

RAPPORT DE STAGE – Master 2 IWOCS

Réalité virtuelle et Automates Programmables Industriels (API): « Un modèle hybride pour l'enseignement de l'informatique industrielle »



Tuteurs entreprise:

Florence LECROQ
Jean GRIEU
Hadhoum BOUKACHOUR

Tuteur enseignant:

Laurent AMANTON

Stagiaire:

Youssef IBRAHIM ELMi

REMERCIEMENT

Je tiens à remercier toutes les personnes qui ont participé de près ou de loin à la réussite de mon stage et à la rédaction de ce rapport.

D'abord, j'adresse ma gratitude envers les trois enseignants chercheurs **Mme LECROQ Florence**, **M. GRIEU Jean** et **Mme BOUKACHOUR Hadhoum** pour avoir proposé un projet de type "Agile" et qui m'ont offert l'opportunité de découvrir le monde de la réalité virtuelle et des Automates Programmables Industriels (API). Ainsi, j'ai pu acquérir de nouvelles compétences dans des domaines qui m'étaient méconnus auparavant.

Ensuite, je remercie le responsable de notre formation **M. Laurent Amanton** pour m'avoir permis d'effectuer ce projet dans le cadre du stage de la formation Master 2 IWOCS (Ingénierie Web, Objets Communicants et Systèmes complexes).

Je remercie mes deux camarades de stage qui avec leur sympathie et leur sérieux ont rendu ce projet d'autant plus agréable qu'il ne l'aurait été s'il avait été en solitaire.

Je remercie également la région Normandie avec le programme FEDER sans qui le projet n'aurait pu avoir lieu.

SOMMAIRE

INTRODUCTION	4
I. Présentation du projet	5
A. Objectif du projet	5
B. Les missions	5
C. Les trois questions fondamentales	6
D. Contraintes techniques	7
1. Factory I/O	
2. PC WORX	
3. WebVisit	
4. SQL Server Management Studio 2012	
5. Cogent Datahub	
II. Analyse et conception	12
A. Interconnexion de tous les acteurs	12
B. Stockage des variables	13
C. Diagramme de gantt	14
D. Les étapes de la réalisation	15
III. Réalisation	16
A. Cahier des charges	16
1. Les intervenants humains ou non	
2. Enoncé des fonctions de service	
3. Mes missions	
B. Mise en place de l'environnement virtuel	17
1. Le scénario	
2. La répartition entre les trois automates	
C. Programmation de la scène	20
1. Les langages utilisés	
2. Exemple de la mise en caisse	
D. Supervision	26
1. Les attentes d'une supervision	
2. Le choix du design	
3. Liaison avec les automates	
E. Les bases de données	31
1. Leurs fonctions	
2. L'organisation des deux bases de données	
3. Création	
F. Difficultés rencontrées	37
IV. Résultats	38
V. Tests et validations	41
Conclusion	42
Lexique	43
Annexe	44

INTRODUCTION

C'est au **PIL** (Pôle Ingénieur et Logistique) que j'ai choisi de faire mon stage. Inauguré en Octobre 2016, le PIL reflète la volonté de l'Etat, de la Région Haute-Normandie, de la CODAH et de l'Université du Havre à vouloir augmenter le potentiel de la région havraise dans les domaines tels que la recherche, l'enseignement et l'innovation et tout ceci dans le domaine de la logistique.

Le stage s'est déroulé du lundi 12 mars 2018 au vendredi 13 juillet. Nous étions trois étudiants sur le même projet, à savoir un **DUT Informatique**, un **DUT Génie Electrique et Informatique Industrielle** et moi même un **Master 2 Informatique**.

D'autre part, en 2010 a été adopté par les États membres de l'Union Européenne la « Stratégie Europe 2020 ». Cette stratégie a pour but de créer une croissance intelligente et durable dans toute l'Union Européenne, et pour financer ceci le « Fonds Européen de Développement Régional » (**FEDER**). Cette organisation agit sur la cohésion territoriale, sociale et économique. En grossissant les traits, elle permet de corriger les déséquilibres présents entre les diverses régions.

De cette envie d'uniformisation sont nés des projets. Le sujet en cours est le suivant « **CLASSE phase 2** », soit « Corridors Logistiques : Application à la vallée de Seine et Son Environnement phase 2 », tout le projet met à l'étude la circulation des marchandises dans un corridor logistique et à ses interfaces. Et dans cette étude intervient tout un tas de sous projets voir de sous, sous projets. C'est de cet élan d'innovation que découle le sujet de mon stage, d'où nos problématiques :

- **Comment, de nos jours, utiliser la réalité virtuelle pour simuler un processus logistique d'entrepôt?**
- **Comment peut-on s'immerger dans un entrepôt virtuel via un outil de virtualisation ?**
- **Comment faire interagir des automates réels avec une simulation virtuelle ?**

Pour répondre à ces problématiques, je vous propose de commencer par le cahier des charges du projet, puis notre démarche découlant de ce dernier et enfin nos résultats pour finir avec les phases de tests et de validations.

I. Présentation du projet

A. Objectifs du projet

Le projet découle directement du programme FEDER (Fonds Européen de Développement Régional) CLASSE 2-Action 5.5. L'objectif de ce projet est donc de concevoir un entrepôt logistique virtuel entièrement automatisé qui introduit le concept de virtualisation hybride, associant des automates programmables industriels avec des processus virtualisés. Nous avons également un module de supervision permettant d'apprécier le bon fonctionnement des processus concernés, mais aussi pour visualiser l'impact des commandes des clients sur le système. Autrement dit, c'est une vérification de la faisabilité d'une connexion « temps réel » entre un environnement virtuel et du matériel physique. Cette vérification permettra d'enrichir les travaux du programme CLASSE 2 en expérience et méthodologie pour la mise en place d'une installation ressemblant à celle de ce stage.

B. Les missions

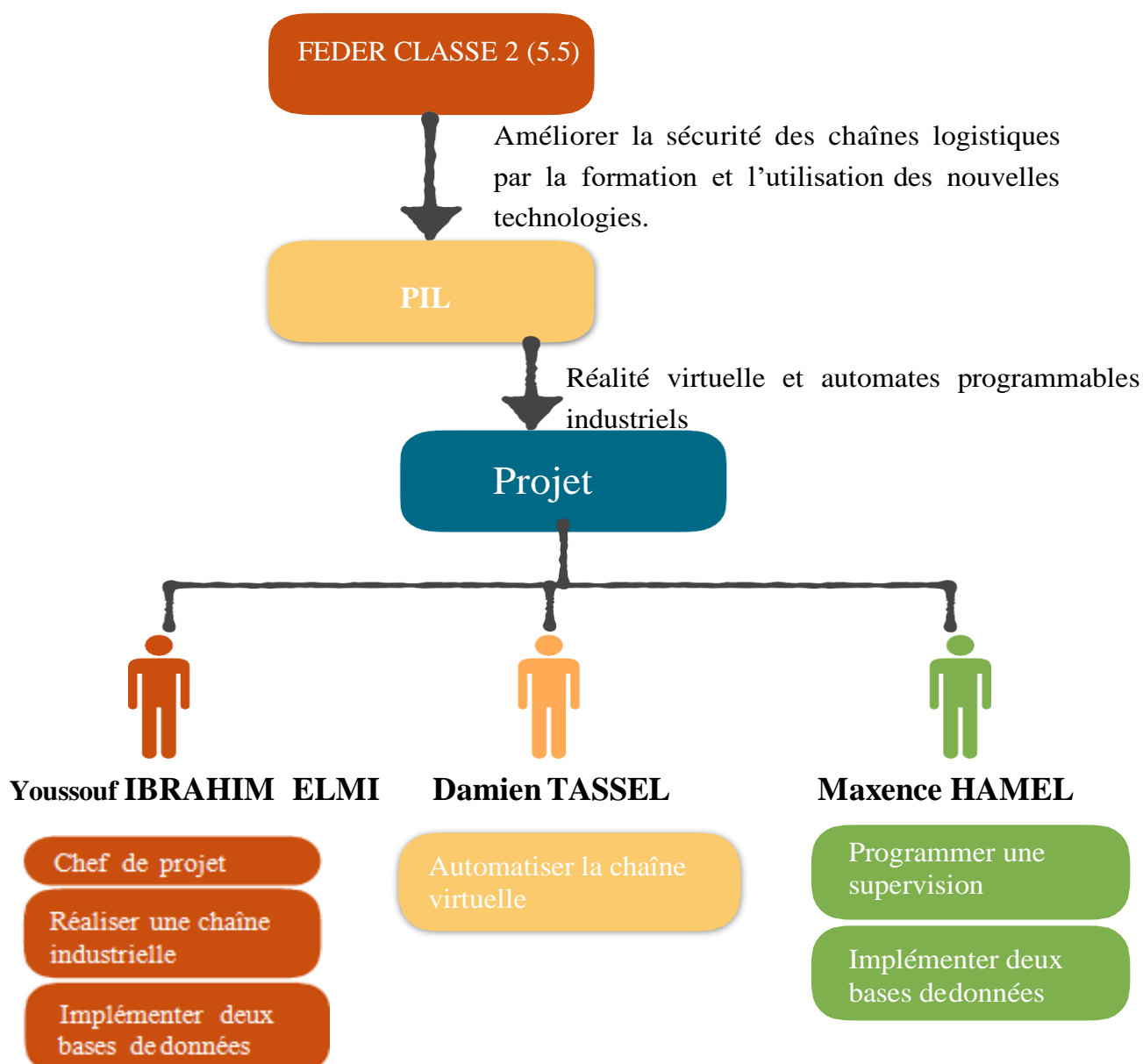


Figure 1 : Origine et missions du projet

C. Les trois questions fondamentales

Cette bête à corne répond aux trois questions auxquelles il faut obligatoirement répondre au début de chaque projet.

À savoir :

- À qui rend-t-il service ?
- Sur quoi agit-il ?
- Dans quel but ?

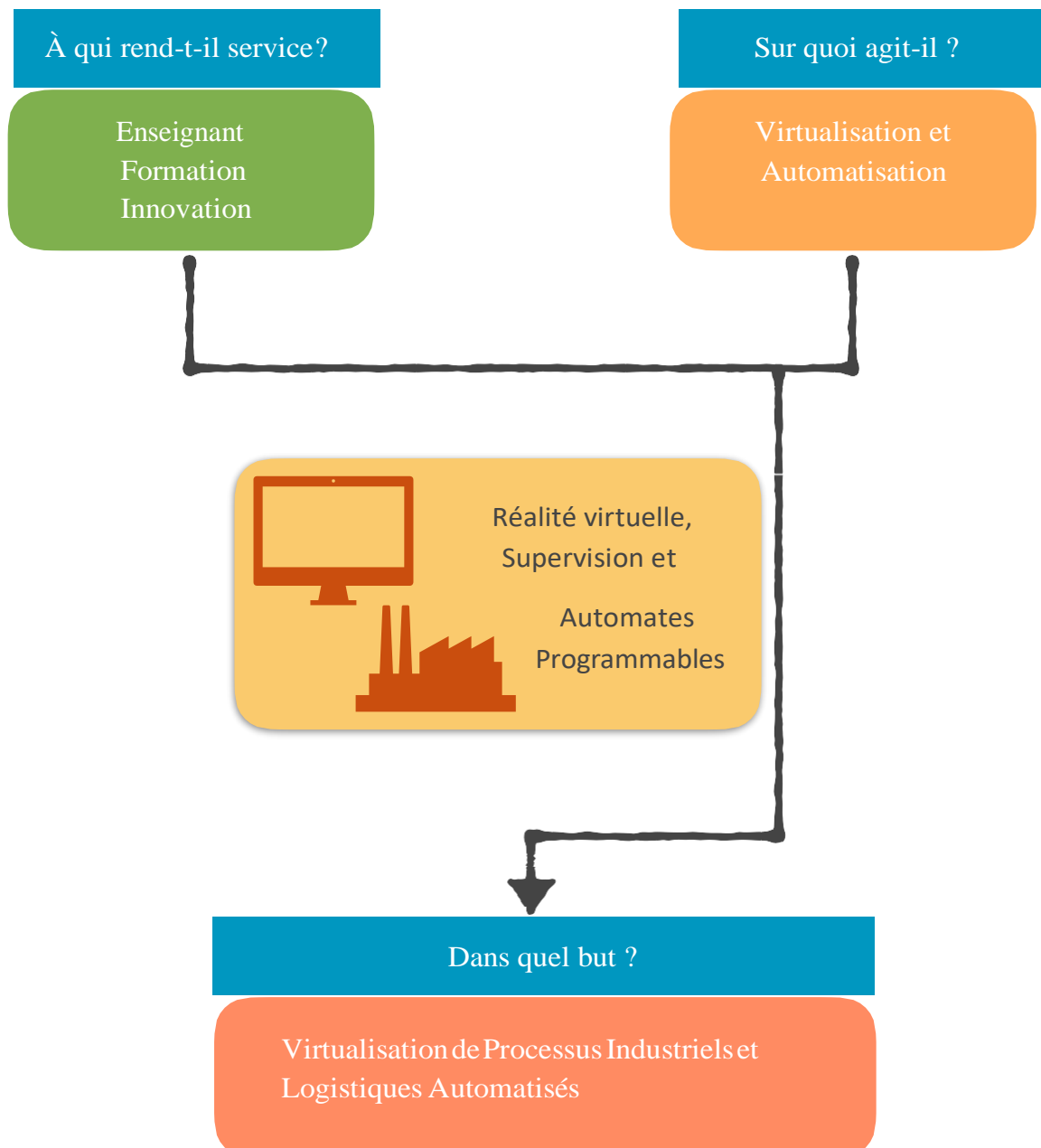


Figure 2 : Bête à corne exposant l'énoncé du besoin

Cette bête à corne nous permet de répondre aux trois questions aux quelles il faut obligatoirement répondre au début de chaque projet.

D. Contraintes techniques

Dans cette partie, nous allons présenter les différents outils utilisés pour mener à bien ce projet. Dans ce projet le choix des outils techniques à utiliser n'étaient pas à faire, en effet les trois automates étant déjà achetés, le logiciel de programmation l'était aussi de ce fait. Il en est de même pour la mise en place de l'environnement industriel virtuel. Ces outils utilisés sont *Factory i/o*, *PC Worx*, *WebVisit* et *SQL* avec pour chacun une utilisation particulière. Nous allons donc présenter ces quatre outils afin de justifier leurs utilisations dans la résolution du problème posé.

1. Factory I/O

Factory I/O est un logiciel de simulation 3D permettant de concevoir et de simuler des systèmes industriels complets. Il est disponible uniquement pour Windows. Ce logiciel est fourni sous sept versions mais nous n'utilisons que celle du « Modbus et OPC Edition » pour les communications Modbus et OPC (voir L_7 et L_8 en lexique page 40).

Avec *Factory I/O*, l'utilisateur peut créer des systèmes en utilisant des pièces industrielles intégrées, basées sur ce qui se trouve couramment dans les installations industrielles typiques. Avec une interface simple et intuitive, les utilisateurs peuvent rapidement construire des systèmes par simple glissé-déposé des pièces désirées dans la scène et ensuite tester leurs capteurs et actionneurs immédiatement en exécutant la simulation à tout moment pendant le processus.

Les utilisateurs peuvent construire un système à partir de zéro ou utiliser l'un des systèmes déjà construit pour commencer. L'utilisation des systèmes de simulation donne aux utilisateurs un environnement sans risque très motivant qui stimule l'apprentissage naturel et permet d'éviter les dommages liés à la mise en œuvre d'équipements industriels dangereux.

1.1. Factory I/O Modbus & OPC Edition

Factory I/O (Modbus & OPC Edition) est donc l'édition que l'on a utilisée dans notre projet. Elle permet de réaliser des parties opératives pour simuler des communications Modbus et OPC.

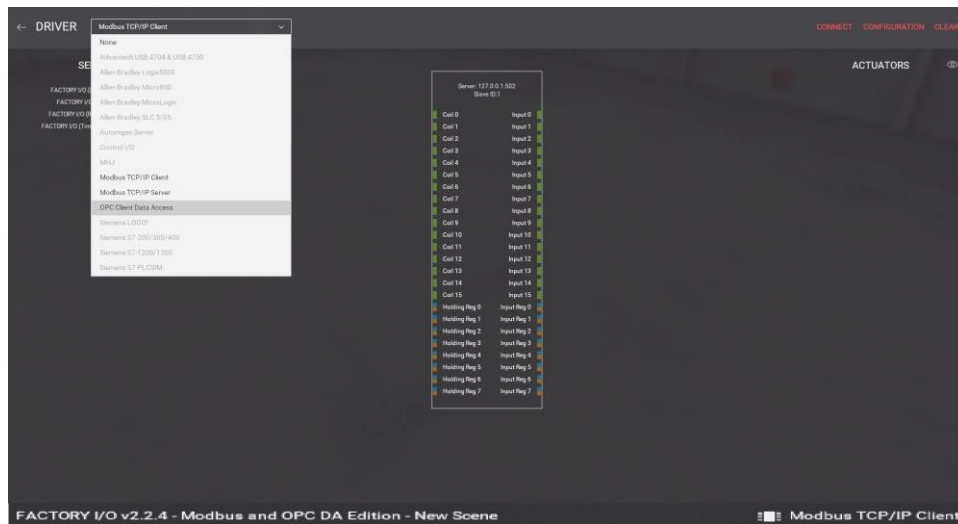


Figure 3 : Configuration du driver OPC de *Factory I/O*

1.2. Aperçu du logiciel

Factory I/O est donc un logiciel présentant un grand entrepôt vide dans lequel nous glissons dépose les objets qui sont fournis dans la bibliothèque à droite de l'illustration ci dessous. Pour simuler rien de plus simple il suffit de cliquer sur le bouton « I » et même si vous voulez stopper la simulation avec l'icône « II ». Vous pouvez choisir ou non d'afficher toutes les variables de la scène tout en modifiant aussi le style de caméra et sa position.

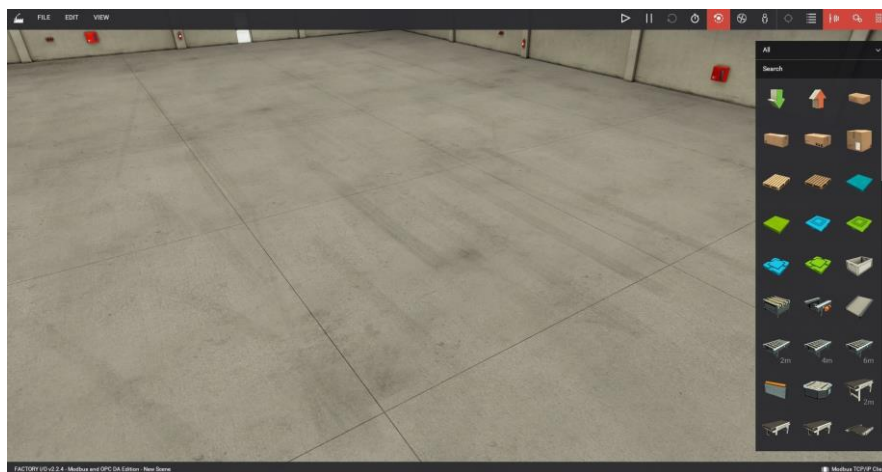


Figure 4 : Scène vierge sur *Factory I/O*

2. PC Worx

PC WORX est un logiciel de programmation d'automate plus particulièrement ceux de Phoenix contact. Plusieurs langages sont à disposition respectant tous la norme « CEI 61131 » (voir *L_4 page 40*). Le but de *PC WORX* est de transférer un programme dans un automate pour que celui-ci exécute le programme automatiquement sur un logiciel de simulation comme *Factory I/O*.

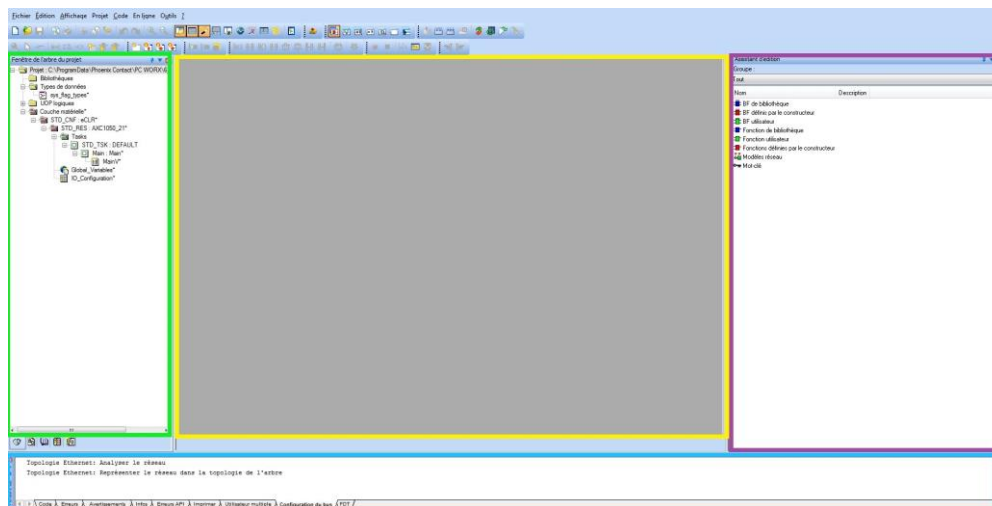


Figure 5 : Page principale de PC WORX

L'image ci-dessus illustre la page d'accueil de *PC WORX*, la zone verte représente l'arborescence du projet, la zone jaune représente la feuille de programmation là où l'on écrit le programme, la zone violette est un raccourci à la programmation en *LADDER*. C'est dans cette zone où se trouvent les blocs fonctionnels et enfin la zone bleue qui est utilisée pour la gestion des erreurs lors de la compilation du programme.

3. WebVisit

Développé par Phoenix Contact, *WebVisit* permet de construire une supervision pour tout type d'automate de la marque Phoenix Contact. La supervision est alors téléchargée dans l'automate via un câble Ethernet, elle est ensuite accessible en tapant l'adresse IP (voir *L_1 page 40*) de l'API sur Internet Explorer exclusivement.

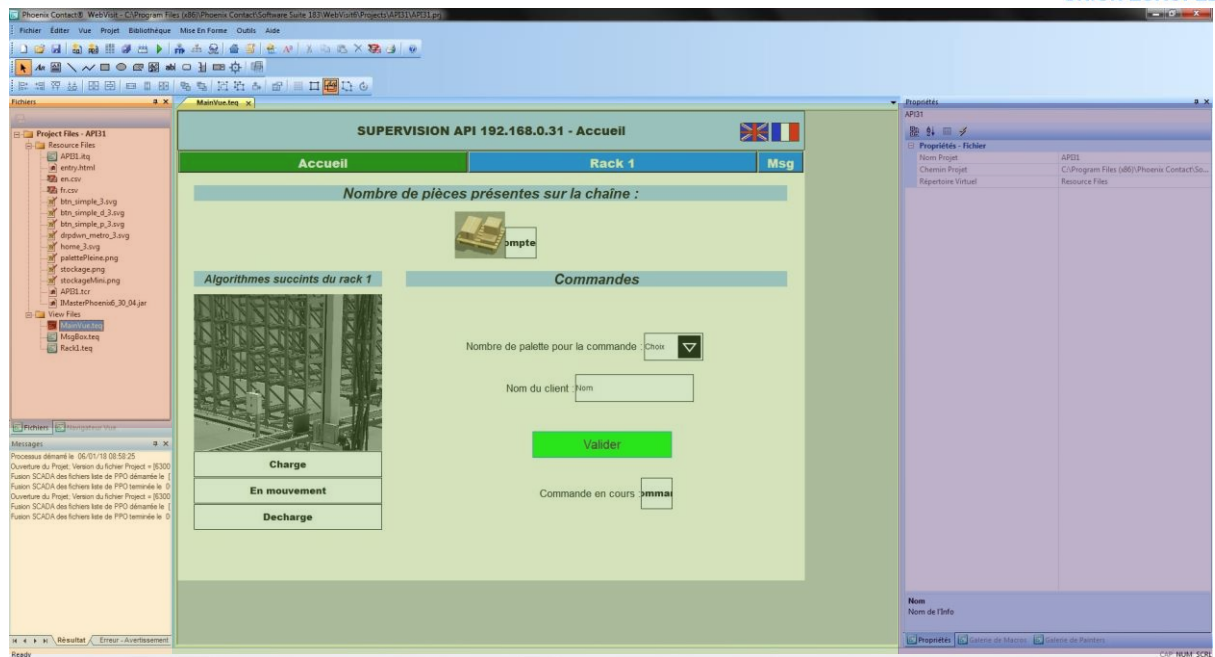


Figure 6 : Page principale de WebVisit

Sur la page principale de *WebVisit* nous avons en orange la liste de tous les fichiers qui composent le projet, en dessous les résultats de la compilation, erreurs et avertissements. Puis au centre, en vert, l'espace de travail et en violet les propriétés de l'objet que l'on a sélectionné.

4. SQL Server Management Studio 2012

Ce logiciel est une alternative au développement textuel des bases de données, il permet de rendre plus facile, plus rapide et plus efficace la mise en place de ces dernières. Il est lancé en 2005 et il est directement inclus dans l'installation d'un *SQL Server* (voir *L_10* page 40) sur une machine Windows ou Linux.

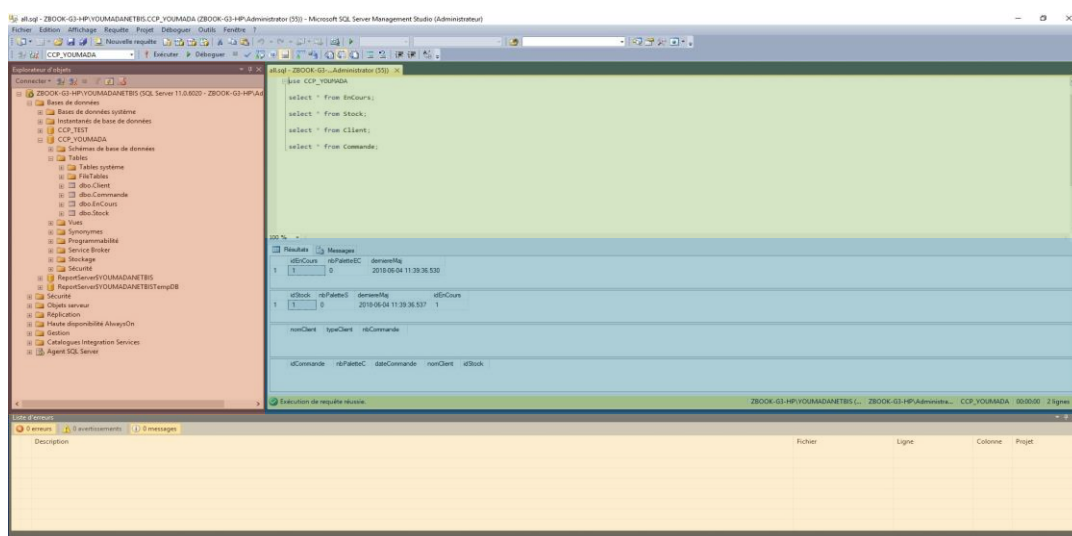


Figure 7 : Page principale de SQL Server Management Studio 2012

Cette découpe de la page principale du logiciel montre l'organisation de ce dernier. En rouge ce trouve l'explorateur d'objets avec à la racine le serveur SQL et en dessous toutes les bases de données du serveur ainsi que tout ce qu'elles contiennent. La zone pour programmer ou rentrer des commandes est celle en vert, celle en bleu affiche les résultats suivant ce que l'on a demandé dans la zone verte. Et la dernière zone, celle tout en bas, nous informe sur les erreurs que l'on a pu faire dans la zone de programmation.

5. Cogent Datahub

Cogent Datahub (voir L_5 page 40) est un moyen d'interconnecter différents réseaux, bases de données, tableurs etc. en faisant attention aux différents formats de ces derniers. C'est de cette fonctionnalité dont nous avons besoin pour faire le lien entre notre base de données et les variables OPC du système. Il est accessible en version démo en renseignant son mail sur le site de Cogent.

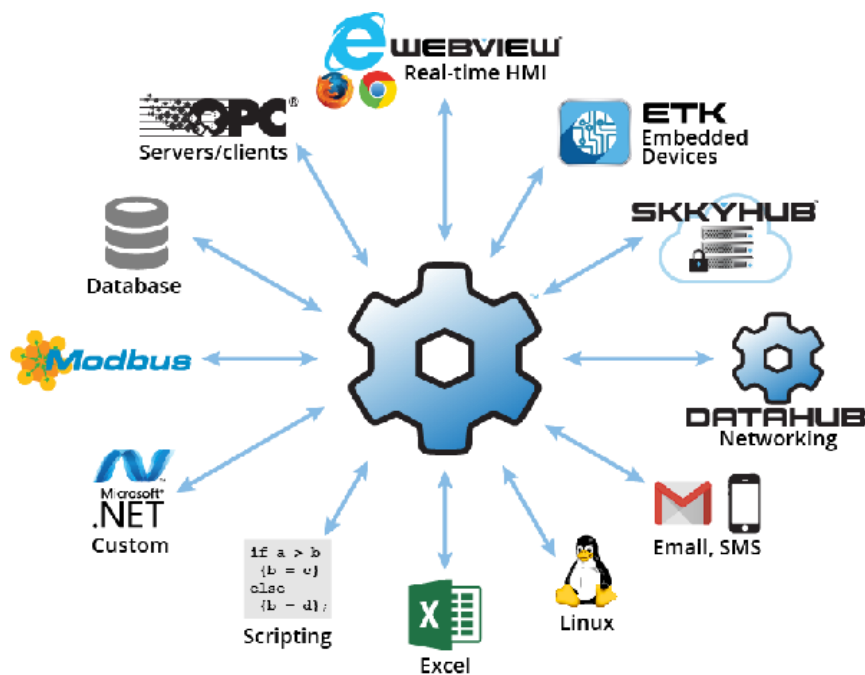


Figure 8 : Cogent DataHub

II. Analyse et conception

A. Interconnexion de tous les acteurs

Pour le bon fonctionnement du projet, il était évident que tous les acteurs de ce dernier puissent communiquer ensemble mais aussi se comprendre.

Le protocole réseau dont nous nous sommes servis pour relier tous les intervenants du projet est « PROFINET ». Ce dernier est un standard de communication dans le monde de l'industrie.

Pour que tous puissent communiquer ensemble il a aussi fallu créer un réseau local et utiliser, pour chacun, des adresses IP d'un sous-réseau local de classe C.

Et finalement un Switch Phoenix Contact était requis pour que nous puissions brancher tous nos appareils ensemble.

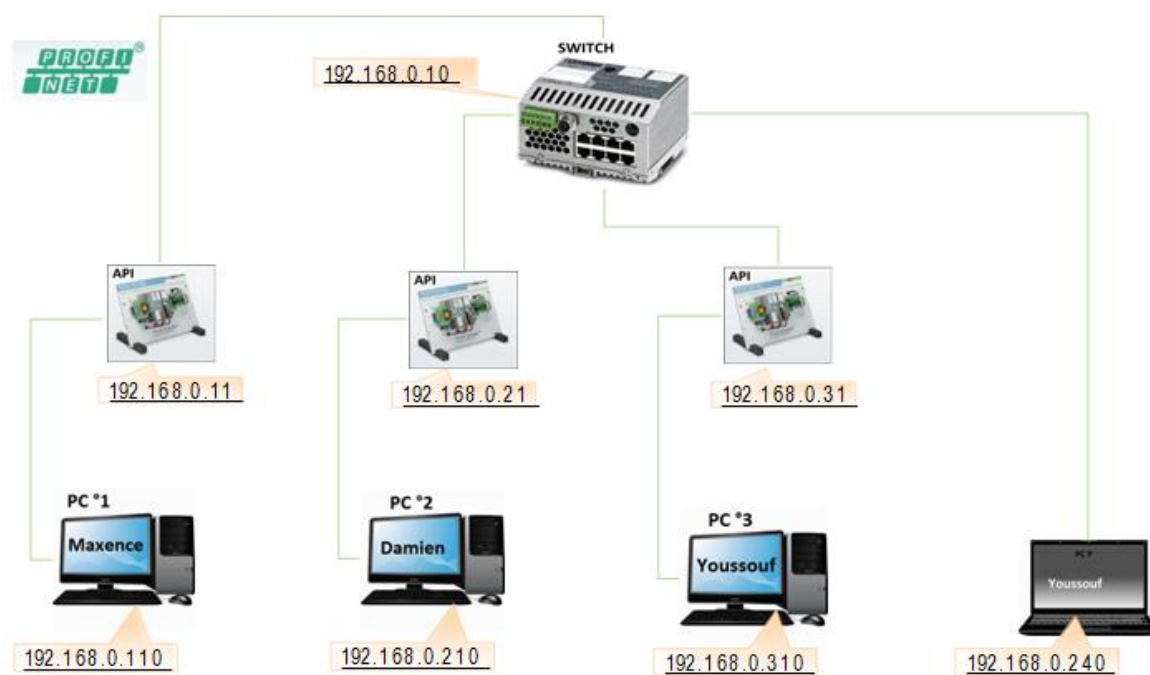


Figure 9 : Placement du réseau local

B. Stockage des variables

Le projet complet repose sur le partage des variables entre tous les acteurs présents. Pour ce faire nous avons utilisé des serveurs OPC qui permettent de stocker un nombre infini de variable pour ensuite les consulter ou même les modifier directement depuis ces serveurs. Ils sont au nombre de trois; soit un par automate, l'automate étant l'hébergeur et le créateur d'un serveur.

En effet du moment que l'on déclare des variables, avec l'option « OPC » depuis le logiciel PC WORX et que l'on compile puis télécharge le programme dans l'automate, ce dernier crée automatiquement un serveur OPC contenant toutes les variables cochées en OPC.

Nous passons ensuite par le logiciel OPC Configurator dans lequel nous créons des instances pour chaque automate en renseignant leur adresse IP. Cette action va alors mettre en place un seul serveur qui sera le regroupement des trois serveurs auxiliaires. Lors de la mise en relation entre les variables OPC et celles de Factory I/O nous n'avons alors qu'un seul serveur à rechercher pour afficher les données OPC et non trois comme le voulait la logique des trois automates.

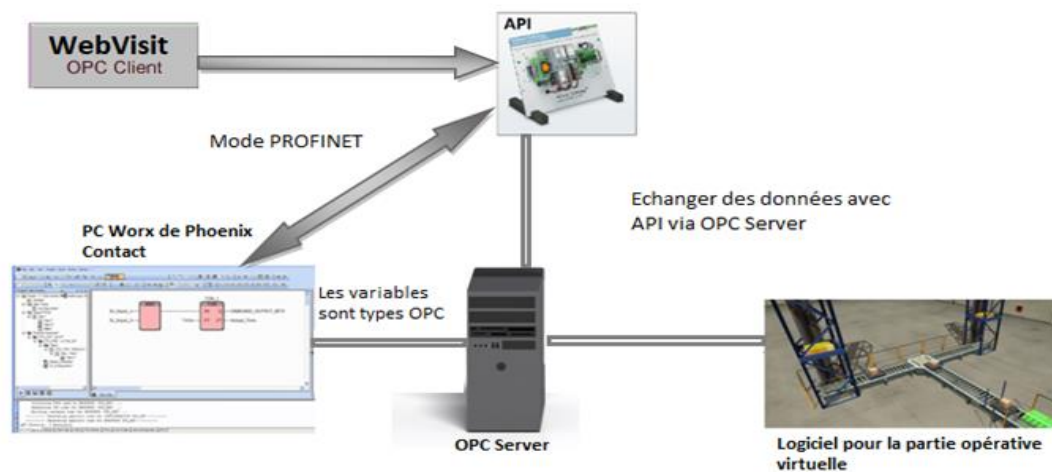


Figure 10 : Utilité OPC Server dans notre projet

C. Diagramme de Gantt

En tant que chef de projet, il était à ma charge de suivre de près le projet et de maintenir son avancement. Le projet étant multidisciplinaire, mais avec la formation en automatisme que j'ai eu au début du stage, cela m'a permis d'avoir deux casquettes (informaticien et automaticien) à la fois. Ainsi, j'ai réparti les tâches avec leurs ressources :

Nom	Rôle par défaut
Youssef IBRAHIM ELMI	Chef de projet
Maxence HAMEL	Non défini
Damien TASSEL	Non défini

Nom	Date de début	Date de fin	Ressources
Preparer le projet	12/03/18	16/03/18	Youssef IBRAHIM ELMI
Formation	19/03/18	30/03/18	Youssef IBRAHIM ELMI
Installation des logiciels	03/04/18	09/04/18	Youssef IBRAHIM ELMI, Maxence HAMEL, Damien TASSEL
Mise en place d'un réseau local	10/04/18	12/04/18	Youssef IBRAHIM ELMI, Maxence HAMEL, Damien TASSEL
Utilisation d'un switch de phoenix contact	13/04/18	13/04/18	Youssef IBRAHIM ELMI, Maxence HAMEL
Liaison entre PC Worx et API via un switch	16/04/18	16/04/18	Youssef IBRAHIM ELMI, Maxence HAMEL
Interagir Factory I/O avec (PC Worx et API)	17/04/18	23/04/18	Youssef IBRAHIM ELMI, Maxence HAMEL, Damien TASSEL
Utilisation Panorama	24/04/18	02/05/18	Maxence HAMEL
créer la chaîne industrielle	24/04/18	02/05/18	Youssef IBRAHIM ELMI, Maxence HAMEL, Damien TASSEL
Découpage de la scène en trois processus	03/05/18	03/05/18	Youssef IBRAHIM ELMI, Maxence HAMEL, Damien TASSEL
Programmer le premier automate	04/05/18	16/05/18	Damien TASSEL
Créer et tester la supervision pour l'automate n°1	04/05/18	16/05/18	Maxence HAMEL
Mode d'emploi	24/04/18	18/05/18	Youssef IBRAHIM ELMI, Maxence HAMEL, Damien TASSEL
Programmer le deuxième automate	17/05/18	17/05/18	Maxence HAMEL, Damien TASSEL
Programmer le troisième automate	18/05/18	25/05/18	Maxence HAMEL, Damien TASSEL
Construire les deux dernières supervisions	28/05/18	01/06/18	Maxence HAMEL
Faire l'arrêt d'urgence	17/05/18	17/05/18	Youssef IBRAHIM ELMI, Maxence HAMEL, Damien TASSEL
Ajouter la Fin de journée depuis la supervision	18/05/18	25/05/18	Maxence HAMEL
Base de données	04/06/18	11/06/18	Youssef IBRAHIM ELMI, Maxence HAMEL

Figure 11 : fiche de ressource

Ensuite, le diagramme de Gantt de notre projet est comme suit :

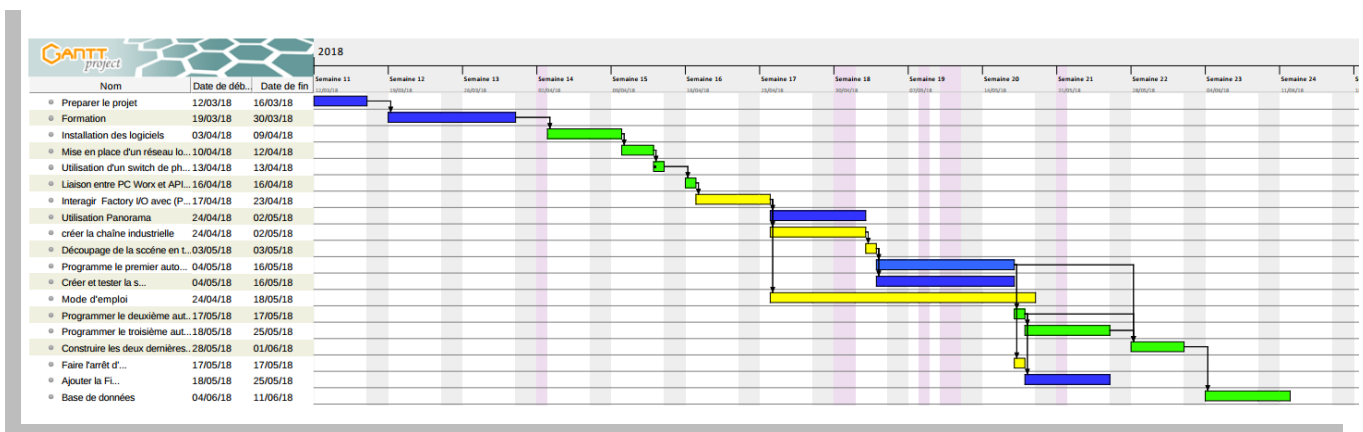


Figure 11 : Diagramme de Gantt

Sur ce diagramme de Gantt, nous avons :

- En bleu les tâches réalisées par un stagiaire,
- En vert les tâches réalisées par deux stagiaires,
- En jaune les tâches réalisées en commun.

D. Les étapes de la réalisation

La réalisation s'est donc faite en plusieurs étapes détaillées ci-dessous :

Figure 12 : Tableau des étapes du projet

<i>Étape</i>	<i>Objectif à valider</i>
1	Faire la liaison entre PC WORX et un seul automate
2	Concevoir une scène très simpliste sur Factory I/O
3	Interagir sur la scène de l'étape 2 avec un interrupteur présent sur l'automate de l'étape 1
4	Créer la chaîne industrielle finale
5	Programmer le premier automate
6	Créer et tester la supervision pour l'automate numéro un
7	Insérer les deux derniers automates en les programmant un par un
8	Construire les deux dernières supervisions
9	Tester l'ensemble de toutes nos productions
10	Ajouter un bouton d'arrêt d'urgence dans la scène de l'étape 2
11	Faire le programme de l'étape 10
12	Ajouter à la supervision la fin de journée

Ici, je n'ai mentionné que les grandes étapes,

III. Réalisation

A. Cahier des charges

Même s'il ne m'était donné que trois missions, j'ai quand même participé à la programmation des automates et à la supervision. Ce cahier des charges présente toutes les étapes du projet.

1. Les intervenants humains ou non

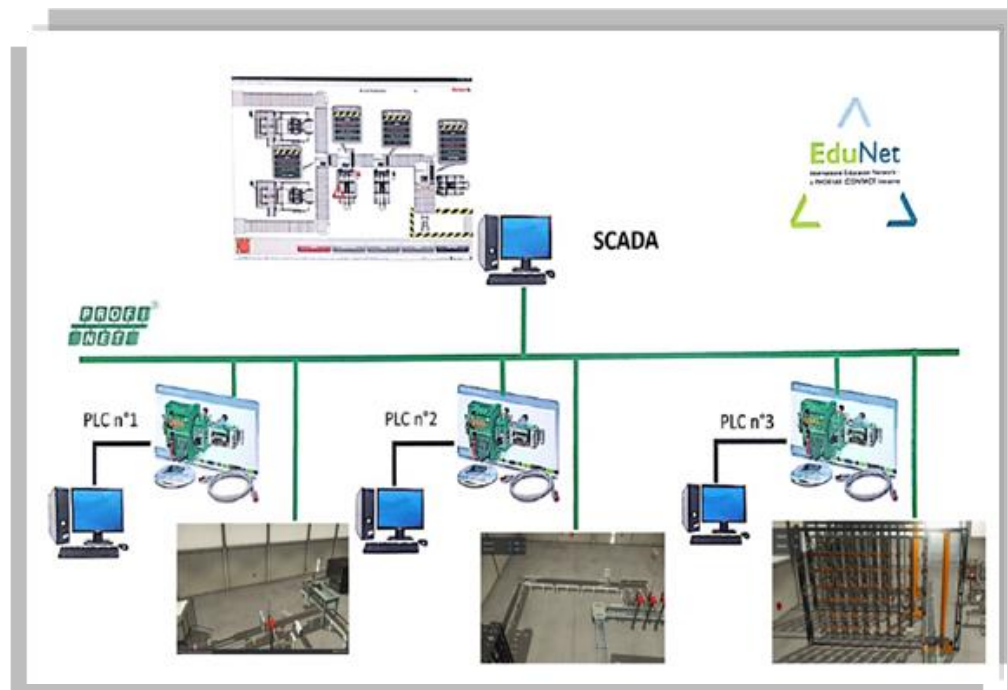
Les intervenants de ce projet sont au nombre de sept. Les voici :

- ✓ 1 PC Portable : affiche l'environnement virtuel de simulation
- ✓ 3 API (*L_2 page 40*) : dirigent toute l'installation
- ✓ 1 Switch (*L_12 page 40*) : permet la communication entre tous les acteurs
- ✓ Client : passe des commandes à travers la supervision
- ✓ Navigateur web : permet de visualiser les trois supervisions

2. Énoncé des fonctions de service

Fonction principale :

Piloter un environnement industriels virtuel grâce à des machines réelles, le tout contrôlé par un poste de supervision et une base de données pour sauvegarder les informations importantes.



Les différentes fonctionnalités attendues :

Aspect Conception :

- C1 : Utiliser le maximum de ressources dans Factory I/O
- C2 : Mettre en place un scénario cohérent et réaliste
- C3 : Prévoir l'utilisation de trois automates au contrôle de la simulation

Aspect Paramétrage :

- P1 : Possibilité de changer d'angle de vue dans l'environnement virtuel
- P2 : Le client peut passer une commande depuis la supervision
- P3 : Le mode « Fin de journée » peut être activé ou non avec la supervision

Aspect Documentation :

- D1 : Documentation Factory I/O
- D2 : Documentation PCWORX
- D3 : Documentation WebVisit

3. Mes missions

Après avoir défini le cahier des charges, faisons un point sur mes missions principales et sous-missions.

Missions :

- Chef de projet (Analyse et conception)
- La mise en place de l'environnement virtuel
- Mettre en place deux bases de données

Mission auxiliaire :

- Participer à la programmation des automates
- Participer à la création des supervisions

Sous-mission :

- Créer un poster de notre projet pour les Journées Innovation Pédagogique Normande (JIPN) qui se déroulent au PIL le 2-3 juillet 2018

B. Mise en place de l'environnement virtuel

1. Le Scénario

L'interface du logiciel Factory I/O est assez simple et dispose de trois menus et plusieurs boutons permettant de gérer les caméras, d'afficher la palette, d'afficher les capteurs, d'afficher les actionneurs, d'effectuer une simulation etc... Ici je n'expliquerai pas le fonctionnement du logiciel puisqu'on a effectué un manuel d'utilisation de ce logiciel. Mais il faut savoir que toutes les composants permettent de concevoir brique par brique notre système automatisé, par exemple des palettes, des tapis roulants, des boutons poussoirs, des voyants etc. se trouve dans le menu suivante :



Vous pouvez concevoir plusieurs types de systèmes en faisant un simple glisser-déposer du composant dans l'interface d'édition.

Il s'agit de concevoir un processus industriel et logistique virtuel entièrement automatisé. Cette Virtualisation hybride associe de vrais automates programmables industriels pilotant des parties opératives virtuelles comme suit :

- **Fraisage**
- **Assemblage**
- **Mise dans une caisse en plastique**
- **Mise en carton**
- **Tri des cartons**
- **Mise sur palette**
- **Racks de Stockage**

Figure 13 : Tableau des fonctions de chaque processus du scénario virtuel

Nom du processus	Fonction
Fraisage	Transformer une plaque en un socle capable d'accueillir un couvercle
Assemblage	Assembler un socle avec un couvercle pour former un module
Mise en caisse	Positionner cinq modules dans une caisse vide puis y rajouter un kit
Mise en carton	Transformer une caisse pleine en trois cartons de taille respective S, M, et L
Tri des cartons	Orienter les trois cartons dans des couloirs en fonction de leur taille
Mise sur palette	Placer les différents cartons sur une palette à des positions prédéfinies
Rack de stockage	Stocker les palettes dans un rack de cinquante quatre places

Les matériaux utilisés ne pouvant pas tous être fabriqués par les machines de Factory I/O nous avons été dans l'obligation de les faire apparaître grâce à des émetteurs configurables. Ces derniers sont aussi la solution au problème que posait la mise en carton car le logiciel ne propose pas de solution pour mettre des pièces dans des cartons, il a donc fallu détruire les caisses pleines pour faire apparaître trois cartons.

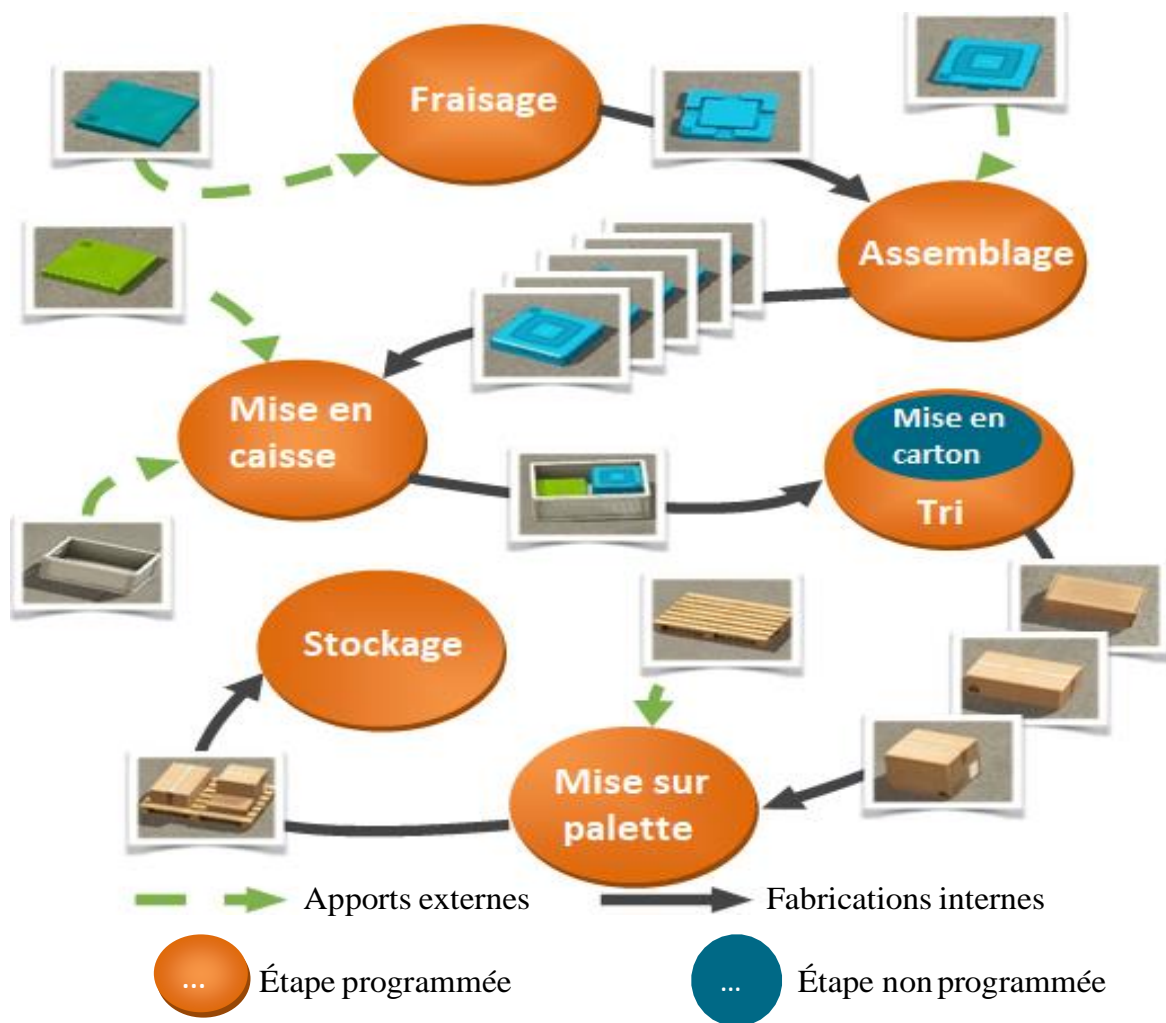


Figure 14 : Scénario de l'environnement Factory I/O

2. La répartition entre les trois automates

Une fois qu'on a fait l'ensemble de l'entrepôt virtuel grâce au logiciel *Factory I/O*, on dispose de trois automates programmables industrielles (API) de Phoenix Contact qui ont les trois adresses IP comme suit :

- ✓ Premier API : adresse IP : 192.168.0.11
- ✓ Deuxième API : adresse IP : 192.168.0.21
- ✓ Troisième API : adresse IP : 192.168.0.31

Dans chaque automate, on aura le programme d'une partie de notre scène virtuelle :

- ✓ API 1 : contiendra le programme de fraisage, assemblage et mise en caisse.
- ✓ API 2 : contiendra le programme de mise en carton, tri et la mise sur palette.
- ✓ API 3 : contiendra le programme de stockage.

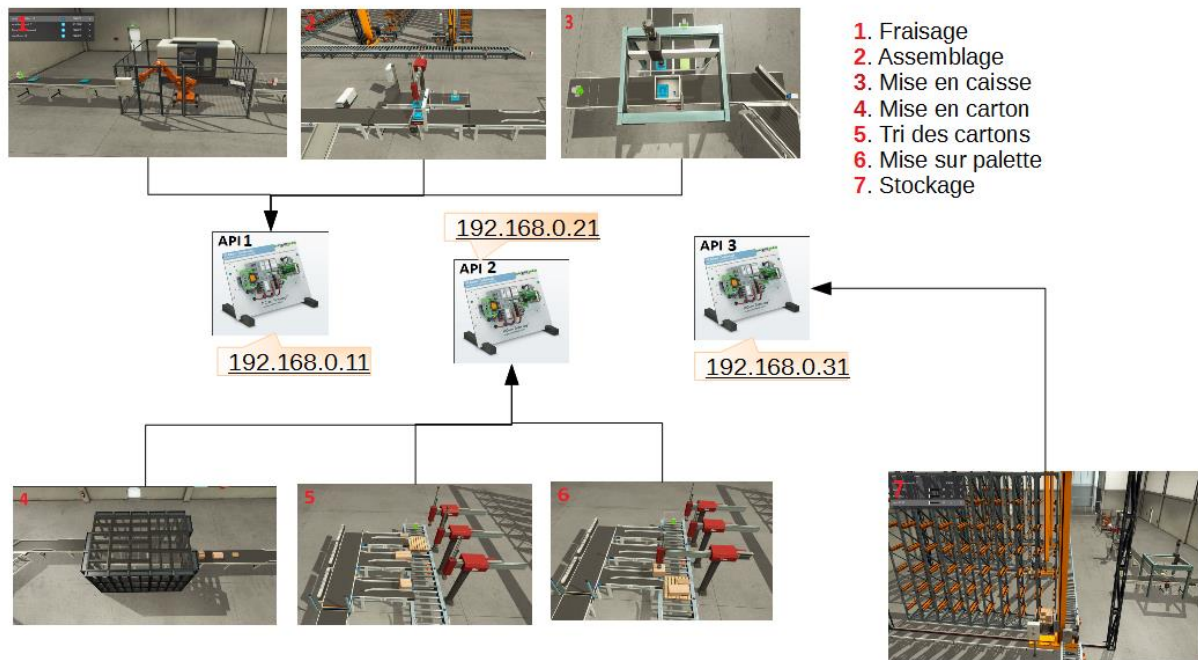


Figure 15 : Répartition de la scène entre les 3 automates.

C. Programmation de la scène

1. Les langages utilisés

1.1. Sequential Function Chart

Une fois que la scène sur *Factory I/O* est faite. La suite a été de programmer toutes les machines qui avaient été mises dans la scène afin de pouvoir tout automatiser. Pour ce qui est de la programmation plusieurs langages ont été utilisés. Le premier étant le **SFC** (ou **Grafcet** (*L_6 page 40*)) qui est un langage qui se lit de haut en bas, voir *Figure 16*, avec des étapes (S017), des transitions (T007) et des actions (descendreGrueP2_11), ci-dessous un exemple de ce langage de programmation séquentiel :

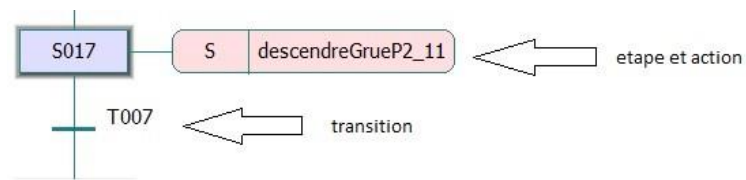


Figure 16 : Organisation usuelle d'une étape en langage SFC

La particularité de ce langage de programmation est sa facilité pour comprendre et programmer, c'est un langage graphique. Il est souvent utilisé pour de la programmation d'automate, il obéit à la norme « *CEI 61131* » qui est une norme industrielle de la Commission Électrotechnique Internationale (CEI).

1.2. Ladder

Le deuxième langage qui a été choisi est le Ladder qui se lit de gauche à droite. C'est aussi un langage graphique composé d'entrées et de sorties avec des blocs fonctionnels permettant de gérer les variables comme les multiplier, les diviser ou encore les additionner entre elles. Ces blocs peuvent aussi créer une temporisation, des compteurs ou bien des opérations logiques.

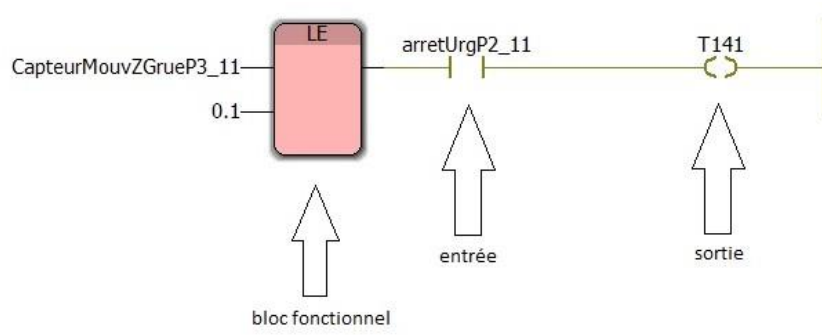


Figure 17 : Représentation d'une transition codée en Ladder

Dans ce cas particulier, la Figure 17 représente l'intérieur d'une transition d'un programme SFC. Le **bloc fonctionnel** « LE » compare si la variable « CapteurMouvZP3_11 » est inférieure à 0,1. L'entrée « arretUrgP2_11 » doit être à 1 pour valider la sortie T141 qui est la transition pour passer à l'étape suivante. Le Ladder ressemble beaucoup à un schéma électrique car pour réaliser un **ET logique** il suffit de mettre les entrées en séries comme des interrupteurs sur un schéma électrique à l'inverse si l'on souhaite faire un **OU logique**, il faut les mettre en parallèle. Ce langage répond aussi à la norme « CEI 61131 ».

1.3. Free Form Ladder Diagram

Pour finir le dernier langage utilisé fut le FFLD, c'est un mélange du Ladder et du (ST) Structured Text (L_11 page 40).

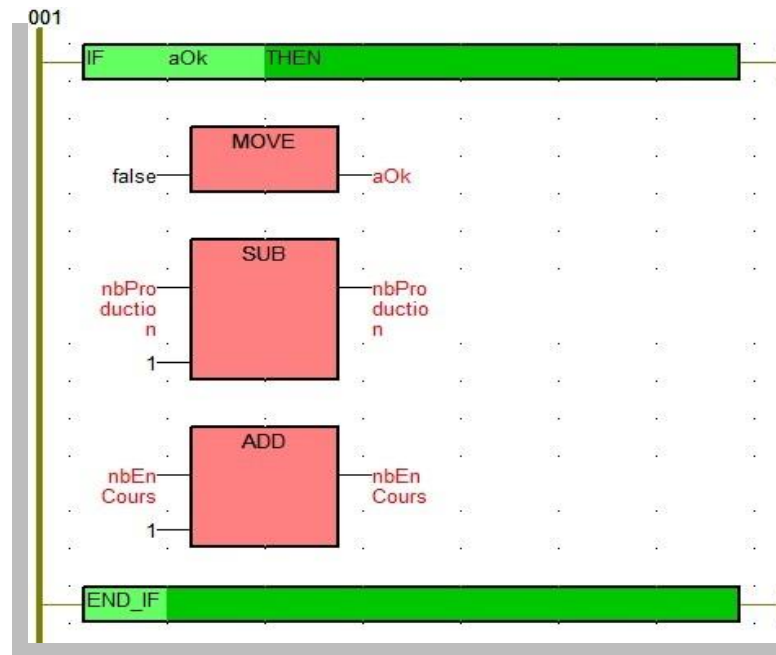


Figure 18 : Exemple de programmation en langage FFLD

Dans cette Figure 18, nous retrouvons les **blocs fonctionnels**, **MOVE** (assigne une valeur), **SUB** (soustraction), **ADD** (addition) qui viennent du *Ladder*, ainsi que le **IF** et le **END_IF** qui viennent du *Structured Text*. Ce mélange offre à ce langage plus de possibilité de programmation.

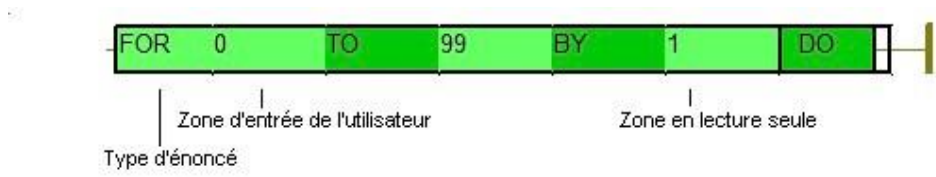


Figure 19 : Syntaxe d'une barre de contrôle de type FOR

Voici, ci-dessus avec la *Figure 19*, une barre d'énoncé de contrôle qui est composée de sept cases avec chacune un rôle bien défini. La première est le type d'énoncé c'est là que l'on va choisir si l'on veut un **FOR**, **IF**, **WHILE**, **REPEAT**, etc. La deuxième est la zone d'entrée de l'utilisateur, elle a pour rôle, dans un FOR, de définir la première valeur de la boucle. La quatrième case indique la valeur finale de la boucle, enfin la zone en lecture seule indique le pas d'incrément.

2. Exemple de la mise en caisse

2.1. Cahier des charges

La mise en caisse consiste, juste après l'assemblage des modules, à empiler trois modules à une première position et deux modules plus un kit à la seconde position. L'ensemble sera dans une caisse représentant un carton de chaque taille.

2.2. Structure du programme

Comme nous pouvons le voir avec la Figure 20, nous avons trois branches; la première va servir à récupérer un module bleu assemblé auparavant et qui va le mettre dans la caisse à une première position; la deuxième branche sert aussi à récupérer un module bleu, mais cette fois-ci il va être placé à une deuxième position dans la caisse. Pour ce qui est de la troisième branche, elle va prendre un kit d'assemblage (module vert) et va la déposer par dessus les modules de la deuxième position.

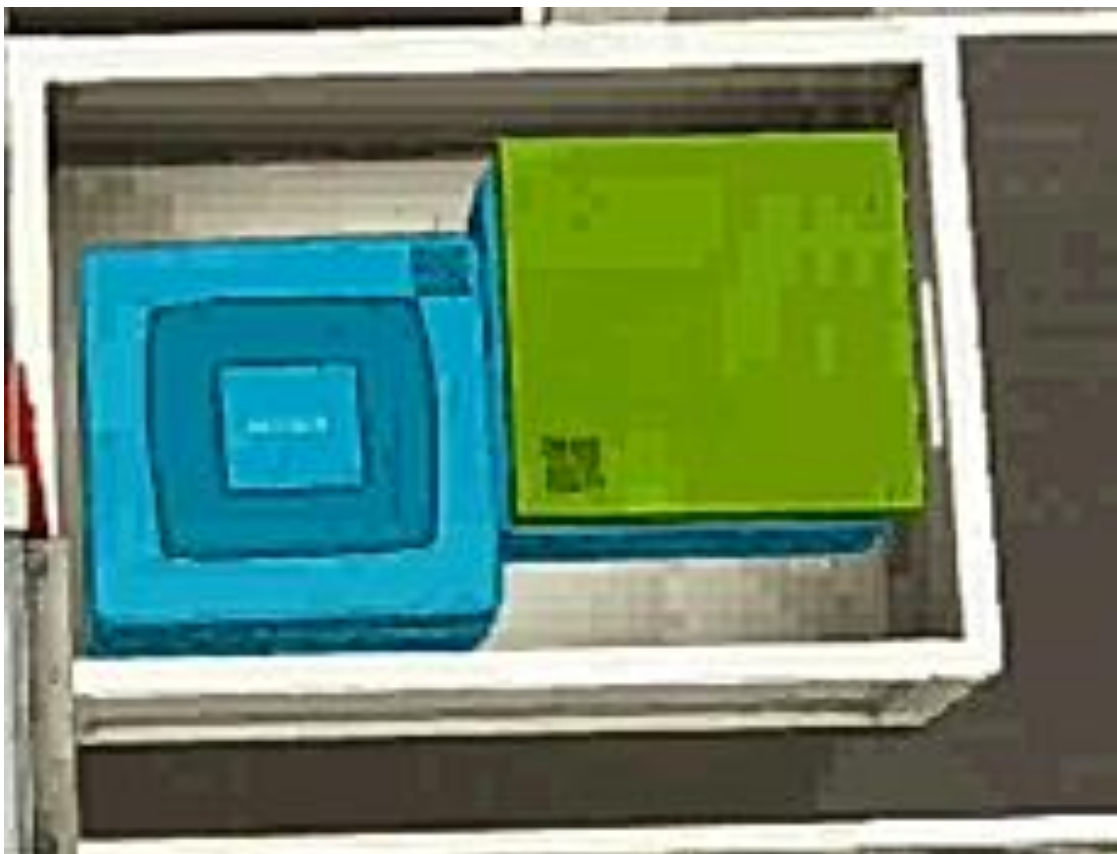
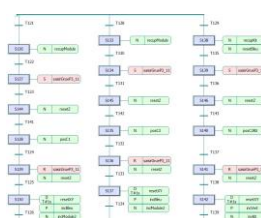


Figure 20 : Programme principal pour la mise en caisse

Comme ceci :



2.3. Explication détaillée

Comme on peut le voir chaque action a des lettres différentes, ce sont des qualificatifs qui sont propres au *GRAFCE*T:

N	L'action restera active tant que la transition qui suit ne sera pas validée, exemple dans la première branche « <i>resetZ</i> » est à 1 tant que « <i>T141</i> » ne sera pas validé.
S	L'action est mise à 1 sur toutes les étapes qui suivent et le restera si on ne l'a remet pas à 0 .
R	L'action est mise à 0 le restera jusqu'à ce qu'on l'active de nouveau
P	Donne une impulsion à l'action
D	Impose une temporisation à l'action, c'est à dire que l'action va s'activer après que le temps soit écoulé de seconde à attendre avant le déclenchement

On peut aussi remarquer que les actions comportent différentes couleurs; **les actions rouges** sont des actions définies dans la table des variables et **les actions vertes** sont des actions créées par l'utilisateur. Les actions vertes pourront faire plusieurs choses en même temps alors que les actions rouges ne font qu'une seule chose à la fois.

Pour faire bouger la *grue* à la position du module, il suffit de rentrer des valeurs dans les trois axes de la *grue*.

On double clique sur « *recupModule* » dans la première branche, on obtient la Figure 13

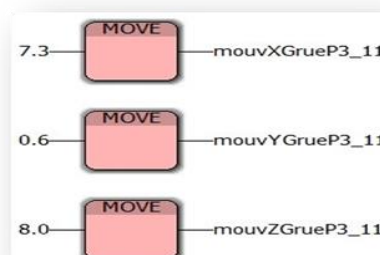


Figure 21 : Positionnement de la grue à une certaine position

Les trois « *MOVE* » servent à faire bouger la grue à une position en X, Y et Z. La grue a un intervalle qui va de zéro à dix pour chaque axe. Ici on lui donne la position de (7.3, 0.6, 8) sur l'axe respectives (X, Y, Z).

Comme vu précédemment il doit y avoir cinq modules dans la caisse donc pour cela, un compteur va compter de zéro à cinq pour ne pas dépasser le nombre maximum et c'est l'action « *incBleu* » qui va se charger d'incrémenter le compteur de module.

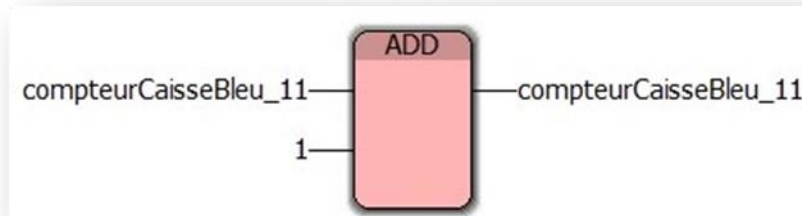


Figure 22 : Incrémentation du compteur de module dans la caisse

Avec la figure 22, on voit l'intérieur de l'action « *incBleu* ». Le bloc ADD va additionner le compteur de module à la fin des deux premières branches, à chaque fois que la grue retourne à sa position initiale. Une fois le compteur arrivé à cinq, le programme va rentrer dans la transition « *T129* ».

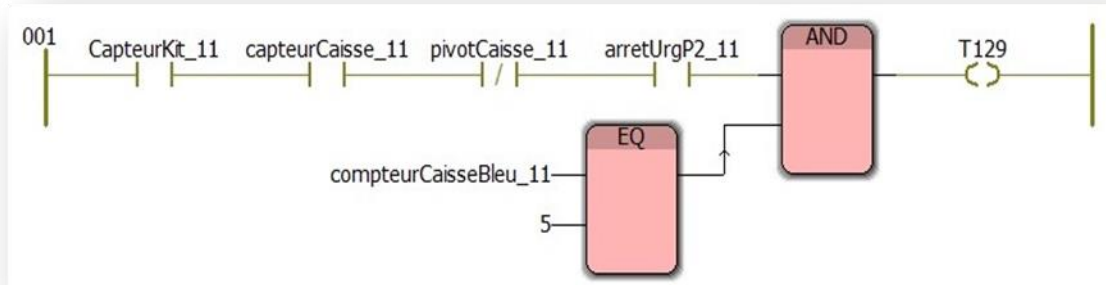


Figure 23 : Transition T129

Le bloc « *EQ* », ci-dessus, signifie « *EQUAL* » cela veut donc dire que si la variable « *compteurCaisseBleu_11* » est égale à cinq alors le programme va rentrer dans la troisième branche qui va aller chercher un kit puis le déposer dans la caisse pour ensuite évacuer la caisse et remettre à zéro le compteur de module présent dans la caisse.

Pour pouvoir placer les modules dans la caisse c'est l'action « *posC1* » qui va définir la première position et le placement sur l'autre position sera l'action « *posC2* ».

Voilà donc pour ce qui est de la mise en caisse des modules.

D. Supervision

1. Les attentes d'une supervision

Une supervision est tout d'abord un moyen de surveillance remplaçant l'humain qui pourrait scruter une chaîne industrielle à la recherche d'erreurs. C'est donc pour cela que l'on attend de cette dernière une retranscription exacte de tout ce qui se trouve sur l'installation. Sa première fonction est alors d'informer en temps réel le superviseur.

Cependant, il est aussi d'usage qu'elle possède une deuxième fonction, celle de permettre l'interaction entre le superviseur et la chaîne. Par exemple, nous pourrions très bien vouloir stopper l'installation non pas en allant appuyer sur le bouton d'arrêt d'urgence qui se trouve, dans la majorité des cas, dans une pièce différente de celle où se trouve le superviseur, mais en cliquant simplement sur un bouton présent sur la supervision. Cette fonction qui prend la main sur le système permet une meilleure réactivité dans des cas d'urgences ou de dysfonctionnements.

La supervision est donc un moyen d'informer le superviseur qui n'est pas forcément proche de l'installation, mais aussi de permettre à ce dernier d'agir sur cette dernière.

Après avoir rappelé les fonctions primaires d'une supervision passons maintenant au choix du design qui en découle directement.

2. Le choix du design

Dans l'optique de respecter les deux fonctions principales annoncées ci-dessus et de les harmoniser avec notre scénario nous avons opté pour une supervision en deux sections distinctes.

Sur ce schéma simplifié, Figure 24 ci-dessous, nous voyons l'architecture type de la page d'accueil des trois supervisions. En premier, nous avons le titre sous la forme « Supervision API 192.168.0.XX - XX » pour avertir l'utilisateur sur quel automate et sur quel onglet il se trouve. Ensuite, une option pour la langue, pour l'instant française ou anglaise. Puis la liste des onglets tels que celle qui est présente sur un moteur de recherche basique. Pour finir, la partie retranscription avec ou non une possibilité d'interaction avec l'installation.

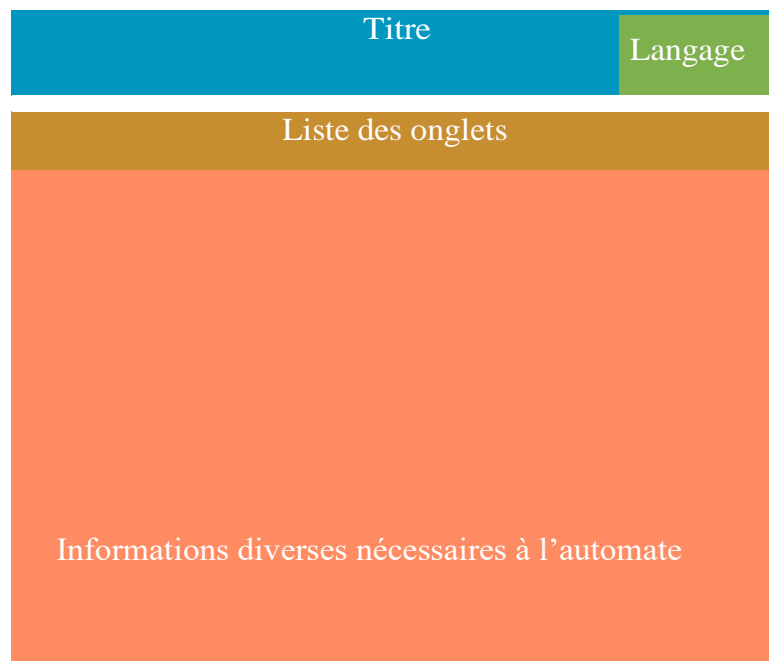


Figure 24 : Schéma simplifié d'une page de la supervision

Voilà donc pour le design générique de la supervision, maintenant nous passons au cas par cas même si de nombreuses ressemblances existent entre les trois supervisions. Nous commencerons par ces similitudes, la première étant l'écran d'accueil qui possède deux parties, sans compter le titre, la langue et les onglets qui eux restent communs à toutes les pages.

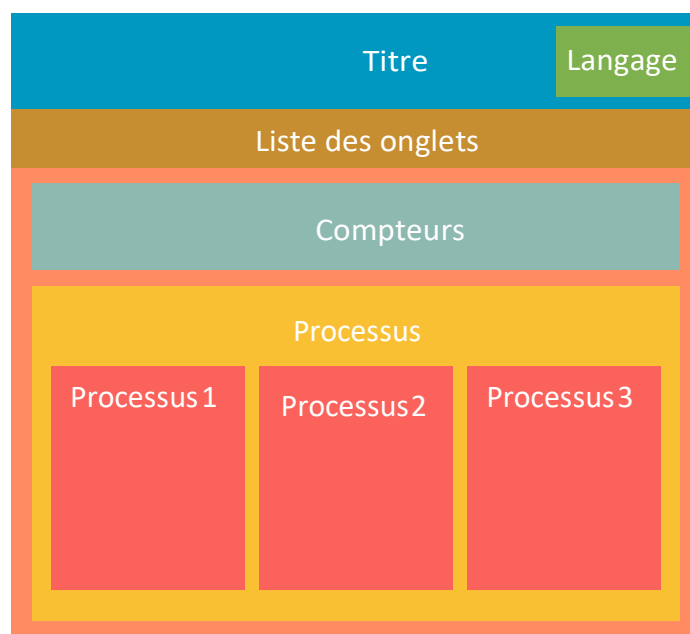


Figure 25 : Schéma structurel d'une page d'accueil

Cette Figure 25 montre l'arrangement de l'accueil sur les automates n°1 et n°2. La section des compteurs, comme son nom l'indique, affiche les compteurs individuels de tous les matériaux; socles, couvercles, caisses, etc. qui ont été traités par l'automate. Ces derniers s'incrémentent en temps réel. Juste en-dessous des compteurs se trouve la section des processus.



Figure 26 : Page d'accueil de l'automate n°1

En effet, chaque automate possède un ou plusieurs processus, par exemple l'automate n°1 (figure 26) a trois processus; le fraisage, l'assemblage et la mise en carton. Ceux sont donc ces processus qui sont affichés avec une photo « noir et blanc » pour éviter de surcharger l'automate, et un algorithme ultra simplifié pour savoir où ils en sont dans leur traitement. Il n'y a que l'automate gérant le stockage qui ne possède qu'un seul processus car il ne contrôle qu'un seul rack.

Le deuxième type de fenêtre, quand à elle, affiche processus par processus les compteurs qui leurs correspondent et toutes leurs variables.

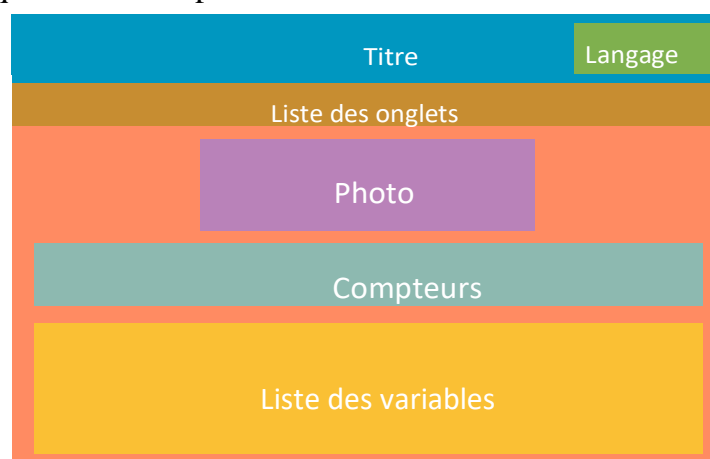


Figure 27 : Schéma structurel d'une page secondaire d'un processus x

Cette représentation de la page d'un processus, voir Figure 27 ci-dessus, est identique pour les six partis opératives virtuelles (bras robotisés, assemblages, mise en caisse, mise en cartons, tri et mise sur palette) présents sur nos onglets de supervision. Cet onglet est composé d'une photo pour qu'au premier coup d'œil nous puissions savoir de quel processus il s'agit. Sous cette photo, sont affichés les compteurs qui eux sont propres aux processus. Et pour finir, un tableau montrant l'état, la valeur des capteurs et actionneurs du processus.

Le dernier onglet commun à tous les automates est celui des messages d'erreurs. Il est simplement composé d'un écran affichant la source des erreurs et leurs raisons ainsi qu'un bouton « OK » pour retourner à la supervision (cf. A_2 page 42).

Après avoir désigné et créé les trois supervisions, il a fallu rendre actives ces dernières qui pour l'instant n'avaient aucune relation avec les automates.

3. Liaison avec les automates

Pour faire cette liaison, ou plutôt ces trois liaisons, il fallait attendre la fin de la programmation des automates. WebVisit allant chercher un fichier sous format tableur contenant toutes les variables du projet, nous devons impérativement attendre que toutes les variables soient déclarées. Il ne restait plus qu'à relier chaque variable à son lieu d'affichage, mais aussi à s'en servir pour animer les algorithmes de la page d'accueil et les compteurs.

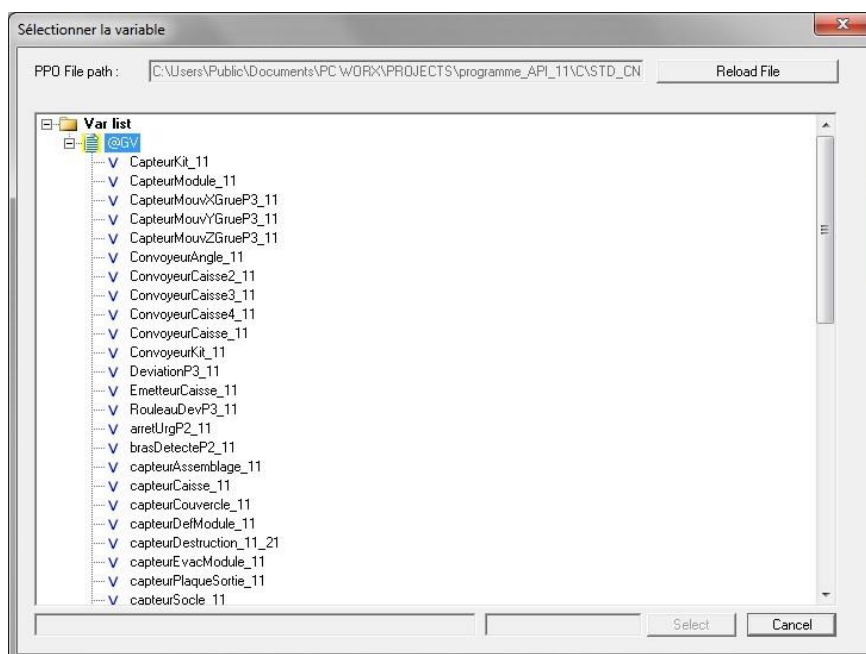


Figure 28 : Liaison des variables dans les processus

Par la suite, pour télécharger la supervision dans l'automate il suffisait que ce dernier et le PC de développement soit sur le même réseau. Puis en rentrant l'adresse IP de l'automate dans WebVisit le transfert de la supervision pouvait se faire.

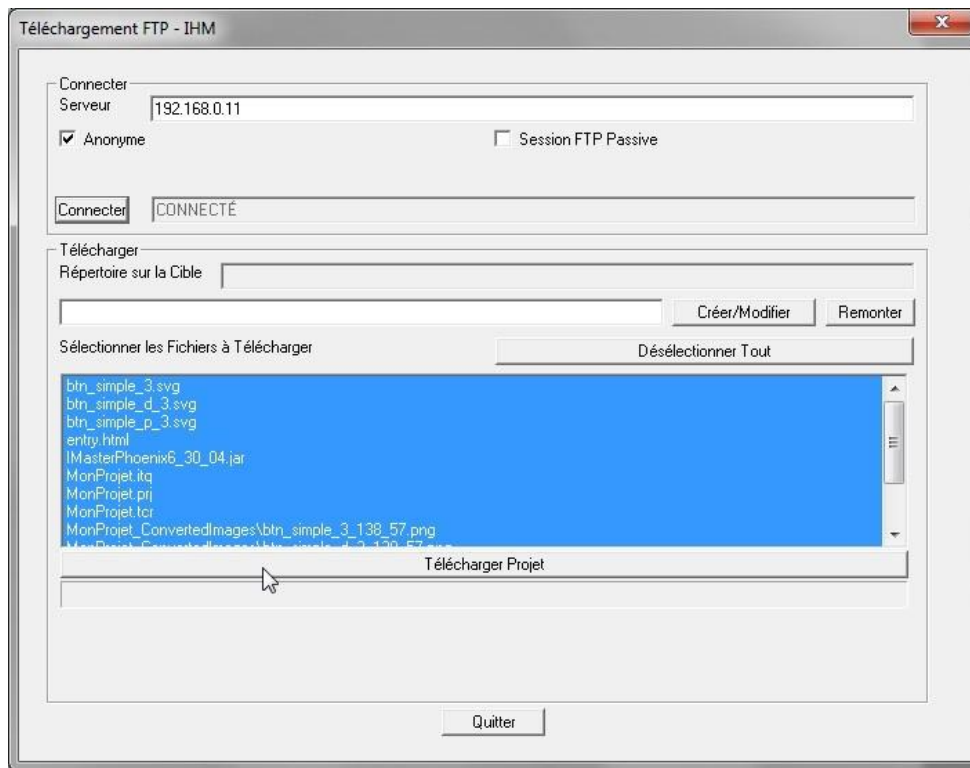


Figure 29 : Télécharger la supervision dans l'automate

De même pour consulter la supervision il fallait un PC connecté aux automates, puis ouvrir Internet Explorer, uniquement, et taper l'adresse IP dans la barre d'URL puis valider.

E. Les bases de données

1. Leurs fonctions

De part l'organisation du projet, il est nécessaire de développer deux bases de données (où BADO (voir L_3 page 40)) pour ne pas mélanger ce qui ne devait pas l'être. En effet, l'utilisateur ayant la possibilité également de passer des commandes par le biais de la supervision nous devons alors sauvegarder toutes ses commandes, mais aussi sauvegarder ses coordonnées en tant que client. Nous avons donc déjà une première base de données, clients et commandes. La deuxième comprendrait des données intrinsèque à notre simulation avec des relevés sur l'efficacité, le temps de fonctionnement ou encore la consommation de notre installation.



Figure 30 : Représentation fonctionnelle des deux bases de données

2. L'organisation des deux bases de données

Pour créer une base de données il faut en tout premier lieu réfléchir sur papier à la structure de cette dernière. Pour cela, nous commençons par l'élaboration du schéma « Entité / Association ».

Base de données n°1, Client Commande Palette (CCP) :

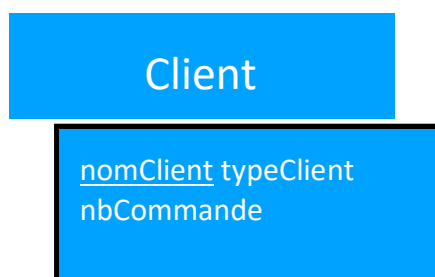


Figure 31 : Schéma entité association d'un Client

Cette représentation graphique, Figure 31, d'un client montre tout ce qu'il faut savoir sur lui. Son identifiant, ici « *nomClient* », veut tout simplement dire qu'un seul client pourra avoir ce nom (dans les cas réels ce n'est pas le cas, mais dans le notre nous ne pouvons pas faire autrement). Il possède aussi un type prenant une valeur entre « *Nouveau* », « *Occasionnel* » ou « *Régulier* » selon le nombre de commandes qu'il passe. Et pour finir, le total des commandes passées par ce client.

Nous passons ensuite en revue toutes les autres entités et faisons le même travail de réflexion qu'avec le client ci-dessus. Tout ça en répondant à la question :

Qu'est-il important de sauvegarder dans ce processus de client et de commande ?

Après avoir créé toutes les entités nécessaires, il est temps de les faire interagir ensemble. En effet dans toutes les bases de données chaque entité est liée à une ou plusieurs autre(s) entité(s). Prenons ici l'exemple d'un client et d'une commande.

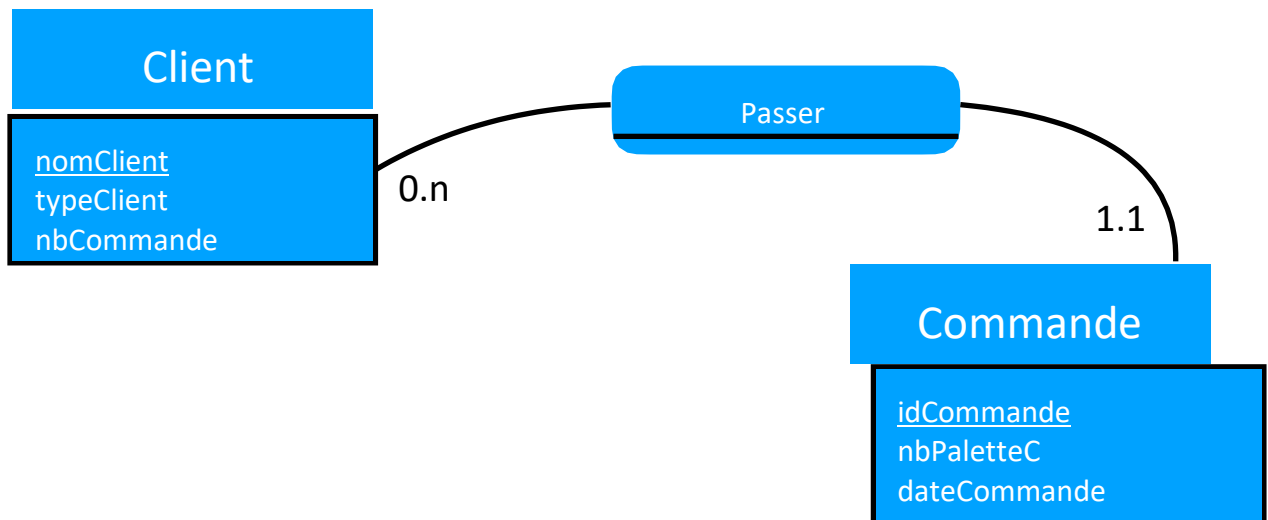


Figure 32 : Schéma entité association d'un Client et d'une Commande

Avec cette Figure32 nous avons toujours l'entité « Client » à laquelle nous avons ajouté celle d'une « Commande ». Néanmoins, nous sommes d'accord qu'un client possède forcément une relation avec une commande, d'où l'association « Passer ». Et de ce fait on peut lire deux choses : soit « Un client passe une commande » ou bien « Une commande est passée par un client ». Pour finir il faut indiquer les cardinalités, c'est à dire combien un client peut passer de commandes et à combien de clients est relié une commande. Ici nous pouvons alors dire, grâce aux cardinalités « (0.n) » et « (1.1) », un client passe aucune ou une infinité de commandes et une commande n'est passée que par un et un seul client. Voir A_3, page 43, pour le schéma entité association complet.

Le dernier stade de la réflexion pour mettre en place une base de données est de traduire ce schéma entité association en schéma logique qui bien que s'appelant schéma s'écrit sous forme textuelle comme suit.

<p>CLIENT (<u>nomClient</u>, typeClient, nbCommande)</p> <p>COMMANDE (<u>idCommande</u>, nbPaletteC, dateCommande, nomClient#)</p>

Figure 33 : Schéma logique de Client et de Commande

Cette écriture, représentée avec la Figure 33, permet de cibler la manière dont les entités vont être reliées ensemble. Certaines règles existent pour le passage entre le schéma entité association et le schéma logique. Ici l'une de ces règles dit que lorsqu'on est en présence de cardinalités « (0.n) » et « (1.1) » alors la clé d'identification du côté du « (0.n) » est transmise à l'entité possédant le « (1.1) ». D'où la présence de « *nomClient#* » dans la ligne de la commande; le dièse indiquant qu'il s'agit d'une clé étrangère. Cela veut donc dire que lors de la création de l'entité « *Commande* » un champ sera alors rajouté et s'appellera « *nomClient* ». Nous pourrions donc juste en regardant la liste de toutes les commandes savoir quel client correspond à quelle commande sans avoir à afficher la liste des clients puisque les noms seront enregistrés dans les commandes.

Après cette retranscription en schéma logique il ne restait plus qu'à créer la base de données (cf. chapitre suivant).

Base de données n°2, Bilan Traitement Analyse (BTA) :

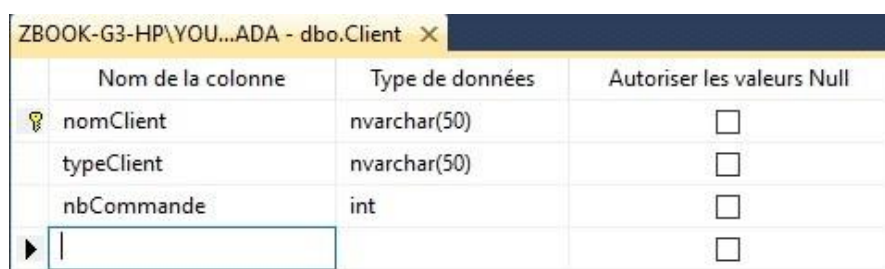
Cette deuxième base de données est plus technique que le numéro une. En effet elle permet de récupérer toutes les informations du type temps de réponse des capteurs, rendement de l'installation etc. pour ensuite afficher tout ceci sous forme de graphique ou juste tel quel. C'est donc dans une optique d'amélioration que cette base de données a été conçue. Cependant ni le logiciel Factory I/O, ni PC WORX ne pouvait fournir ces informations. Il a donc été convenu que seule la partie conception serait faite sans aller jusqu'à l'implantation de la base de données.

Voir schéma entité association A_4, page 44.

3. Création

Comme il vient d'être mentionné seule la base de données CCP sera traitée dans ce chapitre. Pour créer cette base de données **CCP**, il a été dans un premier temps nécessaire d'installer *SQL* (voir *L_9 page 40*) *Server* sur le PC Portable qui était l'hôte de toutes les variables du système. Ensuite il a fallu créer la base « *CCP_YOUMADA* » (voir *L_13 page 40*) dans laquelle nous pourrions insérer les entités dont nous aurions besoin.

Pour créer une entité, aussi appelé « *TABLE* », il suffisait de cliquer droit sur « *CCP_YOUMADA* » puis « Créer une nouvelle table ». Apparaissait alors une interface du même type qu'un tableur où nous devions renseigner le nom des variables, leur type et si elles pouvaient être Nul ou non, c'est à dire sans valeur ou non.



	Nom de la colonne	Type de données	Autoriser les valeurs Null
🔑	nomClient	nvarchar(50)	<input type="checkbox"/>
	typeClient	nvarchar(50)	<input type="checkbox"/>
	nbCommande	int	<input type="checkbox"/>
▶			<input type="checkbox"/>

Figure 34 : Création de la table Client

Nous pouvons voir ici, Figure 34, la création de la table « *Client* » avec son nom associé à une petite clé jaune pour indiquer qu'il s'agit d'un identifiant, son type de client stocké dans une variable de type « *nvarchar(50)* » soit une chaîne de caractère de longueur cinquante maximum ainsi que le nombre de commandes au format « *int* » donc entier. Nous observons aussi qu'aucune des trois variables ne peut être Null car nous ne souhaitons pas avoir de client avec des informations manquantes.

Le second point après la mise en place des entités était la création des triggers ou déclencheurs; ces derniers permettent d'intercepter les insertions par défauts dans la base de données pour les traiter une par une. En effet nous ne voulions pas que l'on puisse créer des clients à tout va car un client peut passer plusieurs commandes et il ne fallait pas le créer autant de fois qu'il a passé de commande.

Le trigger de la Figure 35 va être exclusivement à l'écoute de toutes les insertions dans la table « Stock » pour se déclencher dès qu'il en détecte une. Il va alors regarder d'où vient cette insertion et va ensuite la traiter selon le cas où l'on se trouve. S'il s'agit d'une palette qui vient d'être chargée il incrémentera le nombre de palette en stock, si au contraire la palette est déchargée il décrémentera ce nombre. Il agira aussi sur le nombre de palettes en cours; c'est à dire, combien peut-on faire de palette si nous arrêtons juste l'approvisionnement et laissons fonctionner la chaîne. Par définition ce nombre doit incrémenter au début de la chaîne quand on produit assez de module pour une palette et à la fin, au stock, quand on charge une palette. Ce trigger s'occupe donc de décrémenter la valeur associée quand il stocke une palette. La décrémentation se fait par un trigger propre à la table « *EnCours* ».

```

24 CREATE TRIGGER tr_Stock
25 ON Stock
26 INSTEAD OF INSERT
27 AS
28 BEGIN
29     SET NOCOUNT ON;
30
31     --On regarde nbPaletteS d'avant cette insertion et on voit si on a chargé ou déchargé une palette
32     DECLARE @oldS int;
33     SET @oldS = (SELECT nbPaletteS FROM Stock WHERE idStock = 1);
34
35     -- Si on a chargé une palette alors on recopie le nombre en stock et on décrémente le nombre d'en-cours
36     IF @oldS < (SELECT nbPaletteS FROM INSERTED)
37     BEGIN
38         UPDATE Stock SET nbPaletteS = (SELECT nbPaletteS FROM INSERTED) WHERE idStock = 1;
39         UPDATE EnCours SET nbPaletteEC = (SELECT nbPaletteEC FROM EnCours WHERE idEnCours = 1) - 1 WHERE idEnCours = 1;
40         UPDATE EnCours SET derniereMaj = GETDATE() WHERE idEnCours = 2;
41     END
42     ELSE -- Si on a déchargé on décrémente juste le nombre de palette dans le stock
43         UPDATE Stock SET nbPaletteS = (SELECT nbPaletteS FROM Stock WHERE idStock = 1) - 1 WHERE idStock = 1;
44
45     UPDATE Stock SET derniereMaj = GETDATE() WHERE idStock = 1;
46 END

```

Figure 35 : Trigger pour la table Stock

Il existe ainsi trois triggers dans cette base CCP; un pour le stock comme nous venons de le voir, un pour la table « *EnCours* » et le dernier pour les commandes qui gère le cas des clients déjà existant auxquels il faut juste augmenter de un leur nombre de commandes sans pour autant les récréer.

La dernière étape est la liaison entre la base de données et les variables qui sont pour l'instant stockées sur un serveur OPC. Pour ce faire il était nécessaire de passer par le logiciel Cogent Datahub qui contient une option pour faire la liaison entre SQL et OPC. Cette option affichait alors la fenêtre ci-dessous.

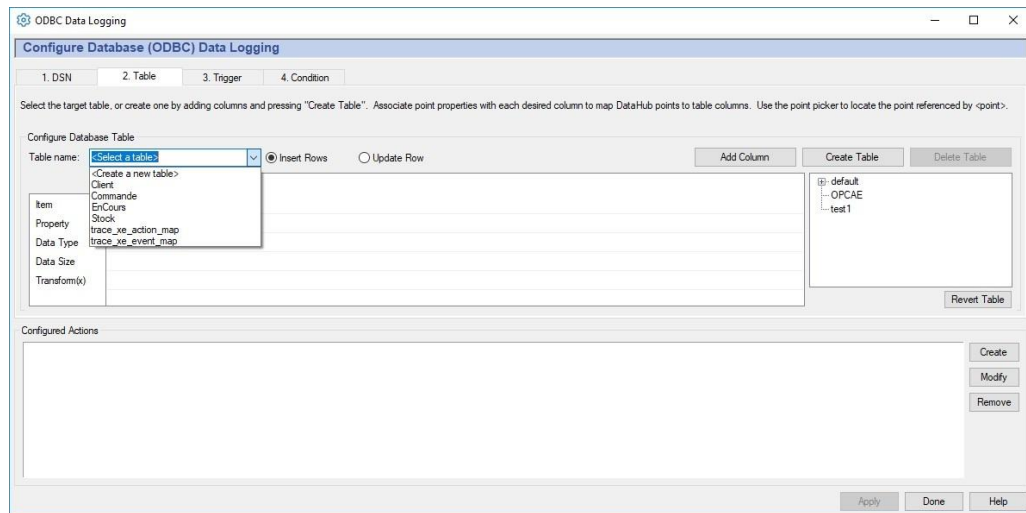


Figure 36 : Fenêtre de configuration entre SQL et OPC

Sur cette fenêtre ci-dessus, nous retrouvons toutes nos tables présentes dans la base de données. En choisissant une d'entre elles nous avons alors accès à toutes ses variables. Et de là, nous pouvons relier les variables *OPC*, présentes dans le petit encadré à droite de l'écran, à celles de la base *CCP*. Puis avec l'onglet « *Trigger* », qui au passage n'a rien à voir avec les triggers *SQL* cités précédemment, nous indiquons quelle variable *OPC* serait à l'initiative d'une insertion dans la table choisit sur l'écran précédant. Il ne reste plus qu'à créer ces fonctions avec le bouton « *Create* » et à recommencer pour toutes les tables que l'on souhaite.

Cette configuration sur *Cogent Datahub* était le dernier point pour avoir une base de données pleinement fonctionnelle.

F. Difficultés rencontrées

Un projet ne serait pas complet sans ses difficultés et effectivement notre projet n'a pas dérogé à la règle.

La première difficulté a été de prendre en main les logiciels PC Worx, Factory I/O, WebVisit et OPC Server. En effet, aucun d'entre nous n'avait connaissance de ces derniers avant le début du stage. De plus la liaison entre Factory et les automates nous a pris énormément du temps puisqu'il fallait passer par OPC Server.

La seconde difficulté a été la supervision. Celle-ci était initialement prévue sur le logiciel Panorama E², mais celui-ci posait problème au moment de lire les variables sur le serveur. Nos automates étant en fonctionnement NPN (Négatif Positif Négatif), c'est à dire que les valeurs logiques valent soit 0 soit -1, Panorama E² prenait la valeur -1 pour une chaîne de caractères et non un booléen. Nous avons donc changé de logiciel pour développer la supervision et nous sommes passé sur WebVisit avec beaucoup moins de fonctionnalités mais compatible à 100% avec nos automates.

La vitesse d'exécution du programme dans les automates a aussi été également un problème car nous avions besoin d'incrémenter des variables; par exemple le nombre de module dans la caisse. En effet, le capteur qui compte les modules mise dans les caisses, s'incrémentait deux fois consécutives donc nous avions des caisses qui partaient avec trois modules au lieu de cinq.

Finalement, la difficulté majeure du projet, celle qui nous a extrêmement limités dans l'environnement virtuel, est la limitation imposée par Factory I/O de cent vingt-huit variables pour configurer le driver OPC. Cette limitation nous a donc contraint à enlever deux racks pour le stockage ainsi que le bras mécanique du début de chaîne pour n'avoir que 128 variables. Nous sommes heureusement arrivés à conserver les processus les plus complexes en arrivant exactement à 128 variables.

IV. Résultats

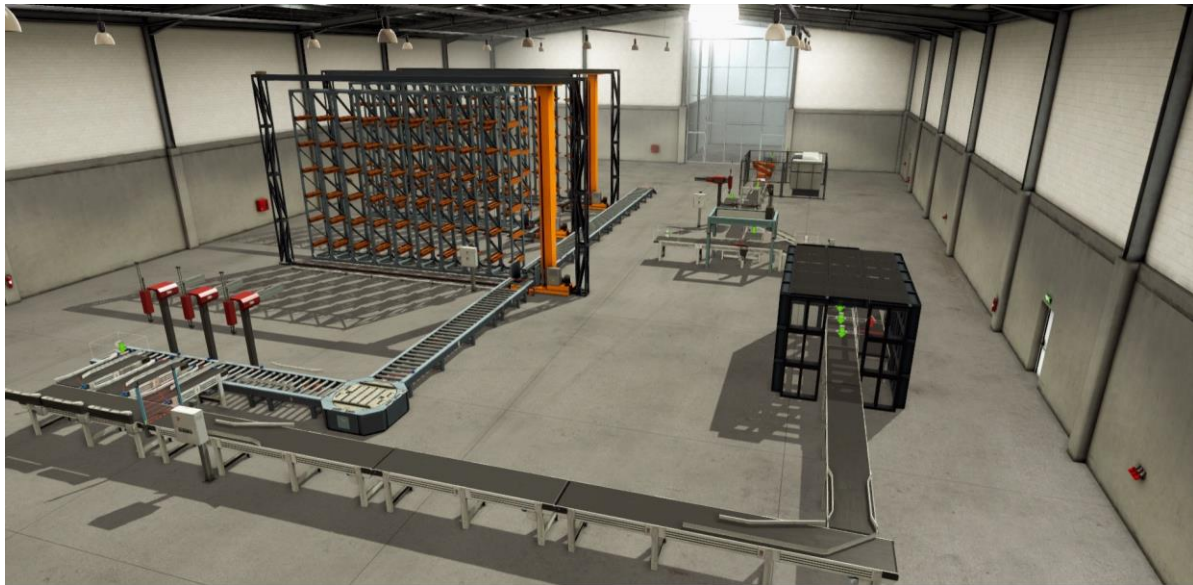


Figure 37 : Visuel de la chaîne industrielle sur Factory I/O

Dans la Figure 37 nous retrouvons tous les processus mentionnés précédemment dans ce rapport. Sont aussi visibles deux racks supplémentaires qui pour l'instant ne sont pas utilisés à cause de la restriction du nombre de variable dans Factory I/O. Voir A_1 page 41 pour le détail de chaque processus.

Résultats

Messages

	idEnCours	nbPaletteEC	derniereMaj	
1	1	3	2018-07-04 14:16:08.430	

	idStock	nbPaletteS	derniereMaj		idEnCours
1	1	7	2018-07-04 14:13:08.750		1



	nomClient	typeClient	nbCommande
1	DAMIEN	Occasionnel	3
2	EMILLY	Occasionnel	2
3	FLORENCE	Occasionnel	2
4	JEAN	Occasionnel	10
5	MAXENCE	Nouveau	1
6	MINA	Occasionnel	2
7	YOUSOUF	Occasionnel	2

	idCommande	nbPaletteC	dateCommande	nomClient	idStock
1	2	2	2018-06-05 12:02:01.453	DAMIEN	1
2	3	1	2018-06-05 12:02:17.893	DAMIEN	1
3	4	1	2018-06-05 12:02:37.457	DAMIEN	1
4	5	2	2018-06-05 12:04:24.700	YOUSOUF	1
5	6	1	2018-06-05 12:04:43.580	YOUSOUF	1
6	7	3	2018-06-05 12:06:10.983	MAXENCE	1
7	8	1	2018-06-14 11:56:17.550	JEAN	1
8	9	0	2018-06-14 11:56:34.770	JEAN	1
9	10	0	2018-06-14 12:01:49.213	JEAN	1
10	11	1	2018-06-14 12:20:44.063	FLORENCE	1
11	12	0	2018-06-14 12:21:02.217	FLORENCE	1

Ci-joint, Figure 38, le résultat d'une simulation de trente minutes lors desquelles nous avons passé sept commandes avec divers noms de clients. Il est important de préciser qu'il y avait d'autre simulation précédemment. C'est pourquoi la table « Client » a enregistré plusieurs commandes passées par plusieurs clients. Nous pouvons remarquer que les clients qui ont passé plus d'une commande sont maintenant des clients « Occasionnels ». Il y a aussi 7 palettes dans le stock ainsi que 3 qui sont en cours de fabrication.








Figure 38 : Base CCP après une simulation de 20 minutes

Les trois images ci-dessous montrent la supervision des trois automates (API_11, API_21 et API_31), permettant à un opérateur de contrôler le bon fonctionnement de l'ensemble. Cette supervision est mise en place grâce au logiciel WebVisit de chez Phoenix Contact.


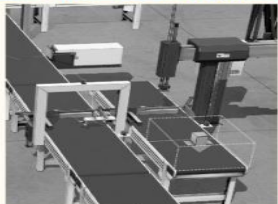

SUPERVISION API 192.168.0.11 - Accueil  

Accueil **Fraisage** **Assemblage** **Mise en caisse** **Msg**



Nombre de pièces présentes sur la chaîne :

Algorithmes succints des trois premières étapes :






  

Plaque en attente	Socle en attente	Couvercle en attente	Module en attente	Kit en attente
Plaque en cours de fraisage	Module en cours d'assemblage		Caisse en attente	
Plaque évacuée	Module évacué		Caisse en cours de remplissage	
			Caisse évacuée	


SUPERVISION API 192.168.0.21 - Accueil  

Accueil **Tri** **Mise sur palette** **Msg**

Nombre de pièces présentes sur la chaîne :

Algorithmes succints des deux étapes :



Carton S	Carton M	Carton L	Carton S attend	Carton M attend	Carton L attend
Carton S évacué	Carton M évacué	Carton L évacué	Palette devant S	Palette devant M	Palette devant L
			Carton S positionné	Carton M positionné	Carton L positionné
			Palette évacuée		

SUPERVISION API 192.168.0.31 - Accueil  

Accueil **Rack 1** **Msg**

Nombre de pièces présentes sur la chaîne :



Algorithmes succints du rack 1 **Commandes**



Charge
En mouvement
Decharge

Nombre de palette pour la commande : 

Nom du client :



Commande en cours :

On a lancé une simulation de quelques minutes. Après notre simulation on a :

- Sept produits dans le stock.
- Un client nommé YOUSOUF passe une commande qu'on a lancé depuis la supervision (cas de la première image).
- Automatiquement le produit sera déchargé du rack pour la livrer au client (cas de la deuxième image ci-dessous).

SUPERVISION API 192.168.0.31 - Accueil

Accueil
Rack 1
Msg

Nombre de pièces présentes sur la chaîne :



7

Algorithmes succints du rack 1



Charge
En mouvement
Decharge

Commandes

Nombre de palette pour la commande : ▼

Nom du client :

Valider

Commande en cours :



Figure 39 : Copies d'écran des deux écrans principaux de l'API 3

V. Tests et validations

Les phases de tests se sont égrainées tout au long de la réalisation du projet. De part l'organisation, étape par étape, nous suivions chacune de ces nouvelles étapes par un test rigoureux en simulant de A à Z l'installation, avec A étant notre première programmation du projet et Z la dernière étape que l'on a programmée. Cela nous permettait de vérifier que l'enchaînement de tous les processus se faisait comme nous le souhaitions.

Pour ce qui est des validations, nous laissions tourner pendant environ trente minutes l'installation dans sa globalité (chaîne virtuelle, automates, et supervisions). Au-bout de ce laps de temps nous considérons comme validé notre avancement si aucun problème n'était survenu.

Ci-dessous le tableau des fonctionnalités attendues et leur état de validation. (cf. *page 13, 14* pour le détail de chaque référence)

Figure 40 : Tableau des fonctionnalités attendues

Référence	État	Commentaire
C1	À améliorer	Environ 75% des ressources utilisées dans Factory I/O
C2	Validé	
C3	Validé	Trois automates permettent de piloter la simulation
P1	Validé	Des caméras sont présentes sur chaque processus
P2	À améliorer	Un et un seul client peut commander, à voir pour plusieurs
P3	Validé	La supervision permet d'enclencher ou non la fin de journée
E1	Validé	
E2	Validé	La supervision est très facile à comprendre et à prendre en main
D1	Fait	Les trois documentations sont rédigées
D2	Fait	
D3	Fait	

CONCLUSION

Ce stage fut très enrichissant grâce à son côté multidisciplinaire avec les trois cursus présents : Master Informatique, DUT GEII et DUT Informatique. En effet le contexte de ce projet m'a permis dans un premier temps de revoir la base de données que j'avais précédemment vu l'année dernière en master première année d'informatique. Dans un second temps, le projet m'a donné un bon aperçu du travail en groupe avec des spécialités différentes.

Le projet est aujourd'hui pleinement fonctionnel, j'entends par là que toute la simulation est automatisée, la supervision est développée et installée sur les trois automates. Quant à la base de données Client Commande Palette, et bien, cette dernière est implémentée, testée et validée.

Des évolutions sont aussi à prévoir pour perfectionner l'installation, premièrement en mettant en place la seconde base de données qui n'est pour l'instant qu'à l'état de croquis. Deuxièmement, rajouter des processus dans l'environnement de simulation pour agrandir le scénario, cependant cela ne sera possible que lorsque le nombre de variable dans *Factory I/O* ne sera plus limité à cent vingt-huit.

Aussi, j'ai pu participer aux **JIPN2018 (Journées des Innovations Pédagogiques Normandes)** qui a eu lieu au Havre les 2 et 3 juillet 2018 sur la thématique : « Plaisir d'apprendre, plaisir d'enseigner » où j'ai présenté mon projet de stage et après j'ai fait des démonstrations durant les deux jours. En final, les gens étaient impressionnés du résultat du projet.

Lexique

Référence	Terme	Définition
L_1	ADRESSE IP	Numéro d'identification, permanent ou non, attribué à une machine
L_2	API	Automate Programmable Industriel, matériel permettant l'automatisation de processus industriel
L_3	BADO	Acronyme de BAse de DONnées
L_4	CEI 61131	Norme industrielle de la Commission électronique internationale qui définit cinq langages de programmation
L_5	DATAHUB	Hébergeur de données
L_6	GRAFCET	Langage de programmation industriel
L_7	MODBUS	Protocole de communication
L_8	OPC	Type de serveur hébergeant des données
L_9	SQL	Langage de programmation de base de données
L_10	SQL SERVER	Système de gestion de base de données
L_11	STRUCTURED TEXT	Langage de programmation d'automates industriels
L_12	SWITCH	Équipement permettant l'interconnexion de plusieurs machines sur un même réseau
L_13	YOUMADA	Nom donné à notre équipe YOUssouf MAXence DAMien

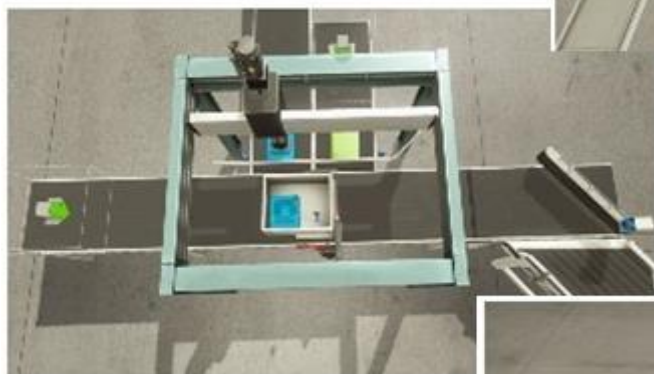
Annexes



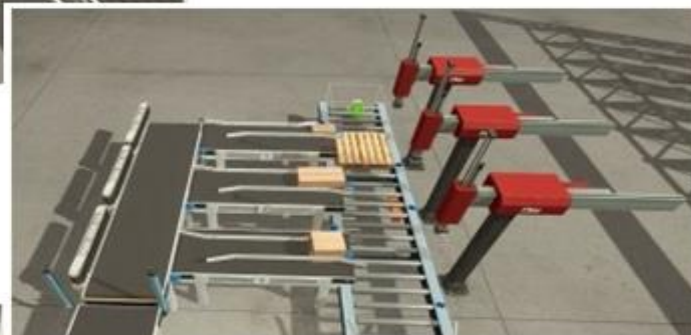
Fraisage



Assemblage



Mise en caisse

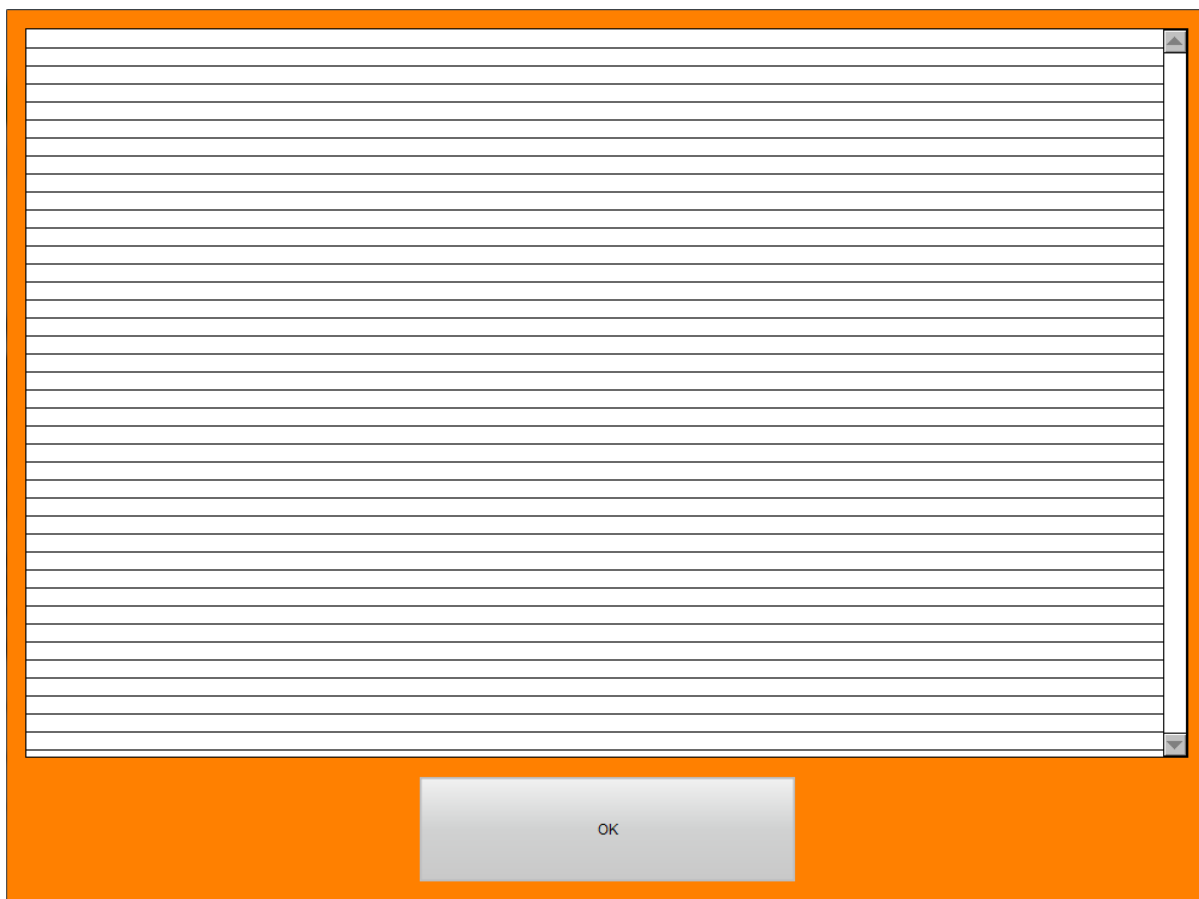


Tri & Mise sur palette



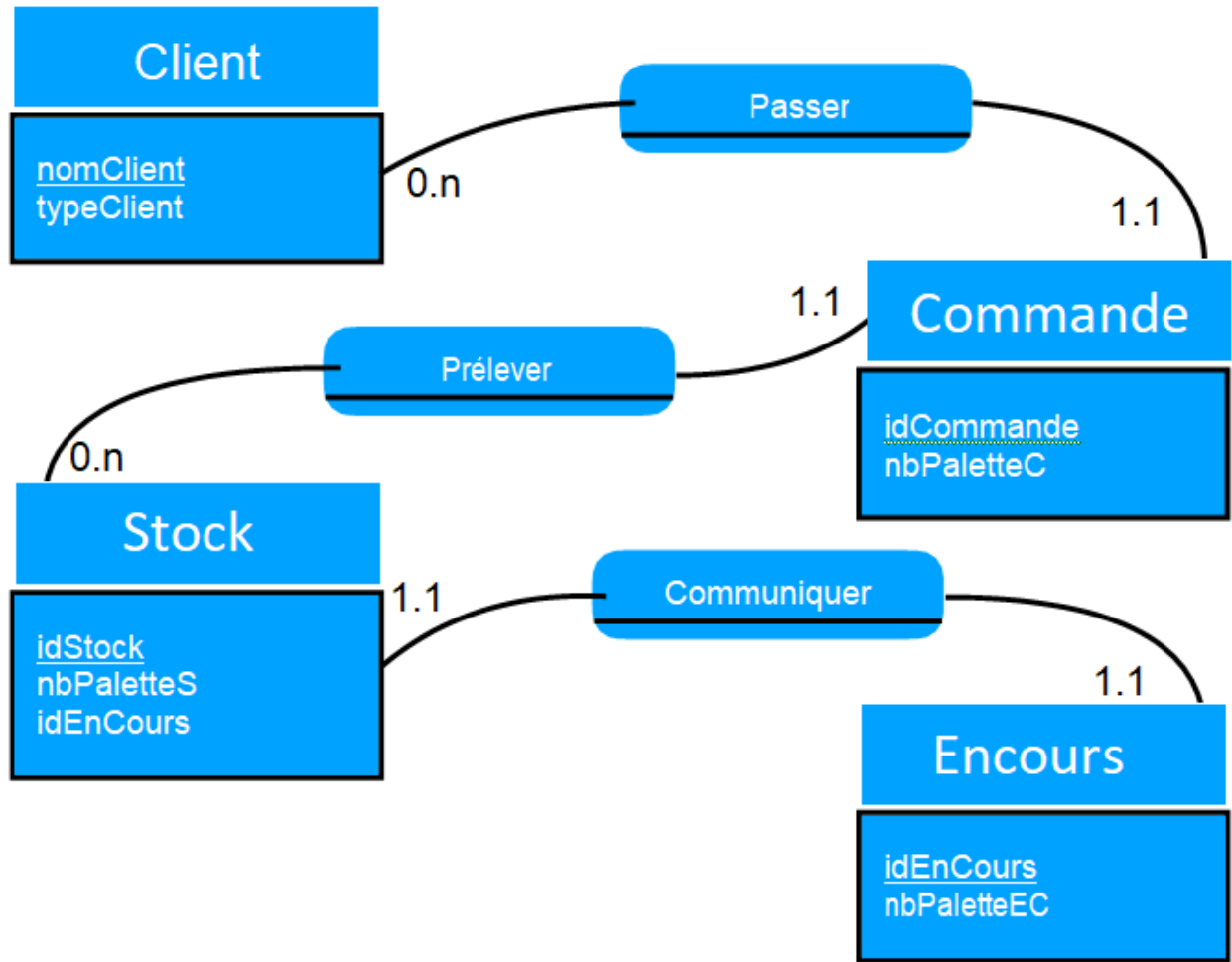
Stockage

A_1 : Photos de tous les processus de l'installation sur Factory I/O

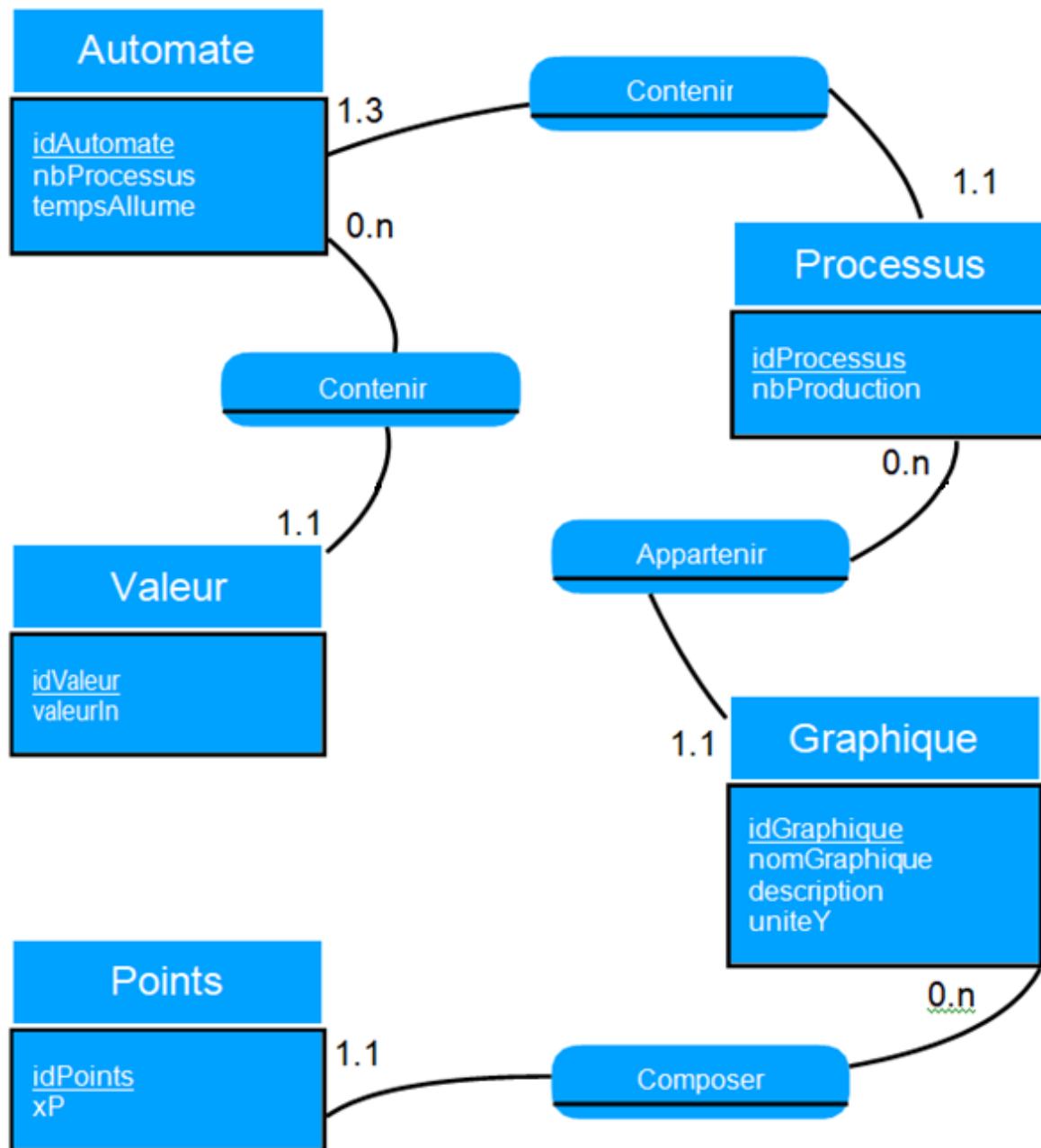


The image shows a screenshot of a software window. The window has an orange border. Inside, there is a large white rectangular area with horizontal lines, resembling a text editor or a list. At the bottom center of the window, there is a gray button with the text "OK".

A_2 : Onglet pour les messages d'erreurs



A_3 : Schéma entité association de la base de données CCP



A_4 : Schéma entité association de la base de données BTA

Résumé

Ce stage se déroulait au Havre et plus précisément au pôle de recherche PIL (Pôle Ingénieur et Logistique). Nous étions trois étudiants sur le même projet soit un DUT Informatique, un DUT GEII et moi-même un Master Informatique. Le projet consistait à simuler un processus industriel et logistique par le biais du logiciel Factory I/O puis à piloter ce dernier avec trois API (Automates Programmables Industriels) qui étaient bien réels. Une supervision était aussi à développer, elle permettrait d'avoir un regard sur l'installation sans avoir à être à proximité de celle-ci. Et la dernière chose à mettre en place était deux bases de données, la première sauvegarderait toutes les commandes ainsi que les clients. La deuxième base est plus technique car elle enregistre des informations comme le temps de réponse de chaque capteur ou encore la consommation énergétique de l'installation.