



RAPPORT PROJET EA Systèmes Avancés

Qualification fonctionnelle d'une plateforme robotique collaborative

Étudiants	- Alae Zerrouq - Adam Aoubiza
Encadrant	- Vincent Idasiak

Plan du projet :

I. Introduction :

L'introduction du rapport présente l'initiation du robot et met l'accent sur les mesures de sécurité essentielles à prendre en compte lors de son utilisation, ainsi que sur les premiers tests effectués pour vérifier son fonctionnement de base et sa sécurité.

II. Initiation du robot, mesures de sécurité, simulation sur RoboDK et tests de programmes de base

Cette section couvre plusieurs aspects, allant de l'installation initiale du robot aux mesures de sécurité à respecter lors de son utilisation. Elle inclut également la simulation des mouvements sur RoboDK avant leur exécution sur le robot réel, ainsi que les tests effectués avec des programmes de base pour vérifier le bon fonctionnement du robot.

III. Traduction entre les bibliothèques RoboDK et PyNiryo

L'algorithme de traduction entre RoboDK et PyNiryo est expliqué dans cette partie. Cette traduction facilite la simulation des mouvements sur RoboDK avant leur exécution sur le robot réel, permettant ainsi de valider les programmes avant leur déploiement.

IV. Programmation Pick & Place du robot

La programmation du robot pour réaliser des opérations de pick & place (avec un programme pyramide) est présentée en détail dans cette partie. Les étapes de développement du programme, y compris le placement précis des objets, sont discutées.

V. Les objectifs pour le prochain Semestre

Les objectifs à atteindre pour le semestre suivant sont définis dans cette partie. Ils incluent l'intégration de nouvelles fonctionnalités, telles que la reconnaissance de couleur et les commandes vocales, ainsi que l'amélioration des programmes existants.

Cette section met également en évidence comment créer un workspace, et d'autres programmes développés mais non encore testés sur le robot, présentant ainsi les futures perspectives de développement.

VI. Conclusion

La conclusion résume les principales réalisations du projet et met en avant les apprentissages et les perspectives futures dans le domaine de la robotique collaborative.

I. Introduction :

Ce rapport présente un guide détaillé des étapes suivies dans le cadre du projet de qualification fonctionnelle d'une plateforme robotique collaborative, destiné à servir de référence pour les novices sur un nouvel ordinateur.

Encadrés par Vincent IDASIAK, nous avons parcouru un chemin méthodique, allant de la familiarisation avec les outils et le matériel à la programmation avancée en Python, en passant par l'utilisation de logiciels spécifiques tels que NiryoStudio et RoboDK.

Nous avons commencé par des manipulations élémentaires sur le robot, en testant des programmes de base pour comprendre ses capacités et son fonctionnement. Par la suite, nous avons approfondi notre compréhension en nous familiarisant pleinement avec le robot, le logiciel NiryoStudio et les environnements de développement Python, tels que PyCharm et Pyzo, sur lesquels nous avons programmé en utilisant la bibliothèque PyNiryo.

Une étape cruciale de notre processus a été l'utilisation de RoboDK pour simuler les mouvements du robot avant de les exécuter. Cette simulation nous a permis d'assurer la sécurité des mouvements et de prévenir les éventuelles collisions. Pour ce faire, nous avons mis en place un algorithme de traduction entre RoboDK et NiryoStudio, nous permettant de valider nos programmes de manière virtuelle avant de les implémenter sur le robot réel.

Dans ce rapport, nous détaillons chaque étape de notre parcours, en fournissant des instructions claires et concises pour chaque action entreprise. De la configuration initiale des logiciels à l'exécution des programmes sur le robot, en passant par la simulation sur RoboDK, chaque étape est soigneusement documentée pour garantir une reproductibilité maximale.

Enfin, nous fixons les objectifs pour le prochain semestre, qui consistent à utiliser la caméra du robot pour développer un programme pick & place capable de trier les objets en fonction de leur couleur. Ce programme servira de base pour étendre les fonctionnalités du robot à la détection de bouchons sur les flacons d'une chaîne industrielle. De plus, nous envisageons d'intégrer la capacité du robot à suivre le mouvement de la main, ainsi que la possibilité pour le robot de donner des commandes vocales en utilisant la bibliothèque GTTS (Google Text-to-Speech) sur Python. Ces ajouts permettront d'améliorer

l'interactivité et la convivialité du système, offrant ainsi de nouvelles possibilités d'interaction entre l'homme et la machine.

En conclusion, ce rapport offre un guide complet pour quiconque souhaite suivre nos pas dans la qualification fonctionnelle d'une plateforme robotique collaborative, tout en posant les bases pour les futures explorations et développements dans le domaine de la robotique et de l'automatisation industrielle.

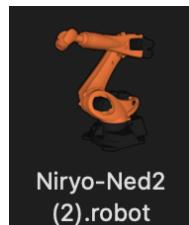
II. Initiation du robot, mesures de sécurité, simulation sur RoboDK et test de programmes de base :

Avant de commencer à utiliser le robot, il est indispensable de lire la documentation : <https://static.generation-robots.com/media/manuel-utilisation-ned2-niryo-fr.pdf>, notamment les consignes de sécurité. Ces consignes peuvent être résumés dans la fiche de sécurité qui suit :

Risques Mineurs	Risques Majeurs
	
 Risques de piégeage ou de coinement : - Les doigts ou les vêtements peuvent être pris dans les parties mobiles du robot lorsqu'il est en marche.	 Risques de blessures graves : - Risque de blessures importantes en cas de contact direct avec les parties mécaniques en mouvement du robot.
 Risque de choc électrique mineur : - Possibilité de choc électrique lors de l'interaction avec les composants électroniques du robot.	 Risque de court-circuit majeur : - Possibilité de court-circuit électrique pouvant causer des dommages importants au robot ou à son environnement.
 Risques liés aux outils de manipulation : - Risque de pincement ou de blessures mineures lors de l'utilisation d'outils attachés au robot pour la manipulation d'objets.	 Risques de projection d'objets : - Possibilité que des objets manipulés par le robot soient projetés à grande vitesse, représentant un risque pour les personnes à proximité.
Risques Modérés	Risques Critiques
 Risques de collision ou de chute : - Le robot peut se déplacer de manière imprévisible et entraîner des collisions ou des chutes d'objets environnants.	 Risques pour la sécurité des données : - Possibilité de compromission des données lors de l'utilisation ou de l'interaction avec les composants informatiques du robot.
 Risque de surchauffe : - Possibilité que certaines parties du robot deviennent chaudes pendant un fonctionnement prolongé.	 Risque d'incendie ou d'explosion : - Risque élevé de défaillance majeure pouvant entraîner un incendie ou une explosion.
 Risque de perturbation des systèmes environnants : - Risque d'interférence avec les systèmes électroniques ou informatiques environnants lors du fonctionnement du robot.	

Maintenant que vous connaissez les mesures de sécurité, voici les étapes à suivre pour pouvoir utiliser le robot :

1. Télécharger NiryoStudio : <https://niryo.com/fr/download-center/>
2. Télécharger Pycharm : <https://www.jetbrains.com/pycharm/download/>
3. Télécharger RoboDK : <https://robodk.com/fr/download>
4. Aller sur Pycharm, ouvrir un nouveau projet, aller sur le terminal et taper la commande ‘pip3 install numpy’ et la commande ‘pip3 install pyniryo’
Vous pouvez désormais exécuter des programmes python
5. Aller sur RoboDK, puis aller sur la bibliothèque (le logo qui suit : ) puis écrire ‘Ned2’ sur ‘Search Library’, appuyer sur ‘Download’, puis



ouvrir le robot avec un double clique sur le fichier téléchargé

. Vous pouvez désormais effectuer des simulations sur le robot directement à partir de RoboDK.

6. Ouvrir NiryoStudio

Pour acquérir une familiarité avec le fonctionnement du robot, une série de manipulations de base peut être entreprise sur NiryoStudio, suivant les étapes suivantes :

1. Brancher le robot : Assurez-vous de connecter correctement la prise du robot à une source d'alimentation électrique.
2. Vérifier l'arrêt d'urgence : Tournez le bouton d'arrêt d'urgence dans le sens des aiguilles d'une montre pour le désactiver, au cas où il aurait été enclenché précédemment.
3. Allumer le robot : Le bouton ON/OFF se situe à l'arrière du robot. Appuyez dessus pour l'allumer.
4. Connexion au réseau du robot : Assurez-vous que le robot est connecté au réseau local.
5. Connexion à NiryoStudio : Lancez l'application NiryoStudio sur votre ordinateur et connectez-vous au robot en appuyant sur le bouton "Connect Robot".
6. Calibrage : Effectuez un calibrage initial pour garantir une précision optimale des mouvements.

À ce stade, vous êtes prêt à tester les mouvements de base du robot en accédant à la fonction "Direct Move" dans NiryoStudio. Cette étape vous permettra de vous familiariser avec les commandes de base et les capacités de déplacement du robot.

Important : Lors de l'utilisation du robot, il est impératif qu'une personne reste près du bouton d'arrêt d'urgence, et que la voie vers celui-ci soit dégagée afin que celui-ci puisse être actionné en cas d'urgence. Le bouton d'arrêt d'urgence devra **UNIQUEMENT** être enclenché en cas d'urgence, pour éteindre le robot,

merci d'utiliser le bouton ON/OFF. Il faudra également dégager de l'entourage tout objet qui risque de gêner le mouvement du robot afin d'éviter tout accident. Quand vous finissez, vous pouvez éteindre le robot, une fois que celui-ci est éteint, vous devez enclencher le bouton d'arrêt d'urgence.

Afin de vérifier que le robot fonctionne correctement et que l'installation de l'univers python s'est bien déroulée, vous pouvez tester quelques programmes de base, en suivant ces étapes :

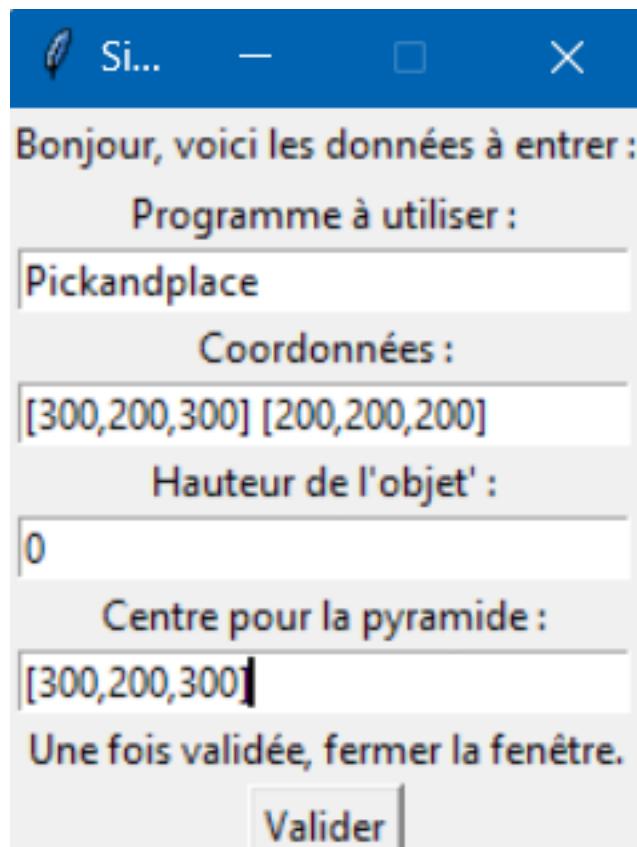
1. Brancher le robot : Assurez-vous de connecter correctement la prise du robot à une source d'alimentation électrique.
2. Vérifier l'arrêt d'urgence : Tournez le bouton d'arrêt d'urgence dans le sens des aiguilles d'une montre pour le désactiver, au cas où il aurait été enclenché précédemment.
3. Allumer le robot : Le bouton ON/OFF se situe à l'arrière du robot. Appuyez dessus pour l'allumer.
4. Connexion au réseau du robot : Assurez-vous que le robot est connecté au réseau local.
5. Ouvrir PyCharm
6. Ouvrir un nouveau script Python sur Pycharm
7. Entrez et exécutez le programme ci-dessous :

```
from pyniryo import *
robot = NiryoRobot("10.10.10.10")
robot.calibrate_auto()
robot.move_joints(0.2, -0.3, 0.1, 0.0, 0.5, -0.8)
robot.close_connection()
```

III. Traduction entre les bibliothèques RoboDK et Pyniryo:

On exécute le programme qui suit, qui nous permet d'effectuer une simulation des programmes sur RoboDK avant d'exécuter le mouvement sur le vrai Robot (à travers Pycharm / Pyzo) : (*Voir scripts ‘programme traduction’, ‘programme execution’ et ‘programme simulation’ dans l’annexe SVP*)

Après l'exécution, fenêtre suivante s'ouvre:



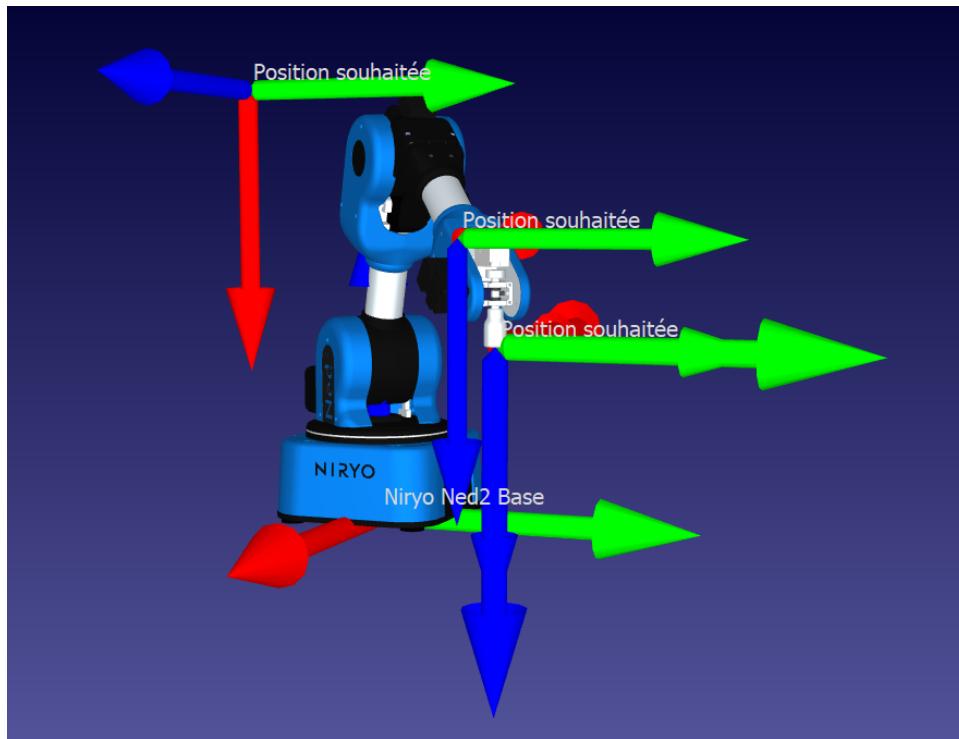
Sur celle-ci, il faudra rentrer le nom du programme qu'on souhaite exécuter et les coordonnées des objets à manipuler.

1. Dans un premier temps, il vous faudra exécuter Programmetraduction, saisir les informations demandées, cliquer sur le bouton valider, fermer la petite fenêtre qui était alors apparue.
2. Ensuite, il faut exécuter le programme intitulé Programmesimulation qui ouvrira alors Robodk, affichant le Ned2 qui effectuera les mouvements demandés
3. Si vous souhaitez effectuer les mouvements avec d'autres coordonnées, il vous faut impérativement ouvrir le dossier où se trouve les scripts,

trouver le fichier texte “Données pour Nyrio” avant de le supprimer. Ensuite vous pourrez répéter une simulation avec des données différentes.

4. Une fois satisfait de la simulation, vous pourrez exécuter le script Programmeexecution avec en parallèle l’application Nyriostudio ouvert et la connexion par wifi avec le robot effectué. Une fois cela fait, le robot répétera les mouvements vu précédemment.

On obtient ce qui suit sur RoboDK :



Maintenant que la simulation est effectuée, on peut directement exécuter le programme sur le robot en toute sécurité.

IV. Programmation Pick & Place avec Pyramide du robot :

```
from pyniryo import *
robot = NiryoRobot("10.10.10.10")
robot.set_arm_max_velocity(100)
Home = robot.get_pose()

def pyramideniryo(x,y,z,o):
    l = [x,y,z]
    lumpy = []
    robot.update_tool()
    c = 0
    for i in l:
        Robot.release_with_tool()
        Robot.move_pose(i)
        Robot.grasp_with_tool()
        Robot.move_pose(Home)
        new_place_pose = o.copy_with_offsets(z_offset=0.01 * c)
        lumpy.append(new_place_pose)
        Robot.move_pose(new_place_pose)
        Robot.release_with_tool()
        Robot.move_pose(Home)
        c = c + 1
    for i in lumpy:
        Robot.release_with_tool()
        Robot.move_pose(lumpy[c-1])
        Robot.grasp_with_tool()
        Robot.move_pose(Home)
        Robot.move_pose(l[c-1])
        Robot.release_with_tool()
        Robot.move_pose(Home)
        c = c - 1
    x = PoseObject(x = 0.240, y = 0.090, z = 0.120, roll = 0,
pitch = 1.57, yaw = 0)
    y = PoseObject(x = 0.203, y = 0.200, z = 0.120, roll = 0,
pitch = 1.57, yaw = 0)
    z = PoseObject(x = 0.242, y = -0.150, z = 0.120, roll = 0,
pitch = 1.57, yaw = 0)
    o = PoseObject(x = 0.235, y = 0.0, z = 0.120, roll = 0,
pitch = 1.57, yaw = 0)

pyramide(x,y,z,o)

robot.close_connection()
```

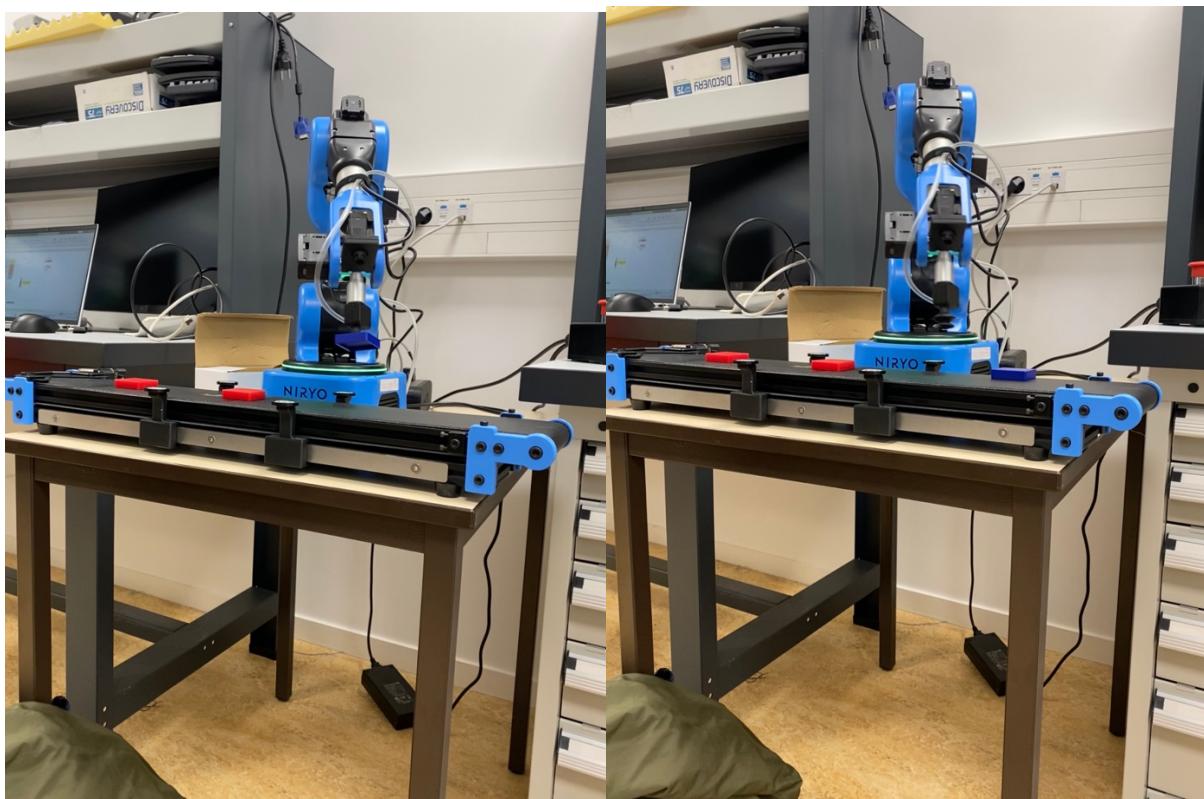
Le programme commence par construire la pyramide d'objets en les plaçant à des positions spécifiques. Pour ce faire, il utilise une série de mouvements du robot Niryo pour saisir chaque objet, le déplacer vers sa position désignée, puis le relâcher. En répétant ce processus pour chaque objet, la pyramide est progressivement construite jusqu'à ce qu'elle soit complète.

Une fois la pyramide construite, le programme procède à sa "destruction" en inversant le processus. Il commence par saisir chaque objet un par un, le déplacer vers une position de base préalablement définie, puis le relâcher. En répétant cette opération pour chaque objet, la pyramide est démontée, ramenant chaque objet à sa position initiale.

Ainsi, le programme démontre la capacité du robot à manipuler des objets de manière précise pour réaliser des tâches de construction et de déconstruction, mettant en évidence les fonctionnalités pratiques de la programmation robotique dans le contexte de la manipulation d'objets complexes pour des scénarios d'assemblage et de désassemblage.

Les étapes sont comme suit :





Ces photos montrent en détail les différentes étapes du programme. On voit sur la première photo les objets à leur position initiale, puis sur la deuxième photo les objets entassés en forme de pyramide, sur la troisième photo le robot qui remet les objets à leur position, et enfin sur la quatrième photo les objets de retour à leurs positions initiales.

V. Les objectifs pour le prochain Semestre :

Objectifs pour le S8 :

- Utilisation de la caméra pour la détection des objets : Nous prévoyons d'exploiter la caméra du robot pour détecter les objets selon leur couleur. Cela permettra de développer des programmes de manipulation plus avancés, en permettant au robot de reconnaître et de prendre en charge différents types d'objets en fonction de leur couleur. Cette fonctionnalité ouvrira de nouvelles possibilités d'automatisation et de manipulation flexible des objets dans divers contextes industriels et domestiques.
- Développement d'un programme de détection et de manipulation des bouchons : L'objectif final est de parvenir à la détection et à la manipulation des bouchons sur les flacons à l'aide de la caméra du robot. Ce programme représente un défi plus avancé et nécessitera une compréhension approfondie de la vision par ordinateur et de la manipulation d'objets. Nous prévoyons de développer des algorithmes de détection des bouchons et des stratégies de manipulation pour permettre au robot d'inspecter les flacons, de détecter les bouchons manquants et de les placer correctement, contribuant ainsi à améliorer l'efficacité et la qualité des processus industriels impliquant des flacons et des bouteilles
- Implémentation du text-to-speech pour les alertes : L'objectif principal est de programmer le robot pour qu'il émette des messages vocaux en cas d'erreur dans le code ou de conditions anormales pendant l'exécution. Cette fonctionnalité améliorera l'interactivité du robot en fournissant des retours vocaux clairs et compréhensibles à l'utilisateur, ce qui facilitera la résolution des problèmes et contribuera à une utilisation plus intuitive du robot, par exemple avec un script éventuellement similaire à celui-ci :

```
- from pyniryo import *
from gtts import gTTS

robot = NiryoRobot("10.10.10.10")

robot.calibrate_auto()

robot.move_joints(0.2, -0.3, 0.1, 0.0, 0.5, -0.8)
tts = gTTS('hello')
tts.save('hello.mp3')
```

```

robot.led_ring_flashing([255,0,0], 10, 20)
robot.say("Les coordonnées de x sont au-delà du domaine autorisé.",
Language.FRENCH)
robot.close_connection()

from pyniryo import *

robot = NiryoRobot("10.10.10.10")

robot.calibrate_auto()
robot.update_tool()
robot.clear_collision_detected()
robot.release_with_tool()
robot.move_pose(0.2, 0, 0.25, 0.0, 1.57, 0.0)
robot.grasp_with_tool()
robot.clear_collision_detected()
robot.move_pose(0.2, 0.1, 0.25, 0.0, 1.57, 0.0)
robot.release_with_tool()

robot.close_connection()

```

Ou encore à celui-ci :

```

from pyniryo import *
from gtts import gTTS
import os

robot = NiryoOne()

if robot.get_pose() == [0, 0, 0, 0, 0, 0]:
    texte = "Attention, le robot est dans une position inattendue. Veuillez vérifier son état."

    tts = gTTS(text=texte, lang='fr')
    tts.save("parole.mp3")

    os.system("mpg321 parole.mp3")

robot.close_connection()

```

Ou finalement celui-ci :

```

from pyniryo import *
from gtts import gTTS
import os

robot = NiryoOne()

if robot.get_pose() == [0, 0, 0, 0, 0, 0]:
    texte = "Attention, le robot est dans une position inattendue. Veuillez vérifier son état."

    tts = gTTS(text=texte, lang='fr')
    tts.save("parole.mp3")

    os.system("mpg321 parole.mp3")

robot.close_connection()

robot.calibrate_auto()

```

```
robot.update_tool()
robot.clear_collision_detected()
robot.release_with_tool()
robot.move_pose(0.2, 0, 0.25, 0.0, 1.57, 0.0)
robot.grasp_with_tool()
robot.clear_collision_detected()
robot.move_pose(0.2, 0.1, 0.25, 0.0, 1.57, 0.0)
robot.release_with_tool()

robot.close_connection()
```

Étapes pour enregistrer un workspace sur le robot :

Nous détaillons ici comment enregistrer un workspace sur NiryoStudio, ce qui nous sera utile dans la suite de notre projet :

1. Ouvrir NiryoStudio
2. Se connecter au robot
3. Aller sur ‘Library’ puis ‘Workspaces’
4. ‘Create new workspace’
5. Retirer la ventouse et mettre un pointeur sur le bras du robot
6. Appuyer sur ‘Freemotion’ sur le bras du robot, déplacer le bras du robot et fixer le pointeur sur les 4 positions du workspace désiré, tout en appuyant sur le bouton ‘save’ situé sur le bras du robot, à chacune des 4 position

VI. Conclusion :

En conclusion, ce rapport a détaillé de manière exhaustive le processus de qualification fonctionnelle d'une plateforme robotique collaborative, en mettant en lumière les étapes clés de l'initiation du robot, de la programmation pick & place, de la traduction entre les bibliothèques RoboDK et PyNiryo, ainsi que des objectifs pour le prochain semestre.

L'initiation du robot a été abordée avec une attention particulière aux mesures de sécurité, garantissant ainsi une utilisation sûre et efficace du robot. Les simulations sur RoboDK ont permis de valider les programmes avant leur exécution sur le robot réel, offrant un moyen précieux de détecter et de prévenir les éventuelles collisions ou erreurs de mouvement.

La programmation pick & place, illustrée par la construction et la déconstruction d'une pyramide d'objets, a mis en évidence les capacités avancées du robot à manipuler des objets avec précision et fiabilité, ouvrant la voie à une utilisation polyvalente dans divers contextes industriels.

La traduction entre les bibliothèques RoboDK et PyNiryo a facilité le processus de développement en permettant la simulation des mouvements sur RoboDK avant leur exécution sur le robot réel, contribuant ainsi à une optimisation du temps et des ressources.

Enfin, les objectifs ambitieux pour le prochain semestre témoignent de notre engagement continu à explorer de nouvelles fonctionnalités et à repousser les limites de la robotique collaborative. De la détection des objets à la manipulation des bouchons, en passant par l'implémentation de commandes vocales, notre travail futur promet d'apporter des avancées significatives sur l'objectif de notre projet de pouvoir travailler sur une chaîne industrielle de flaconnage.

En somme, ce projet a été une expérience enrichissante, nous permettant d'acquérir des compétences techniques avancées tout en explorant les possibilités infinies offertes par la robotique collaborative. Nous espérons que ce rapport servira de guide précieux pour les futures explorations dans ce domaine en constante évolution.

VII. Annexes :

- Programme Exécution :

```
### Programme Execution

##Importations des bibliothèques nécessaires

##Connexion au robot Niryo, calibration automatique et création de la pose
Home
Robot = NiryoRobot("10.10.10.10")
Robot.calibrate_auto()
Robot.set_arm_max_velocity(100)
Home = Robot.get_pose()

##Fonction auxiliaire
def pickandplaceniryo(X,Y):
    x1 = X[0]
    y1 = X[1]
    z1 = X[2]
    x2 = Y[0]/1000
    y2 = Y[1]/1000
    z2 = Y[2]/1000
    xn = PoseObject(x = x1, y = y1, z = z1, roll = 0, pitch = 1.57, yaw =
0)
    yn = PoseObject(x = x2, y = y2, z = z2, roll = 0, pitch = 1.57, yaw =
0)
    Robot.update_tool()
    Robot.release_with_tool()
    Robot.move_pose(xn)
    Robot.grasp_with_tool()
    Robot.move_pose(yn)
    Robot.release_with_tool()
    Robot.move_pose(Home)

def pyramideniryo(l,o):
    lumpy = []
    robot.update_tool()
    c = 0
    op = PoseObject(x = o[0]/1000, y = o[1]/1000, z = o[2]/1000, roll = 0,
pitch = 1.57, yaw = 0)
    for i in l:
        p = PoseObject(x = i[0]/1000, y = i[1]/1000, z = i[2]/1000, roll = 0,
pitch = 1.57, yaw = 0)
        Robot.release_with_tool()
        Robot.move_pose(i)
        Robot.grasp_with_tool()
        Robot.move_pose(Home)
        new_place_pose = o.copy_with_offsets(z_offset=0.01 * c)
        lumpy.append(new_place_pose)
        Robot.move_pose(new_place_pose)
        Robot.release_with_tool()
        Robot.move_pose(Home)
        c = c + 1
    for i in lumpy:
        Robot.release_with_tool()
        Robot.move_pose(lumpy[c-1])
        Robot.grasp_with_tool()
```

```

Robot.move_pose(Home)
Robot.move_pose(l[c-1])
Robot.release_with_tool()
Robot.move_pose(Home)
c = c - 1

def execution(P,l,centrep):
    longueur = len(l)
    if P == "Pickandplace":
        c = 0
        while c+1 < longueur:
            pickpandplaceniryo(l[c],l[c+1])
    else:
        pyramideniryo(l,centrep)

def lecturefichier():
    with open("Données pour Niryo.txt", "r") as fichier:
        contenu = fichier.read()
    # Séparer les données
    Programme, Hauteur_string, Coordonnees_string, Centre_string =
    contenu.split("|")
    # Convertir les coordonnées en liste de listes de réels
    Coordonnees = [list(map(float, paire.strip("[]").split(','))) for paire
in Coordonnees_string.split()]
    Hauteur = float(Hauteur_string)
    Centre = ast.literal_eval(Centre_string)
    return Programme, Coordonnees, Hauteur, Centre

##Execution réelle
Programme = lecturefichier()[0]
Coordonnees = lecturefichier()[1]
Hauteur = lecturefichier()[2]
Centre = lecturefichier()[3]
execution(Programme, Coordonnees,Centre)

```

- Programme simulation :

```

#### Programme Simulation

##Importations des bibliothèques nécessaires

from robodk.robolink import *
from robodk.robomath import *
from tkinter import *
import ast

#Fonction auxiliaire pour trouver le fichier avec son titre
nomfichier = "Niryo-Ned2.robot"
repertoire = os.getcwd()
def find_file(filename):
    for root, dirs, files in os.walk(repertoire):
        if filename in files:
            return os.path.join(root, filename)
    return None
robot_file_path = find_file(nomfichier)
##Fonctions auxiliaires
# #Programme de mouvement à une coordonnée donnée
def mouvrobodk(X):
    x = X[0]

```

```

y = X[1]
z = X[2]
objet = Ned2robodk.AddTarget("Position souhaitée")
Ned2robodk.ShowRoboDK()
obj = Pose(x, y, z, -180.0, 0.0, 180.0)
robot.MoveJ(obj)

def pickandplacerobodk(X,Y):
    mouvrobodk(X)
    mouvrobodk(Y)
    mouvrobodk(Homerobodk)

def pyramiderobodk(l,o):
    lumpy = []
    c = 0
    for i in l:
        mouvrobodk(i)
        mouvrobodk(Home)
        new_place_pose = [o[0],o[1],o[2]*c*Hauteur]
        lumpy.append(new_place_pose)
        mouvrobodk(new_place_pose)
        mouvrobodk(Home)
        c = c + 1
    for i in lumpy:
        mouvrobodk(lumpy[c-1])
        mouvrobodk(Home)
        mouvrobodk(l[c-1])
        mouvrobodk(Home)
        c = c - 1

def simu(P,l,centrep):
    longueur = len(l)
    if P == "Pickandplace":
        c = 0
        while c+1 < longueur:
            pickandplacerobodk(l[c],l[c+1])
    else:
        pyramiderobodk(l,centrep)

def lecturefichier():
    with open(find_file("Données pour Niryo.txt"), "r") as fichier:
        contenu = fichier.read()
    # Séparer les données
    Programme, Hauteur_string, Coordonnees_string, Centre_string =
    contenu.split("|")
    # Convertir les coordonnées en liste de listes de réels
    Coordonnees = [list(map(float, paire.strip("[]").split(','))) for paire
    in Coordonnees_string.split()]
    Hauteur = float(Hauteur_string)
    Centre = ast.literal_eval(Centre_string)
    return Programme, Coordonnees, Hauteur, Centre

##Ouverture de la fenêtre Robodk
Ned2robodk = Robolink()
Ned2robodk.ShowRoboDK()

#Ajout du robot Ned2 et de sa position Home
Ned2robodk.AddFile(robot_file_path)
robot = Ned2robodk.Item(' ', ITEM_TYPE_ROBOT)
Homerobodk = robot.Pose()

```

```

Programme = lecturefichier()[0]
Coordonnees = lecturefichier()[1]
Hauteur = lecturefichier()[2]
Centre = lecturefichier()[3]
simu(Programme, Coordonnees, Centre)

mouvrobodk(Homerobodk)

```

- Programme traduction :

```

- ### Programme Traduction

##Import des bibliothèques de os et tkinter

from tkinter import *
import os

#Programme de création de stockage des données
def creamodif():
    pg = programme.get()
    coord = coordonnees.get()
    Hauteur = hauteur.get()
    Centre = centre.get()
    with open("Données pour Niryo.txt", 'w') as fichier:
        fichier.write(pg + "|" + Hauteur + "|" + coord + "|" +
Centre)
        fichier.close()

##Création de la fenêtre pour saisir les données
fenetre = Tk()
fenetre.title("Simulation et exécution d'un programme de mouvements
sur le robot Ned2'")
fenetre.resizable(width=False, height=False)
label = Label(fenetre, text="Bonjour, voici les données à entrer :")
label.pack()
label = Label(fenetre, text="Programme à utiliser :")
label.pack()
programme = Entry(fenetre, width=30)
programme.pack()
label = Label(fenetre, text="Coordonnées :")
label.pack()
coordonnees = Entry(fenetre, width=30)
coordonnees.pack()
label = Label(fenetre, text="Hauteur de l'objet' :")
label.pack()
hauteur = Entry(fenetre, width=30)
hauteur.pack()
label = Label(fenetre, text="Centre pour la pyramide :")
label.pack()
centre = Entry(fenetre, width=30)
centre.pack()
label = Label(fenetre, text="Une fois validée, fermer la fenêtre.")
label.pack()
button = Button(fenetre, text="Valider", command=creamodif)
button.pack()
fenetre.mainloop()

```