Automatisme pour la robotique

TP 2

v0

IUT d'Annecy, 9 rue de l'Arc en Ciel, 74940 Annecy

Configuration du réseau de la baie du robot

Préambule

Pour rappel, l'objectif de ce module est de réaliser l'intégration d'un robot dans une cellule automatisée. Le premier TP a permis de configurer le projet et de réaliser la communication avec l'automate commandant le convoyeur.

Dans ce TP, nous allons réaliser la communication entre l'automate et le contrôleur du robot.

Objectifs:

- Configurer la communication avec la baie CS9.
- Configurer la communication avec le contrôleur M221.

Les systèmes de production modernes intègrent de plus en plus de robots, et différentes structures de commande sont disponibles pour répondre à cette évolution. Le choix de la structure dépend non seulement des compétences de l'intégrateur, mais aussi de l'environnement dans lequel le robot opère. Les contrôleurs de robots actuels, qu'ils soient collaboratifs ou non, proposent des interfaces numériques et analogiques via des cartes additionnelles. Ces contrôleurs supportent aussi diverses technologies de communication, allant des réseaux de terrain traditionnels comme CanOpen ou Profibus DP, aux réseaux Ethernet industriels déterministes tels que EtherCAT, Ethernet/IP, PowerLink ou encore Profinet.

Dans ces configurations, les contrôleurs de robots peuvent être configurés aussi bien comme maîtres que comme esclaves. Dans le cadre de ce TP, nous allons utiliser un automate programmable pour piloter un robot, ce qui nous permet de l'intégrer dans notre cellule robotisée comme un objet industriel connecté. Le contrôleur du robot conserve sa fonction principale de « motion », en gérant les déplacements du robot. Toutefois, le séquencement de ces mouvements sera dicté par l'automate programmable.

La solution UniVALplc développée par Stäubli s'inscrit parfaitement dans cette logique. Ce TP, ainsi que les suivants, se concentrent sur l'intégration d'UniVALplc dans un réseau Ethernet/IP pour permettre la communication entre l'automate et le contrôleur de robot. L'objectif est d'utiliser les blocs fonctionnels pour commander les mouvements du robot et gérer les échanges de données avec les équipements périphériques, tels que la baie CS9 et le contrôleur M221.

1 Présentation de la cellule automatisée

1.1 Matériel

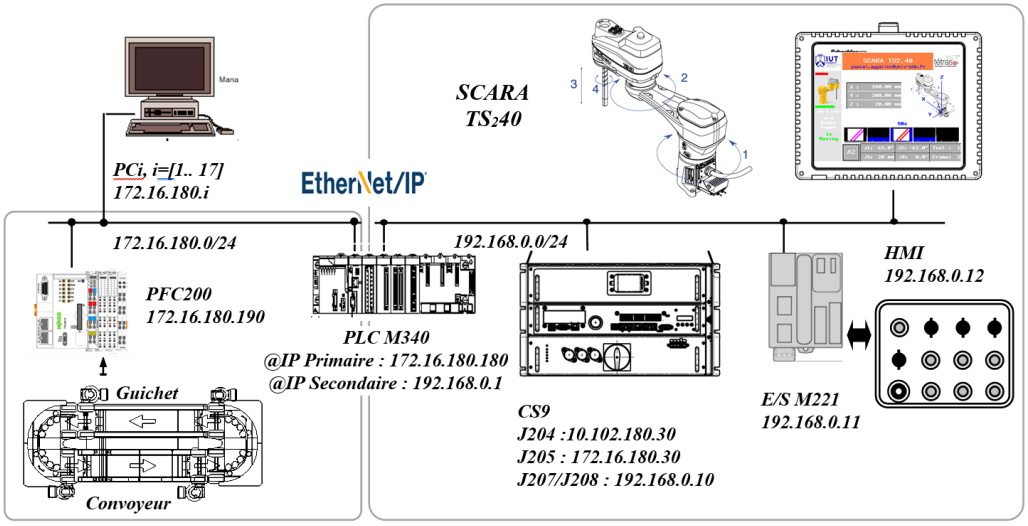

FIGURE 1 – Cellule robotisée

Pour réaliser ce TP, la cellule robotisée, représentée en Figure 1 est composée des éléments suivants :

- **Robot Stäubli Scara TS240** (4 axes, collaboratif) avec son contrôleur **CS9**, équipé d'un pendentif MCP et d'un sélecteur de modes de marche WMS9.
- **Convoyeur TS2plus** de Bosch RexRoth, sur lequel circulent des palettes transportant des produits.
- **Contrôleur PFC200** de WAGO, qui gère partiellement le convoyeur et se comporte comme un îlot d'E/S accessible via Ethernet/IP pour la commande du guichet.
- **Contrôleur M221** de Schneider, utilisé pour la gestion des E/S déportées et la communication avec l'interface HMI via le protocole Modbus-TCP.
- **Pupitre opérateur HMI** permettant de visualiser et superviser les actions du robot et du convoyeur.
- **Automate Schneider M340**, équipé de deux coupleurs réseaux *BMX NOC 0401.2*, chacun permettant, respectivement :
  - ◇ Une communication de type **Ethernet** avec les stations de développement (PC) et le contrôleur PFC200. *Adresse du réseau : 172.16.180.0/24*
  - ◇ Une communication de type **Ethernet/IP** avec la baie CS9 et le contrôleur M221. *Adresse du réseau : 192.168.0.0/24*

2 La solution UniVALplc

2.1 Architecture UniVALplc pour le TP

A propos de UniVALplc

UniVALplc est une solution logicielle développée par Stäubli Robotics pour intégrer les robots de la gamme Stäubli dans des systèmes de production automatisés. Cette solution permet de commander les mouvements du robot et de gérer les échanges de données avec les équipements périphériques.

Architecture UniVALplc

Serveur : Installé sur le contrôleur CS9 du robot. Il répondra aux commandes en provenance de l'automate M340 via le réseau Ethernet/IP.

- Les commandes sont interprétées pour générer des mouvements du robot.
- Les commandes peuvent faire référence à des objets internes du contrôleur CS9 (points, outils, frames, etc.)

Client : Bibliothèque de blocs fonctionnels fournie par le fabricant de l'automate **M340**.

2.2 Présentation du fonctionnement

Les blocs fonctionnels utilisés pour interagir avec le robot se répartissent en deux types :

- Les blocs spécifiques aux contrôleurs Stäubli, identifiables par le préfixe `VAL_`, permettent d'accéder aux fonctionnalités propres à UniVALplc.
- Les blocs conformes à la norme *plcOpen* (préfixe `MC_`), sont utilisés pour commander les mouvements standards, comme la gestion des axes et des positions.

Configuration du serveur

Pour que l'intégration soit réussie, il est nécessaire de respecter l'orientation des données de l'automate client. Dans notre cas, comme il s'agit d'un API M340 de Schneider, l'option *Little Endian* a été configurée lors de l'installation du serveur UniVALplc sur le contrôleur CS9. De plus, pour que les commandes envoyées par l'API soient acceptées, le sélecteur de modes de marche WMS9 du robot doit être positionné sur « Commande à distance ».

2.3 Programmation de l'API

Un objet clé utilisé dans la communication est l'instance du type `T_StaeubliRobot`, qui représente le robot et son contrôleur. Cet objet comprend trois champs principaux :

- `Status` : contient des informations sur l'état actuel du robot et de son contrôleur.
- `Command` : permet de transmettre des commandes au robot.
- `CommInterface` : gère l'interface de communication entre le contrôleur CS9 et l'automate.

Le processus de base à chaque cycle de l'API consiste en trois étapes :

1. Lire l'état du robot à partir d'une mémoire image (T->O) actualisée périodiquement par le réseau Ethernet/IP, en utilisant le bloc fonctionnel `VAL_ReadAxesGroup`.
2. Utiliser les blocs fonctionnels UniVALplc pour commander les mouvements du robot en se basant sur les informations reçues.
3. Écrire les commandes dans la mémoire image (O->T) à l'aide du bloc `VAL_WriteAxesGroup`, afin de les transmettre au contrôleur CS9 via Ethernet/IP.

Ce principe s'applique également pour l'accès aux E/S du contrôleur M221, où la lecture et l'écriture des données se font à travers des services d'assemblage sous Ethernet/IP.

3 Partie TD

3.1 Cartographie du Réseau Ethernet/IP

Lors du premier TP, la configuration du second coupleur BMX NOC 0401.2 (`NOC_ETHIP_ROBOT`) a été réalisée. Son image mémoire commence à l'adresse `%MW64` et comporte 128 mots en entrée ainsi que 128 mots en sortie. Les données produites et consommées par les deux équipements connectés (baie CS9 et contrôleur M221) sont récapitulées dans le tableau ci-dessous :

Équipement	Entrée (T->O) - Données produites	Sortie (O->T) - Données consommées
CS9 (Serveur UniVALplc)	148 octets	124 octets
M221 (Îlot E/S, HMI)	4 octets	40 octets

TABLE 1 – Données échangées sur le réseau Ethernet/IP

Préparation 1: Cartographie de l'espace mémoire %MW

Question 1 À partir des informations du tableau et de l'organisation mémoire du coupleur `BMX NOC 0401.2` (cf. TP n°1), établir la cartographie complète de l'espace mémoire `%MW` de l'automate.

3.1.1 Déclaration des Objets

Le bloc fonctionnel `VAL_ReadAxesGroup` nécessite que les données produites par le serveur UniVALplc soient de type `T_FromRobot`, afin de les convertir en un objet `T_StaeubliRobot`. De même, pour le bloc `VAL_WriteAxesGroup`, les données consommées doivent être de type `T_ToRobot`.

Préparation 2: Déclaration des objets `T_FromRobot` et `T_ToRobot`

Question 2 Utilisez les techniques de chevauchement et d'alias pour déclarer les objets suivants :

- `fromRobotScara` de type `T_FromRobot` (pour les données produites par la baie CS9).
- `toRobotScara` de type `T_ToRobot` (pour les données consommées par la baie CS9).

Pour le contrôleur M221, les blocs fonctionnels `FB_ReadHmi` et `FB_WriteHmi` seront utilisés pour gérer les échanges de données avec l'interface HMI.

Préparation 3: Déclaration des objets `TFromHmi` et `TToHmi`

Question 3 Déclarez les objets suivants :

- `fromHmiScara` de type `TFromHmi` (pour les données produites par le contrôleur M221).
- `toHmiScara` de type `TToHmi` (pour les données consommées par le contrôleur M221).