ÉCOLE DE TECHNOLOGIE SUPÉRIEURE DÉPARTEMENT DE GÉNIE DE LA PRODUCTION AUTOMATISÉE	**Enseignant :** Ahmed Joubair Ing., Ph.D. **Chargé des laboratoires :** Ahmed Joubair Ing., Ph.D. **Trimestre: Été** 2024

GPA-546 ROBOTS INDUSTRIELS

PROJET #2 : Simulation d'une tâche de palettisation

Mise à jour (Hiver 2024, par : Ahmed Joubair Ing. Ph.D.)

1. Objectifs

- Être capable d'opérer un robot ABB IR1600 en mode manuel.
- Être capable de faire la programmation de base d'un robot ABB.
- Être capable d'utiliser les référentiels « objet » (*wobjdata*).

2. Description de la tâche à accomplir

Description générale. Ce projet consiste à programmer un robot ABB pour simuler une tâche de palettisation. Les pièces en acier sont initialement disposées dans un magasin (une glissoire) situé sur une table. La présence et l'orientation (trou à gauche ou à droite) d'une pièce est détectée à l'aide de deux capteurs situés en bas du magasin (DI09_ZS0101 et DI10_ZS0102). Un vérin pneumatique (DO09_FV0101) pousse la pièce pour permettre sa préhension avec la pince du robot. Une palette remplie contient *N* blocs déposés à plat et formant une étoile régulière autour d'un point centre (figure 1). Les blocs doivent être déposés avec les trous circulaires orientés vers l'extérieur. La valeur de *N* peut varier de trois à sept. Les blocs doivent être à une distance constante les uns par rapport aux autres.

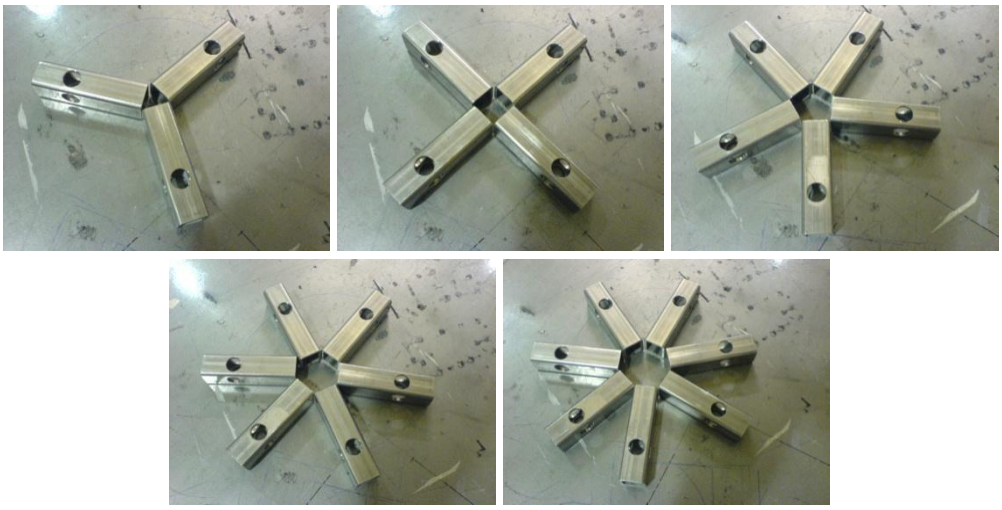

Figure 1 : Les cinq configurations de blocs possibles.

Séquence de travail. Le programme remplit une palette en effectuant la séquence suivante :

a) Avant d'effectuer ses mouvements, le robot doit faire les initialisations appropriées (paramètres de mouvement par défaut pour un mouvement rapide, etc.).

b) Le robot doit s'assurer qu'il est à moins de 50 mm du robtarget *rRetrait*. Si la validation est réussie, alors le système allume pendant 2 secondes la lampe bleue et passe au point suivant sans attendre la fin des deux secondes. Si la validation échoue, le programme affiche un message avec le détail du problème (incluant la valeur de la distance actuelle du robot par rapport à *rRetrait* et l'unité de la distance) et allume la lampe orange pendant 5 secondes. Ensuite, le programme se termine de lui-même (le programme ne recommence pas si le robot est en mode *Continuous Cycle*).

c) Le programme demande le nombre de branches à la forme en affichant un menu semblable au menu présenté à la figure 2.

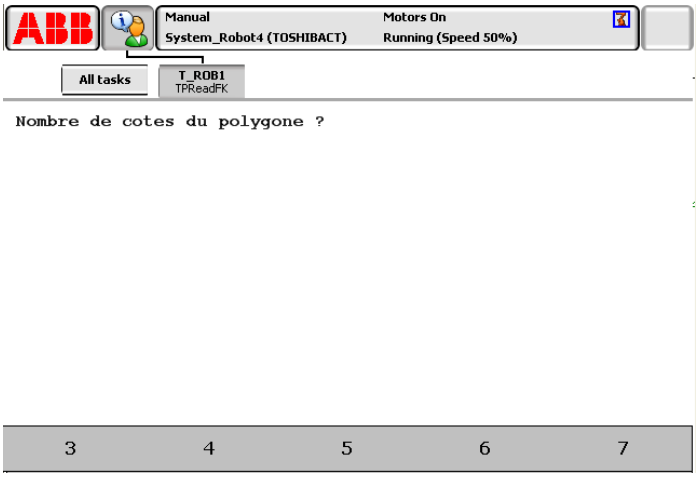

Figure 2 : Un exemple de menu (commande TPReadFK).

d) Le programme effectue les calculs du prochain robtarget selon le nombre de branches choisi.

e) Le robot passe par un point d'approche et de dégagement lors de la prise dans la glissoire et lors du dépôt sur la feuille. Pour pouvoir faire une prise d'un bloc, le robot doit détecter la présence de ce bloc dans la chute avant de démarrer la séquence de prise, sinon le robot attend le bloc. Si l'attente est de plus de 5 secondes, le robot retourne à *rRetrait*, affiche un message d'avertissement et continu l'attente nécessaire.

***Important :** la pince doit toujours prendre le bloc avec les suces du côté du trou sur le bloc, sinon il y aura collision avec les autres blocs lors du dépôt.

f) Le robot dépose le premier bloc sur la feuille (robtrarget *rBloc1*) en le mettant parallèle aux axes du référentiel de la feuille et de telle façon que le centre de l'étoile coïncidera avec le centre de la feuille. Ensuite, le robot répète l'étape d) et dépose le bloc suivant.

g) Après que le dernier bloc a été placé sur la table, le robot retourne au robtarget *jRetrait* (c'est **jointtarget**) en utilisant la commande *MoveAbsJ* et le programme s'arrête de lui-même (le programme ne recommence pas s'il le robot est en mode *Continuous Cycle*).

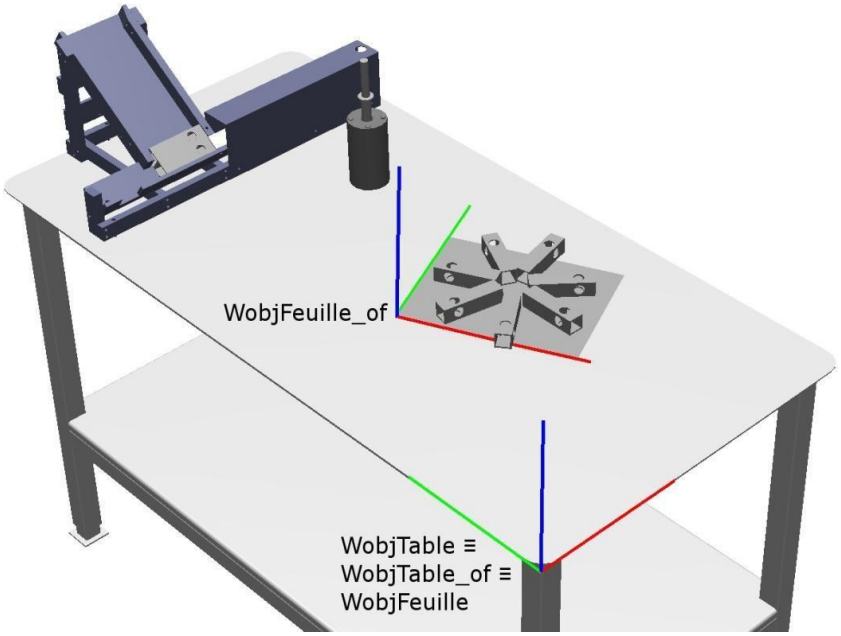

Figure 3: Référentiels objet WobjTable et WobjFeuille (« _of » signifie « object frame »)

Avant d'exécuter le programme. Le programme utilise quatre persistantes (*PERS*) de type wobjdata ou robtarget, enseignées avec le boîtier de commande :

- *WobjTable*, qui est un système de coordonnées de type wobjdata utilisé pour la table;
- *WobjFeuille*, qui est un système de coordonnées de type wobjdata utilisé pour la table et la feuille (figure 3),
- *rRetrait*, représente l'endroit où le robot est dégagé, soit environ 60 cm au-dessus du centre de la table avec une pince qui pointe vers le sol.
- *rGlissoire* qui représente l'endroit où le robot prend la pièce dans le magasin.

Pour définir le référentiel objet, vous devez utiliser la méthode « définition par trois points » à l'aide du FlexPendant. Ces localisations enregistrées doivent s'enseigner rapidement et efficacement par un opérateur. Vous devez tenir compte que la feuille ou la table peuvent bouger entre deux sessions de travail. Voilà pourquoi, vous devez enseigner le robtarget *rGlissoire* par rapport au système *WobjTable* (où *WobjTable.oFrame* := [[0,0,0],[1,0,0,0]]). Les poses des blocs sont calculées par le programme par rapport au référentiel *WobjFeuille.oframe* qui est situé sur la feuille, lequel se rapporte au référentiel utilisateur *WobjFeuille.uFrame* sur le coin de la table qui se rapporte à son tour au référentiel de l'atelier (*wobj0*). Les référentiels *uFrame* des deux workobjets coïncident (*WobjTable.uframe* := *WobjFeuille.uframe*). Vous ne devez pas assumer que l'axe *z* du référentiel de l'atelier est perpendiculaire à la surface de la table.

En tant que spécialiste en robotique, vous devez :

- Respecter les consignes de sécurité émises au laboratoire #1.
- Faire en sorte que l'opérateur ait un minimum de travail à faire pour enseigner les trois objets.
- Minimiser le temps d'exécution du programme tout en respectant les contraintes suivantes :
    1) Vitesses d'approche et de retrait : Lowspeed;
    2) Distances d'approche et de retrait de 200 mm;
    3) Interpolation maximale de 10 cm;
    4) Le programme gère certaines conditions limites : manque de pièce, mauvais choix entré par l'opérateur, etc.

Les fonctions RAPID suivantes peuvent être utiles : AliasIO, CPos, CRobT, MoveAbsJ DefFrame, Distance, Offs, OrientZYX, RelTool, SetDO, TPErase, TPReadFK, TPReadNum, TPWrite, TestDI, WaitDI, WaitTime, WaitUntil, wobjdata. Vous devez examiner le manuel de référence technique « Instructions, fonctions et types de données RAPID » dont une copie électronique est disponible dans la section Documents du site web du cours.

3. Fonctionnement

La durée du projet est de quatre séances (avec présence d'un chargé de laboratoire). La première séance est consacrée à écrire un programme sans erreurs de syntaxe avec le logiciel de simulation RobotStudio (au local A-3240). S'inspirant d'un contexte industriel, la programmation hors-ligne permet d'éviter l'arrêt de la production pour écrire le programme. **Avant le début de la deuxième séance, vous devez remettre un programme fonctionnel (testé en simulation) avec les points a), b) et c) complétés.** Tant que vous n'avez pas remis ce programme et obtenu l'approbation écrite de votre chargé de laboratoire, l'utilisation de votre robot vous est interdite. La troisième séance permet de mettre au point votre programme. À la quatrième séance, vous devez faire une démo et **remettre votre programme final avant le début de ladite période.** Aucune extension ne sera accordée sous prétexte que votre séance (par exemple, mercredi PM) est plus tôt que la séance d'une autre équipe (jeudi PM).

Points importants à considérer :

- Vous pouvez seulement enseigner les 4 éléments suivants : wobjFeuille, rGlissoire, rRetrait et jRetrait. Les positions de dépôt des blocs doivent absolument être calculées.
- Vous devez utiliser des noms appropriés et significatifs pour les entrées et les sorties (exemple : DI10_ZS0102 = DI_BlocTrouAGauche). Vous ne devez pas utiliser les noms par défaut.
- L'opérateur doit avoir le temps de lire les messages qui lui sont envoyés. Il doit confirmer la lecture du message par l'appui d'une touche.
- Vous devez activer et désactiver les configurations des moteurs seulement lorsque c'est nécessaire.
- Votre programme doit être concis avec des noms de variables significatifs. Vous ne devez pas répéter des lignes de code inutilement.
- Lorsque vous interagissez avec l'opérateur, vous ne devez pas faire de fautes de français.
- La mise en page de votre programme doit être bien réalisée (respect des indentations, pas d'espaces inutiles entre les lignes de code, constance des déclarations et des expressions).
- Vos procédures et vos fonctions doivent accomplir une seule tâche. Par exemple, vous ne devez pas avoir une fonction qui s'appelle CalculerVolumeSphere qui calcule le volume d'une sphère et qui

déplace le robot en même temps. Elle doit seulement faire la tâche que son nom indique. Vous devez bien découper votre programme en sous-fonctions.

- Vous devez respecter vos propres conventions de programmation.

Consignes pour la transmission électronique de votre programme sur Moodle. (Voir Moodle)

GPA-546 ROBOTS INDUSTRIELS
Barème de correction pour le laboratoire 2

Approbation /23 points

a) Routine d'initialisation (1)
b) Sécurité : Pas de déplacement avant la validation (3), message de sécurité approprié incluant la valeur de la distance (3), gestion si la validation réussie est fonctionnelle (3), gestion si la validation échouée est fonctionnelle (3) le message en cas d'échec demande l'approbation de l'opérateur (3), le programme termine son exécution si la sécurité échoue (2)
Affichage de la distance (2) et de **l'unité** de la distance (1), en cas d'erreur.
c) Menu : sécuritaire et fonctionnel, gestion des choix invalides si nécessaire (3)

Démonstration sur le robot /48 points

d) Calculs des *robtargets* itératifs (3)
e) Prise : sans accrochage et centrée (3), trou du côté des suces (4), distance d'approche et de retrait respectée (2), gestion si manque de blocs : attente d'un maximum de 5 secondes (2), retour à retrait (1), message approprié (2), retour à la glissoire lorsque présence d'un bloc (1).
f) Dépôt : espace constant entre les blocs (2), 1er bloc déposé au bon endroit (3), trous orientés dans le bon sens (4), toutes les formes fonctionnelles (4), sans accrochages (3).
g) Le programme se termine de lui-même (2)

- Respect des vitesses (2) et des interpolations maximales demandées (2)
- Optimisation des mouvements, donc pas de mouvements inutiles (3)
- Redémarrage rapide avec enseignement d'un seul *workobject* (5)
- Utilisation de ***jRetrait*** en utilisant la commande ***MoveAbsJ*** (2)

Programmation /19 points

- Les fautes de français lors de l'interaction avec l'opérateur 2)
- Alias appropriés pour les entrées et les sorties (4)
- Noms significatifs pour les variables et les routines (3)
- Respect des indentations (2)
- Commentaires pertinents et complets (3)
- Segmentation du programme (2)
- Seulement des appels de routine dans le *main*, sans calculs intermédiaires (3)

Note pour le laboratoire **/90 points**