

# L3ProAII Régulation Industrielle - TP Régulation de temperature de l'eau d'un bac.

BOUFADENE Mehdi COUTAUD Ulysse

*<2022-03-30 Wed>*

## Contents

<b>1</b>	<b>Prise en main et première simulation de picking</b>	<b>2</b>
1.1	Mise en place d'une scène basique . . . . .	2
1.2	Programme et simulation d'un programme de picking simple .	4
<b>2</b>	<b>Ajout d'un effecteur</b>	<b>7</b>
2.1	Mise en place d'un outil de préhension . . . . .	7
2.2	Programmation de la commande du préhenseur . . . . .	7
2.2.1	Lier la pinces aux E/S sorties du robot . . . . .	7
2.2.2	Programmer l'ouverture et la fermeture de la pince . .	9
<b>3</b>	<b>Programmation de tâches conditionnelles</b>	<b>13</b>
3.1	Adaptation de l'environnement . . . . .	13
3.2	Implémentation des sous-programmes de picking . . . . .	14
3.3	Implémentation du programme de manutention conditionnelle .	15
<b>4</b>	<b>Programmation d'un second robot et coordination avec le premier</b>	<b>15</b>
	Sources disponibles sur <a href="https://github.com/coutaudu/TP_Robotique/tree/master">https://github.com/coutaudu/TP_Robotique/tree/master</a> .	

# 1 Prise en main et première simulation de picking

## 1.1 Mise en place d'une scène basique

- Nous avons mis en place une scène basique pour un premier scénario de simulation d'un robot.
- Voir figure1.

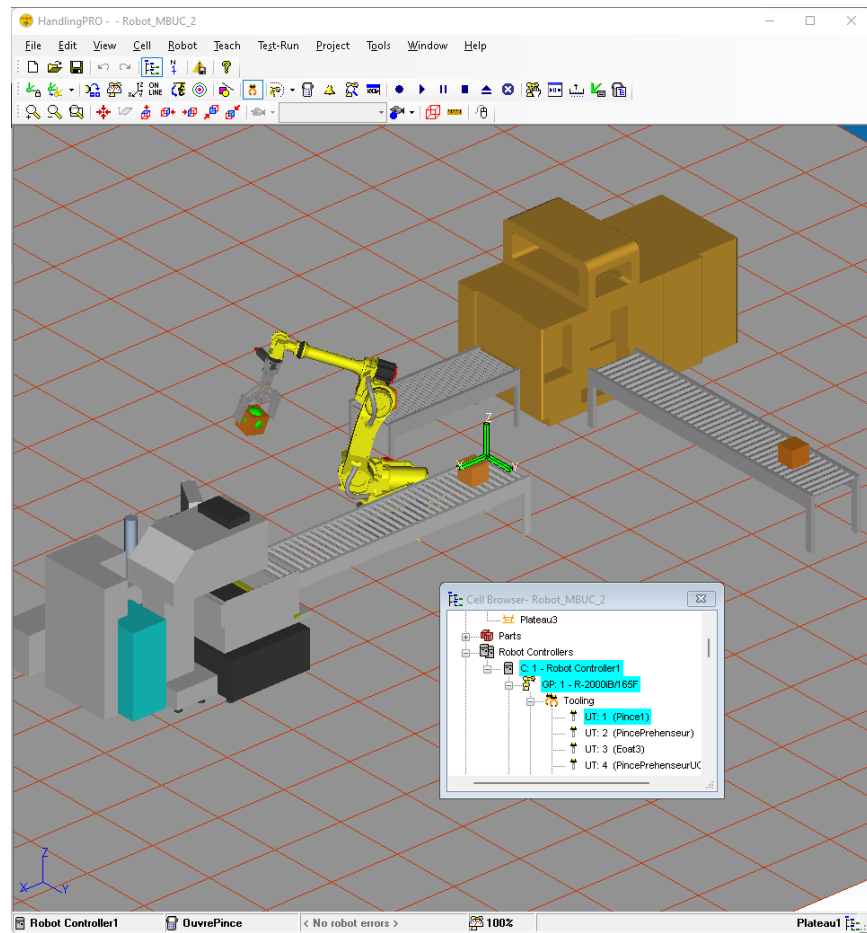


Figure 1:

- Le robot placé au centre est équipé d'une pince non-commandable.

- Le robot est entouré par 2 plateaux qui seront les zones de prise et de dépose des objets.
- Nous avons positionné le repère UTool de la pince installé sur le robot en extrémité de ces pinces.
- Les zones de dépôt sont bien dans la zone de travail du robot équipé de cette pince et vis à vis du point de repère Utool de cette pince, voir fig.2.

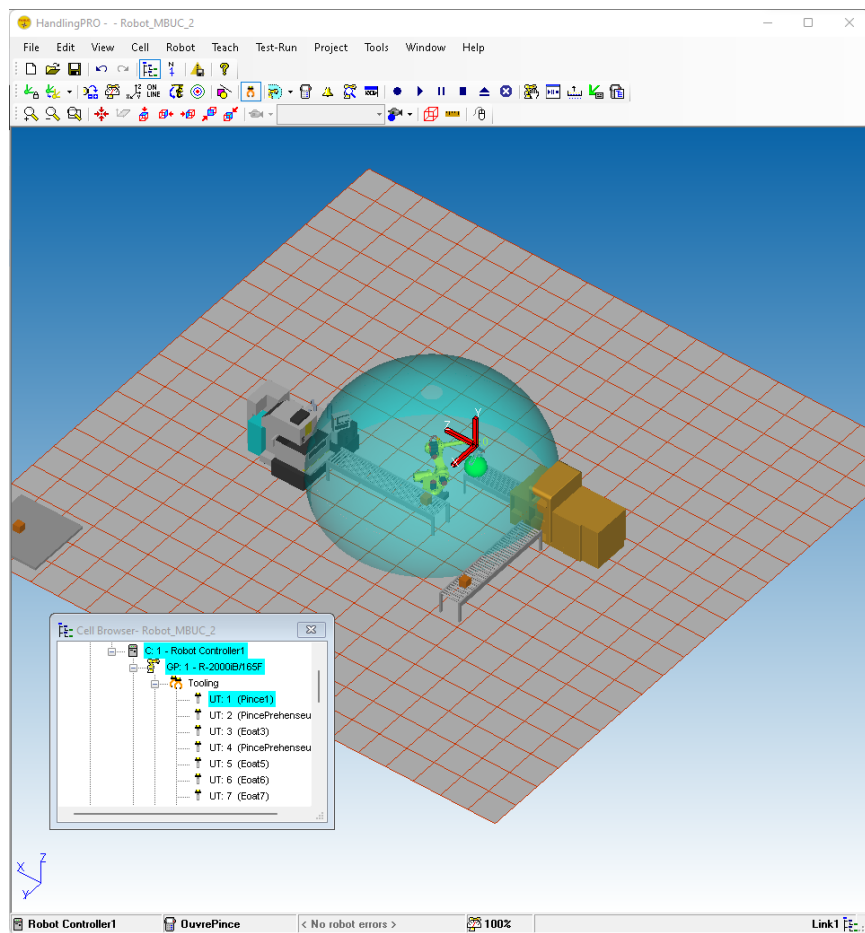


Figure 2:

- Nous avons positionné 2 boîtes sur les plateaux et avons paramétré leur orientation de manière à ce que le robot puisse se positionner pour les attraper par le haut.

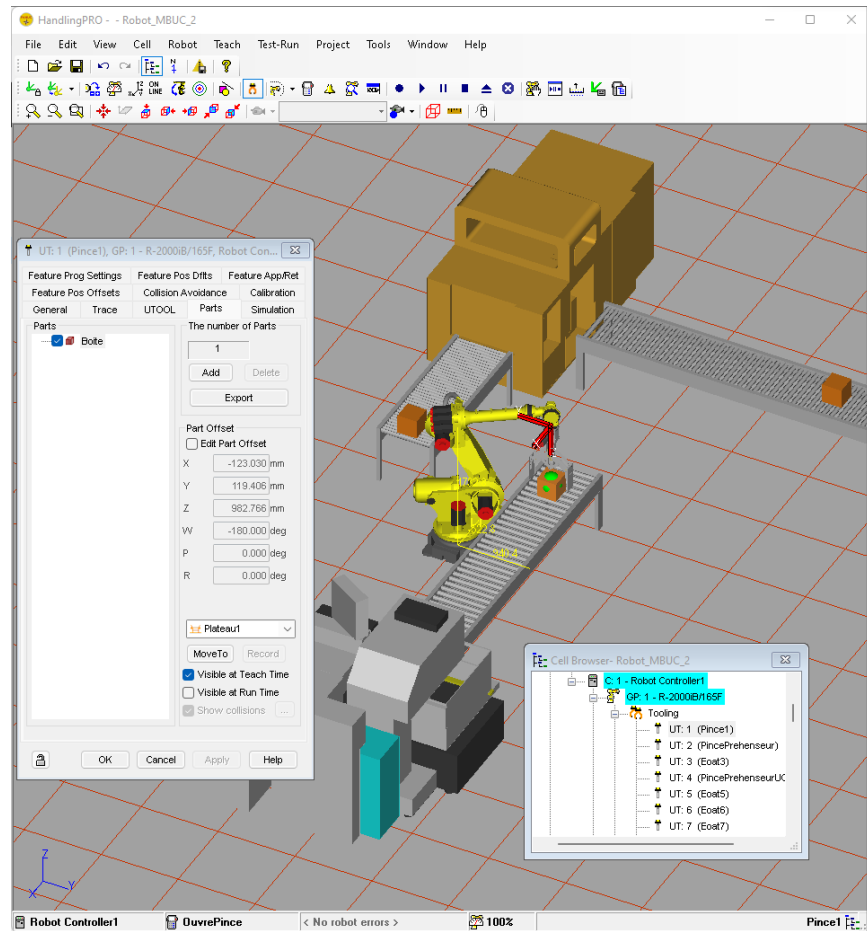


Figure 3: Le robot en position d'interaction avec la boîte sur le plateau 1.

## 1.2 Programme et simulation d'un programme de picking simple

- Nous avons implémenté un programme *Manutention* sur ce robot qui réalise une tâche de picking simple, c'est-à-dire que le robot va prendre la boîte sur le *plateau1* et aller la déposer sur le *plateau2*.

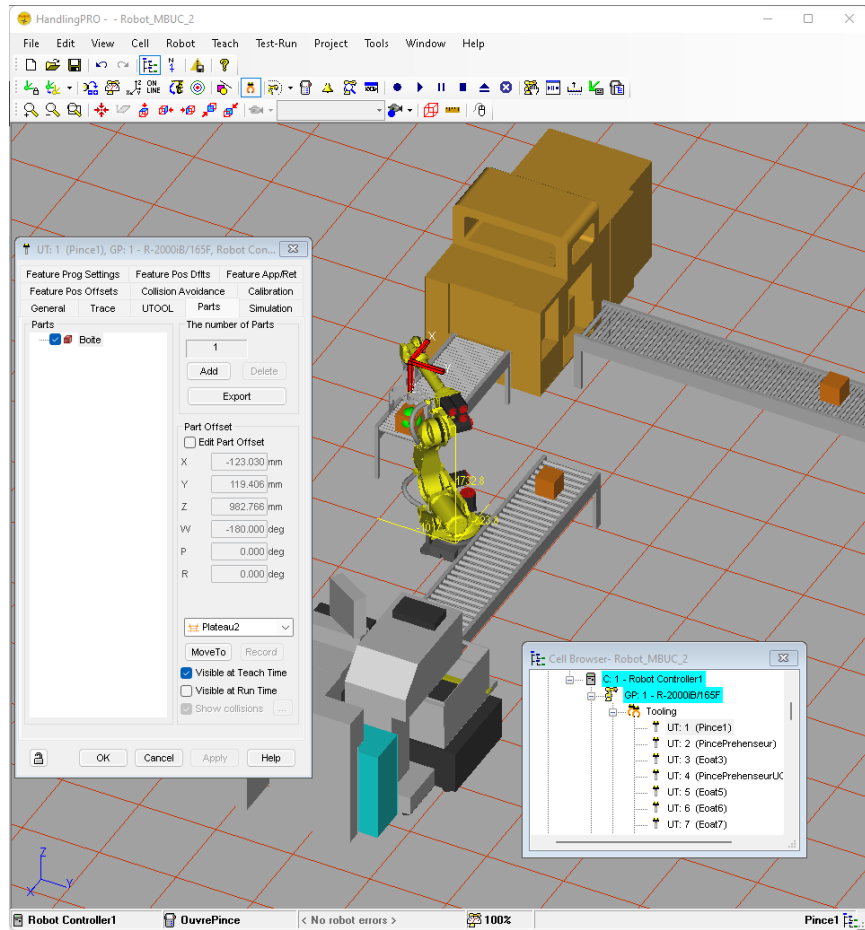


Figure 4: Le robot en position d'interaction avec la boîte sur le plateau 2.

- Pour les mouvements principaux ("longue distance") nous utilisons des mouvement de type  $[J]$  qui laissent libre choix au robot de ses mouvements afin de passer d'une position à la suivante car cela est plus rapide pour le robot.
- Pour les mouvements en phase d'approche, c'ad le dernier mouvement de descente pour saisir la boîte, nous utilisons des mouvements de type  $[L]$  qui forcent le robot à réaliser des mouvement rectilignes.
- La figure 5 montre le déroulé et l'exécution de notre programme *Manutention*.

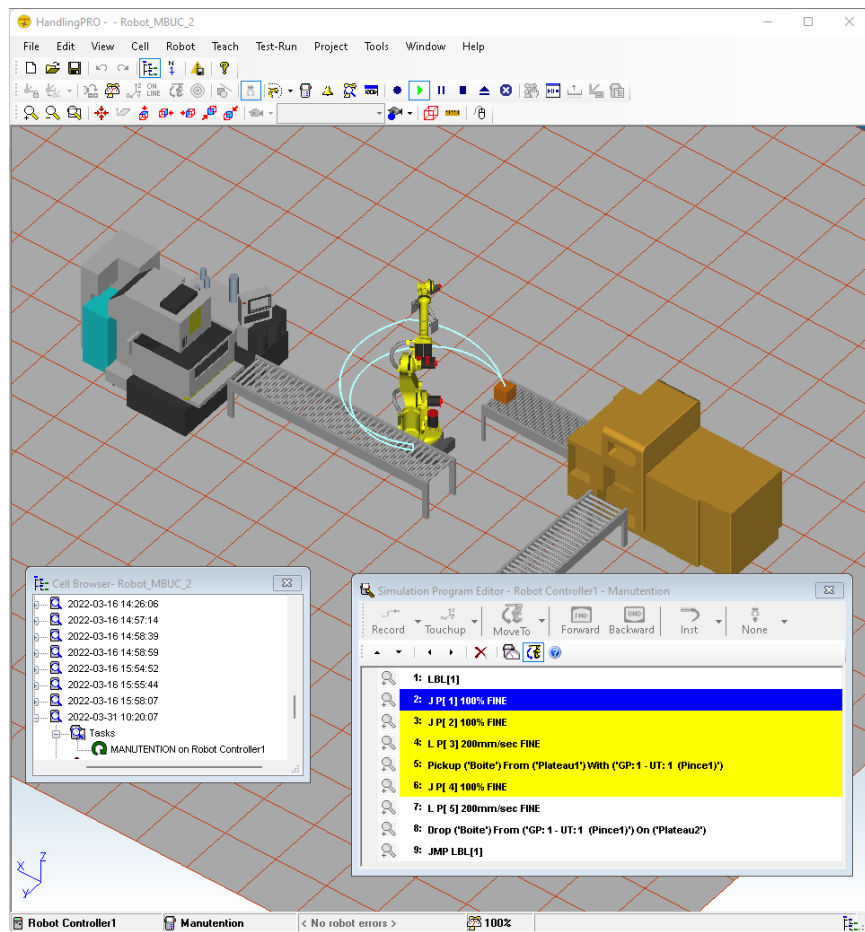


Figure 5: Déroulé et exécution du programme *Manutention*.

## 2 Ajout d'un effecteur

### 2.1 Mise en place d'un outil de préhension

- Nous avons paramétré un nouvel outil sur l'effecteur du robot: une pince commandable. La mise en place a consisté à :
  - "constituer" la pince à partir de 3 objets CAD,
  - mettre à l'échelle pour être cohérent avec le reste de notre déploiement, cad la taille des boites,
  - paramétrer le repère Utool de la nouvelle pince pour que cela corresponde à la nouvelle pince.
- La figure 6 montre la nouvelle pince commandable installée.
- Nous avons testé la non-régression en appliquant le premier programme *Manutention* réalisé précédemment avec la nouvelle pince.

### 2.2 Programmation de la commande du préhenseur

#### 2.2.1 Lier la pincettes aux E/S sorties du robot

- Nous avons paramétré le comportement des 2 mors de la pièce afin de mettre en place commande d'ouverture et une commande de fermeture:
  - ServoHandA1 Inputs (Actionne le mors 1 si la commande est reçue de la part du robot sur le bit DO[1]):
    - \* Si (RobotController1.DigitalOutput[1]) { Position=-50}
    - \* Si (!RobotController1.DigitalOutput[1]) { Position=0}
  - ServoHandA1 Outputs (donne l'état d'ouverture totale sur le bit DI[1] et l'état de fermeture totale sur DI[2]):
    - \* Si (Position=0) { RobotController1.DigitalInput[1]=Vrai} Sinon {RobotController1.DigitalInput[1]=Faux}
    - \* Si (Position=-50){ RobotController1.DigitalInput[2]=Vrai} Sinon {RobotController1.DigitalInput[2]=Faux}
  - ServoHandA2 Inputs (Actionne le mors 2 si la commande est reçue de la part du robot sur le bit DO[1],cad le meme bit sur le mors 1):
    - \* Si (RobotController1.DigitalOutput[1]) { Position=50}

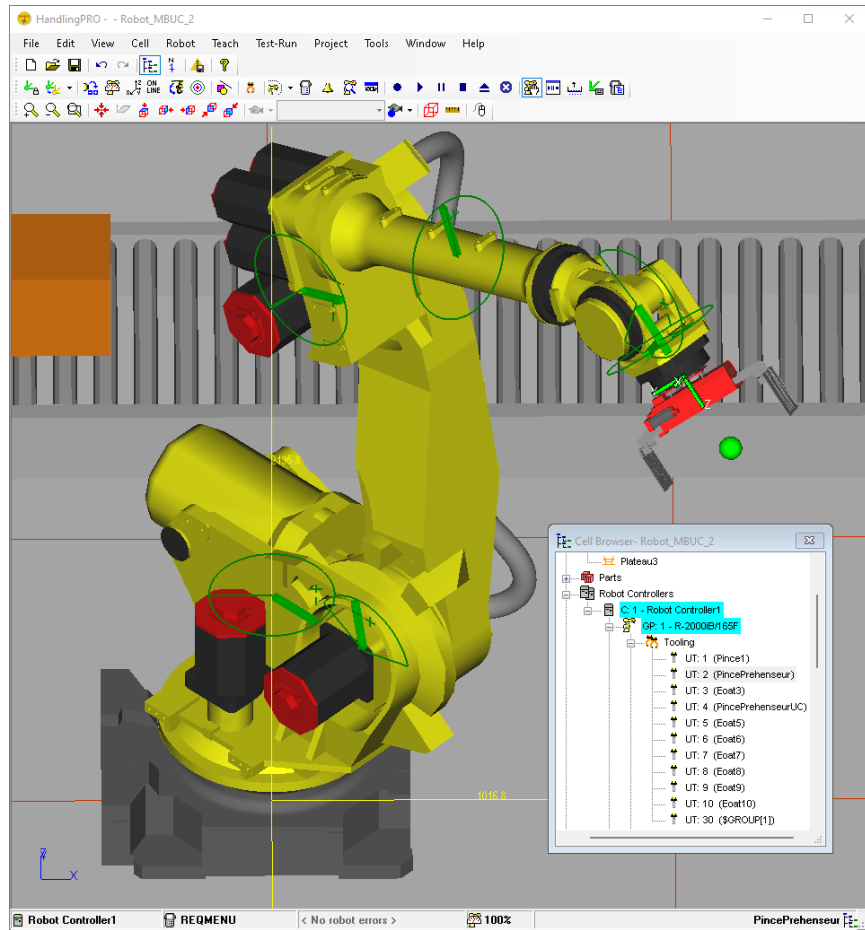


Figure 6: Nouvelle pince *ServoHand* installée sur le préhenseur du robot.



- \* Si (!RobotController1.DigitalOutput[1]) { Position=0}
- ServoHandA2 Outputs (donne les états d’ouverte totale et de fermeture totale sur les bits DI[3] et DI[4], cela permet de prendre en charge des temps de mouvement différents pour les 2 mors notamment):
  - \* Si (Position=0) { RobotController1.DigitalInput[3]=Vrai} Sinon {RobotController1.DigitalInput[3]=Faux}
  - \* Si (Position=-50){ RobotController1.DigitalInput[4]=Vrai} Sinon {RobotController1.DigitalInput[4]=Faux}
- Les figures 7 et 8 montre le parametrage du mouvement de la pince.

### 2.2.2 Programmer l’ouverture et la fermeture de la pince

#### 1. Programme *OuvrePince*

- Il consiste à mettre le bit DO[1] à faux pour déclencher l’ouverture des deux mors, puis attendre que les deux mors notifient l’ouverture complète par leur bits respectifs DI[1] et DI[3], voir figure 9.
- Ce programme servira ensuite de sous-programme dans des programme plus élaborés.


#### 2. Programme *FermePince*

- Il consiste à mettre le bit DO[1] à vrai pour déclencher la fermeture des deux mors, puis attendre que les deux mors notifient la fermeture complète par leur bits respectifs DI[2] et DI[4], voir figure 10.
- Ce programme servira ensuite de sous-programme dans des programme plus élaborés.

#### 3. Programme de test BoucleOuvreEtFermePince

- Pour tester le bon fonctionnement de la pince ainsi que le mécanisme d’appel de fonction nous avons implémenté un programme qui ouvre puis ferme la pince ad vitam eternam.
- Le programme est présenté en figure 11.

#### 4. Programme de picking avec pince commandable


ServoHandA1, Machine2, Machine2
✕

Link CAD

Parts

Simulation

General

Motion

Calibration

Motion Control Type

Device I/O Controlled

Axis Type

☐ Rotary
☒ Linear

Speed

Speed

→

15.00

mm/sec

←

15.00

mm/sec


Inputs

Output Dev	IO Tag	Value	Location
Robot Controller1	DO[1]	ON	-50
Robot Controller1	DO[1]	OFF	0
[none]	[none]	[none]	0

Test

Outputs

Input Dev	IO Tag	Value	Location
Robot Controller1	DI[1]	ON	0
Robot Controller1	DI[2]	ON	-50
[none]	[none]	[none]	0




OK

Cancel

Apply

Help

Figure 7: Paramétrage de la pince commandable, morsA1.


ServoHandA2, Machine2, Machine2
✕

Link CAD

Parts

Simulation

General

Motion

Calibration

Motion Control Type

Device I/O Controlled

Axis Type

☐ Rotary
 ☒ Linear

Speed

Speed

→

←

15.00

mm/sec

15.00

mm/sec


Inputs

Output Dev	IO Tag	Value	Location
Robot Controller1	DO[1]	ON	50
Robot Controller1	DO[1]	OFF	0
[none]	[none]	[none]	0

Test

Outputs

Input Dev	IO Tag	Value	Location
Robot Controller1	DI[3]	ON	0
Robot Controller1	DI[4]	ON	50
[none]	[none]	[none]	0



OK

Cancel

Apply

Help

Figure 8: Paramétrage de la pince commandable, morsA2.

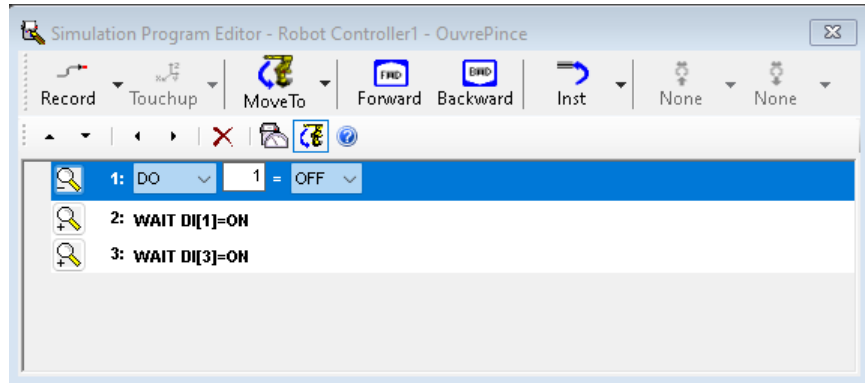


Figure 9: Programme d'ouverture de la pince.

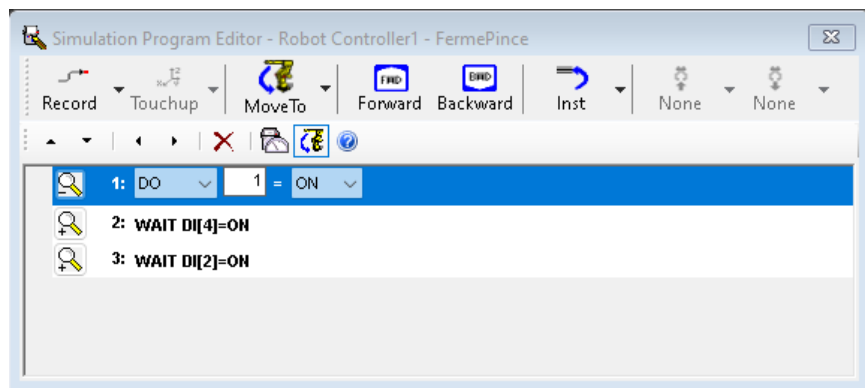


Figure 10: Programme de fermeture de la pince.

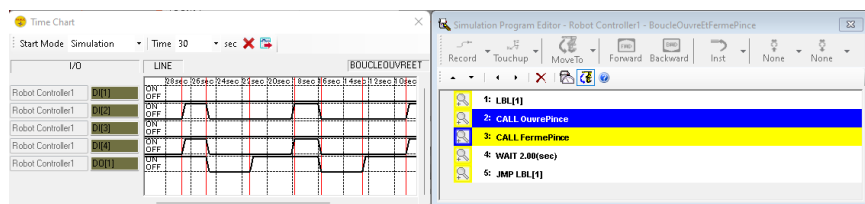


Figure 11: Programme de test d'ouverture/fermeture de la pince en boucle.

- Nous avons repris notre premier programme de manutention simple, cette fois avec la pince commandable.
- Les actions de saisies et de relachement de l'objet transporté par le robot sont pris en compte par les appels de fonction FermerPince et OuvrirPince quand le robot est en position respectivement *PickUp* et *Drop*.
- La figure 12 montre notre programme en cours d'exécution.

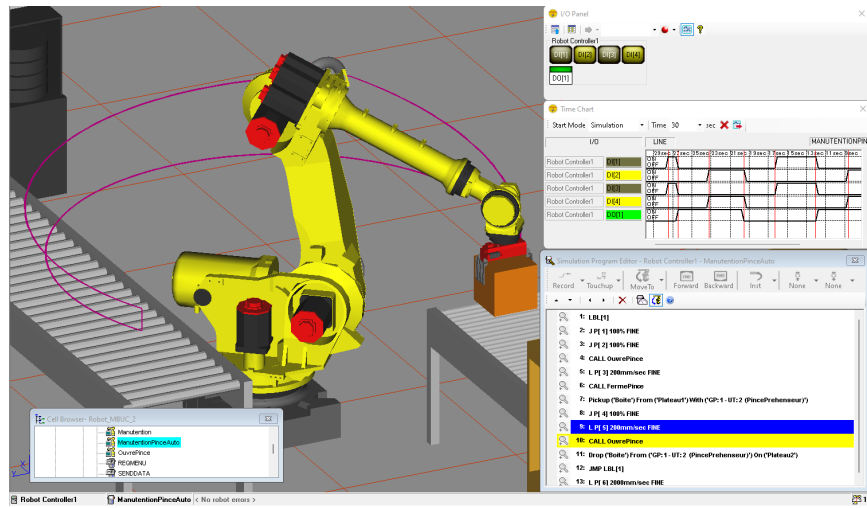


Figure 12: Programme de manutention de boite avec pince commandable.

### 3 Programmation de tâches conditionnelles

#### 3.1 Adaptation de l'environnement

- Nous ajoutons une zone de dépôt (table1) dans la zone de travail de notre outil pince avec préhenseur.
- Nous positionnons l'objet relativement à table1.
- Puis nous vérifions que la position de l'objet sur la table est atteignable par le robot équipé de notre pince, voir figure 13.

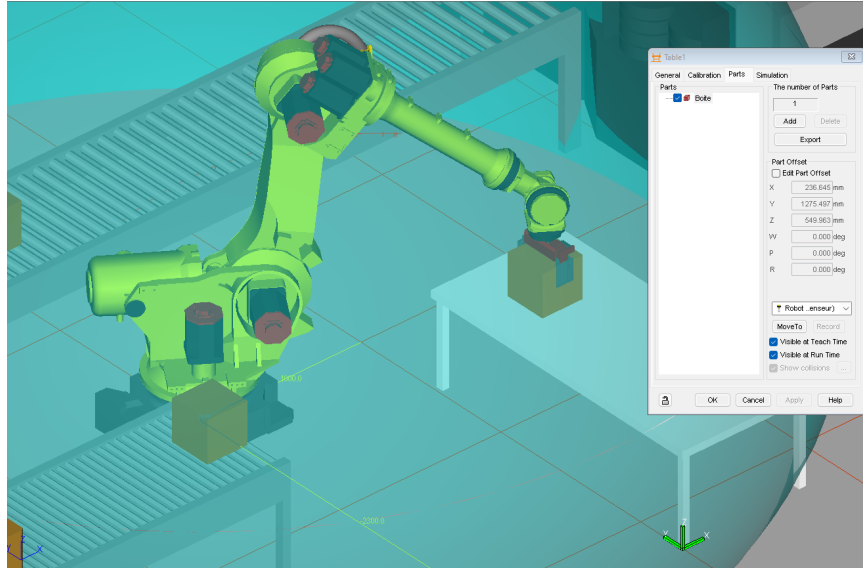


Figure 13: Nouvel environnement avec Zone de dépôt sur Table1 accessible.

### 3.2 Implémentation des sous-programmes de picking

- A partir de la base de programme de picking réalisé précédemment, nous avons implémenté 3 sous-programmes:
  - *Plateau1toPlateau2*
  - *Plateau1toTable1*
  - *ReposToSaisiePlateau1*

Les deux programmes réalisent 1 cycle de PickUp/Drop pour respectivement chacune des zone de dépôt. Ils partent de la même position: la position de saisie de la boîte sur le plateau1 et reviennent dans la même position de repos. Le programme *ReposToSaisiePlateau1* implémente la partie commune, c'est à dire le mouvement de la position de repos jusqu'à la saisie de la boîte sur le plateau1.

- La figure 14 montre le sous-programme *Plateau1toTable1* en cours d'exécution.

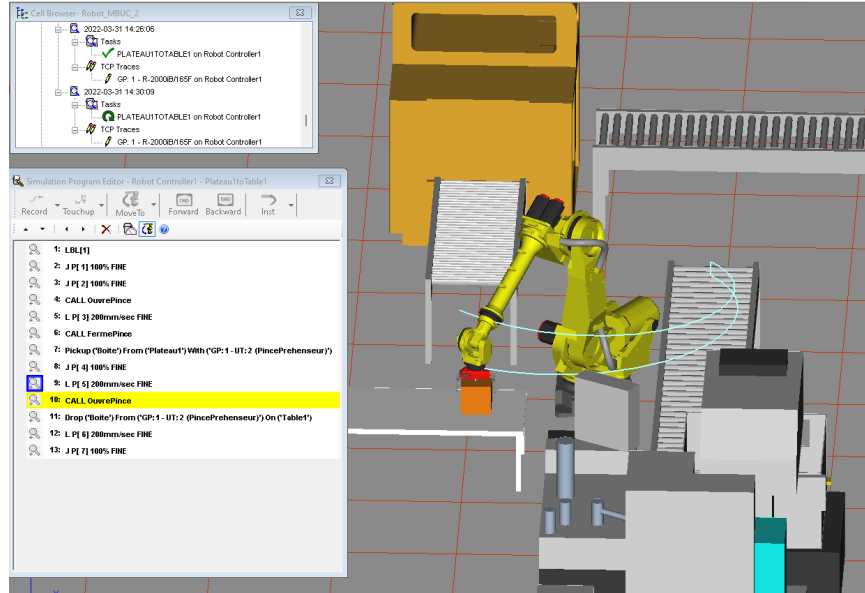


Figure 14: Sous-programme de Drop du plateau1 vers la table 1.

### 3.3 Implémentation du programme de manutention conditionnelle

- Avec nos sous-programmes fonctionnels, il nous suffit maintenant d'agencer l'appel des différentes fonctions dans une bloc *if then else* et de câbler le test de la condition sur un bit du robot.
- Le bloc *if then else* est implémenté via des *IF ... JMP LBL[...]*, c'est-à-dire des *goto* en programmation impérative.
- La condition est câblée sur le bit DI[5] que nous contrôlons manuellement via l'interface *IO/Panel*
- La figure 15 montre notre programme de manutention conditionnelle en cours d'exécution.

## 4 Programmation d'un second robot et coordination avec le premier

- Nous avons ajouté un robot et adapté l'environnement afin de réaliser une tâche supplémentaire:

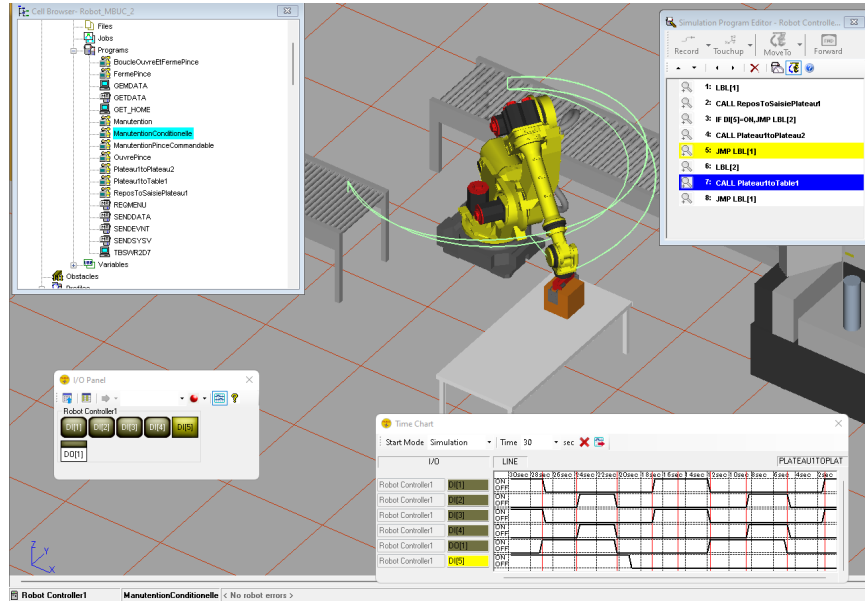


Figure 15: Programme de manutention de boîte conditionnelle: L'activation du bit DI[5] change la destination de dépôt.

- Lorsque une boîte est déposée sur la *Table1*, le Robot2 ramasse la boîte et la dépose sur un autre plateau.
- Nous avons paramétré et testé la capacité du second robot à effectivement ramasser et déposer la boîte dans le nouvel environnement.
- Afin de prévenir d'un éventuel risque de collision des robots nous avons synchronisé leurs actions respectives via un simple mécanisme d'exclusion mutuelle:

#### – **ROBOT2:**

- \* Robot2 est en position de repos et attend 2 conditions avant de déclencher ses mouvements vers le Pickup sur Table1:
  - La présence de la boîte sur la table (sur DI[1])
  - Le signal que Robot1 est en position de sécurité (sur Robot2.DI[2] venant de Robot1.DO[2])
- \* Lorsque Robot 2 passe ce double *Lock*, il pose un *Lock* (sur Robot2.DO[1] partant sur Robot1.DI[6]).
- \* Robot2 enclenche le pickup sur *Table1*



- \* Robot2 libère son *Lock* en sortant de la zone de Table1 (cad juste après sont mouvement éloignement)
- **ROBOT1:**
  - \* Après un dépôt sur la *Table1*, une opération est ajoutée: *Go-ToSafePlace* qui consiste a:
    - se positionner éloigné de *Table1*
    - Libérer le *Lock* en passant Robot1.DO[2]->Robot2.DI[2] à Vrai.
    - Attendre que Robot2 libère son *Lock* en passant Robot2.DO[1]->Robot1.DI[6] à Vrai.
    - Reposer un *Lock* en passant Robot1.DO[2]->Robot2.DI[2] à Faux.
- La figure 16 montre notre système de deux robots coordonnés en cours de simulation.

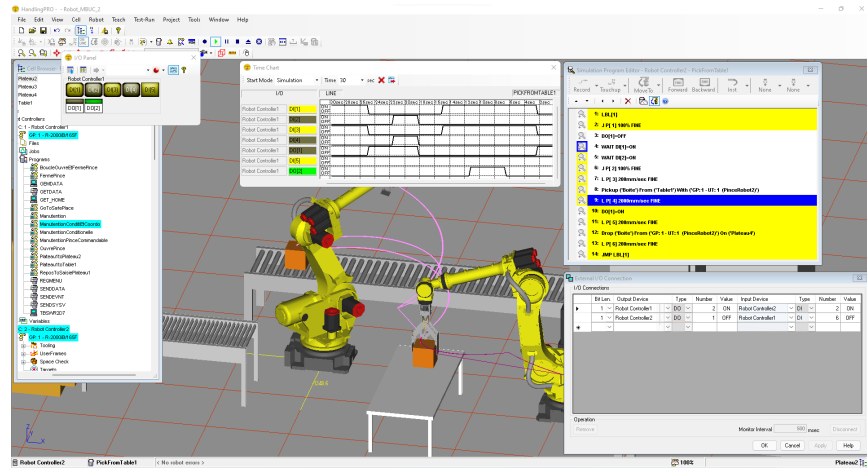


Figure 16: Simulation d'une chaine de manutention comprenant 2 robots avec une coordination entre eux pour prévenir des risques de collision.