Projet Long N7 session 2020

Sujet

TER atelier flexible

Team SALLAG
Promo 2020

Tuteur : Mme U. Ngueveu Sandra
et M Briand Cyril

27 Janvier 2020 — 6 Mars 2020

Table des matières

1 Introduction

L'objectif de ce TER est d'appliquer les connaissances acquises lors des cours de programmation et de réseau de Petri. 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.

2 Présentation de la plateforme

L'alimentation de la cellule permet de mettre sous tension le rail qui par conséquent entraîne 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 lorsqu'elle est trop proche d'un obstacle. Le schéma ci-dessous présente la plateforme :

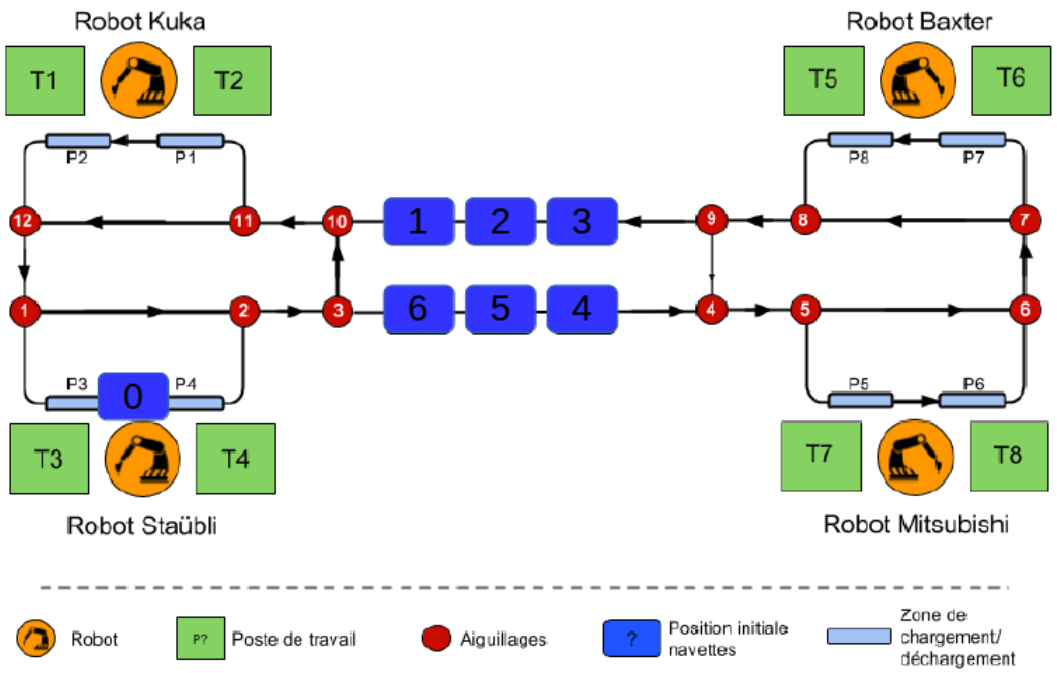

Figure 1 – Schéma général de la maquette

Comme nous pouvons le voir, la cellule est composée de 4 zones de travail, chacune composée de 2 zones de chargement/déchargement (représentées en bleu clair) et 2 postes de travail (représentés en vert) sur lesquelles les 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.

3 Présentation de la simulation

La simulation donne accès aux capteurs et actionneurs suivants :

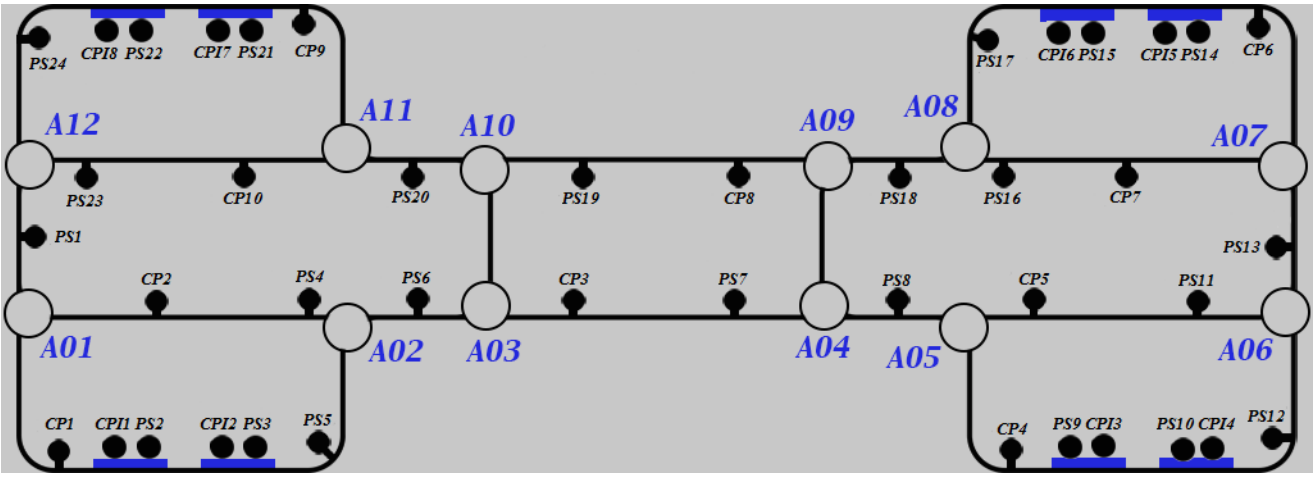

FIGURE 2 – Schéma récapitulatif des capteurs et actionneurs de la cellule

Les aiguillages sont numérotés de A01 à A12 et possèdent deux positions (droite et gauche, reconnaissable dans le sens de la marche). Les rectangles bleus représentent les zones de chargement/déchargement (postes). Le schéma ci-dessus représente également les différents capteurs présents sur le monorail : Les CPx et CPIx sont des capteurs de position qui s'activent lorsqu'une navette est présente. Les PSx sont des capteurs de stop pouvant arrêter la navette et font aussi office de capteur de position. Pour lancer la simulation, il suffit d'exécuter le launch.sh présent à la racine du projet. Il suffit de se placer dans ros_ws/ et d'exécuter catkin_make puis source devel.setup.bash.

4 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. Enfin, la validation se fera par passage du fichier log.txt (fichier détaillant tous les événements qui se sont produits lors de la simulation) dans un "checker" qui vérifiera que la production s'est bien passée. Dans un premier temps, pour faciliter le travail, une approche du système sans robot et sans produit sera réalisée, pour prendre en main l'utilisation des navettes. Dans un second temps, il vous est donné la possibilité d'ajouter des produits sur des postes. Ces produits sont représentés par des petits cubes de couleurs. Les bras robots peuvent être utilisés pour déplacer les produits depuis un poste vers une navette, et inversement. De plus, lorsqu'un

produit est sur un poste, le robot peut effectuer une tâche dessus. Cette tâche est représentée par l'ajout d'un cube (de couleur différente selon le poste) sur le produit. Lorsque la tâche est en cours, le cube est semi-transparent et se colore entièrement lorsque la tâche est finie. Les tables ci-dessous résument les couleurs des cubes "produits" et "tâches" :

Produit	**Couleur**	**Poste**	**Couleur**
1	Rouge	1	Rouge
2	Bleu	2	Bleu
3	Vert	3	Vert
4	Orange	4	Orange
5	Rose	5	Rose
6	Jaune	6	Jaune
-	-	7	Blanc
-	-	8	Noir

Vous programmerez en C++ un réseau de Petri permettant de réaliser la production d'un certain nombre de produit, demandé par l'encadrant/professeur.

5 Réseau de Petri

La programmation du réseau de Petri se fera dans le fichier "main_commande.cpp" qui se trouve dans /ros_ws/src/commande/src. Des fonctions de haut niveau ont déjà été conçues afin de piloter les différents actionneurs de la simulation. Les fonctions à votre disposition sont les suivantes :

Objet	Fonction
Commande cmd	Stop_PS(num_capteur_PS) Ouvrir_PS(num_capteur_PS)
RobotsInterface robot	DeplacerPiece(num_robot, positionA, positionB) FaireTache(num_poste, duree) Evacuer() AjouterProduit(num_poste, num_produit) int TacheFinie(num_poste)
AigsInterface aiguillage	Droite(num_aiguillage) Gauche(num_aiguillage)
Capteurs capteur	bool get_PS(num_PS) bool get_CP(num_CP) bool get_CPI(num_CPI) bool get_DD(num_aiguillage) bool get_DG(num_aiguillage)

Description individuelle des fonctions haut niveaux :

- **Stop_PS** : Permet d'arrêter une navette un niveau d'un actionneur PS du circuit. À activer avant le passage de la navette sur le capteur en question pour plus de robustesse.
- **Ouvrir_PS** : Débloque le passage au niveau d'un actionneur PS.
- **Deplacer_Piece** : Réalise successivement les actions suivantes : déplacer le bras du robot en position A, descendre bras, fermer pince, lever bras, déplacer bras en position B, descendre bras, ouvrir pince, remonter bras.
- **FaireTache** : Rajoute un cube sur le produit au poste num_poste en un certain temps durée
- **Evacuer** : Fait disparaître la pièce sur le poste 3. C'est la seule manière de faire sortir une pièce usinée de la simulation.
- **AjouterProduit** : Fait apparaître un produit de type num_produit sur le poste num_poste.
- **TacheFinie** : Teste si la tâche au poste num_poste est finie.
- **Droite** : Positionne l'aiguillage num_aiguillage à droite dans le sens de la marche.
- **Gauche** : Contre toute attente, positionne l'aiguillage à gauche.
- **get_PS** : Retourne vrai si une navette est présente sur le capteur points de stop (PS).
- **get_CP** : Retourne vrai si une navette est présente sur le capteur de position (CP).
- **get_CPI** : Retourne vrai si une navette est présente sur le capteur de position (CPI).
- **get_DD** : Retourne vrai si l'aiguillage est en position droite.
- **get_DG** : À la surprise générale, retourne vrai si l'aiguillage est en position gauche.
  Remarque : Pour le bon fonctionnement de la simulation, il existe un programme interne (dont vous n'avez pas accès) permettant de suivre les navettes. Ce programme fonctionne correctement à condition que vous orientez assez tôt les aiguillages avant le passage d'une navette. Nous vous invitons donc à ne pas changer les aiguillages trop tardivement dans votre scénario.

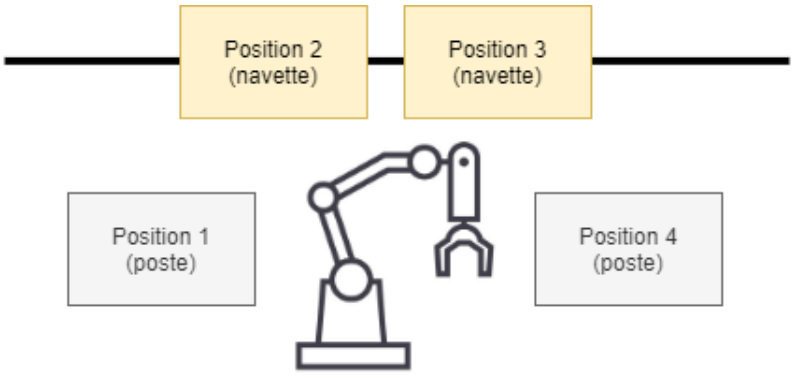

FIGURE 3 – Numérotation des différentes positions d'un robot

6 Utilisation du checker

Au cours de la simulation, un fichier log.txt est créé à la racine du projet. Ce fichier est complété automatiquement à chaque action notable (arrivée d'un produit, évacuation, tâches…). En fin de simulation, vous passerez ce fichier log.txt dans un checker qui vous retournera les informations suivantes :

- Si la production est conforme au cahier des charges inscrit dans le fichier ProductConfiguration.config ;
- Si il a détecté une incohérence dans votre production (écrasement de produit, tâche sur aucun produit, tâche non finie, ...) ;
- Des statistiques sur le temps passé des produits dans l'atelier de production.

Le fichier ProductConfiguration.config permet de configurer les produits qui doivent être fabriqués. Le fichier doit être remplit comme suivant :

Produit (1 à 6) : Tâches (plusieurs tache possible au choix entre 1 et 8) : Durée des tâches : Nombre de produit identique

Prenons un exemple simple, je fais le produit 1 sur lequel j'applique la tache 7 puis 5 avec chacune une durée de 2 secondes, je dois répéter ce produit 3 fois. On obtient donc

Produit : tache1 tache2 : duréetache1 duréetache2 : répétition du produit
1 : 7 5 : 2 2 : 3