

# **Technologie de systèmes ordinés (Électronique programmable et robotique)**

**247-4B6-GG**

**Réaliser un projet de logiciel appliqué à la robotique**

## **Lab 1:**

**Dépendances et comprendre la librairie**

**Enseigné par Rachid Benali**

**Laboratoires et programmation réalisés par  
Félix Chenette-Stewart**

# Téléchargement de la librairie

Dans la console écrire :

Git clone [https://github.com/filoucool/Labos\\_Python\\_Rachid.git](https://github.com/filoucool/Labos_Python_Rachid.git)

Puis se rendre dans le nouveau dossier en écrivant :

cd Labos\_Python\_Rachid

Tous vos codes devront être sauvegardés dans le même dossier que la librairie Ax12.py soit dans Labos\_Python\_Rachid/Ax12/

## Librairies nécessaires

- PySerial pour la communication avec les moteurs
- RPi.GPIO pour l'utilisation du GPIO du raspberry pi 3.

Pour installer les librairies si ce n'est pas déjà fait:

```
>pip install pyserial  
>pip install RPi.GPIO
```

## Dépendances

Tous vos projets et scripts devront commencer avec le code suivant afin d'assurer le bon fonctionnement des librairies pour les AX-12.

```
from time import sleep  
from serial import Serial  
import RPi.GPIO as GPIO  
import time  
from Ax12 import Ax12
```

# Initialisation

Ce bloc de code est nécessaire au début de chaque code pour initier la librairie et s'assurer que les moteurs répondent aux commandes

```
delay_0=0.001
ax12_Lib = Ax12()
ax12_Lib.__init__()
time.sleep(1)

dynamixel_id1 = 17
dynamixel_id2 = 18
ax12_Lib.ping(dynamixel_id1)
time.sleep(delay_0)
ax12_Lib.ping(dynamixel_id2)
time.sleep(delay_0)
```

ax12\_Lib = Ax12()

- Fait appel à la classe et lui donne une référence.

ax12\_Lib = \_\_init\_\_()

- Initialise la librairie et crée une connexion aux moteurs.

Dynamixel\_id1 = (votre id moteur droit)

- Identification du moteur droit.

Dynamixel\_id2 = (votre id moteur gauche)

- Identification du moteur gauche.

ax12\_Lib.ping(dynamixel\_id1)

- Envoi un signal au moteur de droite pour vérifier si il répond aux commandes.

ax12\_Lib.ping(dynamixel\_id2)

- Envoi un signal au moteur de gauche pour vérifier si il répond aux commandes.

Ax12\_Lib.sleep(delay\_0)

- **Fonction très importante!** Permet au code de s'arrêter en attendant la réponse du robot. Oublier de mettre cette ligne de code entre chaque commande pourrait causer des problèmes ou des malfonctionnements.

# API

```
def ping(self, id):
```

Utilisé pour envoyer un signal au moteur (id) pour s'assurer que le moteur répond aux commandes

Utilisation: ax12\_Lib.ping(dynamixel\_id1 OU dynamixel\_id2)

```
def move(self, id, position):
```

Utilisé pour faire tourner le moteur avec précision selon le nombre de « steps ».

La valeur de la position doit être entre 0 et 1023 ou entre 0 et la limite d'angle déterminée par setAngleLimit.

Utilisation : ax12\_Lib.move(dynamixel\_id1, 400)

```
def moveSpeed(self, id, position, speed):
```

Utilisé pour faire tourner le moteur avec précision selon le nombre de « steps » tout en déterminant une vitesse.

La valeur de la position doit être entre 0 et 1023 ou entre 0 et la limite d'angle déterminée par setAngleLimit.

Utilisation : ax12\_Lib.moveSpeed(dynamixel\_id1, 300, 200)

```
def Speed(self, id, speed):
```

Utilisé pour définir la vitesse des moteurs.

La vitesse peut être de 0 à 1023 pour le sens horaire. Pour faire tourner le moteur dans le sens inverse, il faut ajouter 1024 à la valeur initiale.

Utilisation : vitesse = 500

ax12\_Lib.Speed(dynamixel\_id1, vitesse)

ax12\_Lib.Speed(dynamixel\_id2, 1024 + Vitesse)

```
def setAngleLimit(self, id, cwLimit, ccwLimit):
```

Utilisé pour déterminer les limites de rotation des moteurs pour le sens horaire (cwLimit) et le sens anti-horaire (ccwLimit). Les valeurs possibles se situent entre 1 et 1023. Par contre, pour complètement enlever les limites de rotations, la valeur doit être 0.

Utilisation : ax12\_Lib.setAngleLimit(dynamixel\_id1, 1023, 1023)

ax12\_Lib.setAngleLimit(dynamixel\_id1, 0, 0)

# Modes de déplacement

## Wheel Mode

Le mode « wheel » permet de faire tourner les roues continuellement tant et aussi longtemps que la vitesse des moteurs est supérieure à 0.

Le bloc de code suivant permet de mettre les deux moteurs en mode « wheel » en enlevant les limites de rotation horaires et anti-horaires des moteurs.

```
ax12_Lib.setAngleLimit(dynamixel_id, 0, 0)
```

En « wheel » mode, on peut faire avancer le robot pour un certain nombre de secondes en ajoutant un `time.sleep(nombre de secondes)`.

Exemple d'utilisation :

```
vitesse = 200
ax12_Lib.Speed(dynamixel_id1, vitesse)
time.sleep(delay_0)
ax12_Lib.Speed(dynamixel_id2, 1024+vitesse)
time.sleep(1)
ax12_Lib.Speed(dynamixel_id1, 0)
time.sleep(delay_0)
ax12_Lib.Speed(dynamixel_id2, 0)
```

Ce code fera avancer le robot pendant 10 secondes puis arrêtera le robot.

## Stepper Mode

Le mode « stepper » permet de faire tourner les roues avec précision. Chaque revolution completes constituent un total de 1023 « steps » donc 1023 degrés de précision.

Le bloc de code suivant permet de mettre les deux moteurs en mode « stepper » en donnant une limite de rotation en « steps ».

```
ax12_Lib.setAngleLimit(dynamixel_id, 1023, 1023)
```

# Exercices

Dans un document séparé, répondre aux questions suivantes :

Manipulation 1: faire un code complet pour mettre les DEUX moteurs en mode stepper

Manipulation 2: faire un code complet pour mettre les DEUX moteurs en mode moteur (wheel mode)

Manipulation 3: faire un code pour faire...

1. Avancer le robot pendant 3 secondes

2. Reculer le robot pendant 4 secondes

3. Tourner le robot sur lui-même.