Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Desarrollo de un sistema de realidad mixta para la robótica colaborativa

Autor:

Calderón Sesmero, Raúl

Tutor(es):

Gómez García-Bermejo, Jaime

Duque Domingo, Jaime

Departamento de Ingeniería de Sistemas y Automática

Valladolid, febrero 2023.

Resumen

El proyecto que plantea este Trabajo de Fin de Grado está basado en un sistema que muestra la viabilidad del uso de la realidad mixta en el entorno robótico colaborativo. La aplicación desarrollada da soporte a un operario sin conocimientos en robótica a la hora de realizar tareas cotidianas con el robot.

El sistema está compuesto por un dispositivo de realidad mixta y un robot colaborativo que ejecuta las indicaciones del operario mediante gestos y comandos de voz. La comunicación entre los dispositivos se realiza mediante una aplicación que se ejecuta en Robot System Operating (ROS).

El sistema es empaquetado en un contenedor con los paquetes y dependencias justas y necesarias para que funcione correctamente.

Este proyecto se ha llevado a cabo en Fundación CARTIF, situada en el Parque Tecnológico de Boecillo (Valladolid) en el marco del programa “Centros Tecnológicos de Excelencia Cervera” (CER-20211007).

Palabras clave

Realidad mixta, Robots Colaborativos, ROS, Docker, Sockets

Abstract

The project proposed in this Final Degree Project is based on a system that shows the feasibility of the use of mixed reality in the collaborative robotic environment. The developed application supports an operator with no knowledge in robotics when performing daily tasks with the robot.

The system is composed of a mixed reality device and a collaborative robot that executes the operator's instructions through gestures and voice commands. Communication between the devices is done through an application running on Robot System Operating (ROS).

The system is packaged in a container with the necessary packages and dependencies for it to work properly.

This project has been carried out at the CARTIF Foundation, located in the Boecillo Technology Park (Valladolid) within the framework of the "Cervera Technology Centers of Excellence" program (CER-20211007).

Keywords

Mixed Reality, Collaborative Robots, ROS, Docker, Sockets

Agradecimientos

Agradezco enormemente el apoyo y confianza que me han dado todas las personas cercanas a mí durante este periodo universitario lleno de momentos de todo tipo, pero sobre todo de alegría.

Agradezco a mis padres la educación que me han brindado y todo el esfuerzo que han realizado para conseguir que su hijo haya podido completar un grado universitario.

Agradezco a mi hermana todo su amor y apoyo incondicional, siempre estaré orgulloso de ella.

Agradezco a mis amigos, aquellos que han hecho de este periodo universitario una época que siempre recordaré con nostalgia. Espero que el camino que nos queda juntos no tenga fin.

Agradezco a CARTIF y a mis compañeros de trabajo la oportunidad de haber podido hacer frente a este proyecto con dispositivos tan sofisticados como Microsoft Hololens 2 y, poder haberme echado una mano siempre que han podido.

Por último, agradezco a Sofía, por haberme cambiado la vida y sacar la mejor versión de mi en todos los aspectos.

A todos agradezco de corazón el haber estado junto a mí durante todo este ciclo. Mil gracias.

Índice general

Índice de ilustraciones

Índice de tablas

Capítulo 1. Introducción

Este primer capítulo expone una introducción general al proyecto y se divide en cuatro apartados.

En el apartado 1.1 se aborda el marco del proyecto desde una perspectiva histórica-evolutiva de la temática del trabajo haciendo referencia a su impacto global en la sociedad. Además, se presenta brevemente la aplicación a desarrollar.

La justificación de la necesidad de este TFG (Trabajo de Fin de Grado) y las implicaciones personales que este trabajo supone vienen detalladas en el apartado 1.2.

En el apartado 1.3 se establecen los objetivos principales y secundarios que se pretenden alcanzar con el sistema de realidad mixta desarrollado.

Por último, el apartado 1.4 define la estructura de este documento.

1.1 Contexto

A principios del siglo XX, era habitual encontrarse una industria en la que los humanos se sometían a tareas repetitivas y, en algunos casos, peligrosas. De esta problemática, surge la idea de crear un sistema que sea capaz de imitar a los humanos, automatizando las operaciones que realizan en su puesto de trabajo, nace el concepto de *"Robot".*

Karel Čapek introduce esta palabra en su obra teatral de ciencia ficción *"R.U.R (Rossum's Universal Robots)"* publicada en 1920, pero no será hasta la década de 1960 que los robots industriales "Unimate" hagan su aparición en las fábricas de General Motors en Ewing (Nueva Jersey). [1] En la Ilustración 1 se puede contemplar el aspecto de este robot.

Ilustración 1. Primer robot industrial de la historia (Fuente: [1])

Con la llegada de Unimate, comienza la era de los robots industriales y las grandes compañías empiezan a construir sus propios prototipos para implantarlos en sus fábricas. Desde ese momento, el ser humano ha tratado de mejorar estas máquinas haciéndolas más eficientes y capaces de realizar tareas tediosas con mucha más velocidad y precisión que las que puede desempeñar cualquier persona.

Pero tarde o temprano, se descubrió que no todo eran ventajas, los robots industriales se adueñaron del trabajo de mucha gente, eran costosos y ocupaban gran espacio dentro del entorno laboral. *"Con los robots tradicionales, los costos de capital para los propios robots representan solo del 25 al 30 por ciento de los costos totales del sistema",* dice Shepherd. [2]

Debido a los inconvenientes de los robots industriales, surge una nueva dinámica de trabajo que permite la cooperación entre humanos y robots y, además, dotar este entorno de una mayor seguridad y espacio, nacen los *"cobots" o* robots colaborativos. Este término hace alusión al tipo de robots que interactúan físicamente junto a los humanos en el mismo entorno de trabajo.

A medida que esta nueva tecnología se iba desarrollando, la mente humana se embarcaba en una cuarta revolución industrial, *"La Industria 4.0"*. Las tecnologías de fabricación viven una transformación sin precedentes, sofisticados sistemas informáticos, sensores por todas partes, potentes procesadores, novedosas tecnologías de telecomunicación, etc. Todo lo que nos rodea empieza a estar conectado. [3]

La Tabla 1 recoge las etapas en las que se divide la Revolución Industrial dejando constancia de las innovaciones tecnológicas que introdujeron en la sociedad.

Fase	Cronología	Origen	Características
1	1784-1870	Primer telar mecánico	Generación de vapor
2	1870-1969	Primera línea de montaje	Producción en cadena
3	1969-2000	Primer controlador lógico programable	Mejor nivel de automatización
4	2011-Actualidad	Sistemas ciber físicos	Interconectividad y automatización inteligente

Tabla 1. Cronología de las revoluciones industriales

La Industria 4.0 incorpora una serie de herramientas que son capaces de acelerar la curva de aprendizaje del personal que requiere actualizar su formación para adaptarse a estas nuevas tecnologías, un ejemplo es *"La realidad extendida"*.

Wikipedia define este concepto de la siguiente forma: *"La realidad extendida (ER) es el término referente a la unión de elementos virtuales y reales, e interacciones humano-máquina gracias al uso de determinados dispositivos. Dicho término engloba a la realidad virtual, aumentada y mixta".* Más adelante se profundizará en las similitudes y diferencias entre estos tipos de realidad extendida.

Una de las grandes ventajas de la realidad extendida es la de permitir al usuario permanecer en contacto con su entorno de trabajo de forma que su foco de atención no esté en el ordenador, sino en el mundo real. [4]

Grandes compañías apuestan por esta tecnología creando sus propios dispositivos de realidad extendida con la idea de ofrecer mejores prestaciones que su competencia como Microsoft, Google o HTC. El ecosistema de comunicaciones está evolucionando rápidamente y los proveedores de redes ya están trabajando para implementar la infraestructura que satisfaga todas las necesidades. [5]

No obstante, esta tecnología aún tiene mucho que mejorar. El campo de visión de los dispositivos de realidad extendida aún es muy pequeño y la resolución de los objetos digitales que se observan deja mucho que desear. Además, el mercado de aplicaciones industriales de realidad extendida que ofrecen las compañías de desarrollo informático para este tipo de dispositivos es muy escaso y caro.

En este contexto, la idea de combinar los robots colaborativos y los sistemas de realidad extendida parece atractiva, ya que las ventajas que podrían significar para las empresas que la implementasen serían considerables. La calidad del soporte formativo del personal mejoraría, la dinámica de trabajo sería mucho más entretenida, se agilizaría el flujo de trabajo y la variedad de soluciones que se podrían adoptar en el ámbito industrial sería mucho más flexible.

Lo que se pretende con este Trabajo de Fin de Grado (TFG) es hacer realidad esta idea. El planteamiento del proyecto se basa en desarrollar una aplicación que haga uso de un tipo de realidad extendida, la realidad mixta, para integrarla en la práctica robótica colaborativa y, así facilitar las labores de un operario a la hora de realizar ciertas actividades sin una formación previa.

Este proyecto se ha llevado a cabo en Fundación CARTIF, situada en el Parque Tecnológico de Boecillo (Valladolid) en el marco del programa "Centros Tecnológicos de Excelencia Cervera" (CER-20211007). La iniciativa que pretende impulsar este proyecto es la de establecer una red colaborativa, dotada con la necesaria tecnología, herramientas e infraestructuras, para actuar como elemento tractor del desarrollo e introducción de nuevas tecnologías robóticas en el tejido industrial de fabricación.

En la Ilustración 2 se muestra una fotografía realizada al edificio que alberga las instalaciones de Fundación CARTIF.

Ilustración 2. Fundación CARTIF (Fuente: [6])

En las instalaciones de Fundación CARTIF se encuentra una célula equipada con robots colaborativos e instalaciones de vanguardia con demostradores permanentes para el testeo e investigación de tecnologías en escenarios realistas. Es en esta célula donde se han llevado a cabo las labores de investigación y desarrollo que han dado lugar a este TFG. La Ilustración 3 muestra un diseño 3D de la fábrica piloto donde se llevan a cabo las actividades de este proyecto.

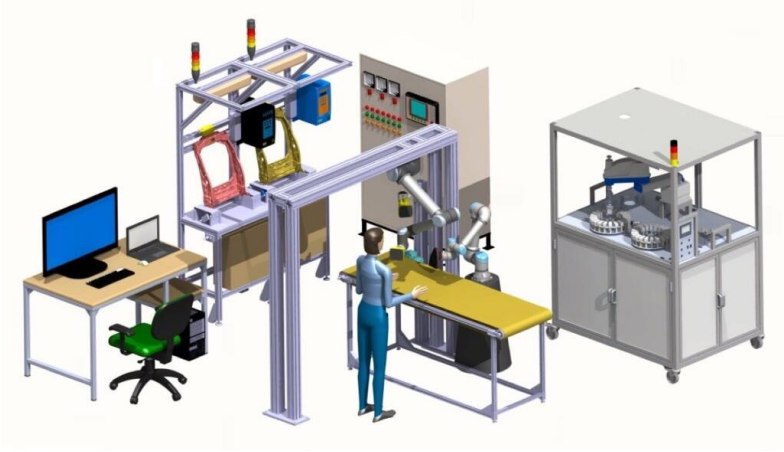


Ilustración 3. Diseño 3D de la Fábrica Piloto para operaciones con materiales flexibles en CARTIF

A lo largo de este documento se muestran imágenes de los ensayos realizados en esta célula piloto.

1.2 Motivación

"La realidad virtual nos va a permitir entrar y navegar dentro de la imagen. Antes la imagen servía para transformar el mundo; ahora la imagen virtual es el mundo" (Phillipe Quéau).

En concordancia con esta cita, se puede hacer una idea de las veces que los medios de comunicación han promulgado los grandes avances de la realidad extendida en la sociedad, asegurando que el futuro de esta tecnología es prometedor.

Esta tecnología está cobrando importancia en muchos de los sectores profesionales: educación, videojuegos, sanidad, etc. Esto es debido a que esta tecnología no solo simplifica y transforma las actividades, sino que además es rentable.

De acuerdo con un reporte elaborado por Boston Consulting Group, las compañías que digitalizan procesos pueden esperar que sus márgenes gananciales aumenten de un 12 a 20% en promedio, y podrán obtener hasta 50% de las ganancias adicionales en el primer año, generando los recursos para financiar el resto de su transformación. [7]

Teniendo presente la inquietud que provoca esta tecnología desde hace años en la sociedad, la oportunidad de indagar en los entresijos que esconde la realidad extendida permite introducirse en el Metaverso y analizar su estructura y funcionamiento.

Si se añade el componente adicional de integrar esta tecnología en una célula de trabajo con robots colaborativos, el resultado que este proyecto ofrece resulta atractiva, ya que se podrían desarrollar aplicaciones que permitan la cooperación humano-robot. De esta manera, se podría conseguir que la dinámica de trabajo de un operario fuese más entretenida y eficiente.

Como resultado, las empresas que incorporasen estos elementos en su área de trabajo podrían ahorrar tiempo y dinero buscando un profesional específico en la labor que se requiere.

1.3 Objetivos

El objetivo principal de este proyecto es el desarrollo de una aplicación que muestre la viabilidad en el uso de la realidad mixta a la hora de hacer frente a las tareas cotidianas de un operario con un robot colaborativo. No obstante, se pretenden cumplir una serie de subjetivos relacionados con el propósito principal de este proyecto.

- Asistencia remota.
  La aplicación de realidad mixta a desarrollar debe de servir de soporte para un operario sin experiencia previa en robótica colaborativa. Esta aplicación debe de guiar al usuario de tal manera que no pierda el foco de atención en el entorno de trabajo.

- Cooperación humano-máquina.
  El robot debe de ser capaz de trabajar con el humano en tiempo real. Por ello, el desarrollo de la actividad que se realice con la aplicación de realidad mixta debe de ejecutarse al mismo tiempo que la del robot.

- Consecución de las maniobras y trayectorias indicadas mediante lenguaje natural.
  El robot tendrá que replicar las trayectorias establecidas en la aplicación de realidad mixta. El método empleado constará de programas que recopilen la información que se envíe desde un dispositivo de realidad mixta y generen las instrucciones indicadas en el lenguaje nativo del robot colaborativo.

- Precisión y exactitud aceptable.
  El registro de las coordenadas indicadas por el usuario y grabadas por el dispositivo de realidad mixta tienen que corresponder en gran medida a las coordenadas locales del robot.

- Automatización de trayectorias.
  La Industria 4.0 se enfoca en la automatización de los procesos y la comunicación de datos en tiempo real. El usuario debe de poder hacer uso de la realidad mixta para establecer maniobras con el robot que puedan repetirse indefinidamente hasta el momento en el que se desee detener su movimiento.

- Otorgar seguridad a la actividad.
  El operario tiene que ser capaz de manipular el robot de una forma segura durante la ejecución de la aplicación de realidad mixta evitando que se puedan producir situaciones de riesgo.

- Repuesta ágil y fiable.
  El robot debe de ser capaz de ejecutar las maniobras programadas una vez se ha enviado la orden desde la aplicación de realidad mixta en un intervalo de tiempo correcto.

1.4 Estructura del documento

La estructura de este documento está dividida en siete capítulos. Tras el capítulo 1 que introduce el proyecto, el documento se organiza de la siguiente manera:

- **Capítulo 2: Análisis de la tecnología empleada.**
  Este capítulo sirve para resaltar algunos de los conceptos técnicos necesarios para entender el resto del documento de manera correcta. Posteriormente, se hace un análisis de los trabajos previos relacionados con este TFG junto con las soluciones y limitaciones que ofrece cada uno.

  Por último, se analizan las problemáticas actuales a la hora de diseñar y desarrollar un sistema de realidad mixta orientado a la robótica colaborativa, resaltando las soluciones y conclusiones recogidas.

- **Capítulo 3: Desarrollo del sistema de realidad mixta.**
  En este capítulo se aborda el núcleo central del proyecto comenzando por enumerar el soporte físico y lógico utilizado.

  A continuación, se describe el diseño y desarrollo del sistema de realidad mixta, dejando constancia de las diversas alternativas valoradas previamente y exponiendo las soluciones que ofrecen las opciones seleccionadas.

  Finalmente, se definen los aspectos de diseño e implementación que engloban este TFG exponiendo los métodos y líneas de trabajo realizados en el marco práctico del proyecto.

- **Capítulo 4: Experimentación y resultados.**
  El capítulo 4 presenta el arranque de todo el sistema y revela las pruebas de precisión y exactitud realizadas para extraer conclusiones y validarlo.

- **Capítulo 5: Gestión del trabajo.**
  En este capítulo se hace constancia de cómo se ha gestionado el tiempo de realización del proyecto.

- **Capítulo 6: Conclusiones y líneas futuras.**
  Este capítulo recoge las conclusiones principales del documento y presenta las líneas futuras de trabajo del proyecto.

Capítulo 2. Análisis de la tecnología empleada

En este capítulo se presentan algunos de los conceptos básicos y fundamentales para entender el lenguaje técnico de este proyecto. Este capítulo se divide en tres apartados.

En el apartado 2.1 se aborda el marco teórico de las dos principales temáticas que se tratan en este proyecto: la realidad mixta y la robótica colaborativa. Además, se analizan algunos de los trabajos realizados por otros autores que integran ambas tecnologías en un mismo sistema.

A continuación, en el apartado 2.2 se realiza un análisis de las problemáticas y circunstancias a tener en cuenta a la hora de diseñar y desarrollar una aplicación de realidad mixta que acompañe en la práctica robótica colaborativa.

Por último, se exponen las conclusiones de este capítulo en el apartado 2.3.

2.1 Marco teórico

En este apartado se introducen algunos de los conceptos técnicos fundamentales que son necesarios para entender el lenguaje técnico de este proyecto. Posteriormente, se analizan algunos de los trabajos previos relacionados con este TFG exponiendo las soluciones que resuelve cada uno y sus limitaciones e inconvenientes.

2.1.1 Conceptos básicos

En primer lugar, se define qué es la realidad mixta, sus tipos y las diferencias entre estos. También se hace referencia a la instrumentación más relevante en el mercado que hace uso de esta tecnología, analizando sus especificaciones técnicas de mayor importancia. De forma similar, se hace con la robótica colaborativa, pero haciendo más hincapié en sus aplicaciones y soluciones en la industria.

2.1.1.1 La realidad mixta

En ocasiones, surge un tema de actualidad que está relacionado con los términos *"realidad virtual"* o *"realidad aumentada"*, pero son muchas las personas que las engloban y no consideran su posible diferencia. Es preciso resaltar las similitudes y diferencias entre estas dos tecnologías para entender el lenguaje técnico de este documento de la mejor manera posible.

Tanto la realidad virtual como la realidad aumentada utilizan mecanismos para crear objetos y espacios virtuales que permiten vivir una experiencia inmersiva gracias al uso de dispositivos visuales. No obstante, la realidad virtual (VR) es más inmersiva, es decir, todo lo que vemos es 100% virtual, y no hay rastro del mundo físico. [8]

Con la realidad virtual la persona queda limitada al lugar físico en el que se encuentra en el momento de la experiencia. [8] Existe la posibilidad de interactuar con los objetos digitales mediante mecanismos físicos como mandos, guantes, etc.

Por ejemplo, el dispositivo Oculus Quest incorpora unos mandos que se sincronizan con las gafas de realidad virtual para obtener la posición de las manos en todo momento. Estos mandos contienen una serie de botones que permiten interactuar con estos objetos virtuales.

Por otro lado, está la realidad aumentada (AR). La principal diferencia que existe entre la AR y la VR es que, en esta ocasión, no se obstruye el sentido de la vista, sino que se añade información. [9]

Para comprender mejor los rasgos que definen a la realidad virtual, se adjunta la Ilustración 4. En esta imagen, se puede apreciar al objeto virtual (pato de goma) sobre un espacio completamente virtual.

Ilustración 4. Escena de realidad virtual (Fuente: [10])

La gran desventaja de la realidad aumentada es que no permite la interacción entre el mundo real y los objetos digitales. Además, la persona se encuentra limitada al lugar físico en el que se encuentra en el momento de la experiencia. No obstante, la AR permite al usuario una mayor movilidad, ya que la tecnología de los dispositivos que hacen uso de esta tecnología son inalámbricos.

En la Ilustración 5 se muestra una escena en la que aparece el mismo objeto digital que en la figura anterior, pero diseñado con realidad aumentada. En esta imagen, se muestra al objeto digital superpuesto en el mundo real, pero se puede observar cómo es “atravesado” por los objetos del entorno en el que se encuentra.

Ilustración 5. Escena con realidad aumentada (Fuente: [10])

Tanto la realidad virtual como la realidad aumentada tienen sus ventajas e inconvenientes, pero ante la cuestión de combinar ambas tecnologías para obtener los mejor de cada una, nace la realidad mixta.

En la MR ya no se superpone información sobre el mundo real, sino que se fusiona el mundo físico con el mundo digital. Esto quiere decir que, si se tiene un elemento, como puede ser una silla modelada en 3D, se va a poder colocarla en el mundo físico y esa silla va a "ser consciente" del mundo que le rodea: va a entender dónde está el suelo y, si pasa alguien por delante, va a tapar dicha silla. [8]

En la Ilustración 6 se muestra el mismo objeto digital que en las dos últimas imágenes en un entorno constituido por la realidad mixta. Ahora, el pato de goma entiende el entorno que le rodea y es capaz de interactuar con él. Se puede contemplar cómo se coloca detrás de la mesa y no es atravesado por ella como ocurría con la realidad aumentada.

Ilustración 6. Escena con realidad mixta (Fuente: [10])

Para entender mejor estos conceptos, en la Ilustración 7 se representa de manera gráfica un espectro de barras que engloba estos tipos de realidades digitales y las sitúa según su grado proximidad al mundo físico y/o digital.

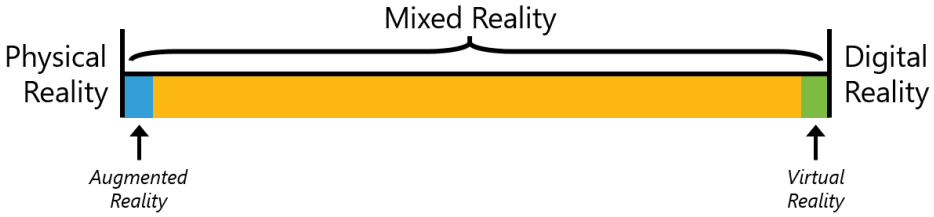


Ilustración 7. Espectro de realidad mixta (Fuente: [11])

La realidad mixta es una tecnología en auge que requiere de dispositivos físicos que sean capaces de procesarla y, a su vez, ofrecer buenas prestaciones. Los dispositivos de realidad mixta de mayor relevancia en el mundo de la realidad mixta se exponen a continuación.

Microsoft Hololens 2:

Este dispositivo, lanzado al mercado en el año 2019, mejora a su antecesora, Microsoft Hololens 1. El diseño de este dispositivo consta de un casco inalámbrico con una visera que incorpora las lentes por las que se observa la información digital. El modelo de HL2 se presenta en la Ilustración 8.

Ilustración 8. Modelo de Microsoft Hololens 2 (Fuente: [12])

Microsoft Hololens 2, de aquí en adelante HL2, utiliza una tecnología llamada MEMS mirror (Microelectrical-mechanical System) que se basa en hacer vibrar 2 micro-espejos entre sí para hacer rebotar los haces de luz RGB de los láseres por los cristales de las gafas y así formar la imagen en el fondo de la retina, no en la superficie del cristal.

Cabe destacar que la resolución que se realiza es de 54.000 veces por segundo. Esto permite a HL2 ser el único dispositivo MR del mercado capaz de ofrecer al usuario información digital en forma de texto.

El visor de HL2 tiene 52 grados de diagonal, mientras el primer HoloLens tenía 34 grados. La relación de aspecto del visor es de 3:2, por lo que nos daría un FOV horizontal de 43º y vertical de 29º. [13]

HL2 posee un sensor de profundidad tipo ToF (Time Of Flight) que es utilizado de dos maneras diferentes:

- Para el seguimiento de las manos.
- Para mapear el entorno. Este último es de destacar ya que es el que da un enfoque de realidad mixta al dispositivo evitando que los objetos digitales traspasen los límites físicos del escenario.

Magip Leap 2:

Tras el lanzamiento de Magic Leap One en 2018, su versión actualizada, Magic Leap 2, lanzado en septiembre de 2022, representa el dispositivo de realidad mixta más reciente del mercado.

El diseño de este dispositivo es mucho más delgado y compacto que HL2. No obstante, Magic Leap 1 y Magic Leap 2 incorporan un pequeño accesorio denominado *"Lightpack"* que actúa como ordenador, y un controlador inalámbrico con botones, touchpad y vibración háptica, tal como se aprecia en la Ilustración 9.

Ilustración 9. Modelo de Magic Leap 1 (Fuente: [14])

Magic Leap 2, de aquí en adelante ML2, posee una característica muy peculiar denominada *"atenuación segmentada"* que permite apagar la pantalla como un par de gafas de sol que pueden atenuar toda la luz del entorno para centrar la atención en la información que proyecta el dispositivo en el visor. De esta manera, ML2 se podría asemejar a un dispositivo de realidad virtual. Su campo de visión es de 70 grados en diagonal y su resolución es de 1440x1760 píxeles en sus pantallas, [14] mayor que el de HL2.

En la Tabla 2 se muestra una tabla comparativa entre las características más importantes de las series Hololens y Magic Leap. Los datos de esta tabla han sido recogidos de la siguiente fuente: https://vr-compare.com/compare?h1=tSCQxsAA_&h2=EkSDYv0cW&h3=1N3k3S4MN&h4=mt3AEYJu5

	Hololens 1	Hololens 2	Magic Leap 1	Magic Leap 2
Precio (euros)	3,000	3,500	2,295	3,299
FOV (º)	30 x 17.5	43 x 29	40 x 30	44 x 53
Resolución por ojo (px)	1268x720	1440x936	1280x960	1440x1760
Tasa de refresco (Hz)	60	60	120	120
Peso (g)	579	556	316	260
Registro visual	x	✓	✓	✓
Controladores	x	x	✓	✓
Almacenamiento (GB)	64	64	128	128
Atenuación segmentada	x	x	✓	✓
Duración de la batería (horas)	3	3	3	3.5

Tabla 2. Características de dispositivos MR

Como se puede contemplar en la Tabla 2, ML2 parece ser que reúne las mejores especificaciones técnicas. No obstante, tiene la gran desventaja de tener que hacer uso del Lightpack para ser utilizado.

Por otro lado, se quiere proyectar información en forma de texto para ofrecer el soporte al operario que vaya a utilizar la aplicación de realidad mixta, por este motivo es mejor opción hacer uso de HL2.

2.1.1.2 La robótica colaborativa

Los cobots han sido creados con el fin de automatizar procesos de producción repetitivos compartiendo un entorno laboral por medio de la interacción entre personas y máquinas. [15] El mercado de los robots colaborativos está en auge y cada vez son más las empresas que necesitan de maquinaria ligera capaz de cooperar con los humanos para poder llevar a cabo ciertas tareas.

El origen de la robótica colaborativa se remonta al año 1996 cuando los profesores J.Edward Colgate y Michael Pesshkin inventaron el primer cobot. En un inicio, estos cobots fueron llamados máquinas de restricción programable, lo cual resaltaba la pasividad y seguridad que estos dispositivos garantizaban a los humanos. [16] En la Ilustración 10 se puede contemplar como la brecha entre las ventas de robots tradicionales y colaborativos se ha ido estrechando con el paso de los años.

Ilustración 10. Robots tradicionales vs Cobots en unidades vendidas (Fuente: [17])

Se estima que el valor del mercado de cobots oscilará entre 9.000 y 12.000 millones de dólares en 2025. Según algunas estimaciones, el mercado de cobots tendrá un valor de 9.500 millones de dólares a finales de 2024 y crecerá a una CAGR del 30 % desde ahora hasta entonces. [17]

Como el concepto de robótica colaborativa es muy moderno, existe gran parte de la población que no distingue entre los cobots y los robots tradicionales. Algunas de sus mayores diferencias se recogen en la Tabla 3.

Características	Tradicionales	Cobots
Tamaño y peso	Grande	Pequeño
Espacio de trabajo	Grande	Pequeño
Carga	Grande	Pequeño
Rentabilidad	A largo plazo	A corto plazo
Seguridad y cooperación	Ninguna	Alto
Mantenimiento	Alto	Bajo

Tabla 3. Diferencias entre robots tradicionales y cobots

Apostar por un robot colaborativo supone una mayor versatilidad y potencial de desarrollo, sin embargo, un robot tradicional supone una garantía para las operaciones que requieren mover grandes pesos de forma más rápida que con los cobots.

A continuación, se analizan algunas de las marcas más importantes en el sector de la robótica colaborativa.

Universal Robots:

La empresa Universal Robots fue fundada en 2005. Actualmente, Universal Robots se especializa en la producción de robots colaborativos antropomórficos. Tal es su impacto en el sector de la robótica colaborativa que su volumen de negocios en 2020 fue de 219 millones de USD y, actualmente, son más de 50.000 cobots los vendidos por esta empresa. [18]

En el catálogo de Universal Robots se presentan cinco cobots cuyos nombres hacen referencia a la carga útil que son capaces de mover. De esta manera, las empresas demandantes tienen la posibilidad de elegir entre cinco opciones el brazo robótico colaborativo que mejor se adapte a sus necesidades. [19] La Tabla 4 muestra un resumen de las características más importantes de los robots colaborativos de Universal Robots.

Cobot	Alcance (mm)	Carga útil (kg)	Huella (mm)	Peso (kg)	Aplicaciones
UR3e	500	3	128	11.2	Ensamblajes ligeros Atornillado
UR5e	850	5	149	20.6	Pick & Place Pruebas de producto
UR10e	1300	12.5	190	33.5	Mantenimiento de máquinas Paletizado y Embalaje
UR16e	900	16	190	33.1	Mantenimiento de máquinas Packaging Atornillado
UR20e	1750	20	245	64	Servicio pesado

Tabla 4. Características de los cobots de UR

Cabe destacar que estos cinco cobots tienen una rotación de ejes de 360º en todas sus articulaciones y una rotación infinita en el último eje. De esta manera, el espacio de trabajo que abarcan es el mismo, diferenciándose, por supuesto, en el alcance de su TCP (Tool Central Point).

Se define el espacio de trabajo de un robot como el volumen que comprende las coordenadas de todos los puntos que puede alcanzar el efector final. Este depende de una serie de factores, incluidas las dimensiones del brazo. [20]

En la Ilustración 11 se contempla una representación gráfica de la zona de trabajo del robot UR5e. En esta figura, se puede observar como el espacio de trabajo está definido por una esfera y un cilindro de 850mm y 200 mm de diámetro respectivamente.

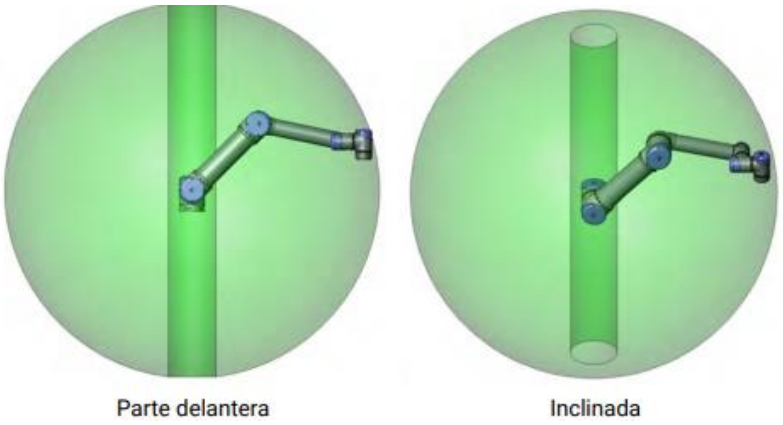


Ilustración 11. Espacio de trabajo del UR5e (Fuente: [21])

Se debe de evitar acercar la herramienta al volumen cilíndrico, ya que esto provocaría que las juntas del robot se movieran rápido cuando la herramienta lo hiciera despacio, lo que causa que el robot trabaje de forma ineficiente y dificulta la realización de la evaluación de riesgos. [21]

ABB:

ABB, acrónimo de Asea Brown Boveri, fue fundada en 1988 y tiene su sede central en Zúrich (Suiza). Con unos ingresos netos de 1,439 millones de USD en 2019, ABB se sitúa como uno de los principales fabricantes de robots industriales del mundo. [22]

Esta empresa comenzó con la fabricación de robots industrial, pero se tuvo que modernizar como muchas otras. Por ello, en 2015 fue lanzado al mercado el robot colaborativo *"YuMI"* que consta de dos brazos articulados de siete grados de libertad, cada uno destinado a realizar tareas de ensamblaje de piezas pequeñas en el sector de la electrónica. En la Ilustración 12 se muestra el modelo de este robot colaborativo.

Ilustración 12. YuMi® - IRB 14000 (Fuente: [23])

Más tarde, ABB presentaría *"YUMI – IRB 14050"* que es igual que el anterior, pero con un solo brazo robótico, *"GoFa"* que destaca por su alcance y carga útil y, por último, *"SWIFTI"* cuya rapidez es la característica más importante.

Los robots de ABB, según sus palabras, están diseñados para aplicaciones de baja carga útil, como la manipulación de piezas pequeñas y tareas de inspección. [23]

KUKA:

La compañía alemana KUKA es otra de los principales fabricantes mundiales de robots industriales. Su origen se remonta al año 1890, cuando prestaba servicios de iluminación pública y doméstica, pero no fue hasta 1973 que su primer robot industrial viese la luz.

Entre los desarrollos más importantes de KUKA se puede encontrar *"FAMULUS"*, primer robot industrial de seis ejes accionados electromecánicamente, o "titan", el robot industrial de 6 ejes más grande y fuerte a nivel mundial.

En cuanto a la robótica colaborativa, KUKA también apuesta por innovar su catálogo con *"LBR iiwa"*. Este cobot incorpora sensores de par integrados en cada eje que permiten interrumpir su movimiento si se topan con la mano del usuario. Además, LBR iiwa está disponible en dos variantes con capacidades de carga de 7 y 14 kg. [24]

El LBR iiwa está especialmente diseñado para tareas de montaje en el sector de la electrónica, aunque también sirve para paletización o preparación de pedidos. En la Ilustración 13 se muestra el diseño de este robot.

Además, KUKA también incorpora en su catálogo otros tipos de cobots como el *"LBR iisy"*, caracterizado por su programación mediante guiado manual y su capacidad para ser integrado en espacios reducidos.

Ilustración 13. LBR iiwa 14 R820 (Fuente: [25])

Como apuesta de futuro, KUKA plantea el prelanzamiento de un nuevo sistema operativo llamado *"iiQKA"* que se implementará en el LBR iisy. De esta manera, la experiencia del usuario será más intuitiva, fácil, potente y rápida. [26]

Entre las diferentes ofertas de mercado que se han analizado, la gama UR es la que tiene un mayor catálogo de robots antropomórficos. Por esta razón, el UR5e es que el que mejor se adapta a las necesidades de este proyecto, ya que su tamaño y alcance es suficiente para cumplir los objetivos que se pretenden.

2.1.2 Antecedentes

A continuación, se expone un análisis de los trabajos previos relacionados con este TFG resaltando las soluciones y las limitaciones que ofrece cada uno. El orden de los trabajos viene dado por la importancia y relación con este proyecto, empezando por los más generales y menos relacionados, y acabando con los más relacionados estrechamente.

Aalto University – "Mirrorlabs - Creating accessible Digital Twins of robotic production environment with Mixed Reality"

El objetivo del proyecto de fabricación de EIT *"Mirrorlabs"* es el desarrollo de una infraestructura de TIC (Tecnologías de la Información y la Comunicación) integrada y fácil de usar para el conjunto de dispositivos de RM disponible en el mercado en combinación con sistemas de producción impulsados por ROS (Robot Operation System) y para ponerlo a disposición de la educación y la investigación. [27]

En el siguiente capítulo se hará más énfasis en la estructura y funcionamiento de ROS. En resumen, se puede definir como un medio de comunicación entre

distintos procesos que utiliza "*topics*" (temas) que sirven para transmitir datos entre estos.

Este proyecto incorpora una serie de aplicaciones que van desde un análisis remoto habilitado por la realidad virtual de conjuntos de datos de líneas de producción operadas por robots, como se muestra en la Ilustración 14, hasta el uso de la realidad aumentada para una estrecha colaboración entre humanos y robots en la planta de producción. [27]

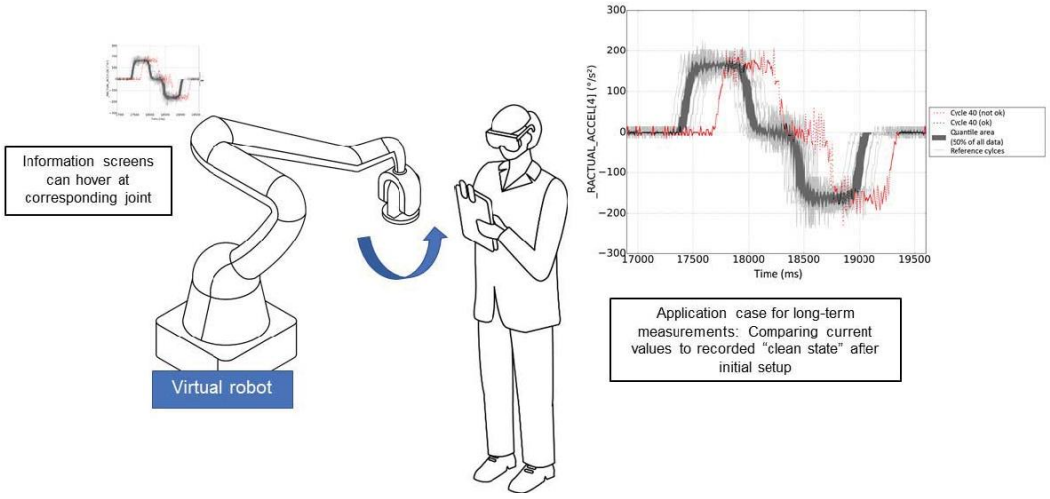


Ilustración 14. Caso de aplicación de realidad virtual para análisis de datos (Fuente: [27]])

Una de las aplicaciones más interesantes que ofrece este proyecto se basa en la visualización de un robot digital que replica el movimiento de un robot real en tiempo real. Este escenario se logra mediante la transmisión de datos por parte del robot a través de la plataforma ROS. Estos datos son almacenados en una base de datos.

Por otro lado, se hace uso de la plataforma Unity para crear el modelo 3D del robot y la programación necesaria para configurar la controladora del robot virtual. La librería Ros-Bridge sirve de puente para establecer la comunicación entre Unity y ROS. De esta manera, se establece el puente de comunicación de datos genérico entre el robot real y el robot virtual creado en Unity 3D.

Por último, Unity incorpora la opción de visualizar la aplicación creada con un dispositivo de realidad mixta como, por ejemplo, Microsoft Hololens 2.

En la Ilustración 15 se muestra el concepto de la arquitectura de Mirrorlabs y en la Ilustración 16 una fotografía tomada con las lentes de HL2 que muestra como el robot virtual replica el movimiento del robot real en tiempo real.

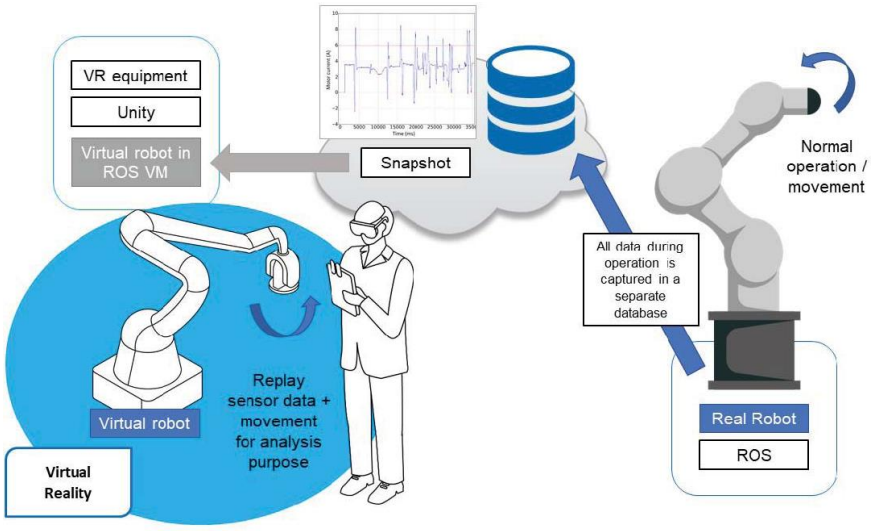


Ilustración 15. Arquitectura de Mirrorlabs para la visualización de datos (Fuente: [27])

Ilustración 16. Visualización del robot virtual de Mirrorlabs (Fuente: [27])

Con esta aplicación se podría ayudar a un operario en la toma de decisiones a la hora de llevar a cabo una maniobra con el robot. Por ejemplo, se podría probar una trayectoria primero en el robot virtual y comprobar si le va a llevar a singularidades o colisiones contra objetos de la célula de trabajo.

No obstante, este trabajo no ofrece ningún tipo de interacción directa con el robot, sino que más bien sirve como herramienta de aprendizaje para simular situaciones reales mediante el uso de la realidad mixta. El usuario solo observa datos del robot, pero no los modifica ni interviene en su movimiento o estado en ningún momento.

Dimitris Mourtzis , John Angelopoulos y Nikos Panopoulos - "Closed-Loop Robotic Arm Manipulation Based on Mixed Reality."

Este proyecto presenta una aplicación de realidad mixta que propone un marco para la navegación remota casi en tiempo real de brazos robóticos. [28]

El método a desarrollar para cumplir este propósito se basa en la creación de un gemelo digital del brazo robótico y la configuración de un marco de comunicación adecuado para la continua comunicación entre la aplicación de realidad mixta integrado en un dispositivo Microsoft Hololens, el robot físico y el gemelo digital. Para ello, se utilizan dos módulos informáticos principales. El primer módulo se encarga de la representación 3D del robot y el segundo de la simulación de este con sus consecuentes cálculos de cinemática.

Para que la representación del movimiento del robot virtual coincida con el del robot real y se realice exitosamente, la controladora del robot ejecuta el programa enviado con las posiciones indicadas por el usuario y envía retroalimentación al backend de la aplicación para confirmar que el movimiento se ha ejecutado con éxito. De esta manera, el usuario puede introducir una nueva trayectoria cuando el robot haya finalizado la anterior.

Cabe destacar que la aplicación de realidad aumentada cuenta con un mecanismo para la visualización de la zona de seguridad del robot. En este escenario, se utilizan objetos geométricos 3D que cubren el volumen del propio robot y su espacio de trabajo.

Este trabajo propone una interacción directa entre el usuario y el robot. Además, la aplicación de realidad mixta incorpora herramientas de visualización 3D que pueden ser de gran utilidad en actividades con un robot colaborativo, por ejemplo, el mapeo 3D del espacio de seguridad del robot.

No obstante, las funcionalidades de la aplicación respecto al movimiento del robot son muy limitadas. Se utilizan controladores virtuales y gestos con la mano para seleccionar puntos del espacio, por lo que, las maniobras del robot quedan reducidas solamente a la reproducción de trayectorias indicadas por el usuario.

En la Ilustración 17 se contempla una fotografía tomada desde las lentes del dispositivo de realidad mixta en la que se puede apreciar un código QR que indica donde se sitúa el sistema de coordenadas global del robot. A partir de este sistema, se establece el espacio de trabajo del robot y la zona de seguridad.

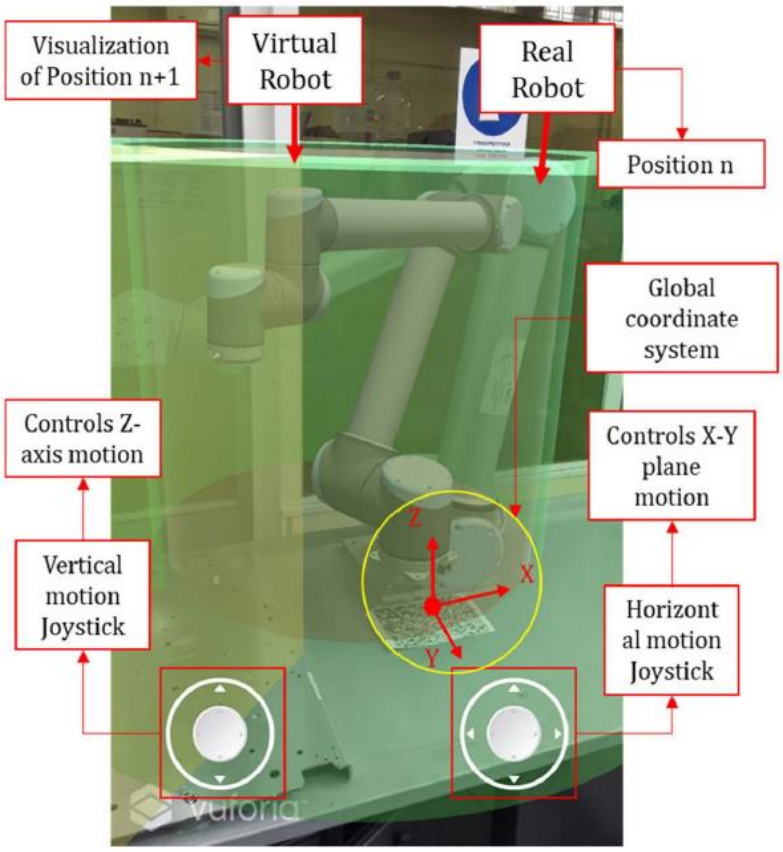


Ilustración 17. Visualización AR de las zonas de seguridad del robot (Fuente: [28])

Inês de Oliveira Soares – "Programming Robots by Demonstration using Augmented Reality".

El sistema de realidad mixta desarrollado en este proyecto permite habilitar a un operador industrial sin experiencia en programación con robots, a programar un robot colaborativo utilizando Microsoft Hololens 2. [29]

Para ello, se utiliza ROS como medio de comunicación entre una aplicación de realidad mixta desarrollada en Unity que permite trazar trayectorias en el espacio mediante gestos con las manos y la controladora del robot.

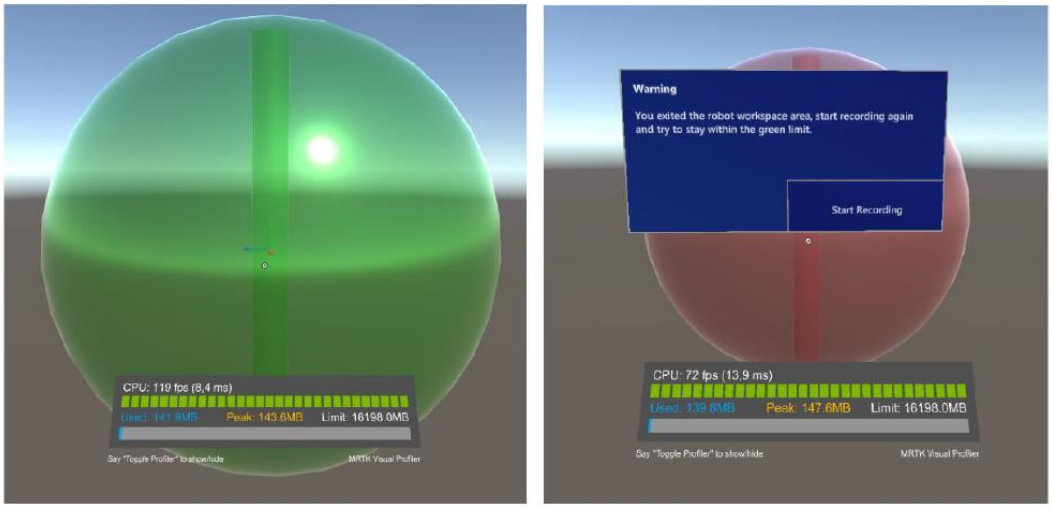
En este caso, la retroalimentación que se realiza desde el robot hasta el usuario solamente es sensorial y, por lo tanto, es la propia persona que está utilizando la aplicación la que debe de estimar las precauciones necesarias para no llevar al robot a singularidades o situaciones de peligro.

En la Ilustración 18 se contempla un esquema que refleja una visión general del sistema de realidad mixta implementado en este proyecto. [29] De nuevo, la plataforma utilizada para el desarrollo del sistema de realidad mixta es Unity, que utiliza el lenguaje C# para crear los programas necesarios para enviar las coordenadas trazadas por el usuario a ROS.

Ilustración 18. Visión general del sistema (Fuente: [29])

La interfaz de la aplicación de realidad mixta hace uso de diálogos digitales que sirven para guiar al usuario por la aplicación. De esta manera, se garantiza que el usuario esté siempre informado sobre la posterior acción que debe de realizar sin cometer errores.

Una de las cosas más interesantes de esta aplicación es que permite definir un sistema de coordenadas virtual sobre el que se proyecta en su origen el espacio de trabajo del robot que evita que el usuario pueda definir trayectorias que sobrepasen sus límites, tal como se aprecia en la Ilustración 19.



(a) Espacio de trabajo original *(b) Espacio de trabajo al sobrepasar los límites*

Ilustración 19. Espacio de trabajo del robot visto desde Unity (Fuente: [29])

Si el usuario indica una coordenada que sale de esa zona, aparece un aviso en forma de diálogo que informa al usuario sobre el deber de permanecer siempre dentro del espacio de trabajo del robot. [29]

El sistema de coordenadas virtual se define mediante dos puntos trazados en el espacio, el primero marca el origen y el segundo la orientación de uno de sus ejes con la condición de que uno de estos debe de apuntar hacia arriba.

El valor de las coordenadas indicadas por el usuario corresponde a la distancia relativa entre el dedo índice con el que marca los puntos y el origen del sistema de coordenadas virtual. De esta manera, la trayectoria enviada al robot corresponde con la del TCP respecto la base del robot.

Con un programa escrito en C++ dentro del espacio de trabajo de ROS, se consigue traducir las coordenadas enviadas por la aplicación de realidad mixta al lenguaje del robot.

El resultado de este trabajo es bastante demostrativo, es decir, se consigue lograr una comunicación efectiva entre el dispositivo de realidad mixta y el cobot haciendo que este último se mueva mediante una orden establecida por el usuario. Con esto se consigue que un operario sin experiencia pueda programar trayectorias en un robot sin necesidad de manejar su Flex Pendant (colgante de enseñanza) o conocer su lenguaje nativo de programación.

No obstante, la actividad que se realiza con esta aplicación no es muy aplicable en muchos ámbitos como el industrial. Esto es debido a que el espacio de trabajo que establece el usuario es igual que el del robot, pero está aislado de él.

En la Ilustración 20 se muestra, en primer lugar, una fotografía tomada desde las lentes de HL2 de una trayectoria trazada en forma de cuadrado dentro de la aplicación y, en segundo lugar, una fotografía tomada al robot colaborativo cuando realiza la maniobra indicada.

(a) Trayectoria indicada en la aplicación de Unity *(b) Robot replicando la trayectoria*

Ilustración 20. Trayectoria indicada por el usuario (Fuente: [29])

En la trayectoria indicada, se envían todos aquellos puntos captados dentro de la aplicación de Unity durante el intervalo de muestreo que se están capturando datos. Por ello, en algunas ocasiones la cantidad de datos

enviados puede ser innecesaria, ya que para que el robot realice una trayectoria similar no necesitará de tanta información.

Esta aplicación permite al operario mover un robot que esté en la misma red que el HL2 desde otra habitación o cualquier lugar dentro del mismo edificio. De esta manera, se proporciona una solución para que el operario pueda acceder remotamente al control del robot.

2.2 Estudio de la problemática

La aplicación real de sistemas de realidad mixta en la práctica es muy compleja, tal como hemos visto en los trabajos anteriores. Son muchos los factores a tener en cuenta a la hora de diseñar un protocolo de comunicaciones que garantice la interacción humano-robot segura y correcta a través de la realidad mixta.

En los estudios revisados se ha comprobado que el rango de maniobras que se pueden realizar con el robot es muy limitado, ya que en estos trabajos el resultado obtenido solo demostraba la capacidad de movimiento del robot a través de órdenes indicadas por el usuario.

Ante esta problemática, el operario es capaz de programar un robot mediante realidad mixta, pero no es capaz de llevar a cabo maniobras típicas como el Pick & Place. Además, la forma de ejecutar estas instrucciones es manual en estos casos, por lo tanto, el usuario debe mantener las manos libres a la hora de utilizar la aplicación de realidad mixta. Sin embargo, con la introducción de comandos de voz se podría hacer que el operario manejase el robot mientras sostiene documentación o un dispositivo informático en sus manos.

La precisión con la que se replica el movimiento del robot también es un tema importante. En *"Mirrorlabs"* se visualiza tridimensionalmente y en tiempo real el movimiento del robot con absoluta precisión, pero esta comunicación es unidireccional. De esta manera, el operario no realiza ningún tipo de interacción directa con el robot.

En *"Closed-Loop Robotic Arm Manipulation Based on Mixed Reality"* la comunicación es bidireccional, pero el movimiento del robot se realiza mediante un joystick virtual que es muy sensible a la interacción manual del usuario y no garantiza lo totalidad de la precisión que requiere una aplicación en el entorno robótico colaborativo.

En *"Programming Robots by Demonstration using Augmented Reality"* el operario traza una trayectoria en el espacio que el robot colaborativo replica, pero sin saber la pose de la referencia del robot. Por lo tanto, si el operario quiere realizar una trayectoria que sitúe el efector final del robot por encima

de algún tipo de objeto, no se garantiza que no vaya a ver una colisión con este.

Además, a la hora de trazar la trayectoria se recogen puntos con un periodo de muestro bastante alto y, debido a que la capacidad del humano de mantener la tensión en la mano a la hora de dibujar trayectorias en el espacio nunca es la ideal, hace que el movimiento del robot sea muy brusco.

Otro factor a tener en cuenta es que la experiencia de programación sea satisfactoria y no suponga riesgos para el operario. El campo de visión de los dispositivos de realidad mixta no es muy extenso y esto supone que el operario pueda no prestar la suficiente atención a la información que muestra la aplicación. Por ello, en la práctica sería recomendable hacer uso de otros elementos que permitiesen obtener información de lo que está pasando tanto en la aplicación como alrededor del operario.

2.3 Conclusiones

En este capítulo se han introducido algunos de los conceptos básicos y necesarios para entender el resto del documento de forma correcta. Se ha hecho especial mención a los conceptos de robótica colaborativa y realidad mixta exponiendo sus rasgos más generales, sus tipos y las empresas más relevantes en el mercado que fabrican este tipo de tecnologías.

Las diferencias entre los tipos de realidad mixta se han expuesto de manera que, ahora en adelante, cuando se mencione una de ellas se debe de tener en cuenta que se hace con intención de especificar su característica principal:

- Cuando se habla de realidad virtual, el usuario está inmerso en un mundo totalmente digital.
- Cuando se habla de realidad aumentada, el usuario observa información digital superpuesta a la realidad.
- Cuando se habla de realidad mixta, la información digital que observa el usuario tiene conciencia espacial, es decir, va a poder interactuar con el entorno.

En cuanto a la robótica colaborativa, cabe destacar que la mayor diferencia de esta respecto de la robótica industrial es que la primera facilita la cooperación del robot con un humano en la misma célula de trabajo, mientras que la última no.

También se han analizado algunos de los trabajos más relevantes a la hora de desarrollar la aplicación de este TFG profundizando especialmente en el trabajo de Inés de Oliveira Soares, que se toma como punto de partida en la arquitectura de este proyecto.

Por último, se han expuesto las problemáticas y limitaciones existentes a la hora de diseñar y desarrollar una aplicación de realidad mixta orientada a la robótica colaborativa, señalando que una aplicación que integre estas dos tecnologías debe de ser intuitiva, segura, precisa y que permita la interacción directa del operario con el cobot.

Capítulo 3. Desarrollo del sistema de realidad mixta

Los aspectos de diseño e implementación de este trabajo se exponen en este capítulo de forma ordenada comenzando por las herramientas utilizadas en el apartado 3.1.

Posteriormente, en el apartado 3.2 se enumeran los objetivos propuestos describiendo los métodos y líneas de trabajo tomados en el marco práctico del proyecto y se justifican las alternativas barajadas para el desarrollo de este.

El diseño de la aplicación de realidad mixta se describe en el apartado 3.3 y su implementación en el sistema se expone en el apartado 3.4.

Por último, en el apartado 3.5 se extraen las conclusiones generales de este capítulo.

3.1 Componentes del sistema

Este apartado se centra en presentar las tecnologías utilizadas en este proyecto, resaltando las funcionalidades de cada una y justificando los motivos de su elección frente a otras opciones de la misma índole.

En primer lugar, se presenta el soporte físico utilizado mencionando los dispositivos de realidad mixta y robots colaborativos elegidos.

Posteriormente, se clasifica el soporte lógico que integra este proyecto en la categoría que le corresponde a cada plataforma.

Por último, se exponen los requisitos y mecanismos de comunicación que establecen la conexión y el paso de mensajes entre los dispositivos completando la idea general de la arquitectura del sistema.

3.1.1 Soporte físico

El soporte físico de este proyecto se compone por el dispositivo de realidad mixta que proporciona la interfaz digital, el cobot y un PC donde se gestiona la comunicación entre ambos.

Adicionalmente, se cuenta con una garra impresa en 3D colocada en el efector final del robot que es accionada mediante un Arduino Uno y una tira de luces LED RGB conectadas directamente a un controlador DMX512 que es gestionado por un gateway IOT2050 de Siemens.

Microsoft Hololens 2

El dispositivo de realidad mixta elegido para guiar al operario en la actividad con el robot colaborativo es Microsoft Hololens 2. Este dispositivo realiza el seguimiento manual, ocular y auditivo de la persona que lo utiliza.

HL2 cuenta con un sensor IMU que realiza los cálculos inerciales para saber la posición y orientación de la cabeza y un sensor TOF que mapea el entorno, como bien se explica en el capítulo 2.

La aplicación de realidad mixta a desarrollar para operar con el cobot estará integrada en HL2, controlándola mediante gestos, la mirada o la voz.

UR5e

Como se adelanta en el capítulo 2, el UR5e es el robot colaborativo de tamaño medio de la gama Universal Robots. Soporta una carga útil de cinco kilogramos y tiene un alcance de 850 milímetros. No obstante, el manual de usuario del UR5e advierte que hay determinadas zonas del espacio de trabajo que requieren atención en relación con los peligros de pinzamiento: *"Se define una zona para movimientos radiales cuando la junta de la muñeca 1 al menos a 750 mm de la base del robot"*. [21]

En la Ilustración 21 se muestra un diagrama del espacio de trabajo del UR5e con las cotas correspondientes a sus ejes.

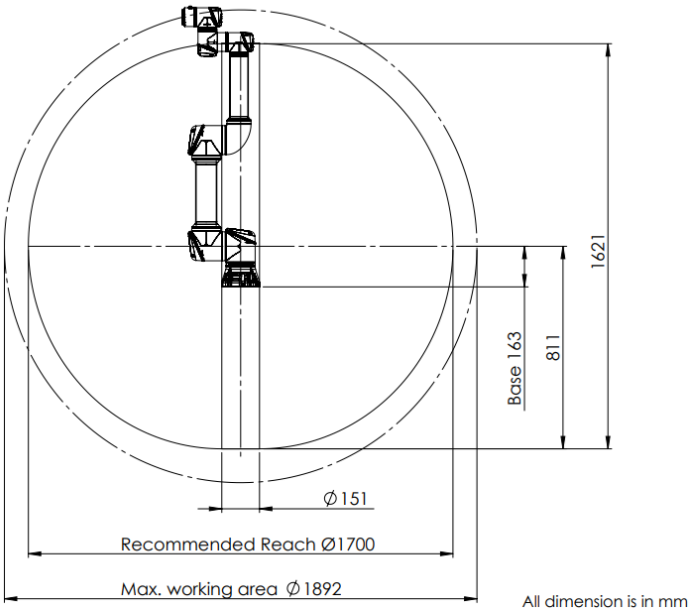


Ilustración 21. Diagrama del espacio de trabajo del UR5e (Fuente: [19])

Aunque la célula también cuenta con un UR10e a su disposición, se ha elegido trabajar con el UR5e debido a la sencillez y seguridad de trabajar con un robot colaborativo más ligero.

En la Ilustración 22 se muestra una fotografía de la célula de trabajo donde se encuentran los robots colaborativos UR5e y UR10e.

Ilustración 22. Disposición de los cobots en la célula de trabajo

PC

El PC es el soporte físico por el que se transmiten los datos entre los diferentes dispositivos. Este equipo debe poder cumplir los requisitos necesarios para ejecutar los programas de desarrollo.

Las especificaciones técnicas más importantes del PC del que se hace uso en este proyecto son las siguientes:

- Modelo: HP Z1 G8 Tower Desktop PC
- S.O: Windows 10 Pro 64 bits
- Procesador: 11th Gen Intel(R) Core (TM) i7-11700 @ 2.50GHz
- Memoria: 128.00 GB
- Tarjeta Gráfica: NVIDIA GeForce RTX 3070

Tira de luces LED RGB 5050 SMD

La mesa de trabajo de los robots colaborativos incorpora cuatro tiras flexibles de luces LED RGB 5050 SMD, cubriendo su perímetro. Estas tiras solo admiten DC5V y son direccionables individualmente. De esta manera, cada píxel puede tener su propio color y brillo, tal como se aprecia en la Ilustración 23.

En su origen, se quiso acompañar la actividad humano-cobot mediante un sistema de iluminación que proporcionase información al operario acerca del estado en el que se encuentra la tarea en cada momento. Entre las diferentes opciones que se tomaron en cuenta, las luces de tipo LED RGB eran las que más posibilidades ofrecían de cara al desarrollo de una actividad con un robot colaborativo.

Ilustración 23. Ejemplo de luces LED RGB 5050 SMD encendidas (Fuente: [30])

Uno de los propósitos era advertir al usuario que el robot se está moviendo, estableciendo el color de las luces a rojo y, cuando este finalizase su movimiento, cambiarlo al color verde. Estas tiras van físicamente conectadas a un controlador DMX que se encarga de distribuir las salidas de los colores a cada una.

Controlador LED RGB de 24 Canales DMX 512 Decoder

Un controlador DMX es un dispositivo que implementa el protocolo DMX 512 (Digital Multiple X) para manejar efectos de luminotecnia. Fue diseñado en 1986 como solución para poder sincronizar y programar efectos de iluminación de diferentes marcas, que sí tenían su propio protocolo de control. [31]

En la Ilustración 24 se muestra un ejemplo del modelo DMX512 utilizado. En su cubierta se indica que los 32 pines situados en la parte superior e inferior sirven para conectar los canales RGB, mientras que los seis pines de uno de sus laterales sirven para conectar la alimentación del dispositivo.

Ilustración 24. Modelo DMX512 de 24 canales (Fuente: [30])

El protocolo DMX contiene 512 canales con valores comprendidos en cada canal entre 0 y 255. Cada canal puede ser regulado. A su vez el circuito que controla esos 512 canales se le llama Universo DMX, cada universo o circuito DMX contiene un máximo teórico de 512 canales. [31]

El controlador DMX utilizado en este proyecto consta de 24 canales, 12 de los cuales son utilizados para la conexión física con la tira de luces RGB. Cada tira de led debe de estar conectado al pin de alimentación.

SIMATIC IOT2050 Advanced (6ES7647-0BA00-1YA2)

El IOT 2050 es un dispositivo electrónico que sirve de puerta de enlace (gateway) inteligente para establecer una comunicación eficaz entre los dispositivos inteligentes y las unidades de control.

SIMATIC IOT2050 puede ejercer como interfaz entre el campo y las TI o la nube. De esta manera, se puede armonizar, analizar y reenviar a los correspondientes destinos la comunicación entre diversas fuentes de datos. [32]

Este dispositivo implementa varios protocolos de comunicación que hacen posible la compatibilidad entre dispositivos basados en la arquitectura TCP/IP y un PLC que utiliza Profinet. Además, incorpora una serie de puertos como el USB, LAN o COM, [32] tal como se aprecia en la Ilustración 25.

El aprovechamiento de este dispositivo abre numerosas posibilidades de aplicación como la conexión y armonización de las comunicaciones de diferentes máquinas y componentes de automatización. [32] De ahí, la necesidad de utilizar este dispositivo como pasarela entre los dispositivos conectados a Internet como el PC y el DMX.

Ilustración 25. Gateway IoT Siemens IOT2050 (Fuente: [32])

Cuando este dispositivo esté encendido, automáticamente ejecutará un servidor que esperará las peticiones de los clientes que quieren modificar el color de las luces.

Herramienta y Arduino UNO

Se ha diseñado una herramienta en una impresora 3D que sirve para agarrar objetos y que está acoplado a la muñeca del UR5e. Esta garra cuenta con un cuerpo donde va anclado un sistema de engranajes que mueve los dos extremos, tal como se aprecia en el diseño 3D de la garra de la Ilustración 26.

El motor que se encarga de mover esta garra es un motor básico de Arduino, un servomotor MG996R. El movimiento que realiza no excede nunca los 180º.

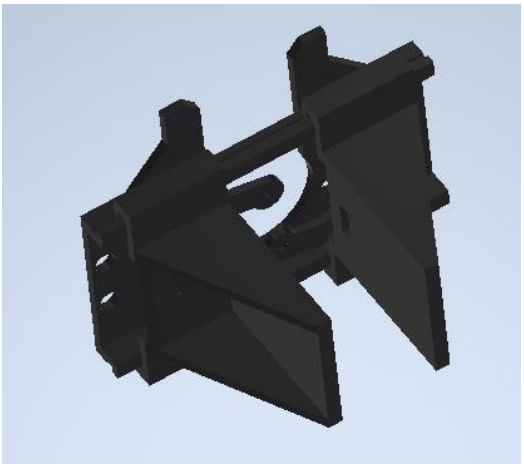
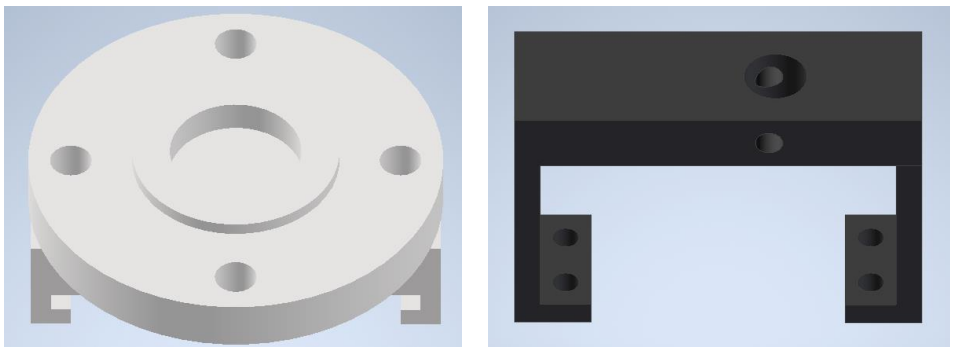


Ilustración 26. Diseño 3D de la herramienta del UR5e

Este mecanismo implementa un sistema de guías lineales y es anclado a la muñeca del robot colaborativo mediante un soporte dividido en dos piezas, una para el acople con la muñeca del robot y otra para la unión de este anclaje con el cuerpo de la garra. En la Ilustración 27 se puede apreciar como este soporte contiene una serie de taladros que sirven para fijar la garra.



a) Anclaje con la muñeca *b) Unión entre anclaje y garra*

Ilustración 27. Soporte 3D de la herramienta del UR5e

El Arduino UNO usa un Módulo Ethernet ENC28J60 que permite la transmisión de datos vía Ethernet con otros dispositivos. De esta manera, el Arduino sirve de servidor del UR5e para cuando éste mande la señal de abrir o cerrar la garra. La Ilustración 28 muestra el aspecto del Módulo Ethernet que se incluye al Arduino UNO.

Ilustración 28. Arduino UNO y Módulo Ethernet ENC28J60 (Fuente: [33])

La primera alternativa considerada fue la de realizar el conexionado entre el Arduino y el armario del robot directamente. De esta manera, cuando el robot activase una salida, la señal se enviaría directamente al Arduino par accionar la herramienta.

No obstante, las señales digitales de salida del armario del robot salen a 24 V y eso es mucho mayor que lo que necesitan los pines digitales del Arduino. Si se conectasen, cuando se activara la salida digital del robot, se correría riesgo de que el Arduino se quemase.

Como solución, se podría utilizar un convertir DC-DC para pasar de 34V a 3.3V, pero la opción de utilizar sockets TCP/IP es más profesional y segura para el sistema.

3.1.2 Soporte lógico

El equipo informático comentado en el punto anterior debe de tener integrado una serie de programas informáticos que permitan el control de la comunicación entre los dispositivos y lleve a cabo la transmisión de los datos sin pérdidas ni errores.

ROS (Robot Operating System)

La página oficial de ROS lo define como conjunto de bibliotecas de programas informáticos y herramientas que ayudan a crear aplicaciones de robot. Desde controladores hasta algoritmos de última generación y potentes herramientas para desarrolladores. [34]

ROS se sitúa en una capa superior al sistema operativo y es compatible con Windows, Linux y macOS. Además, es de código abierto por lo que existe una gran comunidad de usuarios que publican sus propios paquetes y programas en repositorios digitales como GitHub. Esto permite que la inmensa mayoría de la comunidad científica trabaje usando un soporte lógico común.

La posición del nivel de ROS en la pila de arquitectura de un equipo viene representada en la Ilustración 29.

Ilustración 29. Niveles de abstracción en una pila de componentes con ROS (Fuente: [34])

Entre las características más importantes de ROS se encuentran las siguientes:

- Abstracción del soporte físico.
- Control de dispositivos de bajo nivel.
- Procesos distribuidos.
- Soporte a varios lenguajes de programación.
- Diversos mecanismos de comunicación entre procesos.

El mecanismo de comunicación de mayor relevancia en ROS es el denominado *"paso de mensajes"* o *"paso de topics"* entre nodos. Un nodo simplemente es un proceso que realiza cálculos y se combina en un gráfico con otros nodos para la transmisión de topics.

Para entender muchos de los conceptos relacionados con ROS es preciso definir algunos de estos:

- **Topics:** Los temas son buses con nombre sobre los cuales los nodos intercambian mensajes. Los temas están destinados a la comunicación de transmisión unidireccional. [34]
- **Paquete:** Un paquete es la unidad de organización de soporte lógico del código ROS. [34] En otras palabras, es una agrupación de nodos.

En el paso de mensajes existen dos tipos de nodos: el publicador y el suscriptor. Por un lado, el publicador es quien transmite la información mediante topics al nodo maestro. Por otro lado, el suscriptor es el nodo que recibe los mensajes que se publican y que realiza acciones sobre estos.

El nodo maestro es el que permite a los demás nodos en ROS comunicarse entre ellos. Este nodo registra todos los nodos, servicios y topics que se encuentran en ejecución. [35]

En la Ilustración 30 se contempla un esquema que ilustra como el nodo publicador transmite un mensaje en el topic para que lo reciba el nodo suscriptor.

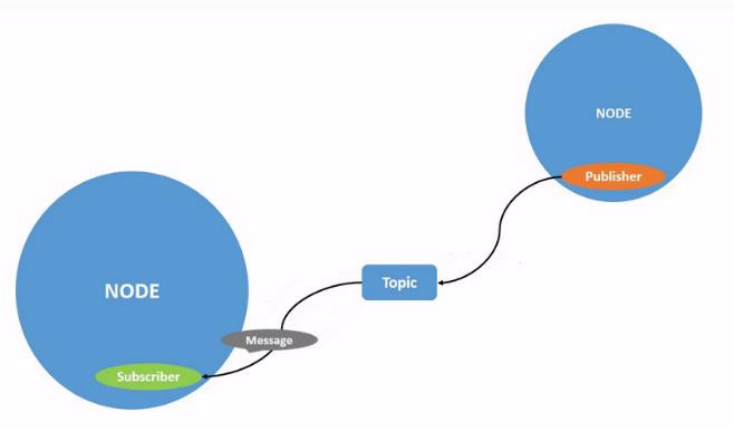


Ilustración 30. Diagrama del paso de mensajes en ROS Fuente: [34])

El papel de ROS en este proyecto es el de servir de núcleo de la comunicación entre HL2 y el UR5e. Se utiliza el paso de mensajes para interpretar los datos recibidos por parte de HL2 y traducirlos al lenguaje nativo del robot antes de enviarlo a la controladora. Esta traducción de los topics se logra mediante un programa escrito en C++ contenido en un paquete creado en el entorno de ROS.

El programa de C++ que traduce y transmite los datos será un nodo suscriptor de la comunicación, ya que recibe la información que envía HL2 al servidor de ROS. Este programa genera el URScript necesario para que el UR5e pueda compilar y ejecutar las instrucciones que se le manden.

URScript es el lenguaje nativo del robot que incorpora las funciones y comandos necesarios para poder realizar acciones con robots de la gama Universal Robots.

Fue considerada la opción de enviar directamente los datos desde HL2 al UR5e, pero ROS ofrece un abanico de posibilidades más amplio a la hora de programar con otros lenguajes de programación.

La distribución de ROS que se utiliza en este proyecto es Melodic Morenia, lanzada en 2018.

Docker Desktop para Windows

Según Wikipedia, Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. [36]

Docker utiliza características de aislamiento de recursos del kernel Linux, tales como *"cgroups"* y espacios de nombres (namespaces) para permitir que *"contenedores"* independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.2 [36]

En este proyecto, se utiliza Docker Desktop para Windows que requiere la instalación del subsistema de Windows para Linux 2 (WSL2). De esta manera, se puede utilizar herramientas y aplicaciones Linux directamente desde el entorno Windows habitual. [37]

Entre las ventajas de Docker, se encuentran las siguientes: [38]

- **100 % Portable.** Docker puede ejecutarse sin problema en cualquier entorno.
- **Encapsulación del entorno de trabajo,** de manera que los desarrolladores pueden estar trabajando en su servidor local con la seguridad de que, al llegar el momento de poner la aplicación en producción, se va a ejecutar con la misma configuración que se ha establecido durante las pruebas realizadas.
- **Ligereza.** Al no virtualizar un sistema completo, el consumo de recursos es mínimo, ahorrando alrededor de un 80% de dichos recursos.

Antes de continuar, merece la pena definir varios conceptos sobre Docker para comprender mejor su funcionamiento.

- **Contenedor:** Los contenedores son procesos que corren sobre la máquina. La gran ventaja de los contenedores respecto de las máquinas virtuales es que ocupan muchos menos recursos sobre el equipo a la hora de ejecutar una aplicación.

  En la Ilustración 31 se contempla como la virtualización de las máquinas virtuales afecta a todas las capas hasta el bloque físico, mientras que en los contenedores afecta solamente a las capas lógicas por encima del sistema operativo.

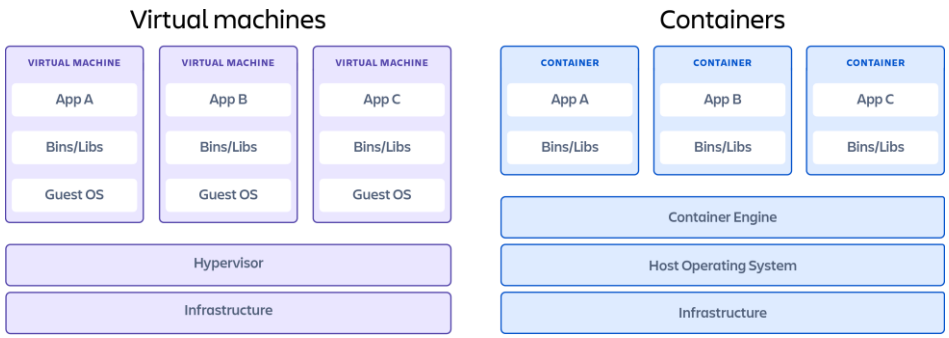


Ilustración 31. Contenedores vs máquinas virtuales (Fuente: [39])

- **Imagen:** La imagen es un paquete que contiene toda la configuración necesaria para que una aplicación se pueda ejecutar. En otras palabras, una imagen de Docker representa el diseño y los planos para construir el contenedor.
- **Dockerfile:** es un documento de texto que contiene todos los comandos que un usuario podría llamar en la línea de comandos para ensamblar una imagen. Utiliza notación específica de Docker.

En este proyecto, se utiliza la imagen de ROS Melodic para crear un contenedor que permita utilizar aplicaciones de una forma liviana, ya que solo se instalarán los paquetes y dependencias que se necesiten. El sistema operativo utilizado en la imagen de ROS Melodic es Ubuntu 18.04.6 LTS.

Unity Hub

Unity es el motor gráfico utilizado para diseñar y desarrollar la aplicación de realidad mixta que se integra HL2.

La configuración y la introducción de las herramientas necesarias para el desarrollo de aplicaciones de realidad mixta para HL2 requiere Unity Hub con Unity 2020.3 LTS o Unity 2019.4 LTS instalado. La versión de Unity utilizada en este proyecto es la 2020.3.31f1 (64-bit).

Unity es lo que se conoce como un motor de desarrollo o motor de juegos. El término motor de videojuego o *"game engine"* hace referencia a un programa informático que tiene una serie de rutinas de programación que permiten el diseño, la creación y el funcionamiento de un entorno interactivo, es decir, de un videojuego.

Otra plataforma de desarrollo gráfico importante es Unreal que junto a Unity incorpora la opción de trabajar con Mixed Reality Toolkit (MRTK). Este kit de herramientas incorpora una colección de scripts y componentes destinados a acelerar el desarrollo de aplicaciones de realidad mixta.

La decisión de elegir Unity frente Unreal fue tomada básicamente durante el proceso de búsqueda de información acerca del desarrollo de realidad mixta en la red. La inmensa mayoría de la comunidad publica sus artículos y programas desarrollados en Unity, por lo que el proceso de aprendizaje con esta plataforma es más fácil y rápido.

El lenguaje de programación con el que se trabaja en Unity es C#. No obstante, no toda la programación se basa en código escrito, la propia interfaz de Unity incorpora funcionalidades muy intuitivas para programar instrucciones mediante el uso de los *"GameObject"*. Cada objeto en el programa es considerado un GameObject, desde un cubo hasta el propio escenario.

Cuando ya se ha terminado la aplicación y se quiere integrar en HL2, se construye la solución del proyecto y se hace uso de Visual Studio 2019 para exportar la aplicación al dispositivo de realidad mixta.

Visual Studio 2019

Un entorno de desarrollo integrado (IDE) es un programa con numerosas características que respalda muchos aspectos del desarrollo de aplicaciones. El IDE de Visual Studio es un panel de inicio creativo que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación. [40]

Una vez completado el proceso de construcción de la aplicación en Unity, se generan una serie de archivos de solución para depurar e implementar en el dispositivo.

3.1.3 Conexionado y mecanismos de comunicación

En este apartado, se expone el proceso de conexionado realizado entre los componentes físicos del sistema haciendo referencia a los mecanismos de comunicación utilizados para realizar la transmisión de los datos entre ellos.

3.1.3.1 Conexionado del soporte físico

Los dispositivos del sistema requieren intercambiar información y compartir recursos para ejecutar las actividades. Por ello, conviene describir como es la arquitectura de alto nivel del sistema, haciendo referencia al tipo de configuración de línea entre equipos y al protocolo de comunicación que emplea cada uno.

En la Ilustración 32 se muestra un diagrama de bloques del conexionado de los equipos informáticos donde los bloques representan los elementos físicos del sistema y las líneas hacen referencia al cableado de conexión de red.

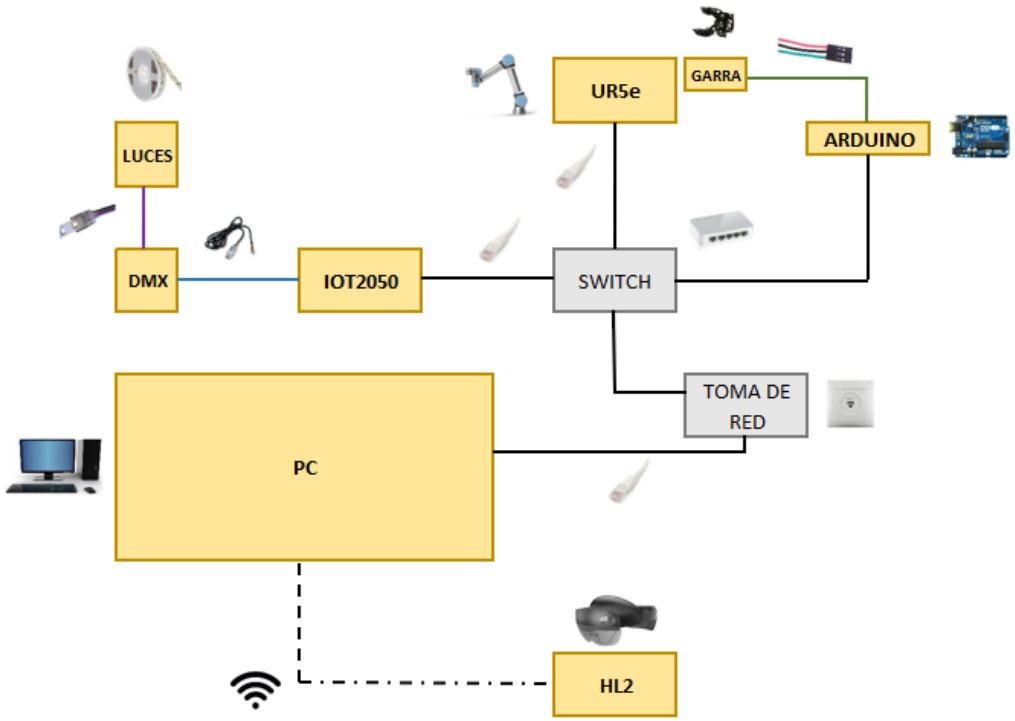


Ilustración 32. Diagrama de bloques del conexionado.

Como se aprecia en la figura anterior, la arquitectura del sistema emplea un switch como intermediario entre las conexiones del robot, el Arduino, el IOT2050 y la toma de red. Esto es debido a que la toma de red solo posee dos puertos RJ45 y a que la gestión del cableado sea más sencilla y ordenada.

El propio switch está conectado a la red directamente mediante un cable Ethernet RJ45 y retransmite los paquetes de datos mediante el protocolo TCP/IP.

Todo el soporte físico del sistema está conectado a la misma red y, esto incluye a los siguientes elementos:

- **Microsoft Hololens 2.** Conexión a la red mediante Wi-Fi.
- **PC.** Conexión directa a la red mediante cable Ethernet RJ45.
- **UR5e.** Conexión al switch mediante cable Ethernet RJ45.
- **Arduino UNO.** Conexión al switch mediante cable Ethernet RJ45.
- **IOT2050.** Conexión al switch mediante cable Ethernet RJ45.

Estos elementos admiten conexión a la red mediante cable Ethernet RJ45, sin embargo, el DMX512 necesita un adaptador USB/RS485 para conectarse con el IOT2050.

Por otro lado, la garra y la tira de luces LED son alimentados mediante conectores. La garra utiliza un conector de tres cables para consumir la energía procedente del Arduino y la tira de luces LED un conector de hipopótamo de tres cables.

3.1.3.2 Mecanismos de comunicación

La comunicación entre los diferentes dispositivos que integran este proyecto está diseñada bajo el protocolo de comunicación TCP/IP, exceptuando el protocolo DMX 512 que utiliza el decodificador.

El protocolo TCP/IP es orientado a la conexión y garantiza la entrega de los paquetes de forma ordenada y sin errores. Todos los dispositivos con conexión a la red mediante Ethernet o Wi-Fi utilizaran este protocolo y se comunican entre ellos vía socket.

Los sockets permiten simplificar laboriosamente la forma de comunicar procesos que corren en equipos distintos (remotos) o, incluso en el mismo ordenador (local). En el dominio de Internet, los sockets permiten acceder a los servicios del nivel de transporte tanto a través de conexiones (TCP) como en modo sin conexión (UDP). Para que esto ocurra deben existir dos nodos: cliente y servidor.

El cliente es el encargado de iniciar la conversación que se efectúa siempre que el servidor se esté ejecutando. Una vez se establece la comunicación, el envío y recepción de mensajes por parte del cliente y servidor puede ser bidireccional.

Por otro lado, el protocolo DMX 512 se basa en la utilización de canales para transmitir órdenes de control a los aparatos que lo soporten. El controlador envía cadenas de datos de 512 valores, un valor por cada canal. A su vez, cada canal trabaja en valores de 8 bits, es decir, cada dato de la señal se puede regular desde el valor 0 hasta el valor 255. [41]

El IOT2050 debe de transformar los mensajes que recibe mediante la línea Ethernet en señales que pueda interpretar el controlador DMX para cambiar el color de la tira de luces LED según ordene el usuario.

En la Ilustración 33 se adjunta el diagrama de bloques mostrado anteriormente, pero se añaden los protocolos de comunicación empleados para permitir la transmisión de los datos entre los dispositivos del sistema. En este diagrama se puede contemplar como la comunicación entre Microsoft Hololens 2 y el PC se hace en realidad con el contenedor de ROS creado dentro de Docker.

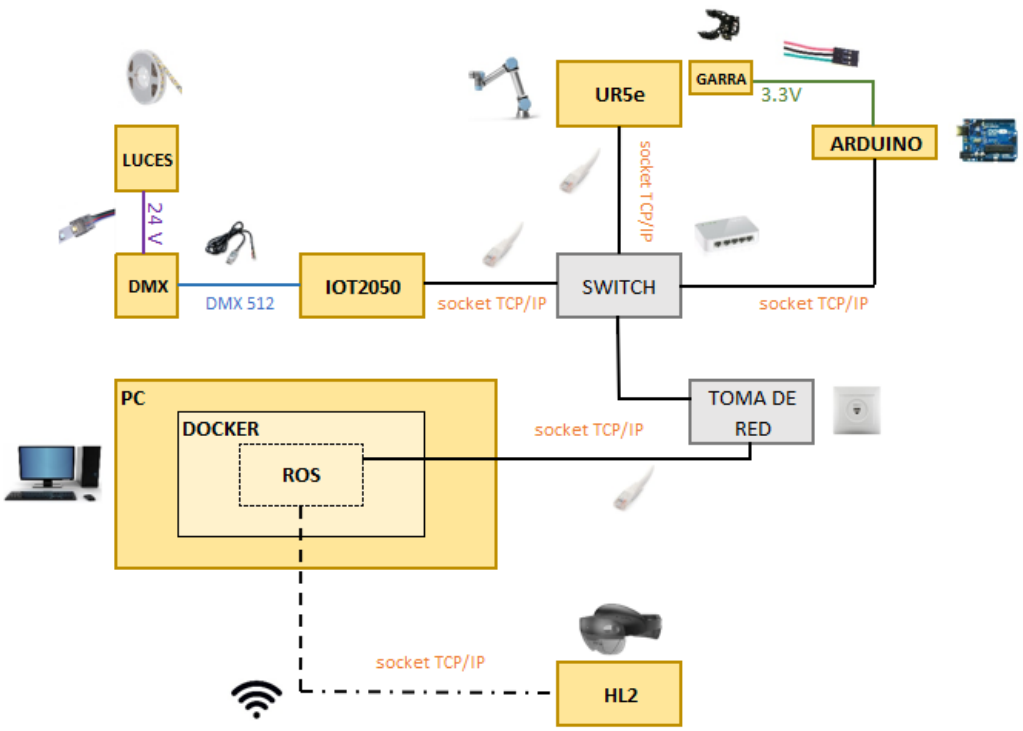


Ilustración 33. Diagrama de bloques del conexionado con los protocolos de comunicación empleados

Esto es debido a que el contenedor es una máquina virtual que, aunque se diferencia bastante en el grado de consumo de recursos con las máquinas virtuales tradicionales, también emula a un ordenador completo. En otras palabras, es un programa informático que puede hacerse pasar por otro dispositivo (como un PC) de tal modo que puedes ejecutar otro sistema operativo en su interior. [42]

En el contenedor de Docker se definen la dirección IP y los puertos que se quieren exponer a la red. De esta manera, para comunicar HL2 con Docker basta con ejecutar un servidor en el contenedor creado con la imagen de ROS que se ejecute en el puerto mapeado al host local. El contenedor de Docker también necesita tener mapeado otro puerto para enviar al robot el script de datos traducidos en ROS.

En la Tabla 5 se clasifica el modelo de comunicación de este proyecto, indicando los dispositivos que actúan como cliente y servidor del sistema, y haciendo referencia al propósito de cada uno a la hora de transmitir los datos desde HL2 hasta el robot.

Dispositivo	C/S	Tarea
HL2	Cliente	Publicar datos mediante topics (Publicador)
Contenedor de Docker	Cliente y Servidor (*)	Suscribirse a los datos publicados por HL2 (Suscriptor), traducirlos y enviárselos al UR5e
UR5e	Cliente y Servidor (*)	Recibir el script con los datos traducidos
Arduino	Servidor	Recibir los mensajes del UR5e para accionar la herramienta
IOT2050	Servidor	Recibir los mensajes del UR5e para transmitir las señales al DMX 512

Tabla 5. Modelo de comunicación TCP/IP del sistema

(*) Conviene aclarar una serie de puntos para comprender como se realiza la comunicación entre procesos en este sistema y la razón de porque algunos dispositivos son cliente y servidor a la vez.

La razón de que el contenedor de Docker sea cliente y servidor de la comunicación es que, por un puerto es servidor de la comunicación con HL2 y, por otro puerto es cliente de la comunicación con el UR5e. En el contenedor corren dos hilos: uno sirve de puente para la publicación de los topics por parte de HL2 y, otro para la suscripción de los topics y el envío del URScript al UR5e.

Algo similar ocurre con el propio robot colaborativo. Este permanece a la escucha de clientes mediante la activación del *"modo remoto"* por un puerto y, por otros dos puertos es capaz de enviar las señales correspondientes al IOT2050 y el Arduino.

- El contenedor de Docker tiene mapeado dos puertos con el host: el **9090** para la recepción de datos por parte del HL2 y el **30002** para el envío del URScript al robot colaborativo.
- El UR5e utiliza el puerto 30002 para recibir el URScript desde ROS.
- El Arduino permanece a la escucha de clientes por el puerto **21** para recibir las señales de accionamiento de la herramienta.
- El IOT2005 permanece a la escucha de clientes por el puerto **10034** para recibir las señales que debe de traducir al DMX 512 por parte del UR5e.

Los puertos empleados en la comunicación mediante sockets se adjuntan en el diagrama de bloques de la Ilustración 34.

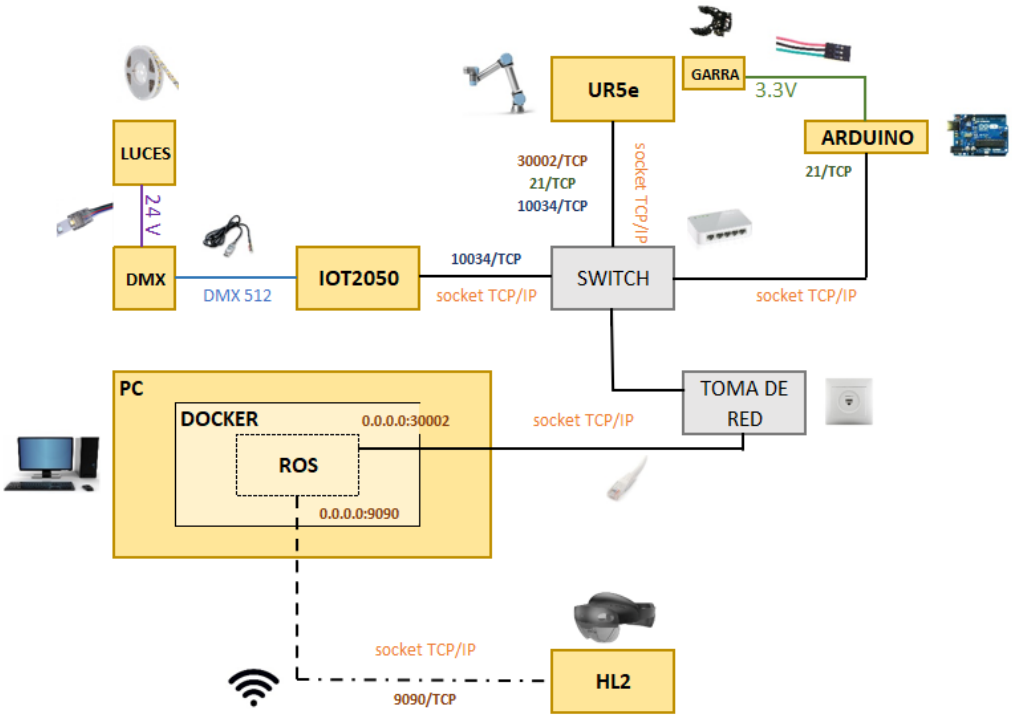


Ilustración 34. Diagrama de bloques del conexionado con los protocolos de comunicación empleados con los puertos empleados

Un ejemplo del proceso de la transmisión que se realiza entre los dispositivos del sistema para que el robot se mueva a una posición y se abra o cierre la garra seguiría los siguientes pasos:

1. El contenedor de ROS abre el puente para establecer la comunicación con HL2 por el puerto 9090 y se suscribe a los topics que se publiquen desde la aplicación de realidad mixta. El UR5e se habilita en modo remoto para permanecer a la escucha de clientes por el puerto 30002. Tanto el Arduino como el IOT2050 siempre que están encendidos ejecutan un programa de servidor en el puerto 21 y 10034 respectivamente.
2. El usuario que lleva puesto el dispositivo HL2 inicia la aplicación de realidad mixta. En este momento, se establece la comunicación entre HL2 y el puente de ROS.
3. El usuario envía la señal de que quiere que el robot se mueva a una posición dada y accione la garra. Se publica un mensaje en el topic correspondiente.

4. El hilo suscriptor reacciona ante el topic publicado y se encarga de enviar el URScript con las instrucciones ordenadas por el usuario al UR5e por el puerto 30002.
5. El robot compila el script recibido y empieza a ejecutar las órdenes indicadas. Entre estas órdenes viene incluida la instrucción de abrir un socket por el puerto 10034 para establecer la conexión con el IOT2050. Antes de que el robot comience su movimiento, se envía la señal de cambio de color hacia el IOT2005, este la traduce y envía la señal correspondiente al DMX512 para que cambie el color de las luces.
6. Cuando se han cambiado las luces a color rojo (indicación de que el robot se está moviendo), se cierra este socket y el robot comienza a moverse hasta alcanzar la posición deseada.
7. En ese momento, se abre un socket en el puerto 21 y se establece la conexión con el Arduino. El UR5e envía un mensaje al Arduino que declare la intención de abrir la garra. El Arduino traduce esta señal y acciona la herramienta por una de sus salidas digitales.
8. La garra se abre y se cierra la conexión entre el Arduino y el robot colaborativo. Posteriormente, se envía una señal de cambio de color a verde desde el UR5e al IOT2050 indicando que el movimiento del robot ha finalizado. Se cierra la conexión entre ambos dispositivos.
9. Finaliza el programa y los dispositivos permanecen a la escucha de las nuevas órdenes que envíe el usuario.

3.2 Planteamiento general.

En este apartado, se pretende profundizar en los objetivos redactados en el capítulo 1 de una manera más específica y mencionando las herramientas y tecnologías empleadas en la arquitectura del sistema.

A continuación, se enumeran y describen los objetivos planteados anteriormente, analizando las diversas alternativas que se pueden adoptar para cumplir estos propósitos y justificando la razón que ha llevado a elegir el método más adecuado para este proyecto.

Asistencia remota.

La aplicación de realidad mixta a desarrollar debe de ser intuitiva proporcionando elementos digitales sencillos como menús de opciones, botones o diálogos. Estos elementos se deben de proyectar en las lentes del dispositivo de manera ordenada consiguiendo que el operario realice la tarea encomendada paso por paso y sin errores.

La opción de ofrecer al usuario asistencia remota mediante una narración auditiva es bastante atractiva. No obstante, esto llevaría a realizar una grabación de voz de una persona con capacidad de llamar la atención del oyente y, que comunicase de manera clara y precisa los mensajes que se desean transmitir durante la ejecución de la aplicación.

Esta opción fue descartada debido ante la dificultad de encontrar una persona con dichas capacidades que se mostrase voluntaria para realizar las diferentes grabaciones. En su lugar, se prefiere hacer uso de sonidos concordantes con los objetos digitales con los que interactúa el operario de manera que la atención del oyente se mantenga firme durante el desarrollo de la actividad.

Cooperación humano-máquina.
El operario debe de ser capaz de ordenar qué maniobra quiere realizar con el cobot mediante el uso de la MR, no solo mostrar información del estado en el que se encuentre el robot en cada instante.

Consecución de las maniobras y trayectorias indicadas mediante lenguaje natural.
El paquete de herramientas MRTK integra una serie de herramientas que permiten desarrollar aplicaciones de realidad mixta que reconozcan las manos, la voz y la mirada del usuario.

Incorporar este paquete a este proyecto supone una gran ventaja, ya que permite la introducción de comandos manuales, vocales y ópticos en el desarrollo de la actividad con el robot colaborativo.

Precisión y repetibilidad aceptables.
La aplicación debe de servir de soporte al operario para realizar tareas cotidianas que requieren cierta precisión como el Pick & Place o la ejecución de trayectorias lineales. Sin embargo, conseguir este propósito requiere trabajar con un sistema de referencia externo al del robot, ya que este se sitúa dentro de la base del robot.

La solución que adopta el sistema de realidad mixta es la de utilizar códigos QR que sirvan para construir una matriz de transformación que se aplique a la hora de transformar las coordenadas tomadas en la referencia de HL2 a la referencia robot.

Esta idea surge de “Closed-Loop Robotic Arm Manipulation Based on Mixed Reality”, donde se utiliza un código QR para establecer el sistema de referencia virtual del robot.

De esta manera, nunca habrá problemas de precisión a la hora de establecer el sistema de coordenadas real porque los códigos QR permanecerán inmóviles en el entorno y estarán situados a una distancia conocida del centro de la base del robot, donde se sitúa el sistema de coordenadas real del UR5e.

Automatización de trayectorias.
Con el propósito de acercarse a la Industria 4.0 y conseguir una aplicación de realidad mixta que en un futuro pueda ser utilizada en el ámbito industrial, la programación realizada permitirá realizar maniobras con el robot que se puedan repetir en bucle.

Maniobras como el Pick & Place podrán repetirse indefinidamente tan solo indicando el punto de origen y destino del objeto que se quiere trasladar. Además, el usuario podrá dejar ejecutándose esta maniobra incluso apagando el dispositivo de realidad mixta una vez ordene el comienzo de esta actividad.

Otorgar seguridad a la actividad.
La aplicación de realidad mixta a desarrollar adopta la función de limitar el espacio de trabajo donde el operario puede mover el robot. Esta idea, propuesta en *"Programming Robots by Demonstration using Augmented Reality",* permite al operario interactuar con el robot colaborativo en un entorno seguro donde no se alcancen singularidades o pinzamientos de los ejes.

El espacio de trabajo virtual a configurar será el de la Ilustración 22 donde su diseño 3D será una esfera verde de 1700 mm de diámetro que cambiará de color a rojo cuando el usuario indique una maniobra fuera del alcance del robot.

Repuesta ágil y fiable.
Una vez se mande la orden por parte del operario, el robot debe recibirla, compilarla y ejecutarla en un plazo relativamente corto. Se propone abrir y cerrar la conexión entre los dispositivos siempre que se desean transmitir datos entre estos para garantizar la fiabilidad de la comunicación ante largos periodos de inactividad.

3.3 Diseño y desarrollo de la aplicación de realidad mixta

El diseño y desarrollo de la aplicación de realidad mixta que se ejecuta en HL2 se ha realizado en la plataforma Unity. Este apartado presenta los pasos realizados cronológicamente a la hora de plantear y realizar el diseño de la interfaz digital y de la experiencia que tiene el operario a la hora de llevar a cabo la actividad.

En el ámbito web esto se llama UI/UX, que tienen la función de que el usuario encuentre lo que busca en el menor tiempo posible y además tenga una buena experiencia al visitar la página web. [43]

La UX (User Experience) se centra en optimizar un sitio web para que sea agradable y eficaz, mientras que la UI (User Interface) se encarga del diseño de todos los aspectos visuales, como la apariencia o la interactividad de una página web. [44]. En la Tabla 6 se recogen varios de factores y elementos que intervienen en el UI/UX.

En el diseño y desarrollo de la interfaz digital de la aplicación se ha intentado trasladar estos dos conceptos al ámbito de la realidad mixta para proporcionar una experiencia satisfactoria al usuario que lleve a cabo la actividad con el HL2.

UX	UI
Percepción Saber qué llama la atención al usuario	**Propósito del sitio** Cumplir con el objetivo principal
Emociones Cómo se generan y qué implicaciones tienen	**Ayuda al usuario** Construir caminos que ayuden a encontrar lo que busque el usuario.
Memoria Pensar en las limitaciones de las personas para recordar	**Mostrar el contenido** Claridad y sencillez
Motivación Satisfacción durante la navegación	**Diseño gráfico funcional** Los elementos gráficos deben ir dirigidos a ayudar al usuario en encontrar lo que quiere

Tabla 6. Factores y elementos del UI/UX de la aplicación de realidad mixta

3.3.1 Funcionalidades

Para esta sección del trabajo, se van a enumerar las funcionalidades utilizadas para cumplir los objetivos propuestos. Es importante tener en cuenta que no todas las aplicaciones de robots son adecuadas para la programación mediante realidad mixta debido a la precisión limitada de HL2.

En esta selección también se tienen en cuenta los rasgos estéticos de los objetos digitales utilizados para orientar al usuario en la ejecución de las tareas con el fin de conseguir una experiencia funcional y satisfactoria.

Navegación mediante diálogos y menús.

El usuario navegará por la aplicación mediante diálogos digitales donde solo bastará con utilizar el dedo índice o la voz para interactuar con estos. Estos diálogos contendrán información de soporte para el usuario que indicarán las alternativas que se pueden tomar en cada momento de la ejecución.

La navegación por la aplicación se puede resumir en tres fases:

1. **Presentación y revisión de los pasos previos.** Se dará la bienvenida al usuario y se le avisará de comprobar los requisitos que intervienen en el correcto funcionamiento del sistema.
2. **Definición del sistema de coordenadas.** Se utilizará la cámara de HL2 para escanear los códigos QR necesarios para construir la matriz de transformación homogénea de la mesa de trabajo respecto de la referencia del dispositivo de realidad mixta.
3. **Selección de la maniobra.** Se mostrará un menú digital que permita la ejecución de tres modos de actividad diferentes: Real Time Mode, Trayectory Mode, Pick & Place y Scan. Más adelante, se profundizará en el propósito de cada modo.

Comandos de voz para enviar coordenadas.

Los comandos de voz servirán para interactuar con los paneles de navegación, pero fundamentalmente permitirán al usuario indicar posiciones de una forma rápida y cómoda.

El procedimiento se basará en ubicar el dedo índice en las coordenadas del espacio que se quieren enviar al robot y utilizar una palabra clave programada en Unity para llamar a la función que realice la acción deseada.

Botonera con maniobras básicas y ayuda.

Se incluirá una botonera digital con tres botones. Cada uno tendrá un propósito que será el siguiente:

- Se utilizará un botón para mover el UR5e a la posición ***"Home".*** El "Home" es la posición de inicio del robot, es decir, la posición en la que todos los ejes del robot están alineados entre sí.
- Se utilizará un botón para mover el robot a una posición donde todos los ejes se encuentren en una posición cómoda donde empezar las diferentes maniobras. A esta posición se la ha denominado ***"Repose".***
- Se utilizará un botón para mostrar un panel de ayuda que incluya una guía de comprobaciones en el sistema que debe de revisar el operario si se encuentra en un momento donde el robot no responde a sus órdenes. Este botón será denominado ***"Help".***

La forma de interactuar con estos botones se basará en realizar un gesto de presión en la superficie de cada uno con cualquiera de las dos manos.

Advertencias y seguimiento de las maniobras

Cuando el usuario indique maniobras fuera del alcance del robot, la aplicación dejará de transmitir datos a ROS y cambiará el color del espacio de trabajo virtual del robot a rojo. Cuando el usuario desee comenzar una nueva maniobra, este cambiará a color verde.

Cada vez que se marque una coordenada en el espacio que es enviada al UR5e se proyectará una esfera amarilla en dicha ubicación.

De esta manera, el operario sabrá de manera anticipada la posición que alcanzará el TCP del robot y, podrá cancelar la maniobra, si observa que el robot puede colisionar con objetos de su alrededor.

Conexión con ROS al iniciar la aplicación.

Se realizará la conexión con el puente de ROS en el momento en el que se inicie la aplicación. Para ello, se utilizará la biblioteca RosSharp3 que proporciona métodos ya implementados para comunicar Unity con ROS.

3.3.2 Requisitos previos

En este apartado se resumen las especificaciones técnicas que se requieren para poder desarrollar una aplicación de realidad mixta con las características de este proyecto.

Los requisitos necesarios son los siguientes: [45] [46]

- Un equipo Windows 10 configurado con las herramientas correctas instaladas.
- SDK de Windows 10 10.0.18362.0 o posterior.
- Un dispositivo Microsoft HoloLens 2 configurado para el desarrollo.
- Unity Hub con Unity 2020.3 LTS o Unity 2019.4 LTS instalado (OpenXR requiere 2020.3.8 o posterior para evitar errores).
- Herramienta de características de Mixed Reality.
- Visual Studio 2019 o posterior con las siguientes cargas de trabajo:
    - Desarrollo de escritorio de .NET
    - Desarrollo de escritorio con C++
    - Desarrollo para la Plataforma universal de Windows (UWP)
    - Desarrollo de juegos con Unity
    - SDK de Windows 10 versión 10.0.19041.0 o 10.0.18362.0 o SDK de Windows 11

Adicionalmente, Unity debe tener incorporado las siguientes bibliotecas:

- La biblioteca ROS# que permite comunicarse con ROS desde aplicaciones .NET, en particular Unity3D.
- La biblioteca NuGet que facilita la creación de paquetes que se distribuyen en un servidor.

3.3.4 Diseño de la interfaz digital

En este apartado, se exponen los pasos realizados para configurar el entorno de desarrollo de Unity con Mixed Reality Toolkit.

Posteriormente, se procede a explicar los elementos digitales utilizados en la escena, así como el protocolo de navegación implementado para guiar al usuario en el desarrollo de la actividad con HL2.

3.3.4.1 Configuración del proyecto

La forma más fácil de desarrollar aplicaciones de realidad mixta en Unity es con Mixed Reality Toolkit (MRTK). MRTK permite definir automáticamente la configuración de un proyecto de realidad mixta y proporciona un conjunto de características para acelerar el proceso de desarrollo. [45]

Una vez se ha creado el proyecto 3D en Unity, se debe de configurar para Windows Mixed Reality. Para ello, se debe de cambiar la plataforma de compilación a *Plataforma Universal de Windows*, instalada anteriormente. Además, se debe de cambiar la arquitectura a ARM64, compatible con el procesamiento de datos de 64 bits. Este es el estándar de arquitectura de CPU propia de HL2.

Esta configuración se realiza en el panel *Build Settings*. La configuración de compilación completa de este proyecto es la siguiente:

- Dispositivo de destino: HoloLens
- Arquitectura: ARM64
- Tipo de compilación: Proyecto D3D
- Versión de SDK de destino: Última instalada
- Versión mínima de la plataforma: 10.0.10240.0
- Versión de Visual Studio: Última instalada
- Build and run on (Build and Run on): Máquina local
- Configuración de compilación: Versión (hay incidencias de rendimiento conocidas con Depuración)

Posteriormente, se debe de incorporar el paquete MRTK al proyecto de Unity añadiendo las características mostradas en la Ilustración 35. El kit de herramientas Mixed Reality Toolkit se descarga gratuitamente desde el Centro de descargas de Microsoft.

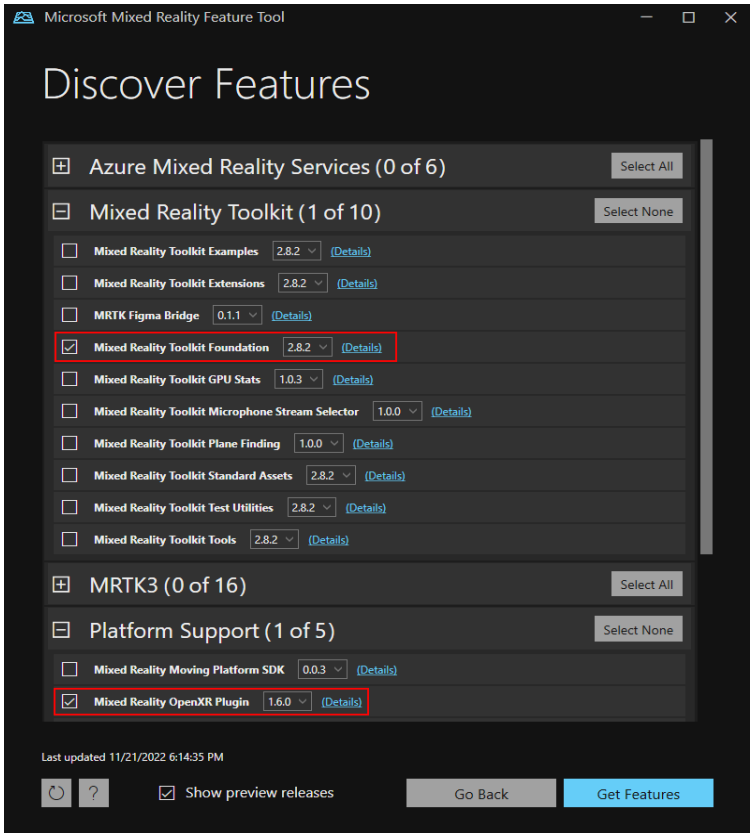


Ilustración 35. Características de MRTK incorporadas en el proyecto

El paquete Mixed Reality Toolkit Foundation es el único necesario que se debe importar y configurar para usar MRTK con el proyecto. Este paquete incluye los componentes principales necesarios para crear una aplicación de realidad mixta. [45]

Una vez se incorpora el paquete al proyecto, se debe habilitar la administración de complementos XR mediante el plugin Unity OpenXR. La API de OpenXR proporciona la predicción de posición principal, el tiempo de fotogramas y la funcionalidad de entrada espacial que necesitará para crear un motor que puede tener como destinos dispositivos de realidad mixta y envolventes. [45]

Por último, asegurarse de añadir los perfiles de interacción que necesita Unity para diseñar la aplicación de realidad mixta como independiente de HL2. Estos perfiles de interacción son los siguientes:

- Eye Gaze Interaction Profile (Perfil de interacción con la mirada)
- Microsoft Hand Interaction Profile (Perfil de interacción manual de Microsoft)
- Microsoft Motion Controller Profile (Perfil del controlador de movimiento de Microsoft)

Por consiguiente, se deben de habilitar los correspondientes grupos de características OpenXR de Microsoft Hololens 2, tal como se aprecia en la Ilustración 36.

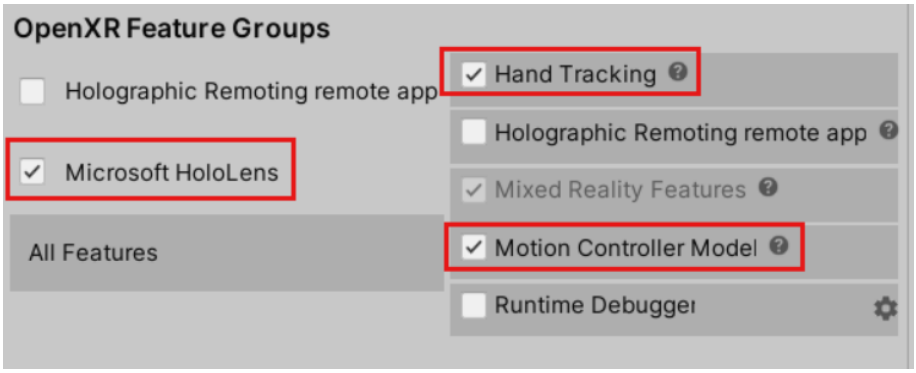


Ilustración 36. Grupos de características de MRTK incorporadas en el proyecto

Es recomendable establecer la reducción del formato de profundidad a 16 bits, ya que puede mejorar el rendimiento de los gráficos en el proyecto.

En este momento, MRTK se agrega a la escena y aparecen dos nuevos objetos en la ventana Hierarchy: MixedRealityToolkit y MixedRealityPlayspace. El objeto MixedRealityToolkit contiene el kit de herramientas en sí y el MixedRealityPlayspace garantiza que los cascos, controladores y otros sistemas necesarios se administren correctamente en la escena. [45]

También se contempla el GameObject Main Camera (Cámara principal), que se vuelve un elemento secundario del objeto MixedRealityPlayspace. Esto permite al espacio de juego administrar la cámara simultáneamente con los SDK. [45]

Un GameObject es el objeto fundamental de Unity que representa cualquier elemento del escenario tanto visible como no visible. Este objeto sirve como una agrupación de componentes que son los que implementan la funcionalidad de este.

3.4.4.2 Diseño de la escena

Los objetos digitales que se muestran en las lentes del dispositivo de realidad mixta tienen que ser previamente diseñados y colocados en la escena.

MRTK incorpora una serie de "*prefabs*" que tienen el propósito de simplificar la tarea de diseño de objetos digitales y optimizar la calidad de representación en la escena. Un prefab simplemente es un GameObject ya creado (prefabricado). La aplicación de realidad mixta que plantea este TFG hace uso de estos prefabs para guiar e interactuar con el usuario en el propósito de crear instrucciones que posteriormente serán enviadas a ROS.

3.4.4.2.1 Prefabs

Cabe destacar que todos estos objetos digitales son interactivos, es decir, que el humano que navega por la aplicación puede realizar ciertas acciones con ellos como trasladarlos o rotarlos en el espacio virtual. A continuación, se exponen los prefabs de MRTK utilizados en la aplicación desarrollada.

Dialog.

En el momento en el que el usuario inicie la aplicación de realidad mixta, se desea darle la bienvenida y, de alguna manera, guiarle por la escena para seguir los pasos requeridos ordenadamente. MRTK incorpora un prefab llamado *"Dialog"* que representa un panel con botones que permite mostrar textos de información.

En la Ilustración 37 se contempla el aspecto del Dialog mencionado anteriormente. En este ejemplo, se da la bienvenida al usuario y se incluye información que indica las opciones que puede seleccionar. También incluye unos mini paneles debajo de los botones que indican al operario que también puede interactuar con el diálogo mediante comandos de voz.

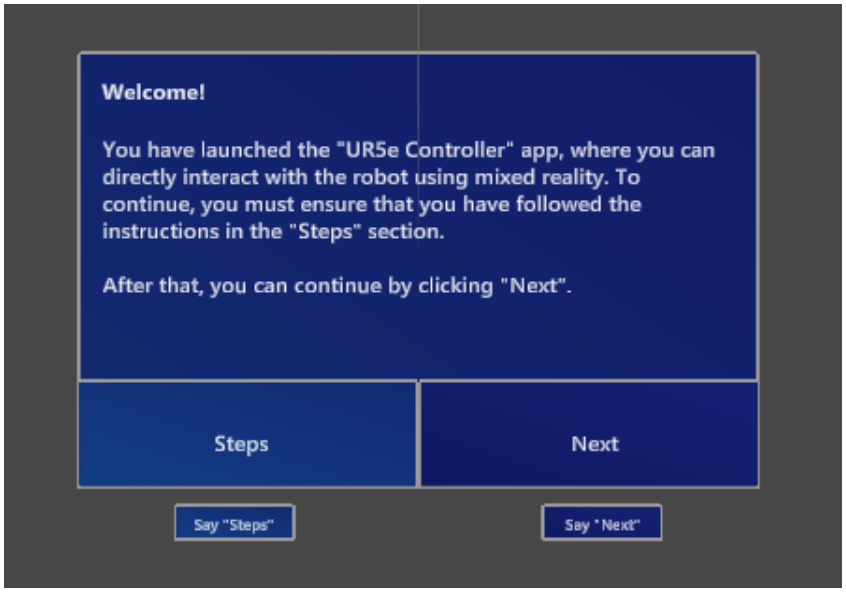


Ilustración 37. Paneles de diálogo digitales de la aplicación de realidad mixta

Este propósito se consigue mediante la adicción de los scripts *"BoxCollider", "Interactable"* y *"NearInteractionTouchable"* que incorpora este GameObject. Estos scripts permiten que el usuario pueda interactuar con los botones manualmente. Mediante el script *"Button Config Helper"* se consigue llamar funciones al pulsar estos botones.

En la Tabla 7 se resumen los diálogos empleados con un título descriptivo de su función dentro de la aplicación.

Nombre	Función	Alternativas
Welcome Dialog	Dar la bienvenida al usuario	Steps/Next
Steps Dialog	Mostrar los pasos previos a realizar con el robot colaborativo	Skip/Continue
ROS Dialog	Mostrar los pasos previos a realizar con el contenedor de Docker	Back/OK
Referential Dialog	Informar del procedimiento que se debe de realizar con los códigos QR. Se bloquea mientras la aplicación no haya recibido las coordenadas de los códigos QR en la referencia del UR5e	Set Origin
Info Dialog	Informar sobre el espacio de trabajo	More
More Dialog	Informa sobre las maniobras que se pueden realizar con el UR5e. Dar paso al comienzo de la actividad	Begin
Warning Dialog	Cuando el usuario indique una posición inalcanzable para el UR5e, se proyectará este diálogo de advertencia que informa al usuario de la no posibilidad de enviar esa coordenada al robot colaborativo	Repeat/Change
MoveLorMoveJ	Una vez se hayan indicado los puntos de una trayectoria, pedir al usuario que seleccione si desee que el robot la realice mediante movimiento lineal o articular de los ejes.	MoveL/MoveJ
Repeat Dialog	Cuando se finaliza una trayectoria o un Pick & Place, pedir al usuario que indique una nueva maniobra o que repita la anterior indefinidamente	New /Loop
PointNotReached Dialog	Realizar el mismo propósito que el diálogo Warning Dialog, pero cuando está activo el modo Pick & Place	Agreed

Tabla 7. Diálogos digitales empleados

Otro tipo de diálogos empleados son los que se utilizan para escanear los códigos QR. El primer botón, denominado Scan, da lugar al escaneo del código QR, mientras que el segundo, denominado Close, cierra el diálogo y da paso al siguiente diálogo. Estos diálogos informan sobre el contenido que alberga el código QR y dan lugar a la función de escaneo.

En la Ilustración 38 se puede apreciar como este diálogo tiene incorporado un botón llamado Scan que se encarga de iniciar la detección de los códigos QR y un botón llamado Close que sirve para cerrar este diálogo.

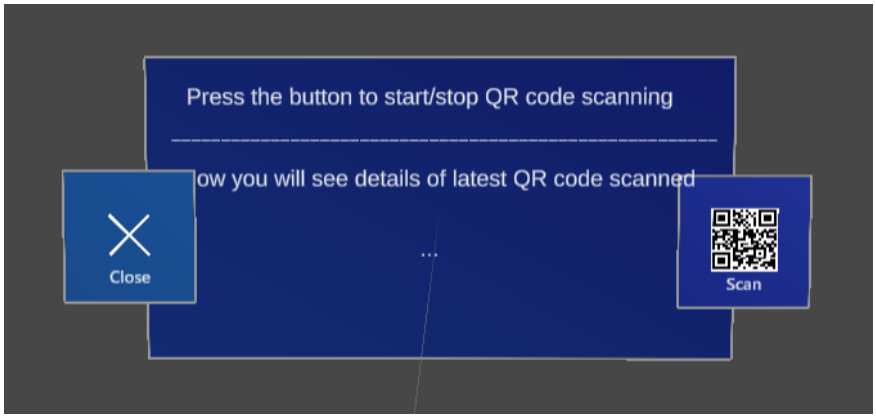


Ilustración 38. Diálogos QR

Menu.

MRTK contiene un prefab llamado *"Menu",* que es similar al Dialog, donde se muestra información en un menú con una serie de botones con los que el usuario puede interactuar.

En el momento en el que ya se hayan escaneado los tres códigos QR, se dará paso a la elección del modo de operación. El objetivo es que, al seleccionar un botón, se active un escenario en el que el usuario pueda utilizar los comandos de voz correspondientes a cada modo de actividad para publicar en ROS los topics con las coordenadas que el usuario indique.

Como se puede contemplar en la Ilustración 39, hay una figura de una chincheta que se ubica en la esquina superior derecha del menú. Al pulsar en el recuadro donde se ubica esta chincheta, el menú comenzará a seguir la mirada del usuario permanentemente hasta que se vuelva a pulsar.

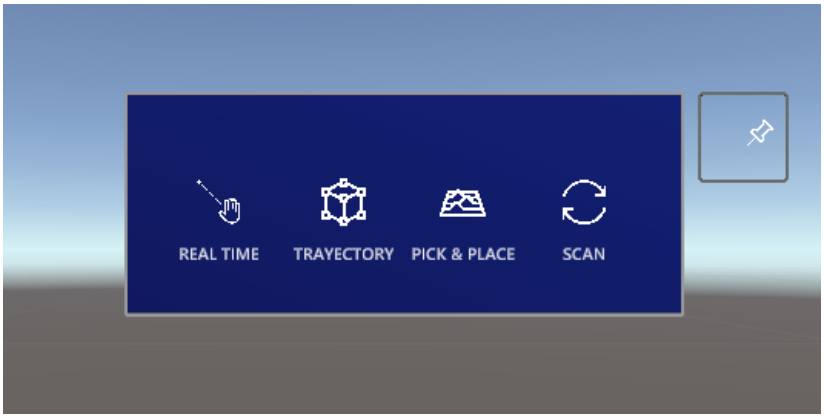


Ilustración 39. Menú de operaciones digitales de la aplicación de realidad mixta

Buttons.

Controlar al UR5e rápidamente es uno de los objetivos de este proyecto. Moverlo al Home o a una posición cómoda en la que pueda empezar a trabajar con la mesa de trabajo son tareas que tradicionalmente se realizan mediante la Flex Pendant y requieren cierto tiempo para poder llevarlas a cabo.

Sin embargo, esta aplicación pretende simplificar en gran escala esta tarea simplemente haciendo uso de una botonera digital que emule una real que tiene establecido un conexionado para que el robot realice ciertas tareas automáticamente. En esta botonera habrá tres botones dedicados a cumplir los siguientes propósitos:

1. **Home.** Mover el UR5e al Home.
2. **Repose.** Mover el UR5e a la posición de reposo.
3. **Help.** Mostrar información de ayuda al operario.

El diseño de esta botonera consta de tres botones azules con una etiqueta que muestra una palabra que resume la tarea que se va a realizar al pulsar cada botón, tal como se aprecia en la Ilustración 40.

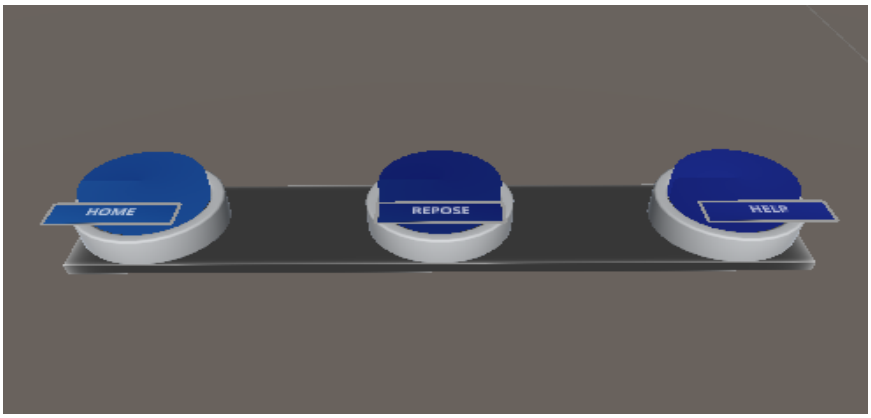


Ilustración 40. Botonera digital de la aplicación de realidad mixta

Panel.

Los diálogos y paneles mostrarán información de interés al usuario. La diferencia entre ambos es que estos últimos no son interactivos, es decir, no contienen botones. Estos paneles se proyectarán en las lentes del dispositivo de realidad mixta al interactuar con los botones del menú y el botón Help de la botonera digital.

En la Tabla 8 se muestran las denominaciones dadas a los paneles empleados en la aplicación junto al propósito que cumple cada uno.

Nombre	Propósito
Instructions	Ayudar a identificar el problema que se puede presentar en el sistema y a cómo continuar la actividad
RealTime Instructions	Informar de los comandos de voz que se deben de emplear para indicar las coordenadas que se desean enviar al robot en este modo
Trayectory Instructions	
PickAndPlace Instructions	

Tabla 8. Paneles digitales empleados

Los paneles del menú indicarán las instrucciones que debe de seguir el usuario para poder indicar y enviar coordenadas al robot. Un ejemplo de estos paneles se muestra en la Ilustración 41.

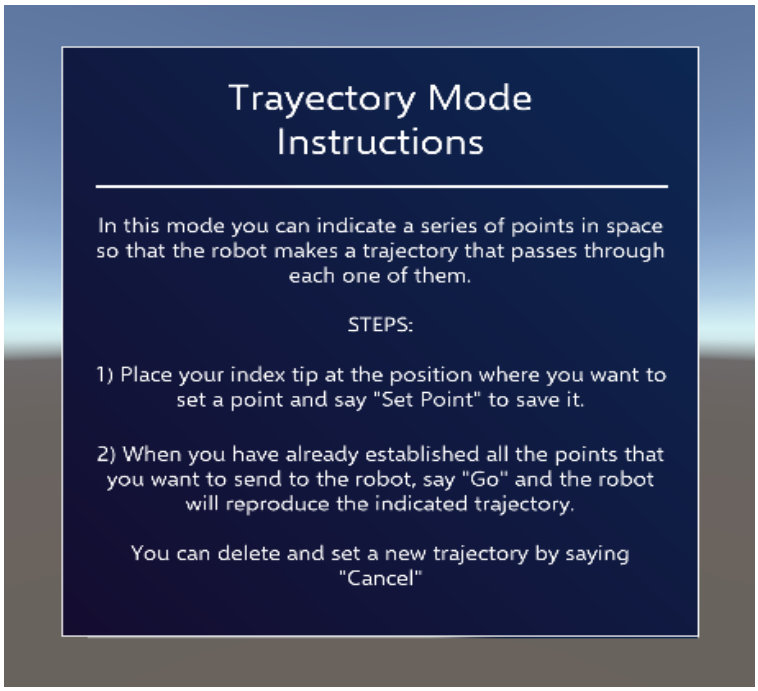


Ilustración 41. Panel de instrucciones del modo Trayectory Mode

3.3.4.2.2 Comandos de voz

La aplicación de realidad mixta incorpora una serie de comandos de voz programados para llamar a funciones dentro del código fuente. Los comandos de voz utilizados en esta aplicación se muestran en la Tabla 7 junto a la función que desempeñan dentro del programa.

Comando de voz	Objetivo
Steps	Ocultar Welcome Dialog y proyectar Steps Dialog
Next	Ocultar Welcome Dialog y proyectar Referential Dialog
Skip	Ocultar Steps Dialog y proyectar Welcome Dialog
Continue	Ocultar Steps Dialog y proyectar ROS Dialog
Back	Ocultar el diálogo ROS Dialog y proyectar Steps Dialog
OK	Ocultar ROS Dialog y proyectar Referential Dialog
Set Origin	Ocultar Referential Dialog y proyectar QRCodePanel
Scan	Iniciar la detección de los códigos QR
More	Ocultar More Dialog y proyectar Info Dialog
Begin	Ocultar Info Dialog y proyectar el menú y la botonera
Repeat	Ocultar Warning Dialog y cambiar el color del espacio de trabajo a verde. Permite trazar nueva maniobra.
Change	Ocultar Warning Dialog , proyectar el menú y la botonera y cambiar el espacio de trabajo de color.
Agreed	Ocultar PointNotReached Dialog y cambiar el color del espacio de trabajo a verde.
New	Ocultar Repeat Dialog y proyectar el menú y la botonera. Permitir realizar una nueva maniobra
Loop	Ocultar Repeat Dialog. Repetir maniobra en bucle
Line	Ocultar MoveLorMoveJ y proyectar Repeat Dialog. Enviar “MoveL” al topic correspondiente
Joint	Ocultar MoveLorMoveJ Dialog y proyectar Repeat Dialog. Enviar “MoveJ” al topic correspondiente
Start	Enviar un topic con la posición del dedo índice del usuario en el instante en el que se utiliza
Stop	Parar el movimiento del robot y proyectar el menú y la botonera. Detener detección de los códigos QR
Set point	Guardar puntos indicados en el modo Trayectory
Go	Enviar un topic con la trayectoria programada
Cancel	Borrar trayectoria trazada
Pick	Enviar un topic con las coordenadas programadas para que el robot agarre un objeto con la pinza
Place	Enviar un topic con las coordenadas programadas para que el robot deposite un objeto con la pinza
Close	Cerrar el diálogo QRCodePanel

Tabla 9. Comandos de voz y teclas que los simulan en Unity

El perfil de entrada de voz de MRTK define las palabras clave. Al asignar el script *“SpeechInputHandler”* , puede hacer que cualquier objeto responda a

las palabras clave definidas en el perfil de entrada de voz. [47] Cuando se pronuncien alguna de estas palabras, el SpeechInputHandler proporcionará la etiqueta de confirmación de voz para mejorar la confianza del usuario. [45]

El dispositivo HL2 procesa continuamente el audio que capta por sus micrófonos y analiza si las palabras que el usuario pronuncia coinciden con las programadas en la aplicación.

3.3.4.2.3 Secuencia de navegación

A continuación, se elabora una descripción del programa de realidad mixta desarrollado para realizar las tareas que se le han integrado. En la Ilustración 42 se muestra un diagrama de flujo que muestra cómo es la navegación por la aplicación de realidad mixta con HL2.

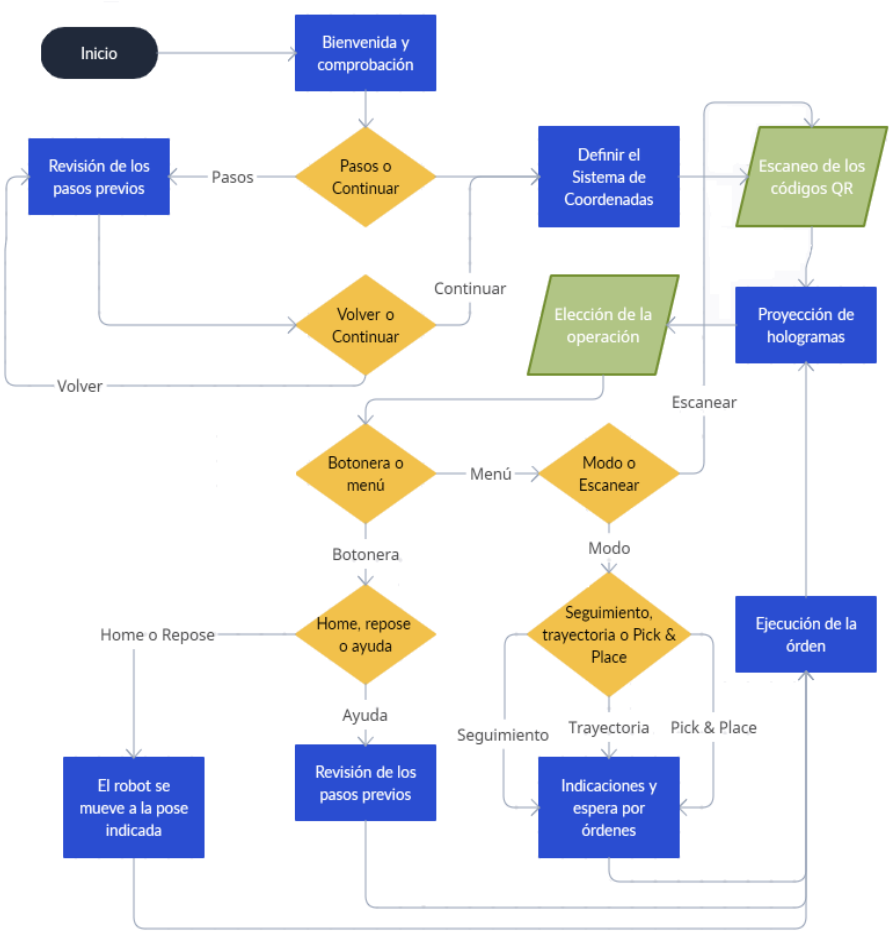


Ilustración 42. Diagrama de flujo de la navegación por la aplicación de realidad mixta

La aplicación comienza proyectando el diálogo de bienvenida (Welcome Dialog), en este momento, el usuario debe de indicar si quiere repasar las instrucciones previas que se tienen que tomar antes de iniciar la actividad o si

desea continuar con el siguiente paso. Si pulsa el botón Steps, se proyecta el diálogo Steps Dialog, si pulsa *Next,* aparece el objeto Referential Dialog.

Las instrucciones previas se dividen en dos diálogos: Steps Dialog y ROS Dialog. Se puede avanzar y retroceder entre ellos las veces que se desee.

Una vez que se han leído las instrucciones previas o se ha indicado Next, aparece el diálogo Referential Dialog que informa del siguiente paso. En este momento el usuario debe de pulsar el botón Set Origin para escanear los códigos QR.

Posteriormente, aparece el dialogo QRCodePanel que contiene el botón Scan que habilita la lectura de los códigos QR. Cuando se detecten los códigos QR y se cierre este diálogo, se proyecta el espacio de trabajo digital del robot, la botonera y el menú. En este momento, el usuario puede tomar la decisión de realizar alguna acción con el robot colaborativo.

La secuencia de los diálogos mostrados hasta el momento de la selección de la maniobra con el robot viene resumida en la Ilustración 43 donde se hace referencia a los botones o comandos de voz que se deben de activar para navegar por la aplicación.

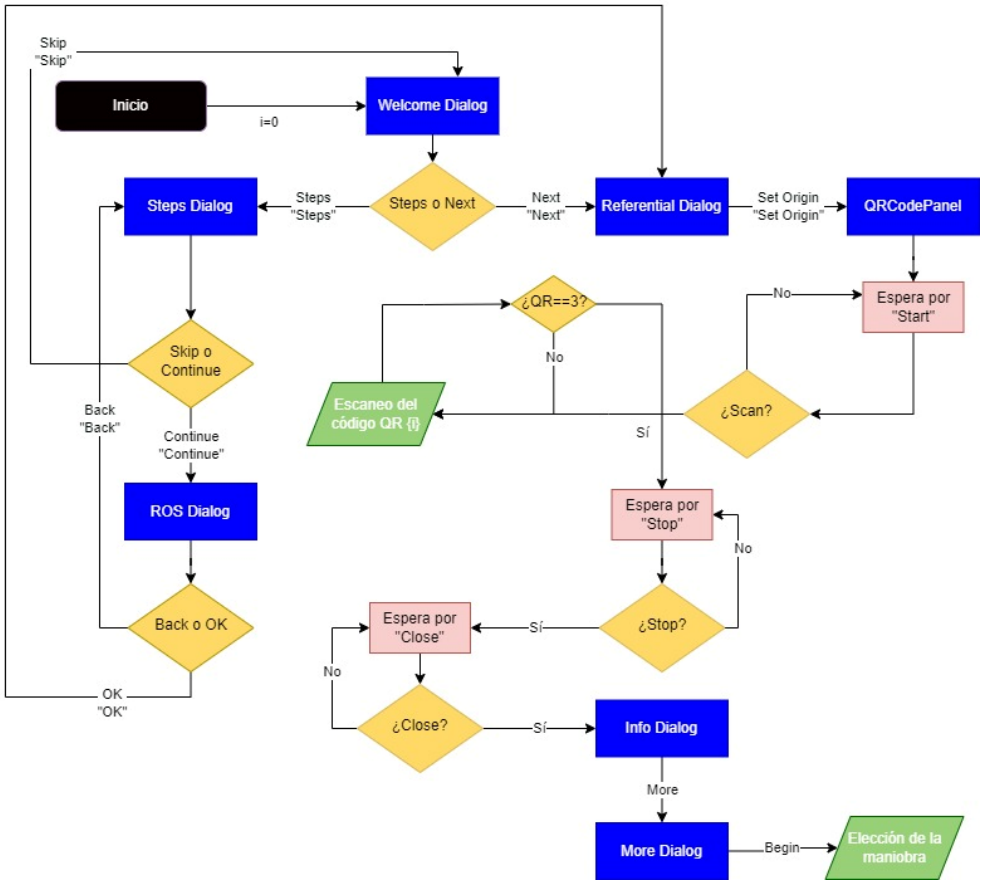


Ilustración 43. Diagrama de flujo de diálogos hasta la selección de la maniobra

En la Ilustración 44 se encuentra esquematizado el proceso de acontecimientos transcurridos desde que se pulsa alguno de los tres botones de la botonera digital hasta que se vuelve al mismo estado de selección de maniobra.

Ilustración 44. Diagrama de flujo de la botonera digital

En este proceso ocurre lo siguiente:

- Si se pulsa el botón Home:
  - Se envía una cadena de caracteres con el mensaje “HOME” al topic correspondiente de ROS.
- Si se pulsa el botón Repose:
  - Se envía una cadena de caracteres con el mensaje “REPOSE” al topic correspondiente de ROS.
- Si se pulsa el botón Help:
  - Se proyecta el panel Instructions que informa al usuario de las medidas que se pueden tomar para comprobar el correcto estado del sistema con sugerencias que ayudan a identificar el posible error.

<u>RealTime Mode.</u>

El proceso para enviar coordenadas de los modos de operación contenidos en el menú digital es algo más complejo. Por lo tanto, conviene ir analizando el proceso que sigue cada uno.

En el modo de tiempo real (Time Real) se pretende hacer un seguimiento del dedo índice de la mano derecha del usuario por parte del TCP del UR5e. Para ello, se deben de analizar y enviar las coordenadas del dedo índice según un periodo de muestreo dado.

La grabación de las coordenadas comienza cuando el operario activa el comando de voz «Start». En este momento, se inicia la grabación de las coordenadas del dedo índice y se analizan calculando la distancia relativa entre las nuevas con las anteriores. Si la distancia absoluta supera el centímetro, se comprueba que esta coordenada se encuentra dentro del espacio de trabajo del robot.

En caso de no pasar este último filtro, se proyecta el diálogo de advertencia (Warning Dialog) donde se informa de la interrupción del modo de actividad y, se pregunta al usuario si quiere repetir la grabación o elegir otra operación.

Cuando la coordenada supera ambos filtros, se publica en el topic correspondiente de ROS. El proceso se repite indefinidamente hasta que el usuario activa el comando de voz «Stop».

En la Ilustración 45 se muestra el proceso de acontecimientos transcurridos desde el momento en el que se activa el modo RealTime hasta que se decide detener el envío de coordenadas a ROS.

Trayectory Mode.

En el modo Trayectory se pretende almacenar una serie de puntos en el espacio para enviarlos a ROS con el fin de que el UR5e pueda recorrer estos puntos una vez sean traducidos en el script de ROS al lenguaje nativo del robot. El procedimiento viene representado en la Ilustración 46.

Una vez se pulsa el botón Trayectory Mode del menú, se habilita la opción de utilizar los comandos de voz propios de este modo: «Set Point», «Cancel» y «Go». El primero sirve para guardar un punto en el espacio, el segundo para cancelar el envío de las coordenadas indicadas, y el último para publicar las coordenadas en ROS.

Este proceso requiere una validación de las coordenadas para comprobar si se encuentran dentro del espacio de trabajo del robot. Cuando el punto indicado se encuentra fuera de los límites, aparece el diálogo de advertencia mencionado anteriormente y se ofrece al usuario la posibilidad de elegir trazar una nueva trayectoria o elegir otro modo de operación.

Si el usuario indica un punto que no desea enviar al robot, puede hacer uso del comando de voz «Cancel» para borrar la trayectoria programada y repetirla de nuevo. En el momento en el que el usuario haya indicado todos los puntos que desea enviar al robot, debe pronunciar «Go».

Posteriormente, se debe de elegir si se quiere que el robot replique la trayectoria trazada mediante movimiento MoveL o MoveJ. Esta decisión se toma pulsando uno de los dos botones del diálogo MoveLorMovej o

pronunciando las palabras clave «Line» o «Joint». La opción elegida publica una cadena de caracteres que la identifica en el topic de ROS correspondiente.

Cualquiera de las dos elecciones hace que se proyecte el diálogo Repeat Dialog. Este diálogo permite volver al estado de elección de maniobra o repetir la trayectoria anteriormente trazada indefinidamente.

Si el operario selecciona la opción Loop, se publica un mensaje en el topic correspondiente que se encarga de gestionar el programa que genera el URScript. Por el contrario, si se selecciona New Trayectory, el usuario tiene la posibilidad de realizar una maniobra diferente. Esta opción también publica un topic en ROS.

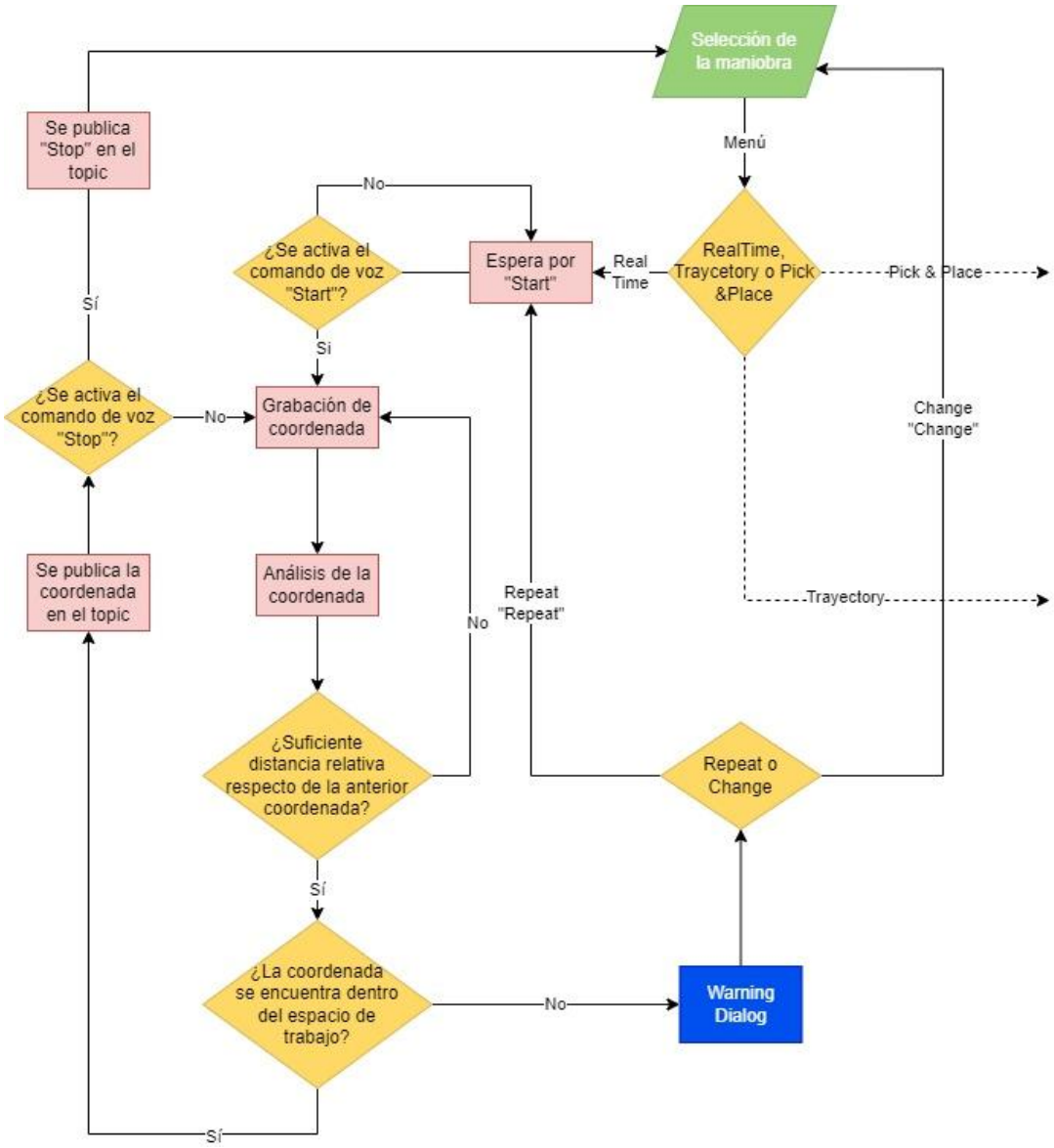


Ilustración 45. Diagrama de flujo del modo Real Time

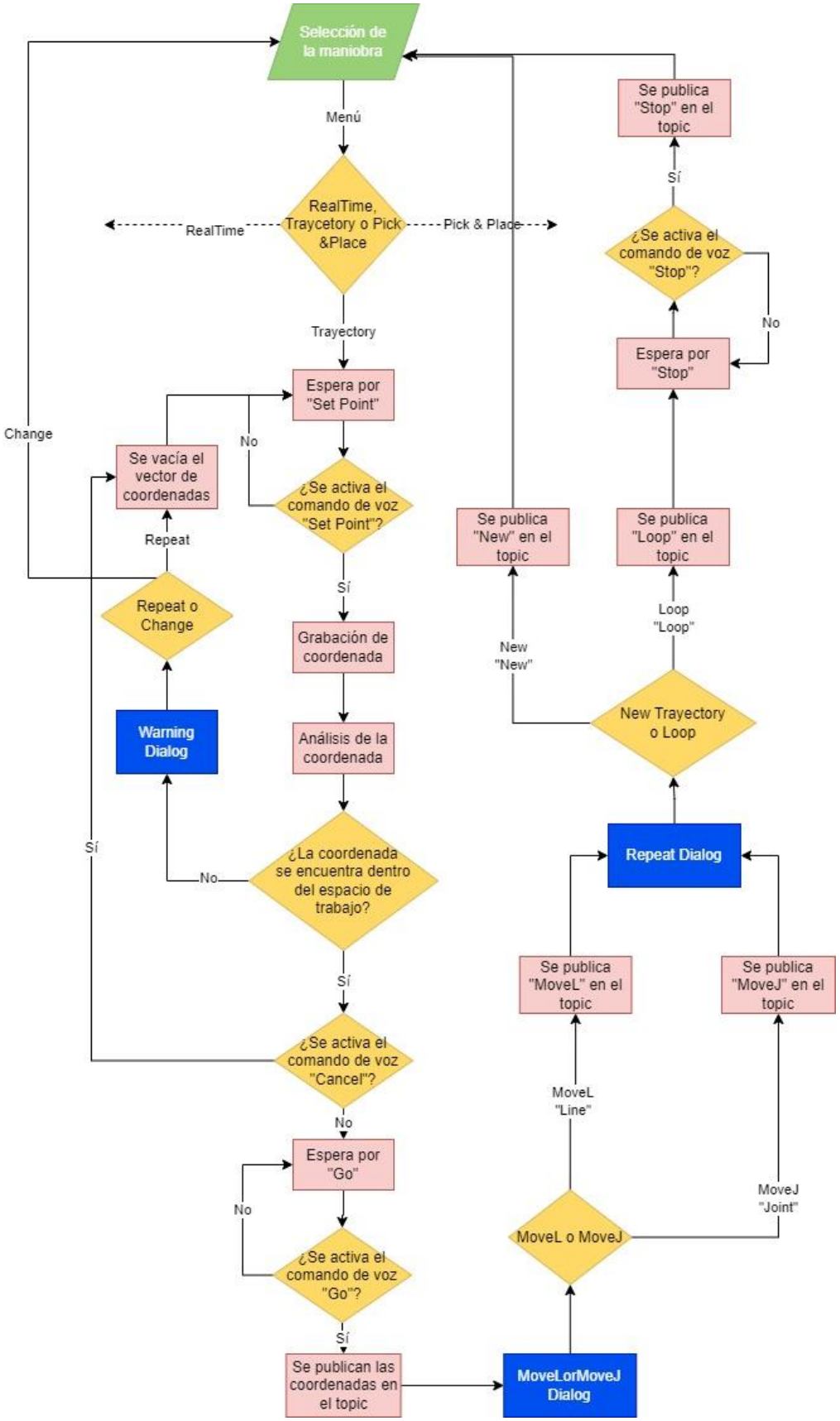


Ilustración 46. Diagrama del flujo del modo Trayectory

Pick & Place.

El último tipo de maniobra introducido en la aplicación es el Pick & Place. El procedimiento es similar al anterior, ya que está basado en la publicación de coordenadas en los topics de ROS.

Para comenzar, el usuario debe de pulsar en el botón Pick & Place del menú digital. A continuación, la aplicación queda a la espera de que el usuario tome la acción correspondiente que, en este caso, es la activación del comando de voz «Pick» para indicar una posición en el espacio.

La posición indicada debe de ser accesible para el robot, por ello, debe de ser analizada según los cálculos convenientes. Si la coordenada indicada se encuentra dentro del espacio de trabajo del robot colaborativo, se publica en el topic de ROS, en el caso contrario, se proyecta el diálogo de advertencia.

Cuando se haya publicado la primera coordenada, se da lugar a la introducción del comando de voz «Place» para publicar una segunda coordenada en el topic correspondiente. La manera de proceder es la misma que para la primera coordenada. En el momento en el que se publique la posición Place, aparecerá en las lentes el diálogo Repeat Dialog que permite realizar un nuevo Pick and Place o realizar esta manobra en bucle.

El diagrama de flujo mostrado en la Ilustración 47 representa la serie de pasos que hay que seguir para navegar en el modo Pick & Place.

Scan.

Conviene mencionar que una de las opciones que se pueden elegir dentro del menú digital es la de volver a escanear los códigos QR de la mesa. Cuando se active este modo, aparece el diálogo QRCodePanel y se puede volver a definir el sistema de referencia siguiendo las instrucciones de navegación realizadas anteriormente.

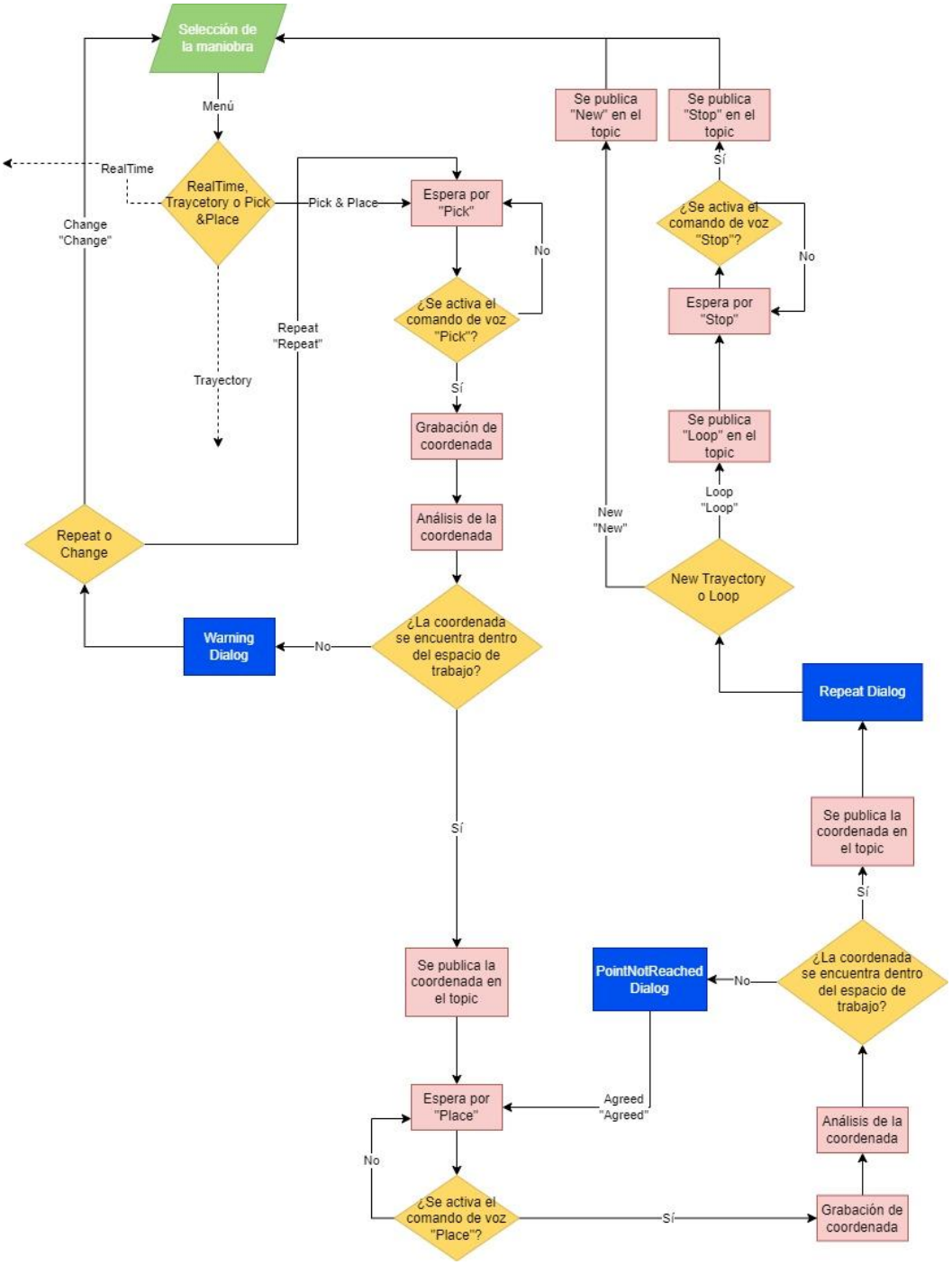


Ilustración 47. Diagrama de flujo del modo Pick & Place

3.3.5 Programación en Unity

En línea con el apartado anterior, conviene describir el código que hace posible el correcto funcionamiento de la aplicación, mencionando los cálculos y funciones de mayor interés en el programa.

Se comienza por mencionar cómo se establece la conexión con ROS y el procedimiento de transmisión de datos entre los dos lados de la comunicación.

Posteriormente, se describe los algoritmos y cálculos diseñados para transformar las coordenadas indicadas con el dispositivo de realidad mixta al lenguaje nativo del robot, haciendo referencia a las funciones que permiten escanear los códigos QR.

Por último, se procede a analizar el código encargado de hacer que el usuario pueda navegar por la aplicación y llevar a cabo las operaciones que desee con los objetos digitales proyectados en las lentes del dispositivo.

3.3.5.1 Comunicación con ROS

Unity3D pertenece a la plataforma .NET y utiliza el lenguaje de programación C# para desarrollar aplicaciones para la plataforma universal de Windows (UWP).

La conexión y posterior comunicación entre aplicaciones .NET y ROS requiere incorporar a los activos del proyecto la biblioteca ROS#. Las aplicaciones .NET permiten la creación y ejecución de servicios web y aplicaciones de Internet. [48] Se debe de incorporar esta biblioteca a los activos del proyecto, moviéndolo a la carpeta Assets o bien importándolo desde la propia interfaz de Unity.

Entre los scripts incluidos en este paquete se encuentra uno llamado "*RosConnector*" que se encarga de realizar la conexión con ROS vía socket TCP/IP. Este script crea el cliente de socket y tiene como entradas públicas la dirección IPv4 y puerto del host remoto donde se ejecuta el servidor de socket. Este script se agrega como componente a un objeto vacío, es decir, un objeto invisible sin propiedades ni componentes, pero que permanece activo durante la ejecución de la aplicación.

Una vez agregado este componente, se debe de introducir la dirección IP y puerto del host remoto. En este caso, el servidor se ejecuta en una máquina ROS dentro de un contenedor de Docker con el puerto 9090 asignado al puerto 9090 del localhost.

Como se aprecia en la Ilustración 48, el tiempo que se permanece a la espera de conexión con el servidor será de 10 segundos, la característica de serialización será de tipo JSON y, el protocolo de comunicación será mediante Web Socket Sharp.

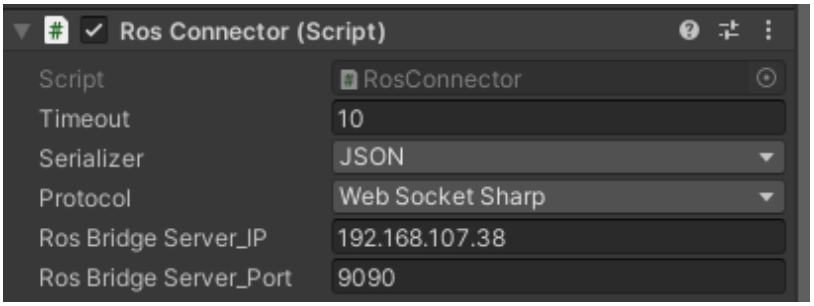


Ilustración 48. Parámetros del cliente de socket TCP/IP entre Unity y ROS

De esta manera, cuando se inicie la aplicación, la primera acción que se realizará será la de establecer la conexión con la máquina de ROS. En este momento, la aplicación se convertirá en un nodo publicador/suscriptor del sistema de ROS.

Los topics que publica este nodo se caracterizan por el prefijo "/HL" (de Hololens) y están destinados a publicar mensajes *std_msgs/String.msg* y *geometry_msgs/Vector3.msg.* La lista de topics que publica este nodo se enumeran en la *Tabla 11* junto a una breve descripción del propósito que cumplen dentro del sistema.

Topic	Objetivo
/HLstream	Publicar coordenadas del dedo índice en el modo Real Time
/HLtray	Publicar coordenadas del dedo índice en el modo Trayectory
/HLpick	Publicar coordenadas del dedo índice en el modo Pick & Place
/HLplace	Publicar coordenadas del dedo índice en el modo Pick & Place
/HLhome	Publicar una cadena de caracteres cuando se pulsa el botón "HOME" de
/HLrepose	Publicar una cadena de caracteres cuando se pulsa el botón "REPOSE"
/HLmove	Publicar una cadena de caracteres cuando se elige entre MoveL o MoveJ en el modo Trayectory
/HLnew	Publicar una cadena de caracteres cuando se pulsa el botón "New" del diálogo Repeat Dialog
/HLloop	Publicar una cadena de caracteres cuando se pulsa el botón Repeat del diálogo Repeat Dialog
/HLstop	Publicar una cadena de caracteres cuando se activa «Stop»

Tabla 10. Lista de topics publicados por el nodo "UR5e Controller"

Cada topic publicado se programa mediante un script agregado como componente al objeto RosConnector y mantiene una estructura similar que varía según el tipo de mensaje transmitido. Un ejemplo del código necesario para crear un topic en Unity es el siguiente:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class PickPublisher :
RosSharp.RosBridgeClient.Publisher<RosSharp.RosBridgeClient.Messages.Geometry.Vector3>
{
    private RosSharp.RosBridgeClient. Messages.Geometry.Vector3 message;
    void Start()
    {
        base.Start();
        message = new RosSharp.RosBridgeClient.Messages.Geometry.Vector3();
    }
    public void pickRobot(Vector3 pickPoint)
    {
        message.x = pickPoint.x; message.y = pickPoint.y; message.z = pickPoint.z;
        Publish(message);
    }
}
```

Este código corresponde al del script *"Pick Publisher"*, encargado de publicar las coordenadas indicadas por el usuario a la hora de realizar la maniobra Pick en el modo Pick & Place. Para poder conseguir este propósito se debe de añadir este script como atributo de los scripts que deseen publicar información en este tema.

El nombre del tema se establece al crear la clase PickPublisher como "hijo" de la clase RosSharp.RosBridgeClient.Publisher encargada de proporcionar la interfaz con Unity.

```
using UnityEngine;
namespace RosSharp.RosBridgeClient
{
    [RequireComponent(typeof(RosConnector))]
    public abstract class Publisher<T> : MonoBehaviour where T: Message
    {
        public string Topic; private string publicationId;
        protected virtual void Start()
        {
            publicationId = GetComponent<RosConnector>().RosSocket.Advertise<T>(Topic);
        }
        protected void Publish(T message)
        {
            GetComponent<RosConnector>().RosSocket.Publish(publicationId, message);
        }
    }
}
```

La variable pública *Topic* corresponde a una cadena de caracteres introducida en la interfaz de Unity por el usuario.

Por otro lado, el nodo también se suscribirá a un topic denominado "/robot_coord" que publicará tres mensajes geometry_msgs/Vector3. Estos mensajes corresponderán a las distancias relativas de los códigos QR respecto del sistema de referencia del robot colaborativo.

3.3.5.2 Definición del sistema de coordenadas mediante códigos QR

La definición de un sistema de coordenadas es esencial para transformar las coordenadas capturadas en la referencia de HL2 a la referencia del robot colaborativo. El método empleado utiliza de tres códigos QR con el fin de construir las matrices de transformación necesarias para cumplir este propósito.

Se disponen tres códigos QR sobre la mesa del trabajo del UR5e que son escaneados por la cámara del HL2, tal como se aprecia en la Ilustración 49.

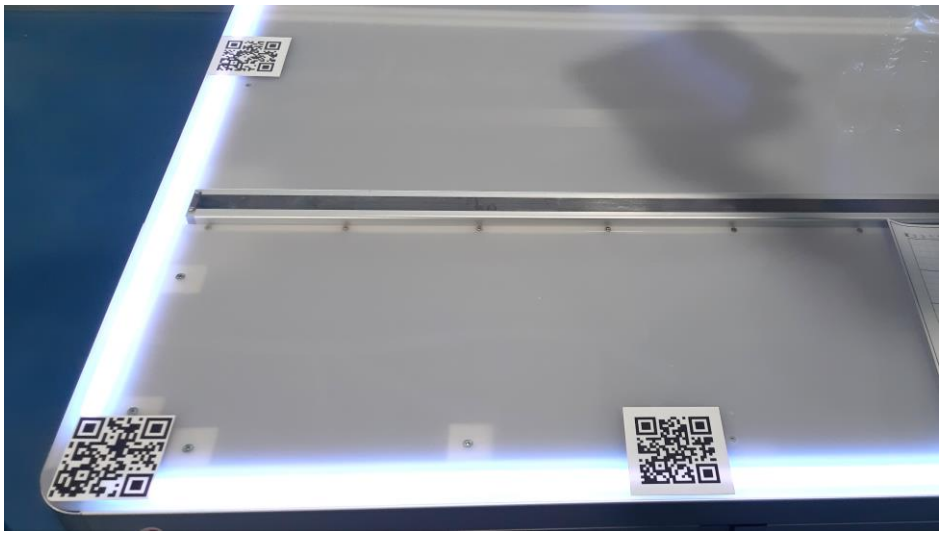


Ilustración 49. Disposición de los códigos QR sobre la mesa de trabajo del robot

Las coordenadas en la referencia del HL2 de una de las esquinas de los cuadrados del código QR, más en concreto el del cuadrado que está alineado con los otros dos, son capturadas y almacenadas en variables dentro del programa. Se utilizan estas tres coordenadas para construir la matriz de transformación homogénea de la mesa de trabajo respecto de HL2.

El círculo rojo adjunto en la Ilustración 50 señala esquina del código QR que ubica su posición en la referencia de HL2.

$$ {}^{H}T_{M} = \begin{pmatrix} Ux & Vx & Wx & Ox \\ Uy & Vy & Wy & Oy \\ Uz & Vz & Wz & Oz \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1) $$

Ilustración 50. Esquina del código QR capturada por HL2

De esta manera, dadas tres coordenadas P1(x1,y1,z1), P2(x2,y2,z2) y P3(x3,y3,z3), se calculan las coordenadas U,V y W del sistema de referencia mesa mediante el siguiente procedimiento:

En primer lugar, se calcula la diferencia entre los puntos P1 y P2:

$$\overline{A} = \overline{P1} - \overline{P2} = (x1 - x2, y1 - y2, z1 - z2) = (Ax, Ay, Az) \quad (2)$$

Después, se normaliza este vector:

$$|\overline{U}| = \frac{(Ax, Ay, Az)}{\sqrt{Ax^2 + Ay^2 + Az^2}} = (Ux, Uy, Uz) \quad (3)$$

Y se calcula también la diferencia entre los puntos P4 y P1:

$$\overline{D} = \overline{P3} - \overline{P1} = (x3 - x1, y3 - y1, z3 - z1) = (Dx, Dy, Dz) \quad (4)$$

Para obtener un vector perpendicular a ambos, se realiza su producto vectorial. Este vector se denominará B:

$$\overline{B} = |\overline{U} \times \overline{D}| = (Bx, By, Bz) \quad (5)$$

Que también se hace unitario:

$$|\overline{W}| = \frac{(Bx, By, Bz)}{\sqrt{Bx^2 + By^2 + Bz^2}} = (Wx, Wy, Wz) \quad (6)$$

De esta manera, se obtiene el vector Y mediante el producto vectorial entre W y U:

$$\overline{V} = |\overline{W} \times \overline{U}| = (Vx, Vy, Vz) \quad (7)$$

Y el origen:

$$\overline{O} = \overline{P1} = (Ox, Oy, Oz) \tag{8}$$

Se aplican estos mismos cálculos para construir la matriz de transformación homogénea de la mesa respecto del robot (${}^{R}T_{M}$). Para ello, solo hay que sustituir las variables P1, P2 y P3 por las coordenadas de los mismos puntos de los códigos QR tomados en la referencia del robot.

Cabe destacar que para calcular estas coordenadas se ha hecho uso de la Flex Pendant para llevar el TCP del robot a la posición deseada y, así observar las coordenadas mostradas por pantalla. Una vez se realice este proceso, se pueden transformar las coordenadas en la referencia de HL2 a la del robot y viceversa. En la Ilustración 51 se muestra un esquema de como habría que realizar las operaciones para pasar de un sistema de referencia a otro.

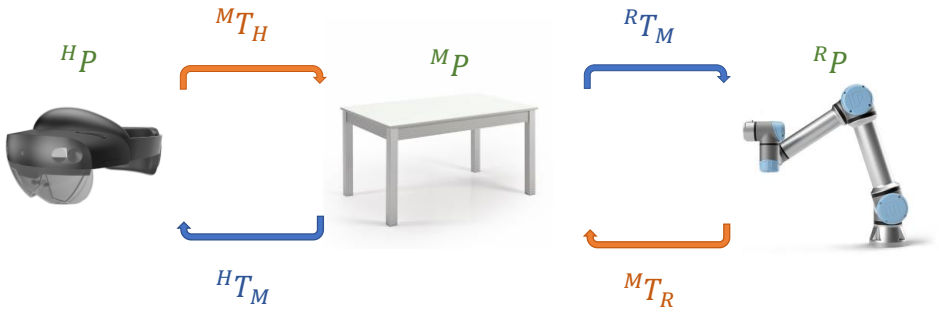


Ilustración 51. Esquema de las transformaciones necesarias para cambiar de referencia

De esta manera, dado un punto P (Hx, Hy, Hz,1) en la referencia de HL2 (${}^{H}P$), se transformará de la siguiente manera para calcularlo en la base del robot:

- Como se conocen las componentes de la matriz ${}^{H}T_{M}$, habrá que calcular su inversa y multiplicarlo por el punto P para llevarlo a la base de la mesa. En primer lugar, se transforma este punto a la referencia de la mesa.

$${}^{M}P = {}^{M}T_{H} \times {}^{H}P = (Mx, My, Mz, 1) \tag{9}$$

- Siendo:

$${}^{M}T_{H} = [{}^{H}T_{M}]^{-1} \tag{10}$$

- Y transformamos este punto a la referencia del robot mediante:

$${}^{R}P = {}^{R}T_{m} \times {}^{M}P = (Rx, Ry, Rz, 1) \tag{11}$$

Para calcular las coordenadas de un punto en la referencia de HL2 partiendo de un punto en la referencia del robot habría que realizar el cálculo inverso.

El escaneo de los códigos QR es posible incorporando NuGetForUnity que es un cliente NuGet creado desde cero para ejecutarse dentro de Unity Editor. NuGet es un sistema de gestión de paquetes que facilita la creación de paquetes que se distribuyen en un servidor y son consumidos por los usuarios. [49]

Mediante NuGetForUnity se debe instalar Microsoft.MixedReality.QR, que es un paquete que incorpora las dependencias y funcionalidades necesarias para crear programas capaces de realizar detección y lectura de los códigos QR.

Una vez instalado el paquete, se crea un objeto vacío, que en este caso se llama QRCodesManager, al que se agrega los scripts *"QRCodesManager"* y *"QRCodesVisualizer"*. Estos scripts gestionan la detección y lectura de los códigos QR. Además, se pretende cumplir el propósito de analizar la información contenida en el código QR para determinar si es válido o no.

Por ello, los códigos QR de la mesa de trabajo contienen un mensaje que identifica el orden con el que fueron medidos respecto del sistema de referencia del robot. Un algoritmo del programa se encarga de clasificar las coordenadas de los códigos QR según la estructura de datos que se recibe al realizar la detección.

En el script QRCodesVisualizer se han programado las siguientes funciones:

- **StartScan():** Inicializa la detección de códigos QR al presionar el botón Scan del QRCodePanel.
- **HandleInputs():** Cuando se detecta un código QR se llama a esta función. En ella, se almacena la información del código QR capturado y proyecta sobre el diálogo un mensaje con la información contenida dentro de este.
- **StopScan():** Almacena la coordenada de cada código QR escaneado y oculta el objeto StepHandlerPanel.

3.3.5.3 Navegación y programación de las maniobras

El código que compone la navegación y la programación de las maniobras en los modos RealTime, Trayectory y Pick & Place y de las demás operaciones que permite al usuario enviar información a ROS, viene expuesto en el script *"Main"*. Este script viene agregado a un objeto vacío llamado Manager como componente de este y permanece activo durante toda la ejecución del programa. A este objeto, también es agregado el script SpeechInputHandle.

Para comprender el desarrollo de este código, se mencionan y describen las funciones programadas en orden cronológico desde el inicio de la aplicación.

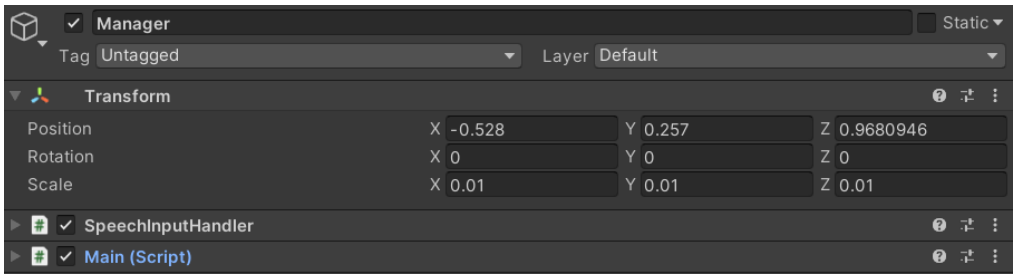


Ilustración 52. Componentes del objeto Manager

Presentación y definición del sistema de referencia

Cada objeto Dialog viene definido por una estructura de objetos que permiten modificar ciertos atributos de los diálogos como el texto que los compone, el aspecto de los botones, la posición en el espacio del objeto, etc.

En la Ilustración 53 se aprecia un claro ejemplo de la estructura de objetos que componen el dialogo de bienvenida de la aplicación. Se denominarán a estos objetos dentro del objeto Welcome Dialog como *"objetos hijos"*, ya que heredan todos los atributos y funciones de éste.

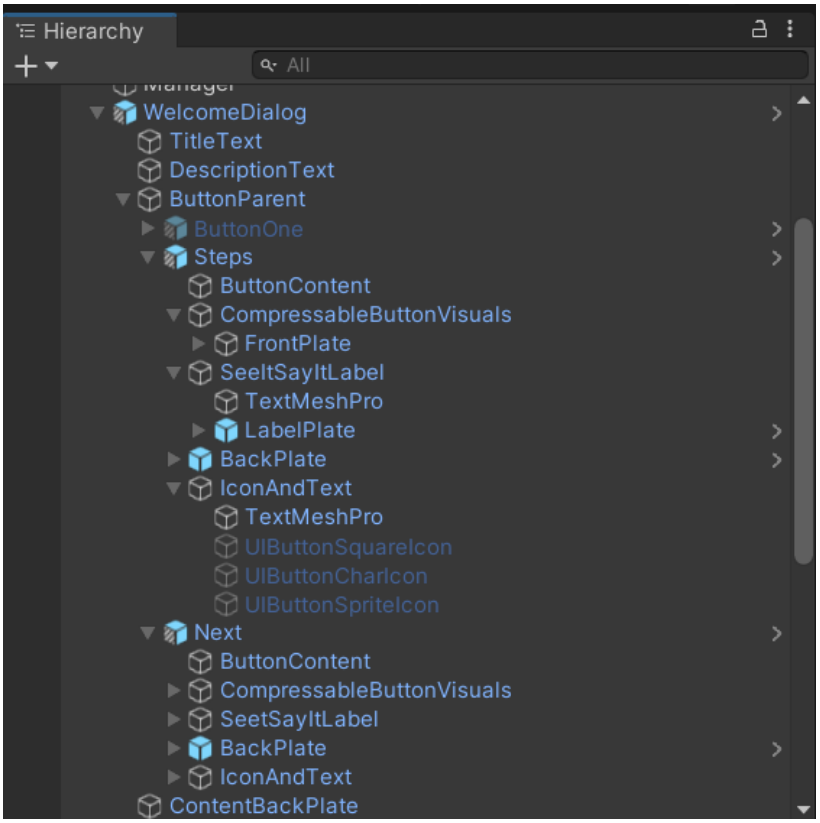


Ilustración 53. Jerarquía del objeto "Welcome Dialog"

Los objetos hijo de un objeto diálogo son los siguientes:

- **TitleText:** Permite introducir un título para el texto del diálogo y modificar su aspecto como se desee.
- **DescriptionText:** Permite introducir un texto más extenso que el del título y modificarlo como se desee.
- **ButtonParent:** Colección de botones. Por defecto, el prefab Dialog incluye tres botones, pero en todos los casos se han utilizado uno o dos.
    - **Steps y Next:** Son los botones que aparecen en el diálogo e incluyen los componentes encargados de procesar las pulsaciones del usuario con el dedo índice.
- **ContentBackPlate:** Compone el fondo del diálogo.

El objeto SeetSayItLabel es un mini panel que aparece debajo de los botones del diálogo para indicar al usuario que comando de voz debe de utilizar para activar la acción que realiza cada botón al pulsarlo manualmente.

La secuencia de navegación entre diálogos se realiza llamando a funciones dentro del script Main al activar los botones de los diálogos. Por ejemplo, al activar el botón Next, se llama a la función pushNext(), introducida dentro del script principal, que oculta el objeto Welcome Dialog y proyecta Referential Dialog.

Dentro del componente "Interactable" del objeto "Next" se encuentra una función llamada OnClicked(). En esta función se agrega el objeto con el que se quiere interactuar y la función a la que se llama. En la Ilustración 54, se muestra cómo se agrega el objeto Manager y se llama a la función pushNext() del componente Main.

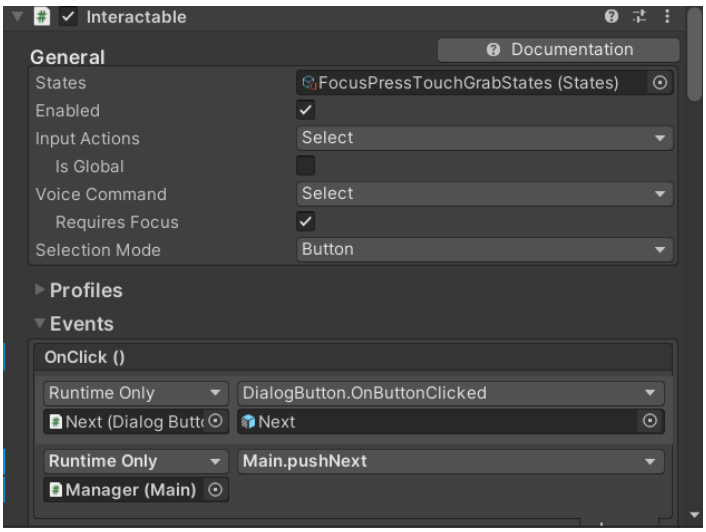


Ilustración 54. Llamada a la función pushNext() desde el objeto Next

En el código de la función se utiliza el método SetActive() para desactivar el objeto Welcome Dialog y activar Referentia Dialog. Esta función también es llamada al activar el comando de voz «Next».

```
public void pushNext()
    {
        if(welcomeDialog.activeSelf)
            referentialDialog.SetActive(true); welcomeDialog.SetActive(false);
        else
            Debug.Log("Inaccessible voice command");
    }
```

Se utiliza el objeto SpeechInputHandle para definir el comando de voz que se quiere programar y las acciones que se realizarán en el programa al activarlo. De igual forma, se agrega el objeto Manager y se introduce la llamada a la función pushNext() del script Main, tal como se aprecia en la Ilustración 55.

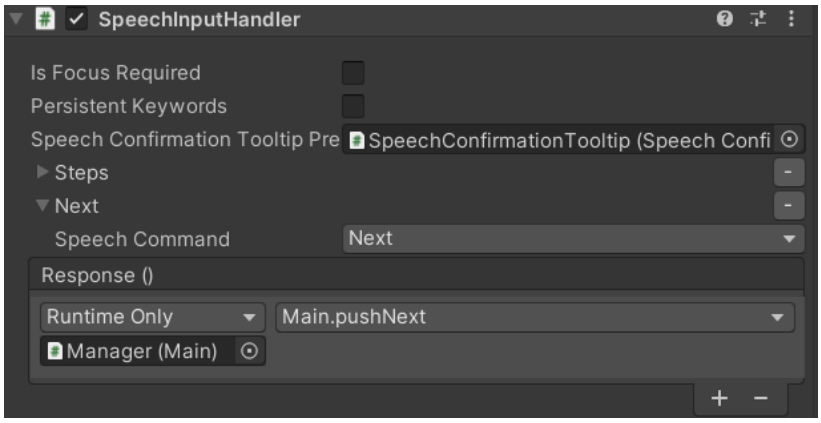


Ilustración 55. Llamada a la función pushNext() con el comando de voz Next

En el caso de los botones del diálogo QRCodePanel las funciones llamadas están introducidas en el script QRCodesVisualizer.

El botón StartScanButton, denominado como Scan, llama a la función QRCodesVisualizer.StartScan() la primera vez que se pulsa, oculta este botón y proyectará el botón StopScanButton. Al pulsar este botón, se detiene la detección de códigos QR al llamar a la función QRCodesVisualizer.StopScan(), se vuelve a activar el objeto StartScanButton y se oculta StopScanButton.

Las funciones a las que llama cada botón y comandos de voz vienen descritas en la Tabla 11. Cabe destacar que no todos los comandos de voz vienen descritos en esta tabla, ya que esta solo recoge los botones y comandos de voz que son utilizados para navegar entre los diálogos.

Las funciones expuestas en esta tabla tienen el objetivo principal de activar y desactivar diálogos. Sin embargo, esto no quiere decir que algunas funciones no cumplan otros objetivos.

Botón /Comando de voz	Objetivo	Llamada a la función
Steps/Steps	Ocultar Welcome Dialog y proyectar Steps Dialog	Main.pushSteps()
Next/Next	Ocultar Welcome Dialog y proyecta rReferential Dialog	Main.pushNext()
Skip/Skip	Ocultar Steps Dialog y proyectar Welcome Dialog	Main.pushSkip()
Continue/ Continue	Ocultar Steps Dialog y proyectar ROS Dialog	Main.pushContinue()
Back/Back	Ocultar ROS Dialog y proyectar Steps Dialog	Main.pushBack()
OK/OK	Ocultar ROS Dialog y proyectar Referential Dialog	Main.pushOK()
SetOrigin/ Set Origin	Ocultar Referential Dialog y proyectar QRCodePanel	Main.pushSetOrigin()
More/More	Ocultar More Dialog y proyectar Info Dialog	Main.pushMore()
Begin/Begin	Ocultar Info Dialog y proyectar el menú y la botonera	Main.pushBegin()
Repeat/Repeat	Ocultar Warning Dialog y cambiar el color del espacio de trabajo a verde	Main.pushRepeat()
Change/Change	Ocultar Warning Dialog , proyectar el menú y la botonera y cambiar el color del espacio de trabajo a verde	Main.pushChange()
Agreed/Agreed	Ocultar PointNotReached Dialog y cambiar el color del espacio de trabajo a verde	Main.pushAgreed()
New / New	Ocultar Repeat Dialog y proyectar el menú y la botonera	Main.pushNew()
Loop/Loop	Ocultar Repeat Dialog	Main.pushLoop()
MoveL/Line	Ocultar MoveLorMoveJ Dialog y proyectar Repeat Dialog	Main.pushMoveL()
MoveJ/Joint	Ocultar MoveLorMoveJ Dialog" proyectar Repeat Dialog	Main.pushMoveJ()

Tabla 11. Funciones llamadas en la navegación entre diálogos

Cabe destacar que a la hora de llamar a la función StopScan(), a su vez se realiza una llamada a la función SavePosition() del script Main. Esto se realiza

con el propósito de mandar las coordenadas de los códigos QR detectados del script QRCodesVisualizer al script Main y, de esta manera, almacenar esta información en variables de la clase Main.

La función SavePosition() recoge las coordenadas de los códigos QR tomadas con la cámara del HL2 y realiza con estas los cálculos necesarios para construir las matrices de transformación para pasar coordenadas de un sistema de referencia a otro. Por ello, hace dos llamadas a la función calculate_matrix() pasándola como argumentos las tres posiciones de los códigos QR en la referencia de HL2 y del robot respectivamente.

```
//Create rotation matrix
rotationMatrix = calculate_matrix(p1, p2, p3 2));
rotationMatrix2 = calculate_matrix(OP, OQ, OR);
```

En esta función, se utilizan estas variables para realizar los cálculos necesarios para construir la matriz de transformación homogénea de la mesa respecto de HL2 (${}^{H}T_{M}$) y respecto del robot (${}^{R}T_{M}$).

Las variables p1, p2 y p3 corresponden a las coordenadas de los códigos QR en la referencia de Hololens 2 y OP, OQ y OR las mismas coordenadas, pero en la referencia del robot.

```
private Matrix4x4 calculate_matrix(Vector3 P1, Vector3 P2, Vector3 P3)
    {
        Vector3 A, U, D, B, W, V, origin;
        Matrix4x4 matrix=new Matrix4x4();
        //Equation 2)
        A = (P1 - P2);
        //Equation 3)
        U = A.normalized;
        //Equation 4)
        D = (P3 - P1);
        //Equation 5)
        B = Vector3.Cross(U, D);
        //Equation 6)
        W = B.normalized;
        //Equation 7)
        V = Vector3.Cross(W, U);
        //Equation 8)
        origin = P1;
        //Equation 1)
        matrix.SetRow(0, new Vector4(U.x, V.x, W.x, origin.x));
        matrix.SetRow(1, new Vector4(U.y, V.y, W.y, origin.y));
        matrix.SetRow(2, new Vector4(U.z, V.z, W.z, origin.z));
        matrix.SetRow(3, new Vector4(0, 0, 0, 1));
        return matrix;
    }
```

En el código anterior se puede apreciar como las ecuaciones definidas anteriormente son convertidas al lenguaje C# para construir una matriz de

cuatro filas y cuatro columnas que represente la matriz de transformación homogénea para pasar de una referencia a otra.

Por último, al pulsar el botón Stop, se llama a la función Main.ClosePanel() que gestiona el número de códigos QR detectados, ocultando y activando los diálogos que corresponden en un orden correcto. En la Tabla 12 se encuentran resumidos las funciones que son llamadas al interactuar con los botones del QRCodePanel.

Objeto	Botón	Función a la que llama	Objetivo
QRCodePanel	StartScanButton	QRCodesVisualizer.StartScan()	Iniciar la detección de códigos QR
	StopScanButton	QRCodesVisualizer.StopScan()	Detener la detección de códigos QR y pasar las coordenadas almacenadas al Main a partir de la función SavePosition()
	CloseButton	Main.ClosePanel()	Cerrar QRCodePanel y activar InfoDialog

Tabla 12. Funciones llamadas por los botones del diálogo QRPanel

Cuando ya se hayan definido las matrices de transformación homogénea, se da paso a ubicar el origen del espacio de trabajo robot en el entorno de HL2 dibujando una pequeña esfera de color gris en dicha posición. Esta coordenada se almacena en la variable de la clase “center_space_robot”.

Es preciso recordar que el centro del espacio de trabajo del robot se sitúa a una distancia de 163 milímetros en la dirección positiva del eje Z. Para ello, se ha definido una función llamada “robot2hololens” que realiza el cálculo necesario para transformar una coordenada en la referencia del robot a coordenada en la referencia de HL2, con el procedimiento descrito anteriormente. El código que compone esta función es el siguiente:

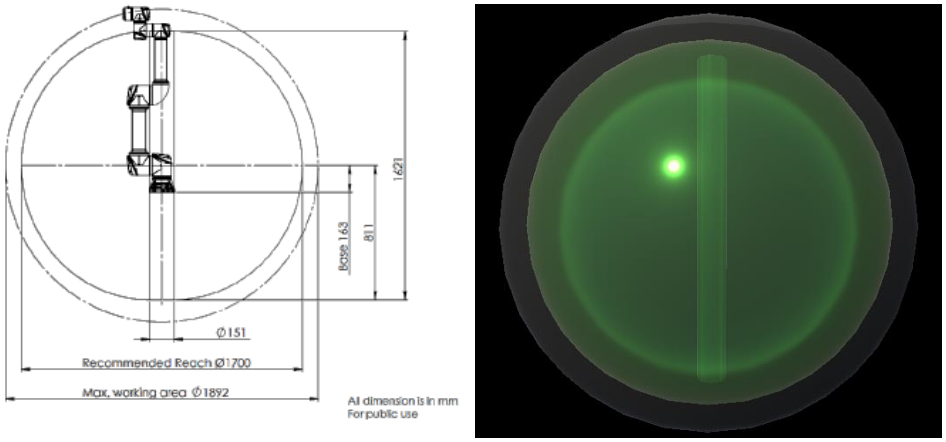
```
private Vector3 robot2hololens (Vector3 point_robot)
    {
        Vector4 point_in_hololens,point_in_mesa;
        point_in_mesa = rotationMatrix2.inverse * new   Vector4(point_robot.x,
point_robot.y, point_robot.z, 1);
        point_in_hololens = rotationMatrix * point_in_mesa;
        return new Vector3(point_in_hololens.x,point_in_hololens.y,point_in_hololens.z);
    }
```

Como se puede apreciar, esta función recibe como argumento una coordenada en la referencia del robot y es multiplicada por la inversa de la matriz de transformación homogénea de la mesa respecto del robot para transformarla a la referencia de la mesa.

Posteriormente, se multiplica el resultado por la matriz de transformación homogénea de la mesa respecto de HL2 para transformarla a la referencia del dispositivo de realidad mixta.

Por último, se dibuja el espacio de trabajo del robot ubicado en la posición de "center_space_robot". Por ello, se llama a la función DrawWorkspace() que crea un cilindro y una esfera, ambas de color verde, con las dimensiones de los límites del UR5e. Estas dimensiones vienen descritas en Ilustración 21.

En la Ilustración 56 se muestra una comparativa entre las dimensiones del espacio de trabajo del robot colaborativo UR5e y el objeto digital diseñado en el entorno de Unity. El propósito de esto es representar el espacio de trabajo real del robot para poder dotar a la aplicación de una manera fácil y precisa de informar al usuario de los límites del UR5e.



a) Diagrama del espacio de trabajo del UR5e b) Espacio de trabajo digital visto en Unity

Ilustración 56. Diseño del espacio de trabajo digital del UR5e.

Botonera digital

En el momento en el que el usuario pulsa en el botón Begin del diálogo More Dialog o activa el comando de voz correspondiente que realiza la misma

acción, llega el momento de elegir la maniobra que envíe información a ROS, bien interactuando con la botonera digital o con los modos de actividad del menú.

La botonera contiene tres botones con propósitos diferentes. Solo los botones Home y Repose envían mensajes a topics, mientras que el botón Help solo proyecta el panel Instructions en las lentes del HL2.

Las funciones que son llamadas a la hora de activar los botones correspondientes del objeto Buttons vienen recopiladas en la Tabla 13.

Botón	Función a la que llama	Objetivo
Home	HomePublisher.goHome()	Pasar como argumento la cadena de caracteres “go_home” que es publicada en el topic de ROS /HLhome
Repose	ReposePublisher.goRepose()	Pasar como argumento la cadena de caracteres “go_repose” que será publicada en el topic de ROS /HLrepose
Help	Main.pushHelp	Pasar como argumento el object Instructions para activarlo/desactivarlo de la escena

Tabla 13. Funciones llamadas por los botones de la botonera digital

RealTime Mode

Programar este modo de actividad supone definir un periodo de muestreo óptimo para recoger los datos de las coordenadas que se están indicando con el dedo índice de la mano derecha y compararlas constantemente para decidir si la distancia que las separa es suficiente como para enviarlas al UR5e.

En el momento de pulsar el botón RealTime del menú digital, aparece encima de este el panel RealTimeInstructions que llama a la función Main.streamMODE(). Además, se pone una variable bandera, denominada “streamON” a verdadera. Esta variable bandera se está continuamente analizando en la función Update(), que es llamada en cada fotograma y, permite introducir los comandos de voz correspondientes a este modo.

La grabación comienza al activar el comando de voz «Start» y, se detiene con «Stop». Cuando se pronuncian estos comandos de voz, se ocultan los elementos digitales de la escena y se llama a la función Main.FollowON(). A

ella, se le pasa como argumento una variable bandera con valor verdadero en el caso de que el comando de voz activado sea «Start» y falso en el caso de que sea «Stop».

En el primer caso, se inicializa a cero una variable llamada “timer” y se pone a verdadero la variable de la clase “startFollow” que abre paso a la grabación y análisis de las coordenadas.

La variable “timer” sirve para llevar la cuenta del tiempo transcurrido entre coordenadas analizadas. Por defecto, el periodo de muestreo que contempla Unity es de 20 milisegundos. Sin embargo, este intervalo resulta excesivo a la hora de comparar posiciones en el espacio, ya que supone tener que realizar trayectorias bastante rápidas para notar una diferencia considerable entre las distancias que separan unas coordenadas respecto de otras.

Tras diversos ensayos, se ha establecido que el tiempo ideal para realizar la comparación entre coordenadas es de 200 milisegundos (ms). A partir de este instante, la aplicación contempla que cada vez que se detecte el dedo índice de la mano derecha en el campo de visión del HL2, se analicen sus coordenadas.

Cada vez que se grabe una coordenada, se transforma a la referencia del robot mediante el uso de la función “hololens2robot()”. El resultado se asigna a la variable de la clase “relativePosition”.

De esta manera, se puede entender que el dedo índice del usuario hace el papel del TCP del robot colaborativo y las coordenadas que se están enviando corresponden a la distancia relativa de este respecto de la base del UR5e.

Si el usuario indica una coordenada que se encuentra fuera de los límites del robot colaborativo, se cambia el color del espacio de trabajo a rojo, se proyecta el diálogo Warning Dialog y se envía la cadena de caracteres “Stop” al topic /HLstop con el objetivo simular el comando de voz «Stop». En el caso contrario, la coordenada se publica en ROS.

En este último caso, el usuario debe de activar el comando de voz «Stop» para publicar un mensaje en el topic al que está suscrito la función que se encarga de enviar un URScript al robot colaborativo con las instrucciones necesarias para detener su movimiento.

La aplicación de realidad mixta proyecta esferas de color amarillo en las coordenadas publicadas en el topic /HLstream. Estas esferas desaparecen en cuanto el usuario activa el comando de voz «Stop».

En la Tabla 14 se resumen las funciones llamadas al interactuar con los botones y comandos de voz del modo Real Time.

Botón / «Comando de voz»	Función a la que llama	Objetivo
RealTime (menú digital)	Main.stramMODE()	Pasar como argumento *true* y proyectar RealTimeInstructions. Permitir la introducción de los comandos de voz «Start» y «Stop»
«Start»	Main.FollowON()	Pasar como argumento *true* y poner la variable timer a cero. Permitir la grabación de las coordenadas del dedo índice. Publicar las coordenadas válidas en el topic /HLstream
«Stop»	Main.FollowON()	Pasar como argumento *false* y poner la variable timer a cero. Detener la grabación de las coordenadas del dedo índice. Publicar un mensaje "Stop" en el topic /HLstop

Tabla 14. Funciones llamadas durante el modo Real Time

Trayectory Mode

El modo trayectoria se activa cuando se pulsa el botón Trayectory Mode del menú digital. Al realizar esta acción, se llama a la función Main.trayectoryMODE(), pasándola como argumento *true*. De esta manera, se proyecta el panel TrayectoryInstructions encima del menú.

El valor del argumento que se pasa a esta función se asigna a la variable bandera "trayON" que permite realizar acciones con los comandos de voz correspondientes al modo Trayectory. Una vez son almacenadas todas las coordenadas que el usuario ha indicado, estas se publican en el topic /HLtray.

Cabe destacar que para que estas coordenadas se publiquen deben de encontrarse dentro de los límites del espacio de trabajo del robot. Si esto no ocurre, son desestimadas.

Las coordenadas en el espacio son indicadas mediante el comando de voz «Set Point». La manera de proceder es indicar con el dedo índice la posición que se desea enviar y, después, pronunciar en alto «Set Point». Sucesivamente, se llama a la función Main.savePoint() que realiza todo este procedimiento.

Esta función a su vez llama a la función Main.DrawPoints() que proyecta esferas de un centímetro de diámetro y color dorado en la posición guardada.

Botón / «Comando de voz»	Función a la que llama	Objetivo
Trayectory (menú digital)	Main.trayectoryMODE()	Pasar como argumento *true* y proyecta Trayectory Instructions Permitir la introducción de los comandos de voz «SetPoint», «Cancel» y «Go»
«Set Point»	Main.savePoint()	Almacenar y analizar la coordenada del dedo índice en una lista de vectores. Proyectar esferas de pequeño tamaño en las posiciones indicadas
«Cancel»	Main.cancelTrayectory()	Vaciar la lista de vectores y borrar las esferas.
«Go»	Main.sendTrayectory()	Publicar la lista de vectores en el topic /HLtray y proyectar MoveLorMoveJ
MoveL / «Line»	Main.pushMoveL()	Publicar el mensaje “MoveL” en el topic /HLmove y proyectar Repeat
MoveJ / «Joint»	Main.pushMoveJ()	Publicar el mensaje “MoveJ” en el topic /HLmove y proyectar Repeat
New / «New»	Main.pushNew()	Ocultar Repeat, vaciar la lista de vectores, borrar las esferas y publicar el mensaje “New” en el topic /HLnew
Loop / «Loop»	Main.pushLoop()	Ocultar Repeat y publicar el mensaje “Loop” en el topic /HLloop
«Stop»	Main.FollowON()	Publicar la cadena de caracteres “Stop” en el topic /HLstop

Tabla 15. Funciones llamadas durante el modo Trayectory

Cuando se hayan indicado todos los puntos de la trayectoria, se debe de pronunciar «Go». En ese momento, se llama a la función Main.sendTrayectory() que proyecta el diálogo MoveLorMoveJ en las lentes del operario y publica los puntos transformados en la referencia del robot. En

este diálogo, si se selecciona MoveL, se publica el mensaje “MoveL” en el topic /HLmove, si se selecciona MoveJ, se publica “MoveJ”.

Por otro lado, si se desea cancelar la trayectoria programada se debe de pronunciar «Cancel». De esta manera, se vaciará la lista de vectores, se borrarán las esferas que se hayan dibujado y se podrán volver a indicar nuevos puntos en el espacio. Este propósito lo realiza la función Main.cancelTrayectory().

En el momento de finalizar la programación de una trayectoria, se da la posibilidad al usuario de trazar una nueva o de repetirla indefinidamente hasta que active el comando de voz «Stop». Esta decisión se toma pulsando uno de los dos botones del diálogo Repeat.

Si se indica Loop, se llama a la función Main.pushLoop() para publicar la cadena de caracteres “Loop” en el topic /HLloop. Entonces, el robot comenzará a realizar la trayectoria programada en bucle y el usuario deberá de activar el comando de voz «Stop» para detener su movimiento.

Si se indica New, se vacía la lista de vectores, se borran las esferas y se publica en el topic /HLnew la cadena de caracteres “New” con el objetivo de reiniciar las variables que se utilicen en el programa traductor. Esta función la realiza pushNew().

Pick & Place

La estructura de esta maniobra es bastante similar al del modo Trayectory. Aquí, la coordenada indicada se publica directamente en el topic correspondiente sin necesidad de almacenarlo en una lista, ya que se pretende que el robot realice la maniobra Pick sin aún haber indicado la coordenada donde se hará el Place.

El procedimiento comienza por pulsar el botón Pick & Place del menú. Al realizar esta acción, se llama a la función Main.PickAndPlaceMode() y se la pasa como argumento *true.* De esta manera, se asigna el valor de la variable bandera “pickMODE” a verdadera.

Ahora, al activar el comando de voz «Pick», se llamará a la función Main.setPick() y se comprobará que, efectivamente, se ha puesto la variable “pickMODE” a verdadero. Después, se pone la variable de la clase “pickON” a verdadera y se incrementa el valor del contador “contPick” con el fin de controlar el uso del comando de voz “Pick”.

Cuando se pronuncia «Pick» por primera vez y se detecta el dedo índice de la mano derecha, se guarda la posición indicada en la referencia de HL2, se transforma a la referencia del robot y se analiza si está dentro de los límites.

Si la coordenada supera este filtro, se publicará en el topic /HLpick y se pondrá la variable "placeON" a verdadero, en el caso contrario, se proyectará el diálogo "PointNotReached" y se reiniciará "contPick".

En este momento, se puede utilizar el comando de voz «Place». La manera de proceder es la misma que con «Pick» con la única condición de haber publicado anteriormente la coordenada de recogida. La coordenada indicada mediante «Place» se publica en el topic /HLplace.

Al finalizar este proceso, se reinicia el contador "contPick", se pone la variable "placeON" a falso y, se llama a la función cleanPoints(). No obstante, se ofrece la posibilidad de volver a trazar una nueva maniobra Pick & Place o repetir la indicada mediante el uso del diálogo Repeat, de la misma forma que en el modo Trayectory.

Botón / «Comando de voz»	Función a la que llama	Objetivo
Pick & Place (menú digital)	Main.pickANDPlaceMODE()	Permitir la introducción de los comandos de voz «Pick» y «Place».
«Pick»	Main.setPick()	Almacenar y analizar la coordenada del dedo índice. Publicar las coordenadas válidas en el topic /HLpick
«Place»	Main.setPlace()	Almacenar y analizar la coordenada del dedo índice. Publicar las coordenadas válidas en el topic /HLplace
New / «New»	Main.pushNew()	Ocultar Repeat, vaciar la lista de vectores, borrar las esferas y publicar el mensaje "New" en el topic /HLnew
Loop / «Loop»	Main.pushLoop()	Ocultar Repeat y publica el mensaje "Loop" en el topic /HLloop
«Stop»	Main.FollowON()	Publicar la cadena de caracteres "Stop" en el topic /HLstop

Tabla 16. Funciones llamadas durante el modo Pick & Place

Scan

Cuando se pulsa el cuarto botón del menú, se llama a la función Main.SystemMovement() que establece el valor de las coordenadas de los códigos QR en la referencia de HL2 a cero e inicia los mecanismos para volver a definir la matriz de transformación homogénea de la mesa respecto de HL2.

3.4 Implementación

Poner en funcionamiento el sistema de realidad mixta desarrollado requiere integrar la aplicación diseñada en Unity en Microsoft Hololens 2.

Se deben de tener en cuenta los aspectos de transmisión y traducción de los datos procedentes del dispositivo de realidad mixta, desarrollando un programa que sea capaz de suscribirse a los topics publicados por la aplicación de HL2 para traducirles al lenguaje nativo del UR5e. De esta manera, se puede interactuar con el robot colaborativo y, a su vez, con el resto de los equipos que abarcan el marco de este proyecto.

3.4.1 Integración de la aplicación de Unity en Microsoft Hololens 2

La integración de la aplicación de realidad mixta desarrollada en el entorno de Unity requiere compilar e implementar la solución generada en Visual Studio.

El archivo de solución tiene la extensión ".sln" y contiene información basada en texto que el entorno usa para buscar y cargar los parámetros de nombre-valor de los datos persistentes y el proyecto VSPackages a los que hace referencia. [50]

La creación de un proyecto de Unity a la plataforma universal de Windows con el backend de secuencias de comandos IL2CPP creará una solución de Visual Studio C++ que contiene tres proyectos: [50]

1. **Il2CppOutputProject** contiene código C++ generado que fue convertido de assemblies managed.
2. **Unity Data Project** contiene todos los archivos de información de Unity: niveles, assets, etc.
3. **El proyecto principal** (coincide con el nombre de su proyecto de Unity). Este es el proyecto que será construido en un paquete de aplicación, que puede ser desplegado a un dispositivo o subido a la Windows Store.

La configuración de Visual Studio realizada para HL2 requiere definir la arquitectura, el modo de lanzamiento y el destino de implementación. Los parámetros elegidos son los siguientes:

- Arquitectura (x64, x86, ARM, ARM64): **ARM64.** Arquitectura propia de HL2.

- Modo de lanzamiento (Debug/Master/MasterWithLTCG/Release): **Release.** Configuración utilizada para perfil el juego.
- Destino de implementación. **Máquina Remota.** Se debe de introducir la dirección IP del HL2.

Una vez la aplicación termina de compilarse, se puede apreciar el menú inicio de Hololens.

3.4.2 Traducción y transmisión de los datos

Con el objetivo de hacer posible la interacción humano-robot, es necesario crear un programa que gestione la entrada y la salida de los datos en el sistema. Este programa tiene que ser ubicado en el entorno de ROS, ya que la transmisión de los datos se realiza mediante el mecanismo de paso de mensajes.

Con la información procedente de la aplicación de realidad mixta, se debe de poder crear instrucciones capaces de ser interpretadas por el UR5e. El fichero URScript generado será capaz de poder llevar a cabo las maniobras programadas con el robot colaborativo.

Es preciso configurar el entorno de ROS en un contenedor de Docker que permita la comunicación vía socket TCP/IP entre ambos dispositivos.

3.4.2.1 Despliegue del contenedor de Docker con la imagen de ROS

Disponer de un programa informático empaquetado en un contenedor de Docker supone diversas ventajas a la hora de desplegar aplicaciones de manera portátil y eficaz.

Como ya se ha comentado, un Dockerfile recoge los comandos necesarios para ensamblar una imagen acoplable a partir de una imagen base. Debido a su simplicidad, crear una imagen personalizada de ROS Melodic con los paquetes y dependencias que necesita el sistema de realidad mixta que abarca este proyecto, resulta una opción adecuada para llevar a cabo la comunicación con HL2 y el robot colaborativo.

Se pretende establecer una conexión vía socket TCP/IP entre la aplicación de realidad mixta y ROS. Para ello, es necesario instalar el paquete “rosbridge-server” en el espacio desarrollo (devel) del directorio de trabajo que se establezca en ROS (catkin_ws). También es necesario instalar el paquete Universal_Robots_ROS_Driver para monitorear el robot colaborativo.

Los puertos expuestos son el 9009 y 30002. Este último es el puerto establecido para los robots colaborativos de Universal Robots destinado a recibir ficheros URScript.

Adicionalmente, se desea instalar el paquete “nano” que es un editor de texto para sistemas Linux y Unix. El documento Dockerfile diseñado es el siguiente:

```
#Base image to start construction
FROM ros:melodic
#Create workspace
WORKDIR /root/catkin_ws
ADD catkin_ws .
#Install packages
RUN apt-get update && \
    apt-get install nano -y && \
    apt-get install ros-melodic-rosbridge-server -y && \
    git clone https://github.com/UniversalRobots/Universal_Robots_ROS_Driver.git
src/Universal_Robots_ROS_Driver && \
    git clone -b melodic-devel https://github.com/ros-industrial/universal_robot.git
src/universal_robot && \
    sudo apt update -qq && \
    rosdep update && \
    rosdep install --from-paths src --ignore-src -y
RUN /bin/bash -c 'source /opt/ros/melodic/setup.bash && catkin_make'
#Default editor
ENV EDITOR=nano
#Exposed ports
EXPOSE 9090 30002
```

En la definición del espacio de trabajo, se incorpora una carpeta denominada “catkin_ws” que contiene otra carpeta denominada “src”. Esta última carpeta será la ubicación del programa encargado de suscribirse a los topics publicados por la aplicación de realidad mixta y de generar y enviar el URScript.

La línea de código:

```
RUN /bin/bash -c 'source /opt/ros/melodic/setup.bash && catkin_make'
```

Activa el entorno de ROS Melodic y compila el espacio de trabajo.

Una vez definido el Dockerfile, se debe de construir la nueva imagen accediendo a la ubicación de este archivo desde una terminal de Windows e introduciendo la siguiente línea de comandos:

docker build –t ur5e_controller:v1 .

De esta manera, se construye la nueva imagen denominada “ur5e_controller”.

Por último, la ejecución de un contenedor con la imagen creada y, los puertos 9090 y 30002 mapeados al equipo local, se logra añadiendo la siguiente línea de comandos.

docker run -it -p 9090:9090 -p 30002:30002 –-restart=always –name ur5e_cont1 ur5e_controller:v1

3.4.2.2 Creación del paquete en catkin_ws

Los pasos seguidos para construir el paquete "hololens_ur5e" se enumeran a continuación.

1. En primer lugar, situarse en la carpeta **~/catkin_ws/src**

   cd ~/catkin_ws/src

2. Después, crear el paquete con las dependencias std_msgs, rospy y roscpp:

   catkin_create_pkg hololens_ur5e std_msgs rospy roscpp

3. Compilar el espacio de trabajo y obtener el archivo de configuración generado:

   cd ~/catkin_ws
   catkin_make
   source devel/setup.bash

4. Agregar las etiquetas de dependencias agregadas en el archivo package.xml para que las dependencias estén disponibles en tiempo de compilación y ejecución.

   rosed hololens_ur5e package.xml

   Y añadir las etiquetas de dependencia para la generación de mensajes.

5. Editar el CMakeLists.txt del paquete agregando el componente *message_generation* y las utilidades generar mensajes y servicios añadidos con la dependencia std_msgs.

   rosed hololens_ur5e CMakeLists.txt

   Cuando se compile el paquete de ROS creado, utilizando el comando "catkin_make", se leerá este archivo para que el entorno sepa que programas son los que tiene que compilar.

3.4.2.3 Programación del traductor

Se ha desarrollado un programa escrito en lenguaje de programación C++ que inicia la conexión con el UR5e vía socket TCP/IP por el puerto 30002 y se suscribe a los topics publicados por la aplicación de realidad mixta con el objetivo de traducir los datos recibidos al lenguaje nativo del robot.

Cada maniobra enviada al robot va acompañada con una instrucción que abra un socket con el IOT2050 y, después, envíe dos mensajes con el color al que se quiere cambiar las luces. El primer color corresponde al rojo y representa el inicio de la maniobra, indicando al operario que el robot está en movimiento y que debe de alejarse de la zona de riesgo. El segundo color corresponde al verde e indica la finalización de la maniobra por parte del UR5e.

En las maniobras Pick & Place se realiza el mismo procedimiento con la dirección IP y puerto donde se ejecuta el servidor del Arduino Uno que acciona el servomotor de la garra.

El código que realiza este propósito se describe a continuación haciendo énfasis a las diferentes partes que integran la estructura de este.

Bibliotecas:

Se importan las librerías de ROS y de socket, junto a las más comunes para el tratamiento de variables.

```
//ROS
#include <ros/ros.h>
#include "std_msgs/String.h"
#include "geometry_msgs/Vector3.h"
#include "sensor_msgs/JointState.h"
#include "geometry_msgs/Vector3.h"
#include "geometry_msgs/Point32.h"
#include "tf2_msgs/TFMessage.h"
//Socket
#include <sys/socket.h>
#include <netinet/in.h>     // in_addr structure
#include <netdb.h>          //hostent struct, gethostbyname()
#include <arpa/inet.h>
#include <sys/types.h>
#include <stdio.h>          // stderr, stdout
#include <arpa/inet.h>      // inet_ntoa() to format IP address
//C++
#include <string.h>
#include <unistd.h>
#include <iostream>
#include <string>
#include <sstream>
#include <stdlib.h>         //exit
#include <string.h>
#include <signal.h>         //catch ctrl+c
#include <pthread.h>
#include <vector>
#include <cmath>
```

Funciones y variables globales.

Declaración de las funciones y variables globales que se utilizan en todo el programa.

```
using namespace std;

//Statement of functions
void MoveToHome();
void MoveToRepose();
void SendURScript(vector<string>);
void ClearVariables();
void SaveCoordinates(double,double,double);

int client_sockfd;              // ID socket
struct sockaddr_in serv_addr; //Estructura genérica
#define PORT 30002              // Port of UR5e
#define HOST "192.168.101.5"    // IP of UR5e

/* GLOBAL VARIABLES */
int cont = 9;            //Counter to keep track of coordinates sent in RealTime MODE
bool contFirst = true;   //Variable to indicate the first coordinate sent in Real Time
MODE
bool FirstPoint = true;  //Indicate the first point sent in Trayectory MODE
bool repeat = false;     //Changes value when the trajectory is marked to repeat the
traced path
int TrayorPick = 0;      //Know if we are in Trajectory MODE or Pick and Place MODE
int contRepeat = 0;      //Increases if repeat trajectory or Pick & Place has been
indicated.
int PickorPlace = 0;     //To know if the robot has already performed the first maneuver.
double req_pan_vel, req_lift_vel, req_elbow_vel, req_wrist_1_vel, req_wrist_2_vel,
req_wrist_3_vel; //TCP velocity
double tool_x,tool_y,tool_z; //TCP position
bool allow_move=true; //Allow the robot movement
bool stop_flag=false; //When say "Stop"
bool loop_flag=false; //When say "Loop"
double tray_x,tray_y,tray_z, pick_x,pick_y,pick_z; //Save the last point sent
int cont_loop_tray=0, pickorplace=0;
```

Los vectores almacenan las instrucciones a enviar al robot en lenguaje URScript.

```
vector<string> vec;      //Complete URScript of Real Time MODE
vector<string> aux;      //Coordinates received from RealTime MODE
vector<string> trayL;    //Stores MoveL coordinates of Trayectory MODE
vector<string> trayJ;    //Stores MoveJ coordinates of Trayectory MODE
vector<string> tray;     //Complete URScript of Trayectory MODE
vector<string> pandp;    //Stores coordinates of Pick
vector<string> qandp;    //Stores coordinates of Place
vector<string> pandp2;    //Stores coordinates of Pick
vector<string> qandp2;    //Stores coordinates of Place
vector<string> looping;  //Stores coordinates of Pick and Place
vector<string> looping2;
vector<string> stopping; //Stores the instruction to stop robot movement
vector<string> aux_loop;
```

Las cadenas *ostringstream* almacenan las instrucciones fila por fila.

```
ostringstream stream_ss;    //String for Real Time MODE
ostringstream auxPoint;     //String for first point of Trayectory MODE
ostringstream ll;           //String for MoveL of Trayectory MODE
ostringstream jj;           //String for MoveJ of Trayectory MODE
ostringstream pp;           //String for Pick
ostringstream qq;           //Cadena for Place
ostringstream pp2;           //String for Pick
ostringstream qq2;           //Cadena for Place
```

Los mensajes *std_msgs* almacenan el valor de los topic que publiquen datos de este tipo y, los *stringstream* sirven para realizar comparaciones entre cadenas de caracteres y los mensajes *std_msgs.*

```
/* std_msgs for compare topics with string*/
std_msgs::String msg1;
std::stringstream go_repose;
std_msgs::String msg2;
std::stringstream go_home;
std_msgs::String moving;
std::stringstream type_move;
std::stringstream type_open;
std_msgs::String msgOpen;
std::stringstream type_close;
std_msgs::String msgClose;
std::stringstream type_stop;
std_msgs::String msgStop;
```

Funciones manejadoras:

Se llaman cada vez que se publica el topic correspondiente que las activan.

- hololensStream():
  - Se activa cada vez que se publica un mensaje en el topic /HLstream.
  - Almacena y envía al robot las coordenadas traducidas al lenguaje del robot de diez en diez con el fin de conseguir un seguimiento del efector final bastante fluido y cercano al tiempo real.
  - La primera coordenada recibida se envía directamente con el fin de mover el robot lo más rápido posible a la posición indicada por el usuario y, a partir de ahí, realizar el seguimiento.
- hololensTrayectory():
  - Se activa cada vez que se publica un mensaje en el topic /HLtray
  - Almacena las coordenadas recibidas en dos vectores: trayL, trayJ. El primero guarda la trayectoria programada para MoveL y el segundo para MoveJ.
  - Tiene en cuenta que la primera coordenada recibida sea tipo MoveJ, ya que es la que hará que el robot evite llegar a singularidades al intentar acceder a la posición indicada.

- hololensMove():
  - Se activa cada vez que se publica un mensaje en el topic /HLmove
  - Genera el URScript con la trayectoria seleccionada por el usuario y el tipo de movimiento indicado para posteriormente enviárselo al robot.
- hololensPick():
  - Se activa cada vez que se publica un mensaje en el topic /HLpick
  - Almacena la coordenada recibida en dos vectores: "pandp" y "looping". La primera será utilizada para generar el URScript que se envíe al indicar la coordenada, y la segunda será utilizada a la hora de seleccionar la opción Loop.
  - Envía al robot el URScript generado con la coordenada Pick indicada por el usuario. En este URScript se introduce el socket cliente del Arduino y el mensaje "close" como la información enviada a este dispositivo.
  - La coordenada Z es establecida de manera predeterminada con el valor de la altura la mesa de trabajo sobre el sistema de coordenadas del robot colaborativo.
- hololensPlace():
  - Se activa cada vez que se publica un mensaje en el topic /HLplace
  - Almacena la coordenada recibida en dos vectores: "qandp" y "looping". La primera será utilizada para generar el URScript que se envíe al indicar la coordenada, y la segunda será utilizada a la hora de seleccionar la opción Loop.
  - Envía al robot el URScript generado con la coordenada Place indicada por el usuario. En este URScript se introduce el socket cliente del Arduino y el mensaje "open" como la información enviada a este dispositivo.
  - La coordenada Z es establecida de manera predeterminada con el valor de la altura la mesa de trabajo sobre el sistema de coordenadas del robot colaborativo.
- hololensLoop():
  - Se activa cada vez que se publica un mensaje en el topic /HLloop
  - Envía al robot el URScript generado con la trayectoria o el Pick & Place indicado.
- hololensNew():
  - Se activa cada vez que se publica un mensaje en el topic /HLnew

    - Reinicia las cadenas *ostrinstream* y vectores que almacenan las instrucciones a enviar al robot.
- **hololensStop():**
    - Se activa cada vez que se publica un mensaje en el topic /HLstop
    - Envía al robot el URScript con las instrucciones para detener su movimiento.
- **hololensHome():**
    - Se activa cada vez que se publica un mensaje en el topic /HLhome
    - Envía al robot el URScript con las instrucciones para moverse a su posición de inicio.
    - Reinicia las cadenas *ostrinstream* y vectores que almacenan las instrucciones a enviar al robot.
- **hololensRepose():**
    - Se activa cada vez que se publica un mensaje en el topic /HLrepose
    - Envía al robot el URScript con las instrucciones para moverse a la posición definida como reposo.
    - Reinicia las cadenas *ostrinstream* y vectores que almacenan las instrucciones a enviar al robot.
- **callback():**
    - Se activa cada vez que se publica un mensaje en el topic /joint_states
    - Actualiza el valor de las variables que almacenan la velocidad actual de los ejes.
- **tool_data():**
    - Se activa cada vez que se publica un mensaje en el topic /tf
    - Actualiza el valor de las variables que almacenan las coordenadas actuales del TCP del robot.

El código adjunto pertenece a la función hololensPick que procesa las coordenadas publicadas en el modo Pick&Place para generar el URscript que se envía al robot colaborativo.

```
// ############# PICK AND PLACE MODE #############
void hololensPick(const geometry_msgs::Vector3::ConstPtr& pick)
{
    SaveCoordinates(pick->x+0.25,pick->y+0.25,pick->z+0.25);
    TrayorPick = 2; //Se indica que estamos en Pick and Place MODE
    cout << "Pick" << endl;

    pandp.push_back("def
myProg():\nvar_1=socket_open(\"192.168.101.92\",10034)\nsocket_send_string(\"ROJO\")\nsock
et_close()\n"); //INICIO PICK
```

```
    looping.push_back("def
myProg():\nvar_1=socket_open(\"192.168.101.92\",10034)\nsocket_send_string(\"ROJO\")\nsock
et_close()\n"); //INICIO PICK EN LOOP

    pp << "movej(p[" << pick->x << "," << pick->y << "," << 0.25 << ", 2.2, 2.2, 0],
a=0.8, v=20, r=0)\nmovel(p[" << pick->x << "," << pick->y << "," << 0.04 << ", 2.2, 2.2,
0], a=0.1, v=3,
r=0)\nvar_1=socket_open(\"192.168.101.205\",21)\nsocket_send_string(\"close\")\nsocket_clo
se()\nsleep(2.)\nmovel(p[" << pick->x << "," << pick->y << "," << 0.25 << ", 2.2, 2.2, 0],
a=0.8, v=20, r=0)\nsleep(2.)\n";
    pp2 << "movej(p[" << pick->x << "," << pick->y << "," << 0.25 << ", 2.2, 2.2, 0],
a=0.8, v=20, r=0)\nmovel(p[" << pick->x << "," << pick->y << "," << 0.04 << ", 2.2, 2.2,
0], a=0.1, v=3,
r=0)\nvar_1=socket_open(\"192.168.101.205\",21)\nsocket_send_string(\"open\")\nsocket_clos
e()\nsleep(2.)\nmovel(p[" << pick->x << "," << pick->y << "," << 0.25 << ", 2.2, 2.2, 0],
a=0.8, v=20, r=0)\nsleep(2.)\n";

    pandp.push_back(pp.str());
    looping2.push_back(pp.str());
    pp.str(std::string());

    pandp.push_back("var_1=socket_open(\"192.168.101.92\",10034)\nsocket_send_string(\"VER
DE\")\nsocket_close()\nend\n"); //FINAL PICK
    //ENVÍO
   SendURScript(pandp);

}
```

En la función SendURScript() se abre el socket que permite la conexión con el UR5e, se establece esta conexión, se envía el URScript con las instrucciones traducidas al lenguaje del robot y se cierra el socket. Se ha decidido abrir y cerrar la comunicación cada vez que se quiere enviar un mensaje al robot colaborativo, ya que se ha comprobado que durante largos periodos de inactividad se pierde la comunicación entre los dispositivos aunque el socket siga abierto.

```
void SendURScript(vector<string> instructions)
{
    if(allow_move || stop_flag)
    {
        client_sockfd = socket(AF_INET, SOCK_STREAM, 0); //Apertura del socket
        if (client_sockfd < 0)
        {
            cout << "Error creando socket" << endl;
            exit(-1);
        }

        bzero((char*)&serv_addr, sizeof(serv_addr)); //bzero - apaga todos os dados da
estrutura (coloca a 0's)
        serv_addr.sin_family = AF_INET;     //AF_INET - endereço IP
        serv_addr.sin_addr.s_addr = inet_addr(HOST);;//INADDR_ANY - aceitar pedidos para
qualquer IP
        serv_addr.sin_port = htons(PORT);

        //Conexión con el servidor
        if (connect(client_sockfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) < 0)
        {
            cout << "No conectado" << endl;
```

```
        }

        for (vector<string>::iterator it = instructions.begin(); it != instructions.end();
++it)
        {
            cout << it->c_str() << endl;
            write(client_sockfd, it->c_str(), it->length());

        }
        close(client_sockfd);
        cout<<"URScript send"<<endl;
    }
    else
    {
        //cout<<"Robot is moving"<<endl;
    }
}
```

En la función SendURScript() se comprueba que el robot colaborativo esté parado antes de enviar las instrucciones. Si este se encuentra ejecutando otra maniobra, las instrucciones actuales son desestimadas.

Main:

Se inicializan las variables *ostringstream* y se registra el nodo de ROS que se suscribe a los topics publicados por HL2 y el UR5e.

```
int main(int argc, char** argv) {

    go_repose << "go_repose"; //Se asigna cadena a la variable stringstream
    msg1.data = go_repose.str(); //Y se guarda
    go_home << "go_home"; //Se asigna cadena a la variable stringstream
    msg2.data = go_home.str(); //Y se guarda
    type_move << "MoveL"; //Se asigna cadena a la variable stringstream
    moving.data = type_move.str(); //Y se guarda
    type_stop << "Stop"; //Asignamos cadena a la variable initiate
    msgStop.data = type_stop.str(); //Y la guardamos en msg2.data

    ros::init(argc, argv, "ur5_sender");
    ros::NodeHandle n_HL;
    ros::Subscriber sub_HL2 = n_HL.subscribe("HLstream", 1000, hololensStream);
    ros::Subscriber sub_tray = n_HL.subscribe("HLtray", 1000, hololensTrayectory);
    ros::Subscriber sub_move = n_HL.subscribe("HLmove", 1000, hololensMove);
    ros::Subscriber sub_home = n_HL.subscribe("HLhome", 1000, hololensHome);
    ros::Subscriber sub_repose = n_HL.subscribe("HLrepose", 1000, hololensRepose);
    ros::Subscriber sub_pick = n_HL.subscribe("HLpick", 1000, hololensPick);
    ros::Subscriber sub_place = n_HL.subscribe("HLplace", 1000, hololensPlace);
    ros::Subscriber sub_new = n_HL.subscribe("HLnew", 1000, hololensNew);
    ros::Subscriber sub_loop = n_HL.subscribe("HLloop", 1000, hololensLoop);
    ros::Subscriber sub_stop = n_HL.subscribe("HLstop", 1000, hololensStop);
    ros::Subscriber sub_ur = n_HL.subscribe("joint_states", 1000, callback);
    ros::Subscriber sub_tool = n_HL.subscribe("tf", 1000, tool_data);
    ros::spin();
    return 0;
}
```

Para incorporar este programa al paquete previamente creado es necesario seguir los siguientes pasos:

1. Acceder a la carpeta *src* del paquete:

roscd hololens_ur5e/src

2. Crear el archivo con la extensión del lenguaje del programa. Se ha denominado a este programa como "urscript_socket.cpp".

nano urscript_socket.cpp

3. Copiar el código del programa.

4. Agregar el ejecutable del programa creado en el archivo CMakeLists.txt.

cd ~/catkin_ws
rosed hololens_ur5e CMakeLists.txt

Se añade al final del código esta información:

```
add_executable(urscript_socket src/urscript_socket.cpp)
target_link_libraries(urscript_socket ${catkin_LIBRARIES})
add_dependencies(urscript_socket hololens_ur5e_generate_messages_cpp)
```

5. Compilar el paquete.

catkin_make

3.4.2.4 Publicador de las coordenadas de los códigos QR

La carpeta *scripts* contenida dentro del espacio de trabajo *catkin_ws* está destinada a almacenar el programa escrito en lenguaje Python que publica las coordenadas de los códigos QR en la referencia del robot.

Este script denominado "rob_coord.py" publica las tres coordenadas en el topic /rob_coord de ROS. A este topic se suscribe la aplicación de realidad mixta nada más iniciar la aplicación y permanece a la espera de que se publique información para almacenar los datos recibidos en su programa. Para introducir este programa en el espacio catkin se deben de seguir los siguientes pasos:

1. Crear la carpeta *scripts*:

mkdir scripts

2. Crear el programa *coord_rob.py*

cd scripts

nano coord_rob.py

3. Escribir el código necesario para publicar las tres coordenadas en el topic /coord_rob

```
#!/usr/bin/env python
import rospy
#from std_msgs.msg import Float64 # from package.[msg/srv] import ["msg"/"srv"]
from geometry_msgs.msg import Pose
pose_msg = Pose()
pose_msg.orientation.x = 0
pose_msg.orientation.y = 0
pose_msg.orientation.z = 0
pose_msg.orientation.w = 1

def talker():
    pub = rospy.Publisher('rob_coord', Pose, queue_size=10) # TOPIC
    rospy.init_node('rob_coord', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    for k in range(3):

        if(k==0):
                pose_msg.position.x = -0.14124
                pose_msg.position.y = 0.26935
                pose_msg.position.z = 0.0124
                pub.publish(pose_msg)
        elif(k==1):
                pose_msg.position.x = -0.774
                pose_msg.position.y = 0.2486
                pose_msg.position.z = 0.0124
                pub.publish(pose_msg)
        else:
                pose_msg.position.x = -0.1409
                pose_msg.position.y = 0.77142
                pose_msg.position.z = 0.0124
                pub.publish(pose_msg)

        rospy.loginfo('SEND DATA: \n%s', pose_msg)

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        Pass
```

4. Agregar lo siguiente al CMakeLists.txt .Esto asegura que el script de python se instale correctamente y use el intérprete de python correcto.

cd ..
rosed hololens_ur5e CMakeLists.txt

```
catkin_install_python(PROGRAMS scripts/roob__coord.py
    DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)
```

5. Compilar el paquete:

catkin_make

3.4.2.5 Servidor de socket en IOT2050

Cuando el dispositivo IOT2050 se encuentra encendido, ejecuta un programa servidor de socket en el puerto 10034 que acepta conexiones de varios clientes y traduce los mensajes recibidos mediante protocolo TCP/IP a protocolo RS485. Estos dos procesos se ejecutan simultáneamente mediante el uso del método *multiprocessing.Process* de la librería *multiprocessing* que incorpora Python.

De esta manera, al recibir la trama con la cadena de caracteres del color que se quiere iluminar las tiras de luces LED, se envía al DMX una lista con los códigos RGB que corresponden a dicho color. El código diseñado para cumplir este propósito es el siguiente:

```
import socket #TCP/IP protocol
import dmx #DMX protocol
import multiprocessing
from multiprocessing import Manager, Value
import time
from asyncio import StreamWriter, StreamReader

manager = Manager()
valor=Value('i',0)
#Instance to create the list with the necessary leds and send them.
def set_leds(rojo_1, rojo_2, rojo_3, rojo_4, verde_1, verde_2, verde_3, verde_4, azul_1,
azul_2, azul_3, azul_4,sender):
    lista = [rojo_1, verde_1, azul_1, rojo_2, verde_2, azul_2, rojo_3, verde_3, azul_3,
rojo_4, verde_4, azul_4]
    lista2 = [0 for i in range (500)]
    lista.extend(lista2)
    lista_leds = bytes(tuple(lista))
    sender.set_data(lista_leds)
    return lista_leds
#Server socket TCP/IP
def server(valor):
    # Create a TCP/IP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_address = ('192.168.101.92', 10034)
    sock.bind(server_address)
    # Listen for incoming connections
    sock.listen(1)
    while True:
        # Wait for a connection
        connection, client_address = sock.accept()
        try:
            # Receive the data in small chunks and retransmit it
            while True:
                try:
                    data = connection.recv(16)
                    if data:
                        if format(data.decode('utf-8'))== 'ROJO':
                            valor.value=1
                            time.sleep(1)
                        elif format(data.decode('utf-8'))== 'VERDE':
                            valor.value=2
                        elif format(data.decode('utf-8'))== 'AMARILLO':
                            valor.value=3
                        elif format(data.decode('utf-8'))== 'BLANCO':
```

```
                            valor.value=0
                            time.sleep(1)
                        print(format(data.decode('utf-8')))
                    else:
                        break
                except:
                    break
        finally:
            # Clean up the connection
            connection.close()

#Instance and initiation of serial communication with DMX decoder
def comm_DMX():
    sender = dmx.DMX_Serial (port="/dev/ttyUSB0")
    sender.start()
    return sender

#Connection with DMX512
def client(valor):
    sender=comm_DMX()
    time.sleep(1)
    set_leds(255,255,255,255,255,255,255,255,255,255,255,255,sender)
    while True:
        print('x = {0}'.format(valor.value))
        if valor.value==1:
            lista=set_leds(255,255,255,255,0,0,0,0,0,0,0,0,sender)
            print(lista)
        elif valor.value==2:
            set_leds(0,0,0,0,255,255,255,255,0,0,0,0,sender)
        elif valor.value==3:
           set_leds(255,255,255,255,255,255,255,255,0,0,0,0,sender)
        elif valor.value== 0:
      set_leds(255,255,255,255,255,255,255,255,255,255,255,255,sender)
        time.sleep(1)

if __name__ == "__main__":
    time.sleep(1)
    y = multiprocessing.Process(target=client, args=(valor,)).start()
    x = multiprocessing.Process(target=server, args=(valor,)).start()
```

3.4.2.6 Servidor de socket en Arduino UNO

La función del Arduino UNO se basa en servir de medio de transmisión para el accionamiento del servomotor que abre y cierra las pinzas de la garra. El programa diseñado acepta la conexión de varios clientes y traduce el mensaje recibido a un tren de pulsos que acciona el servomotor. Este tren de pulsos contiene el ángulo de apertura de la garra.

El servidor se ejecuta en el puerto 21 (Telnet) y la estructura del código diseñado es la siguiente:

```
#include <SPI.h>
#include <EthernetENC.h>
#include <Servo.h>
Servo myservo;  //Object
int angulo;
const int digout1 = 2;
const int digout2 = 4;
String sal = "";
```

```
int cerrar=0; // Pin 2 status
int abrir=0; //Pin 4 status
byte mac[] = {
  0x00, 0x25, 0x64, 0xBD, 0x93, 0xBC };
IPAddress ip(192, 168, 101, 205);
IPAddress myDns(192, 168, 1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);
// telnet defaults to port 23
EthernetServer server(21);
boolean alreadyConnected = false; // whether or not the client was connected previously
void setup() {
  // initialize the ethernet device
  Ethernet.begin(mac, ip, myDns, gateway, subnet);
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
   while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  // Check for Ethernet hardware present
  if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println("Ethernet shield was not found.  Sorry, can't run without hardware.
:(");
    while (true) {
      delay(1); // do nothing, no point running without Ethernet hardware
    }
  }
  if (Ethernet.linkStatus() == LinkOFF) {
    Serial.println("Ethernet cable is not connected.");
  }
  // start listening for clients
  server.begin();
  Serial.print("Chat server address:");
  Serial.println(Ethernet.localIP());
    myservo.attach(3);  // assign pin 3 to the servo.
  pinMode (digout1, INPUT);  //Input pin 2 for the digital output 0 of the robot
  pinMode (digout2, INPUT);  //Input pin 4 for the digital output 1 of the robot
}
void loop() {
  cerrar = digitalRead(digout1);
  abrir = digitalRead(digout2);
  // wait for a new client:
  Ethernet.client client = server.available();
  // when the client sends the first byte, say hello:
  if (client) {
    if (!alreadyConnected) {
      // clear out the input buffer:
      client.flush();
      Serial.println("We have a new client");
      client.println("Hello, client!");
      alreadyConnected = true;
    }
    while (client.available() > 0) {
      // read the bytes incoming from the client:
      char thisChar = client.read();
      sal+=thisChar;
      Serial.write(thisChar);
    }
  }
  if (sal == "close") {
        angulo = 80;
        myservo.write(angulo);
```

```
        Serial.print("Ã¡ngulo:  ");
        Serial.println(angulo);
        sal = "";
        delay(1000);
      }
      else if (sal== "open") {
        angulo = 0;
        myservo.write(angulo);
        Serial.print("Ã¡ngulo:  ");
        Serial.println(angulo);
        sal = "";
        delay(1000);
      }
      else if (sal !="close" && sal != "open" && sal !=""){
        angulo =0.8* sal.toInt();
        myservo.write(angulo);
        Serial.print("Ã¡ngulo:  ");
        Serial.println(angulo);
        sal = "";
        delay(1000);
      }
}
```

3.5 Conclusiones

En este capítulo se describen todos los aspectos de diseño y desarrollo del proyecto que abarca este TFG. Los detalles de la programación de los códigos que hacen posible la correcta conexión entre los dispositivos que rigen el sistema de realidad mixta han sido descritos de manera precisa, ilustrando mediante imágenes y tablas su estructura y funcionamiento.

En la Ilustración 57 se aprecia el diagrama de bloques de la Ilustración 34 completado con la información transmitida entre los dispositivos del sistema.

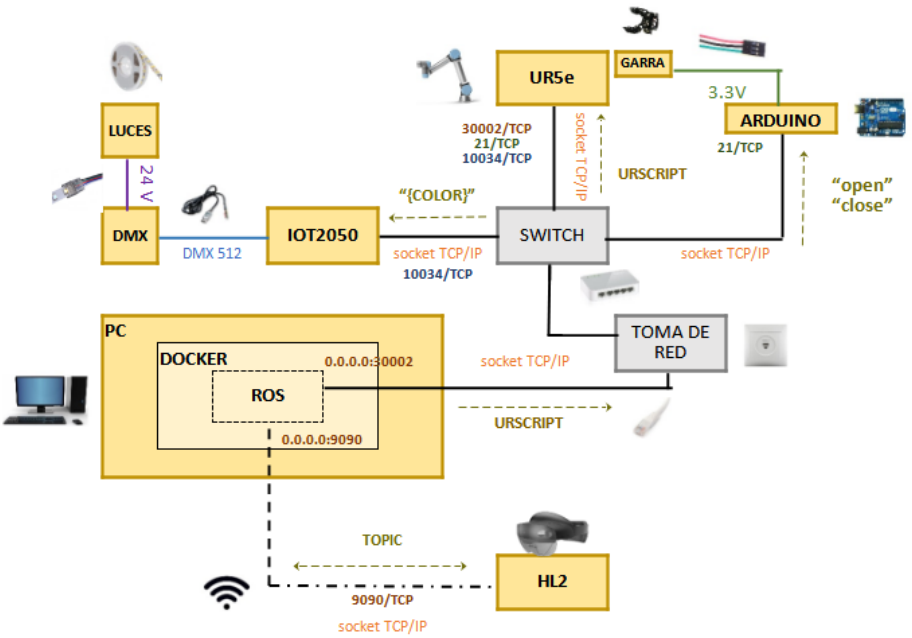


Ilustración 57. Diagrama de bloques de la arquitectura de alto nivel del sistema de realidad mixta

La actividad se resume en realizar una transmisión de los datos desde el dispositivo HL2 hasta ROS mediante el mecanismo publicador/suscriptor. El usuario indica posiciones con su dedo índice o activa comandos de voz que publican información en los topics de ROS. Es entonces cuando un programa ejecutado en ROS traduce los datos enviados por el usuario a lenguaje nativo del robot, generando el fichero URScript correspondiente y enviándolo a la controladora del robot vía socket TCP/IP.

El fichero URScript contiene instrucciones tanto para mover el robot a posiciones dadas, como para transmitir información al IOT2050 y/o el Arduino UNO vía socket. Generalmente, se pretende que las instrucciones que hagan que el robot se mueva abran un socket con el IOT2050 para cambiar el color de las luces LED cuando empiece y finalice el movimiento del robot.

El accionamiento del servomotor de la garra se activa en el modo Pick & Place cuando se abra un socket con el Arduino UNO que envía las señales correspondientes para abrir y cerrar las pinzas.

La actividad finaliza cuando el usuario cierre la aplicación del HL2 o se cierre la conexión entre ROS y el UR5e.

Capítulo 4. Experimentación y resultados

La validación del sistema de realidad mixta desarrollado depende del correcto funcionamiento de la interfaz usuario-robot y de los demás componentes que lo integran. Extraer una conclusión de la viabilidad del sistema requiere de una evaluación objetiva de la experiencia del usuario y de la efectividad de las actividades llevadas a cabo durante su ejecución.

En el apartado 4.1 se enumeran los pasos llevados a cabo para realizar arranque del sistema y, en el apartado 4.2, se evalúan los resultados obtenidos experimentalmente a la hora de probar y validar el sistema desarrollado.

4.1 Arranque del sistema

La ejecución del sistema de realidad mixta parte por tener operativos todos los componentes que integran su arquitectura y configurarlos de manera que sea posible la transmisión cíclica de los datos. La puesta en marcha del sistema se enumera a continuación:

1. Encender todo el hardware del sistema (Opcional). No se requiere que todos los componentes del sistema estén encendidos desde el primer momento, pero es una buena práctica para evitar problemas a la hora de localizar errores en la comunicación y transmisión de los datos.
2. Establecer el robot UR5e en modo remoto. En este momento, el robot queda bloqueado y a la espera de clientes que soliciten la conexión vía socket TCP/IP por el puerto 30002.
3. Crear/Abrir el contenedor de Docker con la imagen ur5e_controller:v1

   $ docker run -it -p 9090:9090 -p 30002:30002 –-restart=always –name ur5e_cont1 ur5e_controller:v1 [Crear]
   $ docker exec –it ur5e_cont1 bash [Abrir]

4. Configurar el entorno ROS del contenedor.
   a. Abrir una terminal y activar el entorno de ROS.

      $ source devel/setup.bash

      Siempre que se habrá una nueva terminal se debe de introducir este comando.
   b. Ejecutar el nodo maestro.

      $ roscore

   c. En una nueva terminal, ejecutar el puente de ROS que ejecutará un servidor de socket por el puerto 9090.

```
$ roslaunch rosbridge_server rosbridge_websocket.launch
```

d. En otra terminal, ejecutar el nodo "traductor", es decir, el nodo que corre el programa desarrollado para suscribirse a los topics publicados por la aplicación de realidad mixta y, genera y envía el URScript al UR5e por el puerto 30002.

```
$ rosrun hololens_ur5e urscript_socket
```

e. Iniciar el controlador del robot.

```
$ roslaunch ur_robot_driver ur5e_bringup.launch
robot_ip:=192.168.101.5
```

f. Si ya se ha iniciado la aplicación de realidad mixta en el HL2. Publicar las coordenadas de los códigos QR en el topic /coord_rob

```
$ rosrun hololens_ur5e urscript_socket
```

5. Comienzo de la actividad.

4.2 Ensayos y validación del sistema

Una vez se ha puesto en marcha el sistema de realidad mixta, el usuario observará un escenario donde:

- El robot esté a la espera de la llegada de ficheros URScript vía socket TCP/IP.
- Las luces LED RGB estén iluminadas de color blanco.
- El IOT 2050 esté a la espera de mensajes vía socket TCP/IP.
- Se esté ejecutando el nodo *urscript_socket* en ROS.
- Las lentes del HL2 estén proyectando el objeto Welcome Dialog.

Entonces, será cuando el usuario solo debe de interactuar con la aplicación de realidad mixta integrada en el HL2.

4.2.1 Definición del sistema de referencia y proyección del espacio de trabajo del robot

En primer lugar, la aplicación informará al usuario sobre el procedimiento para definir el sistema de coordenadas del robot. El diálogo Referential Dialog muestra una imagen que ilustra el orden en el que deben de ser escaneados los códigos QR y la forma de hacerlo.

Este diálogo se bloqueará mientras la aplicación no haya recibido las coordenadas de los códigos QR en la referencia del robot, es decir, mientras

no se haya ejecutado el nodo *coord_rob.* Cuando esto ocurra, se informará al usuario del acontecimiento y deberá ejecutar el programa coord_rob.py del paquete creado en ROS. En el momento en el que la aplicación reciba las coordenadas requeridas, se podrá continuar con la definición del sistema de coordenadas de la mesa de trabajo.

A la hora de escanear los códigos, el usuario observará en el diálogo la información contenida dentro del código QR. De esta manera, se puede saber si los códigos QR escaneados son válidos o no.

En las ilustraciones: Ilustración 58, Ilustración 59, Ilustración 61, Ilustración 62 e Ilustración 63, se muestra la secuencia de navegación de la aplicación según el usuario va realizando la detección de los códigos QR de la mesa de trabajo.

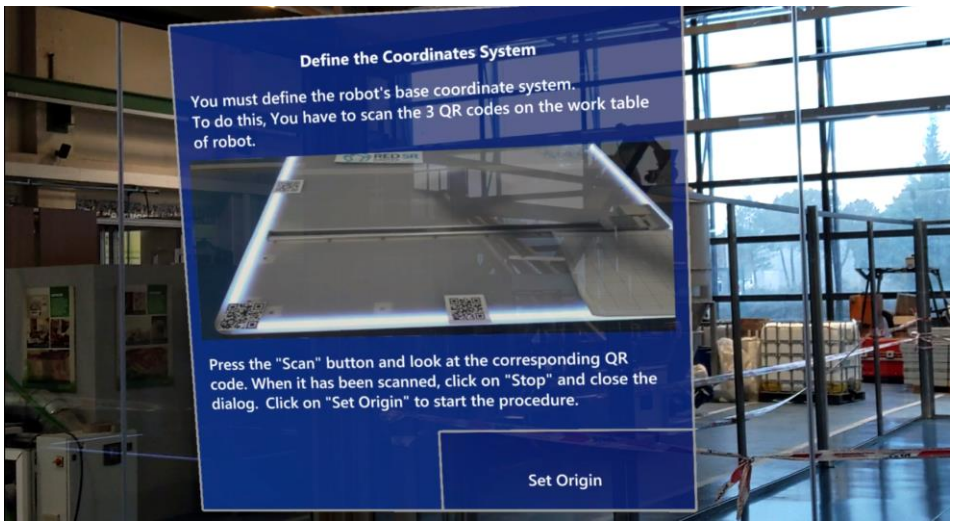


Ilustración 58. Proyección de las indicaciones para definir el sistema de coordenadas mesa

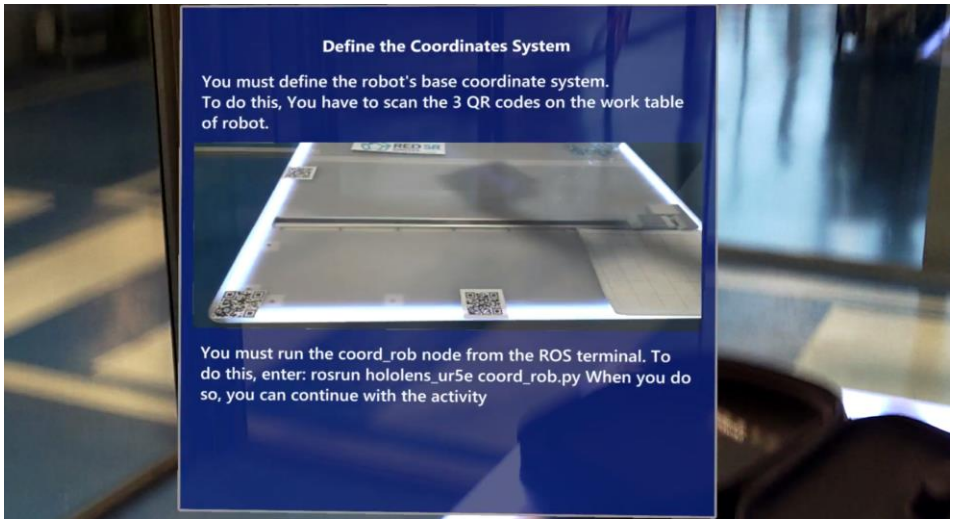


Ilustración 59. Diálogo Referential bloqueado

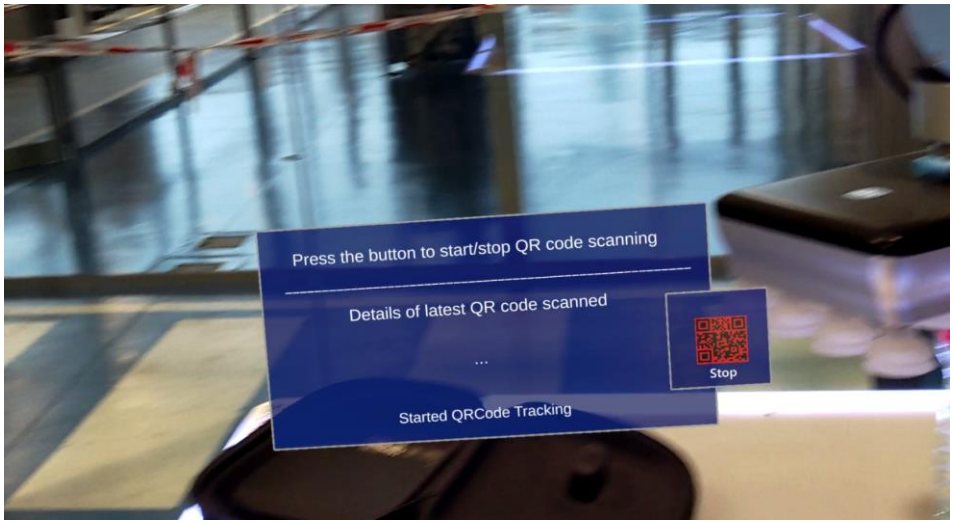


Ilustración 60. Proyección de la información contenida en el código QR nº1

En el caso de que el usuario escanee un código QR incorrecto el diálogo QRCodePanel se bloqueará e informará al operario del deber de escanear los códigos QR válidos.

Un código QR será detectado cuando aparezca sobre este el sistema de referencia mostrado en la Ilustración 61.

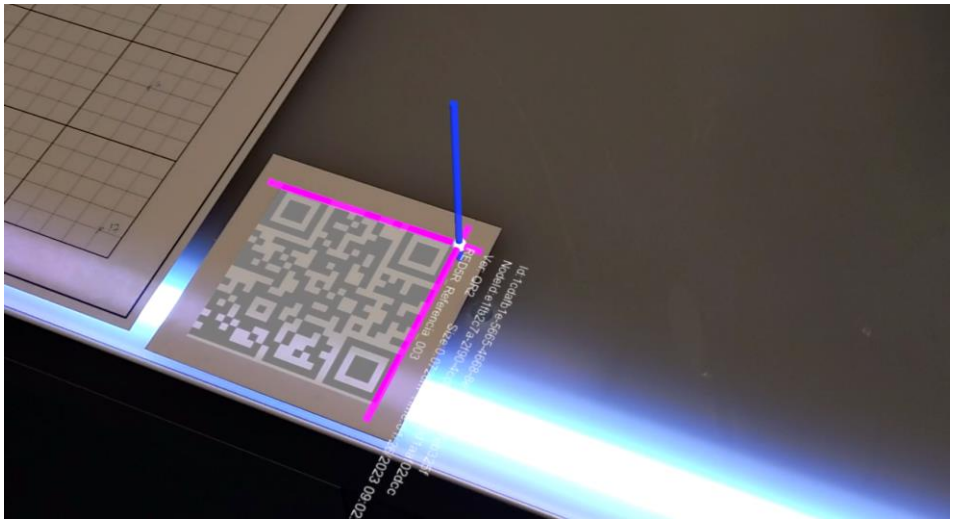


Ilustración 61. Código QR escaneado

Cuando se haya escaneado los códigos QR válidos, se habilitará el botón Close que permite cerrar el diálogo y continuar con el procedimiento. Posteriormente, la aplicación mostrará el espacio de trabajo del robot como una gran esfera verde con un cilindro en su interior.

En la Ilustración 63 se puede contemplar el espacio de trabajo del robot proyectado una vez se ha obtenido el origen de la referencia del robot. La esfera gris que engloba la esfera verde hace referencia a la zona donde el

robot puede alcanzar singularidades. La aplicación solo publicará en los topics las coordenadas indicadas dentro de la esfera verde.

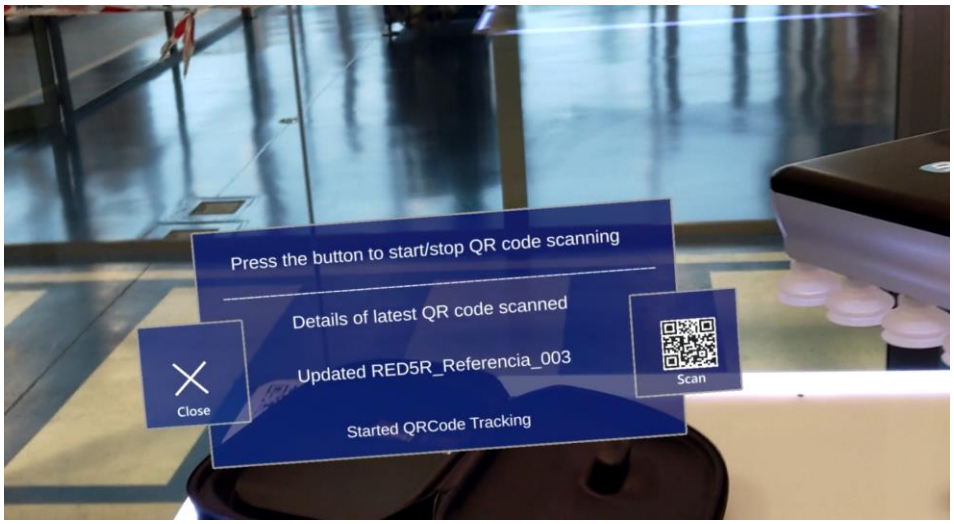


Ilustración 62. Detección válida de los códigos QR

Ilustración 63. Proyección del espacio de trabajo del robot

4.2.2 Modos de actividad

A continuación, se exponen los resultados experimentales obtenidos a la hora de realizar maniobras con el robot. Más en concreto, se apreciará como la información enviada por la aplicación de realidad mixta es correctamente traducida al lenguaje nativo del robot y ejecutada por este.

Se comenzará por exponer las maniobras que son ejecutadas directamente al interactuar con la botonera digital y se continuará describiendo el proceso realizado para programar maniobras mediante el uso de los comandos de voz establecidos en los modos de actividad del menú digital.

4.2.2.1 Botonera digital

Es preciso recordar que la botonera diseñada la constituyen tres pulsadores:

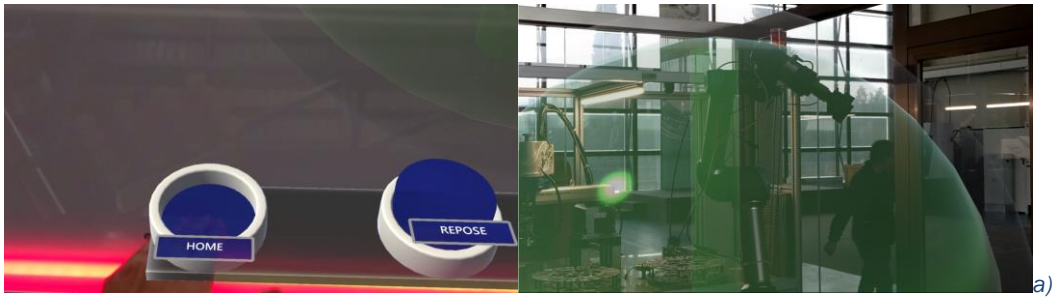
1. Home, cuya función es la de llevar al robot colaborativo a su posición inicial.
2. Repose, cuya función es la de llevar al robot colaborativo a una posición cómoda para comenzar con las maniobras cerca de la mesa de trabajo.
3. Help, cuya función es proyectar el panel “Instructions”

Al interactuar con los dos primeros pulsadores, el usuario observará como las luces LED cambiarán momentáneamente a color rojo indicando que el robot se encuentra en movimiento. Sucesivamente, el robot se trasladará a la posición indicada y, por último, las luces LED cambiarán a color verde haciendo ver que el robot se encuentra operativo para realizar una nueva maniobra.

Cuando el usuario haya pulsado el botón Begin del diálogo More Dialog, se proyectarán el menú y la botonera en las lentes del usuario. En la Ilustración 64 se muestra una fotografía realizada por las lentes del HL2 sobre el aspecto de la botonera que observa el usuario.

Ilustración 64. Proyección de la botonera digital

En la Ilustración 65 se contempla la secuencia de acontecimientos transcurridos desde el instante en el que el operario pulsa el botón Home hasta que el robot se para robot y cambia el color de las luces LED a verde.



a) Se pulsa "Home" y luces a rojo

b) UR5e trasladándose a "Home"

c) UR5e llega a "Home"

d) Luces a verde

Ilustración 65. Maniobra "Home" mostrada desde HL2

Sin embargo, cuando se interactúe con el pulsador Help, se proyectará en las lentes del dispositivo el panel Instructions que informa al usuario de los posibles errores que pueden ocurrir en el sistema y que impiden su correcto funcionamiento, tal como se aprecia en la Ilustración 66.

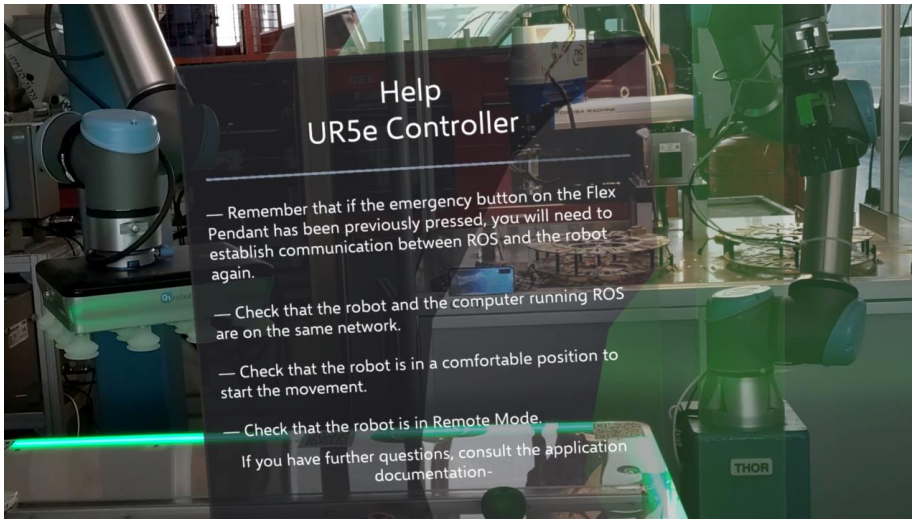


Ilustración 66. Panel de ayuda

4.2.2.2 Real Time

El modo Real Time se activa al interactuar con el primer botón del menú digital (en dirección izquierda a derecha). Este botón proyecta el panel RealTimeInstructions y habilita los comandos de voz Start y Stop, que

permiten publicar información en los topics /HLcoord y /HLstop respectivamente.

En la Ilustración 67 se puede contemplar como el panel de instrucciones del modo Real Time se ubica por encima del menú digital.

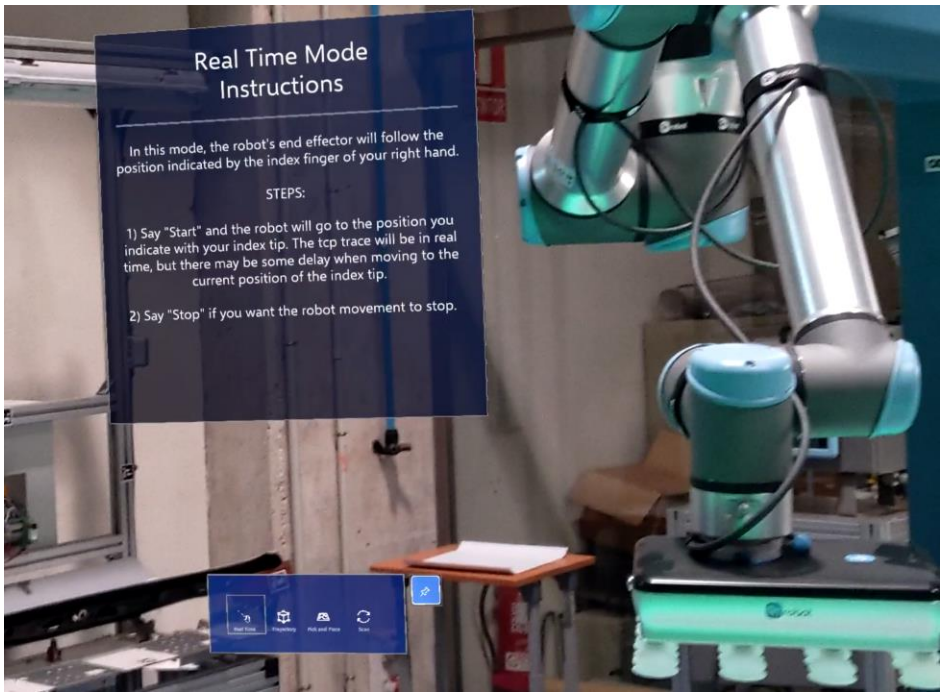


Ilustración 67. Panel de instrucciones del modo Real Time

Cuando el usuario activa el comando de voz «Start», el TCP del robot accede inmediatamente a la posición actual del dedo índice. Después, se irán proyectando pequeñas esferas ubicadas en las coordenadas publicadas en /HLcoord. El UR5e irá ejecutando los URScripts enviados por el nodo de ROS con las coordenadas indicadas.

Se ha observado que existe una ligera demora de tiempo entre la posición del dedo índice y la de la herramienta en un instante dado. Esto es debido a que el robot recibe el URScript con la trayectoria indicada una vez se han recopilado 10 coordenadas. Durante ese instante de tiempo, el operario puede seguir indicando coordenadas, que el robot seguirá ejecutando las instrucciones anteriores.

Al depender de un periodo de muestreo, a su vez, se depende en gran medida de la dinámica que elija el usuario a la hora de enviar de la información que se envía desde el dispositivo HL2.

El resultado visto desde las lentes del dispositivo de realidad mixta se puede apreciar en la Ilustración 68.

Ilustración 68. Ensayo experimental del modo Real Time

4.2.2.3 Trayectory Mode

La programación de una trayectoria por puntos indicados con el dedo índice de la mano derecha es posible en el modo Trayectory. Una vez se pulse el segundo botón del menú digital, se podrá utilizar el comando de voz «Set Point"» para indicar las coordenadas que se desean enviar al UR5e.

Se ha comprobado que la transmisión y traducción de los datos desde HL2 al robot se realiza correctamente. Para ello, se han realizado diferentes ensayos programando tanto trayectorias MoveL como MoveJ.

En el ejemplo mostrado en la Ilustración 69, el usuario indica cuatro puntos en el espacio que corresponden a las esquinas de una caja y, posteriormente, activa la señal «Go» que proyecta el diálogo MoveLorMoveJ para que el usuario elija como quiere que el robot realice la maniobra (MoveL o MoveJ).

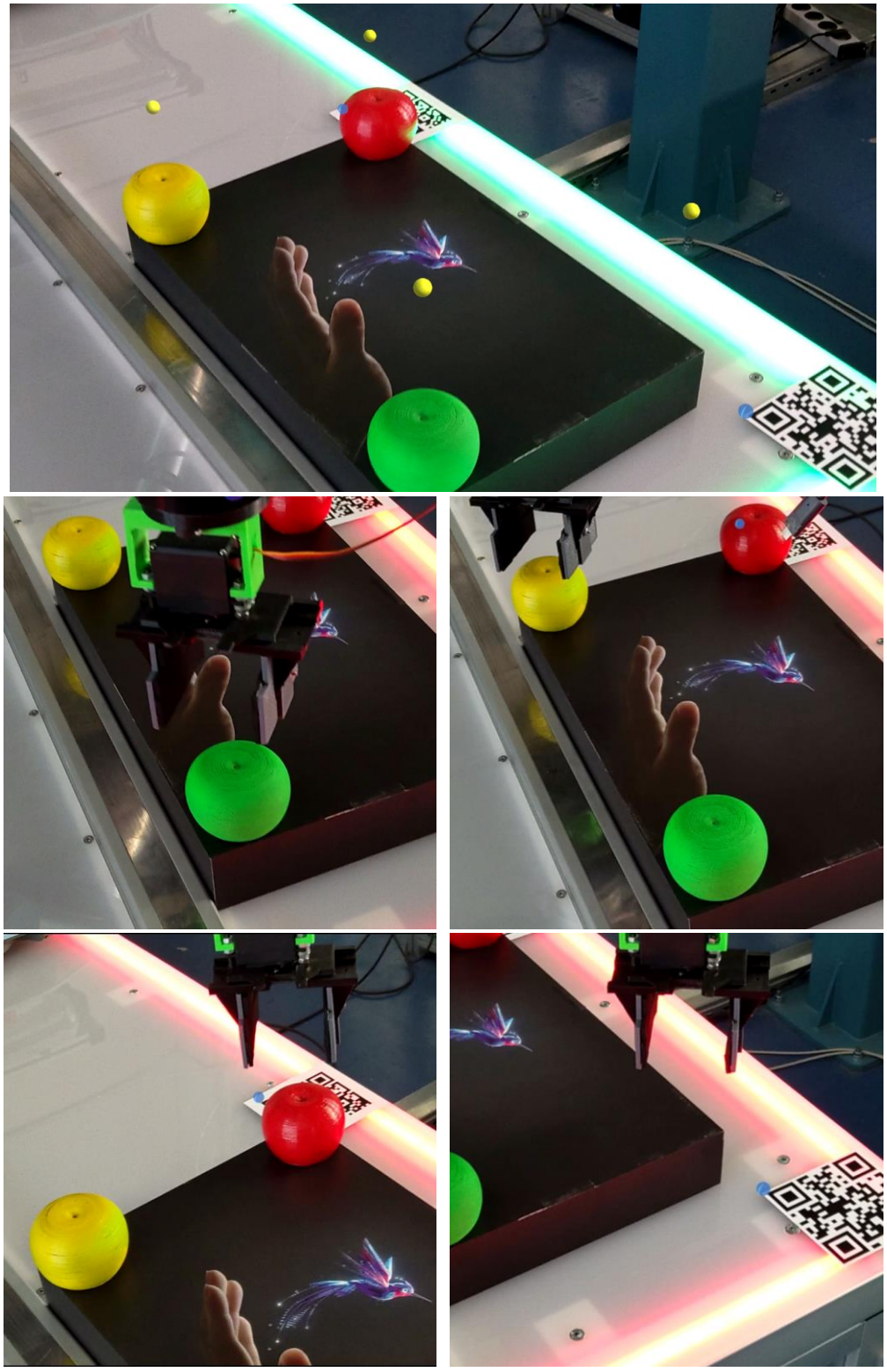


Ilustración 69. Ensayo experimental del modo Trayectory

4.2.2.4 Pick & Place

Por último, el modo Pick & Place supone una cierta habilidad del usuario para indicar el punto exacto que desea enviar al robot. Esto es debido a la precisión con la que el usuario es capaz de señalar justamente la posición en el espacio que desea enviar. No obstante, la ubicación de esta esfera no se encuentra exactamente en la punta del dedo índice, sino entre la falange distal y la media.

La esfera dibujada en el dedo índice del usuario sirve para guiar al usuario en la marcación de los puntos, de la tal manera que, donde esta se sitúe, la aplicación registrará su ubicación en el espacio. El operario debe de tener en cuenta esta esfera a la hora de indicar un punto en el espacio, ya que la coordenada enviada tendrá el valor de la ubicación de la esfera en el instante de tiempo en el que se active el comando de voz «Set Point»

Es preciso apuntar que esta esfera aparece proyectada en el dedo índice de la mano derecha siempre que se esté un modo de operación (Real Time, Trayectory o Pick & Place) activado, pero es en el Pick & Place cuando es más diferencial la indicación del punto que permita recoger y depositar un objeto.

El robot realiza una trayectoria MoveJ hasta la ubicación indicada, pero a 20 milímetros de altura respecto de esta. Posteriormente, realiza un movimiento lineal hasta la posición de la manzana y, se abre un socket entre la controladora del robot y el Arduino UNO en el que se envía un mensaje con la información para cerrar las pinzas. El robot vuelve al reposo y permanece a la espera del usuario para que indique la coordenada Place donde depositar el objeto.

Si el usuario selecciona la opción Loop, el robot cogerá el objeto que acaba de depositar y lo volverá a dejar en la posición donde lo había recogido previamente. Este ciclo se repetirá indefinidamente hasta que el usuario active el comando de voz «Stop».

A la hora de comprobar si este procedimiento se cumple, se ha observado que el ciclo de ida y vuelta se completa siempre que el Pick & Place se complete antes del siguiente mensaje que se publique en el topic /HLloop.

En el ejemplo mostrado en la Ilustración 70, se marca la ubicación de la parte superior de una manzana impresa en 3D. Esta coordenada es tomada en la referencia de HL2 y es transformada a la referencia del robot. No obstante, solo son enviados los valores en la dirección X e Y de este vector, ya que la altura se toma directamente de la mesa de trabajo.

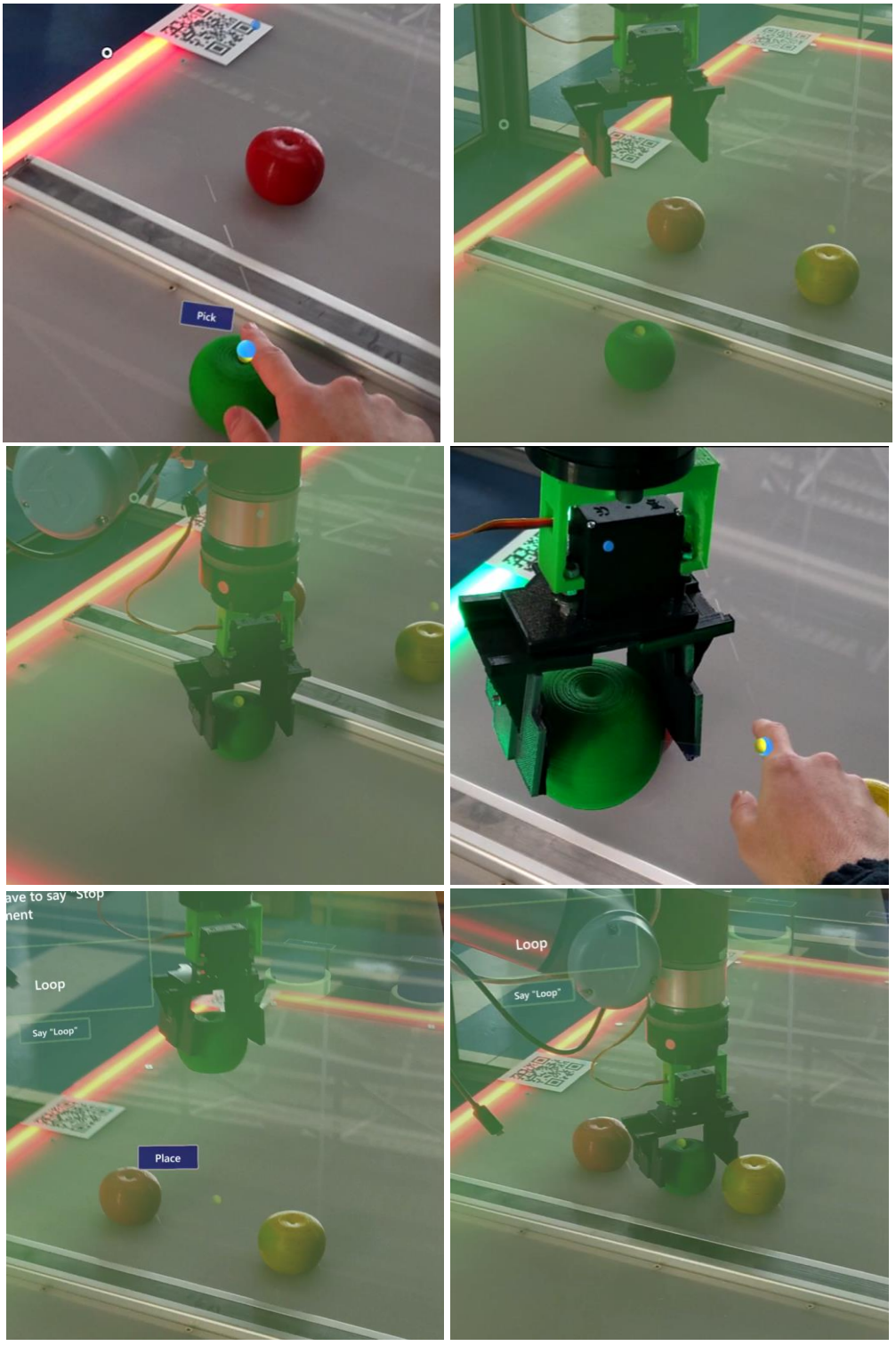


Ilustración 70. Ensayo experimental del modo Pick & Place

4.2.2.5 Coordenada fuera de los límites

En el momento en el que el operario indique una coordenada fuera del espacio de trabajo del robot, se proyectará el diálogo de advertencia que

informa al usuario de que la coordenada indicada se encuentra fuera de los límites del robot colaborativo.

En el ejemplo mostrado en la Ilustración 71 se muestra una situación en la que el usuario indica un punto fuera de los límites del robot. En ese instante, el espacio de trabajo digital del UR5e cambia a color rojo y, se proyecta el diálogo Warning en las lentes de HL2. Cuando el usuario seleccione una de las opciones que muestra este diálogo, el espacio de trabajo volverá a cambiar a color verde.

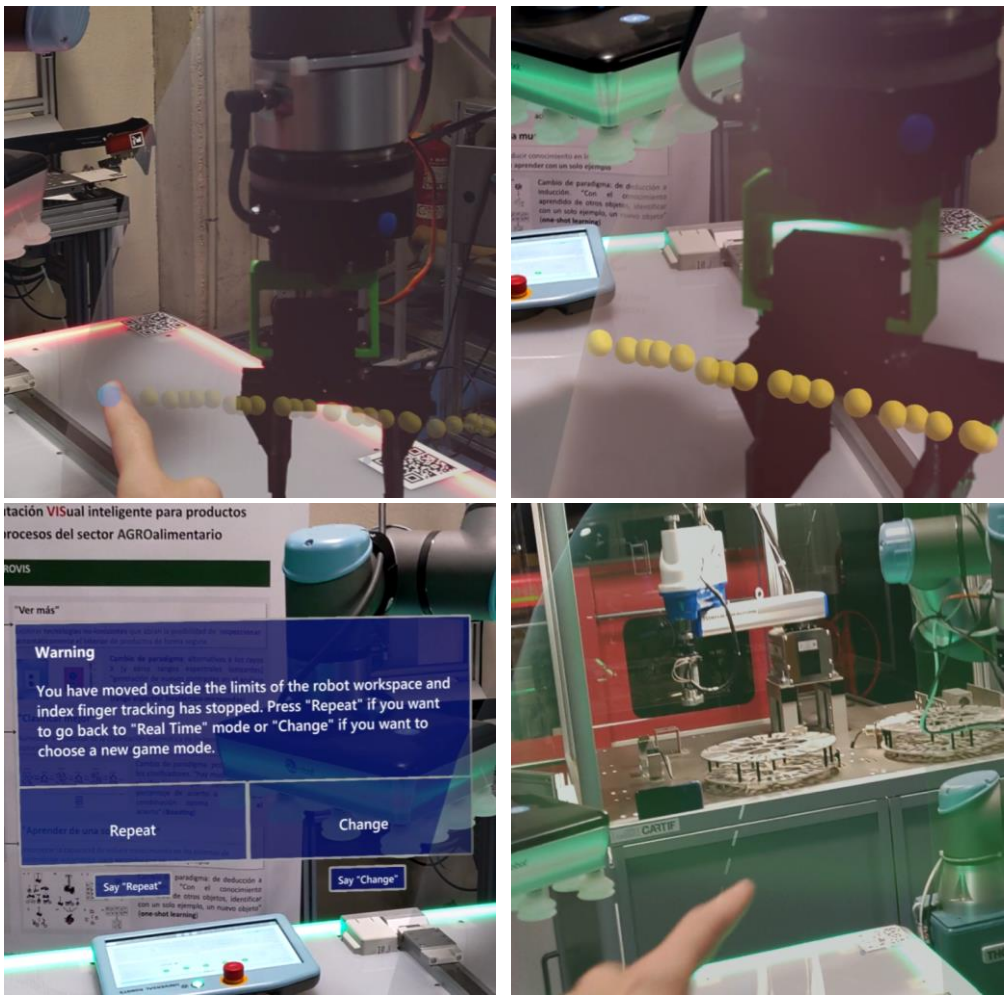


Ilustración 71. Indicación de un punto fuera de los límites del UR5e

4.3 Evaluación de la precisión y exactitud

La medición del sistema de realidad mixta es fundamental para evaluar el desempeño de la aplicación. Se ha analizado la precisión y exactitud del sistema realizando ensayos que presenten evidencias físicas de los posibles errores de la instrumentación de medida.

Por otro lado, se ha evaluado la experiencia de usuario con la aplicación de realidad mixta, realizando la misma actividad con diferentes voluntarios. De esta manera, se han hallado resultados cualitativos y cuantitativos que reflejan el grado de validez del sistema a la hora de realizar maniobras con un robot colaborativo sin tener conocimientos previos en programación robótica.

4.3.1 Error del robot

Se supone un error de precisión y exactitud nulo para el robot, ya que es la propia controladora del robot el que dirige al robot a las posiciones que se indican en su Flex Pendant. No obstante, se ha evaluado que existiese algún tipo de interferencia en la transmisión de los datos desde ROS a la controladora.

El estudio realizado para validar la precisión y exactitud del robot fue basado en la comparación de los datos medidos con el radio de fusión que se le proporciona como argumento en las instrucciones enviadas al UR5e desde ROS.

Normalmente el valor que se emplea para el radio de fusión es nulo en los modos Trayectory y Pick & Place y 0.01 metros en el modo Real Time. Este comando se introduce para suavizar las trayectorias, tal como se representa en la Ilustración 72.

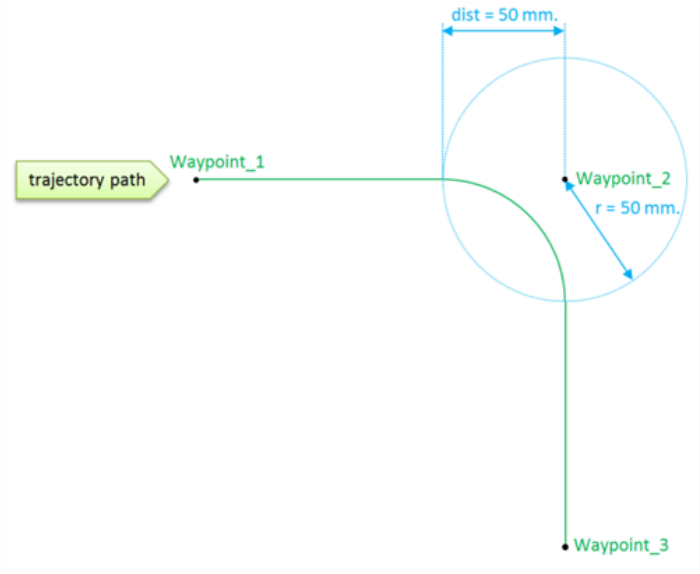


Ilustración 72. Representación del radio de fusión en una trayectoria por puntos (Fuente: [34])

Además, los datos tomados a la hora de medir las coordenadas de los códigos QR en la referencia del robot han sido proporcionados por la pantalla de la Flex Pendant del UR5e. Por ello, es necesario comprobar si estas medidas pueden tener algún tipo de error.

Se ha elaborado un programa de C++ que envía vía socket TCP/IP un URScript al robot colaborativo con una instrucción MoveJ con los siguientes argumentos:

- pose: Posición y orientación del TCP [m y rad]
- a: aceleración de la herramienta [m/s^2]
- v: velocidad de la herramienta [m/s]
- t: tiempo [S]
- r: radio de fusión [m]

Cuando el UR5e ha recibido y procesado el fichero, el robot se ha movido hasta la posición indicada. En ese momento, se han apuntado las coordenadas resultantes mostradas en la Flex Pendant. Este proceso se ha realizado con cinco puntos diferentes y repetido cinco veces para cada uno.

Los resultados obtenidos mostraron un error relativo del **0%** cuando el radio de fusión tenía un valor de 0 metros y del **0,5%** cuando tenía un valor de 0,01 metros. Esto demuestra una gran exactitud del robot colaborativo a la hora de acceder a los puntos que se le indican.

La **desviación estándar** de la muestra medida cuando el radio de fusión es de 1 centímetro fue de **0,05 milímetros.** Por lo tanto, se considera un error del robot nulo a la hora de replicar las maniobras enviadas.

Es preciso indicar que se supone que el sensor que realiza la toma de los datos del robot no muestra errores en sus mediciones.

4.3.2 Error de Microsoft Hololens 2

La precisión con la que el dispositivo de realidad mixta captura las coordenadas que se indican con el dedo índice depende de varios factores que dependen del usuario como:

- La velocidad de la mano.
- La orientación.
- La orientación del dispositivo.
- El tamaño de la mano.
- La posición de la mano dentro del campo de visión del dispositivo.
- Etc.

Cuantificar la influencia de cada uno de estos factores en las medidas de los datos es una tarea bastante tediosa, ya que requiere cierta complejidad abordar cada uno de estos de forma separada. Por ello, el estudio realizado se ha centrado en evaluar la precisión de los datos registrados por la

aplicación de realidad mixta cuando el usuario graba coordenadas sin mover la mano.

Esto es debido a que el factor clave en la medida de las coordenadas se basa en el punto que detecta HL2 como punta del dedo índice de la mano del usuario.

El procedimiento realizado se basa en disponer la mano derecha en un lugar del espacio y abrirla, enfocando con la mirada el dedo índice. Posteriormente, el usuario hace uso del comando de voz «Set Point» para publicar en un topic de ROS las coordenadas grabadas por la aplicación de realidad mixta.

Un programa ejecutado en ROS recopila la información procedente de este topic y genera un fichero en el que se escriban las coordenadas recibidas.

Este proceso se ha repetido para cuatro posiciones diferentes de la mano, y se han guardado 20 vectores con los valores de las coordenadas tomadas por la aplicación de realidad mixta. Los resultados obtenidos muestran una **desviación estándar** de **1,4 milímetros** con un rango de 5 milímetros en la dispersión de los datos. Esto plasma la gran precisión del sistema de medida del HL2.

Al no disponer de la referencia real en las medidas, no se puede cuantificar la exactitud real del sistema. No obstante, sí se puede determinar una aproximación de la medida real realizando la media de las muestras tomadas. Con esto, se calcula un **error relativo del 0,23%**, lo que demuestra también una gran exactitud por parte del sistema de medida.

4.3.3 Error global

Por último, se ha analizado el error global del sistema. Este error se compone por la suma del error del robot, el error del dispositivo HL2 y el error realizado en las transformaciones.

El experimento que se ha llevado a cabo ha consistido en trazar una flecha en un lugar de la mesa de trabajo para que el usuario señalase con el dedo índice de la mano. Este usuario indicaba con el dedo índice esta posición y daba la orden de publicar esta coordenada en un topic de ROS.

Se ha desarrollado un programa en la terminal de ROS cuya función es transformar esta coordenada en una instrucción MoveJ para enviársela al UR5e vía socket TCP/IP. De esta manera, el TCP del robot se traslada hacia esta posición con la pose indicada. El procedimiento se ha repetido en veinte ocasiones para veinte coordenadas diferentes.

Para determinar las medidas reales que proporcionaba la Flex Pendant, se ha colocado una punta de plástico fabricada en una impresora 3D en la muñeca del robot, tal como se aprecia en la Ilustración 73.

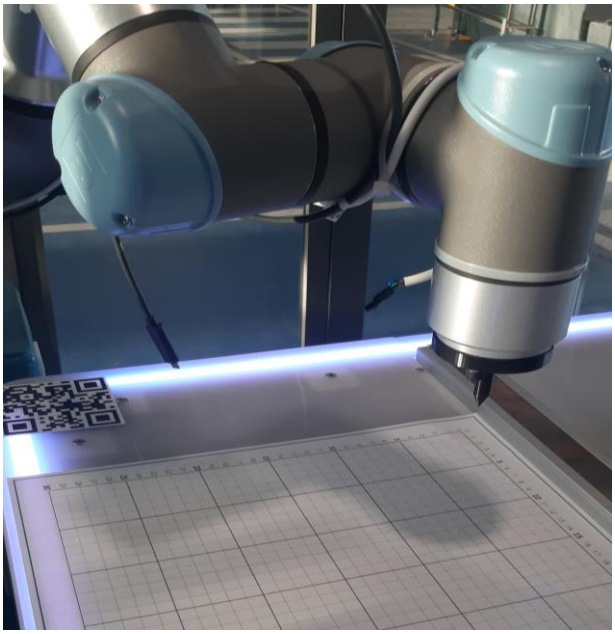


Ilustración 73. UR5e con una punta de plástico en la muñeca

Los resultados obtenidos muestran un error global del **1,31%** y una desviación estándar de **7 milímetros.**

Lógicamente, esto supone un decremento considerable en la exactitud y precisión del sistema. No obstante, en estos resultados se ha tenido en cuenta el factor clave para validar las medidas, que es la habilidad del usuario para indicar el punto en el espacio que desea enviar al robot.

En conclusión, se puede decir que el sistema es bastante preciso y exacto para indicar maniobras que no requieran gran habilidad por parte del usuario para señalar coordenadas con un margen muy pequeño, es decir, con escala milimétrica.

Por ejemplo, un buen uso de esta aplicación para obtener un resultado notable sería el Pick & Place, pero no para realizar trayectorias milimétricas tales como soldaduras o taladrados.

4.3.4 Ensayos UX

Se ha examinado a 10 voluntarios que han querido probar el sistema de realidad mixta. Para ello, se ha planificado la misma actividad para todos. De esta manera, se ha conseguido evaluar objetivamente la experiencia de usuario con la aplicación y comprobar si los objetivos propuestos se cumplen.

La actividad que han realizado los usuarios ha seguido cronológicamente los siguientes pasos:

1) Exposición de la actividad y descripción del funcionamiento básico de HL2.
2) Inicio de la aplicación de realidad mixta e interacción con los diálogos digitales. Se ha descrito previamente a los usuarios el mecanismo de interacción con los objetos digitales, indicando la posibilidad de hacerlo manualmente o mediante comandos de voz.
3) Definición del sistema de coordenadas de la mesa de trabajo. Se ha informado al usuario de la forma correcta de escanear los códigos QR, pero ha sido este el que ha completado el proceso.
4) Navegación por los siguientes diálogos. Se ha descrito rápidamente al usuario el propósito de introducir el espacio de trabajo del robot colaborativo en el entorno de la aplicación.
5) Descripción de los pulsadores. Se ha indicado al voluntario que pulse el botón Repose.
6) Descripción del menú. Se ha orientado al voluntario para que pulse el botón Real Time y lea el panel que describe el procedimiento para interactuar con el robot en este modo. Se ha dejado libertad al usuario para realizar cualquier maniobra con el robot colaborativo.
7) Se ha indicado al voluntario que pulse el botón Trayectory y lea el panel que describe el procedimiento para interactuar con el robot en este modo. El usuario ha tenido que establecer una trayectoria con cuatro puntos que forman una figura de cuadrado. Posteriormente, se ha indicado que se estableciese la trayectoria en MoveL.
8) Se ha orientado al voluntario para que pulse el botón Pick & Place y lea el panel que describe el procedimiento para interactuar con el robot en este modo. El usuario ha tenido que señalar la posición de una de las manzanas fabricadas en una impresora 3D para cogerlo y llevarlo a otra posición situada al otro lado de una guía situada en la mesa de trabajo. Posteriormente, se ha indicado al usuario que realice esta maniobra en bucle pulsando el botón Loop.
9) Se ha indicado al voluntario que pare la maniobra y lleve al robot al Home.
10) Fin de la actividad.

Esta actividad ha sido cronometrada por cada voluntario para evaluar la asistencia remota que es capaz de ofrecer el sistema para hacer frente a tareas cotidianas. De esta forma, se puede garantizar si la aplicación de realidad mixta es intuitiva y eficaz.

La media de tiempos que los 10 voluntarios han obtenido ha sido de **13 minutos y 14 segundos**. No obstante, el tiempo dedicado a la actividad se ha visto reducido en gran medida la segunda vez que se repiten los ensayos. Usuarios afianzados con la aplicación han dedicado un tiempo de **5-6 minutos** para completar las diferentes maniobras que constituyen la actividad.

Además, se ha elaborado un pequeño formulario con ocho preguntas en las que cada voluntario ha podido evaluar cuantitativamente la experiencia con el sistema de realidad mixta. Los resultados obtenidos se muestran a continuación:

1. **¿La aplicación de realidad mixta te ha parecido lo suficientemente intuitiva como para desarrollar una actividad con el robot sin necesidad de soporte externo? (0: Imposible 5: Perfectamente)**
    - Promedio: **3,8**
    - Desviación estándar: **1,03**
    - Observaciones: Se ha dado soporte para indicar el orden de detección de los códigos QR o para informar sobre el objetivo de utilizar o no un botón digital.
2. **¿La interacción con los objetos digitales te ha resultado más efectiva utilizando las manos o los comandos de voz?**
    - El **60%** de los usuarios prefieren la **navegación manual**, mientras el **40%** restante mediante **comandos de voz.**
    - Observaciones: Se ha visto que la aplicación es sensible al ruido exterior, por lo tanto, había ocasiones en las que no se procesaban los comandos de voz utilizados, recurriendo a elevar la voz por parte de los usuarios. También se han visto problemas a la hora de intentar pulsar botones de los diálogos digitales, ya que los usuarios utilizaban diferentes métodos para interaccionar con estos.
3. **¿Consideras eficaz el uso de objetos digitales como pulsadores o el menú de botones a la hora de llevar a cabo la actividad? (0: Definitivamente no 5: Ideal)**
    - Promedio: **4,5**
    - Desviación estándar: **0,7**
    - Observaciones: El 60% de los usuarios ha indicado la máxima nota.
4. **¿Cómo consideras el seguimiento del dedo índice del modo Real Time? (0: Nulo 5: Tiempo Real)**
    - Promedio: **3,8**
    - Desviación Estándar: **0,91**
    - Observaciones: Se ha visto que la mayor parte de los usuarios en su experiencia con este modo de actividad acababan por

salirse de los límites del espacio de trabajo del robot debido al temor de que el robot pudiese colisionar con su mano derecha o por la inconsciencia de las dimensiones de la zona de trabajo.

5. **¿Cómo consideras la capacidad del sistema para programar trayectorias por puntos? (0: El robot no reproduce la trayectoria indicada 5: El robot reproduce perfectamente la trayectoria indicada)**
    - Promedio: **3,9**
    - Desviación estándar: **0,73**
    - Observaciones: Algunos usuarios han señalado la posibilidad de modificar o eliminar algunos puntos indicados.
6. **¿El modo Pick & Place te resulta útil a la hora de coger y depositar un objeto en diferentes posiciones? (0: Inútil 5: Muy útil)**
    - Promedio: **4,5**
    - Desviación estándar: **0,7**
    - Observaciones: La precisión por parte del usuario para señalar un punto situado encima de las manzanas ha sido un factor clave en el resultado de esta prueba.
7. **¿Las tiras de luces LED RGB te resultan útiles a la hora de realizar la actividad? (0: Inútil 5: Muy útil)**
    - Promedio: **4,1**
    - Desviación estándar: **0,87**
    - Observaciones: Un detalle que ha gustado a la mayoría de los voluntarios, pero que no consideran esencial para el desarrollo de la actividad.
8. **Valora tu experiencia en general con el sistema de realidad mixta. (0: Muy mala 5: Muy buena)**
    - Promedio: **4,2**
    - Desviación estándar: **0,63**
    - Observaciones: Al final de la actividad los usuarios han quedado bastante satisfechos.

A efectos de estos resultados, se puede considerar que el sistema de realidad mixta desarrollado es lo suficientemente intuitivo y completo como para programar maniobras con un robot colaborativo sin necesidad de tener conocimientos previos en programación robótica.

Capítulo 5. Gestión del trabajo

La planificación temporal del proyecto abarca un periodo de tiempo de 10 meses. El diagrama de Gantt mostrado en la Ilustración 74 agrupa en diferentes fases la dedicación temporal de las tareas realizadas en este TFG.

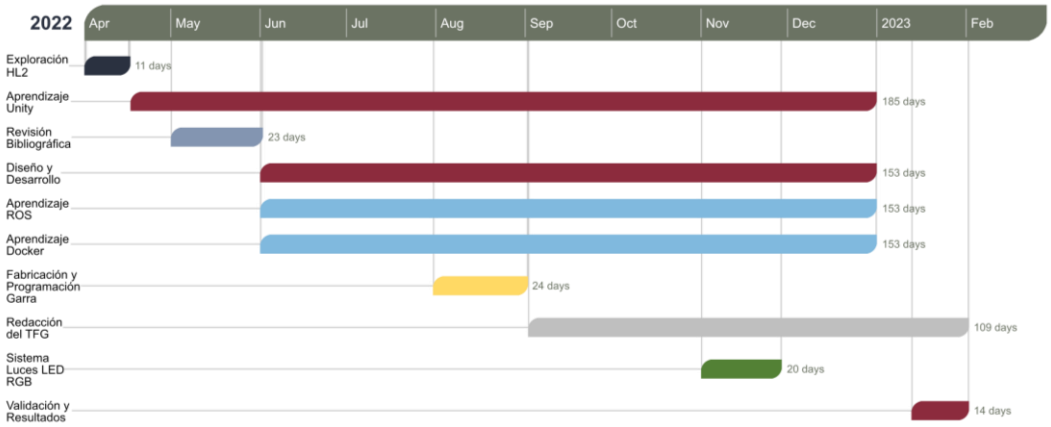


Ilustración 74. Diagrama de Gantt de la planificación temporal del proyecto

El proyecto comenzaba con la exploración del dispositivo Microsoft Hololens 2, indagando en sus funcionalidades y probando algunas de las aplicaciones que vienen instaladas por defecto en el dispositivo.

Una vez se iba familiarizándose con el dispositivo, se iba realizando la búsqueda de plataformas de desarrollo para desarrollar aplicaciones de realidad mixta. Unity fue la plataforma elegida. Después de seguir algunos tutoriales sobre la creación de aplicaciones de realidad mixta, se planteaban los objetivos del proyecto.

Posteriormente, fueron analizadas las soluciones de creadores que hubieran desarrollado proyectos del mismo carácter que este para tratar de encontrar diferentes rutas para construir el sistema de realidad mixta pretendido.

El proceso de desarrollo de este sistema ha sido realizado simultáneamente con el aprendizaje de diferentes herramientas para cumplir los objetivos como son ROS y Docker.

En los últimos meses ha sido añadido el sistema de luces LED RGB y se han realizado diferentes pruebas que mostrasen señales de su rendimiento y validasen si los objetivos han sido cumplidos.

Capítulo 6. Conclusiones y líneas futuras

En este último capítulo se extraen las conclusiones principales del proyecto y se presentan los futuros objetivos a los que hace frente el sistema de realidad mixta desarrollado.

6.1 Conclusiones del proyecto

El proyecto planteado ha cumplido el objetivo principal que se ha propuesto en la fase de planteamiento de manera que usuarios sin experiencia en programación robótica han podido realizar maniobras con un robot colaborativo con ayuda de elementos externos como el sistema de luces LED integrado en la mesa del trabajo del robot.

El método de detección de códigos QR ha sido esencial para transformar las maniobras programadas por el usuario al lenguaje nativo del robot. De esta manera, se ha logrado que usuario puedan trazar trayectorias con un con un margen de error en de tan solo 7 milímetros.

La asistencia remota ha sido clave para cumplir este propósito, ya que la interfaz digital que se proyecta en el dispositivo de realidad mixta ayuda y guía al usuario a la hora de programar las instrucciones que desea enviar al robot. Para ello, se ha hecho uso de diálogos y paneles digitales que incorporan información en forma de texto sobre los gestos y comandos de voz que se deben de utilizar en cada momento de la navegación.

Se puede decir que las indicaciones del usuario son mandadas al robot en un plazo relativamente corto gracias a la comunicación vía socket TCP/IP entre los dispositivos del sistema. El papel de ROS ha sido clave para gestionar la transmisión de datos dentro del sistema.

Además, se ha demostrado que la aplicación es lo suficientemente intuitiva y segura como para guiar a un operario en las labores que se plantean, valorando la experiencia de usuario con un promedio del 4,2/5 por parte de los voluntarios evaluados en las pruebas experimentales.

Con todo esto, se ha conseguido automatizar actividades propias de la Industrial 4.0 que permitan realizar procesos mediante el uso de la realidad mixta y empaquetar el sistema al completo en un contenedor de Docker que ofrezca portabilidad y simplicidad a la hora de ejecutarlo.

6.2 Propuestas de mejora y trabajos futuros

El sistema de realidad mixta desarrollado se va a presentar como hito principal de Cartif en el proyecto Cervera5R para que las otras empresas participes en este puedan incorporarlo a sus fábricas piloto. Para ello, el sistema se empaquetará en un contenedor de Docker con la solución de la

aplicación de realidad mixta y el paquete de ROS que realice el tratamiento de datos entre el dispositivo de realidad mixta y el robot colaborativo.

Se elaborará un manual de usuario que ofrezca soporte a las empresas que dispongan del soporte físico y lógico que requiere este sistema para integrarlo fácilmente en sus instalaciones. Además, se impartirán una serie de videoconferencias en las que se tratarán aspectos relacionados con este proyecto para mejorar y evaluar este sistema una vez haya sido implantado por cada miembro.

Con todo esto, se incrementará el potencial de este sistema con el fin de cubrir las necesidades de las empresas y ofrecer un marco de cooperación que ayude a desarrollar nuevas tecnologías propias de la Industria 4.0.

Bibliografía

[1] Christopher Mims and Michael Bucher, «100 Years of Robots». Available: https://www.wsj.com/story/100-years-of-robots-d44df980 (Accedido 12 octubre 2022)

[2] Natalie Craig, «Innovaciones robóticas generan oportunidades para fabricantes de maquinaria original». Available: https://www.proquest.com/abitrade/docview/2205514514/F07BBA4DB7AF4DC0PQ/1?accountid=14778 (Accedido 12 octubre 2022)

[3] Universidad de Deusto «Industria 4.0: La 4ª Revolución Industrial». Available: https://www.youtube.com/watch?v=Z2DK6QJX1rs&ab_channel=UniversidaddeDeusto%2FDeustukoUnibertsitatea (Accedido 13 octubre 2022)

[4] Ernesto Domingo Fuentes, «Diseño e implementación de una aplicación de previsión del tiempo mediante realidad aumentada usando la tecnología Hololens de Microsoft». Available: https://tauja.ujaen.es/handle/10953.1/14042 (Accedido 13 octubre 2022)

[5] Athena Information Solutions Pvt. Ltd, «5G advancing extended reality to the next level». Available: https://www.proquest.com/abitrade/docview/2661105124/fulltext/B58A7D0049884505PQ/1?accountid=14778 (Accedido 20 octubre 2022)

[6] El día de Valladolid «Cartif lidera un proyecto para mitigar el cambio climático». Available: https://www.eldiadevalladolid.com/Noticia/zc9e53d74-cf29-d85f-10b80dfbd4920427/202106/Cartif-lidera-un-proyecto-para-mitigar-el-cambio-climatico (Accedido 23 octubre 2022)

[7] CNN Expansión, «Panorama de la digitalización en México». Available: https://expansion.mx/empresas/2022/10/03/industria-4-0-la-realidad-virtual-y-aumentada-llegan-a-la-planta (Accedido 2 noviembre 2022)

[8] Telefónica «Diferencias entre Realidad Virtual, Aumentada y Mixta. ¿Cuáles son?». Available: https://www.telefonica.com/es/sala-comunicacion/blog/diferencias-entre-realidad-virtual-aumentada-y-mixta/

[9] Editec «Realidad virtual, aumentada y mixta, Qué son y diferencias». Available: https://editeca.com/realidad-virtual-aumentada-y-mixta-que-son-y-en-que-se-diferencian/ (Accedido 4 noviembre 2022)

[10] Avi Barel «Las diferencias entre VR, AR y MR» . Available: https://medium.com/startux-net/the-differences-between-vr-ar-mr-27012ea1c5 (Accedido 4 noviembre 2022)

[11] TIC Negocios «Caminar con éxito hacia la Industria 4.0: Capítulo 18 – Realidad Aumentada y Virtual». Available: https://ticnegocios.camaravalencia.com/servicios/tendencias/caminar-con-exito-hacia-la-industria-4-0-capitulo-18-realidad-aumentada-y-virtual/ (Accedido 10 noviembre 2022)

[12] Turboquid «Microsoft Hololens 2 modelo 3d». Available: https://www.turbosquid.com/es/3d-models/microsoft-hololens-2-model-1395940 (Accedido 20 noviembre 2022)

[13] Realovirtual «Ya conocemos el campo de visión de Hololens 2» . Available: https://www.realovirtual.com/noticias/6475/ya-conocemos-campo-vision-hololens-2#:~:text=El%20visor%20tiene%2052%20grados,43%C2%BA%20y%20vertical%20de%2029%C2%BA. (Accedido 22 noviembre 2022)

[14] Xataka «Las Magic Leap 2 llegarán en septiembre, pero costarán un dineral (y probablemente no son para ti». Available: https://www.xataka.com/realidad-virtual-aumentada/magic-leap-2-llegaran-septiembre-a-usuarios-empresariales-costaran-dineral#:~:text=En%20Magic%20Leap%20indican%20que,de%20refresco%20de%20120%20Hz. (Accedido 22 noviembre 2022)

[15] Revista de Robots «Robots colaborativos. Qué es un robot colaborativo, características y fabricantes de cobots». Available: https://revistaderobots.com/cobots/cobots-o-robots-colaborativos-caracteristicas-ventajas-y-fabricantes-de-brazos-roboticos-industriales/ (Accedido 29 noviembre 2022)

[16] Gabriela Torres «Cobots-humanos: innovación codo a codo». Available: https://urany.net/blog/cobots-humanos-innovaci%C3%B3n-codo-a-codo (Accedido 29 noviembre 2022)

[17] Seeking Alpha «Cobots: The PCs of The Robot Era». Available: https://seekingalpha.com/article/4174021-cobots-pcs-of-robot-era (Accedido 1 diciembre 2022)

[18] Wikipedia Alpha «Universal Robots». Available: https://en.wikipedia.org/wiki/Universal_Robots (Accedido 1 diciembre 2022)

[19] Universal Robots, Products. Available: https://www.universal-robots.com/es/productos/ (Accedido 1 diciembre 2022)

[20] QUT Robot Academy «Lesson: Robot Workspace». Available: https://robotacademy.net.au/lesson/robot-workspace/ (Accedido 2 diciembre 2022)

[21] Manual de usuario del robot UR5e. Available: https://cfzcobots.com/wp-content/uploads/2018/06/UR5e_User_Manual_es_Global-reducido.pdf (Accedido 2 diciembre 2022)

[22] Wikipedia, ABB . Available: https://es.wikipedia.org/wiki/ABB (Accedido 5 diciembre 2022)

[23] ABB, Robots colaborativos. Available: https://new.abb.com/products/robotics/es/robots-colaborativos (Accedido 10 diciembre 2022)

[24] KUKA, LBR iiwa. Available: https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/robot-industrial/lbr-iiwa (Accedido 10 diciembre 2022)

[25] Direct Industry, Robot articulado LBR iiwa series. Available: https://www.directindustry.es/prod/kuka-ag/product-17587-1650349.html (Accedido 11 diciembre 2022)

[26] Auto-Revista.com . Available: https://www.auto-revista.com/texto-diario/mostrar/3820828/kuka-potencia-oferta-robotica-movil-colaborativa (Accedido 11 diciembre 2022)

[27] D. Aschenbrenner , Jonas SI Rieder et al. «Mirrorlabs - creating accessible Digital Twins of robotic production environment with Mixed Reality» . Available: https://www.semanticscholar.org/paper/Mirrorlabs-creating-accessible-Digital-Twins-of-Aschenbrenner-Rieder/654cd9284c503738d36e0140f52ed0e82fd487a7 (Accedido 17 diciembre 2022)

[28] Dimitris Mourtzis, John Angelopoulos y Nikos Panopoulos. «Closed-Loop Robotic Arm Manipulation Based on Mixed Reality». Available: https://www.mdpi.com/2076-3417/12/6/2972/pdf (Accedido 17 diciembre 2022)

[29] Inês de Oliveira Soares «Programming Robots by Demonstration using Augmented Reality» . Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8434657/ (Accedido 20 diciembre 2022)

[30] Amazon. Available: https://www.amazon.es/ (Accedido 22 diciembre 2022)

[31] LEDBOXBlog «¿Qué es el control DMX512?». Available: https://blog.ledbox.es/que-es-control-dmx512/ (Accedido 22 diciembre 2022)

[32] SIMATIC IOT2050 - Industry Mall - Siemens Spain. Available: https://mall.industry.siemens.com/mall/es/es/Catalog/Products/10373416 (Accedido 23 diciembre 2022)

[33] NAYLAMP Mechattonics «Tutorial Módulo Ethernet ENC28J60 y Arduino» . Available: https://naylampmechatronics.com/blog/17_tutorial-modulo-ethernet-enc28j60-y-arduino.html (Accedido 26 diciembre 2022)

[34] ROS:Home . Available: https://www.ros.org/ (Accedido 26 diciembre 2022)

[35] Rometobots Blog «¿Conceptos básicos de ROS» ? Available: htt5s://jjromeromarras.wordpress.com/2014/08/27/conceptos-basicos-de-ros/ (Accedido 3 enero 2022)

[36] Wikipedia «Docker». Available: https://es.wikipedia.org/wiki/Docker_(software) (Accedido 4 enero 2022)

[37] Digital Guide IONOS «WSL2: Presentación del susbsitema de Windows para Linux 2». Available: https://www.ionos.es/digitalguide/servidores/know-how/wsl2/ (Accedido 4 enero 2022)

[38] Arsys «¿Qué es Docker y qué ventajas tiene trabajar con sus contenedores?». Available: https://www.arsys.es/blog/docker-ventajas-contenedores (Accedido 8 enero 2022)

[39] ATLASSIAN «¿Comparación de contenedores y máquinas virtuales?» . Available: https://www.atlassian.com/es/microservices/cloud-computing/containers-vs-vms#:~:text=La%20diferencia%20clave%20entre%20los,del%20nivel%20del%20sistema%20operativo. (Accedido 9 enero 2022)

[40] Microsoft «Visual Studio» . Available:https://learn.microsoft.com/es-es/visualstudio/get-started/visual-studio-ide?view=vs-2022 (Accedido 9 enero 2022)

[41] FullWat Blog «Protocolo DMX (Parte 1: Funcionamiento)». Available: http://blog.fullwat.com/funcionamiento-del-protocolo-dmx/ (Accedido 11 enero 2022)

[42] Xataka «Máquinas virtuales: qué son, cómo funcionan y cómo utilizarlas» . Available: https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas (Accedido 12 enero 2022)

[43] IEBS School «Qué es la UX y la UI». Available: https://www.iebschool.com/blog/que-es-la-ux-y-la-ui-analitica-usabilidad/ (Accedido 14 enero 2022)

[44] Acumbamail «UI vs UX: ¿qué diferencias existen?» . Available: https://acumbamail.com/blog/ui-vs-ux/?gclid=Cj0KCQiAm5ycBhCXARIsAPldzoWXFSpZ64XDHOu58EhRChhu7VdubMWYXqBuiwY4QDgd7tilwEswjTIaAkkyEALw_wcB (Accedido 16 enero 2022)

[45] Microsoft Learn «Aspectos básicos de HoloLens 2: desarrollo de aplicaciones de realidad mixta?». Available: https://learn.microsoft.com/es-es/training/modules/learn-mrtk-tutorials/1-1-introduction (Accedido 18 enero 2022)

[46] Microsoft Learn «Instalación de las herramientas» . Available: https://learn.microsoft.com/es-es/windows/mixed-reality/develop/install-the-tools (Accedido 18 enero 2022)

[47] Microsoft Learn «Desarrollo de Unity para HoloLens». Available: https://learn.microsoft.com/es-es/windows/mixed-reality/develop/unity/unity-development-overview?tabs=arr%2CD365%2Chl2 (Accedido 20 enero 2022)

[48] Aula21 «.NET: Qué es y cómo funciona». Available: https://www.cursosaula21.com/que-es-net/ (Accedido 22 enero 2022)

[49] GlitchEnzo «NuGetForUnity» . Available: https://github.com/GlitchEnzo/NuGetForUnity (Accedido 28 enero 2022)

[50] Microsoft Learn «Archivo de solución (.sln)». Available: https://learn.microsoft.com/es-es/visualstudio/extensibility/internals/solution-dot-sln-file?view=vs-2022 (Accedido 28 enero 2022)