Escuela Técnica Superior de Ingeniería de Telecomunicación

Universidad Politécnica de Cartagena

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de un guante háptico para entornos virtuales

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

Autor:
Alejandro Jorge López

Directores:
María Francisca Rosique Contreras
Ramón Ruíz Merino

Cartagena, Julio 2019

A Paqui, Ramón y David, por toda la ayuda prestada en la realización de este proyecto.

A Edu y Pedrolo, por cada momento vivido a vuestro lado. Espero que la vida nos vuelva a cruzar muy pronto.

A Laura y Alex, mi familia de Madrid, por haberme ayudado y apoyado tanto como lo han hecho en esta nueva aventura. Nada es lo mismo sin vosotros.

A mis padres, por ayudarme a estudiar mi pasión, por confiar en mí cuando ni yo podía y por mostrarme lo que es querer. Todo lo bueno que tengo os lo debo a vosotros.

A mi abuelo, por haberme enseñado todo lo que sé y por ser ejemplo e inspiración para mí. Eres la persona más fuerte que conozco y todo en lo que quiero convertirme algún día.

A mi hermano, por estar a mi lado a pesar de separarnos kilómetros de distancia y por ser la persona que más quiero en este mundo. Estoy muy orgulloso de ti.

Autor	Alejandro Jorge López
Correo electrónico	Alejandro.jorge.lopez.jl@gmail.com
Directores	María Francisca Rosique Contreras, Ramón Ruíz Merino
Correo electrónico	paqui.rosique@upct.es , ramon.ruiz@upct.es
Título del TFG	Desarrollo de un guante háptico para entornos virtuales.

Resumen

Desarrollar un guante háptico de bajo coste haciendo uso de una placa Arduino y un entorno virtual con Unity.

Titulación	Grado en Ingeniería en Sistemas de Telecomunicación
Fecha de presentación	Julio - 2019

Índice

Índice de figuras

Índice de tablas

1 Resumen

En el presente proyecto se ha desarrollado un guante capaz de aumentar la inmersión de un usuario en entornos virtuales, utilizando la tecnología háptica para estimular el sentido del tacto. Esto se consigue gracias a la actuación de una serie de motores de vibración colocados en la punta de los dedos para simular el tacto que tendrán los objetos tocados en la escena virtual, tanto en forma como presión.

Se han implementado el diseño y las pruebas requeridas para el desarrollo del prototipo del guante háptico, así como la justificación de los componentes utilizados para el diseño de un prototipo de bajo coste que cumpla con las especificaciones requeridas y que a su vez asegure que el prototipo sea lo más ergonómico e intuitivo posible para el usuario.

Además, se ha añadido un esquema de la disposición de los elementos en el diseño del guante háptico, detallando la posición exacta de los sensores Flex, utilizados para la medida de la flexión ejercida por los dedos, y de los motores vibradores, que serán los actuadores en el guante para lograr el estímulo requerido.

Para la puesta en práctica de este prototipo se ha hecho uso del motor de videojuegos multiplataforma Unity para crear una escena con diferentes objetos que poder tocar y sentir. La escena se encuentra diseñada para una visualización por pantalla, pero es completamente exportable a una escena en realidad virtual inmersiva, pudiendo lograr mejores resultados si se integra la solución de la tecnología háptica con otros dispositivos de realidad virtual.

Para el posicionamiento de las manos en el entorno virtual, hemos hecho uso del dispositivo Leap Motion de captura de movimiento por medio de procesamiento de imagen, capaz de replicar la posición de las manos humanas a partir de las imágenes captadas por el dispositivo. Este dispositivo aporta datos de mayor calidad y fiabilidad que el resto de dispositivos similares al mismo. Sin embargo, para garantizar el correcto funcionamiento del prototipo, se han integrado los sensores Flex comentados anteriormente, dotando así de una mayor precisión.

A pesar de que el objetivo de este proyecto es ser capaces de identificar objetos virtuales por medio de estímulos artificiales haciendo uso del tacto de nuestras manos, el propósito del proyecto ha sido el estudio de ambas tecnologías, la háptica y la de captura de movimiento, para integrar ambas y lograr un prototipo funcional que las aunase.

Sin embargo, aunque el proyecto está enfocado a la identificación de objetos, esta aplicación es fácilmente extrapolable para lograr recibir cualquier tipo de sensación por medio de nuestras manos, consiguiendo así implementar una interfaz humano-máquina que nos aporte más datos con el uso de nuestros diferentes sentidos y conseguir así una realimentación por parte de las simulaciones o aplicaciones que el usuario utilice de mayor calidad y con mayor cantidad de información.

2 Introducción

2.1 Introducción

En los últimos años se ha visto una gran evolución de los llamados dispositivos que utilizan la tecnología háptica para permitir al usuario recibir estímulos artificiales. Estos dispositivos se basan en la manipulación del sentido del tacto por medio de diversas tecnologías para, por ejemplo, sentir los objetos alojados en un entorno virtual, siendo capaces de captar su forma, textura o densidad, añadiendo así el tacto a la experiencia con entornos virtuales.

Es decir, estos dispositivos permiten una realimentación artificial al usuario con el fin de obtener más información y, sobre todo, lograr una mayor inmersión en las escenas virtuales, la cual se verá incrementada con la utilización del resto de los sentidos en la experiencia.

Este tipo de realimentación táctil puede darse en diversas partes del cuerpo. Una de las partes a las que más comúnmente se le aplica la tecnología háptica son las manos por la movilidad que tienen frente al resto de partes del cuerpo, haciendo uso de los llamados guantes hápticos.

La mayor parte de las aplicaciones de los guantes hápticos se encuentran en la industria del videojuego, donde la experiencia inmersiva es altamente valorada para conseguir una experiencia mayor.

Sin embargo, también se utiliza la tecnología háptica en diversos campos de la investigación, la medicina o la educación, campos en los cuales se requiere un mayor desarrollo para asegurar la fiabilidad y eficiencia de los dispositivos hápticos.

Si bien es cierto que, actualmente, no se ha conseguido un guante háptico que replique exactamente el sentido del tacto captado en la palma de las manos, la implementación de tecnologías como la vibratoria o el uso de electroimanes nos acerca cada vez más al objetivo buscado: lograr reproducir con total fidelidad el sentido del tacto.

2.2 Motivación y objetivos

La naturaleza y objetivo último de un sistema de realidad virtual es el de representar un entorno artificial de la manera más fidedigna y creíble posible. Por otra parte, sabemos que los sentidos humanos son la herramienta de nuestro cuerpo para percibir, pensar y evaluar el mundo. Por tanto, la integración de los diversos sentidos del cuerpo en los entornos virtuales aportará una mayor inmersión del usuario, así como una construcción pragmática de los entornos virtuales, consiguiendo, además, la creación de entornos y experiencias más familiares que atenúen el “rechazo” natural hacia lo artificial.

Actualmente, el desarrollo de dispositivos que nos posibilitan manipular el sentido de la vista por medio de la inmersión con cascos o gafas que nos permiten ver los entornos

virtuales de los que hacemos uso, se encuentra en auge y con resultados bastante optimizados de inmersión dada la capacidad de controlar lo que vemos con el movimiento de la cabeza, como hace el cuerpo humano.

Por otra parte, la integración de estos dispositivos con mecanismos de audio habilitados para filtrar o ecualizar sonidos e incluso posicionar la fuente del mismo en un espacio 3D de tal forma que nuestro cerebro sea capaz de identificar esa localización, nos aporta una mayor sensación de inmersión.

Además, ya existen dispositivos capaces de simular olores por medio de gases artificiales que nos emulan diferentes escenarios, consiguiendo, de nuevo, variar más nuestra percepción de la realidad a favor del escenario artificial.

Sin embargo, aunque cualitativamente la vista es más relevante para nuestro cerebro y más aún potenciada por la integración del sentido del oído y del olfato, es el tacto el que aporta mayor cantidad de información del entorno que nos rodea al utilizar la piel, el órgano más grande de nuestro cuerpo, como medio para captar este sentido.

Es por ello que en los últimos años se ha desarrollado la tecnología háptica, capaz de hacernos sentir estímulos en nuestra piel, como puede ser el contacto con un objeto, su textura o su densidad. Mediante esta tecnología buscamos ser capaces de simular en el cuerpo sensaciones artificiales generadas a partir de la interacción del usuario con un entorno virtual, logrando así una mayor inmersión al usuario y, por tanto, una mejora en la respuesta obtenida por el mismo.

Sin embargo, la tecnología háptica que encontramos actualmente en el mercado se encuentra poco optimizada, debido a los dispositivos de captura de movimiento con los cuales se consigue replicar a tiempo real las distintas partes del cuerpo en los entornos virtuales. Asimismo, el coste de la tecnología háptica es bastante elevado, debido a que los mecanismos utilizados para su implementación requieren de componentes muy precisos y de alto coste.

A pesar de ello, son muchas las empresas, públicas y privadas, las que invierten en la actualidad en el desarrollo de esta tecnología, que aportará un alto valor y fiabilidad a la utilización de los entornos virtuales en diversos campos, como la medicina o la educación, ya que, como hemos comentado, por medio de la integración de todos los sentidos que miden nuestro entorno, seremos capaces de obtener una mayor inmersión en escenarios virtuales.

El objetivo de este proyecto es desarrollar un guante háptico de bajo coste con el que seremos capaces de sentir, a través del tacto de nuestras manos, objetos que se encuentran en un entorno virtual desarrollado con el motor de videojuegos multiplataforma Unity, logrando así dar un paso más en la inmersión del usuario por medio de la integración de los diferentes sentidos en la utilización de entornos virtuales.

Este proyecto conlleva un estudio de la tecnología háptica, de las posibles tecnologías a utilizar para su desarrollo y de los diferentes dispositivos que existen actualmente en el mercado, en los cuales nos basaremos para el diseño y desarrollo de nuestro prototipo.

Por otra parte, como hemos comentado, uno de los mayores problemas a los que se enfrenta la tecnología háptica es el posicionamiento de los elementos del cuerpo

humano en un entorno virtual en tiempo real. En el proyecto también estudiaremos las diferentes tecnologías de captura de movimiento, centrándonos en este caso en el dispositivo Leap Motion y en la captura de movimiento por medio del procesamiento de imagen. Además, plantearemos una solución a los posibles errores cometidos por esta tecnología mediante el uso de unos sensores Flex integrados en el propio guante.

A su vez, el proyecto también incluye el diseño de un entorno virtual desarrollado en el motor de videojuegos multiplataforma Unity, para la simulación y la puesta en práctica del prototipo para mostrar su funcionalidad, consiguiendo así demostrar la eficiencia en la integración de los diferentes sentidos, con el fin de lograr una mayor inmersión por parte del usuario en la escena.

Por otra parte, el proyecto incluye el desarrollo de un prototipo de guante háptico, con el que emularemos el sentido del tacto por medio de motores vibradores, que nos permiten programar patrones de estímulos vibrotáctiles con el control de la frecuencia de vibración de los zumbadores, haciendo uso de señales PWM.

Además, para asegurar que la respuesta es proporcional a la flexión de los dedos, hemos integrado unos sensores Flex al diseño del guante, como hemos comentado anteriormente. Estos sensores consisten en una tira de resistencia variable con la flexión, por lo que nos proporcionarán una respuesta más realista del estado de flexión de los dedos, atajando así posibles problemas que pudieran surgir con la utilización de la tecnología de captura de movimiento basada en el procesamiento de imágenes.

2.3 Metodología seguida

Para el desarrollo del proyecto se han diferenciado distintas partes en cuanto a la metodología seguida.

Primeramente, debido a que uno de los grandes problemas a los que se enfrenta esta tecnología es el coste, se realizó un estudio de la viabilidad del proyecto para el desarrollo de un prototipo de bajo coste por medio de la utilización de dispositivos de captura de movimiento que.

A continuación, se realizó una búsqueda y recopilación de la información existente en el campo de la tecnología háptica y, en particular, de los dispositivos de guantes hápticos.

Una vez establecido el estado del arte en materia de tecnología háptica, se realizó un estudio y aprendizaje de las tecnologías que se iban a utilizar. Por una parte, los lenguajes de programación C++ y C# así como el desarrollo de escenas de realidad virtual en el motor de videojuegos multiplataforma Unity, así como la familiarización con las tecnologías de captura de movimiento por medio de procesamiento de imagen.

Una vez sentadas las bases sobre las que se trabajaría, se diseñó un prototipo de guante háptico utilizando tecnología basada en motores de vibración e integrando los sensores Flex como solución a los problemas de eficiencia de los dispositivos de captura de movimiento por medio de procesamiento de imagen. Para este paso, se han diseñado ambos circuitos, el de vibración y el de flexión, haciendo uso del programa PSpice para

la elección de los componentes más adecuados para nuestros objetivos, así como su caracterización.

Por último, en cuanto al diseño del prototipo, se ensambló el guante háptico para hacer uso de él en las pruebas con entornos virtuales.

Por otra parte, para el estudio del funcionamiento del guante háptico se diseñó un escenario en realidad virtual de un posible caso de uso de esta tecnología, pudiéndose integrar con dispositivos de visualización de entornos virtuales y dispositivos de audio.

Como última fase del proyecto, se realizó la integración de las distintas partes que desarrolladas que componen el proyecto, así como las pruebas de integración, obtención de resultados y resolución de los problemas aparecidos en la fase de integración, logrando así el objetivo marcado de diseñar un guante háptico de bajo coste para su uso en entornos virtuales.

2.4 Contenido y estructura de la memoria

2.4.1 Introducción

En este apartado, en el cual nos encontramos, pondremos en contexto el proyecto, así como la motivación que nos ha llevado a realizarlo y los objetivos que se pretenden cumplir. Además, también estableceremos la metodología seguida en el periodo de duración del proyecto.

2.4.2 Estado del arte

En este apartado trataremos el Estado del Arte referente a los entornos virtuales, la tecnología háptica, haciendo inciso en los guantes hápticos y los dispositivos de captura de movimiento.

2.4.3 Desarrollo del prototipo

Este apartado engloba el desarrollo completo del proyecto: expondremos un análisis del problema y justificaremos la utilización de los componentes elegidos para este proyecto, trataremos el diseño de los circuitos utilizados, así como el desarrollo software realizado para el proyecto.

2.4.4 Pruebas y resultados

En este apartado revisaremos las pruebas realizadas y los resultados obtenidos, dividiendo las pruebas entre las realizadas con el prototipo, las realizadas con el entorno y las pruebas de integración de ambas partes.

2.4.5 Conclusiones

En este apartado trataremos las conclusiones extraídas de la realización del proyecto, la viabilidad del proyecto y los problemas surgidos en el desarrollo del mismo, así como las soluciones aportadas.

2.4.6 Futuras líneas

En este apartado final trataremos las posibilidades del prototipo de guante háptico desarrollado, así como las posibles mejoras que se pueden realizar sobre el mismo.

2.4.7 Bibliografía

En este apartado se realiza una enumeración de todas las páginas webs, libros, publicaciones, escritos y demás que se han utilizado como apoyo a la hora de realizar este proyecto y tener una base sólida para realizar un buen trabajo de investigación.

2.4.8 Anexo

En este apartado se listan las hojas de características de los componentes utilizados, el funcionamiento del dispositivo Leap Motion y las especificaciones técnicas de la placa de Arduino utilizada para el proyecto.

Además, también se encuentra el código completo del desarrollo software realizado para el prototipo de guante háptico.

3 Estado del arte

En este apartado haremos una revisión de la tecnología que existe actualmente en cuanto a la recepción de estímulos y sensaciones ante objetos existentes en un entorno virtual.

Primeramente, definiremos qué es la tecnología háptica y, en particular, qué es un guante háptico, así como los tipos que existe en la actualidad.

Además, trataremos los diferentes dispositivos existentes en el mercado en materia de guantes háptico, así como los sensores que permiten la correcta medida y recepción de dichos estímulos.

Por otra parte, hablaremos del estado del arte de la tecnología de detección y posicionamiento en un espacio virtual por medio de dispositivos de captura de movimiento de elementos del cuerpo humano, pudiendo obtener datos referentes a posición, movimiento, velocidad y gestos efectuados por las mismas.

3.1 Realidad virtual

En este apartado trataremos los conceptos clave de los sistemas de realidad virtual, su funcionamiento y clasificación, así como algunas de las aplicaciones de la realidad virtual en la actualidad.

3.1.1 Definición de realidad virtual

Según Adrew Rowell, “la Realidad Virtual es una simulación interactiva por computador desde el punto de vista del participante, en la cual se sustituye o se aumenta la información sensorial que recibe.”

Un sistema de realidad virtual pretende la estimulación y manipulación de los sentidos del usuario, induciéndole en un engaño que le hace creer que está en otro ambiente. A medida que se manipulan más sentidos, se garantiza una mayor inmersión sensorial para conseguir así una experiencia virtual mayor.

Se puede considerar como una extensión de los sentidos, la cual puede ser utilizada para aprender e interactuar con elementos de una realidad artificial e irreal, siendo así capaces de percibir ideas o vivir representaciones de la realidad. Por ejemplo, aunque entendemos el concepto de “desaparecer”, para nuestro cerebro es imposible visualizarlo en la vida real, sin embargo, esto puede ser conseguido por medio de la realidad virtual, afianzando así la concepción que tenemos de hacer “desaparecer” un objeto.

En un sistema de realidad virtual se pueden distinguir elementos hardware y elementos software, los cuales serán indispensables para el correcto funcionamiento de cualquier sistema. Los elementos más importantes sin los cuales no puede utilizarse los entornos virtuales son los siguientes [1]:

- Hardware: computador, periféricos de entrada y periféricos de salida.

- Software: modelo geométrico 3D y programas de simulación sensorial, simulación física y recogida de datos.

3.1.2 Inmersión sensorial

Como la propia definición de realidad virtual indica, uno de los aspectos claves en un sistema de realidad virtual es la inmersión sensorial.

Podemos definir la inmersión sensorial como la desconexión de los sentidos del mundo real y la conexión al mundo virtual. Como consecuencia, el usuario deja de percibir el entorno que le rodea y pasa a estar inmerso dentro del mundo virtual que recrea el computador en tiempo real, por este motivo, todo sistema de realidad virtual debe proporcionar estímulos adecuados a nuestros sentidos, aumentando la inmersión al integrar y manipular un mayor número de nuestros sentidos [1].

El grado de inmersión sensorial depende, en primera instancia, de cuáles son los órganos de los sentidos para los cuales el sistema proporciona estímulos adecuados y, en segunda instancia, del alcance, calidad, velocidad y coherencia de estos estímulos.

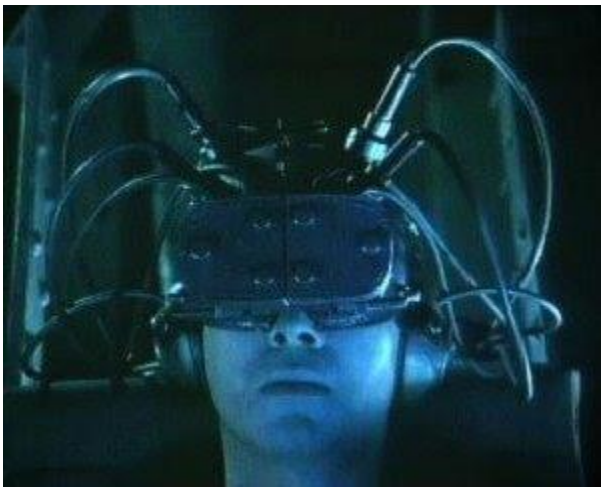

Figura 1 - Inmersión en entorno virtual

3.1.3 Funcionamiento de un sistema de realidad virtual

Las acciones del usuario son registradas en tiempo real por los periféricos de entrada o sensores y se envía al computador, donde es procesada por los módulos de recogida y tratamiento de datos de entrada.

Una vez las acciones del usuario están en un formato adecuado, se utilizarán en la simulación, tanto para introducir posibles cambios en el mundo virtual como para generar imágenes, sonidos y otros datos para completar la simulación sensorial. En nuestro caso, por ejemplo, el dispositivo Leap Motion será el encargado de capturar las

acciones de las manos del usuario, modificando el entorno al replicar la mano en la simulación de realidad virtual.

Este proceso se repite de manera continuada y con el menor retardo posible, pretendiendo incluso que sea en tiempo real, con el fin de que el usuario no perciba dicho retardo introducido por el procesamiento de los datos.

3.1.4 Tipos de entornos virtuales

La realidad aumentada depende del entorno virtual en el que se aplique y el tipo de entorno virtual depende de la interacción que tiene usuario con el entorno. Se pueden diferenciar tres tipos:

1. Entornos pasivos: Se caracterizan porque no son interactivos y el usuario no es capaz de moverse por la escena para observar los elementos que la componen.

2. Entornos exploratorios: Este tipo de entorno sí que permite que el usuario se mueva a su antojo por el entorno y el mismo entorno puede ser también dinámico, dando así al usuario la sensación de libertad que no permitían los entornos pasivos.

3. Entornos interactivos: Este tipo de entorno mejoran la funcionalidad con respecto a los entornos exploratorios y, por tanto, a los entornos pasivos ya que permiten la interacción del usuario con el entorno virtual y permiten modificarlo, logrando así una mayor sensación de inmersión.

3.1.5 Aplicaciones de la realidad virtual

Una vez establecidas las características de los entornos virtuales, su funcionamiento y los tipos, en este apartado trataremos las diferentes aplicaciones de esta tecnología en la actualidad.

3.1.5.1 Medicina y psicología

La industria de la medicina y más en particular, de la cirugía basa su modelo de aprendizaje en “aprender haciendo”. Por tanto, los entornos virtuales se utilizan para ensayar operaciones o intervenciones de manera segura para ahorrar vidas, consiguiendo así una simulación muy fiel a la realidad.

Por ejemplo, mediante la utilización de la realidad virtual, se pueden estudiar casos de manera práctica e incluso crear escenarios adversos para aprender a gestionarlos de manera más eficiente.

Además, la representación 3D de un órgano, dada el gran avance de los escáneres actualmente, es un método eficiente para estudiarlo e interactuar con él.

Por otra parte, en ámbito de la psicología la realidad virtual se utiliza para tratar fobias, por ejemplo, a los insectos o a las alturas, donde el paciente tiene el control y aprende

las herramientas necesarias para exponerse a su fobia de manera controlada y de forma gradual para superarla.

Figura 2 - Realidad virtual en la medicina

3.1.5.2 Industria y diseño

El diseño de una pieza antes de fabricarla es un paso necesario en cualquier proyecto industrial, el uso de la realidad virtual permite ser capaces de visualizar la pieza diseñada, ver cómo encaja con el reto de piezas e incluso modificarla en tiempo real, siendo así capaces de ver la interacción de las diferentes piezas entre ellas.

Además, la realidad virtual nos permite ver una simulación del proceso de montaje y ensamblaje de cualquier máquina o entorno artificial, siendo capaces de seguir el proceso sin requerir de una fábrica y maquinaria real.

Otra aplicación en este campo es en los procesos de corte, que son procesos difíciles, peligrosos y que requieren de un aprendizaje práctico. Por medio de la realidad virtual se evitan riesgos y se reducen costes.

Figura 3 - Realidad virtual en la industria

3.1.5.3 Docencia y pedagogía

Es el campo más beneficiado del uso de esta tecnología. Por medio de la realidad virtual somos capaces de viajar a otras partes del mundo, visitar acuarios o estudiar materias densas de forma visual y práctica

Por ejemplo, la realidad virtual puede ser utilizada en asignaturas de diseño de antenas ya que, por medio de la realidad aumentada, podemos visualizar diagramas de radiación o ver la atenuación de una señal Wifi en un entorno con paredes y obstáculos.

Figura 4 - Realidad virtual en la docencia

3.2 Tecnología háptica

Se denomina “háptica” como el estudio del tacto y la interacción con este sentido, siendo estas interacciones naturales o artificiales, con el fin de obtener información del entorno por medio de estímulos en la piel.

Por tanto, la tecnología háptica es aquella que es capaz de transmitir sensaciones y estímulos artificiales al sujeto, el cual los recibirá a través del tacto.

Para el desarrollo de la tecnología háptica en la actualidad se han implementado diferentes tipos de tecnología capaces de transmitir dichos estímulos, pero el rendimiento de esta tecnología depende de la zona del cuerpo que se estimule por el número de receptores que tenemos en función de la zona de actuación de la tecnología háptica sobre nuestro cuerpo.

La principal aplicación de la tecnología háptica se basa en potenciar el uso de entornos virtuales. Esta práctica, tal y como comentamos en el apartado de Realidad virtual, nos lleva aplicarla a diversos campos, aunque, debido a la poca seguridad que ofrecen actualmente los dispositivos hápticos, se centra en la docencia en diversas áreas. Asimismo, la industria de los videojuegos ha sido la más beneficiada de la aplicación de esta tecnología, que aporta un valor añadido a sus entornos virtuales para una mejora en la experiencia de usuario.

El uso de los dispositivos hápticos en las simulaciones de Realidad Virtual presenta ventajas y desventajas:

Las ventajas incluyen que la comunicación está centralizada a través del tacto y que el mundo digital puede comportarse como el real. Cuando los objetos puedan ser capturados, manipulados, modificados y reescalados digitalmente, el tiempo de trabajo será reducido.

Las desventajas incluyen el problema de la depuración de los programas. Esto es muy complicado puesto que implica un análisis de datos en tiempo real. Además, la precisión del tacto requiere de muchísimos avances en este campo.

Por ello, a pesar de que las ventajas podrían proporcionar grandes avances tecnológicos son principalmente sus desventajas, en el tema de la seguridad, las que han impedido actualmente su implementación en diversas áreas de trabajo que requieran de este requisito [2].

3.2.1 La piel como interfaz

Gracias a los estudios realizados sobre el sentido del tacto[3][2], se concluye la capacidad del cerebro de reconocer objetos basándose solo en la información táctil que tenemos de él, siendo capaces de reconocer objetos en 2-3 segundos con solo tocarlos [3].

Esta capacidad se debe al sentido del tacto que captamos a través de la piel, el órgano más grande de nuestro cuerpo, y su estructura, que cuenta con órganos receptores por toda su superficie. Estos receptores son los que nos permiten recibir estímulos tanto internos como externos.

Estos receptores o también conocidos como “mecanoreceptores” están diseñados para actuar antes estímulos mecánicos sobre nuestra piel, siendo capaces de identificar bordes, texturas y presiones ejercidas sobre ella. Además, encontramos diferentes tipos de receptores en función de la profundidad de nuestra piel.

En el caso de la piel de la palma de las manos, no son tan sensibles a cambios de presión como pudiera ser, por ejemplo, la piel de la cara, sin embargo, tienen un grosor mucho menor que, por ejemplo, la piel del torso.

Esto nos lleva a pensar que la zona más adecuada, por cantidad de receptores, para la estimulación háptica es el torso, sin embargo, la capacidad de flexión de los dedos nos permite identificar los objetos y sus formas de manera más natura, por tanto, es la zona más adecuada para el reconocimiento de objetos virtuales.

Cabe destacar, además, que la velocidad de reconocimiento de los objetos se verá ampliamente incrementada si se combinan la tecnología háptica con dispositivos de realidad virtual.

3.2.2 Realimentación háptica

La mayor parte de los servicios que requieren de entornos virtuales cuentan con dispositivos como mandos o palancas que no son del todo intuitivos en algunas ocasiones y, por supuesto, no aportan una experiencia realista de lo que se está consumiendo.

Es por ello que para conseguir servicios lo más reales posibles es necesario integrar los diferentes sentidos en el entorno en el que se trabaja, además de conseguir sencillez y fiabilidad en el servicio que se consuma, desde videojuegos hasta operaciones médicas.

En 2012 se hicieron estudios experimentales para el estudio de la modificación de la orientación y la precisión por medio de la manipulación de los sentidos, del cual se concluyó que los sujetos que contaban con retroalimentación háptica obtenían resultados mucho más precisos que aquellos que no lo hacían [4].

Asimismo, se hicieron pruebas incorporando la retroalimentación háptica a entornos virtuales de videojuegos, obteniendo resultados que mejoraban la experiencia de usuario, así como la precisión.

Por otra parte, la empresa Disney comenzó el proyecto “Touche” en 2012 mediante el cual se pretende que cualquier objeto, electrónico o no, tenga una respuesta táctil programable [5]. Más tarde las cosas fueron todavía un poco más lejos gracias a una tecnología de "electrovibración reversa" llamada REVEL, que por medio de un generador de señales electrostáticas podría crear un campo que daría una impresión de fricción distinta a la real. Este curioso sistema permitiría dar una textura táctil a objetos en pantallas o superficies planas, e incluso cambiar nuestra percepción táctil de objetos reales [6].

Los investigadores fueron más allá aplicando la tecnología háptica en labores de diseño. Nació así ImmersiveTouch [7], que pretende dar un salto en material de realidad aumentada proporcionando una herramienta de manipulación de objetos 3D a tiempo real por medio del seguimiento de las manos y la cabeza. El usuario observa en una pantalla el objeto y actúa sobre él por medio de la tecnología háptica.

Con estos, además de otros muchos estudios de campo, queda patente la importancia la integración de la tecnología háptica como realimentación para conseguir una mayor inmersión del usuario en el entorno virtual, así como las mejoras y las posibilidades que aporta esta tecnología.

3.3 Guantes hápticos

Como hemos comentado, el tacto es el sentido que nos permite sentir la forma, la textura y la presión de los objetos que tocamos con la piel. Es precisamente por ser el más extenso que podemos recibir los estímulos en cualquier parte del cuerpo. En este proyecto nos centraremos en la transmisión de sensaciones por las manos, haciendo uso de un guante háptico ya que las manos nos aportan una mayor fidelidad a la hora de identificar un objeto por poder conocer su forma con la flexión de nuestros dedos.

El guante háptico será el dispositivo que nos permitirá recibir sensaciones y estímulos a través del tacto de nuestras manos haciendo uso de los distintos tipos de tecnología háptica.

Actualmente, para el desarrollo de guantes hápticos se utilizan mayoritariamente tres tecnologías:

1. Electromagnetismo: Basan su funcionamiento en la limitación de la flexión de los dedos por medio del control de los electroimanes ensamblados en el guante para hacernos sentir que estamos tocando o sujetando un objeto. Es una tecnología disruptiva pero aún se encuentra en desarrollo. Este tipo de guantes cuentan con una estructura fija y otra adaptable ya que, aunque los electroimanes tienen una posición específica, para garantizar el correcto funcionamiento de este tipo de guantes hápticos es necesario adaptar la longitud de los limitadores de movimiento.

2. Exoesqueletos: Tienen un funcionamiento similar al de los guantes hápticos basados en electroimanes, pero estos limitan el movimiento por medio de una serie de motores ensamblados en un exoesqueleto que impiden la flexión de los dedos. Este tipo de guantes hápticos cuentan con una estructura fija en la que se insertan las manos, aunque luego se aseguren y fijen a la estructura para aumentar la efectividad del guante háptico.

3. Vibración: Este tipo de tecnología se basa en la vibración y la presión ejercida en diversos puntos de la mano para sentir el objeto virtual. Si bien permite una experiencia más completa, este tipo de tecnología suele combinarse con una de las anteriores para conseguir de esta manera la limitación del movimiento. Este tipo de guantes hápticos requieren una adaptación previa a las manos del usuario ya que las posiciones de los motores vibradores están estudiadas para que se localicen en una zona exacta para aumentar la efectividad del guante.

Por otra parte, en el mercado actual vemos diversos tipos de guantes hápticos en función de la tecnología de la que hacen uso. Lo más relevantes y que mejores resultados aporta son los siguientes:

1. DextrES: Este guante permite simular el agarre de objetos en realidad virtual mediante el bloqueo del movimiento de los dedos, así como también de ofrecer respuesta háptica en la punta de los mismos. El diseño apuesta por un factor de forma ligero que logra actualmente un peso de tan solo 8 gramos, aunque de momento no incluye batería, dado que funciona con un cable de alimentación. Utilizando impulsos eléctricos a través de un freno electrostático que dificulta la fricción entre filamentos de la lámina, DextrES es capaz de generar el equivalente a 2 kg de resistencia por dedo [8].

Figura 5 - Guante háptico DextrES

2. Plexus: La gran diferencia de los Plexus es que no cuentan con un diseño basado en los típicos guantes que cubren toda la mano, ya que sólo tapan las puntas y la parte exterior de cada dedo. Frente a otros periféricos similares, los guantes de Plexus prometen una experiencia cómoda gracias a la silicona flexible de alta calidad con la que están fabricados y al sensor que tiene patentado la compañía, siendo capaz de ofrecer un seguimiento con una precisión de 0.01 grados. Si bien aún no son capaces de ofrecer resistencia física. Los resistentes actuadores lineales, que están colocados en cada punta de los dedos, proporcionan al usuario una realista retroalimentación háptica. Mientras que los guantes en sí se encargan del seguimiento de los dedos, su elegante tamaño implica, desafortunadamente, el tener que dejar las funciones del seguimiento posicional a otras soluciones ya existentes. En cuanto a la conectividad, los guantes son inalámbricos y funcionan a través de conexión WiFi, en la banda de 2.4 GHz con un protocolo personalizado de baja latencia. Además, disponen de una autonomía de 2 a 3 horas [9].

Figura 6 - Guante háptico Plexus

3. Gloveone: Este dispositivo funciona mediante un complejo sistema de terminales repartido por el interior del guante, cuya función es captar los movimientos de la mano para trasladarlos al universo digital y, al mismo tiempo, reproducir mediante vibraciones de diversa potencia la sensación de tacto de los elementos que se estén

modificando en el mundo virtual. Pese a que el guante es la gran innovación, para probar todo su potencial hace falta un software adecuado y una programación acorde [10].

Figura 7 - Guante háptico Gloveone

3.4 Detección de movimiento

La detección de movimientos y gestos producidos por el cuerpo humano es un método utilizado como interfaz entre el humano y el ordenador que provee de mecanismos más sencillos e intuitivos para el control de dispositivos. Esta tecnología utiliza cámaras y modelos matemáticos para la identificación y posicionamiento de los elementos del cuerpo humano en un entorno virtual, aportando datos en tiempo real que nos permiten, por ejemplo, establecer actuaciones de dispositivos en función del movimiento ejecutado. En este apartado nos centraremos en la tecnología referente a la detección del movimiento de las manos.

Actualmente, en el mercado existen diferentes tecnologías para medir el movimiento.

3.4.1 MYO

Es un producto que permite la medida del movimiento de la mano mediante una serie de electrodos de tipo “electromiográficos”. El dispositivo es un brazalete, como puede observarse en la Figura 8, en donde se alojan los sensores [11].

La producción de MYO fue cancelada el pasado año, por lo que la información referente a este dispositivo está limitada a las páginas externas a la oficial que mantienen la información de dicha tecnología. Aunque de primeras no sería un posible dispositivo que pudiéramos utilizar para el proyecto debido a este último motivo, la tecnología podría llegar a ser válida.

Figura 8 - Dispositivo MYO

3.4.2 Nod

Este dispositivo utiliza acelerómetros y giroscopios para medir el movimiento realizado por las manos. Es un dispositivo en forma de anillo, como puede observarse en la Figura 9, en el que se alojan los sensores y que se conecta vía Bluetooth para trasmitir los datos medidos [12].

Se ajusta a las necesidades de portabilidad y sencillez, sin embargo, no se ajusta a las necesidades de monitorización de datos referentes a la mano ya que se limita al control por gestos y su uso es más específico para control de IoT.

Figura 9 - Dispositivo Nod

3.4.3 Leap Motion

Este dispositivo utiliza las imágenes captadas por dos cámaras infrarrojas con iluminación LED que, previo procesamiento matemático de las imágenes captadas, permite posicionar la mano en un espacio 3D y aportar datos en tiempo real del movimiento que deseamos medir. Es un dispositivo de dimensiones pequeñas, forma rectangular y de poco peso, lo que permite una cómoda portabilidad [13].

Figura 10 - Dispositivo Leap Motion

Es un dispositivo más preciso y de un coste más reducido que el resto de dispositivos anteriormente mencionados.

Leap Motion será el dispositivo elegido para desarrollar este proyecto teniendo en cuenta su versatilidad, bajo coste y fiabilidad en materia de detección de movimiento. Además, cuenta con un SDK que aporta un amplio rango de posibilidades en función de la aplicación que deseamos realizar

4 Desarrollo del proyecto

En este apartado abordaremos todo el desarrollo del proyecto teniendo en cuenta los objetivos presentados en el apartado de Introducción.

Trataremos la finalidad del proyecto, el material utilizado, el diseño de los diferentes circuitos necesarios para la implementación del prototipo y el desarrollo del prototipo final.

4.1 Análisis del problema

El proyecto pretende que seamos capaces de sentir los objetos virtuales que se encuentran en un escenario desarrollado en Unity a través de un guante háptico haciendo uso de Leap Motion.

La tecnología elegida para el desarrollo de este proyecto de las listadas en el apartado de Guantes hápticos ha sido la de vibración ya que nos permite no solo la identificación de la forma de los objetos virtuales sino también su presión al poder variar su actuación por medio de la utilización de señales PWM. Para el desarrollo de la tecnología háptica en este proyecto, hemos diseñado el circuito que realizará la realimentación por medio del Zumbador Pinzhi bi00149-es, cuyas características más relevantes podremos encontrar en el Anexo. Estos zumbadores serán controlados con el microcontrolador con el fin de actuar de forma adecuada a la flexión y la posición de la mano.

Para el desarrollo del guante háptico es necesario utilizar un sistema digital basado en un microcontrolador para recibir la información relativa al entorno virtual alojado en un ordenador, procesarla y reproducir la consiguiente respuesta en el guante.

Un microcontrolador, es un circuito integrado programable capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de numerosos bloques funcionales, los cuales cumplen una tarea específica. Incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida. Entre los microcontroladores más utilizados en el mercado encontramos Raspberry [14] y Arduino [15].

Debido a su versatilidad, fácil programación y funcionamiento, para esta tarea hemos hecho uso del microcontrolador Arduino y, en particular, del modelo Arduino UNO. Hemos hecho uso de la placa de Arduino frente a otros modelos de microcontroladores como Raspberry ya que el procesado de los datos no requiere demasiada potencia de procesamiento, que es la principal ventaja aplicable que aporta Raspberry, y Arduino nos ofrecía una mayor cantidad de entradas y salidas analógicas y digitales aprovechables, además de su fácil integración.

Por otra parte, es cierto que la conectividad de las placas de Raspberry ya sea por Bluetooth o por WiFi, nos aportarían una mayor movilidad al prototipo, lo que podría ser una potencial futura línea.

Para ser capaces de controlar un proceso con el microcontrolador, será necesaria su programación, la cual realizaremos en el lenguaje de programación C++ y haciendo uso de la API de Arduino. En el Anexo podremos encontrar las principales características de la placa de Arduino UNO con la que trabajaremos.

En lo referente a la captura de movimiento, hemos hecho uso del dispositivo Leap Motion para desarrollar este proyecto teniendo en cuenta su versatilidad, bajo coste y fiabilidad en materia de detección de movimiento frente a los otros dispositivos comentados en el apartado del Estado del Arte. Leap Motion nos dará la posición de la mano y nos permitirá verla en el entorno virtual, siendo capaces de actuar sobre el entorno digital de una forma sencilla e intuitiva.

Por otra parte, Leap Motion cuenta con un SDK que aporta un amplio rango de posibilidades en función de la aplicación que deseamos realizar gracias a toda la información que extrae del procesamiento de imagen. En el Anexo podremos encontrar la información relativa a su funcionamiento, software y hardware.

Sin embargo, aunque el dispositivo Leap Motion nos aporta una mayor seguridad y cantidad de datos que el resto de dispositivos de captura de movimiento, el dispositivo tiene una alta probabilidad de error en la medida de datos cuando los dedos se encuentran flexionados. Además, aunque el dispositivo nos permite saber si cada dedo se encuentra o no en flexión, no nos permite conocer el grado en que está flexionado.

Por estos dos motivos hemos hecho uso de los sensores Flex. Estos sensores se tratan de una tira de resistencia variable: al estirar el sensor Flex o, lo que es lo mismo, flexionarlo, la resistencia aumentará.

Al igual que el caso de los vibradores, hemos diseñado el circuito que nos permitirá medir correctamente la flexión utilizando para ello el modelo Sensor Flex BricoGeek 7 cm por su bajo coste, fácil manejo y, lo más importante, su fácil adaptación a la forma de los dedos. En el Anexo podremos encontrar las características más relevantes del modelo de sensor Flex elegido.

Además, para el desarrollo completo del proyecto será necesaria la utilización de un ordenador para alojar el entorno virtual sobre el que actuaremos. Para el diseño de la escena será necesario usar una plataforma de desarrollo de aplicaciones de realidad virtual. Las que mejores prestaciones nos aportan son las siguientes:

- WebVR [16]: es una plataforma que permite disfrutar de contenido web con cualquier dispositivo de realidad virtual y un navegador web compatible. La API es totalmente gratuita y ofrece soporte para los principales controles. Sin embargo, esta plataforma no cuenta con un entorno propio de desarrollo de aplicaciones en 3D, sino que sirve como interfaz entre el contenido web y el hardware de realidad virtual.

- Unreal [17]: Unreal Engine 4 dispone de una interfaz gráfica de desarrollo propia que cuenta con todas las herramientas necesarias para exprimir el potencial de este motor y está especializado en el desarrollo de contenido 3D de muy alta calidad. También está destinado al desarrollo de realidad virtual y es totalmente

compatible con todos los dispositivos actuales. Las principales desventajas de Unreal Engine 4[17] no se ciñen al propio motor, sino más bien a la comunidad que hay detrás de este. Pese a ser un motor muy popular entre las grandes compañías de videojuegos, la comunidad es relativamente pequeña si la comparamos con la de otros motores como Unity. Esto, sumado a una pronunciada curva de aprendizaje, podría dificultar el desarrollo en esta plataforma.

- Unity [18]: Unity3D dispone de IDE propio para el desarrollo de aplicaciones 3D y, además, cuenta con un amplio abanico de "plugins", tanto gratuitos como de pago para ampliar en gran medida su funcionalidad. En la actualidad existen varios SDK destinados al desarrollo de aplicaciones para realidad virtual en Unity que facilitan la puesta en marcha de un proyecto de estas características.

Para el desarrollo de la escena hemos hecho uso del motor de videojuegos multiplataforma Unity por la cantidad de información de código libre que existe en internet, su fácil aprendizaje al estar basado en el lenguaje de programación C# y por tener unas dependencias con el hardware del ordenador que aloje la escena menor que las necesarias para utilizar un entorno virtual diseñado en Unreal.

4.2 Diseño de circuitos

Una vez analizado el problema al que nos enfrentamos y listados los componentes que vamos a utilizar para el desarrollo del guante háptico, en este apartado trataré el diseño de los circuitos que componen el hardware del prototipo.

Como hemos comentado, la placa de Arduino será la encargada de la conexión entre el ordenador en el que se alojará la simulación y el guante háptico. Procesará la información de los sensores Flex, la mandará al ordenador y controlará los vibradores.

El guate háptico se puede dividir en dos circuitos: el primero será el encargado de medir la flexión de los dedos gracias a los sensores Flex y el segundo será el encargado de hacernos sentir los objetos por medio de los zumbadores.

Para el diseño del guante háptico hemos hecho uso del software PSpice además de las hojas de características de los distintos componentes, habiendo reproducido en PSpice los modelos de los componentes para los cuales no existía en las librerías, en particular, del circuito integrado AD8656.

En cuanto al montaje, hemos utilizado una protoboard y jumpers además de los componentes particulares de cada circuito listados en el apartado de Análisis del problema. Por otra parte, hemos hecho uso de elementos de soldadura para ensamblar el hardware de forma fija, así como el material de laboratorio necesario para el correcto montaje y diseño del mismo (osciloscopio, generador de funciones, multímetro, etcétera).

4.2.1 Circuito de flexión

Como hemos comentado, gracias a los sensores Flex seremos capaces de medir la flexión de los dedos con mayor precisión que la que ya aporta el Leap Motion, sorteando así el error de precisión que pueda aparecer, a pesar de ser este dispositivo bastante fiable. Para futuras aplicaciones, la precisión puede llegar a ser un componente necesario, asegurando así el completo funcionamiento del prototipo.

El sensor Flex será el encargado de medir la flexión de los dedos para controlar de forma más precisa la actuación de los vibradores.

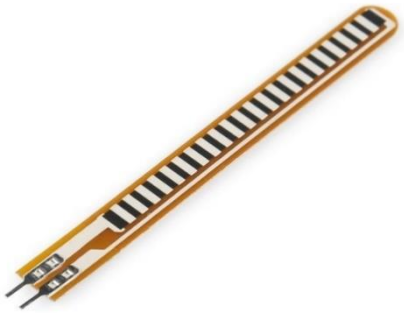

Figura 11 - Sensor Flex

Consiste en una tira flexible que actuará como una resistencia variable. Cuando el sensor Flex se encuentre plano la resistencia tendrá un valor de 25KΩ y al doblarse irá aumentando su resistencia hasta un valor de 125KΩ.

Haciendo uso de la hoja de características del sensor Flex, obtenemos el circuito de la Figura 12.

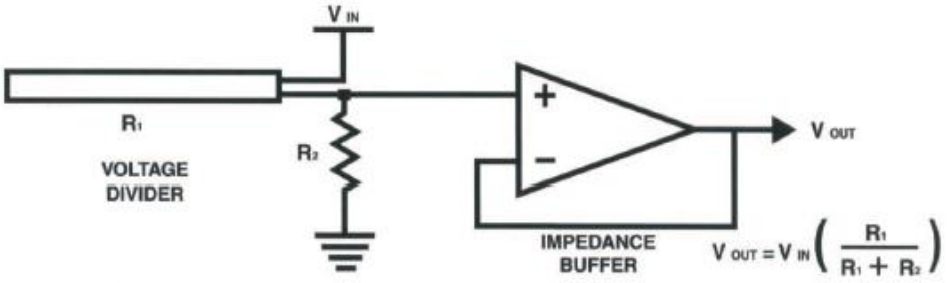

Figura 12 - Circuito de flexión

Como podemos observar, será necesario un amplificador operacional cuya salida dependerá de las resistencias R1, la resistencia equivalente al sensor Flex, y R2, así como de la tensión Vin, la introducida por los pines analógicos de la placa de Arduino.

La función del amplificador operacional es la de aislar el divisor de tensión formado por las resistencia la equivalente al sensor Flex, y la resistencia fija R2 [22].

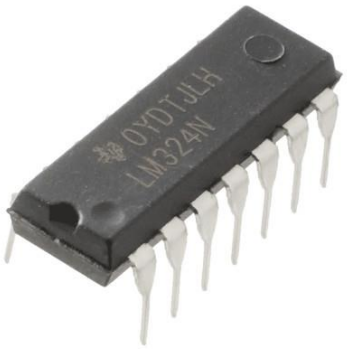

Figura 13 - Circuito integrado LM324

El LM324 consiste en un circuito integrado con 4 Amplificadores Operacionales, un pin de alimentación a 5V y un pin de tierra. En el Anexo podemos encontrar las características más relevantes del LM324.

Como recordatorio, los pines de Arduino introducen una tensión de 5V y, como hemos comentado, el sensor Flex tendrá una mayor resistencia cuanto más doblada esté la resistencia con un rango de entre 25KOhm y 150kOhm. Por lo tanto, la variable que queda por determinar es R2.

Siguiendo la fórmula de cualquier amplificador operacional, tenemos que:

$$Vout = Vin * (\frac{R1}{R1 + R2})$$

Si R1 es mucho mayor que R2, Vin será (aproximadamente) igual a Vout. Si R1 es mucho menor que R2, Vin tenderá a 0. Si R1 es igual a R2, la tensión Vout será la mitad que Vin.

Nuestro cometido mediante este circuito es el de obtener un rango de tensiones Vout lo más amplio posible tal que podamos discretizar lo máximo posible el grado de flexión del sensor Flex. Siguiendo este razonamiento y la fórmula del amplificador operacional, R2 debe ser menor que el valor máximo de R1, pero no más pequeña que el valor mínimo de R1 para asegurar un buen margen dinámico para cualquier grado de flexión.

Para elegir el valor óptimo de la resistencia R2 hemos hecho uso de PSpice. Podemos observar el circuito final en la Figura 14.

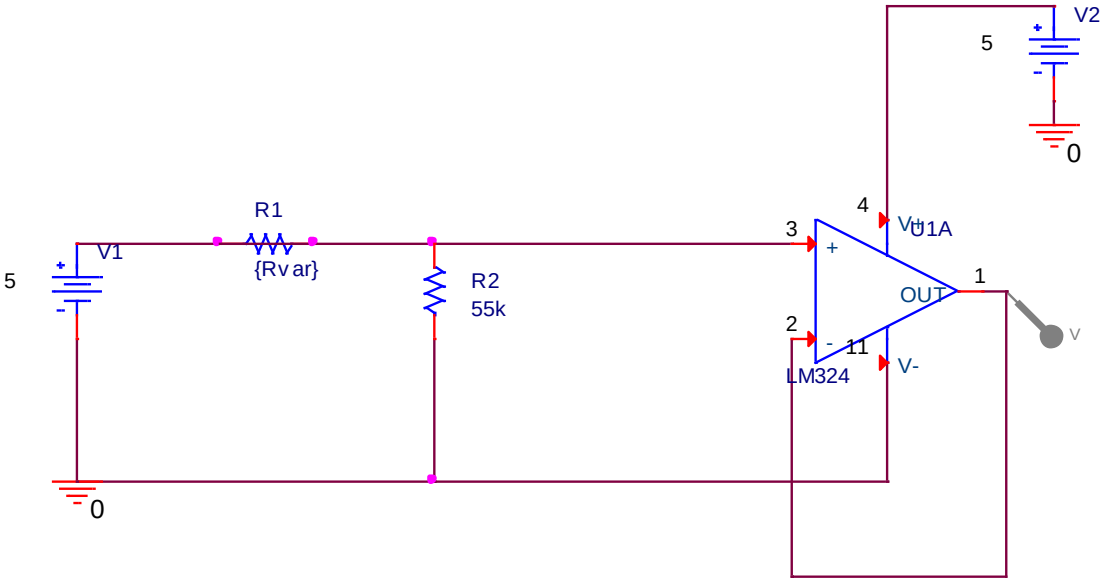

Figura 14 - Circuito de flexión en PSpice

Para elegir el valor más adecuado de la resistencia fija R2 que nos dé un rango de tensiones de salida más amplio haré un análisis paramétrico solapado a un barrido en continua del circuito, obteniendo así las salidas mostradas en la Figura 15 en función de la resistencia fija R2 para saltos de 1kOhm:

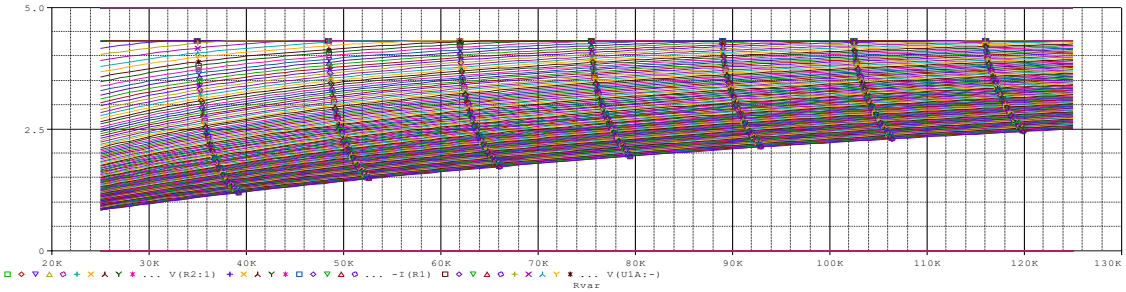

Figura 15 - Análisis paramétrico de la resistencia R2

Por otra parte, en la Figura 16 podemos ver la variación de la salida en función de la resistencia fija R2 para saltos de 5kOhm:

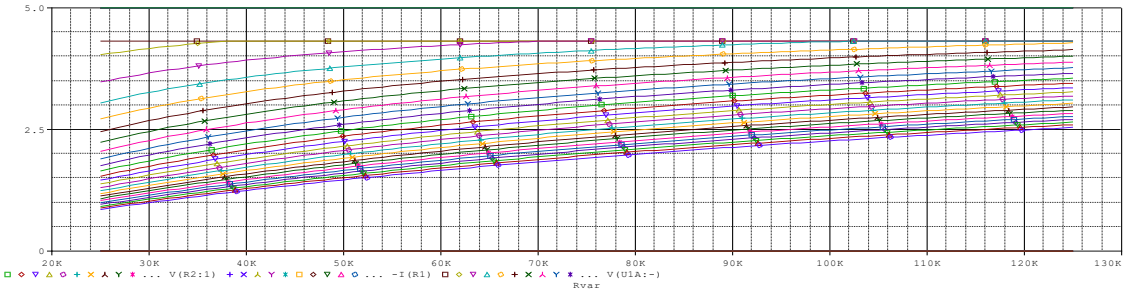

Figura 16 - Variación de la salida en función de la resistencia R2

Donde la curva inferior será la correspondiente a R2=125kOhm y la superior la correspondiente a R2=1kOhm. Como se puede observar, para valores superiores de R2 la curva es más lineal.

Es necesario encontrar el valor de R2 que nos permita un rango dinámico amplio a la salida, pero sin incrementar excesivamente la no linealidad de la misma.

A continuación, se puede observar el margen dinámico en función de la resistencia R2 en la Figura 17:

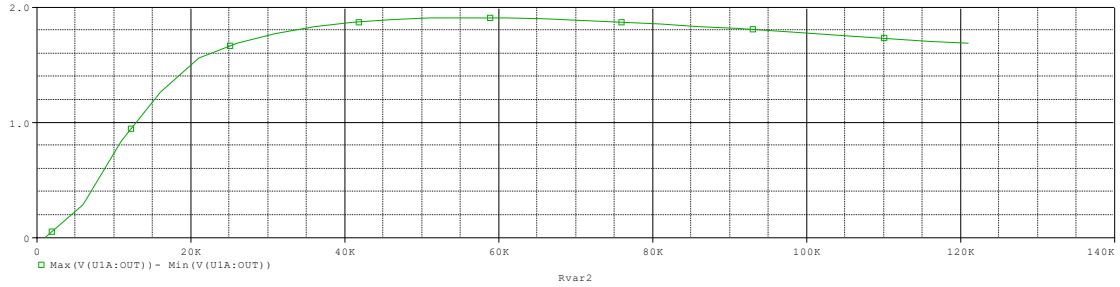

Figura 17 - Margen dinámico en función de la resistencia R2

Podemos observar que para 55kOhm obtenemos el mayor margen dinámico de 1.9101V, a partir de dicho valor, el margen dinámico disminuye hasta los 1.6724V para 120kOhm.

Por otra parte, en la Figura 18 se puede observar la no linealidad de la salida en función de la resistencia R2:

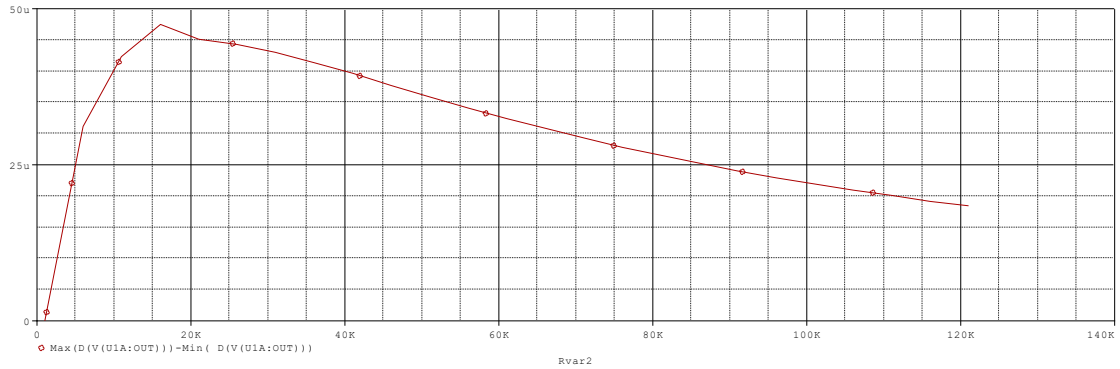

Figura 18 - Linealidad de salida en función de la resistencia R2

A partir de los 18kOhm la no linealidad se ve reducida conforme aumenta R2. Sin embargo, los valores de la pendiente son muy bajos y la variación de la misma al aumentar R2 no son notorios frente a la variación que obtenemos del margen dinámico.

Teniendo en cuenta que la pendiente no varía demasiado al aumentar R2 por encima de 18kOhm, este parámetro de elección de la resistencia fija R2 no es tan relevante como puede ser el margen dinámico. Por tanto, el valor adecuado para R2 es de 55kOhm (56kOhm para un valor comercial).

Específicamente, para un valor de 55kOhm obtendríamos la siguiente salida con un margen dinámico de 1.9101V representada en la Figura 19.

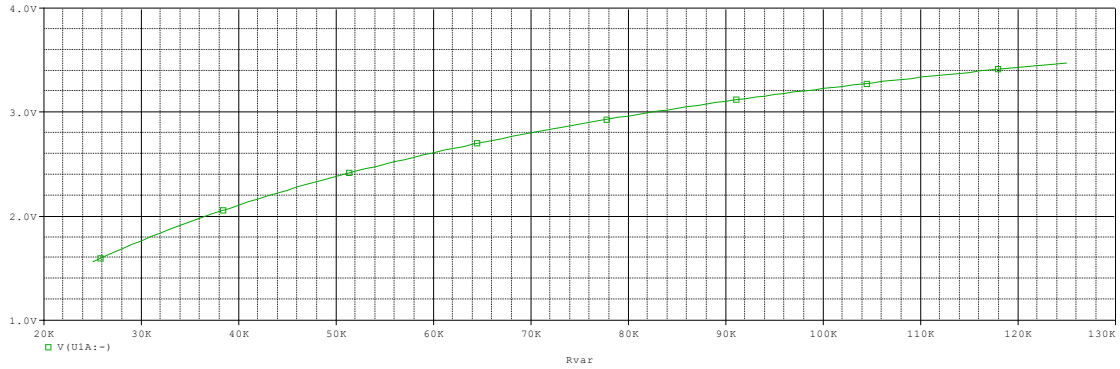

Figura 19 - Salida para un valor de la resistencia R2 de 55kOhm

4.2.2 Circuito de vibración

La tecnología que nos permitirá actuar sobre el tacto de nuestras manos será la de vibración. El zumbador Pinzhi bi00149-es es el encargado de trasmitir la vibración a nuestra palma de la mano para notar el objeto virtual. Será controlado por la placa de Arduino UNO.

Figura 20 - Zumbador Pinzhi bi00149-es

El zumbador consta de un motor que generará la vibración. Es de dimensiones pequeñas para poder adaptarlo bien a las yemas y la palma de la mano.

Hemos caracterizado de dos formas distintas el vibrador: cuantitativamente hemos comprobado la relación V/I y cualitativamente he establecido un umbral de percepción de 6 niveles.

Voltaje (V)	Corriente (mA)
0,5	14,8
0,7	16,7
1	18,4
1,2	23,6
1,4	31,6
1,6	36,8

1,8	40
2	43,6
2,2	46,4
2,4	52
2,6	57,1
2,8	60,4
3	65,6
3,3	70,1
3,5	77,1
3,7	82,6
4	91,4

Tabla 1 - Análisis cuantitativo Voltaje-Intensidad de Zumbador Pinzhi bi00149-es

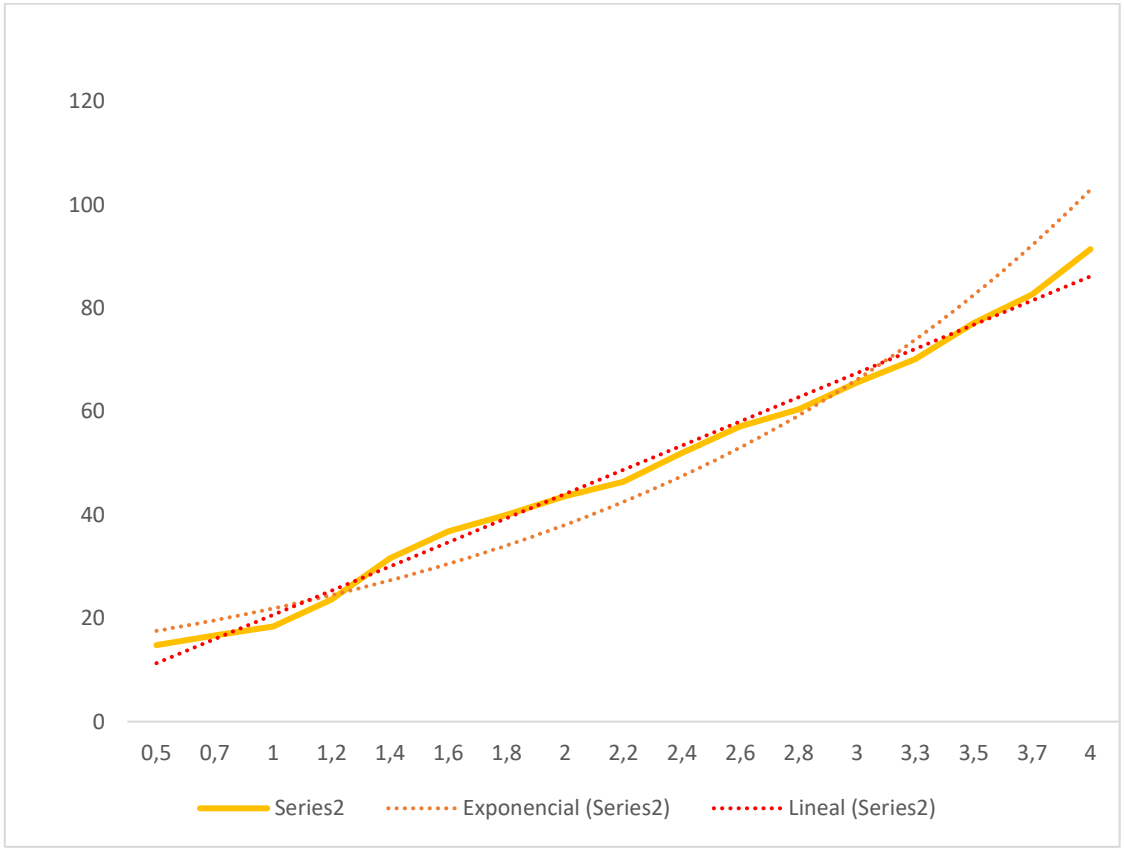

Tabla 2 - Curva de Corriente en función del Voltaje inyectado al Zumbador Pinzhi bi00149-es

Como podemos observar, la curva de la corriente en función de la tensión inyectada al vibrador es bastante lineal. La resistencia calculada teóricamente es de entorno a 45Ω de media. La calculada en la práctica varía entre 37Ω y 47Ω.

Para el análisis cualitativo hemos establecido los niveles de sensación sin el guante, por lo que la percepción es mayor.

El código de colores va del 1 al 6, siendo el 1 agradable y el 6 prácticamente molesto.

Voltaje (V)	Corriente (mA)	Sensación
0,5	14,8	1
0,7	16,7	1
1	18,4	1
1,2	23,6	2
1,4	31,6	2
1,6	36,8	3
1,8	40	3
2	43,6	3
2,2	46,4	4
2,4	52	4
2,6	57,1	4
2,8	60,4	4
3	65,6	5
3,3	70,1	5
3,5	77,1	5
3,7	82,6	5
4	91,4	6

Tabla 3 - Análisis cualitativo Voltaje-Intensidad de Zumbador Pinzhi bi00149-es

Como podemos observar, conforme aportamos una mayor corriente, mayor es la sensación de incomodidad al vibrar más el dispositivo.

Nótese que en las hojas de especificaciones viene establecido que el rango de tensión es 2.5V a 3.8V, sin embargo, a partir de 0.5V ya hay vibración, aunque pudiera ser imperceptible con el guante.

Además, también hemos caracterizado el vibrador para un valor que superara mínimamente el valor máximo establecido (3.8V) y hemos observado que para un voltaje mayor (como es 4V) el dispositivo no solo vibra más, sino que empieza a calentarse, cosa que no ocurre hasta 3.8V.

Para el caso del circuito háptico tenemos el inconveniente de que la placa de Arduino genera una señal analógica de 40mA y los vibradores requieren de 85mA al inicio. Por tanto, es necesario hacer uso de un amplificador de corriente: el AD8656.

El Circuito Integrado AD8656 se encargará de amplificar la corriente aportada por los pines analógicos de la placa de Arduino UNO con el fin de lograr los 85mA necesarios para los vibradores [23].

Figura 21 - Circuito integrado AD8656

Consiste en 2 amplificadores de corriente CMOS de bajo coste, un pin de alimentación a 5V y un pin de tierra.

Para la correcta caracterización y posterior diseño del circuito de vibración hemos reproducido en PSpice el circuito integrado AD8656.

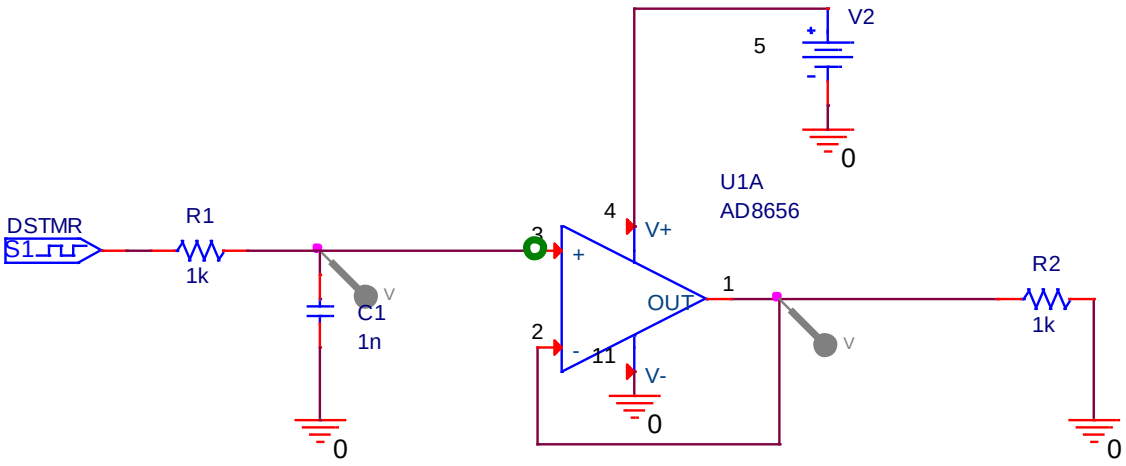

Figura 22 - Circuito de vibración en PSpcie

4.3 Diseño del prototipo

En este apartado trataremos todo lo referente a la programación y el diseño del software utilizado, así como la integración de los circuitos diseñados y utilizados para ensamblar el guante háptico.

Para el desarrollo del proyecto, hemos hecho uso de los lenguajes de programación C++, para la programación de la placa de Arduino, y de #C, para el diseño en Unity de las escenas.

En cuanto a la disposición de los circuitos, veremos detalladamente los puntos en los que se deben colocar cada componente en el guante para aumentar la eficiencia y obtener un mejor resultado.

Dada la limitada cantidad de pines que podemos utilizar de la placa de Arduino UNO, aunque el diseño es extrapolable hasta 8 puntos de vibración, solo hemos diseñado el prototipo para 5 puntos de vibración.

En esencia, el funcionamiento de la solución para nuestro prototipo de guante háptico a nivel de software podemos verla en la Figura 23.

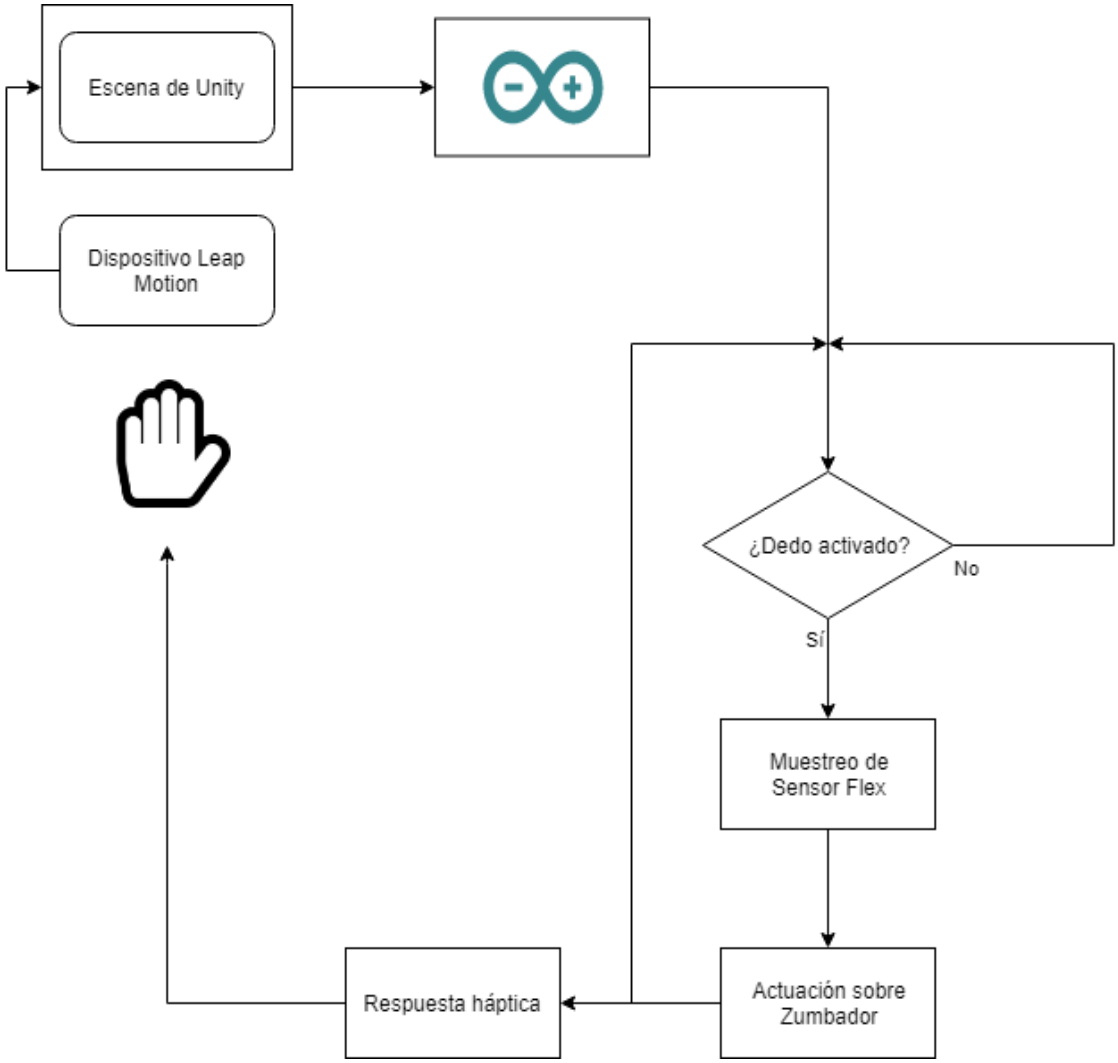

Figura 23 - Diagrama de flujo del software

4.3.1 Arduino

La placa de Arduino será la encargada de muestrear los sensores Flex y controlar la vibración de los zumbadores. En este apartado trataremos todo lo referente a la placa de Arduino, haciendo una revisión de la disposición de los circuitos en la placa y la programación de la misma.

4.3.1.1 Disposición de la placa

La integración de los circuitos comentados en el apartado de Diseño de circuitos se realizará en la placa de Arduino. Para ello, contamos de un guante de lana que servirá de soporte de los sensores Flex y los zumbadores, todos conectados mediante jumpers a una protoboard, donde se alojará el resto del circuito, y a la placa de Arduino.

Gran parte de los componentes han sido soldados para garantizar la fijación de los mismos. Sin embargo, al ser un dispositivo que se ajusta a la mano del usuario, requiere de una preparación previa de algunos elementos, entre ellos, los vibradores.

En cuanto a los pines a los que conectaremos los circuitos, utilizaremos las entradas analógicas A1, A2, A3, A4 y A5 para muestrear los sensores Flex ya que podremos conocer el valor exacto de la resistencia para luego procesar ese dato con el que controlaremos el circuito de vibración.

Para el circuito de vibración, hemos hecho uso de los puertos 5, 6, 9, 10 y 11, los cuales, además de puertos digitales, pueden ser utilizados para la comunicación PWM (Pulse Width Modulation). Como sabemos, las entradas digitales solo aceptan dos valores: en bajo, 0V, o en alto, 5V, pero haciendo uso de PWM, podremos controlar el ciclo de trabajo de estos puertos. Es por eso que el diseño se ha realizado para 5 puntos de vibración, aunque se puede ampliar el diseño.

En la Figura 24 podemos observar la disposición de los componentes para un único dedo, siendo este el diseño el mismo para el resto de los dedos, salvando los pines a los que se conectan en la placa de Arduino.

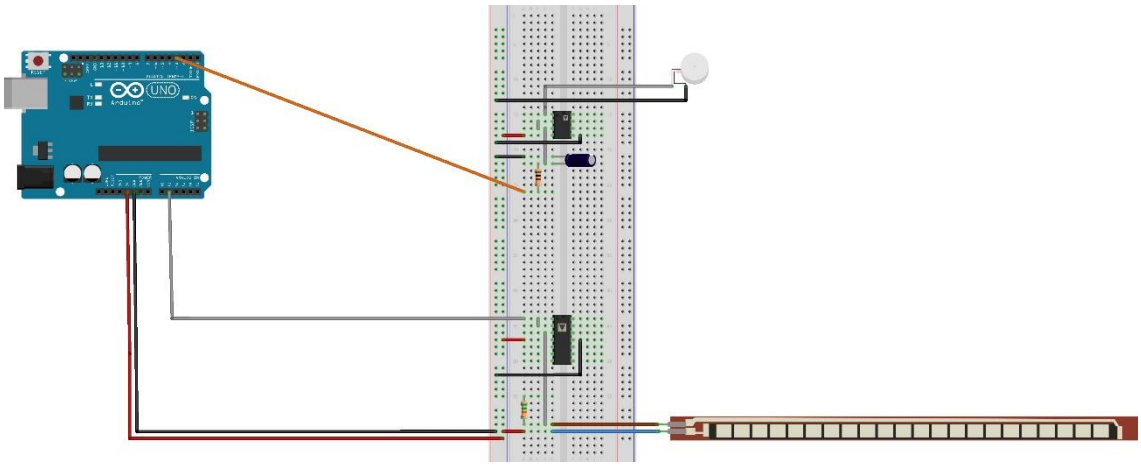

Figura 24 - Disposición de los circuitos en la placa de Arduino para un dedo

La posición de los sensores Flex será en la parte superior de los dedos si la palma se entra apuntando hacia abajo, permitiendo así una mayor movilidad de los dedos del usuario y pudiendo medir de forma más precisa la flexión que si estuvieran en la parte inferior. Podemos observar la posición aproximada en la Figura 25.

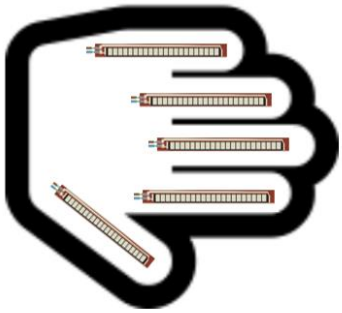

Figura 25 - Disposición de los sensores Flex en el guante háptico

A su vez, la posición óptima de los zumbadores es en la yema de los dedos, permitiendo así obtener una mejor respuesta ante la actuación de los mismos. En particular, el circuito de vibración requerirá de un previo ajuste para asegurar que los vibradores se encuentran en la posición correcta. Podemos observar la posición aproximada en la Figura 26.

Figura 26 - Disposición de los vibradores en el guante háptico

La manera de alimentar la placa de Arduino será conectada al ordenador por el puerto serie del mismo, por el que se alimentará a la placa y se realizará el intercambio de información al tener 4 patillas: una de recepción, una de transmisión, una de alimentación a 5V y otra de tierra.

Cabe destacar que los jumpers utilizados para conectar el guante con la placa de Arduino son cables flexibles y de gran longitud para permitir una mejor movilidad al usuario.

4.3.1.2 Código Arduino

La programación de la placa de Arduino la hemos realizado utilizando el lenguaje de programación C++ además del editor de Arduino. El código, especificado en el Anexo, cuenta con un único archivo para la recepción de datos por puerto serie, muestreo de datos de los sensores Flex y control de los vibradores.

Primero, estableceremos los puertos de la placa de Arduino que utilizaremos en el proyecto.

```
const int FLEX_PIN1 = A1;
const int FLEX_PIN2 = A2;
const int FLEX_PIN3 = A3;
const int FLEX_PIN4 = A4;
const int FLEX_PIN5 = A5;
const int VIBRADOR1 = 5;
const int VIBRADOR2 = 6;
const int VIBRADOR3 = 9;
const int VIBRADOR4 = 10;
const int VIBRADOR5 = 11;
```

Como podemos observar en el código, para el muestreo de los sensores Flex del circuito de flexión hemos utilizado los puertos analógicos de la placa de Arduino (A1, A2, A3, A4, A5). En cuanto al circuito de vibración, hemos hecho uso de los puertos 5, 6, 9, 10 y 11, tal y como comentamos en el apartado de Disposición de la placa.

En el método “set up()” estableceremos estos puertos como entradas y salidas. Además, para la recepción de datos es necesario especificar los baudios a los que recibirá los datos la placa de Arduino, en este caso, a 9600 baudios. Por el puerto serie recibiremos un valor entero obtenido a partir de una ristra de bits procesada en Unity, la cual explicaremos en el apartado de Unity.

```
void setup()
{
    Serial.begin(9600);
    pinMode(FLEX_PIN1, INPUT);
    pinMode (VIBRADOR1, OUTPUT);
    pinMode(FLEX_PIN2, INPUT);
    pinMode (VIBRADOR2, OUTPUT);
    pinMode(FLEX_PIN3, INPUT);
    pinMode (VIBRADOR3, OUTPUT);
    pinMode(FLEX_PIN4, INPUT);
    pinMode (VIBRADOR4, OUTPUT);
    pinMode(FLEX_PIN5, INPUT);
    pinMode (VIBRADOR5, OUTPUT);
    Serial.println("Start");
}
```

Es importante asegurarnos de que los sensores Flex se encuentran calibrados. A pesar de contar con las hojas de características de los sensores Flex, en la práctica, ya sea por la fabricación, por el desgaste o por la mala posición del componente, es necesario

calibrar el valor de ambos valores pico del sensor: la resistencia del sensor Flex estirado y la resistencia del sensor Flex flexionado. Estos valores serán necesarios para muestrear correctamente el valor del sensor Flex y, por tanto, para la actuación de los vibradores.

```
const float BEND_RESISTANCE = 300.0;
const float STRAIGHT_RESISTANCE = 650.0;
```

Para cada bit tendremos asociado un "booleano" para que, una vez el bit está a 1, la actuación del punto del guante háptico sea constante hasta el momento en que pasa a 0 dicho bit.

```
boolean dedo1 = false;
boolean dedo2 = false;
boolean dedo3 = false;
boolean dedo4 = false;
boolean dedo5 = false;
```

Una vez establecidas las variables y función inicial, trataremos ahora el método "loop()" del código.

Primeramente, el valor recibido por el puerto serie lo pasaremos a entero y lo procesaremos como un binario de 8 bits, uno para cada dedo y tres bits para la palma. Estos bits actuarán como "flag" para el muestreo de los sensores y posterior actuación sobre los vibradores.

```
String aux = Serial.readString();
int value_Serial = aux.toInt();
```

Como hemos comentado anteriormente, el muestreo del sensor Flex y la actuación del zumbador permanecerá constante mientras el bit referente a cada punto se mantenga a 1, poniendo a "True" el booleano referente al dedo en cuestión y evitando así el gasto de recursos en muestrear un sensor Flex si el bit referente a su dedo está desactivado.

```
valor = value_Serial & 2;
if (valor == 2){
        dedo2 = true;
}
else {
        dedo2 = false;
}
```

Al poder actuar sobre el entorno virtual con distintos puntos del guante háptico a la vez, haremos uso de una máscara para filtrar el resto de bits para conocer si un bit se encuentra a 1 o a 0. Mediante el uso de operaciones lógicas, seremos capaces de aislar cada bit y podremos conocer así su estado.

Al ser un valor de 8 bits, podremos obtener valores desde 0 hasta 31 para conseguir cada una de las combinaciones de puntos del guante háptico activados, siendo el 0 el valor para desactivar todos los puntos y el 31 el valor para activar todos ellos.

Por otra parte, en la práctica, existen dos valores que hemos: el 0, ya que el puerto serie, cuando no recibe datos, envía de forma automática un 0, y el 50, utilizado como interruptor general para desactivar todos los dedos. De esta forma, al ser el 0 el valor referente a la desactivación de los dedos, se sustituye por el 50.

Al haber calibrado el valor de la resistencia del sensor Flex estirado y el de la resistencia del sensor Flex flexionado, podremos hacer uso de la función "*map()*" para establecer el valor de flexión del sensor Flex.

Aunque el valor muestreado variará cuando variemos la flexión del sensor, al ser este valor el que controlará la vibración de los zumbadores, hemos utilizado los valores 110 y 255 como valores mínimo y máximo del nuevo rango en el que queremos mapear el valor recibido para notar de forma más efectiva la vibración para ángulos de flexión pequeños.

Mediante la utilización de la función "*analogWrite()*" somos capaces de controlar los zumbadores, sin embargo, esta función solo acepta valores entre 0 y 255, es decir, de 8 bits, con los cuales controlaremos el ciclo de trabajo de los zumbadores y su vibración.

Al haber mapeado el valor de los sensores Flex entre 110 y 255, podremos controlar el ciclo de trabajo del zumbador para cada valor tomado por el sensor Flex.

```
if (dedo2 == true){
        Serial.println("2 vibrador");
        int flex1 = analogRead(FLEX_PIN2);
        int flex1_map = map (flex1, STRAIGHT_RESISTANCE, BEND_RESISTANCE, 110,
255);
        Serial.println(flex1_map);
        analogWrite(VIBRADOR2, flex1_map);
}
else {
      analogWrite(VIBRADOR2, 0);
}
```

4.3.2 Unity

En este último apartado referente al Diseño del prototipo, trataremos el desarrollo de la escena en Unity, la creación de la máscara, necesaria para controlar el guante háptico, el intercambio de información y el procesado de datos extraídos del dispositivo Leap Motion.

En nuestro caso, al querer interactuar con el entorno y tener la capacidad de movernos a través de él, hemos utilizado un entorno del tipo Interactivo de los comentados en el apartado de Realidad virtual.

Para el desarrollo, hemos utilizado la versión de Unity 2017.4.19f1 y la versión del SDK de Leap Motion 4.0.0.

Leap Motion nos facilita una API con la que podremos trabajar con el dispositivo Leap Motion. En este caso, utilizaremos el lenguaje de programación C# y el motor de videojuegos multiplataforma Unity.

4.3.2.1 Escena

La creación de la escena 3D para la simulación se ha llevado a cabo creando los diferentes objetos y modelando su forma, posición y textura para conseguir la escena. Se engloba en dos elementos: el escenario y el jugador, además de la luz direccional para iluminar la escena.

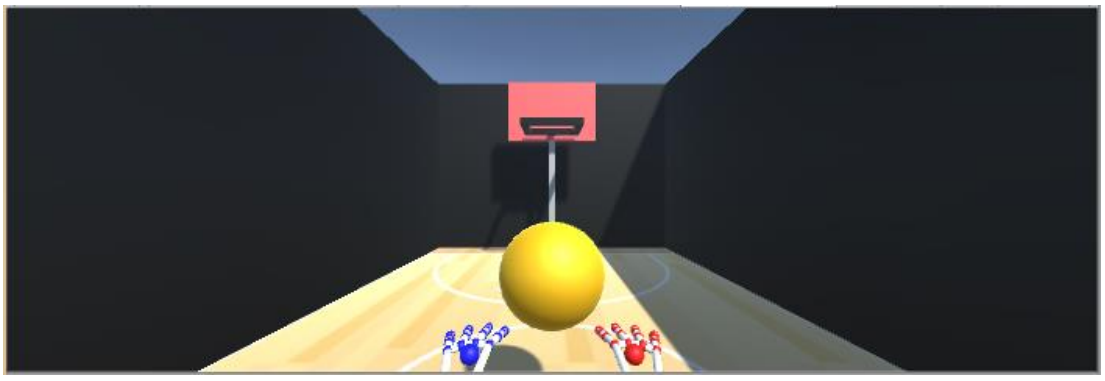

Figura 27 - Escena de Unity

Los objetos propios del escenario, como pueden ser las paredes, el suelo y la canasta, han sido creados con la posición y la rotación fijas para evitar el desplazamiento al provocarse una colisión. Estos objetos tienen los componentes “Rigidbody” y “Collider”.

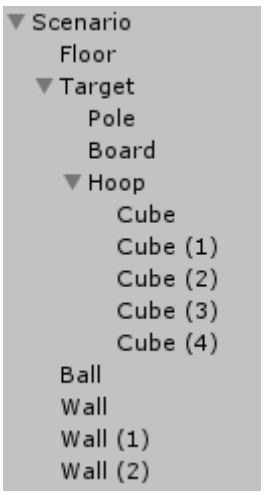

Figura 28 - Jerarquía de la escena de Unity

La pelota “Ball” será el objeto sobre el que actuaremos. Este elemento de la escena ha sido creado de la misma manera que el resto de los objetos 3D, pero con la particularidad de que la posición y la rotación no se encuentran fijadas. De esta forma, al provocarse una colisión, seremos capaces de mover el objeto. La velocidad, la masa y el rozamiento de la pelota pueden ser variados en cualquier momento desde el “Inspector” del objeto.

Por otra parte, el jugador ha sido creado con el “Asset” de “First Person Character”, de tal forma que podemos controlar la posición del jugador con las teclas del ratón, siendo capaces de desplazarnos por la escena. El jugador lleva acoplada la cámara principal con la que podemos ver el entorno según nos desplazamos por el mismo.

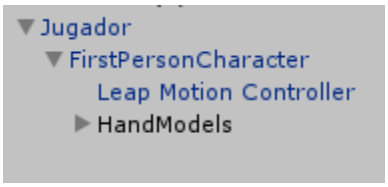

Figura 29 - Jerarquía del Jugador de la escena de Unity

El jugador contiene el controlador de Leap Motion para conseguir un efecto inversivo mayor al utilizar el Leap Motion en el juego ya que veremos las manos como si fueran las nuestras. Los elementos “Leap Motion Controller” y “HandModels” son obtenidos a partir de la API de Leap Motion. Para ello, importaremos el paquete “Core” que podremos encontrar en la página oficial del dispositivo Leap Motion. []

El controlador de Leap Motion es el encargado de manejar los datos ofrecidos por el SDK referentes a la posición de las manos y los dedos.

Por otra parte, el modelo de las manos “HandModels”, elegido entre varios modelos posibles, serán los que nos permitirán visualizar las manos virtuales y dependerán del “Leap Motion Controller”. Como podemos observar en la Figura 30 este modelo de manos se divide a su vez en cuatro componentes: “Capsule Hand Left”, “Capsule Hand Right”, “RigidRoundHand_L” y “RigidRoundHand_R”.

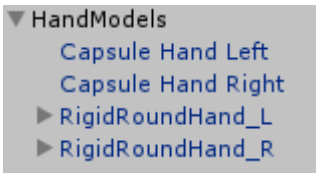

Figura 30 - Jerarquía de las manos de la escena de Unity

Los componentes “Capsule Hand Left” y “Capsule Hand Right” son el modelo de manos que utilizaremos para nuestra escena. Hemos elegido este modelo ya que nos permite visualizar de forma más intuitiva las diferentes conexiones entre los componentes que definen la mano. Podemos observarlo en la Figura 31.

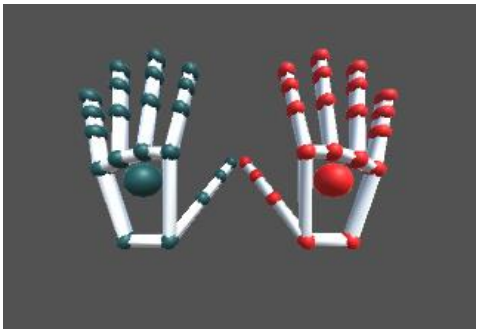

Figura 31 - Forma de "Capsule Hand"

Los componentes “RigidRoundHand_L” y “RigidRoundHand_R” contendrán las diferentes partes que conforman la mano, así como las características físicas de las mismas y contienen el componente “Collider”. Estos componentes se dividen a su vez en los diferentes dedos, la palma de la mano y el antebrazo.

Figura 32 - Jerarquía de la mano izquierda de "Capsule Hand"

A su vez, cada dedo se compone de tres huesos (“bone1”, “bone2” y “bone3”). Estos componentes contendrán el componente “Collider”, que en este caso será muy importante para el correcto funcionamiento del guante háptico, además del “Script” modificará el valor de la ristra que se enviará, el cual comentaremos más adelante.

Cabe destacar que la forma en la que está diseñada cada mano del Leap Motion evita que se produzcan colisiones entre los diferentes componentes de cada mano. Es decir, cada mano se compone de distintos cilindros y esferas que conforman los dedos, la palma y el antebrazo, pero Unity entiende el conjunto de todos estos elementos 3D como un componente entero, por lo que no genera colisiones.

4.3.2.2 Código Máscara

En este apartado trataremos el “Script” Máscara que efectuará el intercambio de información con la placa de Arduino a través del puerto serie. Este “Script” estará alojado en el Jugador y en continuo intercambio de información con los dedos. El código completo se encuentra en el Anexo.

Primeramente, deberemos configurar la conexión con el puerto serie. Deberemos crear la conexión en la cual especificaremos el puerto, los baudios a los que se transmitirán la información, el protocolo de comprobación de la paridad, la longitud estándar de los bits de datos por byte y los bits de parada. Además, en la propiedad “Start()” estableceremos el tiempo de espera a 100 ms.

```
SerialPort sp = new SerialPort("COM5", 9600, Parity.None, 8, StopBits.One);

public void Start()
  {
    sp.ReadTimeout = 100;
  }
```

Es importante asegurarnos de que el Arduino se encuentra conectado en el puerto especificado, en nuestro caso, el COM5.

Por otra parte, estableceremos la variable privada “ristra” que contendrá el valor que queremos transmitir por el puerto serie. A esta variable accederemos con las propiedades “get()” y “set(int valor)”.

La propiedad “get()” nos dará el valor que tiene la ristra para hacer el procesado de la misma en el “Script” Valor_dedo.

```
public int getRistra()
  {
    return ristra;
  }
```

La propiedad “set(int valor)” será la que actualice el valor de la ristra y lo envíe como “String” por el puerto serie que hemos abierto previamente. Primero, nos aseguraremos de abrir el puerto cada vez que mandemos un valor. Asimismo, al finalizar el envío cerraremos la conexión entre Unity y Arduino cerrando el puerto.

Como comentamos en el apartado de Arduino, el puerto serie envía un valor 0 constantemente si no tiene ningún valor que recibir, por lo que sustituiremos ese valor por el 50 para evitar que los dedos se desactiven sin requerirlo.

```
public void setRistra(int valor)
  {
    sp.Open();
    this.ristra = valor;
    if (this.ristra == 0)
    {
      sp.Write(50.ToString());
      Debug.Log(50.ToString());
    }
    else
    {
      Debug.Log(this.ristra.ToString());
```

```
            sp.Write(this.ristra.ToString());
        }
        sp.Close();
    }
```

Cabe destacar que, al estar usando la versión de Unity 2017.4.19f1, será necesario importar el paquete “UnitySerialPort” que obtendremos en el repositorio Github [24].

4.3.2.3 Código Valor_dedo

En este apartado trataremos el “Script” Valor_dedo que se encarga de la actualización del valor de la ristra que se enviará por el puerto serie. Este “Script” se encuentra alojado en el “bone3” de cada dedo, es decir, la falange exterior y en continuo intercambio de información con el valor de la ristra del Jugador. El código completo se encuentra en el Anexo.

El “Script” contiene la variable pública “valor” que contendrá un número entero diferente para cada dedo entre 1, 2, 4, 8 y 16. Este valor, al ser público, lo podremos especificar en el Inspector de Unity, pudiendo utilizar el mismo “Script” para todos los dedos.

Además, contendrá un “GameObject” que instanciará al Jugador, donde se encuentra la ristra que actualizaremos.

El “Script” se ejecutará en el momento en que se detecte una colisión efectuada por el dedo que realiza dicha colisión con el objeto virtual.

Al colisionar, en la propiedad “OnCollisionEnter(Collision collision)” obtendremos el valor actual de la ristra por medio de la propiedad “get()” que hemos comentado en el apartado de Código Máscara. Una vez obtenida, haremos uso de la operación lógica OR para insertar el valor del dedo en la ristra, es decir, activaremos el bit número 1, 2, 3, 4 o 5 respectivamente que actuarán de “flag” en el código de Arduino.

Po último en esta propiedad, utilizaremos la propiedad del “Script” Máscara “setRistra(int valor)” para actualizar el valor de la ristra del Jugador y enviar el valor por el puerto serie para activar el dedo.

```
public void OnCollisionEnter(Collision collision)
    {
        ristra = Jugador.GetComponent<Mascara>().getRistra();
        Debug.Log("colisionando on" + valor);
        Jugador.GetComponent<Mascara>().setRistra(ristra | valor);
    }
```

En el momento en que detectemos que se finaliza la colisión, se ejecutará la propiedad “OnCollisionExit(Collision collision)”. De nuevo, obtendremos el valor de la ristra por medio de la propiedad “get()”, la cual tendrá el bit activado referente al dedo que ha

colisionado y que queremos desactivar. Una vez obtenida la ristra y haciendo uso de la operación lógica NAND, seremos capaces de desactivar el bit referente al dedo que finaliza la colisión al tener el valor del dedo.

El último paso de esta propiedad será actualizar el valor de la ristra del Jugador por medio de la propiedad “setRistra(int valor)” y mandarlo por el puerto serie para desactivar el dedo.

```
public void OnCollisionExit(Collision collision)
  {
     ristra = Jugador.GetComponent<Mascara>().getRistra();
     Debug.Log("colisionando off" + valor);
     Jugador.GetComponent<Mascara>().setRistra(ristra & ~valor);
  }
```

5 Pruebas y resultados

En este apartado final, comentaremos los resultados obtenidos en las pruebas realizadas con el prototipo, los problemas que surgieron y veremos una breve introducción de las diferentes líneas futuras que pudiera seguir el proyecto, aunque se comentarán más a fondo en el apartado de Futuras líneas.

Por una parte, el apartado de Pruebas y resultados del prototipo engloba el funcionamiento del prototipo diseñado y la recepción de datos por parte de la placa para distintos casos posibles. A su vez, el apartado de Pruebas y resultados del entorno virtual abarca el funcionamiento de la escena de Unity y del dispositivo Leap Motion así como el envío de información al puerto serie para distintos casos posibles.

Para ello, al ser dos desarrollos que conllevan varias pruebas de funcionamiento y con funciones distintas, separaremos el análisis de resultados en los obtenidos con el prototipo y los obtenidos con respecto al entorno virtual.

Para finalizar, trataremos los resultados obtenidos de manera conjunta al integrar ambas soluciones que integra el proyecto.

Debemos tener en cuenta que uno de los objetivos propuestos para este proyecto era el de realizar un prototipo de guante háptico de bajo coste y que trabajamos con tecnología nueva de la que no hay demasiadas aplicaciones en el campo de la tecnología háptica, como puede ser el dispositivo Leap Motion. Por tanto, aunque los resultados son satisfactorios, pueden surgir errores derivados de la integración de distintas tecnologías, lenguajes de programación y de los componentes utilizados.

5.1 Pruebas y resultados del prototipo

Cabe destacar que, como hemos comentado en el apartado de Diseño del prototipo, la placa de Arduino solo nos permitía desarrollar hasta 5 puntos de vibración, es por eso que, aunque se puede utilizar el mismo diseño para más puntos de vibración, en nuestro prototipo solo hemos implementado los dedos.

Una vez ensamblado el prototipo del guante háptico, tuvimos que centrarnos en el correcto funcionamiento de la transmisión de información por el puerto serie así como el correcto funcionamiento del prototipo para la actuación de forma simultánea de los diferentes dedos que actúen en el entorno virtual.

Para comprobar su funcionamiento, hicimos uso del monitor serie del que cuenta la API de Arduino para simular distintos posibles casos de actuación.

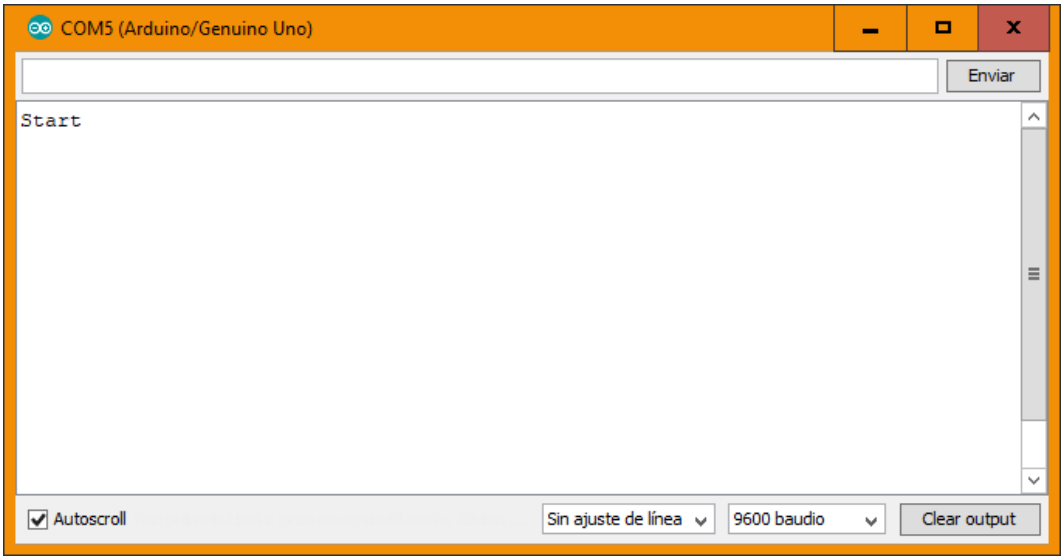

Figura 33 - Monitor serie de la API de Arduino

Mediante esta herramienta, podemos enviar datos por el puerto serie que serán recibidos en la placa de Arduino tal y como funcionaría si fuera el entorno virtual el que enviase estos datos.

A continuación, podemos observar en la Figura 34 el resultado obtenido en caso de enviar “1” por el puerto serie, es decir, en caso de activar el primer dedo.

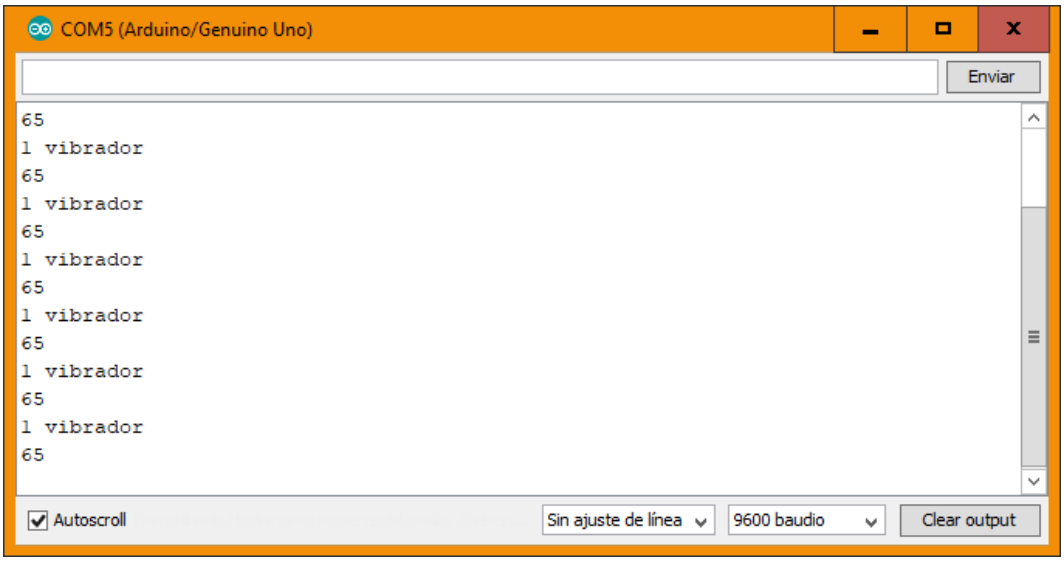

Figura 34 - Actuación del prototipo para valor "1" enviado

Como podemos observar, el muestreo de los sensores Flex y la respectiva actuación de los vibradores, ocurrirá de forma continuada hasta enviar “50” por el puerto serie, tal y como especificamos en el apartado de Arduino.

Asimismo, para el caso de enviar un “2”, un “4”, un “8” o un “16”, actuarán los otros dedos respectiva e individualmente.

Sin embargo, como hemos comentado, debemos asegurar que el funcionamiento es correcto para cualquier valor enviado, es decir, que actúen correctamente todos los dedos para cualquier combinación.

En la Figura 35 podemos observar el funcionamiento del prototipo en el caso de enviar un “11”, que en binario es 00001011, es decir, actuarán los dedos 1, 2 y 4, muestreando y controlando la vibración de sus respectivos zumbadores de forma continuada hasta mandar un “50” por el puerto serie.

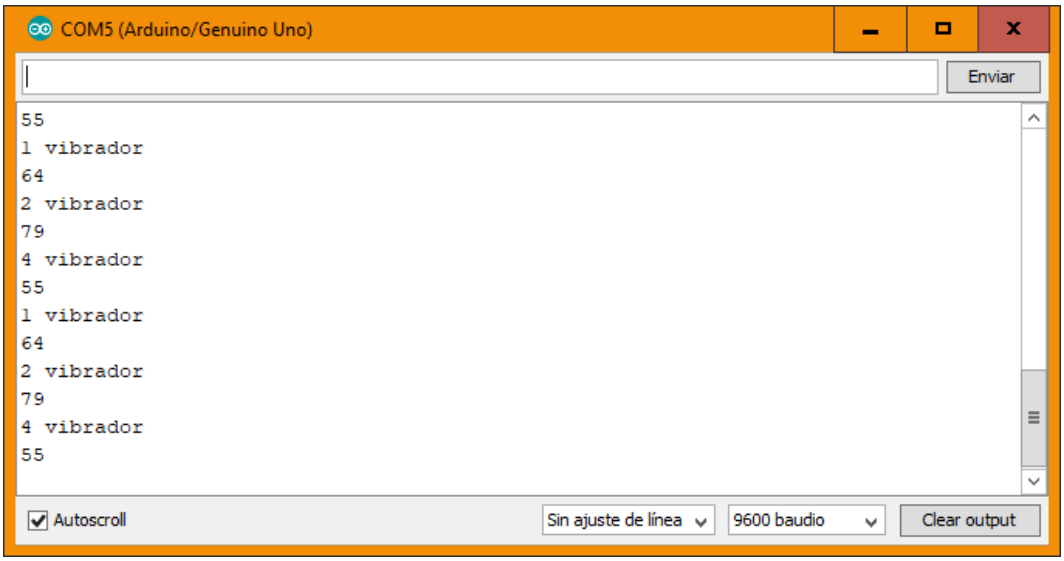

Figura 35 - Actuación del prototipo para valor "11" enviado

Con una ristra de 8 bits, de los cuales utilizamos 5 referentes a cada uno de los dedos, se puede obtener hasta 31 combinaciones, todas ellas plenamente funcionales.

Naturalmente, no podemos dejar tratar en este informe el funcionamiento de todas y cada una de las combinaciones, pero hay una en particular que cabe destacar, y es el caso de enviar un “31” por el puerto serie, lo que implica la actuación simultanea y sin retardo de los 5 dedos.

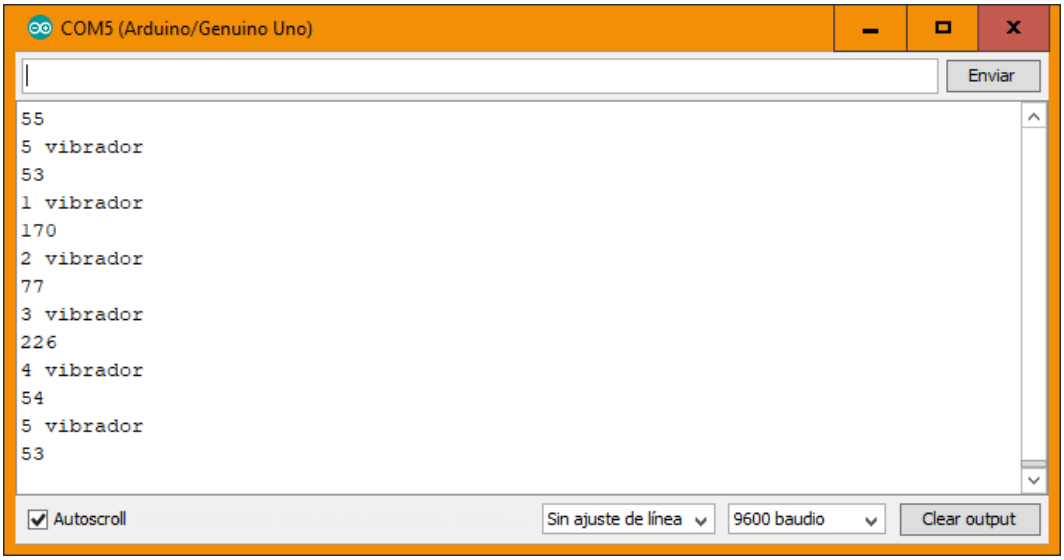

Figura 36 - Actuación del prototipo para valor "31" enviado

Como podemos observar en la Figura 36, los dedos 1 y 3 se encontraban flexionados, lo que produce un aumento de la vibración del zumbador.

Cabe destacar que, incluso para el caso límite de la actuación simultanea de todos y cada uno de los dedos, el funcionamiento del prototipo no se ve afectado en cuanto a retardos y continúa funcionando, muestreando y actuando de manera normal. Si bien es cierto que se producirá un retraso, este es inapreciable.

Por tanto, queda patente el correcto funcionamiento del prototipo del guante háptico para cualquier posible combinación de dedos actuando simultáneamente y para cualquier flexión ejercida sobre los sensores Flex.

5.2 Pruebas y resultados del entorno virtual

Aunque Leap Motion era el dispositivo de los estudiados en el apartado del Estado del Arte que más fiabilidad y mejores resultados nos aportaba, al ser una tecnología relativamente nueva y aún en desarrollo, tiene algunos fallos que afectan al correcto funcionamiento del proyecto.

En ocasiones, el dispositivo Leap Motion se descalibraba provocando que el controlador de Leap Motion de la escena de Unity realizara movimientos que no se correspondían con los ejecutados en la realidad. Para atajar este problema, es importante tener en cuenta la orientación del dispositivo Leap Motion ya que, para movimientos fuera de su rango de actuación o con falta de visibilidad, nos provocará este descalibrado.

Por otra parte, como ya hemos comentado, los dispositivos de captura de movimiento pueden tener algunos fallos a la hora de captar la flexión de los dedos con total precisión. Con la utilización de los sensores Flex, conseguimos solventar dicho problema que podría acarrear fallos en el funcionamiento, permitiendo así tener una doble verificación del estado de flexión de los dedos.

Una vez creada la escena de Unity, debíamos asegurar el correcto funcionamiento de la ristra que controlaría el funcionamiento del prototipo para cualquier comportamiento del modelo de las manos de Leap Motion con el entorno virtual.

En la práctica, hemos fijado la posición y la rotación de la pelota “Ball” sobre la que actuamos para comprobar el funcionamiento del prototipo ya que, al provocarse una sola colisión, se producirá un movimiento de la pelota “Ball”, por tanto, al fijar la posición y la rotación de la misma podemos ver el correcto funcionamiento del prototipo para cualquier cantidad de dedos actuando sobre los elementos de la escena.

A continuación, podemos observar diferentes casos para distintas combinaciones de dedos actuando sobre la pelota “Ball” y comprobar que, efectivamente, se envía por el puerto serie el valor correcto.

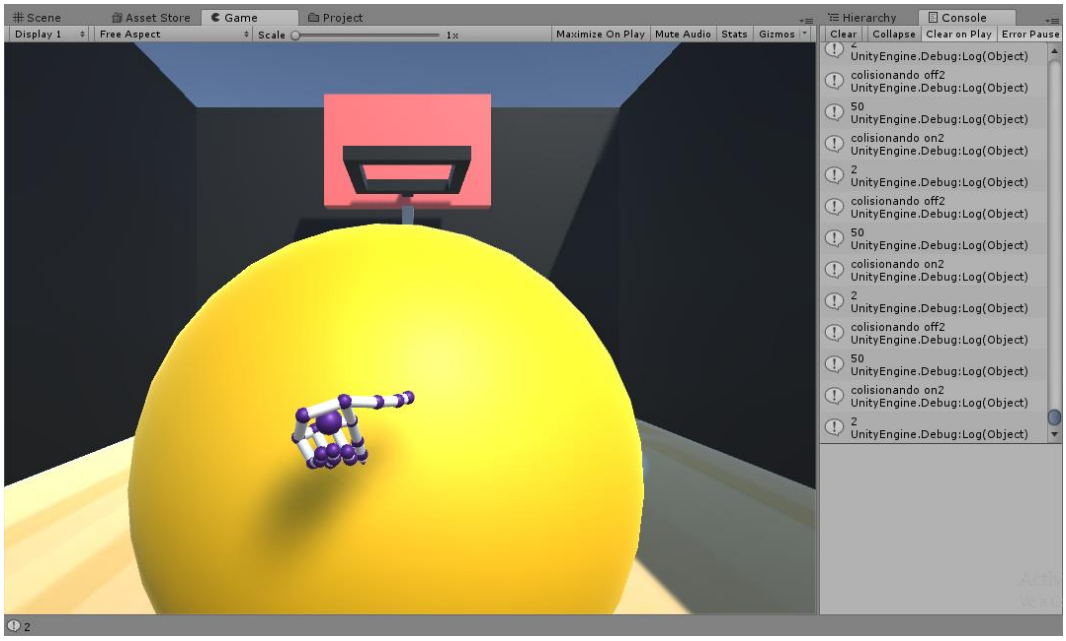

Figura 37 - Inicio de la colisión de un dedo con la escena

Como podemos ver en la Figura 37, al actuar el segundo dedo, el índice de la mano izquierda, se enviará por el puerto serie “2”, lo que se traducirá en la respectiva actuación del dedo índice en el prototipo del guante háptico.

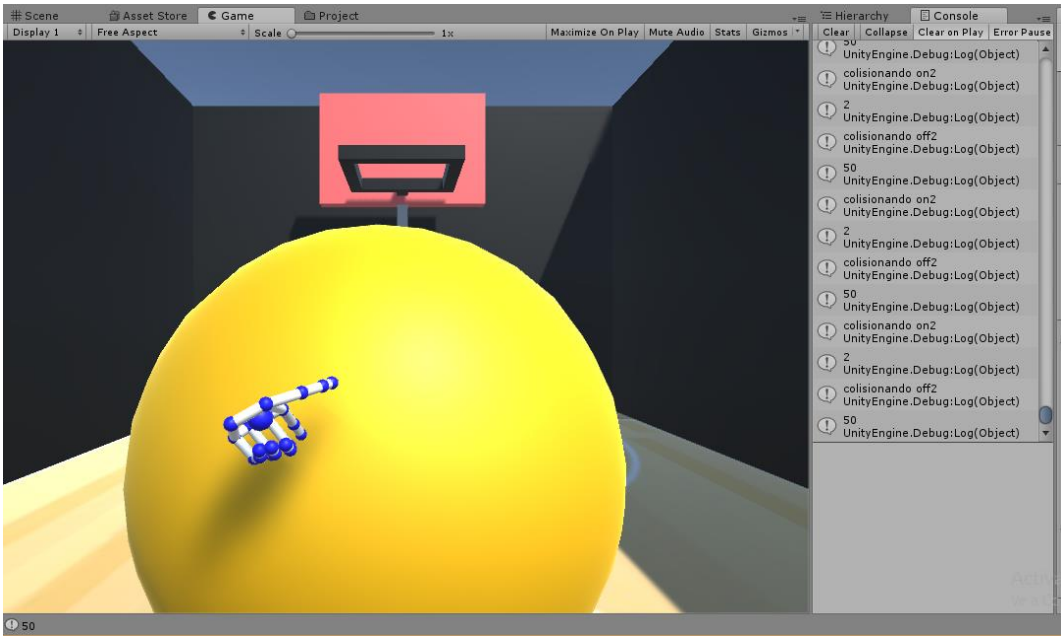

Figura 38 - Finalización de la colisión de un dedo con la escena

Asimismo, al finalizar la colisión con la pelota “Ball”, se enviará “50” para desactivar el dedo índice, como podemos ver en la Figura 38.

A su vez, en la Figura 39 podemos ver el caso de varios dedos actuando sobre la pelota “Bal”. En este caso, actúan todos los dedos salvo el pulgar, que corresponde al primer dedo y, por tanto, al “1”. Efectivamente, al actuar el resto de los dedos, la escena envía “30” por el puerto serie, lo que implica la actuación de todos los dedos excepto el pulgar.

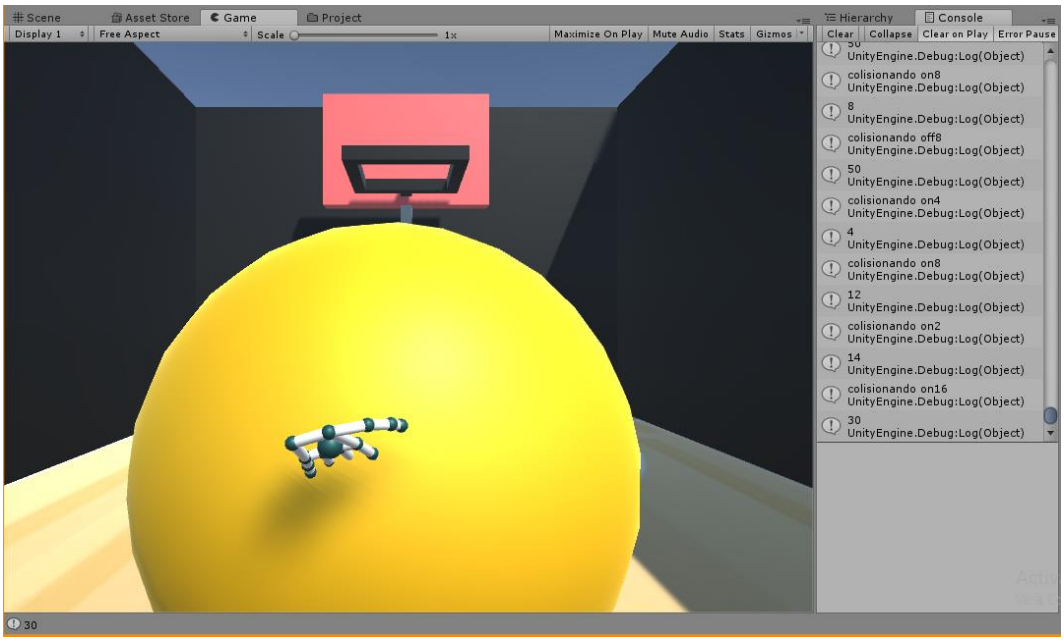

Figura 39 - Colisión de varios dedos con la escena

Por tanto, queda demostrado el correcto funcionamiento de la escena de Unity así como el funcionamiento de la ristra que se enviará para controlar la actuación del prototipo de guante háptico.

5.3 Pruebas y resultados de integración

Por último, en cuanto a las pruebas y resultados obtenidos de la puesta en práctica del prototipo, comentaremos los resultados de la integración de ambas partes: el prototipo del guante háptico aplicado al entorno virtual diseñado.

En los anteriores apartados corroboramos que ambos desarrollos por separado funcionan correctamente, sin embargo, en la integración aparecen problemas de retraso en el envío de información por el puerto serie y, por tanto, en la consiguiente respuesta del prototipo.

Este retraso en el funcionamiento se debe principalmente al ordenador que soporta el proyecto por completo ya que, al alojar tanto la escena de Unity como la API de Leap Motion, requiere una CPU que pueda soportar ambos procesos y mantener el requerimiento de que sea un proceso en tiempo real.

A su vez, el dispositivo Leap Motion funciona correctamente en la integración con el prototipo de Leap Motion, salvando los problemas derivados de la posición de las manos y la perspectiva del dispositivo.

Cabe destacar, así mismo, que ambos desarrollos funcionan de manera correcta al ser utilizados de manera continuada en un largo periodo de tiempo. El único problema que ha surgido en esta prueba ha sido con el dispositivo Leap Motion que con el uso acaba por descalibrarse, lo que requerirá un recalibrado al querer usarlo de nuevo.

Por otra parte, en las pruebas de integración corroboramos que la vista es el sentido que más afecta a la percepción que recibimos ya que, aunque se intentase sugestionar al usuario de que estaba tocando un objeto en el entorno virtual, si no había coherencia con la vista, la capacidad de percibir la forma de los objetos se veía reducida. Asimismo, si se daba el caso contrario de que se percibía la interacción con el objeto por medio de la vista, pero el tacto no se encontraba sincronizado, la capacidad de inmersión aportada por el guante háptico se perdía, pero no se reducía la capacidad de percibir los objetos.

Además, para confirmar la eficiencia en la utilización de esta tecnología en los entornos virtuales, se estableció una prueba mediante la cual se varió la forma del “Collider” del objeto con el que el usuario colisionaría, así como su forma, coincidiendo ambos en un 10% de los casos de prueba. A su vez, la visión estaba distorsionada, siendo incapaz de ver con nitidez la escena el usuario.

En este supuesto, el usuario se basaba en el tacto para identificar cada objeto apoyándose en la vista de una forma mucho menor, siendo capaz de identificar correctamente los objetos haciendo uso del prototipo del guante háptico mientras era incapaz de hacerlo sin la utilización de la tecnología háptica para obtener información del entorno con mayor certeza.

6 Conclusiones

Uno de los objetivos de este proyecto era el de realizar el prototipo de guante háptico con la utilización de materiales de bajo coste. Tal y como ha quedado patente en el apartado de Pruebas y resultados, este objetivo ha sido cumplido. Sin embargo, la utilización de estos materiales, aunque válida, no ofrece resultados lo suficientemente satisfactorios como para utilizar el prototipo en un ámbito más allá del docente debido a la probabilidad de error de los componentes.

A su vez, uno de los problemas que surgen de la utilización del dispositivo Leap Motion y, en general, de cualquier dispositivo de captura de movimiento con procesamiento de imagen, es el error cometido en la captura de los dedos en estado de flexión, siendo incapaces de medir con precisión dicha flexión. Mediante la utilización de los sensores Flex hemos conseguido atajar dicho problema, aumentando así la seguridad en la utilización de estos dispositivos de captura de movimiento con una solución de bajo coste.

Cabe destacar que el diseño del guante háptico es perfectamente extrapolable a cualquier número de puntos de vibración, sin embargo, para ello es necesario la utilización de microcontroladores con un mayor número de salidas PWM con las que controlar la actuación de los zumbadores.

Aunque el diseño del prototipo del guante háptico ha sido realizado para conseguir los mejores niveles de comodidad del usuario, al ser un desarrollo que requiere de una conexión continua, estable y rápida, la placa de Arduino debía ser conectada por puerto serie, por lo que se decidió utilizar cables flexibles y de gran longitud. Estos cables permiten una alta movilidad al usuario, sin embargo, limita el área de movilidad del usuario a la longitud de dichos cables.

Como conclusión de este proyecto extraemos que la tecnología háptica se encuentra aún en pleno desarrollo de sus capacidades y que su utilización nos permite obtener una mayor cantidad de información de un entorno virtual, además de la posibilidad de realizar procesos de manera más segura, como la práctica de operaciones o el aprendizaje de la utilización de elementos de corte industriales. Es cierto, tal y como comprobamos en el apartado de Pruebas y resultados de integración, que la vista continúa aportando información más relevante para el cerebro y que afecta más a la percepción que tenemos del entorno, pero también comprobamos que la detección de objetos se ve acelerada al integrar ambos sentidos en la experiencia de usuario ya que, efectivamente, la vista nos aporta información más cualitativa, pero, a falta de ella o con información limitada, es el tacto el que nos permite analizar el entorno.

Además, los dispositivos que se encuentran actualmente en el mercado son caros aún imprecisos y, al igual que la propia tecnología háptica, con una amplia capacidad de mejora. Por tanto, como última conclusión al proyecto queda remarcar la viabilidad del desarrollo de tecnología háptica con componentes de bajo coste, permitiendo así llegar a una mayor cantidad de usuarios.

7 Futuras líneas

En cuanto a las futuras líneas que se abren respecto al proyecto y, en particular, al prototipo, cabe destacar la necesaria optimización del mismo por medio de la utilización de componentes más precisos, además de mejorar la capacidad de movimiento del usuario que hace uso del prototipo.

Por otra parte, para mostrar el verdadero potencial de los guantes hápticos será necesario definir unos casos de uso precisos que conllevarán, entre otras cosas, el diseño de nuevas escenas virtuales para llevar a la práctica esta tecnología.

Además, la futura línea más importante será integrar los distintos sentidos para conseguir un efecto inmersivo más realista para el usuario, permitiendo así corroborar el aumento de la eficiencia al usar una mayor cantidad de sentidos.

Ambas tecnologías; la háptica y la de captura de movimiento con procesamiento de imagen, se encuentran en una fase relativamente temprana y con mucho margen de mejora. Es por ello que, en las posibles líneas de negocio en las que pueden encajar ambas tecnologías, como puede ser la medicina, requerirán de una fiabilidad y una eficiencia mayor a la que obtenemos con la tecnología actual.

El actual desarrollo del “Deep Learning” y la Inteligencia Artificial, permitirá obtener mejores resultados en el análisis y procesamiento de imágenes, consiguiendo así un gran avance en el área, ya sea integrando modelos entrenados para corregir errores o haciendo más eficiente el reconocimiento de objetos.

Por tanto y como última futura línea, destacaría la necesaria integración de modelos de Inteligencia Artificial tanto en la tecnología háptica como en la tecnología de captura de movimiento para conseguir así una mayor seguridad en las aplicaciones y expandir el mercado al que está dedicado.

Bibliografía

[1] «Computer Science Department». .

[2] A. Vélez Escorial, «Diseño mecánico de un interfaz háptico para realidad virtual», feb. 2011.

[3] «El cerebro puede reconocer los objetos mediante el tacto / Noticias / SINC». [En línea]. Disponible en: https://www.agenciasinc.es/Noticias/El-cerebro-puede-reconocer-los-objetos-mediante-el-tacto. [Accedido: 12-jun-2019].

[4] «Conferences | EuroHaptics». .

[5] «Investigadores de Disney añaden funciones táctiles a todo tipo de objetos cotidianos... ¡hasta peceras! (con vídeo)», *Engadget*. [En línea]. Disponible en: https://www.engadget.com/es/2012/05/06/investigadores-de-disney-anaden-funciones-tactiles-a-todo-tipo-d/. [Accedido: 10-jun-2019].

[6] «La tecnología REVEL de Disney podría convertir toda tu casa en una pantalla táctil», *Engadget*. [En línea]. Disponible en: https://www.engadget.com/es/2012/08/10/revel-disney-tactil/. [Accedido: 10-jun-2019].

[7] ImmersiveTouch, «Home», *Home*. [En línea]. Disponible en: https://www.immersivetouch.com/. [Accedido: 10-jun-2019].

[8] «DextrES, unos guantes hápticos de solo 8 gramos con respuesta de fuerza». [En línea]. Disponible en: https://www.realovirtual.com//noticias/5863/dextres-unos-guantes-hapticos-solo-8-gramos-respuesta-fuerza. [Accedido: 21-may-2019].

[9] «Plexus: precio, características y lanzamiento de los guantes hápticos», *España Virtual*, 11-jul-2018. .

[10] «Glove One: así es el innovador guante almeriense que permite tocar y sentir la realidad virtual», *ELMUNDO*, 29-oct-2017. [En línea]. Disponible en: https://www.elmundo.es/tecnologia/2017/10/29/59f624a0e2704e447b8b45b5.html. [Accedido: 21-may-2019].

[11] «Developing With Myo – Welcome to Myo Support». [En línea]. Disponible en: https://support.getmyo.com/hc/en-us/categories/200376235. [Accedido: 09-jun-2019].

[12] «Home - nod». [En línea]. Disponible en: https://nod.com/. [Accedido: 09-jun-2019].

[13] «Leap Motion». [En línea]. Disponible en: https://www.leapmotion.com/. [Accedido: 09-jun-2019].

[14] «Teach, Learn, and Make with Raspberry Pi – Raspberry Pi». [En línea]. Disponible en: https://www.raspberrypi.org/. [Accedido: 18-jun-2019].

[15] «Arduino - Home». [En línea]. Disponible en: https://www.arduino.cc/. [Accedido: 09-jun-2019].

[16] «WebVR - Bringing Virtual Reality to the Web». [En línea]. Disponible en: https://webvr.info/. [Accedido: 18-jun-2019].

[17] «What is Unreal Engine 4». [En línea]. Disponible en: https://www.unrealengine.com/en-US/what-is-unreal-engine-4. [Accedido: 18-jun-2019].
[18] «Unity». [En línea]. Disponible en: https://unity.com/es. [Accedido: 18-jun-2019].
[19] «C# SDK Documentation — Leap Motion C# SDK v2.3 documentation». [En línea]. Disponible en: https://developer-archive.leapmotion.com/documentation/v2/csharp/index.html?proglang=csharp. [Accedido: 18-may-2019].
[20] «Home - Precision Microdrives». [En línea]. Disponible en: https://www.precisionmicrodrives.com/. [Accedido: 09-jun-2019].
[21] «Custom Linear Potentiometer Manufacturer | Spectra Symbol». [En línea]. Disponible en: https://www.spectrasymbol.com/. [Accedido: 09-jun-2019].
[22] «Analog, Embedded Processing, Semiconductor Company, Texas Instruments - TI.com». [En línea]. Disponible en: http://www.ti.com/. [Accedido: 09-jun-2019].
[23] «Mixed-signal and digital signal processing ICs | Analog Devices». [En línea]. Disponible en: https://www.analog.com/en/index.html. [Accedido: 09-jun-2019].
[24] prossel, *Script to work with serial port in Unity. Optimized to work with TSV, CSV or other character delimited values: prossel/UnitySerialPort*. 2019.

Anexo

I. Arduino UNO

Como comentamos en el apartado del Estado del Arte, la placa de Arduino será la encargada de procesar la información captada por el Leap Motion para controlar el guante en función del entorno virtual.

Arduino Uno es una placa electrónica basada en el microcontrolador ATmega328. Cuenta con 14 entradas/salidas digitales, de las cuales 6 se pueden utilizar como salidas PWM y otras 6 son entradas analógicas. Además, incluye un resonador cerámico de 16 MHz, un conector USB, un conector de alimentación, una cabecera ICSP y un botón de "reset" [18]. La placa de Arduino UNO se programa con el lenguaje de programación C++ y con el IDE proporcionado por Arduino

a. Características técnicas

Microcontrolador	ATmega328
Voltaje de funcionamiento	5V
Voltaje de entrada recomendado	7-12V
Voltaje de entrada límite	6-20V
Pines digitales de E/S	14
Pines analógicos de entrada	6
Corriente DC pines E/S	40 mA
Corriente DC para pines a 3.3V	50 mA
Memoria flash	32 KB
SRAM	2 KB
EEPROM	1 KB
Velocidad de reloj	16 MHz
Longitud	68.6 mm
Anchura	53.4 mm
Peso	25 g

Tabla 4 - Características técnicas de placa Arduino UNO

II. Leap Motion

El módulo Leap Motion será el encargado de captar la información de la posición en el espacio de las manos para luego transmitirlas al ordenador y poder observarlas en el entorno virtual.

A continuación, vamos a estudiar las características del hardware y del software del dispositivo Leap Motion, así como su funcionamiento.

a. Hardware

El dispositivo Leap Motion se trata de un periférico de forma rectangular, de dimensiones 75x25x11 mm (largo, ancho y alto) y de peso aproximado de 50g. Leap Motion cuenta con dos cámaras con sensores ópticos o fotoeléctricos, un microcontrolador y tres LEDs.

Figura 40 - Dispositivo Leap Motion desmontado

Las cámaras son la parte más importante del dispositivo. Su correcto funcionamiento y estado es determinante en la captura de datos. Cada cámara cuenta con un sensor óptico monocromático CMOS sensible a la luz infrarroja (con una longitud de onda de 850nm) capaz de detectar la presencia de objetos basándose en la variación de la intensidad de la luz. Estos sensores fotoeléctricos funcionan a una velocidad de hasta 200fps, dependiendo del rendimiento del ordenador al que está conectado el Leap Motion. Además, cada uno cuenta con una lente para poder recibir toda la luz de la zona de cobertura.

El microcontrolador se trata de un circuito integrado que hace la función de BIOS. Este microcontrolador será el encargado de regular la iluminación y transmitir la información captada al driver instalado en el ordenador al que se conecte el dispositivo Leap Motion.

Los LEDs son los encargados de iluminar la zona de cobertura del Leap Motion por inundación, trabajando en el espectro de luz infrarroja de 850nm a la que son sensibles los sensores ópticos de las cámaras. Estos sensores asegurarán que la imagen tenga un mismo nivel de luz constantemente, variando su consumo en función de la cantidad de luz de la zona de cobertura, es decir, si estamos en un entorno oscuro, el consumo será mayor.

Figura 41 - Cámaras del dispositivo Leap Motion

Para asegurar que los sensores no sufren una saturación de luz, se colocan unas barreras de plástico. De esta manera, además, se asegura que la iluminación de la zona sea uniforme.

Además, Leap Motion soporta USB 2.0 y 3.0 para la conexión al ordenador. Cabe destacar que se obtendrán diferentes velocidades de transmisión dependiendo de las características y el rendimiento del ordenador al que se conecte.

El rango de actuación efectivo de Leap Motion es desde 0.025m hasta 0.6m, consiguiendo un campo de 150º.

Figura 42 - Rango de actuación del dispositivo Leap Motion

Leap Motion utiliza el sistema Cartesiano como sistema de coordenadas. Teniendo en cuenta que el dispositivo ha sido diseñado para colocarse de manera horizontal, es decir, con los sensores hacia arriba, el sistema de coordenadas quedaría como en la Figura 43, siendo solo posible obtener valores positivos del eje Y, como podemos observar.

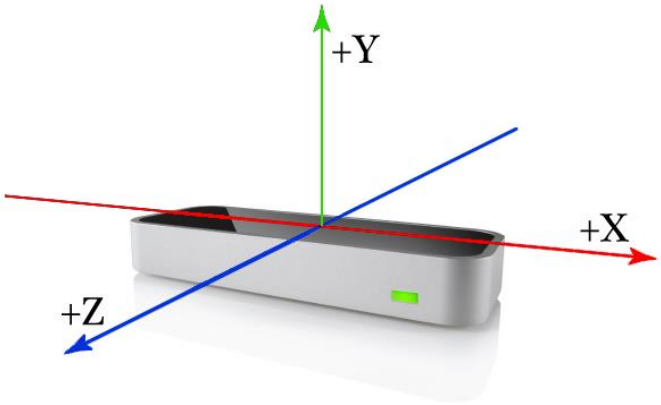

Figura 43 - Sistema de coordenadas del dispositivo Leap Motion

Sin embargo, al ser el dispositivo tan manejable, es fácil de cambiar su orientación, pero tendremos que tener en cuenta que deberá ser modificada también en el software para tomar valores relativos más exactos.

b. Software

Leap Motion nos ofrece un SDK para el manejo de la información captada por el dispositivo. Esta API puede ser programada en los lenguajes Java, C++, C#, Python, JavaScript, Objective-C, Unity y Unreal.

Primeramente, haremos un repaso de las principales características del SDK, pero en este proyecto nos centraremos en los lenguajes C++ y C#, como establecimos en el apartado del Estado del Arte [19].

El SDK de Leap Motion se encarga de analizar cada fotograma captado por el dispositivo en el cual se encuentren las manos. El objeto “Frame” es la raíz del modelo de datos de Leap Motion. Cada objeto “Frame” contiene las manos rastreadas, detallando sus propiedades en un momento dado en el tiempo. Se divide en tres clases: “Hand”, “Finger” y “Arm”, cada una de las cuales tendrá los datos de cada parte capturada por el dispositivo Leap Motion.

Las manos se representan por la clase “Hand”, que proporciona información sobre la identidad, la posición y otras características de una mano detectada, el brazo al que está sujeta la mano y una lista de los dedos asociados con la mano.

El software de Leap Motion utiliza un modelo interno de una mano humana para proporcionar un seguimiento predictivo incluso cuando no se ven partes de una mano. El modelo de mano siempre proporciona posiciones para cinco dedos, aunque el seguimiento es óptimo cuando la silueta de una mano y todos sus dedos son claramente visibles. El software utiliza las partes visibles de la mano, su modelo interno y las observaciones anteriores para calcular las posiciones más probables de las partes que no están visibles actualmente.

Los brazos se representan con la clase “Arm” que proporcionará datos de orientación, longitud, anchura y puntos finales de un brazo. Cuando el codo no está a la vista, el controlador de Leap Motion calcula su posición basándose en observaciones pasadas,

así como en la proporción humana típica. Aunque es interesante comentar la clase "Arm", esta clase no la utilizaremos en el proyecto.

Por último, los dedos son representados por la clase "Finger" y aporta información del movimiento de cada dedo, la dirección a la que apunta, si se encuentra o no flexionado y longitud y anchura del mismo. El controlador Leap Motion proporciona información sobre cada dedo de una mano. Si la totalidad o parte de un dedo no es visible, las características de los dedos se estiman según las observaciones recientes y el modelo anatómico de la mano. Los dedos se identifican por su nombre, a saber, "thumb", "index", "middle", "ring" y "pinky". Cada dedo se encuentra dividido en tres falanges; "bone1", "bone2" y "bone3".

Además, el SDK de Leap Motion es capaz de identificar gestos realizados por la mano o por los dedos.

c. Funcionamiento

La razón principal por la que hemos elegido Leap Motion para la captura de movimiento es que es capaz de tomar datos relativos en un entorno en 3D y reproducir dichos datos. En este apartado trataremos el funcionamiento del dispositivo para obtener los datos.

El problema fundamental al que se enfrentan los dispositivos de captura de movimiento basados en imágenes es la obtención de parámetros de profundidad.

Como hemos comentado, los LEDs inundan la zona con luz infrarroja, la cual, al chocar con un objeto, produce una reflexión de la luz que es captada por los sensores ópticos, los cuales almacenan en una matriz y son transmitidos al ordenador, con un previo ajuste de la imagen para conseguir una resolución de 640 x 120px, es decir, 76.800 píxeles.

Por tanto, el dispositivo no trata los datos de la imagen, solo los captura y los envía al driver donde se procesan dichos datos, permitiendo así que el procesamiento sea rápido.

Estos datos, obtenidos respectivamente por cada uno de los dos sensores, son relativos a la cantidad de luz, cuantificada en 8 bits, de cada pixel de la imagen capturada, generando así una imagen en escala de grises.

Figura 44 - Imágenes capturadas por el dispositivo Leap Motion

Antes de pasar a explicar el funcionamiento del algoritmo que nos dará los datos de profundidad, es importante destacar el funcionamiento de las lentes con las que cuenta cada cámara y la distorsión que estas producen.

Dicha distorsión será distinta para cada punto de la imagen, es por ello que Leap Motion cuenta con un mallado de puntos que se superponen a la imagen captada, como podemos observar en la Figura 45.

Figura 45 - Mallado del dispositivo de Leap Motion

A cada conjunto de datos referente a la iluminación de cada pixel le acompaña otro conjunto de datos referente a la distorsión producida en cada pixel, obtenida gracias al mallado. Al llegar ambos conjuntos de datos al driver, se modifica la imagen para reconstruirla teniendo en cuenta la distorsión.

Por último, para posicionar las manos en el entorno 3D de coordenadas cartesianas de Leap Motion, se utilizan técnicas de visión estereoscópica.

Como hemos comentado, Leap Motion cuenta con dos cámaras con una distancia entre las cámaras (b), lo que nos permite obtener dos imágenes muy parecidas, pero con una pequeña disparidad (d) entre ellas. Teniendo en cuenta que la distancia focal (f) es la misma en ambas cámaras, podemos obtener la profundidad a la que se encuentra el objeto por la disparidad que presentan las imágenes de la siguiente manera:

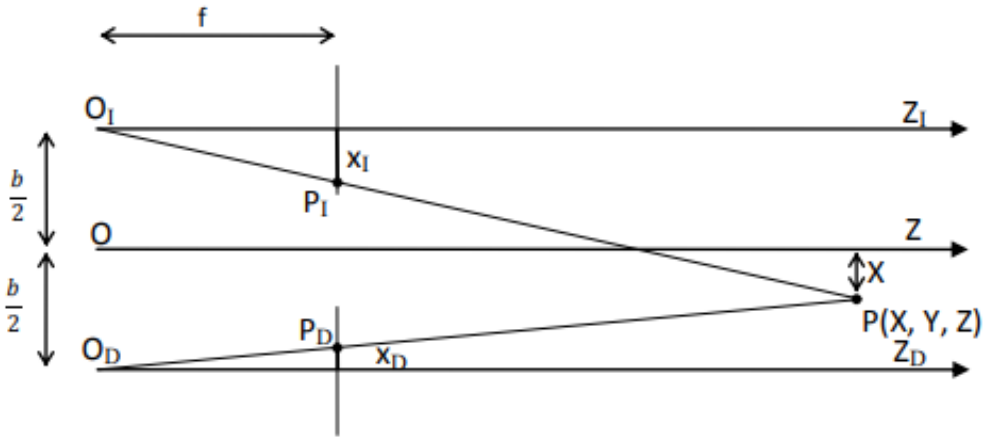

Figura 46 - Esquema de visión estereoscópica

Siendo Oi y Od la posición de las cámaras izquierda y derecha y el punto P, con coordenadas (x, y, z) en el sistema cartesiano en 3D, el punto del que queremos obtener su profundidad (z), obtendremos los puntos Pi y Pd como las proyecciones del punto P en la imagen obtenida por la cámara izquierda y la obtenida por la cámara derecha respectivamente, medimos las coordenadas del punto Pi (xi, yi) y del punto Pd (xd, yd), ambas en el sistema de coordenadas en 2D relativo de cada cámara.

La disparidad de las imágenes se calculará con la siguiente ecuación:

$$d = x_i - x_d$$

Pudiendo de esta forma hallar la profundidad (z) del punto P teniendo en cuenta los valores anteriormente comentados de distancia focal (f) y de distancia entre las cámaras (b). Para ello, haremos uso de la siguiente fórmula:

$$z = \frac{f * b}{d} = \frac{f * b}{x_i - x_d}$$

Es un método similar a la triangulación y es la forma con la que nuestro cerebro es capaz de percibir la profundidad de los objetos.

III. Sensor Flex

Resistencia plana	25KΩ
Tolerancia	±30%
Rango de resistencia	25KΩ – 150kΩ
Potencia	0.5W
Potencia de pico	1W
Longitud	55.37mm
Anchura	6.35mm
Rango de temperatura	-35°C to +80°C

Tabla 5 - Características técnicas del Sensor Flex

IV. Circuito Integrado LM324

Su función es la de aportar un mayor margen dinámico en la toma de medidas realizada por el sensor Flex con tal de poder discretizar los valores lo máximo posible a la hora de controlar la tensión del vibrador [22].

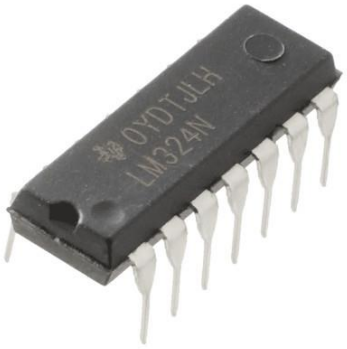

Figura 47 - Circuito integrado LM324

El LM324 consiste en un circuito integrado con 4 Amplificadores Operacionales, un pin de alimentación a 5V y un pin de tierra. En la Figura 48 podemos ver las conexiones del circuito integrado.

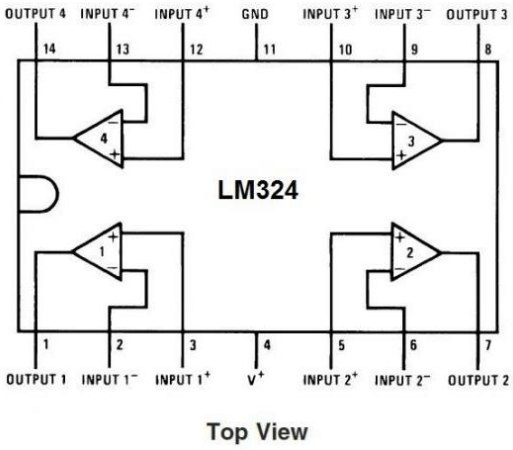

Figura 48 - Conexiones del circuito integrado LM324

V. Zumbador Pinzhi bi00149-es

Voltaje	3V
Diámetro	10mm
Longitud	3.4mm
Peso	1.2g
Rango dinámico	2.5V - 3.8V
Velocidad nominal	12000rpm

Corriente nominal	75mA
Voltaje inicial	2.3V
Corriente inicial	85mA
Resistencia	75Ω
Amplitud de vibración	0.8G

Tabla 6 - Características técnicas de Zumbador Pinzhi bi00149-es

VI. Circuito Integrado AD8656

El Circuito Integrado AD8656 se encargará de amplificar la corriente aportada por los pines analógicos de la placa de Arduino UNO con el fin de lograr los 85mA necesarios para los vibradores [23].

Figura 49 - Circuito integrado AD8656

Consiste en 2 amplificadores de corriente CMOS de bajo coste, un pin de alimentación a 5V y un pin de tierra. En la Figura 50 podemos ver las conexiones del circuito integrado.

Figura 50 - Conexiones del circuito integrado AD8656

VII. Código Arduino

```
const int FLEX_PIN1 = A1; //Pin
const int FLEX_PIN2 = A2;
const int FLEX_PIN3 = A3;
const int FLEX_PIN4 = A4;
const int FLEX_PIN5 = A5;
const int VIBRADOR1 = 5;
const int VIBRADOR2 = 6;
const int VIBRADOR3 = 9;
const int VIBRADOR4 = 10;
const int VIBRADOR5 = 11;
//Calibrar
const float BEND_RESISTANCE = 300.0;//Flex flexionado
const float STRAIGHT_RESISTANCE = 650.0;
//
#include <SoftwareSerial.h>
int valor;
boolean dedo1 = false;
boolean dedo2 = false;
boolean dedo3 = false;
boolean dedo4 = false;
boolean dedo5 = false;
void setup()
{
    Serial.begin(9600);
    pinMode(FLEX_PIN1, INPUT);
    pinMode (VIBRADOR1, OUTPUT);
    pinMode(FLEX_PIN2, INPUT);
    pinMode (VIBRADOR2, OUTPUT);
    pinMode(FLEX_PIN3, INPUT);
    pinMode (VIBRADOR3, OUTPUT);
    pinMode(FLEX_PIN4, INPUT);
    pinMode (VIBRADOR4, OUTPUT);
```

```
    pinMode(FLEX_PIN5, INPUT);
    pinMode (VIBRADOR5, OUTPUT);
    Serial.println("Start");
}
void loop()
{
    String aux = Serial.readString();
    //int value_Serial = Serial.read();
    int value_Serial = aux.toInt();
    //Serial.println (value_Serial);
    if (value_Serial != 0){
     if (value_Serial == 50){
      analogWrite(VIBRADOR1, 0);
      dedo1 = false;
      analogWrite(VIBRADOR2, 0);
      dedo2 = false;
      analogWrite(VIBRADOR3, 0);
      dedo3 = false;
      analogWrite(VIBRADOR4, 0);
      dedo4 = false;
      analogWrite(VIBRADOR5, 0);
      dedo5 = false;
     }
     else{
      valor = value_Serial & 1;
      if (valor == 1){
       dedo1 = true;
      }
      else {
       dedo1 = false;
      }
      ////
      valor = value_Serial & 2;
```

```
    if (valor == 2){
      dedo2 = true;
    }
    else {
      dedo2 = false;
    }
    ////
    valor = value_Serial & 4;
    if (valor == 4){
      dedo3 = true;
    }
    else {
      dedo3 = false;
    }
    ////
    valor = value_Serial & 8;
    if (valor == 8){
      dedo4 = true;
    }
    else {
      dedo4 = false;
    }
    ////
    valor = value_Serial & 16;
    if (valor == 16){
      dedo5 = true;
    }
    else {
      dedo5 = false;
    }
  }
}
if (dedo1 == true){
```

```
      Serial.println("1 vibrador");
      int flex1 = analogRead(FLEX_PIN1);
      int flex1_map = map (flex1, STRAIGHT_RESISTANCE, BEND_RESISTANCE, 110,
255);
      Serial.println(flex1_map);
      analogWrite(VIBRADOR1, flex1_map);
    }
    else {
      analogWrite(VIBRADOR1, 0);
    }
    ////
    if (dedo2 == true){
      Serial.println("2 vibrador");
      int flex1 = analogRead(FLEX_PIN2);
      int flex1_map = map (flex1, STRAIGHT_RESISTANCE, BEND_RESISTANCE, 110,
255);
      Serial.println(flex1_map);
      analogWrite(VIBRADOR2, flex1_map);
    }
    else {
      analogWrite(VIBRADOR2, 0);
    }
    ////
    if (dedo3 == true){
      Serial.println("3 vibrador");
      int flex1 = analogRead(FLEX_PIN3);
      int flex1_map = map (flex1, STRAIGHT_RESISTANCE, BEND_RESISTANCE, 110,
255);
      Serial.println(flex1_map);
      analogWrite(VIBRADOR3, flex1_map);
    }
    else {
      analogWrite(VIBRADOR3, 0);
    }
```

```
      ////
      if (dedo4 == true){
       Serial.println("4 vibrador");
       int flex1 = analogRead(FLEX_PIN4);
       int flex1_map = map (flex1, STRAIGHT_RESISTANCE, BEND_RESISTANCE, 110,
255);
       Serial.println(flex1_map);
       analogWrite(VIBRADOR4, flex1_map);
      }
      else {
       analogWrite(VIBRADOR4, 0);
      }
      ////
      if (dedo5 == true){
       Serial.println("5 vibrador");
       int flex1 = analogRead(FLEX_PIN5);
       int flex1_map = map (flex1, STRAIGHT_RESISTANCE, BEND_RESISTANCE, 110,
255);
       Serial.println(flex1_map);
       analogWrite(VIBRADOR5, flex1_map);
      }
      else {
       analogWrite(VIBRADOR5, 0);
      }
}
```

VIII. Código Máscara

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO.Ports;
using System;

public class Mascara : MonoBehaviour
{
```

```
    SerialPort sp = new SerialPort("COM5", 9600, Parity.None, 8, StopBits.One);

    public int valor;

    private int ristra;

    public void Start()
    {
        sp.ReadTimeout = 100;
    }
    public int getRistra()
    {
        return ristra;
    }
    public void setRistra(int ristra)
    {
        sp.Open();
        this.ristra = ristra;
        if (this.ristra == 0)
        {
            sp.Write(50.ToString());
            Debug.Log(50.ToString());
        }
        else
        {
            Debug.Log(this.ristra.ToString());
            sp.Write(this.ristra.ToString());
        }
        sp.Close();
    }
}
```

IX. Código Valor_dedo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Valor_dedo : MonoBehaviour
{

    public int valor;
    public GameObject Jugador;
    int ristra;
    void Start()
    {
```

```
    }

    public void OnCollisionEnter(Collision collision)
    {
        ristra = Jugador.GetComponent<Mascara>().getRistra();
        Debug.Log("colisionando on" + valor);
        Jugador.GetComponent<Mascara>().setRistra(ristra | valor);
    }
    public void OnCollisionExit(Collision collision)
    {
        ristra = Jugador.GetComponent<Mascara>().getRistra();
        Debug.Log("colisionando off" + valor);
        Jugador.GetComponent<Mascara>().setRistra(ristra & ~valor);
    }
}
```