Introduction générale

La conception, l'apprentissage ou le développement des systèmes industriels impliquent des investissement humains et matériels souvent très coûteux. L'intégration de la simulation dans le domaine industriel permet d'abaisser considérablement ces coûts.

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'Enseignements 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 présents. Les Maquettes Virtuelles son tune 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 recherché dans le domaine de la conception et l'évaluation d'outil de supervision adaptés à l'homme [2]. L'idée de base et 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 programmable 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 cite est à évènements discrets. Il faut aussi rappeler que la simulation des 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 enquetes aux Etats-Unis qui montrent la place de simulation dans l'industrie: l'une d'elles indique que parmi 14 techniques utilisées, la simulation arrive en 2ème position pour 84% des entreprises [3]. Pour les chercheurs, le principal internet est de pouvoir travailler sur un système de

production virtuel, dont le comportement peut êtres très proche du système réel, à moindre cout 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 trois chapitres. Les deux 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 le second chapitre il est question de l'application à réaliser. Le troisième chapitre consiste à présenter l'essentiel des résultats obtenus.

Chapitre1 Généralités sur la simulation et les logiciels de simulation

Introduction

Actuellement, la simulation est utilisée dans de nombreux domaines de recherche et de développement: mécanique, science des matériaux, aéronautique, météorologie…Elle intervient aussi dans les secteurs comme celui de la banque et de 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.

1.1 La simulation

Une fois qu'un modèle a été développé et validé il peut être utilisé pour répondre à toute une série de questions "what if": qu'est-ce que il va se produire si j'applique telle action ou tel changement au système? Comment le système va se comporter d'ici un an? Si je veux améliorer la performance, est-ce que je dois apporter la modification 1 ou plutôt la modification 2?

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 au contraire à l'éthique. Il serait par exemple irrationnel de fermer arbitrairement l'un des guichets d'une banque et 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 les 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.

Exemple Considérons un entrepreneur d'une société industrielle qui se pose la question: doit-on construire ou non une nouvelle usine de production? Sa question est de savoir si le coût de l'élargissement sera amorti par le gain en productivité. Dans ce contexte, il pourrait être intéressant d'avoir un outil qui puisse simuler le changement en productivité au cas de la disponibilité d'une nouvelle usine. Ceci lui permettrait de tester la nouvelle configuration comme si (what if) elle était réalisée.

1.1.1 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.
- Conception et analyse des systèmes informatiques autant que des plateformes hardware.
- Évaluation des stratégies d'ordonnancement.
- Conception des systèmes de communication et des protocoles d'échange des messages.
- Conception et analyse de systèmes de transport.
- Évaluation de la logistique d'organisation de services: hôpitaux, bureau de poste, restaurants. Étude des systèmes complexes et à grand échelle (économie, finance, biologie).
- Applications militaires.

1.1.2 Les raisons du succès de la simulation

La simulation est un des outils le plus communément utilisés dans les sciences et les sciences appliqués.

Les raisons de ce succès sont:

- La disponibilité d'une puissance de calcul toujours en croissance, la disponibilité d'outils logiciels conçus exprès pour la simulation.
- La nécessité et la volonté de manipuler des modèles de plus en plus détaillés et complexes

1.2 La simulation à évènements discrets

La simulation à événements discrets est une technique utilisée dans le cadre de l'étude de la dynamique des systèmes. Une **SED** est une modélisation informatique où le changement de l'état d'un système, au cours du temps, est une suite d'événements discrets. Chaque événement arrive à un instant donné et modifie l'état du système [4].

De nos jours, cette technique est couramment utilisée tant par les industries et les entreprises de services afin de concevoir, optimiser et valider leurs organisations que par les centres de recherche dans l'optique d'étudier les systèmes complexes non-linéaires.

1.3 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 :

1.3.1 Logiciel Simio

Simio est positionné comme un logiciel de simulation dynamique de flux de processus combinant la rapidité dans la modélisation, apportée par l'approche **objet**, avec la souplesse et la puissance propres à la **formulation logique des processus** sous format de workflows graphiques. Ces workflows sont composés à l'aide des blocs fonctionnels (DELAY, DECIDE, ASSIGN, etc) qui avaient été définis dans le langage de simulation SIMAN développé par Dennis Pedgen, auteur également du **Simio** et **Arena.** Il a été lancé en 2009

Avec **Simio** le temps de modélisation est réduit considérablement par rapport aux logiciels plus anciens [5].

1.3.2 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. Unes 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 [6].

1.3.3 Logiciel FlexSim

Le logiciel **FlexSim** est un outil d'analyse et de simulation du 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 nécessaire 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**.

Voilà au dessous des figures présentatives du FlexSim (Figure 1.1 et Figure 1.2) :

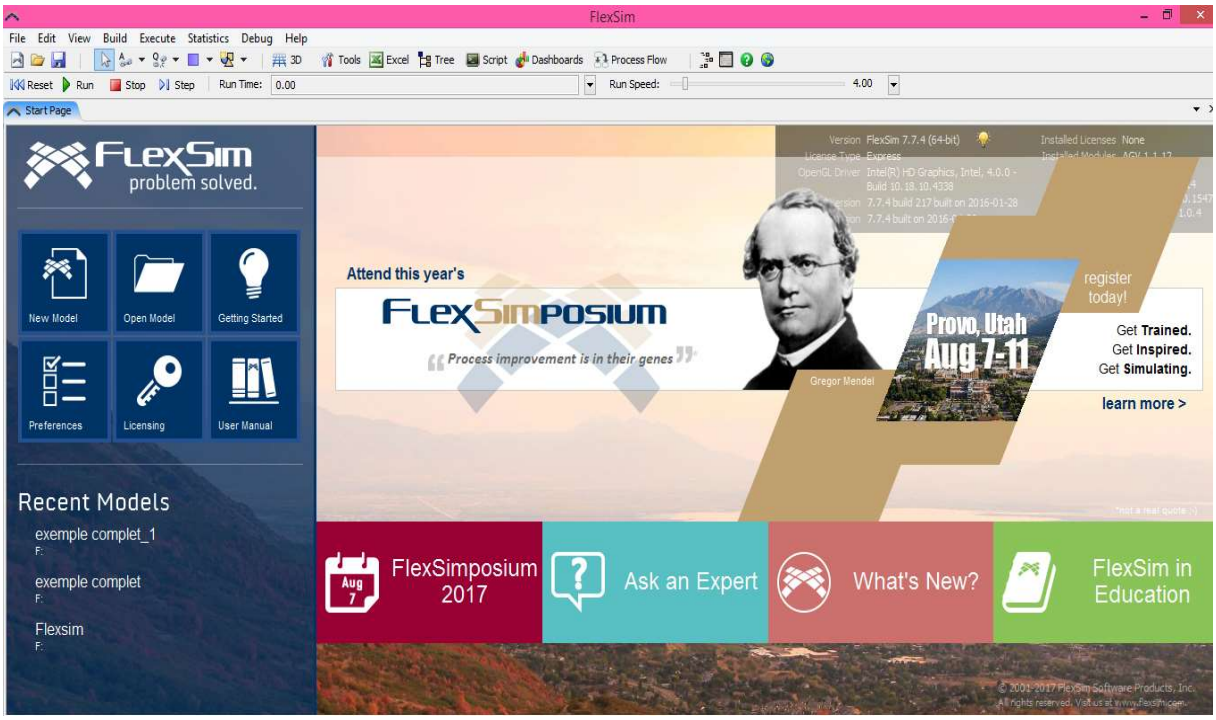

Figure 1.1 Fenêtre du logiciel Flexsim

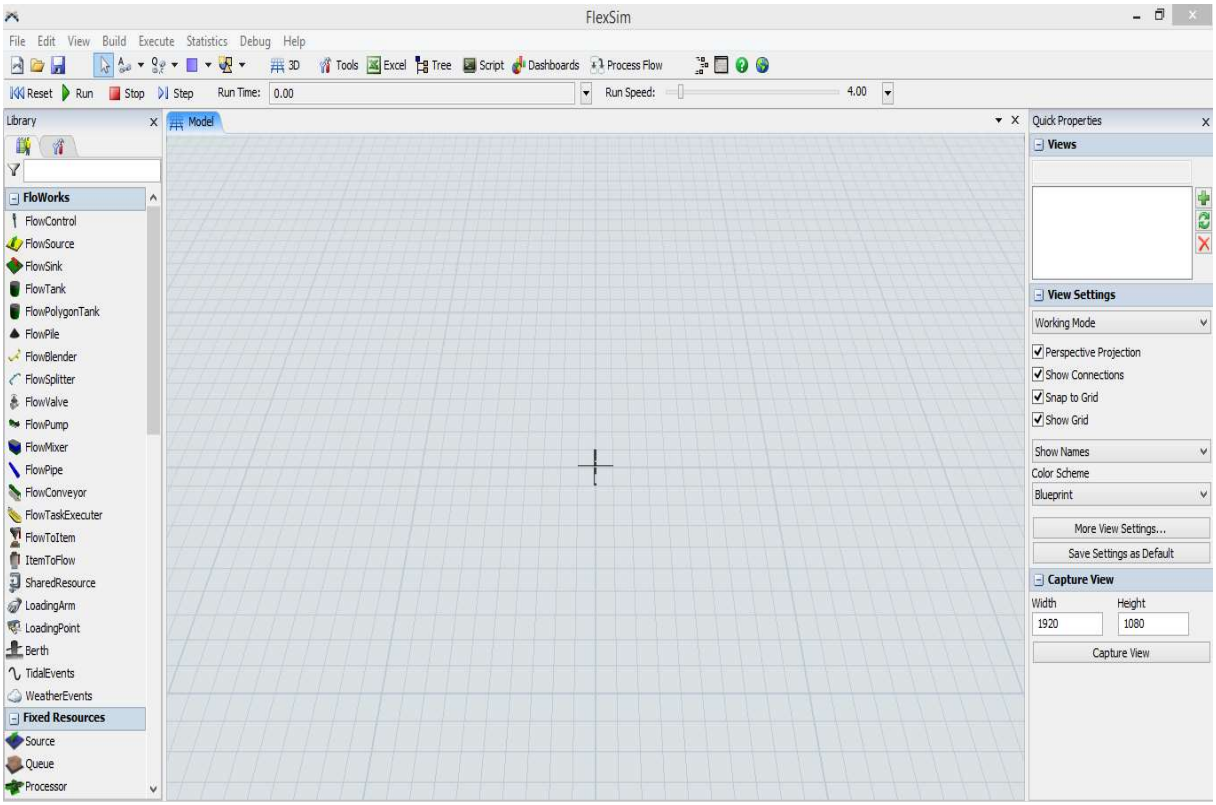

Figure 1.2 Fenêtre du Flexsim lors de création d'un nouveau projet

Le logiciel FlexSim, est le logiciel désiré pour faire la simulation de notre projet.

On va faire une étude abrégée sur FlexSim.

a. 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 Simulation. F&H Simulations fournissait des services de vente, support et formation sur le 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 (Entreprise Dynamics). F&H Simulation a contribué au développement d'objets robustes intégrés dans Taylor ED. En complément, F&H Simulations à 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 Anathony Johnson ont rejoint la sociéte pour superviser le développement d'un logiciel de simulation 3D orienté objet nouvelle génération : Flexsim . F&H Simulations a changé son nom en Flexsim Software Products.

La version 1.0 de Flexsil a été délivrée en février 2003, dotée 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 évènement discrets [8].

b. La librairie FlexSim :

FlexSim possède une bibliothèque très riche en objets qui permet de faire de la simulation à évènement discrets, grâce aux objets discrets qu'elle comporte. Nous, ce qui nous intéresse c'est évidemment les objets de la partie **Floworks** car nous nous faisons de la simulation d'un système hydraulique.

Voici les différentes classes de la bibliothèque FlexSim :

b.1 Floworks :

La classe suivante contient, comme son nom l'indique, des ressources fluides : flowsource, flowcontrol, flowtank….

Le Flow Blender : est utilisé pour extraire des matériaux à partir de plusieurs ports d'entrée en fonction des pourcentages définis par l'utilisateur. Il est le plus couramment utilisé pour le mélange en ligne lorsque le mélange n'est pas effectué par lots.

Figure 1.3 FlowBlender

Flowsource : les flowsources sont l'une des composants les plus importants 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 ».

Figure 1.4 FlowSource

FlowControl : est responsable de l'optimisation du flux via un réseau. Chaque objet connecté au FlowControl fait partie du réseau que FlowControl optimisera. Chaque objet de flux doit faire partie d'un contrôleur FlowControl. Le calculateur calcule tous les taux dans tout le réseau. Une fois que les flux ont été établis, un nouvel événement sera créé pour recalculer le flux. Cela arrive par exemple lorsqu'un réservoir devient plein ou vide.

Figure 1.5 FlowControl

FlowSink : est l'objet que les modélisateurs utilisent lorsqu'ils veulent éliminer le flux du modèle sans le transformer en flux. Il surveille la quantité de matériel qu'il reçoit.

L'utilisateur dispose d'un contrôle total des débits d'entrée de l'évier. Cela comprend une fonction appelée «SetNewFlowRates». Le modélisateur n'a aucun contrôle sur le débit de sortie car le matériel reçu par FlowSink est détruit et ne peut être envoyé en aval.

Figure 1.6 FlowSink

FlowTank : est un objet Buffer qui peut recevoir et envoyer du matériel en même temps. Le modélisateur décide de la capacité maximale du réservoir. La capacité par défaut est calcule en fonction de la taille. Outre les déclencheurs du fond du réservoir et du réservoir, il existe jusqu'à 10 déclencheurs de niveau définis par l'utilisateur. Ces incendies lorsque le contenu du réservoir atteint ce pourcentage. Les paramètres passés sont le courant et le mode (en hausse ou en baisse).

Figure 1.7 Flowtank

FlowPolygonTank : un FlowPolygonTank fonctionne comme un FlowTank régulier, mais il a une forme polygonale hautement personnalisable.Au lieu de la représentation cylindrique, l'utilisateur peut créer sa propre représentation polygonale du réservoir. Les paramètres comprennent: la création d'un réservoir droit ou courbé avec une base de base polygonale dont le nombre et les coordonnées des sommets peuvent être choisis.

Figure 1.8 FlowPolytank

FlowValve : un FlowValve est utilisé pour simuler un traitement qui reçoit en continu et envoie du matériau fluide (comme une cuisinière continue), mais peut très bien fonctionner comme réduction ou augmentation du débit en ajustant le taux maximal.

Figure 1.9 FlowValve

Pipe : un Pipe est utilisé pour simuler le temps nécessaire pour déplacer le matériau d'un objet à l'autre. Il apparaît comme un tuyau cylindrique.

Figure 1.10 FlowPipe

FlowConveyor : un FlowConveyor est utilisé pour permettre un écoulement continu grâce à l'utilisation de plusieurs ports d'entrée et de sortie. Le convoyeur représente une «courroie» sur laquelle un «volume sec» peut être déchargé et transporté jusqu'à la fin du convoyeur. Les entrées peuvent être placées n'importe où sur toute la longueur du convoyeur et l'emplacement d'entrée est visualisé par des flèches. La direction et la vitesse ainsi que la longueur et la largeur du convoyeur sont tous des facteurs qui affecteront le matériau reposant à l'intérieur du convoyeur et quand et à quel port de sortie le matériel sera envoyé.

Figure 1.11 FlowConvoyor

FlowToItem : un FlowToItem est utilisé pour convertir le flux continu en éléments discrets. Flow entre cet objet et les flux sortent du FlowToItem. La quantité de matériau nécessaire pour un débit ainsi que les détails des flux peuvent être définis par l'utilisateur.

Figure 1.12 FlowToItem

ItemToFlow : un ItemToFlow est utilisé pour écouler des flux. Flowitems entre dans cet objet et le flux sort du ItemToFlow. La quantité de flux générée par un flowitem ainsi que la quantité maximale d'éléments pouvant être stockés dans ItemToFlow (fonctionnalité de mise en mémoire tampon) peut être définie par l'utilisateur.

Figure 1.13 ItemToFlow

FlowMixer : un FlowMixer sert à combiner les produits en un seul, nouveau produit. Les différents matériaux peuvent être tirés séquentiellement ou en parallèle. Le Mixer fonctionne toujours en lots. Il n'envoie aucun matériel tant qu'il n'a pas reçu et traité tous les documents qu'il a été configuré pour recevoir

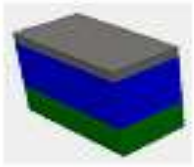

Figure 1.14 FlowMixer

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 Flowsource :

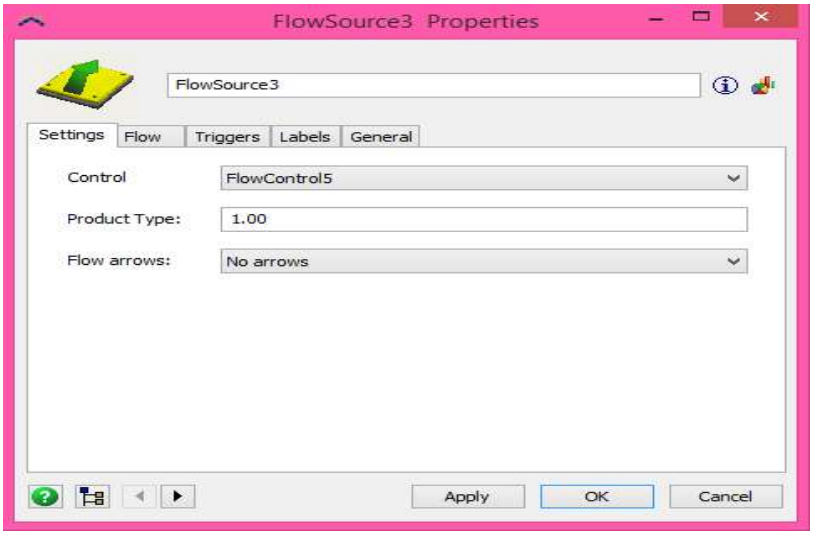

Figure 1.15 Propriétés de Flowsource

Avec :

L'onglet settings contient :

Control - L'objet de contrôle auquel l'objet est connecté.

Product type - Il s'agit de l'ID produit du matériel que cet objet lance.

Flow arrows - Ce paramètre indique si FlexSim doit dessiner des flèches indiquant à partir de quels objets cet objet reçoit un flux et / ou sur quels objets ces objets évoluent. Voir les flèches pour plus d'information.

L'onglet Flow :

Débit - Débit maximal de l'objet.

Règle de débordement - Spécifie la règle d'entrée.

Règle de sortie - Spécifie la règle de sortie.

L'onglet Triggers :

OnReset - Cette fonction est appelée sur la réinitialisation du modèle.

OnMessage - Cette fonction est appelée quand FlowToItem reçoit un message.

OnRateChange - Cette fonction est appelée lorsque le débit d'entrée ou de sortie de cet objet FloWorks change.

OnInput - Cette fonction est appelée lorsque le montant d'entrée réglé par la fonction SetInputTriggerAmount (double montant) est atteint. Une fois que le déclencheur a déclenché, le déclencheur est enclenché à nouveau lorsque, à partir du déclencheur précédent, la quantité d'entrée est atteinte.

OnOutput - Cette fonction est appelée lorsque le montant de sortie réglé par la fonction SetOutputTriggerAmount (double montant) est atteint. Une fois que le déclencheur a déclenché, la gâchette est réglée à nouveau lorsque, à partir du déclencheur précédent, la quantité de sortie est atteinte.

Custom Draw - Cette fonction vous permet de définir votre propre code de tirage.

L'onglet Labels : dans cet 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 et la rotation, de notre objet.

c.Langage de programmation de FlexSim

FlexSim peut être programmé de trois manières différentes, soit avec le langage C++ ou bien **FlexScript** 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.**

Conclusion :

Ce chapitre contient des généralités sur la simulation, le type de simulation utilisé, et enfin, une vue globale sur le logiciel FlexSim ainsi qu'une esquisse sur les différents éléments qui composent sa bibliothèque. Dans le chapitre qui suit, on donnera une présentation sur la base de ce projet, qui se base sur la simulation Hrdware-In-The-Loop, et un aspect général sur les systèmes automatisés.

Chapitre 2 La simulation Hardware-In-The-Loop (HIL)

Introduction

Dans ce 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 des systèmes à simuler en plus d'une introduction sur les systèmes automatisés, les automates programmables et leurs caractéristiques.

2.1 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+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 fait 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 : haut/basse température, fortes accélération 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.

Voici ci-dessous illustration d'une simulation HIL :

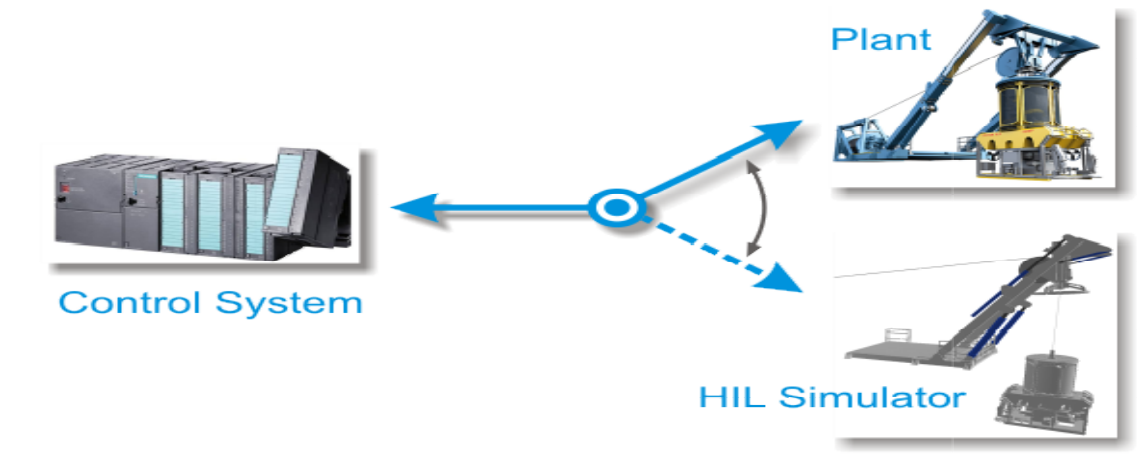

Figure 2.1 Simulation Hardware-In-The-Loop

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 répandue est celui des systèmes de contrôle d'un débit pour un remplissage d'un réservoir, c'est ce qui explique le choix de simulation de systèmes dans ce projet.

2.1.1 Pourquoi le choix d'une partie virtuelle

Dans les systèmes de productions ou les systèmes de contrôles industriels, les spécialistes ou les concepteurs veulent toujours faire des modifications sur ces systèmes sans causer des risques sur eux ou sur le matériel, cette problématique nous a poussés de trouver une solution fiable pour sécuriser le matériel et le personnel.

La simulation HIL est la solution la plus convenable pour ce qu'on a désiré, cette partie virtuelle nous a permet de convertir le système réel en un système virtuel pour pouvoir modifier le système avec des conditions précises.

Les avantages offerts par la simulation Hardware-In-The-Loop sont multiples :

- ✓ La conception et les tests du système de contrôle peuvent être faits sans le système réel (le travail peut s'effectuer en « labo »).
- ✓ Les conditions d'essais sur système de contrôle matériel peuvent être poussés à l'extrême (exemple : haute/basse température, forte débit…).
- ✓ Il est possible d'effectuer des essais sur les effets des défauts et pannes des actionneurs, capteurs et ordinateurs sur l'ensemble du système.

- ✓ Expériences reproductibles et répétables à souhait.
- ✓ Opérations facilitées avec différentes interfaces hommes-machines.
- ✓ Economie de temps et d'argent dans le processus de développement.

2.2 Les systèmes automatisés

« Depuis toujours l'homme est en quête de bien être. Cette réflexion peut paraitre 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 à commencer 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.

Mais au fait qu'est-ce qu'un système ? Bien difficile de répondre à une telle question ! Notre point de vue porte sur les systèmes de production et les systèmes pluri-techniques en général, nous pouvons néanmoins en donner une définition plus large.

Système : toute structure dont la fonction globale est de conférer une valeur ajoutée à un ensemble de matières d'œuvre, dans un contexte donné [10].

Simples ou complexes, les systèmes automatisés sont partout dans notre environnement quotidien.

Ils vont probablement se développer de plus en plus et prendre une place plus importante dans la manière de travailler, tant dans les ateliers de production que dans les divers bureaux des entreprises. Connaître leur fonctionnement permet aussi de mieux comprendre notre environnement.

Un système automatisé ou automatique est un système réalisant des opérations et pour lequel l'homme n'intervient que dans la programmation du système et dans son réglage. Les buts d'un système automatisé sont de réaliser des tâches complexes ou dangereuses pour l'homme, effectuer des tâches pénibles ou répétitives ou encore gagner en efficacité et en précision.

Schéma ci-dessous présente c'est quoi un système automatisé :

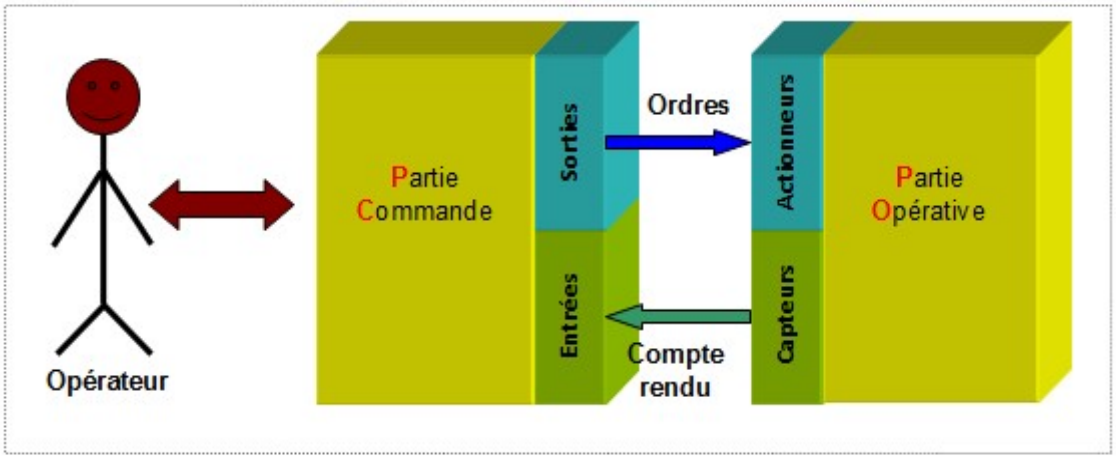

Figure 2.2 Schéma représentatif d'un système automatisé

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 2.3 Barrière d'un parking

Figure 2.4 Distributeur de billets

Figure 2.5 Chaine de production automatisée

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

2.2.1 La partie commande :

Appelée également « partie traitement des informations », elle regroupe tous les composants de traitement des informations nécessaire à la bonne marche de la partie opérative.

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 [11]

Cette partie qui est constituée essentiellement d'un automate programmable industriel qui est le cerveau du pupitre, et une CPU interface qui est connecté à l'automate et qui contient les logiciels de programmations afin d'effectuer la réalisation du programme et le transférer vers l'automate.

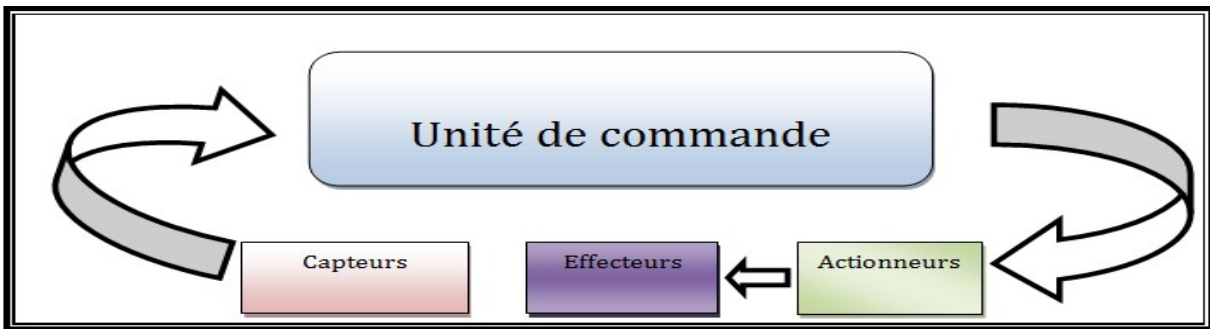

Figure 2.6 Schéma représente la partie commande d'un système automatis

a.Les Automates Programmables Industriels (API)

Quand on dit automatique, c'est évidemment on s'adresse les **API,** ces derniers sont apparus aux Etats-Unis vers 1969 où ils répondaient aux désirs des industries de l'automobile de développer des chaînes de fabrication automatisées qui pourraient suivre l'évolution des techniques et des modèles fabriqués.

Un **API** est une machine électronique programmable par un personnel non informaticien et destiné à piloter en ambiance industrielle et en temps réel des procédés industriels. Un **API** est adaptable à un maximum d'application, d'un point de vue traitement, composants, langage. C'est pour cela qu'il est de construction modulaire.

Il est en général manipulé par un personnel électromécanicien. Le développement de l'industrie à entraîner une augmentation constante des fonctions électroniques présentes dans un automatisme c'est pour ça que l'API s'est substitué aux armoires à relais en raison de sa souplesse dans la mise en œuvre, mais aussi parce que dans les coûts de câblage et de maintenance devenaient trop élevés [12].

L'automatisation permet d'apporter des éléments supplémentaires à la valeur ajoutée par le système. Ces éléments sont exprimables en termes d'objectifs par :

- Accroître la productivité (rentabilité, compétitivité) du système
- Améliorer la flexibilité de production ;
- Améliorer la qualité du produit
- Adaptation à des contextes particuliers tel que les environnements hostiles pour l'homme (milieu toxique, dangereux.. nucléaire…), adaptation à des tâches physiques ou intellectuelles pénibles pour l'homme (manipulation de lourdes charges, tâches répétitives parallélisées…),
- Augmenter la sécurité, etc…

L'automatisation permet à l'entreprise d'améliorer sa compétitivité (coûts des produits, qualité, adaptabilité à la demande, ...). Elle a pour objet d'associer moyens de production et moyens de commande automatique qui permettent d'assurer la reproductibilité du résultat de la manière la plus autonome possible (plus au moins indépendant des interventions humaines).

b. 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 faire des ordres à la partie opérative en vue d'exécuter un travail précis comme par exemple la sortie ou la

rentrée d'une tige du vérin, l'ouverture ou la fermeture d'une vanne. La partie opérative lui donnera en retour des informations relatives à l'exécution du travail.

Voici ci-dessous une figure illustrative d'une structure d'un API (Figure 2.7) :

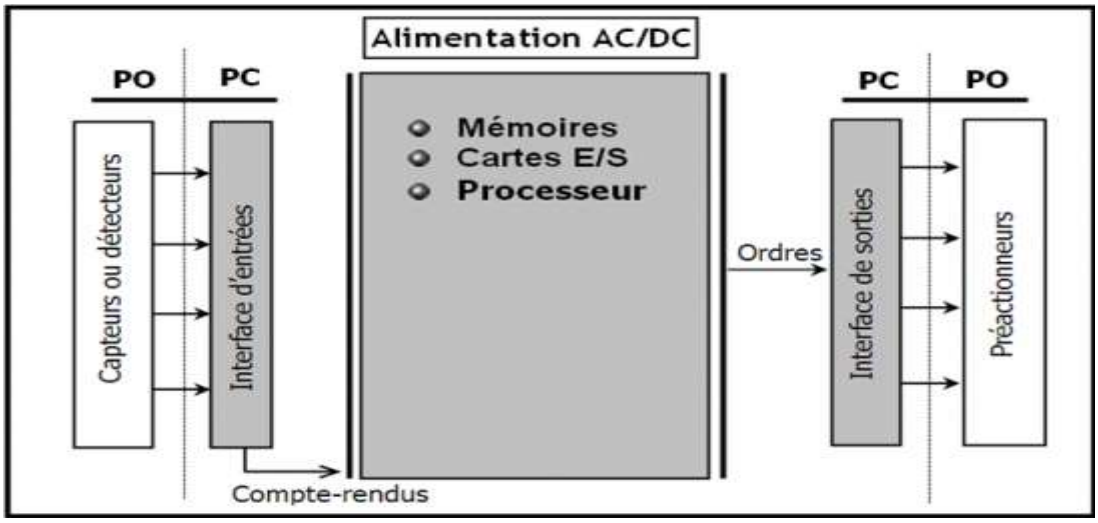

Figure 2.7 Structure interne d'un API

Un API se compose de quatre parties principales : une mémoire, un processeur, des interfaces d'E/S, une alimentation (240 V_{ac} → 24 V_{cc}) [13]

c. Fonctionnement

L'automate programmable **reçoit** les informations relatives à l'état du système et puis **commande** les pré-actionneurs suivant le programme inscrit dans sa mémoire. Généralement les automates programmables industriels ont un fonctionnement cyclique .Le **microprocesseur** réalise toutes les fonctions logiques ET, OU, les fonctions de temporisation, de comptage, de calcul… Il est connecté aux autres éléments (mémoire et interface E/S) par des liaisons **parallèles** appelées ' **BUS** ' qui véhiculent les informations sous forme binaire.. Lorsque le fonctionnement est dit synchrone par rapport aux entrées et aux sorties, le cycle de traitement commence par la prise en compte des entrées qui sont figées en mémoire pour tout le cycle.

Le processeur exécute alors le programme instruction par instruction en rangeant à chaque fois les résultats en mémoire. En fin de cycle les sorties sont affectées d'un état binaire, par mise en communication avec les mémoires correspondantes. Dans ce cas, le temps de réponse à une variation d'état d'une entrée peut être compris entre un ou deux temps de cycle (durée moyenne d'un temps de cycle est de 5 à 15 ms [14].

d.Descriptions 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'une part à gérer les instructions du programme.

- **Les modules d'Entrées/Sorties**

L'interface d'Entrées comporte les adresses d'entrées, une pour chaque capteur relié. Tandis que, l'interface de Sorties comporte des adresses de sorties, une pour chaque pré-actionneur. Le nombre de d'E/S varie suivant le type d'automate. Les cartes d'Entrées/Sorties ont une modularité de 8,16 ou 32 voies. Elle 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 gè8re et exécuter le programme. Elle reçoit également des informations en prévenance des capteurs. Il existe dans les automates plusieurs types de mémoires qui remplissent des fonctions différentes :

- La conception et l'élaboration du programme font appel à la RAM et l'EPROM.
- 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} .

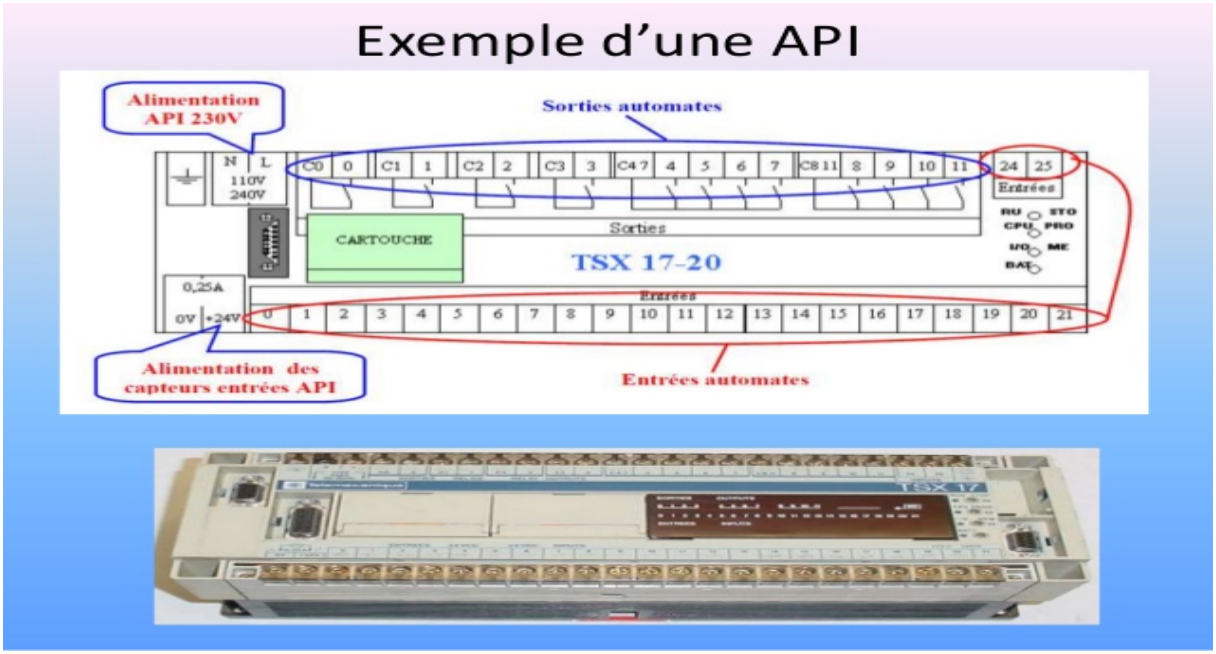

Figure 2.8 Exemple d'un API réel avec ses éléments

e. Les 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**. Cette norme 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** 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 tous types 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.

f. Critères de choix d'un automate

Le choix d'un automate programmable est en premier lieu le choix d'une société ou d'un groupe et les contacts commerciaux et expériences vécues sont déjà un point de départ.

Il faut ensuite quantifier les besoins :

- Nombre d'entrées / sorties : le nombre de cartes peut avoir une incidence sur le nombre de racks dès que le nombre d'entrées / sorties nécessaires devient élevé.

- Type de processeur : la taille mémoire, la vitesse de traitement et les fonctions spéciales offertes par le processeur permettront le choix dans la gamme souvent très étendue.

- Fonctions ou modules spéciaux : certaines cartes (commande d'axe, pesage …) permettront de « soulager » le processeur et devront offrir les caractéristiques souhaitées (résolution, …).

- Fonctions de communication : l'automate doit pouvoir communiquer avec les autres

systèmes de commande (API, supervision …) et offrir des possibilités de communication avec des standards normalisés (Profibus …).

2.2.2 La partie opérative

Appelée parfois partie puissance », la partie opérative d'un automatisme assure la transformation de la matière d'œuvre.

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 les commandes et les actionneurs qui agissent et transforment le produit. Les capteurs recueillent les informations : état ou position du produit, alarmes, etc, traduisent un changement d'état du procédé [16].

La partie opérative d'un automatisme est le sous-ensemble qui effectue les actions physiques et rend compte à la partie commande.

a. Capteurs

Un capteur est un dispositif qui soumis à l'action d'une grandeur physique, qui va fournir un signal pour la partie commande. Le capteur est l'interface entre le monde physique et le monde électrique. On distingue deux fameux type de capteurs ; un analogique et l'autre TOUT ou RIEN.

Les capteurs qui nous intéressent c'est bien que des capteurs analogiques. Ceux-ci servent à transformer une grandeur physique en un autre type de variations d'impédance, de capacité, d'inductance ou tension.

b. Actionneurs

Un actionneur est un dispositif qui transforme l'énergie qui lui est fournis en un phénomène physique qui fournit un travail, modifie le comportement ou l'état d'un système. Comme les moteurs par exemple transforment l'énergie électrique en énergie mécanique.

2.3 Généralité sur les GRAFCET

Le **GRAFCET** est un outil graphique qui permet de décrire le fonctionnement d'un automatisme séquentiel. Il peut être utilisé pour représenter l'automatisme dans toutes les phases de la conception : de la définition du cahier des charges, à la mise en œuvre (programmation d'un automate programmable industriel, utilisation de séquenceurs ou autres technologies) en passant par l'étude des modes de marches et d'arrêts. Le Grafcet repose sur l'utilisation d'instructions précises, l'emploi d'un vocabulaire bien défini, le respect d'une syntaxe rigoureuse et l'utilisation de règles d'évolutions. Il permet, entre autre, d'adopter une démarche progressive dans l'élaboration de l'automatisme. Il décrit les relations entre les sorties et les entrées booléennes du système de commande. Le Grafcet est une représentation alternée d'étapes et de transitions. Une seule transition doit séparer deux étapes.

2.3.1 Élément de base d'un GRAFCET

- **Etape initiale**

L'état initial caractérise l'état du système au début de fonctionnement. Elle se différencie en doublant les cotés 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. À 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. À chaque transition est associée une réceptivité qui exprime la condition nécessaire pour passet 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 capteur
- Etat actif ou inactif d'autres étapes.

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

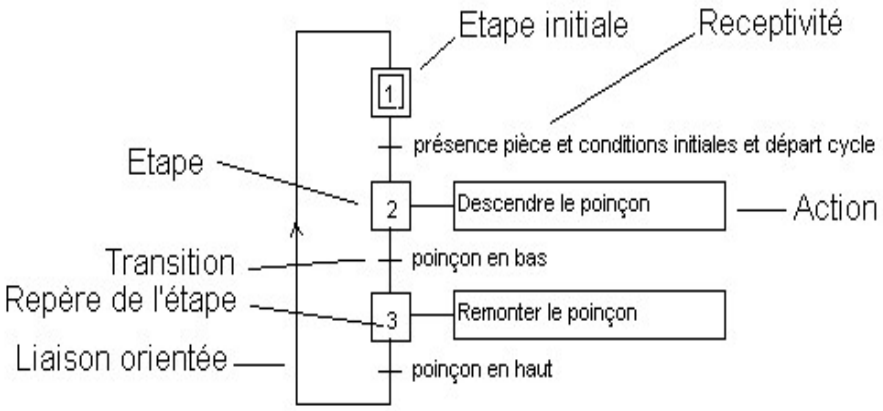

Figure 2.9 Exemple récapitulatif montrant les éléments de base d'un Grafcet

- **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 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.

2.3.2 Règles d'évolution d'un GRAFCET

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

- **Règle 1 : étape initiale**

Les étapes initiales sont celles qui sont activées au début du fonctionnement. L 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 actives**

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

- **Règle 4 : évolution simultanée**

Plusieurs transitions simultanément franchissables sont simultanément franchies.

- **Règle 5 : activation et désactivation simultané d'une étape**

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

3.3.3 Règle de construction d'un GRAFCET

- **Arc orienté**

On relie étapes et transition, qui doivent strictement alterner, grâce à des arcs orientés (figure 2-.10). 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.

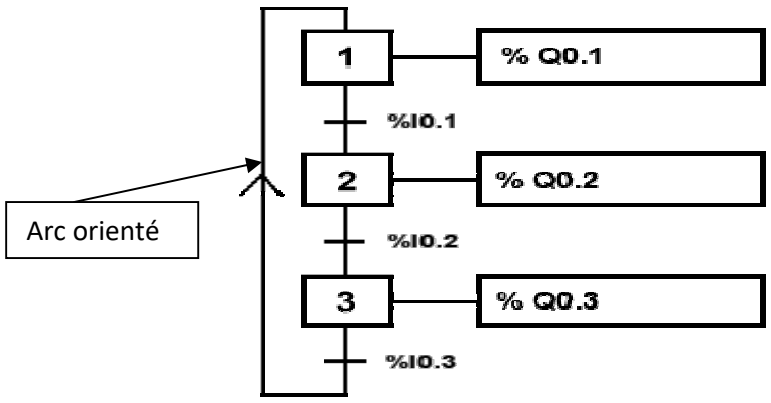

Figure 2.10 Figure mettant en valeur l'arc orienté

- **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 ».

Si plusieurs étape doivent être issus 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 ».

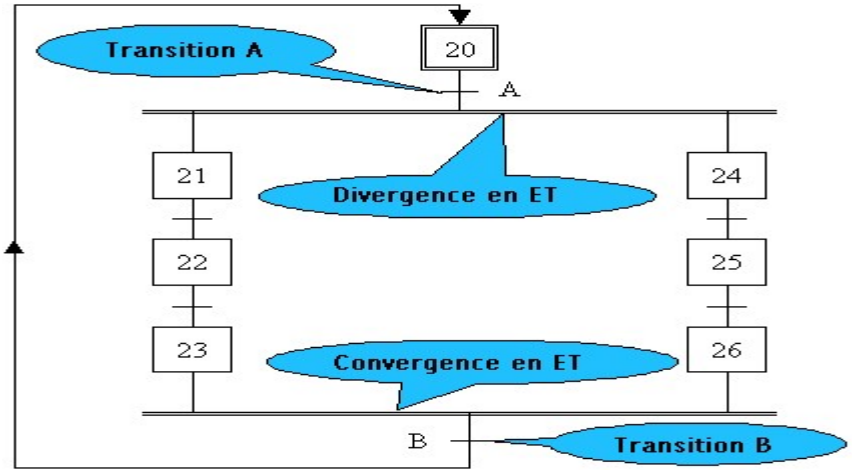

Figure 2.11 La convergence et la divergence en **ET**

- **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 ».

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

La (figure 2.12) au dessous présente la convergence et la divergence en OU :

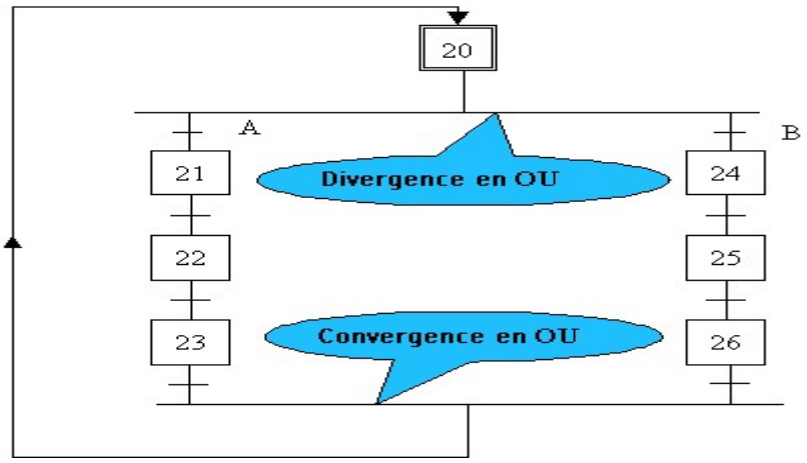

Figure 2.12 La convergence et la divergence en **OU**

- **Saut 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.

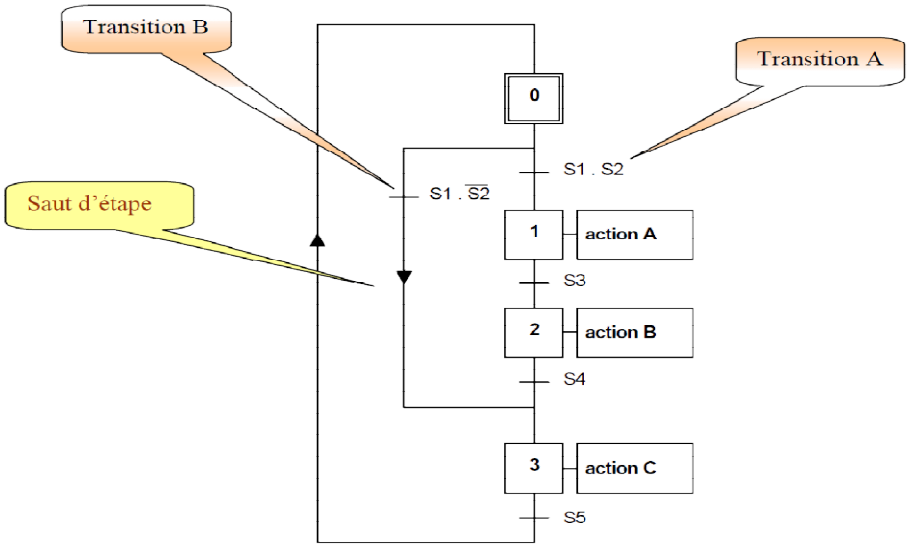

Figure 2.13 Figure mettant en évidence le saut d'étape

- **Reprise d'étape**

La reprise d'étape au contraire, permet de recommencer plusieurs fois si nécessaire une même séquence (**Figure II.14**). 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.

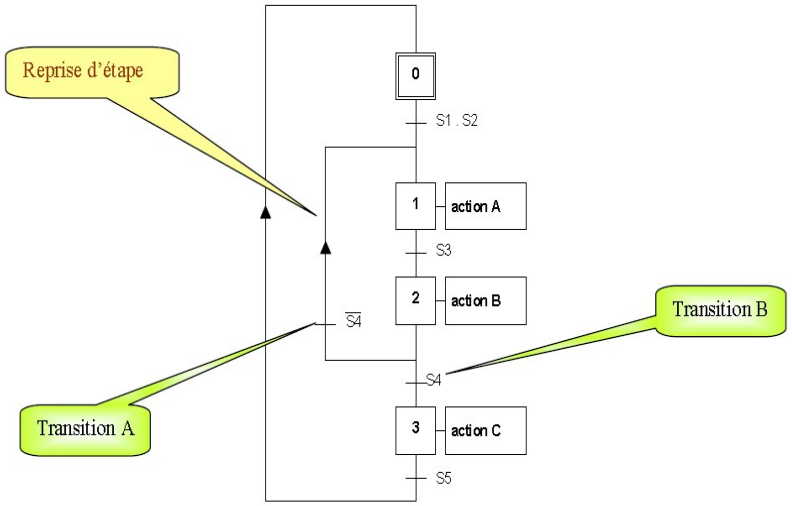

Figure 2.14 Figure mettant en évidence une reprise d'état

2.3.4 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.

2.3.5 Domaine d'application du Grafcet

Le diagramme fonctionnel est indépendant des techniques séquentielles "tout ou rien», pneumatique, électrique ou électronique, câblées ou programmées, pouvant être utilisées pour réaliser l'automatisme de commande. Mais l'utilisation de séquenceurs, d'une part, et d'automates à instructions d'étapes d'autre part, permet une transcription directe du diagramme fonctionnel. Cette représentation graphique concise et facile à lire est aisément compréhensible par toute personne en relation avec le système automatisé, du concepteur à l'utilisateur sans oublier l'agent de maintenance.

Utilisé industriellement, le Grafcet est aussi enseigné dans les options techniques et l'enseignement supérieur.

Depuis les premières publications le concernant et surtout depuis **NF** C03-190 de 1982, cet outil a été travaillé et enrichi par le groupe systèmes logiques de **l'AFCET**.

2.4 Application

L'application réalisée dans ce projet, consiste à créer un système hydraulique sous le logiciel de simulation Flexsim, en utilisant la partie fluide, puis commander le système par un automate programmable, d'autre terme le rôle de l'automate sera l'envoi des actions, par exemple si on veut remplir un réservoir à partir de deux vannes, le fonctionnement sera le suivant :

- Préciser le réservoir par deux niveaux , niveau haut et niveau bas. Chaque niveau est détecté par un capteur.
- Lorsque le liquide atteint le premier niveau, un message envoie vers l'automate qui va transmettre en retour une action à la première vanne de se fermer.
- La deuxième vanne reste ouverte jusqu'à ce qu'elle reçoit un ordre de l'API.

Conclusion

Comme cité précédemment, on veut des simulations qui soient plus proche possible de la réalité d'où, dans ce chapitre on a étudié l'objectif du choix d'une simulation Hardware-In-The-Loop (HIL), les systèmes automatisés et leurs principes de fonctionnements, et les automates programmable industriels. Le chapitre suivant consiste à faire la communication entre la partie virtuelle (simulation HIL) et la partie physique (API), ceci est impossible ce qui nous exige à ajouter un intermédiaire, dans ce cas là on va choisir la carte Arduino qui rend la communication possible.

Introduction

Dans l'optique de bonifier la simulation proposée par ce travail et de permettre aux étudiants de bien assimiler les bases de l'automatique, avec une carte Arduino on aura la possibilité de faire communiquer le système simulé avec FlexSim et API.

Ce dernier chapitre consiste à faire une présentation pour la carte Arduino utilisée et l'automate programmable. Ainsi que, il englobe les résultats du système simulé.

Pour le choix et la conception des systèmes à simulé, 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.

3.1 La carte Arduino

Le projet Arduino est issu d'une équipe d'enseignants et d'étudiants de l'école d' « Interaction Design Institue d'**Ivera** » située, comme son nom l'indique, dans la ville italienne d'Ivera. 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 la distribution de l'alimentation électronique 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 tous connaitre ni tous 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 support disponibles en ligne [17].

3.1.1 Logiciel

L'environnement de programmation Arduino (**IDE** en anglais) est une application écrite en Java inspirée du langage **Processing**. L'IDE permet notamment d'écrire, de modifier un programme et de 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 Schield (qu'on va aborder par la suite), sur la programmation du module GSM, des programmes simples comme allumer et éteindre LEDs. Cela sa pour but d'accélérer la prise en main avec la carte.

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

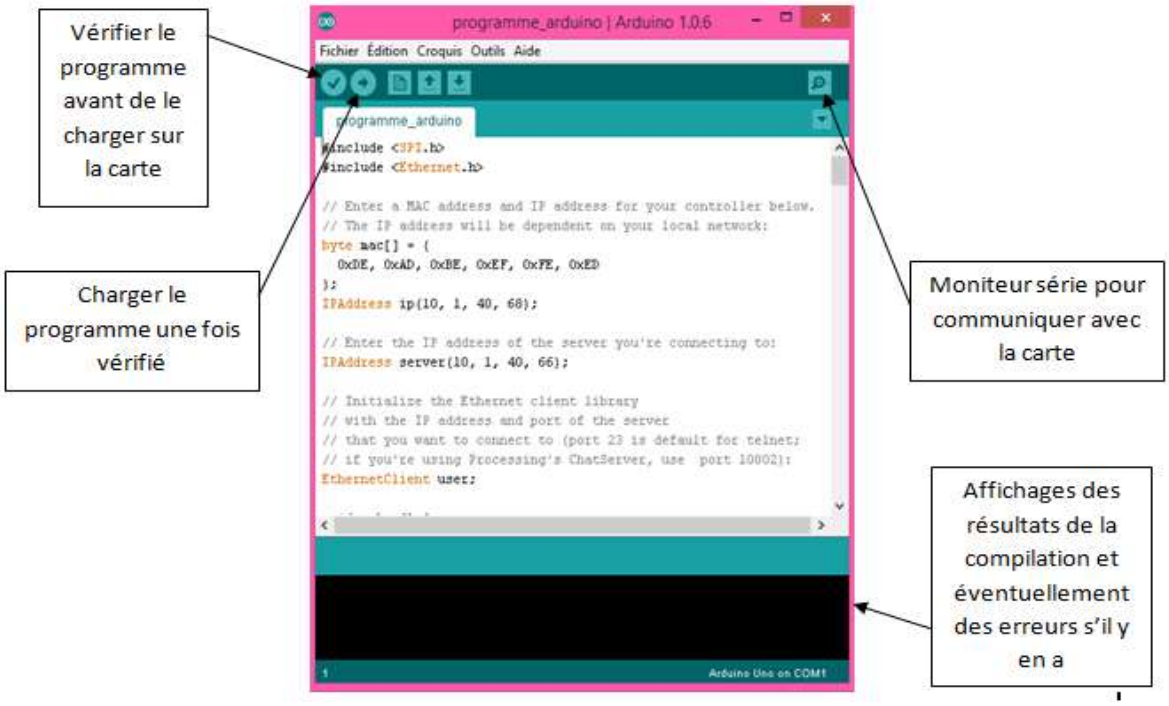

Figure 3.1 Fenêtre de programmation du logiciel Arduino

3.1.2 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 ses commandes son adaptées aux besoins de la carte, qui consistent essentiellement à communiquer avec ses différents connecteurs.

3.1.3 Arduino Mega 2560

La carte Arduino Mega 2560 est une carte à microcontrôleur basé sur un **ATmega2560** .

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** (port série matériel), d'un **quartz** 16 Mhz, d'une connexion USB, d'un connecteur d'alimentation jack, d'un connecteur **ICSP** (programmation « in-circuit »), et d'un bouton de réinitialisation (Reset)[18].

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'intefrace du logiciel Arduino et là voilà fin prête pour accomplir une tâche précise.

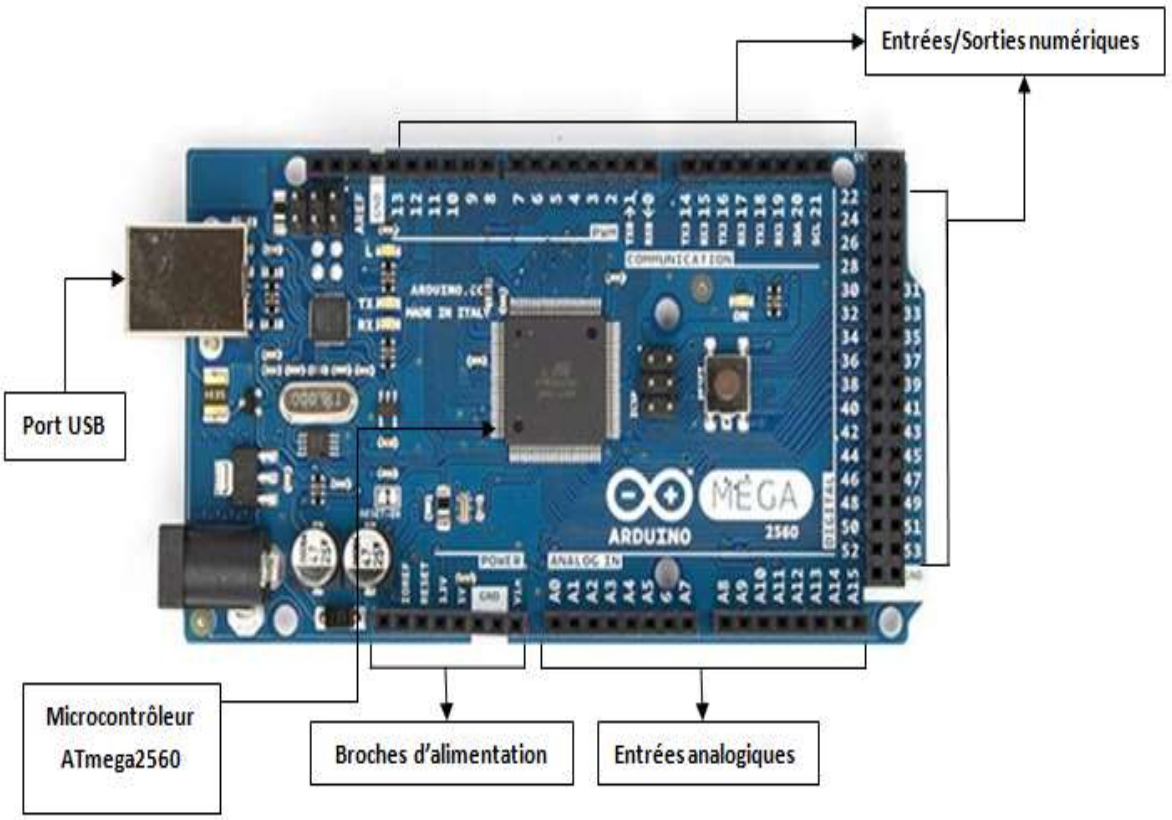

Figure 3.2 Carte Arduino Mega 2560

3.2 Ethernet Schield W5100

La plateforme de développement Arduino bénéficie d'un grand nombre de modules, l'un des ses modules ; l'Ethernet Schield 5100. Il permet d'ajouter une connexion réseau ethernet. Il est basé sur le circuit intégré WIZnet qui fournit une pile réseau (**IP**) adapté, à la fois, au protocole de télécommunication **TCP** et **UDP.**il supporte jusqu'à quatre connexions simultanées. Il suffit d'utiliser la librairie Ethernet pour écrire des programmes qui se connectent à l'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 [19].

Voici une image de l'Ethernet Schield W5100 :

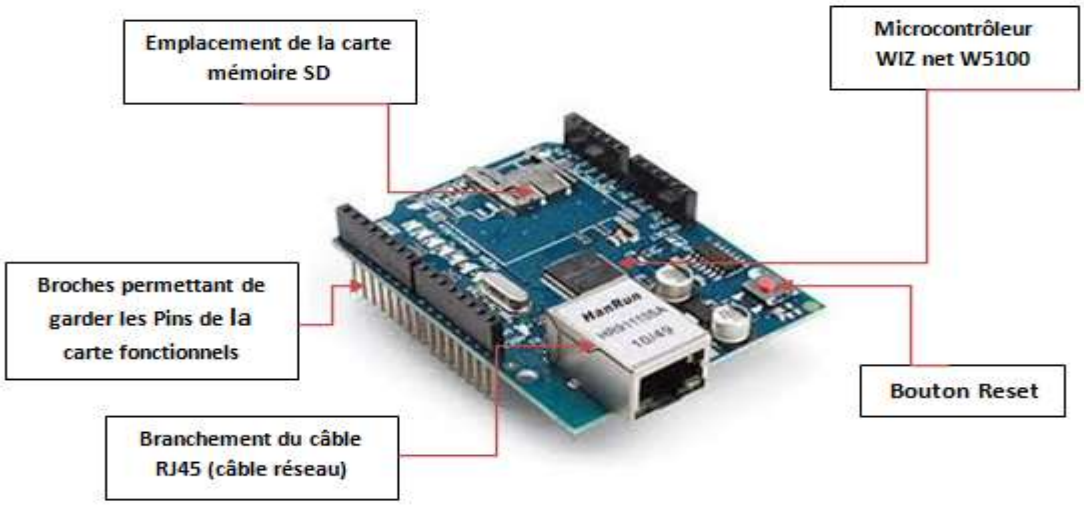

Figure 3.3 Ethernet Schield W5100

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

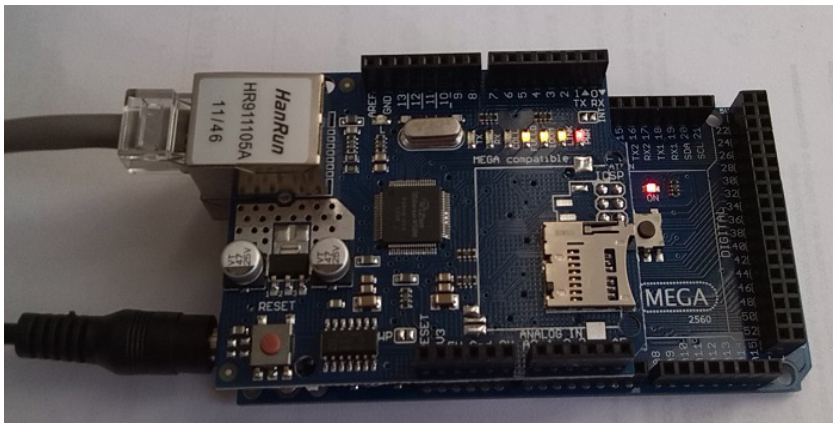

Figure 3.4 Embranchement de l'Ethernet Schield au –dessus de la carte Mega2560

Comme c'était mentionné que la carte Arduino avec l'Ethernet Schield (Figure 3.4) vont établir une communication entre Flexsim et l'API. L'utilisation de la carte Arduino présente un avantage non pas négligeable, 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 E/S de la carte Arduino et celle de l'API sont différentes, elle est de l'ordre 5V pour la carte et de 24V pour l'API. Câbler directement les E/S 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.

3.3 Circuit imprimé à base d'optocopleur

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

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

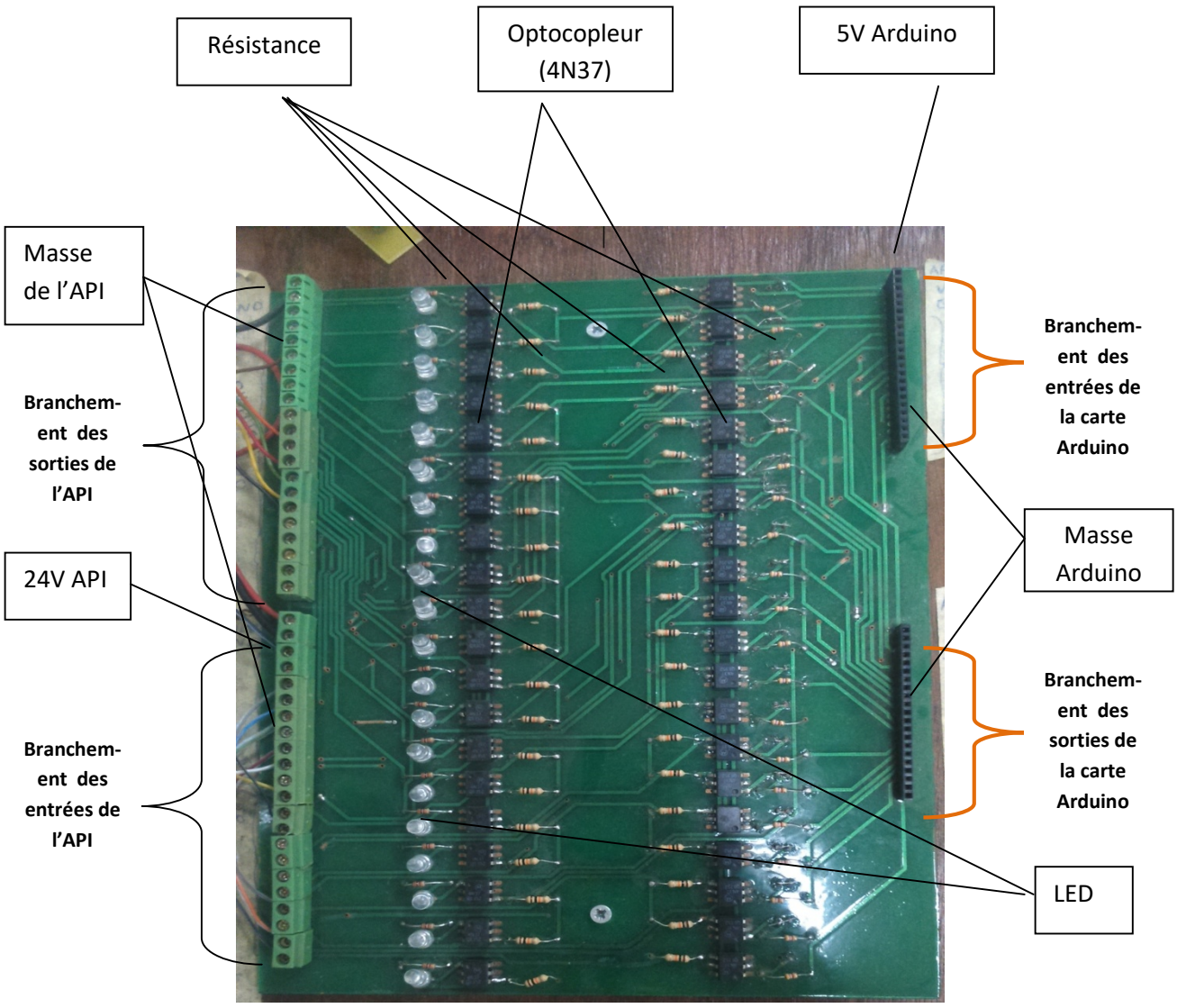

Figure 3.5 Carte d'interface entre la carte Arduino et l'API à base d'optocopleurs

3.4 L'optocopleur et son principe de fonctionnement

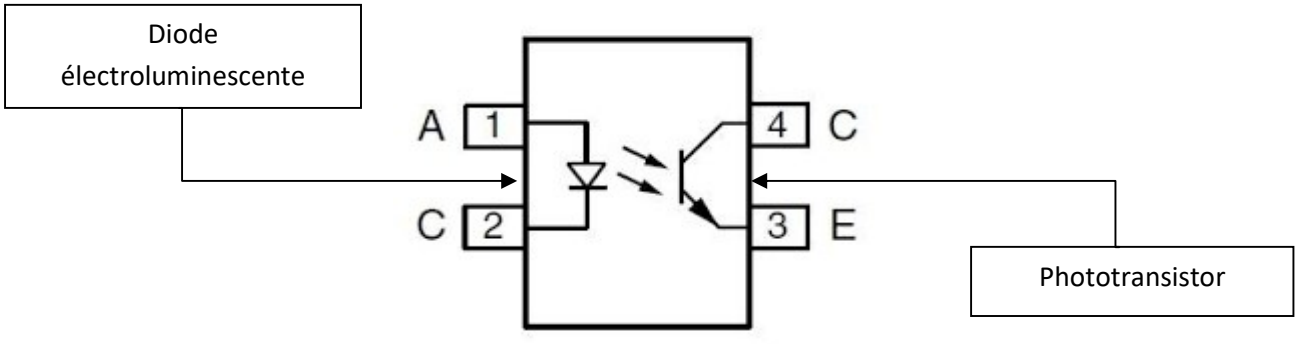

Figure 3.6 Schéma d'un **Optocopleur**

Sur la figure au dessus on a remarqué que l'optocopleur inclut deux composants connus : une diode électroluminescent qui émettant de la lumière infrarouge (émission spontané) 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, non plus du courant, mais de la lumière visible ou infrarouge.

3.5 Le Schneider TWDLCAA24DRF

L'automate programmable utilisé pour la simulation des systèmes, est le Schneider TWDLCAA24DRF. C'est un automate compacte connu par sa flexibilité et sa portabilité. Il est programmé grâce au logiciel TwidoSuite. Il dispose de 24entrées/sorties **TOR** 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 [20].

Voici ci-dessous l'automate Schneider TWDLCAA24DRF (figure 3.7) et le logiciel TwidoSuite (figure 3.8) :

Figure 3.7 Schneider TWDLCAA24DRF

Figure 3.8 Le logiciel TwidoSuite

3.6 Système hydraulique sous FlexSim

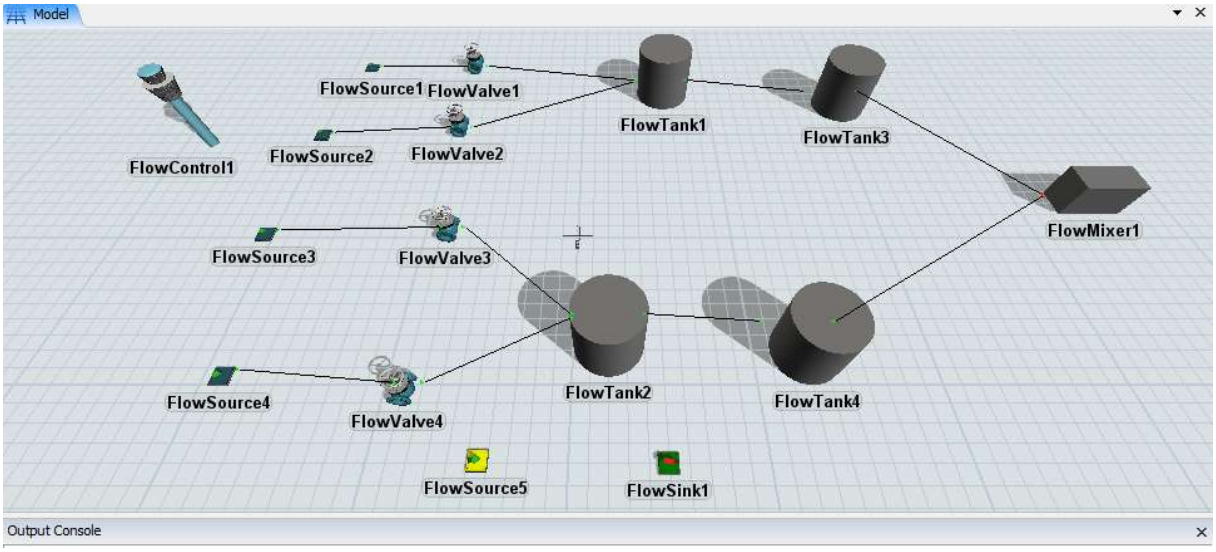

Figure 3.9 Système hydraulique sous le logiciel Flexsim

La figure ci-dessus montre, en effet, un système hydraulique simulé sur Flexsim, dont voici le cahier de charge :

Le système considéré comporte cinq sources (Src1→ Src5), quatre vannes (V1→V4), quatre réservoirs (T1→T4), dont les sorties de T3 et T4 réservoir nommées V5 et V6, un mixeur et un sink. Dont les sources (Src1→Src4) donnent la matière première et la source Src5 vérifie la mise à jour du produit fournit.

Chaque réservoir (T1 et T2) est précisé avec deux niveaux, niveau haut et niveau bas, ceux-ci sont détectés par des capteurs de niveaux. Dont, le capteur C1 détecte le niveau bas et C2 détecte le niveau haut de T1, le capteur C3 détecte le niveau bas et C4 détecte le niveau haut du réservoir T2

Les réservoirs (T3 et T4) sont précisés par un seul niveau, ceci est détecté par un capteur de niveau, C3 détecte le niveau du T3 et C6 Détecte le niveau du T4.

Si le capteur C1 détecte le niveau bas du réservoir T1 alors la vanne V1 s'arrête.

Si le capteur C3 détecte le niveau bas du réservoir T2 alors la vanne V3 s'arrête.

Si le capteur C2 détecte le niveau haut du réservoir T1 alors la vanne V2 s'arrête.

Si le capteur C4 détecte le niveau haut du réservoir T1 alors la vanne V4 s'arrête.

Les réservoirs T1 et T2 sont atteint aux ses niveaux hauts, donc ils ouvrent ses vannes V5 et V6 et les réservoirs T3 et T4 commencent à se remplir.

Si le capteur C5 détecte le niveau du réservoir T3 alors il donne un ordre à sa source T1 de fermer sa sortie V5.

Si le capteur C6 détecte le niveau du réservoir T4 alors il donne un ordre à sa source T1 de fermer sa sortie V6.

Lorsque les réservoirs (T3 et T4) atteignant ses niveaux alors le mixeur commence à se remplir de T3 et T4.

3.7 Schéma Grafcet du système hydraulique

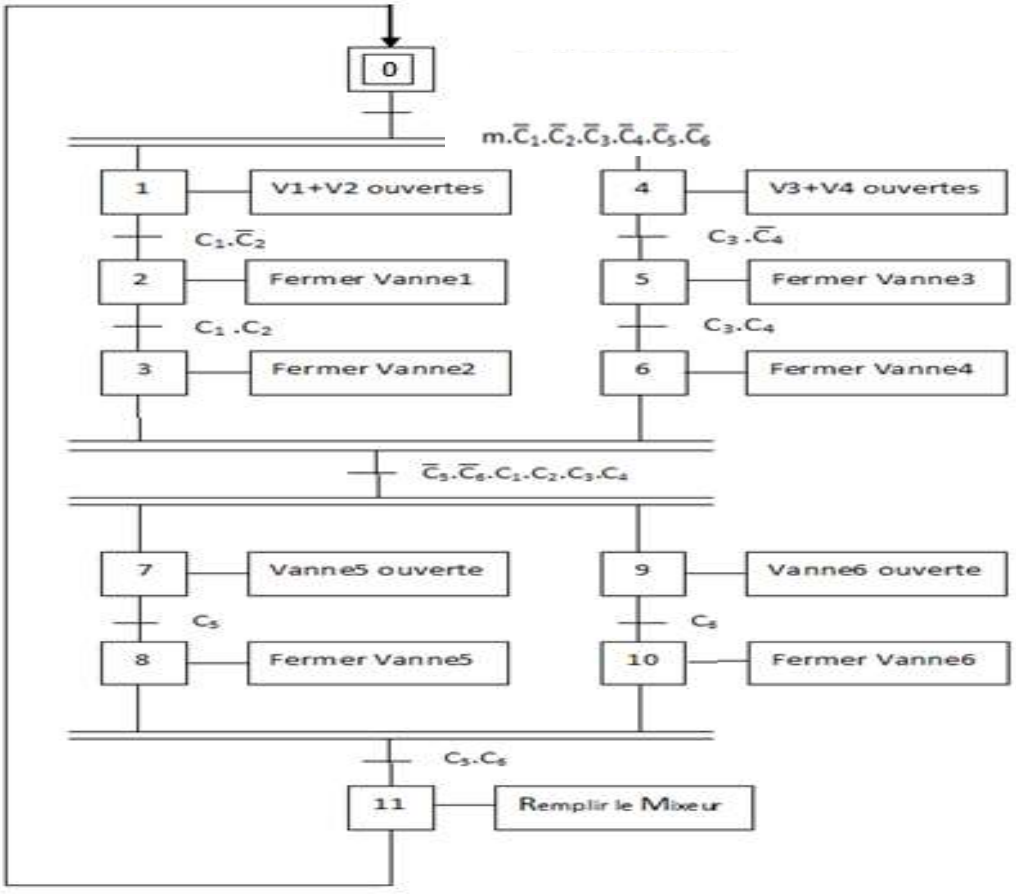

Figure 3.10 Schéma Grafcet du système hydraulique

À l'état initial du système les capteurs des niveaux C1, C2, C3, C4, C5 et C6 sont mis à zéro.

Les vannes V1, V2, V3 et V4 s'ouvrent en parallèle (étape1 et 4), (convergence en ET).

C1=1 et C2=0, fermer la vanne V1 (étape 2)

C3=1 et C4=0, fermer la vanne V3 (étape 5)

C1=1 et C2=1, fermer la vanne V2 (étape 3)

C3=1 et C4=0, fermer la vanne V1 (étape 6)

Les vannes V1, V2, V3 et V4 sont fermées et, les capteurs C1, C2, C3 et C4 sont à 1. C5=0 et C6=0, les réservoirs T3 et T4 commencent à se remplir en parallèle (convergence en ET), (étape 7 et 9).

C5=1, fermer la vanne V5 du réservoir T1 (étape 8)

C6=1, fermer la vanne V6 du réservoir T2 (étape 10)

Les vannes V5 et V6 sont fermées, les capteurs C5 et C6

Le mixeur ouvre ses entrées de fluide et se remplit

Le système finit à exécuter toutes les actions d'ouvertures et de fermetures, on revient alors, à son état initial.

3.8 Application

Dans cette application on a, au départ, établi la communication entre le logiciel de simulation Flexsim et l'API, avec un intermédiaire représenté par la carte Arduino. Dont la connexion entre la carte Arduino et le logiciel Flexsim se fait par un réseau Ethernet.

Le principe de fonctionnement de cette application est le suivant :

Flexsim étant programmé comme un serveur et Arduino comment client. Tout d'abord on démarre la simulation à partir de Flexsim (serveur), la communication s'établit instantanément avec le duo Arduino-Ethernet, puis, la carte Arduino reçoit, à travers un

réseau, les messages envoyés par Flexsim : il s'agit de l'état des capteurs, elle les convertit en signaux et les envoie à l'automate à travers une carte d'interface **(Figure 3.5)**, qui va permettre le passage du 5V au 24V et vice-versa. L'API activera ou désactivera ses sorties en fonction de ses signaux. À la fin, la carte Arduino intercepte ces derniers (toujours à travers la carte d'interface), les convertit en actions et les envoie sous forme de messages à Flexsim, qui changera d'état en fonction des messages reçus.

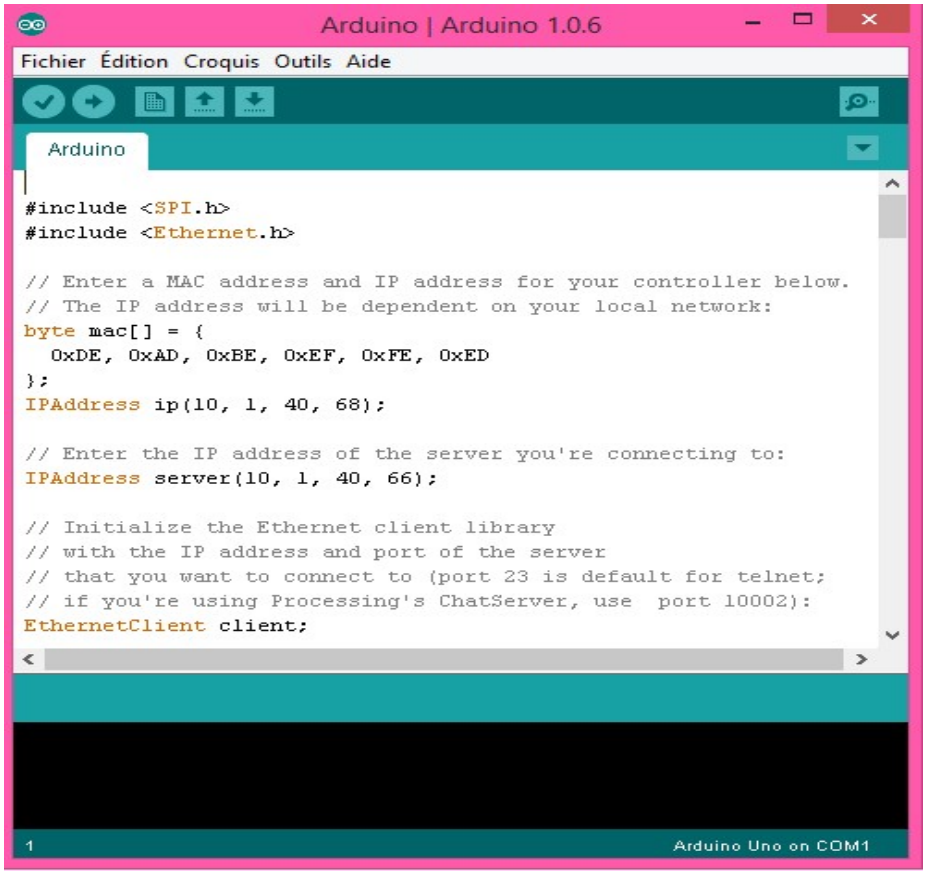

Figure 3.11 Programme de création d'un réseau Ethernet

Pour commencer de créer le réseau, il va falloir configurer notre module Ethernet pour qu'il puisse travailler correctement. Pour cela, il va falloir donner une adresse MAC et une adresse IP.

De manière programmatrice, on a le programme mentionné au dessus (Figure III.11)

Alors l'échange d'informations entre Flexsim et Arduino est réalisé.

Au lancement du logiciel de simulation Flexsim les réservoirs commencent à se remplir, comme il apparait au-dessous :

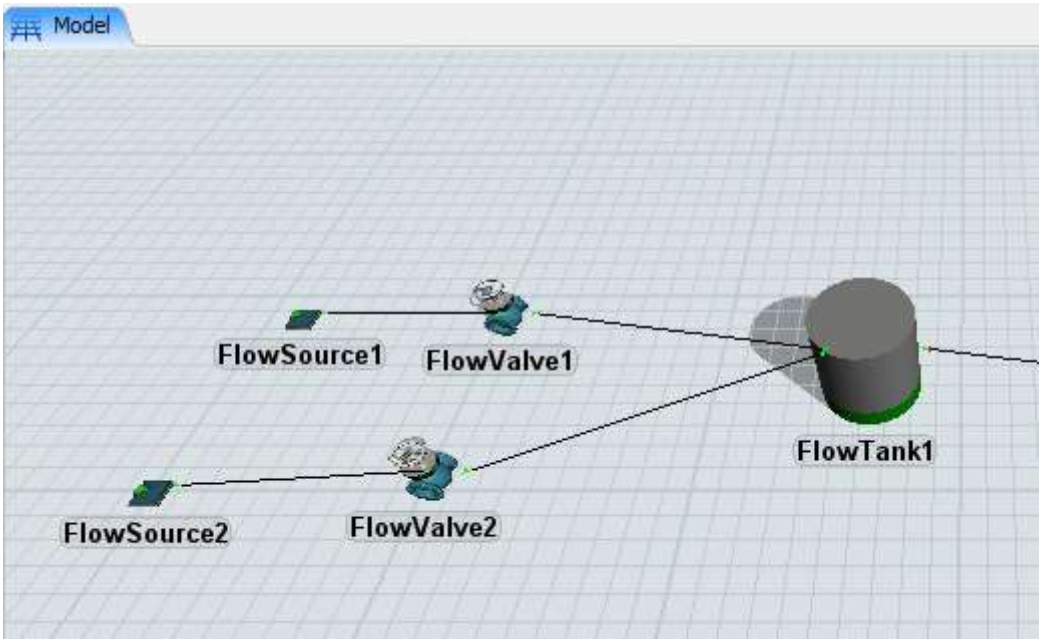

Figure 3.12 Remplissage de T1 à partir de V1 et V2

La couleur verte montre la présence du liquide

Lorsque le réservoir T1 est allé à son niveau bas, un message est envoyé à la carte Arduino

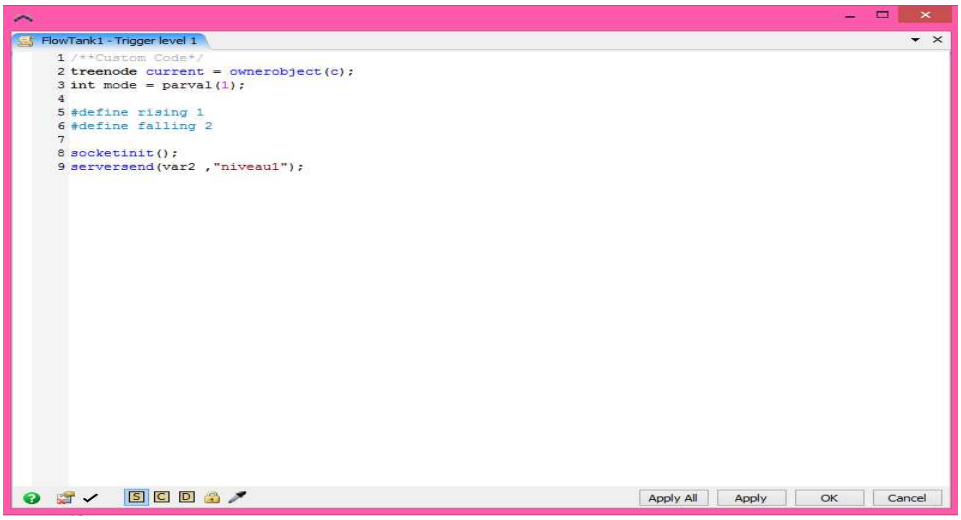

Figure 3.13 Programme qui assure l'envoi du message en FleximScript

La carte reçoit le message de Flexsim par socket et l'envoi à l'API en passant par le circuit d'interface.

En retour, l'automate fait le traitement et envoie une action (fermeture ou ouverture) sous forme d'un message à la carte Arduino en passant par le circuit, la carte l'intercepte et l'envoie à Flexsim pour exécuter l'action

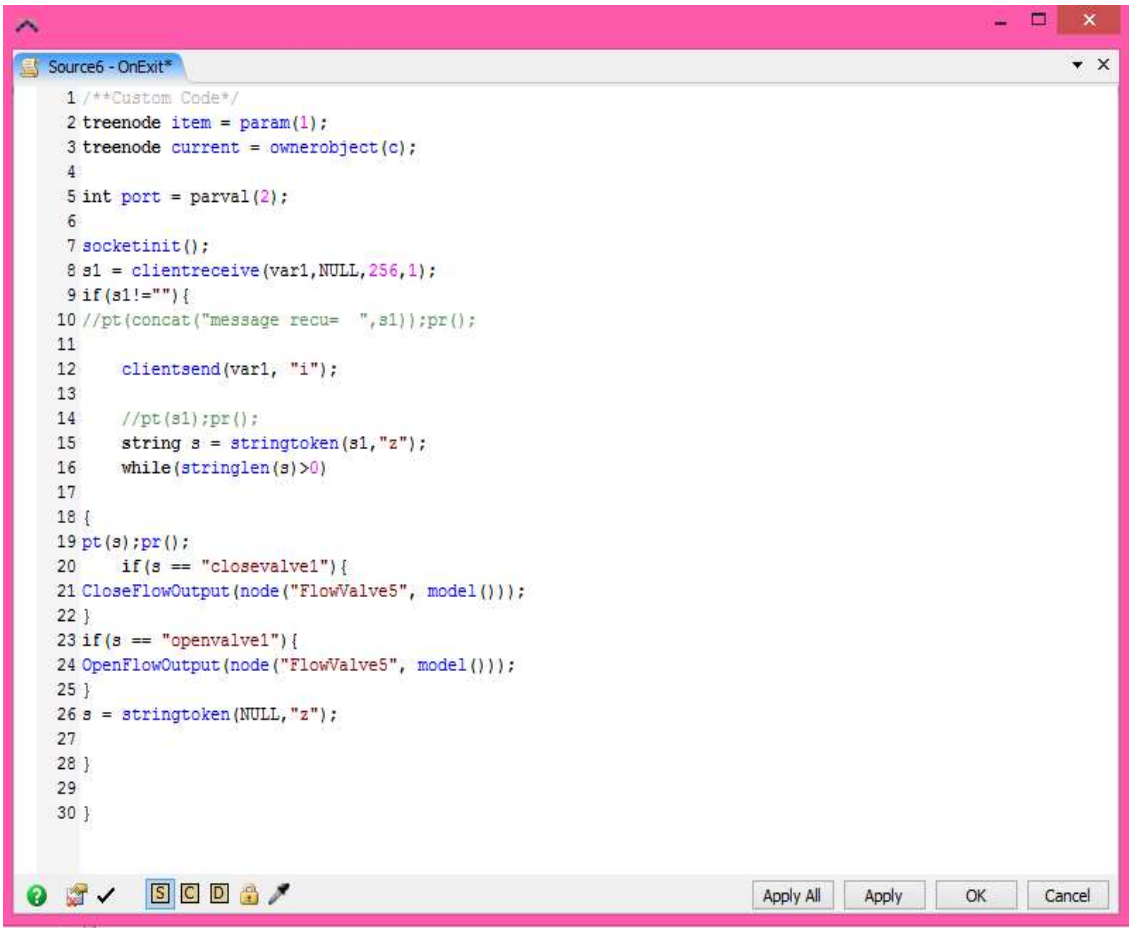

Figure 3.14 Programme sous FleximScript pour le message d'action

Toutes les autres requêtes du système simulé, se font de même manière, remplissage des réservoirs et, fermeture et l'ouverture des vannes.

On peut afficher des messages sur la console, qui assure que l'action est bien faite

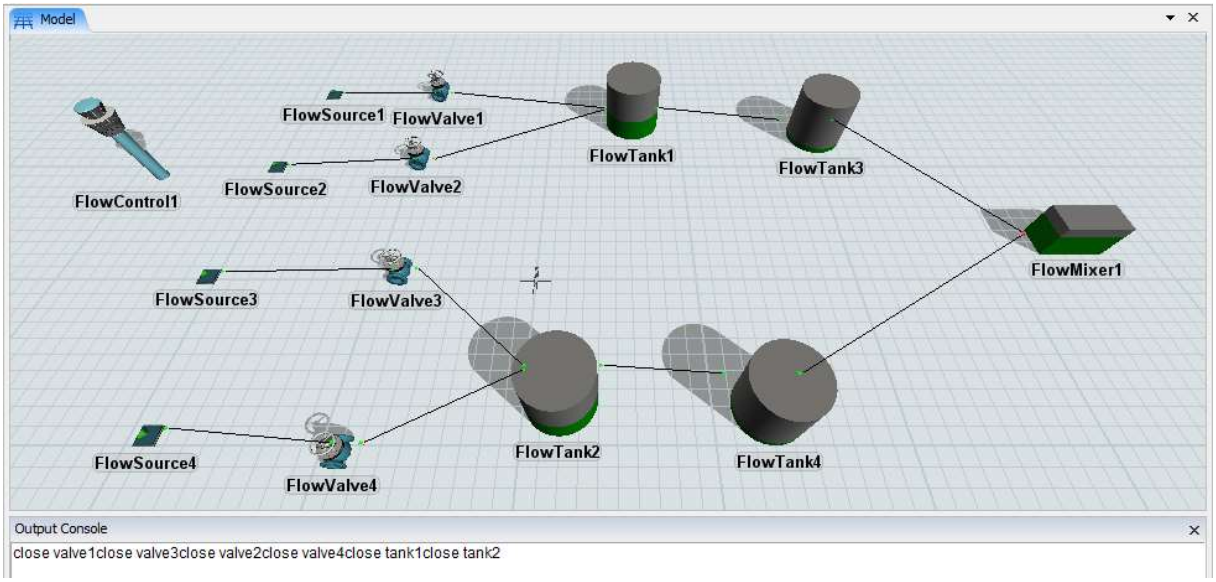

Figure 3.15 Des messages affichés montrant que les actions sont bien faites

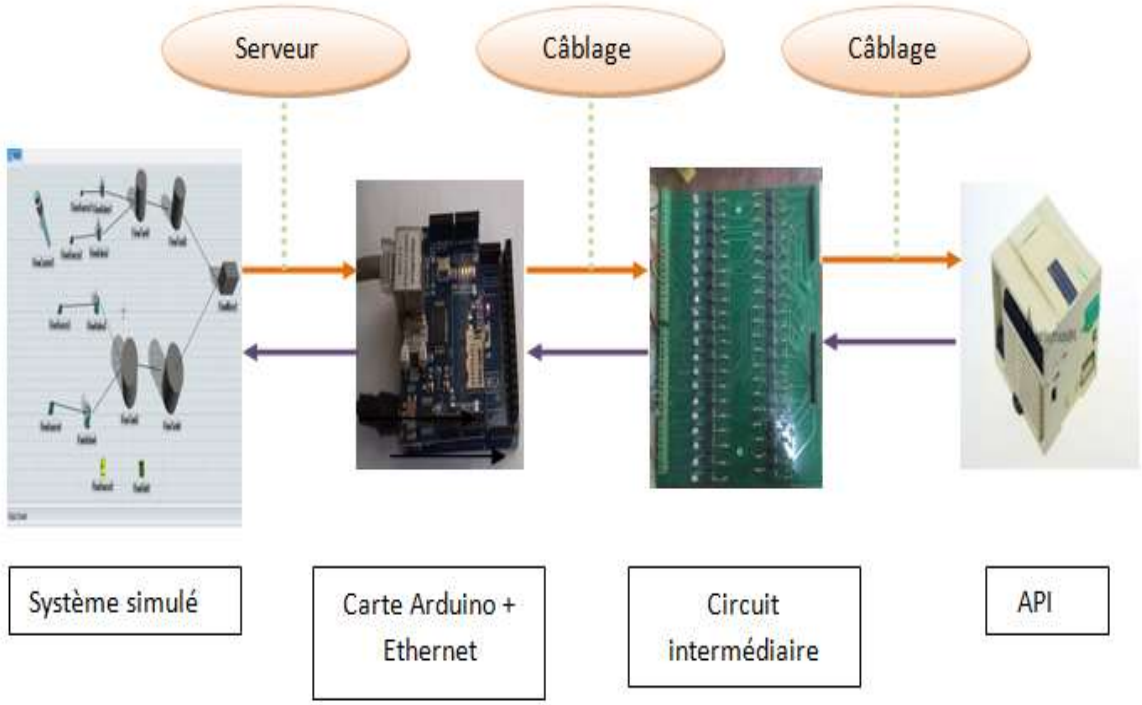

Figure 3.16 Figure qui illustre le principe de fonctionnement de l'application

3.9 Résultat

La commande du système hydraulique par un API a fonctionné avec succès, ainsi que toutes les communications mises en œuvre. C'est le cas de l'envoi et réception des messages entre Flexsim et la carte Arduino en plus de la communication entre Arduino et l'API.

Conclusion

Ce chapitre comporte le système hydraulique simulé et commandé par un automate programmable à l'aide d'une carte programmable Arduino, un réseau est établi, est nécessaire, afin d'effectuer la communication entre le logiciel de simulation Flexsim et la carte Arduino, et les résultats obtenus lors de la réalisation d'application.

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 on été présentées ainsi que le choix du logiciel Flexsim pour la simulation des systèmes industriel.

Par suite, on a introduit l'architecture de l'application qui consiste à commander des systèmes simulés, par un automate programmable industriel.

Afin d'améliorer la simulation et permettre aux étudiants de bénéficier de la totalité des fonctionnalités d'un automate, une couche logicielle qui fait le lien entre l'API et le logiciel de simulati Flexsim a été établi par une carte programmable Arduino et un Ethernet Schield. 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 on été rencontrées sans être exhaustif, nous citons :

- Pour la réalisation de ce projet nous avons dû apprendre à manipuler deux logiciels : le logiciel de simulation Flexsim et celui de la programmation d'Arduino.
- 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 le 5V.

Conclusion générale

La manipulation de la carte Arduino représente un véritable plus dans notre apprentissage et nous ouvre d'autres perspectives d'avoir en raison de nombreuses applications qu'elle peut accomplir notamment dans le domaine de l'automatique.

Comme perspectives, il est préférable d'établir la communication directement entre le logiciel de simulation (PC) et la carte Arduino sans passer par d'autres serveurs (java par exemple). Cela évitera la perte du temps en essayant de programmer avec d'autre logiciel (java par exemple), surtout pour ceux qui ne le maitrise pas.

Il serait préférable, de trouver une solution pour câbler plus rapidement les entrées/sorties de l'automate, en évitant les tâches contraignantes, comme l'utilisation d'un tournevis. Concevoir des cartes qui joueront le rôle d'interface de câblage va permettre d'éviter la perte du temps et de faire des erreurs du câblage.