Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

ÁREA DE ROBÓTICA

PUESTA EN MARCHA Y DEMOSTRACIÓN DE FUNCIONAMIENTO DEL ROBOT UR3e EN EL ENTORNO ROS

D. Menéndez García, Alejandro
TUTOR: D. Álvarez Prieto Diego

FECHA: (Junio de 2022)

ÍNDICE

INDICE DE FIGURAS

1. INTRODUCCIÓN Y OBJETIVOS

En el campo de la robótica, la robótica colaborativa es un sector emergente en constante crecimiento. Los robots colaborativos son sistemas los cuales permiten ser programados de manera sencilla, capaces de realizar sus funciones junto a los operarios, sin la necesidad de tener en consideración los sistemas de seguridad tradicionales. Estos robots incorporan sistemas de control de fuerza que esquivan los obstáculos. Otra cualidad de los robots colaborativos es que montan sensores de fuerza y de consumo, pudiendo detectar colisiones con el entorno y apagar sus sistemas impidiendo golpear a los operarios.

La robótica colaborativa permite que la automatización robótica sea más accesible para las pequeñas y medianas empresas. Permite la modernización y optimización de sectores donde hasta día de hoy, no había sido posible la llegada de procesos de automatización, lo que implica una mayor flexibilidad y competitividad en la realización de sus diversas tareas.

En este trabajo se va a realizar una puesta en marcha del robot colaborativo UR3e de la empresa Universal Robots. El UR3e de Universal Robots es un robot industrial colaborativo ultraligero y compacto más pequeño de la gama, ideal para tareas de montaje ligeras y bancos de trabajo automatizados. Este compacto robot de sobremesa pesa 11 kg, pero cuenta con una capacidad de carga de 3 kg, una rotación de 360° en todos sus ejes y una rotación infinita en el último eje.

Este robot colaborativo UR3e ha sido instalado recientemente en el laboratorio de robótica 2.B.02 de la EPI y los objetivos de este trabajo es realizar una primera toma de contacto con el robot, aprendiendo a cómo trabajar con él y realizar una demostración de cómo empezar a funcionar con el robot en el entorno de programación de ROS. Todo lo realizado para cumplir este último objetivo va a ser documentado en forma de un manual de usuario que podrá ser utilizado por la siguiente persona que vaya a trabajar con el robot, para que pueda ponerse a trabajar en ROS con el robot más rápidamente.

2. ROBOT COLABORATIVO UR3E

Para entender los objetivos y la escala del trabajo que se va a realizar, hay que conocer definiciones y características sobre el tema que se va a abordar. En este caso es la robótica, específicamente la robótica colaborativa.

2.1. DEFINICIÓN DE ROBOT

Según la RAE (Real Academia Española) un robot es una máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas sólo a las personas. Aunque una definición más exacta, dada por la IFR (Federación Internacional de Robótica) es que un robot es una máquina automática, reprogramable y multifuncional con tres o más ejes, que mediante herramientas o dispositivos especiales puede posicionar, orientar y/o mecanizar materias o piezas, ejecutando trabajos diversos en las diferentes etapas de la producción industrial, según uno o varios programas establecidos ,siendo éstos modificables o adaptables de forma automática, mediante sensores, o de forma manual por el usuario.

2.2. UTILIDADES DE LOS ROBOTS

Hoy en día los robots se han vuelto indispensables en la realización de distintas tareas. Comúnmente, un robot se emplea para sustituir al hombre en la realización de tareas industriales las cuales resultan repetitivas, adaptándolos de manera inmediata y automática a cualquier cambio en la producción. Aunque su uso puede ser adaptado a distintos campos no necesariamente industriales como, por ejemplo:

- **Exploración o transporte de personas**: Donde el tipo de robot empleado son robots móviles, los cuales pueden comprender y navegar sobre su entorno de forma independiente.

- **Asistencia médica**: Donde, por ejemplo, un robot colaborativo puede dar apoyo y ayuda al personal médico.

- **Actos militares**: Donde la robótica militar sería la rama encargada de diseñar y producir robots con el objetivo de ser utilizados en el ámbito militar, como en guerras o en conflictos armamentísticos.

2.3. ROBOTS COLABORATIVOS

Los robots colaborativos o cobots están diseñados para trabajar junto con el operario. A diferencia de la robótica tradicional, son pequeños, ligeros y no necesitan de los mismos sistemas de seguridad, como por ejemplo una jaula de protección. Las ventajas que presenta la robótica colaborativa respecto a la robótica tradicional son:

- **Seguridad y colaboración**: Como se dijo anteriormente, una característica diferencial de la robótica colaborativa respecto a la robótica industrial es que los robots colaborativos pueden ser colocados sin la necesidad de instalar una jaula de protección.

- **Flexibilidad**: Otra ventaja que ofrecen los robots colaborativos gracias a su reducido tamaño es su flexibilidad. Podemos cambiarlos de ubicación y de tarea de forma rápida y sencilla. Poseen la agilidad necesaria para automatizar prácticamente cualquier trabajo manual.

- **Facilidad de programación**: La tecnología patentada que incorporan normalmente los robots colaborativos hace posible que operarios sin experiencia en programación puedan configurar y manejar rápidamente los robots desde un panel de control que posee una interfaz intuitiva con una visualización tridimensional.

- **Rápida instalación**: Normalmente estos robots, al ser de un tamaño inferior a un robot industrial, el tiempo medio necesario para su instalación en el lugar de trabajo

es mucho menor que un robot industrial. El tiempo de instalación medio, por ejemplo, del robot UR3e, según el fabricante es de menos de un día.

2.4. NORMATIVAS Y ESTANDARES DE LOS ROBOTS COLABORATIVOS

Actualmente existen normativas aplicables al uso de robots colaborativos. El organismo ISO ya ha implementado normativas en relación con los robots colaborativos aparte de las normas que tienen que ver con los robots de carácter general (**ISO 13849-1:2008 y ISO 10218-1:2011**). La norma aplicable a los robots colaborativos tiene que ver con los tipos de robots colaborativos que existen y que medidas de seguridad se deben tomar dependiendo de cuál va a ser la función que va a desempeñar el robot colaborativo en la aplicación. Esta norma es la norma **ISO/TS 15066:2016**.

ISO / TS 15066: 2016 especifica los requisitos de seguridad para los sistemas de robots industriales colaborativos y el entorno de trabajo, y complementa los requisitos y la orientación sobre la operación de robots industriales colaborativos dados en **ISO 10218-1** e **ISO 10218-2. ISO / TS 15066: 2016** se aplica a los sistemas de robots industriales como se describe en **ISO 10218-1 e ISO 10218-2**. No se aplica a los robots no industriales, aunque los principios de seguridad presentados pueden ser útiles para otras áreas de la robótica.

Esta normativa, hoy en día, todavía se encuentra en revisión, con lo que la normativa todavía es susceptible a cambios o ampliaciones de ella misma. Dicho esto, con la normativa vigente, en este trabajo, el robot colaborativo UR3e al detenerse cuando existe un movimiento brusco o contacto, se podría deducir según la norma que se está trabajando en una aplicación colaborativa de “Limitaciones de potencia y Fuerza”.

2.5. ESTRUCTURA Y COMPONENTES DEL ROBOT COLABORATIVO UR3e

Como se ha dicho anteriormente el robot colaborativo UR3e con el que se va a trabajar es un robot fabricado por la empresa Universal Robots. Es un modelo industrial especializado en el sistema educativo o para operaciones que requieran de poco peso. Por lo demás, el robot UR3e presenta las mismas prestaciones que un robot más grande.

2.5.1. ESTRUCTURA DEL ROBOT

El robot colaborativo o manipulador UR3e está formado por elementos por tubos y juntas unidos mediante articulaciones, los cuales permiten que haya un movimiento relativo entre cada dos eslabones consecutivos. Este movimiento es producido por los actuadores y el tipo de movimiento de cada articulación puede ser de desplazamiento o de giro. Los elementos básicos que componen el brazo robótico son:

- **Base**: Donde se monta el brazo robótico.
- **Hombro y Codo**: Permiten realizar movimientos amplios.
- **Muñeca 1 y Muñeca 2**: Se encargan de realizar los movimientos más precisos.
- **Muñeca 3**: Donde se instala la pinza o herramienta a la Brida de la herramienta.

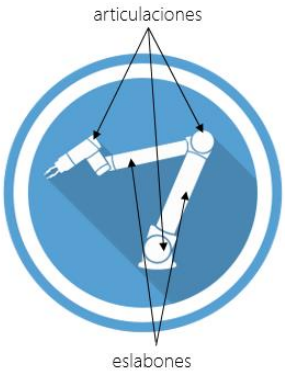


Figura 1: Eslabones y Articulaciones del robot

El robot colaborativo UR3e presenta seis grados de libertad en su estructura. Este número de grados de libertad viene determinado por la suma de las articulaciones que los componen. Los grados de libertad equivalen al número de parámetros independientes que fijan la situación del elemento terminal.

Los elementos y sistemas que están situados internamente dentro de la estructura del robot son:

- **Un subsistema mecánico**.
- **Un subsistema eléctrico**.
- **Un subsistema de actuadores** formado por switches, pistones y motores.
- **Unos subsistemas de accionamiento y sensores** como de proximidad o presión.
- **Un panel de control**: Elemento externo pero que también pertenece a la estructura del robot. Posee internamente sistemas de control tanto en el terminal como en el robot para poder controlar el robot remotamente.

Figura 2: Elementos y Sistemas del Robot

2.5.2. COMPONENTES DEL ROBOT

Dentro del robot colaborativo UR3e y de cualquier otro robot suelen haber ciertos componentes básicos que permiten el correcto funcionamiento del robot. Esos componentes son:

- **Componentes de accionamiento**
    1. Motores eléctricos con reductores
    2. Accionamientos hidráulicos
    3. Accionamientos neumáticos.

- **Sensores**

  Los sensores son dispositivos que permiten a un robot percibir su entorno. Un sensor es un dispositivo que convierte algún fenómeno físico en señales eléctricas las cuales el microprocesador del robot puede leer. La misma propiedad física puede medirse por varios sensores. Algunos sensores presentes en el robot son:

  1. Encoders
  2. Tacómetros
  3. Finales de carrera
  4. Detectores de presión
  5. Detectores inductivos y capacitivos

- **Controladores Digitales**

  Reciben señales de entrada y calcula las señales de salida.

- **Fuente de potencia**

  Es utilizada por unos actuadores independientes los cuales controlan y regulan cada movimiento del manipulador, tanto como si el movimiento es lineal como si es rotacional.

- **Elementos terminales**

  Son las herramientas que se instalan en la muñeca3 del robot. Normalmente estas herramientas suelen ser pinzas, pero depende de la función que se quiera que desempeñe el robot, la herramienta puede tener distintas formas y utilidades no similares a una pinza.

Figura 3: Herramienta instalada en el robot

2.6. FICHA DE ESPECIFICACIONES TÉCNICAS

Tipo de robot	UR3e
Peso	11.1 kg / 24.5 lb
Carga máxima (consulte la sección 4.4)	3 kg / 6.6 lb
Alcance	500 mm / 19.7 in
Rango giro juntas	Rotación infinita de la última junta, ± 360 ° para todas las demás juntas
Velocidad	Todas las juntas de la muñeca: Máx. 360 °/s, Otras juntas: Máx. 180 °/s. Herramienta: Aprox. 1 m/s / Aprox. 39.4 in/s.
Repetibilidad	± 0.03 mm / ± 0.0011 in (1.1 mils)
Espacio necesario	Ø128 mm / 5.0 in
Grados de libertad	6 juntas giratorias
Tamaño de la caja de control (ancho × alto × largo)	460 mm × 445 mm × 260 mm / 18.2 in × 17.6 in × 10.3 in
Puertos de E/S de la caja de control	16 entradas digitales, 16 salidas digitales, 2 entradas analógicas, 2 salidas analógicas
Puertos de E/S de la herramienta	2 entradas digitales, 2 salidas digitales, 2 entradas analógicas
Fuente de alimentación de E/S	24 V 2 A en caja de control y 12 V/24 V 600 mA en herramienta
Comunicación	TCP/IP 1000 Mbit: IEEE 802.3ab, 1000BASE-TX Ethernet socket, MODBUS TCP & EtherNet/IP Adaptador, Profinet
Programación	Interfaz gráfica de usuario PolyScope en pantalla táctil de 12”
Ruido	70 dB(A)
Clasificación IP	IP54
Clasificación de sala blanca	Brazo robótico: ISO Clase 5 Caja de control: ISO Clase 6
Consumo de energía	Aprox. 150 W utilizando un programa típico
Funcionamiento colaborativo	17 funciones de seguridad avanzada. De acuerdo con: EN ISO 13849-1:2008, PLd y EN ISO 10218-1:2011, cláusula 5.10.5
Materiales	Aluminio, plástico PP
Temperatura	El robot puede funcionar en un intervalo de temperatura ambiente de -5-50 °C A una velocidad de junta continuamente elevada, la especificación de temperatura ambiente máxima se reduce.
Fuente de alimentación	100-240 VAC, 47-440 Hz
Cables	Cable entre el robot y la caja de control (6 m / 236 in) Cable entre la pantalla táctil y la caja de control (4.5 m / 177 in)

2.7. CAJA DE CONTROL

En la caja de control se encuentran las entradas y salidas digitales y analógicas de control que entran o salen, así como el resto de la interfaz eléctrica. Los componentes de entrada y salida principales en la caja de control son: la conexión a la red de suministro, la conexión al robot, las E/S de controlador, la E/S de la herramienta y la conexión Ethernet.

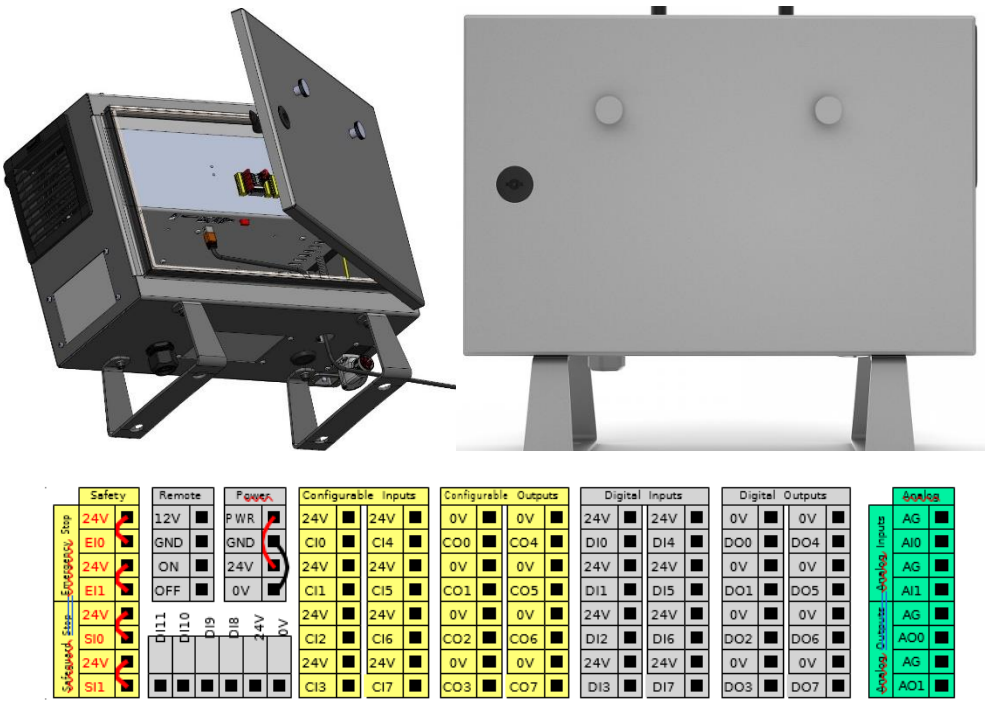


Figura 4: Caja de Control

En la parte inferior interior de la caja de control se encuentran disponibles distintos puertos que permiten conexiones adicionales. Dichos puertos son un puerto para tarjetas SD, puerto Ethernet, dos puertos Usb (uno para Usb 2.0 y otro para Usb 3.0), un Teach Pendant y un puerto Mini. El fusible que también se puede ver debe contar con el certificado UL, tipo enchufable mini con corriente nominal máxima de 10 A y tensión nominal de 32 V.

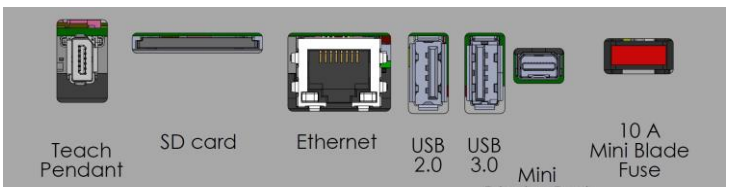


Figura 5: Puertos

2.8. PANEL DE CONTROL

La gran mayoría de robots pueden ser manipulados a través de un panel de control o consola portátil que opera el brazo robótico a través de una pantalla táctil. La Interfaz de usuario gráfica (IGU) instalado en la consola portátil es PolyScope. PolyScope es un visor de C ++ / Python e interfaz de usuario para datos 3D y todo programa para el robot se crea, carga y ejecuta en PolyScope.

Figura 6: Panel de Control

2.9. POLYSCOPE

La interfaz de PolyScope está dividida en:

- **A: Encabezado:** Donde se ofrecen las distintas pestañas con las que interactuar.
- **B: Pie de página:** Con botones para conectarse al robot y conectar los programas ejecutados.
- **C: Pantalla:** Que cambia dependiendo de la pestaña en la que se esté y que permite gestionar y supervisar los movimientos del robot.

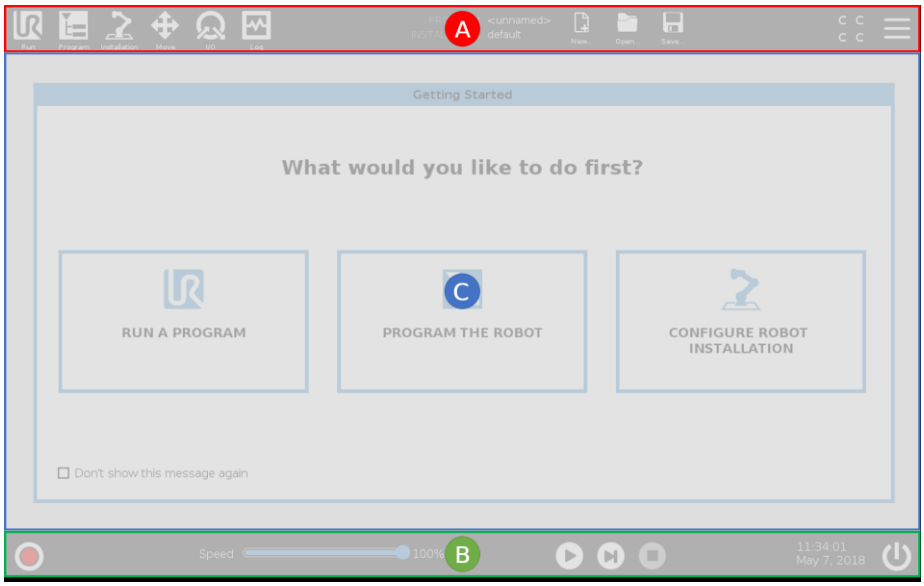


Figura 7: División de la interfaz de PolyScope

2.9.1. INICIALIZACIÓN DEL ROBOT

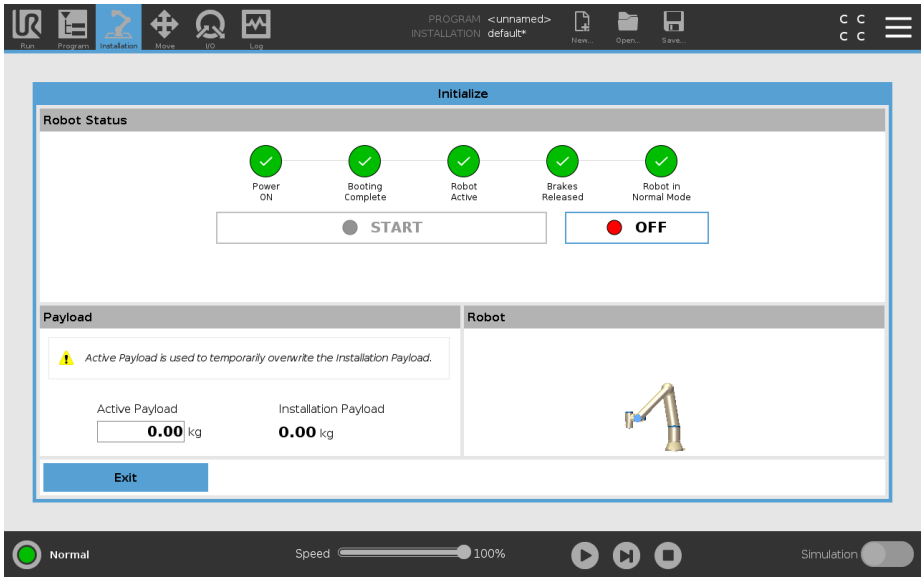


Figura 8: Pestaña de inicialización del robot

En la parte inferior izquierda de la pantalla se indica en qué estado está el brazo robótico. Se diferencian tres estados: Apagado (Color rojo), Inactivo (Color Amarillo), Normal (Color Verde).

2.9.2. PESTAÑA EJECUTAR

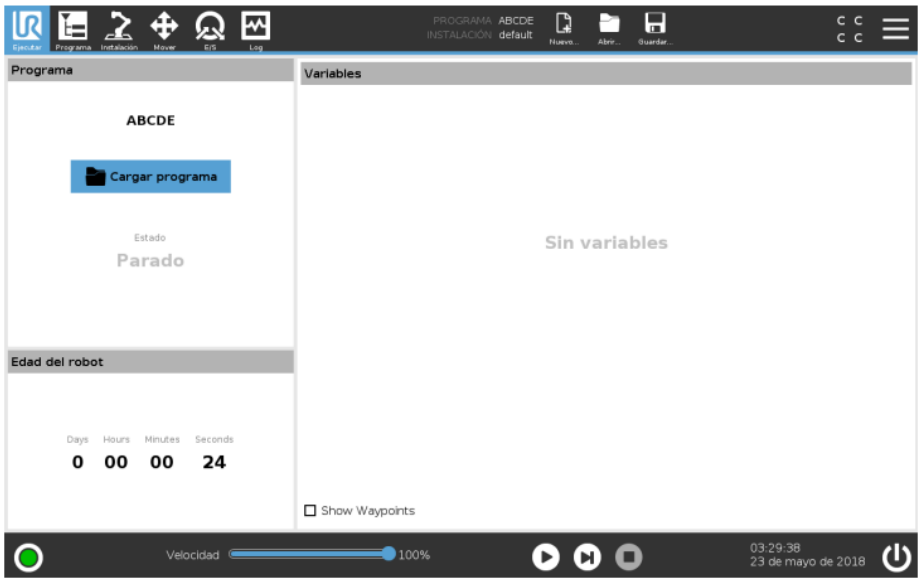


Figura 9: Pestaña Ejecutar

En esta pestaña se puede cargar un programa creado y ejecutarlo siempre que haya una transición del flanco de entrada externa.

2.9.3. PESTAÑA PROGRAMA

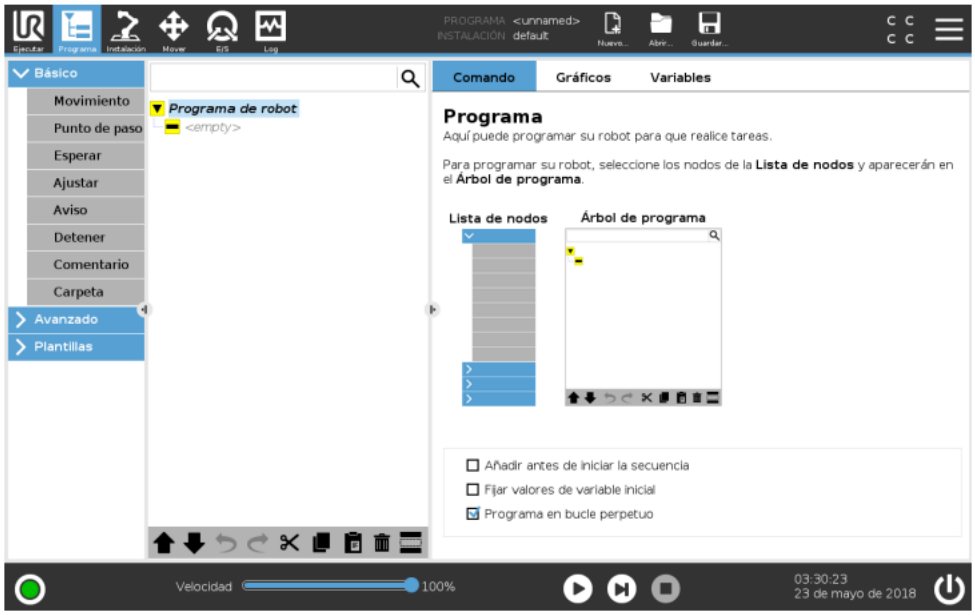


Figura 10: Pestaña Programa

Esta pestaña permite crear programas con el objetivo de controlar el robot. Se ofrecen distintos comandos (básicos y avanzados), que a la vez tienen dentro del mismo comando opciones que configuran el comando. Todos estos comandos añaden nodos de programa al árbol de programa.

Algunos de los comandos básicos son, por ejemplo:

- **Mover**: Permite moverse en una secuencia de distintos puntos y dentro del comando 'Mover' se puede configurar el tipo de movimiento que se quiere que se realice (Mover L, Mover P, Mover J).
- **Esperar**: Espera un tiempo configurable antes de saltar a la siguiente línea de comando.
- **Detener**: Detiene el programa hasta que se le dé permiso para que continue.

Los comandos avanzados ya son comandos que se usan comúnmente en cualquier entorno de programación, como por ejemplo if, bucle while, switch, subprogramas ….

También hay unas plantillas, rutinas ya creadas ofrecidas por el fabricante que realizan funciones que comúnmente desempeñan estos robots.

2.9.4. PESTAÑA INSTALACIÓN

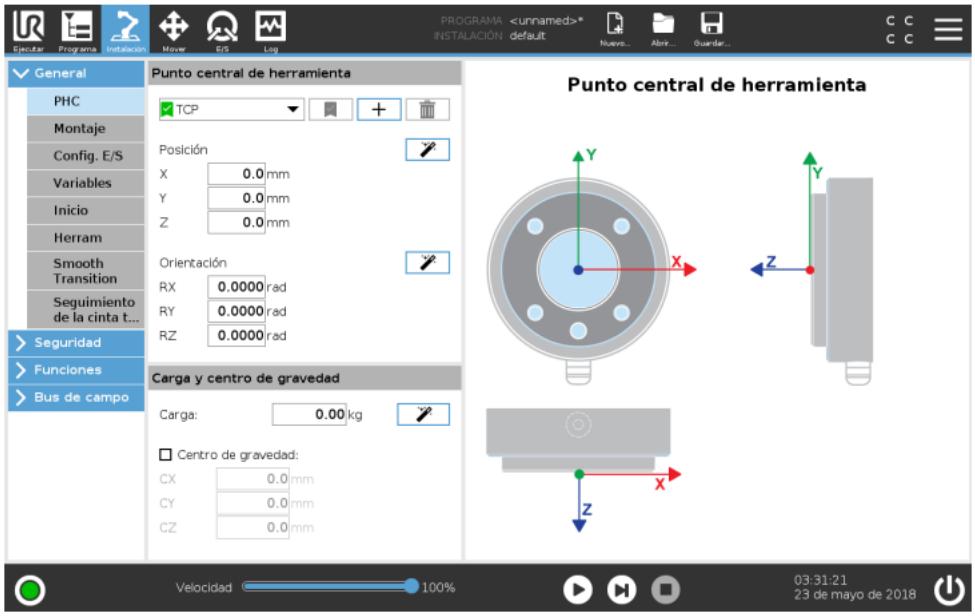


Figura 11: Pestaña Instalación

En la pestaña instalación se pueden configurar los distintos ajustes que afectan al rendimiento general del robot y PolyScope. Te permite configurar apartados como la posición central de la herramienta (PCH) pudiendo variar su posición y orientación por ejemplo.

También en esta pestaña se pueden cambiar ajustes de Seguridad como la potencia, momentos, fuerza máxima con la que trabajará el robot. Otra opción muy útil que ofrece los cambios en los ajustes de seguridad es la opción de crear planos que el robot no puede sobrepasar, lo cual es útil por ejemplo en casos donde el robot esté instalado junto a una pared.

2.9.5. PESTAÑA MOVER

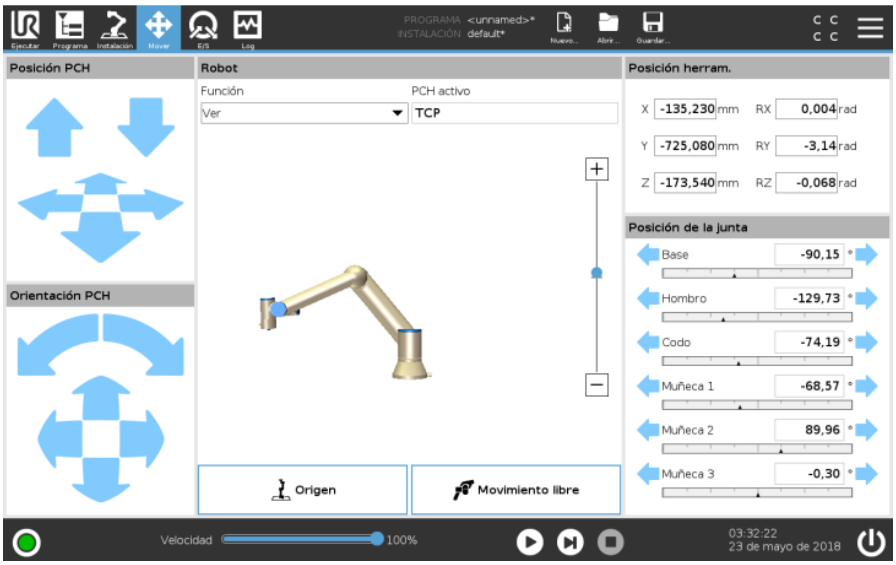


Figura 12: Pestaña Mover

Esta pestaña permite mover el brazo robótico directamente. Permite tanto desplazar o mover la herramienta como mover o desplazar una a una las juntas del robot. La velocidad a la que se mueven las juntas puede ser configuradas en la parte inferior del robot.

2.9.6. PESTAÑA E/S

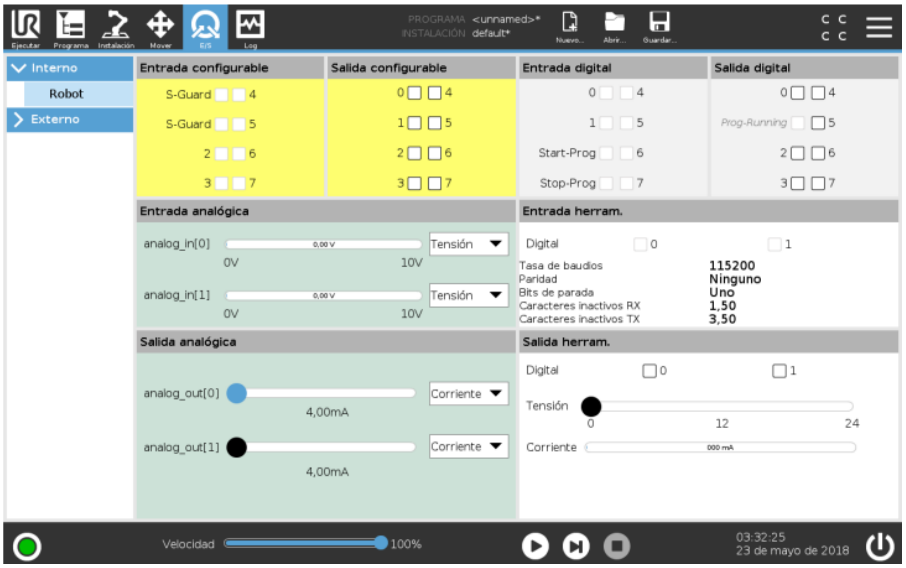


Figura 13: Pestaña E/S

Esta pestaña permite supervisar y ajustar las señales de entrada y salida que proceden o van a la caja de control del robot.

2.9.7. REGISTRO

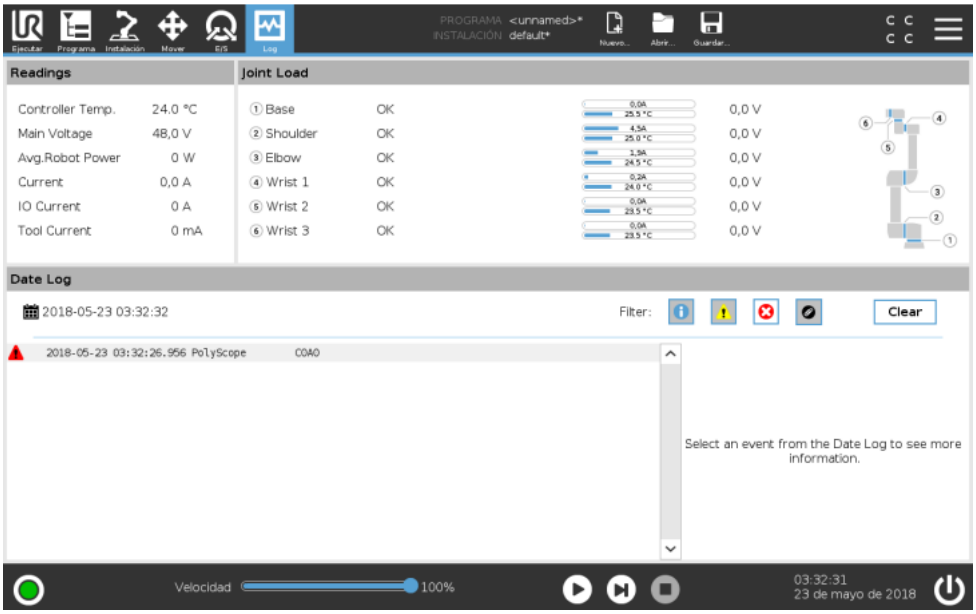


Figura 14: Pestaña Registro

Esta pestaña muestra la información correspondiente al estado actual del robot y de la Caja de Control.

2.9.8. CREAR, ABRIR Y GUARDAR PROGRAMAS

Figura 15: Opciones de Abrir, Crear y Guardar programas

Como el propio nombre indica, estas opciones ofrecen la oportunidad de crear y guardar programas y configuraciones referentes a la instalación del robot.

2.9.9. MENÚ HAMBURGUESA

Este menú desplegable situado arriba a la derecha ofrece las opciones de ayuda, información genérica referente al robot, ajustes donde poder observar por ejemplo la configuración IP del robot y por último la opción de apagar el robot.

3. ROS

3.1. DEFINICIÓN DE ROS

ROS (Robotic Operating System) es un middleware de código abierto y gratuito. ROS está divido en dos partes: una parte corresponde con el sistema operativo 'ros' y la otra con 'ros-pkg' que incluye un conjunto de paquetes, herramientas y librerías las cuales ofrecen la posibilidad de controlar un robot o realizar un conjunto de tareas. De esos conjuntos de paquetes que incluye ROS, están incluidos los entornos de programación de Python y C++ siendo capaz de compilarlos, aunque estén escritos en distintos lenguajes. Durante este trabajo se realizarán programaciones en Python y XML, aunque también se trabajarán con archivos programados en C++, con lo que, se requerirá conocimiento de estos lenguajes de programación.

Normalmente ROS se usa con el apoyo del sistema operativo de Ubuntu porque, aunque como se dijo anteriormente ROS tiene una parte de sistema operativo, este no es capaz de hacer las funciones de gestionar los recursos del hardware, organizar y priorizar los servicios de las aplicaciones del software. ROS también es compatible con los sistemas operativos de Windows, Mac OS X y Debian, pero no está tan optimizado como para Ubuntu, con lo que su uso junto a estos sistemas operativos es menos frecuente.

3.2. VERSIÓN DE ROS

Como ROS es de código abierto y gratuito, este está siendo periódicamente modificado y mejorado. Para este proyecto se ha utilizado la versión Noetic de ROS. Esta

no es la última versión disponible para ROS, porque ROS recientemente recibió una actualización donde se pasó de ROS a ROS2 y la versión de ROS2 que se está usando actualmente es 'Foxy'. La versión Noetic que se está empleando en el proyecto es la última versión disponible para ROS no teniendo en cuenta ROS2.

La versión de ROS Noetic, la cual va a tener soporte hasta el año 2025, requiere de la versión 20.04 del sistema operativo de Ubuntu. Se tiene que descargar la versión ROS Noetic completa, porque te incluyen los simuladores en 3D (Gazebo y RVIZ), los cuales son necesarios para la realización del proyecto.

3.3. INSTALACIÓN DE ROS

Para instalar ROS en la versión Noetic para el sistema operativo de Ubuntu, hay que ir a la página web http://wiki.ros.org/noetic/Installation/Ubuntu y seguir los pasos descritos, los cuales son:

1. Configurar el Pc para que pueda aceptar software de packages.ros.org, escribiendo en el terminal el comando:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Configurar las llaves mediante el comando:

```
sudo apt install curl # if you haven't already installed curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

3. Actualizar los paquetes disponibles:

```
sudo apt update
```

4. Descargar e instalar el paquete de ROS Noetic completo:

```
sudo apt install ros-noetic-desktop-full
```

3.4. ELEMENTOS BÁSICOS DE ROS

- **Nodos**: Los nodos son los elementos encargados de la parte de computación. Un nodo es un archivo ejecutable dentro de un paquete de ROS. La comunicación entre nodos forma un complejo modular que garantiza el correcto funcionamiento de los proyectos creados en ROS.

- **Máster:** El ROS Máster proporciona el registro del nombre y consulta el resto del grafo de computación. Rastrea quien está subscrito a los tópicos y servicios. Esto permite que nodos individuales de ROS se ubiquen entre sí.

- **Tópicos:** Un tópico es el canal de comunicación entre los nodos. El intercambio de información entre nodos se hace a través de tópicos los cuales son buses de datos. Un nodo se encarga de publicar un mensaje en un tópico al que se ha subscrito un segundo nodo para leer los mensajes ahí depositados.

- **Servicios:** Cuando la comunicación entre los nodos mediante mensaje no se puede realizar, ROS permite la comunicación a través de servicios.

- **Rosbags:** Los Rosbags son archivos en donde se guardan los datos de los mensajes publicados por los nodos que interesan para una posterior utilización. Esto es muy útil para por ejemplo guardar los datos proporcionados por los sensores del robot.

4. IMPLEMENTACIÓN DEL PROYECTO

Como se ha explicado anteriormente, el objetivo principal de este proyecto es realizar una primera toma de contacto con el robot, aprendiendo a como trabajar con él y realizar una demostración de cómo empezar a funcionar con el robot utilizando ROS. Otro objetivo o finalidad de este proyecto, es que el siguiente estudiante que vaya a trabajar con el robot pueda utilizar el manual que se ha redactado a la hora de realizar este proyecto y implementar estos primeros pasos de toma de contacto con el robot en menos tiempo pudiendo invertir más tiempo en programar aplicaciones del robot más complejas.

Para controlar el robot UR3e utilizando ROS se va a crear un paquete Moveit que nos permitirá también ver como se comportará el robot en el entorno de simulación donde está situado el robot en la realidad. Para la creación de los elementos del entorno de simulación, se utilizará un macro propio de ROS, los cuales son los Xacro.

Para visualizar el entorno de simulación y la planificación de trayectorias se van a utilizar otras herramientas propias de ROS como Gazebo y RVIZ.

4.1. HERRAMIENTAS USADAS

4.1.1. GAZEBO

Gazebo es un software que efectúa simulación realista tanto de la física e interacciones de los objetos, así como de la dinámica y sensores de diversos robots. Gazebo funciona utilizando el core de ROS, por lo que permite simular el comportamiento de los robots al utilizar los paquetes desarrollados para ROS. Emplearemos Gazebo para comprobar cómo se comportará nuestro robot en el entorno y si serán seguras las trayectorias que le mandemos hacer a nuestro robot.

4.1.2. MOVEIT

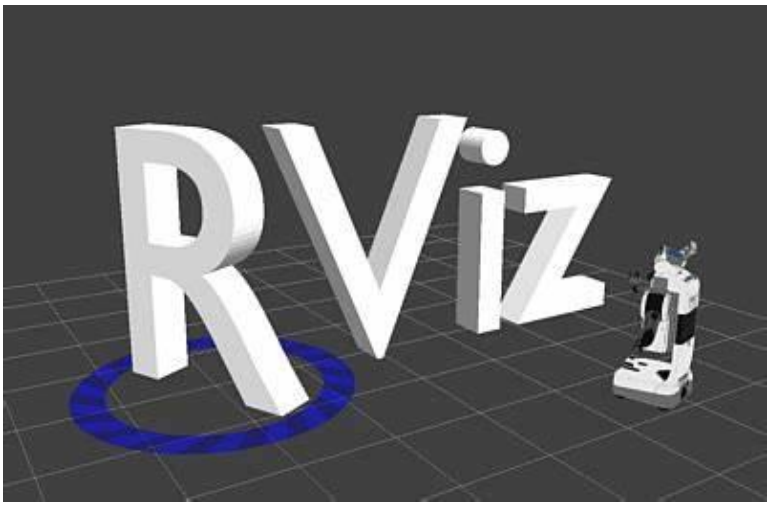
MoveIt! es un software de código abierto para ROS. MoveIt está diseñado para la programación de la manipulación y movimientos de los robots. MoveIt permite crear y simular el entorno de trabajo del robot utilizando objetos de malla 3D diseñados en cualquier programa Gazebo, permitiendo además la interacción entre el robot y este entorno de trabajo.

Con MoveIt se puede planificar el movimiento del robot a cualquier posición teniendo en cuenta la situación actual de los objetos y el robot presentes en el entorno, evitando así colisiones. Pero lo más interesante es que no sólo planifica trayectorias evitando los obstáculos, sino que además permite interactuar con los objetos del entorno, pudiendo coger cualquier objeto del entorno e incluirlo como si fuera parte del robot a la hora de planificar la trayectoria a la posición deseada.

4.1.3. RVIZ

Rviz es otra herramienta de simulación y visualización 3D. El archivo RVIZ del robot se creará cuando se programe el paquete Moveit. Va a permitir crear trayectorias y mandárselas al robot o al archivo de Gazebo para que represente como sería su funcionamiento en un modo de simulación. Es altamente configurable y posee distintos

tipos de plugin y formatos de visualización. Se puede usar para mostrar lecturas de sensores, datos devueltos por la visión estereoscópica (Cloud Point), hacer SLAM (localización y mapeo simultáneo) evitando obstáculos, etc.



4.2. ENTORNO DE TRABAJO

El entorno de trabajo en el que se va a trabajar va a ser la línea de comandos de linux o el intérprete de comandos bash. Con él se va a poder acceder al sistema sin utilizar la interfaz gráfica, es decir, realizar todo tipo de tareas en formato texto mediante órdenes.

Figura 16: Entorno de Trabajo de ROS

4.3. PROYECTO INICIAL

Para este proyecto se ha empleado un proyecto inicial que estaba colgado en Github. De este proyecto, se va a aprovechar que ya tiene creado una descripción del robot (URDF), los archivos de lanzamiento en Gazebo y los drivers necesarios para crear una conexión desde nuestro PC al robot.

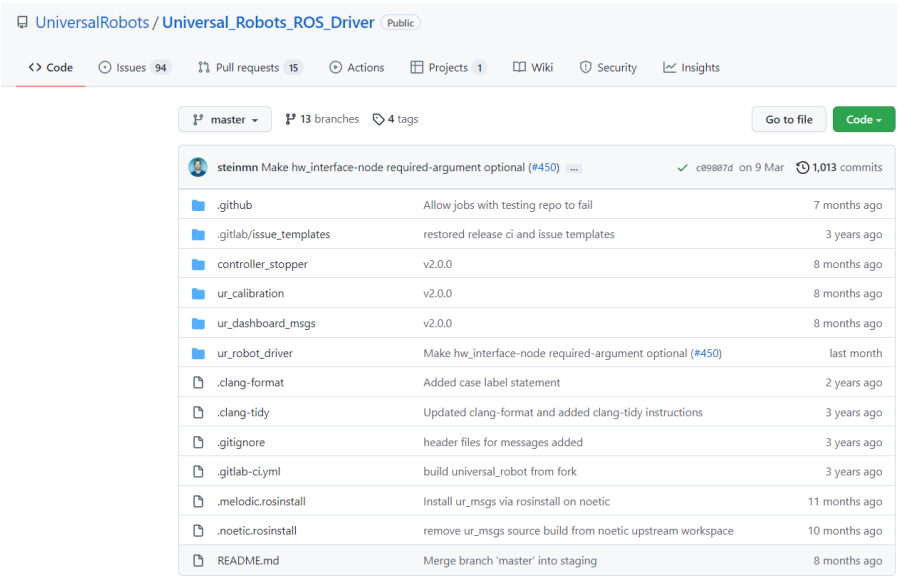


Figura 17: Proyecto Github

Aunque en este proyecto venga una descripción (URDF) del robot que lanzar en Gazebo, este va a tener que ser modificado porque el objetivo de esta parte de simulación en Gazebo es representar no solo el robot, sino también representar el lugar de trabajo donde está situado.

4.4. CREACIÓN DEL LUGAR DE TRABAJO

Para la creación del lugar de trabajo se tiene en cuenta el entorno en el que va a trabajar el robot. En el laboratorio de la universidad, el robot está situado sobre una Base, atornillada encima de una mesa. También existe una regleta en la esquina derecha del fondo.

Para la creación de estos elementos, como se explicó anteriormente, se va a emplear un sistema propio de ROS formado por marcos denominado XACRO, que utilizará un lenguaje como los archivos URDF (*Unified Robot Description Format*) de XML.

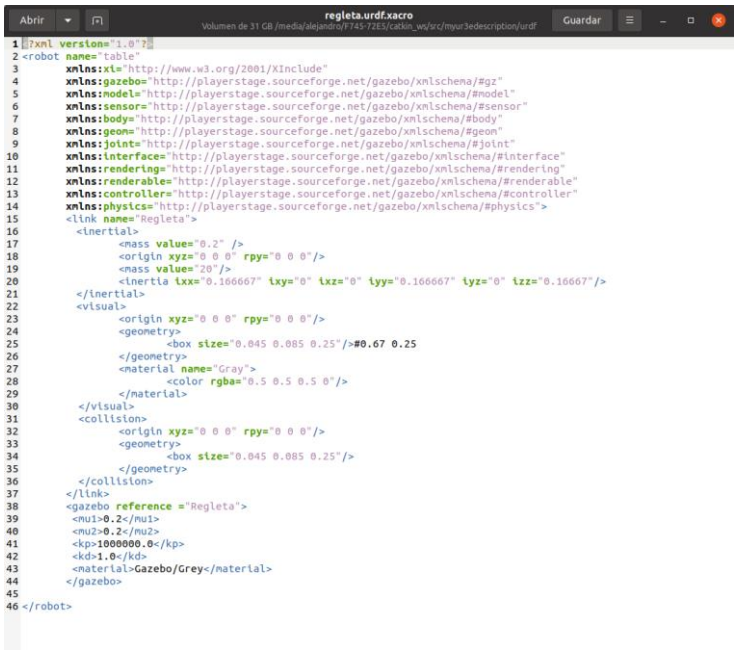


Figura 18: Ejemplo .Xacro

Una vez creado todos estos elementos en distintos archivos '.xacro', se importarán al archivo donde está el URDF del robot, estableciendo 'links' entre cada 'joint' de los componentes creados, situándolos con unas distancias uno respecto al otro que equivaldrían al lugar donde están representados en la realidad.

Al lanzar el Gazebo con este nuevo URDF el resultado será:

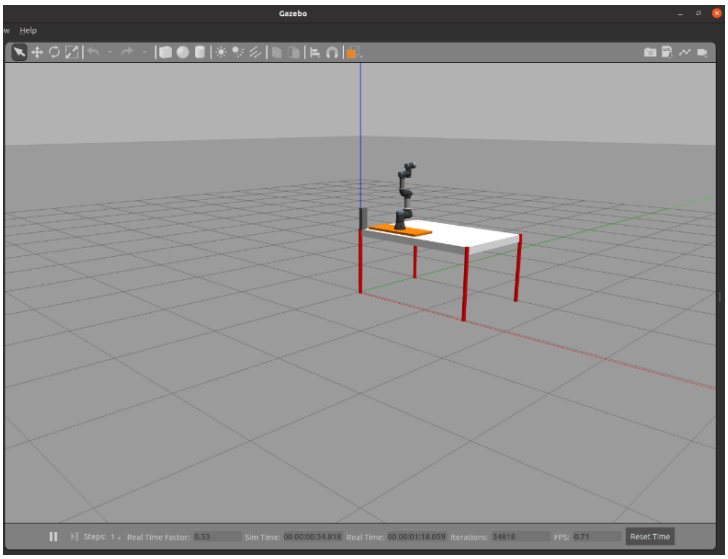


Figura 19: Gazebo final

Con este nuevo URDF del robot, donde se tienen en cuenta los elementos del entorno con los que puede interferir el robot, se puede empezar a crear el paquete Moveit para hacer manipulaciones y planificaciones de trayectorias del robot.

4.5. CREACIÓN DEL PAQUETE MOVEIT

Este apartado está explicado en el manual elaborado, donde se puede hacer un seguimiento paso por paso para la creación del paquete Moveit. Lo que se va a mostrar en este informe son apartados que considero más importantes a la hora de elaborar el paquete Moveit propio.

Para la creación del paquete Moveit se empleó un asistente propio de Moveit, al cual se le pasó el URDF del robot modificado anteriormente, donde se tiene en cuenta el lugar de trabajo donde está situado el robot.

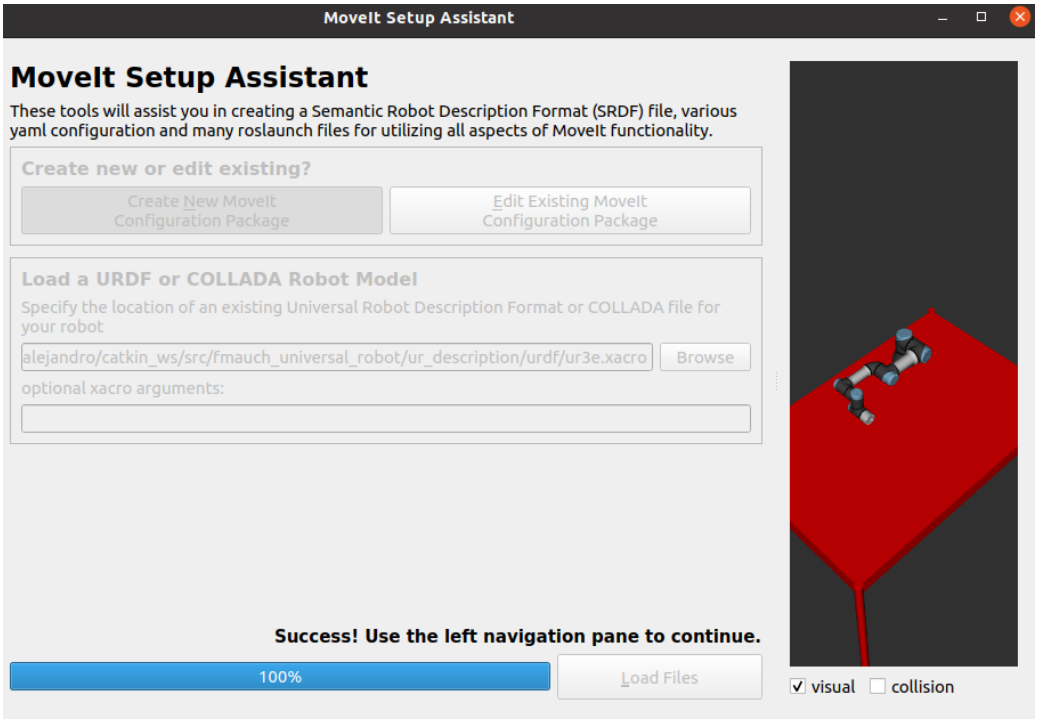


Figura 20: Moveit Setup Assistant

El siguiente paso es generar todas las colisiones y auto colisiones que se pueden dar cuando esté trabajando el robot.

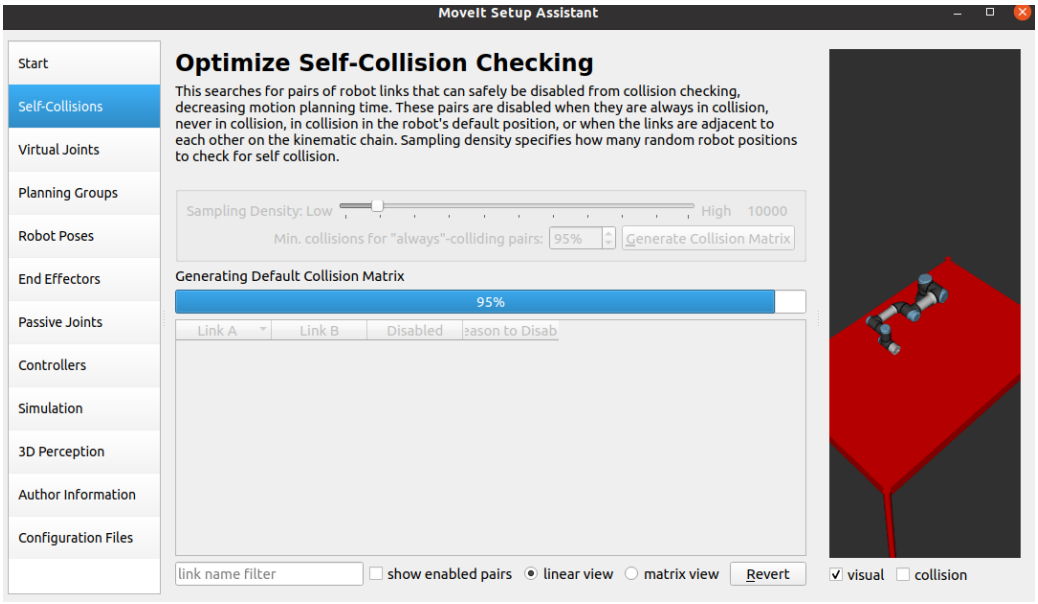


Figura 21: Moveit Self-Collision

El siguiente paso importante es añadir la articulación virtual que fijará el ‘link’ de la base de nuestro robot al mundo.

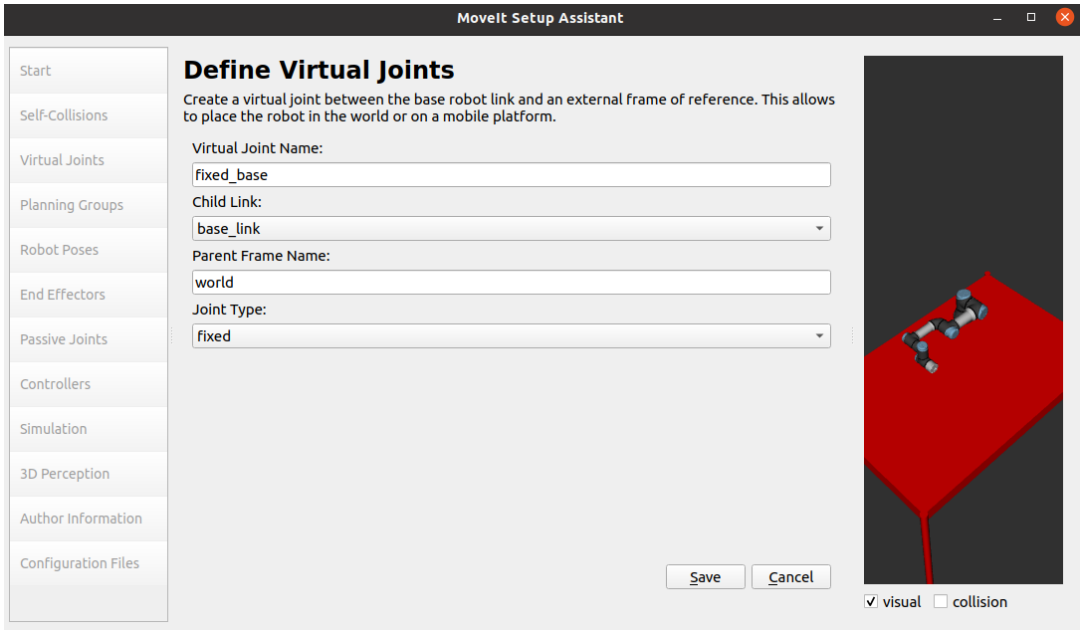


Figura 22: Moveit Virtual Joints

Lo siguiente será generar los Planning Groups (Grupos de Planificación). En este apartado se debe tener en cuenta que componentes van a realizar movimientos cuando se planifiquen trayectorias, los cuales son el robot (que van desde el link de la base del robot ‘base_link’ hasta la muñeca 3 ‘wrist3_link’) y la herramienta que se instale en el robot. También a cada uno de estos dos Planning Groups hay que asignarles que función se va a encargar de solucionar los problemas Kinemáticos y que conexión se va a establecer (en este caso RDT Connect).

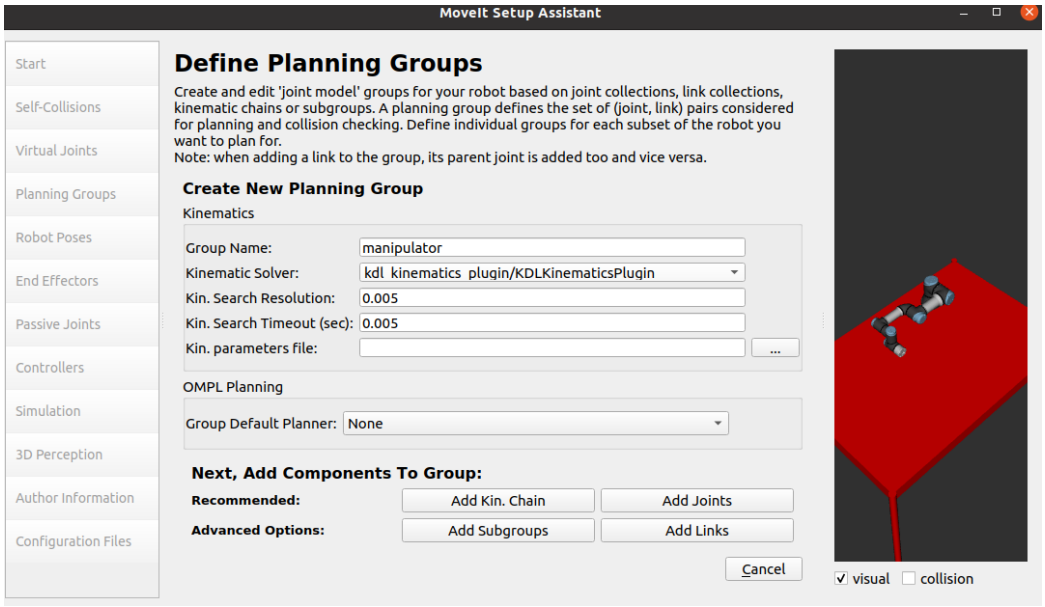


Figura 23: Moveit Planning Groups

Otro punto importante que hay que configurar a la hora de hacer el paquete Moveit, es configurar los controladores que va a usar el controlador mánager, el cual operará con el hardware físico del robot. Esto significa, que el controlador va a operar con los links del robot (base_link, shoulder_pan_joint, shoulder_lift_joint, wrist1_joint, wrist2_joint, wrist3_joint). Este controlador va a ser el 'pos_joint_traj_controller' que es un controlador común que está tanto en el gazebo del robot como en hardware físico del robot.

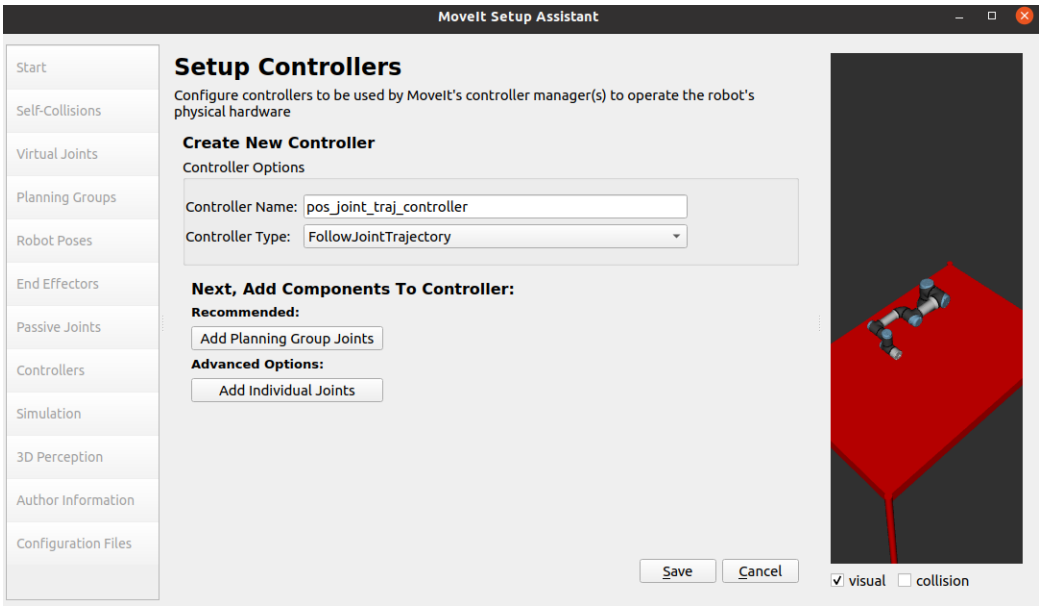


Figura 24: Moveit Setup Controllers

Una vez hecho todos los pasos descritos en el manual ya solo queda elegir el directorio en donde se quiere guardar el paquete, el nombre con el que lo queremos guardar y por último darle a generar el paquete Moveit.

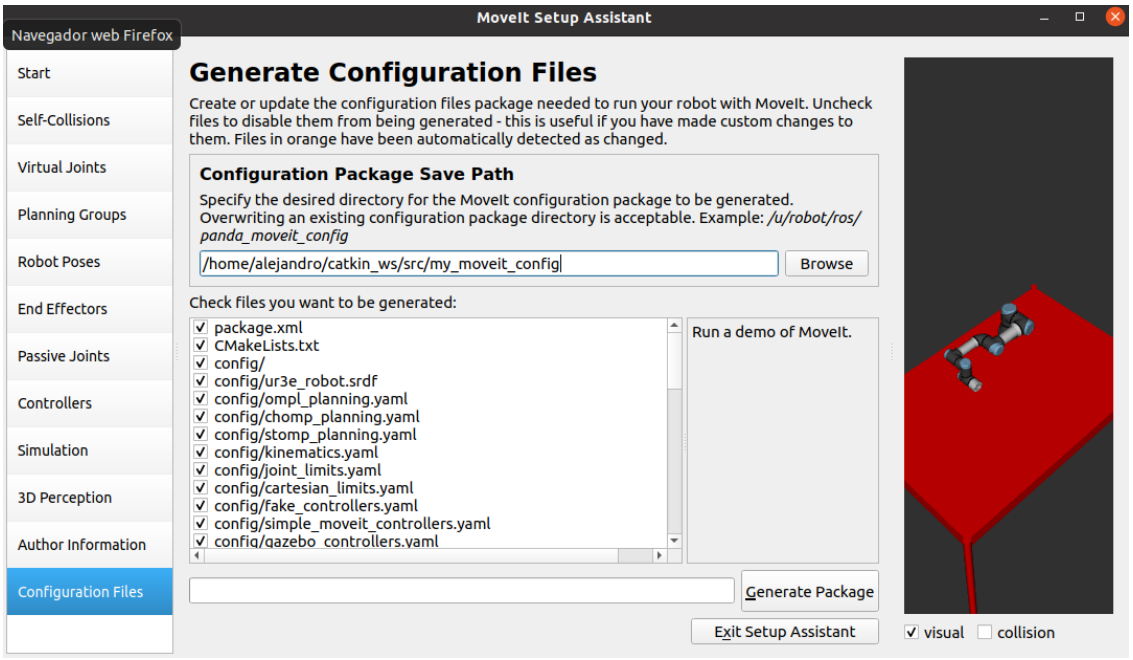


Figura 25: Moveit Generate Configuration Files

4.6. MODOS DE FUNCIONAMIENTO

En este apartado se va a explicar los dos modos de funcionamiento que se han planteado a la hora de querer interactuar o ver el comportamiento del robot en el entorno, una vez que se creó el paquete Moveit del robot.

El modo con el que se podrá visualizar cómo se comportaría el robot en el entorno sin llegar a interactuar con el robot físicamente se denominará Modo Simulación. Al modo en el que ya sí que se esté interactuando físicamente con el robot se le denominará Modo de Control Remoto.

En la realidad, el Modo Simulación es importante realizarlo previamente antes de interactuar con el hardware físico del robot para cerciorarse que las trayectorias mandadas son viables y no interfieren con ningún elemento del entorno con el que el que pueda chocar el robot. La forma en la que se calcularán y se mandarán las trayectorias tanto para el Modo Simulación como para el Modo de Control Remoto será mediante el archivo RVIZ generado cuando se creó el paquete Moveit o mediante un archivo Python.

4.6.1. MODO SIMULACIÓN

En este modo se utilizaría el Gazebo del robot para muestrear el comportamiento del robot en el entorno y como se explicó anteriormente, RVIZ o un archivo Python para planificar y mandar las trayectorias al Gazebo para que sean representadas.

Para poder trabajar en este modo es necesario usar una pestaña del terminal para lanzar el Gazebo mediante el comando en este caso:

```
alejandro@alejandro-VirtualBox:~/catkin_ws$ roslaunch ur_gazebo ur3e_bringup.lau
nch
```

En otra pestaña del terminal lanzar el comando que habilita que se puedan muestrear las trayectorias en modo de simulación:

```
alejandro@alejandro-VirtualBox:~/catkin_ws$ roslaunch my_moveit_config ur3e_move
it_planning_execution.launch sim:=true
```

Y en una tercera pestaña del terminal que archivo será el encargado de calcular las trayectorias:

- Usando RVIZ.

```
alejandro@alejandro-VirtualBox:~/catkin_ws$ roslaunch my_moveit_config moveit_rv
iz.launch config:=true
```

- Usando un archivo Python.

```
laboratorio@eliot-ThinkCentre-M710s:~/catkin_ws$ rosrun ur_robot_driver test_move
_Alejandro_v1.py
```

4.6.2. MODO DE CONTROL REMOTO

Una vez que se ha realizado el Modo Simulación y comprobado que las trayectorias programadas son viables con el entorno en el que se está trabajando, se procederá a controlar el robot UR3e físicamente de manera remota.

La interconexión se hará desde el PC al Robot y viceversa vía Ethernet. Esto significa que para empezar a controlar el robot hay que, desde el ordenador, mandar la dirección IP del robot que se va a controlar y desde el panel de control del robot la dirección IP del PC que va a controlar el robot.

Para poder trabajar en el Modo de Control Remoto, en el terminal se lanzará un archivo que habilitará la interconexión PC-robot y RVIZ o un archivo Python para planificar y mandar las trayectorias al igual que en el Modo Simulación.

La conexión PC-robot se establecerá mediante el comando:

```
laboratorio@eliot-ThinkCentre-M710s:~/catkin_ws$ roslaunch example_organization_ur_launch
ex-ur3e-1.launch
```

En otra pestaña del terminal lanzar el comando que habilita que se puedan muestrear las trayectorias:

```
~/catkin_ws$ roslaunch my_moveit_config ur3e_moveit_planning_execution.launch
```

Y en una tercera pestaña del terminal, al igual que en el modo simulación, el archivo que será el encargado de calcular las trayectorias:

- Usando RVIZ.

```
alejandro@alejandro-VirtualBox:~/catkin_ws$ roslaunch my_moveit_config moveit_rv
iz.launch config:=true
```

- Usando un archivo Python.

```
laboratorio@eliot-ThinkCentre-M710s:~/catkin_ws$ rosrun ur_robot_driver test_move
_Alejandro_v1.py
```

5. EJEMPLO DE USO

En este apartado se van a exponer imágenes de cómo se comportaría el robot cuando se esté trabajando en el modo Simulación y de cómo se comportaría el robot cuando se trabaja en el modo de Control Remoto.

5.1. FUNCIONAMIENTO EN MODO SIMULACIÓN

Siguiendo los pasos de como funcionar en el modo Simulación, que se describen en el Apartado 4.6.1, lo primero que se hará será ejecutar el entorno para la simulación usando Gazebo donde se mostrarán aparte del robot los distintos elementos que son influyentes en el entorno.

```
alejandro@alejandro-VirtualBox:~/catkin_ws$ roslaunch ur_gazebo ur3e_bringup.lau
nch
```

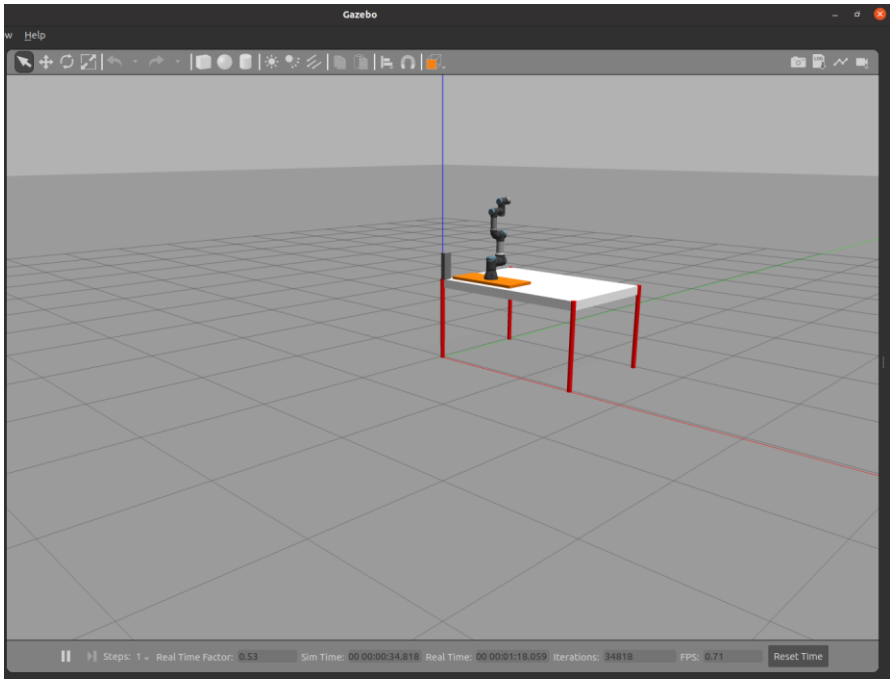


Figura 26: Gazebo inicialmente simulado

El siguiente paso es lanzar el archivo *'ur3e_planning_execution.launch'* en modo simulación.

```
alejandro@alejandro-VirtualBox:~/catkin_ws$ roslaunch my_moveit_config ur3e_move
it_planning_execution.launch sim:=true
```

Después de ejecutar el último comando, queda por elegir si se va a utilizar RVIZ para realizar las trayectorias del robot o se va a emplear un archivo Python.

5.1.1. MEDIANTE RVIZ

Ejecutando el comando:

```
alejandro@alejandro-VirtualBox:~/catkin_ws$ roslaunch my_moveit_config moveit_rv
iz.launch config:=true
```

Se abrirá la interfaz de RVIZ:

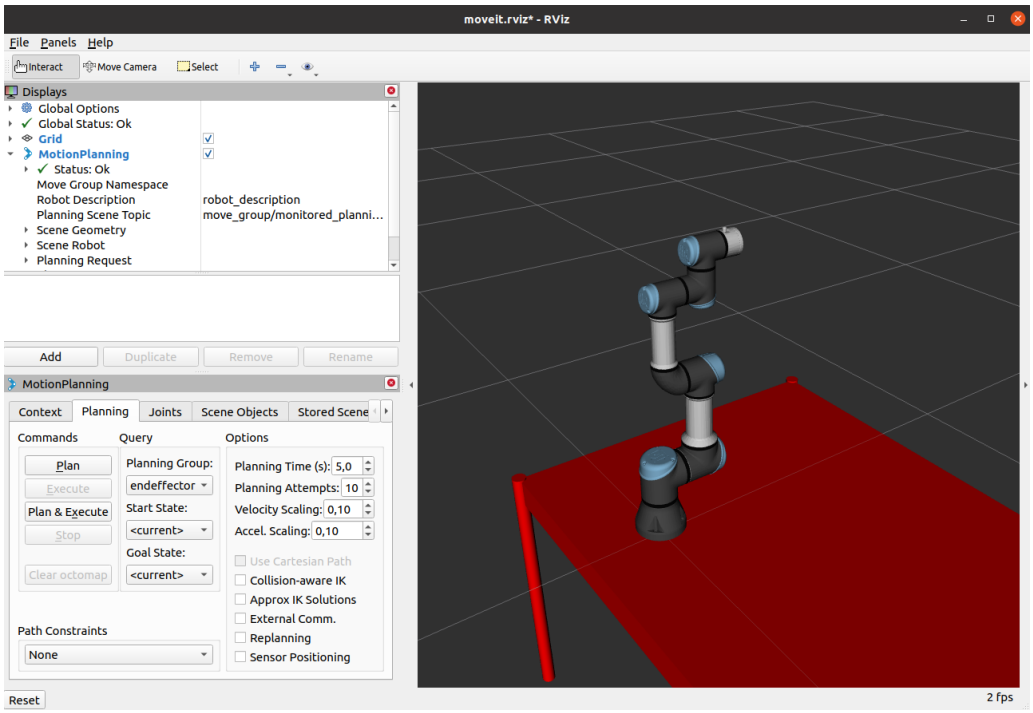


Figura 27: Interfaz RVIZ

En esta interfaz de RVIZ se muestra el robot y la mesa sobre la que trabaja. Para que el robot vaya a un punto de interés en su espacio de trabajo se puede manipular, por ejemplo, en la pestaña de 'Joints', la posición de cada una de las juntas.

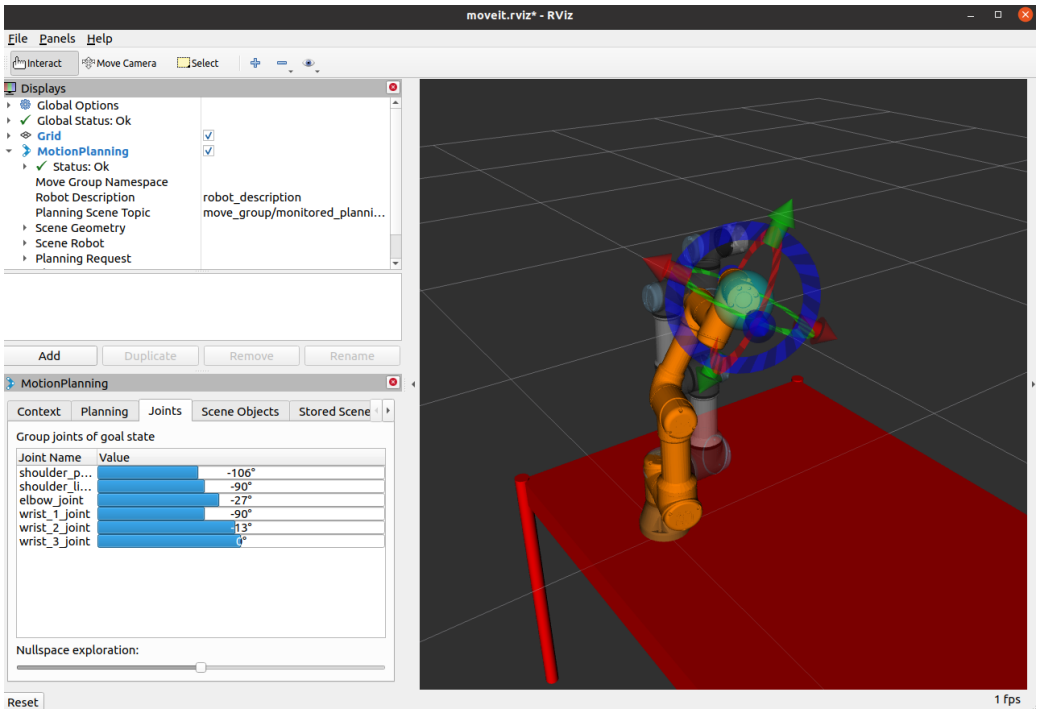


Figura 28: Planificar trayectorias mediante RVIZ

Una vez escogido el punto objetivo, lo siguiente será darle al botón de Plan. Este botón calculará una trayectoria viable hasta el punto objetivo. Una vez calculada la trayectoria solo quedaría ejecutar esa trayectoria y visualizar como se representa esa misma trayectoria en Gazebo.

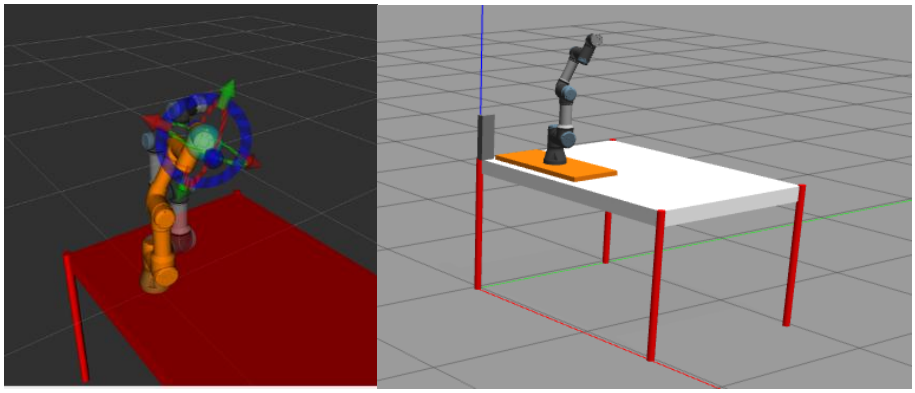
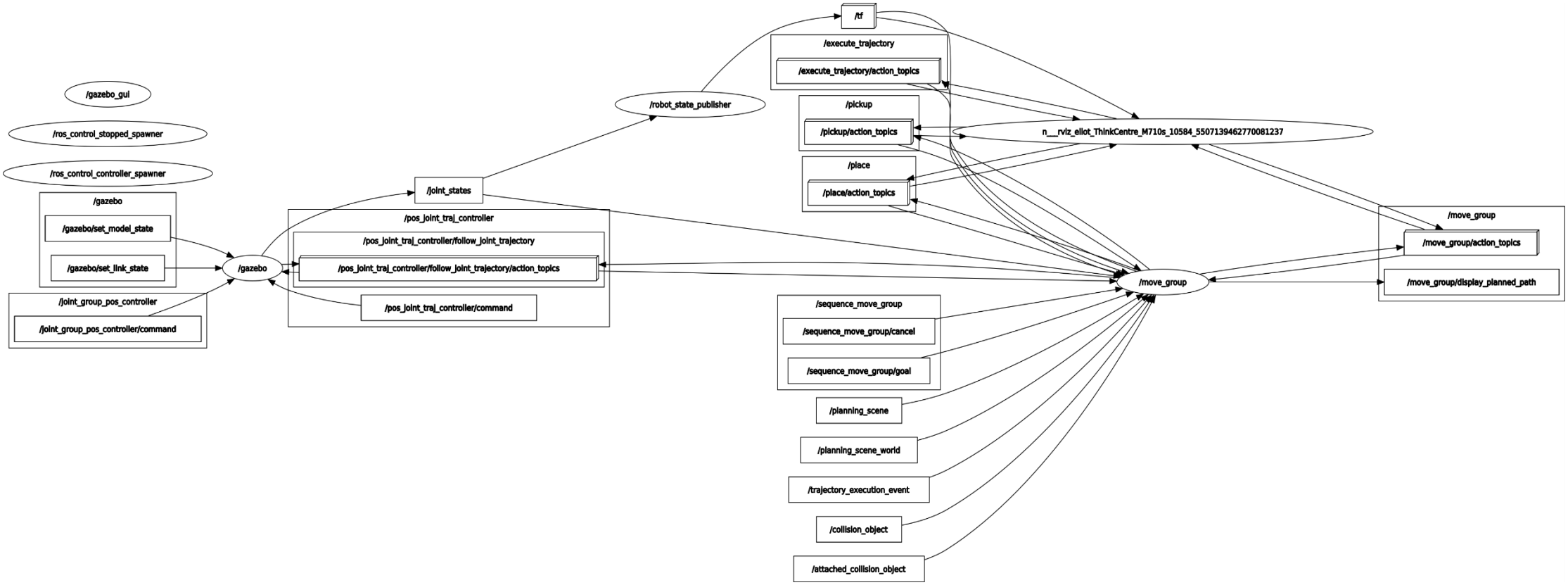


Figura 29: Gazebo simulado con RVIZ

Tras la ejecución de los comandos anteriores también se puede obtener el esquema de nodos y tópicos, y la forma con la que se relacionan (rqt_graph). Este paso es conveniente realizarlo para comprobar que todas las comunicaciones entre nodos estén conectadas correctamente y de cómo se podrían implementar nuevas mejoras al proyecto.


/gazebo_gui
/ros_control_stopped_spawner
/ros_control_controller_spawner
/gazebo
/gazebo/set_model_state
/gazebo/set_link_state
/joint_group_pos_controller
/joint_group_pos_controller/command
/gazebo
/joint_states
/robot_state_publisher
/pos_joint_traj_controller
/pos_joint_traj_controller/follow_joint_trajectory
/pos_joint_traj_controller/follow_joint_trajectory/action_topics
/pos_joint_traj_controller/command
/tf
/execute_trajectory
/execute_trajectory/action_topics
/pickup
/pickup/action_topics
/place
/place/action_topics
n__rviz_eliot_ThinkCentre_M710s_10584_5507139462770081237
/move_group
/move_group
/move_group/action_topics
/move_group/display_planned_path
/sequence_move_group
/sequence_move_group/cancel
/sequence_move_group/goal
/planning_scene
/planning_scene_world
/trajectory_execution_event
/collision_object
/attached_collision_object

Como se puede observar en el esquema existen cuatro nodos fundamentales (‘/gazebo’, ‘/robot_state_publisher’, ‘n__rviz…’ y ‘move_group’). Todo nodo está conectado entre sí por tópicos o servicios.

El primer nodo corresponde con Gazebo, donde se lanzan los controladores del Gazebo y la posición actual de las juntas del robot ‘joint_states’. El controlador ‘pos_joint_traj_controller’ es el controlador común con el paquete Moveit y el Gazebo.

El segundo nodo ‘/robot_state_publisher’ envía información sobre el estado actual de las juntas ‘joint_states’ del robot en Gazebo.

El tercer nodo ‘n__rviz…’ corresponde con RVIZ. A este nodo le llegan los tópicos o servicios del archivo ‘planning_execution’, la posición de las juntas y las trayectorias que esté realizando el robot simulado. Con esta información, RVIZ realiza los respectivos cálculos de las trayectorias a seguir y reenvía dicha información en dirección hacia el último nodo ‘move_group’.

El último nodo ‘move_group’, es el nodo que comunica todos los temas, servicios y acciones de ROS con el controlador común ‘pos_joint_traj_controller’ de Gazebo, el cual se encarga de representar las trayectorias calculadas mediante RVIZ en Gazebo.

El trabajo en conjunto de estos cuatro nodos constituye el funcionamiento del robot simulado.

En resumen, cuando se lanza Gazebo, se genera el modelo del robot, se lanza el controlador común ‘pos_joint_traj_controller’ y un nodo que envía la situación actual de las juntas ‘joint_states’. Utilizando los tópicos y servicios generados al lanzar el archivo ‘planning_execution’ y la posición de las juntas se planificarán las trayectorias mediante el uso de la interfaz RVIZ. RVIZ enviará las trayectorias a realizar al nodo ‘move_group’,

que a su vez enviará información de las distintas posiciones en esa trayectoria al controlador 'pos_joint_traj_controller', que representará dicha información en Gazebo.

5.1.2. MEDIANTE PYTHON

En este apartado, en vez de usar RVIZ para planificar las trayectorias, se va a utilizar un archivo de Python, el cual le va a pasar al robot (real o simulado) una trayectoria formada por puntos. Mediante el comando:

```
laboratorio@eliot-ThinkCentre-M710s:~/catkin_ws$ rosrun ur_robot_driver test_move
_Alejandro_v1.py
```

El robot seguirá la trayectoria formada por los puntos:

```
position_list = [[0, -1.57, -1.57, 0, 0, 0]]
position_list.append([0, -1.57, -1.57, 0, 0, 0])
position_list.append([-1.57, -2.36, -1.71, -0.61, 1.59, 0])
position_list.append([-1.57, -2.36, -1.71, -0.61, 1.59, 0])
position_list.append([-0.5, -1.57, -1.2, 0, 0, 0])
position_list.append([-0.5, -1.57, -1.2, 0, 0, 0])
position_list.append([0, -1.57, 0, -1.57, 0, 0])
```

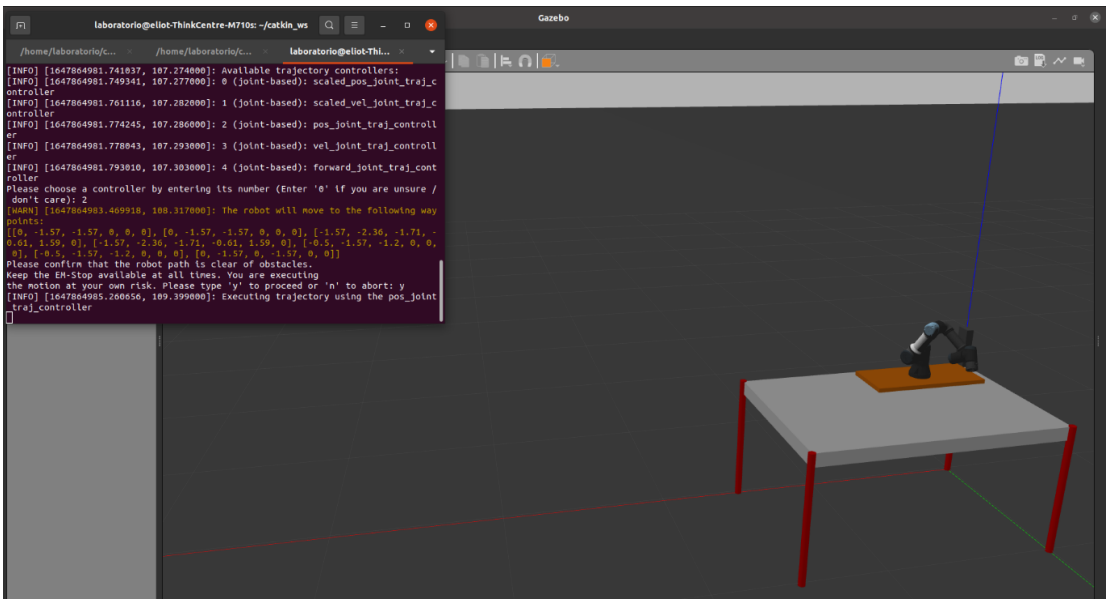
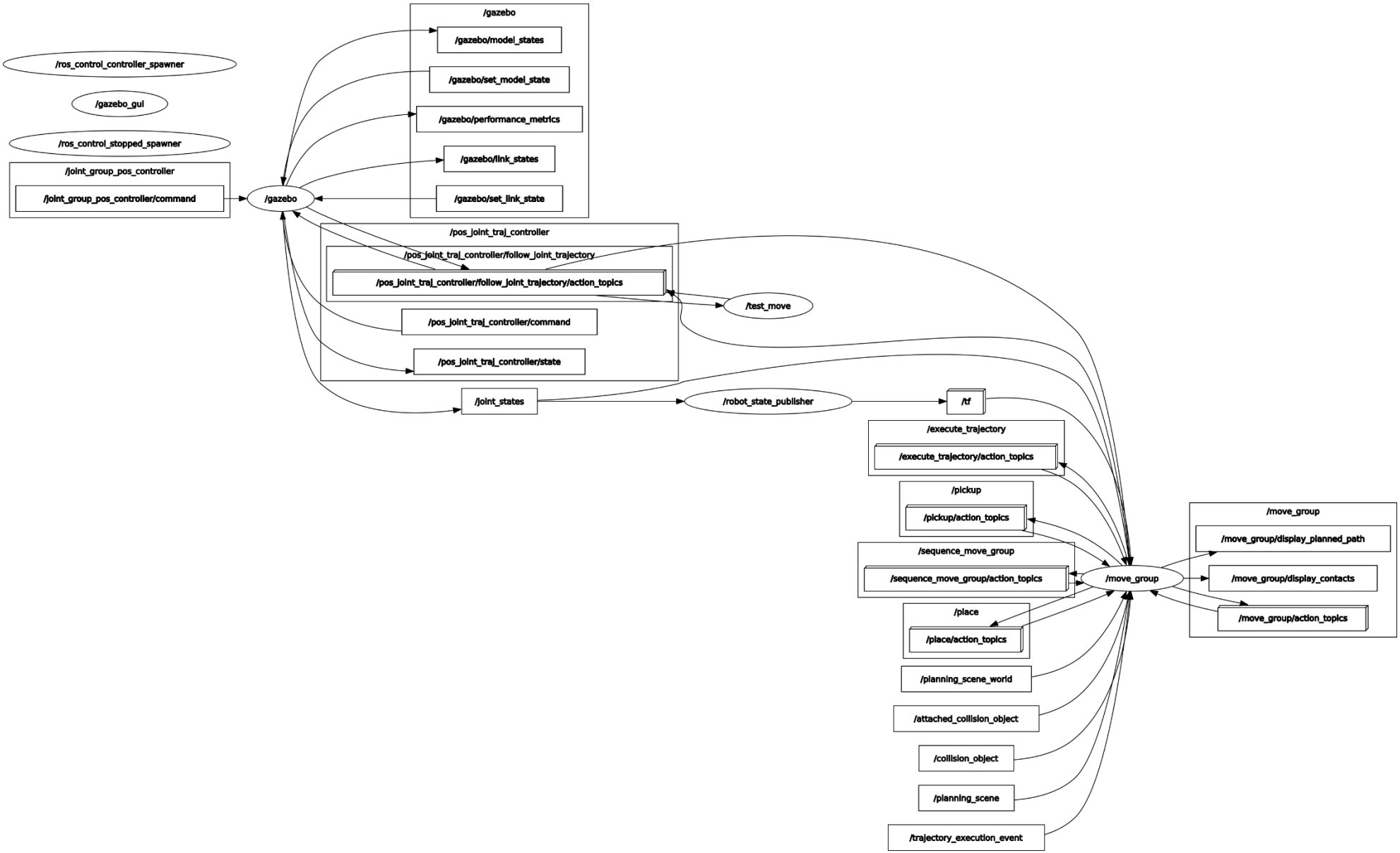


Figura 30: Gazebo simulado con archivo Python

Al igual que con RVIZ el esquema de nodos y topics, y la forma con la que se relacionan será:


/gazebo
/gazebo/model_states
/gazebo/set_model_state
/gazebo/performance_metrics
/gazebo/link_states
/gazebo/set_link_state
/ros_control_controller_spawner
/gazebo_gui
/ros_control_stopped_spawner
/joint_group_pos_controller
/joint_group_pos_controller/command
/gazebo
/pos_joint_traj_controller
/pos_joint_traj_controller/follow_joint_trajectory
/pos_joint_traj_controller/follow_joint_trajectory/action_topics
/test_move
/pos_joint_traj_controller/command
/pos_joint_traj_controller/state
/joint_states
/robot_state_publisher
/tf
/execute_trajectory
/execute_trajectory/action_topics
/pickup
/pickup/action_topics
/sequence_move_group
/sequence_move_group/action_topics
/move_group
/place
/place/action_topics
/planning_scene_world
/attached_collision_object
/collision_object
/planning_scene
/trajectory_execution_event
/move_group
/move_group/display_planned_path
/move_group/display_contacts
/move_group/action_topics

Como se puede observar, el esquema es parecido al anterior. Existen cuatro nodos (‘/gazebo’, ‘/robot_state_publisher’, ‘test_move’ y ‘move_group’). El funcionamiento es muy similar al esquema cuando se utiliza RVIZ. La única diferencia es que aquí no existe el nodo correspondiente a RVIZ. Otro cambio, es la existencia de un nodo ‘test_move’ cuyo funcionamiento es realizar una previa prueba de que el movimiento a realizar es viable.

5.2. FUNCIONAMIENTO EN MODO CONTROL REMOTO

Siguiendo los pasos de como funcionar en el modo Simulación, que se describen en el Apartado 4.6.2, lo primero que se hará será ejecutar el archivo que contiene los drivers que permiten la conexión del PC con el robot. El comando que permite la conexión del PC con el robot será:

```
laboratorio@eliot-ThinkCentre-M710s:~/catkin_ws$ roslaunch example_organization_ur_launch
ex-ur3e-1.launch
```

Después hay que habilitar en el panel de control del robot la conexión Robot con el PC.

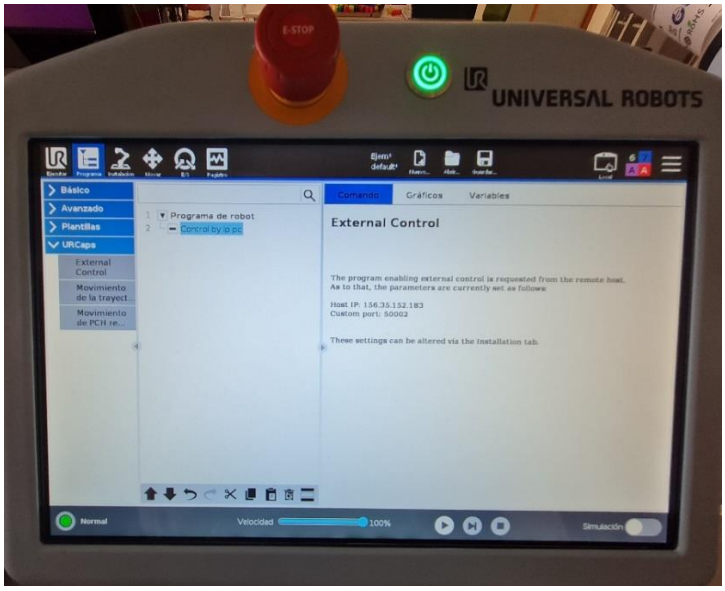


Figura 31: Habilitar conexión robot-PC

El siguiente paso es lanzar el archivo *'ur3e_planning_execution.launch':*

```
~/catkin_ws$ roslaunch my_moveit_config ur3e_moveit_planning_execution.launch
```

Después de ejecutar el último comando, queda por elegir de nuevo si se va a utilizar RVIZ para realizar las trayectorias del robot o se va a emplear un archivo Python.

5.2.1. MEDIANTE RVIZ

Usando el comando que lanza RVIZ:

```
alejandro@alejandro-VirtualBox:~/catkin_ws$ roslaunch my_moveit_config moveit_rv
iz.launch config:=true
```

Ya se puede empezar a planificar trayectorias usando RVIZ y que el robot se mueva al punto al cual se quiere llegar.

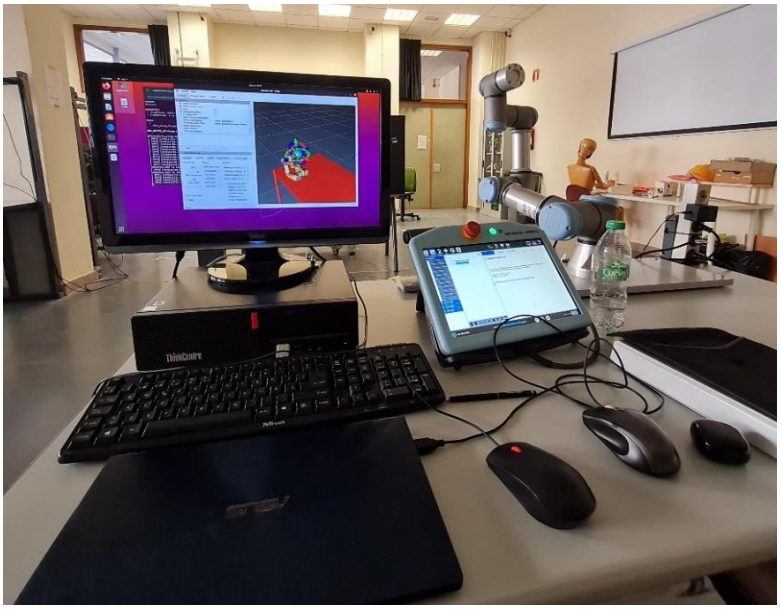
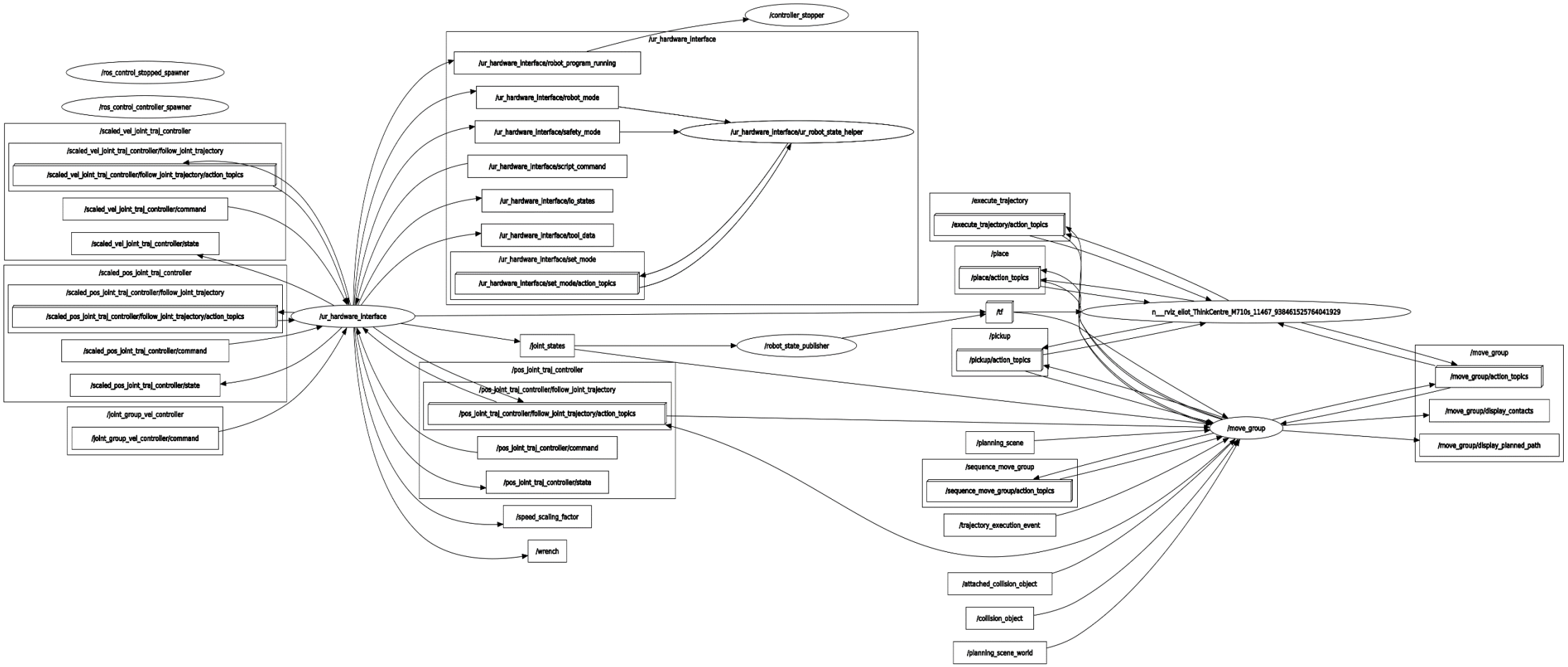


Figura 32: Control del robot UR3e mediante RVIZ

El esquema de nodos y topics, y la forma con la que se relacionan será:


/controller_stopper
/ur_hardware_interface
/ur_hardware_interface/robot_program_running
/ur_hardware_interface/robot_mode
/ur_hardware_interface/safety_mode
/ur_hardware_interface/ur_robot_state_helper
/ur_hardware_interface/script_command
/ur_hardware_interface/io_states
/ur_hardware_interface/tool_data
/ur_hardware_interface/set_mode
/ur_hardware_interface/set_mode/action_topics
/ros_control_stopped_spawner
/ros_control_controller_spawner
/scaled_vel_joint_traj_controller
/scaled_vel_joint_traj_controller/follow_joint_trajectory
/scaled_vel_joint_traj_controller/follow_joint_trajectory/action_topics
/scaled_vel_joint_traj_controller/command
/scaled_vel_joint_traj_controller/state
/scaled_pos_joint_traj_controller
/scaled_pos_joint_traj_controller/follow_joint_trajectory
/scaled_pos_joint_traj_controller/follow_joint_trajectory/action_topics
/scaled_pos_joint_traj_controller/command
/scaled_pos_joint_traj_controller/state
/joint_group_vel_controller
/joint_group_vel_controller/command
/ur_hardware_interface
/joint_states
/robot_state_publisher
/pos_joint_traj_controller
/pos_joint_traj_controller/follow_joint_trajectory
/pos_joint_traj_controller/follow_joint_trajectory/action_topics
/pos_joint_traj_controller/command
/pos_joint_traj_controller/state
/speed_scaling_factor
/wrench
/execute_trajectory
/execute_trajectory/action_topics
/place
/place/action_topics
/tf
/pickup
/pickup/action_topics
n__rviz_eilot_ThinkCentre_M710s_11467_93846152576404192 9
/move_group
/move_group/action_topics
/move_group/display_contacts
/move_group/display_planned_path
/move_group
/planning_scene
/sequence_move_group
/sequence_move_group/action_topics
/trajectory_execution_event
/attached_collision_object
/collision_object
/planning_scene_world

En este esquema, a diferencia del esquema del modo Simulación, existe un nuevo nodo correspondiente al hardware del robot (‘ur_hardware_interface’) que sustituye al nodo de Gazebo. También existen tópicos correspondientes al hardware y a los controladores con los que puede trabajar el robot.

Existe también otro nuevo nodo, el ‘controller_stoper’, que interrumpirá el uso del controlador si existe algún problema de seguridad con el robot. Los otros nodos (‘/robot_state_publisher’, ‘n__rviz…’ y ‘move_group’) realizan la misma función que en el modo Simulación.

En resumen, en este modo se lanzan primero los drivers que generan una conexión PC-robot, los controladores con los que puede trabajar el robot (‘pos_joint_traj_controller’, ‘scaled_vel_traj_controller’, ‘scaled_pos_traj_controller’, ‘joint_group_vel_controller’) y un nodo que envía la situación actual de las juntas ‘joint_states’. Utilizando los tópicos y servicios generados al lanzar el archivo ‘planning_execution’ y la posición de las juntas se planificarán las trayectorias mediante el uso de la interfaz RVIZ. RVIZ enviará las trayectorias a realizar al nodo ‘move_group’, que a su vez enviará información de las distintas posiciones en esa trayectoria al controlador que se esté utilizando en ese momento, que representará dicha información en el robot.

5.2.2. MEDIANTE PYTHON

Se usará el mismo archivo Python que el empleado en el modo Simulación. Esto significa que se recorrerá la misma trayectoria, pasando por los puntos:

```
position_list = [[0, -1.57, -1.57, 0, 0, 0]]
position_list.append([0, -1.57, -1.57, 0, 0, 0])
position_list.append([-1.57, -2.36, -1.71, -0.61, 1.59, 0])
position_list.append([-1.57, -2.36, -1.71, -0.61, 1.59, 0])
position_list.append([-0.5, -1.57, -1.2, 0, 0, 0])
position_list.append([-0.5, -1.57, -1.2, 0, 0, 0])
position_list.append([0, -1.57, 0, -1.57, 0, 0])
```

Ejecutando el siguiente comando el robot recorrerá la trayectoria anterior:

```
laboratorio@eliot-ThinkCentre-M710s:~/catkin_ws$ rosrun ur_robot_driver test_move
_Alejandro_v1.py
```

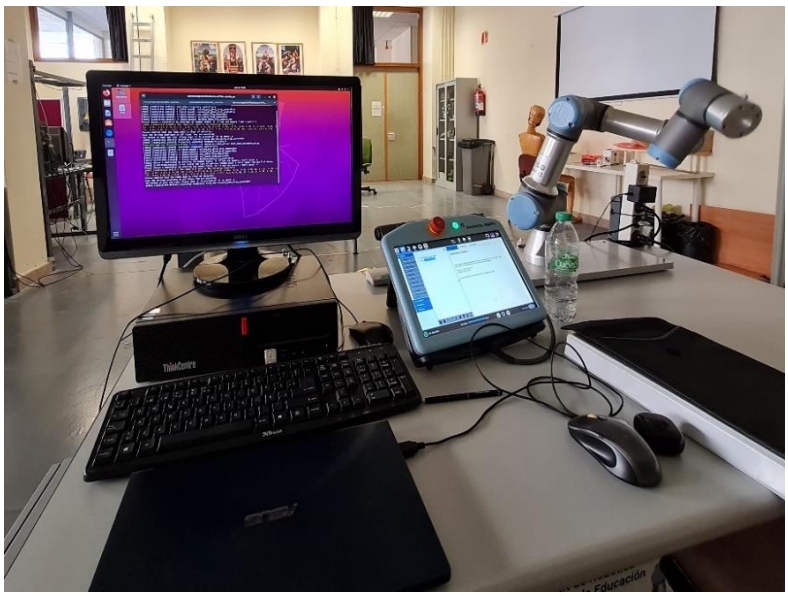
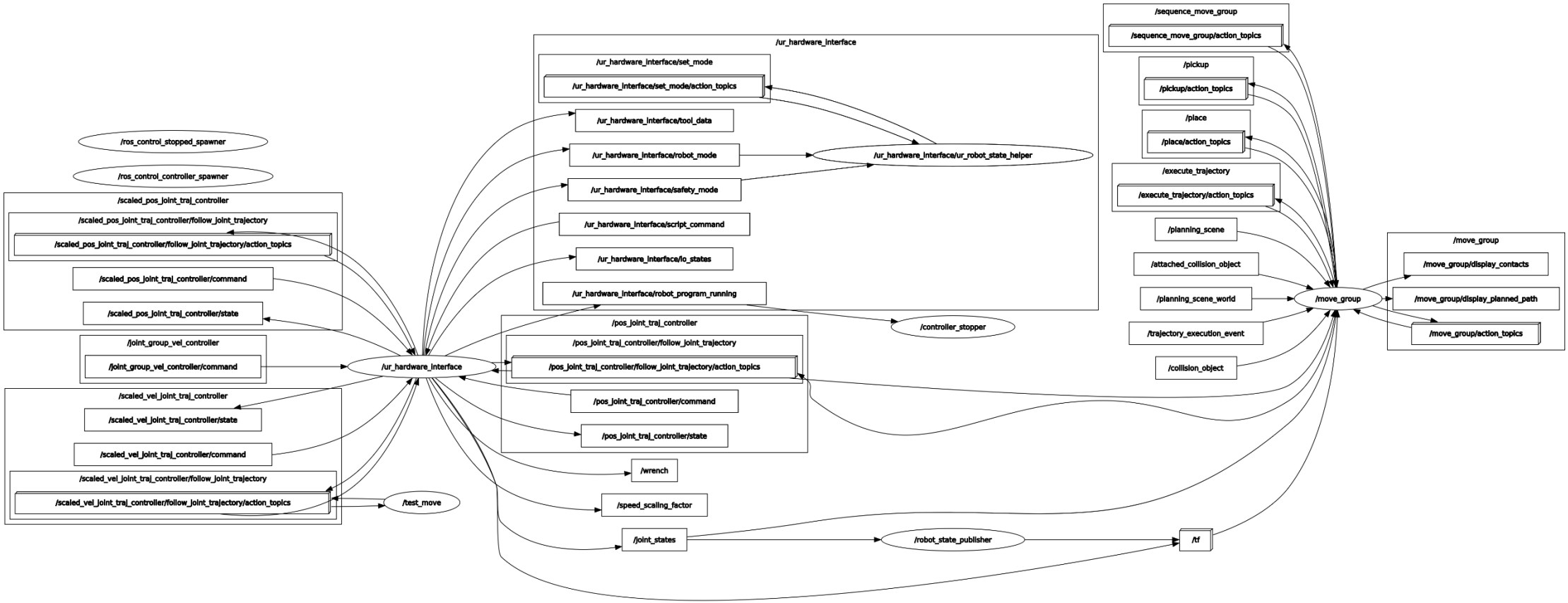


Figura 33: Control del robot UR3e mediante Python

El esquema de nodos y tópicos, y la forma con la que se relacionan será:


/sequence_move_group
/sequence_move_group/action_topics
/ur_hardware_interface
/ur_hardware_interface/set_mode
/ur_hardware_interface/set_mode/action_topics
/pickup
/pickup/action_topics
/place
/place/action_topics
/execute_trajectory
/execute_trajectory/action_topics
/ur_hardware_interface/tool_data
/ur_hardware_interface/robot_mode
/ur_hardware_interface/ur_robot_state_helper
/ur_hardware_interface/safety_mode
/ur_hardware_interface/script_command
/ur_hardware_interface/io_states
/ur_hardware_interface/robot_program_running
/controller_stopper
/ros_control_stopped_spawner
/ros_control_controller_spawner
/scaled_pos_joint_traj_controller
/scaled_pos_joint_traj_controller/follow_joint_trajectory
/scaled_pos_joint_traj_controller/follow_joint_trajectory/action_topics
/scaled_pos_joint_traj_controller/command
/scaled_pos_joint_traj_controller/state
/joint_group_vel_controller
/joint_group_vel_controller/command
/ur_hardware_interface
/pos_joint_traj_controller
/pos_joint_traj_controller/follow_joint_trajectory
/pos_joint_traj_controller/follow_joint_trajectory/action_topics
/pos_joint_traj_controller/command
/pos_joint_traj_controller/state
/scaled_vel_joint_traj_controller
/scaled_vel_joint_traj_controller/state
/scaled_vel_joint_traj_controller/command
/scaled_vel_joint_traj_controller/follow_joint_trajectory
/scaled_vel_joint_traj_controller/follow_joint_trajectory/action_topics
/test_move
/wrench
/speed_scaling_factor
/joint_states
/robot_state_publisher
/tf
/planning_scene
/attached_collision_object
/planning_scene_world
/trajectory_execution_event
/collision_object
/move_group
/move_group
/move_group/display_contacts
/move_group/display_planned_path
/move_group/action_topics

Como se puede observar, el esquema vuelve a ser parecido al anterior. Existen cuatro nodos principales (‘/ur_hardware_interface’, ‘/robot_state_publisher’, ‘test_move’ y ‘move_group’). Los tópicos creados en el ‘planing_execution’ y exportados al archivo Python son los encargados de planificar la trayectoria y el nodo ‘move_group’ el encargado de comunicar las trayectorias al robot.

6. CONCLUSIONES

El campo de la robótica colaborativa está aún en fase de crecimiento y expansión. El número de aplicaciones que se le puede dar a la robótica colaborativa es muy amplio gracias a la versatilidad y seguridad que ofrecen y a la facilidad de instalar un robot colaborativo en cualquier entorno.

En lo que se refiere a posibles trabajos futuros, se pueden implementar gran cantidad de ampliaciones al proyecto. En este proyecto lo que se hizo fue implementar y documentar en forma de manual, los pasos iniciales necesarios para después poder crear aplicaciones complejas utilizando ROS, Moveit y Python. Una posible mejora que se le puede hacer al proyecto es incluir, a la hora de crear el paquete Moveit, una pinza y sus funcionalidades. Otra posible mejora puede ser, a la hora de programar el archivo Python, mediante el uso también de sensores de posición instalados en el robot, detectar la existencia de obstáculos gracias a los datos obtenidos del sensor, los cuales han sido tratados en el archivo Python y usados por ejemplo para calcular una nueva trayectoria que eviten dichos obstáculos.

Este proyecto me ha ayudado a ampliar mi conocimiento sobre la robótica, aprender nuevas técnicas, entornos de programación (ROS), lenguajes de programación (XML) y descubrir nuevos programas de simulación 3D (Gazebo y RVIZ). También me ha servido para mejorar mi capacidad para trabajar de manera autónoma, recurriendo a diversas fuentes de información para resolver los problemas que surgieron durante el desarrollo del trabajo.

7. BIBLIOGRAFÍA

[1] https://www.youtube.com/watch?v=ayp87SjrwPc&t=706s (Última visita: 07/06/2022)

[2]https://www.youtube.com/watch?v=BxCik8OI1Fw (Última visita: 07/06/2022)

[3] https://ros-planning.github.io/moveit_tutorials/

[4] http://wiki.ros.org/ur3_moveit_config

[5] https://www.universal-robots.com/es/productos/robot-ur3/

[6] https://cfzcobots.com/wp-content/uploads/2018/06/UR3e_User_Manual_es_Global.pdf

[7] https://www.ros.org/

[8] https://dle.rae.es/rob%C3%B3tica

[9] https://github.com/hxn8439/fmauch_universal_robot

[10] Universal Robots e-Series Manual de Usuario