UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Diseño y desarrollo de aplicaciones de control de robots mediante el dispositivos háptico de seis grados de libertad 3D SYSTEMS TOUCH

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

AUTOR/A: Poquet Sánchez, Mario

Tutor/a: Valera Fernández, Ángel

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Diseño y desarrollo de aplicaciones de control de robots mediante el dispositivo háptico de seis grados de libertad 3D SYSTEMS TOUCH

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

AUTOR/A: Poquet Sánchez, Mario

Tutor/a: Valera Fernández, Ángel

CURSO ACADÉMICO: 2021-2022

AGRADECIMIENTOS

Con este trabajo de fin de grado pongo fin a esta primera etapa de mi vida universitaria en la Universitat Politècnica de València. Han sido unos años intensos en los que he crecido como profesional y como persona.

En primer lugar, quiero agradecer a mi tutor Ángel Valera Fernández y a todos los compañeros del laboratorio de robótica por la ayuda y medios que me han brindado para poder hacer real este trabajo.

Del mismo modo, quiero agradecer a mis amigos y compañeros del grupo cinco por haber estado siempre presentes y haber logrado hacer de esta experiencia una de las más enriquecedoras de mi vida.

Cabe hacer una mención especial a mi familia, por haberme dado la libertad siempre de hacer lo que considere correcto para mi persona y por el apoyo constante recibido.

Por último, dar las gracias a cada uno de mis amigos de la UPB, por ser mucho más que solo amigos. Vixca la UPB!

RESUMEN

El presente proyecto consiste en la implementación de un dispositivo háptico de seis grados de libertad en un entorno industrial. De esta manera, se busca poder monitorizar un robot utilizando el dispositivo háptico 3D Systems Touch.

La aparición de los dispositivos hápticos ha supuesto un importante avance tanto en el diseño de mecanismos de simulación virtual como de teleoperación. No obstante, los hápticos se han ido introduciendo cada vez más en entornos como la medicina, artes gráficas o entretenimiento, pero menos en la industria.

Así pues, el objetivo de este trabajo consiste en llevar a cabo el diseño y desarrollo de aplicaciones con las que poder controlar un robot industrial mediante el uso del dispositivo háptico 3D Systems Touch.

Para ello, se realizará un profundo análisis del funcionamiento del dispositivo háptico, incluyendo tanto el software utilizado como un estudio de su cinemática directa y inversa, para posteriormente establecer una conexión entre este y un robot industrial (en el entorno virtual de RobotStudio) con el fin de lograr la teleoperación del mismo proporcionando una retroalimentación de fuerzas al teleoperador.

Finalmente, se abordarán posibles aplicaciones reales de nuestro caso de estudio.

PALABRAS CLAVE

Dispositivo háptico, robot, cobot, interacción, teleoperación, control, efector.

ABSTRACT

This project consists of the implementation of a haptic device with six degrees of freedom in an industrial environment. In this way, the aim is to be able to monitor a robot using the 3D Systems Touch haptic device.

The appearance of haptic devices has meant an important advance in the design of virtual simulation and teleoperation mechanisms. However, haptics have been increasingly introduced in environments such as medicine, graphic arts or entertainment, but less so in industry.

Therefore, the aim of this work is to carry out the design and development of applications with which to control an industrial robot using the 3D Systems Touch haptic device.

To this end, an in-depth analysis of the operation of the haptic device will be carried out, including both the software used and a study of its direct and inverse kinematics, in order to subsequently establish a connection between it and an industrial robot (in the virtual environment of RobotStudio) with the aim of achieving its teleoperation by providing force feedback to the teleoperator.

Finally, possible real applications of our case study will be discussed.

KEYWORDS

Haptic device, robot, cobot, interaction, teleoperation, control, effector.

RESUM

Aquest projecte consisteix a implementar un dispositiu hàptic de sis graus de llibertat en un entorn industrial. D'aquesta manera, es busca poder monitoritzar un robot utilitzant el dispositiu hàptic 3D Systems Touch.

L'aparició dels dispositius hàptics ha suposat un important avanç tant en el disseny de mecanismes de simulació virtual com de teleoperació. No obstant això, els hàptics s'han anat introduint cada cop més en entorns com la medicina, arts gràfiques o entreteniment, però menys a la indústria.

Així doncs, l'objectiu d'aquest treball consisteix a dur a terme el disseny i el desenvolupament d'aplicacions per poder controlar un robot industrial mitjançant l'ús del dispositiu hàptic 3D Systems Touch.

Per això, es realitzarà una profunda anàlisi del funcionament del dispositiu hàptic, incloent-hi tant el programari utilitzat com un estudi de la seva cinemàtica directa i inversa, per posteriorment establir una connexió entre aquest i un robot industrial (a l'entorn virtual de RobotStudio) amb la fi d'aconseguir la teleoperació del mateix proporcionant una retroalimentació de forces al teleoperador.

Finalment, s'abordaran les possibles aplicacions reals del nostre cas d'estudi.

PARAULES CLAU

Dispositiu hàptic, robot, cobot, interacció, teleoperació, control, efector.

ÍNDICE

DOCUMENTOS CONTENIDOS EN EL TFG

1. Memoria
2. Presupuesto

ÍNDICE DE LA MEMORIA

ÍNDICE DEL PRESUPUESTO

ÍNDICE DE FIGURAS

MEMORIA

ÍNDICE DE LA MEMORIA

Diseño y desarrollo de aplicaciones de control de robots mediante el dispositivo háptico de seis grados de libertad 3D Systems Touch

1. INTRODUCCIÓN Y OBJETIVOS

1.1 Introducción

El trabajo de final de grado desarrollado forma parte del plan de estudios del Grado en Ingeniería en Tecnologías Industriales impartido por la Universidad Politécnica de Valencia. Para poder llevar acabo este proyecto se han utilizados medios proporcionados por el departamento de Ingeniería de Sistemas y Automática.

El trabajo que se plantea tiene como finalidad el desarrollo de un mecanismo de conexión y control mediante el cual el usuario pueda teleoperar manualmente un robot industrial, utilizando para ello el dispositivo háptico 3D Systems Touch que le permite recibir información en forma táctil, para finalmente poder llevar a cabo diferentes aplicaciones industriales.

Como bien trataremos más adelante, la importancia del uso de dispositivos hápticos en entornos industriales radica en que mejoran la experiencia del usuario en muchas aplicaciones, como por ejemplo las de telemanipulación. Cuando se utilizan para dirigir un robot, como es el caso de este proyecto, los dispositivos hápticos permiten al operador realizar con más velocidad y precisión tareas que son remotas en espacio y/o escala ya que permiten controlar con precisión la posición del efector final del robot, además de aportar información sensorial relacionada con el sentido del tacto. Es decir, el operador es capaz de sentir fuerzas generadas por el dispositivo háptico que restringen ciertos movimientos. Como vemos, este elemento otorga al usuario una mayor inmersión, lo cual juega a su favor a la hora de tomar decisiones relacionadas con la actividad desempeñada.

Finalmente se abordará la explicación de la célula desarrollada mediante el software de simulación y programación offline de ABB, RobotStudio, y el 3D Systems Touch, así como las diferentes aplicaciones que se pueden llevar a cabo gracias a este trabajo.

1.2 Objetivos

A continuación, se van a concretar los objetivos fijados para la realización de este trabajo fin de grado.

- Búsqueda de información acerca de las características, funcionamiento y aplicaciones de los dispositivos hápticos y robots industriales, así como colaborativos, con el fin de constituir el estado del arte.
- Puesta a punto del dispositivo háptico 3D Systems Touch y análisis de la interfaz de programación OpenHaptics.
- Estudio de la comunicación mediante paquetes de información *sockets,* con el fin de establecer una comunicación entre el dispositivo háptico y el robot ABB IRB 140.
- Desarrollo de una célula robótica por medio de RobotStudio con la que diseñar una aplicación de control remoto de un robot industrial mediante el dispositivo háptico 3D Systems Touch.
- Diseño de un modelo de fuerzas coherente con la anterior aplicación para lograr disponer de una aplicación de control remoto de un robot industrial segura.
- Aplicaciones industriales derivadas de la célula diseñada.

2. ESTADO DEL ARTE

2.1 Dispositivos hápticos

A la hora de interactuar con un dispositivo, las personas pueden utilizar diferentes canales sensoriales: visual, auditivo y háptico, siendo este último de interés para el desarrollo del proyecto.

La háptica es la ciencia que incorpora el sentido del tacto y el control a las aplicaciones informáticas mediante la fuerza (cinestésica) o respuesta táctil. Por tanto, llamamos dispositivos hápticos a aquellos dispositivos de entrada/salida que permiten al usuario tocar, sentir o manipular objetos simulados en entornos virtuales tridimensionales y sistemas teleoperados, es decir, son dispositivos que proporcionan una realimentación de fuerza (producción mecánica de información sensorial por el sistema kinésico) directamente al usuario que interactúa con entornos virtuales o remotos. Tales dispositivos trasladan una sensación de presencia al operador.

Algunas de las ventajas de la utilización de los dispositivos hápticos son la manipulación de los entornos virtuales de forma natural y en 3D, no con proyecciones 2D, pudiendo establecer entre el usuario y el entorno virtual una transferencia bidireccional y en tiempo real de información mediante el empleo de interfaces de tipo háptico (interacción multisensorial). De esta manera conseguimos mejorar la inmersión y plausibilidad, lo cual facilita la realización de distintas tareas y hace que sean idóneos para trabajar en entornos de realidad virtual.

También mejoran la experiencia del usuario en aplicaciones de telemanipulación cuando se utilizan para dirigir un sistema de movimiento robótico o sistemas cooperativos, ya que asisten activamente al operador para aumentar la velocidad y precisión de las tareas que son remotas en espacio y/o escala. Esto se consigue debido a que permiten controlar con precisión la posición del efector final del robot (el extremo del brazo robótico que sostiene la herramienta). Además, el perfil de control del robot puede incluir información de límites 3D para evitar el movimiento en áreas confinadas que pueden causar lesiones, lo que hace que el uso de controles hápticos sea imperativo para aplicaciones de cirugía robótica, operaciones submarinas o aplicaciones de manipulación de piezas teleoperadas y/o cooperativas. Otra ventaja que presentan es que se pueden integrar perfectamente con la mayoría de los dispositivos informáticos actuales.

Por otra parte, algunas de las desventajas de su utilización (aunque en el futuro puede que cambie su situación) son su alto coste, su complejidad a la hora de diseñar o fabricar estos dispositivos, la ausencia de estándares, reglas, leyes que permitan un desarrollo eficaz de estos dispositivos y la poca familiarización de la gran mayoría de usuarios con estos dispositivos.

2.1.1 Tipos de dispositivos hápticos y clasificación

Un sistema de control en lazo cerrado o realimentado (activo) es aquel que busca reducir el error y llevar la salida del sistema a un valor deseado manteniendo una relación determinada entre la salida y la entrada de referencia. Mientras que en los sistemas de control en lazo abierto la salida no tiene efecto sobre la acción de control ni se compara con la entrada de referencia, en un sistema en lazo cerrado se utiliza la diferencia entre la salida y la referencia (señal de error) como medio de control.

En un sistema háptico realimentado, la entrada es generada por la acción del usuario, mientras que la salida es proporcionada por el sistema y es la encargada de originar el estímulo háptico en el primero.

Una de las clasificaciones tradicionales propuestas para los sistemas hápticos activos se basa en el tipo de realimentación que va a ser proporcionada al usuario y que genera estímulos de distinta naturaleza. De esta manera, los dispositivos hápticos pueden dividirse en:

- Sistemas de realimentación táctil. Son aquellos referidos a la sensación cutánea y llevan a cabo acciones que permiten dar a conocer información acerca del contorno, la geometría y otras características de la superficie del objeto que se está tocando, como la rugosidad o temperatura.
- Sistemas de realimentación por fuerza. Son aquellos referidos a la sensación cinestésica. Estos dispositivos producen estímulos y fuerzas mecánicas que informan acerca de la dureza y el peso del objeto con el que se está interaccionando.

No obstante, los dispositivos hápticos activos pueden presentar simultáneamente ambos tipos de realimentación, lo que supone intensificar la experiencia háptica del usuario.

Otra clasificación de los dispositivos hápticos es la que se da desde el punto de vista de la arquitectura de control en lazo cerrado. Existen dos tipos de dispositivos hápticos activos dependiendo del papel desempeñado por los actuadores y sensores (entradas y salidas):

- Dispositivos hápticos activos por control de impedancia. Son aquellos en los que los sensores miden la posición de los movimientos realizados por el usuario y los actuadores funcionan como una fuente de fuerza que le es aplicada de vuelta (ver Figura 1.2). Sus entradas son desplazamientos y sus salidas fuerzas.

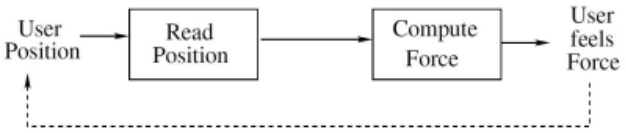

Figura 1. Esquema de control de impedancia

- Dispositivos hápticos activos por control de admitancia. Son aquellos en los que los sensores miden la fuerza aplicada por el usuario y los actuadores funcionan como una fuente de posición, sintiendo el usuario un movimiento en respuesta (ver Figura 1.3). Sus entradas son fuerzas y sus salidas desplazamientos.

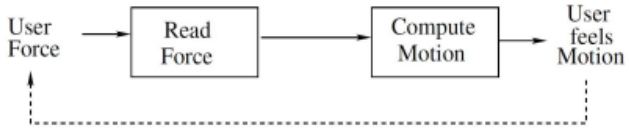

Figura 2. Esquema de control de admitancia

Los de impedancia han sido los empleados en este estudio y además son los más comunes, por lo que nos centraremos principalmente en los de esta clase. Los eventos que llevan a cabo en su flujo de control son:

1. Los sensores de desplazamiento recogen el movimiento.

2. Envío mediante driver de este desplazamiento al simulador software.

3. Interacción con el mundo virtual y cálculo de fuerzas.

4. Se envía señal al driver.

5. El dispositivo produce la fuerza necesaria.

De manera general se utiliza un único punto de interacción que se conoce como "punto de interfaz háptico" o HIP por sus siglas en inglés. Dado que la concentración en un único punto del sentido del tacto resulta contraintuitivo, los interfaces hápticos acostumbran a presentar algún tipo de mecanismo que mejore la ergonomía. Pueden presentarse soluciones que se asemejen a bolígrafos, como es el caso de los dispositivos desarrollados por la compañía 3D Systems.

En la mayor parte de las aplicaciones son utilizados en conjunto con mecanismos de audio y vídeo, de tal manera que pueda complementarse la experiencia envolvente que proporcionan estos dispositivos

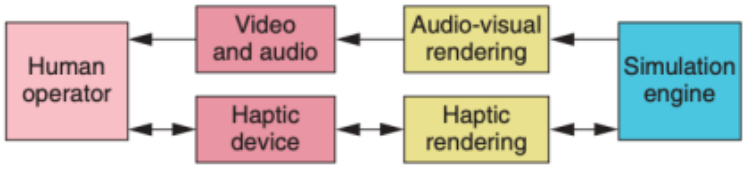

Figura 3. Arquitectura genérica de la aplicación de un dispositivo háptico controlado por impedancia

Otros tipos de clasificación de los dispositivos hápticos pueden ser las siguientes:

a) Conforme a los grados de libertad: 2DOF, 4DOF, 6DOF, etc.

b) Según el tipo de efector: tipo guante, tipo lápiz (fue el usado en este estudio).

c) De acuerdo con su ámbito de aplicación: específicos (medicina, por ejemplo), para propósito general.

2.1.2 Modelos de contacto

En el proceso de diseño de un modelo, se deberá atender a las características y necesidades prefijadas a las que se busque atender. Si bien en algunas aplicaciones la capacidad del dispositivo para renderizar contornos de escenarios o piezas será suficiente, para otras, la posibilidad de generar superficies rugosas o elásticas, por ejemplo, en una simulación dirigida a la formación en medicina, puede mejorar la experiencia formativa del usuario.

En el desarrollo de estos modelos, los programadores se enfrentan a una serie de desafíos. En primer lugar, los interfaces hápticos cuentan con un área de trabajo nominal, en la que el fabricante garantiza un funcionamiento equitativo y preciso en todas direcciones, y un área de trabajo real, determinada por las limitaciones físicas del dispositivo y donde pueden producirse comportamientos indeseados. Así pues, nos encontramos con las siguientes áreas de trabajo:

- Espacio de trabajo nominal. Es el volumen en el cual el fabricante del dispositivo garantiza la precisión y realimentación de fuerza especificada en la hoja de características.

- Espacio de trabajo real. Se trata del volumen que es posible recorrer con el extremo del manipulador, incluyendo una zona marginal donde el comportamiento del dispositivo puede ser inaceptable para algunas aplicaciones además de inestable.
- Espacio de trabajo de la aplicación. Es el volumen de la aplicación que se está simulando, luego es diferente para cada sistema y se ha de implementar en el código de manera que se haga coincidir a los tres de la mejor forma posible

La parte clave para una simulación de un programa con dispositivos hápticos es, sin duda, una correcta implementación de las fuerzas que se debe ejercer para que el usuario pueda percibirlas con fluidez y realismo durante los movimientos. Deberá valorarse la conveniencia de realizar un programa capaz de presentar un funcionamiento correcto en la totalidad del área de trabajo real, o buscar una forma de restringir el movimiento del interfaz a la zona de trabajo nominal, a fin de evitar comportamientos inaceptables.

Figura 4. Área de trabajo nominal (izq) y real (dcha) de un dispositivo Phantom Omni

En segundo lugar, un dispositivo háptico ideal no debería generar oposición al desplazarse a través de un entorno carente de obstáculos, sin embargo, es inevitable que existan rozamientos internos, por lo que se podría utilizar el modelo para contrarrestarlos.

Finalmente, los procesos requeridos para la representación háptica requieren de muchos recursos en un sistema informático, por lo que durante la fase de diseño se debe considerar una compensación entre la complejidad del modelo de contacto y los recursos necesarios para la operación. El modelo óptimo debería tener una representación háptica en seis grados de libertad (fuerzas y momentos en las tres direcciones cartesianas) porque este es el modelo verdaderamente observable, pero ya sea para reducir la carga computacional o para buscar implementar un modelo más simple, a veces se utilizan solo tres grados de libertad, eliminando momentos torsores y reduciendo la interacción del usuario con el entorno virtual a un único punto, el mencionado HIP (haptic interface point).

Una de las filosofías que utilizan tres grados de libertad, es el llamado método proxy. Para obtener unas fuerzas que se parezcan a las que sucederían si tocásemos los objetos de forma real habrá que distinguir entre dos puntos: el HIP, representación del dispositivo en el mundo virtual, y el PROXY que es la representación del dispositivo virtual, el cual coincidirá con el HIP cuando no exista colisión. El proxy (también conocido como "SCP" u "objeto-dios") es un punto que sigue de cerca la posición del dispositivo háptico. La posición del proxy se limita al exterior de las superficies de todas las formas táctiles. El motor de renderizado háptico actualiza continuamente la posición del proxy, intentando moverlo para que coincida con la posición del dispositivo háptico, pero sin permitir que se mueva dentro de ninguna forma. Aunque la posición real del dispositivo háptico puede estar dentro de una forma, el proxy siempre estará fuera. Cuando no esté en contacto con una forma, el proxy siempre se

colocará en la posición del dispositivo, pero cuando esté en contacto con una forma, el dispositivo háptico penetrará en la superficie de la forma y el proxy permanecerá por fuera de la superficie.

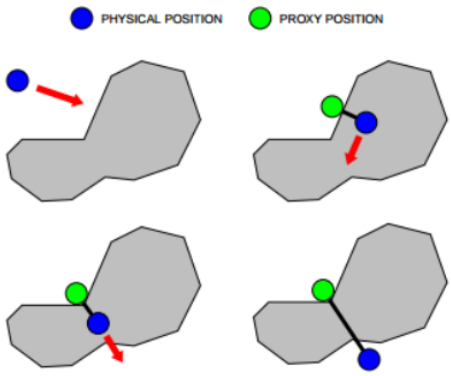

Figura 5. Representación PROXY y HIP

En caso de colisión, la interfaz háptica continúa representando la posición real del sistema físico, pero la posición del proxy representa la ubicación del entorno virtual, que no puede abarcar ningún elemento físico de la simulación. Una primera aproximación podría ser a partir de la ley de Hooke (usando un modelo lineal), de manera que si existe una diferencia en las posiciones de los dos se crea un sistema muelle-amortiguador entre ellos y, a medida que aumenta la distancia entre ellos, se produce una fuerza mayor en dirección y sentido normales a la superficie de contacto, evitando así la intención del usuario de transgredir los límites virtuales.

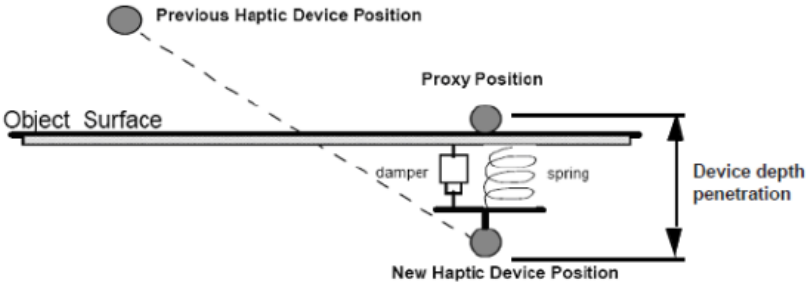

Figura 6. Situación del puntero durante un contacto

Las fuerzas generadas responden a la siguiente ecuación en cada una de las tres dimensiones:

$$F\vec{\imath} = (-K \cdot \Delta x)\vec{\imath} + \left(-b \cdot \frac{\partial \Delta x}{\partial t}\right)\vec{\imath}$$

En donde F es la fuerza en una dirección genérica $\vec{\imath}$ (X, Y o Z), K la constante elástica del muelle, ΔX la distancia en dirección $\vec{\imath}$ entre el punto de interfaz háptico y el punto proxy, tomando b como la constante viscosa del amortiguador. El valor máximo de la K viene dado por el límite en magnitud de representación de fuerza.

En condiciones normales, la distancia entre ambos puntos será aquella que resulte en la ejecución de una menor fuerza, sin embargo, si se produce un cambio de posición y el punto proxy ha de transitar por una región con mayor energía potencial en comparación con la que posee en ese momento debido a la existencia de un obstáculo, no podrá continuar avanzando en esa dirección.

Por otro lado, si lo que estás buscando es un modelo de seis grados de libertad, no puedes reducir la interacción a un solo punto porque no puedes aplicar un momento torsor a un elemento 1D, y aquí entra en juego el concepto de "god object". Este god object se refiere a un objeto que permite al usuario relacionarse con el medio, de manera que las restricciones de movimiento causadas por otros objetos o cualquier otro tipo de restricción no pueden exceder el contorno del objeto.

2.1.3 Aplicaciones y ejemplos comerciales

Dentro de los usos de los interfaces hápticos, podemos encontrar el diseño asistido por ordenador. El desarrollo de prototipos virtuales para la evaluación de productos se basa en técnicas de visualización que han logrado un alto nivel de reproducción de la información, buscando soluciones que satisfagan las necesidades del mercado a un menor precio. Por otro lado, la tecnología háptica no ha seguido el ritmo de los avances en el desarrollo de productos. Con respecto a las pruebas de prototipos virtuales, la tecnología se ha limitado a los aspectos de visualización. Sin embargo, la evaluación es más adecuada para valorar los aspectos estéticos del producto y no se aplica a los aspectos de la interacción con el usuario. Evaluar la usabilidad de un elemento requiere capturar la sensación táctil y cómo el usuario manipula físicamente el objeto mientras realiza la tarea. Para cumplir este requisito, se han desarrollado dispositivos de control háptico sencillos para probar la interacción del usuario con los productos.

Las mejoras y reducciones de costes en los sistemas informáticos asociados al desarrollo de software especializado (como Geomagic Sculpt o Geomagic Freeform) han permitido una mejor implementación de estos sistemas en el entorno profesional y de investigación. Los enfoques para usar estos dispositivos en el mundo del diseño incluyen tomar una pieza de material y dar forma a su superficie interactuando con puntos de interfaz háptico, presionando o tirando de superficies preexistentes. Su principal ventaja es que pueden generar formas muy complejas más rápido que otros programas CAD y siempre pueden trabajar con las tres dimensiones de un objeto.

Hay muchos ejemplos del uso de interfaces hápticos en el campo médico, utilizando el tacto como herramienta didáctica. Uno es la creación de un entorno simulado en el que se practica la extirpación de la glándula submandibular, como se explica en el ensayo []. Se utilizó la interfaz háptica Geomagic Touch, modificada para parecerse a los materiales quirúrgicos que se utilizarían en la operación real, diseñada específicamente para parecerse a unas tijeras. Como bien se detalla en el artículo, a medida que aumenta la práctica se reduce notablemente el número de errores cometidos, además de conseguirse disminuir el tiempo de operación respecto a un grupo que no tuvo preparación con el sistema. De esta manera, queda demostrado que la utilización de dispositivos hápticos para la formación de profesionales consigue resultados positivos de cara a operaciones con pacientes reales.

No obstante, la aplicación que sirvió como motivación para la realización de este trabajo tiene que ver con el control preciso de robots utilizando precisamente dispositivos hápticos. Los robots son capaces de realizar movimientos muy precisos, pero deben ser guiados con exactitud para aprovechar su potencial. Siguiendo la línea del anterior ejemplo, un cirujano humano, con todos sus conocimientos y experiencia, tiene que practicar dónde sondear, cortar o coser antes de poder desarrollar las habilidades necesarias para hacer una sutura limpia con el grado de tensión adecuado a la profundidad correcta o una incisión de la profundidad adecuada. En cambio, un brazo robótico puede moverse con

más consistencia y precisión que el del mejor cirujano humano. La clave es guiar el brazo del cirujano robótico con la experiencia humana y proporcionar al cirujano que controla el brazo robótico un control de movimiento de bucle cerrado para garantizar que el brazo robótico haga lo que se supone que debe hacer, y es aquí donde entra en juego la tecnología háptica, que es capaz de proporcionar información sensorial al cirujano humano y le ayuda a guiar el movimiento robótico.

Los dispositivos hápticos como el empleado en la realización de este trabajo pueden controlar con precisión la posición del efector final del robot (el extremo del brazo robótico que sostiene la herramienta). Además, la información de los límites en 3D puede incluirse en el perfil de control del robot para evitar el movimiento en zonas restringidas en las que podría causar daños, lo que hace que el uso de controles hápticos sea ideal para aplicaciones de cirugía robótica. La tecnología también permite amplificar o "escalar" las dimensiones entre el extremo robótico del sistema y el operador humano para que éste pueda realizar movimientos cómodos a escala humana mientras controla las operaciones robóticas a una escala mucho menor. Esta capacidad tiene poderosas implicaciones en el campo de la nanotecnología, incluida la nanocirugía, pero también puede extenderse a otras operaciones que deban manejarse a escala molecular o casi molecular. Pensad en la posibilidad de "sentir" realmente las células o las moléculas y manipularlas mediante actuadores robóticos precisos. Más allá de la cirugía robótica, la tecnología puede utilizarse para controlar robots que trabajen en zonas peligrosas (por ejemplo, en operaciones mineras), robots que manipulen materiales peligrosos (por ejemplo, residuos nucleares) o robots dedicados a la fabricación de productos a nanoescala, como micromáquinas para aplicaciones biomédicas.

A modo de resumen, se puede concluir que los ámbitos de aplicación de los dispositivos hápticos son variados y están en continua expansión. Entre ellos se encuentran:

- Simulación quirúrgica y formación médica, así como micro robots para cirugía mínimamente invasiva (MIS).

- Pintura, escultura y CAD.

- Aplicaciones militares, como el entrenamiento y la simulación aeroespacial y militar.

- Tecnología de asistencia para ciegos o discapacitados visuales.

- Videojuegos mediante realidad virtual, simuladores.

- Sistemas cooperativos y de telemanipulación mediante el control de un sistema robótico.

A continuación, se presentan algunos ejemplos de los dispositivos hápticos realimentados comerciales más relevantes, señalando algunas de sus características de funcionamiento:

- Novint Falcon: Se trata de un dispositivo de escritorio (desktop) con diseño en configuración delta desarrollado por la compañía Novint. Cuenta con conexión USB y fue concebido para reemplazar el ratón del ordenador en videojuegos y otras aplicaciones. El control se realiza mediante una empuñadura diseñada para ser extraíble y poder cambiarse según el uso que vaya a hacerse del sistema (ver Figura 1.4). Está provisto de realimentación de fuerza gracias a los tres motores que posee, cuyo movimiento es monitorizado por un encoder en cuadratura con 320 pulsos por revolución. Introduce 3 grados de libertad y su interfaz con el ordenador trabaja a una frecuencia de muestreo de entre 800 y 1000Hz.

Figura 7. Novint Falcon

- Guantes con feedback de fuerza CYBERGRASP Immersion Co: El Cybergrasp consiste en una estructura exoesquelética fijada a la parte posterior de la mano, que es accionada por unos actuadores instalados fuera de ésta, en una caja de control, con el objetivo de facilitar su manejo aligerando su peso, de aproximadamente 450 gr. La fuerza máxima que puede aplicar sobre cada dedo es de 12N.

Figura 8. Guantes Cybergrasp

- Sigma.7: Se trata del interfaz háptico más avanzado que ha desarrollado la empresa Force Dimension (Suiza). Introduce 7 grados de libertad activos en una configuración delta rediseñada, incluyendo un mecanismo para permitir una sujeción de alta precisión. Proporciona realimentación de fuerza y torsión al usuario, compensando perfectamente el efecto de la gravedad. Fue concebido para soportar aplicaciones de la industria médica y aeroespacial. Su frecuencia de renderizado de las fuerzas alcanza los 8kHz.

Figura 9. Force Dimension – sigma.7

- 3D Systems Touch: Este dispositivo de escritorio (desktop) desarrollado por 3D Systems permite a los usuarios tocar y manipular objetos virtuales gracias a sus 6 grados de libertad que habilitan el libre movimiento de la mano y giro de la muñeca (ver Figura 1.7). Proporciona

una realimentación de fuerza en los tres ejes (x,y,z) directamente a la mano del usuario y está provisto sensores de posición que monitorizan a cada instante la posición de su lápiz. Más adelante se analizará más detalladamente.

Figura 10. 3D Systems Touch

2.2 Robots industriales y colaborativos

El reciente análisis de Frost & Sullivan concluye que el mercado mundial de la robótica industrial alcanzará unos ingresos de 38.300 millones de dólares en 2024, frente a los 22.200 millones de dólares de 2020, con una CAGR del 12,2%. Aunque la industria se vio frenada por la pandemia de COVID-19 y la incertidumbre en el negocio de la automoción, se espera que la creciente demanda de otros sectores de alto crecimiento la impulse en los próximos cinco años. Los productos farmacéuticos serán el segmento de más rápido crecimiento, con una CAGR del 17,2% de 2019 a 2024, alcanzando 3,33 mil millones de dólares al final del período previsto, seguido por los alimentos y bebidas (F&B) y la electricidad y la electrónica, que se expanden al 15,8% y al 15,1%, respectivamente.

Asia-Pacífico sigue dominando el mercado mundial de la robótica industrial, y se estima que los ingresos superarán los 25.080 millones de dólares en 2024, con China, Japón y Corea del Sur como motores del progreso. La región europea es la segunda más importante, impulsada por la industria de la automoción y Alemania, el quinto país más grande del mundo en robótica industrial. La tendencia actual de Norteamérica a la automatización de la producción y a mantener todas las operaciones de fabricación en la empresa la sitúa en la tercera posición, con unos ingresos previstos de 6.190 millones de dólares en 2024.

2.3.1 Robots industriales

Un robot industrial es aquel robot que ha sido desarrollado para automatizar tareas de producción intensivas como las que requiere una línea de montaje en constante movimiento. Al tratarse de robots grandes y pesados, se colocan en posiciones fijas dentro de una planta industrial y, en torno a ellos, giran el resto de tareas y procesos de los trabajadores. Las características de los robots industriales, variarán según los fabricantes, las necesidades y el escenario en el que se vaya a localizar.

La definición de un robot industrial según la norma internacional ISO 8373:2012 es 'un manipulador multifuncional, reprogramable y controlado automáticamente, programable en tres o más ejes que puede estar fijo en un área o móvil para su uso en aplicaciones de automatización industrial'.

Los robots industriales no suelen tener forma humanoide, aunque son capaces de reproducir movimientos y comportamientos humanos pero con la fuerza, precisión y rapidez de una máquina.

En la siguiente tabla extraída del informe World Robotics 2021 se aprecia la evolución y previsión de instalaciones, por año, de robots industriales.

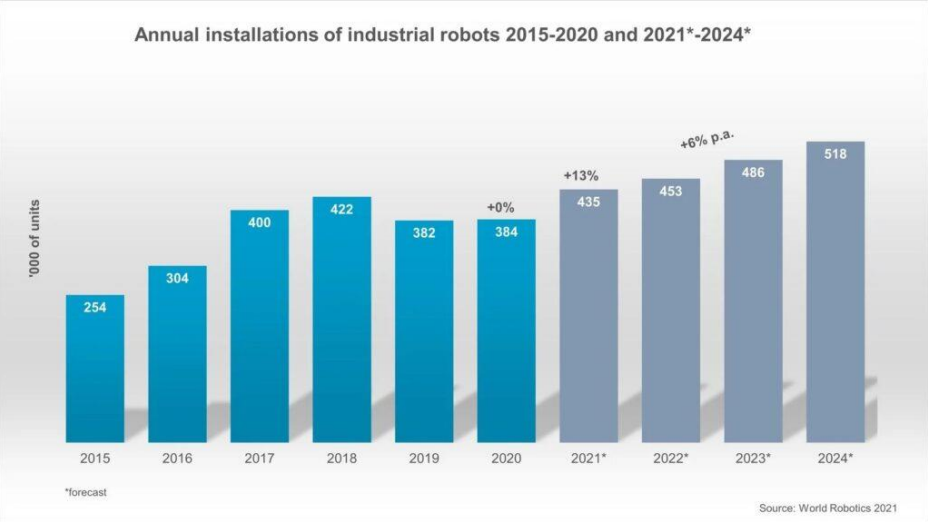

Figura 11. Evolución de la instalación anual de robots industriales

Actualmente, no solo las grandes empresas tienen posibilidad de acceso a los robots industriales. Cada vez más pymes están experimentando un incremento en la rentabilidad y una reducción en los costes de producción gracias a la automatización de ciertos procesos. Uno de los objetivos de la robótica industrial es optimizar las líneas de producción haciéndolas más ágiles y adaptables a las necesidades específicas de cada cliente.

Existe una gran variedad de aplicaciones de los robots industriales. Aunque hay muchas formas de aplicar la tecnología robótica, los objetivos de estas máquinas son similares en todas las industrias y procesos: mejorar la producción. Los robots son capaces de mejorar la eficiencia y la seguridad de muchas industrias al asumir trabajos aburridos, sucios y peligrosos.

2.3.2 Robots colaborativos o "cobots"

Los robots colaborativos son robots que pueden trabajar de forma segura junto a un humano y con él. Aunque ya hay algunas máquinas en la industria manufacturera que necesitan a los humanos para funcionar, los cobots interactúan con un humano y trabajan junto a él en el mismo paso de su producción.

Hablando un poco de sus orígenes, los robots colaborativos se inventaron en 1996 gracias a una iniciativa que comenzó dos años antes. En 1994, Prasad Akella, del Centro de Robótica de General Motors, y más tarde con una beca de investigación de la Fundación General Motors en 1995, quiso

encontrar una forma de que los robots y las personas trabajaran juntos de forma segura. El primer robot colaborativo fue inventado en 1996 por los profesores de la Universidad Northwestern J. Edward Colgate y Michael Peshkin y patentado un año después. El primer robot colaborativo no tenía una fuente de energía interna que pudiera moverse por sí mismo. Para que fuera seguro, era el humano el que tenía el control de su potencia. Por aquel entonces, no se llamaban robots colaborativos o cobots, sino "Dispositivo de Asistencia Inteligente". Colgate y Peshkin pasarían a crear su propia empresa, Cobotics, para producir los primeros cobots utilizados en la industria manufacturera. Se utilizaron en la línea de montaje final de una empresa automovilística. La empresa fue comprada por Stanley Assembly Technologies en 2003. Desde entonces, otras empresas han desarrollado diferentes tipos de robots colaborativos para satisfacer las demandas de diversas industrias.

En 2005 se da el siguiente paso en el desarrollo de esta tecnología con la aparición de la empresa Universal Robots. Esta compañía nace con el objetivo de crear un brazo robótico, capaz de operar de manera segura para los trabajadores, pero sin necesidad de que uno de ellos determine manualmente sus movimientos. Esta meta se alcanza tres años después con el lanzamiento del cobot UR5. Este sistema presenta limitaciones en cuanto a velocidad de movimientos y la fuerza a ejercer, además de contar con materiales ligeros y unos bordes más redondeados, con el objetivo de minimizar daños en caso de un accidente inesperado. Otra característica que los diferencia de sus predecesores es el entorno de programación, estando este desarrollado de manera que cualquier usuario con formación básica en programación sea capaz de reproducir sus instrucciones básicas.

A partir de la aparición del sistema creado por Universal Robots, diversas empresas comenzaron a diseñar sus propios sistemas, pero manteniendo los principios anteriormente mencionados del cobot UR5. Podemos poner como ejemplo los cobots “ABB YuMi“, “KUKA–LBR iiwa” o “Baxter” de la empresa “Rethink Robotics”, los cuales, mediante el uso de sensores, cámaras o la combinación de ambas, pueden reaccionar a un entorno cambiante y modificar sus respuestas en consecuencia.

Figura 12. Robot colaborativo UR5 (izq) y robot articulado YuMi IRB 14000 (dcha)

Por tanto, los cobots y los robots industriales tradicionales se diferencian en muchos aspectos. El robot industrial produce de forma masiva, ocupa mucho espacio y, a menudo, permanece en una posición fija. En cambio, el cobot es compacto y ocupa poco espacio. Además, se puede reubicar en cualquier

lugar sin ningún tipo de problemas. Otra diferencia es que los robots colaborativos están construidos para la seguridad, por lo que las posibilidades de que se produzcan riesgos en el lugar de trabajo al interactuar con los cobots son relativamente bajas. La Oficina de Estadísticas Laborales indica que el 23,5% de las lesiones no mortales provienen del contacto con el equipo.

Los robots colaborativos están construidos para reducir el riesgo de que esto ocurra en el lugar de trabajo, basándose en su construcción y manejo. Los robots colaborativos están fabricados con materiales ligeros y con bordes redondeados. Así se evitan lesiones accidentales al manipular estos robots en el lugar de trabajo. Para que sean más seguros, los cobots tienen un límite de velocidad y fuerza en el movimiento y cuentan con sensores para determinar las características clave de seguridad. Gracias a sus características de seguridad, el personal de la industria manufacturera puede operar directamente con los cobots en lugar de la antigua medida de seguridad que consistía en encerrar a los robots en jaulas o salas para que funcionaran y así garantizar que nadie se lesionara con su peso o sus rápidos movimientos.

En el informe de Frost & Sullivan, Natarajan añadió: "Los robots colaborativos (cobots) están experimentando un rápido crecimiento del mercado gracias a su utilidad, su facilidad de instalación y su precio en constante descenso, lo que los convierte en una solución asequible y viable para una amplia gama de aplicaciones. Será el segmento de más rápido crecimiento para 2024, registrando una CAGR del 32,8% (2019-2024) y alcanzando 1,78 millones de dólares en ingresos globales. Los avances en el 5G y la computación de borde serán fundamentales para equipar a los cobots con una flexibilidad mejorada y una implementación más fácil."

2.3.3 Aplicaciones de los cobots

La Federación Internacional de Robótica (IFR) divide los robots en dos categorías: robots para fabricación y robots para uso doméstico y profesional. La mayoría de los robots colaborativos pertenecen a la primera categoría. Al igual que otras máquinas utilizadas con fines industriales y de fabricación, los cobots tienen una variedad de aplicaciones basadas en factores como su adaptabilidad, tamaño, movilidad y características de seguridad.

Los humanos pueden realizar la mayoría de las tareas de una línea de producción, pero muchos robots actuales se inventaron para realizar ciertas tareas de forma más rápida y eficiente. Sin embargo, aunque los estudios sugieren que alrededor del 8,5% de los puestos de fabricación estarán automatizados en 2030, sólo hay algunas tareas que requieren tanto a los humanos como a los robots.

No obstante, hay muchas aplicaciones de robots colaborativos en todos los sectores. Entre ellas se encuentran el ensamblaje, la dispensación, el acabado, la atención a las máquinas, la manipulación de materiales, la soldadura, la eliminación de materiales, las inspecciones de calidad, etc. A continuación, resumimos tres clases comunes de despliegue de cobots.

Manipulación de materiales

La manipulación de materiales es uno de los trabajos más peligrosos en la fabricación. Materiales como el metal, el plástico y otras sustancias pueden suponer un gran riesgo para los trabajadores humanos. Además, muchas tareas de manipulación de materiales son repetitivas, lo que puede dar lugar a lesiones por esfuerzo repetitivo. Las industrias que utilizan robots en la fabricación pueden observar

cómo se reduce significativamente el numero de lesiones en el lugar de trabajo. Los materiales pesados pueden levantarse y transportarse fácilmente por los suelos de las fábricas utilizando plataformas robóticas móviles. Mientras tanto, las tareas de mantenimiento de máquinas, incluidas las que implican máquinas CNC pesadas, también están dentro de las capacidades de los cobots. Universal Robots.

Montaje y control de calidad

Los cobots están diseñados específicamente para trabajar junto a los empleados humanos y aliviarlos de las tediosas y difíciles tareas de montaje. Esto incluye la soldadura de piezas pequeñas, la perforación de tornillos y otras tareas de montaje similares. Los cobots también pueden utilizarse para ayudar a garantizar la calidad durante el proceso de producción. A diferencia de los humanos, los cobots realizan la misma tarea de la misma manera, cada vez, sin cansarse ni sufrir ninguna pérdida de rendimiento. Por ejemplo, los cobots pueden colocar una cámara en el mismo lugar para realizar tantas mediciones y posiciones en tantas piezas como sea necesario, todo ello sin necesidad de recalibración óptica. Las tareas de Pick & Place que utilizan los cobots incluyen la adición de piezas a los productos que se están ensamblando, el llenado de cajas de productos terminados y el apilamiento de palés.

Retirada de material

Otra de las tareas que también son cruciales para la producción y que pueden ser manejadas por cobots. Por ejemplo, la retirada de material por parte de robots es necesaria para cualquier proceso que implique el llenado de moldes. Estos pequeños robots pueden evaluar la pieza moldeada y encargarse de recortar cualquier exceso de metal o plástico sin dañar la pieza ni someter a los trabajadores a riesgo de lesiones. Por otra parte, los cobots equipados con herramientas de dispensación y hardware pueden utilizarse para añadir cola y otros adhesivos, mientras que los cobots equipados con un kit de lijado pueden utilizarse para pulir las piezas para obtener un acabado brillante y suave.

Para terminar, cabe destacar la importancia de la implantación de los robots colaborativos en la industria, donde llegan a suponer el 28% del total de los robots en el sector automotriz, según la Federación Internacional de Robótica (IFR por sus siglas en inglés).

Nissan, por ejemplo, tomó la decisión de comenzar a utilizar estos sistemas en una de sus fábricas en Japón junto a empleados humanos. Así, se consigue aprovechar el espacio permitiendo a los empleados compartir el entorno con las máquinas trabajando de forma simultánea, ahorrándose así tiempo respecto a una planificación secuencial de los mismos eventos.

Otra empresa, esta vez del sector agroalimenticio, llamada Atria Scandinavia, fabrica una serie de productos con características variadas, por lo que se debe ajustar frecuentemente la cadena de producción. Gracias a la existencia de cobots, que permiten no disponer de vallados de seguridad, los empleados pueden realizar estos cambios con mayor celeridad, revirtiendo de manera positiva en la producción.

2.3 Teleoperación Maestro-Esclavo

Los sistemas de teleoperación maestro-esclavo se caracterizan por disponer de dos unidades idénticas o con funciones completamente similares que están conectadas por un sistema de comunicación bilateral (two-way communication). Estas instrucciones de operación se pueden ejecutar a través de la Cinemática Directa o Cinemática Inversa (dependiendo de la funcionalidad que se requiera) del dispositivo esclavo.

La semejanza funcional completa significa que las características de movimiento del dispositivo esclavo son totalmente reconocidas por el dispositivo maestro y pueden ser controladas directamente por el usuario. En otras palabras, la funcionalidad del esclavo es una representación directa de la funcionalidad del dispositivo maestro, que incluye el movimiento y su relación con su espacio de trabajo.

La comunicación entre los dispositivos maestro y esclavo se puede realizar utilizando diferentes protocolos y procedimientos. Una de las características más importantes de este enfoque es que permite al usuario interactuar con el dispositivo maestro a varios kilómetros de distancia del dispositivo esclavo en un entorno de trabajo seguro, independientemente de las duras condiciones en las que opera el dispositivo esclavo.

2.4 Comunicación mediante sockets

Los sockets (también llamados conectores) son un mecanismo de comunicación entre procesos que permiten la comunicación bidireccional tanto entre procesos que se ejecutan en una misma máquina como entre procesos lanzados en diferentes máquinas. La comunicación entre procesos a través de sockets se basa en la filosofía Cliente-Servidor.

Existen diferentes tipos de sockets. Los dos mas empleados son:

- Socket de flujo (SOCK_STREAM): Define un servicio orientado a conexión confiable (sobre TCP por ejemplo). Los datos se envían sin errores o duplicación y se reciben en el mismo orden de como fueron enviados. El control de flujo está integrado para evitar overruns. No se imponen límites en el intercambio de datos, que se considera flujo de bytes. Ejemplo de aplicación que use sockets de flujo es el Programa de Transferencia de Ficheros (FTP).
- Socket de datagrama (SOCK_DGRAM): Define un servicio no orientado a conexión (sobre UDP por ejemplo). Los datagramas se envían como paquetes independientes. El servicio no proporciona garantías; los datos se pueden perder o duplicar y los datagramas pueden llegar fuera de orden. No realiza desensamblado y reensamblado de paquetes. Ejemplo de aplicación que utilice sockets de datagrama es el Sistema de Ficheros de Red.

Los sockets se crean dentro de un dominio de comunicación que indica el formato de las direcciones que podrán tomar los sockets y los protocolos que soportarán dichos sockets (familia de sockets o familia de direcciones). Aunque existen diferentes dominios de comunicación, nos centraremos en los sockets de Dominio de Internet (AF_INET), ya que son el tipo más común de sockets y permiten la comunicación entre procesos ejecutados en la misma máquina o en diferentes máquinas.

Pasamos a definir ahora la terminología siguiente:

- Una *dirección de socket* es la tripleta: {*protocolo, dirección-local, proceso-local*}

En la familia TCP/IP, por ejemplo: {*tcp, 193.44.234.3, 12345*}

- Una *conversación* es el enlace de comunicación entre dos procesos.
- Una *asociación* es la quíntupla que especifica completamente los dos procesos que comprende una conexión:

{*protocolo, dirección-local, proceso-local, dirección-externa, proceso-externo*}

En la familia TCP/IP, por ejemplo: {*tcp, 193.44.234.3, 1500, 193.44.234.5, 21*} podría ser una asociación válida.

Una *media-asociación* es: {*protocolo, dirección-local, proceso-local*} o {*protocolo, dirección-externa, proceso-externo*} que especifica cada mitad de una conexión.

- La media-asociación se denomina también *socket* o *dirección de transporte*. Esto es, un socket es un punto terminal para comunicación que puede nombrarse y direccionarse en una red.

A continuación se lista algunas llamadas de la interfaz socket básica:

- Inicializar un socket

FORMATO: int sockfd = socket(int familia, int tipo, int protocolo)

donde:

  - familia simboliza familia de direccionamiento . Puede tomar valores tales como AF_UNIX, AF_INET, AF_NS y AF_IUCV. Su propósito es especificar el método de direccionamiento utilizado por el socket.
  - tipo simboliza el tipo de interface de socket a usar. Puede tomar valores tales como SOCK_STREAM, SOCK_DGRAM, SOCK_RAW, y SOCK_SEQPACKET.
  - protocolo puede ser UDP, TCP, IP o ICMP.
  - sockfd es un entero (similar a un descriptor de fichero) que devuelve llamada socket.

- Ligar (registro) un socket a una dirección de puerto

FORMATO: int bind(int sockfd, struct sockaddr *localaddr, int addrlen)

donde:

  - sockfd es el mismo entero devuelto por la llamada socket.
  - localaddr es la dirección local que devuelve la llamada bind.

Nótese que después de la llamada bind, ya se tienen valores para los tres primeros parámetros dentro de la asociación 5-tupla: {protocolo, dirección-local, proceso-local, dirección-externa, proceso-externo}

- Indicar premura para recibir conexiones

FORMATO: int listen(int sockfd, int tamaño-cola)

donde:

- sockfd es el mismo entero que devuelve la llamada socket.
- tamaño-cola indica el número de conexiones pedida que puede introducir en la cola el sistema mientras el proceso local no haya emitido la llamada accept.

- Aceptar una conexión

FORMATO: int accept(int sockfd, struct sockaddr *dirección-externa, int addrlen)

donde:

- sockfd es el mismo entero que devuelve la llemada socket.
- dirección-externa es la dirección del proceso externo (cliente) que devuelve la llamada accept.

Nótese que esta llamada accept la emite un proceso servidor en vez de un proceso cliente. Si existe una petición de conexión esperando en la cola por este socket de conexión, accept toma la primera petición de la cola y crea otro socket con las mismas propiedades que sockfd; en caso contrario, accept bloqueará al proceso llamador hasta que llegue una petición de conexión.

- Petición de conexión al servidor

FORMATO: int connect(int sockfd, struct sockaddr *dirección-externa, int addrlen)

donde:

- sockfd es el mismo entero devuelto por la llamada socket.
- dirección-externa es la dirección del proceso externo (servidor) que devuelve la llamada connect.

Nótese que esta llamada la emite un proceso cliente mejor que un proceso servidor.

- Enviar y/o recibir datos

Las llamadas read(), readv(sockfd, char *buffer, int addrlen), recv(), readfrom(), send(sockfd, msg, len, flags), write() pueden usarse para recibir y enviar datos en una asociación de socket establecida (o conexión). Nótese que estas llamadas son similares a las llamadas del sistema de E/S de ficheros estándar read y write.

- Cerrar un socket

FORMATO: int close(int sockfd)

donde sockfd es el mismo entero que devuelve la llamada socket.

Figura 13. Secuencia de llamadas para una comunicación orientada a conexión

3. DESCRIPCIÓN DE LOS SISTEMAS UTILIZADOS

3.1 Interfaz háptico

3.1.1 Hardware

El dispositivo háptico utilizado en el desarrollo de este proyecto es el 3D Systems Touch de la compañía 3D Systems.

Figura 14. Sistema háptico 3D Systems Touch

Se trata de un interfaz háptico de 6 grados de libertad con control por impedancia, que utiliza el método proxy como modelo de contacto. Touch es un dispositivo motorizado que aplica retroalimentación de fuerza a la mano del usuario, lo que le permite sentir objetos virtuales y producir sensaciones táctiles reales a medida que el usuario manipula los objetos 3D en la pantalla. Como puede observarse en la imagen, este dispositivo cuenta con un efector en forma de lápiz o bolígrafo, el cual recibe el nombre de "stylus". Su manipulación por parte del usuario determinará la posición del HIP dentro del escenario en el que se utilice el interfaz háptico.

Figura 15. Localización del HIP en el stylus

A continuación se muestran sus especificaciones técnicas:

SPECIFICATIONS	TOUCH™
Workspace	~6.4 W x 4.8 H x 2.8 D in > 160 W x 120 H x 70 D mm
Range of motion	Hand movement pivoting at wrist
Nominal position resolution	> 450 dpi ~0.055 mm
Maximum exertable force and torque at nominal position (orthogonal arms)	0.75 lbf/3.3 N
Stiffness	x-axis > 7.3 lb/in (1.26 N/mm) y-axis > 13.4 lb/in (2.31 N/mm) z-axis > 5.9 lb/in (1.02 N/mm)
Force feedback (3 Degrees of Freedom)	x, y, z
Position sensing/input (6 Degrees of Freedom) [Stylus gimbal]	x, y, z (digital encoders) [Roll, pitch, yaw (± 5% linearity potentiometers)]

Figura 16. Especificaciones técnicas del dispositivo háptico 3D Systems Touch

Características:

- Diseño portátil y compacto
- Espacio de trabajo compacto con retroalimentación de fuerza
- Detección posicional de 6 grados de libertad y retroalimentación de fuerza de 3 grados de libertad
- Puerto Ethernet o USB integrado compatible con RJ45
- Cómodo palpador de goma moldeada con pintura texturizada
- Palpador extraíble
- Dos interruptores momentáneos integrados para el lápiz
- Tintero para el lápiz
- Diseño ergonómico
- Compatible con el kit de herramientas OpenHaptics con la micro API QuickHaptics
- Fabricado con componentes metálicos y plásticos moldeados por inyección

Los dispositivos hápticos de pueden medir de forma precisa la posición espacial 3D (a lo largo de los ejes X, Y y Z) y la orientación (giro, inclinación y dirección) del lápiz de mano.

La información relativa a la situación espacial del HIP es generada por los valores aportados por los encoders digitales situados en las articulaciones J1, J2 y J3 del sistema, mientras que la relativa a la orientación es dependiente de los valores arrojados tanto por dichos encoders como por los potenciómetros situados en J4, J5 y J6 (yaw, pitch y roll respectivamente).

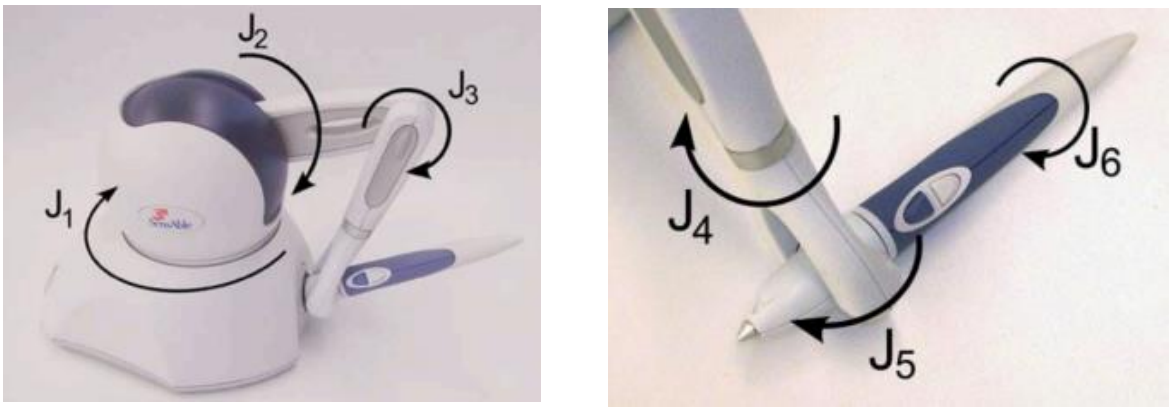

Figura 17. Articulaciones con encoders, posición (izq) y articulaciones cn potenciómetro, orientación (dcha)

A continuación, definimos el siguiente sistema de coordenadas, estando el origen de coordenadas a 153 mm de altura respecto la base, 95 mm en dirección positiva del eje Z respecto el borde inferior del inkwell (orificio en el que se guarda el stylus en la posición de reposo), y dentro del plano de simetría de la base.

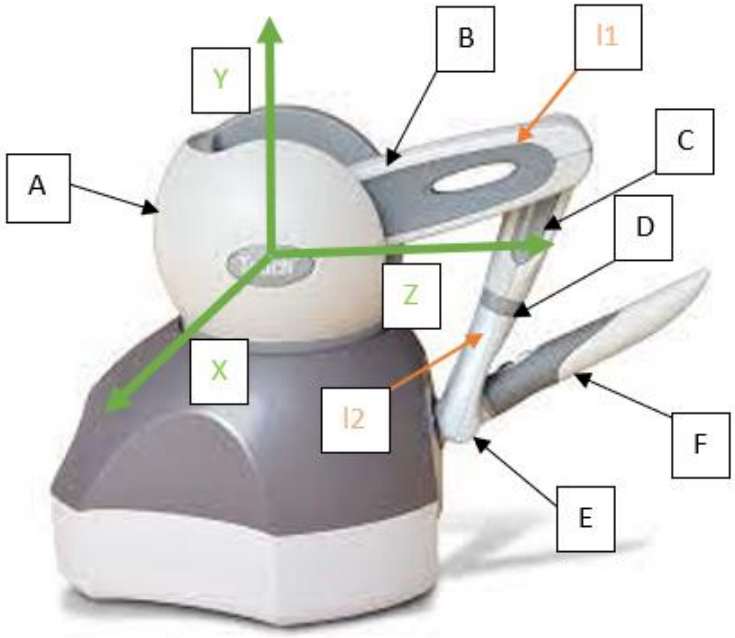

Figura 18. Direcciones ejes del sistema de referencia

Figura 19. Localización del Inkwell

El dispositivo cuenta además con una serie de botones que nos permiten sumar usos, como puede ser la sujeción de objetos o cualquier otra opción que el usuario plantee introducir mediante software. Dos de ellos se sitúan en el stylus, a los que se ha identificado como 0 y 1, estando presente un tercero en el inkwell. Este último únicamente sirve para informar si el dispositivo se encuentra en posición de reposo, en cuyo caso la renderización de fuerzas es desactivada.

Figura 20. Dos botones ubicados en el stylus

Por último, indicar que el inkwell cuenta con un diodo LED que nos informa acerca del estado del dispositivo. Si iniciamos una simulación éste se iluminará con mayor intensidad y si se produce un error durante la simulación parpadeará durante aproximadamente un segundo antes de retornar al estado de menor intensidad.

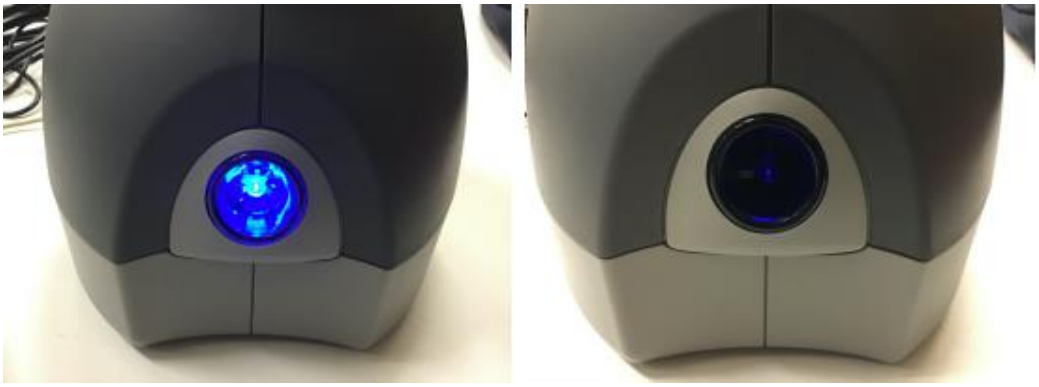

Figura 21. Diodo LED del inkwell

3.1.1.1 Cinemática del 3D Systems Touch

Un brazo robótico puede ser modelado como una cadena articulada en bucle abierto con cuerpos rígidos (eslabones) conectados en serie por juntas revolutivas o prismáticas accionadas por actuadores. Un extremo de la cadena está unido a un sistema de coordenadas fijo en la base del manipulador, mientras que el otro extremoes libre. Para el control de los manipuladores queremos conocer la descripción espacial del extremo libre extremo libre con respecto al sistema de coordenadas de referencia fijo de la base (Spong & Vidyasagar, 1989, Schilling, 2000, Tsai, 1999).

El estudio analítico de la geometría de los manipuladores con respecto a un sistema de coordenadas de referencia fijo sin considerar las fuerzas/momentos que originan dicho movimiento se conoce como estudio cinemático (Barrientos, 1999, Fu, 1987). Existen dos problemas a resolver en la cinemática de un brazo robótico; a la relación que existe entre las coordenadas coordenadas (θ) con las coordenadas

cartesianas (x) y su orientación se le conoce como cinemática directa, mientras que la función inversa se conoce como cinemática inversa. La cinemática se divide en tres: posición velocidad y aceleración.

El problema de la cinemática directa se reduce en encontrar la matriz de transformación homogénea que relaciona el extremo del manipulador con un sistema de referencia en la base del dispositivo. La matriz de transformación homogénea está compuesta de una matriz de rotación (R), un vector de posición (P) cartesiana, un vector de perspectiva (f) y la escala, como se muestra en la ecuación 1.

$$T = \begin{bmatrix} \mathbf{R}_{(3\times3)} & \mathbf{P}_{(3\times1)} \\ \mathbf{f}_{(1\times3)} & 1 \end{bmatrix} \quad (1)$$

Para lograr esta matriz, se utiliza el método conocido como parámetros de Denavit – Hartenberg, el cual se basa en la obtención de cuatro parámetros θi, αi, di y ai, relacionadas con los eslabones i del robot, para definir las matrices de transformación homogéneas de cada eslabón ${}^{i-1}A_i$ se representan como un producto de cuatro transformaciones básicas, como se muestra en la ecuación 2, (Angeles, 1997, Fu, 1987, Kelly, 2003, Reyes, 2011).

$${}^{i-1}A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \quad (2)$$

La matriz de transformación homogénea propuesta por Denavint y Hartenberg se muestra en la ecuación 3.

$${}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Por tanto, en nuestro caso definimos los siguientes sistemas de referencia para el dispositivo háptico, y definimos la siguiente tabla de parámetros Denavit-Hartenberg para cada eslabón del dispositivo:

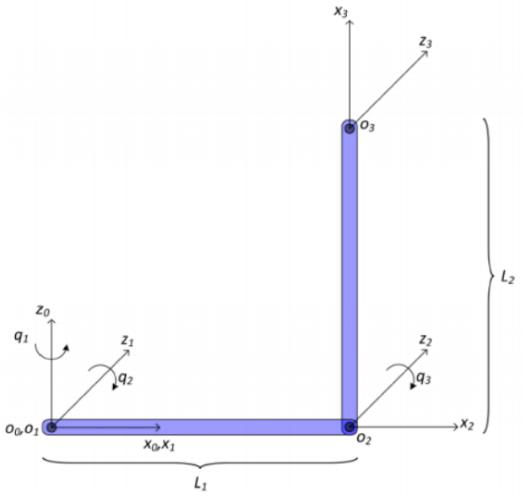

Figura 22. Sistemas de referencia asignados al dispositivo háptico 3D Systems Touch

Link	α_i	θ_i	a_i	d_i
1	$-\frac{\pi}{2}$	q_1	0	0
2	0	q_2	L_1	0
3	0	$q_3 - \frac{\pi}{2}$	L_2	0

Figura 23. Tabla parámetros Denavit.Hartenberg del dispositivo háptico 3D Systems Touch

Ahora ya nos encontramos en disposición de calcular la cinemática directa de nuestro háptico, empezando por las matrices de cada articulación, resultando:

$$A_{01}\begin{bmatrix} cos(\theta 1) & 0 & -sin(\theta 1) & 0 \\ sin(\theta 1) & 0 & cos(\theta 1) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$A_{12}\begin{bmatrix} cos(\theta 2) & -sin(\theta 2) & 0 & l_1 * cos(\theta 2) \\ sin(\theta 2) & cos(\theta 2) & 0 & l1 * sin(\theta 2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$A_{23}\begin{bmatrix} cos\left(\theta 3aux - \frac{\pi}{2}\right) & -sin\left(\theta 3aux - \frac{\pi}{2}\right) & 0 & l_2 * cos\left(\theta 3aux - \frac{\pi}{2}\right) \\ sin\left(\theta 3aux - \frac{\pi}{2}\right) & cos\left(\theta 3aux - \frac{\pi}{2}\right) & 0 & l1 * sin\left(\theta 3aux - \frac{\pi}{2}\right) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Obteniendo así la matriz total del háptico (desde la base al extremo del stylus). La cinemática directa se obtiene multiplicando las matrices elementales T=A01*A12*A23.

En la 4ª columna de T se tienen las coordenadas X (T(1,4)), Y (T(2,4)), Z (T(3,4))

T =

[cos(θ1)*cos(θ2)*cos(θ3 $_{aux}$ - pi/2) - cos(θ1)*sin(θ2)*sin(θ3 $_{aux}$ - pi/2),

- cos(θ1)*cos(θ2)*sin(θ3 $_{aux}$ - pi/2) - cos(θ1)*cos(θ3 $_{aux}$ - pi/2)*sin(θ2),

-sin(θ1),

l1*cos(θ1)*cos(θ2) + l2*cos(θ1)*cos(θ2)*cos(θ3 $_{aux}$ - pi/2) - l2*cos(θ1)*sin(θ2)*sin(θ3 $_{aux}$ - pi/2)]

[cos(θ2)*cos(θ3 $_{aux}$ - pi/2)*sin(θ1) - sin(θ1)*sin(θ2)*sin(θ3 $_{aux}$ - pi/2),

- cos(θ2)*sin(θ1)*sin(θ3 $_{aux}$ - pi/2) - cos(θ3 $_{aux}$ - pi/2)*sin(θ1)*sin(θ2),

cos(θ1),

l1*cos(θ2)*sin(θ1) + l2*cos(θ2)*cos(θ3 $_{aux}$ - pi/2)*sin(θ1) - l2*sin(θ1)*sin(θ2)*sin(θ3 $_{aux}$ - pi/2)]

[- cos(θ2)*sin(θ3 $_{aux}$ - pi/2) - cos(θ3 $_{aux}$ - pi/2)*sin(θ2),
sin(θ2)*sin(θ3 $_{aux}$ - pi/2) - cos(θ2)*cos(θ3 $_{aux}$ - pi/2),

0,

- l1*sin(θ2) - l2*cos(θ2)*sin(θ3 $_{aux}$ - pi/2) - l2*cos(θ3 $_{aux}$ - pi/2)*sin(θ2)]

[0, 0, 0, 1]

(7)

Por lo que respecta a la cinemática inversa, se tienen las siguientes ecuaciones:

$$q_1 = \arctan\frac{Y}{X} \quad (8)$$

$$q_2 = \arctan\frac{-Z}{\sqrt{X^2+Y^2}} - \arccos\frac{K^2+L_1^2-L_2^2}{2L_1K} \quad (9)$$

$$q_3 = -\arccos\frac{-K^2+L_1^2+L_2^2}{2L_1L_2} + (\frac{3\pi}{2}) \quad (10)$$

donde:

$$k = \sqrt{Z^2+X^2+Y^2} \quad (12)$$

Finalmente, se realiza la validación cinemática directa e inversa. Se pueden dar valores a los ángulos y obtener las coordenadas XYZ del sensor a partir de la cinemática directa. A partir de estos valores, se puede calcular la cinemática inversa y comparar los nuevos valores de los ángulos que se obtienen. También se pueden obtener de nuevo los valores de XYZ a partir de los ángulos de la cinemática inversa, y deberán coincidir con los primeros. Como se puede observar en las siguientes capturas de Matlab, los valores coinciden en toda la trayectoria descrita.

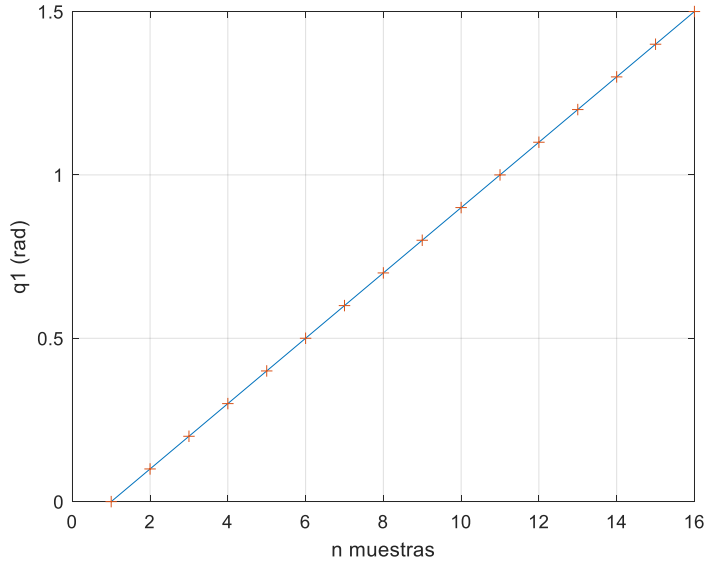

Figura 24. Validación experimental del modelo cinemático del dispositivo háptico 3D System Touch

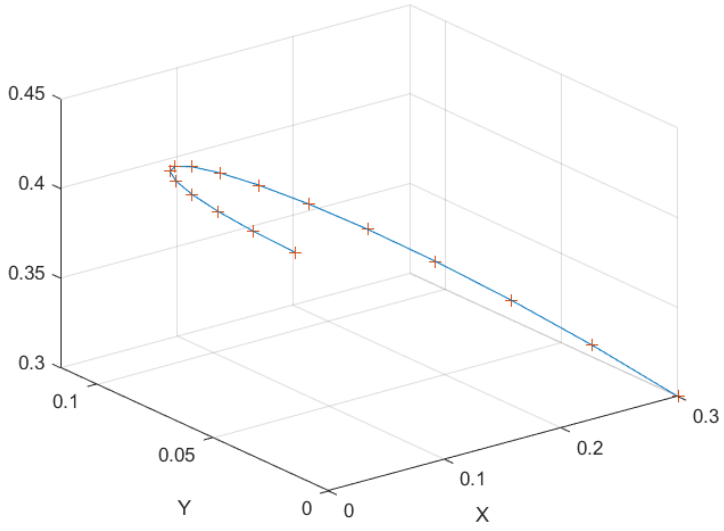

Figura 25. Validación experimental del modelo cinemático del dispositivo háptico 3D System Touch

3.1.1.2 Renderizado de fuerzas

Pasando al ámbito de la renderización de fuerzas, esta consiste en hacer percibir una sensación externa a través de la generación de estímulos sucesivos. Los dispositivos hápticos generan fuerzas miles de veces por segundo, de tal manera que un usuario pueda tener la sensación de tocar e interactuar con un modelo virtual a tiempo real. En la realidad no se trata de una acción constante, si no que se realiza a una frecuencia tan alta que el cerebro se ve engañado y la percibe como si lo fuera.

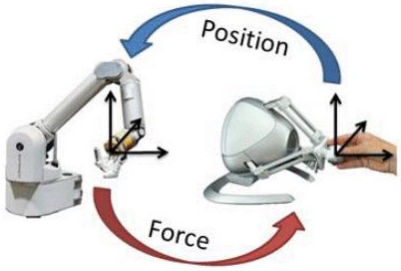

Figura 26. Feedback fuerza posición

Las principales claves para lograr que esta ilusión sea percibida como real, y así obtener una buena interfaz háptica, son:

- Rango de representación de fuerzas ideal: nulo cuando el movimiento es libre, pero adecuado (en dirección y magnitud) cuando se toca o agarra un objeto en el entorno virtual.
- La interfaz debe operar dentro de las habilidades y limitaciones de una persona.
- Sensación de continuidad: continuando con el símil de la generación de animaciones mediante la exposición a imágenes rápidamente cambiantes, el sistema háptico tiene que ser capaz de representar cambios en las fuerzas representadas con la suficiente continuidad como para ser percibidos como continuos. Se requiere una tasa de actualización de al menos 1000Hz para que no se perciban golpes o discontinuidades, especialmente importante a la hora de interactuar con sólidos rígidos.
- Intervalo, resolución y ancho de banda adecuados. El usuario no debe ser capaz de poder atravesar objetos rígidos por exceso de rango de fuerza (Incluye parada mecánica). No puede haber vibraciones involuntarias. Los objetos sólidos deben sentirse.
- La magnitud y la tasa de cambios no son especialmente relevantes, dado que es difícil percibir pequeños incrementos continuos en los valores de las fuerzas. El hecho de que el dispositivo tenga una limitación en la magnitud de representación de la fuerza no impide que éstas se puedan escalar y conseguir los efectos deseados.

Los dispositivos hápticos utilizan una serie de servomotores ubicados en el brazo para crear las fuerzas de retorno en la mano del usuario para simular el tacto y la interacción con objetos virtuales. Los dispositivos proporcionan una retroalimentación de fuerza de 3 o 6 grados de libertad (DOF), según puedan controlar sólo las posiciones del lápiz o también las rotaciones de este. Como ya hemos comentado anteriormente, el dispositivo cuenta con tres motores: uno por cada una de las tres primeras articulaciones (J1, J2 y J3), dejando libres en su movimiento a las tres últimas (J4, J5 y J6). A causa de ello, el sistema solo podrá generar fuerzas en la dirección de los ejes cartesianos (3 GDL).

En modelos más avanzados de la gama de dispositivos hápticos de 3D Systems, sí que se pueden encontrar motores en todas las articulaciones, lo que permite la programación de momentos de torsión en los últimos eslabones (6 GDL). El punto de aplicación de fuerzas, como ocurre en los interfaces hápticos de esta clase, se sitúa en el mismo lugar que el HIP.

3.1.2 Software

Para el control de los dispositivos hápticos a través del PC, es necesario establecer una comunicación continua entre ambos sistemas. Existen librerías para esta tarea, una de las cuales es OpenHaptics, que es la proporcionada por el fabricante del dispositivo. Para ello, deberemos descargar el OpenHaptics® Toolkit versión 3.5.0. En este Toolkit se incluyen el Quick Haptics Micro API, la Haptic Device API (HDAPI), Haptic Library API (HLAPI), los drivers del sistema (PDD), códigos de ejemplo y una serie de recursos.

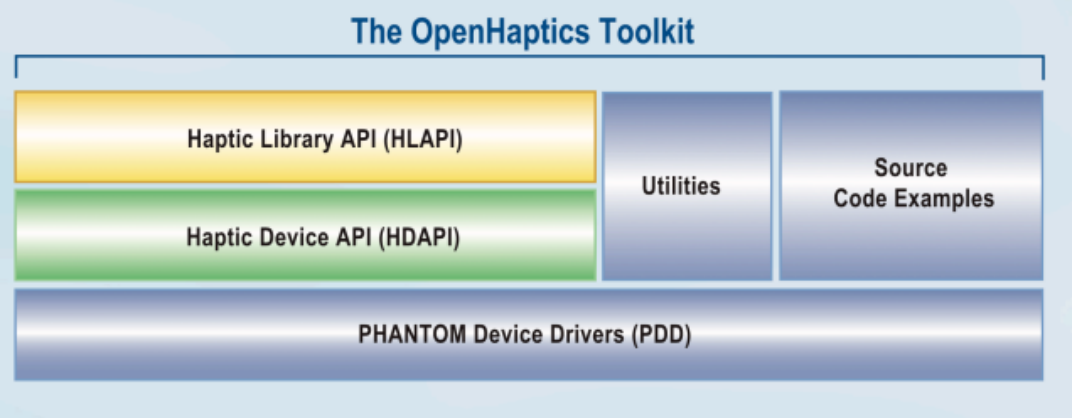

Figura 27. Contenido del OpenHaptics Toolkit

OpenHaptics es fruto de la colaboración de varios fabricantes de dispositivos hápticos con la misión de proporcionar una interfaz común para su programación independientemente del modelo de dispositivo del que se trate. Con OpenHaptics se puede controlar y programar cualquiera de las características de un dispositivo concreto como su configuración de posición o bien la magnitud y el sentido de las fuerzas que se desean generar.

OpenHaptics intenta simplificar la programación encapsulando los pasos básicos comunes a todas las aplicaciones hápticas/gráficas. Esta encapsulación se implementa en las clases C++ de la micro API de QuickHaptics. Al anticiparse a los escenarios de uso típicos, se establece una amplia gama de ajustes de parámetros por defecto que permiten al usuario codificar aplicaciones habilitadas para la háptica de forma muy eficiente.

Los pasos comunes requeridos por las aplicaciones hápticas/gráficas incluyen:

- Analizar los archivos de geometría de los paquetes de animación más populares

- Creación de ventanas gráficas e inicialización del entorno OpenGL

- Inicialización de uno o varios dispositivos hápticos

- Diseño de escenas y cámaras

- Asignación de parámetros de fuerza y rigidez a los objetos de la escena

- Configuración de las callbacks responses a las interacciones

Cabe mencionar que proporcionan una serie de ejemplos en su manual como carga de objetos y eventos callback (picked apples), distintas vitas de ventanas (multiplewindow), propiedades de

cuerpos deformables (spongycow), implementación de fuerzas (skullcoulombforce) o renderizado de modelos hápticos (shapedepthfeedback).

OpenHaptics consta de una serie de librerías ordenadas a modo de lenguajes de distinto nivel de programación: QuickHaptics, HL y HD.

Son APIs que permiten programar los dispositivos hápticos. A continuación, se describirán ordenadas de más alto nivel a más bajo. Obvia decir que en cualquier momento se puede emplear una interfaz de más bajo nivel a la que se estaba usando.

QuickHaptics es una micro API que hace que sea rápido y fácil escribir nuevas aplicaciones hápticas o añadir hápticos a las aplicaciones existentes. Los analizadores de geometría incorporados y los parámetros inteligentes por defecto permiten configurar escenas hápticas/gráficas con una cantidad mínima de código. QuickHaptics proporciona librerías para la creación de múltiples ventanas, carga de ficheros de malla y primitivas de colisión para desarrollo rápido.

La HLAPI proporciona un renderizado háptico de alto nivel y está diseñada para que resulte familiar a los programadores de la API OpenGL®. Permite una reutilización significativa del código OpenGL existente y simplifica en gran medida la sincronización de los hilos hápticos y gráficos. Los Geomagic Touch Device Drivers son compatibles con todos los dispositivos táctiles disponibles actualmente. HL crea dos hilos (threads) de ejecución, el servo-loop (1000Hz) y el de los callbacks de colisión (100Hz), (un callback es una función "A" que se usa como argumento de otra función "B". Cuando se llama a "B", ésta ejecuta "A". Para conseguirlo, usualmente lo que se pasa a "B" es el puntero a "A").

La HDAPI proporciona acceso de bajo nivel al dispositivo háptico, permite a los programadores representar las fuerzas directamente, ofrece control sobre la configuración del comportamiento en tiempo de ejecución de los controladores, y proporciona cómodas funciones de utilidad y ayudas para la depuración.

- Callback asíncrono: función que se ejecuta en cada paso (tick) del planificador (scheduler) a la frecuencia que oscila entre 500 y 2000Hz.
- Callback síncrono: función que se ejecuta una vez, y por la que el hilo principal del programa espera. Se emplea para consultar datos instantáneos sin que haya conflictos entre ambos hilos.

El siguiente diagrama ilustra la relación entre la micro API de QuickHaptics y las capas HD y HL existentes de OpenHaptics.

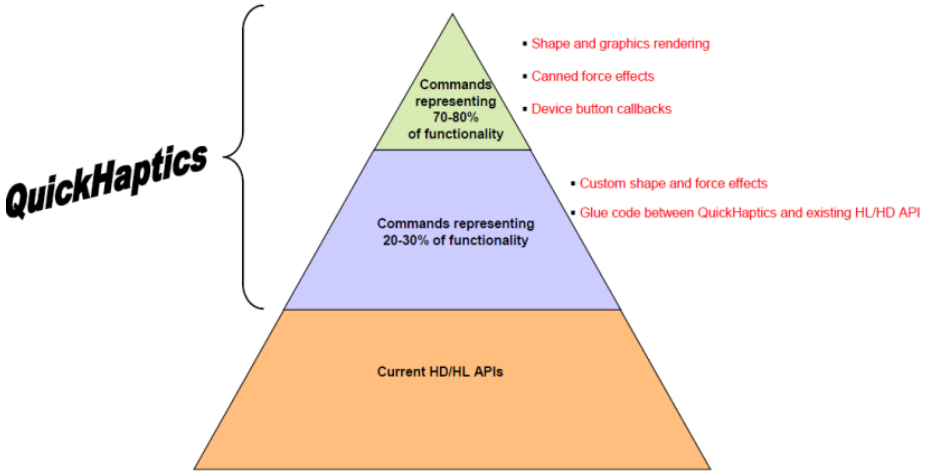

Figura 28. Pirámide de niveles de las librerías OpenHaptics

Nos centramos ahora en la programación multihilo. En sistemas operativos, un hilo de ejecución, hebra o subproceso es una secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo. Un hilo es simplemente una tarea que puede ser ejecutada al mismo tiempo que otra tarea.

Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, la situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente, aunque requieran diferentes tasas de actualización (por ejemplo, el bucle del háptico a 1000Hz y el del programa a ~60Hz). Los hilos de ejecución que comparten los mismos recursos, sumados a estos recursos, son en conjunto conocidos como un proceso. El hecho de que los hilos de ejecución de un mismo proceso compartan los recursos hace que cualquiera de estos hilos pueda modificar estos recursos. Cuando un hilo modifica un dato en la memoria, los otros hilos acceden a ese dato modificado inmediatamente. Debido a este motivo, es muy importante no modificar variables desde distintos hilos, ya que los valores de estas podrían ser inconsistentes.

El proceso sigue en ejecución mientras al menos uno de sus hilos de ejecución siga activo. Cuando el proceso finaliza, todos sus hilos de ejecución también lo han hecho. Asimismo, en el momento en el que todos los hilos de ejecución finalizan, el proceso deja de existir y se liberan todos los recursos.

OpenHaptics ya de por sí segmenta las tareas de representación gráfica en un número de hilos que dependerá de la capa de abstracción que se emplee (QH, HL, HD). Dado que la renderización háptica requiere actualizaciones más frecuentes que las aplicaciones gráficas típicas, el motor de renderización crea, además del hilo principal de la aplicación, dos hilos adicionales que utiliza para el renderizado háptico: el servoloop o ”hilo servo” y el hilo de colisión. El hilo principal de la aplicación en un programa se denomina client thread o “hilo cliente”. El hilo cliente es el hilo en el que se crea el contexto de renderización y en el que las funciones son llamadas por los programas cliente. El código se escribirá para ejecutarse en su subproceso de cliente sin necesidad de conocer sobre que hilo se trata, aunque hay casos en los que sea necesario la ejecución en uno de estos subprocesos. El hilo de alta prioridad (servoloop) tiene un rango de frecuencias entre 500Hz y 2000Hz, este maneja la comunicación directa con el dispositivo háptico. Lee la posición y orientación del dispositivo y actualiza la fuerza enviada al

dispositivo a una velocidad alta (usualmente 1000 hz). Este hilo se ejecuta en una prioridad elevada con el fin de mantener una representación háptica estable. Este hilo solo es visible en el caso del HDAPI, HLAPI y QH oculta el servo del usuario (con la excepción de efectos de fuerza personalizados).

El ServoLoop se refiere al bucle de control usado para calcular fuerzas para enviar al dispositivo háptico. Con el fin de renderizar una respuesta estable, este bucle se debe ejecutar a una tasa de 1khz consistente o mejor. Con el fin de mantener una tasa de actualización tan alta, el bucle de servo es generalmente ejecutado en un hilo separado, de alta prioridad.

Normalmente, los dispositivos hápticos no se emplean de forma aislada. Suelen utilizarse para mejorar la experiencia del usuario junto con un entorno gráfico virtual en 3D. El primer problema que hay que tener en cuenta al combinar la háptica con los gráficos 3D es que la frecuencia de actualización de refresco para mostrar las fuerzas en el dispositivo háptico es más de un orden de magnitud superior a la frecuencia de refresco necesaria para la visualización de imágenes en la pantalla. Esta diferencia se debe a la psicofísica de la percepción humana. Normalmente, una aplicación gráfica refrescará el contenido del framebuffer unas 30-60 veces por segundo para dar al ojo humano la impresión de movimiento continuo en la pantalla. Sin embargo, una aplicación háptica refrescará las fuerzas que se representan en el dispositivo háptico unas 1000 veces por segundo para dar la sensación cinestésica de contacto rígido. Si la velocidad de fotogramas de una aplicación gráfica se ejecuta a una velocidad inferior a 30 Hz, el usuario puede percibir discontinuidades en una animación de tal manera que ya no sea visualmente suave. Del mismo modo, el usuario puede percibir discontinuidades de fuerza y una pérdida de fidelidad cuando el dispositivo háptico se actualiza a una velocidad inferior a 1000 Hz. Como resultado, la háptica y el renderizado de gráficos se realizan normalmente de forma concurrente en hilos separados para que cada bucle de renderizado pueda ejecutarse a su respectiva tasa de refresco.

Más allá de la frecuencia de actualización de los dos bucles de renderizado, también es importante tener en cuenta el tiempo de duración de cada cuadro renderizado. Para el renderizado de gráficos, mantener una frecuencia de refresco de 30 Hz requiere que todo el renderizado tenga lugar en 1/30 de segundo (es decir, 33 ms). Comparándolo con la duración de 1 ms del bucle de renderizado háptico, se hace evidente que hay una disparidad significativa en la cantidad de tiempo disponible para realizar los cálculos de renderizado háptico frente a los cálculos de renderizado gráfico. En ambos casos, los bucles están generando fotogramas, excepto que el bucle de háptica está generando ~33 fotogramas cada vez que el bucle de gráficos genera uno. Este es un punto muy importante a tener en cuenta, especialmente en una aplicación en la que más de un objeto se mueve en la pantalla o se siente por la simulación háptica al mismo tiempo.

A continuación, se muestra el diagrama de flujo empleado en un programa de Open Haptics y en él se puede ubicar este servoloop, que sirve como canal de información entre HDAPI y el dispositivo conectado.

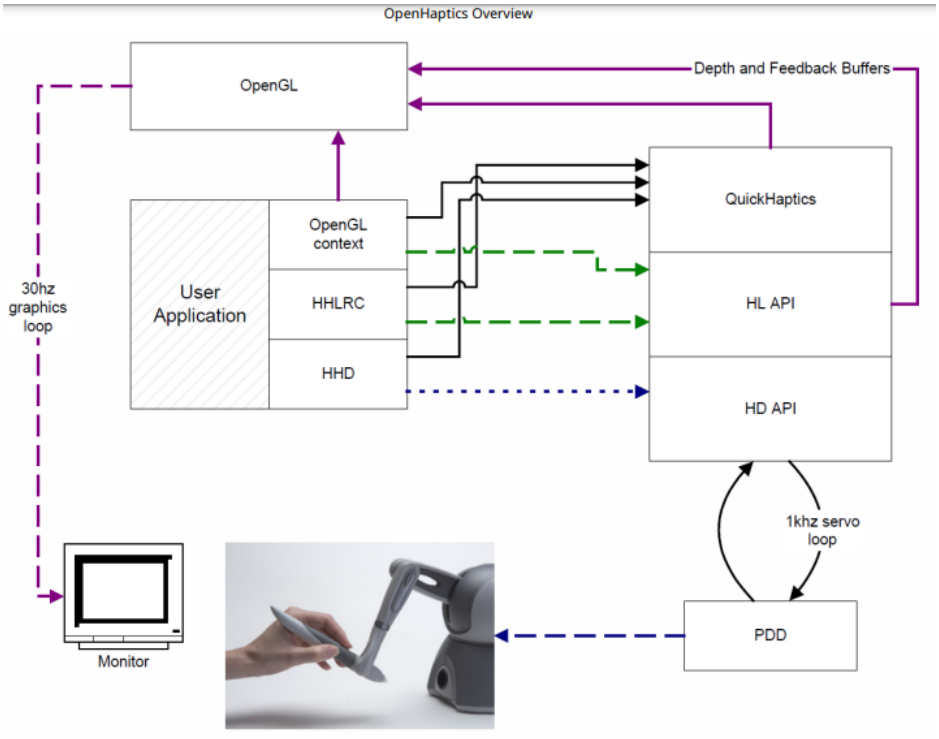

Figura 29. Diagrama funcionamiento OH con servoloop

El bucle de hilo principal o servoloop se ejecuta en paralelo con otros hilos. Para que el programa sepa a qué hilo prevalezca sobre otro y evitar conflictos entre ellos, es necesario definir en que consiste la prioridad de un hilo.

La prioridad de un hilo indica la preferencia que tiene un hilo sobre otro. Se utiliza para decidir cuándo pasar a ejecutar otro hilo (cambio de contexto).

- Cuando un hilo cede el control (por abandono explícito o por bloqueo), se ejecuta a continuación el que tenga mayor prioridad.
- Un hilo puede ser desalojado por otro con prioridad más alta, tan pronto como este desee hacerlo. Pero en la práctica la cantidad de CPU que recibe cada hilo depende además de otros factores como la forma en que el sistema operativo implementa la multitarea.

El scheduler o “programador” gestiona un hilo de alta prioridad para enviar fuerzas y recuperar información del estado del dispositivo. Normalmente, las actualizaciones de la fuerza necesitan ser enviadas a una frecuencia de 1000 Hz para crear una respuesta de fuerza convincente y estable. El scheduler permite que la aplicación se comunique de forma efectiva con el subproceso del bucle de una manera segura y agregue operaciones. Las diferentes tareas que puede gestionar son:

- Tareas asíncronas: típicamente se ejecutan una vez por ciclo y de manera continua, independientemente del resto del programa.
- Tareas síncronas: se ejecutan sólo una vez y el servoloop espera a que hayan finalizado. Se emplean principalmente para transferir información entre el servoloop y el resto del programa, ya que en ese momento varios hilos no pueden estar modificando los mismos datos (posición, fuerza). Es decir, como las variables de estado y datos se comparten puede haber problema de inconsistencias en su lectura o escritura cuando varios hilos tratan de acceder a

un mismo dato simultáneamente. Estas tareas permiten el acceso a esas distintas frecuencias que tendrán los hilos.

La sincronización de estado se vuelve importante cuando se administra una interfaz de usuario que incluya tanto gráficos hápticos como gráficos, porque hay dos bucles de renderización que necesitan acceso a la misma información. Esto suele implicar la realización de copias seguras de los datos cada subproceso como una instantánea de estado.

En ciertas ocasiones es posible no manejar directamente el ServoLoop, y programar únicamente funciones que definen el comportamiento que debe presentar el dispositivo ante situaciones previstas (son los llamados callbacks): pulsación de botones, detección de contacto contra otros objetos, etc.

Definimos callbacks como funciones que se llaman como resultado de un evento, son funciones de argumentos ya establecidos para modificar el algoritmo de un código existente, en este caso la librería. Esto permite desarrollar capas de abstracción de código genérico a bajo nivel que pueden ser llamadas desde una subrutina (o función) definida en una capa de mayor nivel.

3.1.2.1. Haptic Device API (HDAPI)

Como ya hemos comentado anteriormente, la HDAPI nos permite acceso de bajo nivel al sistema háptico, otorgándonos la capacidad tanto de conocer pormenores del estado actual del sistema como de manipularlos. Un ejemplo de ello es que nos permitirá modificar el servoloop, de tal manera que cumpla con las necesidades del usuario.

Las operaciones del dispositivo háptico incluyen todas las operaciones asociadas con la obtención y la definición del estado. Dichas operaciones deben realizarse exclusivamente dentro del servoloop mediante el uso de las callbacks del programador. Estas llamadas se pueden hacer de forma segura en la aplicación, pero sólo cuando el programador no se está ejecutando.

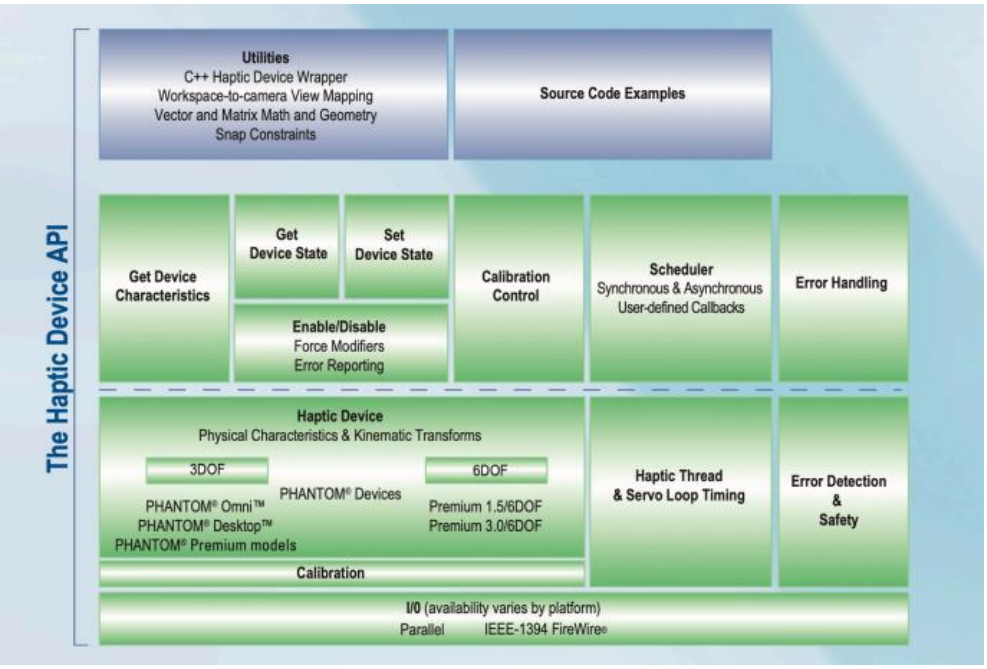

Figura 30. Contenido de la HDAPI

Algunas otras funcionalidades que podremos extraer del sistema a través de la HDAPI son las siguientes:

Estado del sistema

Podemos conocer posición, orientación y velocidad del stylus respecto al eje de coordenadas cartesianas del dispositivo, los ángulos de rotación descritos por las articulaciones y las lecturas de los encoders/DAC, además del estado de los botones. Podremos conocer también la fuerza actual que está ejerciendo el sistema, duración del último ciclo de servicio (servo loop), además de una larga lista de otros datos que se detallan en la guía "API Reference Guide".

Características del sistema

Indica modelo, versión, número de serie, dimensiones del espacio de trabajo, valores máximos de fuerza y velocidad entre otros, temperatura de los motores o capacidades de calibración. De nuevo, la lista completa se encuentra en la "API Reference Guide". Interacción con un robot colaborativo mediante un interfaz háptico 41 Modificar el estado del sistema Renderizar fuerzas y torques (torques solo en modelos que lo permitan) en el espacio cartesiano, colocar los motores en posiciones determinadas o modificar el estado de los LEDs del sistema.

Activación y desactivación

De fuerzas, restricción de movimiento respecto a un punto o superficie (constraint) o comprobaciones y limitaciones de excesos de fuerza o velocidad.

Envío de errores

Errores acerca del funcionamiento del sistema, renderización de fuerzas, scheduler...

Calibración

Tanto manual como automática.

Scheduler

Ajustable de tal manera que se pueda conseguir ciclos de servicio determinados, llamadas (peticiones de servicios) en instantes determinados o sincronización entre los hilos del proceso háptico y gráfico de la manera determinada por el usuario.

Recursos

Adaptación de la programación del sistema a C++, generación del espacio virtual en función del espacio de trabajo del dispositivo, geometría y cálculos de vectores y matrices.

Trabajar a través de la HDAPI nos permite, como ya se ha comentado un control muy detallado del sistema, pudiendo obtener los valores en cada momento de la situación espacial del equipo o dar consignas concretas de fuerza.

Por ejemplo, para obtener información del dispositivo se utiliza la función *hdGet (Parameter Values)*, pudiéndose obtener información tanto de la última posición leída por el dispositivo o del estado de los botones del stylus:

```
HDdouble position[3];
hdGetDoublev(HD_CURRENT_POSITION,position);
```

```
HDint buttons;
hdGetIntegerv(HD_CURRENT_BUTTONS,&buttons);
```

Para ello, primero se ha definido un array de tipo *double* con al menos tres posiciones para almacenar el dato de la posición (proporcionado en mm), y después se ha llamado a la función *hdGetDoublev*, indicando entre paréntesis primero el parámetro que queremos conseguir, en este caso la posición y en segundo lugar la variable en la que queremos almacenarlo. Ídem con el ejemplo de los botones, pero definiendo una variable *int* y llamando a *la función hdGetIntegerv.*

Si en lugar de extraer un dato lo que queremos es fijar una consigna, por ejemplo, queremos que el dispositivo ejerza una fuerza determinada, debemos utilizar la función *hdSet (Parameter Values):*

```
hduVector3Dd forces(0.2,0.5,0);
hdSetFloatv(HD_CURRENT_FORCE,forces);
```

Para manipular el dispositivo háptico mediante la HDAPI se necesita un conocimiento previo acerca del funcionamiento del dispositivo y una idea muy concisa del resultado que quiere obtener de su programación. Los modelos deberán crearse íntegramente por el programador, y a pesar de que su uso puede combinarse con funcionalidades de la HLAPI, que pertenece a un más alto nivel y en consecuencia es más sencillo a la hora de programar, continúa poseyendo una complejidad considerable.

3.1.2.2 Haptic Library API (HLAPI)

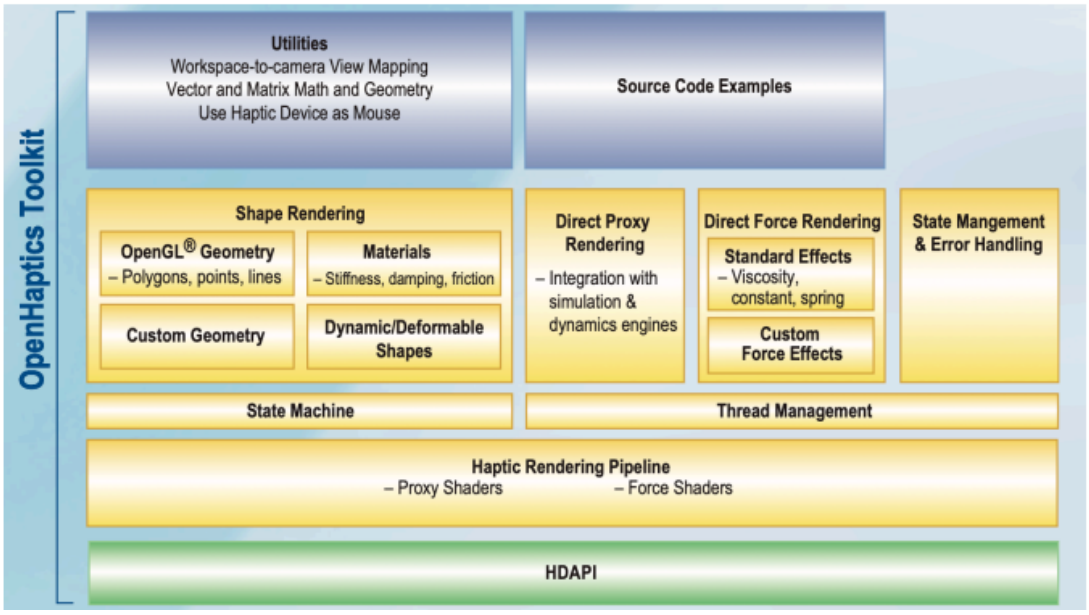

Figura 31. Contenidos de la HLAPI

Como también se ha discutido anteriormente, la HLAPI permite la renderización háptica a mayor nivel que la HDAPI, habiendo sido diseñada para resultar familiar a los usuarios que posean un conocimiento de la OpenGL API, pero que no cuenten con la misma experiencia en el ámbito de la renderización háptica. Se podría decir que busca una mayor sencillez a cambio de la perdida de prestaciones. Cuenta con una serie de efectos hápticos definidos (como muelle o viscosidad entre otros) que permiten al

dispositivo emular esos comportamientos en función de los parámetros determinados para cada efecto, pero se pierde la capacidad de introducir fuerzas en direcciones concretas con valores en newtons.

Se pueden introducir efectos varios como viscosidad, fricción o efecto muelle sin tener que generar todo un modelo matemático que relacionara la fuerza generada con la velocidad del sistema. Con la HLAPI se puede obtener por medio de funciones como *hlEffectd,* que establece el valor de una propiedad de efecto que se aplicará al efecto generado por la siguiente llamada a las funciones *hlStartEffect()* o *hlTriggerEffect().*

```
hlBeginFrame();
hlEffectd(HL_EFFECT_PROPERTY_DURATION, pulseLength);
hlEffectd(HL_EFFECT_PROPERTY_GAIN, 0.4);
hlEffectd(HL_EFFECT_PROPERTY_MAGNITUDE, 0.4);
hlTriggerEffect(HL_EFFECT_FRICTION);
hlEndFrame();
```

De acuerdo con lo anterior se consigue una programación más sencilla, pero tanto en cuanto los valores de Gain (ganancia del efecto) y Magnitude (fuerza máxima que se puede ejercer) solo pueden tomar valores entre 0 y 1 (menor y mayor valor posibles), perdemos control sobre las fuerzas generadas en comparación con la HDAPI.

Presenta las siguientes funcionalidades:

Generación de formas

Uso de primitivas de OpenGL (polígonos, puntos y líneas) para la generación de objetos. En HDAPI tendríamos que introducir uno a uno los vértices de los objetos para generar contactos.

Efectos hápticos

Efecto muelle, viscosidad, fricción, fuerzas constantes y vibración.

Modelos de contacto

Dos modalidades: contacto y restricción (anclaje a un punto).

Propiedades de superficies

Fricción, valores de dureza y amortiguación superficiales en una o ambas caras de una pieza.

Dinámica y objetos deformables

Capacidad de integración de productos creados por terceras partes.

3.1.2.3 HDAPI vs HLAPI

La decisión de usar HDAPI o HLAPI se tomará en función del uso previsto del dispositivo por parte de cada usuario. HLAPI está diseñado para que sea fácil de usar por personas que tienen capacitación previa en OpenGL pero que carecen de experiencia en la representación de fuerzas hápticas. También se recomienda si desea agregar física u objetos al escenario virtual, ya que su implementación será más sencilla (permite el uso de bibliotecas con físicas, a diferencia de HDAPI).

Por otro lado, si queremos centrarnos en implementar modelos que realmente representen el modelo físico y que no impliquen una integración compleja de objetos, es muy recomendable hacerlo desde HDAPI, ya que nos dará mucho más control sobre la interfaz.

Las dos API no son mutuamente excluyentes, como se indica en el manual de programación, HDAPI puede usarse para "complementar" la funcionalidad de HLAPI. En otras palabras, es posible decidir utilizar efectos hápticos preparados por HLAPI, como la vibración, junto con instrucciones que reproducen una determinada fuerza, como la gravedad.

En las aplicaciones realizadas durante este proyecto, se ha optado por la utilización de HDAPI, ya que en nuestro caso no ha sido necesario simular un entorno virtual cargado de objetos sino que más bien se ha buscado implementar un modelo matemático con el que representar ciertas fuerzas según el estado de nuestro robot.

3.2 Robot

3.2.1 ABB IRB 140

Uno de los robots disponibles en el laboratorio de robótica es el robot articulado IRB 140, desarrollado por el grupo ABB Automation Technologies. Se trata de un robot industrial multiporpósito de 6 ejes que maneja una carga de 6 kg con un espeso alcance (810 mm). Todos los brazos mecánicos están completamente protegidos con IP67, lo que hace que el IRB 140 sea fácil de integrar y adecuado para una variedad de aplicaciones. Presenta un radio de área de trabajo excepcionalmente extendido debido al mecanismo de flexión hacia atrás de la parte superior del brazo, así como un eje 1 de rotación de 360º incluso montado en la pared. El diseño compacto y robusto con cableado integrado aumenta la flexibilidad general. La opción de detección de colisiones, con retracción de trayectoria completa, garantiza que el robot sea confiable y seguro.

A continuación se muestra la hoja de especificaciones técnicas de dicho robot:

Tipo	articulado
Número de ejes	6 ejes
Funciones	de manipulación, para ensamblaje, de empaque, de soldadura al arco, de desbastado, de tratamiento de superficie, para pulverización
Instalación / movilidad	de suelo, para instalación en pared
Tipo de protección	IP67
Otras características	de alta velocidad, compacto
Carga máxima	6 kg (13,228 lb)
Radio de acción	800 mm (31,5 in)
Velocidad	100 mm/s, 500 mm/s, 1.000 mm/s
Repetibilidad	0,03 mm (0,0012 in)
Peso	98 kg (216,1 lb)

Figura 32. Especificaciones técnicas robot ABB IRB 140

Este ha sido finalmente el robot escogido para la realización de las aplicaciones, por estar disponible dentro de RobotStudio, el software de simulación y programación offline de ABB del que luego hablaremos.

Figura 33. Robot industrial ABB IRB 140

3.2.2 Universal Robots UR3

Otro de los robots del que disponemos es el El UR3e de Universal Robots. Se trata de un robot industrial colaborativo ultraligero y compacto, por lo que se convierte en un robot ideal para la aplicación sobre mesas de trabajo. Su tamaño reducido lo convierte en el más adecuado para implementarse directamente dentro de maquinaria o en otros espacio de trabajo pequeños en los que se deben manipular objetos ligeros. Este cobot pesa tan solo 11 kg, pero tiene una carga útil de 3 kg. Su rotación ±360 grados en todas las articulaciones y su rotación infinita en el extremo hacen que sea ideal para ensamblajes ligeros y aplicaciones de atornillado. Su alcance es de 500 mm y su huella, es decir, el espacio que ocupa en el espacio de trabajo, es de 128 mm.

Figura 34. Robot colaborativo UR3

3.2.3 Software

El programa utilizado para la simulación del entorno de trabajo del robot ABB IRB 140 ha sido RobotStudio. RobotStudio es el software de simulación y programación offline de ABB, basado en ABB VirtualController, una copia exacta del software real que ejecuta sus robots en producción. Esto permite realizar simulaciones muy realistas, utilizando programas de robot reales y archivos de configuración idénticos a los utilizados en el taller.

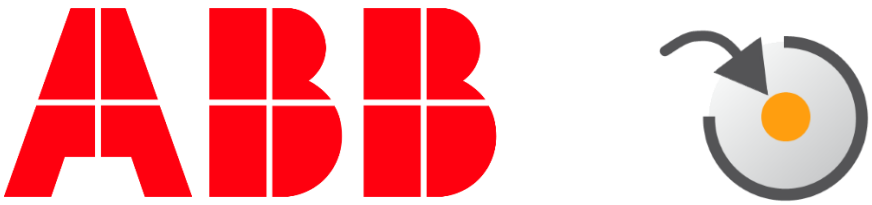

Figura 35. Logotipo de ABB (izq) y del software robotstudio (dcha)

La herramienta de programación offline RobotStudio permite a los usuarios crear, simular y probar una instalación completa de robot en un entorno virtual 3D sin tener que visitar o perturbar su línea de producción real.

RobotStudio aporta herramientas que aumentan la rentabilidad del sistema de robots, pues permite realizar tareas tales como formación, programación y optimización de programas sin alterar la producción. Esto añade muchas ventajas, entre ellas:

- Reducción de riesgos

- Arranques más rápidos

- Menor tiempo para modificaciones

- Aumento de la productividad

3.2.3.1 Lenguaje RAPID (Robotics Application Programming Interactive Dialogue)

RAPID es un lenguaje de programación textual de alto nivel desarrollado por la empresa ABB. Se trata de un lenguaje específico orientado totalmente a la programación del robot y de sus periféricos, que permite al usuario crear sus propias instrucciones y que además permite la ejecución de varias tareas en paralelo.

Se encuentra estructurado a dos niveles: módulos por una parte y funciones y rutinas por la otra. De esa manera, el programa estará compuesto por uno o más módulos de programa, que a su vez contendrán los datos y rutinas específicos, siendo solo uno de estos módulos de programa los que contengan la rutina principal (main). También disponemos de los módulos sistema, que definen datos y rutinas específicos (sistemas de referencia base del robot, brida elemento terminal…) que deben permanecer en el sistema incluso al cambiar de programa.

4. DESARROLLO DE APLICACIONES DE CONTROL REMOTO

En este capítulo se abordará en primer lugar el tema de la comunicación entre la máquina y el robot (en este caso entre la máquina y el software RobotStudio), para seguidamente realizar una explicación completa de la estación desarrollada en RobotStudio *detecta_colisiones* a partir de la cual se han llevado a cabo las aplicaciones mediante el uso del dispositivo háptico.

4.1 Conexión dispositivo 3D Systems Touch

El primer paso para lograr la correcta implementación del dispositivo háptico es comprobar que la conexión con la máquina se ha realizado sin problemas. En primer lugar, es necesario definir un nombre al dispositivo y asignar el puerto USB al que corresponde. Para ello se puede emplear un toolkit o "kit de herramientas" que trae OpenHaptics llamado GEOMAGIC, el cual permite a los desarrolladores integrar dispositivos hápticos con aplicaciones de terceros existentes y nuevas aplicaciones. En este kit se incluyen dos programas Geomagic Touch Setup y Geomagic Touch Diagnostic. El primero se emplea para la tarea anteriormente dicha de conectar dispositivos por primera vez.

Tras conectar un dispositivo háptico el programa reconoce en que puerto se ha conectado. Lo único que es necesario hacer, es asignar un nombre al dispositivo, en este caso se llamará *Default Device* puesto que solo tenemos un dispositivo conectado.

El siguiente paso consiste en calibrar el dispositivo y comprobar la correcta conexión anteriormente realizada. Para ello se emplea el programa Geomagic Touch Diagnostic. La calibración simplemente se efectúa colocando el lápiz en el tintero. Es posible observar la posición en tiempo real del dispositivo en ese momento, como se puede ver en la siguiente imagen.

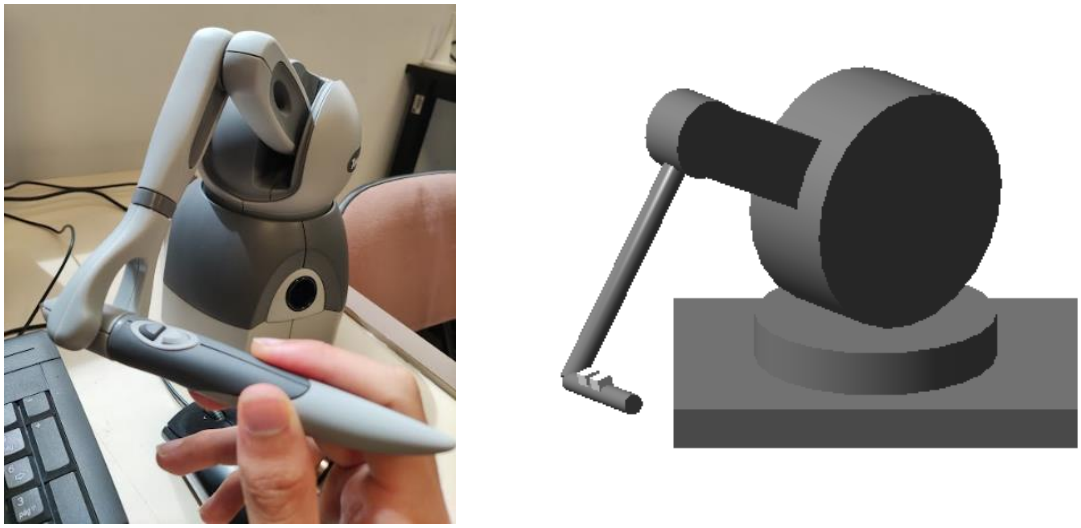

Figura 36. Representación en Touch Diagnostics de la posición del interfaz háptico

4.2 Estación *detecta_colisiones* RobotStudio

Nos centramos ahora en el desarrollo de la estación llamada *detecta_colisiones*, el objetivo de la cual es el de contar con un sistema que permita al robot industrial detectar objetos que se encuentren dentro de su entorno de trabajo. Por tanto, pasaremos en primer lugar a explicar los diferentes elementos que conforman la estación.

Como bien hemos comentado en el apartado dedicado al robot a utilizar, en este caso se trata del robot industrial ABB IRB 140, el cual se encentra disponible dentro de la biblioteca de robots de ABB que encontramos en RobotStudio. Así pues, seleccionamos dicho robot y lo implementamos tal y como se puede observar en la siguiente figura.

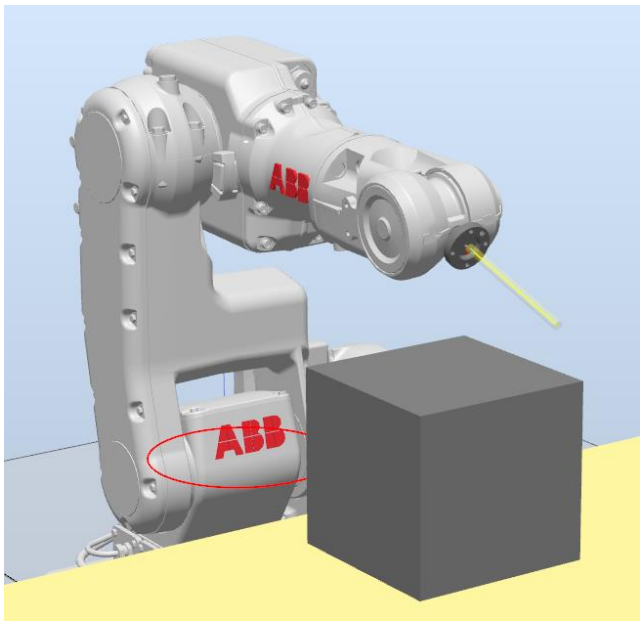

Figura 37. Robot ABB IRB 140 dentro de RobotStudio

En segundo lugar, tratamos de buscar alguna solución que nos permitiese detectar posibles objetos con los que nuestro robot pudiese colisionar sin ser deseado. Por tanto, decidimos implementar un sensor que nos indicara en todo momento si existe algún objeto en las inmediaciones del robot.

En las siguientes imágenes podemos observar la lógica implementada en la estación, en la que se puede advertir el proceso de obtención de la distancia del sensor a cualquier objeto con el que interactúe. Vemos como nuestro sensor tiene solamente como entrada la señal que lo activa (salida del robot) y como por medio de unos sensores de posición procedentes del sensor de línea (amarillo) y las relaciones matemáticas que se observan finalmente se obtiene la posición a la cual se encuentra el objeto de nuestro robot en las tres direcciones XYZ, siempre y cuando se “corte” dicho sensor de línea. Cabe destacar la distancia que posee el sensor respecto de la posición de nuestro robot, no se encuentra sobre la superficie del robot sino que tiene varios centímetro de longitud, ya que nos permitirá aplicar con seguridad una ley de fuerzas que abordaremos más adelante.

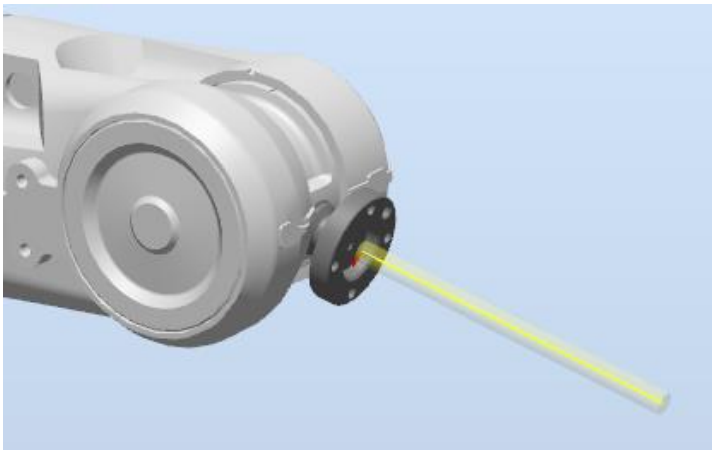

Figura 38. Sensor de distancia

Figura 39. Lógica de la estación

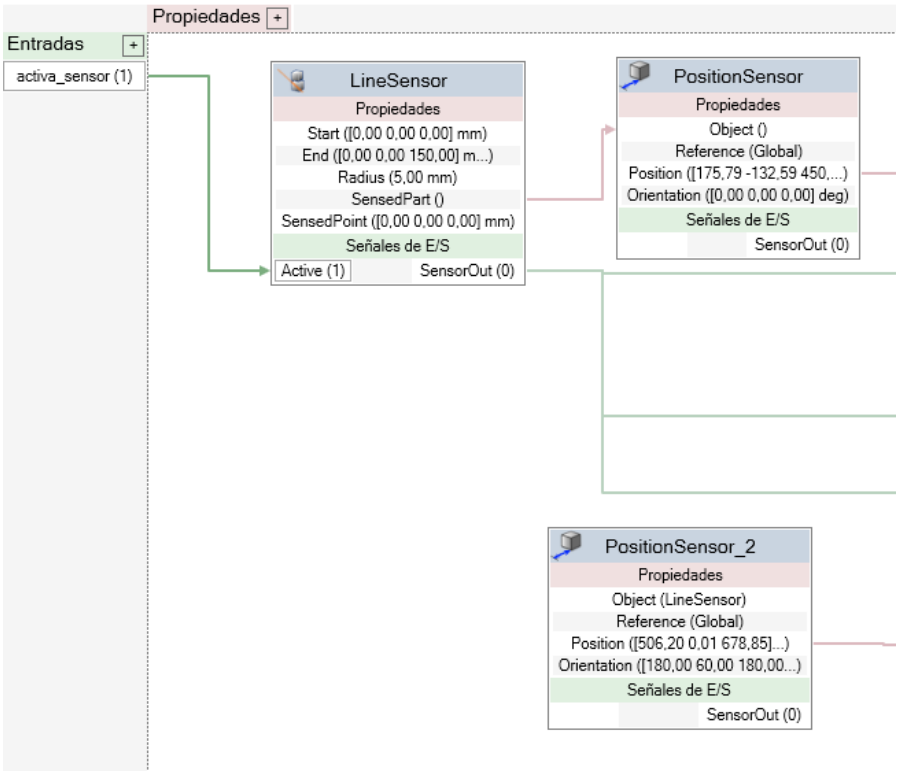

Figura 40. Primera parte de las propiedades del sensor de posición

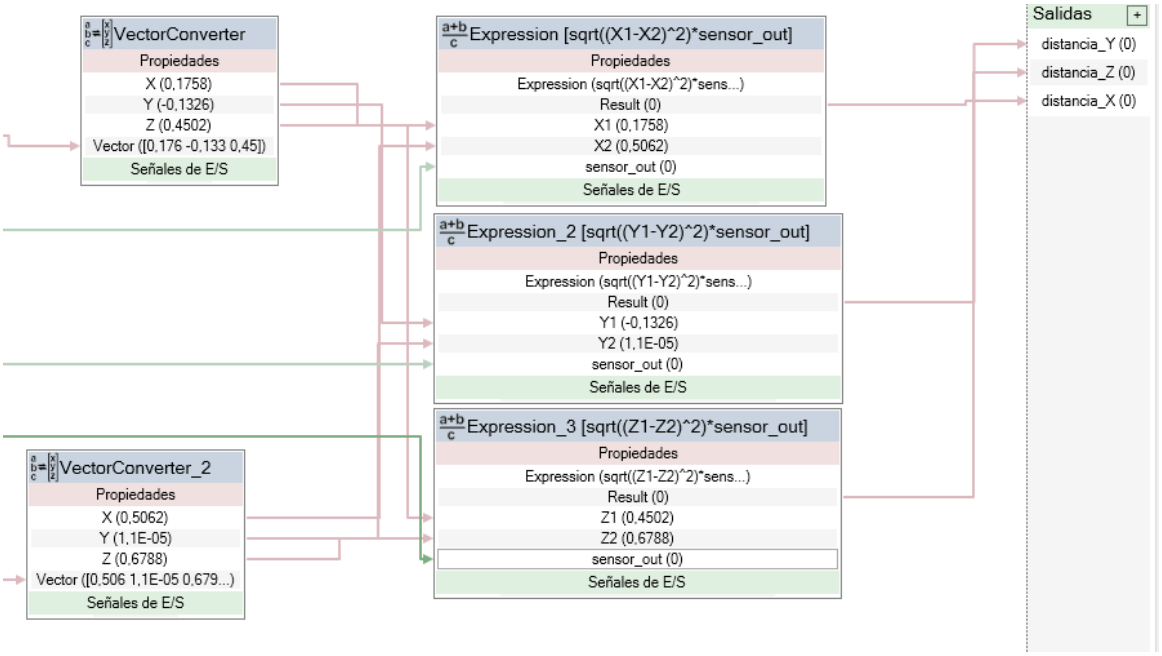

Figura 41. Segunda parte de las propiedades del sensor de posición

4.3 Teleoperación del robot mediante el dispositivo 3D Systems Touch

Una vez contamos ya con la estación que nos permite detectar objetos, el siguiente paso es el de lograr que el robot se mueva al mismo tiempo que el efector de nuestro dispositivo háptico, es decir, que el efector del robot esclavo siga la posición objetivo dada por el efector del controlador maestro. El control se lleva a cabo a través de la cinemática inversa del brazo, con lo cual sólo se necesita saber la posición cartesiana de ambas puntas efectoras (esclavo (x_s, y_s, z_s) y maestro (x_M, y_M, z_M)) y posteriormente se realiza una transformación lineal proporcional para llevar la posición del efector maestro desde su propio espacio de trabajo ((x_M, y_M, z_M)), hacia el workspace del brazo esclavo (slave goal (x_{SG}, y_{SG}, z_{SG})). El brazo esclavo entonces se moverá desde su posición actual (x_s, y_s, z_s) hasta el punto objetivo (x_{SG}, y_{SG}, z_{SG}) determinado por la posición del efector maestro.

Para ello, en primer lugar se ha de crear el socket que permita la conexión entre ambos dispositivos. Una vez creado y establecido la conexión, el programa espera hasta que se pulse por primera vez el botón 1 del dispositivo háptico, indicando que a partir de que se pulse el usuario está listo para ejecutar la aplicación. Una vez pulsado, se realiza una llamada sincronizada para copiar el estado más actual del dispositivo (copyDeviceDataCallback). Esta llamada del programador asegura que el estado del dispositivo se obtiene de una manera segura para los hilos. De esta manera, una vez se pulsa el botón, el háptico manda la posición del efector por medio de una cadena de caracteres, que es recibida por el programa RAPID del robot. Realmente lo que manda son las coordenadas XYZ del efector, separadas por un ";".

```
hdBeginFrame(hdGetCurrentDevice());

/* Retrieve the current button(s). */
hdGetIntegerv(HD_CURRENT_BUTTONS, &nButtons);

/* In order to get the specific button 1 state, we use a bitmask to
   test for the HD_DEVICE_BUTTON_1 bit. */
gServoDeviceData.m_buttonState =
        (nButtons & HD_DEVICE_BUTTON_1) ? HD_TRUE : HD_FALSE;

/* Get the current location of the device (HD_GET_CURRENT_POSITION)
   We declare a vector of three doubles since hdGetDoublev returns
   the information in a vector of size 3. */
hdGetDoublev(HD_CURRENT_POSITION, gServoDeviceData.m_devicePosition);

/* Also check the error state of HDAPI. */
gServoDeviceData.m_error = hdGetError();
```

Figura 42. Obtención del estado de los botones y de la posición del efector del dispositivo háptico

Una vez recibida la información de la posición, es necesario separar la cadena de caracteres y almacenar el valor de cada una de las coordenadas XYZ en una variable numérica, pero teniendo en cuenta que en RobotStudio el sistema de coordenadas es distinto del que emplea el dispositivo háptico, por lo que la coordenada Y que mandamos desde el háptico debemos almacenarla como coordenada Z en RAPID, la X como Y y la Z como X. Se trata de una transformación sencilla de sistema de coordenadas.

```
PROC lectura_socket()
    SocketReceive socket1\Str:=stReceive;
        TPWrite stReceive;
    longitud_trama:=StrLen(stReceive);

    IF longitud_trama>0 THEN
        posRX:=StrFind(stReceive,1,";");
        posRY:=StrFind(stReceive,posRX+1,";");
        posRZ:=StrFind(stReceive,posRY+1,";");

        ok:=StrToVal(StrPart(stReceive,1,posRX-1),valor_RY);
        ok:=StrToVal(StrPart(stReceive,posRX+1,posRY-posRX-1),valor_RZ);
        ok:=StrToVal(StrPart(stReceive,posRY+1,posRZ-posRY-1),valor_RX);
```

Figura 43. Lectura de la posición del efector del háptico y transformación de la base de coordenadas

Así pues, lo único que queda por hacer es programar la instrucción de movimiento del robot a la posición almacenada, multiplicada por un factor de escala dependiente de la aplicación y el entorno de trabajo.

A partir de haberse pulsado por primera vez el botón 1 del stylus, se crea un bucle que permite al dispositivo háptico estar continuamente enviando la posición de su efecto para que de esta manera el robot se mueva al mismo tiempo que él, logrando una sensación de continuidad. De esta manera se logra que el robot se mueva en consonancia con el dispositivo háptico, como se puede ver en las siguientes capturas.

Mencionar que se reserva el botón 2 del háptico para congelar este proceso mientras se mantenga pulsado, con el fin de permitir la detención del proceso en caso de necesitarse, por ejemplo, la extracción de algún objeto en el workspace del robot.

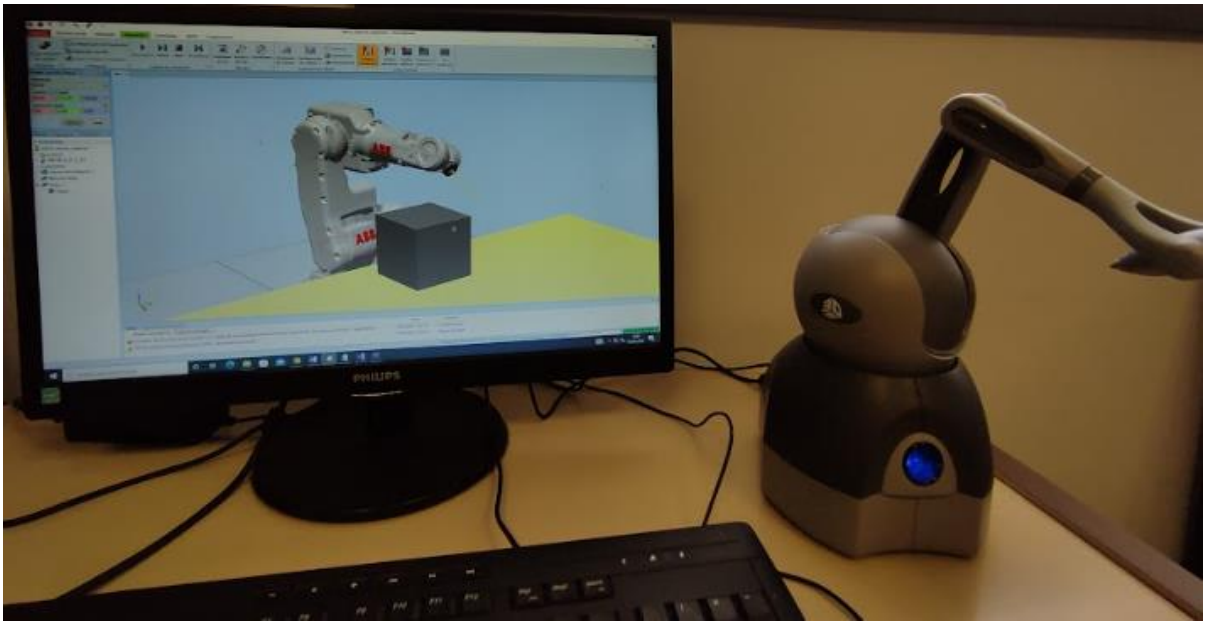

Figura 44. . Movimiento del robot en sinconía con el efector del 3D System Touch

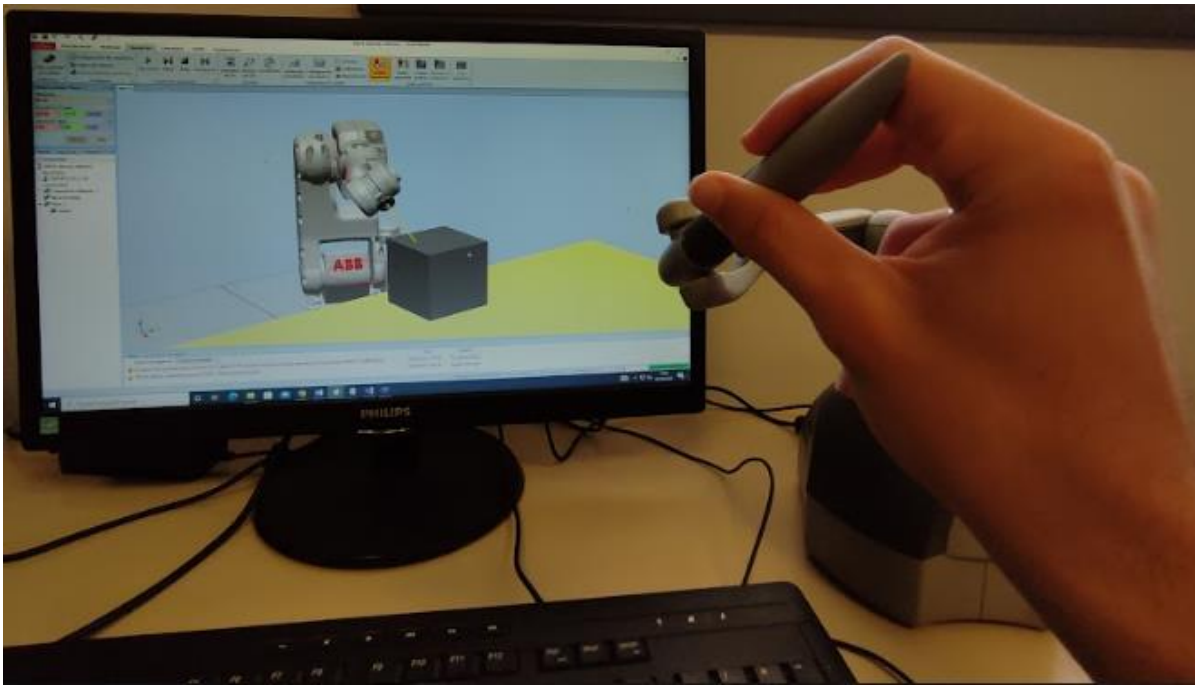

Figura 45. Movimiento del robot en sinconía con el efector del 3D System Touch

4.4 Célula robótica anti-colisiones

En esta aplicación lo que se busca es programar el dispositivo háptico para que en el momento en el que el sensor del robot detecte que hay un objeto dentro de su trayectoria, este produzca una fuerza que impida mover el stylus (y en consecuencia el robot) con el fin de evitar un posible impacto que podría resultar fatal.

Por tanto, siguiendo con lo programado anteriormente, ahora se quiere conseguir una retroalimentación de fuerzas cuando el sensor muestre valores de distanciamiento de un objeto. Si bien cuando el robot se mueve libremente por el espacio de trabajo el sensor está continuamente dando un valor de 0 en las tres coordenadas, en el momento en el que el sensor lineal (haz de luz amarillo) es atravesado por un objeto provoca que el sensor aporte un valor de distancia en cada una de las coordenadas. En las siguientes capturas puede observarse como en primer lugar los valores de *distancia X, distancia Y* y *distancia Z* valen 0 debido a que el sensor lineal no es atravesado por ningún objeto. En cambio, cuando se coloca un objeto que atraviesa dicho sensor, puede observarse como las variables anteriormente mencionadas toman valores distintos de 0.

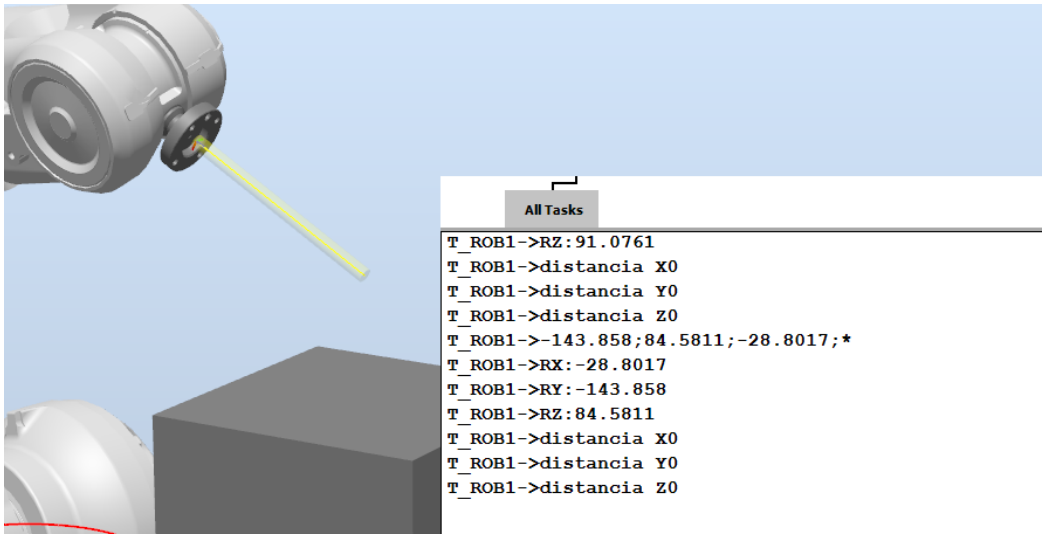

Figura 46. Valor de las variables distancia X, distancia Y, distancia Z cuando el robot se mueve libremente

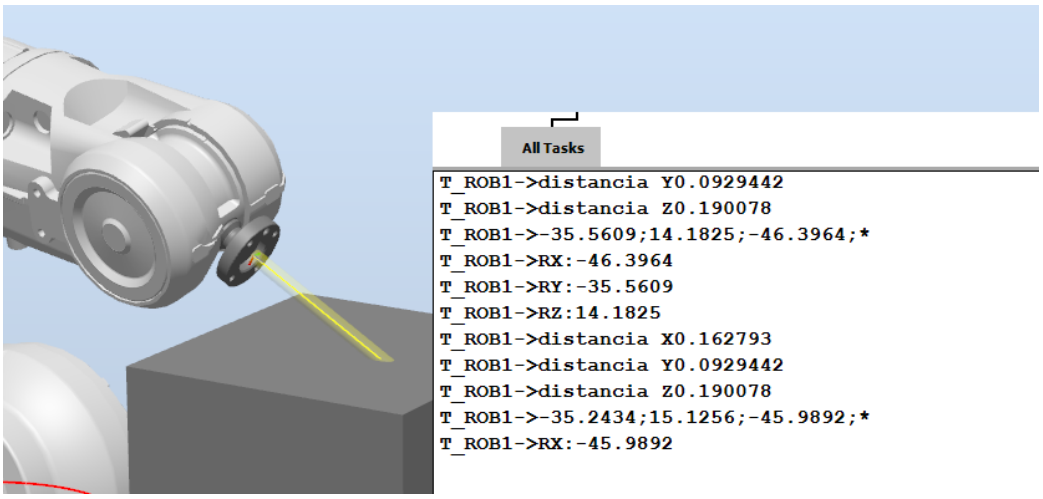

Figura 47. Valor de las variables distancia X, distancia Y, distancia Z cuando el sensor detecta un objeto

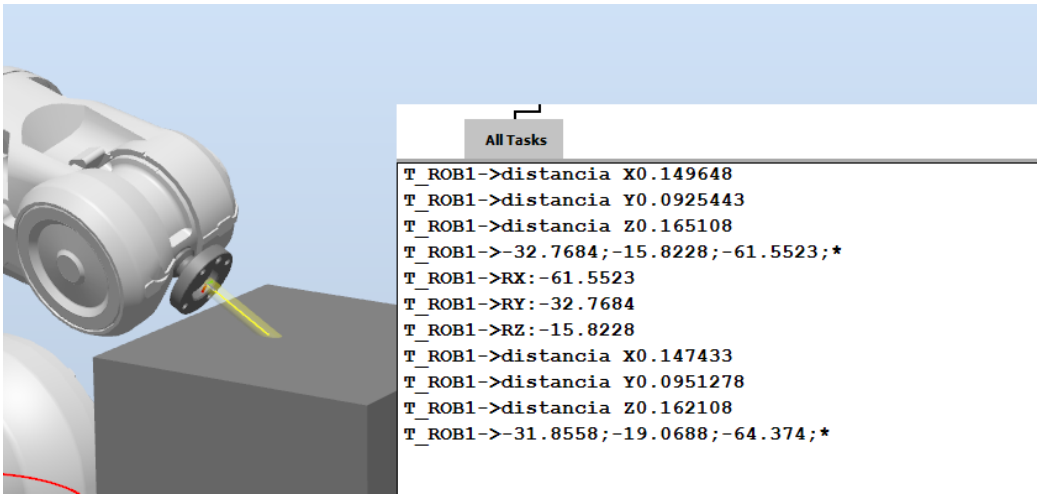

Figura 48. Valor de las variables distancia X, distancia Y, distancia Z cuando el sensor detecta un objeto

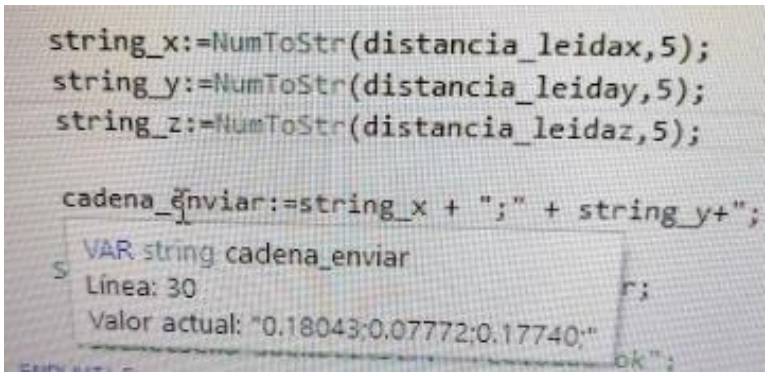

Figura 49. Valor que contiene las coordenadas enviado en forma de cadena de caracteres

Así pues, una vez comprobado el correcto funcionamiento del sensor, el siguiente paso a lograr es definir un conjunto de fuerzas con el fin de evitar que el robot continue en la dirección del objeto. Se trata de conseguir reproducir unas fuerzas en el efector del dispositivo háptico que el usuario pueda sentir y que impidan, por tanto, que este pueda seguir desplazando el stylus en la dirección en la que el robot encuentra un obstáculo.

Como bien se ha comentado en el apartado anterior correspondiente al debate entre que tipo de librería de OpenHaptics utilizar, en este caso se ha optado por la utilización de HDAPI, ya que permite implementar un modelo matemático relativamente sencillo con el que representar las fuerzas necesarias.

Por tanto, vamos a simular una fuerza dependiente del movimiento. Una fuerza que depende del movimiento significa que se calcula en función del movimiento del dispositivo háptico. Existen varios tipos de fuerzas que dependen del movimiento que pueden ser simuladas como bien puede ser las fuerzas de fricción, de amortiguación o de muelle, que es el tipo de fuerza que se ha utilizado en la realización de esta aplicación. La fuerza de un muelle puede calcularse aplicando la ley de Hooke (F = kx, donde k es una constante de rigidez y x es un vector de desplazamiento). El vector de desplazamiento x = - es tal que la fuerza del muelle se dirige siempre hacia la posición de anclaje fija. La fuerza que se siente se denomina fuerza de restauración del muelle, ya que el muelle trata de restablecerse a su longitud de reposo, que en este caso es cero. La constante de rigidez k dicta la

agresividad con la que el muelle intentará volver a su longitud de reposo. Una constante de rigidez baja se sentirá floja, mientras que una constante de rigidez alta se sentirá rígida.

Centrándonos en el funcionamiento de nuestra aplicación, lo que se consigue es mandar la distancia leída por el sensor desde RAPID hasta el programa del dispositivo háptico gracias a la conexión por sockets. Una vez recibida dicha distancia (al igual que antes se manda una cadena de caracteres que ha de separarse y almacenarse en variables numéricas), simplemente se hace una llamada sincronizada de manera que se produce una fuerza como la siguiente en las tres direcciones que permite el 3D Systems Touch:

```
// Hooke's law explicitly:
double k = planeStiffness;
hduVector3Dd x = penetrationDistance * forceDirection;
hduVector3Dd f = k * x;

hdSetDoublev(HD_CURRENT_FORCE, f);
```

Figura 50. Ley de fuerzas en una dirección

Vemos como se genera una mayor fuerza a medida que se intenta penetrar en la dirección del objeto, evitando así que llegue a impactar el robot con el objeto. En las siguientes imágenes puede advertirse como, en el caso de la primera, intento ejercer fuerza hacia abajo en dirección del objeto, respondiendo el háptico generando una fuerza; en el caso de la segunda puede observarse como dejando el stylus en el aire, este no cae debido a la fuerza generada.

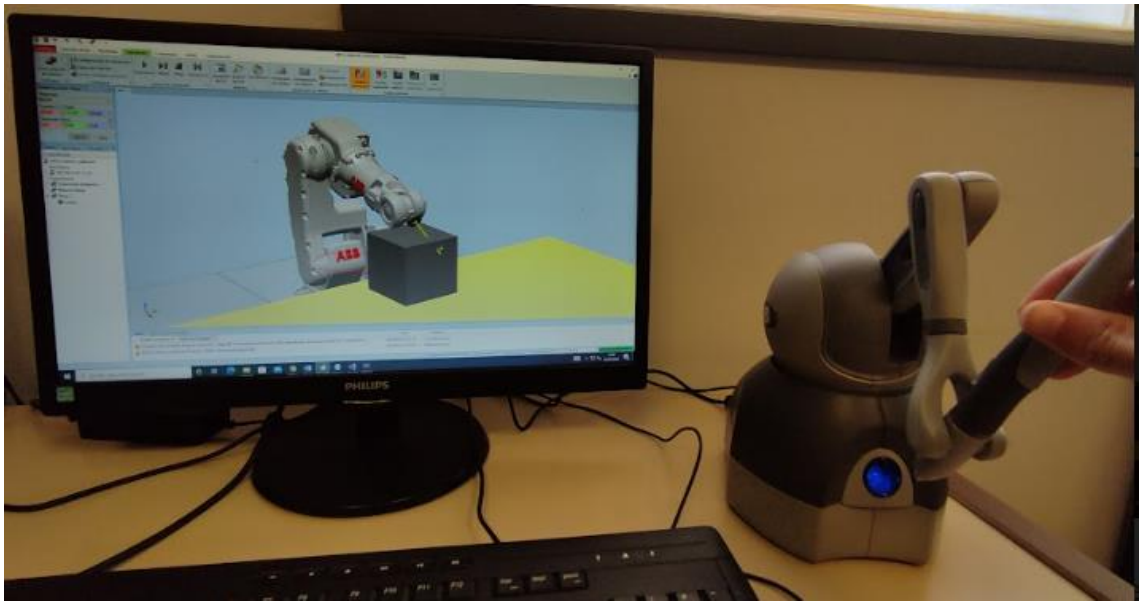

Figura 51. Comprobación de la fuerza generada por el dispositivo hático 3D Systems Touch

Figura 52. Comprobación de la fuerza generada por el dispositivo hático 3D Systems Touch

4.5 Aplicaciones del trabajado desarrollado

Esta aplicación cobra sentido en caso de querer implementarla en tareas, por ejemplo, de pick & place, en las que se desea que el robot realice su función sin que se produzca ningún accidente fatal durante la manipulación de piezas. De esta manera se conseguiría que el trabajador pudiera tener un control remoto total sobre el brazo robótico y que en caso de existencia de algún objeto extraviado o simplemente por un despiste del propio trabajador, el robot no resultase colisionando contra dicho objeto o contra cualquier elemento que se encuentre dentro de su zona de trabajo, como bien pueden ser cintas transportadoras. Otra posible aplicación de este modelo desarrollado sería en casos de asistencia remota en los que se debe trabajar minuciosamente en el montaje de chips a pequeña escala o en la asistencia de pacientes y se busca poder trabajar con la tranquilidad de saber que ante objetos inadvertidos o despistes puntuales, se cuenta con la protección de que el robot no llegará a colisionar por impedir el movimiento del dispositivo háptico.

Siguiendo en la línea de lo anteriormente comentado, podría adaptarse esta aplicación para generar un campo de fuerza alrededor de un objeto sobre el que se está trabajando. Imaginemos un escenario en el que se esté trabajando en una operación de acabado de materiales que requiera de la precisión de un ser humano experimentado. Con el fin de evitar accidentes inoportunos, puede generarse una fuerza de un amortiguador cuya principal utilidad es reducir la vibración y que es proporcional a la velocidad del efector final. La ecuación estándar del amortiguador es F = -bv, donde b es la constante de amortiguación y v es la velocidad del efector final. La fuerza siempre apunta en la dirección opuesta del movimiento. De esta manera conseguiría evitarse que en las aproximaciones del objeto la velocidad con la que se incide sea demasiado elevada, generando una fuerza opuesta al movimiento y evitando así errores fatales.

5. CONCLUSIONES

Tras la realización del presente trabajo y observando los objetivos planteados al inicio del mismo, puede concluirse que el proyecto desarrollado cumple con las expectativas.

En un primer lugar, se ha llevado a cabo una búsqueda exhaustiva de información acerca de los dispositivos hápticos, tanto por lo que respecta a su ámbito de uso y aplicaciones, como en cuanto a aspectos de funcionamiento de estos, incluyendo también varios ejemplos de dispositivos a parte del utilizado.

Siguiendo en la línea de lo mencionado anteriormente, el análisis de los distintos recursos incluidos en el OpenHaptics Toolkit permitió conocer a fondo las distintas posibilidades de utilización de los distintos recursos que ofrece, concluyéndose que la mejor opción en el caso de nuestra aplicación era la utilización de la API de bajo nivel HDAPI.

En cuanto a la de comunicación entre dispositivo háptico y robot, se ha logrado desarrollar un control remoto en tiempo real del robot utilizando para ello el lápiz de nuestro háptico, gracias a la correcta comunicación mediante sockets establecida.

Así pues, una vez lograda la telemanipulación del robot, se optó por generar una célula robótica en RobotStudio con la que se detectaran objetos indeseados dentro del espacio de trabajo del robot. De esta manera, se consiguió desarrollar una aplicación mediante la cual, si se detectaba un objeto en la trayectoria del robot, el háptico representaba una serie de fuerzas en función de la distancia del efector del robot al objeto en cuestión con la que poder evitar impactos o colisiones fatales.

Finalmente, se comentaron posibles aplicaciones industriales de la célula robótica desarrollada.

5.1 Líneas futuras

El presente trabajo se plantea como una primera aproximación a la implementación de los dispositivos hápticos en aplicaciones de control remoto de robots industriales. En consecuencia, la célula desarrollada podría servir como un primer paso en el desarrollo de aplicaciones más refinadas que puedan abarcar un gran espectro de posibilidades. A continuación, se describen algunas futuras líneas de investigación relacionadas con el proyecto llevado a cabo:

- Respecto al sensor del que dispone la célula robótica, este solo es efectivo si el objeto indeseado se encuentra en la trayectoria del efector del robot, con lo que una posible mejora radica en el hecho de poder disponer de sensores de posición alrededor de todo el robot industrial, o bien de cámaras de visión artificial con las que detectar objetos inapropiados dentro de su workspace.
- En cuanto a la comunicación con el robot, se tiene una comunicación por sockets con una frecuencia inferior a los 1000 Hz, por lo que si se quisiera utilizar una realimentación háptica basada en los contactos ejercidos por el robot, se debería de aumentar la velocidad de comunicación enormemente.

- Lograr un control total del movimiento articular, es decir, dado que ambos sistemas consisten en brazos articulados con 6 grados de libertad, sería posible desarrollar una aplicación que generara fuerzas a través de una interfaz háptica con el fin de identificar los ángulos que forman las articulaciones del robot. Esto permitiría el control posicional de la orientación y posición del robot TCP.
- Implementar la célula robótica desarrollada en aplicaciones remotas en las que el espacio de trabajo del robot se encuentre sujeto a cambios constantes debido a situaciones adversas, como bien puede ser la teleoperación de robots en operaciones subacuáticas.

Bibliografía

[1] 3DSystems. (n.d.). *OpenHaptics Toolkit Version 3.5.0 API Reference Guide.*

[2] 3DSystems. (n.d.). *OpenHaptics Toolkit Version 3.5.0 Programmer's Guide.*

[3] 3DSystems. (n.d.). *Touch.* Retrieved from https://es.3dsystems.com/haptics-devices/touch

[4] 3DSystems. (n.d.). *Touch Specifications.* Retrieved from https://es.3dsystems.com/haptics-devices/touch/specifications

[5] ABB. (n.d.). *RobotStudio.* Retrieved from https://new.abb.com/products/robotics/es/robotstudio

[6] AIStanford. (n.d.). *Novint Falcon.* Retrieved from http://ai.stanford.edu/~conti/falcon.html

[7] CyberGolve. (n.d.). *CyberGrasp.* Retrieved from http://www.cyberglovesystems.com/cybergrasp

[8] Forcedimension. (n.d.). *Sigma.7.* Retrieved from https://www.forcedimension.com/products/sigma

[9] García, M. (n.d.). *Dispositivos hápticos.*

[10] Industry, D. (n.d.). Retrieved from https://www.directindustry.es/prod/abb-robotics/product-30265-565894.html

[11] Jake J. Abbott, G. D. (2003, Octubre 31). Steady-Hand Teleoperation with Virtual Fixtures. 7.

[12] Jorge Gudiño Lau, F. C. (2018, Diciembre). Direct and inverse kinematic model of the OMNI PHANToM.

[13] Leah Piggott, S. W. (2016). Haptic Neurorehabilitation and Virtual Reality for Upper Limb Paralysis: A Review.

[14] Martin. (n.d.). Retrieved from Puertos y Sockets: http://personales.upv.es/rmartin/TcpIp/cap02s10.html

[15] Oscar Dénis, A. Q. (n.d.). *Comunicación mediante sockets.* Retrieved from http://sopa.dis.ulpgc.es/progsis/material-didactico-teorico/tema7_1transporpagina.pdf

[16] P. Chotiprayanakul, D. L. (2009, Junio). Workspace Mapping and Force Control for Small Haptic Device.

[17] Rafael Aracil, J. B. (2004, Septiembre). Identificación y modelado de un sistema maestro-esclavo para teleoperación.

[18] *Robotnik*. (2022, Marzo 14). Retrieved from https://robotnik.eu/es/que-es-un-robot-industrial-definicion-y-caracteristicas/

[19] Soares Falcao, M. S. (2014). Applications of Haptic Devices & Virtual Reality in Consumer Products Usability Evaluation.

[20] Takehiro Miki, T. I. (2016). Development of a virtual reality training system for endoscope-assisted submandibular gland removal.

[21] TechBriefs. (2010, Agosto 1). *Controlling Robotics Precisely With Haptic Technology*. Retrieved from https://www.techbriefs.com/component/content/article/tb/supplements/mct/features/articles/9278#:~:text=Haptic%20devices%20are%20input%2Doutput,drives%20a%20robotic%20motion%20system.

[22] UniversalRobots. (2020, November 10). *Cobot Applications.* Retrieved from https://www.universal-robots.com/blog/cobot-applications/

[23] UniversalRobots. (2020, Noviembre 10). *Universal Robots.* Retrieved from https://www.universal-robots.com/blog/cobot-applications/

[24] UniversalRobots. (n.d.). *Universal Robots UR3e.* Retrieved from https://www.universal-robots.com/es/productos/robot-ur3/

[25] Vera, R. (2021, Diciembre 10). *Interfaces hápticas y el desafío de diseñarlas después de la pandemia.* Retrieved from https://blog.ida.cl/diseno/interfaces-hapticas-y-el-desafio-de-disenarlas-despues-de-la-pandemia/

PRESUPUESTO

ÍNDICE DEL PRESUPUESTO

1. DESCRIPCIÓN

En este capítulo se estudia el presupuesto del proyecto para determinar su viabilidad económica mediante el cálculo del coste que requeriría dicho proyecto. Para determinar la viabilidad, se ha presupuestado la mano de obra y la maquinaria utilizados durante la realización de este trabajo, obteniéndose el cuadro de precios parcial para cada uno de ellos.

A partir de estos recursos, se ha determinado el presupuesto de ejecución material, el presupuesto de ejecución por contrata y, finalmente, el presupuesto general del proyecto o también conocido como presupuesto base de licitación

A continuación, se muestra el código con el que se identifica a los recursos del presupuesto.

Código	Descripción del código
MO	Mano de Obra
MQ	Maquinaria

Tabla 1. Código de identificación de los recursos empleados en la realización del presupuesto

2. CUADRO DE PRECIOS DE MANO DE OBRA

En este apartado se describen los diferentes precios de mano de obra en la realización de este TFG. En la cantidad de horas empleadas, se han considerado tanto las horas de estancia en el laboratorio como las dedicadas a recopilación de información y planificación de las sesiones. Se ha considerado como mano de obra la del autor del TFG, así como la del tutor del mismo.

Código	Descripción	Importe		
		Coste (€/h)	Cantidad (h)	Precio (€)
MO.1	Graduado en Ingeniería en Tecnologías Industriales	13.50	200	2700
MO.2	Director Responsable del Proyecto	37.70	30	1131
Precio Total Mano de Obra				3831

Tabla 2. Cuadro de precios del presupuesto parcial de mano de obra

3. CUADRO DE PRECIOS MAQUINARIA

Para los equipos utilizados se ha calculado la amortización según la Ecuación 13, considerando un periodo de amortización según se indica en la Tabla 3.

$$C_A = \frac{C_{eq} \cdot t_{uso}}{Per_{amort}} \tag{13}$$

Siendo las variables de la ecuación:

C_{eq}: Coste del equipo adquirido

t_{uso}: Tiempo de uso

Per_{amort}: Periodo de amortización

Código	Descripción	Importe			
		Coste (€)	Amortización (años)	Tiempo (mes)	Precio (€)
MQ.1	Dispositivo háptico 3D Systems Tocuh	2240	10	2	37.33
MQ.2	Ordenador	500	6	3	20.83
MQ.3	Periféricos	150	6	3	6.25
MQ.4	Software Visual Studio Enterprise	500	6	2	13.89
MQ.5	Robot ABB IRB 140	8000	12	1	55.55
Precio Total Mano Maquinaria					133.85

Tabla 3. Cuadro de precios del presupuesto parcial de material Inventariable

4. PRESUPUESTO DE EJECUCIÓN MATERIAL, POR CONTRATA Y BASE LICITACIÓN

En primer lugar, se obtiene el presupuesto de ejecución material como la suma de los presupuestos parciales de mano de obra y maquinaria en los que es necesario considerar unos costes indirectos del 25% del total de costes directos.

El presupuesto de ejecución por contrata resulta de la suma del presupuesto de ejecución material y los gastos generales aplicables a este presupuesto (13% del presupuesto de ejecución material).

Por último, para obtener el presupuesto base de licitación se debe aplicar el porcentaje de IVA (21%) sobre este último presupuesto.

Presupuesto de Ejecución Material, por Contrata y Base de Licitación

1. Presupuesto Parcial Mano de Obra	3831.00 €
2. Presupuesto Parcial Maquinaria	133.85 €
Total Parcial	3964.85 €
Costes Indirectos (25%)	991.21 €
Total Ejecución Material	**4956.06 €**

El presupuesto de Ejecución Material asciende a la expresada cantidad de **CUATRO MIL NOVECIENTOS CINCUENTA Y SEIS EUROS CON SEIS CÉNTIMOS**

Presupuesto de Ejecución Material	4956.06 €
Gastos GENERALES (13%)	644.29 €
Beneficio Industrial (0%)	0 €
Total Ejecución por Contrata	**5600.35 €**

El presupuesto de Ejecución por Contrata asciende a la expresada cantidad de **CINCO MIL SEISCIENTOS EUROS CON TREINTA Y CINCO CÉNTIMOS**

Presupuesto de Ejecución por Contrata	5600.35 €
IVA (21%)	1176.07 €
Total Base de Licitación	**6776.42 €**

El presupuesto Base de Licitación asciende a la expresada cantidad de **SEIS MIL SETECIENTOS SETENTA Y SEIS EUROS CON CUARENTA Y DOS CÉNTIMOS**