

Grado en Robótica EPSE-USC Lugo Física II

Práctica 1

INTRODUCCIÓN AL ROBOT NIRYO ONE CINEMÁTICA DIRECTA



Índice

1. Objetivo de la práctica	2
2. Material necesario	2
3. Estructura, características y métodos de control del robot	2
4. Primeros pasos con el robot	4
4.1. Encendido	4
4.2. Apagado	4
4.3. Problemas de conexión a los motores	4
4.4. Ejecutar secuencia de movimientos pregrabada	4
5. Descripción básica del software Niryo One Studio	4
5.1. Conexión al Niryo One	4
5.2. Desconectar el robot	5
5.3. Calibración	5
5.4. Control del robot	5
5.5. Modo de aprendizaje	5
5.6. Programación del Niryo One con Blockly	6
5.6.1. Entorno de programación	6
5.6.2. Tipos de bloques	6
6. Programación del Niryo One en Python	7
6.1. Definición de constantes del robot en la API Python para Niryo One	7
6.2. Definición de métodos del robot en la API Python para Niryo One	7
6.3. Primer script Python para control del Niryo One	8
6.4. Script Python para mover el robot Niryo One entre dos posiciones	9
6.5. Script Python para control de pinza	9
7. Ejercicios prácticos	11
7.1. Explora las articulaciones del robot.	11
7.2. Practica el modo de aprendizaje del robot.	11
7.3. Programación básica de movimientos con Blockly.	12
7.4. Programación del Niryo One con Python.	12
7.5. Resolución del problema cinemático directo para el Niryo One.	12
8. Apéndice	14
8.1. Código para resolución del problema cinemático directo con Niryo One	14
8.2. Ejemplo de código para mover el Robot cíclicamente entre 3 posiciones	15

1. Objetivo de la práctica

En esta práctica nos familiarizaremos con los detalles técnicos y prácticos del brazo robótico Niryo One y realizaremos algunos ejercicios de Cinemática Directa.

2. Material necesario

- Robot Niryo One
- Ordenador con tarjeta Wifi y software Niryo One Studio
- Tu código para resolver el problema cinemático directo

3. Estructura, características y métodos de control del robot

- Se trata de un brazo robótico de 6 grados de libertad, con 6 eslabones y 6 articulaciones de revolución (ver portada del guión). Casi todas sus piezas están impresas en 3D, aunque también tiene algunas partes de aluminio.

grados	J1	J2	J3	J4	J5	J6
MIN	-175	-90	-80	-175	-100	-147.5
MAX	175	36.7	90	175	100	147.5

radianes	J1	J2	J3	J4	J5	J6
MIN	-3.054	-1.571	-1.396	-3.054	-1.745	-2.574
MAX	3.054	0.6405	1.571	3.054	1.745	2.574

Tabla 1: Valores máximos y mínimos que pueden alcanzar las diferentes articulaciones.

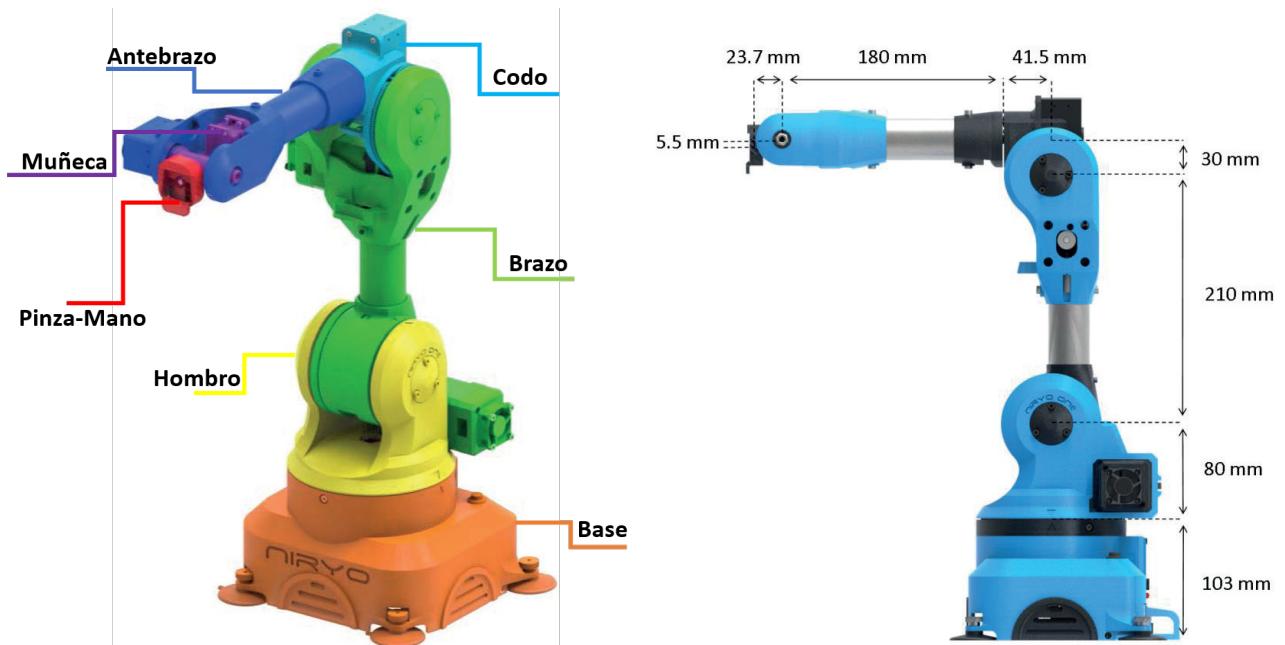


Figura 1: Representaciones del robot Niryo One en su posición cero, donde se diferencian en distintos colores las partes de su estructura y se especifican sus dimensiones.

- Tiene disponibles 5 tipos de pinzas (Figura 2), incluyendo un electroimán y una ventosa, aunque los archivos STL de su estructura están disponibles públicamente y se pueden modificar para incorporar otras herramientas.
- Tiene 3 motores paso a paso en la base, hombro y codo y 3 servomotores Dynamixel (muy precisos) para el resto de las articulaciones.



Figura 2: Tipos de pinzas disponibles para el Niryo One. La estándar se utiliza para agarrar objetos pequeños con precisión, la grande para agarrar objetos más grandes, o que están alejados, la adaptativa para objetos frágiles con formas irregulares complejas, el electroimán se utiliza para mover objetos pequeños con partes metálicas y la bomba de vacío para mover objetos con superficies suaves y no porosas.

- El robot tiene su propio ordenador, una tarjeta Raspberry Pi, que procesa las instrucciones dadas por el usuario y controla la operación de todos los sensores y actuadores necesarios para realizar tareas. Además se le pueden conectar sensores externos adicionales a los puertos accesibles en su parte posterior.
- Tiene 3 capas de software: una aplicación de escritorio (Niryo one Studio), el software del robot (Niryo One Raspberry Pi 3B image) y el firmware de los motores (Niryo Stepper); ordenados de más alto a más bajo nivel. Los 3 diferentes niveles de software se comunican entre ellos y han de ser compatibles, por lo que la actualización de uno de ellos implica la actualización de los demás. Admite programación en ROS utilizando el paquete específico del Niryo One. Además hay un paquete Python que envuelve comandos ROS y también se puede controlar directamente mediante el software Niryo One Studio, que facilita mucho el control del robot a través de código Blockly.
- Tiene varias interfaces físicas en su parte posterior, como se muestra en la figura 3:
 1. Botón superior de apagado/encendido e inicializador de secuencias preprogramadas.
 2. Puerto Ethernet de la Raspberry Pi 3B.
 3. 4 puertos USB.
 4. LED indicador del estado del robot.
 5. Conexión del bus CAN para los Niryo Steppers (no se utiliza).
 6. Conector Dynamixel XL-320 (para la bomba de vacío).
 7. Conector Dynamixel XL-430 (no se utiliza).
 8. Salidas de 12 V programables mediante software.
 9. Paneles GPIO: 6 pines digitales programables mediante software.
 10. Interruptor de encendido/apagado del robot.
 11. Conector del adaptador de potencia, que tiene que tener una salida de 11.1 V y ser capaz de proporcionar 6A.



Figura 3: Interfaces físicas del Niryo One.

- El robot se puede conectar al ordenador directamente al ordenador mediante Wifi o por un cable Ethernet.
- Su peso total es de 3.2 kg, su alcance máximo es de 440 mm, es capaz de levantar una carga de 300 g, y su repetitividad es de 1 mm.

4. Primeros pasos con el robot

Antes de encender el robot asegúrate de que está situado en una superficie plana con los 4 soportes de su base bien apoyados y que no se balancea. Asegúrate también de que el robot tiene suficiente espacio a su alrededor para trabajar sin impactar con ningún objeto. El interruptor debe estar en la posición de apagado antes de enchufarlo.

4.1. Encendido

- Enchufa el robot a la toma de corriente y presiona el interruptor a la posición de encendido. El piloto luminoso que está en la parte posterior del robot debe ponerse en color rojo. **IMPORTANTE:** No desconectes el robot en este momento porque la tarjeta Raspberry Pi se está inicializando y podría dañarse si se corta la corriente en este proceso.
- Espera a que la luz cambie a color azul o verde, lo cual indica que el robot está listo para conectarse.

4.2. Apagado

Para evitar daños, **nunca desenchufes el robot directamente**, sin seguir los pasos que se explican a continuación:

- Presiona durante 3 segundos el botón que está en la parte posterior del robot (ver figura) hasta que la luz se ponga de color púrpura (3 segundos).
- Cuando el piloto esté de color púrpura, suelta el botón y espera a que la luz se ponga de color rojo. Cuando eso suceda, pon el interruptor en la posición de apagado y desenchufa el robot.



4.3. Problemas de conexión a los motores

Si el LED se pone de color azul o verde pero en modo intermitente, significa que hay un problema de conexión a los motores.

4.4. Ejecutar secuencia de movimientos pregrabada

El robot tiene preprogramada una secuencia de movimientos que se puede modificar. Activa la secuencia presionando una vez el botón de la parte posterior.

OJO!! NO MANTENGAS EL BOTÓN PRESIONADO, PÚLSALO Y SUÉLTALO INMEDIATAMENTE O SE INICIARÁ EL PROCESO DE APAGADO DEL ROBOT.

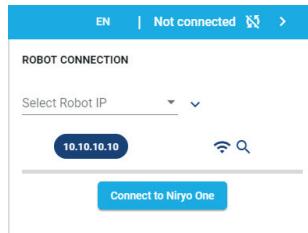
5. Descripción básica del software Niryo One Studio

Es una aplicación de escritorio que permite controlar el Niryo One. Permite configurar el robot, cambiar parámetros, moverlo y programar una secuencia.

5.1. Conexión al Niryo One

- Inicia la aplicación.
- Enciende el robot y el ordenador (en Linux).
- Conecta el ordenador a la red Wifi interna del laboratorio: TP-Link_57BD (password: 17714457).
- Inicia el software Niryo One Studio haciendo click en el icono que está en el escritorio.

- Haz click en el menú que está en la parte superior derecha de la pantalla y se abrirá un panel como el de la figura.
- Haz click en el símbolo de "Wi-Fi Search" ( ). En la casilla "IP address" (despliégalas) aparecerán las IP de todos los robots que están conectados a la misma Wi-Fi. Si el robot tiene un nombre personalizado, veremos el nombre en lugar de la IP. Una vez que escogemos el robot, hacemos click en "Connect to Niryo One" y ya deberíamos estar conectados.
- El robot también puede conectarse directamente a través de un cable Ethernet o utilizando su Wifi Interna pero nosotros utilizaremos el método anterior.



5.2. Desconectar el robot

- Para desconectar el robot haz click en la opción "Disconnect from Niryo One".

5.3. Calibración

La calibración del robot permite la identificación de la posición relativa de los eslabones del robot.

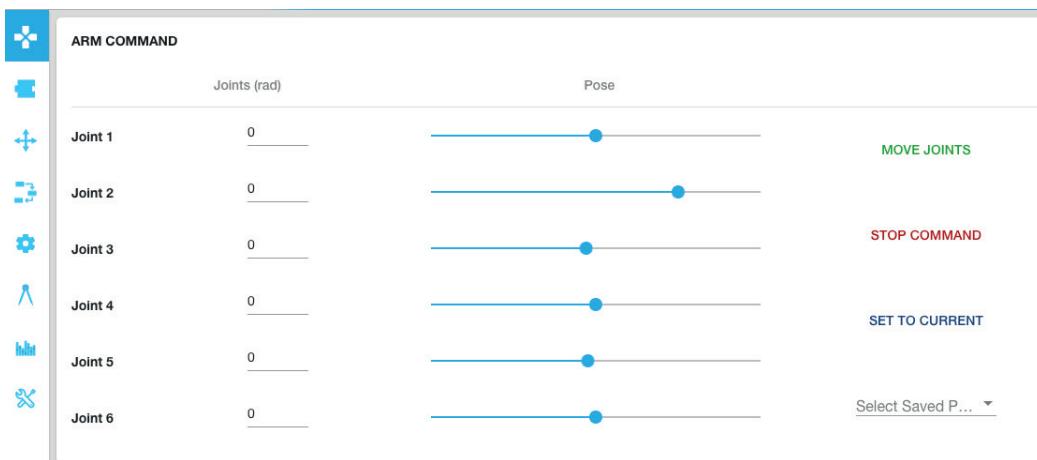
- Calibración automática: se recomienda en el primer uso del Niryo One. El robot realizará una secuencia de movimientos hasta las posiciones máximas de las articulaciones, en las que se bloqueará. El robot alcanzará la posición de la figura. No es recomendable abusar de la calibración automática para evitar el desgaste de las piezas.
- Calibración manual: es una opción igualmente válida y más rápida. Para ello hay que colocar los eslabones manualmente en la posición de la figura, alineando las flechas (triángulos en relieve) que aparecen entre la base y el primer eslabón y entre el primer y segundo eslabón.



5.4. Control del robot

Para controlar de manera directa las coordenadas de las articulaciones, haz click en la primera opción del menú que aparece en la barra de la izquierda de la pantalla. Verás un menú con 4 columnas.

- En la primera columna aparecen los nombres de las articulaciones: Joint 1 ··· Joint 6.
- En la segunda columna puedes introducir los ángulos de rotación para cada uno de ellos
- En la tercera columna hay unas barras de desplazamiento con las que puedes cambiar directamente la posición del punto central de la herramienta (TCP) directamente en coordenadas cartesianas de la base.
- La última columna permite activar o parar el movimiento a la configuración especificada en las columnas 2 y 3, así como actualizar los valores de estos parámetros de acuerdo con la configuración real del robot.



5.5. Modo de aprendizaje

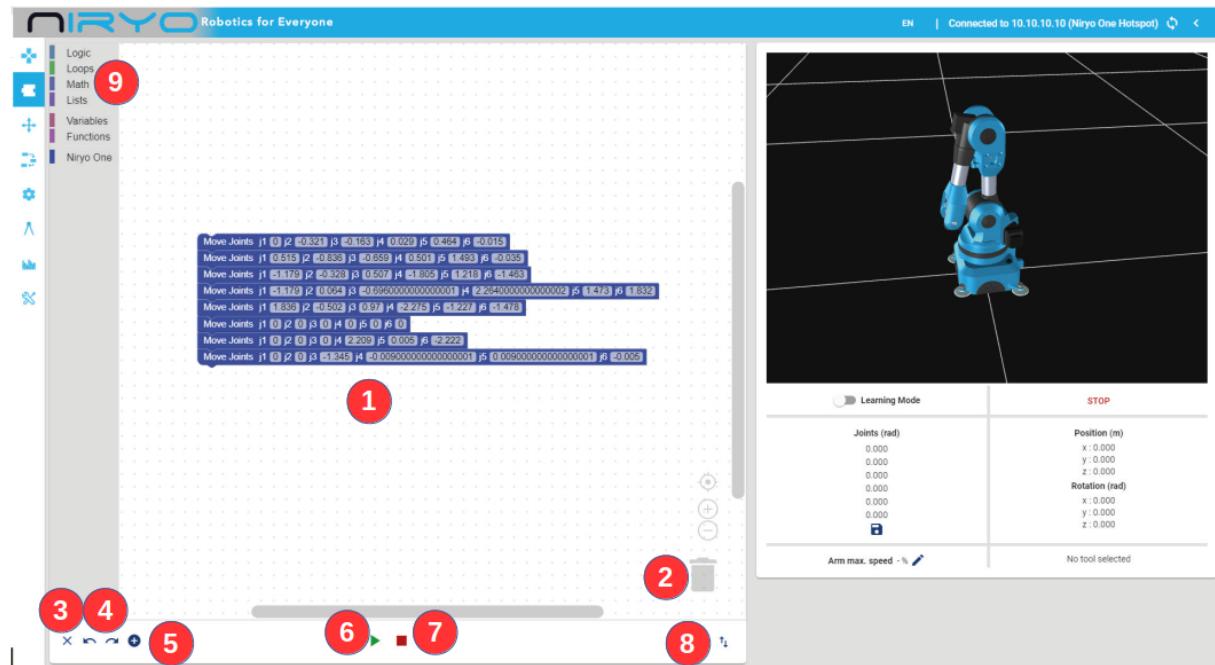
El "learning mode" es un método de programación en el cual el operador humano coloca el robot en una determinada configuración y la salva para ejecutarla más tarde. Haz click en el botón "Learning mode" para activar o desactivar este modo en la interfaz del Niryo One Studio.

5.6. Programación del Niryo One con Blockly

Blockly es la librería de Google que se utiliza en el proyecto Scratch del MIT y en la que también está basada la interfase de programación visual de Niryo One Studio.

5.6.1. Entorno de programación

El entorno de programación tiene los siguientes elementos y funcionalidades (ver números en la siguiente figura):



1. Espacio de trabajo en el que se introduce el código.
2. Arrastre de bloques a la papelera para borrarlo.
3. Borrado del espacio de trabajo.
4. Deshacer/Rehacer.
5. Añadir un bloque.
6. Ejecutar la secuencia que se muestra en el espacio de trabajo.
7. Importar o exportar una secuencia.
8. Lista de bloques de programación.

5.6.2. Tipos de bloques

Hay diferentes tipos de bloques de programación, tal como se puede ver en la figura 4. Su uso y significado es intuitivo para vosotros, que tenéis experiencia en programación.



Figura 4: Tipos de bloques disponibles en la interfase de programación visual Blockly de Niryo One Studio.

6. Programación del Niryo One en Python

Es posible operar el robot directamente usando Python. Primero debemos conectarnos al Niryo One mediante SSH, para lo cual necesitamos conocer su IP. Si el robot está en modo "Hotspot" su IP es 10.10.10.10 mientras que si está conectado vía Wi-Fi, debemos extraer su IP tal como explicamos en la sección 5.1.

- Para conectarnos desde una terminal unix, tecleamos: "ssh niryo@(ip address)", entonces nos pedirá un password y tecleamos "robotics".

6.1. Definición de constantes del robot en la API Python para Niryo One

- Nombre de las herramientas: TOOL_NONE, TOOL_GRIPPER_1_ID, TOOL_GRIPPER_2_ID, TOOL_GRIPPER_3_ID, TOOL_ELECTROMAGNET_1_ID, TOOL_VACUUM_PUMP_1_ID
- Pines digitales: PIN_MODE_OUTPUT, PIN_MODE_INPUT, PIN_HIGH, PIN_LOW, GPIO_1A, GPIO_1B, GPIO_1C, GPIO_2A, GPIO_2B, GPIO_2C, SW_1, SW_2
- Para la función shift_pose: AXIS_X, AXIS_Y, AXIS_Z, ROT_ROLL, ROT_PITCH, ROT_YAW

6.2. Definición de métodos del robot en la API Python para Niryo One

- **calibrate_auto**: calibra los motores del robot automáticamente moviendo los ejes. Si la calibración no es necesaria, este método no hace nada.
- **calibrate_manual**: calibra los motores del robot manualmente. El robot debe estar en la posición cero y debe haber sido autocalibrado al menos una vez. Si la calibración no es necesaria, este método no hace nada.
- **activate_learning_mode**: activa o desactiva el modo de aprendizaje (activa o desactiva el torque en los motores). Tiene como parámetro "True" ó "False".
- **move_joints**: Mueve el brazo a las posiciones de las articulaciones indicadas como argumentos. Los parámetros son los 6 ángulos (en radianes).
- **move_pose**: Mueve el brazo a una pose definida por un conjunto de 3 coordenadas cartesianas y 3 ángulos. Los parámetros son las coordenadas x, y, z (en metros) y los ángulos de rotación (en radianes) entorno a los ejes x, y, z.
- **shift_pose**: Mueve el brazo incrementando la pose actual en las cantidades dadas como parámetros: pos.x, pos.y, pos.z, rot.x, rot.y, rot.z. Las posiciones están en metros y los ángulos en radianes.
- **set_arm_max_velocity**: Selecciona la velocidad máxima del robot. El parámetro es el porcentaje respecto a la velocidad máxima permitida por los motores.
- **enable_joystick**: Activa o desactiva el modo joystick (para controlar el robot con un joystick. El parámetro es "True" ó "False".
- **pin_mode**: Define un pin digital en modo INPUT ó OUTPUT. Los parámetros son el ID del pin (número de GPIO) y el modo (0 para OUTPUT y 1 para INPUT).
- **digital_write**: Selecciona un pin GPIO a modo HIGH ó LOW. El pin debe haber sido seleccionado previamente en modo OUTPUT. Los parámetros son el número del pin y el estado (0 para LOW y 1 para HIGH).
- **digital_read**: Devuelve el estado del pin GPIO actual (0 si LOW y 1 si HIGH). El parámetro es el número del pin GPIO.
- **change_tool**: Cambia la herramienta. Antes de ejecutar una acción con una herramienta, es necesario seleccionarla con este método. El parámetro es el ID de la herramienta.
- **open_gripper**: Abre la pinza a la velocidad seleccionada. Los parámetros son la ID de la herramienta y la velocidad (un valor entre 0 y 1000, aunque se recomiendan valores entre 100 y 500).
- **close_gripper**: Cierra la pinza a la velocidad seleccionada. La pinza parará si detecta un objeto. Los parámetros son la ID de la herramienta y la velocidad.

- **pull_air_vacuum_pump:** Activa la bomba de vacío (coge un objeto). El parámetro es la ID de la herramienta.
- **push_air_vacuum_pump:** Apaga la bomba de vacío (suelta un objeto). El parámetro es la ID de la herramienta.
- **setup_electromagnet:** Configura el electroimán con los inputs/outputs digitales (selecciona el modo GPIO a OUTPUT). Es necesario configurar el electroimán antes de usarlo. Los parámetros son la ID de la herramienta y el número del pin GPIO.
- **activate_electromagnet:** Activa el electroimán con los inputs/outputs digitales (recoge un objeto). Pone el GPIO a HIGH. Los parámetros son la ID de la herramienta y el número del pin GPIO.
- **deactivate_electromagnet:** Desactiva el electroimán con los inputs/outputs digitales (suelta un objeto). Pone el GPIO a LOW. Los parámetros son la ID de la herramienta y el número del pin GPIO.
- **get_saved_position_list:** Obtiene todas las posiciones almacenadas en el robot.
- **wait:** Bloquea y espera durante el número especificado de segundos. El parámetro es el tiempo en segundos.
- **get_joints:** Devuelve una tabla con los ángulos de los 6 ejes en radianes.
- **get_arm_pose:** Devuelve la pose (posición en metros y orientación en radianes) del elemento terminal.
- **get_hardware_status:** Devuelve un objeto con información útil sobre los motores (estado, conexión, temperatura en C, etc).
- **get_learning_mode:** Devuelve un booleano que indica si está o no activado el modo de aprendizaje.
- **get_digital_io_state:** Devuelve un objeto con información para todos los pines digitales (6 de 5V + 2 de 12V). La información que devuelve es el modo (INPUT/OUTPUT) y el estado (HIGH/LOW).

En [este repositorio](#) aparece el listado de comandos Python para controlar el Niryo One, junto con ejemplos:

6.3. Primer script Python para control del Niryo One

- Genera un archivo "test.py" en el ordenador local con el siguiente contenido:

```
#!/usr/bin/env python
import rospy
print "Welcome!"
```

Fíjate que el argumento del comando "print" está entre comillas y no entre paréntesis. Esto es porque el Robot tiene instalada la versión 2.7 de Python.

- Ahora conéctate al robot de nuevo. Abres una terminal y utilizas de nuevo el comando "ssh". Te pedirá el password (robotics):

```
ssh niryo@(ip_address)
```

- Crea un directorio con tu nombre (**no se te ocurra utilizar espacios ni caracteres no estándar como "ñ" o acentos en el nombre del directorio**):

```
mkdir tunombre
```

- Copiamos el script en el robot. Para ello abrimos otra terminal y nos movemos al directorio donde grabamos el script. Utiliza los comandos "ls", "cd" y "pwd" para listar el contenido del directorio en el que te encuentras, para moverte de directorio y para identificar el directorio en el que estás, respectivamente. Una vez que estés en la misma carpeta que tu script Python, puedes copiarlo al robot con la siguiente instrucción:

```
scp test.py niryo@(ip_address):tunombre
```

Al escribir este comando te pedirá de nuevo el password (robotics).

- Comprueba desde la otra terminal, desde la que estás conectado al robot, que el archivo está en la carpeta que creaste. Utiliza para ello el comando "ls".
- Haz ejecutable el script:

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

- Introduce el siguiente comando:

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

- Ejecuta el script en la terminal del robot:

```
python test.py
```

6.4. Script Python para mover el robot Niryo One entre dos posiciones

Vamos a desarrollar un código para mover el robot entre las siguientes 2 posiciones:

```
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]
```

Crea un archivo ej1.py en el ordenador local con el siguiente contenido:

```
#!/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()

n.activate_learning_mode(False) # desactiva el modo de aprendizaje

n.move_pose(-0.03, -0.156, 0.48, -0.58, -0.58, -0.145) # mueve el robot a P1
time.sleep(3)
pose_actuel_1 = n.get_arm_pose()
print pose_actuel_1

n.move_pose(-0.136, -0.133, 0.255, -0.081, 0.744, -2.535) # mueve el robot a P2
time.sleep(3)
pose_actuel_2 = n.get_arm_pose()
print pose_actuel_2

n.activate_learning_mode(True)
```

Transfiere el código al robot y ejecútalo igual que hiciste antes con el archivo "test.py":

```
python ej1.py
```

6.5. Script Python para control de pinza

Código para mover el robot desde la posición inicial a la posición P1, abrir la pinza, esperar 1 segundo, cerrar la pinza, mover el robot a la posición P2 y finalmente volver a la posición inicial:

```
#!/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
```

Sigue los mismos pasos de los códigos anteriores y ejecuta el script:

```
python ej2.py
```

7. Ejercicios prácticos

7.1. Explora las articulaciones del robot.

- Utilizando el **Niryo One Studio** mueve las distintas articulaciones de manera independiente.
- Mueve todas las articulaciones simultáneamente.
- Haz una autocalibración del robot.
- Después de la calibración mueve las articulaciones del robot a los siguientes valores:

$$\theta = [1.6, 0.016, -0.1, -0.026, -0.005, 0]$$

- Haz click en "SET TO CURRENT" para leer los valores de las coordenada en las que se encuentra el robot.
- Cambia la configuración del robot a una posición aleatoria.
- Haz click en "POSE" y cambia los valores:

$$x = 0.08; \quad y = -0.04; \quad z = 0.55$$

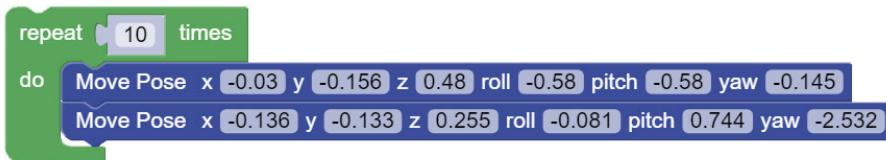
- Haz click en "MOVE POSE".
- Haz click en "SET TO CURRENT" para leer los valores de las coordenada en las que se encuentra el robot.
- Haz click en el icono de "guardar" dale un nombre a la posición.
- Pon a cero valor de las coordenadas de todas las articulaciones para llevar el robot a su configuración cero.
- Haz click en "MOVE JOINTS, SET TO CURRENT" y salva esta posición como "posición cero".
- En la pestaña "JOINTS" haz click en el botón "SELECT SAVED POSITION".
- Escoge la posición que salvaste previamente y haz click en "MOVE JOINTS".

7.2. Practica el modo de aprendizaje del robot.

- Activa el "learning mode" del robot.
- Mueve el robot con la mano a una posición cualquiera.
- Manteniendo el robot en esta posición, haz click en "UPDATE VALUES" (alternativamente, puedes presionar el botón superior del robot).
- Desactiva el "learning mode".
- Ejecuta un movimiento a la posición grabada haciendo click en "MOVE AXES".
- Activa el "learning mode" y coloca el robot en una posición totalmente vertical.
- Salva la posición con el nombre "vertical" haciendo click en "UPDATE VALUES".
- Selecciona esta posición y ejecútala haciendo click en "MOVE AXES".

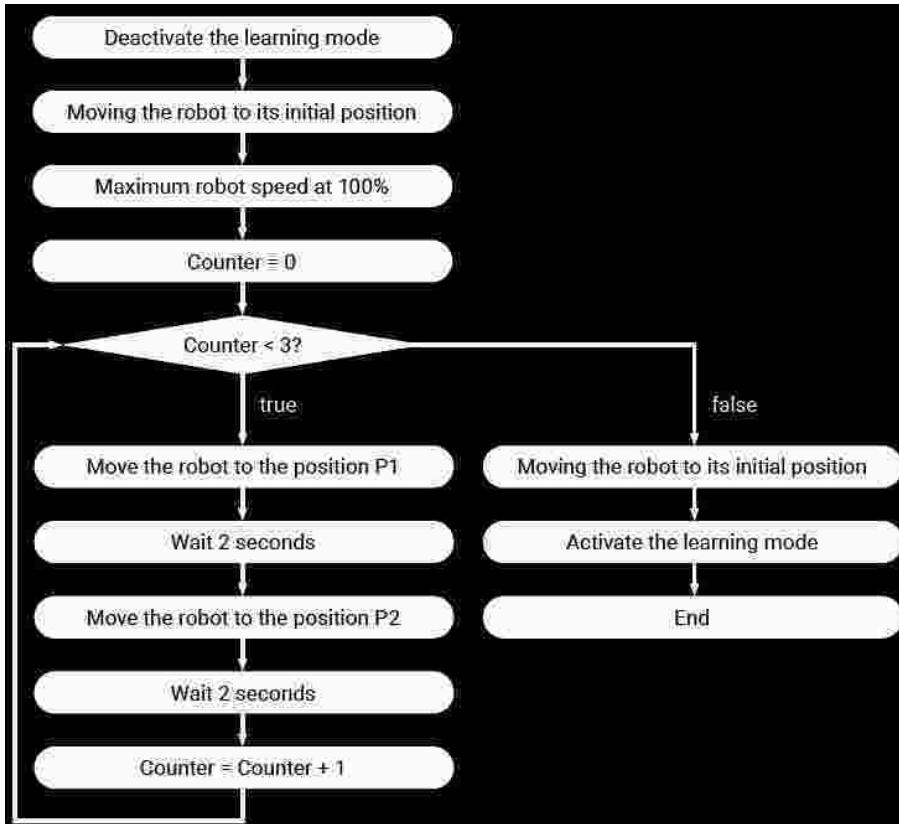
7.3. Programación básica de movimientos con Blockly.

- Introduce el siguiente código, ejecútalo y observa el movimiento del robot:



Haz click en el botón de "import/export" para guardar el código en el robot.

- Desarrolla un código que ejecute el siguiente programa (observa que necesitas el bloque correspondiente a la activación/desactivación del modo de aprendizaje):



7.4. Programación del Niryo One con Python.

- Modifica el código de la sección 6.4 para que el robot se mueva entre las posiciones P1 y P2 4 veces seguidas.
- Coloca un objeto ligero al alcance del robot y desarrolla un script Python para que lo agarre, lo desplace y lo suelte en un lugar cercano predeterminado.

7.5. Resolución del problema cinemático directo para el Niryo One.

- Considerando las dimensiones del robot descritas en la figura 1, obtén la Matriz de Transformación Homogénea para la posición cero del robot, los ejes helicoidales de las 6 articulaciones, y desarrolla un código que resuelva el producto de matrices exponenciales para determinar la posición y orientación del elemento terminal a partir de las coordenadas de las articulaciones.
- La **efectividad** de un robot depende de su **precisión** y **repetitividad**. La precisión en este contexto podemos entenderla como una medida de la habilidad del robot para alcanzar una posición y orientación predeterminadas. La repetitividad es la capacidad del robot para reproducir el mismo valor, independientemente de que este sea o no muy exacto.
- Desarrolla un código que mueva el robot cíclicamente entre un mínimo de 3 posiciones bien separadas unas de las otras. En cada posición haz una pausa, lee tanto las coordenadas de las articulaciones como

las coordenadas cartesianas del punto central del elemento terminal, y los correspondientes ángulos que definen su orientación, almacena los valores de un archivo, compáralos con los valores teóricos (sacados del código que resuelve el problema cinemático directo) y discute los resultados en función de la precisión y repetitividad del robot. Haz al menos 15 repeticiones para tener un poco de estadística. Repite el proceso utilizando al menos 3 conjuntos diferentes de posiciones. Puedes hacer el programa tanto en Blockly como en Python. Si lo resuelves en Blockly, intenta después hacerlo en Python.

- Propón un análisis para los resultados anteriores y preséntalos gráficamente. Puedes utilizar los métodos de análisis que aprendiste en la asignatura de estadística y algoritmia para cuantificar la efectividad del robot.

8. Apéndice

8.1. Código para resolución del problema cinemático directo con Niryo One

```

#!/usr/bin/env python
#
# By Angel.Pineiro at usc.es
# Version April, 2021
#
# EJEMPLO DE USO: python CinematicaDirectaNiryo.py -j '90 90 90 90 90 90'
#

import numpy as np
import optparse

desc="Resolución del problema cinemático directo para el Robot Niryo One."

def VecToso3(omg): # Convierte un vector de 3 componentes en una matriz antisimétrica
    return np.array([[0, -omg[2], omg[1]], [omg[2], 0, -omg[0]], [-omg[1], omg[0], 0]])

def VecTose3(V): # convierte un vector giro o eje helicoidal en matriz 4x4 se3
    return np.r_[np.c_[VecToso3([V[0], V[1], V[2]]), [V[3], V[4], V[5]]], np.zeros((1, 4))]

def so3ToVec(so3mat): # extrae un vector de 3 componentes de una matriz antisimétrica so3
    return np.array([so3mat[2][1], so3mat[0][2], so3mat[1][0]])

# convierte matriz se3 en una matriz de transformación homogénea a través de la exponencial
def MatrixExp6(se3mat):
    se3mat = np.array(se3mat) # vector giro en representación matricial se3 (4x4)
    v=se3mat[0: 3, 3] # extraemos el vector v*theta (velocidad lineal)
    omgmattheta=se3mat[0: 3, 0: 3] # extraemos omega*theta en forma matricial 3x3 (so3)
    omgtheta = so3ToVec(omgmattheta) # lo pasamos a forma vectorial

    if (np.linalg.norm(omgtheta))<1.e-6: # en el caso de que no haya giro (omega despreciable)
        return np.r_[np.c_[np.eye(3), v], [[0, 0, 0, 1]]] # concatena columnas y filas. Sólo traslación

    else: # caso general
        theta = np.linalg.norm(omgtheta)
        omgmat = omgmattheta / theta # omega en forma matricial 3x3 (so3) Normalizada
        # a continuación aplicamos la definición de matriz exponencial que vimos en clase (slide 42)
        G_theta=np.eye(3)*theta+(1-np.cos(theta))*omgmat+(theta-np.sin(theta))*np.dot(omgmat,omgmat)
        R=np.eye(3)+np.sin(theta)*omgmat+(1.-np.cos(theta))*np.dot(omgmat,omgmat)
        return np.r_[np.c_[R,np.dot(G_theta,v)/theta],[[0, 0, 0, 1]]]

def R2Euler(R):
    sy=np.sqrt(R[0,0]*R[0,0]+R[1,0]*R[1,0])
    singular=sy<1.e-6
    if not singular:
        x=np.arctan2(R[2,1],R[2,2])
        y=np.arctan2(-R[2,0],sy)
        z=np.arctan2(R[1,0],R[0,0])
    else:
        x=np.arctan2(-R[1,2],R[1,1])
        y=np.arctan2(-R[2,0],sy)
        z=0.
    return np.array([x,y,z])

def main():
    parser = optparse.OptionParser(description=desc, version='%prog version 1.0')
    parser.add_option('-j', '--joints', help='Coordenadas de las articulaciones', action='store')
    parser.set_defaults(joints='0 0 0 0 0 0')
    options, arguments = parser.parse_args()
    ****
    joints=str(options.joints).split()
    t=[]

```

```

for i in range (0,6,1): t.append(np.deg2rad(np.float(joints[i])))
#for i in range (0,6,1): t.append((np.float(joints[i])))
print("Coordenadas en radianes", np.round(t,5))

# Definimos ejes de rotación
w=[]
w.append(np.array([0,0,1]))
w.append(np.array([0,-1,0]))
w.append(np.array([0,-1,0]))
w.append(np.array([1,0,0]))
w.append(np.array([0,-1,0]))
w.append(np.array([1,0,0]))

# Definimos los eslabones
scalefactor=0.001 # por si quiero los resultados en otras unidades
L=np.array([103.0, 80.0, 210.0, 30.0, 41.5, 180.0, 23.7, -5.5])*scalefactor

# Definimos los vectores que van de cada eje al siguiente
q=[]
q.append(np.array([0,0,L[0]]))
q.append(np.array([0,0,L[1]]))
q.append(np.array([0,0,L[2]]))
q.append(np.array([L[4],0,L[3]]))
q.append(np.array([L[5],0,0]))
q.append(np.array([L[6],0,L[7]]))

# Calculamos los ejes de giro y vectores posición en la configuración final del robot
qs=[]; ws=[]
qs.append(np.array(q[0]))
ws.append(np.array(w[0]))
for i in range(1,6,1):
    ws.append(np.array(w[i]))
    qs.append(np.array(q[i])+qs[i-1])

# Calculamos las velocidades lineales para construir los ejes helicoidales
vs=[]; Si=[]
for i in range(0,6,1):
    vs.append(np.cross(qs[i],ws[i]))
    Si.append(np.r_[ws[i],vs[i]])

M=np.array([[1,0,0,L[6]+L[5]+L[4]], [0,1,0,0], [0,0,1,L[0]+L[1]+L[2]+L[3]+L[7]], [0,0,0,1]])
T=np.eye(4)
for i in range(0,6,1):
    T=np.dot(T,MatrixExp6(VecTose3(Si[i]*t[i])))
T=np.dot(T,M)
print("\nMatriz de transformación homogénea: \n", np.round(T, 3))
print("\nCoordenadas (x,y,z) del TCP: ", np.round(T[0: 3, 3],3))
R=(np.round(T[0: 3, 0: 3],3))
print("\nÁngulos de Euler en radianes", R2Euler(R))
print("\nÁngulos de Euler en grados:")
print(np.array([np.rad2deg(R2Euler(R)[0]), np.rad2deg(R2Euler(R)[1]), np.rad2deg(R2Euler(R)[2])]))

if __name__=="__main__":
    main()

```

8.2. Ejemplo de código para mover el Robot cíclicamente entre 3 posiciones

```

#!/usr/bin/env python
#
# By Angel.Pineiro at usc.es
# Version April, 2021
#
from niryo_one_python_api.niryo_one_api import *

```

```
import rospy
import time
import numpy as np

rospy.init_node('niryo_one_run_python_api_code')

n= NiryoOne()

joints=[[0,0,0,0,0,0],[30,30,30,30,30,30],[-30,-30,-30,-30,-30,-30]]

jr=[]
for i in range (0,3,1):
    aux=[]
    for j in range (0,6,1):
        aux.append(np.deg2rad(joints[i][j]))
    jr.append(aux)

pos=[]; ang=[]

for k in range(0,10,1):
    for i in range (0,3,1):
        n.move_joints(jr[i])
        time.sleep(1)
        aux=n.get_arm_pose().position
        print("\npos= ", aux)
        pos.append([aux.x, aux.y, aux.z])
        aux=n.get_arm_pose().rpy
        print("\nang= ", aux)
        ang.append([aux.roll, aux.pitch, aux.yaw])

np.savetxt('pos.csv', pos, delimiter=',')
np.savetxt('ang.csv', ang, delimiter=',')

n.activate_learning_mode(1)
```