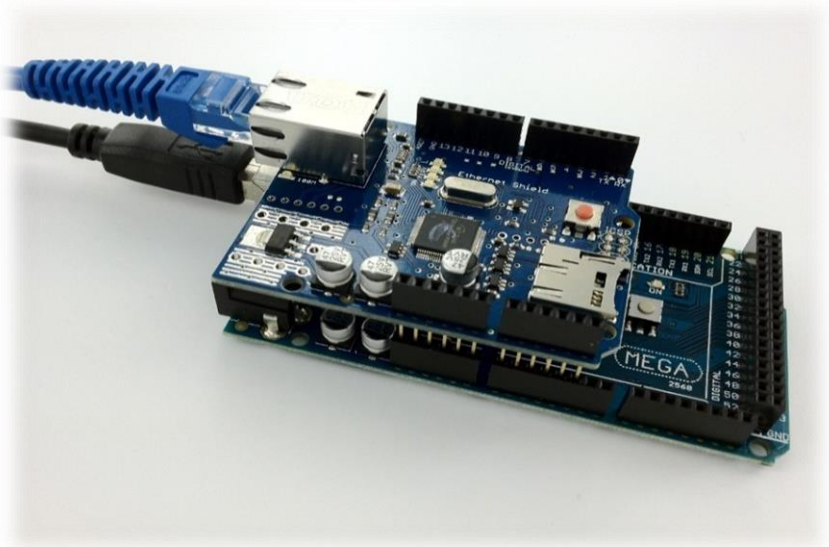


Figure I-4. Embranchement de l'Ethernet Shield au-dessus de la carte Mega 2560

Comme mentionné dans le chapitre précédent, la carte Arduino avec l'Etchernet Shield (Figure III-4) prendront la place du serveur OPC, c'est-à-dire que le principe restera le même : établir une communication entre FlexSim et l'API. Ce qui va changer, par contre, c'est la communication entre la carte Arduino, qui a remplacé le serveur OPC, et l'automate. Dans le cas où c'était Java-OPC qui faisait le lien entre l'API et FlexSim, c'était OPC qui assurait l'échange d'informations avec l'automate. L'utilisation de la carte Arduino présente un avantage non pas négligeable, comme nous l'avions ébauché dans l'introduction de ce chapitre : elle permet, entre autres, de câbler les entrées/sorties de l'API ce qui représente un atout considérable étant donné que l'étudiant ou l'industriel bénéficiera de la totalité des fonctionnalités de l'automate en matière d'entrées/sorties.

Cela dit, on se retrouve face à un obstacle de taille : la tension des entrées/sorties de la carte Arduino et celle de L'API sont différentes, elle est de l'ordre de 5V pour la carte et de 24V pour l'API. Câbler directement les entrées/sorties de l'API avec celles de la carte, est tout simplement impossible. C'est pourquoi, on a élaboré un circuit imprimé permettant le passage d'information entre l'API et la carte Arduino réalisé sur le logiciel Eagle[21].

[21] Eagle est un logiciel permettant de créer des schémas électriques et de réaliser des circuits imprimés de ces schémas

III.3 Circuit imprimé à base d'optocoupleur

La carte (Figure III-5) est une carte à base d'optocoupleur, ce composant électronique permet, entre autres, la communication entre deux circuits fonctionnants avec des tensions différentes tout en assurant l'isolation galvanique entre les deux.

Le rôle des résistances est de protégé l'optocoupleur, car il accepte en entrée, un courant limité (environs 10 mA), c'est avec cette référence que se fera le calcul des résistances pour chacune des deux tensions : 5 et 24 V. Des LEDs sont utilisées pour la vérification du passage de courant.

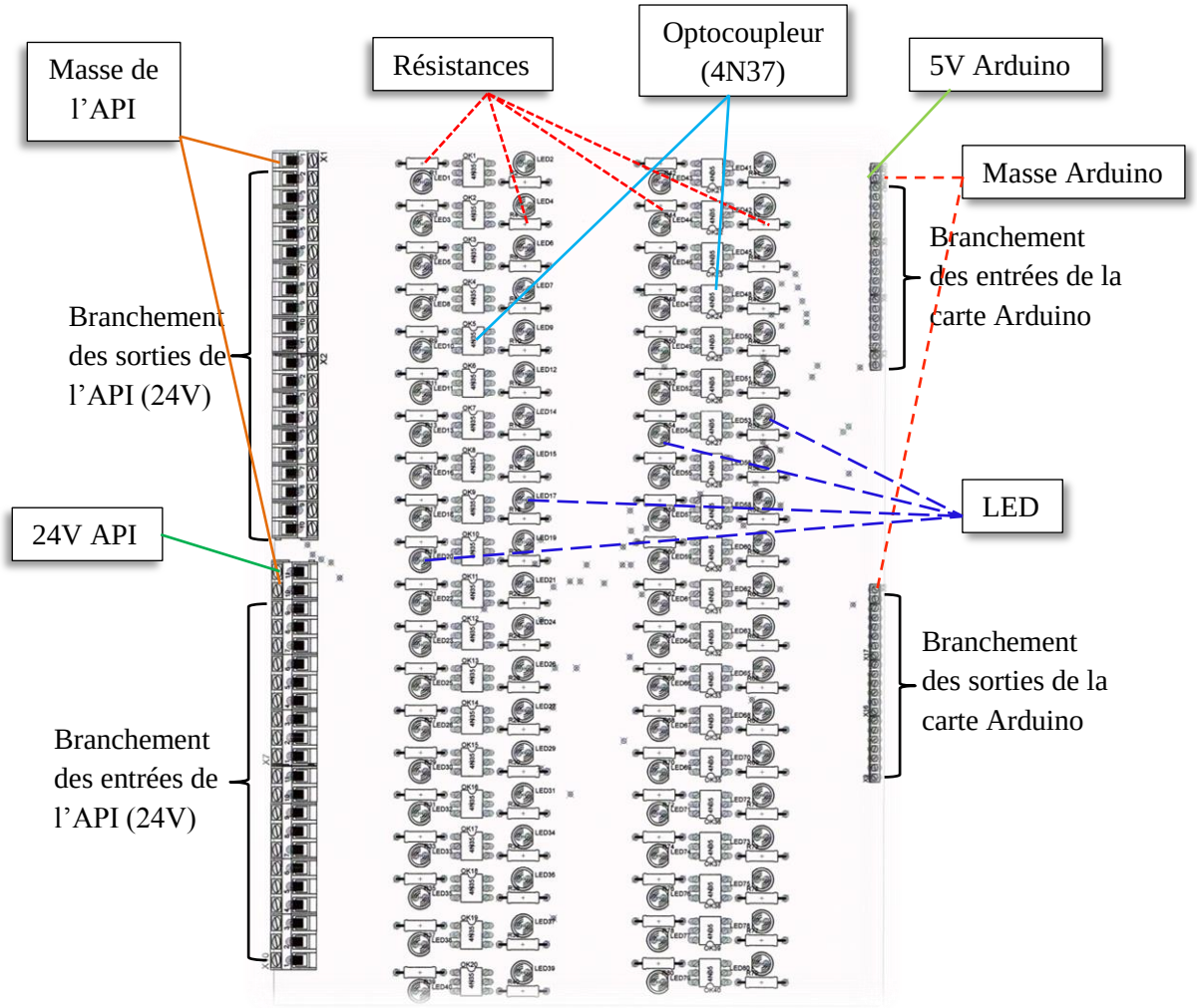


Figure I-5. Carte intermédiaire entre la carte Arduino et l'API à base d'Optocoupleurs

III.4 L'optocoupleur et son principe de fonctionnement

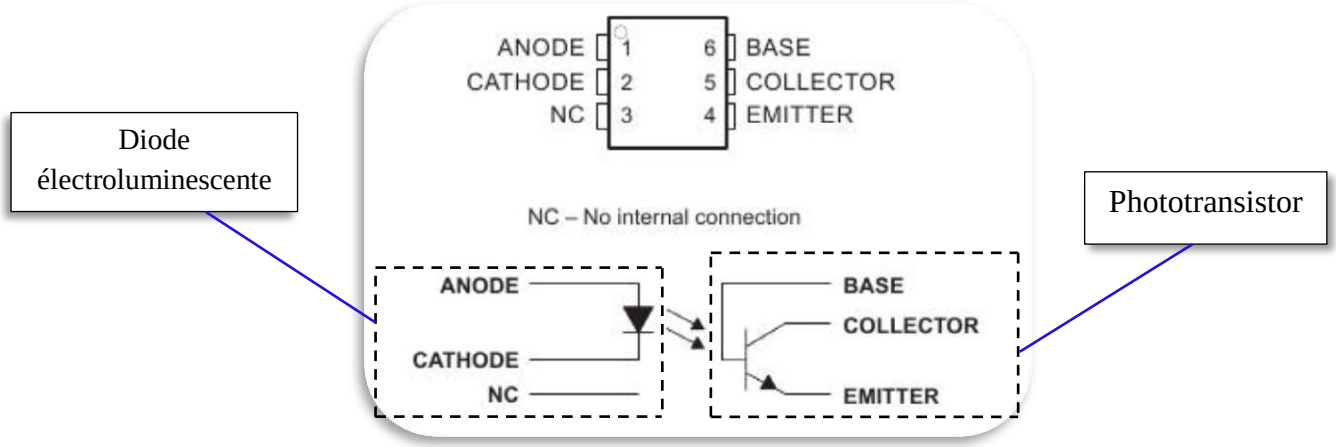


Figure I-6. Schéma d'un Optocoupleur

Sur la Figure III-6 on peut voir que l'optocoupleur inclut deux composants connus : une diode électroluminescente qui émet de la lumière infrarouge (émission spontanée) lorsqu'elle est soumise à une polarisation directe et d'un phototransistor composé de 3 zones : l'émetteur, la base et le collecteur. Lorsqu'un courant passe dans la diode, et à partir d'une certaine tension (en général 1,5 Volt), elle s'allume et le courant passe du collecteur vers l'émetteur à condition que la base reçoive de la lumière infrarouge et non pas un courant.

III.5 Système d'emballage avec Arduino

Toujours dans l'optique d'améliorer la simulation proposée initialement, une simulation du système d'emballage, vu dans le précédent chapitre (Figure II-8), avec la seconde approche a été établi en utilisant comme intermédiaire, entre l'automate et le système simulé, le duo Java-Arduino.

Dans cette nouvelle configuration Java est un serveur, FelxSim le client n°1 et Arduino le client n°2. Le serveur Java a pour but de gérer les messages transitant entre les deux clients, en d'autres termes : il assure la communication entre FlexSim et la carte Arduino. Cette dernière, reçoit, à travers le réseau, les messages envoyés par FlexSim : il s'agit de l'état des capteurs, ensuite, elle les converti en signaux qu'elle envoie à l'automate à travers le circuit imprimé (Figure III-5). L'API activera ou désactivera ses sorties en fonction de ses signaux. À la fin, la carte Arduino intercepte ces derniers (toujours à travers le circuit imprimé) pour les convertir en messages qu'elle enverra à FlexSim qui changera d'état en fonction des messages reçus.

Voici le système d'emballage (Figure III-7) avec lequel on a établi une simulation HIL avec Java-Arduino comme interface de communication :

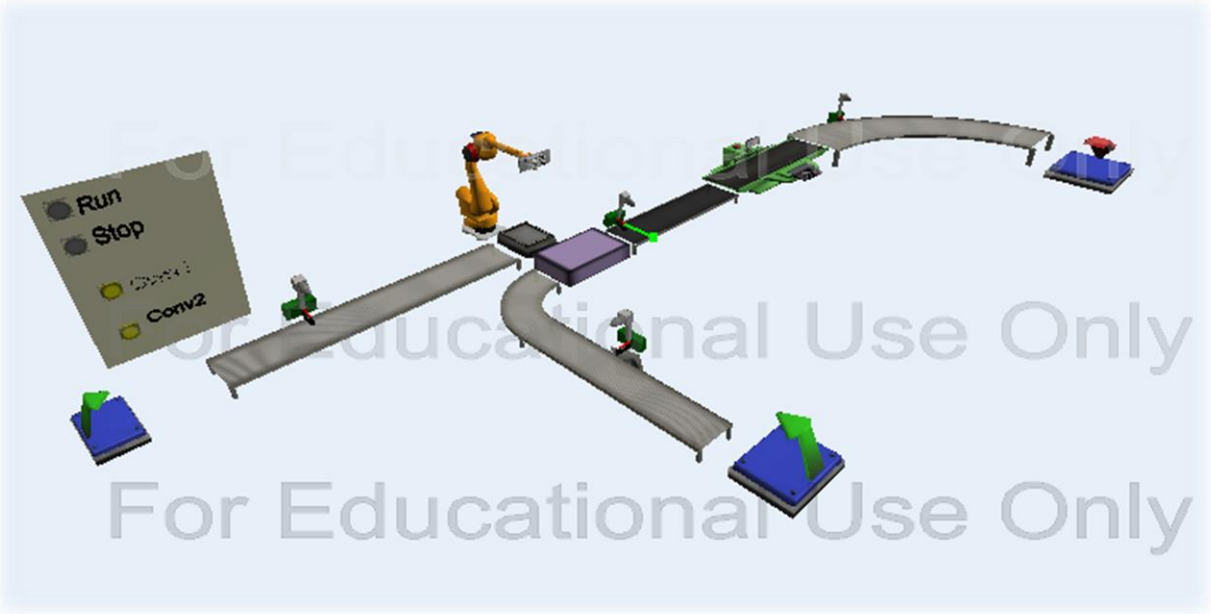


Figure I-7. Système d'emballage simulé avec Java-Arduino

Résultat

La commande du système d'emballage par l'automate programmable a fonctionné avec succès, ainsi que toutes les communications mises en œuvre. C'est le cas de l'envoie et réception des messages entre FlexSim et la carte Arduino en plus de la communication électronique entre la carte Arduino et l'API.

Conclusion

Dans ce chapitre, la carte Arduino a remplacé le serveur OPC pour améliorer, d'un point de vue didactique, la simulation Hardware-in-The-Loop des systèmes proposés. Il a été constaté qu'un module spécifique à la carte Arduino Mega 2560 était nécessaire pour la connecter au réseau. Ainsi, nous avons bénéficié de cette connexion afin de communiquer avec FlexSim en plus du câblage des entrées/sorties numériques pour l'échange d'informations avec l'automate.

On conclut donc, que la seconde configuration a fonctionné avec succès.

Introduction

Le dernier chapitre englobe les résultats de la simulation de deux systèmes plus complexes que celui vu dans le chapitre précédent.

Pour le choix et la conception des systèmes à simuler, on s'est appuyé sur des systèmes réels ainsi, nous aurons touché à une large gamme de systèmes que peuvent croiser les étudiants dans l'industrie.

IV.1 Système de production et d'assemblage de machines à café

Le premier système est un système de production de machines à café, pour être dans l'esprit d'un système de production industriel.

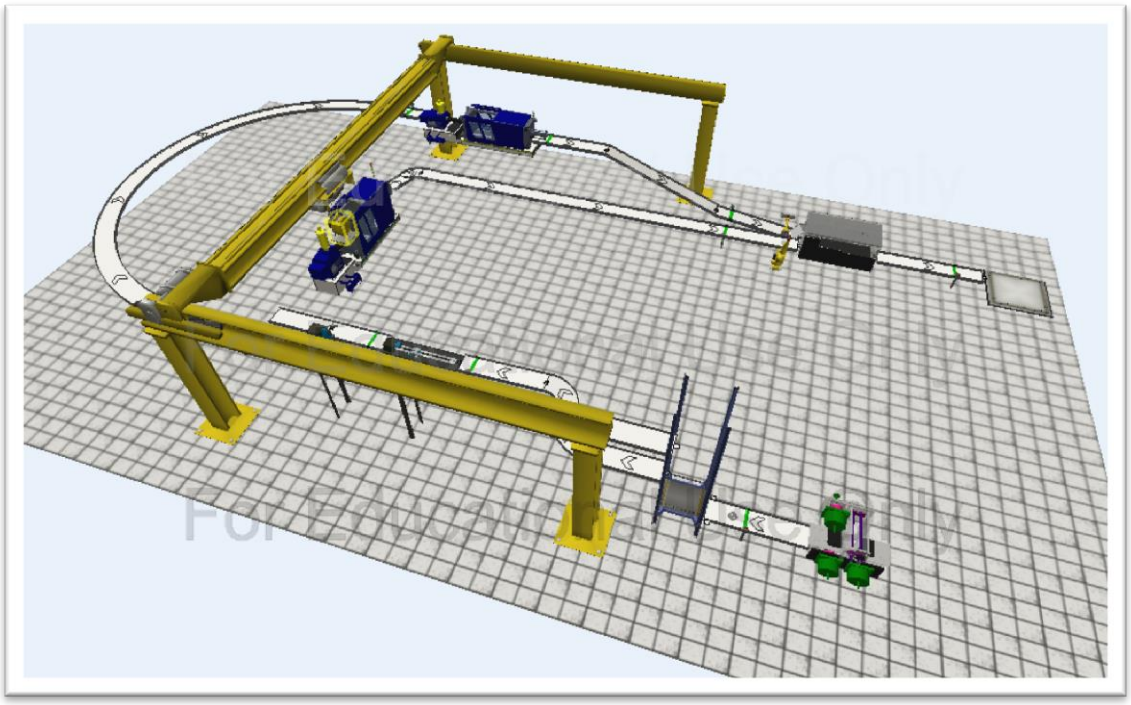


Figure IV-1. Système de production et d'assemblage de machine à café sur FlexSim

La Figure IV-1 montre, en effet, un système de production simulé sur FlexSim dont voici le cahier des charges :

IV.1.1 Cahiers de charges

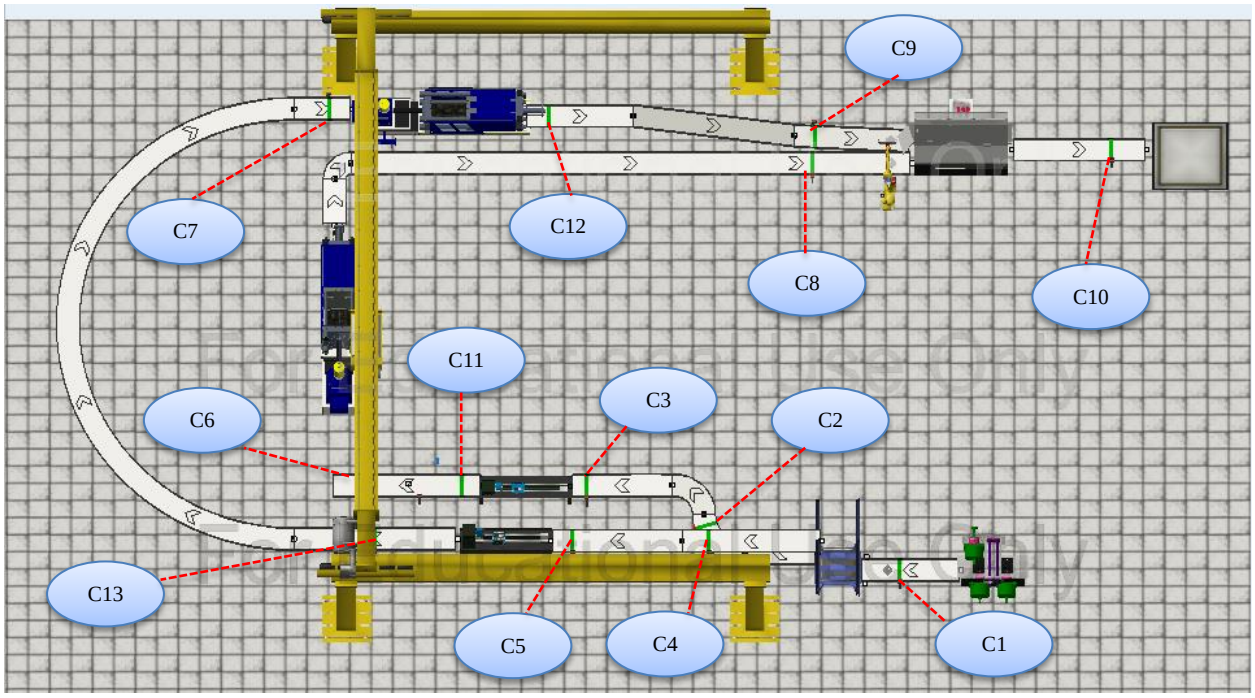


Figure IV-2. Emplacement des capteurs

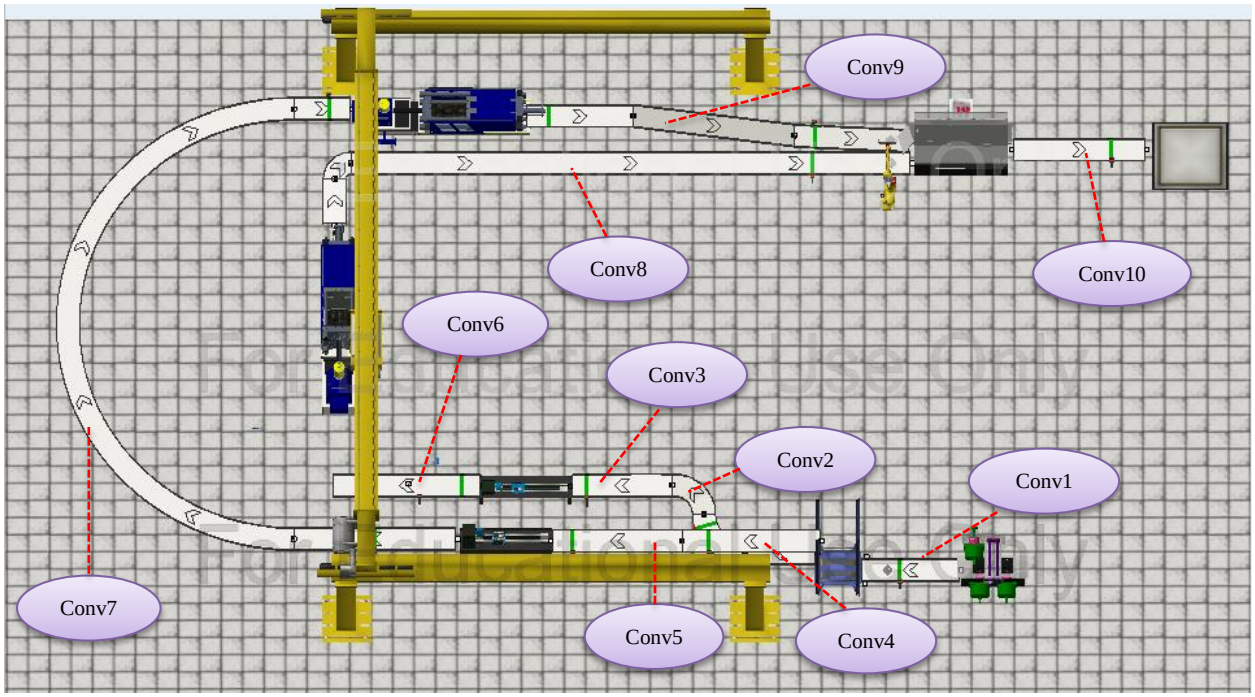


Figure IV-3. Emplacement des convoyeurs

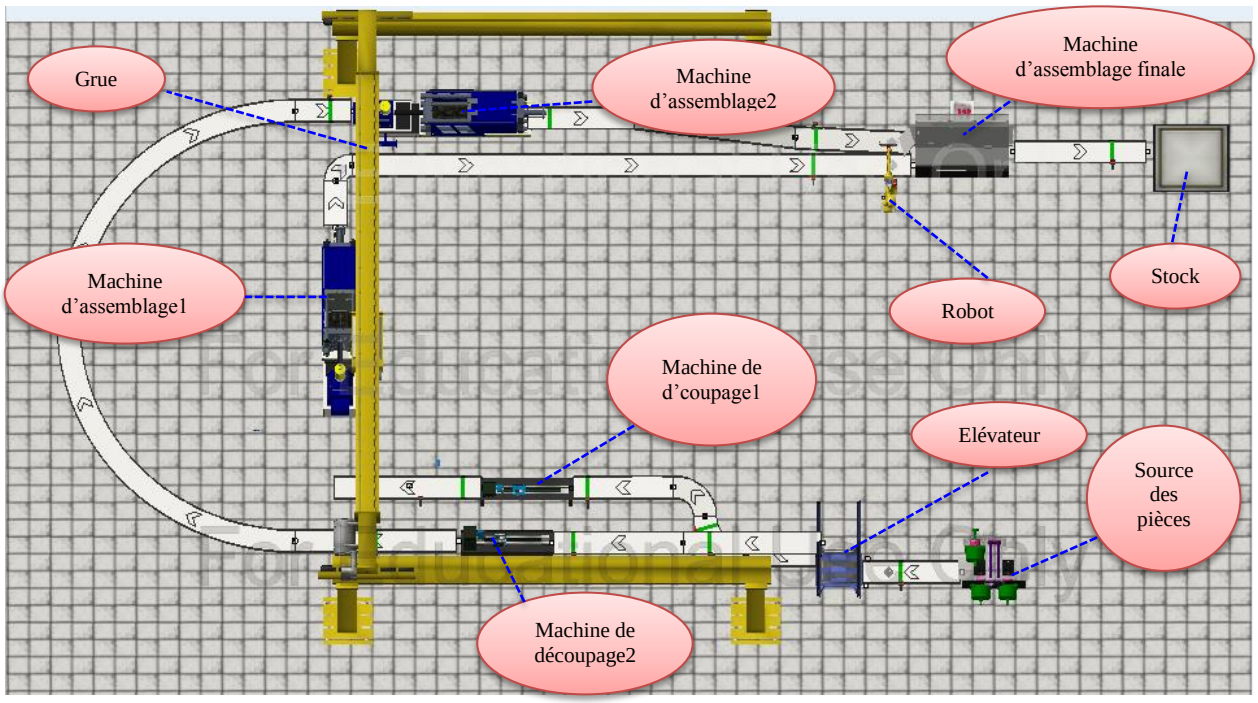


Figure IV-4. Emplacement des machines et des exécuteurs de tâches

Le système considéré comporte dix convoyeurs (Conv1→Conv10), 13 capteurs (C1→C13), un élévateur, une grue, un robot, une machine d'assemblage, deux machines de production, deux machines de découpage, une source donnant la matière première.

Le capteur 'C1' détecte le type du produit : si c'est une barre en métal il l'oriente vers le 'Conv2' et si c'est un cylindre métallique il l'oriente vers le 'Conv4'. S'il y a présence d'objet au niveau des capteurs 'C2' et 'C3' le convoyeur 'Conv1' s'arrête. Le capteur 'C2' vérifie s'il y a présence de produit au niveau du capteur 'C3' si c'est le cas le convoyeur 'Conv2' s'arrête. Le capteur 'C3' incrémente un compteur 'Compt1'qui compte une seule unité, et qui lorsqu'il est activé fait arrêter le convoyeur 'Conv3' car la 'Machine de découpage1' ne peut traiter qu'un seul produit à la fois. Le capteur 'C11' est le reset de ce compteur 'Compt1', les capteurs 'C4', 'C5' et 'C13' ont les mêmes rôles que les capteurs 'C2','C3' et 'C11' respectivement pour les convoyeurs 'Conv4' et 'Conv5', pour la 'Machine de découpage2' sachant que 'C5' incrémente un compteur 'Compt2' et c'est 'C13' fait sont Reset. Une fois les pièces arrivées au niveau des machines de découpages 1 et 2 ('Machine de découpage1' fait le découpage de la barre en métal et la deuxième machine celui des cylindres métalliques) la barre va être découpée en 4 morceaux qui vont être déplacés vers la machine de production grâce à une grue qui est actionnée par la présence des pièces découpées pour la production des machines à café préfabriquées au niveau du capteur 'C6'. La pièce cylindrique va transiter par le convoyeur 'Conv7' pour être découpée en 2 morceaux pour produire des tasses, le capteur 'C7' incrémente un compteur avant l'entrée

de machine de production des tasses et le capteur ‘C12’ est son Reset. Ensuite les tasses et les machines vont être dirigées vers la machine d’assemblage, dès qu’il y a une présence des tasses et des machines au niveau des capteurs ‘C9’ et ‘C8’ respectivement le robot s’actionne en mettant les deux produit dans la machine d’assemblage qui va nous donner le produit final.

Après l’obtention du produit fini, les machines de café vont être transportées au stock en passant par un capteur ‘C10’ qui va incrémenter un compteur d’arrivage des machines à café ‘Compt3’, dès qu’il atteindra la valeur de 30 unités il va provoquer l’arrêt total du système.

IV.1.2 Schéma Grafcet du système de production

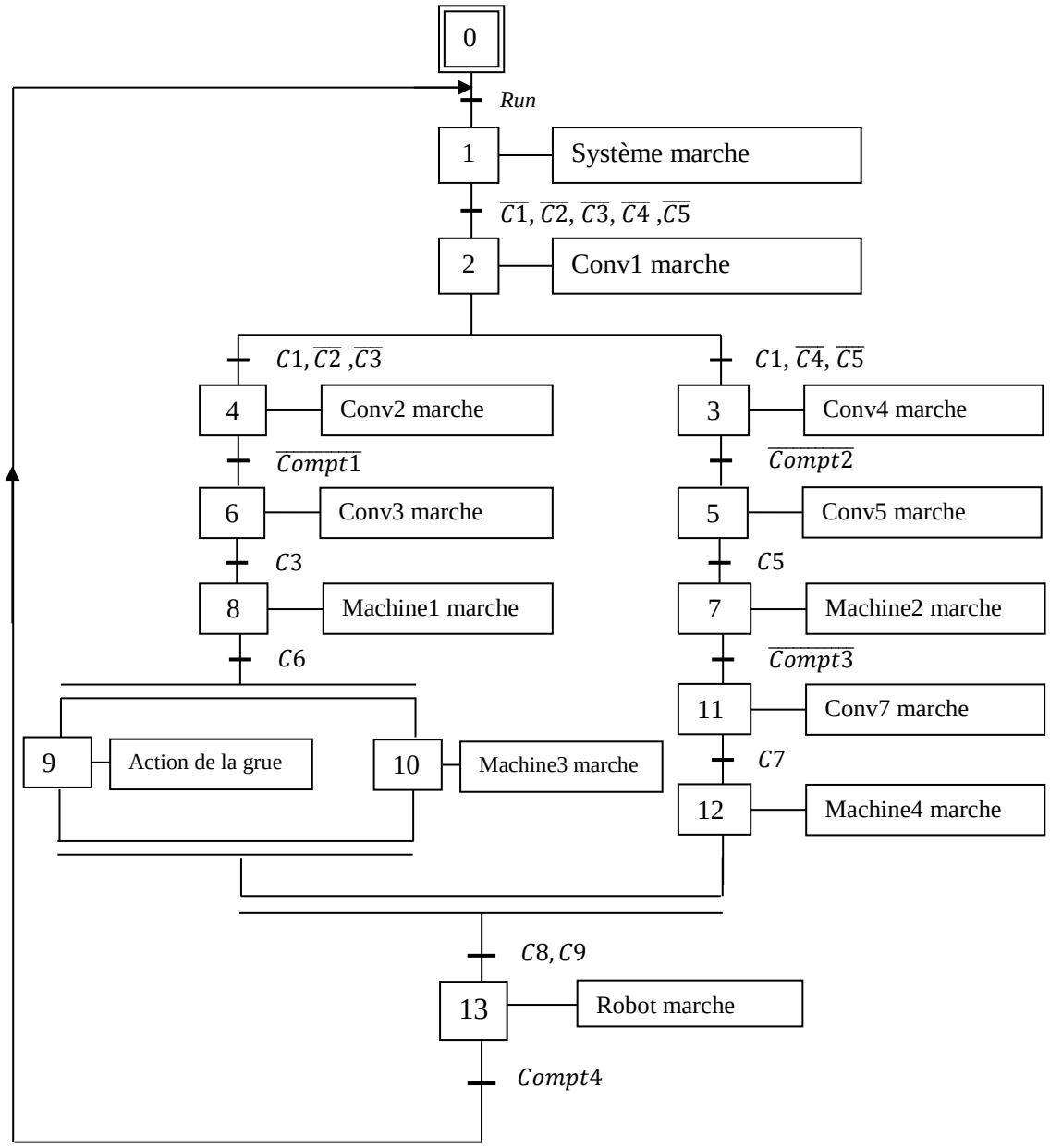


Figure IV-5. Grafcet du système de production

IV.2 Système d'emballage de boites de lait

Le second système est un système d'emballage plus complet que celui traité précédemment (chapitre 3).

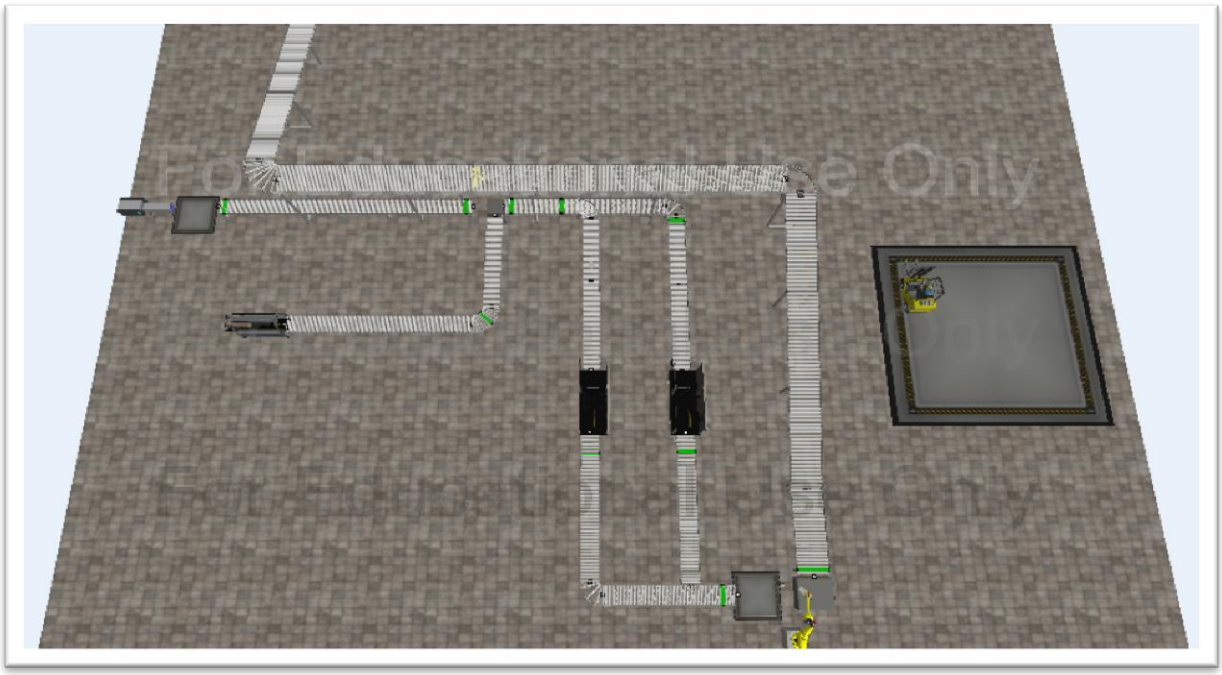


Figure IV-6. Système d'emballage des boites de lait sur FlexSim

IV.2.1 Cahiers de charges du système d'emballage de boites de lait

Comme vu sur les figures IV-7 et IV-8 : le convoyeur en haut est le convoyeur principal et celui d'en bas est le convoyeur secondaire (de secours). On veut éviter la surcharge du convoyeur principal (il peut contenir jusqu'à 5 boites maximum). Dès que le convoyeur principal est surchargé, la prochaine boite qui arrive doit être dirigé vers le convoyeur secondaire.

Figure IV-7. Surcharge du convoyeur de la chaine principale

Figure IV-8. Orientation des boites vers la chaine secondaire

Voici (Figure IV-9) l'emplacement des capteurs sur les convoyeurs du système :

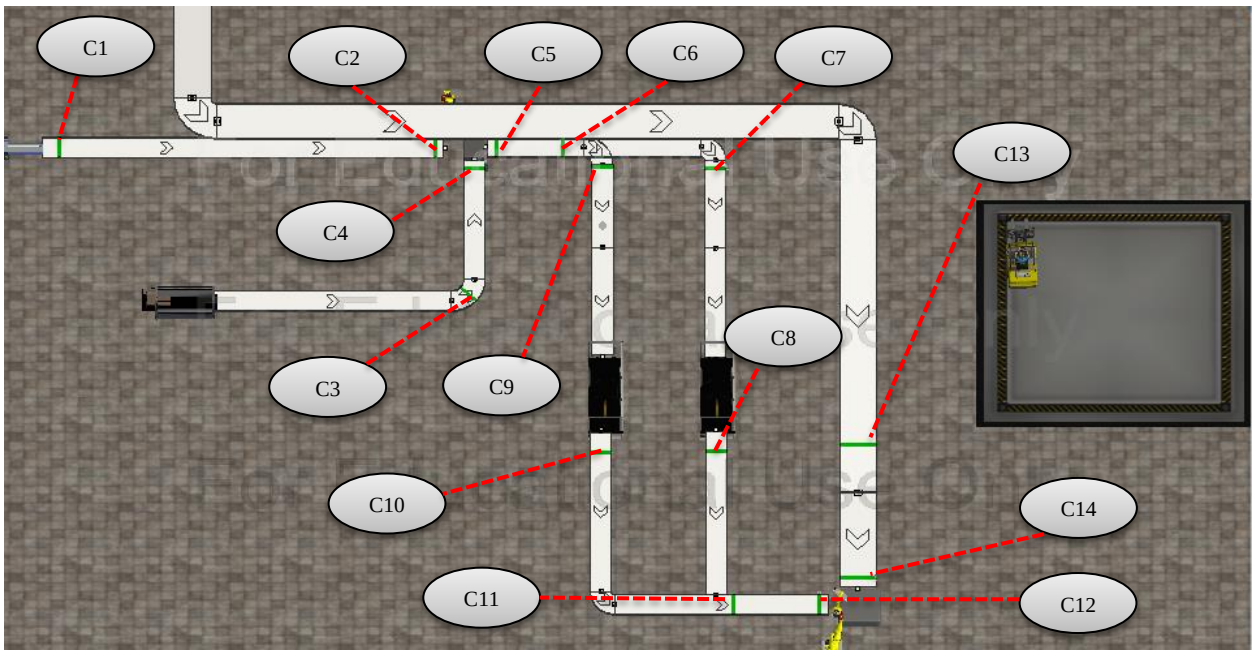


Figure IV-9. Emplacement des capteurs sur la chaine d'emballage du second système

Voici (Figure IV-10) l'emplacement des machines et des convoyeurs du système :

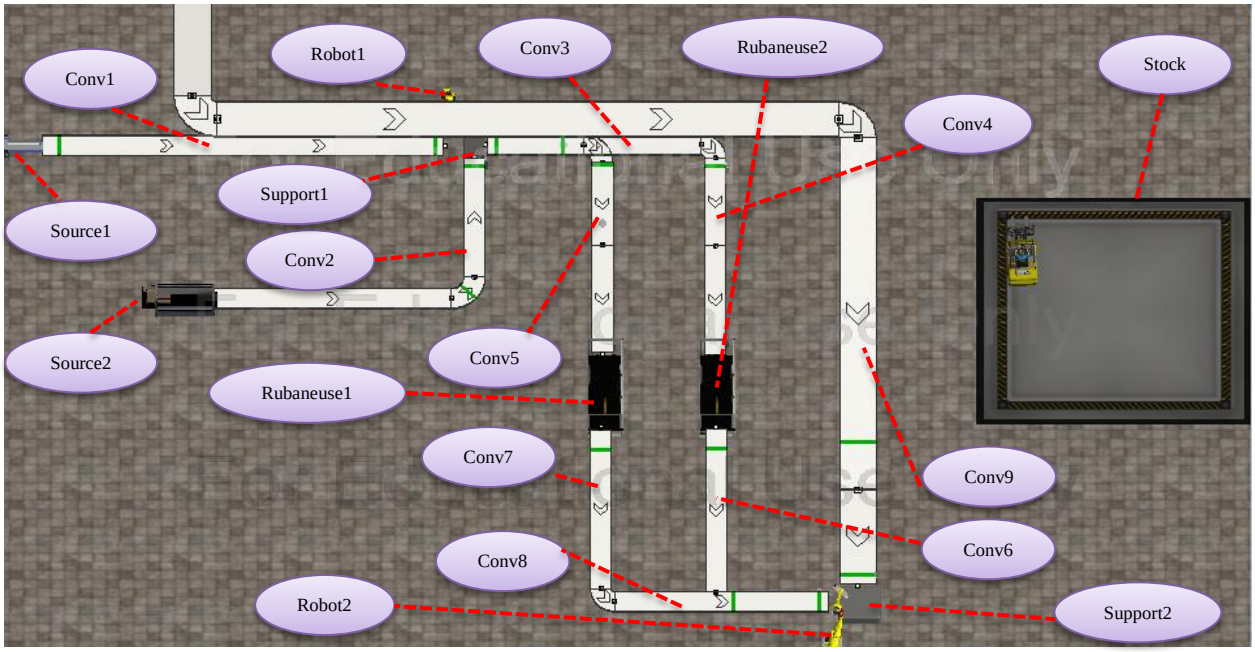


Figure IV-10. Emplacement des convoyeurs et des machines dans le système d'emballage

D'après les figures IV-9 et IV-10 l'arrivée des boites de lait se fait par le biais de la machine qui leur pose un bouchon, à leur sortie, les boites se retrouvent sur le convoyeur

‘Conv1’ puis elles passent par le capteur ‘C1’ qui incrémente un compteur ‘Compt1’ qui compte jusqu’à 8, puis elles passent par un second capteur ‘C2’ qui décrémente le compteur. Ainsi, la surcharge du convoyeur est évitée. Dès que la valeur du compteur est supérieure ou égale à 8 le convoyeur ‘Conv1’ s’arrête. En parallèle, l’arrivée des boites en carton, qui serviront à l’emballage des boites de lait, se fait par le biais des convoyeurs ‘Conv2’ et ‘Conv3’ qui forment un ‘L’, au milieu de ces deux convoyeur il y a un le capteur ‘C3’ dont on mentionnera l’utilité par la suite. C’est le même principe qu’on a appliqué au niveau du convoyeur ‘Conv2’ ainsi les capteurs ‘C3’ et ‘C4’ incrémente et décrémente respectivement un compteur ‘Compt2’ qui compte jusqu’à 1 donc tant qu’il y a une boite en attente de produit au niveau du support le convoyeur ‘Conv2’ s’arrête.

Les boites de lait et les boites en carton (boites d’emballage) vont s’intersecter au niveau d’un support ‘Support1’ pour l’emballage du produit et ce par 4 unités, et c’est un robot ‘Robot1’ qui est responsable de cette action et qui sera actionné par le passage des boites au niveau du capteur ‘C2’ sans oublier que s’il y a présence d’une boite en carton au niveau du ‘Support1’, la prochaine boite qui arrivera, va arrêter les convoyeurs ‘Conv2’ et ‘Conv3’ et ce dès qu’elle sera au niveau du capteur ‘C3’.

Dès la sortie des boites en carton contenant le produit, elles se dirigent vers les machines qui vont les scotcher et cela par l’intermédiaire du convoyeur ‘Conv4’ pour la ‘Rubaneuse1’ et du convoyeurs ‘Conv5’ pour la ‘Rubaneuse2’. Dès que le convoyeur ‘Conv4’ contient 5 boites, elles sont redirigées vers le convoyeur ‘Conv5’ et cela grâce à l’incrémentation et décrémentation d’un compteur ‘Compt3’ par les capteurs ‘C6’ et ‘C7’ respectivement. De même pour la chaine secondaire, dite de secours, si le convoyeur ‘Conv5’ contient 5 boites, cela provoquera l’arrêt total du système. On a utilisé le même principe que pour la chaine principale c’est-à-dire qu’on a eu recours à un compteur ‘Compt4’ qui sera incrémenté et décrémenter respectivement par les capteurs ‘C6’ et ‘C9’. Quand les rubaneuses contiennent un produit elles envoient un signal à l’API pour arrêter les convoyeurs ‘Conv4’ et ‘Conv5’, à leur sortie, les boites fermées réactionnent les deux convoyeurs 4 et 5 grâce aux capteurs ‘C8’ et ‘C10’. Après cela, les boites transitent vers le convoyeur ‘Conv8’ en passant par un capteur de présence ‘C11’ qui incrémente un compteur ‘Compt6’ qui compte le nombre de boites que contient le convoyeur ‘Conv8’ (7 unités), ce compteur est décrémenté par le capteur ‘C12’ qui actionne, avec la présence des palettes au même temps, le robot ‘Robot2’. Le même principe est appliqué pour l’arrivage des palettes avec le compteur ‘Compt7’ qui est incrémenté et décrémenté par les capteurs ‘C13’ et ‘C14’.

À la fin, dès qu'une palette contiendra 8 boites, un transporteur la conduira vers le stock principal.

IV.2.2 Schéma Grafcet du système d'emballage de boites de lait

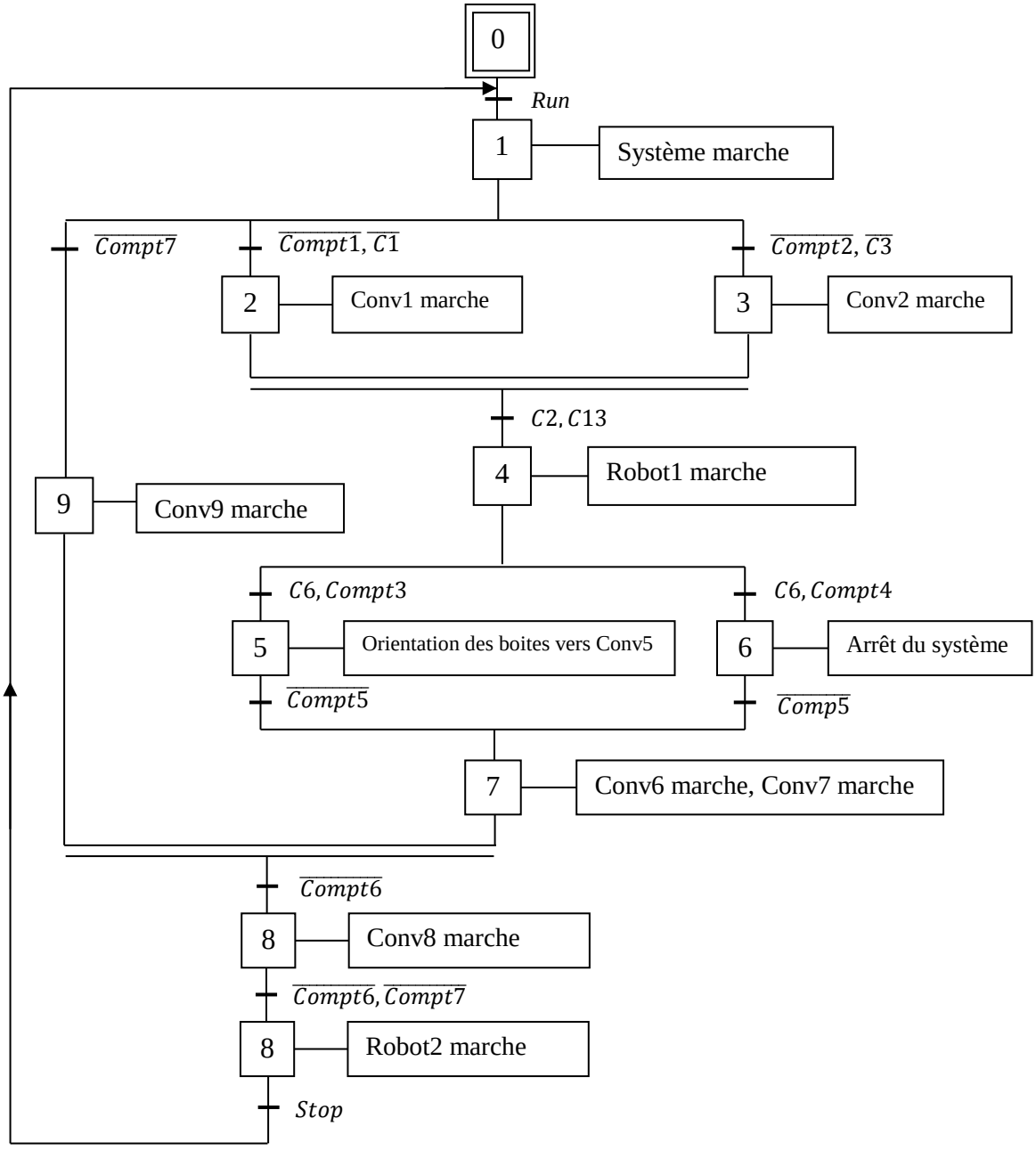


Figure IV-11. Grafcet du système d'emballage

IV.3 Application

Pour la simulation des deux systèmes traités dans ce chapitre, on a un système simulé sous le logiciel FlexSim qui est connecté autant que client n°1 avec le serveur Java. La carte Arduino

quant à elle, est connectée comme client n°2 au même serveur par l'intermédiaire de son Ethernet Shield. Le serveur Java assure la communication réseau entre les deux clients.

Le principe de fonctionnement de l'application est le suivant : FlexSim envoie des requêtes à Arduino via le serveur Java qui est toujours à l'écoute en attendant qu'un client se connecte, ce dernier assure la transmission des messages par Socket entre les deux clients. La carte Arduino traite les messages reçus puis les envoie à l'API grâce à la carte adaptative à base d'optocoupleur qui va permettre le passage du 5V au 24V et vice-versa. L'automate exécute les ordres envoyés par la carte Arduino en passant par le chemin inverse, c'est-à-dire que, l'API envoie un signal de 24V qui va être transmit à Arduino en un signal de 5V en passant par le circuit imprimé. Finalement, la carte Arduino va convertir le signal en action et l'envoyer, sous forme de message, à FlexSim via le réseau, à ce moment-là FlexSim exécute une tâche spécifique selon le message qu'il a reçu.

Voici une figure qui illustre le principe de fonctionnement de l'application :

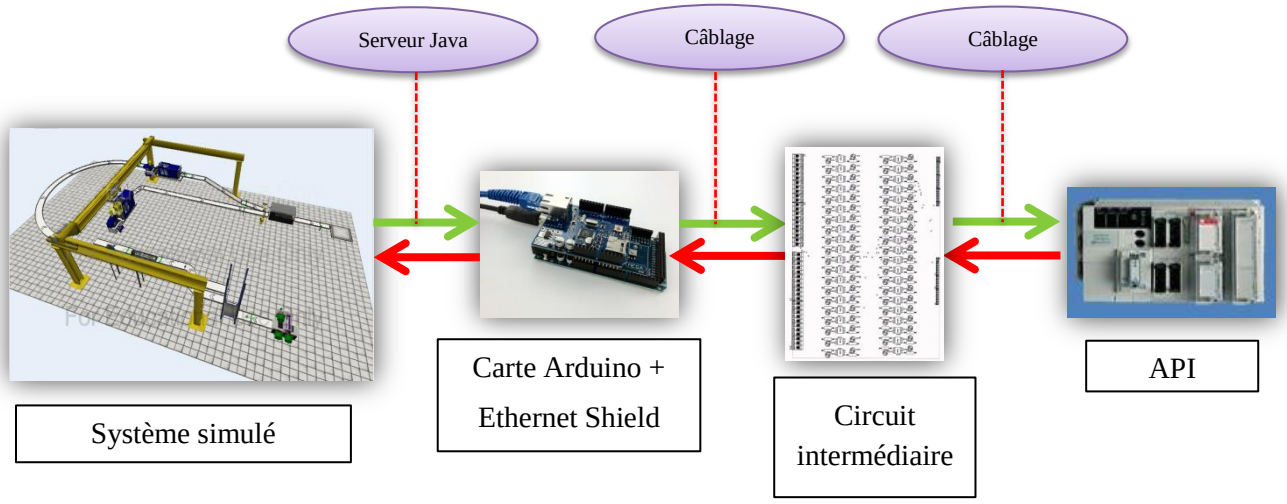


Figure IV-12. Figure qui illustre le principe de fonctionnement de l'application

IV.4 Résultat

La simulation des deux systèmes proposés dans ce chapitre, a abouti à de bons résultats. En effet, on a pu établir une simulation Hardware-in-The-Loop en utilisant Java-Arduino comme interface de communication entre l'automate programmable et le système simulé.

Conclusion

Ce chapitre comporte deux systèmes de production simulés, et commandés par un automate programmable à l'aide d'une carte programmable Arduino. Un serveur programmé avec Java est nécessaire, afin d'effectuer la communication entre le logiciel de simulation et la carte Arduino.

On a exposé l'essentiel de notre travail dans ce chapitre, c'est-à-dire, la réalisation de maquettes virtuelles pédagogiques.

Conclusion générale

Tout d'abord, nous tenons à souligner l'expérience très enrichissante acquise au cours du stage au sein du Centre de Développement des Technologies Avancées, pour l'élaboration de ce projet de fin d'étude.

Par ailleurs, des généralités sur la simulation ont été présentées ainsi que le choix du logiciel FlexSim pour la simulation des systèmes industriels, que ce soit pour des fins pédagogiques ou pour un projet industriel.

Par la suite, on a introduit l'architecture de l'application qui consiste à commander des systèmes simulés, par un automate programmable industriel. Le duo composé des logiciels NetBeans sous le langage Java et du serveur OPC a été utilisé pour la mise en œuvre de cette simulation. Après simulation, on a constaté qu'il y avait un temps de réponse assez large entre les systèmes simulés et l'API.

Afin d'améliorer la simulation et permettre aux étudiants de bénéficier de la totalité des fonctionnalités d'un automate, une couche logicielle, qui faisait le lien entre l'API et le reste de l'application, a été éliminée, en l'occurrence le serveur OPC, pour la remplacer avec une carte Arduino. Celle-ci, présente un avantage considérable : la possibilité de câbler les entrées/sorties de l'API avec cette même carte. Ainsi, l'API se comporte de la même manière qu'avec un système réel.

Néanmoins, durant la réalisation de ce projet plusieurs difficultés ont été rencontrées. Sans être exhaustif, nous citons :

- Pour la réalisation de ce projet nous avons dû apprendre à manipuler quatre nouveaux logiciels : FlexSim, NetBeans (Java), KEPServerEX (OPC) et le logiciel de programmation d'Arduino.
- Quand il a été question d'enlever les couches logicielles, initialement on devait aussi enlever Java, afin d'établir une communication directe entre le PC et la carte Arduino. Ça a été une réussite en premier lieu, puis on a perdu la connexion.
- Trouver un moyen de communication entre l'API, qui fonctionne avec une tension de 24V, et la carte Arduino qui délivre et accepte des tensions qui ne dépassent pas les 5V.

La manipulation de la carte Arduino représente un véritable plus dans notre apprentissage et nous ouvre d'autres perspectives d'avenir en raison des nombreuses applications et tâches qu'elle peut accomplir notamment dans le domaine de l'Automatique. En effet, en remplaçant le serveur OPC, elle a rendu la maquette virtuelle plus pratique d'un point de vu pédagogique, contrairement à la première configuration, qui nécessitait la programmation d'un serveur. Ceci, nous a permis de concevoir des systèmes plus complexes, afin de considérer la simulation de systèmes industriels à grande échelle.

Comme perspectives, il est préférable d'établir la communication directement entre le logiciel de simulation (PC) et la carte Arduino sans passer par le serveur Java. Cela évitera de perde du temps en essayant de programmer avec Java, surtout pour ceux qui ne le maîtrise pas.

Il serait également intéressant de simuler les défaillances qui peuvent se produire au cours du fonctionnement des systèmes réalisés. Par exemple le dysfonctionnement d'une machine, le blocage d'un vérin ou bien la défaillance d'un capteur. Cela aura pour but de préparer l'étudiant à faire face à des situations similaires dans l'industrie.

Annexe A • FlexSim

Définition

Flexsim est un logiciel de simulation à évènements discrets des systèmes manufacturiers qui a été développé par FlexSim Software Products, Inc. Nous allons montrer, ci-dessous, les étapes à suivre pour créer un système et démarrer sa simulation :

Etape1 : Démarrage de FlexSim

Pour pouvoir démarrer FlexSim il suffit de cliquer sur son icône qui est sur le bureau de votre ordinateur. Une fois la fenêtre de démarrage apparue, vous devez sélectionner «new model »

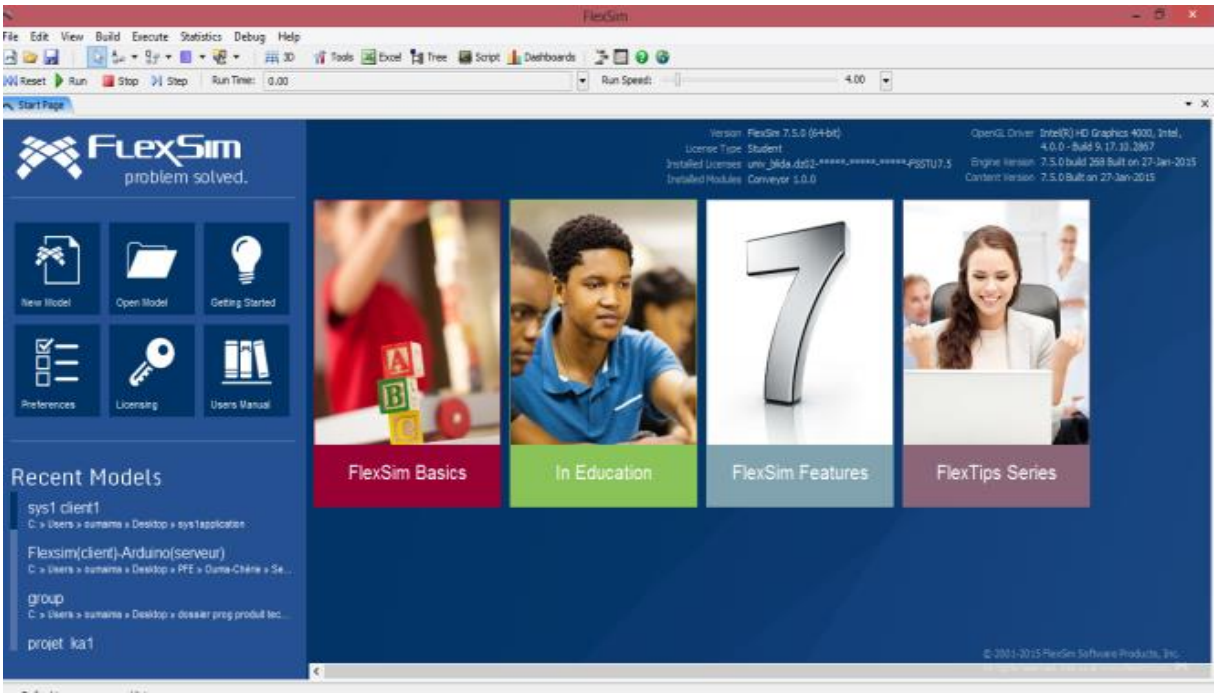


Figure A.1. Fenêtre qui apparait à l'ouverture de FlexSim

Etape2 : La création d'objet

Vous pouvez créer les objets de deux façons différentes :

1. Entrez dans le mode de création d'objets en cliquant sur le bouton  sur la barre d'outils dans le haut de l'écran, puis cliquez, par la suite, sur un objet dans la librairie d'objets située à gauche de l'écran. Cliquez sur la zone où mettre l'objet dans l'espace quadrillée de construction du modèle.

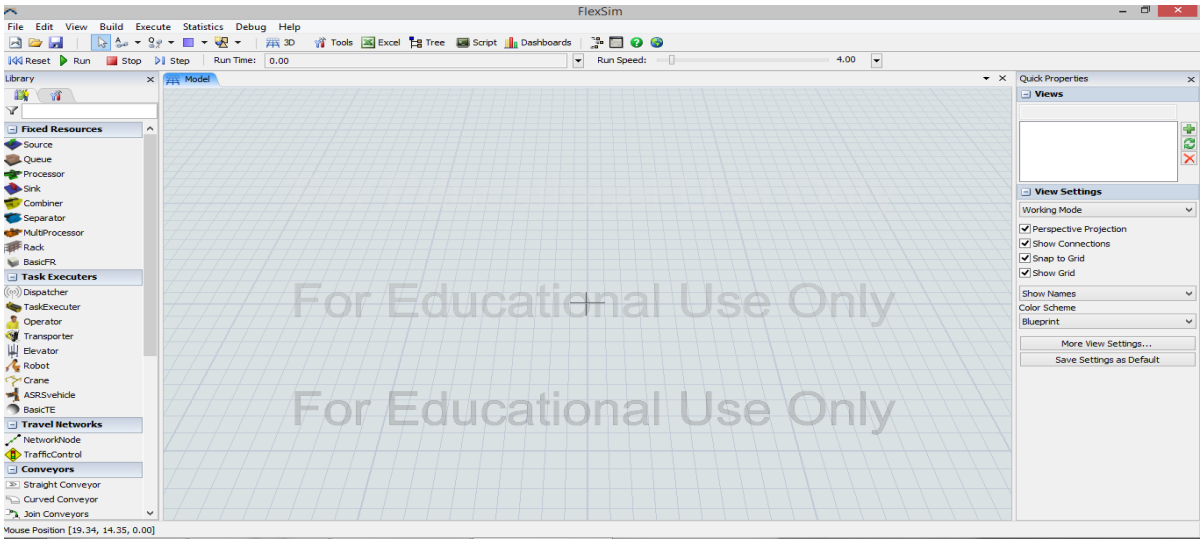


Figure A.2. Fenêtre de placement des objets FlexSim

2. Cliquez, ensuite, sur un objet de la librairie et maintenez le bouton de gauche de la sourie enfoncée et allez positionner l'objet dans le plan du modèle.

La librairie

La librairie de Flexsim comporte deux types d'objet :

1. les objets discrets pour la simulation à évènement discret dédié à la simulation des systèmes opérationnel à titre d'exemple : les systèmes de production.

2. les « fluides » pour la simulation à événement continue, servant à faire des simulations précises ou les résultats varient en fonction du temps de simulations citant à titre d'exemple : le niveau de remplissage d'un réservoir.

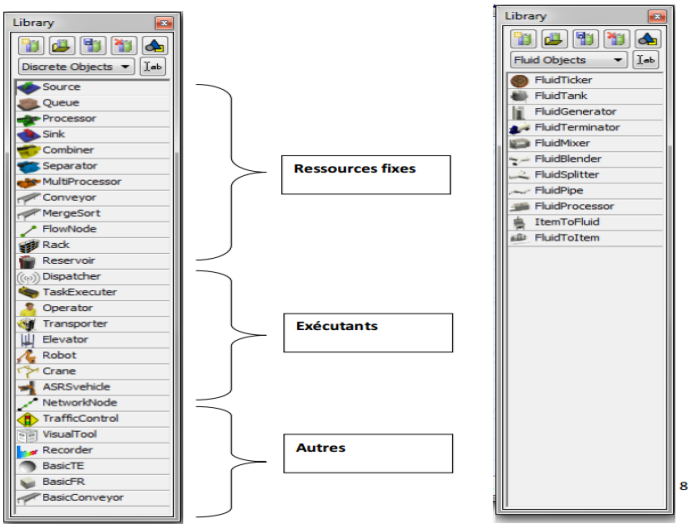


Figure A.3. Librairie FlexSim

Vous pouvez aussi créer votre propre librairie, avec des objets externes dont vous aurez défini les paramètres. Pour les ajouter il suffit d'ouvrir la fenêtre « file» sélectionner « User New Library». Une fois la Library est ajoutée il faut définir ces paramètres et la sauvegarder en cliquant sur « Save Library»

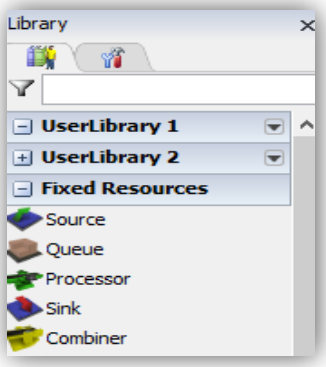


Figure A.4. Exemple de librairies personnalisables

Exemple de création d'objet

Voici les étapes à suivre pour la création d'un système comprenant une source, un convoyeur, un procédé (*Processor*) et un *Sink* pour le stockage.

Vous n'avez qu'à sélectionner le type d'objet dans la librairie et glisser sur le plan de simulation.

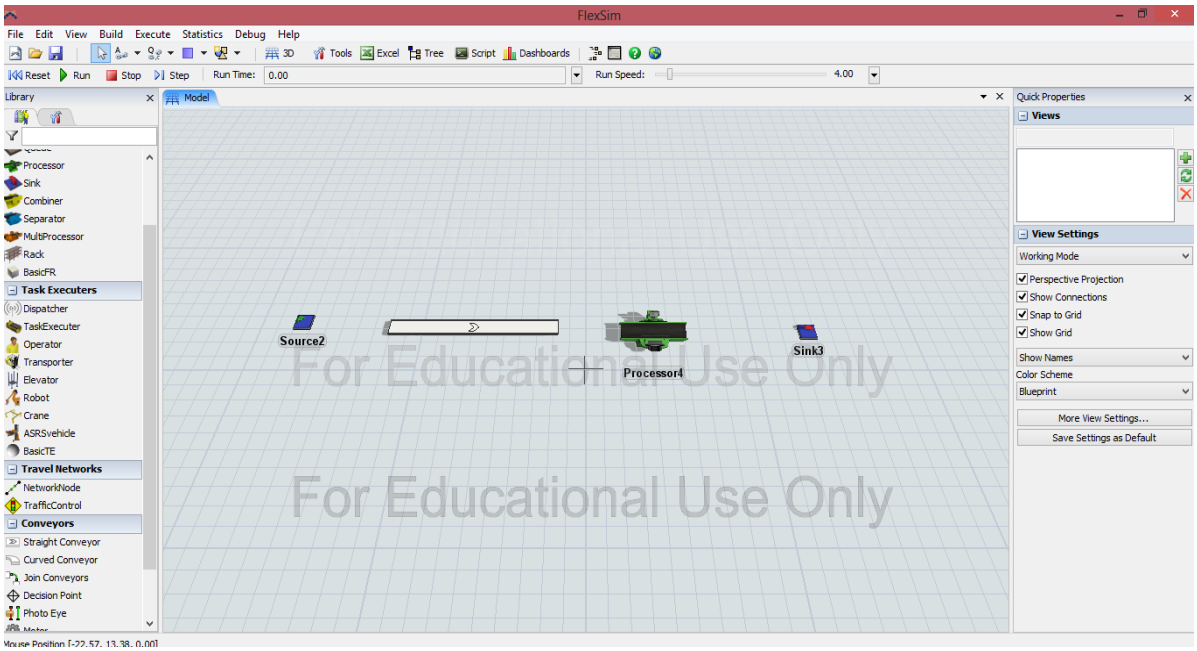


Figure A.5. Exemple de création d'objet

Il est important de se rappeler que chaque système doit comporter au moins une seul *Source* et une terminaison *Sink* ou *Queue.*

Vous devez par la suite, établir les relations entre les objets afin de déterminer la séquence et les paramètres qui permettront la simulation du modèle.

Modification des paramètres

Tous les objets de la simulation possèdent des variables opérationnelles pouvant être modifiées. Vous pouvez accéder au menu d'édition en cliquant deux fois sur l'objet ou bien en cliquant sur le bouton droit sur l'objet et en sélectionnant « Properties ». Chaque classe d'objet possède des caractéristiques qui lui sont propres. La fenêtre de propriétés « Properties » possède plusieurs onglets reliés à différentes options de configuration.

Citons à titre d'exemples le *Processor* :

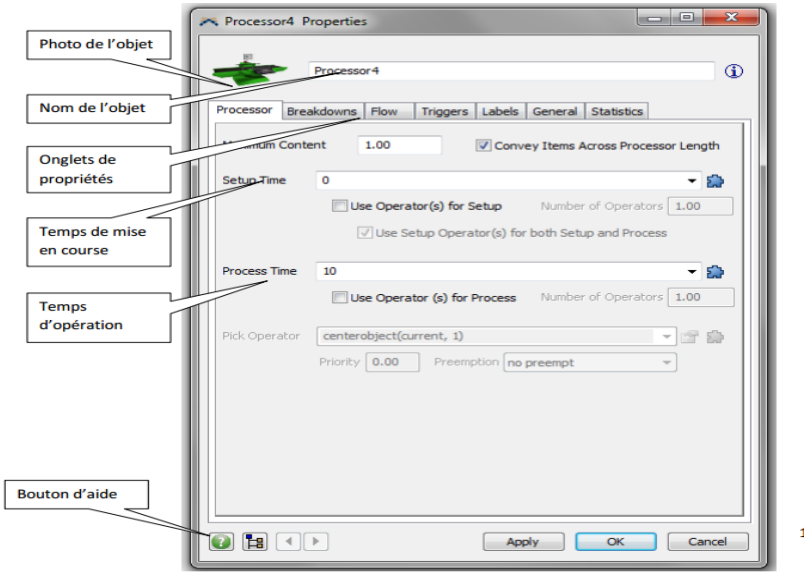


Figure A.6. Fenetre de configuration du Processor

Chaque onglet à ces propres propriétés.

- Onglet *General* : dédié à la position et la rotation des objets ainsi que le dimensionnement.
- Onglet *Labels* : définit des Attributs associés à l'objet par l'utilisateur.
- Onglet *Statistics* : donne des informations sur les statistiques de l'objet, incluant des chartes et des graphiques.
- Onglet *Flow* : détermine la logique du flux relié à l'objet. Ce qui arrive à l'objet et ce qui en sort.
- Onglet *Triggers* : fonctions optionnelles de l'objet pour améliorer le comportement son comportement face à différentes situations lors de la simulation.
- Onglet *Breakdowns* : détermine des informations et paramètres de fiabilité de l'objet en question.

Annexe B • NetBeans

Définition

NetBeans IDE est un environnement de développement intégré (EDI) modulaire basé sur des normes, écrit dans le langage de programmation Java. Le projet de NetBeans IDE consiste en un EDI Open Source complet écrit dans le langage de programmation Java qu'est en lui-même un langage extensible, portable, qui a la notion d'héritage, et en une plate-forme d'application cliente riche, qui peut être utilisée comme structure générique pour créer n'importe quel type d'application. Il comporte plusieurs classes dans son projet il est configurable vous pouvez l'utiliser comme client ou comme serveur tout dépond de notre besoin.

Création d'un projet sur Java NetBeans

Pour créer un projet il suffit de cliquer sur l'icône « NetBeans» située sur le bureau de votre PC. Une fois la page apparue vous allez sélectionner la fenêtre «File» puis cliquer sur « New Project » (vous devez lui donner un nom) et le projet est créé.

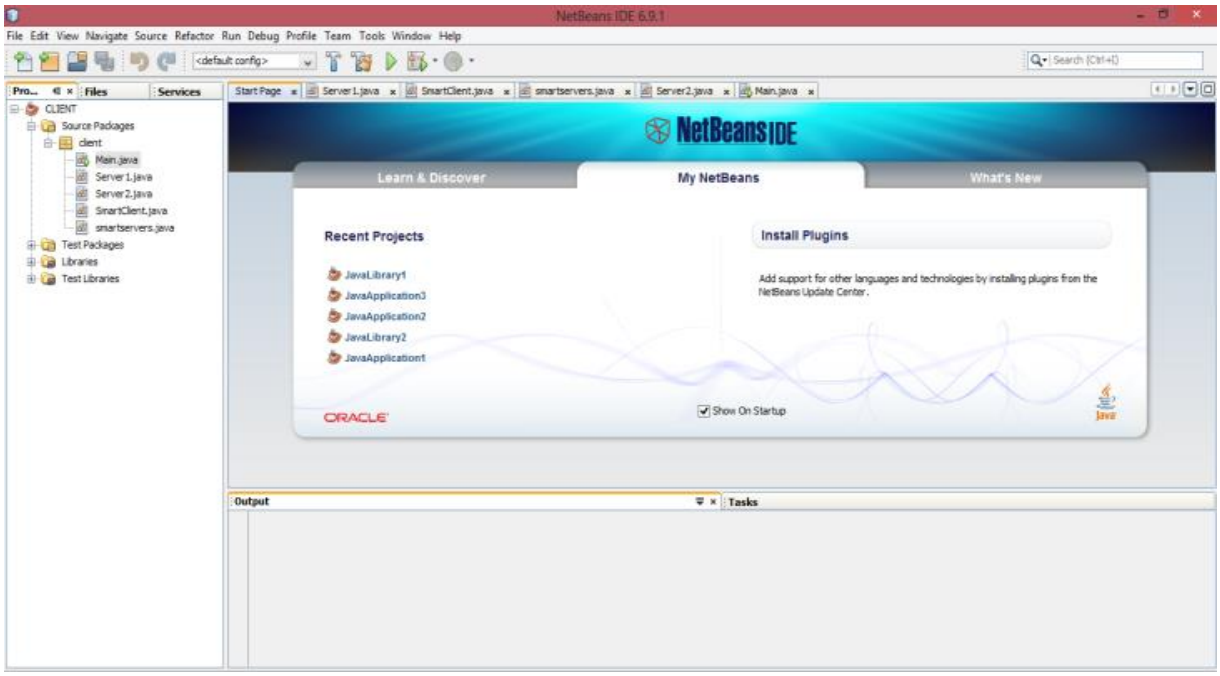


Figure B.1. Fenêtre d'accueil du logiciel NetBeans

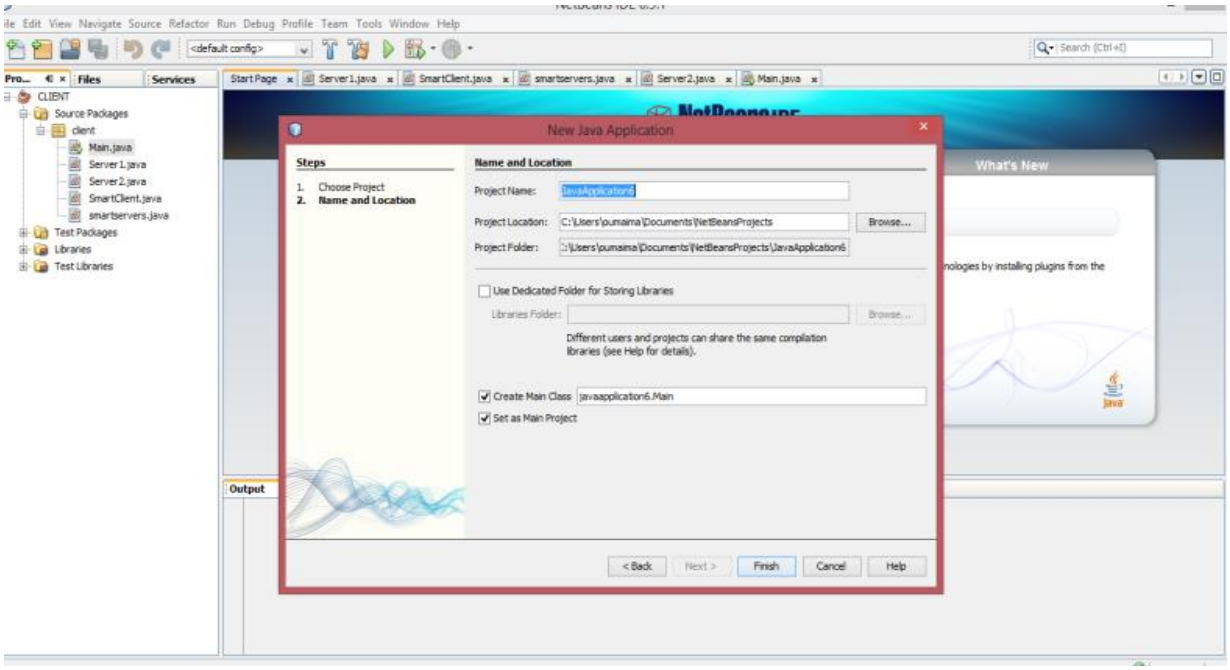


Figure B.2. Exemple de création d'un projet

Vous pouvez aussi ouvrir un projet déjà éxistant sur Le PC en cliquant sur « Open Project» sur la fenêtre « File».

Création de l'entité « Main »

Si vous voulez configurer un serveur qui comporte plusieurs classes. Commençez par la classe «main.java ». Cette classe est le noyau du programme. Vous pouvez la créer en cliquant sur « Le projet» puis sur « New» après sur «Entity Class ». Ainsi la classe est ajoutée.

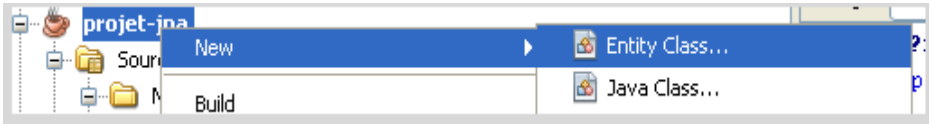


Figure B.3. Création d'une entité

Voici comment créer une entité « Main »

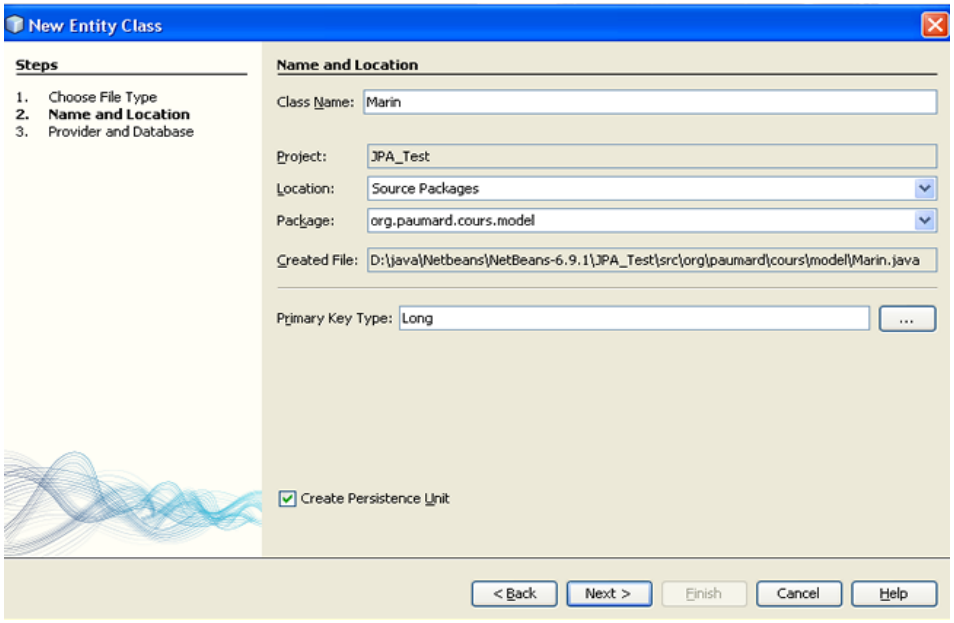


Figure B.4. Exemple de création d'une entité « Main »

La création des « Java Class » contenant le programme serveur

La création des « Java Class » de la même manière que la création de la classe « Main », sauf que vous devez choisir « Java Class » à la place de « Entity Class ».

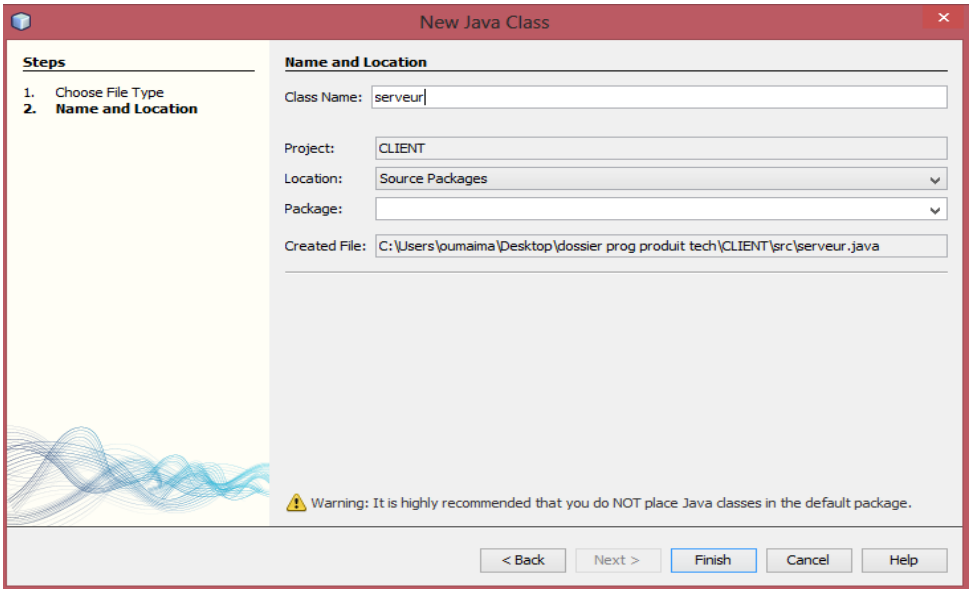


Figure B.5. Exemple de création d'une Java Class

La classe serveur comporte des fonctions et des messages dédiés à l'échange de message entre un serveur et un client.

Pour ajouter une autre classe Java, il suffit de suivre les mêmes étapes que précédemment.

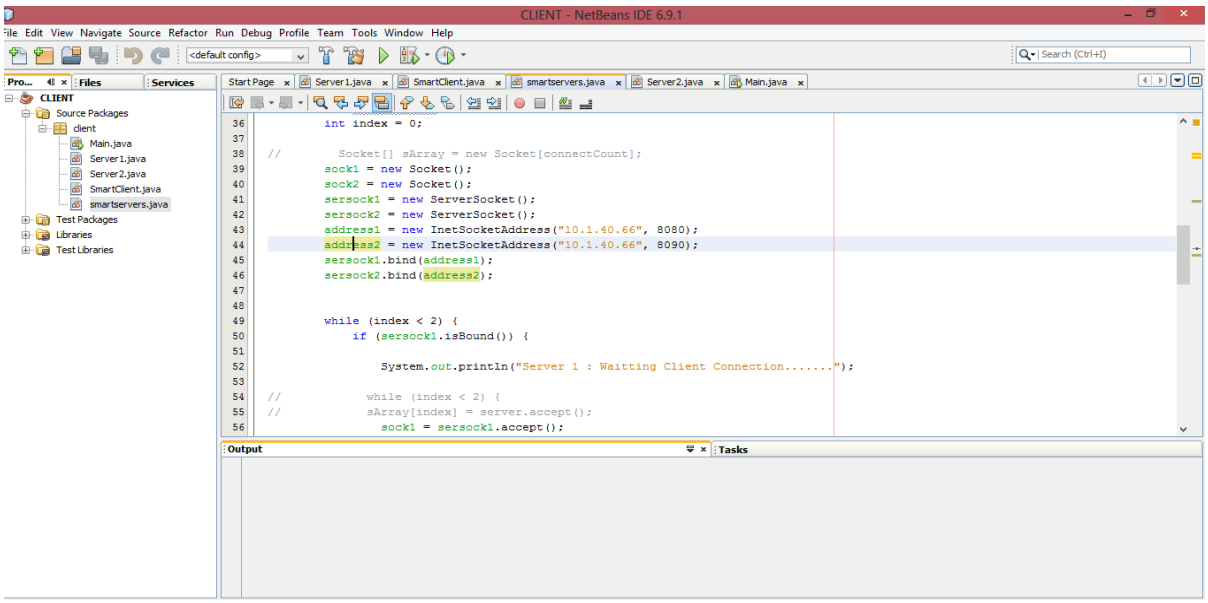


Figure B.6. Fenêtre de programmation d'un programme Java

Annexe C • Rappel sur le Grafcet

Le GRAFCET (**GRA**phe **F**onctionnel de **C**ommande **É**tape **T**ransition) est un langage graphique de modélisation destiné à représenter le fonctionnement d'automatismes séquentiels dont les informations sont de type logique (vrai ou faux). Il a été proposé par ADEPA (Agence pour le Développement de la Productique Appliquée à l'industrie) en 1977, il peut se câbler par séquenceur, être programmé sur automate ou sur ordinateur. Le Grafcet représente l'évolution d'un cycle comprenant des étapes, des transitions et de liaison. Les étapes peuvent être actives ou inactives [21, 11].

Eléments de base d'un GRAFCET

Etape initiale

L'étape initiale caractérise l'état du système au début du fonctionnement. Elle se différencie en doublant les côtés du carré.

Etape

Une étape représente un état particulier du système à un moment donné de son cycle de fonctionnement. Les étapes sont numérotées dans l'ordre croissant. A chaque étape on peut associer une ou plusieurs actions.

Transition

Une transition indique la possibilité d'évolution qui existe entre deux étapes et donc la succession de deux activités dans la partie opérative. Lors de son franchissement, elle va permettre l'évolution du système. A chaque transition est associée une réceptivité qui exprime la condition nécessaire pour passer d'une étape à une autre. On représente une transition par un petit trait horizontal sur une liaison verticale.

Liaison

Une liaison est un arc orienté, qui doit toujours aller d'une étape à une transition ou d'une transition à une étape.

Réceptivité

La réceptivité est une fonction combinatoire d'information booléenne telle que :

- Etat des capteurs
- Impulsion sur le bouton poussoir
- Action d'un temporisateur, d'un compteur.
- Etat actif ou inactif d'autres étapes

L'exemple suivant illustre l'emplacement des éléments cités ci-dessus dans un GRAFCET :

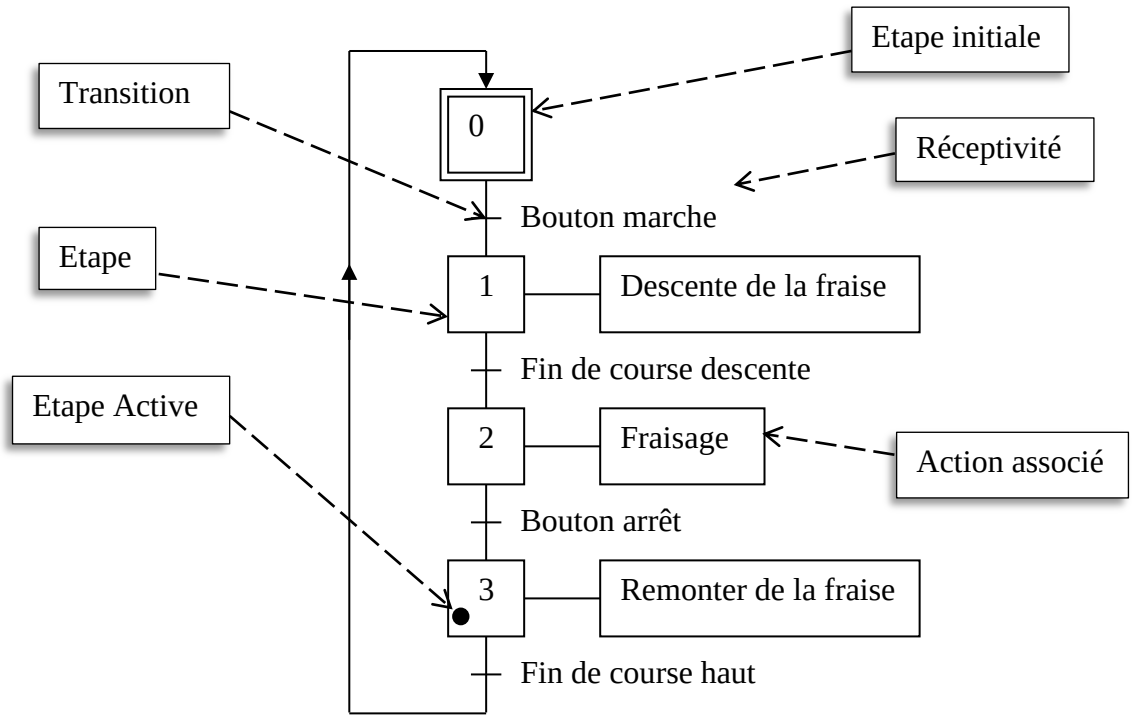


Figure C.1. Exemple récapitulatif montrant les éléments de base d'un Grafcet

Actions

À une étape d'un grafcet on peut associer une ou plusieurs actions, ces derniers caractérisent ce que la partie opérative doit faire à chaque fois que l'étape associée est active. Ces actions peuvent être destinées à l'extérieur de PC (sortie) ou à l'intérieur de PC (lancement de temporisation, comptage...).

Une action est décrite selon le niveau du grafcet d'une manière littérale ou symbolique à l'intérieur d'un ou plusieurs rectangles reliés à l'étape à laquelle elles sont associées. On trouve :

Action continue : l'action associée à l'étape se continue tant que l'étape à laquelle elle est associée est vraie.

Action conditionnelle : c'est une action continue dont l'exécution est soumise à la réalisation d'une condition logique

Action temporisée : c'est une action conditionnelle dans laquelle le temps intervient comme condition logique.

Action mémorisée : l'action associée à cette étape sera maintenue même si l'étape de mise à 1 est désactivée, jusqu'à l'exécution de l'étape de mise à zéro.

Règles d'évolution d'un GRAFCET

Ces règles définissent les conditions dans lesquelles les étapes peuvent être actives ou inactives.

Règle 1 : étape initiale

Les étapes initiales sont celles qui sont activées au début du fonctionnement. Il doit toujours y avoir au moins une.

Règle 2 : franchissement d'une transition

Une transition est soit validée soit non validée. Elle est validée lorsque toutes les étapes qui la précèdent immédiatement sont actives. Elle ne peut être franchie que lorsqu'elle est validée, **et** que la réceptivité associée à la transition est vraie. Elle est alors obligatoirement franchie.

Règle 3 : évolution des étapes active

Le franchissement d'une transition entraîne l'activation de toutes les étapes immédiatement suivantes et la désactivation des étapes immédiatement précédentes.

Règle 4 : évolution simultanées

Plusieurs transitions simultanément franchissables sont simultanément franchies.

Règle 5 : activation et désactivation simultanées d'une étape

Si, au cours du fonctionnement, une étape activée et désactivée au même temps elle reste activée.

Règles de construction d'un GRAFCET

Arc orienté

On relie étapes et transition, qui doivent strictement alterner, grâce à des arcs orientés (Figure C.2). Par convention, étapes et transitions sont placées suivant un axe vertical. Les arcs orientés sont de simples traits verticaux, lorsque la liaison est orientée de haut en bas, et sont munis d'une flèche vers le haut lorsque la liaison est orientée vers le haut.

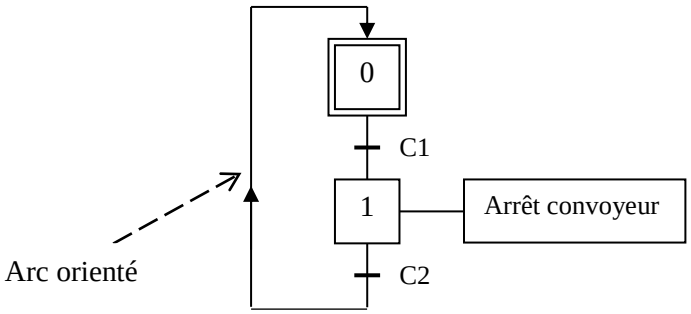


Figure C.2. Figure mettant en valeur l'Arc orienté

Convergence et divergence en « ET »

Si plusieurs étapes doivent être reliées vers une même transition, alors on regroupe les arcs issus de ces étapes à l'aide d'une double barre horizontale appelée convergence « en ET ».

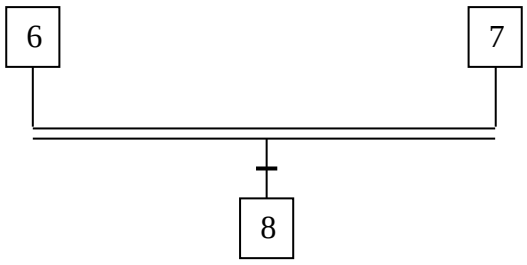


Figure C.3. Convergence en « ET »

Si plusieurs étapes doivent être issues d'une même transition, alors on regroupe les arcs allant vers ces étapes à l'aide d'une double barre horizontale appelée divergence en « ET ».

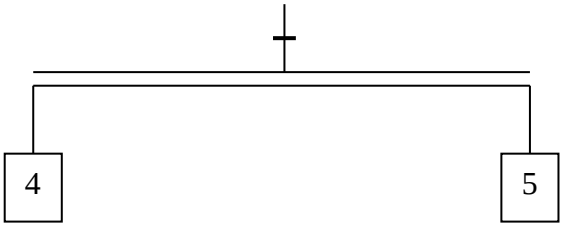


Figure C.4. Divergence en « ET »

Convergence et divergence en « OU »

Lorsque plusieurs transitions sont reliées à une même étape, on regroupe les arcs par un simple trait horizontal et l'on parle de convergence « en OU ».

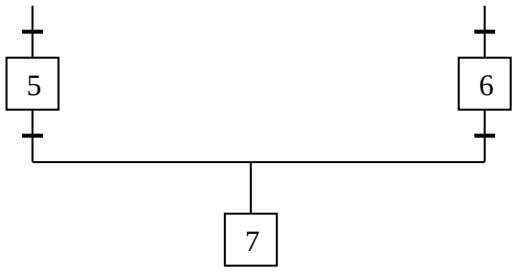


Figure C.5. Convergence en « OU »

Lorsqu'une étape est reliée à plusieurs transitions, on regroupe les arcs par un simple trait horizontal et on parle de divergence « en OU ».

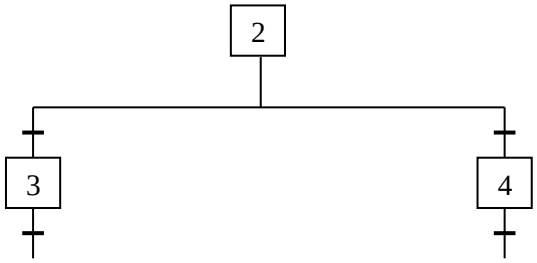


Figure C.6. Divergence en « OU »

Saut d'étape

Les sauts d'étapes permettent de sauter plusieurs étapes en fonction des conditions d'évolution. Le saut d'étape comprend au minimum le saut d'une étape.

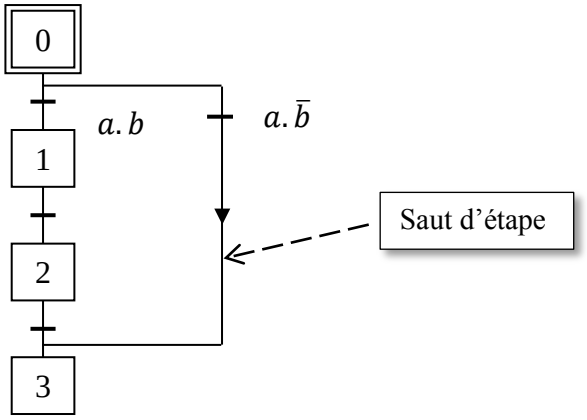


Figure C.7. Figure mettant en évidence un Saut d'étape

Dans le GRAFCET (Figure C.7), si la réceptivité « $a.\bar{b}$ » est vraie alors les étapes 1 à 2 sont sautées.

Reprise d'étape

La reprise d'étapes au contraire, permet de recommencer plusieurs fois si nécessaire une **même** séquence (Figure C.8). La reprise d'une séquence doit comporter au moins trois étapes **puisque** l'activation d'une étape comporte la désactivation de l'étape précédente et la validation de l'étape suivante.

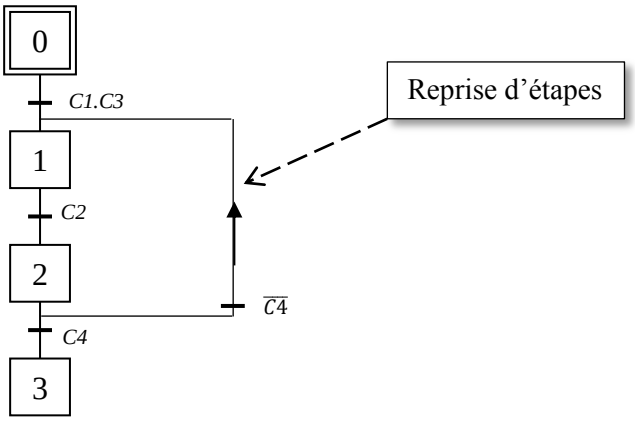


Figure C.8. Figure mettant en évidence une reprise d'étapes

Niveau d'un GRAFCET

GRAFCET niveau 1

Appelé aussi niveau de la partie commande, il décrit l'aspect fonctionnel du système et les actions à faire par la partie commande en réaction aux informations provenant de la partie opérative indépendamment de la technologie utilisée. Les réceptivités sont décrites en mots et non en abréviations, on associe le verbe à l'infinitif pour les actions.

GRAFCET niveau 2

Appelé aussi niveau de la partie opérative, il tient compte de plus de détails des actionneurs, des pré-actionneurs et des capteurs, la représentation des actions et réceptivités est écrite en abréviations et non en mots, on a associé une lettre majuscule à l'action et une lettre minuscule à la réceptivité.

Mise en équation d'un GRAFCET

Soit le grafcet simple suivant (Figure C.9) :

A chaque étape i est associée une variable **Xi** :

- **Xi=1** si l'étape i est **active**
- **Xi=0** si l'étape i est **inactive**

La réceptivité Ri a pour valeur :

- **Ri=0** si la réceptivité est **fausse**
- **Ri=1** si la réceptivité est **vraie**

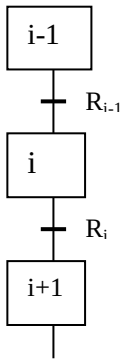


Figure C.9. Exemple d'un Grafcet

Le **but** est de déterminer les variables qui interviennent dans l'activité de l'étape i :

- La **C**ondition d'**A**ctivation de l'étape i donne :

$$\mathbf{CAX_i = X_{i-1}\ R_{i-1}}$$

- La **C**ondition de **D**ésactivation de l'étape i donne :

$$\mathbf{CDXi = X_i\ R_i = X_{i+1}}$$

- Si la **CA** et la **CD** de l'étape i sont fausses, l'étape i reste dans son état (effet mémoire).

Bibliographie

[1] B. Riera, F. Gellot, J.-P. Chemla et S. Triki, «L'utilisation pédagogique et l'ensegenement des TIC dans les automatismes,» *J3eA - Journal sur l'enseignement des sciences et technologies de l'information et des systèmes*, p. 5, 2001.

[2] B. Riera, G.Martel, M.Lambert, E.CHerifi (1998). ''Experimental Platform for Supervision of Complex Automated Systems '', Proceedings of the Seventh IFAC/IFIP/IFORS/IEA on Analysis Design and Evaluation of MMS, pp. 305-310. Kyoto, Japan

[3] A. Law et W. Kelton, Simulation Modeling and Analysis, Editions McGraw-Hill, 2nd edition, 1991.

[4] P.-J. Erard et P. Déguenon, Simulation par évenements discrets, Lausanne: Presses polytechniques et universitaires romandes, 1996.

[5] D. Abdou, Modélisation et Simulation d'un Problème d'Ordonnancement, Angers, Septembre 2005.

[6] G. Eric, «Ressources pour l'enseignement de la simulation avec WITNESS,» Lanner Group, Paris, 2014.

[7] D. Désilets, «Guide utlisateur Flexsim,» 2009.

[8] Flexsim company, «FlexSim Software Products, Inc.,» Flexsim company, [En ligne]. Available: https://www.flexsim.com/company/. [Accès le Mars 2015].

[9] D. Weber, «La simulation Hardware-in-the-loop (HIL),» AB Mécatronique, [En ligne]. Available: http://www.abmecatronique.com/la-simulation-hardware-in-the-loop-hil_540611/. [Accès le Avril 2015].

[10] C. Jossin, «Buts de l'automatisation,» Encyclopédia Universalis.

[11] L. Bergougnoux, Automates Programmables Industriels, Marseille, 2004-2005.

[12] W. Bolton, Les Automates Programmables Industriels, Paris : Dunod, 2010.

[13] «Les Automates Progammables Industriels,» GEEA.

[14] M. Bertrand, «Automates programmables industriels,» *Techniques de l'ingenieur* , n° %1S 8015, p. 19, 2001.

[15] P. Jargot, «Langages de programmation pour API. Norme IEC 1131-3,» *Techniques de l'ingenieur*, n° %1S 8030, p. 23, 1999.

[16] F. High-Tech, «Java,» [En ligne]. Available: http://www.futura-sciences.com/magazines/high-tech/infos/dico/d/internet-java-485/. [Accès le Mars 2015].

[17] «Définition de Langage JAVA & JAVA SCRIPT,» [En ligne]. Available: http://ipeti.forumpro.fr/t21-definition-de-langage-java-java-script. [Accès le Mars 2015].

[18] P. Bonnet, «Supervision serveur OPC,» Université Lille 1-sciences et technologies, Lille, 2008.

[19] R. Martin, «Connexion entre un automate et une base de données,» 2008.

[20] M. Condemine, «Introduction à OPC».

[21] Y. Granjon, Automatique : systèmes linéaires, non linéaires, à temps continu, à temps discret, représentation d'état, Paris: Dunod, 2010.

[22] Schneider Electric, *Fiche produit TWDLCAA24DRF*.

[23] M. Margolis, La boîte à outils Arduino 105 techniques pour réussir vos projets, Paris: Dunod, 2012.

[24] Référence Arduino Français, «La carte Arduino Mega 2560,» [En ligne]. Available: http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.MaterielMega2560. [Accès le Avril 2015].

[25] Référence Arduino Français, «Le module ethernet,» [En ligne]. Available: http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.MaterielEthernetShield. [Accès le 2015 Avril].