2014/ 2015

[RAPPORT DE PROJET EI4 AGI : ROBOTISATION D'UN SYSTEME TRANSITIQUE]

Réalisé par Mickael Auvin – Xiang Fong – Jordan Durand

Encadré par Laurent Hardouin

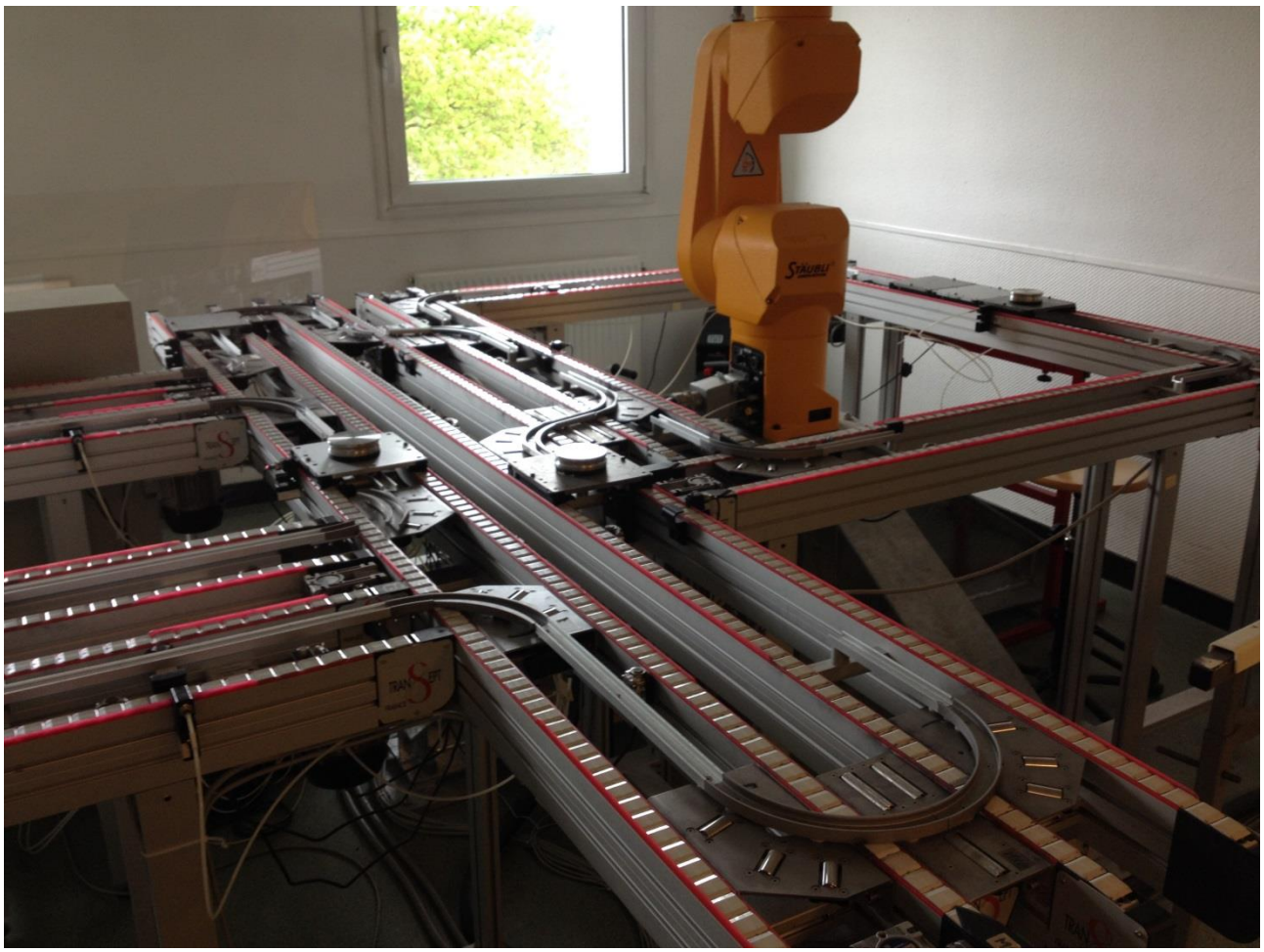


Table des matières

1 Remerciements

Nous tenons tout d'abord à remercier notre tuteur, Monsieur Laurent Hardouin, pour nous avoir proposé ce projet et nous avoir suivis durant le projet. Nous tenons à remercier également, Monsieur Rémy Guyonneau, pour le TP Profibus qui nous a permis de programmer le convoyeur. Et enfin, l'ISTIA pour nous avoir fourni du matériel fonctionnel, proche de la réalité industrielle afin de mener à bien notre projet tutoré.

2 Définitions

- Programmer : C'est le façon pour concevoir et réaliser les fonctions de l'application (automate, robot, logiciel, etc..). Le programmation nous permet de définir et de régler les comportements du système.

- Superviser : C'est une technique utilisée dans l'industrie pour suivre et piloter en temps réel un système automatisé. La supervision permet l'acquisition de différentes données (mesures, détection des capteurs, alarmes) permettant de visualiser le système sans obligatoirement être à proximité.

- Automatiser : C'est à dire mettre le système déroule automatiquement sans la participation humaine.

- Base de données : Une base de données est un outil ce qui nous permet de stocker et d'acquis les informations bruts. L'administration de la base de données permet la réalisation de la supervision du système.

3 Introduction

De nos jours, les entreprises utilisent des systèmes de production automatisés pour assurer une productivité élevée et donc un meilleur rendement. Leurs polyvalences leurs permettent aussi une adaptabilité rapide aux différents besoins du marché et du client. Les systèmes de convoyage automatisés sont très présents dans l'industrie agroalimentaire ou les centres de tris afin de gagner en facilité et rapidité la manutention des éléments au sein de l'infrastructure.

En effet des machines telles que des robots ou des palettiseurs permettent le chargement ou le déchargement des éléments sur le convoyeur de façon autonome.

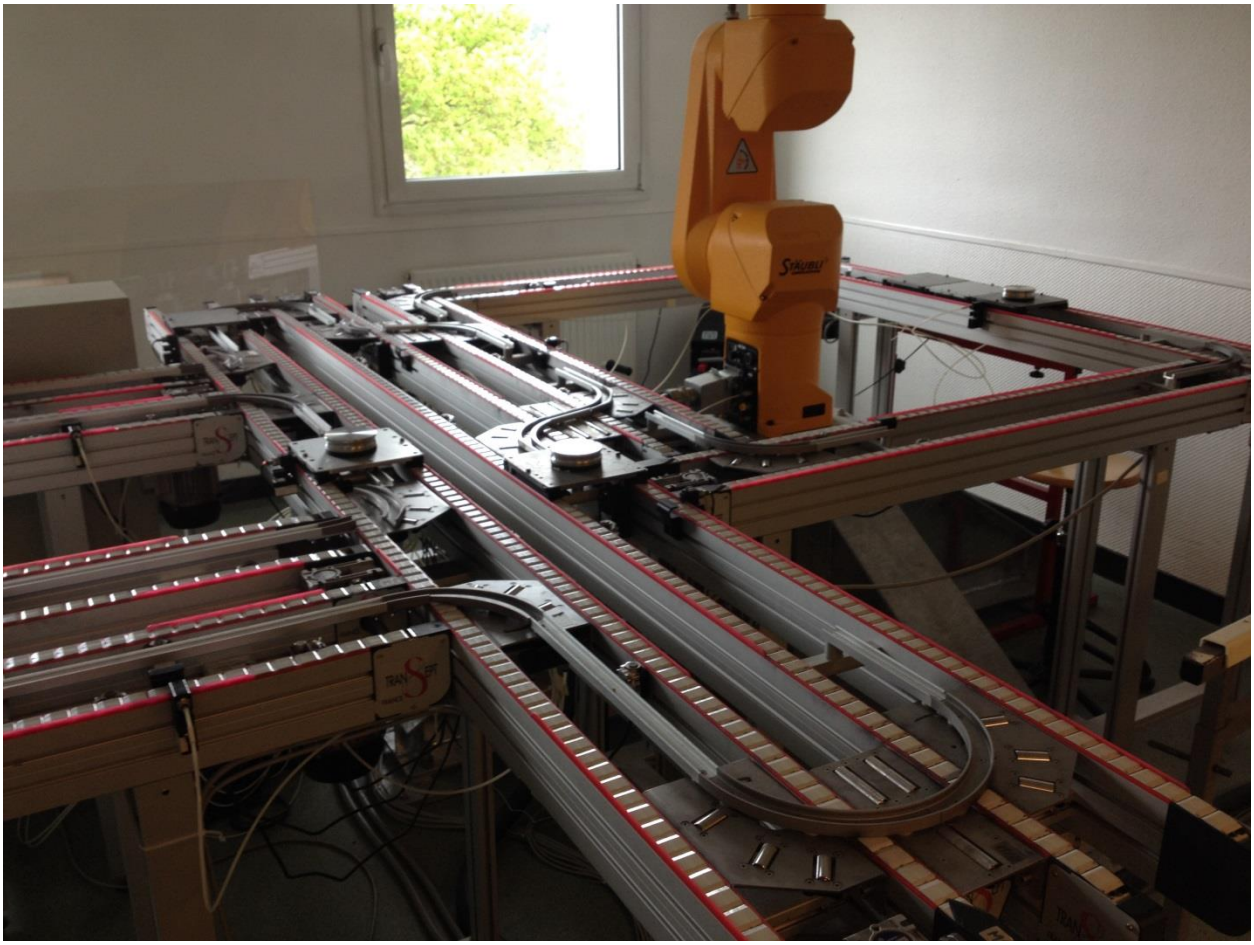


Figure 1 : Ligne transitique de notre projet

La ligne transitique située en salle 216 à l'école d'ingénieur ISTIA est une réplique des lignes automatisées que l'on peut retrouver dans le monde industrielle. Ce système est composé d'un tapis roulant permettant aux palettes d'avancer, de butées pour réguler le trafic et d'aiguillages et d'aiguillages. Un bras robotisé Staubli fait aussi partie du système, il permet le chargement et le déchargement en pièces des palettes.

Notre projet de 2e année de cycle ingénieur consiste à automatiser le convoyeur permettant le transport de palettes en interaction avec le robot. Les différents états de production du système pourront être visibles en temps réel via la partie supervision.

4 Cahier des charges

4.1 Ligne transitique

4.1.1 Description fonctionnelle

Cette ligne de production est composée de trois parties (contrôlée chacune par un automate distinct) : un magasin rectangulaire, un hippodrome et deux épis. Afin de communiquer correctement entre toutes ces différentes parties des aiguillages et des butées sont présents sur la ligne transitique.

4.1.1.1 Partie Hippodrome

Cette partie permet l'aiguillage des palettes entre les parties épis et magasin. Au sein de l'hippodrome une lecture de l'état de la palette est effectuée. Cette lecture est binaire, 0 la matière première sur la palette n'a pas été traitée, 1 la matière première sur la palette a été traitée. Si la matière première d'une palette n'a pas été traitée alors elle est envoyée vers un des deux épis sinon elle est reste dans cette même partie du convoyeur.

4.1.1.2 Partie Magasin rectangulaire

Les palettes sont chargées et déchargées de matière première dans le magasin grâce au robot Staubli. Un module d'écriture permet d'inscrire sur la palette que la matière première sur cette dernière n'a pas subie de traitement.

Sur le convoyeur d'entrée/sortie du magasin une lecture de l'état de la palette est effectuée. Cette lecture est binaire, 0 la matière première sur la palette n'a pas été traitée, 1 la matière première sur la palette a été traitée. En fonction de cette valeur un aiguillage gère la destination de la palette. Si la matière première d'une palette n'a pas été traitée alors elle est envoyée vers l'hippodrome sinon elle est envoyée vers le magasin.

4.1.1.3 Partie Epis

Les palettes non traitées sont envoyés dans ces deux parties afin qu'elles subissent des opérations de traitement (non géré dans ce projet). A la fin de ces dernières un module d'écriture inscrit sur la palette que la matière première a été traité.
Afin d'améliorer la vitesse de production il est préférable que les palettes soit envoyées dans les épis en fonction du nombre des palettes présentes dans ces derniers.
Chaque épi n'accepte qu'une palette en usinage à la fois. Si le premier épi n'est pas disponible alors la palette est envoyée vers le second. Si une palette est déjà présente dans ce dernier alors la palette continue sur l'hippodrome.

Un module de lecture est également présent dans cette partie afin que les palettes usinées soient envoyées vers la partie magasin tandis que les autres continues sur l'hippodrome afin qu'ils subissent l'usinage selon la disponibilité des épis.

4.1.2 Communication

La ligne transitique est composée de 3 automates (magasin, hippodrome et épis). Il faut donc effectuer des sémaphores entre eux pour la lecture des mots communs.

De plus l'automate de la partie magasin ce cette ligne doit pouvoir communiquer avec le robot Staubli afin de lancer des ordres de début et de fin d'un cycle. Il y a deux opérations à effectuer :

- Déchargement de matière traitée : Lors de l'arrivée d'une palette en C2 le robot Staubli doit décharger la matière traitée sur la palette. Un ordre de fin de cycle sera renvoyé depuis le robot Staubli afin que la palette puisse être chargée plus loin dans la chaine.
- Chargement de matière première : Lors de l'arrivée d'une palette en C3 le robot Staubli doit charger la matière première sur la palette. Un ordre de fin de cycle sera renvoyé depuis le robot Staubli afin que la palette puisse être libérée du magasin.

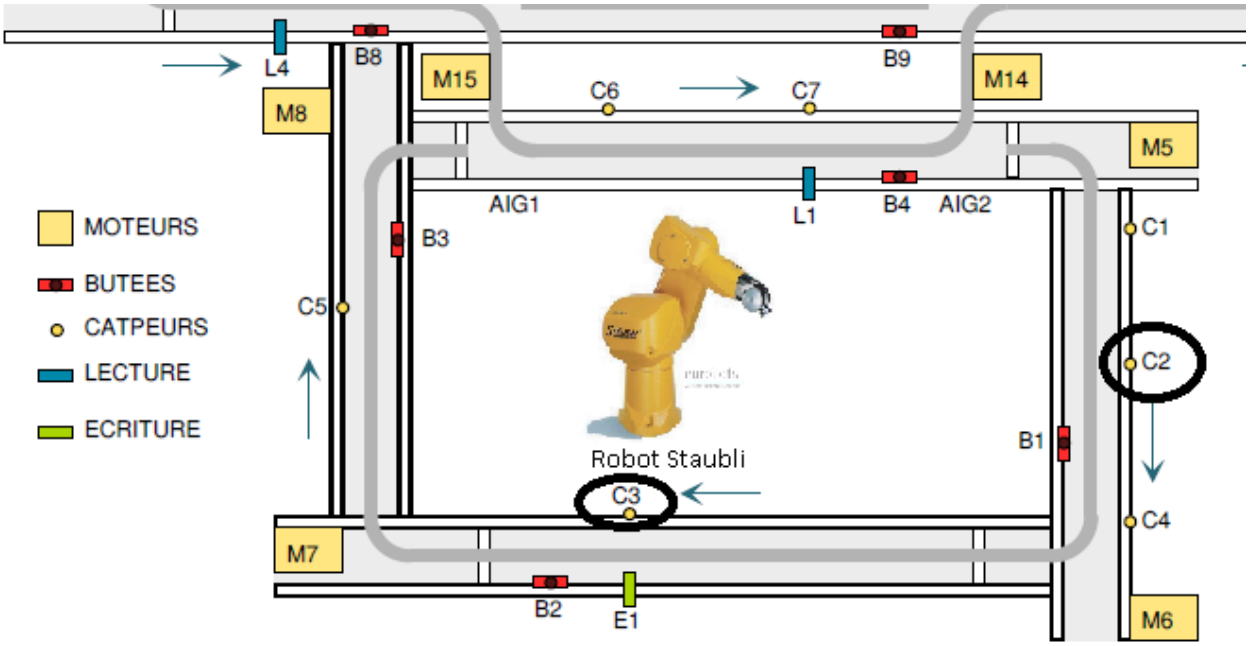


Figure 2 : Représentation des capteurs C2 et C3 sur la ligne transitique

4.1.3 Gestion de défauts

Comme tout système industriel, parfois des défauts inopinés peuvent survenir. Pour cela le système doit pouvoir indiquer à toute personne d'une défaillance. Une voyant lumineux étant présent sur le système de convoyeur nous allons l'utiliser afin d'indiquer qu'une palette est bloqué à n'importe quel endroit du convoyeur. Pour cela nous allons réutiliser les capteurs de présence initialement prévu pour l'aiguillage des palettes.

4.2 Robot

4.2.1 Communication

Le robot Staubli possède 12 entrées et 6 sorties qui sont reliées à la station Magasin (automate N°5).

4.2.2 Définition des opérations effectuées

Le robot Staubli est en charge du déchargement et du chargement des différentes palettes présentent dans le magasin. Quand une palette traitée arrive à l'entrée du magasin, l'automate envoie une demande de déchargement au robot qui effectuera le déchargement et enverra un signal à l'automate comme quoi le déchargement de la palette a bien été effectué. Puis, à la sortie de magasin, si une palette se présente, l'automate envoie une demande de

chargement au robot qui va effectuer un déplacement vers le dépôt, prendre une pièce et aller charger la palette. Le robot enverra ensuite une condition de chargement effectué à l'automate.

4.3 Supervision

La partie supervision sera composée de deux grandes parties, la partie interface où l'utilisateur pourra visualiser en temps réel le système automatisé et la partie base de données où l'on sauvegardera les changements d'états de différentes entrées du système.

De plus l'utilisateur pour enregistrer le nombre de palette qu'il veut émettre en fabrication. Toutes ces données seront enregistrées dans une base de données. Ces dernières seront envoyées aux automates afin de lancer la fabrication.

4.3.1 IHM

Cette partie sera codée dans un premier temps en HTML, PHP pour que l'on puisse suivre l'avancement des différentes palettes du système ainsi que la position du robot Staubli sur une page internet.

Dans un deuxième temps, selon l'avancement du projet, on pourra concevoir une application android pour superviser la plateforme via une tablette android.

Ces deux applications iront chercher les informations des différents états des entrées des automates dans une base de données où elles sont stockées.

La partie supervision pourra aussi interférer avec le système afin de minimiser l'accumulation de palette uniquement si le mode est manuel.

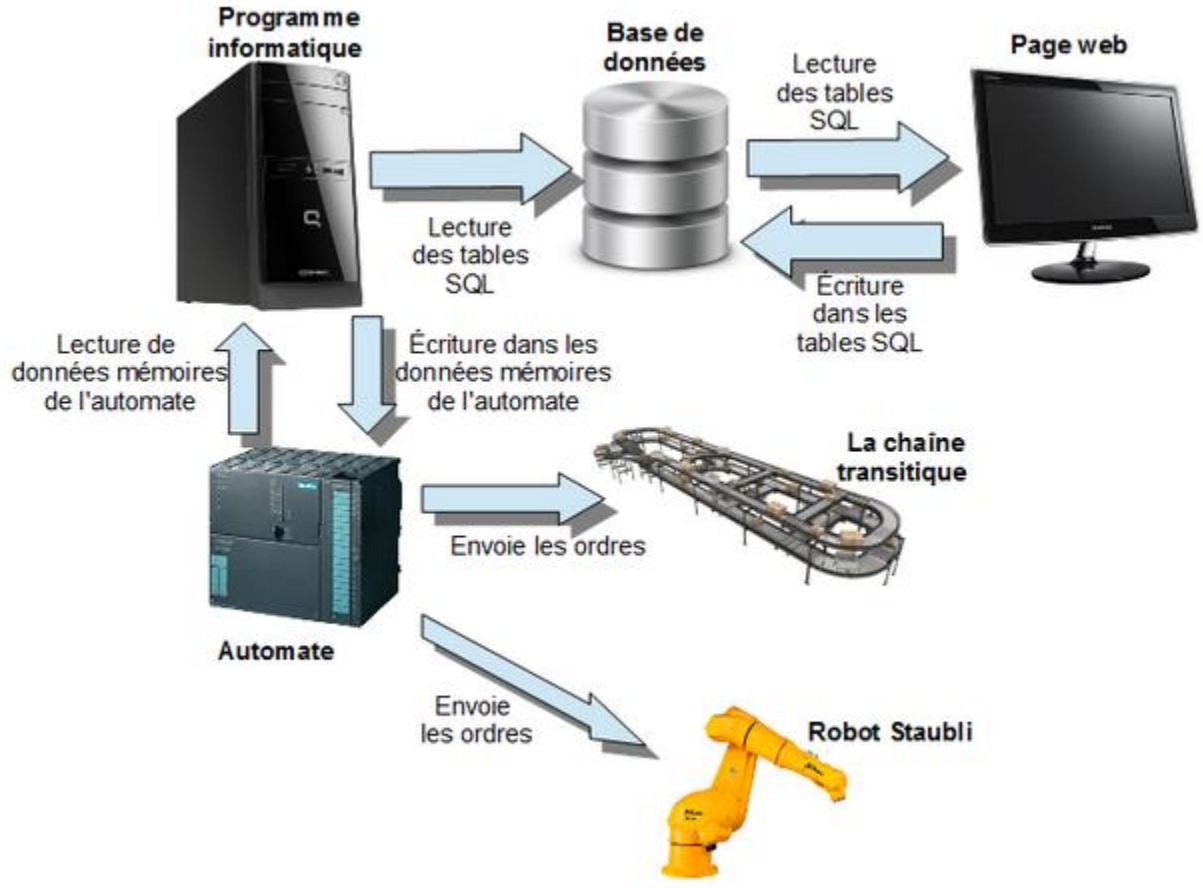


Figure 3 : Architecture du projet

4.3.2 Base de données

Cette partie sera codée en langage C via le logiciel CodeBlocks. La programmation permettra d'aller chercher les états des capteurs, l'état du défaut et des autres entrées dans les trois automates pilotant le système.

Il faudra stocker les données reçues dans une base de données mysql et mettre à jour cette base de données dès qu'une entrée change d'état avec la date et l'heure de ce changement. On pourra la programmer telle qu'un message d'alerte soit envoyé à l'IHM dès qu'un défaut est signalé.

5 Organisation du projet

5.1 Vue d'ensemble

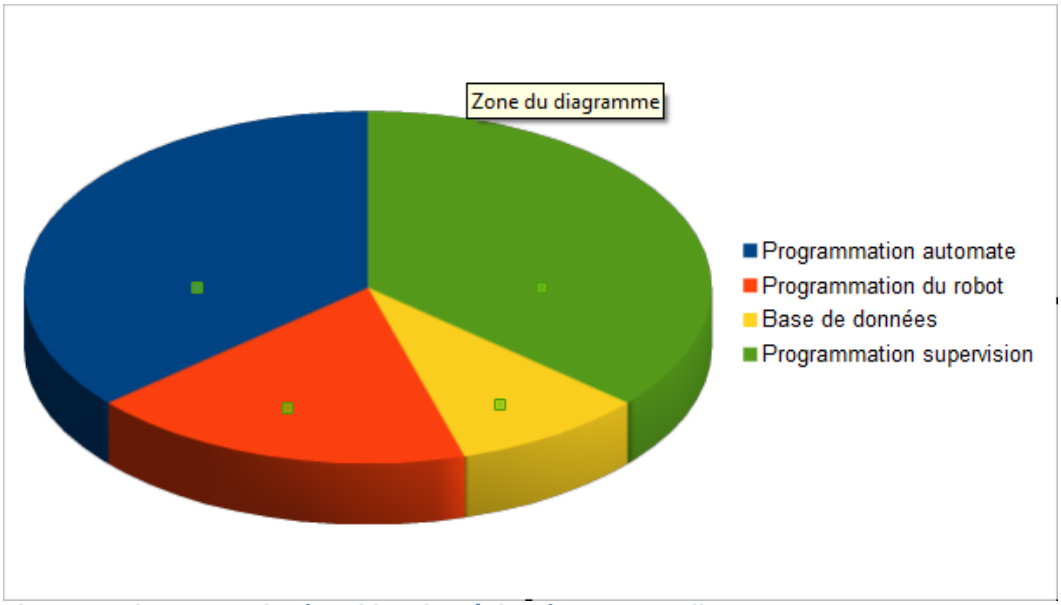


Figure 4 : Diagramme de répartition des tâches à en accomplir

Au niveau du projet dans son ensemble, la partie programmation automate et supervision furent les deux plus grosses parties de ce projet, donc celles qui ont nécessité le plus de temps.

La création de la base de données fut assez rapide à réaliser même s'il a fallu y revenir plusieurs fois pour la modifier au fur et à mesure que le programme avançait.

La partie programmation du robot fut plus longue au niveau de la compréhension du langage robotique et du calibrage du robot. Une fois cela compris et codé, il a fallu le faire interagir avec les automates.

5.2 Par personne

5.2.1 Mickael

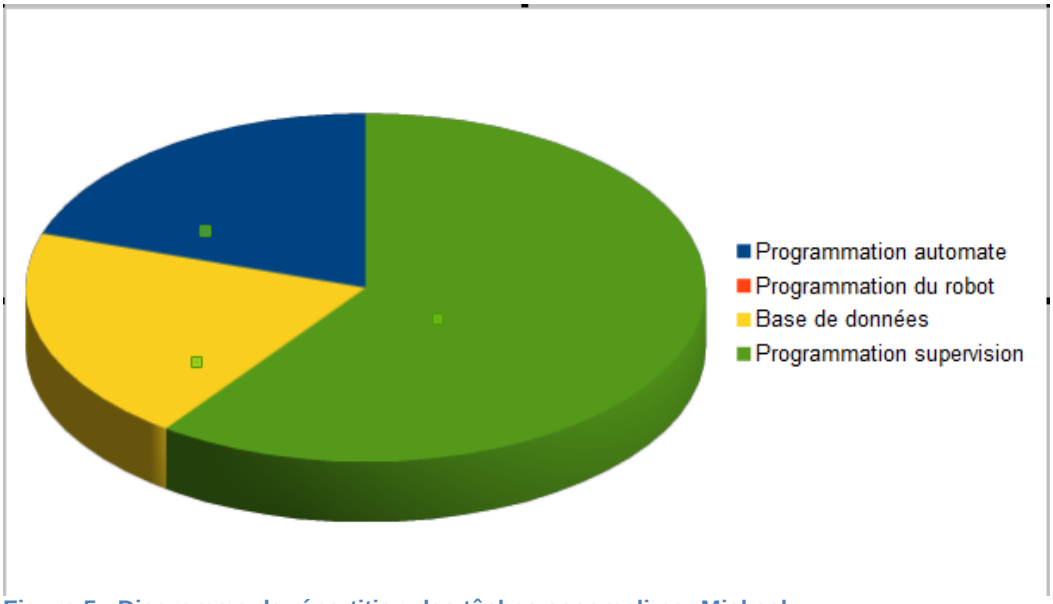


Figure 5 : Diagramme de répartition des tâches accompli par Mickael

Partie dominante: La programmation supervision
Partie moins importante: Base de données, Programmation automate.

5.2.2 Jordan

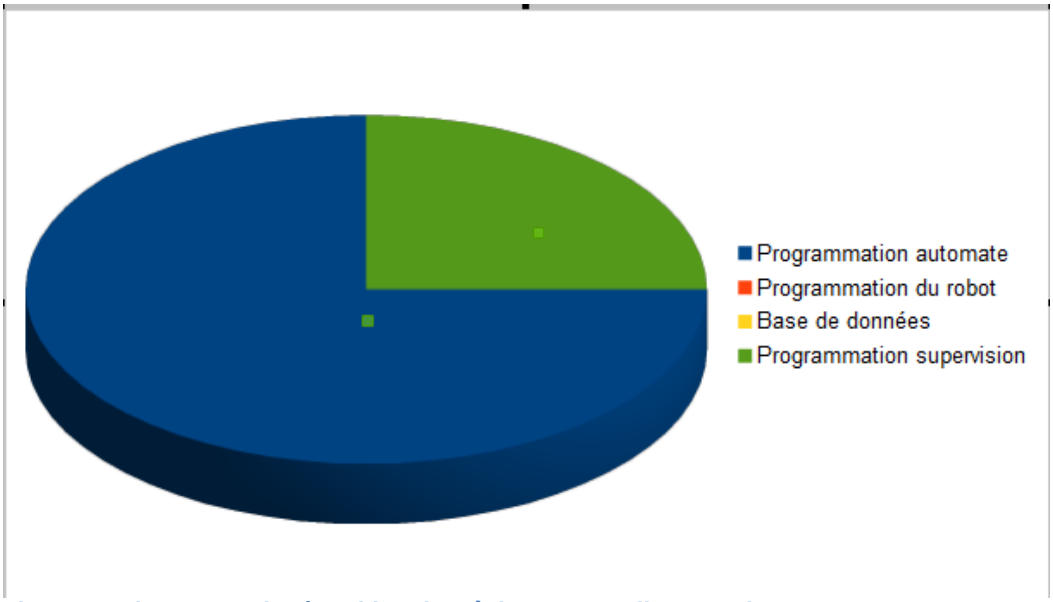


Figure 6 : Diagramme de répartition des tâches accompli par Jordan

Partie dominante: La programmation automate
Partie moins importante: la programmation supervision.

5.2.3 Xiang

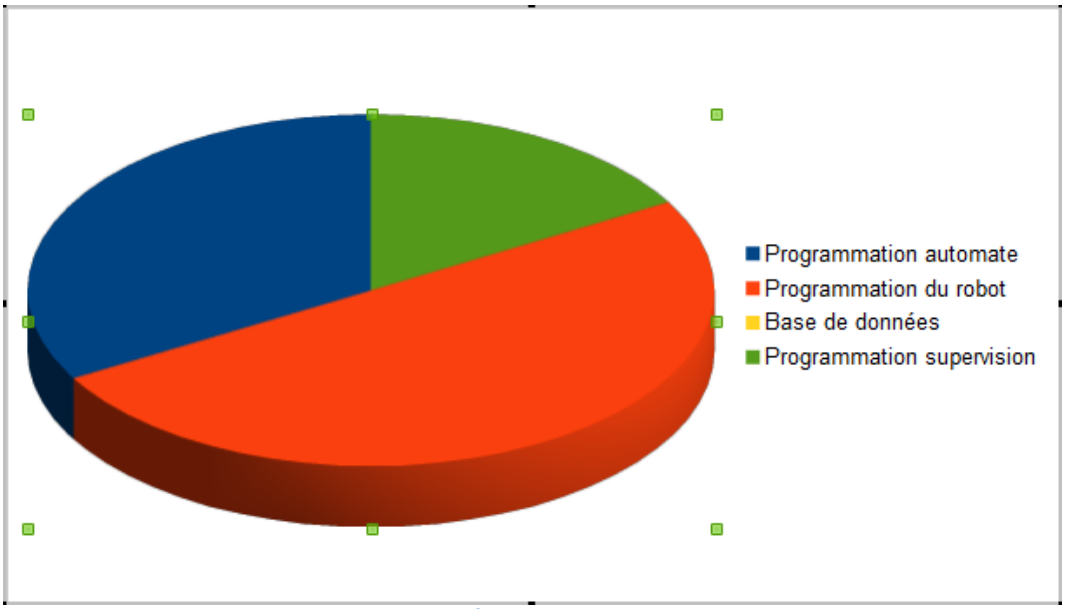


Figure 7 : Diagramme de répartition des tâches accompli par Xiang

Partie dominante: La programmation du robot

Partie moins importante: Programmation automate, Programmation supervision.

6 Travail effectué

Suite à la rédaction du cahier des charges et de la répartition des taches nous nous sommes lancés sur les différents objectifs à accomplir. Afin que cela se fasse correctement, nous faisions un briefing matinal pour nous fixer des objectifs pour la journée et un débriefing en fin de journée afin de valider l'avancement du projet ainsi que sur la concordance des différents mots, bits communs que nous devions utiliser pour la communication entre les différents automates.

6.1 Programmation du convoyeur

La programmation de la ligne transitique a été effectuée en fonction du cahier des charges précédemment expliqué.

6.1.1 La table de données globales

Lorsque plusieurs automates nécessitent une communication entre eux, au sein d'un réseau industriel, une table de données globales doit être définie. Cette table définie les mots qui communiqueront entre les stations et attribue à chaque d'entre elles l'accès en lecture ou en écriture à chacun des mots. Dans notre cas nous avons définis 4 mots en écriture et 8 mots en lecture pour chaque station. Ci-dessous la table de données globales de notre projet :

Données globales du sous-réseau 'Init-pgbase\MPI(1)'

	Identificateur GD	Station epis\ CPU314	Station hippodrome\ CPU314(1)	Station magasin\ CPU314	
1	GD 1.1.1	>MB124:4	MB124:4	MB124:4	
2	GD 2.1.1	MB120:4	>MB120:4	MB120:4	
3	GD 3.1.1	MB116:4	MB116:4	>MB116:4	
4	GD				
5	GD				
6	GD				
7	GD				
8	GD				

Figure 8 : Table de données globales des trois automates (épis, magasin et hippodrome)

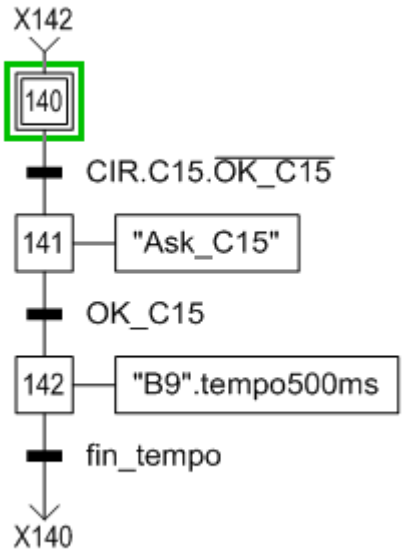
Par exemple, dans notre cas, la station (automate) "épis" pourra écrire du bit 124.0 à 127.7 et lire les bits des autres stations (116.0 à 119.7 pour la station "hippodrome" et 120.0 à123.7 pour la station "magasin"). Le fonctionnement est le même pour les deux autres automates du projet.

6.1.2 Traduction G7 / List

La programmation des automates Siemens se fait en List[1]. Cependant il est très dur de penser un algorithme solide avec ce langage, c'est pourquoi, afin que ce dernier soit le mieux pensé possible, nous avons auparavant élaboré des Grafcet[2].

[1] List : Langage proche de l'automate, a pour avantage de prendre peu de place en mémoire

Suite à cela il est alors plus facile de transcrire les Grafcet en List. Voici en exemple de traduction de Grafcet en List :




```
U    "etape140"
U    "CIR"
U    "C15"
UN   "OK_C15"
R    "etape140"
S    "etape141"

U    "etape141"
U    "OK_C15"
R    "etape141"
S    "etape142"

U    "etape142"
U    "tempo142"
R    "etape142"
S    "etape140"
```

Figure 9 : Grafcet (à gauche) traduit en langage List (à droite)

Ici nous pouvons voir un exemple de traduction faite du langage Grafcet (à gauche) au langage List (à droite). De plus on observe que la gestion des sorties n'est pas la même en List. Il faut respecter l'unicité des sorties, c'est-à-dire qu'une sortie ne peut être appelé qu'une seule et unique fois dans le programme. C'est pourquoi ces dernières sont rassemblées dans un même fichier (FB10). De plus les étapes initiales ne sont pas automatiquement implémentées. Il faut les codées dans le bloc FB1. Voici en exemple du bloc FB10 et FB1 :

[2] Grafcet : Langage particulièrement adapté aux systèmes à évolution séquentielle, c'est-à-dire décomposable en étape

```
O     "etape163"          M30.3
O     "etape164"          M30.4
=     "reponse_c16"       M123.3

O     "etape173"          M31.3
O     "etape174"          M31.4
=     "reponse_c18"       M123.4

U     "etape181"          M32.1
=     "recu_defaut"       M123.5

U     "etape305_1"        M11.5
L     S5T#10S
```

```
S   "etape140"
R   "etape141"
R   "etape142"
```

Figure 10 : Exemple d'un bout de bloc FB10 à gauche et Fb1 à droite

On peut constater que "etape163" ou "etape164" active la sortie (bit commun plus précisément). De plus les temporisations sont également gérées dans ce bloc :

```
U   "etape142"
L   S5T#500MS
SE  "tempo142"
```

Figure 11 : Exemple de gestion d'une temporisation de 500 millisecondes

6.1.3 Les sémaphores

Les sémaphores sont des mécanismes permettant de résoudre des problèmes de synchronisation entre tâches. Dans l'exemple ci-dessous on comprend le mécanisme des sémaphores. Les deux Grafcet représentent une programmation séquentielle de deux automates distincts. Les sémaphores sont faites au niveau des transitions de l'étape 51 sur la station épis et des transitions de l'étape 121 sur la station hippodrome. Avant que chacun des automates ne continue son programme on vérifie que nous avons bien reçu la bonne valeur venant de l'autre automate.

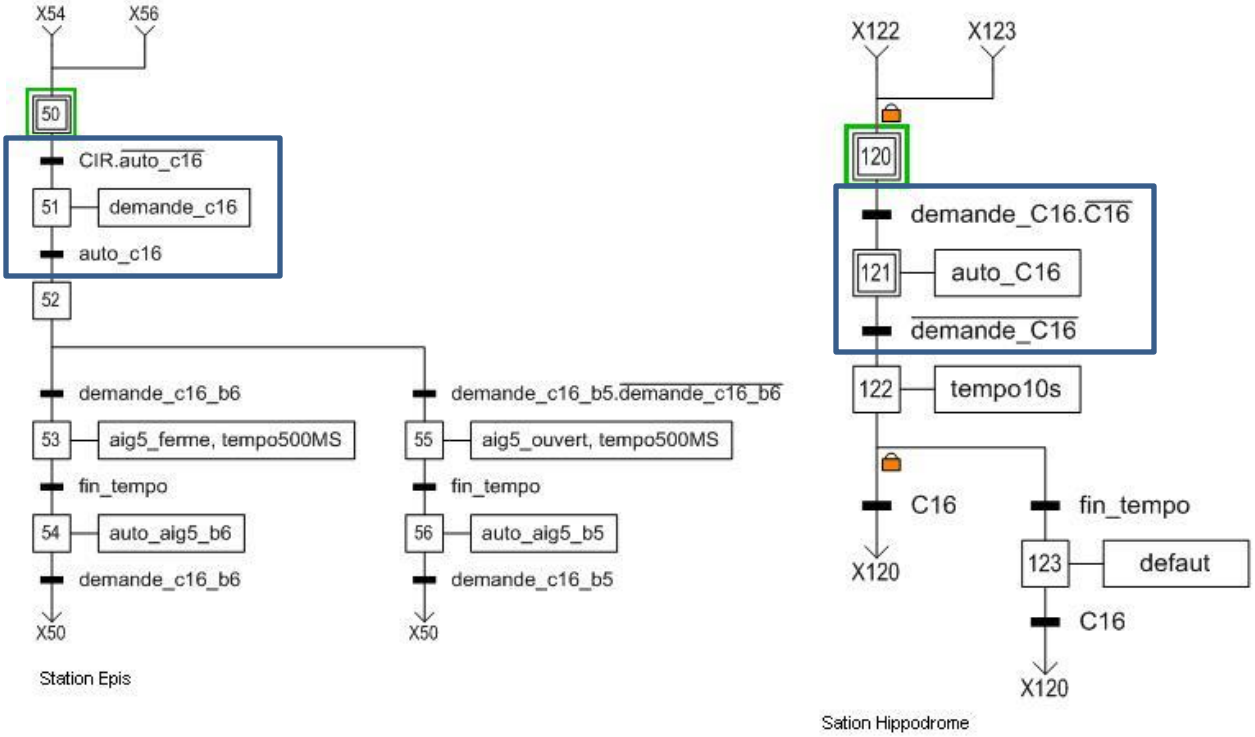


Figure 12 : Exemple de sémaphores

6.1.4 La gestion des défauts

La gestion des défauts est primordiale pour un tel système. En effet nous ne sommes pas à l'abri d'une défaillance technique. Ces dernières sont rapidement apparues au fur et à mesure de la programmation de la ligne transitique. Nous avons remarqué que les palettes restent souvent bloquées lors d'un changement de tapis. Cependant ces changements étant très nombreux nous avons décidé de nous focaliser sur les plus importants, c'est-à-dire sur ceux qui font intervenir un aiguillage.

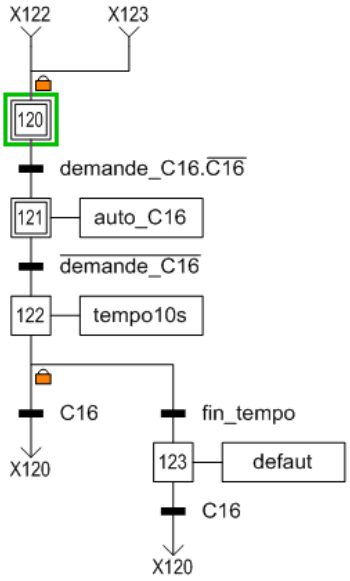


Figure 13 : Exemple de gestion de défaut si la palette met plus de 10 secondes à changer de tapis

Dans cet exemple nous considérons qu'au moment où nous n'avons plus "demande_C16" (moment où la palette commence le virage, ce qui entraîne un changement de tapis) la palette a 10 secondes pour atteindre le capteur C16. Si ce n'est pas le cas alors le programme génère un défaut.

Un voyant lumineux est disponible au niveau de l'alimentation de la ligne transitique. A la détection d'un ou plusieurs défauts ce même voyant est actionné.

6.1.5 Les choix de programmation

6.1.5.1 Partie épis

Les deux épis ont pour objectif de simuler les mêmes opérations de traitements. Ces derniers sont dupliqués afin d'augmenter la cadence de production de la ligne transitique comme nous pourrions le voir dans le monde industriel.

De plus nous avons décidé de ne mettre qu'une seule palette au maximum dans chaque épi. En effet l'objectif étant de simuler un système de convoyage industriel nous avons considéré qu'une seule palette à la fois ne pouvait être envoyé à l'usinage. Si le premier épi est occupé alors la palette est convoyée vers le second. Si le second est également occupé alors la palette refait un tour de l'hippodrome afin de retourner dans l'un des deux épis selon la disponibilité de ces derniers (voir annexe pour le Grafcet de cette station).

Nous avons remarqué que lorsqu'un grand nombre de palettes est envoyé simultanément en production on observe que beaucoup d'entre elles font plusieurs tours de l'hippodrome. Ce dernier étant relativement long nous perdons finalement beaucoup de temps de production à les envoyer à la suite. Nous allons donc gérer cela par la suite de la partie magasin.

6.1.5.2 Partie hippodrome

Cette partie sert exclusivement à aiguiller les pièces vers les épis ou le magasin. Une palette reste dans l'hippodrome uniquement si cette dernière n'a pas subi les opérations de traitement simulées dans les épis (voir annexe pour le Grafcet de cette station).

6.1.5.3 Partie magasin

Cette partie doit gérer principalement le chargement et le déchargement des pièces présentent sur les palettes grâce au robot Staubli (voir annexe pour le Grafcet de cette station).

Comme expliqué précédemment dans la "Partie épi" nous avons observé en élaborant le programme que la vitesse de production pouvait être accéléré en envoyant les palettes en production à des temps précis. La programmation en List ne permettant pas de gérer des timers nous avons décidé de communiquer avec le programme de supervision afin qu'il ordonne quand une palette doit être envoyée.

De plus le nombre de pièces à usiner est entré grâce à la page web de supervision. Lorsque la production est terminée nous demandons au programme de supervision de nous envoyer le nombre de pièces à usiner (valeurs stockées dans une base de données).

```
L     "piece_production"          MB118
L     "cpt_prod"                  Z1
==I
=     "prod_done"                 M15.1

L     "cpt_prod"                  Z1
L     0
==I
=     "production_0"              M15.0
```

Figure 14 : Compteur qui permet de savoir la production est terminé ou non

6.2 Programmation du robot

Le robot Staubli est connecté sur la station magasin, il est donc possible de faire directement les écritures et lectures sur l'automate. La programmation du robot se fait en V++. Le programme se compose en deux parties: un programme main et les programmes qui définissent les mouvements du robot.

Les algorithmes sont les suivants:

Les mouvements du robot:

- Déplacer à la position
- Charger/décharger
- Envoyer un signal à l'automate

Le main:

- Si une palette arrive à l'entrée du magasin et le robot reçoit un signal de chargement alors charger la palette
- Si une palette arrive à la sortie du magasin alors faire le déchargement
- Si le robot est en repos pour une longue durée (15 secondes), remettre le robot à la position initiale.
- Les trois boucles if sont placées dans une boucle while contrôlée par un switch du robot (switch 1012)

- Le chargement est exécuté en priorité sauf si le nombre de palette stockée à la sortie du magasin est supérieur à 4.

Afin de réaliser les timers du temps réel sans bloquant le roulement de la boucle, on utilise un compteur avec un délais de courte durée pour analyser la durée approximativement (voir annexe pour le code du robot).

6.3 Programmation de la supervision

6.3.1 La base de données

La base de données utilisée pour ce projet a été appelé "base_de_données_supervision". Elle est constituée de deux tables : production et supervision_systeme.

```
mysql> show tables
    -> ;
+---------------------------------------+
| Tables_in_base_de_donnees_supervision |
+---------------------------------------+
| production                            |
| supervision_systeme                   |
+---------------------------------------+
2 rows in set (0.09 sec)
```

Figure 15 : Tables présentes dans la base de données

6.3.1.1 La table production

La table production est constituée de:

- Un identifiant en clé primaire qui s'auto-incrémente.
- Un Nombre de pièces (en entier) disponible que l'utilisateur va rentrer lors d'un arrivage directement sur la page web.
- Une colonne Executer (en boolean) pour savoir si l'information a bien été envoyé et donc que les pièces sont bien en production.
- Une date de début (en varchar) pour informer de l'heure de commencement de la production des pièces en stock.
- Une date de fin (en varchar) pour informer de l'heure de fin du cycle de production des pièces en stock.
- Un temps moyen de production (en entier) qui permet de voir le temps que met une pièce à être produite.

```
mysql> select * from production;
+----+----------+---------------------+----------+---------------------+---------
--+------------+
| ID | NBPieces | Date                | Executer | Date_Debut          | Date_Fi
n | Temps_Moy |
+----+----------+---------------------+----------+---------------------+---------
--+------------+
|  1 |        4 | 2015-04-01 17:35:43 |        1 | 01/04/2015 17:39:40 | NULL
  |       NULL |
|  2 |        2 | 2015-04-01 17:40:56 |        0 | NULL                | NULL
  |       NULL |
+----+----------+---------------------+----------+---------------------+---------
--+------------+
2 rows in set (0.00 sec)
```

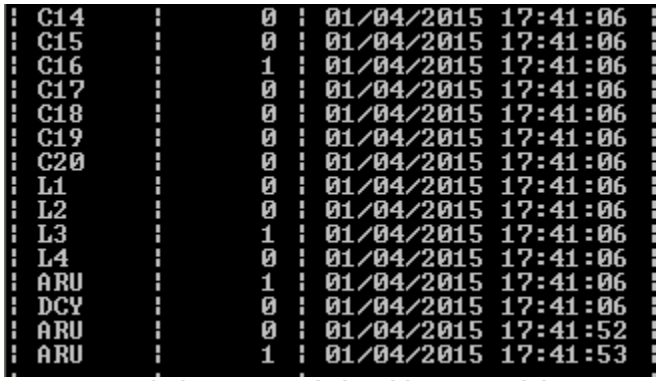
Figure 16 : Extrait de contenu de la table "production"

D'un point de vue industriel, cette table est très importante car elle permet de voir son historique de production et son temps moyen pour usiner une pièce. Grâce aux différentes dates, on peut savoir, en fonction du planning, s'il on est dans les temps ou en retard.

6.3.1.2 La table supervision_systeme

La table production est constituée de:

- Un nom de capteur (varchar) présent sur le système qui a changé d'état.
- L'état du capteur (boolean) en fonction s'il détecte (1) ou non (0) une palette.
- Sa date de changement d'état (varchar).



C14	0	01/04/2015 17:41:06
C15	0	01/04/2015 17:41:06
C16	1	01/04/2015 17:41:06
C17	0	01/04/2015 17:41:06
C18	0	01/04/2015 17:41:06
C19	0	01/04/2015 17:41:06
C20	0	01/04/2015 17:41:06
L1	0	01/04/2015 17:41:06
L2	0	01/04/2015 17:41:06
L3	1	01/04/2015 17:41:06
L4	0	01/04/2015 17:41:06
ARU	1	01/04/2015 17:41:06
DCY	0	01/04/2015 17:41:06
ARU	0	01/04/2015 17:41:52
ARU	1	01/04/2015 17:41:53

Figure 17 : Extrait de contenu de la table "supervision_systeme"

Cette table permet le suivi comme l'historique des différentes détections de palette de chaque capteur. Ça peut être utile pour repérer une panne d'un capteur.

6.3.2 La programmation

6.3.2.1 Explication

La partie programmation de la supervision en langage C permet d'aller lire des données ciblées dans les automates, interagir avec la base de données et d'écrire dans ces automates.

Dans un premier temps, le programme va aller lire les entrées de l'automate (capteurs) et les mettre dans un tableau pour les mémoriser et les historiser. Lors du premier cycle du programme, tous les capteurs vont être insérés dans la table SQL puis lors des prochains cycles va tester si la valeur du capteur a changé. Si le capteur a détecter une palette, sa valeur devient 1. Le programme détecte le changement et va insérer dans la table SQL "supervision_systeme" le ou les capteur(s) qui ont changé d'état avec leurs dates de changement.

Dans un second temps, des sémaphores sont envoyés à l'automate qui gère le Magasin. Une sémaphore pour demander l'autorisation à l'automate d'écrire un nombre de pièces pour alimenter le système et une autre pour écrire dans l'octet choisit au préalable, le nombre de pièces. Ainsi le nombre de pièces est inscrit dans la table SQL "Production".

Le programme informatique gère aussi le temps de production. Il permet de savoir l'heure à laquelle la production a débuté, c'est à dire quand le nombre de pièces est écrite dans l'octet mémoire, et l'heure de fin de production, c'est à dire quand toutes les pièces ont été

produite et revenues au magasin. Pour cela, il fallait faire attention au nombre de lignes dans la base de données. En effet si l'utilisateur insert un nombre de pièce à produire dans la base de donnée (c'est à dire qu'une ligne a été ajouté dans le tableau SQL), il faut que la production commence et aussi gérer le temps de fin de la première production. Pour gérer ça, le programme compte le nombre d'entrées dans le magasin (le nombre de pièces usinées en fin de production). Si le nombre de pièces entrées est égal au nombre de pièce à produire de la bonne ligne dans le tableau SQL Production, alors on affiche la date de fin. Le comptage des pièces entrant se fait par front descendant du capteur.

6.3.2.2 Optimisation

Puisque dans les entreprises, le temps c'est de l'argent, il nous a été demandé d'optimiser le temps de production des pièces. Dès qu'une pièce a fini d'être usinée dans les épis, une autre est prête à rentrer pour perdre le moins de temps de production possible. Il a donc fallut chronométrer le temps que mettait une pièce à sortir du magasin pour aller jusqu'aux deux épis.

Le programme calcule la durée entre deux palettes et envoie à l'automate l'autorisation pour en libérer une du magasin (Cf Annexe 1).

6.3.3 La page internet

La page internet a été conçue pour pouvoir visualiser en temps réel le bon déroulement ou pas de la production. Elle permet entre autre de se libérer des bases de données pas très compréhensible pour les non-initiés et présenté un résumé des données compréhensible pour tout le monde. En effet la page web va lire les valeurs des capteurs et les afficher dans un tableau récapitulatif avec leur date de changement d'état. Un schéma de la chaîne de production à été mis en ligne pour visualiser en temps réel quels capteurs détectent ou pas.

Avec cette page web, nous pouvons aussi écrire dans la base de données, notamment dans la table Production. Elle permet de réinitialiser la table, d'ajouter des pièces pour la production et de visualiser le temps moyen de production d'une pièce.

Cette page internet permet une supervision simple et efficace pour tout le monde.

7 Bilan du projet

7.1 Planning prévisionnel

Nom de la tâche	Durée	Début	Fin	Prédécesseurs	Noms ressources
Choix de solution & construction du planning	**20 jours**	**Ven 19/12/14**	**Ven 16/01/15**		
Etude de faisabilité technique, planning réalisé	4 jours	Ven 19/12/14	Mer 07/01/15	3	Fang XIANG[25%];Jordan DURAND[25%];Mickael AUVIN[25%]
Etude de faisabilité technique, planning réalisé	0 jour	Ven 16/01/15	Ven 16/01/15		Laurent Hardouin
Programmation du projet & tests	**91 jours**	**Mer 10/12/14**	**Mer 15/04/15**		
Programmation automate	26 jours	Mer 10/12/14	Mer 14/01/15		Fang XIANG[50%];Jordan DURAND[50%];Mickael
Programmation robot	10 jours	Jeu 15/01/15	Mer 11/02/15	8	Fang XIANG
Programmation de la supervision	30 jours	Jeu 15/01/15	Mer 11/03/15	8	Mickael AUVIN
Programmation Android	25 jours	Jeu 12/03/15	Mer 15/04/15	10	Mickael AUVIN
Résultats de tests	0 jour	Mer 15/04/15	Mer 15/04/15		Fang XIANG;Jordan DURAND;Mick
Vérification client	**1 jour**	**Mer 22/04/15**	**Mer 22/04/15**		
Vérification tuteur	1 jour	Mer 22/04/15	Mer 22/04/15	12	Fang XIANG;Jordan DURAND;Mick
Validation tuteur	0 jour	Mer 22/04/15	Mer 22/04/15		Laurent Hardouin

Figure 18 : Planning prévisionnel élaboré au début du projet

7.2 Planning final

Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs	Noms ressources
	Choix de solution & construction du planning	**20 jours**	**Ven 19/12/14**	**Ven 16/01/15**		
	Etude de faisabilité technique, planning réalisé	4 jours	Ven 19/12/14	Mer 07/01/15	3	Fang XIANG[25%];Jordan DURAND[25%];Mickael AUVIN[25%]
	Etude de faisabilité technique, planning réalisé	0 jour	Ven 16/01/15	Ven 16/01/15		Laurent Hardouin
	Programmation du projet & tests	**91 jours**	**Mer 10/12/14**	**Mer 15/04/15**		
	Programmation automate	81 jours	Mer 10/12/14	Mer 01/04/15		Fang XIANG[25%];Jordan DURAND[50%];Mickael
	Programmation robot	10 jours	Jeu 15/01/15	Mer 11/02/15	8	Fang XIANG[75%]
	Programmation de la supervision	55 jours	Jeu 15/01/15	Mer 15/04/15	8	Mickael AUVIN[75%]
	Programmation Android	0 jour	Mer 15/04/15	Mer 15/04/15	10	Mickael AUVIN
	Résultats de tests	0 jour	Mer 15/04/15	Mer 15/04/15		Fang XIANG;Jordan DURAND;Mick
	Vérification client	**1 jour**	**Mer 22/04/15**	**Mer 22/04/15**		
	Vérification tuteur	1 jour	Mer 22/04/15	Mer 22/04/15	12	Fang XIANG;Jordan DURAND;Mick
	Validation tuteur	0 jour	Mer 22/04/15	Mer 22/04/15		Laurent Hardouin

Figure 19 : Planning final du projet

7.3 Bilan final

La partie supervision et programmation automate a été plus longue que prévu. Du coup nous avons manqué de temps pour la dernière phase de notre projet qui était de créer une application Android. Néanmoins la page internet que nous avons créé remplace cette application.

Nous avons réservé aussi pas mal de temps les derniers jours à la rédaction de ce document ce qui a influencé notre avancement dans le projet.

8 Conclusion

Ce projet, qui nous a occupé six mois, 1 jour par semaine, nous a permis de nous servir et d'appliquer des compétences acquises durant nos années d'études précédentes mais aussi d'apprendre de nouvelles méthodes de travail notamment en équipe. De plus l’autonomie laissée dans le projet est valorisant pour nous, futurs ingénieurs, à apprendre à prendre des décisions et gérer notre temps. Nous avons également acquis de nombreuses connaissances sur la programmation d’un robot industrielle, la gestion d’un système transitique et la régulation de la communication entre tous les systèmes grâce à la partie informatique industrielle.

Nous avons pu mettre en place une solution qui répond en grande partie aux attentes du cahier des charges.

9 Annexe

9.1 Annexe 1 : code programmation

```
#include <winsock2.h>
#include <stdio.h>
#include <string.h>
#include "Applicom.h"
#include <winsock.h>
#include <MYSQL/mysql.h>
#include <time.h>
#include <semaphore.h>
#include <sys/types.h>
#include <pthread.h>
#include <unistd.h>
#include <stdlib.h>

#define MAX_SEM_COUNT 10
#define THREADCOUNT 12

HANDLE ghSemaphore;
DWORD WINAPI ThreadProc( LPVOID );
BYTE nbyte;
BYTE pbyte;

sem_t sem_NbPieces;
sem_t sem_Autorisation;
sem_t sem_autorisation_sortie;
int autorisation = 0;
int Accord=0;
int NbPieces=0;
int *nbpieces;
short int  status;
short int neq;
short int neq1;
short int nb;
short int nb1;
short int adr;
short int adr1;
short int nchan;
short int id = 0;
short int nbligne=0;
short int nblignep = 0;
char requete[500]="";

int valeurint = 0;
int valeurint2 = 0;

void finish_with_error(MYSQL *con)
```

```
{
	fprintf(stderr,"%s\n",mysql_error(con));
	mysql_close(con);
	exit(1);
}

void *demauto(void *arg)		//semaphore demande autorisation
{
	printf("demande autorisation\n");
	neq = 5;    //Le numéro d'équipement
	nb = 1;		// Nombre de donnees à lire
	adr= 116;   // Adresse de la première variable
	nchan = 0;  // Numero du canal

	initbus(&status);
	readpackbyte(&nchan, &neq, &nb, &adr, &nbyte, &status); // Va lire l'octet n° 116 de
l'equipement 5 (Magasin)

	if (!status)	//Si la lecture c'est bien passé
	{
	autorisation = (nbyte & 32) >>5;
	printf("autorisation = %d\n",autorisation); //Affichage du résultat de la lecture
	}
	else	//sinon
	{
	printf("Probleme sur l'equipement %d",neq);
	}

	if(autorisation >0)
	{
	Accord = 1;
	}
	else
	{
	Accord = 0;
	}
	sem_post(&sem_Autorisation);
	return NULL;
}

void *production(void *arg) //Semaphore Production
{
	printf("production\n");
	short int tablprod[64];
	int result;


	initbus(&status);
	nchan=0;
	neq=5;
	nb=1;
```

```
	adr=118;

	tablprod[0] = valeurint;
	writepackbyte(&nchan,&neq, &nb, &adr, tablprod, &status);	//Ecriture du nombre de pièces dans l'octet 118 du magasin
	printf("Nombre enregistre \n");


	nchan=0;
	neq=5;
	nb=1;
	adr=90;

	tablprod[0] = 1;
	writepackbyte(&nchan,&neq, &nb, &adr, tablprod, &status);	//Ecriture de la confirmation de l'envoie du nombre de pièce dans l'octet 90 du magasin
	printf("envoi production = 1\n");
	if(!status)
	{
	printf("Proleme d'ecriture sur l'equipement %d\n",neq);
	}
	else
	{
	printf("L'ecriture s'est bien faite");
	}

	valeurint = 0;
	tablprod[0] = valeurint;
	writepackbyte(&nchan,&neq, &nb, &adr, tablprod, &status);	//Remise à 0 de la confirmation
	printf("envoi production = 0\n");


	if(!status)
	{
	printf("Proleme d'ecriture sur l'equipement %d\n",neq);
	}
	else
	{
	printf("L'ecriture s'est bien faite");
	}

	sem_post(&sem_NbPieces);
	return NULL;
}

void *autorisation_sortie(void *arg)	//Semaphore permettant de donner l'autorisation aux palettes de sortir du magasin
{
	printf("autorisation sortie\n");
	short int tablauto[64];
```

```
	initbus(&status);
	nchan=0;
	neq=5;
	nb=1;
	adr=50;

	tablauto[0] = 1;
	writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);    //Ecriture de l'autorisation dans l'octet 50 du magasin
	printf("\nenvoi autorisation = 1\n");
	if(!status)
	{
	printf("Proleme d'ecriture sur l'equipement %d\n",neq);
	}
	else
	{
	printf("L'ecriture s'est bien faite\n");
	}

	tablauto[0] = 0;
	writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status); //Remise à 0 de l'autorisation de sortie des palettes
	printf("envoi autorisation = 0\n");
	if(!status)
	{
	printf("Proleme d'ecriture sur l'equipement %d\n",neq);
	}
	else
	{
	printf("L'ecriture s'est bien faite");
	}

	sem_post(&sem_autorisation_sortie);
	return NULL;
}

int main(int argc, char **argv)
{
	char nom[32][5] = {"C1","C2","C3","C4","C5","C6","C7","C8","C9","C10","C11","C12","C13","C14","C15","C16","C17","C18","C19","C20","L1","L2","L3","L4","ARU","DCY"};
	nchan = 0;   /* Numéro de canal */
	neq = 4;          /* Numéro d'équipement */
	neq1 = 4;
	nb = 64;          /* Nombre de variables *///32+32 dans une seule API
	nb1 = 64;
	/* Status */
	long adr = 3;    /* Adresse de la première variable */
	long adr1 = 3;
	short int tabl[64];         /* Table recevant les données */
```

```
        short int tablbit[64];    /* Table contenant les valeurs éclatées */
        short int tableau[64];
        short int tableau_choix[64];
        short int c=0;
        short int  Tabmemo[32];
        short int tableau_NB_Pieces[32];
        int i,init;
        int debut = 1;
        int fin = 0;
        int num = 0;
        short int envoi_prod =0;
        short int tableau_autorisation[20];

        short int acceptation = 0;

        double diff_t;   // calcul de la différence de temps entre de type time_t
        double diff_t2;
        double diff_t3;
        double diff_t4;

        time_t start_t;  //date de début de production de pieces
        time_t end_t;              //date de fin de production de pièces
        time_t end_t_piece;
        time_t start_t_piece;
        time_t T_epis1 = time(0);
        time_t T_epis2;

        int Presence_piece=0;
        int cpt_nb_pieces = 0;
        int terminer = 0;
        int cpt_NbpiecesProd = 0;
        int presence_piece_epis1 = 0;
        int presence_piece_epis2 = 0;

//semaphore
        HANDLE aThread[THREADCOUNT];
        DWORD ThreadID;



//-------
        for (init =0; init < 32; init++)    //initialisation de tableaux
        {
        Tabmemo[init]=0;
        tableau[init]=0;
        tableau_NB_Pieces[init]=0;
        }

        MYSQL *mysql = mysql_init(NULL); //Initialisation de MySql
```

```
	mysql_options(mysql,MYSQL_READ_DEFAULT_GROUP,
"MYSQL_READ_DEFAULT_GROUP");	//Option de connexion


	if(mysql_real_connect(mysql, "localhost", "root","root", "base_de_donnees_supervision",
0, NULL,0)) //Test de la connexion avec la base de données : base_de_donnees_supervision
	{
	do
	{

	MYSQL_RES *result = NULL;
	MYSQL_ROW row = NULL;

	unsigned int a = 0;
	unsigned int num_champs = 0;
	initbus(&status);

	for (i=1; i<=26; i++)
	{
		tableau[i]=Tabmemo[i];
	}

	nbligne = 0;
	sprintf(requete,"SELECT * From Production");	//Créer une requete SQL visant la
table Production qui selectionne toutes les données de la table
	mysql_query(mysql,requete);			//Permet d'éxecuter la requete SQL
	result = mysql_use_result(mysql);		//Récupération du résultat dans la variable
result de type MYSQL_RES *

	//On récupere le nombre de champs
	num_champs = mysql_num_fields(result);
	unsigned char * valeur=malloc(64);
	while((row = mysql_fetch_row(result)))	//Test du nombre de ligne dans la table
Production
	{
		nbligne++;
	}

	//--------------------

	if((nbligne!=0) && (nblignep!=nbligne))	// Si la table n'est pas vide et que l'on ne
traite pas la derniere ligne de la table
	{
		nblignep = nbligne;
		sprintf(requete,"SELECT ID From Production WHERE Executer = 0 ORDER BY
Date ASC LIMIT 1");  //Requete qui selectionne le premier Identifiant où la colonne Executer = 0
de la table Production

		mysql_query(mysql,requete);
		result = mysql_use_result(mysql);
```

```
                //On récupere le nombre de champs
                num_champs = mysql_num_fields(result);
                unsigned char * valeur2=malloc(64);
                while((row = mysql_fetch_row(result)))
                {
                // Pointeur pour stocker la taille des valeurs
                unsigned long *lenghts;

                // Stockage de la taille dans le pointeur
                lenghts = mysql_fetch_lengths(result);

                //Pour obtenir les valeurs de chaque champs
                for (a=0; a<num_champs; a++)
                {
                sprintf(valeur2, "%s", row[0] ? row[a] : "NULL");
                }
                printf("\n");
                id = atoi(valeur2); //Récupération de la valeur de l'identifiant
                num = id;
                }

        }
        //---------------------

        //Lire la date et l'heure actuelle
        time_t now = time(NULL);
        struct tm tm_now =*localtime(&now);
        char date_now[sizeof "JJ/MM/AAAA HH:MM:SS"];
        strftime(date_now,sizeof date_now, "%d/%m/%Y %H:%M:%S", &tm_now);
        //---------------------- Station magasin --------------------------
        // CAPTEUR
        neq = 5;    //Le numéro d'équipement
        nb = 64;
        adr= 0;             //--> e0.0 --> C1(e0.1) --> Adresse de la première variable

        readpackibit(   &nchan,&neq,&nb,&adr, tabl,&status); //Lire les valeurs des 64 premiers
bits du magasin

        //(n° canal,n° equipement,nb bit a lire,adr 1er bit a lire,tableau destination,status
echange)
        if (!status)
        {
                transwordbit( &nb,tabl,tablbit, &status);
                //(nb bit a eclater, tab contenant les entiers a eclater, tableau recevant le resultat,
status de l'échange)
        }
        else
        {
                printf("Problème sur l'équipement %d",neq);
        }
```

```
	if(Tabmemo[2] == 1)
	{
		if(tablbit[2] == 0)
		{
		cpt_nb_pieces++;
		Presence_piece--;

		}
	}

	if(Tabmemo[3] == 1)
	{
		if(tablbit[3] == 0)
		{
		Presence_piece++;
		}
	}
	printf("\n cpt = %d \n" ,cpt_nb_pieces);
	printf("\n Presence piece = %d \n" ,Presence_piece);

	//Tabmemo[0]=tablbit[0];
	Tabmemo[1]=tablbit[1];  //C1
	Tabmemo[2]=tablbit[2];  //C2
	Tabmemo[3]=tablbit[3];  //C3
	Tabmemo[4]=tablbit[4];  //C4
	Tabmemo[6]=tablbit[6];  //C6
	Tabmemo[7]=tablbit[7];  //C7
	Tabmemo[14]=tablbit[12];//C14
	Tabmemo[15]=tablbit[13];//C15
	Tabmemo[17]=tablbit[15];//C17
	Tabmemo[21]=tablbit[0]; //L1


	//---------------------- Station Hippodrome ---------------------------
	// CAPTEUR
	neq = 4;    //Le numéro d'équipement

	readpackibit(  &nchan,&neq,&nb,&adr, tabl,&status); //Lire les valeurs des 64 premiers bits de l'hippodrome

	//(n° canal,n° equipement,nb bit a lire,adr 1er bit a lire,tableau destination,status echange)
	if (!status)
	{
		transwordbit( &nb,tabl,tablbit, &status);
		//(nb bit a eclater, tab contenant les entiers a eclater, tableau recevant le resultat, status de l'échange)
	}
	else
	{
		printf("Problème sur l'équipement %d",neq);
```

```
        }
        //printf(" l'equipement %d \n  C13:%d C16:%d C18:%d C5:%d C8:%d L2:%d L3:%d L4:%d DCY:%d ARU:%d\n\n", neq,tablbit[5],tablbit[6],tablbit[11],tablbit[3],tablbit[2],tablbit[7],tablbit[8],tablbit[10],tablbit[1],tablbit[0]);

        //Mémorisation de l'état des capteurs
        Tabmemo[13]=tablbit[5]; //C13
        Tabmemo[16]=tablbit[6]; //C16
        Tabmemo[18]=tablbit[11]; //C18
        Tabmemo[5]=tablbit[3];  //C5
        Tabmemo[8]=tablbit[2];  //C8
        Tabmemo[22]=tablbit[7]; //L2
        Tabmemo[23]=tablbit[8]; //L3
        Tabmemo[24]=tablbit[10];//L4
        Tabmemo[25]=tablbit[0]; //ARU
        Tabmemo[26]=tablbit[1]; //DCY

        //---------------------- Station épis ----------------------------
        // CAPTEUR
        neq = 6;    //Le numéro d'équipement

        readpackibit(  &nchan,&neq,&nb,&adr, tabl,&status); //Lire les valeurs des 64 premiers bits des épis

        //(n° canal,n° equipement,nb bit a lire,adr 1er bit a lire,tableau destination,status echange)
        if (!status)
        {
                transwordbit( &nb,tabl,tablbit, &status);
                //(nb bit a eclater, tab contenant les entiers a eclater, tableau recevant le resultat, status de l'échange)
        }
        else
        {
                printf("Problème sur l'équipement %d",neq);
        }
        //printf(" l'equipement %d \n  C9:%d C10:%d C11:%d C12:%d C19:%d C20:%d\n\n" , neq,tablbit[0],tablbit[1],tablbit[2],tablbit[3],tablbit[4],tablbit[5]);

        Tabmemo[9]=tablbit[0];  //C9
        Tabmemo[10]=tablbit[1]; //C10
        Tabmemo[11]=tablbit[2]; //C11
        Tabmemo[12]=tablbit[3]; //C12
        Tabmemo[19]=tablbit[4]; //C19
        Tabmemo[20]=tablbit[5]; //C20

        for (c=0; c<26; c++)   //Pour le nombre de capteurs
        {
                if ((tableau[c]!=Tabmemo[c])|| debut == 1)  //Si c'est le premier cycle du programme ou si la valeur du capteur à changé entre deux cycles
```

```
		{
		sprintf(requete,"INSERT INTO supervision_systeme VALUES('%s','%d','%s')",nom[c-1],Tabmemo[c],date_now); //Creer un requete qui écrit le nom du capteur, ca valeur (0->ne detecte pas, 1->detecte), et la date du changement d'état
		mysql_query(mysql,requete); //Execution de la requete
		}
	}
	if((Tabmemo[8] == 1) && debut == 1)
	{
		presence_piece_epis1 = 1;
	}



//-----------------------------------------SEMAPHORE-----------------------------------------------------------

	int resultat_requete = 0;
//int sem_init(sem_t *sem, int pshared, unsigned int value)
//The sem_init() function is used to initialise the unnamed semaphore referred to by sem. The value of the initialised semaphore is value.
	sem_init(&sem_Autorisation,0,0);

	//Création de threads
	pthread_t tid_d;
	pthread_create(&tid_d, NULL, demauto, NULL);	//Création de la semaphore de demande autorisation

//The sem_wait() function locks the semaphore referenced by sem by performing a semaphore lock operation on that semaphore.
	sem_wait(&sem_Autorisation);
	printf("autorisation demande\n");



	if(Accord == 1 && id<=nbligne && nbligne!=0)	// Si nous avons l'accord de l'automate pour écrire un nombre de pièce, que la table contienne des données et que l'identifiant corresponde à une ligne
	{
		sprintf(requete,"SELECT NbPieces From Production WHERE ID = %d",id);
	//Créer une requete pour récupérer la cellule où est présent le nombre de pièces de l'identifiant donné
		mysql_query(mysql,requete);
		result = mysql_use_result(mysql);

		//On récupere le nombre de champs
		num_champs = mysql_num_fields(result);
		unsigned char * valeur=malloc(64);
		while((row = mysql_fetch_row(result)))
		{
		// Pointeur pour stocker la taille des valeurs
		unsigned long *lenghts;
```

```
                // Stockage de la taille dans le pointeur
                lenghts = mysql_fetch_lengths(result);

                //Pour obtenir les valeurs de chaque champs
                for (a=0; a<num_champs; a++)
                {
                sprintf(valeur, "%s", row[0] ? row[a] : "NULL");
                }
                printf("\n");
                valeurint = atoi(valeur);   //Récupération de la valeur du nombre de pieces
                valeurint2 = valeurint;

                }
                tableau_NB_Pieces[id] = valeurint2; //Copie du nombre de piece dans un tableau en fonction de l'identifiant de la table Production

                cpt_NbpiecesProd++;
                sem_init(&sem_NbPieces,0,0);
                pthread_t tid_p;
                pthread_create(&tid_p,NULL,production,NULL);      //Création de la sempahore Production
                sem_wait(&sem_NbPieces);
                sprintf(requete,"UPDATE Production SET Executer = 1 WHERE ID = %d",id); // Modification de la cellule Executer à 1 où id de la ligne est donnée
                mysql_query(mysql,requete);
                //if(Tabmemo[3]==0 || debut == 1)
                //{
                // if(tableau[3]==1)
                //{
                if(cpt_NbpiecesProd<=2)
                {
                        printf("autorisation sortie\n"); //Partie sur l'envoi d'autorisation pour la sortie de pieces
                        short int tablauto[64];

                        initbus(&status);
                        nchan=0;
                        neq=5;
                        nb=1;
                        adr=20;

                        tablauto[0] = 1;
                        writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);     //Ecrit sur l'octet n°20 l'autorisation
                        printf("\nenvoi autorisation = 1\n");
                        if(!status)
                        {
                        printf("Proleme d'ecriture sur l'equipement %d\n",neq);
                        }
                        else
```

```
                        {
                        printf("L'ecriture s'est bien faite\n");
                        }

                        tablauto[0] = 0;
                        writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);    //Réinitialise
l'octet d'autorisation à 0
                        printf("envoi autorisation = 0\n");
                        if(!status)
                        {
                        printf("Proleme d'ecriture sur l'equipement %d\n",neq);
                        }
                        else
                        {
                        printf("L'ecriture s'est bien faite");
                        }
                        /*sem_init(&sem_autorisation_sortie,0,0);
                        pthread_t tid_as;
                        pthread_create(&tid_as,NULL,autorisation_sortie,NULL);
                        sem_wait(&autorisation_sortie);*/
                }
                else
                {
                        end_t_piece = time(NULL);
                        diff_t3 = difftime(end_t_piece, T_epis1);
                        if((diff_t3>=10))          //Si la différence de temps entre les 2 mesures est
égale ou suppérieur à 10 on autorise l'envoie d'une palette
                        {
                        printf("autorisation sortie\n");
                        short int tablauto[64];

                        initbus(&status);
                        nchan=0;
                        neq=5;
                        nb=1;
                        adr=20;

                        tablauto[0] = 1;
                        writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
                        printf("\nenvoi autorisation = 1\n");
                        if(!status)
                        {
                        printf("Proleme d'ecriture sur l'equipement %d\n",neq);
                        }
                        else
                        {
                        printf("L'ecriture s'est bien faite\n");
                        }

                        tablauto[0] = 0;
                        writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
```

```
printf("envoi autorisation = 0\n");
if(!status)
{
printf("Proleme d'ecriture sur l'equipement %d\n",neq);
}
else
{
printf("L'ecriture s'est bien faite");
}

T_epis1 = time(NULL);

/*sem_init(&sem_autorisation_sortie,0,0);
pthread_t tid_as;
pthread_create(&tid_as,NULL,autorisation_sortie,NULL);
sem_wait(&autorisation_sortie);*/
}

diff_t4 = difftime(end_t_piece, T_epis2);
if((diff_t4>=10)&&diff_t3<10)
{
printf("autorisation sortie\n");
short int tablauto[64];

initbus(&status);
nchan=0;
neq=5;
nb=1;
adr=20;

tablauto[0] = 1;
writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
printf("\nenvoi autorisation = 1\n");
if(!status)
{
printf("Proleme d'ecriture sur l'equipement %d\n",neq);
}
else
{
printf("L'ecriture s'est bien faite\n");
}

tablauto[0] = 0;
writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
printf("envoi autorisation = 0\n");
if(!status)
{
printf("Proleme d'ecriture sur l'equipement %d\n",neq);
}
else
{
```

```
                printf("L'ecriture s'est bien faite");
                }
                T_epis2 = time(NULL);
                /*sem_init(&sem_autorisation_sortie,0,0);
                pthread_t tid_as;
                pthread_create(&tid_as,NULL,autorisation_sortie,NULL);
                sem_wait(&autorisation_sortie);*/
                }

                if(presence_piece_epis1 == 1)
                {
                presence_piece_epis1 = 0;
                }
                else
                {
                presence_piece_epis1 = 1;
                }
                if(presence_piece_epis2 == 1)
                {
                presence_piece_epis2 = 0;
                }
                else
                {
                presence_piece_epis2 = 1;
                }
        //  }
        //}
        }


        start_t = time(NULL);
        struct tm tm_now =*localtime(&start_t);
        char date_now[sizeof "JJ/MM/AAAA HH:MM:SS"];
        strftime(date_now,sizeof date_now, "%d/%m/%Y %H:%M:%S", &tm_now);

        sprintf(requete,"UPDATE Production SET Date_Debut = '%s' WHERE ID =
'%d'",date_now,id); //Créer une requete pour modifier la cellule du tableau Production
        mysql_query(mysql,requete);

        if(id>1 && (tableau_NB_Pieces[num] == cpt_nb_pieces))   // Si l'id pointe sur une
ligne de la table et que le nombre de pieces en production est égale au nombre de pieces rentré
dans le magasin
        {
        num++;  //incrémente num
        sprintf(requete,"UPDATE Production SET Date_Fin = '%s' WHERE Date_Fin is
NULL ORDER BY Date ASC LIMIT 1",date_now);
        //On modifie la date de fin de la ligne de production qui vient de se terminer
        mysql_query(mysql,requete);
        end_t = time(NULL);
        struct tm tm_now =*localtime(&end_t);
        char date_now[sizeof "JJ/MM/AAAA HH:MM:SS"];
```

```
                strftime(date_now,sizeof date_now, "%d/%m/%Y %H:%M:%S", &tm_now);
                diff_t = difftime(end_t, start_t);
                diff_t2 = diff_t/valeurint2;
                sprintf(requete,"UPDATE Production SET Temps_Moy = '%g' WHERE
Temps_Moy is NULL ORDER BY Date ASC LIMIT 1",diff_t2);
                //on calcule et modifie le Temps moyen de production d'une piece
                mysql_query(mysql,requete);
                cpt_nb_pieces = 0;
                terminer = 1;
                cpt_NbpiecesProd--;
                }
                fin = 1;


                printf("Nombre Pieces ecrite\n");

                id++;

        }
        else if(Accord ==0 && id!=0)  //Si on a plus ou pas d'accord de l'automate pour écrire
        {
                printf("attends permission\n");
                //if(Tabmemo[3]==0 || debut == 1)
                //{

                if(cpt_NbpiecesProd<2)  //On traite les pieces qui sont déja en production
                {
                        cpt_NbpiecesProd++;
                        printf("autorisation sortie\n");
                        short int tablauto[64];

                        initbus(&status);
                        nchan=0;
                        neq=5;
                        nb=1;
                        adr=20;

                        tablauto[0] = 1;
                        writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
                        printf("\nenvoi autorisation = 1\n");
                        if(!status)
                        {
                        printf("Proleme d'ecriture sur l'equipement %d\n",neq);
                        }
                        else
                        {
                        printf("L'ecriture s'est bien faite\n");
                        }

                        tablauto[0] = 0;
                        writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
```

```
		printf("envoi autorisation = 0\n");
		if(!status)
		{
		printf("Proleme d'ecriture sur l'equipement %d\n",neq);
		}
		else
		{
		printf("L'ecriture s'est bien faite");
		}
		/*sem_init(&sem_autorisation_sortie,0,0);
		pthread_t tid_as;
		pthread_create(&tid_as,NULL,autorisation_sortie,NULL);
		sem_wait(&autorisation_sortie);*/
}
else
{
		end_t_piece = time(NULL);
		diff_t3 = difftime(end_t_piece, T_epis1);
		if((diff_t3>=10)&presence_piece_epis1)
		{
		printf("autorisation sortie\n");
		short int tablauto[64];

		initbus(&status);
		nchan=0;
		neq=5;
		nb=1;
		adr=20;

		tablauto[0] = 1;
		writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
		printf("\nenvoi autorisation = 1\n");
		if(!status)
		{
		printf("Proleme d'ecriture sur l'equipement %d\n",neq);
		}
		else
		{
		printf("L'ecriture s'est bien faite\n");
		}

		tablauto[0] = 0;
		writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
		printf("envoi autorisation = 0\n");
		if(!status)
		{
		printf("Proleme d'ecriture sur l'equipement %d\n",neq);
		}
		else
		{
		printf("L'ecriture s'est bien faite");
```

```
}

T_epis1 = time(NULL);
T_epis2 = time(NULL);
/*
sem_init(&sem_autorisation_sortie,0,0);
pthread_t tid_as;
pthread_create(&tid_as,NULL,autorisation_sortie,NULL);
sem_wait(&autorisation_sortie);*/
}

diff_t3 = difftime(end_t_piece, T_epis2);
if((diff_t3>=13)&presence_piece_epis2)
{
printf("autorisation sortie\n");
short int tablauto[64];

initbus(&status);
nchan=0;
neq=5;
nb=1;
adr=20;

tablauto[0] = 1;
writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
printf("\nenvoi autorisation = 1\n");
if(!status)
{
printf("Proleme d'ecriture sur l'equipement %d\n",neq);
}
else
{
printf("L'ecriture s'est bien faite\n");
}

tablauto[0] = 0;
writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
printf("envoi autorisation = 0\n");
if(!status)
{
printf("Proleme d'ecriture sur l'equipement %d\n",neq);
}
else
{
printf("L'ecriture s'est bien faite");
}
T_epis2 = time(NULL);
T_epis1 = time(NULL);
/*sem_init(&sem_autorisation_sortie,0,0);
pthread_t tid_as;
pthread_create(&tid_as,NULL,autorisation_sortie,NULL);
```

```
                        sem_wait(&autorisation_sortie);*/
                        }

                        if(presence_piece_epis1 == 1)
                        {
                        presence_piece_epis1 = 0;
                        }
                        else
                        {
                        presence_piece_epis1 = 1;
                        }
                        if(presence_piece_epis2 == 1)
                        {
                        presence_piece_epis2 = 0;
                        }
                        else
                        {
                        presence_piece_epis2 = 1;
                        }
                }
                // }
                //}
        }
        else    // si on a un accord d'écriture mais qu'aucune pièce n'est disponible en stock on traite la date de fin et le temps moyen + les pieces qui sont déja en production
        {
                printf("Veuillez remplir les stocks!!!");

                if((tableau_NB_Pieces[num] == cpt_nb_pieces) && id!=0)
                {
                num++;
                end_t = time(NULL);
                struct tm tm_now =*localtime(&end_t);
                char date_now[sizeof "JJ/MM/AAAA HH:MM:SS"];
                strftime(date_now,sizeof date_now, "%d/%m/%Y %H:%M:%S", &tm_now);
                sprintf(requete,"UPDATE Production SET Date_Fin = '%s' WHERE Date_Fin is NULL ORDER BY Date ASC LIMIT 1",date_now);
                mysql_query(mysql,requete);

                diff_t = difftime(end_t, start_t);
                diff_t2 = diff_t/valeurint2;
                sprintf(requete,"UPDATE Production SET Temps_Moy = '%g' WHERE Temps_Moy is NULL ORDER BY Date ASC LIMIT 1",diff_t2);
                mysql_query(mysql,requete);
                cpt_nb_pieces = 0;
                fin = 0;
                cpt_NbpiecesProd--;

                }
```

```
end_t_piece = time(NULL);
diff_t3 = difftime(end_t_piece, T_epis1);
if((diff_t3>=10)&presence_piece_epis1)
{
printf("autorisation sortie\n");
short int tablauto[64];

initbus(&status);
nchan=0;
neq=5;
nb=1;
adr=20;

tablauto[0] = 1;
writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
printf("\nenvoi autorisation = 1\n");
if(!status)
{
printf("Proleme d'ecriture sur l'equipement %d\n",neq);
}
else
{
printf("L'ecriture s'est bien faite\n");
}

tablauto[0] = 0;
writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
printf("envoi autorisation = 0\n");
if(!status)
{
printf("Proleme d'ecriture sur l'equipement %d\n",neq);
}
else
{
printf("L'ecriture s'est bien faite");
}

T_epis1 = time(NULL);
T_epis2 = time(NULL);
/*sem_init(&sem_autorisation_sortie,0,0);
pthread_t tid_as;
pthread_create(&tid_as,NULL,autorisation_sortie,NULL);
sem_wait(&autorisation_sortie);*/
}

diff_t3 = difftime(end_t_piece, T_epis2);
if((diff_t3>=13)&presence_piece_epis2)
{
printf("autorisation sortie\n");
short int tablauto[64];
```

```
initbus(&status);
nchan=0;
neq=5;
nb=1;
adr=20;

tablauto[0] = 1;
writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
printf("\nenvoi autorisation = 1\n");
if(!status)
{
printf("Proleme d'ecriture sur l'equipement %d\n",neq);
}
else
{
printf("L'ecriture s'est bien faite\n");
}

tablauto[0] = 0;
writepackbyte(&nchan,&neq, &nb, &adr, tablauto, &status);
printf("envoi autorisation = 0\n");
if(!status)
{
printf("Proleme d'ecriture sur l'equipement %d\n",neq);
}
else
{
printf("L'ecriture s'est bien faite");
}
T_epis2 = time(NULL);
T_epis1 = time(NULL);
/*sem_init(&sem_autorisation_sortie,0,0);
pthread_t tid_as;
pthread_create(&tid_as,NULL,autorisation_sortie,NULL);
sem_wait(&autorisation_sortie);*/
}

if(presence_piece_epis1 == 1)
{
presence_piece_epis1 = 0;
}
else
{
presence_piece_epis1 = 1;
}
if(presence_piece_epis2 == 1)
{
presence_piece_epis2 = 0;
}
else
```

```
                {
                presence_piece_epis2 = 1;
                }
        }
        mysql_free_result(result);
        debut = 0;

        }
        while(1);
        }
        else
        {
        printf("erreur lors de la connexion a la base de donnee");
        }
        return 0;
}
```

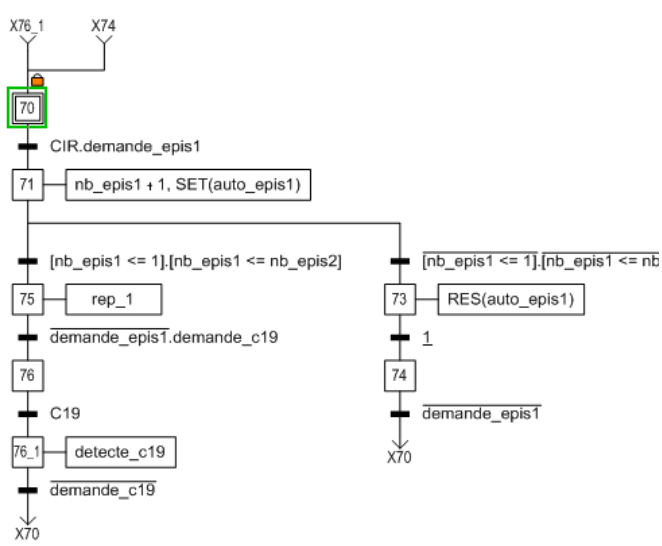
9.2 Annexe 2 : programmes du robot:

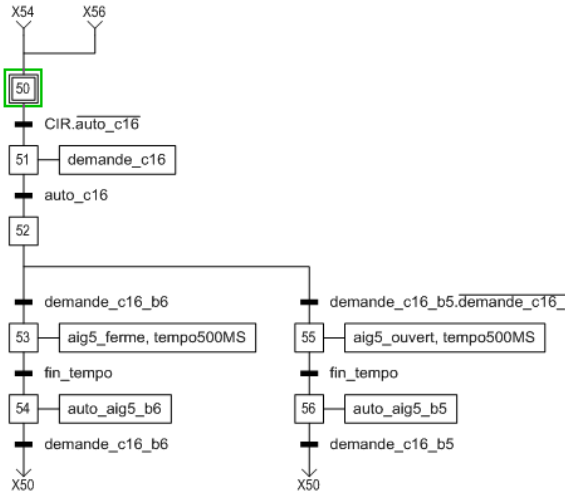
```
.PROGRAM main.pg()          //programme main du robot
        a = 0               //compteur repos
        b = 0               //compteur priorité
        READY               //mettre en position initiale
        BREAK               //attendre jusqu'à la fin du mouvement

        WHILE SIG(1012) DO          //tant que le switch 1012 est actif
           DELAY 2                  //un délai de 2 seconds

           IF SIG(1003) THEN        //si le nombre de palette
                                    //stockée à la sortie du magasin est
                                    //supérieur à 4
                a = 0               //reset compteur repos
                b = 2               //déactive la priorité
           END

           IF SIG(1001) AND SIG(-1002) THEN     //s'il y a demande de déchargement
                                                //et pas de demande de chargement
                b = b+1                         //le compteur augmente chaque tourne
                IF b == 2 THEN                  //2 tournes effectués = timer de 4 seconds
                   CALL sb1.pg()                //faire déchargement
                   BREAK
                   b = 0                        //reset les compteurs
                   a = 0
                END
           END

           IF SIG(1002) AND SIG(-1003) THEN     //s'il y a demande de chargement et pas de
                CALL sb2.pg()                   //faire le chargement
                BREAK
                a = 0                           //reset compteur repos
           END
```

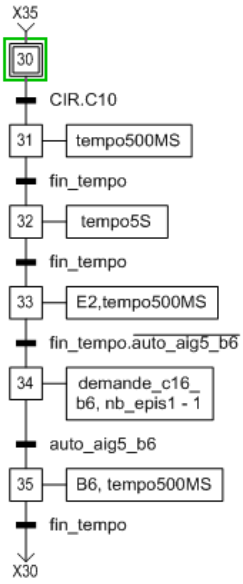
```
            IF SIG(-1001) AND SIG(-1002) THEN      //s'il n'y aucun demande
                a = a+1
                DELAY 0.5                  //délai de 0.5 second
                IF a == 10 THEN            //10 boucles effectués = timer de 25 seconde
                   READY                   //remettre a la position initiale
                   BREAK
                   a = 0                   //reset compteur repos
                END
        END
.END

.PROGRAM sb1.pg()                  //programme du déchargement
        MOVE #b1                   //déplacer à la sortie du magasin
        BREAK
        OPENI                      //ouvrir la pince
        BREAK
        CLOSEI                     //fermer la pince
        BREAK
        SIGNAL 1                   //envoyer 1 à la sortie 1
        DELAY 1                    //délai de 1 second
        SIGNAL -1                  //envoyer 0 à la sortie 1
.END

.PROGRAM sb2.pg()                  //programme du chargement
        MOVE #b2                   //déplacer à l'entrée du magasin
        BREAK
        OPENI                      //ouvrir la pince
        BREAK
        CLOSEI                     //fermer la pince
        BREAK
        SIGNAL 2                   //envoyer 1 à la sortie 2
        DELAY 1
        SIGNAL -2                  //envoyer 1 à la sortie 2
.END
```

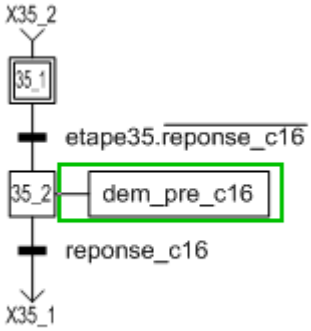
9.3 Annexe 3 : Grafcet

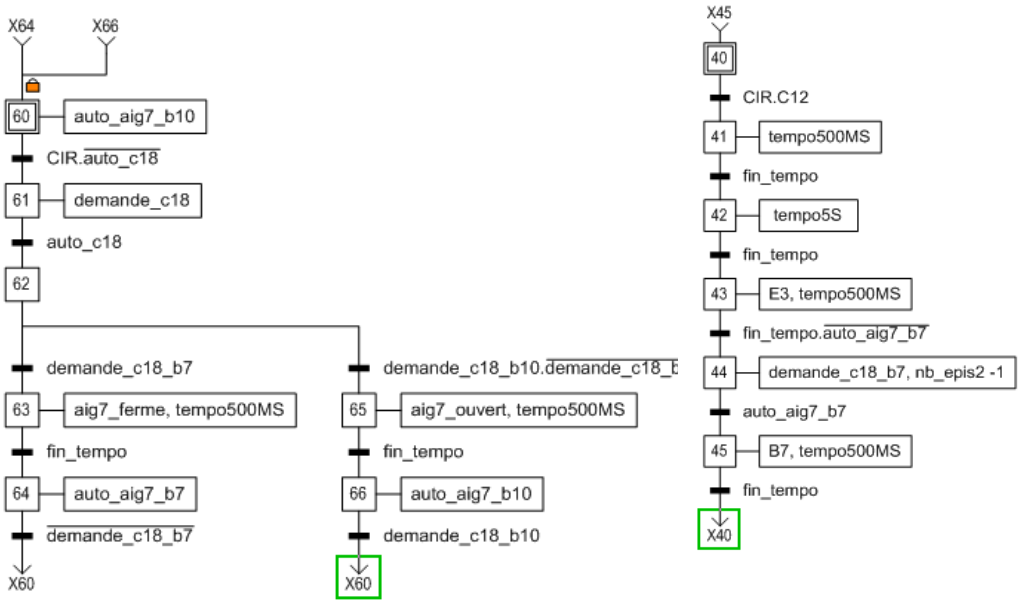
9.3.1 Station épis

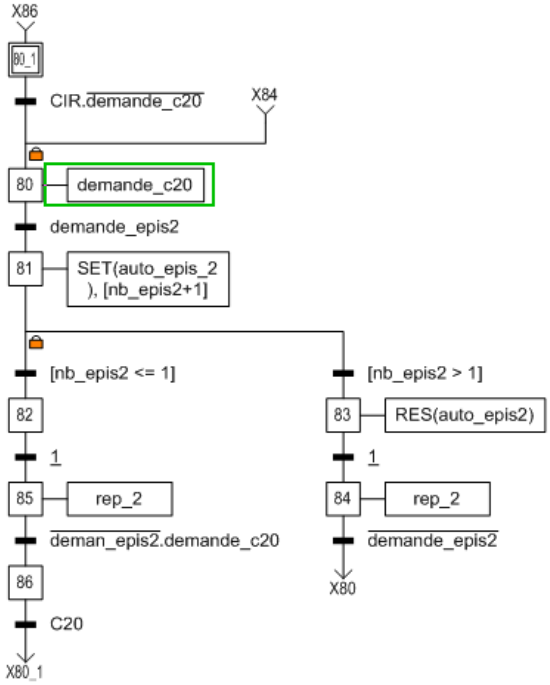




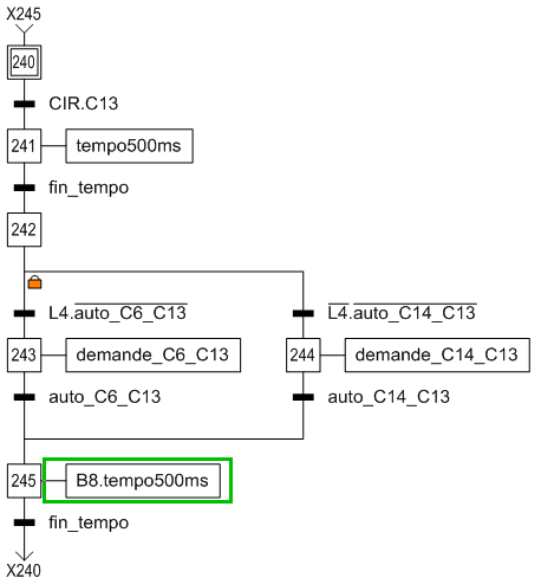


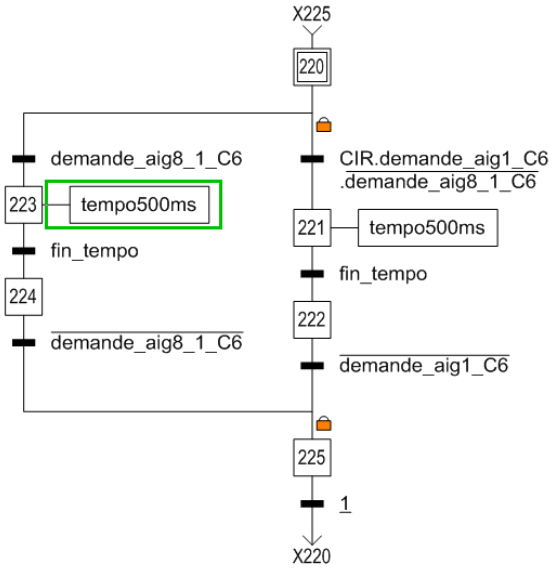


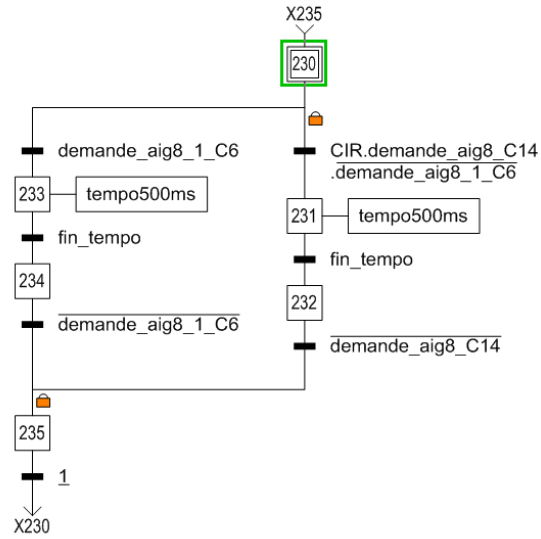

X64
X66
60
auto_aig7_b10
CIR.auto_c18
61
demande_c18
auto_c18
62
demande_c18_b7
63
aig7_ferme, tempo500MS
fin_tempo
64
auto_aig7_b7
demande_c18_b7
X60
demande_c18_b10.demande_c18_b
65
aig7_ouvert, tempo500MS
fin_tempo
66
auto_aig7_b10
demande_c18_b10
X60
X45
40
CIR.C12
41
tempo500MS
fin_tempo
42
tempo5S
fin_tempo
43
E3, tempo500MS
fin_tempo.auto_aig7_b7
44
demande_c18_b7, nb_epis2 -1
auto_aig7_b7
45
B7, tempo500MS
fin_tempo
X40

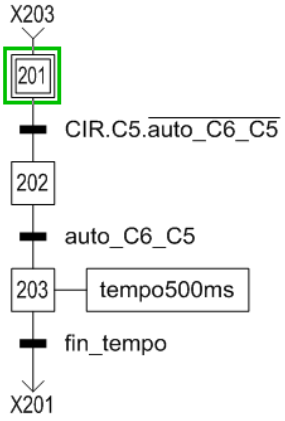

X86
80_1
CIR.demande_c20
X84
80
demande_c20
demande_epis2
81
SET(auto_epis_2
), [nb_epis2+1]
[nb_epis2 <= 1]
82
1
85
rep_2
deman_epis2.demande_c20
86
C20
X80_1
[nb_epis2 > 1]
83
RES(auto_epis2)
1
84
rep_2
demande_epis2
X80

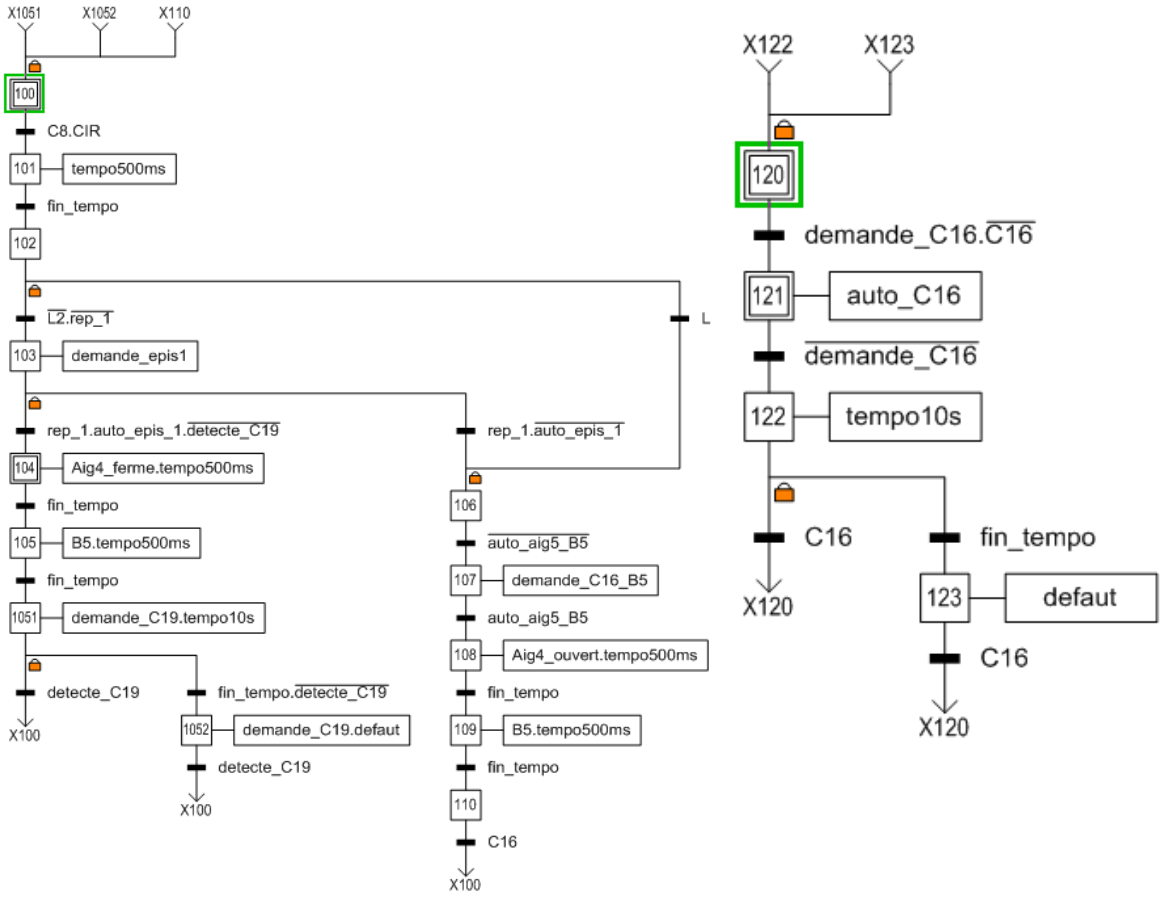
9.3.2 Station hippodrome

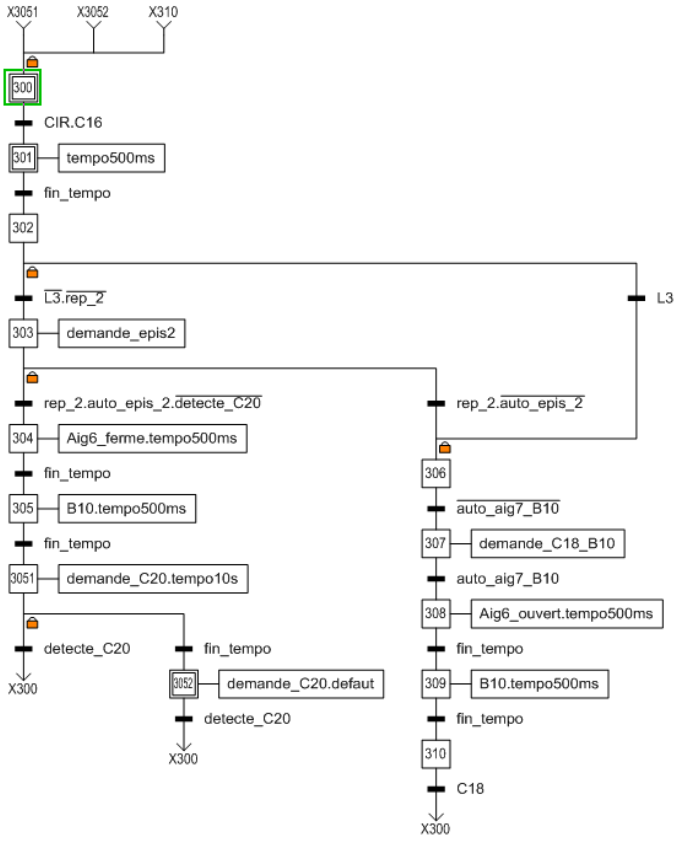


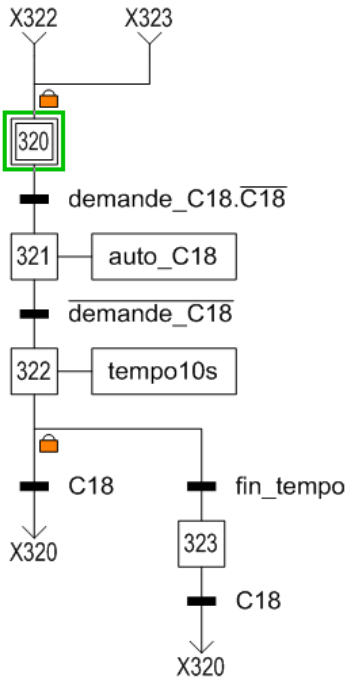


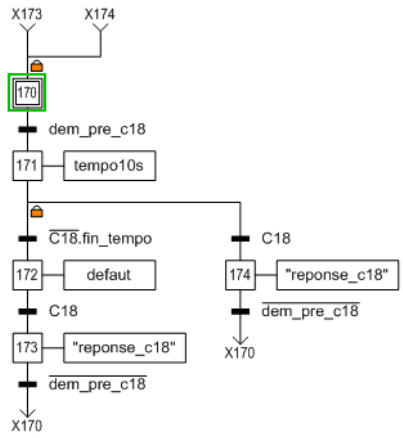


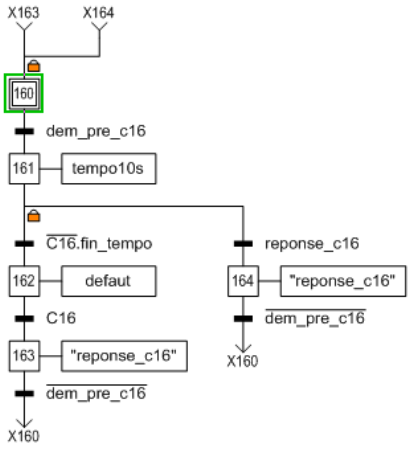


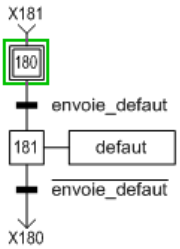

X1051
X1052
X110
100
C8.CIR
101
tempo500ms
fin_tempo
102
L2.rep_1
L
103
demande_epis1
rep_1.auto_epis_1.detecte_C19
rep_1.auto_epis_1
104
Aig4_ferme.tempo500ms
fin_tempo
105
B5.tempo500ms
fin_tempo
1051
demande_C19.tempo10s
detecte_C19
fin_tempo.detecte_C19
X100
1052
demande_C19.defaut
detecte_C19
X100
106
auto_aig5_B5
107
demande_C16_B5
auto_aig5_B5
108
Aig4_ouvert.tempo500ms
fin_tempo
109
B5.tempo500ms
fin_tempo
110
C16
X100
X122
X123
120
demande_C16.C16
121
auto_C16
demande_C16
122
tempo10s
C16
fin_tempo
X120
123
defaut
C16
X120


X3051
X3052
X310
300
CIR.C16
301
tempo500ms
fin_tempo
302
L3.rep_2
303
demande_epis2
rep_2.auto_epis_2.detecte_C20
rep_2.auto_epis_2
304
Aig6_ferme.tempo500ms
fin_tempo
305
B10.tempo500ms
fin_tempo
3051
demande_C20.tempo10s
detecte_C20
fin_tempo
3052
demande_C20.defaut
detecte_C20
X300
L3
306
auto_aig7_B10
307
demande_C18_B10
auto_aig7_B10
308
Aig6_ouvert.tempo500ms
fin_tempo
309
B10.tempo500ms
fin_tempo
310
C18
X300

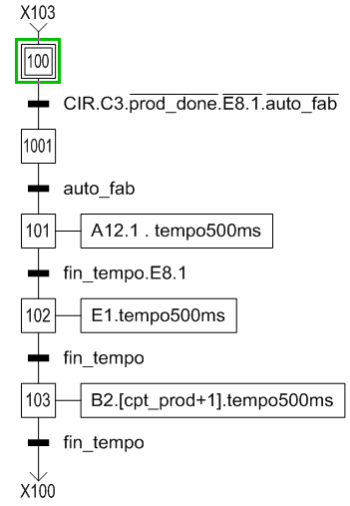

X322
X323
320
demande_C18.C18
321
auto_C18
demande_C18
322
tempo10s
C18
fin_tempo
X320
323
C18
X320

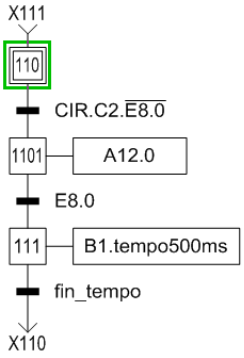

X173
X174
170
dem_pre_c18
171
tempo10s
C18.fin_tempo
C18
172
defaut
174
"reponse_c18"
C18
dem_pre_c18
173
"reponse_c18"
dem_pre_c18
X170
X170

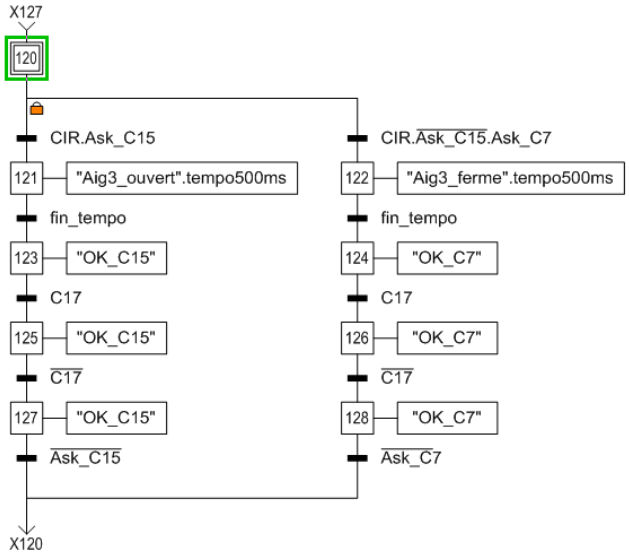

X163
X164
160
dem_pre_c16
161
tempo10s
C16.fin_tempo
reponse_c16
162
defaut
164
"reponse_c16"
C16
dem_pre_c16
163
"reponse_c16"
dem_pre_c16
X160
X160

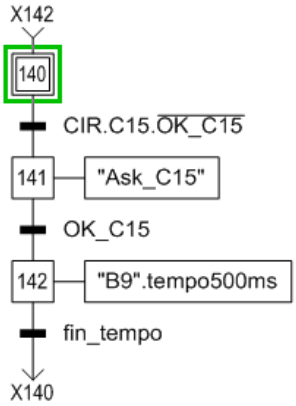

X181
180
envoie_defaut
181
defaut
envoie_defaut
X180

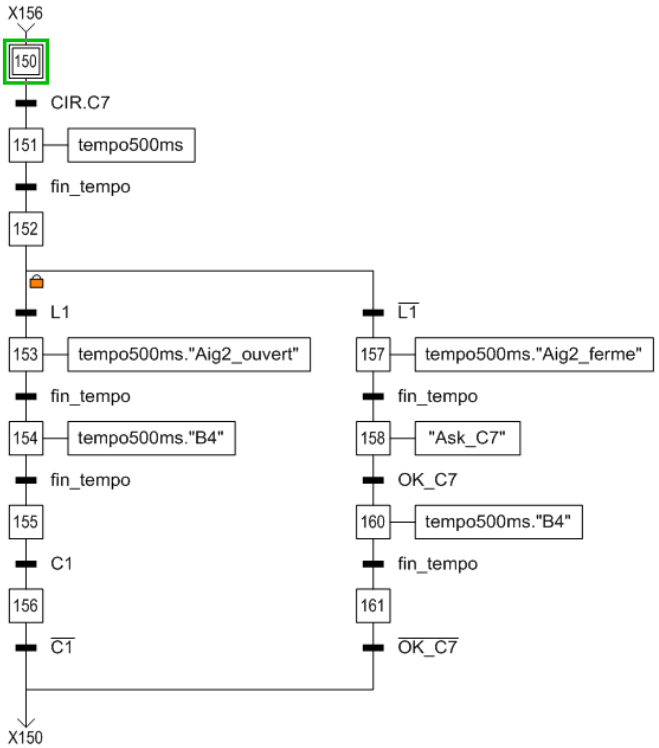
9.3.3 Station magasin

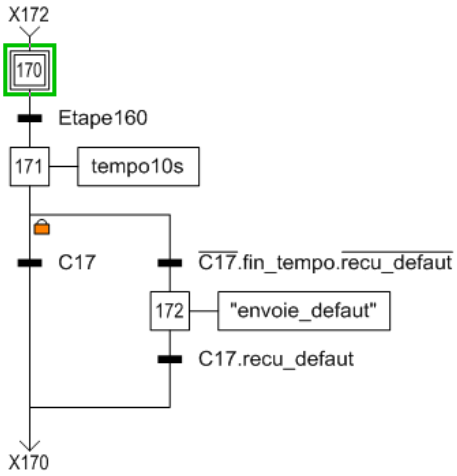


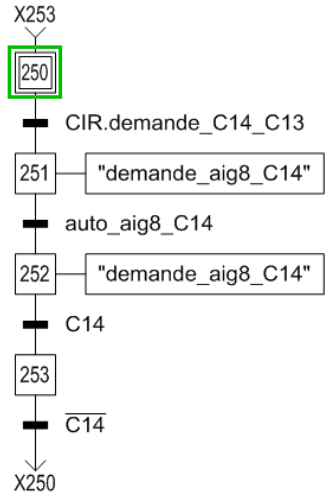


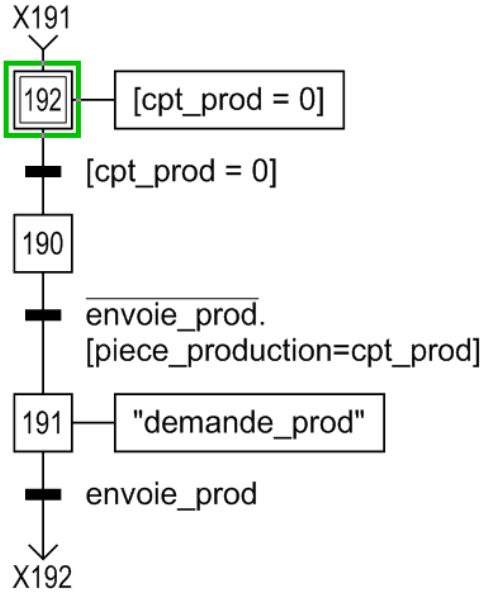


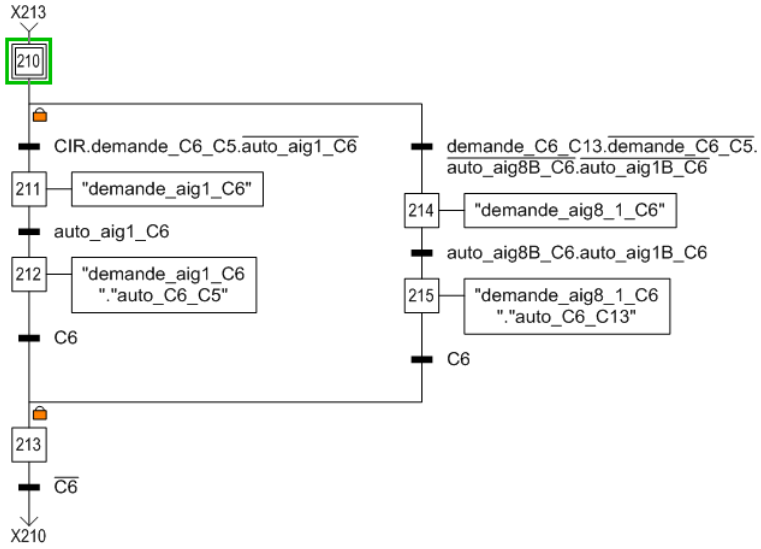



X156
150
CIR.C7
151
tempo500ms
fin_tempo
152
L1
153
tempo500ms."Aig2_ouvert"
fin_tempo
154
tempo500ms."B4"
fin_tempo
155
C1
156
C1
L1
157
tempo500ms."Aig2_ferme"
fin_tempo
158
"Ask_C7"
OK_C7
160
tempo500ms."B4"
fin_tempo
161
OK_C7
X150


X172
170
Etape160
171
tempo10s
C17
C17.fin_tempo.recu_defaut
172
"envoie_defaut"
C17.recu_defaut
X170


X253
250
CIR.demande_C14_C13
251
"demande_aig8_C14"
auto_aig8_C14
252
"demande_aig8_C14"
C14
253
C14
X250


X191
192
[cpt_prod = 0]
[cpt_prod = 0]
190
envoie_prod.
[piece_production=cpt_prod]
191
"demande_prod"
envoie_prod
X192


X213
210
CIR.demande_C6_C5.auto_aig1_C6
211
"demande_aig1_C6"
auto_aig1_C6
212
"demande_aig1_C6
"."auto_C6_C5"
C6
demande_C6_C13.demande_C6_C5.
auto_aig8B_C6.auto_aig1B_C6
214
"demande_aig8_1_C6"
auto_aig8B_C6.auto_aig1B_C6
215
"demande_aig8_1_C6
"."auto_C6_C13"
C6
213
C6
X210

Titre : Robotisation d'un système transitique

RESUME

Ce projet tutoré a été effectué en EI4 option AGI, d'une durée de 6 mois par 3 étudiants de l'école d'ingénieur de l'ISTIA.

Notre mission consistait à réaliser l'automatisation d'une ligne transitique avec la robotisation et la supervision pour simuler une ligne de production dans une entreprise qui fabrique des pièces à la chaîne. L'alimentation en pièce se fait par le robot Staubli et la supervision permet de vérifier en temps réel que tout est en ordre de marche.

Ce projet nous a permis d'améliorer nos compétences et de travailler en équipe sur un système qui l'on peut rencontrer dans notre possible futur entreprise. Ce fut une très bonne expérience et un projet que je conseillerai.

MOTS-CLES

Transitique, Automatisation, Robotisation, Supervision, GRAFCET, langage C, langage V+, sémaphores.

ABSTRACT

This tutored project was made in EI4 option AGI, with a duration of 6 months by 3 students of the engineering school of the ISTIA.

Our mission consisted in realizing automation of a transitique line with robotization and supervision to feign a production line in a company which makes products in the chain. The supply of products is made by the robot Staubli and supervision allows to verify in real time that everything is in working order.

This project allowed us to improve our skills and to work in team on a system which we can meet in our possible future company. It was a very good experience and a project which I shall recommend.