**Rapport des tests**

**Projet KERAAL - Groupe 10**

**Equipe:** Baumstimler Philippe, Ma Ziqi, Aziz Ghaddab Mohamed





# Sommaire

[**Introduction**](#_w91t907exh44) **3**

[**Tests informatiques**](#_l340r9e33bm9) **4**

[Test logiciel robot POPPY](#_bzibkm7dt7ev) 4

[Test fonctionnement système du bloquage et débloquage](#_p0i3gyjeeq9v) 7

[Test du système à l’état initial](#_stfniwaxauss) 9

[Test contrôle POPPY](#_6nsioxyofi90) 10

[Test enregistrement mouvement](#_n0sj4rli6z44) 11

[**Test mécanique**](#_169x6bu0w8gp) **13**

[Test matériel d’assemblage du robot POPPY](#_s76w2n22egk1) 13

[Test fonctionnement de l’alimentation](#_15is35534lee) 13

[Test connectique du vérin électrique](#_svgrsyyligto) 14

[Test fonctionnement de l’optocoupleur](#_izx6ile86ind) 14

[**Conclusion**](#_qut34d1wtluu) **16**

[**Annexe**](#_jixgqqt18zny) **17**

[check\_port.py](#_baewyhgfqsti) 17

[test\_blocage.py](#_5gtsrxbj6vh5) 17

[test\_ecart\_limite.py](#_pwl7v55khfiv) 18

[test\_record.py](#_ha9vimjvx00m) 19

# 

# **Introduction**

Le rapport de tests décrit l’ensemble des trames de tests à réaliser pour valider la conception réalisée en amont. Ce document n’a pour autant pas pour but de valider la conception, mais fait preuve de résultats objectifs vis-à-vis de la performance des systèmes conceptualisés par rapport aux exigences imposées. Ces tests font partie intégrante de la vie du projet, et sont présents dans l’ensemble des pôles, informatique et mécanique.

# **Tests informatiques**

Les tests informatiques sont l’ensemble des tests réalisés pour valider le bon fonctionnement et de la bonne installation des logiciels propres au projet. Nous avons fait le choix de mettre dans cette partie également les tests de fonctionnement du robot POPPY dans son ensemble, puisqu’ils passent par logiciel et par des scripts python.

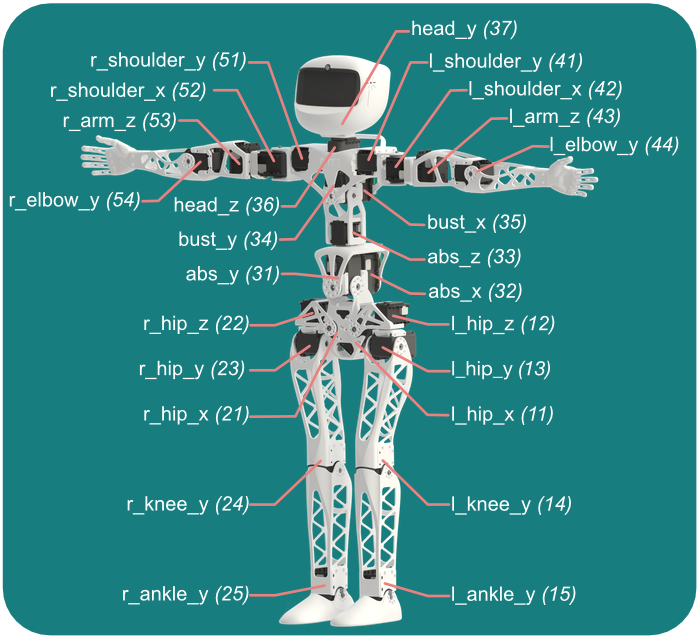
## **Test logiciel robot POPPY**

Nous commençons tout d’abord par la réalisation des tests d’installation des logiciels et de la librairie POPPY. Nous vérifions également que la connexion entre la carte et les moteurs se fait bien. Les commandes décrites sont directement écrites dans un terminal python en ssh sur la Raspberry Pi.

| **Test** | **Commandes/test réalisé** | **Résultats et Validation** |
| --- | --- | --- |
| **Bonne alimentation du robot** | On branche le robot | L’ensemble des LEDs s’allument, test validé |
| **Bonne connexion entre le PC et la carte** | Connexion au même wifi | La carte communique bien avec le logiciel, test validé |
| **Bonne installation des librairies Poppy** | from poppy\_humanoid import PoppyHumanoid | Aucune erreur d’installation, test validé |
| **Bonne initialisation du robot** | poppy = PoppyHumanoid() | Erreur, les moteurs du haut du corps ne sont pas détectés, le message d’erreur est du type:  IOError: Connection to the robot failed! No suitable port found for ids [3, 5, 7, 11, 13, 17]. These ids are missing [3, 5, 7, 11, 13, 17] ! |

Nous faisons ici face à un premier problème, nous n’arrivons pas à détecter les moteurs de tout le haut du corps. Nous essayons de passer par les fichiers sources pour savoir si l’erreur vient de la librairie ou bien de la connectique. Finalement l’erreur a été trouvée, il s’agissait du câble qui reliait le tronc à la tête qui était endommagé.

| **Bonne initialisation du robot** | poppy = PoppyHumanoid(camera = ‘dummy’) | Aucune erreur, test validé,  on récupère de cette commande la liste des moteurs détecté par le software |
| --- | --- | --- |



On exécute le fichier python préparé en amont dénommé “check\_ports.py”, disponible en annexe, et qui nous renvoie la liste des moteurs détectés par la carte et dont voici la sortie:

/dev/ttyACM1

[31, 32, 33, 34, 35, 36, 37, 41, 42, 43, 44, 51, 52, 53, 54]

/dev/ttyACM0

[11, 12, 13, 14, 15, 21, 22, 23, 24, 25]

Voici les résultats du test :

| **ID - Moteur** | **Test à réaliser** | **Validation** |
| --- | --- | --- |
| **Jambes** | | |
| **R\_ANKLE\_Y 25** | Détection du moteur via | Test validé |
| **R\_KNEE\_Y 24** | Détection du moteur | Test validé |
| **R\_HIP\_X 21** | Détection du moteur | Test validé |
| **R\_HIP\_Y 23** | Détection du moteur | Test validé |
| **R\_HIP\_Z 22** | Détection du moteur | Test validé |
| **L\_ANKLE\_Y 15** | Détection du moteur | Test validé |
| **L\_KNEE\_Y 14** | Détection du moteur | Test validé |
| **L\_HIP\_X 11** | Détection du moteur | Test validé |
| **L\_HIP\_Y 13** | Détection du moteur | Test validé |
| **L\_HIP\_Z 12** | Détection du moteur | Test validé |
| **Tronc et tête** | | |
| **ABS\_Y 31** | Détection du moteur | Test validé |
| **ABS\_X 32** | Détection du moteur | Test validé |
| **ABS\_Z 33** | Détection du moteur | Test validé |
| **BUST\_X 35** | Détection du moteur | Test validé |
| **BUST\_Y 36** | Détection du moteur | Test validé |
| **HEAD\_Z 36** | Détection du moteur | Test validé |
| **HEAD\_Y 37** | Détection du moteur | Test validé |
| **Bras** | | |
| **R\_SHOULDER\_Y 51** | Détection du moteur | Test validé |
| **R\_SHOULDER\_X 52** | Détection du moteur | Test validé |
| **R\_ARM\_Z 53** | Détection du moteur | Test validé |
| **R\_ELBOW\_Y 54** | Détection du moteur | Test validé |
| **R\_XL-320 55** | Détection du moteur | Non validé |
| **R\_XL-320 56** | Détection du moteur | Non validé |
| **L\_SHOULDER\_Y 41** | Détection du moteur | Test validé |
| **L\_SHOULDER\_X 42** | Détection du moteur | Test validé |
| **L\_ARM\_Z 43** | Détection du moteur | Test validé |
| **L\_ELBOW\_Y 44** | Détection du moteur | Test validé |
| **L\_XL-320 45** | Détection du moteur | Non validé |
| **L\_XL-320 46** | Détection du moteur | Non validé |

Les résultats nous montrent que l’ensemble des moteurs principaux du robot POPPY sont détectés et sont en capacité de communiquer avec la carte Raspberry Pi. Mais les moteurs des poignets n’ont pas été détectés. Nous pensons que cela vient du fait que ces moteurs ne sont pas dans les versions originales de POPPY et que la librairie que nous avons ne les prend pas en compte. Ils ne sont pas cruciaux à ce stade du projet, mais il serait intéressant de trouver une version plus récente des librairies ou un moyen de prendre en compte ces moteurs.

## **Test fonctionnement système du bloquage et débloquage**

L’une des particularités de la programmation du robot POPPY est que l’on peut bloquer et débloquer les moteurs séparément à notre guise. Nous souhaitons donc tester cette fonctionnalité, primordiale à la stabilité du robot. Cela passe par de l’attribut .compliant . On met tous les moteurs m.compliant = False pour bloquer les moteurs, après on met m.compliant = True pour débloquer les moteurs, on observe physiquement si les moteurs sont bel et bien bloqués ou débloqués. Voici les résultat du test:

| **ID - Moteur** | **Compliant = false** | **Compliant = true** |
| --- | --- | --- |
| **Jambes** | | |
| **R\_ANKLE\_Y 25** | Immobile (test validé) | mobile (test validé) |
| **R\_KNEE\_Y 24** | Immobile (test validé) | mobile (test validé) |
| **R\_HIP\_X 21** | Immobile (test validé) | mobile (test validé) |
| **R\_HIP\_Y 23** | Immobile (test validé) | mobile (test validé) |
| **R\_HIP\_Z 22** | Immobile (test validé) | mobile (test validé) |
| **L\_ANKLE\_Y 15** | Immobile (test validé) | mobile (test validé) |
| **L\_KNEE\_Y 14** | Immobile (test validé) | mobile (test validé) |
| **L\_HIP\_X 11** | Immobile (test validé) | mobile (test validé) |
| **L\_HIP\_Y 13** | Immobile (test validé) | mobile (test validé) |
| **L\_HIP\_Z 12** | Immobile (test validé) | mobile (test validé) |
| **Tronc et tête** | | |
| **ABS\_Y 31** | Immobile (test validé) | mobile (test validé) |
| **ABS\_X 32** | Immobile (test validé) | mobile (test validé) |
| **ABS\_Z 33** | Immobile (test validé) | mobile (test validé) |
| **BUST\_X 35** | Immobile (test validé) | mobile (test validé) |
| **BUST\_Y 34** | Immobile (test validé) | mobile (test validé) |
| **HEAD\_Z 36** | Immobile (test validé) | mobile (test validé) |
| **HEAD\_Y 37** | Immobile (test validé) | mobile (test validé) |
| **Bras** | | |
| **R\_SHOULDER\_Y 51** | Immobile (test validé) | mobile (test validé) |
| **R\_SHOULDER\_X 52** | Immobile (test validé) | mobile (test validé) |
| **R\_ARM\_Z 53** | Immobile (test validé) | mobile (test validé) |
| **R\_ELBOW\_Y 54** | Immobile (test validé) | mobile (test validé) |
| **R\_XL-320 55** | test non validé | test non validé |
| **R\_XL-320 56** | test non validé | test non validé |
| **L\_SHOULDER\_Y 41** | Immobile (test validé) | mobile (test validé) |
| **L\_SHOULDER\_X 42** | Immobile (test validé) | mobile (test validé) |
| **L\_ARM\_Z 43** | Immobile (test validé) | mobile (test validé) |
| **L\_ELBOW\_Y 44** | Immobile (test validé) | mobile (test validé) |
| **L\_XL-320 45** | test non validé | test non validé |
| **L\_XL-320 46** | test non validé | test non validé |

Les moteurs répondent bien à la fonction de blocage/déblocage. Bien évidemment, les moteurs des poignets ne réagissent pas puisqu’ils ne sont pas pris en compte par la carte qui ne les détecte pas.

## **Test du système à l’état initial**

Nous souhaitons maintenant tester la mise en position initiale. En effet, chaque moteur possède une position de départ, qui correspond à un robot droit et debout. Afin de réaliser le test, nous repassons en mode moteurs bloqués puis nous utilisons la fonction m.goto\_position(0, 0.5, wait = True) qui amène le moteur à la position 0. On exécute pour cela le script python préparé en avance “test\_blocage.py”, disponible en annexe.

Après exécution, on observe physiquement la position initiale des moteurs.On a trouvé trois moteurs mal orienté et on trouve les solutions possible, on les arrange dans le tableau suivant:

| **ID - Moteur** | **Problème** | **Solution** |
| --- | --- | --- |
| **L\_HIP\_Z 12** | Mauvaise position à l'initialisation ==>  Mauvaise configuration du moteur | Ajouter un offset au moteur ⇒ Problème résolu |
| **HEAD\_Z 36** | Mauvaise position à l'initialisation ==>  Problème de montage | Refaire le montage ⇒ Problème résolu |
| **L\_ARM\_Z 43** | Mauvais étiquetage + mauvaise position à l'initialisation ==>  Problème de montage | Ajouter un offset au moteur ⇒ Problème résolu |

Nous avons finalement réglé tous les problèmes d’initialisation. Ces problèmes étaient importants à régler, car cela pourrait fausser le positionnement du moteur par la suite.

## **Test contrôle POPPY**

Une fois les tests d’initialisation réalisés, il nous faut également vérifier les écarts entre l’angle moteur demandé et l’angle réel mesuré, mais aussi voir si l’intervalle d’angle dans lequel chaque moteur peut évoluer correspond à l’intervalle connu. Pour cela:

* On mesure l’écart mesuré pour angle demandé à 0.
* On donne deux commandes:
  + m.goto\_position(180, 0.5, wait = True)
  + m.goto\_position(-180, 0.5, wait = True)

pour voir les limites de mouvements des moteurs.

On exécute donc le script python “test\_ecart\_limite.py” disponible en annexe et on inscrit les résultats dans le tableau suivant:

| **ID - Moteur** | **Ecart Initialisation (à 0)** | **Motor range** | **Données officielles** |
| --- | --- | --- | --- |
| **Jambes** | | | |
| **R\_ANKLE\_Y 25** | 0.5 | [-44.1, 44.6] | offset : 0, [-45,45] |
| **R\_KNEE\_Y 24** | 0.4 | [-2.6, 132.7] | offset : 0, [-134,3.5] |
| **R\_HIP\_X 21** | 0.8 | [-28, 30.4] | offset : 0, [-28,30] |
| **R\_HIP\_Y 23** | -0.3 | [-103.2, 84.5] | offset : 0, [-85,105] |
| **R\_HIP\_Z 22** | 0.9 | [-23.7, 88.5] | offset : 0, [-90,25] |
| **L\_ANKLE\_Y 15** | -0.2 | [-44.4, 43.8] | offset : 0, [-45,45] |
| **L\_KNEE\_Y 14** | 0.3 | [-3, 132.6] | offset : 0, [-3.5,134] |
| **L\_HIP\_X 11** | -2.2 | [-29.9, 27] | offset : 0, [-30,28.5] |
| **L\_HIP\_Y 13** | 0.3 | [-104.2, 81.4] | offset : 2, [-104,84] |
| **L\_HIP\_Z 12** | -0.9 | [-89.4, 21.3] | offset : 0, [-25,90] |
| **Tronc et tête** | | | |
| **ABS\_Y 31** | -1 | [-12.3, 48.4] | offset : 0, [-50,12] |
| **ABS\_X 32** | -0.8 | [-44.6, 44.1] | offset : 0, [-45,45] |
| **ABS\_Z 33** | -0.4 | [-89.7, 90.3] | offset : 0, [-90,90] |
| **BUST\_X 35** | -0.2 | [-37.6, 36.8] | offset : 0, [-40,40] |
| **BUST\_Y 34** | 6 | [-21.9, 65.3] | offset : 0, [-67,27] |
| **HEAD\_Z 36** | -0.44 | [-88.1, 89.3] | offset : 0, [-90,90] |
| **HEAD\_Y 37** | 0.22 | [-25.1, 25.6] | offset : 20, [-45,6] |
| **Bras** | | | |
| **R\_SHOULDER\_Y 51** | -1.4 | [-179.6, 63.5] | offset : 90, [-155,120] |
| **R\_SHOULDER\_X 52** | -0.85 | [-179.1, 17.7] | offset : 90, [-110,105] |
| **R\_ARM\_Z 53** | 0 | [-104.7, 104.8] | offset : 0, [-105,105] |
| **R\_ELBOW\_Y 54** | 0.2 | [-146.3, 0.8] | offset : 0, [-1,148] |
| **R\_XL-320 55** | - | - | - |
| **R\_XL-320 56** | - | - | - |
| **L\_SHOULDER\_Y 41** | -1.1 | [-178.8, 62.9] | offset : 90, [-120,155] |
| **L\_SHOULDER\_X 42** | 0.1 | [-10.5, 179.1] | offset : -90, [-105,110] |
| **L\_ARM\_Z 43** | -0.1 | [-104.5, 104.9] | offset : 0, [-105,105] |
| **L\_ELBOW\_Y 44** | -0.5 | [-145.1, 0.3] | offset : 0, [-148,1] |
| **L\_XL-320 45** | - | - | - |
| **L\_XL-320 46** | - | - | - |

L’erreur entre la valeur demandée et la valeur mesurée existe pour deux raison :

* une erreur lié à la vitesse du moteur et au temps de réaction du système
* une erreur lié à la qualité de la mesure de l’angle

Nous considérons donc que l’erreur est acceptable si elle est en dessous de 5°, au-delà elle devient visible à l'œil nu. On observe que tous les moteurs respectent ce critère pour une initialisation à 0, sauf le moteur BUST\_Y\_34. Nous n’avons pas eu le temps de terminer d’où venait l’erreur, physique ou logiciel, mais visuellement nous n’avons rien remarqué de choquant. Concernant les intervalles d’amplitude des moteurs, on observe un phénomène étrange où les valeurs d’intervalles sont parfois inversées. Cela signifie peut-être que le moteur ne tourne pas dans le bon sens. Nous n’avons pas pu confirmer ce détail, il se fera sûrement ressentir en essayant directement sur un script d’exercices de kinésithérapie.

## **Test enregistrement mouvement**

Une des fonctionnalités de la librairie POPPY est que l’on peut effectuer l’enregistrement du mouvement d’un ou plusieurs moteurs durant un temps donné, et que l’on peut rejouer ce mouvement. Pour vérifier cela nous exécutons le code “*test\_record*.*py*”, disponible en annexe, qui permet d’effectuer l’enregistrement du mouvement des moteurs du bras gauche et de le rejouer

| **Test** | **Description test** | **Résultat** |
| --- | --- | --- |
| **Test enregistrement mouvement** | Exécution du script *test\_record.py* | Le mouvement rejoué est similaire au mouvement effectué |

Malheureusement, nous n’avons pas pu effectuer de tests sur chacun des moteurs du robot, mais nous nous sommes assurés par la présente du bon fonctionnement d’un point de vue général de cette fonctionnalité POPPY.

L’ensemble des tests logiciels réalisés permet de confirmer le bon fonctionnement du robot POPPY. Ce dernier est donc capable d’effectuer les tâches qui lui sont demandées. Néanmoins, il reste encore à confirmer son fonctionnement pour les scripts d’exercices de kinésithérapie déjà fait, en s’assurant notamment de la rotation dans le bon sens des moteurs.

# 

# **Test mécanique**

Les tests mécaniques sont l’ensemble des tests qui permettent de valider les éléments de conception de l’équipe mécanique. Cela comprend une validation matériel de l’assemblage du robot POPPY mais aussi du matériel de système de changement de position que nous avons commandé.

## **Test matériel d’assemblage du robot POPPY**

Afin de s’assurer que le montage du robot s’est bien déroulé, nous avons effectué, avant les tests logiciels décrits plus haut, un dernier test matériel sur le robot. Ce test consiste simplement, pour chaque moteur, à vérifier que celui tourne bien selon l’axe donné par sa nomenclature, et ne soit pas bloqué par une autre pièce ou bien la longueur d’un câble. Un second test est de vérifier que les pièces sont bien fixées entre elles et qu’il n’y a pas de jeu trop important.

Ce test qui peut paraître anodin est en réalité très important. Nous avons pu ainsi, trouver des anomalies dans l’assemblage qui bloquait certains moteurs. Voici finalement le résultat du test.

| **Test** | **Description test** | **Résultat** |
| --- | --- | --- |
| **Test matériel de chacun des moteurs** | On fait tourner à la main chaque moteur pour vérifier qu’il ne soit pas bloqué. | Tous les moteurs sont libérés et aucune pièce ne bloque leur rotation dans le champ du possible |
| **Test fixation pièces** | On applique de légères pressions et mouvements sur chaque fixation pour vérifier leur maintien et le jeu existant. | Toutes les pièces sont bien fixées entre elles, il n’y a pas de jeu entre elles. |

## **Test fonctionnement de l’alimentation**

Nous avions fait la commande d’une alimentation adaptée pour le vérin mais la commande n’a jamais été passée. M. Taruffi nous a alors fourni une autre alimentation ayant les mêmes caractéristiques. Nous souhaitons vérifier son fonctionnement. Pour cela nous le branchons à du matériel électronique adapté, et on observe si le matériel s’allume.

| **Test** | **Description test** | **Résultat** |
| --- | --- | --- |
| **Test alimentation vérin** | On branche l’alimentation à du matériel électronique adapté | Aucun courant ne passe |

L’alimentation fournie par M.Taruffi ne fonctionne pas, nous avons utilisé la seconde alimentation de rechange du robot POPPY pour faire les tests qui suivent.

## **Test connectique du vérin électrique**

On s’intéresse maintenant au vérin électrique. Ce dernier ne possédait pas une connectique que nous pouvions utiliser. Pour cela nous avons procédé à une phase de soudage pour transformer cette connectique inhabituelle en deux fils électriques distincts. Il nous fallait alors vérifier que le vérin électrique fonctionne et que la nouvelle connectique suite à la soudure soit également fonctionnelle et sécurisée. Pour pouvoir effectuer ce test, il nous a suffit de brancher les fils à un adaptateur LED Terminal, que l’on peut alors brancher directement sur l’alimentation, dans un sens ou dans l’autre pour soit faire sortir, soit faire rentrer le vérin.

| **Test** | **Description du test** | **Résultat** |
| --- | --- | --- |
| **Connectique sens 1**  **(vérin rentrant)** | Brancher le vérin à l’alimentation dans le sens 1, c’est-à-dire interchanger le branchement des fils par rapport au sens 2 | Le vérin rentre. Quand il arrive en butée, le moteur s’arrête. Test validé |
| **Connectique sens 2**  **(vérin sortant)** | Brancher le vérin à l’alimentation dans le sens 2, c’est-à-dire interchanger le branchement des fils par rapport au sens 1 | Le vérin sort. Quand il arrive en butée, le moteur s’arrête. Test validé |
| **Sécurité connectique** | Vérifier la sécurité de la nouvelle connectique, pour le vérin et les utilisateurs | La connectique est entourée de gaine protectrice. Aucune surchauffe n’a été aperçu. |

Les tests permettent de confirmer le fonctionnement du vérin mais aussi de la nouvelle connectique réalisée cette année. Nous en avons profité pour réaliser une mesure de vitesse du vérin sans charge, par moyenne sur plusieurs mesures. Nous obtenons une vitesse de 16.2cm/s.

## **Test fonctionnement de l’optocoupleur**

En plus du vérin et de son alimentation, nous avons également passé commande d’un optocoupleur, équipement permettant d’agir comme un interrupteur. Le test que nous souhaitions réaliser aurait été de brancher l’optocoupleur à une LED et une petite alimentation, et de vérifier que lorsqu’un courant est appliqué en entrée de l’optocoupleur, la LED s’allume. L’atelier de l’U2IS contient le matériel nécessaire.

| **Test** | **Description test** | **Résultat** |
| --- | --- | --- |
| **Test optocoupleur** | On branche l’optocoupleur en sortie à une LED et une alimentation, et on vérifie qu’elle s’allume en appliquant une tension en entrée de l’optocoupleur. | Test non réalisé |

Nous n’avons malheureusement pas eu le temps de réaliser ce test à cause du retard d’assemblage.

# 

# **Conclusion**

La série de tests que nous avons réalisés nous a permis de vérifier le fonctionnement du robot POPPY et de nous assurer que le robot que nous fournissons à la prochaine équipe soit fonctionnel, tout comme le matériel du système de changement de position. Mais cela ne veut pas dire que toutes les fonctionnalités sont entièrement validées. Il reste certains aspects, comme le contrôle des moteurs de poignets, qui n’ont pas pu être validés, qui ne sont pas cruciaux à l’instant t mais pourront le devenir plus tard.

# 

# **Annexe**

## **check\_port.py**

import pypot.dynamixel

ports = pypot.dynamixel.get\_available\_ports()

if not ports:

raise IOError('no port found')

print(ports[0])

dxl\_io1 = pypot.dynamixel.DxlIO(ports[0],baudrate=1000000,use\_sync\_read=False)#upper

print(dxl\_io1.scan(range(60)))

print (ports[1])

dxl\_io2 = pypot.dynamixel.DxlIO(ports[1],baudrate=1000000,use\_sync\_read=False)#lower

print(dxl\_io2.scan(range(60)))

## **test\_blocage.py**

from poppy\_humanoid import PoppyHumanoid

import time

fichier = "init\_motor.txt"

poppy = PoppyHumanoid(camera = 'dummy')

def pause():

programPause = input("Press the <ENTER> key to continue...")

return

with open(fichier,"w+") as f:

f.write("-------------------------Initialisation---------------------------\n")

for m in poppy.motors:

m.compliant = False

m.goto\_position(0, 0.01, wait=True)

print(m.name,", initialisation: ", m.present\_position)

pause()

# f.write("%s, initialisation: %f\n"%(m.name,m.present\_position))

print("Fin de l'initialisation")

f.close()

for m in poppy.motors:

m.compliant = True

## **test\_ecart\_limite.py**

from poppy\_humanoid import PoppyHumanoid

import time

fichier = "test\_motor.txt"

poppy = PoppyHumanoid(camera = 'dummy')

def pause():

programPause = input("Press the <ENTER> key to continue...")

return

with open(fichier,"w+") as f:

f.write("\n-------------------------Test---------------------------\n")

for i in range (15,len(poppy.motors)):

m = poppy.motors[i]

print(m.name)

m.compliant = False

pause()

m.goto\_position(0, 0.5, wait=True)

pause()

print(m.name,", initialisation: ", m.present\_position)

f.write("%s, initialisation: %f\n"%(m.name,m.present\_position))

pause()

m.goto\_position(180, 0.5, wait=True)

pause()

print(m.name,", Angle max: ",m.present\_position)

f.write("%s, Angle max: %f\n"%(m.name,m.present\_position))

pause()

m.goto\_position(-180, 0.5, wait=True)

pause()

print(m.name,", Angle min: ",m.present\_position)

f.write("%s, Angle min: %f\n"%(m.name,m.present\_position))

pause()

m.goto\_position(0, 0.5, wait=True)

pause()

print(m.name,", retour à l'état initial: ",m.present\_position)

f.write("%s, retour à l'état initial: %f\n"%(m.name,m.present\_position))

m.compliant = True

f.close()

## **test\_record.py**

from pypot.primitive.move import MoveRecorder

from poppy\_humanoid import PoppyHumanoid

from pypot.primitive.move import MovePlayer

from copy import deepcopy

poppy = PoppyHumanoid(camera = "dummy")

for m in poppy.motors:

m.compliant = False

m.goto\_position(0, 0.01, wait=True)

print(m.name,", initialisation: ", m.present\_position)

# f.write("%s, initialisation: %f\n"%(m.name,m.present\_position))

motors = [poppy.l\_shoulder\_y, poppy.l\_shoulder\_x, poppy.l\_arm\_z, poppy.l\_elbow\_y]

for m in motors:

m.compliant = True

record = MoveRecorder(poppy, 50, motors)

import time

def pause():

programPause = input("Press the <ENTER> key to continue...")

return

# Give you time to get ready

print('Get ready to record a move...')

pause()

# Start the record

record.start()

print('Now recording !')

# Wait for 10s so you can record what you want

time.sleep(30)

# Stop the record

print('The record is over!')

record.stop()

pause()

print("write in a document.")

my\_recorded\_move = deepcopy(record.move)

with open('my-first-demo.move', 'w') as f:

my\_recorded\_move.save(f)

print(len(my\_recorded\_move.positions()))

print(my\_recorded\_move.positions())

pause()

print("Robot movement.")

player = MovePlayer(poppy, my\_recorded\_move)

player.start()

player.wait\_to\_stop()