# Module Robotique :
# Bras robotique industriel
# Application aux robots STAUBLI

BUT 2 AII

IUT GEII Toulouse

2023

B. Vandeportaele

# Sources

- https://bvdp.inetdoc.net/files/staubliiut/robot_srs_et_controleur/TX260%20Manuel.PDF
- https://bvdp.inetdoc.net/files/staubliiut/robot_srs_et_controleur/Contr%C3%B4leur%20CS9%20manuel.PDF
- https://bvdp.inetdoc.net/files/staubliiut/robot_srs_et_controleur/Contr%C3%B4leur%20CS9%20elec.PDF
- https://bvdp.inetdoc.net/files/staubliiut/robot_srs_et_controleur/guide-de-la-robotique-staubli-robotics-2020.pdf
- https://bvdp.inetdoc.net/files/staubli/documentations/Val3.PDF

  https://bvdp.inetdoc.net/files/staubliiut/pince_schunk/IM0010510.PDF
- Cours Staubli : Programmation Hors Ligne - Simulation - Section 1-2-4 - SRS2019.pptx

- Certains slides sont inspirés du cours de Mr Fabio MORBIDI : https://home.mis.u-picardie.fr/~fabio/Eng/documenti/Teaching/INRO22-23/InitRob2.pdf

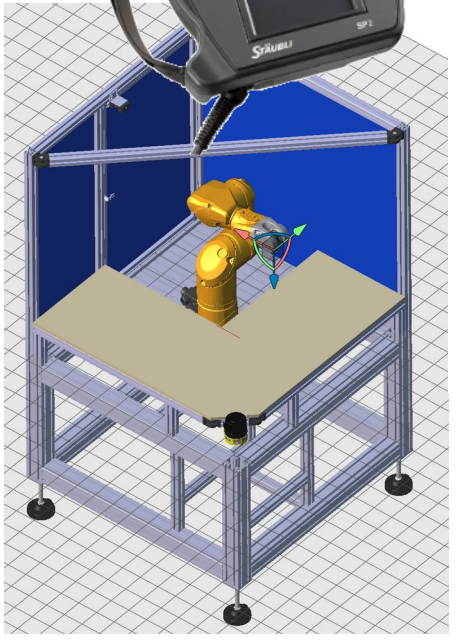
# Sommaire

- Cellule robotique
- Bras 6R,repères et outils
- Boîtiers de mode et de commande manuelle
- Tâche robotique et programmation du robot

# La cellule robotique

- Un ou plusieurs robots
  - Bras
  - Contrôleur
  - Boitier sélection de mode de marche (WMS)

    Boitier de commande manuel/pendant (MCP)
  - Outils effecteurs
    - Par exemple pince + mors adaptés à la tâche
    - Éventuellement changeur d'outils
- Dans un environnement adapté
  - Table/support/tapis pour le robots et les pièces à manipuler
  - Avec des éléments de sécurité :
    - Protections mécaniques
    - Capteurs
- En interaction avec
  - Des capteurs, des actionneurs, des automates
  - Interfaçage via E/S physiques, réseau….
- Monitoré(s) par un ordinateur pour le développement, la simulation (PHL) et la supervision

# Le bras anthropoïde (6R)

Le bras est constitué de maillons reliés entre eux par des articulations. Les différents maillons sont les suivants : La base **(A)**, l'épaule **(B)**, le bras **(C)**, le coude **(D)**, l'avant-bras **(E)** et le poignet **(F)**.

Les mouvements des articulations du bras sont générés par des servomoteurs couplés à des codeurs de position sûrs. Des freins moteur maintiennent le bras en position immobile quand les servomoteurs sont à l'arrêt. Un processeur numérique sûr (DSI9) **(G)** traite en séquence les informations transmises au contrôleur et commande les freins et les électrovannes.

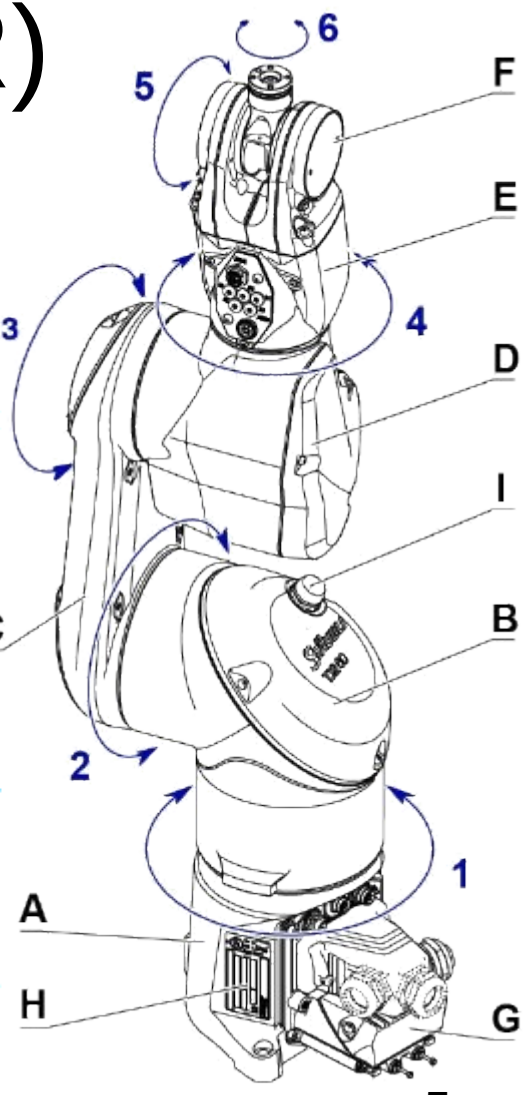
Le bras renferme le câblage électrique et les tuyaux pneumatiques à disposition de l'utilisateur, reliant les connecteurs de la base à ceux de l'avant-bras. L'avant-bras a également des points de fixation pour d'éventuels dispositifs à raccorder au préhenseur du robot.

Le bras peut être monté dans n'importe quelle position avec une sortie de câbles horizontale ou verticale en fonction des contraintes spécifiques de la machine.

Différentes options sont proposées pour que le bras se conforme à la configuration et aux exigences environnementales de l'utilisateur final, telles qu'une salle blanche (SCR), l'utilisation d'huile alimentaire (H1), une version UL, un environnement humide (HE) ou bio-contaminé (Stericlean).

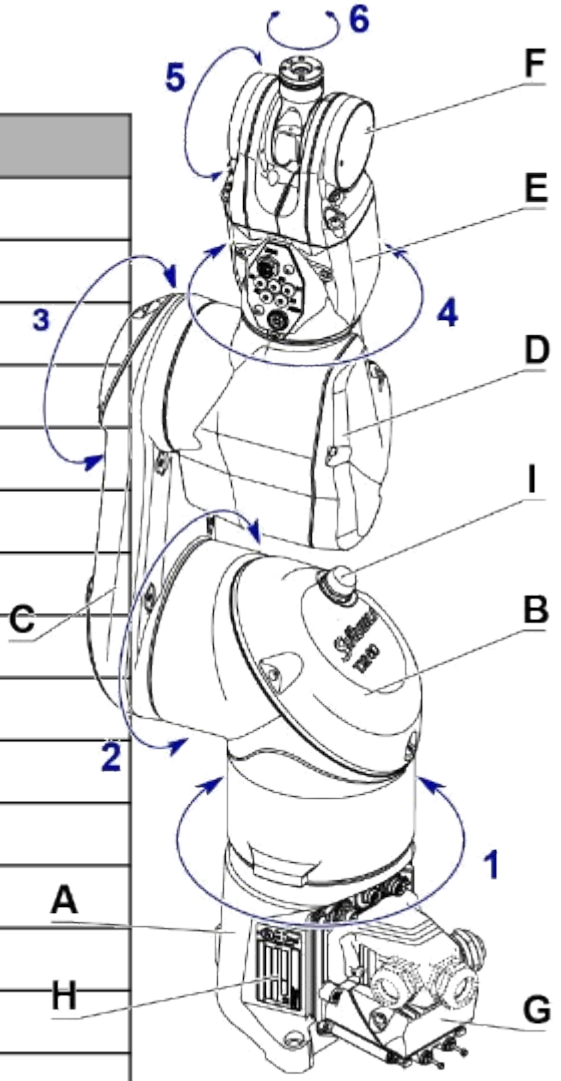
Pour la version UL, un témoin d'alimentation jaune **(I)** est monté sur le coude du robot pour indiquer que la puissance est disponible, que les mouvements sont possibles et qu'ils constituent une source de risque pour l'opérateur.

La vitesse, la précision et la répétabilité du bras permettent un large éventail d'applications telles que la construction automobile et la fabrication d'équipements, le secteur alimentaire, la médecine et la pharmaceutique, le chargement de machines, le plastique, etc...

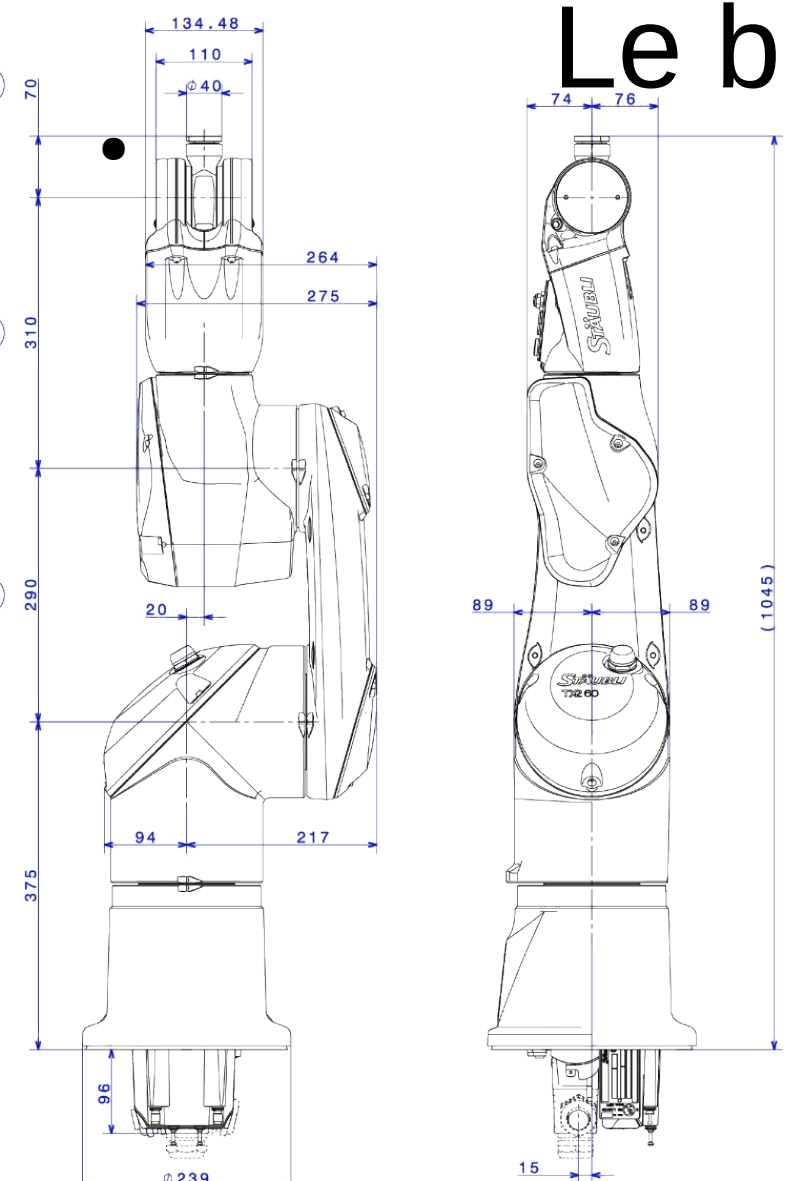


# Le bras

| Modèle | TX2–60 | TX2–60L |
|---|---|---|
| Caractéristiques | | |
| Charge limite (voir chapitre 3.2.4) | 9 kg | 5 kg |
| Charge nominale | 3.5 kg | 2 kg |
| Rayon (entre l'axe 1 et 6) (voir chapitre 3.2.2) | 670 mm | 920 mm |
| Nombre de degrés de liberté | 6 | 6 |
| Répétabilité - ISO 9283 | ± 0.02 mm | ± 0.03 mm |
| Vitesse maximale au centre de gravité de la charge | 8.4 m/s | 11.1 m/s |
| Vitesse linéaire caractéristique | (tbd) | (tbd) |
| Bruit en conditions nominales | 70 dBA | |
| Cycle 25.300.25 mm (max) | (tbd) | (tbd) |
| Consommation d'énergie - VDMA 24608 | (tbd) | (tbd) |
| Norme de propreté - ISO 14644-1 | Classe ISO5 | |
| Classe de protection - EN 60529 | IP65 (et IP67 avec la version pressurisée) | |
| Poids (voir chapitre 4.1) | 52.2 kg | 52.9 kg |
| Contrôleur Stäubli | CS9 | |
| Freins | Tous les axes | |

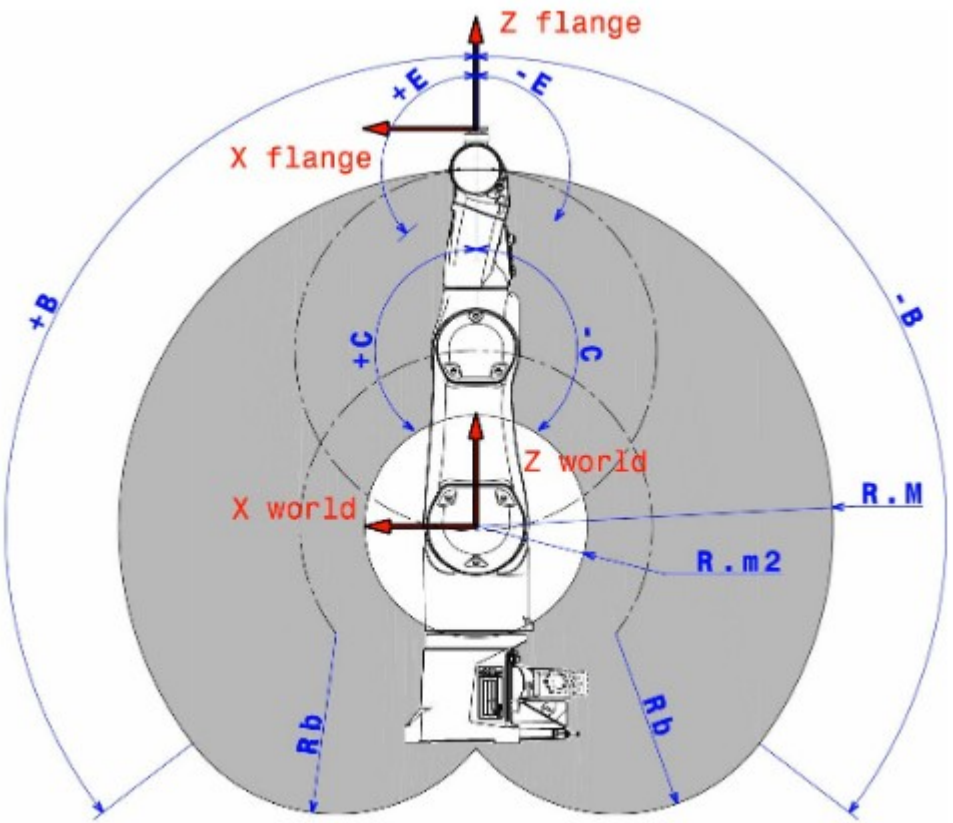


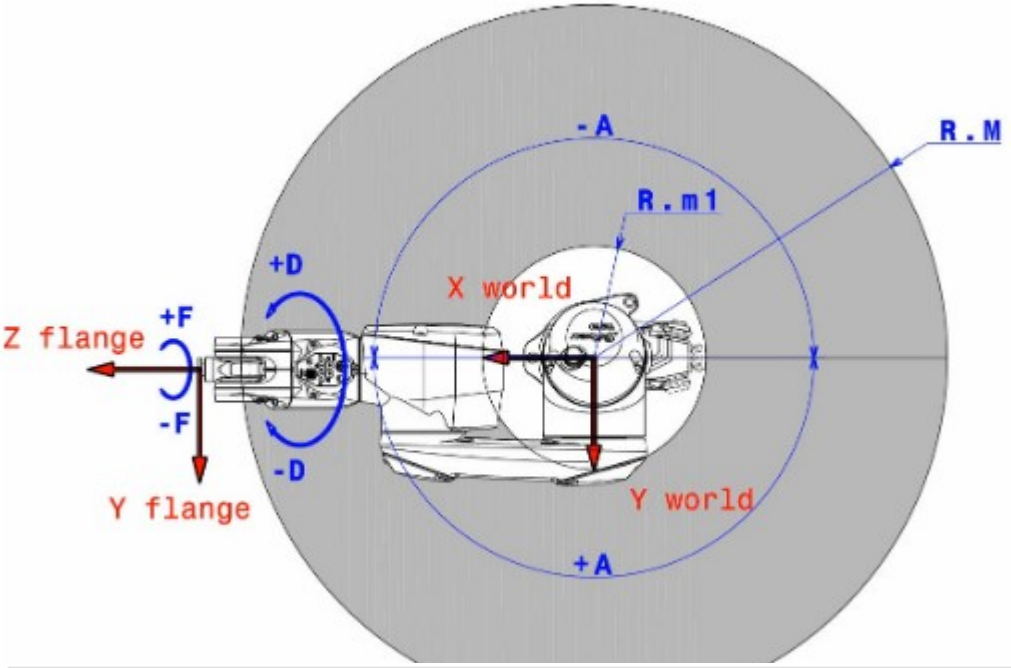
# Le bras : géométrie




| Axe | | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Amplitude (°) | TX2–60 | 360 | 255 | 285 | 540 | 253.5 | 540 (1) |
| | TX2–60L | | | 305 | | | |
| Limites des articulations (°) | TX2–60 | A ± 180 | B ± 127.5 | C ± 142.5 | D ± 270 | E + 132.5 - 121 | F ± 270 |
| | TX2–60L | | | C ± 152.5 | | | |
| Vitesse nominale (°/s) TX2–60 | | 301 | 301 | 453 | 431 | 336 | 735 |
| Vitesse nominale (°/s) TX2–60L | | 301 | 301 | 453 | 431 | 336 | 735 |
| Vitesse maximale (°/s) TX2–60 (2) | | 435 | 410 | 540 | 995 | 1065 | 1445 |
| Vitesse maximale (°/s) TX2–60L (2) | | 435 | 385 | 500 | 995 | 1065 | 1445 |
| Résolution angulaire (°.$10^{-3}$) | | 0.007 | 0.007 | 0.007 | 0.015 | 0.015 | 0.021 |

# Le bras : volume de travail



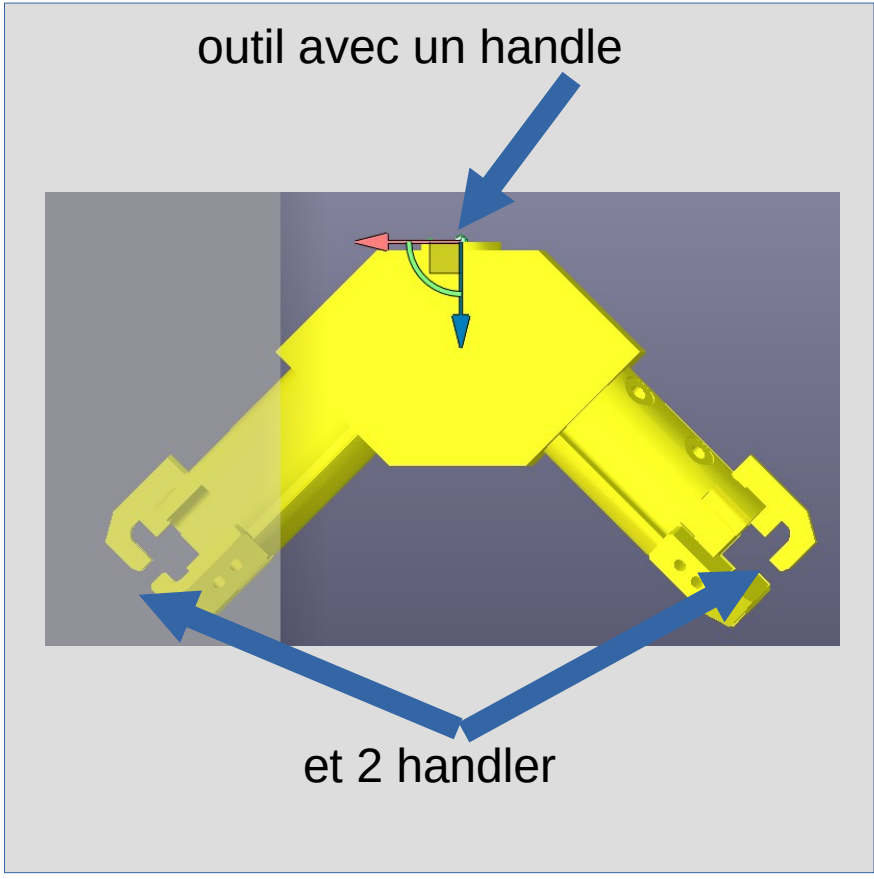




| Espace de travail | Bras standard |
|---|---|
| R.M rayon de travail maxi. entre axes 1 et 5 | 600 mm |
| R.m1 rayon de travail mini. entre axes 1 et 5 | 190 mm |
| R.m2 rayon de travail mini. entre axes 2 et 5 | 189 mm |
| R.b rayon de travail entre axes 3 et 5 | 310 mm |

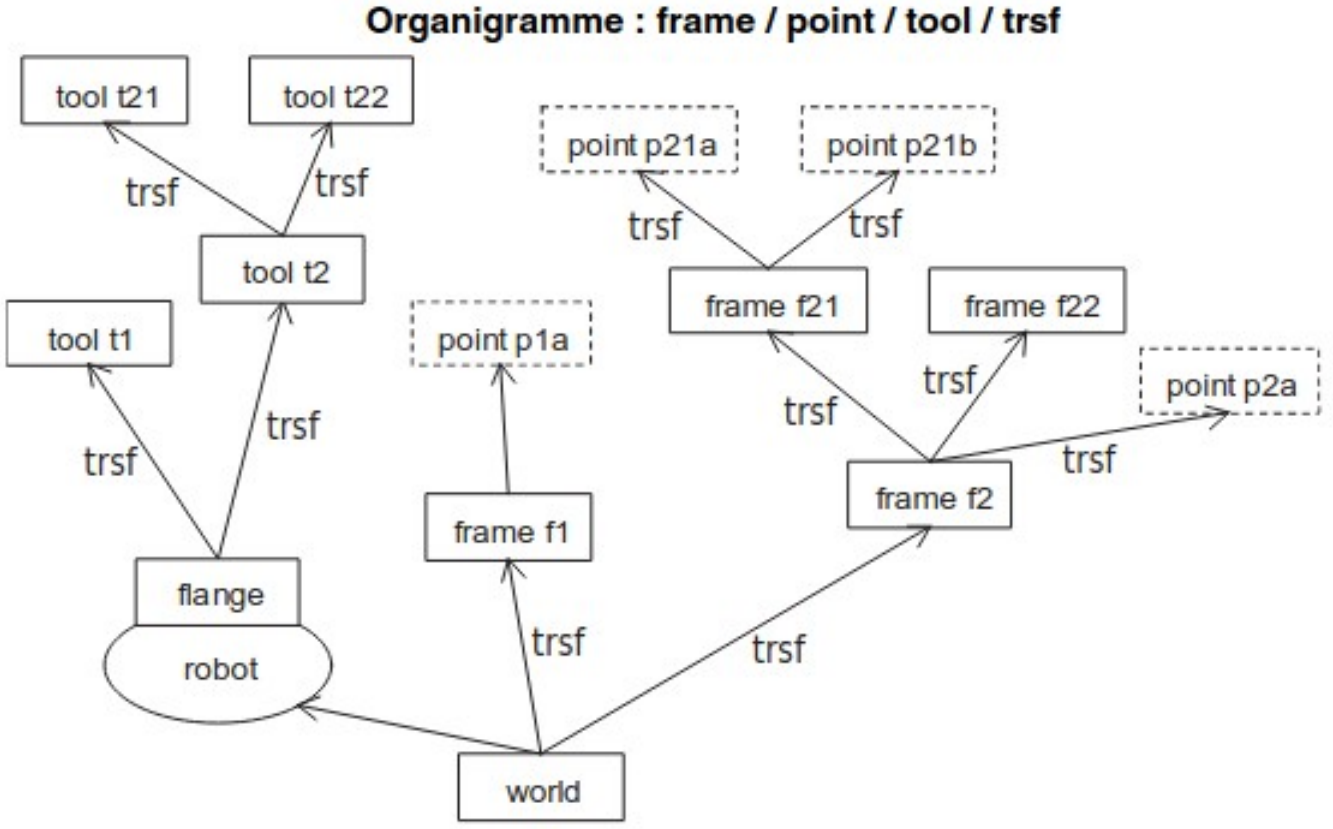
Ne tient pas compte de l'orientation du poignet !

# Outils (organe terminal) et repères

- **TCP: Tool Center Point**
- Repères :
  - Handle : comment l'outil se fixe à la bride (EN: flange)
  - Handler: comment l'outil interagit avec les pièces (dont d'autres outils)
- Un robot a un handle (base) et un handler (bride)
- Les outils ont un handle et au moins un handler.
- Les pièces ont au moins un handle.
- Principe : un handler peut saisir un handle.
- Exemple : le handler d'un robot peut saisir le handle d'un outil.
- Possible de changer d'outils pendant le fonctionnement : changeur d'outils



# Outils (organe terminal) et repères



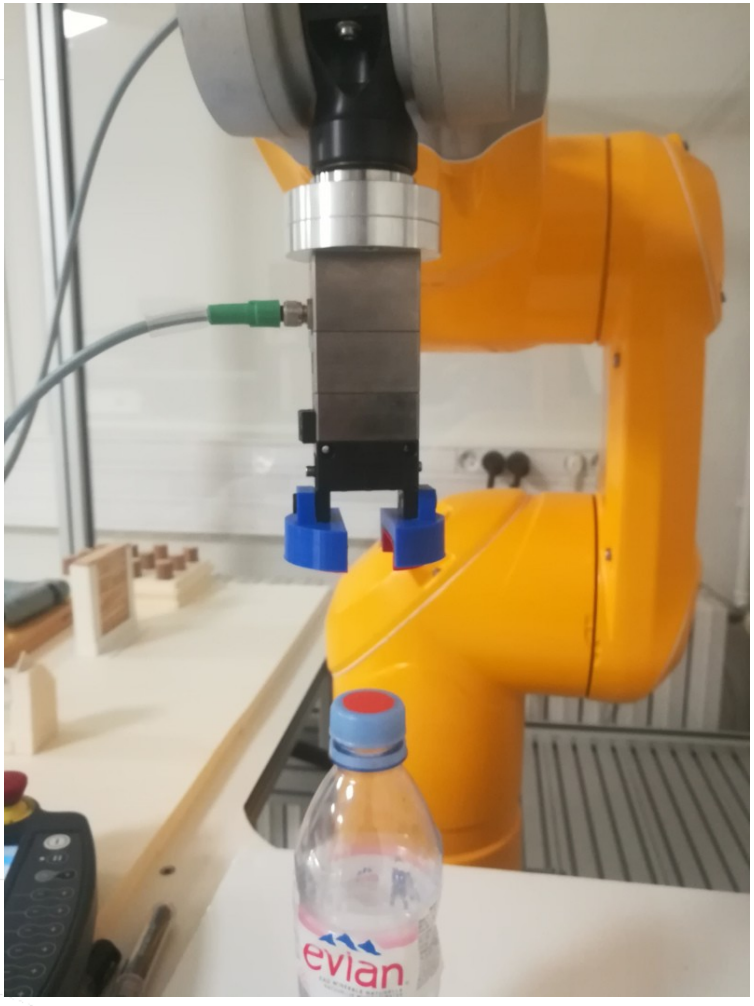

Les repères de construction permettent de définir des points dans ce repère. Le repère entraîne tous ses points avec lui lorsqu'il est déplacé

Exemple d'utilisation : des positions de saisie sur une palette

- Arbre de changement de repères (EN :frame)
- Transformations rigides (trsf)

# Outils

- Différents types d’actions
  - Saisie, peinture, soudure, perçage…
- Cablage, pneumatique...
- Modèle 3D CAO
  - Ex : Pince + mors
- Changement de repère entre le handle et le handler
- Durée d’actuation (ex : ouverture/fermeture pince)
- Sortie physique qui pilote l’actuation



# Boitier sélection de mode de marche (WMS)

- Le robot est potentiellement dangereux (pour lui ou pour l'environnement/opérateur)
- Il faut être sûr de ce que le robot peut faire avant d'entrer dans son volume de travail

- + dispositifs d'affichage de l'état d'activité du robot

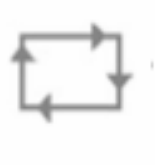
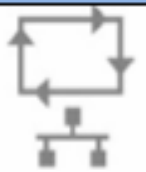
# Boitier sélection de mode de marche (WMS)


S'il est autorisé par la configuration de la sécurité, le sélecteur de mode de marche (WMS9) est nécessaire pour modifier le mode de marche du robot et pour l'acquittement du redémarrage de sécurité. Quand le WMS9 est désactivé par la configuration de la sécurité, le mode de marche du robot est sélectionné avec le boîtier de commande manuel.

Le WMS9 est équipé de 3 dispositifs de sécurité :

- Un sélecteur à clé à 3 positions **(1)** pour la sélection du mode de marche. La clé peut être retirée pour verrouiller le mode de marche sélectionné.
- Un bouton poussoir **(2)** pour l'acquittement du redémarrage après un arrêt d'urgence.
- Un bouton d'arrêt d'urgence **(3)** SIL3/PLe pour forcer le passage du robot à l'état sûr à tout instant.

# Modes de marche

| Mode Automatique (Production) | | Mode Manuel | |
|---|---|---|---|
| **Mode local** | **Mode déporté** | **Mode lent** | **Mode test** |
|  |  |  |  |
| Le robot exécute une application stockée dans le contrôleur | Le robot est commandé par un programme déporté (sur un PC) | Let robot est contrôlé par l'opérateur qui a le boîtier à la main | L'opérateur peut exécuter une certaine tâche étape par étape |

- La vitesse max et la sécurité du robot dépend du mode

# Boitier de commande manuel/pendant (MCP)


Le boîtier d'apprentissage est un terminal portable pour la commande
d’un robot qui offre un moyen pratique pour:

- Déplacer manuellement le robot
- Faire l’apprentissage des points
- Exécuter les programmes
- Le terminal de programmation est un outil très utile, qui permet à l'utilisateur de s’éloigner du terminal de l'ordinateur hôte et de contrôler le robot à distance

# Boitier de commande manuel/pendant (MCP)



Le SP2 MCP (boîtier de commande manuelle) est nécessaire pour contrôler le robot manuellement. Il peut également être utilisé comme interface de production en mode automatique.

**Emplacement et manipulation du MCP**

Le MCP est équipé de 2 dispositifs de sécurité :

- Un bouton d'arrêt d'urgence **(1)** SIL3/PLe pour forcer le passage du robot à l'état sûr à tout instant.
- Un bouton de validation SIL3/PLe **(2)** permettant au robot de quitter l'état de sécurité en mode manuel. L'emplacement du bouton de validation permet une utilisation par une personne gauchère ou droitière.

En fonction de la configuration de la sécurité de la cellule, le mode de marche du robot peut également être sélectionné avec l'interface du MCP.

Un stylet est fourni avec chaque SP2. Celui-ci doit être utilisé uniquement avec l'écran tactile.

Il ne doit pas être utilisé pour activer les touches du clavier.

# Boitier de commande manuel (MCP)


**Bouton de mise sous puissance du bras (1)**

Ce bouton éclairé permet d'activer ou de désactiver la mise sous puissance du bras. Quand le témoin blanc est allumé fixe, le bras est sous puissance.

**Arrêt d'urgence (2)**

Le bouton d'arrêt d'urgence déclenche un arrêt contrôlé du bras et maintient le robot à l'état sûr. Un acquittement de redémarrage de sécurité est nécessaire pour reprendre le fonctionnement.

**Touches de mouvements (3)**

Ces touches sont activées en mode vitesse réduite manuelle et contrôlent le mouvement du bras sur chaque axe ou selon des coordonnées cartésiennes, en fonction du mode de mouvement sélectionné (voir les chapitres 6.3.3.2 et 6.3.3.3).

**Touche d'ajustement de la vitesse (4)**

Cette touche permet de faire varier la vitesse dans la limite imposée par le mode de déplacement.

La vitesse en cours est affichée dans la barre d'état du MCP.

La vitesse varie en fonction de valeurs prédéfinies ou peut être éditée en appuyant sur la valeur affichée.

**Touches d'interface et de navigation (5)**

**Touche Accueil**
La touche accueil permet de passer au menu principal.

**Touche Retour**
La touche Retour permet d'annuler les dernières modifications ou de revenir à la page précédente.

**Touche Menu**
La touche menu ouvre le menu contextuel de la page correspondante (si la page contient un menu contextuel, ce qui n'est pas toujours le cas pour toutes les pages).

# Boitier de commande manuel (MCP)


**Bouton de validation (6)**

Le bouton de validation commande un arrêt de protection en mode manuel. Quand il est actionné en position médiane, l'arrêt de protection est annulé et il est possible de remettre le bras sous tension, ou de desserrer un frein. L'arrêt de protection est activé quand le dispositif n'est pas actionné, ou appuyé à fond (panique).

L'arrêt de protection du bouton de validation peut être configuré pour un arrêt SS1 (bras hors tension) ou pour un arrêt SS2 (arrêt sécurisé sous tension).

Le boîtier de commande manuelle est conçu pour permettre une utilisation par des opérateurs droitiers ou des opérateurs gauchers.

**Touches d'activation de sorties digitales (7)**


En mode manuel, ces touches font basculer l'état des sorties digitales qui leur sont associées.

A l'IUT, la pince est pilotée par le bouton 2

**Touche mouvement / pause (8)**

La touche mouvement / pause commande les déplacements programmés quand le bras est sous tension.
La led clignote quand les mouvements programmés sont en pause. Elle s'allume fixe quand le programme en cours commande les mouvements du bras.

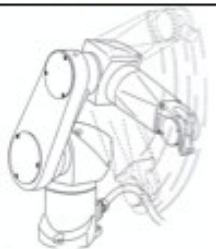
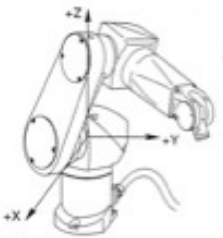
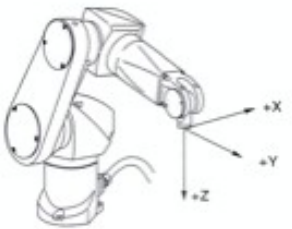
- En mode manuel, pour les mouvements de connexion, la touche doit être maintenue appuyée pour permettre les mouvements du bras.
- En mode automatique local, l'appui sur la touche permet de basculer entre l'état de mouvement et l'état de pause.

# Boitier de commande manuel (MCP)


Différents profils d’utilisateurs pour :
-Lancer des applications
-Apprendre des points
-Définir des butées logicielles sur les articulations
-Configurer les Entrées/Sorties
-Recalibrer le robot

# Boitier de commande manuel (MCP)

Modes de déplacement

| Joint | Frame | Tool | Point |
|---|---|---|---|
|  |  |  |  |
| Déplacer le robot par rapport aux axes des articulations | Déplacer le robot par rapport à son repère de base (Z vers le haut) | Déplacer le robot par rapport au repère outil mobile (Z vers l'extérieur) | Déplacer le robot jusqu'à un certain point ou position articulaire (déjà enregistré) |
|  |  |  | |

Vitesse de déplacement

Il faut préalablement avoir choisi l'outil

Il faut préalablement avoir choisi le point

# Tâche robotique

- Définir ce que le robot doit faire (ou pas)
- Comment il interagit avec :
  - Les pièces sur lesquelles il agit
  - L’environnement (les éléments fixes et mobiles/variables de la cellule)
  - Les éventuels opérateurs humains
- Choix des capteurs et actionneurs
- Le robot est généralement un élément d’une chaîne :
  - Gestion des flux de pièces en entrée et sortie

# Tâche robotique

- Une même cellule robotique peut servir à réaliser plusieurs tâches différentes dans différentes parties de la cellule
- Robot Collaboratif avec des humains
  - Certains volumes peuvent être interdits (ou nécessiter une vitesse réduite) lorsqu'un humain y intervient
- Tâche partagée entre plusieurs robots
  - Séquentiellement ou vrai coopération
- En contexte industriel :
  - Environnement du robot maîtrisé le plus possible
    - Éviter l'incertain : position/orientation des pièces, humains éloignés
    - Reproductibilité des mouvements du robot
    - Répétabilité > Précision
  - Rapidité d'exécution d'une tâche répétitive (optimisation du temps de cycle)
  - Intervenir sur le robot peut nécessiter d'interrompre une chaîne de production : Réapprentissage de points, changement de programme, maintenance…
    - Intérêt de la PHL

# Étapes de la programmation du robot

1) Définition de la tâche robotique (définir le maximum de contraintes en amont)
2) Identification des sous tâches (décomposition)
3) Positionnement des éléments de la cellule (en simulation idéalement et dimensionnement (choix) du robot)
4) Paramétrisation des trajectoires
   - Quels sont les points « réglables » et les repères dans lesquels les définir
5) Programmation  (Approche Coarse to Fine : trajectoires grossières et/ou saccadées dans un premier temps)
6) Simulation
7) Apprentissage des points et repères sur le vrai robot
8) Adaptation de la simulation au points mesurés sur le vrai robot
9) test sur le vrai robot à vitesse réduite
10) Mesure des performances en simulation et/ou en réel (par exemple temps de cycle)

A  chaque étape, il est possible de rétroagir sur les étapes précédentes

Approche modulaire (Idéalement): Simulation des éléments matériels et logiciels en interaction avec le robot

Simulation de la tâche idéalement en amont de la construction de la cellule (par exemple pour choisir le bon robot après en avoir essayé plusieurs, et positionner au mieux les éléments de la cellule)

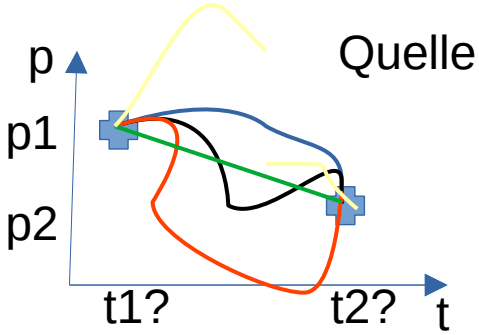
# Programmation du robot

- Approche simpliste 1 (mais non disponible directement sur le Staubli) : Apprentissage par démonstration
  - L’opérateur déplace le robot à la main (littéralement) pour amener l’outil en différentes positions en passant par des trajectoires
  - Le robot compense son propre poids
  - Un capteur d’efforts est nécessaire pour mesurer les mouvements du robot déplacé par l’opérateur

- Le robot rejoue les trajectoires (éventuellement à une vitesse différente)

# Programmation du robot

- Approche simpliste 2 (mais non disponible directement sur le Staubli) : Apprentissage au pendant
  - L'opérateur déplace le robot avec le pendant pour amener l'outil en différentes positions
  - Les positions sont enregistrées dans des variables
- Le robot rejoue les trajectoires passant par les points appris




Quelle trajectoire entre les points ?

Vitesse constante
Trajectoire lisse
Trajectoire avec variation instantanée de vitesse
Ceci n'est pas une trajectoire
Ceci n'est pas une trajectoire

Quelles dates pour les points ?

# Programmation du robot

- Programme exécuté par le contrôleur (compétence: informatique)
- Langage et outils propre à chaque marque de robot
- Programme simple:
  - Commande en boucle ouverte : enchaînement d'ordres de mouvements figés
- Programme plus complexe :
  - Adaptation aux paramètres variables (les points deviennent variables)
  - Mesure par capteur en début de tâche
    - Ex :Travail relatif à la pièce, par exemple perçage perpendiculairement à la surface
- Commande en boucle fermée (compétence:automatique)
  - Mesure par capteur pendant la tâche
    - Ex :Travail sur une pièce en mouvement (suivi)

- Description de trajectoire (compétence: mathématique) : métier « trajectoiriste »
- Prise en compte des contraintes physiques: inertie mécanique, spécificités liées à l'outil etc. (compétence: mécanique)

# Trajectoire

- Description du mouvement effectué par le robot (et l'outil)
  - Courbe continue paramétrée par des points discrets
  - Dans l'espace articulaire : valeur de chaque articulation à chaque instant (6ddl)

MGD Univoque : A chaque position articulaire correspond une position cartésienne

MGI NON Univoque : A chaque position cartésienne correspondent zéro à plusieurs positions articulaires

  - Dans l'espace cartésien: position et orientation du repère outils (TCP) à chaque instant (6ddl)
- Décrite par des portions élémentaires
  - Elles mêmes paramétrisées par des points (positions cartésiennes ou articulaires):
  - Interpolation dans l'espace cartésien pour maîtriser la trajectoire de l'outil :
    - Mouvement en ligne droite
    - Mouvement en cercle
    - Mouvement type courbes polynomiales...
  - Interpolation dans l'espace articulaire
    - Mouvement au plus rapide (indépendamment sur chaque articulation)
    - Pas de risque de singularité
    - Trajectoire de l'outil peut être erratique
  - Descripteur de mouvement :
    - Vitesse et accélération/décélération maximales, type d'interpolation, tolérances pour le lissage...

Staubli appelle :
-**Joint** une position articulaire
-**Point** une position cartésienne pour le repère outil associée à une configuration du robot

# Points

- Définis en articulaire:
  - Indépendant de tout repère mais effet dépendant de la position du robot relativement à la scène
  - Permet de contraindre complètement le robot (pas d'ambiguïté)
  - Permet de spécifier des orientations au delà de +-180 degrés (pratique par exemple pour un poignet qui visse)
  - pas facile à adapter en cas de changement de la scène : déplacer la pièce à saisir nécessite de réapprendre la position articulaire du robot pour saisir la pièce
- Définis en cartésien:
  - Position et orientation du repère outil (TCP)
  - Associé à un repère (Monde ou autre) et relativement indépendant de la position du robot
  - Ne contraint pas complètement le robot (solutions multiples pour le Modèle Géométrique Inverse, nécessite des informations supplémentaires pour sélectionner parmi les solutions)
  - Pas toujours atteignable (absence de solutions pour le Modèle Géométrique Inverse)
  - Ne permet pas de spécifier des orientations au delà de +-180 degrés (modulo)
  - Facile à adapter en cas de changement de la scène
  - Peut être calculé relativement à un autre point (par exemple 4cm au dessus d'un point  perpendiculairement a un plan)

# Points définis en articulaire

- Type **Joint** chez Staubli
- La consigne articulaire est représentée par :
  - Un vecteur à 6 composantes, chacune indiquant la valeur articulaire d’une articulation en degré, avec possibilité d’indiquer des angles sur plusieurs tours si les articulations du robot le permettent

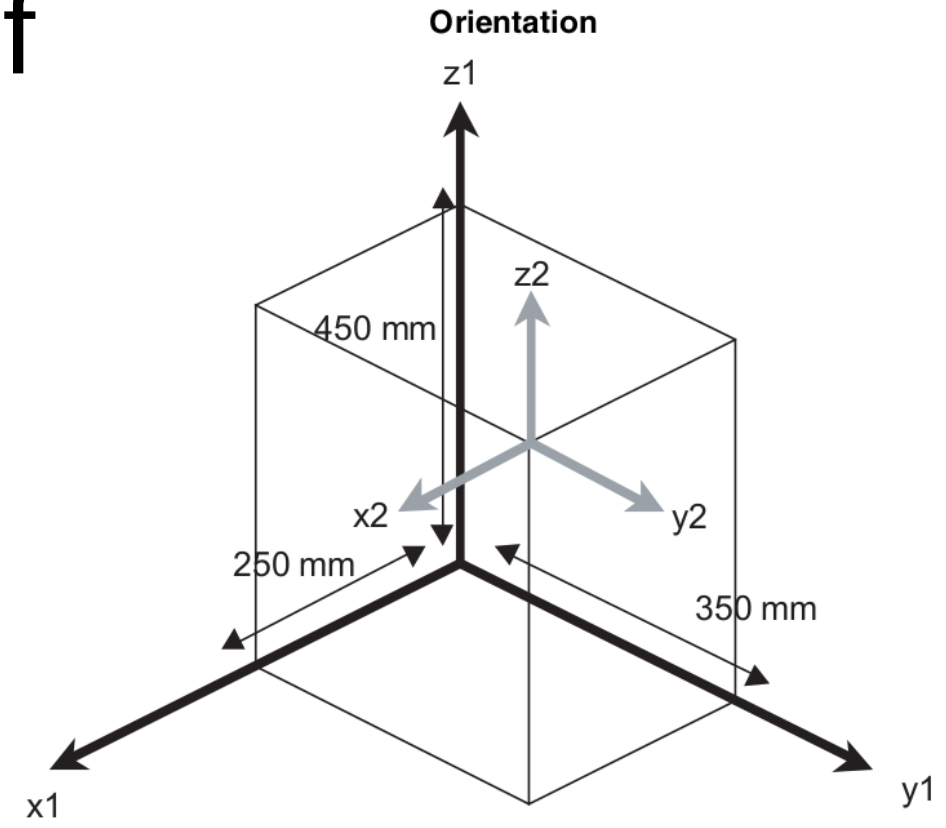
# Points définis en cartésiens

- Type **Point** chez Staubli
- La consigne cartésienne est représentée par :
  - Une pose (position et orientation du repère outils dans un repère de référence) codée sous forme d'une transformation rigide (type **trsf** chez Staubli, qui sert aussi à indiquer les changements de repère entre **frame**)
  - Une valeur de configuration permettant de sélectionner parmi les multiples configurations articulaires permettant d'atteindre la pose (type **config** chez Staubli)
    - Contrainte (lefty/righty), libre (free), identique à celle du robot précédente (same)

# trsf

- Une transformation (type trsf) décrit un changement de position et/ou d'orientation. C'est l'assemblage mathématique d'une translation et d'une rotation.
- La transformation ne représente pas elle-même une position dans l'espace, mais elle peut être interprétée comme la position et l'orientation d'un point ou d'un repère cartésien par rapport à un autre repère.
- Le type ttrsf est un type structuré dont les champs sont, dans l'ordre :
  - num xTranslation sur l'axe x
  - num yTranslation sur l'axe y
  - num zTranslation sur l'axe z
  - num rxRotation autour de l'axe x
  - num ryRotation autour de l'axe y
  - num rzRotation autour de l'axe z
- Les champs x, y et z sont exprimés dans l'unité de longueur de l'application (millimètre ou inch, voir chapitre Unité de longueur). Les champs rx, ry et rz sont exprimés en degrés.
- Les coordonnées x, y et z sont les coordonnées cartésiennes de la translation (ou la position d'un point ou d'un repère dans le repère de référence). Lorsque rx, ry et rz sont nuls, la transformation est une translation sans changement d'orientation.
- Par défaut, une variable de type trsf est initialisée à la valeur {0,0,0,0,0,0}.

# trsf



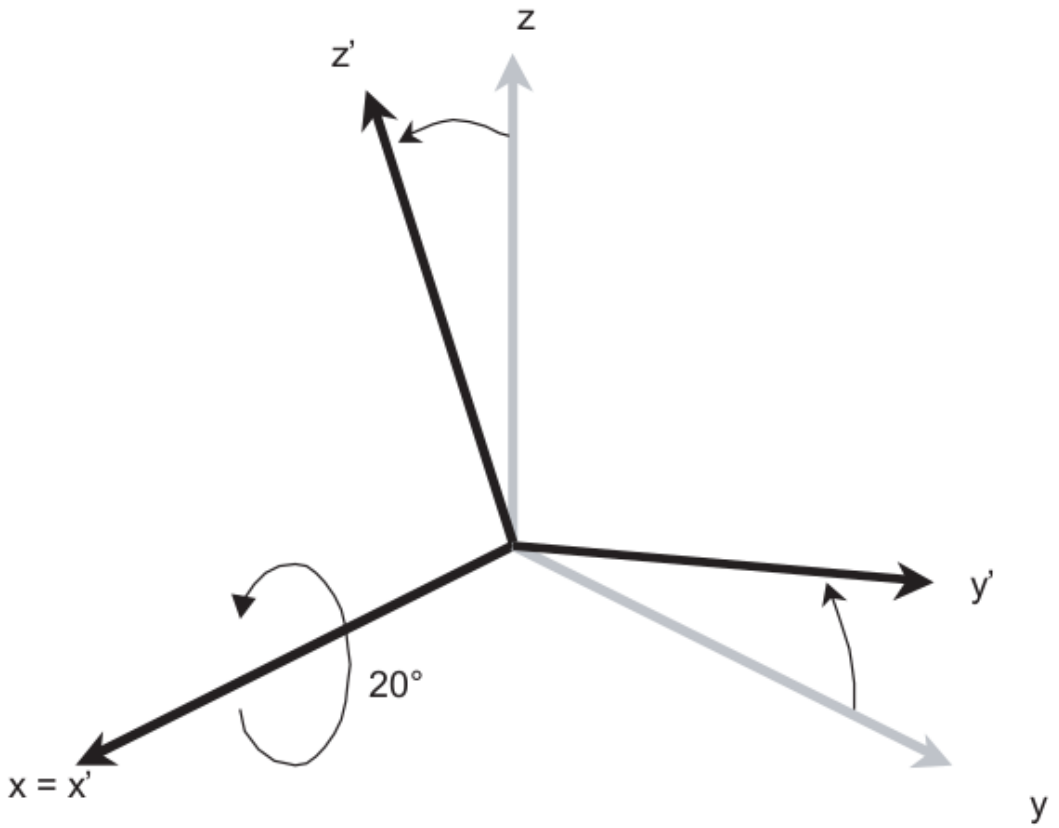

La position du repère **R2**(gris) par rapport à **R1**(noir) est :
x = 250mm, y = 350 mm, z = 450mm, rx = 0 , ry = 0 , rz = 0

Les coordonnées **rx, ry** et **rz** correspondent aux angles de rotation qui doivent être appliqués successivement autour des axes **x, y** et **z** pour obtenir l'orientation du repère.

Par exemple, l'orientation **rx = 20 , ry = 10 , rz = 30** est obtenue de la manière suivante. Le repère **(x,y,z)** est d'abord tourné de **20** autour de l'axe **x**. On obtient un nouveau repère **(x',y',z')**. Les axes **x** et **x'** sont confondus.

trsf

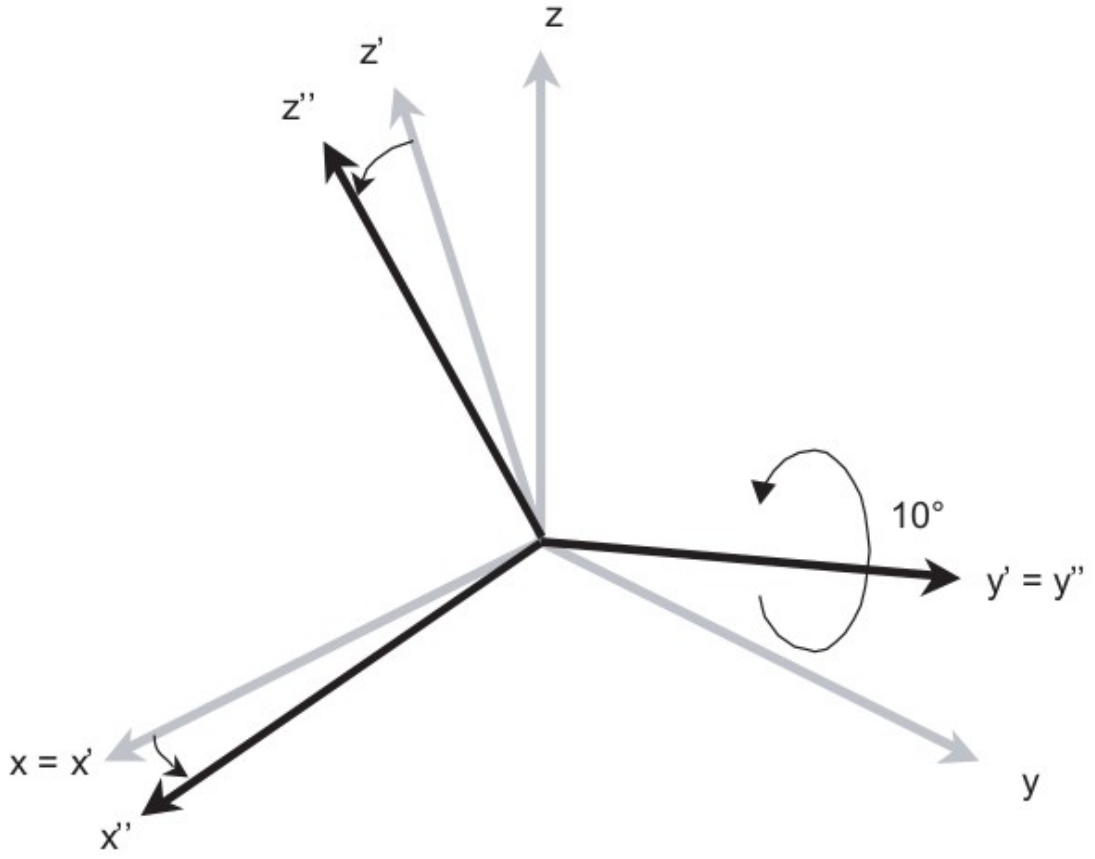
**Rotation repère par rapport à l'axe : X**




Le repère est ensuite tourné de **20** autour de l'axe **y'** du repère obtenu dans l'étape précédente. On obtient un nouveau repère **(x'',y'',z'')**. Les axes **y'** et **y''** sont confondus.

trsf

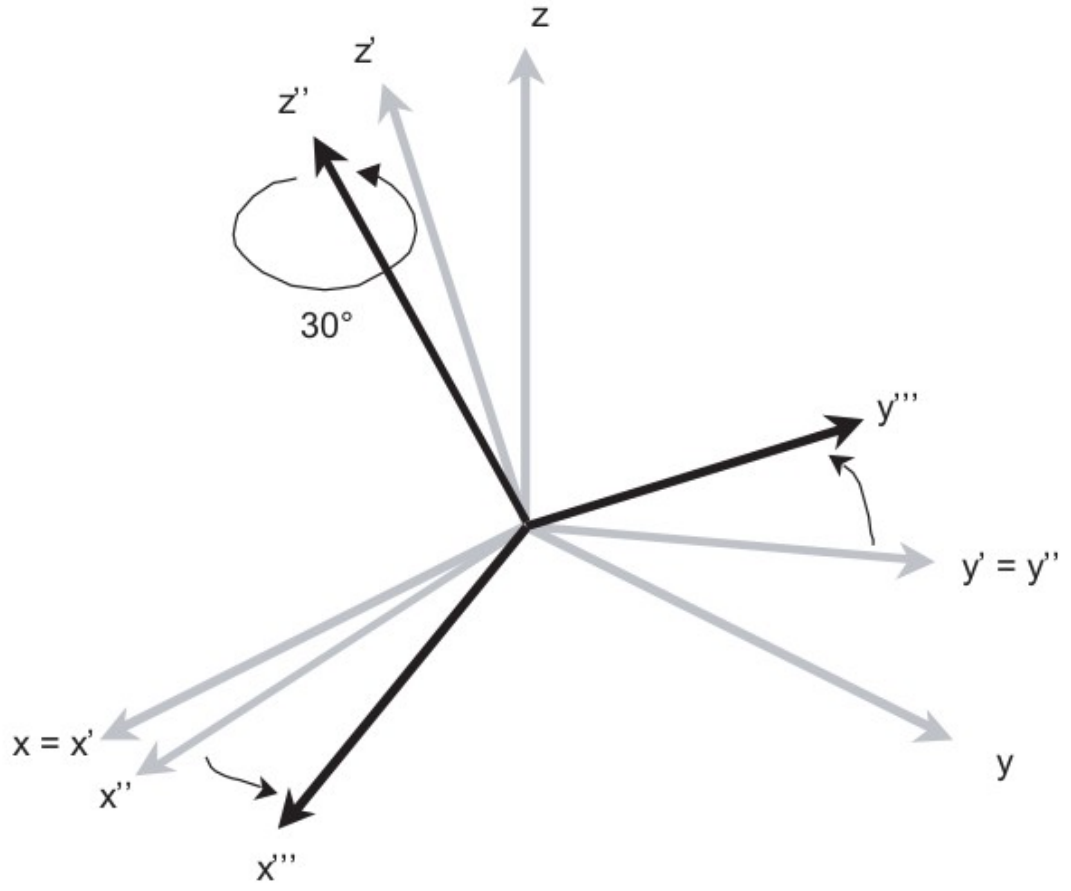
**Rotation repère par rapport à l'axe : Y'**




Enfin, le repère est tourné de **20** autour de l'axe **z''** du repère qu'on a obtenu à l'étape précédente. Le nouveau repère **(x''',y''',z''')** obtenu est celui dont l'orientation est définie par **rx, ry, rz**. Les axes **z''** et **z'''** sont confondus.

trsf

**Rotation repère par rapport à l'axe : Z''**

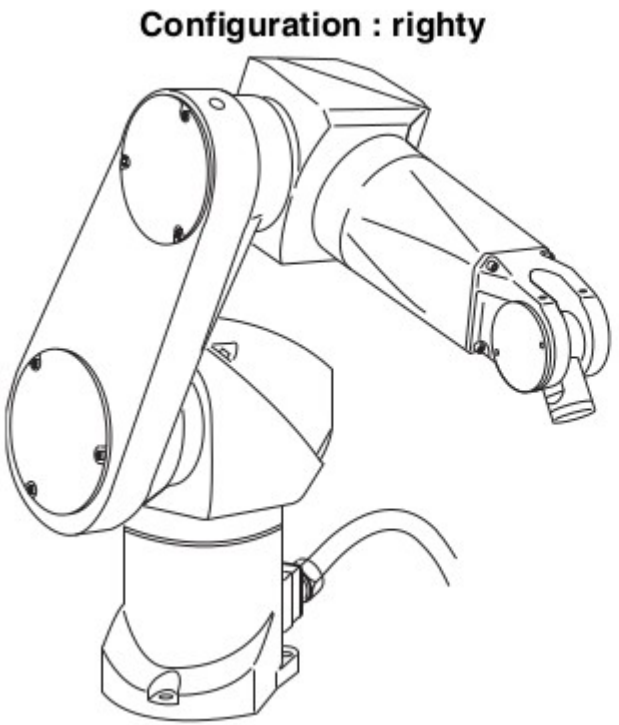


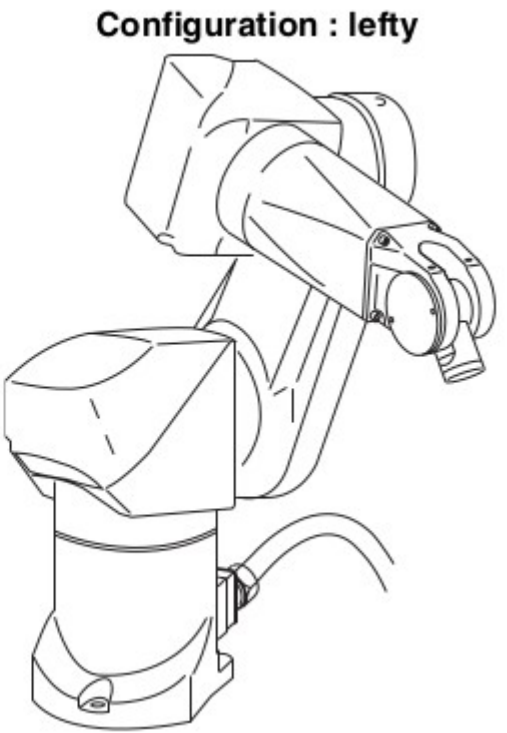

La position du repère **R2**(gris) par rapport à **R1**(noir) est :
x = 250mm, y = 350 mm, z = 450mm, rx = 20 , ry = 10 , rz = 30

# trsf

Les valeurs de **rx, ry** et **rz** sont définies modulo **360** degrés. Lorsque le système calcule **rx, ry** et **rz**, leurs valeurs sont toujours comprises entre **-180** et **+180** degrés. Il reste alors encore plusieurs valeurs possibles pour **rx, ry,** et **rz** : Le système garantit qu'au moins deux coordonnées se situent entre **-90** et **90** degrés (sauf si **rx** vaut **+180** et si **ry** vaut **0**). Lorsque **ry** vaut **90** degrés **(modulo 180)**, **rx** est choisi nul.

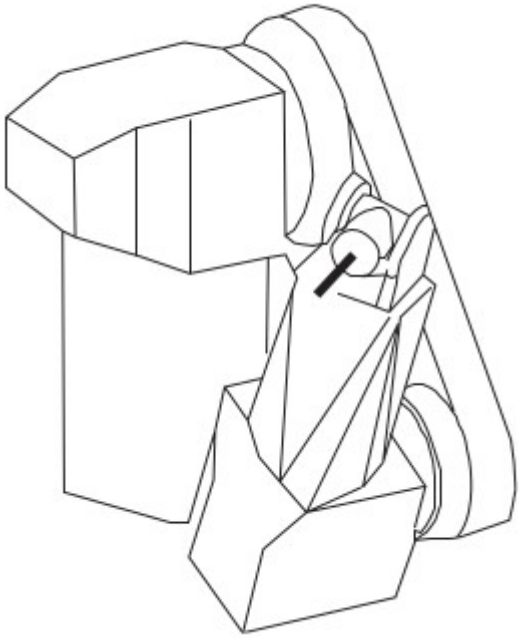
# Configurations du robot : épaule (EN : shoulder)



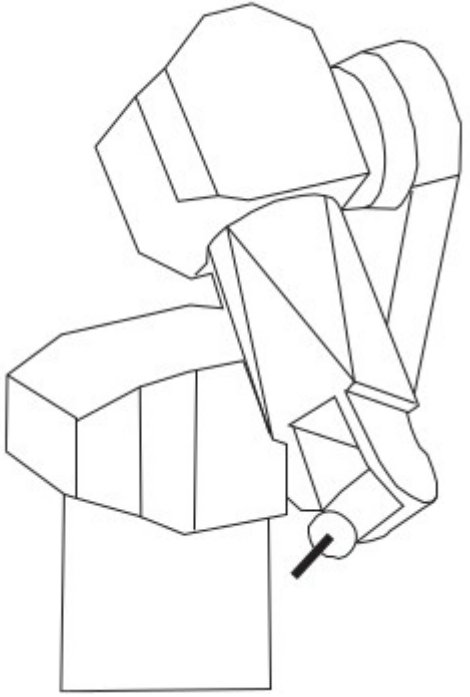



# Configurations du robot : coude (EN : elbow)

Configuration : enegative

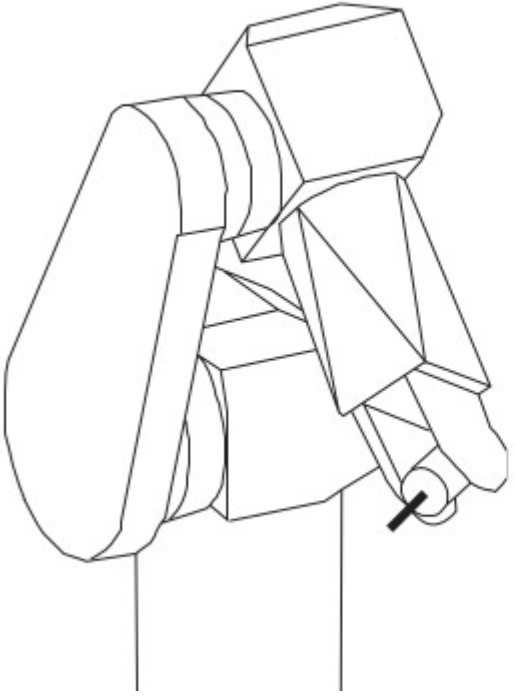


Configuration : epositive

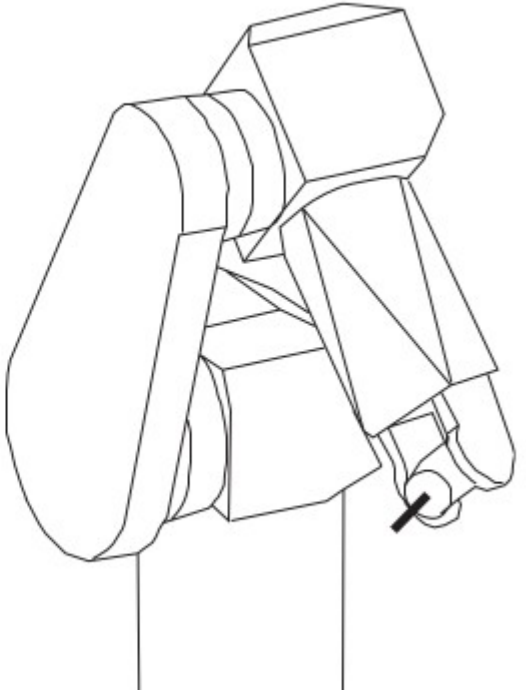


# Configurations du robot : poignet (EN : wrist)

Configuration : wnegative



Configuration : wpositive



# Interpolation de tronçon de trajectoire

- Type d'ordre de mouvement :
  - Interpolation en articulaire (**movej**) aussi appelé **point à point**:
    - Toujours possible en termes de calcul
    - Peut générer des trajectoires avec des vitesses linéaires importantes (ex : bras tendu)
    - Absence de problème de singularité
    - Adapté lorsque le robot :
      - Est loin des collisions avec l'environnement
      - Doit bouger rapidement
      - Est proche d'une singularité (exemple : pour sortir de la configuration bras tendu)
  - Interpolation en cartésien (**movel,movec**):
    - Pas toujours possible en termes de calcul
    - Peut générer des trajectoires pour lesquelles l'outil suit une trajectoire bien déterminée
    - Problème de singularité
      - Perte de degré de liberté pour certaines positions du robotr
      - Impossible de garantir la vitesse linéaire car correspond à vitesse articulaire infinie au niveau de la singularité
    - Adapté lorsque le robot :
      - Est proche des collisions avec l'environnement (voir au contact)
      - Doit déplacer l'outil de manière maîtrisée
      - Est loin d'une singularité (exemple : poignet bien « cassé »)

# Interpolation de tronçon de trajectoire : descripteur de mouvement

- Type **mdesc** chez Staubli
- Associé individuellement à chaque ordre de mouvement vers un point
- Permet de régler:
  - les accélérations/décélérations/vitesses linéaires et angulaires
  - Le lissage de trajectoire :
    - Désactivé, articulaire ou cartésien
    - Quelles tolérances sont autorisées avant d'arriver à un point en mm (**reach** chez staubli) et après le point pour retourner sur la trajectoire de consigne (**leave** chez staubli)

# Récapitulatif : Trajectoire

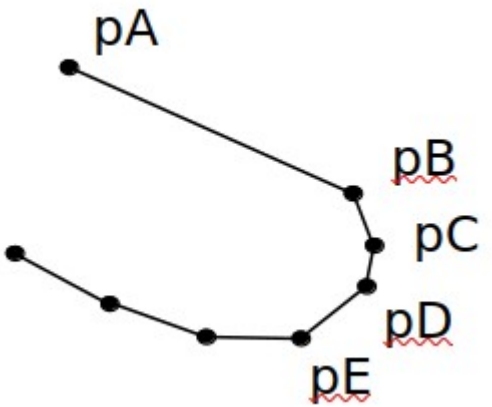
- La trajectoire est une courbe continue paramétrée par :
  - une suite d'ordres de mouvements élémentaires définis :
    - par des points discrets (définis en articulaires ou cartésiens)
    - reliés par des tronçons de trajectoire continus obtenus par interpolation (dans l'espace articulaire ou cartésien)
    - associés à des tolérances pour que le robot soit autorisé à passer plus ou moins loin des points (lissage, lui même réalisé dans l'espace articulaire ou cartésien)

# Trajectoire

- Les ordres de mouvements du programme sont déposés dans une file d'attente (FIFO)
  - Évite les carences de données
  - Possibilité d'anticipation
  - Possibilité de lissage/approximation
- Le(s) programme(s) de l'utilisateur interagi(ssen)t avec la FIFO :
  - Dépôt d'ordres
  - Synchronisation par attente de vidage (par exemple pour commander la pince une fois que le robot est arrivé sur la pièce)
  - Consultation de l'indice en cours d'exécution
  - Communication avec l'extérieur…
  - Ce n'est pas (directement) votre programme qui fait bouger le robot !
- Un autre programme du contrôleur pilote les axes du robots en prélevant les ordres depuis la FIFO
- Programmation Parallèle (multitâche)

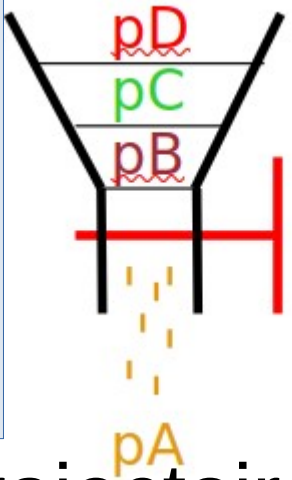
# Trajectoire

Le programme utilisateur dépose des instructions de mouvements (movej,movel,movec) associées à des points et des descripteurs de mouvements



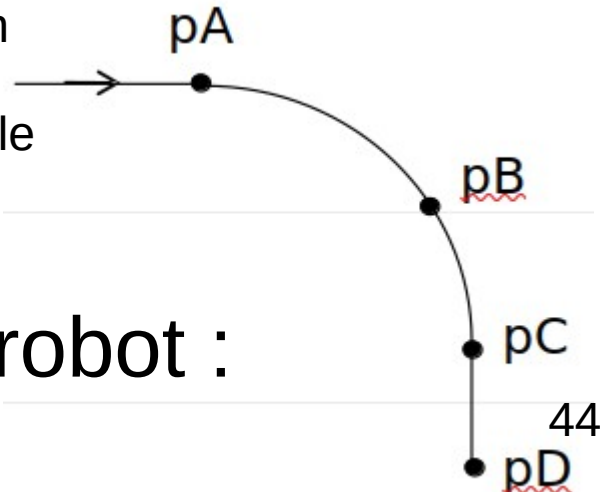

Instructions de réglage de la vitesse : SetMonitorSpeed, getMonitorSpeed

Instructions d'interaction avec FIFO : waitEndMove (attente vidage et pas de lissage),
isEmpty, isSettled (consultation),
getMoveId,setMoveId (numérotation)
stopMove/restartMove (pause),
resetMotion (vidage sans mouvement)





Mise en pause manuelle




## Réalisation de la trajectoire par le robot :

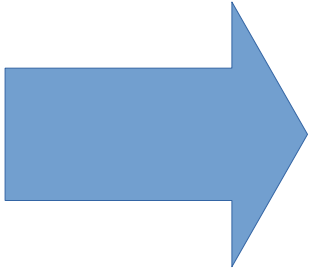
# Gestion des modes d’arrêt

- Arrêt d’urgence (pression du bouton ou pénétration dans une zone de sécurité)
  - Activation de freins mécaniques
- Ordre impossible
  - Hors d’atteinte du bras
    - trop loin en cartésien, hors des butées en articulaire, mauvaise orientation, auto-collision
  - Dans un volume interdit
- Détection de collision
  - Consommation excessive des moteurs
  - Capteur (ex : peau)
- Détection de sortie de l'enveloppe de trajectoire
- Durée trop longue (Timeout) pour la communication avec capteur/actionneur

- Nécessité d’une reprise après arrêt (souvent manuelle) pour remettre le robot dans une configuration permettant le redémarrage du programme
- « Mouvement de connexion » pour retourner au dernier point connu de la trajectoire si le robot a été déplacé manuellement.

# IDE Staubli Robotic Suite

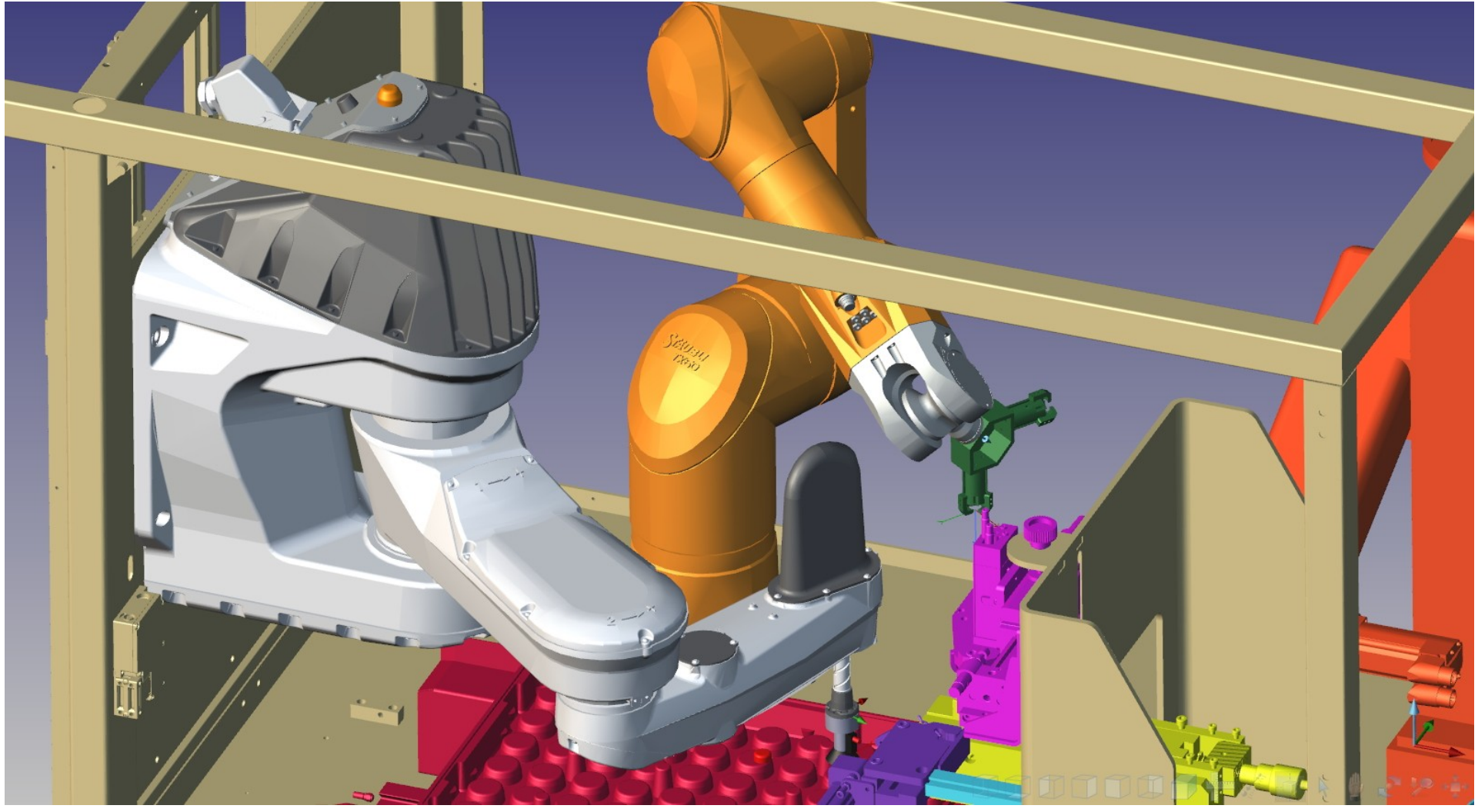
- Application intégrée pour la Programmation Hors Ligne et le téléchargements vers/depuis le contrôleur
  - IDE clickodrome...
- Langage et outils de développement propriétaires (et payant)
  - 14 licences utilisables au département
  - peu de code source disponible par la communauté d'utilisateurs
  - peu d'aide accessible en ligne
  - Documentation pas toujours complète
- Notion de projet intégrant :
  - Le(s) robot(s) et les éléments de la cellule
  - Les variables (données) et le code (programmes)
  - Les paramètres de configuration du(es) contrôleur(s)
    - Par exemple liens entre variables et ressources physiques (I/O, réseau etc...)
- Projet stocké dans des fichiers texte .xml
  - Un fichier projet (PJX)
  - Un fichier de données (DTX)
  - Plusieurs fichiers de programme/fonction (PGX)



Suivi de versions possible et très conseillé avec l'outil GIT

# Programmation Hors Ligne

- Configuration de la cellule
  - Modèles 3D colorés et hiérarchisés (.WRL)
    - Cage, tapis, socle du robot, table, supports etc...
  - Choix du type de robot(s) et type de fixation
  - Supports, positions initiales des pièces, limites de la cellule….
    - Pour l'affichage animé
    - Pour la détection des collisions
    - Pour la mécanique : masse et inertie des pièces
  - Organisation hiérarchiques des éléments
    - Description des liens entre repères (frame) et points.
- Éditeur 3D intégré
  - Très limité dans SRS, utiliser un outil de CAO 3D externe pour modéliser les éléments
  - Positionnement modifiable des éléments
- Visualisation du volume accessible par le robot
  - En position par le point à l'intersection des axes du poignet
- Éléments de sécurité: scrutateurs laser, contacteurs de portes...



# Programmation Hors Ligne

- Exécution du programme en simulation
  - Émulation du contrôleur du robot et du MCP
    - Possibilité d’apprendre à piloter manuellement le robot en simulation
    - IHM du MCP active pendant la simulation
  - Simulation de la scène 3D
    - Tracé des positions successives du repère de l’outil dans la cellule
    - Mesure des plages de valeurs articulaires du robot pendant les cycles
    - Détection des collisions (définition de volumes d’avertissement)
    - Effort mécaniques par simulation "physique": le simulateur connaît la distribution de masses des éléments des différents robots, il est capable de simuler l'inertie. Il est possible de spécifier la masse ou la densité de l'outil et des pièces manipulées par le robot
    - Mesure des temps de cycle
    - Génération de vidéos

# Programmation Hors Ligne

- Exécution du programme en simulation
  - Possibilité de points d’arrêt dans le programme
  - Observation/modification des variables
  - Accès aux positions angulaires et articulaires pendant l’exécution
    - ATTENTION: par défaut l’appel d’une fonction de mouvement n’est pas bloquante, elle positionne juste l’ordre dans la FIFO, et le robot n’effectue pas l’intégralité du mouvement jusqu’à ce point (à cause du lissage)

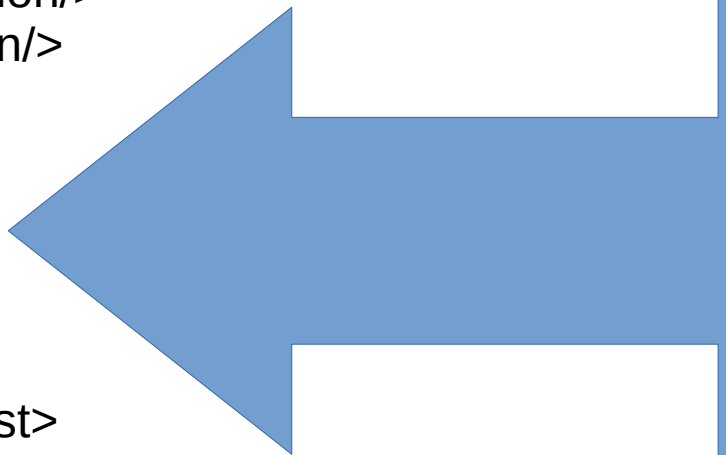
# Langage de Programmation VAL3

- Langage interprété
- Limité en terme algorithmique :
  - Pas de déclaration de variable ou de prototype dans le code (clickodrome)
  - Une fonction est un programme et doit être décrite dans un fichier indépendant
- Documentation complète: https://bvdp.inetdoc.net/files/staubli/documentations/Val3.PDF

# Exemple de Programme VAL3

```
<?xml version="1.0" encoding="utf-8" ?>
<programList xmlns="ProgramNameSpace”>
<program name="demo" public="false”>
<description/>
<paramSection/>
<localSection/>
<source>
<code>

</code>
</source>
</program>
</programList>
```



```
Begin
//sortie de configuration singuliere par
mouvement en espace articulaire
movej(p[3],flange,mNomSpeed)
WaitEndMove()
While true
//répétition d’un cycle
  movel(p[0],flange,mNomSpeed)
  movel(p[1],flange,mNomSpeed)
  movej(p[3],flange,mNomSpeed)
  movel(p[0],flange,mNomSpeed)
  movec(p[2],p[1],flange,mNomSpeed)
  movej(p[3],flange,mNomSpeed)
  waitEndMove()
endWhile
end
```

# Communication en VAL3

- Communication avec l'extérieur gérée à l'aide de variables associées à des ressources de communication
- Permet de récupérer l'état de capteurs, de piloter des actionneurs, de dialoguer avec un automate/PC
- E/S digitales ↔ variable booléenne (type **dio** chez Staubli)
- Réseau
  - Sockets UDP/TCP et ports série ↔ variables chaînes de caractères particulières (type **sio** chez Staubli)
  - ...

# Nommage des variables

- **Vous veillerez à respecter les règles de nommages des variables en fonction de leurs types (voici ce que nous manipulerons):**
- variable bool (booléen) commençant par b
- variable frame (repère) commençant par f
- variable jointRx (configuration en articulaire) commençant par j
- variable pointRx (configuration en cartésien) commençant par p
- variable mdesc (descripteur de mouvement) commençant par m
- variable num (valeur numérique) commençant par n
- variable string (chaine de caractères) commençant par s
- variable tool (outils) commençant par t
- variable trsf (transformation) commençant par tr
- variable sio (ressource de communication) commençant par si

# Nommage des variables

- Ceci permettra notamment d'ajouter automatiquement une variable à partir du code en faisant clic droit sur la variable puis “ajouter donnée”. Le type de la variable est alors inféré d'après la convention de nommage.
- Les noms des variables locales sont de plus préfixés par l_ à ajouter en tant que nouvelle variable locale (pas donnée).Nous Utiliserons les variables locales pour les variables qui n'ont pas à être réglées depuis le pendant (par exemple les variables numériques de compteurs). Dans le projet, les variables locales n'apparaissent pas dans les “Données” mais dans “Cellule”→Controller1→Projet→start()→Variables locales.
- Les nom des paramètres sont préfixés par x_.

TODO :

Trsf : lien avec matrices

Open/close de la pince → waitEndMove
Sous programme et paramètres, Paramètres passés par valeur et référence

Frame et points relatifs aux frames : dire qu'un point est une pose de repère outils souhaité

Exemple de trajectoire de type moveL entre 2 points (dessiner plan x,y puis x(t) et y(t) )
Exemple de trajectoire de type moveL avec 3 points (dessiner plan x,y puis x(t) et y(t) )
  Lissage avec interpolation cartésienne=>coplanaire
  Lissage avec interpolation articulaire=>rapide