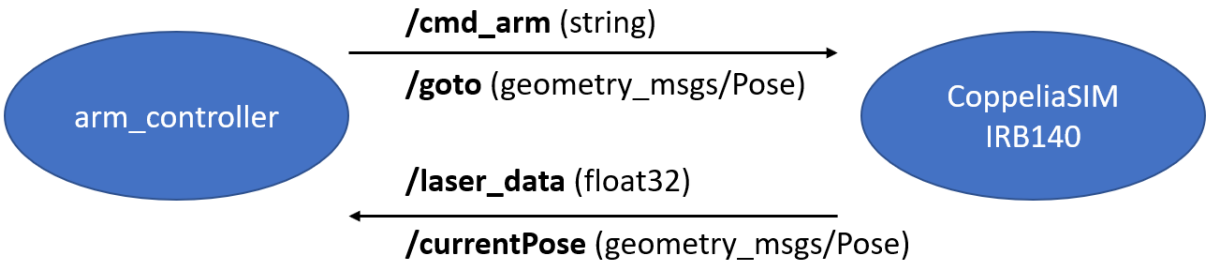
Control y Programación de Robots

Ejercicio Evaluable II

CoppeliaSim + ROS2

En este entregable se pide implementar una escena de CoppeliaSim con un robot IRB140, dotado de un puntero láser colocado en su efector final y apuntando hacia el suelo. El objetivo será comandar los movimientos del robot como si se dispusiera de una consola de mando (ver imagen), cuyos comandos de movimiento procedan de un nodo ROS2. Para ello, crea un nuevo nodo ROS2 llamado "arm_controller" que permita operar el brazo manipulador IRB140 de CoppeliSim desde tu teclado (asignando diferentes teclas para las funciones que tiene que realizar). Concretamente las teclas a asignar son:

Tecla	Topic ROS2	Tipo	Mensaje	Funcionalidad
q, w	/cmd_arm	std_msgs/msg/String	"X+" , "X-"	Incrementar o decrementar en 0.1m la posición del target en el eje X.
a, s	/cmd_arm	std_msgs/msg/String	"Y+" , "Y-"	Incrementar o decrementar en 0.1m la posición del target en el eje Y.
z, x	/cmd_arm	std_msgs/msg/String	"Z+" , "Z-"	Incrementar o decrementar en 0.1m la posición del target en el eje Z.
h	/cmd_arm	std_msgs/msg/String	"home"	Colocar el robot en la posición de home, la cual corresponde a una configuración vertical del mismo (sin importar la orientación).
p	/currentPose	geometry_msgs/msg/Pose	TargetPose	Guardar en una variable de tu nodo, la pose actual del target del robot (posición y orientación). Para ello, el robot debe publicar continuamente la pose del target en el topic /currentPose.
g	/goto	geometry_msgs/msg/Pose	GoalPose	Mover el target del robot a la pose especificada, la cual corresponderá a la última pose guardada con la opción "P".
	/laser_data	std_msgs/msg/Float32	Laser [m]	El robot debe enviar continuamente las medidas del láser que dispone el brazo IRB140.



IMPORTANTE: El robot debe subscribirse a los topics **/cmd_arm** y **/goto** para realizar las acciones requeridas, y publicar en los topics **/currentPose** y **/laser_data** de forma continuada.

LECTURA DEL TECLADO: La lectura de caracteres desde teclado no es algo trivial, especialmente cuando no queremos pulsar la tecla "enter" tras cada comando introducido. A continuación, se facilita un snipped de código que debe ser usado dentro del nodo "arm_controller" para leer el teclado. Este código no está completo, y deberá ser completado por el alumno para que, una vez leído un carácter por teclado, se envíe el comando apropiado al simulador.

```
// Set console raw mode. To avoid pressing enter after each character
system("stty raw");

// Read 1 char
char input = getchar();
switch (input)
{
    case 'q':
        RCLCPP_INFO(this->get_logger(), "Detected q key\r\n");
        …
        …
        break;
    case 'w':
        RCLCPP_INFO(this->get_logger(), "Detected w key\r\n");
        …
        …
        break;

    case 0x20:
        RCLCPP_INFO(this->get_logger(), "Detected SPACE = Exit");
        …
        …
        //restore the console
        system("stty cooked");
        break;
    default:
        // ignoramos caracteres no deseados
        …
        …
}
```

Nota: El poner la consola en modo "raw" no impide terminar el programa con "Ctrl+c". Por ello se recomienda configurar una tecla para salir de este modo. En el script anterior, se ha usado la barra espaciadora "0x20", pero puedes usar cualquier otra tecla.

MOVIMIENTO DEL IRB140 DENTRO DE COPPELIA: Lo más sencillo y directo es usar la función **sim.setObjetPose** que nos proporciona Coppelia para mover el brazo según el comando recibido. No obstante, la función **sim.moveToPose** sabemos que da movimientos más suaves y realistas. El problema de esta segunda función es que no se puede utilizar dentro de un callback, con lo que su uso es algo más complicado**. No siendo obligatorio**, una forma de conseguirlo es una vez recibido un comando de ROS2, en tu callback tan solo escribir los datos necesarios en un "signal" (tal y como hacemos con el láser). En otro thread script aparte del principal, leeremos continuamente este signal para ver si hay que realizar un movimiento, y en caso afirmativo emplearemos nuestro **sim.moveToPose**.

API COPPELIASIM ROS2: A continuación, se detallan las funciones de la API de Coppelia para acceder a topics de ROS2 y/o publicarlos:

Topic Publisher:

```
# Declaration
self.my_publisher = simROS2.createPublisher('topic_name', 'msg_type')

# Publish a msg (where msg is the correct data type)
simROS2.publish(self.my_publisher, msg)
```

Topic Subscriber:

```
# Declaration
self.my_subscriber = simROS2.createSubscription('topic_name', 'msg_type', 'my_callback@func')

# Callback Function
def my_callback(msg):
    print("Executing my_callback function")
```

ENTREGA

- **Escena de CoppeliaSim**

- **Paquete de ROS2** que hayas creado, conteniendo a tu nodo "arm_controller" (siendo lo importante la carpeta "src", el resto pueden ser borradas para que ocupe menos espacio)

- **Vídeo (máx. de 1 min)**, en el que se vea inicialmente la fecha, asignatura y tu nombre (por ejemplo, en un terminal o editor de texto). Tras ello, el vídeo debe mostrar un caso en el que el robot IRB140 realice la siguiente secuencia de movimientos:
  a) Mover el robot con las teclas de movimiento a una pose deseada (la que más te guste, pero que sea distinta de home).
  b) Guardar esa pose con la opción "P" (se debe ver en el terminal las teclas pulsadas y el valor de la pose guardada).
  c) Ir a Home, empleando para ello la tecla "H".
  d) Regresar a la pose guardada, empleando la tecla "G".
  En el vídeo debe visualizarse la simulación de CoppeliSim, el terminal donde escribas los comandos con el teclado (tu nodo), y otro terminal mostrando la pose actual del target (ros2 topic echo …).