Initiation à la Robotique

ME 5.1a

Licence Professionnelle Automatisme et Robotique

Session 2025 - Amiens

Fabio MORBIDI

Laboratoire MIS
Équipe Perception Robotique
Université de Picardie Jules Verne

E-mail : fabio.morbidi@u-picardie.fr

Organisation du cours

n°	Date	matin/a.m.	CM	TD	Contrôle	Lieu
1	Ven. 4 oct. 2024	a.m.	CM1			Promeo, salle A120
2	Mer. 16 oct. 2024	a.m.	CM2	TD1		Dépt. EEA, TP204
3	Jeu. 17 oct. 2024	matin	CM3	TD2		Dépt. EEA, TP204
4	Jeu. 5 déc. 2024	a.m.	CM4	TD3		Dépt. EEA, TP204
5	Ven. 6 déc. 2024	a.m.			**DS**	Dépt. EEA, TP204
6	Ven. 10 jan. 2025	a.m.			**TP1**	Dépt. EEA 
7	Jeu. 23 jan. 2025	matin			**TP2**	Dépt. EEA 
8	Ven. 24 jan. 2025	matin			**TP3**	Dépt. EEA 

Matin: 8h30-12h15, pause 10h30-10h45 – **Après-midi**: 13h15-17h00, pause 15h30-15h45

Chargé de TD : Daniel Rodrigues da Costa (laboratoire MIS, UPJV)

Plan du cours

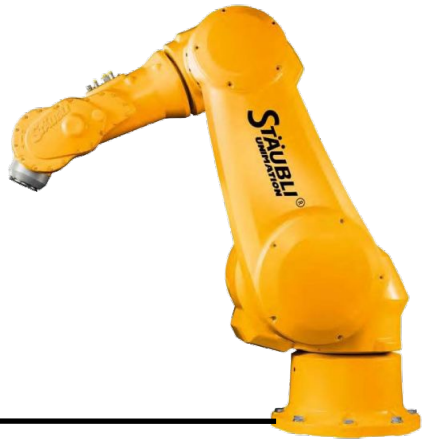
- Introduction
- Constituants et caractéristiques d'un robot
- Gammes de robots et secteurs d'activités
- Les baies de commandes, le boîtier d'apprentissage, les modes et la programmation d'un robot
- Actionneurs et capteurs d'un robot
- Repères et transformations homogènes
- Étude de cas: cellule robotisée de soudage

Baies de commandes: robot Stäubli TX60

Ordinateur hôte

L'utilisateur peut créer son application sur l'ordinateur déporté (Stäubli Robotics Suite)

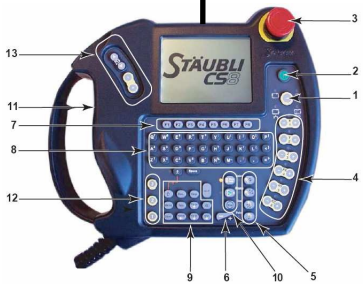
Contrôleur CS8C



S.M.A.

- Nombre de DDL: 6
- Charge nominale: 3.5 kg
- Charge maximale: 9 kg
- Rayon d'action: 670 mm
- Répétabilité: ± 0.02 mm

Dernière séance



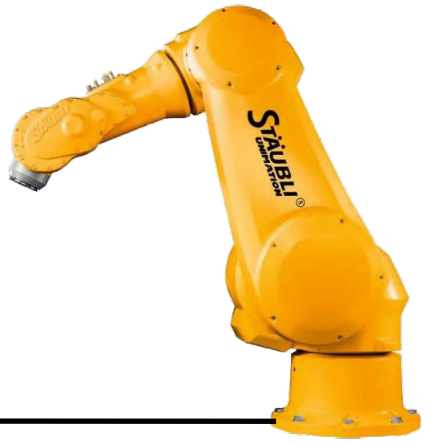
Boîtier d'apprentissage

Baies de commandes: robot Stäubli TX60

Ordinateur hôte

L'utilisateur peut créer son application sur l'ordinateur déporté (Stäubli Robotics Suite)

Contrôleur CS8C



S.M.A.

- Nombre de DDL: 6
- Charge nominale: 3.5 kg
- Charge maximale: 9 kg
- Rayon d'action: 670 mm
- Répétabilité: ± 0.02 mm

Boîtier d'apprentissage

Boîtier d'apprentissage

- Le *boîtier d'apprentissage* (ou *teach pendant, pupitre de commande*) est un terminal portable pour la commande d'un robot qui offre un moyen pratique pour :
  - Faire le déplacement du 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

Boîtier d'apprentissage:
recto (à droite) et verso (à gauche)

- Lorsque le boîtier est activé, il faut appuyer sur l'interrupteur de *corps mort* pour autoriser les mouvements. En le relâchant, le robot s'arrête immédiatement

- C'est également le cas lors d'un appui sur le *bouton rouge* d'arrêt d'urgence

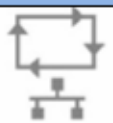
Boîtier d'apprentissage

Fonctionnalités :

1. Mode button
2. Arm power on/off
3. Emergency stop
4. Movement keys
5. Movement mode (*joint*, *frame*, etc.)
6. Speed adjust
7. Function keys
8. Keyboard
9. Navigation keys
10. Application control (*run*, *pause*, etc.)
11. Enable button (*dead man*)
12. Digital I/O buttons
13. Jog keys

Boîtier d'apprentissage

1. Sélection du mode

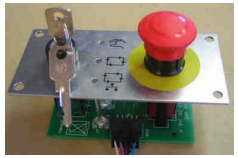
Mode Automatique (Production)		Mode Manuel	
Mode local	**Mode déporté**	**Mode lent**	**Mode test**
			
Le robot exécute une application stockée dans le contrôleur	Le robot est commandé par un programme déporté (sur un PC)	Let robot est contrôlé par l'opérateur qui a le boîtier à la main	L'opérateur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

2. Bouton de power on/off

Cette touche permet d'activer/couper l'alimentation électrique du robot 

3. Bouton d'arrêt d'urgence

L'arrêt d'urgence doit être poussé afin de bloquer le robot, peu importe la tâche qu'il accomplit ou la configuration réelle

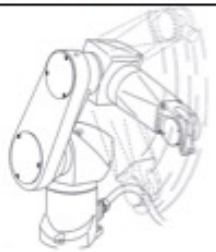
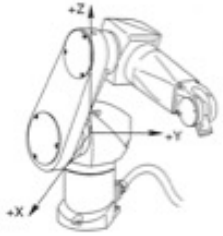
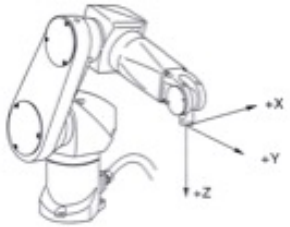


4. Boutons de mouvement

Ces touches permettent à l'utilisateur de déplacer le robot en fonction du mode de déplacement (voir le point 5. ci-dessous)

Boîtier d'apprentissage

5. Sélection du mode 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)
			

6. Contrôle de vitesse

Pour contrôler la vitesse du robot manuellement (en pourcentage) 

Boîtier d'apprentissage

7. Le menu contextuel
Utilisé pour valider l'information donnée sur l'écran juste en haut des boutons (F1, F2,..., F8)

8. Clavier alphanumérique
Utile pour entrer des données de l'application

9. Boutons d'interface et de navigation
Une interface typique avec les boutons de haut, bas, droite, gauche ainsi que les boutons sélection de menu, Esc, Aide

10. Les boutons de l'application
Ils sont les boutons qui permettent d'exécuter une *application programmée*

- **Move/Hold** : bouton de lecture/pause
- **Run** : bouton pour charger l'application à exécuter
- **Stop** : bouton pour mettre fin à l'application courante

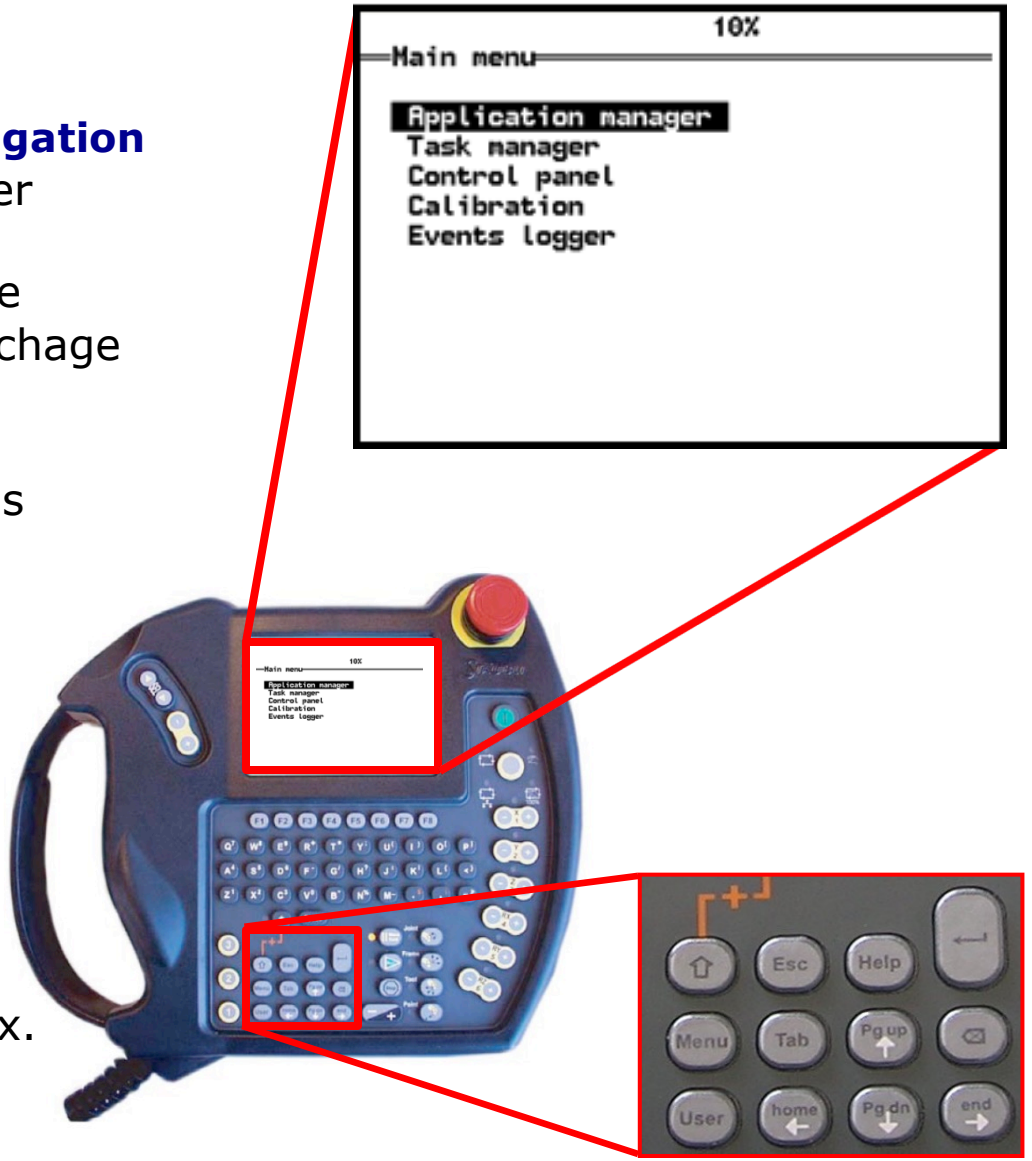
Menu du boîtier

Touches fléchées pour la navigation

- *Haut*/*Bas* pour sélectionner les éléments
- *Droite* pour entrer/éténdre
- *Gauche* pour réduire l'affichage

- **Application manager**
  - Pour charger, enregistrer et modifier les applications
- **Task manager**
  - Pour le débogage
- **Control panel**
  - Visualisation de l'état et paramètres du système, IO (entrée/sortie)
- **Calibration**
  - Entretien du robot
- **Events Logger**

  Il permet d'acceder, par ex. aux messages d'erreur

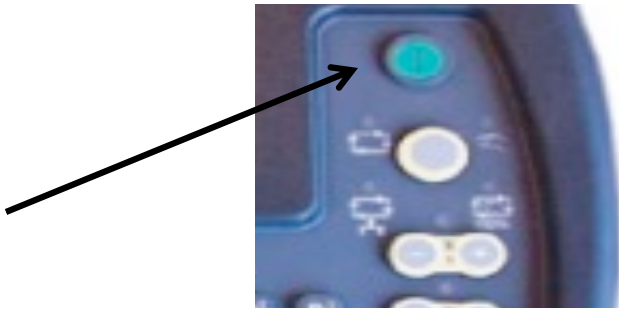


Source: "*VAL3 basics*", S. van Delden, USC Upstate, 2014

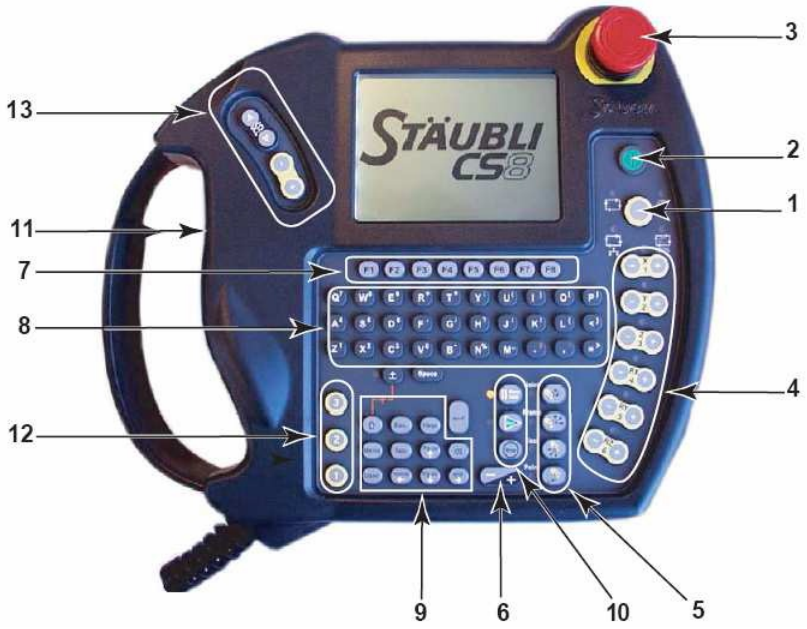
9. Touches

Mode manuel

- Mettre sous tension le bras
  - Touche corps mort
  - Bouton d'alimentation du bras
- Vitesse
  - Limitée à 250 mm/s



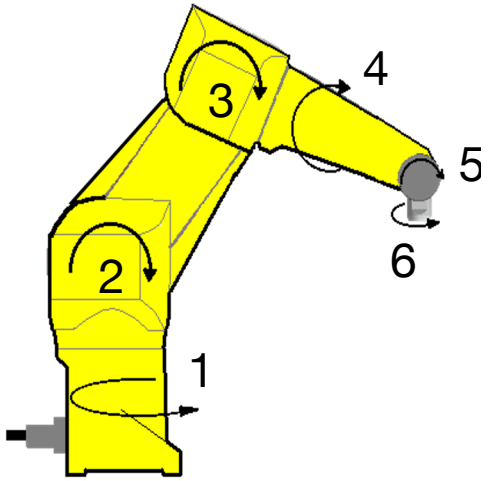
Remarque : en mode manuel le robot peut être « déplacé à la main »



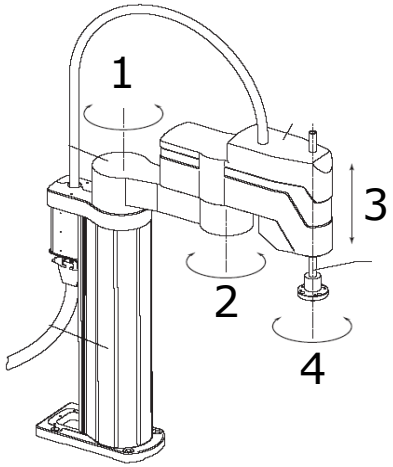
Mode manuel: « joint »

- Déplace chaque articulation séparément
- Simple et facile à comprendre
- *Grands mouvements rapides* du robot dans la cellule

- 6 axes (robot TX60)
  - Un angle pour chaque axe
- 4 axes (robot SCARA)
  - Un angle pour les articulations 1, 2 et 4
  - Une distance pour l'articulation 3

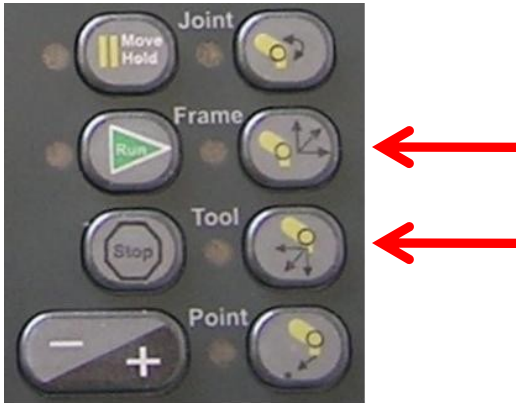


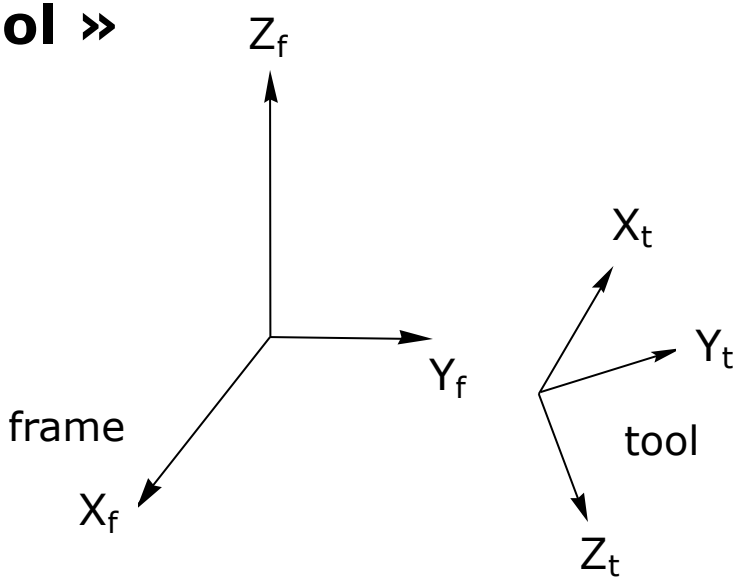
6 axes



4 axes

Mode manuel: « frame » et « tool »





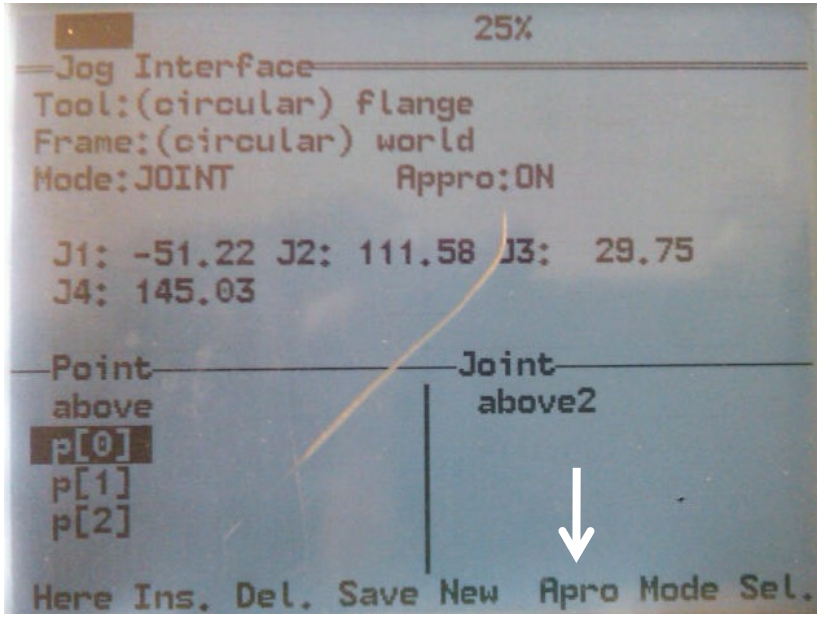
- « frame » et « tool »: mouvement manuel basé sur un *repère cartesien* (système de coordonnées 3D)
- Mouvement coordonné de toutes les articulations
- Permet de realiser trajectoires linéaires et rotations autour des axes du repère
- Efficace pour des *mouvements réduites* et *précis*
- 6 quantités à définir:
  - X, Y, Z : **partie translationnelle**
  - R_X, R_Y, R_Z : **partie rotationnelle**

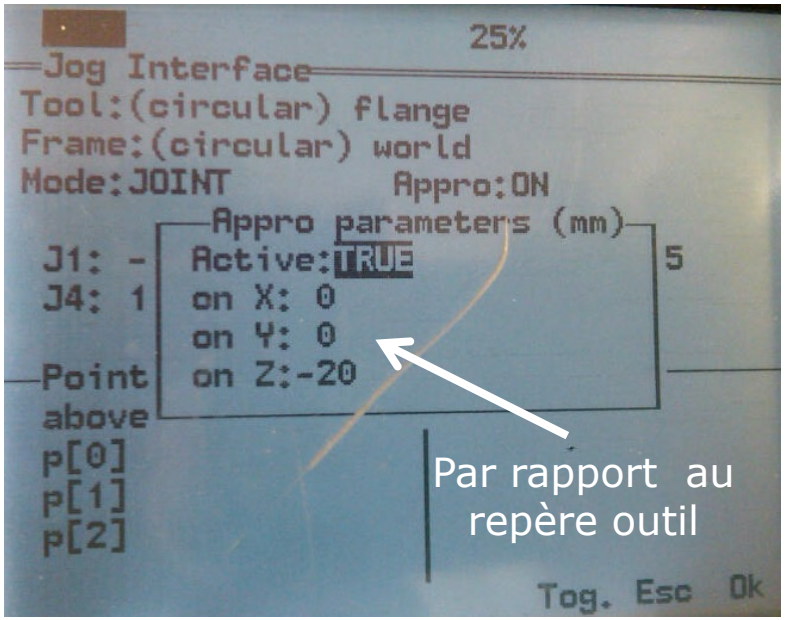
Mode manuel: « point »

- Il peut être utilisé pour faire un déplacement jusqu'à (ou par rapport à) un point qu'a été appris auparavant
- a) Déplacer le curseur vers le point appris et b) appuyer sur **Move/Hold**
- Il peut être aussi utilisé pour approcher les points (par rapport au repère outil)

II. Appuyer et maintenir enfoncé

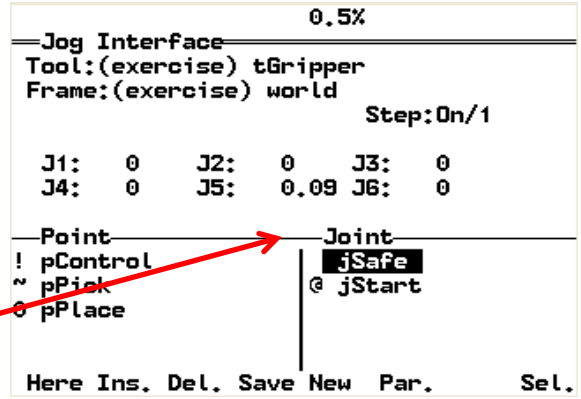
I. Appuyer

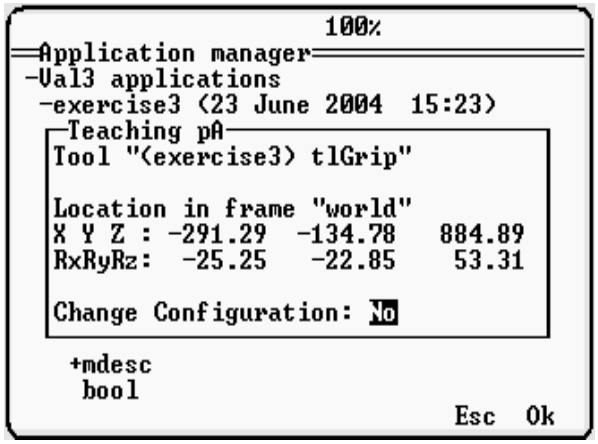


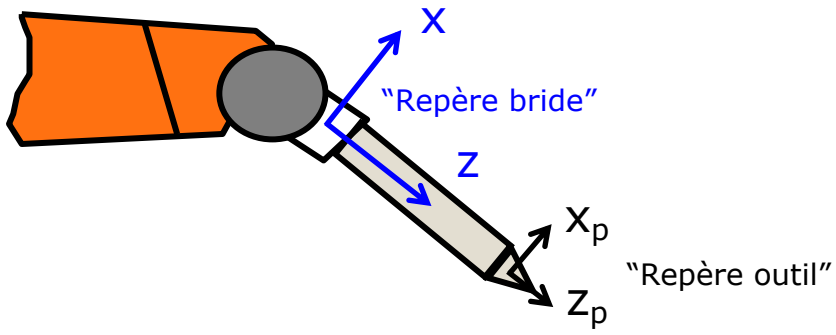


Apprentissage des emplacements

- Appuyer sur **New** pour créer une nouvelle variable « location »
  - Location **point**
    - X-Y-Z et Yaw-Pitch-Roll sont stockés
  - Location **joint**
    - Angles des articulations sont stockés
- Après avoir déclaré la nouvelle variable, pour la mettre en place appuyer sur **Here**
- Sélectionner l'outil
  - **N.B.** : Il faut définir une transformation appropriée entre le repère de la bride (flange) et le repère outil du robot
  - **N.B.** : Yaw-Pitch-Roll en VAL3 sont des rotations autour des axes X-Y-Z, respectivement







Yaw: lacet

Pitch: tangage

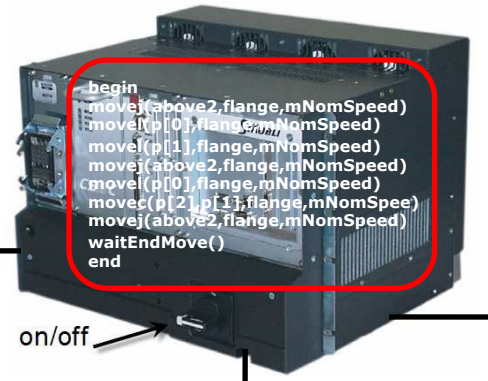
Roll: roulis

Baies de commandes: robot Stäubli TX60

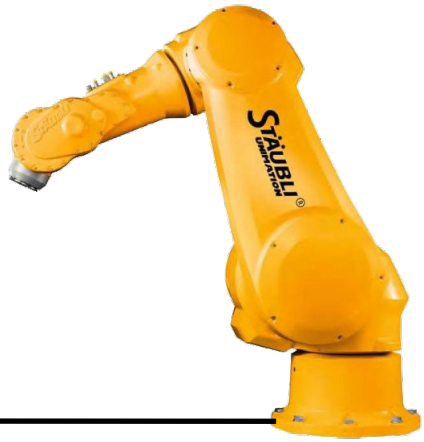
Ordinateur hôte

L'utilisateur peut créer son application sur l'ordinateur déporté (Stäubli Robotics Suite)

Contrôleur CS8C



Boîtier d'apprentissage



S.M.A.

- Nombre de DDL: 6
- Charge nominale: 3.5 kg
- Charge maximale: 9 kg
- Rayon d'action: 670 mm
- Répétabilité: ± 0.02 mm

Création des applications

- Les applications/projets contiennent des *données* et des *instructions* utilisées par le robot pour effectuer une tâche assignée
  - Un dossier avec fichiers **xml** :
    - Un *fichier projet* (PJX)
    - Un *fichier de données* (DTX)
    - Plusieurs *fichiers de programme/fonction* (PGX)
      - Diverses fonctions peuvent être définies sous le même projet
  - Les applications/projets peuvent être :
    - Stockées sur un disque dur
    - Chargées dans la mémoire pour être exécutées

Exemple de fichier programme/fonction PGX

```
<?xml version="1.0" encoding="utf-8" ?>
<programList xmlns="ProgramNameSpace”>
 <program name="demo" public="false”>
  <description/>
  <paramSection/>
  <localSection/>
  <source>
   <code>
        begin
            movej(p[3],flange,mNomSpeed)
            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()
        end
   </code>
  </source>
  </program>
  </programList>
```

Variables VAL3

- Types simples
  - **num** (valeurs numériques : nombres entiers ou à virgule flottante)
  - **bool** (valeurs booléennes : true/false)
  - **string** (chaînes de caractères : caractères Unicode)
  - **dio** (entrées-sorties digitales)
  - **aio** (entrées-sorties analogiques)
  - **sio** (entrées-sorties sur liaison série et socket TCP/IP)
- Types structurés (*structures*)
  - **trsf** (pour les transformations géometriques cartésiennes)
  - **joint** (pour les positions articulaires du robot)
  - **point** (pour les positions cartésiennes de l'outil)
  - **frame** (pour les repères géométriques cartésiens)
  - **config** (pour les configurations du robot)
  - **tool** (pour les outils montés sur le robot)
  - **mdesc** (pour les paramètres de déplacement du robot)

Pour plus d'infos, voir le "*Manuel de Référence VAL3*", Stäubli, ver. 7, 2010 (ch. 2)

Variables VAL3

- **Exemple 1**

  Dans l'exemple suivant, *bBool* est une variable booléenne, *nPi* une variable numérique et *sChaîne* une variable de chaîne

```
bBool = true
nPi = 3.141592653
sChaîne = "ceci est une chaîne"
```

- **Exemple 2**
  - La valeur d'une donnée de *type structuré* est définie par la séquence des valeurs de champ entre les accolades, séparées par des virgules. Les valeurs des champs vides sont remplacées par 0
  - Un tableau ou une collection d'un type structuré doit être initialisé élément par élément
  - Par exemple, le type *point* est fait d'un *trsf* et d'un type *config*. Une variable peut être initialisée de la manière suivante :

```
pPosition = {{100, -50, 200, 0, 0, 0}, {sfree, efree, wfree}}
```

  où sfree "configuration de l'épaule libre", efree "configuration du coude libre" et wfree "configuration du poignet libre" du robot

Opérateurs

- *Mathématiques*:
  - = affectation
  - + somme, concaténation
  - – soustraction
  - * multiplication
  - / division
  - % module
  - ! négation

- *Comparaisons*:
  - == égalité
  - != inégalité
  - > supérieur à
  - >= supérieur ou égal à
  - < inférieur à
  - <= inférieur ou égal à

Instructions

Quelques exemples:

- Abs
- Min
- Max
- Limit (renvoie une valeur bornée)
- Power (renvoie la puissance d'un nombre)

- Left
- Right
- Mid
- Chr
- Asc

} Opérations sur les caractères

- Size (renvoie le nombre de valeurs accessibles avec la variable)

Conteneurs

- Le *conteneur* de données définit la manière dont les valeurs sont stockées:
  - Un conteneur **élément** se compose d'une valeur unique: booléenne (true/false), numérique ou chaîne (string)
  - Un conteneur **tableau** se compose d'un ensemble de valuers identifiées par 1, 2 ou 3 indices entiers
    - L'*indice initial* dans les tableaux est toujours 0
    - L'indice utilisé pour accéder à une valeur dans un tableau est toujours compris entre 0 et la taille de la dimension moins 1
    - Une valeur dans un tableau est accessible par ses indices numériques entre crochets après le nom de la variable : par ex. `n1dArray[nIndex]`, `n2dArray[nIndex1, nIndex2]`, `n3dArray[nIndex1, nIndex2, nIndex3]`
  - Un conteneur **collection** se compose d'un ensemble de valuers identifiées par *une clé* (entre crochets), représentée par une chaîne de caractères

Conteneurs

Remarque:

- Un conteneur tableau à 1 dimension avec une seule valeur (indice 0) est considéré comme un *conteneur élément*

- Il est possible d'accéder à sa valeur sans crochets : par ex. `n1dArray` est équivalent à `n1dArray[0]`

- **Exemple**

Initialisation de variables de type simple avec différents conteneurs (élément, tableau et collection, respectivement) :

```
nPi = 3.141592653
sMois[0] = "Janvier"
sNomProduit ["D 243 064 40 A"] = "VAL 3 CdRom"
```

Instructions de contrôle

- **If**

Exemple : conversion d'un jour écrit dans un **string** (sDay) en un **num** (nDay)

Syntaxe

```
if <bool bCondition>
  <instructions>
[elseif <bool bConditionAlternée1>
  <instructions>]
../..
[elseif <bool bConditionAlternéeN>
  <instructions>]
[else
  <instructions>]
endIf
```

```
put("Entrer un jour : ")
get(sDay)
if sDay=="Lundi"
 nDay=1
elseIf sDay=="Mardi"
 nDay=2
elseIf sDay=="Mercredi"
 nDay=3
elseIf sDay=="Jeudi"
 nDay=4
elseIf sDay=="Vendredi"
 nDay=5
else
 // Week-end !
 nDay=0
endIf
```

Instructions de contrôle

- **While**

Syntaxe

```
while <bool bCondition>
  <instructions>
endWhile
```

Exemple

```
// Ce programme simple fait clignoter un signal lumineux tant que le robot est en mouvement
diLampe = false
while (isSettled()==false)
//Inverse la valeur de diLampe : vrai faux
 diLampe = !diLampe
//Attend ½ s
 delay(0.5)
endWhile
diLampe = false
```

Instructions de contrôle

- **Do … until**

Syntaxe

```
do
  <instructions>
until <bool bCondition>
```

Exemple

```
// Ce programme s'exécute en boucle jusqu'à ce qu'on appuie sur la touche Enter
do
// Attend l'appui d'une touche
 nKey = get()
// Teste le code de la touche Enter
until (nKey == 270)
```

Instructions de contrôle

- **For**

Syntaxe

```
for <num nCompteur> = <num nDébut> to <num nFin> [step <num nPas>]
  <instructions>
endFor
```

Exemple

```
// Ce programme fait tourner l'axe 1 de -90° à +90° par pas de -10°
for nPos = 90 to -90 step -10
 jDest.j1 = nPos
 movej(jDest, flange, mNomSpeed)
 waitEndMove()
endFor
```

Programmes/fonctions

- Une application peut être composée de *plusieurs fonctions* qui peuvent communiquer entre eux
- Il y a deux fonctions qui sont toujours créées lorqu'une nouvelle application est générée:
  - **start()** (ou begin())
    - Cette fonction est toujours appelée lorsque le programme est exécuté
  - **stop()** (ou end())
    - Cette fonction est toujours appelée lorsque le programme est terminé
- On peut placer toute sorte d'instruction/code VAL3 entre ces deux fonctions

TaskCreate et Call

- **taskCreate**
  - Engendre un processus distinct
  - Plusieurs processus peuvent être éxecutés ensemble en parallèle
  - L'unité centrale (CPU) alterne entre toutes les tâches reposant sur une priorité (entre 1 et 100)
  - *Arguments*: nom de la tâche, nivéau de priorité et nom du programme
    - taskCreate sNom, nPriorité, programme()

Exemple

```
// Création d'une nouvelle tâche, t1, de priorité 10 pour lire un message
taskCreate "t1", 10, read(sMessage)
// On attend la fin de la tâche t1
wait(taskStatus("t1") == –1)
// Le message est utilisé
sOutput = sMessage
```

TaskCreate et Call

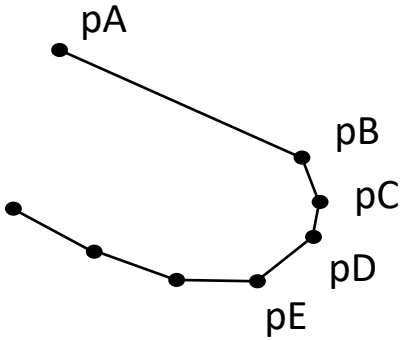
- **call**
  - L'exécution du programme est transferée à un sous-programme
    - call programme([Paramètre1], [Paramètre2])

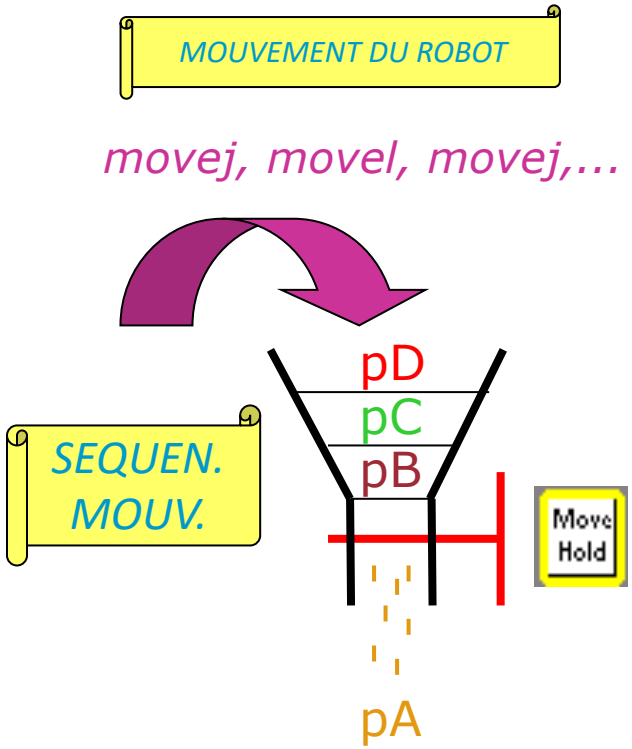
Exemple

```
// Appel les programmes pick() et place() pour i,j entre 1 et 10
for i = 1 to 10
 for j = 1 to 10
   call pick(pPallet1[i,j])
   call place(pPallet2[i,j])
 endFor
endFor
```

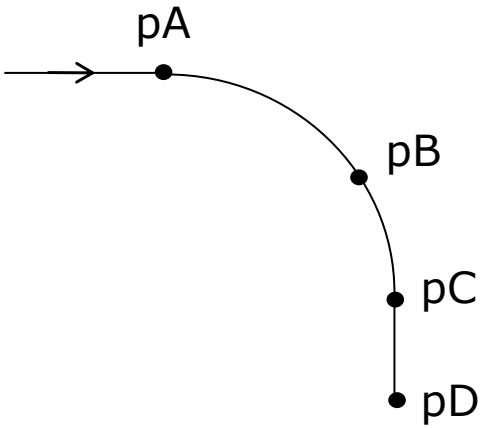
Mouvement

- *Instructions*
  - movej
  - movel
  - movec



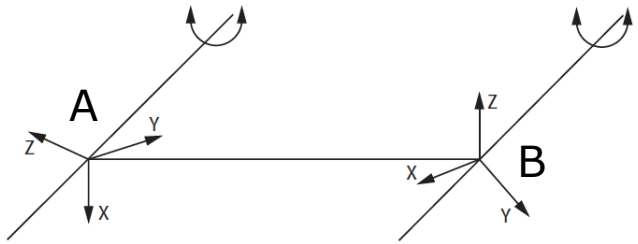


- *Séquençage des mouvements*
  - stopMove
  - restartMove
  - resetMotion
  - waitEndMove



Différents types de mouvement

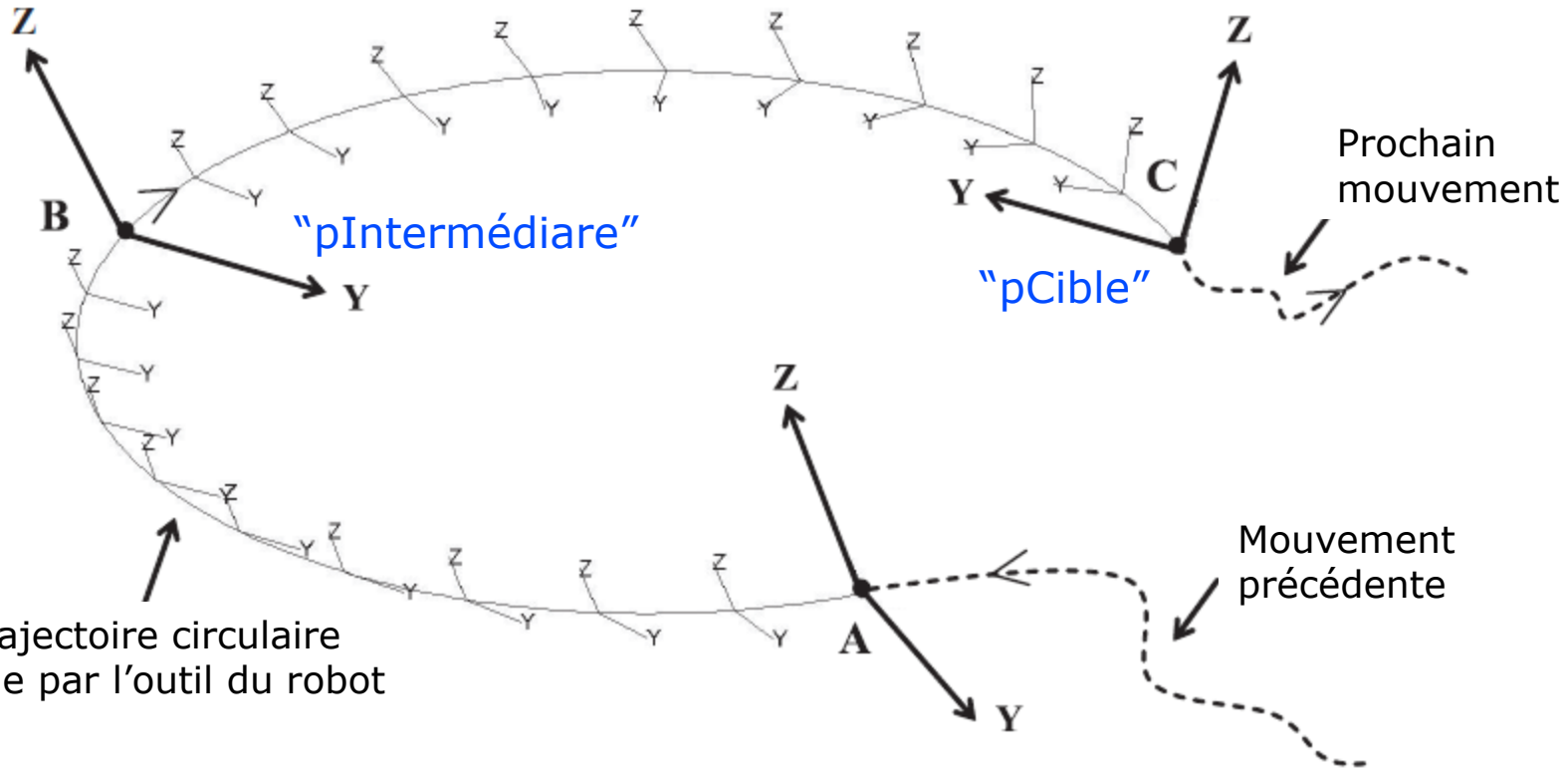
- **Point à point**
  - **movej(point/joint location, tOutil, mDesc)**
    - Mouvement articulaire vers un emplacement désiré spécifié par “point location” ou “joint location”
    - Le 2e paramètre “tOutil” est le repère outil utilisé. Un repère outil est fourni pour chaque commande de mouvement
    - Le type “mDesc” est utilisé pour définir les paramètres de déplacement du robot. “mNomSpeed” est la variable de default (ou *built-in*): elle se réfère à un mouvement normal. La vitesse est spécifiée en %
- **Linéaire**
  - **movel(point location, tOutil, mDesc)**
    - Comme **movej**, mais mouvement en ligne droite. Passage sur les singularités: problématique



- **Circulaire**
  - **movec(pIntermédiaire, pCible, tOutil, mDesc)**
    - Mouvement circulaire qui part de la destination précédente et s’achève au point **pCible**, en passant par le point **pIntermédiare**

Différents types de mouvement

Exemple: mouvement circulaire (movec(...))



Différents types de mouvement

Le type **mdesc** permet de définir les paramètres d'un mouvement (vitesse, accélération, lissage)

Le type **mdesc** est un type structuré dont les champs sont, dans l'ordre :

num accel	Accélération articulaire maximale autorisée, en **%** de l'accélération nominale du robot.
num vel	Vitesse articulaire maximale autorisée, en **%** de la vitesse nominale du robot.
num decel	Décélération articulaire maximale autorisée, en **%** de la décélération nominale du robot.
num tvel	Vitesse maximale autorisée de translation du centre outil, en mm/s ou pouce/s selon l'unité de longueur de l'application.
num rvel	Vitesse maximale autorisée de rotation de l'outil, en degrés par seconde.
blend blend	Mode de lissage : **off** (pas de lissage), **joint** ou **Cartesian** (lissage cartésien).
num leave	Dans les modes de lissage **joint** et **Cartesian**, la distance entre le point cible auquel commence le lissage et le point suivant, en mm ou en pouces, selon l'unité de longueur de l'application.
num reach	Dans les modes de lissage **joint** et **Cartesian**, la distance entre le point cible auquel finit le lissage et le point suivant, en mm ou en pouces, selon l'unité de longueur de l'application.

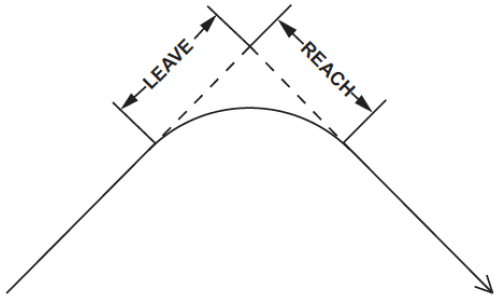
Par défaut, une variable de type **mdesc** est initialisée avec `{100,100,100,9999,9999,joint,50,50}`

Différents types de mouvement

- **Exemple de programme**

```
sOutput = "Mon programme"
mDesc.vel = 80
mDesc.blend = joint
mDesc.leave = 10
mDesc.reach = 10
movej(pA, tOutil, mDesc)
movej(pB, tOutil, mDesc)
waitEndMove()
movej(pC, tOutil, mDesc)
movej(pD, tOutil, mDesc)
```

À quelle distance (en millimètres) du point d'arrivée on quitte la trajectoire nominale (début du lissage, "leave") et à quelle distance on la rejoint (fin du lissage, "reach")

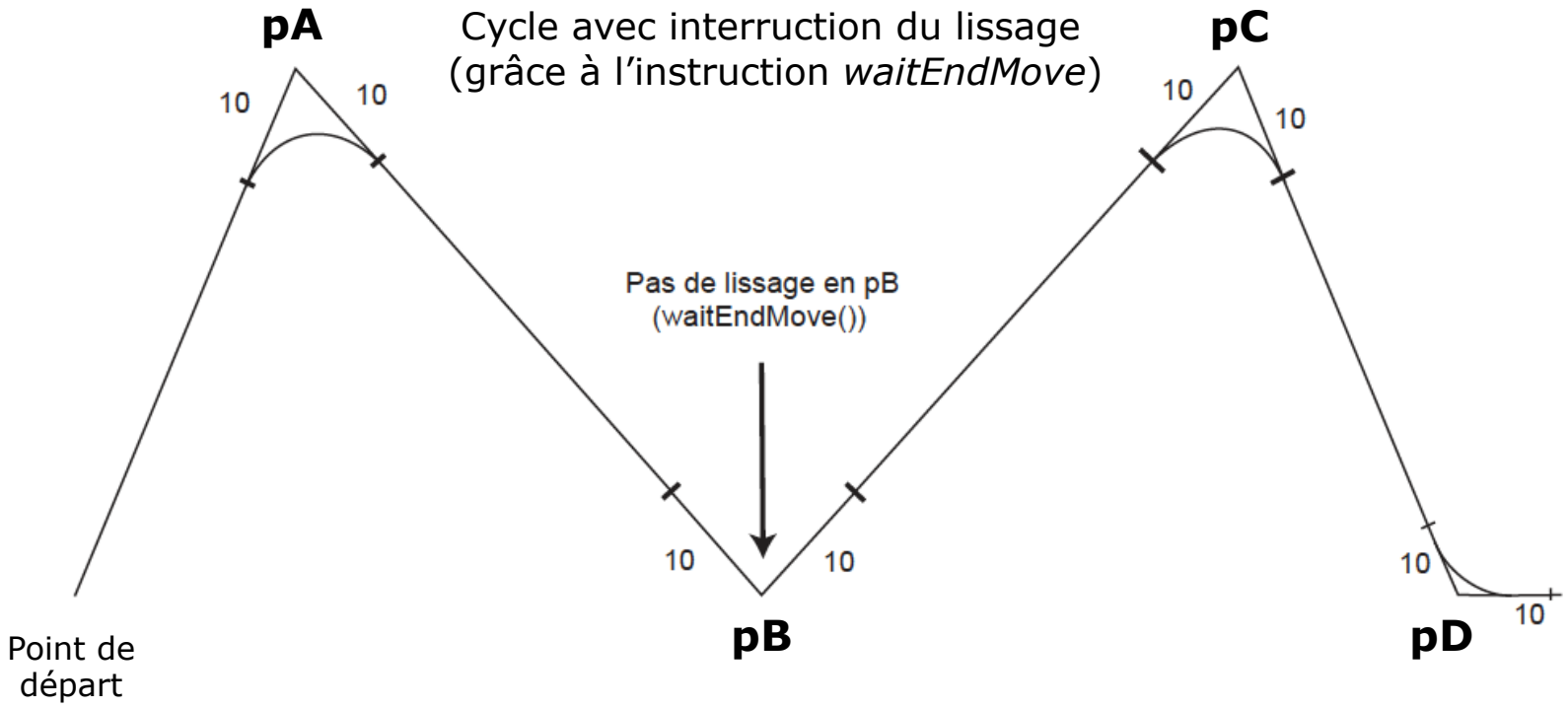


- Les deux premiers "movej" seront exécutés alors que le robot commence son mouvement vers pA
- Puis l'exécution du programme sera bloquée sur l'instruction "waitEndMove()" jusqu'à ce que le robot soit stabilisé au point pB
- Un fois le mouvement du robot stabilisé en pB, le programme reprendra son exécution

Différents types de mouvement

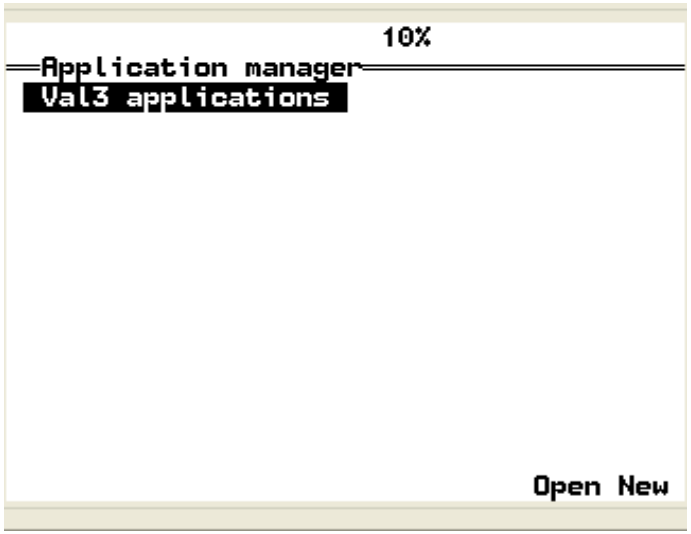
- **Exemple de programme**

  Voici la vraie trajectoire suivie par le robot

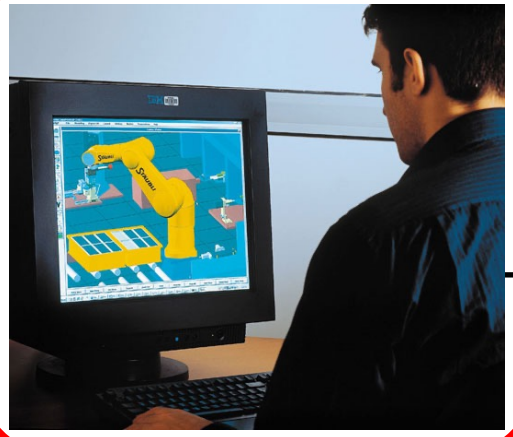


Exécution d'une application

- Défiler pour charger du disque ou pour créer un nouveau programme et le stocker dans le disque dur
- Il faut être en mode **Local** pour pouvoir lancer le programme
  - Appuyer sur le bouton *Mode* pour faire défiler les différents modes
- Appuyer le bouton **Run** et choisir le programme à lancer
- Si le programme implique le mouvement du robot, il faut aussi appuyer **Move/Hold**



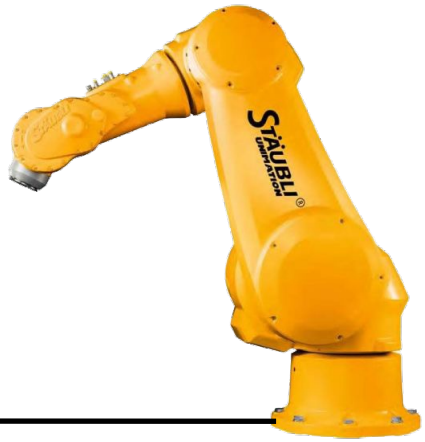
Baies de commandes: robot Stäubli TX60



Ordinateur hôte

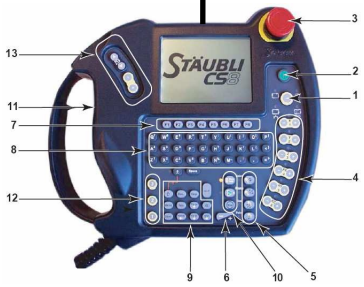
L'utilisateur peut créer son application sur l'ordinateur déporté (Stäubli Robotics Suite)

Contrôleur CS8C



S.M.A.

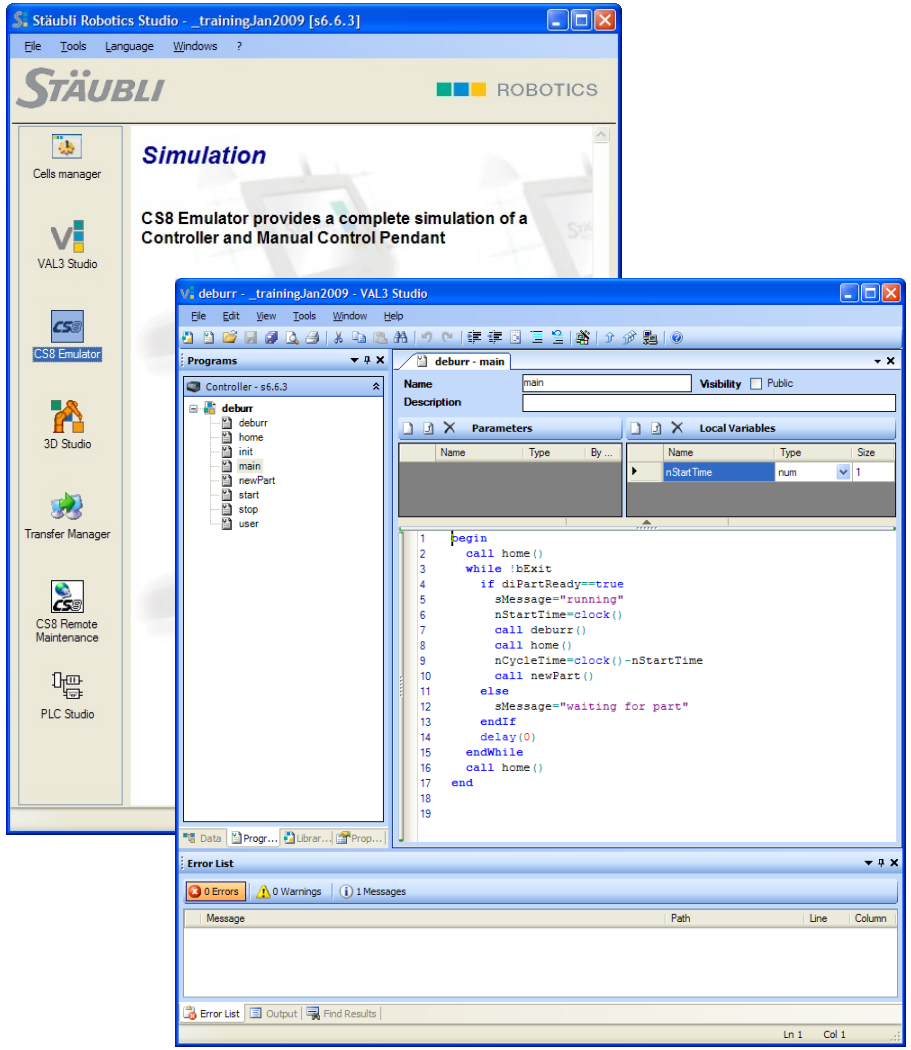
- Nombre de DDL: 6
- Charge nominale: 3.5 kg
- Charge maximale: 9 kg
- Rayon d'action: 670 mm
- Répétabilité: ± 0.02 mm



Boîtier d'apprentissage

Stäubli Robotics Suite (SRS)

- Du code VAL3 peut être aussi développé avec la **Stäubli Robotics Suite**
- La SRS est un IDE (*Integrated Development Environment*) qui est installé sur un ordinateur deporté (hôte)
- La SRS est plus conviviale et facile à utiliser que le boîtier
- Programmation *hors ligne*
  - Modification
  - Test
  - Transfert de programme

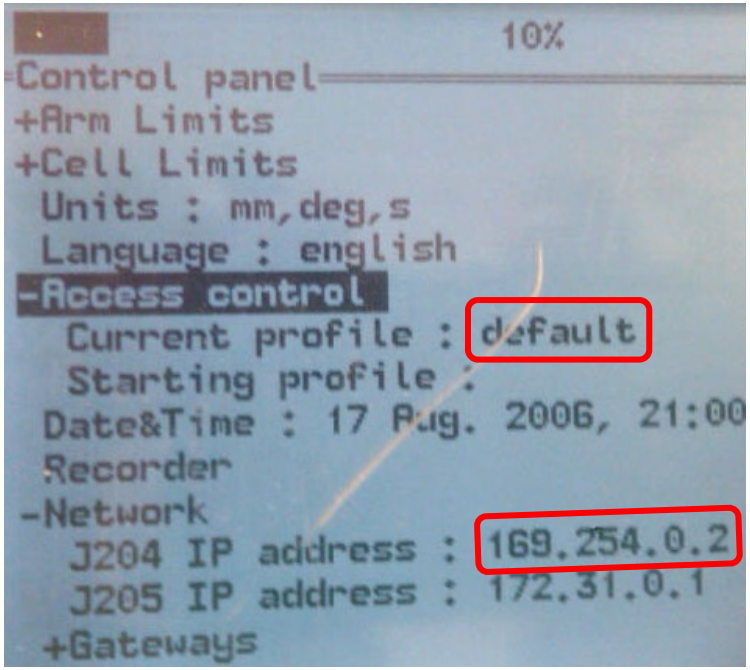


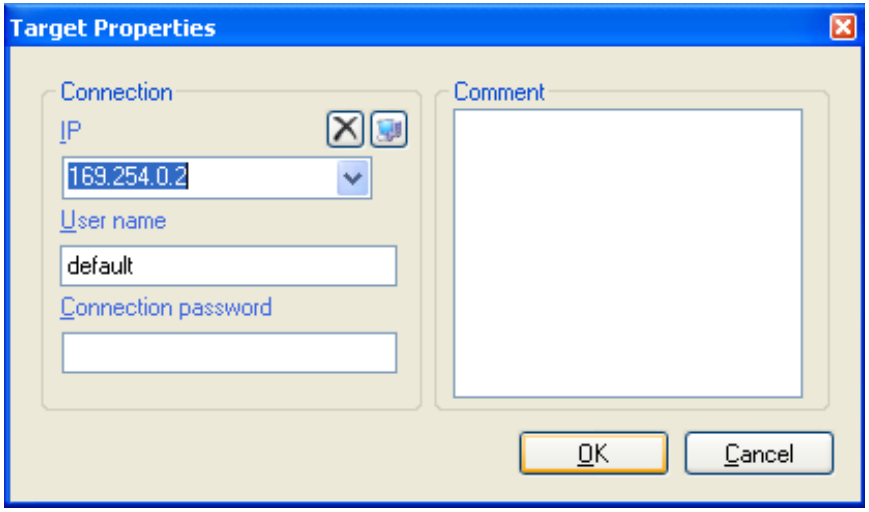
Utilisation de la Stäubli Robotics Suite

- Un projet peut être envoyé par FTP entre le contrôleur CS8C et l'ordinateur sur lequel est installée la SRS (et vice versa)
- La SRS est un environnement de développement flexible qui permet de créer des projets complexes sans difficulté
- Toutefois, il faut encore *apprendre* les points grâce au boîtier d'apprentissage …

Utilisation de la Stäubli Robotics Suite

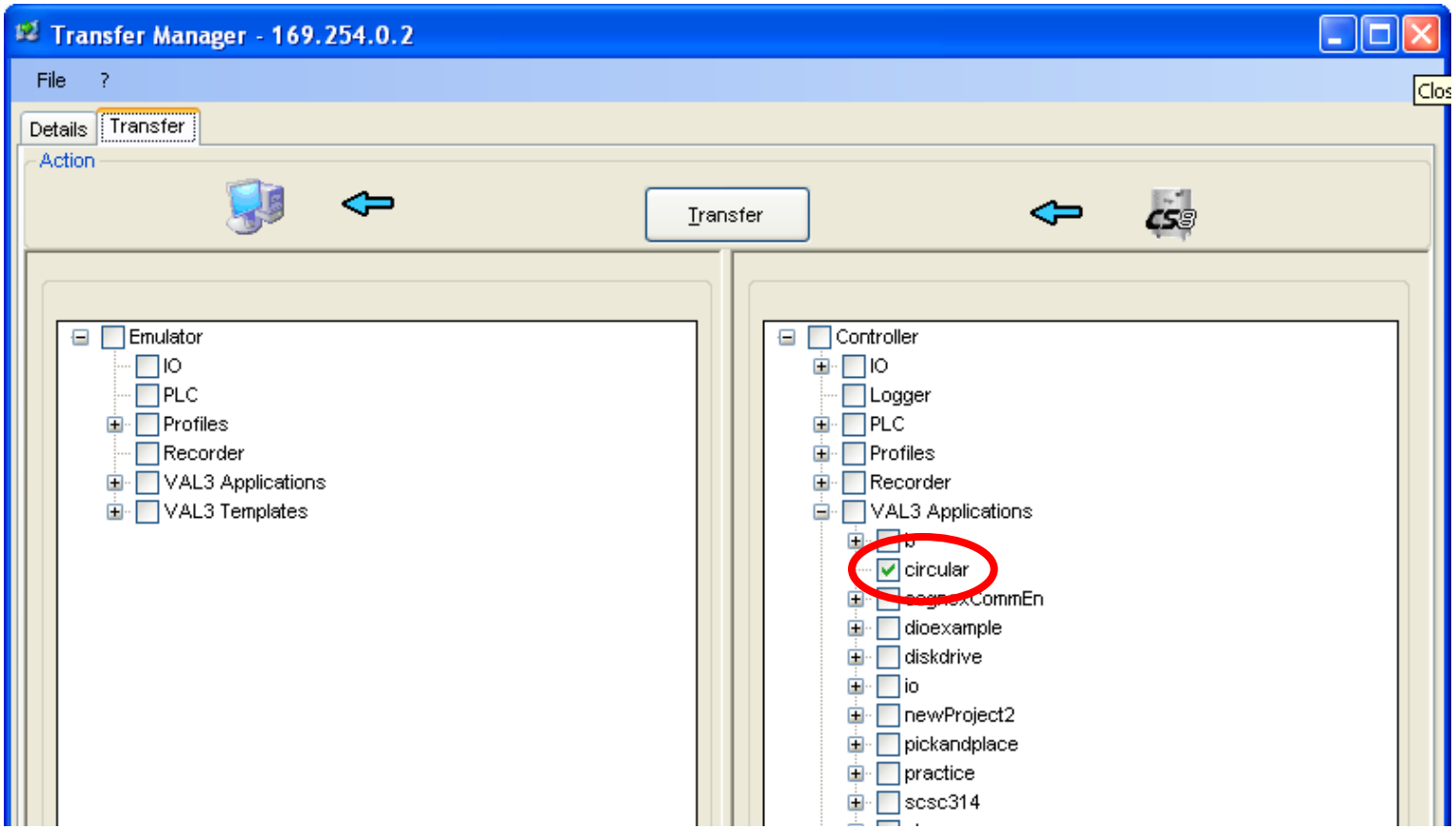
- Pour transférer un projet au contrôleur, il faut établir une connexion FTP
- Il faut connaître l'adresse IP du contrôleur et l'username/password
  - L'adresse IP on peut le trouver dans les menus sur le *Control Panel*
- Entrer les informations dans l'outil FTP de la SRS





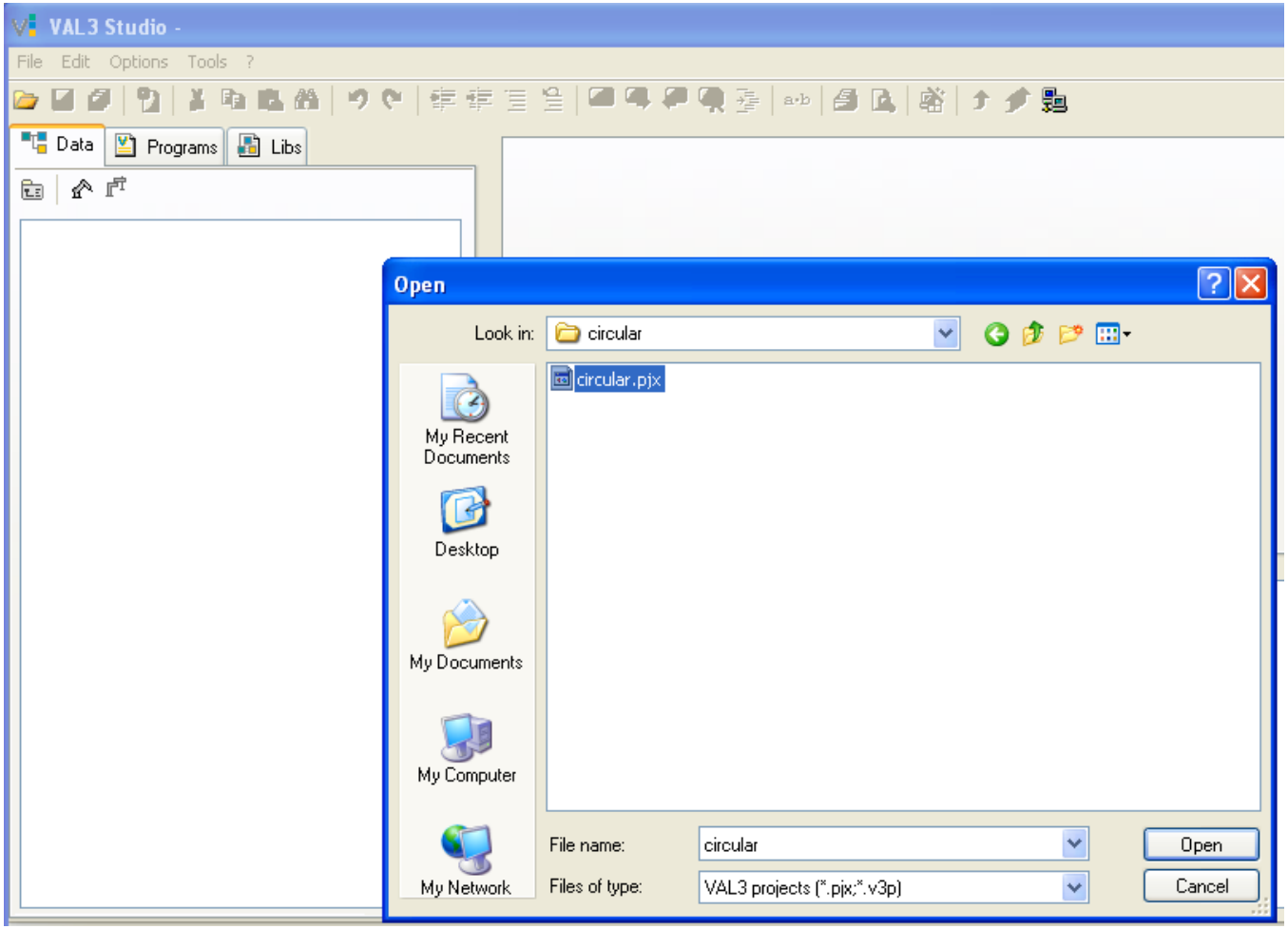
Utilisation de la Stäubli Robotics Suite

- Trouver le projet qu'on souhaite transférer



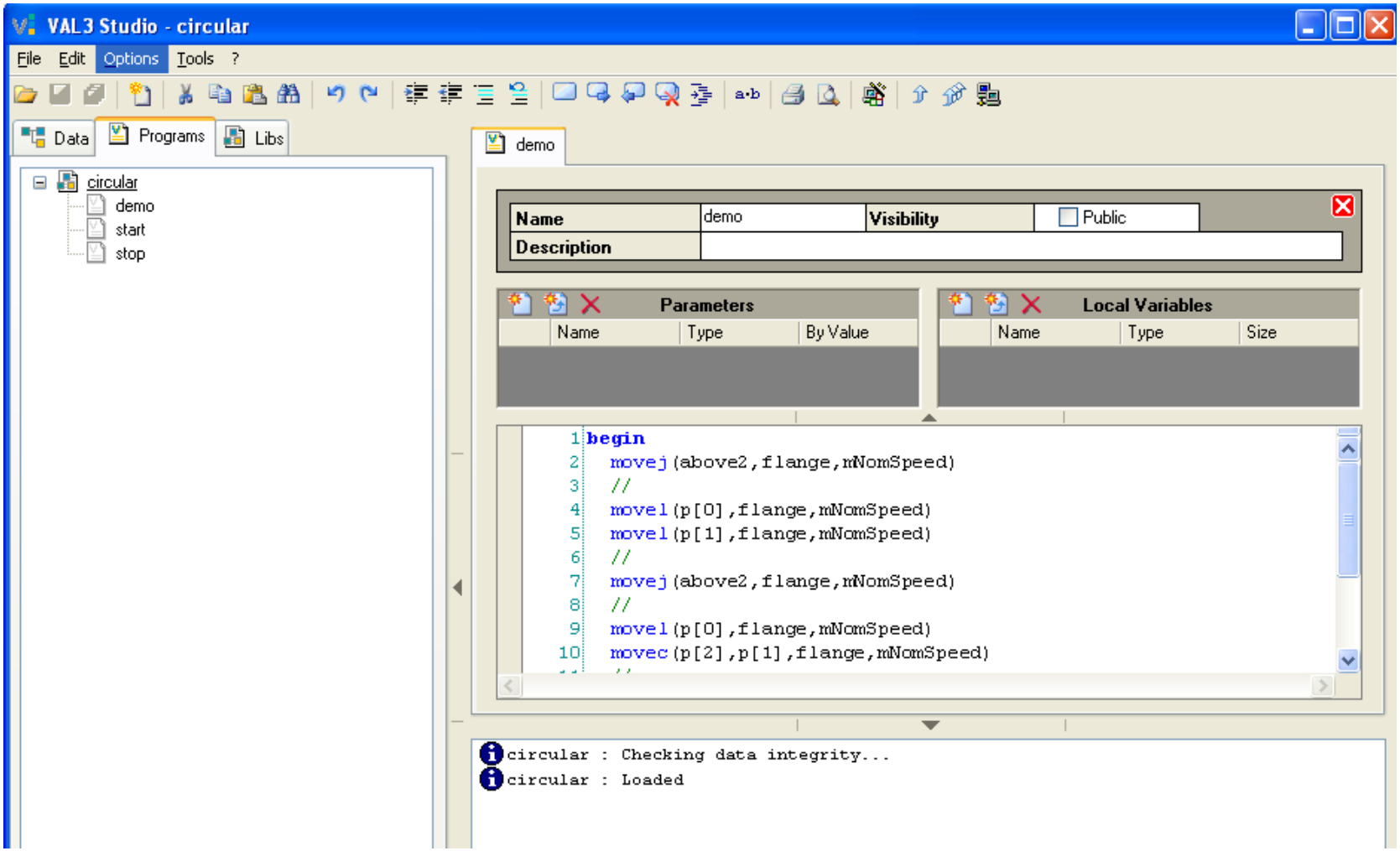
Utilisation de la Stäubli Robotics Suite

- Ouvrir le projet sur la SRS



Utilisation de la Stäubli Robotics Suite

- Modification du code et transfert au contrôleur



Plan du cours

- Introduction
- Constituants et caractéristiques d'un robot
- Gammes de robots et secteurs d'activités
- Les baies de commandes, le boîtier d'apprentissage, les modes et la programmation d'un robot
- Actionneurs et capteurs d'un robot
- Repères et transformations homogènes
- Étude de cas: cellule robotisée de soudage

Effecteur *vs* actionneur

- Un **effecteur** est tout dispositif (contrôlé par un robot) pouvant *affecter l'environnement*
  - Grande variété d'effecteurs: pince, ventouse, main, scalpel, etc.

- Un **actionneur** est le mécanisme qui permet à l'effecteur d'exécuter une action
  - Les actionneurs incluent typiquement les les moteurs électriques et les systèmes hydrauliques ou pneumatiques

Système d'actionnement d'un robot

Éléments du système d'actionnement d'un robot manipulateur :

- *Alimentation*
- *Amplificateur de puissance*
- *Servomoteur*
- *Transmission* (réducteurs)

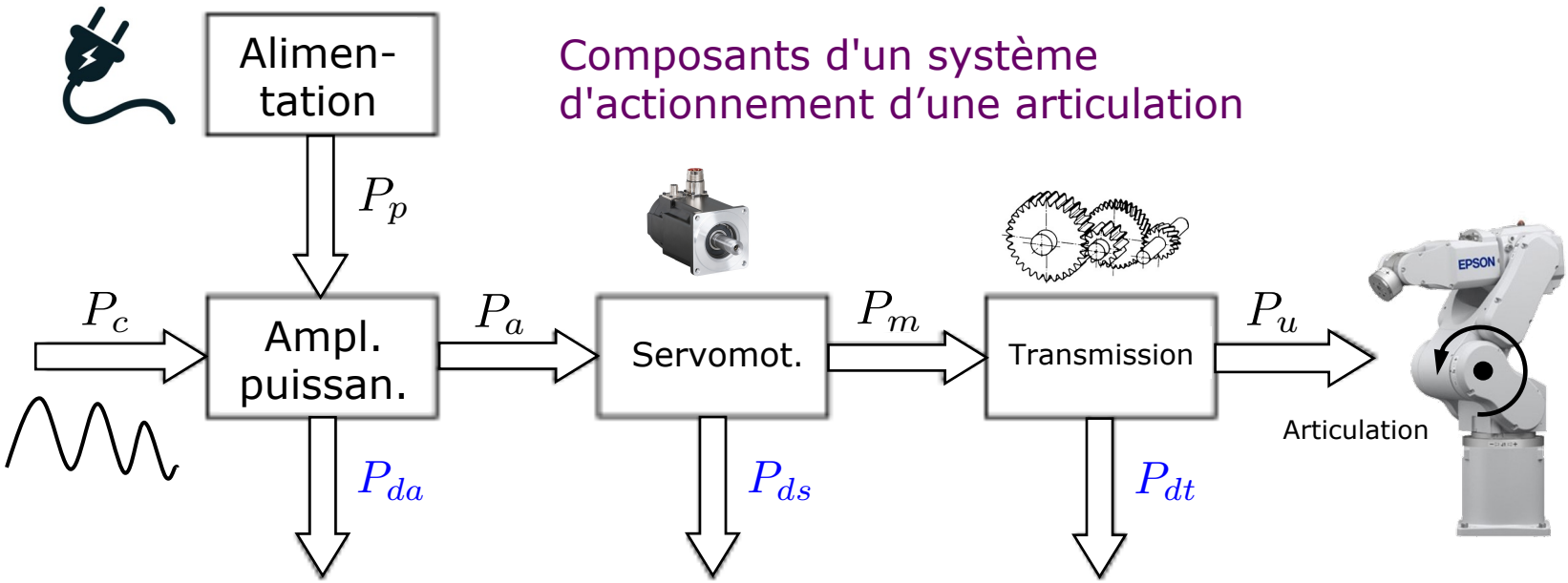
Deux types de servomoteurs sont généralement utilisés:

- *Servomoteurs électriques* pour l'actionnement des articulations des manipulateurs de petite ou moyenne taille (par ex. le Stäubli TX-60)

- *Servomoteurs hydrauliques* pour l'actionnement de manipulators de grande taille

KUKA KR 1000 titan (charge max: 1300 kg)

Actionnement de l'articulation d'un robot



P_p : puissance fournie par la source primaire (de la même nature que P_a)

P_c : puissance (typiquement électrique) associée au signal de la loi de commande

P_a : puissance à fournir au moteur (de type électrique, hydraulique ou pneumatique)

P_m : puissance mécanique générée par le moteur

P_u : puissance mécanique requise par l'articulation pour mettre en place le mouvement

P_{da}, P_{ds}, P_{dt} : *pertes en puissance* par dissipation dans les conversions effectuées par l'amplificateur, le moteur et la transmission (train d'engrenages)

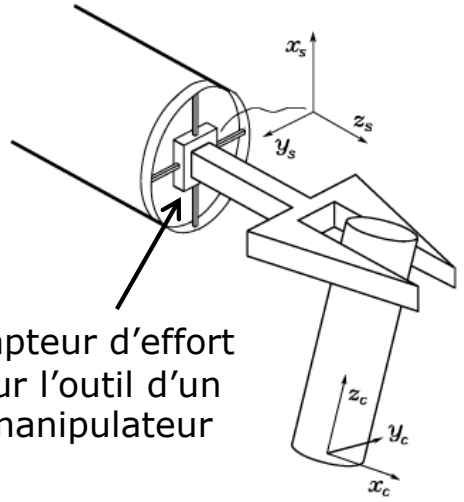
Classification des capteurs: deux axes principaux

Capteurs proprioceptifs: ils permettent de mesurer des quantités qui caractérisent l'*état interne* d'un robot

- Encodeurs et résolveurs pour la mesure de la position d'une articulation
- Tachymètres pour la mesure de vitesse des articulations
- Capteurs pour la mesure des couples au niveau des articulations

Capteurs extéroceptifs: le but de ces capteurs est d'extraire l'information qui caractérise l'interaction du robot avec l'environnement

- Capteurs d'effort pour la mesure de force/couple sur l'outil
- Capteurs de distance/proximité pour la detection d'objects dans l'espace de travail (par ex. capteurs à ultrasons, télémetrès lasers)
- Capteurs de vision pour mesurer les paramètres caractéristiques des objects (par ex. caméras)



Capteur d'effort sur l'outil d'un manipulateur

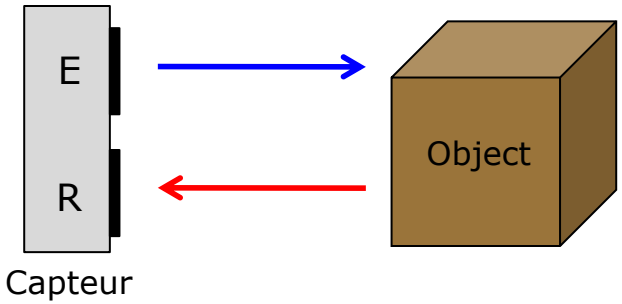
Classification des capteurs: deux axes principaux

Capteurs passifs

- Ils sont des « vrais » observateurs de l'environnement
- Ils capturent les signaux qui sont émis par les autres sources de l'environnement (par ex. caméras, lumière; thermomètres, température; microphones, ondes sonores)

Capteurs actifs

- Ils émettent de l'*énergie*
- L'énergie est réfléchie par l'environnement au capteur. Il existe toujours un élément émetteur-récepteur (par ex. sonars, télémètres laser)



Caractéristiques d'un capteur (1/4)

- *Champ de perception/vue* (ex. 53° horiz. pour une caméra)
- *Portée ou plage de mesure* [« range » en anglais]
  - Valeurs max et min d'un paramètre qui sont mesurables (ex. de 4 à 0.05 mètres pour un télémètre laser)
- *Plage dynamique* [« dynamic range » en anglais]
  - Rapport entre la valeur maximale d'entrée et la valeur mesurable minimale d'entrée
    - Exprimé en décibel «dB» (rapport de « puissances »)

$$L_P = 10 \log_{10}\left(\frac{P}{P_0}\right) \text{ dB}$$ où

P : Puissance maximale

P_0 : Puissance mes. minimale

Exemple:

Mesure de puissance de 1mW à 20W

$$L_P = 10 \log_{10}\left(\frac{20}{0.001}\right) \text{ dB} \simeq 43 \text{ dB}$$

Caractéristiques d'un capteur (2/4)

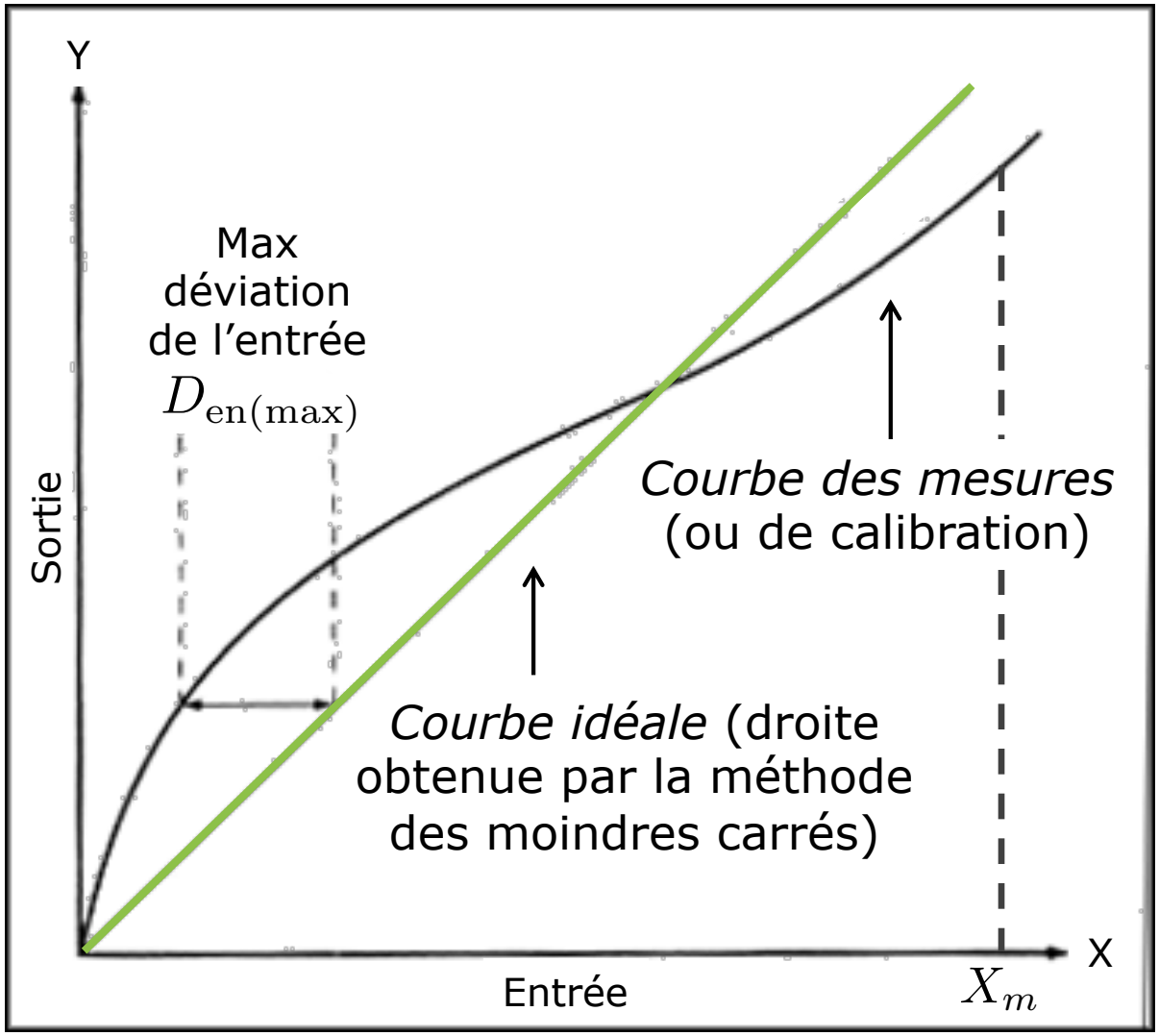
- *Résolution*
  - La plus petite variation ΔX de l'entrée qui peut être détectée en sortie (ex. 0.1° pour un encodeur optique)
- *Fréquence ou largeur de bande*
  - Débit de mesures du capteur
    - Borne supérieure due à la fréquence d'échantillonnage (cf. le théorème de Shannon)
    - Délai (déphasage) possible
- *Linéarité*
  - La linéarité d'un capteur est une expression de la plage d'écart entre la courbe des mesures du capteur et la courbe idéale
  - La linéarité est souvent exprimée en *pourcentage de non linéarité*:

$$\text{Nonlinéarité (\%)} = 100\,\frac{D_{\text{en}}(\max)}{X_m}$$

$D_{\text{en}}(\max)$: max déviation de l'entrée

X_m : valeur fond échelle de l'entrée

Caractéristiques d'un capteur (2/4)

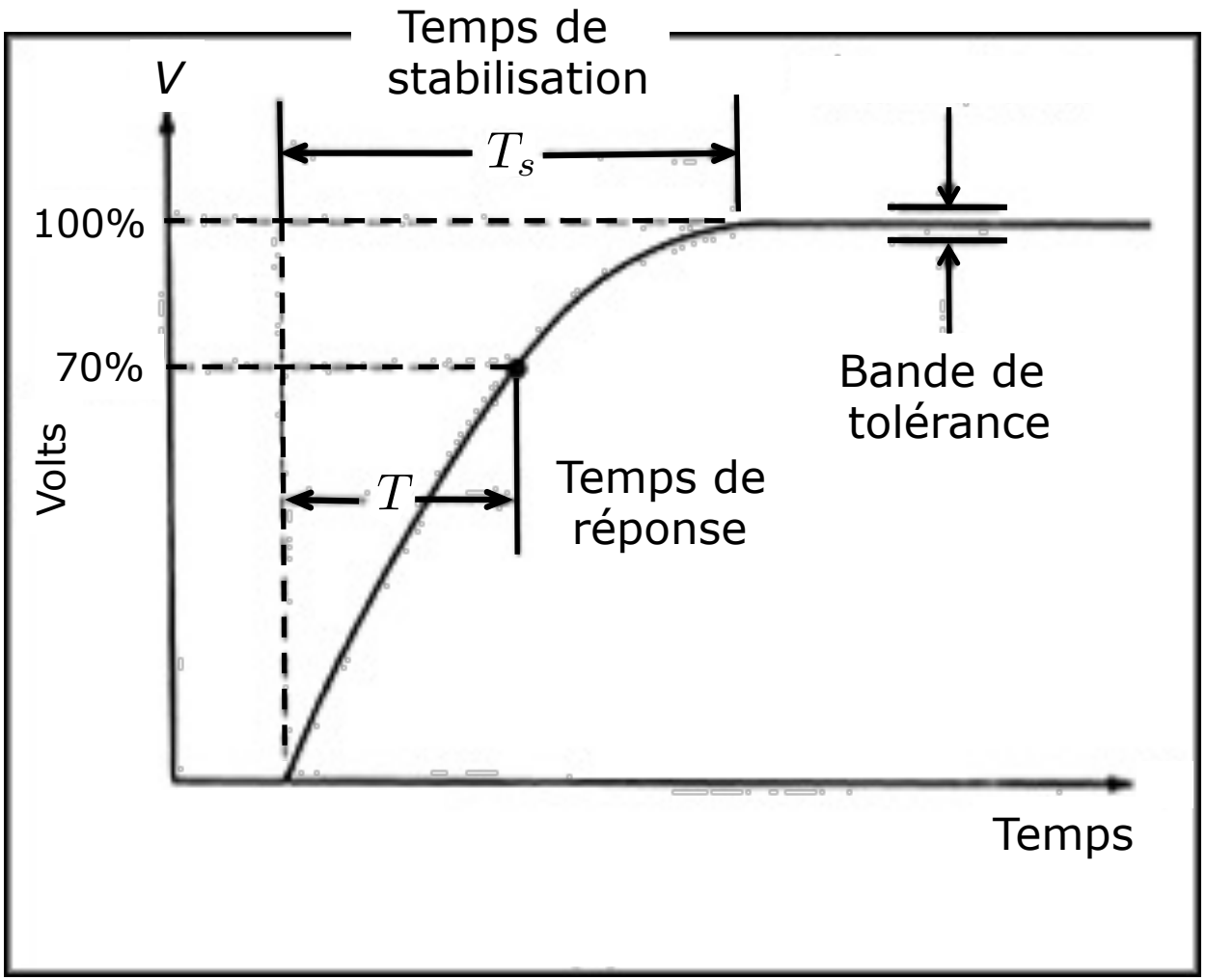


Caractéristiques d'un capteur (2/4)

- *Linéarité dynamique*
  - La linéarité dynamique d'un capteur est une mesure de sa capacité à suivre des *changements rapides* du signal d'entrée
- *Temps de réponse*
  - La sortie d'un capteur ne change pas *immédiatement* lorsque le signal d'entrée varie
  - Au contraire, la sortie met un certain temps pour changer d'état, appelé le *temps de réponse* (T dans la figure suivante)
  - Le temps de réponse peut être défini comme le temps nécessaire pour que la sortie du capteur passe de son état précédent à une valeur finale dans une certaine bande de tolérance

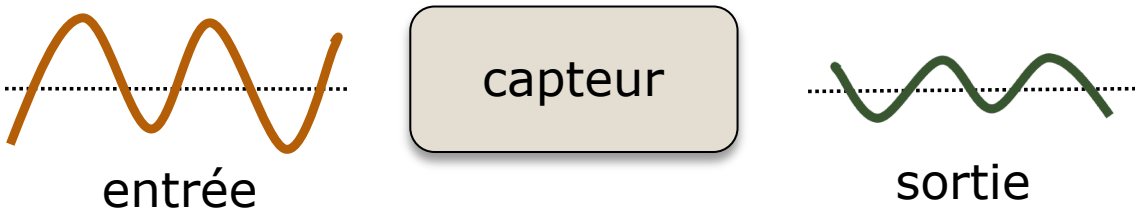
Caractéristiques d'un capteur (3/4)

Exemple:



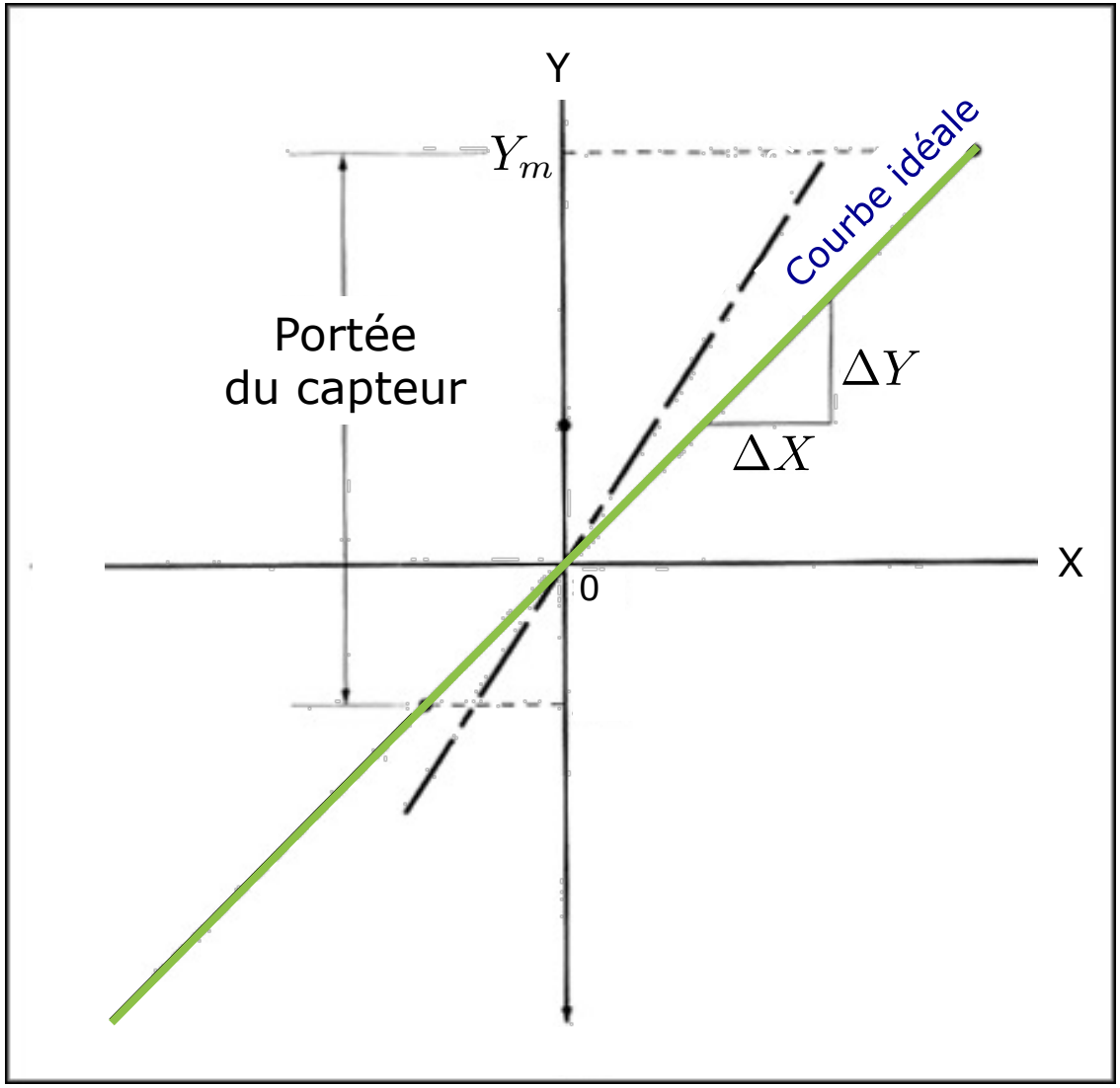
Caractéristiques d'un capteur (3/4)

- *Sensibilité*
  - La sensibilité d'un capteur est définie comme la pente de la courbe caractéristique ou idéale de sortie ($\Delta Y / \Delta X$ dans la diapo suivante)
  - De façon plus générale, elle est la *valeur minimale* du signal d'entrée qui produit un changement détectable dans la sortie



- *Erreur de sensibilité*
  - L'erreur de sensibilité est l'écart par rapport à la pente idéale de la courbe caractéristique (voir la courbe pointillée dans la figure suivante)

Caractéristiques d'un capteur (3/4)



Caractéristiques d'un capteur (4/4)

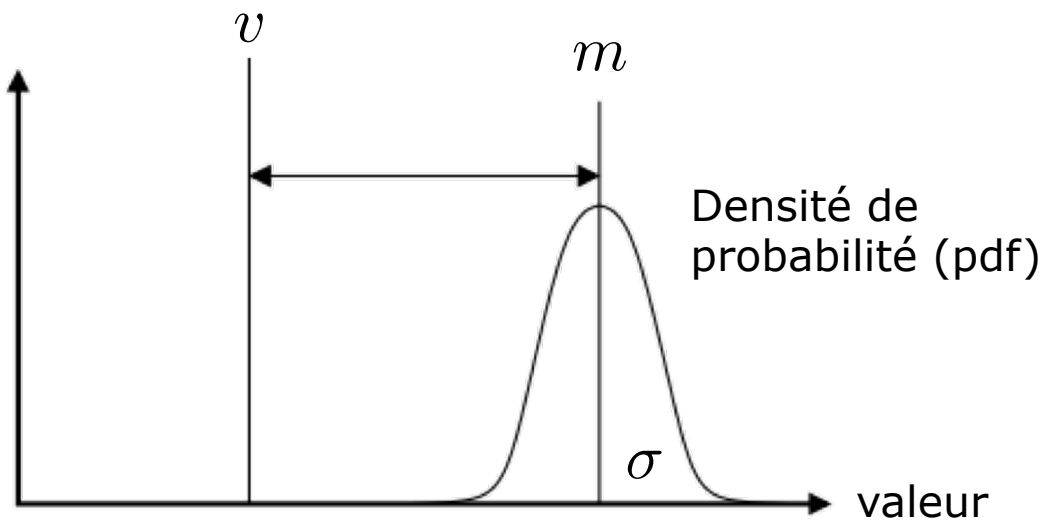
- *Sensibilité croisée*
  - Sensibilité à d'autres influences de l'environnement (ex. les matériaux ferreux pour le compas fluxgate)
  - Influence d'autres capteurs actifs (ex. 2 sonars face-à-face)

- *Exactitude* (opposée à l'erreur) [« accuracy » en anglais]
  - Degré de conformité entre la *mesure* et la *valeur réelle*
  - Elle est exprimée en pourcentage (%) de la valeur réelle

- *Précision*
  - Liée à la reproductibilité des résultats d'un capteur
  - Si exactement la même quantité était mesurée plusieurs fois, un capteur *idéal* fournirait *exactement* la même valeur à chaque fois

Attention: *exactitude* n'est pas synonyme de *précision* !

Caractéristiques d'un capteur (4/4)



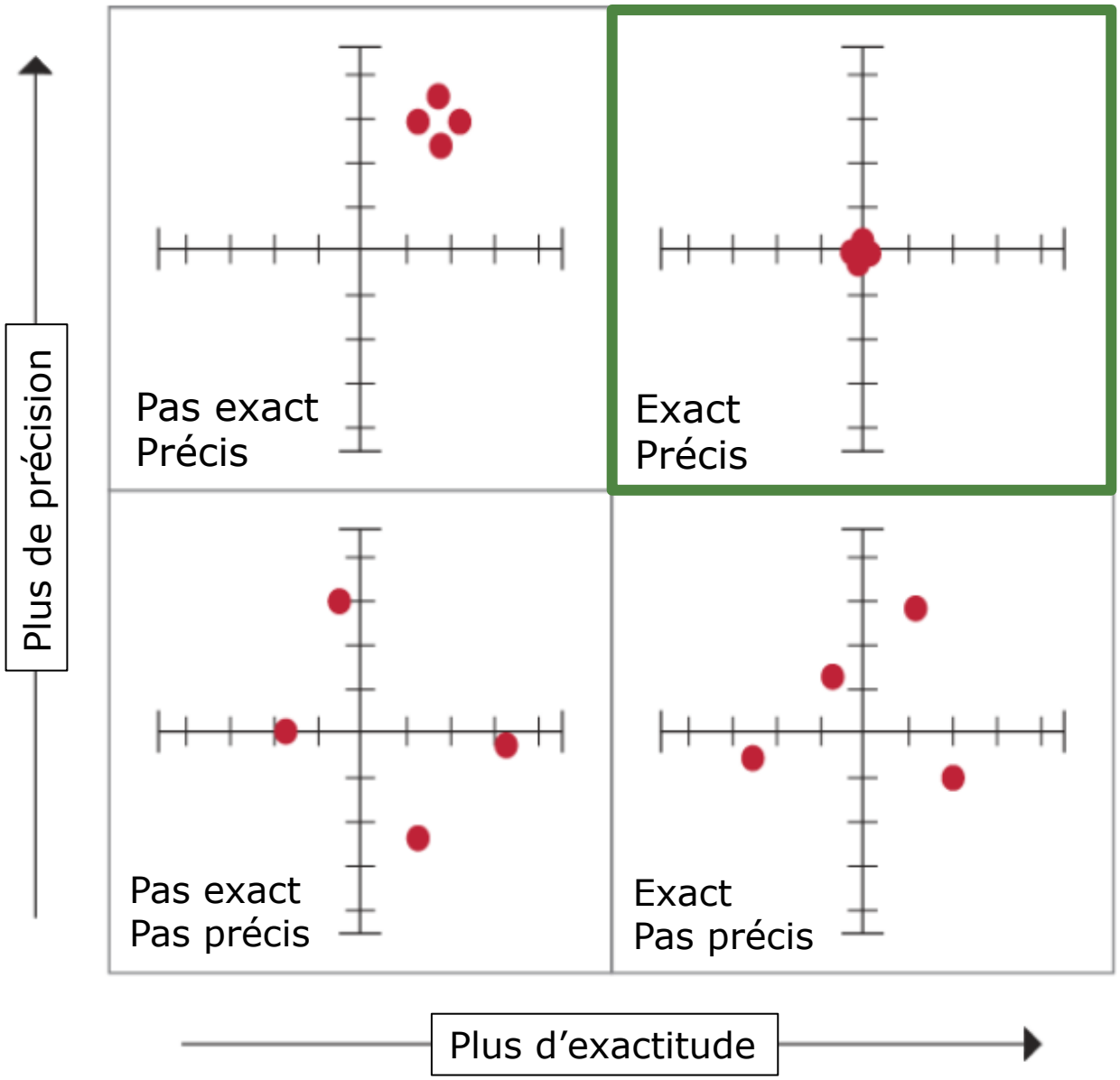
- Exactitude (%) $= 100\left(1 - \frac{|m - v|}{v}\right)$
- Précision $= \frac{Y_m}{\sigma}$

v : valeur mesurée

m : valeur réelle (référence)

σ : écart type de la mesure

Y_m : valeur fond échelle de la sortie


Plus de précision
Pas exact
Précis
Exact
Précis
Pas exact
Pas précis
Exact
Pas précis
Plus d'exactitude

Autres caractéristiques d'un capteur …

- *Coût* (ex. un télémètre laser est typiquement beaucoup plus coûteux qu'une caméra)

- *Taux d'erreur*
  - Niveau d'atténuation et/ou de perturbation d'un signal transmis par le capteur

- *Robustesse* (ex. par rapport aux champs électromagnétiques dans l'environnement)

- *Charge de calcul* (ex. une caméra implique plus de traitements de données qu'un sonar)

- *Taille*, *poids*, *consommation énergétique*, *interface*, *étanchéité*, *etc.*

Quelques exemples de capteurs

Capteurs actifs

1. Capteurs temps-de-vol (extéroceptifs)
2. Encodeurs rotatifs (proprioceptifs)
3. Capteurs d'effort (extéroceptifs)

Capteurs passifs

Caméras (extéroceptifs), voir le Module ME 5.2c « Capteurs et préhenseurs en robotique »

1. Capteurs temps-de-vol

- Information de distance, élément clé pour:
  - Évitement d'obstacles
  - Modélisation de l'environnement
- Sonars et télémètres lasers
- Exploitation de la vitesse de propagation de l'onde
- Pour une onde harmonique monochromatique (laser):

$$v = \lambda f = \frac{\lambda}{T}$$

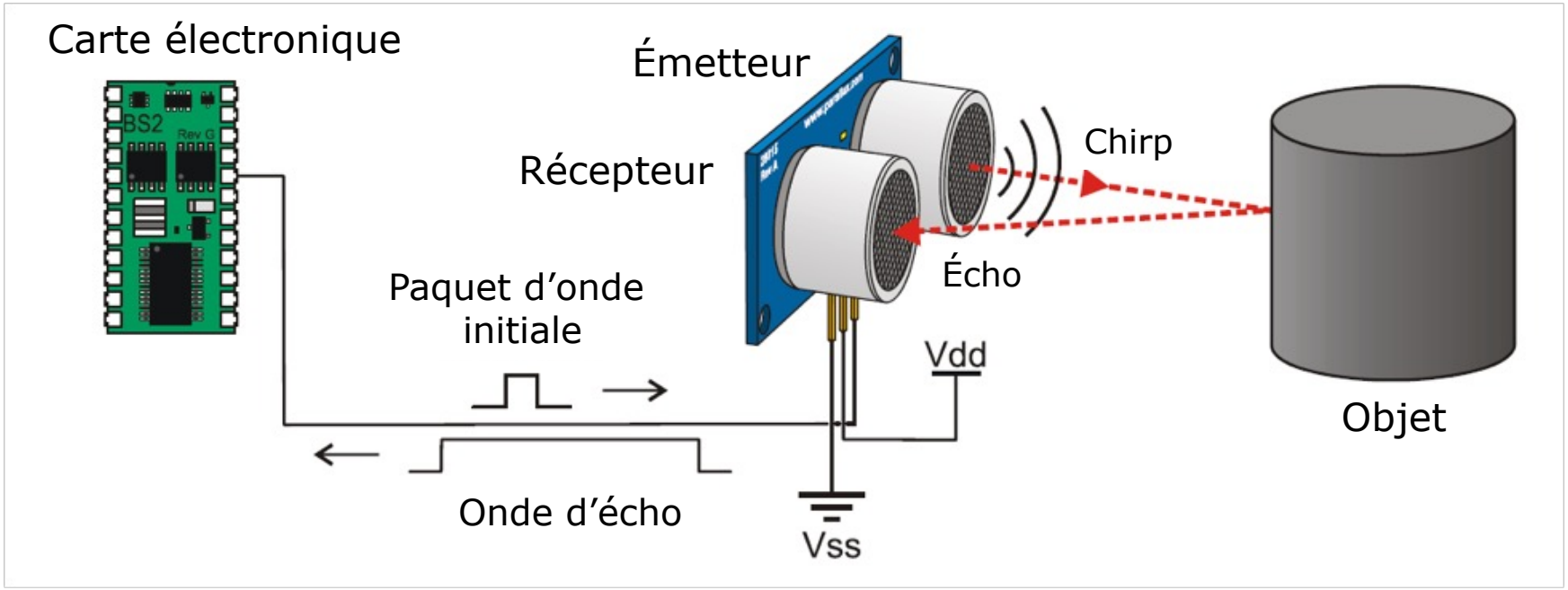
  - v : vitesse de propagation de l'onde
  - f, T : fréquence, période de l'onde
  - λ : longueur d'onde
- $d = \frac{1}{2} v t$: distance parcourue, où t est le temps-de-vol

Capteurs temps-de-vol

- Caractéristiques
  - **Vitesse du son** : 0.3 m/ms (c'est-à-dire, 300 m/s)
    - Sonar : 3 m ⇔ 10 ms
  - **Vitesse de la lumière** : 0.3 m/ns (c'est-à-dire, 300000 km/s)
    - Télémètre laser : 3 m ⇔ 10 ns
      - Besoin d'électronique très rapide
      - Télémètres laser beaucoup plus chers et difficiles à concevoir
- La qualité dépend de:
  - Inexactitudes dans la mesure du temps-de-vol
  - Angle d'ouverture du rayon transmis (pour les sonars)
  - Interaction avec la cible (surface, réflexions spéculaires/diffuses)
  - Variation de la vitesse de propagation (son suivant le milieu)

Exemple: sonar

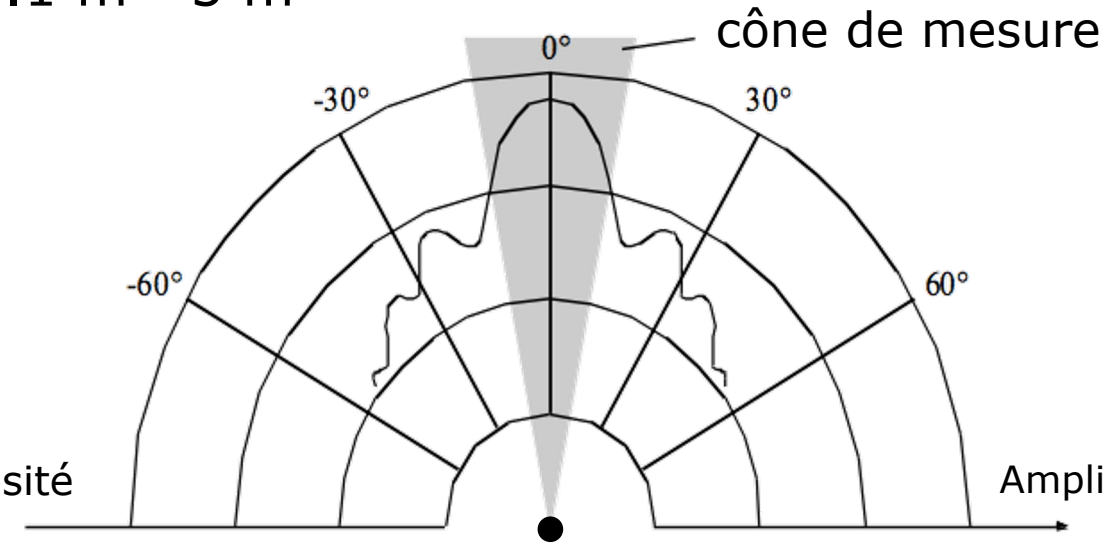
- Principe de fonctionnement du sonar



Parallex PING

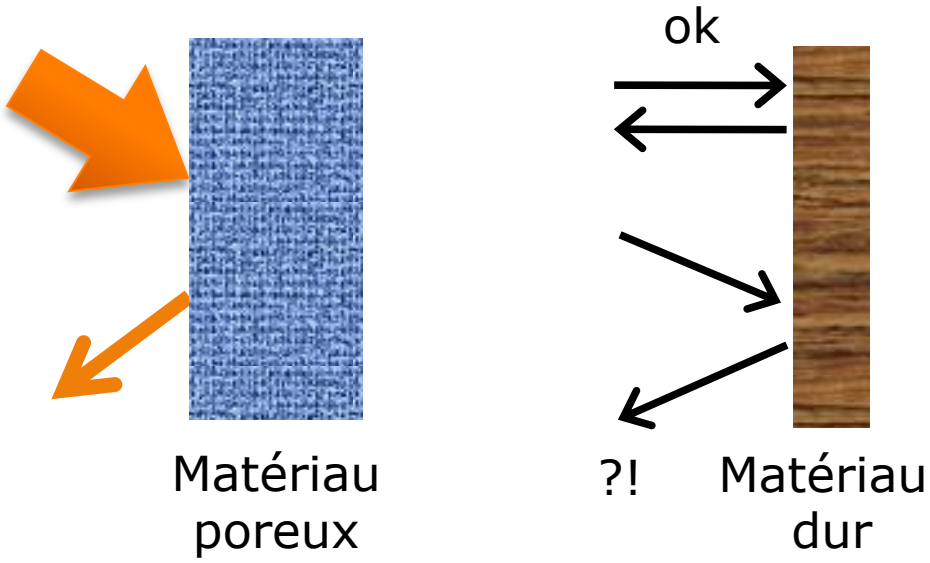
Exemple: sonar

- Fréquences typiques: 40-180 kHz
- Génération de l'onde sonore
  - Piézo-transducteur
  - Émetteur et récepteur séparés ou non
- Le rayon sonore se propage dans un cône
  - Angles d'ouverture entre 20° et 70°
  - Portée: 0.1 m - 5 m



Exemple: sonar

- Problèmes pratiques:
  - Certaines surfaces *absorbant* le son
  - Surface loin d'être perpendiculaire à la direction du son: réflexion spéculaire !



2. Encodeurs rotatifs

Un (en)codeur rotatif est un dispositif qui convertit la *position angulaire* d'un axe ou d'un arbre en code analogique ou binaire

- Résolveurs: trasducteurs électromagnétiques
- Encodeurs mecaniques et optiques (les plus utilisés en robotique)

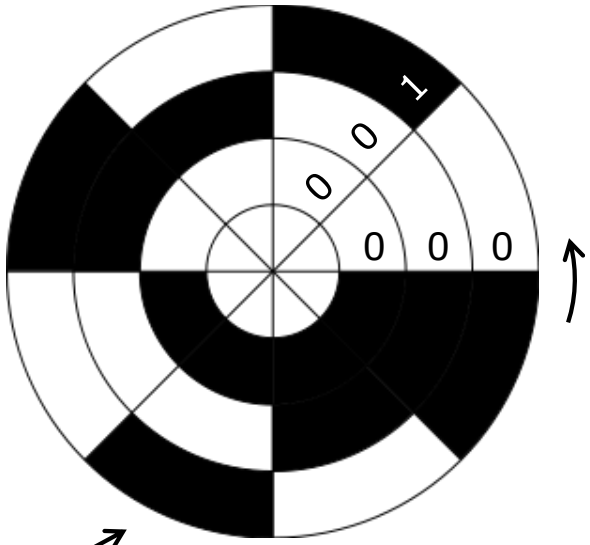
Deux types d'encodeur:

- **Absolu**: la sortie de l'encodeur indique la position courante de l'axe (transducteur d'angle). L'information *n'est pas perdue* lorsque l'alimentation est coupée: elle est disponible à nouveau lorsque l'alimentation est rétablie

- **Incrémental** (ou **relatif**): la sortie de l'encodeur nous informe du mouvement de l'axe. Cette information est normalement traitée ultérieurement pour obtenir des mesures de vitesse, de distance ou de position

Encodeurs rotatifs: exemples

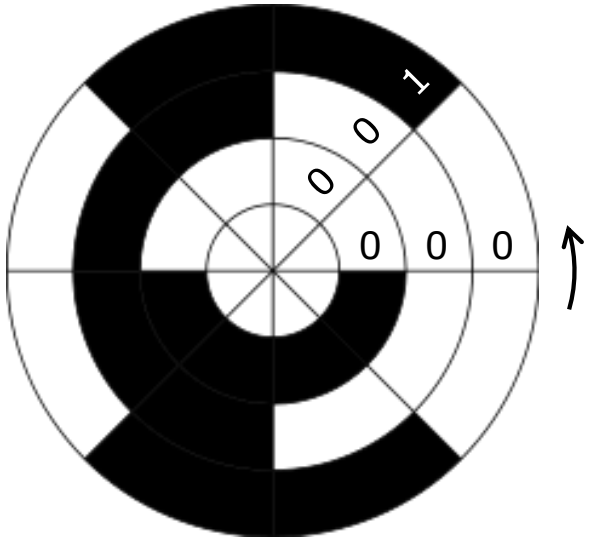
- **Encodeur optique absolu** avec codage binaire (3 bits; noir 1, blanc 0)
  - Un couple émetteur/récepteur par bit



Surface transparente (“blanc”)

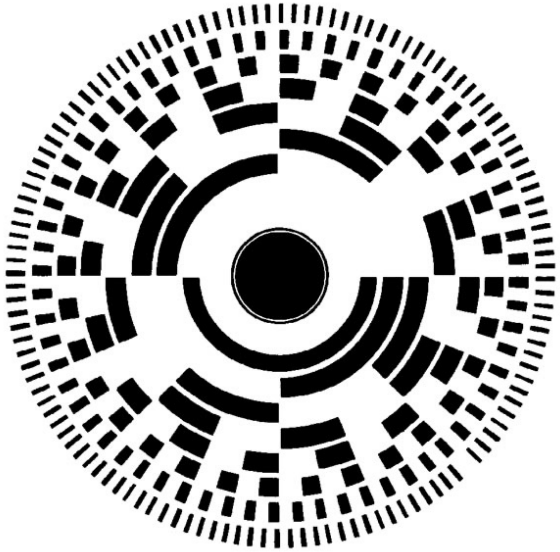
Surface opaque (“noir”)

- **Encodeur optique absolu** avec codage Gray (BRGC: “binary-reflected Gray code"), (3 bits; noir 1, blanc 0)
  - Un couple émetteur/récepteur par bit

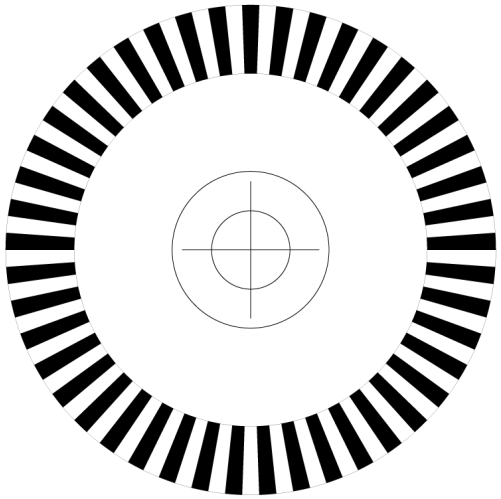


Encodeurs rotatifs: exemples

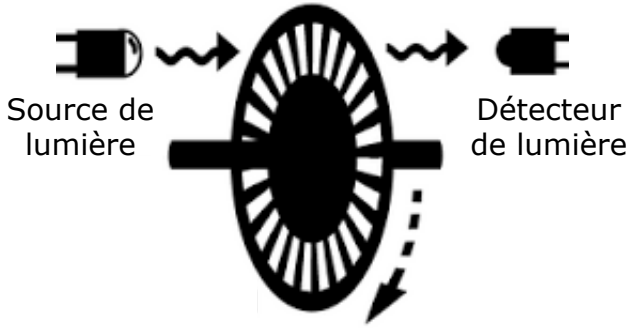
- **Encodeur optique absolu** avec codage binaire (8 bits, codage mixte sur 4 + 4 bits)



- **Encodeur optique incrémental**



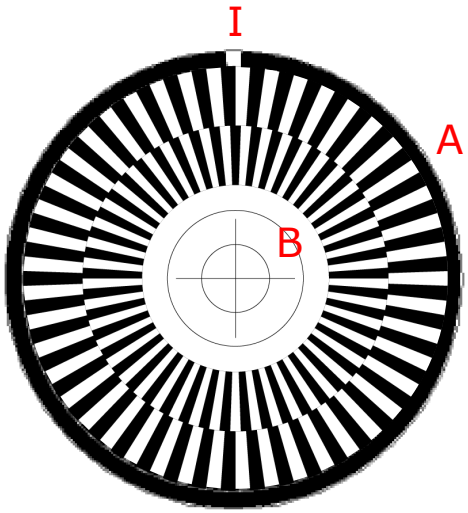
Roue codeuse

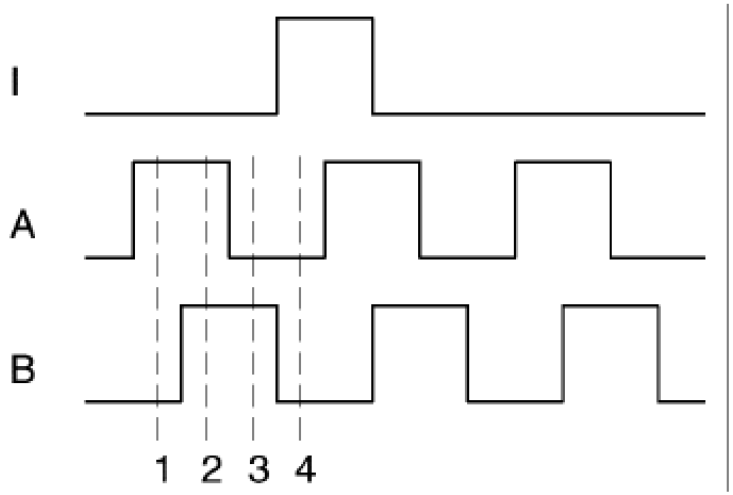


Disque rotatif

Encodeurs rotatifs: exemples

- **Encodeur optique incrémental en quadrature**
  - *Deux canaux*: la relation de phase entre les trains d'impulsions sur les canaux A et B permet de déterminer le sens de rotation
  - Une fente unique sur la piste externe produit une impulsion (index, I) de référence par révolution





état	Cn A	Cn B
E1	haut	bas
E2	haut	haut
E3	bas	haut
E4	bas	bas

3. Capteurs d'effort

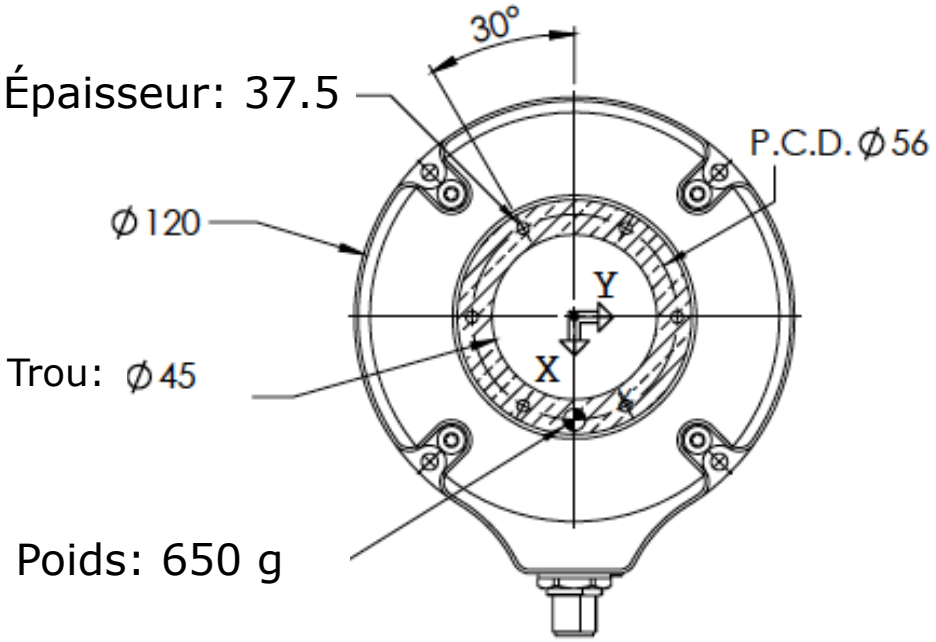
- Ils permettent de mesurer les efforts en *force* et/ou en *couple* (par ex. sur l'effecteur du robot)

- *Capteur actif*: il traduit une variation de sa structure interne (en raison d'une force/couple) en signal électrique

- *Inconvénients* :
  - Coût élevé
  - Étalonnage
  - Fragilité
  - Plages de mesures faibles

Exemple: capteur d'effort à 6 DDL FT 150 de Robotiq

- Mésure du torseur d'effort (Fx, Fy, Fz, Mx, My, Mz) (Forces : Fx, Fy, Fz ; Couples : Mx, My, Mz)
- Compatible avec les robots Universal Robots, Yaskawa, Fanuc, Stäubli et ABB (environnement ROS)



Unité : mm

Exemple: capteur d’effort à 6 DDL FT 150 de Robotiq

Exemples d’application

Assemblage

Guidage de
mains robotisées

Finition

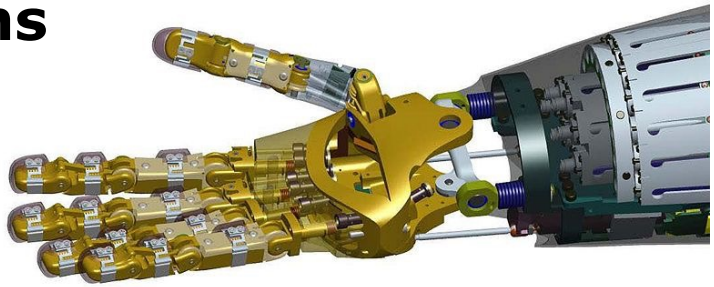
Exemple: capteur d'effort à 6 DDL FT 150 de Robotiq

Caractéristiques techniques du capteur

Plage de mesure	Fx, Fy, Fz Mx, My, Mz	± 150 N ± 15 Nm
Résolution effective	Fx, Fy, Fz Mx, My, Mz	± 0.2 N ± 0.02 Nm
Bruit du signal	Fx, Fy, Fz (combinées) Mx, My, Mz (combinées)	± 0.5 N ± 0.03 Nm
Débit de données en sortie		100 Hz
Tension d'entrée		6-28 V-DC
Puissance absorbée maximale		2 W
Interface électrique		RS-485, RS-232, USB

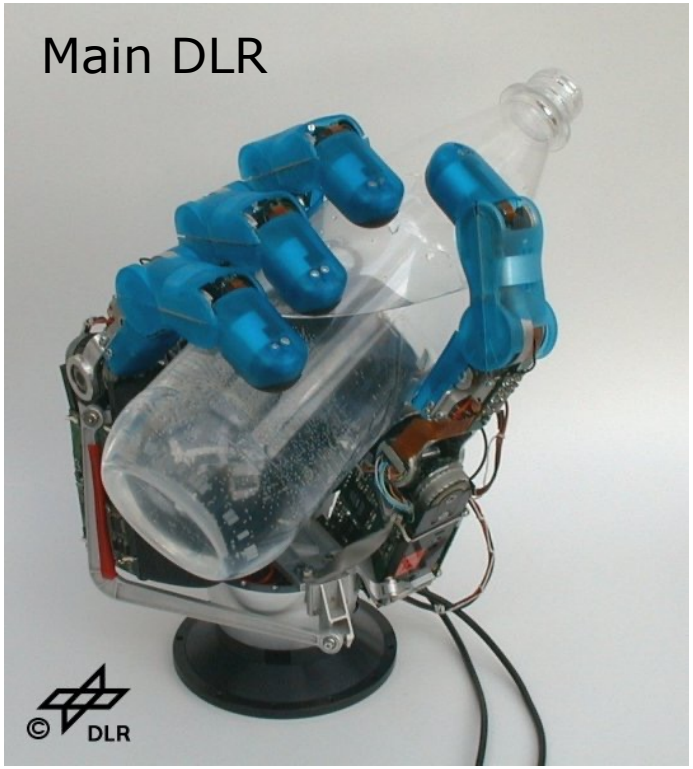
Capteurs d'effort: applications

a) Mains robotiques



Main Robonaut, Nasa

Peau flexible avec une matrice intégrée de transistors organiques

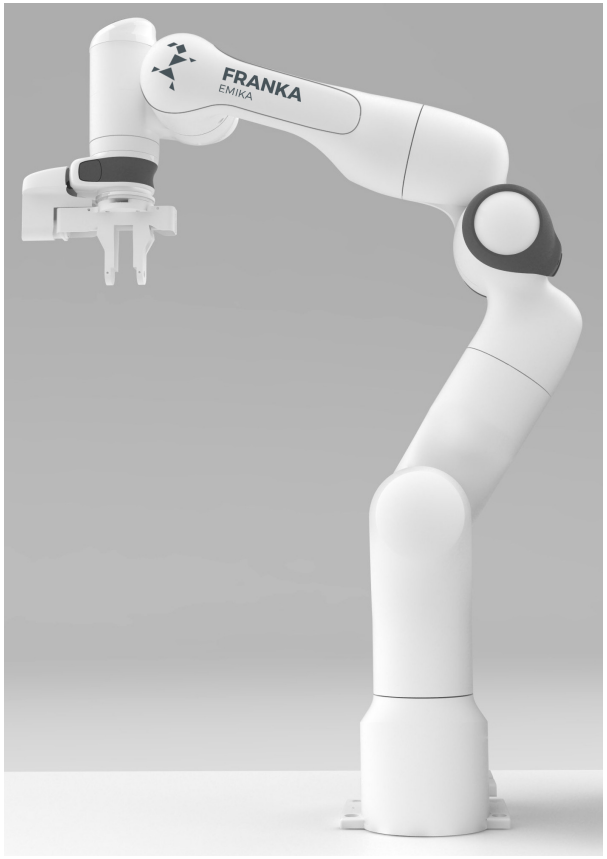


Capteurs d'effort: applications

b) Robots industriels collaboratifs ("Cobots")

Capteurs de couple sur les 7 articulations des robots

LBR IIWA ("lightweight", "intelligent industrial work assistant") de *KUKA*



Panda de *FRANKA EMIKA*

TD1

Tableau récapitulatif : typologies de capteur

A : Actif
P : Passif
PC : Proprioceptif
EC : Extéroceptif

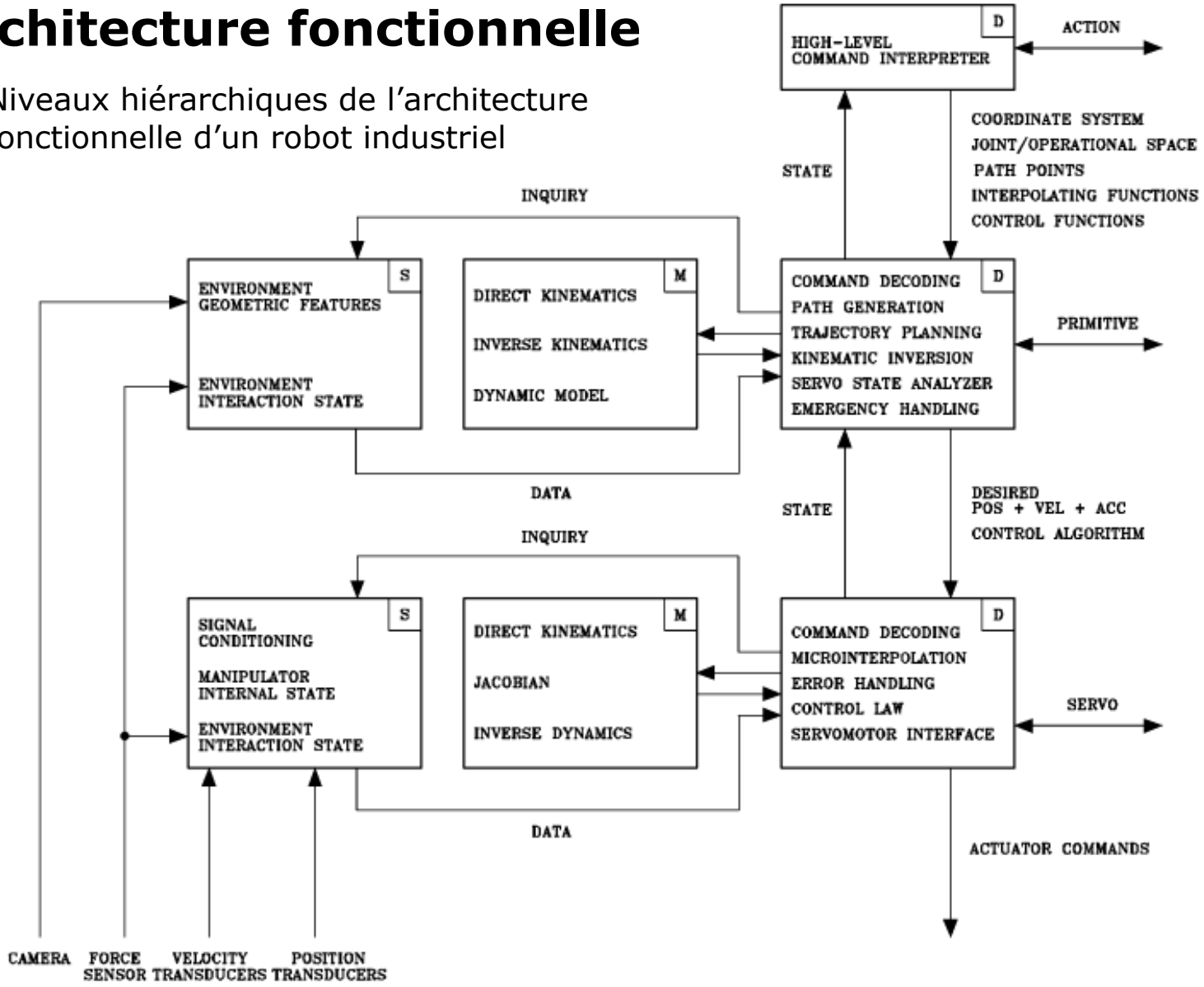
Classification (utilisation typique)	Capteur/système de perception	PC ou EC	A ou P
Capteurs tactiles (détection de contact physique ou de proximité, interrupteurs de sécurité)	Interrupteurs de contacts, bumpers	EC	P
	Barrière optique	EC	A
Capteurs de roue/moteur (vitesse et position de roue/moteur)	Encodeurs à balais	PC	P
	Potentiomètres	PC	P
	Encodeurs optiques	PC	A
	Encodeurs magnétiques	PC	A
	Encodeurs inductifs	PC	A
	Encodeurs capacitifs	PC	A

Classification (utilisation typique)	Capteur/système de perception	PC ou EC	A ou P
Capteurs d'orientation (orientation du robot en relation à un référentiel fixe)	Compas	EC	P
	Gyroscope	PC	P
	Inclinomètre	PC	A/P
Basé balise (localisation dans un référentiel fixe)	GPS	EC	A
	Balise active optique ou radio	EC	A
	Balise active à ultrasons	EC	A
	Balises réflectives	EC	A

Classification (utilisation typique)	Capteur/système de perception	PC ou EC	A ou P
Télémétrie active (réflectivité, temps de vol et triangulation géométrique)	Capteurs de réflectivité	EC	A
	Capteur à ultrasons	EC	A
	Télémètre laser	EC	A
	Triangulation optique (1D)	EC	A
	Lumière structurée (2D)	EC	A
Capteurs de mouvement/vitesse (vitesse relative à des objets statiques ou fixes)	Doppler radar	EC	A
	Doppler sonore	EC	A
Capteurs de vision (télémétrie visuelle, analyse de l'image complète, segmentation, reconnaissance d'objet)	Caméras à capteur CCD/CMOS	EC	P

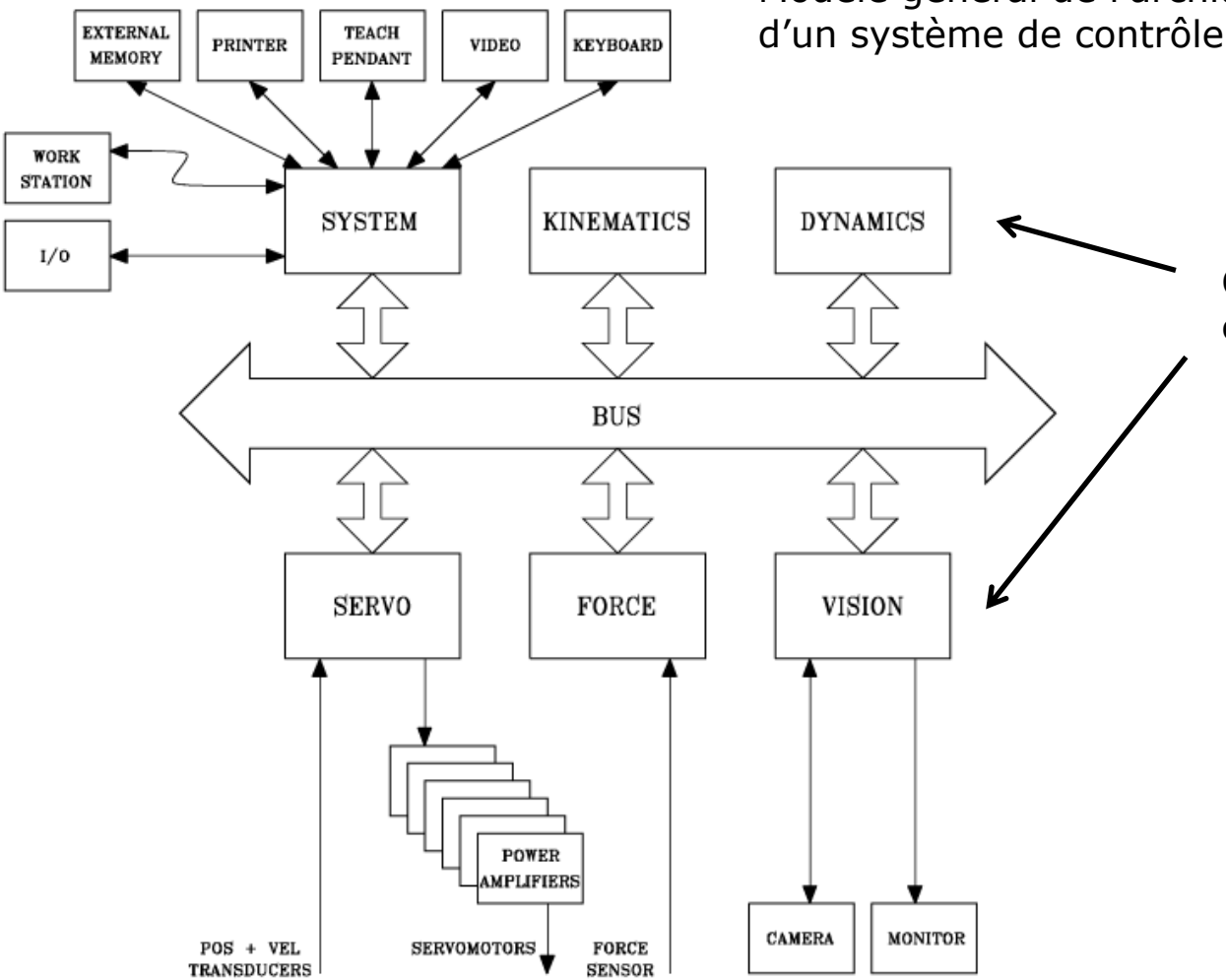
Architecture fonctionnelle

- Niveaux hiérarchiques de l'architecture fonctionnelle d'un robot industriel



Architecture de contrôle

Modèle général de l'architecture hardware d'un système de contrôle d'un robot industriel

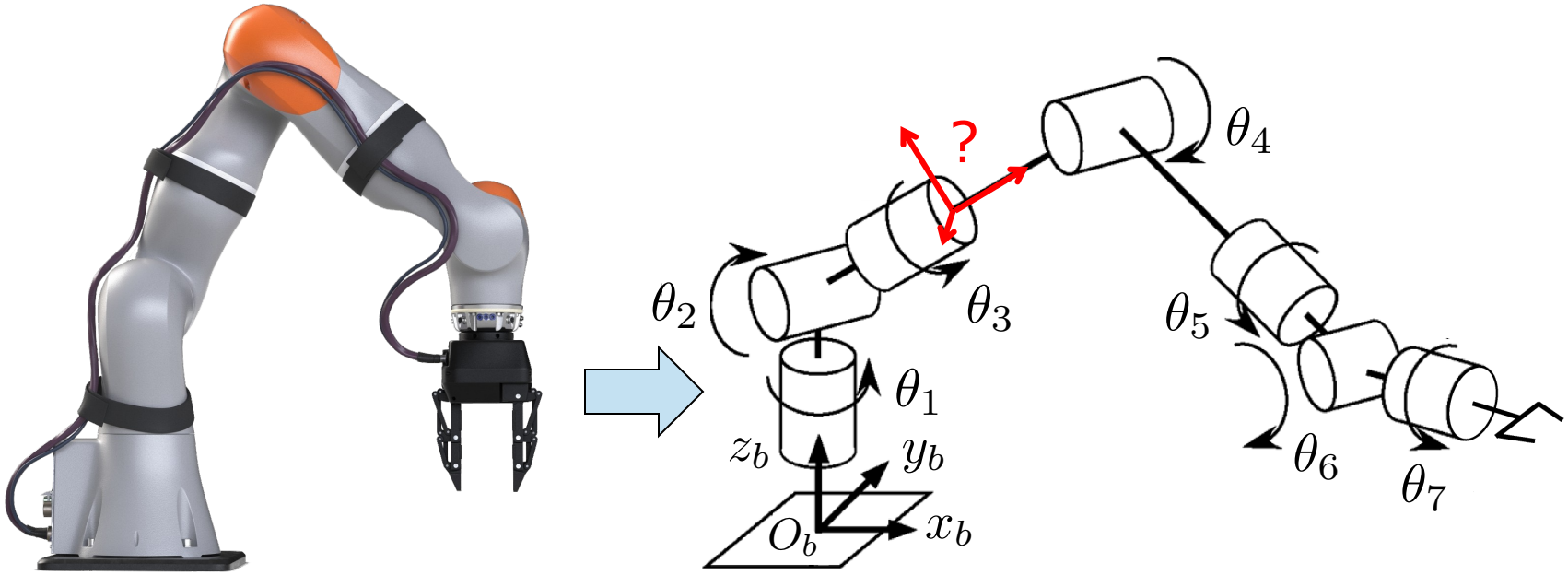


Capacité de calcul autonome

Plan du cours

- Introduction
- Constituants et caractéristiques d'un robot
- Gammes de robots et secteurs d'activités
- Les baies de commandes, le boîtier d'apprentissage, les modes et la programmation d'un robot
- Actionneurs et capteurs d'un robot
- Repères et transformations homogènes
- Étude de cas: cellule robotisée de soudage

Choix des repères



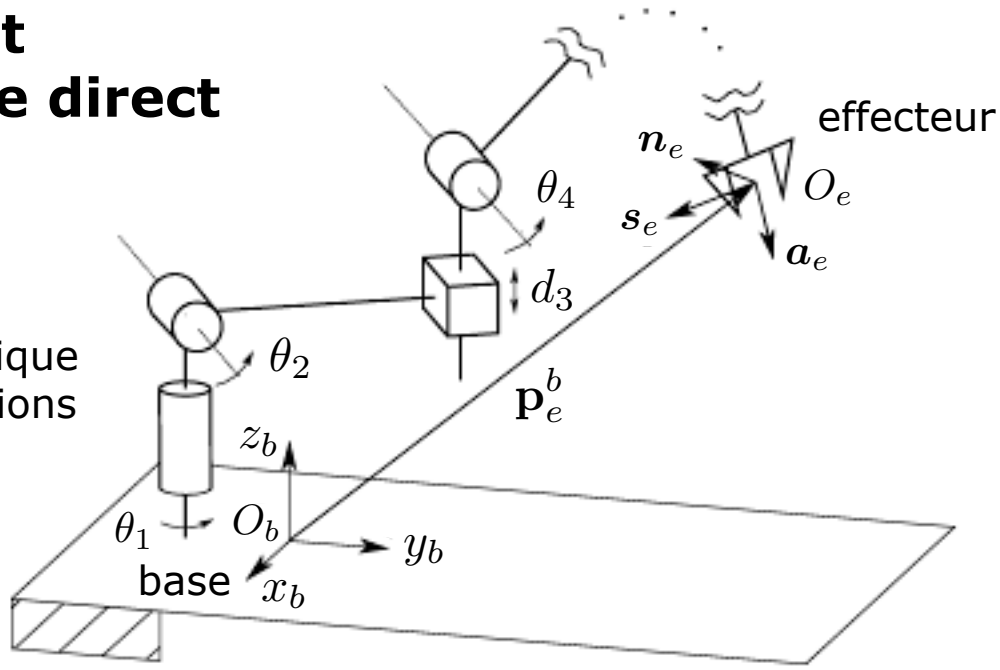
Problème :

Comment positionner les différentes systèmes de coordonnées (*repères*) sur les segments d'un robot à partir du repère de la base $O_b - x_b\, y_b\, z_b$?

Pour un *choix systématique* des repères sur toutes les segments d'un robot, on utilise la **convention de Denavit-Hartenberg (DH)**

Convention de DH et modèle géométrique direct

Robot générique
à n articulations



• Par rapport au repère de la base $O_b - x_b\, y_b\, z_b$, le *modèle géométrique direct* est exprimé par la matrice de transformation homogène 4 × 4:

où

$$\mathbf{T}_e^b(\mathbf{q}) = \begin{bmatrix} \boldsymbol{n}_e^b(\mathbf{q}) & \boldsymbol{s}_e^b(\mathbf{q}) & \boldsymbol{a}_e^b(\mathbf{q}) & \mathbf{p}_e^b(\mathbf{q}) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{q} \in \mathbb{R}^n$: vecteur des variables des articulations

$\boldsymbol{n}_e^b$, $\boldsymbol{s}_e^b$, $\boldsymbol{a}_e^b$: vecteurs unitaires du repère de l'effecteur par rapport à la base

$\mathbf{p}_e^b$: vecteur qui décrit l'origine du repère de l'effecteur par rapport à la base