ROBOTISATION DE LA LIGNE TRANSITIQUE

RAPPORT DE PROJET EI4-SAGI

2017-2018

Réalisé par : Pierre Baugé – Mathilde Paris – Thibault Poulhalec

Encadré par : Laurent Hardouin

IstiA ÉCOLE D'INGÉNIEURS
UNIVERSITÉ D'ANGERS

REMERCIEMENTS

Remerciements

Nous tenions, tous les trois, à remercier M. Laurent Hardouin pour ses conseils et son écoute tout au long de nos séances de projet et pour nous avoir proposé un projet très complet qui nous a permis d'enrichir nos connaissances. Nous remercions également M. Franck Mercier et M. Hassan Bouljroufi pour leurs aides techniques sur la maquette. Nous remercions également M. Rémy Guyonneau, pour son encadrement dans le cours de Réseaux Industriels ce qui nous a permis d'avoir un programme automate bien avancé.

Table des matières

INTRODUCTION

Introduction

Dans le cadre de notre projet de quatrième année d'école d'ingénieur en spécialité Systèmes Automatisés et Génie Informatique nous avons travaillés sur un projet durant 84 heures lors de 11 journées.

Nous avons choisi le projet de la ligne transitique. Notre mission a été de programmer entièrement cette ligne qui se trouve en salle 216 à l'Istia. Ce projet a permis de concrétiser plusieurs notions que nous avions vues en cours, mais nous n'avions pas encore eu d'exemples aussi concrets que lors de ce projet.

Nous avions déjà travaillé sur la maquette en Réseaux industriels, les TP nous ont permis d'appréhender le fonctionnement de la maquette. Grâce à ce TP, nous avons commencé ce projet avec un programme automatisé déjà bien avancé puisqu'il ne restait que quelques dysfonctionnements à corrigé.

Nous allons vous présenter les différentes étapes de notre projet. Nous avons commencé par l'automatisation de la ligne, puis nous avons découvert la robotisation : une notion que nous n'avions pas encore vue à l'école. Nous avons continué avec la supervision de la maquette, qui représente un aboutissement de tout le travail en amont. Cela nous a montré un exemple concret de l'utilisation d'un programme C. De plus, nous avons vu comment échanger des informations entre des automates, un programme C, une basse de données et un site Web.

Présentation du projet

1. LE CONTEXTE

Dans le cadre de notre projet EI4, nous avons eu l'occasion de programmer la ligne transitique située en salle 216 à l'ISTIA. Ce projet a pour but d'automatiser et de superviser le système de transport de palettes. De plus, il faut programmer le robot Staubli RX90 afin qu'il fasse des actions en corrélation avec les ordres envoyés par les automates.

Pour améliorer le système programmé en TP de réseaux industriels, nous devons ajouter un système de défauts qui indique lorsqu'une palette est bloquée sur le convoyeur. Pour indiquer ce défaut aux opérateurs présents près de la ligne, nous allons utiliser le voyant défaut présent sur le pupitre.

Pour la partie supervision, nous devons récupérer et stocker tous ces défauts ainsi que l'état des capteurs dans une base de données grâce à un programme en C. Grâce à ces données, on pourra ainsi calculer le nombre de palettes produites, la performance de la ligne (nombre de palettes par heure), le temps moyen entre les défauts et leur temps de résolution. Ces données seront accessibles en temps réel grâce à une IHM. De cette IHM, nous pourrons également envoyer des ordres de production.

Pour la partie robotique, nous devons programmer le robot pour qu'il accède à l'un des trois postes de travail de la ligne transitique suivant l'ordre qu'il a reçu de l'automate.

2. LA MAQUETTE

Cette ligne transitique est composée de trois zones contrôlées chacune par un automate différent : un magasin un hippodrome ainsi que deux épis. Les liens entre les différentes zones sont gérés grâce à des butées et des aiguillages.

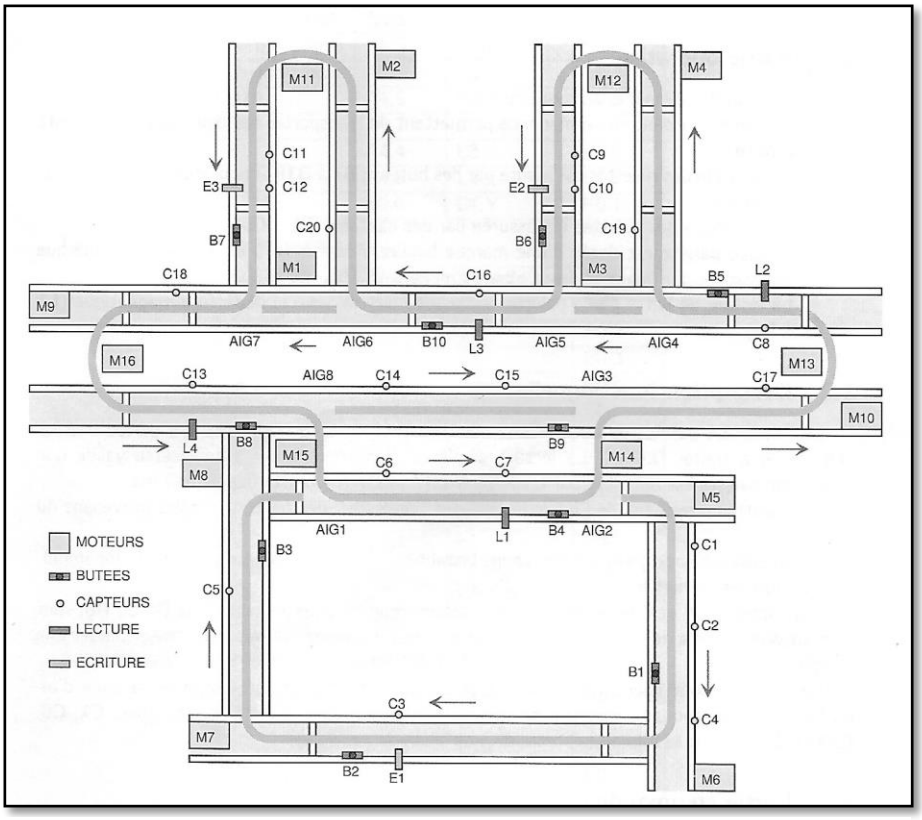


Figure 1 : Schémas de la ligne transitique

1. Zone Magasin

Dans le magasin, il y a trois postes de travail où le robot intervient. Ces postes sont marqués par la présence d'une butée qui empêche les palettes d'avancer tant qu'elles n'ont pas été traitées. Au niveau du deuxième poste de travail, il y a un également un module d'écriture qui permet d'indiquer que la palette a bien été traitée dans cette zone. Nous avons imaginé un poste de perçage (butée B1), un poste de rivetage (butée B2) ainsi qu'un poste de soudure (butée B3). Chaque poste a une capacité de stockage de palettes, au poste de perçage, nous pouvons accumuler quatre palettes et aux postes de rivetage et de soudure, nous pouvons en accumuler cinq.

2. Zone Hippodrome

Cette zone permet l'aiguillage des palettes aussi bien ver les épis que vers le magasin. Dans l'hippodrome une lecture de l'état de la palette est faite. Si une palette n'a pas été traitée, elle doit aller dans le magasin. Dans le cas contraire, elle est dirigée vers les épis.

3. Zone Épis

Les palettes sont envoyées dans ces parties pour qu'un opérateur récupère la pièce finie et remette de la matière première sur la palette. Une fois ceci fait, un module d'écriture efface le fait que la palette a été traitée précédemment dans la zone du magasin. Un module de lecture est présent entre les deux épis pour voir si la palette a été traitée ou non. Il ne peut y avoir que deux palettes simultanément dans un épi, donc lorsque les épis sont pleins, les palettes déjà traitées doivent continuer de circuler sur l'hippodrome.

3. LE PLANNING

Ce projet a été découpé une séance de quatre heures et 10 séances de huit heures. Lors de ces séances, nous avons réparti les différentes tâches de la façon suivante.

ACTIVITÉ	DÉBUT	DURÉE	Thibault	Mathilde	Pierre
Finalisation des programmes automates	1	1	X	X	X
Diagnostique des pannes sur la ligne/commande des pieces	1	2	X	X	X
Instalation des pièces (Réparation des pannes)	10	1			X
Détections des defauts	2	3		X	
Prise en main du robot	3	1	X		X
Creation et programmation d'un sénariode test pour le robot	4	2	X		X
Programation finale du robot	5	2	X		X
Adaptation du programme automate	5	2		X	
Supervision Programmation C	6	6			X
Supervision Programation PHP	7	3	X	X	
Rédaction du rapport	9	3	X	X	X
Création du site	10	2	X		

Figure 3 : Diagramme de Gantt

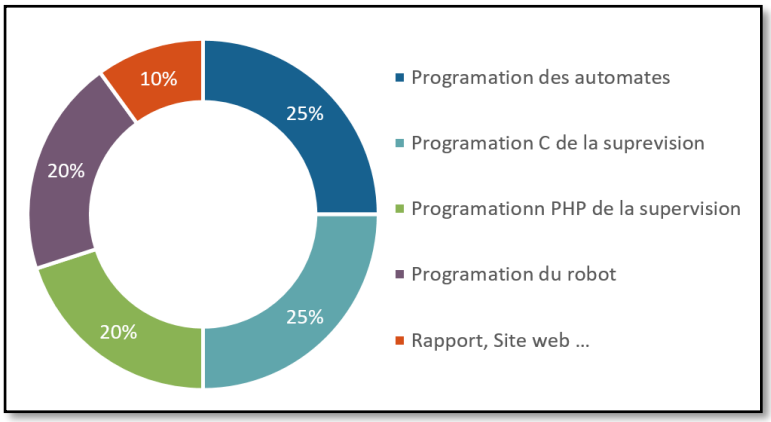


Figure 2 : Part des activités dans le projet

Ce projet comportait plusieurs grandes parties. Ainsi, nous retrouvons dans le diagramme ci-dessous la part des activités principales de ce le projet.

Automatisation

1. MATERIEL

Pour la programmation, nous avons pu utiliser un ordinateur sur lequel est installé Simatic Manager STEP 7 et nous avons codé en List. Au début, nous ne maitrisions pas bien ce langage, nous avons du bien le comprendre et l'appréhender. L'inconvénient principal de cette configuration est que l'interface n'est pas très pratique et du fait du langage la relecture pour trouver nos erreurs est plutôt compliquée. Pour parer ce problème, nous avons systématiquement fait des grafcets sur papier pour corriger nos éventuelles erreurs avec plus de facilité.

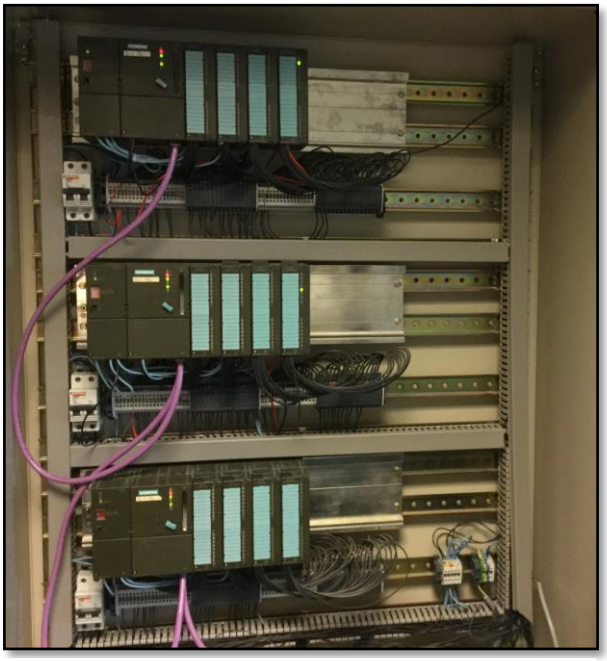


Figure 4 : Automates

2. PROGRAMMATION

L'essentiel de la programmation de la ligne transitique a été fait en TP de réseaux, nous avons repris le programme sur lequel nous avions travaillé et nous l'avons retravaillé pour qu'il corresponde au cahier des charges que nous nous étions fixé. De plus, nous avons revu certaines parties du code, pour permettre une évolutivité plus aisée pour la suite de notre projet. Par exemple, dans notre programme de base, la programmation des butées B1 et B2 étant en série nous avons préféré, dissocié les deux actions. Ce qui nous a fait gagner du temps quand nous avons programmé le robot. En effet, avec le robot les actions de B1 et B2 ne sont plus cycliques, mais dépendent du nombre de palettes en attente et de la politique de priorité.

3. ORGANISATION DES PROGRAMMES

L'ensemble des capteurs et actionneurs de la ligne sont dispatchés en trois zones "hippodrome", "magasin" et "épis". Ces trois zones sont reliées à trois automates. Ainsi, il a fallu programmer cette ligne au travers de ces trois automates.

Pour la programmation de l'ensemble de cette ligne, nous avons programmé séparément chaque butée et chaque aiguillage. De plus, du fait que certaines butées ou aiguillages se trouvent en limite de zone il a fallu faire de la communication entre les différents automates. Nous avons pour cela utilisé des mots réseau implémentés dans des sémaphores.

4. SEMAPHORE

1. Pourquoi ?

Un sémaphore permet la communication de données entre deux automates qui sont sur le même réseau en s'assurant de la véracité de l'information. S'il y a des problèmes sur le réseau, il ne faut pas qu'une information qui n'est plus à jour soit communiquée à un autre automate. Cela nous permet d'être certains de ne pas avoir d'erreurs lors de nos communications entre les différents API, ce qui est indispensable pour ne pas endommager la partie opérative.

2. Fonctionnement d'un sémaphore

Les sémaphores sont composés de deux séquences disposées dans deux automates. Chaque sémaphore comprend une demande et une autorisation articulées comme dans la figure ci-dessus.

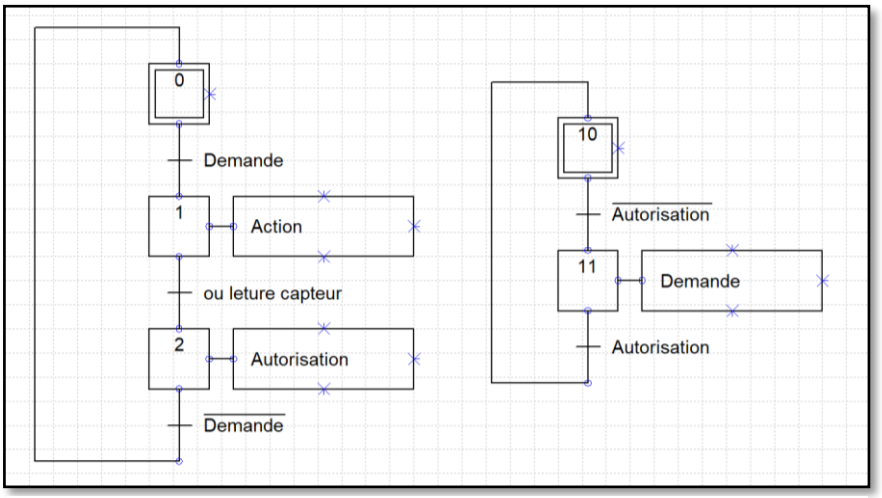


Figure 5 : Fonctionnement d'un sémaphore

5. GESTION DES DEFAUTS

Pour nous assurer qu'aucune palette ne reste coincée, nous créons pour chaque passage difficile un indicateur : un défaut. Si cet indicateur est à 1, cela voudra dire qu'une palette est coincée dans la zone qu'il contrôle.

Les zones difficiles sont les virages, mais aussi le passage des aiguillages.

Nous avons conçu un grafcet type que nous adaptons ensuite selon les cas, mais la forme reste identique pour chacun des défauts ce qui permet un déploiement rapide pour les 21 zones à risque que nous avons identifiées.

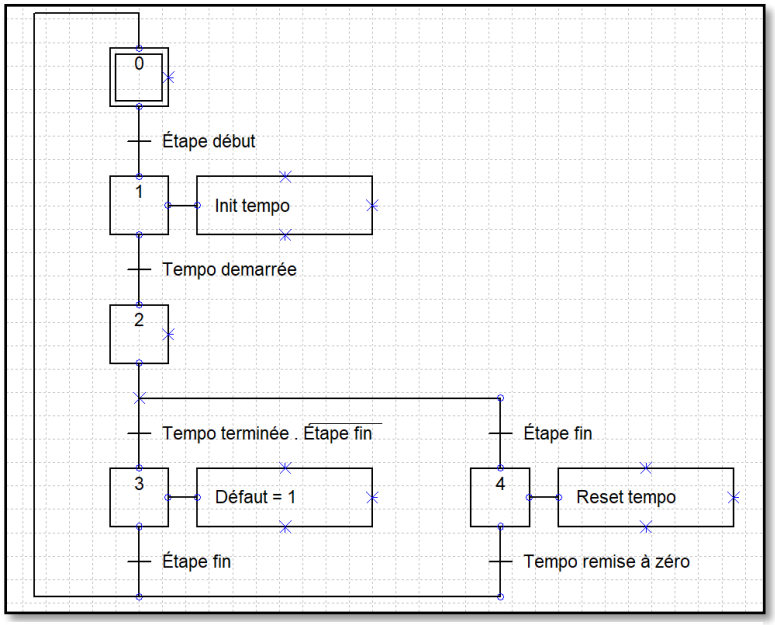


Figure 6 : Grafcet type pour un défaut

Si la palette passe l'étape ou la cellule du début de la zone à risque, le grafcet commence. Si la palette passe le point de fin avant le temps que nous lui avons impartis en fonction de la distance et des virages, alors il n'y a pas de défaut. Dans le cas contraire, on déclenche un défaut. Il se remet à zéro lorsque la palette passe le point de fin avec ou sans intervention humaine.

Par la suite, dans la supervision, on pourra identifier les zones où les blocages sont récurrents et qui demandent, éventuellement, de la maintenance ou une réparation.

Robotisation

1. FONCTIONNEMENT ROBOT

Le robot RX 90 est un robot polyarticulé à 6 degrés de liberté. Il est composé d'une baie de commande avec un écran et d'un bras mécanique qui sont liés grâce à un câble. Sur la baie, il y a un pendant qui nous permet de diriger le robot pour créer des points ou des trajectoires.

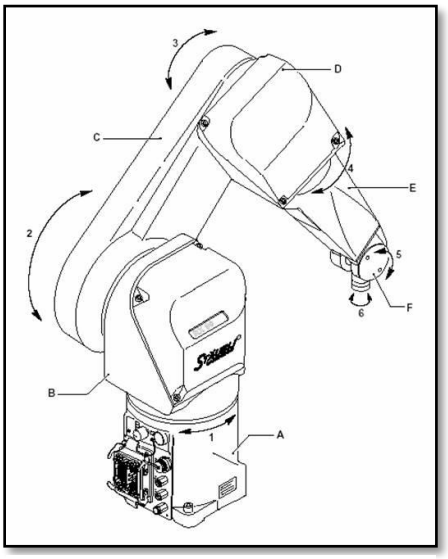


Figure 7 : Robot Staubli RX90

Ce robot possède trois modes de déplacement, un mode où les coordonnées du robot dépendent de l'outil (mode tool), un mode où les coordonnées sont rapportées au pied du robot (mode world) et un mode où les déplacements se font autour des 6 axes en même temps (mode joint).

Pour utiliser le robot, nous avons d'abord créé des points à l'aide du pendant puis, nous avons utilisé ces points dans un programme écrit en langage V+.

2. SCENARIO D'UTILISATION

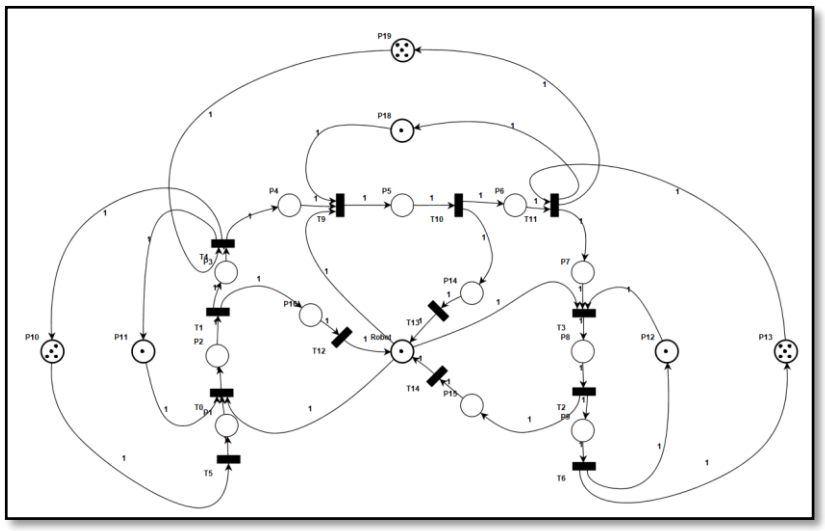


Figure 8 : Réseau de Pétri de la maquette

Dans un premier temps, nous avons créé un scénario pour apprendre à utiliser le robot. Une fois le premier scénario fonctionnel, nous avons fait le scénario final.

1. Premier scénario

Le robot va se déplacer sur 3 postes :

- Un poste de déchargement (Butée B1)
- Un poste d'écriture (Butée B2)
- Un poste de chargement (Butée B3)

Le robot se déplacera selon 9 points prédéfinis, soit 3 points par poste :

- Un point en position basse
- Un point en position d'approche
- Un point en position de transition inter-poste.

On utilise le convoyeur pour déplacer physiquement les palettes en imaginant que le robot les déplace entre chaque poste. Ainsi le robot devra être synchronisé avec le convoyeur et les butées.

- Lorsque le capteur "C2" est actif, on demande au robot de venir décharger une palette.

- Au poste de déchargement, on positionne le robot en position basse, on ferme la pince et on le remonte en position d'approche. À ce moment, on envoie l'information que le mouvement est terminé, ce qui déclenchera l'abaissement de la butée "B1".
- On donne donc l'ordre au robot de se placer au poste d'écriture
- Au poste d'écriture, on positionne le robot en position d'approche. Lorsque la palette sera devant le capteur "C3" on envoie l'ordre au robot de se mettre en position basse et d'ouvrir sa pince. Ensuite, on monte le robot en position d'approche le temps de l'écriture. Enfin, une fois l'écriture terminée, on récupère la pièce en positionnant le robot en position basse et on ferme la pince puis on remonte en position d'approche. À ce moment, on envoie l'information que le mouvement est terminé, ce qui déclenchera l'abaissement de la butée "B2" s'il n'y a pas de palette en "C5".
- On donne donc l'ordre au robot de se placer au poste de chargement.
- Au poste de chargement, on met le robot en position d'approche en attendant que la palette arrive par le convoyeur. Lorsque la palette arrive au capteur "C5", le robot passe en position basse ouvre la pince puis remonte en position d'approche. On envoie alors l'information que le mouvement est terminé, ce qui déclenchera l'abaissement de la butée "B3" si les conditions du grafcet sont réunies.
- Si aucune palette ne se trouve au poste de déchargement le robot se met en position "repos" (do ready) sinon il recommence un cycle au poste de déchargement.
- Pour passer d'un poste à l'autre, on oblige le robot à passer par les positions inter-poste. Cela nous assure la maitrise de la rotation principale du robot.

2. Scénario final

Pour le scénario final, très peu de choses changent. Nous avons toujours trois postes de travail, mais leur fonction change : nous avons maintenant un poste de perçage, un poste de rivetage et un poste de soudure.

Mais le changement principal est que le robot n'a plus un scénario cyclique, c'est désormais l'automate qui décide où il doit aller. Pour cela, nous avons établi deux politiques :

Au plus juste :

On fait rentrer le maximum de palettes dans le magasin et on traite à chaque poste de travail la quantité maximum que peut atteindre le poste suivant (ou le nombre de palettes présentes s'il est inférieur à la capacité).

Au plus tard :

On fait rentrer le moins de palettes possible dans le magasin (nous avons choisi deux palettes) et on les fait avancer deux par deux dans le magasin (sauf s'il n'y en a qu'une seule).

3. PROGRAMMATION ROBOT

Pour la programmation du robot, nous avons fait quatre programmes. Un programme pour aller sur chaque poste de travail et un main qui récupère les ordres de l'automate pour exécuter l'un des trois autres programmes.

Le main est composé d'une boucle infinie et de quatre "if ", un "if" pour chaque poste et un dernier pour la position de repos du robot (DO READY). Ensuite dans chaque "if" on regarde à quel poste se trouve le robot pour savoir quelle trajectoire il va devoir effectuer (s'il est déjà au bon poste, il n'a pas besoin de bouger de son emplacement). Une fois en position "inter-poste" du poste de destination, on lance le sous-programme correspondant.

Ensuite, chaque sous-programme est identique, seules les coordonnées de points sont différentes. L'algorithme est donc le suivant :

- Aller en position d'approche
- Aller en position basse
- Ouvrir la pince
- Fermer la pince
- Aller en position d'approche
- Envoyer un signal à l'automate

4. AMELIORATION COTE AUTOMATE

Nous avons commencé l'amélioration du programme automate avec un premier scénario très simple, cyclique. Nous avons donc envoyé des signaux au robot pour lui commander une action et lui nous renvoie un signal quand il a fini l'action demandée. À la fin de l'action, la butée peut se baisser.

Quand toute cette démarche a fonctionné, nous avons imaginé un scénario plus complexe qui faisait intervenir des priorités. Le fonctionnement du robot est resté inchangé par rapport au premier scénario puisque c'est l'automate qui contrôle le processus.

1. Politique de priorité

Nous avons alors créé deux modes de fonctionnement, un au plus juste qui fait rentrer le minimum de palette dans le magasin pour leur permettre de sortir rapidement. Et l'autre au plus tard qui va faire rentrer le maximum de palette à la fois dans le magasin, cette méthode a une performance supérieure par rapport à la précédente, mais autorise le blocage (plus de 4 palettes dans le premier poste, plus de 5 palettes dans les deux autres postes) . Dans le grafcet qui gère ces

priorités (voir Annexe 1) on a défini un mot ‘priorité’ dont chaque valeur correspond à un poste. Tant que ce mot ne change pas de valeur, le robot travaillera au même endroit.

Nous avons alors dû contrôler la quantité de palettes présente à chaque poste, pour savoir quand il y a des blocages, c’est pourquoi nous avons mis en place des compteurs qui sont utilisés dans les transitions de notre grafcet de priorité.

Nous avons souhaité faire un programme évolutif, donc si l’on veut changer les modes de gestion des priorités, il n’est pas nécessaire de changer tous les programmes, on ne devra modifier qu’un grafcet, celui des gestions des priorités.

2. Ordre de production

Nous pouvons choisir, dans la supervision, un nombre de palettes à produire. La supervision va communiquer cette information avec les automates. Un compteur dans les API signifiera à la supervision le nombre de palettes produites.

Lorsque l’objectif est atteint, la ligne va s’arrêter grâce au bit "arrêt", cela se contrôle dans deux grafcets qui gèrent le démarrage et l’arrêt des moteurs comme suit :

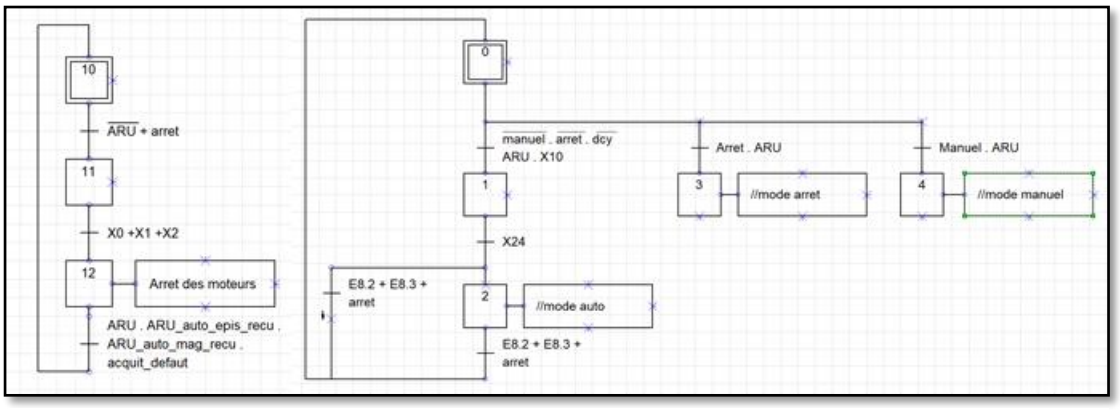


Figure 9 : Grafcets modifiés pour l'arrêt de la ligne

Supervision

Afin de pouvoir visualiser un certain nombre d'informations au sujet de la ligne de production et de pouvoir interagir avec celle-ci, nous avons réalisé une supervision.

1. FONCTIONNEMENT

La supervision que nous avons réalisée au cours de ce projet est articulée autour d'une base de données MySQL et d'un programme C. Dans un premier temps, le programme C récupère les informations dans les automates grâce à une carte "Applicom". Ensuite, ce même programme update les informations collectées dans les APIs via des requêtes SQL. Enfin une page web réalisée en PHP, dialogue avec la base de données pour afficher les informations à l'utilisateur.

Le choix de la politique de production et la variable donnant aux automates le nombre de palettes à produire, sont-elles, entrées dans la page de supervision par l'utilisateur. Ainsi ces deux variables effectuent le chemin inverse. Dans un premier temps, elles sont inscrites dans la base de données par la page PHP. Puis, dans un second temps, elles sont récupérées dans la base par le programme C via des requêtes SQL. Enfin, elles sont inscrites dans les automates via la carte "Applicom" par le code C.

Ainsi, le principe de la supervision mise en place ce résume par le schéma de la figure ci-dessous.

Figure 10 : Schémas de la supervision

2. PROGRAMME C

1. Librairies

Le code s'exécute entre les automates et la base de données. Ainsi, afin de pouvoir bénéficier de fonctions allant lire et écrire des informations nous avons utilisé deux librairies. La librairie "Applicom.h" pour dialoguer avec les automates et la librairie "mysql.h" pour dialoguer avec la base de données et pouvoir ainsi effectuer des requêtes SQL à l'intérieur du code C. De plus, nous avons utilisé la librairie "time.h" afin d'effectuer la gestion des temps de production et durées de défaut.

2. Architecture du code

Le programme C s'articule de la manière suivante. Avec, au début, l'initialisation de la carte "Applicom" des variables et de la connexion à la base.

```
/// Initialisation aplicom
initbus(&statut);        //initialisation applicom

/// Initialisation des variables
for(i = 0; i<21;i++)
{
do
{
    /// Initialisation api MySQL :
    mysql=mysql_init(NULL); // initialisation de la base
    mysql_options(mysql,MYSQL_READ_DEFAULT_GROUP,"option"); // spécification des options de conection

    /// Lecture des capteurs :
    lecture_capteurs(tab_capteurs,statut);

    /// Update de la table capteur :
    update_capteurs(tab_capteurs,mysql,host,user,passwd,db);

    time(&instant);
    instant+=82800;

    /// Lecture etat de la ligne
    lecture_depart(&depart,statut);

    /// Lecture des defauts :
    lecture_defauts(tab_defauts,statut);

    /// Traitement et Update de la table defauts :
    update_defauts(&instant,tab_defauts,stock_defauts,cpt_defauts,&depart,heure_apparition,heure_resolution,
    moy_entre,duree_entre,moy_resolution,duree_resolution,mysql,host,user,passwd,db);

    /// Lecture et update de la politique :
    lecture_politique(&politique,mysql,host,user,passwd,db);
    update_politique(&politique,statut);

    /// Lecture et update des ordres de production :
    lecture_ordre(&ordre_production,mysql,host,user,passwd,db);
    update_ordre(&ordre_production,statut);
    lecture_production(&production,statut);

    /// Traitement des ordre de production :
    traitement_production(&instant,&production,&heure_debut,&heure_fin,&performance,&depart, &stock_depart);

    /// Update des ordre de production :
    update_production(&production,&heure_debut,&heure_fin,&performance,mysql,host,user,passwd,db);

}while(statut == 0);
```

Figure 11 : Structure du programme C

Ensuite, de manière cyclique nous exécutons un ensemble de fonctions :

- Une lecture de la valeur des capteurs et une écriture dans la base.
- Une lecture dans les APIs d'un bit correspondant à l'état "démarré" ou "arrêté" de la ligne.
- Une lecture dans les APIs de la valeur des défauts, un traitement pour le calcul des temps de défauts puis un update dans la base.
- Une lecture dans la base des ordres de productions (nombre de palette à produire et choix de politique) et une écriture de ces variables dans les APIs.
- Et enfin, une lecture du nombre de palettes produite suivie d'un traitement afin d'écrire les données de production dans la base.

3. Dialogue avec la base de données

Pour dialoguer avec la base de données, nous avons utilisé la librairie "mysql.h" pour faire des lectures et écritures dans la base.

Exemple de lecture

Dans la figure 12 nous récupérons la valeur du nombre de palettes à produire par la ligne. Ainsi pour effectuer cette lecture on utilise plusieurs fonctions de "mysql.h" :

- "mysql_init" et "mysql_real_connect ", pour initialiser et se connecter à la base
- "mysql_query" pour envoyer notre requête SQL
- "mysql_store_result" et "mysql_fetch_row" pour récupérer le résultat de notre requête
- "mysql_free_result" et "mysql_close" pour fermer la connexion.

```
void lecture_ordre(int *ordre,MYSQL *mysql,char *u_host,char *u_user,char *u_passwd,char *u_db)
{//lecture du nombre de palettes à produire dans la base
    MYSQL_RES *result = NULL; // résultat de requette SQL
    MYSQL_ROW row; // stock de la prochaine ligne de resultat

    mysql_init(mysql); // initialisation de la base
    if(mysql_real_connect(mysql,u_host,u_user,u_passwd,u_db,0, NULL, 0)) //Connection à la base
    {
        mysql_query(mysql,"SELECT ordre FROM ordre_prod");
        result = mysql_store_result(mysql);  //resultat de la requette

        while((row = mysql_fetch_row(result))) // tant qu'il y a des lignes de resultat
        {
            *ordre=atoi(row[0]);
        }
        mysql_free_result(result); //libere la memoire de MYSQL_ROW
        mysql_close(mysql);//deconnection
    }
    else{ // si erreur de connection
        printf("Erreur lors de la connection a la BDD !");
    }
}
```

Figure 12 : Exemple de lecture SQL

Exemple d'écriture

De même que pour la lecture de données dans la base MySQL, nous utilisons les fonctions de "mysql.h ". Cependant, ici la requête est un update ainsi, nous n'avons pas besoin de récupérer de résultat.

La figure ci-dessous montre l'update de la valeur des capteurs dans la base.

```
void update_capteurs(short *t_capteurs,MYSQL *mysql,char *u_host,char *u_user,char *u_passwd,char *u_db)
{//Update des capteurs
    char query[512];
    unsigned int i = 0;

    mysql_init(mysql);

    if(mysql_real_connect(mysql,u_host,u_user,u_passwd,u_db,0, NULL, 0)) //Connection à la base
        {
            for(i=1;i<23;i++)
            {
                sprintf(query,"UPDATE capteur SET valeur =%d WHERE id = %d",t_capteurs[i-1],i);
                mysql_query(mysql,query);
            }
            mysql_close(mysql);//deconnection
        }
        else{ // si erreur de connection
        printf("Erreur lors de la connection a la BDD !");
        }
}
```

Figure 13 : Exemple d'écriture SQL

4. Dialogue avec les automates

Pour dialoguer avec les automates, nous avons utilisé la librairie "Applicom.h" pour faire des lectures et écritures dans la mémoire des automates.

Exemple de lecture

```
void lecture_capteurs_hippodrome(short *t_hippo,short etat)
{//lecture des capteurs de la station 4
    short nchan = 0;          // Numéro de canal
    short neq = 4;            // Numéro d'équipement
    short nb = 16;            // Nombre de variables
    long  adr = 0;            // Adresse de la première variable
    short tabl[1];            // Table recevant les données

    readpackibit(&nchan, &neq, &nb, &adr, tabl, &etat);
    if (!etat)
    {
        transwordbit(&nb, tabl, t_hippo, &etat);
    }
    else printf(" problème sur l'équipement n° %hd \n",neq);
}
```

Figure 14 : Exemple de lecture dans un automate

Pour lire des informations dans la mémoire des automates, nous utilisons ici la fonction "readpackibit" qui permet de récupérer une plage de bits qui correspondent aux entrées de

l'automate dans cet exemple, car nous voulons lire la valeur des capteurs. Ensuite, cette fonction est couplée avec une fonction "transwordbit" qui permet-elle de récupérer chaque bit indépendamment dans un tableau.

D'autres fonctions nous ont été utiles pour lire des informations dans les automates comme la fonction "readpackbit" qui permet de lire des bits dans la mémoire de l'automate ou encore "readword" pour lire directement des mots.

Ces fonctions ont besoin de plusieurs paramètres comme l'adresse de la première variable, le nombre de variables à lire ou encore l'adresse MPI de l'équipement où lire la valeur.

Exemple d'écriture

```
void update_ordre(int *ordre,short etat)
{//update du nombre de palettes à produre dans la station hippodrome
    short nchan = 0;          // Numéro de canal
    short neq = 4;            // Numéro d'équipement
    short nb = 1;             // Nombre de variables
    long adr = 50;            // Adresse de la première variable
    short tabl[1];            // Tableau recevant les données

    tabl[0]=*ordre;
    writeword(&nchan, &neq, &nb, &adr, tabl, &etat);
    if (etat)
    {
        printf(" problème sur l'équipement n° %hd \n",neq);
    }
}
```

Figure 15 : Exemple d'écriture dans un automate

De même que pour la lecture, nous avons utilisé la fonction dédiée "writeword" pour écrire dans un mot mémoire. Dans la figure ci-dessous nous pouvons voir l'écriture du nombre de palettes à produire dans le mot mémoire M150.0.

3. PROGRAMMATION PHP

1. Rôle du PHP

Nous avons créé un site PHP pour créer un lien entre la supervision et l'utilisateur. À partir de cette interface, l'utilisateur pourra définir la politique de priorité que la maquette va adopter. Le rôle principal de cette interface va être de montrer à l'utilisateur l'état de la maquette : on peut y voir l'état des capteurs de la maquette, mais aussi toutes les statistiques liées aux défauts.

2. Définition de l'apparence du site

Nous avons choisi de faire un site au design sobre Nous sommes partis de la couleur bleue de l'Istia et la couleur mauve de la formation SAGI.

3. Programmation de la page

La programmation de ce site, nous a permis de revoir des notions de PHP, HTML et surtout CSS que nous avions survolé en cours. Nous avons beaucoup cherché sur Internet des tutoriels et des cours pour apprendre le PHP.

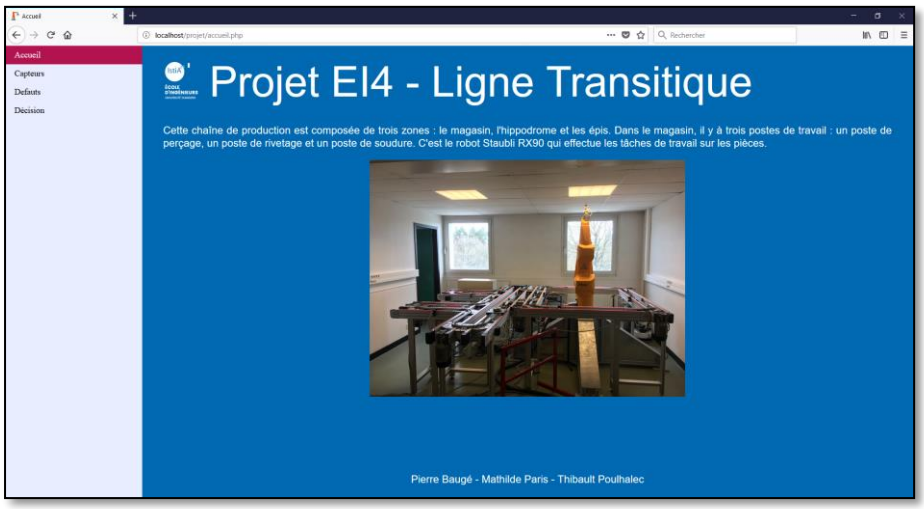


Figure 16 : Page d'accueil du site PHP

Nous avons fait en sorte de créer plusieurs programmes que nous pouvons réutiliser grâce à la fonction :

include(nomDeLaPage);

Comme cela sur chacune de nos pages seuls les articles changent. De plus, si l'on changer quelque chose dans l'entête (header) par exemple, on ne doit le faire qu'à un seul endroit.

Figure 17 : Structure d'une page PHP

4. Fonctionnalités de la page

En plus de la page d'accueil, nous créons trois autres pages. La première comprend l'état en temps réel des capteurs. La seconde contient un tableau avec pour chaque défaut, le nombre de fois où il est apparu, le moyen temps entre deux défauts et le temps moyen de résolution du défaut. Et la dernière page est celle des décisions, elle a deux fonctions : la première permet de choisir la politique de priorité (au plus juste ou au plus tard), mais aussi de voir la politique en cours, la deuxième fonction nous permet de choisir le nombre de palettes à produire. Un tableau nous permet de voir combien de palettes il reste à produire et la performance de la ligne en palette/heure.

Nous voulions que la page se rafraichisse toute seule pour avoir une page à jour. Pour cela nous avons trouvé un code qui le permet sans voir de clignotements lors du rafraichissement :

```
<script src="http://code.jquery.com/jquery-latest.js
"></script>
<script>
        var refreshId = setInterval(function()
        {
                $('#capteur').fadeOut(0).load('AfficheCapteurs.php').fadeIn(0)
```

Figure 18 : Code du rafraichissement automatique

"Capteur" correspond à l'ID de la section dans laquelle on va faire le rafraichissement. "AfficheCapteurs.php" correspond au code qui va être appelé lors du rafraichissement. Les valeurs de "fadeOut" et de "fadeIn" sont des valeurs en millisecondes qui correspondent aux temps d'apparition et de disparition du rafraichissement. Nous avons choisi la valeur 0 pour que l'on ne voie rien. Le "1000" est la fréquence en millisecondes du rafraichissement.

Nous avons dû communiquer avec la base de données autant en lecture qu'en écriture. Pour ce faire nous avons utilisé les bouts de code des figures 19 et 20.

```php
  try
  {
     // On se connecte à MySQL
     $bdd = new PDO('mysql:host=localhost;dbname=mabdd;charset=utf8', 'root', '');
     $sql = "UPDATE politique SET valeur = $value";
     $stmt = $bdd->prepare($sql);
     $stmt->execute();
  }
  catch(Exception $e)
  {
     // En cas d'erreur, on affiche un message et on arrête tout
     die('Erreur : '.$e->getMessage());
  }

  $conn = null;
}
```

Figure 19 : Code d'écriture dans une BDD en PHP

```php
try
{
   $bdd = new PDO('mysql:host=localhost;dbname=mabdd;charset=utf8', 'root', '');
}
catch (Exception $e)
{
  die('Erreur : ' . $e->getMessage());
}
// Exécution de la requête
$reponse = $bdd->query('SELECT * FROM ordre_prod');
while($donnees = $reponse->fetch()){
?>
  <tr>
    // Affichage des données dans un tableau
    <td><?php echo $donnees['id']; ?></td>
    <td><?php echo $donnees['nom']; ?></td>
  </tr>
<?php
}
$reponse->closeCursor(); // Termine le traitement de la requête
```

Figure 20 : Code d'une lecture de BDD en PHP

Conclusion

Conclusion générale :

Au cours de ce projet dans lequel nous avons réalisé la supervision d'une ligne de production industrielle, nous avons abordé de nombreux domaines. Nous sommes partis de la programmation des automates pour faire fonctionner cette ligne de manière automatique. Ensuite, nous avons amélioré les programmes automates pour effectuer la détection de défauts. Nous avons par la suite effectué la programmation du robot avec, dans un premier temps, un scénario de test afin d'appréhender ce qui pour nous était un nouveau type de programmation. Puis dans un second temps avec un programme final nous permettant de commander le robot par l'automate via une communication.

Par la suite, nous avons réalisé de nouvelles modifications sur les programmes automate, afin d'implémenter de nouvelles fonctionnalités comme la gestion des ordres de production ou encore le choix et la gestion du type de politique de traitement des palettes à adopter. Après cela, nous avons développé une supervision composée d'un programme C pour dialoguer avec les automates, d'un ensemble de pages PHP pour communiquer avec l'utilisateur et d'une base de données MySQL pour faire le lien entre le page PHP et le programme C.

Ce projet aurait pu être complété par un portage de nos pages de supervision PHP sur un terminal mobile avec un accès à distance à la base de données. Cependant, nous n'avons pas pu réaliser cette fonctionnalité par manque de temps.

Nous retenons de ce projet une expérience complète et enrichissante qui nous sera sans aucun doute profitable et utile à l'avenir.

Pierre :

Personnellement, j'ai beaucoup apprécié ce projet. Au cours des différentes séances, j'ai pu revoir des notions acquises l'année passée comme la programmation C, la gestion de base de données ou encore la programmation d'automates. De plus, j'ai aussi découvert de nouvelles choses comme la programmation d'un robot industriel et la communication entre automate,programme C et base de données. Je trouve que ce projet fut une expérience enrichissante appuyée par le fait qu'il nous a mis dans une situation que l'on est fortement susceptible de rencontrer après nos études.

CONCLUSION

Mathilde :

J'ai trouvé ce projet vraiment intéressant et enrichissant, il nous as permis de voir un exemple concret d'application très complet. De plus, nous avons pu découvrir comment programmé les différentes inter-connexions entre une base de données, un programme C et un programme automate. Nous avons pu aussi appréhender la programmation d'un robot, chose que nous n'avions encore jamais fait au sein de l'école.

Thibault :

J'ai apprécié participer à ce projet, il m'a permis de développer mes connaissances dans plusieurs domaines tel que la programmation d'automates, le PHP, grâce au site que nous avons eu à réaliser, et la programmation de robot que j'avais déjà eu l'occasion d'aborder lors de mon stage de fin de DUT. De plus, j'aurais aimé que l'on soit un peu plus de temps pour pouvoir faire l'application Android destiné à la supervision. Malgré cela, je trouve que ce projet est très complet et nous a permis de toucher à plusieurs domaines différents.

Bibliographie

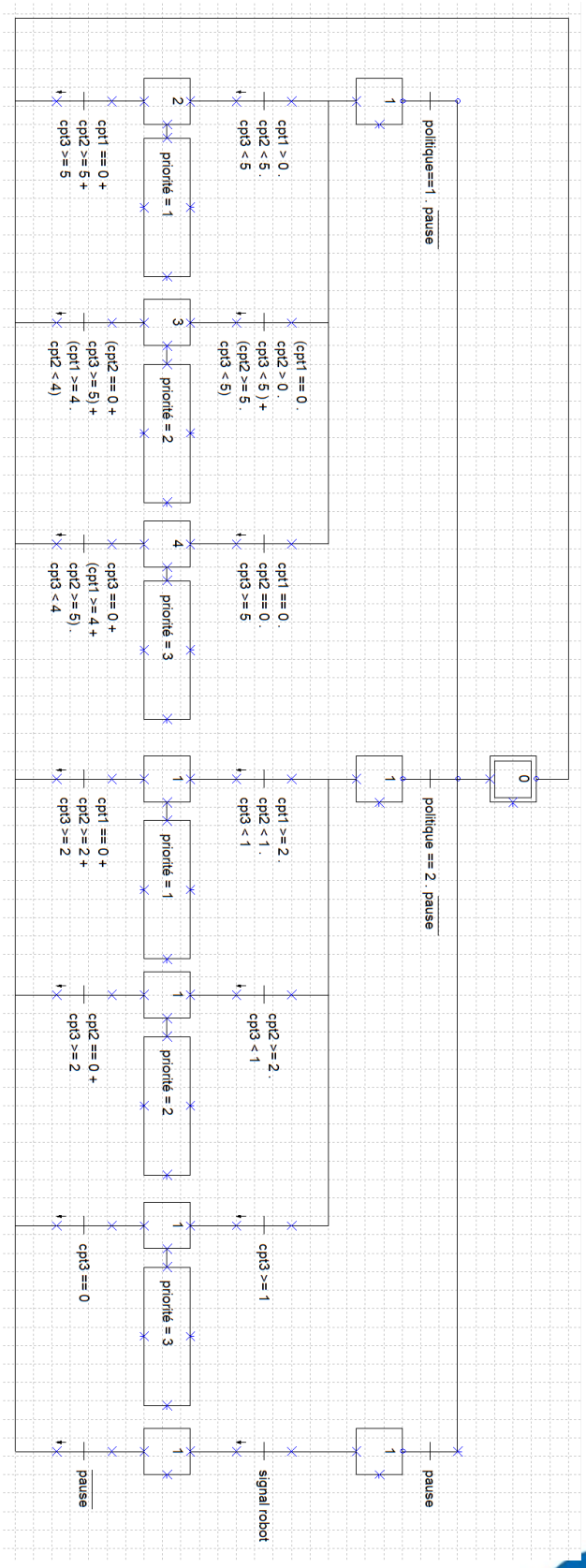
Documentation LIST :

https://cache.industry.siemens.com/dl/files/446/45523446/att_79273/v1/s7awl_c.pdf

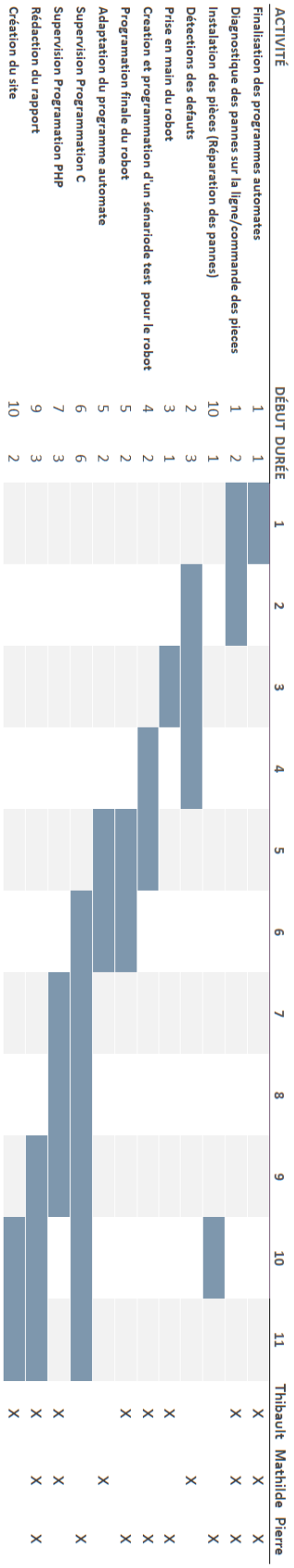
Table des illustrations

Annexes

1. POLITIQUE DE PRIORITE



2. DIAGRAMME GANTT



3. PROCEDURES DU ROBOT

Démarrage du robot

1) Allumer la baie et le terminal
2) Vérifier que commutateur soit sur "Terminal"
3) Attendre le point "." (Appuyer sur "entrer")
4) On tape "ENABLE POWER" ou "Comp/Pwr" sur le pendant
5) Dans les 15 secondes, appuyer sur "Arm power on" sur l'amoire
6) On tape "SPEED 10"
7) On tape "DO READY" (mise en position initiale)

Procédure d'arrêt du robot

1) "Abort" sur le terminal
2) "Disable power"
3) Éteindre l'écran
4) Arrêter la baie

Procédure création d'un point

1) Mettre le robot dans la position souhaitée
2) Taper "HERE #nom du point"
3) Pour se rendre à ce point taper "DO MOVE #nom du point"

Astuces :

Pour repasser sur le pendant appuyer sur "Comp/Pwr".

Pour afficher tous les dossier (équivalent ls) "fdirectory"

4. PROGRAMME ROBOT

1. Main.pg

```
SPEED 10
READY
SIGNAL -1, -2, -3, -4, -5
poste = 0
SET #p1i = #PPOINT(98.835,-47.497,89.927,0.46,87.154,1.218)
SET #p2i = #PPOINT(-12.593,-62.221,91.155,0,84.495,-11.229)
SET #p3i = #PPOINT(-114.554,-47.497,89.927,0.46,87.154,1.218)

WHILE (1) DO
  IF SIG(1001) THEN
        IF (poste <> 1) THEN
          IF (poste == 2) THEN
                MOVE #p2i
          END
          IF (poste == 3) THEN
                MOVE #p3i
                MOVE #p2i
          END
          MOVE #p1i
        END
        CALL poste1.pg()
        SIGNAL -1
        poste = 1
  END
  IF SIG(1002) THEN
        IF (poste <> 2) THEN
          IF (poste == 1) THEN
                MOVE #p1i
          END
          IF (poste == 3) THEN
                MOVE #p3i
          END
          MOVE #p2i
        END
```

```
            CALL poste2.pg()
            SIGNAL -2
            poste = 2
        END
        IF SIG(1004) THEN
            IF (poste <> 3) THEN
              IF (poste == 1) THEN
                  MOVE #p1i
                  MOVE #p2i
              END
              IF (poste == 2) THEN
                  MOVE #p2i
              END
              MOVE #p3i
            END
            CALL poste3.pg()
            SIGNAL -4
            poste = 3
        END
        IF SIG(1005) THEN
            READY
            SIGNAL 5
            READY
            SIGNAL -5
            poste = 0
        END
    END
.END
```

2. Poste1.pg

```
.PROGRAM poste1.pg()
    SET #p1i = #PPOINT(98.835,-47.497,89.927,0.46,87.154,1.218)
    SET #p1a = #PPOINT(102.581,-18.177,89.927,0.46,94.597,1.217)
    SET #p1b = #PPOINT(102.236,-11.072,89.927,1.66,102.802,12.434)

    MOVE #p1a
    MOVE #p1b
```

```
        OPENI
        WAIT 1
        CLOSEI
        MOVE #p1a
        SIGNAL 1
        MOVE #p1a
.END
```

3. Poste2.pg

```
.PROGRAM poste2.pg()
        SET #p2i = #PPOINT(-12.593,-62.221,91.155,0,84.495,-11.229)
        SET #p2b = #PPOINT(-12.593,-50.739,163.696,0,68.855,-11.231)
        SET #p2a = #PPOINT(-12.593,-54.997,151.307,0,84.495,-11.229)

        MOVE #p2a
        MOVE #p2b
        OPENI
        WAIT 1
        CLOSEI
        MOVE #p2a
        SIGNAL 2
        MOVE #p2a
.END
```

4. Poste3.pg

```
.PROGRAM poste3.pg()
        SET #p3i = #PPOINT(-114.554,-47.497,89.927,0.46,87.154,1.218)
        SET #p3b = #PPOINT(-114.554,-11.072,89.927,1.66,87.154,1.218)
        SET #p3a = #PPOINT(-114.554,-18.177,89.927,0.46,87.154,1.218)
        MOVE #p3a
        MOVE #p3b
        OPENI
        WAIT 1
        CLOSEI
        MOVE #p3a
        SIGNAL 4
        MOVE #p3a
.END
```

5. PROGRAMME C

```
#include <winsock2.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include "api_mysql\MySQL\mysql.h"
#include "Applicom.h"

///Fonction pour les capteurs :
void lecture_capteurs_hippodrome (short *t_hippo,short etat);
void lecture_capteurs_magasin (short *t_mag,short etat);
void lecture_capteurs_epis(short *t_epi,short etat);
void lecture_capteurs(short *t_capteurs,short status);
void update_capteurs(short *t_capteurs,MYSQL *mysql,char *u_host,char *u_user,char
*u_passwd,char *u_db);

///Fonction pour les defauts :
void lecture_defauts_hippodrome (short *t_defauts_hippo,short etat);
void lecture_defauts_magasin (short *t_defauts_mag,short etat);
void lecture_defauts_epis(short *t_defauts_epi,short etat);
void lecture_defauts(short *t_defauts,short status);
void update_defauts(time_t *now,short *t_defauts,short *t_stock,int *c_defauts,int*depart,time_t
*h_apparition,time_t *h_resolution,time_t *m_entre,time_t *d_entre,time_t *m_resolution,time_t
*d_resolution,MYSQL *mysql,char *u_host,char *u_user,char *u_passwd,char *u_db);

///Fonction pour la politique :
void lecture_politique(int *pol,MYSQL *mysql,char *u_host,char *u_user,char *u_passwd,char
*u_db);
void update_politique(int *pol,short etat);

///Fonction pour les ordres de production :
void lecture_ordre(int *ordre,MYSQL *mysql,char *u_host,char *u_user,char *u_passwd,char
*u_db);
void update_ordre(int *ordre,short etat);
```

```
void lecture_production (int *production,short etat);
void traitement_production(time_t *now,int *production,time_t *h_debut,time_t *h_fin,double
*performance,int*depart, int *s_depart);
void update_production(int *production,time_t *h_debut,time_t *h_fin,double
*performance,MYSQL *mysql,char *u_host,char *u_user,char *u_passwd,char *u_db);

void lecture_depart(int *depart,short etat);

int main()
{
   /// Paramètres de la base :
   char* host = "localhost";
   char* user = "root";
   char* passwd = "";
   char* db = "mabdd";

   /// Variables :
   MYSQL *mysql;             // Structure MySQL
   short statut;             // Status
   short tab_capteurs[24];    // Image des capteurs
   short tab_defauts[21];     // Image des défauts
   int cpt_defauts[21];       // Compteurs sur chaque défauts
   time_t moy_entre[21];      // Moyenne d'apparition entre chaque défauts
   time_t moy_resolution[21];  // Moyenne de résolution de chaque défauts
   time_t instant;           // Temps Courant
   time_t heure_fin=0;
   time_t heure_debut=0;
   int politique;
   int ordre_production;
   int production=0;
   double performance=0.0;
   unsigned int i=0;
   int depart;
   short stock_defauts[21];
   int stock_depart=0;

   time_t duree_entre[21];
```

```
time_t duree_resolution[21];
time_t heure_apparition[21];
time_t heure_resolution[21];

/// Initialisation applicom
initbus(&statut);       //initialisation applicom

/// Initialisation des variables
for(i = 0; i<21;i++)
{
   cpt_defauts[i]=0;
   tab_defauts[i]=0;
   stock_defauts[i]=0;
   moy_entre[i]=82800;
   moy_resolution[i]=82800;
   heure_apparition[i]=82800;
   heure_resolution[i]=82800;
   duree_entre[i]=82800;
   duree_resolution[i]=82800;
}

do
{
   /// Initialisation api MySQL :
   mysql=mysql_init(NULL); // initialisation de la base
   mysql_options(mysql,MYSQL_READ_DEFAULT_GROUP,"option");
   // spécification des options de conection

   /// Lecture des capteurs :
   lecture_capteurs(tab_capteurs,statut);

   /// Update de la table capteur :
   update_capteurs(tab_capteurs,mysql,host,user,passwd,db);

   time(&instant);
   instant+=82800;
```

```
        /// Lecture etat de la ligne
        lecture_depart(&depart,statut);

        /// Lecture des defauts :
        lecture_defauts(tab_defauts,statut);

        /// Traitement et Update de la table defauts :
update_defauts(&instant,tab_defauts,stock_defauts,cpt_defauts,&depart,heure_apparition,heure_r
esolution,moy_entre,duree_entre,moy_resolution,duree_resolution,mysql,host,user,passwd,db);

        /// Lecture et update de la politique :
        lecture_politique(&politique,mysql,host,user,passwd,db);
        update_politique(&politique,statut);

        /// Lecture et update des ordres de production :
        lecture_ordre(&ordre_production,mysql,host,user,passwd,db);
        update_ordre(&ordre_production,statut);
        lecture_production(&production,statut);

        /// Traitement des ordre de production :
traitement_production(&instant,&production,&heure_debut,&heure_fin,&performance,&depart,
&stock_depart);

        /// Update des ordre de production :
update_production(&production,&heure_debut,&heure_fin,&performance,mysql,host,user,passw
d,db);

    }while(statut == 0);

    exit(0);
    return 0;
}

///Fonction pour les capteurs :
```

```
void lecture_capteurs_hippodrome(short *t_hippo,short etat)
{//lecture des capteurs de la station 4
   short nchan = 0;        // Numéro de canal
   short neq = 4;          // Numéro d'équipement
   short nb = 16;          // Nombre de variables
   long  adr = 0;          // Adresse de la première variable
   short tabl[1];          // Table recevant les données

   readpackibit(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (!etat)
   {
      transwordbit(&nb, tabl, t_hippo, &etat);
   }
   else printf(" problème sur l'équipement n° %hd \n",neq);
}

void lecture_capteurs_magasin(short *t_mag,short etat)
{//lecture des capteurs de la station 5
   short nchan = 0;        // Numéro de canal
   short neq = 5;          // Numéro d'équipement
   short nb = 16;          // Nombre de variables
   long  adr = 0;          // Adresse de la première variable
   short tabl[1];          // Table recevant les données

   readpackibit(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (!etat)
   {
      transwordbit(&nb, tabl, t_mag, &etat);
   }
   else printf(" problème sur l'équipement n° %hd \n",neq);
}
```

```
void lecture_capteurs_epis(short *t_epi,short etat)
{//lecture des capteurs de la station 6
   short nchan = 0;        // Numéro de canal
   short neq = 6;          // Numéro d'équipement
   short nb = 16;          // Nombre de variables
   long  adr = 0;          // Adresse de la première variable
   short tabl[1];          // Table recevant les données

   readpackibit(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (!etat)
   {
      transwordbit(&nb, tabl, t_epi, &etat);
   }
   else printf(" problème sur l'équipement n° %hd \n",neq);
}

void lecture_capteurs(short *t_capteurs,short status)
{//tri dans le tableau passé en parametre les valeurs des capteurs

   short tab_hippo[16];     // Valeur des capteurs de l'hypodrome
   short tab_mag[16];       // Valeur des capteurs du magasin
   short tab_epi[16];       // Valeur des capteurs des épis

   lecture_capteurs_hippodrome(tab_hippo,status);
   lecture_capteurs_magasin(tab_mag,status);
   lecture_capteurs_epis(tab_epi,status);

   t_capteurs[0]=tab_mag[1];
   t_capteurs[1]=tab_mag[2];
   t_capteurs[2]=tab_mag[3];
   t_capteurs[3]=tab_mag[4];
   t_capteurs[4]=tab_hippo[3];
   t_capteurs[5]=tab_mag[6];
   t_capteurs[6]=tab_mag[7];
   t_capteurs[7]=tab_hippo[2];
   t_capteurs[8]=tab_epi[0];
```

```
    t_capteurs[9]=tab_epi[1];
    t_capteurs[10]=tab_epi[2];
    t_capteurs[11]=tab_epi[3];
    t_capteurs[12]=tab_hippo[5];
    t_capteurs[13]=tab_mag[12];
    t_capteurs[14]=tab_mag[13];
    t_capteurs[15]=tab_hippo[6];
    t_capteurs[16]=tab_mag[15];
    t_capteurs[17]=tab_hippo[11];
    t_capteurs[18]=tab_epi[4];
    t_capteurs[19]=tab_epi[5];
    t_capteurs[20]=tab_mag[0];
    t_capteurs[21]=tab_hippo[7];
    t_capteurs[22]=tab_hippo[8];
    t_capteurs[23]=tab_hippo[10];
}

void update_capteurs(short *t_capteurs,MYSQL *mysql,char *u_host,char *u_user,char
*u_passwd,char *u_db)
{//Update des capteurs
    char query[512];
    unsigned int i = 0;

    mysql_init(mysql);

    if(mysql_real_connect(mysql,u_host,u_user,u_passwd,u_db,0, NULL, 0)) //Connection à la base
        {
            for(i=1;i<23;i++)
            {
                sprintf(query,"UPDATE capteur SET valeur =%d WHERE id = %d",t_capteurs[i-1],i);
                mysql_query(mysql,query);
            }
            mysql_close(mysql);//deconnection
        }
        else{ // si erreur de connection
        printf("Erreur lors de la connection a la BDD !");
        }
```

```
}
///Fonction pour les defauts :
void lecture_defauts_hippodrome (short *t_defauts_hippo,short etat)
{//lecture des defauts de la station 4
   short nchan = 0;        // Numéro de canal
   short neq = 4;          // Numéro d'équipement
   short nb = 10;          // Nombre de variables
   long  adr = 150*8;      // Adresse de la première variable
   short tabl[1];          // Table recevant les données

   readpackbit(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (!etat)
   {
      transwordbit(&nb, tabl, t_defauts_hippo, &etat);
   }
   else printf(" problème sur l'équipement n° %hd \n",neq);
}

void lecture_defauts_magasin (short *t_defauts_mag,short etat)
{//lecture des defauts de la station 5
   /// premier bloc
   short nchan = 0;        // Numéro de canal
   short neq = 5;          // Numéro d'équipement
   short nb = 1;           // Nombre de variables
   long adr = 117*8;       // Adresse de la première variable
   short tabl[1];          // Tableau recevant les données
   short t_mag[8];         // Tableau intermediaire
   int i=0;

   readpackbit(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (!etat)
   {
      transwordbit(&nb, tabl, t_mag, &etat);
   }
   else printf(" problème sur l'équipement n° %hd \n",neq);

   t_defauts_mag[0]=t_mag[0];
```

```
   /// deuxieme bloc
   adr = 150*8;      // Adresse de la première variable
   nb = 5;           // Nombre de variables

   readpackbit(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (!etat)
   {
      transwordbit(&nb, tabl, t_mag, &etat);
   }
   else printf(" problème sur l'équipement n° %hd \n",neq);

   for(i=1;i<6;i++)
   {
      t_defauts_mag[i]=t_mag[i-1];
   }
}

void lecture_defauts_epis(short *t_defauts_epi,short etat)
{//lecture des defauts de la station 6
   /// premier bloc
   short nchan = 0;        // Numéro de canal
   short neq = 6;          // Numéro d'équipement
   short nb = 1;           // Nombre de variables
   long adr = 126*8;       // Adresse de la première variable
   short tabl[1];          // Table recevant les données
   short t_epi[8];         // Tableau intermediaire
   int i=0;

   readpackbit(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (!etat)
   {
      transwordbit(&nb, tabl, t_epi, &etat);
   }
   else printf(" problème sur l'équipement n° %hd \n",neq);

   t_defauts_epi[0]=t_epi[0];
```

```
    /// deuxieme bloc
    adr = 150*8;      // Adresse de la première variable
    nb = 4;           // Nombre de variables

    readpackbit(&nchan, &neq, &nb, &adr, tabl, &etat);
    if (!etat)
    {
       transwordbit(&nb, tabl, t_epi, &etat);
    }
    else printf(" problème sur l'équipement n° %hd \n",neq);

    for(i=1;i<5;i++)
    {
       t_defauts_epi[i]=t_epi[i-1];
    }
}
```

```
void lecture_defauts(short *t_defauts,short status)
{//tri dans le tableau passé en parametre les valeurs des defauts
   short tab_hippo[16];     // Valeur des defauts de l'hypodrome
   short tab_mag[16];       // Valeur des defauts du magasin
   short tab_epi[16];       // Valeur des defauts des épis
   int i=0;

   lecture_defauts_hippodrome(tab_hippo,status);
   lecture_defauts_magasin(tab_mag,status);
   lecture_defauts_epis(tab_epi,status);

   for(i=0;i<10;i++)
   {
      t_defauts[i]=tab_hippo[i];
   }
   for(i=10;i<16;i++)
   {
      t_defauts[i]=tab_mag[i-10];
   }
   for(i=16;i<21;i++)
   {
      t_defauts[i]=tab_epi[i-16];
   }
}

void update_defauts(time_t *now,short *t_defauts,short *t_stock,int *c_defauts,int*depart,time_t
*h_apparition,time_t *h_resolution,time_t *m_entre,time_t *d_entre,time_t *m_resolution,time_t
*d_resolution,MYSQL *mysql,char *u_host,char *u_user,char *u_passwd,char *u_db)
{//Update des defauts
   char query[512];
   char r_entre[512];
   char r_resolution[512];
   unsigned int i = 0;

   if (*depart!=0)
   {
```

```
  for(i=0;i<21;i++)
  {
    if(t_defauts[i]==1 && t_defauts[i]!=t_stock[i])//apparition du defaut
    {
      c_defauts[i]+=1;
      h_apparition[i]=*now;
      if(c_defauts[i]!=1)
      {
        d_entre[i]=h_apparition[i]-h_resolution[i]+82800;
        m_entre[i]=(m_entre[i]*(c_defauts[i]-2)+d_entre[i])/(c_defauts[i]-1);
      }
    }
    if(t_defauts[i]==0 && t_defauts[i]!=t_stock[i]&&c_defauts[i]!=0)//disparition du defaut
    {
      h_resolution[i]=*now;
      d_resolution[i]=h_resolution[i]-h_apparition[i]+82800;
      m_resolution[i]=(m_resolution[i]*(c_defauts[i]-1)+d_resolution[i])/c_defauts[i];
    }
    t_stock[i]=t_defauts[i];
  }
}

mysql_init(mysql);
if(mysql_real_connect(mysql,u_host,u_user,u_passwd,u_db,0, NULL, 0)) //Connection à la base
  {
    for(i=1;i<22;i++)
    {
      strftime(r_entre,512,"%H:%M:%S\n",localtime(&m_entre[i-1]));
      sprintf(query,"UPDATE defauts SET entre ='%s' WHERE id = %d",r_entre,i);
      mysql_query(mysql,query);

      strftime(r_resolution,512,"%H:%M:%S\n",localtime(&m_resolution[i-1]));
      sprintf(query,"UPDATE defauts SET resolution ='%s' WHERE id = %d",r_resolution,i);
      mysql_query(mysql,query);

      sprintf(query,"UPDATE defauts SET nombre =%d WHERE id = %d",c_defauts[i-1],i);
      mysql_query(mysql,query);
```

```
        }
        mysql_close(mysql);//deconnection
      }
      else{ // si erreur de connection
      printf("Erreur lors de la connection a la BDD !");
      }
}

///Fonction pour la politique :
void lecture_politique(int *pol,MYSQL *mysql,char *u_host,char *u_user,char *u_passwd,char
*u_db)
{//Lecture du choix de politique dans la base
   MYSQL_RES *result = NULL; // résultat de requette SQL
   MYSQL_ROW row; // stock de la prochaine ligne de resultat

   mysql_init(mysql); // initialisation de la base
   if(mysql_real_connect(mysql,u_host,u_user,u_passwd,u_db,0, NULL, 0)) //Connection à la base
   {
      mysql_query(mysql,"SELECT valeur FROM politique");
      result = mysql_store_result(mysql);  //resultat de la requette

      while((row = mysql_fetch_row(result))) // tant qu'il y a des lignes de resultat
      {
         *pol=atoi(row[0]);
      }
      mysql_free_result(result); //libere la memoire de MYSQL_ROW
      mysql_close(mysql);//deconnection
   }
   else{ // si erreur de connection
      printf("Erreur lors de la connection a la BDD !");
   }
}
```

```
void update_politique(int *pol,short etat)
{//update de la politique dans la station magasin
   short nchan = 0;        // Numéro de canal
   short neq = 5;          // Numéro d'équipement
   short nb = 1;           // Nombre de variables
   long adr = 161;         // Adresse de la première variable
   short tabl[1];          // Tableau recevant les données

   tabl[0]=*pol;
   writeword(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (etat)
   {
      printf(" problème sur l'équipement n° %hd \n",neq);
   }
}


///Fonction pour les ordres de production :
void update_ordre(int *ordre,short etat)
{//update du nombre de palettes à produre dans la station hippodrome
   short nchan = 0;        // Numéro de canal
   short neq = 4;          // Numéro d'équipement
   short nb = 1;           // Nombre de variables
   long adr = 50;          // Adresse de la première variable
   short tabl[1];          // Tableau recevant les données

   tabl[0]=*ordre;
   writeword(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (etat)
   {
      printf(" problème sur l'équipement n° %hd \n",neq);
   }
}
```

```
void lecture_ordre(int *ordre,MYSQL *mysql,char *u_host,char *u_user,char *u_passwd,char
*u_db)
{//lecture du nombre de palettes à produire dans la base
   MYSQL_RES *result = NULL; // résultat de requette SQL
   MYSQL_ROW row; // stock de la prochaine ligne de resultat

   mysql_init(mysql); // initialisation de la base
   if(mysql_real_connect(mysql,u_host,u_user,u_passwd,u_db,0, NULL, 0)) //Connection à la base
   {
      mysql_query(mysql,"SELECT ordre FROM ordre_prod");
      result = mysql_store_result(mysql);  //resultat de la requette

      while((row = mysql_fetch_row(result))) // tant qu'il y a des lignes de resultat
      {
         *ordre=atoi(row[0]);
      }
      mysql_free_result(result); //libere la memoire de MYSQL_ROW
      mysql_close(mysql);//deconnection
   }
   else{ // si erreur de connection
      printf("Erreur lors de la connection a la BDD !");
   }
}
```

```
void lecture_production (int *production,short etat)
{// Lecture du nombre de palette produite dans la station hippodrome
   short nchan = 0;        // Numéro de canal
   short neq = 4;          // Numéro d'équipement
   short nb = 1;           // Nombre de variables
   long adr = 118;         // Adresse de la première variable
   short tabl[1];          // Tableau recevant les données

   readword(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (!etat)
   {
      *production=tabl[0];
   }
   else printf(" problème sur l'équipement n° %hd \n",neq);
}

void traitement_production(time_t *now,int *production,time_t *h_debut,time_t *h_fin,double
*performance,int*depart, int *s_depart)
{//Traitement des ordres de productions
   double diff=0.0;
   if(*depart==1 && *s_depart==0)
      {
         *h_debut=*now+3600;
         *h_fin=*now+3600;
         *s_depart=1;
      }
   else if(*depart==0) *s_depart=0;
   else
   {
      *h_fin=*now+3600;
      diff=*h_fin-*h_debut;
      if(*production>=1) *performance = *production /(diff/3600);
   }
}
```

```
void update_production(int *production,time_t *h_debut,time_t *h_fin,double
*performance,MYSQL *mysql,char *u_host,char *u_user,char *u_passwd,char *u_db)
{//Update des ordres de productions dans la base
   char query[512];
   char r_debut[128];
   char r_fin[128];

   mysql_init(mysql);

   if(mysql_real_connect(mysql,u_host,u_user,u_passwd,u_db,0, NULL, 0)) //Connection à la base
   {
      sprintf(query,"UPDATE ordre_prod SET production =%d",*production);
      mysql_query(mysql,query);

      strftime(r_debut,128,"%H:%M:%S\n",localtime(h_debut));
      sprintf(query,"UPDATE ordre_prod SET h_debut ='%s'",r_debut);
      mysql_query(mysql,query);

      strftime(r_fin,128,"%H:%M:%S\n",localtime(h_fin));
      sprintf(query,"UPDATE ordre_prod SET h_fin ='%s'",r_fin);
      mysql_query(mysql,query);

      sprintf(query,"UPDATE ordre_prod SET performance =%.2f",*performance);
      mysql_query(mysql,query);

      mysql_close(mysql);//deconnection
   }
   else{ // si erreur de connection
      printf("Erreur lors de la connection a la BDD !");
   }
}
```

```
void lecture_depart(int *depart,short etat)
{//Lecture du bit signifiant le demarrage de la ligne
   short nchan = 0;        // Numéro de canal
   short neq = 4;          // Numéro d'équipement
   short nb = 8;           // Nombre de variables
   long adr = 80*8;        // Adresse de la première variable
   short tabl[1];          // Tableau recevant les données
   short t_depart[8];      // Tableau intermediaire

   readpackbit(&nchan, &neq, &nb, &adr, tabl, &etat);
   if (!etat)
   {
      transwordbit(&nb, tabl, t_depart, &etat);
      if(t_depart[0]==1)
      {
         *depart=1;
      }else{*depart=0;}
   }
   else printf(" problème sur l'équipement n° %hd \n",neq);
}
```

6. PROGRAMME PHP

5. Accueil.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css"/>
    <title>Accueil</title>
  </head>

  <body>
  <?php include("entete.php"); ?>


  <!-- Le menu -->
  <nav id="menu">
    <div class="vertical-menu">
      <ul>
        <li><a class="active" href="accueil.php">Accueil</a></li>
        <li><a href="capteurs.php">Capteurs</a></li>
        <li><a href="defauts.php">Defauts</a></li>
        <li><a href="decision.php">Décision</a></li>
      </ul>
    </div>
  </nav>
  <!-- Fin du menu -->

  <!-- Le corps -->
<article>
  <div id="corps">
    <p>
      Cette chaîne de production est composée de trois zones : le magasin, l'hippodrome et les épis.
      Dans le magasin, il y à trois postes de travail : un poste de perçage, un poste de rivetage et un poste de soudure.
      C'est le robot Staubli RX90 qui effectue les tâches de travail sur les pièces.
```

```
        <div style="text-align: center">
        <img src="Chaine.jpg" width="600" height=auto >
       </div>
      </p>
   </div>
</article>
   <!-- Le pied de page -->

<?php include("pied.php"); ?>

   </body>
</html>
```

6. AfficheCapteurs.php

```
<table style="width:1300px" align=center>
<?php
try
{
        // On se connecte à MySQL
        $bdd = new PDO('mysql:host=localhost;dbname=mabdd;charset=utf8', 'root', '');
}
catch(Exception $e)
{
        // En cas d'erreur, on affiche un message et on arrête tout
    die('Erreur : '.$e->getMessage());
}

$data = $bdd->query('SELECT * FROM capteur');
?>
<tr>
<?php
        while ($donnees = $data->fetch())
        {?>
                <th><?php echo $donnees['nom'];?></th>
        <?php   } ?>
</tr>
```

```
<tr>
<?php
        $data = $bdd->query('SELECT * FROM capteur');
        while ($donnees = $data->fetch())
        {
        ?>
                <td align="center">
                <?php
                        if($donnees['valeur']==1)
                                echo '<img src="vert.png" width="20" height="20" >';
                        else {
                                echo '<img src="rouge.png"width="20" height="20"  >';
                         }
                 ?>
                 </td>
                 <?php } ?>
 </tr>
<?php
/*echo'<img src="ligne.png" width="650" height="650" align = middle vspace="-200"
hspace="200">';*/
$data->closeCursor(); // Termine le traitement de la requête
?>
</table>
```

7. Capteurs.php

```
<!DOCTYPE html>
<html>
        <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css"/>
        <title>Defauts</title>
        <script src="refresh" > </script>
        <script>
                var refreshId = setInterval(function()
                {$('#capteur').fadeOut(0).load('AfficheCapteurs.php').fadeIn(0); }, 1000);
         </script>
```

```
        </head>
        <body>
                <?php include("entete.php"); ?>

        <!-- Le menu -->
                <nav id="menu">
                        <div class="vertical-menu">
                        <ul>
                                <li><a href="accueil.php">Accueil</a></li>
                                <li><a class="active" href="capteurs.php">Capteurs</a></li>
                                <li><a href="defauts.php">Defauts</a></li>
                               <li><a href="decision.php">Décision</a></li>
                        </ul>
                        </div>
                </nav>
        <!-- Fin du menu -->
        <article>
         <div id="capteur">
                <table>
                        <td>
                        <!-- Affichage capteurs -->
                        <?php include("AfficheCapteurs.php"); ?>
                        <!-- Fin Affichage capteurs -->
                        </td>
                </table>
        </div>
        </article>
        <!-- Le pied de page -->
        <?php include("pied.php"); ?>
        </body>
</html>
```

8. Corps.php

```
<table id="defaut">
 <thead>
  <tr>
   <th>Nom</th>
   <th>Nombre</th>
   <th>Entre</th>
   <th>Résolution</th>
  </tr>
 </thead>
 <tbody>
 <?php
  try
  {
    $bdd = new PDO('mysql:host=localhost;dbname=mabdd;charset=utf8', 'root', '');
  }
  catch (Exception $e)
  {
   die('Erreur : ' . $e->getMessage());
  }
  $reponse = $bdd->query('SELECT * FROM defauts');
  while($donnees = $reponse->fetch()){
  ?>
  <tr>
   <td><?php echo $donnees['nom']; ?></td>
   <td><?php echo $donnees['nombre']; ?></td>
   <td><?php echo $donnees['entre']; ?></td>
   <td><?php echo $donnees['resolution']; ?></td>
  </tr>
  <?php
  }
  $reponse->closeCursor(); // Termine le traitement de la requête
 ?>
</tbody>
</table>
```

9. Decision.php

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="style.css"/>
  <title>Defauts</title>
  <script src="refresh" > </script>
  <script>
  var refreshId = setInterval(function()
  {
   $('#decision').fadeOut(0).load('decision2.php').fadeIn(0);
  }, 1000);
  </script>
 </head>

 <body>
  <?php include("entete.php"); ?>

  <!-- Le menu -->
  <nav id="menu">
   <div class="vertical-menu">
    <ul>
     <li><a href="accueil.php">Accueil</a></li>
     <li><a href="capteurs.php">Capteurs</a></li>
     <li><a href="defauts.php">Defauts</a></li>
     <li><a class="active" href="decision.php">Décision</a></li>
    </ul>
   </div>
  </nav>
  <!-- Fin du menu -->

  <article id="contenu">
   <div>
    <p> Quelle type de politique voulez-vous adopter ? </p>
    <form method="POST">
```

```
 <input type="radio" name="politique" value=1> Au plus juste<br>
 <input type="radio" name="politique" value=2> Au plus tôt<br><br>
 <input type="submit" value="OK"/>
</form>
<?php //Affichage du formulaire
if(isset($_POST['politique'])){
 $value = $_POST['politique'];

 try
 {
      // On se connecte à MySQL
      $bdd = new PDO('mysql:host=localhost;dbname=mabdd;charset=utf8', 'root', '');
   $sql = "UPDATE politique SET valeur = $value";
   $stmt = $bdd->prepare($sql);
   $stmt->execute();
 }
 catch(Exception $e)
 {
      // En cas d'erreur, on affiche un message et on arrête tout
   die('Erreur : '.$e->getMessage());
 }

 $conn = null;
}

//Affichage de la politique en cours
try
{
    $bdd = new PDO('mysql:host=localhost;dbname=mabdd;charset=utf8', 'root', '');
 $data = $bdd->query('SELECT valeur FROM politique');
}
catch(Exception $e)
{
 die('Erreur : '.$e->getMessage());
}

while ($donnees = $data->fetch())
```

```
 {
 ?>
  <p> <?php echo "La politique choisie est ";
  if ($donnees['valeur']==1) {echo "au plus juste.";}
  else {echo "au plus tôt.";} ?> </p>
 <?php } ?>
</div>
<div>
  <form method="post" action="decision.php">
   <label for="uname">Veuillez choisir le nombre de palettes à produire : </label>
   <input type="text" id="uname" name="name" pattern="[0-9]*" required>
   <input type="submit" name="submit" value="OK">
   <?php
   if(isset($_POST['name'])){
    $value = $_POST['name'];
    try
    {
      // On se connecte à MySQL
      $bdd = new PDO('mysql:host=localhost;dbname=mabdd;charset=utf8', 'root', '');
      $sql = "UPDATE ordre_prod SET ordre = $value, reste = 0";
      $stmt = $bdd->prepare($sql);
      $stmt->execute();
    }
    catch(Exception $e)
    {
      // En cas d'erreur, on affiche un message et on arrête tout
      die('Erreur : '.$e->getMessage());
    }
   }
   ?>
  </form>
  <br><br><br>
 </div>
 <div id="decision">
  <table id="defaut">
   <thead>
    <tr>
```

```
        <th>Objectif</th>
        <th>Reste</th>
        <th>Heure de début</th>
        <th>Heure de fin</th>
        <th>Performance (palette/h)</th>
      </tr>
    </thead>
    <tbody>
      <?php

      try
      {
        $bdd = new PDO('mysql:host=localhost;dbname=mabdd;charset=utf8', 'root', '');
      }
      catch (Exception $e)
      {
       die('Erreur : ' . $e->getMessage());
      }
      $reponse = $bdd->query('SELECT * FROM ordre_prod');
      while($donnees = $reponse->fetch()){
      ?>
       <tr>
        <td><?php echo $donnees['ordre']; ?></td>
        <td><?php echo $donnees['ordre']-$donnees['reste']; ?></td>
        <td><?php echo $donnees['h_debut']; ?></td>
        <td><?php echo $donnees['h_fin']; ?></td>
        <td><?php echo $donnees['performance']; ?></td>
       </tr>
      <?php
      }
      $reponse->closeCursor(); // Termine le traitement de la requête

      ?>
    </tbody>
   </table>
  </div>
</article>
```

```
  <?php include("pied.php"); ?>
 </body>
</html>
```

10. Decision2.php

```
<table id="defaut">
 <thead>
  <tr>
   <th>Objectif</th>
   <th>Reste</th>
   <th>Heure de début</th>
   <th>Heure de fin</th>
   <th>Performance (palette/h)</th>
  </tr>
 </thead>
 <tbody>
  <?php
  try
  {
    $bdd = new PDO('mysql:host=localhost;dbname=mabdd;charset=utf8', 'root', '');
  }
  catch (Exception $e)
  {
   die('Erreur : ' . $e->getMessage());
  }
  $reponse = $bdd->query('SELECT * FROM ordre_prod');
  while($donnees = $reponse->fetch()){
  ?>
   <tr>
    <td><?php echo $donnees['ordre']; ?></td>
    <td><?php echo $donnees['ordre']-$donnees['reste']; ?></td>
    <td><?php echo $donnees['h_debut']; ?></td>
    <td><?php echo $donnees['h_fin']; ?></td>
    <td><?php echo $donnees['performance']; ?></td>
   </tr>
  <?php
```

```
  }
  $reponse->closeCursor(); // Termine le traitement de la requête
  ?>
 </tbody>
</table>
```

11. Defauts.php

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="style.css"/>
  <title>Defauts</title>
  <script src="refresh" > </script>
  <script>
  var refreshId = setInterval(function()
  {
   $('#corps').fadeOut(0).load('corps.php').fadeIn(0);
  }, 1000);
  </script>
 </head>

<body>
  <?php include("entete.php"); ?>

<!-- Le menu -->
<nav id="menu">
  <div class="vertical-menu">
    <ul>
      <li><a href="accueil.php">Accueil</a></li>
      <li><a href="capteurs.php">Capteurs</a></li>
      <li><a class="active" href="defauts.php">Defauts</a></li>
      <li><a href="decision.php">Décision</a></li>
    </ul>
  </div>
</nav>
```

```
<!-- Fin du menu -->

  <!-- Le corps -->
<article>
  <div id="corps">
    <table id="defaut">
     <thead>
      <tr>
       <th>Nom</th>
       <th>Nombre</th>
       <th>Entre</th>
       <th>Résolution</th>
      </tr>
     </thead>
     <tbody>
     <?php
      try
      {
            $bdd = new PDO('mysql:host=localhost;dbname=mabdd;charset=utf8',
'root', '');
      }
      catch (Exception $e)
      {
       die('Erreur : ' . $e->getMessage());
      }
      $reponse = $bdd->query('SELECT * FROM defauts');
      while($donnees = $reponse->fetch()){
      ?>
      <tr>
       <td><?php echo $donnees['nom']; ?></td>
       <td><?php echo $donnees['nombre']; ?></td>
       <td><?php echo $donnees['entre']; ?></td>
       <td><?php echo $donnees['resolution']; ?></td>
      </tr>
      <?php
      }
      $reponse->closeCursor(); // Termine le traitement de la requête
```

```
        ?>
       </tbody>
     </table>
   </div>
</article>
   <!-- Le pied de page -->
   <?php include("pied.php"); ?>

   </body>
</html>
```

12. Entete.php

```
<link rel="stylesheet" href="style.css"/>
<link rel="icon" type="image/png" href="favicon.png" />
<header>
 <body>
   <table align = "center" style="width:1400px">
    <td>
   <p>
    <img class="logo" src="istia.png" alt="Istia" />
   </p>
 </td>
 <td>
    Projet EI4 - Ligne Transitique
</td>
</table>
 </body>
</header>
```

13. Pied.php

```
<link rel="stylesheet" href="style.css"/>
<footer>
<p class="foot">Pierre Baugé - Mathilde Paris - Thibault Poulhalec</p>
</footer>
```

14. Style.css

```
body
{
 background-color: RGB(0,105,178);
 color: white;
 height: 97%;
 position: relative;
}

html
{
 height:100%;
}

.logo
{
 height: 70px;
 width: auto;
}

#corps
{
   display: inline-block;
}

.entete
{
 margin: auto;
 max-width: 90%;
 height: auto;
}

header{
 position: absolute;
 left : 17%;
 top: 0;
```

```
  font-family: fantasy;
  font-size: 80px;
  color: white;
}

.flotte
{
  float: left;
  font-size: 80px;
  text-align: center;
}

p{
  text-align: justify;
  font-size: 20px;
  font-family: fantasy;
}

 article{
   position: absolute;
   left: 17%;
   top : 120px;
   width: 81%;
   margin: 0;
   padding: 0;
 }

 footer
 {
   position: absolute;
   left: 17%;
   color: white;
   width:81%;
   bottom: 25px;
   height: 25px;
 }
```

```
/*Menu*/
 ul {
   list-style-type: none;
   margin: 0;
   padding: 0;
   width: 15%;
   top: 0px;
   left: 0px;
   background-color: #e8edff; /*ou menu*/
   height: 100%; /* Full height */
   position: fixed; /* Make it stick, even on scroll */
   overflow: auto;
}

li a {
   display: block;
   color: #000;
   padding: 8px 16px;
   text-decoration: none;
}

li a.active {
   background-color: RGB(179,20,80);
   color: white;
}

li a:hover:not(.active) {
   background-color: #555;
   color: white;
}
/*Fin Menu*/
/*Pied*/
.foot
{
  text-align: center;
}
/*Fin pied*/
```

```
/*Tableau defaut*/
#defaut
{
 position: absolute;
 left: 50%;
 width: 500px;
 margin-left: -250px;
 font-size: 12px;
 border-collapse: collapse;
 text-align: center;
}

#defaut th
{
 font-size: 13px;
   font-weight: normal;
   padding: 7px;
   background: #b9c9fe;
   border-top: 2px solid #d3ddff;
   border-bottom: 1px solid #fff;
   color: #039;
}

#defaut td
{
   padding: 7px;
   border-bottom: 1px solid #fff;
   color: #669;
   border-top: 1px solid #fff;
   background: #e8edff;
}

#defaut tr td {
   background: #e8edff;
   font-size: 16px;
   color: #99c;
   text-align:center; }
```

```
#defaut tbody tr:hover td {
   background: #d0dafd;
   color: #339;
}
#defaut a:hover {
   text-decoration:underline;
}

/*Fin tableau défaut*/

/*Ordre de production*/
h4{
  text-align: justify;
  font-size: 20px;
  font-family: fantasy;
}
```

ROBOTISATION DE LA LIGNE TRANSITIQUE

Réalisé par : Pierre Baugé – Mathilde Paris – Thibault Poulhalec

Encadré par : Laurent Hardouin

Résumé

La robotisation de la ligne transitique a commencé par l'automatisation de cette ligne ainsi que la gestion de ces défauts. Le projet c'est continué avec la programmation du robot qui suit deux politiques de priorité : au plus tard ou au plus juste. La dernière étape du projet a été la supervision. Elle permet de contrôler la ligne avec un programme C et une page Web.

Mots-clés

Automatisation, Robotisation, Supervision, Programmation, Base de données

Summary

The robotization of the conveying line starts by the automation of this line as well as the management of defaults. The project continue by the programming of the robot who follow two politics of priority: later or at the least. The last step was the supervision. It allows to control the line with a C program and a Web page.

Key words

Automation, Robotization, Supervision, Programming, Data base