

## Séance 2

# Réalisation expérimentale d'un robot sériel : montage, caractérisation et commande des actionneurs.

Pour rappel, et comme indiqué dans l'introduction, les questions repérées par une marque noir sur la marge sont les questions pour lesquelles il est demandé une réponse détaillée et argumentée dans votre compte-rendu final.

### 1 Présentation du système robotique

L'objectif de ce projet est la réalisation et le contrôle d'un robot de type "bras" fonctionnant dans le plan, et réalisé à partir de moteurs tous identiques. Il s'agira plus particulièrement :

- de connecter le robot à une alimentation et à son contrôleur;
- de caractériser chacun des actionneurs du système;
- et de contrôler en position l'organe terminal du bras.

Comme pour le TP précédent, le contrôle des différents moteurs s'effectuera depuis Python via une librairie fournissant une interface USB qui permet de dialoguer avec chacun des moteurs constituant le robot. Une fois le robot assemblé et ses moteurs fonctionnels, il s'agira alors d'établir différents modes de contrôle permettant au bras de réaliser une tâche donnée (écriture, prise/dépôt d'une pièce, etc.) Pour cela, les fonctions utilisées lors du TP1 seront réutilisées autant que possible, moyennant leur adaptation à la modélisation du robot réel (voir pour cela l'annexe).

#### 1.1 Les moteurs DYNAMIXEL

DYNAMIXEL est une gamme de servomoteurs programmables pour robots prenant la forme d'un module tout-en-un (réducteur, contrôleur, fonction réseau et pilote). Petits, légers, puissants, les actionneurs utilisés dans ce projet font partie de la gamme AX, regroupant les servomoteurs d'entrée de gamme. Ces moteurs s'appuient sur un protocole TTL afin de dialoguer entre eux et/ou avec leur interface USB, et exploitent un protocole de communication série entièrement documenté. Plus particulièrement, le constructeur fournit des interfaces de programmation (API) pour différents langages permettant d'exécuter toutes ou partie de fonctions supportées par les moteurs. Les caractéristiques des moteurs AX12 utilisés pendant ce projet sont synthétisées dans le tableau 1.

TABLE 1 – Dynamixel AX12

Tension de fonctionnement	9 à 12V
Couple décrochage	1,52 Nm à 12V et 1,5A
Vitesse hors-charge	59 tours/minute (à 12V)
Rapport de réduction	254 :1
Vitesse de communication	7343bps ~ 1 Mbps
Lien (Physique)	TTL Level Multi Drop (daisy chain type Connector)
Protocole	Half duplex Asynchronous Serial Communication (8bit,1stop, No Parity)
Retour	position, température, charge, tension d'alimentation, etc.
Poids	54,6g

Les moteurs sont identifiés par un ID unique normalement 1,2 et 3 suivant l'emplacement des moteurs. Si un des moteurs n'est pas reconnu, il faudra le flasher (voir annexe). Pour les moteurs Dynamixel AX-12, la plage de position qui va de 0 à 1023 (10 bits), ce qui correspond à une plage de rotation d'environ 300° ( $\pm 150^\circ$  autour de la position centrale), cf. figure 1. La vitesse de déplacement d'une position à une autre est fixe et peut être réglée.

#### 1.2 Les interfaces USB2DYNAMIXEL ou U2D2

USB2DYNAMIXEL et U2D2 sont tous deux une interface USB qui émule un port série vers lequel les communications pour les moteurs DYNAMIXEL doivent transiter. Les moteurs sont ensuite connectés sur cette interface USB en série, comme indiqué sur la figure 2 (pour l'interface USB2DYNAMIXEL).

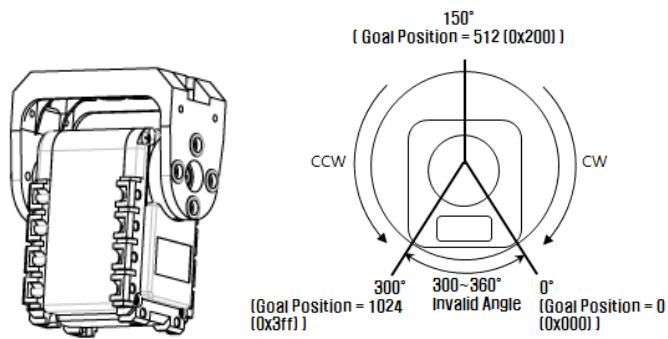


FIGURE 1 – (gauche) Moteur AX12. (droite) Convention pour le repérage des positions angulaires du moteur.

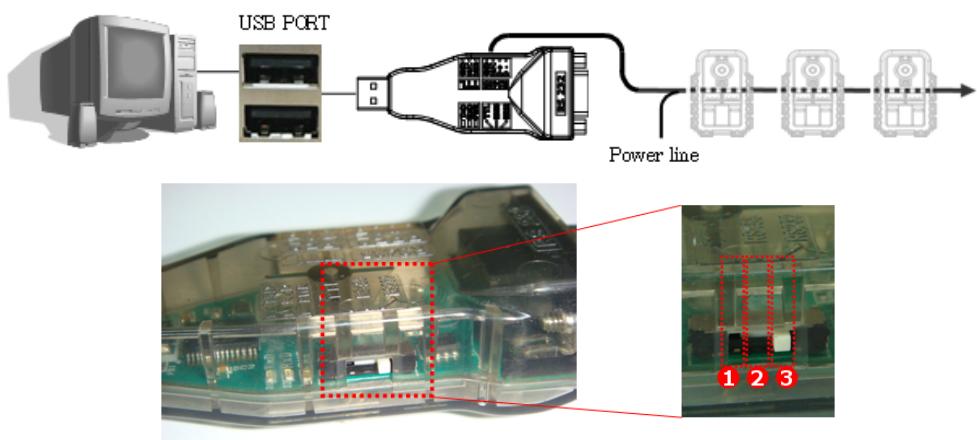


FIGURE 2 – Interface USB2Dynamixel pour le contrôle des moteurs.

Il est nécessaire de connaître l'identifiant de ce port série pour pouvoir y envoyer des commandes. Voici différents moyens de le trouver selon votre machine. Tout d'abord, alimentez de le robot puis connectez le via USB à votre machine.

- Linux : tapez `ls /dev/ttyUSB*` dans le terminal (retourne par exemple '`/dev/ttyUSB0`').
- Mac : tapez `ls /dev/tty.usbserial*` dans le terminal (retourne par exemple '`/dev/tty.usbserial-XXXXXX`' )
- Windows : Allez dans votre gestionnaire de périphériques puis dans la section Port (COM et LPT), vous verrez USB Serial Port (COM). Le numéro COM sera le nom du port à connaître (cf. figure 3).

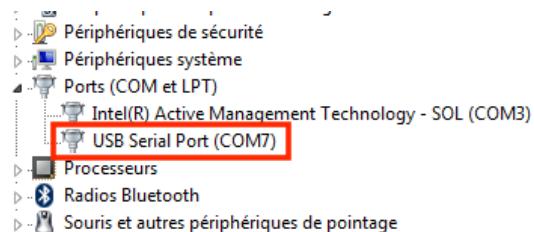


FIGURE 3 – Détermination du numéro du port série sur Windows.

Les interfaces USB2DYNAMIXEL et U2D2 permettent également de spécifier différents modes de communication vers les moteurs. Pour son bon fonctionnement avec les moteurs AX12 utilisés pendant ce projet, il est nécessaire de placer le commutateur visible sur la figure 2 en position 1 (TTL). Pour l'interface U2D2, le câble 3 broches doit être connectée sur la prise TTL (c'est de toute façon la seule disponible à 3 fils).

### 1.3 Librairie Python

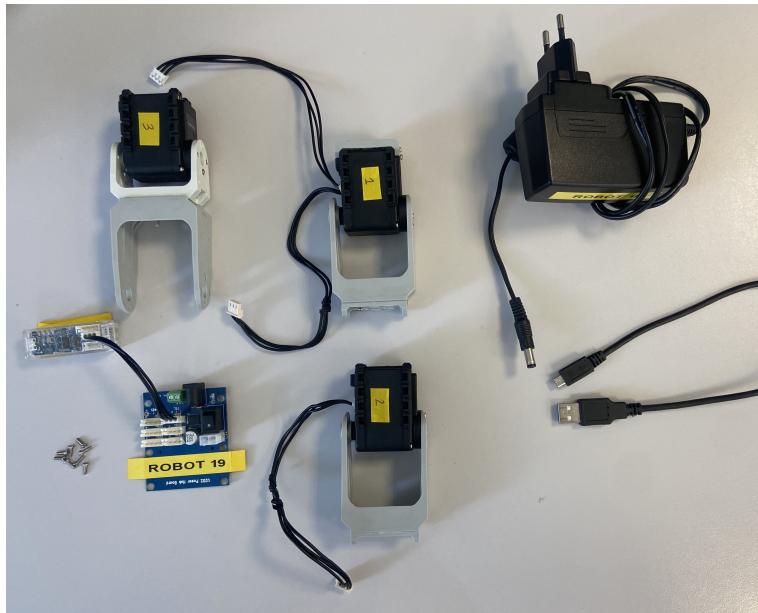
Pour communiquer avec les moteurs Dynamixel nous vous proposons d'utiliser une bibliothèque Python qu'il faut télécharger (`pip install dynamixel-controller`)

1. Pour mieux comprendre comment programmer les moteurs Dynamixel, nous avons fait un tutoriel sur notebook `test_moteur_dynamixel.ipynb`. Munissez vous d'un moteur Dynamixel AX12-A et suivez le tutoriel.

### 1.4 Construction de la maquette

Vous disposez des éléments suivant :

- une interface USB2DYNAMIXEL ou U2D2 et sa rallonge USB,
- une entrée d'alimentation possiblement équipée d'un interrupteur,
- 3 moteurs AX12-A et ses supports
- une alimentation,
- différents câbles de raccordement (alimentation et communication).
- 8 vis pour assembler le robot



1. Branchez les moteurs un à un en série.
2. Puis assemblez le robot à l'aide des vis fournies si celui ci n'est pas monté.
3. A l'aide des connecteurs fournis, brancher le moteur à l'USB2DYNAMIXEL. Les 2 connecteurs présents sur le moteurs peuvent être utilisés indifféremment. Vérifier que l'interrupteur présent sur l'USB2DYNAMIXEL est bien en position 1 (TTL).
4. Connecter l'alimentation sur la maquette et le secteur et connecter la rallonge USB à l'ordinateur et à l'USB2DYNAMIXEL. Allumer la puissance grâce à l'interrupteur présent sur la maquette. La diode présente sur le moteur doit normalement s'allumer un instant, puis s'éteindre.

## 2 Travail demandé

### 2.1 Prise en main du contrôle des moteurs

1. Editer le contenu du fichier `main.py`:

- modifier la valeur de la variable `portName` de façon à ce qu'elle soit identique au numéro du port série identifié),
  - modifier la valeur de la variable `connected` à True.
2. Utiliser la fonction `set_position` de la bibliothèque dynamixel-controller pour bouger le moteur commandant le bout du bras en une position pos quelconque ( l'appel des moteurs ce fait via l'instance de Univers : `univers.motors` est la liste des moteurs donc `univers.motors[0].set_position(500)` permet déplacer le moteur 1 à la position 500). Commenter.
  3. Vous pouvez aussi modifier la valeur de la vitesse de rotation du moteur à l'aide de la fonction `set_velocity`, et ré-exécuter différents mouvements (consulter la section dédiée à la vitesse pour déterminer les paramètres/-valeurs à fournir à la fonction).

## 2.2 Caractérisation des actionneurs

Comme vous avez pu le remarquer, les appels aux fonctions ne sont pas bloquants. Ainsi, il est possible, par exemple, de lire la position et/ou vitesse du moteur tandis que celui-ci est en mouvement. Nous allons utiliser cette possibilité pour caractériser le fonctionnement de ces actionneurs.

Travail suivant dans la classe `Motor`.

### Utilisation du moteur en bout de chaîne cinématique :

4. Exprimer la relation mathématique reliant les valeurs des pas moteurs lues depuis l'actionneur en valeur angulaire exprimée en degrés.
5. En déduire une fonction '`pos_to_deg`' qui permet de convertir une valeur de pas moteur ('`step`') en un angle '`q`' exprimé en degrés.
6. Inverser la relation obtenue précédemment pour obtenir l'expression reliant cette fois un angle exprimé en degrés à la valeur de pas moteur correspondante.
7. En déduire une fonction `deg_to_pos` qui convertit un angle '`q`' exprimé en degrés en une valeur de pas moteur '`p`'.
8. Pour un mouvement allant de la position moteur 0 à 1023 à une vitesse moyenne, tracer en fonction du temps :
  - (a) Cette position sera exprimée d'abord en "pas moteur", puis en degrés ou radians.
  - (b) Effectuer ce tracé pour différentes valeurs de consignes de vitesse.
  - (c) Sur la base des tracés en position obtenus, vérifier que la consigne de vitesse est effectivement respectée. Expliquer la méthode et commenter.
9. La bibliothèque permet également de mesurer directement la vitesse de rotation et position du moteur à l'aide de la fonction. En tenant compte des spécifications rappelées dans la section dédiée, écrire une fonction `getSpeed` qui convertit la valeur brute renvoyée par `get_velocity` en une vitesse `v` exprimée en tours/-minute.
10. Pour un mouvement allant de la position moteur 250 à 950, tracer l'évolution de la vitesse (exprimée en tour/min) de l'actionneur en fonction du temps. Comparer cette valeur mesurée avec la valeur de consigne, réglée à l'aide de la fonction `set_velocity`. Commenter et justifier.
11. La courbe obtenue en 8. permet également d'estimer la vitesse de rotation du moteur par différence finie. Tracer cette vitesse (estimée à partir de la courbe de la position en fonction du temps), et la comparer avec celle lue directement sur le moteur. Conclure.
12. Effectuer plusieurs essais successifs identiques, et évaluer la répétabilité des caractéristiques du mouvement (vitesse et position au cours du temps plus particulièrement) :
  - (a) Choisir une position initiale et finale, et amener le moteur plusieurs fois successives en ces positions, pour différentes vitesses;
  - (b) Estimer la moyenne des erreurs de positions en fonction de la vitesse de déplacement utilisée. Conclure quant à l'influence de la vitesse sur la précision en position angulaire du moteur.

### 2.3 Maquette du robot 3R plan

Il s'agit maintenant de contrôler un robot 3R plan que nous allons commander à l'aide des fonctions utilisées lors du TP1 pour modéliser le robot réel.

1. Représenter le schéma cinématique du robot réalisé. Nommer les angles de rotation et identifier les paramètres géométriques nécessaires.
2. À l'aide de la fonction MGD, déterminer en simulation la position '(x, y)' de l'organe terminal pour deux commandes motrices 'q1' et 'q2' de votre choix. Appliquer ces deux commandes au robot réel et vérifier la correspondance entre la position atteinte et celle obtenue en théorie.
3. Faire faire une trajectoire articulaire de votre choix à la maquette. Pendant le mouvement du robot, lire en continu les positions articulaires des moteurs et afficher le comportement du modèle sur l'écran. Comparer les mouvements de la maquette réelle et de son modèle.