

TRABAJO FIN DE GRADO

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Programación de robots manipuladores
mediante Realidad Virtual

Curso 2023/2024

Alumno/a:

Antonio Martínez Navarro

Director/es:
José Carlos Moreno Úbeda
Fernando Cañadas Aránega



UNIVERSIDAD DE ALMERÍA
ESCUELA SUPERIOR DE INGENIERÍA



GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y
AUTOMÁTICA

TRABAJO FIN DE GRADO

Programación de robots manipuladores mediante Realidad Virtual

Alumno: Antonio Martínez Navarro
Director: José Carlos Moreno Úbeda
Codirector: Fernando Cañas Aránega
Fecha: Julio 2024

Dedicado a mis abuelos.

Índice general

| | Página |
|--|-------------|
| Agradecimientos | IX |
| Resumen | XI |
| Abstract | XIII |
| Siglas y acrónimos | XV |
| Índice de figuras | XIX |
| Índice de tablas | XXI |
| 1 Introducción | 3 |
| 1.1 Introducción e interés | 3 |
| 1.2 Motivación del TFG | 6 |
| 1.3 Objetivos | 6 |
| 1.4 Contexto | 7 |
| 1.5 Resumen de resultados | 9 |
| 1.6 Fases de desarrollo y planificación | 9 |
| 1.7 Competencias empleadas en este trabajo | 11 |
| 1.7.1 Competencias básicas, generales y transversales | 11 |
| 1.7.2 Competencias específicas del título | 12 |
| 1.8 Estructura del TFG | 14 |
| 2 Revisión bibliográfica | 15 |
| 2.1 Introducción | 15 |
| 2.2 Robótica industrial en la actualidad. | 17 |
| 2.3 Métodos de programación y control de un robot real | 22 |
| 2.4 Conceptos y desarrollo de la RV | 24 |
| 2.4.1 Realidad Virtual (RV) | 24 |
| 2.4.2 Realidad Aumentada | 26 |
| 2.4.3 Realidad Mixta | 28 |
| 2.5 Aplicaciones de la realidad virtual en la robótica | 28 |
| 3 Materiales y métodos | 31 |
| 3.1 Aplicaciones software | 31 |
| 3.1.1 Motor gráfico Unity | 31 |

ÍNDICE GENERAL

| | | |
|----------|--|-----------|
| 3.1.2 | RobotStudio | 33 |
| 3.1.3 | <i>Robot Operating System (ROS)</i> | 34 |
| 3.2 | Métodos | 35 |
| 3.2.1 | Cinemática Directa | 35 |
| 3.2.2 | Cinemática Inversa | 36 |
| 3.2.3 | Método IK Solver | 37 |
| 3.3 | Protocolo de comunicación | 38 |
| 3.4 | Materiales | 40 |
| 3.4.1 | Oculus Rift S | 40 |
| 3.4.2 | IRB 140 | 42 |
| 3.4.3 | IRB 1090 | 43 |
| 3.4.4 | Schunk LWA 4P | 45 |
| 4 | Desarrollo de la aplicación | 49 |
| 4.1 | Introducción | 49 |
| 4.2 | Selección motor gráfico | 50 |
| 4.3 | Selección protocolo de comunicación | 52 |
| 4.4 | Entorno virtual | 54 |
| 4.4.1 | Primera estación | 54 |
| 4.4.2 | Segunda estación | 55 |
| 4.4.3 | Tercera estación | 56 |
| 4.4.4 | Cuarta estación | 57 |
| 4.5 | Capacidades del avatar virtual | 59 |
| 4.6 | Panel principal | 62 |
| 4.7 | Panel movimiento del robot | 64 |
| 4.7.1 | Movimiento manual | 66 |
| 4.7.2 | Movimiento individual | 66 |
| 4.7.3 | Movimiento por ejes | 67 |
| 4.7.4 | Movimiento Home | 68 |
| 4.8 | Panel de programación/simulación | 68 |
| 4.8.1 | Panel Aprendizaje | 69 |
| 4.8.2 | Panel Programación | 71 |
| 4.9 | Panel de comunicación | 73 |
| 4.10 | Comunicación Unity | 76 |
| 4.11 | Comunicación RobotStudio y Conexión del robot real | 78 |
| 4.12 | Conexión ROS | 81 |
| 5 | Discusión | 85 |
| 5.1 | Análisis | 85 |
| 5.1.1 | Entorno Virtual | 85 |
| 5.1.2 | Robots IRB140 e IRB1090 simulados | 86 |
| 5.1.3 | Entorno SCHUNK LWA real | 87 |
| 5.2 | Limitaciones del proyecto | 88 |
| 6 | Conclusiones y trabajos futuros | 89 |
| 6.1 | Conclusión | 89 |
| 6.2 | Trabajos Futuros | 90 |

ÍNDICE GENERAL

| | |
|---------------|-----------|
| Anexos | 91 |
|---------------|-----------|

ÍNDICE GENERAL

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mi director José Carlos Moreno Úbeda y a mi codirector Fernando Cañadas Aránega por su invaluable ayuda en la superación de las dificultades que enfrenté al escribir y desarrollar este trabajo. Su eterna paciencia e implicación constante fueron fundamentales para orientarme y motivarme, aspectos cruciales sin los cuales la realización de este TFG no habría sido posible.

Agradezco profundamente a mis padres y abuelos por su confianza inquebrantable en mí y por impulsarme siempre a perseguir mis sueños con determinación y pasión. A mi padre, en particular, le agradezco por haberme inculcado desde niño esta pasión que ha guiado mi camino hasta aquí. Su apoyo incondicional y sus sabios consejos han sido el motor que me ha llevado hasta aquí. A mi hermano, le agradezco especialmente por su constante apoyo y los conocimientos que generosamente compartió conmigo, facilitándome el camino hacia la culminación de este proyecto académico. A mis amigos, quienes han sido pilares fundamentales a lo largo de esta travesía académica, les agradezco de corazón por su presencia constante y su apoyo incondicional en los momentos de alegría y dificultad. Su amistad ha sido una fuente inagotable de ánimo y motivación.

A mi novia y a C, quiero dedicarles un agradecimiento especial. Su apoyo constante, comprensión y amor han sido mi roca durante los momentos de mayor desafío y desesperación que este proyecto me ha presentado. Su ayuda incondicional y la pasión que compartimos por esta profesión han sido la principal fuente de inspiración y motivación que me han permitido culminar este periodo estudiantil con el desarrollo de este proyecto tan desafiante para mí.

Resumen

En la actualidad, la evolución de la robótica juega un papel fundamental en el ámbito del desarrollo industrial y social. Ante la creciente necesidad de automatización y robotización de procesos industriales y actividades cotidianas, surgen nuevas demandas en cuanto a la interacción con los usuarios. Por ello, la integración de nuevas tecnologías en el ámbito de la robótica es crucial para ofrecer nuevas oportunidades. Este Trabajo de Fin de Grado (TFG) explora una novedosa forma de interacción humano-robot basada en la tecnología de realidad virtual, la cual se encuentra en un momento álgido de su desarrollo. El objetivo se centra en el desarrollo de un entorno virtual realista en Unity donde se emula el comportamiento de tres brazos robóticos (IRB 140, IRB 1090 y SCHUNK LWA 4P), accesible mediante la tecnología de las gafas de realidad virtual "Oculus". Este entorno permite interactuar con robots mediante movimientos manuales y por ejes, ofreciendo diferentes opciones de programación como guardado de puntos, simulación de E/S digitales y creación de programas básicos. La comunicación se realiza mediante el protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*), conectando el entorno con RobotStudio para los robots de IRB y ROS (*Robot Operating System*) para el Schunk, llegando a conectar este último con el modelo real. En el presente proyecto, se busca explorar las posibilidades emergentes de la realidad virtual en el campo de la robótica y contribuir al avance de la interacción humano-robot.

Palabras clave: *Brazos robóticos, Unity, RobotStudio, ROS.*

Abstract

Today, the evolution of robotics plays a key role in industrial and social development. With the growing need for automation and robotization of industrial processes and everyday activities, new demands arise in terms of interaction with users. Therefore, the integration of new technologies in the field of robotics is crucial to offer new opportunities. This Final Degree Project (TFG) explores a novel form of human-robot interaction based on virtual reality technology, which is at the peak of its development. The objective focuses on developing a realistic virtual environment in Unity where the behavior of some robotic arms (IRB 140, IRB 1090, and SCHUNK LWA 4P) is emulated, and accessible through the technology of the virtual reality glasses "Oculus". This environment allows interaction with robots through manual and axis movements, offering different programming options such as saving points, simulation of digital I/O, and creation of basic programs. Communication is via TCP/IP (Transmission Control Protocol/Internet Protocol), connecting the environment with RobotStudio for the ABB robots and ROS (Robot Operating System) for the Schunk, and even connect the latter to the real model. In the present project, the aim is to explore the emerging possibilities of virtual reality in the field of robotics and contribute to advancing human-robot interaction.

Keywords: *Robotic arms, Unity, RobotStudio, ROS.*

Siglas y acrónimos

| | |
|---------------|--|
| ABB | <i>Asea Brown Boveri</i> |
| CCD | <i>Cyclic Coordinate Descent</i> |
| CPU | <i>Central Processing Unit</i> |
| IDE | <i>Integrated Development Environment</i> |
| MCD | Método Cinemático Directo |
| MCI | Método Cinemático Indirecto |
| PUMA | <i>Programmable Universal Machine for Assembly</i> |
| RA | Realidad Aumentada |
| RAM | <i>Random Access Memory</i> |
| RM | Realidad Mixta |
| ROS | <i>Robot Operating System</i> |
| RV | Realidad Virtual |
| SCARA | <i>Selective Compliance Assembly Robot Arm</i> |
| TCP/IP | <i>Transmission Control Protocol/Internet Protocol</i> |
| TFG | Trabajo Fin de Grado |
| UAL | Universidad de Almería |
| UDP | <i>User Datagram Protocol</i> |
| ARM | Automática, Robótica y Mecatrónica |

Índice de figuras

| | | |
|------|---|----|
| 1.1 | "Unimate", primer robot industrial. Extraído de [1] | 4 |
| 1.2 | Sensorama. Extraído de [2] | 5 |
| 1.3 | Virtual Boy de Nintendo. Extraído de [3] | 5 |
| 1.4 | HTC Vive. Extraído de [4] | 8 |
| 1.5 | Meta Quest. Extraído de [5] | 8 |
| 1.6 | Apple Vision Pro. Extraído de [6] | 8 |
| 2.1 | IRB 6. Extraído de [7] | 16 |
| 2.2 | Instalación anual de robots industriales (2012-2022). Extraído de [8] | 16 |
| 2.3 | Robot móvil Summit. Extraído de [9] | 18 |
| 2.4 | Robot zoomórfico con estructura inspirada en un gato. Extraído de [10] | 19 |
| 2.5 | Robot Articulado. Extraído de [11] | 19 |
| 2.6 | Humanoide. Extraído de [12] | 19 |
| 2.7 | Cobot. Extraído de [13] | 20 |
| 2.8 | Robot híbrido. Extraído de [14] | 20 |
| 2.9 | Partes de un brazo robótico. Extraído de [15] | 20 |
| 2.10 | Esquema de control de un robot. Extraído de [16] | 21 |
| 2.11 | <i>Teach Pendant</i> de ABB. Extraído de [17] | 22 |
| 2.12 | Guantes Hapticos. Extraído de [18] | 25 |
| 2.13 | <i>VR Walking Platform</i> . Extraído de [19] | 25 |
| 2.14 | Simulación operación. Extraído de [20] | 26 |
| 2.15 | Aprendizaje partes corazón con VR. Extraído de [21] | 26 |
| 2.16 | Pokémon Go. Extraído de [22] | 27 |
| 2.17 | Renacimiento ruinas romanas, Teatro Romano de Cartagena. Extraído de [23] | 28 |
| 2.18 | Prototipo BMW. Extraído de [24] | 29 |
| 3.1 | RobotStudio. Extraído de [25] | 34 |
| 3.2 | ROS. Extraído de [26] | 35 |
| 3.3 | Método Ccd IK Solve. Extraído de [27] | 38 |
| 3.4 | Oculus Rift S. Extraído de [28] | 41 |
| 3.5 | IRB 140. Extraído de [29] | 43 |
| 3.6 | Espacio de trabajo de IRB 140. Extraído de [30] | 43 |

ÍNDICE DE FIGURAS

| | | |
|------|---|----|
| 3.7 | IRB 1090. Extraído de [31] | 45 |
| 3.8 | Espacio de trabajo de IRB 1090. Extraído de [32] | 45 |
| 3.9 | SCHUNK LWA 4P. Extraído de [33] | 47 |
| 4.1 | Unreal Engine. Extraído de [34] | 51 |
| 4.2 | Epic Games. Extraído de [35] | 51 |
| 4.3 | Unity. Extraído de [36] | 51 |
| 4.4 | Modelo Cliente-Servidor. Extraído de [37] | 53 |
| 4.5 | Vista general de la escena | 54 |
| 4.6 | Vista de la primera estación (Parte trasera del atril) | 55 |
| 4.7 | Vista de la primera estación (Parte delantera del atril con logo de la UAL) | 55 |
| 4.8 | Vista general, segunda estación | 56 |
| 4.9 | Vista IRB1090 | 56 |
| 4.10 | Vista general, tercera estación | 57 |
| 4.11 | Vista IRB140 | 57 |
| 4.12 | Vista general, cuarta estación | 58 |
| 4.13 | Vista SCHUNK LWA 4P | 58 |
| 4.14 | Representación de las manos humanas relajadas | 59 |
| 4.15 | Representación de las manos humanas accionadas | 59 |
| 4.16 | Función de cada botón del mando | 60 |
| 4.17 | Proyección de un rayo con disco de teletransporte | 60 |
| 4.18 | Mecanismo de agarre pantallas | 61 |
| 4.19 | Mecanismo de agarre al robot | 61 |
| 4.20 | Interacción con la pantalla | 62 |
| 4.21 | Panel principal | 63 |
| 4.22 | Entorno iluminado, para IRB 1090 | 63 |
| 4.23 | Entorno iluminado, para IRB 140 | 64 |
| 4.24 | Entorno iluminado, para SCHUNK LWA 4P | 64 |
| 4.25 | Tutorial de controladores junto con funcionalidades de botones | 65 |
| 4.26 | Panel del movimiento del robot | 65 |
| 4.27 | Robot en movimiento manual | 66 |
| 4.28 | Robot en movimiento individual | 67 |
| 4.29 | Movimiento por ejes | 67 |
| 4.30 | Panel escoger acción | 68 |
| 4.31 | Editor de puntos | 69 |
| 4.32 | Teclado del editor de puntos | 70 |
| 4.33 | Editor de trayectorias | 71 |
| 4.34 | Editor de trayectorias | 71 |
| 4.35 | Ejemplo Programa | 73 |
| 4.36 | Gestor entradas/salidas | 73 |
| 4.37 | Conexión en directo | 74 |

| | |
|---|----|
| 4.38 Simulación de puntos | 74 |
| 4.39 Simulación trayectorias | 75 |
| 4.40 Simulación programas | 75 |
| 4.41 Comunicación Unity-Robotstudio | 80 |
| 4.42 Comunicación Unity-ROS | 83 |

Índice de tablas

| | | |
|-----|--|----|
| 1.1 | Distribución temporal de las fases de desarrollo del TFG | 11 |
| 4.1 | Código 1: Ejemplo de comunicación en C# | 77 |
| 4.2 | Código 2: Ejemplo de comunicación en Rapid | 79 |
| 4.3 | Código 3: Ejemplo de comunicación en Python | 82 |

**PROGRAMACIÓN DE ROBOTS
MANIPULADORES MEDIANTE
REALIDAD VIRTUAL**

Capítulo 1

Introducción

1.1. Introducción e interés

La robótica, según la Real Academia Española, se define como una técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales [38]. En la actualidad, el término robótica engloba mucho más, ya que no sustituye al humano en ciertas tareas, sino que los robots se han convertido en un complemento tanto para los operarios industriales como para los usuarios de la vida cotidiana.

Mediante el desarrollo de la robótica cooperativa, enfocada principalmente al uso industrial, y la robótica social, cuyo principal objetivo es contribuir al bienestar de las personas, el uso de robots se encuentra mucho más generalizado e integrado en la sociedad. Es importante aclarar que el término robot no solo está referido únicamente a brazos robóticos, sino a todos aquellos dispositivos que disponen de distintos grados de libertad y que, mediante una correcta programación, son capaces de desempeñar cualquier acción impuesta como objetivo. Sin embargo, los robots deben ser programados bajo las Leyes de la robótica o Leyes de Asimov que establecen los principios éticos en los cuales se debe basar [39]. Estas leyes son:

- **Primera ley:** Un robot no hará daño a un ser humano, ni por inacción permitirá que un ser humano sufra daño.
- **Segunda ley:** Un robot debe cumplir las órdenes dadas por los seres humanos, a excepción de aquellas que entre en conflicto con la primera ley.
- **Tercera ley:** Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la primera o con la segunda ley.

La implementación del primer robot industrial fue el denominado Unimate, el cual se puede observar en la Figura 1.1. Tuvo lugar en la planta de General Motors en 1956 y marcó un

1.1. INTRODUCCIÓN E INTERÉS

hito en la automatización industrial, aumentando significativamente la eficiencia, precisión y seguridad. Esto permitió a las empresas reducir costes y aumentar la calidad de sus productos, además, como valor añadido, permitió generar nuevos empleos dedicados al mantenimiento y desarrollo de esta tecnología. Además, contribuyó al desarrollo tecnológico de otros campos, ya que al liberar a trabajadores de líneas peligrosas y trabajos repetitivos, permitió que las nuevas generaciones invirtieran sus recursos en generar nuevos empleos más cualificados y creativos. De esta forma, la robótica supuso un gran avance tecnológico, no solo en el ámbito industrial, sino que permitió su extrapolación a otras áreas que se vieron impulsadas por esta tecnología, como la informática, la cual juega un papel fundamental en el ámbito de la industria, la automatización y la robotización [40].



Figura 1.1. "Unimate", primer robot industrial. Extraído de [1]

En consecuencia, la informática experimentó un gran avance gracias a la robotización de las industrias ya que la incorporación de robots generó nuevas necesidades informáticas, impulsando la investigación y desarrollo de un *hardware* más potente y especializado, imposible de diseñar de manera artesanal. Paralelamente, también se ha favorecido el desarrollo de *software* avanzados, aplicados tanto en comunicación, programación e incluso en los propios robots y generando nuevas necesidades tanto en programación, como en la interacción humano-máquina. De esta forma, puede entenderse el inicio de la robotización y automatización industrial, como el comienzo de un ciclo continuo, donde los avances industriales conllevan un desarrollo informático y a la inversa.

En este contexto de continuo desarrollo, y ante la creciente necesidad de acceder al conocimiento y experimentar estímulos de manera personal e inmediata, surge la Realidad Virtual (RV), la cual no solo brinda la disponibilidad instantánea de estos conocimientos y experiencias, sino que también permite la capacidad de crear y sumergirse en mundos alternativos que, hasta ahora, solo estaban disponibles a través de una pantalla. La RV ofrece una capacidad de inmersión y posibilidad de interacción con los entornos simulados, cuyo desarrollo está en constante

evolución. Asimismo, la forma en la que el humano interactúa con esta tecnología y el mundo que le rodea, le permite sumergirse en experiencias sensoriales que van más allá de los límites de la realidad física [41].

Esta tecnología tuvo sus inicios en la década de los 60 con dispositivos como Sensorama (Figura 1.2) donde nacieron los primeros cascos de RV. Su desarrollo se acentuó en los años 90 con productos como el Virtual Boy de Nintendo [42] (Figura 1.3). Pero no fue hasta el año 2010 cuándo los avances en la tecnología en pantallas, sensores y capacidades de procesamiento gráfico hicieron posible que compañías como Oculus Rift y HTC Vive revolucionaran la industria, haciendo la RV accesible y más inmersiva [43, 44].



Figura 1.2. Sensorama. Extraído de [2]



Figura 1.3. Virtual Boy de Nintendo. Extraído de [3]

Las amplias capacidades en cuanto a la accesibilidad e inmersión hizo posible implementar la RV en campos muy diversos, desde videojuegos y educación hasta medicina y entrenamiento profesional. Incluyendo entre estos, el de la robótica y la industria, ya que al optimizar la interacción humano máquina no solo se mejoran las posibilidades de control, sino que aparecen nuevas posibilidades en cuanto al análisis de mecanismos, simulación de sistemas, optimización de recursos y análisis de seguridad.

1.2. MOTIVACIÓN DEL TFG

1.2. Motivación del TFG

La elección de este Trabajo de Fin de Grado (TFG) surge del interés por la robótica y la fascinación que esta disciplina ha despertado desde una edad temprana. La posibilidad de observar de primera mano cómo las evoluciones tecnológicas, la automatización de tareas y la robotización han contribuido al desarrollo tanto industrial como social, establece los cimientos sobre el deseo de contribuir activamente a este desarrollo tecnológico.

En este contexto, la Universidad de Almería (UAL) brinda la oportunidad de explorar esta tecnología, permitiendo que la robótica, automatización y electrónica sean accesibles a todo aquel que comparta el mismo objetivo. No solo con el grado en Ingeniería Electrónica Industrial y Automática, sino con numerosas oportunidades que se ofrecen a los estudiantes, como la posibilidad de estudiar una doble titulación de Automática Industrial en colaboración con la Universidad de Brescia (Italia), numerosos talleres de aprendizaje y competiciones que permiten acercar la robótica a todos los públicos.

De esta forma, surge la posibilidad de realizar este TFG sobre la Programación de brazos manipuladores mediante realidad virtual, el cual despierta un gran interés, ya que no solo trabaja de forma activa con la robótica y sus aplicaciones, sino que incluye una de las tecnologías más novedosas y desarrolladas en los últimos años, como es la RV.

Esta tecnología ha supuesto un punto de inflexión en la ingeniería, al facilitar la interacción directa con máquinas y componentes, mejorar la comprensión práctica de ensayos y simulaciones, y permitir la representación realista de ideas o prototipos sin necesidad de ejecutarlos. Esta convergencia tecnológica podría impulsar una nueva forma de programación inmersiva, accesible tanto para ingenieros como para no especialistas en robótica. Además, ofrece ventajas como la realización de simulaciones seguras e interactivas, así como la programación remota en tiempo real de robots. Por todo ello, este TFG presenta la oportunidad de explorar los límites de la robótica e introducirse en el mundo de la RV, aplicando los conocimientos adquiridos durante el grado.

1.3. Objetivos

Este TFG tiene como principal finalidad desarrollar una aplicación inmersiva dotada de modelos representativos de los brazos robóticos IRB 140, IRB 1090 y SCHUNK LWA 4P. Debe ofrecer una experiencia virtual agradable para al usuario donde se desea incluir distintos paneles de control, permitiendo escoger diferentes métodos de cálculo de trayectorias en brazos robóticos y la posibilidad de programar puntos, trayectorias, simulación de entradas/salidas digitales y elaborar programas básicos.

Además, se utilizará el protocolo de comunicación TCP/IP (*Transmission Control Protocol*)

l/Internet Protocol) brindando la oportunidad de transmitir información desde la aplicación de RV hacia las otras aplicaciones de control de los distintos robots, Robotstudio para el caso de los ABB (*Asea Brown Boveri*) y ROS (*Robot Operating System*) para el Schunk. Para facilitar la elaboración de este TFG podemos descomponerlo en objetivos parciales que se irán abordando durante las distintas fases de desarrollo. Dichos objetivos parciales son:

- **Objetivo 1:** Desarrollar un entorno de RV, en el motor gráfico de Unity, configurado correctamente para trabajar con la tecnología de Oculus Rift S.
- **Objetivo 2:** Incluir en este entorno modelos representativos de cada uno de los robots y dotar a cada uno de ellos del tipo de movimiento que los caracteriza.
- **Objetivo 3:** Crear un sistema de paneles de control con el que navegar entre los distintos métodos de control.
- **Objetivo 4:** Crear un sistema de almacenamiento y creación de puntos, trayectorias y un editor básico de programas.
- **Objetivo 5:** Establecer un canal de comunicación entre Unity y cada una de las aplicaciones encargadas de controlar los robots, el cual deberá permitir una transmisión bidireccional de información en tiempo real.
- **Objetivo 6:** Configurar cada una de las aplicaciones receptoras para gestionar la información procedente de Unity y transmitirla al controlador del robot de forma eficiente.
- **Objetivo 7:** Transferir los programas creados a los distintos robots y solucionar los posibles problemas que puedan surgir de la aplicación práctica.

Una vez se hayan completado todos los objetivos parciales satisfactoriamente, el TFG concluirá con la elaboración de esta memoria y se dispondrá de una aplicación donde, mediante la RV, se permitirá programar y controlar los distintos robots. Los códigos empleados en el desarrollo de este TFG se encuentran disponibles en el repositorio del grupo ARM en GitHub [45].

1.4. Contexto

En los últimos años, la RV ha experimentado un desarrollo significativo, impulsado por los avances en pantallas y procesadores gráficos. Esta evolución ha convertido lo que antes era una idea futurista en una realidad accesible para el público general. Empresas de renombre, como Meta y Apple han lanzado productos al mercado en este campo, facilitando su adopción masiva. La disponibilidad de dispositivos como HTC Vive, Meta Quest y Apple Vision las cuales se pueden observar en las Figuras 1.4, 1.5 y 1.6) respectivamente, han democratizado el acceso a

1.4. CONTEXTO

experiencias inmersivas, permitiendo que cada vez más personas experimenten e introduzcan la RV en sus actividades cotidianas.



Figura 1.4. HTC Vive. Extraído de [4]



Figura 1.5. Meta Quest. Extraído de [5]



Figura 1.6. Apple Vision Pro. Extraído de [6]

Dicha democratización de las experiencias inmersivas ha permitido que los usuarios experimenten y desarrollen aplicaciones muy diversas que abarcan desde el entretenimiento, pasando por la salud y la educación, hasta la aplicación industrial. Esta integración de RV en la industria es un proyecto que lleva años consolidado y en vías de desarrollo. Compañías como Siemens han implementado esta tecnología para simular y optimizar procesos de fabricación, permitiendo a los ingenieros ver y manipular modelos 3D de líneas de producción. NVIDIA, por su parte, posee su propia plataforma de simulación de robots en un entorno virtual para crear simulaciones detalladas y realistas. Además, la RV ha supuesto un punto de inflexión en el diseño de maquinaria, ya que ofrece la posibilidad de simular e interactuar de forma directa con el producto diseñado, sin la necesidad de fabricación de prototipos, lo que reduce notablemente los costes y tiempos de producción.

Estos ejemplos de aplicaciones de la RV a nivel industrial también han sido extrapolados al ámbito de la robótica. Fabricantes como ABB han incluido en su propio *software* (RobotStudio) la tecnología de RV, que permite visualizar las simulaciones y programas ejecutados desde un punto más inmersivo. En la robótica educativa, esta tecnología ha tenido una gran relevancia, ya que da la posibilidad a los estudiantes de visualizar un modelo realista de cualquier tipo de robot o mecanismo para poder realizar simulaciones con ellos.

De esta forma, la convergencia de la RV con la robótica y la automatización industrial no solo está transformando estos campos, sino que también genera nuevas oportunidades para la innovación y el aprendizaje. Este TFG permite explorar y desarrollar nuevas formas de interacción con los robots, siendo una contribución al avance tecnológico y establece un nuevo rumbo para futuros desarrollos en este ámbito.

1.5. Resumen de resultados

Este trabajo ha dado como resultado el desarrollo de una aplicación de RV innovadora, destacando por la integración de modelos realistas de brazos robóticos dotados de diversos tipos de movimientos. Además, se ha creado una interfaz intuitiva que permite almacenar y probar puntos, trayectorias y programas en un entorno de simulación seguro y controlado utilizando Unity. Para optimizar la interacción con los usuarios, se ha diseñado un entorno virtual realista similar a un laboratorio de robótica, donde los objetos presentan propiedades físicas análogas a las reales.

En relación con las pruebas realizadas con los robots, se ha logrado un alto rendimiento en la simulación del robot IRB 1090 [46] utilizando RobotStudio, lo que demuestra la efectividad de los algoritmos de control y los protocolos de comunicación desarrollados. Aunque no se llevaron a cabo pruebas físicas debido a la falta del robot en el laboratorio, los resultados de la simulación son prometedores para su futura implementación práctica. Asimismo, la simulación del IRB 140 [47] también ha sido satisfactoria, aunque estuvo limitada por la ausencia del paquete *PC Interface* en el controlador físico. Para estas simulaciones, se han realizado dos vídeos, pudiendo comprobarse el correcto funcionamiento del programa en [48] para el IRB 1090 y en [49] para el IRB 140.

Sin embargo y como principal resultado, para el robot Schunk LWA 4P si que se ha podido lograr establecer una conexión efectiva entre el entorno virtual y el robot físico a través de Unity y ROS Melodic. Los puntos, trayectorias y programas fueron ejecutados con precisión conforme a lo programado en la aplicación de RV. Sin embargo, se enfrentaron dificultades debido a los tiempos de retardo establecidos por la transmisión en directo, lo que obligó a aumentar la frecuencia de transmisión de datos a 0.4 segundos. Esto resultó en una ligera disminución de la precisión de los movimientos. Aun así, el resultado superó las expectativas del proyecto, pudiéndose observar el correcto funcionamiento en el vídeo [50].

Para poder establecer la conexión, los códigos diseñados se encuentran en el repositorio del grupo de investigación ARM en GitHub [45], acompañado de una guía de uso y instalación.

1.6. Fases de desarrollo y planificación

Para abordar este TFG, se ha dividido en distintas fases de desarrollo, las cuales se muestran a continuación:

- **Fase 1: Revisión del estado del arte.** Se llevará a cabo un análisis de los enfoques adoptados en proyectos similares por otras entidades.
- **Fase 2: Exploración de motores gráficos y estudio de los distintos robots utiliza-**

1.6. FASES DE DESARROLLO Y PLANIFICACIÓN

dos. Se estudiarán diversas plataformas de desarrollo gráfico en RV, seleccionando aquella que sea compatible con tecnología Oculus y un entorno de programación adecuado. Se analizarán los brazos robóticos utilizados, evaluando las capacidades de sus controladores y obteniendo modelos representativos.

- **Fase 3: Desarrollo de la aplicación de RV.** Se creará una escena virtual optimizada para dispositivos Oculus, asegurando una configuración adecuada para movilidad e interacción con objetos.
- **Fase 4: Implementación de sistemas de control para brazos robóticos.** Se aplicarán restricciones de movimiento y se implementará un sistema de cinemática inversa para calcular los ángulos de articulación.
- **Fase 5: Configuración de un panel de control.** Se integrarán métodos para gestionar el movimiento del robot, incluyendo desplazamiento manual, control por eslabón individual y manejo mediante *joystick*.
- **Fase 6: Sistema de almacenamiento y simulación.** Se crearán sistemas para almacenar puntos, trayectorias y programas, además de implementar un entorno de simulación en la escena virtual.
- **Fase 7: Protocolo de comunicación.** Se desarrollará un protocolo para la transmisión de información sobre el movimiento del robot simulado, a la aplicación que controla el robot real, en tiempo real.
- **Fase 8: Conexión y ensayos.** Se establecerá la conexión necesaria para permitir el movimiento del robot físico mediante el controlador virtual. Se realizarán ensayos en el robot real para corregir fallos, ajustar tiempos de transmisión y optimizar la velocidad de ejecución.
- **Fase 9: Escritura de la memoria.** Se ha escrito el documento en Latex, en el que se detallan todos los aspectos relativos a este TFG, así como la justificación del tema, resultados obtenidos y el desarrollo del trabajo. Para finalizar, se encuentran las conclusiones en las que se valora el trabajo realizado.

En la Tabla 1.1, se recoge la distribución temporal de las distintas fases:

| Mes | Semana | Fase de desarrollo | | | | | | | | | Total semana | Total mes |
|---------------------|--------|--------------------|----|----|----|----|----|----|----|----|--------------|-----------|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | |
| Febrero | 01-08 | 5 | 6 | 10 | | | | | | | 21 | 71 |
| | 09-16 | 3 | 8 | 5 | | | | | | | 16 | |
| | 17-24 | | 4 | 5 | 7 | | | | | | 16 | |
| | 25-03 | | | 10 | 8 | | | | | | 18 | |
| Marzo | 04-11 | | | 15 | 12 | | | | | | 27 | 72 |
| | 12-19 | | | 6 | 5 | 7 | | | | | 18 | |
| | 20-27 | | | 2 | 3 | 6 | 8 | | | | 19 | |
| | 28-04 | | | 2 | 3 | 3 | | | | | 8 | |
| Abril | 05-12 | | | 4 | 5 | 8 | | | | | 17 | 63 |
| | 13-20 | | | | 2 | 5 | 6 | | | | 13 | |
| | 21-28 | | | | | 3 | 6 | 8 | | | 17 | |
| | 29-06 | | | | | | 4 | 7 | 10 | | 21 | |
| Mayo | 07-14 | | | | 6 | 8 | | | | | 14 | 37 |
| | 15-22 | | | | | 2 | 4 | 5 | | | 11 | |
| | 23-30 | | 3 | 9 | | | | | | | 12 | |
| Junio | 31-07 | | | | 3 | | | 8 | | 10 | 21 | 110 |
| | 08-15 | | | | | | | 15 | 10 | | 25 | |
| | 16-23 | | | | | | | | 10 | 19 | 29 | |
| | 24-01 | | | | | | | | | 35 | 35 | |
| Horas por actividad | | 8 | 18 | 53 | 57 | 54 | 28 | 43 | 30 | 64 | Total | 353 |

Tabla 1.1. Distribución temporal de las fases de desarrollo del TFG

1.7. Competencias empleadas en este trabajo

El Grado en Ingeniería Electrónica Industrial y Automática, destaca por ser multidisciplinar en todos los contenidos que se imparten, de esta manera, se les otorga a los estudiantes la formación necesaria para que en el futuro puedan resolver todos los problemas industriales de diversos ámbitos del conocimiento [51].

1.7.1. Competencias básicas, generales y transversales

Se definen competencias de carácter básico, aquellas que deben adquirirse en cualquier grado, según se estipula en el Real Decreto 1393/2007.

- **Poseer y comprender conocimientos (CB1):** Se ha demostrado poseer y comprender conocimientos en diversas áreas de estudio para poder realizar correctamente este TFG.
- **Aplicación de los conocimientos (CB2):** Se aplican los conocimientos adquiridos en el grado en la elaboración y defensa de este TFG.
- **Capacidad de emitir juicios (CB3):** Poseer la capacidad de reunir e interpretar datos para emitir juicios y reflexiones sobre temas necesarios para elaborar este TFG.

1.7. COMPETENCIAS EMPLEADAS EN ESTE TRABAJO

- **Capacidad de comunicar y aptitud social (CB4):** Se evaluará en la exposición ante un tribunal.
- **Habilidad para el aprendizaje (CB5):** Medios de aprendizaje adquiridos durante el grado y los cuales se plasman en este TFG.

Por otro lado, encontramos las competencias transversales, las cuales han sido aprobadas por el Consejo de Gobierno de la UAL, y son comunes a todos los grados de la misma.

- **Conocimientos básicos de la profesión (UAL1):** Conocimiento, habilidades y aptitudes que posibilitan la comprensión de nuevas teorías, interpretaciones, métodos y técnicas dentro del ámbito de la ingeniería, necesarios para poder realizar este TFG.
- **Habilidad en el uso de las TIC (UAL2):** Realizar este TFG basado en la programación de una aplicación de RV y establecer protocolos de comunicación entre distintas aplicaciones como Unity, ROS y Robotstudio.
- **Capacidad para resolver problemas (UAL3):** Esta competencia se ha utilizado constantemente, ya que este TFG es un trabajo ingenieril en el que se han ido solventando distintos problemas que han ido surgiendo.
- **Comunicación oral y escrita en la propia lengua (UAL4):** Redactando esta memoria y su futura exposición se plasmará dicha competencia
- **Capacidad de crítica y autocritica (UAL5):** A la hora de escoger la información correcta que se necesita para abordar los distintos problemas que han ido surgiendo.
- **Conocimiento de una segunda lengua (UAL7):** La mayoría de las fuentes bibliográficas se encuentran en inglés. Además, al estar matriculado en el Doble título Internacional “Mecatrónica para la automatización industrial”, este TFG se traducirá al italiano y posteriormente se expondrá en dicha lengua.
- **Compromiso ético (UAL8):** Este TFG respeta los distintos principios de carácter universal.
- **Capacidad para aprender a trabajar de forma autónoma (UAL9):** En este TFG se ha contado con la colaboración del tutor y cotutor, pero el principal desarrollo se realiza de manera individual.

1.7.2. Competencias específicas del título

Además de las competencias definidas anteriormente, existen unas competencias exclusivas del grado en Ingeniería Electrónica Industrial y Automática [52]:

- **Conocimiento en materias básicas y tecnológicas, que les capacite el aprendizaje de nuevos métodos y teorías, y les dote de versatilidad para adaptarse a nuevas situaciones (CT3):** Además de aplicar conocimientos de robótica ya adquiridos se han abordado nuevos desafíos intelectuales al trabajar con la realidad virtual.
- **Capacidad de resolver problemas con iniciativas, toma de decisiones, creatividad, razonamiento crítico y de comunicar y transmitir conocimientos, habilidades y destrezas en el campo de la Ingeniería Industrial (CT4):** Se ha explicado anteriormente en la UAL3 y UAL5; ya que a lo largo del grado en ingeniería se han tenido que resolver problemas usando la creatividad, lo cual es la principal característica de este TFG.
- **Capacidad para aplicar los principios y métodos de la calidad (CT8):** Al llevar a cabo este trabajo se han empleado técnicas y software avanzados, logrando un producto de calidad.
- **Capacidad de trabajar en un entorno plurilingüe y multidisciplinar (CT10):** Se ha explicado anteriormente en la competencia UAL4 y UAL7, ya que la mayoría de los documentos utilizados para la realización de este TFG están escritos en inglés y posteriormente se va a traducir y realizar una exposición en italiano.
- **Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería (CB3):** Este TFG presenta un desafío en cuanto a programación, ya que se trabaja con distintos lenguajes de programación entre los cuales existe un flujo continuo de información.
- **Capacidad de visión espacial y conocimiento de las técnicas de representación gráfica, tanto por métodos tradicionales de geometría métrica y geometría descriptiva, como mediante las aplicaciones de diseño asistido por ordenador (CB5):** En el desarrollo de un entorno virtual se han abordado numerosos desafíos en cuanto a la creación de modelo 3D de robots, objetos y el renderizado de estos.
- **Conocimientos de principios y aplicaciones de los sistemas robotizados (CTEE9):** El pilar fundamental de este TFG es la robótica, por lo que conocer los modelos robóticos es un hecho imprescindible.
- **Conocimiento aplicado de informática industrial y comunicaciones (CTEE10):** Se han desarrollado distintos canales de comunicación basados en el protocolo TCP/IP y el uso de sockets.
- **Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería Industrial de naturaleza profesional en el que se**

1.8. ESTRUCTURA DEL TFG

sinteticen e integren las competencias adquiridas en las enseñanzas (TFG): El desarrollo de este TFG constituye una actividad puramente creativa, permitiendo resolver los problemas planteados con originalidad. Todo ello genera un proyecto que se expondrá frente a un tribunal evaluador.

1.8. Estructura del TFG

Este documento presenta una estructura de seis capítulos, con índices de contenidos, figuras, tablas y sus respectivos anexos.

El capítulo 1 consta de una introducción, cuya función es proporcionar una primera toma de contacto con los temas que se van a abordar en este TFG. Incluye la motivación por la cual se ha llevado a cabo este trabajo, detalla los primeros resultados obtenidos, incorpora la planificación temporal que se ha seguido y presenta el listado de competencias empleadas durante el desarrollo del TFG.

El capítulo 2, titulado "Revisión bibliográfica" permite contextualizar el desarrollo de este TFG abordando temas como el papel de la robótica en la actualidad, los distintos tipos de robots y formas de programación existentes. Junto con los conceptos básicos sobre la realidad virtual y sus aplicaciones.

El capítulo 3, denominado "Materiales y métodos", expone los distintos recursos empleados en este TFG, desde las aplicaciones utilizadas, como son Unity, RobotStudio y ROS, hasta los métodos empleados para resolver los distintos problemas presentados, métodos cinemáticos y protocolos de comunicación. Además, incluyen los medios materiales necesarios para la correcta puesta en marcha del proyecto.

El capítulo 4, titulado "Resultados" presenta la aplicación desarrollada en Unity, abordando las funcionalidades de los robots, capacidades de interacción y los protocolos de comunicación entre las distintas aplicaciones.

El capítulo 5, nombrado "Discusión" lleva a cabo un análisis de los distintos ensayos llevados a cabo para cada uno de los robots, valorando los resultados obtenidos y las posibilidades de mejora. Además, incluye demostraciones en vídeo de las pruebas efectuadas.

El capítulo 6, titulado "Conclusiones y trabajos futuros", presenta las conclusiones derivadas del trabajo realizado y las posibles líneas de investigación y desarrollo para trabajos futuros.

Capítulo 2

Revisión bibliográfica

2.1. Introducción

El inicio de la robótica tuvo lugar en los años 70 para sustituir a los humanos en tareas pesadas, peligrosas y monótonas, mediante la realización de tareas básicas de recoger y colocar. Al comprobar el potencial de estas nuevas máquinas, se les añadieron sensores externos, de forma que podían procesar e interpretar situación y así realizar operaciones más complejas. De esta forma, el uso de robots se expandió a tareas como el triturado, el lijado, la soldadura y el ensamblaje. Desde este instante, la robotización ha permitido reducir los costes, aumentar la productividad, mejorar la calidad de los trabajos y ayudar con las tareas peligrosas y dañinas para los humanos [53].

La primera empresa que desarrolló un robot industrial específico en 1967 fue ABB, (empresa creadora de dos de los tres robots utilizados en este TFG), siendo actualmente una de las empresas más punteras en el sector. En esta misma línea de desarrollo, el robot de la Figura 2.1 es el IRB 6, el primer robot industrial totalmente eléctrico y controlado por microprocesadores comercializado, creado en 1974, por ABB [54].

2.1. INTRODUCCIÓN



Figura 2.1. IRB 6. Extraído de [7]

Desde ese momento la robótica ha seguido evolucionando y su uso se ha extendido a distintos sectores, jugando un papel fundamental en la industria. Prueba de ello se puede observar en la Figura 2.2, donde se puede ver un gráfico que corresponde con las instalaciones anuales de robots industriales en Asia/Australia, Europa y Las Américas, entre 2012 y 2022. Asia/Australia destaca por su crecimiento constante y exponencial, alcanzando 405,000 unidades en 2022. Europa y las Américas también han mostrado incrementos, aunque más moderados, con 84,000 y 52,000 unidades respectivamente en 2022. Este aumento refleja una adopción cada vez mayor de la automatización industrial a nivel global. [8].

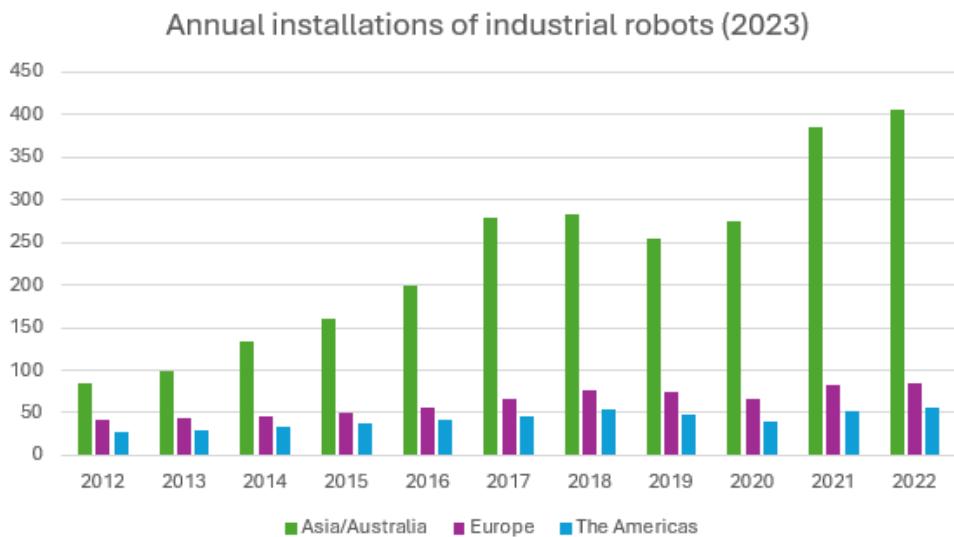


Figura 2.2. Instalación anual de robots industriales (2012-2022). Extraído de [8]

En la actualidad, se espera que se instalen casi dos millones de nuevas unidades de robots industriales en fábricas de todo el mundo, por lo que se observa que la evolución de la robótica y la automatización son vitales para hacer frente a las nuevas tendencias de consumo (demanda de variedad de productos y los retos derivados de las barreras comerciales).

2.2. Robótica industrial en la actualidad.

La robótica juega un papel fundamental en la industria, especialmente en las empresas manufactureras, ya que permite optimizar los recursos para conseguir la máxima eficacia, seguridad y ventaja competitiva en el mercado actual. Puesto que los robots desempeñan tareas repetitivas, minimizan los márgenes de error hasta tasas insignificantes y permiten a los trabajadores humanos centrarse en áreas más productivas y creativas de la empresa. En este escenario, un robot industrial es un sistema autónomo dotado de sensores, controladores y actuadores que ejecuta funciones y operaciones específicas en una línea de fabricación o proceso. Estos robots suelen funcionar de forma continua a través de ciclos de movimiento repetitivos según un conjunto de instrucciones programadas. Aunque, mediante el desarrollo de nuevas tecnologías y la incorporación de la industria 4.0, la complejidad de las tareas realizables por los robots y su capacidad de interpretación de la realidad y adaptación están en continua evolución [55].

Debido al crecimiento y la validación continua de la robótica y la automatización en la industria contemporánea, algunas personas podrían percibir estos avances como una amenaza para los empleos tradicionales. Sin embargo, las ventajas y posibilidades que ofrecen estos sistemas superan el potencial humano en términos de producción. Las principales ventajas de la robótica industrial incluyen [56]:

- **Alta productividad:** gracias a la robótica industrial, se pueden realizar tareas con precisión y de forma repetitiva, sin necesidad de paradas o descansos, esta capacidad de operar de forma continua y sin fatiga hace que los robots sean muy productivos. La precisión de los robots también implica que se cometan menos errores en el proceso de producción, de esta forma, se disminuyen los residuos y los costes de producción.
- **Velocidad y calidad constante:** los robots pueden realizar tareas rutinarias con una calidad y velocidad constante, lo que permite obtener una producción más numerosa y predecible, que garantiza la elaboración de los productos siempre en las mismas condiciones de calidad.

2.2. ROBÓTICA INDUSTRIAL EN LA ACTUALIDAD.

- **Mejora de la seguridad en el trabajo:** al tener un margen de error menor, se obtiene una mayor seguridad en el lugar de trabajo, ya que los robots pueden trabajar en entornos peligrosos y realizar tareas que se consideren de alto riesgo para los humanos. Además, al eliminar el factor humano de la actividad de producción se reducen los riesgos de accidente.
- **Mejor utilización del espacio en la planta:** la utilización de robots precisa un menor espacio e instalaciones que el personal humano, lo que permite a las empresas desarrollar complejas y optimizadas distribuciones en planta.
- **Liberación de personal:** al precisar menor número de operarios para realizar tareas monótonas, estos pueden centrar sus recursos en llevar a cabo actividades más creativas e incorporarse en sectores más productivos de la empresa.

Para abordar de forma adecuada el desarrollo de este TFG, es necesario conocer aquellos aspectos más representativos de la robótica y en especial de los brazos robóticos, ya que estos constituyen el pilar fundamental del proyecto. Para ello es necesario conocer los distintos tipos de robots, y las distintas partes de un brazo robótico. Actualmente, existen diferentes grupos de robots [57, 58]:

- **Robots móviles autónomos:** Se mueven por el entorno y toman decisiones en tiempo real gracias a que incorporan tecnología como cámaras y sensores. En la Figura 2.3, se puede observar el robot móvil Summit.



Figura 2.3. Robot móvil Summit. Extraído de [9]

- **Robots zoomórficos:** Son robots inspirados en animales en cuanto a estructura y movimiento se refieren, se basan en la copia de partes de animales vivos o extintos. En la siguiente Figura 2.4, se puede observar un robot zoomórfico con la estructura inspirada en un gato [8].



Figura 2.4. Robot zoomórfico con estructura inspirada en un gato. Extraído de [10]

- **Robots articulados (brazos robóticos):** Tienden a emular las funciones de un brazo humano. Un ejemplo de estos se puede observar en la Figura 2.5.



Figura 2.5. Robot Articulado. Extraído de [11]

- **Humanoides:** Son robots que realizan funciones típicas de las personas y suelen adoptar formas parecidas a las del ser humano, como el mostrado en la Figura 2.6.

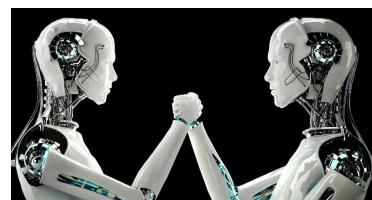


Figura 2.6. Humanoide. Extraído de [12]

2.2. ROBÓTICA INDUSTRIAL EN LA ACTUALIDAD.

- **CoBots o robots colaborativos:** Están diseñados para funcionar junto a los humanos o directamente con ellos, como por ejemplo el robot de la Figura 2.7.



Figura 2.7. Cobot. Extraído de [13]

- **Robots híbridos:** Son distintos tipos de robots que se combinan para crear soluciones capaces de realizar tareas más complejas, tal y como se observa en la Figura 2.8.



Figura 2.8. Robot híbrido. Extraído de [14]

En este TFG se trabajará con tres brazos articulados, todos ellos compuestos por seis segmentos metálicos, unidos por tres articulaciones, lo que proporciona seis grados de libertad. Esta disposición se puede observar en la Figura 2.9.

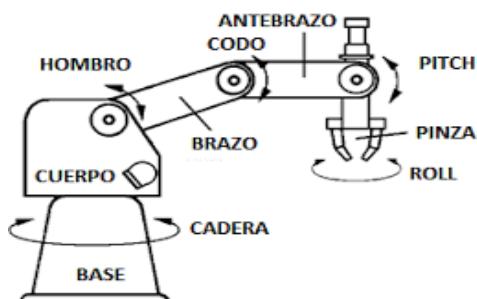


Figura 2.9. Partes de un brazo robótico. Extraído de [15]

En este caso, el brazo robótico presenta una gran similitud con un brazo humano (hombro, codo y muñeca). Donde, normalmente el hombro se encuentra montado en una estructura de base fija, mientras que el resto articulaciones son libres, y disponen de servomotores independientes que permiten su movimiento.

Para comprender el comportamiento de un brazo robótico y abordar este TFG de una forma adecuada, es necesario conocer los distintos componentes del robot, ya que estos condicionarán sus capacidades y, por tanto, las necesidades de este proyecto. El brazo robótico está formado por cinco componentes principales, los cuales se pueden observar en la Figura 2.10 [59, 60].

- **Controlador:** El controlador del robot es un dispositivo o sistema capaz de gestionar y coordinar los movimientos y funciones del robot. Para ello traduce las órdenes procedentes del usuario o programa y las combina con información de los sensores, generando acciones físicas precisas, ajustando la velocidad y posición de cada articulación en tiempo real.
- **Sensores:** Los sensores proporcionan a los robots industriales información sobre su espacio de trabajo. Los más comunes son los sistemas ópticos, estos permiten al robot adaptarse dinámicamente a su entorno de trabajo enviando señales al controlador del robot.
- **Brazo robótico:** Es el conjunto de eslabones y articulaciones que se utiliza para posicionar el efecto final en la posición deseada.
- **Efecto final:** se encuentran unidos al extremo de un brazo robótico y actúan como la mano del robot, sus características varían en función del tipo de aplicación. Algunos robots están equipados con múltiples efectores finales, por lo que pueden cambiarse automáticamente con el uso de un intercambiador de herramientas, pudiendo realizar diferentes tipos de aplicaciones.
- **Accionamiento:** son motores paso a paso conectados a cada una de las articulaciones de los robots y que permiten efectuar el movimiento de estas de una forma precisa. Estos motores pueden ser hidráulicos, eléctricos o neumáticos.

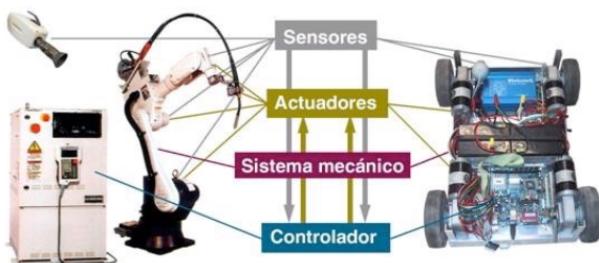


Figura 2.10. Esquema de control de un robot. Extraído de [16]

2.3. MÉTODOS DE PROGRAMACIÓN Y CONTROL DE UN ROBOT REAL

La robótica industrial se encuentra en una continua evolución, impulsada por avances tecnológicos como la Industria 4.0, la inteligencia artificial y la RV. La integración de estos sistemas permitirá una mayor automatización, adaptabilidad y eficiencia en los procesos de fabricación. La Industria 4.0 promueve la interconexión de máquinas y sistemas, optimizando la producción y el mantenimiento predictivo. La inteligencia artificial dotará a los robots de capacidades de aprendizaje y toma de decisiones en tiempo real, mejorando su rendimiento y versatilidad. Por su parte, la RV facilitará la programación y simulación de robots, permitiendo un diseño y prueba más intuitivos y eficientes. Estas innovaciones auguran un futuro en el que la robótica industrial será aún más esencial para enfrentar los desafíos y demandas del mercado global [61, 62].

2.3. Métodos de programación y control de un robot real

La programación de robots ha evolucionado a lo largo de los años, impulsados por los avances tecnológicos y la creciente demanda de la automatización en la industria. A medida que los robots han evolucionado, estos deben realizar tareas más diversas y con requisitos más restrictivos, lo que obliga a la evolución y aparición de nuevas necesidades en la interacción humano-robot. De esta forma podemos diferenciar distintas formas de programación, entre las que caben destacar [63, 64]:

- **Manual mediante Teach Pendant:** Este tipo de programación es considerada la más básica y es común en todo tipo de robots. No requiere dispositivos externos adicionales, siendo intuitivo y accesible tanto para técnicos como para operarios del robot. La tecnología de *Teach Pendant* consiste en una botonera o un dispositivo táctil con opciones básicas de programación, permitiendo la elaboración de programas y simplicidad a la hora de realizar desplazamientos. En la siguiente Figura 2.11, se muestra el *Teach Pendant* de ABB.



Figura 2.11. *Teach Pendant* de ABB. Extraído de [17]

- **Programación *offline*:** Se trata de un método clásico, donde un técnico utiliza software de control del robot, como RobotStudio para robots ABB, o software libre como ROS o MATLAB, que pueden usarse con diferentes tipos de robots. Este tipo de programación se realiza mediante lenguajes como C/C++, Python, Fortran o RAPID, en los cuales se utilizan líneas de código para definir el comportamiento del robot. Algunas aplicaciones incluyen métodos de programación por bloques, útiles para principiantes. La principal ventaja de este método de programación, es que estas aplicaciones suelen incluir un entorno de simulación, que permite estudiar y corregir el comportamiento del robot en un entorno virtual seguro.
- **Programación por demostración:** En este método, un operario realiza un movimiento utilizando un dispositivo electrónico o mecánico, como un guante con sensores de presión. La información del movimiento se registra y luego se repite por el robot. Por ejemplo, un operario puede agarrar un objeto de cierta manera, y los datos de presión se transforman en posiciones objetivo para el robot, permitiendo que este repita el mismo movimiento de agarre.
- **Programación mediante sensores auditivos:** En este tipo de programación el robot dispone de una serie de actividades preconfiguradas asociadas a palabras o frases específicas. De esta forma, el operario interactúa con el robot de forma oral y el robot dispone de un micrófono capaz de captar esta información e interpretarla. Si la información recibida concuerda con alguna de las opciones preconfiguradas, se ejecutará la orden en cuestión.
- **Programación mediante sensores ópticos:** Este tipo de programación es similar al auditivo, sin embargo, en este caso el programa de gestión de datos debe procesar imágenes en busca de determinados gestos del operario o distintas posiciones de un objeto y si alguno de ellos se corresponde con una acción preconfigurada se llevará a cabo.
- **Programación mediante Inteligencia artificial y Aprendizaje automático:** En este caso, no se lleva a cabo una programación de acciones, sino que el robot se entrena mediante algoritmos de inteligencia artificial y aprendizaje automático. Además, se suelen dotar de una gran cantidad de sensores de diversos tipos, permitiendo al robot combinar la información de varios sensores para comprender la situación en la que se encuentra y así adaptarse y llevar a cabo la acción adecuada de una forma optimizada.
- **Programación mediante RV:** Este es un método de programación novedoso, que puede definirse como una programación manual del robot, pero mediante un entorno virtual programado y configurado correctamente. Además, el entorno de RV representa un lugar seguro e interactivo donde llevar a cabo las simulaciones. Este tipo de programación se desarrollará a lo largo de este TFG.

2.4. CONCEPTOS Y DESARROLLO DE LA RV

Comprender las distintas formas de programación y control de robots es crucial para este TFG, ya que permite identificar las ventajas y limitaciones de cada método. Esta base de conocimiento es fundamental para el desarrollo de una aplicación de RV que facilite la programación intuitiva y efectiva de un brazo robótico, integrando y combinando los aspectos más representativos de estas tecnologías para mejorar la interacción humano-robot.

2.4. Conceptos y desarrollo de la RV

Para comprender el potencial y relevancia de este TFG, es necesario familiarizarse con las distintas tecnologías inmersivas y comprender las capacidades y aplicaciones que ofrece cada una de ellas y las posibilidades en que estas pueden contribuir a la interacción humano-robot. Estudiar estas tecnologías permite comprender cómo pueden aplicarse e integrarse en el ámbito de la robótica, para mejorar la eficiencia, la precisión y la accesibilidad de los sistemas robóticos.

2.4.1. Realidad Virtual (RV)

La RV se basa en la utilización de motores gráficos para llevar a cabo la creación de un entorno virtual, el cual puede ser una representación de un espacio real o una escena imaginaria, pero que, en ambos casos, transmite a los usuarios una sensación de completa realidad.

La inmersión se lleva a cabo mediante un casco o gafas de RV el cual aísla al usuario del exterior, colocando frente a sus ojos dos pantallas que proporcionan dos ángulos de vista distintos de una misma escena, lo que permite experimentar la sensación de inmersión y profundidad en la escena. Además, mediante el uso de sensores incorporados en el casco, permite reconocer el movimiento de la cabeza y trasladarlo a la escena, haciendo que la visión del entorno simulado sea tan real como la propia realidad. Esta tecnología generalmente se utiliza de forma conjunta con unos mandos que permiten la interacción y desplazamiento por el entorno simulado. Pero la experiencia inmersiva va mucho más allá y es que en la actualidad existen simuladores de RV profesionales dotados de tecnologías que permiten transmitir el resto de las acciones del cuerpo al entorno simulado. [65, 66]. Algunas de estas tecnologías más conocidas son:

- **Guantes Hapticos:** Estos guantes se pueden observar en la Figura 2.12 y se colocan en las manos del usuario proporcionando una retroalimentación táctil, es decir el movimiento real de las manos del usuario se transmite de forma directa al entorno virtual y del mismo modo, si un objeto virtual interacciona con el avatar, el usuario real puede percibir esta interacción mediante sensores de presión [67].



Figura 2.12. Guantes Hápticos. Extraído de [18]

- **VR Walking Platform:** Este es uno componentes más novedosos y que junto a las gafas constituyen los dispositivos más inmersivos, ya que mediante un sistema de sensores de presión y cintas o plataformas deslizantes permiten transmitir el movimiento de desplazamiento, que el usuario realiza de forma estática, en un movimiento virtual. Es decir, con estos dispositivos permiten andar, correr y saltar en el entorno virtual a partir de los propios movimientos del usuario. Un ejemplo de este tipo de dispositivo, se puede observar en la Figura 2.13 [66].



Figura 2.13. VR Walking Platform. Extraído de [19]

Existen muchos más dispositivos destinados principalmente al entretenimiento, como son las simulaciones de armas o chalecos de presión que pueden llevar la experiencia de videojuegos a nivel de la realidad.

La RV tiene infinidad de aplicaciones, no es de extrañar que la más notable sea el entretenimiento y los videojuegos, ya que las primeras aplicaciones de RV disponibles para los usuarios consistían en vídeos que podían ser reproducidos en cualquier dispositivo móvil y mediante una carcasa de plástico se reproducían frente a los ojos. Ante el desarrollo de esta tecnología, compañías como PlayStation comenzaron a lanzar los primeros productos, destinados a usar sus propios juegos de una forma mucho más realista.

2.4. CONCEPTOS Y DESARROLLO DE LA RV

Pero la RV no solo tiene como objetivo el entretenimiento. Uno de los campos donde más impacto ha tenido es la educación, y es que, sin moverse de clase, cualquier estudiante puede tener acceso a experiencias inmersivas donde visitar distintos lugares del mundo, conocer sus culturas e incluso interactuar con sus habitantes. Además, en el ámbito de la educación y la ingeniería permite disponer de cualquier mecanismo o dispositivo e interactuar con él para conocer cada una de sus partes o su comportamiento.

En este mismo ámbito educativo, uno de los ejemplos más llamativos es el de la medicina 2.14 2.15 y es que el poder enseñar cualquier parte del cuerpo de forma física y permitiendo explorar todo su funcionamiento o composición ha llevado este tipo de enseñanzas a otro nivel. Además, se han desarrollado entornos de simulación avanzados que permiten a médicos reales simular en primera persona cualquier tipo de operación, conociendo así cualquier complicación que pudiese surgir o practicando distintos abordajes [68].

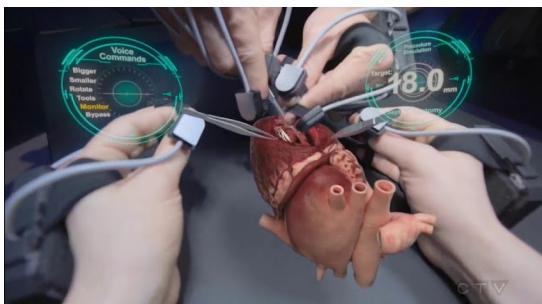


Figura 2.14. Simulación operación. Extraído de [20]



Figura 2.15. Aprendizaje partes corazón con VR. Extraído de [21]

2.4.2. Realidad Aumentada

La realidad aumentada (RA) generalmente se lleva a cabo mediante un dispositivo móvil dispuesto de una cámara, la cual permite captar el entorno real y cargar, sobre la representación en pantalla de este, contenido virtual. Este contenido virtual puede añadir información adicional sobre el objeto o entorno enfocado, o hacer que aparezcan sobre estos objetos ficticios con los que poder interactuar. La RA puede clasificarse en [69,70]:

- **Mediante el uso de marcadores:** Este tipo de tecnología es la más utilizada por ser la más simple y la que menos recursos precisa. Se necesita una aplicación específica, un *smartphone* con cámara y un marcador o imagen en 2D. En este caso, se escanea continuamente el entorno captado por la cámara en busca del marcador, una vez localizado dicho marcador en el entorno real, se posiciona sobre este el objeto 3D que se desea representar.
- **Sin marcadores:** En este tipo de RA, no existen marcadores, sino que la aplicación es capaz de analizar el entorno capturado por la cámara y generar un mapa tridimensional,

para así poder posicionar un objeto 3D en cualquiera de sus partes, de la forma adecuada. Un tipo específico de RA sin marcadores es la RA por geolocalización, donde mediante la tecnología de mapeo y la posición del dispositivo, permite posicionar el objeto 3D en una posición específica del entorno.

Uno de los ejemplos más conocidos de RA es el famoso juego de Pokémon Go, mostrado en la Figura 2.16, que permite localizar distintas criaturas y objetos ficticios por las calles de una ciudad.

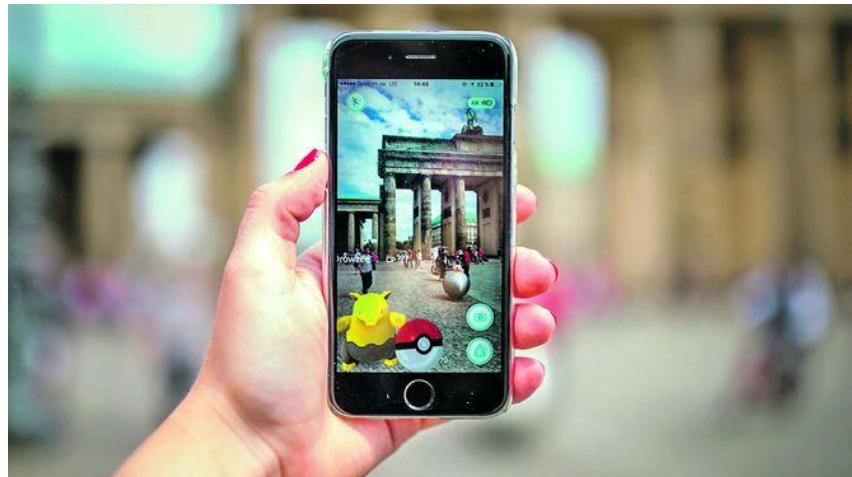


Figura 2.16. Pokémon Go. Extraído de [22]

Pero esta tecnología no solo se utiliza en juegos, sino que es muy versátil y sencilla de aplicar a cualquier ámbito. Por lo que contribuye a diversos campos, como la educación, donde puede ser de gran utilidad, ya que ofrece una visión más interactiva a la hora de explorar los conocimientos, sin necesidad de disponer de un entorno virtual completo, como en el caso de la RV.

En el turismo, la cultura y comercio ha tenido un gran impacto, ya que mediante la geolocalización permite que los turistas conozcan en todo momento información o el estado original del lugar donde se encuentran, mediante guías interactivas. Incluso se hacen aparecer figuras de personajes conocidos o autores de obras que transmiten información relacionada con esta, como por ejemplo en la Figura 2.17, se observa el renacimiento de ruinas romanas, en el Teatro Romano de Cartagena.

En la industria también se está comenzando a aplicar esta tecnología, sobre todo en temas de mantenimiento de equipo industrial, donde a partir de un determinado dispositivo puede obtenerse su información o una representación del mantenimiento a realizar.



Figura 2.17. Renacimiento ruinas romanas, Teatro Romano de Cartagena. Extraído de [23]

2.4.3. Realidad Mixta

La RM es la tecnología inmersiva más novedosa y completa, porque combina la tecnología de RV y RA, donde los elementos característicos de ambas coexisten e interactúan en tiempo real. De esta forma, la RM integra el mundo real y virtual, ya que permite que los objetos virtuales puedan interactuar con los objetos físicos y viceversa. Para lograr este nivel de integración es necesario usar sensores avanzados, cámaras y tecnologías de seguimiento que permitirán a los dispositivos de RM mapear el entorno y anclar objetos físicos en él, de una forma coherente [71–73].

Para dotar a esta experiencia inmersiva de una interacción natural, los usuarios dispondrán de un dispositivo de reconocimiento de movimiento, como unos guantes, o será el propio sistema de visionado el que centre parte de sus dispositivos en interpretar el movimiento de las manos del usuario. Así, el usuario puede utilizar sus propios gestos o comandos de voz para interaccionar con los objetos virtuales. Para garantizar que la interacción se lleve a cabo de una forma eficiente y realista, es necesario utilizar tecnología avanzada que garantice un seguimiento preciso y fluido de todos los elementos del entorno y una respuesta en tiempo real. De esta forma, la mayor parte de las aplicaciones de la RV o RA se pueden extrapolar a esta tecnología. Uno de los ejemplos más representativo en la actualidad es el de la tecnología Meta Quest o Apple Vision Pro que en su funcionamiento predeterminado actúan como sistemas de RM, donde el usuario permanece en el entorno físico, pero además, puede añadir a este numerosas pantallas o avatares que le ayudarán en su desempeño diario.

2.5. Aplicaciones de la realidad virtual en la robótica

La RV, ofrece un gran número de soluciones tecnológicas para distintas áreas, entre las que encontramos [74, 75]:

- **Medicina:** Los últimos avances han permitido incorporar la RV en la medicina, ya que,

permiten desde prácticas quirúrgicas vitales que generan un aprendizaje motriz realista, hasta la disminución de los efectos del Párkinson en pacientes.

- **Comunicación:** Un claro ejemplo de la RV en el entorno de la comunicación son los documentales interactivos o *streaming* de noticias en 360° que transportan a los usuarios a cualquier parte, de manera realista.
- **Mantenimiento:** Las aplicaciones de RV permiten asistencia remota completa en ciertas operaciones, con la posibilidad de incluir mecanismos colaborativos controlables mediante dichas aplicaciones.
- **Prototipado:** BMW revolucionó la industria del diseño automovilístico, gracias a la incorporación de sistemas de RV para el prototipado de sus vehículos, ejemplo que se puede observar en la Figura 2.18. Esto permitió probar las distribuciones del interior del vehículo antes de terminarlo, sentando las bases de una nueva forma de diseño, ampliamente desarrollada en los últimos años.
- **Arquitectura:** De forma similar al prototipado, un arquitecto puede simular el estado final de una construcción y visualizar de una manera más precisa el resultado permitiendo realizar modificaciones previas a la finalización del proyecto.
- **Formación técnica:** Es posible formar a los alumnos con el manejo de máquinas industriales y equipo pesado, mediante la realidad virtual, evitando así el riesgo que estos puedan generar.
- **Preparación en solución de incidentes:** Es posible simular entornos realistas de incidentes, para validar las acciones de los individuos ante una contingencia, simulacros de accidentes, incendios, derrumbes, escapes, etc.
- **Aplicación militar:** El entrenamiento de combate en entornos simulados resulta muy atractivo para la tecnología militar, funcionando de forma similar a los videojuegos.



Figura 2.18. Prototipo BMW. Extraído de [24]

2.5. APLICACIONES DE LA REALIDAD VIRTUAL EN LA ROBÓTICA

Capítulo 3

Materiales y métodos

3.1. Aplicaciones software

En el panorama contemporáneo de la ingeniería y la ciencia computacional, el uso efectivo de herramientas de *software* especializadas se ha convertido en un componente esencial para la investigación y el desarrollo en una variedad de campos. Entre las aplicaciones objeto de estudio en este trabajo, se destacan el motor gráfico Unity, RobotStudio y ROS. Cada una de estas aplicaciones posee características distintivas que las hacen fundamentales en el desarrollo de este TFG.

3.1.1. Motor gráfico Unity

Para desarrollar una aplicación compatible con la tecnología de RV es necesario utilizar un motor gráfico avanzado, que permita desarrollar y simular entornos digitales con una gran cantidad de cuerpos tridimensionales detallados y realistas dotados de propiedades físicas y comportamientos similares a los de la realidad [76].

Unity es un motor gráfico y una plataforma de desarrollo integral para la creación de videojuegos en 2D y 3D. Entre sus características principales destacan [77, 78]:

- **Editor visual:** Permite diseñar y organizar escenas y objetos de manera visual.
- **Motor de físicas:** Simulación de físicas en tiempo real para crear movimientos y comportamientos realistas.
- **Scripting en C#:** Proporciona un alto grado de control y personalización del comportamiento de los objetos del juego.
- **Multiplataforma:** Permite desarrollar juegos para ordenadores, consolas, móviles y RV.

3.1. APLICACIONES SOFTWARE

- **Asset Store:** Dispone de tienda donde los desarrolladores pueden comprar y vender recursos como modelos 3D, texturas y *scripts*.
- **Soporte para gráficos avanzados:** Herramientas para renderizado de gráficos de alta calidad, incluyendo iluminación, sombreado y efectos especiales.
- **Herramientas de colaboración:** Facilita el trabajo en equipo mediante herramientas de control de versiones y colaboración en proyectos.

Unity opera mediante un ciclo de desarrollo iterativo, donde los desarrolladores crean y ajustan los elementos del juego en el editor, prueban el juego en el simulador o en dispositivos reales, y repiten el proceso para refinar y optimizar el juego. El motor utiliza un sistema de componentes, donde los objetos del juego son combinaciones de componentes que determinan su apariencia y comportamiento. En el contexto de desarrollo, el proceso típico de elaboración de una aplicación incluye:

- **Importar assets:** Traer modelos 3D, texturas, sonidos y otros recursos al proyecto.
- **Configurar escenas:** Diseñar niveles o escenarios del juego colocando y organizando objetos.
- **Programar comportamiento:** Usar *scripts* en C# para definir cómo interactúan los objetos y cómo responde el juego a las acciones del jugador.
- **Probar y ajustar:** Ejecutar el juego para encontrar y corregir errores, optimizar el rendimiento y mejorar la jugabilidad.
- **Exportar y publicar:** Compilar el juego para las plataformas deseadas y lanzarlo al público.

Unity es ampliamente utilizado tanto por desarrolladores novatos como por profesionales debido a su accesibilidad y a la gran comunidad de usuarios que contribuyen con tutoriales, *plugins* y soporte.

Además, es una plataforma líder en el desarrollo de experiencias en realidad virtual, proporcionando integración con dispositivos RV de distintos fabricantes, herramientas para la interacción en entornos virtuales y optimización del rendimiento para asegurar experiencias fluidas y eficientes. La capacidad de crear gráficos realistas y sonidos envolventes en RV mejora significativamente la sensación de inmersión. Unity combina facilidad de uso con capacidades avanzadas, convirtiéndose en una herramienta esencial en la industria del desarrollo de videojuegos y aplicaciones de realidad virtual. [79]

3.1.2. RobotStudio

RobotStudio, cuyo logotipo se muestra en la Figura 3.1, es un *software* desarrollado por ABB, líder mundial en tecnologías de automatización y robótica, que proporciona un entorno virtual para la simulación, programación y optimización de sistemas robóticos industriales. Esta aplicación permite a los ingenieros y programadores, diseñar y configurar células robóticas completas antes de implementarlas en el entorno de producción real. Además, ofrece herramientas avanzadas para la programación de robots, la optimización de trayectorias, la verificación de colisiones y la realización de análisis de rendimiento, lo que contribuye significativamente a mejorar la eficiencia y la seguridad en los procesos industrializados [80–82].

Entre sus funcionalidades principales, se pueden destacar:

- **Simulación Precisa:** RobotStudio ofrece una representación precisa del comportamiento del robot en un entorno virtual, lo que permite a los usuarios validar y optimizar sus programas antes de la implementación en el mundo real.
- **Programación Intuitiva:** Mediante una interfaz intuitiva y herramientas de programación avanzadas, los usuarios pueden desarrollar y depurar programas para robots industriales de forma eficiente y efectiva. Cabe destacar que Robotstudio utiliza como lenguaje de programación Rapid, que es un lenguaje especializado para la programación de robots.
- **Optimización de Trayectorias:** La aplicación permite optimizar las trayectorias de los robots para mejorar la eficiencia en la ejecución de tareas y minimizar el tiempo de ciclo.
- **Verificación de Colisiones:** RobotStudio lleva a cabo un análisis de colisiones automático para identificar posibles conflictos entre el robot, las herramientas y el entorno de trabajo, lo que ayuda a prevenir accidentes y daños en el equipo.
- **Análisis de Rendimiento:** Los usuarios pueden llevar a cabo análisis de rendimiento para evaluar la productividad y la eficiencia de sus sistemas robóticos, identificando áreas de mejora y optimización.

Robotstudio ofrece amplias oportunidades a la hora de programar sus robots ABB, aunque no es compatible con el uso de robots de otras marcas. Además, una de las principales limitaciones de Robotstudio y los robots ABB es que a la hora de programarlos, Rapid no dispone de librerías de recursos como otros lenguajes de programación, sino que se habla de módulos que deben ser comprados al fabricante, lo que limita las funcionalidades de los robots.

3.1. APLICACIONES SOFTWARE



Figura 3.1. RobotStudio. Extraído de [25]

El papel fundamental de RobotStudio en el desarrollo de este TFG es programar los controladores digitales que recibirán y procesarán la información procedente de Unity para generar los movimientos adecuados en el robot. El objetivo es programar dicho controlador digital y comprobar su validez mediante el entorno de simulación de Robotstudio, para luego descargar el código generado en el controlador físico del robot.

3.1.3. *Robot Operating System (ROS)*

ROS, cuyo logo se muestra la Figura 3.2), es un conjunto de herramientas de código abierto desarrollado por la comunidad para facilitar el desarrollo de software para robots. Aunque recibe el nombre de "sistema operativo", ROS no es un sistema operativo en el sentido tradicional, sino un conjunto de bibliotecas, herramientas y convenciones que proporcionan funcionalidades comunes requeridas en el desarrollo de *software* para robots. En cuanto a sus principales funcionalidades, cabe destacar [83,84]:

- **Arquitectura de Comunicación:** ROS proporciona una arquitectura de comunicación distribuida que permite a los diferentes componentes de un sistema robótico comunicarse entre sí de manera eficiente. Utiliza un sistema de mensajes asíncrono basado en temas (*topics*) y servicios, lo que facilita la integración de diferentes módulos de *software* y *hardware*.
- **Gestión de Paquetes:** ROS utiliza un sistema de gestión de paquetes que facilita la instalación, actualización y distribución de software para robots. Los paquetes de ROS pueden contener código fuente, bibliotecas, archivos de configuración y modelos 3D.

- **Herramientas de Desarrollo:** ROS proporciona una variedad de herramientas de desarrollo que facilitan la creación, depuración y pruebas de software para robots. Estas herramientas incluyen un sistema de compilación (*catkin*), un entorno de desarrollo integrado (IDE) compatible con varias plataformas, herramientas de visualización y depuración de gráficos, y simuladores robóticos como Gazebo.
- **Soporte Multiplataforma:** ROS es compatible con una amplia variedad de plataformas de *hardware* y sistemas operativos, lo que permite a los desarrolladores crear aplicaciones robóticas para una variedad de robots y entornos. Además, ROS es compatible con una amplia gama de lenguajes de programación, incluyendo C++, Python y otros.



Figura 3.2. ROS. Extraído de [26]

En comparación con RobotStudio, ROS es un entorno mucho más versátil, por la amplia gama de oportunidades que ofrece en cuanto a programación. En el desarrollo de este TFG, ROS juega un papel fundamental, ya que permitirá realizar la conexión entre Unity y el controlador del robot Schunk LWA 4P, al igual que lo hacía Robotstudio entre Unity y los robots ABB.

En este caso no se desarrollará un programa ROS, ya que se utilizará la interfaz desarrollada [85], la cual permite controlar los desplazamientos del robot mediante las posiciones de las articulaciones y el tiempo de movimiento. Para ello será necesario dotar a dicha interfaz con un protocolo de comunicación que permita la conexión con Unity y un sistema capaz de gestionar la información recibida.

3.2. Métodos

En este apartado, se abordarán dos aspectos fundamentales para el desarrollo de la aplicación de RV: métodos cinemáticos y protocolos de comunicación.

3.2.1. Cinemática Directa

El Modelo Cinemático Directo (MCD) es fundamental en robótica, permitiendo calcular la posición y orientación del elemento terminal de un robot, en función de las posiciones de sus

3.2. MÉTODOS

articulaciones. Este enfoque se basa en la cinemática, que es la rama de la mecánica que estudia el movimiento de los cuerpos sin tener en cuenta las fuerzas que lo producen. El objetivo del MCD es determinar la transformación espacial entre el sistema de coordenadas de la base del robot y el sistema de coordenadas del extremo del efecto final (la herramienta o la parte final del robot que interactúa con el entorno) [86, 87].

El MCD se fundamenta en la geometría y la trigonometría para modelar la relación entre las articulaciones del robot, su posición y orientación en el espacio. Utilizando la geometría de las articulaciones, junto con las longitudes de los eslabones y los ángulos de articulación, se pueden derivar las ecuaciones cinemáticas que describen la posición y orientación del efecto final del robot [88, 89].

Estas ecuaciones cinemáticas pueden expresarse de diversas formas, dependiendo de la configuración del robot. Para robots con estructuras seriales, como el popular robot manipulador de tipo SCARA (*Selective Compliance Assembly Robot Arm*) o el robot antropomórfico de tipo PUMA (*Programmable Universal Machine for Assembly*), las ecuaciones cinemáticas pueden expresarse mediante la matriz de transformación homogénea, que representa la posición y orientación en un espacio tridimensional.

Para calcular el MCD, en la asignatura de Robótica impartida en el cuarto curso del grado de Ingeniería Electrónica Industrial y Automática, se emplea el conocido método de Denavit Hartenberg [90]. El MCD es fundamental para una variedad de aplicaciones en robótica, incluyendo la planificación de trayectorias, la simulación de movimientos y el control de robots.

Comprender y aplicar este método es esencial para el desarrollo de este TFG, ya que permitirá conocer en todo momento la posición del extremo del robot a partir de los valores de sus articulaciones. Dichas posiciones permitirán calcular la distancia entre posiciones del robot y ajustar así, de forma adecuada, las velocidades de traslación, respetando los límites de seguridad impuestos.

3.2.2. Cinemática Inversa

El Modelo Cinemático Inverso (MCI) permite determinar el movimiento de una cadena de articulaciones para lograr que un actuador final se ubique en una posición concreta. El cálculo es un problema complejo que consiste en la resolución de una serie de ecuaciones cuya solución normalmente no es única. La búsqueda de la solución suele realizarse mediante el uso de técnicas numéricas iterativas, como por ejemplo el método de Newton-Raphson, descomposición de velocidades y Métodos Heurísticos [91–93].

El objetivo es encontrar los valores que deben tomar las coordenadas articulares del robot para que su extremo se posicione y oriente según una determinada localización espacial. Siempre que se especifica una posición de destino y una orientación en términos cartesianos, debe calcularse la cinemática inversa para poder despejar los ángulos de articulación requeridos.

Al aplicar el MCI, pueden encontrarse los siguientes casos:

- **Múltiples Soluciones:** Un robot puede tener múltiples configuraciones articulares para alcanzar una misma posición del extremo efecto. Se deben establecer criterios adicionales (como minimizar el uso de energía o evitar colisiones) para seleccionar la solución más adecuada.
- **Inexistencia de Solución:** En algunos casos, la posición deseada del extremo efecto puede estar fuera del alcance del robot. Los algoritmos deben manejar estos casos devolviendo una solución aproximada o indicando la imposibilidad de alcanzar la posición.
- **Singularidades:** Posiciones en las que el robot pierde uno o más grados de libertad, haciendo que el control y el movimiento sean más complicados.

El MCI es esencial en robótica y en animación por ordenador, permitiendo el control preciso y eficiente de los movimientos de los robots y personajes animados. A pesar de los desafíos que presenta, como las múltiples soluciones posibles, la inexistencia de soluciones en ciertos casos y las singularidades, los avances en algoritmos y técnicas de optimización continúan mejorando su aplicabilidad y precisión. Este método no solo optimiza la programación y operación de robots industriales, sino que también amplía las capacidades en áreas tan diversas como la biomecánica y la medicina, demostrando su importancia y versatilidad en la tecnología moderna.

En el desarrollo de este TFG juega un papel fundamental a la hora de generar las animaciones en los robots, ya que permite que todos los eslabones del robot ejecuten sus respectivos desplazamientos de una forma coordinada, para alcanzar una posición determinada en el extremo.

3.2.3. Método IK Solver

Una vez comprendidos los conceptos de cinemática directa e inversa, es fundamental saber cómo se aplican en el desarrollo del TFG. Los brazos robóticos en el entorno simulado pueden moverse; para ello, se toma el elemento terminal y se desplaza hacia el punto deseado. En un robot real, las uniones físicas entre los eslabones hacen que todos los eslabones se adapten a la posición del elemento terminal, ya que estas uniones son rígidas. Sin embargo, en un robot simulado, estas uniones no tienen restricciones. Por lo tanto, para que todos los eslabones se posicionen en función del extremo del robot, es necesario conocer el ángulo de giro de cada articulación. Esto se logra aplicando el método cinemático inverso [94, 95].

Aunque existen métodos para calcular la cinemática inversa, estos procedimientos son complejos y requieren una gran carga computacional, lo que es problemático, ya que la aplicación necesita calcular el valor de cada articulación en cada *frame*. En su lugar, se suelen usar métodos iterativos que, aunque pueden generar un pequeño error, son rápidos y aplicables a cualquier es-

3.3. PROTOCOLO DE COMUNICACIÓN

tructura articulada. Estos métodos se conocen como CCD (*Cyclic Coordinate Descent*), también llamados IK *solver* en los programas gráficos.

Este método simplifica el robot a una cadena cinemática. Una vez definida esta cadena, se establecen tres puntos: P_c en la articulación que se va a mover, P_e en el extremo del robot, y P_f en el punto objetivo. Se selecciona la articulación más cercana al extremo del robot, P_c , y se calculan los vectores P_cP_e (desde P_c al extremo del robot) y P_cP_f (desde P_c al objetivo), una imagen ilustrativa sobre este proceso se puede observar en la Figura 3.3 [27]

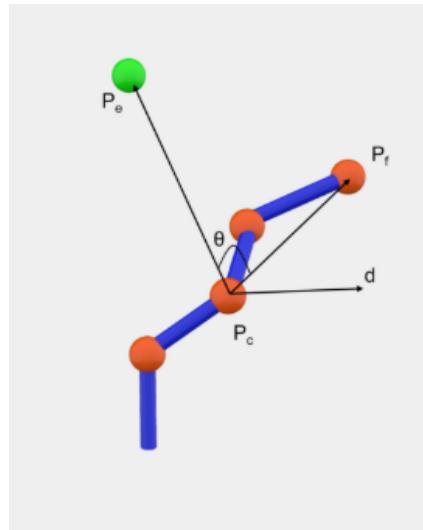


Figura 3.3. Método Ccd IK Solve. Extraído de [27]

Posteriormente, se calcula el ángulo necesario para alinear estos dos vectores y se aplica la rotación a la articulación, moviendo el extremo del robot más cerca del objetivo. Se realiza la misma operación con la siguiente articulación hacia la base del robot, y así sucesivamente hasta completar todas las iteraciones para todos los eslabones.

Después de iterar todas las articulaciones, el extremo del robot estará más cerca del punto final. Este método se aplica de forma iterativa hasta que la distancia entre el extremo del robot y el punto objetivo sea menor que un valor establecido.

Con este método, tras pocas iteraciones, se obtiene una posición muy cercana a la posición objetivo con una menor carga computacional comparada con los cálculos exactos que conllevan aplicar el MCI por métodos Heurístico u otros procedimientos matemáticos.

3.3. Protocolo de comunicación

TCP/IP es el conjunto de protocolos que permite la comunicación entre dispositivos en redes como Internet. Consiste en dos protocolos principales: el Protocolo de Internet (IP) y el Protocolo de Control de Transmisión (TCP). [96]

- **IP:** Este protocolo se encarga de la dirección y el enrutamiento de los paquetes de datos. Cada dispositivo conectado a una red tiene una dirección IP única que permite su identificación. IP fragmenta los datos en paquetes y determina la mejor ruta para su envío a través de múltiples redes hasta llegar al destino. Los paquetes pueden tomar diferentes rutas y llegar en distinto orden.
- **TCP:** TCP garantiza la transmisión confiable de datos entre dos dispositivos. Antes de enviar los datos, TCP establece una conexión entre el dispositivo emisor y el receptor. Los datos se dividen en segmentos que son enviados en paquetes. TCP asegura que todos los paquetes lleguen correctamente, sin errores y en el orden adecuado. Si algún paquete se pierde o se daña, TCP solicita su retransmisión.

El funcionamiento de TCP/IP comienza con la encapsulación de datos en el dispositivo emisor. Los datos se dividen en paquetes y se envían a través de la red. IP gestiona la dirección y el enrutamiento de los paquetes, mientras que TCP se encarga de la integridad y el orden de los datos. Al llegar al destino, TCP reensambla los paquetes para reconstruir los datos originales y los entrega a la aplicación correspondiente.

TCP/IP permite la interoperabilidad entre diferentes dispositivos y sistemas operativos, haciéndolo esencial para la comunicación en redes globales. Su capacidad de manejar y asegurar la transmisión de datos lo hace ideal para redes pequeñas y grandes, como Internet. [97]

La utilización de TCP/IP implica una serie de ventajas con respecto a la utilización de otro tipo de protocolos, las mas representativas son:

- **Interoperabilidad:** Facilita la comunicación entre dispositivos y sistemas operativos diferentes. Fiabilidad: TCP asegura la entrega correcta de datos, controlando errores y retransmitiendo paquetes perdidos o dañados.
- **Escalabilidad:** Soporta desde pequeñas redes locales hasta la vasta red de Internet.
- **Flexibilidad:** Adapta la transmisión de datos según la condición de la red, gestionando el control de flujo y evitando la congestión.
- **Estándares Abiertos:** Desarrollado como un conjunto de estándares abiertos, facilitando la implementación y expansión de redes por distintos proveedores y desarrolladores. TCP/IP es fundamental para la comunicación en redes modernas, asegurando la transmisión correcta y eficiente de datos, y permitiendo la interconexión de dispositivos a nivel global.

3.4. Materiales

El diseño y desarrollo de sistemas robóticos modernos se encuentran intrínsecamente ligados a la selección y aplicación de materiales tecnológicos de vanguardia. En este apartado, se explicará la selección de componentes y dispositivos clave, que desempeñan un papel fundamental en el desarrollo y utilización de este TFG.

3.4.1. Oculus Rift S

Las Oculus Rift S (Figura 3.4), son un dispositivo de RV diseñado para ofrecer una experiencia inmersiva y envolvente en el mundo digital. Desarrolladas por Oculus, una división de Facebook Reality Labs, las Rift S representan una evolución significativa en la tecnología de RV, ofreciendo una combinación de comodidad, rendimiento y calidad visual para una amplia gama de aplicaciones, desde juegos, hasta simulaciones y entrenamiento. Entre las características principales se encuentran [98–100], :

- **Resolución de Pantalla Mejorada:** Las Oculus Rift S cuentan con pantallas LCD con una resolución total de 2560x1440 píxeles (1280x1440 por ojo), lo que proporciona una experiencia visual nítida y detallada. Esta mejora en la resolución ayuda a reducir el efecto de rejilla (*screen-door effect*) y mejora la calidad de las imágenes y texturas en los entornos virtuales.
- **Seguimiento de 6 Grados de Libertad (6DoF):** Las Rift S incorporan un sistema de seguimiento de 6 DoF, que permite a los usuarios moverse libremente dentro del espacio virtual, inclinarse, agacharse y realizar movimientos naturales con las manos y la cabeza. Este seguimiento preciso y fluido mejora la sensación de inmersión y realismo en las experiencias de realidad virtual.
- **Sistema de Seguimiento *Inside-Out*:** Las Rift S utilizan un sistema de seguimiento donde los sensores de seguimiento están integrados en el propio dispositivo. Esto elimina la necesidad de sensores externos y simplifica el proceso de configuración, permitiendo a los usuarios iniciar rápidamente experiencias de realidad virtual sin la necesidad de configuraciones complicadas.
- **Diseño Ergonómico y Cómodo:** El diseño de las Oculus Rift S se ha optimizado para ofrecer comodidad durante sesiones de uso prolongadas. Cuentan con una diadema ajustable y acolchada, así como almohadillas faciales suaves y transpirables que se adaptan cómodamente a una variedad de formas faciales.
- **Compatibilidad con PC:** Las Oculus Rift S están diseñadas para funcionar con una PC compatible, lo que proporciona acceso a una amplia variedad de juegos, aplicaciones y experiencias de realidad virtual disponibles en la plataforma *Oculus Storey* otras tiendas en línea.



Figura 3.4. Oculus Rift S. Extraído de [28]

Para utilizar las Oculus Rift S, es necesario que el PC cumpla con ciertos requisitos mínimos de hardware y software para garantizar un rendimiento óptimo y una experiencia de realidad virtual satisfactoria. A continuación, se detallan los requisitos mínimos recomendados por Oculus:

- **Central Processing Unit (CPU):** Procesador Intel Core i3-6100 / AMD Ryzen 3 1200, o superior.
- **Memoria RAM (Random Access Memory):** 8 GB de RAM o más.
- **Tarjeta Gráfica:** Tarjeta gráfica compatible con DirectX 12 y al menos 6 GB de VRAM. Las tarjetas gráficas son: NVIDIA GTX 1050 Ti / AMD Radeon RX 470 o superior.
- **Puertos de Conexión:** Un puerto *DisplayPort* compatible con *DisplayPort* 1.2 y un puerto USB 3.0.
- **Sistema Operativo:** Windows 10.

Por otro lado, los requisitos expuestos anteriormente son los requisitos mínimos del sistema, es decir, aquellos que permitirán la utilización de la tecnología Oculus, pero estos no garantizan el correcto funcionamiento del dispositivo, para ello se recomiendan los siguientes requisitos:

- **CPU:** Procesador Intel Core i5-4590 / AMD Ryzen 5 1500X, o superior.
- **Memoria RAM:** 16 GB de RAM.
- **Tarjeta Gráfica (GPU):** Tarjeta gráfica NVIDIA GTX 1060 / AMD Radeon RX 480 o superior.
- **Puertos de Conexión:** Se recomienda disponer de varios puertos USB 3.0 para otros dispositivos adicionales, como controladores y periféricos.
- **Sistema Operativo:** Windows 10.

3.4. MATERIALES

Es importante tener en cuenta que estos son los requisitos mínimos y recomendados proporcionados por Oculus. Sin embargo, el rendimiento puede variar según la configuración específica del PC y las aplicaciones utilizadas en la experiencia de realidad virtual. Además, se recomienda tener actualizados los controladores de gráficos y otros componentes del sistema para garantizar la compatibilidad y estabilidad.

Por último, Oculus ofrece una herramienta llamada “*Oculus Compatibility Check*” que puede ayudarte a verificar si tu PC cumple con los requisitos mínimos para utilizar las Oculus Rift S.

3.4.2. IRB 140

El ABB IRB 140 (Figuras 3.5 y 3.6) es un robot industrial, que en su lanzamiento representó la vanguardia en tecnología de automatización, diseñado meticulosamente para una amplia gama de aplicaciones en la industria moderna. Con un enfoque en la versatilidad, la precisión y la eficiencia, este robot ofrece soluciones robustas para desafíos de fabricación, ensamblaje y manipulación en entornos diversos y exigentes [101].

En cuanto al diseño y la construcción, el IRB 140 es un robot compacto y elegante, optimizado para maximizar la eficiencia y la flexibilidad en las operaciones industriales. Con una estructura modular y una construcción robusta, este robot es capaz de resistir condiciones adversas y cumplir con los estándares de rendimiento más exigentes, incluso permitiendo trabajos subacuáticos. Por otro lado, las características principales son:

- **Capacidad de Carga Útil:** El IRB 140 está disponible en varias versiones con diferentes capacidades de carga útil, que van desde 5 kg hasta 6 kg, lo que permite adaptarse a una amplia variedad de aplicaciones y requisitos de manipulación.
- **Alcance y Movilidad:** Con un alcance máximo de varios metros, este robot es capaz de acceder a áreas de trabajo amplias y realizar una amplia gama de movimientos con precisión y rapidez.
- **Precisión y Repetibilidad:** Equipado con avanzados sistemas de control y tecnología de servoaccionamiento, el IRB 140 ofrece una precisión excepcional y una repetibilidad precisa en cada ciclo de trabajo, garantizando la calidad y la consistencia en las operaciones de fabricación y ensamblaje [102].

El IRB 140 se utiliza en una amplia variedad de industrias y aplicaciones, incluyendo automoción, electrónica, metalurgia, alimentos y bebidas, entre otras. Algunas de las aplicaciones comunes incluyen ensamblaje de componentes, paletizado, manejo de materiales, soldadura, pintura y manipulación de piezas.

Con herramientas de programación intuitivas y una interfaz de usuario amigable, el IRB 140 se integra fácilmente en sistemas de automatización existentes y se programa rápidamente para



Figura 3.5. IRB 140. Extraído de [29]

adaptarse a requisitos específicos de producción. Esto permite a los fabricantes y operadores maximizar la eficiencia y la productividad en sus operaciones.

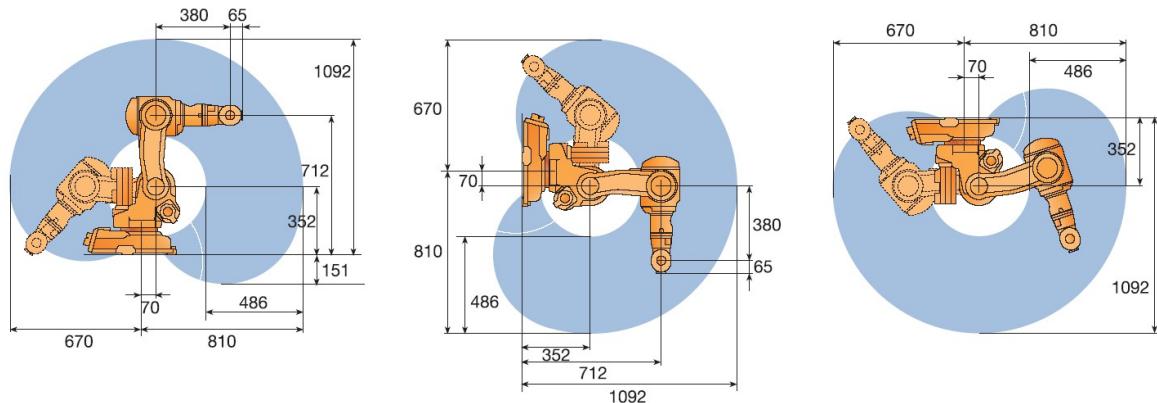


Figura 3.6. Espacio de trabajo de IRB 140. Extraído de [30]

3.4.3. IRB 1090

El ABB IRB 1090 (Figuras 3.7 y 3.8) es un robot industrial diseñado para aplicaciones exigentes en entornos de fabricación, ensamblaje y manipulación. Fabricado por ABB Robotics, líder mundial en tecnologías de automatización, el IRB 1090 ofrece una combinación de potencia, precisión y fiabilidad, lo que lo convierte en una solución versátil para una variedad de aplicaciones industriales [103–105].

El IRB 1090 presenta un diseño robusto y duradero, construido para resistir condiciones de trabajo exigentes y garantizar un rendimiento óptimo en entornos industriales. Con una estructura modular y componentes de alta calidad, este robot ofrece una larga vida útil y un funcionamiento fiable en aplicaciones de alta velocidad y carga.

En cuanto a las características principales encontramos:

3.4. MATERIALES

- **Capacidad de Carga Útil:** El IRB 1090 está diseñado para manejar cargas pesadas con facilidad, con una capacidad de carga útil de hasta 90 kg. Esto lo hace adecuado para una amplia gama de aplicaciones que requieren manipulación de materiales pesados y ensamblaje de componentes grandes.
- **Alcance y Movilidad:** Con un alcance máximo de varios metros, este robot es capaz de acceder a áreas de trabajo amplias y realizar una amplia gama de movimientos con precisión y rapidez, lo que lo hace ideal para aplicaciones de paletizado, manipulación de materiales y ensamblaje.
- **Precisión y Repetibilidad:** Equipado con avanzados sistemas de control y tecnología de servoaccionamiento, el IRB 1090 ofrece una precisión excepcional y una repetibilidad precisa en cada ciclo de trabajo, garantizando la calidad y la consistencia en las operaciones de fabricación y ensamblaje.

El IRB 1090 se utiliza en una amplia variedad de industrias y aplicaciones, incluyendo automoción, metalurgia, logística, alimentos y bebidas, entre otras. Algunas de las aplicaciones comunes incluyen paletizado, manipulación de materiales, ensamblaje de componentes, soldadura y mecanizado.

Con herramientas de programación intuitivas y una interfaz de usuario amigable, el IRB 1090 se integra fácilmente en sistemas de automatización existentes y se programa rápidamente para adaptarse a requisitos específicos de producción. Esto permite a los fabricantes y operadores maximizar la eficiencia y la productividad en sus operaciones.

En comparación con el IRB140, el IRB1090 presenta una mayor capacidad de carga y un alcance superior, lo que lo hace ideal para manejar objetos pesados y trabajar en áreas más amplias. Mientras el IRB140 es compacto y ágil, adecuado para tareas de alta precisión en espacios reducidos, el IRB1090 es más robusto y se utiliza en aplicaciones que requieren manipulación de grandes componentes y tareas industriales más intensivas.



Figura 3.7. IRB 1090. Extraído de [31]

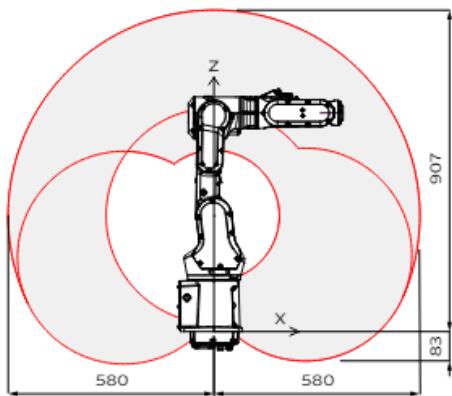


Figura 3.8. Espacio de trabajo de IRB 1090. Extraído de [32]

3.4.4. Schunk LWA 4P

El SCHUNK LWA 4P (Figura 3.9), es un brazo robótico desarrollado por SCHUNK, una de las empresas líderes en tecnología de automatización y robótica. Este brazo robótico es conocido por su alta precisión, flexibilidad y capacidad de adaptarse a diversas aplicaciones industriales, incluyendo ensamblaje, manipulación de materiales y tareas de automatización complejas. Además, es un brazo colaborativo lo que amplía sus entornos de utilización a otros ámbitos [106–109].

Este robot presenta un diseño modular y compacto, ideal para su integración en una variedad de entornos industriales. Su estructura ligera y robusta está diseñada para ofrecer una alta durabilidad y un rendimiento confiable, incluso en condiciones de trabajo exigentes.

Por otro lado, en cuanto a las características técnicas:

3.4. MATERIALES

- **Grados de Libertad:** El LWA 4P cuenta con hasta 7 grados de libertad, lo que le proporciona una gran flexibilidad y capacidad de maniobra. Esto permite al brazo robótico realizar movimientos complejos y precisos en múltiples direcciones.
- **Capacidad de Carga Útil:** Este brazo robótico puede manejar cargas útiles de hasta 10 kg, lo que lo hace adecuado para una amplia gama de aplicaciones, desde la manipulación de pequeñas piezas hasta el ensamblaje de componentes más pesados.
- **Alcance y Movilidad:** Con un alcance de aproximadamente 1 metro, el LWA 4P puede cubrir un área de trabajo considerable, permitiendo la manipulación eficiente de materiales y componentes en diversas configuraciones de producción, por lo que su zona de trabajo es de 70 cm de radio con 330° de giro.
- **Precisión y Repetibilidad:** Equipado con avanzados sistemas de control y servomotores de alta precisión, el LWA 4P ofrece una precisión excepcional y una alta repetibilidad en cada ciclo de trabajo, garantizando la calidad y consistencia en las operaciones industriales.
- **Integración y Programación:** El LWA 4P es compatible con diversos sistemas de control y software de programación, lo que facilita su integración en líneas de producción existentes y su adaptación a diferentes tareas de automatización. Los usuarios pueden programar y controlar el brazo robótico de manera intuitiva utilizando las herramientas de *software* proporcionadas por SCHUNK.

El SCHUNK LWA 4P es utilizado en una amplia variedad de industrias, incluyendo la automoción, la electrónica, la industria farmacéutica, y la fabricación de bienes de consumo. Sus aplicaciones comunes incluyen:

- **Ensamblaje de Componentes:** Gracias a su alta precisión y flexibilidad, el LWA 4P es ideal para el ensamblaje de componentes electrónicos y mecánicos, donde se requiere una manipulación cuidadosa y precisa.
- **Manipulación de Materiales:** Su capacidad para manejar cargas útiles de hasta 10 kg lo hace adecuado para tareas de manipulación de materiales en líneas de producción, almacenes y centros de distribución.
- **Automatización de Procesos:** El LWA 4P puede ser utilizado para automatizar una variedad de procesos industriales, mejorando la eficiencia y reduciendo la necesidad de intervención manual.

El uso de este robot presenta diferentes beneficios, entre los que destacan:

- **Flexibilidad:** Con hasta 7 grados de libertad, el LWA 4P ofrece una gran flexibilidad para realizar movimientos complejos y adaptarse a diversas tareas industriales.

- **Precisión:** La alta precisión y repetibilidad del LWA 4P garantizan la calidad en las operaciones de ensamblaje y manipulación de materiales.
- **Integración Fácil:** La compatibilidad con múltiples sistemas de control y software de programación facilita su integración en líneas de producción existentes.

El SCHUNK LWA 4P ofrece alta precisión y flexibilidad con 7 grados de libertad, lo que lo hace ideal para tareas colaborativas y automatización en espacios reducidos, destacando su facilidad de integración y programación. En comparación, el ABB IRB140 es igualmente preciso y ágil, adecuado para tareas de alta precisión en espacios pequeños, pero no es colaborativo y maneja una carga menor que el IRB1090. El IRB1090, por su parte, supera al SCHUNK en capacidad de carga y alcance, siendo ideal para aplicaciones industriales intensivas que requieren manipulación de grandes componentes, aunque es menos flexible y no está diseñado para espacios reducidos o colaboración directa con humanos como si es el caso de este robot.



Figura 3.9. SCHUNK LWA 4P. Extraído de [33]

3.4. MATERIALES

Capítulo 4

Desarrollo de la aplicación

4.1. Introducción

Como ya se ha introducido en capítulos anteriores, el principal objetivo de este TFG es desarrollar una aplicación inmersiva de RV, dotada de modelos representativos de los brazos robóticos IRB 140, IRB 1090 y SCHUNK LWA 4P. Dicha aplicación debe ofrecer una experiencia interactiva agradable para el usuario, por lo que es necesario que disponga de un correcto sistema de visionado, mecanismos para desplazarnos por la escena y una forma intuitiva de interactuar con los elementos que la componen.

Puesto que el objetivo final se centra en el control de un brazo robótico mediante la utilización de la aplicación desarrollada, será necesario incluir en esta, distintos paneles de control mediante los cuales podremos escoger diferentes métodos de control de los brazos robóticos y la posibilidad de programar puntos, trayectorias, simulación de entradas/salidas digitales y la elaboración de programas básicos.

Además, debe establecerse un protocolo de comunicación con una configuración adecuada, el cual permitirá transmitir la información generada en la aplicación de RV a las aplicaciones de control de los distintos robots, Robotstudio para el caso de los ABB y ROS para el Schunk.

A lo largo de este capítulo, se justificará la elección del motor gráfico, se detallará cada uno de estos componentes esenciales. Se explicará cómo se han implementado los sistemas de visionado y los mecanismos de navegación en la escena y se describirán los distintos métodos de control de los brazos robóticos que se han integrado, incluyendo la programación de puntos, trayectorias, simulación de entradas y salidas digitales, y la creación de programas básicos. Por último, se abordará la configuración y establecimiento del protocolo de comunicación necesario para sincronizar la información entre la aplicación de RV y las plataformas de control de los robots, Robotstudio y ROS y la forma en que estas procesan los datos para transferirlos al robot real.

4.2. Selección motor gráfico

La capacidad de crear estos entornos virtuales, sus características y las capacidades inmersivas que estos ofrecen dependen en gran parte del tipo de motor gráfico elegido y la forma en que este gestiona los distintos recursos. La selección del motor gráfico no solo condiciona las posibilidades de interacción con el entorno simulado, sino aspectos como el nivel de renderizado o el lenguaje de programación y con ello, la dificultad que su utilización conlleva.

Un motor gráfico debe cumplir numerosos requisitos para garantizar una gratificante experiencia inmersiva [110, 111]:

- **Compatibilidad multiplataforma:** Capacidad de conectarse con distintos dispositivos de RV. En este caso, es indispensable es que sea compatible con la tecnología Oculus.
- **Interactividad:** La capacidad de interactuar con el entorno y los objetos virtuales en tiempo real sin generar retardos y permitiendo al usuario llevar a cabo acciones intuitivas y precisas.
- **Propiedades físicas:** Dado que estamos construyendo un entorno virtual que pretende ser una representación fiel de la realidad, es necesario dotar a los objetos virtuales de sus características y comportamientos físicos reales.
- **Calidad Visual:** Los motores gráficos ofrecen la posibilidad de aplicar complejos paquetes de texturas, iluminación y sombreados, renderizándolos complejamente. Cuanto más complejos sean estos paquetes, más realistas será la experiencia, pero esto conlleva un mayor consumo de recursos, lo cual puede generar deficiencias en la transmisión de información o problemas de latencia en la propia aplicación.
- **Entorno de desarrollo:** Cada motor gráfico dispone de su propio sistema de gestión de escenas y *assets* y probablemente utilicen entornos de programación o lenguajes específicos.

Una vez analizados los requisitos que debe cumplir un motor gráfico para trabajar de forma adecuada con la tecnología de realidad virtual y más específicamente con la tecnología Oculus, es posible destacar los dos principales motores gráficos:

- **Unreal Engine 4.1:** Fue desarrollado por la famosa compañía de videojuegos Epic Games 4.2 y ha sido el principal promotor de este tipo de tecnología durante años. La principal característica de este motor gráfico es el potencial en cuanto ha renderizado, ya que es capaz de desarrollar experiencias visuales con un alto grado de realidad, lo que conlleva una mayor complejidad en cuanto ha renderizado y programación, basada en el lenguaje C++. Está destinado principalmente a usuarios experimentados que desarrollan experiencias profesionales de una alta calidad. [112].



Figura 4.1. Unreal Engine. Extraído de [34]



Figura 4.2. Epic Games. Extraído de [35]

- **Unity 4.3:** Unity ha sido el único motor gráfico que ha llegado a competir con Unreal Engine y el secreto de su evolución radica en una interfaz amigable y su accesibilidad, y es que Unity, aunque ofrece una menor capacidad en cuanto ha renderizado y físicas, este dispone de un entorno intuitivo y fácil de usar. Por este motivo, Unity ha sido la plataforma preferida por desarrolladores *amateur*, además este motor gráfico trabaja con Csharp que comparado con C++ es más fácil de utilizar [76,113].



Figura 4.3. Unity. Extraído de [36]

Tras analizar las dos principales opciones y llevar a cabo diferentes pruebas con ellas, no es de extrañar que el motor gráfico seleccionado para el desarrollo de este TFG sea Unity y es que, en este caso, no se prioriza el lograr una experiencia inmersiva lo más fiel a la realidad, sino lograr los objetivos de la forma más optimizada posible. Además, dado que el hecho de trabajar con motores gráficos y con este nivel de programación es una tarea ardua, que no es objeto de estudio en el grado de ingeniería electrónica industrial y automatización, resulta lógico escoger el entorno gráfico que más facilidades de aprendizaje posee y es que su gran comunidad ofrece infinidad de tutoriales, curso y recursos que serán empleados en el desarrollo de este TFG.

4.3. Selección protocolo de comunicación

Para el correcto desarrollo del TFG es necesario establecer un protocolo de comunicación, el cual permitirá enviar en todo momento la información relativa a las posiciones de cada robot desde Unity a la aplicación de destino correspondiente. Además, de la información de posiciones también es necesario enviar mensajes de control de una forma bidireccional. Para comprender los requisitos que debe cumplir dicho protocolo de comunicación, antes es necesario conocer las partes que intervienen en una comunicación [114, 115].

- **Servidor:** Se trata de la aplicación encargada de abrir un canal de comunicación con distintos clientes. El servidor permanece continuamente en estado de escucha, esperando la solicitud de conexión de un cliente, si este acepta la conexión con el cliente establecerá un canal de comunicación con él. Además, el servidor es el encargado de gestionar las medidas de seguridad y garantizar que solo se conecten aquellos clientes autorizados.
- **Cliente:** El cliente es la aplicación solicitante de un servicio, por tanto, es el que inicia la comunicación, mediante una solicitud de conexión al servido. Si el servido acepta la comunicación con el cliente, abrirá un canal de comunicación por el cual el cliente enviará la petición de servicios y el servidor la satisfacerá.
- **Red de comunicación:** La red de comunicación se caracteriza por el medio físico por el cual se lleva a cabo la comunicación, como cables Ethernet, redes WIFI u otros tipos de infraestructura de red. Además del medio físico, la red de comunicación está fuertemente condicionada por el protocolo de transporte que se utilice, entre los cuales encontramos TCP/IP (*Transmission Control Protocol*) y UDP/IP (*User Datagram Protocol*).
- **Código:** Para transmitir el mensaje por el canal de comunicación es necesario que ambas partes, tanto servidor como cliente, compartan un mismo lenguaje con el interpretar los mensajes enviados y recibidos. En nuestro caso usaremos ASCII (*American Standard Code for Information Interchange*)

Al explicar los elementos de una red de comunicación se ha mencionado, que uno de los elementos más determinantes es el protocolo de comunicación empleado, no solo por las características de este, sino porque también determinara la forma en la que se crearán el cliente y servidor. Los protocolos de comunicación más conocidos son:

- **UDP/IP:** Este protocolo de comunicación se caracteriza por la entrega de datos sin conexión, es decir, se envían datos sin comprobar si la otra parte los ha recibido. Este protocolo es muy usado en transmisiones en directo o videojuegos, donde prioriza la mantención una retransmisión rápida y fluida sobre la fiabilidad de disponer de todos los paquetes.

- **TCP/IP:** Este protocolo de comunicación se caracteriza por realizar una conexión fiable donde al enviar un mensaje se intercambian mensajes para comprobar si el paquete ha sido recibido correctamente o no y en caso de que haya existido algún fallo en la retransmisión se producirá un reenvío de paquetes perdidos. El hecho de verificar una correcta transmisión y recepción hace que sea más lento que UDP, pero garantiza una correcta transmisión de todos los datos. Se usa en aquellas aplicaciones donde prioriza la transmisión del mensaje completo a la velocidad, por ejemplo en la transferencia de archivos [115].

Una vez expuestos los distintos elementos que intervienen en la comunicación, se extrapolan estos conocimientos al caso práctico aplicado en este TFG. En cuanto al protocolo de comunicación empleado, en primer lugar, parece lógico usar un protocolo de transmisión UDP para transmitir el movimiento del robot en tiempo real de una forma fluida, sin embargo, este no es el protocolo empleado y es que el programa de Robotstudio no permite las comunicaciones UDP con los controladores de sus robots por razones de seguridad internas. De esta forma, para unificar la transmisión de datos se ha escogido el protocolo TCP/IP como protocolo para transferir la información entre Unity y Robotstudio para el caso de los robots ABB y entre Unity y ROS para el caso de Schunk.

En cuanto a la comunicación cliente-servidor, debe seleccionarse que aplicación actuará como cliente y como servidor. Al igual que ocurre con el protocolo de comunicación, Robotstudio no permite que los controladores físicos de sus robots actúen como clientes, de esta forma deben establecerse como servidores Robotstudio y ROS. El servidor comprobará continuamente el estado de los clientes, los cuales se conectarán cuando así se ordene desde Unity. Además, el elegir un robot u otro determinará el servidor que reciba la conexión. Un esquema general de comunicación puede observarse en la Figura 4.4.

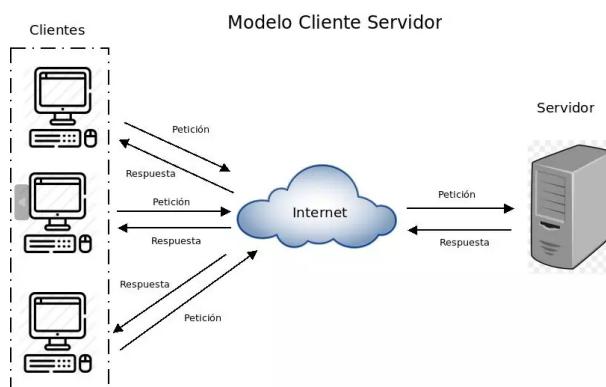


Figura 4.4. Modelo Cliente-Servidor. Extraído de [37]

4.4. Entorno virtual

Dado que el TFG se centra en el desarrollo de una aplicación de RV, disponer de un entorno simulado lo más real posible es indispensable. En este contexto, la escena desarrollada representa una nave industrial o taller con un suelo de hormigón y paredes compuestas por paneles del mismo material. En la escena se diferencian claramente cuatro estaciones de trabajo distintas, las cuales se observan en la siguiente Figura 4.5:

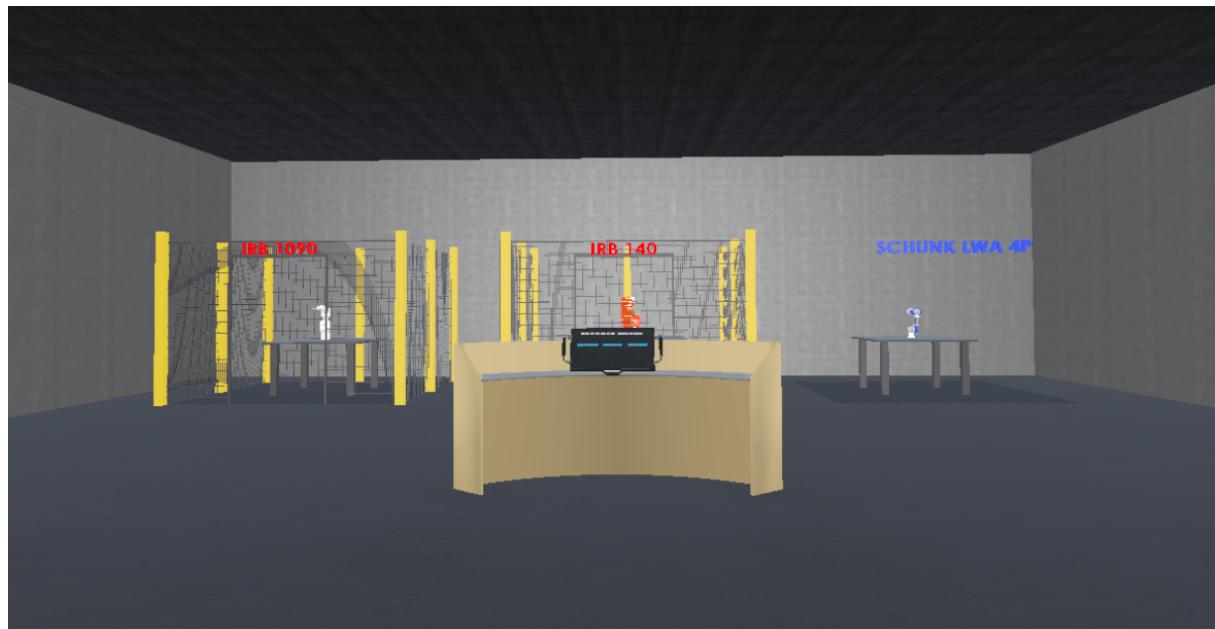


Figura 4.5. Vista general de la escena

4.4.1. Primera estación

La primera estación (Figuras 4.6 y 4.7) es un entorno de reposo, ya que en ella no se puede interactuar con ningún robot. Es el punto inicial de la escena y donde el avatar debe volver tras finalizar la interacción con los robots. Esta estación consta de un atril con el logo de la UAL y, sobre este, un panel principal con distintas opciones para elegir el robot a manipular y un pequeño panel de tutorial.

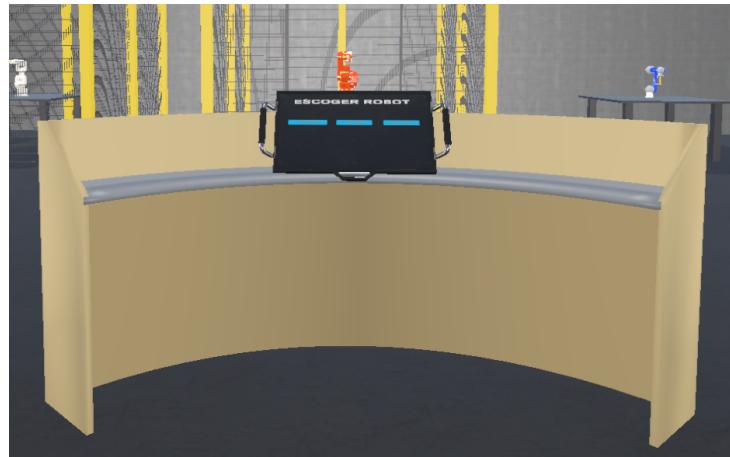


Figura 4.6. Vista de la primera estación (Parte trasera del atril)



Figura 4.7. Vista de la primera estación (Parte delantera del atril con logo de la UAL)

4.4.2. Segunda estación

La segunda estación (Figuras 4.8 y 4.9), se localiza a la izquierda del punto de inicio de la escena y corresponde al entorno de trabajo del IRB 1090. En ella se encuentra un modelo realista del robot en cuestión, ubicado sobre una mesa de dimensiones 3x3 metros a una altura de 90 cm del suelo. Esta estación está rodeada por una valla de seguridad de 5x5 metros, dejando espacio suficiente entre la valla y la mesa para interactuar con el robot. La valla tiene postes diferenciados de color amarillo y un suelo antideslizante de seguridad. Para identificar correctamente el robot en cada estación, sobre la entrada hay un cartel llamativo identificativo. En este caso, el nombre "IRB 1090" se encuentra en color rojo.

4.4. ENTORNO VIRTUAL

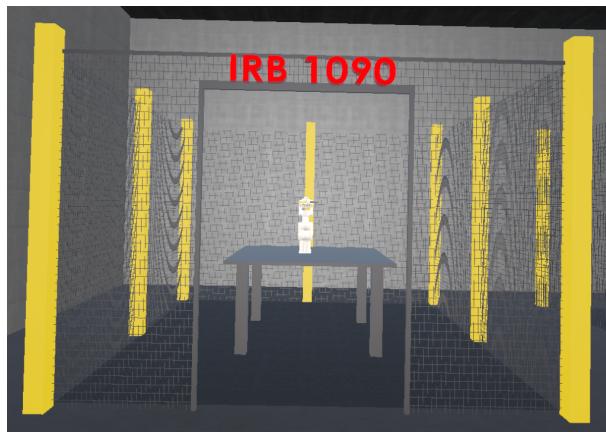


Figura 4.8. Vista general, segunda estación

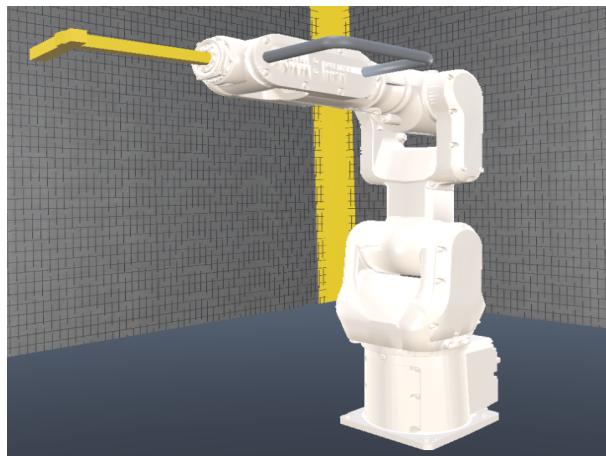


Figura 4.9. Vista IRB1090

4.4.3. Tercera estación

La tercera estación (Figuras 4.10 y 4.11), es la estación central y constituye el entorno de trabajo del IRB 140. En ella se encuentra un modelo realista de dicho robot, representado en su característico color naranja. El entorno donde se localiza es idéntico al del IRB 1090: una mesa de 3x3 metros a 90 cm del suelo con una valla de seguridad. Sobre la valla, se encuentra el cartel “IRB 140” en color rojo.

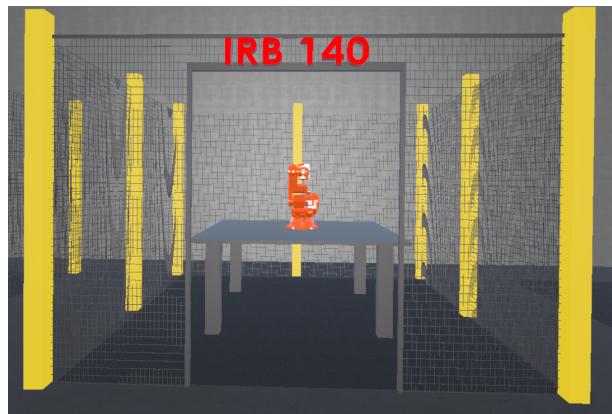


Figura 4.10. Vista general, tercera estación

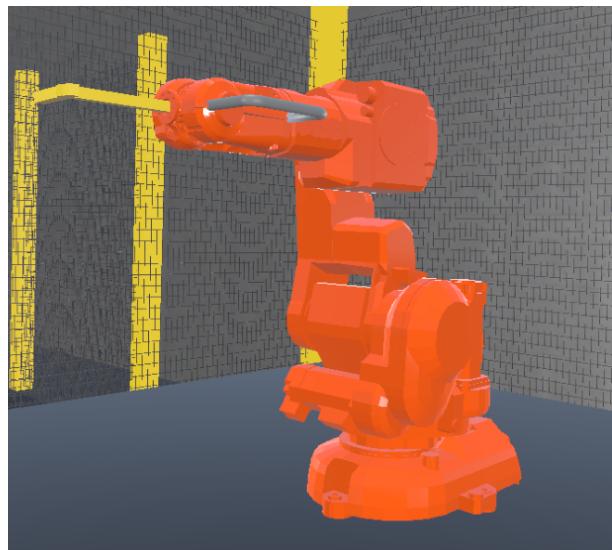


Figura 4.11. Vista IRB140

4.4.4. Cuarta estación

La cuarta y última estación (Figuras 4.12 y 4.13) corresponde al robot SCHUNK LWA 4P, que no necesita verja de seguridad. En este caso, el robot se localiza sobre una mesa de 2x2 metros a 90 cm del suelo de seguridad antideslizante. Sobre la estación hay un cartel identificativo con el nombre “SCHUNK LWA 4P” en color azul.

4.4. ENTORNO VIRTUAL



Figura 4.12. Vista general, cuarta estación



Figura 4.13. Vista SCHUNK LWA 4P

4.5. Capacidades del avatar virtual

Como ha sido mencionado anteriormente, uno de los principales requisitos consiste en dotar a la aplicación de unas capacidades de interacción y movilidad agradables para el usuario.

La tecnología Oculus y su incorporación en Unity permite llevar a cabo una serie de movimientos predeterminados. Estos movimientos son los propios del individuo, y es que los sensores de las gafas captan el movimiento del usuario por su entorno y trasladan tanto el movimiento de desplazamiento como los movimientos de giro de la cabeza a Unity. Además, lleva a cabo un seguimiento de los mandos que se utilizan y permiten visualizarlos dentro de la escena, pero estos no disponen de ninguna configuración preestablecida.

Para facilitar la interacción con los objetos de la escena se han sustituido las imágenes asociadas a los controladores que vienen preestablecidas, por una representación de unas manos humanas (Figuras 4.14 y 4.15). Además, estas manos proyectan un rayo que permitirá realizar determinadas acciones a distancia.

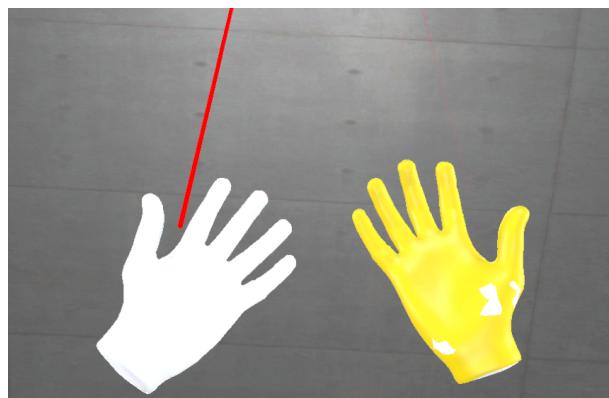


Figura 4.14. Representación de las manos humanas relajadas

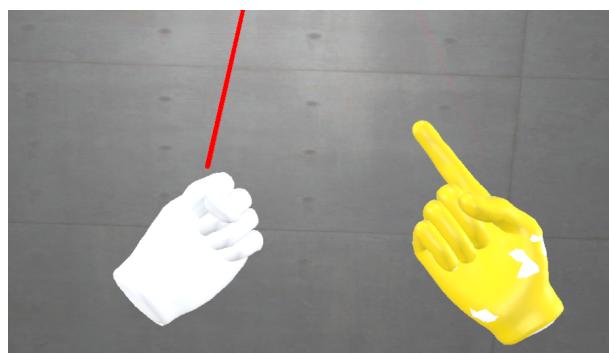


Figura 4.15. Representación de las manos humanas accionadas

Considerando que para una aplicación de estas características, las limitadas posibilidades de desplazamiento y las nulas capacidades de interacción son insuficientes, es necesario configurar

4.5. CAPACIDADES DEL AVATAR VIRTUAL

los controladores de Unity para dotar al avatar de distintas capacidades. Estas capacidades serán descritas a continuación, pero se pueden observar en la Figura 4.16.

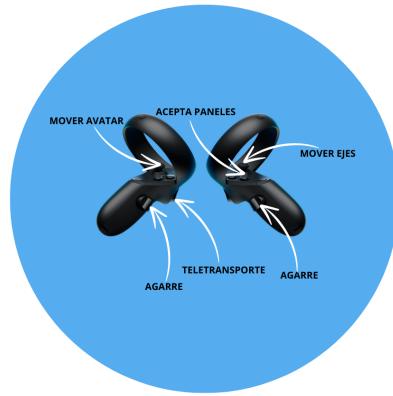


Figura 4.16. Función de cada botón del mando

El primer requisito en cuanto a movilidad consiste en permitir al usuario desplazarse libremente por la escena de Unity sin necesidad de llevar a cabo este desplazamiento en la realidad. Para ello, mediante un sistema de gestión de entradas, Unity reconoce cuando el individuo interactúa con el *joystick* del controlador izquierdo y se programa de forma sencilla un avance a velocidad constante de 0.6 m/s en dirección del *joystick*.

Continuando con las posibilidades en cuanto a desplazamiento, se ha incluido una segunda forma de desplazarse por la escena. Este es el teletransporte. Y es que como se ha mencionado anteriormente, las manos proyectan un rayo, como se muestra en la Figura 4.17, el rayo de la mano izquierda será de color rojo cuando apuntamos a un objeto o pared, pero se observa que al apuntar al suelo cambia a de color y aparece un disco en el punto de intersección del rayo con el suelo. El disco en cuestión representa la posición de teletransporte y para aceptar esta orden solo será necesario pulsar el botón *Drop* del controlador izquierdo. De esta forma, apuntando a un determinado lugar de la escena, el avatar puede teletransportarse de forma inmediata a dicho lugar.

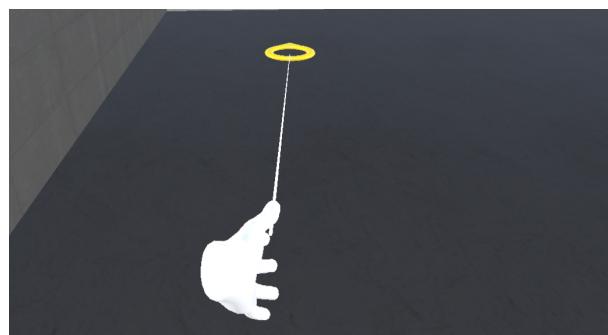


Figura 4.17. Proyección de un rayo con disco de teletransporte

En cuanto a las posibilidades de interacción con los objetos, ambas manos disponen de un mecanismo de agarre, el cual se activa en la mano derecha al pulsar el botón *Trigger* del controlador derecho y en la mano izquierda al pulsar *Trigger* del controlador izquierdo. Dicho mecanismo de agarre se puede observar en la Figura 4.18 y consiste en cerrar el puño, lo que permite al individuo agarrar los objetos previamente configurados para ello. Los objetos que permiten el agarre son las pantallas, que disponen de dos asas laterales para desplazarlas y colocarlas en una posición cómoda para el individuo y los robots que disponen de una serie de asas que se mostrarán u ocultarán en función del robot y tipo de movimiento que se encuentre accionado (Figura 4.19).



Figura 4.18. Mecanismo de agarre pantallas

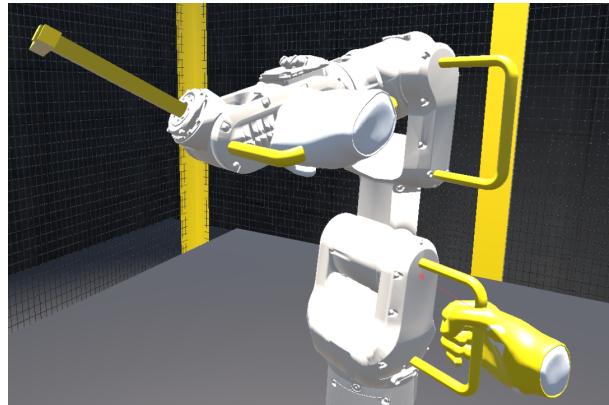


Figura 4.19. Mecanismo de agarre al robot

4.6. PANEL PRINCIPAL

A la hora de llevar a cabo la interacción con las distintas pantallas (Figura 4.20), para desplazarse por los menús y tener acceso a las configuraciones es necesario emplear el *joystick* derecho. Y es que el rayo que este proyecta se tornara de color verde cuando se localice sobre alguno de los botones interactivos y para presionar este botón a distancia bastará con presionar el botón A del controlador derecho. Otra forma de interactuar con los paneles es de una forma táctil y es que como podemos observar al presionar el botón A la mano adopta una posición en la que todos los dedos excepto el índice están recogidos. En esta posición, si se acerca el dedo al panel y se presiona el botón, este se accionará de una forma similar a la que se realizaría en la vida real.

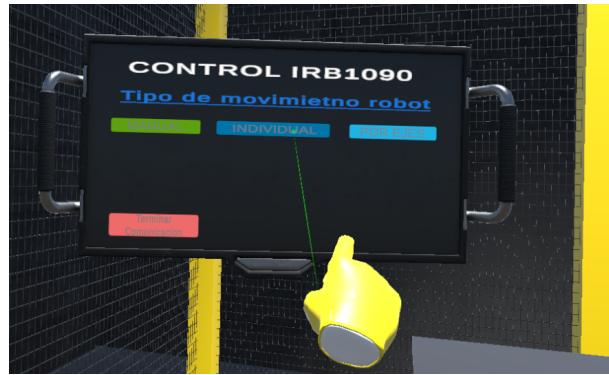


Figura 4.20. Interacción con la pantalla

Los códigos correspondientes a la programación de cada una de estas acciones se pueden encontrar en el GitHub. [45].

4.6. Panel principal

El Panel Principal es el panel del área de reposo y se encuentra inclinado 30 grados sobre el atril. Sus funcionalidades son básicas, permite seleccionar el robot que se desea controlar y acceder a un tutorial donde conocer las funcionalidades de cada botón. Dicho panel presenta el aspecto que se observa en la Figura 4.21.

Al pulsar el botón correspondiente a cada uno de los robots, el entorno virtual cambia su iluminación, centrando la atención en la estación de reposo donde se encuentra el individuo, y la estación correspondiente al robot seleccionado, como se puede observar en las Figuras 4.22, 4.23 y 4.24. Para ello, la iluminación de la escena disminuye y aparecen dos focos, uno para la estación seleccionada y otro para la estación de reposo. Entre ambas estaciones aparece un camino diferenciado y delimitado por elementos luminosos. Sobre la estación seleccionada, el cartel del robot se ilumina resaltando el robot elegido. Además, dentro de la estación de trabajo aparecerán dos pantallas en sus posiciones predeterminadas, una de ellas está destinada a controlar el movimiento del robot y la otra controla las opciones de programación/simulación.



Figura 4.21. Panel principal

La localización y funcionalidades de dichas pantallas serán analizadas posteriormente.

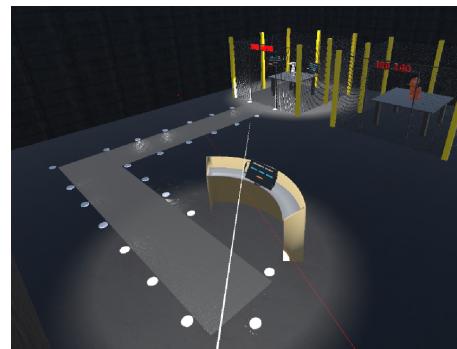


Figura 4.22. Entorno iluminado, para IRB 1090

4.7. PANEL MOVIMIENTO DEL ROBOT

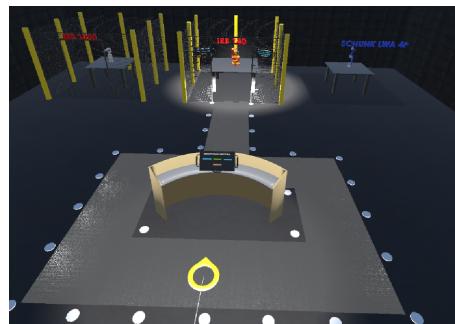


Figura 4.23. Entorno iluminado, para IRB 140

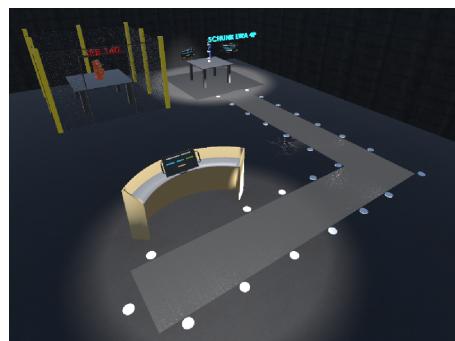


Figura 4.24. Entorno iluminado, para SCHUNK LWA 4P

En cuanto al botón de tutorial, este simplemente muestra frente al individuo un panel donde se representan los controladores junto con las funcionalidades de cada uno de ellos (Figura 4.25).

4.7. Panel movimiento del robot

El principal potencial de esta aplicación radica en la forma en la cual se puede interactuar con los robots. Para permitir al individuo, que utiliza la aplicación, interactuar de una forma intuitiva, se incorporan tres tipos de movimiento para cada uno de los robots.

Para seleccionar el movimiento deseado se utilizará el panel de movimiento del robot, cuyo aspecto se puede ver en la Figura 4.26.



Figura 4.25. Tutorial de controladores junto con funcionalidades de botones



Figura 4.26. Panel del movimiento del robot

En los procesos de desplazamiento se emplea, el ya explicado CCD, donde a partir de la posición objetivo del extremo del robot, que en la escena de Unity se puede localizar como *IK Effector*, permite conocer la posición objetivo de cada articulación.

El proceso de movimiento de los robots se basa en la utilización del MCI, donde se obtienen los ángulos de giro de cada uno de los ejes, que son necesarios para alcanzar la posición objetivo del extremo. De esta forma, cuando el usuario desplaza el extremo del robot o el extremo de una articulación a un punto específico, el MCI actúa de forma inmediata, permitiendo obtener el ángulo objetivo para cada eje. Una vez se dispone del ángulo objetivo se aplica un desplazamiento, que utiliza una interpolación para suavizar el movimiento.

En el desplazamiento del robot el MCI se lleva a cabo continuamente para todas las articulaciones, para evitar que la excesiva densidad de cálculos genere retardos, se sustituye por el método MCD, el cual se ha explicado en capítulos anteriores.

Los códigos correspondientes a la programación de cada una de estas acciones se pueden

4.7. PANEL MOVIMIENTO DEL ROBOT

encontrar en el GitHub donde se recoge la información de interés que complementa esta memoria. [45].

4.7.1. Movimiento manual

El primer tipo de movimiento (Figura 4.27), el movimiento “MANUAL”, permite posicionar el robot en el punto deseado a partir de la manipulación única de su extremo. Al seleccionar este tipo de movimiento aparecerá un agarre en el extremo del robot, de color gris, el cual permitirá manipular el robot. Al coger dicho agarre se podrá desplazar el extremo del robot al punto deseado, ya que todos los eslabones del robot se posicionarán automáticamente y en tiempo real en la configuración adecuada para alcanzar dicho punto.

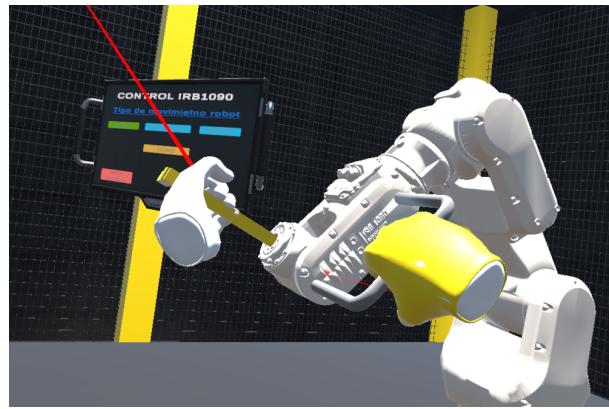


Figura 4.27. Robot en movimiento manual

Al seleccionar este tipo de movimiento se observa que el robot también dispone de un elemento terminal, representado con una barra de color amarillo, el manipular este elemento permite modificar la inclinación del elemento terminal. Además, al actuar sobre el *joystick* del controlador derecho puede modificarse la rotación del elemento terminal.

4.7.2. Movimiento individual

El seleccionar el movimiento de tipo “INDIVIDUAL” (Figura 4.28), ofrece una interacción similar a la del movimiento manual, pero en este caso el control del robot se lleva a cabo mediante la manipulación individual de cada uno de los eslabones. En este caso aparecerán tres agarres de color amarillo con los que el individuo puede modificar la posición de cada uno de ellos, variando así la posición global del robot.

Este tipo de movimiento permite manipular el elemento terminal del robot, modificando su inclinación y rotación de igual forma que en el caso del movimiento manual.

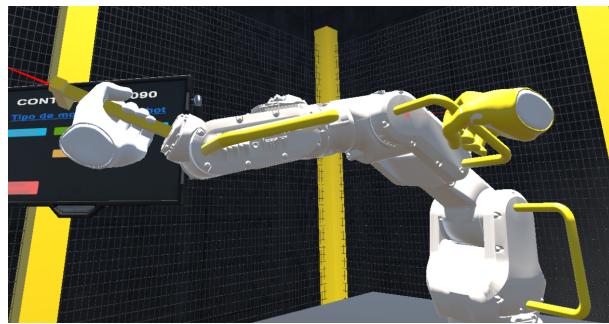


Figura 4.28. Robot en movimiento individual

4.7.3. Movimiento por ejes

El movimiento por ejes es uno de los movimientos más típicos en la premoción manual de robots. Se basa en la utilización de un sistema de joystick o botoneras, el cual permite seleccionar un eje en específico y variar su posición. Este mismo sistema se aplica en Unity mediante un panel con 6 botones numerados, los cuales se corresponden cada uno de los ejes del robot. Al pulsar sobre un botón se seleccionará su correspondiente eje y mediante el movimiento del joystick del controlador derecho puede moverse dicho eje, en sentido horario o antihorario.

El panel de movimiento por ejes incluye además un *slider* por cada uno de ellos. Al posicionarse sobre el *slider* este adquiere en el valor actual de su eje y al seleccionarlo, con el botón de interacción A, permite su desplazamiento, modificando así la posición del robot.

Todos estos aspectos se pueden observar en la Figura 4.29.

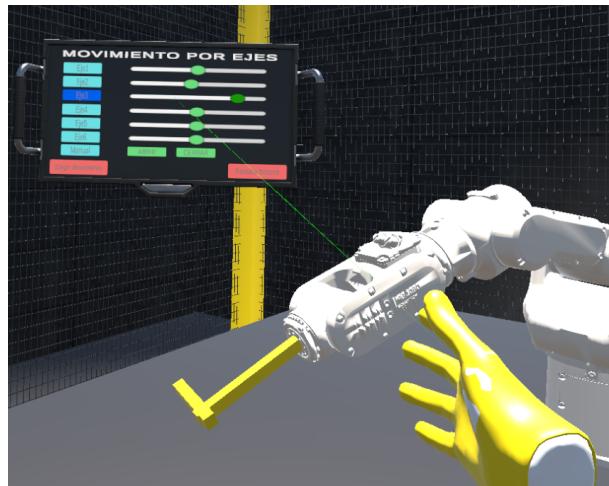


Figura 4.29. Movimiento por ejes

4.8. PANEL DE PROGRAMACIÓN/SIMULACIÓN

4.7.4. Movimiento Home

Como movimiento extra se dispone de un botón que permite posicionar el robot en su posición de Home de una forma sencilla y a velocidad reducida.

4.8. Panel de programación/simulación

El panel de programación proporciona diversas opciones para configurar tareas en el robot, permitiendo almacenar puntos, trayectorias y programas básicos, como se detallará a continuación. Es importante mencionar que toda la información generada durante la simulación puede guardarse para su posterior uso.

Cada robot tiene un sistema de archivos que se carga al iniciar el programa. Estos archivos contienen los diferentes puntos, trayectorias y programas almacenados. Para guardar los datos generados mientras se usa la aplicación, el usuario debe presionar el botón de guardar datos, disponible en el panel de programación, como se muestra en la Figura 4.30. Guardar los datos manualmente permite al usuario tener copias de seguridad de sus programas. Los archivos de almacenamiento se ubican en la carpeta de Unity y se identifican con el nombre de la información que contienen (puntos, trayectorias o programas) y el robot al que pertenecen.

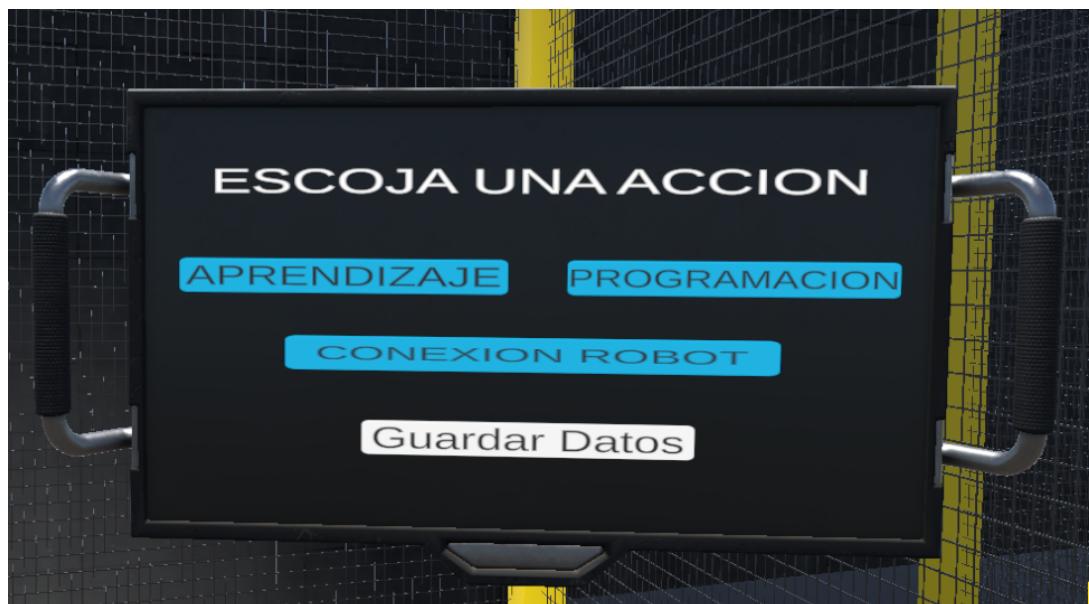


Figura 4.30. Panel escoger acción

Este panel consta de tres secciones. La primera de ellas es la sección de aprendizaje, en la cual pueden generarse puntos y trayectorias, la segunda de ellas es la sección de programación que permite al usuario confeccionar programas básicos y la última de las secciones constituye el entorno de simulación que permite conectar el robot virtual con el real.

4.8.1. Panel Aprendizaje

Al accionar el panel de aprendizaje, la pantalla mostrará las dos acciones posibles, el programar puntos o trayectorias.

- **Editor de Puntos:** El editor, que se observa en la Figura 4.31, dispone de una lista de puntos en la cual inicialmente aparecerán, un punto predeterminado, "home", y todos aquellos puntos guardados en sesiones previas. Para editar la lista de puntos, se dispone de dos botones que permiten añadir un nuevo punto a la lista y un botón "Borrar" que permite eliminar el punto seleccionado en la lista. Al añadir un punto a la lista estos lo hacen en orden numérico, es decir, si el último punto disponible es el Punto3, el nuevo será Punto4. De igual forma, al eliminar un punto de la lista, el resto de puntos posteriores se renombra para respetar el orden numérico, es decir, el Punto 4 pasa a llamarse Punto 3.

Al generar un punto en la lista, este tendrá como coordenadas predeterminadas las coordenadas de "home", es decir, todos los ángulos en cero. Para asignarle el valor deseado, se utilizará el panel de movimiento para posicionar el robot y, una vez colocado en el punto objetivo, se presiona el botón "Grabar".

Para comprobar que el punto se ha almacenado de forma correcta, este panel dispone de un botón de "Mover a Punto", el cual desplaza el robot al punto seleccionado en un tiempo preestablecido (dos segundos). El tiempo que tarda el robot en realizar dicho desplazamiento se puede modificar introduciendo un nuevo valor mediante el panel numérico que aparece sobre la pantalla al presionar el cuadro de "Enter Time", como se puede observar en la Figura 4.32.

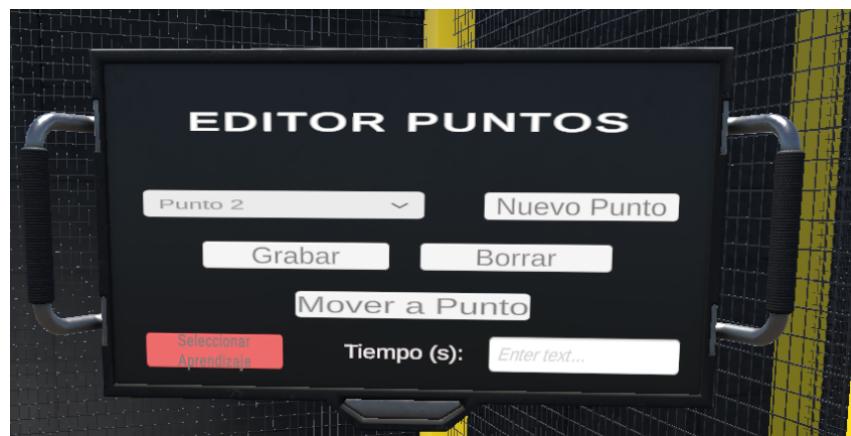


Figura 4.31. Editor de puntos

4.8. PANEL DE PROGRAMACIÓN/SIMULACIÓN

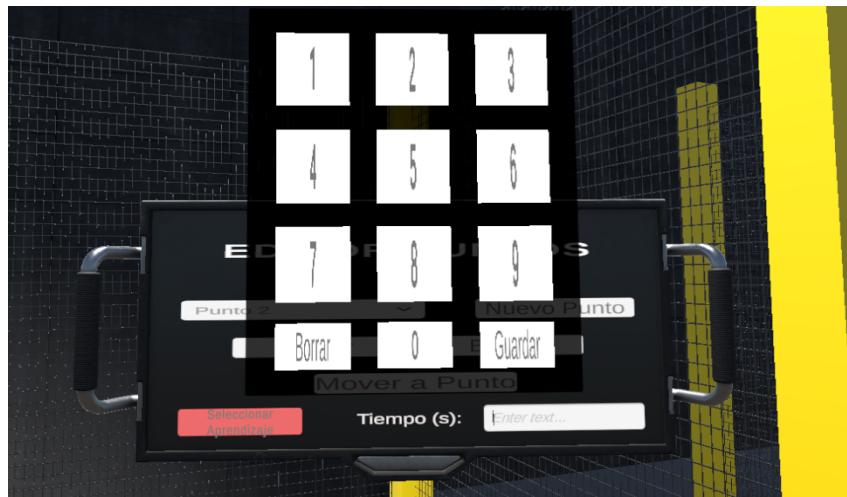


Figura 4.32. Teclado del editor de puntos

- **Editor de Trayectorias:** El editor de trayectorias de la Figura 4.33 presenta un aspecto similar al editor de puntos; en él, se dispone de una lista de trayectorias, un botón para añadir una nueva trayectoria y un botón para borrar la trayectoria seleccionada, de igual forma que en el editor de puntos.

Para almacenar una trayectoria, se posiciona el robot en un punto inicial y, en el momento deseado, se presiona el botón “P.Inicial”, iniciando así el recorrido que continuará hasta que el usuario accione el botón ”P.Final”. Durante el transcurso de la trayectoria se almacenarán tanto las posiciones por las que pasa el robot como los tiempos entre puntos, de esta forma si el robot realiza una parada durante la trayectoria, esta quedará registrada.

De igual forma que en el editor de puntos, se dispone de una interfaz que permite simular la trayectoria seleccionada. Para ello, al pulsar el botón “Mover a Punto”, el robot se posiciona en la posición de inicio de la trayectoria. Una vez posicionado, si se selecciona el botón “Iniciar Trayectoria”, ejecutará el recorrido de forma idéntica a la almacenada. En caso de querer variar la velocidad de la trayectoria, se dispone de un control de velocidad porcentual, que permitirá aplicar un porcentaje de entre 10 % y el 1000 % sobre la velocidad original.



Figura 4.33. Editor de trayectorias

4.8.2. Panel Programación

La sección de programación permite crear un programa con un número ilimitado de condiciones, en las cuales se pueden incluir acciones como moverse a una posición, ejecutar una trayectoria o activar una salida digital. Este panel se puede observar en la Figura 4.34 y su funcionamiento se explica a continuación.

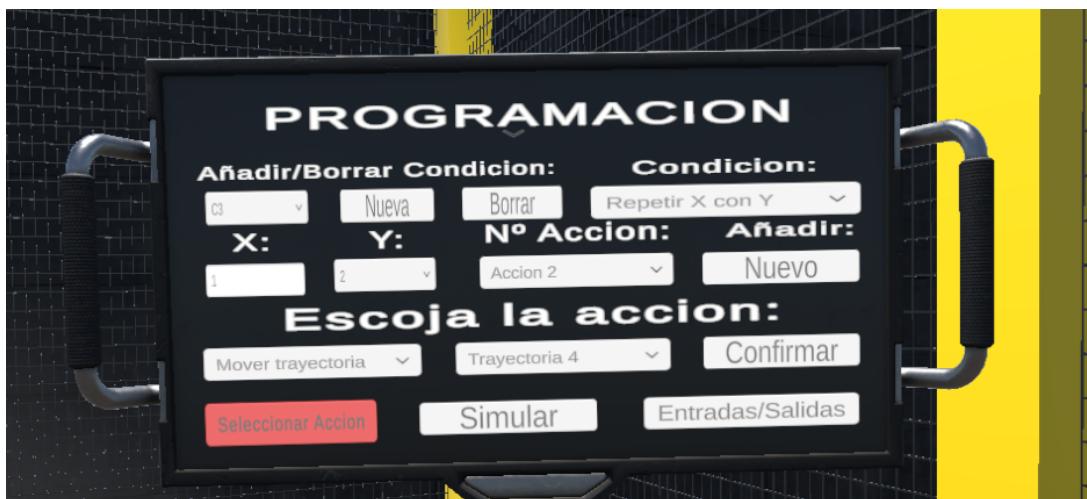


Figura 4.34. Editor de trayectorias

- **Añadir/Borrar Condiciones:** Este panel permite añadir un número arbitrario de con-

4.8. PANEL DE PROGRAMACIÓN/SIMULACIÓN

diciones al programa. Para gestionar dichas condiciones, se dispone de una lista numerada con las condiciones, representadas como C1, C2, C3, etc., la cual puede editarse añadiendo o eliminando condiciones.

Al generar una condición, debe seleccionarse el tipo de condición mediante el menú desplegable del panel. Los tipos disponibles son:

- **Repetir X veces:** Las acciones asociadas a esta condición se ejecutarán en bucle un número X de veces cuando se inicie el programa.
 - **Repetir X veces cuando Y:** Similar a la condición anterior, pero las acciones se ejecutarán un número X de veces una vez que la entrada digital Y se haya activado.
 - **Repetir X veces cuando no Y:** Contraria a la condición anterior, las acciones se iniciarán cuando la entrada Y esté desactivada.
-
- **Añadir valores X e Y:** Las condiciones vienen dadas por el valor X, número de repeticiones, y el valor Y, número de entrada a comprobar. Para que dicha condición funcione de una forma adecuada es indispensable hacer uso de los paneles X e Y para añadir sus correspondientes valores mediante el uso del teclado virtual.
 - **Acciones:** Dentro de cada condición, se puede ejecutar un número indefinido de acciones. Al igual que con las condiciones, se dispone de un desplegable para listar las acciones, y mediante el botón "Nuevo" se permitirá añadir acciones. Las acciones pueden ser de tres tipos:
 - **Mover a Punto:** Al seleccionar este tipo de acción, aparecerá la lista de puntos disponibles en el desplegable de su derecha.
 - **Mover Trayectoria:** Al seleccionar esta acción, se muestra la lista de trayectorias, de donde se escogerá la deseada.
 - **Activar Salida:** Esta acción permite activar la salida digital especificada, en el controlador del robot, si está disponible. Para ello se han creado cinco salidas digitales para cada robot.

Una vez generado el programa, puede visualizarse un esquema mediante el botón de "Confirmar", como en el ejemplo de la Figura 4.35. Además, se puede simular el programa pulsando el botón "Simular". Tras la simulación, el usuario tendrá acceso al panel de gestión de entradas y salidas, el cual está disponible mediante "Entrada/Salidas", como se puede observar en la Figura 4.37, para así avanzar en la ejecución del programa.

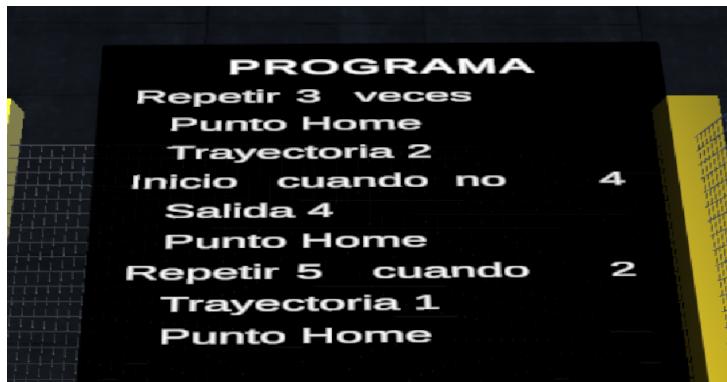


Figura 4.35. Ejemplo Programa



Figura 4.36. Gestor entradas/salidas

4.9. Panel de comunicación

Este último panel está orientado al control del robot, es decir, no se pueden generar nuevas trayectorias, puntos o programas, sino que permite simular los ya existentes.

Al seleccionar la opción “Conexión con el Robot”, se mostrará un desplegable con distintas opciones y un botón que permite establecer la conexión con el robot real. Las opciones disponibles son:

- **Conexión en Directo:** Esta opción permite retransmitir en tiempo real el movimiento del robot virtual al robot real, mediante el panel de la Figura 4.37. Para ello, se seleccionará uno de los tipos de movimiento del robot y se actuará sobre el modelo virtual del robot seleccionado. El programa desarrollado captará estos movimientos con una frecuencia de 0.4 segundos y los enviará de forma continua al controlador del robot real o a la aplicación de simulación de RobotStudio.
- **Puntos:** En este caso se dispondrá de un desplegable en el que seleccionar el punto al que se quiere desplazar el robot. Al seleccionar la comunicación con el robot real, aparecerá

4.9. PANEL DE COMUNICACIÓN



Figura 4.37. Conexión en directo

una nueva opción llamada "Limitar velocidad", la cual se encuentra seleccionada de forma predeterminada, pero que podrá deseleccionarse. Esta opción permite establecer un límite máximo de velocidad de 0.2 m/s para los robots ABB y una velocidad máxima de 2 m/s para el Schunk; en caso de no seleccionarla, no existirá ningún límite de velocidad, salvo los impuestos por el propio controlador del robot. De esta forma, el desplazamiento simulado se realizará en el mismo periodo de tiempo que el desplazamiento en el robot real. El panel de transmisión de puntos puede observarse en la Figura 4.38



Figura 4.38. Simulación de puntos

- **Trayectorias:** Al igual que en el menú de puntos, se dispone de un desplegable donde seleccionar la trayectoria, junto con los botones de "mover a punto" y "mover trayectoria", los cuales ejecutarán la trayectoria deseada tanto en simulación como en el robot real. Al seleccionar la comunicación con el robot real, aparecerá de nuevo preseleccionado el "Límite de velocidad", que establece las mismas restricciones que en el desplazamiento a puntos 4.39.



Figura 4.39. Simulación trayectorias

- **Programas:** Este panel permite visualizar el programa generado, gestionar las entradas y salidas, y simular el programa o transferirlo al robot real. También se dispondrá del límite de velocidad preestablecido. Para llevar a cabo la simulación de programas se utiliza el panel disponible en la Figura 4.40.



Figura 4.40. Simulación programas

4.10. Comunicación Unity

Una vez presentado el entorno de simulación, es crucial entender cómo Unity gestiona la comunicación con robots y aplicaciones intermedias como ROS y RobotStudio. Como se ha mencionado previamente, la comunicación entre aplicaciones utiliza el protocolo TCP/IP, que establece normas para interconectar sistemas en redes, garantizando la transferencia fiable de datos. Aunque la búsqueda de una transmisión fluida en tiempo real podría sugerir UDP/IP como el protocolo más adecuado, no es aplicable debido a que RobotStudio y los robots ABB no soportan este protocolo de comunicación por razones de seguridad.

La aplicación desarrollada en Unity cuenta con un panel de conexión con el robot, permitiendo transferir información al destino establecido mediante tres pasos:

- **Creación del Cliente:** Unity actúa como cliente. Para establecer una conexión cliente-servidor, el servidor debe estar disponible cuando el cliente solicita la conexión. Esta configuración permite a Unity activar o desactivar la conexión según sea necesario. De lo contrario, el cliente tendría que enviar continuamente mensajes mientras espera la conexión del servidor, lo cual podría saturar la red y afectar aplicaciones como RobotStudio. La función de cliente de Unity facilita el control y gestión de la conexión desde la aplicación de simulación.
- **Establecimiento de la Conexión:** Una vez configurado el cliente (Unity), se procede a establecer la conexión con el servidor (por ejemplo, un robot o una aplicación como ROS o RobotStudio). Esto implica enviar una solicitud de conexión al servidor y esperar una respuesta que confirme la conexión exitosa. Durante esta fase inicial, el cliente saluda al servidor con "*Hello, server*". Si el servidor recibe correctamente este mensaje, responde con "*Hello*"; de lo contrario, responderá con "Error". Estos intercambios iniciales verifican la correcta conexión antes de la transferencia de información.
- **Transferencia de Datos:** Con la conexión establecida, se inicia la transferencia de datos entre Unity y el servidor. Esta transferencia incluye comandos para informar al robot sobre la acción a realizar, ya sea una transmisión en directo, un movimiento a un punto, una trayectoria o un programa. La comunicación es bidireccional: cada vez que Unity envía un mensaje (ya sea de control o coordenadas), RobotStudio lo procesa y envía una respuesta. Unity espera estas respuestas para continuar con la ejecución del código, asegurando la sincronización entre cliente y servidor. Además de la información de control, el servidor envía datos adicionales necesarios para calcular velocidades, como la distancia entre puntos. Después de completar una acción específica, como mover a un punto, la conexión no se cierra; el servidor permanece activo y a la espera de nuevos mensajes de Unity.

La gestión de la comunicación en Unity se realiza mediante el uso de *sockets*, que son interfaces de programación que permiten enviar y recibir datos a través de una red. Para manejar la creación y uso de *sockets* y canales de comunicación, se utilizan principalmente las librerías de *Csharp System.Net*. Estas librerías permiten crear *sockets* y canales a partir de una dirección IP y un puerto especificados, correspondientes a la IP del servidor y a los puertos 10000 para IRB140, 11000 para IRB1090 y 12000 para Schunk, facilitando así el intercambio de información. Un breve ejemplo de cómo establecer este tipo de comunicación puede observarse en el siguiente ejemplo Tabla 4.1.

```
// EJEMPLO COMUNICACION C# //

// Incorporacion librerias
using System;
using UnityEngine;
using System.Net.Sockets;

// Definir clase para Unity
public class Ejemplo : MonoBehaviour
{
    public Class1()
    {
        // Definir el cliente y el canal en Ip y Puerto especificos
        TcpClient tcpClient = new TcpClient(IP, Port);
        NetworkStream tcpStream = tcpClient.GetStream();

        // Enviar mensajes al servidor
        string message = ''Hello, server'';
        byte[] data = Encoding.UTF8.GetBytes(message);
        tcpStream.Write(data, 0, data.Length);

        // Recibir mensajes
        byte[] buffer = new byte[1024];
        int bytesRead = tcpStream.Read(buffer, 0, buffer.Length);
        string response = Encoding.UTF8.GetString(buffer, 0, bytesRead);

        // Cerrar el cliente y el canal
        tcpClient.Close();
        tcpStream.Close();
    }
}
```

Tabla 4.1. Código 1: Ejemplo de comunicación en C#

Una vez establecida la comunicación por parte del cliente de Unity, se procederá a la transmisión de información por parte de Unity. Esta información se constituye de dos tipos de mensajes, por un lado, los mensajes de control que informan al servidor sobre la información que se va a transferir y, por otro lado, los mensajes de transmisión de datos, entre los cuales pueden encontrarse puntos, distancias entre puntos y velocidades. La información enviada y recibida dependerá del tipo de robots que se está manipulado, por una parte, se encuentra el IRB 140 junto con el IRB 1090 donde el servidor es gestionado por Robotstudio y, por otra parte, el Schunk LWA 4P donde el servidor se encuentra en ROS.

4.11. Comunicación RobotStudio y Conexión del robot real

Tanto en los controladores digitales creados en RobotStudio como en los controladores físicos de los robots ABB, el tipo de robot condiciona la comunicación. En este contexto, no se pueden utilizar libremente las bibliotecas estándar como en Unity mediante C#; en su lugar, se requieren módulos específicos que deben ser adquiridos y descargados en la interfaz del controlador. Por ejemplo, para el IRB 140, es imprescindible el módulo *PC-Interface*, que se corresponde con la librería *System.Net.Socket* en *Csharp*. En cambio, para el IRB 1090, no se necesita esta librería, ya que viene preconfigurado con los módulos necesarios para estas comunicaciones.

Como se mencionó anteriormente, el controlador de ABB actúa como servidor. Su tarea inicial es generar un *sockets* y permanecer a la escucha en un puerto específico y una dirección IP determinada. El controlador ABB se inicia antes que Unity y espera a que se establezca la conexión. Cuando recibe un mensaje, verifica el puerto y la dirección IP del cliente. Si coinciden con los valores especificados, acepta la conexión.

Una vez aceptada la conexión por parte del cliente, se realiza un intercambio de mensajes de sincronización. Después de este intercambio, el servidor vuelve a estar en espera de nuevas órdenes. Un ejemplo de este procedimiento de gestión de la comunicación por parte de robtstudio se puede observar en la tabla Tabla 4.2

```

!! EJEMPLO COMUNICACION RAPID !!

!Definir variables
VAR socketdev serverSocket;
VAR socketdev clientSocket;
VAR string Ip;
VAR intnum TcpPort;
VAR string dataTcp;

!Definir modulo principal
MODULE main
PROC main()
    !Crear ya abrir servidor en Ip y puerto
    SocketCreate serverSocket;
    SocketBind serverSocket, Ip, TcpPort;

    !Esperar y aceptar conexion cliente
    SocketListen serverSocket;
    SocketAccept serverSocket, clientSocket, \Time:=WAIT_MAX;

    !Enviar mensaje
    control :='ok';
    SocketSend clientSocket, \Str:=control;

    !Recibir mensaje
    SocketReceive clientSocket, \Str:=dataTcp;

    !Cerrar conexion
    SocketClose clientSocket;
    SocketClose serverSocket;

ENDPROC
ENDMODULE

```

Tabla 4.2. Código 2: Ejemplo de comunicación en Rapid

Para que un cliente pueda conectarse al controlador de ABB, es suficiente con que tanto el cliente como el servidor estén conectados a la misma red. La conexión se realiza típicamente a través de un puerto Ethernet, mediante un cable cruzado. Una vez conectados a la red, el servidor debe abrir un *sockets* en su dirección IP propia y en el puerto correspondiente (por ejemplo, 10000 para IRB 140 y 11000 para IRB 1090). Si la conexión se realiza con un controlador físico, este debe tener previamente el código adecuado en su FlexPendant para facilitar la comunicación.

Con el objetivo de comprender el intercambio de información existente entre el cliente Unity y el servidor Robotstudio, la siguiente Figura 4.41 muestra de forma esquematizada el flujo de información.

4.11. COMUNICACIÓN ROBOTSTUDIO Y CONEXIÓN DEL ROBOT REAL

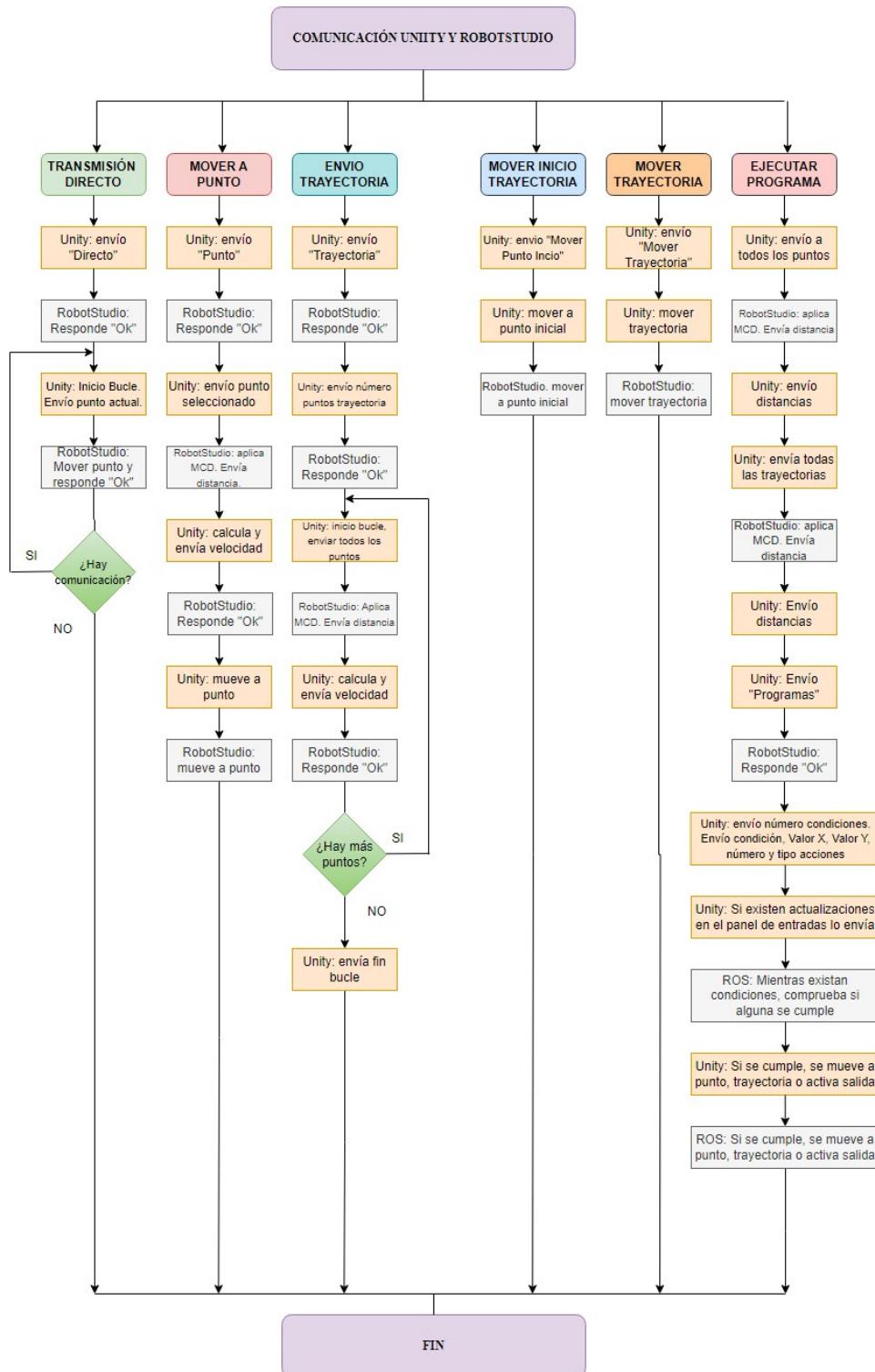


Figura 4.41. Comunicación Unity-Robotstudio

4.12. Conexión ROS

En cuanto a la conexión con el robot Schunk, tanto la comunicación como la conexión se realizan directamente con el controlador real del robot. Para llevar a cabo esta comunicación, se utiliza un proyecto previamente desarrollado [85], que proporciona una interfaz donde cada movimiento está definido por los valores de las seis articulaciones del robot y el tiempo necesario para alcanzarlas. Esta interfaz permite una comunicación directa y en tiempo real con el controlador físico del robot Schunk.

Para poder compatibilizar Unity con el proyecto base de ROS, es necesario implementar en este la lógica de gestión de información y un protocolo de comunicación. Para ello se utiliza como lenguaje de programación Python 2 junto con la librería *Sockets* para crear un servidor *socket* en la dirección IP del dispositivo donde está instalado el proyecto. Una vez creado el *socket*, se inicializa y queda en modo de escucha hasta recibir una comunicación por parte del cliente Unity. Un ejemplo de establecimiento de conexión puede ser observado en el ejemplo de la Tabla 4.3.

Para manejar el flujo de información, se emplea un código de gestión similar al utilizado para los robots ABB. Sin embargo, en este caso, el cálculo de distancias no se realiza en la aplicación del servidor. En su lugar, Unity implementa un método cinemático directo utilizando los datos del robot para calcular la distancia entre puntos, así como la velocidad y el tiempo necesarios para moverse entre ellos. Para comprender el intercambio de información existente entre Unity y ROS, en la Figura 4.42 muestra de forma esquematizada el flujo de información.

La conexión entre el cliente (Unity) y el servidor (Python) se realiza mediante un cable Ethernet, configurando correctamente la red y utilizando *sockets* para establecer la conexión. En cuanto a la conexión de Python con el controlador físico, esta funcionalidad ya está integrada en el proyecto existente.

4.12. CONEXIÓN ROS

```
## EJEMPLO COMUNICACIÓN PYTHON ##

# Importar librería socket
import rospy
import actionlib
import socket

def main():
    # Crear el servidor
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    host = 'Ip del servidor'
    port = 12000
    server_socket.bind((host, port))

    # Abrir la escucha y crear el cliente

    server_socket.listen(5)
    client_socket, addr = server_socket.accept()

    # Recibir cadena de datos:
    data = client_socket.recv(1024).decode('utf-8')

    # Enviar cadena de datos:
    response = 'ok'
    client_socket.send(response.encode('utf-8'))

    # Cerrar conexión
    client_socket.close()
    server_socket.close()

if __name__ == '__main__':
    main()
```

Tabla 4.3. Código 3: Ejemplo de comunicación en Python

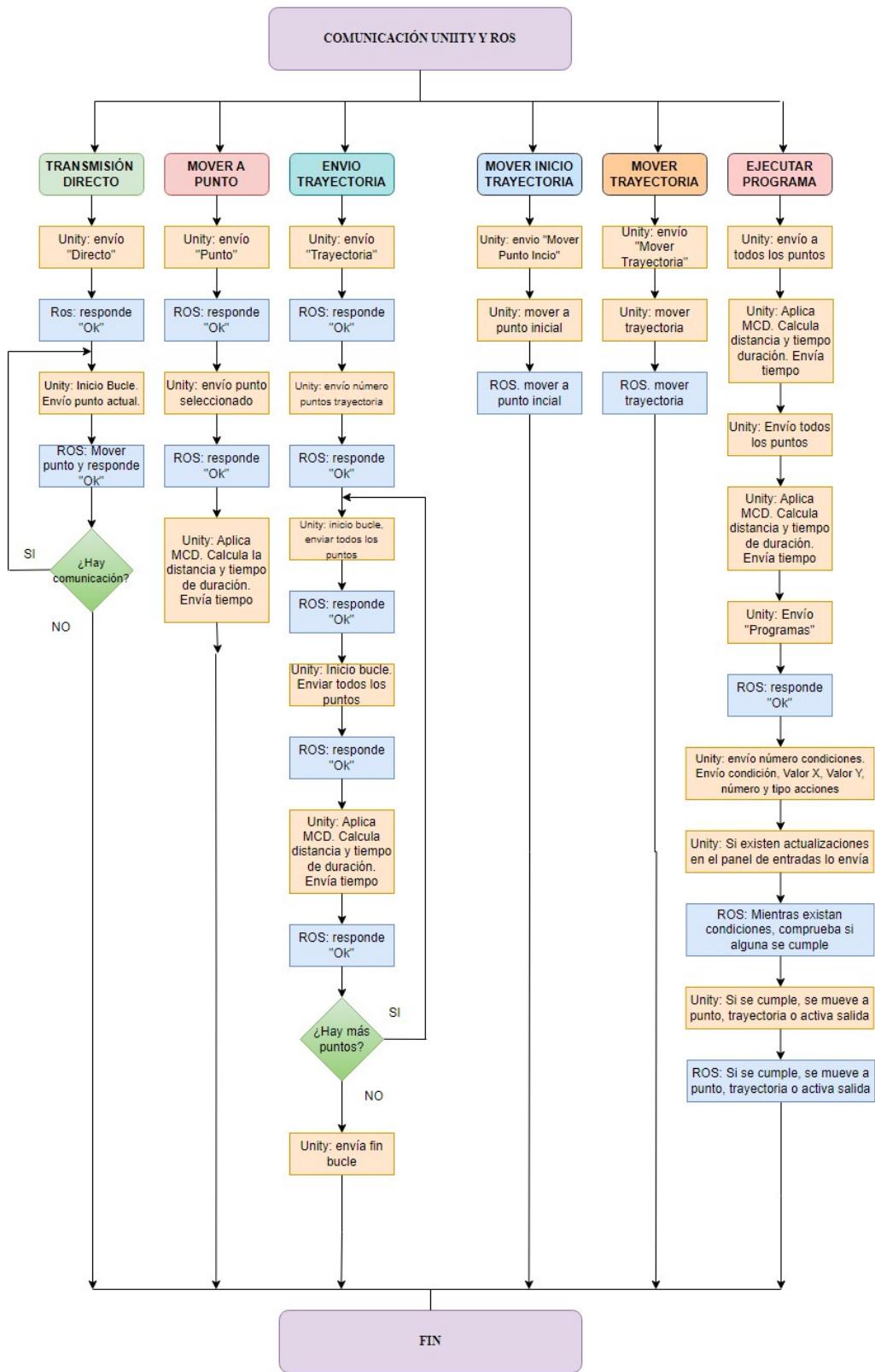


Figura 4.42. Comunicación Unity-ROS

4.12. CONEXIÓN ROS

Capítulo 5

Discusión

5.1. Análisis

5.1.1. Entorno Virtual

El desarrollo de la aplicación de RV y la integración de modelos realistas de los brazos robóticos representa el punto álgido de este TFG, ya que ofrece una innovadora forma de programación y control de robots a través de un entorno de RV. En este contexto, la mayor carga de trabajo y los esfuerzos principales se han centrado en desarrollar la aplicación de RV. Este enfoque ha sido fundamental para asegurar que la herramienta sea intuitiva, eficiente y capaz de cumplir con los objetivos planteados en el proyecto. Logrando como producto una herramienta robusta y eficiente que transforma la manera en que los usuarios pueden programar y manipular los robots, marcando un avance significativo en el campo de la robótica y la automatización.

Para mostrar el producto obtenido durante el desarrollo de la aplicación, en el capítulo 4 se pueden observar los distintos elementos y funcionalidades de la aplicación desarrollada. Además, los videos [48–50] donde se efectúan las pruebas sobre cada uno de los robots presentan una demostración donde se muestra el entorno de RV desarrollado en Unity y utilizado en este proyecto. En los vídeos se pueden apreciar la alta fidelidad de los modelos 3D de los robots, la inactividad de los paneles de control, y las capacidades de movimiento e interacción que se han integrado.

5.1. ANÁLISIS

5.1.2. Robots IRB140 e IRB1090 simulados

En cuanto a las comprobaciones de funcionalidades sobre cada uno de los robots, el primer ensayo realizado se ha llevado a cabo sobre el robot IRB 140 simulado en RobotStudio, mediante el uso de un controlador virtual programado en RAPID. Los resultados obtenidos de la simulación en RobotStudio han sido satisfactorios, validando el correcto funcionamiento de la aplicación de realidad virtual y los controladores desarrollados. No obstante, la ausencia del paquete *PC-Interface* en el controlador del robot ha impedido la realización de pruebas físicas. Esta limitación es significativa, ya que sin este módulo, no se puede establecer la comunicación necesaria para probar la comunicación en un entorno real. A pesar de esta restricción, los resultados simulados ofrecen confianza en la capacidad de la aplicación para manejar el IRB 140 de manera efectiva una vez se disponga del paquete requerido.

Para validar estos resultados, el vídeo [48] muestra el resultado de las pruebas de conexión realizadas entre Unity y RobotStudio, las cuales han arrojado resultados satisfactorios, aunque en ocasiones pueden aparecer pequeños retrasos en RobotStudio debido al tiempo de intercambio de mensajes. En este vídeo se lleva a cabo una demostración de las capacidades de la aplicación desarrollada, mediante la programación de puntos trayectorias y elaboración de programas, haciendo uso de las distintas formas de control de las que disponen los robots. En cuanto a la segunda parte del vídeo, muestra la conexión con el controlador virtual del IRB140 y se llevan a cabo sobre este desplazamiento a puntos, trayectorias, movimiento en directo y la ejecución de un programa básico de repetición iniciado mediante el accionamiento de una entrada digital.

Al igual que el IRB 140, el robot IRB 1090 ha mostrado un desempeño excelente en el entorno de simulación de RobotStudio. Todos los algoritmos de control y protocolos de comunicación desarrollados para este robot han funcionado correctamente en simulación, lo que sugiere que la aplicación está bien preparada para interactuar con este modelo. Sin embargo, no se han podido efectuar pruebas físicas debido a que el laboratorio, aún no dispone de dicho robot. A pesar de esto, los resultados de la simulación proporcionan una base sólida para esperar un funcionamiento igualmente eficiente cuando el robot esté disponible. Además, cabe destacar que en este caso no se ha hecho uso del módulo *PC-Interface* por lo que una vez se disponga del robot correctamente instalado en el laboratorio se podrán llevar a cabo pruebas sobre él.

Para validar estos resultados, se ha grabado un vídeo [49], que muestra las distintas acciones realizadas en la simulación. En este caso la estructura del video es idéntica a la del IRB140, aunque se han reducido los tiempos de ensayo, ya que una vez verificada la conectividad con RobotStudio la portabilidad a otro modelo de robot ABB no supone cambios adicionales en el controlador digital.

5.1.3. Entorno SCHUNK LWA real

En cuanto al robot Schunk LWA cabe destacar que no se dispone de un sistema de simulación, por lo que todas las pruebas de funcionamiento y comunicación se han llevado a cabo directamente sobre el robot real disponible en el laboratorio. Como paso previo a la puesta en marcha del robot, se estableció una primera conexión entre el programa de gestión del Schunk y la interfaz de Unity. Esta primera conexión permite, mediante mensajes de control mostrados en pantalla, comprobar el correcto funcionamiento del canal de comunicación y el programa de gestión de la información. Una vez comprobado que el programa gestiona correctamente los datos recibidos, se hizo uso del proyecto ya desarrollado que conecta Python con Ros para comprobar el funcionamiento conjunto de Unity con el robot real. Como resultado de estas comprobaciones y tras solucionar algunos errores iniciales, se ha obtenido una conexión exitosa en cuanto a ejecución de trayectorias, punto y programas procedentes de Unity.

En cuanto a la transmisión en directo, inicialmente generaba un retardo debido a los tiempos de intercambio de mensajes entre Unity y Ros, pero tras eliminar los mensajes de control la comunicación ha resultado exitosa, obteniendo una precisión de movimiento en cuanto a posiciones y tiempos de desplazamiento más favorable de lo que en un principio cabía esperar. Aunque ha de tenerse en cuenta que al eliminar los mensajes de control, en el improbable caso de que uno de estos mensajes no alcanzase el servidor ROS, el programa continuaría ejecutándose, de esta forma se prioriza una transmisión fluida en directo frente al estricto alcance de todos los puntos.

De esta forma se puede considerar los resultados obtenidos como exitosos y para demostrar su veracidad, el vídeo [50] muestra la prueba llevada a cabo sobre el robot real. Esta prueba consiste en una demostración completa de las funcionalidades de la aplicación y la comunicación con el controlador virtual. En este caso, al tratarse de un robot real, se han comprobado sobre todas las posibilidades de movimiento y ejecución de puntos trayectorias, programas y transmisión en directo.

5.2. Limitaciones del proyecto

Aunque el desarrollo de este proyecto puede considerarse exitoso, es importante destacar que se han enfrentado varias limitaciones que han generado dificultades o impedido la realización completa de ciertas tareas. Estas limitaciones han variado desde la disponibilidad de los robots físicos hasta la integración de ciertos paquetes de software necesarios para la comunicación efectiva entre la aplicación y los dispositivos reales. Algunas de las más relevantes son:

- **Limitación 1:** Las limitaciones de hardware han sido una preocupación significativa durante el desarrollo de este proyecto. El uso de tecnología Oculus y el motor gráfico Unity, que maneja el renderizado de texturas y ejecuta una gran cantidad de código simultáneamente, demandan recursos mínimos que a menudo no son suficientes. Esta carga se incrementa aún más con los protocolos de comunicación continua y la conexión con controladores físicos, lo que puede ocasionar tiempos de ejecución prolongados y, en ocasiones, en un rendimiento deficiente de la aplicación. Además, la disponibilidad de un ordenador con las especificaciones requeridas por Oculus no siempre está garantizada, lo que añade otro nivel de complejidad a la utilización del sistema.
- **Limitación 2:** Otra barrera importante del proyecto de realidad virtual es la ausencia de un mecanismo integrado para controlar la apertura o cierre de las pinzas del robot. Aunque este no sería difícil de implementar, ya que la apertura de la pinza puede ser interpretado como una séptima coordenada del robot.
- **Limitación 3:** Un inconveniente significativo fue la ausencia del paquete *PC-Interface* en el robot IRB 140, lo cual impidió efectuar pruebas físicas con este robot. Como resultado, las pruebas se vieron restringidas a la simulación en RobotStudio, sin poder verificar completamente la precisión y el desempeño del robot en un entorno físico.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusión

Con el desarrollo de esta memoria, se concluye que los objetivos establecidos al inicio de este TFG han sido cumplidos satisfactoriamente. Inicialmente, se planteó integrar un único brazo robótico para interactuar de manera intuitiva, pero a lo largo del proyecto se exploraron diversas formas de movimiento que enriquecieron las capacidades del robot, facilitando al usuario alcanzar posiciones objetivo de manera eficiente. La inclusión de tres brazos robóticos diferentes permitió trabajar con múltiples lenguajes de programación y plataformas como Unity, ROS, RobotStudio y Matlab, ampliando así la funcionalidad de la aplicación y proporcionando al usuario una experiencia más diversa con robots colaborativos e industriales. La exploración de las posibilidades de Unity y la RV fue fundamental para desarrollar una aplicación amigable y orientada a la interacción y movilidad del usuario. Aunque la conexión física se logró únicamente con el robot colaborativo Schunk, debido a limitaciones técnicas con otros modelos como el IRB 1090 y el IRB 140, la simulación en RobotStudio confirmó el correcto funcionamiento de la aplicación de RV, los protocolos de comunicación y los controladores desarrollados.

La conexión física con el robot Schunk permitió ejecutar todas las acciones programadas desde la aplicación de RV y manipular en tiempo real su análogo físico. Este proyecto ha sido una experiencia enriquecedora donde se aplicaron los conocimientos adquiridos en el grado de Ingeniería Electrónica Industrial y Automática cursado en la Universidad de Almería, alcanzando de manera satisfactoria todos los objetivos establecidos.

6.2. Trabajos Futuros

Para continuar avanzando en este campo de investigación y desarrollo, se vislumbran varias líneas de trabajo futuro que podrían ampliar y mejorar los resultados obtenidos en este proyecto. Dos áreas destacadas para futuros trabajos incluyen:

- **Trabajo futuro 1:** Una posible línea de trabajo futuro sería la integración de este proyecto con el trabajo [116] , el cual trata del desarrollo de una interfaz para escanear y conectar controladores ABB dentro de una red. Al combinar ambos proyectos, sería factible implementar la capacidad de conectar de forma remota cualquier controlador disponible en la red desde la aplicación de RV. Esto permitiría no solo establecer conexión con los controladores, sino también descargar y ejecutar programas *Rapid* de manera remota, todo desde una interfaz unificada y accesible en el entorno de RV. Este avance no solo ampliaría las capacidades de programación y control, sino que también podría facilitar la gestión y operación de múltiples robots desde una única plataforma integrada, facilitando así el desarrollo de un entorno cooperativo.
- **Trabajo futuro 2:** Otra dirección prometedora para futuros desarrollos sería la implementación de un sistema de visión o escaneo dentro del entorno de trabajo del robot. Este sistema estaría diseñado para detectar los objetos presentes en el entorno físico, generar modelos virtuales de estos objetos y representarlos en tiempo real en Unity. Integrando esta funcionalidad con la capacidad de controlar las pinzas del robot, se podría permitir a las pinzas interactuar con estos objetos detectados. Esto abriría la posibilidad de agarrar y desplazar objetos tanto en el entorno virtual como en el mundo real de manera precisa y controlada, mejorando así la capacidad de manipulación y automatización en aplicaciones industriales y de investigación.

Bibliografía

- [1] Media.linkedin, “Figura unimate.” Último acceso: 22 de junio de 2024.
- [2] Y. Fernández, “El primer simulador de rv de la historia,” 2021. Último acceso: 22 de junio de 2024.
- [3] Nintenderos, “Virtual boy nintendo,” 2022. Último acceso: 22 de junio de 2024.
- [4] Amazon, “Htc vive,” 2024. Último acceso: 22 de junio de 2024.
- [5] Amazon, “Meta quest,” 2024. Último acceso: 22 de junio de 2024.
- [6] E. Dans, “Apple vision pro,” 2024. Último acceso: 22 de junio de 2024.
- [7] HistoryofInformatio.com, “Irb 6,” 1977. Último acceso: 22 de junio de 2024.
- [8] D. A. S. Mejia and J. A. A. Guevara, “Dimensionamiento de actuadores para mecanismos de robot zoomórfico,” 2023.
- [9] Robotshop, “Robot autónomo.” Último acceso: 22 de junio de 2024.
- [10] Multirobotica, “Robot zoomórficos,” 2015. Último acceso: 22 de junio de 2024.
- [11] Free3d, “Brazo robotico,” 2021. Último acceso: 22 de junio de 2024.
- [12] R. GONZÁLEZ, “Robots humanoides,” 2022. Último acceso: 22 de junio de 2024.
- [13] KOLVI, “Robots cobots,” 2020. Último acceso: 22 de junio de 2024.
- [14] Europapress, “Un robot que vuela, conduce y se estruja con los mismos motores,” 2019. Último acceso: 22 de junio de 2024.
- [15] Intelitek, “Descripción de las características del brazo,” 2003. Último acceso: 22 de junio de 2024.
- [16] U. Robótica, “Control de un robot,” 2024. Último acceso: 22 de junio de 2024.
- [17] Shop.bvs, “Flexpendant, reacondicionamiento de productos, recambios, piezas nuevas, servicio.,” 2024. Último acceso: 22 de junio de 2024.

BIBLIOGRAFÍA

- [18] D. xr, “Haptx lanza su nueva versión de guantes hápticos,” 2021. Último acceso: 22 de junio de 2024.
- [19] Knoxlabs, “Kat walk c2, 2nd-generation personal vr treadmill.,” 2023. Último acceso: 22 de junio de 2024.
- [20] T. Hoang, “Aplicación rv en cirugía,” 2019. Último acceso: 22 de junio de 2024.
- [21] INMERSIVO, “Rv en hospitales,” 2018. Último acceso: 22 de junio de 2024.
- [22] L. Prensa, “Pokémon go, un fenómeno social que impulsa las acciones de nintendo,” 2024. Último acceso: 22 de junio de 2024.
- [23] Reasonwhy, “Ruinas romanas en rv,” 2018. Último acceso: 22 de junio de 2024.
- [24] A. Rueda, “Prototipo bmw,” 2020. Último acceso: 22 de junio de 2024.
- [25] uc3m, “Logo robotstudio,” 2024. Último acceso: 22 de junio de 2024.
- [26] tknika, “Ros,” 2022. Último acceso: 22 de junio de 2024.
- [27] D. Cerrillo Vacas, *Metodología de Modelado de Robots Hiperredundantes Actuados por Cables*. PhD thesis, Industriales, 2022.
- [28] Amazon, “Oculus rift s,” 2019. Último acceso: 22 de junio de 2024.
- [29] M. Pollák, J. Török, J. Zajac, M. Kočiško, and M. Telišková, “The structural design of 3d print head and execution of printing via the robotic arm abb irb 140,” in *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*, pp. 194–198, IEEE, 2018.
- [30] R. Ramirez, “Irb 140 espacio de trabajo,” 2013. Último acceso: 22 de junio de 2024.
- [31] ABB, “Irb 1090,” 2023.
- [32] library.e.abb, “Manual irb 1090,” 2023.
- [33] ruc.udc, “Proyecto powerball,” 2016. Último acceso: 22 de junio de 2024.
- [34] wikipedia, “Logo unreal,” 2024. Último acceso: 22 de junio de 2024.
- [35] wikipedia, “Logo epic games,” 2024. Último acceso: 22 de junio de 2024.
- [36] wikipedia, “Logo unity,” 2024. Último acceso: 22 de junio de 2024.
- [37] A. S. en Tutoriales de Hosting, “Modelo cliente servidor,” 2019. Último acceso: 22 de junio de 2024.
- [38] Real Academia Española, “Real academia española,” 2019. Último acceso: 10 de junio de 2024.

- [39] I. Asimov, *I, robot*, vol. 1. Spectra, 2004.
- [40] J. Wallén, *The history of the industrial robot*. Linköping University Electronic Press, 2008.
- [41] R. M. Baños, C. Botella, I. Rubió, S. Quero, A. García-Palacios, and M. Alcañiz, “Presence and emotions in virtual environments: The influence of stereoscopy,” *CyberPsychology & Behavior*, vol. 11, no. 1, pp. 1–8, 2008.
- [42] G. C. Burdea and P. Coiffet, *Virtual reality technology*. John Wiley & Sons, 2003.
- [43] J. Lanier, *Dawn of the new everything: Encounters with reality and virtual reality*. Henry Holt and Company, 2017.
- [44] A. Suh and J. Prophet, “The state of immersive technology research: A literature analysis,” *Computers in Human behavior*, vol. 86, pp. 77–90, 2018.
- [45] ual arm, “Vr_arm_rob.” https://github.com/ual-arm/VR_ARM_ROB, 2024. Accedido: 22 de junio de 2024.
- [46] ABB, “Especificaciones del producto irb 1090,” 2023. Último acceso: 22 de junio de 2024.
- [47] ABB, “Especificaciones del producto irb 140,” 2004. Último acceso: 22 de junio de 2024.
- [48] A. Martínez, “Programando con realidad virtual el robot irb140.” YouTube vídeo, 2024.
- [49] A. Martínez, “Programando con realidad virtual el robot irb1090.” YouTube vídeo, 2024.
- [50] A. Martinez, “Programando con realidad virtual el robot schunk lwa 4p.” YouTube video, 2024.
- [51] UAL, “Competencias ual,” 2008. Último acceso: 10 de junio de 2024.
- [52] UAL, “Competencias ual electrónica industrial y automática,” 2010. Último acceso: 10 de junio de 2024.
- [53] Robotnik, “Historia de los robots y la robótica,” 11 2021.
- [54] ABB, “El amanecer del robot,” 2014. Último acceso: 10 de junio de 2024.
- [55] M. P. Groover, *Automation, production systems, and computer-integrated manufacturing*. Pearson Education India, 2016.
- [56] E. Brynjolfsson and A. McAfee, *The second machine age: Work, progress, and prosperity in a time of brilliant technologies*. WW Norton & Company, 2014.
- [57] DEISER, “Robótica industrial: Qué es, usos y aplicaciones,” 2023. Último acceso: 10 de junio de 2024.

BIBLIOGRAFÍA

- [58] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*, vol. 200. Springer, 2008.
- [59] Z. Chan, “Partes de un robot industrial: Componentes básicos para su funcionamiento,” 11 2023.
- [60] M. J. Mataric, *The robotics primer*. MIT press, 2007.
- [61] J. M. Stritch, N. Darnall, L. Hsueh, and S. Bretschneider, “Green technology firms and sustainable public purchasing,” *IEEE Engineering Management Review*, vol. 46, no. 1, pp. 128–131, 2018.
- [62] D. Rawat, C. Brecher, H. Song, and S. Jeschke, “Industrial internet of things: Cybermanufacturing systems,” *Cham, Switzerland: Springer*, 2017.
- [63] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Force control*. Springer, 2009.
- [64] D. G. Caldwell, *Robotics and automation in the food industry: current and future technologies*. Elsevier, 2012.
- [65] G. C. Burdea and P. Coiffet, *Virtual reality technology*. John Wiley & Sons, 2003.
- [66] W. R. Sherman and A. B. Craig, *Understanding virtual reality: Interface, application, and design*. Morgan Kaufmann, 2018.
- [67] A. El Saddik, M. Orozco, M. Eid, and J. Cha, *Haptics technologies: Bringing touch to multimedia*. Springer Science & Business Media, 2011.
- [68] M. W. C. Barcelona, “El boom de los videojuegos de realidad virtual,” *Mobile World Capital Barcelona*, 2022.
- [69] R. T. Azuma, “A survey of augmented reality,” *Presence: teleoperators & virtual environments*, vol. 6, no. 4, pp. 355–385, 1997.
- [70] B. Furht, *Handbook of augmented reality*. Springer Science & Business Media, 2011.
- [71] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, “Recent advances in augmented reality,” *IEEE computer graphics and applications*, vol. 21, no. 6, pp. 34–47, 2001.
- [72] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, “Augmented reality: A class of displays on the reality-virtuality continuum,” in *Telemanipulator and telepresence technologies*, vol. 2351, pp. 282–292, Spie, 1995.
- [73] C.-N. Carolina, “Surround-screen projection-based virtual reality: The design and implementation of the cave,” in *ACM SIGGRAPH 93*, 1993.

- [74] W. R. Sherman and A. B. Craig, *Understanding virtual reality: Interface, application, and design*. Morgan Kaufmann, 2018.
- [75] F. Biocca and M. R. Levy, *Communication in the age of virtual reality*. Routledge, 2013.
- [76] P. Hernández, “Estado de los juegos de vr–los motores de juego y la convencionalidad presente,” 2024. Último acceso: 10 de junio de 2024.
- [77] E. Gil Romero *et al.*, “Un estudio acerca del desarrollo de videojuegos mediante el motor grafico unity 3d,” 2014.
- [78] I. Ouazzani, “Manual de creación de videojuego con unity 3d,” 2012.
- [79] C. Todd, “Game development in unity using oculus quest vr.”
- [80] A. Robotics, “Robotstudio,” 2020. Último acceso: 11 de junio de 2024.
- [81] A. Robotics, “Robotstudio user manual,” 2020. Último acceso: 11 de junio de 2024.
- [82] e. a. Bladh, T., “Virtual commissioning of industrial robot systems using abb robotstudio,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10073–10078, 2017.
- [83] O. S. R. Foundation, “Ros: An open-source robot operating system.,” 2022. Último acceso: 11 de junio de 2024.
- [84] e. a. Quigley, M., “Ros: An open-source robot operating system.,” *CRA Workshop on Open Source Software, Kobe, Japan*, vol. 1, no. 1, pp. 1050–4729, 2009.
- [85] F. C. Aránega.
- [86] A. O. Baturone, *Robótica: manipuladores y robots móviles*. Marcombo, 2005.
- [87] C. Peter, “Robotics, vision and control: fundamental algorithms in matlab,” 2011.
- [88] J. J. Craig, “Introduction to robotics.,” 2005.
- [89] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020.
- [90] A. Barrientos, M. Álvarez, J. Hernández, J. Del Cerro, and C. Rossi, “Modelado de cadenas cinemáticas mediante matrices de desplazamiento. una alternativa al método de denavit-hartenberg,” *Revista Iberoamericana de Automática e Informática industrial*, vol. 9, no. 4, pp. 371–382, 2012.
- [91] J. J. Craig, “Introduction to robotics.,” 2005.
- [92] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Force control*. Springer, 2009.

BIBLIOGRAFÍA

- [93] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.
- [94] J. D. R. C. Manuel Leonardo Ayala G, “Cinemática directa e inversa para robots,” 2016. Último acceso: 11 de junio de 2024.
- [95] uclm, “Cinemática directa e inversa,” 2014. Último acceso: 11 de junio de 2024.
- [96] G. J. Cebrián Chaustre, “Diseño de una plataforma didáctica para el estudio de protocolos de encaminamiento en arquitecturas tcp/ip,” 2021.
- [97] A. E. Corona *et al.*, “Protocolos tcp/ip de internet,” 2004.
- [98] Oculus, “Oculus rift s,” 2022. Último acceso: 11 de junio de 2024.
- [99] B. Lang, “Oculus rift s review: An advanced vr headset, inside out,” 2019. Último acceso: 11 de junio de 2024.
- [100] Oculus, “Oculus rift s: Our most advanced pc-powered headset,” 2019. Último acceso: 11 de junio de 2024.
- [101] A. Robotics, “Abb irb 140 - compact robot for arc welding and handling applications.” Último acceso: 22 de junio de 2024.
- [102] RobotWorx., “”abb irb 140 robot.”” Último acceso: 22 de junio de 2024.
- [103] A. Robotics., “Irb 1090 - industrial robot,” 2022.
- [104] A. Robotics., “Irb 1090 product manual,” 2022.
- [105] A. Robotics., “Irb 1090 brochure.,” 2022.
- [106] SCHUNK, “Lwa 4p - lightweight arm.,” 2022.
- [107] SCHUNK, “Lwa 4p product manual.,” 2022.
- [108] SCHUNK, “Lwa 4p brochure,” 2022.
- [109] J. L. S. Escudero, “Control cinemático y de fuerza de una mano robótica para el agarre estable,” 2014. Último acceso: 11 de junio de 2024.
- [110] J. L. Bustos, “¿Qué es un motor gráfico en realidad aumentada? [2024],” 6 2024.
- [111] D. Eberly, *3D game engine design: a practical approach to real-time computer graphics*. CRC Press, 2006.
- [112] N. Valcasara, *Unreal engine game development blueprints*. Packt Publishing Ltd, 2015.

- [113] J. Zhang, Y. Lyu, Y. Wang, Y. Nie, X. Yang, J. Zhang, and J. Chang, “Development of laparoscopic cholecystectomy simulator based on unity game engine,” in *Proceedings of the 15th ACM SIGGRAPH European conference on visual media production*, pp. 1–9, 2018.
- [114] K. R. Fall and W. R. Stevens, *Tcp/ip illustrated*, vol. 1. Addison-Wesley Professional, 2012.
- [115] A. S. Tanenbaum and D. J. Wetherall, “Computer networks, 5th,” 2011.
- [116] M. F. Núñez.

BIBLIOGRAFÍA



Resumen/Abstract

En la actualidad, la evolución de la robótica juega un papel fundamental en el ámbito del desarrollo industrial y social. Ante la creciente necesidad de automatización y robotización de procesos industriales y actividades cotidianas, surgen nuevas demandas en cuanto a la interacción con los usuarios. Por ello, la integración de nuevas tecnologías en el ámbito de la robótica es crucial para ofrecer nuevas oportunidades. Este Trabajo de Fin de Grado (TFG) explora una novedosa forma de interacción humano-robot basada en la tecnología de realidad virtual, la cual se encuentra en un momento álgido de su desarrollo. El objetivo se centra en el desarrollo de un entorno virtual realista en Unity donde se emula el comportamiento de tres brazos robóticos (IRB 140, IRB 1090 y SCHUNK LWA 4P), accesible mediante la tecnología de las gafas de realidad virtual "Oculus". Este entorno permite interactuar con robots mediante movimientos manuales y por ejes, ofreciendo diferentes opciones de programación como guardado de puntos, simulación de E/S digitales y creación de programas básicos. La comunicación se realiza mediante el protocolo TCP/IP (\textit{Transmission Control Protocol/Internet Protocol}), conectando el entorno con RobotStudio para los robots de IRB y ROS (\textit{Robot Operating System}) para el Schunk, llegando a conectar este último con el modelo real. En el presente proyecto, se busca explorar las posibilidades emergentes de la realidad virtual en el campo de la robótica y contribuir al avance de la interacción humano-robot.

Today, the evolution of robotics plays a key role in industrial and social development. With the growing need for automation and robotization of industrial processes and everyday activities, new demands arise in terms of interaction with users. Therefore, the integration of new technologies in the field of robotics is crucial to offer new opportunities. This Final Degree Project (TFG) explores a novel form of human-robot interaction based on virtual reality technology, which is at the peak of its development. The objective focuses on developing a realistic virtual environment in Unity where the behavior of some robotic arms (IRB 140, IRB 1090, and SCHUNK LWA 4P) is emulated, and accessible through the technology of the virtual reality glasses "Oculus". This environment allows interaction with robots through manual and axis movements, offering different programming options such as saving points, simulation of digital I/O, and creation of basic programs. Communication is via TCP/IP (Transmission Control Protocol/Internet Protocol), connecting the environment with RobotStudio for the ABB robots and ROS (Robot Operating System) for the Schunk, and even connect the latter to the real model. In the present project, the aim is to explore the emerging possibilities of virtual reality in the field of robotics and contribute to advancing human-robot interaction.