NIRYO

HUMAN - MOTION - ROBOT

PYTHON 1

Programmer le Niryo One avec python™

V1

OBJECTIFS

- Comprendre l'API Python
- Tester quelques programmes afin de se familiariser avec l'API Python
- Être capable de créer des algorithmes pour commander le robot

PRÉREQUIS

- Des connaissances en algorithmique et un niveau basique en Python
- Connaître le fonctionnement matériel du Niryo One

CE DONT VOUS AUREZ BESOIN

- Un ordinateur équipé du WIFI ou d'un port Ethernet
- Ubuntu ou Windows (sous Windows, il sera nécessaire d'installer le logiciel PuTTY depuis l'adresse https://www.putty.org/)
- Niryo One Studio (interface de programmation graphique de Niryo One)
- Un Niryo One
- 6 objets à manipuler, un contenant, une pente (fichiers 3D fournis) et un Gripper Large

COMMENT UTILISER L'API PYTHON SUR LE NIRYO ONE

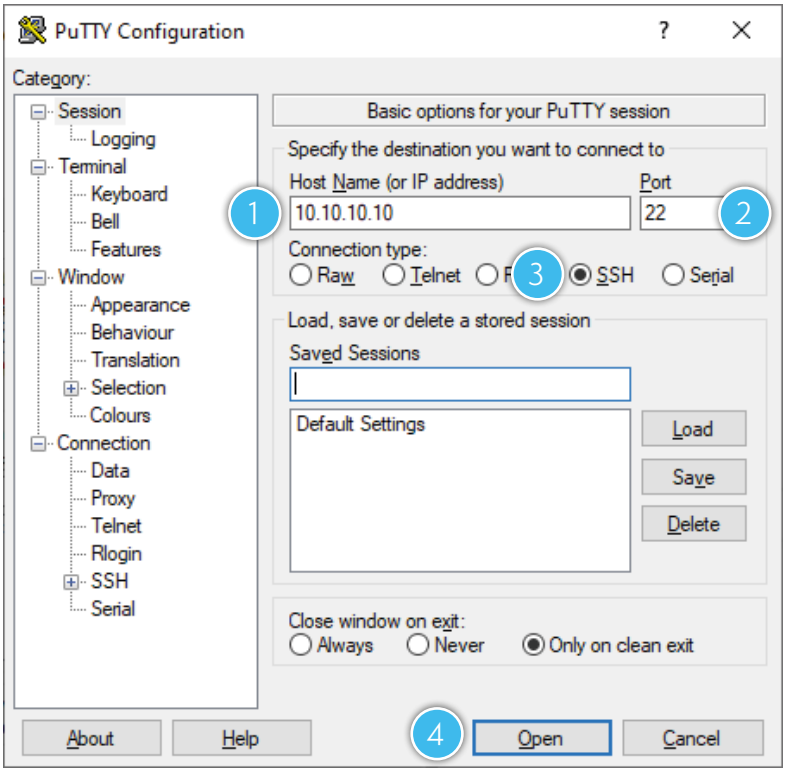
CONNEXION AU ROBOT

Dans cette partie, nous allons nous **connecter au robot via SSH** afin de pouvoir le programmer. Il est pour cela nécessaire de connaître l'adresse IP du Niryo One.

- Si le robot est en mode Hotspot, son adresse IP est 10.10.10.10 (IP statique),
- Si le robot est connecté au réseau WIFI, vous pouvez retrouver son adresse IP depuis l'onglet de connexion dans Niryo One Studio grâce à la fonction de recherche de robots sur le réseau.

Pour effectuer la connexion sous Ubuntu, ouvrez un terminal et entrez "`ssh niryo @(ip_address)`" puis "robotics" lorsqu'un mot de passe sera demandé.

Pour effectuer la connexion sous Windows, lancez le logiciel PuTTY puis renseignez les éléments suivants :



1. Entrez l'IP du robot
2. Indiquez le port 22
3. Sélectionnez SSH
4. Cliquez sur Open

IDENTIFIANT :
« *niryo* »

MOT DE PASSE :
« *robotics* »

PREMIER SCRIPT

- Créez un dossier /TP1

```
mkdir TP1
```

- Créer un fichier Python nommé test.py dans le dossier /TP1

```
cd TP1

touch test.py
```

- Afin de rendre le fichier exécutable, tapez la commande suivante:

```
chmod +x test.py
```

- Ouvrez le fichier test.py avec un éditeur de texte

```
nano test.py
```

- Copier/coller les lignes suivantes :

```
#!/usr/bin/env python

import rospy

print «Welcome !»
```

- Avant de lancer votre script, tapez :

```
source ~/catkin_ws/devel/setup.bash && export PYTHONPATH=${PYTHONPATH}:/
home/niryo/catkin_ws/src/niryo_one_python_api/src/niryo_python_api
```

- Dans le terminal, lancez votre script :

```
python test.py
```

LES FONCTIONS DE BASE DE L'API PYTHON

Retrouvez ci-après les fonctions de base de l'API Python du Niryo One.

Pour plus de détails :
https://github.com/NiryoRobotics/niryo_one_ros/tree/master/niryo_one_python_api

CONSTANTES

Constante	Description
TOOL_NONE	ID des outils
TOOL_GRIPPER_1_ID	
TOOL_GRIPPER_2_ID	
TOOL_GRIPPER_3_ID	
TOOL_ELECTROMAGNET_1_ID	
TOOL_VACUUM_PUMP_1_ID	
PIN_MODE_OUTPUT	Digital pins
PIN_MODE_INPUT	
PIN_HIGH	
PIN_LOW	
GPIO_1A	
GPIO_1B	
GPIO_1C	
GPIO_2A	
GPIO_2B	
GPIO_2C	
SW_1	
SW_2	
AXIS_X	Pour la fonction shift_pose
AXIS_Y	
AXIS_Z	
ROT_ROLL	
ROT_PITCH	
ROT_YAW	

MÉTHODES

Méthode	Paramètres (arguments)	Description
calibrate_auto	-	Calibrer les moteurs du robot automatiquement (en déplaçant les axes). Si la calibration n'est pas nécessaire, cette méthode ne fera rien.

calibrate_manual	-	Calibrer les moteurs du robot manuellement (le robot doit simplement être en position "home" et avoir été auto-calibré au moins une fois). Si la calibration n'est pas nécessaire, cette méthode ne fera rien.
activate_learning_mode	• Activation (True ou False)	Activer ou désactiver le mode d'apprentissage (= activer/désactiver le couple sur les moteurs).
move_joints	• Articulation (liste de 6 valeurs angulaires)	Déplacer le bras avec des positions articulaires.
move_pose	• Position x (m) • Position y (m) • Position z (m) • Rotation x (rad) • Rotation y (rad) • Rotation z (rad)	Déplacer le bras avec une commande de pose.
shift_pose	• Axe (0: pos.x, 1: pos.y, 2: pos.z, 3: rot.x, 4: rot.y, 5: rot.z) • Valeur (m)	Déplacer le bras en incrémentant la pose actuelle par...
set_arm_max_velocity	• Pourcentage (1-100)	Définir la vitesse maximale du bras.
enable_joystick	• Activation (True ou False)	Activer ou désactiver le mode joystick (pour contrôler le robot avec une manette).
pin_mode	• Pin (numéro GPIO) • Mode (0: OUTPUT, 1: INPUT)	Définir une broche d'entrée/sortie numérique en mode INPUT ou OUTPUT.
digital_write	• Pin (numéro GPIO) • État (0: LOW, 1: HIGH)	Définir une broche d'entrée/sortie numérique sur LOW ou HIGH. Notez que la broche doit avoir déjà été définie sur OUTPUT.
digital_read	• Pin (numéro GPIO)	Renvoie l'état actuel de la broche (0: LOW, 1: HIGH).

change_tool	• ID de l'outil (0 pour détacher l'outil)	Changer l'outil attaché. Avant d'exécuter une action sur un outil, vous devez le sélectionner avec cette méthode.
open_gripper	• ID de l'outil • Vitesse (0-1000, recommandé : 100-500)	Ouvrir le préhenseur à la vitesse sélectionnée.
close_gripper	• ID de l'outil • Vitesse (0-1000, recommandé : 100-500)	Fermer le préhenseur à la vitesse sélectionnée. Celui-ci s'arrêtera lorsqu'il détectera un objet.
pull_air_vacuum_pump	• ID de l'outil	Activer la pompe à vide (prendre un objet).
push_air_vacuum_pump	• ID de l'outil	Désactiver la pompe à vide (placer un objet).
setup_electromagnet	• ID de l'outil • Pin (numéro GPIO)	Configurer l'électro-aimant sur les entrées/sorties numériques (définissez le mode des broches sur OUTPUT). Il est nécessaire de configurer l'électromaimant avant de l'utiliser.
activate_electromagnet	• ID de l'outil • Pin (numéro GPIO)	Activer l'électroaimant sur les entrées/sorties numériques (objet pick). Cela définit l'état de la broche sur HIGH.
deactivate_electromagnet	• ID de l'outil • Pin (numéro GPIO)	Désactiver l'électroaimant sur les entrées/sorties numériques (objet de position). Cela définit l'état de la broche sur LOW.
get_saved_position_list	-	Obtenir toutes les positions sauvegardées sur le robot.
wait	• Temps (secondes)	Bloquer et attendre le nombre de secondes indiqué.
get_joints	-	Retourner un tableau contenant les angles actuels pour les 6 axes (en radians).
get_arm_pose	-	Retourner la pose (position en mètres et orientation en radians) de l'organe terminal.

get_hardware_status	-	Renvoyer un objet contenant des informations utiles sur l'état des moteurs, la connexion, la température, etc. (unité de température: °C).
get_learning_mode	-	Renvoyer un booléen indiquant si le mode d'apprentissage est activé ou non.
get_digital_io_state	-	Renvoyer un objet contenant des informations pour toutes les broches numériques 6 * 5V + 2 * 12V (mode : entrée ou sortie ; état : HIGH ou LOW).

PROGRAMMATION DU NIRYO ONE AVEC L'API PYTHON

EXERCICE 1

Nous désirons déplacer le Niryo One aux deux positions P1 et P2 avec :

- P1 = [x = -0.03 ; y = -0.156 ; z= 0.48 ; roll= -0.58 ; pitch = -0.58 ; yaw = -0.145]
- P2 = [x = -0.136 ; y = -0.133 ; z = 0.255 ; roll= -0.081 ; pitch = 0.744 ; yaw = -2.535]

Quelle est la fonction qui permet de déplacer le robot vers une position en fournissant les coordonnées cartésiennes x, y et z ?

- Créer un fichier Python exercice_1.py :

```
touch exercice_1.py
chmod +x exercice_1.py
```

- Copier/coller le code suivant :

```
#!/usr/bin/env python
from niryo_one_python_api.niryo_one_api import *
import rospy
import time


rospy.init_node('niryo_one_example_python_api')


n = NiryoOne()


n.calibrate_auto()


# desactiver le mode apprentissage
n.activate_learning_mode(False)
# deplacer le robot vers la position P1
n.move_pose( -0.03,  -0.156, 0.48,  -0.58,  -0.58, -0.145)
time.sleep(3)
pose_actuel_1 = n.get_arm_pose()
print pose_actuel_1
# deplacer le robot vers la position P2
n.move_pose( -0.136,  -0.133,  0.255, -0.081, 0.744, -2.535)
time.sleep(3)
pose_actuel_2 = n.get_arm_pose()
print pose_actuel_2
n.activate_learning_mode(True)
```

- Exécuter le code :

```
python exercice_1.py
```

Apporter les modifications nécessaires pour que le code ci-dessus déplace le robot de la position P1 à la position P2 quatre fois consécutives.

EXERCICE 2

Nous désirons déplacer le robot de sa position initiale à P1, ouvrir le gripper 1, attendre 1 seconde, fermer le gripper 1, déplacer le robot vers P2 et enfin faire retourner en position initiale.

- Copier/coller le code suivant dans un nouveau fichier exercice_2.py :

```
#!/usr/bin/env python

from niryo_one_python_api.niryo_one_api import *

import rospy

import time

rospy.init_node('niryo_one_example_python_api')


n = NiryoOne()


try:

      n.change_tool(TOOL_GRIPPER_1_ID)

      n.open_gripper(TOOL_GRIPPER_1_ID, 500)

      n.wait(2)

      n. close_gripper(TOOL_GRIPPER_1_ID, 500)

except NiryoOneException as e:

    print e

    # Handle errors here
```

Exécutez le code et commentez. Apportez les modifications nécessaires pour que le code accomplisse la séquence souhaitée.

CAS PRATIQUE : PICK & PACK

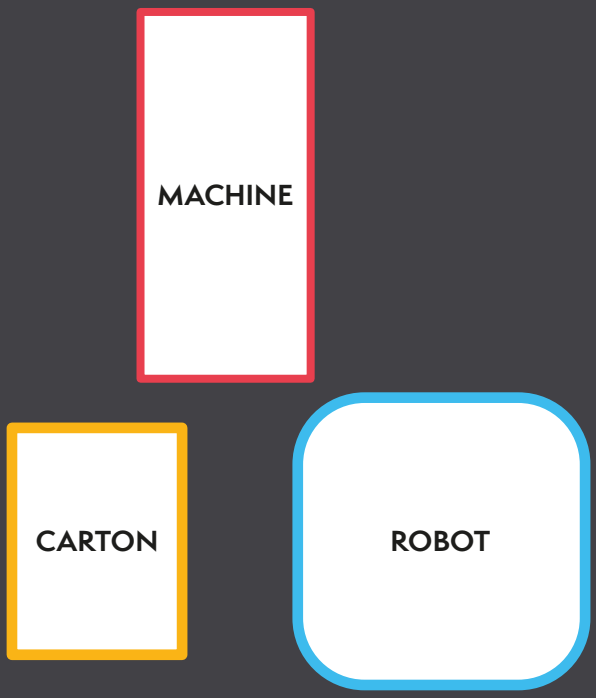
Dans le cadre de la rénovation de l'outil de production de la société BOX qui fabrique et distribue des produits cosmétiques, on vous demande d'automatiser le conditionnement en fin de chaîne qui consiste à déposer des flacons remplis dans des cartons.

Six flacons, disposés sur la machine, symbolisée ici par la pente, doivent être placés dans un carton qui est parfaitement aligné avec le robot. Il sera nécessaire de réaliser la programmation nécessaire pour atteindre cet objectif avec l'API Python du Niryo One.

Mise en place du matériel

- 6 flacons
- Une pente (qui représente une machine)
- Le contenant (qui représente un carton)
- Niryo One
- Gripper large

Schéma



Photo

Travail demandé

Le robot est dans sa position initiale. Lorsque l'opérateur appuie sur le bouton pour indiquer l'arrivée des flacons sur la machine, le conditionnement commence.
L'automatisation de ce système est décomposée en quatre tâches :

1. Le robot se déplace vers la machine pour récupérer un flacon
2. Le robot prend le flacon
3. Le robot se déplace vers le carton pour poser le flacon
4. Le robot pose le flacon

Ces quatre tâches se répètent jusqu'à ce que le carton soit rempli, puis le robot retourne à sa position initiale.

Sur papier, réaliser un organigramme qui traduit le processus de conditionnement.

Brancher et tester le préhenseur avec le code suivant :

```
#!/usr/bin/env python
from niryo_one_python_api.niryo_one_api import *
import rospy
import time
rospy.init_node('niryo_one_example_python_api')

n = NiryoOne()

try:
    n.change_tool(TOOL_GRIPPER_2_ID)
    n.open_gripper(TOOL_GRIPPER_2_ID, 500)
    time.sleep(1)
    n.close_gripper(TOOL_GRIPPER_2_ID, 500)
except NiryoOneException as e:
    print e
    # Handle errors here
```

Étudier le code suivant et décrire la tâche réalisée :

```
pose_1 = [x_1, y_1, z_1, roll_1, pitch_1, yaw_1]
pose_2 = [x_2, y_2, z_2, roll_2, pitch_2, yaw_2]
n.open_gripper(TOOL_GRIPPER_2_ID, 500)
time.sleep(1)
n.move_pose(*pose_1)
time.sleep(1)
n.move_pose(*pose_2)
n.close_gripper(TOOL_GRIPPER_2_ID, 500)
```

En utilisant le mode apprentissage, déterminer les positions de pick et de place.

```
#!/usr/bin/env python
from niryo_one_python_api.niryo_one_api import *
import rospy

rospy.init_node('niryo_one_leanring_mode')
n = NiryoOne()

print "--- Start"

try:
    raw_input("Placez le robot a sa position initiale et appuyez sur
entree")
    pose_initiale = n.get_arm_pose()
    print "pose_initiale = ", pose_initiale

    raw_input("Deplacez vers la position 1 et appuyez sur entree")
    pose_pick_1 = n.get_arm_pose()
```

(Suite du code page suivante)

```
        print("pose_pick_1 = ", pose_pick_1)


        raw_input("deplacez le robot vers une position pick 2 et ensuite
    appuyez sur entree")
        pose_pick_2 = n.get_arm_pose()
        print("pose_ pick_2 = ", pose_pick_2)
    except NiryoOneException as e:
        print e


    print "--- End"
```

Écrire le cycle complet de remplissage du carton et commenter les choix effectués.

CORRECTION : EXERCICE 1

Nous désirons déplacer le Niryo One aux deux positions P1 et P2 avec :

- P1 = [x = -0.03 ; y = -0.156 ; z= 0.48 ; roll= -0.58 ; pitch = -0.58 ; yaw = -0.145]
- P2 = [x = -0.136 ; y = -0.133 ; z = 0.255 ; roll= -0.081 ; pitch = 0.744 ; yaw = -2.535]

Quelle est la fonction qui permet de déplacer le robot vers une position en fournissant les coordonnées cartésiennes x, y et z ?

Il s'agit de la fonction move_pose.

- Créer un fichier Python exercice_1.py :

```
touch exercice_1.py
chmod +x exercice_1.py
```

- Copier/coller le code suivant :

```
#!/usr/bin/env python
from niryo_one_python_api.niryo_one_api import *
import rospy
import time

rospy.init_node('niryo_one_example_python_api')

n = NiryoOne()

n.calibrate_auto()
```

(Suite du code page suivante)

```
# desactiver le mode apprentissage
n.activate_learning_mode(False)
# deplacer le robot vers la position P1
n.move_pose( -0.03,  -0.156, 0.48,  -0.58,  -0.58, -0.145)
time.sleep(3)
pose_actuel_1 = n.get_arm_pose()
print pose_actuel_1
# deplacer le robot vers la position P2
n.move_pose( -0.136,  -0.133,  0.255, -0.081, 0.744, -2.535)
time.sleep(3)
pose_actuel_2 = n.get_arm_pose()
print pose_actuel_2
n.activate_learning_mode(True)
```

- Exécuter le code :

```
python exercice_1.py
```

Apporter les modifications nécessaires pour que le code ci-dessus déplace le robot de la position P1 à la position P2 quatre fois consécutives.

```
#!/usr/bin/env python
from niryo_one_python_api.niryo_one_api import *
import rospy
import time


rospy.init_node('niryo_one_example_python_api')


n = NiryoOne()
```

(Suite du code page suivante)

```
# desactiver le mode d'apprentissage
n.activate_learning_mode(False)
for i in range(1,4) :
        # deplacer le robot vers la position P1
        n.move_pose( -0.03,  -0.156, 0.48,  -0.58,  -0.58, -0.145)
        time.sleep(3)
        pose_actuel_1 = n.get_arm_pose()
        print pose_actuel_1
        # deplacer le robot vers la position P2
        n.move_pose( -0.136,  -0.133,  0.255, -0.081, 0.744, -2.535)
        time.sleep (3)
        pose_actuel_2 = n.get_arm_pose()
        print pose_actuel_2
n.activate_learning_mode(True)
```

CORRECTION : EXERCICE 2

Nous désirons déplacer le robot de sa position initiale à P1, ouvrir le gripper 1, attendre 1 seconde, fermer le gripper 1, déplacer le robot vers P2 et enfin faire retourner en position initiale.

- Copier/coller le code suivant dans un nouveau fichier exercice_2.py :

```
#!/usr/bin/env python
from niryo_one_python_api.niryo_one_api import *
import rospy
import time
rospy.init_node('niryo_one_example_python_api')


n = NiryoOne()


try:
     n.change_tool(TOOL_GRIPPER_1_ID)
     n.open_gripper(TOOL_GRIPPER_1_ID, 500)
     n.wait(1)
    n. close_gripper(TOOL_GRIPPER_1_ID, 500)
except NiryoOneException as e:
    print e
    # Handle errors here
```

Exécutez le code et commentez. Apportez les modifications nécessaires pour que le code accomplisse la séquence souhaitée.

```
#!/usr/bin/env python
from niryo_one_python_api.niryo_one_api import *
import rospy
import time
```

(Suite du code page suivante)

```
rospy.init_node('niryo_one_example_python_api')


n = NiryoOne()
try:
    '''si le robot n'est pas calibre, decommenter la ligne suivante'''
    #n.calibrate_auto()
    n.change_tool(TOOL_GRIPPER_1_ID)


    # deplacer le robot vers la position P1
    n.move_pose( -0.03, -0.156, 0.48, -0.58, -0.58, -0.145)
    # ouvrir le gripper
    n.open_gripper(TOOL_GRIPPER_1_ID, 500)
    n.wait(1)
    # fermer le gripper
    n.close_gripper(TOOL_GRIPPER_1_ID, 500)


    # deplacer le robot vers la position P2
    n.move_pose( -0.136, -0.133, 0.255, -0.081, 0.744, -2.535)
    n.wait(1)
    # retourner vers la position initiale (vous pouvez definir la
votre)
    n.move_pose( 0.231, -0.083, 0.417, -0.010, 0, -0.347)
    n.wait(1)
    n.activate_learning_mode(True)


except NiryoOneException as e:
    print(e)
    # Handle errors here
```

CORRECTION : CAS PRATIQUE

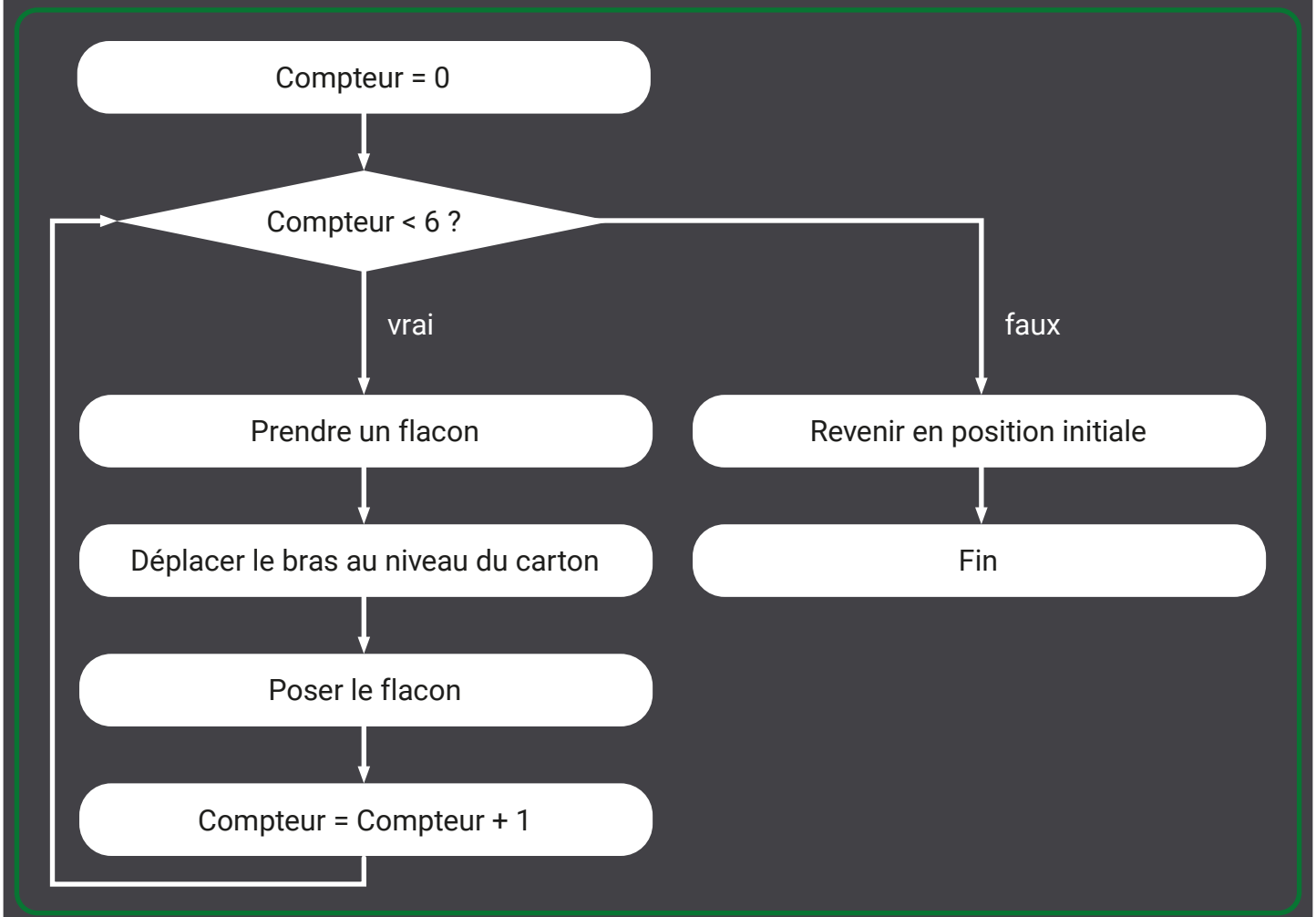
Travail demandé

Le robot est dans sa position initiale. Lorsque l'opérateur appuie sur le bouton pour indiquer l'arrivée des flacons sur la machine, le conditionnement commence.
L'automatisation de ce système est décomposée en quatre tâches :

1. Le robot se déplace vers la machine pour récupérer un flacon
2. Le robot prend le flacon
3. Le robot se déplace vers le carton pour poser le flacon
4. Le robot pose le flacon

Ces quatre tâches se répètent jusqu'à ce que le carton soit rempli, puis le robot retourne à sa position initiale.

Sur papier, réaliser un organigramme qui traduit le processus de conditionnement.



Brancher et tester le préhenseur avec le code suivant :

```
#!/usr/bin/env python
from niryo_one_python_api.niryo_one_api import *
import rospy
import time
rospy.init_node('niryo_one_example_python_api')


n = NiryoOne()


try:
    n.change_tool(TOOL_GRIPPER_2_ID)
    n.open_gripper(TOOL_GRIPPER_2_ID, 500)
    time.sleep(1)
    n.close_gripper(TOOL_GRIPPER_2_ID, 500)
except NiryoOneException as e:
    print e
    # Handle errors here
```

Étudier le code suivant et décrire la tâche réalisée :

```
pose_1 = [x_1, y_1, z_1, roll_1, pitch_1, yaw_1]
pose_2 = [x_2, y_2, z_2, roll_2, pitch_2, yaw_2]
n.open_gripper(TOOL_GRIPPER_2_ID, 500)
time.sleep(1)
n.move_pose(*pose_1)
time.sleep(1)
n.move_pose(*pose_2)
n.close_gripper(TOOL_GRIPPER_2_ID, 500)
```

Le code définit tout d'abord la position P1 puis la position P2.

Ensuite, le robot ouvre son préhenseur, attend une seconde, se déplace vers P1, attend une seconde, se déplace vers P2 puis ferme le préhenseur.

En utilisant le mode apprentissage, déterminer les positions de pick et de place.

```
#!/usr/bin/env python

from niryo_one_python_api.niryo_one_api import *

import rospy


rospy.init_node('niryo_one_leanring_mode')

n = NiryoOne()


print "--- Start"


try:

   raw_input("Placez le robot a sa position initiale et appuyez sur
entree")

   pose_initiale = n.get_arm_pose()

   print "pose_initiale = ", pose_initiale


   raw_input("Deplacez vers la position 1 et appuyez sur entree")

   pose_pick_1 = n.get_arm_pose()

   print("pose_pick_1 = ", pose_pick_1)


   raw_input("deplacez le robot vers une position pick 2 et ensuite
appuyez sur entree")

   pose_pick_2 = n.get_arm_pose()

   print("pose_ pick_2 = ", pose_pick_2)
```

(Suite du code page suivante)

```
except NiryoOneException as e:
    print e


print "--- End"
```

Écrire le cycle complet de remplissage du carton et commenter les choix effectués.

La solution présentée ci-dessous n'est pas la seule méthode possible.

La zone de conditionnement étant alignée avec le robot, il nous sera possible de calculer les six positions de "place" par rapport à la première.

```
#!/usr/bin/env python
from niryo_one_python_api.niryo_one_api import *
import rospy


def copy_position_with_offsets(copied_pose, x_offset=0.0, y_offset=0.0,
z_offset=0.0):
    new_pose = [v for v in copied_pose]  # copier toutes les valeurs
    new_pose[0] += x_offset  # adjust x
    new_pose[1] += y_offset  # adjust y
    new_pose[2] += z_offset  # adjust z
    return new_pose



# -- Poses a changer pour adapter a votre cas
# pose initiale
pose_start = [0.1, 0.1, 0.2, 1.57, 0.0, 1.57]
```

(Suite du code page suivante)

```
# pose de picking
pose_pick = [0.1, 0.0, 0.1, 1.57, 0.0, 1.57]


# calculer la position de survol au dessus de la zone de picking pour
permettre le passage par cette position avant de se deplacer vers
pose_pick
pose_pick_high = copy_position_with_offsets(pose_pick, z_offset=0.05)


# pose du robot pour placer le premier flacon en haut a gauche
pose_place_top_left = [0., 0.1, 0.1, 1.57, 0, 1.57]


# parametres de l outil
tool = TOOL_GRIPPER_2_ID
speed_tool = 500


def pick_n_place(niryo_one):
   niryo_one.change_tool()
   # ouvrir le gripper
   niryo_one.open_gripper(tool, speed_tool)
   # deplacer vers la position initiale
   niryo_one.move_pose(*pose_start)
   for i in range(2):  # iteration sur les deux rangees
       for j in range(3):  # iteration sur les trois colonnes
           # deplacement au dessus du flacon
           niryo_one.move_pose(*pose_pick_high)
           # Pick du flacon et retour en position de survol
           niryo_one.move_pose(*pose_pick)
           niryo_one.close_gripper(tool, speed_tool)
```

(Suite du code page suivante)

```
            niryo_one.move_pose(*pose_pick_high)


            # calcul de la position suivante
            next_place_pose = copy_position_with_offsets(pose_place_
top_left, x_offset=0.05 * i, y_offset=0.05 * j)
            next_place_pose_high = copy_position_with_offsets(next_
place_pose, z_offset=0.05)


            niryo_one.move_pose(*next_place_pose_high)
            niryo_one.move_pose(*next_place_pose)
            niryo_one.open_gripper(tool, speed_tool)
    niryo_one.move_pose(*pose_start)



if __name__ == '__main__':
    rospy.init_node('niryo_one_example_python_api')
    n = NiryoOne()
    try:
        pick_n_place(n)
    except NiryoOneException as e:
        print e
```