Institut National Polytechnique de Toulouse

École nationale supérieure d'électrotechnique, d'électronique, d'informatique, d'hydraulique et des télécommunications

Sujet de TER

ATELIER FLEXIBLE

INTRODUCTION

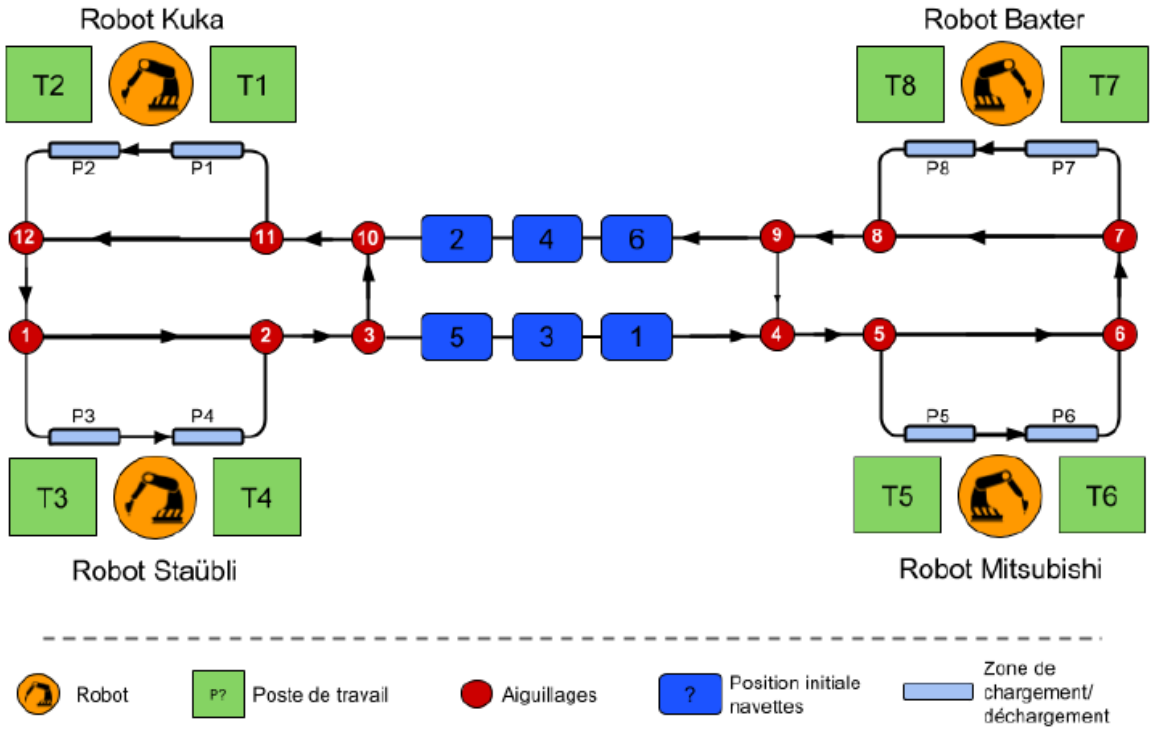
L'objectif de ce TER est d'appliquer les connaissances acquises lors des cours de programmation et de réseau de pétri. Pour cela, vous avez à votre disposition une plateforme de production robotisée constituée d'un réseau de transport monorail, de navettes et de quatre robots.

1. PRESENTATION DE LA PLATEFORME

L'alimentation de la cellule permet de mettre sous tension le rail qui par conséquent entraine la mise en mouvement des navettes. Ces dernières peuvent être contrôlées en commandant les actionneurs se trouvant sur les rails via des automates programmables.

Afin d'éviter tout risque de collision, chaque navette est munie d'un capteur de proximité frontal qui permet de l'arrêter lorsque qu'elle est trop proche d'une autre. Le schéma ci-dessous présente la plateforme:

Figure 1: Platforme



Comme nous pouvons le voir, la cellule est composée de 4 zones de travail (chacune séparée en 2 zones de chargement/déchargement représentées en bleu et 2 postes de travail en vert) sur lesquelles 4 robots réalisent des opérations sur les produits transportés par les navettes. Ces zones sont desservies par des monorails en aluminium sur lesquels les navettes circulent, tout en respectant le sens de circulation indiqué par les flèches noires. Ce sens unique de circulation est imposé par l'alimentation latérale des rails et permet d'éviter les risques de collisions frontales. Le routage est réalisé grâce aux 12 aiguillages présents (A1 à A12 représentés par des cercles rouges sur la figure). Le tout est divisé en 5 zones (ici invisibles), contrôlées chacune par un automate programmable.

2. CAHIER DES CHARGES

L'objectif principal est de réussir à assurer le traitement de plusieurs produits selon leur gamme de production avec un nombre limité de ressources : les robots et les navettes.

A l'aide du fichier de configuration "ProductConfiguration" qui vous a été donné, vous définirez l'ordre et le séquencement du traitement de plusieurs produits. Ce fichier est à lire absolument pour comprendre le déroulement d'une gamme de production. La programmation du réseau de pétri est faite dans un autre fichier qui sera présenté par la suite.

Enfin, la validation se fera par comparaison entre le fichier Log (fichier détaillant tous les événements qui se sont produits lors de la simulation) et le fichier de configuration afin de savoir si le traitement a été effectué de la façon souhaitée.

Dans un premier temps pour faciliter le travail, une approche du système **sans robot** sera réalisée. Le traitement de produits au niveau des postes pourra être remplacé par une temporisation de quelques secondes.

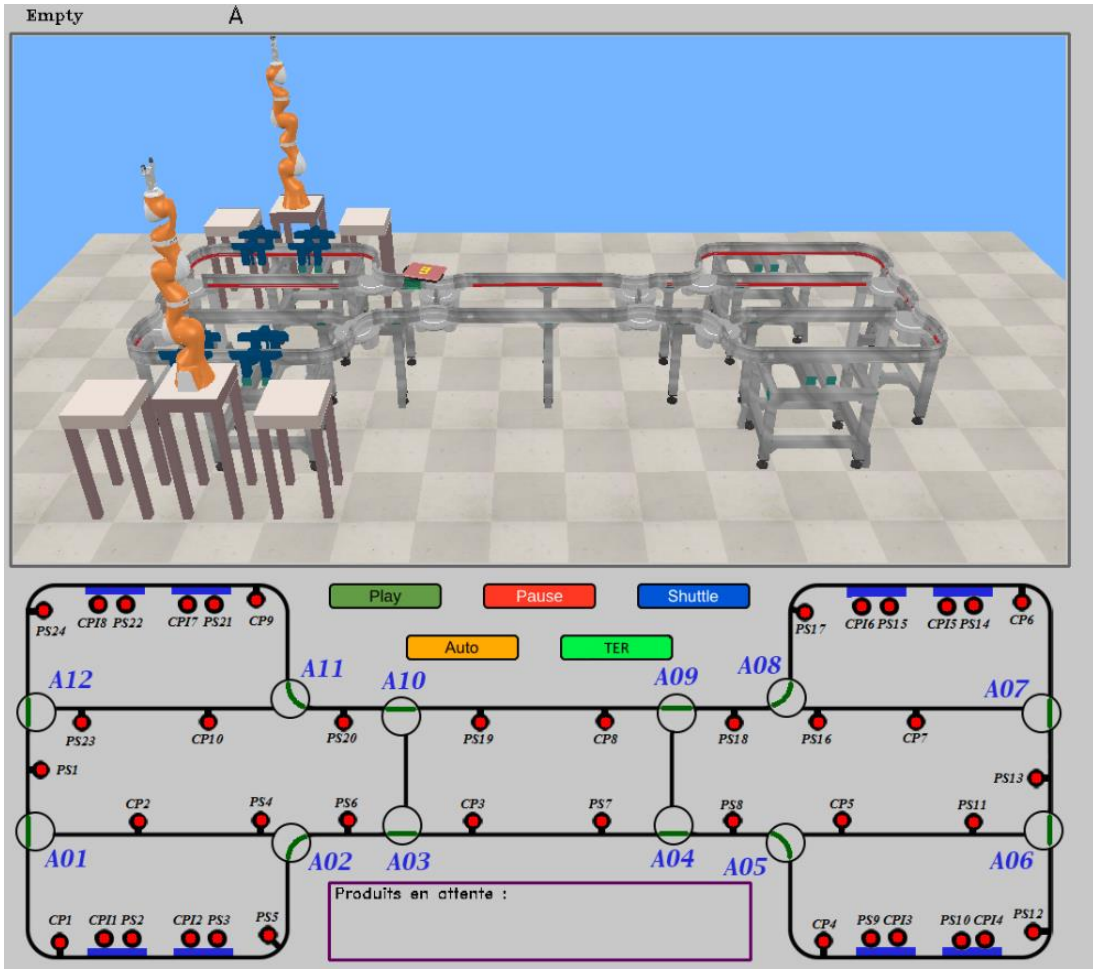
Dans un second temps, les robots seront pris en compte avec leurs fonctions afin de simuler le système global.

La réalisation physique de la plateforme n'étant pas encore opérationnelle, le travail se fera uniquement en simulation. Ainsi, vous programmerez en C++ le réseau de pétri représentatif du cahier des charges.

3. La simulation

La simulation se présente de la façon suivante :

Figure 2: Simulation



Comme sur la figure1 présentant schématiquement la plateforme de travail, les Axx représentent les aiguillages et les Rectangles bleus les zones de chargement/déchargement.

Les voyants rouges représentent les différents capteurs présents sur le monorail:

- Les CPx représentent des capteurs de position qui s'activent lorsqu'une navette est présente.
- Les PSx représentent des capteurs de Stop pouvant arrêter la navette.

Les boutons Play et Pause permettent de respectivement lancer la simulation et de la mettre en pause.

Le bouton Mode permet de sélectionner le mode de fonctionnement de la simulation et le bouton TER permet d'utiliser votre réseau de pétri. Il y a trois modes disponibles. Ils ne diffèrent que par l'initialisation du process de traitement et utilisent tous le code issu du réseau de pétri. Le traitement de produits et le séquencement sont ceux définis dans le fichier de configuration.

- Auto/TER: L'initialisation est faite à partir du fichier de configuration
- Manual: Il active le bouton shuttle qui permet de lancer des navettes. Ces dernières peuvent être vide ou avoir un produit qui est au préalable choisi à partir de la trackbar présente au-dessus de la cellule dans la simulation
- Random: Il choisit aléatoirement le produit présent sur la navette.

Afin de lancer la simulation, il faut :

- Ouvrir un nouveau terminal
- Se positionner dans le repertoire contenant le projet avec "cd Nom_Repertoire"
- Taper "./launch.sh"

4. LA PROGRAMMATION DU RESEAU DE PETRI

La programmation du réseau de pétri se fait dans le fichier "main_commande" qui se trouve dans Repertoire_Projet/ros_ws/src/commande/src.

Des fonctions de haut niveau ont déja été conçues afin de faciliter la programmation du réseau de pétri. Les fonctions dont vous aurez besoin sont les suivantes:

Objet	**Fonctions**
Robots	void EnvoyerRobot(int numRobot, int position)
	void EnvoyerAngles(int numRobot, int angle1..., int angle7)
	void DescendreBras(int numRobot)
	void MonterBras(int numRobot)
	void FermerPince(int numRobot)

	void OuvrirPince(int numRobot)
	void ControlerRobot(int Position, int numPosition, int bras, int pince)
	int RobotInitialise(int numRobot)
	int RobotEnPosition(int numRobot)
	int BrasEnPosition(int numRobot)
	int PinceEnPosition(int numRobot)
	int TraitementFini(int numTache)
Produits	void PiecePrise (int numPoste)
	void PieceDeposee(int numPoste)
	int ProduitSurNavette(int handle)
Navettes	int NouvelleNavette()
	void ReinitialiserNouvelleNavette
	int NavetteStoppeeVide(int numPoste)
	int NavetteStoppee(int numPoste)
	void NavettePartie(int numPoste)
	void DefinirDestination(int handle, int destination)
	int NavetteDisponible()
	void DestroyShuttle(int numNavette)

L'explication détaillée de ces fonctions se trouve dans la partie annexe à la fin de ce document.

A ce stade, vous êtes capable de commencer le projet.

ANNEXE

Lors de ce TER, vous avez la possibilité de commander les différents composants de la simulation mise à votre disposition.

LES ROBOTS :

Les 4 robots industriels présents dans cette simulation sont des robots 6 axes de type KUKA

LBR IIWA 14 R820. Les fonctions listées ci-dessous vous permettront de contrôler chacun de ces robots. Les arguments de ces fonctions pourront être choisis à l'aide du schéma récapitulatif de la cellule flexible.

- **EnvoyerPosition(numRobot, numPosition) :**

4 positions ont été prédéfinies pour chacun des robots : les deux zones de chargement/déchargement et les deux postes de travail. Pour y accéder, l'argument numRobot permet de choisir le robot que vous souhaitez contrôler et l'argument numPosition identifie la position dans laquelle le robot choisi sera envoyé. Chacun des arguments est un entier compris 1 et 4 (cf schéma récapitulatif).

- **EnvoyerAngles(numRobot, angle1, angle2, angle3, angle4, angle5, angle6, angle7) :**

Si vous souhaitez envoyer un robot dans une autre position que celles prédéfinies, l'argument numRobot vous permet de sélectionner ce robot. Les arguments angle1 à angle6 sont les angles en degrés des différents axes du robot et l'argument angle7 permet de faire pivoter la pince.

Par exemple, la position prédéfinie n°1 d'un robot est obtenue à l'aide des angles : angle1 = 128 ; angle2 = 90 ; angle3 = 90 ; angle4 = 80 ; angle5 = 90 ; angle6 = -90 ; angle7 = -40.

- **DescendreBras(numRobot) et MonterBras(numRobot) :**

Ces fonctions vont vous permettre de descendre ou de monter le bras du robot numRobot suivant l'axe vertical. Pour utiliser ces fonctions, veillez à être dans une des positions prédéfinies du robot.

- **OuvrirPince(numRobot) et FermerPince(numRobot) :**

Ces fonctions permettent d'ouvrir ou de fermer la pince du robot défini par numRobot.

- **ControlerRobot(numRobot, numPosition, etatBras, etatPince) :**

Cette fonction vous permettra d'envoyer le robot numRobot dans une des positions prédéfinies, tout en plaçant le bras dans l'état souhaité du bras ainsi que de la pince. L'argument numPosition vous permet de choisir la position prédéfinie. Le bras sera placé dans l'état bas si l'argument etatBras vaut -1 et dans l'état haut si l'argument a pour valeur 1. L'argument etatPince permet de définir l'état souhaité de la pince, en le fixant à la valeur -1 la pince sera ouverte alors que la valeur 1 fera ouvrir la pince. Les actions se dérouleront de manière successive.

Les fonctions listées ci-dessus permettent de contrôler les différents mouvements possibles des robots. Les fonctions présentées dans la suite de ce document permettent d'avoir un retour de ces robots et de savoir si les mouvements ont bien été réalisés.

- **RobotInitialise(numRobot) :**

Cette fonction retourne 1 si le robot numRobot a été correctement initialisé. Le robot numRobot ne pourra pas être contrôlé s'il n'est pas initialisé.

- **RobotEnPosition(numRobot) :**

Cette fonction retourne 1 lorsque le mouvement précédemment commandé au robot numRobot est terminé.

- **BrasEnPosition(numRobot) :**

Cette fonction retourne 1 si le bras du robot numRobot est en position haut, -1 s'il est descendu.

- **PinceEnPosition(numRobot) :**

Cette fonction retourne 1 si la pince du robot numRobot est fermée, -1 si la pince est ouverte.

- **TraitementFini(numTache) :**

Cette fonction retourne 1 lorsque le poste numTache a fini le traitement de son produit.

Les Navettes

- **NouvelleNavette() :**

Cette fonction retourne le numéro de la navette qui vient d'être créée par le mode automatique ou de manière manuelle.

- **ReinitialiserNouvelleNavette() :**

Cette fonction permet de réinitialiser la variable qui permet de gérer l'arrivée des navettes.

- **NavetteStoppeeVide(numPoste) :**

Cette fonction renvoie le numéro de la navette stoppée au niveau de la zone de chargement/déchargement numPoste si cette navette est vide, 0 sinon.

- **NavetteStoppee(numPoste) :**

Cette fonction renvoie le numéro de la navette stoppée au niveau de la zone de chargement/déchargement numPoste si cette navette contient un produit, 0 sinon.

- **NavettePartie(numPoste) :**

Cette fonction doit être appelée lorsqu'une navette part de la zone de chargement/déchargement numPoste pour réinitialiser les variables associées à la présence d'une navette.

- **DefinirDestination(numNavette, nDestination) :**

Cette fonction permet de définir la destination suivante nDestination de la navette numNavette.

- **NavetteDisponible() :**

Cette fonction retourne le numéro d'une navette vide circulant dans la partie centrale D0 de la cellule.

Les Produits

- **ProduitSurNavette(numNavette) :**

Cette fonction renvoie la prochaine destination du produit se trouvant sur la navette ayant pour handle numNavette. Cette valeur est comprise entre 1 et 4 pour les quatre postes disponibles. Cependant, si la navette ne transporte aucun produit, la fonction ProduitSurNavette renvoie 0.

- **PiecePrise(numPoste) :**

Cette fonction signale à la zone de chargement/déchargement numPoste qu'un produit a été récupéré par le robot associé et fait repartir la navette. On rappelle que le bras du robot doit être en position haute avant de redémarrer une navette.

- **PieceDeposee(numPoste) :**

Cette fonction signale à la zone de chargement/déchargement numPoste qu'un produit a été déposé par le robot associé et fait repartir la navette. On rappelle que le bras du robot doit être en position haute avant de redémarrer une navette.