Laboratorio No. 05
Pincher Phantom X100 - ROS Humble - RVIZ

Figura 1: Robot PhantomX

1. Objetivos

- Crear todos los *Joint Controllers* con ROS para manipular servomotores Dynamixel AX-12 del robot *Phantom X Pincher*.
- Manipular los tópicos de estado y comando para todos los *Joint Controllers* del robot *Phantom X Pincher*.
- Manipular los servicios para todos los *Joint Controllers* del robot *Phantom X Pincher*.
- Conectar el robot *Phantom X Pincher* con Python usando ROS 2.

2. Requisitos

- Ubuntu versión 22.xx preferible 22.04 LTS con ROS Humble.
- Espacio de trabajo para *colcon build* correctamente configurado.
- Paquetes de Dynamixel Workbench. `https://github.com/labsir-un/ROB_Intro_ROS2_Humble_Phantom_Pincher_X100.git`
- Paquete del robot Phantom X: `https://github.com/labsir-un/ROB_Intro_ROS2_Humble_Phantom_Pincher_X100_RVIZ.git`.
- Python.
- Un (1) manipulador Phantom X Pincher con su entorno de trabajo.

3. Ejercicio en el laboratorio

*Cada grupo tendrá a cargo un robot **Phantom X Pincher**. Siga las indicaciones del laboratorista y profesores para hacer buen uso de los robots del laboratorio.*

Mediciones:

- Establezca las longitudes de eslabón para cada articulación del robot *Phantom X Pincher*, para este proceso apóyese en un **CALIBRADOR**. Recuerde que la longitud de eslabón es la mínima distancia que conecta dos juntas consecutivas. Genere un diagrama como el presentado en la figura 2 con los datos medidos.
- Es posible que su grupo deba trabajar con una versión nueva del Phantom X Pincher, la cual varía ligeralemente su geometría. Asegúrese de representar adecuadamente este cambio en el diagrama de longitudes del robot.

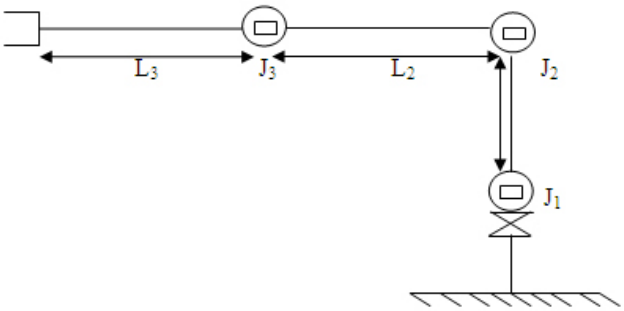

Figura 2: Ejemplo de diagrama.

Análisis:

- Con las dimensiones medidas obtenga los parámetros DH del robot *Phantom X Pincher*.
- Genere diagrama del robot con las tablas de parámetros articulares (Utilice algún software de ilustración).

ROS 2:

- Con base en la documentación de los motores Dynamixel en ROS, cree un script que publique a los tópicos y llame a los servicios correspondientes para realizar el movimiento de cada una de las articulaciones del manipulador (*waist, shoulder, elbow, wrist*). La lógica del script puede ser la siguiente:
  - Se debe realizar el movimiento entre dos posiciones angulares características: una de home y otra objetivo.
  - El movimiento de las articulaciones debe realizarse de forma secuencial iniciando por la articulacion de la base, agregue una pequeña espera entre cada movimiento para facilitar la grabación de videos de demostración.

Conexión con Python:

- Cree un *script* que permita publicar en cada tópico de controlador de articulación, se deben validar los límites articulares de cada junta.
- Cree un *script* que permita suscribirse a cada tópico de controlador de articulación, el *script* debe retornar la configuración de 5 ángulos en grados.

Python + ROS + Toolbox:

- Cree un código en Python que envíe la posición en ángulos deseada a cada articulación del robot utilizando las herramientas de ROS + Dynamixel, el programa deberá graficar la configuración del robot usando las herramientas del *toolbox*, esta configuración deberá coincidir con la obtenida en el robot real.

- Pruebe las siguientes poses generadas a partir de los valores articulares de $q1$, $q2$, $q3$, $q4$, $q5$ (Recuerde que los valores de configuración se toman respecto a home, para el cual todos los valores articulares son cero):
  1. 0, 0, 0, 0, 0.
  2. 25, 25, 20, -20, 0.
  3. -35, 35, -30, 30, 0.
  4. 85, -20, 55, 25, 0.
  5. 80, -35, 55, -45, 0.

  Cuide que las poses no interfieran con los límites articulares o algún objeto en el espacio de trabajo. Puede ajustar el valor de los ángulos de las poses para esquivar algún obstáculo.
- Cree una interfaz de usuario, o HMI, donde se muestre al usuario lo siguiente:
  1. Nombres, logos y datos de los integrantes del grupo.
  2. Imagen perspectiva de la posición actual del manipulador con la ultima posición enviada.
  3. Opción para seleccionar 1 de las 5 poses y enviarlas al manipulador.
  4. Valores de los valores articulares reales de cada motor.
  5. Imagen perspectiva de la posición actual del manipulador con los valores articulares.

  Adicionalmente, la interfaz gráfica debe organizarse en varias pestañas con, al menos, las siguientes funcionalidades:
  - **Pestaña de control en espacio articular mediante deslizadores:** Se debe desarrollar una interfaz gráfica que permita mover el robot en el espacio de las configuraciones (espacio articular), de manera que cada una de las articulaciones del robot pueda moverse haciendo uso de un *slider*. Cada *slider* debe estar ajustado de forma que prevenga choques, evitando superar los límites físicos del robot.
  - **Pestaña de ingreso numérico articular:** En otra pestaña se debe tener la posibilidad de mover cada articulación a un valor determinado por el operador del robot. El operador deberá insertar el valor en grados al cual desea que se mueva la articulación, respetando igualmente los límites articulares establecidos.
  - **Pestaña de control en el espacio de la tarea:** En otra pestaña se debe poder mover el robot en el espacio de la tarea. Por medio de *sliders* se debe tener la posibilidad de mover el TCP del robot a lo largo de los ejes X, Y y Z individualmente y, además, poder rotar el TCP del robot alrededor de estos mismos ejes.
  - **Pestaña de visualización en RViz:** En otra pestaña se debe visualizar el modelo del robot en RViz. Los movimientos presentados en el visualizador deben imitar en tiempo real los movimientos de posición y orientación del manipulador real.
  - **Pestaña de visualización numérica de la pose cartesiana:** Se debe agregar una pestaña en la cual se muestre tanto la posición en X, Y y Z del TCP del robot, como la orientación del TCP en ángulos *Roll-Pitch-Yaw* (RPY). Estos valores deben actualizarse en tiempo real de acuerdo con el estado del manipulador.

Entrega

1. **Forma de trabajo:** Grupal de 2 personas. **Importante:** Cada integrante deberá poner la URL del repo creado.
2. **Entregables: Se deberá crear un repositorio en GitHub con:**
   - Descripción detallada de la solución planteada.
   - Diagrama de flujo de acciones del robot utilizando la herramienta *Mermaid*.
   - Plano de planta de la ubicación de cada uno de los elementos.
   - Descripción de las funciones utilizadas.
   - Código del script utilizado para el desarrollo de la práctica.
   - Vídeo del brazo alcanzando cada posición solicitada.
   - Vídeo demostración de uso de la interfaz de usuario.
   - Los videos deben comenzar con la introducción oficial del laboratorio LabSIR Intro LabSIR.
   - Gráfica digital de las poses comparándola con la fotografía del brazo real en la misma configuración.
3. **Fecha de entrega:** según actividad en Moodle.