# Rapport TP: Découverte cellule robotique STAUBLI RX60B
# Programmation hors ligne

Fleytoux Yoann
Bernier Aurélien

## **Partie utilisation:**

-Rappel des règles de sécurité à appliquer pour l'utilisation des cellules. Quelle norme?

Son utilisation exige une attention de tous les instants et le respect strict de procédures de test et mise en sécurité tant des matériels que des opérateurs. Le droit à l'erreur n'existe pas dans un tel environnement. Le risque de blessure grave et de casse matérielle majeure est réel.

Il y a des règles "évidentes" comme ne pas entrer dans l'enceinte du robot quand celui ci est allumé et l'implémentation d'un bouton homme-mort pour faciliter le respect de ces règles.

Les normes ISO 10218-1 (Robots pour environnements industriels –Exigences de sécurité- Partie 1 : Robots) et 10218-2 (Robots pour environnements industriels –Exigences de sécurité- Partie 2 : Système Robot et Intégration) sont équivalentes à la norme "Machines" 2006/42/CE et permettent de formaliser les règles pour généraliser la sécurité lors de l'utilisation des robots. L'ISO 11161 (Sécurité des machines : Sécurité des systèmes automatisés) et l'ISO 10218 définissent les règles de sécurité tandis que l'ISO 13855 (Sécurité des machines - Positionnement des dispositifs de protection par rapport à la vitesse d'approche des parties du corps) définit l'énergie cinétique maximale de la partie mobile, ce qui se traduit en l'effort maximal exercé sur les parties du corps de l'éventuel opérateur entrant en contact avec ce mouvement.

-Description de la cellule utilisée.

On utilise la cellule robotique STAUBLI RX60B, équipé d'une pince. C'est bras industrielle avec 6 degrées de libertés.

## **Programmation:**

-Explication sur les différentes références base et en quoi cela est utile (world,tool,base).

Les différents repères permettent de décrire la position du robot dans l'espace:

Outil :

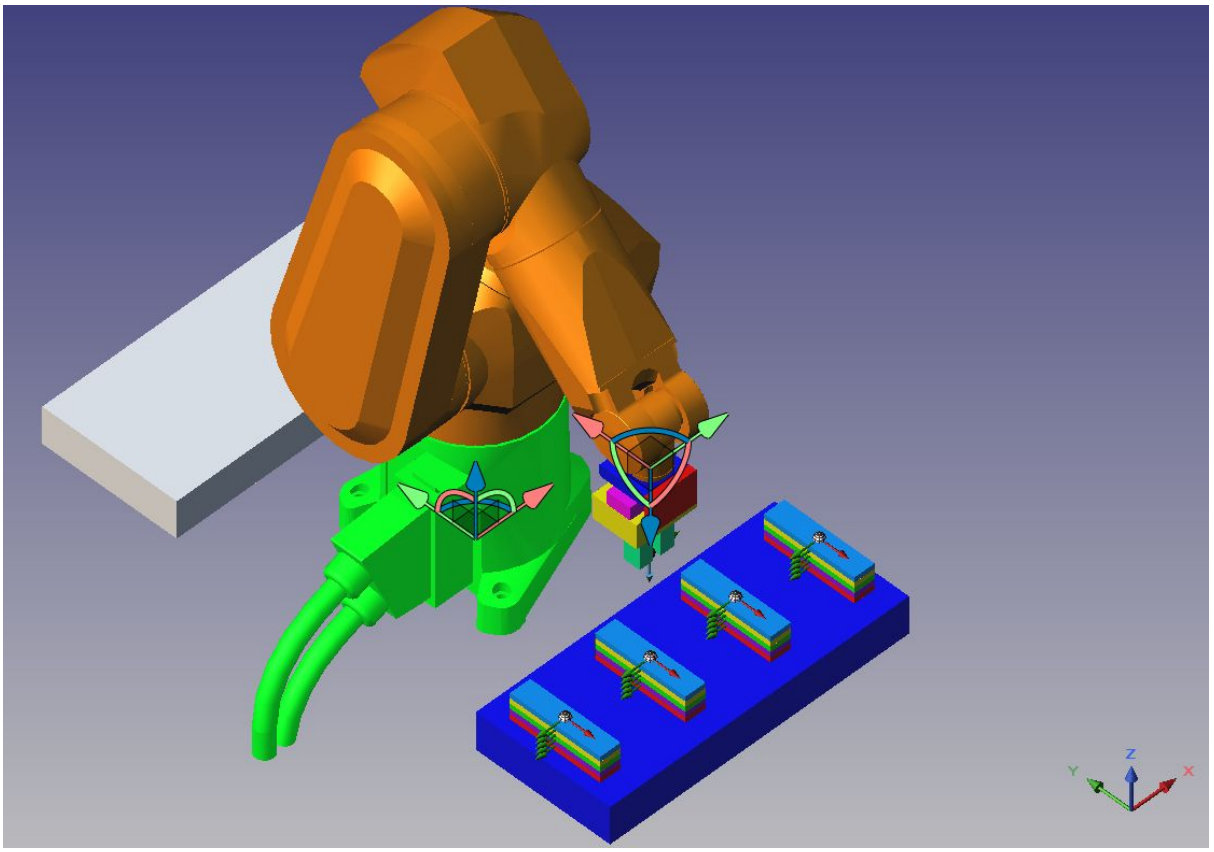
Le repère outil à comme position 0 le centre de l'outil. Il définit donc la position et l'orientation de l'outil. Le centre de l'outil se déplace à la position demandée. Tous les robots ont un repère outil prédéfini qui peut s'avérer utile pour bouger un bras sans changer l'orientation de l'outil.

Monde :

Le repère monde à comme position 0 le centre de la cellule, il est utile quand on utilise plusieurs robots, ou des robots qui se font déplacer. Par défaut, le repère monde est le même que le repère base.

Base :

Le repère base à comme position 0 la base du robot, ce qui rend ses mouvements prédictibles pour les robots fixés. C'est donc pratique pour le déplacer, en temps normal, avec un joystick on peut le bouger sur l'axe X en poussant ou tirant le joystick, sur l'axe Y avec gauche-droite et sur l'axe Z en le tournant sur lui-même.


Z
Y
X

* **num movec(point pIntermédiaire, point pCible, tool tOutil, mdesc mDesc)**

Fonction:

Cette instruction enregistre une commande pour un mouvement circulaire qui part de la destination du mouvement précédent et s'achève au point pCible en passant par le point pIntermédiaire.

Elle renvoie l'identifiant de mouvement attribué à ce mouvement et incrémente d'un l'identifiant de mouvement pour la prochaine commande de mouvement.

* **movej(joint, tool, mdesc)**: move to a (point or joint) coordinate with specified tool and motion descriptor

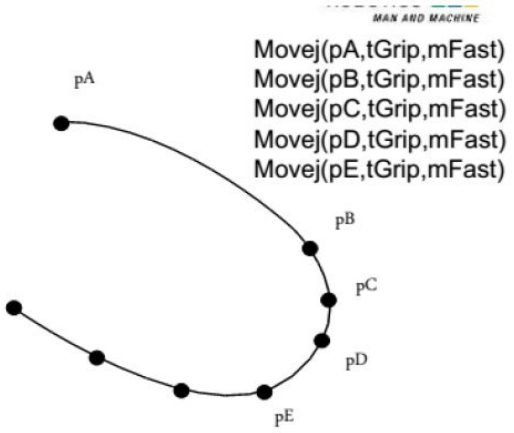
# Movement with movej

***MOVEMENT : MOVEJ***

Movej(point,tool,mdesc)

or

Movej(joint,tool,mdesc)




Joint Interpolation : curved movement

Speed and acceleration described by motion descriptor

No problem of singularity crossing

Motion to use if no constraint : Obstacle, insertion, . . .

* **movel(point, tool, mdesc):** move linearly a point coordinate with specified tool and motion descriptor

# Movement with movel

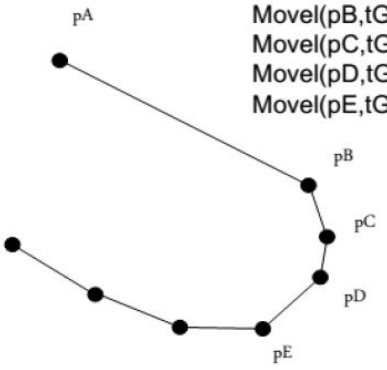
**MOVEMENT : MOVEL**

Movel(point,tool,mdesc)

Not available on JOINT



Movel(pA,tGrip,mFast)
Movel(pB,tGrip,mFast)
Movel(pC,tGrip,mFast)
Movel(pD,tGrip,mFast)
Movel(pE,tGrip,mFast)




Cartesian Interpolation : straight line movement
Speed and acceleration described by motion descriptor
Problem of singularity crossing => slow down
Motion to use in case of constraint : obstacle, insertion, process,
. . .

**Partie expérimentale:**
Un compte rendu des manipulation demandées dans le TP
Code source des programmes effectués + vidéo ( dans le zip en fichier joint)
Explication des choix effectués et des difficultés rencontrées.

La découpe:
Réalisé. La nous avons appris à coder l'interface utilisateur et à prendre en compte les données rentrée par l'utilisateur.

Empilement de 4 pièces pour faire une tour:
Réalisé. Ici nous avons appris à générer des objets et la gestions des collisions.

Création de mur:
Réalisé. L'appel de plusieurs programmes dans un même programme.

Création de sol:
Réalisé

Avec la prise en compte d'un angle rentrée par l'utilisateur:
Réalisé.

Choix et difficultés rencontrées.:

Nous avons trouvé le programme peu ergonomique au premier abord (devoir déclarer les variables via l'interface graphique de l'application ou l'utilisation du pendant pour exécuter notre code par exemple), et nous avons eu quelques bugs, parfois nous faisant perdre du travail (nous aurions pu sauvegarder plus régulièrement…).
Notre connaissance du logiciel et du langage Val3 s'est améliorée au fur et à mesure du projet, ce qui a pu influencer la qualité du code et les choix techniques choisis, ce qui a pu expliquer notre difficulté au démarrage.

**Conclusion:**
Conclusion sur les notions abordées sur le projet et commentaires.

Ce projet nous a permis d'apprendre à modéliser et à utiliser le robot STAUBLI RX60B par le biais de la Staubli Robotics Suite. Nous avons pour cela programmé les différentes séquences d'instructions en Val3.

**Code Start :**

```
begin
   userPage(sEcran)
  cls(sEcran)
   putln(sEcran, "Quel programme voulez vous lancer : ")
  putln(sEcran, "   1. Découpe")
  putln(sEcran, "   2. Tour de 4 pièces")
  putln(sEcran, "   3. Constuire tour")
  get(sEcran, nYN)
  if nYN == 1
          call test_decoupe()
  elseIf nYN==2
          call saisirpiecev1()
  elseIf nYN==3
          call Tour()
  endIf
end
```

**Code Découpe:**

```
userPage(sEcran)
cls(sEcran)
//on vas à la position initiale

movej(Starting_point, tTool, mNomSpeed)
waitEndMove()

//    demande si point de départ custo
putln(sEcran, "Voulez vous changer le point de départ de la découpe : ")
putln(sEcran, "    1. Oui")
putln(sEcran, "    2. Non")
get(sEcran, nYN)
//    si oui recuperation
if nYN == 1
        putln(sEcran, "x = ")
        get(sEcran, nx)

        putln(sEcran, "y = ")
        get(sEcran, ny)

        putln(sEcran, "z = ")
        get(sEcran, nz)

        putln(sEcran, "Rx = ")
        get(sEcran, nRx)

        putln(sEcran, "Ry = ")
        get(sEcran, nRy)

        putln(sEcran, "Rz = ")
        get(sEcran, nRz)

        Real_start = {{nx,ny,nz,nRx,nRy,nRz}, {sfree, efree, wfree}}
else
        //    sinon point de depart par defaut
        Real_start=Starting_point
endIf

//récupération largeur et hauteur de la découpe:
//    demande si dimension custo
putln(sEcran, "Voulez vous changer les dimensions de la découpe : ")
putln(sEcran, "    1. Oui")
putln(sEcran, "    2. Non")
get(sEcran, nYN)
```

```
if nYN == 1
//    si oui recuperation
putln(sEcran, "")
putln(sEcran, "Veuillez entrer les dimensions de votre découpe :")

put(sEcran, "Hauteur = ")
get(sEcran, Dim_height)
put(sEcran, Dim_height)
putln(sEcran, "")

put(sEcran, "Largeur = ")
get(sEcran, Dim_width)
put(sEcran, Dim_width)
putln(sEcran, "")
else
        //    sinon demension par defaut
        Dim_height = 5
        Dim_width = 10
endIf

//découpe (par defaut on comence en haut à gauche)
B = {{Real_start.trsf.x - Dim_height/2, Real_start.trsf.y + Dim_height/2, Real_start.trsf.z,
Real_start.trsf.rx, Real_start.trsf.ry, Real_start.trsf.rz}, {sfree, efree, wfree}}
C = {{Real_start.trsf.x - Dim_height, Real_start.trsf.y, Real_start.trsf.z, Real_start.trsf.rx,
Real_start.trsf.ry, Real_start.trsf.rz}, {sfree, efree, wfree}}
D = {{Real_start.trsf.x - Dim_height, Real_start.trsf.y + Dim_width, Real_start.trsf.z,
Real_start.trsf.rx, Real_start.trsf.ry, Real_start.trsf.rz}, {sfree, efree, wfree}}
E = {{Real_start.trsf.x, Real_start.trsf.y + Dim_width, Real_start.trsf.z, Real_start.trsf.rx,
Real_start.trsf.ry, Real_start.trsf.rz}, {sfree, efree, wfree}}


movel(Real_start, tTool, mNomSpeed)
waitEndMove()

movec(B,C,tTool, mNomSpeed)
waitEndMove()

movel(D, tTool, mNomSpeed)
waitEndMove()

movel(E, tTool, mNomSpeed)
waitEndMove()

movel(Real_start, tTool, mNomSpeed)
waitEndMove()
```

```
  //on vas à la position initiale
  movej(Starting_point, tTool, mNomSpeed)
  waitEndMove()

  put(sEcran, "Operation réalisé")
end
```

**Code Tour:**

```
begin

  userPage(sEcran)
  cls(sEcran)

  //on va a la position de depart
   movej(Starting_point, tTool, mNomSpeed)
  waitEndMove()
  //on ouvre la pince
        open(tTool)
  for i = 0 to 3

        //on vas au dessus de la zone à saisir
        Real_start=pCentrepiece1[1]
        //note:plus z est grand plus le bras sera haut
        //la on est au dessus
        movej(Real_start, tTool, mNomSpeed)
        waitEndMove()

        //on vas a la piece on met le 123 pour descendre sinon on est trop haut
        B = {{Real_start.trsf.x , Real_start.trsf.y, Real_start.trsf.z-180+(DepotStock[1])*7,
Real_start.trsf.rx, Real_start.trsf.ry, Real_start.trsf.rz}, {sfree, efree, wfree}}
        movel(B, tTool, mNomSpeed)
        waitEndMove()
        //on ferme la pince
        close(tTool)
        //on revient au dessus
        movel(Real_start, tTool, mNomSpeed)
        waitEndMove()

        //Real_Stop=pCentre_Arrive

        //on amene la piece au dessus de la position où construire
        movej(pCentre_Arrive[6], tTool, mNomSpeed)
        waitEndMove()
   //on baisse la pince
        Real_Stop = {{pCentre_Arrive[0].trsf.x , pCentre_Arrive[0].trsf.y,
pCentre_Arrive[0].trsf.z-165+nNbEtage*7, pCentre_Arrive[0].trsf.rx, pCentre_Arrive[0].trsf.ry,
pCentre_Arrive[0].trsf.rz}, {sfree, efree, wfree}}
        movel(Real_Stop, tTool, mNomSpeed)
        waitEndMove()
        //on ouvre la pince
        open(tTool)
        //on relève la pince
```

```
            movel(pCentre_Arrive[6], tTool, mNomSpeed)
            waitEndMove()
            DepotStock[1]=DepotStock[1]-1
            nNbEtage=nNbEtage+1
    endFor

    //on retourne à la position de départ
    movej(Starting_point, tTool, mNomSpeed)
    waitEndMove()
   //on ferme la pince
  close(tTool)

    put(sEcran, "Operation réalisé")
  end
```

**Code Mur:**

```
begin

  userPage(sEcran)
  cls(sEcran)

  //on va a la position de depart
   movej(Starting_point, tTool, mNomSpeed)
  waitEndMove()
  //on ouvre la pince
  open(tTool)

  for i = 0 to 3
          //on prend le depot ou il y a des pièces en stock
          while(DepotStock[indiceStock]<1)
          indiceStock=indiceStock+1
          endWhile

          //on vas au dessus de la zone à saisir
          //pCentrePiece est un tableau qui contient les position au dessus des stocks
          Real_start=pCentrepiece1[indiceStock]
          //note:plus z est grand plus le bras sera haut
          //la on est au dessus
          movej(Real_start, tTool, mNomSpeed)
          waitEndMove()

          //on vas a la piece on met le 123 pour descendre sinon on est trop haut
          B = {{Real_start.trsf.x , Real_start.trsf.y,
Real_start.trsf.z-180+(DepotStock[indiceStock])*7, Real_start.trsf.rx, Real_start.trsf.ry,
Real_start.trsf.rz}, {sfree, efree, wfree}}
          movel(B, tTool, mNomSpeed)
          waitEndMove()
          //on ferme la pince
          close(tTool)
          //on revient au dessus
          movel(Real_start, tTool, mNomSpeed)
          waitEndMove()

          //Real_Stop=pCentre_Arrive

          //on amene la piece au dessus de la position où construire
          //pCentre_Arrive contient un tableau avec les 6 positions possibles de depots
          movej(pCentre_Arrive[6], tTool, mNomSpeed)
          waitEndMove()
    //on baisse la pince
```

```
        Real_Stop = {{pCentre_Arrive[i].trsf.x , pCentre_Arrive[i].trsf.y,
pCentre_Arrive[i].trsf.z-165+nNbEtage*10, pCentre_Arrive[i].trsf.rx, pCentre_Arrive[i].trsf.ry,
pCentre_Arrive[i].trsf.rz}, {sfree, efree, wfree}}

        //on modifi l'angle si necessaire
        if i >1
        Real_Stop.trsf.rz=Real_start.trsf.rz+nValeurAngle
        endIf

        movel(Real_Stop, tTool, mNomSpeed)
        waitEndMove()
        //on ouvre la pince
        open(tTool)
        //on relève la pince
        movel(pCentre_Arrive[6], tTool, mNomSpeed)
        waitEndMove()
        DepotStock[indiceStock]=DepotStock[indiceStock]-1
        if i==1
        nNbEtage=nNbEtage+1
        endIf
  endFor

  //on retourne à la position de départ
  movej(Starting_point, tTool, mNomSpeed)
  waitEndMove()
 //on ferme la pince
close(tTool)

  put(sEcran, "Operation réalisé")
end
```

**Code Sol:**

```
begin

  userPage(sEcran)
  cls(sEcran)

  //on va a la position de depart
   movej(Starting_point, tTool, mNomSpeed)
  waitEndMove()
  //on ouvre la pince
  open(tTool)

  for i = 0 to 5
        //on prend le depot ou il y a des pièces en stock
        while(DepotStock[indiceStock]<1)
        indiceStock=indiceStock+1
        endWhile

        //on vas au dessus de la zone à saisir
        //pCentrePiece est un tableau qui contient les position au dessus des stocks
        Real_start=pCentrepiece1[indiceStock]

        //note:plus z est grand plus le bras sera haut
        //la on est au dessus
        movej(Real_start, tTool, mNomSpeed)
        waitEndMove()

        //on vas a la piece on met le 123 pour descendre sinon on est trop haut
        B = {{Real_start.trsf.x , Real_start.trsf.y,
Real_start.trsf.z-180+(DepotStock[indiceStock])*7, Real_start.trsf.rx, Real_start.trsf.ry,
Real_start.trsf.rz}, {sfree, efree, wfree}}
        movel(B, tTool, mNomSpeed)
        waitEndMove()
        //on ferme la pince
        close(tTool)
        //on revient au dessus
        movel(Real_start, tTool, mNomSpeed)
        waitEndMove()

        //Real_Stop=pCentre_Arrive

        //on amene la piece au dessus de la position où construire
        //pCentre_Arrive contient un tableau avec les 6 positions possibles de depots
        movej(pCentre_Arrive[6], tTool, mNomSpeed)
        waitEndMove()
```

```
   //on baisse la pince
          Real_Stop = {{pCentre_Arrive[i].trsf.x , pCentre_Arrive[i].trsf.y,
pCentre_Arrive[i].trsf.z-165+nNbEtage*10, pCentre_Arrive[i].trsf.rx, pCentre_Arrive[i].trsf.ry,
pCentre_Arrive[i].trsf.rz}, {sfree, efree, wfree}}
          //on modifi l'angle si necessaire
          if i >1
          Real_Stop.trsf.rz=Real_start.trsf.rz+nValeurAngle
          endIf

          movel(Real_Stop, tTool, mNomSpeed)
          waitEndMove()
          //on ouvre la pince
          open(tTool)
          //on relève la pince
          movel(pCentre_Arrive[6], tTool, mNomSpeed)
          waitEndMove()
          DepotStock[indiceStock]=DepotStock[indiceStock]-1
          if i==1
          nNbEtage=nNbEtage+1
          endIf
  endFor

  //on retourne à la position de départ
  movej(Starting_point, tTool, mNomSpeed)
  waitEndMove()
 //on ferme la pince
close(tTool)

  put(sEcran, "Operation réalisé")
end
```