UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Control Robótico mediante Tecnologías Inmersivas Documentación Técnica

Presentado por Víctor de la Iglesia García
en Universidad de Burgos — 12 de junio de 2022
Tutor académico: Carlos Cambra Baseca
Tutor empresarial (ITCL): Alejandro Langarica Aparicio

CAMBRA BASECA CARLOS - 18053084Q

Firmado digitalmente por CAMBRA BASECA CARLOS - 18053084Q
Fecha: 2022.06.12 19:44:13 +02'00'

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

El plan de proyecto software relaciona todo aquello referido a la gestión del proyecto. Es un proceso muy importante para realizar antes de comenzar a trabajar en el desarrollo. En este punto, se llevan a cabo cosas como la planificación de tiempos para realizar las tareas, se definen los costes económicos del proyecto o l,a forma de trabajo.

Veremos dentro de la planificación temporal la manera en la que el trabajo y sus cargas han sido repartidas a lo largo de este tiempo desarrollando el proyecto. Todo esto tras haber analizado tanto el número de tareas a realizar, como la naturaleza de todas estas.

Por otra parte, a través del estudio de viabilidad podremos poner la vista en las partes ecnómicas del proyecto. Analizaremos los costes derivados del proyecto y la gestión de las licencias utilizadas.

Podremos encontrar dos partes diferenciadas dentro de ese apartado:

- Estudio de la viabilidad económica: Costes del proyecto dividiendo y analizando tanto en hardware, software como en personal.

- Estudio de la viabilidad legal: Análisis de las licencias para el uso de software especializado en el proyecto.

A.2. Planificación temporal

Como se comenta dentro de la memoria de este proyecto, la manera de trabajar y la metodología utilizada es la de SCRUM[2] dentro del marco de las metodologías ágiles de desarrollo de software. Planificando dividiendo el tiempo en sprints que han de cubrirse en el tiempo marcado, con revisiones periódicas del trabajo.

Dentro de esto podemos ver que las 300 horas asignadas desde la Universidad de Burgos e ITCL[4] han sido divididas en un total de seis sprints, cuya duración son de un total de dos semanas en todos los casos.

Además, comentar que podríamos considerar como un sprint previo al trabajo la primera semana donde se realizó la toma de decisiones en referencia a lo que se iba a hacer, planificación de tareas y la metodología a seguir en cuanto a la forma de trabajar. También vimos que repositorios utilizaríamos, herramientas para la gestión o el control de versiones.

Sprint 1 (07/03/2022-18/03/2022)

En este primer sprint se llevaron a cabo las instalaciones de aquellos softwares necesarios para el correcto desarrollo de esta parte del trabajo. También se establecieron unos puntos a cumplir relacionados en primer lugar con el 'aterrizaje' en el proyecto y en segundo lugar con el aprendizaje relativo a los guantes hápticos Nova [6].

Las tareas dispuestas a cumplir son:

- Continuar con el estudio de algunos aspectos de Unity[9].
- Estudio documentación relativa a los guantes y su SDK.
- Monitorizar y parametrizar lo que ofrecen los guantes.
- Desarrollo script que recoja datos de los guantes y aplique fuerzas hápticas[11] o vibraciones.
- Desarrollo escena de representación 3D con
    1. Paneles que controlen las fuerzas y vibraciones.
    2. Paneles que muestren datos de los sensores.

El desarrollo de este primer sprint se saldó de manera correcta sirviendo así para poder adaptarme fácilmente al proyecto y la empresa.

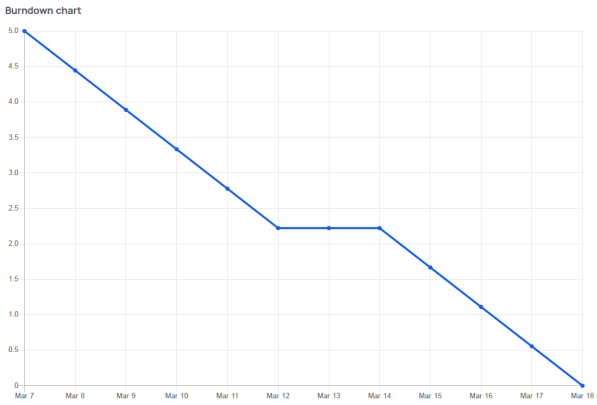

Figura A.1: Cierre de tareas del SPRINT 1

Sprint 2 (21/03/2022-01/04/2022)

En este segundo sprint los objetivos iban encaminados a mezclar por primera vez tanto el dispositivo Oculus Quest 2[1] y los guantes Nova. Se llevó a cabo la instalación del software de Oculus y llevamos a cabo la primera conexión. Se realizaron pruebas con ellas.

Para dar por correcta esta fusión de ambas partes teníamos las siguientes tareas:

- Estudio de la componente[7] Transform de Unity, enfatizando en la posición.
- Análisis de las posibilidades de seguimiento (tracking) de Oculus.
- Sincronización de la vista en el mundo virtual con la visión de las gafas.
- Crear una escena con la adaptación de la primera escena a la realidad virtual.
- Cambiar el panel de fuerzas por uno que muestre datos de seguimiento.
- Representación tridimensional de la mano con desplazamiento en el espacio.

Para llevar a cabo estas tareas surgieron otras como que hizo falta repasar algunos conceptos de Unity, así como desarrollar scripts que funcionaran como conectores entre tecnologías para representar correctamente en el

espacio las manos. También derivado de estas tareas fue el montaje del soporte de Oculus para los guantes.

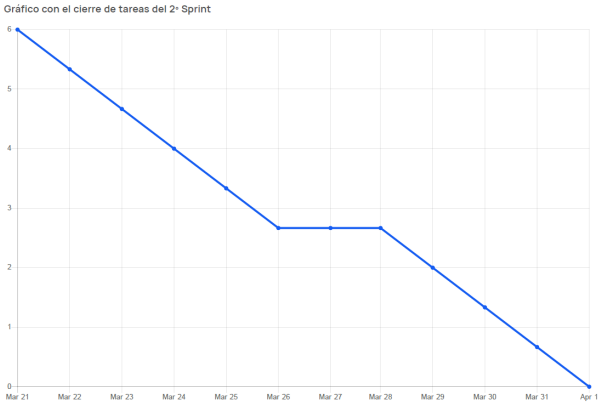

Figura A.2: Cierre de tareas del SPRINT 2

Sprint 3 (04/04/2022-15/04/2022)

Una vez llevada a cabo la conexión de ambas tecnologías, el objetivo era buscar poco a poco la manera de trabajar con el robot a nivel virtual. Para ello, intentaríamos crear un robot virtual a partir de un modelo diseñado que se comporte de maneras similares.

Las tareas dispuestas para este sprint fueron:

- Estudio de la componente[7] Transform de Unity, enfatizando en la rotación y sus valores.
- Creación de nueva escena para la replicación virtual de la pinza del robot.
- Implementación de métodos:
    1. Get y Set de la posción en el espacio tridimensional.
    2. Get y Set de la apertura de la pinza.
    3. Movimiento en el espacio dada una velocidad lineal y una posición objetivo.
    4. Movimiento que controla la apertura de pinza dada una velocidad de apertura y una apertura objetivo.

- Unificar rotación de cada brazo de la pinza para abrir y cerrar de manera correcta.
- Ejecución de pruebas para comprobar el funcionamiento.

En este sprint he visto clave el tener que conocer a fondo la manera en la que funcionan las rotaciones dentro de este motor gráfico, ya que tuve alguna complicación a la hora de gestionar estos aspectos en el proyecto.

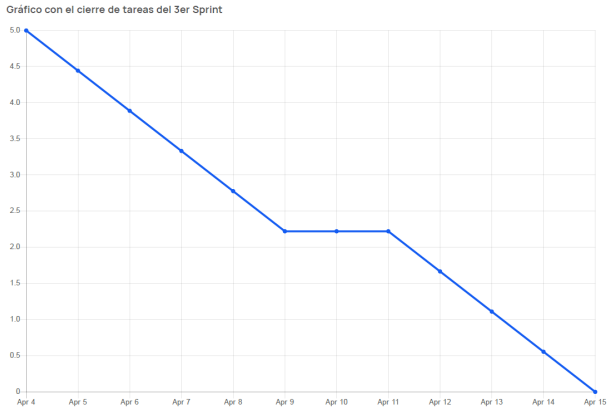

Figura A.3: Cierre de tareas del SPRINT 3

Sprint 4 (18/04/2022-29/04/2022)

El objetivo de llevar a cabo la implementación de los métodos descritos en el anterior sprint en un script, era para poder dotar a la pinza de funciones reales como si fuera el robot. De esta manera, en este sprint el objetivo buscado era fusionar tanto la replicación virtual del robot con la tecnología de los guantes y las gafas desarrollada en el segundo sprint.

Las tareas seleccionadas para este sprint fueron:

- Mejorar la automatización de la apertura y cierre de la pinza.
- Creación de nueva escena donde serán apreciados los resultados.
- Adaptación de la escena de la pinza a esta nueva mediante realidad virtual[12]
- Mostrar virtualmente la representación de las manos en la escena.

- Diseñar idea que permita conectar los guantes y la pinza.
- Desarrollo del script conector que
  1. Desplace la pinza a una posición cercana de las manos a una velocidad dada.
  2. Abra o cierre la pinza simulada en función de la apertura de nuestro dedo índice y pulgar, dada una velocidad.
- Fusionar todos los puntos en la escena.
- Ejecutar pruebas de desplazamiento y apertura para comprobar el correcto funcionamiento.

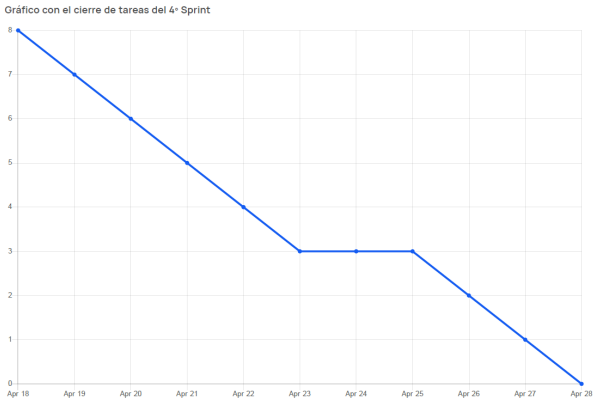

Figura A.4: Cierre de tareas del SPRINT 4

Sprint 5 (02/05/2022-13/05/2022)

Este sprint estaba pensado para comenzar a trabajar con el robot real, ya buscando resultados que den valor a lo realizado durante el proyecto. De esta manera tanto este como el siguiente sprint se antojan como unos de los más complicados, ya que nunca había trabajado con robots previamente a este proyecto.

Los objetivos marcados para cumplir este sprint vienen dados por la consecución de las siguientes tareas:

- Investigación y recopilación de información sobre el brazo robótico Kinova

- Estudio breve a modo de introducción de ROS [5]
- Establecer conexión correcta con el robot.
- Acceder al robot y a ROS para buscar la lista de topics(funciones) disponibles.
- Ejecución de pruebas con el brazo robótico desde otro proyecto.
- Parametrización de valores y datos necesarios.
- Desarrollo de una nueva escena de conexión con ROS y el robot desde Unity.
- Desarrollar un script que permita desplazar la pinza del robot en el espacio.

A la correcta consecución de las tareas de este sprint comenzamos a ver poco a poco cual sería el funcionamiento finalmente del robot.

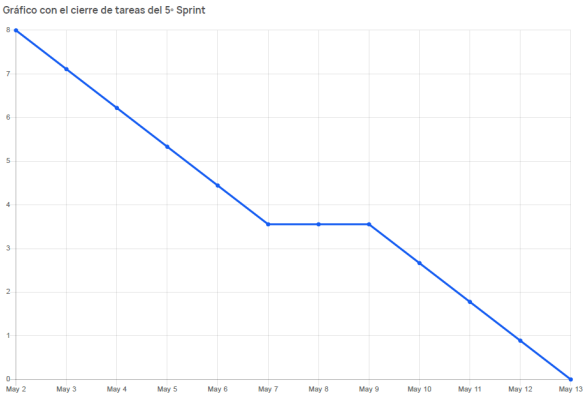

Figura A.5: Cierre de tareas del SPRINT 5

Sprint 6 (16/05/2022-27/05/2022)

Llegados al último sprint dispuesto para el proyecto, nos encontramos muy cerca de los resultados finales deseados. Por ello para este final sprint nos encargaremos de pulir algunos problemas y finalmente dejar perfilada la resolución del proyecto. En la consecución de este sprint encontraremos también la resolución positiva de los objetivos principales de proyecto marcados.

Aquí al acabar este sprint quedarían fusionadas todas las partes y tecnologías tanto vistas, como utilizadas dando lugar al resultado final.

Para finalizar el proyecto de una manera positiva y que se asemeje a lo trazado al comenzar el proyecto deberíamos cumplir las siguientes tareas:

- Desarrollo de script que linkee nuestra mano con la pinza del robot y el desplazamiento en el espacio.
- Implementación y sincronización de rotación de la pinza con rotación de la mano.
- Diseño y desarrollo de un script que gestione la apertura y cierre de la pinza del robot.
- Limitar y gestionar valores relativos a la posición para una satisfactoria experiencia del usuario.
- Establecer un comando para regreso a posición 'home' para el robot.
- Desarrollo de la escena final donde utilizar todo lo llevado a cabo en el proyecto
- Ejecución de pruebas de funcionamiento.
- Ejecución de pruebas con desplazamiento de objetos.
- Filmación de vídeos que muestren el funcionamiento de lo realizado.

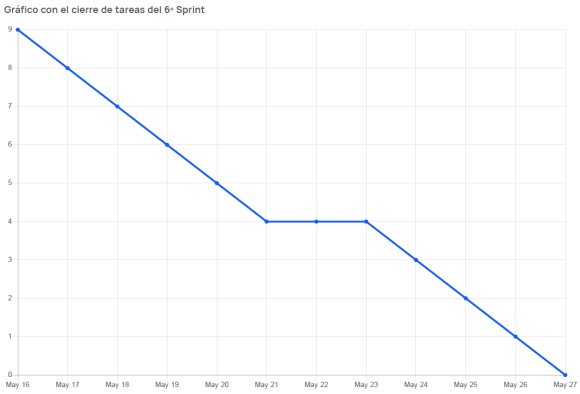

Figura A.6: Cierre de tareas del SPRINT 6

Final de Proyecto

Llegados a la última semana dentro de ITCL[4](30/05/2022-03/06/2022) podemos asegurar que todos los objetivos principales dispuestos han sido cumplidos correctamente, además varios de los secundarios también.

Cabe destacar que la manera de dividir las tareas y organizar el trabajo dentro del proyecto ha sido una de las cosas que ha permitido el ir consiguiendo los objetivos marcados en el tiempo marcado.

A partir de este punto el objetivo que me marco en este sprint 'personal' es el de continuar con la generación de documentación con la herramienta Overleaf, con LaTeXque ya había ido desarrollando en paralelo con la parte de software del proyecto. De gran ayuda es para el desarrollo de esta documentación el haber ido labrando un documento a modo de diario que iba cumplimentando todas las semanas con lo hecho en esos días.

La entrega y final de esta etapa de TFG finaliza el próximo 13 de Junio de 2022 y hasta entonces el tiempo está dedicado a preparar toda la documentación, anexos y entregables.

A.3. Estudio de viabilidad

Viabilidad económica

Uno de los aspectos más importantes a la hora de llevar a cabo el desarrollo de un proyecto es el de controlar los costes y derivados del mismo.

Es por ello que en este apartado dentro del anexo veremos y analizaremos estos costes. Para ello y con vista de que sea más fácil de comprender y desarrollar, dividiremos los costes totales en tres claramente diferenciados:

1. Coste a nivel **Hardware**
2. Coste a nivel **Software**
3. Coste a nivel de **Personal**

Coste a nivel Hardware

Dentro de nuestro proyecto algo que juega un papel fundamental es la parte tecnológica, relacionada tanto con la tecnología de realidad virtual[12] como con la robótica[10] y el ordenador utilizado. Esta parte tiene un determinado coste a nivel de hardware, ya que el utilizado en el proyecto es sofisticado y especializado, que podríamos considerar por encima de lo común.

A continuación se muestra una tabla donde podemos ver tanto el nombre del producto, como el coste propio del producto y la amortización, estableciendo una amortización de cinco años y un uso de tres meses del producto.

Producto	Coste	Amortización
SenseGlove Nova	4499€	899,8€
Oculus Quest 2	350€	70€
Robot Kinova	20.000€	4000€
Ordenador ITCL	750€	150€
Total	25.599€	5119,8€

Tabla A.1: Coste a nivel Hardware

Coste a nivel Software

Otro de los aspectos que tiene vital importancia a la hora de desarrollar software es el coste específico de todos aquellos programas o licencias necesarias para llevar a cabo el desarrollo. Es por ello que a continuación vemos una tabla con el coste de software o de las licencias, considerando dos años de amortización.

Producto	Coste	Amortización
Windows 10 Pro	259€	129,5€
Unity Student	0€	0€
JetBrains Rider (Student)	0€	0€
Overleaf(LaTeX) University	0€	0€
Total	259€	129,5€

Tabla A.2: Coste a nivel Software

Coste a nivel de Personal

El último aspecto que queda por cubrir a nivel de costes es el destinado al pago del sueldo de los integrantes del personal. En este caso un único desarrollador que ha trabajado durante 300 horas dispersadas en 12 semanas, considerando un sueldo de 25000€ anuales brutos(19.880€ netos), el coste de personal sería:

Concepto de pago	Coste
Retención por IRPF	333.91€
Cuota a la Seg. Social	147,17€
Sueldo neto mensual (12 pagas)	1.602,5€
Coste mensual trabajador	2083,58€
Coste total 12 semanas	**6250,74€**

Tabla A.3: Costes de Personal

Estos costes se han trazado siguiendo un porcentaje de retención de IRPF del 16,03 % y una cuota de seguridad social del 28,3 %.

Costes Totales

Tipo Gasto	Coste
Hardware	25.599€
Software	129,5€
Personal	6250,74€
Total	**31.979,24€**

Tabla A.4: Costes Totales

Apéndice B

Especificación de Requisitos

B.1. Introducción

A lo largo de este apartado veremos documentadas las posibilidades funcionales de lo desarrollado en el proyecto, centrándonos en el software final. Se especificarán cuáles eran los objetivos minimos requeridos para considerar de un buen resultado el desarrollo del proyecto, veremos tanto los requisitos funcionales como no funcionales de la aplicación

B.2. Objetivos generales

El objetivo general dentro de este proyecto a nivel de desarrollo, era el de llevar a cabo una aplicación que nos habilite a conseguir los siguientes puntos:

- Establecer conexión inalámbrica entre el robot y el ordenador.
- Conectar el dispositivo Oculus Quest 2 a Unity[9]
- Conectar los guantes Nova al ordenador y a Unity.
- Recoger datos relevantes de seguimiento y posicionamiento de los controladores Oculus.
- Recoger datos relevantes de los sensores de los guantes Nova.
- Representar en las tres dimensiones del espacio virtualmente nuestra mano.

- Desplazar la pinza del robot en las tres dimensiones del espacio siguiendo el movimiento de nuestra mano.
- Abrir y cerrar la pinza con un gesto de nuestra mano.

B.3. Catalogo de requisitos

Aquí veremos el catálogo de requisitos de nuestro software dividiendo en los siguientes dos:

Requisitos Funcionales

- RF.1 Conexión entre el robot ROS[5] y el ordenador (ROSInitializer)
  1. RF.1.1 Introducir el número de IP asociada al robot.
  2. RF.1.2 Introducir el número de puerto.
  3. RF.1.3 Botón de play.
- RF.2 Recogida de datos importantes con actualización constante.
  1. RF.2.1 Calibración Nova
  2. RF.2.2 Recibir datos de posición de Oculus Quest 2.
  3. RF.2.3 Recibir datos de rotación de Oculus Quest 2.
  4. RF.2.4 Recibir datos de sensores de Nova.
- RF.3 Adaptación y Aplicación de datos con actualización constante.
  1. RF.3.1 Linkeo de manos con Nova en la escena.
  2. RF.3.2 Creación de mensaje de posición predeterminada(casa).
  3. RF.3.3 Creación de mensajes ROS de posición y rotación.
  4. RF.3.4 Creación de mensajes ROS de apertura de la pinza.
- RF.4 Envío de mensajes ROS al robot.
  1. RF.4.1 Envío de mensaje de posición y rotación.
  2. RF.4.2 Envío de mensaje de apertura.
- RF.5 Finalizar ejecución.
  - RF.5.1 Botón de finalizar conexión.

Requisitos No Funcionales

A continuación veremos los requisitos no funcionales de lo desarrollado, que se refieren a todos aquellos requisitos que describen características de funcionamiento y no las tareas que realiza o la información que guarda.

RNF.1 - Eficiencia: El uso del software desarrollado tiene que ser eficiente, que no provoque errores y se ejecute cumpliendo con los requisitos funcionales en tiempos lógicos(sin desarrollar grandes delays).

RNF.2 - Escalabilidad: Este proyecto sirve como cimiento o base de uno de mayores proporciones, por lo que nuestro software tiene que poder ser escalable. Ya que gracias a él se podrán ver otro tipo de softwares.

RNF.3 - Rendimiento: El resultado de este proyecto necesita de un alto rendimiento a la hora de ejecutarlo ya que de no ser así, la experiencia del usuario se vería afectada y el software perdería valor.

RNF.4 - Portabilidad: Es un requisito indispensable de este proyecto, ya que cumpliendo con ello se podrá realizar una fácil implementación junto a otros tipos de robot o de dispositivos.

B.4. Especificación de requisitos

1. Conexión entre el robot ROS y el ordenador.	
Requisito	1.1 - Introducir el número de IP asociada al robot.
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se introduce el número de IP asociado.
Descripción	El usuario de esta aplicación debe introducir el número de IP asociada al robot.
Precondición	Hallar el número de dirección IP.
Postcondiciones	A la espera del número de puerto.
Excepciones	Ninguna.

Tabla B.1: 1.1 Introducir el número de IP asociada al robot.

Requisito	1.2 - Introducir el número de puerto.
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se introduce el número de puerto.
Descripción	El usuario de esta aplicación debe introducir el número de puerto asociada al robot.
Precondición	Hallar el número de puerto.
Postcondiciones	Ninguna.
Excepciones	Ninguna.

Tabla B.2: 1.2 - Introducir el número de puerto.

Requisito	1.3 - Botón de play.
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se pulsa el botón de play..
Descripción	El usuario de esta aplicación debe pulsar el botón de play para ejecutar.
Precondición	Introducir IP y número de puerto.
Postcondiciones	Ninguna.
Excepciones	Ninguna.

Tabla B.3: 1.3 - Botón de play.

2. Recogida de datos importantes con actualización constante.	
Requisito	2.1 - Calibración Nova
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se calibra el guante háptico Nova.
Descripción	Al comenzar la ejecución se realiza una calibración de los guantes Nova.
Precondición	Ejecución y tener colocado el guante.
Postcondiciones	Ninguna.
Excepciones	Ninguna.

Tabla B.4: 2.1 - Calibración Nova

Requisito	2.2 Recibir datos de posición de Oculus Quest 2
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se recogen los datos de posición de Oculus Quest 2.
Descripción	Se recogen los datos referidos a la posición de los controladores de Oculus, para poder representar correctamente en el espacio nuestras manos.
Precondición	Tener las gafas cerca y estar dentro de la zona de juego de Oculus.
Postcondiciones	Aplicar los datos.
Excepciones	Ninguna.

Tabla B.5: 2.2 Recibir datos de posición de Oculus Quest 2

Requisito	2.3 Recibir datos de rotación de Oculus Quest 2
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se recogen los datos de rotación de Oculus Quest 2.
Descripción	Se recogen los datos referidos a la rotación de los controladores de Oculus, para poder representar correctamente en el espacio nuestras manos.
Precondición	Tener las gafas cerca y estar dentro de la zona de juego de Oculus.
Postcondiciones	Aplicar los datos.
Excepciones	Ninguna.

Tabla B.6: 2.3 Recibir datos de rotación de Oculus Quest 2

Requisito	2.4 Recibir datos de sensores de Nova
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se recogen los datos de los sensores de Nova
Descripción	Se recogen los datos necesarios de los sensores de los guantes hápticos Nova. Datos de flexión de los dedos.
Precondición	Tener los guantes puestos y calibrados.
Postcondiciones	Aplicar los datos.
Excepciones	Ninguna.

Tabla B.7: 2.4 Recibir datos de sensores de Nova

3. Adaptación y Aplicación de datos con actualización constante	
Requisito	3.1 Linkeo de manos con Nova en la escena
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se representan nuestras manos con los datos de los sensores
Descripción	Se representan nuestras manos en el espacio dentro de la escena de Unity, de manera realista y con actualización constante.
Precondición	Tener los guantes puestos y calibrados con los controladores montados.
Postcondiciones	Transmitir al robot.
Excepciones	Ninguna.

Tabla B.8: 3.1 Linkeo de manos con Nova en la escena

Requisito	3.2 Creación de mensaje de posición predeterminada(casa).
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se crea un mensaje de posición predeterminada.
Descripción	Se crea un mensaje que llevará la pinza de nuestro brazo robótico al punto del espacio que hemos seleccionado como predeterminado.
Precondición	Tener conexión con el robot.
Postcondiciones	Envíar mensajes.
Excepciones	Ninguna.

Tabla B.9: 3.2 Creación de mensaje de posición predeterminada(casa).

Requisito	3.3 Creación de mensajes ROS de posición y rotación.
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se crean mensajes de posición y rotación para nuestro robot.
Descripción	Se crean mensajes que nuestro robot interpreta y que llevará la pinza de nuestro brazo robótico al punto del espacio y con la rotación que le indiquemos con nuestro guante.
Precondición	Tener conexión con el robot, los guantes puestos y las gafas conectadas para seguir nuestras manos.
Postcondiciones	Envíar mensajes.
Excepciones	Ninguna.

Tabla B.10: 3.3 Creación de mensajes ROS de posición y rotación.

Requisito	3.4 Creación de mensajes ROS de apertura de la pinza.
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se crean mensajes de apertura de la pinza para nuestro robot.
Descripción	Se crean mensajes que nuestro robot interpreta y que abrirá o cerrará la pinza del robot según le indiquemos con nuestra mano.
Precondición	Tener conexión con el robot, los guantes puestos y las gafas conectadas para seguir nuestras manos.
Postcondiciones	Envíar mensajes.
Excepciones	Ninguna.

Tabla B.11: 3.4 Creación de mensajes ROS de apertura de la pinza.

4. Envío de mensajes ROS al robot	
Requisito	4.1 Envío de mensajes de posición y rotación
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se envían los mensajes de posición y rotación de la pinza para nuestro robot.
Descripción	Se envían los mensajes que nuestro robot interpretará y que colocará la pinza del robot en el espacio según le indiquemos con nuestra mano.
Precondición	Creación de mensajes actualizados.
Postcondiciones	Ninguna.
Excepciones	Ninguna.

Tabla B.12: 4.1 Envío de mensaje de posición y rotación

Requisito	4.2 Envío de mensajes de apertura
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se envían los mensajes de apertura de la pinza para nuestro robot.
Descripción	Se envían los mensajes que nuestro robot interpretará y que abrirá o cerrará la pinza del robot según le indiquemos con nuestra mano.
Precondición	Creación de mensajes actualizados.
Postcondiciones	Ninguna.
Excepciones	Ninguna.

Tabla B.13: 4.2 Envío de mensajes de apertura

5. Finalizar ejecución	
Requisito	5.1 Botón de finalizar conexión.
Autor	Víctor de la Iglesia García
Versión	1.0
Accion/es	Se pulsa el botón que finaliza la ejecución.
Descripción	Se finalizan todos los procesos y se termina la conexión con el robot.
Precondición	Ninguno.
Postcondiciones	Ninguna.
Excepciones	Ninguna.

Tabla B.14: 5.1 Botón de finalizar conexión.

Apéndice C

Especificación de diseño

C.1. Introducción

A lo largo de este punto veremos la especificación de diseño, donde veremos en profundidad que tipo de datos maneja y trata nuestra aplicación.

C.2. Diseño de datos

Aquí veremos qué datos se recogen, se generan y se aplican en nuestro software.

Los tipos datos de entrada que son utilizados en esta aplicación podríamos dividirlos en dos debido a su naturaleza, teniendo en cuenta que ambos son recogidos por los dispositivos de realidad virtual [12] de los que disponemos.

Datos de Entrada

- **Coordenadas de posición:** El tipo de datos más importante dentro de nuestro software es aquel referente a las coordenadas de posición que recoge Unity a través de nuestros controladores y las gafas de Oculus. Son coordenadas englobadas en un tipo de Unity que es un vector que tiene tres valores, ya que representan las tres dimensiones del espacio X Y Z. Como hemos visto previamente en este documento y en la memoria, este tipo de datos es el que nos da la posibilidad de representar correctamente nuestras manos en el espacio tridimensional, además de ser indispensable para poder enviar las instrucciones de posición necesarias a nuestro robot.

Estos datos se recogen gracias a las gafas que a través de sus cámaras y sensores permite conocer en que punto de la zona establecida para usar Oculus se encuentran los controladores.

Estos datos una vez recogidos, se procesan de manera que cumplan con el estándar establecido por mi previamente en búsqueda de una experiencia del usuario correcta.

- **Datos de flexión y aducción**: Este tipo de datos es el que nos dará la posibilidad de aplicar un cierre o apertura determinado a nuestra pinza robótica. Se recogen directamente de los sensores dispuestos en el guante háptico y reflejan en un vector de tres partes los valores de pronación/supinación, flexión/extensión y adducción/abudcción de los dedos.

  Los que utilizamos principalmente son los de flexión/extensión, ya que gracias a ellos podemos definir correctamente un comportamiento para la aplicación asociado a flexionar índice y pulgar simulando la pinza.

  Estos datos se recogen garcias al script *GettingData* y se pasan en forma de mensajes(instrucciones) para la pinza y así abrir o cerrar la pinza.

Datos Generados

Los datos que se generan en esta aplicación son mensajes ROS que se publican en los topics determinados, para hacer al robot cumplir con la tarea determinada. En este caso se generan dos tipos de mensajes específicos, unos referentes a la posición y rotación de la pinza y otros en relación con la apertura/cierre de la misma.

Estos mensajes se generan de manera constante ya que sus funciones se ejecutan constántemente mientras la aplicación esté funcionando. De esta manera conseguimos un resultado óptimo sin demasiado delay.

C.3. Diseño arquitectónico

En cuanto a la arquitectura dentro de este proyecto cabe destacar, que aunque el software final es una recopilación de objetos y técnicas utilizadas a lo largo del proyecto, hasta llegar a ese punto ha sido necesario del desarrollo de varias escenas para el aprendizaje y el análisis de cómo se gestionan o recogen los datos.

Cada una de esas escenas fue llevándose a cabo para cumplir pequeñas tareas y utilizando scripts que nos permitirían en un futuro, alcanzar el resultado que tenemos.

En el siguiente diagrama podemos ver la arquitectura del software final, donde cada una de las funciones o procedimientos que vemos están realizadas por scripts desarrollados:

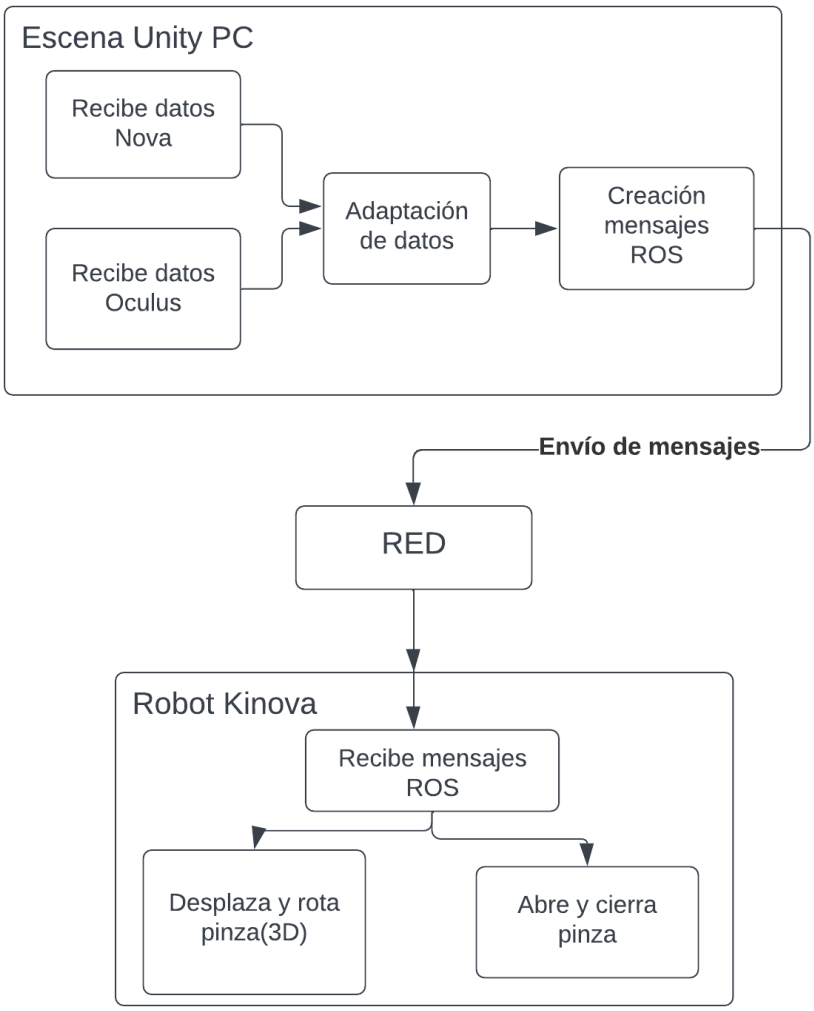

Figura C.1: Diagrama de la arquitectura del software final

Apéndice D

Documentación técnica de programación

D.1. Introducción

En esta parte del anexo pasaremos a ver de qué manera se halla organizado el proyecto con su estructura de directorios y archivos, también veremos todo aquello fundamental para la instalación del proyecto y poder ejecutarlo o trabajar con ello. También se verá la manera en qué se compila y se ejecuta, todo ello con documentación gráfica sobre los procesos.

A parte de aquellos requerimientos que sean especificados a lo largo de este apéndice, también será fundamental para usarlo o trabajar con ello conocimientos específicos tanto de Unity como de C#.

D.2. Estructura de directorios

El directorio del proyecto es el que se muestra a continuación, aunque pueda contener variaciones debido a temas de confidencialidad:

- **Proyecto/Assets** En esta carpeta se hayan contenidos todos los ficheros relacionados con las escenas, scripts y plugins añadidos.
    - **/Assets/Art** Carpeta que contiene el arte utilizado en el proyecto.
    - **/Assets/MyScripts** Carpeta que contiene los Scripts desarrollados para el proyecto.

    - **/Assets/Oculus** Carpeta derivada de la descarga del plugin de Oculus para Unity.
    - **/Assets/Prefabs** Carpeta que contienen objetos de las escenas.
    - **/Assets/Resources**
    - **/Assets/RobotConnector** Carpeta que contiene Scripts para el trabajo con el robot.
    - **/Assets/ROSBridgeLib-master** Carpeta con librerías que sirven de puente entre ROS y Unity.
    - **/Assets/Scenes** Carpeta que contiene las escenas desarrolladas por mi en el proyecto. Es la carpeta que contiene desde los pequeños resultados que se iban consiguiendo, hasta la escena *Ros* con el software final.
    - **/Assets/SenseGlove** Carpeta que contiene las librerías para el uso de SenseGlove Nova en Unity.
    - **/Assets/TextMesh Pro** Carpeta derivada de la instalación del plugin TextMeshPro.
    - **/Assets/XR** Carpeta derivada de la instalación del plugin XR.
- **Proyecto/Library** En esta carpeta se hayan contenidas las librerías de las que dispone Unity.
- **Proyecto/Logs** En esta carpeta se contienen logs(registros).
- **Proyecto/obj**
- **Proyecto/Packages** Carpeta que contiene datos sobre los paquetes instalados en Unity.
- **Proyecto/ProjectSettings** Carpeta que contiene información sobre la configuración de proyecto.
- **Proyecto/Temp** Carpeta que contiene archivos temporales.
- **Proyecto/UserSettings** Carpeta que contiene configuración del usuario.

D.3. Manual del programador

En esta sección prepararemos nuestro entorno de cara a poder compilar, instalar y ejecutar el proyecto en sí. Para ello veremos qué instalaciones son requeridas para el correcto trabajo, los pasos necesarios y aquellos requerimientos de PC destacados. En primer lugar vemos los requisitos mínimos y recomendados de Unity para ser instalado y que funcione correctamente:

	Requisitos mínimos	Requisitos recomendados
Sistema operativo	Windows 7(SP1+), windows 10. versiones de 64 bits	Windows 11 64 bit
CPU	Arquitectura x64 con instrucciones sse2 (Intel Core 2 Duo, Extreme o Quad, Intel Core i3, AMD Sempron, Phenom I y II y Athlon II)	Arquitectura x64 con instrucciones sse2 (Intel core i5, i7, AMD FX Y Ryzen)
API de graficos	DX 10 Y 11	DX 12
Requisitos adicionales	Drivers de hardware compatibles	Drivers de hardware compatibles
Graficos	Tarjeta de video de 512 mb (GTX 650 en adelante)	RTX 3070(para proyectos 3D)
RAM	Desde 4Gb	16Gb
Espacio en disco	10Gb	10gb

Figura D.1: Requisitos mínimos y recomendados Unity [3]

Instalando Unity

Para comenzar con la instalación es necesario acceder a su página web, donde debemos registrarnos y seleccionar el tipo de cuenta por la licencia de uso y se nos descargará el ejecutable UnityHub en su última versión (La 3.1.2 actualmente).

Deberemos de seleccionar el lugar de instalación de este programa que nos permitirá gestionar nuestros proyectos. Después, se nos abrirá una ventana del Unity Hub donde en el apartado de Projects podremos en un futuro ver qué proyectos tenemos, vamos a la pestaña de instalaciones y veremos lo siguiente:

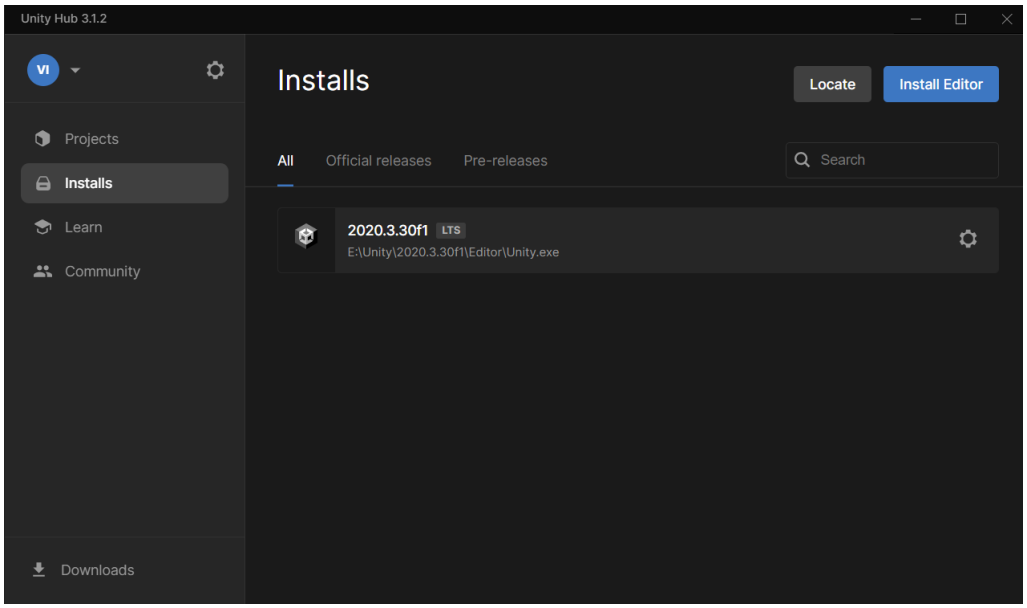

Figura D.2: Instalación Editor Unity 1

Llegados a este punto seleccionando en instalar editor nos aparecerá la siguiente ventana que nos permitirá seleccionar la versión del editor que queramos utilizar. Recomiendo para el uso de este proyecto la versión 2020.3.30f1 ya que es con el que ha sido desarrollado y el cambio de versiones de editor puede acarrear fallos o problemas inesperados.

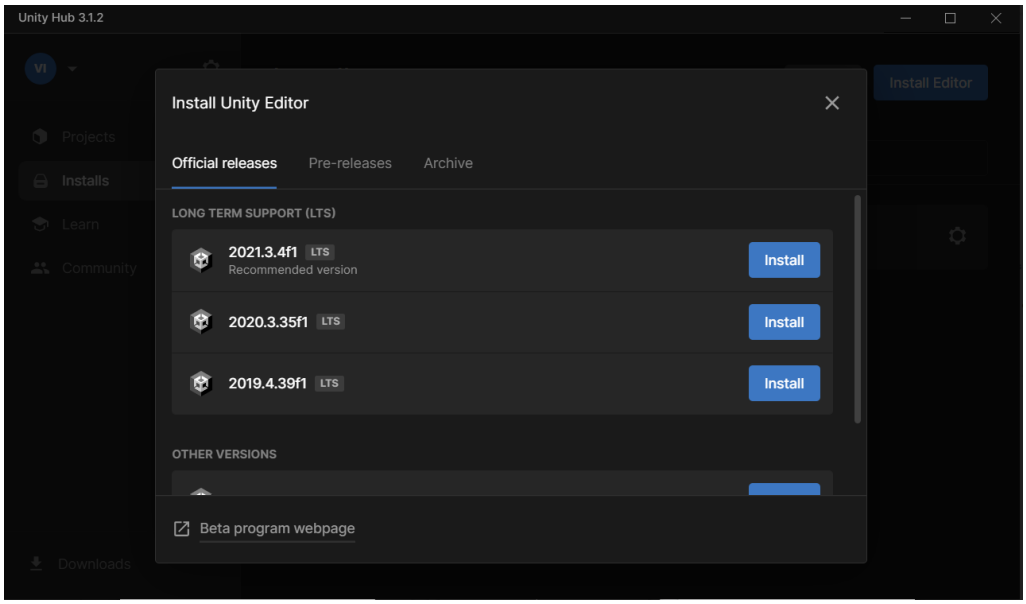

Figura D.3: Instalación Editor Unity 2

Componentes Software

Para poder hacer uso de nuestros dispositivos de realidad virtual necesitaremos del software especializado requerido. Por ello aquí veremos qué necesitamos instalar en nuestro ordenador.

- **SenseCom**: Este software es el necesario de instalar para poder utilizar nuestros guantes hápticos, para ello nos dirigiremos a la página web de *senseglove.com* y accederemos al apartado de developers. Desde ese apartado podremos adentrarnos en el proyecto de GitHub SenseGlove-API, el cual contiene un ejecutable que instalará este software.

  Se muestra a continuación una imagen del contenido de lo descargado junto a la pantalla de instalación, tras ejecutar el instalador.

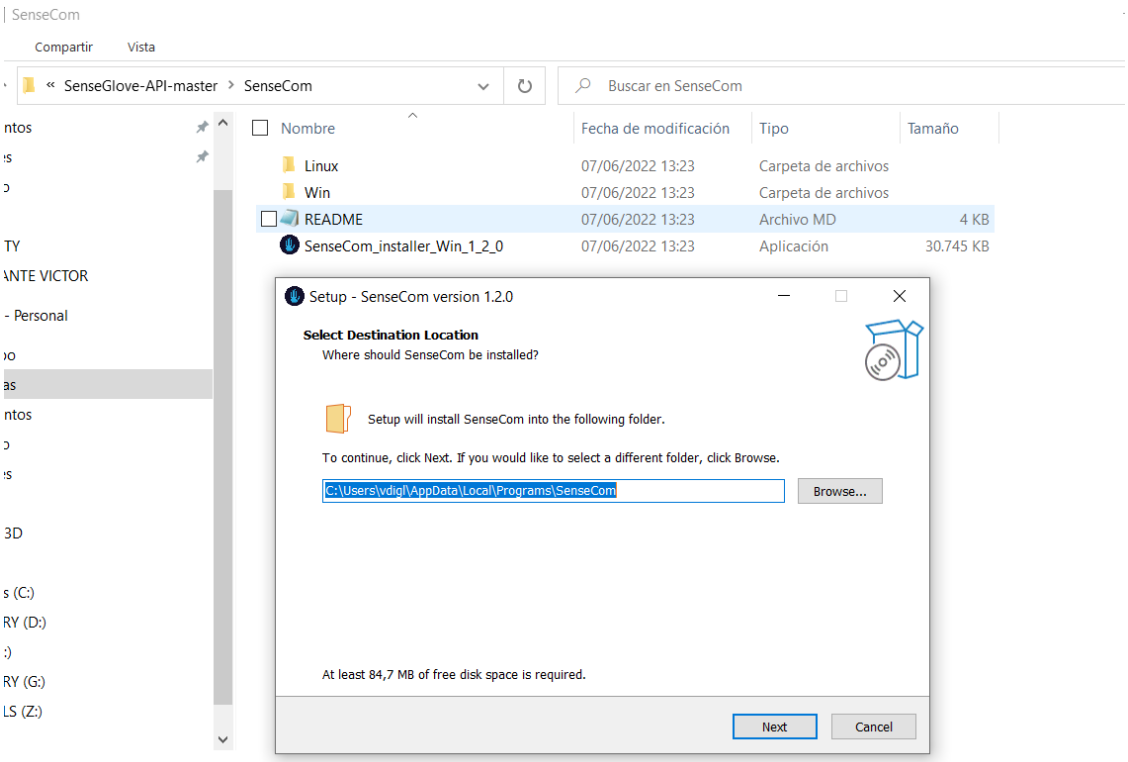

Figura D.4: Instalación Software SenseGlove

- **Oculus**: El siguiente software que debemos instalar en nuestro PC es el de Oculus[1] ya que sin el, no podremos conectar nuestro dispositivo Oculus Quest 2 ni sus controladores a Unity. Para esta instalación, accederemos a la siguiente página web => *https://store.facebook.com/es/quest/setup/* y descargaremos el software.

  En este punto se nos descargará un archivo *OculusSetup.exe*, el cual debemos ejecutar, y tras revisar los términos y condiciones nos aparecerá la siguiente pantalla:

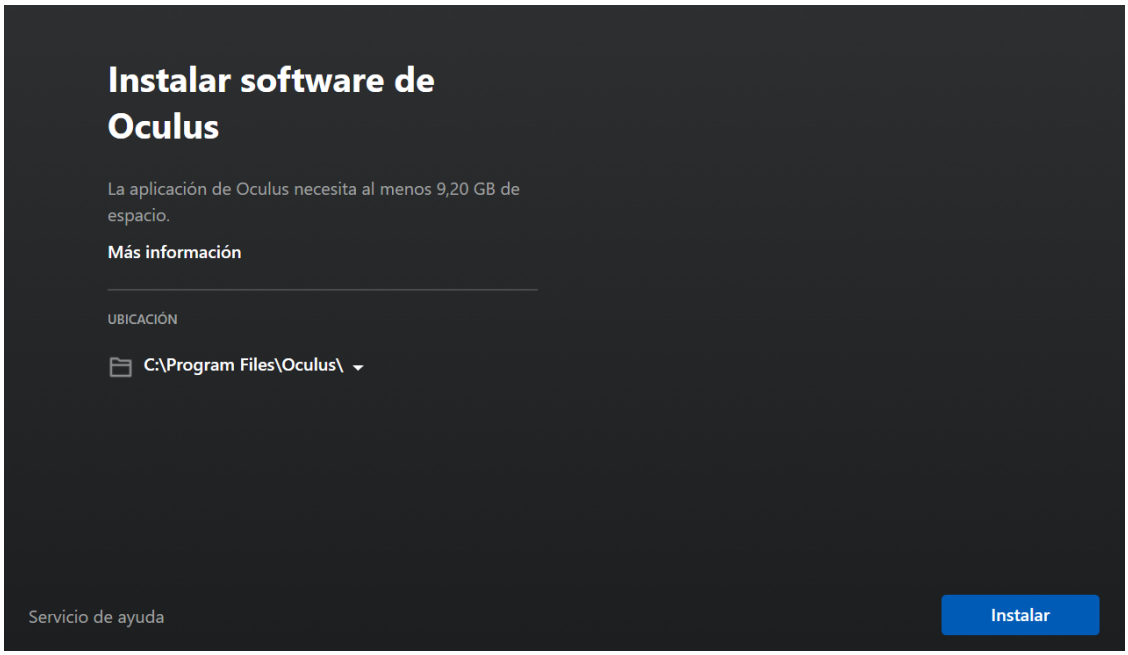

Figura D.5: Instalación Software Oculus

Tras realizar la instalación se nos requerirá de la creación de una cuenta que puede ser de *Facebook* y posteriormente a todo este proceso, ya podremos configurar nuestro hardware y simplemente con tener las gafas conectadas al PC podremos usar los dispositivos desde Unity.

- **IP Advanced Scanner**: Para descargar este software, que nos servirá de gran ayuda a la hora de preparar la conexión con nuestro robot debemos acceder a la web oficial que es la siguiente=> *https://www.advanced-ip-scanner.com/es/* y realizar la descarga. Este software está configurado para poder instalarse en el ordenador o para ser ejecutado de manera única con su versión portátil.

  Al correr el ejecutable se nos abre la siguiente ventana que nos permite elegir:

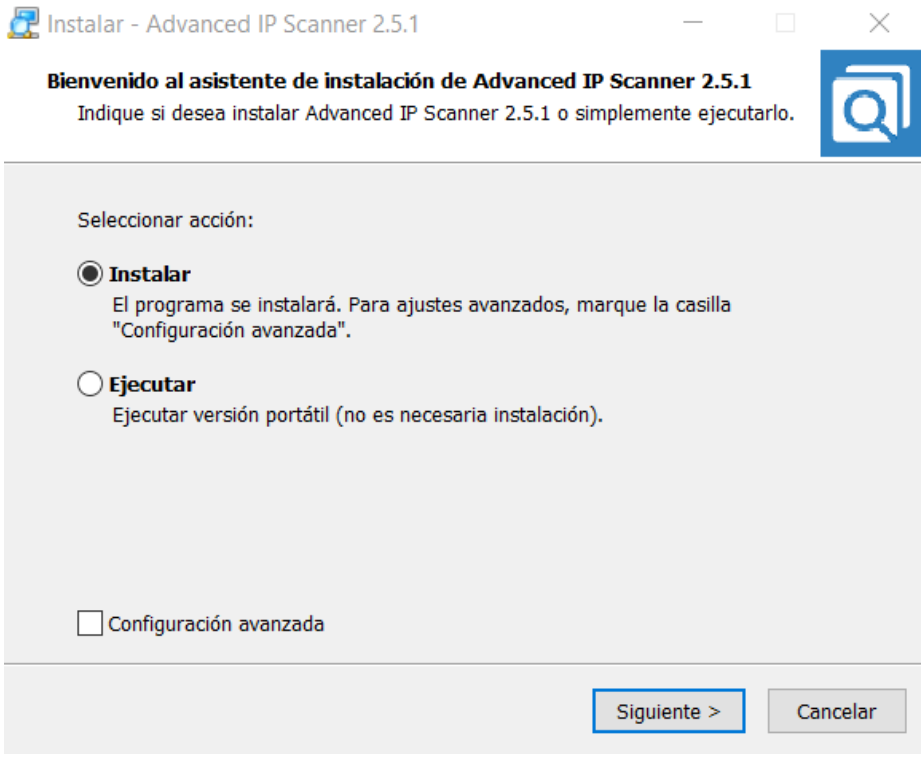

Figura D.6: Instalación Escáner de RED

Una vez aquí podemos seleccionar la opción que prefiramos, en caso de de instalar, seleccionaremos la ruta de destino de la instalación y ya tendríamos este software.

D.4. Compilación, instalación y ejecución del proyecto

Llegados a este punto del anexo, en caso de querer utilizar el proyecto, deberíamos tener instaladas ya las aplicaciones comentadas previamente. Ahora bien, comenzaremos con la compilación, instalación y forma de ejecutar este proyecto.

En primer lugar tendremos que comprobar tres cosas:

1. Comprobar con SenseCom que tenemos conexión con los guantes, para ello debemos de tenerlos encendidos y añadidos a dispositivos bluetooth de nuestro PC. Tras esto, al abrir el software nos saldrá en pantalla que se encuentran conectados y veremos una representación de nuestras manos.

2. Comprobar la conexión de Oculus con su software, se puede conectar vía WIFI o lo recomendado por cable, ya que la conexión es directa.

3. Comprobar con el escáner de red instalado, la dirección IP del robot.

Los siguientes pasos son ya relacionados con Unity.

Unity

Abrimos Unity Hub y seleccionaremos el botón de *Open* ya que es el que nos permite añadir proyectos a nuestro Unity.

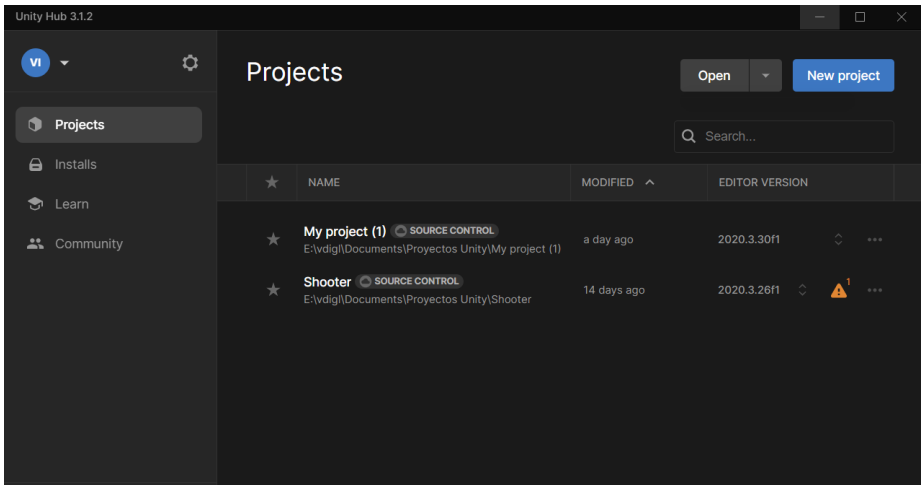

Figura D.7: Importar Proyecto

Después de esto se nos abrirá la clásica ventana de nuestro explorador que nos permitirá llegar al punto donde tengamos la carpeta del proyecto, para así poder abrirlo e importarlo. Deberemos seleccionar de las versiones del editor que tengamos instaladas la que queramos utilizar para este proyecto. La recomendada por mi es la *2020.3.30f1*

Recuerdo de nuevo que se recomienda siempre utilizar la versión del editor correspondiente a la de quién realizó el proyecto, ya que pueden aparecer errores o problemas de importación.

Una vez tengamos todo importado, en la parte baja de la pantalla aparece el explorador de Unity y accedemos dentro de la carpeta *Assets* a la carpeta de nombre *Scenes*. Posteriormente debemos seleccionar la escena *Ros* ya que es el software final del proyecto.

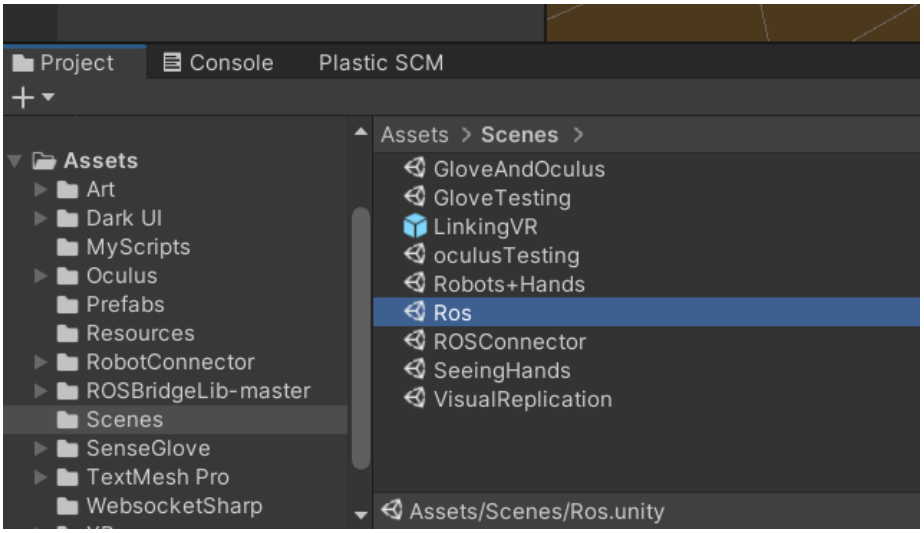

Figura D.8: Explorador archivos de proyecto

Una vez abierta esta escena en la parte superior izquierda de la pantalla, nos aparecerá una jerarquía con los *GameObjects*[8] de nuestra escena. Debemos de seleccionar *ROSInitializer* para configurar la conexión. En el lado derecho en el *Inspector* nos aparecerá el siguiente cuadro:

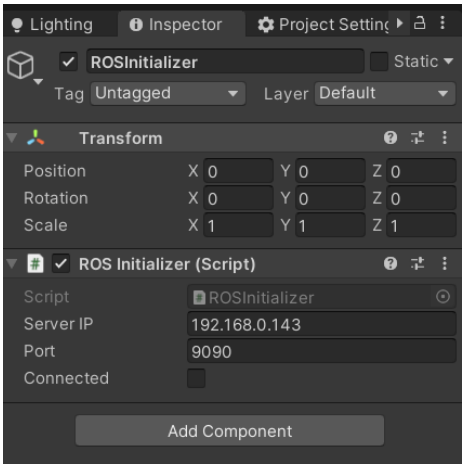

Figura D.9: Configuración de conexión con Robot

En este cuadro tendremos que rellenar con el número de IP y el puerto para que a la hora de ejecutar sea capaz de conectar con el robot. El resto de los valores asociados a cada variable o componente de los *GameObjects* de esta escena se deben de mantener igual.

Con todo esto ya tendremos listo el software para llevar a cabo la *build* y empezar a usarlo. Para ello en la barra superior de herramientas accedemos a => *File->Build Settings* y se nos mostrará la siguiente ventana:

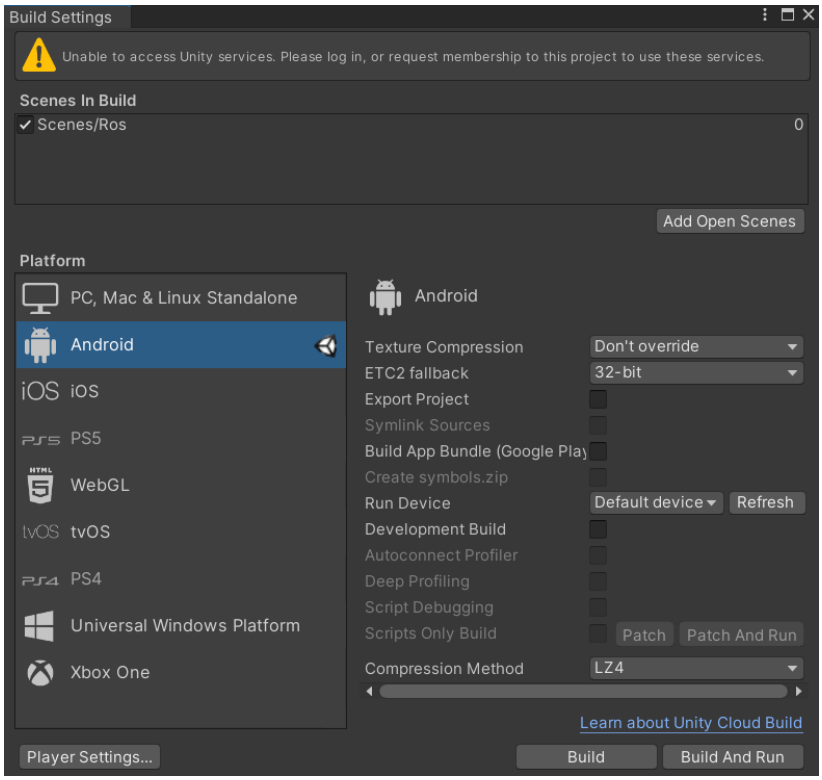

Figura D.10: Realizando *Build* del proyecto

En el apartado *Run Device* debemos o dejarlo en dispositivo por defecto o seleccionar el dispositivo Android VR[12] (Las Oculus Quest 2 en mi caso) que queramos utilizar con Unity. Seleccionamos *Build* y se abrirá un explorador de archivos que nos permitirá alojar nuestro archivo *.apk* que correremos en el dispositivo Android.

En caso correcto, en la carpeta de destino seleccionada debemos ver el siguiente fichero .apk con el nombre seleccionado (En este caso *Build1.apk*).

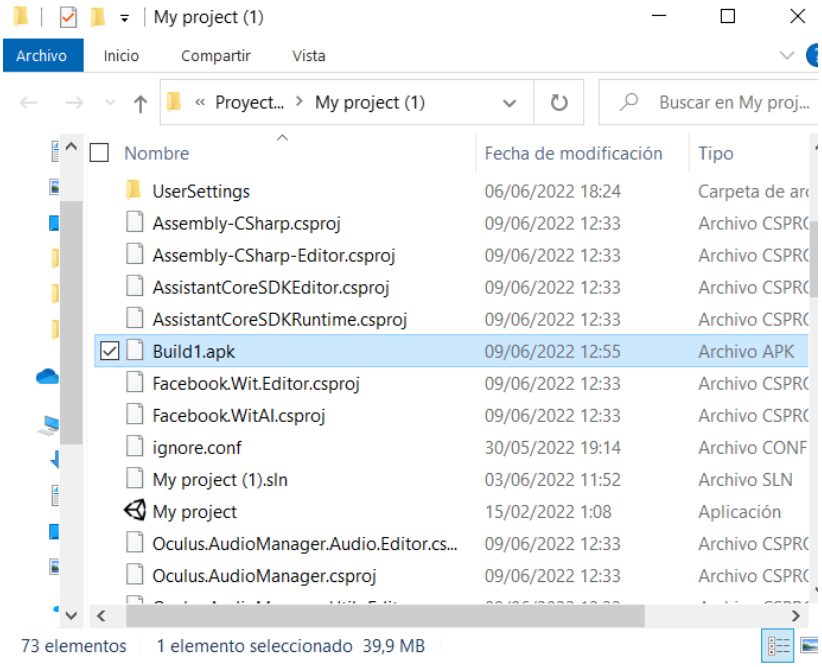

Figura D.11: Realizando *Build* resultante de proyecto

A partir de aquí, si tenemos conectados todos los dispositivos, solo nos quedaría pulsar el botón *Run/Play* de Unity para que se ejecute el proyecto y comenzar a trabajar junto al robot y demás dispositivos.

Apéndice E

Documentación de usuario

E.1. Introducción

Llegados al fin del proyecto quedaría como resultado para el usuario, una aplicación software que desde Unity con unas gafas como las Oculus Quest 2 se nos permita hacer uso de un robot mediante teleoperación.

E.2. Requisitos de usuarios

Para poder hacer uso del software desarrollado necesitamos fundamentalmente dos dispositivos hardware relacionados con la realidad virtual distintos que son las Oculus Quest 2 y los Nova de SenseGlove, un robot con ROS como el Kinova 6DOF y un ordenador con Unity. Para poder instalar y ejecutar correctamente Unity los requisitos son los mismos que los expuestos previamente:

	Requisitos mínimos	Requisitos recomendados
Sistema operativo	Windows 7(SP1+), windows 10. versiones de 64 bits	Windows 11 64 bit
CPU	Arquitectura x64 con instrucciones sse2 (Intel Core 2 Duo, Extreme o Quad, Intel Core i3, AMD Sempron, Phenom I y II y Athlon II)	Arquitectura x64 con instrucciones sse2 (Intel core i5, i7, AMD FX Y Ryzen)
API de graficos	DX 10 Y 11	DX 12
Requisitos adicionales	Drivers de hardware compatibles	Drivers de hardware compatibles
Graficos	Tarjeta de video de 512 mb (GTX 650 en adelante)	RTX 3070(para proyectos 3D)
RAM	Desde 4Gb	16Gb
Espacio en disco	10Gb	10gb

Figura E.1: Requisitos minimos y recomendados Unity

E.3. Instalación

En cuanto a los pasos para completar la instalación del proyecto en el ordenador de un usuario, son los mismos que los expuestos en el previo apartado de *Compilación, instalación y ejecución del proyecto*

E.4. Manual del usuario

Tras la debida preparación en apartados anteriores de nuestos dispositivos, una vez tengamos todos calibrados y con conexión con el PC o el robot, lo único que tenemos que hacer es ejecutar el software y aplicar sobre nuestra mano los debidos desplazamientos deseados para mover la pinza o gestualizar el cierre de la misma para controlar su apertura.

Bibliografía

[1]

[2] We are Marketing. Scrum. https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html, 2022.

[3] FortalezaGamer. Requerimientos unity. https://www.lafortalezagamer.online/unity-que-es-requisitos-minimos-peso-en-disco-pros-y-contras/, 2022.

[4] itcl. Itcl. https://itcl.es/, 2022.

[5] Wiki ROS. https://wiki.ros.org/es/ROS/Introduccion.

[6] SenseGlove. Nova. https://www.senseglove.com/product/nova/, 2022.

[7] Unity. Componentes. https://docs.unity3d.com/es/530/Manual/Components.html, 2022.

[8] Unity. Gameobjects. https://docs.unity3d.com/es/2018.4/Manual/GameObjects.html, 2022.

[9] Unity. Unity. https://unity.com/es, 2022.

[10] Universidad ORT Uruguay. Robotica. https://fi.ort.edu.uy/blog/que-es-la-robotica-y-cuales-son-sus-usos, 2022.

[11] Wikipedia. Háptica. https://en.wikipedia.org/wiki/Haptic_technology, 2022.

[12] Wikipedia. Virtual reality. https://es.wikipedia.org/wiki/Realidad_virtual, 2022.