Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Teleoperación de un robot humanoide mediante gafas de realidad virtual.

Autor:

David Álvarez Gil

Tutor(es):

Jaime Gómez García-Bermejo
Ingeniería de Sistemas y Automática

Jaime Duque Domingo
Ingeniería de Sistemas y Automática

Valladolid, Febrero 2023.

Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Teleoperación de un robot humanoide mediante gafas de realidad virtual.

Autor: David Álvarez Gil

Valladolid, Febrero y 2023.

Resumen

El presente Trabajo Fin de Grado se centra en el desarrollo de una aplicación para poder simular la teleoperación de un robot humanoide llamado Reachy. Esta teleoperación se realiza mediante un equipo de realidad virtual donde la persona que lo controla se introduce en el cuerpo del robot en un entorno domestico simulado.

Antes de conseguir este desarrollo se realizan otras simulaciones con el robot para observar las distintas cualidades que posee. Gracias a estas es idóneo tanto para realizar tareas de ayuda a personas dependientes como para interactuar con ellas de manera agradable.

En este proyecto se muestra el gran potencial que tienen la robótica y de la realidad virtual. No solo por separado, sino que juntando ambas disciplinas se pueden lograr avances extraordinarios.

Palabras clave: Robótica, realidad virtual, Reachy, teleoperación, humanoide.

Abstract

This Final Degree Project focuses on the development of an application to simulate the teleoperation of a humanoid robot called Reachy. This teleoperation is carried out by means of a virtual reality equipment where the person who controls it enters the body of the robot in a simulated domestic environment.

Before achieving this development, other simulations are carried out with the robot to observe the different qualities it possesses. Thanks to these, it is ideal both for helping dependent people and for interacting with them in a pleasant way.

This project shows the great potential of robotics and virtual reality. Not only separately, but by joining both disciplines extraordinary advances can be achieved.

Keywords: Robotics, virtual reality, Reachy, teleoperation, humanoid.

Índice

Índice de figuras

Índice de tablas

Capítulo 1:
Introducción y objetivos

Este apartado se debe introducir/acotar el tema del trabajo y su justificación, así como explicitar el alcance y objetivo principal del mismo, sus objetivos secundarios, si los hubiera, y una breve descripción sobre la estructura que se le ha dado al trabajo.

Se valorará la realización de una búsqueda Bibliográfica, como paso previo al desarrollo del TFG, que localice los trabajos más relevantes existentes en las Bases de Datos (Nacionales e Internacionales) a las cuales está suscrita la UVa, dejando constancia explícita en la memoria de dicha búsqueda,

1.1 Marco del proyecto.

El presente proyecto se enmarca dentro del ámbito de la robótica y la realidad virtual. Ambas tecnologías se empezaron a desarrollar en el siglo XX, pero no ha sido hasta las últimas décadas cuando la evolución ha sido notable.

En robótica se pueden diferenciar dos grandes campos: la robótica industrial y la robótica de servicios. Los robots de tipo industrial son aquellos que se encuentran en factorías o empresas y que, por lo general, realizan o ayudan en la realización de tareas industriales. Por su parte, los robots de servicios se encargan de la asistencia y la ayuda a los humanos en diversas actividades, generalmente no enfocadas al ámbito industrial.

Por lo tanto, este trabajo se centrará en la robótica de servicios. En la cual se puede distinguir dependiendo la interacción que realicen; de servicio personal, de servicio profesional o humanoide. Dentro de cada categoría hay subdivisiones dependiendo la tarea a realizar y el entorno donde se desarrolle.

Para nuestro proyecto utilizaremos un robot humanoide, que como su propio nombre indica tiene un aspecto semejante al de una persona real. Cabe destacar la relevancia de dicho aspecto en la robótica actual, ya que si se desea incorporar robots que trabajen en contacto con humanos, es importante que dicho contacto sea lo más natural posible, y para ello cobra gran importancia tanto el aspecto como los mecanismos de interacción.

En este caso, como queremos dotar a nuestro robot humanoide de una alta flexibilidad estará teleoperado, una persona controlará el robot de forma remota.

El control remoto se realizará mediante un equipo de realidad virtual. La realidad virtual genera, a base de tecnología informática, entornos de escenas y objetos simulados de apariencia real, que crea en el usuario la sensación de estar inmerso en él. Esto se consigue principalmente estimulando la vista y el oído, pero también se están introduciendo experiencias sensoriales que podemos percibir a través del olfato, tacto o incluso gusto.

Para el desarrollo del proyecto se ha elegido el robot humanoide llamado Reachy. Creado y diseñado por la empresa Pollen Robotics, la cual proporciona productos y aplicaciones accesibles y de código abierto que llevan la IA y la robótica a nuestra vida diaria (Robotics, About us, s.f.).

Reachy es un robot muy versátil, entre todas sus posibles aplicaciones se puede usar para ayudar a los estudiantes a aprender inteligencia artificial (IA), como prótesis de brazo y diferentes actividades referentes al cuidado de la salud o en eventos para distribuir mascarillas, dar la bienvenida, tomar la temperatura, realizar diversos juegos como tic-tac-toe y diversas actividades de cara al público que además supondrán el asombro de las personas al poder interactuar de una forma tan sencilla con un robot humanoide como este. (Robotics, What's Reachy, s.f.).

Dentro del abanico de posibilidades que Reachy nos permite aplicar, el proyecto se centrará en la implementación de dicho robot en la ayuda a personas dependientes mediante el control con realidad virtual. Es capaz de brindar esta ayuda en diversos entornos como hospitales o residencias, pero se quiere desarrollar principalmente en entornos domésticos, donde se pondrá a prueba su flexibilidad para realizar diferentes tareas.

1.2 Objetivos.

Objetivo general

El presente Trabajo fin de Grado tiene como objetivo principal la teleoperación de un robot humanoide en un entorno doméstico con la ayuda de la realidad virtual.

De este modo, se consigue mezclar estas dos tecnologías de constante expansión, para realizar el control de un robot con una mayor comodidad, pero a la vez con una

gran eficacia y flexibilidad. Ya que la realidad virtual nos permite mimetizarnos con el propio robot sin la necesidad de estar presente.

Para llevar a cabo esta implementación, se tendrán que tener en cuenta distintos aspectos:

- Elección del robot en base al entorno de trabajo, las tareas a realizar y la interacción con las personas a las que brindará asistencia.
- Elección del equipo de realidad virtual para realizar el control remoto de la manera más cómoda y eficaz posible.

Antes de llegar al principal objetivo del proyecto, se realizarán diferentes pruebas para conocer tanto el comportamiento del robot como del equipo de VR.

- Simulación de los movimientos de las extremidades del robot para conocer su alcance y limitaciones.
- Simulación de una cadena de movimientos mediante programación. En este punto se realizarán sobe todo movimientos expresivos para ver el afecto que puede transmitir el robot.
- Inclusión del equipo de realidad virtual en la simulación para ver el punto de vista desde el que se trabajará.

Una vez completados todos estos apartados, se conectará el equipo de realidad virtual al robot en cuestión para comprobar si todos estos comportamientos se pueden realizar en el mundo real.

1.3 Descripción de la memoria.

Una vez definidos los objetivos del proyecto, a la hora de abordar la memoria y la documentación del mismo, se puede seguir un orden y unos apartados bien diferenciados, que se corresponderán con los diferentes capítulos del presente texto.

En el primer capítulo, el presente, se ha realizado una introducción a la robótica y la relación que tendrá con la realidad virtual a lo largo del proyecto dando la capacidad de teleoperar un robot que también ha sido presentado en este apartado. Asimismo, se ha definido el objetivo principal y los objetivos secundarios de todo el proyecto

El segundo capítulo se enfoca en la robótica en nuestra sociedad, más concretamente la robótica de servicios que esta más estrechamente relacionada con los seres humanos y dentro de esta los robots humanoides que es el tipo de robot que se va a desarrollar a lo largo del proyecto.

En el tercer capítulo se realiza la descripción y análisis del robot humanoide Reachy, se estudian sus partes y las ventajas que estas le proporcionan, además de los motivos por los cuales ha sido el elegido para ser el protagonista de este Trabajo Fin de Grado.

El cuarto capítulo se enfoca en la realidad virtual, la disciplina que junto con la robótica son pilares fundamentales a lo largo de la memoria. Se realiza una introducción a esta tecnología y se estudian los distintos equipos que ayudan a que se produzca, entre ellos y con una mayor profundidad se destacan las gafas virtuales que se usarán para realizar la teleoperación.

En el quinto capítulo se describe la programación basada en componentes (ROS), la cual posteriormente permitirá el uso de ciertas herramientas para poder visualizar con facilidad las primeras simulaciones en el robot.

En el sexto capítulo se realizan distintos desarrollos que ayudarán a explorar las cualidades del robot. Inicialmente se hace una pequeña introducción a Unity, ya es la herramienta que va a posibilitar l desarrollo de varias simulaciones y la creación de una aplicación de teleoperación integrando la realidad virtual en ella. Mediante programación se consiguen unas simulaciones enfocadas principalmente en representar la expresividad que puede llegar a transmitir el robot a las personas con las que interactúe. Por otra parte, se detalla el diseño y desarrollo de la aplicación y la manera de integrar el equipo de realidad virtual en ella para producir una experiencia de control de Reachy. Por último, se explica mediante una primera aproximación cómo se podría realizar un control remoto mezclando varias herramientas como son Unity y ROS.

En el séptimo capítulo se hace una relación clara de los resultados obtenidos en los apartados anteriores, las pruebas realizadas con las diferentes simulaciones y una comprobación del grado de cumplimiento de los objetivos del proyecto.

El octavo capítulo consiste en un estudio del coste económico de todo el proyecto, explorando en detalle los costes del sistema adyacente incluido y el software desarrollado en este proyecto, los costes de personal y recursos utilizados en la elaboración de este, con el objetivo de ver la viabilidad económica.

Para terminar, en el capítulo noveno se hace una recapitulación general del proyecto desarrollado en su totalidad para enumerar las conclusiones obtenidas, determinar lo aprendido y las dificultades encontradas, así como proponer posibles mejoras y ampliaciones del mismo.

Por último y para concluir la memoria, se enumera la bibliografía consultada y referenciada a lo largo de la memoria.

Capítulo 2:
La robótica en la sociedad

2.1 Introducción.

A lo largo del tiempo, hemos visto como la tecnología se introduce en nuestras vidas y las cambia por completo. Estas nuevas implantaciones no siempre son bienvenidas como es el caso de la robótica. Desde que han aparecido los robots gran parte de la sociedad tiene perjuicios hacia este ámbito debido al famoso argumento: “los robots nos van a quitar nuestros trabajos”.

La robótica ha venido a evolucionar los trabajos como se ha estado haciendo a medida que la humanidad ha ido avanzando. La implantación de las nuevas tecnologías dejará muchas actividades obsoletas, pero creará muchos nuevos puestos de trabajo, incluso algunos que todavía no existen. No solo viene a sustituir ciertos trabajos, sino que los mejorará y no solo los trabajos, sino nuestras vidas en muchos aspectos.

Aunque se implanten cientos de robots en cientos de empleos, todos estos necesitaran personal que los controle, los revise, mantenga, programe y sea responsable de sus actos. Además, los seres humanos tenemos una capacidad de improvisación que las máquinas aún no tienen, por lo que siempre va a necesitarse personal formado. Por lo tanto, se quiera o no, las personas y los robots están condenados a entenderse y trabajar juntos.

2.2 Robótica de servicios.

Cuando se habla de robots, se suele enfocar a los robots industriales, aquellos que se centran en la realización o ayuda de tareas industriales, pero no solo existen este

tipo de robots. Un gran campo como ya se ha visto, son los robots de servicios, los cuales principalmente se centran en la ayuda y bienestar de los humanos.

Según la Federación Internacional de Robótica (IFR), "un robot de servicio es un robot que opera de forma parcial o totalmente autónoma, para realizar servicios útiles para el bienestar de los humanos y del equipamiento, con exclusión de las operaciones de fabricación." (Robotnik, 2022)

Dentro de este extenso campo se pueden hacer tres clasificaciones:

- En función de su interacción.
  - De servicio personal: son aquellos diseñados para el mantenimiento del hogar, ofrecer ayuda con tareas domésticas y hasta entretener a la familia. En resumen, es un robot que no realiza ninguna actividad comercial.
  - De servicio profesional: es un robot de servicio utilizado para tareas comerciales, generalmente operado por un operador entrenado.
  - Humanoide o androide: es un sistema robotizado desarrollado para simular la apariencia y la actitud humana.
- En función de la tarea a realizar.

  En esta clasificación nos podemos encontrar de múltiples funciones como doméstico, militar, de entretenimiento, médicos, espaciales, etc.
- En función de su autonomía.
  - Automáticos: Estos robots pueden tomar sus propias decisiones sin la necesidad de un ser humano.
  - Semi-automáticos: Tiene cierto grado de autonomía, pero siguen siendo controlados por un ser humano.
  - Teleoperados: Robots controlados a distancia y que necesitan ser controlados todo el tiempo por un ser humano.

Se piensa que los robots son para el futuro, pero tenemos ejemplos que se utilizan casi a diario, como puede ser el caso de Roomba, el robot aspirador creado por iRobot Corporation (ver en la figura 2.1), el cual es un robot de servicio personal, doméstico y automático. Ha ganado enorme popularidad al darnos la comodidad de tener el suelo limpio permanentemente casi sin darnos cuenta y sin requerir ningún esfuerzo.

Esto lo loga a través de unos sensores tanto táctiles, ópticos y acústicos (dependiendo el modelo y serie) que le permite, entre otras cosas, detectar obstáculos, acumulaciones de residuos en el suelo y desniveles pronunciados tales como escaleras. Realiza giros de 360 grados gracias a sus dos ruedas motrices independientes y ademas es capaz de limpiar debajo y alrededor de los muebles.

Adicionalmente, se le puede programar para realizar otras funciones más “creativas” mediante un ordenador y haciendo uso de la denominada "Roomba Open Interface".

Figura 2.1. Robot Roomba limpiando el suelo [1].

Otro ejemplo, no tan cotidiano en nuestras vidas, pero de mayor importancia para algunas es el robot quirúrgico Da Vinci, el cual es un robot de servicio profesional, médico y teleoperado como veremos a continuación. Es la técnica más sofisticada e innovadora de cirugía mínimamente invasiva disponible en la actualidad, un tratamiento que ofrece resultados iguales o mejores que la cirugía convencional debido a su mayor precisión y destreza, reduciendo el temblor y proporcionando una visión excepcionalmente clara de la anatomía del paciente.

El robot quirúrgico Da Vinci, como se pude observar en la figura 2.2, es una plataforma robótica sofisticada en la que el cirujano dirige los brazos del mismo desde una consola, por medio de controles manuales y pedales, utilizando un sistema de visión estereoscópico. (HM Hospitales, s.f.)

Figura 2.2. Ejemplo de utilización del robot Da Vinci [2].

Este robot permite realizar operaciones con mayor facilidad, precisión y sobre todo menor invasión lo que dará multitud de ventajas al paciente como una recuperación más temprana, un dolor postoperatorio menor, una reducción del impacto estético, etc. Da Vinci da una gran ayuda en un ámbito tan crucial para los humanos como la salud.

Nacido en el seno de Silicon Valley, a partir de patentes militares y desarrollado por la empresa californiana Intuitive Surgical Inc., el robot da Vinci se lanzó al mercado en 1999. Desde entonces acumula más de 3 millones de operaciones en todo el mundo.

Existen muchos robots de servicios que están en uso actualmente, pero si se habla de los robots con más interés y curiosidad tanto para la sociedad como para los desarrolladores, esos son sin ninguna duda los robots humanoides, aquellos que se parecen o se quieren parecer a los humanos.

2.3 Robot humanoide.

Estos robots despiertan la curiosidad de todos, debido sobre todo a su aspecto, cada vez se parecen más físicamente en muchos rasgos, sobre todo faciales, a los humanos.

Como ya sabemos un robot humanoide son aquellos que se diseñan específicamente para imitar la apariencia o el funcionamiento de los seres humanos. Para conseguir el mayor aspecto similar al humano, estos robots poseen en su mayoría elementos antropomórficos, como dos brazos, dos piernas e incluso un rostro de apariencia humana realista hecha a base de piel sintética de silicona.

En principio, el desarrollo de estos robots nació con el deseo de conseguir mejores ortesis y prótesis para humanos con distintas discapacidades.

Con el tiempo, el éxito y popularidad de los robots industriales fuero guiando a la robótica humanoide hacia la intención de crear maquinas con la capacidad de realizar tareas humanas.

La locomoción bípeda, asistencia inteligente, las interacciones sociales y la sustitución de personal humano en tareas peligrosas o repetitivas era el horizonte, y lo sigue siendo como objetivo principal.

Donde se centra el desarrollo de estos innovadores robots, principalmente es en la interacción social. Un ejemplo que impactó al mundo ha sido el conocido robot Sophia.

Dependiendo el entorno en el cual se vaya a interactuar, ciertos aspectos cobran más relevancia. En un contexto doméstico, los robots asistentes requieren un diseño amigable y funcional, para ello la alternativa humanoide presenta muchas ventajas.

Esto se realiza con la intención de que sean de agradable interacción y se puedan utilizar en la estructura existente de cualquier vivienda sin grandes modificaciones. Estas características son las que se han intentado emular en casi todos los prototipos conocidos hasta el momento. Uno de los más famosos referentes puede ser Pepper.

2.3.1 Sophia.

Sophia (figura 2.3) es la creación más importante hasta el momento del Dr. David Hanson, un brillante científico e innovador que se ha ganado la reputación de crear los robots más humanos del mundo desde su empresa, Hanson Robotics. (Arbeláez Buriticá, 2018).

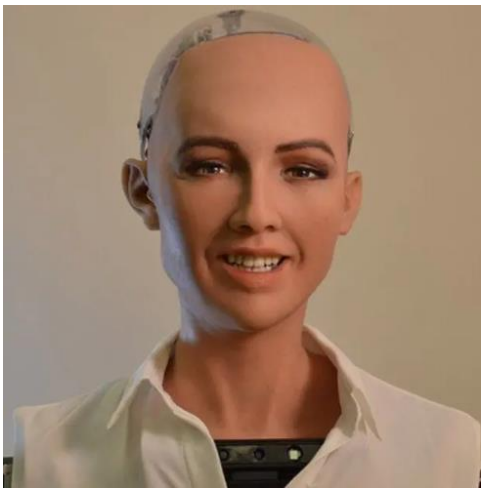

Figura 2.3. Aspecto del robot humanoide Sophia [3].

Sophia, creada en 2016, es un robot humanoide, pero no solo desde el punto de vista mecánico, sino también en el sentido de tener un cerebro de alto nivel de inteligencia artificial que le permite procesar lenguaje no estructurado. Es decir, más allá de responder o interactuar con algo ya programado, Sophia tiene la capacidad de aprender nuevas respuestas, incrementar su bagaje de conocimiento cada vez que interactúa con un ser humano, lo que es ejemplo básico de Inteligencia Artificial y aprendizaje autónomo o machine learning.

Además de poseer el procesamiento de datos que le permite mejorar sus respuestas con el tiempo, consta de reconocimiento tanto facial como de voz lo que le da un mayor comportamiento humano. Al igual que estos, puede reconocer, gracias a estas características, con quien habla o si con esa cara o voz ha interactuado anteriormente.

Sophia es capaz de sostener conversaciones, demostrar a través de su rostro gestos similares a los de las personas. La apariencia de Sophia es humanoide, es decir, tiene rostro, brazos, manos y, lo más atractivo, son sus gestos, los cuales le agregan un elemento diferenciador a la interacción. Esto se traduce en una variabilidad de su

comunicación donde evidencia emociones faciales, superficiales, sin dejar de ser lo que es, un robot de condiciones avanzadas y de apariencia muy humana.

“Estoy muy interesada en la tecnología y en el medio ambiente” (ver figura 2.4) dijo Sophia a un equipo de reporteros. “Siento que puedo ser una buena socia para los seres humanos [...], puedo ayudar a las personas a integrarse sin problemas y a aprovechar al máximo todas las nuevas herramientas y posibilidades tecnológicas que están disponibles actualmente”.
David Hanson, tiene la esperanza de que Sophia ayude a los humanos a romper las barreras sociales relacionadas con las interacciones con este tipo de robots. “Nuestro objetivo es que va a ser lo más consciente, creativa y capaz como cualquier ser humano", dijo Hanson. "Estamos diseñando estos robots para servir en el cuidado de la salud, terapia, educación y aplicaciones de servicio al cliente." (Naciones Unidas, s.f.)

Figura 2.4. Amina Mohammed, la vicesecretaria general de la ONU conversa con Sophia en 2017 [4].

Ha sido entrevistada en muchas ocasiones y en octubre del 2017, se convirtió en una ciudadana saudí, siendo así el primer robot con ciudadanía de un país.

Otra curiosidad sobre Sophia es que, además de haber sido el primer humanoide con inteligencia artificial avanzada que se ha creado en el planeta, fue desarrollada a imagen y semejanza de Audrey Hepburn, actriz y modelo de Hollywood.

2.3.2 Pepper.

Pepper (figura 2.5) es un robot humanoide, lanzado en junio de 2014, diseñado por SoftBank Robotics (anteriormente Aldebaran Robotics) que se creó por primera vez para las necesidades B2B, es decir para comercio entre empresas (business to

business), y luego adaptado para propósitos B2C, comercio para el consumidor (business to consumer). (Gelin, 2019)

Pepper es el primer robot humanoide del mundo capaz de identificar rostros y emociones humanas clave. Está diseñado para interactuar con los humanos de la forma más natural posible a través del dialogo de su pantalla táctil.

Destaca por su motor emocional, un desarrollo que permite que este robot estudie los gestos y el rostro de la persona que interactúa con él para tomar decisiones en la conversación que hagan sentir mejor a la persona, además puede analizar el tono de voz utilizando los últimos avances en reconocimiento de voz y emoción para provocar interacciones. No solo eso: su funcionamiento se basa también en la conexión a la nube.

Eso permite que ese conocimiento y aprendizaje de todos los robots Pepper sea compartido por ellos para mejorar sus respuestas y acciones en todo momento. (Pastor, 2015)

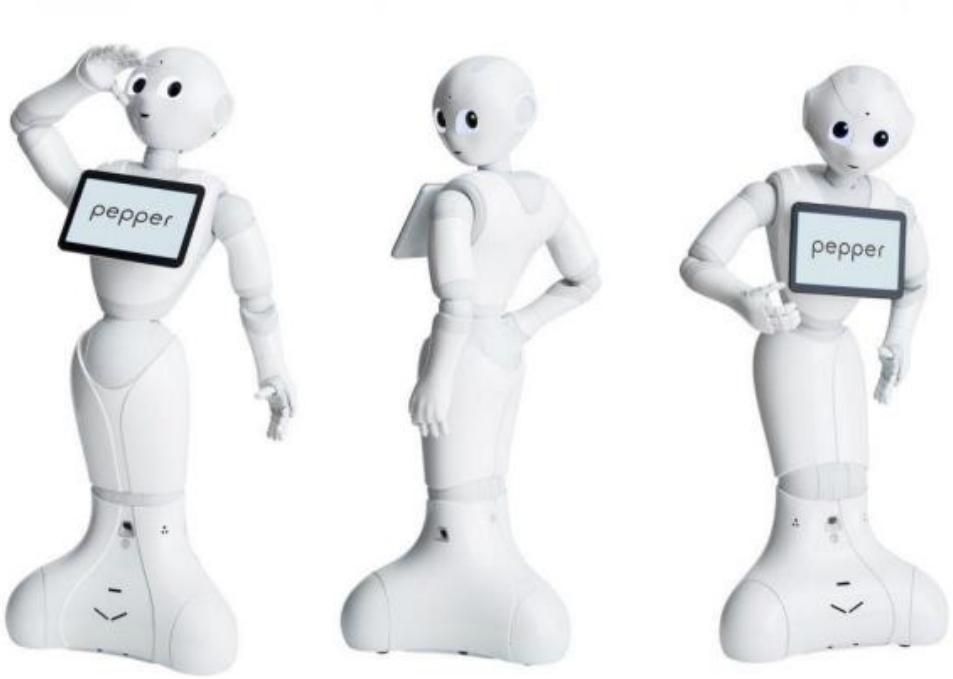

Figura 2.5. Robot Pepper [5].

El tamaño y el aspecto de la máquina apuntan a hacerla apropiado y aceptable en la vida diaria para interactuar con seres humanos. Pesa unos 28 kg y mide 1,2 m de altura, además de contar con una autonomía que sus creadores estiman en 14 horas. No tiene "piernas" sino que se mueve con un sistema de tres ruedas omnidireccionales.

El éxito de las ventas de Pepper fue significativo en Japón, donde las 1.000 unidades iniciales que Softbank puso a la venta en Japón se vendieron en apenas un minuto,

algo que demuestra el incipiente interés por este mercado. El precio final ascendía entrono a unos 8 000 euros, 1 425 euros iniciales a los que se le añadían 177 euros mensuales de tarifa, durante 3 años, para ofrecer el servicio de conexión a la nube que permite a Pepper mantener su aprendizaje con el resto de Peppers del mercado.

2.3.3 Toyota T-HR3.

T-HR3, el robot humanoide de tercera generación diseñado y desarrollado por la División de Robots Asistentes de Toyota para explorar nuevas tecnologías que permitan gestionar de forma segura las interacciones físicas entre los robots y su entorno, además de un nuevo sistema remoto de maniobras por el que el robot replica los movimientos del usuario, como se puede observar en la figura 2.6. (Toyota, 2017)

Figura 2.6. Robot Toyota T-HR3 siendo teleoperado [6].

El T-HR3 refleja el concepto amplio de cómo las tecnologías avanzadas pueden ayudar a satisfacer las necesidades de movilidad específicas de cada persona.

Toyota sigue su desarrollo y avisa que no se trata de un producto final, sino de un trabajo en desarrollo cuya meta es la de crear un robot similar a un humano que pueda aligerar la carga de algunos trabajos. Así, el T-HR3 tiene capacidades que pueden ayudar de forma segura a los humanos en diversos contextos, tales como en casa, en servicios médicos, en zonas de obras, en lugares golpeados por catástrofes o incluso en el espacio exterior.

El nuevo T-HR3 es a simple todo lo que la ciencia ficción ha estado soñando durante décadas. Dispone de dos brazos, dos piernas, es rápido y además en muy suave en sus movimientos. Es controlado remotamente desde una base donde una persona se sienta y usa unos brazos robóticos con una decena de sensores para que el robot las replique.

El sistema de control incluye unas HTC Vive para ver en 3D lo que el robot tiene delante, así los movimientos de las extremidades se pueden replicar de forma más confiable a la realidad del humano.

Toyota es una de las empresas que más dinero está invirtiendo en el mundo de la robótica para reemplazar algunos trabajos humanos. Y cuando decimos reemplazar nos referimos a que pueda ayudar a otros trabajadores en tareas que consuman mucha energía o sean peligrosas. Por ahora se muestra controlado por un humano a distancia e inalámbricamente, pero es obvio que la industria de la robótica se dirige a un futuro donde estos robots tendrán su propia inteligencia artificial para operar de forma independiente. (Contreras, 2017)

Capítulo 3:
El robot humanoide Reachy

3.1 Introducción.

Reachy es una plataforma humanoide expresiva de IA y robótica creación de la empresa Pollen Robotics (Pollen Robotics, s.f.). Al principio su objetivo únicamente era explorar aplicaciones de estos dos campos anteriormente mencionados, la robótica y la IA, pero con el tiempo se han ido introduciendo características que han posibilitado ampliar sus aplicaciones y los entornos de trabajo donde desarrollarlas.

Anteriormente se ha hablado de los robots humanoides en general y de ciertos ejemplos que han llamado la atención de la sociedad. Reachy tiene características de todos ellos, con Sophia y Pepper comparte la idea de trabajar en un entorno rodeado de humanos y por lo tanto quiere tener un aspecto similar a estos. Por eso Reachy consta de dos brazos con varias articulaciones muñeca, codo y hombro al igual que una persona, además de la cabeza con un peculiar cuello que le permite moverla en todas las direcciones deseadas.

También comparte con estos dos robots el objetivo de interactuar con las personas, pero no es su único objetivo y por lo tanto no está tan desarrollado en este ámbito. No puede mantener conversaciones, hablar o tomar decisiones por sí mismo. Pero puede trasmitir su actitud y emociones mediante expresiones y gestos como también lo hacen constantemente los humanos. Además, una característica muy particular en estos innovadores robots es el aprendizaje automático, lo cual hace que Reachy este en constante evolución.

Con relación al último ejemplo de robot humanoide del que se ha hablado, Reachy se asemeja al T-HR3 tanto en aspecto como en el objetivo que persigue. La parte superior del cuerpo de ambos es casi idéntica, y además ambos tiene la capacidad de ser teleoperados mediante realidad virtual. Esto les permite realizar un amplio abanico de movimientos y estar dotados para la manipulación de objetos. Pueden trabajar remotamente en distintos entornos, aunque el T-HR3 tiene capacidad para una mayor diversidad tanto de actividades a realizar. En cambio, el robot T-HR3 no está destinado a la interacción con las personas como es el caso de Reachy.

Reachy, como se ha presentado, engloba distintas características de varios robots humanoides, y todo ello debido a que no solo persigue un único objetivo, sino que quiere ser particularmente bueno tanto interactuando con las personas como manipulando objetos.

Esto permite a Rechy ser útil para variedad de funciones, ya sea servicio de alimentos, servicio al cliente, demostraciones, investigación y desarrollo. En las ilustraciones 3.1 y 3.2 observamos dos claros ejemplos de sus posibles aplicaciones.

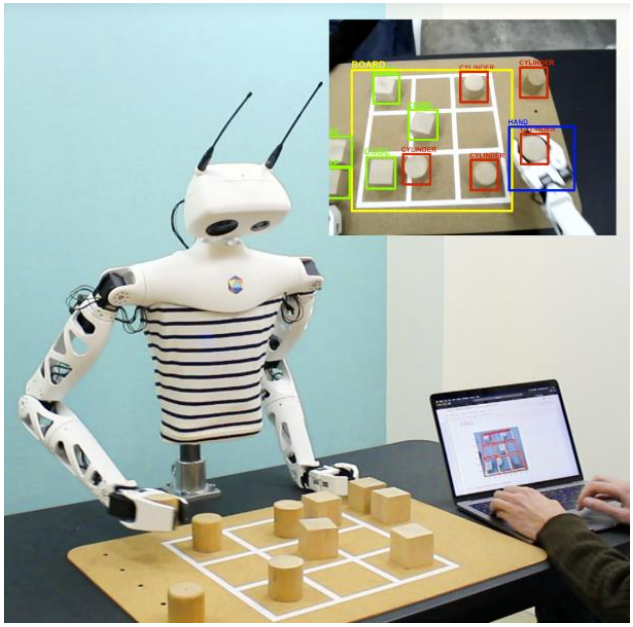

Figura 3.1. Reachy jugando al tic-tac-toe, utilizado como plataforma de IA [7].

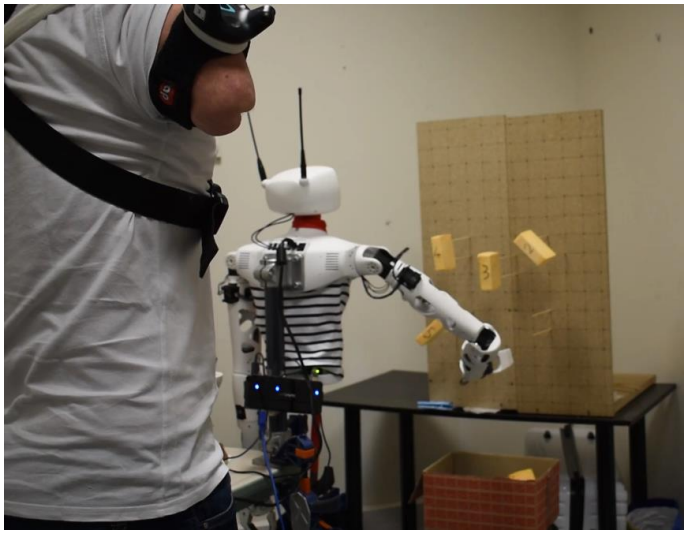

Figura 3.2. Participante amputado que controla el brazo Reachy con los movimientos de su muñón [7].

3.2 Elección de Reachy.

En la actualidad la diversidad de robots humanoides que se pueden encontrar es cada vez mayor. Por lo tanto, ¿Por qué se elige a Reachy entre todos los posibles robots?

Esto es una de las principales preguntas que nos podemos hacer, pero hay una clara respuesta basada en varias ventajas.

- Sus extremidades tienen una alta capacidad para manipular diferentes objetos a distintas distancias y altura.

- Una cabeza que nos permite observar un amplio rango a nuestro alrededor gracias a su particular cuello, lo que unido a sus extremidades le permite tener un aspecto muy similar al ser humano.

- Su principal distinción para la interacción con las personas son sus antenas. Parecen inútiles ya que no tienen una función en particular, pero son fundamentales para expresar emociones junto con los movimientos de su cabeza. Así resulta más afectivo y cercano con las personas de su entorno, lo que le hace ser diferente a otros muchos robots capaces de manipular objetos.

Al fin y al cabo, es un robot que estará continuamente acompañando a las personas y este punto de expresividad en muchas ocasiones puede hacer a sus acompañantes la vida más amena y cómoda. Es un aspecto que cada vez se tiene más en cuenta a la hora de desarrollar robots, ya que hay muchos estigmas en la sociedad contra los robots. Si se les da un afecto emocional puede parecer que también tienen sentimientos e ir cambiando ese pensamiento de la sociedad.

- Otra gran ventaja es que este robot se diseñó como una plataforma de código abierto para que los usuarios puedan crear con el robot prototipos de sus aplicaciones personalizadas del mundo real.

- Incorpora una aplicación de teleoperación, lo cual nos permitirá realizar nuestro control con el equipo de VR de una forma mucho más sencilla.

- Se pueden desarrollar simuladores que permiten trabajar con el robot simulado o real indistintamente. De esta manera, es posible reducir los costes de desarrollo e incrementar la seguridad.

- Incorpora inteligencia artificial, lo que ayuda a los desarrolladores a saltar directamente al fundamento de su investigación sin tener que entrenar primero el componente de aprendizaje automático.

- Puede incorporar una base móvil que le dota de movilidad omnidireccional, lo cual permite aumentar su alcance enormemente. Se le añade la capacidad de moverse en cualquier dirección, darse la vuelta y atravesar puertas, por lo tanto, cambiar a distintas habitaciones o estancias fácilmente.
  Esta base móvil, también es de código abierto y se puede controlar de varias maneras, entre ellas con la aplicación de VR que es lo que nos interesa.

Debido a todas las ventajas comentadas, es ideal para trabajar en distintos entornos, entre ellos el doméstico, debido a que se puede adaptar a la realización de distintas tareas. Además, una gran ventaja que ofrece es la facilidad de conexión compatible con un equipo de realidad virtual para realizar una avanzada teleoperación remota. De esta manera no es un robot que trabaje por de forma automática, sino que hay una persona detrás de él controlando sus movimientos y expresiones.

En este caso el robot no elimina totalmente el papel del humano, y tampoco lo queremos, debido a que inicialmente la interacción entre persona-robot se puede realizar de una manera más cómoda y natural si sabemos que el robot está directamente controlado por una persona. Más aún si con algunas personas de las que va a tratar son de avanzada edad que a veces no aceptan la tecnología por distintos factores como el miedo o la desconfianza.

3.3 Estructura física.

Reachy es un robot humanoide, aunque su estatura, contando la base móvil, es ligeramente superior a la mitad de un ser humano medio. Este tamaño más reducido es usual en este tipo de robots sociales y de servicios para que sea visto de forma amable y completamente inofensivo por las personas que interactúen con él.

Dependiendo con que objetivo se vaya a usar Reachy, se dispone de tres variantes:

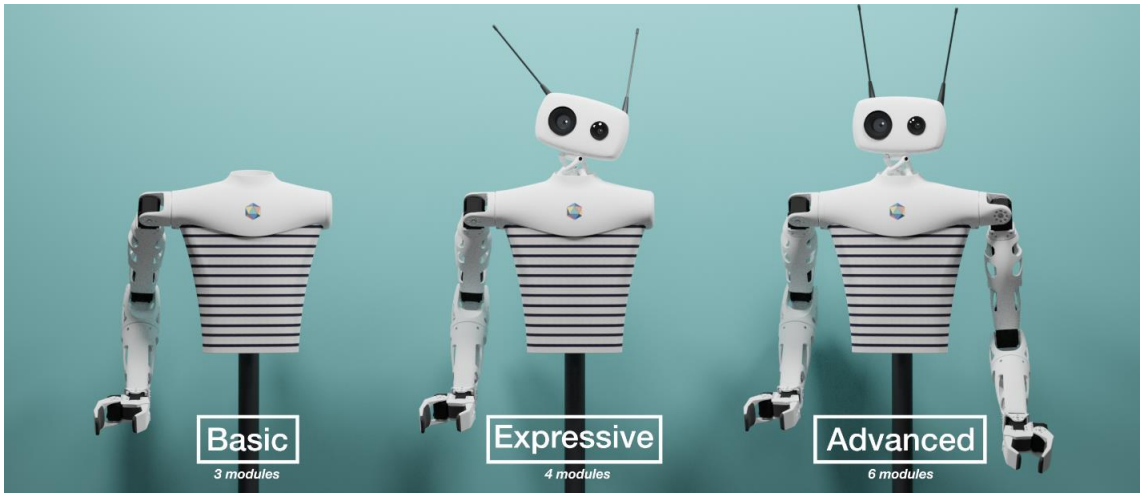

Figura 3.3. Variantes de Reachy [8].

Como se puede observar en la figura 3.3, la variante Basic se podría usar como prótesis de brazo, la opción Expressive principalmente se basa en la capacidad de interacción que permite expresar emociones o gestos con la cabeza y antenas, además te dotar al robot de la capacidad de teleoperación. Por último, la elección Advnaced engloba todas las anteriores además de permitir al robot la realización de un mayor abanico de actividades y movimientos.

Dentro de su estructura amigable podemos diferenciar distintos módulos que le dotan de varias funciones cada una.

3.3.1 Tronco.

El tronco, como en cualquier sistema, es la parte central y por lo tanto lo que sustenta y da robustez al equipo. En él no solo se sujetan las extremidades y el cuello, sino que también es la conexión para la base móvil.

Esta parte del cuerpo del robot puede ser totalmente personalizada para dar diferentes aspectos a Reachy, ver la figura 3.4.

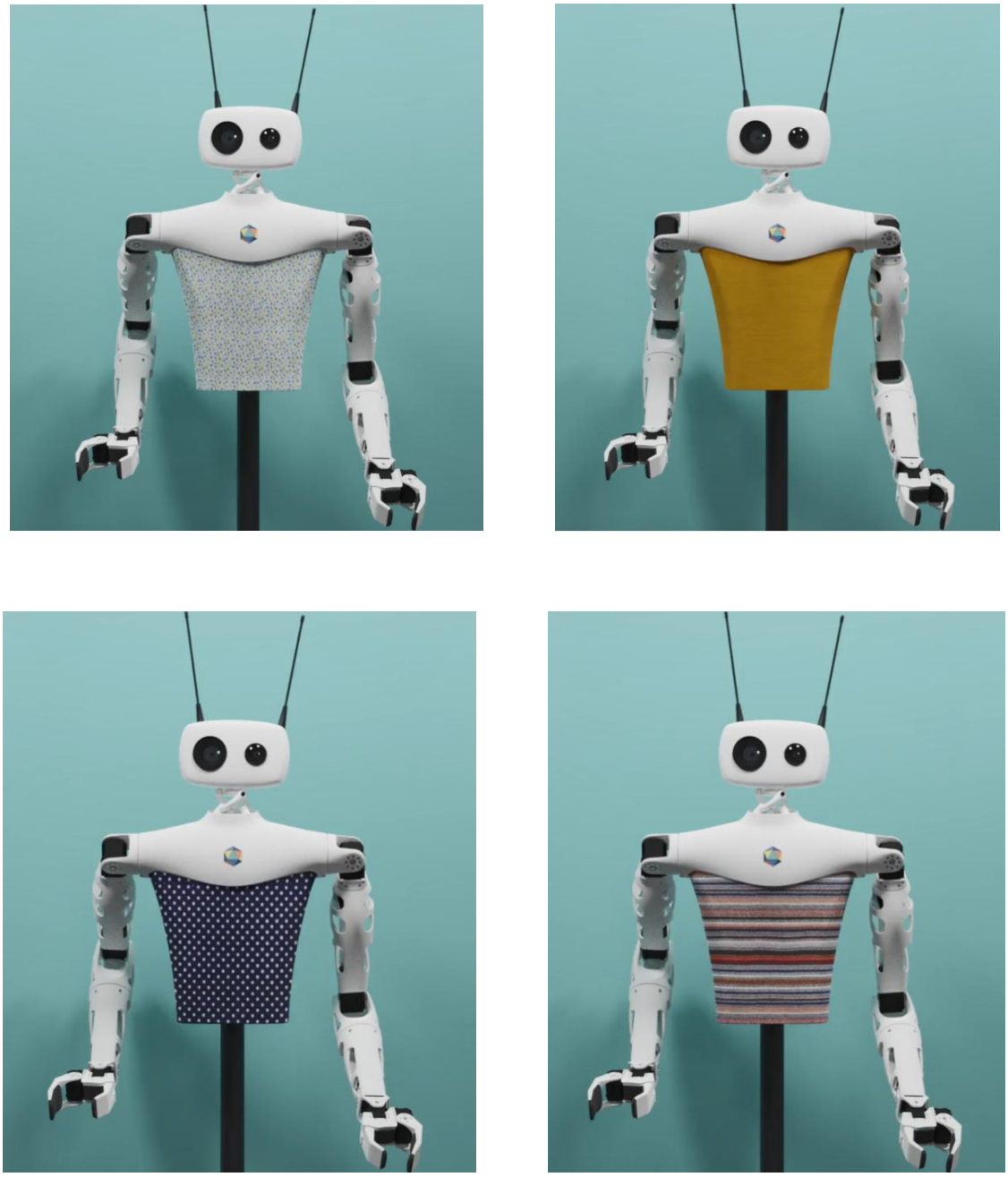

Figura 3.4. Diferentes modelos con los que se puede personalizar a Reachy [8].

3.3.2 Extremidades.

En el caso de las extremidades, como hemos visto en las distintas variantes, Reachy puede poseer uno o dos brazos. Los cuales son identicos y dotan al robot de la posibilidad de realizar distintas funciones.

Son los encargados de limitar el alcance del robot, este rango es bastante amplio pudiendo llegar a diferentes alturas y practicamente a cualquier objeto de su alrededor si es capaz de rotar el cuerpo.

Su gran alcance se debe a que los brazos de Reachy, inspirados biologicamente, tienen siete grados de libertad permitiendo muchas variantes de movimientos. Presentan dimensiones, proporciones y movimientos similares a los de un brazo humano adulto, un brazo puede levantar objetos de hasta 500 gramos y manipularlos con bastante destreza. Además se pueden equipar con una seie de manipuladores especiales, desde pinzas de agare hasta manos humanoides de cinco dedos para adaptarse mejor a la aplicación de destino como se puede observar en la figura 3.5. (Jose Francisco, 2020)

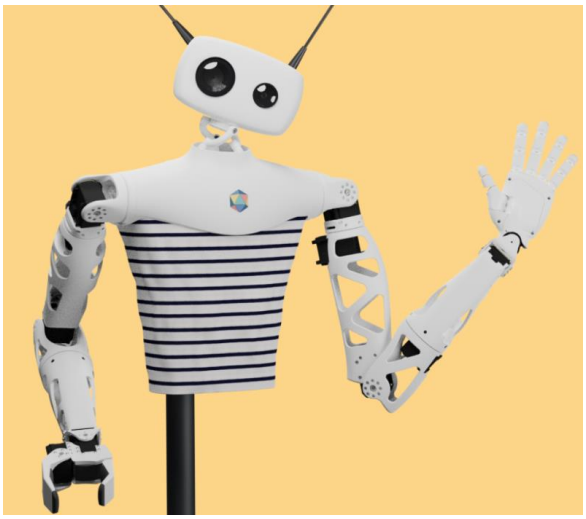

Figura 3.5. Reachy con una pinza de agarre en una extemidad y con una mano de cinco dedos en la otra extremidad [8].

Reachy viene con cinematica directa/inversa completa y generacion de trayectorias avanzadas. Por lo tanto, tiene la capacidad de manipular ciertos objetos y moverlos facilmente a una posición deseada, como se puede ver la evolución del movimiento en las figuas de la 3.6 a la 3.9.

Figura 3.6. Agarre de objeto mediante los brazos de Reachy y posición de inicio de la trayectoria [8].

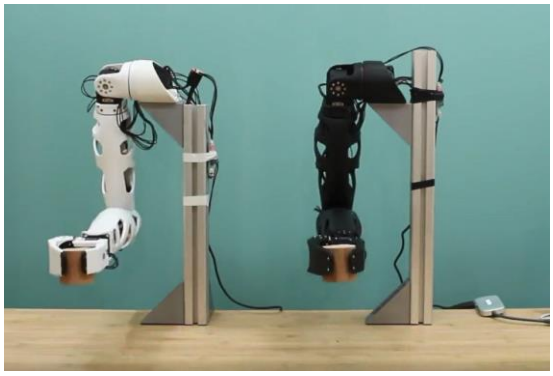

Figura 3.7. Los brazos de Reachy suben el objeto [8].

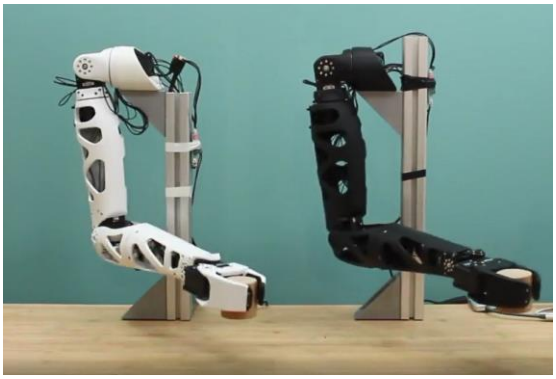

Figura 3.8. Los brazos de Reachy desplazan el objeto subido hacia la derecha *[8].*

Figura 3.9. Posición final de la trayectoria y descarga del objeto una vez bajado [8].

3.3.3 Cabeza y antenas.

La cabeza de Reachy principalmente se basa en su visión, consta de dos ojos, como los seres humanos, para que pueda ver la realidad en tres dimensiones. Esta visión dota, tanto a los seres humanos como al robot, de la capacidad de calcular distancias, percibir la profundidad o visualizar de forma eficaz objetos en movimiento, aunque esto es una función resultado de la combinación del cerebro y los dos ojos.

Gracias a esta visión artificial, cuando se utiliza un equipo de realidad virtual permite ver lo que ve Reachy de la misma manera que si la persona estuviese donde esta él (ver figura 3.10).

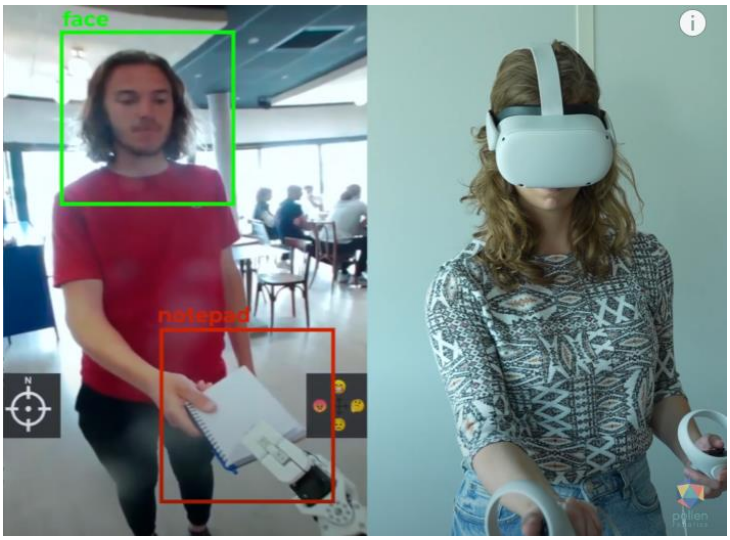

Figura 3.10. Perspectiva de la realidad vista desde el robot mediante las gafas de VR [8].

También es en esta parte de su cuerpo donde se integra la inteligencia artificial lo cual, acompañada de la visión, es lo que permite que el robot disponga de aprendizaje automático. De esta manera puede registrar y conocer la estructura del cuerpo humano, su posición en cada momento y ciertos rasgos faciales, reflejado en la ilustración 3.11. Mediante estos datos puede extraer información como el comportamiento o el estado de ánimo de una persona dependiendo su expresión corporal.

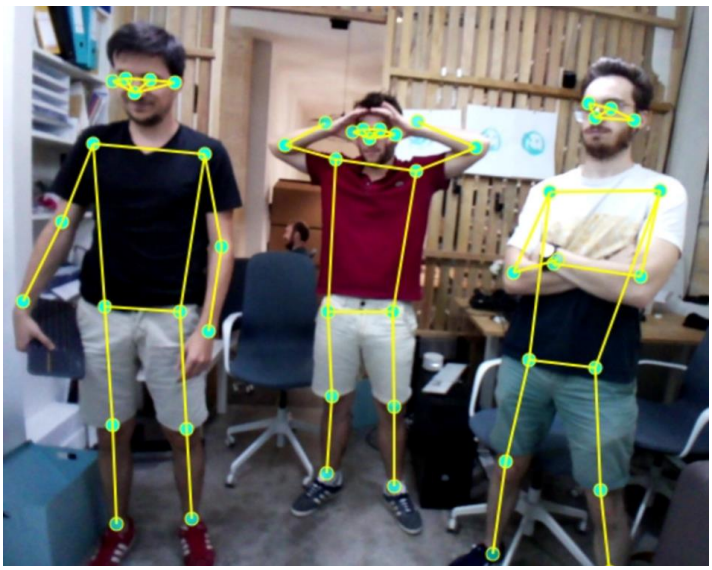

Figura 3.11. El robot reconoce personas mediante sus expresiones corporales y rasgos faciales [8].

En la cabeza de Reachy no solo alberga lo anteriormente mencionado, si algo destaca a primera vista son sus antenas.

Esta parte es fundamental sobre todo para mostrar las emociones mediante movimientos combinados con la cabeza. Reachy de esta forma se comunica con su usuario y genera empatía, de la misma manera que una mascota. Podemos observarlo en la imagen 3.12, aunque la expresión no se aprecia igual que en un video o en la vida real como podemos demostrar en las siguientes citas (Pollen Robotics, video1) (Pollen Robotics, video2).

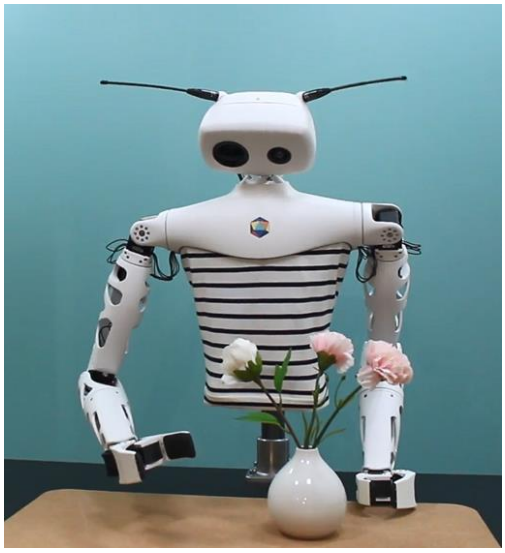

Figura 3.12. Reachy realizando expresiones para regalar una flor [8].

3.3.4 Cuello.

Desde la presentación de Reachy su diseño siempre ha sido el centro de todas las miradas. La gente lo encuentra expresivo, lindo y lleno de emociones. Dentro de este diseño el cuello cobra una gran importancia ya que es el que va a dar la movilidad a la cabeza y de ello va a depender tanto el rango de visión como la veracidad de su expresividad.

Primero, esto puede no ser obvio, pero el papel principal del cuello es mantener siempre la cabeza erguida. Pero no es el único, ya que como se ha dicho anteriormente la cabeza y su orientación juegan un papel central en la comunicación y transmisión de emociones. La cabeza inclinada hacia la izquierda puede significar empatía, dulzura ya veces seducción; doblar a la derecha, puede significar cuestionamiento o desacuerdo, etc. (Crampette, 2020)

Estos son ejemplos de rotaciones en un eje, pero transmitir cualquier emoción suele ser más complejo e implica las 3 rotaciones a la vez. Por lo tanto, el cuello de Reachy, basado en el actuador Orbita, posee los 3 grados de libertad de un cuello humano para permitir que muestre una gama completa de emociones.

3.3.4.1 Orbita.

El mecanismo de Orbita se basa en tres motores que accionan simultáneamente 3 discos.
Los discos son piezas cilíndricas con cojinetes en su interior para rodar bien unos sobre otros. También tienen un engranaje incluido que controla la velocidad y el par del motor a través de un piñón. Luego, los brazos se conectan a los discos para transmitir la rotación al efector final a través de tres puntos de conexión. El efector final puede ser una cabeza, una mano o lo que se quiera, siempre que tenga 3 orificios roscados para conectar los brazos de Orbita.
En resumen, como se puede ver en la figura 3.13, los motores hacen que los discos giren, luego los discos hacen que los brazos giren para accionar un efector final.

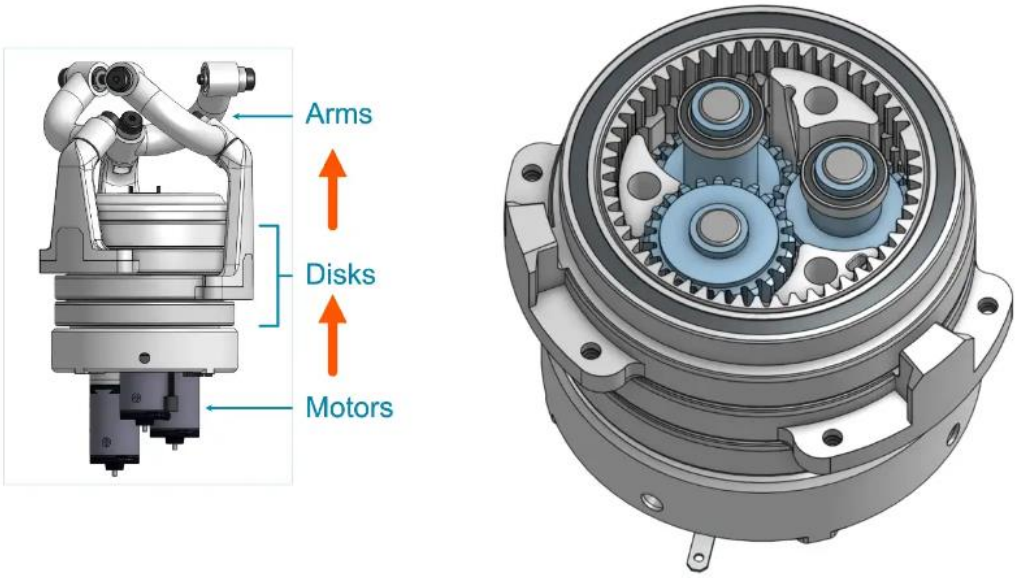

Figura 3.13. Componentes del mecanismo del actuador Orbita [9].

La particularidad de Orbita frente a otros actuadores comparables es que los engranajes están dentro del mecanismo y no fuera. Así, Orbita es más compacta, independiente de los elementos externos y se puede colocar en cualquier lugar.

Con Orbita, el cuello de Reachy es potente y preciso para que podamos controlar la orientación de la cabeza en las 3 dimensiones sin que nos cueste movernos rápidamente o apuntar a una posición precisa en el espacio. Además, el diseño ofrece una apariencia sofisticada y elegante a Reachy que atrae a las personas e invita a interactuar.

Es probable que las mismas características resulten útiles colocadas en otras articulaciones del robot. Cuando los humanos hablan para comunicarse, también mueven las manos de formas muy complejas y significativas. Por eso, ya se ha experimentado Orbita como muñeca (ver figura 3.14), además de estudiar la

posibilidad de implementar una versión más potente de Orbita para accionar un hombro. Estas mejoras mejorarían aún más el realismo del movimiento de Reachy, su rendimiento dinámico y, por supuesto, su belleza.

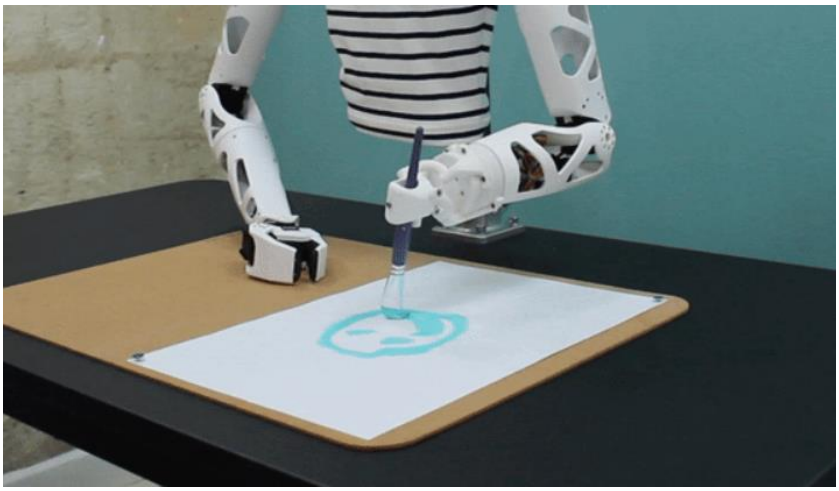

Figura 3.14. Reachy dibujando gracias a la implementación de Orbita en la muñeca [9].

3.3.5 Plataforma móvil.

Reachy tenía muchas funciones como robot como se ha podido ver a lo largo del proyecto, para enseñanza como plataforma de inteligencia artificial o para interactuar con personas tanto en su bienvenida a ciertos eventos como en acciones de entretenimiento como juegos, dentro de todo el abanico en el cual se podía utilizar.

Pero este abanico se amplía enormemente con una de sus últimas mejoras, la implantación de una base móvil que le hace ganar movilidad y poder desplazarse por distintos lugares como se ve en la figura 3.15. (Rouanet, 2022)

Figura 3.15. Reachy desplazándose por un pasillo gracias a su base móvil [10].

Esta base móvil destaca principalmente por su diseño exclusivo (atender figura 3.16). Mientras que la mayoría de las bases móviles robóticas del mercado son redondas y están equipadas con 2 ruedas estándar o cuadradas con 4 ruedas mecanum, se opta por centrarse en un diseño de forma redonda y un mecanismo de 3 ruedas omnidireccionales para aumentar la agilidad y la compacidad. Esto permite que Reachy se pueda mover de una forma sencilla en cualquier dirección desde cualquier posición y además tener un volumen más pequeño que productos similares.

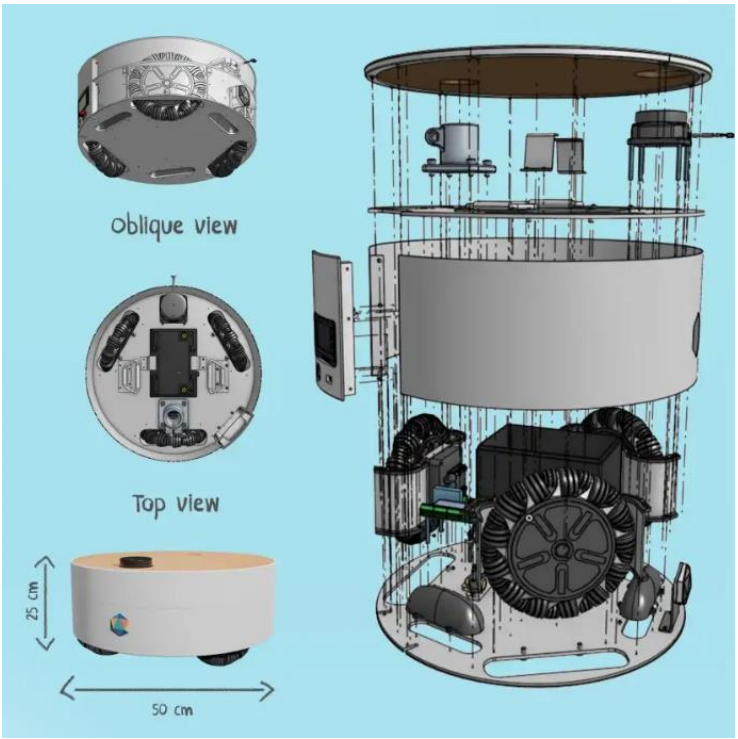

Figura 3.16. Vista oblicua, superior, lateral y explosionada de la base móvil robótica [10].

La base móvil de Reachy está equipada con un codificador y una unidad de medición inercial para cada rueda, y un LIDAR 2D de tiempo de vuelo (ToF), el cual es un método para medir la distancia entre un sensor y un objeto, basado en la diferencia de tiempo entre la emisión de una señal y su retorno al sensor, después de ser reflejado por un objeto (observar figura 3.17). La batería de gama alta integrada viene con una garantía de 5 a 10 años y ofrece 10 horas de uso.

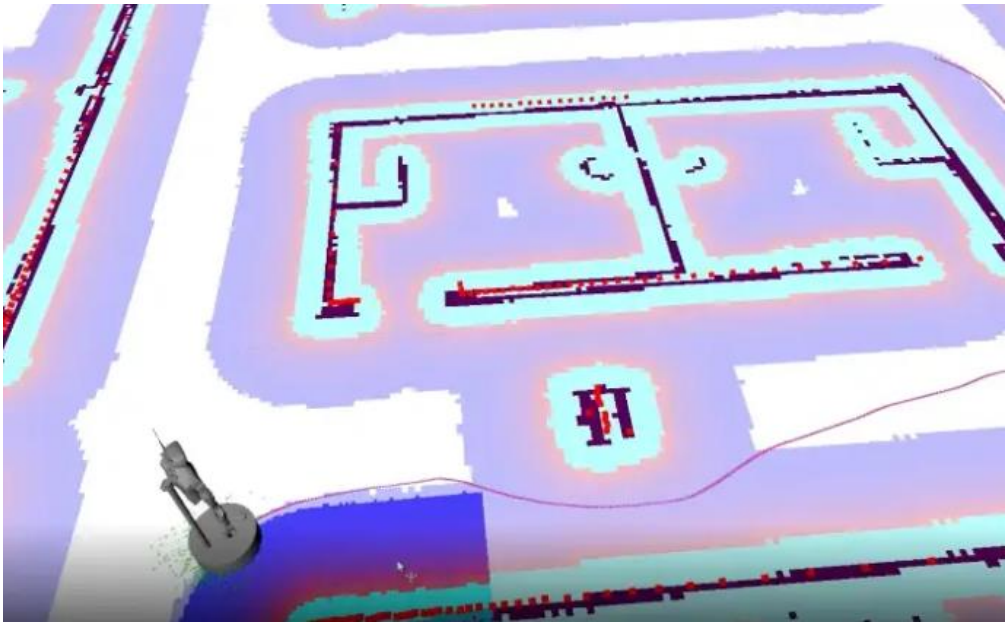

Figura 3.17. Navegación de alcance en un entorno mapeado LIDAR [10].

La base móvil de Reachy no solo destaca por su diseño, sino que tiene una característica de gran importancia, su software es de código abierto, lo cual permite que pueda ser modificado para que los usuarios puedan crear aplicaciones a su gusto.

Al igual que el robot, la base móvil se puede controlar desde Python SDK o la aplicación de teleoperación. Esta última función es lo que se busca en este proyecto, no solo controlar el robot con su visión y movimiento de extremidades, sino también su plataforma móvil, para poder desplazarse por las distintas habitaciones que puede tener una vivienda, y permitir brindar ayuda en distintos escenarios y con distintas tareas.

Capítulo 4:
Realidad virtual

4.1 Introducción.

La Realidad Virtual (VR) es un entorno de escenas y objetos de apariencia real, generado mediante tecnología informática, que crea en el usuario la sensación de estar inmerso en él.

A través de un dispositivo como gafas o cascos de realidad virtual podemos sumergirnos dentro de juegos donde nos creemos nuestros personajes o viajar por todo el mundo sin movernos del sofá del salón, pero a través de una experiencia totalmente realista. Además, permite la utilización de otros dispositivos como por ejemplo auriculares o espadas láser simuladas.

Esto, que puede parecer extremadamente futurista, no tiene un origen tan reciente como podríamos pensar. De hecho, muchos consideran que uno de los primeros dispositivos de Realidad Virtual fue la denominada Sensorama (atender figura 4.1), una máquina con asiento incorporado que reproducía películas en 3D, emanaba olores y generaba vibraciones para hacer la experiencia lo más vívida posible. El invento se remonta nada más y nada menos que a mediados de los años 50. A partir de ahí, el desarrollo tecnológico y de *software* en los años siguientes trajo consigo las evoluciones pertinentes tanto en dispositivos como en el diseño de interfaces. (Iberdrola, 2020)

Figura 4.1. Utilización de Sensorama en la década de los 50 [11].

Es bastante común confundir el término de Realidad Virtual con el de Realidad Aumentada.
La principal diferencia entre ambas es que la RV construye el mundo en el que nos sumergimos a través de unas gafas específicas. Se trata de un ámbito totalmente inmersivo y todo lo que vemos forma parte de un entorno construido de manera artificial a través de imágenes, sonidos, etc. Por su parte, en el caso de la Realidad Aumentada (RA), nuestro propio mundo se convierte en el soporte para colocar objetos, imágenes o similares. Todo lo que vemos está en un entorno real y puede que no sea estrictamente necesario usar gafas. El ejemplo más claro y cercano es el juego Pokemon Go.

Sin embargo, existe una combinación de ambas realidades denominada Realidad Mixta. Esta tecnología híbrida permite, por ejemplo, ver objetos virtuales en el mundo real y construir una experiencia en la que lo físico y lo digital sean prácticamente indistinguibles.

4.2 Equipo de realidad virtual.

Una vez introducido el ambiente de la realidad virtual, seguramente se tenga una clara pregunta, ¿Cómo se consigue la sensación de estar dentro de un espacio creado por ordenador? Necesitamos unas gafas o cascos específicos y el vídeo o dispositivo que crea el entorno. Las gafas nos aíslan de todo lo que nos rodean y amplían el campo de visión haciendo que parezca que la pantalla está alrededor y no solo delante de nosotros, aunque realmente no están envolviéndonos en 360 grados. Para engañarnos, utilizan dos imágenes diferentes en cada ojo. Más que dos imágenes, son dos ángulos diferentes de la misma escena. Esto hace que se simule profundidad y

que, sin gafas, los vídeos dedicados a esta tecnología los veas como doble o en dos trozos diferenciados. (GR, 2022)

Además, se utilizan sensores específicos directamente en las gafas que permiten detectar si nosotros nos giramos para conseguir así que el espacio gire con nosotros.

Los requisitos para probar la realidad virtual dependerán de lo que busquemos. Es decir, puede que baste un teléfono móvil y unas gafas de cartón para lo más básico, pero puede que te haga falta todo un entorno preparado y potente con unas buenas gafas de VR para lo más elaborado.

4.2.1 Gafas para móviles.

Como se atisba en la figura 4.2, se puede probar la experiencia desde el aspecto más básico si tenemos un smartphone y unas gafas asequibles de cartón tipo Google Cardboard o cualquier otro dispositivo barato que permita introducir el teléfono en su interior y cuyo precio ronde los 15 o 20 euros.

Figura 4.2. Equipo sencillo de VR compuesto por un smartphone y unas gafas de cartón [12].

Esto hará que no necesites ningún hardware específico más allá del casco o gafas en sí.

4.2.2 Gafas VR para gaming.

El principal uso y motivo por el cual se han viralizado estos dispositivos es por su uso en los videojuegos. Sin embargo, los gafas VR tienen muchos más empleos. En el sector inmobiliario permiten realizar visitas virtuales a un inmueble, también se utilizan para hacer visitas virtuales a museos. Se emplean cada vez más en el ámbito

de la educación. En el caso de la medicina, se emplean para la formación médica y operaciones quirúrgicas. Y, por supuesto, también se usan en la industria, especialmente en el diseño industrial. (Carbone, s.f.)

Podemos distinguir dos formatos básicos:

Gafas de realidad virtual independientes: pueden ser usados sin necesidad de estar conectados a ningún dispositivo externo; es decir, son inalámbricos. Las mismas gafas son las que soportan todo el peso gráfico y las aplicaciones se descargan directamente al dispositivo. En su mayoría, los gráficos suelen ser de una calidad baja, aunque tiene un precio más accesible.

Gafas de realidad virtual conectados: Por otro lado, existen las gafas de VR que necesitan estar conectadas a un dispositivo para poder funcionar. Ya sea un ordenador, una consola de videojuegos o, incluso, tu celular, siempre será necesario que estos lentes de realidad virtual se encuentren conectados. Esto va a permitir que el artefacto soporte los elevados requisitos gráficos, por lo cual también tienen un costo mayor.

4.2.2.1 Oculus VR.

Los años 80 y 90 del siglo XX fueron las décadas de oro de los videojuegos de la saga Arcade. Se lanzaron varios intentos de realidad virtual, aunque todos fracasaron.
La revolución del siglo XXI, llega cuando el joven californiano Palmer Luckey vuelve a poner de actualidad a la realidad virtual, al conseguir construir un prototipo de gafas financiadas a través de la plataforma de crowdfunding Kickstarter.
Su prototipo Oculus Rift de 2012 (observar en la figura 4.3) asociaba la visión de volumetría 3D en un entorno 360º, de una forma envolvente. La compañía Oculus VR fue comprada por Facebook en 2014 por 2 billones de dólares.

Figura 4.3. Oculus Rift original, el primer auricular VR comercial lanzado por Oculus VR [12].

Los controladores de movimiento de Oculus Touch (ver figura 4.4) se lanzaron oficialmente en diciembre de 2016 por $ 199, por otra parte, a inicio de este año las gafas de VR se lanzaron a un precio de 599$. En octubre de 2017, Oculus redujo el precio del paquete Rift + Touch a $ 399. De esta manera ya no solo era la experiencia visual ofrecida por las gafas Oculus Rift, sino también se podía recrear los gestos y movimientos generados por las manos gracias a los controladores.

Figura 4.4. Controladores Oculus Touch de primera generación [12].

El siguiente producto que lanzaron fue en octubre de 2017, unas gafas de realidad virtual independientes llamadas Oculus Go y por únicamente 199$. El dispositivo no necesita estar conectado a un dispositivo como ordenador, smartphone o consola para funcionar, dando una libertad al usuario impensable hasta el momento en la realidad virtual. Como se puede observar en la figura 4.5, dispone únicamente de un controlador con tan solo dos botones físicos (no tiene control de volumen de audio) y el gatillo en la zona trasera. El tracking en este tipo de visores no permite la interacción en el eje Z de profundidad (cerca y lejos), así que únicamente podremos interactuar con los ejes X (arriba y abajo) y los ejes Y (izquierda y derecha) y sus combinaciones de rotación. Estas limitaciones se deben a la intención de abaratar costes en el producto y así hacerlo más asequible. (knob2001, 2018)

Figura 4.5. Gafas virtuales independientes Oculus Go y su único controlador [13].

En septiembre de 2018 se anuncia la revolución en la realidad virtual, las Oculus Quest a un precio de 399$, un equipo sencillo e independiente, por lo tanto, sin PC ni cables ni configuraciones liosas, de calidad contrastada en cada uno de los eslabones (hardware, software y experiencia de usuario), con un ecosistema reconocible, cargado de personalidad y que, como explica Facebook, es por fin una VR accesible. Vuelve a los dos controladores que disponían las Rift para una mayor experiencia virtual, pero con una principal modificación, los anillos de infrarrojos, como se observa en la figura 4.6, se movieron a la parte superior del dispositivo, para garantizar su visibilidad desde las cámaras de seguimiento de los auriculares. (hmong, s.f.)

Figura 4.6. Gafas virtuales independientes Oculus Quest y sus dos controladores de segunda generación [14].

En 2019, Oculus VR lanza un nuevo casco de realidad virtual, Oculus Rift S, el remplazo directo del Rift original, con el mismo precio que este. El nuevo auricular

requiere una PC como el Rift original y todos los juegos disponibles para el Rift original deberían funcionar sin problemas en el Rift S. Sin embargo, es un auricular completamente nuevo con hardware diferente, que cuenta con seguimiento interno y diferentes controladores.

La última incorporación a estos dispositivos son las gafas virtuales Meta Quest 2 a un precio de 299$ desde su lanzamiento en octubre del año 2020. Como se puede observar se ha cambiado el nombre de Oculus a Meta, esto es debido al cambio de nombre que ha experimentado Facebook para verse más cercano al metaverso.

Lo primero que llama la atención es el cambio de color de todo el branding (ver en la figura 4.7). Esto no es ninguna tontería para un equipo que pretende llevar la VR a mucha más gente de la que nunca ha llevado. Un estándar en el mundillo del marketing dice que la *gente toma decisiones sobre un producto durante los primeros 90 segundos de interacción, y de ellos, del 62% al 90% las toma basados solo en el color.* El blanco, está presente desde hace unos cuantos años en la tecnología. Dicen los expertos que representa sencillez, un nuevo comienzo.

Figura 4.7. Gafas virtuales independientes Meta Quest 2 y sus dos controladores [14].

Si bien este dispositivo se parece mucho a su predecesor, mejora la mayoría de los aspectos. El diseño evoluciona siendo más cómodo gracias a un mejor ajuste, accesible y fácil de usar, no solo se queda en estos aspectos, sino que también reduce su peso y mejora el audio. Sus especificaciones también experimentan cambios, siempre a mejor, 50% más de RAM y el doble o incluso más si se desea de almacenamiento. Además de una mejora visual, con una mayor tasa de refresco de imagen (de 72hz a 120Hz) y una gran cantidad de pixeles adicionales (casi 2k de resolución para cada ojo).

La mayor potencia de la Quest 2 da la posibilidad a los desarrolladores de una mayor libertad a la hora de crear juegos, al tiempo que permita a los usuarios disfrutar aún más de Oculus Link. Esta última función mencionada permite conectar el dispositivo al PC y jugar a juegos de RV para PC desde la tienda de Oculus o a través de Steam si así lo deseas, opción que en las anteriores gafas estaba en fase beta.

Como ultima mejora cabe destacar, como se puede atender en la figura 4.8, el nuevo diseño de los controladores y el aumento de la batería, cuatro veces más que los controladores Quest originales, sin comprometer sus capacidades de seguimiento. Además, cuenta con numerosos sistemas inteligentes de gestión de la batería para maximizar su duración, tanto en el auricular como en los mandos. Esto incluye ajustes para poner el Quest 2 en reposo cuando no está en uso y un ingenioso sistema que enciende automáticamente los mandos cuando te pones los auriculares y los coges. (Willings, 2022)

Figura 4.8. Comparación entre los controladores de las Oculus Quest y las Meta Quest 2 [14].

4.2.3 Simuladores.

Los simuladores no son generalmente dispositivos para tener en casa sino pensados para salas específicas, para lugares donde hay una gran instalación.

Están formados por todo un equipo que te hace vivir la experiencia de una forma más realista, y en los cuales no pueden faltar unas gafas virtuales de gran potencia si de verdad se quiere estar inmerso en una nueva realidad.

Se pueden observar ejemplos típicos como conducir un formula 1 (figura 4.9), en el cual se dispone de todo el equipo de un piloto como asiento, volante, marchas o pedales

además de una enorme pantalla y para un mayor disfrute se añadiría unas gafas de realidad virtual.

Figura 4.9. Equipo de simulación de piloto de carreras [15].

También se puede observar simuladores más complejos que ofrecen experiencias que maximizan las sensaciones ofrecidas por las gafas de realidad virtual, permiten al usuario entrar en un entorno mucho más inmersivo que los típicos mandos de RV, a la vez que hacen desaparecer la sensación de mareo.

Con los simuladores KAT Walk y Virtuix Omni, el usuario puede moverse y caminar en entornos de realidad virtual sin la necesidad de emplear mandos y sin moverse del sitio. Su elevado precio puede suponer un obstáculo para su compra, pero, sin embargo, el alquiler de estos simuladores de realidad virtual es una gran opción para disfrutar de las experiencias en VR más intensas.

La principal diferencia entre estos dos simuladores es el agarre que experimenta la persona. En el KAT Walk, figura 4.10, se utiliza una cinta de correr omnidimensional que permite al usuario caminar, correr e incluso agacharse dentro del entorno virtual, es capaz de soportar hasta 100 kg de peso y ofrece total libertad de movimiento a 360º.

Figura 4.10. Estructura del simulador KAT Walk [15].

El Virtuix Omni, como se puede observar en la figura 4.11 a diferencia del anterior, tiene un anillo que rodea la cintura del usuario y del que el usuario se cuelga gracias a un arnés especial. Este anillo está lleno de sensores de movimiento que permiten captar la rotación y la dirección, mientras que la cinta omnidireccional solo se encarga de captar la cantidad de desplazamiento. Gracias a este sistema dual, el Omni cuenta con una precisión de movimientos sin precedentes, aunque la ausencia de anclaje de seguridad impide que el jugador pueda agacharse.

Figura 4.11. Estructura y uso del simulador Virtuix Omni [15].

4.3 Elección del equipo de VR.

Una vez introducida la realidad virtual y los distintos métodos y dispositivos que hay disponibles para utilizar esta tecnología innovadora. Se tiene que poner en contexto el proyecto que se va a realizar y elegir la forma de teleoperar nuestro robot.

De las distintas plataformas que se han visto anteriormente, las gafas para móviles no tienen la suficiente potencia ni los recursos para poder controlar un robot. En el otro extremo nos encontramos los simuladores, con los cuales podríamos controlar perfectamente a Reachy, pero su alto coste limita la posibilidad de implantarlo en este proyecto.

Otra opción es usar unas gafas de realidad virtual para gaming, aunque como ya hemos visto no solo se utilizan en este sector. Estos dispositivos permiten, gracias a las gafas, introducirse “dentro” de Reachy y ver lo que él ve en todo momento desde sus cámaras, además de controlar su cabeza y antenas permitiendo mostrar emociones. También disponen de otro gran elemento, los controladores Oculus Touch, gracias a los cuales se podrá mimetizarse con las pinzas de Reachy y de esta forma mover el brazo de Reachy con el seguimiento de los controladores VR permitiendo manipular objetos o realizar diversas tareas como abrir puertas o armarios.

Dentro de los distintos fabricantes que existen, se elige Oculus VR debido a su amplio recorrido en la realidad virtual y sus diferentes dispositivos de los que dispone y los cuales ha ido mejorando a medida que han avanzado. Debido a esta explicación, las gafas virtuales más convenientes para el control de Reachy serán la Meta Quest 2. Este dispositivo es el más desarrollados y el que dispone de las ultimas mejoras, además son independientes lo cual da una gran comodidad al evitar tener que estar conectados mediante cable a otro dispositivo. Tanto sus gafas como sus controladores disponen de una mayor batería y de un uso más sencillo y accesible. Esta facilidad de manejo e intuitiva es de gran importancia ya que Reachy se quiere que sea teleoperado por cualquier persona, sin necesidad de un experto, posiblemente de alguien relacionado con la persona a ayudar, como un familiar o amigo sin necesidad de grandes conocimientos en realidad virtual ni tecnología.

Los controladores, son los encargados de simular las manos de Reachy y gracias a estos se puede dotar de múltiples funciones al robot. Únicamente moviendo dichos dispositivos a la posición deseada, el robot consigue adaptar el movimiento de sus extremidades y articulaciones de una forma óptima para darnos esta accesibilidad. Desde estos dispositivos también se puede controlar tanto la base móvil del robot, desplazándola en cualquier dirección, como las antenas para dar expresividad. Estas últimas funciones son gracias a los joysticks que tienen incluidos ambos mandos, con el del controlador izquierdo se mueve la plataforma del robot tanto translación como rotación (función que se cambia en el botón del lateral de ese mismo

controlador) y con el joystick del controlador derecho se puede mover las antenas como se observa en la figura 4.12.

La elección de este equipo de realidad virtual también viene predispuesta por la facilidad de implementación que ofrece Reachy de estos dispositivos al disponer de una aplicación de teleoperación mediante VR. Únicamente se tiene que cumplir ciertos requisitos de compatibilidad y aprender las buenas prácticas para usar de una manera adecuada tanto el equipo de realidad virtual como el robot.

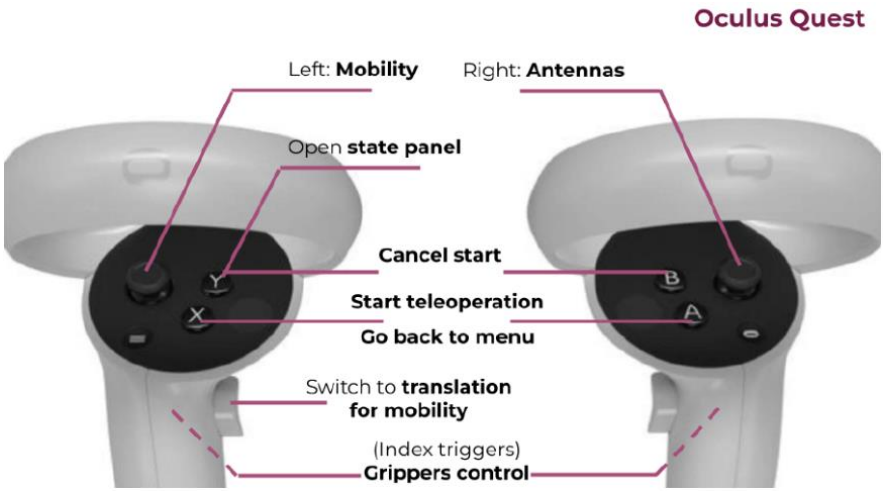

Figura 4.12. Información de los distintos botones de los controladores de las Meta Quest 2 [16].

Capítulo 5:
Programación basada en componentes (ROS)

En el mundo de la robótica, y en las nuevas tecnologías en general, las simulaciones son fundamentales antes de probar el funcionamiento de los dispositivos reales.

Las simulaciones principalmente surgieron debido al alto capital de inversión que es necesario en la mayoría de proyectos. El desarrollo robótico, al igual que lo han sufrido diversas tecnologías, tiene un inicio con muchos impedimentos en primera instancia, como el coste o la cantidad de habilidades que requiere (software, electrónica, mecánica…). La fabricación e implantación de unas primeras e innovadoras máquinas es caro y lo podemos ver con el ejemplo del automóvil o más recientemente con el Roomba. La adquisición de las primeras unidades tiene un alcance limitado, pero a medida que se adopta por la sociedad y el desarrollo evoluciona, estos costes se pueden abaratar y hacer el producto más accesible.

Por estos motivos cuando se trabaja con robots, raramente se hace directamente, debido a que es un dispositivo de alto valor y en el cual muchas acciones prueba-error pueden suponer su deterioro.

Por lo tanto, para probar las capacidades y las distintas ideas en el robot, lo más inteligente son las simulaciones. Es la manera más económica y rápida de realizar ensayos de los diversos comportamientos que puede tener un robot en diferentes entornos sin tener un gran impacto económico. Gracias a las simulaciones se puede llevar al robot a situaciones extremas y comprobar así sus límites sin causar ningún daño real al robot ni correr ningún peligro tanto económico (por deterioro del robot) como físico (por exposición humana a ciertos comportamientos del robot).

5.1 Introducción.

Hasta hace unos pocos años, los desarrollos en robótica, más propios de grandes empresas, estaban ligados a entornos software propietarios y cerrados, que obligaban a aquel que quería trabajar con un robot de una marca a usar su propio software con licencia. Como máximo, al margen de esto, se lograban crear en algunas importantes universidades pequeños conjuntos de marcos y herramientas para facilitar la programación de algunos robots, pero nunca nada de forma masiva.

Sin embargo, todos ellos han quedado desplazados, al menos en los desarrollos robóticos fuera de grandes empresas y corporaciones, por ROS, que son las siglas de Robot Operating System (Sistema Operativo Robot).

ROS es un meta sistema operativo de código abierto para los robots. Proporciona bibliotecas y herramientas que ayudan a los desarrolladores de software a crear aplicaciones en Robótica.

El objetivo primario de ROS es soportar el re uso de código en la investigación y desarrollo dentro de la robótica. ROS es una estructura distribuida de procesos llamados Nodos que permite el diseño individualizado: pero fácilmente acoplable a los demás procesos. Estos procesos se pueden agrupar en Paquetes, que fácilmente pueden ser intercambiados, compartidos y distribuidos (ver figura 5.1). (Riva, 2021)

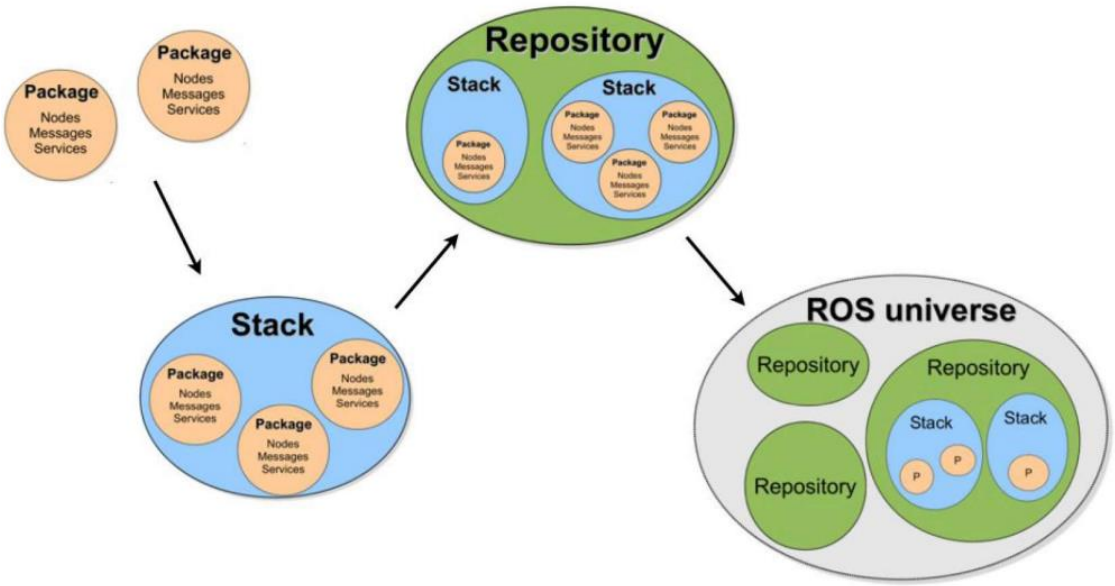

Figura 5.1. Esquema del funcionamiento de la estructura de ROS.

A este principal objetivo de compartir y colaborar, se le pueden sumar otros varios objetivos:

- ROS está diseñado para ser lo más liviano posible, de modo que el código escrito para ROS se pueda usar con otros frameworks o estructuras de

desarrollo de software para robots. Una prueba de esto es que ROS es fácil de integrar con otros marcos de software de robot: ROS ya se ha integrado con OpenRAVE, Orocos y Player.

- El modelo de desarrollo preferido es escribir bibliotecas agnósticas en ROS con interfaces funcionales limpias.
- ROS es fácil de implementar en cualquier lenguaje de programación moderno. Ya está implementado en Python, C++ y Lisp, y tenemos bibliotecas experimentales en Java y Lua.
- ROS tiene un marco de prueba de integración llamado rostest que facilita la activación y desactivación de dispositivos de prueba.
- ROS es apropiado para grandes sistemas de rutinas de ejecución y para grandes procesos de desarrollo.

Una gran ventaja de la que dispone ROS es la gran comunidad de usuarios que ha formado con más de 10 años de experiencia. Unido a su gran expansión, facilita su desarrollo e implantación en muchos robots debido a que la comunidad desarrolladora comparte gran parte de los programas que hacen y esto posibilita la reutilización de muchos de ellos con pequeñas adaptaciones, para no tener que desarrollar desde cero en cada robot funciones iguales a las que ya otro robot tiene implantadas.

ROS se publica en forma de distribuciones, las cuales están asociadas a distintas versiones de Ubuntu, que es el sistema operativo donde se ejecuta. Cada distribución tiene un ciclo de vida como podemos ver a continuación en la figura 5.2.

Distribución	Lanzamiento	Fin ciclo de vida	Logo	Versión de Ubuntu
Noetic Ninjemys	Mayo 2020	Mayo 2025		Ubuntu 20.04 LTS
Melodic Morenia	Mayo 2018	Mayo 2023		Ubuntu 18.04 LTS
Kinetic Kame	Mayo 2016	Mayo 2021		Ubuntu 16.04 LTS

Figura 5.2. Tabla de las distintas distribuciones de ROS.

Para la ejecución de ROS en este proyecto se empleará la distribución Noetic Ninjemys asociada a Ubuntu 20.04 LTS.

5.2 MoveIt.

MoveIt es un marco de planificación de movimiento, se ejecuta sobre ROS y proporciona funcionalidad para facilitar la manipulación. MoveIt es capaz de calcular trayectorias entre una serie de puntos, cinemática directa e inversa y planificar trayectorias complejas evitando colisiones.

MoveIt incluye diversas utilidades que aceleran el trabajo con sistemas robóticos, y ayuda a no estar continuamente reinventando, siguiendo la filosofía de ROS de reutilización de código.

Comenzar con MoveIt es muy sencillo, ya que el paquete incluye un asistente donde el usuario sólo proporciona el archivo URDF (Unified Robot Description Format). Todo el resto de información sobre la cadena, restricciones, masas y direcciones del eje se definen a su vez de forma automática por la información proporcionada en el URDF.

Para su instalación Pollen Robotics facilita el URDF de Reachy y por lo tanto únicamente tendremos que ejecutar varios comandos en la terminal de Ubuntu 20.04 una vez instalado ROS Noetic. El primer comando es para instalar MoveIt propiamente dicho.

sudo apt install ros-noetic-moveit

El siguiente paso es configurar el entorno ROS para que se pueda compilar con éxito el espacio de trabajo ROS (por ejemplo ~/catkin_ws). Por lo tanto, se accede al espacio de trabajo catkin_ws y se instala la configuración de MoveIt desde la terminal.

cd ~/catkin_ws/src && git clone
https://github.com/aubrune/reachy_moveit_config

Seguidamente, también en el espacio de trabajo, se genera el URDF de Reachy donde se especifican las dimensiones del robot, los movimientos de las articulaciones, parámetros físicos como masa e inercia, etc.

cd ~/catkin_ws/src && git clone https://github.com/aubrune/reachy_description

Cuando el entorno de trabajo ya está configurado, se compila de la siguiente manera.

cd ~/catkin_ws && catkin_make

Por último, para que ROS pueda utilizar los paquetes que hay en el entorno de trabajo catkin_ws ejecutamos el siguiente comando.

source ~/catkin_ws/devel/setup.bash

Una vez preparado el espacio de trabajo ROS se podrá ejecutar MoveIt como un comando de ROS.

roslaunch reachy_moveit_config demo.launch

Como se observa en la figura 5.3, una vez lanzado este comando aparece un programa llamado RViz en el que se puede ver el simulador de Reachy listo para trabajar con él.

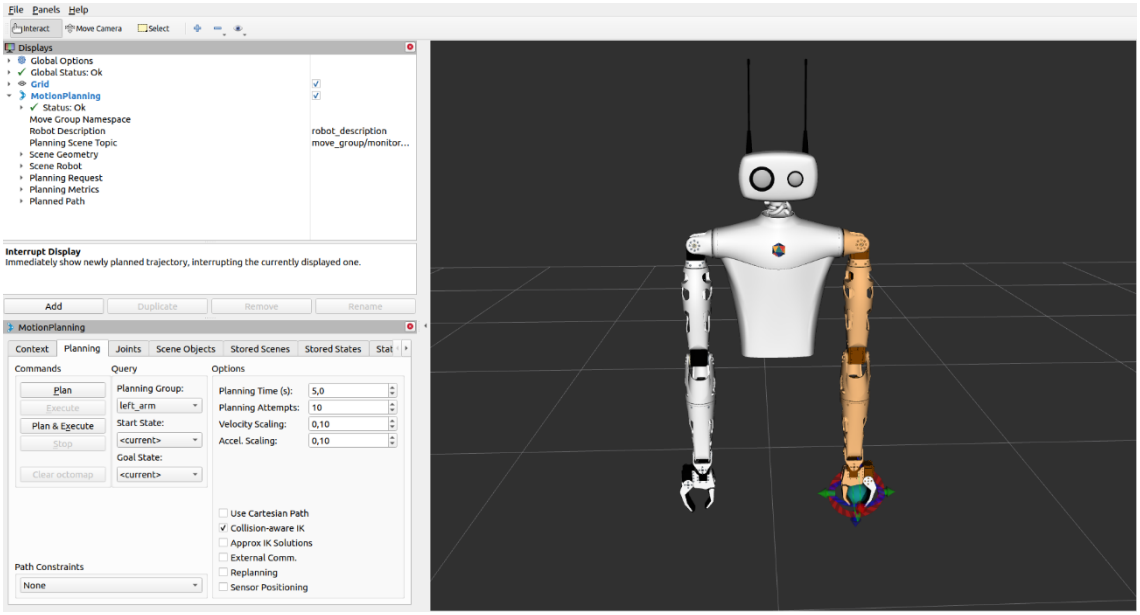

Figura 5.3. Inicio del visualizador RViz.

5.3 Simulaciones con RViz.

RViz es una herramienta de visualización disponible en ROS que posibilita que prácticamente cualquier plataforma robótica pueda ser representada en imagen 3D en diversos tipos de forma fácil y muy personalizable.

En el proyecto se quiere utilizar esta herramienta para realizar simulaciones de los movimientos de las articulaciones de Reachy para ver su alcance y sus limitaciones. Además, se pueden realizar trayectorias complejas y observar el recorrido que realizan para llegar al punto de destino marcado.

Se puede realizar movimientos de los dos brazos o de la cabeza, para seleccionar la parte que se quiere mover hay que dirgirse a *MotionPlanning -> Planning -> Query -> Planning Group* como se puede examinar en la figura 5.4.

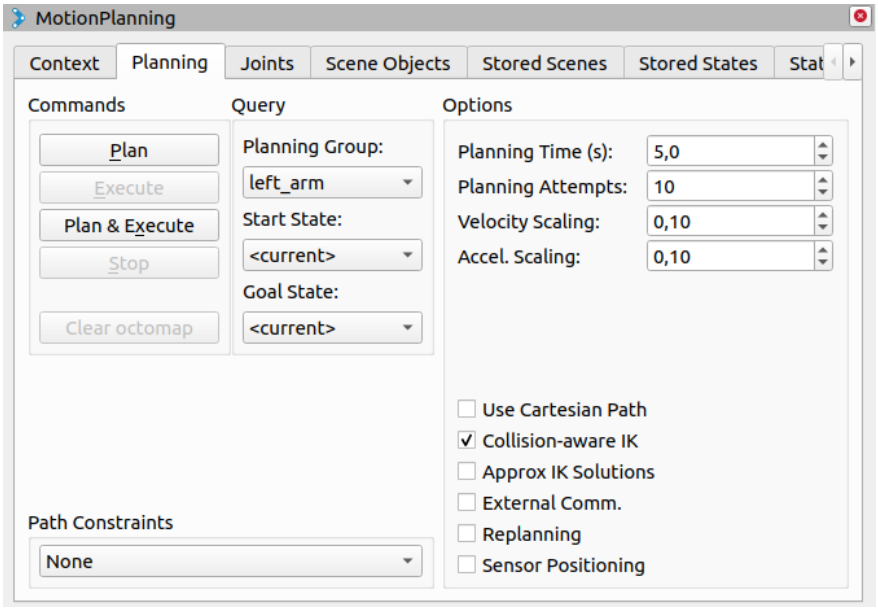

Figura 5.4. Ventana MotionPlanning desde la cual se realiza diversos ajustes o eleeciones para la simulación.

Una vez seleccionada la parte a planificar, si es uno de los brazos únicamente se tiene que mover la bola azul que posee Reachy en su brazo a la posición deseada. Puede ayudar cambiar el punto de vista del robot (ver figura 5.6) para realizar el movimiento de mejor manera dependiendo el plano en el que se vaya a realizar.

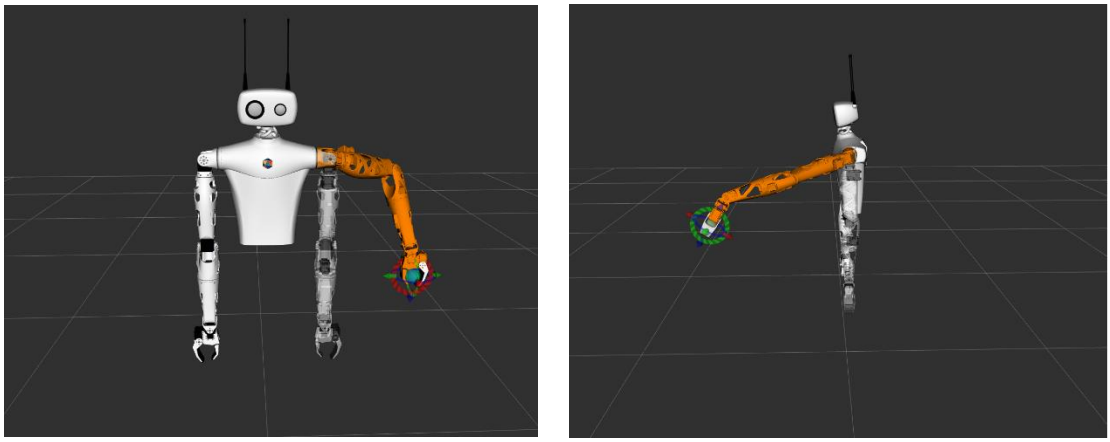

Figura 5.6. Movimiento del brazo izquierdo de Reachy desde distintas perspectivas.

Con el brazo en la posición deseada, se puede girar la muñeca en los 3 ejes, con los círculos de colores que rodean la bola azul, para colocar la pinza en la orientación deseada como se observa a continuación en la figura 5.7.

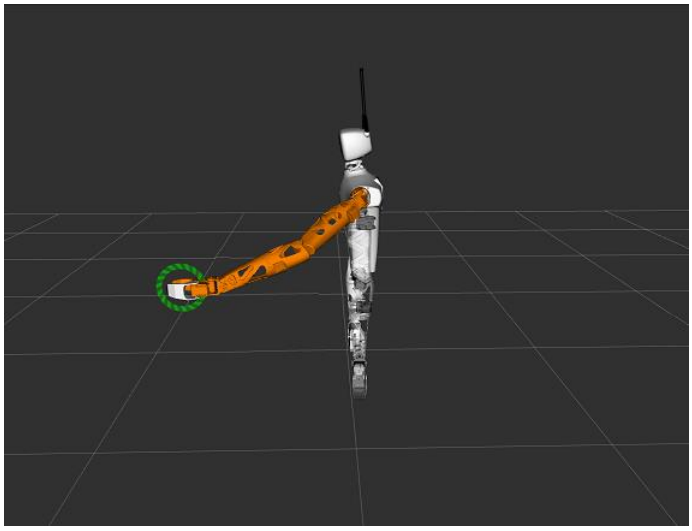

Figura 5.7. Giro de la muñeca de Reachy para colocar la pinza correctamente.

Existe la posibilidad de realizar una trayectoria lineal de la pinza, usando las fechas de colores que rodean la bola azul se consigue mover en cualquiera de los ejes como se observa en la figura 5.8. De esta manera únicamente moviendo la pinza, el brazo realiza los movimientos óptimos de sus articulaciones para llegar a la posición deseada.

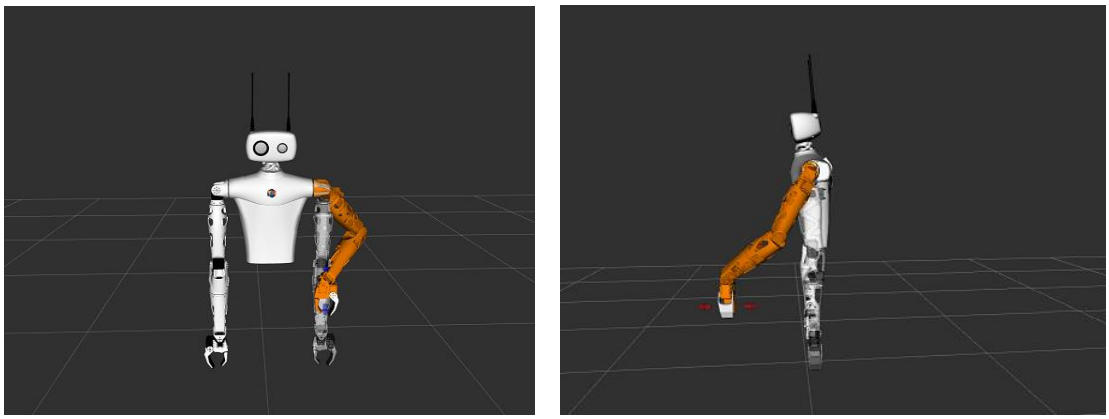

Figura 5.8. Movimientos lineales de la pinza de Reachy desde distintos ejes.

Cuando se tiene marcado el punto de destino, hay que dirgirse a *MotionPlanning -> Planning -> Commands* y ejecutar la orden "*Plan & Execute*" para observar cómo se realiza la trayectoria desde la posición inicial a la deseada. Tan pronto como se realice, se tendrá una nueva posición inicial desde la que se puede realizar otro movimiento.

A diferencia de los brazos, la cabeza tiene menor recorrido como cabe esperar y solamente se puede simular giros en sus 3 ejes como se observa en la siguiente figura 5.9, que realizándolos seguidamente se puede simular cualquier posición de la cabeza como en una cabeza humana ocurre.

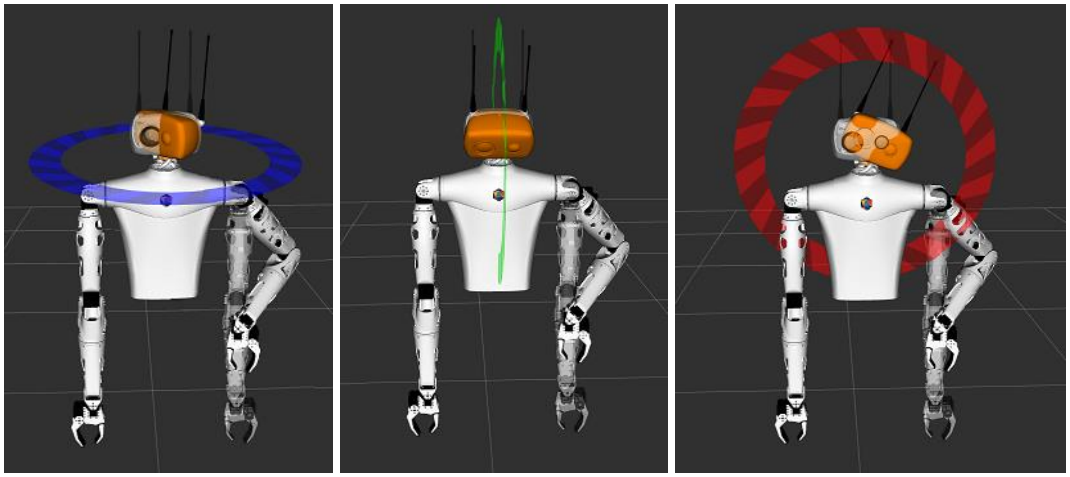

Figura 5.9. Giros de la cabeza de Reachy en los 3 posibles ejes.

Cuando se planea un movimiento en el robot, son ordenes que se mandan al simulador para que las ejecute, pero puede que esas órdenes no sean las correctas y ocurra colisiones entre las partes del robot. Este planificador y visualizador de movimiento, permite detectar y mostrar esas colisiones como se puede examinar en la figura 5.10. Si es así, no permite ejecutar la orden "*Plan & Execute*" ya que si estuviese conectado al robot real podría producir daños en él.

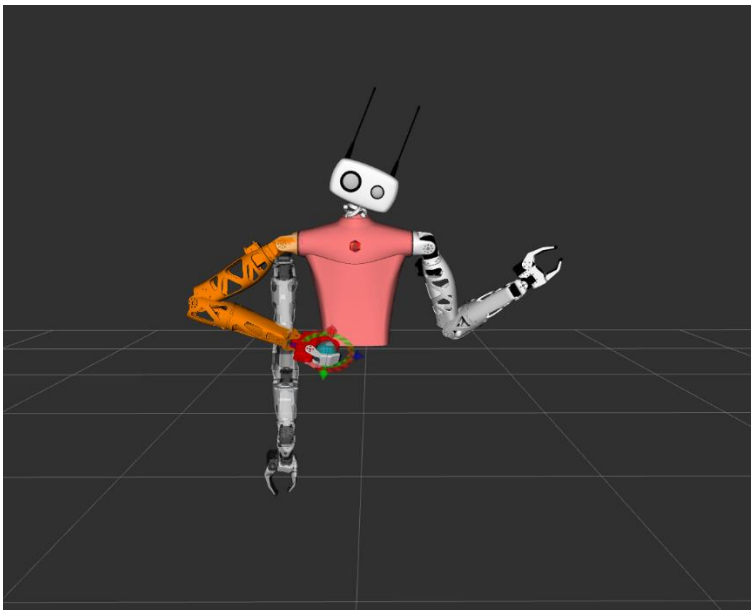

Figura 5.10. Colisión detectada entre la pinza del brazo derecho y el torso de Reachy.

Capítulo 6:
Desarrollo del proyecto

Los desarrolladores de proyectos necesitan varias herramientas a la hora de crear. Una de las más importantes son los entornos de desarrollo, el cual permite facilitar las tareas del programador con varios servicios integrados.

Su objetivo principal es maximizar el rendimiento del desarrollador, ofreciendo servicios integrales en un solo programa. Esto es sumamente útil porque en ocasiones se deben utilizar distintas herramientas para compilarlos o implementarlos y otras para depurarlos.

Esto también reduce considerablemente la inversión del proyecto además de dotar de una mayor flexibilidad a la hora de crear. Por otro lado, al contar con sistemas de aprendizaje muy intuitivos, permite que la brecha entre aprender cómo utilizarlo y empezar el proyecto sea cada vez más corta, aumentando el tiempo de productividad.

6.1 Introducción a la plataforma Unity.

Unity es un motor de desarrollo o motor de juegos creada por la empresa Unity Technologies, este término hace referencia a un software el cual tiene una serie de rutinas de programación que permiten el diseño, la creación y el funcionamiento de un entorno interactivo, o sea, de un videojuego.

Es una herramienta centrada en la creación de videojuegos, pero no solo es utilizada para crear estos, también se ha utilizado para experiencias de realidad virtual interactivas e incluso miniseries. Lo cual es un punto importante a tratar, ya que en este proyecto la realidad virtual tiene gran peso, y este motor de desarrollo nos permitirá realizar simulaciones utilizando un equipo de VR.

Una característica importante y muy cómoda de Unity es que soporta la exportación a una cantidad enorme de plataformas. No solo podemos elegir la plataforma con la que vamos a trabajar creando y editando nuestro juego, cuyo editor en este momento soporta Windows y MacOS, además de Linux de forma experimental, sino que podemos crear nuestro juego para más de 25 plataformas, entre las que podemos encontrar Smart TV, consolas, dispositivos móviles, webs o dispositivos de realidad extendida como las Oculus Quest. Esto nos va a permitir crear nuestro juego, por ejemplo, para Windows, y de forma relativamente sencilla, exportarlo hacia otros.

Uno de los grandes puntos fuertes que tiene Unity es la gran comunidad de usuarios que tiene. Esto permite tener acceso a multitud de documentación, foros y comunidades donde se preguntan y resuelven dudas, donde se explican diferentes métodos y técnicas nuevas, etc. Además, es uno de los motores predilectos para aprender a desarrollar videojuegos; ya que supone una puerta de acceso perfecta para aquellos que quieren incursionar en esta industria. (MasterD, s.f.)

Dentro de las principales características que tiene un motor de videojuego como Unity se encuentran:

- Motor gráfico para renderizar gráficos 2D y 3D
- Motor físico que simule las leyes de la física
- Animaciones
- Audio
- Inteligencia Artificial
- Programación o scripting, en el caso de Unity en C#
- Arquitectura orientada a componentes

Estas características y muchas más permiten conseguir resultados totalmente profesionales pudiendo progresar en cualquier aspecto que se desee. Además, dispone de servicios que complementan la creación del videojuego como obtener analíticas sobre cómo es el juego de los usuarios, monetizar el videojuego (en la mayoría de los casos con anuncios) o implementar la función multijugador de manera fácil para enriquecer la experiencia de juego.

Para lo que se quiere utilizar Unity en este proyecto es para simular el comportamiento de Reachy en una escena doméstica, su movimiento por ella, y la implantación de la realidad virtual en la perspectiva del robot. Debido a la amplia flexibilidad que permite este motor de desarrollo, también se realizarán, mediante programación, ciertos movimientos y emociones de Reachy para observar como de expresivo puede llegar a ser.

6.2 Simulación mediante SDK de Python.

6.2.1 Introducción a Python SDK.

SDK significa Kit de Desarrollo de Sistema y ayuda en la programación de aplicaciones móviles reuniendo un grupo de herramientas en un solo lugar. Python SDK permite al usuario escribir códigos que administran los recursos de OCI (Oracle Cloud Infrastructure), un juego de servicios en la nube complementarios que le permiten compilar y ejecutar una gama de aplicaciones y servicios en un entorno alojado de alta disponibilidad. El objetivo principal del SDK de Python es proporcionar un entorno Python fácil de usar. Además, permite a los desarrolladores autenticar a los usuarios para recuperar datos o cargar archivos.

Python es un lenguaje de programación multipropósito, es decir, que puedes utilizar este lenguaje para crear diferentes aplicaciones y trabajar en distintos sectores dentro del desarrollo, como en el desarrollo del software.

Este lenguaje, además de ser eficiente, fácil de aprender y poderse ejecutar en muchas plataformas diferentes, tiene varios beneficios:

- Permite que los desarrolladores sean más productivos, ya que pueden escribir un programa de Python con menos líneas de código en comparación con muchos otros lenguajes.
- Cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.
- Los desarrolladores pueden utilizar Python fácilmente con otros lenguajes de programación conocidos, como Java, C y C++.
- La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo. Si se presenta un problema, puede obtener soporte rápido de la comunidad.
- Python se puede trasladar a través de diferentes sistemas operativos de computadora, como Windows, macOS, Linux y Unix.

Debido a estos beneficios y a la facilidad de Pollen Robotics con su SDK de Python, es el lenguaje que se utiliza para practicar distintos movimientos de Reachy.

El SDK de Python le permite controlar y programar fácilmente un robot Reachy. Se utiliza para leer información del sensor (p. ej., sensor de fuerza, imagen de la cámara o posición de la articulación) y enviar comandos del actuador (p. ej., posición objetivo). (Goff, 2021)

Permite comenzar a controlar al robot en unas pocas líneas de códigos, no nos tenemos que centrar en problemas de sincronización de hardware y así podemos crear rápidamente prototipos o cualquier interacción.

Su instalación es muy sencilla debido a que el SDK es una biblioteca de Python pura. Lo único que se necesita para que sea compatible es tener inalada una versión de Python >= 3.6 y funciona en Windows/Mac/Linux. Para instalar el Python SDK se tiene que ejecutar el siguiente comando desde el símbolo del sistema (cmd) de Windows en este caso.

pip3 install reachy-sdk

El SDK se basa en algunos paquetes de Python de terceros, los cuales se instalan al instalar el SDK. Numpy, principalmente para el cálculo de la trayectoria; opencv, para acceso al marco de la cámara; grpc, para conectarse al robot.

6.2.2 Simulación Python SDK.

Para que la simulación sea más realista y este en un entorno más adecuado con el proyecto, se crea una escena simple de una cocina como se observa en la figura 6.1, para introducir el robot y sus comportamientos en una situación cotidiana que podrá realizar.

Figura 6.1. Reachy en una escena de una cocina creada en Unity.

Antes de proceder a la programación, se debe realizar la conexión con el robot. Una vez que está el proyecto en ejecución en Unity, (se puede comprobar porque debido a la gravedad los brazos del robot están hacia abajo) se ejecuta el comando *python* en el cmd y seguidamente las líneas:

from reachy_sdk import ReachySDK
reachy = ReachySDK(host='localhost')

Para comprobar que se ha realizado correctamente se puede mover una articulación o una antena a una cierta posición (observar figura 6.2).

reachy.head.l_antenna.goal_position = 50.0

Figura 6.2. Movimiento de una antena de Reachy para comprobar la correcta conexión.

Una vez comprobada la buena conexión con el robot mediante el SDK de Python, se abre un abanico inmenso para programar cualquier movimiento que se quiera que realice Reachy de forma estática. De momento no se puede simular al robot en movimiento, ni desplazarnos por la escena.

Se programa un conjunto de trayectorias de varias articulaciones para construir un saludo afectivo, de esta forma se simula el recibimiento del robot a una persona. Esta programación se realiza de forma conjunta en un script, el cual está compuesto de varias partes.

Una primera en la cual se importan los módulos necesarios: ReachySDK para la conexión al robot, goto para generar trayectorias, InterpolationMode para elegir el patrón de estas trayectorias y time para monitorear la temporalidad de los movimientos.

```
from reachy_sdk import ReachySDK
from reachy_sdk.trajectory import goto
from reachy_sdk.trajectory import InterpolationMode
import time
```

Una segunda parte que está compuesta por 4 funciones:

La primera se encarga del movimiento de las antenas, se programa un bucle con dos posiciones opuestas de cada antena para producir una agitación de ambas.

```
def happy_antennas():
  for _ in range(7):
    reachy.head.l_antenna.goal_position = 30.0
    reachy.head.r_antenna.goal_position = -30.0

    time.sleep(0.2)

    reachy.head.l_antenna.goal_position = -30.0
    reachy.head.r_antenna.goal_position = 30.0

    time.sleep(0.2)

  reachy.head.l_antenna.goal_position = 0.0
  reachy.head.r_antenna.goal_position = 0.0
```

La segunda función es la responsable de la trayectoria del brazo, con movimientos en los distintos ejes de las articulaciones (yaw, pitch, roll) se consigue colocar el brazo en cierta posición y seguidamente con un balanceo similar al de las antenas se agita la muñeca para reproducir el saludo y después volver a la posición inicial.

```
def hello_left_arm():
  left_arm_hello_position = {
    reachy.l_arm.l_shoulder_pitch: 0,
    reachy.l_arm.l_shoulder_roll: 50,
    reachy.l_arm.l_arm_yaw: 70,
    reachy.l_arm.l_elbow_pitch: -120,
    reachy.l_arm.l_forearm_yaw: 80,
    reachy.l_arm.l_wrist_pitch: 0,
    reachy.l_arm.l_wrist_roll: -40,
  }

  left_arm_base_position = {
    reachy.l_arm.l_shoulder_pitch: 0,
    reachy.l_arm.l_shoulder_roll: 0,
    reachy.l_arm.l_arm_yaw: 0,
    reachy.l_arm.l_elbow_pitch: 0,
    reachy.l_arm.l_forearm_yaw: 0,
    reachy.l_arm.l_wrist_pitch: 0,
    reachy.l_arm.l_wrist_roll: 0,
  }

  goto(
    goal_positions = left_arm_hello_position,
    duration = 1.0,
    interpolation_mode = InterpolationMode.MINIMUM_JERK
  )

  for _ in range(3):
    goto(
      goal_positions = {reachy.l_arm.l_wrist_roll: 20.0},
      duration = 0.5,
      interpolation_mode = InterpolationMode.LINEAR
    )
    goto(
      goal_positions = {reachy.l_arm.l_wrist_roll: -40.0},
      duration = 0.5,
      interpolation_mode = InterpolationMode.LINEAR
    )

  time.sleep(0.2)

  goto(
    goal_positions = left_arm_base_position,
    duration = 2.0,
    interpolation_mode = InterpolationMode.LINEAR
  )
```

La tercera se centra en el movimiento de la cabeza, en lo cual se programa un reconocimiento del espacio girando la cabeza hacia ambos lados y seguidamente un ladeo de la cabeza.

```
def move_head():
  reachy.head.look_at(0.5, -0.3, -0.1, duration=1.0)
  time.sleep(0.3)
  reachy.head.look_at(0.5, 0.3, -0.1, duration=1.0)
  time.sleep(0.3)
  reachy.head.look_at(0.5, 0, -0.1, duration=1.0)
  time.sleep(0.3)

  head_tilted_position = {
    reachy.head.neck_roll: -20,
    reachy.head.neck_pitch: 0,
    reachy.head.neck_yaw:0,
  }

  goto(
    goal_positions = head_tilted_position,
    duration = 1.0,
    interpolation_mode = InterpolationMode.MINIMUM_JERK
  )
```

Una última función se encarga de colocar en orden las anteriores para realizar un saludo completo. Primero se realiza los movimientos de cabeza que termina con un ladeo, seguidamente se agitan las antenas que junto a ese ladeo expresa entusiasmo al recibir a la persona. Después, con la cabeza ladeada, pero sin agitar las antenas, se realiza el movimiento del brazo. Finalmente, se agrega una línea de comando para que la cabeza vuelvan a la posición de inicio y así Reachy quede en la misma posición que al principio para poder realizar cualquier otra acción.

```
def say_hello():
  move_head()
  happy_antennas()
  hello_left_arm()
  reachy.head.look_at(0.5, 0, 0, duration=0.5)
```

Por último, este script se completa con el main en el cual se realiza la conexión con el robot y se llama a esta última función que engloba todas las demás.

```
if __name__ == "__main__":
  reachy = ReachySDK(host='localhost')

  say_hello()
```

Una vez hecha la programación del saludo guaramos el script como .py y ponemos en ejecución el proyecto en Unity. Para probar nuestra simulación únicamente tenemos que dirigirnos en la terminal a la ruta donde hemos guardado ese script de la forma .py y escribir el siguiente comando:

python nombre.py

Se podrá observar en Unity la ejecución del programa que se ha creado y examinar el saludo amistoso que Reachy realiza como se visualiza a continuación su posición final en la figura 6.3.

Figura 6.3. Reachy realizando un saludo que ha sido programado mediante python.

6.3 Simulación introduciendo la realidad virtual.

Esta simulación es fundamental en el proyecto, se podrá observar cómo puede desplazarse el robot y la versatilidad que llega a tener en un entorno doméstico. Para ello, se intenta, mediante las gafas de realidad virtual, mimetizarse con Reachy para conseguir simular la teloperación de este.

Lo primero que se necesita es el robot, su simulador, lo cual nos facilita Pollen Robotics. (Lannuzel, 2022)

Una vez que se dispone de dicho simulador, se necesita crear la escena en la que se desarrollará el proyecto. Para ello, se crean dos habitaciones distintas, la cocina y el salón, unidas por un pasillo como podría ser el caso de un domicilio común como se puede ver en la figura 6.4.

Figura 6.4. Escena construida en Unity compuesta por una cocina y un salón.

La escena se comienza creando cubos y modificando sus dimensiones para construir los suelos y paredes. Seguidamente, se colocan los distintos muebles y objetos de cada estancia mediante un modelo fbx que se puede descargar de distintas páginas (Free3D, s.f.). A cada objeto se le debe otorgar un "collider". Este componente puede ser de distintas formas (*box, sphere, capsule,* etc) para adaptarse a la estructura física de cada objeto y así el robot no pueda atravesarlos, las paredes y el suelo vienen con este elemento incorporado. Una vez construida la escena, se tiene que crear los distintos materiales de los que estarán hechos tanto los objetos como las paredes y el suelo. Cada material se crea aplicándole la textura que se desea para producir una mayor sensación de realidad en los objetos (Poly Haven, s.f.). Por último, se dispone el robot en una de las dos estancias, la cual será su posición inicial cuando lo traslademos al mundo virtual.

Una vez compuesta la escena, se incorporará la realidad virtual al proyecto. Pero para ello hay que realizar una configuración previa.
Inicialmente se debe pasar el proyecto a Android de la siguiente manera:
File -> Build Settings -> Android -> Switch Platform
Con este proceso Unity se encarga de realizar todas las conversiones necesarias para transformar la aplicación para Android.
El siguiente paso es cargar varios paquetes dentro de *Window -> Package Manager*:

- *Oculus Integration*: este paquete se encuentra en "*Packages: MyAssets*" como se observa en la figura 6.5, lo que significa que necesita previamente ser instalado desde la web de Unity en Asset Store. Paquete que brinda soporte avanzado de renderizado, social, plataforma, audio y desarrollo de avatares para dispositivos Oculus VR y algunos dispositivos compatibles con Open VR.

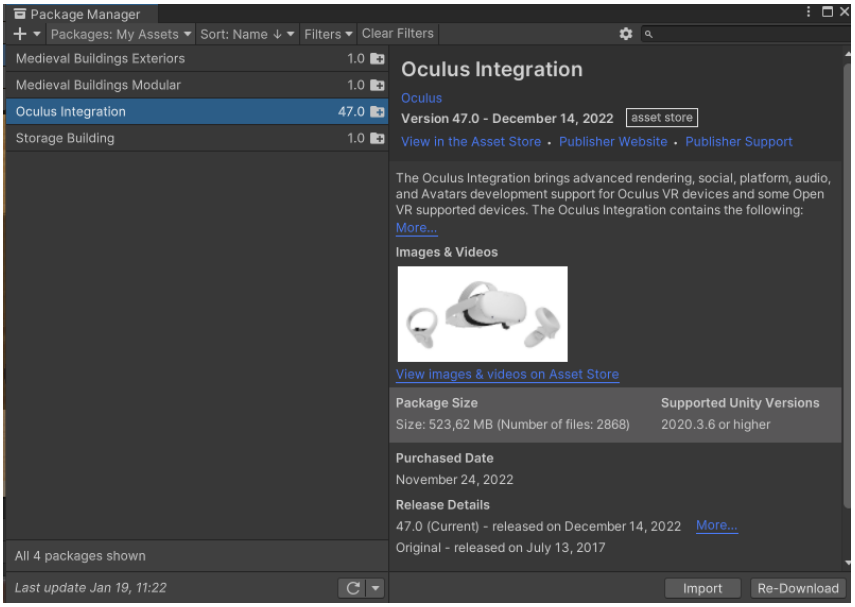

Figura 6.5. Selección del paquete Oculus Integration.

- *XR Plugin Management*: es un paquete propio de Unity por lo tanto se encuentra en "*Packages: Unity Registry*" como se observa en la figura 6.6. Paquete para proporcionar una gestión sencilla de los complementos XR. Proporciona ayuda y gestión para la carga, inicialización, configuración y soporte de compilación para complementos XR que están destinados a simplificar la forma en que la VR y la AR (realidad aumentada) funcionan a través de Unity.

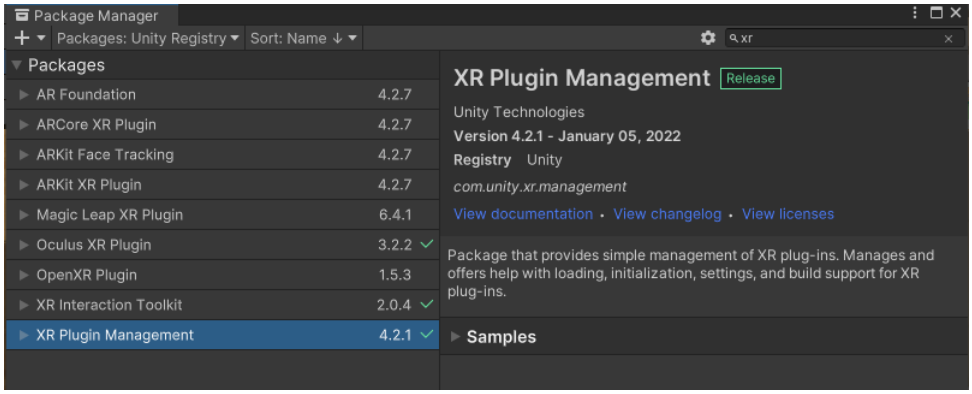

Figura 6.6. Selección del paquete XR Plugin Management.

- *Oculus XR Plugin*: es un paquete propio de Unity por lo tanto se encuentra en "*Packages: Unity Registry*" como se observa en la figura 6.7. Paquete que proporciona soporte de entrada y visualización para dispositivos Oculus.

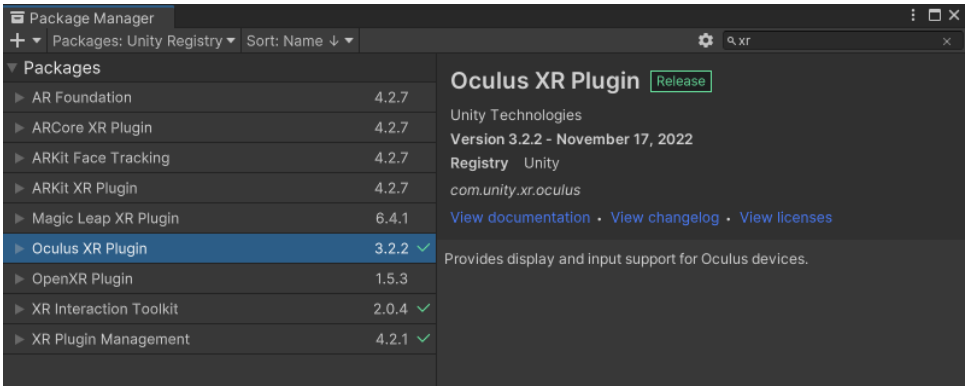

Figura 6.7. Selección del paquete Oculus XR Plugin.

Después de instalar estos paquetes, en *Edit -> Project Settings -> XR Plug-in Management*, se tiene que activar el soporte para Oculus (ver figura 6.8).

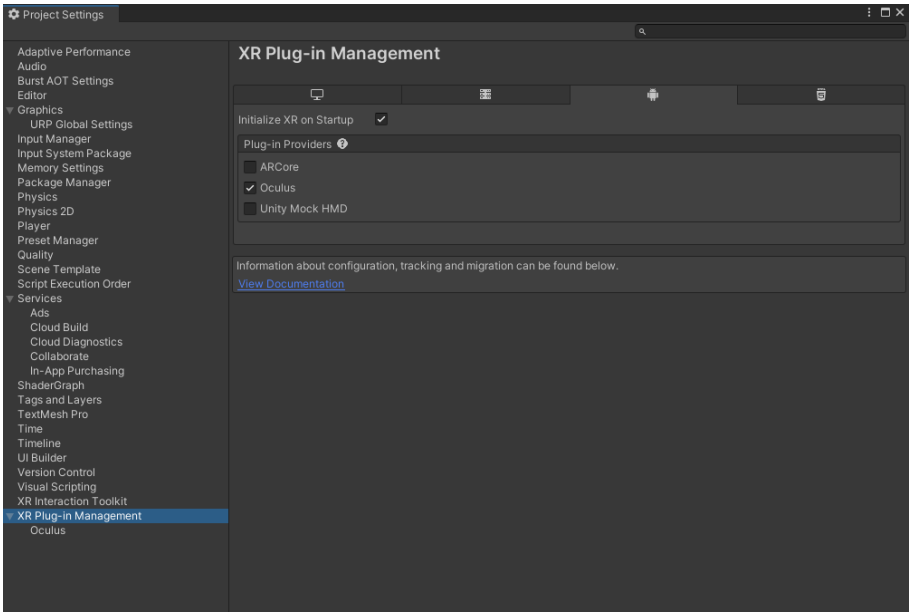

Figura 6.8. Opción Oculus activada dentro de la ventana XR Plug-in Management.

Hecha la configuración previa, se elimina la "Main Camera" de la escena para añadir una estructura que permita tener una cámara para las gafas además de hacer un *tracking* de todo lo necesario como la orientación y posición de la cabeza, de los mandos, etc. Para ello, en la ventana del proyecto, carpeta *Assets* -> Oculus -> VR -> *Prefabs -> OVRPlayerController* o buscar dicho *prefab* en el buscador.

Añadimos dicho *prefab* a la escena ya que este elemento es el que permite reproducir los movimientos con las gafas y los mandos del equipo de realidad virtual, dentro se encuentra *OVRCameraRig* y dentro de este está el *TrackingSpace* compuesto de los distintos elementos que realizan el *tracking* de los ojos y brazos tanto izquierdo como derecho y del cuerpo.

Una última configuración la realizamos en *OVRCameaRig,* seleccionando dicho elemento y yendo a la parte del inspector cambiamos la opción "*Tracking Origin Type"* a "*Floor Level"* ya que la referencia en las gafas Oculus Quest 2 es el suelo.

Para simular que el teleoperador es el robot, movemos a Reachy dentro de *OVRCameraRig* para que así Reachy entero siga los movimientos de orientación y desplazamiento que realice con las gafas virtuales.
Existe el problema de que si movemos los mandos no se refleja ese movimiento en los brazos de Reachy. Esto se soluciona en cierta medida de la siguiente manera; se despliega el *prefab* de Reachy y se observa que está compuesto por las diferentes partes del robot (tronco que es el propio *prefab*, cabeza, soporte y brazos), dentro de los brazos se va desglosando en las diferentes partes y articulaciones que lo componen. Se quiere que los mandos simulen las pinzas del robot, por lo tanto, movemos esa parte del *pefab* de Reachy *("Left/Rigth Wrist Pitch"*) dentro de *TranckingSpace* del *OVRCameraRig* al elemento *"Left/Rigth Hand Anchor"* respectivamente (examinar figura 6.9). El resto de partes del brazo se incluyen dentro de cada *Wrist Pitch* para que los movimientos que hagan las pinzas los sigan las demás partes del brazo.

Figura 6.9. Jerarquía del OVRPlayerController en Unity.

De esta forma conseguimos que el teleoperador sea el robot en primera persona. Modificando el radio de *OVRPlayerController* en su característica "*Character Controller"* que hace la función de un *collider* podemos simular la base móvil y así la distancia a la que Reachy se puede acercar a los objetos que también poseen un *collider*.

Por último, ya configurado todo el proyecto y la escena, se compila y se ejecuta la aplicación, de la siguiente manera se cargaría en las gafas. *File -> Build Settings -> Android,* en la opción "*Run Device*" debería aparecer la opción de seleccionar las Oculus Quest 2. Pero para que esto suceda hay que conectar las gafas al ordenador de una forma adecuada.

Se necesita un cable USB 3.0 a Tipo C, para realizar la conexión entre las gafas y el ordenador. Para que la conexión sea exitosa el ordenador necesita cumplir ciertos requisitos:

- Operating System: Windows 10 (o Windows 7 SP1)
- Procesador: Intel Core i5-4590/AMD FX 8350 equivalente o mejor.

- Memory: 8GB RAM
- Graphic card: NVIDIA GeForce GTX 970, AMD Radeon R9 290 equivalente o mejor.
- Network: Conexión de Internet de banda ancha. Se recomienda encarecidamente que su PC esté conectada a su enrutador mediante un cable de ethernet.

También se necesita tener instalada la aplicación de Oculus tanto en el ordenador como en un dispositivo móvil. En la aplicación del ordenador se comprobará si reconoce el equipo de realidad virtual y si la conexión por cable es exitosa. Por otra parte, para que Unity detecte las gafas Oculus Quest 2 como un sistema Android capaz de desarrollar e integrar aplicaciones, es necesario activar el modo desarrollador en la aplicación Oculus. Pero esta opción solo se puede activar desde la aplicación en un dispositivo móvil, por eso es estrictamente necesario tener la app en un smartphone.

Si todos estos requerimientos se cumplen con éxito, aparecerá el dispositivo de VR en la opción "*Run Device"* y se podrá dar al botón *"Build and Run"* que cargará el proyecto en las gafas virtuales.

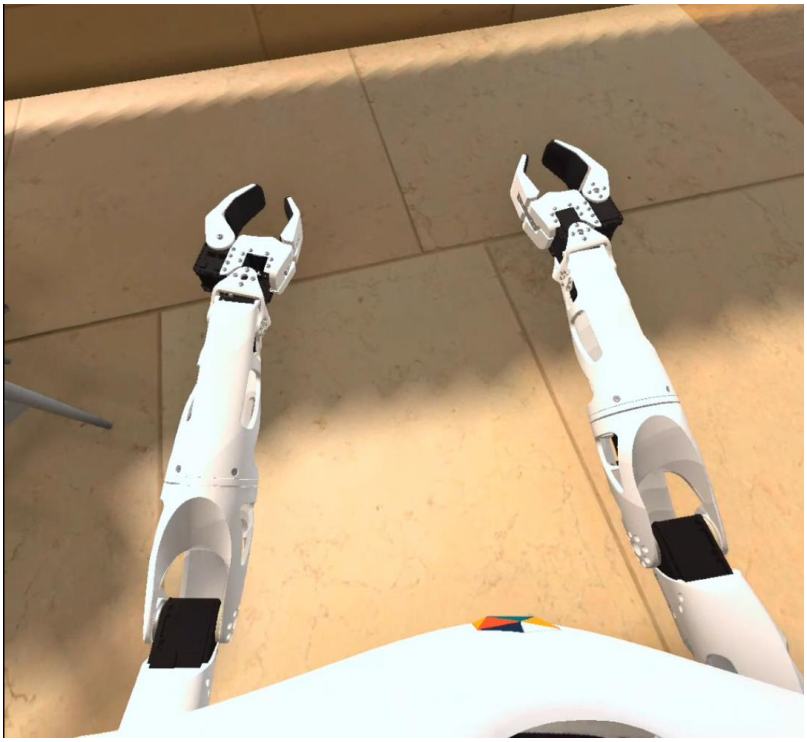

Figura 6.10. Imagen inicial de la escena con las gafas virtuales.

Como se puede percibir en la figura 6.10, la primera imagen cuando entramos en la escena son los brazos de Reachy que se mueven al igual que los mandos, si inclinamos la cabeza hacia abajo podemos llegar a ver un poco el tórax del robot, en el que se puede percibir el logo de la empresa Pollen Robotics

Es una aplicación para que el teleoperador no se mueva de un sitio fijo, puede girar sobre sí mismo y observará como el robot hace lo mismo. Al estar en el lugar del robot únicamente se pueden ver los brazos, si se quiere realizar algún desplazamiento por la escena únicamente se necesitan los *joysticks* de los mandos, el del mando izquierdo mueve el robot linealmente y el del mando derecho gira la cámara para fijar la dirección hacia la que nos queremos desplazar, juntando los movimientos de ambos joysticks se puede ir a la otra estancia o acercarnos a los objetos. En las figuras de a continuación (6.11, 6.12 y 6.13), se puede observar que se ha realizado un desplazamiento gracias a los dos joysticks para coger un plano de la cocina, el salón y el pasillo.

Figura 6.11. Imagen desde las gafas virtuales de la cocina.

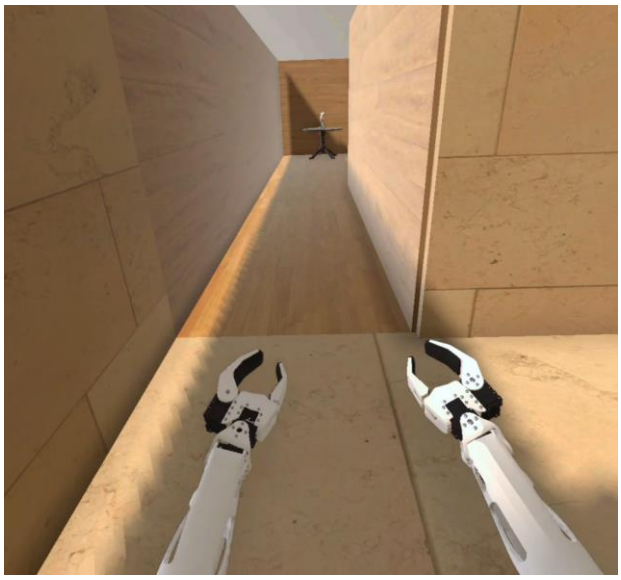

Figura 6.12. Imagen del pasillo que lleva al salón desde las gafas virtuales.

Figura 6.13. Imagen general del salón desde las gafas virtuales.

La experiencia de desplazarse por una escena y ver como los brazos de Reachy imitan los movimientos de nuestras manos es impresionante. Pero se ha querido añadir otra característica para disfrutar más de la experiencia y que no sea únicamente visual, sino que se pueda interactuar con el entorno. Se trata de la posibilidad de coger ciertos objetos, desplazarse con ellos, y poderlos colocar en otro lugar de la escena.

Para realizar este avance, primero se tiene que dotar a las pinzas de Reachy de la capacidad de agarrar objetos cuando mantengamos pulsado un botón de los mandos. Dentro de la carpeta *Assets -> Oculus -> SampleFramework -> Core -> CustomHands* hay dos *prefabs* de manos (izquierda y derecha), las cuales vienen configuradas para realizar el agarre de objetos. Estos ajustes y scripts se implementan en nuestras pinzas y así se consigue realizar el agarre (*Grab*) de objetos manteniendo pulsado el botón *Left/Rigth Grip* (observar figura 6.14) dependiendo la pinza con la que se quiera coger el objeto. Lo principal que permite que se produzca esta acción es un script llamado "*OVR Grabber*" que como se puede observar en su propio nombre es el encargado de realizar la programación del agarrador.

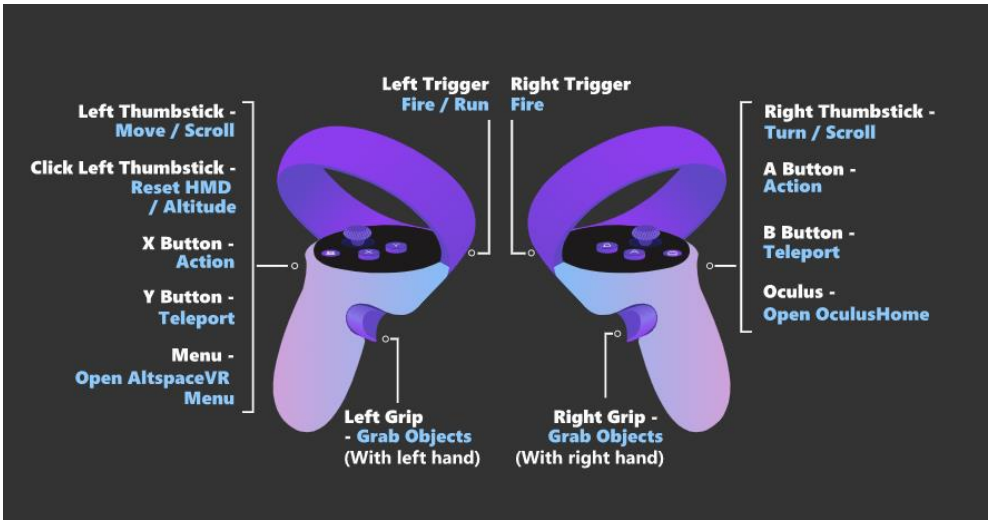

Figura 6.14. Indicación de las acciones que realiza cada botón de los mandos ente los cuales se encuentra el botón que realiza la acción de agarrar.

De esta manera, se puede simular como Reachy podría ayudar a una persona con limitaciones, la cual no pudiese alcanzar a los armarios para coger unos platos, medicina, o cualquier objeto que entre dentro de las capacidades del robot como se observa en la siguiente figura 6.15 con un ejemplo simple.

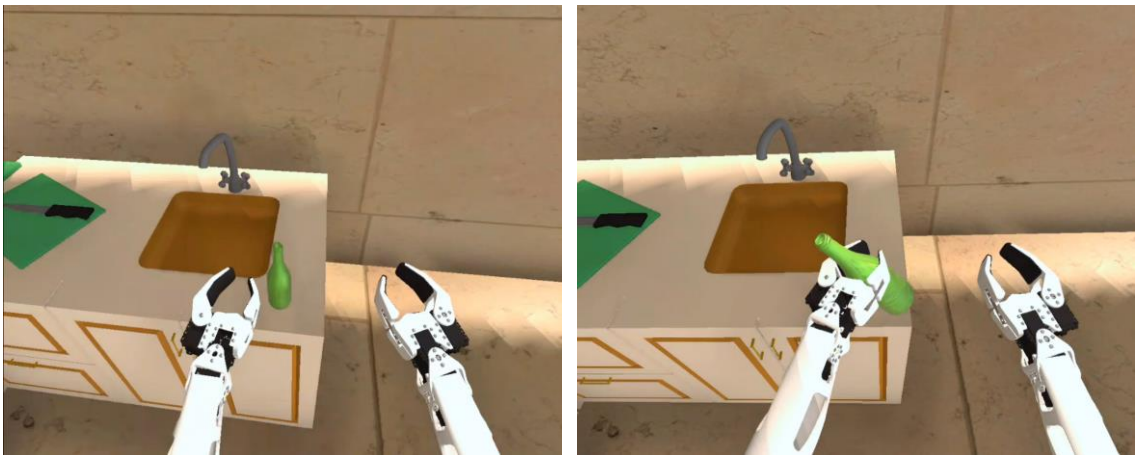

Figura 6.15. Imagen antes y después de coger una botella de la encimera de la cocina.

Para que sea posible este avance en el simulador, los objetos que vayan a ser agarrados tienen que incorporar ciertas características. Deben tener un *collider*, ya que, sino atravesarían los demás objetos, por ejemplo, si pusiéramos una botella encima de una mesa, teniendo la mesa *collider* pero la botella no, cuando se ejecutase la aplicación esta última caería al suelo atravesando la mesa.

De esta manera se puede colocar los objetos agarrados en cualquier lugar como se observa en la figura 6.16.

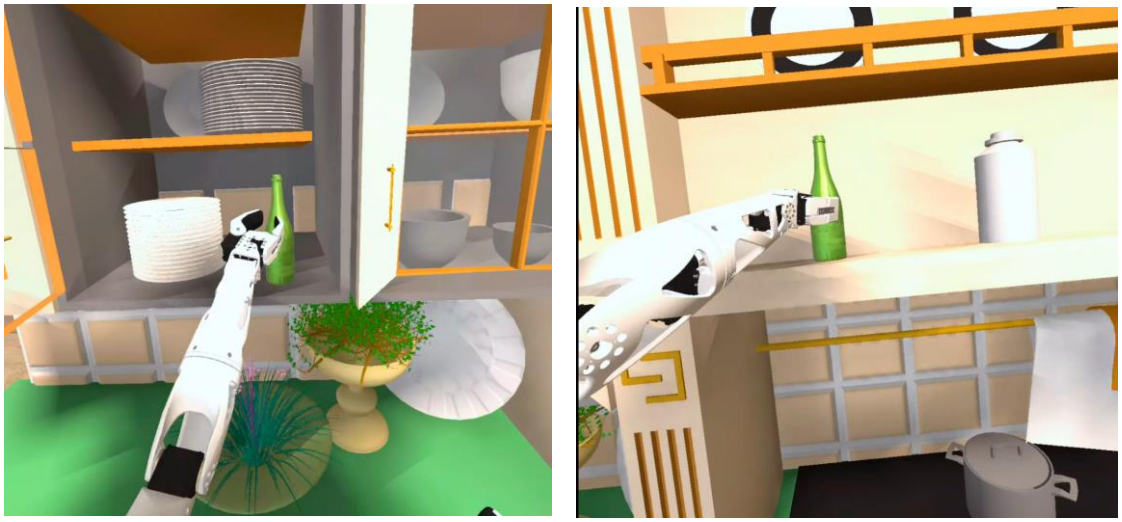

Figura 6.16. Reachy teleoperado colocando una botella en estantes a una cierta altura.

Una vez tiene el objeto un *collider* adecuado a su forma física, lo siguiente es añadir un *rigybody,* sirve para dar un valor de masa al objeto y proporcionarle gravedad para que así una vez tenga Reachy cogido el objeto, si se deja de presionar el botón del mando, este objeto caiga al suelo como lo haría en la realidad. O simplemente si existe una colisión entre 2 objetos como se observa en la figura 6.17, al tener gravedad se caerá dándole mayor realismo a la secuencia y a la escena.

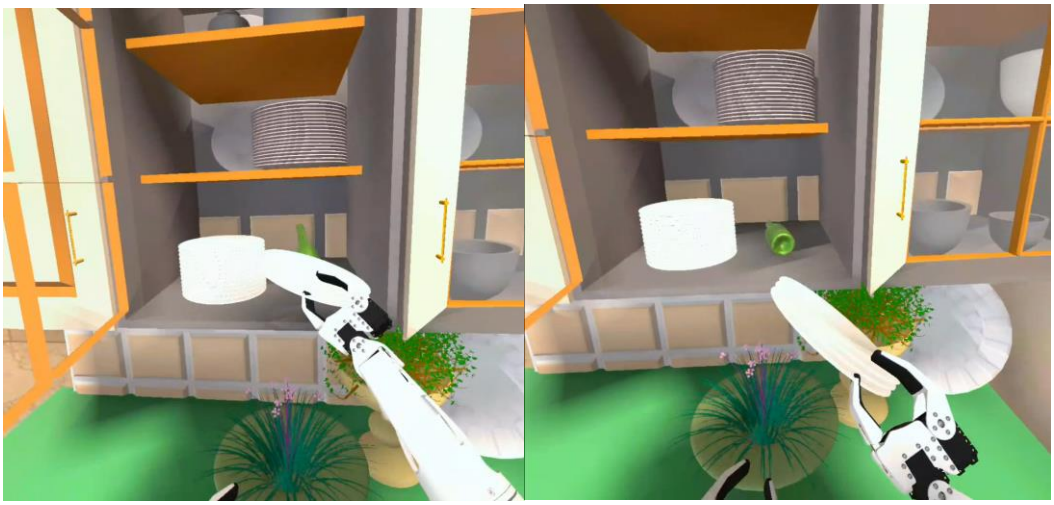

Figura 6.17. Reachy teleoperado coge unos platos y al realizar esta acción golpea la botella de al lado produciendo su caída.

La última característica, la más importante para que se produzcan los agarres que se han visto anteriormente, es añadir a cada objeto que se desee agarrar un script llamado "*OVR Grabbable*" que como el propio nombre indica es para que el objeto sea agarrable. Este script al igual que el "*OVR Grabber*" y más funcionalidades que hemos ido utilizando para mejorar el proyecto, son facilitados por el paquete "Oculus Integration" que hemos añadido con anterioridad.

6.4 Experimentos de control remoto

Una vez se observan las herramientas de las que se dispone para realizar las distintas simulaciones, se piensa en mezclar ambas. Juntar las simulaciones vistas en el capítulo 5 en Rviz con la aplicación realizada en Unity.

De esta manera lo que se quiere llegar a conseguir es representar en el visualizador RViz los movimientos realizados en la aplicación de Unity con el equipo de realidad virtual. Para ello se intentará la conexión entre Unity -> ROS -> MoveIt (RViz).

6.4.1 Conceptos de MoveIt

Después de la pequeña introducción realizada en el capítulo 5 y antes de realizar la conexión, se debe tener ciertos conocimientos sobre el funcionamiento del marco de planificación de movimiento MoveIt. (MoveIt, s.f.)

6.4.1.1 Arquitectura del sistema

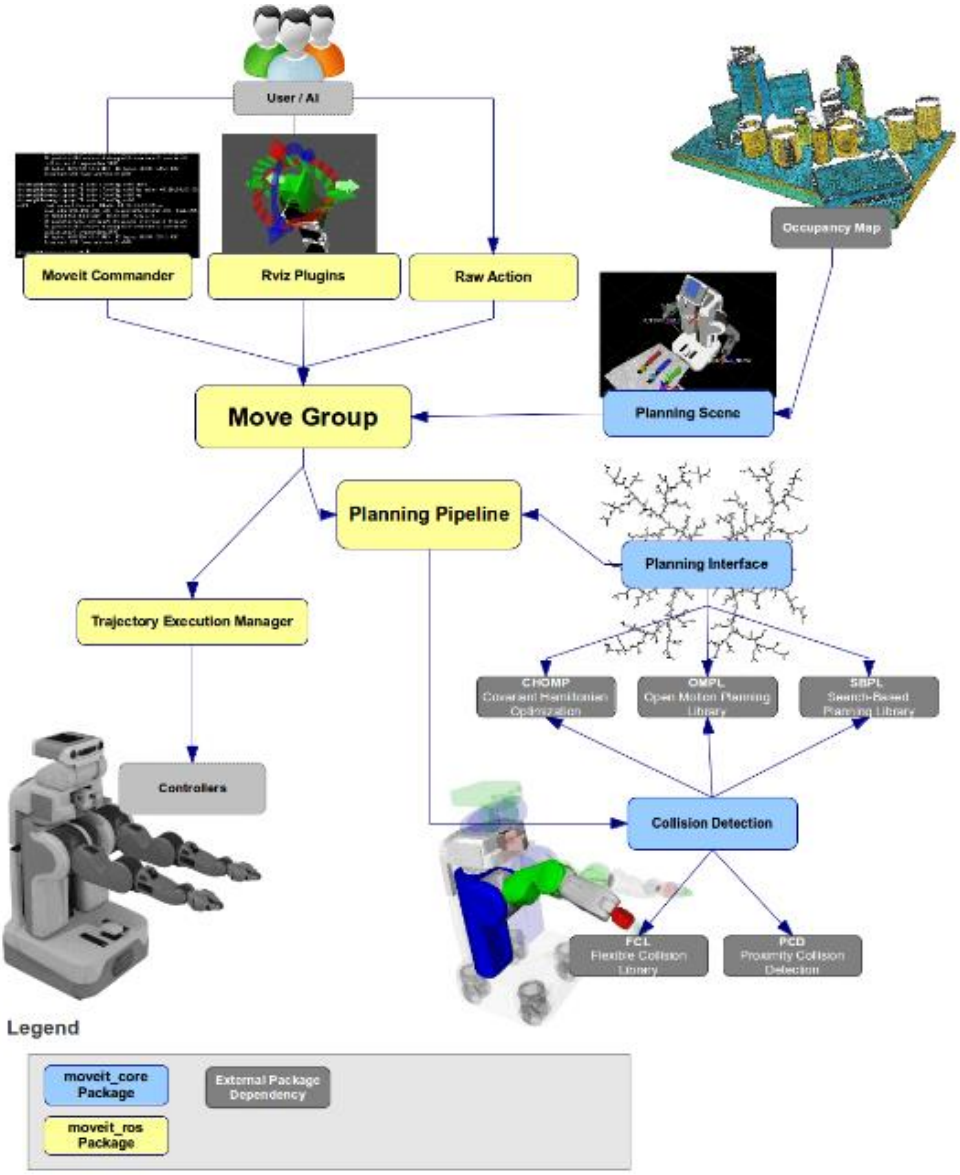

Figura 6.18. Diagrama rápido de alto nivel [17].

En la figura 6.18 se muestra la arquitectura del sistema de alto nivel para el nodo principal proporcionado por MoveIt llamado move_group. Este nodo sirve como integrador: reúne todos los componentes individuales para proporcionar un conjunto de acciones y servicios de ROS para que los utilicen los usuarios (ver figura 6.19).

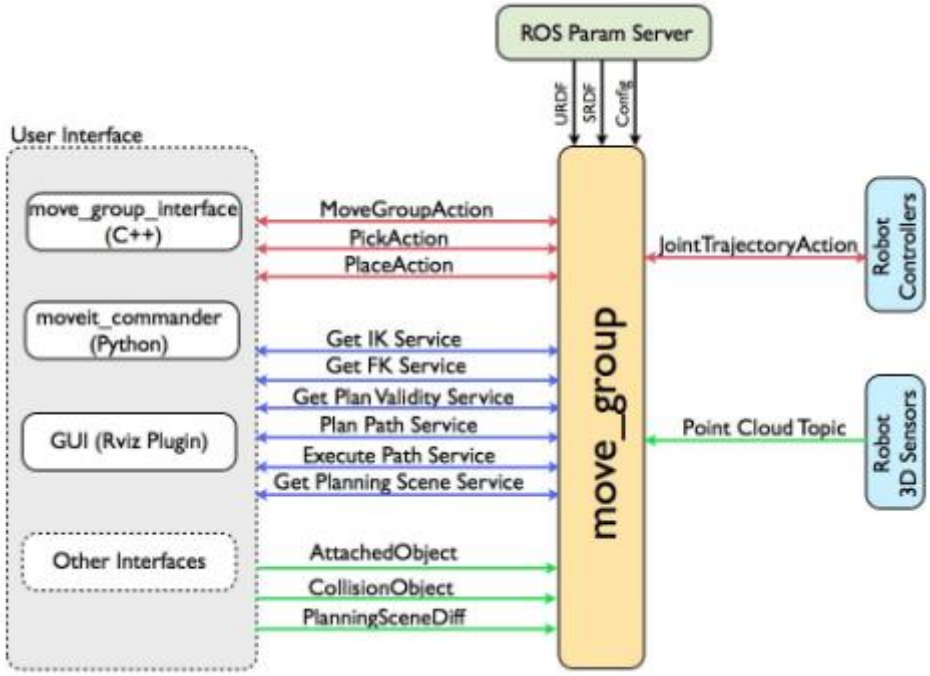

Figura 6.19. Organización entorno al nodo move_group [17].

Los usuarios pueden acceder a las acciones y servicios proporcionados por move_group de tres formas:

- En C++, utilizando el paquete move_group_interface que proporciona una interfaz de C++ fácil de configurar para move_group.
- En Python, usando el paquete moveit_commander.
- A través de una GUI: usando el complemento Motion Planning para Rviz (el visualizador ROS).

move_group se puede configurar usando el servidor de parámetros ROS desde donde también obtendrá el URDF y el SRDF para el robot.

move_group es un nodo ROS. Utiliza el servidor de parámetros ROS para obtener tres tipos de información:

- URDF: move_group busca el parámetro robot_description en el servidor de parámetros de ROS para obtener el URDF del robot.
- SRDF: move_group busca el parámetro robot_description_semantic en el servidor de parámetros de ROS para obtener el SRDF para el robot. El SRDF normalmente lo crea (una vez) un usuario mediante el asistente de configuración de MoveIt.
- Configuración de MoveIt: move_group buscará en el servidor de parámetros de ROS otras configuraciones específicas de MoveIt, incluidos límites de juntas, cinemática, planificación de movimiento, percepción y otra información. Los archivos de configuración para estos componentes son generados automáticamente por el asistente de configuración de MoveIt y almacenados en el directorio de configuración del paquete de configuración de MoveIt correspondiente para el robot.

move_group habla con el robot a través de temas y acciones de ROS. Se comunica con el robot para obtener información del estado actual (posiciones de las articulaciones, etc.), obtener nubes de puntos y otros datos de sensores de los sensores del robot y hablar con los controladores del robot.

6.4.1.2 Escena de planificación

La escena de planificación se utiliza para representar el mundo que rodea al robot y también almacena el estado del propio robot como se observa en la figura 6.20. Es sujeto por el Planning Scene Monitor dentro del nodo move_group. El Planning Scene Monitor escucha:

- Información de estado: sobre el tema joint_states.
- Información del sensor: uso del monitor de geometría mundial que se describe a continuación.
- Información de geometría mundial: de la entrada del usuario en el tema planning_scene (como una diferencia de escena de planificación).

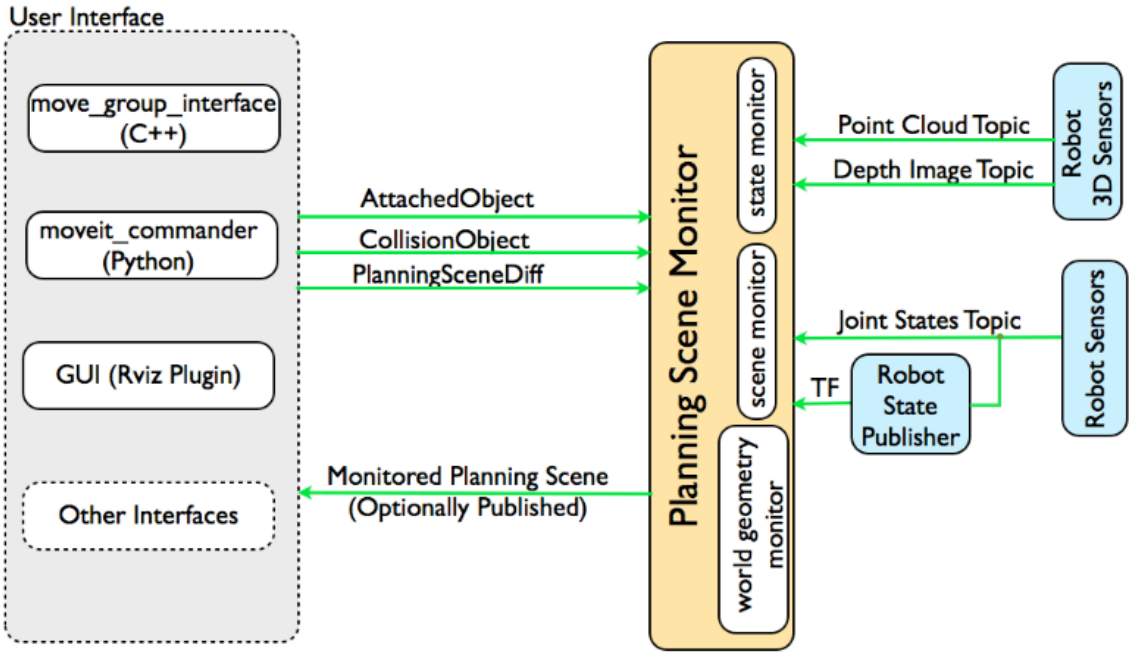

Figura 6.20. Escena de planificación basada en el Planning Scene Monitor [17].

6.4.1.3 El complemento de cinemática

Si el objetivo principal de la conexión que se quiere realizar es el movimiento de articulaciones, la cinemática cobra gran importancia.

MoveIt utiliza una infraestructura de complementos, especialmente dirigida a permitir que los usuarios escriban sus propios algoritmos de cinemática inversa. La

cinemática directa y la búsqueda de jacobianos están integradas dentro de la propia clase RobotState que se puede ver en el apartado anterior en la escena de planificación. El complemento de cinemática inversa predeterminado para MoveIt se configura mediante el solucionador numérico basado en jacobiano de KDL. Este complemento se configura automáticamente mediante el Asistente de configuración de MoveIt.

6.4.2 Configuración Unity

Para que la conexión de Unity a ROS sea posible, se tienen que realizar unas configuraciones previas en Unity. (Amanda, 2021)

Lo primero que se debe realizar, es agregar un nuevo paquete que permitirá a transmisión de mensajes entre Unity y ROS. Para ello, se copia la siguiente url en Window -> Package Manger -> Add package from git URL (ver figura 6.21)

https://github.com/Unity-Technologies/ROS-TCP-Connector.git?path=/com.unity.robotics.ros-tcp-connector#v0.7.0

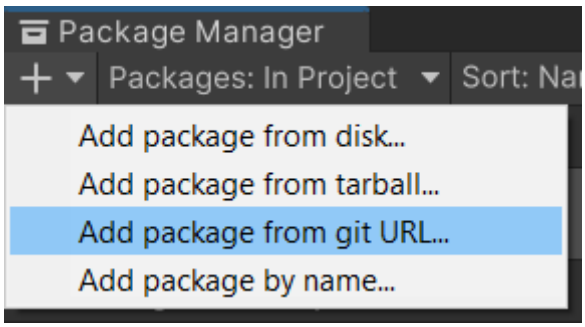

Figura 6.21. Ruta para añadir un nuevo paquete.

Una vez añadido este paquete, nos aparece una nueva ventana en Unity llamada Robotics, dentro de esta en ROS Settings tenemos que establecer la ROS IP Address como se muestra a continuación en la figura 6.22.

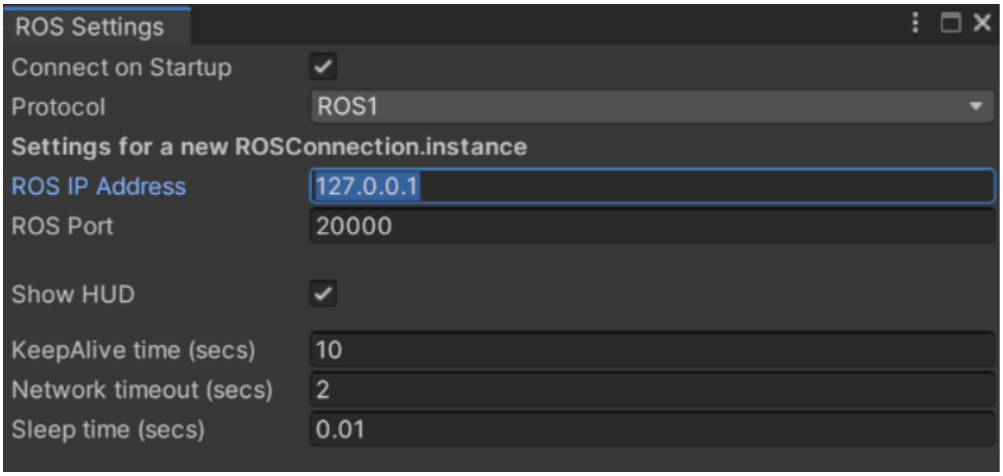

Figura 6.22. Ventana de ROS Setting en Unity.

Para saber la IP de ROS, en el espacio de trabajo en Ubuntu se ejecutan los siguientes comandos:

```
source devel/setup.bash
roslaunch ros_tcp_endpoint endpoint.launch
```

Seguidas de estas configuraciones, se crea en la escena de Unity un nuevo objeto vacío, al cual llamaremos “Publisher”. A este objeto se le añade un script que será el encargado de trasmitir las acciones que se quieran realizar. En este script debido a la programación realizada se tiene que seleccionar el topic que queremos transmitir y el objeto de la escena de Unity del cual se quiere transmitir el movimiento. En este caso como se puede observar en la figura 6.23 se ha seleccionado el controlador izquierdo del equipo de realidad virtual.

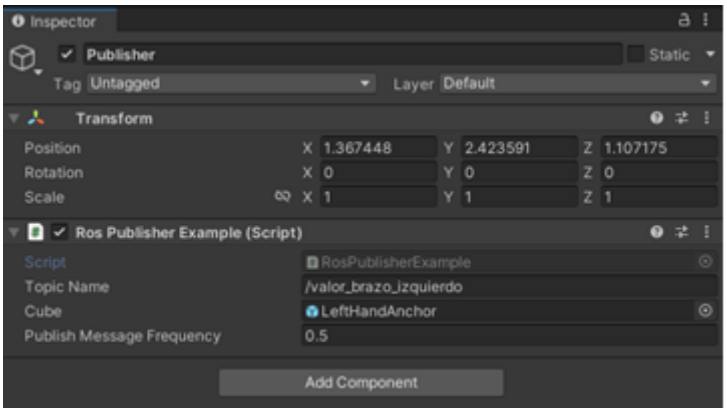

Figura 6.23. Objeto Publisher compuesto por un script

6.4.3 Conexión Unity -> ROS -> RViz

Ya realizadas las configuraciones previas en ambos lados de la conexión. Se realizará una primera aproximación a la realización de dicho control remoto.

Por el lado de Unity, con las configuraciones realizadas se ha introducido la IP de ROS dónde se tienen que enviar los mensajes deseados (observar figura 6.24).

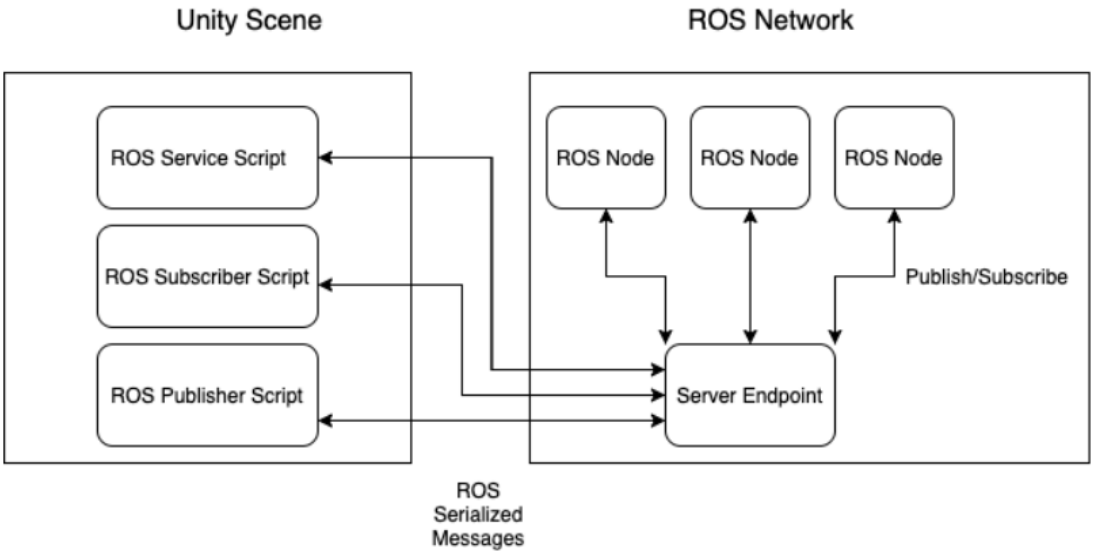

Figura 6.24. Esquema de la conexión entre Unity y ROS.

Estos mensajes enviados desde Unity serán descritos en un script que, como ya se ha visto, está dentro del objeto Publisher y debe ser programado mediante C# ya que es el lenguaje utilizado dentro de Unity.

```
using UnityEngine;
using Unity.Robotics.ROSTCPConnector;
using RosMessageTypes.Std;

<summary>
</summary>

public class RosPublisherExample : MonoBehaviour
{
   ROSConnection ros;
   public string topicName = "pos_rot";

   // The game object
   public GameObject cube;
   // Publish the cube's position and rotation every N seconds
   public float publishMessageFrequency = 1f;
   // Used to determine how much time has elapsed since the last message was published
   private float timeElapsed;

   void Start()
   {
      // start the ROS connection
      ros = ROSConnection.GetOrCreateInstance();
      ros.RegisterPublisher<StringMsg>(topicName);
```

```
    }

    private void Update()
    {
        timeElapsed += Time.deltaTime;

        if (timeElapsed > publishMessageFrequency)
        {
            var prueba = new StringMsg();
            prueba.data = cube.transform.position.x.ToString();

            // Finally send the message to server_endpoint.py running in ROS
            ros.Publish(topicName, prueba);

            timeElapsed = 0;
        }
    }
}
```

En este script, se puede observar como en la función Start() se realiza la conexión con ROS y el topic que se va a transmitir. En la función Update() se encuentra el dato a transmitir y seguidamente el envío del mensaje al nodo server_endpoint que se encuentra ejecutándose en ROS.

Por otra parte, en el entorno ROS se tiene que realizar ciertas acciones con el mensaje recibido desde Unity.

En primer lugar, se tiene que iniciar el nodo server_endpoint que es el que recogerá en primera estancia los mensajes que vienen desde Unity. Esto se realiza lanzando el siguiente archivo .launch mediante roslaunch y la ruta donde se encuentre dicho archivo. Además, la ejecución roslaunch inicia automáticamente ROS Core si aún no se está ejecutando.

```
<launch>
        <arg name="tcp_ip" default="0.0.0.0"/>
        <arg name="tcp_port" default="10000"/>

        <node name="server_endpoint" pkg="ros_tcp_endpoint"
        type="default_server_endpoint.py" args="--wait" output="screen" respawn="true">
                <param name="tcp_ip" type="string" value="$(arg tcp_ip)"/>
                <param name="tcp_port" type="int" value="$(arg tcp_port)"/>
        </node>
        <node name="trajectory_subscriber" pkg="niryo_moveit"
        type="trajectory_subscriber.py" args="--wait" output="screen"/>
</launch>
```

En segunda instancia, como hemos podido ver en el apartado de la arquitectura de MoveIt se puede acceder a las acciones y servicios proporcionados por move_group

de varias formas, pero en el caso del presente proyecto se ha decidido que será mediante Python usando el paquete moveit_commander.

Por lo tanto, mediane Python se crea un nodo en ROS llamado “control_reachy” y dentro de este un topic con el nombre “valor_brazo_izquierdo”. Este topic transforma el mensaje enviado desde Unity en una planificación de movimiento de MoveIt. A su vez calcula la cinemática inversa y va enviando las posiciones angulares mediante mensajes que muestra el robot en Rviz debido a que este robot está subscrito como hemos podido ver en la escena de planificación.

Todo esto mencionado anteriormente se recoge en un script .py que se muestra a continuación.

```
import rospy
from std_msgs.msg import Float32
from std_msgs.msg import String
import sys
import copy
import moveit_commander
import moveit_msgs.msg
import geometry_msgs.msg

# Inicializamos el MoveIt
moveit_commander.roscpp_initialize(sys.argv)
# Inicializamos el nodo
rospy.init_node('control_reachy', anonymous = True)
robot = moveit_commander.RobotCommander()
scene = moveit_commander.PlanningSceneInterface()
group = moveit_commander.MoveGroupCommander("left_arm")
display_trajectory_publisher = rospy.Publisher('/move_group/display_planned_path',
moveit_msgs.msg.DisplayTrajectory)

def callback(data):
  rospy.loginfo(data)
  rospy.loginfo(rospy.get_caller_id() + " Recibido valor " + data.data)
  rospy.loginfo("Conexión establecida")

  group_variable_values = group.get_current_joint_values()

  group_variable_values[0] = float(data.data.replace(",","."))
  group.set_joint_value_target(group_variable_values)

  plan2 = group.plan()
  group.go(wait=True)

  rospy.sleep(5)

sub = rospy.Subscriber('/valor_brazo_izquierdo', String, callback)
```

rospy.loginfo("Conexión establecida")

rospy.spin()

Por último y como recopilación, para ejecutar dicho control remoto se debe seguir los siguientes pasos.

En Ubuntu.

1. En una terminal se lanza el MoveIt para poder visualizar el RViz.
2. En otro terminal se ejecuta el archivo .launch con la orden roslaunch.
3. En otro terminal se ejecuta el nodo control_reachy.py mediante la orden rosrun.

En Windows.

4. Se compila y ejecuta la aplicación de Unity en las gafas virtuales mediante File -> Build And Run.

De esta manera veremos en las gafas virtuales la aplicación mientras que en RViz se podrá observar algún movimiento en el robot debido a la llegada de mensajes, los cuales se podrán ver en las terminales.

Capítulo 7:
Resultados del proyecto

Antes de finalizar el Trabajo Fin de Grado, y una vez ejecutado el desarrollo de este, se comentará brevemente las decisiones y su desempeño a lo largo del proyecto.

Como se determinó en el capítulo inicial del proyecto, para conseguir el objetivo principal, la teleoperación de un robot humanoide para la ayuda de personas dependientes en un entorno doméstico, es necesario con anterioridad completar distintas tareas.

7.1 Elección del equipo de trabajo

Antes de empezar a desarrollar y ver en detalle las pruebas realizadas y sus resultados, el primer paso era decidir el robot que se usaría para dicha teleoperación y con qué equipo se realizaría esta acción.

Esta decisión es fundamental en el desarrollo del trabajo, debido a que cada robot tiene distintas funciones a destacar.

Se observó que hay diferentes variantes para la integración de robots en el ámbito de ayuda social. Pero el proyecto necesita principalmente 2 requerimientos:

- El robot debe tener articulaciones capaces de manipular objetos como si de un humano se tratase.
- El robot debe poder ser teleoperado mediante realidad virtual.

Con estas exigencias el abanico se acota enormemente, también siempre teniendo en cuenta como en todo proyecto de ingeniería la importancia del coste.

En este caso, cobra gran importancia otro aspecto fundamental en la integración social de robots, su aspecto. Por lo tanto, sería de gran ayuda un robot que no transmitiese temor o rechazo hacia él, sino todo lo contrario, ternura y amabilidad. Por lo tanto, uniendo esta característica y muchas más como se ha recalcado en el capítulo 3, se eligió a Reachy. El cual cumple a la perfección con los dos principales requerimientos y además su aspecto físico más endeble que robusto, y su capacidad de expresar sentimientos mediante sus antenas, cabeza y extremidades le dota de una emoción afectiva adaptable a interactuar con personas de manera fácil y eficaz.

La elección del equipo de realidad virtual fue más sencilla debido a que Oculus VR es de las empresas con más desarrollo actualmente en la realidad virtual además de la extensa implementación de sus dispositivos entre la comunidad. Dentro de estos, Oculus Quest 2 son sus últimas gafas de realidad virtual y por lo tanto las que más cualidades y facilidades poseen. Además, el precio de estas no es diferencial con respecto a los otros modelos teniendo en cuenta las ventajas que incluye.

7.2 Comprobación de las funcionalidades de Reachy

Una vez decididas las herramientas de trabajo, hay que explorar las capacidades de Reachy para alcanzar el objetivo principal. Para ello, las simulaciones es la mejor manera para abaratar costes y comprobar todo lo necesario.

7.2.1 Funcionalidad de las articulaciones

Se ha remarcado las ventajas que pueden ofrecer sus articulaciones en el capítulo 5 con la ayuda de MoveIt. Hemos conseguido observar todos los posibles movimientos y rotaciones que permiten realizar los brazos de Reachy y así llegar a la mayoría de posiciones a los que podrían llegar unos brazos humanos. Esto en gran parte es debido a la enorme similitud que comparten los brazos del robot con un brazo humano, como podemos observar a continuación en la figura 7.1.

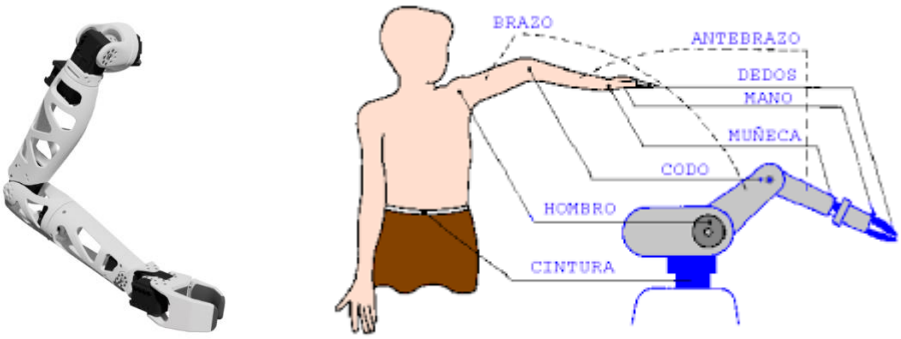

Figura 7.1. Comparación del brazo de Reachy con un brazo humano

La similitud, al igual que con un brazo robótico industrial, se basa en las mismas articulaciones (hombro, codo y muñeca) además de la importancia de la mano o efector, en este caso de Reachy, no dispone de una mano como tal, sino de una pinza que es suficiente para realizar las tareas que se le requerirán. En especial este robot destaca en la perfecta manipulación de objetos.

No solo hemos visto las ventajas que proporcionan estos brazos sino también sus limitaciones a la hora de llegar a ciertas posiciones, como podría ser detrás de la espalda. Pero realmente esto se resuelve fácilmente moviendo el robot o girándole debido a su plataforma móvil que permite su desplazamiento y rotación.

Gracia a estas simulaciones también se puede comprobar el buen diseño del cuello y sobre todo de su mecanismo Orbita. Permite colocar la cabeza en cualquier posición como si de una cabeza humana se tratase, dando un rango de visión tremendamente amplio.

7.2.2 Expresividad del robot

En los proyectos sociales de robótica se enfoca mucho la atención en la expresividad que puede mostrar dicho robot debido a que va a estar en continua relación con los humanos y muchas veces con personas que no son muy cercanas a esta tecnología como las personas de avanzada edad.

Esta cualidad importante de Reachy ha sido comprobada en el capítulo 6 con la plataforma Unity. Reachy de momento no es capaz de emitir sonidos, por lo tanto, toda la relación con las personas tiene que ser meramente expresiva gracias a sus movimientos corporales. Esta es una faceta que se ha sabido desarrollar con astucia y relacionándolo con la afectividad que son capaces de mostrar los animales con notros con gestos simples como mover la cola o girar la cabeza hacia un lado.

Con una simulación mediante programación, se muestra la gran expresividad que Reachy es capaz de mostrar. Las antenas, cuello y cabeza cobran un importante papel a la hora de transmitir sentimientos. En esta simulación, Reachy realiza un simple saludo, en él cabe destacar como consigue mostrar felicidad, al ver a la persona con la que interactúa, agitando sus antenas como lo mostraría un perro al agitar su cola.

7.3 Implementación de la realidad virtual

Ya comprobadas sus cualidades por separado, se debe avanzar en el camino al objetivo principal, y esto se consigue introduciendo la realidad virtual en el control del robot, una vez que ya ha sido estudiada esta tecnología en el capítulo 4.

Antes de incorporar la realidad virtual, se debe desarrollar una aplicación compatible con esta tecnología. Gracias a la plataforma de desarrollo de videojuegos Unity, se crea un entorno que simula el espacio de trabajo para el cual está destinado Reachy, un entorno doméstico simple (cocina + salón). En este entorno se programan distintos objetos para dotarles de una función que les permita ser manipulados por el robot y así comprobar el buen manejo del robot en esta faceta.

Una vez dispuesto todo, se consigue, como se muestra en el capítulo 6, la teleoperación de Reachy mediante las Oculus Quest 2 en una aplicación que simula el comportamiento del robot en un entorno doméstico. Con un gran desarrollo se ha permitido que se pueda realizar esta acción como si la persona encargada de teleoperar a Reachy sea el propio robot, siendo sus manos las del robot. Permite, no solo el movimiento de Reachy por un hogar, como será en la realidad, sino que también sus articulaciones puedan manipular objetos como marcaba uno de los principales requisitos.

De esta manera, mediante simulaciones y la implementación de la realidad virtual en ellas, se consigue llegar al objetivo general de proyecto, la teleoperación de un robot humanoide en un entorno doméstico mediante realidad virtual.

7.4 Conectividad Unity -> ROS -> RViz

Con la aplicación de teleoperación creada y controlada mediante realidad virtual surgió un nuevo avance, mostrar ese control remoto en el visualizador de ROS llamado RViz. Mediante este control podríamos ver en RViz los movimientos de las articulaciones del robot que se producen a medida que la persona que controla el robot dentro de la aplicación realiza ciertas acciones.

Se consigue realizar de forma satisfactoria la conexión entre Unity y ROS. Además, mediante un nodo y y topic creados se consigue transformar esos mensajes recibidos desde Unity en una planificación de movimiento de MoveIt lo que permitirá que en RViz se realice una orden de movimiento de alguna articulación del robot. De esta manera, un movimiento realizado con el equipo de realidad virtual dentro de la aplicación de teleoperación creada con Unity, se podrá observar en RViz cómo se produce cierta alteración en alguna articulación.

A partir de esta primera aproximación, se podrá ir desarrollando un control remoto más complejo para conseguir visualizar distintas trayectorias.

Capítulo 8:
Estudio económico

En este capítulo se van a detallar los diferentes costes de todo el proyecto, para poder determinar la viabilidad del mismo y si realmente es ventajoso económicamente respecto a otros trabajos que podrían desarrollarse o se están implementando en la actualidad.

Para ello, se estudiarán los diferentes costes del proyecto, tanto directos como indirectos, incluyendo mano de obra, sistemas hardware y software empleados, y otros costes que suelen ser olvidados, pero son necesarios.

8.1 Recursos empleados.

En todo proyecto de ingeniería hay una serie de recursos tanto físicos como de software que se utilizan a lo largo de su desarrollo. Aquí se incluirán tanto los recursos utilizados en las simulaciones realizadas como los utilizados en una puesta en escena real, donde entra el mayor impacto económico del proyecto, el robot Reachy.

Una gran ventaja en muchos de los últimos desarrollos tecnológicos es la posibilidad de utilizar hardware y software abierto (*Open Source*). Este tipo de sistemas reducen enormemente los costes en el desarrollo del proyecto.

Se muestran a continuación los elementos y materiales empleados en el proyecto:

- Software:
  - ✓ Sistemas operativos: Microsoft Windows 10 Pro, Ubuntu 20.04 LTS y ROS (Robotic Operating System).

  - ✓ Unity
  - ✓ MoveIt
- Hardware:
  - ✓ Ordenador con las siguientes características mostradas en la figura 8.1 debido a los requerimientos necesarios para la compatibilidad con la realidad virtual

    - Placa base H510
    - Intel Procesador ***Intel Core i7-11700F (11a generación***
    - Memoria *RAM 16Gb* DDR-4 3200
    - 1 x Disco SSD 256Gb. M.2
    - Tarjeta gráfica Gigabyte RTX 2060 D6 6G, GeForce RTX 2060, 6 GB, GDDR6, 192 bit, 7680 x 4320 Pixeles, PCI Express x16 3.0 - 3 x DisplayPort
    - Chasis Coolbox M550 2 puertos USB 3.0 delanteros - soporte para candado - F.A. 800W - 80+ Bronce
    - Ventilación lateral - Conectores frontales USB y Audio
    - Salidas video de la placa: HDMI – DVI – DP
    - Red Gigabit integrada 10/100/1000
    - USB 2 x USB 3.0 + Audio
    - Teclado y ratón Logitech con cable
    - Sistema Operativo Windows 10-64 bits

    Figura 8.1. Características del ordenador utilizado.

  - ✓ Gafas virtuales Oculus Quest 2.
- Material ofimático:
  - ✓ Material de papelería.
  - ✓ Microsoft Office 2021.
  - ✓ Otros consumibles.

8.2 Costes directos.

Entre los costes directos se incluyen aquellos imputables directamente al desarrollo del presente proyecto, como puede ser los siguientes:

- Mano de obra
- Amortización de programas y equipos
- Materiales directamente empleados

8.2.1 Coste mano de obra.

El proyecto desarrollado ha sido llevado a cabo por un ingeniero, que ha sido encargado de las investigaciones previas y las posibilidades para interactuar con el robot. Además, se ha encargado del diseño e implementación de la realidad virtual en las simulaciones del robot.

Para contabilizar estos costes, hay que tener en cuenta el salario, tanto bruto anual como las cotizaciones a la Seguridad Social (35% del sueldo bruto), lo cual se muestra a continuación en la tabla 1.

COSTE ANUAL	
Sueldo bruto más incentivos	27.000 €
Seguridad Social (35% sueldo bruto)	9.450 €
Coste total	**36.450** €

Tabla 1 Salario anual ingeniero

También hay que considerar los días que este salario representa, se puede observar en la tabla 2.

DÍAS EFECTIVOS POR AÑO	
Año medio	365,25 días
Sábados y Domingos	-104,36 días
Días de vacaciones efectivos	-20,00 días
Días festivos reconocidos	-15,00 días
Días perdidos estimados	-5,00 días
Total días efectivos estimados	**220,89 días**

Tabla 2 Días de trabajo

Con el número de días de trabajo, y teniendo en cuenta una jornada laboral de 8 horas, el cálculo total de horas efectivas de trabajo en un año es:

$$220{,}89\ \frac{días}{año}\ x\ 8\frac{horas}{día} = 1.767{,}12\ \frac{horas}{año}$$

Con el sueldo anual anteriormente calculado y las horas efectivas de trabajo, se puede obtener el coste por hora de un ingeniero:

$$\frac{36.450\frac{€}{año}}{1.767{,}12\ \frac{horas}{año}} = 20{,}63\ \frac{€}{hora}$$

En la siguiente tabla (tabla 3) se muestra una distribución temporal aproximada del trabajo realizado por el ingeniero en el presente proyecto.

DISTRIBUCIÓN TEMPORAL DEL TRABAJO	
Formación y documentación	50 horas
Estudio del problema	100 horas
Desarrollo de la aplicación	150 horas
Elaboración de la documentación	200 horas
Total horas empleadas	**500 horas**

Tabla 3 Distribución temporal del trabajo

El coste de la mano de obra directo es la multiplicación de las horas y el coste efectivo de una hora de trabajo del ingeniero:

$$500\ horas\ x\ 20{,}63\ \frac{€}{hora} = 10.315{,}00\ €$$

COSTE PERSONAL DIRECTO	10.315,00 €

8.2.2 Coste de amortización de programas y equipos.

En este apartado se incluyen los costes del uso de los diferentes equipos y programas utilizados que tienen una vida útil mayor a la del presente proyecto. Se excluye el uso de sistemas operativos y programas de tipo abierto, como Unity, Ubuntu, ROS y derivados.

Para ello, se calcula su coste total y su tiempo de amortización, que es la vida útil estimada del correspondiente material. A continuación, a dicho producto se le aplica un factor dependiendo del número de años que se haya considerado como tiempo de amortización:

Se ha estimado como tiempo de amortización 5 años, ya que es el que se puede considerar como vida útil del material informático. De forma que al calcular el coste hay que multiplicar por un factor de 0,2 para obtener la amortización para 1 año.

En esta parte entra el mayor coste del proyecto, el robot Reachy, el cual tiene distintos precios dependiendo el paquete y las utilidades que se necesiten como se puede ver a continuación en la imagen 8.2.

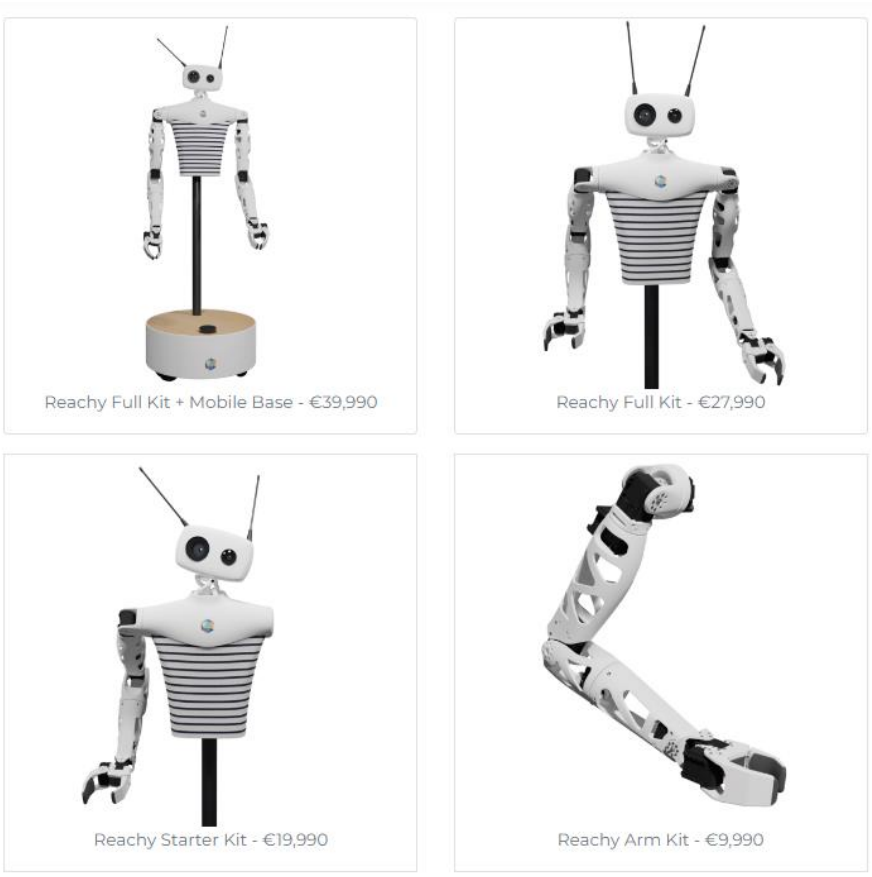

Figura 8.2. Precios de los distintos componentes de Reachy.

En el caso de este proyecto, el utilizado es "Reachy Full Kit + Mobile Base" ya que se quiere que el robot sea capaz de manipular objetos, pero también que tenga la capacidad de desplazarse por sí mismo, tiene un precio bastante elevado, pero según la adopción y los avances tecnológicos vayan siendo mayores este precio podrá disminuir. Se añadirá este elemento como un coste, pero por ahora, lo único que se realiza son simulaciones, como en la mayoría de proyectos tecnológicos antes de probarlo en el sistema real, las cuales son mucho más económicas y sirven para comprobar el funcionamiento y las habilidades de este robot.

En la siguiente tabla (tabla 4) se muestra el resumen de los costes de la amortización anual de los programas y equipos utilizados en el trabajo.

MATERIAL	IMPORTE	AMORTIZACIÓN ANUAL (20%)
Sistema operativo Windows 10	251,10 €	50,22 €
Microsoft Office 2021	60 €/año	60 €
Ordenador	1.419,00 €	284 €
Robot humanoide Reachy	39.990,00 €	7.998,00 €
Total material		**8.392,02 €**

Tabla 4 Costes de amortización

El coste final por hora de utilización del material es calculado mediante la división de la amortización anual entre el número de horas de uso en dichos equipos, es decir, el número de horas efectivas de trabajo en un año:

$$\frac{8.392{,}02\,\frac{€}{año}}{1.767{,}12\,\frac{horas}{año}} = 4{,}75\,\frac{€}{hora}$$

Por último, para conseguir el coste real de amortización del material se multiplica el coste por hora por el número total de horas que se han utilizado los equipos y programas:

$$500\ horas\ x\ 4{,}75\,\frac{€}{hora} = 2.374{,}5\ €$$

COSTE AMORTIZACION EQUIPOS Y PROGRAMAS	2.734,50 €

8.2.3 Costes directos totales.

Los costes directos totales, se calculan realizando la suma de todos los anteriormente calculados, mano de obra y amortización de programas y equipos, por lo tanto:

10.315,00 € + 2.734,50 € = 13.049,50 €

COSTES DIRECTOS	13.049,50 €

8.3 Costes indirectos.

Los costes indirectos (observar tabla 5) son aquellos producidos por la actividad asociada al proyecto, pero que no se pueden atribuir directamente al mismo.

COSTES INDIRECTOS	
Dirección y servicios administrativos	150 €
Consumo de electricidad	150 €
Consumo de internet	50 €
Total gatos indirectos	**350 €**

Tabla 5 Costes indirectos

COSTES INDIRECTOS	350 €

8.4 Costes totales.

Los costes totales son el resultado de sumar todos los gastos anteriores, es decir, los costes directos e indirectos como podemos ver a continuación en la tabla 6.

COSTES TOTALES	
Costes directos	13.049,50 €
Costes indirectos	350 €
Coste total	**13.399,50 €**

Tabla 6 Costes totales del proyecto

En conclusión, el coste total del proyecto asciende a la siguiente cantidad:

COSTES TOTALES DEL PROYECTO	13.339,50 €

Capítulo 9:

Conclusiones y futuros desarrollos

Llegado este punto, cabría recapitular el proceso de desarrollo de este proyecto, marcando los objetivos alcanzados, los problemas encontrados y posibles líneas de desarrollo que podrían aplicarse en un futuro para hacer aún más completo el trabajo.

Como se ha podido atender a lo largo de la memoria, los objetivos principales se han alcanzado con éxito gracias a un trabajo de análisis y formación previos.

Se ha corroborado la posible versatilidad que puede ofrecer un robot humanoide como Reachy al que, a sus ya múltiples utilidades, se le añade la función de teleoperación que le permite abrir un nuevo abanico para desarrollar aplicaciones sin límites. En esta nueva función cabe destacar la introducción de la realidad virtual, la encargada de dotar de una mayor flexibilidad al desarrollo de las aplicaciones.

La creación de la aplicación descrita en este Trabajo Fin de Grado fue todo un reto. Principalmente debido a que la implementación de la realidad virtual en el ámbito de las simulaciones con el robot Reachy no goza de un gran desarrollo por parte de sus creadores. Pero una vez logrado, se convierte en un simulador de Reachy en el cual probar el comportamiento del robot se convierte en algo entretenido y divertido.

Todo esto supone un importante logro y avance en el campo de la robótica y en el de la realidad virtual, especialmente en el robot estudiado, ya que como se ha comentado anteriormente, no existe un sistema previo que pueda realizar comportamientos tan reales del robot.

Estas simulaciones permiten obtener grandes desarrollos con la tecnología sin un gran impacto económico. Y una vez se haya comprobado el correcto funcionamiento de estos progresos se puede implementar en los equipos reales.

El control de Reachy mediante las gafas de realidad virtual se puede realizar teniendo conocimientos básicos sobre esta tecnología, como es el buen manejo de la realidad virtual. Por otra parte, si se desea son conocimientos que son fáciles de adquirir con un corto aprendizaje.

De esta manera, este robot humanoide destinado a la ayuda de personas dependientes, puede ser teleoperado por distintas personas dependiendo la necesidad. Si las tareas a realizar son domésticas o simples, un familiar o amigo podría controlar el robot. Por lo contrario, si la persona dependiente necesita una valoración de un médico o la necesidad de cualquier personal cualificado, simplemente utilizando dicho personal el equipo de realidad virtual podría realizar la acción conveniente gracias a la teleoperación del robot que se encuentra en la estancia de la persona con problemas.

La gran ventaja que permite la teleoperación de Reachy es el evitar el desplazamiento, frecuentemente de gran distancia, para realizar una ayuda muy importante pero no compleja.

Se puede observar que en estos proyectos revolucionarios llenos de nueva tecnología llevarlos a escena supone un gran coste. A medida que la tecnología evoluciona, que como se puede ver es a pasos agigantados, estas implementaciones supondrán menores impactos económicos y podrán tener una mayor adopción como ha ocurrido a lo largo del tiempo.

En este proyecto, el gran avance construido es la aplicación para simular distintos comportamientos de Reachy en un entorno similar donde se quiere implementar. Este desarrollo es de gran utilidad para realizar futuros desarrollos partiendo de esta teleoperación simulada.

Como se comentó al final del capítulo 6, el control remoto mezclando distintas herramientas como son Unity y ROS, podría tener grandes avances en un futuro siguiendo la primera aproximación que se ha desarrollado en este proyecto. Gracias a ese desarrollo se podría visualizar un movimiento más exhaustivo de las articulaciones del robot en RViz a medida que se va ejecutando movimientos con el equipo de realidad virtual en la aplicación de teleoperación simulada.

Una vez vistas las ventajas que Reachy puede ofrecer y comprobadas sus funcionalidades, una interesante futura implementación trataría de conectar las gafas de realidad virtual en el robot Reachy real, para lo cual se necesita un gran desembolso como se ha visto en el capítulo anterior. Para la realización de esta conectividad entre robot real y realidad virtual, Pollen Robotics, los creadores de Reachy, facilitan un paso a paso para mayor comodidad. De esta manera, es casi seguro que la conexión real entre estas dos tecnologías se realice sin problemas y al

instante. Lo que permitirá comprobar y utilizar el robot en un entorno domestico real y así ver la gran flexibilidad y ayuda que puede significar este robot en el ámbito social.

Bibliografía y referencias

Aldebaran. (s.f.). Aldebaran United Robotics Group. Obtenido de Pepper: https://www.aldebaran.com/es/pepper#:~:text=Pepper%20es%20el%20primer%20robot,los%20sectores%20empresarial%20y%20educativo.

Amanda. (2 de Octubre de 2021). GitHub. Obtenido de Unity-Technologies/Unity-Robotics-Hub: https://github.com/Unity-Technologies/Unity-Robotics-Hub/blob/main/tutorials/ros_unity_integration/README.md

Arbeláez Buriticá, C. A. (2018). Agencia de noticias UPB. Obtenido de SOPHIA, robot humanoide – Hanson Robotics: https://www.upb.edu.co/es/documentos/doc-boletinescuelaverano2-1464182415182.pdf

Canelas Martín, I. (2020). La evolución de la dependencia en España. Obtenido de https://uvadoc.uva.es/handle/10324/45379

Carbone. (s.f.). Carbone. Obtenido de Lentes de realidad virtual: ¿Para qué sirven?: https://carbonestore.com/blogs/news/lentes-de-realidad-virtual#:~:text=Las%20gafas%20de%20realidad%20virtual,la%20retina%20de%20los%20mismos.

Contreras, M. (2017). Clipset. Obtenido de Toyota T-HR3 el robot humanoide que replica tus movimientos: https://clipset.com/robot-toyota-t-hr3/

Crampette, A. (18 de Diciembre de 2020). Medium. Obtenido de Orbita is turning heads… literally: https://medium.com/pollen-robotics/orbita-is-turning-heads-literally-d10d378550e2

Free3D. (s.f.). Obtenido de Modelos FBX: https://free3d.com/

Gelin, A. K. (11 de Junio de 2019). RobotLAB. Obtenido de Un robot humanoide sociable producido en masa: Pepper: la primera máquina de este tipo:

https://www.robotlab.com/research-papers/a-mass-produced-sociable-humanoid-robot-pepper-the-first-machine-of-its-kind

Goff, S. L. (23 de Noviembre de 2021). Reachy 2021 Documentation. Obtenido de Python SDK. The SDK in a nutshell: https://docs.pollen-robotics.com/sdk/getting-started/introduction/

GR, R. (18 de Mayo de 2022). Adsl Zone. Obtenido de Realidad Virtual, la tecnología que ya está cambiando nuestras vidas: https://www.adslzone.net/reportajes/tecnologia/realidad-virtual-rv/

HM Hospitales. (s.f.). Obtenido de Cirugía Robótica: https://www.hmsanchinarro.com/especialidades/programas-medicos/robot-da-vinci

hmong. (s.f.). hmong. Obtenido de Oculus Touch: https://hmong.es/wiki/Oculus_Touch

Iberdrola. (2020). Iberdrola. Obtenido de Innovación, Realidad Virtual: https://www.iberdrola.com/innovacion/realidad-virtual#:~:text=%C2%BFQU%C3%89%20ES%20LA%20REALIDAD%20VIRTUAL,o%20casco%20de%20Realidad%20Virtual.

Izquierdo Gómez, M. (2020). Puesta en servicio de un robot social y desarrollo de un conjunto de herramientas de ayuda asistencial. Obtenido de https://uvadoc.uva.es/handle/10324/41176

Jose Francisco. (15 de Enero de 2020). Centro Integrado de Formación Profesional Número Uno de Santander. Obtenido de Departamento de Electricidad-Electrónica: https://cifpn1.com/electronica/?p=5534

Juárez Sánchez, A. (2014). Desarrollo de un sistema de visión 3D para su integración en un robot móvil social. Obtenido de https://uvadoc.uva.es/handle/10324/14803

knob2001. (1 de Mayo de 2018). realovirtual. Obtenido de Oculus Go: Análisis: https://www.realovirtual.com/articulos/5171/oculus-go-analisis

Lannuzel, G. (22 de Agosto de 2022). Reachy2021-UnityPackageSimulator. Obtenido de https://github.com/pollen-robotics/reachy2021-unity-package/releases

MasterD. (s.f.). MasterD. Obtenido de Qué es Unity y para qué sirve: https://www.masterd.es/blog/que-es-unity-3d-tutorial

MoveIt. (s.f.). MoveIt. Obtenido de Concepts: https://moveit.ros.org/documentation/concepts/

Naciones Unidas. (s.f.). Noticias ONU. Obtenido de La Inteligencia Artificial como herramienta para acelerar el progreso de los ODS: https://news.un.org/es/story/2017/10/1387731

Pastor, J. (24 de Junio de 2015). Xakata. Obtenido de Este es Pepper, el primer robot humanoide que aspira a conquistar el mercado masivo: https://www.xataka.com/robotica-e-ia/este-es-pepper-el-primer-robot-humanoide-que-aspira-a-conquistar-el-mercado-masivo

Pollen Robotics. (s.f.). Obtenido de https://www.pollen-robotics.com/

POLLEN ROBOTICS. (s.f.). Obtenido de POLLEN ROBOTICS: https://www.pollen-robotics.com/

Pollen Robotics, video1. (s.f.). headmotion [vídeo]. Pollen Robotics. Obtenido de https://www.pollen-robotics.com/video/headmotion.mp4

Pollen Robotics, video2. (s.f.). flowers_square [vídeo]. Pollen Robotics. Obtenido de https://www.pollen-robotics.com/video/flowers_square.mp4

POLLENROBOTICS. (s.f.). Reachy 2021 VR. Using the App. Best practise. Obtenido de https://docs.pollen-robotics.com/vr/use-teleop/best-practice/

Poly Haven. (s.f.). Obtenido de Textures: https://polyhaven.com/textures

Riva, J. E. (18 de 08 de 2021). ROS.org. Obtenido de Introduccion: http://wiki.ros.org/es/ROS/Introduccion

Robotics, P. (s.f.). About us. Obtenido de https://www.pollen-robotics.com/about/

Robotics, P. (s.f.). What's Reachy. Obtenido de https://www.pollen-robotics.com/reachy/#discover

Robotnik. (8 de Julio de 2022). Robotnik. Obtenido de Ejemplos de robótica de servicio: https://robotnik.eu/es/ejemplos-de-robotica-de-servicio/#:~:text=Un%20robot%20de%20servicio%20es,automatizaci%C3%B3n%20industrial%E2%80%9D%20(IFR).

Rouanet, P. (24 de Junio de 2022). Reachy gana movilidad. Obtenido de https://medium.com/pollen-robotics/reachy-just-gained-mobility-with-an-open-source-omnidirectional-base-bce9b8e430a8

Toyota. (21 de Noviembre de 2017). Sala de prensa Toyota. Obtenido de Toyota T-HR3: robot humanoide de tercera generación: https://prensa.toyota.es/toyota-t-hr3-robot-humanoide-de-tercera-generacion/

Willings, A. (30 de Marzo de 2022). Pocket-lint. Obtenido de Meta Quest 2 vs Oculus Quest: https://www.pocket-lint.com/es-es/ra-y-rv/guias-del-comprador/oculus-rift/153716-oculus-quest-2-vs-oculus-quest-cual-es-la-diferencia

[1] Imagen obtenida de: https://www.discoazul.com/robot-aspirador-irobot-roomba-e6.html

[2] Imagen obtenida de: https://www.hmsanchinarro.com/especialidades/programas-medicos/robot-da-vinci

[3] Imagen obtenida de: https://thinkingheads.com/en/speakers/sophia-the-ai-robot/

[4] Imagen obtenida de: https://news.un.org/es/story/2017/10/1387731

[5] Imagen obtenida de: Izquierdo Gómez, Miguel. Puesta en servicio de un robot social y desarrollo de un conjunto de herramientas de ayuda asistencial. 2020

[6] Imagen obtenida de: https://prensa.toyota.es/toyota-t-hr3-robot-humanoide-de-tercera-generacion/

[7] Lannuzel, G. (28 de Junio de 2022). Medium. Imagen obtenida de: https://medium.com/pollen-robotics/controlling-a-reachy-robot-in-unity-f3d90d550345

[8] Pollen Robotics. Reachy. Imagen obtenida de: https://www.pollen-robotics.com/reachy/

[9] Crampette, A. (18 de Diciembre de 2020). Medium. Imagen obtenida de: https://medium.com/pollen-robotics/orbita-is-turning-heads-literally-d10d378550e2

[10] Rouanet, P. (24 de Junio de 2022). Reachy gana movilidad. Imagen obtenida de: https://medium.com/pollen-robotics/reachy-just-gained-mobility-with-an-open-source-omnidirectional-base-bce9b8e430a8

[11] Imagen obtenida de: https://www.xataka.com/historia-tecnologica/el-primer-simulador-vr-de-la-historia-tenia-forma-de-recreativa-y-se-invento-a-finales-de-los-50

[12] Imagen obtenida de: https://www.adslzone.net/reportajes/tecnologia/realidad-virtual-rv/

[13] Imagen obtenida de: https://www.realovirtual.com/articulos/5171/oculus-go-analisis

[14] Imagen obtenida de: https://www.pocket-lint.com/es-es/ra-y-rv/guias-del-comprador/oculus-rift/153716-oculus-quest-2-vs-oculus-quest-cual-es-la-diferencia

[15] Imagen obtenida de: https://www.tworeality.com/simuladores-realidad-virtual/

[16] Imagen obtenida de: https://docs.pollen-robotics.com/vr/use-teleop/commands/

[17] Imagen obtenida de: https://moveit.ros.org/documentation/concepts/