Práctica de Laboratorio N° 3

Objetivos principales

1. Implementar clases y objetos especiales para la representación gráfica de manipuladores robóticos.
2. Obtener parámetros para modelado de manipuladores robóticos utilizando el algoritmo de D-H.

Objetivos secundarios

1. Aprender sobre las diferentes formas de importación de módulos en Python.
2. Aprender a utilizar el módulo `teach` del toolbox desde un terminal de Python.
3. Aprender a generar un plot interactivo de matplotlib en JupyterNotebooks.

Importación de los módulos necesarios para los ejercicios

En Python, es necesario importar explícitamente los paquetes que vamos a utilizar. Para nuestros ejercicios, necesitamos el toolbox de robótica y algunos otros módulos estándar. Los paquetes se importan usando la palabra reservada import. A continuación, veremos algunas formas de hacerlo.

```
from roboticstoolbox import * # 1
from roboticstoolbox import DHLink # 2
import roboticstoolbox as rtb # 3
```

1. Esto significa "importar todo lo que haya en el módulo roboticstoolbox" al entorno. Utilizando el asterisco `*` (wildcard), todas las clases y funciones del módulo se importan en el script y se pueden acceder directamente sin usar el nombre del paquete.
2. Esto significa "importar solo la clase DHLink del paquete roboticstoolbox", permitiendo un control más fino sobre las importaciones y especificando su origen. La desventaja es que, al usar muchas clases, la sección de importaciones se extiende mucho.
3. Esto significa "importar todo el módulo roboticstoolbox y asignarle el alias rtb". Es común abreviar los nombres de los módulos para evitar escribir el nombre completo cada vez que se llama a una clase o función del módulo.

Cabe destacar que, en los casos 1 y 2, los componentes del módulo se importan al nivel superior del script, permitiendo invocarlos directamente sin necesidad de especificar su origen. Por ejemplo, se puede llamar directamente a `fkine()`. En el caso 3, es necesario utilizar el alias, por ejemplo, `rtb.fkine()`.

Uso de la función `teach` en el toolbox

Se utiliza para interactuar gráficamente con el modelo de un robot a través de un entorno de simulación. Este comando abre una interfaz gráfica donde puedes mover las articulaciones del robot manualmente usando controles deslizantes o *sliders*.

Para nuestro caso de uso, esta interfaz nos va a permitir interactuar y observar cómo el movimiento de cada una de las articulaciones del robot afecta la pose del efector final, proporcionando una forma intuitiva de aprender sobre la configuración y el control del robot.

Lastimosamente esta funcionalidad no esta soportada a través del uso de Jupyter Notebooks y se debe acceder a través de una consola de Python. A continuación se muestran unas instrucciones resumidas.

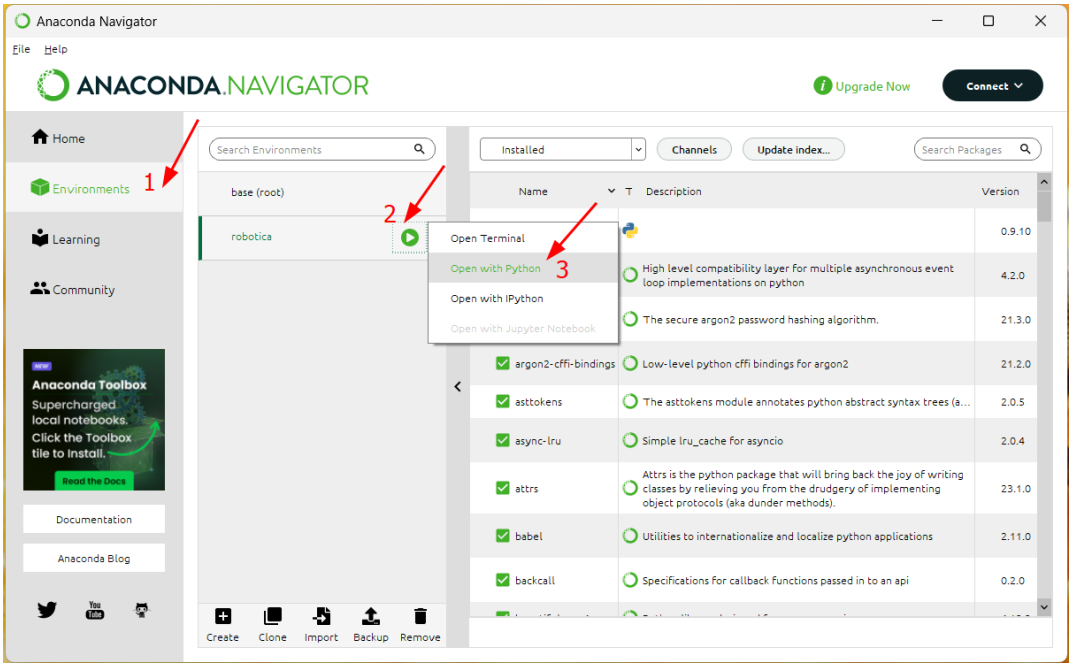

Figura 1. Abrir un terminal Python desde Anaconda

1. Ir a la pestaña de `Environments`.
2. Click en el ícono verde a la derecha del environment que creamos en la practica anterior.
3. Click en `Open with Python`.

Esto nos va a llevar a un terminal de Python listo para usar. La terminal vacía debería verse parecida a la figura 2.

Hay que tener en cuenta que hay que realizar todos los `import` necesarios para que el `roboticstoolbox` se encuentre disponible en el entorno. Para esto, procedemos a escribir, línea por línea los siguientes imports:

```
import roboticstoolbox as rtb # 1
import numpy as np # 2
puma = rtb.models.DH.Puma560() # 3
estado_inicial = np.zeros(puma.n) # 4
```

```
puma.teach(estado_inicial) # 5
```

1. Importar `roboticstoolbox` con el alias `rtb`.
2. Importar `numpy` con el alias `np`.
3. Crear una instancia del robot Puma560 y llamar a la variable `puma`.
4. Crear una lista inicializada en cero, es decir, que todos sus elementos contentan el número cero, y con cantidad de elementos igual a `puma.n`, que es la cantidad de articulaciones que posee el robot. En este ejemplo, obtendremos una lista de 6 ceros: `[0, 0, 0, 0, 0, 0]`.
5. Invocar el método `teach` pasándole como argumento el estado inicial de las articulaciones, que nosotros convenientemente inicializamos en cero.

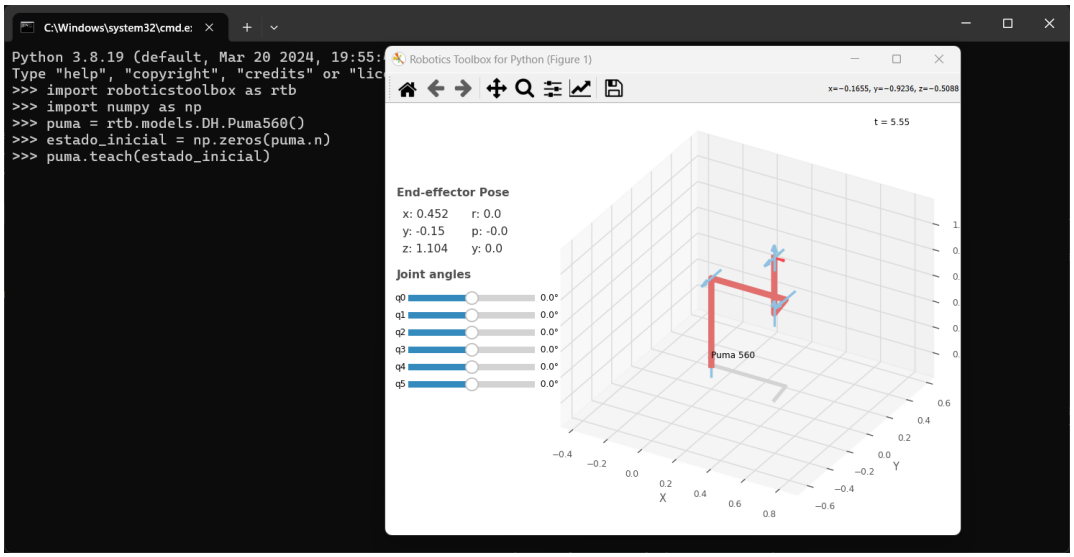

Figura 2. Abrir un terminal Python desde Anaconda

Ejercicios

Ejercicio 1

Utilizando las clases *DHLink* y *SerialLink* y sus respectivos métodos construya y represente en forma gráfica el manipulador cuyos parámetros de D-H se muestran en la tabla 1. Luego compare su resultado con la figura 1.

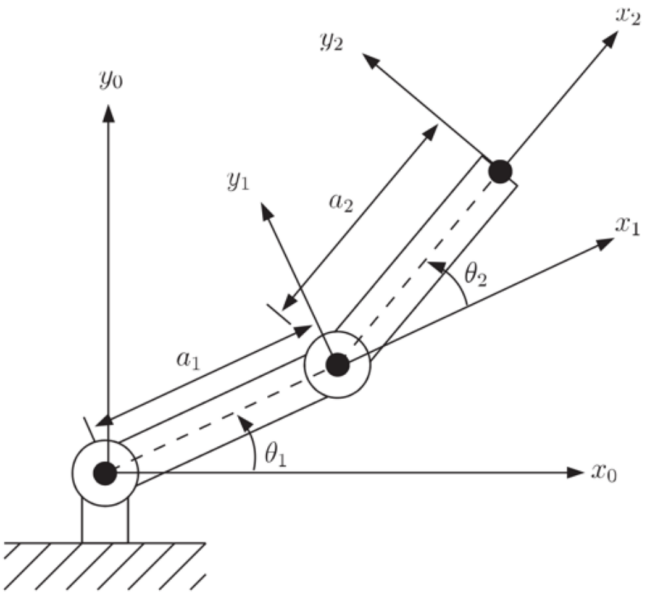

Figura 3. Robot de 2GDL

Tabla 1. Configuración de robot de 2GDL

Articulación	θ	d	a	α
1	q_1	0	1	0
2	q_2	0	1	0

Ejercicio 2

Utilizando las clases *DHLink* y *SerialLink* y sus respectivos métodos construya y represente en forma gráfica el manipulador cuyos parámetros de D-H se muestran en la tabla 2. Luego compare su resultado con la figura 2.

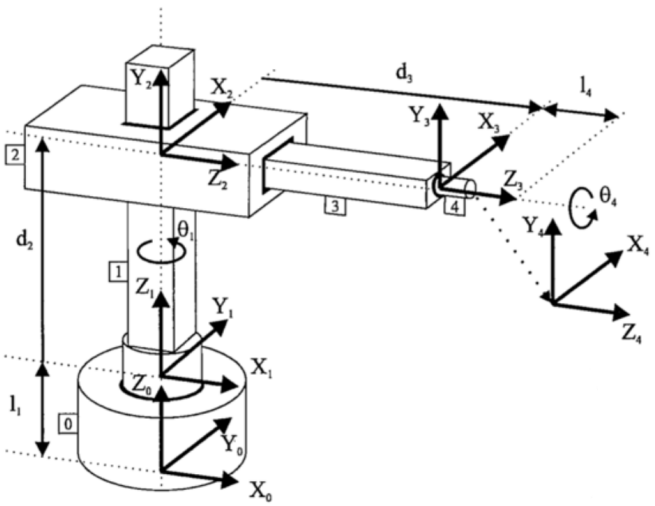

Figura 4. Robot de 4GDL

Tabla 2. Configuración de robot de 4GDL

Articulación	θ	d	a	α	Σ
1	q_1	l_1	0	0	0

Articulación	θ	d	a	α	Σ
2	90	d_2	0	90	0
3	0	d_3	0	0	0
4	q_4	l_4	0	0	0

Ejercicio 3

Siga las instrucciones:

- Invoque el modelo del manipulador FANUC10L utilizando el comando mdl_fanuc10l.
- Identifique la clase correspondiente al robot en el workspace y obtenga la lista de parámetros de D-H correspondiente al manipulador.
- Utilizando el método plot represente el robot en forma gráfica con todas las articulaciones en la posición cero.
- Experimente con el método teach para interactuar con la representación gráfica.

Ejercicio 4

Siga las instrucciones:

- Invoque el modelo del manipulador Cobra600 utilizando el comando mdl_cobra600.
- Identifique la clase correspondiente al robot en el workspace y obtenga la lista de parámetros de D-H correspondiente al manipulador.
- Utilizando el método plot represente el robot en forma gráfica con todas las articulaciones en la posición cero.
- Experimente con el método teach para interactuar con la representación gráfica.